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

センサーロボット②

第4回

迷路から脱出しよう

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。

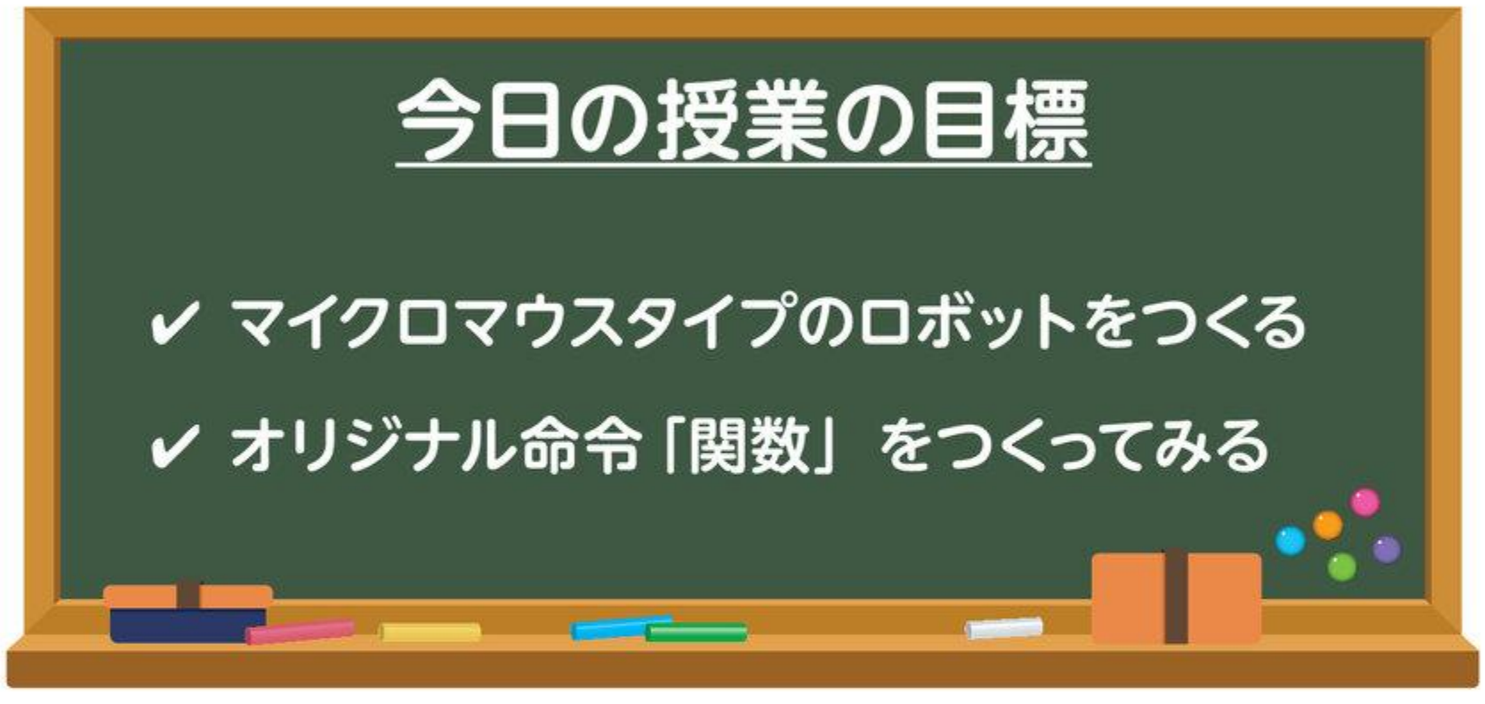
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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）

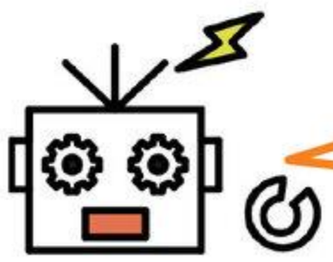
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0．迷路から脱出しよう（目安 10 分）

0.0. 「迷路から脱出しよう」でやること



今回は、迷路をぬけられるロボットをつくってみましょう。
迷路をぬける方法って知っていますか？ どうやって行うのでしょうか？ 野生の勘で進みますか？ いやいや、もっと論理的な方法があります。
今回は、そんなアルゴリズムをどうやってロボットにインストール（組み込み）していったら良いかという部分まで学んでいきましょう。学習型などの例外を除いて、基本的にはロボットはつくった（プログラムした）人間以上にはカシコくなれません。だからこそどういったアルゴリズムを考え出していくかが腕の見せ所なのです。前回少し使ってみたオリジナル関数についても、もう少し理解を深めていきますよ。
そして最終的には「マイクロマウス」という大会に出るようなロボットに改造しましょう！
「マイクロマウス」というのはとても歴史のあるロボットコンテストで、ロボットが自動で迷路を走りぬける時間を競い合うコンテストです。初めて見る人は、瞬く間に迷路を駆けぬけるロボットの速さに驚愕することでしょう。興味のある人は、「マイクロマウス」のキーワードで動画検索すると競技の映像がたくさん見られますよ。
今回はそんなロボットを目指してつくり上げていきましょう！



