講師用

ロボット博士養成講座

ロボティクスプロフェッサーコース

オムニホイールロボット③

（第5回/第6回テキスト）

必ず、生徒に授業日と自分の名前を記入させるようご指導をお願いいたします。

第5回授業日（だいかいじゅぎょうび） 2024年　　月　　日

第6回授業日（だいかいじゅぎょうび） 2024年　　月　　日

名前（なまえ）

Human ヒューマンアカデミー ジュニア STE"AMスクール

ロボット博士養成講座 ロボティクスプロフェッサーコース

2024年6月授業分

ロボット博士養成講座

ロボティクスプロフェッサーコース

オムニホイールロボット③

第5回

センサーを使ったカシコイロボット

講師用

目　　次

○　授業開始にあたって

授業のはじめは、着席させ、大きな声であいさつしてから始めます。

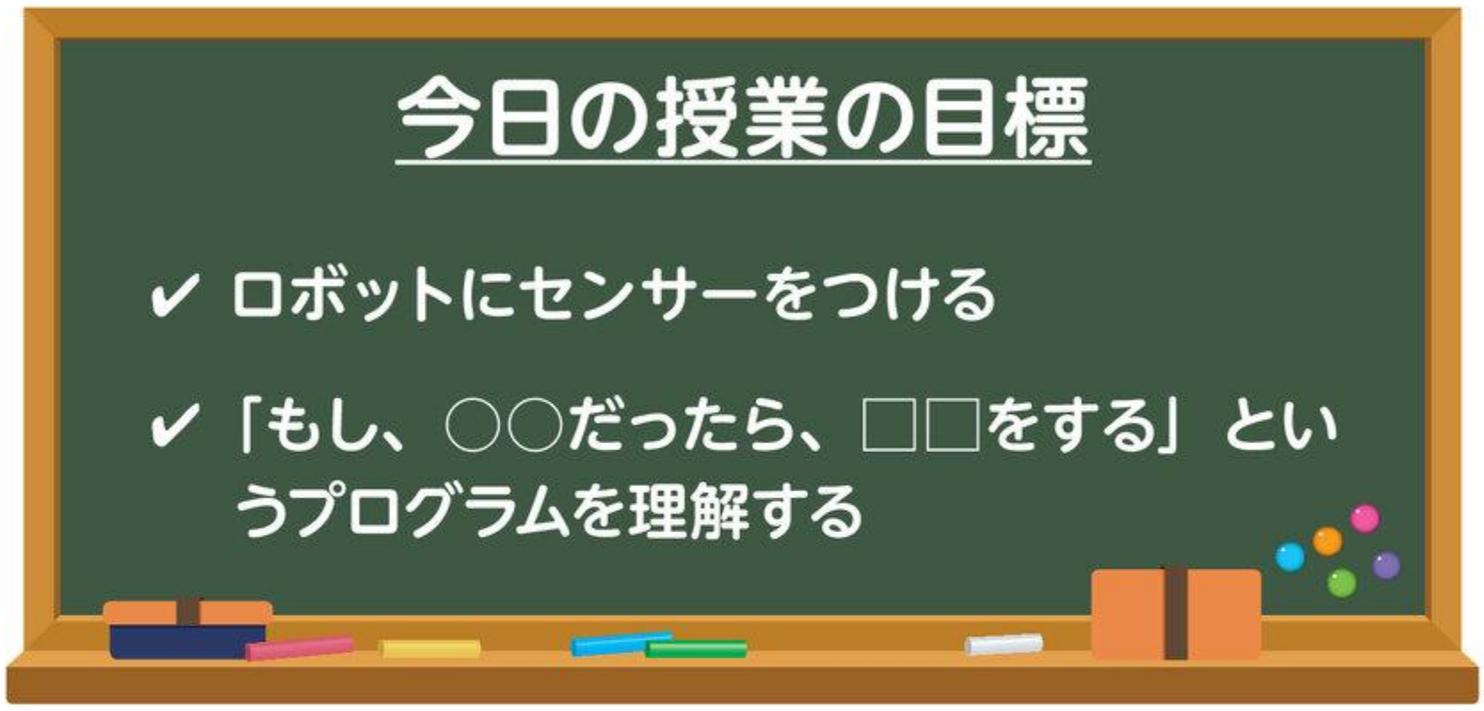
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

目安時間は授業時間 120 分のうち、休憩 10 分程度を取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0. センサーを使ったカシコイロボット（目安 5 分）

0.0. 「センサーを使ったカシコイロボット」でやること



今回は、オムニホイールロボットを改造して自動でかべ沿いに移動したり、障害物をさけて進んだりするカシコイロボットをつくります。そのために、センサーを使います。
センサーの反応でロボットの動作を切りかえることを体感しましょう。カシコイ機械が動作を行うときに基本となる、「もし、○○だったら、□□をする」という動作の仕方をくり返し考えながらプログラムを組んでいきます。
「感じて」、「考えて」、「動く」、よりカシコイロボットへの第一歩となります。

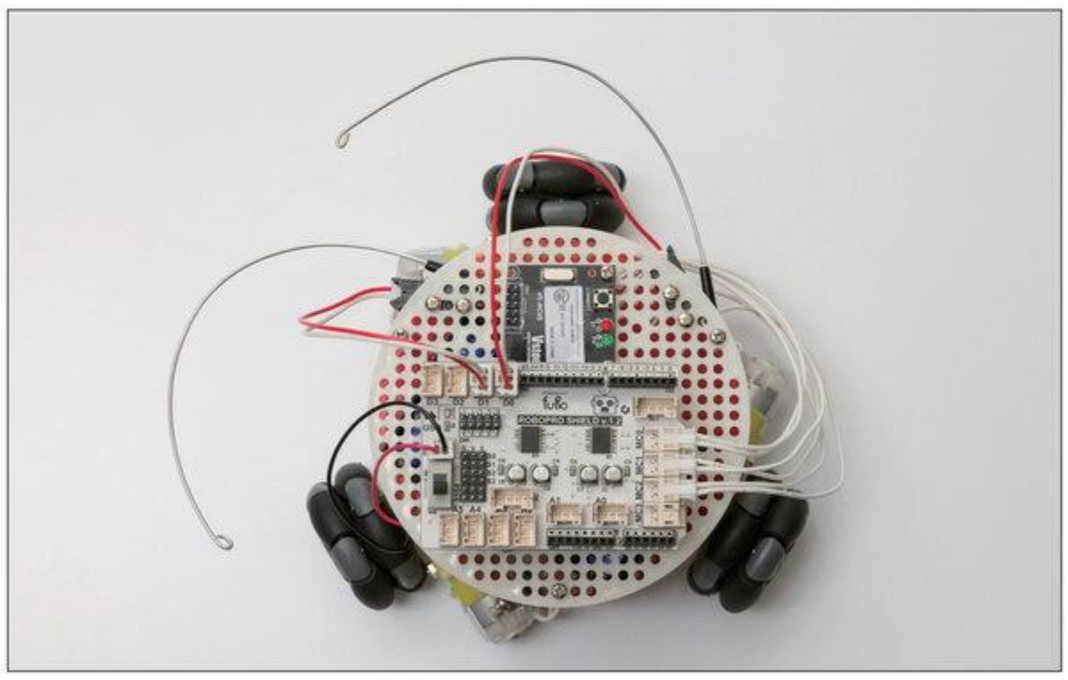
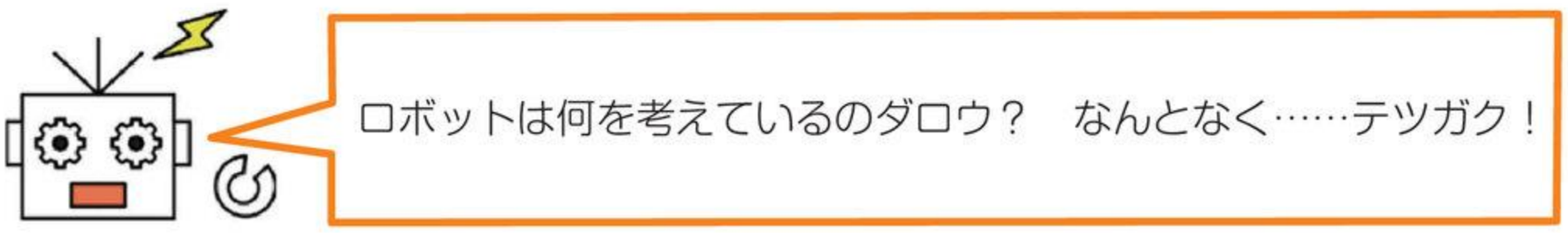


図 0-0　タッチセンサーを取り付けたオムニホイールロボット



0.1. 必要なもの

前回使ったロボットと、以下のパーツを準備しておきましょう。なお、リボンケーブルは今回使用しないため、取り外しておきましょう。

ラジオペンチ 1	ドライバー 1	USB ケーブル 1	タッチセンサー 2
			
100mm ビニールチューブ 1	**M2.6L8 タッピングネジ (B) 4**	**200mm 針金 2**	
			

図 0-1　必要なもの

講 今回は、リボンケーブルを付けたままだと誤作動を起こすことがあるので取り外しておきます。

1．触角センサーの取り付けと調整（目安 25 分）

今回使うセンサーはタッチセンサーです。タッチセンサーが使えるようになると、ロボットがかべなどに接触したことをマイコンで「感じる」ことができるようになり、よりカシコイロボットがつくれるようになります。それでは早速、オムニホイールロボットにタッチセンサーを取り付けていきましょう。

1.0. 触角センサーの取り付け

＜組み立て手順①＞

まず、ラジオペンチを使って、200㎜針金の両端を折り曲げて、触角をつくります。片側を**図 1-0** の左側のように「？」形に折り曲げて、引っかかりにくくしましょう。さらに、タッチセンサーに取り付けるもう一方の根元側（**図 1-0** の右側）も、少し長め（15mm 程度）に折りたたみます。これを２本つくります。

