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

不思議アイテムⅠ-2③

（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　2024年　月　日

第6回授業日　2024年　月　日

名前

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3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

不思議アイテムI-2③

第5回

ウルトラソニックロボット

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。

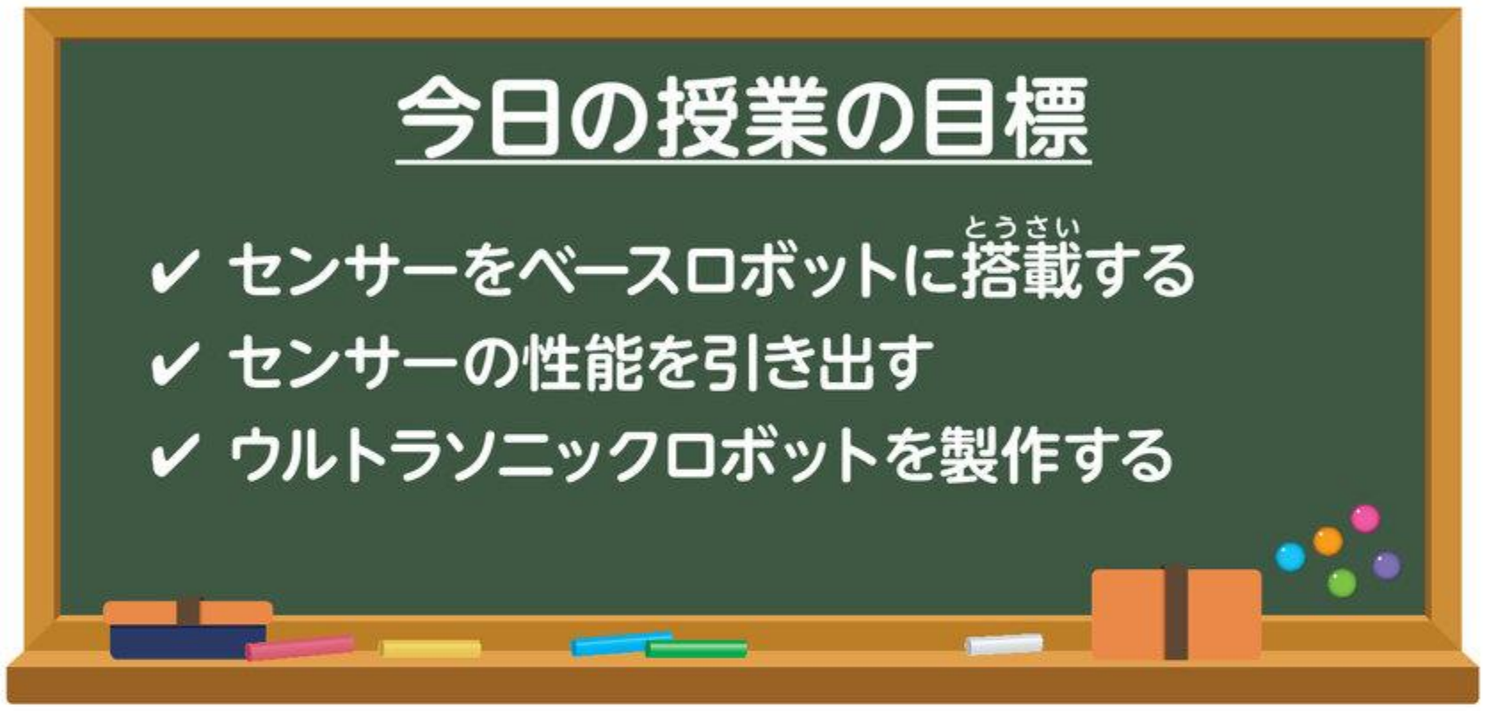
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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．ウルトラソニックロボット（目安 20 分）

0.0. 「ウルトラソニックロボット」でやること



今回の授業では、いよいよベースロボットにセンサーを組み込みます。そしてセンサーの情報を「感じて」、それをうけて「考えて」、「動く」ロボットをつくります。まずは、ベースロボットに超音波距離センサーを組み込みます。その後、センサーを使ううえで大切なことを学び、実際にサンプルプログラムを改造しながら、センサーの性能をさらに引き出していきます。最後に、ロボットの前にかざした手を追跡する「ウルトラソニックロボット」をつくりましょう！　センサーの特徴をいかすことがカギとなります。

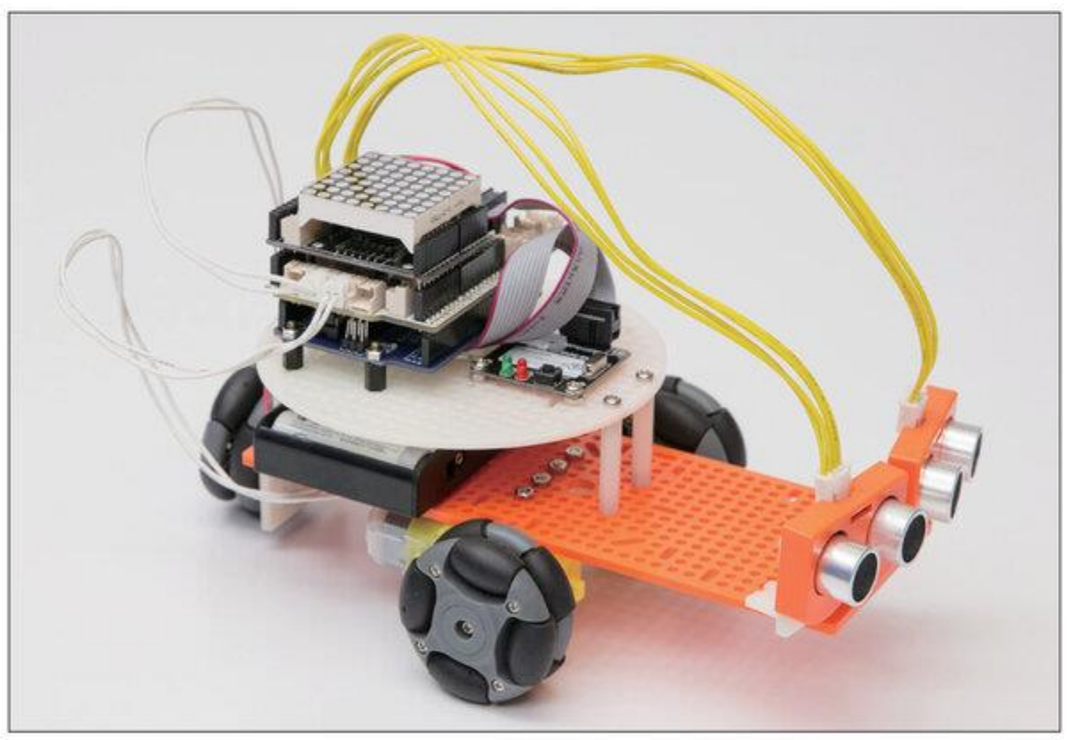
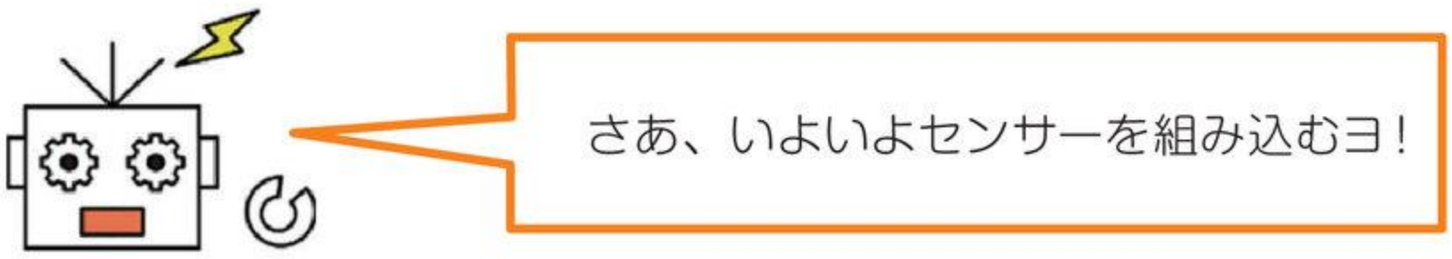


図 0-0　ウルトラソニックロボット



0.1. 必要なもの

前回使ったベースロボットと、以下のパーツを準備しておきましょう。
なお、前回使ったカラーセンサーとスピーカーはベースロボットから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	M2.6L8 タッピングネジ (B) 4
			
