Bionic Robots

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

fischertechnik®

(D) BIONIC ROBOTS

Das Begleitheft zum Baukasten.
Für alle, die wissen wollen,
„was dahinter steckt“

(GB+USA) BIONIC ROBOTS

Activity Booklet for the Assembly Kit.
For everyone who wants to know
“what’s behind it.”

(F) BIONIC ROBOTS

Le manuel d’accompagnement du
jeu de construction.
Pour tous ceux qui veulent savoir
« ce qu’il y aderrière »

(NL) BIONIC ROBOTS

Begeleidend boekje bij
de bouwdoos.
Voor iedereen die wil weten
wat„erachter zit“

(E) BIONIC ROBOTS

Cuaderno adjunto a la caja de
costrucción.
Para todos aquellos que quieren
saber «qué hay detrás de las cosas»

(P) BIONIC ROBOTS

O auxiliar do kit Para
Todos osque querem saber
«como a coisa funciona por
dentro»

(D) INHALT

(GB+USA) CONTENTS

(F) SOMMAIRE

(NL) INHOUD

(E) CONTENIDO

(P) CONTÉUDO

D

1. Bionic – Die Natur als Vorbild

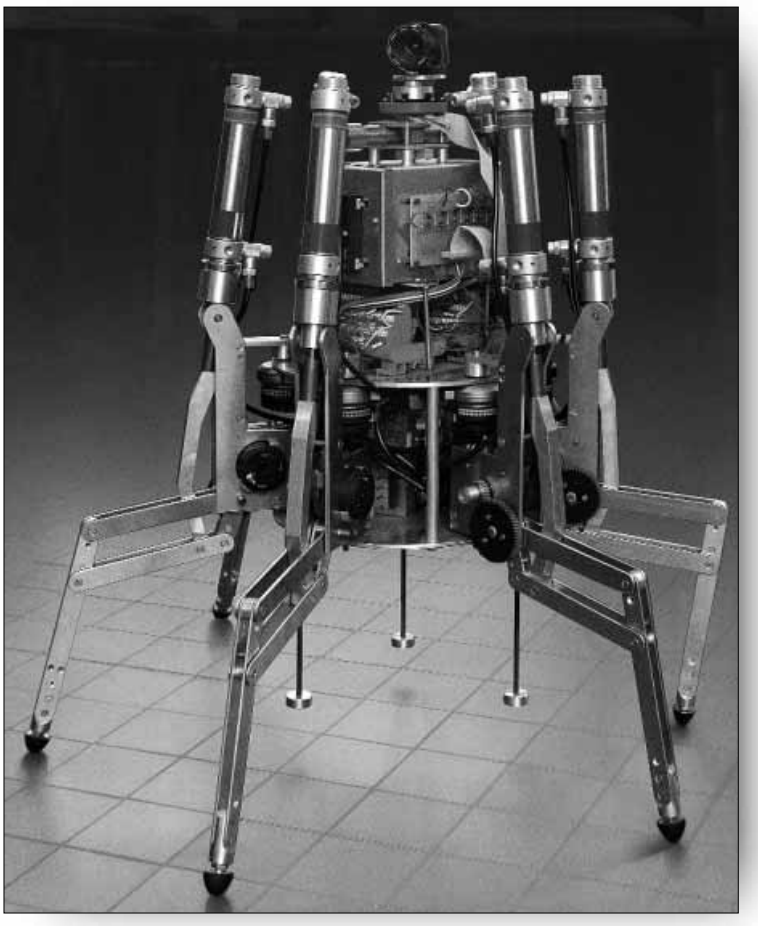
Der Begriff Bionic setzt sich zusammen aus den beiden Begriffen Biologie und Technik. Dieser Zweig der Wissenschaft versucht stets, sich bei technischen Lösungen an der Natur zu orientieren.
So hat der Mensch mit dem Ziel, sich weiter, schneller und effektiver fort zu bewegen als von Natur aus möglich, immer wieder Maschinen erfunden, die dies den jeweiligen Anforderungen entsprechend auf unterschiedliche Weise gewährleisten. Fahrzeuge rollen auf Rädern. In schwierigem Gelände, wo Radfahrzeuge versagen, werden Kettenfahrzeuge eingesetzt. Schiffe schwimmen auf dem Wasser oder sind in der Lage zu tauchen. Bei einigen Fortbewegungsarten dient auch die Natur als Vorbild. So gleicht beispielsweise ein Flugzeug einem segelnden Vogel.

Seit einigen Jahren beschäftigen sich Wissenschaftler mit einer weiteren, in der Natur sehr verbreiteten Bewegungsform, dem Gehen bzw. Laufen.
Es werden Roboter entwickelt, die in der Lage sind, sich auf Beinen fort zu bewegen. Solche Laufmaschinen könnten überall dort eingesetzt werden, wo Rad- und Kettenfahrzeuge kaum mehr eine Chance hätten, so z. B. in äußerst unebenem oder nachgiebigem Gelände, beim Klettern über Hindernisse, Treppen steigen, Überwinden von Gräben oder beim Einsatz an schwer zugänglichen und gefährlichen Stellen in Kernkraftwerken, Bergwerkstollen oder bei Rettungsaktionen.

Die ersten ernsthaften Versuche bei der Entwicklung von Laufmaschinen entstanden 1967 an einer Universität in Tokio. Erstmals orientierte man sich an Stelle von Insekten an der menschlichen Gangart. Die ständige Weiterentwicklung dieser Versuche führte 1985 zu der ersten zweibeinigen Laufmaschine. Inzwischen besitzen diese Roboter über 50 Freiheitsgrade und zahlreiche Mikroprozessoren. Mit Hilfe einer Kamera können Sie z. B. Noten lesen und Orgel spielen.
Man kann sich sogar mit ihnen unterhalten.

Ein Beispiel für einen sechsbeinigen Laufroboter ist der an der Königlichen Militärakademie in Brüssel entwickelte elektropneumatische Laufroboter „Achille“. Ausgestattet mit einer Kamera oben und an den sechs Beinen, soll dieser Roboter mechanisch auf erhöhte oder vertiefte Hindernisse (Gegenstände oder Löcher) reagieren.

Nun hat sich auch fischertechnik diesem spannenden Thema gewidmet und laufende Roboter konstruiert, die dann mit dem Intelligent Interface und der Software LLWin „zum Leben erweckt“ werden.



2. Voraussetzungen und Einstieg

Damit du die Modelle des Computing-Baukasten „Bionic Robots“ bauen kannst, benötigst du zusätzlich zum Baukasten noch folgende Artikel:

Intelligent Interface, Art.-Nr. 30402
Software LLWin (ab Version 3.0), Art.-Nr. 30407
Stromversorgung Accu Set, Art.-Nr. 34969

Wenn du dich mit der Software LLWin und dem Interface noch nicht auskennst, solltest du zunächst das Handbuch der Software LLWin durchlesen. Dort ist beschrieben, wie die Software installiert und das Interface angeschlossen wird. Es ist außerdem bestens dazu geeignet, erste Erfahrungen zu sammeln, wie man fischertechnik-Modelle über den PC steuert.
Mit einigen wenigen Bauteilen aus dem Baukasten (Motor und Taster) kannst du dir zunächst ganz einfache Modellsteuerungen aufbauen.

Sobald du mit der Software und dem Interface vertraut bist, kannst du dich dann durchaus an die anspruchsvolleren Bionic-Robots-Modelle wagen.

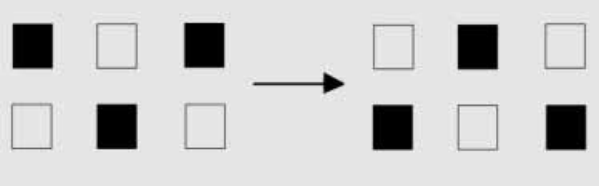
Im Baukasten ist eine CD-ROM enthalten, auf der sich LLWin-Beispielprogramme für die Modelle des Baukastens befinden. Um die Programme öffnen zu können, benötigst du die Software LLWin ab Version 3.0. Du kannst entweder die Beispielprogramme auf der CD lassen und aus LLWin heraus mit dem Befehl DATEI – ÖFFNEN aufrufen oder den kompletten Ordner BIONIC-ROBOTS von der CD in das Projektverzeichnis von LLWin auf die Festplatte kopieren und die Beispiele von dort aus öffnen.

Bevor du die Modelle baust, musst du auch noch einige Teile montieren, z. B. Kabel und Stecker.
Was genau zu tun ist, wird in der Bauanleitung beschrieben.
So, jetzt ist es so weit. Nun kannst du eintauchen in die faszinierende Welt der laufenden fischertechnik-Roboter. Sobald du das erste Modell fertig hast und es auf beinahe gespenstische Weise anfängt sich fort zu bewegen, wirst du begeistert sein von dieser Technik, die in der Natur schon seit Millionen von Jahren für die Fortbewegung benutzt wird.

3. Laufen auf 6 Beinen

3.1 Gangart der Insekten

Die Gangart der Insekten eignet sich hervorragend als Vorbild für den Antrieb von „maschinellen Sechsbeinern". Beim sogenannten Dreifußgang heben immer drei der sechs Beine gleichzeitig vom Boden ab, das vordere und hintere Bein der einen Seite zusammen mit dem mittleren Bein der anderen Seite:

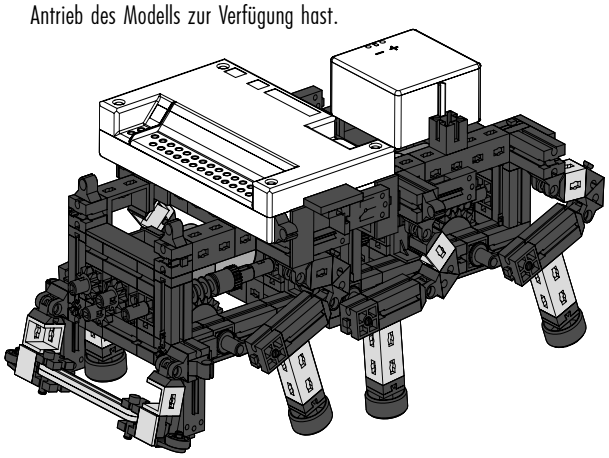


Die Beine, die auf dem Boden stehen (schwarz dargestellt), bilden ein stabiles Dreibein, so dass das Modell immer sicher steht und beim Laufen nicht umkippt.

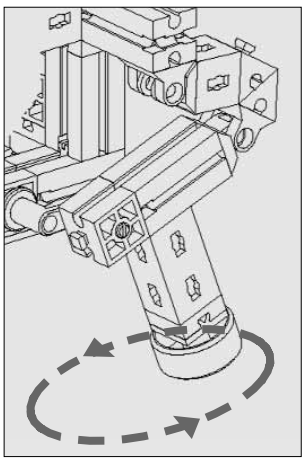
3.2 Modell Mike

3.2.1 Konstruktion des Modells

Baue nun den Sechsbeiner Mike (siehe Bauanleitung S. 4). Lade während des Bauens den Akkupack auf, damit du später genügend Energie zum Antrieb des Modells zur Verfügung hast.



Die Beine des Modells sind so konstruiert, dass sie ein sogenanntes Viergelenkgetriebe ergeben. Die Bauform des hier verwendeten Viergelenks nennt man „Kurbelschwinge". Angetrieben von einer Kurbel führen die beweglich gelagerten Glieder des Getriebes schwingende Bewegungen aus.



Die Abstände zwischen den einzelnen Gelenken und die Lage des Fußpunktes (das ist das untere Ende des Beins), sind so gewählt, dass der Fußpunkt eine elliptische Bewegung beschreibt, wenn sich die Antriebskurbel dreht. Dadurch entsteht eine Bewegung, die einem Schritt beim Laufen ähnelt. Die 6 Kurbeln, die die Beine antreiben, müssen genau so justiert werden, wie in der Bauanleitung gezeigt. Die drei Beine, die gleichzeitig auf dem Boden aufsetzen, haben die gleiche Kurbelstellung. Die Kurbeln der 3 Beine, die zu diesem Zeitpunkt in der Luft stehen, sind dazu um 180° verdreht. Die richtige Stellung der Kurbeln zueinander gewährleistet, dass das Modell in der richtigen Schrittfolge, dem Dreifußgang, laufen kann.

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

Die rechte und linke Seite des Modells werden von je einem Motor angetrieben (das wird für das Kurvenlaufen benötigt). Deshalb muss dafür gesorgt werden, dass sich das mittlere Bein der einen Seite immer in der gleiche Stellung befindet wie die beiden äußeren Beine der anderen Seite. Diese Synchronisation erfolgt softwaregesteuert über die Taster E1 und E2.

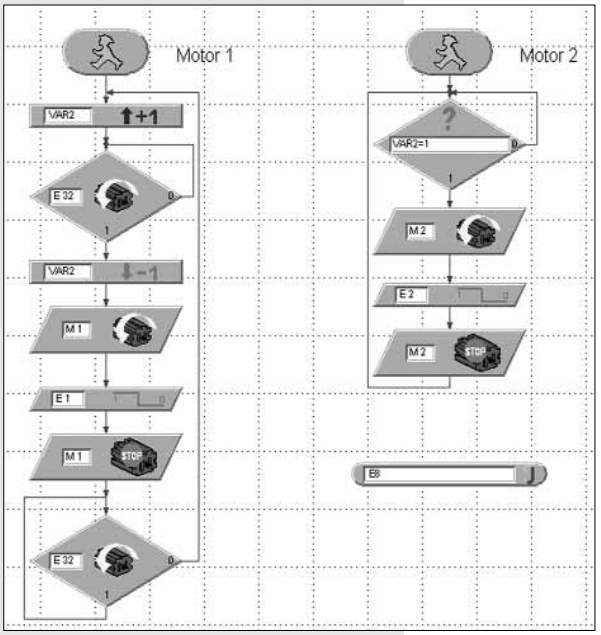
Teste mit der Interfacediagnose, ob alle Taster und Motoren richtig angeschlossen sind. Drehrichtung der Motoren: linke Drehrichtung = vorwärts

3.2.2 Das erste Programm

Nun beginnen wir damit, Mike etwas beizubringen. Zuerst soll das Modell nur geradeaus laufen. Um das Kurvenlaufen und das Reagieren auf Hindernisse kümmern wir uns später.

Aufgabe 1:
Programmiere das Modell so, dass es im Dreifußgang geradeaus läuft. Benutze die Taster E1 und E2 zur Synchronisation der linken und rechten Beine. Beachte dabei, dass immer die beiden äußeren Beine der einen Seite und das mittlere Bein der anderen Seite die gleiche Stellung haben. Verwende außerdem den Taster E8 als Reset-Taster.

Tipps:
Programmiere für jeden Motor einen eigenen Ablauf. Steuere den Ablauf für Motor M2 mit Hilfe einer Variablen VAR2. Wenn du während des Programmierens das Interface nicht benötigst, solltest du die Stromverbindung zwischen Akkupack und Interface unterbrechen um Energie zu sparen.



Lösung:
Das Programm für das Geradeauslaufen sieht wie folgt aus:

Die Variable VAR2 gibt den Impuls, dass Motor M2 startet. Dann wird Motor M1 gestartet. Sobald der Taster E1 betätigt wird, stoppt M1. Sobald E2 betätigt wird, stoppt M2. Der erste Ablauf wartet, bis M2 angehalten hat (Zustand des Motors M2 wird über E32 abgefragt; siehe auch „Abfragen des Motorzustands" im Handbuch LLWin).

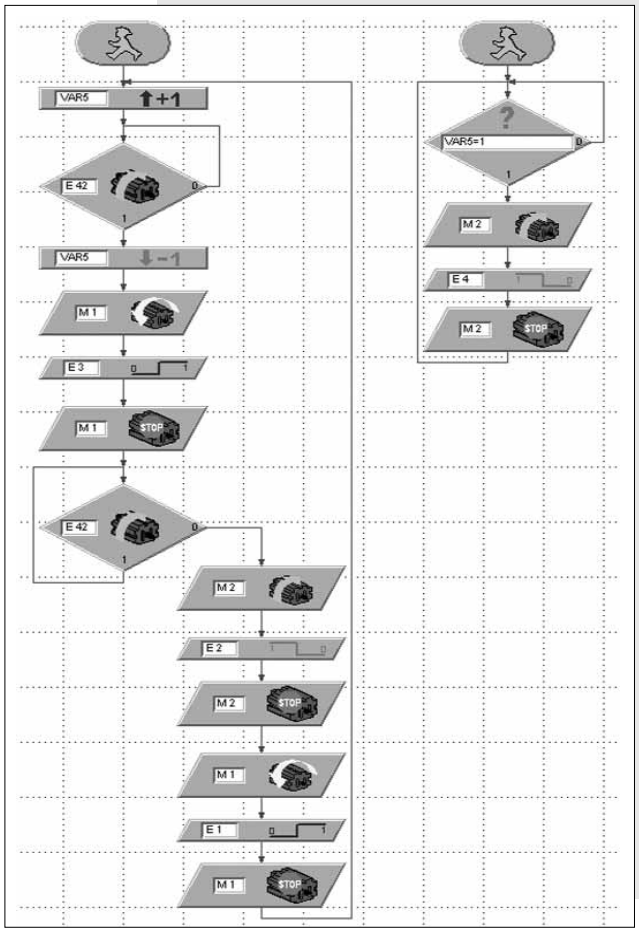
Übrigens, wenn du keine Lust hast diesen Ablauf selbst zu erstellen, findest du ihn als Beipielprojekt MIKE_GERADE.MDL auf der beiliegenden CD.
Starte das Projekt. Wenn du alles richtig programmiert hast, kommt jetzt Leben in das Modell und es läuft geradeaus. Herzlichen Glückwunsch.
Der erste Schritt ist getan.

3.2.3 Die Linksdrehung

Es reicht uns natürlich noch längst nicht, dass Mike nur geradeaus läuft. Als Nächstes wollen wir, dass er sich auf der Stelle dreht.

Aufgabe 2:
Programmiere Mike so, dass er sich nach links dreht.

Tipps:
Das Modell dreht sich nach links, wenn sich M1 nach links und M2 nach rechts dreht.
Du kannst das Modell natürlich unsynchronisiert betreiben. Es dreht sich dann auch, allerdings gibt es dann Stellungen, in denen das Modell nach vorne kippt. Das kann man vermeiden. Und zwar mit folgendem Ablauf:



Mit Hilfe der Taster E1-E4 bewegen sich die linke und die rechte Seite des Modells zuerst einen Schritt gleichzeitig, dann macht die linke Seite einen Schritt, anschließend die rechte usw. So kippt das Modell nie nach vorne. Probiere es aus! Dann fällt es dir auch leichter, diese Reihenfolge nachzuvollziehen.
Auch diesen Ablauf findest du als Projekt MIKE_LINKS.MDL auf der CD.

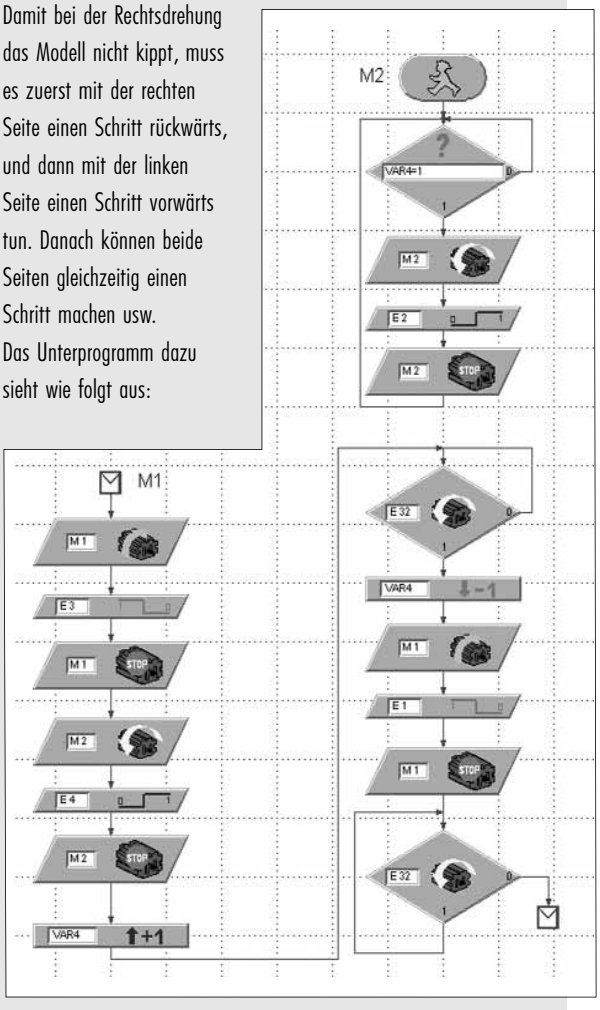
Jetzt kann das Modell geradeaus laufen und sich nach links drehen. Es fehlt noch das Rückwärtslaufen und die Rechtsdrehung. Das Rückwärtslaufen funktioniert im Prinzip wie das Vorwärtslaufen, nur mit umgekehrter Motordrehrichtung. Die Rechtsdrehung funktioniert im Prinzip umgekehrt wie die Linksdrehung.

3.2.4 Links, rechts, vor, zurück

Aufgabe 3:
Programmiere nun jede der Funktionen GERADEAUS, ZURÜCK, LINKS und RECHTS als Unterprogramm, damit du sie später in verschiedenen Projekten flexibel einsetzen kannst.

Tipps:
Wie man einen vorhandenen Ablauf in ein Unterprogramm kopiert, ist im LLWin-Handbuch beschrieben.
Verwende in jedem Unterprogramm eine andere Variable (VAR2-VAR5) um den Ablauf für Motor M2 zu starten.

Damit bei der Rechtsdrehung das Modell nicht kippt, muss es zuerst mit der rechten Seite einen Schritt rückwärts, und dann mit der linken Seite einen Schritt vorwärts tun. Danach können beide Seiten gleichzeitig einen Schritt machen usw.
Das Unterprogramm dazu sieht wie folgt aus:

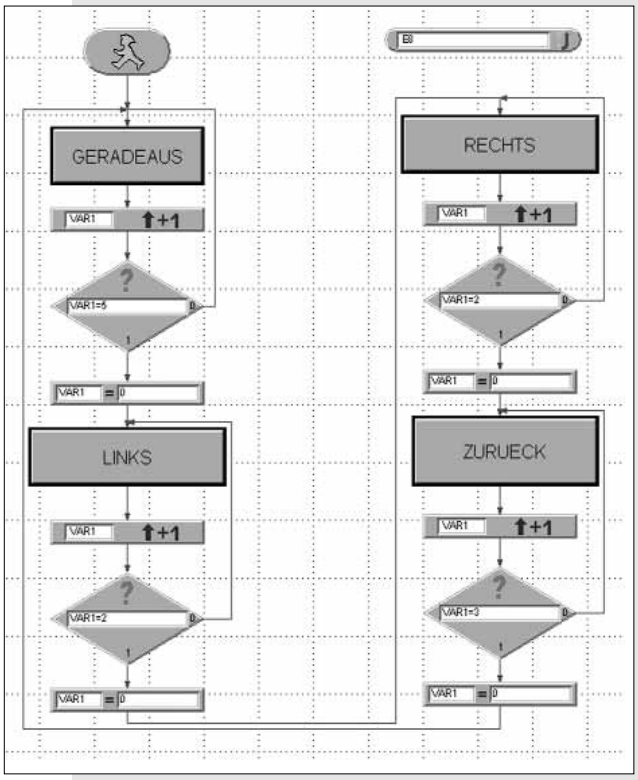


Die anderen Unterprogramme haben wir an dieser Stelle nicht abgedruckt. Falls du beim Programmieren eines Ablaufs Schwierigkeiten hast, findest du die fertigen Unterprogramme in der Datei MIKE_VORLAGE.MDL auf der CD. Das Hauptprogramm dieses Projekts ist leer. Im Bausteinfenster unter dem Reiter „Unterprogramme“ findest du dann die Liste mit den vorhandenen Unterprogrammen, die du im Hauptprogramm einfügen kannst.

Aber schau nicht gleich dort nach wie es geht. Versuche zunächst einmal selbst auf die Lösung zu kommen. Wenn du es nicht schaffst, kannst du immer noch nachsehen.
Um alle Unterprogramme auszuprobieren, wollen wir Mike jetzt tanzen lassen.

Aufgabe 4:
Programmiere Mike so, dass er 5 Schritte nach vorne macht, sich 2 Schritte nach links dreht, dann 2 Schritte nach rechts, anschließend 3 Schritte zurück und dann wieder von vorne beginnt. Verwende als Zählvariable für die Anzahl der Schritte die Variable Var1. Benutze E8 als Reset-Taster.

Lösung:



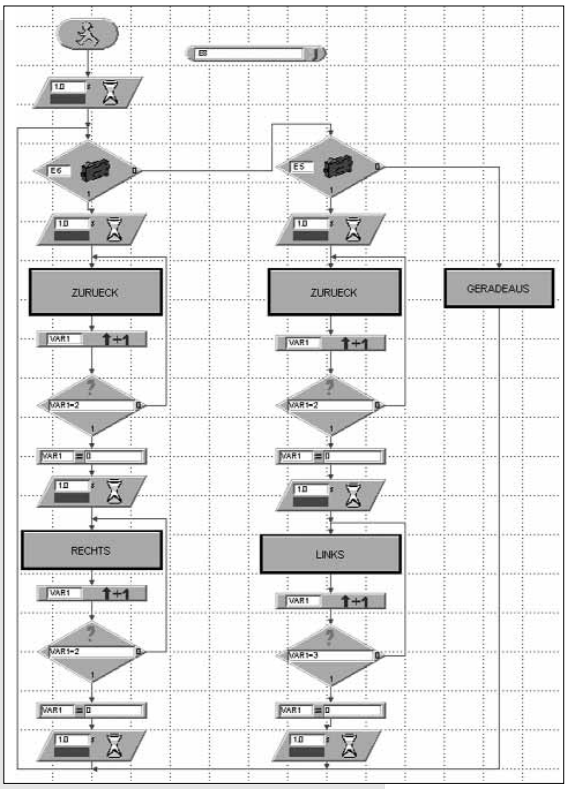
Dieses Projekt heißt MIKE_TANZ.MDL.

3.2.5 Hindernisse erkennen

Zuletzt wollen wir Mike noch dazu bringen, dass er mit seiner beweglichen Stoßstange (oder nennen wir es besser „Fühler“) Hindernisse erkennt und ihnen ausweicht.

Aufgabe 5a:
Programmiere Mike so, dass er bei einem Hindernis an seinem linken Fühler (Taster E6) zuerst 4 Schritte zurück und dann 2 Schritte nach rechts ausweicht. Befindet sich ein Hindernis an seinem rechten Fühler (Taster E5) soll er 4 Schritte zurück und dann 3 Schritte nach links ausweichen.

Lösung:



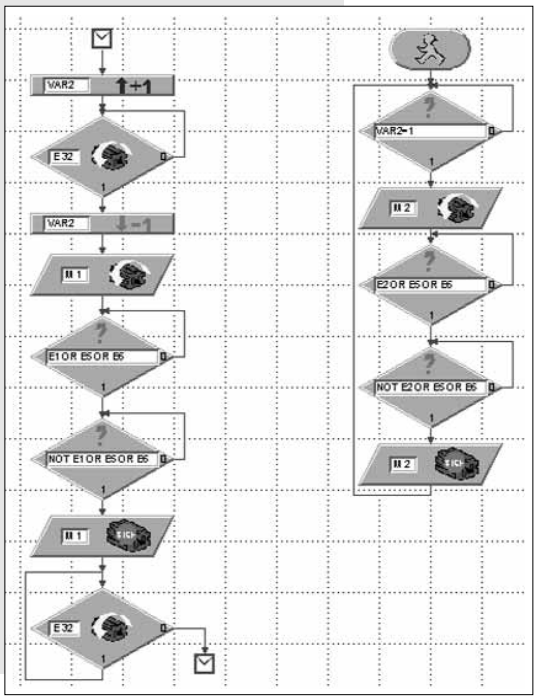
Mike läuft zunächst immer geradeaus. Nach jedem Schritt werden die Taster E5 und E6 abgefragt. Ist E6 gedrückt, verzweigt das Programm in den linken Ablauf (erst zurück, dann rechts). Ist E5 gedrückt, geht es in den mittleren Ablauf (erst zurück, dann links).

Da die Taster E5 und E6 nur nach jedem vollen Schritt abgefragt werden, dauert es relativ lange, bis Mike auf ein Hindernis reagiert.

Aufgabe 5b:
Optimiere das Unterprogramm GERADEAUS so, dass Mike schneller auf ein Hindernis reagieren kann.

Tipp:
Verwende zum Abfragen der Taster E1 und E2 nicht den Baustein FLANKE sondern den Baustein VERGLEICH. Frage damit zusätzlich ab, ob E5 oder E6 gedrückt ist.

Lösung:

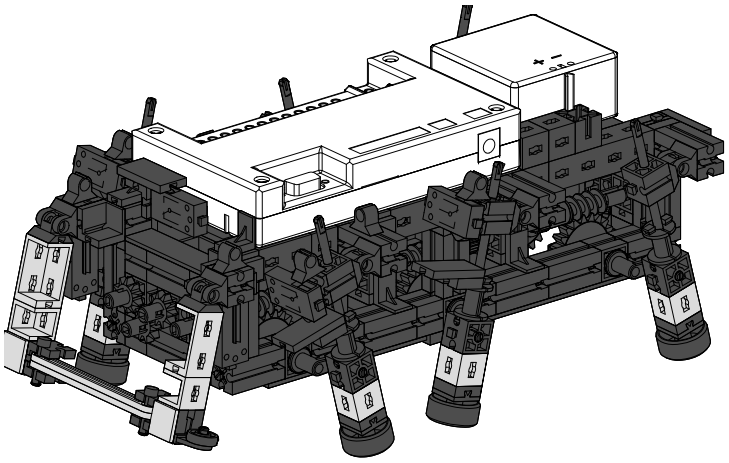


Nun sollte Mike perfekt funktionieren. Dieses Progamm befindet sich ebenfalls auf der CD unter MIKE_HINDERNIS.MDL. Das verbesserte Unterprogramm kannst auch in das Projekt MIKE_VORLAGE.MDL einarbeiten. Werden in einem anderen Programm E5 und E6 nicht abgefragt, stört das überhaupt nicht. Diese verbesserte Vorlage haben wir unter MIKE_VORLAGE_HINDERNIS.MDL abgespeichert.

Nachdem wir nun den ersten Sechsbeiner ausführlich behandelt haben, wenden wir uns dem zweiten Modell zu, das ebenfalls 6 Beine besitzt. Wir nennen es „Jack".

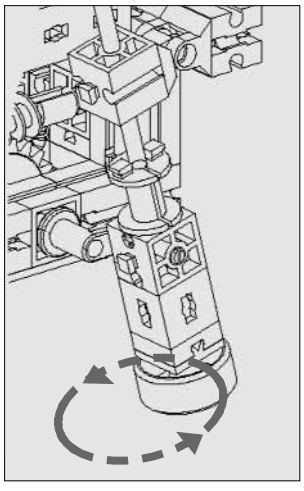
3.3 Modell Jack

Jack gehört ebenfalls zur Gattung der sechsbeinigen fischertechnik-Modelle. Allerdings unterscheidet er sich in der Konstruktion seiner Beine erheblich von Mike.
Baue nun das Modell wie in der Bauanleitung ab S. 12 beschrieben. Übrigens, die Baustufen 1-13 sind bei Mike und Jack identisch. Du brauchst also Mike nicht völlig zu zerlegen, bevor du anfängst Jack zu bauen.



3.3.1 Die Konstruktion

Bei der Beinkonstruktion von Jack handelt es sich ebenfalls um ein sogenanntes Viergelenkgetriebe. Die hier verwendete Bauform nennt man „schwingende Kurbelschleife". Die Schubstange ist in einer beweglichen Längsführung gelagert, die hin und her schwingt, wenn sich die Kurbel dreht. Die Kurve, die der Fußpunkt des Beins beschreibt, ist nicht so ellipsenförmig wie beim Modell Mike, sondern mehr ein Kreis.



Dadurch hebt und senkt sich Jacks Körper während des Laufens stärker als der von Mike. Die Schritte sind kürzer. Dafür kann Jack kleine Hindernisse überwinden, wozu Mike nicht in der Lage ist. Außerdem erinnert diese Getriebebauform eher an ein Bein als dies bei Mike der Fall ist. Wenn Jack läuft, sieht es aus, als ginge er auf Stelzen.
Er bewegt sich ebenfalls im Dreifußgang der Insekten. Auch bei diesem Modell ist es wichtig, die Kurbeln genau wie in der Bauanleitung beschrieben zu justieren und die Zangen- und Nabenmuttern gut fest zu drehen.

3.3.2 Die Programmierung

Es liegt nahe zu denken, dass man für Jack die Programme verwenden könnte, mit denen auch Mike funktioniert. Versuche es!

Aufgabe 1:
Betreibe Jack mit dem Programm MIKE_HINDERNIS.MDL.
Was kannst du beobachten?

Beobachtung:
Vorwärts und rückwärts läuft das Modell einwandfrei. Beim Drehen nach links und rechts jedoch kippt es nach vorne.

Aufgabe 2:
Wie erklärst du dir das?

Lösung:
Die beiden Modelle besitzen unterschiedliche Beinkonstruktionen. Auch die Taster E1-E4 werden an einer anderen Position des Beins betätigt. Die Art und Weise wie Mike sich dreht, muss demnach für Jack noch lange nicht funktionieren – Pech gehabt.

Selbstverständlich gefällt uns das gar nicht und wir wollen so schnell wie möglich eine Lösung finden.

Aufgabe 3:
Versuche Jack so zu programmieren, dass er wie Mike Hindernisse erkennt, aber beim Drehen nicht nach vorne kippt.

Tipps:
Speichere das Projekt MIKE_HINDERNIS.MDL unter dem Namen JACK_HINDERNIS.MDL und nimm dort die notwendigen Änderungen vor.

Wenn Jack sich dreht, können immer beide Motoren gleichzeitig laufen. Das abwechselnde Ein- und Ausschalten entfällt. Entscheidend ist, dass sich die Beine zu Beginn des Drehens, also nach dem Rückwärtslaufen, in der richtigen Ausgangsposition befinden.

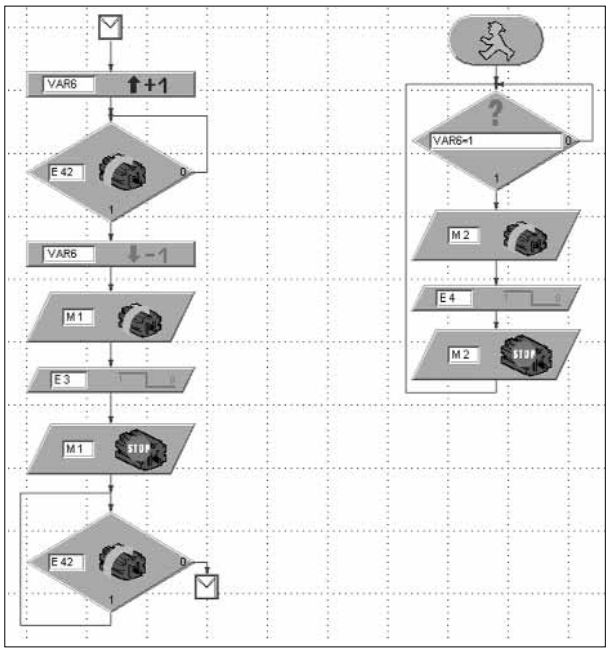
Linksdrehung:
Soll sich das Modell nach links drehen, muss zu Beginn der Drehung die Kurbel des vorderen linken Beines nach hinten und die des vorderen rechten Beines nach vorne zeigen. Dies ist der Fall, wenn beim Rückwärtslaufen auf der linken Seite der Taster E2 und auf der rechten Seite der Taster E1 betätigt und wieder los gelassen wurde.

Rechtsdrehung:
Soll sich das Modell nach rechts drehen, muss zu Beginn der Drehung die Kurbel des vorderen linken Beines nach vorne, die des vorderen rechten Beines nach hinten zeigen. Dies ist der Fall, sobald beim Rückwärtslaufen auf der linken Seite der Taster E4 und auf der rechten Seite der Taster E3 betätigt und wieder los gelassen wurde.

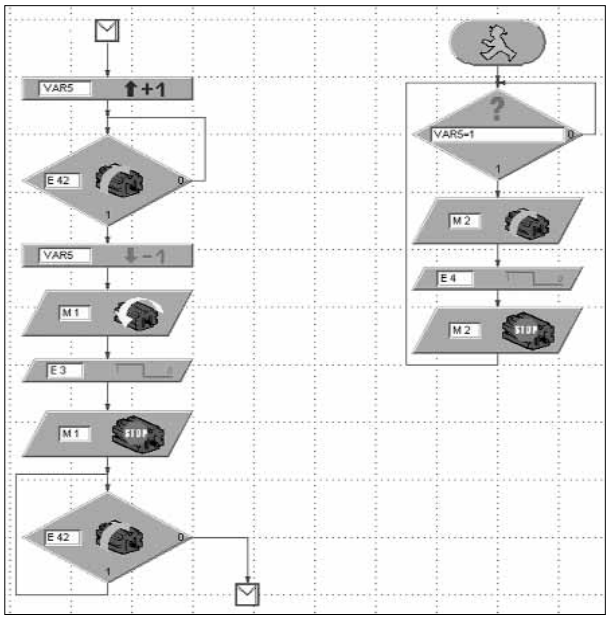
Ziemlich kompliziert, nicht wahr? Aber keine Sorge, wir haben es gleich:

Für die Rückwärtsbewegung müssen zwei unterschiedliche Unterprogramme benutzt werden.
Soll sich das Modell nach links drehen, synchronisierst du die Schritte beim Rückwärtslaufen mit den Tastern E1 und E2. Dies entspricht dem Unterprogramm ZURUECK aus dem Projekt MIKE_HINDERNIS.MDL.
Soll sich das Modell nach rechts drehen, werden die Schritte rückwärts mit E3 und E4 synchronisiert.

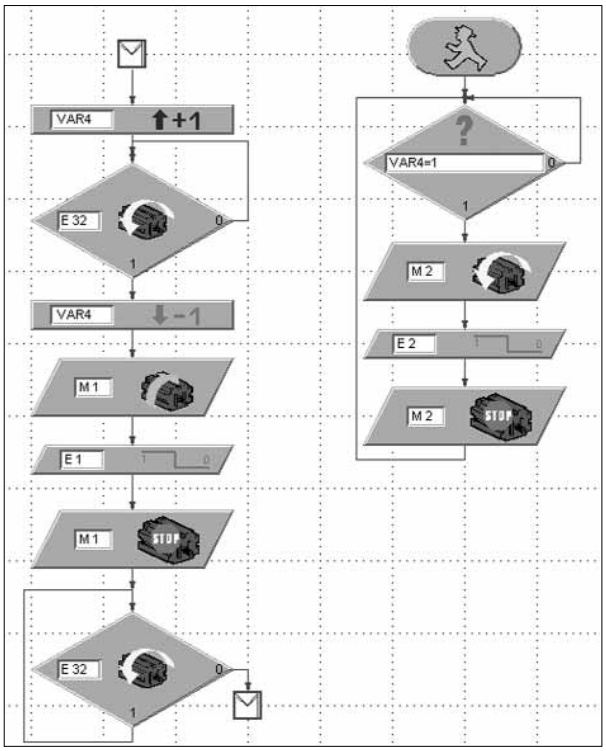
Du benennst also das Unterprogramm ZURUECK mit dem Befehl UNTERPROGRAMM – UMBENENNEN in ZURUECK_L um. Danach kopierst du es mit UNTERPROGRAMM – KOPIEREN in ein zweites Unterprogramm ZURUECK_R. Dort änderst du die Tasterbezeichnungen für die Synchronisierung auf E3 und E4 um. Vergiss auch nicht, für ZURUECK_R eine neue Variable VAR6 für die Synronisierung zu benutzen, sonst ist das Chaos perfekt. ZURUECK_R sieht dann wie folgt aus:



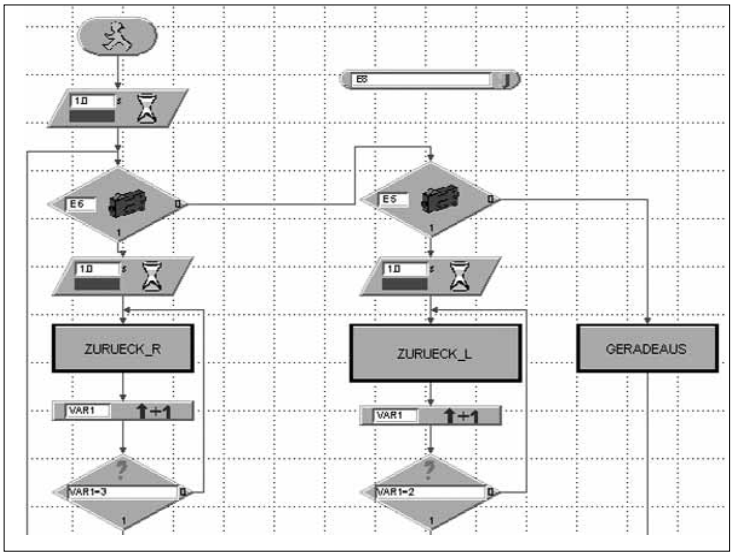
Jetzt müssen noch die Unterprogramme für die Drehung selbst geändert werden, so dass sich die beiden Motoren immer gleichzeitig drehen. Das Unterprogramm LINKS besteht aus folgenden Bausteinen:



Du siehst, gegenüber dem Unterprogram im Projekt MIKE_HINDERNIS.MDL können einige Bausteine entfallen.
Das Unterprogramm für die rechte Drehrichtung sieht ähnlich aus, nur mit anderen Motordrehrichtungen. Außerdem werden die Taster E1 und E2 zur Synchronisierung der beiden Motoren verwendet:



Zuletzt ersetzt du im Hauptprogramm in der Verzweigung für das Ausweichen nach rechts das Unterprogramm ZURUECK_L durch ZURUECK_R:



Der Rest des Hauptprogramms bleibt unverändert.

