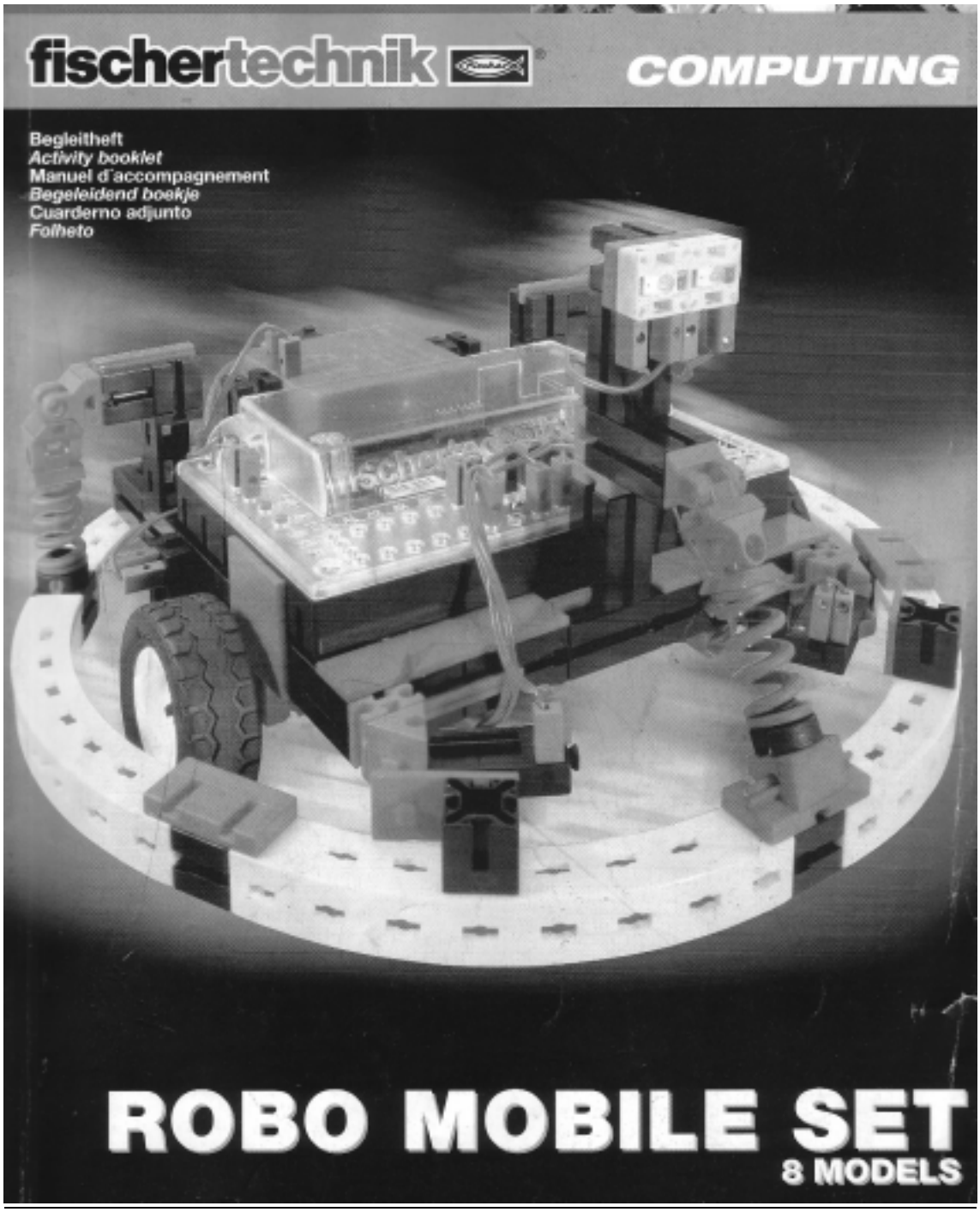
ROBO 自走・歩行ロボットキット
マニュアル



＜　目　　次　＞

＜はじめに＞

なぜロボットが必要なのでしょうか？

■「ロボット」の語源は旧チェコスロバキアの劇作家カレル・チャペックが１９２０年に書いた劇曲「ロッサム万能ロボット製造会社」の中で、チェコ語で「労働や苦役」を意味する「ROBOTA」から人造人間を指す「ロボット」を造語し、登場させたことに由来しています。

１９３０年代から４０年代にかけて「ロボット」は自動装置の一種となってきました。
そのロボットに、人間的な外面的特徴を与えようとする様々な試み（例えば、瞬く目を顔につけたり、スピーカーを通して声を出したり）が行われましたがこれらは今日で見ればばかばかしく思われるものがほとんどでした。
その時代では、人間を超えたロボットによる人間支配への恐怖は話題になりましたがそれは現実的なものではありませんでした。

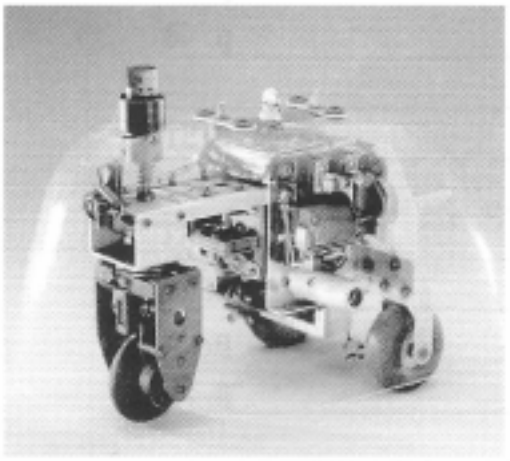
制御技術の発達がロボットにも大きな影響を与え、ロボットのデザインも、より現実的なものになってきております。また今日では、ロボットの知能すなわち人工知能については、多くの研究所や大学での重要な研究テーマとして取り上げられています。

■ いわゆる人工知能学（cybernetics）ですが、その語源は、ギリシア語の「Kybernete」に由来しています。「Kybernetes」とはギリシア時代の船の航海者を意味しています。航海者は船の位置を測定し、進むべきコースを計算する能力を必要としていました。
そこでロボットの知能を人工知能に置き換えて話しを進めますとまず人工知能をもったロボットとは具体的にどのようなものになるのでしょうか。
皆さんは、街灯に群がる蛾の行動や習性を見たことがあるでしょう。
蛾は光を見つけると、その方向へ飛び、ぶつからないように光源の手前で上手に飛び続けています。蛾がそのような行動をする為には、「光源を見つけ」、「進路を決め」、「その方向に飛ぶ」能力が必要であり、蛾には既に備わっているということが分かります。実際、これらの能力は昆虫の本能による行動パターンともいえるでしょう。

では、これらの能力を、人工知能を作るための技術システムに転換してみましょう。
技術的にはまず、光センサーを使って光源を見つけ（センサー)、モーター等の動力装置を使って移動し（モーターコントロール)、進路及び移動距離との関係を

計算（プログラミング）するシステムが必要です。

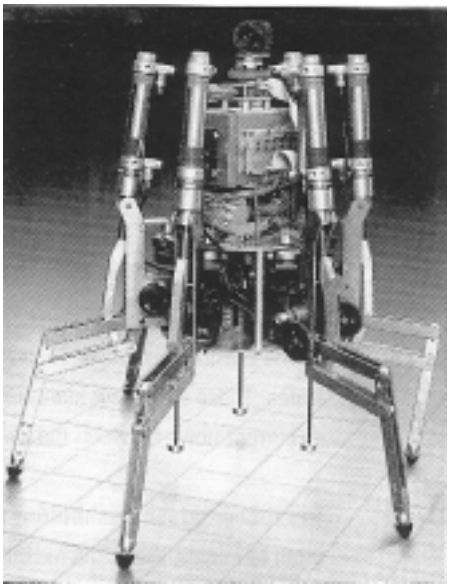
■実は、ウォルター・グレイというイギリス人が１９５０年代に上記の技術を既に構築し実験をしていました。彼は単純なセンサーとモーターと電子回路を用い、



いくつかの人工知能を持つ動物（ロボット）を開発していたのです。
それらのロボットは正に蛾が本来持つ特有の動きを行うことが可能でした。
ロボットにはまず色々なセンサー（光センサーなど）を付け、出力側（モーター、
リレー、ランプなど）と共に電子回路で制御し、
その結果（見たところ）知能を持った
動きを行うことが出来たのです。
左の写真はアメリカのスミソニアン博物館に
展示されているそのレプリカです。
こられの考えを基にして同じような行動を取るロボットを作ってみましょう。

■では、ここでもう一ロボットの必要性を考えてみましょう。
私たちが想像した蛾の行動をロボットに当てはめて見ます。まず光追捕という単純な行動を考えて見ます。光源の代わりに床面に明るいテープを張り、センサーを先端ではなく
下部に取り付けます。こうすれば例えば倉庫内で、自走型ロボットは方位決定する為の
情報（光）を得る事が出来ます。
しかも、バーコードなどを利用すればパレットへの荷物の積み下ろしなどの作業も
併せてさせることが出来ます。
実際そういうロボットシステムは既に存在しています。
大きな病院では、シーツなどの交換の為にとても長い距離を移動しなければならない
ことがあります。それらの作業は看護婦さん達によって行われていますが、
時間も労力も大きく消耗してしまいます。そうなれば、当然患者さん達へ費やされる
時間なども減ってしまいます。そういった労働をなくすためにも、自走型ロボットが
現代社会では重要となってくるのです。
表題の答えはこの点だけでも十分ではないでしょうか？

■過去数十年にわたり科学者たちは違った形での走行について研究してきました。
それは自然界にはありきたりな「歩行走行」です。
最近ではロボットも歩行走行できるようになって来ました。



左の写真はブリュッセルの王立軍事研究所が開発した歩行ロボット「ACHILLE」で電子式空気圧利用の六足歩行ロボットです。
各足および頭部にカメラを搭載していて障害物を避けて歩行できるようになっています。
このような歩行ロボットは従来の車輪式やキャタピラー式では走行できないような荒地や柔らかな土壌、あるいは障害物を乗り越える必要のある所、原子力発電所のような危険な所での操作や作業などに極めて有効です。

現代社会においてロボットの果たす役割の重要性は十分に認識できるかと思います。

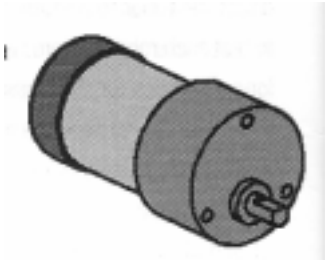
第１章　フィッシャーテクニックのロボット

このキットでは新しく開発されたプログラミングソフト Robopro および
新インターフェイスを使って自走ロボット・歩行ロボットが組み立てられるキットです。
八種類のロボットを組立ながらプログラムの作り方を解説しています。
添付の Robopro プログラミングガイドも合わせ参照しながら
プログラムの作り方を学んでいってください。