センサー L 字ステイ 1	マトリクス LED シールド 1	マトリクス LED 1	超音波距離センサー 2
			
センサーカバー 2	センサーケーブル 2	M3L8 ネジ 4	M3 ナット 4
			

図 0-1　必要なもの

0.2. 組み立てと配線

＜組み立て手順 ①＞

ベースロボットのロボプロシールドに、マトリクス LED シールドとマトリクス LED を取り付けます。第 1 回でやったようにピンを曲げないように慎重（しんちょう）にさし込みましょう。また、上下の向きもまちがえないように注意しましょう。

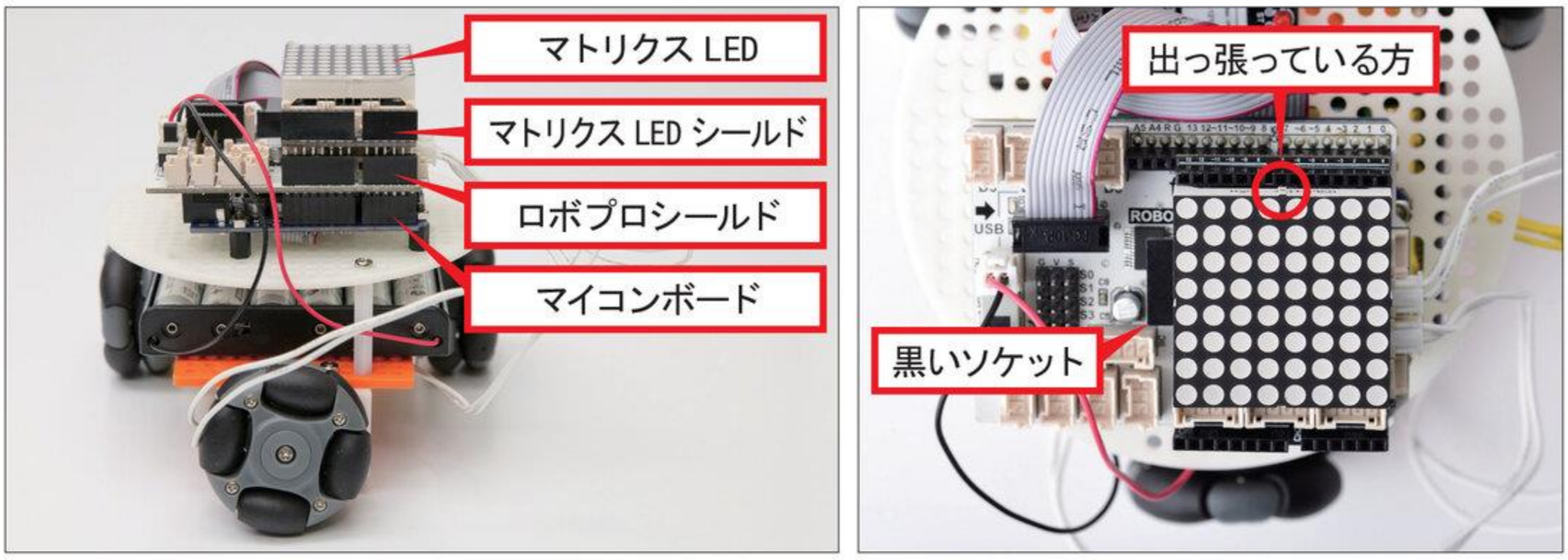


図 0-2　マトリクス LED シールドとマトリクス LED の取り付け

＜組み立て手順 ②＞

超音波距離（ちょうおんぱきょり）センサーをセンサーカバーとセンサー L 字ステイではさみ込み、M2.6L8 タッピングネジ（B）で固定します。

これを 2 セットつくりましょう

図 0-3　超音波距離（ちょうおんぱきょり）センサーの組み立て

＜組み立て手順 ③＞

ベースロボットの前方部に、超音波距離センサー（× 2）を取り付けます。センサーは**図 0-4** を参考に、M3L8 ネジ（× 4）と M3 ナット（× 4）で固定します。細かな向きは後で調整するので、ここでは取り付けられていれば大丈夫です。

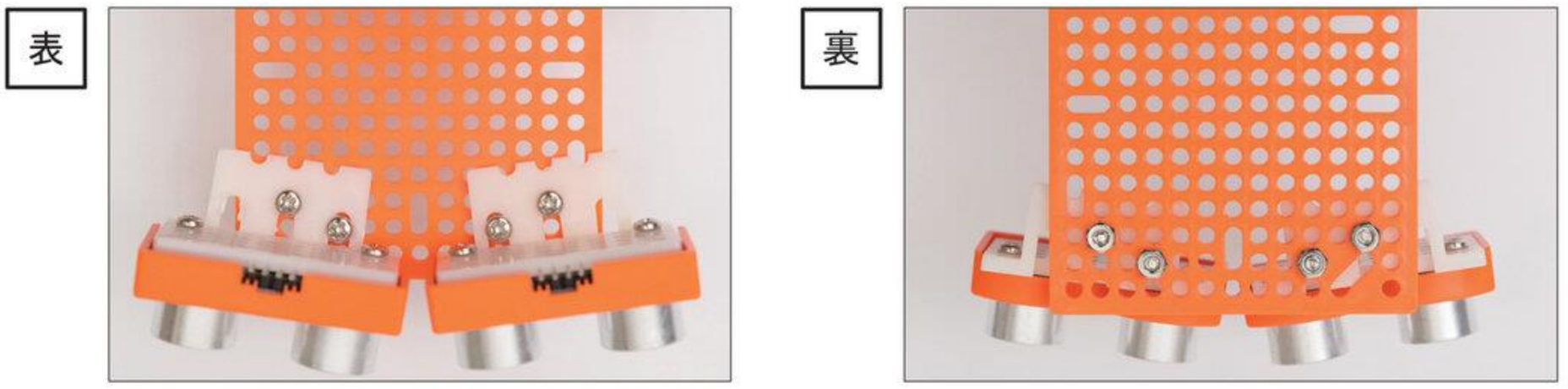


図 0-4　超音波距離センサーの取り付け位置

＜組み立て手順 ④＞

固定した超音波距離センサーをマトリクス LED シールドのコネクターにセンサーケーブル(× 2)で接続します。前進方向から見て右側のセンサーを US1 に、左側のセンサーを US2 につなげます。

