Mobile Robots II

- Begleitheft
- Activity booklet
- Manuel d'accompagnement
- Cuaderno adjunto
- Folheto

fischertechnik®

(D) INHALT

D

1 Wozu brauchen wir Roboter ?

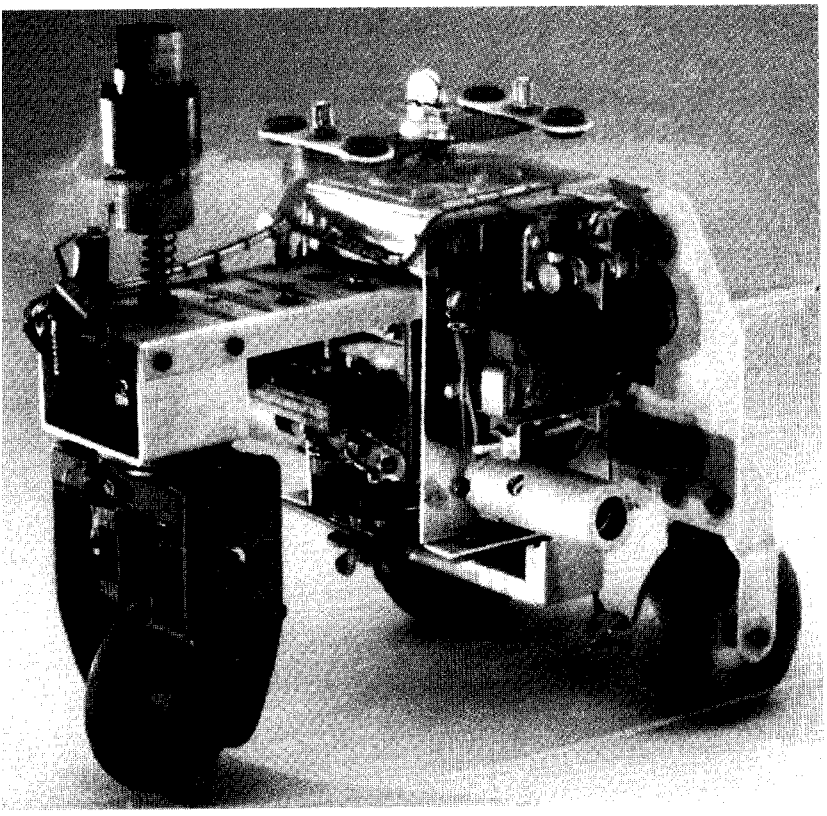
Bevor wir beginnen uns praktisch mit Robotertechnik zu befassen, wollen wir versuchen, die leicht provokativ gestellte Frage in der Überschrift zu beantworten.

Der Begriff des „Roboters" wird erstmals 1923 im Roman „Golem" von Carel Capek verwendet. Diese düstere künstlich erschaffene Figur sollte mit ihren Fähigkeiten menschliche Arbeit ersetzen.

Wie so oft bei literarischen Gestalten ist dies auch hier an Zwänge gebunden und ein gewisses Mißtrauen schlägt der Figur entgegen. In den 30er und 40er Jahren des 20. Jahrhunderts wird aus dem Roboter eher eine Art Automat. Diverse Versuche, ihn mit äußeren menschlichen Eigenschaften zu versehen, z. B. einem Kopf mit blinkenden Lampen als Augen sowie primitiver Sprachausgabe per Lautsprecher, wirken aus heutiger Sicht naiv. Offenbar sind die Befürchtungen einer potentiellen Herrschaft der Roboter über die Menschen nicht so einfach zu widerlegen.

Doch bei diesen ersten einfachen Versuchen ist von einer Mobilität oder gar Intelligenz der konstruierten Maschinen wenig zu bemerken. Erst mit dem Aufkommen elektronischer Schaltungen wurde der Aufbau von Robotern realistisch.

Eng mit der eigentlichen Robotertechnik verknüpft, ist das Problem nach den erforderlichen Steuerprinzipien. Diese 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 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). Unser Gedankenexperiment kombiniert nun einen optischen Sensor mit einem Motor und einer Logik, so dass dieses Fahrzeug den Motor immer in Richtung der Lichtquelle steuert. Dieses Fahrzeug würde sich demnach genau wie eine Motte verhalten, oder nicht?

Eine technische Realisierung des eben geschilderten Experimentes erfolgte in den 50er Jahren durch den Briten Walter Grey. 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.

Diese Maschinen stellen einen wichtigen Schritt auf dem Weg zu modernen mobilen Robotern dar. Die Sensoren (Fotowiderstand, Fühler,...) der Geräte steuerten mit Hilfe ihrer Elektronik die Aktoren (Motoren, Relais, Lampen,...) so dass ein (scheinbar?) intelligentes Verhalten zustande kam. In der Abbildung ist ein Nachbau der „kybernetischen" Schildkröte, welche im Smithsonian Museum, Washington ausgestellt ist, zu sehen.

Basierend auf diesen Überlegungen erstellen wir für unsere Roboter entsprechende „Verhaltensmuster" und versuchen diese in Form von Programmen dem Roboter verständlich zu machen.

Doch wie ist mit diesen Überlegungen die eingangs gestellte Frage nach dem Nutzen von mobilen Robotern zu beantworten? Um diese Frage konkret zu beantworten, versuchen wir nun, die bislang eher abstrakten Verhalten unserer „Gedankenmotte" auf technische Belange 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 von Strichkode 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 bereitstehende Zeit ein.

Wir erkennen also, dass mobile Roboter einen wichtigen Platz in der modernen Gesellschaft einnehmen können. Doch wie hängt dies mit den fischertechnik-Baukästen zusammen?

Für einen Roboter brauchen wir außer den Sensoren und Aktoren viele mechanische Teile um ein Modell zu konstruieren. Der fischertechnik-Baukasten Mobile Robots II ist hierzu eine ideale Grundlage. Wir können die Mechanik-

teile in einer beinahe unerschöpflichen Vielfalt miteinander kombinieren und erhalten robuste Roboterfahrzeuge. Mit dem dazugehörigen „Intelligent Interface" (Art.-Nr. 30402, zusätzlich erforderlich) verfügen wir auch über genug Rechenpower um anspruchsvolle Programme zu entwerfen. Über dieses Interface erfolgt die Ankopplung und Auswertung einer Vielzahl verschiedener Sensoren und Aktoren.
Die Sensoren wandeln physikalische Messgrößen, wie Lichtmenge oder Temperatur, in elektrisch erfassbare Werte um. Dabei gibt es sowohl analoge auch digitale Messgrößen. Unter digitalen Größen verstehen wir solche, die entweder logisch wahr oder logisch falsch sein können. Diese Zustände werden mit 0 bzw. 1 gekennzeichnet. Ein Schalter ist ein gutes Beispiel für einen digitalen Sensor.

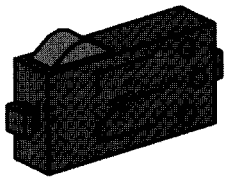
Viele Messgrößen ändern sich jedoch kontinuierlich zwischen ihren Extremwerten, diese nennt man analoge Werte. Sie können nicht einfach als 0 oder 1 dargestellt werden. Damit diese Größen von einem Computer verarbeitet werden können, müssen sie in entsprechende Zahlenwerte umgewandelt werden. Das fischertechnik-Interface stellt hierzu zwei Analogeingänge EX und EY bereit. Der an diesen Klemmen angelegte Widerstandswert wird in einem Zahlenwert gespeichert. Die Messwerte eines Temperatursensors, z. B. 0...5 kΩ, werden somit in einem Bereich von 0...1024 erfasst und stehen für eine nachfolgende Bearbeitung zur Verfügung.
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.
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 hohem Niveau unsere Programme zu erstellen. Wir können nämlich mit Hilfe grafischer Symbole Programme bzw. Algorithmen entwerfen. Der Computer des Intelligent Interface kann eigentlich nur Befehle aus seinem sogenannten Maschinenbefehlssatz ausführen. Das sind im Wesentlichen einfache logische bzw. arithmetische Kontrollstrukturen, deren Anwendungen für Einsteiger außerordentlich schwierig sind. Die PC-Software LLWin (Art.-Nr. 30407, nicht im Baukasten enthalten) stellt deswegen Grafikelemente bereit, die anschließend in eine für das Interface ausführbare Sprache übersetzt werden.

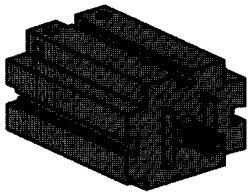
Wir werden bei unserem Eindringen in die faszinierende Welt der mobilen Roboter schrittweise vorgehen. Wir beginnen mit einem einfachen Testaufbau zur Prüfung der Grundfunktionen von Interface und Sensorik. Danach starten wir mit einfachen Modellen denen bestimmte Aufgaben zugeordnet sind und versuchen uns dann mit immer komplizierteren Systemen. Damit auftretende Fehler nicht zu permanentem Verdruss führen, werden wir uns in einem Kapitel mit Eigenschaften und Besonderheiten von Sensoren und Aktoren vertraut machen und für ganz „hartnäckige" Fälle gibt es am Ende einen Abschnitt „Fehlersuche und Behebung".

Ein sehr wichtiger Punkt ist die Sorgfalt beim Aufbau und der Inbetriebnahme unserer Roboter. Wir haben es mit komplexen Maschinen zu tun, deren einziger Unterschied zu wirklichen Robotersystemen ihre vergleichsweise geringe Größe ist. Beim Aufbau 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 Spielarmut in den Getrieben und Befestigungen. Nirgendwo müssen wir beim Zusammenbau „Gewalt" anwenden. Es ist unserer Kreativität vorbehalten, neue Programme zu schreiben und damit neue „Verhalten" zu definieren, deren Komplexität nur durch die zur Verfügung stehenden Ressourcen an Speicherkapazität und Rechenleistung begrenzt ist. Die folgenden Beispiele geben dazu einige Anregungen.

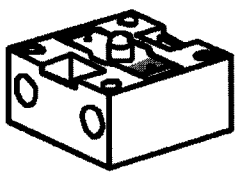
2 Erste Schritte

Nach den theoretischen Überlegungen wollen wir nun endlich beginnen eigene Experimente durchzuführen. Sicherlich möchte der eine oder andere sofort starten, vielleicht sogar mit dem komplizierten automatischen Gabelstapler. Das ist natürlich möglich und bei sorgfältiger Beachtung der Bauanleitung gelingt der Aufbau des Modells auf Anhieb.
Doch was tun, wenn es nicht funktioniert? In diesen Fällen muß systematisch nach der Fehlerursache gesucht werden. Ganz zwangsläufig tauchen dabei Fragen zur Funktionsweise und den Eigenschaften der verwendeten Komponenten auf. Offenbar ist ein gewisses Maß an Grundlagenwissen zu Sensoren und Aktoren unerlässlich. Bevor wir jedoch beginnen, uns mit diesen Dingen vertraut zu machen, prüfen wir das Zusammenspiel von Computer und Interface.
Entsprechend den Vorgaben aus dem LLWin-Handbuch wird die Steuersoftware auf dem PC installiert.
Mit Hilfe der Interface-Diagnose testen wir die unterschiedlichen Sensoren bzw. Aktoren. Wir können jetzt z. B. einen Taster mit zwei Leitungen an den Eingang E1 anschließen und sehen dann, welcher logische Schaltzustand vom Interface erkannt wird. Eine Betätigung des Tasters muss zu einer Zustandsänderung am entsprechenden Eingang führen.



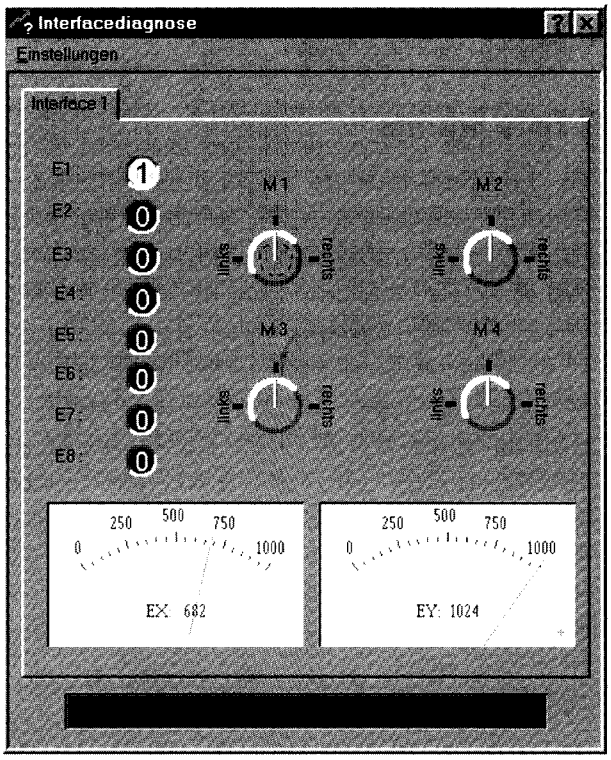


Die Ausgänge überprüfen wir, indem wir einen Motor mit einem Motorausgang, z. B. M1, verbinden. Mit der Maus gelingt es uns, den Motor in Drehung zu versetzen. Wollen wir auch den Analogeingang testen, so funktioniert dies, indem wir einen Fototransistor als Analogsensor verwenden. Während beim Motor 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.



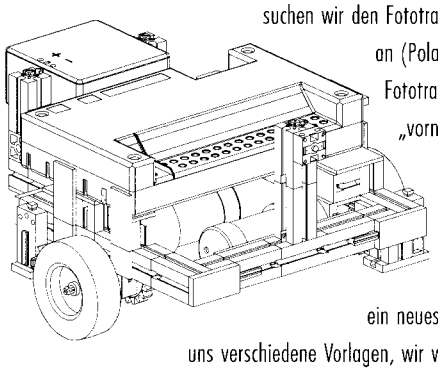
Ein Kontakt des Transistors ist mit einer roten Markierung versehen, diesen verbinden wir mit einem roten Anschlußstecker, den nicht gekennzeichneten Kontakt mit einem grünen Stecker.
Der zweite rote Stecker kommt in die näher am Rand des Interfaces liegende Buchse des Eingangs EX, der zweite grüne Stecker passt in die weiter innen liegende Buchse von EX. Nun können wir mit Hilfe einer Taschenlampe die Beleuchtungsstärke des Fototransistors variieren und damit den Zeigerausschlag 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 Zeigerausschlag ändert sich dann, wenn wir den Fototransistor abdecken.

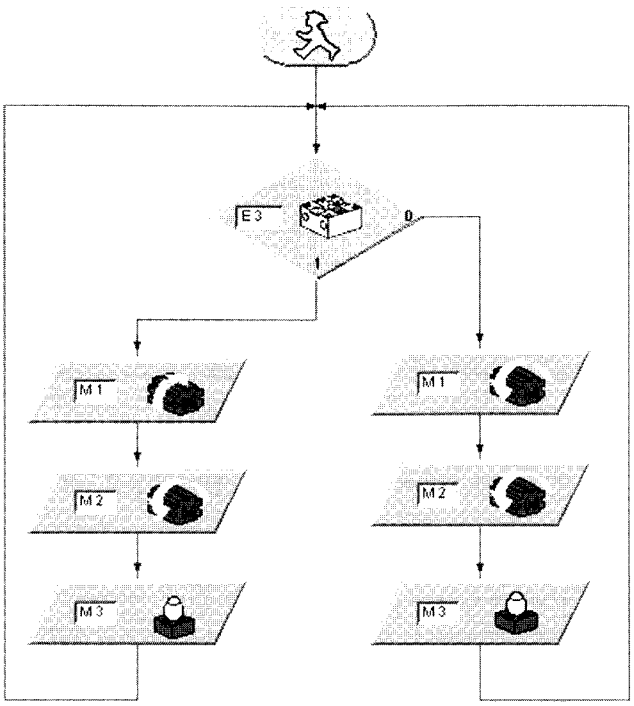


Um nochmals kurz auf die Farbzuordnung der Stecker zu kommen: Wir achten streng darauf beim Zusammenbau immer einen roten Stecker an die rote Leitung anzuschließen und einen grünen Stecker mit der grünen Leitung zu verbinden. Wenn wir in einem Schaltungsaufbau polarisierte Signale verwenden, nehmen wir immer eine rote Leitung für den Pluspol und eine grüne Leitung für den Minuspol. Das mag ein wenig pedantisch erscheinen (und es ist dem Strom auch egal welche Farbe eine Leitung hat), 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. Wir bauen das Basismodell mit den beiden Antriebsmotoren und dem Stützrad auf entsprechend der Bauanleitung auf. Wir schließen nur die Motoren an die Ausgänge M1 und M2 an. Außerdem suchen wir den Fototransistor und schließen ihn an den Eingang E3 an (Polarität beachten). Zuvor befestigen wir den Fototransistor so am Grundmodell, dass er nach „vorn" schaut. Wer will, kann noch die Linsenlampe an M3 anschließen und sie z. B. neben dem Akkupack befestigen, aber das ist nicht unbedingt notwendig. Wir öffnen das Programm LLWin und legen ein neues Projekt an (PROJEKT – NEU). LLWin bietet uns verschiedene Vorlagen, wir wählen „leeres Projekt" und geben ihm einen Namen, z. B. „Step1". Nachdem wir den [OK]-Button gedrückt haben, erscheint ein leeres Arbeitsblatt mit einem Ampelmännchen und dem Bausteinfenster. Das grüne Ampelmännchen symbolisiert den Programmstart.



Von diesem Startpunkt aus beginnen alle unsere Programme. Die verschiedenen Programmteile holen wir mit der Maus aus dem Bausteinfenster. Die dort vorhandenen Symbole stehen für die Ein-/Ausgänge am Interface. Mit der linken Maustaste können wir das gewünschte Symbol platzieren und mit der rechten Maustaste ändern wir die Eigenschaften.



Das Tastersymbol kennzeichnet einen Eingang. Für unser Programm platzieren wir mit der Maus den Taster unter das Startsymbol. Wenn wir das Symbol loslassen, erscheint ein Auswahldialog. Wir wählen den Fototransistor aus. Wenn wir später weitere Änderungen wünschen, können wir diesen Dialog mit der rechten Maustaste aktivieren. Bei den Motoren ordnen wir die Ausgänge zu und weisen die gewünschte Drehrichtung an. Wir wollen, dass die Motoren gleichsinnig drehen, wenn kein Licht auf den Fototransistor fällt und gegenläufig, wenn Licht erkannt wird. Dann werden die Elemente mit der Zeichenfunktion verbunden. Die Lampe an M3 signalisiert den Zustand des Fototransistors.

Die Abbildung zeigt die genaue Verbindung der Programmzweige. Wer sich nicht sicher ist, ob alles richtig ist, vergleicht sein Programm mit dem Programm Step1.mdl. Dazu wird das eigene Programm vorher gespeichert und die Datei Step1.mdl von der CD-ROM geladen, die im Baukasten enthalten ist. Ist alles in Ordnung, wird das Programm per Download in das Interface geladen und gleich gestartet (RUN - DOWNLOAD).

Unser erster Roboter dreht sich jetzt auf der Stelle. Er tut dies solange, bis wir ihn mit einer Lichtquelle anlocken. Sobald der Fototransistor das Licht erkennt, fährt der Roboter geradeaus auf die Lichtquelle zu. Falls er sich von der Lichtquelle entfernt, müssen wir beide Motoren umpolen. Wahrscheinlich wird seine Bahn nicht exakt geradeaus zeigen, so dass nach einiger Wegstrecke der Fototransistor den Kontakt zur Lichtquelle verliert. Daraufhin wird seine Bewegung von Fahrt voraus auf Drehung umgeschaltet und die Lichtsuche beginnt erneut. Damit wir erfolgreich experimentieren, haben wir vorher entsprechend Platz geschaffen, denn leider kann unser Roboter Hindernisse auf seinem Weg (noch) nicht wahrnehmen.

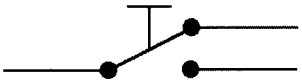
3 Sensoren und Aktoren

Wir wissen nun, dass unsere wichtigste Baugruppe, das Interface, zusammen mit dem PC „spielt". Jetzt geht es darum, wie die Signale aus der Umwelt von unserem Interface erkannt werden können.

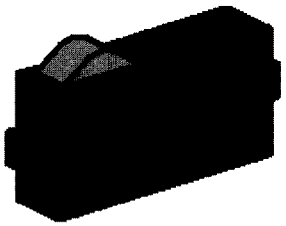
Beginnen wir mit den Eingängen. In der Fachsprache werden Eingänge bzw. Eingangssignale oft als Input bezeichnet. Für einen Computer, und um einen solchen handelt es sich beim Intelligent Interface, gibt es nur die Möglichkeit elektrische Signale zu erkennen und zu verarbeiten. Wir müssen die Umweltreize also „computertauglich" machen. Alle Sensoren sind deshalb Umsetzer, die den gewünschten „Sinn" in ein elektrisches Signal wandeln. Da wir nicht ausschließlich Bauanleitungen „blind" nachbauen wollen, ist es sinnvoll, sich mit den grundlegenden Eigenschaften der vorhandenen Sensoren zu befassen.

Dies ist um so wichtiger, da das Interface später durchaus um neue, selbst definierte Anwendungen ergänzt werden kann.

3.1 Der Schalter als digitaler Sensor

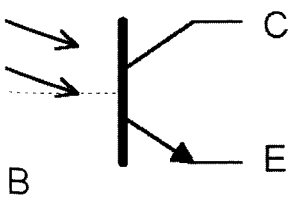


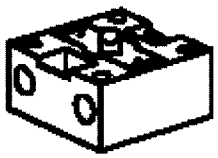
Die einfachen logischen Pegel „0" und „1" sind mit Hilfe eines Schalters darstellbar. Im System der fischertechnik-Baukästen werden Präzisionssprungschalter mit Wechselkontakten verwendet.



Die Besonderheit der Sprungschalter liegt in ihrem Schaltverhalten. Bei vorsichtigem langsamen Betätigen des roten Schaltknopfs spüren wir einen deutlichen Druckpunkt, bei dessen Überschreiten der Schaltkontakt mit einem leisen Knackgeräusch umschaltet. Lassen wir den Schalthebel nun wieder langsam los, müssen wir den Hebel deutlich weiter über den ursprünglichen Einschaltpunkt „herauslassen", um ein Zurückschalten zu erreichen. Diesen Unterschied zwischen mechanischer Einschalt- und Ausschaltposition wird als Hysterese bezeichnet. Die Schalthysterese von Kontakten oder anderen elektronischen Schaltungen ist eine wichtige Eigenschaft. Gäbe es sie nicht, d.h. Einschaltpunkt wäre gleich Ausschaltpunkt, wären große Probleme in der Signalauswertung die Folge. Winzige Störungen, wie ein ganz leichtes Zittern im Umschaltzeitpunkt, würde dem Interface mehrere Kontaktbetätigungen „vorgaukeln", ein exaktes Zählen von Ereignissen wäre nicht möglich. Der Schalter ist als Wechselschalter ausgeführt. Wir können somit beide denkbaren Ausgangslagen, d.h. im Ruhezustand geschlossen bzw. im Ruhezustand geöffnet, bei unseren Experimenten auswerten.

3.2 Lichterkennung mit dem Fototransistor





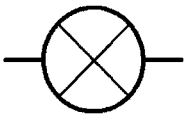
Beim Fototransistor handelt es sich um ein Halbleiterbauelement, dessen elektrische Eigenschaften lichtstärkeabhängig sind. Ein gewöhnlicher Transistor ist ein Bauelement mit drei Anschlüssen. Diese Anschlüsse werden als Emitter, Basis und Collektor bezeichnet. Seine Hauptaufgabe ist die Verstärkung schwacher Signale. Ein schwacher Strom, der von einem Signal in die Basis des Transistors fließt, hat einen viel stärkeren Strom am Collektor des Transistors zur Folge. Die Stromverstärkung kann Faktoren von über 1000 erreichen. Der Fototransistor aus dem Baukasten besitzt jedoch nur zwei Anschlüsse. Wo ist der dritte Anschluß geblieben?

Wir wollen mit unserem Transistor Licht erkennen. Jeder kennt Solarzellen, mit denen aus Sonnenlicht Strom gewonnen wird. Der Fototransistor ist als Kombination von Minisolarzelle und Transistor zu verstehen. Der Basisanschluss wird nicht nach außen geführt (deshalb in der Abbildung gestrichelt). An seiner Stelle erzeugen auftreffende Lichtimpule (Photonen) einen sehr kleinen Fotostrom, der dann vom Transistor verstärkt am Collektor zur Auswertung bereitsteht. Damit dies so wie beschrieben funktioniert, benötigt der Fototransistor eine zusätzliche Außenbeschaltung. Da diese im Interface enthalten ist, braucht sie uns nicht weiter zu interessieren.

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.

3.3 Signalausgabe mit der Glühlampe

Zur Ausgabe einfacher Lichtsignale dient die Glühlampe. Um bei unseren Fachbezeichnungen zu bleiben; die Glühlampe wird zum optischen Aktor. Der Aufbau einer Glühlampe ist recht simpel. In einem Glaskolben, in dem sich ein Vakuum befindet, ist eine Wendel aus dünnem Wolframdraht zwischen zwei Anschlußstiften aufgespannt. Fließt Strom durch die Wendel, erhitzt sich der Draht bis zur Weißglut. Da kein Sauerstoff im Glaskolben vorhanden ist, verbrennt der Draht nicht und die Lampe erreicht somit eine hohe Lebensdauer. Infolge der starken thermischen Beanspruchung der Glühwendel dehnt sich die Drahtwendel bei jedem Einschalten aus und zieht sich beim Ausschalten wieder zusammen. Diese minimalen Bewegungen führen dann infolge Materialermüdung irgendwann zum „Durchbrennen" der Glühlampe.



Eine Einsatzmöglichkeit der Glühlampe ist das Anzeigen von Schaltzuständen. Durch das Programmieren einer blinkenden Lampe können auch Warnmeldungen erzeugt werden.

Wir brauchen die Lampe noch in einem weiteren Fall. Zusammen mit zwei Fototransistoren entsteht ein Spezialsensor, mit dessen Eigenschaften Linien erkannt werden. Die Lampe arbeitet als Lichtquelle, damit der Fototransistor anhand des unterscheidlich stark reflektierten Lichtes eine Farbmarkierung erkennt.

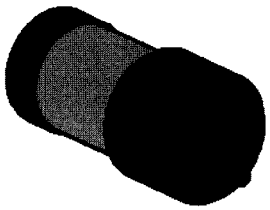
Eine Besonderheit der im fischertechnik-Baukasten zu Einsatz kommenden Glühlampe ist die im Glaskolben enthaltene optische Linse. Damit wird der Lichtstrahl besser gebündelt und wir erkennen z. B. Markierungen zuverlässiger.

3.4 Gleichstrommotoren als Kraftquelle

Gleichstrommotoren sind wichtige Aktoren für mobile Systeme. Im Baukasten „Mobile Robots II" sind zwei verschiedene Typen von Motoren enthalten. Mechanisch recht unterschiedlich, ist ihr elektrischer Aufbau jedoch identisch.

Gleichstrommotoren bestehen aus einem sich drehenden „Rotor" und einem fest stehenden „Stator". Der Rotor ist vom Prinzip her als eine Leiterschleife zu verstehen, die sich im magnetischen Feld des Stators befindet. Fließt nun

ein Strom durch die Leiterschleife entsteht eine Kraft, die zur Auslenkung des Leiters im Magnetfeld führt; der Rotor bewegt sich. Für praktische Anwendungen ist die einfache Leiterschleife als Spule (mit oder ohne Eisenkern zur Verstärkung des Magnetfeldes) ausgeführt. Sehr viele Gleichstrommotoren erzeugen das notwendige Magnetfeld mit Hilfe von Dauermagneten, die in den Metallmantel des Stators eingeklebt sind. Die Stromzufuhr zum drehenden Rotor erfolgt mit Hilfe von Schleifkontakten.



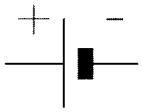
Diese Kontakte sorgen gleichzeitig für die Stromrichtungssumkehr in der Leiterschleife, die für eine ununterbrochene Drehbewegung notwendig ist.

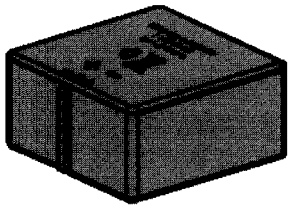
Die Drehzahl üblicher Motoren bewegt sich im Bereich von einigen tausend Umdrehungen pro Minute. Ein Getriebe sorgt für niedrigere Drehzahlen bei größerem Drehmoment.

Der Baukasten enthält zwei verschiedene Motortypen, den Minimotor und den Powermotor. Der kleine kompakte Minimotor mit Schnecke ist für Hilfsantriebe oder Sonderzwecke mit geringen Leistungsanforderungen gedacht. Er benötigt immer ein Getriebe zur Drehzahlreduzierung.

Der Power-Motor stellt viel größere Drehmomente zur Verfügung. Er hat ein fest angeflanschtes Getriebe mit einer Untersetzung von 50:1. Damit ist er ideal für unsere Antriebsanforderungen beim Roboter geeignet. Es gibt ihn auch in einer Variante mit Untersetzung 8:1 (nicht in diesem Baukasten enthalten). Hier wäre die Drehzahl auf der Abtriebswelle jedoch für den Roboterantrieb zu groß.

3.5 Stromversorgung





Mobile Systeme benötigen eine autonome Stromversorgung. Von dieser Energiequelle werden alle Verbraucher gespeist. Die Anforderungen an die Stromversorgung sind unterschiedlich. Während Antriebsmotoren sich mit unstabilisierten Spannungen zufrieden geben, benötigen viele Sensoren stabile Spannungen um exakte Ergebnisse liefern zu können.

Aus wirtschaftlichen Gründen ist der Einsatz von Batterien oder Akkus der einzig sinnvolle Weg, mobile Roboter mit Strom zu versorgen. Solarzellen oder Brennstoffzellen sind leider noch nicht leistungsfähig genug, um bei angemessenem Aufwand praktikable Ergebnisse zu liefern.

Akkus sind Batterien vorzuziehen, weil sie vielfach wieder aufgeladen werden können. Einen guten Kompromiss zwischen Energieinhalt und Größe stellt der fischertechnik-Akku-Pack dar. Der Akku-Pack ist nicht Bestandteil des Baukastens und kann zusammen mit einem speziellen Ladegerät als „Accu Set" unter der Art.-Nr. 34969 erworben werden. In der Abbildung ist das Schaltzeichen und der Akku dargestellt. Im Normalfall wird im Schaltzeichen die Polarität nicht angegeben. Mit Hilfe einer Eselsbrücke kann man sich leichter merken, welcher Anschluss Plus ist: „Den langen Strich kann man durchschneiden und zu einem Plus zusammensetzen."

Es ist sehr wichtig, beim Anschluss der Spannungsquelle an das Interface immer auf die richtige Polarität zu achten.

3.6 Zusätzliche Sensoren

Das fischertechnik-System ist relativ einfach um weitere Sensoren zu ergänzen. Im einfachsten Fall verwenden wir Sensoren aus anderen Kästen, z. B. den Wärmesensor oder den Magnetsensor aus dem Baukasten „Profi Sensoric" Art.-Nr. 30491.

Wir können jedoch auch völlig andere Sensoren verwenden. Im Fachhandel werden die unterschiedlichsten Bausätze und Komponenten angeboten. Selbst solch exotische Sensoren wie Gasmelder oder Radarsensoren sind verwendbar. Da wir das Intelligent Interface keinesfalls mit zu hohen Eingangspannungen oder falschen Lasten zerstören wollen, sollten jedoch nur erfahrene Bastler eigene Lösungen realisieren. Der sicherste Weg zum Anschluss weiterer Sensoren ist die galvanische Trennung zwischen Sensor und Interface. Eine Reihe von Sensoren besitzen ein Relais, das sich hierzu gut eignet. Die Schaltkontakte des Relais werden wie ein gewöhnlicher fischertechnik-Schalter angeschlossen und signalisieren nun das Auftreten neuer Umweltreize. Ein Tipp: im Internet werden viele derartige Erweiterungen von begeisterten „Fischertechnikern" veröffentlicht.

4 Die Robotermodelle

Mit den folgenden Aufbauvorschlägen werden einige Varianten mobiler autonomer Roboter vorgestellt. Wir beginnen mit einem einfachen Modell. Darauf aufbauend erkennst und erprobst du den Einsatz unterschiedlicher Sensoren. Dabei kommt es darauf an, sowohl interne Zustände des Roboters, z. B. Wegstreckenmessung durch Impulsräder, als auch externe Umweltsignale, wie 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 LLWin-Programme zu den einzelnen Aufgaben befinden sich auf der CD-ROM, die im Baukasten enthalten ist. Lass dir zu den Modellen aber auch eigene Aufgaben einfallen.

Das einfachste Modell ist das Basismodell. Hier werden die Antriebsmotoren mit dem Interface zu einer kompakten Einheit zusammengebaut. Zwei Motoren sorgen für die Antriebskraft des Roboters. Sie sind so gegenüber angeordnet, dass jeder Motor auf ein Antriebsrad wirkt. Damit dieser Roboter nicht umkippt, sorgt ein Stützrad für Stabilität. Eine derartige Anordnung der Motoren wird als „differential drive" bezeichnet. Sie gewährt die höchste Beweglichkeit mit minimal notwendigem Bewegungsraum. Es sind sogar Drehungen auf der Stelle möglich. Der Mittelpunkt beider Motoren ist dabei der Drehpunkt, um den sich der Roboter bewegt. Auf diese Weise gelingt es ihm unter schwierigsten Verhältnissen mit wenig Rechenaufwand zu navigieren.

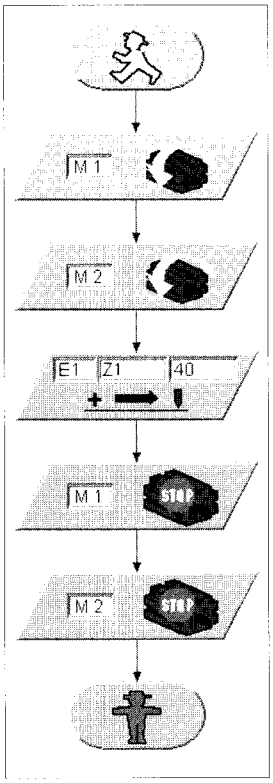
Die Motoren können über zwei unterschiedliche Getriebeuntersetzungen die Räder antreiben (langsam 100 : 1 bzw. schnell 50 : 1). Für die langsamere Variante wird der Antrieb zusätzlich mit fischertechnik-Zahnrädern im Verhältnis 2:1 untersetzt. Bei den Modellen ist jeweils angegeben, welche Untersetzung verwendet wird.

4.1 Das Basismodell

Zuerst bauen wir das Basismodell (Getriebeuntersetzung 100:1) entsprechend der Bauanleitung auf. Da uns dieses Modell bei vielen Experimenten als Grundlage dient, gehen wir beim Aufbau besonders sorgfältig vor. Wenn mechanisch alles fertig ist, prüfen wir die Leichtgängigkeit der Motoren. Dazu wird jeder Motor kurz ohne das Interface direkt mit dem Akku verbunden.

Aufgabe 1:

Programmiere das Interface so, dass das Modell 40 Impulse geradeaus fährt. Zur Messung der Impulse verwende den Zähltaster E1, als Resetschalter den Taster an E8.



Modifiziere danach das Programm so, dass das Modell unterschiedlich lange Strecken, z. B. 80 cm lang, abfährt. Wie groß ist dabei die Wiederholgenauigkeit?

Aufgabe 2:

Das Modell soll sich nach den 80 Impulsen Geradeausfahrt um 180° drehen. Beachte die unterschiedlichen Drehrichtungen der Antriebsmotoren bei Geradeausbewegung bzw. Drehung.

Lösung:

Im Mittel legt das Modell pro Zählimpuls 1 cm Wegstrecke zurück. Die Wiederholgenauigkeit liegt in der gleichen Größenordnung, etwa 1cm bei 80 Impulsen.
Sie schwankt jedoch in Abhängigkeit vom Untergrund, auf dem sich der Roboter bewegt. Besonders ungünstig sind dicke oder flauschige textile Bodenbeläge.

Bevor wir uns dieser Aufgabe zuwenden, wollen wir zwei Dinge klären.
Zum einen haben wir einen neuen Funktionsbaustein in unserem Programm verwendet, den Baustein POSITION. Hier handelt es sich um einen Baustein, der genauso lange aktiv bleibt, bis die eingestellte Anzahl von Impulsen am festgelegten Eingang (hier E1) erkannt wurde. Aus Sicht des Programms bedeutet dies, dass wir hier eine definierte Wartebedingung einsetzen.
Im ersten Versuch haben wir diese Funktion als Wegstreckenmessung zur Geradeausfahrt verwendet.
Soll sich der Roboter drehen, ist dies praktisch derselbe Ansatz, wir müssen nur die Drehrichtung der Motoren ändern. Nun brauchen wir nur noch die Anzahl der Impulse einzutragen und unser Roboter dreht sich auf der Stelle.
Und nun unser zweiter Punkt. Wir wollen nicht einfach solange probieren, bis der Roboter sich um 180° dreht; wir wollen vorher diesen Wert ausrechnen.
Die Antriebsmotoren sind als differential drive konfiguriert, d.h. die Räder des Roboters bewegen sich bei der Drehung auf dem Umfang eines Kreises, dessen Durchmesser vom Abstand der Räder bestimmt wird. Für eine Drehung um 180° muss jedes Rad deshalb eine Strecke von genau der Hälfte dieses Kreisumfangs zurücklegen.
Wir berechnen zuerst den Kreisumfang u:

$$u = \pi \cdot d = 630mm$$

d : Durchmesser (Radabstand ca. 200 mm)

Vorhin haben wir eine Wegstrecke von ca. 1 cm/Impuls ermittelt, damit brauchen wir für die 314 mm Wegstrecke (halber Umfang) 30,5 Impulse. Da wir nur ganzzahlige Werte berechnen können, müssen wir uns für 30 oder 31 Impulse entscheiden. Wir testen welcher tatsächliche Wert die größere Genauigkeit liefert.

Fazit:

Im Ergebnis unserer Messungen mit dem Impulsrad stellen wir fest, dass die erzielbare Genauigkeit unserer Messungen nicht allzuhoch ist. Insbesondere wenn mehrere Stecken hintereinander bzw. wiederholend abgefahren werden, summiert sich der absolute Messfehler auf. Genauso problematisch ist der Fehler, der durch noch nicht vollständig erfasste Takte zustande kommt.
Die Möglichkeiten diese Fehler zu minimieren sind begrenzt. Zum Einen kann man die Wegimpulse pro Streckeneinheit erhöhen. Idealerweise würde der Zähler direkt auf der Motorwelle sitzen. Neben der Tatsache, dass wir nicht an diese Welle herankommen, tritt hier das Problem der begrenzten Abtastrate des Interface auf. Wenn innerhalb einer Zeiteinheit zuviele Impule kommen, „vergisst" das Interface eventuell einige. Damit wird eine genauere Wegberechnung illusorisch.
Andere Fehler, wie z. B. den Schlupf der Räder auf verschiedenen Untergründen oder abweichende Raddurchmesser, können wir überhaupt nicht erfassen. Wir trösten uns mit dem Gedanken, dass diese Probleme teilweise auch von wesentlich komplexeren und teureren kommerziellen Systemen nur ungenügend gelöst sind.

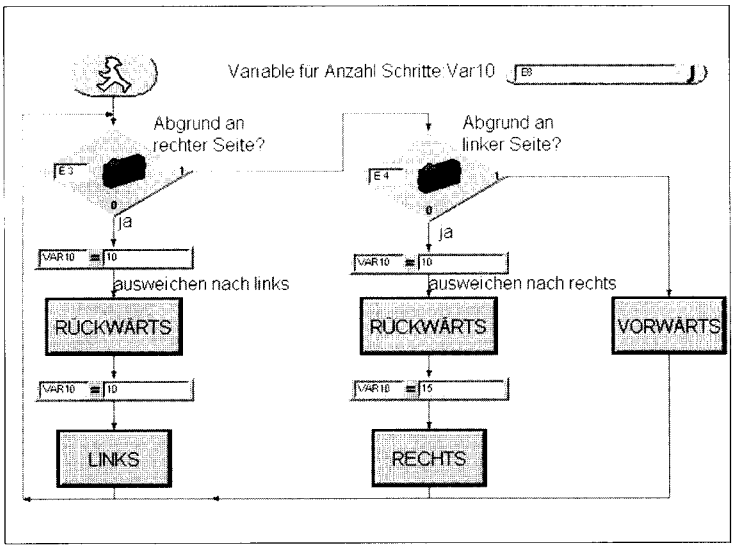
4.2 Roboter mit Kantenerkennung

Nachdem wir nun unser Basismodell entsprechend ausgiebig untersucht haben, wollen wir nun versuchen, dem Roboter die „Angst vorm Abgrund" beizubringen. Bisher ist der Roboter auf der Tischplatte umhergewuselt, von uns mit Argusaugen beobachtet, damit er ja nicht von der Platte stürzt. Dies scheint nun wahrlich kein besonders intelligentes Verhalten. Wir wollen es deswegen ändern.
Damit der Roboter eine Kante erkennt, benötigt er dazu einen Kantendetektor. Ein ebenso einfaches wie brauchbares Verfahren bedient sich zweier Hilfsräder. Diese werden ähnlich einem Fühler in Fahrtrichtung vor dem Roboter mit einem Schalter versehen. Die Räder sind so konstruiert, dass sie sich vertikal bewegen können. Eine Kante lässt das Hilfsrad nach unten fallen und löst somit den Sensor aus.

Aufgabe 3:

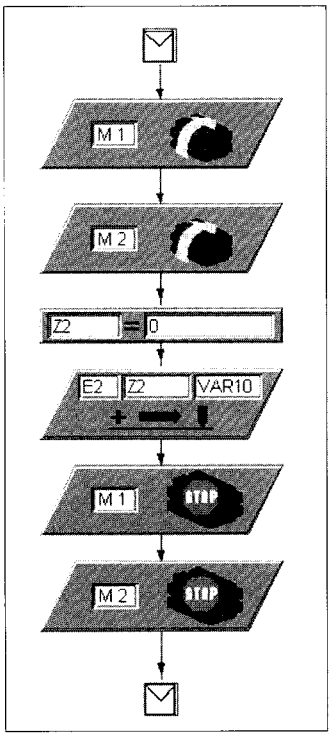
Baue das Modell „Roboter mit Kantenerkennung" entsprechend der Bauanleitung auf (Getriebeuntersetzung 50:1). Das Modell soll geradeaus fahren. Sobald es links an einen Abgrund kommt, soll es nach rechts ausweichen, ist rechts ein Abgrund, soll es nach links ausweichen. Zur besseren Übersicht werden bestimmte Bewegungen als Unterprogramme (Vorwärts, Links und Rechts) eingesetzt.
Die Schrittzahl wird über Zähltaster erkannt. Die Anzahl der Schritte wird in einer Variablen VAR10 vorgegeben. Diese Vorgabe ist für die Unterprogramme Links und Rechts verschieden. Die unterschiedlichen

Vorgabenwerte vermindern das Risiko in einer Ecke steckenzubleiben.



Lösung:

Aus der Aufgabenstellung erkennen wir, dass wir den Roboter in Abhängigkeit von den Kantendetektoren steuern müssen. Wir zerlegen die Aufgabe deshalb in kleinere Teile. Zuerst fragen wir die Taster (Kantendetektoren) ab. Ist kein Taster aktiv, fährt der Roboter vorwärts. In der Abbildung ist dies als Block „Vorwärts" zu sehen. Hinter einem solchen für uns neuen Baustein verbirgt sich ein Unterprogramm. Unterprogramme verbessern die Übersichtlichkeit komplexer Systeme und zudem können sie durch mehrfache Verwendung die Leistung von Computersystemen besser ausnutzen. Unser erstes Unterprogramm ist sehr einfach, es setzt lediglich die Motoren M1 und M2 in Bewegung.



Wir brauchen jedoch weitere Unterprogramme. Der Roboter soll ausweichen, je nach Lage entweder nach links, rechts oder zurück. In diesem Fall genügt es nicht mehr, nur einfach die Motoren zu schalten. Es muss ein bestimmter Bewegungsablauf programmiert werden.

Schauen wir uns ein weiteres Unterprogramm an. Damit soll der Roboter beim Erkennen einer Kante zurückfahren.

Wir schalten dazu die Motoren auf Rückwärtsfahrt um. Dann soll unser Impulsrad eine definierte Strecke erkennen. Die Streckenlänge legen wir mit der Variablen VAR10 fest. Neu ist der Zuweisungsblock Z2 = 0. Nach einigem Überlegen erkennen wir den Sinn. Setzen wir die Zählvariable nicht auf Null, funktioniert der Mechanismus nur einmal, denn wenn Z2 einmal den Wert der Variablen VAR10 erreicht hat, würde bei jedem folgenden Durchlauf unser Wegstreckenzähler versagen. Aus der Sicht professioneller Programmierer haben wir es an dieser Stelle mit lokalen und globalen Variablen zu tun. Die lokale Variable Z2 wird vor jeder Benutzung im Unterprogramm initialisiert.

Fazit:

Unterprogrammaufrufe erhöhen die Übersichtlichkeit von Programmen.

Wir benutzen Variablen um verschiedene Werte, hier Weglängen, zu messen. Die unterschiedlichen Weglängen brauchen wir, damit sich unser Roboter auch aus Ecken „befreien" kann. Wären die Wege absolut identisch, könnte es passieren, dass der Roboter in der Ecke immer hin und her fährt.

Bei der Benutzung von Variablen achten wir auf ihren Gültigkeitsbereich. Lokale Variablen, d.h. Variablen, die wir nur innerhalb eines Unterprogramms verwenden, initialisieren wir vor ihrem ersten Einsatz.

Wir stellen außerdem fest, dass unser Roboter einen gewissen Bewegungsspielraum braucht, damit er richtig funktioniert. Trifft er innerhalb einer Ausweichbewegung erneut auf eine Kante, kann der Roboter darauf nicht reagieren. Die ganz großen Tüftler können versuchen, ein Lösung dafür zu finden.