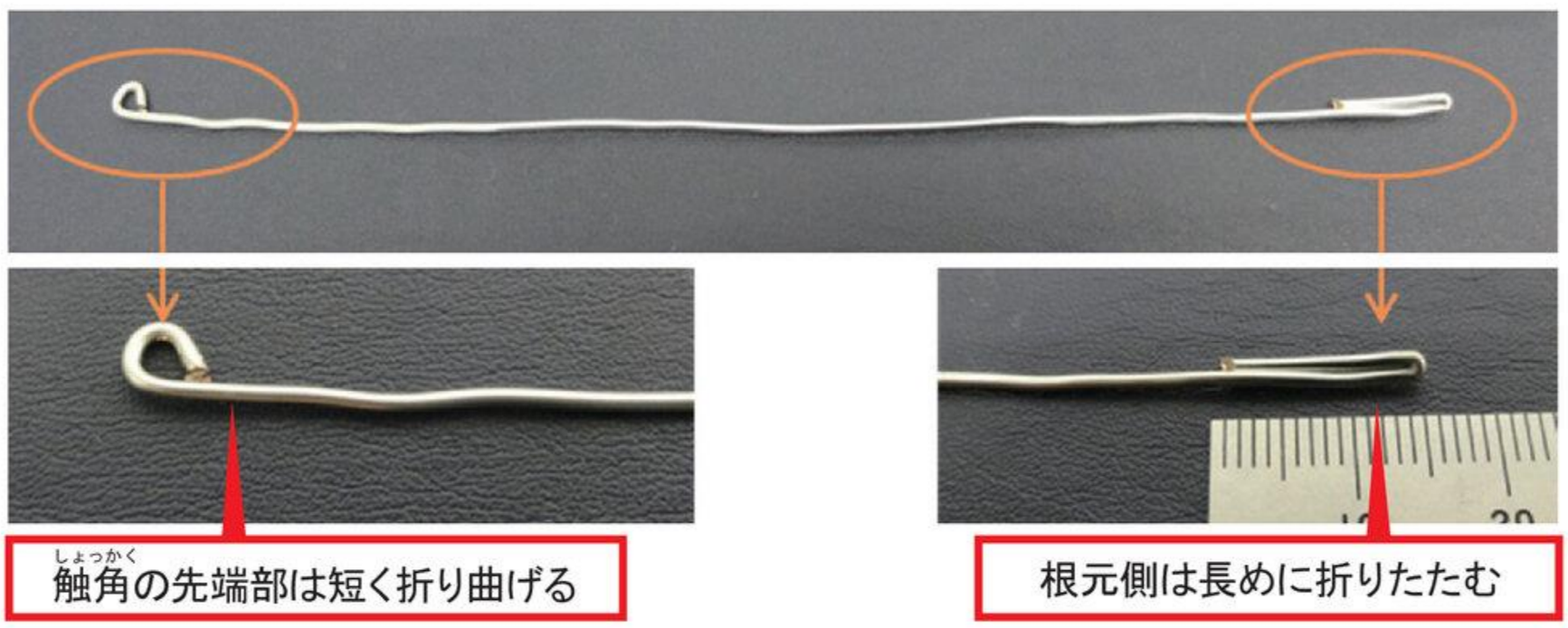


図 1-0　200mm 針金の折り方

＜組み立て手順②＞

次に、100㎜ビニールチューブを 15 ～ 20mm 程度の長さにカットします。

こちらも 2 本用意しましょう。

図 1-1　100mm ビニールチューブのカット

＜組み立て手順③＞

折り曲げた200㎜針金を**図 1-2** のように、100㎜ビニールチューブをカットしたものを使って、タッチセンサーに固定します。これを２セットつくります。

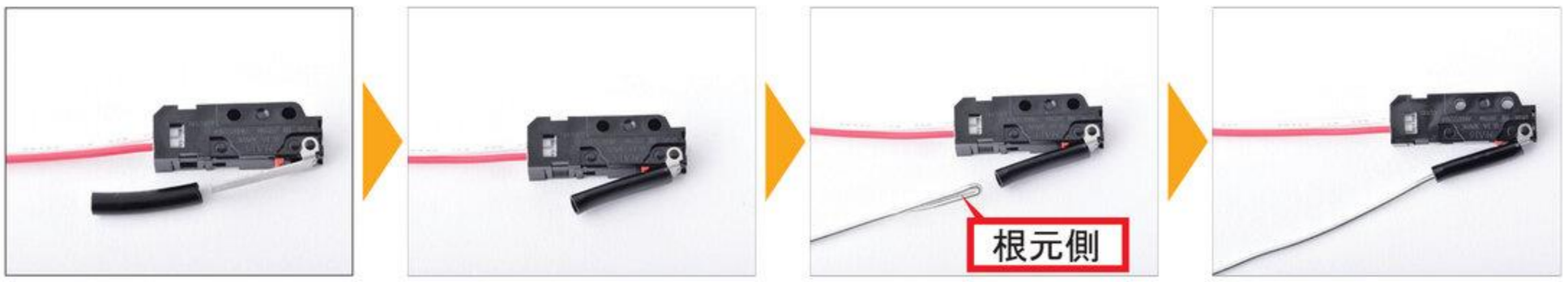


図 1-2　触角（しょっかく）センサーの作成

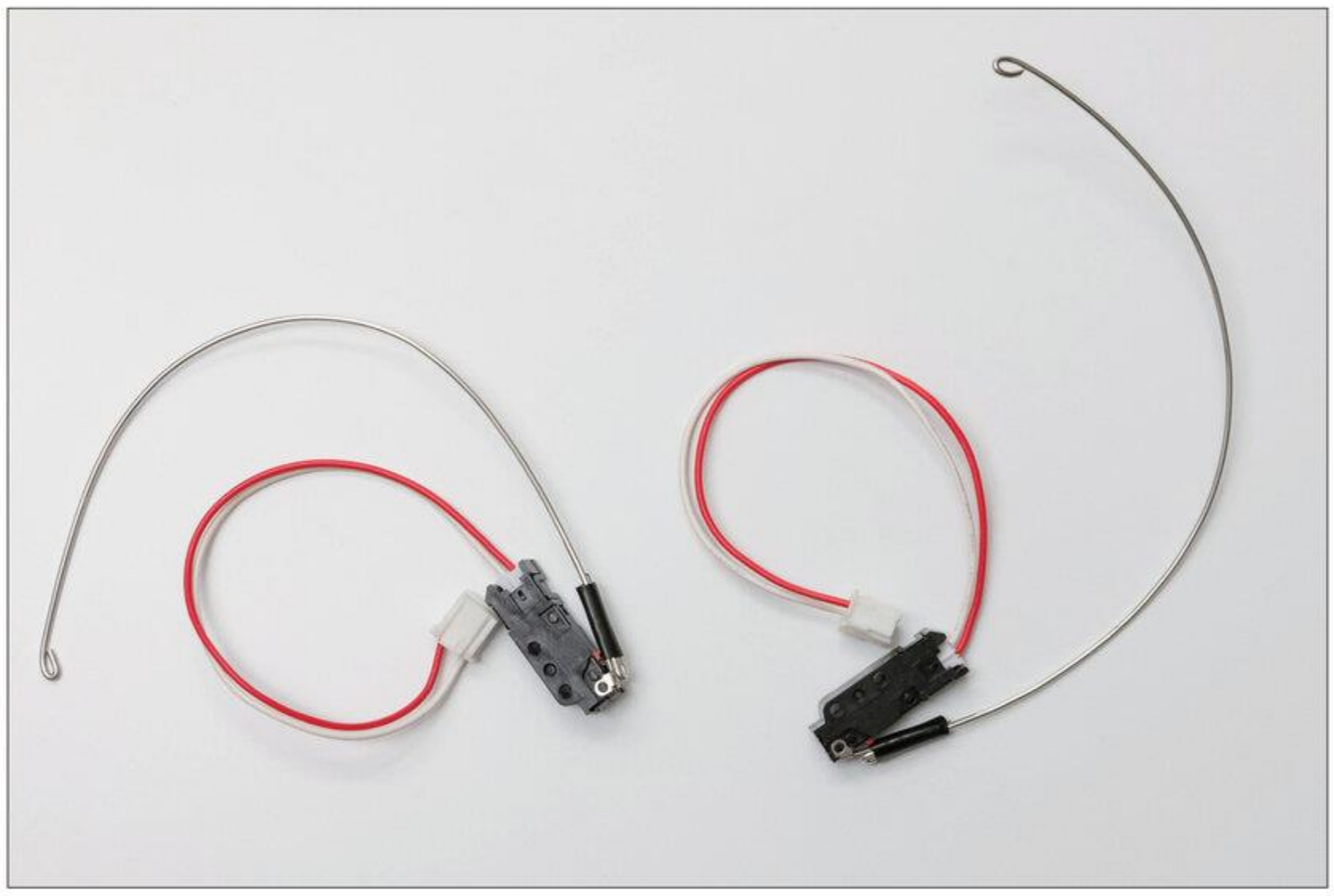


図 1-3　触角（しょっかく）センサーの完成図

講　針金の先端の折り曲げは、ラジオペンチに巻きこむ様に行います。固定用ビニールチューブに針金を入れる際に無理な力がかかるとタッチセンサーの金属プレートが外れたりすることがありますので、金属プレートを支えながら押し入れてください。