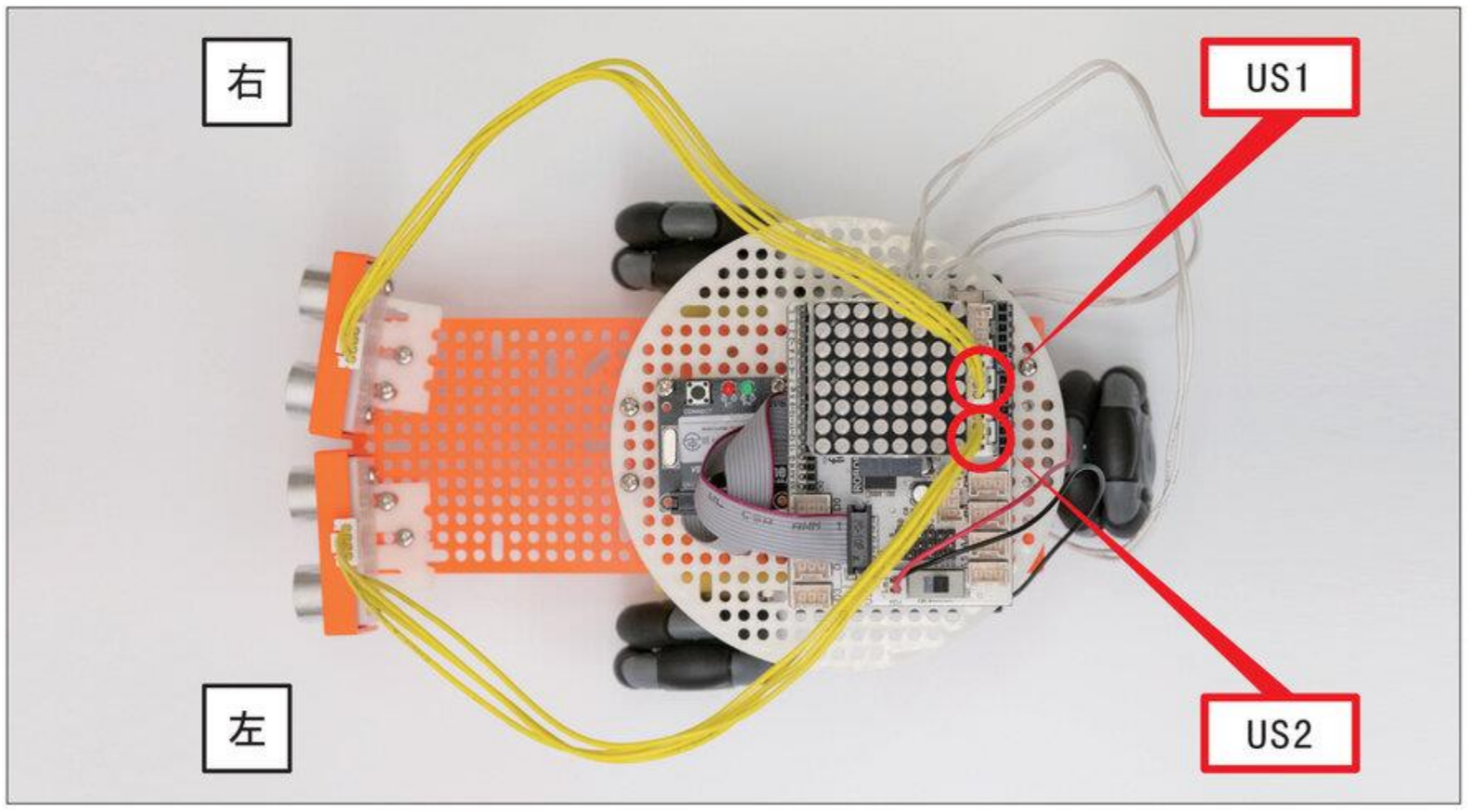


図 0-5　センサーケーブルの取り付け

これで、ロボットは完成です。

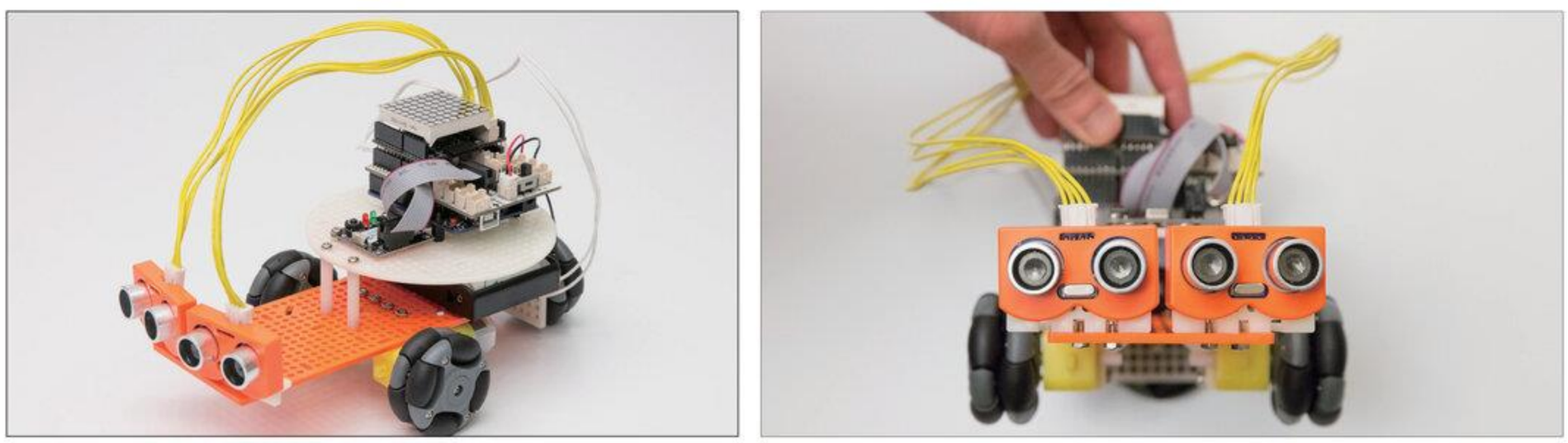


図 0-6　ウルトラソニックロボットの完成図

1．センサーについて（目安 20 分）

1.0.　センサーを知る

ここまで、いくつかのセンサーにふれてきましたが、世界には他にもさまざまな種類のセンサーがあります。今回は、あらかじめセンサーを選んでおいたわけですが、本来はみなさん自身でたくさんのセンサーの中から、それぞれの目的に合ったものを選ぶわけです。つまり、各センサーがどのような性能･性質を持っているかを知り、自分のロボットに合ったものを選べなくてはなりません。
そんなときにぜひ調べてほしいのが、説明書です。ここでは説明書の中のどのような情報に着目したらいいのかを学びながら、実際に授業で使う超音波距離センサーについても理解を深めましょう。

1）サイズ

たとえば今回の超音波距離センサーの説明書にも、**図 1-0** のような製品のサイズなどの情報がのっています。センサーを安定して取り付けるために、とても大事な情報です。

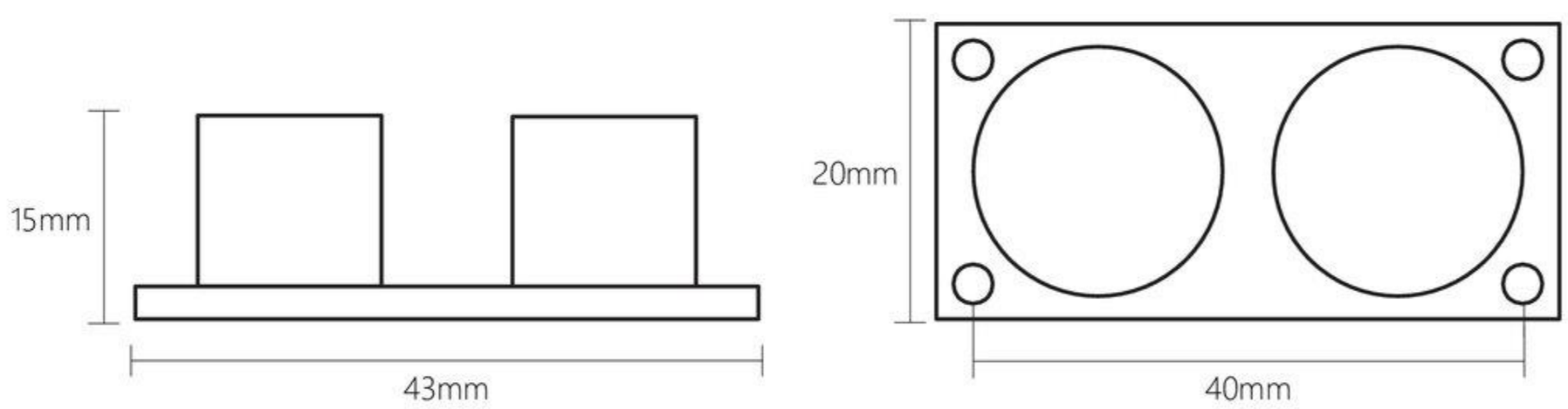


図 1-0　製品のサイズイメージ

2）性能

超音波距離センサーの性能で知っておいてほしいことは、「測定可能な角度範囲」と「測定可能な距離」です。
製品説明書には英語でさまざまなことが書かれていますが、今回知りたい情報は以下のようにまとめられていました。

POINT

●測定可能な角度範囲（Effectual Angle）：15 度以下
●測定可能な距離（Ranging Distance）：2cm - 400 cm

メーカーが、センサーの性能を教えてくれているわけです。
これらの情報をもとにして、センサーの性能を上手く利用していきましょう。

1.1. センサーの性能の確認と調整

１）右側のセンサーの表示

では実際に、センサーの動作確認をします。以下のプログラムを実行し、右側のセンサーの前に手をかざして、近づけたり遠ざけたりしてみましょう。なお、USB ケーブルからの給電では遠い距離(きょり)が計測できない可能性があるので、ロボプロシールドのスイッチを ON にしましょう。

プログラムの書き込み

RoboticsProfessorCourse1 ＞ MagicItemB5 ＞ step0_USStest

右側のセンサーが読み取った距離(きょり)の値が 00 ～ 99 でマトリクス LED に表示されます。

今回や次回で扱(あつか)うロボットは超音波距離(ちょうおんぱきょり)センサーを 2 つ使います。 US1 のセンサーがはかった距離(きょり)と、 US2 のセンサーがはかった距離(きょり)は基本的にばらばらなので、2 種類の距離(きょり)を入れておくために 2 種類の変数を用意する必要があります。
今回は US1 センサーの値を入れるのに dist1 、 US2 センサーの値を入れるのに dist2 という変数を使いましょう。