まずロボットを動かす各種要素について解説します。

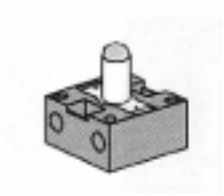
アクチュエーター

●パワーモーター



ＤＣ（直流）モーターはロボットにおいて大変重要な
アクチュエーターです。
このキットには２個のＤＣモーター（９Ｖ　2.4Ｗ）が入っていて
減速比は 50：１になっています。
モーターの回転速度は、毎分数千回ですが減速ギアを
利用することにより、回転速度は落としトルクを
大きくしているのです。

●レンズ付きランプ



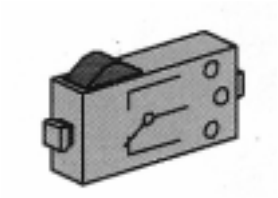
白熱電球（９VDC、１５０mA）は光信号の出力と使用します。
レンズが付いていることで光源を集約して光線を作るのに役立ちます。
白熱電球の使用方法の一つとしてスイッチの状態などの表示が
挙げられます。
警告メッセージ等も明滅するランプによってできるでしょう。
さらに、別の使用方法も挙げられます。センサーの補助パーツとして使用出来ます。
このキットではランプは２つのフォトトランジスターと組み合わせて
ライン検知用に使いますが、その時、ランプは光源として働き、フォトトランジスターは、
異なる強さで反射した光を検知することによりラインなどをよりはっきりと検知します。

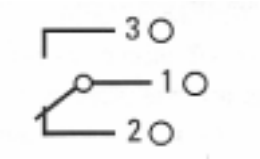
センサー

●プッシュボタン

プッシュボタンはデジタルセンサーの一種です。
デジタルは 2 つの異なる状態を表示しますがそれは「0」と「1」の信号を、
スイッチを用いて表現します。すなわち「０」なら電流の流れていない状態を、
「１」は電流の流れている状態を表します。

このフィッシャーテクニックのプッシュボタンには3本の接続端子がありますが、
赤いボタンを押すと1と3がつながり、1と2の接続が切断されます。





スイッチの機械的なオン・オフの切り替えに至る時間差を「ヒステリシス」と言いますが
電子スイッチの「スイッチング・ヒステリシス」はとても重要な特性です。
もし、それらが存在しなかった場合、例えば、スイッチのオンのポイントとオフのポイントとが信号認識において同じになってしまいます。
又、ヒステリシスのポイントが非常に小さなインターフェイスでは、故意ではない
接触が生じてしまうこともあるでしょう。つまり、正確にカウントが出来ないと
いうことになります。だから、フィッシャーのスイッチはスイッチを切り替えるものとして設計されています。従って、オン・オフの両方の状態を正しく検知出来るのです。

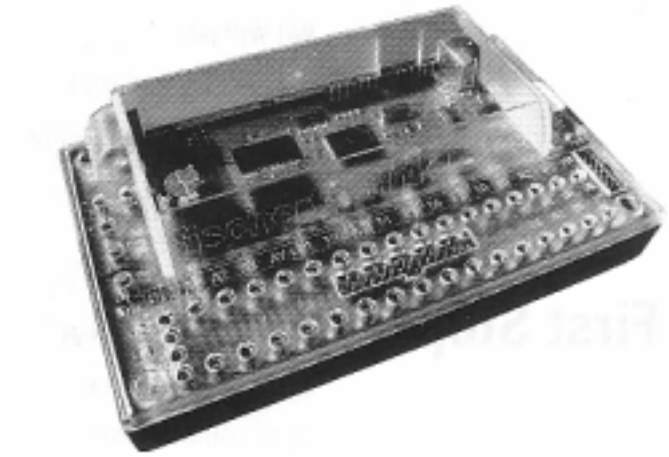
●フォトトランジスター

フォトトランジスターは半導体(電気的な特性は光に依存します)です。
太陽光で発電する太陽電池は皆さんよくご存知だと思います。
フォトトランジスターは、ミニ太陽電池とトランジスターを
組み合わせたものと理解して下さい。
ベース接続は、外部にありません。
従って、それは図面上では点線として現われます。
その場所で、光パルス(光子)が非常に小さな光電流を発生させ、それはコレクターで
利用可能なまで増幅されます。これらが機能するためには、光トランジスターに
付加的な外部配線が必要ですがこれは、インターフェイスに含まれているので、
とくに皆さんにとっては重要ではありません。
フォトトランジスターは、デジタル・センサーおよびアナログ・センサーの両方として
使用することができます。デジタル・センサーとして使用する場合では、はっきりした
黒いラインの検知として使用できます。又、光の強さを区別することもできます。
その場合は、フォトトランジスターはアナログ・センサーとして作動します。

注意：フォトトランジスターは接続極性を間違わないようにしてください。
赤い印がプラスです。最大電圧は３０Ｖです。

ロボインターフェイス

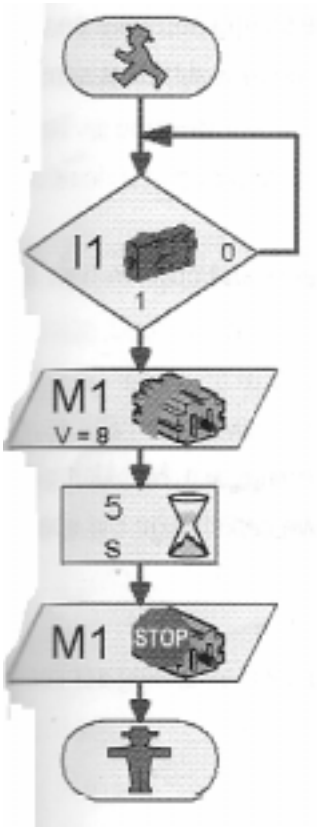
インターフェイスを用いることで必要なプログラムを読み込むことができます。
また多数のセンサーやモーターなど入出力装置もこのインターフェイスで制御されます。
インターフェイスには 8 個のデジタル入力のほかに数個のアナログセンサーの



入力点があります。
たとえば端子 AX および AY に入力された 0 から 5．5ｋΩの抵抗は 0 から 1024 の値に
変換されます。またフォトトランジスターを使って明るさの強さも数値に
置き換えられて表示されます。
0 から 10V の電圧も A1 と A2 に接続された 0 から 10ｖの電圧も数値に換算して
表示可能です。
インターフェイスの最も重要な機能は入力値の論理的結合にあります。
そのためにはプログラムが必要となります。プログラムで、インターフェイスに入力された情報とそれに対応した出力の制御（例えばセンサーが働けばモーターを回転させるなど）を行っているのです。

ソフトウエア Robopro

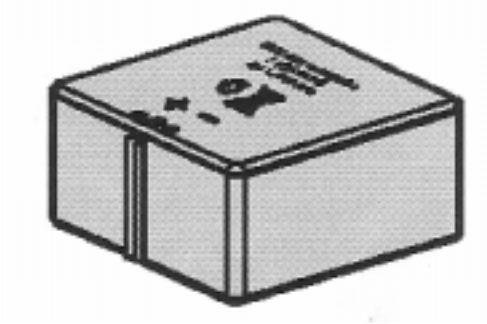
そしてそのプログラムを作成するために、グラフィカルで
簡単に作成できるソフトとして新しく Robopro というソフトが
用意されています。
ワークシート上で皆さんがハイレベルなプログラムを簡単に
作る事が可能になります。
しかも、インターフェイスに組みこまれているコンピューターは、マシンとコンピューターを介し接続することで、
実際にマシンをプログラムで組んだ
命令通り実行させる事が出来るのです。但し、プログラミングはグラフィカルなブロックで行いますが、実際インターフェイスへ送られている信号は違う言語へ変換されています。従って、



Robopro はグラフィッカルにプログラミングが出来る上、
インターフェイスが使用することができる言語に翻訳までしてくれるのです。

電源

このキットでは、独立した電源としてニッカド電池充電セット（PA-02 ニッカド電池充電セット　税込価格７３５０円）が必要になります。



モーターやセンサーなどのエネルギーは全て
この電源から得るわけですが、
モーターが電力を消費しても、センサーが正確な
結果を出す為には安定した電圧が
必要となります。充電池は何度も充電することが
できるので、普通の電池より経済的と
言えるでしょう。
十分に充電した上で電池は使用してください。また電源を接続する時は極性に必ず注意を払って下さい。

ロボットの組立て方法

それではロボットの組立にかかりますがまず最初はインターフェイスやセンサーの基本的な機能をチェックすることから始めます。
つぎに最初に単純なロボットを組立て、これをベースにして順次複雑な機能を持った
ロボットに仕上げていきます。
もしプログラムを作る過程でプログラム作成に時間がかなり過ぎる場合は
まずサンプルプログラムをダウンロードし、このプログラムを使ってロボットを
動かすことができます。
またこのマニュアルの最後にトラブルシューティングが記載されています。

またロボットを組み立てる際や動かす際は十分に注意をしてください。
導線の接続については二重、三重にチェックしてください。
またロボットの組立に際しては特に動きがスムーズかどうかをよく確認してください。

第２章　ロボット組立ての第一歩

ロボットの組立を始める前に、パーツの特性等を含む理論的な知識を身に付けましょう。
すぐにロボットの製作にかかりたいとか複雑な歩行ロボットを組み立てたいと思う人もいるでしょうが（組み立て説明書を注意深く見ながらであればもちろん可能です。)、
それが正しく作動しない場合はどうしたら良いでしょうか？
このような場合は、エラーの原因を系統的に調べなければなりません。
そのためにはセンサーやアクチュエーターに関するある程度の基礎知識が必要です。

まず、パソコンとインターフェイスの相互作用をチェックしてみましょう。
Robopro マニュアルの 1 章・2 章にソフトウエアのインストール方法や接続方法が
記載されています。
「テストインターフェイス」を使っていろいろなセンサーやモーターのチェックが
できます。