＜組み立て手順④＞

タッチセンサー（× 2）をロボット本体に取り付けます。取り付け位置は**図 1-4** を参考にしましょう。

タッチセンサー本体は白円形ボードの裏(うら)側に設置し、上から M2.6L8 タッピングネジ (B) で固定します。

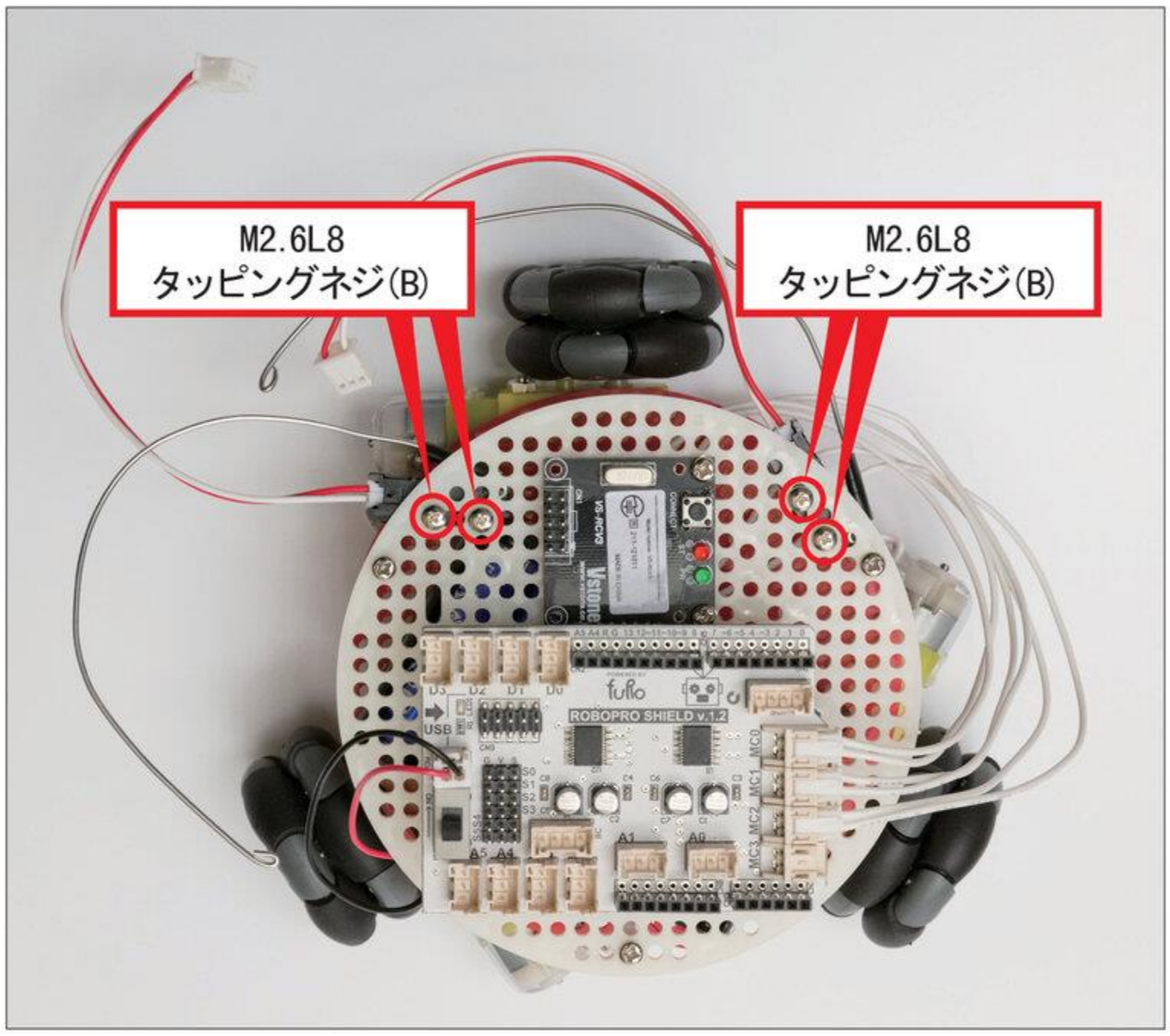


図 1-4　タッチセンサーを取り付けるネジの位置

＜組み立て手順⑤＞

タッチセンサーのケーブルをロボプロシールドに接続します。
前方方向に対して右側のタッチセンサーはロボプロシールドの D0 に、左側は D1 にそれぞれ接続します。

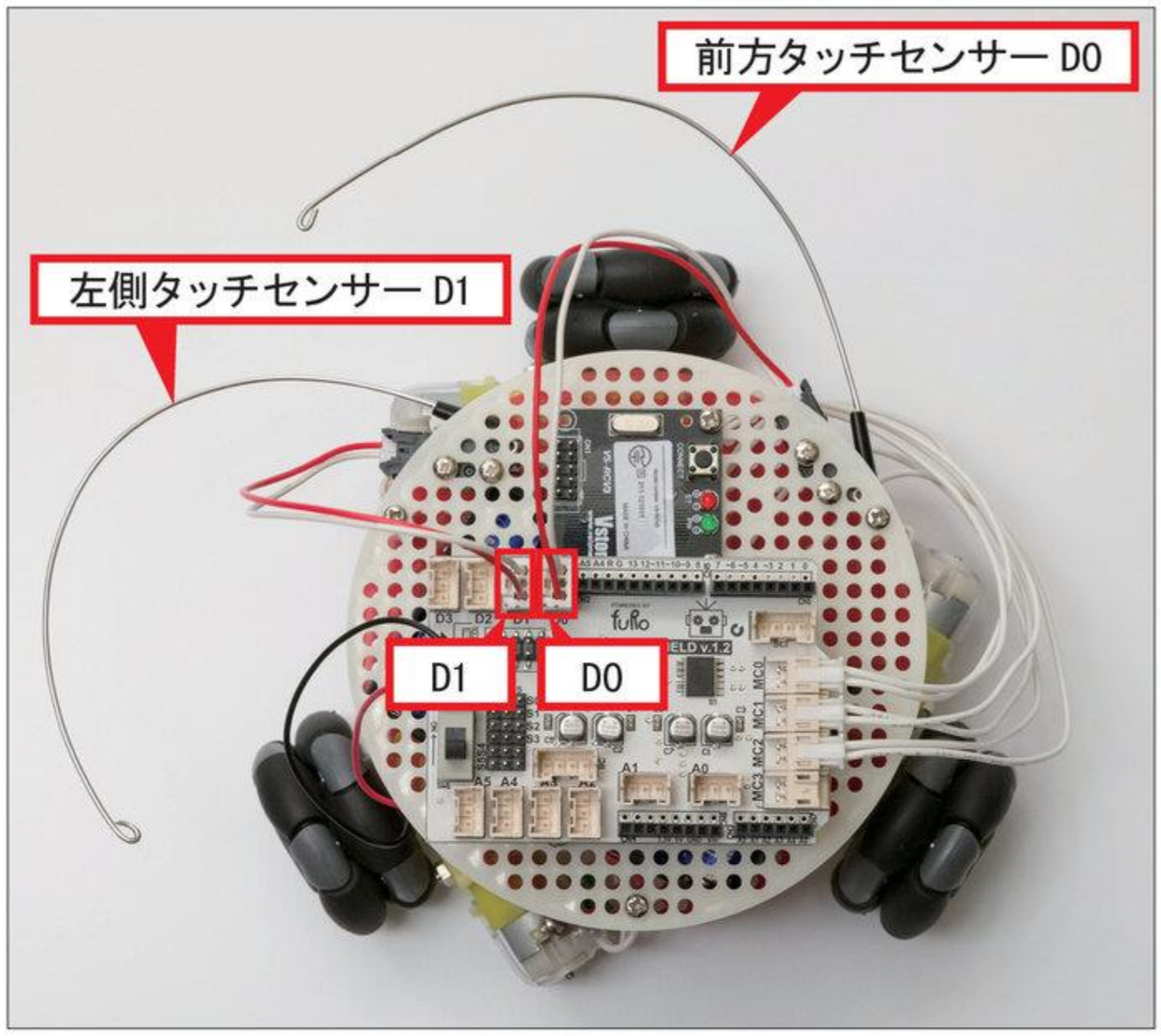


図 1-5　触角センサーの配線

＜組み立て手順⑥＞

触角の形を簡単に調整します。細かい調整は、後で実際のセンサーの反応を見ながらしましょう。ここでは**図 1-5** と似た形にしておきます。

これでタッチセンサーの取り付けはすべて終了です。
今回は、解説の都合上、D0 に接続しているものは「D0 センサー」、D1 に接続しているものは「D1 センサー」と呼びます。

講　タッチセンサーの差し込み場所を間違えないように注意してください。

1.1. 触角（しょっかく）センサーの動作確認

取り付けたタッチセンサーの動作確認をします。

1）D0 センサー

まずは、「D0 センサー」です。以下のプログラムを実行しましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > PreCourse > Switch0

図 1-6 の位置にあるマイコンボードの LED を横からのぞき込（こ）み、「前方タッチセンサー D0」を押します。オレンジ色の LED が点灯すれば成功です。

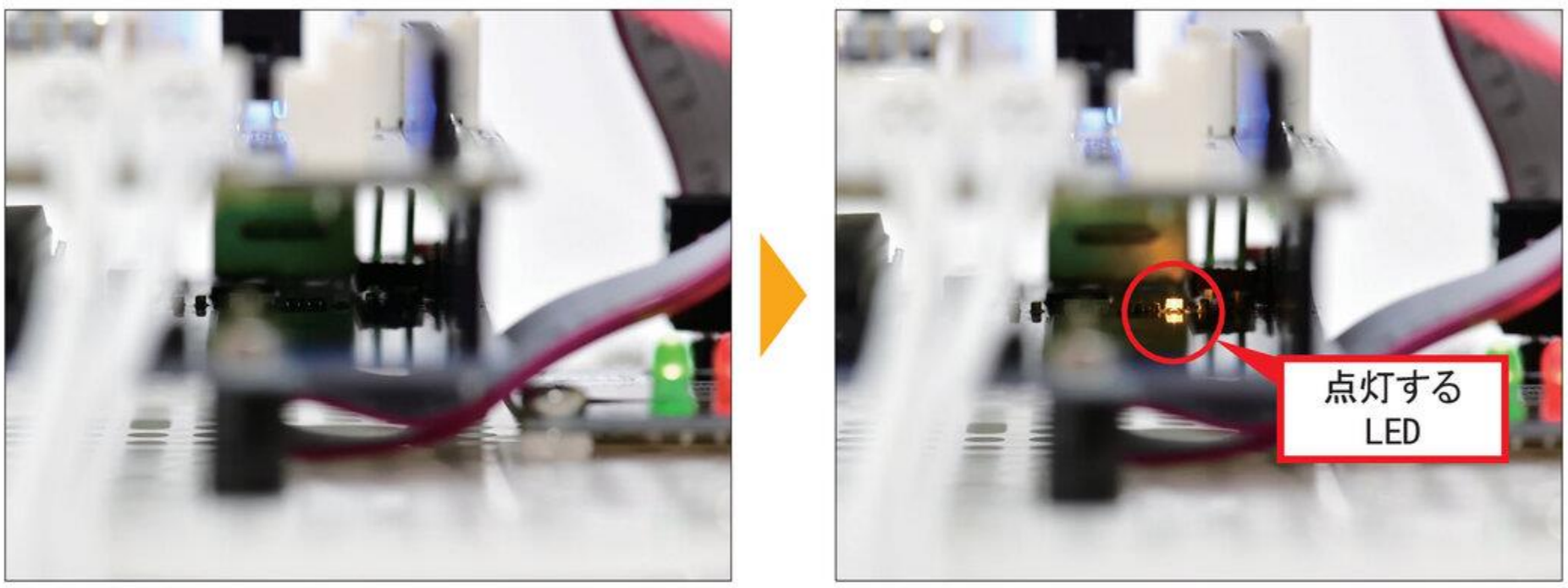


図 1-6　タッチセンサーの反応により点灯する LED

2）D1 センサー

今度は、「D1 センサー」です。以下のプログラムをマイコンボードに書き込（こ）み、反応を確認しましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > PreCourse > Switch1

２．センサーで感じて動くカシコイロボット（目安10分）

2.0.　カシコイロボットとは？

では、ここで実際にカシコイロボットに進化させてみましょう。でもその前に、まずはカシコイロボットの仕組みについて学んでいきましょう。そもそもどういうロボットが「カシコイ」といえるのでしょうか？

突然ですが､スタートアップ授業で勉強した「ロボットとは？」の話をおぼえていますか？ロボットとは、「感じて」、「考えて」、「動く」、機械のことでした。もう少しくわしく説明すると、センサーで「感じて」、マイコン（コンピューター）で「考えて」、モーターで「動く」、機械のことです。最低限これらができれば、カシコイロボットといえそうですよね。では、今回のロボットではどのパーツがそれぞれの役割を担当しているのか確認していきましょう。