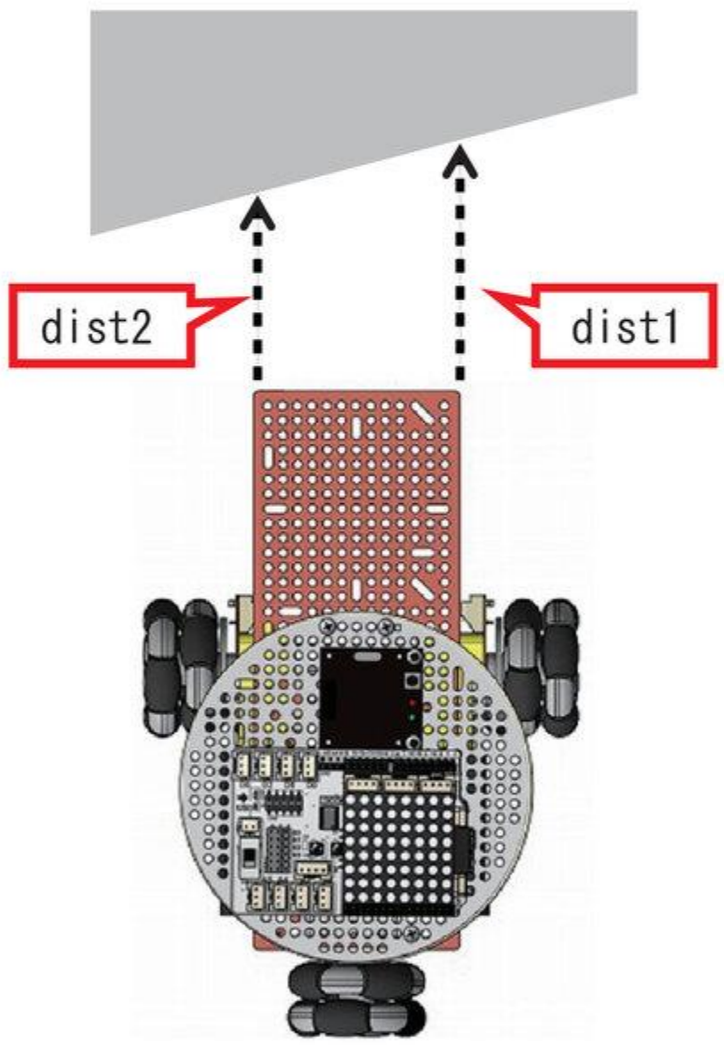


図 1-1　左右のセンサーに別々の変数を使う

プログラム「step0_USStest」でも、使っているのは US1 のセンサーだけですが、変数の宣言(せんげん)は dist1 と dist2 どちらも書いてあります。

プログラム「step0_USStest」より抜粋(ばっすい)

```
int dist1;                    // 変数intをdist1という名前で使う
int dist2;                    // 変数intをdist2という名前で使う
```

命　　令「int」
実行結果：指定した文字列を変数として使う
使 い 方：int dist1;　// dist1 という文字列を、変数として使う

これで、dist1 や dist2 という文字列に、数値を入れることができるようになりました。
あとは、センサーで読み取った値を、変数に入れるよう命令するだけです。

プログラム「step0_USStest」より抜粋（ばっすい）

```
dist1 = ussRead(US1);          // US1の距離データを変数dist1に入れる
//dist2= ussRead(US2);         // US2の距離データを変数dist2に入れる
```

イコール（等号）は算数・数学でもおなじみの記号ですね。プログラミングの世界では「 = の右側（右辺）の値を、左側（左辺）の変数に入れる」といった場合に使います。
ussRead(US1) は、「センサーが読み取った値」を呼び出すための呪文（じゅもん）のようなものです。これで US1 センサーがはかった距離（きょり）の値を、左辺にある変数 dist1 に入れているというわけです。
2 行目の US2 センサーに対する命令には、// という記号がついています。この記号は、うしろに書かれている文が、たとえ正しいプログラムの形になっていたとしてもプログラムとして実行されないようにするものです。
「とりあえずプログラムの形を作っておくけど、まだ実行したくない」という部分や、「この行がどんな意味なのか文章でメモしておこう」という部分に活用されます（コメントアウトと呼ばれる機能です）。

やってみよう！

プログラム「step0_USStest」は、右側のセンサー US1 の値しか確認できないね！
プログラムを書きかえて、左側のセンサー US2 の値が確認できるようにしてみよう！

ヒント
コメントアウトを活用すれば、書きかえる場所をぐっと減らせるね！

講 //dist2= ussRead(US2); と、その数行下にある //dispNum=dist2; から // を消去し、それぞれ 1 つ上の行に // を追加します。

2）左右のセンサーの表示

今度は、左右のセンサーを同時にチェックしましょう。以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemB5 > step0_USStest2

実行結果：10 の位に右側のセンサーの結果が、１の位に左側のセンサーの結果が表示される。
それぞれの距離値の 10 分の 1 の値です。10 センチごとに 1 ずつ増え、それぞれ最大で 9 まで表示されます。例えば、距離が 10 センチであれば「1」、20 センチであれば「2」と表示されます。

図 1-2 のように、2 個のセンサーの前に障害物をかざして、遠ざけたり近づけたりしてみましょう。左右のセンサーが、おおよそ同じ値を表示すれば、センサーの動作に問題はありません。

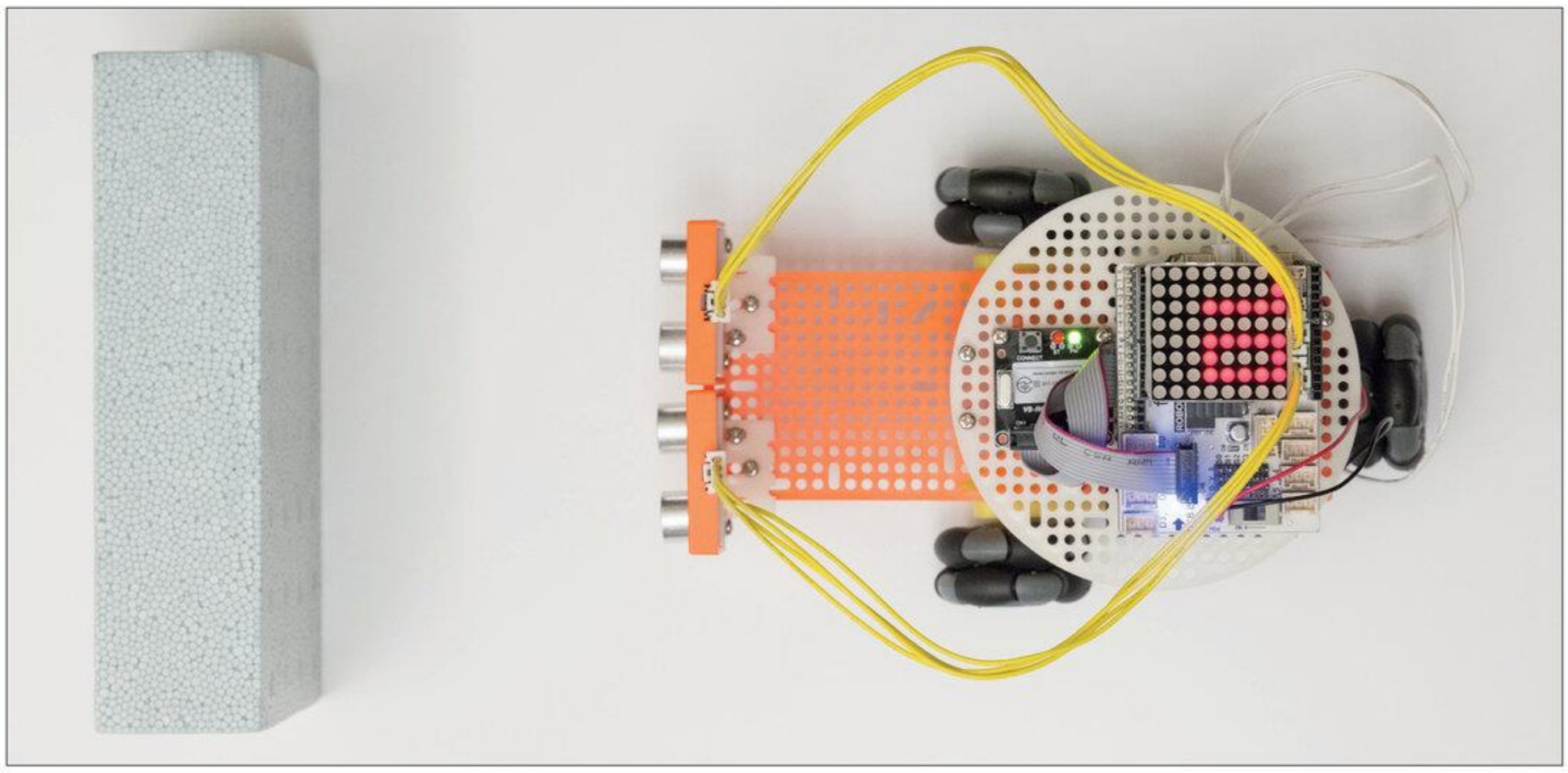


図 1-2　左右のセンサーの表示の確認

3）左右のセンサーの向きの調整

最後に、できるだけ広範囲を読み取れるように左右のセンサーの向きを調整してみましょう。まずは、センサーの読み取りの性能から有効範囲を考えてみます。センサーをまっすぐに設置した場合は、**図 1-3** の扇形の部分が読み取れる範囲となります。しかし、この状態では読み取る範囲が重なり、無駄が多くなります。