● プッシュボタン

たとえばインターフェイスの I1 にプッシュボタンを接続し、プッシュボタンを押したときに入力状態がどう変化するか、チェックしてみて下さい。

● パワーモーター

たとえばモーター出力の M1 にモーターを接続して、マウスの左ボタンをクリックして
モーターの回転方向や回転速度をチェックすることができます。

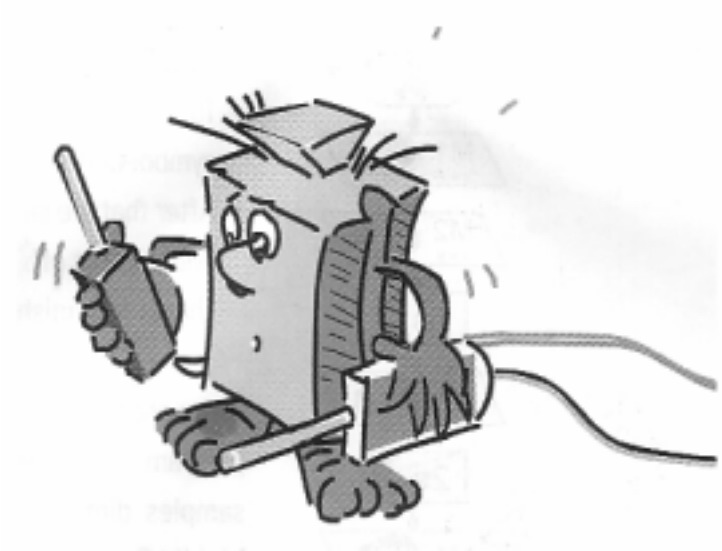
●フォトトランジスター

アナログ入力はたとえばフォトトランジスターを接続端子 AX につなぐことで
チェックできます。ただしモーターやプッシュボタンの場合は、極性はどちらでもいいですがフォトトランジスターの場合は極性に注意してください。
（赤い印のある側にプラスを接続します）

さて懐中電灯を使ってフォトトランジスターにあたる光の強度を変えて見ましょう。
すると AX の緑色の横棒の長さが変化します。
懐中電灯を消しても AX の横棒が消えないときは室内の照明が明るすぎるからです。

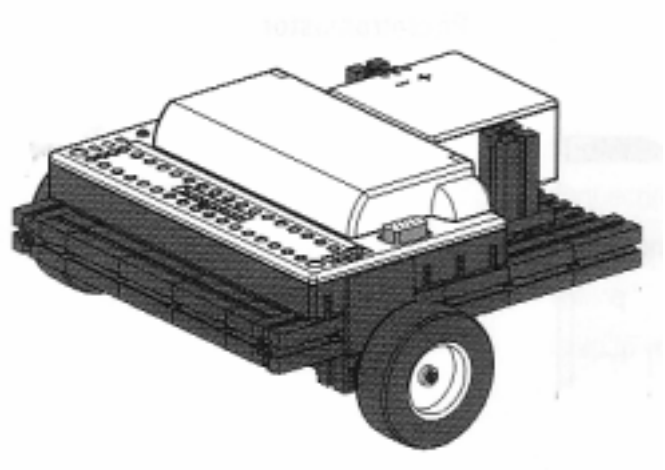
またフォトトランジスターにカバーをかけると横棒の位置も変わります。

つぎにコネクターの色について注意しましょう。
組立の際は赤のコネクターには赤色の導線を、緑色のコネクターには緑色の導線を接続します。また回路で極性が大事なときは必ず赤い導線はプラスに、緑の導線はマイナスに接続することが大切です。



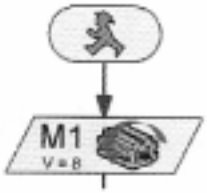
簡単なプログラムの練習として Robopro マニュアルの第 3 章にあるガレージドアを使ってみてください。
ロボットを作らなくても使えますのでプログラムの練習としては最適です。
インターフェイスに 3 個のプッシュボタンと 1 個のモーターを取り付けるだけで実験できます。

第３章　単純なロボットを組み立てる



インターフェイスのテストおよび
ガレージドアシステムが終わると
いよいよロボットの製作にかかります。
まず最初に 2 個のモーターを使った
単純な自走ロボットを製作します。
これは簡単ですのですぐに組み立てられると思います。

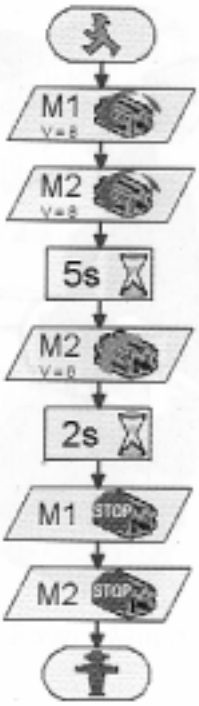
Robopro を開いてプログラムを作成しますがまずレベルを設定します。
このプログラム作成ではレベル１で十分です。
空白のページが開かれ、左側に要素ウインドウが開いています。



ここからプログラム要素を選択してマウスの左クリックで空白の作業スペースに
貼り付ければいいのです。
右クリックでプログラム要素のプロパティを変更することができます。

[課題１(レベル１)]

5 秒間前進し、次に 2 秒間回転してから停止する
という内容でプログラムを作成してください。



[ヒント]

まず始めてですので順番に進めていきましょう。
●まず緑色の人の形をした「スタートブロック」を
選択します。
●つぎに「モーター出力ブロック」を選択し、
スタートブロックの下に貼り付けると自動的に
接続線がつながります。
プロパティウインドウを開いて、M1 と CCW を
選択して OK で確定させます。
●同じようにして次の「モーター出力ブロック」を選択し、
今度は M2 とします。

●ある一定の時間待機するには「遅延ブロック」を使用します。このプロパティで
５秒を設定します。
●次にモーター２の回転方向を CW に変更し、２秒待って両方のモーターの回転を
停止させます。
●そして最後に「終了ブロック」を配置してプログラムは完成します。
　前ページの左図がプログラムの内容です。

作成したプログラムが正しいかどうか自信が無ければサンプルプログラムと
つき合わせてください。サンプルプログラムは[Robopro]の[Sample Programs]
ディレクトリーにある[Simple Robot1.rpp]として保管されていますので
これと比較してみてください。
間違いなければこのプログラムをインターフェイスにダウンロードします。
ダウンロードボタンをクリックするとポップアップウインドウが開きますので
ダウンロード先を Flash memory 1 にし、Start program after download(ダウンロード後
すぐ実行)を選択すると、ダウンロードが完了するとロボットはすぐに動き始めます。
もしダウンロード後にすぐ実行させず、インターフェイス上のプログラムボタンを
押してから実行するようにするには Start program using button on Interface
(インターフェイス上のボタンを押してから実行)を選択します。

プログラムをもう一度実行させるにはプログラムボタンをしばらく押すと
また実行されます。
またプログラムはフラッシュメモリーに保存されているので電源を切っても
プログラムは消去されません。
電源を再び入れたときは Prog1 の LED が点灯するまでプログラムボタンを
押し続けます。
そしてプログラムを実行するにはボタンを押せばいいのです。
ここまでうまくいきましたか。

[課題２(レベル１)]
つぎにロボットに７秒後に止まらずに次の動作をするように
プログラムを作って見ましょう。
●ロボットを前進、右折、左折、後退を、時間を変え、速度を変えてやらせる。
●プログラムはインターフェイスの Prog ボタンを押すまで停止しないようにします。

[ヒント]
●モーターの回転方向の設定はすでに学びました。

●モーターの回転速度は 1 から 8 まで 8 段階に設定できます。
もしモーターM1 と M2 が同じ方向に回転していても回転速度が異なると
ロボットは回転します。
●プログラムを停止させないためには停止ブロックを使わず、
またスタート位置に戻るよう接続線をつなぎます。
●このプログラムは Simple Robot 2 として保管されていますので
あなたが作ったプログラムと比較して間違いないか確認してください。

うまくいきましたか？
このロボットはさほど賢いとはいえません。
なぜなら障害物や机のくぼみなどを感知することができないので
よく見守っておかないと障害物にぶつかったり、机から落ちたりします。
この解決方法を次の章で学習しましょう。

第４章　自走ロボット

ロボットが周囲の状況を判断するためにセンサーが必要です。
次に組み立てるロボットでは数種類の異なったセンサーを使用することができます。
センサーの入力情報はたとえばロボットの内部から、たとえばパルス歯車を使って
回転数から走行距離を求めたり、光探査やライントレースのように外部信号を
使うこともできます。
プログラムは robopro の sample programs/Robo Mobile Set の中に
入っておりますが自分なりにプログラムを作成することをお勧めします。

４－１基本モデル

このモデルでは 2 個のセンサーを使って走行距離を計測することができます。
それぞれはプッシュボタンとパルス歯車から成り立っております。

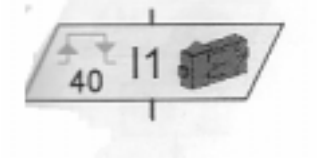
組立マニュアルに従ってこのロボットを組み立ててください。
組立が完了するとインターフェイスを接続せずに直接モーターに電源を接続して
すべてがうまく動くかテストしてみてください。

[課題１]

ロボットが 40 パルス分直進するプログラムを作りなさい。
パルスの入力にはインターフェイスの I1(インプット１)を使用します。
またパルス単位あたりの走行距離も求めなさい。
このテストを 3 回実施して表を仕上げなさい。

[ヒント]

●両方のモーターの回転方向を CCW に設定
●入力 I1 でのパルスカウントにはパルスカウンター設定ブロックを使います。



パルスカウントでは両方のエッジを使用します。
（０－＞１と１－＞０の両方）
カウント方法はプロパティウインドウで設定可能です。

●それからモーターを停止してプログラムを終了させます。
●このプログラムは Basic Model1.rpp として保管されています。

	パルス数	走行距離	距離／パルス
テスト１	40		
テスト２	40		
テスト３	40		

ロボットは恐らく１パルスあたり約１ｃｍ走行することでしょう。
ここでロボットの走行方向とモーターの回転方向について
次の表でまとめておきましょう。

ロボットの走行方向	モーターM1 の回転方向	モーターM2 の回転方向
直進	CCW	CCW
後退		
左折		
右折		
停止		

すべての場合について 2 個のモーター出力を設定しなければなりません。
この煩雑さはサブプログラムを作成することで省力できます。
サブプログラムの作成方法については Robopro マニュアルの第４章に
説明しておりますのでまずマニュアルを読んでから次に進んでください。
プログラムレベルはレベル２になります。

[課題 2(レベル２)]

●それぞれの走行方向についてサブプログラムを作りなさい。
●できればサブプログラムを使って１m四方の正方形上をロボットが走行するようにしなさい。
●繰り返し走行させた場合の正確性はどうでしょうか。

[ヒント]

●最初に「直進」のサブプログラムを作って見ましょう。
その他のサブプログラムは「直進」サブプログラムをコピーしたあと、
モーターの回転方向等を修正するだけで簡単に作成できます。
●左折・右折のときはモーターの回転速度を落とすことで正確性が高まります。
●もう一度パルス数をカウントするのにパルスカウンターを使用します。
90 度回転するのに何パルス必要であるかを調べるときは
プログラムを RAM にダウンロードして行います。

この方がフラッシュメモリーよりもずっと早いからです。
それからフラッシュメモリには耐用回数があり概ね 10 万回のダウンロードまでです。
●このプログラムは Basic Model2.rpp に保管されています。