Geschafft! Wenn du nirgends einen Fehler gemacht hast, müsste Jack jetzt laufen ohne beim Drehen zu kippen. Falls irgendetwas nicht funktioniert und du absolut nicht weißt warum, mach dir nichts draus, das war auch wirklich eine harte Nuss. Du hast auf jeden Fall die Möglichkeit, das fertige Projekt JACK_HINDERNIS.MDL einfach von der CD aufzurufen und das Modell damit zu betreiben.

Falls du es selbst geschafft hast das Problem zu lösen, kannst du stolz sein, denn dann gehörst du ab jetzt zu den Profis unter den Programmierern.

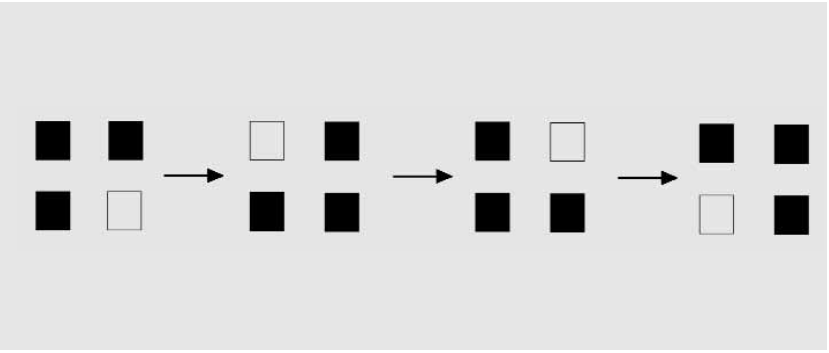
D

4. Laufen auf 4 Beinen

4.1 Gangarten der Säugetiere

Um einen Laufroboter mit 4 Beinen zu konstruieren, nehmen wir wieder die Natur zum Vorbild und schauen uns an, in welchen Gangarten sich Säugetiere fort bewegen.

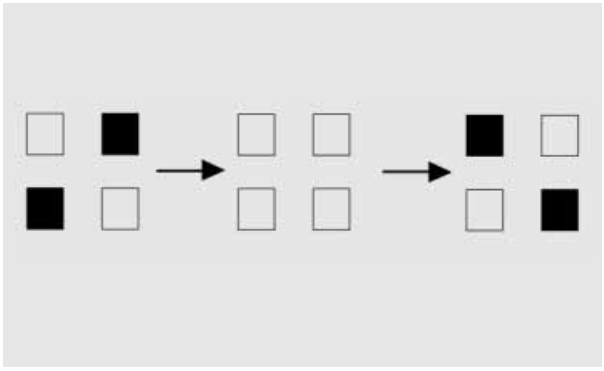
Die langsamste und sicherste Gangart ist der sogenannte Schritt. Ein Bein sucht einen neuen Standort, während sich der Körper des Tieres auf drei Beine stützt. Die Tiere bewegen sich im Kreuzgang mit der Schrittfolge: rechtes Vorderbein, linkes Hinterbein, linkes Vorderbein, rechtes Hinterbein vorwärts.



Die schwarzen Flächen stellen die Beine dar, die auf dem Boden stehen, die weißen Flächen das abgehobene Bein.

Wollen wir diese Gangart für einen Laufroboter verwenden, müssen wir folgendes bedenken:
Stell dir vor, du sägst an einem Tisch mit 4 Beinen ein Bein ab. Was passiert? Richtig, der Tisch kippt um. Die drei Beine bilden also kein stabiles Dreibein mehr, wie das bei den Sechsbeinern der Fall war. Das erschwert die Konstruktion eines vierbeinigen Roboters.

Je schneller sich die Säugetiere fort bewegen, desto instabiler wird ihre Gangart. Sehen wir uns noch kurz die Gangart „Trab" an. Im Trab werden die Beine in der Diagonalen synchron abgehoben. Bevor sie aber den Boden berühren springen die beiden anderen Beine bereits ab. Dies bedeutet, dass zeitweise der Bodenkontakt ganz verloren geht.

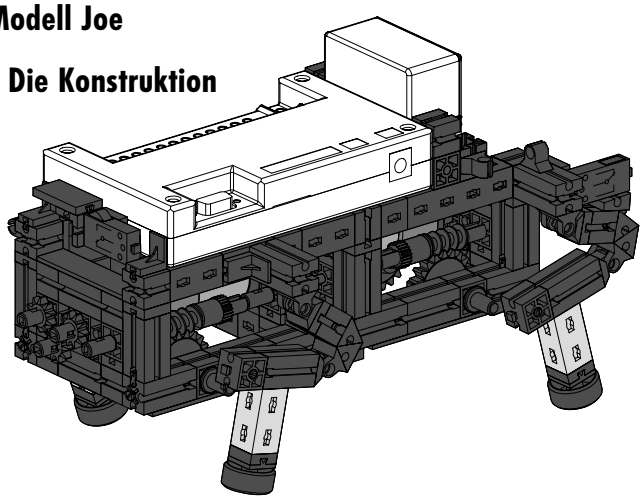


Du kannst dir sicher vorstellen, dass eine Gangart, bei der der Bodenkontakt zeitweise verloren geht, für ein fischertechnik-Modell, auf dem sich das Interface und der Akkupack befinden sollen, nicht unbedingt geeignet ist.

Versuchen wir es also mit der Gangart „Schritt".

4.2 Modell Joe

4.2.1 Die Konstruktion



Baue das Modell so zusammen, wie es in der Bauanleitung ab S. 20 beschrieben ist.

Die Konstruktion der Beine ist gleich wie bei Mike. Die Stellung der Kurbeln, die die Beine antreiben, muss bei Joe völlig anders sein. Die Kurbeln sind zueinander um jeweils 90° versetzt. Du musst sie genau so justieren, wie in der Bauanleitung angegeben. Die Synchronisierung der linken und rechten Seite erfolgt wieder über die beiden Taster E1 und E2. Damit erhalten wir die benötigte Schrittfolge.
Damit uns das Modell nicht umkippt, sobald ein Bein angehoben wird, muss der Schwerpunkt des Modells so liegen, dass das Modell im richtigen Augenblick kippt und von dem jeweils gerade entlasteten Bein aufgefangen wird.

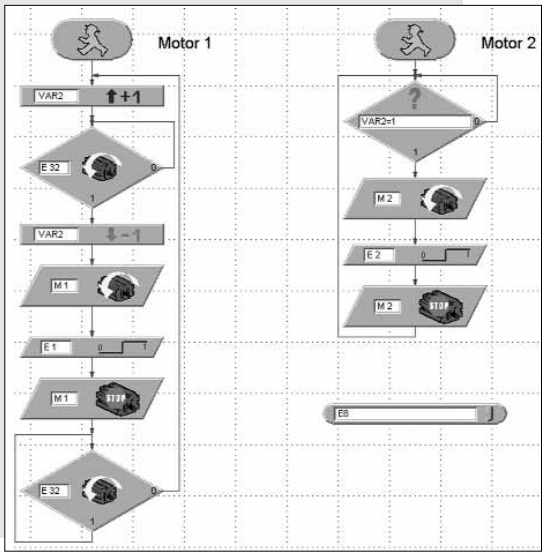
4.2.2 Die Programmierung

Bei diesem Modell wollen wir uns mit dem Geradeauslauf begnügen.

Aufgabe 1:
Programmiere Joe so, dass er sich in der Gangart „Schritt" vorwärts bewegt.

Tipps:
Verwende für jeden Motor einen getrennten Ablauf und synchronisiere die beiden Seiten mit den Tastern E1 und E2.
Benutze E8 als Reset-Taster.

Lösung:



Bei diesem Programm benutzen wir die 0-1 Flanke der Taster für die Synchronisierung. In dem Moment, wenn die Taster gedrückt sind, haben die Kurbeln aller 4 Beine die richtige Stellung zueinander. Das Projekt nennen wir JOE.MDL.

Du siehst, dass sich Joe weitaus schwerfälliger bewegt als Mike und Jack. Durch die notwendige Gewichtsverlagerung schwankt der Körper ziemlich stark und der Laufstil ist bei weitem nicht so elegant wie bei den Sechsbeinern.

Falls du noch Lust hast, diesem Modell das Kurvenlaufen beizubringen, versuche einfach, ob du es schaffst. Viel Glück dabei.

5. Laufen auf zwei Beinen

5.1 Zweibeinige Läufer

Das Laufen auf zwei Beinen ist nicht in der Gattung der Säugetiere entstanden, sondern wird auch von einigen Reptilienarten praktiziert. Warane, Leguane, Agamen und Rennechsen benutzen auf der Flucht nur ihre Hinterbeine. So erreichen sie sehr große Schrittweiten und werden dadurch unheimlich schnell. Sie benötigen dazu kräftige Hinterbeine, einen langen Balancierschwanz und ebenes Gelände.

Vögel gehören ebenfalls zu den Zweibeinern. Zu den schnellsten Laufvögeln gehört der Strauß. Er erzielt Dauergeschwindigkeiten von bis zu 60 km/h.

Den perfektesten Zweibeiner stellt der Mensch dar. Der völlig aufrechte Gang verlangt die Streckung des Hüftgelenks. Diese wird durch den großen Gesäßmuskel gewährleistet. Außerdem können die Beine im Kniegelenk „einegerastet" werden und so in einer energiearmen Haltung fixiert werden.

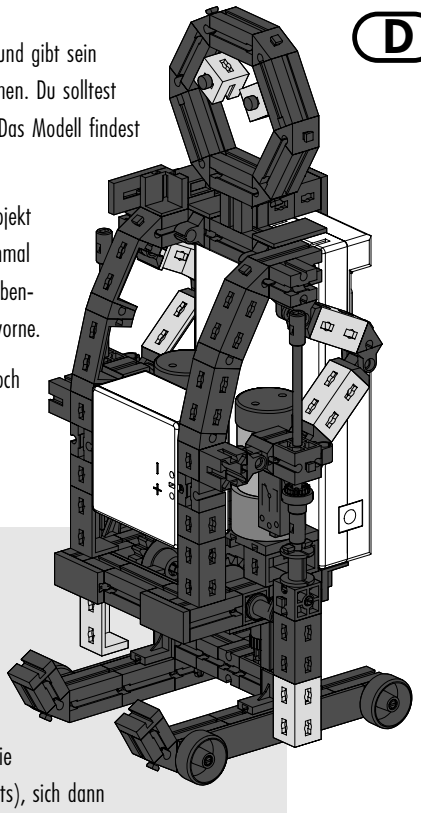
Die Fortbewegung auf zwei Beinen stellt die schwierigste aller Gangarten dar, denn sie erfordert neben den beschriebenen anatomischen Voraussetzungen einen ausgeprägten Gleichgeweichtssinn. Uns Menschen erscheint das Gehen auf zwei Beinen selbstverständlich und einfach. Doch wenn wir uns vor Augen führen, dass beim Abheben eines Beines der ganze Körper nur auf einem Bein ruht und so ausbalanciert werden muss, erkennen wir, dass gerade das Halten des Gleichgewichts diese Fortbewegungsart so kompliziert gestaltet. Selbst ein neugeborener Mensch ist nicht sofort in der Lage auf zwei Beinen zu gehen. Er krabbelt zuerst „auf allen Vieren", bevor er sich aufrichtet und „Laufen lernt".

An der Waseda-Universität in Tokio wurden bereits zweibeinige Roboter entwickelt, die sich mit Hilfe zahlreicher Gelenke, verschiedenster Sensoren, Kameras und leistungsfähiger Mikroprozessoren bewegen und durch Gewichtsverlagerung das Gleichgewicht halten.

Für unseren Baukasten Bionic Robots jedoch wäre das zu aufwendig und zu kompliziert. Wir haben gesehen, dass wir bereits beim Laufen auf vier Beinen mit einem fischertechnik-Modell langsam an unsere Grenzen stoßen.

5.2 Modell Jim

Damit wir dieses Kapitel aber nicht nur theoretisch betrachten, haben wir uns zum Schluss noch erlaubt, wenigstens einen zweibeinigen Skifahrer, wir nennen ihn Jim, zu konstruieren. Er hat zwar wenig mit einem zweibeinigen Läufer zu tun, ist aber unheimlich nett und gibt sein Bestes, um irgendwie vorwärts zu kommen. Du solltest dir diesen Spaß nicht entgehen lassen. Das Modell findest du in der Bauanleitung auf S. 27.



Als Programm kannst du einfach das Projekt JOE.MDL verwenden. Du musst nicht einmal was daran verändern. Jim funktioniert ebenfalls damit und stochert langsam nach vorne.

Eine Aufgabe wollen wir dir nun aber doch noch stellen:

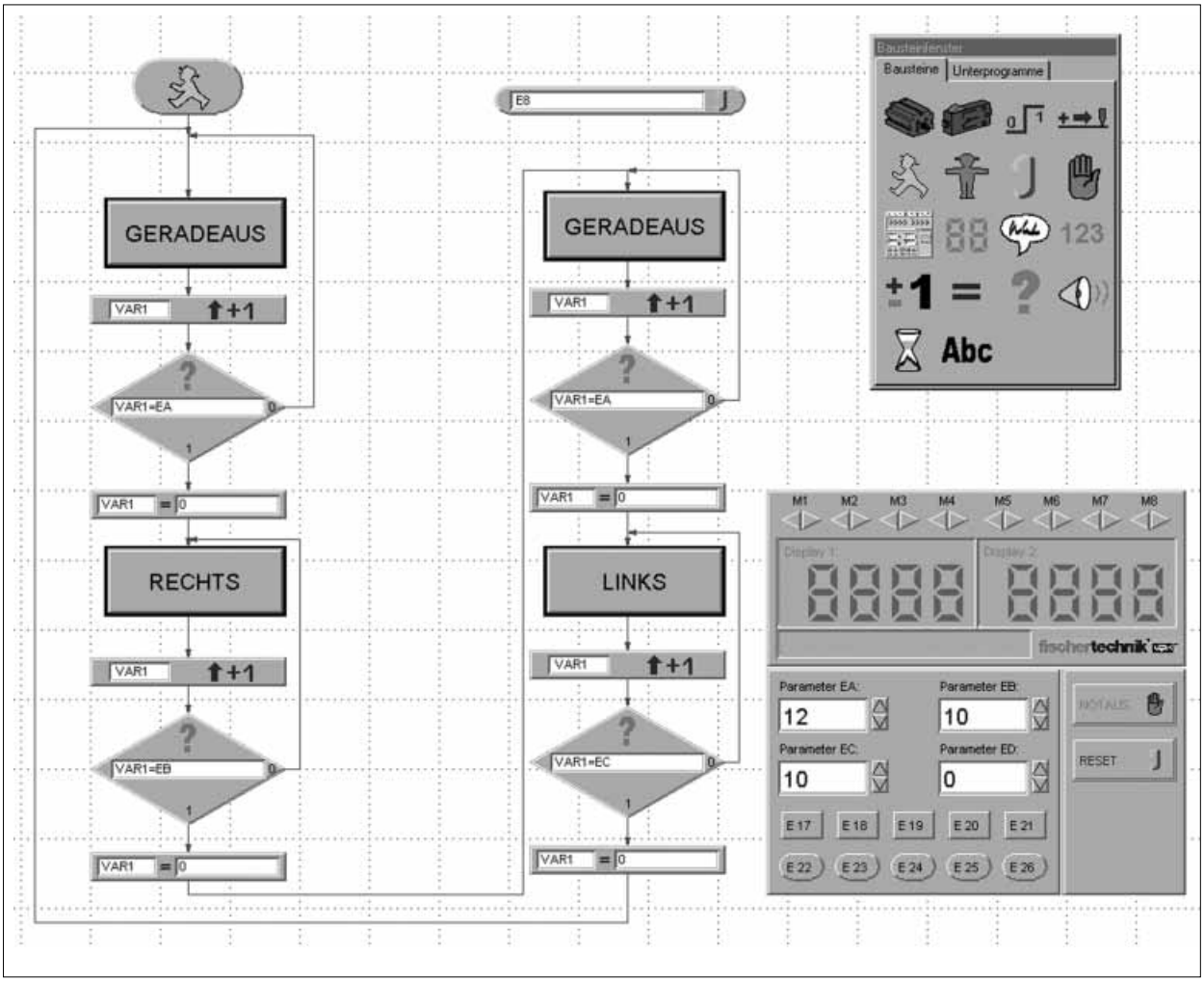
Aufgabe 1:
Programmiere Jim so, dass er ca. 50 cm vorwärts läuft, sich dann nach rechts um 180° dreht, die gleiche Strecke zurück läuft (vorwärts), sich dann 180° nach links dreht, wieder die gleiche Strecke läuft usw. Verwende für die Anzahl der Schritte geradeaus den Terminalparameter EA, für die Anzahl der Schritte links EB und für rechts EC. Verwende wieder E8 als Reset-Taster.

Tipps:
Speichere das Projekt JOE.MDL unter JIM.MDL ab. Mache darin aus dem Hauptprogramm ein Unterprogramm „Geradeaus" (Bausteine markieren und ausschneiden, über BEARBEITEN – UNTERPROGRAMM neues Unterprogramm erstellen, Bausteine einfügen, SUBIN und SUBOUT ergänzen, siehe auch LLWin-Handbuch).

Erstelle aus diesem Unterprogramm mit dem Befehl UNTERPROGRAMM – KOPIEREN die benötigten Unterprogramme LINKS und RECHTS. Ändere darin die Motordrehrichtung und Abfrage der Motordrehrichtungen entsprechend ab und verwende für jedes Unterprogramm eine andere Steuervariable für Motor M2.

Danach programmierst du das Hauptprogramm ähnlich wie bei MIKE_TANZ.MDL. Nur dass du für die Anzahl der Schritte die einstellbaren Terminalparameter EA-EC verwendest. Wie viele Schritte Jim benötigt um sich um 180° zu drehen bzw. einen halben Meter vorwärts zu kommen, musst du ausprobieren.

Lösung:
Nachfolgend bilden wir das Hauptprogramm ab. Die Unterprogramme kannst du dir falls nötig wieder direkt am Bildschirm ansehen. Auch bei uns heißt das Projekt JIM.MDL.



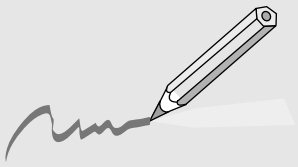
Wir haben in dem Projekt gleich noch das Unterprogramm ZURUECK ergänzt, auch wenn es hier direkt nicht benötigt wird. Aber bestimmt willst du, dass Jim noch andere Wege geht. Vielleicht muss er sich dabei ja auch einmal rückwärts bewegen.

6. Zusammenfassung

Auf deiner Reise durch die Welt der Bionic Robots von fischertechnik hast du sicher festgestellt, dass es nicht immer ganz leicht war, die vier Jungs zum Laufen zu bringen. Es ist eben schon schwieriger, sich auf Beinen fort zu bewegen, als einfach auf Rädern zu rollen. Besonders das Programmieren der Synchronisierung zwischen der linken und rechten Seite beim Drehen nach links oder rechts erfordert etwas Konzentration. Aber für diejenigen, die sowieso mehr Spaß am Bauen der Modelle finden, haben wir ja alle Programme fix und fertig auf die CD gebrannt, so dass jeder die Modelle bauen und betreiben kann.

Falls du zu den Profi-Programmierern gehörst, hast du sicherlich noch viele Ideen, welche Aufgaben man für Mike, Jack, Joe oder Jim noch programmieren kann, sei es mit zusätzlichen Sensoren, damit sie nicht vom Tisch fallen, oder dass sie sich in einem Labyrinth zurecht finden.

Mit zusätzlichen Bauteilen kannst du sie auch noch mit einem Kopf, Rüssel, oder Schwanz ausstatten. Deiner Phantasie sind dabei keine Grenzen gesetzt. Lass dir was einfallen!



1. Bionic – Nature as a Model

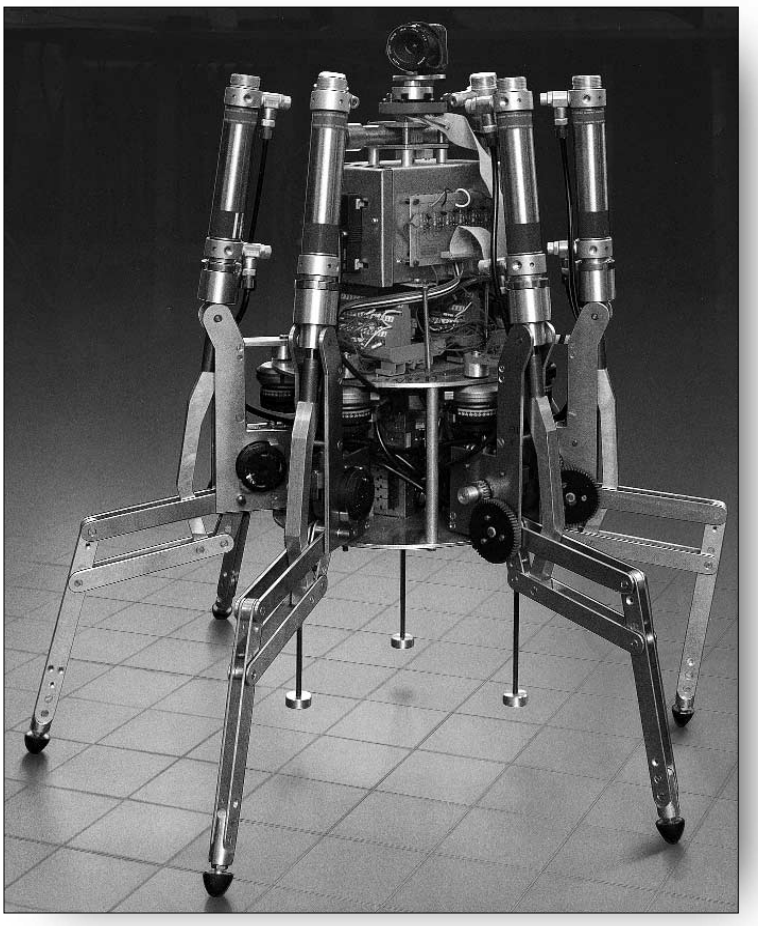
The term bionic is composed of the two terms biology and technique. This branch of science always tries to base its technical solutions on nature. Man has invented machines time and time again with the objective of moving farther, faster and more efficiently than possible naturally. This has been achieved in different ways corresponding to the respective requirements. Vehicles roll on wheels. In difficult terrains, which vehicles with wheels cannot handle, tracked vehicles or Caterpillars are used. Ships float on water or can operate underwater. In a few types of movement, nature serves as the model. For example, an airplane resembles a flying bird.

Over the past few years, scientists have been looking into another very widespread movement form: walking or running. Robots have been developed, which can move on legs. Such walking machines can be used in all places where wheel and tracked vehicles would hardly have a chance, for example, in very uneven or soft terrains, for climbing over obstacles, going up stairs, crossing ditches or for use in difficult to access and dangerous spots in nuclear power plants, mines or for rescue operations.

The first serious experiments in developing walking machines were conducted at a university in Tokyo in 1967. For the first time, work was oriented to the human way of walking instead of that of insects. The continual further development of these experiments results in the first two-legged walking machine in 1985. Today, these robots have more then 50 degrees of flexibility and numerous microprocessors. For example, they can read notes and play the organ with the help of a camera. You can even have a discussion with them.

An example of a six-legged walking robot is the electro-pneumatic walking robot Achilles, developed at the Royal Military Academy in Brussels. Equipped with a camera at the top and on its six legs, this robot reacts to raised or recessed obstacles (objects or holes).

Now fischertechnik has also devoted its attention to this exciting topic and has constructed robots, which are wakened to life with the Intelligent Interface and the LLWin software.



2. Requirements and Startup

You need the following articles in addition to the construction kit, so that you can build the models of the computing kit "Bionic Robots."

Intelligent Interface, Art. no. 30402
LLWin software (from Version 3.0), Art. no. 30407
Power supply unit, Art. no. 34969

If you are not familiar with the LLWin software and the interface, you should first read through the LLWin software manual. It describes how you install the software and connect the interface. It is also very well suited for getting acquainted with how you can control fischertechnik models via a PC. Using a few components from the construction kit (motor and pushbuttons), you can first construct very simple model controls.

As soon as you are familiar with the software and the interface, you can then tackle the more challenging Bionic Robot models.

A CD-ROM is included in the construction kit, which contains LLWin example programs for the kit models. You need the LLWin software starting from Version 3.0 to open the programs. You can either leave the example programs on the CD and call them from LLWin with the command File - Open, or copy the complete BIONIC_ROBOTS folder from the CD in the project directory of LLWin onto your hard disk and open the examples from there.

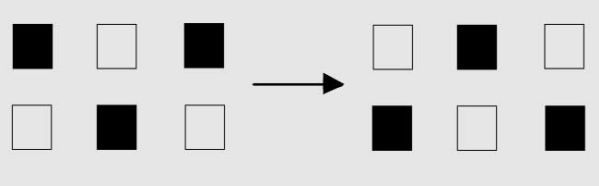
Before you construct the models, you need to assemble a few parts, e.g., cables and plugs. The construction instructions explain exactly what you have to do.

Now we're ready to start. You can now delve into the fascinating world of walking fischertechnik robots. As soon as you have constructed the first model and it starts to move in an almost eerie way, you will be thrilled by this technique, which has already been used in nature for moving for millions of years.

3. Walking on Six Legs

3.1 The Way Insects Walk

The way insects walk is excellently suited as a model for the drive of mechanical six-legged robots. In what is called a three-foot gait, three of the six legs are always raised at the same time from the ground: the front and back leg of one side together with the middle leg of the other side.

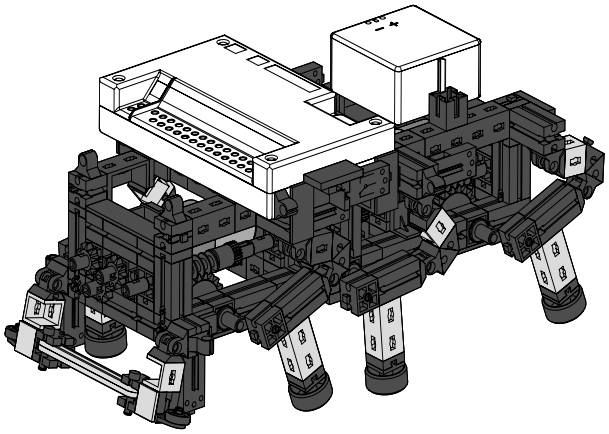


The legs, which remain on the ground (displayed in black), form a stable three-legged construct, so that the model always remains stable upright and does not fall over during walking.

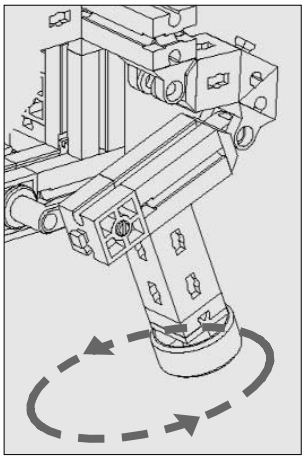
3.2 Model Mike

3.2.1 Assembly of the Model

Build the six-legged robot Mike now (see the assembly instructions on page 4). Load the power unit while you assemble it, so that you have sufficient energy to drive the model later.



The legs of the module are designed in such a way that a four-bar mechanism is produced. The construction type of the four-bar mechanism is called a crank and rocker mechanism. Driven by a crank, the movable elements of the gears sway back and forth. The distances between the individual joints and the position of the base (the lower end of the leg) are selected so that the base moves elliptically when the drive crank rotates. This creates movement, which is similar to a step when walking. The six cranks, which drive the legs, must be aligned precisely as shown in the assembly instructions. The three legs, which touch the ground at the same time, have the same crank setting. The cranks of the three legs, which are in the air at this time, are rotated by 180° for this. The correct setting of the cranks in relation to each other ensures that the model can walk in the correct step sequence, the three-foot gait.



The binding pieces and nuts, with which you attach the screws and toothed gears to the axles must be tight, so that they do not shift during the walking.

The right and left sides of the model are each driven by a motor (this is required for walking around curves). You must make certain for this that the middle leg on the one side is always in the same position as the front and back legs on the other side. This is synchronized by the software via the pushbuttons E1 and E2.

Use the interface diagnosis to test whether all pushbuttons and motors are connected correctly. Rotation direction of the motors: counter-clockwise rotation = forward.

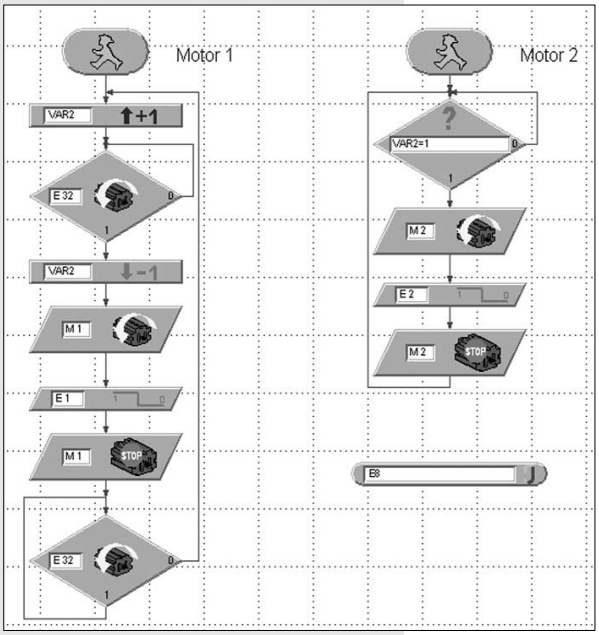
3.2.2 The First Program

Now we are going to start to teach Mike something. The model should first walk straight. We will deal with walking around curves and reacting to obstacles later.

Task 1:
Program the model, so that it walks straight with a three-foot gait. Use the pushbuttons E1 and E2 for synchronizing the left and right legs. Make certain that the front and back legs on one side and the middle leg on the other side always have the same position. Also use pushbutton E8 as reset button.

Tips:
Program a separate control process for each motor. Control the process for motor M2 using a variable VAR2.
If you do not need the interface during programming, interrupt the power supply between the power unit and the interface to save energy.

Solution:
The program for walking straight looks like this:



The variable VAR2 provides the impulse for starting motor M2. The motor M1 is started. As soon as pushbutton E1 is activated, M1 stops. As soon as E2 is activated, M2 stops. The first process waits until M2 has stopped (the status of motor M2 is queried via E32; also refer to the Querying the Motor Status in the LLWin manual).

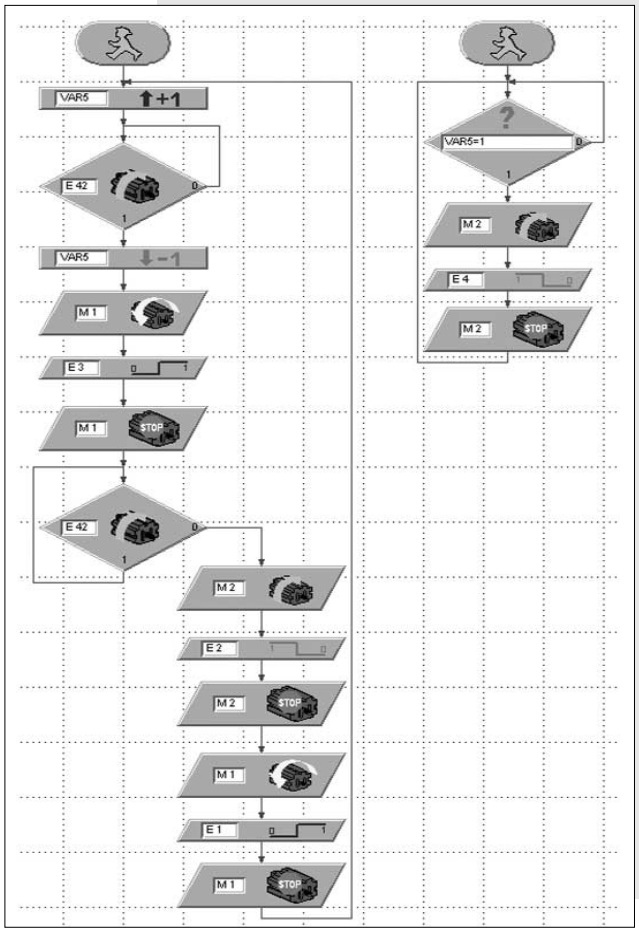
By the way, if you do not feel like creating this process yourself, you can find it as example project MIKE_STRAIGHT.MDL on the enclosed CD.
Start the project. If you have programmed everything correctly, the model comes to life and walks straight ahead. Congratulations! The first step has been taken.

3.2.3 Turning Left

Of course, we are not satisfied that Mike can only walk straight ahead. As the next step, we want him to turn around.

Task 2:
Program Mike, so that he turns to the left (counter-clockwise).

Tips:
The model turns to the left if M1 turns to the left and M2 to the right. Of course, you can also drive the model unsynchronized.
It also turns then, but then there are spots where the model can fall over forward. This can be avoided with the following process:



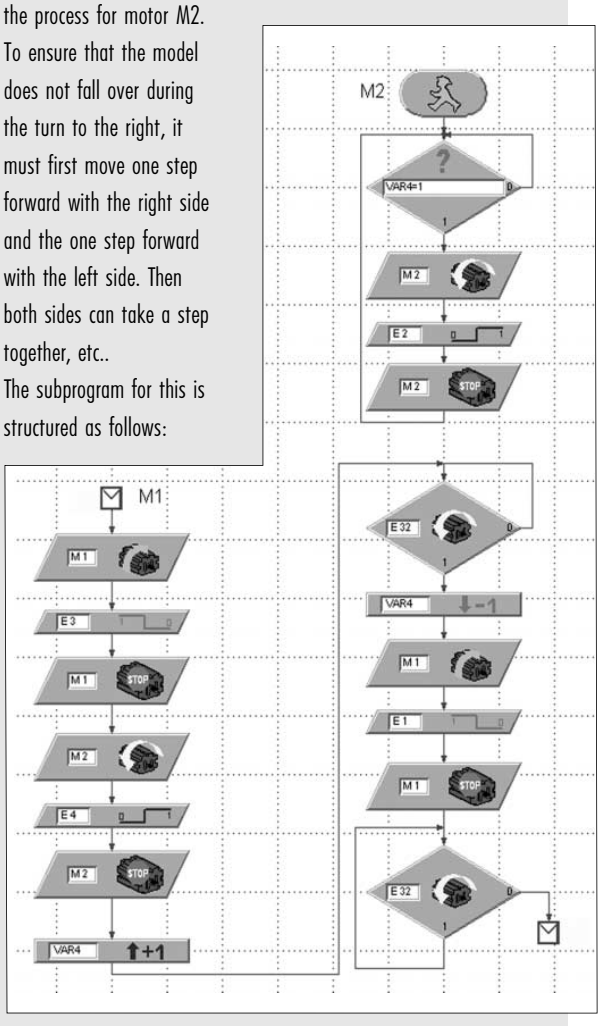
Using the pushbuttons E1-E4, the left and right sides of the model first move together one step, then the left side takes a step, then the right, etc. In this way, the model never falls over forward. Try it! Then it will be easier for you to understand this sequence.
You can also find this process as project MIKE_LEFT.MDL on the CD.

Now the model can walk straight ahead and turn left. Only walking backward and turning to the right are missing. In principle, walking backward functions the same as walking forward, only with reversed motor rotation. Turning to the right (clockwise) functions opposite to turning left in principle.

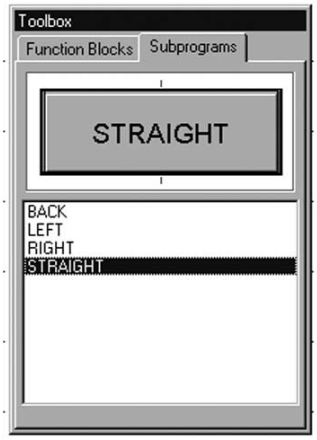
3.2.4 Left, Right, Forward, Backward

Task 3:
Now program each of the functions STRAIGHT, BACK, LEFT and RIGHT as subprograms, so that you can use them flexibly in various projects later.

Tips:
The procedure for copying an existing process into a subprogram is described in the LLWin manual.
Use a different variable (VAR2-VAR5) in each subprogram to start the process for motor M2.
To ensure that the model does not fall over during the turn to the right, it must first move one step forward with the right side and the one step forward with the left side. Then both sides can take a step together, etc..
The subprogram for this is structured as follows:



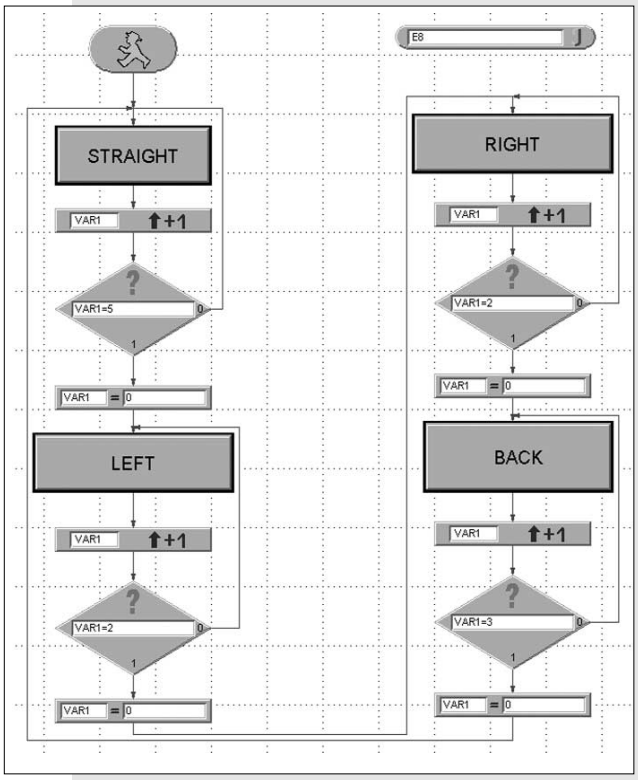
We have not printed the other subprograms here. If you have difficulty programming a process, you can find the finished subprograms in the MIKE_TEMPLATE.MDL file on the CD. The main program of this project is blank. You can find a list with existing subprograms, which you can insert in the main program, in the function block under the Subprograms heading.



But don't look right away there to see how it works. First try to figure out the solution yourself. If you do not succeed, you can always look up the solution. To try out all subprograms, we now want to get Mike to dance.

Task 4:
Program Mike, so that he takes five steps forward, turns two steps to the left, then two steps to the right and then three steps back before starting all over again. Use the variable Var1 as count variable for the number of steps. Use E8 as reset pushbutton.

Solution:



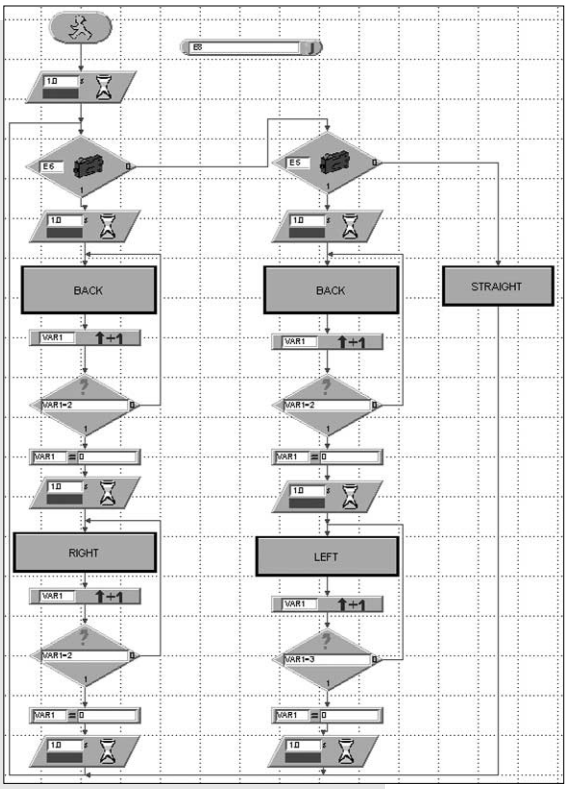
This project is called MIKE_DANCE.MDL.

3.2.5 Detecting Obstacles

As a last step, we want to get Mike to detect obstacles with his movable fender ("sensor" would be a better term) and avoid them.

Task 5a:
Program Mike to that when he detects an obstacle at his left sensor (pushbutton E6), he first takes four steps back and then two steps to the right. If there is an obstacle at his right sensor (pushbutton E5), he should take four steps back and then three steps to the left.

Solution:



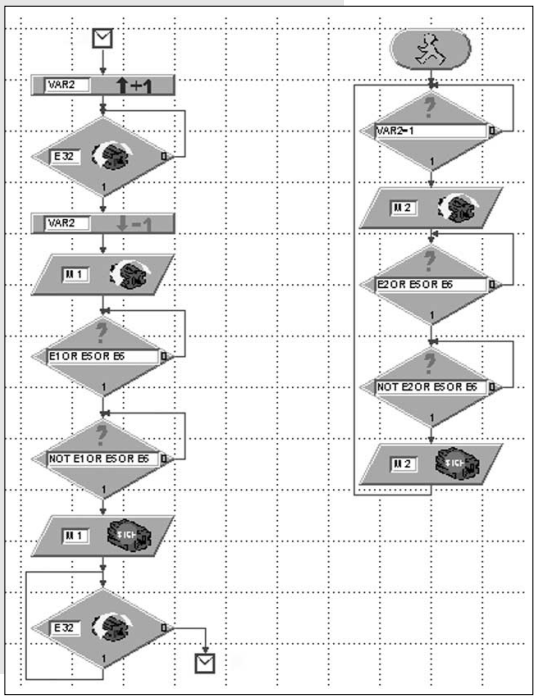
Mike first walks straight ahead. After each step, the pushbuttons E5 and E6 are queried. If E6 is pressed, the program branches to the left process (first back, then right). If E5 is pressed, the program branches to the middle process (first back, then left).

Because the pushbuttons E5 and E6 are only queried after each full step, it takes a relatively long time until Mike reacts to an obstacle.

Task 5b:
Optimize the STRAIGHT subprogram, so that Mike can react more quickly to an obstacle.

Tip:
Do not use the EDGE function block but instead the COMPARE function block for querying the pushbuttons E1 and E2. Also query whether E5 or E6 is pressed.