どんな迷路でも脱出できるロボットをつくってみたいナ！

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。

ラジオペンチ 1	ドライバー 1	USB ケーブル 1	リボンケーブル 1
			
コントローラー 1	**無線受信モジュール 1**		
			

図 0-0　必要なもの

0.2.　組みかえとセンサーの配線

前回のものから若干変更する必要があるので、以下の写真や図などを手がかりに組みかえましょう。
前回から 30mm 角スペーサーとカラーセンサーの取り付け位置を変更しています。
なお、今回もネジ穴の位置などはテキストと完全に一致せずとも動作が可能であれば問題ありません。

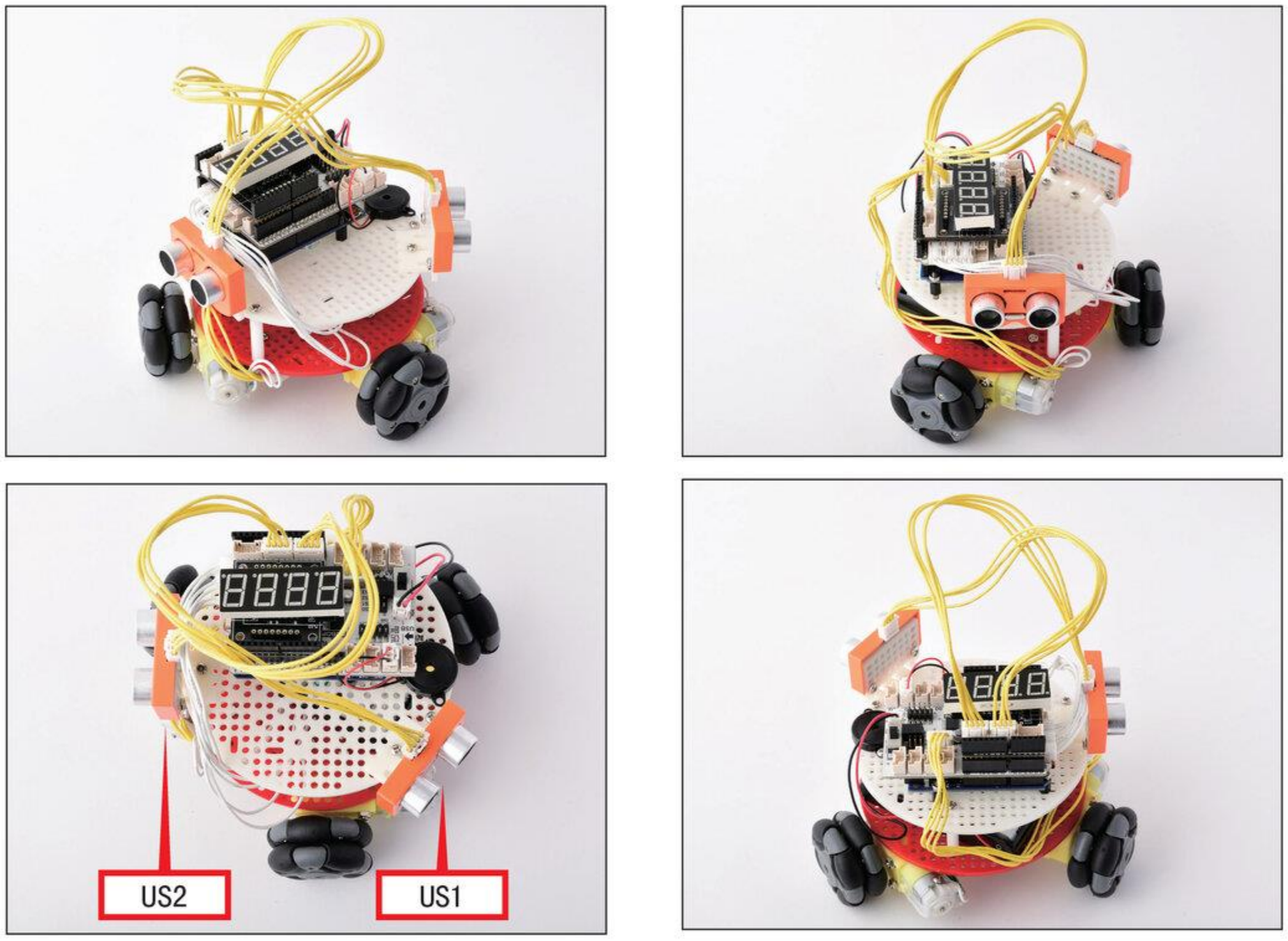


図 0-1　センサーロボットの外観

POINT

配線は以下のようにします。
・前方のモーターをロボプロシールドの MC3 に接続。
・右のモーターをロボプロシールドの MC1 に接続。
・左のモーターをロボプロシールドの MC2 に接続。
・超音波距離センサー（左前）をマトリクス LED シールドの US1 に接続。
・超音波距離センサー（右）をマトリクス LED シールドの US2 に接続。
・カラーセンサーをロボプロシールドの IIC に接続。
・スピーカーをロボプロシールドの D2 に接続。