４－２光追尾ロボット

つぎにロボットが周囲からの信号に対応することができるようにしたいと思います。
丁度第１章に描いた蛾と同じ様に光を見つけてそれに向かって動いていくような
ロボットを作ってみます。
このロボットには２個の
フォトトランジスターを使用します。
それぞれのフォトトランジスターが
モーターの動きに対応します。
このプログラムは２つの部分から
成り立っています。
ひとつは「光源を探すプログラム」で
もうひとつは「光に向かって動いていくプ
ログラム」です。
このためにまた「サブプログラム」を
使用します。

プログラムを実行するとまず「光探査」のサブプログラムが実行され
これは光が見つかるまで繰り返されます。
光が見つかるとメインプログラムに戻って光に向かって動いていきます。
ロボットの方向が正しいラインと異なる場合は光センサーのうちの 1 つには光が当たりません。すると対応するモーターはストップしその結果、2 つのセンサーで再び光源を検知することができるのです。

[課題１(レベル２)]

●最初のプログラムは「Light seeking」です。
ロボットはゆっくりとどちらかの方向に少なくとも 360 度回転します。
そして光を見つければその場で停止しますが見つからなければ逆の方向に
360 度回転します。それでも見つからない場合は 5 秒間待ってから同じ探査を
繰り返します。
●運良く光が見つかれば、ロボットは光源に近づいていきます。

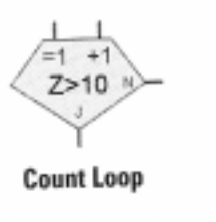
仮に光源が左右に移動してもロボットは光を追いかけて行きます。
しかし光源を見失うと、ロボットは再び光源を捜し始めます。
このようなプログラムを作って見ましょう。
またこのプログラムの原理が分かれば友達と一緒に懐中電灯でロボットを誘導し
障害物コースを通りぬける事が出来るかトライしてみてください。

[ヒント]

●違った方向への走行に基本モデルで使ったサブプログラムを使用します。

●Basic model2.rpp をロードすると Robopro の「要素グループの窓 element Group window」にサブプログラムが表示されますのでこれを新しいプログラムに挿入すればいいのです

●サブプログラム「光探査 Light-seeking」で「カウンターループ」ブロックを使用しています。



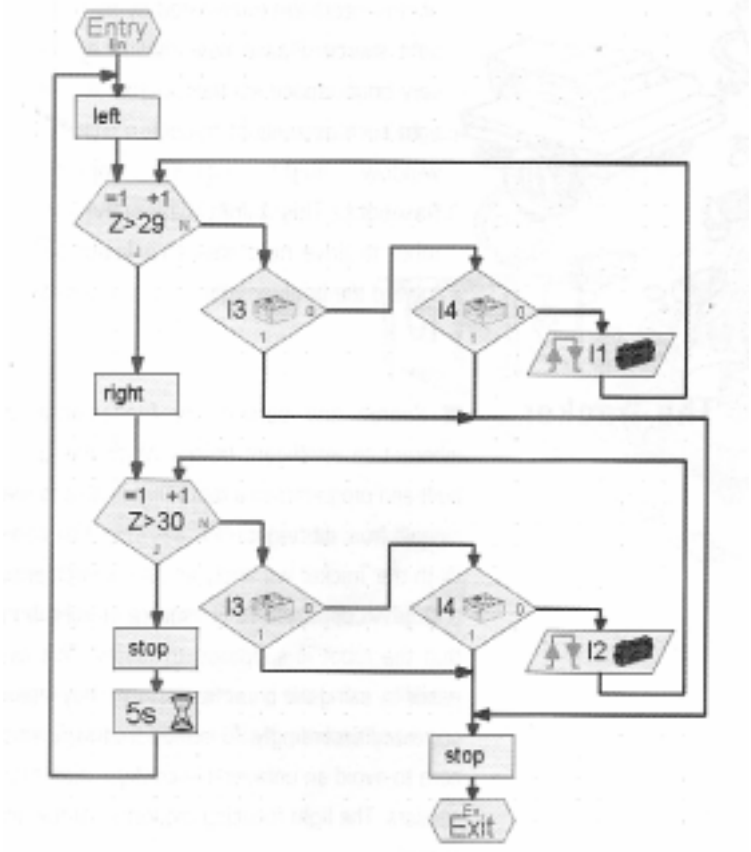
このブロックでは N（No）なら＋１加算されて
その値が Z>10 かどうかを判断し N か J(Yes)かに
よってまた分岐します。

●このプログラムではロボットが光を発見するか、360 度回転するまで繰り返します。
Z の値がいくらでロボットが 360 度回転するかを調べ、その値を使います。

●2 番目のループでは反対方向に回転した場合で同じように値を調べて入れます。

● ロボットが光を発見すると
ロボットは停止し、サブプログラムから抜け出します。

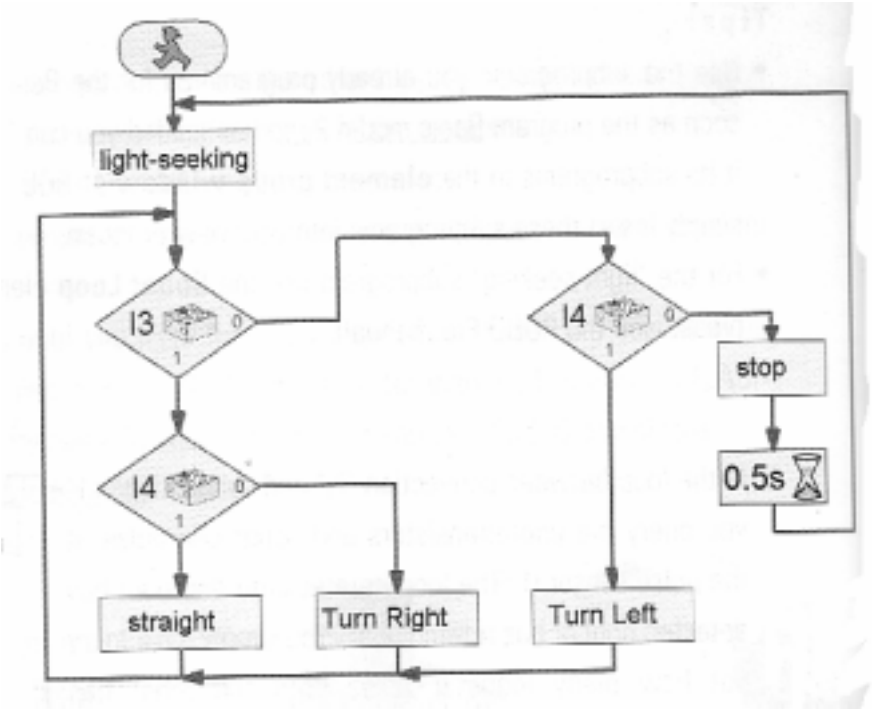
●サブプログラムは右のようになります。



●メインプログラムでは再びフォトトランジスターを調べその結果に基づいて
モーターの動きを制御します。

I3 と I4 で光感知	前進
I3 で光感知	右折
I4 で光感知	左折
光感知せず	停止して再び光を探す

●モーターは同じ方向に回転しながら速度を変えて左右に回転させます。
●メインプログラムは次のようになります。



●このプログラムは Lightseeker.rpp として保管されています。
●懐中電灯を光源として使用してください。その際、あまり焦点を絞り込まないことです。
そうしないと両方のフォトトランジスターに光が当たらない場合が出てくるからです。
また室内に懐中電灯以上に強い光源があるとその光に邪魔されてロボットは
プログラムどおりに動作しませんから注意してください。

４－３ライントレースロボット

探索と追跡は知的生物が持つ本質的な特性と言えます。
そこで今回は追跡機能（ライントレース）を備えたロボット作りにチャレンジしましょう。
前回は光源を目標として移動させましたが、今回はロボットが進むべきラインを
引きそれを追跡させます。この課題では光センサーを用いてラインからの
光の反射と床面からの光の反射の違いを利用してモーターを制御して
ラインにしたがってロボットを移動させるということです。
そのためにまずラインから反射する光に光センサーが正しく反応しているかどうか
事前に調べておく必要があります。
ランプと光センサーの配置が正しくない為、センサーへ光が正確に当たっていないことがありますのでその確認もしてください。このキットで使用する電球は光学用レンズで光を集光していますのでセンサーへの反応は大変効果的といえます。
組立マニュアルに従ってライントレースロボットを組み立てて下さい。

[課題１]

●最初にラインを探すプログラムをサブプログラムで組みます。
そのためにはロボットは最初に１回転します。
●もしそこでラインが見つからない場合は少し前進しそこで再びラインを探し始めます。
●そしてロボットがラインを見つけるとそのラインを追って進みます。
●ラインの端までたどり着いたり、あるいはラインを見失った場合（例えばラインが極端に曲がっているため）は、新たにラインを探し始めるようにします。

[ヒント]

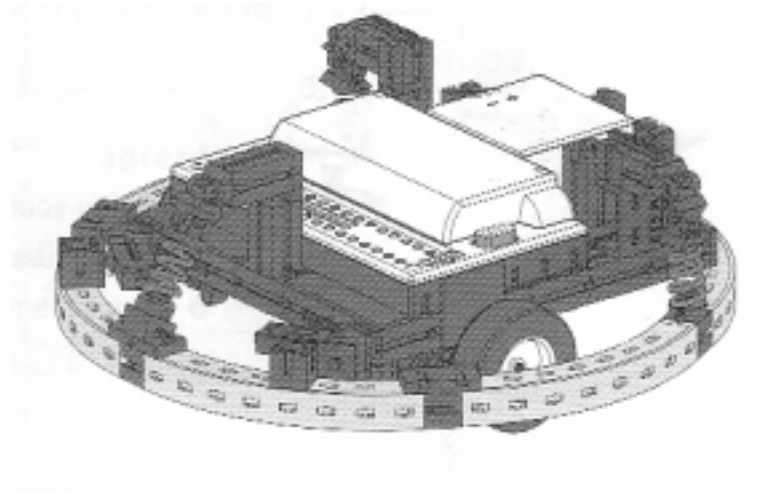
●ランプのスイッチが入った後、光トランジスターが反応するまでに、

短いディレイタイム(約 1 秒)があります。
そうでなければ、光トランジスターは「ぼやけたところ」(何も無いところ)でも
検知してしまうからです。
●ラインは巾約 2 cm 位の黒い絶縁テープを貼って作ります。
（黒のサインペンの場合は誤動作の出る場合がありますので絶縁テープをお勧めします）
●ランプを点灯したときフォトトランジスターが黒い線の上にあるときは
両センサーの値は「０」(黒い)で、白い床面上にあるときは反射光に反応して
「１」になります。
●プログラムは Tracker.rpp として保管されています。