Solution:

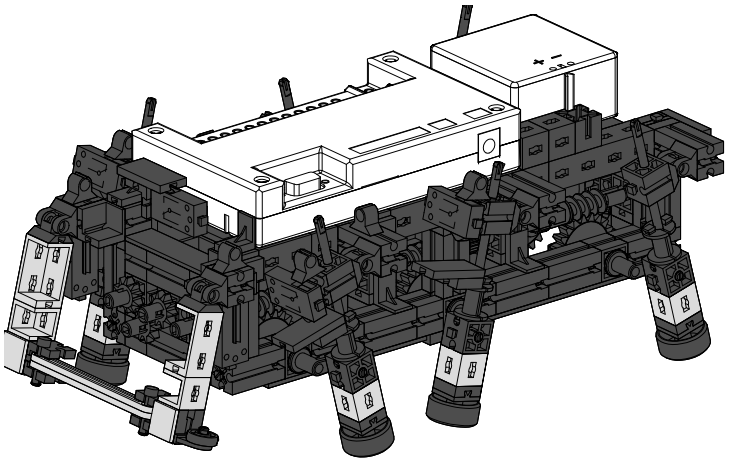


Now Mike should function perfectly. This program is also on the CD under MIKE_OBSTACLE.MDL. You can also integrate the improved subprogram in the MIKE_TEMPLATE.MDL project. If E5 and E6 are not queried in another program, that does not matter at all. We have stored the improved template under MIKE_TEMPLATE_OBSTACLE.MDL.

Now that we have dealt with the first six-legged robot in detail, let's turn to the second model, which also has six legs. We call it Jack.

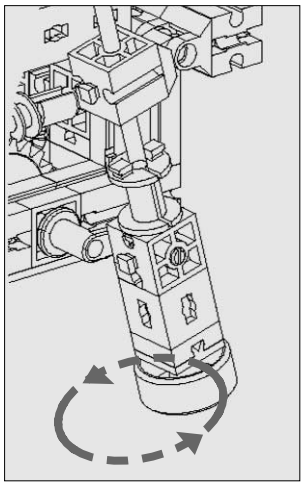
3.3 Model Jack

Jack is also one of the six-legged fischertechnik models. However, the design of his legs differs considerably from those of Mike's.
Now build the model as described in the assembly instructions starting on page 12. By the way, the assembly steps 1-13 are the same for Mike and Jack. Consequently, you need not disassemble Mike completely before you start to build Jack.



3.3.1 The Design

The leg design of Jack also involves a four-bar mechanism. The construction type used here is called an inverted slider crank. The connecting shaft is ositioned in a movable longitudinal guide, which swings back and forth when the crank rotates. The curve, which the base of the leg moves along, does not have such an ellipse shape as the model Mike does, but is more of a circle.



As a result, Jack's body goes up and down more during walking than that of Mike. The steps are shorter. But Jack can overcome small obstacles, which Mick cannot. This gear design also reminds us more of a leg than is the case with that of Mike's. When Jack walks, it looks like he is walking on stilts.
He also moves in the three-foot gait of insects.
It is also important for this model to align the cranks precisely as in the assembly instructions, and to tighten the binding pieces and nuts well.

3.3.2 The Programming

It is logical that you can use programs for Jack with which Mike functioned too. Try it!

Task 1:
Operate Jack using the MIKE_OBSTACLE.MDL program. What can you observe?

Observation:
The model walks forward and backward without any problems. But it falls over forward when it turns to the left or right.

Task 2:
How can you explain this?

Solution:
The two models have different leg designs. The pushbuttons E1-E4 are also activated at a different position on the leg. Consequently, the way in which Mike turns need in no way function for Jack – bad luck.

Of course, we don't like this at all and want to find a solution as quickly as possible.

Task 3:
Try to program Jack, so that he detects obstacles as Mike does, but does not fall over forward when he turns.

Tips:
Save the MIKE_OBSTACLE.MDL project under the name JACK_OBSTACLE.MDL and make the necessary changes there.

When Jack turns, both motors can also run simultaneously. The alternating switching on and off is eliminated. The legs must be in the correct starting position at the beginning of turning, i.e., after walking backward.

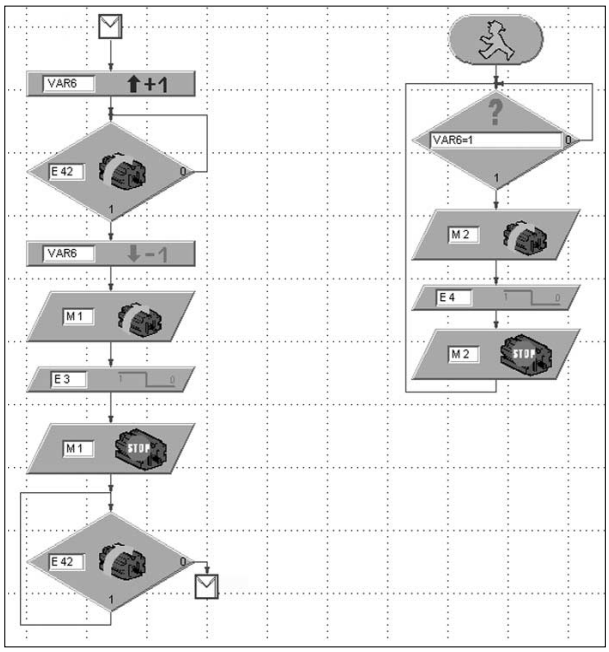
Left turn (counter-clockwise):
If the model should turn to the left, the crank of the front left leg must point backward at the beginning of the turn, and that of the front right leg must point forward. This is the case when the pushbutton E2 on the left side and the pushbutton E1 on the right side are pressed during walking backward and then released again.

Right turn (clockwise):
If the model should turn to the right, the crank of the front left leg must point forward at the beginning of the turn, and that of the front right leg must point backward. This is the case when the pushbutton E4 on the left side and the pushbutton E3 on the right side are pressed during walking backward and then released again.

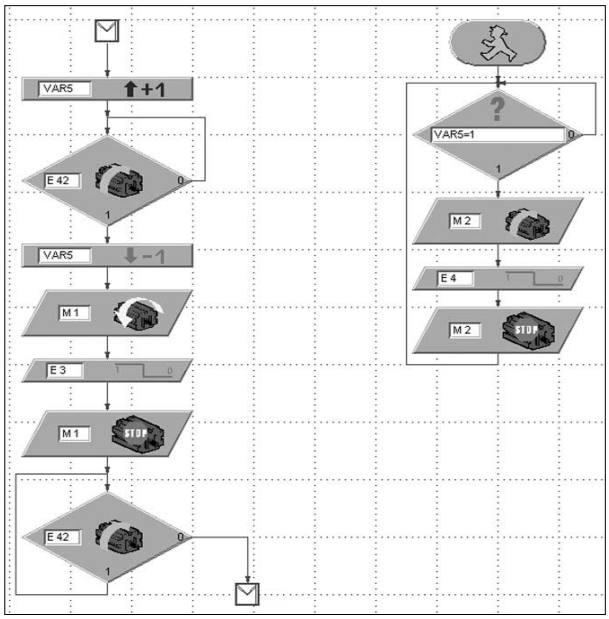
Rather complicated, isn't it? But don't worry; we'll have it solved soon.

Two different subprograms must be used for the backward movement. If the model should turn left, you synchronize the steps during walking back with the pushbuttons E1 and E2. This corresponds to the BACK subprogram from the MIKE_OBSTACLE.MDL project.
If the model should turn right, you synchronize the steps during walking back with the pushbuttons E3 and E4.

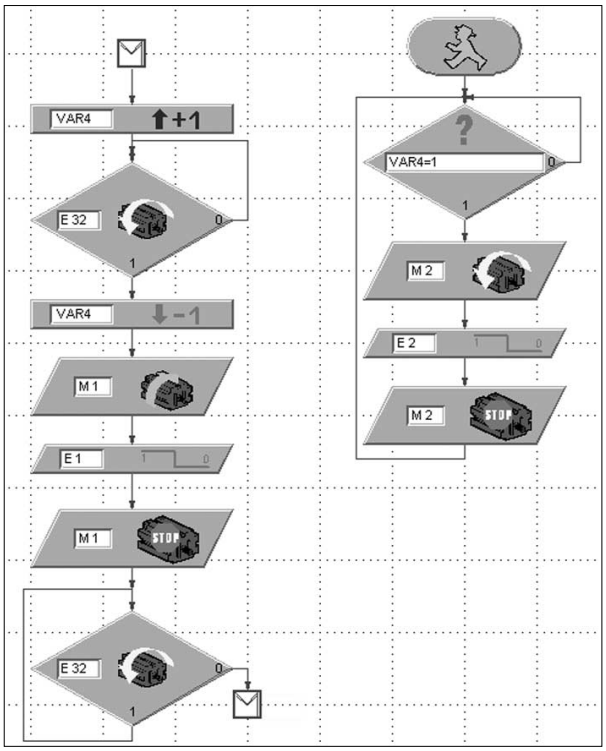
Rename the subprogram BACK, and rename it to BACK_L using the command SUBPROGRAM - RENAME. Then copy it using SUBPROGRAM - COPY into a second subprogram BACK_R. Change the pushbuttons for the synchronization there to E3 and E4. Don't forget to use a new variable VAR6 for BACK_R for the synchronization; otherwise, you will have perfect chaos. BACK_R then looks like this:



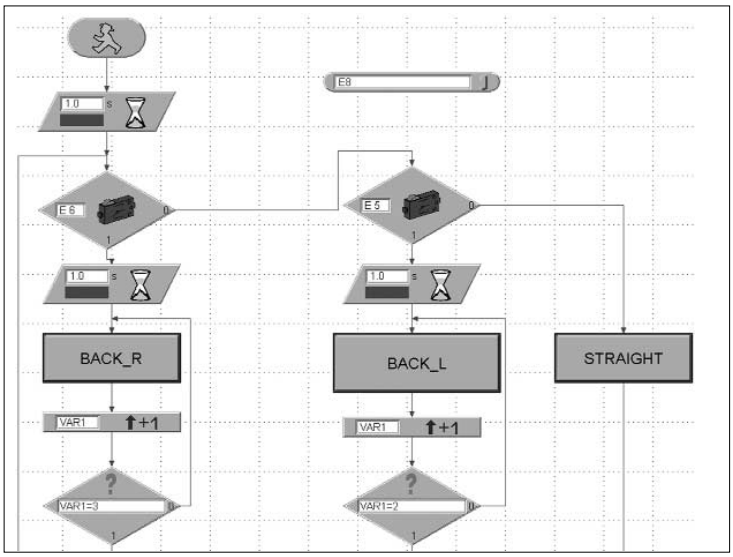
Now you need to change the subprograms for the turning itself, so that the two motors always run simultaneously. The LEFT subprogram is composed of the following blocks:



You can see that a few blocks can be eliminated compared to the subprogram in the MIKE_OBSTACLE.MDL project.
The subprogram for the turn to the right looks similar, only with different motor rotation directions. The pushbuttons E1 and E2 are also used for synchronizing the two motors.



First replace the subprogram BACK_L with BACK_R for turning to the right in the branch in the main program.



The rest of the main program remains unchanged.

You did it! If you didn't make any mistakes, Jack should now walk without falling over when he turns. If something does not function and you have no idea why, don't worry about it because it really was a hard nut to crack. At any rate, you can simply call the finished JACK_OBSTACLE.MDL project from the CD and run the model with it.

If you figured out how to solve the problem yourself, you have reason to be proud, because now you are a pro among programmers.

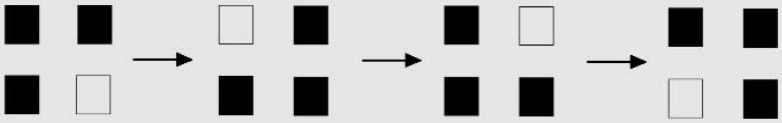
GB+USA

4. Walking on Four Legs

4.1 The Way Mammals Walk

To design a walking robot with four legs, let's use nature as a model again and take a look at how mammals move.

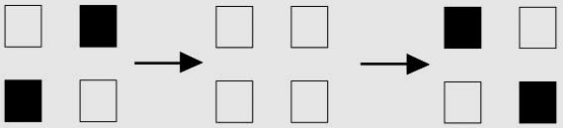
The slowest and most reliable gait is a step. One leg looks for a new place to position itself, while the body of the animal is support on three legs. The animals move by taking steps on opposite sides in the following sequence: rechtes Vorderbein, linkes Hinterbein, linkes Vorderbein, rechtes Hinterbein vorwärts. The black boxes represent the legs, which are on the ground, and the white boxes represent the lifted leg.



If we want to use this gait for a walking robot, we need to consider the following:

Imagine that you saw off one leg from a table with four legs. What happens? Right, the table falls over. Consequently, the three legs no longer form a stable tripod, as was the case with the six-legged models. This complicates the design of a four-legged robot.

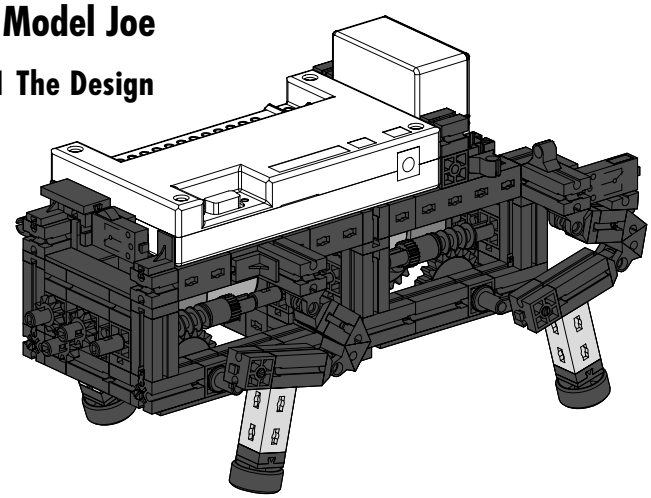
The faster mammals move, the more instable their gait becomes. Let's take a brief look at the gait trotting. The legs are lifted synchronously in a diagonal direction during trotting. But before they touch the ground, the two other legs are already lifted. This means that there is no contact at all to the ground at times.



You can certainly imagine that a gait in which there is no contract to the ground at times is not necessarily suited to a fischertechnik model, in which the interface and power unit should be contained.
Consequently, let's try it using the gait "step."

4.2 Model Joe

4.2.1 The Design



Now build the model as described in the assembly instructions starting on page 20.

The design of the legs is the same as for Mike. But the position of the cranks, which drive the legs, must be completely different for Joe. The cranks are offset from one another by 90°. You must align it exactly as described in the assembly instructions. The left and right sides are again synchronized via the two pushbuttons E1 and E2. Then we get the required step sequence.
To make sure that the model does not fall over as soon as a leg is lifted, the center of gravity of the model must be set so that the model tips at the correct moment and is supported by the leg taking the load just then.

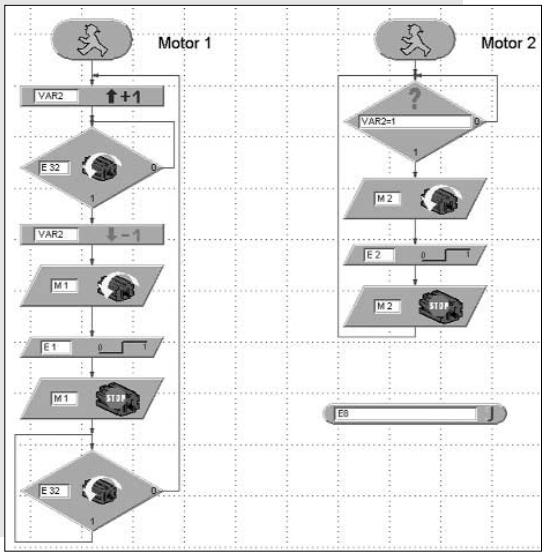
4.2.2 The Programming

We will only program walking straight ahead for this model.

Task 1:
Program Joe, so that he moves forward in the gait step.

Tips:
Use a separate process for each motor, and synchronize the two sides using the pushbuttons E1 and E2.
Use E8 as reset pushbutton.

Solution:



fischertechnik®

We use the 0-1 edge of the pushbuttons for synchronizing in this program. In the moment when the pushbuttons are pressed, the cranks of all four legs have the correct position with respect to each other. We'll call the project JOE.MDL.

You can see that Joe moves a lot more slowly and ponderously than Mike and Jack. Due to the necessary shifting of weight, the body wavers rather strongly and the walking style is far from being as elegant as that of the six-legged robots.

If you feel like it, try to get this model to walk around curves. Simply try to see if you can do it. Good luck!

5. Walking on Two Legs

5.1 Two-Legged Walkers

Walking on two legs did not originate in the species of mammals, but instead was first practiced by a few reptiles. Monitor lizards, iguanas, agamidae and run-lizards only use their hind legs when running away. This lets them achieve big steps and consequently become very fast. They need strong hind legs for this, a long balancing tail and flat terrain.

Birds are also two-legged creatures. The ostrich is the fastest among the birds that run. It can achieve continuous speeds of up to 60 km/h.

The most perfect two-legged creature is man. The completely upright gait requires the stretching of the hip joint. This is achieved by the large buttocks muscle. The legs can also be "locked in place" in the knee joints and consequently fixed in an energy-saving position.

Movement on two legs is the most difficult of all gaits, because it requires a pronounced sense of balance in addition to the described anatomic prerequisites. People think of walking on two legs as a matter of course and simple. But if we consider that when one leg is lifted, the whole body is resting on only one leg and must be balanced, we realize that keeping your balance during this movement type is structured in a very complicated fashion. Even a newborn person is not able to walk immediately on two legs. He first crawls "on all fours" before he stands up and learns to walk.

At Waseda University in Tokyo, two-legged robots have already been developed, which move using numerous joints, various sensors, cameras and high-performance microprocessors and maintain their balance by shifting their weight.

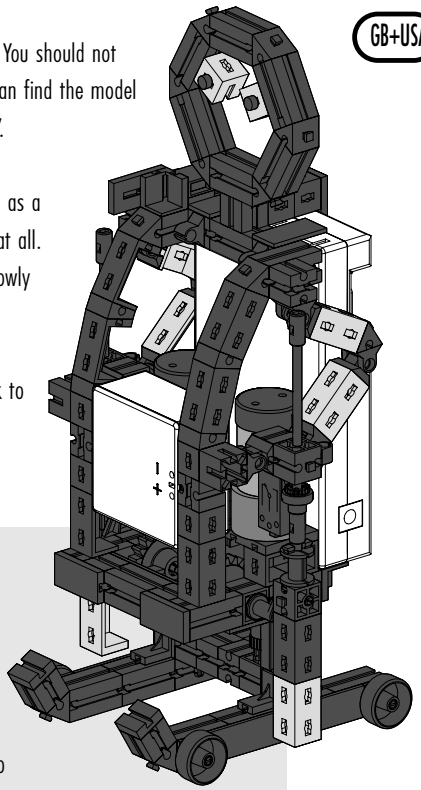
But that would be too much work and complicated for our Bionic Robots construction kit. We have seen that we are already reaching the limits of what we can do with walking on four legs with a fischertechnik model.

5.2 Model Jim

But to make sure that we do not only consider this topic in theory, we have designed a two-legged skier here at the end. We call him Jim. Although he does not have a lot to do with two-legged walkers, he is very nice and tries his best to move forward in some way. You should not miss the chance to have this fun. You can find the model in the assembly instructions on page 27.

You can simply use the JOE.MDL project as a program. You need not even change is at all. Jim also functions using it and prods slowly forward.

But we want to give you one more task to perform.



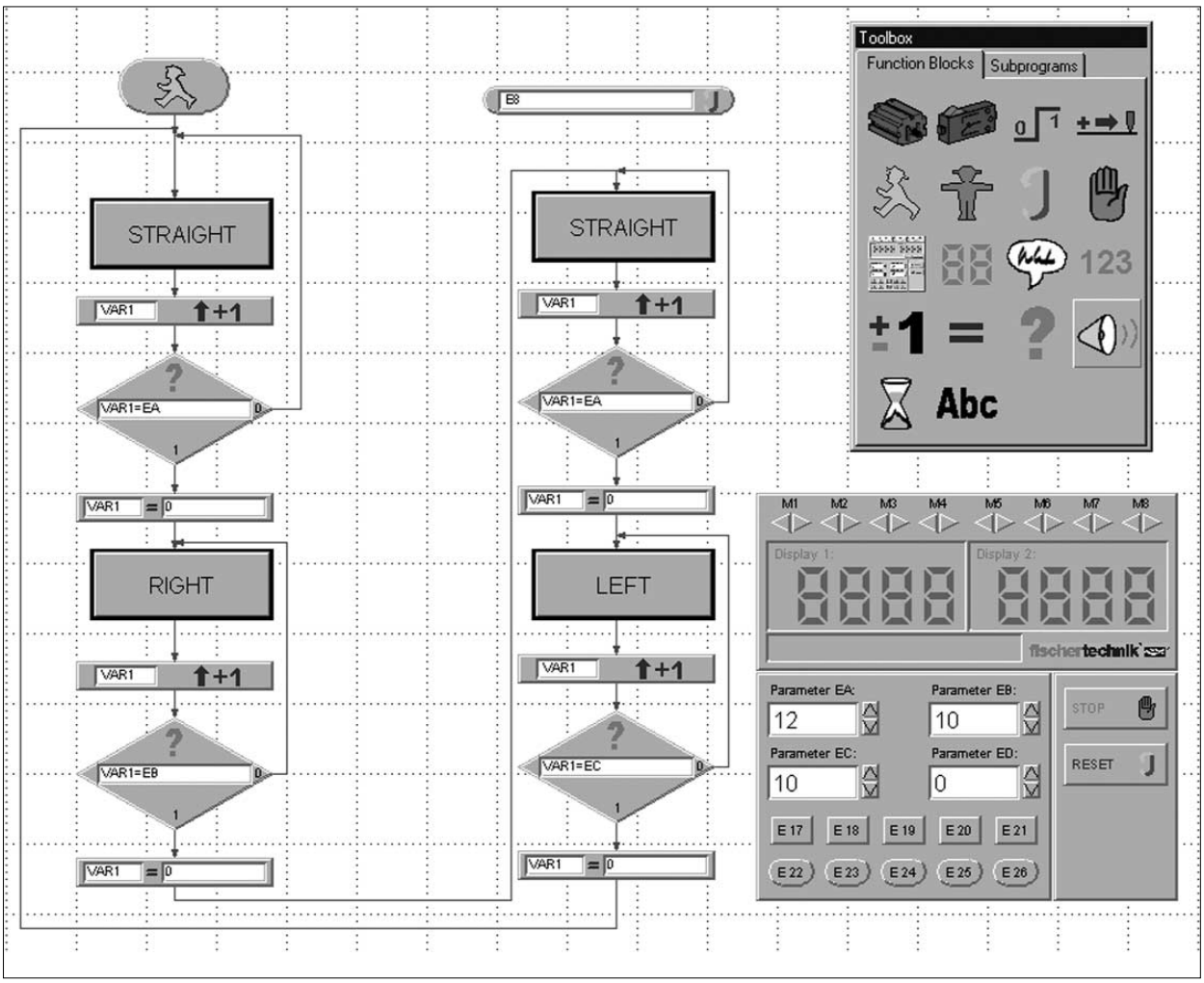
Task 1:
Program Jim, so that he walks approx. 50 cm forward, the turns to the right 180°, walks the same path back (forwards), then turns 180° to the left, walks the same path again, etc. Use the terminal parameter EA for the number of steps straight ahead, EB for the number of steps left and EC for the steps right. Use E8 as reset pushbutton again.

Tips:
Save the JOE.MDL project under JIM.MDL. Make a Straight ahead subprogram out of the main program (select and cut out function blocks, create a new subprogram via EDIT - SUBPROGRAM, insert the function blocks, and supplement SUBIN and SUBOUT; also refer to the LLWin manual).

Create the required subprograms LEFT and RIGHT from this subprogram using the command SUBPROGRAM - COPY. Change the motor rotation direction in it and the query of the motor directions correspondingly. Use a different control variable for motor 2 in each ubprogram.

Then program the main program similar to MIKE_DANCE.MDL, except that you use the settable terminal parameters EA-EC for the number of steps. You have to try out how many steps Jim needs to turn 180° or to move forward a half meter.

Solution:
The main program is shown below. If required, you can see the subprograms directly on the screen. We also call the project JIM.MDL.



We added something directly after the BACK subprogram in the project, even though it was not absolutely necessary here. But you certainly want Jim to take other paths too. Maybe he must also move backward sometimes.

6. Summary

You have most certainly seen on your journey through the world of Bionic Robots from fischertechnik that it was not always easy to get the four guys walking. It is simply more difficult to move on legs than to roll on wheels. The programming of synchronization between the left and right sides for turning right or left especially requires a bit of concentration. But for all those who have fun building the models anyway, we all copied all programs ready-to-use onto the CD, so that everyone can build and operate the models.

If you are among the pros for programming, you certainly have a lot more ideas about which tasks you can program for Mike, Jack, Joe or Jim, either using additional sensors, so that they do not fall off a table or can find their way around in a labyrinth.

Using additional components, you can also equip them with head, snouts or trunks, or tails. There are not limits to your imagination. Use it!

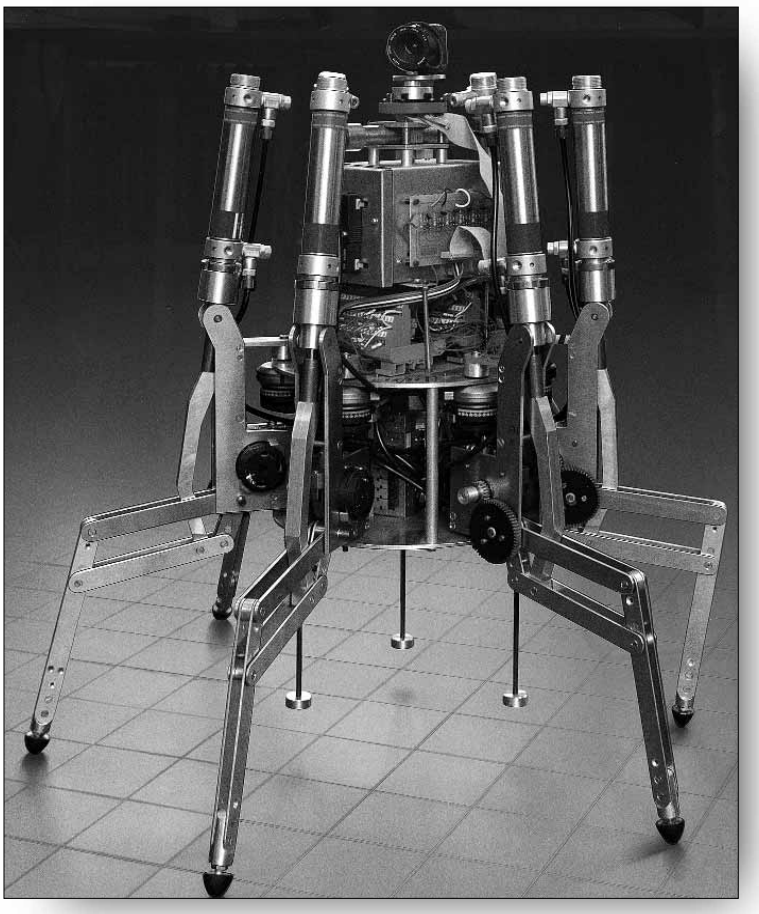
F

1. Bionique – La Nature comme modèle

Le terme Bionique se compose des deux termes Biologie et Technique. Cette bran-che de la science essaie constamment de s'inspirer de la Nature lors de la recherche de solutions techniques.
Ainsi, dans le but de se déplacer plus loin, plus vite et de façon plus efficace qu'il est possi-ble de par Nature, l'Homme a toujours inventé des machines qui le lui permi-rent de différentes manières en fonction des exigences spécifiques. Les véhicules roulent sur des roues. En terrain difficile, là où les véhicules à roues ne fonctionnent plus, on utilise des véhicules à chenilles. Les bateaux flottent sur l'eau ou sont en mesure de plonger. Dans quelques modes de locomotion, la Nature sert également de modèle. Ainsi, un avion ressemble par exemple à un oiseau en vol plané.

Depuis quelques années, les scientifiques s'intéressent à un autre mode de déplacement très utilisé dans la Nature, à savoir la marche ou la course. Ils mettent au point des robots en mesure de se déplacer sur leurs pattes. On pourrait utiliser de telles machines mobiles partout là où les véhicules à roues ou à chenilles n'ont pratique-ment plus de chances de se déplacer, par exemple sur des terrains mous ou extrê-mement accidentés, pour le franchissement d'obstacles ou de fossés, pour monter des escaliers ou pour des interventions sur des lieux dangereux ou difficilement accessibles tels que les cen-trales nucléaires, les galeries de mines ou lors d'opéra-tions de sauvetage.

Les premières tentatives sérieuses de mise au point de machines mobiles eurent lieu en 1967 dans une université de Tokyo. On s'y est inspiré pour la première fois du mode de déplacement humain, et non pas de celui des insectes. Le perfectionne-ment constant de ces tentatives conduisit en 1985 à la première machine mobile bipède. Depuis, ces robots ont acquis plus de 50 degrés de liberté et de nombreux microprocesseurs. À l'aide d'une caméra, ils peuvent par exemple lire des partitions de musique et jouer de l'orgue. On peut même bavarder avec eux.



Un exemple de robot mobile à six pattes est le robot électropneumatique « Achille » mis au point au Prytanée Militaire Royal de Bruxelles. Équipé d'une caméra à sa partie supérieure et à ses six pattes, ce robot doit réagir de façon mécanique aux obstacles en hauteur ou en profondeur (objets et trous). C'est ainsi que fischertechnik s'est également consacré à ce sujet passion-nant et a construit des robots mobiles auxquels il s'agit à présent de « donner vie » à l'aide de l'Intelligent Interface et du logiciel LLWin.

2. Conditions requises et mise en route

Pour pouvoir construire les maquettes du jeu de construction informatique « Bionic Robots », vous aurez besoin, en plus du coffret lui-même, des articles suivants :

Intelligent Interface, Référence 30402
Logiciel LLWin (Version 3.0 ou supérieure), Référence 30407
Bloc d'alimentation Accu Set, Référence 34969

Si vous ne connaissez pas encore le logiciel LLWin ni l'Intelligent Interface, vous devriez tout d'abord lire le manuel du logiciel LLWin. Ce manuel décrit la manière d'installer le logiciel et de raccorder l'interface. En outre, il permet de façon excellente d'acquérir une expérience préliminaire sur la manière de piloter les maquettes de fischertechnik par l'intermédiaire d'un ordinateur. À l'aide de quelques composants provenant du jeu de construction (moteur et contacteurs), vous pourrez vous construire d'abord des dispositifs de pilotage de maquettes très simples.

Aussitôt que vous vous serez familiarisé avec le logiciel et l'interface, vous pourrez entreprendre sans problème la construction des maquettes plus compliquées du jeu de construction Bionic Robots.

Le coffret comprend un CD - ROM contenant des exemples de programme LLWin destinés aux maquettes du coffret. Pour pouvoir ouvrir ces programmes, vous avez besoin du logiciel LLWin à partir de la version 3.0. Vous pouvez soit laisser les exem-ples de programme sur le CD et les appeler depuis LLWin à l'aide de la commande Fichier – Ouvrir, soit copier le répertoire complet BIONIC_ROBOTS du CD dans le répertoire de projet de LLWin se trouvant sur le disque dur, et ouvrir ces exemples depuis le répertoire.

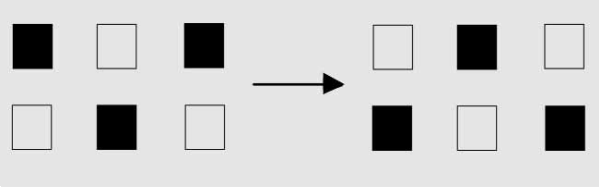
Avant de construire les maquettes, il vous faut aussi assembler quelques pièces telles que les câbles et les connecteurs. Reportez-vous à la notice de montage qui décrit ce qu'il faut faire exactement.

Voilà, tout est prêt. Vous pouvez à présent pénétrer dans le monde fascinant des robots mobiles de fischertechnik. Aussitôt que vous aurez construit la première maquette et qu'elle commencera à se déplacer de façon presque hallucinante, vous serez enthousiasmé de cette technique déjà utilisée depuis des millions d'années dans la Nature pour le déplacement des êtres vivants.

3. Marche sur 6 pattes

3.1 Mode de déplacement des insectes

Le mode de déplacement des insectes constitue un excellent modèle pour l'entraîne-ment des « animaux mécaniques hexapodes ». Lors de ce qu'on appelle l'allure en trépied, trois des six pattes quittent toujours le sol simultanément, à savoir les pattes avant et arrière d'un côté donné ainsi que la patte médiane de l'autre côté :

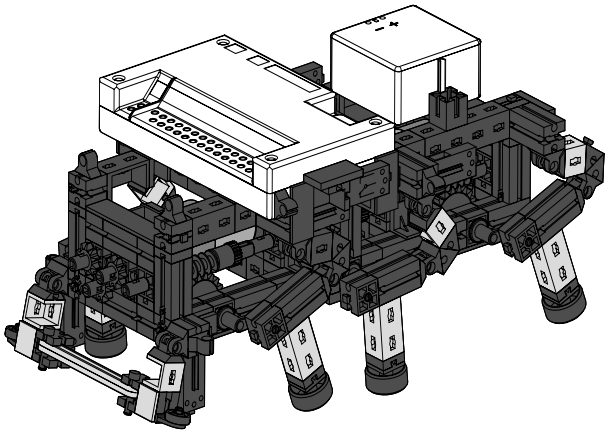


Les pattes en contact avec le sol (représentées en noir) forme un trépied stable, de sorte que la maquette est toujours en équilibre et ne bascule pas lors de la marche.

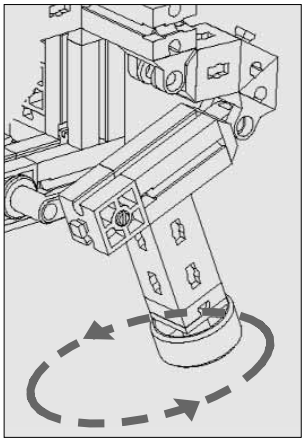
3.2 Maquette Mike

3.2.1 Conception de la maquette

Construisez à présent l'animal hexapode Mike (voir Notice de montage p. 4). Pendant la construction, chargez le paquet d'accumulateurs afin de disposer ensuite de suffisamment d'énergie pour la propulsion de la maquette.



Les pattes de la maquette sont conçues de telle façon qu'elles forment ce qu'on appelle un quadrilatère articulé. Le mode de construction du quadrilatère articulé utilisé ici s'appelle « mécanisme à manivelle et bielle oscillante ». Entraînés par une manivelle, les membres à articulations mobiles du mécanisme exécutent des mouve-ments oscillants. L'espacement des différentes articulations et la position du point d'appui (c'est l'extrémité inférieure de la patte) sont choisis de telle façon que le point d'appui décrit un mouvement elliptique lorsque la manivelle d'entraînement tourne. Il en résulte un mouvement ressemblant à un pas effectué lors de la marche.



Les 6 manivelles entraînant les pattes doivent être réglées exactement comme c'est indiqué dans la Notice de montage. Les trois pattes qui se posent simultanément sur le sol possèdent la même position de manivelle. Les manivelles des 3 pattes qui se trouvent dans l'air à ce moment-là présentent une rotation de 180° par rapport à l'au-tre groupe de manivelles. La position correcte des manivelles les unes par rapport aux autres garantit que la maquette peut effectuer les pas dans l'ordre qui convient, c'est-à-dire suivant l'allure en trépied.

Il faut bien serrer les écrous des pinces et des moyeux fixant les pignons et les vis sans fin sur les essieux pour que les manivelles ne se dérèglent pas pendant la marche.

Les côtés droit et gauche de la maquette sont chacun entraînés par un moteur (c'est nécessaire pour effectuer des virages). C'est pourquoi il faut veiller à ce que la patte médiane de l'un des côtés se trouve toujours dans la même position que les deux pattes extérieures de l'autre côté. Cette synchronisation est pilotée par logiciel au moyen des deux contacteurs E1 et E2.

Vérifiez à l'aide du diagnostic d'interface que tous les contacteurs et moteurs sont correctement raccordés. Sens de rotation des moteurs : sens de rotation antihoraire = progression vers l'avant.

3.2.2 Le premier programme

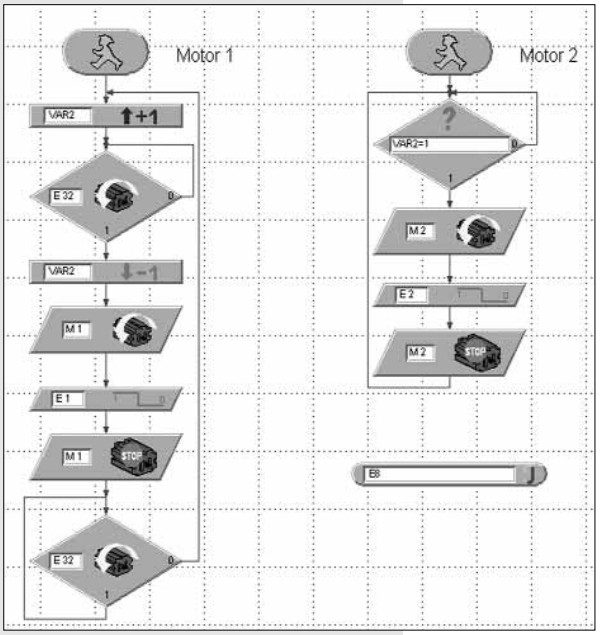
Commençons par apprendre quelque chose à Mike. Dans un premier temps, la ma-quette va seulement marcher en ligne droite. Nous nous occuperons plus tard de l'exécution de virages et de la réaction aux obstacles.

Problème 1 :

Programmez la maquette de telle façon qu'elle se déplace tout droit suivant l'allure en trépied. Utilisez les contacteurs E1 et E2 pour la synchronisation des pattes droite et gauche. Veillez à ce que les deux pattes extérieures de l'un des côtés et la patte médiane de l'autre côté se trouvent toujours dans la même position. Utilisez en outre la touche E8 comme touche de remise à zéro.

Conseils :

Programmez une séquence propre pour chaque moteur. Pilotez la séquence du moteur M2 à l'aide d'une variable VAR2. Si vous n'avez pas besoin de l'interface durant la programmation, vous devriez interrompre la connexion électrique existant entre l'Accu Set et l'interface pour économiser de l'énergie.



Solution :

Le programme permettant la marche en ligne droite a l'aspect suivant :

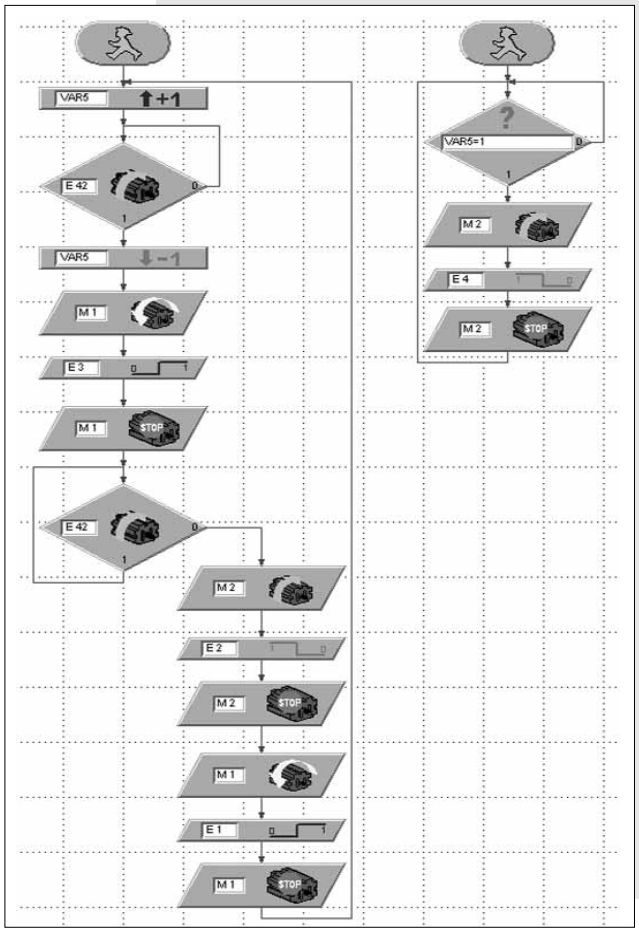
La variable VAR2 fournit l'impulsion faisant démarrer le moteur M2. Ensuite, c'est le moteur M1 qui est mis en marche. Aussitôt que le contacteur E1 est actionné, M1 s'arrête. Aussitôt que E2 est actionné, M2 s'arrête. La première séquence attend que M2 s'arrête (l'état du moteurs M2 fait l'objet d'une interrogation par l'intermédiaire de l'entrée E32; voir aussi « Interrogation de l'état du moteur » dans le manuel LLWin).

D'ailleurs, si vous n'avez pas envie d'élaborer vous-même cette séquence, vous la trouverez sous forme d'exemple de projet intitulé MIKE_TOUT DROIT.MDL sur le CD joint.
Lancez le projet. Si vous avez tout programmé correctement, la maquette s'anime et se déplace en ligne droite. Félicitations ! Vous avez fait le premier pas.

3.2.3 Rotation vers la gauche

Nous n'allons naturellement pas nous contenter de ne faire marcher Mike qu'en ligne droite. Nous allons maintenant le faire tourner sur place.