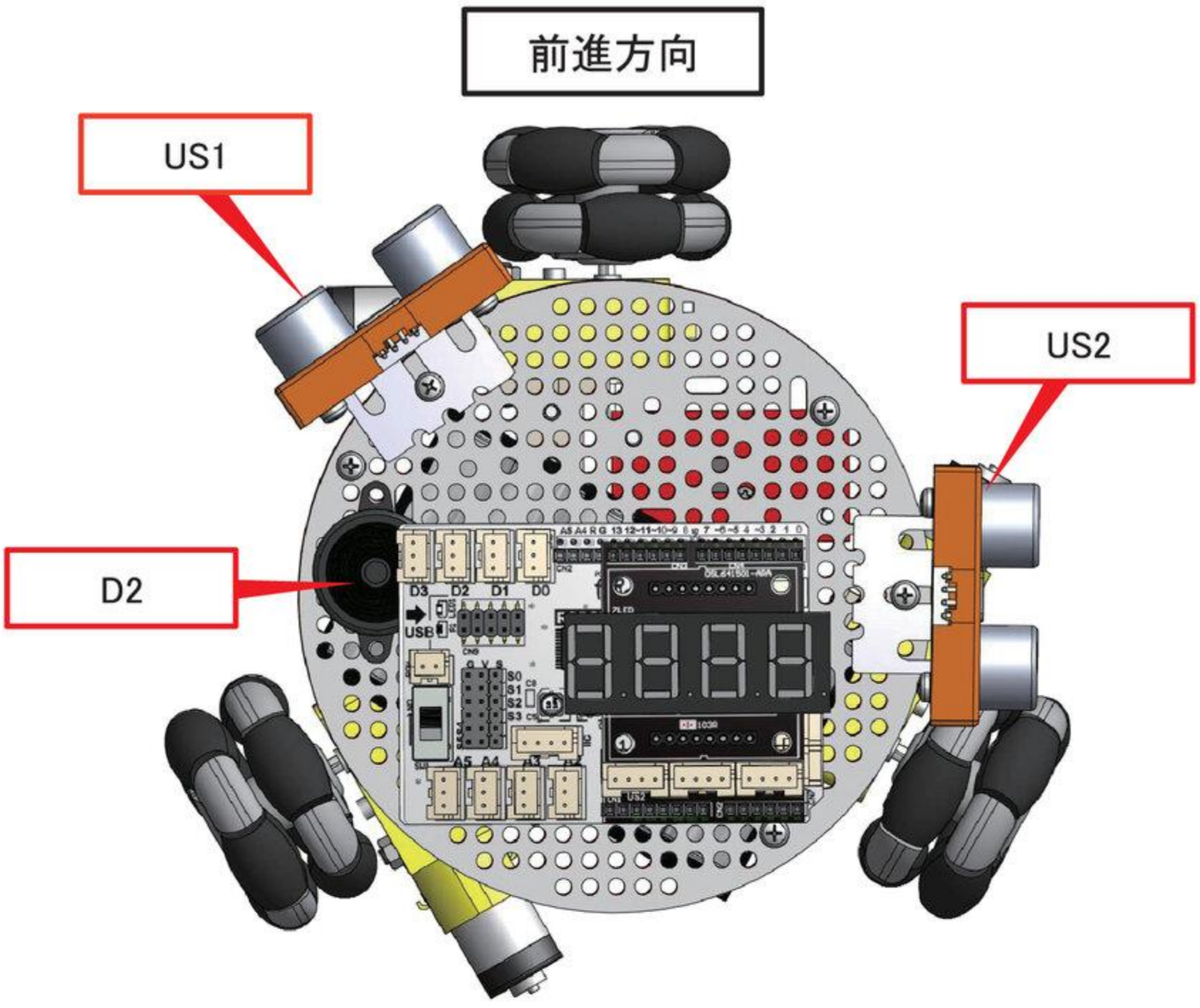


図 0-2　センサーロボットの配線指示図（表面）

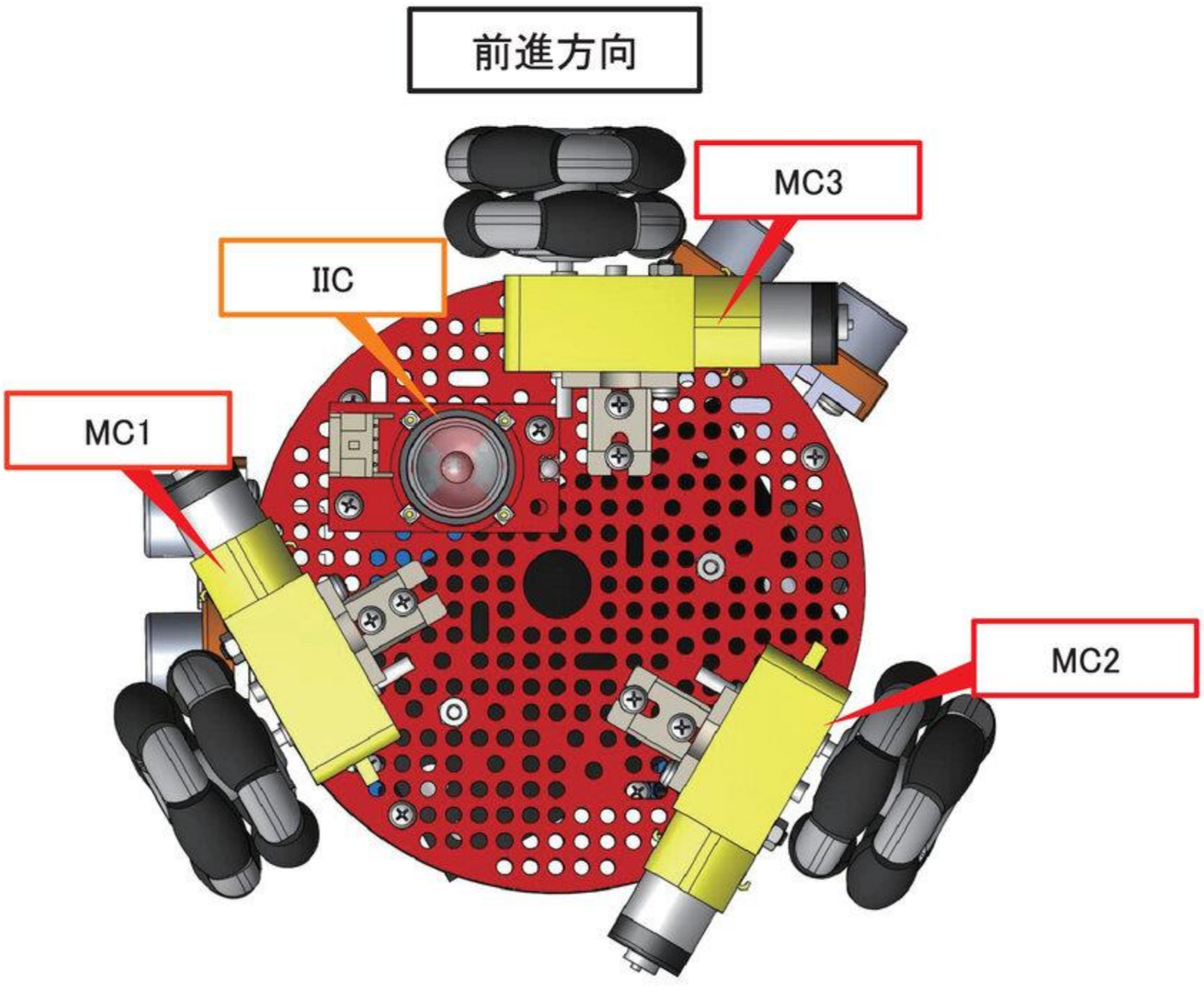


図 0-3　センサーロボットの配線指示図（裏面）

1．迷路ぬけロボット（目安 70 分）

1.0.　ロボットの調整

迷路を正確に走行するために、各ホイールの速度を調整します。まず、無線受信モジュールとリボンケーブルをロボプロシールドに接続し、コントローラーとペアリングをしましょう。なお、無線受信モジュールはこのパートでしか使用しないため、必ずしもネジ等で円形ボードに固定する必要はありません。そして、ロボプロシールドの MC3 に接続されている**前方のモーターを一時的に MC0 に接続し直します。**さらに、以下のプログラムを用いて、ロボットの走行調整値をあらためて確認し、下の調整結果の欄(らん)に記入しておきましょう。

プログラムの書き込み

RoboticsProfessorCourse1 > OmniWheelRobot2 > Adjust

POINT

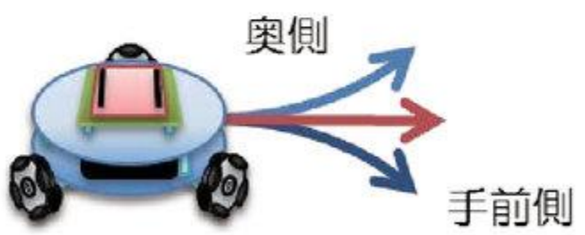
```
RPomniDirectomniBot(  f,  f,  f,  f );
                    ①   ②   ③   ④
```