[課題２]

●曲がり角度の異なるトレースラインを作ります。
ロボットがトレースできる最小の半径はいくらぐらいですか？
●ラインを修正したとき今度は M1、M2 の回転速度を変えて
どのような組み合わせがロボットの走行に最も適しているか調べてみます。
●次に円周を作ってロボットが最も早く走行できるスピードを調べてみてください。
これは数台のロボットで競争に使えます。

４－４障害物感知ロボット



これまで作ってきたロボットではある一定の距離を走行したり、
光を探したり、ラインをトレースすることができましたが
障害物に出くわすとどうなったでしょうか。
障害物を押しのけるか、押しのけることが出来なくて電源が消耗して
停止するかになります。しかしあらかじめ障害物を感知して、それを避けて走行することが出来ればかなり知能的といえます。
そのために押しボタンセンサーを取付けたバンパーを使用します。
このバンパーを使って右側か左側かあるいは後部に障害物にぶつかったかを感知し、
プログラムでそれを回避する動きをさせます。

まず最初に、障害物感知ロボットを組み立てて下さい。
センサーI1 は走行距離を測定するのに必要です。
そこで I2 のプッシュボタンを取り除いてこれを障害物感知用に使います。

[課題１(レベル２)]
ロボットはまず前進します。そして左側で障害物（I4）を感知したら
少し戻って右側に移動します。
右側（I3）で障害物を感知したら少し戻って左側に移動します。
このプログラムを作ってください。

[ヒント]
●後退するときに障害物を感知する場合に付いては後で考えます。
●メインプログラムでプッシュボタンの状態をチェックします。
すなわちどのプッシュボタンが作動したら右に曲がるか、左に曲がるかを

プログラムします。もちろんその動きはサブプログラムで作成します。
● 左折の場合と右折の場合ではパスルカウント数は異なります。
(右側では３パルス、左では５パルスになります）さもないとコーナーに入ってしまって
身動きできなくなるからです。
●このプログラムは Obstacle1.rpp として保管されています。

まだロボットが障害物に出くわしたときの対応として２つのケースが残っています。
すなわち真正面に障害物がある場合と真後にある場合です。
真正面にある場合は両方のセンサーが感知する場合です。すなわち I3 と I4 が同時に
反応した場合は真正面に障害物がある場合で、このときは直ちに９０度回転します。
また後退しているときに I5 が反応すれば後に障害物がある場合です。

障害物	センサー(プッシュボタン)	反応
右側	I3	左折(約３０度)
左側	I4	右折(約４５度)
正面	I3 と I4	９０度左折
背後	I5	停止してから動き出す

このプログラムでは新しいプログラム要素として「演算子」が出てきますが
これはプログラムを作るうえで大変便利なものです。
このレベル３ではオレンジの矢印を使うことで異なる要素間でのデーター変換を行うことができます。このレベルになることでこれらの操作が大変楽になります。
詳しくは Robopro マニュアルを参照してください。

[課題２(レベル３)]

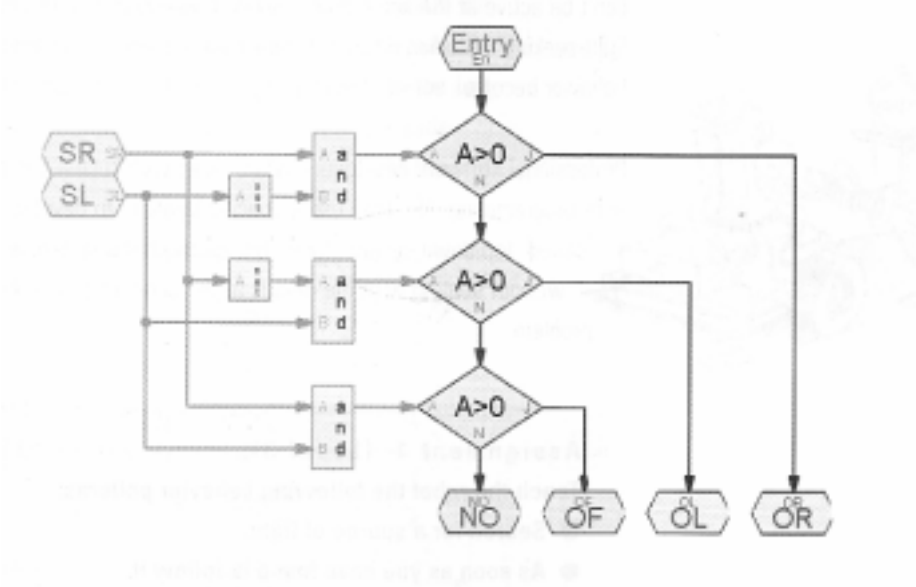
プログラムを上のケースに対応するよう修正して下さい。
ロボプロレベル３を有効に使ってください。

[ヒント]

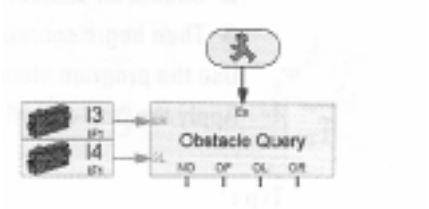
●サブプログラム「障害物探査 Obstacle Query」の中で「演算子」を使って
すべての組み合わせに対応します。
そしてそれぞれの場合について出口を作ります。

データー入力 SR=右側のセンサー(sensor right)
データー入力 SL=左側のセンサー(sensor left)
出口 NO=障害物なし(no obstacle)

出口 OF=障害物正面(obstacle in front)
出口 OL=障害物左側(obstacle left)
出口 OR=障害物右側(obstacle right)

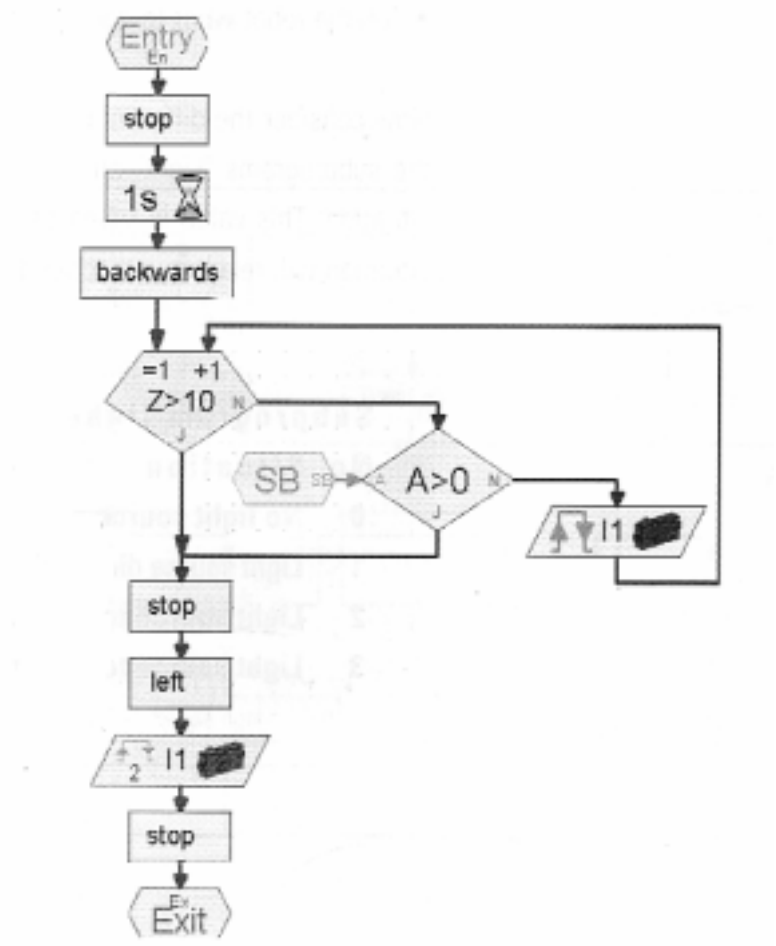


●次にメインプログラムでオレンジの矢印を使って「データー入力」とつなぎます。
メインプログラムは次のようになります。

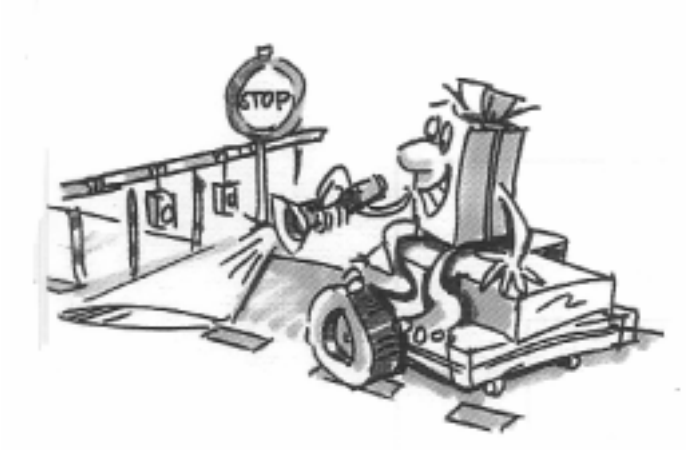


●それぞれのサブプログラムにおいて I5 はロボットが後退するときに
チェックされます。ロボットはセットされたカウンター数だけ後退するか
または I5 に障害物が当たるまで後退します。
I5 はメインプログラムでも配置され常に監視することができるようになっています。

●このプログラムは Obstacle3.rpp として保管されています。
このプログラムの利点はメインプログラム上でサブプログラムのセンサーの動きを
直接監視できる事です。
入力を変更しようと思えばすべてのサブプログラムを調べることも無く
簡単に変更できます。その上、明快に論理操作を行うことができます。
このことは分岐ブロックを使っても行うことができますが複数の場合わけを行う必要が
ありますので大変わずらわしくなります。


Entry
En
stop
1s
backwards
=1 +1
Z>10
N
J
SB
SB
A
A>0
N
J
I1
stop
left
I1
stop
Ex
Exit

４－５障害物感知機能付き光追尾ロボット



ロボットのいろいろな動きも大体以上で終わりですので、ここで
ロボットのいろいろな機能の複合化に取り組んでみたいと思います。
すなわち光追尾機能と障害物回避機能を結合しようと思います。
もちろんこの機能が同時に働くことはありません。
まず光追尾を行い、その過程で障害物を感知すると障害物回避の動きをとります。

専門のプログラマーはこの場合のように簡単にプログラムが作成できないときは
特殊な手法で作成します。その手法の一つに「トップダウンデザイン方式」があります。
この方法は最初の段階では詳細にプログラムを作らず全体をまず組み上げ、
その後に詳細を決めていくやり方でこの課題にはこの手法が適しています。

[課題１（レベル３）]