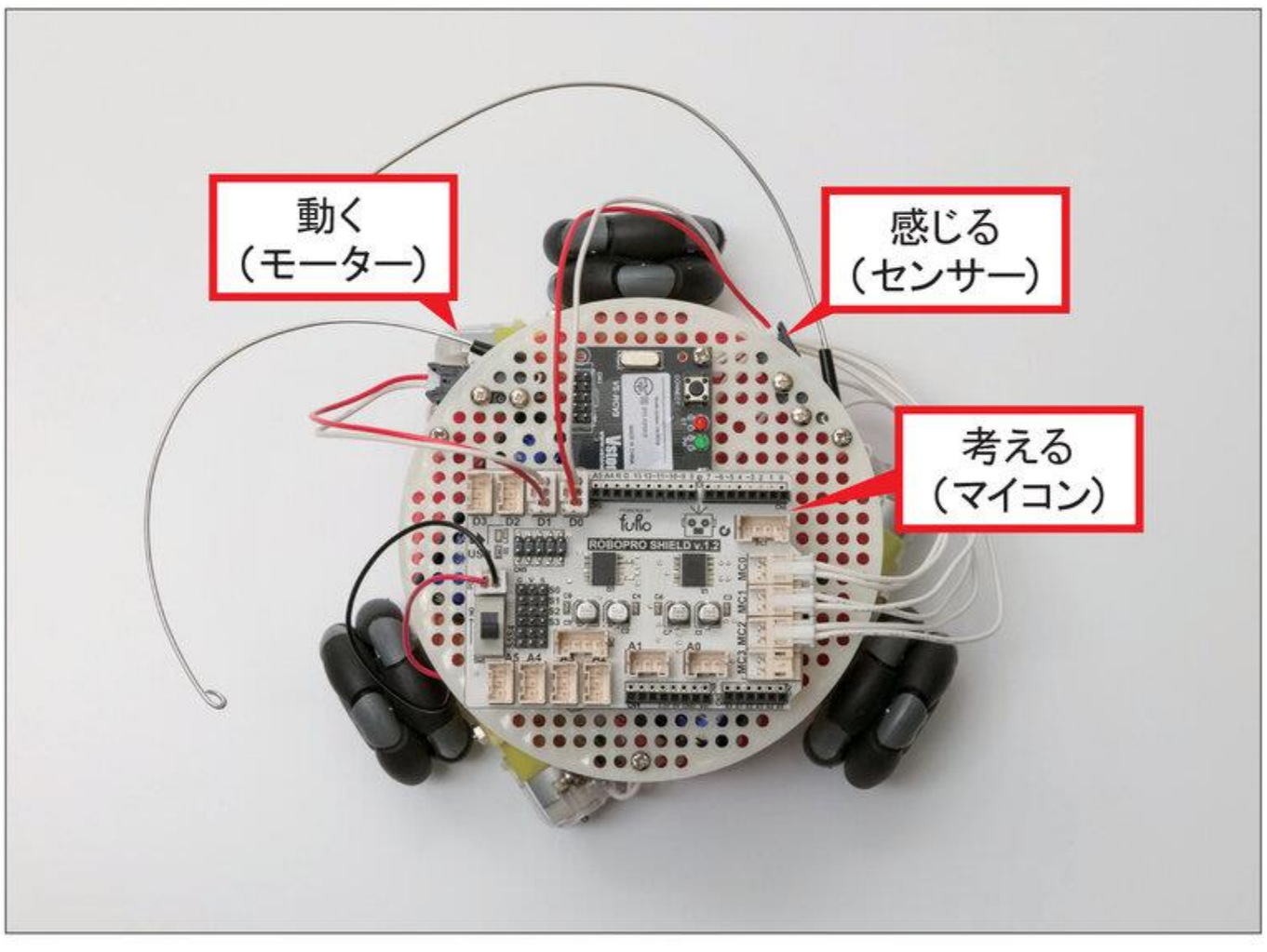


図 2-0　カシコイロボット

2.1. タッチセンサー

「感じて」の部分は、タッチセンサーが担当しています。タッチセンサーは、マイクロスイッチとも呼ばれ、スイッチ部の「押されている・押されていない」という状態によって「ON・OFF」が切りかわるセンサーです。とても軽い力で反応することが特徴で、種類も豊富なことから、ロボットのみならず、電化製品や組立工場に至るまで、いろいろなところで使われています。

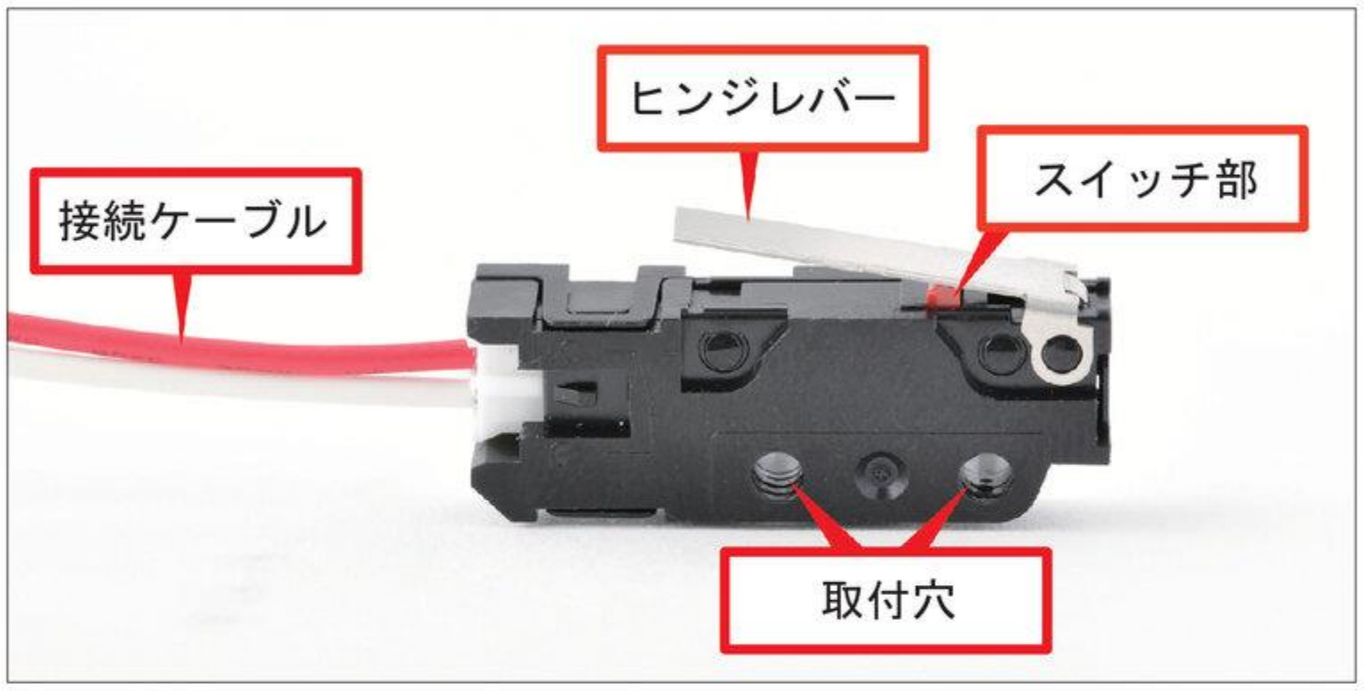
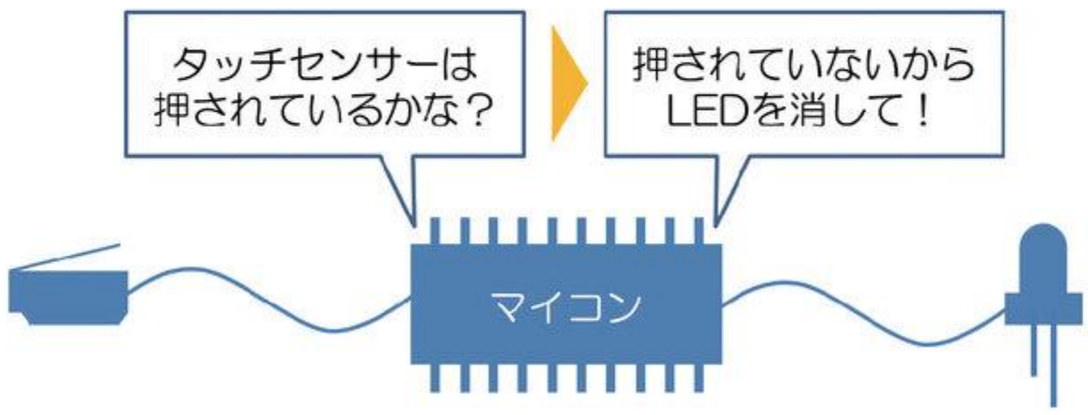


図 2-1　タッチセンサーの仕組み

2.2. マイコンによるタッチセンサーの監視

では次の「考えて」は、どのパーツが担当しているのでしょうか？ 書き込みを行った動作確認プログラムでは、マイコンが、タッチセンサーの状態を絶えず見張っていました。そして、「押されている・押されていない」という状態を検知して、その結果に応じて、今度はボード上の LED を「点灯・消灯」させていたのです。今回のプログラムでは LED が反応しただけでしたが、実際のロボットではマイコンの判断を受けてモーターなどが「動く」という役割を担当しているわけです。



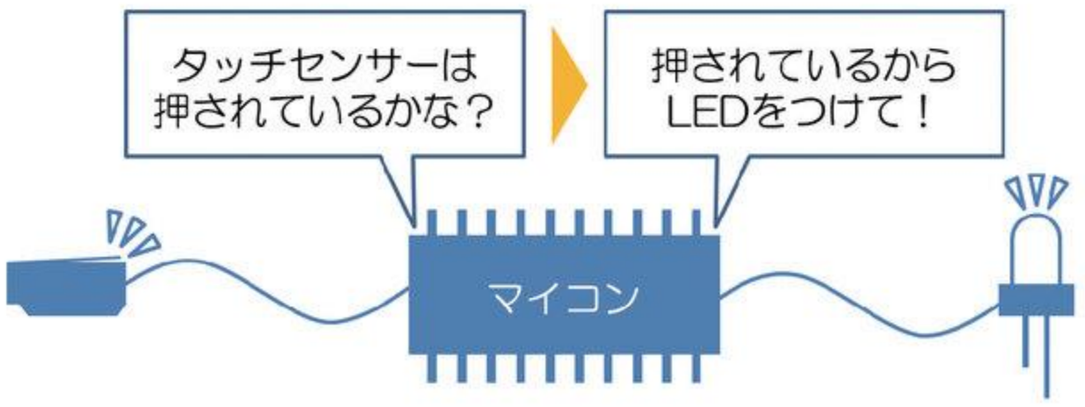


図 2-2　マイコンボードのはたらき

3. アルゴリズムとプログラム（目安 30 分）

3.0. アルゴリズム

タッチセンサーの監視（かんし）や、それに応じたLEDの「ON・OFF」の手順は、いわば命令書であるプログラムの中にかかれています。
こうしたプログラムに記述された、マイコンに行わせる一連の動作手順のことを「アルゴリズム」と呼びます。ロボットのカシコサの程度は、このアルゴリズムの内容によって決まります。