Problème 2 :
Programmez Mike de telle façon qu'il tourne vers la gauche.
Conseils :
La maquette tournera vers la gauche si M1 tourne dans le sens antihoraire et M2 tourne dans le sens horaire.
Vous pouvez bien entendu faire fonctionner la maquette de façon non synchronisée. Elle tournera tout de même, mais il y aura des situations où elle basculera vers l'avant. On peut l'éviter. À savoir avec la séquence suivante :

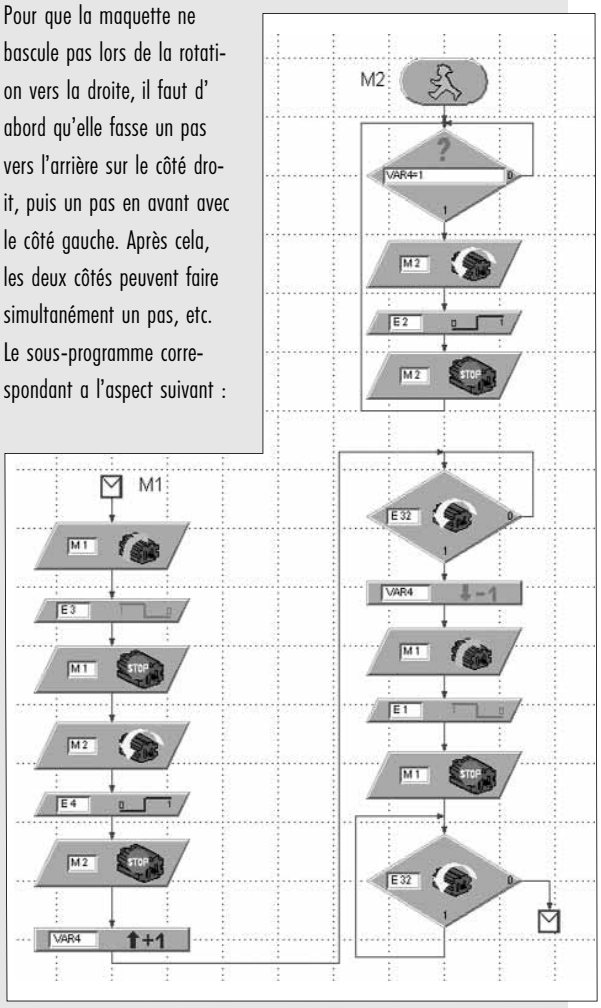


À l'aide des contacteurs E1 à E4 les côtés droit et gauche de la maquette font d'abord simultanément un pas, puis le côté gauche fait un pas, suivi par le côté droit, etc. De cette façon, la maquette ne bascule jamais vers l'avant. Essayez de le faire ! Il vous sera alors plus facile de comprendre l'ordre de ces opérations.
Cette séquence également, vous pourrez la trouver comme projet sur le CD sous la dénomination MIKE_GAUCHE.MDL.
À présent, la maquette est capable de marche en ligne droite et de tourner vers la gauche. Il ne manque plus que la marche en arrière et la rotation vers la droite. La marche en arrière fonctionne dans son principe comme la marche en avant, le sens de rotation des moteurs étant simplement inversé. La rotation vers la droite se fait elle aussi à l'inverse de la rotation vers la gauche.

3.2.4 Gauche, droite, en avant, en arrière

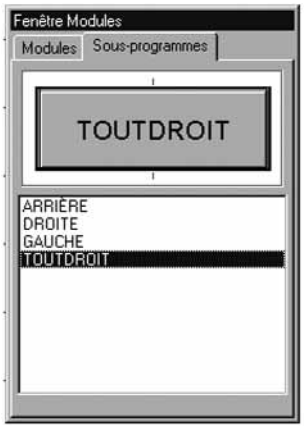
Problème 3 :
Programmez maintenant chacune des fonctions TOUTDROIT, ARRIÈRE, GAUCHE et DROITE sous forme de sous-programmes pour pouvoir les utiliser plus tard de façon très souple dans différents projets.

Conseils :
La méthode à suivre pour copier une séquence existante dans un sous-programme est décrite dans le manuel LLWin.
Utilisez dans chaque sous-programme une autre variable (VAR2 à VAR5) afin de faire démarrer la séquence destinée au moteur M2.
Pour que la maquette ne bascule pas lors de la rotation vers la droite, il faut d'abord qu'elle fasse un pas vers l'arrière sur le côté droit, puis un pas en avant avec le côté gauche. Après cela, les deux côtés peuvent faire simultanément un pas, etc.
Le sous-programme correspondant a l'aspect suivant :



Mike_Séquence_droite.bmp

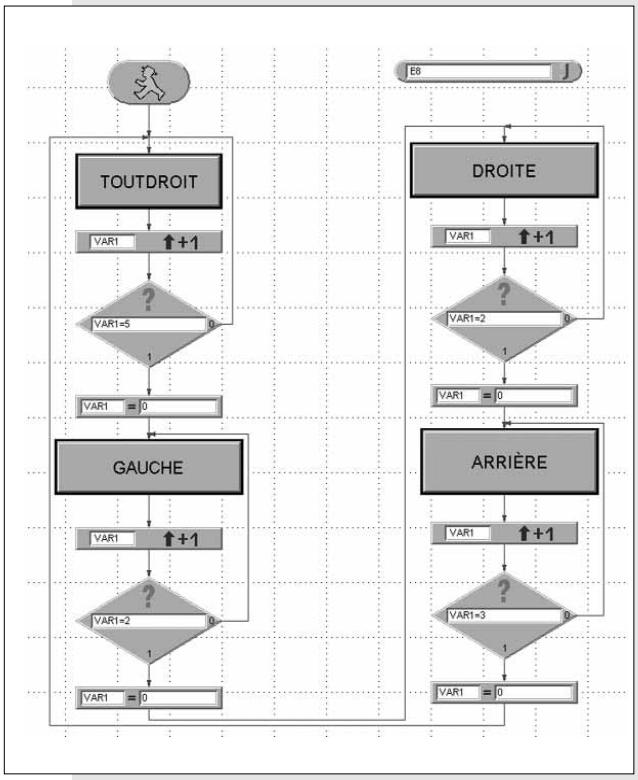
Nous n'avons pas reproduit ici les autres sous-programmes. Si vous avez des difficultés lors de la programmation d'une séquence, vous trouverez les sous-program-mes prêts à l'emploi dans le fichier MIKE_MODÈLE.MDL se trouvant sur le CD. Le programme principal de ce projet est vide. Dans la Fenêtre Modules, sous l'onglet « Sous-programmes », vous trouverez la liste des sous-programmes existants que vous pouvez insérer dans le programme principal.



Ne vous reportez cependant pas tout de suite à cet endroit pour voir comment ça fonctionne. Essayez d'abord de trouver la solution vous-même. Si vous n'y arrivez pas, vous pourrez toujours consulter cette solution.
Afin de tester tous les sous-programmes, nous allons maintenant faire danser Mike.

Problème 4 :
Programmez Mike de telle façon qu'il fasse 5 pas en avant, tourne de 2 pas vers la gauche, fasse ensuite 2 pas vers la droite, et fasse finalement 3 pas en arrière pour recommencer de nouveau depuis le début. Pour le nombre de pas à effectuer, utilisez comme compteur la variable VAR1. Utilisez E8 comme touche de remise à zéro.

Solution :



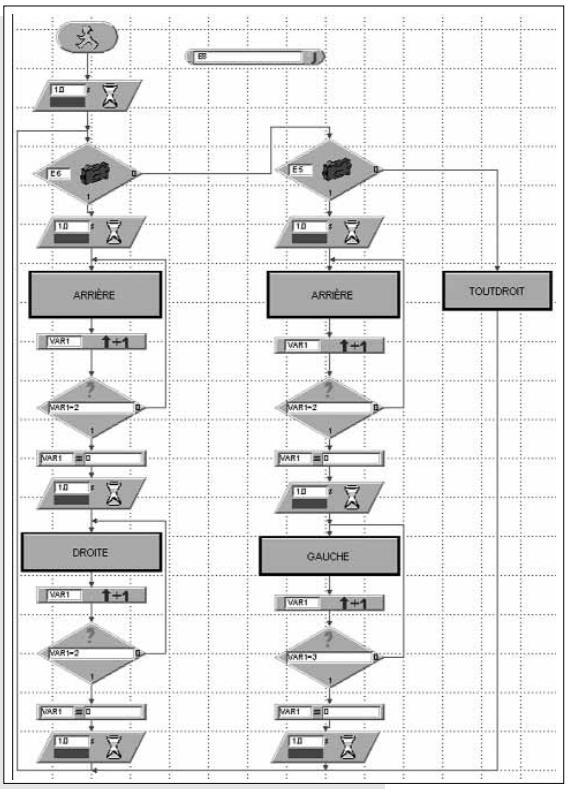
Ce projet est intitulé MIKE_DANCE.MDL.

3.2.5 Reconnaître les obstacles

Pour finir, nous voulons encore amener Mike à reconnaître les obstacles à l'aide de son pare-chocs mobile (ou appelons-ça plutôt une « antenne ») et à les éviter.

Problème 5a :
Programmez Mike de telle façon qu'il évite un obstacle capté par son antenne gauche (contacteur E6) en faisant d'abord 4 pas en arrière, puis 2 pas vers la droite. S'il rencontre un obstacle sur son antenne droite (contacteur E5), il devra faire 4 pas en arrière, puis 3 pas vers la gauche pour l'éviter.

Solution :

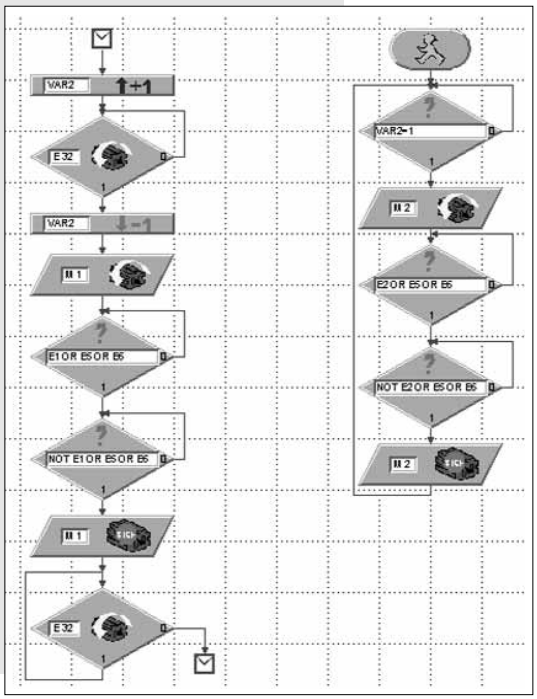


Mike se déplace d'abord constamment en ligne droite. À chaque pas, les contacteurs E5 et E6 font l'objet d'une interrogation. Si E6 est actionné, le programme fait bifurquer la procédure vers la séquence de gauche (d'abord en arrière, puis vers la droite). Si c'est E5 qui est actionné, le programme passe à la séquence se trouvant au centre (d'abord en arrière, puis vers la gauche).
Étant donné que les contacteurs E5 et E6 ne sont interrogés qu'après un pas complet, Mike réagit relativement lentement à un obstacle.

Problème 5b :
Optimisez le sous-programme TOUTDROIT de telle façon que Mike puisse réagir plus rapidement à un obstacle.

Conseil :
Utilisez, pour l'interrogation des contacteurs E1 et E2, non pas le module ARÊTE mais le module COMPARAISON. Demandez en plus si c'est E5 ou E6 qui est actionné.

Solution :

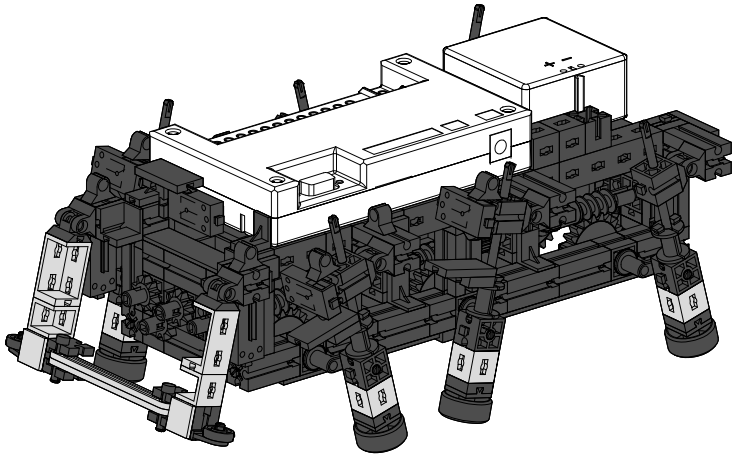


À présent, Mike devrait fonctionner de façon parfaite. Ce programme se trouve également sur le CD sous le nom MIKE_OBSTACLE.MDL. Le sous-programme amélioré se laisse aussi intégrer dans le projet MIKE_MODÈLE.MDL. Si, dans un autre pro-gramme, E5 et E6 ne sont pas interrogés, ceci ne dérange en aucune façon. Ce modèle amélioré a été stocké sous le nom MIKE_MODÈLE_OBSTACLE.MDL.

Maintenant que nous avons traité en détail le premier animal hexapode, nous allons nous intéresser à la deuxième maquette qui a également 6 pattes. Nous allons l'appeler « Jack ».

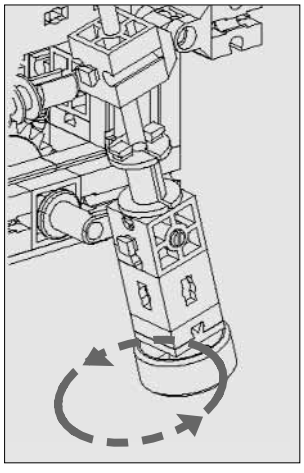
3.3 Maquette Jack

Jack fait aussi partie de l'ordre des maquettes hexapodes de fischertechnik. Cepen-dant, il se distingue considérablement de Mike par la conception de ses pattes.
Construisez maintenant la maquette comme décrit dans la Notice de montage à partir de la page 12. Les étapes 1 à 13 de la construction sont d'ailleurs identiques pour Mike et pour Jack. Il n'est donc pas nécessaire de démonter entièrement Mike avant de vous mettre à construire Jack.



3.3.1 La conception

La conception des pattes de Jack repose également sur ce qu'on appelle un quadri-latère articulé. Le mode de construction utilisé ici se nomme « mécanisme à coulisse oscillante ». La bielle est logée dans une glissière longitudinale mobile qui oscille lorsque la manivelle tourne. La courbe que décrit le point d'appui de chaque patte n'est pas aussi elliptique que sur la maquette Mike, mais plutôt circulaire.



C'est pourquoi le corps de Jack se soulève et s'abaisse davantage que celui de Mike lors de la marche. Les pas sont plus courts. En contrepartie, Jack est en mesure de franchir de petits obstacles, ce que Mike ne peut pas faire. De plus, la construction de ce mécanisme rappelle davantage celle d'une patte que ce n'est le cas avec Mike. Lorsque Jack se déplace, on dirait qu'il marche sur des échasses.
En tout cas, il se déplace lui aussi suivant l'allure en trépied des insectes. Sur cette maquette également, il est important de régler les manivelles exactement comme c'est décrit dans la Notice de montage, et de bien serrer à bloc les écrous des pinces et des moyeux.

3.3.2 La programmation

Il est tentant de penser qu'on peut utiliser pour Jack les programmes sous lesquels Mike fonctionne déjà. Essayez-le !

Problème 1 :
Faites fonctionner Jack à l'aide du programme MIKE_OBSTACLE.MDL. Que pouvez-vous constater ?

Observation :
En avant et en arrière, la maquette marche de façon irréprochable. Lors de la rotation vers la gauche ou vers la droite cependant, elle bascule vers l'avant.

Problème 2 :
Comment vous expliquez-vous ce phénomène ?

Solution :
Les deux maquettes possèdent des pattes d'une conception différente. De plus, les contacteurs E1 à E4 sont actionnés dans une position différente de chaque patte. La façon dont Mike effectue une rotation ne convient donc pas forcément pour Jack – c'est la faute à pas de chance !

Bien entendu, ça ne nous plaît pas du tout et nous voulons trouver une solution aussi rapidement que possible.

Problème 3 :
Essayez de programmer Jack de telle façon qu'il puisse reconnaître les obstacles comme Mike le fait, mais sans basculer vers l'avant lors de la rotation.

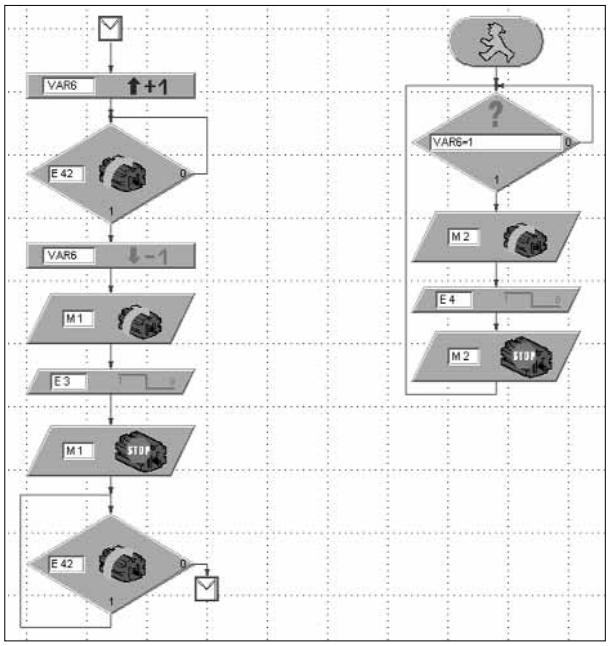
Conseils :
Sauvegardez le projet MIKE_OBSTACLE.MDL sous le nom JACK_OBSTACLE.MDL et effectuez dans ce programme les modifications nécessaires.

Lorsque Jack tourne, les deux moteurs peuvent toujours fonctionner simultanément. Il n'est pas nécessaire de les arrêter et de les faire démarrer à tour de rôle. Ce qui est décisif, c'est que les pattes se trouvent dans la position initiale correcte au début de la rotation, c'est-à-dire après la marche en arrière.

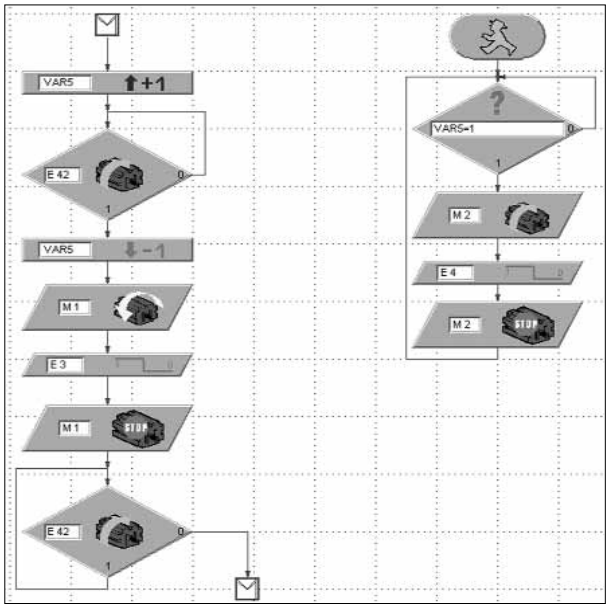
Rotation vers la gauche :
Si la maquette doit tourner vers la gauche, au début de la rotation la manivelle de la patte avant gauche doit être dirigée vers l'arrière et celle de la patte avant droite vers l'avant. C'est le cas lorsque, durant la marche en arrière, on a actionné et relâché de nouveau le contacteur E2 du côté gauche et le contacteur E1 du côté droit.

Rotation vers la droite :
Si la maquette doit tourner vers la droite, au début de la rotation la manivelle de la patte avant gauche doit être dirigée vers l'avant et celle de la patte avant droite vers l'arrière. C'est le cas aussitôt que, durant la marche en arrière, on a actionné et relâché de nouveau le contacteur E4 du côté gauche et le contacteur E3 du côté droit.

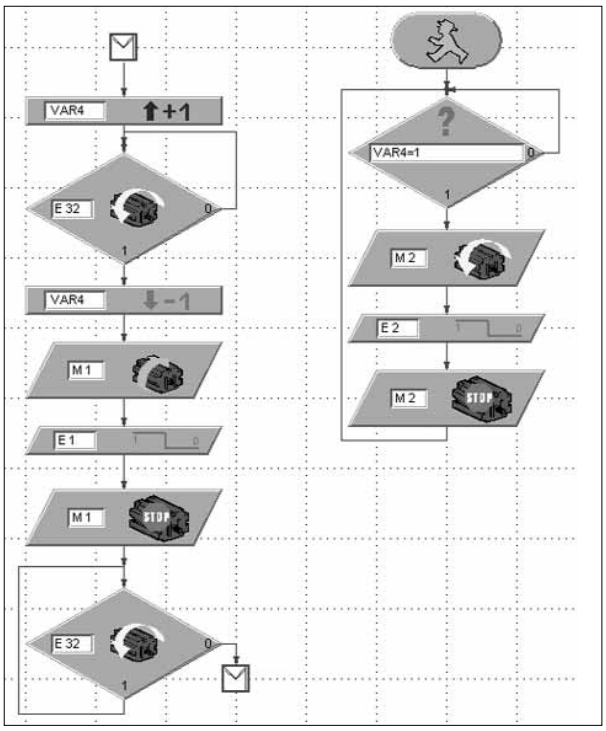
C'est assez compliqué, pas vrai ? Ne vous faites pas de souci, nous allons résoudre le problème en un clin d'œil :
Pour le déplacement vers l'arrière, il faut utiliser deux sous-programmes différents. Si la maquette doit tourner vers la gauche, synchronisez les pas de la marche en arrière à l'aide des contacteurs E1 et E2. Ceci correspond au sous-programme ARRIÈRE provenant du projet MIKE_OBSTACLE.MDL.
Si la maquette doit tourner vers la droite, on synchronise les pas à faire en arrière à l'aide des contacteurs E3 et E4.
Vous allez donc renommer le sous-programme ARRIÈRE à l'aide de la commande SOUS-PROGRAMME - RENOMMER en ARRIÈRE_G pour la gauche. Ensuite, à l'aide de la commande SOUS-PROGRAMME - COPIER, vous copiez celui-ci sous la forme d'un deuxième sous-programme ARRIÈRE_D pour la droite. Dans ce sous-programme, vous modifiez la désignation des contacteurs pour obtenir la synchro-nisation par E3 et E4. N'oubliez pas non plus, dans ARRIÈRE_D, d'utiliser une nouvelle variable VAR6 pour cette synchronisation, sinon c'est le chaos complet. ARRIÈRE_D a alors l'aspect suivant :



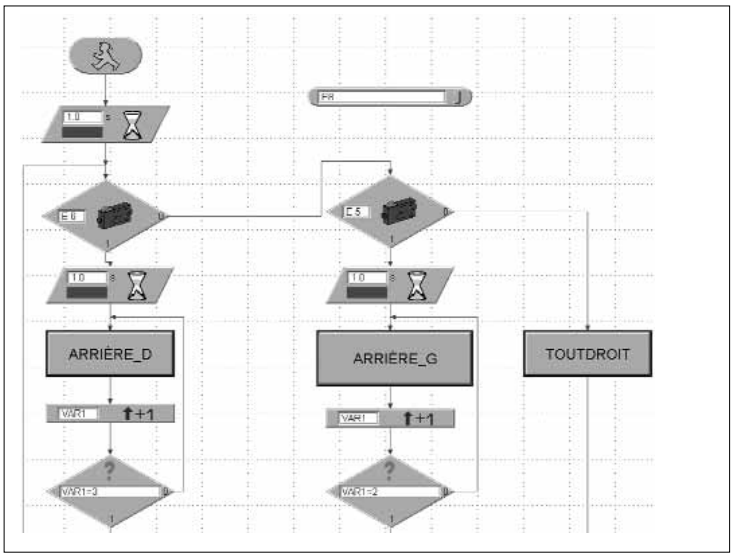
À présent, il faut encore modifier les sous-programmes pour la rotation elle-même, de façon que les deux moteurs fonctionnent toujours simultanément. Le sous-programme GAUCHE se compose des modules suivants :



Comme vous le voyez, on peut se passer de quelques modules par rapport au sous-programme du projet MIKE_OBSTACLE.MDL.
Le sous-programme pour la rotation vers la droite a le même aspect, les moteurs tournant simplement dans l'autre sens. En outre, on utilise les contacteurs E1 et E2 pour effectuer la synchronisation des deux moteurs :



Pour finir, remplacez dans le programme principal le sous-programme ARRIÈRE_G par ARRIÈRE_D à l'aiguillage conduisant à l'évitement vers la droite :



Le reste du programme principal demeure inchangé.

Ouf ! Si vous n'avez fait aucune erreur nulle part, Jack devrait à présent marcher sans basculer lors des rotations. Si quelque chose ne fonctionne pas et que vous ne savez absolument pas pourquoi, n'en faites pas un drame, c'était un vrai casse-tête à résoudre. De toute façon, il vous est possible d'appeler tout simplement le projet terminé JACK_OBSTACLE.MDL du CD joint et de faire fonctionner la maquette avec ce programme.

Au cas où vous auriez réussi tout seul à résoudre le problème, vous pouvez être fier de vous car vous faites désormais partie des programmeurs professionnels.

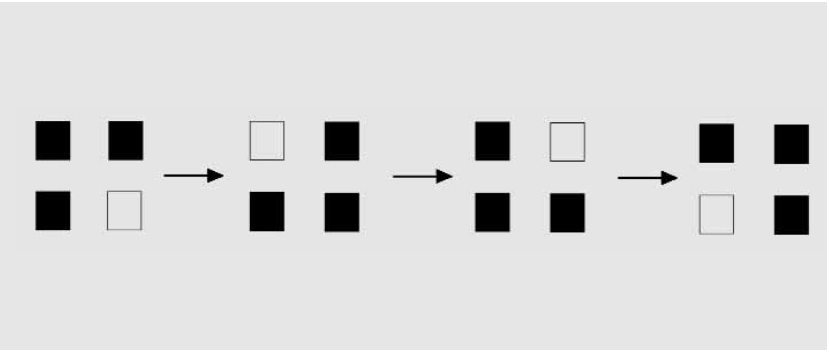
F

4. Marche sur 4 pattes

4.1 Modes de déplacement des mammifères

Pour construire un robot mobile à 4 pattes, nous allons de nouveau prendre la Nature comme modèle et regarder quels modes de locomotion les mammifères utilisent pour se déplacer.

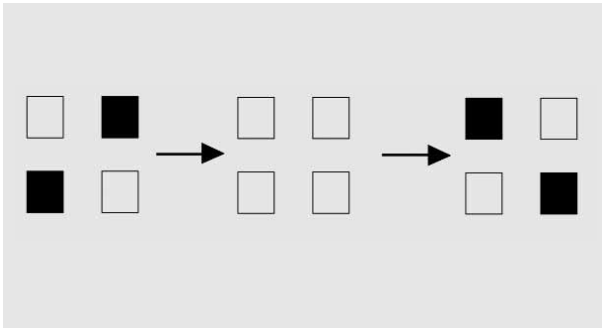
L'allure la plus lente et la plus sûre est ce qu'on appelle le pas. Une des pattes cher-che une nouvelle position tandis que le corps de l'animal s'appuie sur trois pattes. Les animaux se déplacent alors suivant l'allure croisée dans l'ordre suivant : ils déplacent vers l'avant la patte avant droite, patte arrière gauche, patte avant gauche, patte arrière droite.



Les surfaces noires représentent les pattes en contact avec le sol, les blanches la patte levée.

Si nous voulons utiliser cette allure sur un robot mobile, il nous faut réfléchir aux points suivants :
Imaginez que vous sciez l'une des pattes d'une table à 4 pattes. Qu'est ce qu'il va se passer ? C'est juste : la table basculera. Les trois pattes ne forment donc plus un trépied stable comme c'était le cas chez les hexapodes. Ceci rend plus difficile la conception d'un robot quadrupède.

Plus les mammifères se déplacent rapidement, plus leur allure devient instable. Exa-minons par exemple brièvement l'allure appelé « trot ».
Au trot, deux pattes situées en diagonale se soulè-vent de façon synchrone. Cependant, avant qu'elles repren-nent contact avec le sol, les deux autres pattes ont déjà bondi. Ceci signifie que, par intermittence, l'animal perd tout contact avec le sol.

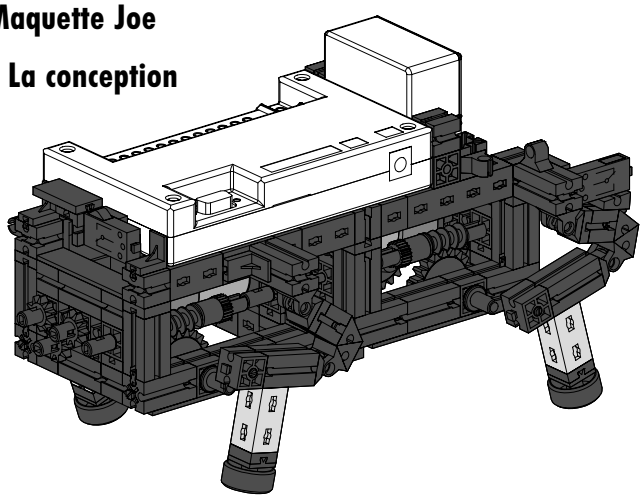


Vous pouvez vous imaginer qu'une allure au cours de laquelle l'animal perd régulièrement le contact avec le sol ne convient pas vraiment à une maquette de fischertechnik devant porter l'Intelligent Interface et l'Accu Set.

Essayons donc avec l'allure appelée « pas ».

4.2 Maquette Joe

4.2.1 La conception



Assemblez la maquette comme c'est décrit dans la Notice de montage à partir de la page 20.

La conception des pattes est la même que sur Mike. Mais la position des manivelles actionnant les pattes doit être complètement différente sur Joe. Les manivelles sont désaxées de 90° les unes par rapport aux autres. Vous devrez les régler exactement comme c'est décrit dans la Notice de montage. La synchronisation des côtés droit et gauche se fait de nouveau au moyen des deux contacteurs E1 et E2. Nous obtenons ainsi la séquence de marche nécessaire.
Pour que la maquette ne bascule pas complètement aussitôt qu'une patte se lève, il faut que son centre de gravité se trouve dans une position telle que la maquette bascule au bon moment et soit rattrapée par la patte qui vient juste d'être levée.

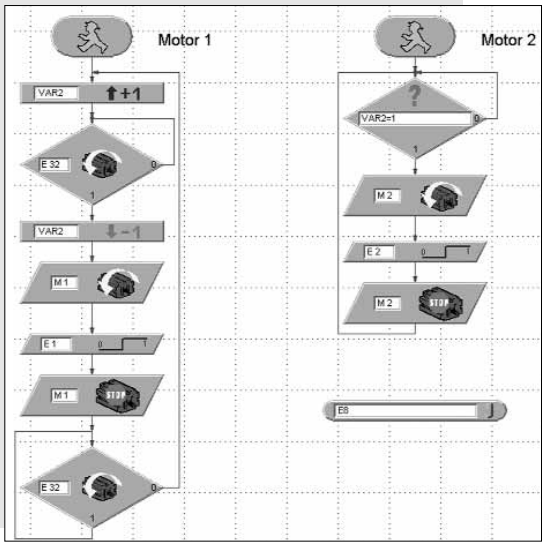
4.2.2 La programmation

Avec cette maquette, nous allons nous contenter de la marche en ligne droite.

Problème 1 :
Programmez Joe de telle façon qu'il se déplace vers l'avant suivant l'allure appelée « pas ».

Conseils :
Utilisez pour chaque moteur une séquence séparée et synchronisez les deux côtés à l'aide des contacteurs E1 et E2. Utilisez E8 comme touche de remise à zéro.

Solution :



Dans ce programme nous utilisons l'arête de l'impulsion 0-1 des contacteurs pour assurer la synchronisation. À l'instant où on actionne les contacteurs, les manivelles de toutes les 4 pattes ont la position correcte les unes par rapport aux autres. Nous appelons ce projet JOE.MDL.
Comme vous le voyez, Joe se déplace avec beaucoup plus de lourdeur que Mike et Jack. En raison du report de charge nécessaire, le corps titube assez fortement et sa manière de marcher est de loin moins élégante que celle des maquettes à six pattes.
Au cas où vous en auriez envie, vous pouvez apprendre à cette maquette à effectuer des virages – essayez tout simplement pour voir si vous y parvenez. Bonne chance !

5. Marche sur deux pieds

5.1 Les bipèdes

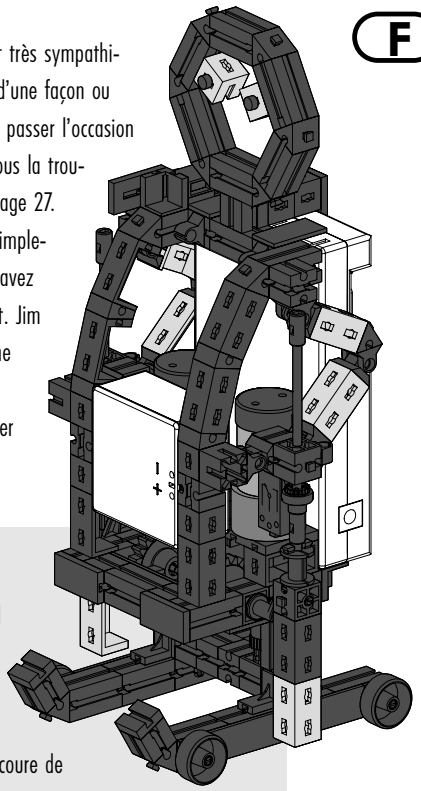
La marche sur deux pattes n'est pas apparue dans l'ordre des mammifères, mais est pratiquée également par quelques espèces de reptiles. Varans, iguanes, agames et lézards coureurs n'utilisent que leurs pattes arrière pour la fuite. Ils peuvent ainsi faire de très grandes enjambées et devenir extrêmement rapides. Pour cela, ils ont besoin de puissantes pattes arrière, d'une longue queue leur servant de balancier, et d'un terrain plat.
Les oiseaux font également partie des bipèdes. Parmi les oiseaux les plus rapides, il faut citer l'autruche. Elle atteint des vitesses de croisière de 60 km/h.
Le bipède le plus perfectionné est l'Homme. Sa démarche entièrement verticale exige l'extension de l'articulation de la hanche. Cette extension est assurée par le grand muscle fessier. De plus, il peut « verrouiller » ses pieds dans l'articulation du genou et les immobiliser ainsi dans une attitude consommant peu d'énergie.
La locomotion sur deux pieds représente le mode de déplacement le plus difficile qui soit car elle exige un sens de l'équilibre développé en plus des données anatomi-ques décrites précédemment. À nous les Hommes, la marche sur deux pieds nous semble facile et naturelle. Cependant, lorsque nous nous représentons que, quand nous levons un pied, notre corps tout entier ne repose que sur une jambe et doit ainsi rester en équilibre, nous nous rendons compte que c'est précisément le main-tien de l'équilibre qui rend ce mode de déplacement si compliqué. Même un nouveau-né n'est pas immédiatement en mesure de marcher sur deux pieds. Il lui faut d'abord pendant un certain temps ramper « à quatre pattes » avant de pouvoir se redresser et « apprendre à marcher ».

À l'université Waseda de Tokyo on a déjà mis au point des robots bipèdes capables de se déplacer à l'aide de nombreuses articulations, différents capteurs, des camé-ras et des microprocesseurs performants tout en conservant leur équilibre par report de leur poids.

Cependant, pour notre jeu de construction Bionic Robots, ceci serait trop onéreux et trop compliqué. Nous avons vu qu'avec la marche sur quatre pattes simulée avec une maquette de fischertechnik, nous butons déjà sur les limites de nos capacités.

5.2 Maquette Jim

Cependant, afin de ne pas nous limiter à des considérations théoriques dans ce chapitre, nous nous sommes tout de même permis de concevoir au moins un skieur bipède que nous appelons Jim. Il a certes peu de choses en commun avec un coureur bipède, mais il est très sympathique et fait de son mieux pour avancer d'une façon ou d'une autre. Vous ne devriez pas laisser passer l'occasion de vous divertir avec cette maquette. Vous la trouverez dans la Notice de montage à la page 27.
Comme programme, vous pouvez tout simplement utiliser le projet JOE.MDL. Vous n'avez même pas à y modifier quoi que ce soit. Jim fonctionne également avec ce programme et avance lentement avec ses bâtons.
Nous voudrions tout de même vous poser encore un problème :



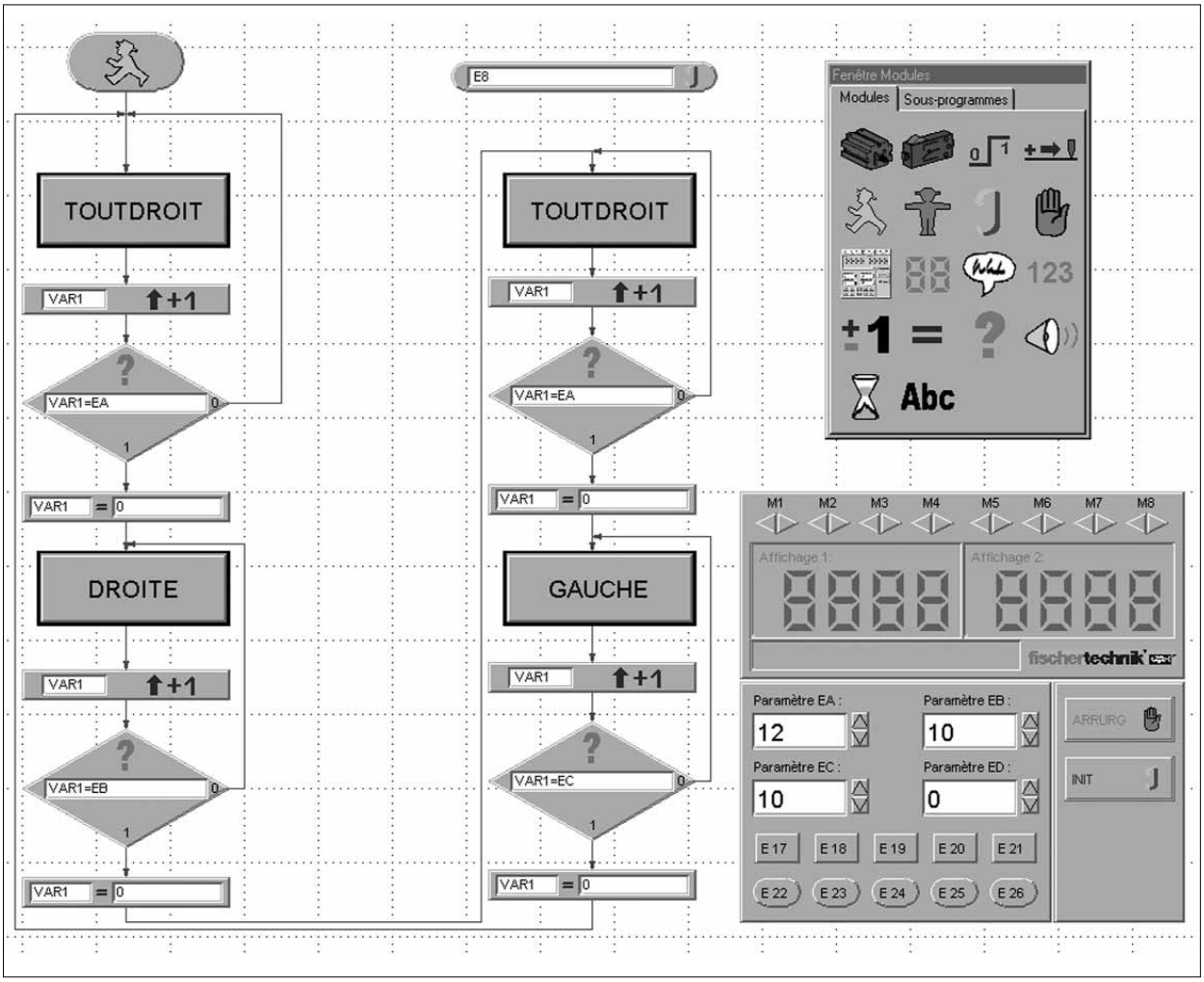
Problème 1 :
Programmez Jim de telle façon qu'il avance d'environ 50 cm, puis tourne de 180° vers la droite, revienne sur ses pas (vers l'avant), tourne de 180° vers la gauche, parcoure de nouveau la même distance, etc. Pour le nombre de pas en ligne droite utilisez le paramètre terminal EA, pour le nombre de pas vers la gauche EB et pour le nombre de pas vers ladroite EC. Utilisez de nouveau E8 comme touche de remise à zéro.

Conseils :
Enregistrez le projet JOE.MDL sous la dénomination JIM.MDL. Dans ce fichier, faites du programme principal un sous-programme « Toutdroit » (sélectionnez les modu-les et couper-les, élaborez un nouveau sous-programme par l'intermédiaire de ÉDITER – SOUS-PROGRAMME, insérez les modules, complétez SUBIN et SUBOUT, voir aussi le manuel LLWin).

À partir de ce sous-programme, élaborez au moyen de la commande SOUS-PROGRAMME – COPIER les sous-programmes nécessaires GAUCHE et DROITE. Modifiez dans ces fichiers le sens de rotation des moteurs et l'interrogation du sens de rotation des moteurs de façon correspondante, et utilisez pour chaque sous-programme une autre variable de pilotage pour le moteur 2.

Après cela, vous programmerez le programme principal de la même façon que vous l'avez fait pour MIKE_DANCE.MDL. Simplement, vous utiliserez en plus les para-mètres terminaux réglables EA à EC pour le nombre de pas à effectuer. Il faudra que vous testiez de combien de pas Jim a besoin pour tourner de 180° et pour parcourir une distance d'environ cinquante centimètres vers l'avant.

Solution :
Nous reproduisons ci-dessous le programme principal. Si nécessaire, vous pourrez de nouveau consulter les sous-programmes directement sur l'écran. Sur notre CD également, le projet s'appelle JIM.MDL.



Dans la foulée, nous avons encore complété le projet du sous-programme ARRIÈRE même si on en n'a pas directement besoin ici. Mais vous voudrez certainement que Jim suive d'autres chemins. Et peut-être qu'il lui faudra alors marcher en arrière.

6. Résumé

Lors de votre excursion dans le monde des Robots Bioniques de fischer-technik, vous avez certainement constaté qu'il n'a pas toujours été très simple de faire marcher les quatre gaillards. Il est précisément plus difficile de se déplacer sur des pattes que de rouler sur des roues. C'est surtout la programmation de la synchronisa-tion entre les côtés droit et gauche lors de la rotation vers la gauche ou vers la droite qui exige un peu de concentration. Mais pour ceux qui trouvent plus de plaisir dans la construc-tion des maquettes, nous avons de toute façon gravé tous les programmes prêts à l'emploi sur le CD, de sorte que chacun peut construire et faire fonctionner ces maquettes.