図 1-3　センサーをまっすぐ設置したときの読み取り可能な範囲

センサーが同じところを見ているのはもったいないですね？ この設置の向きを工夫して、**図 1-4** のように少し外側に広げてみると、わずかですが、読み取れる範囲が広がります。せっかくなので、広くモノを見えるロボットにしましょう。めいっぱい読み取りの範囲を広げてみてください。

図 1-4　センサーをかたむけて設置したときの読み取り可能な範囲

やってみよう！

もう一度、プログラム「step0_USStest2」を実行して、読み取りの範囲が広がるようにセンサーの向きを変えてみよう。ネジをつけ直してもいいよ。

2．追跡ロボット（目安 60 分）

続いてはロボットに動きを加えて、追跡ロボットを作りましょう。今回は、前にかざされた手を追跡するロボットにしてみます。

やってみよう！

下の図のような状況のとき、ロボットが「手を追跡する（ぶつからないように追いかける）」ためにはどのような動作が必要かな？

反時計回りに少し回転し、手が正面に来たら前進する。手が近すぎるときは、停止したり後退したりする。

実際に手を追跡するためには、次のような動作が必要ですね。

POINT

① ロボットの正面に手がくるまで、左右に回転する。
② 手が正面にあるときは、手との距離に応じて前進したり後退したりする。

つまり、「左右」と「前後」の 2 種類の動作をプログラムすることになります。
それぞれちがった形のプログラムになるので、別々につくって最後に合体させるイメージで作業していきましょう。

2.0. 前後に追跡する動きをつくる

手までの距離を一定に保って、前後に動くロボットにしてみましょう。まずは、以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemB5 > step1

実行結果：**図 2-0** のような画面が表示される。かざす手の距離に応じて**図 2-1** ～ **2-3** のように表示が変わり、**図 2-1** の表示になったときだけ、ロボットは動作（前進）する。

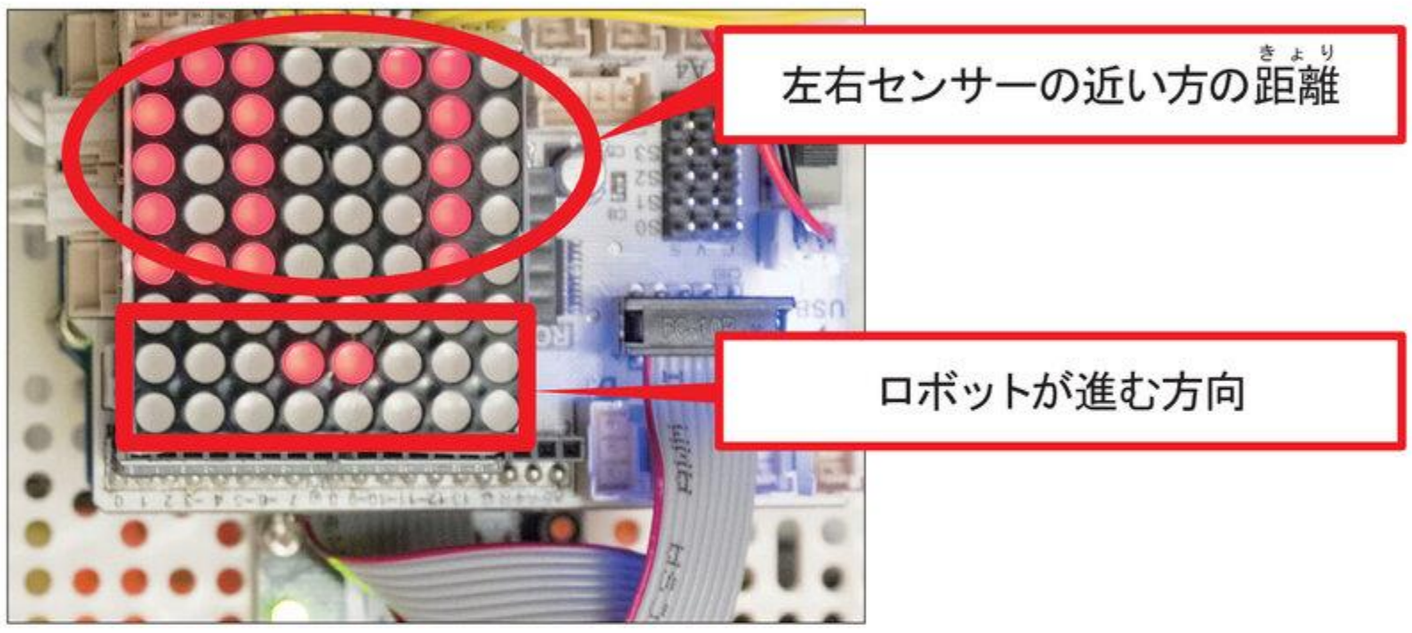


図 2-0　プログラム「step1」を実行したときの表示

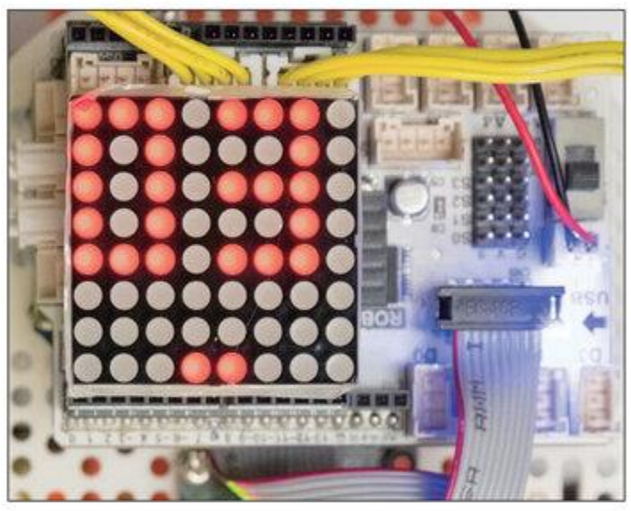


図 2-1　前進表示

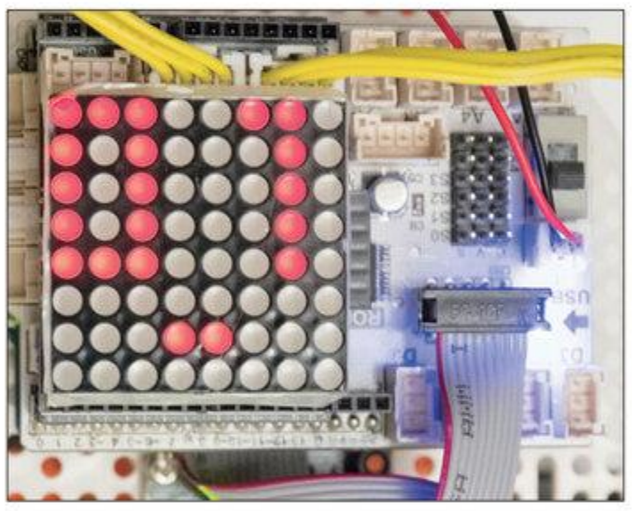


図 2-2　停止表示

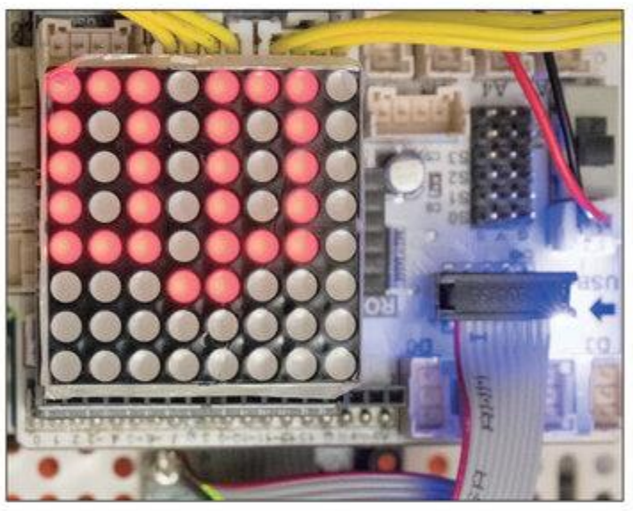


図 2-3　後退表示

ただし、このプログラムでは**図 2-1** の表示以外では、「前進」も「後退」もしないため不完全です。そのため、このあと動作を追加し、プログラムを完成させましょう。