1）「もし○○だったら」ルール

アルゴリズムといっても、難しく考える必要はありません。今回は、もっとも基本となるルールをマスターしましょう。

POINT

もし○○だったら、□□をする。

このルールは、さまざまな機械で使われています。たとえば自動ドアは「もし人がいれば、ドアを開ける」ですし、冷蔵庫は「もし中の温度が上がってきたら、冷風を出す」ことで中を冷やし続けています。
また、機械やコンピューター以外でも、たとえば皆さんは出かけるとき「もし雨が降っていたら、かさを持っていく」でしょうし、車を運転する人たちは「もし信号が赤だったら、ブレーキをふんで停止する」でしょう。
「もし○○だったら」ルールは、私たちの生活でも日常的に登場しているものなのです。

豆知識

英語で「もし」は「if」、「○○ならば」は「then」といいます。そのため、「もし○○だったら」ルールのことを「if-then」ルールと呼ぶこともあります。

2）プログラムの確認

プログラム「Switch0」では、次のような「もし○○だったら」ルールがあります。

POINT

もしD0センサーが押されていたら、LEDを点灯させる。

このルールをプログラムにかくと、どのようになるのか見てみましょう。

プログラム「Switch0」より抜粋（ばっすい）

```
if(digitalRead(swPin) == HIGH){     // スイッチが入ったら
    digitalWrite(ledPin, HIGH);     // ledPinに接続されたLEDを光らす
}
```

命　　令「if」

実行内容：指定した条件が成り立つときだけ、指定の動作を行う

使 い 方：if(digitalRead(swPin) == HIGH){
　　// もしスイッチが押されていたら、{ } の動作を行う

if(digitalRead(swPin) == HIGH) というのが「もし○○なら」にあたる部分です。

( ) の中の digitalRead(swPin) == HIGH というのは、「 swPin と名付けたスイッチ（D0センサー）が押されている」ことを表す文です。

この通りの状況になっているときだけ、{ } の中の命令を実行します。今回は { } の中に「LEDをつける」という命令（緑の部分）がかかれているので、D0センサーが押されたときだけLEDがつくようになっているのです。

これは「if文」などとよばれる、プログラミングの世界できわめて有名な命令です。

しかし、「Switch0」の void loop() の中身を見てみると、if文以外の命令もかかれていますね。

プログラム「Switch0」より抜粋（ばっすい）

```
void loop(){
    if(digitalRead(swPin) == HIGH){   // スイッチが入ったら
        digitalWrite(ledPin, HIGH);   // ledPinに接続されたLEDを光らす
    }
    else{
        digitalWrite(ledPin, LOW);    // ledPinに接続されたLEDを消す
    }
}
```

if 文さえあれば LED をつけることができるはずなのに、水色の部分はなぜ必要なのでしょうか。
ためしに、この部分がなくなったときの動作を確認してみましょう。

やってみよう！

プログラム「Switch0」のうち、水色の部分を削除してみよう。
終わったらプログラムをロボットに書き込み、動作がどう変わったか確かめてみよう！
またその結果から、水色の部分がどのような命令なのか推理してみよう！

どう変わった？　LED が消えなくなった。

水色の部分はどんな命令？　もし D0 センサーが押されていなければ LED を消す。

プログラム「Switch0」の動作は、次の図のような流れになっています。

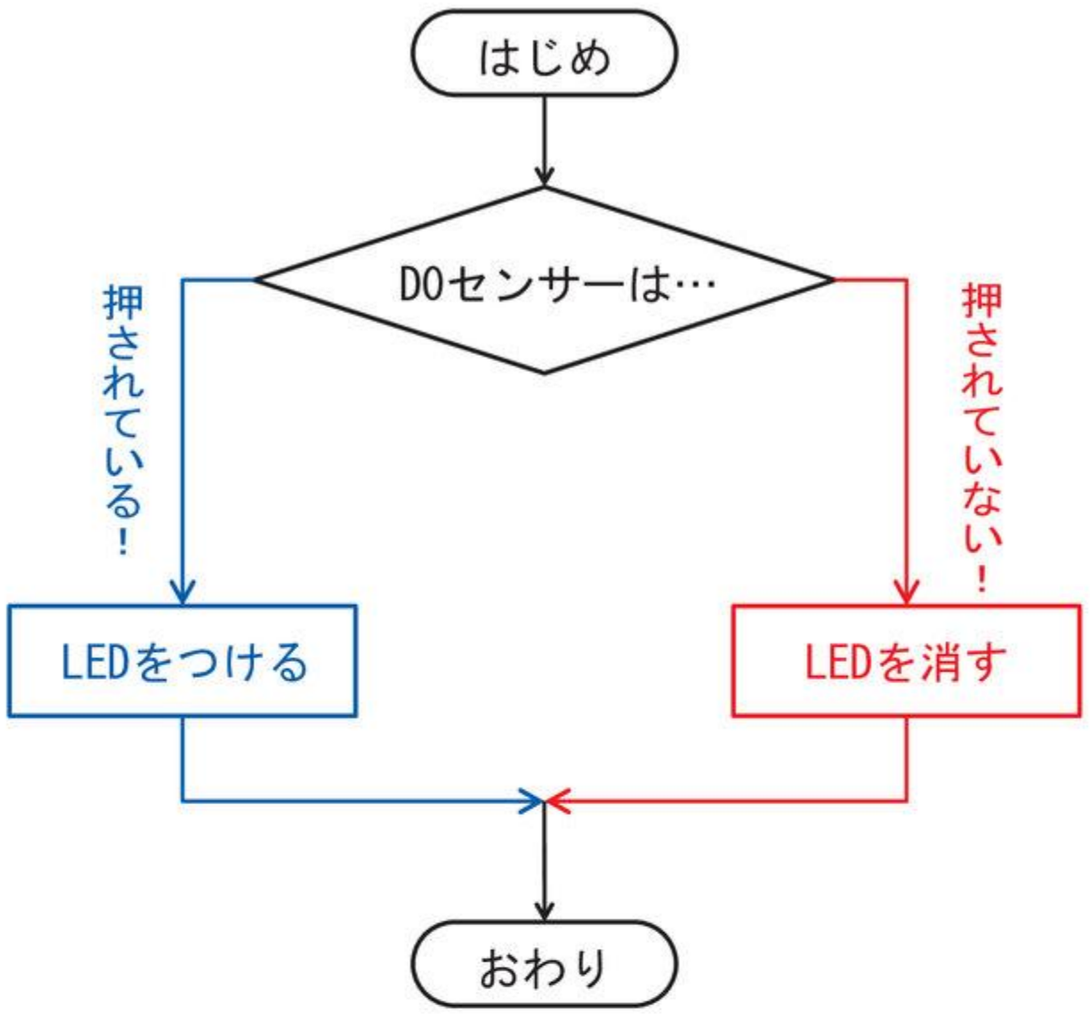


図 3-0　プログラム「Switch0」の流れ

D0 センサーが押されているときは青、押されていないときは赤のルートに入ります。1 回のループで、LED をつける命令か消す命令、どちらか片方しか実行できないことがわかりますね。

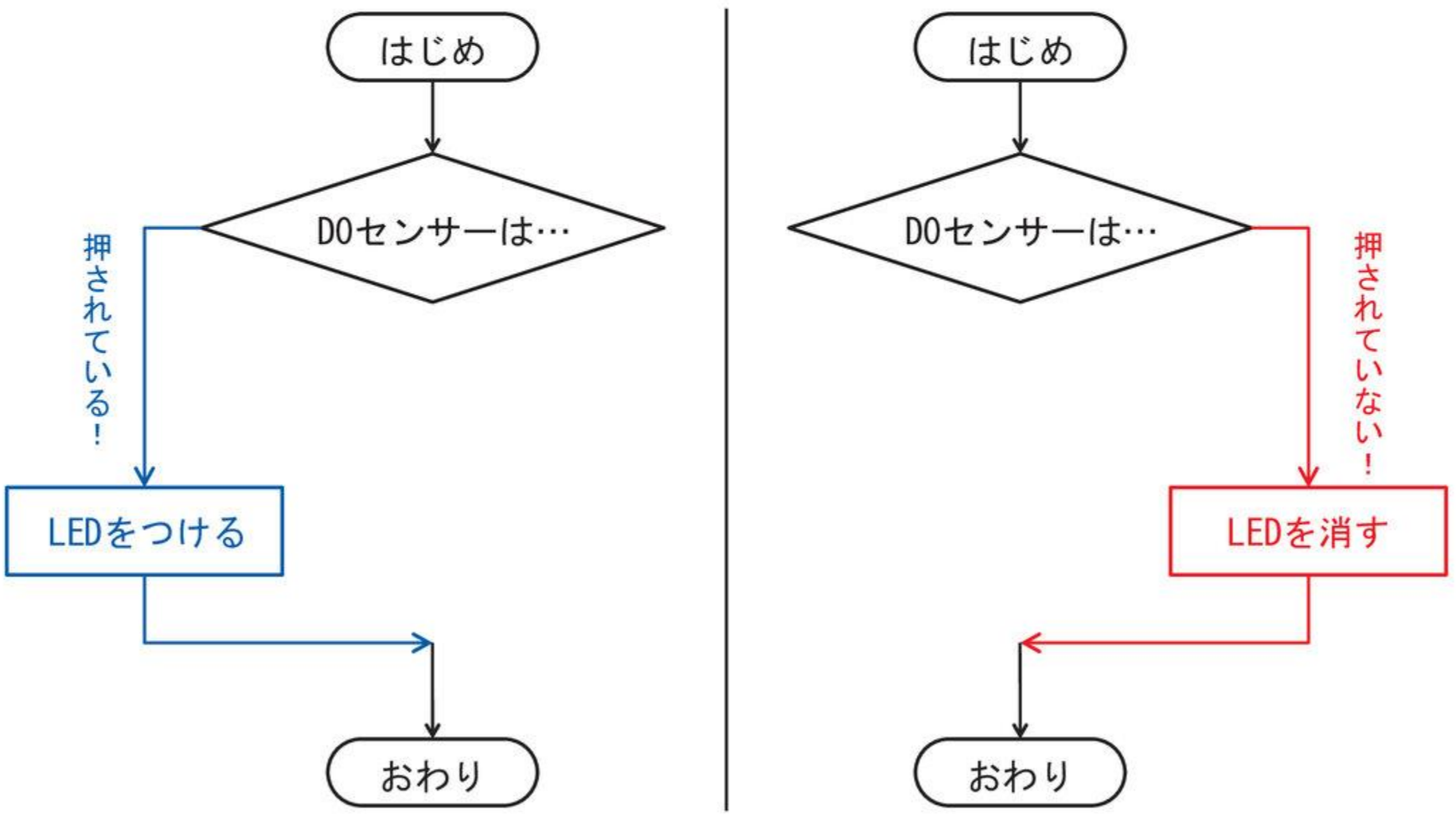


図 3-1　どちらか片方のルートしか通れない

オムニホイールロボットのコンピューター（マイコン）は、人間には意識できないほどのスピードでこれを何度もくり返しています。そのため、D0 センサーが押されている間は LED がつきっぱなしになっているように見えるのです。