Si vous faites partie des programmeurs professionnels, vous avez certaine-ment beaucoup d'autres idées sur les tâches qu'on peut encore programmer pour Mike, Jack, Joe et Jim, que ce soit avec des capteurs supplémentaires pour qu'ils ne tombent pas de la table, ou pour qu'ils s'y retrouvent dans un labyrinthe.

À l'aide d'autres composants, vous pourrez aussi les équiper d'une tête, d'une queue ou d'une trompe. Votre fantaisie n'a pas de limite. Donnez libre cours à votre imagination !

NL

1. Bionic – de natuur als voorbeeld

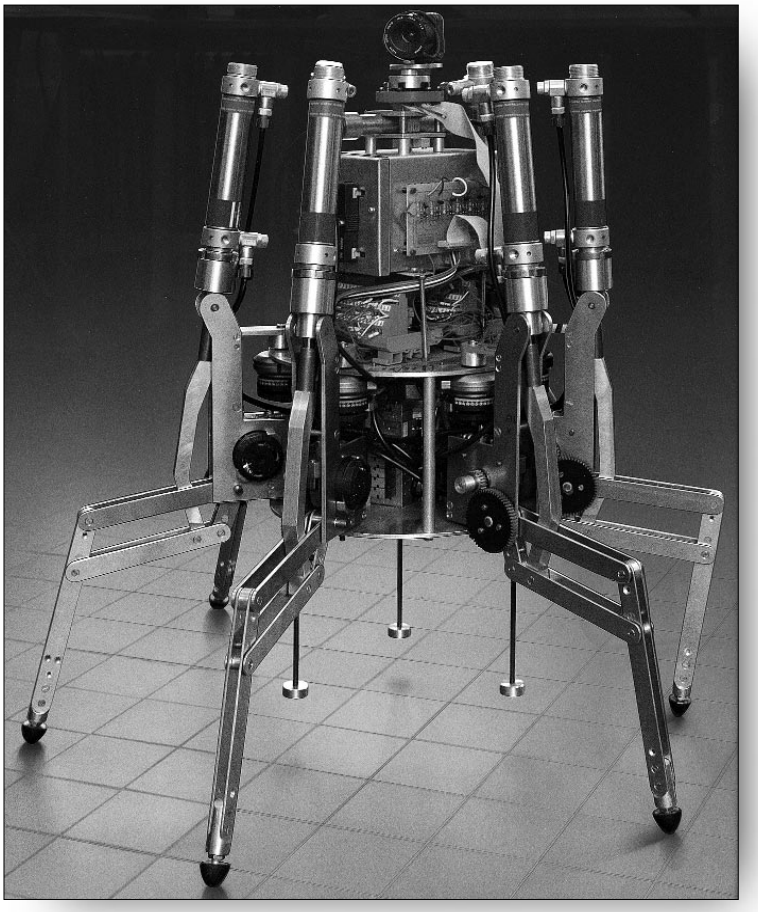
Het begrip „bionic" bestaat uit de twee begrippen biologie en techniek. Deze tak van de wetenschap probeert zich voortdurend aan de natuur te oriënteren om technische problemen op te lossen.
Om zich steeds verder, sneller en effectiever voort te bewegen dan de natuur toestond, heeft de mens steeds weer machines uitgevonden die hier overeenkomstig de gestelde eisen op verschillende manieren voor zorgen. Voertuigen rijden op wielen. Op moeilijk terrein, waarop voertuigen op wielen zich niet kunnen verplaatsen, worden rupsvoertuigen ingezet. Schepen zwemmen op het water of kunnen duiken. Voor een aantal manieren van voortbeweging dient ook de natuur als voorbeeld. Zo lijkt een vliegtuig op een zwevende vogel.

Sinds een aantal jaren houden wetenschappers zich met een andere manier van voortbeweging bezig die in de natuur veelvuldig voorkomt, namelijk het lopen. Ze ontwikkelden robots die zich op benen kunnen voortbewegen. Deze loopmachines zouden overal kunnen worden ingezet waar voertuigen op wielen of rupsvoertuigen nauwelijks van nut zijn, bijv. op extreem oneffen of zacht terrein, om over hindernissen te klimmen, trappen op te lopen, sloten te passeren of om te worden ingezet op moeilijk toegankelijke en gevaarlijke plaatsen in kerncentrales, mijnen of tijdens reddingsacties.

De eerste serieuze pogingen om loopmachines te ontwikkelen werden in 1967 ondernomen aan een universiteit in Tokio. Voor het eerst oriënteerden de wetenschappers zich aan de gang van de mens in plaats van aan insecten. De voortdurende verdere ontwikkeling van deze experimenten leverde in 1985 de eerste tweebenige loopmachine op. Inmiddels beschikken deze robots over ruim 50 vrijheidsgraden en een groot aantal microprocessors. Met behulp van een camera kunnen ze bijv. noten lezen en orgel spelen. Je kunt zelfs een gesprek met ze voeren.

Een voorbeeld van een zesbenige looprobot is "Achille", een elektropneumatische looprobot die aan de Koninklijke Militaire Academie in Brussel is ontwikkeld.
Uitgerust met een camera boven en aan de zes benen moet deze robot mechanisch op hoger en lager gelegen hindernissen (voorwerpen of kuilen) kunnen reageren.

Nu heeft fischertechnik zich ook met dit spannende onderwerp bezig gehouden en lopende robots geconstrueerd die vervolgens met de Intelligent Interface en de software LLWin „tot leven" worden gewekt.



2. Voorwaarden en voorbereiding

Om de modellen van het computing-bouwpakket „Bionic Robots" te kunnen bouwen, heb je behalve het bouwpakket nog de onderstaande artikelen nodig:

Intelligent Interface, art.-nr. 30402
Software LLWin (vanaf versie 3.0), art.-nr. 30407
Stroomvoorziening accuset, art.-nr. 34969

Als je nog nooit met de software LLWin en de interface hebt gewerkt, kun je beter eerst het handboek van de software LLWin doorlezen. Daarin staat beschreven hoe je de software installeert en de interface aansluit. Bovendien is dit een prima manier om alvast een beetje ervaring op te doen hoe je modellen van fischertechnik via de PC bestuurt. Met een paar bouwcomponenten uit het bouwpakket (motor en knoppen) kun je om te beginnen eenvoudige modelbesturingen opbouwen.

Zodra je vertrouwd bent met de software en de interface, kun je je beslist aan de moeilijkere modellen Bionic Robots wagen.

In het bouwpakket zit tevens een CD-ROM met LLWin-voorbeeldprogramma's voor de modellen van het bouwpakket. Om deze programma's te kunnen openen, heb je de software LLWin vanaf versie 3.0 nodig. Je kunt de voorbeeldprogramma's op de CD laten en vanuit LLWin met het commando Bestand– Openen oproepen of de complete map BIONIC_ROBOTS van de CD in de projectdirectory van LLWin op de harde schijf kopiëren en de voorbeelden vanaf deze locatie openen.

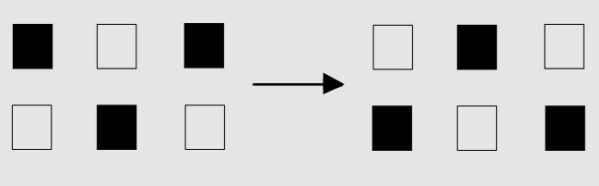
Voordat je de modellen bouwt, moet je ook nog een aantal onderdelen monteren, zoals het snoer en de stekker. Hoe je dat precies moet doen staat in de bouwhandleiding beschreven.

Zo, nu is het zover. Welkom in de fascinerende wereld van de lopende fischertechnik-robots. Zodra je het eerste model af hebt en het bijna als een spook begint te bewegen, zul je versteld staan van deze techniek die in de natuur al miljoenen jaren wordt toegepast voor de voortbeweging.

3. Lopen op zes benen

3.1 Gang van de insecten

De gang van insecten is uitermate geschikt als voorbeeld voor de aandrijving van „mechanische zesbeners". Bij de zgn. drievoetsgang worden steeds drie van de zes benen tegelijkertijd van de grond getild, het voorste en achterste been van de ene kant samen met het middelste been van de andere kant:

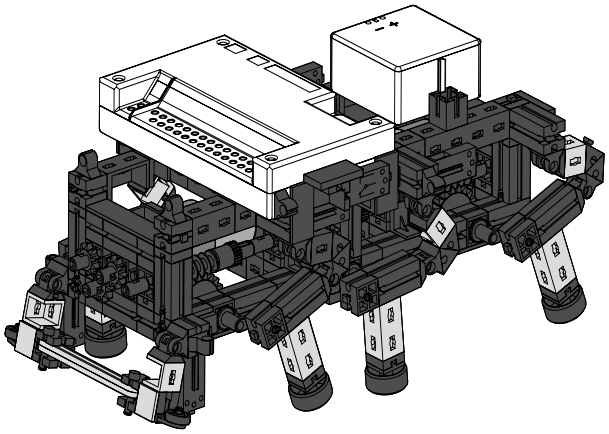


De benen die op de grond staan (zwarte blokjes) vormen een stabiele driepoot, zodat het model steeds stevig staat en niet omvalt tijdens het lopen.

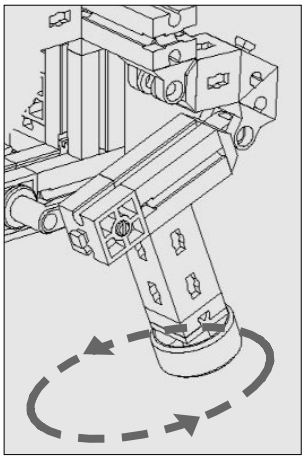
3.2 Model Mike

3.2.1 De constructie

Zet nu het zesbenige model Mike (zie bouwhandleiding p. 4) in elkaar. Laad tijdens het bouwen het accupack op, zodat je later voldoende energie hebt voor de aandrijving van het model.



De benen van het model zijn zodanig geconstrueerd dat ze een overbrenging via vier gewrichten vormen. De bouwvorm van het hier gebruikte viervoudige gewricht wordt ook wel kruk- en zwenkdrijfwerk genoemd. De beweegbaar gelagerde delen van de aandrijving voeren, aangedreven door een kruk, slingerende bewegingen uit. De afstanden tussen de afzonderlijke gewrichten en de stand van het voetpunt (het onderste deel van het been) zijn zo gekozen dat het voetpunt een elliptische beweging maakt als de aandrijfkruk draait. Hierdoor ontstaat een beweging die lijkt op een stap tijdens het lopen. De zes krukken die de benen aandrijven moeten precies volgens de bouwhandleiding worden afgesteld. De drie benen die tegelijkertijd op de grond neerkomen, hebben dezelfde krukstand. De krukken van de drie benen, die op dit moment in de lucht staan, zijn daarvoor 180° gedraaid. De juiste stand van de krukken ten opzichte van elkaar zorgt ervoor dat het model op de juiste manier, namelijk in de drievoetsgang, kan lopen.



De moeren waarmee de wormen en tandwielen op de assen worden gefixeerd, moeten goed worden aangehaald, zodat de krukken tijdens het lopen op hun plaats blijven.

De rechter- en linkerkant van het model worden elk door een afzonderlijke motor aangedreven (dit is noodzakelijk om bochten te kunnen maken). Om die reden is het van belang dat het middelste been van de ene kant zich altijd in dezelfde positie bevindt als de twee buitenste benen aan de andere kant. Deze synchronisatie wordt gestuurd door de software via de knoppen E1 en E2.

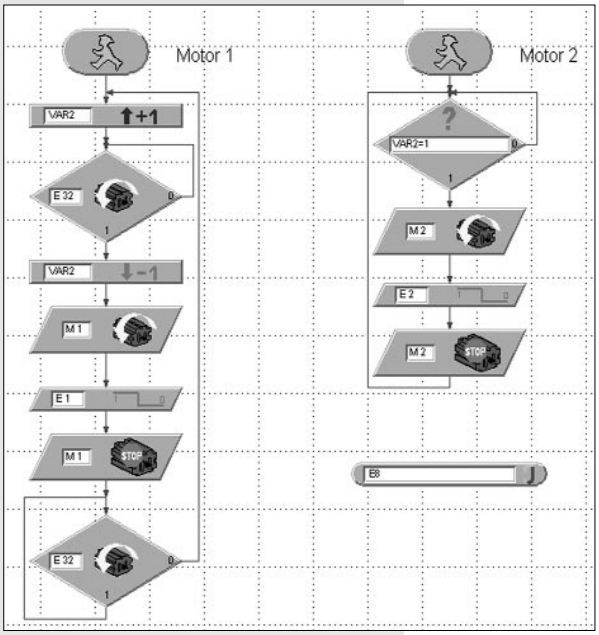
Test met behulp van de interfacediagnose (Check Interface) of alle knoppen en motoren juist zijn aangesloten. Draairichting van de motoren: linkse draairichting = vooruit.

3.2.2 Het model programmeren

Nu gaan we Mike iets aanleren. Om te beginnen is het voldoende als het model vooruit loopt. Bochten maken en reageren op hindernissen komt later aan de orde.

Opgave 1:
Programmeer het model zodanig dat het in de drievoetsgang recht vooruit loopt. Gebruik de knoppen E1 en E2 om de benen aan de linker- en rechterkant te synchroniseren. Let erop dat de twee buitenste benen van de ene kant altijd in dezelfde stand staan als het middelste been van de andere kant. Gebruik bovendien knop E8 als resetknop.

Tips:
Programmeer voor iedere motor een eigen afloop. Stuur de afloop van motor M2 met behulp van een variabele VAR2.
Als je tijdens het programmeren geen gebruik maakt van de interface, kun je de stroomverbinding tussen het accupack en de interface onderbreken om energie te sparen.



Oplossing:
Het programma voor recht vooruitlopen ziet er als volgt uit:

NL

De variabele VAR2 geeft de impuls dat motor M2 start. Daarna wordt motor M1 gestart. Zodra schakelaar E1 wordt ingedrukt, stopt M1. Zodra E2 wordt ingedrukt, stopt M2. De eerste afloop wacht tot M2 is gestopt (toestand van de motor M2 wordt via E32 opgevraagd; zie ook „Checking Motor Conditions" in het handboek LLWin).

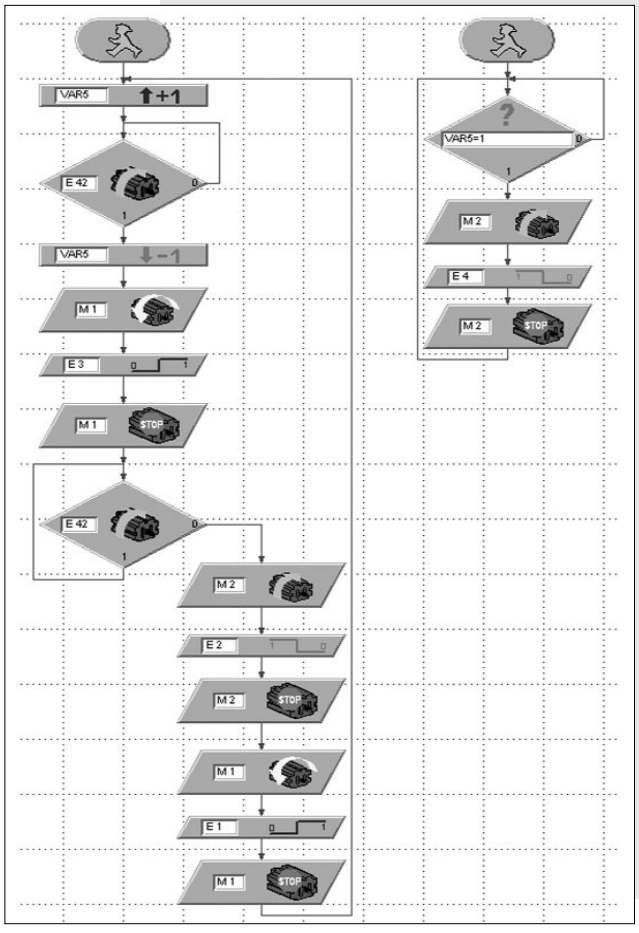
Wie trouwens niet zoveel moeite wil doen om deze afloop zelf te maken, vindt hem als voorbeeldproject MIKE_VOORUIT.MDL op de bijgevoegde CD. Start het project op. Als je alles goed hebt geprogrammeerd, komt er beweging in het model en loopt het vooruit. Van harte gefeliciteerd! De eerste stap is gezet.

3.2.3 Linksom draaien

We zijn natuurlijk nog niet tevreden met het feit dat Mike alleen maar vooruit loopt. Nu willen we dat hij zich op de plaats omdraait.

Opgave 2:
Programmeer Mike zodanig dat hij linksom draait.

Tips:
Het model draait linksom als M1 linksom en M2 rechtsom draait.
Je kunt het model natuurlijk gebruiken zonder dat het is gesynchroniseerd. Het kan dan weliswaar omdraaien, maar er zijn bepaalde posities waarin het model voorover valt. Dat kun je voorkomen.
En wel met de volgende stappen:



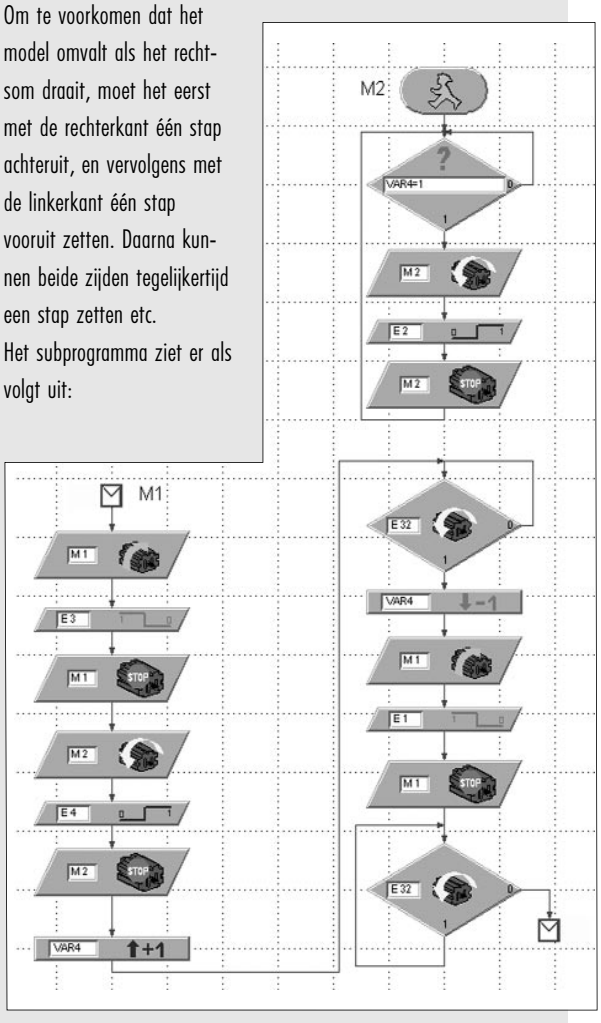
Met behulp van de schakelaars E1-E4 bewegen de linker- en rechterkant van het model eerst één stap tegelijk, vervolgens zet de linkerkant één stap, daarna de rechterkant etc. Op die manier valt het model nooit voorover. Probeer het eens! Zo is ook de volgorde gemakkelijker te begrijpen.
Ook deze afloop kun je als project MIKE_LINKS.MDL op de CD vinden.

Nu kan het model vooruit lopen en linksom draaien. Wat er nog ontbreekt is achteruit lopen en rechtsom draaien. Het achteruit lopen werkt in principe als het vooruit lopen, maar dan met de omgekeerde draairichting van de motor. Het rechtsom draaien is in feite het omgekeerde principe van het linksom draaien.

3.2.4 Links, rechts, vooruit, achteruit

Opgave 3:
Programmeer nu de functies VOORUIT, ACHTERUIT, LINKS en RECHTS als subprogramma, zodat je ze later in verschillende projecten flexibel kunt inzetten.

Tips:
Hoe je een bestaande afloop in een subprogramma kopieert, staat beschreven in het LLWin-handboek.
Gebruik in ieder subprogramma een andere variabele (VAR2-VAR5) om de afloop voor motor M2 te starten.
Om te voorkomen dat het model omvalt als het rechtsom draait, moet het eerst met de rechterkant één stap achteruit, en vervolgens met de linkerkant één stap vooruit zetten. Daarna kunnen beide zijden tegelijkertijd een stap zetten etc.
Het subprogramma ziet er als volgt uit:



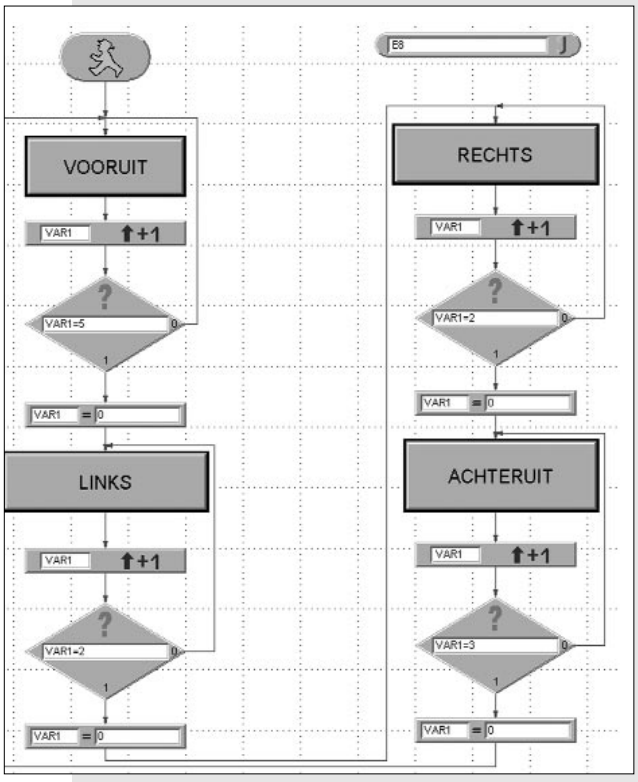
fischertechnik®

De andere subprogramma's hebben we hier niet afgedrukt. Als het programmeren van een afloop moeilijkheden oplevert, vind je de subprogramma's kant en klaar in het bestand MIKE_VOORBEELD.MDL op de CD. Het hoofdprogramma van dit project is leeg. In het bouwvenster onder de indexkaart „Subprogramma's" vind je een lijst met de beschikbare subprogramma's die je in het hoofdprogramma kunt invoegen.

Maar kijk niet meteen na hoe het moet. Probeer eerst zelf achter de oplossing te komen. Als het niet lukt, kun je het immers nog altijd opzoeken. Om alle subprogramma's eens te testen willen we Mike nu laten dansen.

Opgave 4:
Programmeer Mike zodanig dat hij 5 stappen naar voren zet, twee stappen naar links draait, dan twee stappen naar rechts, vervolgens 3 stappen achteruit zet en daarna opnieuw begint. Gebruik als telvariabele voor het aantal stappen de variabele Var1. Gebruik E8 als resetknop.

Oplossing:



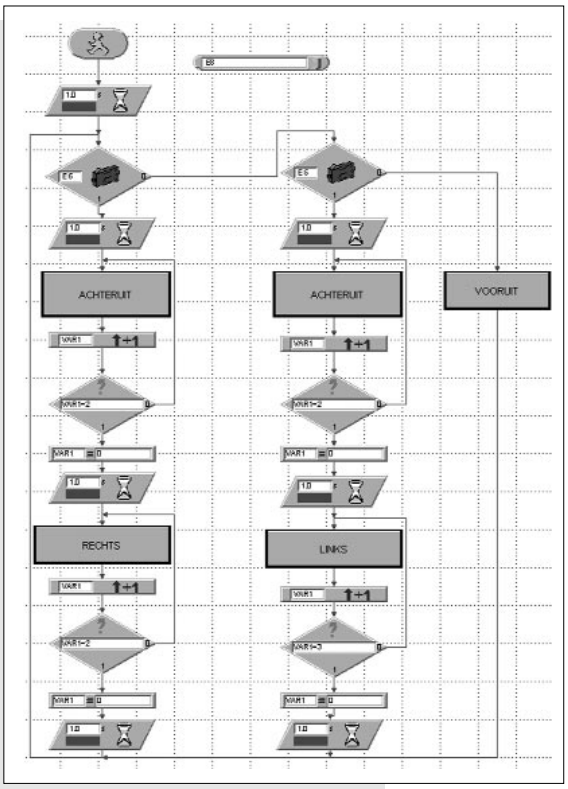
Dit project heet MIKE_DANS.MDL.

3.2.5 Hindernissen herkennen

Ten slotte willen we Mike nog zover krijgen dat hij met zijn beweegbare stoterstang (of liever „voeler") hindernissen herkent en uitwijkt.

Opgave 5a:
Programmeer Mike zodanig dat hij bij een hindernis aan zijn linkervoeler (knop E6) eerst 4 stappen achteruit en dan 2 stappen naar rechts uitwijkt. Als er sprake is van een hindernis aan zijn rechtervoeler (knop E5) moet hij 4 stappen achteruit en vervolgens 3 stappen naar links uitwijken.

Oplossing:



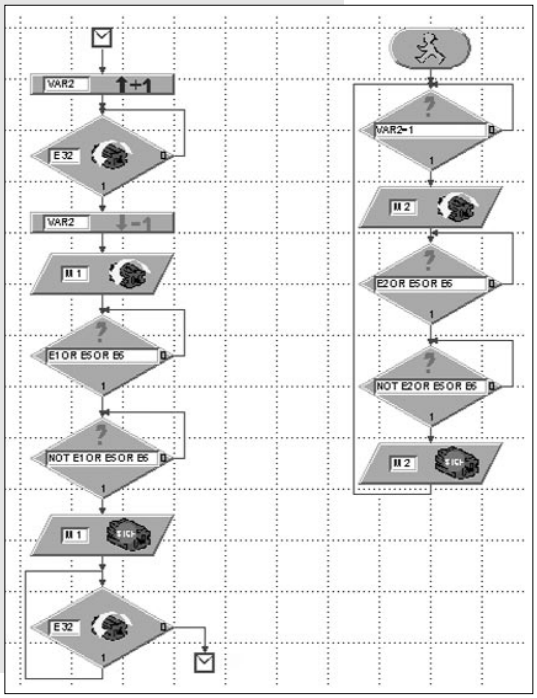
Mike loopt eerst altijd rechtdoor. Na iedere stap worden de knoppen E5 en E6 gepolst. Als E6 is ingedrukt, springt het programma naar de linkerafloop (eerst achteruit, dan naar rechts). Als E5 is ingedrukt, springt het naar de middelste afloop (eerst achteruit, dan naar links).

Aangezien de knoppen E5 en E6 alleen na iedere volledige stap worden gepolst, duurt het nogal lang totdat Mike op een hindernis reageert.

Opgave 5b:
Optimaliseer het subprogramma VOORUIT zodanig dat Mike sneller op een hindernis kan reageren.

Tip:
Gebruik voor het polsen van de knoppen E1 en E2 niet de component EDGE maar de component COMPARE. Pols daarmee bovendien of E5 of E6 zijn ingedrukt.

Oplossing:



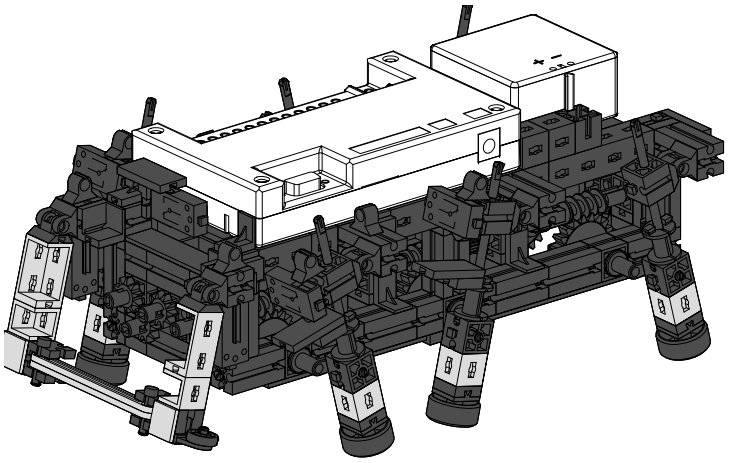
NL

Nu zou Mike perfect moeten functioneren. Dit programma staat eveneens op de CD onder MIKE_HINDERNIS.MDL. Je kunt het verbeterde subprogramma ook in het project MIKE_VOORBEELD.MDL integreren. Als E5 en E6 in een ander programma niet worden gepolst, is dat helemaal niet erg. Dit verbeterde voorbeeld hebben we onder MIKE_VOORBEELD_HINDERNIS.MDL opgeslagen.

Nadat we nu de eerste zesbenige robot uitvoerig hebben behandeld, richten we onze aandacht op het tweede model dat eveneens zes benen heeft. We noemen het „Jack".

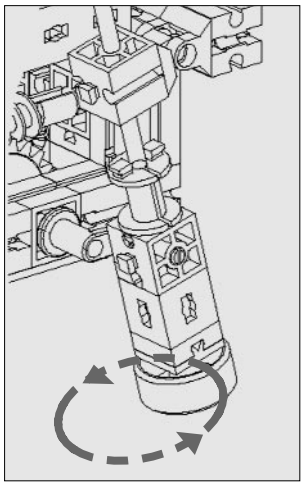
3.3 Model Jack

Jack maakt eveneens deel uit van het geslacht van de zesbenige fischertechnik-modellen. Hij heeft echter een volstrekt andere beenconstructie dan Mike.
Zet nu het model in elkaar volgens de bouwhandleiding vanaf p. 12.
De stappen 1-13 zijn bij Mike en Jack overigens identiek. Je hoeft Mike dus niet volledig te demonteren voordat je begint Jack te bouwen.



3.3.1 De constructie

De beenconstructie van Jack is eveneens een overbrenging met vier gewrichten. De hier toegepaste bouwvorm wordt „slingerend kruk- sleufmachanisme" genoemd. De drijfstang is in een beweegbare langsgeleider gelagerd die heen en weer slingert als de kruk draait. De cirkelbeweging die het voetpunt van het been maakt, is niet elliptisch als bij het model Mike, maar rond.



Hierdoor beweegt Jack's lichaam tijdens het lopen meer op en neer dan Mike. De stappen zijn ook korter. Jack kan echter in tegenstelling tot Mike hindernissen passeren. Bovendien doet deze overbrengingsbouwvorm meer aan een been denken dan bij Mike het geval is. Als Jack loopt, lijkt het alsof hij op stelten loopt.
Hij beweegt eveneens in de drievoetsgang net als insecten. Ook bij dit model is het belangrijk om de krukken precies volgens de bouwhandleiding af te stellen en de moeren goed aan te halen.

3.3.2 Het model programmeren

Het ligt voor de hand te denken dat voor Jack dezelfde programma's kunnen worden gebruikt als voor Mike. Probeer het maar eens!

Opgave 1:
Laat Jack met het programma MIKE_HINDERNIS.MDL. lopen.
Wat gebeurt er?

Waarneming:
Het model loopt foutloos vooruit en achteruit. Maar als het naar links en rechts draait, valt het voorover.

Opgave 2:
Hoe verklaar je dat?

Oplossing:
De twee modellen beschikken over verschillende beenconstructies. Bovendien worden de knoppen E1-E4 op een andere positie van het been ingedrukt. De manier waarop Mike draait, hoeft derhalve nog lang niet geschikt te zijn voor Jack – helaas.

Daar nemen we natuurlijk geen genoegen mee en we gaan zo snel mogelijk een oplossing vinden.

Opgave 3:
Probeer Jack zodanig te programmeren dat hij net als Mike hindernissen herkent, maar niet voorover valt als hij draait.

Tips:
Sla het project MIKE_HINDERNIS.MDL op als JACK_HINDERNIS.MDL en breng de noodzakelijke wijzigingen aan.

Als Jack draait, kunnen de twee motoren steeds tegelijkertijd lopen. Het afwisselend in- en uitschakelen vervalt. Het kritieke punt is dat de benen zich aan het begin van de draaibeweging, dus na het achteruit lopen, in de goede uitgangspositie bevinden.

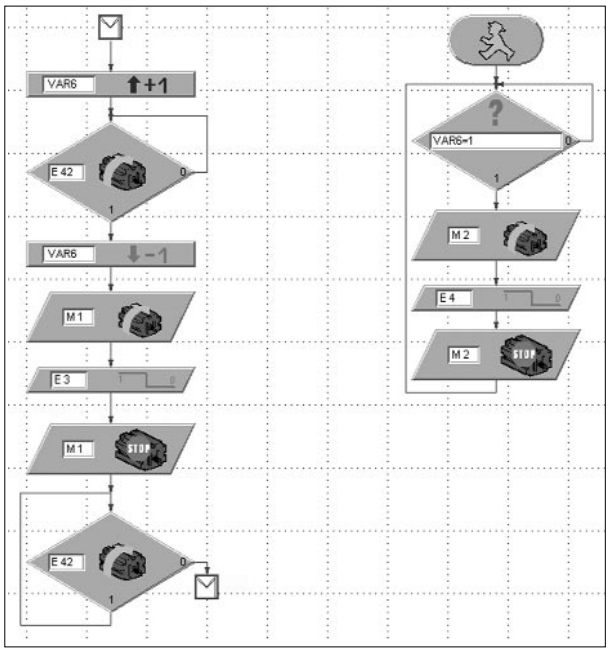
Linksom draaien:
Als het model linksom moet draaien, moet de kruk van het voorste linkerbeen aan het begin van de draaibeweging naar achter en de kruk van het voorste rechterbeen naar voren wijzen. Dit is het geval als tijdens het achteruit lopen aan de linkerkant de knop E2 en aan de rechterkant de knop E1 wordt ingedrukt en weer los wordt gelaten.

Rechtsom draaien:
Als het model rechtsom moet draaien, moet de kruk van het voorste linkerbeen aan het begin van de draaibeweging naar voren en de kruk van het voorste rechterbeen naar achter wijzen. Dit is het geval als tijdens het achteruit lopen aan de linkerkant de knop E4 en aan de rechterkant de knop E3 wordt ingedrukt en weer los wordt gelaten.

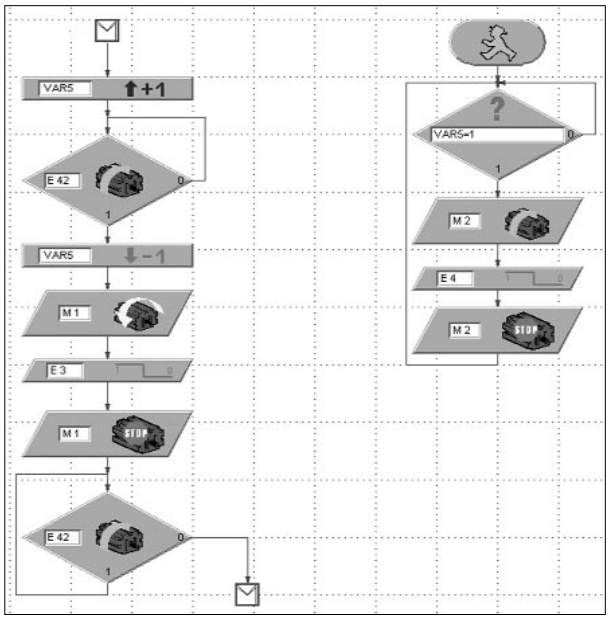
Tamelijk ingewikkeld, hè? Maar we zijn er bijna:

Voor de beweging achteruit moeten twee verschillende subprogramma's worden gebruikt.
Als het model naar links moet draaien, synchroniseer je de stappen tijdens het achteruit lopen met behulp van de knoppen E1 en E2. Dit komt overeen met het subprogramma ACHTERUIT uit het project MIKE_HINDERNIS.MDL.
Als het model naar rechts moet draaien, worden de stappen achteruit met behulp van E3 en E4 gesynchroniseerd.

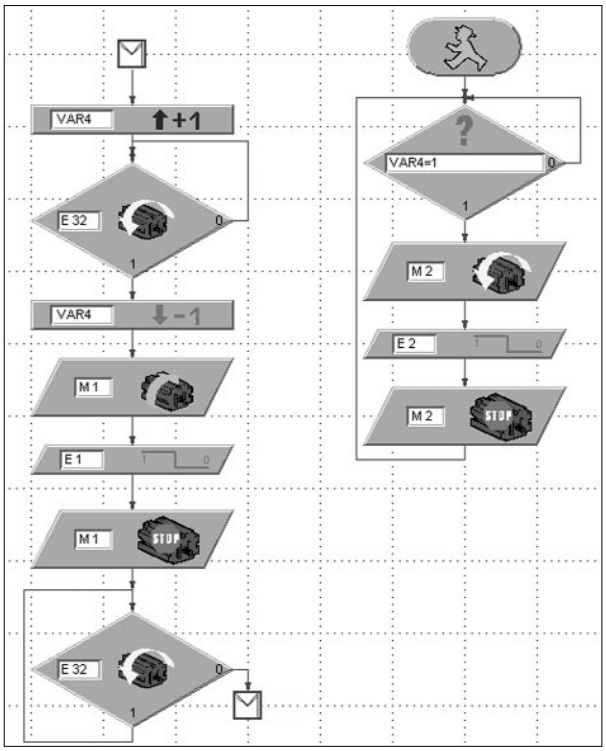
Je slaat dus het programma ACHTERUIT met het commando SUBPROGRAM - RENAME onder de naam ACHTERUIT_L op. Vervolgens kopieer je het met SUBPROGRAM - COPY in een tweede subprogramma ACHTERUIT_R. Daar verander je de knopdefinities voor het synchroniseren in E3 en E4. Vergeet niet voor ACHTERUIT_R een nieuwe variabele VAR6 voor het synchroniseren te gebruiken, anders gaat het behoorlijk mis. ACHTERUIT_R ziet er dan als volgt uit:



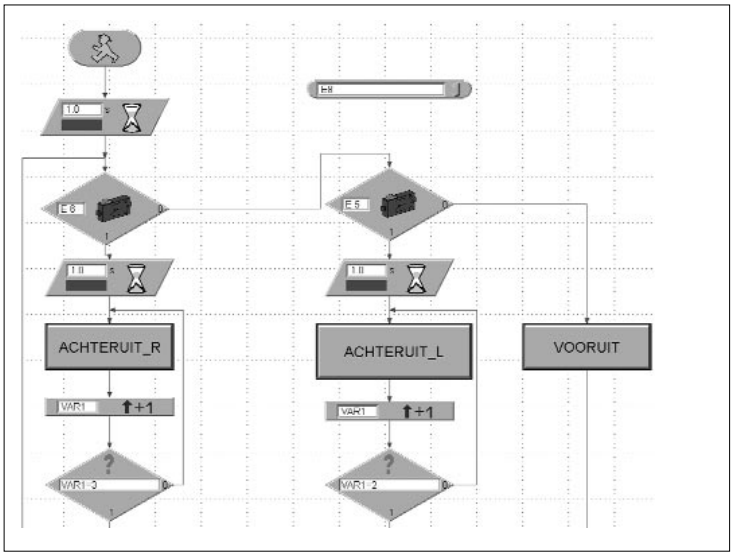
Nu moeten nog de subprogramma's voor het draaien zelf worden gewijzigd, zodat beide motoren steeds tegelijkertijd draaien. Het subprogramma LINKS bestaat uit de volgende componenten:



Je ziet dat ten opzichte van het subprogramma in het project MIKE_HINDERNIS.MDL een aantal componenten kunnen vervallen
Het subprogramma voor rechtsom draaien ziet er ongeveer hetzelfde uit, alleen met andere draairichtingen van de motoren. Bovendien worden de knoppen E1 en E2 gebruikt om de motoren te synchroniseren:



Als laatste vervang je in het hoofdprogramma in de onderverdeling voor het uitwijken naar rechts het subprogramma ACHTERUIT_L door ACHTERUIT_R:



Het hoofdprogramma blijft voor de rest ongewijzigd.

Klaar! Als je nergens een fout hebt gemaakt, kan Jack nu lopen zonder om te vallen bij het draaien. Als iets niet functioneert en je er niet achterkomt waarom, moet je daar niet te zwaar aan tillen, het was immers bepaald geen gemakkelijke klus. Je hebt in ieder geval nog de mogelijkheid om het uitgewerkte project JACK_HINDERNIS.MDL gewoon van de CD op te roepen en het model daarmee te besturen.

Als je het probleem zelf kon oplossen, mag je trots zijn, want dan ben je vanaf nu een professionele programmeur.

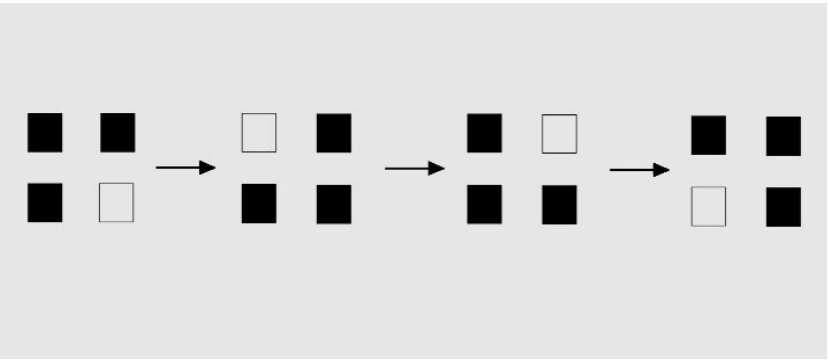
NL

4. Lopen op vier benen

4.1 Gangen van de zoogdieren

Om een looprobot met vier benen te construeren nemen we de natuur weer als voorbeeld en kijken we eens hoe zoogdieren zich voortbewegen.

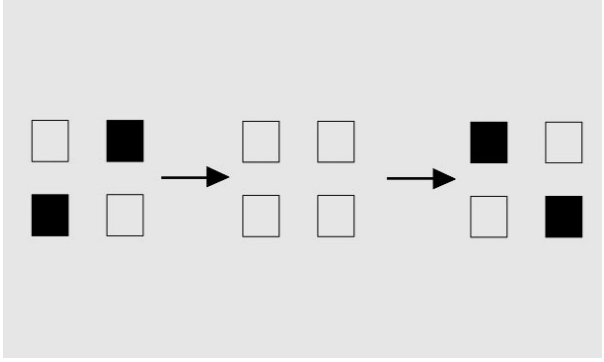
De langzaamste en veiligste gang is de zogeheten pas. Eén been zoekt een nieuwe plaats om op te staan, terwijl het lichaam van het dier op drie benen steunt. De dieren verplaatsen hun benen kruiselings in de volgorde: rechtervoorbeen, linkerachterbeen, linkervoorbeen, rechterachterbeen vooruit.