4.3 Roboter mit Hinderniserkennung

Kanten können wir nun recht gut erkennen. Problematisch wird es bei gewöhnlichen Hindernissen. Wir müssen das Prinzip der Kantenerkennung modifizieren. Ein entsprechender Aufprallkontakt ersetzt die Hilfsräder. Bei der Gelegenheit denken wir auch über einen Mangel des Kantendetektors nach; es gibt gefährliche Situationen, bei denen der Roboter während der Rückwärtsfahrt ohne Sensor hinten „blind" in der Abgrund stürzt.

Ein dritter Taster vermeidet diesen Mangel. Mittlerweile sind wir recht ausgefuchste Programmierer geworden. Ein Glück, denn nun wird die Aufgabe komplizierter, wie ein Blick auf das Programm zeigt.

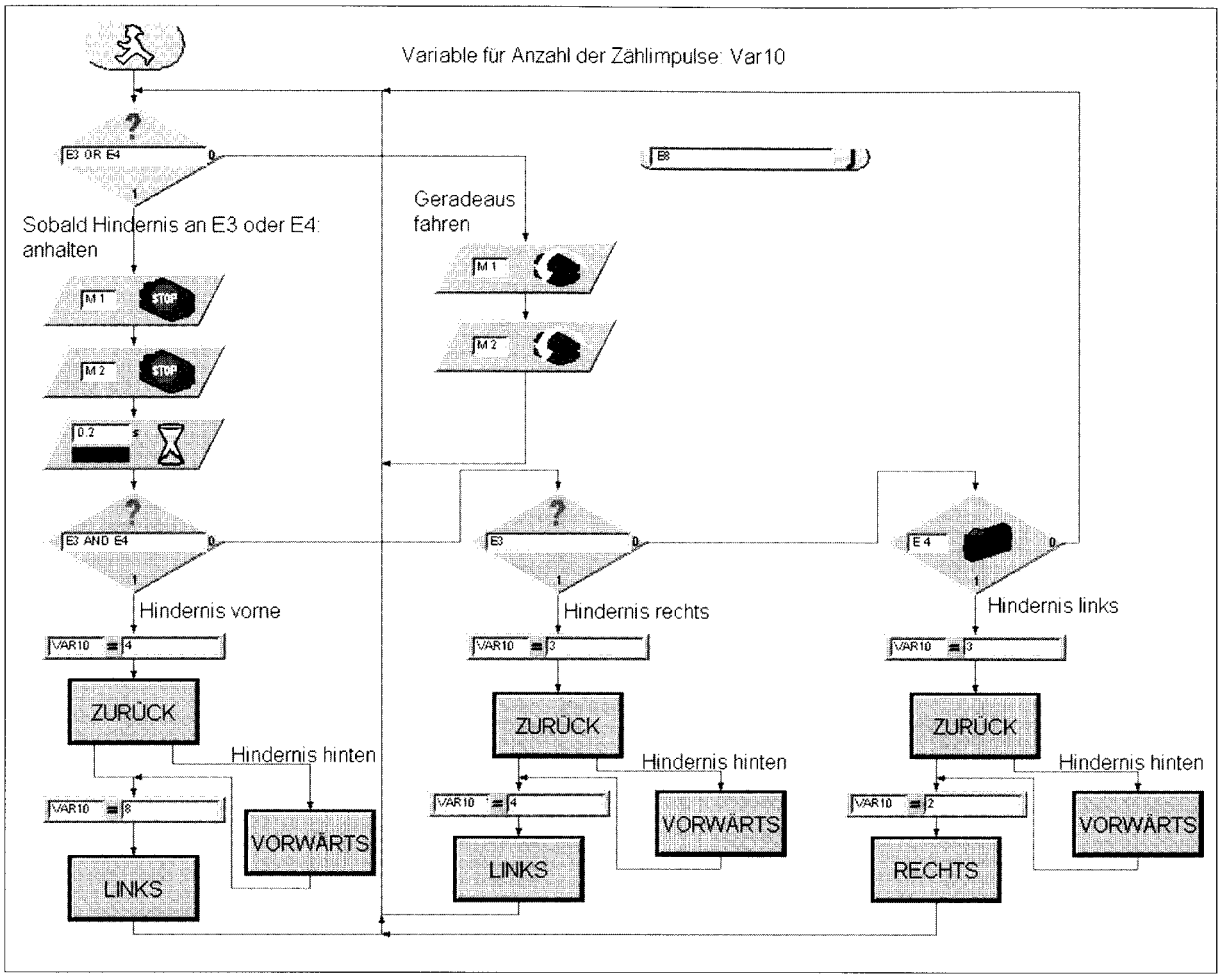
Das Programm beinhaltet weitere neue Funktionsblöcke. Wir fügen an bestimmten Programmpositionen WARTE-Bausteine ein. Das ist einfach verständlich, hier wird die eingetragene Zeit abgewartet, bevor die nächste Programmfunktion abgearbeitet wird.

Richtig neu sind logische Vergleiche. Bislang haben wir immer gleich bei der Tasterabfrage einen Vergleich ausgeführt und unser Programm entsprechend verzweigt. Einen ähnlichen Vergleich gibt es auch bei der Impulszählung, den Vergleich mit einer bestimmten Anzahl von Impulsen. Neu ist auch der logische Vergleich mit mehreren Ausdrücken in einem Baustein VERGLEICH.

Aufgabe 4:

Der Roboter wird, wie in der Bauanleitung beschrieben, mit der schnelleren Getriebeuntersetzung 50:1 aufgebaut. Das Modell soll geradeaus fahren. Sobald ein Hindernis an einem der vorderen Taster (E3 oder E4) erkannt wird, stoppt der Roboter. Falls ein Hindernis rechts erkannt wurde, fährt der Roboter zurück und weicht dann nach links aus (ca 30°). Bei einem Hindernis von links weicht der Roboter nach dem Zurückstoßen nach rechts aus (ca 45°). Die ungleiche Gradzahl ist notwendig damit er auch aus einer Ecke heraus findet. Falls das Hindernis direkt von vorn erkannt wird, soll der Roboter zurück und um 90° ausweichen. Falls beim Rückwärtsfahren ein Hindernis von hinten kommt, soll er kurz nach vorne und dann wie geplant ausweichen.

Lösung:



Fazit:

Die Komplexität der Programme wird größer. Wir versuchen dieser Schwierigkeitssteigerung durch bessere Programmiertechniken zu begegnen. Unterprogramme erweisen sich dabei als gutes Mittel. Als neue Kontrollstruktur erkennen wir logische Vergleiche mit anschließender Programmverzweigung. Wir erkennen auch, dass für neue Eigenschaften bzw. eine bessere Realisierung bereits bekannter Versuche immer mehr Sensoren nötig werden.

4.4 Der Lichtsucher

Bislang haben wir den Roboter eher passiv auf Umweltsignale reagieren lassen. Nun wollen wir den Roboter aktiv auf die Suche schicken. Der Baukasten enthält zwei Fototransistoren, die wir als Lichtdetektor einsetzen. Jeder Sensor wirkt dabei auf einen Motor, damit ist eine „Leitstrahl-Verfolgung" möglich.

Das Programm besteht aus zwei Teilen. Der eine Teil beinhaltet die Suche nach einer Lichtquelle und im anderen Teil wird die Verfolgung bzw. das Ansteuern der Lichtquelle realisiert.

Wir machen natürlich wieder von den Möglichkeiten der Unterprogrammtechnik Gebrauch. 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 zu steuern. Immer wenn die Richtung des Roboters stark von der Ideallinie abweicht, wird einer der Lichtsensoren nicht mehr von der Lichtquelle bestrahlt. Daraufhin wird der entsprechende Motor kurz gestoppt, so dass beide Sensoren wieder die Lichtquelle erkennen können. Daraus resultiert die Aufgabenstellung.

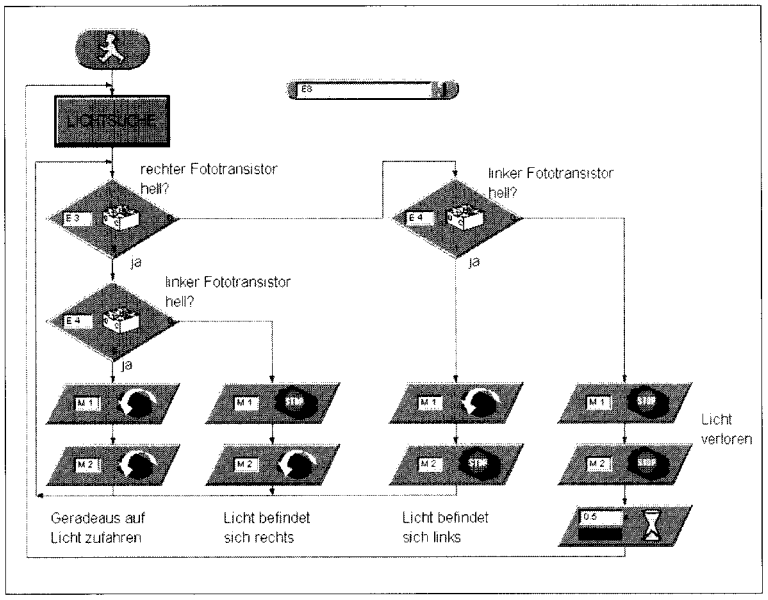
Aufgabe 5:

Wir bauen das Modell Lichtsucher mit der langsameren Untersetzung 100 : 1. Programmiere zuerst die Funktion „Licht suchen" (am Besten gleich als Unterprogramm). Der Roboter dreht sich dabei um mindestens 360° in einer Richtung. Anschließend dreht er um mindestens 360° in die andere Richtung. Wenn während der Suche ein Licht gefunden wird, stoppt der Roboter. Wird nach beiden Umdrehungen kein Licht gefunden, wird die Suche abgebrochen und das Programm beendet.

Ist die Lichtsuche erfolgreich, soll das Modell die Lichtquelle ansteuern. Bewegt sich die Lichtquelle nach links oder rechts, folgt der Roboter den Bewegungen des Lichtes. Verliert er den Kontakt zur Lichtquelle, beginnt das Programm mit der erneuten Lichtsuche. Probiere aus, ob du den Roboter mit der Taschenlampe anlocken und durch einen Hindernisparcours führen kannst.

Tipp:
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 wird. Der Roboter fährt dann unter Umständen an deiner Lampe vorbei auf das hellere Licht zu.

Lösung:



Fazit:
Wir haben nun einen Roboter konstruiert, der aktiv von seiner Umgebung Notiz nimmt. Seine Sensoren sind darauf programmiert, eine Lichtquelle zu orten und falls er sie lokalisiert hat, diese Quelle anzusteuern bzw. ihr zu folgen.
Wir stellen fest, dass das Programm den Roboter stilllegt, wenn keine Lichtquelle gefunden wurde. Wenn zum Beispiel nur ein kleines Hindernis den direkten Sichtkontakt zum Roboter unterbricht, würde dieser, obwohl eine Lichtquelle da ist, diese nicht erkennen und finden können. Offenbar wäre es sinnvoll, nach einer erfolglosen Lichtsuche den Roboter mehr oder weniger zufällig auf eine andere Stelle zu bewegen und dort neu nach Licht zu suchen.
Doch was passiert, wenn das Hindernis, welches das Licht nicht zum Roboter lässt, genau in der Fahrtrichtung des Roboters liegt? Nun, diese Fragestellung scheint interessant genug, sie noch genauer zu untersuchen.

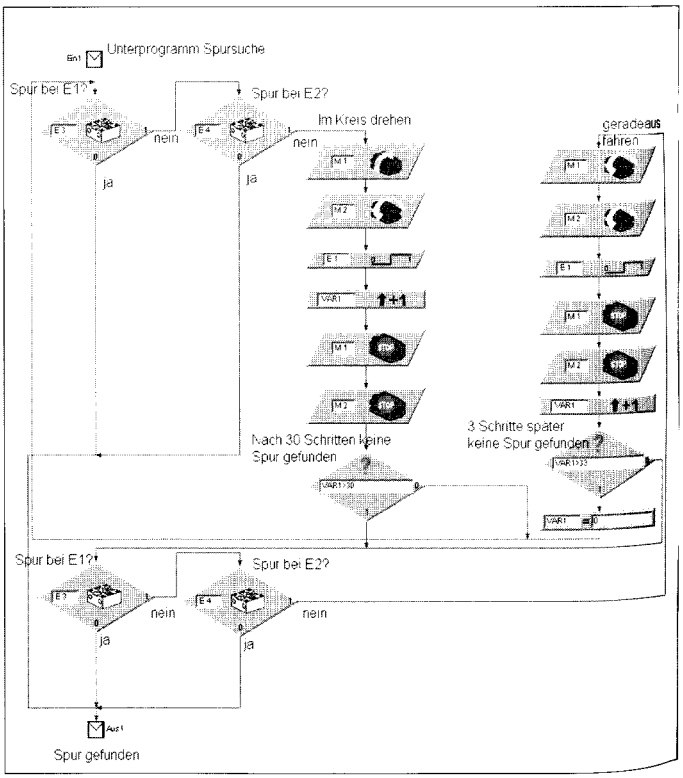
4.5 Der Spurensucher

Suche und Verfolgung sind wesentliche Eigenschaften, die intelligente Wesen besitzen. Wir haben einen Roboter gebaut und programmiert, der auf direkte Signale von seinem Ziel oder potentiellen Opfer reagiert hat.
Mit dem Spurensucher wenden wir ein anderes Suchprinzip an. Anstelle der zielgenauen Fahrt zur Lichtquelle markieren wir eine Linie, der der Roboter folgen soll. Mit den optischen Sensoren ist diese Aufgabe relativ leicht zu lösen. Wir messen das reflektierte Licht der Markierung und korrigieren danach die Motoren. Damit das auch exakt funktioniert, beleuchten wir die Linie mit unserer Lampe. Dabei achten wir darauf, dass durch ungünstige Anordnung von Lampe und Fotosensoren letztere nicht von Streulicht geblendet werden. Besonders günstig wirkt sich in diesem Zusammenhang die Lichtbündelung der optischen Linse unserer Glühlampe aus.

Aufgabe 6:
Baue das Modell Spurensucher (Getriebe 100 : 1). Als Erstes schreibe ein Unterprogramm, mit dem die Spur gesucht wird. Der Roboter dreht sich dazu 360° im Kreis herum. Wenn keine Spur gefunden wird, fährt der Roboter ein Stück geradeaus und sucht erneut. Zur Spurerkennung werden die Fotosensoren abgefragt. Hat der Roboter die Spur entdeckt, folgt er ihr. Ist die Spur zu Ende, oder verliert der Roboter diese, z. B. wegen einer starken Richtungsänderung der Spur, beginnt die Suche von neuem.

Tipp:
Nach dem Einschalten der Lampe muß eine kurze Wartezeit, ca. eine Sekunde verstreichen, bevor die Fototransistoren abgefragt werden, sonst erkennt der Fototransistor „dunkel", also eine Spur, wo keine ist. Als Spur verwenden wir ca. 20 mm breites schwarzes Isolierband oder malen uns mit Filzstift eine schwarze Spur in dieser Breite auf ein weißes Blatt Papier.

Lösung:

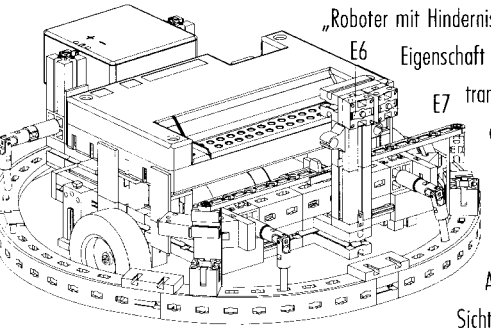


Fazit:

Hat unser Roboter die Spur gefunden, folgt er ihr unermüdlich. Wenn wir die Spur als geschlossene Linie anlegen, dreht der Roboter darauf seine Runden. Wir müssen nur zu enge Kurven vermeiden, in denen die Fotorsensoren den Kontakt zur Linie verlieren könnten. Zwar versucht der Roboter dann erneut die Spur zu lokalisieren, doch dies ist, so geben wir fairerweise zu, nur in praktisch hindernisfreier Zone möglich.

4.6 Die elektronische Motte

Das Problem der Kopplung verschiedener Verhaltensweisen, wie Suche, Verfolgen und Ausweichen ist nun bereits mehrfach aufgetaucht. Wir wollen deshalb in einem weiteren Experiment das Zusammenspiel solcher Verhalten untersuchen. Fassen wir unsere bisherigen Ergebnisse kurz zusammen. Der einfache Lichtsucher folgt einer vorgehaltenen Taschenlampe mehr oder minder blindlings. Der Roboter geht stillschweigend davon aus, dass, wo vorher die Lampe war, kein Hindernis auftauchen kann. Genauso nimmt er an, dass er die Lampe nicht wirklich erreicht. Diese Annahmen entsprechen jedoch nur in Ausnahmefällen der Realität. Wir sind gezwungen, den Roboter mit der Lampe um jedes Hindernis herum zu leiten. Wir bräuchten eigentlich einen Roboter, der Hindernissen selbstständig ausweichen kann. Dazu ergänzen wir, wie in der Abbildung dargestellt, das Modell



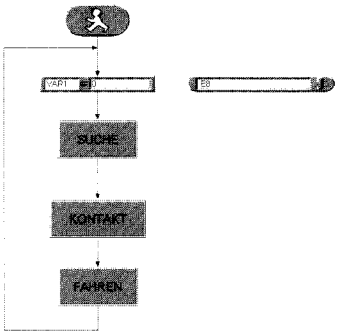
„Roboter mit Hinderniserkennung" um die Eigenschaft der Lichtsuche. Die Fototransistoren schließen wir an E6 und E7 an, alles Andere übernehmen wir von dem Modell Hinderniserkennung. Aus wissenschaftlicher Sicht haben wir den Roboter dann mit zwei Verhaltensweisen ausgestattet. Da jedoch nicht beide Verhaltensmuster gleichzeitig aktiv sein können, erhalten Sie unterschiedliche Prioritäten.

Für uns stellt sich dies so dar, dass der Roboter im Normalfall auf Lichtsuche ist. Wird ein Hindernis erkannt, quasi eine Gefahr für den Roboter, wird das Verhalten Hindernisvermeidung aktiv. Ist alles wieder im grünen Bereich, kann der Roboter weiter nach der Lichtquelle forschen.

Damit haben wir dem Roboter wesentliche Fähigkeiten vermittelt, die es ihm erlauben, selbstständig zu navigieren und Gefahren auszuweichen. Wer einen Freund hat, der ebenfalls im Besitz eines Fischertechnik-Kastens ist, kann dieses Experiment noch weiter treiben. An jeden der beiden Roboter wird einfach eine Lichtquelle montiert und dann suchen sich die Roboter gegenseitig.

Bislang haben wir bei der Programmierung meist nach dem Prinzip „Versuch und Irrtum" gearbeitet. Du hast wahrscheinlich bei deinen Programmen „einfach" angefangen und im Laufe deiner Experimente erkannt, wie (oder wie besser nicht) der Roboter dann das getan hat, was er tun sollte. Professionelle Softwareentwickler können natürlich bei ihren Produkten keinesfalls auf eine solche Entwurfsmethode zurückgreifen. In solchen Fällen muss, bevor die erste Programmzeile erstellt wird, eine genaue Entwurfsstrategie definiert werden. Das macht man nicht einfach so, sondern benutzt feste Richtlinien. Durchgesetzt haben sich Verfahren, die solche seltsamen Bezeichnungen wie z. B. „Top-Down-Entwurf" tragen. Man versucht hier das Gesamtsystem von oben herab zu definieren, ohne sich am Anfang mit allen Details zu befassen. Wir versuchen unsere Motte nach dem Top-Down-Entwurf zu programmieren. Dazu beginnen wir mit dem Hauptprogramm, der sogenannten „Main-Loop".

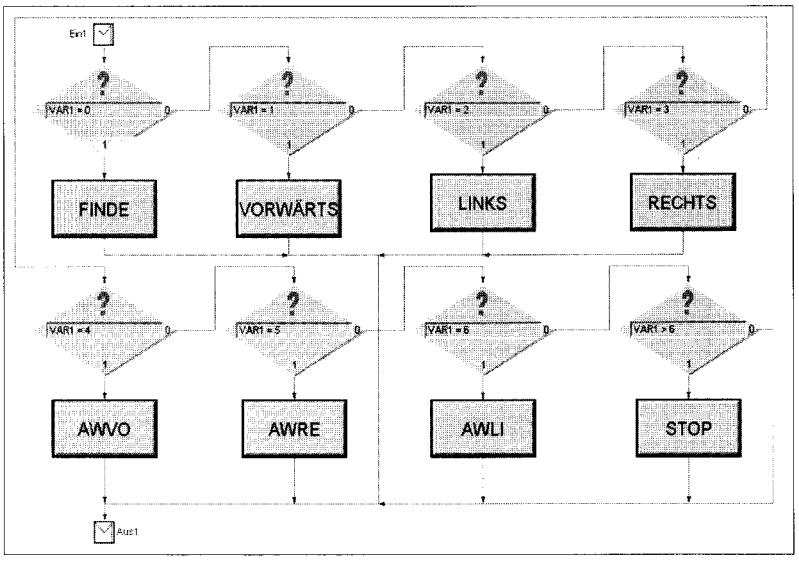
Die Abbildung zeigt den einfachen Entwurf. Die Main-Loop besteht lediglich aus den für den Roboter notwendigen Verhaltensweisen, SUCHE, KONTAKT und FAHREN. Seltsam erscheint uns die Variable Var1 und wie priorisieren wir mit einer solch einfachen Schleife die Verhaltensweisen des Roboters?



Nun, die Priorisierung erfolgt durch die Reihenfolge der Unterprogramme und zur Zuweisung der Fahraufträge dient die Variable Var1. Wenn wir genau überlegen, können wir die notwendigen Fahrbewegungen des Roboters auf eine klar definierte Anzahl Manöver reduzieren. Der Roboter benötigt eine Suchbewegung, eine Ausweichbewegung und eine Korrekturbewegung. Ausweich- und Korrekturbewegungen müssen noch nach Richtungen, rechts oder links, unterschieden werden. Damit wird deutlich, dass das Unterprogramm SUCHE eine von n (n = Gesamtzahl der Bewegungen) möglichen Bewegungen errechnet, genauso wie das Unterprogramm KONTAKT. Da KONTAKT nach SUCHE ausgeführt wird, überschreibt es die Anweisungen von SUCHE. Das Unterprogramm FAHREN führt nur noch aus, was ihm befohlen wird. Das Top-Down-Entwurfsverfahren scheint uns wirklich zu nützen.

Doch noch wissen wir nicht, wie die Unterprogramme aussehen. Beginnen wir mit dem ersten Unterprogramm SUCHE. Dieses Unterprogramm ist für die Abfrage der Fotosensoren zuständig. In Abhängigkeit von den beiden Sensoren gibt es vier mögliche Varianten, linker Sensor, rechter Sensor, linker und rechter Sensor, kein Sensor. Damit werden exakt vier mögliche Fahrmanöver definiert, FINDE, VORWÄRTS, LINKS, RECHTS. Auch diese Fahrmanöver werden Unterprogrammen zugeordnet; wir gehen streng nach Top-Down-Entwurfskriterien vor. Jetzt ist klar, wie das erste Unterprogramm SUCHE aussieht. SUCHE liefert einen Parameter der in FAHREN umgesetzt wird.

Das eigentliche Unterprogramm ist sehr einfach. Über mehrfache Vergleiche wird der Ausgangsparameter eingestellt. Im fertigen Programm wird danach im Unterprogramm KONTAKT geprüft, ob Hindernisse die Taster am Roboter ausgelöst haben. Der Einfachheit halber übergehen wir dieses Programm und sehen uns an, wie in FAHREN die Motorbewegungen aktiviert werden.



Mit einigem Erstaunen stellen wir fest, dass in diesem Unterprogramm weitere Unterprogramme aufgerufen werden. Hört das denn niemals auf? Wir sind immer noch in der Entwurfsphase (Top-Down nächste Stufe). FAHREN legt fest, wann welches Fahrmanöver gestartet wird. Erst hinter den verschiedenen Unterprogrammen FINDE, VORWÄRTS,... verbirgt sich „richtiger" Programmcode. Wir sind endlich ganz unten angekommen. Jetzt werden die Motoren an- bzw. ausgeschaltet. Jedes dieser Unterprogramme hat eine klar begrenzte Aufgabe, die wir übersichtlich programmieren können.

Fazit:

Komplexe Programme erfordern völlig neuartige Lösungsansätze. Es ist sinnvoll, einen durchgehenden Lösungsansatz anzustreben. Wir erstellen ein Programm nach dem Top-Down-Entwurfsansatz. Die notwendigen Lösungsansätze werden in (zuerst) formalen Strukturen formuliert, z. B. SUCHE oder FAHREN. Erst im weiteren Verlauf werden diese Ansätze verfeinert und zuletzt mit dem eigentlichen Programmcode gefüllt. Damit trennen wir die Programmstruktur vom eigentlichen Programmcode. Beides können wir getrennt betrachten und, das ist sehr wichtig, auch getrennt nach Fehlern analysieren.

4.7 FTS – Fahrerloses Transportsystem

Verlassen wir das Gebiet der wissenschaftlichen Experimente, böse Zungen sprechen oft von Spielerei, und begeben wir uns in den Bereich der praktischen Anwendungen.

Wir wollen nun einen Roboter nicht nur um seiner selbst willen bauen und programmieren, sondern er soll eine Aufgabe erledigen. Zu diesem Zweck wird der Spurensucher mit einer beweglichen Transportgabel ausgerüstet. Damit entsteht ein Gabelstabler-Roboter, der in der Lage ist auf einer Palette gelagertes Material zu transportieren.

So einfach dieser Versuch erscheinen mag, enthält er doch alle Komponenten einer industriellen Anwendung. Und, derartige Transportsysteme sind zum Teil bereits heute im Einsatz.

Mittlerweile sind wir, was die Programmiererfahrung betrifft, gut geschult, so dass wir uns frohen Mutes an dieses doch sehr komplexe Programm wagen.

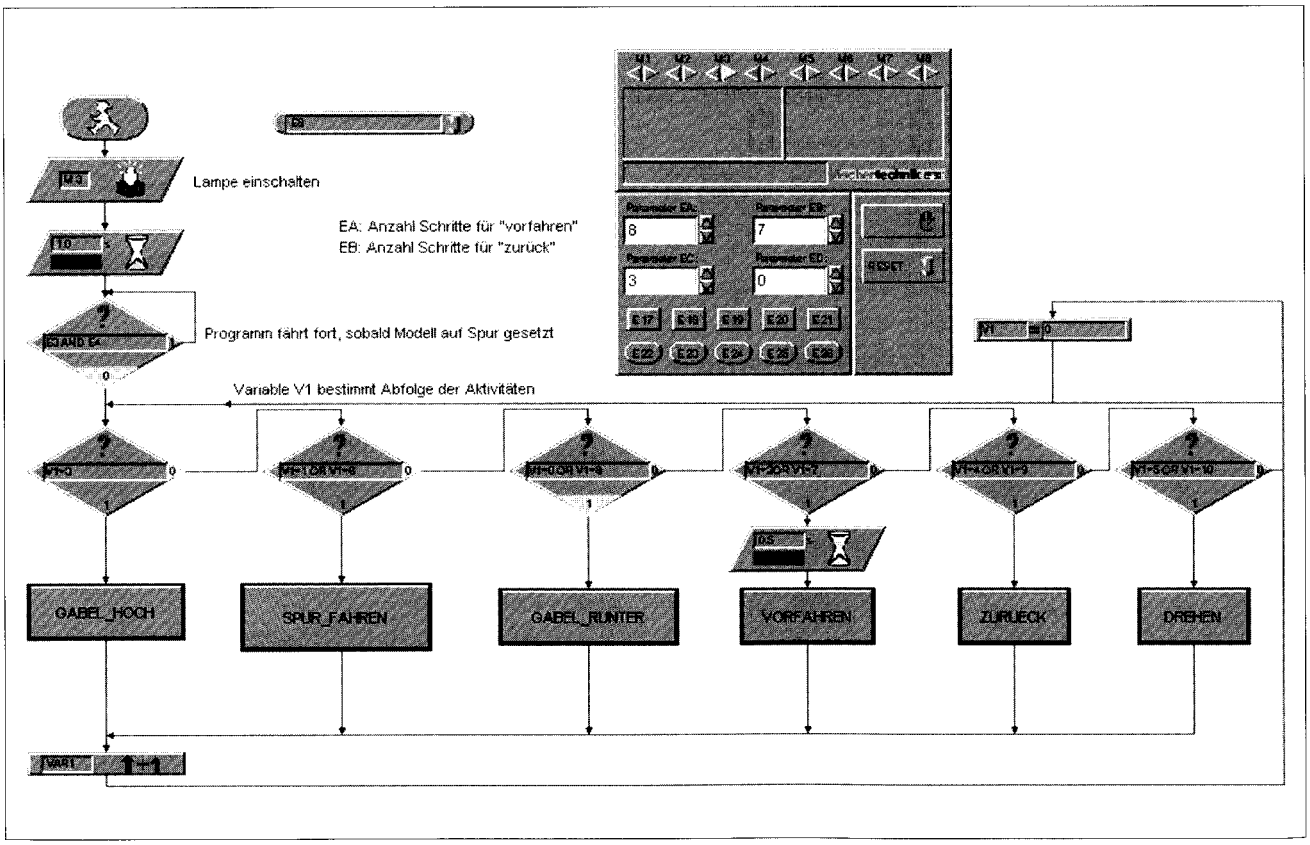
Aufgabe 7:

Das fahrerlose Transportsystem (FTS) soll entlang einer schwarzen Spur fahren. Am Ende der Spur nimmt es eine Palette auf, dreht sich um 180° und fährt auf der Spur zurück. Wird das Spurende detektiert, setzt der Roboter die Palette dort ab, wendet erneut und holt die nächste Palette.

Aufgabe 8:

Als Erweiterung der Aufgabe 7 ergänzen wir das Programm so, dass die Palette an den Endpunkten der Spur nur jeweils kurz abgesetzt wird. Der Roboter sucht also die Palette, fährt zum Spurende, setzt die Palette kurz ab, nimmt sie wieder auf, fährt zum anderen Ende,...usw.

Lösung:



Fazit:

Sobald wir einen Blick auf das Programm werfen, erkennen wir sofort den bereits bekannten Programmansatz. Eine Zustandsvariable steuert das Verhalten unseres fahrerlosen Transporters.

Mit Hilfe unserer mittlerweile recht ausgefeilten Programmiertechniken gelingt es uns, vielfältige Steuerungen zu realisieren. Mit dem fahrerlosen Transportsystem demonstrieren wir die Möglichkeiten, die die Kombination der fischertechnik-Baukästen mit dem Intelligent Interface bietet. Beginnend mit einfachen mobilen Robotern haben wir nun einen Stand erreicht, der sich kaum noch von der industriellen Steuer- und Regeltechnik unterscheidet.

5 Fehlersuche

Experimentieren macht Spaß. Doch nur solange alles funktioniert. Am besten wäre es, wenn alles immer auf Anhieb gelänge. Doch leider ist dies nicht so. 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.

Bevor wir mit unseren Experimenten beginnen, müssen wir einige Komponenten aus dem Fischertechnik-Baukasten erst fertigstellen. Im Wesentlichen handelt es sich um die Kabelverbindungen. Hier werden 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. Bevor wir schneiden, prüfen wir genau, ob die Längen stimmen. 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. Stimmt auch die Farbzuordnung, roter Stecker rotes Kabel, grüner Stecker grünes Kabel, legen wir das Kabel zur Seite und prüfen das nächste.

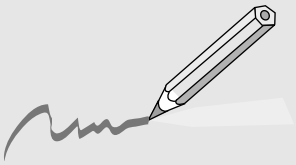
Arbeitet das Programm (auch das mitgelieferte) nicht mit unserem Modell zusammen, starten wir die Interfacediagnose. Dieses Hilfsprogramm gestattet uns, die Ein- und Ausgänge separat zu testen. Hier muss jeder Sensor die entsprechende Aktion am Interface auslösen.

Wenn wir das geprüft haben, wissen wir, dass elektrisch alles in Ordnung sein sollte. Über die Motorsteuerknöpfe schalten wir die Aktoren einzeln ein bzw. aus. Ist hier ebenfalls alles in Ordnung, suchen wir die mechanische Ursache.

Ein übler Fehler sind Wackelkontakte. Zum einen können die Anschlussstecker lose in den Buchsen sitzen. Ist dies der Fall, werden mit einem kleinen Uhrmacherschraubenzieher die Kontaktfedern der Stecker etwas aufgeweitet. Vorsicht, zu starkes Aufweiten führt zum Bruch der Kontakte oder zu starker 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. 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 Computer auf dem Interface ausgelöst. Ein solcher Fehler ist zum Teil sehr schwer zu finden, da das Programm ja immer erst einmal funktioniert.

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.Führt dies alles nicht zum Erfolg, bleibt noch der Kontakt zum fischertechnik-Service.



GB+USA

CONTENTS

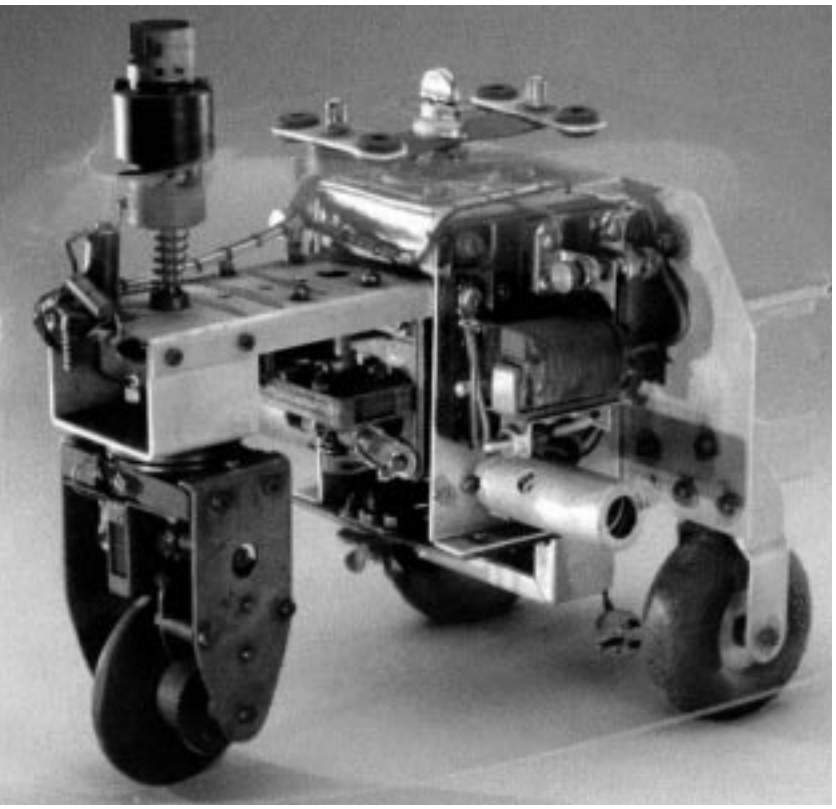
GB+USA

1 What Do We Need Robots for?

Before we deal with the practical aspects of robot technology, we want to try to answer the slightly provocative question in the heading.
The term "Robot" was first used in the novel "Golem" by Carel Capek in 1923. This gloomy, artificially created figure was supposed to replace human work with its capabilities.

As often with literary figures, it is subject to compulsive behavior and we regard the figure with a certain degree of distrust. Robots became more of a kind of automatic device in the 30s and 40s of the twentieth century. Various attempts to give them external human characteristics, e.g., a head with blinking lights as eyes or primitive speech output via a loudspeaker, seem naive from today's point of view. Apparently, the fears of potential domination by robots over people cannot be refuted so easily.

But little mobility, not to mention intelligence, can be seen in these first simple experiments with the constructed machines. The building of robots only became realistic with the invention of electronic circuits.
The problem of required control principles is closely linked with actual robot technology. This question about the "intelligence" of robots is the object of research and investigation of many companies, institutes and universities today.



The first solution strategies were seen in cybernetics. The term "cybernetics" is derived from the Greek word Kybernetes. Kybernetes was the navigator on Greek deep-water ships. He had to determine the ship's location and calculate the necessary course.
Consequently, it is clear that cybernetics should make robots "intelligent." How can we imagine such intelligent behavior in general?

We want to try to make this clear by experimenting with our imagination. Each of us has already observed the behavior of a moth in the light cone of a lamp. The moth detects the light source, flies toward it, and then avoids the lamp shortly before crashing into it. It is clear that the moth has to detect the light source, determine the path to it and then fly to it in order to behave as it does. These abilities are based on the instinctive, intelligent behavior patterns of the insect.

Now let's try to transfer these abilities into a technical system. We have to detect the light source (optic sensors), perform movement (control motors), and we need to establish a meaningful relation between detection and movement (the program).
Our experiment with our imagination then combines an optic sensor with a motor and logic, so that this motor controls the vehicle to always move in the direction of the light source. This vehicle would then act exactly like a moth, wouldn't it?
A British man named Walter Grey conducted the described experiments with such technology in the 50s. Using simple sensors, monitors and electronic circuitry, he created several "cybernetic" animals, which had very specific behavior such as a moth.
These machines represent an important step on the path to modern, mobile robots. The sensors (photoelectric resistance, detectors, etc.) of the machines controlled the actuators (motors, relays, lamps, etc.) using their electronics, so that (apparently?) intelligent behavior resulted. The picture shows a copy of the "cybernetic" turtle, which is displayed in the Smithsonian Museum, Washington, D.C.

Based on these considerations, we are going to create corresponding "behavior patterns" and try to make them comprehensible to a robot in the form of programs.
But how can be answer the question about the benefits of mobile robots asked at the beginning of this text? To answer this question concretely, let's try to apply the previously rather abstract behavior of our "imagined moth" to technical issues. A simple example of this is the search for light. We modify the light source in that we place a bright strip, a guiding line, on the floor and direct the sensors down instead of forward as previously. Using such guidelines, mobile robots can be given a sense of orientation in a warehouse, for example. Additional information, e.g., in the form of a barcode at specific spots of the line cause the robot to perform further operations at such positions, e.g., to pick up or put down pallets.

Such robot systems actually already exist. In large hospitals, there are sometimes very long transport routes for consumables such as sheets. Transport of these materials by nurses or other care-givers takes a lot of time and involves hard manual labor in part. In addition, such work uses up the time available for caring for patients.
Consequently, we can see that mobile robots can take an important place in modern society. But what is the relation of this to fischertechnik construction kits?

In addition to sensors and actuators for a robot, we need many mechanical parts to construct a model. The fischertechnik construction kit Mobile Robots II

is an ideal basis for this. We can combine the mechanical parts in an almost unlimited variety and build sturdy robotic vehicles. Using the associated "Intelligent Interface" (Art. no. 30402, which must be purchased separately), we also have sufficient computing power to design demanding programs. Numerous different sensors and actuators are linked and evaluated via this interface. The sensors convert physical quantities such as light quantity or temperature into electrically recordable values. This includes both analog and digital quantities. Digital quantities are those that can be either logically true or false. These states are identified with 0 or 1. A switch is a good example of a digital sensor.

But many quantities change continually between their maximum and minimum values; these are called analog values. They cannot be displayed simply as 0 or 1. For a computer to be able to process these quantities, they must be converted into corresponding numeric values. The fischertechnik interface provides two analog inputs EX and EY. The resistance value applied to these terminals is stored in a numeric value. The measured values of a temperature sensor, for example 0...5 kΩ, are recorded in a range from 0 to 1024 and are available for subsequent processing. The most important function of the interface is in the logical linkage of input values. The interface requires a program for this. The program decides how the sensor signals, corresponding output data, motor control signals, etc. are created from input data.

A graphic work sheet exists, so that we can write programs necessary for the interface as effectively as possible. There is a software program behind the term “work sheet,” which makes it possible for us to create our programs on a very high level. In fact, we can design programs and algorithms using graphic symbols. The computer of the Intelligent Interface can actually only execute commands from its machine command set. These are essentially simple logical or arithmetic control structures, whose use is extremely difficult for beginners. Consequently, the PC software LLWin (Art. no. 30407, not included in the construction kit) provides graphic elements, which can then be translated into a language that the interface can use.

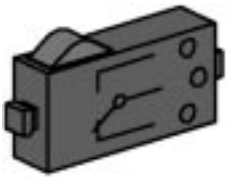
We will proceed step-by-step into the fascinating world of mobile robots. We will begin with a simple test construction to check the basic functions of interface and sensor technology. Then we will start with simple models, to which specific tasks are allocated. Later we will try out increasingly complicated systems. To ensure that any errors, which occur, do not result in permanent dissatisfaction, we will get to know the properties and special characteristics of sensors and actuators in one chapter. There is a “Troubleshooting” section at the end for very “stubborn” cases.

It is very important to take a great deal of care when setting up and starting operation of our robots. We are dealing with complex machines, whose only difference to other robot systems in use throughout the world is their comparatively small size. When we assemble electric components, we should follow the guidelines carefully and check two or three times to make certain that everything is correct. In mechanical constructions, including our own creations, we should pay a lot of attention to softunning and lack of play in gears and attachments. We should never use “force” during assembly.
Writing new programs and consequently defining new “behavior,” whose complexity is only limited by the available resources for memory and computing power, depends on our own creativity. The following examples should provide some ideas for this.

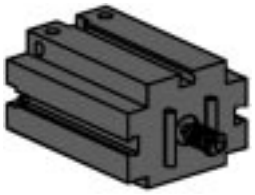
2 First Steps

Following the theoretical considerations, we now want to finally begin our own experiments. Some of you would certainly like to start immediately, maybe even with the complicated automatic forklift. This is of course possible, and if you follow the construction instructions carefully, you can construct the model right away.

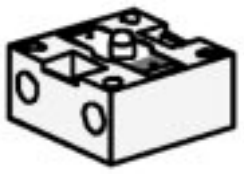
But what can you do if it doesn’t work? In such cases, you must search for the cause of the error systematically. When you do this, questions inevitably arise about the mode of operation and the properties of the components used. A certain degree of basic knowledge about sensors and actuators is apparently indispensable. But before we start to learn about these things, we should check the interaction between computer and interface.
Install the control software on your PC in line with the guidelines from the LLWin manual.



Using the interface diagnosis (Check Interface), we can test the different sensors and actuators. For example, we can connect a pushbutton with two lines at input E1 and then see which logical switching state the interface detects. When you press the pushbutton, the state must change at the corresponding input.

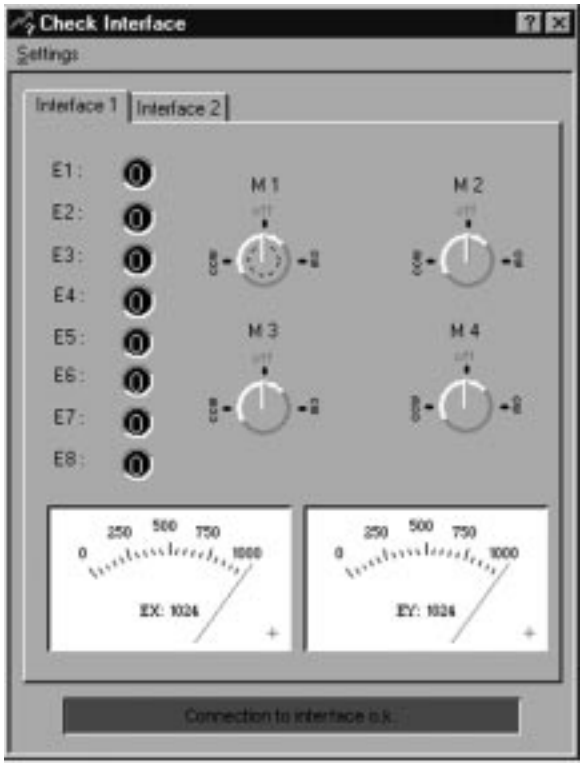


We can check the outputs by connecting a motor with a motor output, e.g., M1. Using the mouse, we can change the rotation of the motor. If we want to test the analog input, we can use a phototransistor as an analog sensor.
While the polarity does not play any role for motor or pushbuttons (the motor rotates in the reverse direction in an unfavorable case), the correct connection of the phototransistor is imperative for correct functioning. One contact of the transistor is identified with a red marking; we connect this one with a red connection plug, and the contact not marked with a green plug.



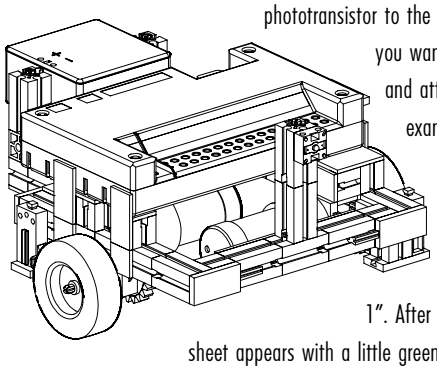
The second red plug is inserted in the socket closer to the edge of the interface of the EX input; the second green plug is inserted in the socket of EX placed further inside. Then we can vary the lighting strength of the phototransistor using a flashlight and consequently change the pointer deflection.

If the pointer does not move away from its maximum deflection, we should check the connections of the phototransistor once again. On the other hand, if the pointer points to zero when the flashlight is switched off, the lighting in the room, i.e., the ambient brightness, can be excessive. The pointer deflection changes when we cover the phototransistor.