調整手順１　△ボタン
調整手順２　○ボタン
調整手順３　×ボタン

調整手順１
△ボタンを押して，

調整手順２
○ボタンを押して，

調整手順３
×ボタンを押して，



調整の仕方は１年目の「オムニホイールロボット」の第２回のテキストも参照してください。
調整を終えたら、リボンケーブルと無線受信モジュールを外し、MC0 に接続した**モーターを MC3 に接続し直しておきましょう。**

1.1. ステップ① 迷路ぬけの方法を考える

まずは、「迷路ぬけロボット」の動作に必要な要素を考えてみます。そもそもセンサーやモーターなどを複数組み込んだ複合ロボットを完成させるには以下の点が重要になります。

POINT

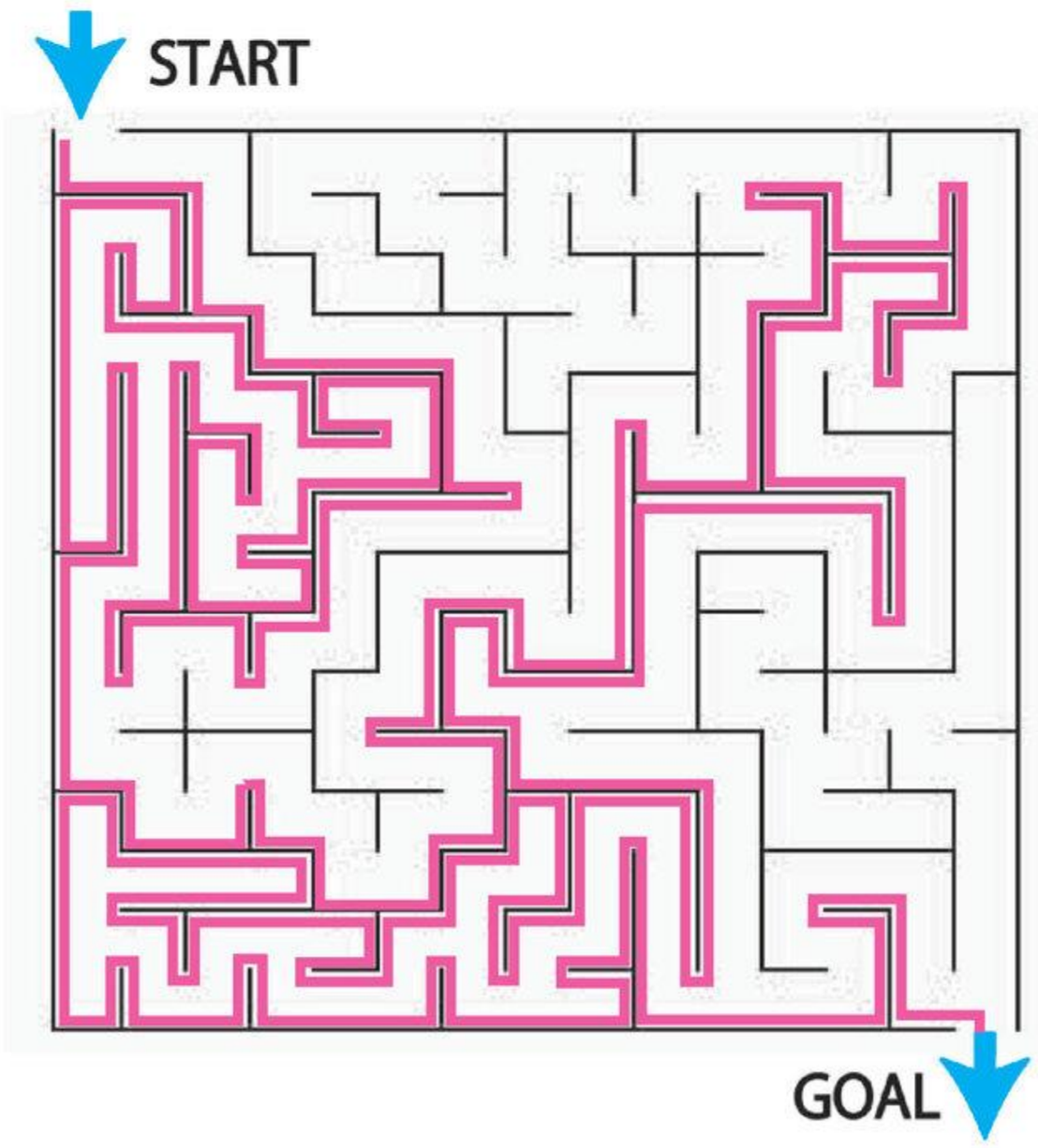
1. センサーやモーターなどが機能しているか。
2. ロボットの動きが調整できているか。
3. センサーやモーターが十分に動作するための、電池残量があるか。
   （電池残量が低下すると、精度が低下します。）
4. 動作手順が整理されているか。