De zwarte vlakken zijn de benen die op de grond staan, de witte vlakken het opgetilde been.

Als we deze gang bij een looprobot willen toepassen, moeten we over het volgende nadenken:
Stel je voor dat je een been afzaagt van een tafel met vier poten. Wat gebeurt er? Juist, de tafel valt om. De drie benen vormen dus geen stabiele driepoot meer zoals bij de zesbenige soortgenoten het geval was. Dat maakt het construeren van een vierbenige robot moeilijker.

Hoe sneller de zoogdieren zich voortbewegen, hoe onstabieler hun gang wordt. Laten we nog even naar de gang „draf" kijken. Hierbij worden de benen diagonaal synchroon opgetild. Voordat ze echter de grond raken, worden de twee andere benen al opgetild. Dit betekent dat tijdelijk het bodemcontact volledig verloren gaat.

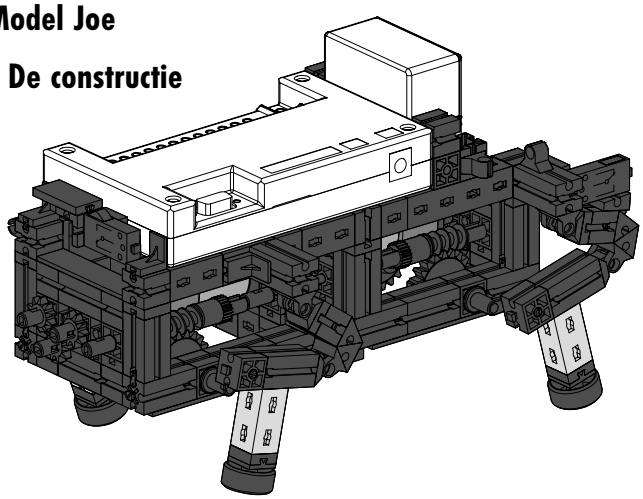


De kunt je zeker voorstellen dat een gang waarbij het bodemcontact tijdelijk verloren gaat voor een fischertechnik-model, waarop de interface en accupack zijn aangebracht, niet bepaald geschikt is.

Laten we het dus maar eens proberen met de gang „stap".

4.2 Model Joe

4.2.1 De constructie



Zet het model in elkaar zoals in de bouwhandleiding vanaf p. 20 staat uitgelegd.

De constructie van de benen is dezelfde als bij Mike. De stand van de krukken die de benen aandrijven moet bij Joe volledig anders zijn.
De krukken zijn ten opzichte van elkaar steeds 90° verdraaid. Je moet ze precies volgens de bouwhandleiding afstellen. Het synchroniseren van de linker- en rechterkant geschiedt wederom via de knoppen E1 en E2. Zodoende bereiken we de vereiste opeenvolging van stappen.
Om te voorkomen dat het model omvalt zodra een been wordt opgetild, moet het zwaartepunt van het model zo liggen dat het model op het juiste moment kantelt en door het zojuist ontlaste been wordt opgevangen.

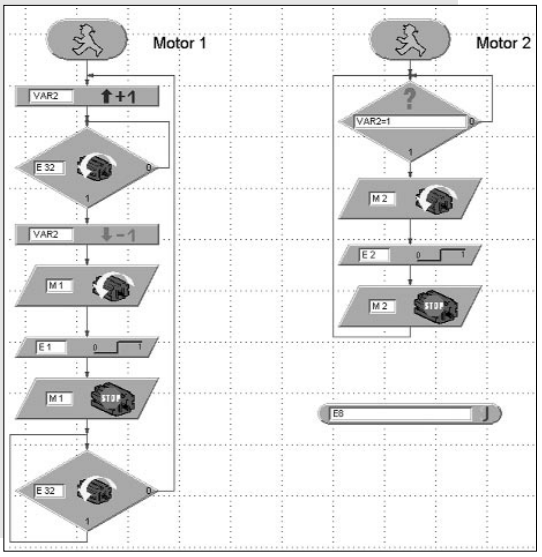
4.2.2 Het model programmeren

Bij dit model nemen we genoegen met het vooruit lopen.

Opgave 1:
Programmeer Joe zodanig dat hij in de gang „stap" vooruit beweegt.

Tips:
Gebruik voor elke motor een afzonderlijke afloop en synchroniseer de motoren met behulp van de knoppen E1 en E2.
Gebruik E8 als resetknop.

Oplossing:



Bij dit programma gebruiken we de 0-1 zijkant van de knoppen voor het synchroniseren. Op het moment dat de knoppen zijn ingedrukt, hebben de krukken van de 4 benen de juiste positie ten opzichte van elkaar. Dit project noemen we JOE.MDL.

Je ziet dat Joe veel moeizamer beweegt dan Mike en Jack. Door de noodzakelijke gewichtsverplaatsing schommelt het lichaam nogal sterk en zijn gang is zeker niet zo elegant als die van onze zesbenige vrienden.

Misschien wil je dit model nog aanleren om bochten te nemen, probeer gewoon of het lukt. Veel succes.

5. Lopen op twee benen

5.1 Tweebenige lopers

Het lopen op twee benen is niet in het geslacht van de zoogdieren ontstaan, maar komt ook bij een aantal reptielsoorten voor. Varanen, leguanen, agamen enz. gebruiken tijdens de vlucht alleen hun achterpoten. Op die manier bereiken ze zeer hoge snelheden. Daartoe hebben ze stevige achterpoten nodig, een lange staart om in evenwicht te blijven en een vlak terrein.

Vogels zijn eveneens tweebeners. De struisvogel hoort tot de snelste loopvogels. Hij bereikt snelheden tot wel 60 km/h.

De meest perfecte tweebener is de mens. Om rechtop te kunnen lopen moet het heupgewricht gestrekt zijn. Hiervoor wordt gezorgd door de grote bilspier. Bovendien kunnen de benen in het kniegewricht worden „vastgezet" en zodoende in een energiearme houding worden gefixeerd.

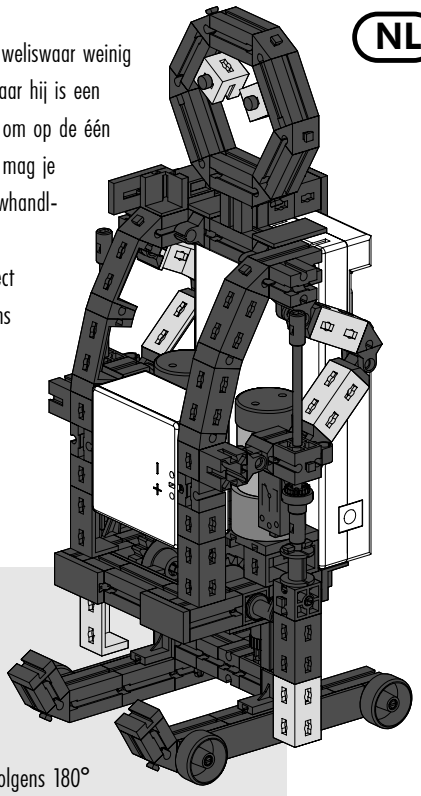
Het voortbewegen op twee benen is de moeilijkste manier, want ze veronderstelt naast de hierboven beschreven anatomische voorwaarden een zeer goed ontwikkeld evenwichtsgevoel. Voor ons is het vanzelfsprekend om op twee benen te lopen. Maar als we erbij stilstaan dat bij het optillen van één been het hele lichaam op slechts één been steunt en op die manier in balans moet worden gehouden, merken we dat juist het evenwicht behouden voor moeilijkheden zorgt bij deze manier van voortbeweging. Zelfs een pasgeboren soortgenoot kan niet meteen op twee benen lopen. Hij kruipt eerst „op armen en benen" voordat hij gaat staan en „leert te lopen".

Aan de Waseda-universiteit in Tokio werden reeds tweebenige robots ontwikkeld die met behulp van een groot aantal gewrichten, uiteenlopende sensoren, camera's en efficiënte microprocessors bewegen en in evenwicht blijven door hun gewicht te verplaatsen.

Voor ons bouwpakket Bionic Robots zou dat echter te kostbaar en ingewikkeld zijn. We hebben gezien dat we al bijna onze grenzen bereiken als een fischertechnik-model op vier benen loopt.

5.2 Model Jim

Om dit hoofdstuk echter niet alleen van de theoretische kant te bekijken, hebben we ten slotte besloten om ten minste een tweebenige skiër, we noemen hem Jim, te construeren. Hij heeft weliswaar weinig met een tweebenige loper te maken, maar hij is een aardige kerel die zijn uiterste best doet om op de één of andere manier vooruit te komen. Dit mag je niet missen. Het model staat in de bouwhandleiding op p. 27.
Als programma kun je gewoon het project JOE.MDL gebruiken. Je hoeft er niet eens iets aan te veranderen. Jim werkt ook met dit programma en kwakkelt naar voren.



Eén opgave hebben we nog voor je:

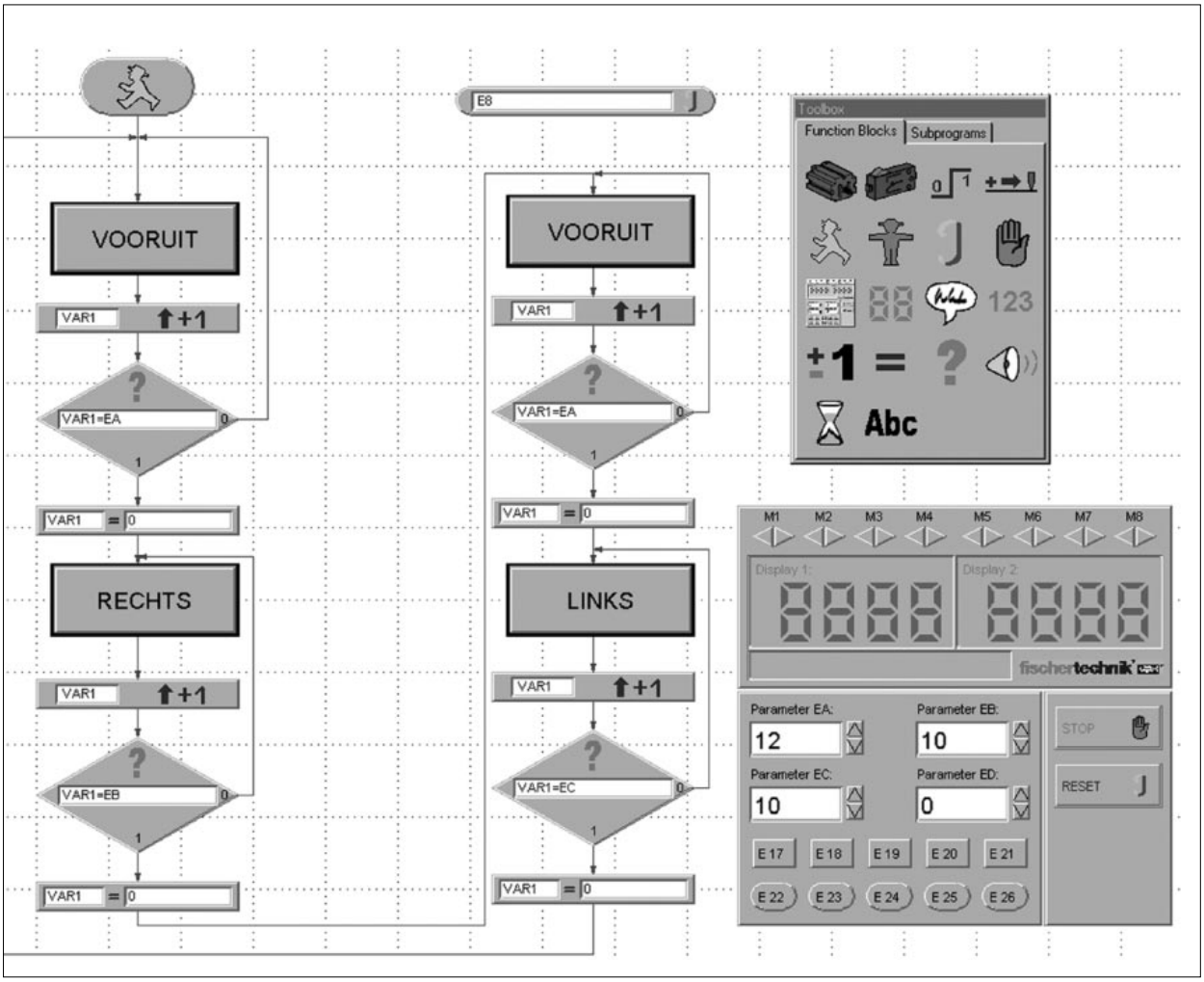
Opgave 1:
Programmeer Jim zodanig dat hij ongeveer 50 cm vooruit loopt, vervolgens 180° rechtsom draait, hetzelfde stuk terug loopt (vooruit), dan weer 180° linksom draait, hetzelfde stuk terug loopt etc. Gebruik voor het aantal stappen vooruit de terminale parameter EA, voor het aantal stappen linksom EB en voor rechtsom EC. Gebruik weer E8 als resetknop.

Tips:
Sla het project JOE.MDL op als JIM.MDL. Maak daarin van het hoofdprogramma een subprogramma „Vooruit" (componenten markeren en uitknippen, via EDIT – SUBPROGRAM een nieuw programma maken, componenten toevoegen, SUBIN en SUBOUT aanvullen, zie ook LLWin-handboek).

Maak van dit subprogramma de vereiste subprogramma's LINKS en RECHTS met het commando SUBPROGRAM – COPY. Wijzig hierin de draairichting van de motor en de opvraag van de draairichtingen van de motoren en gebruik voor ieder subprogramma een andere besturingsvariabele voor motor 2.

Vervolgens programmeer je het hoofdprogramma vergelijkbaar als bij MIKE_DANS.MDL. Het verschil is dat je nu de instelbare terminale parameters EA-EC gebruikt voor het aantal stappen. Hoeveel stappen Jim nodig heeft om 180° te draaien resp. een halve meter vooruit te komen, moet je even uitproberen.

Oplossing:
Hieronder volgt een schema van het hoofdprogramma. De subprogramma's kun je indien nodig direct op het beeldscherm bekijken. Bij ons heet het project eveneens JIM.MDL.



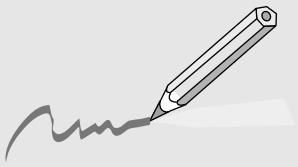
We hebben in het project ook meteen het subprogramma ACHTERUIT aangevuld, ook al is dat hier niet meteen nodig. Maar je wilt beslist dat Jim nog andere pasjes leert. Misschien moet hij daarbij ook een keer achteruit bewegen.

6. Samenvatting

Op je reis door de wereld van de Bionic Robots van fischertechnik ben je er zeker achter gekomen dat het niet altijd gemakkelijk was om onze vier vrienden te laten lopen. Het is nu eenmaal moeilijker om op benen te lopen dan op wielen te rollen. Met name het programmeren van de synchronisatie tussen de linker- en rechterkant bij het linksom of rechtsom draaien vereist een beetje concentratie. Maar voor iedereen die toch liever bezig is met het bouwen van de modellen, hebben we alle programma's kant en klaar op een CD gebrand, zodat iedereen de modellen kan bouwen en besturen.

Als je jezelf tot de professionele programmeurs mag tellen, heb je beslist nog een heleboel ideeën om nog een aantal opgaven voor Mike, Jack, Joe of Jim te programmeren, hetzij met extra sensors, zodat ze niet van de tafel naar beneden vallen, of dat ze in een labyrint niet verdwalen.

Met behulp van extra componenten kun je de modellen ook nog voorzien van een hoofd, slurf of staart. Laat je fantasie de vrije loop. Maak er iets van!



E

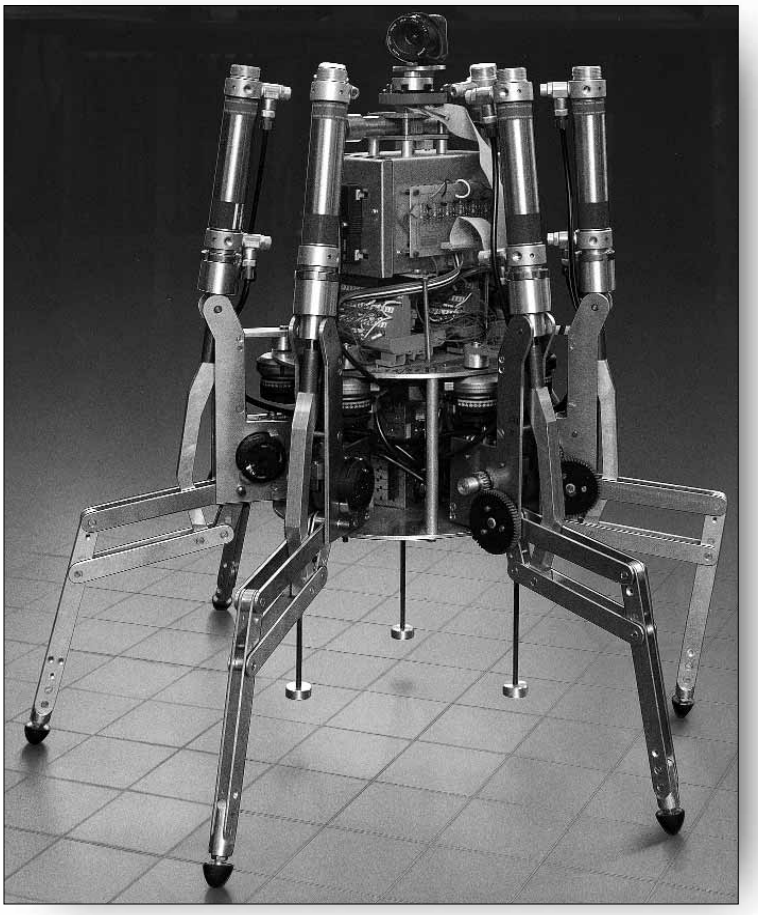
1. Biónica – La naturaleza como modelo

El término Biónica se compone de los dos términos Biología y Técnica. Este ramo de la ciencia intenta constantemente orientarse en la naturaleza para encontrar soluciones técnicas.
En su afán de moverse más lejos, rápido y eficaz de lo que le permite la naturaleza, el ser humano siempre ha inventado nuevas máquinas que facilitan este propósito de diferentes formas, según los respectivos requisitos. Los vehículos se desplazan sobre ruedas. En terrenos difíciles en los que fracasan los vehículos con ruedas, se utilizan vehículos con cadena. Los barcos flotan en el agua o son capaces de sumergirse. En algunos modos de desplazamiento, se utiliza la naturaleza como modelo. Así por ejemplo un avión parece un ave que planea.

Desde hace algunos años, los científicos investigan otra forma de movimiento muy corriente en la naturaleza, el andar o correr. Se desarrollan robots que son capaces de moverse sobre patas. Tales máquinas andantes pueden utilizarse en todas aquellas partes en las que apenas podrían utilizarse vehículos con ruedas o cadenas, por ejemplo en terrenos extremadamente accidentados o blandos, para trepar por encima de obstáculos, subir escaleras, atravesar fosos o para el uso en puntos difícilmente accesibles y peligrosos en plantas nucleares, minas o acciones de rescate.

Los primeros experimentos serios en el desarrollo de máquinas andantes, se hicieron en 1967 en una universidad de Tokio. Por primera vez los científicos se orientaron en el modo de andar humano, en lugar del de los insectos. El desarrollo constante de estos experimentos se tradujo en 1985 en la primera máquina andante de dos patas. Entretanto, estos robots tienen más de 50 grados de libertad y numerosos microprocesadores. Con la ayuda de una cámara, pueden, por ejemplo leer notas y tocar el órgano. Incluso es posible conversar con ellos.

Un ejemplo de un robot andante de seis patas es el robot andante electroneumático Achille", desarrollado en la Academia Militar Real de Bruselas. Equipado con una cámara en la parte superior y en las seis patas, este robot debe reaccionar mecánicamente a obstáculos elevados o hundidos (objetos o agujeros).
Ahora fischertechnik también se ha dedicado a este apasionante tema, y ha construido robots andantes, a los que luego se les da vida" por medio del Intelligent Interface y el software LLWin.



2. Requisitos e iniciación

Para que puedas construir los modelos del kit de construcción de computación Robots biónicos", además del kit de construcción necesitas los siguientes artículos:

Intelligent Interface, Art. N 30402
Software LLWin (a partir de la versión 3.0), Art. N 30407
Fuente de alimentación Accu Set, Art. N 34969

Si aún no estás familiarizado con el software LLWin y el interfaz, primero deberías leer el manual del software LLWin. En éste se describe cómo se instala el software y se conecta el interfaz. Además, se presta extraordinariamente para hacer primeras experiencias de cómo controlar modelos de fischertechnik a través del PC. Con unos pocos componentes del kit de construcción (motor y pulsador) puedes construir unos procesos de control muy sencillos para los modelos.

En cuanto estés familiarizado con el software y el interfaz, podrás atreverte con los modelos más complicados de Robots biónicos.

El kit de construcción incluye un CD-ROM que contiene programas de ejemplo de LLWin para los modelos del kit de construcción. Para poder abrir los programas, necesitas el software LLWin a partir de la versión 3.0. Puedes dejar los programas de ejemplo en el CD y llamarlos desde LLWin con el comando Archivo - Abrir, o bien copiar la carpeta completa ROBOTS_BIONICOS del CD al directorio del proyecto de LLWin en el disco duro, y abrir los ejemplos desde este lugar.

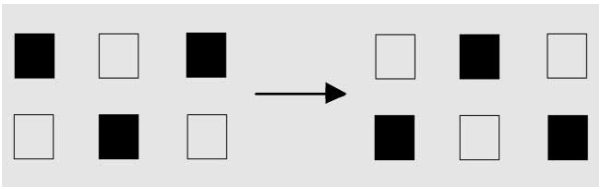
Antes de construir los modelos, tendrás que montar algunas piezas, p.ej. el cable y el conector. En el manual de construcción se describe qué es lo que debe hacerse exactamente.

Bueno, ya estás listo. Ahora puedes sumergirte en el fascinante mundo de los robots andantes de fischertechnik. En cuanto hayas terminado el primer modelo y éste comience a moverse de forma casi fantasmagórica, estarás entusiasmado con esta técnica que en la naturaleza se emplea desde hace millones de años para el movimiento.

3. 3. Andar con 6 patas

3.1 Modo de andar de los insectos

El modo de andar de los insectos es un modelo extraordinario para el accionamiento de máquinas de seis patas". En el modo de andar a tres patas, siempre se alzan tres de las seis patas simultáneamente del suelo, es decir las patas delantera y trasera de un lado junto con la pata central del otro lado:

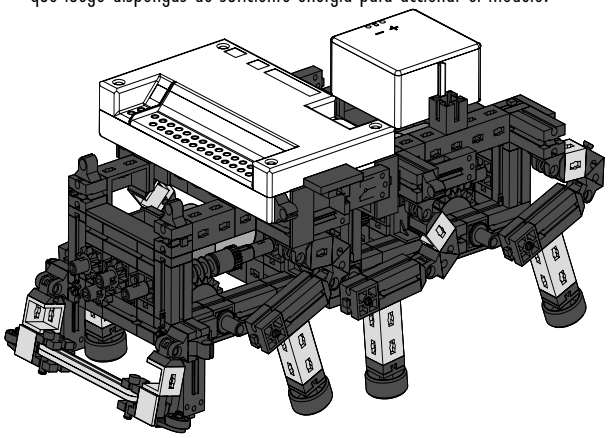


Las patas que permanecen en el suelo (representadas en color negro), forman un estable trípode, de modo que el modelo siempre está bien estabilizado y no vuelca al andar.

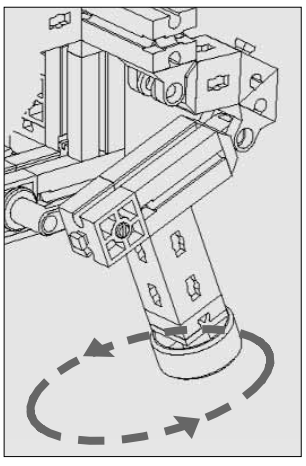
3.2 Modelo Mike

3.2.1 Construcción del modelo

Construye ahora el modelo de seis patas Mike (ver el manual de construcción, pág. 4). Carga el paquete de baterías durante la construcción, para que luego dispongas de suficiente energía para accionar el modelo.



Las patas del modelo están construidas, de tal modo que forman un llamado cuadrilátero articulado. La forma constructiva del cuadrilátero utilizado en este caso, se llama manubrio oscilante. Accionados por una manivela, los elementos articulados del cuadrilátero efectúan unos movimientos oscilantes.



Las distancias entre cada una de las articulaciones y la posición del pie (es decir el extremo inferior de la pata), se han diseñado de tal modo que el pie realice un movimiento elíptico cuando gira la manivela de accionamiento. De este modo se genera un movimiento que se asemeja a un paso al andar. Las 6 manivelas que accionan las patas, han de ajustarse exactamente como lo indica el manual de construcción. Las tres patas que se colocan simultáneamente sobre el suelo, tienen la misma posición de manivela. Las manivelas de las 3 patas que en este momento se encuentran en el aire, están giradas en 180° respecto a las otras patas. La posición correcta de las manivelas entre sí, garantiza que el modelo pueda andar en la secuencia de pasos correcta, es decir en el modo de andar a tres patas.

Las tuercas de las pinzas y de los cubos, con las que se fijan los tornillos sin fin y las ruedas dentadas en los ejes, han de estar bien apretadas, para que no se desajusten las manivelas al andar.

Los lados derecho e izquierdo del modelo son accionados por un motor, respectivamente (esto se necesita para realizar curvas). Por este motivo hay que cuidar de que la pata central de un lado siempre se encuentre en la misma posición que las dos patas exteriores del otro lado. Esta sincronización es realizada a través de los pulsadores E1 y E2, controlada por software.

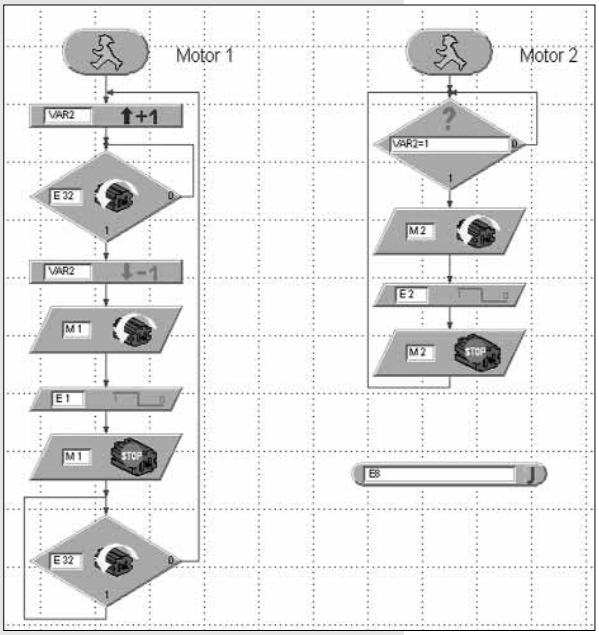
Prueba con el diagnóstico de interfaz, si están correctamente conectados todos los pulsadores y motores. Sentido de giro de los motores: Sentido de giro a la izquierda=adelante

3.2.2 Primer programa

Ahora comenzaremos a enseñarle algunas cosas a Mike. Primero el modelo sólo debe andar recto. Más adelante nos ocuparemos de las curvas y de la reacción ante obstáculos.

Tarea 1:
Programa el modelo de tal manera que ande recto en el modo de andar a tres patas. Utiliza los pulsadores E1 y E2 para sincronizar las patas izquierdas y derechas. Cuida de que siempre tengan la misma posición las dos patas exteriores de un lado y la pata central del otro lado. Además, utiliza el pulsador E8 como pulsador de reset.

Consejos:
Programa una secuencia propia para cada motor. Controla la secuencia para el motor M2 mediante una variable VAR2. Si no necesitas el interfaz durante la programación, deberías interrumpir la conexión eléctrica entre el paquete de baterías y el interfaz para ahorrar energía.



Solución:
El programa para andar recto tiene el siguiente aspecto:

La variable VAR2 da el impulso para que arranque el motor M2. Seguidamente arranca el motor M1. En cuanto se accione el pulsador E1, se detiene M1. En cuanto se accione E2, se detiene M2. La primera secuencia espera hasta que se haya parado M2 (el estado del motor M2 es interrogado a través de E32; ver el apartado Interrogación de los estados de los motores en el manual de LLWin).

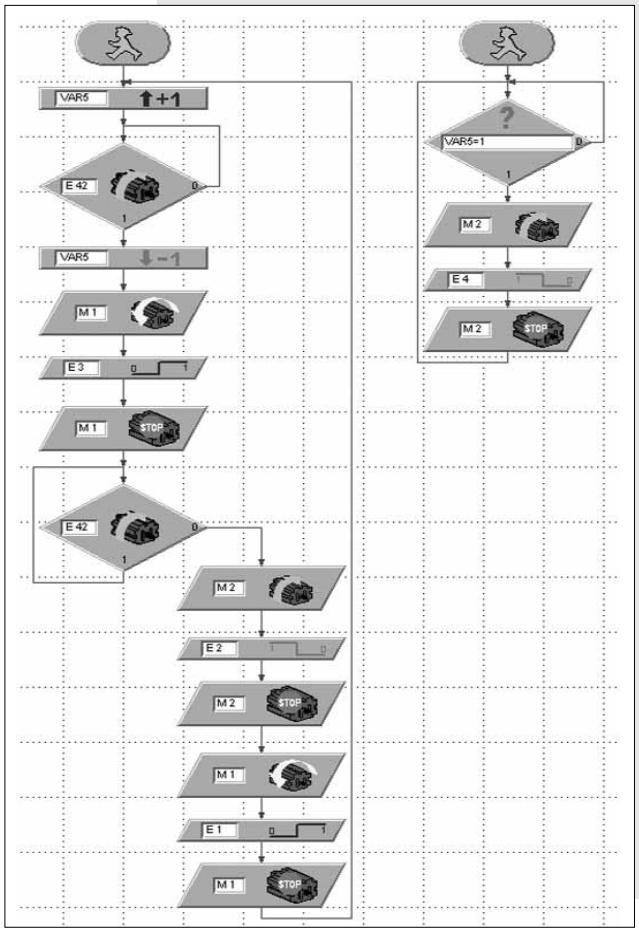
Por cierto: Si no te apetece crear esta secuencia, podrás encontrarla como proyecto de ejemplo MIKE_RECTO.MDL en el CD adjunto.
Inicia el proyecto. Si has programado todo correctamente, tu modelo cobrará vida y andará en línea recta. Enhorabuena. Has dado el primer paso.

3.2.3 Giro a la izquierda

Naturalmente no nos es suficiente que Mike sólo ande recto. Ahora queremos que gire sin que ande.

Tarea 2:
Programa Mike de tal modo que gire a la izquierda.

Consejos:
El modelo gira a la izquierda, si M1 gira a la izquierda y M2 gira a la derecha.
Naturalmente también puedes operar el modelo sin sincronización. En este caso también gira, pero hay posiciones en las que el modelo vuelca hacia delante. Puedes evitar esto con la siguiente secuencia:



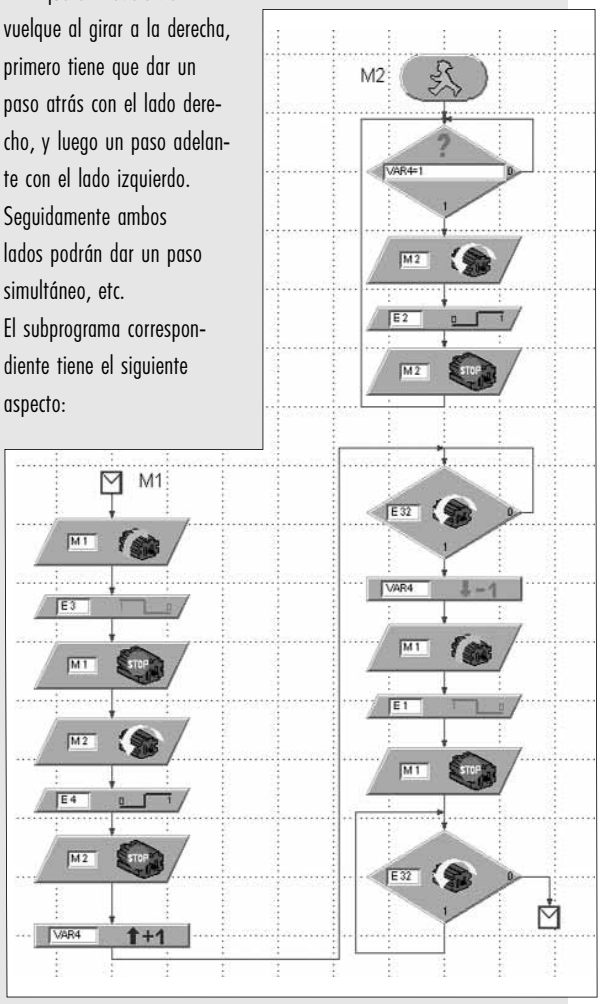
Por medio de los pulsadores E1-E4, los lados izquierdo y derecho del modelo primero dan simultáneamente un paso, luego el lado izquierdo da un paso, luego el derecho, etc. De este modo el modelo nunca vuelca hacia delante. ¡Inténtalo! Esto hará que comprendas mejor este orden.
Esta secuencia también puedes encontrarla como proyecto MIKE_IZQUIERDA.MDL en el CD.

Ahora el modelo sabe andar recto y girar a la izquierda. Sólo falta el andar hacia atrás y el giro a la derecha. En principio el andar hacia atrás funciona como el andar adelante, pero con el sentido de giro del motor invertido.
En principio, el giro a la derecha funciona al revés que el giro a la izquierda.

3.2.4 Izquierda, derecha, adelante, hacia atrás

Tarea 3:
A continuación, programa cada una de las funciones RECTO, HACIA ATRÁS, IZQUIERDA Y DERECHA como subprograma, para que puedas utilizarlas posteriormente de forma flexible en diferentes proyectos.

Consejos:
En el manual de LLWin se describe, cómo se copia una secuencia existente a un subprograma.
Utiliza en cada subprograma otra variable (VAR2-VAR5) para iniciar la secuencia para el motor M2.
Para que el modelo no vuelque al girar a la derecha, primero tiene que dar un paso atrás con el lado derecho, y luego un paso adelante con el lado izquierdo. Seguidamente ambos lados podrán dar un paso simultáneo, etc.
El subprograma correspondiente tiene el siguiente aspecto:

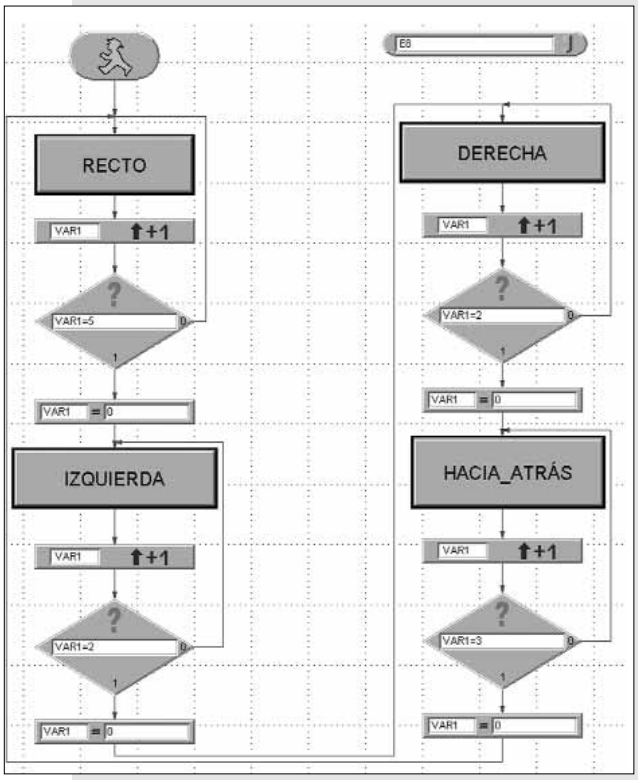


No hemos representado los otros subprogramas. Si tienes problemas a la hora de programar una secuencia, consulta los subprogramas terminados en el archivo MIKE_PLANTILLA.MDL en el CD. El programa principal de este proyecto está vacío. En la pestaña Subprogramas" de la ventana de bloques funcionales, encontrarás la lista con los subprogramas disponibles que podrás insertar en el programa principal.

Pero espera antes de mirar. Intenta encontrar la solución tú mismo. Si no lo logras, aún puedes mirar.
Para probar todos los subprogramas, ahora vamos hacer que Mike baile.

Tarea 4:
Programa Mike de tal modo que dé 5 pasos adelante, gire 2 pasos a la izquierda, luego 2 pasos a la derecha, a continuación 3 pasos hacia atrás y luego comience desde el principio. Utiliza la variable Var1 como variable de contaje para la cantidad de pasos. Utiliza E8 como pulsador de reset.

Solución:



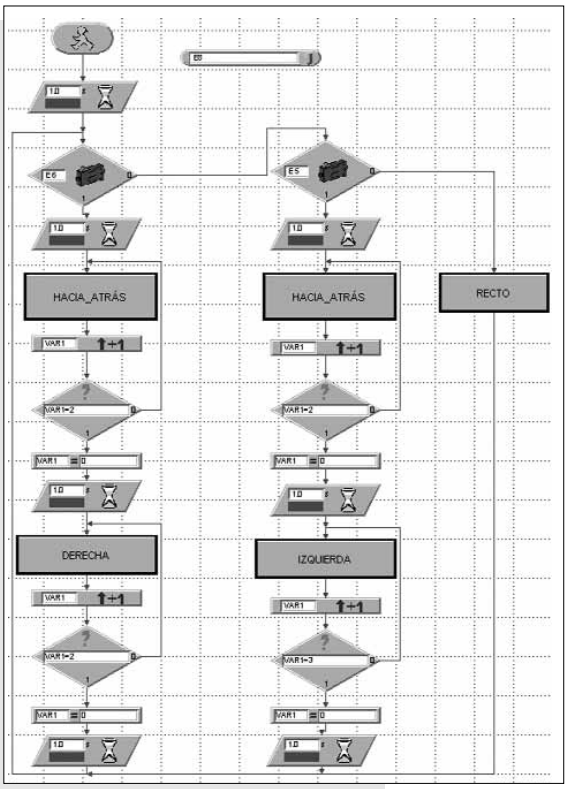
Este proyecto se llama MIKE_BAILE.MDL.

3.2.5 Reconocer obstáculos

Para terminar queremos que Mike detecte y sortee obstáculos con su parachoques móvil (o digamos mejor sensor).

Tarea 5a:
Programa Mike de tal modo que en caso de un obstáculo en su sensor izquierdo (pulsador E6) primero dé 4 pasos hacia atrás y luego 2 pasos a la derecha para sortearlo. Si hay un obstáculo en su sensor derecho (pulsador E5), deber dar 4 pasos hacia atrás y luego 3 pasos a la izquierda para sortearlo.

Solución:

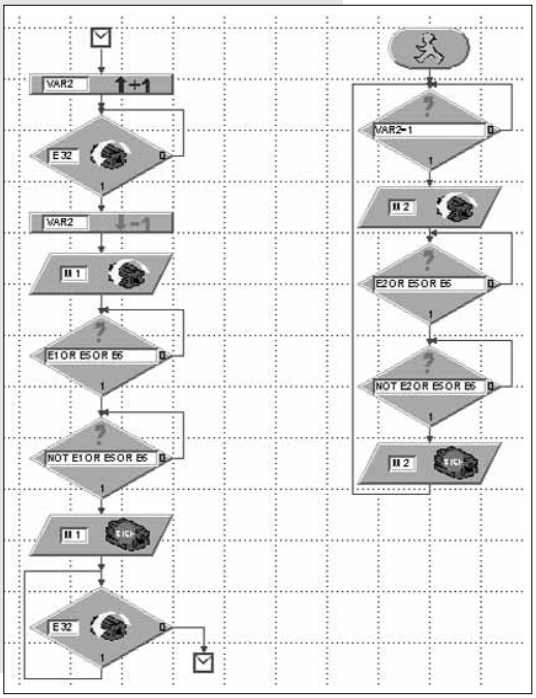


En primer lugar Mike siempre anda en línea recta. Después de cada paso se interrogan los pulsadores E5 y E6. Si está accionado E6, el programa se bifurca a la secuencia izquierda (primero hacia atrás, luego a la derecha). Si está accionado E5, pasa a la secuencia central (primero hacia atrás, luego a la izquierda). Dado que los pulsadores E5 y E6 sólo son interrogados después de cada paso completo, Mike tarda relativamente mucho tiempo hasta que reacciona ante un obstáculo.

Tarea 5b:
Optimiza el subprograma RECTO de tal modo que Mike pueda reaccionar más rápidamente ante un obstáculo.

Consejo:
No utilices el bloque funcional FLANCO para interrogar los pulsadores E1 y E2, sino el bloque funcional COMPARACIÓN. C on éste, interroga adicionalmente si está accionado E5 ó E6.

Solución:

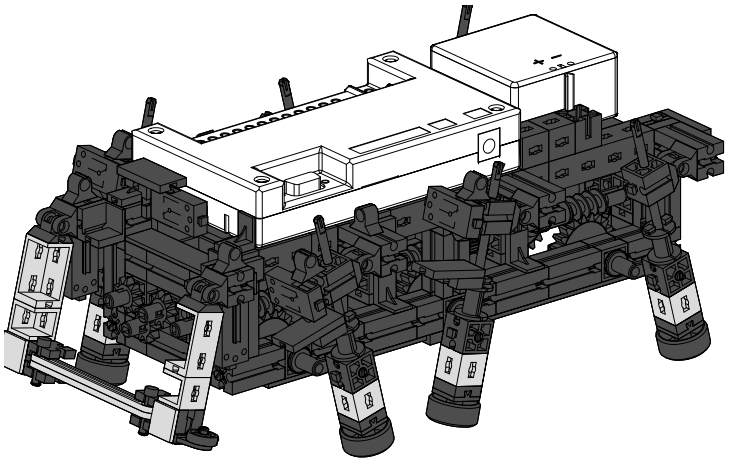


Ahora Mike debería funcionar perfectamente. Este programa también se encuentra en el CD, bajo MIKE_OBSTACULO.MDL. También puedes integrar el subprograma optimizado en el proyecto MIKE_PLANTILLA.MDL. Si en otro programa no se interrogan E5 y E6, esto no molestará. Hemos guardado esta plantilla optimizada como MIKE_PLANTILLA_OBSTACULO.MDL.