ところで、さきほど if 文以外の部分を消したときに動きがおかしくなったのではないでしょうか。具体的には、D0 センサーから手をはなしても LED が消えなくなってしまいましたね。これは、消した部分が**図 3-0** の赤のルートにあたる部分だったからです。

3）else

あらためて、消した部分を確認してみましょう。

プログラム「Switch0」より抜粋（ばっすい）

```
else{
    digitalWrite(ledPin, LOW);      // ledPinに接続されたLEDを消す
}
```

命　　令「else」
動作内容：直前の if 文が「いいえ」だったとき、指定の動作を実行する
使 い 方：else{　// もしスイッチが押されていなければ、{ } 内の動作を行う

「else」は英語で「○○でなければ」という意味です。if 文の後ろにかいておくと、「もし○○でなければ」という命令にすることができます。{ } の中身は「LED を消す」という命令です。まとめると、「もし D0 センサーが押されていれば、LED をつける。そうでなければ、LED を消す」という意味になりますね！
この部分を消してしまえば、ロボットは「LED をつける」ことはできても、「LED を消す」ことはできなくなってしまうことがわかります。

4）フローチャート

図 3-0 のように、命令や処理がかかれたブロックを矢印でつないだ図を「フローチャート」といいます。
「Switch0」のように if と else を使っている場合、実際には以下のような形でかかれることが多いです。

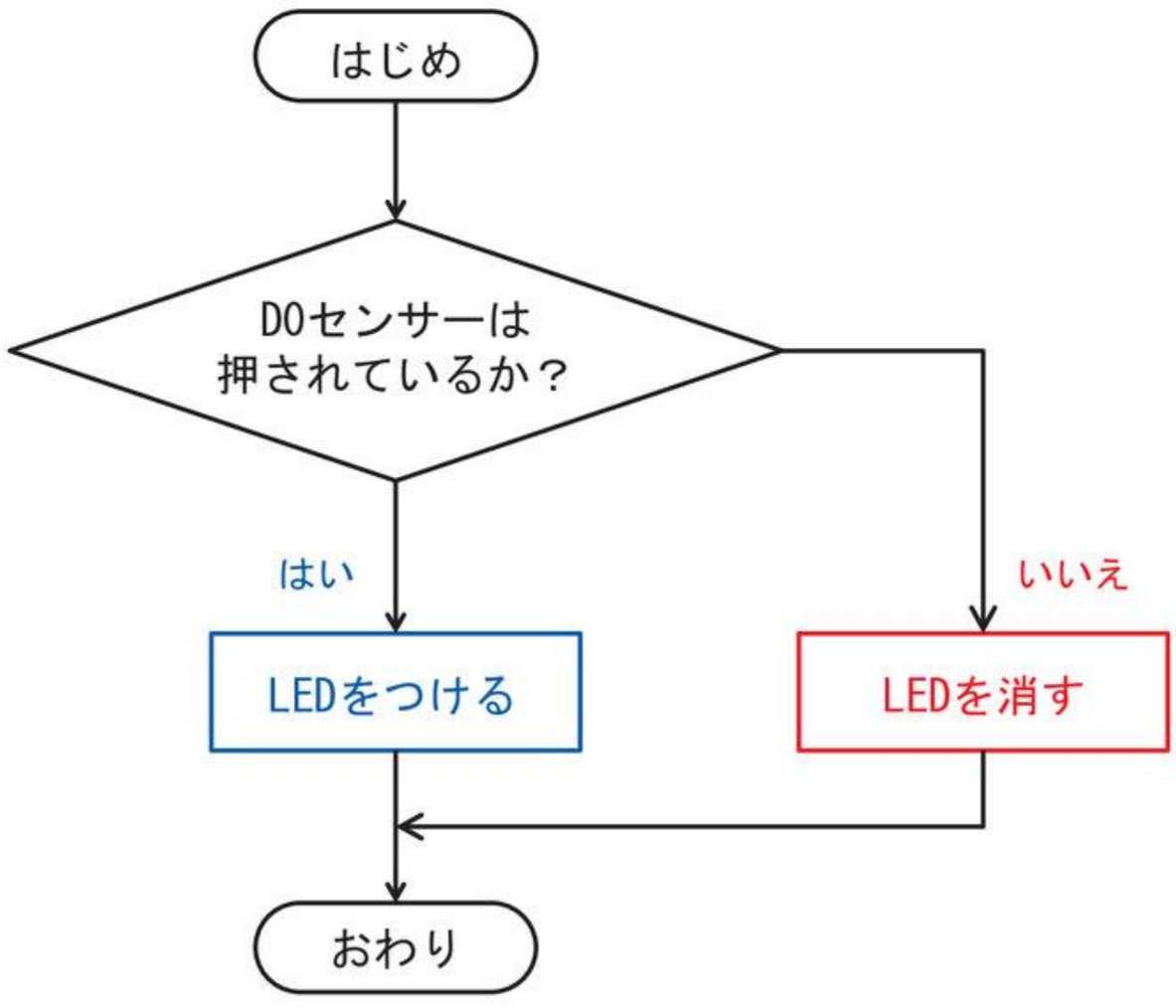


図 3-2　プログラム「Switch0」のフローチャート

3.1. センサーを使ったロボットを動かす

それでは、実際にロボットの動きが変化するプログラムを試してみましょう。プログラムには、先ほど学んだ if 文が記載されています。このプログラムによって動くロボットの様子を、実際に動かしながら体験してみましょう。
では、次のプログラムを書き込んで動作を確認してください。

プログラムの書き込み

RoboticsProfessorCourse1 > OmniWheelRobot5 > Sensor1

POINT

なお、この回でもロボットの調整値も忘れずに修正してから実行しましょう。

```
RPomniDirect omniBot(1.0f,1.0f,1.0f,50.0f);
```

実行結果：ロボットは何もなければ前進し、目の前にかべがあるとしばらくの間後退する。ロボットを机に置いて電源を入れ、適当なかべをロボットの前に置いてみてください。行ったり来たりをくり返す動作になるはずです。

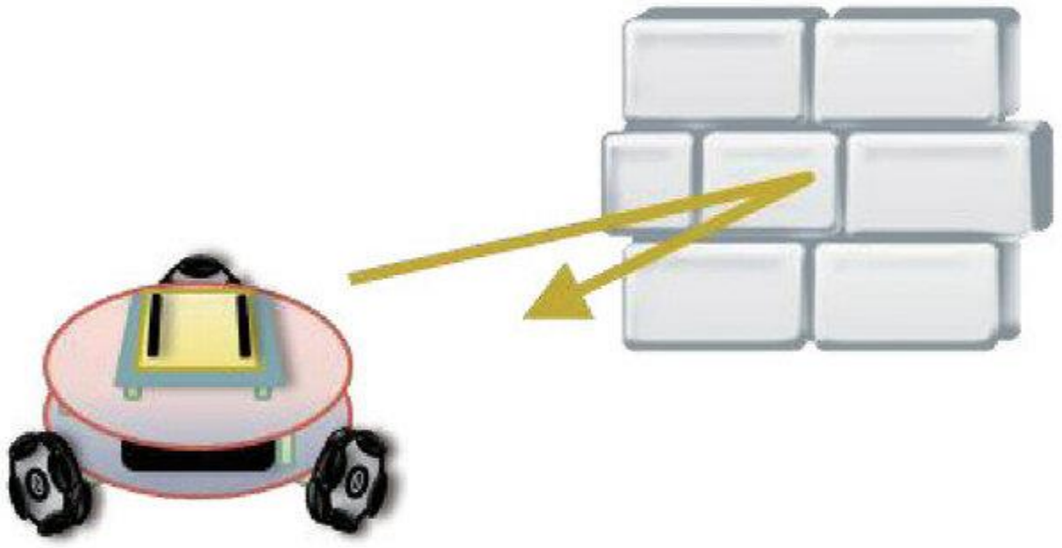


図 3-3　前進・後退の切りかえ動作

講　プログラム「Sensor1」でセンサーの基本的な働きを理解させます。

3.2. プログラム

1）アルゴリズムを翻訳（ほんやく）する

プログラム「Sensor1」のアルゴリズムを考えてみましょう。
かべに当たるまでは前に進み、かべにあたったら後ろに戻っていたので、次の**図 3-4** のようなフローチャートになるはずです。

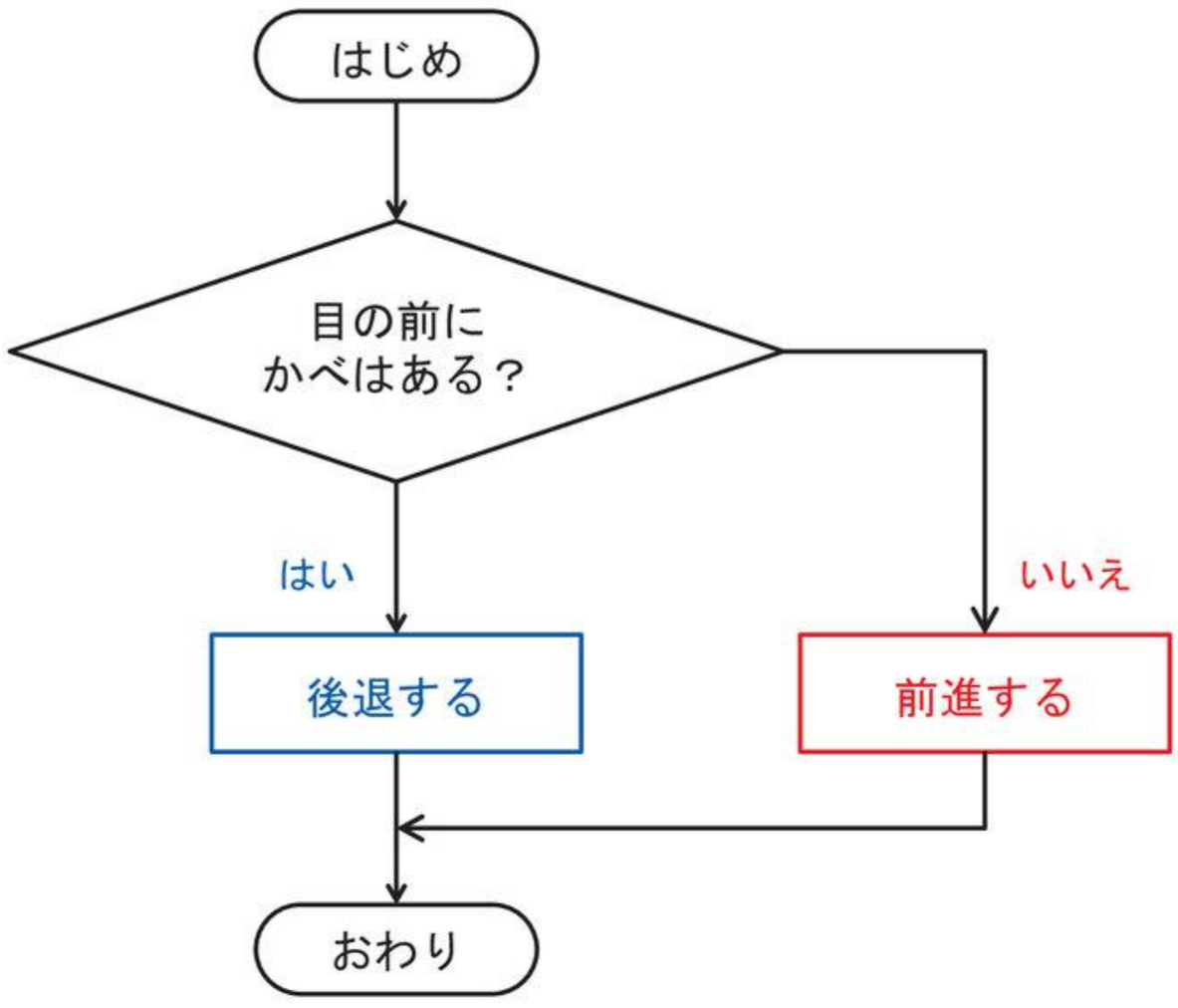


図 3-4　プログラム「Sensor1」のフローチャート

さて、いきなりですが、このフローチャートを少し改良してみましょう。
といっても、ブロックを増やしたり、矢印のつなぎかたを変えたりするわけではありません。ここで必要なのは「ロボットにも理解できるかき方にする」ということです。