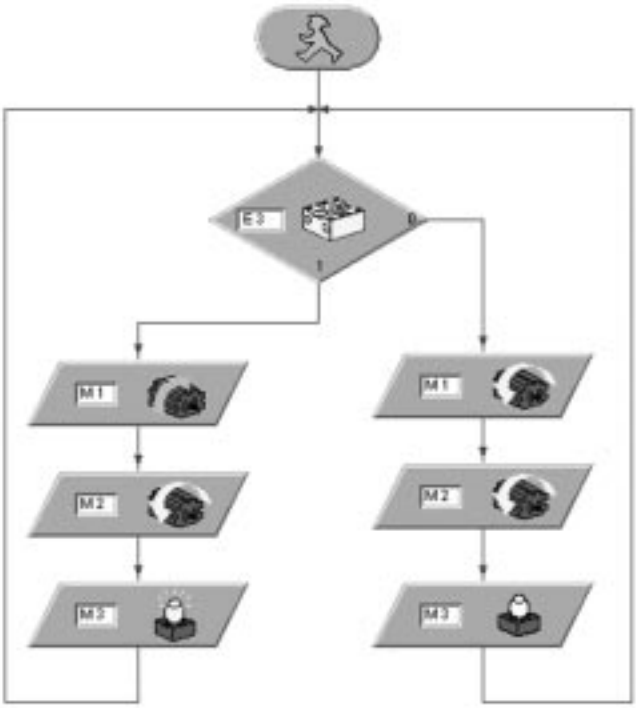


We would like to talk briefly about the color assignment of the plugs again. We should pay very close attention during assembly to ensure that a red plug is connected to a red wire and a green plug to a green wire. If we use polarized signals in circuit design, we always use a red wire for the plus pole and a green wire for the minus pole. This might seem a bit pedantic (and the electric current does not care which color a wire has), but clear and unique color assignment simplifies systematic searches for errors substantially.

We want to conclude our first step into the area of robotics with a simple program. Set up the basic model with the two drive motors and the support wheel according to the construction instructions. Only connect the motors to the outputs M1 and M2. Also take the phototransistor and connect it to input E3 (pay attention to the polarity). Before you do this, attach the phototransistor to the basic model, so that it looks "forward." If you want to, you can connect the lens lamp to M3 and attach it next to the battery pack, for example, but this is not absolutely necessary.



Open the LLWin program and create a new project (PROJECT – NEW). LLWin provides us various templates; select "Empty project" and give it a name, e.g., "Step 1". After you press the [OK] button, a blank work sheet appears with a little green man and the block window. The little green man symbolizes the program start.
All our programs begin from this starting point. We retrieve the various program parts from the block window using the mouse. The symbols there stand for the inputs and outputs on the interface. We can place the desired symbol using the mouse button on the left, and change the properties using the mouse button on the right.



The pushbutton symbol identifies an input. Place the pushbutton under the start symbol for our program. Once you release the symbol, a selection dialog appears. Select phototransistor. If you want to make changes later, you can activate this dialog using the mouse button on the right. Assign the outputs to the motors and indicate the desired rotational direction. We want the motors to rotate in the same direction when no light hits the phototransistor, and in the opposite direction when light is detected. Then link the elements using the draw function. The lamp on M3 signals the state of the phototransistor.

The drawing shows the precise link of the program branches. If you are not certain whether everything is correct, compare your program with the Step1.mdl program. Save your own program before you do this, and load the Step1.mdl file from the CD-ROM, which is contained in the construction kit.
If everything is correct, the program is downloaded into the interface and started immediately (RUN - DOWNLOAD).

The first robot then rotates on one spot. It does this until we attract him with a light source. As soon as the phototransistor detects the light, the motors, which previously rotated in opposite directions, are rotated in the same direction and the robot moves straight toward the light source. If it moves away from the light source, we must change the poles of both motors. It will probably not move in a precisely straight direction, so that the phototransistor will lose contact to the light source after part of the way. Then its movement will switch from Move forward to Rotation, and the light search starts anew. Provide sufficient space for the robot, since it can unfortunately not (yet) detect obstacles in its path.

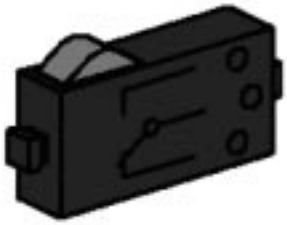
3 Sensors and Actuators

You now know that our most important unit, the interface, "plays" together with the PC. Now it is a question of how the interface can detect the signals from our environment.

Let's start with the inputs. In technical language, input signals are often simply called inputs. There is only one possibility for a computer, and this includes our Intelligent Interface, to detect and process signals. We have to make the environmental stimuli "computer-compatible." Consequently, all sensors are converters, which convert the desired "sense" into an electric signal. Because we do not want to follow the construction instructions "blindly," it makes sense to look into the basic properties of the available sensors.

This is even more important for expanding the interface later to handle new applications that you define yourself.

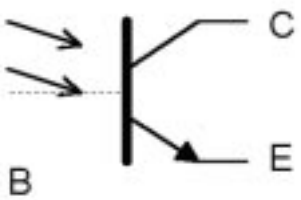
3.1 Switches as Digital Sensors

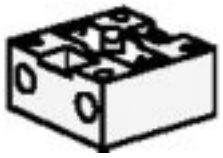


The simple, logical levels "0" and "1" can be depicted using a switch. Precision snap switches are used in the system of fischertechnik construction kits. The special property of snap switches lies in their switching behavior. If you press the red button carefully and slowly, you feel a clear pressure point when the switch contact switches over with a slight clicking sound. If we release the switch lever slowly, we have to let the lever go substantially further than the original switch on point to switch it back. This difference between mechanical switching on and off positions is called hysteresis. The switching hysteresis of contacts or other electronic switches is an important property. If they did not exist, i.e., the switching-on point would be the same as the switching-off point, substantial programs in signal assessment would result. Tiny interferences such as very slight jittering in the switching time point would result in several unintentional contact activations; it would not be possible to count events precisely. The switch is designed as a change over switch. Consequently, you can assess both imaginable starting positions, i.e., closed and open when idle, in your experiments.

3.2 Light Detection using the Phototransistor

The phototransistor is a semi-conductor element, the electric properties of which are dependent on light. A normal transistor is a component with three connections. These connections are called emitter, base and collector. Its main task is to amplify weak signals. Weak current, which flows from a signal into the base of the transistor, results in much stronger current at the collector of the transistor.





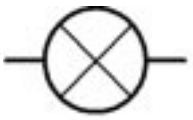
The current amplification can reach factors of more than 1,000. But the phototransistor from the construction kit only has two connections. What happened to the third connection?

We want to detect light with our transistor. Everybody is familiar with solar cells, with which power is generated using sunlight. The phototransistor should be understood as a combination of mini solar cell and transistor. The base connection is not to the outside (consequently, it appears as a dotted line in the drawing). In its place, light pulses (photons) generate a very small photocurrent, which is then available amplified from the transistor at the collector for assessment. In order for this to function as described here, the phototransistor requires additional external wiring. Because this is contained in the interface, this is not important for us.

The phototransistor can be used both as a digital sensor and an analog sensor. In the first case, it serves for detecting clear light-dark transitions, e.g., a marked line. But it can also differentiate the strength of light; then the phototransistor operates as an analog sensor.

3.3 Signal Output using an Incandescent Bulb

An incandescent bulb serves for outputting simple light signals. To put this in technical language, we will call the incandescent bulb an optic actuator. The structure of an incandescent bulb is very simple. A filament made of thin tungsten is mounted between two connection pins in a glass bulb, in which there is a vacuum. If current flows through the filament, the tungsten filament heats until it glows white. Because there is no oxygen in the glass bulb, the filament does not burn and consequently the lamp has a long operating life. Due to the strong thermal stress on the coiled filament, the wire filament expands each time the light is switched on and contracts when it is switched off. These minimal movements due to material wear result in "burning out" of the incandescent bulb at some time.



One possible use of incandescent bulbs is for displaying switching states. Warning messages can also be generated by the programming of a blinking lamp.

We also need the bulb in another case. A special sensor is created together with two phototransistors, with which lines can be detected. The bulb works as a light source, so that the phototransistor can detect color marking based on light reflected with different strengths.

A special feature of the incandescent bulb, which is used in fischertechnik construction kits, is the optic line contained in the glass bulb. This improves focusing of the light rays, and markings are detected more reliably as a result, for example.

3.4 Direct Current Motors as Power Source

Direct current motors are important actuators for mobile systems. Two different types of motors are contained in the "Mobile Robots II" construction kit. Although they differ greatly in a mechanical sense, their electric structures are identical.

Direct current motors are made of a rotating "rotor" and a fixed "stator." The rotor should be understood as a conductor loop in principle, which is in the magnetic field of the stator. If current flows through the conductor loop,

power is generated, which results in displacement of the conductor in the magnetic field. The rotor moves. The simple conductor loop is designed as a coil for practical applications (with or without iron core for amplifying the magnetic field). Very many direct current motors generate the required magnetic field using permanent magnets, which are stuck into the metal coating of the stator. Current is fed to a rotating rotor using sliding contacts. These contacts provide the current direction reversal in the conductor loop at the same time, which is necessary for uninterrupted rotational movement. The rotational speed of normal motors is in the range of a few thousand revolutions per minute. Gears provide for lower rotational speeds with large torque.

The construction kit contains two different motor types: the mini motor and the power motor. The small, compact mini motor with worm is designed for auxiliary drive or special purposes with small power requirements. It always requires gears for reducing rotational speed.

The power motor provides much larger torque. It has fixed, flanged gears with a reduction ratio of 50:1. Consequently, it is ideal for the drive requirements for our robot. But it also exists in a variant with a reduction ratio of 8:1 (not contained in this construction kit). But the rotational speed would be too great on the drive shaft for the robot drive.

3.5 Power Supply

Mobile systems require an autonomous power supply. All energy users are supplied from this energy source. The power supply requirements differ. While drive motors are content with destabilized voltages, many sensors require stable voltages to provide precise results.

For economic reasons, the use of batteries or accumulators is the only meaningful way to supply robots with current. Solar cells or fuel cells are unfortunately not sufficiently powerful to provide practical results with a reasonable amount of expense.

Accumulators are preferable to normal batteries, because they can be recharged many times. The fischertechnik Accu Set provides a good compromise between energy supply and size. The Accu Set is not a component of this construction kit. It can be purchased together with a special charger as an "Accu Set" with article number 34969. The switching symbol and the accumulator are displayed in the drawing. In a normal case, the polarity is not shown in a switching symbol. You can remember more easily which connection is plus using this simple example: "You can cut through the long line and make it a plus."

It is very important to pay attention to correct polarity when you connect voltage sources.

3.6 Additional Sensors

The fischertechnik system can be expanded relatively easily with additional sensors. In the simplest case, we can use sensors from other kits, e.g., the thermal sensor or the magnetic sensor from the "Profi Sensoric" construction kit, article no. 30491.
But we can also use completely different sensors. Very different kits and components are for sale in specialized stores. Even exotic sensors such as gas detectors or radar sensors can be used. But because we do not want to destroy the Intelligent Interface with high input voltages or incorrect loads, only experienced do-it-yourselfers should create their own solutions. The most reliable way to connect additional sensors is to separate sensor and interface galvanically. A number of sensors have a relay, which is well suited for this. The switching contacts of the relay are connected like a customary fischertechnik switch and then signal the occurrence of new environmental stimuli. Tip: Such expanded experiments are published by enthusiastic "fischer technicians" in the Internet.

4 Robot Models

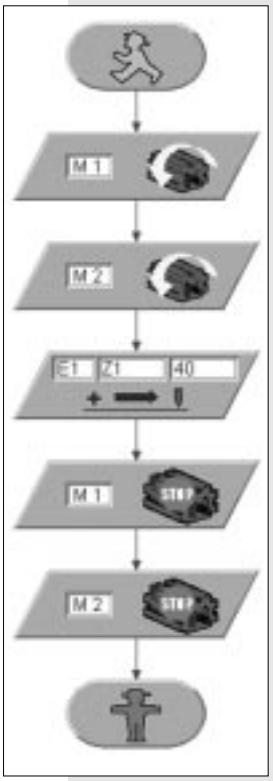
A few variants of mobile, autonomous robots are presented in the following construction suggestions. We'll start with a simple model. Based on that, you can use your imagination and try using different sensors. Here is it a question of linking both the internal states of the robot, e.g., distance measurement by impetus wheels, and the external environmental signals such as light or searching for lanes. Then specific tasks are set for each model. They are designed to stimulate your imagination and make you familiar with the material. The LLWin programs for the individual tasks are on the CD-ROM, which is included in the construction kit. But try to think up your own tasks for the models. The simplest model is the basic model. The drive motors are assembled with the interface to create a compact unit with it. Two motors provide the drive power of the robot. They are arranged opposite each other, so that each motor runs a drive wheel. A support wheel provides stability, so that this robot does not tip over. Such an arrangement of the motors is called differential drive. It provides the highest degree of mobility with the minimally required movement space. Turning in place is even possible. But the center point of both motors is the center of rotation around which the robot moves. In this way, it is able to navigate in the most difficult situations with little computing work.
The motors can drive the wheels via two different gear reductions (slow 100:1 or fast 50:1). For the slower variant, the drive is reduced additionally at a ratio of 2:1 using fischertechnik toothed gears. Which gear reduction is used is indicated for the models.

4.1 The Basic Model

Let's build the basic model (gear reduction 100:1) first in accordance with the construction instructions. Because this model serves as a basis for many experiments, proceed very carefully when assembling it. After all mechanical components have been assembled, check the soft running of the motors. Each motor is connected directly with the accumulator without the interface for this.

Task 1:

Program the interface, so that the 40 Pulse model moves straight ahead. Use the counter button E1 to measure the pulses; use the E8 button as reset switch. Then modify the program, so that the model moves differently long distances, e.g., 80 cm. How high is the repetition accuracy?



Task 2:

The model should turn 180° after 80 pulses of moving straight ahead. Note the different rotational directions of the drive motors during the straight-ahead movement and the turn.

Solution:

The model moves an average of 1 cm on its path per counted pulse. The repetition accuracy is in the same range, approximately 1 cm with 80 pulses. It fluctuates dependent on the surface on which the robot moves. Thick or fluffy floor coverings are especially unfavorable for accurate measurements.

Before we deal with this task, we want to clarify two things. First, we are using a new function block in our program, the POSITION block. This is a block, which remains active until the set number of pulses is detected at a fixed input (E1 here). From the viewpoint of the program, this means that we are using a defined delay condition here. In the first experiment, we used this function as distance measurement for moving straight ahead.
If the robot should turn, practically the same procedure is used; we only need to change the rotational direction of the motors. Now you only need to enter the number of pulses, and the robot rotates on the spot.
And now to the second point. You don't want to simply try out values until the robot turns 180°, but instead you want to calculate this value in advance.

The drive motors are configured as differential drive, i.e., the wheels of the robot move around the circumference of a circle, the diameter of which is determined by the distance between the wheels. Consequently, each wheel must travel the distance of exactly half of this circumference for a turn of 180°.

Calculate the circumference u first:

$$u = \pi \cdot d = 630mm$$

d: diameter (wheel distance approx. 200 mm)

We previously determined a distance of approx. 1 cm/pulse. As a result, we need 305 pulses for the 314 mm distance (half of the circumference). Because we can only calculate whole number values, we have to select either 30 or 31 pulses. Test which value provides greater accuracy.

Conclusion:

You see that the result of our measurements with the pulse wheel does not provide a very high degree of accuracy. The absolute measurement error becomes greater especially when several distances are traveled one after another or repeatedly. The error caused by the clock pulse, which was not entered precisely, also results in problems.
We only have limited possibilities for minimizing these errors. On one hand, you can increase the path pulses per unit of distance. The counter would ideally be mounted directly on the motor shaft. In addition to the fact that we cannot access this shaft, there is also the problem of the limited sampling rate of the interface. If too many pulses are received within a time unit, the interface might "forget" a few. Then precise path calculation becomes an illusion.
We cannot register other errors at all, such as the slippage of the wheels on different surfaces or deviating wheel diameter. We can console ourselves with the thought that these problems have in part not been solved satisfactorily by substantially more complex and expensive commercial systems either.

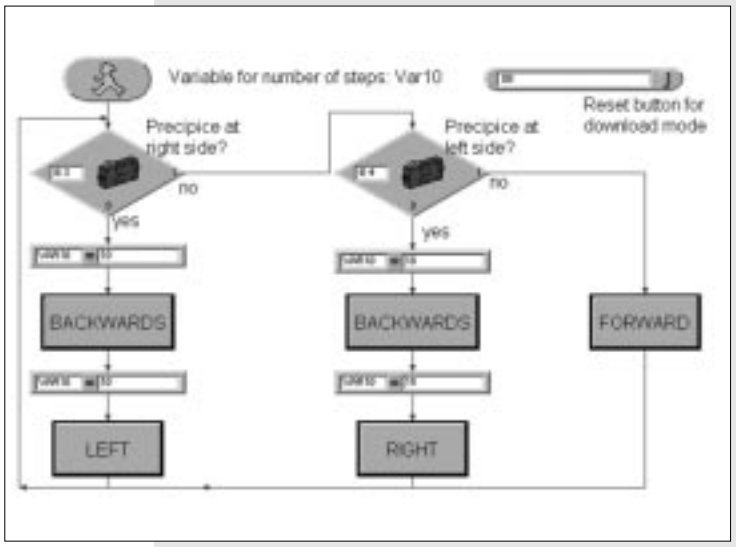
4.2 Robot with Edge Detection

Now that we have experimented extensively with our basic model, we want to now try to teach our robot "fear" of sheer drops. We previously watched the robot like a hawk as it scurried around a tabletop, so that it did not plunge off the edge. This really does not seem to represent especially intelligent behavior. Consequently, we want to change it.
The robot needs an edge detector for this. Two auxiliary wheels provide a simple and useful procedure. The wheels are equipped with a switch, similar to a sensor, in front of the robot's direction of movement. The wheels are designed, so that they can move vertically. An edge lets the auxiliary wheel fall downward and consequently triggers the sensor.

Task 3:

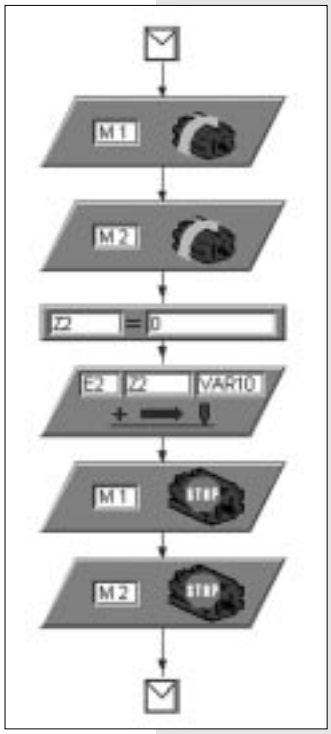
Build the "Robot with Edge Detection" model in accordance with the construction instructions (gear reduction 50:1). The model should move straight ahead. As soon as it reaches a drop on the left, it should avoid this by moving to the right; if there is a drop on the right, it should move to the left. To make this clearer, specific movements are programmed as subprograms (forward, left and right).
The number of steps is recorded by the counter button. The number of steps is set in a variable VAR10. This value differs for the left and right subprograms.

The different values reduce the risk of getting stuck in a corner.



Solution:

We can deduce from the task that we have to control the robot dependent on the edge detectors. Consequently, we should first split up the task into smaller parts. We first query the sensors (edge detectors). If no sensor is active, the robot moves forward. This is the "Forward" block in the diagram. A subprogram is behind such a block, which is new to us. Subprograms improve the clarity of complex systems, and they can also take better advantage of your computer's performance with multiple use. Our first subprogram is very simple; it only triggers movement of the motors M1 and M2.



But we need additional subprograms. The robot should move left, right or backward to avoid falling depending on the situation. In this case, it no longer suffices to simply switch on the motors. A specific sequence of movements must be programmed. Let's take a look at another subprogram. This should cause the robot to move backward when it detects an edge.

We want to switch the motors to backward movement for this. Then our pulse wheel should detect a defined distance. Set the distance with the variable VAR10. The assignment block Z2 = 0 is new. After you think about it for a while, you will understand the sense of this. If you do not set the count variable to zero, the mechanism only works once, because when Z2 reaches the value of the variable VAR10, our distance counter would fail us in each subsequent run. From the viewpoint of professional programmers, we are dealing with local and global variables here. The local variable Z2 is initialized before each use in the subprogram.

Conclusion:

Subprogram calls increase the clarity of programs. We use variables to measure various values, in this case path distances. We need the different path distances, so that our robot can "free" itself from corners. If the paths were absolutely identical, a robot could always move back and forth in a corner.

When you use variables, pay attention to their validity range. We initialize local variables, i.e., variables that are only used within a subprogram, before their use.

We have also seen that our robot needs a certain amount of movement leeway for it to function properly. If it runs into an edge during a movement to avoid another edge, the robot cannot react to it. Those of you who really enjoy solving such puzzles can try to find a solution for this.

4.3 Robot with Obstacle Detection

Our robots can now detect edges very well. But there is still a problem with ordinary obstacles. We have to modify the principle of edge detection. A corresponding crash contact replaces the auxiliary wheels. At this time, you should also think about one shortcoming of the edge detector; there are dangerous situations when the robot can "blindly" plunge into the abyss during backward movement without a sensor in the back.

A third sensor can compensate for this shortcoming. In the meantime, we have become experienced programmers. Fortunately, because the task is more complicated this time, as a look at the program shows.

The program contains additional, new function blocks. We add a WAIT block at specific program positions. This is easy to understand: the program waits until the time entered here has finished before the next program function is processed.

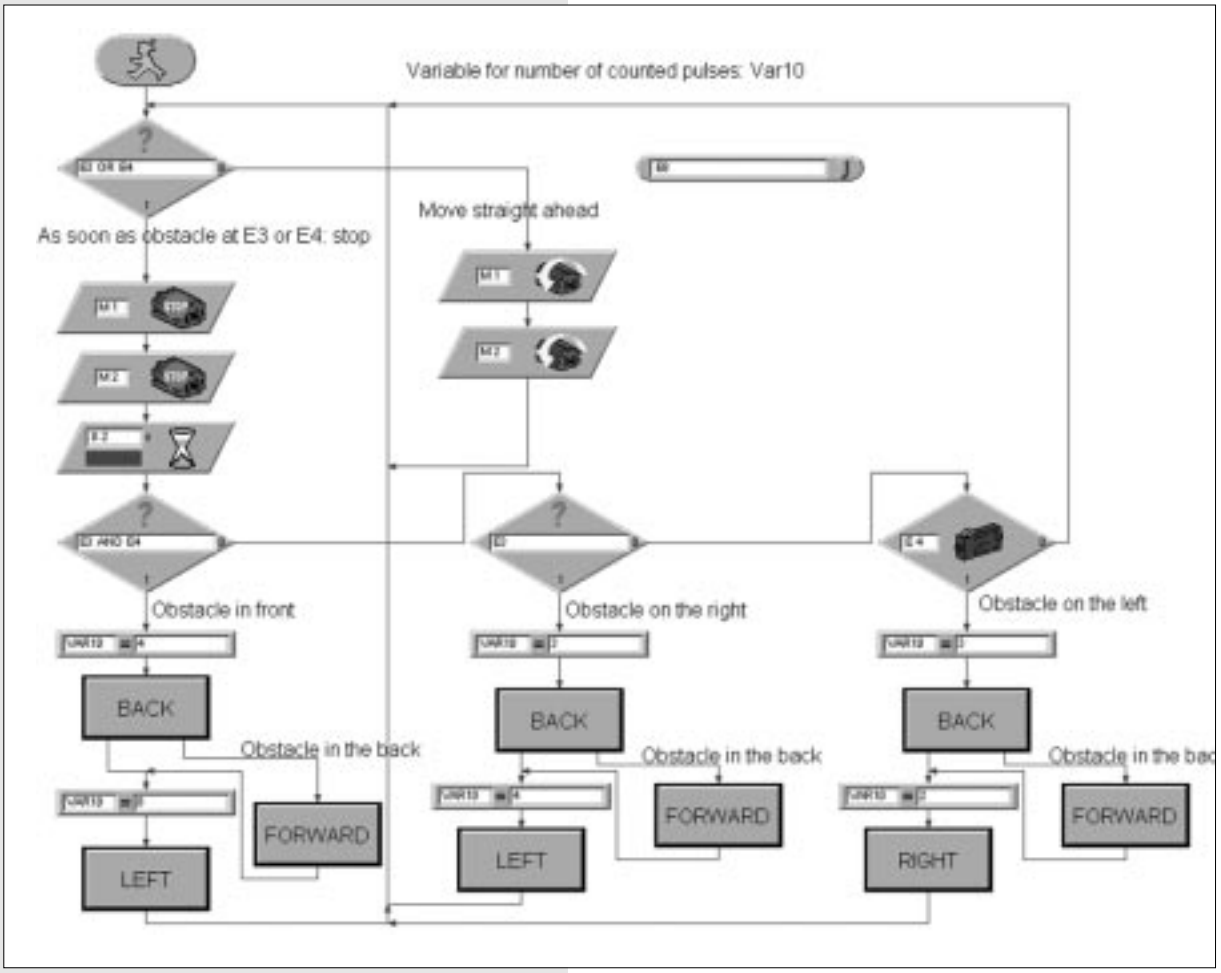
Logical comparisons are really new. Up to now, we have always made a comparison with the sensor queries and put a corresponding branch in our program. A similar comparison exists for pulse counting: comparison with a specific number of pulses. The logical comparison with several expressions is also new in a COMPARE block

Task 4:

Assembly the robot as described in the construction instructions with the faster gear reduction 50:1. The model should move straight ahead. As soon as an obstacle is detected at one of the front sensors (E3 or E4), the robot stops. If an obstacle is detected on the right, the robot moves back and then to the left to avoid it (approx. 30°). If there is an obstacle on the left, the robot moves back and then to the right (approx. 45°). The unequal number of degrees is necessary, so that the robot can find its way out of a corner.

If an obstacle is detected directly in front, the robot should move back and turn 90°. If the robot detects an obstacle when moving back, it should move a short distance forward and then avoid the obstacle as in the other cases.

Solution:



Conclusion:

The complexity of the programs is increasing. We'll try to deal with this increased difficulty with improved programming techniques. Subprograms are a good means for doing this. We are now familiar with logical comparisons with subsequent program branching as a new control structure. We also now know that an increasing number of sensors are required for new properties or improved functioning of experiments, which we have already conducted.

4.4 The Light Finder

Till now we have let the robot react rather passively to environmental signals. Now we want to send the robot out to find things actively. The construction kit contains two phototransistors, which we can use as light detectors. Each sensor acts on one motor in order for "light ray tracking" to be possible.

The program is composed of two parts. One part contains the search for a light source, and the other part handles the tracking or controls movement in the direction of the light source.

Of course, we again take advantage of the possibilities of the subprogram technique. The LIGHT SEARCH subprogram is activated after the program is started. This subprogram is only exited after a light source has been located.

The main program tries to control the robot to move toward the light source. Whenever the direction of the robot deviates greatly from the ideal line, one of the light sensors is no longer receives light from the light source. Then the corresponding motor is stopped for a brief time, so that the two sensors can again detect the light source. This gives us our task.

Task 5:

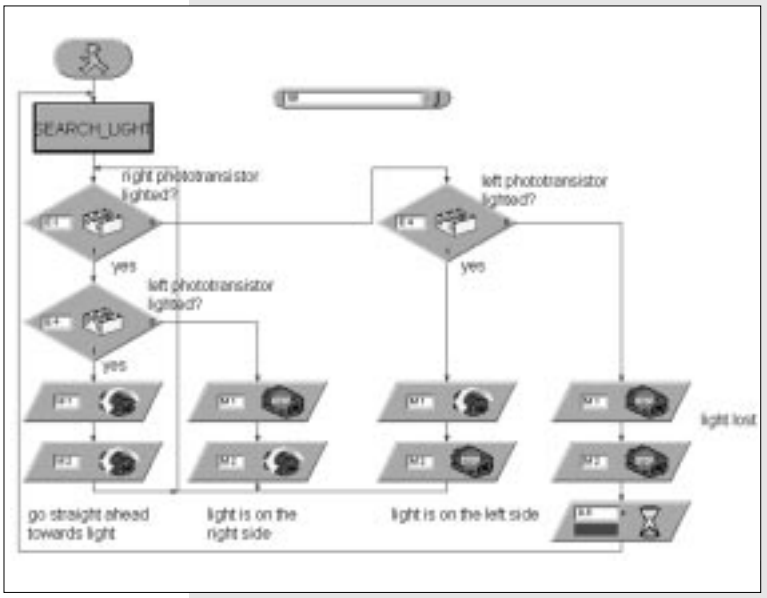
Construct the Light Finder model with the slower gear reduction 100:1. First program the "Find light" function (preferably as a subprogram from the start). The robot rotates in this by at least 360° in one direction. Then it rotates by at least 360° in the other direction. If a light is found during the search, the robot stops. If no light is found after the rotations, the search is aborted and the program ends.

If the search for light is successful, the model should move toward the light source. If the light source moves to the left or right, the robot follows the movement of the light. If it loses contact to the light source, the program starts to search for light again.

Try to attract the robot with a flashlight and see if you can guide it through an obstacle course.

Tip:
Use a flashlight as a light source. Try not to focus the light beam too small, so that both photosensors can receive the light source. Note that other light sources in bright rooms, e.g., sunlight from a window, can be stronger than the light from your flashlight. The robot might move past your flashlight and toward brighter light.

Solution:



Conclusion:
We have now constructed a robot, which detects its environment actively. Its sensors are programmed to locate a light source and – if it succeeds – to move toward or follow this source.
We saw that the program switched off the robot if no light source was found. For example, if only a small obstacle interrupts the direct line of view to the robot, it does not detect the light source and cannot find it, although it is there. It would apparently make sense to have the robot move more or less randomly to another spot after an unsuccessful light search, so that it could search for light again there.
But what happens if an obstacle, which prevents light from reaching the robot, is precisely in the robot's movement of direction? This question seems sufficiently interesting for us to examine it more closely.

4.5 The Tracker

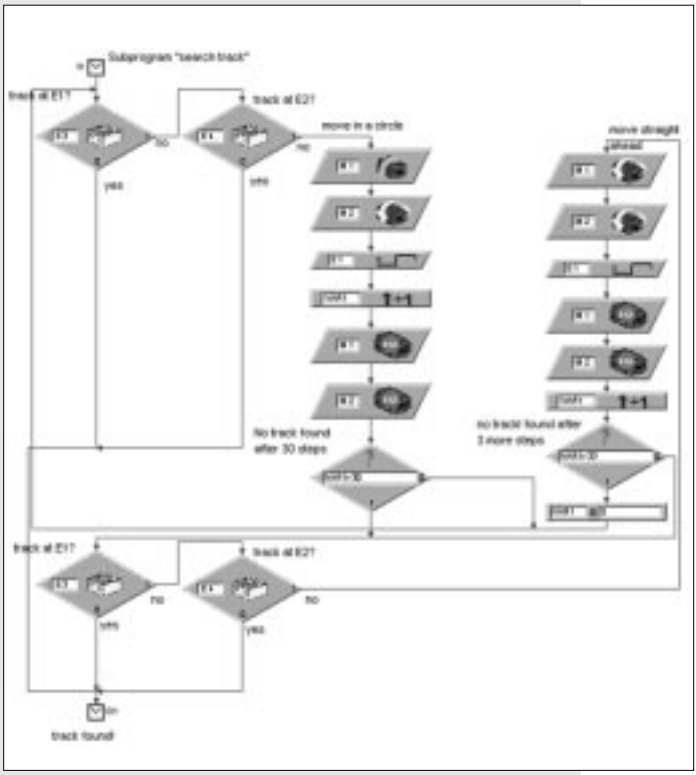
Searching and following are two essential characteristics, which intelligent beings have. We built and programmed a robot, which reacted to direct signals from its target or potential victim.
We use a different search principle with the tracker. Instead of targeted, precise movement to a light source, we mark a line that the robot should follow. This task can be solved relatively easily using optic sensors.

We measure the reflected light of the marking and then adjust the motors. You should also light the line with your flashlight, so that this functions precisely. Make certain that the photosensors are not dazzled by stray light due to unfavorable arrangement of lamps and photosensors. Concentration of light by the optical lens of our incandescent bulb works very favorably in this context.

Task 6:
Build the Tracker model (gear reduction 100:1). First write a subprogram, with which the track is sought. The robot rotates 360° in a circle for this. If no track is found, the robot moves a bit straight ahead and searches again. The photosensors are queried about track detection. If the robot finds the track, it follows it. If the track ends or the robot loses it, e.g., if the track changes direction too suddenly, the search begins anew.

Tip:
After the lamp is switched on, there must be a short delay time (approx. one second) before the phototransistors are queried. Otherwise, the phototransistor detects "dark," i.e., a track where there is none. Use an approx. 20 mm wide strip of black insulating tape for the track or draw a black line in this width on a sheet of white paper using a felt pen.

Solution:

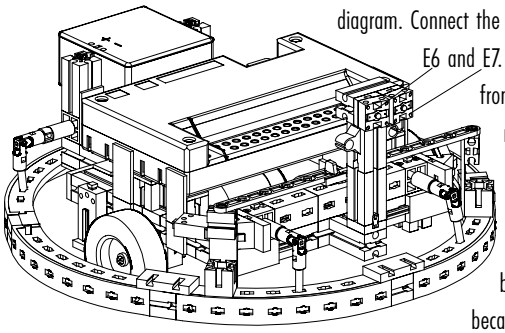


Conclusion:

If the robot finds the track, it follows it tirelessly. If we create the track as a closed loop, the robot goes round it. But you must avoid too sharp curves, where the photosensors could lose contact to the line. Although the robot tries to locate the track again, we have to admit that this is only possible where there are no obstacles.

4.6 The Electronic Moth

The problem of linking different modes of behavior such as searching, following and avoiding has already arisen several times. Consequently, we want to investigate the interaction of such behavior in another experiment. Let's summarize our previous results. The simple light finder follows a flashlight held in front of it more or less blindly. The robot assumes that no obstacle can exist where the flashlight previously was. It also assumes that it does not really reach the light. But these assumptions only correspond to reality in exceptional cases. We are forced to guide the robot with the flashlight around each obstacle. We actually would need a robot, which can avoid obstacles on its own. To do this, we can supplement the "Robot with Obstacle Detection" model with the property of light search, as shown in the diagram. Connect the phototransistors to E6 and E7. We use everything else from the obstacle detection model. From a scientific point of view, we have equipped the robot with two behavior modes. But because both behavior patterns cannot be active simultaneously, they are assigned different priorities. In this case, the robot is normally searching for light. If an obstacle is detected, which is more or less a danger for the robot, the behavior mode of obstacle avoidance becomes active. Once there is no longer an obstacle, the robot can again search for the light source.



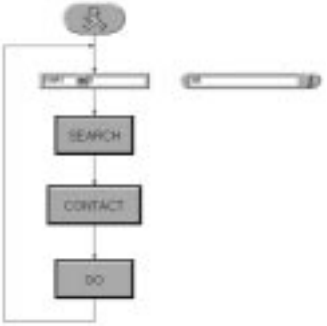
As a result, we have provided the robot with the essential capability to navigate independently and avoid dangers. If you have a friend, who also owns a fischertechnik construction kit, you can carry the experiment even further. A light source is mounted an each of the two robots, and then the robots search for each other.

We have worked according to the principle of "trial and error" until now. You probably "simply" started with your programs and have discovered during the course of experiments how the robot did what it was supposed to (or how it didn't do that).

Of course, professional software developers cannot use such design methods for their products. In such cases, a precise design strategy must be defined before the first program line is written. That is not something that you can simply do, but instead you use fixed guidelines. Procedures have become standard, which use such strange names as "top-down design." An attempt is made to define the overall system from the top down, without dealing with all details from the start. Let's try to program our moth according to the top-down design. To do this, we have to begin with the main program, called the "main loop."

The diagram shows the simple design. The main loop is composed simply of the behavior modes necessary for the robot: SEARCH, CONTACT and GO. The variable VAR1 seems strange to us, and how can be give priority to the behavior modes of the robot with such a simple loop?

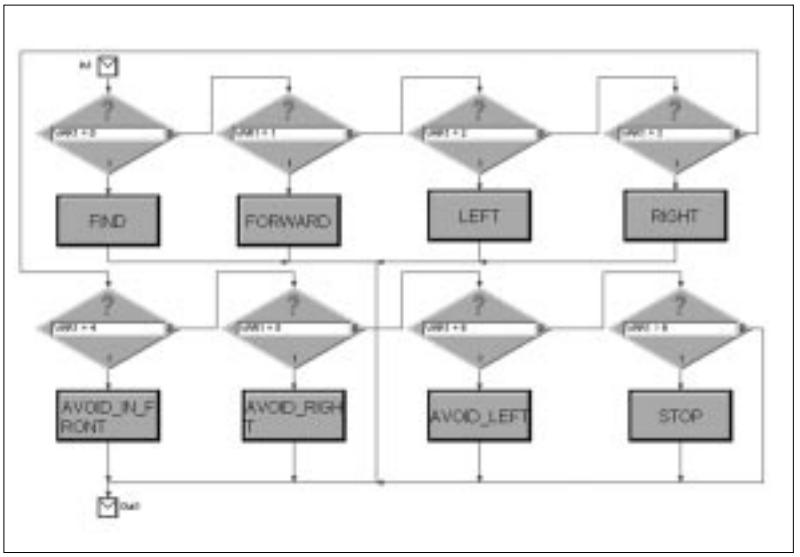
Well, priority is assigned by the sequence of the subprograms, and the variable VAR1 serves for assigning the movement orders. If we think this over very carefully, we can reduce the necessary drive movements of the robot to a clearly defined number of maneuvers. The robot needs a search movement, an avoidance movement and a correction movement. Avoidance and correction movements must be distinguished according to directions: right or left. This makes it clear that the SEARCH subprogram can calculate one of n (n = total number of movements) possible movements. The same applies to the CONTACT subprogram. Because CONTACT is executed after SEARCH, it overwrites the instructions of SEARCH. The GO subprogram only executes what it is commanded to do. The top-down design procedure really seems to be useful. But we don't know what the subprograms look like. Let's start with the first subprogram, SEARCH. This subprogram is responsible for querying the photosensors. Depending on the two sensors, four possible variants exist: left sensor, right sensor, left and right sensors, and no sensor. These define exactly four possible drive maneuvers: FIND, FORWARD, LEFT and RIGHT: These drive maneuvers are also assigned to subprograms; we are proceeding strictly according to top-down design criteria.



The design of the first subprogram, SEARCH, is now clear. SEARCH provides a parameter, which is converted into GO.

The actual subprogram is very simple. The output parameter is set via several comparisons. A check is conducted in the CONTACT subprogram in the finished program to determine whether obstacles triggered the sensor on the robot. For simplicity's sake, we will ignore this program now and take a look at how the motor movements are activated in GO.

Surprisingly, this subprogram calls additional subprograms. Doesn't this ever



end? We are still in the design phase (next level of top-down). GO sets when which drive maneuver is started.
"Real" program code is only hidden behind the various subprograms such as FIND and FORWARD. We have finally reached the bottom. Now the motors are switched on and off. Each of these subprograms has a clearly defined task, which we can program with more of an overview.

Conclusion:

Complex programs require completely new types of solution strategies. It makes sense to aim for an overall solution strategy. We wrote a program according to the strategy of top-down design. The required solution approaches are (first) formulated in formal structures, e.g., SEARCH or GO. These approaches are only refined in further steps and filled with the actual program code at the end. As a result, we separate program structure from the actual program code. We can consider both separately and analyze errors separately too, which is very important.

4.7 FTS – Driverless Transport System

Let's leave the area of scientific experiments now, which malicious gossip refers to as playing around, and deal with the area of practical applications. We now want to build and program a robot not just for its own sake, but instead it should handle a task. The tracker is equipped with a mobile transport fork for this. This creates a forklift robot, which can transport material stored on a pallet.

As simple as this experiment seems, it contains all the components of an industrial application. And such transport systems are already in use in part today. In the meantime, you have had good training in programming, so that you certainly are willing to take on this complex programming task in good spirits.

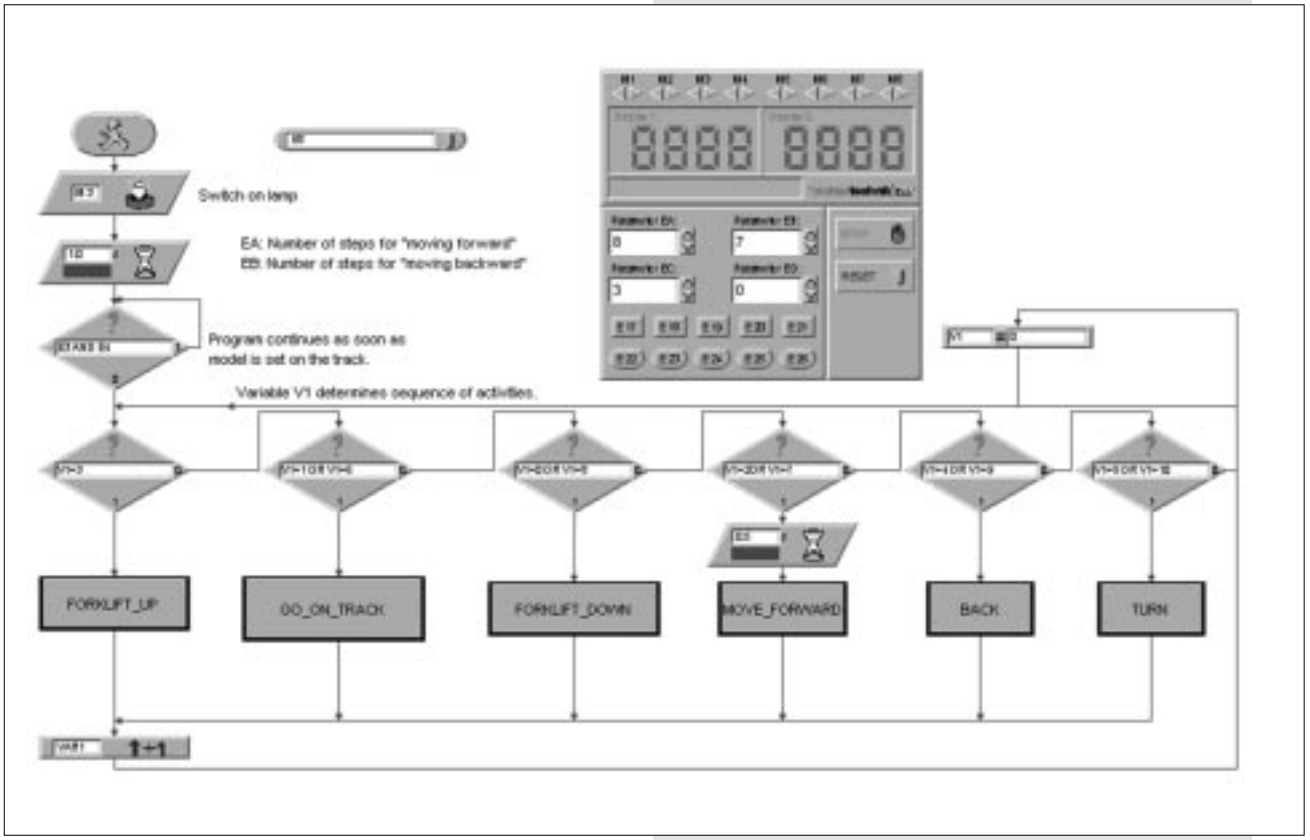
Task 7:

The driverless transport system should drive along a black track. At the end of the track, it picks up a pallet, turns 180° and drives back along the track. When the end of the track is detected, the robot puts the pallet down, turns again and gets the next pallet.

Task 8:

As an extension to Task 7, supplement the program so that the pallet is only put down for a short time at the end points of the track. In other words, the robot searches for the pallet, drives to the end of the track, puts the pallet down for a short time, picks it up again, drives to the other end, etc.

Solution:



Conclusion:

As soon as we take at look at the program, we immediately recognize the already known program strategy. A state variable controls the behavior of our driverless transporter.

Using the refined programming techniques you have acquired, you can implement multifaceted controls. The driverless transport system demonstrates the possibilities, which the combination of fischertechnik construction kit with the Intelligent Interface provides. Starting with simple mobile robots, you have reached a state, which hardly differs from industrial control engineering.

5 Troubleshooting

Experimenting is fun, but only as long as everything functions well. The best would be if everything succeeded at first try. Unfortunately, this is usually not the case.

You only know whether you have understood a mechanism precisely if a model does not work and you find the error immediately.

With mechanical errors, you can still see (assembled incorrectly) or feel (runs hard) something. When there are electric problems in addition, it becomes more difficult.

Professionals use a number of very different measurement instruments for troubleshooting (searching for errors), for example, voltmeters or oscillographs. Not everyone has such equipment on hand. Consequently, we want to try to pinpoint errors and correct them using simple means.

Before we start with the experiment, you have to put together a few components from the fischertechnik construction kit. These are essentially cable connections. The supplied plugs are connected to the individual cable sections here.

First, you cut the cable. Measure the given lengths and cut the sections. Before you cut, check very carefully to make sure the lengths are correct.

Then you need to test each cable. You need the accumulator and the lamp for this. If the lamp lights after it is connected with the accumulator, the cable is okay. If the color assignment is also correct – red plug with red cable and green plug with green cable – put the cable aside and check the next one.

If the program (including the one supplied) does not work with our model, start "Check Interface". This auxiliary program lets us test inputs and outputs separately. Each sensor must trigger the corresponding action on the interface here.

After you check that, you know that all the electric connections should be okay. Switch the actuators on and off individually via the motor control buttons. If everything works here too, look for the mechanical cause.

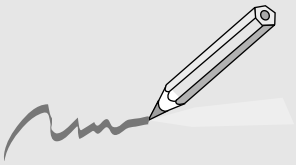
Loose connections represent bad errors. On one hand, connection plugs can be loose in the sockets. If this is the case, the contact springs of the plug are widened a bit using a small clockmaker screwdriver. Be careful, because widening them too much can result in breaking the contacts or make it very difficult to plug them in.

Another cause of loose connections is created by loose terminal connections on the screw-in spots of the plug. Screw them tight carefully! You can use this occasion to check whether any of the thin cooper wires have broken off.

If there is an unexplainable outage during operation, an almost empty accumulator can be the cause. The voltage decreases briefly when a load is switched (motor on), and then a reset is triggered for the computer on the interface. Such an error is difficult to find, because the program always functions correctly at first.

If errors occur in programs you have written yourself, which you cannot explain, install a supplied program, which is as similar as possible to yours, to rule out electric or mechanical defects.

If you are still not successful in finding the error, you can always contact the fischertechnik service department.