Después de habernos ocupado extensamente del primer modelo de seis patas, nos dedicaremos al segundo modelo que también tiene seis patas. Lo llamamos Jack".

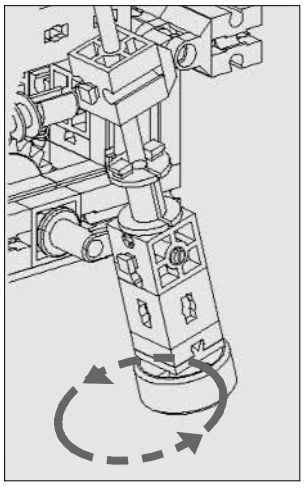
3.3 Modelo Jack

Jack también pertenece al género de modelos de seis patas de fischertechnik. Sin embargo, se distingue considerablemente de Mike, en lo que a la construcción de sus patas se refiere.
Construye el modelo, tal y como se describe en el manual de construcción, a partir de la pág. 12. Por cierto: las etapas de construcción 1-13 son idénticas para Mike y Jack. Es decir que no tienes que desensamblar completamente a Mike, antes de empezar a construir a Jack.



3.3.1 Construcción

Las patas de Jack también están construidas como cuadrilátero articulado. La forma constructiva utilizada en este caso, se llama manubrio oscilante con corredera. La biela está alojada en una guía longitudinal móvil que oscila en ambas direcciones cuando gira la manivela. La curva que realiza el pie de la pata no es tan elíptica como en el modelo Mike, sino más bien circular.



De esta manera el cuerpo de Jack sube y baja más que el de Mike mientras anda. Los pasos son más cortos. En cambio, Jack puede superar pequeños obstáculos, lo cual no puede hacer Mike. Además, esta forma constructiva del cuadrilátero recuerda más a una pata que en el caso de Mike. Cuando Jack anda, parece que anduviese con zancos. También se mueve a tres patas como los insectos. En este modelo también es importante ajustar las manivelas exactamente como lo describe el manual de construcción, y apretar bien las tuercas de las pinzas y de los cubos.

3.3.2 Programación

Es natural pensar que para Jack podrían utilizarse los programas con los que funciona Mike. ¡Inténtalo!

Tarea 1:
Opera Jack con el programa MIKE_OBSTACULO.MDL. ¿Qué puedes observar?

Observación:
El modelo avanza y retrocede perfectamente. Pero al girar a la izquierda y derecha, vuelca hacia delante.

Tarea 2:
¿Cómo te lo explicas?

Solución:
Las patas de ambos modelos están construidas de distintas formas. Además, los pulsadores E1-E4 se accionan en otra posición de la pata. Por lo tanto, la manera en la que gira Mike, no tiene por qué funcionar para Jack - mala suerte.

Claro que esto no nos gusta, y queremos encontrar una solución cuanto antes.

Tarea 3:
Intenta programar Jack de tal manera que detecte obstáculos del mismo modo que Mike, pero que no vuelque hacia delante al girar.

Consejos:
Guarda el proyecto MIKE_OBSTACULO.MDL con el nombre JACK_OBSTACULO.MDL y realiza los cambios necesarios.

Cuando Jack gira, siempre pueden marchar ambos motores al mismo tiempo. La conexión y desconexión alternativas se suprimen. Es decisivo que al comenzar el giro, es decir después de retroceder, las patas se encuentren en la posición inicial correcta.

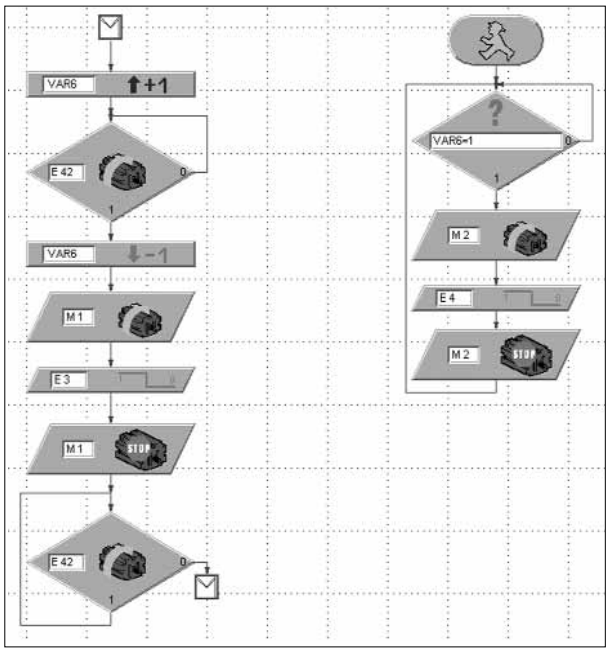
Giro a la izquierda:
Si el modelo debe girar a la izquierda, al comenzar el giro la manivela de la pata delantera izquierda debe señalar hacia atrás y la de la pata delantera derecha hacia delante. Este es el caso, si durante la marcha atrás se han accionado y se han vuelto a soltar el pulsador E2 en el lado izquierdo y el pulsador E1 en el lado derecho.

Giro a la derecha:
Si el modelo debe girar a la derecha, al comenzar el giro la manivela de la pata delantera izquierda debe señalar hacia delante y la de la pata delantera derecha hacia atrás. Este es el caso, en cuanto durante la marcha atrás se hayan accionado y se hayan vuelto a soltar el pulsador E4 en el lado izquierdo y el pulsador E3 en el lado derecho.

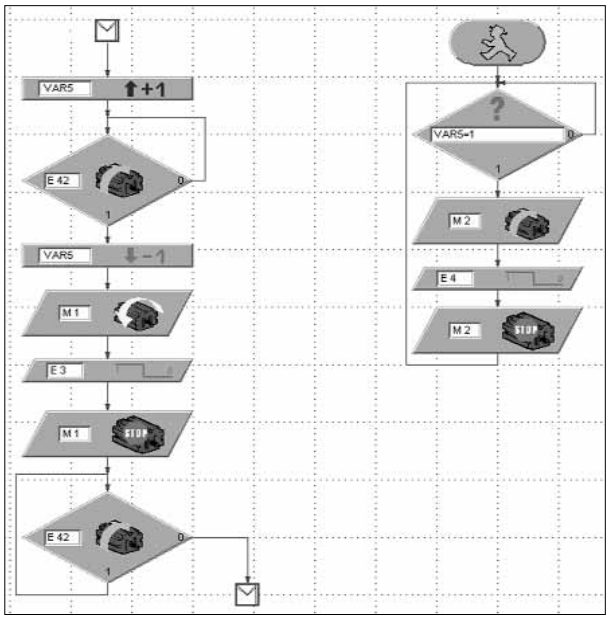
Bastante complicado, ¿no te parece? Pero tranquilo, en seguida lo tendremos:

Para el movimiento de retroceso han de utilizarse dos subprogramas distintos.
Si el modelo debe girar a la izquierda, sincroniza los pasos durante el retroceso con los pulsadores E1 y E2. Esto equivale al subprograma HACIA ATRÁS del proyecto MIKE_OBSTACULO.MDL.
Si el modelo debe girar a la derecha, los pasos hacia atrás se sincronizan con E3 y E4.

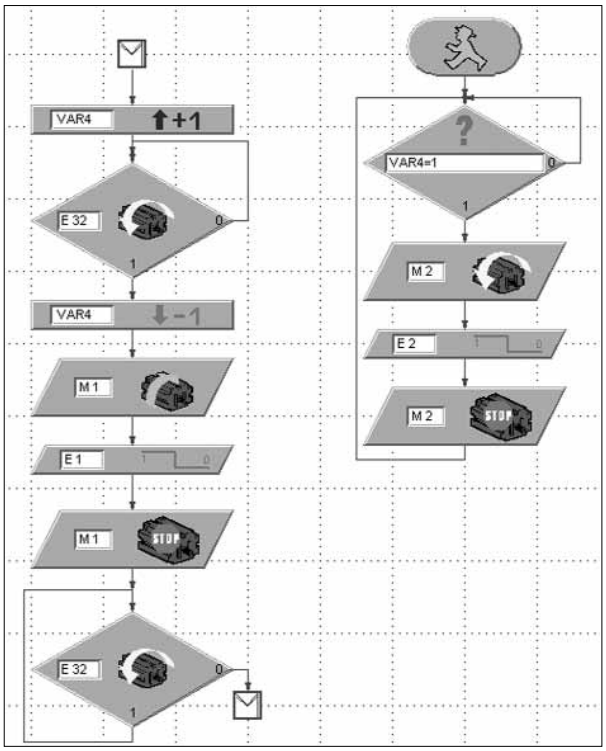
Por lo tanto, renombra el subprograma HACIA ATRÁS con el comando SUBPROGRAMA - RENOMBRAR en ATRAS_IZQ. Seguidamente, cópialo con SUBPROGRAMA - COPIAR a un segundo subprograma ATRAS_DCH. En éste, cambia los nombres de los pulsadores para la sincronización, a E3 y E4. Tampoco olvides utilizar para ATRAS_DCH una nueva variable VAR6 para la sincronización, porque si no el caos será total. ATRAS_DCH tendrá el siguiente aspecto:



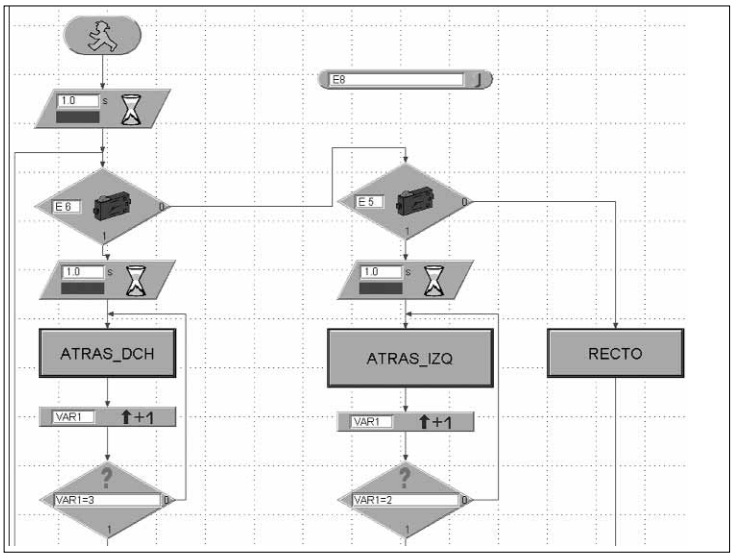
Ahora han de modificarse los subprogramas para el propio giro, de modo que los dos motores siempre giren simultáneamente. El subprograma IZQUIERDA consta de los siguientes bloques funcionales:



Como ves, frente al subprograma en el proyecto MIKE_OBSTACULO.MDL, pueden suprimirse algunos bloques funcionales.
El subprograma para el sentido de giro a la derecha es similar, pero con otros sentidos de giro de motor. Además, se utilizan los pulsadores E1 y E2 para sincronizar los dos motores:



Por último, reemplaza en el programa principal, en la bifurcación para la maniobra de sorteo a la derecha, el subprograma ATRAS_IZQ por ATRAS_DCH:



El resto del programa principal no cambia.

¡Lo has logrado! Si no has cometido ningún error, ahora Jack debería andar sin volcar cuando gira. Si alguna cosa no funciona y no sabes por qué, no te preocupes, pues éste ha sido un hueso verdaderamente duro de roer.
En cualquier caso tienes la posibilidad de llamar simplemente el proyecto terminado JACK_OBSTACULO.MDL del CD y de operar el modelo con el mismo.

Si has logrado solucionar el problema puedes estar orgulloso, pues a partir de ahora formas parte de los programadores profesionales.

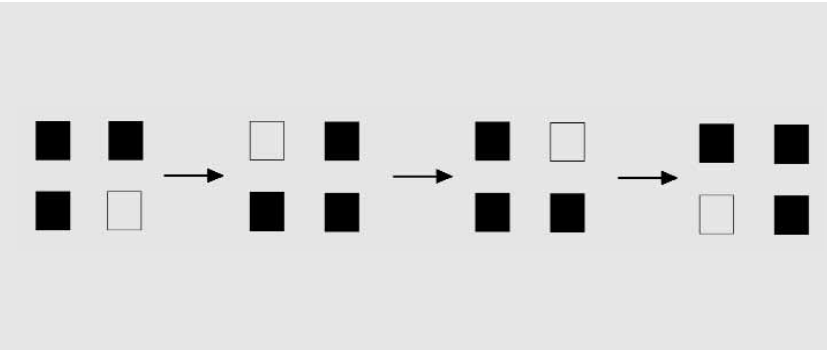
E

4. Andar con 4 patas

4.1 Modos de andar de los mamíferos

Para construir un robot andante de 4 patas, tomamos de nuevo la naturaleza como modelo y observamos cuáles son los modos de andar de los mamíferos.

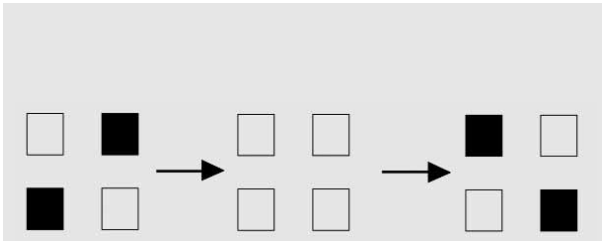
El modo de andar más lento y más seguro es el paso. Una pata busca una nueva posición, mientras el cuerpo del animal se apoya en tres patas. Los animales avanzan de forma cruzada, con la siguiente secuencia de pasos: pata delantera derecha, pata trasera izquierda, pata delantera izquierda, pata trasera derecha.



Los cuadros negros representan las patas colocadas en el suelo, y los cuadros blancos la pata alzada.

Si queremos utilizar este modo de andar para un robot, tenemos que tener en cuenta lo siguiente:
Imagínate que sierras una pata de una mesa de 4 patas. ¿Qué ocurre? Correcto, la mesa vuelca. Por lo tanto las tres patas ya no forman ningún trípode estable, lo cual era el caso en los modelos de seis patas. Esto dificulta la construcción de un robot cuadrúpedo.

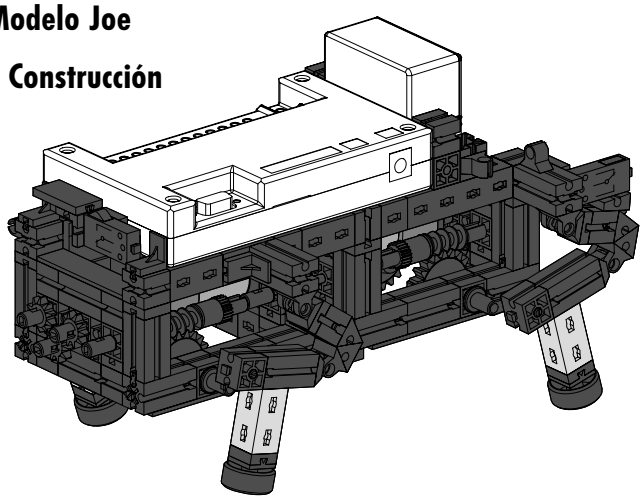
Cuanto más rápidamente avanzan los mamíferos, más inestable su modo de andar. Contemplemos brevemente el modo de andar trote". Durante el trote, las patas se alzan sincrónicamente en diagonal. Pero antes de que toquen el suelo, se alzan las otras dos patas. Esto significa que por un tiempo se pierde totalmente el contacto con el suelo.



Seguramente podrás imaginarte que un modo de andar en el que se pierde temporalmente el contacto con el suelo, no es muy adecuado para un modelo de fischertechnik, en el que deben encontrarse el interfaz y el paquete de baterías.
Por lo tanto, intentémoslo con el modo de andar paso".

4.2 Modelo Joe

4.2.1 Construcción



Construye el modelo al igual que se describe en el manual de construcción, a partir de la pág. 20.

La construcción de las patas es idéntica a la de Mike. La posición de las manivelas que accionan las patas, tiene que ser completamente distinta en Joe. Las manivelas están desplazadas entre sí en 90°, respectivamente. Debes ajustarlas exactamente como lo indica el manual de construcción.
La sincronización de los lados izquierdo y derecho se efectúa de nuevo a través de los dos pulsadores E1 y E2. De esta manera obtenemos la secuencia de pasos requerida.
Para que no vuelque el modelo en cuanto alce una pata, el centro de gravedad del modelo ha de estar situado de tal modo, que el modelo se incline en el momento preciso y se apoye en la pata aliviada en ese momento, respectivamente.

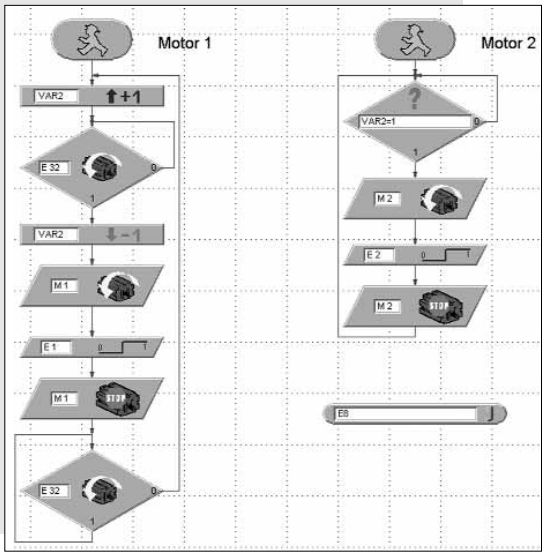
4.2.2 Programación

En este modelo vamos a conformarnos con el movimiento en línea recta.

Tarea 1:
Programa Joe de tal modo que avance en el modo de andar paso.

Consejos:
Utiliza una secuencia separada para cada motor y sincroniza ambos lados con los pulsadores E1 y E2.
Utiliza E8 como pulsador de reset.

Solución:



En este programa utilizamos el flanco 0-1 de los pulsadores para la sincronización. En el momento en que estén accionados los pulsadores, las manivelas de las 4 patas tienen la posición correcta entre sí. El proyecto se llama JOE.MDL.

Como ves, Joe se mueve con mucha más dificultad que Mike y Jack. Debido al desplazamiento de peso necesario, el cuerpo se tambalea fuertemente y el estilo de andar es mucho menos elegante que en los modelos de seis patas.

Si te apetece enseñarle al modelo a realizar curvas, intenta a ver si lo logras. Mucha suerte.

5. Andar con 2 patas

5.1 Andadores de 2 patas

El andar con dos patas no es una cosa que haya salido del género de los mamíferos, sino que también es practicado por algunos tipos de reptiles. Los varanos, las iguanas, las agamas y lagartos corredores sólo utilizan sus patas traseras para huir. De este modo alcanzan unos pasos muy grandes, y por consiguiente una velocidad muy alta. Para ello precisan de unas patas traseras muy fuertes, una cola compensadora larga y un terreno llano.

Las aves también pertenecen a la especie de los bípedos. Una de las aves corredoras más rápidas es el avestruz. Esta alcanza unas velocidades constantes de hasta 60 km/h.

El ser humano es el bípedo más perfecto. Su forma de andar completamente erguida requiere que se estire la articulación de la cadera. Esto queda garantizado por el glúteo mayor. Además, las piernas pueden encastrarse en la rótula, quedando fijadas de este modo en una postura pobre en energía.

El movimiento a dos patas es el modo de andar más difícil, porque aparte de los requisitos anatómicos descritos, requiere un sentido de equilibrio bien desarrollado. A los humanos el andar con las dos piernas nos parece algo natural y sencillo. Pero si consideramos que al alzar una pierna el cuerpo entero descansa sobre una sola pierna y ha de ser equilibrado de esta forma, apercibimos que precisamente el mantener el equilibrio hace que esta forma de desplazamiento sea tan complicada. Incluso un ser humano recién nacido no puede andar inmediatamente con las dos piernas. Primero anda a gatas, antes de erguirse y aprender a andar.

En la Universidad Waseda de Tokio ya se han desarrollado robots de dos patas, que se mueven por medio de numerosas articulaciones, diferentes sensores, cámaras y potentes microprocesadores, y mantienen el equilibrio desplazando el peso.

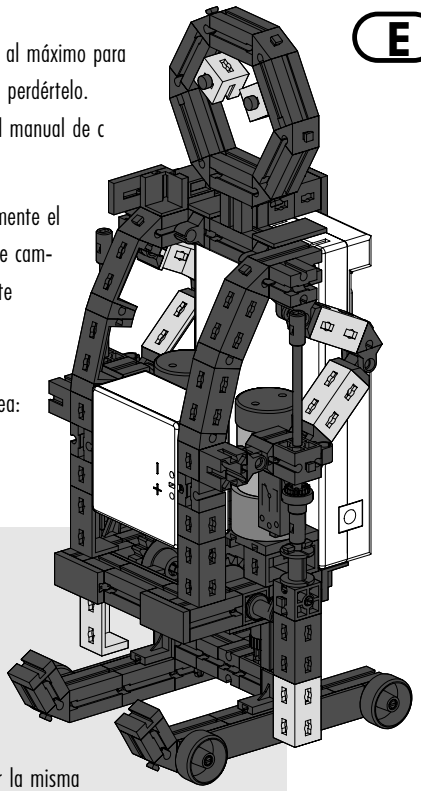
Sin embargo, para nuestro kit de construcción Robots biónicos esto sería algo demasiado complicado. Hemos visto que al andar con cuatro patas con un modelo de fischertechnik, vamos alcanzando nuestros límites.

5.2 Modelo Jim

Pero para que no contemplemos este capítulo solamente de forma teórica, para concluir nos hemos permitido construir al menos un esquiador bípedo que se llama Jim. Aunque sólo tiene poco que ver con un modelo andante bípedo, es muy simpático y se esfuerza al máximo para avanzar de alguna manera. No deberías perdértelo. Encontrarás el modelo en la pág. 27 del manual de c onstrucción.

Como programa puedes utilizar sencillamente el proyecto JOE.MDL. Ni siquiera tienes que cambiar nada. Jim también funciona con este programa y avanza lentamente.

No obstante, si queremos darte una tarea:



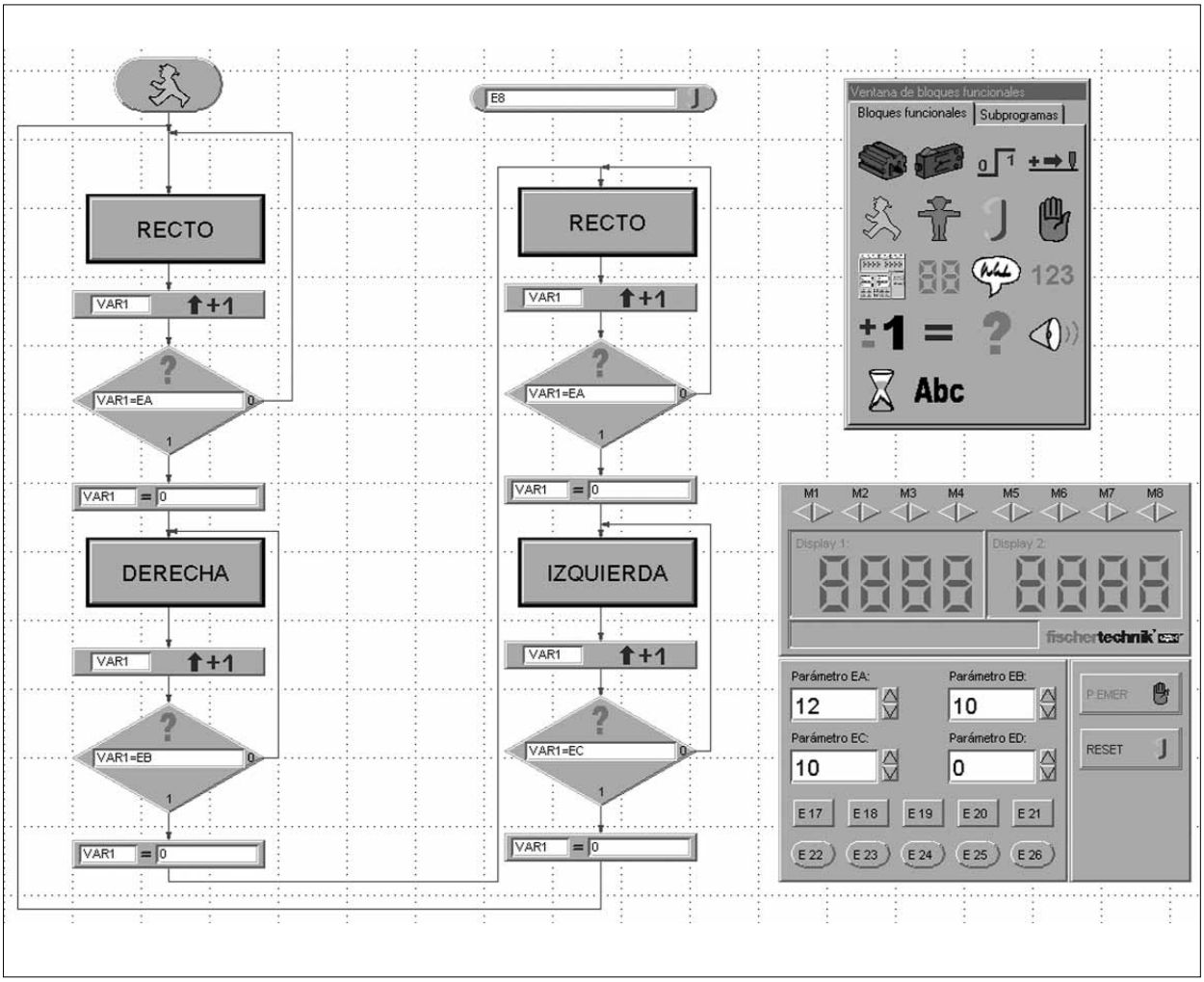
Tarea 1:
Programa Jim de tal manera que avance aprox. 50 cm, luego gire 180° a la derecha, retroceda la misma distancia (adelante), gire 180° a la izquierda, vuelva a andar la misma distancia, etc. Para el número de pasos en línea recta, utiliza el parámetro del terminal EA, para el número de pasos a la izquierda EB y para el número de pasos a la derecha EC. Vuelve a utilizar E8 como pulsador de reset.

Consejos:
Guarda el proyecto JOE.MDL como JIM.MDL. Dentro del proyecto, convierte el programa principal en un subprograma Recto (seleccionar y cortar los bloques funcionales, crear un nuevo subprograma a través de EDITAR - SUBPROGRAMA, insertar los bloques funcionales, agregar SUBIN y SUBOUT, ver el manual de LLWin).

A partir de este subprograma, crea con el comando SUBPROGRAMA - COPIAR los subprogramas IZQUIERDA y DERECHA requeridos.
Modifica en éstos de forma correspondiente el sentido de giro del motor y la interrogación de los sentidos de giro de los motores, y utiliza para cada subprograma otra variable de control para el motor 2.

Seguidamente, programa el programa principal de forma similar a MIKE_BAILE.MDL. Con la diferencia de que debes utilizar los parámetros del terminal ajustables EA-EC para el número de pasos. Tú mismo debes probar cuántos pasos requiere Jim para girar 180° y para avanzar medio metro.

Solución:
A continuación representamos el programa principal. Si es necesario, puedes volver a ver directamente en pantalla los subprogramas. Nosotros también llamamos el proyecto JIM.MDL.



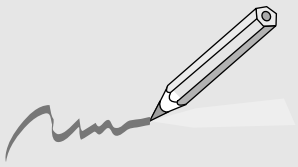
En el proyecto hemos agregado el subprograma HACIA ATRÁS, aunque aquí no se necesite directamente. Pero seguramente querrás que Jim recorra otros caminos. Tal vez para ello tenga que retroceder en un momento dado.

6. Resumen

En tu viaje por el mundo de Robots biónicos de fischertechnik seguramente hayas comprobado que no siempre ha sido fácil poner en marcha los cuatro muchachos. Al fin y al cabo es más difícil moverse con patas que usar ruedas para rodar. Especialmente la programación de la sincronización entre los lados izquierdo y derecho al girar a la izquierda o a la derecha, requiere un poco de concentración. Pero para aquéllos a quienes de todas formas les divierte más construir los modelos, hemos grabado todos los programas en el CD, de modo que cualquiera puede construir y operar estos modelos.

Si eres un programador profesional, seguramente tengas muchas ideas sobre qué tareas pueden programarse para Mike, Jack, Joe o Jim, sea con sensores adicionales para que no caigan de la mesa, o para que se orienten en un laberinto.

Con unos componentes adicionales, también podrás equiparlos con una cabeza, una trompa o una cola. Tu fantasía no tiene límites. ¡Deja volar tu imaginación!



P

1. Bionic – A Natureza como modelo

O conceito Bionic é composto pelos dois conceitos biologia e técnica. Este ramo da ciência, no que respeita a soluções técnicas, tenta sempre se orientar pela Natureza.
Assim, com o objetivo de se deslocar mais longe, mais rapidamente e com maior eficiência do que a Natureza permite, o Homem foi inventando máquinas que, de acordo com os respectivos requisitos, o garantem de modos diversos. Veículos rodam sobre rodas. Em terrenos difíceis, onde os veículos de rodas falham, são utilizados veículos com lagartas. Navios flutuam sobre a água ou têm condições de mergulhar. Em alguns modos de locomoção a Natureza também serve de modelo. Assim, um avião se a assemelha a um pássaro que está planando.

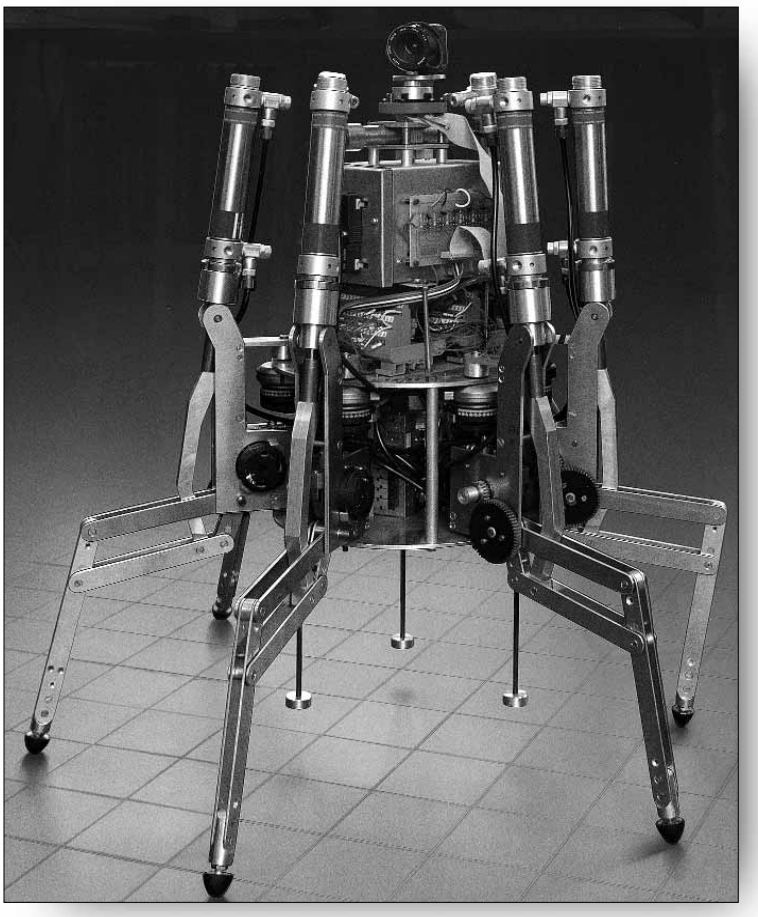
Desde há alguns anos que os cientistas se estão ocupando de uma outra forma de locomoção muito usual na Natureza, que é o andar. Estão sendo desenvolvidos robôs que têm condições de se deslocar sobre pernas. Tais máquinas andantes podem ser utilizadas em situações em que os veículos de rodas ou de lagartas já não têm chances, p. ex. em terreno extremamente inclinado ou instável, superando obstáculos, subindo escadas, atravessando valas, ou em operações em locais de difícil acesso e perigosos em centrais nucleares, minas ou em operações de resgate.

As primeiras experiências sérias no desenvolvimento de máquinas andantes foram feitas em 1967 em uma universidade em Tóquio. Pela primeira vez os insetos serviram de referência, em vez do andar humano. A evolução constante destas experiências conduziu em 1985 à primeira máquina andante bípede. Entretanto estes robôs possuem mais de 50 graus de liberdade e numerosos microprocessadores. Com a ajuda de uma câmera eles podem ler notas de música e tocar órgão. Até podemos bater um papo com eles.

Um exemplo de um robô andante hexápode é o robô eletropneumático „Achille", desenvolvido na Real Academia Militar de Bruxelas.

Equipado de uma câmera em cima e nas seis pernas, este robô deverá reagir mecanicamente a obstáculos elevados ou rebaixados (objetos ou buracos).

Agora a fischertechnik também se dedicou a este tema apaixonante e construiu robôs andantes que ganham vida mediante a Intelligent Interface e o software LLWin.



2. Requisitos e primeiros passos

Para que você possa montar os modelos do kit de computação Bionic Robots, adicionalmente ao kit são ainda necessários os seguintes artigos:

Intelligent Interface, Artigo N° 30402
Software LLWin (a partir da versão 3.0), Artigo N° 30407
Bloco de alimentação Accu Set, Artigo N° 34969

Se você ainda não está familiarizado com o Software LLWin e com a interface, deverá primeiramente ler o manual do software LLWin. Aí está descrita a maneira de instalar o software e de fazer a ligação da interface. Além disso, tratase da melhor forma de obter as primeiras experiências sobre a maneira de comandar modelos da fischertechnik através do PC. Com alguns poucos componentes do kit (motor e botoeira), em uma primeira fase você pode montar comandos para modelos bem simples.

Logo que você esteja familiarizado com o software e a interface, poderá então dedicar-se aos modelos mais complexos Bionic-Robots.

O kit integra um CD-ROM que contém programas-exemplo LLWin para os modelos do kit. Para poder abrir os programas, é necessário o sotfware LLWin a partir da versão 3.0. Você pode deixar os programas-exemplo no CD e chamá-los a partir do LLWin com o comando abrir – arquivo, ou copiar a pasta completa BIONIC_ROBOTS do CD para o diretório de projeto de LLWin no disco rígido e abrir os exemplos a partir de lá.

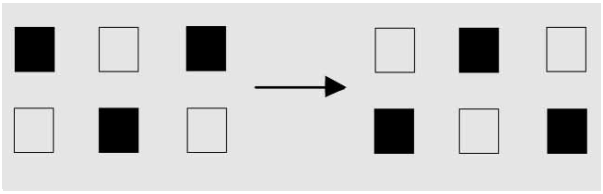
Antes de você montar o modelo, também ainda é necessário montar alguns componentes, p.ex. cabos e conectores. Na instrução de montagem está detalhadamente descrito de que se trata.

Bom, agora chegou o momento. Você pode mergulhar no mundo fascinante dos robôs andantes da fischertechnik. Logo que você tenha terminado o primeiro modelo e ele comece se movimentando como num passo de mágica, você vai ficar maravilhado com esta técnica, que já é usada há milhões de anos na Natureza para a locomoção.

3. Andar sobre 6 pernas

3.1 O andar dos insetos

O andar dos insetos é um excelente modelo do acionamento de hexápodes mecânicos. No chamado andar trípode, três das seis pernas elevam-se sempre ao mesmo tempo do solo, a perna da frente e a perna de trás de um lado juntamente com a perna central do outro lado:

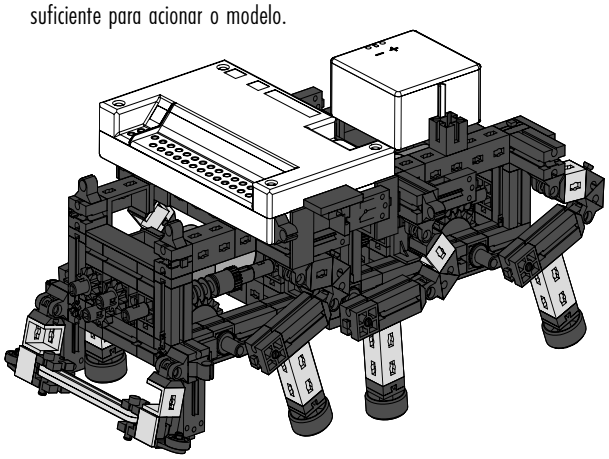


As pernas que estão no solo (representadas a preto) formam um tripé estável, de modo que o modelo está sempre de pé com segurança e não tomba quando anda.

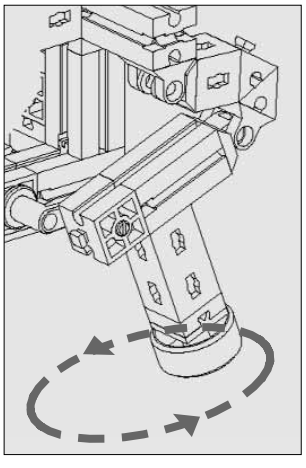
3.2 Modelo Mike

3.2.1 Construção do modelo

Monte agora o hexápode Mike (veja instrução de montagem na pág. xx). Durante a montagem carrregue a bateria, para mais tarde poder ter energia suficiente para acionar o modelo.



As pernas do modelo estão construídas de tal modo, que daí resulta uma engrenagem de quatro articulações. O tipo de construção da articulação quádrupla aqui utilizada tem o nome de „mecanismo de biela e manivela. Acionados por uma manivela, os elementos da engrenagem, apoiados de maneira móvel, executam movimentos oscilantes. As distâncias entre cada uma das articulações a posição do pé (isso é a extremidade inferior da perna) foram scolhidas de tal modo que o pé descreve um movimento elíptico sempre que a manivela de ecionamento gira. Daí resulta um movimento semelhante a um passo quando se anda. As 6 manivelas que acionam as pernas têm que ser ajustadas precisamente como está indicado na instrução de montagem. As três pernas que, simultaneamente, estão apoiadas no solo possuem a mesma posição de manivelas. As manivelas das 3 pernas, que neste momento se encontram no ar, estão torcidas de 180°. A posição correta das manivelas entre si garante que o modelo possa andar com a seqüência de passos correta, ou seja, andar trípode.



As porcas de pinça e de cubo, com as quais se faz a fixação dos parafusos sem-fim e das rodas dentadas nos eixos, têm que ficar bem apertadas para que, durante o andar, as manivelas não fiquem desajustadas.

O lado direito e o lado esquerdo do motor são acionados por um motor, respectivamente (isso é necessário para descrever as curvas). Por isso é necessário que a perna central de um lado esteja sempre na mesma posição que as duas pernas exteriores do outro lado. Esta sincronização é comandada mediante software através das boteiras E1 e E2.

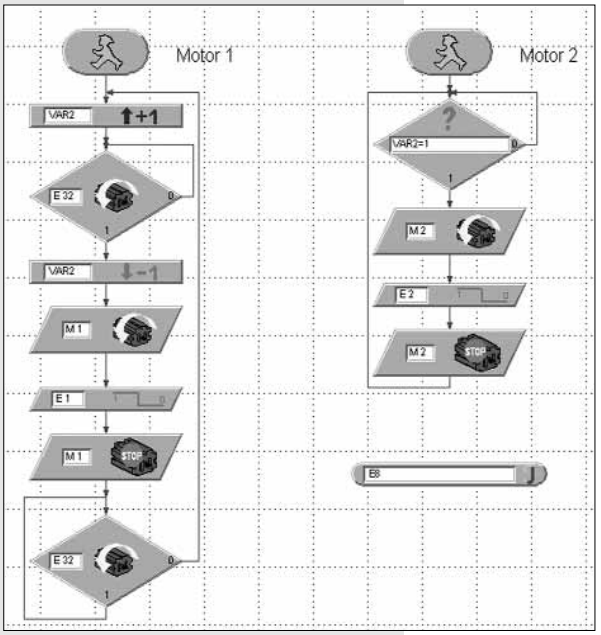
Utilize a diagnose de interfaces para testar se todas as botoeiras e todos os motores estão conectados corretamente. Sentido de rotação dos motores: sentido de rotação esquerdo=para a frente

3.2.2 O primeiro programa

Agora vamos começar a ensinar algumas coisas ao Mike. Primeiramente o modelo só deverá andar em frente. Mais tarde vamos tratar das curvas e da reação aos obstáculos.

Tarefa 1:
Programe o modelo de modo que ele ande em frente com andar trípode. Utilize as botoeiras E1 e E2 para fazer a sincronização da perna esquerda e da perna direita. Preste atenção ao fato de que as duas pernas exteriores de um lado e a perna central do outro lado têm de estar na mesma posição. Além disso, utilize a botoeira E8 como botoeira de reset.

Dicas:
Programe para cada motor um processo próprio. Comande o processo para o motor M2 com a ajuda de uma variável VAR2. Se você durante a programação não necessitar a interface, interrompa aligação de corrente ntre a bateria e a interface para economizar energia.



Solução:
O programa para marcha em frente tem o seguinte aspecto:

P

A variável VAR2 gera o impulso que dá partida ao motor M2. Depois é dada partida ao motor M1. Logo que a botoeira E1 seja acionada, M1 pára. Logo que a botoeira E2 seja acionada, M2 pára. O primeiro processo espera até que M2 tenha parado (o estado do motor M2 é consultado através de E32); vide também „Consulta ao estado do motor" no manual do LLWin).