講　マトリクスＬＥＤの表示位置で、進む方向を表しています。
図では、写真の上がロボットの後ろ側、下が前側となっています。

このプログラムでは「手がロボットの近くにあるか、遠くにあるか」を判定させるだけでよいので、2つのセンサーを同時にチェックする必要がありません。そのため、US1 と US2 のうちより値が小さい、つまり手が近くにある方のセンサーだけを使うようにします。

プログラム「step1」より抜粋

```
if(dist1 < dist2){
    minDist = dist1;
}
else{
    minDist = dist2;
}
```

新たに minDist という変数を使っています。もし dist1 < dist2 、つまり US1 センサーの方が値が小さければ minDist に US1 の値を、そうでないときには minDist に US2 の値をそのまま入れています。

あとは、手との距離 minDist の値に応じて、前進や後退、あるいは停止をするようプログラムを書いてあげれば完成です。

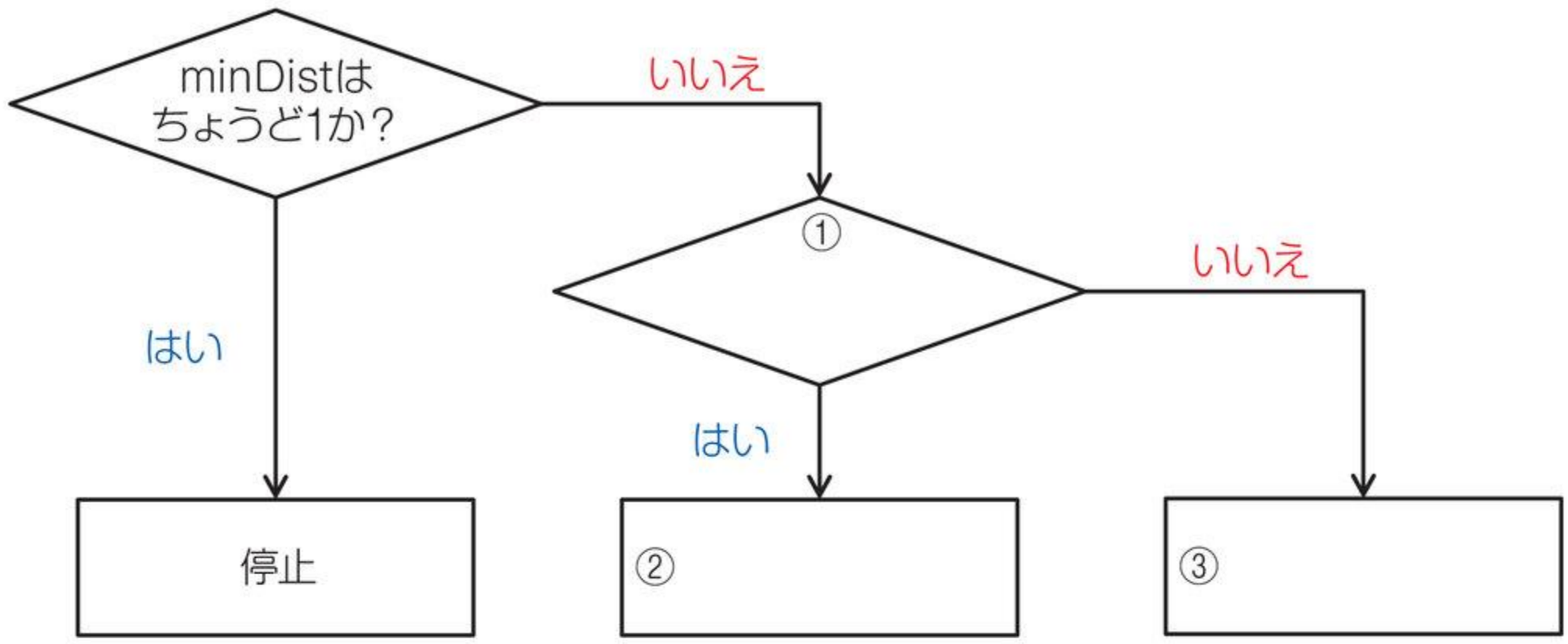
プログラム「step1」より抜粋

```
if(minDist == 1){        // minDistが1だったら停止
    mc1.rotate(0);
    mc2.rotate(0);
}
```

このプログラムでは、超音波距離センサーがはかった距離を 10 でわっています（マトリクス LED に表示できるけた数におさえるため）。
つまり if(minDist == 1) という if 文は、「もし手との距離が 10cm くらいなら」という意味になりますね。
このとき、ロボットは停止するようにプログラムされています。

やってみよう！

手とぶつからないように追跡するロボットにするためには、minDist がどのような値のとき、どのような動作をさせる必要があるかな？　下の空らんに、分岐の条件と動作の内容を書きこんでみよう！



ステップアップ

上の「やってみよう！」で完成させた図をもとに、プログラム「step1」を書きかえてみよう！

講

解答例は以下の通りです。
やってみよう：① minDist は 1 より大きいか？　② 前進　③ 後退
「1 以上」と書くと minDist == 1 の範囲と被るので、「より大きい」というのをキーワードにしてください。
また、①を「minDist は 1 未満か?」としても構いません。その場合は②と③が入れ替わります。

ステップアップの解答例は以下のプログラムです。
RoboticsProfessorCourse1 > MagicItemB5 > step1_answer
上記のプログラムは 4 分岐になっていますが、フローチャート通り 3 分岐にまとめてしまっても問題ありません。

2.1. 回転して追跡する動きをつくる

続いては手を追跡して、回転するロボットにしてみましょう。こちらも、まず以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemB5 > step2

実行結果：**図 2-4** のような画面が表示される。かざす手の距離に応じて**図 2-5** ～ **2-7** のように表示が変わり、**図 2-7** の表示になったとき（左のセンサーの前に手をかざしたとき）だけ、ロボットは左回転する。