ロボットが次の行動を取れるようにプログラムしてください。

- ●光源を探す
- ●光源を探せばそれを追捕する
- ●途中障害物があればそれを避ける
- ●改めて光源を探す

レベル３のプログラム要素を使用し、作成手法としては
「トップダウン」方式で行います。

[ヒント]

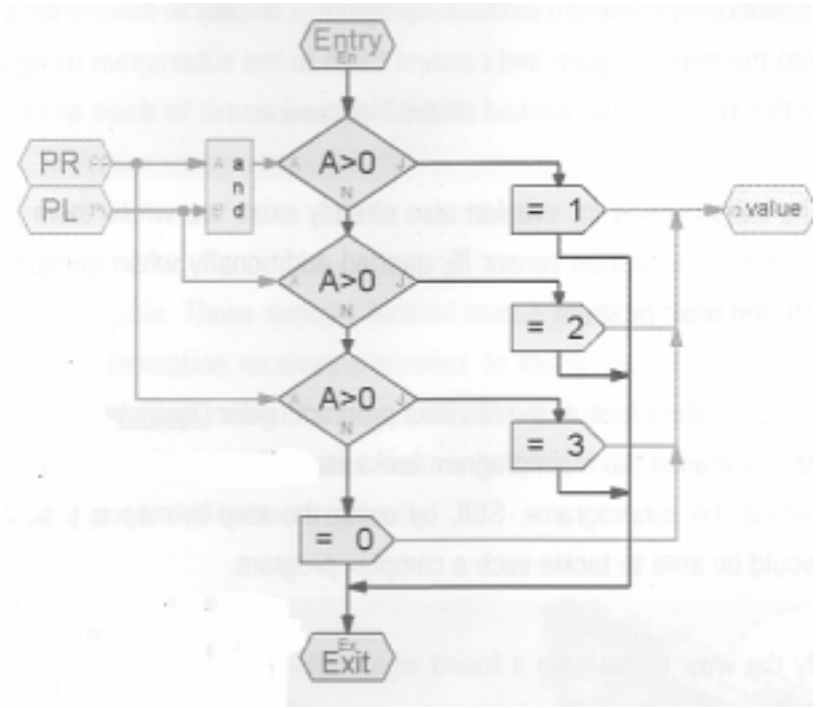
●まずこの課題を３つの部分に分けます。

光源を見つける(サブプログラム「Light」)
障害物にぶつかる(サブプログラム「Obstacle」)
これらの場合にどのように動くか(サブプログラム「Driving」)

●サブプログラム「Light 光源」の場合わけ

No	状態	センサー	反応
0	光源なし	I6=0:I7=0	光探査
1	正面に光源あり	I6=1:I7=1	前進
2	左側に光源あり	I7=1	左ターン
3	右側に光源あり	I6=1	右ターン

●サブプログラム「Light」



PR=phototoransister right

PL=phototransister left

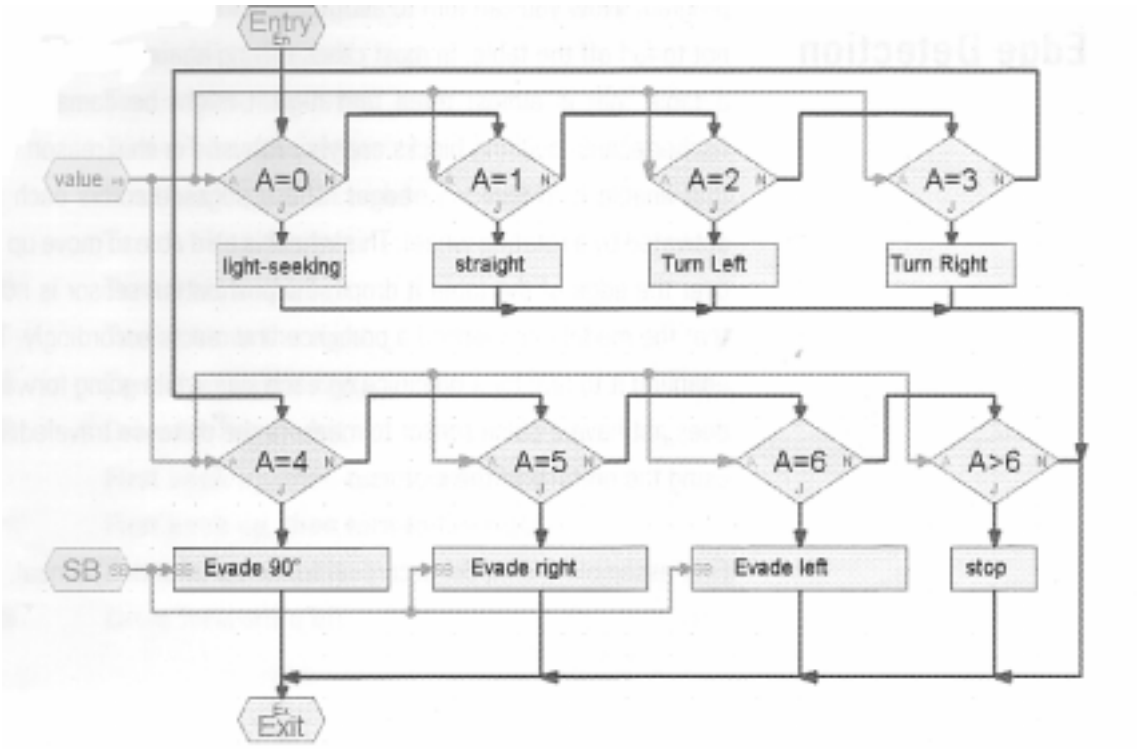
前回と同じようにメインプログラムにフォトトランジスター探査の
プログラム要素を配置するとサブプログラムでの動きを監視することができます。

コマンドからの値を保管する変数もメインプログラムに配置します。
これはいろいろなサブプログラムで使用されますので
データー出力を使ってサブプログラムに接続します。

同様のやり方でサブプログラム「障害物」も作成してください。
サブプログラム「Driving」では変数の現在値を、分岐ブロックを使ってチェックします。

●サブプログラム「Obstacle 障害物」の場合分け

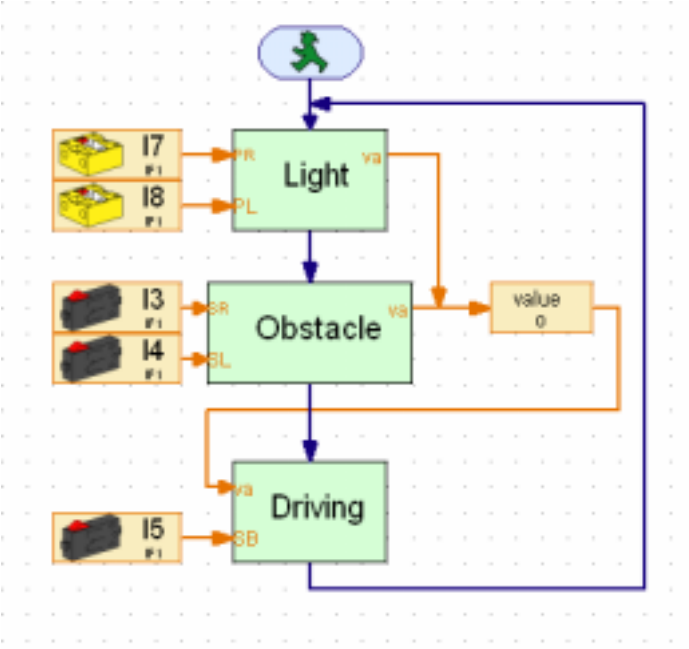
No	状態	センサー	反応
4	ロボット正面に障害物	I3=1:I4=1	９０度回避
5	右側に障害物	I3=1	左へ回避
6	左側に障害物	I4=1	右に回避



最後にサブプログラムの中に使われるサブプログラムを作る必要がありますが
ほとんどのサブプログラムはすでに作っております。
例えば光探査のサブプログラムは光探査のプログラムからコピーすることができます。
その方法が分からなければ Robopro マニュアルの第４章を参照して下さい。
但し注意する点があります。
光探査モデルではフォトトランジスターは I3 と I4 に接続されていましたが
今回は I6 と I7 に接続されています。
また回転時のパルス数を計測するために I1 と I2 が使われていましたが
今回は I1 のみが使われています。
従ってプログラムをコピーした後修正が必要ということです。

障害物回避のサブプログラムもすでに存在しています。
これは障害物回避のプログラムの中で作成しました。

完成プログラムは Obstacle-Light.rpp として保管されていますので
チェックしてみてください。メインプログラムは下のようになっています。



４－６くぼみ感知ロボット

ここまでは基本モデルを用いての簡単な実験でしたが次にロボットの「脱出移動」について取り組んでみたいと思います。
テーブルの上を動き回りながらその端から落ちないようにするにはどうすればよいかが課題です。落ちないためには「くぼみを探知するセンサー」が必要です。
２つの補助車輪を組み合わせ単純かつ有用なセンサーを組み立てます。
車輪には、スイッチを取り付け、ロボットの移動方向の前に取り付けます。そして、それぞれの車輪が上下に移動することが出来るようにします。車輪がくぼみに差し掛かると下方向へ車輪が落ちセンサー（スイッチ）が働き仕組みにします。

[課題１(レベル３)]

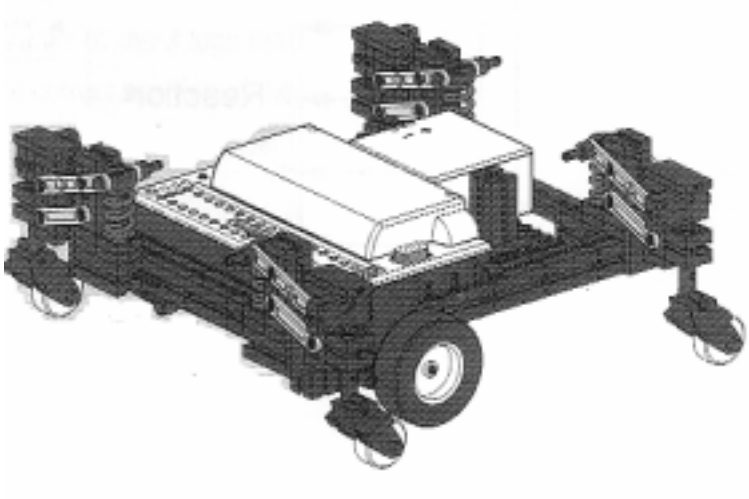
まずロボットが端に来たらどのように反応すべきかを考える。
その際センサーのいろいろな組み合わせがあることに気づくでしょう。
４個のセンサーのうち１個が反応する場合、２個又は３個が同時に反応する場合、あるいはすべてのセンサーが同時に反応する場合などが考えられます。

[ヒント]