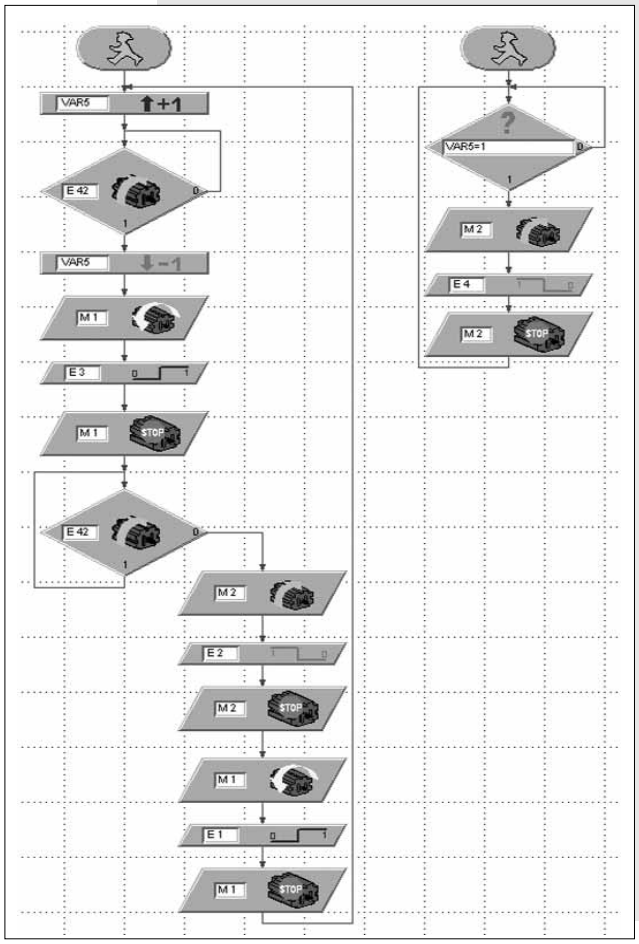
Se você não tiver paciência para criar este processo, ele encontra-se no CD anexo como projeto-exemplo MIKE_EMFRE.MDL.
Inicie o projeto. Se você tiver programado tudo certinho, agora o modelo se enche de vida e marcha em frente. Os nossos parabéns. Já foi dado o primeiro passo.

3.2.3 A rotação para a esquerda

É claro que ainda não é suficiente o Mike só andar em frente. Seguidamente queremos que ele rode sem sair do lugar.

Tarefa 2:
Programe o Mike de modo que ele rode para a esquerda.

Dicas:
O modelo roda para a esquerda, quando M1 roda para a esquerda e M2 para a direita.
Você também pode operar o modelo sem sincronização. Ele também roda, no entanto existem posições em que o modelo tomba para a frente. Isso pode ser evitado. Mais precisamente através do seguinte processo:



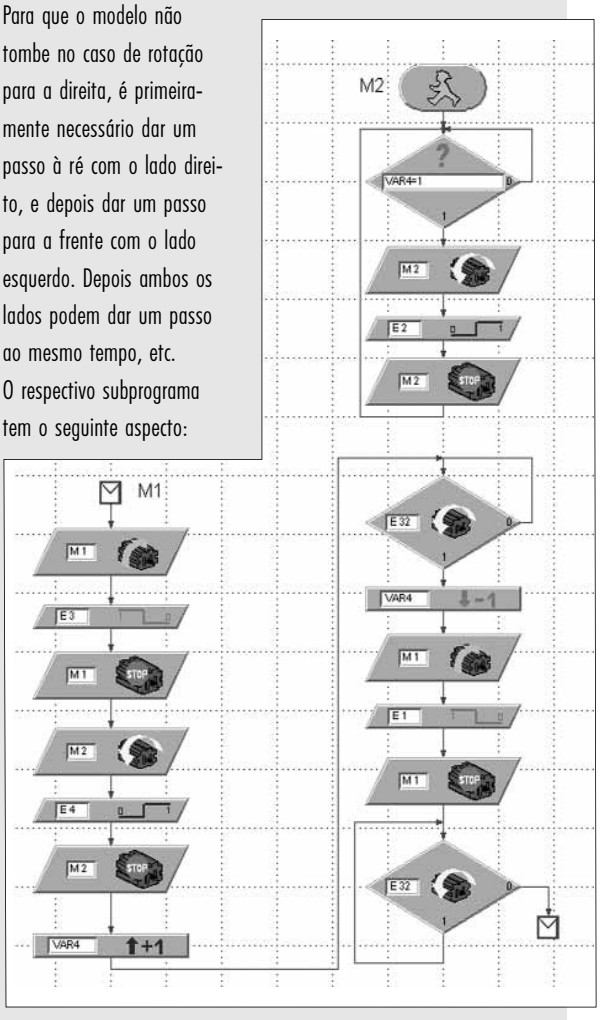
Com a ajuda das botoeiras E1-E4, o lado esquerdo e o lado direito do modelo dão primeiramente um passo simultâneo, depois o lado esquerdo dá um passo, em seguida o lado direito, etc. Desta maneira o modelo nunca tomba para a frente. Experimente! Assim você também terá mais facilidade em entender esta seqüência.
Você também encontra este processo como projeto MIKE_ESQ.MDL no CD.

Agora o modelo pode andar em frente e rodar para a esquerda. Ainda falta a marcha à ré e a rotação para a direita. A marcha à ré funciona como a marcha para a frente, só que o motor roda no sentido inverso. A rotação para a direita funciona de maneira inversa da rotação para a esquerda.

3.2.4 Esquerda, direita, em frente, à ré

Tarefa 3:
Programe agora cada uma das funções EM FRENTE, À RÉ, ESQUERDA e DIREITA COMO subprograma, para que, mais tarde, os possa utilizar de modo flexível em diversos projetos.

Dicas:
No manual do LLWin está descrito como é possível copiar um processo existente para um subprograma.
Utilize em cada subprograma outras variáveis (VAR2-VAR35) para iniciar o processo para o motor M2.
Para que o modelo não tombe no caso de rotação para a direita, é primeiramente necessário dar um passo à ré com o lado direito, e depois dar um passo para a frente com o lado esquerdo. Depois ambos os lados podem dar um passo ao mesmo tempo, etc.
O respectivo subprograma tem o seguinte aspecto:

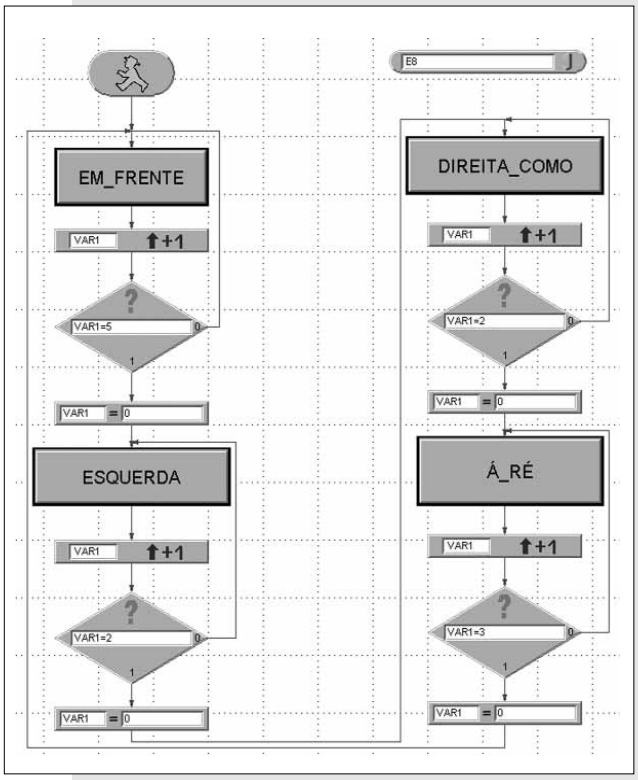


Os outros subprogramas não são aqui mostrados. Caso você tenha dificuldades ao programar um processo, os subprogramas prontos estão no arquivo MIKE_MODELO.MDL no CD. O programa principal deste projecto está vazio. Na janela de módulo abaixo do marcador „subprogramas" você encontra a lista com os subprogramas existentes, que poderá inserir no programa principal.

Mas não vá já ver como a coisa funciona. Primeiro, tente você próprio encontrar a solução. Se não conseguir, então poderá dar uma olhadinha.
Para experimentar todos os subprogramas, agora queremos que o Mike dance.

Tarefa 4:
Programe o Mike de modo que ele dê 5 passos para a frente, rode 2 passos para a esquerda, depois 2 passos para a direita, em seguida 3 passos à ré e depois comece do início. Utilize a variável VAR1 como variável de contagem para o número de passos. Utilize E8 como botoeira reset.

Solução:



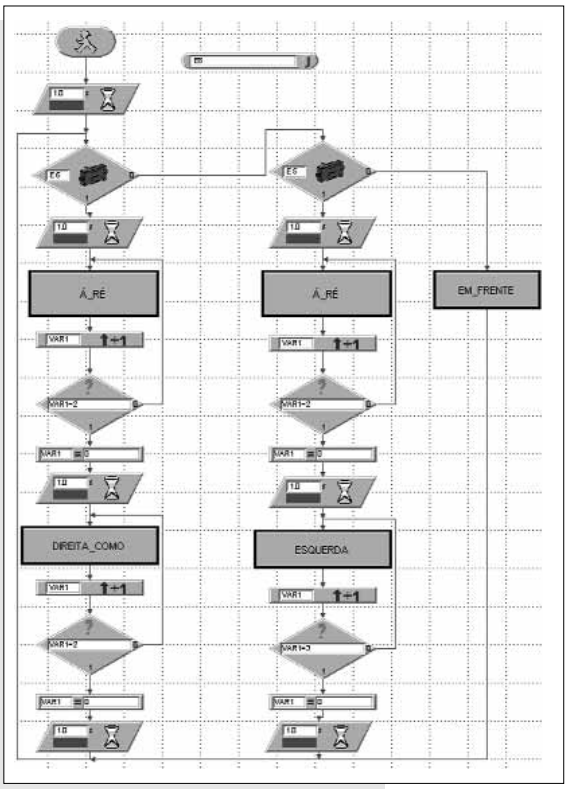
Este projeto tem o nome MIKE_DANCA.MDL.

3.2.5 Reconhecer obstáculos

Por último queremos que o Mike, com o seu pára-choque móvel (é melhor chamar-lhe „sensor") reconheça obstáculos e se desvie deles.

Tarefa 5a:
Programe o Mike de maneira que quando houver um obstáculo no sensor esquerdo (botoeira E6), ele dê primeiro 4 passos à ré e depois se desvie, dando 2 passos para a direita. Caso se encontre um obstáculo em seu sensor direito (botoeira E5), ele deverá recuar 4 passos e depois desviar-se 3 passos para a esquerda.

Solução:



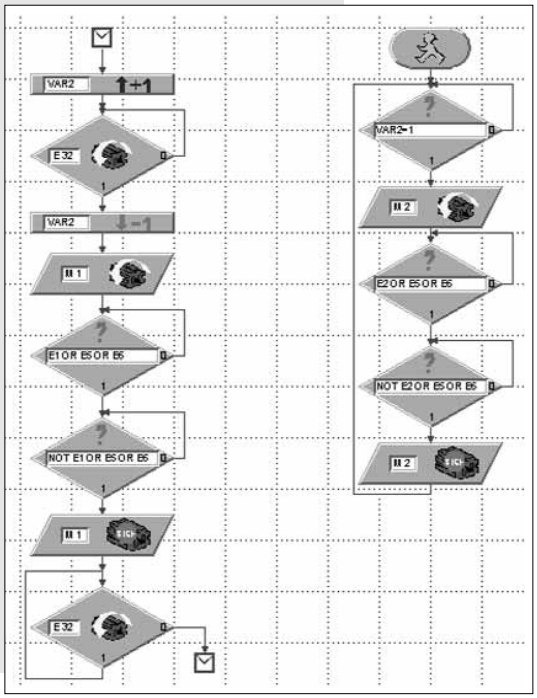
O Mike marcha sempre em frente. Após cada passo são consultadas as botoeiras E5 e E6. Se E6 estiver apertada, o programa ramifica para o processo esquerdo (primeiro à ré, depois para a direita). Se E5 estiver apertada, passa para o processo central (primeiro à ré, depois para a esquerda).

Dado que as botoeiras E5 e E6 só são consultadas após cada passo completo, demora relativamente bastante tempo até o Mike reagir a um obstáculo.

Tarefa 5b:
Otimize o subprograma EM FRENTE, de modo que o Mike possa reagir mais rapidamente a um obstáculo.

Dica:
Para consultar as botoeiras E1 e E2, não utilize o módulo FLANCO e sim o módulo COMPARAÇÃO. Pergunte adicionalmente se E5 ou E6 está apertada.

Solução:



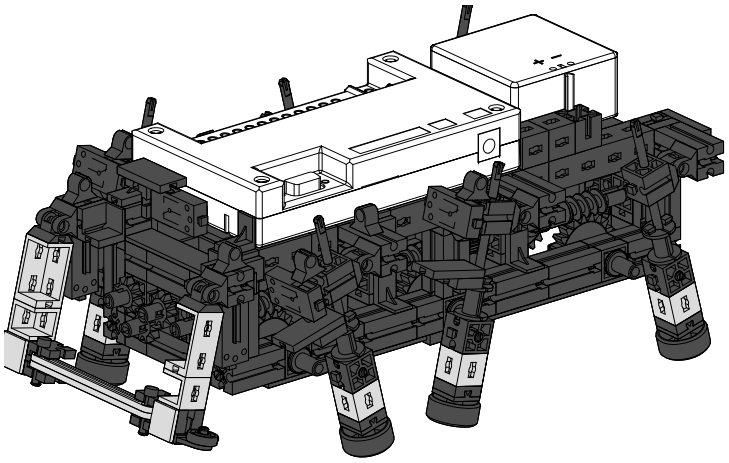
P

Agora o Mike deveria funcionar perfeitamente. Este programa também se encontra no CD em MIKE_OBSTACULO.MDL. Você também pode incorporar o subprograma melhorado no projeto MIKE_MODELO.MDL. Se em um outro programa E5 e E6 não forem consultados, isso não tem importância nenhuma. Este modelo melhorado está armazenado em MIKE_MODELO_ OBSTACULO.MDL.

Depois de termos abordado detalhadamente o primeiro hexápode, vamo-nos dedicar ao segundo modelo, que também tem 6 pernas. Vamos chamá-lo de Jack.

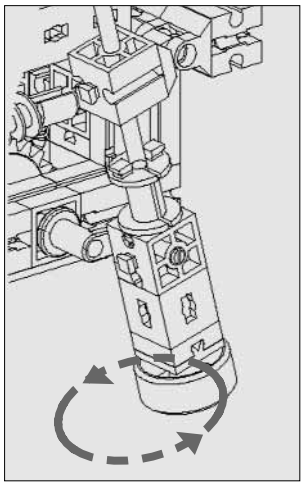
3.3 Modelo Jack

O Jack também pertence à espécie dos modelos de seis pernas da fischertechnik. Porém, na construção de suas pernas ele é bem diferente do Mike. Monte o modelo como descrito na instrução de montagem a partir da pág. xx. Os níveis de montagem 1-13 são idênticos, tanto no Mike quanto no Jack. Você não precisa de desmontar o Mike completamente antes de começar com a montagem do Jack.



3.3.1 A construção

A construção das pernas do Jack também é uma chamada engrenagem de quatro articulações. O tipo de construção aqui utilizado é chamado de manivela deslizante oscilante". O puxavante está apoiado sobre uma guia longitudinal móvel, que oscila para a frente e para trás quando a manivela gira. A curva descrita pelo pé da perna não tem uma forma tão elíptica como no caso do modelo Mike, sendo mais um círculo.



Por isso, durante a marcha, o corpo do Jack se eleva e se abaixa mais acentuadamente em comparação com o corpo do Jack. Os passos são mais curtos. Mas em contrapartida o Jack pode transpor pequenos obstáculos, coisa que o Mike não consegue. Além disso, esta forma construtiva da engrenagem faz mais lembrar uma perna, o que não é bem o caso do Mike. Quando o Jack caminha, dá a impressão que ele está andando sobre andas.
Ele também tem o andar trípode dos insetos. Também neste modelo é importante ajustar as manivelas tal como descrito na instrução de montagem e apertar bem as porcas de pinça e de cubo.

3.3.2 A programação

Seria lógico que para o Jack também se pudessem utilizar os mesmos programas, com os quais o Mike funciona. Tente!

Tarefa 1:
Opera o Jack com o programa MIKE_OBSTACULO.MDL. O que você está vendo?

Observação:
O modelo marcha perfeitamente para a frente e para a ré. Mas ao rodar para a esquerda e para a direita, ele tomba para a frente.

Tarefa 2:
Como é que você explica isso?

Solução:
Ambos os modelos possuem diferentes construções das pernas. Também as botoeiras E1-E4 são acionadas em uma outra posição das pernas. A maneira como o Mike gira não significa que se possa aplicar ao Jack. – Azar.

É claro que a gente não está gostando disso e queremos arranjar uma solução o mais rápido possível.

Tarefa 3:
Tente programar o Jack de modo que, tal como Mike, ele reconheça obstáculos mas que não tombe para a frente quando rodar.

Dicas:
Armazene o projeto MIKE_OBSTACULO.MDL sob o nome de JACK_OBSTACULO.MDL e proceda às necessárias alterações.

Quando o Jack roda, os dois motores podem funcionar ao mesmo tempo. Não existe o ligar e desligar alternados. Decisivo é que no início da rotação, ou seja, após a marcha à ré, as pernas se encontrem na posição de partida correta.

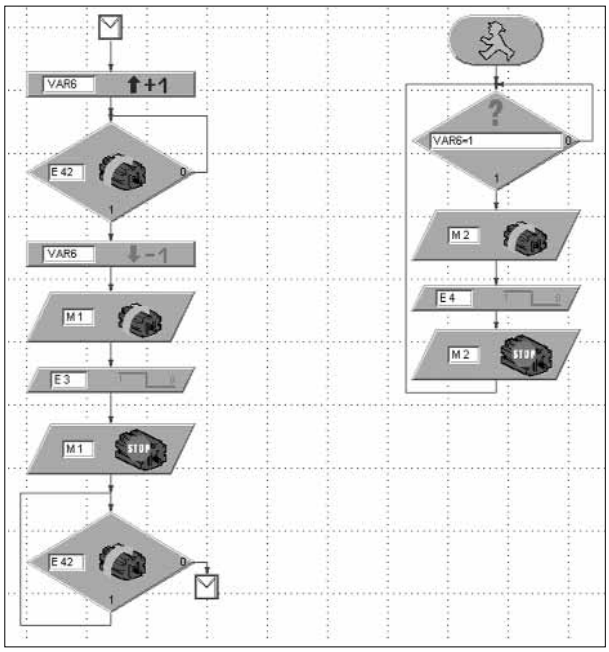
Rotação para a esquerda:
Se pretendermos que o modelo rode para a esquerda, no início da rotação a manivela da perna esquerda da frente tem de estar orientada para trás e a manivela da perna direita da frente tem de estar orientada para a frente. Este é o caso quando na marcha à ré, no lado esquerdo foi acionada e largada a botoeira E2 e no lado direito foi acionada e largada a botoeira E1.

Rotação para a direita:
Se pretendermos que o modelo rode para a direita, no início da rotação a manivela da perna esquerda da frente tem de estar orientada para a frente e a manivela da perna direita da frente tem de estar orientada para trás. Este é o caso quando na marcha à ré, no lado esquerdo foi acionada e largada a botoeira E4 e no lado direito foi acionada e largada a botoeira E3.

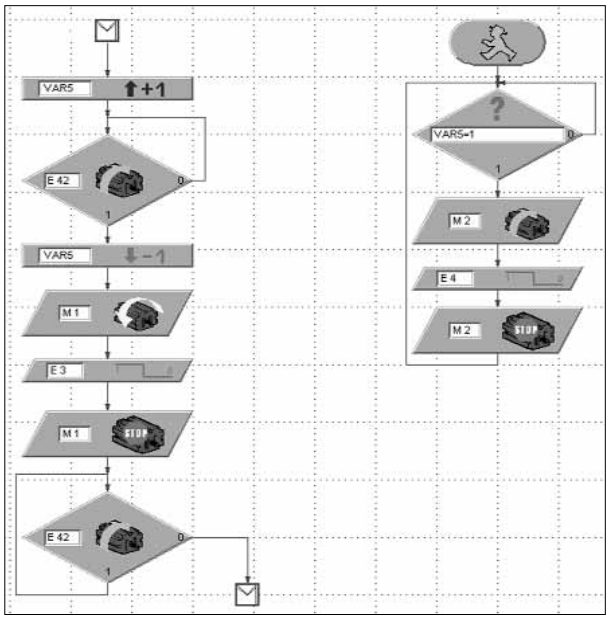
Bastante complicado, não é? Mas não se preocupe, já vamos arranjar uma solução:

para o movimento à ré é necessário utilizar dois subprogramas diferentes. Caso o modelo deva rodar para a esquerda, sincronize os passos na marcha à ré com as botoeiras E1 e E2. Isso corresponde ao subprograma RÉ do projeto MIKE_OBSTACULO.MDL.
Caso o modelo deva rodar para a direita, os passos são sincronizados para a ré com E3 e E4.

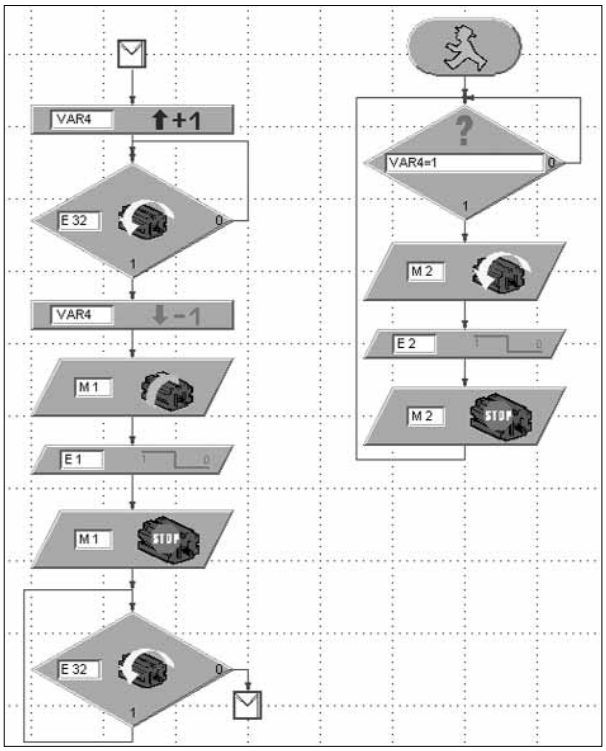
Você dá o nome de RÉ ao subprograma e depois com o comando RENOMEAR SUBPRORAMA você altera o nome para RE_ESQ. Depois, com COPIAR SUBPROGRAMA, você faz cópia para um segundo subprograma RE_DIR.
Aí você altera as designações das teclas para a sincronização para E3 e E4. Para RE_DIR também não esqueça de utilizar uma nova variável VAR6 para a sincronização, senão teremos o caos total. RE_DIR tem então o seguinte aspecto:



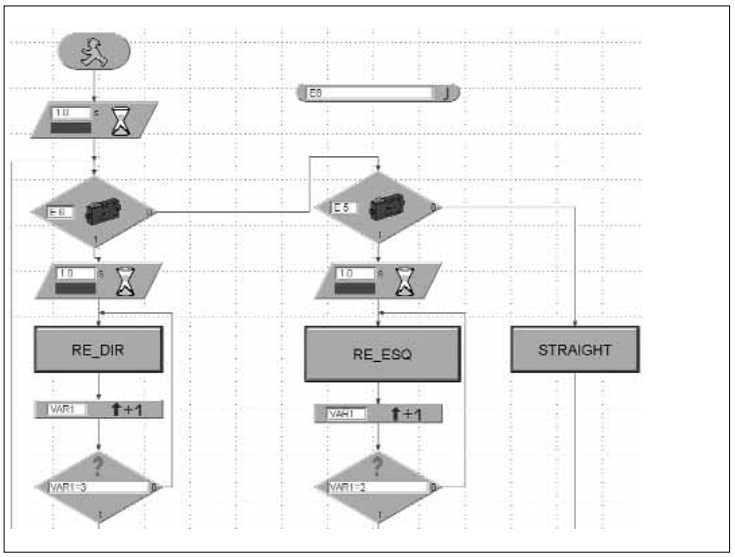
Agora ainda é necessário alterar os subprogramas para a rotação propriamente dita, de modo que os dois motores rodem sempre simultaneamente. O subprograma ESQUERDA é composto dos seguintes módulos:



Você está vendo que, em relação ao subprograma no projeto MIKE_OBSTACULO.MDL , podem não existir alguns módulos.
O subprograma para o sentido de rotação direito é semelhante, só que com outros sentidos de rotação do motor. Além disso, as botoeiras E1 e E2 são utilizadas para a sincronização de ambos os motores:



Por último, no programa principal a derivação para o desvio para direita, você substitui o subprograma RE_ESQ por RE_DIR:



O resto do programa principal permanece inalterado.

Você agora conseguiu! Se você não fez nenhum erro, então agora o Jack poderá caminhar sem tombar quando roda. Caso alguma coisa não esteja funcionando e você não saiba por quê, não se preocupe, pois também foi um osso duro de roer. De qualquer modo você tem a possibilidade de chamar do CD o projeto pronto JACK_OBSTACULO.MDL e de operar o modelo com ele.
Se você teve condições de solucionar o problema, então pode estar orgulhoso, pois a partir de agora você é um profissional entre os programadores.

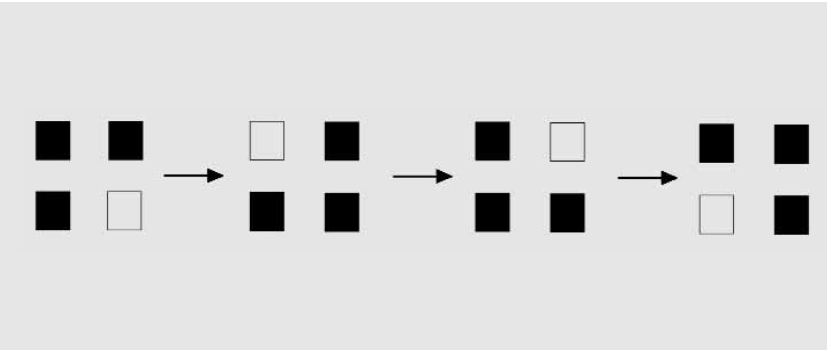
P

4. Andar sobre 4 pernas

4.1 Andaduras dos mamíferos

Para construir um robô andante com 4 pernas, voltamos a utilizar a Natureza como modelo e vamos ver quais as andaduras que os mamíferos utilizam para se locomover.

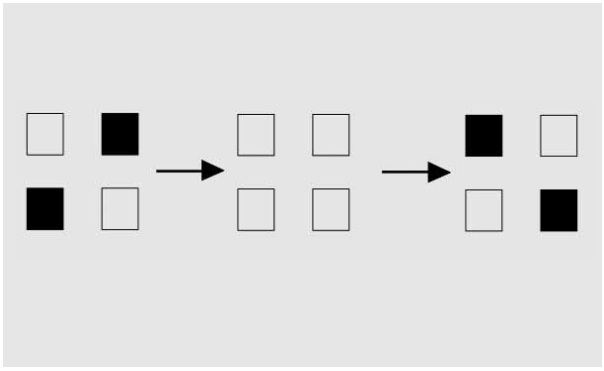
A andadura mais lenta e mais segura é o passo. Uma perna procura uma nova posição enquanto o corpo do animal se apóia em três pernas. Os animais se movimentam em marcha cruzada com a seguinte seqüência de passos: perna da frente direita, perna de trás esquerda, perna da frente esquerda, perna de trás direita para a frente.



As superfícies pretas representam as pernas com estão em contato com o solo, as superfícies brancas representam a perna levantada.

Se pretendermos utilizar esta andadura em um robô, é necessário levar em consideração o seguinte:
Imagine que você serra uma perna de uma mesa com 4 pernas. O que acontece? Correto, a mesa tomba. Isso significa que as três pernas já não formam um tripé estável, como era o caso dos hexápodes. Isso dificulta a construção de um robô com quatro pernas.

Quanto mais rápida for a locomoção dos mamíferos, mais insegura será a sua andadura. Vamos agora observar a andadura trote. No trote, as pernas são elevadas na diagonal com sincronismo. Mas antes de elas tocarem o solo, as outras duas pernas já estão levantando. Isso significa que, por alguns momentos se perde completamente o contato com o solo.

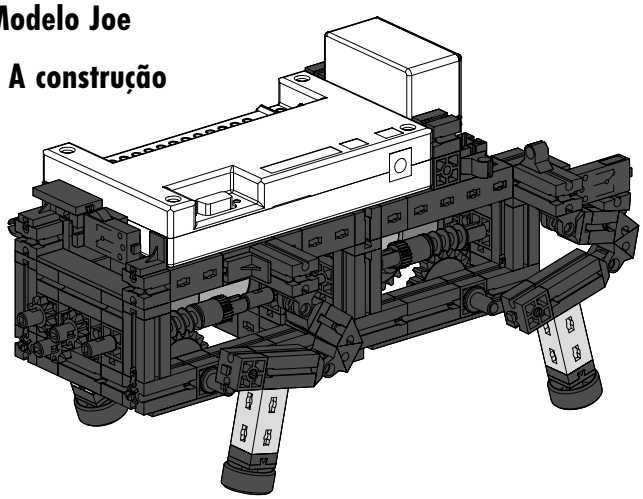


Você pode imaginar que uma andadura em que, por vezes, se perde o contato com o solo, não é muito apropriada para um modelo da fischertechnik, no qual se encontram a interface e a bateria.

Vamos então tentar com a andadura passo.

4.2 Modelo Joe

4.2.1 A construção



Monte o modelo como descrito na instrução de montagem a partir da pág. 20.
A construção das pernas é idêntica à do Mike. No caso do Joe, a posição das manivelas que acionam as pernas tem que ser completamente diferente. As manivelas estão desalinhadas entre si de 90°, respectivamente. Você tem que ajustar precisamente como está indicado na instrução de montagem.
A sincronização do lado esquerdo e do lado direito volta a ser feita através das duas botoeiras E1 e E2. Assim obtemos a seqüência de passos necessária.
Para que o modelo não tombe sempre que uma perna é levantada, o centro de gravidade do modelo deverá estar situado de modo que o modelo tombe no momento certo e seja recuperado pela perna não sobrecarregada.

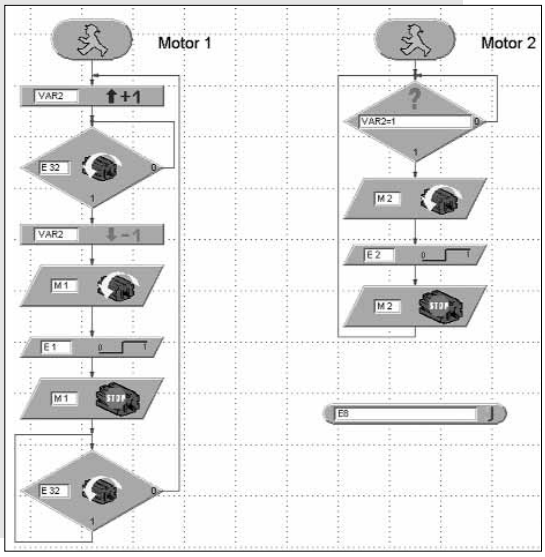
4.2.2 A programação

Neste modelo vamo-nos contentar com a marcha em frente.

Tarefa 1:
programe o Joe de modo que ele se movimente para a frente na andadura „passo".

Dicas:
Utilize para cada motor um processo separado e sincronize ambos os lados com as botoeiras E1 e E2.
Utilize E8 como botoeira reset.

Solução:



fischertechnik®

Neste programa utilizamos o flanco 0-1 das botoeiras para a sincronização. No momento em que as botoeiras estão apertadas, as manivelas das 4 pernas têm a posição correta entre si. Damos ao projeto o nome de JOE.MDL.

Você vê que o Joe se movimenta com mais dificuldade que o Mike e o Jack. Devido ao necessário deslocamento do peso, o corpo oscila bastante e a maneira de andar não é tão elegante como no caso dos hexápodes.

Se você ainda tiver vontade de ensinar o modelo a descrever curvas, tente e veja se vai conseguir. Muito sucesso.

5. Andar sobre duas pernas

5.1 Andante bípede

O andar sobre duas pernas não surgiu na espécie dos mamíferos e também é praticado em algumas espécies de répteis. Varanos, iguanas, agamas e lagartos-corredores utilizam unicamente as pernas traseiras quando fogem. Assim eles conseguem atingir grandes larguras de passo, sendo extremamente rápidos. Para isso eles precisam fortes pernas traseiras, uma longa cauda balanceadora e um terreno plano.

Os pássaros também fazem parte dos bípedes. A avestruz faz parte das aves cursoras mais rápidas. Ela atinge velocidades constantes de até 60 km/h.

Mas o bípede mais perfeito é o ser humano. O andar completamente ereto exige a extensão da articulação da anca. Isso é proporcionado pelo grande músculo das nádegas. Além disso, as pernas podem engatar na rótula, ficando fixadas em uma posição que consome pouca energia.

A locomoção sobre duas pernas representa a mais difícil de todas as andaduras, porque além dos requisitos anatômicos citados, ela também exige um sentido de equilíbrio muito desenvolvido. Para nós, seres humanos, andar sobre duas pernas parece-nos ser a coisa mais natural e mais simples. Mas se tivermos presente que quando levantamos uma perna todo o corpo fica apoiado sobre a outra perna e tem que ser de tal modo equilibrado, então temos que reconhecer que precisamente manter o equilíbrio torna este tipo de locomoção tão complicado. Mesmo um recém-nascido não tem condições de começar a andar imediatamente sobre duas pernas. Primeiro ele engatinha „de quatro", antes de ficar ereto e de aprender a andar.

Na Universidade de Waseda em Tóquio já foram desenvolvidos robôs bípedes que se movimentam com a ajuda de numerosas articulações, diferentes sensores, câmeras e potentes microprocessadores, e que mantêm o equilíbrio mediante o deslocamento do peso.

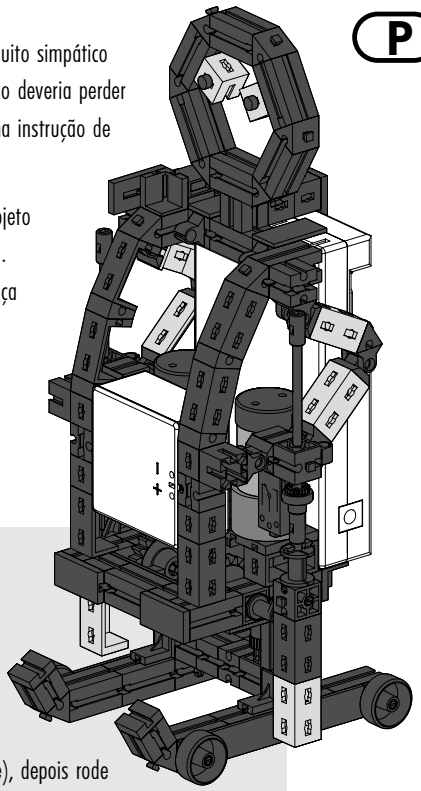
Para o nosso kit Bionic Robots isso seria muito dispendioso e muito complicado. Já vimos que andando com quatro pernas com um modelo da fischertechnik estamos atingindo o nosso limite.

5.2 Modelo Jim

Mas para que este capítulo não seja somente tratado de um modo teórico, agora no final tomamos a liberdade de construir, pelo menos, um bípede esquiador que tem o nome de Jim. Ele não tem grande semelhança com um andante bípede, mas é um camarada muito simpático e se esforça muito em avançar. Você não deveria perder esta curtição. Você encontra o modelo na instrução de montagem, na pág. xx.

Como programa você pode utilizar o projeto JOE.MDL. Não é necessário alterar nada. O Jim também funciona com ele e avança lentamente.

Mas ainda queremos que você cumpra uma tarefa:



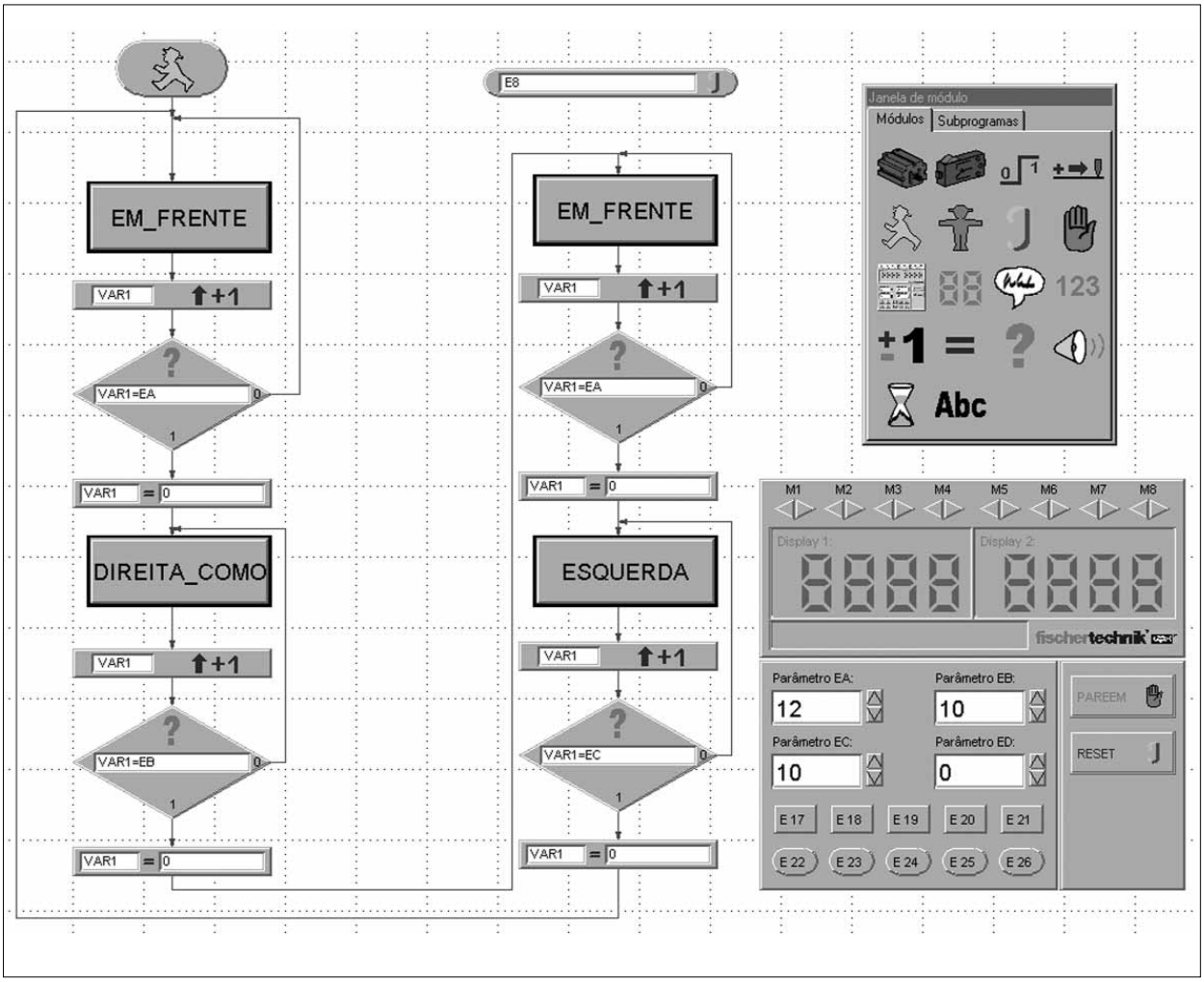
Tarefa 1:
programe o Jim de modo que ele avance aprox. 50 cm, depois rode para a direita 180°, regresse pelo mesmo percurso (para a frente), depois rode 180° para a esquerda, percorra o mesmo percurso, etc. Para o número de passos em frente utilize o parâmetro de terminal EA, para o número de passos para a esquerda EB e para direita EC. Utilize de novo E8 como botoeira reset.

Dicas:
Armazene o projeto JOE.MDL sob o nome de JIM.MDL. Transforme o programa principal em um subprograma „em frente" (Marcar e recortar módulos, mediante EDITAR SUBPROGRAMA criar um novo subprograma criar, inserir módulos, completar SUBIN e SUBOUT, veja também o manual do LLWin).

A partir deste subprograma e com o comando COPIAR SUBPROGRAMA, criar os subprogramas ESQUERDA e DIREITA necessários. Altere o sentido de rotação do motor e a consulta do sentido de rotação do motor de modo correspondente e para cada subprograma uma outra variável de comando para o motor 2.

Depois você programa o programa principal de modo idêntico ao que fez em MIKE_DANCA.MDL. Só que para o número de passos você vai utilizar os parâmetros ajustáveis de terminal EA-EC. Você tem que experimentar quantos passos o Jim necessita para rodar de 180° ou para poder avançar.

Solução:
Seguidamente vamos ilustrar o programa principal. Se necessário, os subprogramas podem ser vistos diretamente no monitor.
O projeto também tem o nome de JIM.MDL.



No projeto completamos ainda o subprograma RÉ, se bem que ele aqui não seja necessário diretamente. Mas certamente você pretende que o Jim vá por outros caminhos. É possível que ele também tenha de andar à ré.

6. Resumo

Na viagem que você acaba de fazer através do mundo dos Bionic Robots da fischertechnik, certamente que constatou que nem sempre foi fácil colocar os quatro camaradas a andar. É bem mais difícil locomover-se sobre pernas do que sobre rodas. Especialmente a programação da sincronização entre o lado esquerdo e o lado direito quando da rotação para a esquerda ou para a direita exige um pouco de concentração. Para todos aqueles que curtem mais a montagem dos modelos, nós gravamos todos os programas no CD, de modo que cada um de vocês pode montar e operar os modelos.

Caso você faça parte dos programadores profissionais, decerto que tem muitas idéias sobre as tarefas que ainda podem ser programadas para o Mike, Jack, Joe ou Jim, quer seja com sensores adicionais para que eles não caiam da mesa, quer seja para eles descobrirem a saída de um labirinto.

Com componentes adicionais você também os pode equipar com uma cabeça, uma tromba ou uma cauda. A sua fantasia não tem limites. Puxe pela cabeça!

Bionic Robots

- Begleitheft
- Activity booklet
- Manuel d'accompagnement
- Begeleidend boekje
- 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: fischertechnik@fischerwerke.de
http://www.fischertechnik.de

fischertechnik®

62 965 • ZE • 11/01 • Printed in Germany • Technische Änderungen vorbehalten • Subject to technical modifications