1. ～ 3. までは、それぞれの動作確認で解消できるので、ここでは、4. について考えていきましょう。今回は迷路の進み方について考えます。

さて、皆さんは「迷路ぬけ」には攻略法があるのを知っていますか？ いわゆる「右手法」とよばれるものです。方法はいたって簡単で、右手を壁につけて迷路を進むだけです。迷路の中では、絶対に右手を壁から離してはいけません。不思議なことにこれをするだけで、基本的にはゴールまで辿り着くことができます（例外的に、ゴールが真ん中にあったりすると辿り着けず、ふり出しにもどってしまう場合もあります）。ちなみに、右手法は最短のコースを辿れるわけではありません。しかし、規則性があり、ロボットにプログラムを組み込むうえでは適した方法とも言えるわけです。それでは、右手法を使った迷路脱出を実際にやってみましょう。

やってみよう！

右手法を使って以下の迷路を脱出してみよう。実際に通る場所に線を引きながらゴールを目指してみてね。



1.2.　ステップ②　プログラムを考える

複雑な処理があるときは、いつも通り処理を細かく分割して考えていきましょう。フローチャートさえ完成すれば、プログラムはつくれたようなものです。まずは、超音波距離センサーとロボットが置かれている状況の関係を整理してみましょう。

1）センサーの反応パターンを整理する

超音波距離センサーの状態は**表 1-0** のように 4 つのパターンにわけることができます。センサーを増やしたり、特別なプログラム上の工夫をしたりしないかぎり、この 4 パターンのみです。

表 1-0　超音波距離センサーの反応パターン

	US1（前方）反応あり	US1（前方）反応なし
US2（右側）反応あり	パターン 1	パターン 2
US2（右側）反応なし	パターン 3	パターン 4

つまり、このロボットは左側や後ろに壁があるのかどうかはわかりませんが、「右手法」に従う動きになれば迷路をぬけることができるわけです。
左前向き、右向きにつけられた超音波距離センサーの US1 ・ US2 と、ロボットの置かれている状況を対応付ければ、迷路の攻略に大きく近づけそうです。