次のようなケースが考えられます。

No	右前 I3	左前 I4	右後 I5	左後 I6	反応の内容
0					前進
1	●	●	●	●	停止
2	●	●	●		少し右へ
3	●	●		●	少し左へ
4	●		●	●	少し左へ

5		●	●	●	少し右へ
6	●	●			少し戻ってから右回転
7	●		●		少し左へ
8	●			●	少し左へ
9		●	●		少し右へ回転
１０		●		●	少し右へ回転
１１			●	●	少し前進
１２	●				少し戻ってから左右に
１３		●			少し戻ってから右に
１４			●		少し前進
１５				●	少し前進

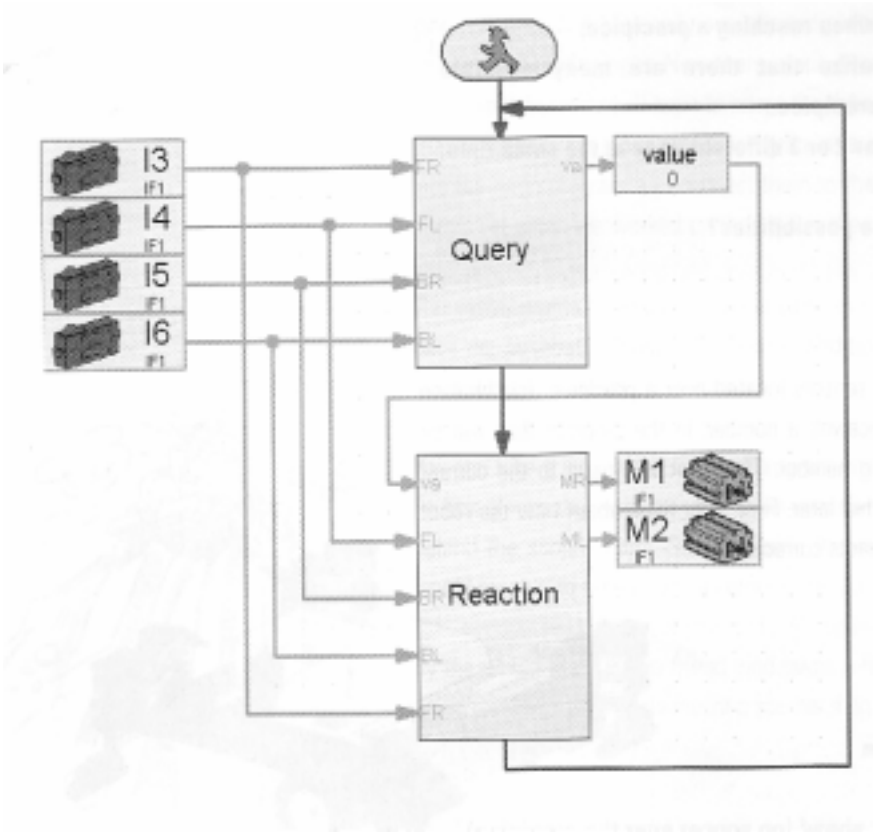


このプログラムは Edges.rpp として保管されています。

最も重要なプログラム要素はメインプログラムに配置されていて
プログラムの全体の順序流れが分かるようになっています。
センサーの動きやモーターの回転はすべてサブプログラムに記載されています。

まずはメインプログラムを見て見ましょう。
プッシュボタンセンサーのチェックから始まります。
左を見てみればどのセンサーがチェックされているかが分かります。
これらはデーター入力を通してサブプログラムと接続しています。サブプログラム
「Query」ではどのプッシュボタンが押され、その場合先ほどの場合わけに応じた
値が割り当てられます。そしてその値はメインプログラムの変数 value で
確認することができます。

そして変数の値はサブプログラム Reaction に送られ、その値によって
モーターの動きがコントロールされます。
プッシュボタンはセンサーとしてサブプログラム Reaction でも
チェックされます。
もし必要ならサブプログラムを見ることなくモーターの接続や
プッシュボタンセンサーの接続をメインプログラム上で変更することが可能です。



サブプログラムをいろいろと違ってモデルで使用しようとする場合に
いまだどの入力や出力を使うか決まっていない場合には
プログラムの作成テクニックが重要になってきます。

[課題２(レベル 3)]
プログラムをロードしてロボットをテーブル上で動かして見ましょう。
正しく反応しましたか
センサーの組み合わせにしたがって違った動きをしましたか
必要に応じてプログラムを変更してください。

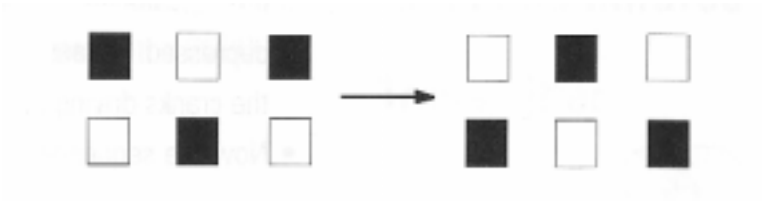
第５章　歩行ロボット

自走ロボットに続いて今度は歩行ロボットを作ってみましょう。

昆虫の歩き方がそのまま六足歩行ロボットの動きに使えます。
いわゆる「三足歩行」といわれるもので 6 本の足のうち 3 本が常に持ち上げなら
歩行するやり方です。

■　三足歩行

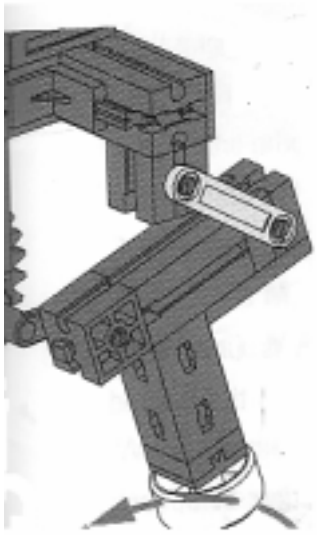
その歩き方は６本ある足のうち３本が必ず同時に地面からはなれています。
そして片側の前と後、反対側の真中の足が同時に動いていきます。
地面についている３本の足で安定した３本立ち構造で立っているため
走行時も安定していて転倒することはありません。
床についている足は黒色で示しています



■　クランクロッカー

フィッシャーテクニックの歩行ロボットではいわゆる 4 点接続ギアを使用します。
このメカニズムは「クランクロッカー」と呼ばれるものです。
すなわち歩行の足はクランクとロッカー（振り子）で構成されています。
クランクの回転運動によって駆動部分の足が前後に動くようになっています。
そして固定部分と足の先との距離はクランクが円運動をしたときに
その先が楕円形を描いて動くように設定されています。
これによって歩行のような動きになるのです。
したがって６本のクランクの位置関係が大変重要になっています。
マニュアルどおりに固定されているか再度チェックしてください。
すなわち地面に設置している３本の足のクランクは同じ位置にあります。

そして他の３本の足のクランクは１８０度回転した位置にあります。
またクランクにつながっているギアの締め付けにもゆるみがないか
チェックしてください。これがゆるんでいるとクランクがずれてしまいます。
また左右の足はそれぞれ単独に動くようになっていますが
片方の前足・後足と他方の真中の足が常に同じ位置にあることが必要です。
そしてその動きを同調させるのにスイッチ I1 と I２を使いますが
その動きはソフトウエアを使って制御します。
すべての結線が終われば「テストインターフェイス」を使って
てモーターおよびプッシュボタンが正しく作動しているかも
チェックしてください。
とりわけモーターの回転方向に間違いがないか
もう一度チェックしてください。
　　反時計回り＝前進　時計回り＝後退になります。



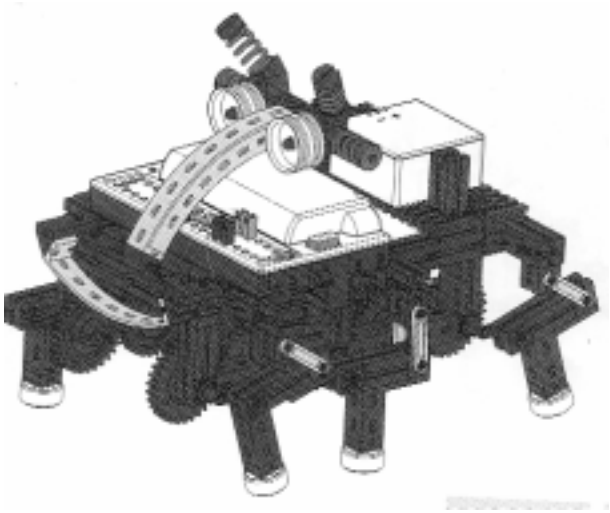
[課題１(レベル 1)]

ロボットが三足歩行して前進するようにプログラムしなさい。
プッシュボタン I1 と I2 を使って左右の足の動きを同調させなさい。
その際、片側の外側の 2 本の足ともう一方の側の真中の足の位置が常に同じである
必要があります。

[ヒント]

●まず足の位置をスタート位置にします。両方のモーターを同時に動かすと、
このようになります。
●両方のプッシュボタンが同時に押されない（このチェックはロボットが第２歩を
踏み出すときに必要です。）限り一連の動きは続いていきます。
●モーターは再び適切なプッシュボタンが押されるまで動き続けます。
ここでは両方のプッシュボタンが押されるまでロボットが動かないということが重要です。
なぜならその時にそれぞれの足が正しい位置にあることになるからです。
もちろんクランクが正しくセットされていることが前提ですが。
●ここで動きが連続してロボットは第二歩を踏み出します。
これによってロボットはプログラムが終了するまで前進し続けます。
●完成したプログラムは Walking Robot 1.rpp として保管されています。

自走ロボットの基本モデルのときと同じように歩行ロボットの左折、右折、後退の
プログラムはモーターの回転方向を変えることで作成できます。
そしてステップ数の計測には I1 か I2 を使います。

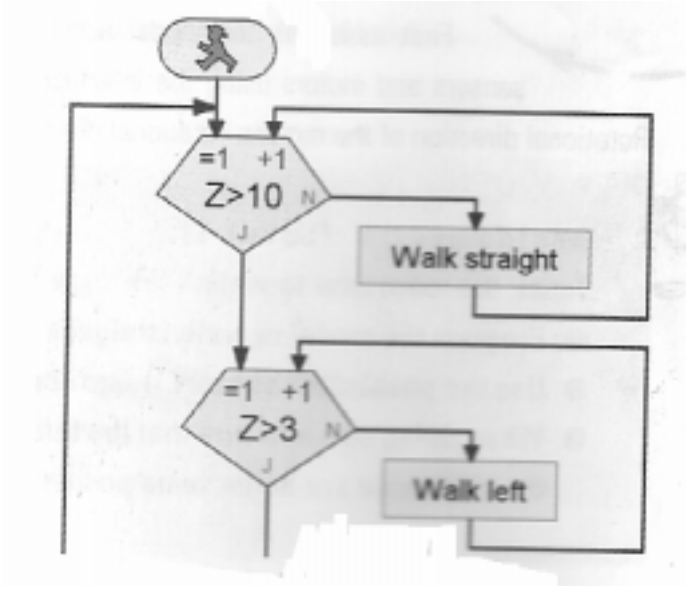


[課題２(レベル 2)]