(F) SOMMAIRE

F

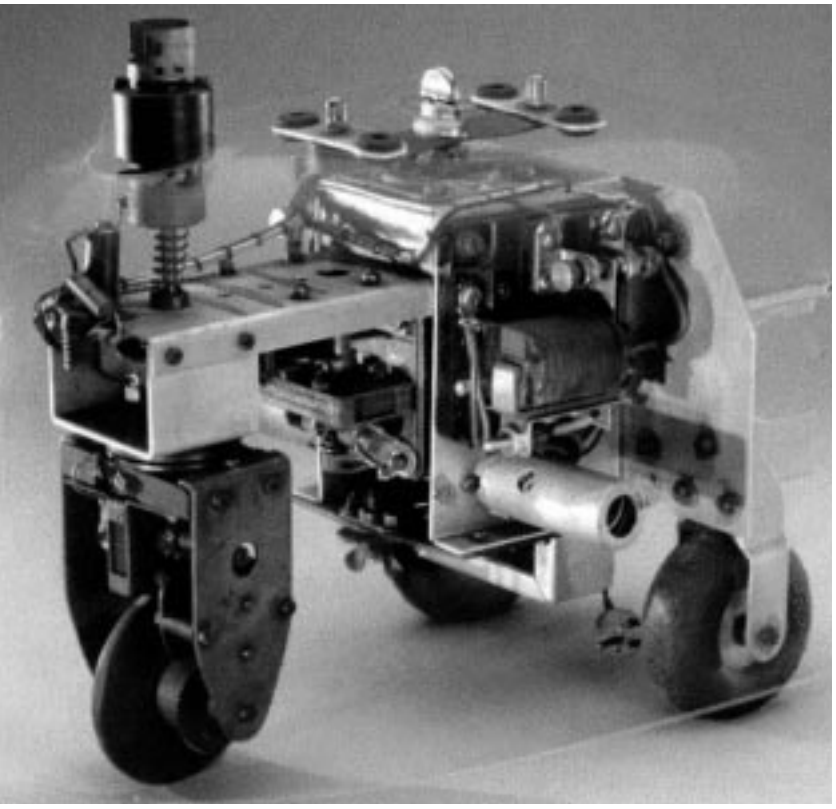
1 Pourquoi avons-nous besoin de robots ?

Avant de nous lancer de façon pratique dans l'ingénierie des robots, nous allons essayer de répondre à la question quelque peu provocatrice posée en titre. Le concept de « Robot » fut utilisé pour la première fois en 1923 dans le roman « Golem » de Karel Capek. Ce sombre personnage créé de façon artificielle était censé posséder des facultés lui permettant de remplacer le travail humain.

Comme c'est souvent le cas avec les personnages littéraires, ces facultés étaient également liées à des contraintes et font naître une certaine méfiance envers le personnage. Dans les années 30 et 40 du 20e siècle, le robot devient plutôt une sorte d'automate. Diverses tentatives de le munir de traits extérieurs humains tels qu'une tête avec des lampes qui clignotent en guise d'yeux ainsi que d'une parole primitive par haut-parleurs ont l'air naïves dans l'optique d'aujourd'hui. De toute évidence, les craintes d'une domination possible des robots sur les hommes ne sont pas si faciles à dissiper.

Cependant, lors de ces premières tentatives élémentaires, on ne remarque rien d'une mobilité ou même d'une intelligence de ces machines construites de toute pièce. C'est seulement avec l'apparition des circuits électroniques que la construction des robots devint réaliste.

L'ingénierie des robots proprement dite est étroitement liée au problème des principes nécessaires à leur commande. Cette question relative à « l'intelligence » des robots est aujourd'hui encore l'objet d'études et de recherches dans de nombreux instituts, firmes et universités.



Les premières solutions, espérait-on, seraient fournies par la cybernétique. Le terme « cybernétique » provient du mot grec cubernêtês. Le cubernêtês était le navigateur, celui qui tenait le « gouvernail », sur les bateaux à rames grecs. Il devait faire le point et calculer le cap à suivre pour parvenir à destination.

Il est donc clair que la cybernétique devait rendre le robot « intelligent ». Mais comment peut-on se représenter d'une façon générale un tel comportement intelligent ? Nous allons essayer de l'illustrer à l'aide d'une expérience mentale. Chacun de nous a déjà observé le comportement d'une mite dans la sphère lumineuse d'une lampe. La mite détecte la source de lumière, vole dans sa direction, et évite la lampe juste avant de la percuter. Il est clair que, pour ce comportement, la mite doit détecter la source de lumière, déterminer un chemin qui y mène, puis voler dans cette direction. Ces facultés sont basées sur des schémas de comportement instinctifs et intelligents de l'insecte.

Essayons à présent de transférer ces facultés sur un système technique. Nous devons percevoir la source de lumière (capteurs photoélectriques), effectuer un déplacement (commander des moteurs) et devons enfin établir un lien judicieux entre la perception et le déplacement (le programme). Notre expérience mentale combine alors un capteur photoélectrique avec un moteur et une logique de façon que ce véhicule pilote toujours le moteur dans la direction de la source de lumière. Ce véhicule se comporterait donc exactement comme une mite, n'est-ce pas ?

Une réalisation technique de l'expérience décrite précédemment fut effectuée dans les années 50 par le britannique Walter Grey. À l'aide de capteurs, moteurs et circuits électroni-ques simples, il créa différents animaux « cybernétiques » possédant alors un comportement tout à fait spécifique tel que, par exemple, celui d'une mite.

Ces machines représentent une étape essentielle vers les robots mobiles modernes. Les capteurs (photorésistances, palpeurs, ...) des appareils pilotaient les actionneurs (moteurs, relais, lampes, ...) à l'aide de leur électronique et produisaient ainsi un comportement (apparemment ?) intelligent. On peut voir sur l'illustration une copie de la tortue « cybernétique » exposée au musée Smithson de Washington.

Sur la base de ces considérations, nous allons élaborer pour nos robots des « schémas de comportement » correspondants et essayerons de les rendre compréhensibles au robot sous la forme de programmes.

Néanmoins, comment peut-on avec ces considérations répondre à la question posée au début en ce qui concerne l'utilité des robots mobiles ? Pour répondre à cette question de façon concrète, essayons à présent d'appliquer les comportements plutôt abstraits de notre « mite mentale » à des aspects techniques. Un exemple simple en est la recherche de lumière. Modifions la source de lumière en apposant une bande claire, la ligne directrice, sur le sol et en dirigeant les capteurs non plus vers l'avant, mais vers le bas. Une telle ligne directrice permet à un robot mobile de se déplacer par exemple dans un hall d'entreposage. Des informations supplémentaires, p. ex. sous la forme de codes barres se trouvant à certains endroits de la ligne, incitent le robot à exécuter d'autres actions à ces endroits, comme p. ex. le prélèvement ou la dépose d'une palette.

De tels systèmes à robots existent déjà dans la réalité. Dans de grands hôpitaux, il faut parcourir des trajets parfois très longs pour transporter du matériel d'usage tel que p. ex. le linge de lit. Le transport de ce matériel par le personnel infirmier est compliqué et parfois lié à un travail physique fatiguant. En outre, de telles tâches limitent le temps que ce personnel peut consacrer aux soins à donner aux patients. On s'aperçoit donc que les robots mobiles peuvent occuper une place importante dans la société moderne.

Cependant, quel est le rapport avec les coffrets fischertechnik ?

Pour un robot, nous aurons besoin de nombreuses pièces mécaniques, en plus des capteurs et des actionneurs, pour construire une maquette. Dans ce

but, le coffret fischertechnik 'Mobile Robots II' constitue donc une base idéale. Nous pourrons combiner les pièces mécaniques les unes avec les autres d'une variété de façons pratiquement infinie, et obtiendrons ainsi de robustes véhicules-robots. Grâce à « l'Intelligent Interface » (Référence 30402, nécessaire en plus), nous disposons également d'une capacité de calcul suffisante pour élaborer des programmes ambitieux. C'est par l'intermédiaire de cette interface que se fera la connexion et l'analyse d'une multitude de différents capteurs et actionneurs.

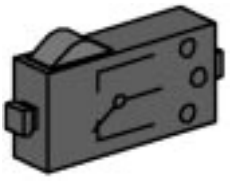
Les capteurs transforment des grandeurs physiques telles que quantité de lumière et température en des valeurs mesurables électriquement. Il s'agit aussi bien de grandeurs analogiques que de grandeurs numériques. Les grandeurs numériques sont celles qui peuvent être logiquement vraies ou logiquement fausses. Ces états sont identifiés par 0 ou par 1. Un interrupteur est un bon exemple de capteur numérique.
De nombreuses grandeurs varient cependant de façon continue entre leurs valeurs extrêmes, on les appelle des valeurs analogiques. Elles ne peuvent pas se représenter simplement par 0 ou par 1. Pour que ces grandeurs puissent être traitées par un ordinateur, il faut les transformer en des valeurs numériques correspondantes. À cet effet, l'interface fischertechnik fournit deux entrées analogiques EX et EY. La valeur de résistance appliquée à ces bornes est stockée sous forme de valeur numérique. Les valeurs mesurées par un capteur de température, p. ex. 0 à 5 kΩ, font ainsi l'objet d'une saisie dans une plage allant de 0 à 1024 et sont disponibles pour un traitement ultérieur.

La fonction essentielle de l'interface consiste à relier logiquement les grandeurs d'entrée. Pour cela, l'interface a besoin d'un programme. Ce programme décide de quelle façon les grandeurs d'entrée, les signaux des capteurs, doivent produire des valeurs de sortie, c'est-à-dire des signaux de commande pour moteurs, etc.
Pour pouvoir élaborer de façon aussi efficiente que possible les programmes nécessaires à l'interface, il existe une fiche de travail graphique. Derrière le concept de « fiche de travail » se cache un logiciel qui nous permet d'élaborer nos programmes à un très haut niveau. Nous pouvons en effet élaborer des programmes ou des algorithmes à l'aide de symboles graphiques. L'ordinateur de l'Intelligent Interface ne peut en fait exécuter que des instructions se trouvant dans ce qu'on appelle son jeu d'instructions machine. Ce sont essentiellement des structures de contrôle logiques ou arithmétiques simples dont l'utilisation est extrêmement difficile pour les débutants. Le logiciel d'ordinateur individuel LLWin (Référence 30407, non inclus dans le coffret) fournit par conséquent des éléments graphiques qui sont ensuite traduits dans un langage exécutable par l'interface.

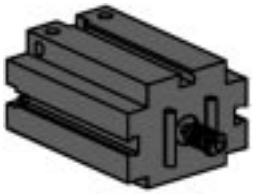
Nous procéderons pas à pas lors de notre incursion dans le monde fascinant des robots mobiles. Nous commencerons par une structure d'essai simple destinée à tester les fonctions de base de l'interface et des capteurs. Nous lancerons ensuite des maquettes simples devant exécuter des tâches déterminées, et nous essayerons enfin à des systèmes de plus en plus compliqués. Pour que les erreurs qui apparaîtraient ne conduisent pas à une contrariété permanente, nous consacrerons un chapitre à nous familiariser avec les propriétés et particularités des capteurs et actionneurs, et pour les cas vraiment « tenaces », on trouvera à la fin une section « Dépistage et élimination des défauts ».
Un point très important est le soin à apporter lors de la construction et de la mise en marche de nos robots. Nous avons affaire à des machines complexes dont la seule différence avec les systèmes à robots réels est leur taille relativement petite. Lors de l'assemblage des compo-sants électriques, nous nous en tiendrons étroitement aux directives et contrôlerons plutôt deux ou trois fois que tout est correct. Avec les constructions mécaniques, même les créations propres, nous veillerons particulièrement à ce que les engrenages et fixations présentent peu de jeu et tournent facilement. Nous ne devons jamais utiliser la « force » lors du montage. Notre créativité aura le privilège d'écrire de nouveaux programmes et de définir ainsi de nouveaux « comportements » dont la complexité ne sera limitée que par les ressources en mémoire et en capacité de calcul dont nous disposons. Les exemples qui suivent donnerons quelques suggestions en ce sens.

2 Premiers pas

Après ces considérations théoriques, nous allons enfin commencer à exécuter quelques expériences. L'un ou l'autre d'entre vous voudrait certainement démarrer tout de suite, et même commencer par le gerbeur à fourche automatique assez compliqué. C'est possible, bien entendu, la construction de la maquette réussissant du premier coup si on observe attentivement la notice de montage. Mais que faire si ça ne fonctionne pas ? Dans un tel cas, il faut systématiquement chercher la cause du défaut. Il surgit alors inévitablement des questions relatives au mode de fonctionnement et aux propriétés des composants utilisés.



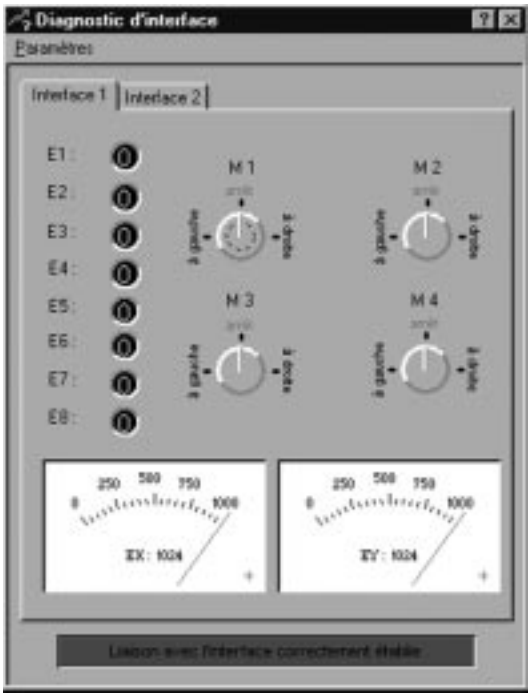
Manifestement, il est indispensable d'avoir certaines connaissances de base relatives aux capteurs et actionneurs. Mais avant de nous familiariser avec toutes ces choses, vérifions les interactions existant entre l'ordinateur et l'interface. On installe le logiciel de commande sur l'ordinateur personnel conformément aux directives données dans le manuel LLWin.



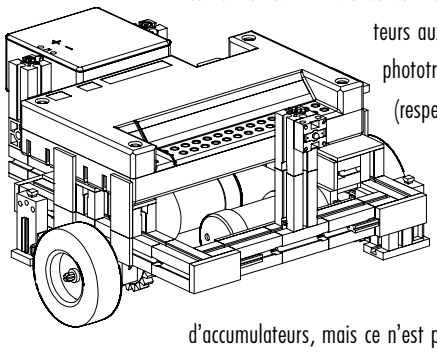
À l'aide du diagnostic d'interface, nous testons les différents capteurs et actionneurs. Nous pouvons p. ex. connecter un contacteur à deux conducteurs se trouvant à l'entrée E1 et constatons alors quel état logique l'interface reconnaît. L'actionnement du contacteur doit entraîner un changement d'état à l'entrée correspondante.

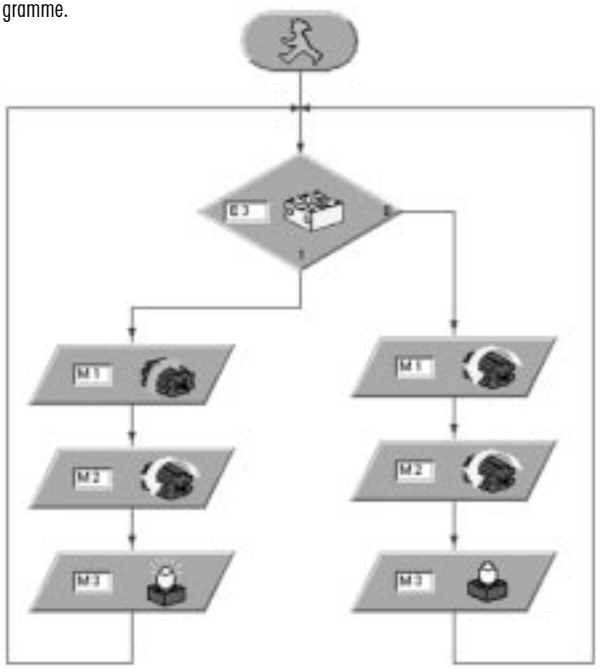
Nous vérifions les sorties en reliant un moteurà une sortie de moteur, p. ex. M1. À l'aide de la souris, nous pouvons mettre le moteur en rotation. Si nous voulons tester également l'entrée analogique, ça fonctionnera si nous utilisons un phototransistor en guise de capteur analogique. Alors que la polarité des raccordements ne joue aucun rôle dans le cas du moteur ou du contacteur (dans le cas le plus défavorable, le moteur tournera dans le mauvais sens), il est impératif d'effectuer le raccordement correct du phototransistor pour qu'il fonctionne correctement. Un contact du transistor est muni d'un repère rouge, et nous le relions à une fiche de raccordement rouge, alors que le contact sans repère se relie à une fiche verte. La deuxième fiche rouge s'em-

boîte dans la prise de l'entrée EX se trouvant plus près du bord de l'interface, alors que la deuxième fiche verte s'ajuste dans la prise d'EX se trouvant plus loin vers l'intérieur. À présent, nous pouvons faire varier l'intensité de l'éclairement du phototransistor à l'aide d'une lampe de poche et modifier ainsi la déviation de l'aiguille. Si l'aiguille ne s'écarte pas de sa position maximale, examinons encore une fois les raccordements du phototransistor. Si par contre l'aiguille se trouve au zéro même lorsque la lampe de poche est éteinte, il se peut que l'éclairage de la pièce, donc la luminosité de l'environnement, soit trop forte. La déviation de l'aiguille change alors lorsque nous recouvrons le phototransistor.



Revenons encore une fois brièvement sur les couleurs attribuées aux fiches : lors du montage, nous veillerons toujours strictement à relier une fiche rouge à un conducteur rouge et une fiche verte à un conducteur vert. Lorsque nous utiliserons des signaux polarisés dans un circuit que nous construisons, prenons toujours un conducteur rouge pour le pôle plus et un conducteur vert pour le pôle moins. Ça peut sembler un peu tatillon (car la couleur du conducteur lui est égale, au courant), mais une assignation claire des couleurs facilite considérablement une recherche systématique des erreurs. Nous allons terminer nos premiers pas dans le domaine de la robotique avec un programme simple. Nous construisons la maquette de base avec les deux moteurs d'entraînement et la roue d'appui conformément à la notice de montage. Nous connectons seulement les moteurs aux sorties M1 et M2. Nous cherchons aussi le phototransistor et le raccordons à l'entrée E 3 (respecter la polarité). Auparavant, nous attachons le phototransistor à la maquette de base de telle façon qu'il soit dirigé « vers l'avant ». Ceux qui veulent peuvent encore connecter l'ampoule lentille à M3 et la fixer p. ex. à côté du paquet d'accumulateurs, mais ce n'est pas absolument nécessaire. Nous ouvrons le programme LLWin et créons un nouveau projet (PROJET – NOUVEAU). LLWin nous propose divers modèles, nous sélectionnons « projet vide » et lui donnons un nom, p. ex. « Step1 ». Après avoir actionné la touche [OK], une fiche de travail vide apparaît avec un bonhomme en feu tricolore et la fenêtre modules. Le bonhomme en feu vert symbolise le début du programme.



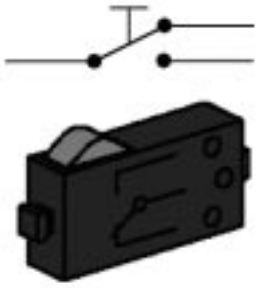


C'est depuis ce point de départ que nous commencerons tous nos programmes. À l'aide de la souris, nous allons chercher les différentes parties de programme dans la fenêtre modules. Les symboles qui y figurent présentent les entrées et sorties de l'interface. La touche de gauche de la souris permet de placer le symbole souhaité, la touche de droite permettant d'en modifier les propriétés. Le symbole à connecteur caractérise une entrée. Pour notre programme, nous plaçons le connecteur sous le symbole de début à l'aide de la souris. Quand nous relâchons le symbole, un dialogue de sélection apparaît. Nous choisissons le phototransistor. Si nous souhaitons effectuer d'autres modifications plus tard, nous pourrons activer ce dialogue avec la touche droite de la souris. Pour les moteurs, nous affectons les sorties et imposons le sens de rotation souhaité. Nous voulons que les moteurs tournent dans le même sens lorsque le phototransistor ne reçoit pas de lumière, et dans le sens contraire lorsqu'il détecte de la lumière. Nous relions ensuite les éléments à l'aide de la fonction dessin. La lampe se trouvant en M3 indique l'état du phototransistor. L'illustration montre la liaison exacte des branches du programme. Ceux qui ne sont pas sûrs que tout est correct peuvent comparer leur programme avec le programme Step1.mdl. À cet effet, on stocke tout d'abord son propre programme et on charge le fichier Step1.mdl depuis le CD-ROM inclus dans le coffret. Si tout est en ordre, on effectue le téléchargement du programme dans l'interface et on le lance immédiatement (RUN - DOWNLOAD). Notre premier robot tourne maintenant sur place. Il le fera jusqu'à ce que nous l'attirions avec une source de lumière. Aussitôt que le phototransistor détecte la lumière, les deux moteurs qui tournaient jusqu'alors dans en sens contraire sont actionnés dans le même sens et le robot se dirige tout droit vers la source de lumière. S'il s'éloigne de la source, il nous faut commuter les pôles des deux moteurs. Sa trajectoire ne sera sans doute pas parfaitement rectiligne, de sorte que le phototransistor perdra le contact avec la source de lumière au bout d'un certain trajet. Ses mouvements vont alors commuter de la commande Avance tout droit à Rotation, et la recherche de lumière va de nouveau commencer. Pour pouvoir expérimenter avec succès, nous avons auparavant fait suffisamment de place car notre robot ne peut malheureusement pas (encore) percevoir les obstacles se trouvant sur son chemin.

3 Capteurs et actionneurs

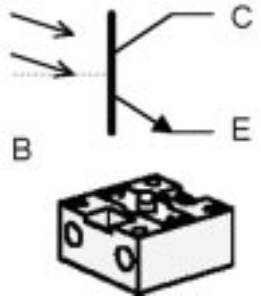
Nous savons à présent que notre bloc le plus important, l'interface, « interagit » avec l'ordinateur. Ce qui nous intéresse maintenant, c'est la façon dont notre interface peut détecter les signaux provenant de l'environnement. Commençons par les entrées. Dans le langage du métier, les entrées ou les signaux d'entrée sont souvent appelés des inputs. Or, un ordinateur, et l'Intelligent Interface en est bien un, peut seulement détecter et traiter des signaux électriques. Nous devons donc rendre les stimuli de l'environnement « traitables par l'ordinateur ». C'est pourquoi tous les capteurs sont des convertisseurs transformant le « sens de la perception » souhaité en un signal électrique. Étant donné que nous ne voulons pas exclusivement suivre en « aveugles » les notices de montage, il est indiqué de consacrer un peu de temps aux propriétés fondamentales des capteurs dont nous disposons. Ceci est d'autant plus important que nous pourrons très bien, plus tard, compléter l'interface de nouvelles applications que nous définirons nous-mêmes.

3.1 L'interrupteur en tant que capteur numérique



Les niveaux logiques simples « 0 » et « 1 » sont représentables à l'aide d'un interrupteur. Le système des coffrets fischertechnik utilise des interrupteurs à saut de précision munis de contacts inverseurs. La particularité des interrupteurs à saut réside dans leur comportement à la commutation. Si nous actionnons lentement et prudemment le bouton de commande rouge, nous sentons un point de poussée net au dépassement duquel le contact de commande effectue la commutation avec un léger déclic. Si nous relâchons lentement le levier de commutateur, il nous faut « faire sortir » le levier nettement plus loin que le point de commutation initial pour obtenir une commutation inverse. Cette différence entre les positions mécaniques de branchement et de débranchement est appelée hystérésis. L'hystérésis de commutation des contacts ou autres circuits électroniques est une propriété importante. Si elle n'existait pas, c'est-à-dire si le point de branchement était identique au point de débranchement, il s'ensuivrait de graves problèmes dans le traitement des signaux. Des parasites minimes tels qu'un tout léger tremblement à l'instant de la commutation « simuleraient » pour l'interface plusieurs actionnements, et il ne serait pas possible de compter des événements avec précision. L'interrupteur est en version à inverseur. Nous pourrons ainsi dans nos expériences exploiter les deux positions de départ imaginables, c'est-à-dire ouvert au repos et fermé au repos.

3.2 Détection de la lumière à l'aide du phototransistor



Le phototransistor est un composant électronique semi-conducteur dont les propriétés électriques dépendent de l'intensité lumineuse. Un transistor ordinaire est un composant à trois raccordements. Ces raccordements sont appelés Émetteur, Base et Collecteur. Sa fonction principale est d'amplifier les signaux faibles. Un courant faible provenant d'un signal et circulant dans la base du transistor entraîne un courant beaucoup plus fort au collecteur du transistor. L'amplification du courant peut être d'un facteur supérieur à 1000. Le phototransistor du coffret ne possède cependant que deux raccordements. Où est passé le troisième ? Avec notre transistor, nous voulons détecter de la lumière. Tout le monde connaît les cellules photovoltaïques permettant de produire du courant à partir de lumière solaire. Le phototransistor est à considérer comme une combinaison de la cellule photovoltaïque miniature et du transistor. La base n'est pas conduite vers l'extérieur (c'est pourquoi elle est en pointillé sur la figure). À sa place, les impulsions de lumière incidente (photons) produisent un très faible courant photoélectrique amplifié ensuite par le transistor et disponible pour traitement au collecteur. Pour que tout ceci puisse fonctionner comme décrit, le phototransistor nécessite une connexion supplémentaire. Comme celle-ci est incluse dans l'interface, nous n'avons pas besoin de nous y intéresser davantage. Le phototransistor peut s'utiliser aussi bien comme capteur numérique que comme capteur analogique. Dans le premier cas, il sert à la détection de transitions clair/sombre nettes, p. ex. une ligne marquée. Il peut cependant aussi distinguer des quantités de lumière d'intensité différente et le phototransistor fonctionne alors capteur analogique.

3.3 Émission de signaux à l'aide de la lampe à incandescence

L'émission de signaux lumineux simples est assurée par la lampe à incandescence. Pour conserver nos termes techniques : la lampe devient un actionneur optique. La structure d'une lampe à incandescence est vraiment simple. Une ampoule de verre dans laquelle on a fait le vide renferme un mince fil de tungstène enroulé en spirale et tendu entre deux broches de raccordement. Lorsque du courant circule à travers la spirale, le fil s'échauffe jusqu'à l'incandescence. Comme il n'y a pas d'oxygène dans l'ampoule, le fil ne brûle pas et la lampe a donc une longue durée de vie. Par suite des fortes contraintes thermiques auxquelles est soumis le filament boudiné, celui-ci se dilate à chaque branchement et se rétracte de nouveau au débranchement. Ces mouvements minimes conduisent un jour ou l'autre au « grillage » de la lampe par suite d'une fatigue du matériau.Une application possible de la lampe à incandescence est l'affichage d'états de contact. Grâce à la programmation d'une lampe clignotante, on peut également produire des messages d'avertissement. Nous aurons encore besoin de la lampe dans un autre cas. En la combinant à deux phototransistors on obtient un capteur spécial dont les propriétés permettent de détecter des lignes. La lampe fonctionne comme source de lumière permettant au phototransistor de détecter un marquage de couleur au moyen de la lumière différemment réfléchie. Une particularité de la lampe utilisée dans le coffret fischertechnik est la lentille optique contenue dans l'ampoule de verre. Le faisceau de lumière est ainsi mieux focalisé et la détection des marquages p. ex. est plus fiable.

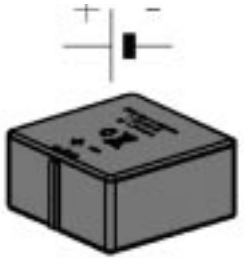
3.4 Moteurs à courant continu comme source de force mécanique

Les moteurs à courant continu sont des actionneurs importants pour les systèmes mobiles. Le coffret « Mobile Robots II » contient deux types de moteurs. Bien qu'ils soient assez différents mécaniquement, leur structure

électrique est cependant identique. Les moteurs à courant continu sont constitués d'un « rotor » qui tourne et d'un « stator » fixe. Dans son principe, le rotor est à considérer comme une boucle formée par un conduc-teur et se trouvant dans le champ magnétique du stator. Un courant qui circule dans la boucle conductrice produit une force qui dévie le conducteur dans le champ magnétique et fait ainsi tourner le rotor. Pour les applications pratiques, la boucle conductrice est une simple bobine (avec ou sans noyau de fer amplifiant le champ magnétique). De nombreux moteurs à courant continu produisent le champ magnétique nécessaire à l'aide d'aimants permanents collés dans l'enveloppe métallique du stator. L'amenée du courant vers le rotor qui tourne se fait au moyen de contacts frotteurs. Ces contacts assurent simultanément l'inversion de courant nécessaire dans la boucle conductrice pour produire un mouvement de rotation ininterrompu. La vitesse de rotation des moteur habituels est de l'ordre de quelques milliers de tours par minute. Un engrenage assure des vitesses plus faibles et un couple plus important. Le coffret contient deux types de moteur différents, le moteur miniature et le moteur de puissance. Le petit moteur miniature compact avec vis sans fin est destiné à des entraînements auxiliaires ou à des usages spéciaux exigeant une faible puissance. Il nécessite toujours un engrenage réduisant sa vitesse. Le moteur de puissance fournit des couples beaucoup plus importants. Il a un engrenage fermement bridé assurant une démultiplication de 50 : 1. Il convient ainsi de façon idéale à l'entraînement exigé par notre robot. Il en existe aussi une variante assurant une démultiplication de 8 : 1 (non contenue dans le coffret). Sur ce moteur, la rotation de l'arbre de sortie serait cependant trop rapide pour l'entraînement du robot.

3.5 Alimentation électrique

Les systèmes mobiles nécessitent une alimentation électrique autonome. Cette source d'énergie alimente tous les circuits consommateurs. Les exigences posées à l'alimentation électrique sont diverses. Alors que les moteurs d'entraînement se contentent d'une tension non stabilisée, beaucoup de capteurs nécessitent des tensions stables pour pouvoir livrer des résultats précis. Pour des raisons économiques, l'utilisation de batteries ou d'accumulateurs est le seul moyen judicieux d'alimenter les robots mobiles en courant. Les cellules photovoltaïques et les piles à combustible ne sont malheureusement pas encore assez puissantes pour donner des résultats utilisables à un coût raisonnable.



Les accumulateurs sont à préférer aux batteries car on peut les recharger de nombreuses fois. Le paquet d'accumulateurs fischertechnik représente un bon compromis entre taille et énergie emmagasinée. Le paquet d'accumulateurs ne fait pas partie du coffret, mais on peut se le procurer avec un chargeur spécial comme « Accu Set » vendu sous la Référence 34969. L'illustration montre le symbole de couplage et l'accumulateur. Normalement, la polarité n'est pas indiquée sur le symbole de couplage. Un pense-bête permet de retenir plus facilement quel raccordement est le plus : « On peut couper le trait long et assembler les bouts pour en faire un plus. » Lors du raccordement de la source de tension à l'interface, il est essentiel de toujours veiller à respecter la polarité.

3.6 Capteurs additionnels

Le système fischertechnik est relativement facile à compléter d'autres capteurs. Le cas le plus simple est d'utiliser des capteurs provenant d'autres coffrets, p. ex. le capteur thermométrique ou le capteur magnétique du coffret « Profi Sensoric » Référence 30491. Nous pouvons cependant utiliser aussi des capteurs tout à fait différents. Le commerce spécialisé propose les kits et composants les plus divers. On peut même utiliser des capteurs aussi exotiques que les détecteurs de gaz ou les sondes à radar. Mais comme nous ne voulons en aucun cas détruire l'Intelligent Interface par des tensions d'entrée trop élevées ou des charges incorrectes, seuls des bricoleurs expérimentés devraient réaliser des solutions propres. La méthode la plus sûre pour raccorder d'autres capteurs est d'assurer la séparation galvanique entre le capteur et l'interface. Toute une série de capteurs possède un relais qui s'y prête bien. On raccorde les contacts de branchement du relais comme un interrupteur ordinaire fischertechnik, et ils signaleront l'apparition de nouveaux stimuli provenant de l'environnement. Un conseil : des « fischer-techniciens » mordus publient sur Internet de nombreuses extensions de ce genre.

4 Les maquettes de robots

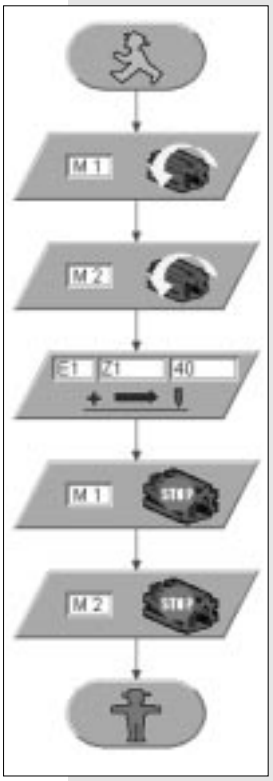
Les constructions suggérées dans ce qui suit présentent quelques variantes de robots mobiles autonomes. Nous commencerons par une maquette simple. En l'élargissant, tu reconnaîtras et testeras l'utilisation des différents capteurs. Ce qui sera déterminant, ce sera de coupler non seulement des états internes du robot, p. ex. la mesure du trajet parcouru par des roues à impulsions, mais aussi des signaux externes de l'environnement tels que la recherche de lumière ou d'une piste. Sur chaque maquette, certains problèmes te seront posés. Ils sont destinés à stimuler ton imagination et à te familiariser avec le sujet. Les programmes LLWin relatifs aux différents problèmes se trouvent sur le CD-ROM contenu dans le coffret. Tu peux aussi te poser des problèmes sur les différentes maquettes. La maquette la plus simple est la maquette de base. Sur celle-ci, les moteurs d'entraînement sont assemblés avec l'inter-face pour former une unité compacte. Deux moteurs fournissent la force d'entraînement du robot. Ils sont disposés l'un en face de l'autre de telle façon que chaque moteur agisse sur une roue. Pour que ce robot ne bascule pas, une roue d'appui en assure la stabilité. Une telle disposition des moteurs est appelée differential drive (entraînement différentiel). Elle garantit une mobilité maximale nécessitant un espace minimal. Il est même possible d'exécuter des rotations sur place. Le milieu du segment séparant les deux moteurs est alors le centre de rotation autour duquel le robot se déplace. De cette façon, il réussit toujours à se piloter dans les conditions les plus difficiles avec très peu de calculs. Les moteurs peuvent entraîner les roues avec deux démultiplications différentes (une lente de 100 : 1 et une rapide de 50 : 1). Pour la variante la plus lente, l'entraînement est démultiplié une nouvelle fois au moyen de roues dentées fischertechnik dans le rapport 2 : 1. Pour chaque maquette, il est indiqué quelle démultiplication il faut utiliser.

4.1 La maquette de base

Nous construisons d'abord la maquette de base (démultiplication 100 : 1) conformément à la notice de montage. Étant donné que cette maquette nous servira de base dans de nombreuses expériences, nous la construisons avec un soin particulier. Lorsque tout est terminé mécaniquement, nous contrôlons que les moteurs tournent facilement. Pour cela, nous raccordons brièvement chaque moteur directement à l'accumulateur, sans l'interface.

Problème 1 :

Programme l'interface de telle façon que la maquette avance tout droit pendant 40 impulsions. Pour mesurer le nombre des impulsions, utilise le contacteur de comptage E1, et comme interrupteur de remise à zéro le contacteur se trouvant en E8. Modifie ensuite le programme de telle façon que la maquette parcoure des trajets de différentes longueurs, p. ex. 80 cm. Quelle est alors la précision à la répétition pour de tels trajets ?



Problème 2 :

La maquette doit tourner de 180° après le trajet rectiligne de 80 impulsions. Tiens compte des différents sens de rotation des moteurs d'entraînement lors du déplacement rectiligne et de la rotation.

Solution :

En moyenne, la maquette parcourt 1 cm par impulsion comptée. La reproductibilité est du même ordre de grandeur, environ 1 cm pour 80 impulsions. Elle fluctue cependant en fonction du support sur lequel le robot se déplace. Les revêtements de sol textiles épais ou moelleux sont particulièrement défavorables.

Avant de nous tourner vers ce problème, nous allons éclaircir deux choses. D'une part, nous avons utilisé dans notre programme un nouveau module fonctionnel appelé POSITION. Il s'agit là d'un module qui reste actif jusqu'à ce que le nombre d'impulsions réglé soit détecté à l'entrée spécifiée (ici E1). Dans l'optique du programme, ceci signifie que nous utilisons ici une condition d'attente définie. Lors du premier essai, nous avons utilisé cette fonction comme mesure du trajet pour l'avance rectiligne.

Si le robot doit tourner, l'approche est pratiquement la même, nous devons simplement modifier le sens de rotation des moteurs. À présent, nous n'avons plus qu'à entrer le nombre des impulsions et notre robot tournera sur place.

Venons-en au deuxième point. Nous n'allons pas simplement faire des essais jusqu'à ce que le robot tourne de 180° ; nous allons auparavant calculer cette valeur.

Les moteurs d'entraînement ont une configuration de differential drive, c'est-à-dire que les roues du robot se déplacent, lors de la rotation, sur la circonférence d'un cercle dont le diamètre est déterminé par l'écartement des roues. Pour une rotation de 180° chaque roue devra donc parcourir exactement la moitié de cette circonférence.

Calculons tout d'abord la circonférence u :

$u = \pi \cdot d = 630$ mm

d = diamètre (écartement des roues env. 200 mm)

Précédemment, nous avons déterminé un trajet d'env. 1 cm/impulsion, nous avons donc besoin de 30,5 impulsions pour le trajet de 314 mm (demi-circonférence). Comme nous ne pouvons calculer que des valeurs entières, nous devons nous décider pour 30 ou 31 impulsions. Nous testons quelle valeur réelle fournit la plus grande précision.

Conclusion :

Après les mesures effectuées avec la roue d'impulsion, nous constatons que la précision que nous pouvons atteindre dans nos mesures n'est pas très élevée. En particulier lorsque le robot parcourt plusieurs trajets les uns derrière les autres ou de façon répétée, l'erreur absolue de la mesure s'accumule. C'est tout aussi problématique avec l'erreur qui apparaît en raison de cycles qui n'ont pas encore été entièrement saisis.

Les possibilités de minimiser ces erreurs sont limitées. On peut d'une part augmenter le nombre d'impulsions de déplacement par unité de longueur. L'idéal serait de monter le compteur directement sur l'arbre du moteur. En plus du fait que nous n'avons pas accès à cet arbre, il apparaît ici le problème de la vitesse de détection limitée de l'interface. S'il lui parvient trop d'impulsions par unité de temps, il se peut que l'interface en « oublie » quelques-unes. Un calcul précis du trajet devient alors illusoire.

Il existe d'autres erreurs que nous ne pouvons pas du tout chiffrer, comme p. ex. le glissement des roues sur différents supports ou les variations dans le diamètre des roues. Nous nous consolons à la pensée que ces problèmes ne sont parfois qu'insuffisamment résolus même par des systèmes commerciaux beaucoup plus complexes et beaucoup plus chers.

4.2 Robot avec détection des bords

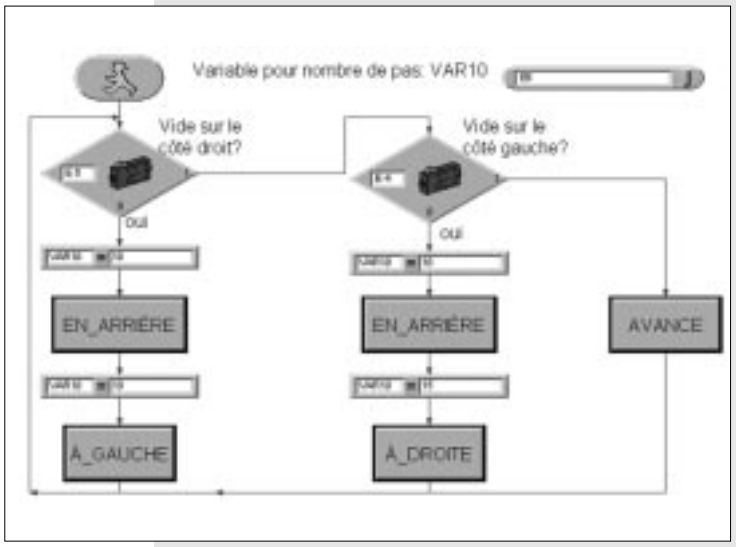
Maintenant que nous avons examiné en détail notre maquette de base, nous allons essayer de faire apprendre au robot la « peur » des précipices. Jusqu'à présent, le robot s'est agité sur le dessus de table sous notre surveillance attentive pour qu'il ne tombe pas de la table. En vérité, ce n'est pas un comportement particulièrement intelligent. Nous allons donc changer cet état de choses.

Pour détecter un bord, le robot a besoin d'un détecteur de bords. Un procédé à la fois simple et praticable utilise deux roues auxiliaires. Comme c'est le cas pour une antenne d'insecte, celles-ci sont placées dans le sens du déplacement devant le robot, et équipées d'un interrupteur. Ces roues sont construites de façon à pouvoir se déplacer verticalement. Un bord fait tomber la roue auxiliaire vers le bas et déclenche ainsi le capteur.

Problème 3 :

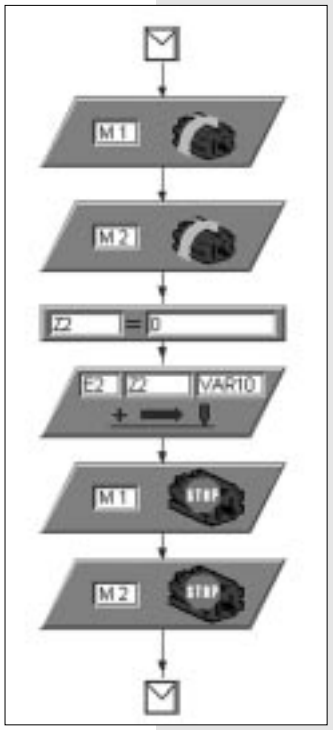
Construis la maquette « Robots avec détection des bords » conformément à la notice de montage (démultiplication 50 : 1). La maquette doit avancer tout droit. Aussitôt qu'elle arrive à un précipice se trouvant à gauche, elle devra l'éviter vers la droite, s'il se trouve un précipice à droite, elle devra l'éviter vers la gauche. Pour obtenir une meilleure vue d'ensemble, on utilisera certains déplacements sous forme de sous-programmes (En avant, À gauche et À droite). Le nombre de pas sera détecté par des contacteurs de comptage.Il sera fixé dans une variable appelée VAR10. Cette donnée sera différente pour les sous-programmes À gauche et À droite.

Deux données différentes réduisent le risque de ne pas pouvoir sortir d'un coin.



Solution :

Le problème posé nous permet de percevoir que nous devons piloter le robot en fonction des détecteurs de bords. Nous décomposons donc le problème en des parties plus petites. Nous interrogeons tout d'abord les contacteurs (détecteurs de bords). Si aucun n'est actif, le robot avance. Dans l'illustration, on peut le voir sous la forme du bloc « En avant ». Derrière un tel module qui est nouveau pour nous se cache un sous-prgramme. Les sous-programmes améliore la clarté de systèmes complexes et permettent, quand on les utilise plusieurs fois, de mieux mettre à profit la puissance des systèmes d'ordinateur. Notre premier sous-programme est très simple puisqu'il met seulement en mouvement les moteurs M1 et M2. Nous avons cependant besoin d'autres sous-programmes. Le robot doit, selon sa position, esquiver vers la gauche, vers la droite ou en arrière. Dans un tel cas, il ne suffit plus de mettre simplement les moteurs en marche. Il faut programmer une certaine séquence de déplacements. Regardons un autre sous-programme de plus près. Il doit permettre au robot de reculer lorsqu'il détecte un bord. À cet effet, nous mettons les moteurs en marche arrière. Notre roue d'impulsion doit alors identifier un trajet défini. Nous fixons la longueur de ce trajet à l'aide de la variable VAR10. Le bloc d'affectation Z2 = 0 est nouveau pour nous. Après un peu de réflexion, nous en reconnaissons le sens. Si nous ne mettons pas la variable de comptage à zéro, le mécanisme ne fonctionnera qu'une fois car, dès que Z2 atteint la valeur de la variable VAR10, notre compteur de trajet échouerait à chaque passage suivant. Dans l'optique des programmateurs professionnels, nous avons affaire à cet endroit à des variables locales et globales. La variable locale Z2 doit être initialisée avant chaque utilisation dans le sous-programme.



Conclusion :

L'appel de sous-programmes augmente la clarté des programmes. Nous utilisons des varia-bles afin de mesurer différentes valeurs, ici des trajets. Nous avons besoin de trajets de lon-gueur différente pour que notre robot puisse se « libérer » même s'il se trouve dans un coin. Si les trajets étaient absolument identiques, il pourrait arriver que le robot exécute toujours le même va-et-vient dans le coin.

Quand nous utilisons des variables, nous faisons attention à leur plage de validité. Les varia-bles locales, c'est-à-dire les variables que nous n'utilisons qu'à l'intérieur d'un sous-programme, doivent être initialisées avant leur première mise en œuvre. Nous constatons en outre que notre robot a besoin d'une certaine liberté de mouvement pour pouvoir fonctionner. S'il rencontre de nouveau un bord au cours d'un mouvement d'esquive, il ne pourra pas réagir. Les bricoleurs très forts peuvent essayer de trouver une solution à ce problème.