実は、ロボットは「そんなの説明されなくてもわかるだろう！」と言いたくなるほど融通（ゆうずう）がききません。つまり、自分の知らないことを自分なりに考える、ということが全くできないのです。
たとえば、**図 3-4** のフローチャートをそのままロボットに渡したとして、「目の前にかべがある」という状況はロボットには理解できません。
ロボットに命令するためには、「目の前にかべがある」というのをどう言いかえれば、ロボットにも理解できる内容になるか、を考えなければならないのです。

図 3-5　ロボットはアドリブに弱い

今回であれば、D0 や D1 のタッチセンサーが押されている、という状況なら、ロボットにも理解できます。ですから、次のようなフローチャートにしておくと、ロボットにとっては「わかりやすい」と言えますね。

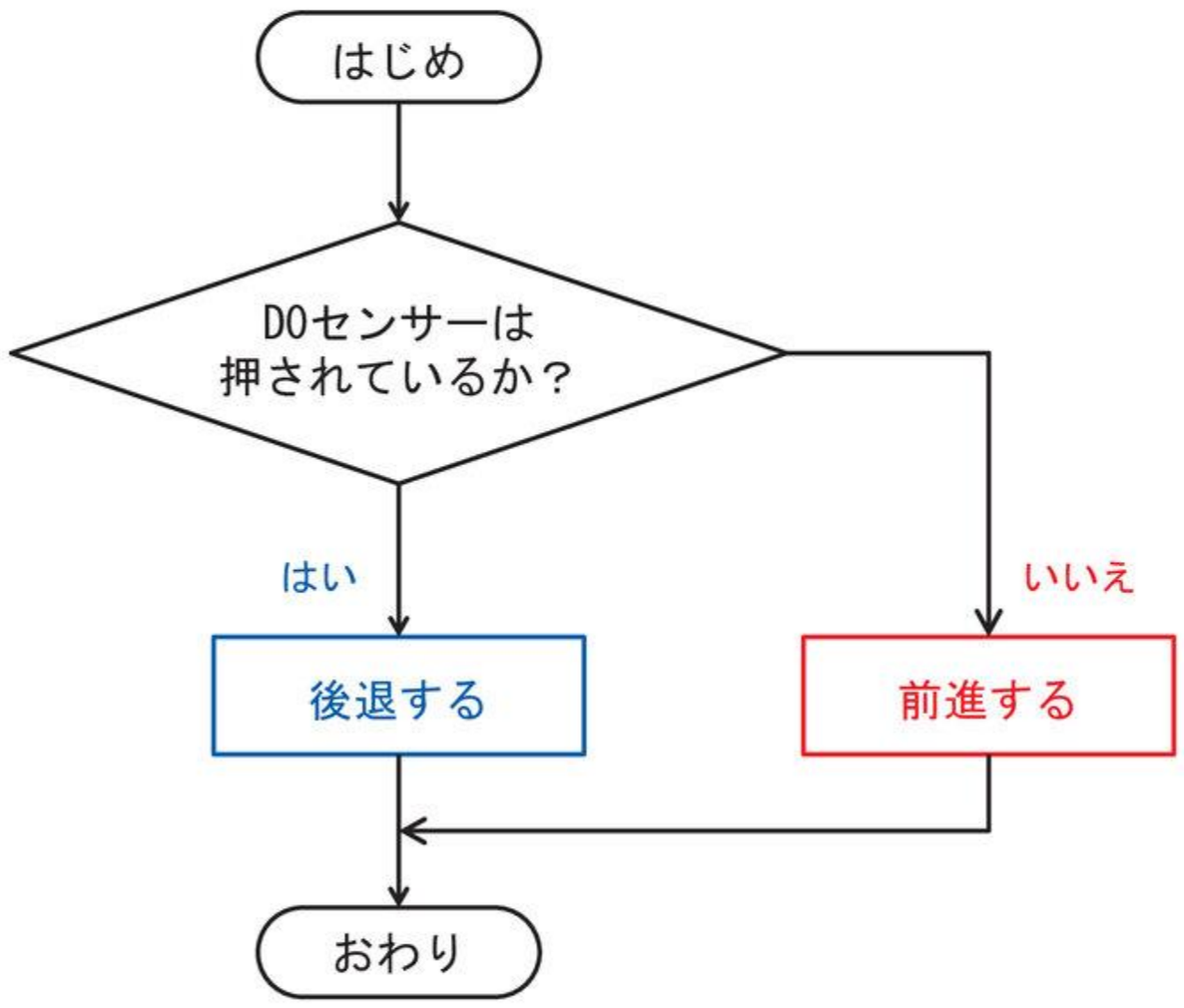


図 3-6　プログラム「Sensor1」のフローチャート（ロボット用）

もちろん、私たちが「どんな動きにすればいいのかな」と考えるときであれば、**図 3-4** のフローチャートでも全く問題ありません。むしろこちらのほうが考えやすい、という人も多いでしょう。
しかし、プログラムをかくときには**図 3-6** のようなフローチャートが役立つことが多いのも確かです。どちらの視点も持っておくことが大切ですね。

2）「if-then」ルール

では、プログラムを見ていきましょう。

プログラム「Sensor1」より抜粋（ばっすい）

```
void setup(){
    pinMode(D0, INPUT_PULLUP);      // D0を入力にする
}

void loop(){
    if(touchSensor(D0) == ON){      // もし前方タッチセンサー D0がONだったら
        omniBot.move(0, -20, 0);    // 後退する
        delay(1000);                // 1秒間休憩
    }

    else{
    // そうでなかった(前方タッチセンサー D0がOFFだった）ら
        omniBot.move(0, 20, 0);     // 前進する
        delay(10);                  // 0.01秒間休憩

    }
}
```

今回も if と else が登場していますね。 if の条件である touchSensor(D0) == ON は「D0 センサーが押されている（ON である)」という意味です。このときだけ、{ } 内の命令を実行しています。

omniBot.move(0, -20, 0); は覚えていますか？　はじめの値で左右の速度、2 つ目の値で前後方向の速度、3 つ目の値で回転速度を決める命令でしたね。今回はマイナスがついているので、速度 20 で後退という命令になっています。

ON を OFF にかきかえれば、「押されていない（OFF である）とき」を判定することもできます。

D0 を D1 にかきかえれば D1 センサーを使うこともできます。ただし、その場合は void setup() 内も以下のようにかきかえてください。

```
void setup(){
    pinMode(D1, INPUT_PULLUP);
}
```

4. プログラムの応用（目安 35 分）

4.0. かべ回避動作

プログラム「Sensor1」はとてもシンプルなプログラムですが、if-then ルールを使っているので、工夫しだいで様々な状況に対応できます。
たとえば、「Sensor1」のロボットはかべから逃げることはできますが、前進と後退をくり返すだけなのでまた同じかべにぶつかってしまいますね。これを、以下のような動きに変えることで「改良」してみましょう。

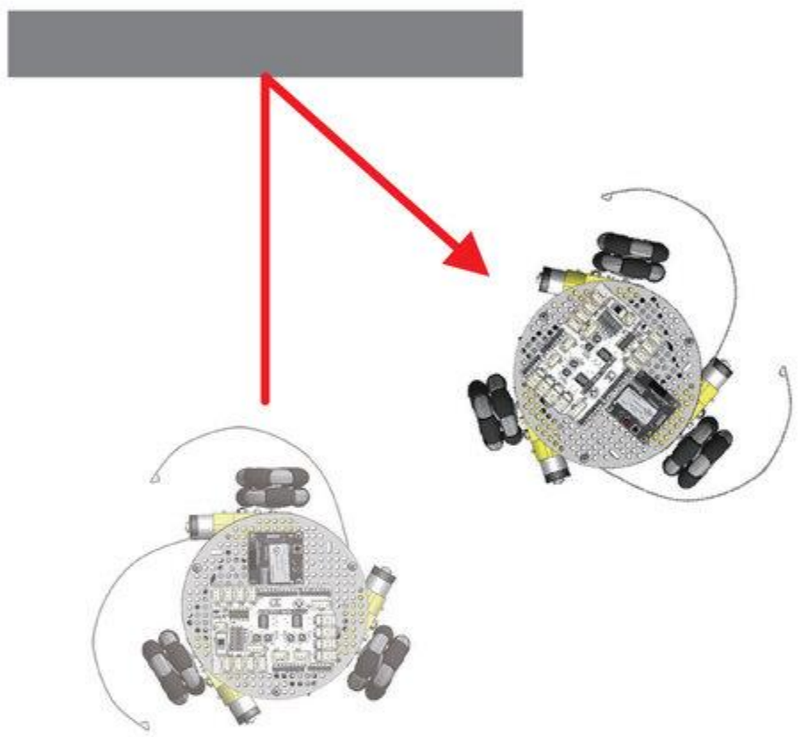


図 4-0　かべを回避するロボット

やってみよう！

プログラム「Sensor1」をかきかえ、**図 4-0** のようにかべから逃げるロボットにしてみよう！

ヒント

「Sensor1」のフローチャートのうち、どのブロックをどのように変更すればよいか考えてみよう！
プログラムの中でかきかえるべき場所がどこなのか、すぐわかるようになるよ！