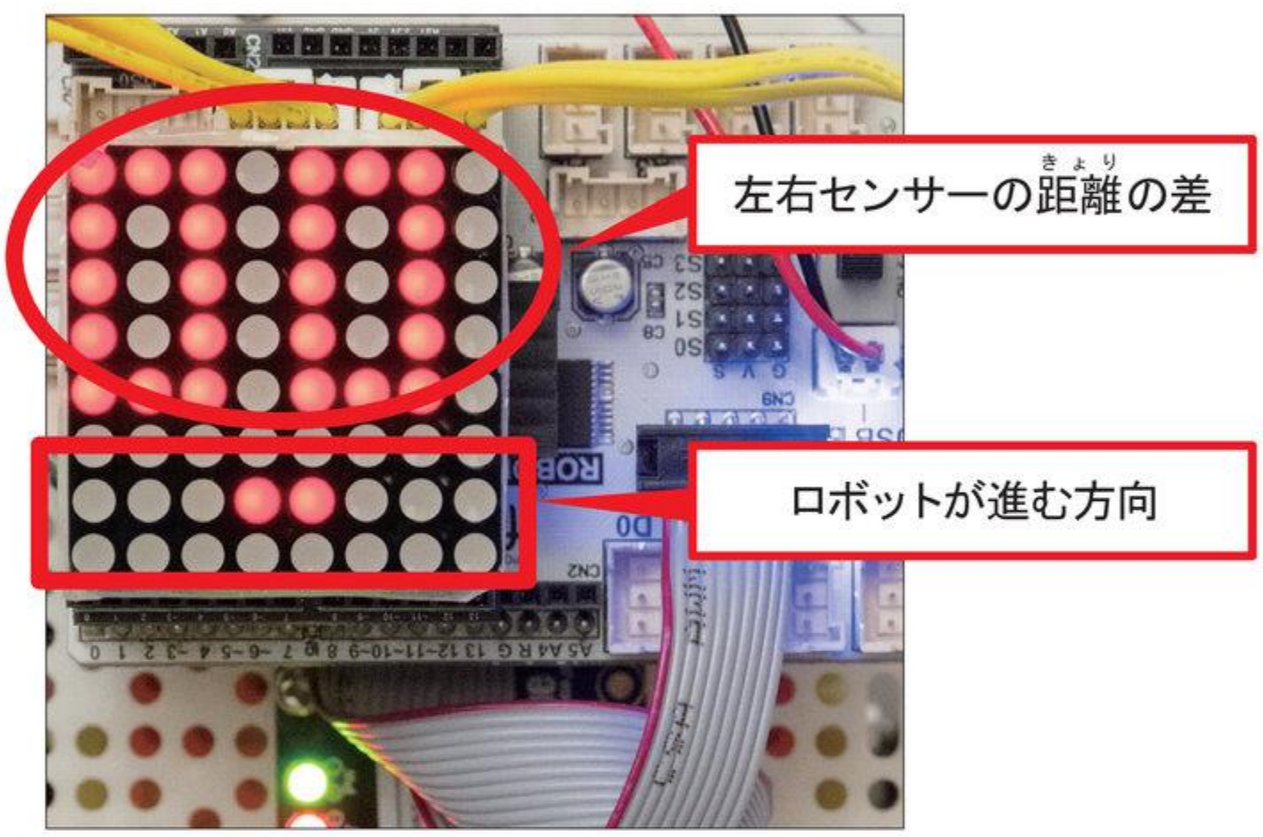


図 2-4　プログラム「step2」を実行したときの表示

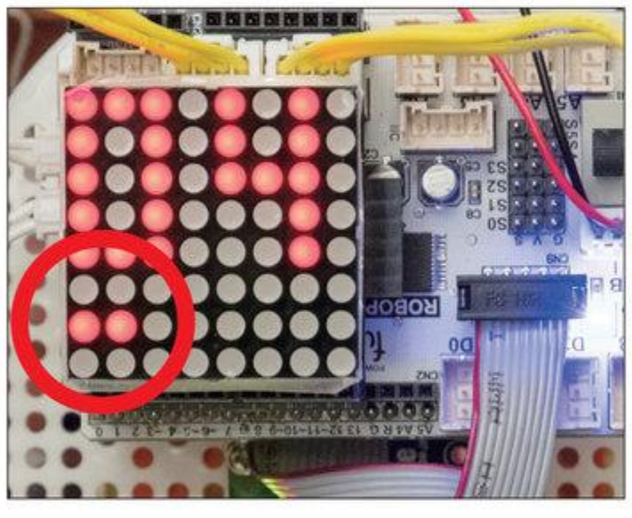


図 2-5　右回転表示

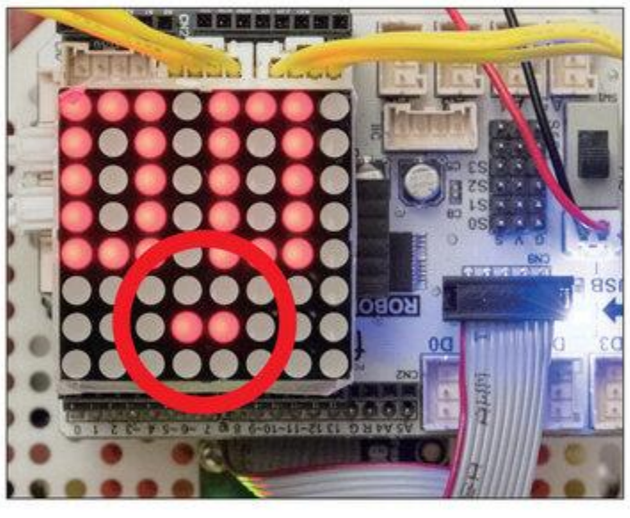


図 2-6　停止表示

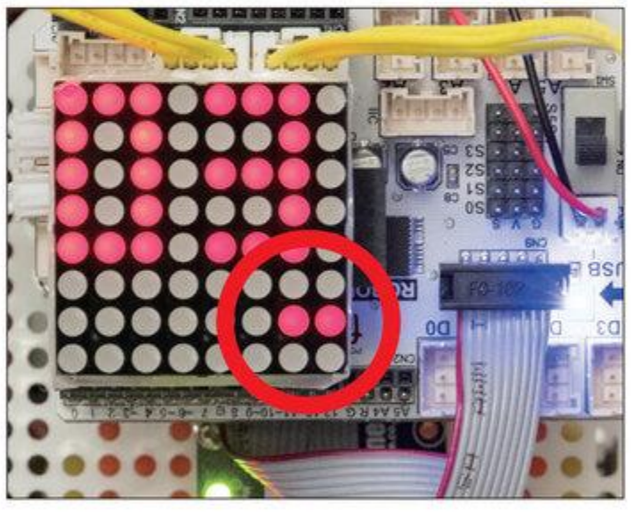


図 2-7　左回転表示

ここでは「手が正面にくるよう左右に回転する」という動作をめざしています。そのため「手が正面にきたら止まる」という処理が必要ですね。

プログラム「step2」より抜粋（ばっすい）

```
sub    = dist1 - dist2;          // US1とUS2の距離の差を変数subに入れる
if(abs(sub) <= 2){
    // 差が小さかったら停止
    mc1.rotate(0);
    mc2.rotate(0);
}
```

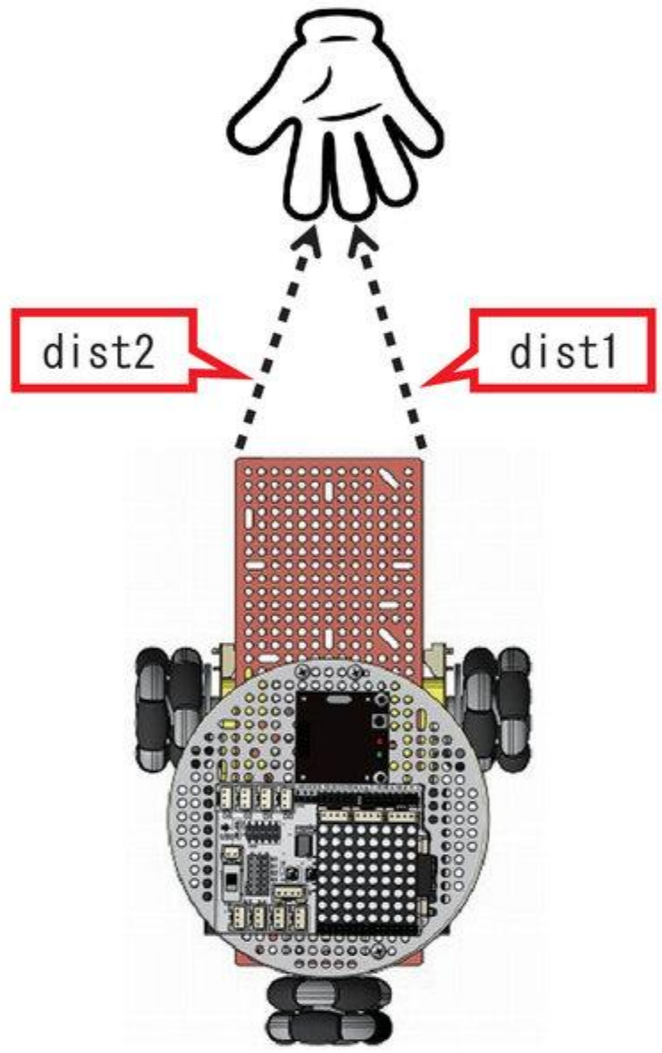


図 2-8　手が正面にあるとき

ここには、手が正面にあるときは停止するという命令が書かれています。
手が正面にあるということは、dist1 と dist2 はほぼ同じ値になるため、ひき算で差を求めるとかなり小さな値になるはずです。よって、dist1 と dist2 との差を変数 sub に入れ、これが 2 以下なら停止するようにしています。