4.3 Robot avec détection des obstacles

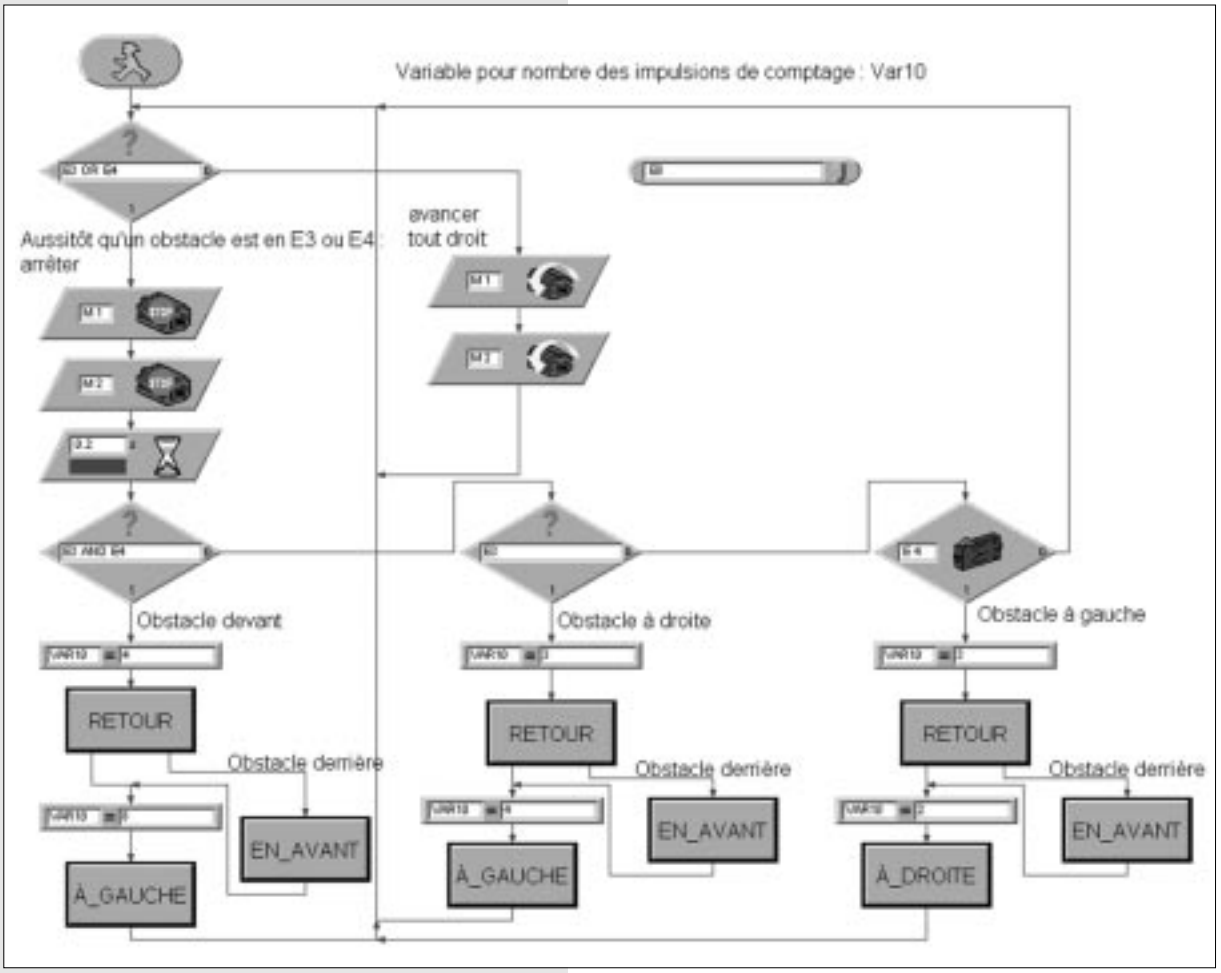
Nous pouvons à présent détecter les bords relativement bien. Les obstacles ordinaires sont par contre problématiques. Nous devons modifier le principe de la détection des bords. Un contact au choc approprié va remplacer les roues auxiliaires. À cette occasion, nous méditons également sur un défaut que présentent les détecteurs de bords ; il y a des situations dangereuses dans lesquelles le robot, sans capteur à l'arrière, tombe « en aveugle » dans le précipice lors de sa marche arrière. Un troisième contacteur évitera ce défaut. Entretemps, nous sommes devenus des programmateurs assez futés. C'est une chance car le problème devient à présent plus compliqué comme le montre un coup d'œil sur le programme. Le programme contient d'autres blocs fonctionnels nouveaux. À certains endroits du programme, nous in-sérons des modules ATTENTE. C'est facilement compréhensible car on attend ici la durée introduite avant d'exécuter la prochaine fonction du programme. Ce qui est vraiment nouveau, ce sont les comparaisons logiques. Jusqu'à présent nous avons toujours, lors de l'interrogation des contacteurs, effectué immédiatement une comparaison et ramifié notre programme de façon correspondante. Il y a également une comparaison simi-laire lors du comptage des impulsions, c'est la comparaison avec un certain nombre d'impul-sions. La comparaison logique avec plusieurs expressions dans un module COMPARAISON est également nouvelle pour nous.

Problème 4 :

Comme décrit dans la notice de montage, on construit le robot avec la démultiplication rapide de 50 : 1. La maquette doit avancer tout droit. Aussitôt qu'elle détecte un obstacle à l'un des contacteurs avant (E3 ou E4), le robot s'arrête. S'il a détecté un obstacle à droite, le robot reculera et contournera par la gauche (env. 30°). En cas d'obstacle à gauche le robot, après un recul, contournera par la droite (env. 45°). Le nombre inégal de degrés est nécessaire pour qu'il puisse aussi sortir d'un coin.

Au cas où l'obstacle serait détecté directement devant, le robot doit reculer et l'éviter à 90°. Au cas où un obstacle se présenterait derrière lors de la marche arrière, il devra faire un mouvement bref vers l'avant et ensuite contourner comme prévu.

Solution :



Conclusion :

La complexité des programmes augmente. Nous essayons de remédier à cet accroissement de la difficulté en améliorant nos techniques de programmation. Les sous-programmes s'avèrent être un bon moyen à cet effet. Comme nouvelle structure de contrôle, nous identifions les comparaisons logiques suivies d'une ramification du programme. Nous constatons également que, pour intégrer de nouvelles propriétés ou améliorer la réalisation d'expériences déjà connues, de plus en plus de capteurs sont nécessaires.

4.4 Le chercheur de lumière

Jusqu'à présent, nous avons fait réagir le robot de façon plutôt passive à des signaux de son environnement. À présent, nous voulons que le robot cherche de façon active. Le coffret contient deux phototransistors que nous allons mettre en œuvre comme détecteurs de lumière. Chaque capteur va alors agir sur un moteur, ce qui va permettre une « poursuite d'un rayon de guidage ».

Le programme se compose de deux parties. Une section contient la recherche d'une source de lumière et l'autre section réalise la poursuite ou l'approche de la source de lumière.

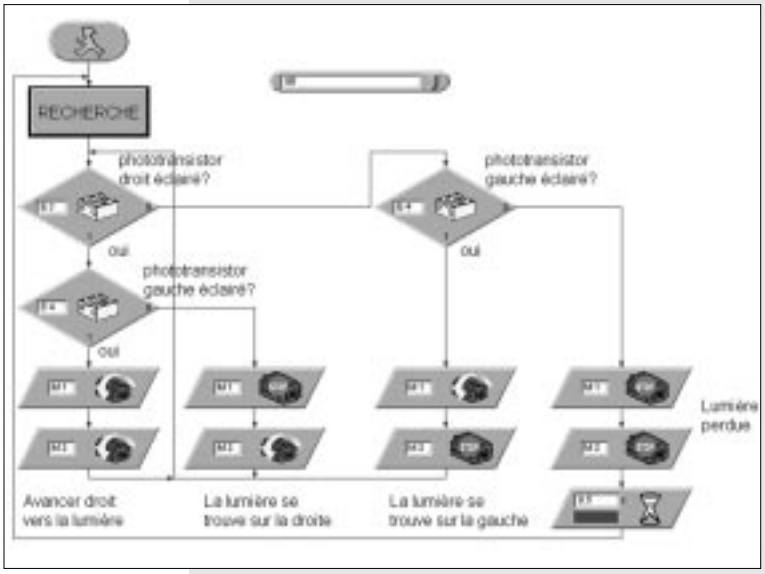
Bien entendu, nous utilisons de nouveau les possibilités qu'offre la technique des sous-programmes. Le lancement active un sous-programme RECHERCHE. On ne quitte ce sous-programme que lorsqu'une source de lumière a été trouvée. Le programme principal essaie de diriger le robot vers la source de lumière. Chaque fois que la direction prise par le robot s'écarte fortement de la ligne idéale, un des capteurs photoélectriques n'est plus éclairé par la source de lumière. Le moteur correspondant s'arrête alors un court instant pour que les deux capteurs puissent de nouveau détecter la source de lumière. Il en résulte le problème que nous te posons.

Problème 5 :

Nous construisons la maquette « Chercheur de lumière » avec la démultiplication lente de 100 : 1. Programme d'abord la fonction « Recherche lumière » (utilise directement un sous-programme).
Le robot tournera alors d'au moins 360° dans un sens. Ensuite, il tournera d'au moins 360° dans l'autre sens. S'il trouve une lumière durant la recherche, le robot s'arrêtera. S'il ne trouve pas de lumière durant les deux rotations, il interrompt la recherche et le programme se termine. Si la recherche de lumière est couronnée de succès, la maquette doit se diriger vers la source de lumière. Si la source de lumière se déplace vers la gauche ou vers la droite, le robot suivra les déplacements de la lumière. S'il perd le contact avec la source de lumière, le programme se met de nouveau à chercher la lumière. Essaie d'attirer le robot avec la lampe de poche et de le conduire à travers une course d'obstacles.

Conseil :
Utilise comme source de lumière une lampe de poche. Essaie de ne pas focaliser trop forte-ment le rayon lumineux pour que les deux capteurs photoélectriques soient éclairés par la source de lumière. Tiens compte du fait que, dans des pièces très claires, ta lampe de poche sera surpassée par d'autres sources de lumière, p. ex. la lumière du soleil pénétrant à travers une grande fenêtre. Dans un tel cas, il se peut que le robot passe à côté de ta lampe et se dirige vers la lumière plus forte.

Solution :



Conclusion :
Nous avons à présent construit un robot qui tient activement compte de son environnement. Ses capteurs sont programmés pour localiser une source de lumière et, s'ils l'ont localisée, à se diriger vers cette source ou à la poursuivre.
Nous constatons que le programme immobilise le robot s'il n'a trouvé aucune source de lumière. Si par exemple un simple petit obstacle coupe le contact visuel du robot avec la source, celui-ci ne peut pas la trouver ni la détecter, bien qu'une source de lumière soit pourtant présente. Manifestement il serait judicieux, après une recherche de lumière restée sans succès, de faire se déplacer le robot, plus au moins au hasard, vers un autre endroit pour qu'il puisse y chercher de nouveau une lumière.
Cependant, qu'est-ce qu'il se passera si l'obstacle qui cache la lumière au robot se trouve exactement dans la direction d'avance du robot ? Ce problème semble suffisamment intéressant pour l'examiner d'un peu plus près.

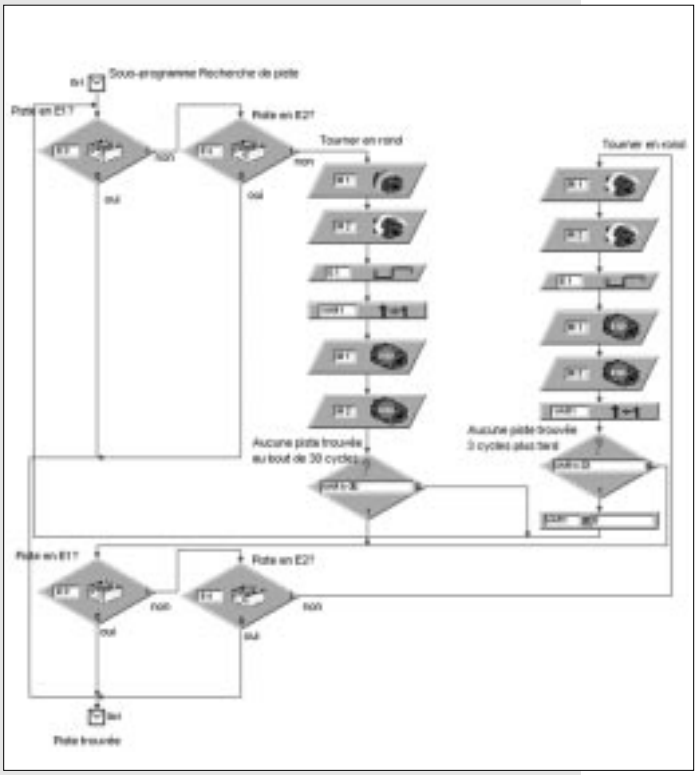
4.5 Le chercheur de piste

Recherche et poursuite sont des propriétés essentielles des êtres intelligents. Nous avons construit et programmé un robot qui réagit aux signaux directs émis par son objectif ou des victimes potentielles. Avec le chercheur de piste, nous allons utiliser un autre principe de recherche. À la place de l'avance précise vers l'objectif constitué par la source de lumière, nous traçons une ligne que le robot devra suivre. À l'aide des capteurs photo-électriques, ce problème est relativement facile à résoudre. Nous mesurons la lumière réfléchie par la marque et corrigeons les moteurs en rapport. Pour que ça fonctionne avec précision, nous éclairerons la ligne avec notre lampe. Nous veillerons alors à ce qu'aucune disposition défavorable de la lampe et des capteurs de lumière n'éblouisse ces derniers par de la lumière parasite. Dans ce contexte, la focalisa-tion de la lumière par la lentille optique de notre lampe s'avère particulièrement avantageuse.

Problème 6 :
Construis la maquette « Chercheur de piste » (engrenage de 100 : 1). Écris d'abord un sous-programme permettant de chercher la piste. À cet effet, le robot tournera en rond de 360°. S'il ne trouve aucune piste, le robot avancera un peu tout droit et cherchera de nouveau. Pour la détection de la piste, on interroge les capteurs photo-électriques. Si le robot a découvert la piste, il la suivra. Si la piste s'arrête ou si le robot la perd, p. ex. en raison d'un changement brusque de direction, la recherche doit recommencer.

Conseil :
Après avoir allumé la lampe, il faut attendre un court instant, env. une seconde, avant d'interroger les deux phototransistors, sinon le phototransistor détecterait « sombre », donc une piste, là où il n'y en a aucune. Comme piste, nous utiliserons un ruban isolant noir d'env. 20 mm de large ou bien nous dessinerons une piste noire d'une telle largeur au crayon feutre sur une feuille de papier blanc.

Solution :

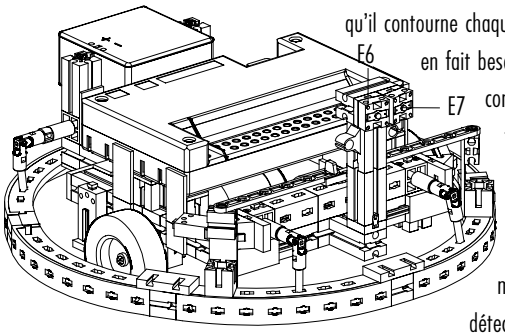


Conclusion :

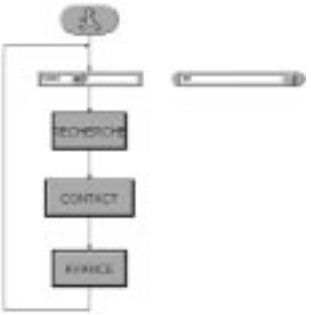
Une fois que notre robot a trouvé la piste, il la suit inlassablement. Si nous traçons la piste sous forme de ligne fermée, le robot y fera ses tours. Nous devons simplement éviter les virages serrés dans lesquels les capteurs optiques pourraient perdre le contact avec la ligne. Certes, le robot essayera alors de localiser à nouveau la piste, cependant il nous faut être francs et admettre que ça ne sera possible que dans une zone pratiquement exempte d'obstacles.

4.6 La mite électronique

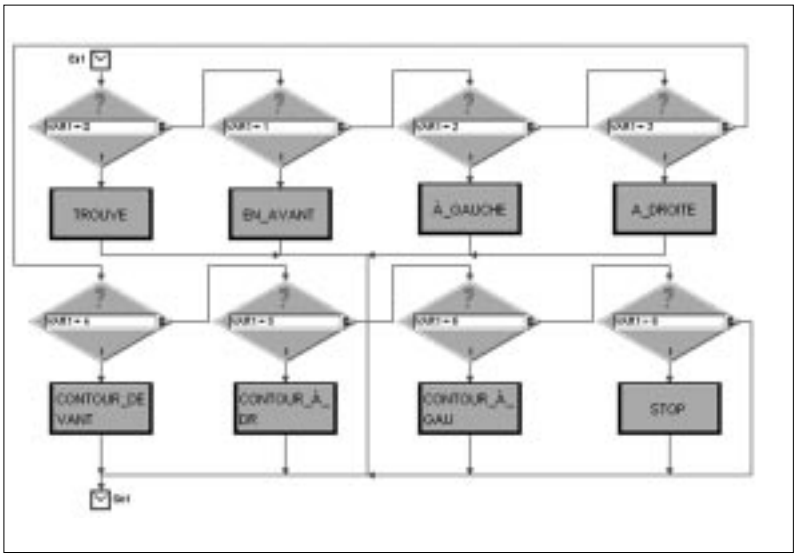
Le problème du couplage de plusieurs modes de comportement tels que recherche, poursuite et évitement est déjà apparu plusieurs fois. Dans une autre expérience, nous allons examiner l'interaction de tels comportements. Résumons brièvement les résultats que nous avons obtenus jusqu'à présent. Le chercheur de lumière simple suit plus ou moins en aveugle la lampe de poche qu'on tient devant lui. Le robot part tacitement du fait que, là où se trouvait précédemment la lampe, il ne peut apparaître aucun obstacle. De la même façon, il suppose qu'il n'atteindra jamais vraiment la lampe. Cependant, ces suppositions ne correspondent à la réalité que dans des cas d'exception. Nous sommes obligés de guider le robot avec la lampe pour qu'il contourne chaque obstacle. Nous aurions en fait besoin d'un robot qui puisse contourner les obstacles de façon autonome. Dans ce but, nous allons compléter, comme le montre l'illustration, la maquette « Robots avec détection des obstacles » de la propriété Recherche de lumière. Nous connectons les deux phototransistors à E6 et E7, et reprenons tout le reste de la maquette Détection des obstacles. Du point de vue scientifique, nous avons alors équipé le robot de deux modes de comportement. Étant donné cependant que les deux schémas de comportement ne peuvent pas être actifs simultanément, ils reçoivent des priorités différentes. La situation qui se présente à nous est alors la suivante : dans le cas normal, le robot est à la recherche de la lumière. S'il détecte un obstacle, pour ainsi dire un danger pour le robot, le comportement Évitement des obstacles sera activé. Lorsque tout rentre dans l'ordre, le robot peut continuer à chercher la source de lumière.Nous avons ainsi procuré au robot des aptitudes essentielles qui lui permettent de se piloter de façon autonome et d'éviter les dangers. Ceux d'entre vous qui ont un ami qui possède également un coffret fischertechnik pourront pousser l'expérience encore plus loin. On installe simplement une source de lumière sur chaque robot, et ceux-ci pourront alors se chercher mutuellement. Lors de la programmation, nous avons jusqu'à présent travaillé suivant le principe « Essai et erreur ». Pour écrire tes programmes, tu t'es sans doute mis « simplement » à la tâche et observé au cours de tes expériences comment le robot faisait alors (ou plutôt ne faisait pas) ce qu'il devait faire. Bien entendu, les concepteurs de logiciels professionnels ne peuvent en aucun cas recourir à une telle méthode de conception avec leur produits. Dans une telle situation, ils doivent définir une stratégie de conception précise avant même d'écrire la première ligne de pro-gramme. Et ça ne se fait pas comme ça,



mais suivant des règles intangibles. Il s'est imposé des procédés portant des noms bizarres tels que p. ex. « Conception structurée » (Top-Down). On essaie ici de définir le système tout entier « de haut en bas », sans se préoccuper des détails au début. Nous allons essayer de programmer notre mite suivant la conception structurée. Pour cela, nous commencerons par le programme principal, ce qu'on appelle la « boucle principale » (en anglais : Main loop). L'illustration montre la conception élémentaire. La boucle principale se compose simplement des modes de comportement nécessaires au robot, à savoir RECHERCHE, CONTACT et AVANCE. Ce qui nous paraît bizarre, c'est la variable VAR1, et comment allons-nous donner une priorité aux modes de comportement du robot avec une boucle aussi simple ? La priorité résulte de l'ordre des succession des sous-programmes, la variable VAR1 servant à attribuer les commandes de déplacement.



En y réfléchissant bien, les mouvements nécessaires au robot peuvent se réduire à un nombre précis de manœuvres. Le robot nécessite un mouvement de recherche, un mouvement d'évitement et un mouvement de correction. Les mouvements d'évitement et de correction doivent en plus se subdiviser suivant les directions, à droite ou à gauche. Il devient donc clair que le sous-programme RECHERCHE calcule un mouvement possible parmi n (n = nombre total de mouvements), de même que le sous-programme CONTACT. Étant donné que CONTACT s'exécute après RECHERCHE, il effacera les instructions de RECHERCHE. Le sous-programme AVANCE ne fait plus qu'exécuter ce qu'on lui ordonne. Le procédé de la conception structurée semble vraiment être utile. Cependant, nous ne savons pas encore à quoi vont ressembler les sous-programmes. Commençons par le premier sous-programme RECHERCHE. Ce sous-programme se charge de l'interrogation des capteurs photoélectriques. En fonction de ces deux capteurs, il existe quatre variantes possibles : capteur gauche, capteur droit, capteurs gauche et droit, pas de capteur. On définit ainsi exactement quatre manœuvres d'avance possibles : TROUVE, EN AVANT, À GAUCHE, À DROITE. À chacune de ces manœuvres d'avance, on peut également assigner un sous-programme ; nous procédons de façon stricte suivant les critères de la conception structurée. Il est clair à présent à quoi va ressembler le premier sous-programme RECHERCHE. RECHERCHE fournit un paramètre qui sera concrétisé par AVANCE. Le sous-programme proprement dit est très simple. Des comparaisons multiples permettent de régler le paramètre de sortie.



F

Dans le programme déjà écrit [on examine ensuite dans le sous-programme CONTACT si des obstacles ont déclenché les contacteurs se trouvant sur le robot. Pour plus de simplicité, nous sautons ce programme et examinons comment les mouvements des moteurs sont activés dans AVANCE. Nous sommes un peu surpris de constater que ce sous-programme appelle d'autres sous-programmes. Est-ce que ça va s'arrêter un jour, ce procédé ? Nous nous trouvons toujours dans la phase de conception (étape suivante de l'approche structurée). AVANCE fixe quelle manœuvre d'avance doit commencer, et à quel moment.C'est seulement derrière les sous-programmes TROUVE, EN AVANT, ... que se cache un « vrai » code programme. Nous sommes enfin arrivés tout en bas. Maintenant, les moteurs sont mis en marche et arrêtés. Chacun de ces sous-programmes renferme une tâche bien limitée que nous pouvons programmer de façon claire.

Conclusion :
Les programmes complexes exigent des approches complètement nouvelles. Il est judicieux de chercher une approche qui procède d'un bout à l'autre. Nous élaborons un programme suivant le principe de la conception « De haut en bas ». Les approches nécessaires se formulent (tout d'abord) en des structures formelles, p. ex. RECHERCHE ou AVANCE. Par la suite ces approches font l'objet d'un raffinement, et seront remplies du code programme proprement dit seulement à la fin. Nous séparons ainsi la structure du programme du code programme proprement dit. Nous pouvons contempler ces deux éléments de façon séparée et, ce qui est essentiel, en analyser aussi les erreurs de façon séparée.

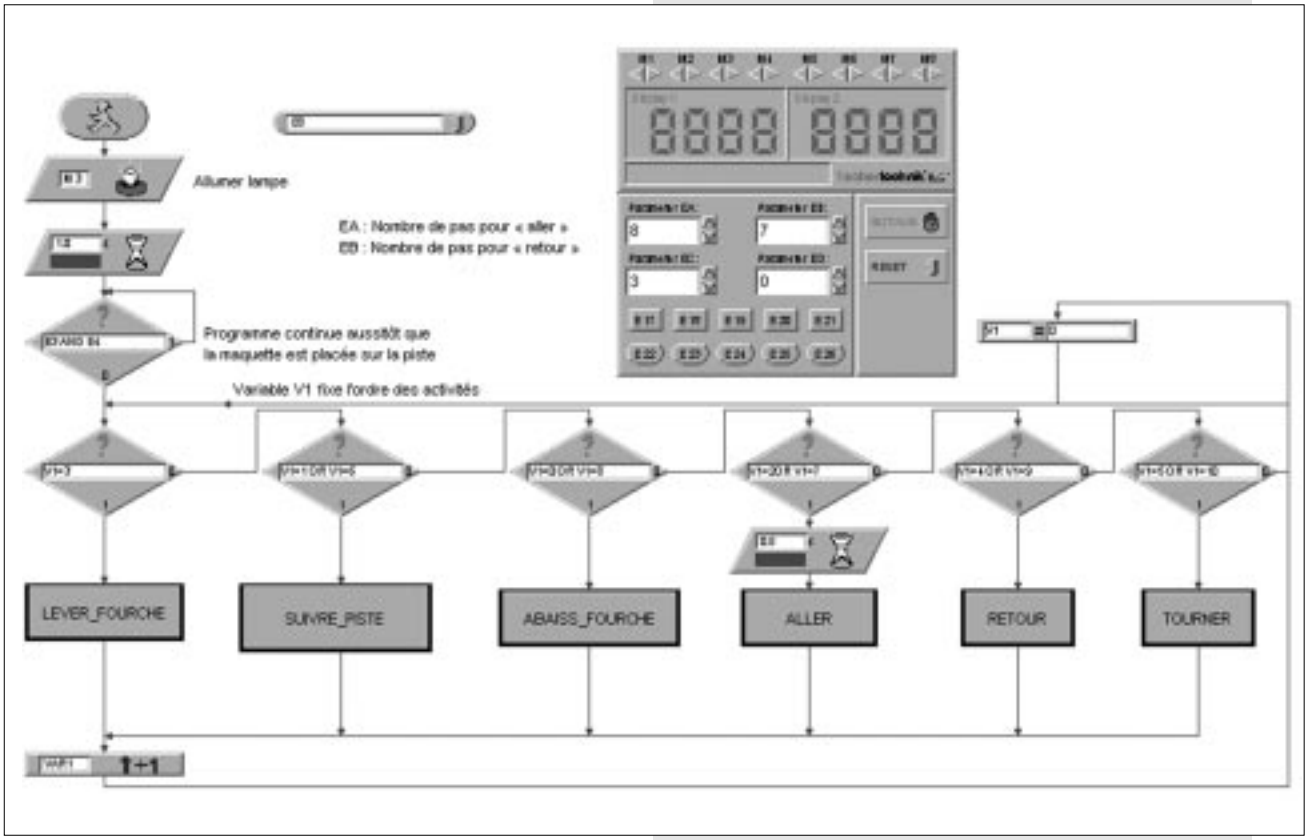
4.7 FTS - Système de transport sans chauffeur

Quittons le domaine de l'expérimentation – des mauvaises langues prétendent souvent que c'est un jeu – et pénétrons dans le domaine des applications pratiques. Nous ne voulons plus seulement construire et programmer un robot pour l'amour de l'art, mais voulons qu'il exécute une tâche. Dans ce but, nous équipons le chercheur de piste d'une fourche de transport mobile. Il en résulte un gerbeur à fourche robot qui est en mesure de transporter du matériel stocké sur une palette.
Bien que cet essai puisse paraître simple, il renferme cependant toutes les composantes d'une application industrielle. Et en plus, de tels systèmes de transport sont déjà partiellement mis en œuvre de nos jours.
Entre-temps, nous avons acquis pas mal d'expérience de programmation et pouvons donc nous risquer avec confiance à élaborer ce programme qui est tout de même très complexe.

Problème 7 :
Le système de transport sans chauffeur (FTS) devra rouler le long d'une piste noire. À la fin de la piste, il prend une palette, tourne de 180° et fait le trajet inverse. Lorsqu'il détecte la fin de la piste, il y dépose la palette, tourne de nouveau, et va chercher la palette suivante.

Problème 8 :
En guise d'extension du problème 7, nous compléterons le programme de telle façon que la palette est seulement déposée un court instant aux bouts de la piste. Le robot va donc chercher la palette, roule jusqu'au bout de la piste, dépose la palette un court instant, la reprend, roule jusqu'à l'autre bout de la piste, etc.

Solution :



Conclusion :

Dès que nous jetons un coup d'œil sur le programme, nous reconnaissons immédiatement l'approche bien connue. Une variable d'état pilote le comportement de notre transporteur sans chauffeur.

À l'aide de nos techniques de programmation entre-temps assez perfectionnées, nous réussissons à réaliser toute une diversité de commandes. Ce système de transport sans chauffeur démontre les possibilités qu'offre la combinaison des coffrets fischertechnik avec l'Intelligent Interface. En partant de robots mobiles simples, nous avons atteint un niveau qui se distingue à peine de la technique de commande et de régulation industrielle.

5 Un chapitre de dépistage des défauts

Effectuer des expériences est amusant. Mais tant que tout fonctionne. L'idéal serait que tout réussisse toujours d'emblée. Malheureusement, ce n'est pas le cas.

C'est seulement lorsqu'une maquette ne fonctionne pas qu'on voit si on a exactement compris le mécanisme et si on trouve le défaut immédiatement. En cas de défauts mécaniques, on peut tout de même voir quelque chose (montage incorrect) ou sentir quelque chose (les roues tournent difficilement). S'il s'y ajoute des problèmes électriques, ça devient plus ardu. Pour le dépistage des défauts, les professionnels utilisent toute une série d'instruments de mesure très divers comme p. ex. voltmètre ou oscillographe. Tout le monde n'a pas de tels appareils à portée de la main. Nous allons donc essayer de cerner et d'éliminer un défaut avec de simples moyens.

Avant de commencer nos expériences, il nous faut d'abord préparer certains composants provenant du coffret fischertechnik. Il s'agit pour l'essentiel des connexions par câbles. Ici, il nous faut connecter les fiches livrées aux différents tronçons de câble.

On coupe d'abord le câble à dimensions. Pour cela, nous mesurons les longueurs spécifiées et coupons les tronçons à ces longueurs. Avant de couper, nous contrôlons exactement si les longueurs sont correctes. Et testons chaque câble quand il est prêt. Pour cela, nous avons besoin de l'accumulateur et de la lampe. Si la lampe s'allume quand on la raccorde à l'accumulateur, c'est que le câble est en ordre. Si l'assignation des couleurs est aussi correcte, fiche rouge avec câble rouge, fiche verte et câble vert, nous mettons le câble sur le côté et contrôlons le suivant.

Si le programme écrit (ainsi que celui contenu dans le coffret) ne tourne pas avec notre maquette, nous lançons le diagnostic d'Interface. Ce programme de service nous permet de tester séparément les entrées et les sorties. Ici, chaque capteur doit déclencher l'action correspondante à l'interface.

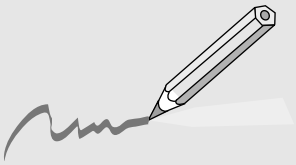
Après ce contrôle, nous savons que tout le système électrique devrait être en ordre. Au moyen des boutons de commande des moteurs nous mettons en marche et arrêtons séparément chacun des actionneurs. Si tout est en ordre ici aussi, nous chercherons la cause mécanique.

Les mauvais contacts constituent un défaut sournois. D'une part, les fiches de raccordement peuvent être lâches dans les prises. Si c'est le cas, on en élargit un peu les lames de contact à l'aide d'un tournevis d'horloger. Attention, les contacts peuvent casser si on les élargit trop, ou bien ils s'emboîtent très mal.

Une autre cause de mauvais contacts sont les liaisons par serrage qui se sont desserrées aux points de vissage des fiches. Serrer le vissage avec prudence ! À cette occasion, on vérifie qu'aucun des minces fils de cuivre n'est cassé. S'il se produit des ratés inexplicables durant le fonctionnement, la cause peut en être un accumulateur presque vide. La tension tombe brièvement quand on branche une charge (mise en marche d'un moteur), ce qui déclenche une initialisation de l'ordinateur sur l'interface. Un tel défaut est parfois très difficile à localiser puisque le programme se met toujours d'abord à fonctionner.

S'il apparaît des défauts inexplicables avec les programmes qu'on a écrit soi-même, on devrait par précaution faire tourner un des programmes fournis qui soit aussi semblable que possible au programme écrit, ce qui permettra d'éliminer les défaut électriques ou mécaniques.

Si ces mesures n'ont pas de succès, il reste encore la possibilité de prendre contact avec le service après-vente de fischertechnik.



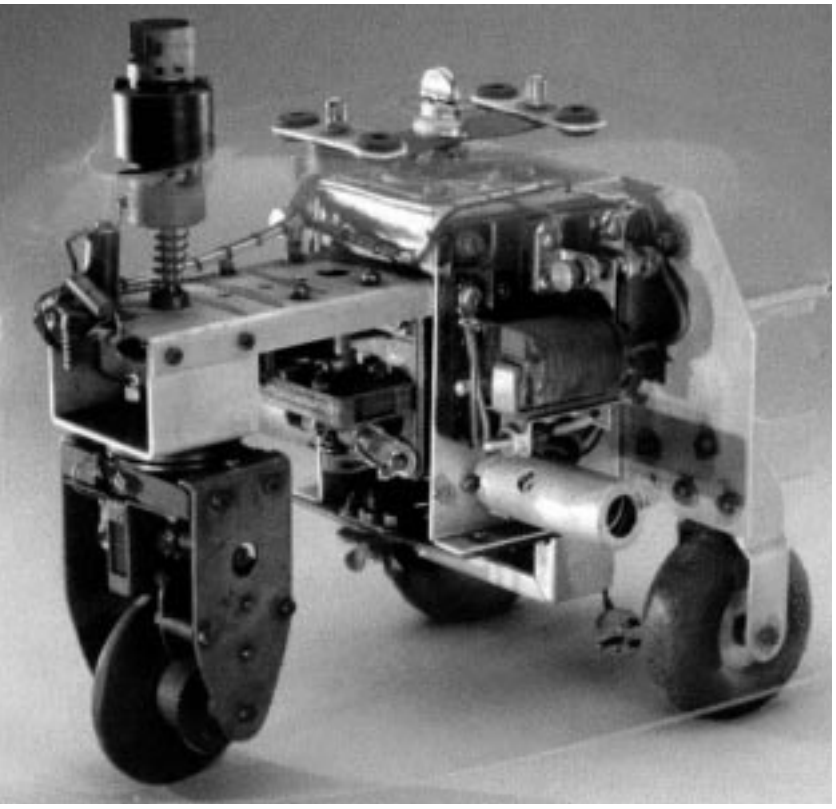
(E) CONTENIDO

E

1 ¿Para qué son útiles los robots?

Antes de empezar a tratar prácticamente la técnica robótica, intentaremos dar respuesta a la provocativa pregunta que antecede.
El término "robot" aparece por primera vez en 1923 en la novela "Golem" de Carel Capek. Esta tenebrosa figura creada artificialmente tenía por misión utilizar sus facultades para llevar a cabo trabajos que el hombre hacía. Como ocurre a menudo con los personajes literarios, este se encuentra sujeto la voluntad del autor quien le confiere una forma de ser que provoca cierto recelo. En los años 30 y 40 del siglo XX el robot se convierte en una suerte de autómata. Las múltiples tentativas de conferirle rasgos humanos, como por ejemplo una cabeza con lámparas intermitentes en los ojos o primitiva respuesta vocal a través de un altavoz nos parecen hoy en día un tanto ingenuas. Evidentemente no es fácil desvirtuar los temores de una potencial dominación de los robots sobre el hombre.

Sin embargo, en los primeros robots construidos se observa muy poca movilidad y aún menos inteligencia. Recién con la llegada de los circuitos electrónicos se puede hablar de una construcción realista de robots.
El problema de los principios de control necesarios está íntimamente ligado a la técnica robótica en sí. Hoy en día, en muchas empresas, institutos y universidades, la "inteligencia artificial" sigue siendo un tema de investigación y estudio.



De la cibernética se esperaban los primeros visos de solución. El término "cibernética" se deriva de la palabra griega kybernetes. El kybernetes era el navegante de las naves de remos griegas. Su obligación era determinar la posición de la nave y calcular el curso hacia el destino.
Así nos podemos dar cuenta, que se esperaba que la cibernética concediese "inteligencia" a los robots. Pero, ¿a qué nos referimos cuando hablamos de conducta inteligente?
Intentaremos aclarar esto con la ayuda de un experimento mental.

Ciertamente todos hemos observado alguna vez el comportamiento de una polilla volando alrededor de una bombilla. La polilla reconoce la fuente de luz, vuela hacia ella y se desvía antes de chocar con la lámpara. Esta claro que para este comportamiento la polilla debe reconocer la fuente de luz, calcular el trayecto y a continuación volarlo. Estas facultades están condicionadas a los patrones de comportamiento instintivos e inteligentes del insecto.
Ahora, tratemos de trasladar estas facultades a un sistema técnico. Tendremos que reconocer la fuente de luz (sensores ópticos), llevar a cabo un movimiento (controlar motores) y debemos establecer una relación razonable entre el reconocimiento y el movimiento (programa). Nuestro experimento mental combina ahora un sensor óptico con un motor y una lógica, de modo que este vehículo dirija siempre al motor en dirección de la fuente de luz. ¿Se comportará este vehículo con ello de manera idéntica que una polilla? En los años 50 el británico Walter Grey llevó a cabo técnicamente el experimento que acabamos de describir. Sirviéndose de simples sensores, motores y circuitos eléctricos se crearon diferentes animales "cibernéticos" que poseían un comportamiento muy específico, como por ejemplo el de una polilla. Estas máquinas se constituyeron en un eslabón decisivo hacia los modernos robots móviles. Los sensores (fotorresistencia, captador,...) de los aparatos controlaban, sirviéndose de su electrónica, los activadores (motores, relees, lámparas,...) de manera que se daba lugar a un comportamiento (¿aparentemente?) inteligente.
La ilustración se muestra el modelo de una tortuga "cibernética", la misma que se encuentra expuesta en el Smithsonian Museum en Washington.
Sobre la base de estas reflexiones, fijamos para nuestros robots los correspondientes "patrones de comportamiento" y los convertimos en programas a fin de que los robots los entiendan.
¿Nos facultan estas reflexiones para responder la pregunta acerca de la utilidad de los robots móviles? Para dar una respuesta concreta a esta pregunta, intentaremos a continuación aplicar el hasta ahora abstracto comportamiento de nuestra "polilla mental" a ámbitos técnicos. Un buen ejemplo de ello es la búsqueda de luz. Modificaremos la fuente de luz colocando una línea clara, la línea guía, sobre el suelo y orientando los sensores hacia abajo y ya no hacia delante. Sirviéndose de tales líneas guías, un robot puede orientarse por ejemplo en un almacén. Algunas informaciones adicionales, por ejemplo en forma de código de barras en determinados puntos de la línea, hacen que el robot ejecute allí otras acciones, como por ejemplo la carga o descarga de una paleta. Tales sistemas robóticos ya existen. En grandes hospitales existen extensos trayectos de transporte para materiales de uso, como por ejemplo ropa de cama. El transporte de estos materiales mediante el personal lleva mucho tiempo e implica también un duro trabajo corporal. Además, tales trabajos acortan el tiempo que se puede dedicar al cuidado de los pacientes.
Es así que admitimos que los robots se pueden adjudicar un importante lugar en la sociedad moderna. Pero ¿cuál es la relación de esto con el sistema modular de construcción de Fischertechnik?

Para construir un modelo de robot se necesita, aparte de sensores y ectivadores, muchas piezas mecánicas. El sistema modular de construcción Mobile Robots II de fischertechnik es la base ideal para ello. Podemos combinar las piezas mecánicas de casi infinitas maneras para obtener vehículos robotizados robustos. Con el correspondiente "Intelligent Interface" (Número de artículo 30402, disponible opcionalmente) disponemos de la

capacidad informática que nos permitirá diseñar programas minuciosos. A través de esta interfaz se efectúa el acoplamiento y evaluación de un gran número de diferentes sensores y activadores. Los sensores convierten magnitudes físicas, tales como cantidad de luz o temperatura, en valores eléctricos evaluables. Entre ellas cuentan las magnitudes analógicas y las digitales. Las unidades digitales son aquellas que pueden considerarse sea como lógicamente verdaderas o como lógicamente falsas. Estas opciones se marcan con 0 ó con 1. Un interruptor es un buen ejemplo de un sensor digital.

Sin embargo, muchas unidades oscilan continuamente entre sus valores extremos, éstos se denominan valores analógicos.

Estos valores no pueden ser definidos simplemente como 0 ó 1. Para que un ordenador pueda procesarlas, estas unidades deberán convertirse en valores numéricos correspondientes. La interfaz de fischertechnik dispone para este fin de dos entradas analógicas: EX y EY. La resistencia que entra a estos contactos se guarda en un valor numérico. Los valores de medición de un sensor de temperatura, por ejemplo 0...5 kΩ, son clasificados en un ámbito de 0...1024 y se encuentran listos para su procesamiento posterior.

La función más importante de la interfaz es el enlace lógico de las unidades de entrada. Para ello, la interfaz necesita un programa. El programa decide de qué manera se convertirán los datos de entrada en señales de sensor, en datos de salida correspondientes, en señales de control de motores, etc. Para poder elaborar, en lo posible eficazmente, los programas necesarios de interfaz existe una superficie gráfica de programación. El término "superficie gráfica de programación" supone un software que nos posibilita elaborar programas a un nivel muy alto. Con la ayuda de símbolos gráficos podemos concebir programas o algoritmos. El ordenador del Intelligent Interface puede ejecutar únicamente comandos realizados desde el llamado conjunto de órdenes de máquina. Estos son por lo general simples estructuras de control lógico o aritmético, cuya aplicación es extremadamente compleja para principiantes. El software para PC LLWin (Número de artículo 30407, no incluido en el sistema modular de construcción) proporciona por ello elementos gráficos, que posteriormente se traducen en un lenguaje ejecutable para la interfaz.

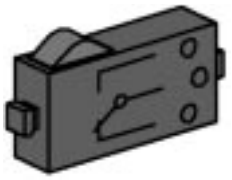
Nos sumergiremos a continuación paso a paso en el fascinante mundo de los robots móviles. Empezaremos con una construcción de prueba para comprobar las funciones básicas de la interfaz y de los elementos sensoriales. Después empezaremos con modelos simples a los cuales se asignarán determinadas tareas, probando sistemas cada vez más complejos. Con la finalidad de evitar situaciones frustrantes a causa de la aparición de fallos, hemos dedicado un capítulo entero a la familiarización con las características y singularidades de los sensores y activadores. Para problemas más "serios" al final hemos adjuntado la sección "Búsqueda y eliminación de fallos".

Un punto vital es el esmero en la construcción y en la puesta en marcha de nuestros robots. Trataremos con máquinas complejas, cuya única diferencia con los verdaderos sistemas robóticos es su tamaño relativamente reducido. En la construcción de los componentes eléctricos, nos atendremos a las instrucciones, probando de preferencia dos o tres veces si todo está en orden. En cuanto a las construcciones mecánicas tendremos cuidado, también en las creaciones propias, de que se muevan sin dificultad y que no posean demasiado juego en los engranajes y sujeciones. No aplicaremos nunca la "fuerza bruta" durante la construcción. Depende de nuestra creatividad elaborar nuevos programas a fin de definir nuevos "comportamientos" cuya complejidad esté limitada únicamente por los recursos de que se disponga en cuanto a memoria y potencia del ordenador. Los ejemplos que siguen dan algunas sugerencias al respecto.

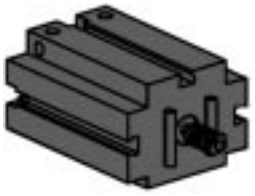
2 Primeros pasos

Después de las reflexiones teóricas, llevaremos a cabo por fin algunos experimentos. Muchos de nosotros quisiéramos empezar de inmediato a construir, incluso quizás la compleja carretilla elevadora automática de horquilla. Por supuesto que esto es factible, y el éxito de la construcción del modelo está fundamentado en la esmerada observación de las instrucciones. ¿Y que hacer cuando no funciona? En estos casos se deberá buscar sistemáticamente la causa del fallo. Durante esta búsqueda emergen inevitablemente preguntas acerca del funcionamiento y características de los componentes utilizados. Evidentemente, es indispensable disponer de conocimientos básicos de sensores y activadores. Sin embargo, antes de empezar a familiarizarnos con estas cosas, probaremos el funcionamiento armónico del ordenador y de la interfaz. El software de control se instala en el PC según las instrucciones dadas en el manual de LLWin.

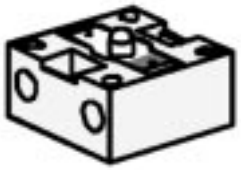
Sirviéndose del diagnóstico de interfaz probaremos los diferentes sensores o activadores. Podemos, por ejemplo, conectar un sensor on dos cables a la entrada E1 y observar que estado de circuito reconoce la interfaz. Si se acciona el palpador deberá ocurrir una modificación del estado en la entrada correspondiente.



Comprobaremos las salidas, conectando un motor con una salida de motor, por ejemplo M1. Con el ratón podremos poner en funcionamiento el motor. Si deseamos probar la entrada analógica, utilizaremos para este efecto un fototransistor como sensor analógico.