10 ステップ前進し、次に 3 ステップ右に、3 ステップ左に、最後に 10 ステップ後退する
プログラムを作成してください。
それぞれの歩行にサブプログラムを作成します。
ステップ数の計測にはカウンターループを使用します。

[ヒント]

●まず Walking Robot 1.rpp を
サブプログラムにコピーします。
●このサブプログラムを必要な数だけコピー
し、モーターの回転方向を変更することで
左折、右折、後退のサブプログラムに
仕上げます。
●ステップ数をカウントするのに
「カウンターループ」ブロックを使用します。
サブプログラムのステップ数は 1 ステップな
ので 10 ステップならループ回数を
10 にします。



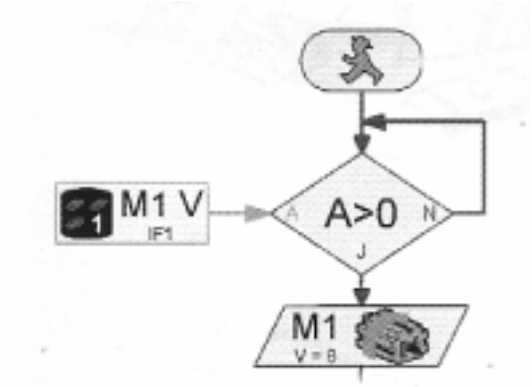
●完成プログラムは Walking Robot 2.rpp として保管されています。

障害物回避の歩行についてはすでに自走ロボットのところでプログラムを作っているので
ここでは繰り返しませんが歩行ロボットについても
障害物回避や光追捕をする歩行ロボットをつくりプログラムも作成して下さい。
組み立てるのに必要な部品はすべてキットに含まれています。
プログラムも自走ロボットのプログラムを参考にすれば作成可能です。

第６章　拡張機能

６－１赤外線リモコンの使用

ROBO インターフェイスは赤外線受信ダイオードを備えていますので
赤外線リモコン(品番 PA-07　税込価格 13650 円)を使うことができます。
ソフトウエアによって発信機のどのボタンが押されているかをデジタル入力として
利用でき、モーターのオンオフに使えます。
サンプルプログラムとして歩行ロボット用のプログラムを作成しております。
発信機の 4 つの矢印を使ってロボットを前進・右折・左折・後退させることができます。
プログラムは Walking Robot IR.rpp として保管されていますのでこのプログラムを
ダウンロードするだけですぐに試すことができます。

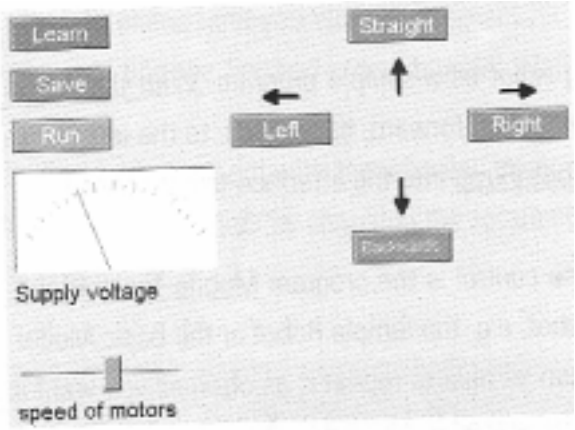


このリモコンに関連して有効なプログラムが保管
されています。
それは Mobile-Teach-IR.rpp というプログラムで、
このティーチインプログラムを
使うことで自走ロボットを制御できます。
ロボットは走行した経路を記憶して、その同じ経
路を何度でも走行させることが
可能になります。しかし一度プログラムが停止す
るとその記憶は消されます。
このようなことが可能になるのは Robopro ソフトウエアに入っているプログラム要素
「List」を使うからです。多くの変数がこの中に保管され、それが呼び戻されることで
可能になっているのです。
プログラム自身はかなり複雑ですが操作は大変簡単なものです。
１プログラム Mobile-Teach-IR.rpp をインターフェイスのフラッシュメモリに
ダウンロードし、プログラムをスタートさせます。
２リモコンの［**M1→／⇒**］ボタンを押します。これは学習モードになります。
３リモコンの矢印を使ってロボットを思う方向に動かします。

４今度は**[M2　⇒]**ボタンを押しと、今までの動きが記憶されます。
すなわち保管モードです。
５つぎに**[M3　⇒]**ボタンを押すと保管された内容にしたがってロボットが走行します。
インターフェイスのプログラムボタンを押してプログラムを停止させると記憶された
内容は消えてしまいますので必要なら移動経路はノートに記入しておいてください。

６－２Robo Data Link の使用

無線通信キット RoboDataLink を使用するとパソコンとインターフェイスをケーブルで
接続する必要がなくなります。これは大変すばらしいものです。
まずプログラムをダウンロードするときにケーブルをつないだりはずしたりする必要が
なくなります。つぎにケーブルを接続することなくオンラインモードでプログラムを
実行することができます。このことによってエラーチェックが簡単にできるように
なります。そして最後にパソコン上の画面にパネルを作成してオンラインモードで
ロボットを操作することができます。
この場合はリモコン操作と異なり変数やアナログ入力の値などインターフェイスから
供給されるデーターを逐次画面に表示してくれます。
ティーチインソフトを修正して画面上の表示板を使ってロボットを制御できる
サンプルプログラムを用意しています。
プログラムは Mopbile-Teach-RF.rpp として保管されています。
もちろんケーブルをつないで使うこともできますがこれではケーブルがもつれたり、
短すぎたりして不便です。
このプログラムで RF DATA LINK の実用性が理解できるでしょう。



プログラムをダウンロードし、オンラインモードでプログラムを実行します。
パソコン画面の表示板を使ってロボットを操作することができます。
１「Learn 学習」ボタンを押すと、学習モードがスタートします。
２望む方向に矢印をクリックしてロボットを動かします。
３「Save 保管」ボタンを押すことで、ロボットの動いた経路が保管されます。

４「Run 実行」ボタンを押すと、ロボットが保管された経路を繰り返し走行します。
もちろんプログラムが終了すると保管された経路は消されます。

６－３拡張インターフェイス

多くのセンサーを使ったりモーターを使ったりしてインターフェイスだけでは
入出力が足りなくなった場合、拡張インターフェイスを使うことで解決できます。
拡張インターフェイスには 8 個のデジタル入力、4 個のモーター出力と
1 個のアナログ(抵抗)入力端子があります。またこのインターフェイスにさらに
1 個または 2 個の拡張インターフェイスを接続することができます。
したがって最大で 16 個のモーター出力､32 個のデジタル入力､5 個のアナログ(抵抗)入力、
2 個の電圧入力、2 個の距離センサー入力端子を持つことができます。

これでも足りない場合はパソコンに複数個のインターフェイスを取り付けます。
たとえばシリアル RS232 に 1 個のインターフェイスを、USB に別のインターフェイスを
取り付けるか、2 個のインターフェイスを USB 接続します。
そしてそれぞれに拡張インターフェイスをつなぐと先ほどの倍の入出力端子を備えることができます。詳しくはマニュアルの第 6 章を参照してください。

第７章　トラブルシューティング

すべてが上手く機能していれば、実験は実に楽しいものです。
特に最初の一回で全てが成功出来れば最高です。が通常は不幸にもそう上手くは
いきません。もし、組み立てたロボットが動かず、エラーをすぐに見つけた場合は、
皆さんはロボットのメカニズムを正確に理解しているということです。
単なる機械的なエラーであれば今まで通り不正確な組立てを見つけて直す事が出来ます。
しかし、電気的な問題がさらにある場合は、エラーを発見するのは困難になります。
専門家は、トラブルシューティング用として多くの測定道具を使用します。(例えば電圧計、
オシログラフ)。しかし、誰でもそのような機器を持っているとは限りません。
だから、皆さんにはエラーを正確に見つけ出し、単純な方法でそれらのエラーを修正
しましょう。

■接続ケーブルの断線有無

組み立てマニュアルの２ページを見て接続ケーブルを作りますがその際指示された
長さに切断します。切断する前に、まず長さを注意深くチェックして下さい。
その後、各ケーブルをテストする必要があります。
充電池とランプを用意して下さい。接続しランプが点灯すればケーブルは OK です。
皮膜の取り方は両端の皮膜を４mmほどニッパー等で取り除きます。
先を折り返してプラグの中に差込みねじを締めます。

■導線へのプラグ取り付け

組立ての際、細心の注意を必ず払って頂きたいことは、
赤い導線には赤いプラグ、緑の導線には緑のプラグと接続するということです。
極性を持つ回路の場合は、通常プラス極に赤導線、マイナス極には緑の導線を
使用します。
又、インターフェイス側への差し込み方ですが赤いプラグは、入力の外側の
差し込みに挿入し、緑のプラグは、内側の差し込みに挿入します。

■接続チェック

プログラムをダウンロードしてもロボットが作動しない場合は
「チェックインターフェース」で動きを確認して下さい。この機能は大変便利です。
入力状態、出力状態等各スイッチ、モーターのひとつひとつを別々にテストすることが
出来ます。それらを全てチェックすれば、電気的な接続がすべて大丈夫であることが
分かります。モータ・コントロール・ボタンによって個々にアクチュエーターを
切り替えてください。そして、すべてが作動する場合は、エラーは機械的な原因と言える
でしょう。

■機械的な接続ミス
プラグの接続がゆるい場合はエラーが起こりやすくなりますので、もしゆるくなった場合ドライバー（またはカッターナイフの刃）で少し広げてください。広げすぎるとプラグが折れたりして使用できなくなりますので注意して下さい。
またワイヤーと接続するためのねじがゆるんでいる場合もあります。
ゆるんだ場合はもう一度しっかり締め直して下さい。

■電池のチェック
作動中に停止する場合は、充電池をチェックして下さい。
電圧が低下するとインターフェイスが働かなくなります。

■プログラムミス
エラーがみなさんの作成したプログラムによって生じる場合は、サンプルプログラムをインストールしてその動きを見て下さい。

以上を確認した上でなおエラーの原因が発見出来ない場合は発売元までご連絡ください。

輸入発売元：株式会社のもと
〒531-0077 大阪市北区大淀北１－１－１１
電話　06-6458-6831
FAX　06-6458-6811
URL　http://www.kknomoto.co.jp
E-mail　info@kknomoto.co.jp