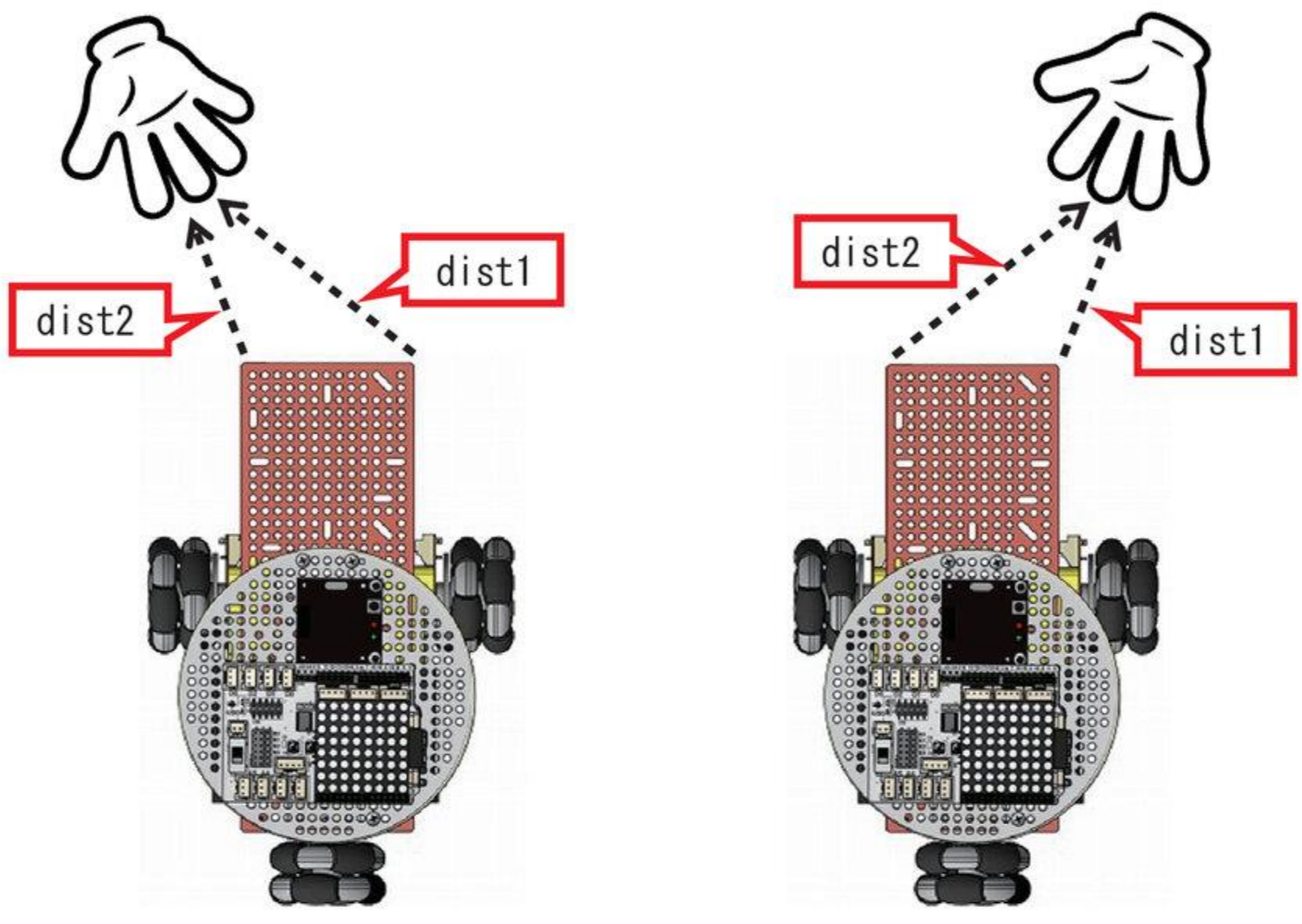
この条件に当てはまらない、つまり手が正面に無いときは、左または右に回転する必要がありますね。
まだ左右回転を使い分けるプログラムになっていないため、自分で完成させてみましょう。

ステップアップ

プログラム「step2」を書きかえ、ロボットがきちんと手の方に向き直れるようにしてみよう！

ヒント

手がロボットの左や右にあるとき、それぞれ dist1 と dist2 はどのような関係になるかな？　次の図をヒントにしてみよう！



講 解答プログラムは以下となります。
RoboticsProfessorCourse1 ＞ MagicItemB5 ＞ step2_answer
解答例は巻末に記載します。

チャレンジ課題

前後移動・左右回転の動作プログラムを合体させて、手を追跡（ついせき）するロボットのプログラムを完成させよう！
どこにどんな順番で if 文を入れていけばいいかな？

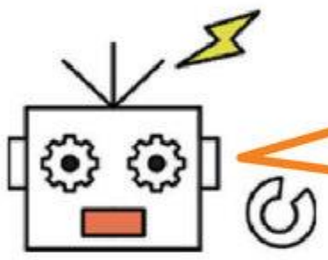
ヒント

わからなくなったらいったんプログラムから目をはなして、分岐（ぶんき）条件や動作を文章にまとめたり、図にしたりして整理してみよう！

講 解答プログラムは以下となります。
RoboticsProfessorCourse1 ＞ MagicItemB5 ＞ step3
解答例は巻末に記載します。

3. まとめ（目安５分）

第 5 回の授業も終了です。
第 1 回～第 4 回では状況に応じて動きを分ける if や、モーターを回転させる rotate などの命令をばらばらに学んできましたが、今回はそれらを組み合わせたプログラムをつくりました。その結果、だいぶ「ロボット」らしい動きになってきましたね。
次回は超音波距離(ちょうおんぱきょり)センサーをさらに活用して、さまざまな場面に対応して動くロボットをつくっていきます。
やはり使うのは今回と同じ命令文たちなので、次回にそなえてじっくり復習しておきましょう！



センサーをつけると、ロボットが一気にカシコクなったネ！
ボクにもセンサーをつけてほしいな、100 個くらい！！

講

○　以下の授業の目標を再確認します。
・センサーをベースロボットに搭載する
・センサーの性能を引き出す
・ウルトラソニックロボットを製作する

○　次回テーマは「カシコイセンサーロボット」であることを告知します。

≪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。

ラジオペンチ 1	ドライバー 1	USB ケーブル 1	センサー L 字ステイ 1
			
M3 ナット 2	**M3L8 ネジ 2**	**ユニバーサルバー 1**	
			

図 3-0　次回必要なもの

P.13　ステップアップ　解答例（step1_answer）

```
if(minDist == 1){                    // minDistが1だったら停止
    mc1.rotate(0);
    mc2.rotate(0);
}
else if(minDist > 1){                // minDistが1より大きかったら前進
    mc1.rotate(-50);
    mc2.rotate(50);
}
else if(minDist < 1){                // minDistが1より小さかったら後退
    mc1.rotate(50);
    mc2.rotate(-50);
}
else{                                //そうでなければ停止
    mc1.rotate(0);
    mc2.rotate(0);
}
```

P.16　ステップアップ　解答例（step2_answer）

```
else if(dist1 > dist2){
    // 左のセンサーが近いので左に回転
    mc1.rotate(-50);
    mc2.rotate(-50);
}        // ------------追加分
else if(dist1 < dist2){
    // 右のセンサーが近いので右に回転
    mc1.rotate(50);
    mc2.rotate(50);
}
```

P.16　チャレンジ課題　解答例（step3）

```
// ライブラリを取り込む
#include <Sprite.h>
#include <Matrix.h>
#include <RPlib.h>

#define  NEUTRAL  0
#define  LEFT   1
#define  RIGHT  2
#define  FORWARD  3
#define  BACKWARD  4

int gCmdTmp     = NEUTRAL;

Matrix myMatrix = Matrix(11, 13, 1);  // マトリクスLEDを使うときのオマジナイ
RPmotor mc1(MC1);
RPmotor mc2(MC2);

void setup(){
    myMatrix.setBrightness(3);        // マトリクスLEDの明るさを設定(0−8)
    myMatrix.clear();
}

void loop(){
    static int DIV = 10;              // 一けたに収まりそうな数にする分母
    int cmd        = NEUTRAL;
    int sub;                          // 変数intをsubという名前で使う
    int minDist;                      // 変数intをminDistという名前で使う
    int dist1;                        // 変数intをdist1という名前で使う
    int dist2;                        // 変数intをdist2という名前で使う
    int dispNum = 0;                  // 変数intをdispNumという名前で使う（初期値は0）

    myMatrix.clear();
    dist1 = ussRead(US1);             // US1の距離データを変数dist1に入れる
    dist2 = ussRead(US2);             // US2の距離データを変数dist2に入れる

    if(dist1 > 99) dist1 = 99;
    if(dist2 > 99) dist2 = 99;

    dist1 /= DIV;
    dist2 /= DIV;
```

```
// 小さいほうの値をminDistにいれる
if(dist1 < dist2){
    minDist = dist1;
}
else{
    minDist = dist2;
}
// 距離の差
sub = dist1 - dist2;

if(dist1 > 6 && dist2 > 6){
    // どっちも距離が遠すぎるときは停止
    mc1.rotate(0);
    mc2.rotate(0);
}
else{
    if(abs(sub) <= 2){
        if(minDist == 1){
            mc1.rotate(0);
            mc2.rotate(0);
        }
        else if(minDist > 1){
            mc1.rotate(-50);
            mc2.rotate(50);
        }
        else if(minDist < 1){
            mc1.rotate(50);
            mc2.rotate(-50);
        }
    }
    else if(dist1 > dist2){
        // 左のセンサーが近いので左に回転
        mc1.rotate(-50);
        mc2.rotate(-50);
    }
    else if(dist1 < dist2){
        // 右のセンサーが近いので右に回転
        mc1.rotate(50);
        mc2.rotate(50);
    }
}

(中略)

}
```