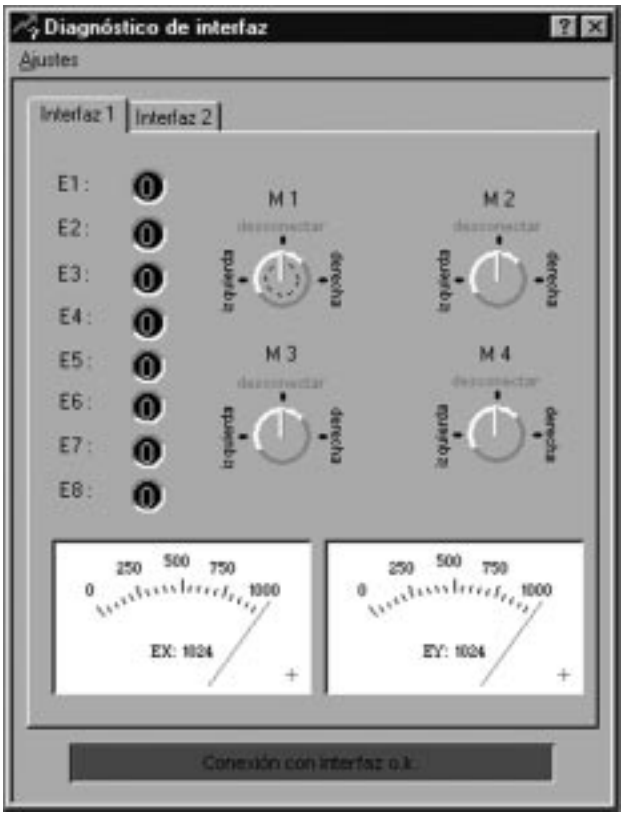


Mientras que en el caso del motor o del palpador la polaridad de las conexiones no tiene importancia alguna (en el peor de los casos el motor girará en sentido invertido), para el correcto funcionamiento del fototransistor es absolutamente necesario efectuar la conexión adecuada. Un contacto del transistor dispone de una marca roja, a este contacto se deberá conectar el enchufe rojo. El contacto sin marca alguna está previsto al enchufe verde. El segundo enchufe rojo entra en la toma de la entrada EX ubicada cerca al borde de la interfaz, en segundo enchufe verde entra en la toma EX ubicada hacia adentro. A continuación, con una linterna podemos variar la potencia lumínica del fototransistor y de esta manera modificar la desviación.



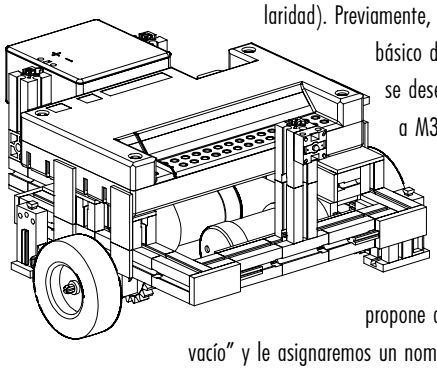
Si la aguja no se moviese de su posición máxima, volveremos a controlar las conexiones del fototransistor. Si por el contrario, la aguja se encontrase

en cero con la linterna apagada, puede ser que haya demasiada iluminación en la habitación o que la claridad del entorno sea demasiado grande. La desviación de la aguja cambiará si cubrimos el fototransistor.

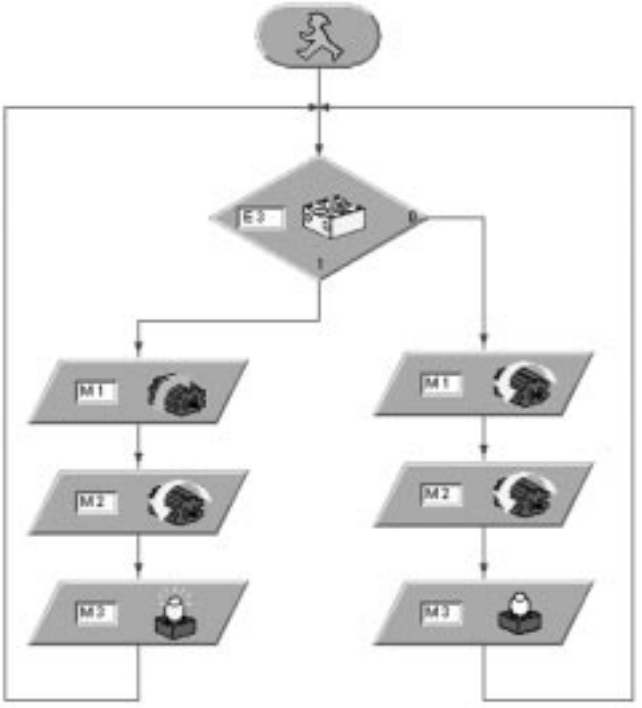


Repasemos brevemente la asignación de color de los enchufes: durante la construcción observaremos rigurosamente que se conecte un enchufe rojo al cable rojo y un enchufe verde al cable verde. De usar señales polarizadas en una construcción de circuitos, tomaremos siempre un cable rojo para el polo positivo y un cable verde para el polo negativo.
Esto puede parecer un tanto exagerado (y a la electricidad no le importa el color del cable), pero para la búsqueda sistemática de fallos la asignación de colores es de gran utilidad.

Queremos concluir los primeros pasos en el ámbito de la robótica con un programa sencillo. Construiremos el modelo básico con ambos motores de tracción y la rueda de soporte según las instrucciones de ensamblaje. Conectaremos únicamente los motores a las salidas M1 y M2. Además buscaremos el fototransistor y lo conectaremos a la entrada E3 (observar la polaridad). Previamente, instalaremos el fototransistor al modelo básico de tal manera que "mire hacia adelante". Si se desea, se puede conectar una lámpara de lente a M3 y fijarla por ejemplo junto al paquete de baterías, pero esto no es absolutamente necesario. Abriremos el programa LLWin y crearemos un nuevo proyecto (PROYECTO – NUEVO). LLWin nos propone diferentes plantillas, elegiremos "proyecto vacío" y le asignaremos un nombre, por ejemplo "Step1". Luego de haber pulsado la tecla [Aceptar], aparecerá una hoja de trabajo vacía con un hombrecillo de semáforo y la ventana del módulo.



El hombrecillo verde simboliza el inicio del programa. Todos nuestros programas se inician desde este punto de partida.



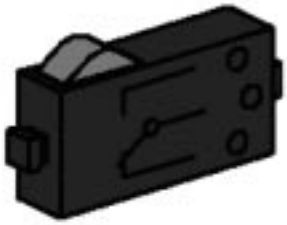
Con el ratón podremos arrastrar los diferentes segmentos de programa desde la ventana del módulo. Los símbolos que se encuentran allí representan entradas y/o salidas de la interfaz. Con la tecla izquierda del ratón podemos colocar el símbolo deseado y con la tecla derecha modificaremos las propiedades. El símbolo de palpador representa una entrada. Para nuestro programa, utilizaremos el ratón para colocar el pulsador debajo del símbolo de inicio. Cuando soltemos el símbolo, aparecerá un diálogo de opciones. Elegiremos el fototransistor. Si posteriormente se desea otras modificaciones, podremos activar este diálogo con la tecla derecha del ratón. Asignaremos las salidas a los motores y estableceremos la dirección de rotación deseada. Queremos que los motores giren en un mimo sentido cuando el fototransistor no reciba luz y que giren en sentido contrario cuando reconozca la luz. Uniremos los elementos con la función de dibujo. La lámpara en M3 señaliza el estado del fototransistor. La ilustración muestra la conexión exacta de las ramas del programa. Quien no esté seguro de que todo está correcto, deberá comparar su programa con el programa Step1.mdl. A tal fin, deberá guardar previamente el programa propio y cargar el archivo Step1.mdl desde el CD-ROM incluido con el módulo de construcción. Si todo estuviese correcto, el programa será descargado a la interfaz y se iniciará de inmediato (INICIAR - DESCARGA).
Nuestro primer robot comenzará a girar sobre su posición. Continuará girando hasta que lo estimulemos con una fuente de luz. Apenas reconozca el fototransistor la luz, los motores que estaban rotando en diferentes sentidos el mismo lo harán ahora en el mismo senfido y el robot irá directamente hacia la fuente de luz. En caso que se aleje de la fuente de luz, deberemos cambiar la polaridad de los motores. Es probable que su trayecto no sea completamente recto, de tal modo que después de algún trecho el fototransistor pierda el contacto con la fuente de luz. A continuación su movimiento será conmutado de Andar adelante a Girar, volviendo a empezar la búsqueda de luz. Para experimentar exitosamente, habremos hecho anteriormente suficiente espacio, ya que nuestro robot (todavía) no puede percibir obstácu los en su camino.

3 Sensores y activadores

Ahora sabemos que nuestro elemento principal, la interfaz, "armoniza" con el PC. Ahora se trata de determinar como reconoce nuestra interfaz las señales del entorno. Empezaremos con las entradas. En el lenguaje técnico las entradas o las señales de entrada se denominan input. Para un ordenador, y el Intelligent Interface es uno, dispone únicamente de la posibilidad de reconocer y procesar señales eléctricas. Por esta razón deberemos "computarizar" los estímulos del entorno. Por ello todos los sensores son convertidores que convierten el "sentido" determinado en una señal eléctrica. Ya que no es nuestro deseo construir "a ciegas" según las instrucciones, es recomendable informarse de las propiedades básicas de los sensores existentes.

Este aspecto gana importancia ya que en el futuro la interfaz puede complementarse con nuevas aplicaciones creadas por uno mismo.

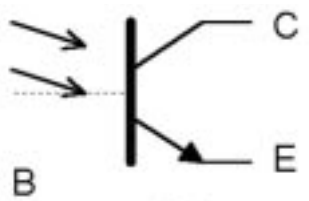
3.1 El interruptor como sensor digital

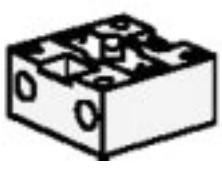


Las marcas lógicas simples "0" y "1" se pueden representar con la ayuda de un interruptor. En el sistema modular de construcción de fischertechnik se utilizan interruptores de salto de precisión con contactos de conmutación. La singularidad de los interruptores de salto yace en su comportamiento de conmutación. Al accionar lenta y cuidadosamente el pulsador se siente claramente un punto de presión, que al ser transpuesto conmuta el contacto con una suave crepitación. Si volvemos a soltar lentamente la leva de conmutación, deberemos hacerla pasar claramente más allá de punto de activación original a fin de alcanzar un restablecimiento de conmutación originario. Esta diferencia entre las posiciones mecánicas de encendido y apagado en conocida bajo el nombre de histéresis. La histéresis de conmutación de contactos o de otros circuitos electrónicos es una propiedad importante. De no existir ésta, es decir si el punto de encendido fuese el mismo que el punto de apagado, habría grandes problemas en la evaluación de las señales. Las perturbaciones más ínfimas, como un ligero temblor en el punto de conmutación, podría señalizar varios accionamientos a la interfaz y no sería posible un recuento exacto de eventos. El conmutador ha sido construido como conmutadorinversor. Con ello podemos evaluar en nuestros experimentos ambas posiciones originarias de conmutación, es decir estado de reposo abierto o estado de reposo cerrado.

3.2 Fotodetección con el fototransistor

El fototransistor es un dispositivo semiconductor, cuyas propiedades eléctricas dependen de la intensidad luminosa. Un transistor es un componente con tres contactos. Estos contactos se denominan Emisor, Base y Colector. Su tarea principal es la amplificación de señales débiles. Una corriente débil que fluye desde una señal en la base del transistor posee en consecuencia una corriente mucho más fuerte en el colector del transistor. La amplificación de flujo puede alcanzar factores superiores a 1000. Sin embargo, el fototransistor de sistema modular de construcción posee únicamente dos contactos. ¿Dónde se encuentra el tercero?

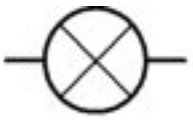




Nuestra intención es que nuestro transistor reconozca la luz. Todos conocemos las celdas solares con las que se obtiene electricidad a partir de la luz solar. El fototransistor deberá entenderse como una combinación de minicelda solar y transistor. El contacto base no llega hasta el exterior (por eso es una línea intermitente en la ilustración). En su lugar, los impulsos luminosos que llegan (fotones) generan un flujo luminoso muy pequeño que el transistor entrega amplificado en el colector. Para que esto funcione según lo descrito, el fototransistor requiere de una disposición exterior adicional. Ésta no nos tiene que preocupar, puesto que ya se encuentra incluida en la interfaz. El fototransistor puede utilizarse tanto como sensor digital como sensor analógico. En el primer caso sirve para reconocer transiciones de claridad-oscuridad, por ejemplo una línea marcada. Pero también se pueden diferenciar intensidades luminosas. El sensor trabaja entonces como sensor analógico.

3.3 Salida de señal con la bombilla

La bombilla sirve como una señal luminosa simple. Para continuar utilizando términos técnicos: la bombilla se convierte en activador óptico. La estructura de una lámpara incandescente es muy simple. En una ampolla de vidrio, dentro de la cual se ha generado un vacío, se dispone de un filamento en espiral de alambre delgado de tungsteno dispuesto sobre dos patillas de contacto. Cuando la electricidad fluye a través del filamento, el alambre se calienta al rojo blanco. Ya que en la ampolla de vidrio no hay oxígeno, el alambre no se quema y la lámpara es de gran duración. Como consecuencia de del esfuerzo térmico del filamento de la lámpara, el alambre se dilata con cada encendido y se contrae al apagar la bombilla. Estos diminutos movimientos traen consigo que el material se fatigue y que la lámpara en algún momento se "queme".



La lámpara incandescente se puede usar para mostrar estados de conmutación. Mediante la programación de una lámpara intermitente se pueden generar avisos de emergencia. Existe un caso más en el cual podemos utilizar la lámpara. Conjuntamente con dos fototransistores se crea un sensor especial con cuyas propiedades se reconocen líneas. La lámpara funciona como fuente de luz, de modo que el fototransistor reconozca una marca de color a través de las diferentes intensidades de luz reflejadas. Una singularidad en la lámpara incandescente utilizada en el sistema modular de construcción es el lente óptico incorporado en la ampolla de vidrio. Mediante él, el haz de luz se agrupa mejor haciendo más confiable el reconocimiento, por ejemplo de marcas.

3.4 Motores de corriente continua como fuente de energía

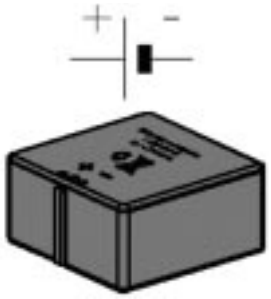
Los motores de corriente continua son activadores importantes en los sistemas móviles. El sistema modular de construcción "Mobile Robots II"

contiene dos tipos de motores. Aunque mecánicamente son muy diferentes, su estructura eléctrica es idéntica. Los motores de corriente continua están conformados por un "rotor" que gira y un "inductor" fijo. Básicamente el

rotor es un bucle de conductor que se encuentra en el campo magnético del inductor. Al fluir electricidad a través del bucle de conductor se origina una fuerza que trae consigo la desviación del conductor en el campo magnético y el rotor se mueve. En aplicaciones prácticas el bucle de conductor tiene la forma de una bobina (con o sin núcleo de hierro para la amplificación del campo magnético). Muchos motores de corriente continua generan el campo magnético necesario sirviéndose de imanes permanentes que se encuentran pegados en la cobertura metálica del inductor. La alimentación eléctrica del rotor giratorio tiene lugar mediante contactos deslizantes. Estos contacto son a la vez responsables del cambio de dirección en el bucle de contacto, el mismo que es necesario para un movimiento permanente e ininterrumpido. Las revoluciones por minuto de los motores comunes se encuentran en el ámbito de varios miles. Un mecanismo de transmisión proporciona gran momento de torsión a bajas revoluciones. El sistema modular de construcción contiene dos tipos diferentes de motores, el minimotor y el motor de potencia. El pequeño y compacto minimotor con tornillo sin fin ha sido diseñado para mecanismos auxiliares de transmisión o trabajos específicos con pequeñas exigencias de rendimiento. Este motor necesita siempre un mecanismo de transmisión para la reducción de las revoluciones. El motor de potencia proporciona momentos de torsión mucho mayores. Este motor dispone de un mecanismo de transmisión con bridas con un engranaje de reducción de 50:1. Con estas características es un motor ideal para las exigencias de tracción de nuestros robots. Asimismo, existe una variante de este motor con un engranaje de reducción de 8:1 (no incluido en este sistema modular de construcción). Sin embargo, las revoluciones en el eje de transmisión de este motor son demasiado altas para la tracción de robots.

3.5 Alimentación eléctrica

Los sistemas móviles requieren alimentación eléctrica independiente. Todas las piezas eléctricas son alimentadas desde aquí. Las exigencias de alimentación eléctrica son diversas. Mientras que a los motores de tracción les



basta tensiones no estabilizadas, muchos sensores requieren tensiones estables para poder entregar resultados exactos. Por razones económicas, el uso de baterías o pilas recargables son la única alternativa viable para alimentar eléctricamente los robots móviles. Las celdas solares o las pilas de combustible no son lo suficientemente potentes para proporcionar resultados practicables con adecuada aplicación. Se deberá preferir pilas y baterías recargables pues ofrecen la posibilidad de recarga. El bloque de pilas de fischertechnik es una buena alternativa en cuanto a la relación rendimiento y tamaño. El bloque de pilas no es parte del sistema modular de construcción y se encuentra disponible junto con un cargador especial en forma de un "Accu Set" bajo el número de artículo 34969. La ilustración muestra el símbolo gráfico y la pila. Por lo general, en el símbolo gráfico no se detalla la polaridad. Sirviéndose de una ayuda mnemotécnica es muy fácil darse cuenta cuál es el polo positivo: "Cortando la línea larga se puede formar un símbolo de más." Un punto vital es el observar la correcta polaridad al conectar la fuente de energía.

3.6 Sensores adicionales

Es relativamente fácil ampliar el sistema fischertechnik con sensores adicionales. Lo más fácil es utilizar sensores de otros sistemas, como por ejemplo el sensor térmico o el sensor magnético del sistema modular de construcción "Profi Sensoric" con el número de artículo 30491. Pero también podemos utilizar sensores completamente diferentes. Las tiendas del ramo ofrecen sistemas y componentes de construcción muy variados. Incluso se pueden utilizar sensores sofisticados, tales como alarmas de gas o sensores por radar. Puesto que no es nuestra intención destruir la Intelligent Interface con tensiones de entrada demasiado altas o cargas inadecuadas, recomendamos la realización de proyectos propios únicamente a quienes dispongan de la experiencia necesaria. La manera más segura de conectar sensores adicionales es el aislamiento galvánico entre el sensor y la interfaz. Particularmente adecuados son los sensores que disponen de un relee. Los contactos de circuito del relee se instalan como un contacto de fischertechnik común y están a partir de ese momento listos para señalizar la aparición de nuevos estímulos en el entorno. Consejo: aficionados entusiastas de la fischertechnik publican en Internet muchos trabajos de complementación del sistema.

4 Los modelos de robot

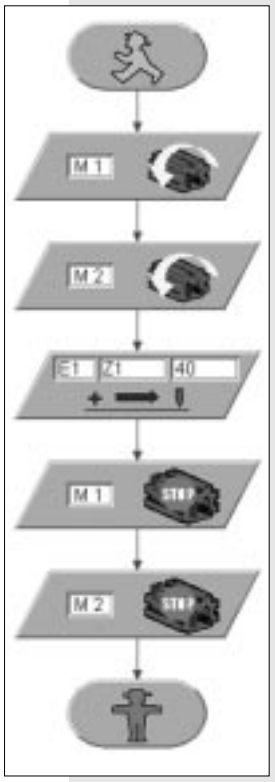
A continuación se presentan algunas variantes de robots móviles autónomos a manera de sugerencia de construcción. Empezaremos con un modelo sencillo. En base a él, podrás reconocer y probar la utilización de diferentes sensores. Aquí se trata de combinar los estados internos del robot, como por ejemplo medición de trayectoria mediante ruedas de impulso, así como las señales externas del entorno, como la búsqueda de luz o de pistas. Se darán determinadas tareas para cada modelo. Éstas te servirán como sugerencia y para ayudarte a familiarizarte con la materia. Los programas de LLWin correspondientes a cada una de las tareas se encuentran en el CD-ROM que viene con el sistema modular de construcción. Te sugerimos crear también algunas tareas propias. El modelo más sencillo es el modelo básico. En él se ensamblan los motores con la interfaz en una unidad compacta. La fuerza motriz del robot está dada por dos motores. Éstos están dispuestos de tal modo que cada uno impele una rueda propulsora. El robot cuenta con una rueda de apoyo que le brinda estabilidad y evita que se vuelque. Tal disposición de los motores es denominada differential drive. Ella concede movilidad elevada en un mínimo de espacio necesario. Incluso es posible efectuar girar sobre el mismo punto. El punto medio de ambos motores es aquí el punto de giro, alrededor del cual se mueve el robot. De esta manera le es posible navegar con escaso esfuerzo de cálculo bajo las condiciones más difíciles. Los motores pueden impulsar las ruedas mediante dos diferentes engranajes de reducción. (lento 100 : 1 ó rápido 50 : 1). Para la variante lenta, la tracción viene desmultiplicada adicionalmente con engranajes fischertechnik en la relación de 2:1. En cada modelo se especifica la reducción utilizada.

4.1 Modelo básico

Primeramente construiremos el modelo básico (engranaje de reducción 100:1) según las instrucciones de ensamblaje. Ya que este modelo nos servirá de base en muchos experimentos, tendremos especial esmero durante la construcción. Una vez concluido el ensamblaje mecánico verificaremos que los motores funcionen sin dificultad. Para ello conectaremos brevemente cada motor directamente a la batería sin usar la interfaz.

Tarea 1:

Programa la interfaz de tal manera que el modelo se desplace recto 40 impulsos. Utiliza para la medición de los impulsos el contador E1 y el pulsador E8 para la puesta a cero. Modifica a continuación el programa de tal manera que el modelo se desplace sobre trayectorias de diferente magnitud, por ejemplo 80 cm. ¿De qué tamaño es aquí la repetibilidad de trayectoria?



Tarea 2:

El modelo deberá girar 180° después de haberse desplazado 80 impulsos. Observa las diferentes direcciones de giro de los motores de tracción durante los movimientos de desplazamiento recto o de giro.

Solución:

En promedio, el modelo recorre 1 cm de trayectoria por cada impulso de recuento. La repetibilidad de trayectoria posee la misma magnitud, aproximadamente 1 cm en 80 impulsos. Sin embargo, varía dependiendo de la superficie sobre la cual se mueve el robot. Particularmente desfavorables son las superficies rugosas o textiles vellosos.

Antes de entregarnos a estas tareas, quisiéramos aclarar dos cosas. Por un lado hemos utilizado un nuevo componente funcional en nuestro programa, el componente POSICION. Este es un componente que permanece activo hasta que se reconozca el número configurado de impulsos en la entrada determinada (aquí E1). Desde la perspectiva del programa, esto significa que utilizamos una condición definida de espera. En la primera prueba hemos utilizado esta función para la medición de la trayectoria de desplazamiento recto. Si el robot tiene que girar, se trata del mismo principio. Únicamente deberemos modificar la dirección de giro de los motores. Ahora solamente nos falta ingresar el número de impulsos y nuestro robot girará sobre su posición.Vayamos ahora al segundo punto por aclarar. No queremos simplemente seguir probando hasta que el robot gire 180°; queremos calcular previamente este valor. Los motores de tracción están configurados como un differential drive, es decir que las ruedas del robot se mueven durante el giro sobre una circunferencia cuyo diámetro está dado por la distancia entre las ruedas. Por consiguiente, para un giro de 180° cada rueda debe recorrer una trayectoria de exactamente la mitad de tal circunferencia.

Calcularemos primeramente la circunferencia u:

$u = \pi \cdot d = 630mm$
d - diámetro (distancia entre ruedas, aproximadamente 200 mm)

Anteriormente habíamos determinado una trayectoria de aproximadamente 1 cm por impulso, consiguientemente necesitaremos para la trayectoria de 314 mm (la mitad de la circunferencia) 30,5 impulsos. Ya que podemos calcular únicamente valores en números enteros, nos deberemos decidir por 30 o 31 impulsos. Probaremos cual de estos valores proporciona mayor exactitud.

Resultado:

El resultado de las mediciones con la rueda de impulso arroja que la exactitud realizable de nuestras mediciones no es muy alta. Especialmente cuando se recorren varias trayectorias repetida o consecutivamente, se acumula el error absoluto de medición. Asimismo es un problema el error resultante de tactos no registrados completamente.
Las posibilidades de minimizar estos errores son limitadas. Por un lado se pueden aumentar los impulsos de trecho por cada unidad de trayectoria. Lo ideal sería que el contador se encontrase directamente en el árbol de transmisión. Además del hecho de no tener acceso al árbol, aquí se suma el problema de la limitada velocidad de muestreo de la interfaz. Cuando dentro de una unidad de tiempo vienen demasiados impulsos, la interfaz "se olvida" eventualmente de algunos. Así, el cálculo exacto del trayecto es ilusorio.
No estamos en capacidad de registrar otros errores, como por ejemplo el resbalamiento de las ruedas sobre superficies desiguales o la divergencia de los diámetros de las ruedas. Nos consolamos con la idea de que no se ha podido dar solución integral a estos problemas en sistemas comerciales mucho más complejos y caros.

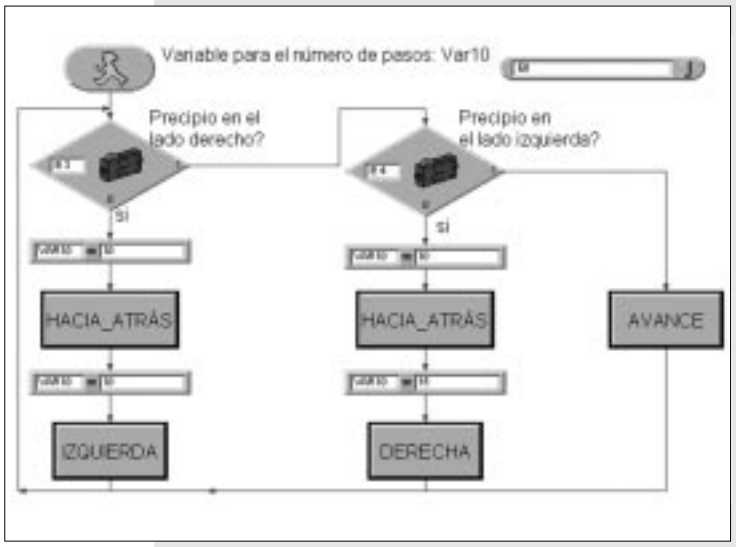
4.2 Robot con detección de bordes

Una vez que hayamos experimentado copiosamente con nuestro modelo básico, intentaremos enseñar al robot lo que significa tener "miedo" al abismo. Hasta ahora el robot se ha movido de un lado para el otro sobre la mesa mientras lo vigilábamos con ojos de lince para que no se caiga de la mesa. Esto no parece ser un comportamiento verdaderamente inteligente. Por ello, haremos algunas modificaciones.
Para que el robot reconozca un borde, requiere de un detector de bordes. Un proceso tan simple y útil necesita dos ruedas auxiliares. Éstas se instalan con un interruptor de manera similar a un sensor en la dirección de desplazamiento del robot. Las ruedas están construidas de tal modo que se pueden mover verticalmente. Un borde hace que la rueda auxiliar caiga y active con ello el sensor.

Tarea 3:

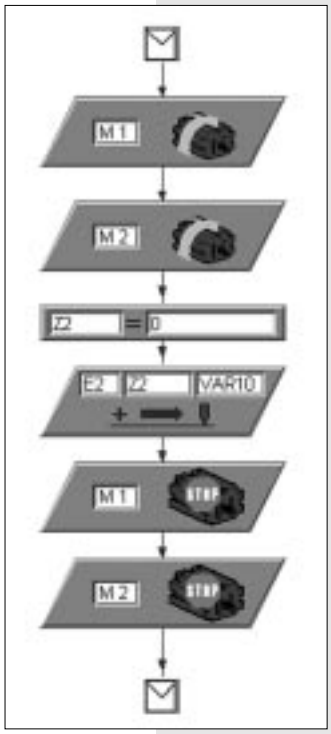
Construye el modelo "Robot con detección de bordes" según las instrucciones de ensamblaje (engranaje de reducción 50:1). El modelo deberá desplazarse hacia adelante. Tan pronto como llegue a un abismo sobre el lado izquierdo, deberá desviarse a la derecha, de haber un abismo a la derecha deberá desviarse a la izquierda. Para disponer de una mejor vista general, algunos movimientos se aplican como subprogramas (adelante, a la izquierda y a la derecha). El contador detectará el número de pasos. El número de pasos se predetermina en una variable VAR10. Esta norma es diferente para los subprogramas a la derecha y a la izquierda.

La diversidad de normas reduce el riesgo de quedarse atascado en una esquina.



Solución:

La tarea nos revela que deberemos controlar el robot en dependencia de los detectores de bordes. Por esta razón, fraccionamos la tarea en partes pequeñas. Primeramente consultaremos los sensores (detectores de bordes). Si ningún sensor estuviese activo, el robot se desplaza hacia adelante. En la ilustración esto se muestra como bloque "Adelante". Detrás de cada nuevo componente se esconde un subprograma.



Los subprogramas mejoran la claridad de sistemas complejos y su uso repetido mejora el aprovechamiento de la capacidad de los sistemas computarizados. Nuestro primer subprograma es muy sencillo, pues pone únicamente en movimiento los motores M1 y M2. Sin embargo, necesitaremos más subprogramas. Según la situación, el robot deberá desviarse hacia la izquierda, hacia la derecha o hacia atrás. En este caso no bastará con detener los motores. Se deberá programar una secuencia determinada de movimientos. Echemos un vistazo a otro subprograma. Con éste subprograma, el robot retrocederá al detectar un borde.

Para ello pondremos los motores en marcha atrás. Nuestra rueda de impulso deberá entonces detectar un trayecto definido. La longitud de trayecto se determina con una variable VAR10. El bloque de asignación Z2 = 0 es nuevo. Luego de reflexionar un tanto, nos damos cuenta del sentido que tiene. Si no ponemos la variable en cero, el mecanismo funcionará solamente una vez, ya que cuando Z2 haya alcanzado el valor de la variable VAR10, el contador de trayecto fallaría en cada uno de los pasos siguientes. Desde la perspectiva del programador profesional, nos encontramos ahora con variables locales y globales. La variable local Z2 se inicia en el subprograma antes de cada uso.

Resultado:

La llamada de subprogramas incrementan la claridad de los programas. Utilizaremos las variables para medir diferentes valores, en este caso la longitud de trayectos. Necesitaremos diferentes longitudes de trayecto para que el robot se pueda "liberar" de rincones. Si los trayectos fuesen absolutamente idénticos, podría ocurrir que el robot se mueva de un lado a otro en un rincón. Al utilizar las variables observaremos sui ámbito de validez. Iniciaremos las variables locales, es decir las variables que se usan dentro de un subprograma, antes de su primera aplicación.

No damos también cuenta que nuestro robot precisa de libertad de movimientos para que funcione correctamente. Si durante su movimiento de desvío el robot encontrase nuevamente un borde, no podría reaccionar ante él. Quienes aceptar grandes retos pueden intentar dar solución a este problema.

4.3 Robot con detección de obstáculos

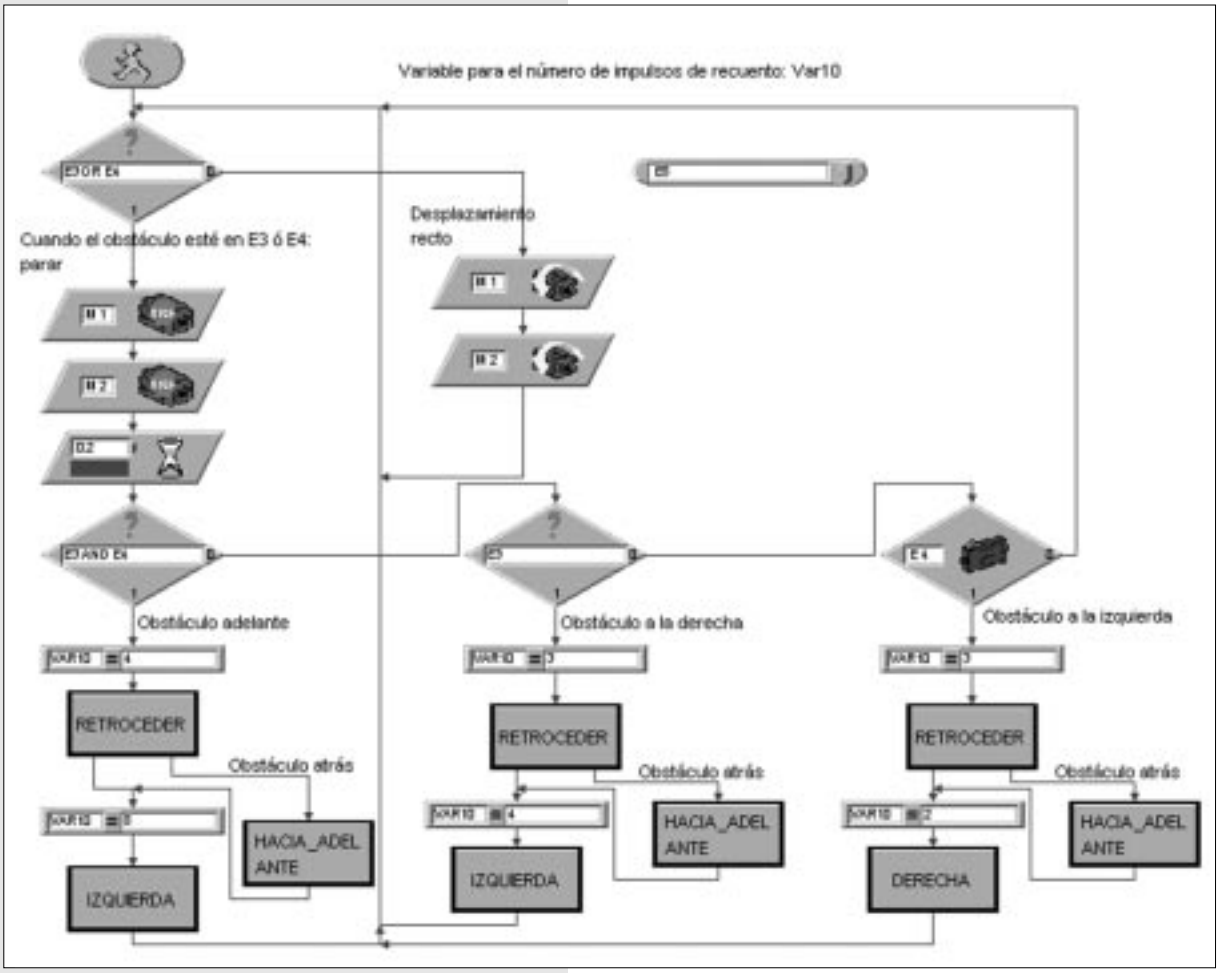
Hasta ahora hemos podido detectar bordes con éxito. Los obstáculos normales representan un problema. Tendremos que modificar el principio de detección de bordes. Un contacto de choque adecuado reemplazará a las ruedas auxiliares. Es ocasión de reflexionar sobre la imperfección del detector de bordes. Existen situaciones peligrosas, durante las cuales el robot puede caer al abismo durante su retroceso por estar "ciego" atrás. Un tercer sensor resuelve tal imperfección. Entretanto, nos hemos convertido en astutos programadores. Esto nos ayudará puesto que la tarea se hace más compleja como se puede observar si echamos un vistazo al programa. El programa contiene nuevos bloques funcionales. Insertaremos el componente ESPERA en determinadas posiciones del programa. Esto es fácil de entender, aquí se esperará el tiempo ingresado, antes de que se lleve a cabo la siguiente función del programa.

Lo verdaderamente nuevo aquí son las ecuaciones lógicas. Hasta ahora habíamos llevado a cabo de inmediato una ecuación al consultar el sensor y dividido correspondientemente nuestro programa. Una ecuación similar existe también en el recuento de impulsos, la ecuación con un número determinado de impulsos. También es nueva la ecuación lógica con varias expresiones en un componente ECUACION.

Tarea 4:

Construye el robot según las instrucciones de ensamblaje con el engranaje de reducción rápido 50:1. El modelo deberá desplazarse hacia adelante. Cuando se detecte un obstáculo en uno de los sensores delanteros (E3 ó E4), el robot se detendrá. Si se detecta un obstáculo a la derecha, el robot retrocede y se desvía hacia la izquierda (aproximadamente 30°). Si se detecta un obstáculo a la izquierda, el robot se desvía hacia la derecha (aproximadamente 45°) después de retroceder. Los diferentes valores de ángulos son necesarios para que el robot salga de un rincón. Si se detecta un obstáculo directamente enfrente, el robot deberá retroceder y desviarse aproximadamente 90°. Si se detecta un obstáculo al retroceder, el robot deberá avanzar un poco y desviarse según lo planeado.

Solución:



Resultado:

Se incrementa la complejidad del programa. Intentaremos hacer frente al incremento de las dificultades con mejores técnicas de programación. Aquí los subprogramas demuestran ser de mucha utilidad. Detectamos una nueva estructura de control en forma de ecuaciones lógicas con la subsiguiente división del programa. También nos damos cuenta que se necesitarán cada vez más sensores para las nuevas propiedades o para llevar a cabo pruebas ya conocidas.

4.4 El detector de luz

Hasta ahora hemos permitido que los robots reaccionen un tanto pasivamente a las señales del entorno. Ahora queremos que los robots busquen activamente. El sistema modular de construcción contiene dos fototransistores que aplicaremos como detectores de luz. Cada sensor envía señales a un motor, haciendo posible la "persecución del haz guía". El programa consta de dos partes. Una de ellas se ocupa de la búsqueda de una fuente de luz y en la otra se lleva a cabo la persecución o navegación hacia la fuente de luz.

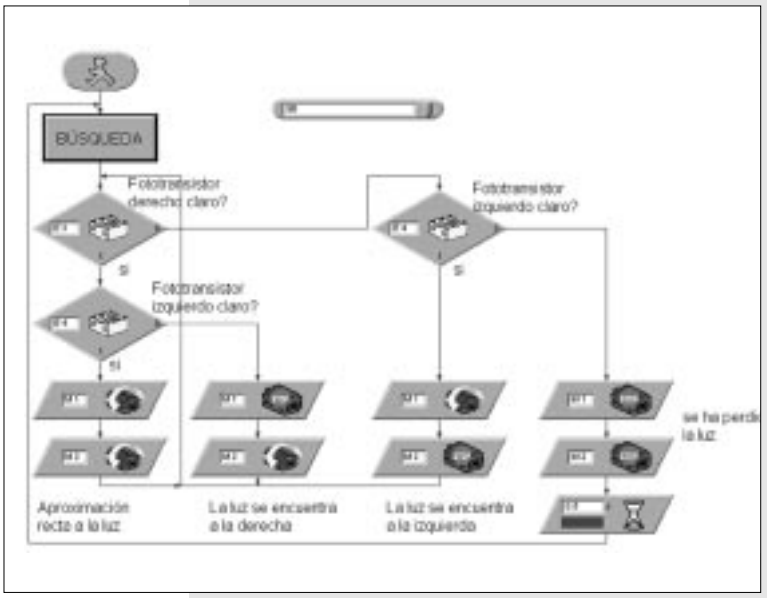
Naturalmente nos volvemos a servir de la técnica de subprogramación. Luego del encendido se activa el subprograma BÚSQUEDA DE LUZ. Este subprograma se abandonará una vez que se haya encontrado una fuente de luz. El programa principal intenta llevar al robot hacia la fuente de luz. Cada vez que la dirección del robot se desvíe fuertemente de la línea ideal, la fuente de luz dejará de iluminar a uno de los sensores de luz. A continuación se detendrá el motor respectivo, de modo que ambos sensores puedan volver a detectar la fuente de luz. De esto se desprende la tarea.

Tarea 5:

Construiremos el modelo "Detector de luz" con la reducción lenta de 100 : 1. Programa primeramente la función "buscar luz" (mejor directamente como subprograma). El robot deberá girar al menos 360° en una dirección. A continuación deberá girar al menos 360° en la otra dirección. Si durante la búsqueda se encuentra una luz, el robot se detendrá. Si durante los giros no se encuentra una luz, se interrumpirá la búsqueda y se terminará el programa. Si se encuentra la luz con éxito, el modelo se dirigirá hacia la fuente de luz. Si la fuente de luz se mueve hacia la izquierda o hacia la derecha, el robot seguirá los movimientos de la luz. Si el robot perdiese contacto con la fuente de luz, el programa iniciará una nueva búsqueda de luz. Prueba si puedes atraer al robot con una linterna y hacerlo pasar a través de una pista de obstáculos.

Consejo:
Utiliza una linterna como fuente de luz. Procura no enfocar el haz de luz demasiado, de manera que ilumines ambos sensores con la fuente de luz. Ten en cuenta que en habitaciones muy iluminadas otras fuentes de luz (por ejemplo la luz del sol a través de una gran ventana) podrían resplandecer sobre tu linterna. Entonces es posible que el robot ignore tu linterna y se dirija hacia la luz más intensa.

Solución:



Resultado:
Hemos construido un robot que reacciona activamente ante su entorno. Sus sensores se encuentran programados para ubicar una fuente de luz y, en caso de haberla localizado, de dirigirse hacia ella o seguirla. Nos damos cuenta que el programa detiene el robot cuando no se encontró ninguna fuente de luz. Cuando por ejemplo un obstáculo muy pequeño impide el contacto visual directo con el robot, éste no detectará ni encontrará la fuente de luz a pesar de haber una. Evidentemente, sería recomendable que después de una búsqueda de luz sin éxito el robot vaya más o menos aleatoriamente a otro lugar y que busque luz allí nuevamente. Pero, ¿qué pasaría si el obstáculo que impide el contacto entre luz y el robot se encontrase directamente en la dirección de desplazamiento del robot? Bueno, esta pregunta parece ser lo suficientemente interesante para ocuparnos de su contenido.

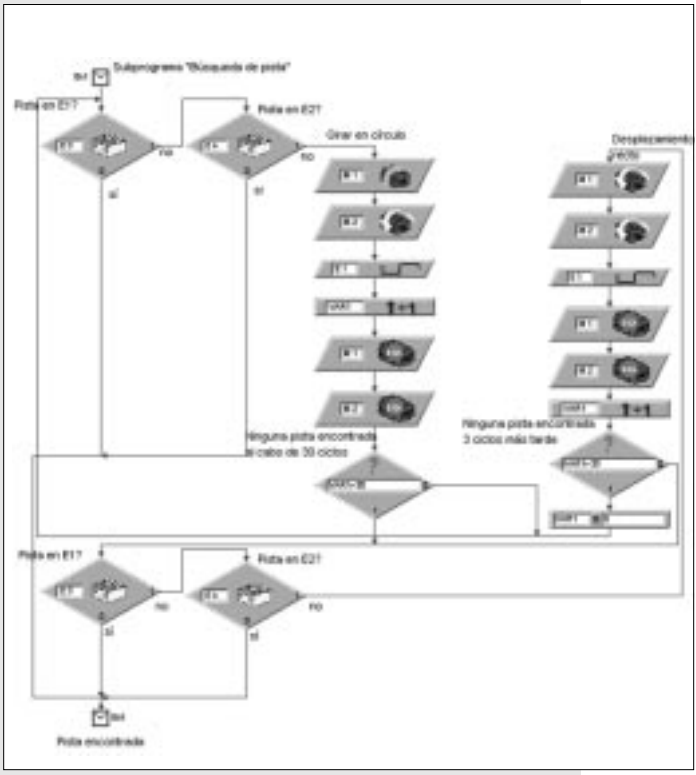
4.5 El detector de pistas

Búsqueda y persecución son propiedades que los seres inteligentes poseen. Hemos construido y programado un robot que ha reaccionado ante señales directas de su destino o víctima potencial.
Con el detector de pistas aplicaremos otro principio de búsqueda. En lugar del desplazamiento directo hacia la fuente de luz, marcaremos una línea que el robot deberá seguir. Con los sensores ópticos esta tarea es relativamente fácil de solucionar. Mediremos la luz reflejada de la marca y corregiremos los motores según ella. Para que esto funcione de manera exacta, iluminaremos la línea con nuestra linterna. Procuraremos siempre colocar la lámpara y los fotosensores de tal manera que la luz dispersa no deslumbre éstos últimos. En estas circunstancias el haz de luz del lente óptico de nuestra bombilla demuestra ser de mucha utilidad.

Tarea 6:
Construye el modelo detector de pistas (engranaje 100 : 1). Escribe primeramente un subprograma para buscar la pista. El robot deberá girar 360° en círculo. Si no se encuentra ninguna pista, el robot se desplaza un poco hacia delante y vuelve a buscar. Para la detección de pistas se consultarán los fotosensores. Una vez que el robot haya descubierto la pista, la seguirá. Si la pista se termina o el robot la pierde, por ejemplo a causa de un cambio considerable en la dirección de la pista, la búsqueda se iniciará nuevamente.

Consejo:
Una vez que se encienda la lámpara se deberá esperar alrededor de un segundo antes de consultar los fototransistores, pues de lo contrario el fototransistor "oscuro" detectará una pista no habiendo ninguna. Utilizaremos como pista cinta aislante negra de 20 mm de ancho o dibujaremos una pista del mismo ancho con un rotulador de fieltro sobre un papel blanco.

Solución:

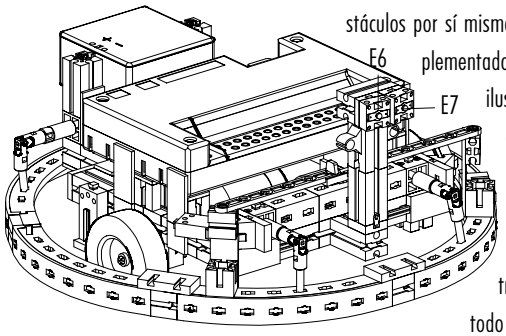


Resultado:

Una vez que nuestro robot haya descubierto la pista, la seguirá infatigablemente. Si hemos hecho una pista en forma de una línea cerrada, el robot se desplazará en ese circuito. Deberemos evitar curvas muy cerradas pues los fotorsensores perdería allí el contacto con la línea. Si bien el robot intentará localizar la pista nuevamente, le será únicamente posible en una zona prácticamente libre de obstáculos.

4.6 La polilla electrónica

El problema de la combinación de diferentes formas de comportamiento tales como búsqueda, persecución y desvío ha aparecido repetidamente hasta ahora. Por esa razón, quisiéramos efectuar otro experimento para verificar la combinación de tales comportamientos. Resumamos los resultados obtenidos hasta ahora. El detector de luz sigue más o menos ciegamente a una linterna colocada delante de él. El robot supone tácitamente que donde hubo una linterna no aparecerán obstáculos. Asimismo parte del principio que no alcanzará realmente a la linterna. Estas suposiciones corresponden a la realidad solamente en casos excepcionales. Nos encontramos obligados a conducir al robot alrededor de cada uno de los obstáculos. Necesitaríamos un robot que esquive los obstáculos por sí mismo. Para eso hemos complementado, como se presenta en la ilustración, el modelo "robot con detección de obstáculos" con la propiedad de búsqueda de luz.