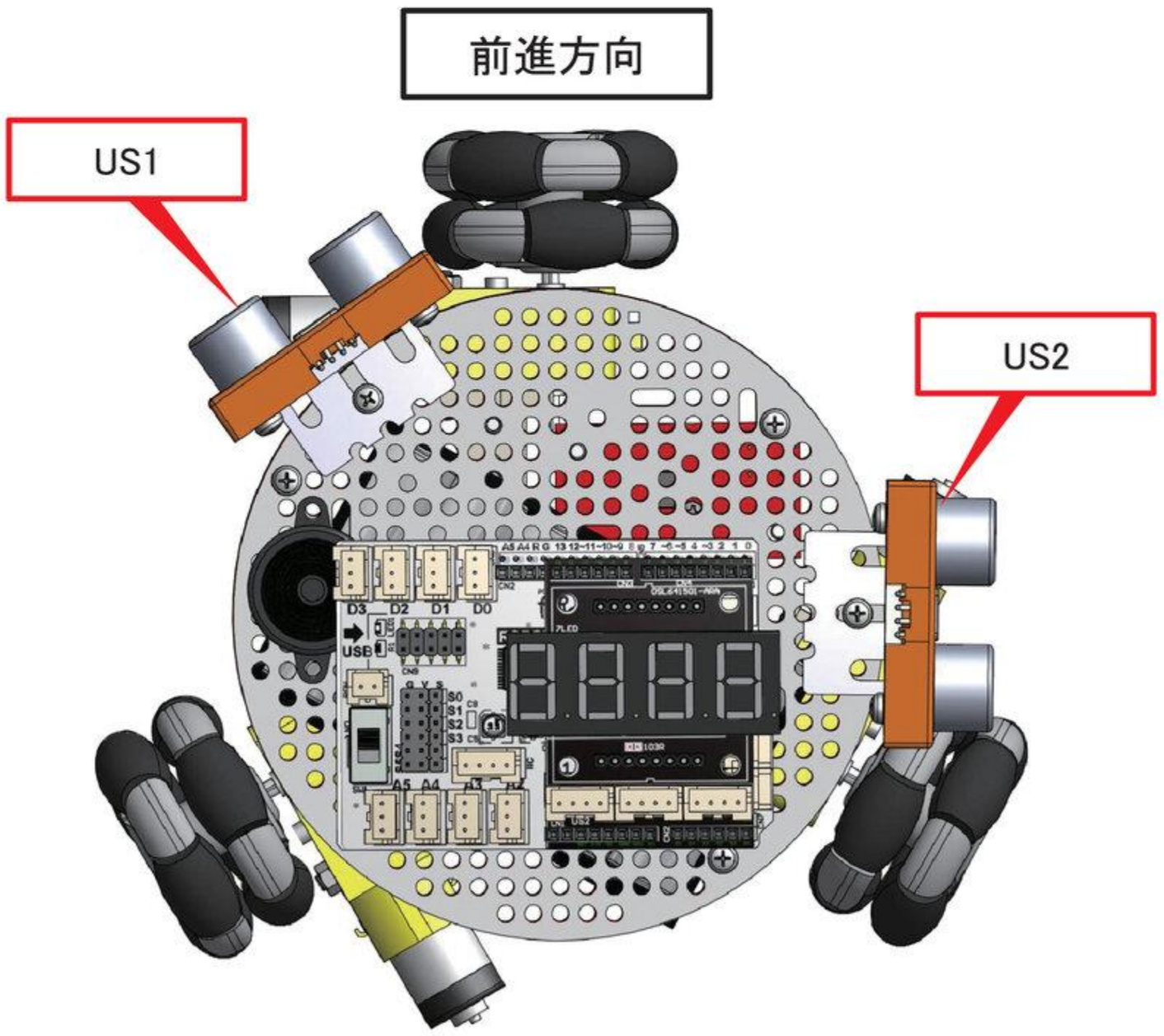


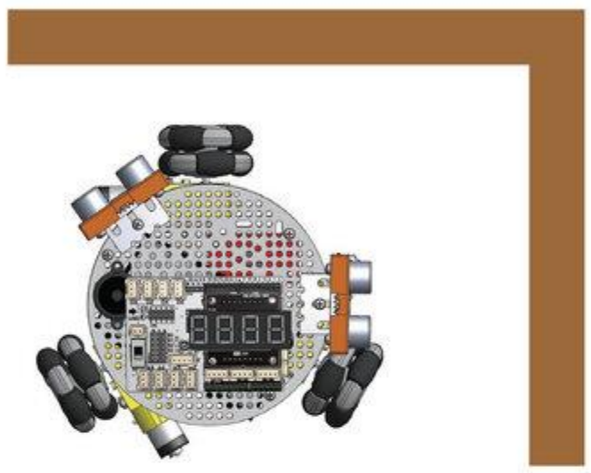
図 1-0　超音波距離センサーの配置

2）各パターンの状況を推察する

やってみよう！

パターン１～４でのロボットに対しての壁（かべ）の位置をイラストにし、必要な動作を文章にしてみよう。パターン１は参考になるようにかいておいたよ。なお、超音波距離（ちょうおんぱきょり）センサーの US1 は、実際は進行方向に対して、やや左向きで取り付けているけど、ここでは真正面を向いていると考えてね。