講

解答例は以下のプログラムです。
RoboticsProfessorCourse1 > OmniWheelRobot5 > Sensor2

壁に当たった（D0 センサーが押された）とき、後退する代わりにその場で回転するように変更しています。

4.1. かべ沿い動作

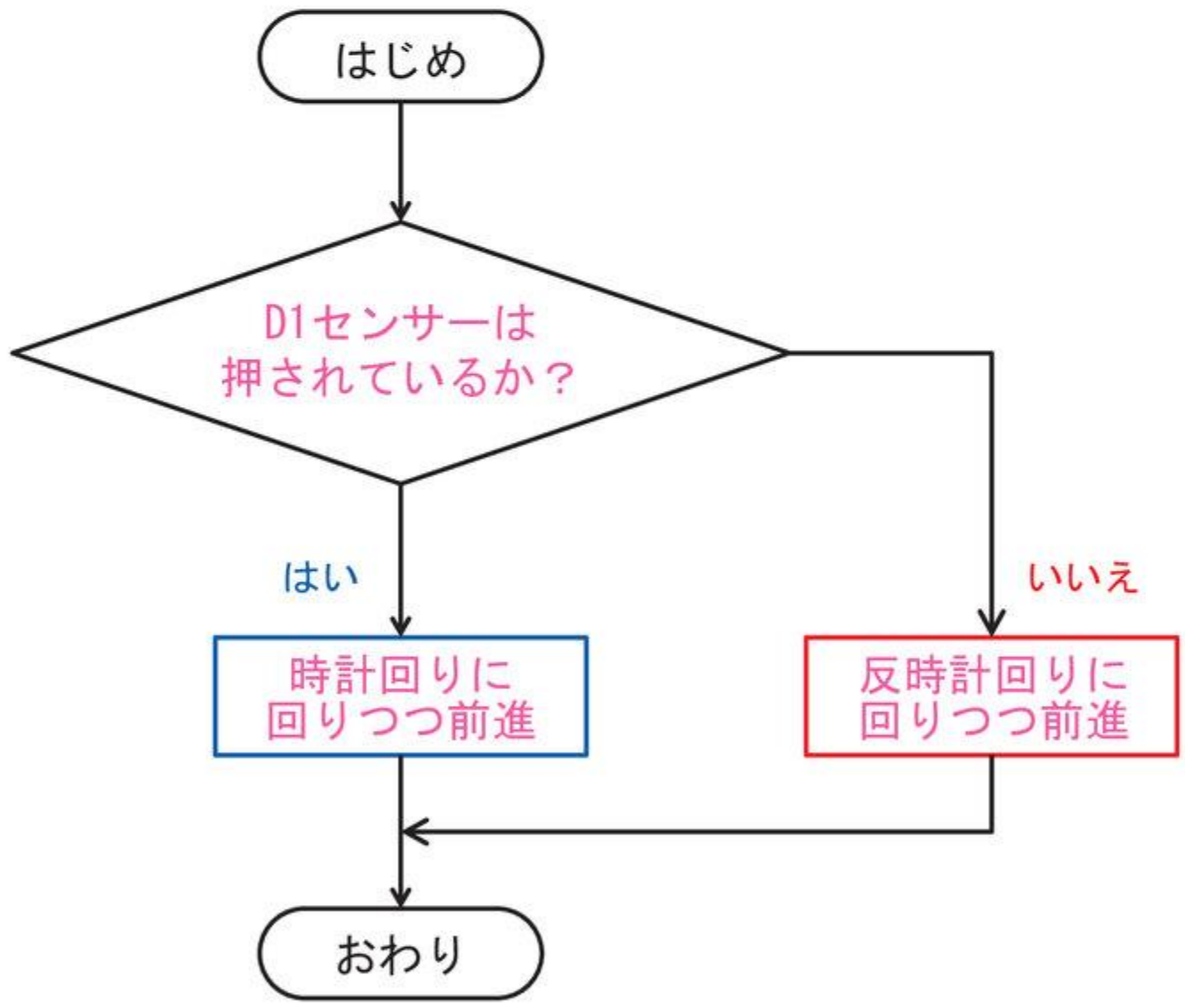
たったいま「かべ回避ロボット」づくりで実感したと思いますが、ロボットをつくるときには、「この動作を実現するためには、ロボットにはどのような命令を出せばいいんだろう？」という視点が必要です。

続いては、次のようなロボットをつくります。どのような命令が必要か、具体的に考えてみましょう。

図 4-1　かべ沿いロボット

やってみよう！

図 4-1 のようなかべ沿いロボットをつくるには、どのような流れのプログラムが必要かな？
フローチャートの空欄をうめてみよう！



ヒント

かべとの距離を保つには、かべに近づいたりかべから離れたりをくり返す必要があるね！

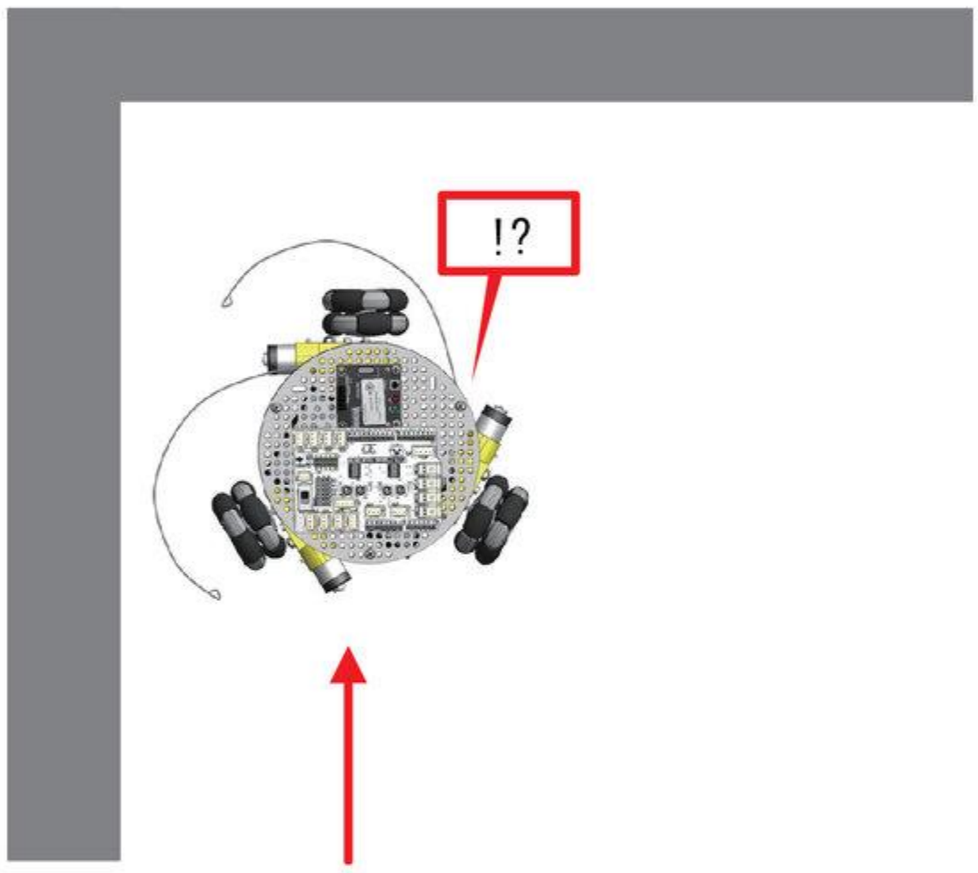
ステップアップ

「やってみよう！」でつくったフローチャートを参考にプログラム「Sensor1」をかきかえ、かべ沿いロボットのプログラムを完成させよう！

講　解答例は以下のプログラムです。
RoboticsProfessorCourse1 > OmniWheelRobot5 > Sensor3

チャレンジ課題

かべ沿いロボットは横にだけかべがあるときには正しく動作するけど、正面にもかべがあるとお手上げだね！



では、「Sensor1」をかきかえて、上のような状況にも対応できるロボットをつくってみよう！

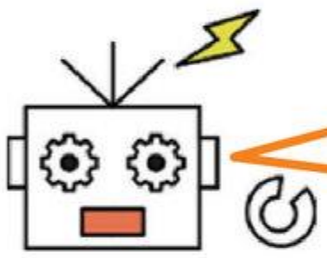
ヒント

正面も横も見なければならないね！
フローチャートも今までより少し複雑（ふくざつ）になるよ！

講　解答例は巻末に記載します。

5．まとめ（目安５分）

ここでは、タッチセンサーを使って自分で判断して動作するカシコイロボットについて学びました。カシコイロボットを実現するための動作手順を表したアルゴリズムも、実は意外なほど簡単であることがわかったはずです。また、このアルゴリズムを実現する「if-then」ルールによるプログラムも十数行で記述でき、ほんの少し変更（へんこう）しただけで、ロボットの動きがどんどんカシコクなりました。
そう、重要なのは、アルゴリズムであり、プログラムはアルゴリズムをロボットに行わせるための手段なのです。みなさんも、アルゴリズムを理解して、もっとカシコイロボットをつくってみましょう！



つまりアルゴリズムが「アルファ」という名の始まりであり、
プログラムが「オメガ」という名の終わりである。……う〜ん、テツガク！

講

○　以下の授業の目標を再確認します。
　・ロボットにセンサーをつける
　・「もし、○○だったら、□□をする」というプログラムを理解する
○　今回の授業で学んだ感想や面白かったことなどを、生徒から聞いてみましょう。
○　次回テーマは「ロボット競技大会」であることを告知します。
　ペットボトルやボールを使用した障害物競争やサッカー競技を行うので、ペットボトルやボールを持っている生徒に持参していただくようにお声がけください。

≪次回必要なもの≫

次回は、今回使ったロボットと以下のパーツを持ってきてください。
タッチセンサーは取り外し、代わりにリボンケーブルを接続しておきましょう。

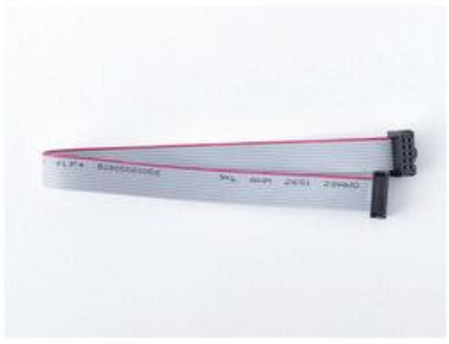
USB ケーブル　1	リボンケーブル　1	コントローラー　1
		

図 5-0　次回必要なもの

P.21　チャレンジ課題　解答例

解答例は以下のプログラムです。
RoboticsProfessorCourse1 > OmniWheelRobot5 > Sensor4
かべ回避とかべ沿いの動きを組み合わせることで、正面と横のどちらに壁があっても対応できるようにしています。
D0 センサーと D1 センサーを同時に使うので、`void setup()` 内に `pinMode(D0, INPUT_PULLUP);` と `pinMode(D1, INPUT_PULLUP);` が両方必要です。
フローチャートを描かせる場合、解答例は以下を参考にしてください。