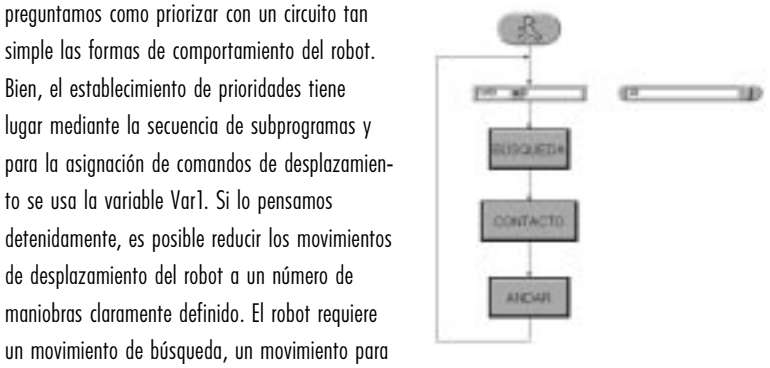
Conectaremos los fototransistores a E6 y E7, todo lo demás lo asumiremos del modelo con detección de obstáculos.

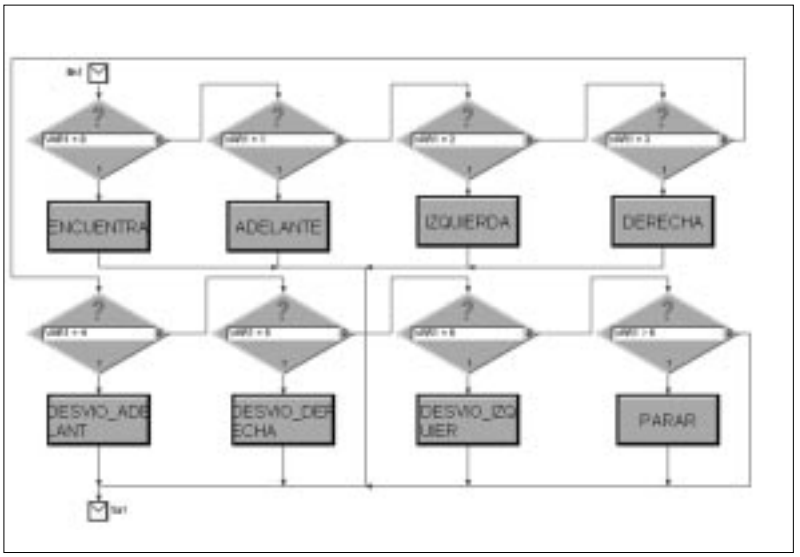
Desde el punto de vista científico, hemos equipado el robot con dos tipos de comportamiento. Puesto que ambos patrones de comportamiento no pueden estar activos simultáneamente, reciben diferentes prioridades. Para nosotros significa esto que normalmente el robot buscará luz. Cuando se detecte un obstáculo, casi un peligro para el robot, se activará el comportamiento para esquivar obstáculos. Una vez salvada la situación, el robot podrá volver a buscar luz.

De esta manera hemos concedido al robot facultades vitales que le permiten navegar autónomamente y evitar peligros. Quien tenga un amigo que también posea un sistema modular de construcción fischertechnik, puede continuar experimentando en esta materia. En cada uno de los robots se montará sencillamente una fuente de luz y los robots se buscarán uno al otro.

Hasta ahora hemos trabajado en la programación principalmente según el principio de "intento y fallo". Tú has empezado con programas simples y te habrás dado cuenta a lo largo de los experimentos como el robot ha hecho lo que tenía que hacer.

Quienes diseñan software profesionalmente no se pueden de ninguna manera echar mano de tales métodos de diseño. En tales casos, se deberá definir una estrategia detallada de diseño antes de escribir la primera línea del programa. Esto no se hace por hacer, sino que se utilizan directivas ya establecidas. Se han establecido procesos que llevan nombres raros como por ejemplo "diseño top down". Con ellos se intenta definir el sistema entero desde arriba hacia abajo sin ocuparse al principio de los detalles. Intentaremos programar nuestra polilla según el diseño top down. Empezaremos para ello con el programa principal, el llamado "main loop". La ilustración muestra el diseño sencillo. El main loop está conformado simplemente por las formas de comportamiento necesarias para el robot, BÚSQUEDA, CONTACTO y ANDAR. La variable Var1 nos parece rara y nos preguntamos como priorizar con un circuito tan simple las formas de comportamiento del robot. Bien, el establecimiento de prioridades tiene lugar mediante la secuencia de subprogramas y para la asignación de comandos de desplazamiento se usa la variable Var1. Si lo pensamos detenidamente, es posible reducir los movimientos de desplazamiento del robot a un número de maniobras claramente definido. El robot requiere un movimiento de búsqueda, un movimiento para esquivar y un movimiento de corrección. Los movimientos para esquivar y de corrección deberán diferenciarse según la dirección, a la derecha o a la izquierda. Con esto se hace evidente que el subprograma BÚSQUEDA calcula un movimiento de n (n = número total de movimientos) movimientos posibles, del mismo modo que el subprograma CONTACTO lo hace. Ya que CONTACTO se ejecuta después de BÚSQUEDA, sobrescribe las instrucciones de BÚSQUEDA. El subprograma ANDAR ejecuta únicamente lo que se le ordena. El proceso de diseño top down parecen sernos de gran utilidad. Pero aún no sabemos como son los subprogramas. Empecemos con el primer subprograma BÚSQUEDA. Este subprograma es responsable de la consulta de los fotosensores. Dependiendo de ambos sensores existen cuatro variantes posibles: sensor izquierdo, sensor derecho, sensor derecho e izquierdo, ningún sensor. Con ello se definen exactamente cuatro maniobras, ENCUENTRA, ADELANTE, IZQUIERDA, DERECHA. Estas maniobras se asignan también a subprogramas; actuaremos rígidamente según los criterios de diseño top down. Ahora sabemos como es el subprograma BÚSQUEDA. BÚSQUEDA entrega un parámetro que es transformado en ANDAR. El subprograma en sí es muy sencillo. El parámetro de salida se configura a través de múltiples ecuaciones. En el programa listo se verificará en el subprograma CONTACTO si obstáculos han estimulado los sensores del robot. Por razones de simplicidad pasamos por alto este programa y no ocupamos





en saber como se activan los movimientos de los motores en ANDAR.
Nos sorprende descubrir que en este subprograma se llaman otros subprogramas. ¿Es esto algo de nunca acabar?
Nos encontramos todavía en la fase de diseño (siguiente fase top down). ANDAR determina cuando y que maniobra ha de iniciarse.
Recién detrás de los diferentes subprogramas ENCUENTRA, ADELANTE,... se halla el código de programa "verdadero". Finalmente hemos llegado a la parte inferior. Ahora se encenderán o apagarán ambos motores. Cada uno de estos subprogramas tiene una tarea definida claramente, la misma que podemos programar.

Resultado:
Programas complejos exigen soluciones completamente novedosas. Es recomendable buscar un principio de solución integral. Elaboraremos un programa según el diseño top down. Las soluciones necesarias se formulan (primeramente) en estructuras formales, por ejemplo BÚSQUEDA o ANDAR. En pasos posteriores se refinarán estas soluciones y se llenarán con el código de programa en sí. Con ello separamos la estructura del programa del código de programa en sí. Podemos observar ambos por separado y analizarlos por separado para buscar fallos, lo cual es muy importante.

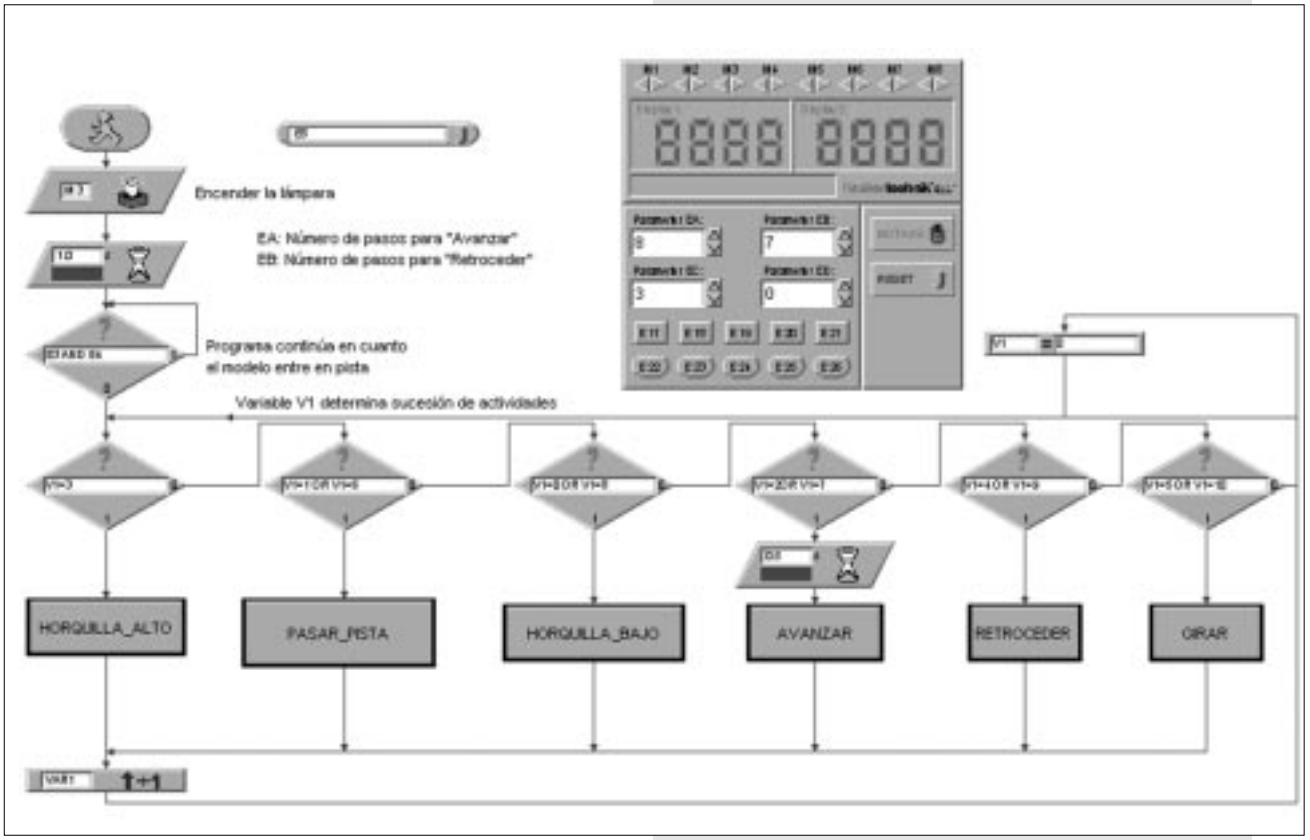
4.7 FTS – Sistema de transporte sin conductor

Abandonemos el área de los experimentos científicos. Las malas lenguas están hablando que son jueguitos. Ingresemos en el área de las aplicaciones prácticas. Nuestra intención no es únicamente construir robots y programarlos, sino también darles una tarea a realizar. Para ello, equiparemos el detector de pistas con una horquilla de transporte elevable. De esta manera nace el robot de carretilla elevadora de horquilla que es capaz de transporta material almacenado sobre una paleta.
Aunque esta prueba parece sencilla, contiene todos los elementos de una aplicación industrial. Tales sistemas de transporte se encuentran ya en uso hoy en día.
Entretanto ya estamos entrenados y disponemos de experiencia en programación, de modo que podemos atrevernos con confianza en nosotros mismos a elaborar programas mucho más complejos.

Tarea 7:
El sistema de transporte sin conductor (FTS) debe desplazarse a lo largo de una pista negra. Al final de la pista recoge una paleta, gira 180° y retorna a la pista. Una vez detectado el final de la pista, el robot deposita la paleta allí, vuelve a girar y trae la siguiente paleta.

Tarea 8:
Como ampliación de la tarea 7, complementaremos el programa de manera que la paleta sea colocada brevemente en los puntos finales de la pista respectivamente. El robot busca la paleta, va al final de la pista, deja la paleta allí, la vuelve a tomar, la lleva al otro extremo, etc., etc.

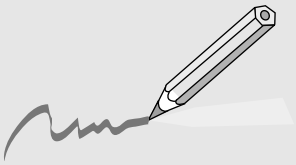
Solución:



Resultado:
Basta con echar un vistazo al programa para darnos cuenta de inmediato del enfoque programático ya conocido. Una variable de estado controla el comportamiento de nuestro sistema de transporte sin conductor.
Sirviéndonos de nuestras refinadas técnicas de programación nos será posible llevar a acabo controles variados. Con el sistema de transporte sin conductor demostramos las viabilidades que ofrece la combinación del sistema modular de construcción fischertechnik con el Intelligent Interface.
Habiendo empezado con un sencillo robot móvil hemos alcanzado un nivel que se diferencia muy poco de la técnica de control y regulación industrial.

5 Un capítulo búsqueda de fallos

Experimentar es muy divertido. Pero siempre y cuando todo funcione. Lo ideal sería que todo funcione en el primer intento. Lamentablemente esto no siempre es así.
Recién cuando un modelo no funciona, se hace evidente si hemos entendido correctamente el mecanismo y encontramos el fallo rápidamente.
En caso de los fallos mecánicos, a menudo es posible ver algo (ensamblaje defectuoso) o escuchar algo (mal funcionamiento). Si a ello se suman problemas eléctricos, la cosa se complica.
Los profesionales buscan los fallos usando diferentes instrumentos de medición, como por ejemplo el voltímetro o el osciloscopio. Tales aparatos no están al alcance de todos. Por esta razón, intentaremos determinar y eliminar el fallo con medios sencillos.
Antes de empezar con nuestros experimentos, debemos construir algunos componentes del sistema modular de construcción fischertechnik.
Principalmente se trata de cables de contacto. Aquí habrá de colocar las clavijas suministradas a cada uno de los cables.
Primeramente se cortarán los cables. Para ello, se medirán las longitudes dadas y se cortarán los segmentos. Antes de cortar, nos cercioraremos que las longitudes sean correctas. Cada cable deberá probarse después de su fabricación. Para ello necesitamos la pila y la lámpara. Si la lámpara se enciende al conectarla a la pila, el cable estará en buenas condiciones. Si la asignación de color es correcta, espiga roja cable rojo, espiga verde cable verde, colocaremos el cable a un lado y probaremos el siguiente.
Si el programa (también el programa suministrado) no funciona con nuestro modelo, iniciaremos el diagnóstico de la interfaz. Este programa auxiliar nos permite probar separadamente las entradas y las salidas. Aquí deberá cada sensor producir la acción correspondiente en la interfaz.
Una vez concluida la prueba sabremos que la parte eléctrica se encuentra en buen estado. Mediante los botones de control de motor encendemos y apagamos los activadores individualmente. De estar todo correcto en este ámbito, buscaremos la causa mecánica.
Un fallo desagradable son los contactos flojos. Puede ser que las espigas no estén firmemente enchufadas en las tomas. De ser este el caso, se ensancharán los muelles de contacto de la espiga con un pequeño destornillador de relojero. Habrá que tener cuidado de no ensanchar demasiado los muelles, pues esto puede ocasionar que los contactos se rompan o que sea difícil enchufarlos.
Otra causa de los contactos flojos son los acoplamientos de tornillo de las espigas. ¡Ajustar cuidadosamente! Habrá también que verificar si los delgados hilos de cobre están rotos.
Si durante en funcionamiento hay interrupciones inexplicables, puede tratarse de una pila casi vacía. La tensión cae brevemente al encender un usuario (encendido del motor) y se produce una reiniciación del ordenador en la interfaz. Tales fallos son difíciles de ubicar, puesto que el programa casi siempre funciona en un primer momento.
Si ocurriesen fallos inexplicables en los programas propios, se deberá ejecutar por razones de seguridad un programa suministrado que se le parezca, a fin de descartar defectos mecánicos o eléctricos.
Si no se tiene éxito con ello, habrá que consulta la asistencia técnica de fischertechnik.



(P) CONTEÚDO

P

1 Para que precisamos de robôs?

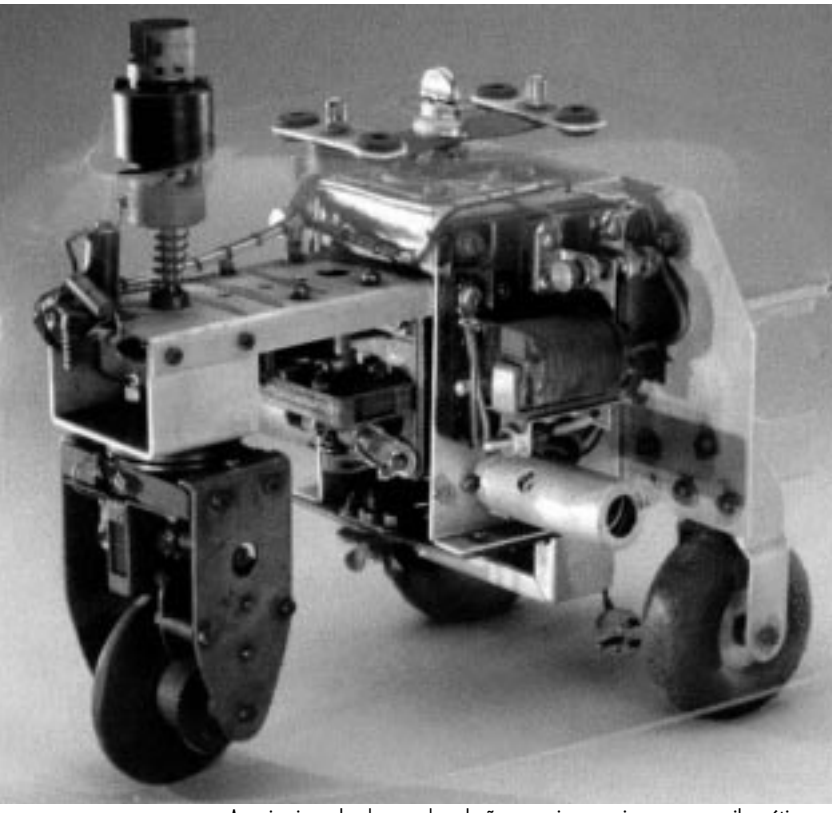
Antes de nos dedicarmos à tecnologia da robótica na prática, vamos procurar responder à pergunta do título, colocada aqui como uma ligeira provocação.

A termo "Robô" foi utilizado pela primeira vez no romance "Golem", publicado em 1923, de Carel Capek. Esta figura artificial sinistra foi criada com suas habilidades para substituir o trabalho humano.

Como acontece freqüentemente na literatura, também esta figura está presa a constrangimentos e causa uma certa desconfiança. Nas décadas de 30 e 40 do século 20 o robô transforma-se cada vez mais numa espécie de autômato. Diversas tentativas de conferir-lhe característica humanas, como por exemplo, uma cabeça com lâmpadas que piscam como olhos e emissão de fala primitiva via alto falante, na perspectiva de hoje nos parecem até ingênuas. Aparentemente, os receios de um potencial domínio dos robôs sobre a humanidade não podem ser dispersos tal facilmente.

Mas nestas primeiras tentativas simples pode-se constatar muito pouco em termos de mobilidade ou até de inteligência nas máquinas construídas. Somente com o surgimento dos circuitos eletrônicos é que a construção de robôs tornou-se realista.

Estreitamente relacionado com a tecnologia da robótica propriamente dita está o problema dos princípios de controle necessários. A questão da "inteligência" do robô continua sendo até hoje um objeto de pesquisas e estudos em muitas empresas, instituições e universidades.



As primeiras abordagens de soluções promissoras vieram com a cibernética. A palavra "cibernética" tem sua origem na palavra grega Kybernetes. Kybernetes era o nome dado ao navegador nas galeras gregas. Ele tinha que determinar a localização do barco e calcular o curso para chegar ao destino. Com isto fica claro, que a cibernética deveria tornar o robô "inteligente".

Mas como poderíamos imaginar um tal comportamento inteligente?

Vamos tentar visualizar isto através de um experimento imaginário. Todos já devem ter observado uma vez o comportamento de uma traça ao redor da luz de uma lâmpada. A traça identifica a fonte luminosa, voa em sua direção e desvia o seu trajeto pouco antes do impacto com a lâmpada. É claro que para se comportar desta forma, a traça tem que ser capaz de detectar a fonte luminosa, estabelecer o trajeto e voar em sua direção. Estas habilidades são baseadas em padrões de comportamento inteligentes e instintivos do inseto.

Agora vamos tentar transferir estas habilidades para um sistema tecnológico. Precisamos identificar uma fonte luminosa (sensores óticos), realizar um movimento (comandar motores) e constituir uma interligação racional entre a identificação e o movimento (o programa).

Nosso experimento imaginário combina um sensor ótico com um motor e uma lógica, de tal forma que o veículo guia o motor sempre em direção à fonte luminosa. Assim o veículo iria se comportar exatamente como a traça, ou não? Uma realização técnica do nosso experimento imaginário acima descrito foi executada nos anos 50 pelo britânico Walter Grey.

Com o uso de sensores simples, motores e circuitos eletrônicos foram criados diversos bichos "cibernéticos", aos quais foram atribuídos comportamentos bem específicos, como por exemplo, o da traça.

Estas máquinas representaram um passo importante no caminho em direção ao moderno robô móvel. Os sensores (fotoresistência, sensores, ...) dos aparelhos comandavam por meio de sua eletrônica os atuadores (motores, relês, lâmpadas, ...) de tal forma, que o resultado obtido foi um comportamento (aparentemente?) inteligente. Na figura pode ser vista uma réplica da tartaruga "cibernética" em exposição no museu Smithsonian na cidade de Washington.

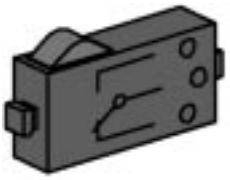
Baseado neste raciocínio, nós elaboramos "padrões de comportamento" para os nossos robôs e tentamos torná-los compreensíveis para o robô através de programas. Mas como podemos, com este raciocínio, responder à pergunta inicial sobre a utilidade de robôs móveis? Para responder esta pergunta de forma concreta, vamos tentar aplicar o comportamento até então mais abstrato da nossa "traça imaginária" a necessidades técnicas. Um exemplo simples é a identificação da luz. Alteramos a fonte luminosa, transformando-a numa faixa luminosa, a linha-guia, aplicada no piso e os sensores não são mais direcionados para frente, e sim para baixo. Por meio de linhas-guia deste tipo é possível um robô móvel se orientar dentro de um galpão de almoxarifado. Informações adicionais, como por exemplo, sob a forma de códigos de barras em pontos estratégicos da linha induzem o robô a executar outras ações, como, por exemplo, o carregamento ou a colocação de uma palete. Estes sistemas de robô já existem de fato. Por exemplo: em hospitais existem longos trajetos de transporte para materiais de consumo, como por exemplo, roupa de cama. O transporte destes materiais por pessoal auxiliar é dispendioso e em parte implica em grande esforço físico. Alem disto, a execução destas tarefas por pessoal auxiliar reduz significativamente o seu tempo disponível para o tratamento dos pacientes. Portanto, podemos perceber que robôs móveis podem ocupar uma posição importante numa sociedade moderna. Mas o que isso tem a ver com os kits da fischertechnik? Para a construção de um robô, além de sensores e atuadores precisamos ainda de muitas peças mecânicas. O kit "Mobile Robots II" da fischertechnik fornece uma base ideal para isto. Podemos combinar peças mecânicas numa variedade de opções quase inesgotável e obtermos veículos-robôs bem robustos. Com a utilização da respectiva "Intelligent Interface" (artigo n. 30402, acessório à parte) temos à disposição uma capacidade de processamento suficiente para desenvolver

programas sofisti cadas. Através desta interface é feita a interligação e a interpretação de inúmeros sensores e atuadores.
Os sensores transformam grandezas de medição físicas, como a intensidade da luz ou a temperatura, em valores que podem ser registrados eletricamente. Neste processo existem tanto grandezas de medição analógicas e digitais. Como grandezas digitais entendemos aquelas que podem ou ser logicamente verdadeiras ou logicamente falsas. Estas condições serão identificadas com 0 ou 1 respectivamente. Um comutador representa um bom exemplo para um sensor digital.
Entretanto, muitas grandezas de medição variam de modo contínuo entre seus dois valores extremos e são chamadas de grandezas analógicas. Elas não podem ser simplesmente representadas por 0 ou 1. Para que estas grandezas possam ser processadas por um computador é preciso que sejam transformadas em valores numéricos. Para esta finalidade a interface da fischertechnik dispõe de duas entradas analógicas EX e EY. O valor da resistência conectada nestes terminais é gravado como um valor numérico. Os valores de medição de um sensor térmico, como por exemplo, 0 ... 5 kΩ, serão, portanto, registrados numa faixa entre 0 ... 1024 e estarão disponíveis para um posterior processamento.
A função mais importante da interface consiste na interligação lógica das grandezas de entrada. Para tanto a interface precisa de um programa. O programa determina de que forma a partir dos dados de entrada, os sinais dos sensores, serão gerados os respectivos dados de saída, os sinais de comando para motores, etc.
Para que possamos desenvolver os programas para a interface com a maior eficiência possível, existe uma superfície gráfica de programação. Por trás da denominação “superfície programável” esconde-se um software, o qual permite a elaboração de programas num nível muito elevado. Assim, por meio de símbolos gráficos podemos desenvolver programas ou algoritmos. O computador da "intelligent interface" está capacitado para executar somente comandos de sua assim chamada lista de comandos da máquina. Estes são basicamente simples estruturas de controle lógicas ou aritméticas, cujas aplicações são extremamente difíceis para iniciantes. Neste sentido o software PC LLWin (artigo n. 30407, não acompanha o kit) disponibiliza elementos gráficos, os quais em seguida podem ser traduzidos para uma linguagem executável para a interface.

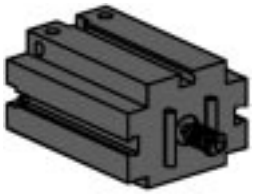
Durante esta nossa fascinante viagem ao mundo dos robôs móveis, vamos avançar passo a passo. Começamos com uma simples montagem de teste para uma verificação das funções básicas da interface e dos sensores. Em seguida continuamos com modelos simples, aos quais são associadas tarefas específicas, e depois vamos tentar desenvolver sistemas cada vez mais complexos. Para que você não fique desanimado com as falhas que surgirem, dedicamos um capítulo inteiro às características e particularidades dos sensores e atuadores, e para os casos muito “difíceis” você pode c onsultar no final o capítulo dedicado à "identificação e eliminação de falhas".
Um ponto muito importante é o cuidado na montagem e na colocação em funcionamento de nossos robôs. Estamos lidando com máquinas complexas, cuja única diferença para os sistemas profissionais de robótica está no seu tamanho relativamente pequeno. Na montagem dos componentes elétricos vamos obedecer estritamente às instruções e verificar duas ou três vezes, se está tudo correto. Nas estruturas mecânicas, também nos projetos de criação própria, vamos prestar muita atenção quanto à mobilidade e folgas reduzidas nas engrenagens e fixações. Nunca será preciso exercer “força bruta” para a montagem. Dependendo da nossa criatividade podemos escrever novos programas e definir assim novos “comportamentos”, cuja complexidade será limitada exclusivamente pela disponibilidade de recursos e capacidade de processamento de dados. Os exemplos a seguir são sugestões neste sentido.

2 Primeiros passos

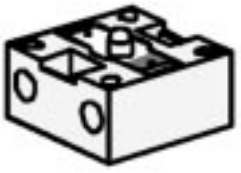
Depois das reflexões teóricas agora vamos finalmente começar a executar nossos próprios experimentos. Com certeza alguém gostaria de começar imediatamente, talvez até com a complicada empilhadeira automática. Claro que isto é possível e seguindo atentamente o manual de instruções, o modelo pode ser montado na primeira tentativa.
Porém, o que faremos se não funcionar? Nestes casos é preciso procurar sistematicamente a causa da falha. E conseqüentemente surgem questões quanto ao modo de funcionamento e às propriedades dos componentes empregados. Evidentemente, um certo conhecimento básico sobre sensores e atuadores é indispensável. Antes de começarmos a nos familiarizar com estes elementos, vamos verificar a interação entre a interface e o computador.
Conforme as instruções do manual LLWin, instalamos o software de controle no computador.



Com o auxilio do diagnóstico de interface vamos testar os diversos sensores e atuadores. Agora podemos, por exemplo, conectar um sensor de contato com dois cabos na entrada E1 e verificar qual condição de comutação lógica é reconhecida pela interface. Um acionamento do sensor de contato deve resultar em uma alteração da condição na respectiva entrada.



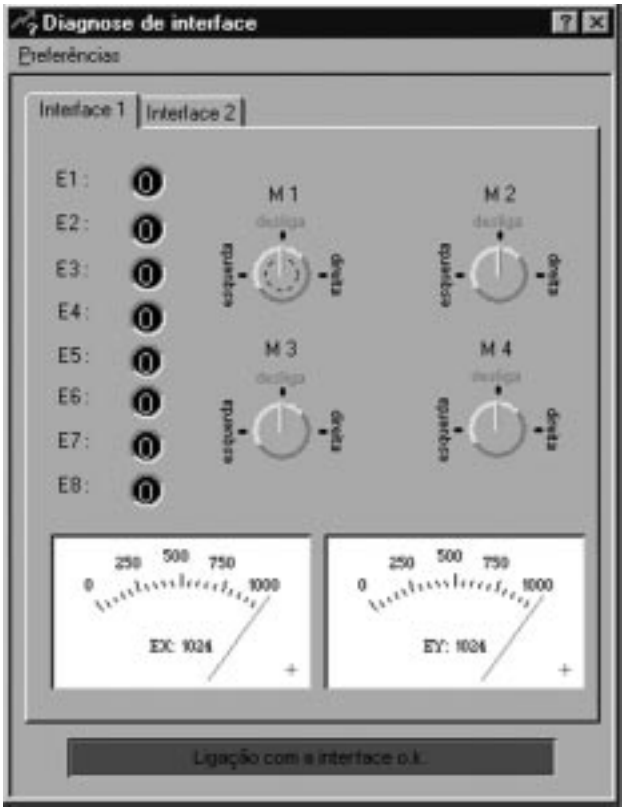
Controlamos as saídas conectando um motor numa saída de motor, por exemplo, M1. Com o mouse conseguimos colocar o motor em rotação. Se queremos testar também a entrada analógica, podemos usar um fototransistor como sensor analógico. Enquanto no motor e no sensor a polaridade das ligações não tem importância (na pior das hipóteses o motor gira no sentido contrário), a polaridade certa na ligação do fototransistor é obrigatória para que funcione corretamente. O contato do transistor possui uma marca vermelha, o qual vamos ligar com um plugue vermelho, o contato não marcado é ligado com um plugue verde.



O segundo plugue vermelho nós colocamos na tomada da entrada EX localizada mais perto da borda da interface, o segundo plugue verde conectamos com a tomada localizada mais internamente da entrada EX.

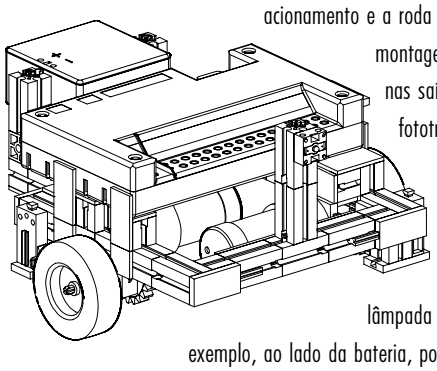
Agora podemos, com o auxílio de uma lanterna, variar a intensidade da luminosidade sobre o fototransistor e com isto alterar a indicação do ponteiro.

Caso o ponteiro não saia da indicação máxima, vamos verificar novamente as ligações do fototransistor. Porém, se a posição do ponteiro permanecer em zero mesmo com a lanterna desligada, é possível que a luminosidade do ambiente está clara demais. Então a posição do ponteiro se altera quando cobrimos o fototransistor.



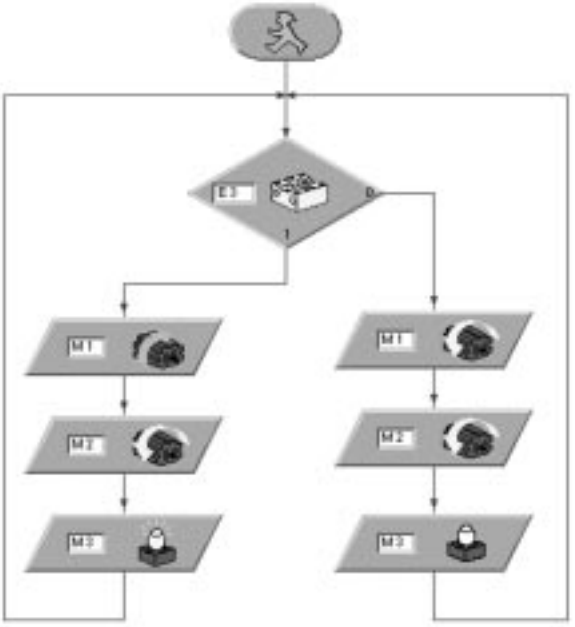
Voltamos mais uma vez rapidamente à atribuição das cores dos plugues: durante a montagem devemos prestar estrita atenção para conectar sempre o plugue vermelho com cabo vermelho e o plugue verde com o cabo verde. Quando em um esquema de circuito são utilizados sinais polarizados, utilizamos sempre o cabo vermelho para o pólo positivo e o cabo verde para o pólo negativo. Isto pode parecer um pouco chato (e para a corrente elétrica tanto faz qual é a cor do cabo), mas para uma identificação sistemática de falhas a correta atribuição das cores proporciona uma enorme facilidade.

Agora vamos encerrar os nossos primeiros passos na área da robótica com um programa simples. Montamos o modelo básico com os dois motores de acionamento e a roda de apoio, conforme a instrução de montagem. Agora ainda vamos conectar os motores nas saídas M1 e M2. Além disso, procuramos o fototransistor e ligamos o mesmo na entrada E3 (atenção quanto à polaridade). Antes ainda fixamos o fototransistor no modelo básico, de modo que ele fique "olhando para frente". Quem quiser pode ligar a lâmpada lenticular na saída M3 e fixar a mesma, por exemplo, ao lado da bateria, porém isto não é estritamente necessário.



Abrimos o programa LLWin e criamos um novo projeto (PROJETO – NOVO). O LLWin oferece diversos modelos, escolhemos "projeto vazio" e damos um nome, como por exemplo, "Passo 1".

Ao pressionar a tecla [OK], aparece uma folha de trabalho vazia, com uma figura de semáforo de pedestre e a janela de módulos. A figura do semáforo de pedestre verde simboliza o início do programa.



Todos os nossos programas partem deste ponto inicial. As diversas partes do programa são trazidos com o mouse da janela de módulos. Os ícones lá disponíveis representam as características de entrada e de saída na interface. Com a tecla esquerda do mouse podemos posicionar o ícone desejado e com a tecla direita do mouse alteramos as propriedades. O ícone de tecla representa uma entrada. Para o nosso programa posicionamos o ícone de tecla embaixo do ícone Iniciar. Quando soltamos a tecla sobre o ícone, abre-se uma caixa de opções. Escolhemos o fototransistor. Se posteriormente quisermos fazer alterações, podemos ativar esta caixa de diálogo novamente com a tecla direita do mouse. Para os motores atribuímos as saídas correspondentes e indicamos o sentido de rotação desejado. Queremos que o sentido de rotação dos motores seja igual na ausência de luz sobre o fototransistor, e em sentido contrário quando uma luz for identificada. Então ligamos os elementos com a função de desenho. A lâmpada ligada em M3 sinaliza a condição do fototransistor. A figura mostra a ligação exata das ramificações do programa. Quem não tem certeza, se tudo está correto, pode comparar o seu programa com o programa Step.mdl. Para isso rimeiro salvamos o nosso próprio programa e carregamos o arquivo Step1.mdl do CD-ROM que faz parte do kit. Quando tudo está correto, carregamos o programa via download na interface e inicializamos o mesmo logo em seguida (RUN – DOWNLOAD). O nosso primeiro robô agora está girando no mesmo lugar. Ele vai fazer isto até que nós o atraímos com uma fonte de luz. Quando o fototransistor registra a luz, os motores até então acionados em sentidos apotos de rotação, passam a girar serem acionados no mesmu sentido e o robô movimenta-se em linha reta, em direção à fonte luminosa. Caso que ele for se distanciando da fonte luminosa, então é preciso inverter a polaridade de ambos os motores. Provavelmente o seu trajeto não será exatamente reto, assim é possível que o sensor ótico perca o contato com a fonte de luz depois de algum tempo. Em seqüência o seu movimento de será alterado de marchar em frente para girar e a busca da fonte de luz começa novamente. Para que possamos fazer experimentos com sucesso, preparamos espaço livre suficiente, pois o nosso robô (ainda) não consegue reconhecer obstáculos no seu caminho.

3 Sensores e atuadores

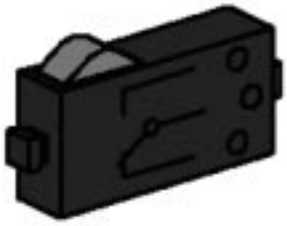
Sabemos agora que o módulo mais importante do conjunto, a interface, se "comunica" com o PC. A questão agora é, como os sinais do ambiente podem ser registrados por nossa interface.

Vamos começar com as entradas. Na linguagem técnica as entradas ou os sinais de entrada são freqüentemente chamados de Input. Para um computador, que é exatamente o caso da Intelligent Interface, só existe uma possibilidade de reconhecer e de processar os sinais elétricos. Portanto, precisamos traduzir os estímulos ambientais para a linguagem do computador. Assim todos os sensores são transformadores, que modificam a "impressão" desejada em um sinal elétrico. Como não queremos seguir "cegamente" as instruções de montagem, é interessante estudarmos as características básicas dos sensores existentes.

Isso é tanto mais importante, quanto à interface mais tarde pode ser complementada com novas aplicações que foram definidas por nós mesmos.

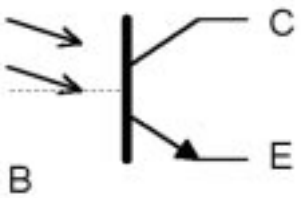
3.1 O interruptor como um sensor digital

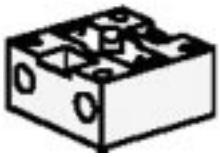
Os simples níveis lógicos "0" e "1" podem ser representadas por meio de um interruptor. No sistema dos kits da fischertechnik são utilizados interruptores mola de precisão com contatos de duas vias.



A característica especial de um interruptor de mola consiste em seu comportamento durante a comutação. Durante um acionamento lento e cuidadoso da tecla vermelha sentimos claramente um ponto de resistência, e ao ultrapassar o mesmo, ouvimos um pequeno ruído de estalo durante a mudança dos contatos. Soltando lentamente a alavanca comutadora, será preciso "ultrapassar" o ponto de ligação inicial, para que possamos reverter novamente a comutação. Esta diferença entre a posição mecânica de liga e desliga é denominada histerese. A histerese de comutação de contatos e outras comutações elétricas é uma característica muito importante. Se ela não existisse, ou seja, se o ponto de ligação fosse igual ao ponto de desligamento, haveria grandes problemas na interpretação de sinais. Minúsculas interferências, como uma leve vibração no momento de comutação, seriam capaz de "simular" para a interface vários acionamentos falsos do contato e conseqüentemente seria impossível realizar uma contagem exata de eventos. O comutador é projetado como um interruptor de duas vias. Assim podemos utilizar ambas as possíveis posições iniciais durante os nossos experimentos, ou seja, fechado na posição de descanso ou aberto na posição de descanso.

3.2 Identificação da luz com o fototransistor





No caso de um fototransistor trata-se de um elemento semicondutor, cujas propriedades elétricas dependem do grau de luminosidade. Um transistor simples é um elemento com três conexões. Estas conexões são identificadas como emissor, base e coletor. Sua tarefa principal é a ampliação de sinais fracos. Uma corrente fraca, que flui de um sinal para a base do transistor, gera uma corrente muito mais forte no coletor do transistor. A ampliação da corrente pode atingir fatores de multiplicação acima de 1000. Entretanto, o fototransistor do kit possui somente duas conexões. Onde ficou a terceira conexão?

Queremos identificar a luz com nosso transistor. Todo mundo conhece as células fotovoltáicas, com as quais obtemos energia elétrica da luz solar. O fototransistor deve ser compreendido como combinação entre uma célula fotovoltaica em miniatura e um transistor. A conexão da base não é conduzida para o exterior (por este motivo está representada com linhas tracejadas na figura). No seu lugar, os impulsos luminosos captados (fótons) produzem uma minúscula foto-corrente, a qual é ampliada pelo transistor e disponibilizada para interpretação no coletor. Para que isto funcione conforme a nossa descrição, o fototransistor precisa de um circuito externo adicional. Como este está disponível na própria interface, podemos ficar despreocupados quanto a isso. O fototransistor pode ser utilizado tanto um como sensor digital, como também como um sensor analógico. No primeiro caso ele destina-se à identificação de mudanças nítidas entre claro e escuro, por exemplo, uma linha marcada. Também é possível diferenciar a intensidade luminosa. Neste caso, o fototransistor funciona como sensor analógico.

3.3 Emissão de sinais com a lâmpada

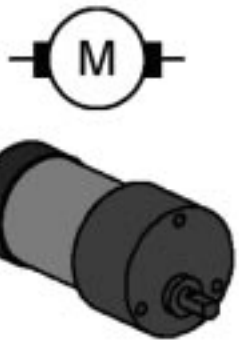
A lâmpada serve para a emissão de sinais de luz simples. E para continuar usando uma linguagem técnica; a lâmpada transforma-se num atuador ótico. A estrutura de uma lâmpada é bem simples. Dentro de um cilindro de vidro e sob vácuo, encontra-se uma espiral de fio de tungstênio fino, estendido entre dois pinos de conexão. Quando passa uma corrente nesta espiral, o fio se aquece até ficar incandescente. Pelo fato de não existir oxigênio no cilindro, o fio não queima, e assim a lâmpada alcança uma longa vida útil. Em conseqüência do grande esforço térmico da espiral incandescente, o fio espiralado se expande cada vez que é a lâmpada é ligada e se retrai ao ser desligada. Estes movimentos mínimos provocam uma fadiga do material, até que algum dia a lâmpada "queima".

Uma possibilidade de aplicação da lâmpada é a indicação de condições de comutação. Por meio da programação de uma lâmpada piscando, podemos também gerar avisos de alerta. Precisamos da lâmpada ainda num outro caso. Juntamente com dois fototransistores forma-se um sensor especial, cujas propriedades permitem a identificação de linhas. A lâmpada funciona como fonte de luz, para que o fototransistor possa identificar uma marcação de cores através das diferentes intensidades da luz refletida. Uma característica especial da lâmpada do kit fischertechnik é a lente ótica contida no cilindro de vidro. Esta permite concentrar o feixe de luz e assim nós podemos, por exemplo, identificar marcações com maior confiabilidade.

3.4 Motores de corrente contínua como fonte de força

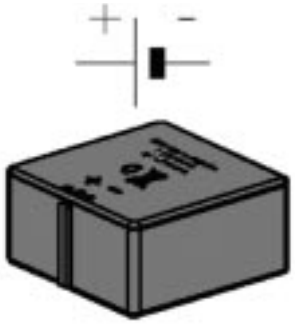
Os motores de corrente contínua são importantes atuadores para os sistemas móveis. No kit "Mobile Robots II" estão disponíveis dois tipos distintos de motores. A sua estrutura mecânica é bastante diferente, porém sua

estrutura elétrica é idêntica. Os motores de corrente contínua são compostos de um "rotor" que gira, e um "estator" fixo. Em princípio devemos entender o rotor como um anel de condutores, o qual se encontra no campo magnético do estator. Quando passa uma corrente elétrica através deste anel de condutores, surge uma força que provoca um desvio do condutor no campo magnético. O rotor começa a se mover. Nas aplicações práticas o anel de condutores simples está configurado como uma bobina (com ou sem núcleo ferroso para o reforço do campo magnético). Muitos motores de corrente contínua geram o campo magnético necessário com o auxílio de imãs permanentes colados no manto metálico do estator. A alimentação de energia do rotor em rotação é feita através de escovas de contato. Estas escovas garantem ao mesmo tempo a inversão da direção da corrente no anel de condutores, a qual é necessária para um movimento contínuo da rotação. O número de giros de motores usuais encontra-se em torno de alguns milhares de rotações por minuto. Uma engrenagem de redução se encarrega de fornecer rotações menores com um torque maior. O kit contém dois tipos de motor distintos, o Minimotor e o Powermotor. O pequeno e compacto Minimotor com rosca sem fim destina-se a acionamentos auxiliares ou aplicações especiais que precisam de pouca força. Ele precisa sempre de uma engrenagem para a redução do número de rotações. O Powermotor disponibiliza torques muito maiores. Ele está equipado com uma engrenagem de flange fixo com uma relação de redução de 50:1. Assim ele atende de forma ideal as necessidades de acionamento do nosso robô. Também existe uma variante deste motor, com uma relação de redução de 8:1 (não está presente neste kit). Neste caso o número de rotações no eixo de saída seria alto demais para o acionamento do robô.



3.5 Alimentação de energia

Os sistemas móveis precisam de uma alimentação de energia autônoma. A partir desta fonte de energia são alimentados todos os consumidores. Os requisitos à alimentação de energia são diferenciados. Enquanto os motores de acionamento se contentam com tensões não estabilizadas, muitos sensores necessitam de tensões estabilizadas para poder fornecer resultados exatos. Por questões econômicas o uso de pilhas ou baterias é a única maneira racional para se alimentar robôs móveis com energia. As células fotovoltaicas ou células de combustível infelizmente ainda não são suficientemente potentes para fornecer resultados que justifiquem o seus custos. As baterias têm preferência em relação às pilhas, pois na maioria das vezes são recarregáveis. A bateria da fischertechnik apresenta uma boa relação entre quantidade de energia e tamanho. A bateria não faz parte do kit e pode ser adquirida junto com um carregador especial sob a denominação de "Accu Set", artigo n.o. 34969. A figura mostra a bateria e o seu respectivo símbolo. Em condições normais o símbolo não indica a polaridade. Por meio de um macete é mais fácil memorizar qual é o pólo positivo: "o traço comprido pode ser cortado e formar a cruz do sinal de positivo".



Na hora da conectar a fonte de energia à interface é muito importante prestar atenção para que a polaridade esteja correta.

3.6 Sensores adicionais

O sistema fischertechnik pode ser facilmente complementado com mais sensores. A maneira mais simples é o uso de sensores de outros kits, como por exemplo, o sensor térmico ou o sensor magnético do kit "Profi Sensoric", artigo n.o 30491. Mas podemos utilizar também outros sensores totalmente diferentes. No comércio especializado são oferecidos os mais variados kits e componentes. Até sensores exóticos, como detectores de gás ou sensores de radar podem ser utilizados. Como não pretendemos destruir o "Intelligent Interface" com tensões de entrada muito altas ou cargas inadequadas, é aconselhável deixar a realização de projetos próprios para os usuários experientes. O caminho mais seguro para a conexão de outros sensores é a separação galvânica entre sensor e interface. Muitos sensores possuem um relê adequado para isso. Os contatos de comutação do relê são ligados como um interruptor comum da fischertechnik e sinalizam agora a ocorrência de novos estímulos ambientais. Uma dica: na Internet são divulgadas muitas opções de expansão por "técnicos fischer" apaixonados.

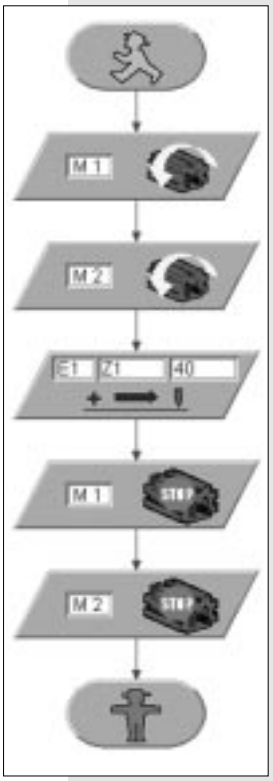
4 Os modelos de robôs

Com as sugestões de montagem a seguir apresentamos algumas variantes de robôs móveis autônomos. Iniciamos com um modelo simples. Baseado neste modelo, você pode conhecer e testar a aplicação de diversos sensores. Aqui é importante considerar tanto as condições internas do robô, como por exemplo, a medição de distâncias através de rodas de impulsos, como também a captação de sinais ambientais, como a identificação da luz ou de rastros. Para cada modelo são colocadas tarefas específicas. Eles devem ser compreendidos como sugestões com o intuito de familiarizar você com a matéria. Os programas LLWin relacionados a cada tarefa encontram-se no CD-ROM que acompanha o kit. Mas você também pode desenvolver suas próprias aplicações para cada modelo. O modelo mais simples é o modelo básico. Neste caso, os motores de acionamento são montados junto com a interface e formam uma unidade compacta. Dois motores fornecem a força de acionamento ao robô. Eles estão posicionados de tal forma, que cada motor aciona uma roda. Uma roda de apoio proporciona estabilidade, para que o robô não possa tombar. Este tipo de arranjo dos motores é chamado de differential drive. Ele proporciona a maior mobilidade com o menor espaço de movimentação necessário. É possível até girar parado no mesmo local. O centro de ambos os motores representa aqui o ponto de rotação em torno do qual o robô se movimenta. Desta forma é possível que ele possa navegar sob as condições mais difíceis, com o processamento de um menor volume de dados. Os motores podem acionar as rodas utilizando duas relações de redução diferentes. (devagar 100: 1 ou rápido 50: 1). Para a variante mais lenta, o acionamento sofre uma redução adicional de 2:1 através de uso de rodas dentadas da fischertechnik. Em todos os modelos é indicada a relação de redução utilizada.

4.1 O modelo básico

Primeiro montamos o modelo básico (relação de redução 100:1) conforme a instrução de montagem. Como este modelo serve como base para muitos experimentos, vamos proceder com muito cuidado na montagem. Quando a parte mecânica estiver toda montada, examinamos a mobilidade dos motores. Para tanto ligamos cada motor por um breve momento diretamente à bateria, sem o uso da interface.

Tarefa 1:
Programe a interface de tal forma, que o modelo movimente-se em linha reta por 40 pulsos. Para a medição dos pulsos utilizamos o contador E1, e como botão de reset a tecla E8.



Modifique depois o programa de tal forma, que o modelo percorre cada vez distâncias diferentes, por exemplo, uma distância de 80 cm. Qual será a grau de precisão da repetição?

Tarefa 2:
Após 80 pulsos de marcha em linha reta o modelo deve girar 180°. Considere os diferentes sentidos de rotação dos motores de acionamento em marcha reta e na rotação.

Solução:
O modelo percorre em média 1 cm do trajeto a cada 1 impulso de contagem. A precisão da repetição está situada na mesma ordem de grandeza, ou seja, 1 cm a cada 80 impulsos. Porém, ela varia em função do piso sobre o qual o robô se movimenta. Extremamente desfavoráveis são os pisos de tapetes grossos ou macios.

Antes de nos dedicar a esta tarefa, precisamos esclarecer duas coisas. Por um lado usamos em nosso programa um novo módulo funcional, o módulo POSIÇÃO. Trata-se aqui de um módulo que permanece ativo até que o número de impulsos programado seja registrado na entrada (neste caso E1). Do ponto de vista do programa isto significa, que nós operamos com uma condição de espera definida. Na primeira experiência usamos esta função como medidor de distância de trajeto para a marcha em linha reta. Quando queremos que o robô gire, trata-se praticamente do mesmo conceito, precisamos somente alterar o sentido de rotação dos motores. Agora precisamos apenas introduzir o número de impulsos, e nosso robô gira sobre o seu próprio eixo.

E agora vem o ponto número dois. Não queremos simplesmente experimentar até que nosso robô gire em 180 , mas sim queremos calcular este valor antecipadamente.

Os motores de acionamento são configurados como differential drive, ou seja, as rodas do robô giram sobre a circunferência de um circulo cujo diâmetro é igual à distância entre as rodas. Para um giro de 180° é preciso que cada roda percorra exatamente a metade desta circunferência.

Calculamos primeiro a circunferência u:

$u = \pi \cdot d = 630$ mm
d = diâmetro (distância entre rodas aprox. 200 mm)

Anteriormente determinamos um trajeto percorrido de aproximadamente 1 cm / impulso, por tanto para 314 mm de distância (meia circunferência) precisamos de 30,5 impulsos. Como não podemos operar com frações, precisamos optar por 30 ou 31 impulsos. Testamos agora qual dos dois valores proporciona a maior precisão.

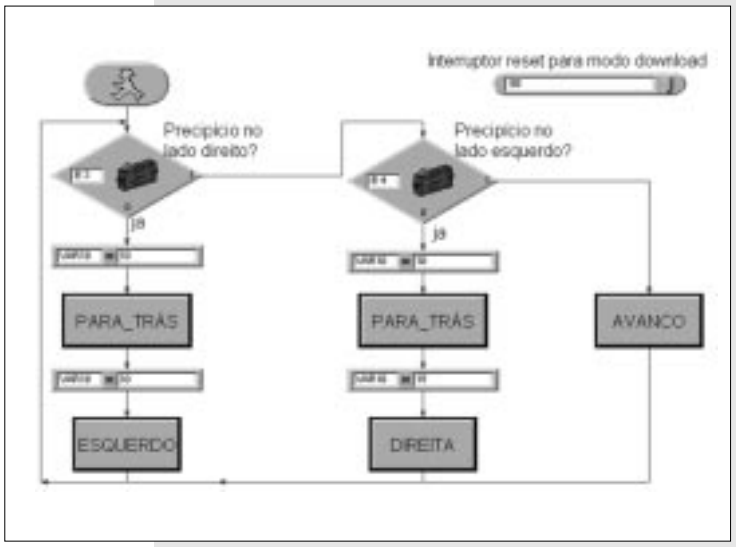
Resultado:
Como resultado das nossas medições com a roda de impulsos, constatamos que a precisão atingida não é muito alta. Principalmente quando são percorridos trajetos diferentes em seqüência ou os mesmos trajetos repetidamente, o erro de medição absoluto se acumula. Da mesma forma é problemático o erro oriundo dos ciclos que não são registrados por completo. As possibilidades de minimizar estes erros são limitadas. Por um lado podemos aumentar a quantidade de impulsos por unidade de distância. A localização do contador diretamente no eixo do motor seria a solução ideal. Além do fato de que não temos acesso a este eixo, surge o problema da taxa de exploração limitada da interface. Quando ocorrem muitos impulsos dentro de uma unidade de tempo, é possível que a interface eventualmente "esqueça" alguns. Assim torna-se ilusório chegar a um cálculo mais preciso do trajeto.
Outros erros, como, por exemplo, o deslizamento das rodas sobre as diversas superfícies ou desvios dos diâmetros das rodas, estão além do nosso alcance. Para nosso consolo resta lembrar que estes problemas em parte também ainda não foram solucionados em sistemas comerciais muito mais caros e complexos.

4.2 Robô com reconhecimento de bordas

Agora que já pesquisamos bastante o nosso modelo básico, queremos ensinar ao nosso robô a ter "medo" de cair no abismo. Até agora o robô se movimentou sobre o tampo da mesa, vigiado atentamente por nossos olhos de águia, para que ele não caia acidentalmente da mesa. Este comportamento realmente não parece ser muito inteligente. Por isso queremos alterar o mesmo.
Para que um robô possa reconhecer uma borda, ele precisa possuir um identificador de bordas. Um procedimento simples e aplicável consiste no uso de duas rodas auxiliares. Estas serão posicionadas como sensores antes do robô, no sentido do deslocamento, e equipados com um interruptor. As rodas são construídas de tal modo que permita movimentos verticais. Ao passar da borda, a roda cai e aciona o sensor.

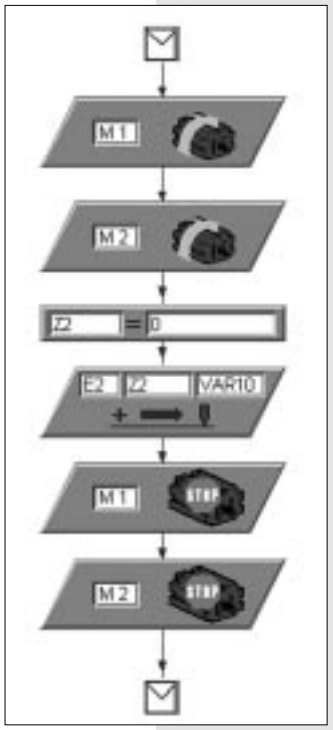
Tarefa 3:
Monte o modelo "Robô com identificação de borda" conforme a instrução de montagem (relação de redução 50:1). O modelo deve marchar em linha reta. Tão logo ele chegue a um abismo no lado esquerdo, ele deve se desviar para a direita, e quando o abismo está do lado direito, ele deve se desviar para a esquerda. Para permitir uma visão melhor, certos movimentos são utilizados como subprogramas. (em frente, à esquerda e à direita). O número de passos é identificado por meio de um sensor de contagem. O número de passos é predeterminado pela variável VAR10. Esta predeterminação é diferente para os subprogramas esquerda e direita.

As diferentes predeterminações têm por objetivo reduzir o risco de o robô ficar preso num canto.



Solução:

Pela tarefa colocada reconhecemos a necessidade de comandar o robô em função dos sensores de bordas. Por isso dividimos a tarefa em partes menores. Primeiro consultamos os sensores (sensores de bordas). Quando nenhum dos sensores está ativado, o robô segue em frente. Na figura isso é representado pelo bloco "Em_frente". Por trás de cada um destes novos módulos esconde-se um subprograma. O uso de subprogramas melhora a visualização de sistemas complexos, além disso, eles podem ser usados repetidamente e contribuem assim para um melhor aproveitamento da capacidade de processamento dos computadores. Nosso primeiro subprograma é muito simples, e apenas inicia o funcionamento dos motores M1 e M2. Porém nós precisamos de mais subprogramas. O robô deve se desviar, dependendo da situação, para a esquerda, para a direita ou para trás. Neste caso não será suficiente apenas acionar os motores. É preciso programar uma certa eqüência de movimentos. Vamos examinar mais um subprograma. Com este subprograma o robô deve recuar ao detectar uma borda. Para tanto ligamos os motores em marcha ré. Em seguida a nossa roda de impulsos deve detectar uma determinada distância. Determinamos o comprimento da distância com a variável VAR10. Uma novidade é o bloco de atribuição Z2 = 0. Após algumas reflexões percebemos o seu sentido. Caso a variável de contagem não esteja valendo zero, o mecanismo funciona somente uma única vez, pois quando Z2 tiver atingido o valor da variável VAR10, o nosso contador de distância iria falhar em cada trajeto subseqüente. Do ponto de vista de programadores profissionais, aqui trata-se de lidar com variáveis locais e globais. A variável local Z2 será zerada no subprograma antes de cada uso.



Resultado:

A chamada de subprogramas melhora a visualização dos programas. Utilizamos variáveis para medir diversas grandezas, aqui se trata de distâncias de trajeto. Precisamos de distâncias diferentes para que o nosso robô também possa se "libertar" dos cantos. Se as distâncias fossem absolutamente idênticas, poderia acontecer que o robô iria marchar para lá e para cá ao chegar num canto, sem conseguir sair.

Ao usar variáveis devemos ficar atentos em relação à sua área de validade. Variáveis locais, ou seja, variáveis usadas somente dentro de um subprograma, nós zeramos antes do seu primeiro uso. Vamos constatar também que o nosso robô precisa de um certo espaço de movimentação para o seu correto funcionamento. O robô não saberá reagir se, durante um movimento de desvio, encontrar novamente uma borda. Os grandes "engenheiros" entre vocês podem tentar achar uma solução para isso.

4.3 Robô com reconhecimento de obstáculos

As bordas nós agora já sabemos reconhecer muito bem. Mas os obstáculos comuns continuam sendo problemáticos. Precisamos modificar o princípio do reconhecimento de bordas. Um contato de colisão vai substituir as rodas auxiliares. Nesta ocasião refletimos também sobre uma deficiência do detector de bordas, pois existem situações perigosas, nas quais o robô despenca "cegamente" no abismo durante a marcha ré e sem sensores voltados para trás. Um terceiro sensor elimina esta deficiência. Neste meio tempo nós já nos tornamos programadores experientes. Por sorte, pois agora a tarefa ficou mais complicada, como nos mostra uma olhada sobre o programa.

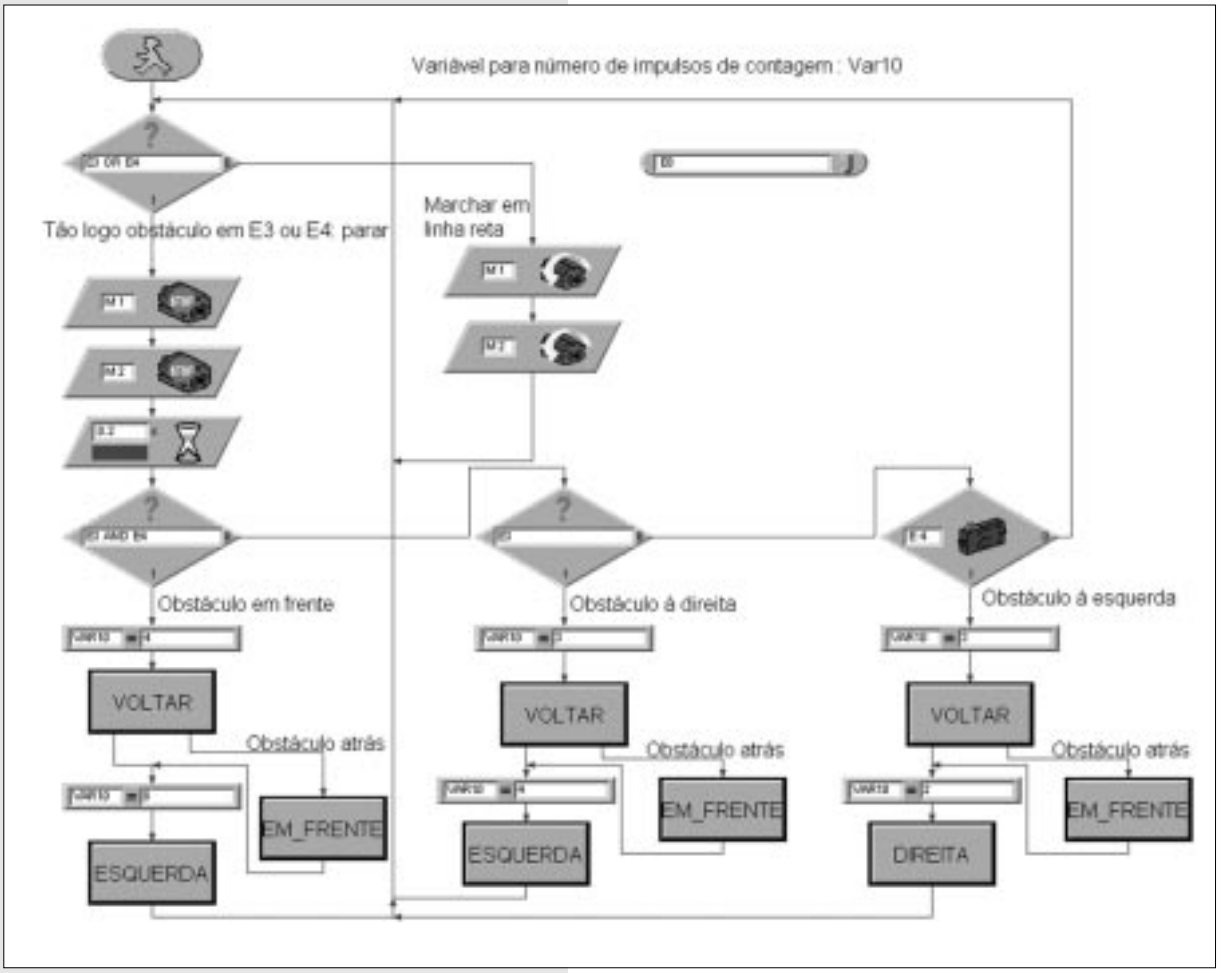
O programa contém agora novos blocos funcionais. Incluímos em determinadas posições do programa os módulos de ESPERA. Isto é facilmente compreensível, pois significa somente a espera de um tempo programado, antes da execução da próxima função do programa. Uma novidade é as comparações lógicas. Até agora sempre realizamos logo uma comparação em cada consulta ao sensor e ramificamos o nosso programa de forma adequada. Uma comparação similar existe também na contagem de impulsos, ou seja, a comparação com um determinado número de impulsos. Outra novidade também é a comparação lógica com várias expressões num módulo COMPARAÇÃO.

Tarefa 4:

O robô será montado com a relação de redução mais veloz 50:1, conforme descrito na instrução de montagem. O modelo deve seguir em linha reta em frente. Ao identificar um obstáculo num dos sensores (E3 ou E4) dianteiros, o robô pára imediatamente. Caso um obstáculo foi detectado no lado direito, o robô recua e desvia pelo lado esquerdo (aproximadamente 30°). No caso de um obstáculo detectado no lado esquerdo, após recuar robô desvia para o lado direito (aproximadamente 45°). O número desigual de graus dos ângulos é necessário para que ele também consiga sair de um canto.

Caso o obstáculo seja detectado diretamente à frente, o robô deve recuar e desviar num ângulo de 90°. Caso durante o seu retorno surja um obstáculo por trás, ele deve avançar um pouco e depois desviar como planejado.

Solução:



Resultado:

A complexidade dos programas vai aumentando. Nós tentamos compensar o aumento da dificuldade através da aplicação de melhores técnicas de programação. E os subprogramas demonstraram ser um ótimo recurso. Como uma nova estrutura de controle reconhecemos as comparações lógicas com subseqüente ramificação do programa. Mas nós reconhecemos também que para as novas propriedades, ou uma melhor realização dos experimentos já conhecidos, é necessário um número cada vez maior de sensores.

4.4 O identificador de luz

Até o momento deixamos o robô reagir mais passivamente aos sinais ambientais. Agora queremos que o robô saia ativamente em busca de sinais. O kit possui dois fototransistores, que usaremos como identificadores de luz. Cada sensor age sobre um dos motores, e com isso será possível realizar um "acompanhamento de feixe de luz".

O programa consiste de duas partes. Uma parte contém a busca por uma fonte de luz e na outra parte é realizado o acompanhamento ou o direcionamento à fonte de luz. É claro que nós vamos utilizar de novo as possibilidades da técnica de subprogramas. Após ligar será ativado o subprograma BUSCA_LUZ. Vamos sair deste subprograma somente quando for encontrada uma fonte de luz.

O programa principal procura guiar o robô em direção à fonte de luz. Sempre que a direção do robô se desviar muito da linha ideal, um dos sensores de luz não receberá mais um sinal da fonte de luz. Em seguida, o respectivo motor será parado por um instante, até que os dois sensores reconheçam de novo a fonte de luz. Disto resulta a atribuição de tarefas.

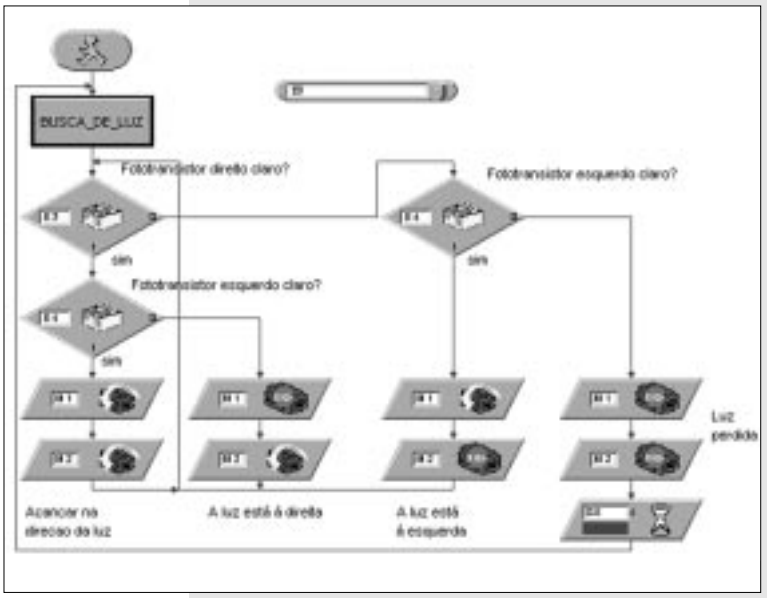
Tarefa 5:

Nós montamos o modelo identificador de luz com a relação de redução mais lenta 100: 1. Programe primeiro a função "busca_luz" (preferencialmente como um subprograma) Com isso o robô gira, no mínimo, por 360° em um sentido. Depois ele gira, no mínimo, por 360° no sentido contrário. Quando durante a busca por luz é encontrada uma fonte de luz, o robô pára. Se após as duas voltas completadas nenhuma luz foi encontrada, a busca por luz será interrompida e o programa encerrado.

Quando a busca por luz é bem sucedida, o modelo deve movimentar-se em direção à fonte de luz. Quando a fonte de luz se movimenta para a esquerda ou para a direita, o robô segue estes movimentos da luz. Quando ele perde o contato com a fonte de luz, começa novamente o programa de busca por luz. Experimente, se você consegue atrair o robô com uma lanterna e dirigir o mesmo através de um percurso de obstáculos.

Dica:
Use uma lanterna como fonte de luz. Não use um foco muito pequeno, para que ambos os fotosensores recebam a luz da fonte luminosa. Considere que em ambientes muito iluminados, a luz da lanterna será superada por outras fontes de luz, por exemplo, a luz do sol vinda de uma grande janela. Possivelmente o robô se movimentará em direção à luz mais clara, ignorando a sua lanterna.

Solução:



Resultado:
Agora nós já construímos um robô que registra ativamente o seu ambiente. Os seus sensores são programados para localizar uma fonte de luz e ir ao seu encontro, ou seguí-la, caso consiga localizá-la.
Nós constatamos que o programa paralisa o robô, quando nenhuma fonte luminosa é detectada. Quando, por exemplo, apenas um pequeno obstáculo interrompe o contato visual direto entre o robô e a fonte de luz, ele não é capaz de identificá-la e ir ao seu encontro, mesmo com a fonte de luz presente. Aparentemente seria conveniente, que após uma busca por luz sem sucesso o robô se desloque de forma mais ou menos aleatória, para um outro local e comece novamente a procurar por luz.
Mas o que acontece quando o obstáculo, que impede a passagem da luz para o robô, fica exatamente no caminho do robô para a fonte luminosa? Esta questão é suficientemente interessante para que seja examinada mais a fundo.

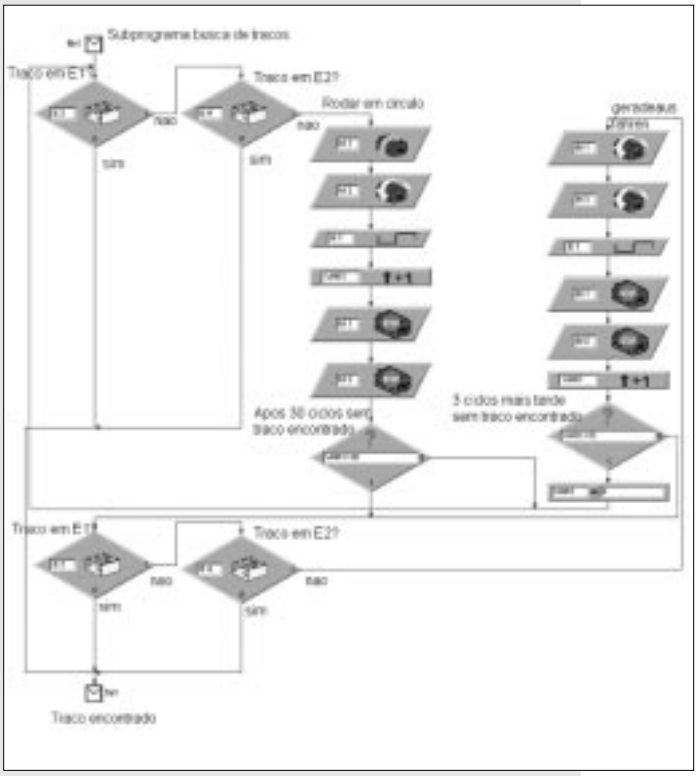
4.5 O rastreador

Procurar e encontrar são características essenciais dos seres inteligentes. Nós já construímos e programamos um robô, que reage diretamente aos sinais dos seus alvos ou "vítimas" em potencial.
Com o rastreador aplicamos um outro princípio de busca. Em vez do movimento direcionado para uma fonte de luz, marcamos uma linha no piso, a qual o robô deve seguir. Com os sensores óticos podemos resolver esta tarefa de forma relativamente fácil. Nós medimos a luminosidade da luz refletida pela marcação e corrigimos os motores de forma adequada. Para que isso funcione com exatidão, iluminamos a linha marcada com nossa lanterna. Tomamos cuidado para que um posicionamento inadequado da lanterna e dos sensores óticos não desoriente os sensores devido à difusão da luz. Nesta situação, a concentração do feixe de luz da lente ótica da nossa lâmpada se mostra muito favorável.

Tarefa 6:
Monte o modelo rastreador (relação de redução 100: 1).
Primeiramente escreva um subprograma, através do qual será procurado o rastro. Para tanto o robô gira em 360° sobre o seu eixo. Não encontrando um rastro, o robô movimenta-se um pouco em linha reta para frente, e procura novamente. Para a identificação do rastro serão consultados os sensores óticos. Quando o robô identificou o rastro, ele segue o mesmo. Quando o rastro termina ou o robô perde o mesmo, por exemplo, devido a um desvio brusco do rastro, ele começa a busca novamente.

Dica:
Após ligar a lâmpada é preciso um pequeno tempo de espera, aproximadamente 1 segundo, até que os fototransistores possam ser consultados, caso contrário o fototransistor identifica "escuro", ou seja, enxerga um rastro onde não existe nenhum. Como rastro usamos uma fita isolante preta com aproximadamente 20 mm de largura ou pintamos um rastro preto com um pincel atômico da mesma largura sobre uma folha branca de papel.

Solução:

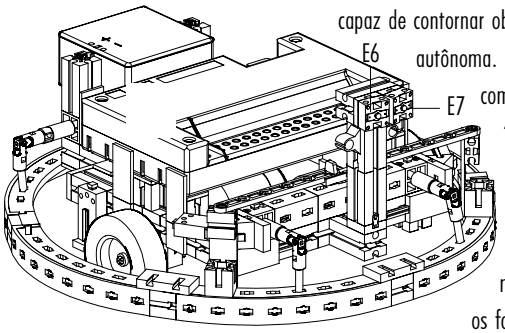


Resultado:

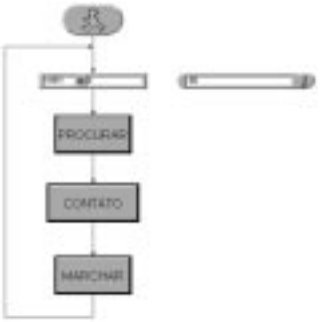
Quando o nosso robô identificou o rastro, ele segue o mesmo sem descanso. Quando desenhamos o rastro na forma de um círculo fechado, o robô percorrerá o mesmo, dando voltas sobre ele. É preciso apenas evitar curvas fechadas demais, para que os sensores óticos não percam o contato com a linha. Embora o robô procure localizar o rastro novamente, devemos confessar que isso será possível somente em áreas praticamente sem obstáculos.

4.6 A traça eletrônica

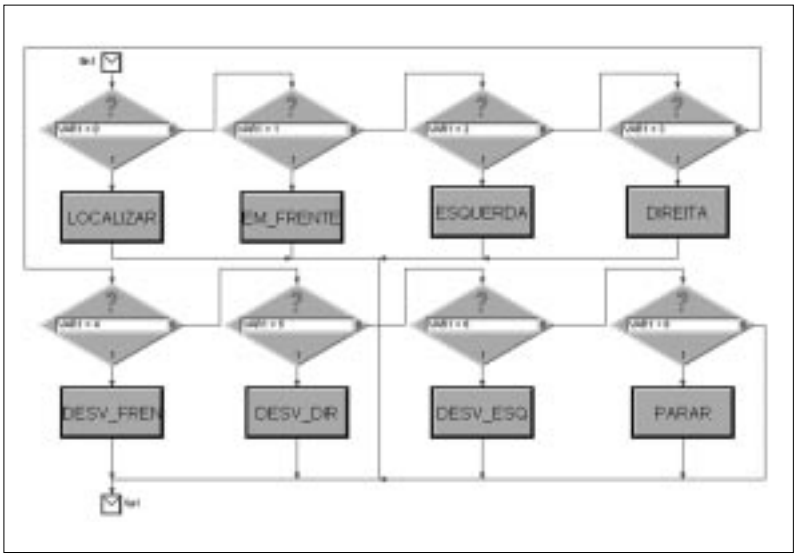
O problema da conjugação de diferentes características de comportamento, como buscar, perseguir e desviar já surgiu diversas vezes. Por isso queremos pesquisar a interação destes comportamentos em mais um experimento. Vamos resumir os nossos resultados obtidos até agora. O identificador de luz simples segue uma lâmpada colocada à sua frente mais ou menos às cegas. O robô considera impossível que no caminho até a lâmpada não possa surgir um obstáculo. Ele considera também que nunca vai realmente alcançar a lâmpada. Entretanto, estas suposições correspondem à realidade somente em casos excepcionais. Somos obrigados a guiar o robô com a lâmpada em volta de cada obstáculo. Na realidade, precisamos de um robô capaz de contornar obstáculos de forma autônoma. Para tanto vamos complementar o modelo "robô com identificação de obstáculos" com a capacidade de identificar a luz, como mostra a figura. Ligamos os fototransistores em E6 e E7, e todo o resto copiamos do modelo de identificação de obstáculos. Do ponto de vista científico, equipamos o nosso robô com duas características de comportamento. Mas como as duas características de comportamento não podem estar ativas simultaneamente, elas receberão prioridades distintas. Para nós isto quer dizer que na situação normal o robô está em busca de luz. Quando um obstáculo é detectado, considerado quase como um perigo para o robô, será ativado o procedimento de desvio de obstáculo. Quando o perigo passou e tudo voltou ao normal, o robô pode novamente procurar por luz. Com isso proporcionamos ao robô características que permitirão uma navegação independente e a capacidade de se desviar de perigos. Quem tem um amigo que também possui um kit fischertechnik, pode estender esta experiência ainda mais. Em cada um dos robôs pode ser montada uma fonte luminosa e então os robôs se procurarão mutuamente. Até agora fizemos a programação geralmente seguindo o princípio de "tentativa e erro". Provavelmente você começou "simplesmente" a programar e, no decorrer dos seus experimentos, percebeu como o robô fez (ou talvez não fez como devia) para cumprir a tarefa a ele conferida. Naturalmente, os programadores de software profissionais jamais podem recorrer a tais métodos de projeto. Nestes casos, já antes que a primeira linha do programa seja escrita, deve ser elaborada uma estratégia de projeto precisa. Isto não é feito de qualquer forma, e sim se utilizando diretrizes bem definidas. Alguns procedimentos foram bem sucedidos e receberam denominações estranhas, como por exemplo, "projeto top-down". Neste conceito procura-se definir todo o projeto do programa de cima (top) para baixo (down), sem entrar em detalhes inicialmente. Nós também vamos tentar programar nossa traça segundo o sistema de projeto top-down. Para isso começamos com o programa principal, o chamado "Main-Loop".



A figura mostra o projeto simples. O main-loop é composto somente pelas características de comportamento necessário ao robô PROCURAR, CONTATO e MARCHAR. A variável Var1 nos parece estranha, como podemos priorizar as características de comportamento do robô com o uso deste loop tão simples? Bom, a priorização ocorre através da seqüência dos subprogramas e para a atribuição das diversas ordens de marcha é usada a variável Var1. Quando pensamos bem, podemos limitar os movimentos necessários do robô a um número claramente definido de manobras. O robô precisa de um movimento de busca, um movimento de desvio e um movimento de correção. Os movimentos de desvio e correção devem ser diferenciados para os lados esquerdo e direito. Com isso fica claro, que o subprograma PROCURAR calcula um de n (n = o número total dos movimentos) movimentos possíveis, como também é o caso do subprograma CONTATO. Como CONTATO é executado depois PROCURAR, ele sobrescreve as instruções de PROCURAR. O subprograma MARCHAR executa somente os comandos que lhe são dados. O procedimento de projeto Top-Down parece ser realmente útil. Mas ainda não sabemos como ficarão os subprogramas. Vamos começar com o primeiro subprograma PROCURAR. Este subprograma é responsável pelas consultas aos fotosensores. Em decorrência dos dois sensores existem quatro variantes possíveis, sensor esquerdo, sensor direito, sensor esquerdo e direito ou nenhum sensor. Com isso são definidas exatamente quatro manobras: LOCALIZAR, EM_FRENTE, A_ESQUERDA, A_DIREITA. A estas manobras também são atribuídos subprogramas; vamos proceder rigorosamente conforme os critérios de projeto Top-Down. Agora está claro, como ficará o subprograma PROCURAR. PROCURAR fornece um parâmetro que será transformado em MARCHAR .



O subprograma propriamente dito é muito simples. Através de diversas comparações é definido o parâmetro inicial. Dentro do programa definitivo verifica-se então no subprograma CONTATO, se foram encontrados obstáculos que acionaram os sensores do robô. Por motivos de simplificação vamos pular este programa e ver como são ativados os movimentos dos



motores em MARCHAR. Constatamos com alguma surpresa, que neste subprograma ainda são solicitados outros subprogramas. Isso não termina nunca? Nós ainda estamos na fase de projeto (próximo estágio do projeto Top-Down). MARCHAR define quando será iniciada qual manobra de marcha. Somente por trás dos diversos subprogramas LOCALIZAR, EM_FRENTE, ... são encontrados os "verdadeiros" códigos de programação. Chegamos finalmente na base. Agora os motores são ligados ou desligados. Cada um destes subprogramas tem uma tarefa claramente delimitada, que podemos programar com uma visão clara.

Resultado:

Programas complexos requerem abordagens de solução totalmente novas. É conveniente procurar utilizar uma abordagem de solução contínua. Nós desenvolvemos um programa segundo o conceito de projeto Top-Down. As abordagens de solução necessárias são formuladas (inicialmente) em estruturas formais, por exemplo, PROCURAR ou MARCHAR. Somente na seqüência mais adiante vamos refinar estas abordagens e finalmente escrever o código de programação propriamente dito. Com isso separamos a estrutura do programa do próprio código de programação. Podemos analisar ambos separadamente e, o que é muito importante, e também verificar os dois separadamente quanto à presença de falhas.

4.7 FTS – Sistema de transporte sem condutor

Deixamos a área dos experimentos científicos e passamos agora para a área das aplicações práticas, pois muitas vezes as más línguas dizem que estamos apenas brincando. Queremos agora construir e programar um robô, não apenas pelo robô em si, mas sim para executar uma tarefa. Para atender esta finalidade, equipamos o rastreador com um garfo de transporte móvel. Com isso criamos um robô-empilhadeira, o qual será capaz de transportar material acomodado sobre um palete.
O experimento pode parecer muito simples, porém contém todos os componentes de uma aplicação industrial. E sistemas de transporte deste tipo já estão em uso hoje em dia.
Agora que já estamos bem treinados e confiantes em relação à nossa experiência de programação, podemos nos arriscar neste programa bastante complexo.

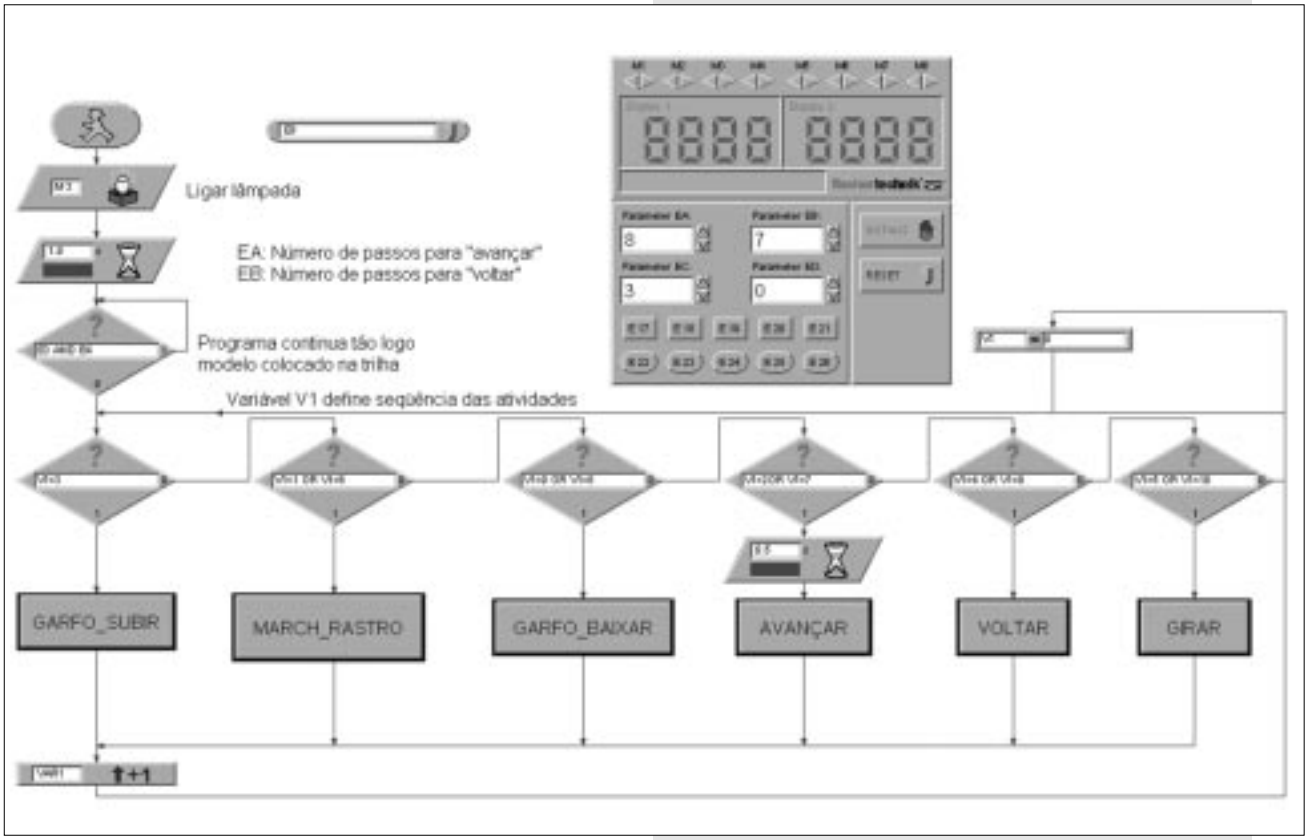
Tarefa 7:

O sistema de transporte sem condutor (FTS) deve andar ao longo de um rastro preto. No final do rastro o sistema carrega um palete, gira em 180° e retorna seguindo novamente o rastro. Quando é detectado o final do rastro, o robô deposita o palete, gira novamente, e vai em busca do próximo palete.

Tarefa 8:

Como ampliação da tarefa 7 completamos o programa de tal forma, que o palete será depositado somente por poucos instantes nos pontos finais do rastro. Por tanto, o robô procura o palete, vai até o outro final do rastro, deposita o palete por uns instantes, carrega-o novamente e vai até o final do rastro do lado oposto, e assim por diante.

Solução:



Resultado:

Quando damos uma olhada no programa, reconhecemos imediatamente a mesma abordagem de programação. Uma variável de condição comanda o comportamento do nosso sistema de transporte sem condutor.

Por meio das técnicas de programação, neste ínterim já bastante refinadas, conseguimos realizar os diferentes comandos. Com o sistema de transporte sem condutor demonstramos as possibilidades oferecidas pela combinação do kit fischertechnik com a Intelligent Interface. Iniciando com simples robôs móveis, alcançamos um grau de conhecimento, que pouco difere da tecnologia de comando e controle industrial.

5 Um capítulo sobre a identificação de falhas

É divertido fazer experimentos. Porém, somente enquanto tudo funciona. Seria ótimo, se tudo sempre funcionasse na primeira tentativa. Mas infelizmente isso não é bem assim. Somente quando um modelo não funciona, é que vamos ver se nós compreendemos exatamente o mecanismo e identificamos o erro imediatamente. No caso de erros mecânicos pelo menos ainda podemos ver (montagem errada) ou sentir (mecanismo emperrado) alguma coisa. Com problemas elétricos já fica mais difícil. Os profissionais utilizam para a procura de falhas uma série de instrumentos de medição, como por exemplo, voltímetros e oscilógrafos. Estes equipamentos não estão ao alcance de todo mundo. Por isso vamos tentar, com meios simples, localizar um erro e corriglo.

Antes de começar nossos experimentos é preciso preparar alguns componentes do kit fischertechnik. Trata-se essencialmente dos cabos de conexão. Os plugues, também fornecidos, serão conectados a seções de cabo. Primeiro dimensionamos o cabo. Para isso medimos os comprimentos previstos e cortamos o cabo de acordo. Antes de cortar verificamos, com cuidado, se os comprimentos estão corretos. Cada cabo será testado depois de pronto. Para tanto precisamos da bateria e da lâmpada. Quando a lâmpada acende depois de ligada à bateria, o cabo está OK. Quando a atribuição de cores também está correta, ou seja: plugue vermelho com cabo vermelho e plugue verde com cabo verde, deixamos este cabo de lado e testamos o próximo.

Quando o programa (inclusive o que acompanha o kit) não trabalha de modo compatível com o nosso modelo, iniciamos o diagnóstico da interface. Este programa auxiliar permite o teste independente das entradas e saídas. É preciso que cada sensor dispare a respectiva ação na interface.

Quando testamos isso com êxito, sabemos que eletricamente está tudo em ordem. Através das botoeiras dos motores ligamos e desligamos os tuadores individualmente. Se aqui também está tudo em ordem, procuramos o motivo da falha na parte mecânica.

Uma falha terrível é os maus contatos. Por um lado, os plugues podem ficar bambos dentro das tomadas. Se este é o caso, podemos abrir um pouco as molas de contato com uma chave de fenda de relojoeiro. Cuidado, uma abertura excessiva da mola pode levar à ruptura dos contatos ou à dificuldade de encaixe do plugue.

Um outro motivo para maus contatos são os prensa-cabos frouxos nos plugues. Por favor, aparafuse cuidadosamente! Nesta ocasião verificamos também se nenhum dos finos fios de cobre está quebrado.

Se ocorrem paradas inexplicáveis durante a operação, o motivo da falha pode ser também uma bateria quase descarregada. A tensão cai por um instante quando uma carga adicional (ligação do motor) é ligada, e com isso é disparado pela interface um reset para o computador. Uma falha deste tipo é difícil de ser detectada, porque o programa funciona inicialmente.

Se ocorrem erros inexplicáveis em programas escritos por nós mesmos, é aconselhável carregar programas similares fornecidos já prontos, para poder excluir falhas elétricas ou mecânicas.

Se tudo isso não resolver, você ainda pode pedir socorro através do serviço de assistência da fischertechnik.

Mobile Robots II

- Begleitheft
- Activity booklet
- Manuel d'accompagnement
- Cuaderno adjunto
- Folheto

fischerwerke
Artur Fischer GmbH & Co. KG
Weinhalde 14–18
D-72178 Waldachtal
Telefon: 0 74 43/12-43 69
Fax: 0 74 43/12-45 91
email: info@fischertechnik.de
http://www.fischertechnik.de

78 897 • ZE • 10/02 • Printed in Germany • Technische Änderungen vorbehalten • Subject to technical modifications

fischertechnik®