パターン１：US1（前方）反応あり／US2（右側）反応あり。

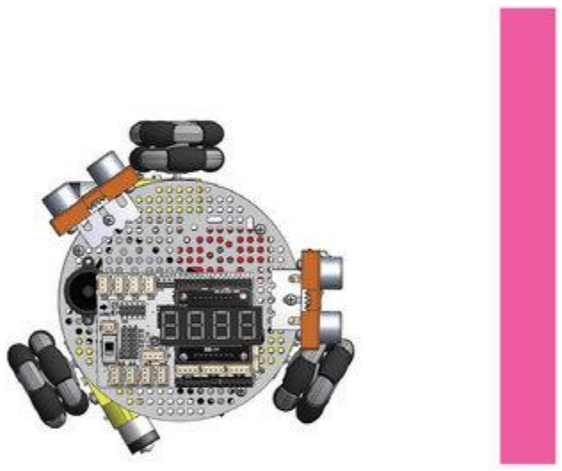


動作：

前方から壁（かべ）がなくなるまで反時計回りに

回転する。

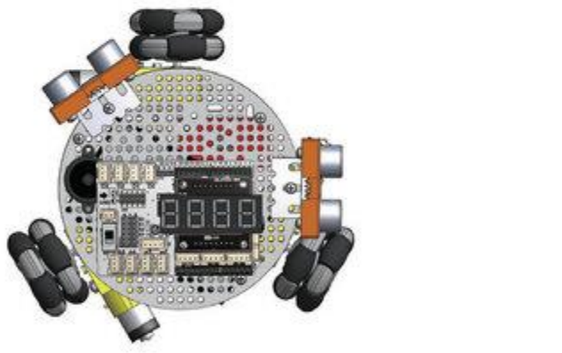
パターン２：US1（前方）反応なし /US2（右側）反応あり。



動作：

右に壁がある状態を保ちながら前進する。

パターン３：US1（前方）反応あり /US2（右側）反応なし。



動作：

右に壁が見つかるまで、右側に移動する。

パターン４：US1（前方）反応なし /US2（右側）反応なし。

動作：

右に壁が見つかるまで、右側に移動する。

1.3. ステップ③ フローチャートで整理する

パターン１～４までの動作が確認できたら、次にフローチャートで１つのプログラムにまとめましょう。ここではゴールアクションの機能も追加して考えてみますよ。

やってみよう！

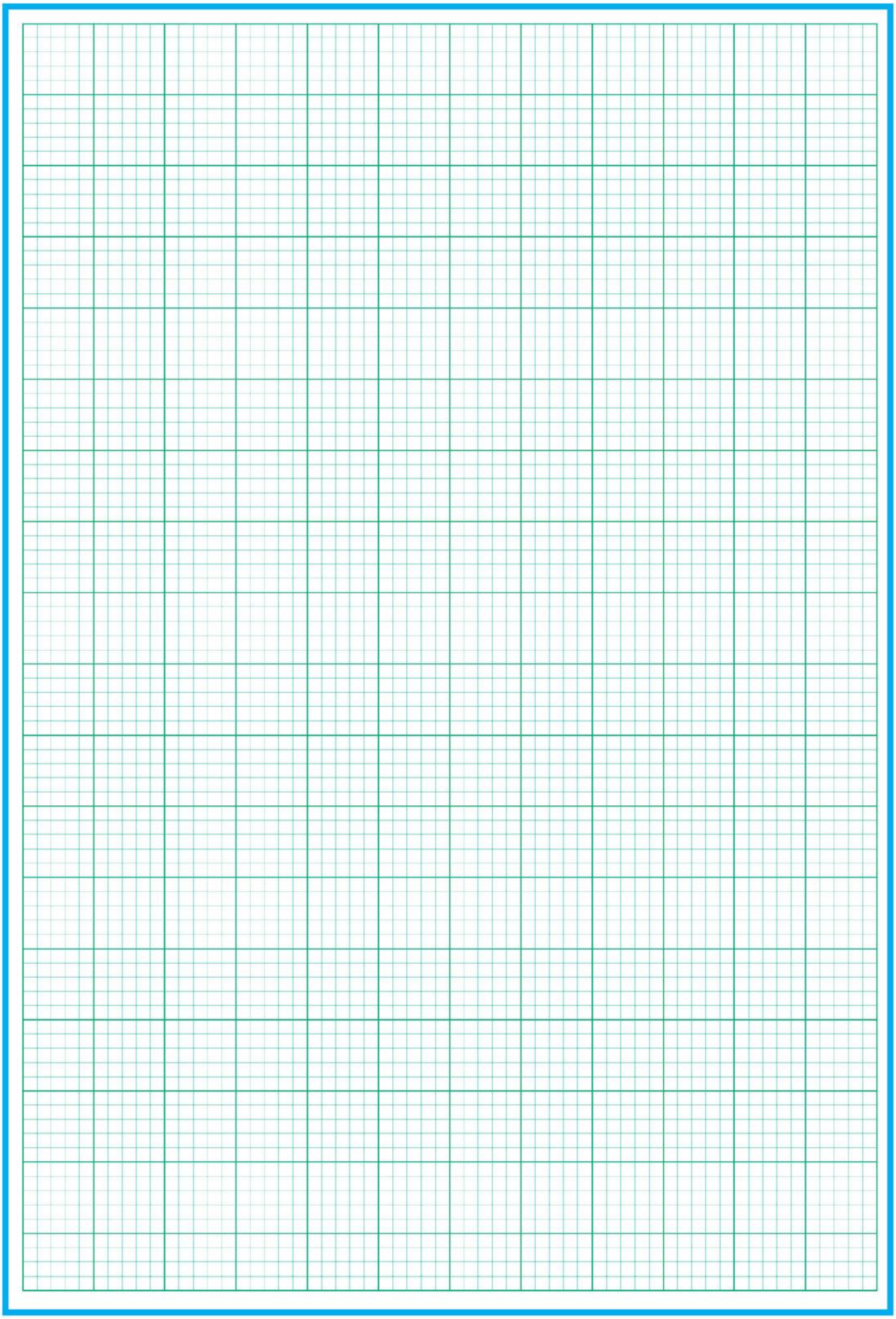
「迷路ぬけロボット」のプログラムをつくるため、まずはフローチャートにまとめてみよう！
今回は「開始」から「終了」のブロックまですべて１からかいていこう！　以下の条件を取り入れてつくってみてね。

POINT

「迷路ぬけロボット」の条件

① 超音波距離センサーの値を読み込み、右手法を使って迷路を攻略する。
② 迷路のゴール地点には緑色のカードを置いておき、カラーセンサーで読み取ることでゴール判定をする。
③ もしゴール地点にたどり着いていたら停止し、ゴール動作として「喜びの音を出す」「7 セグメント LED をグルグル光らせる」という 2 つの動作を行う（前回つくった「落ち着き動作」と同じだよ！）。

フローチャートは次のページの方眼紙にかき込もう！



フローチャートの解答例は次の通りです。

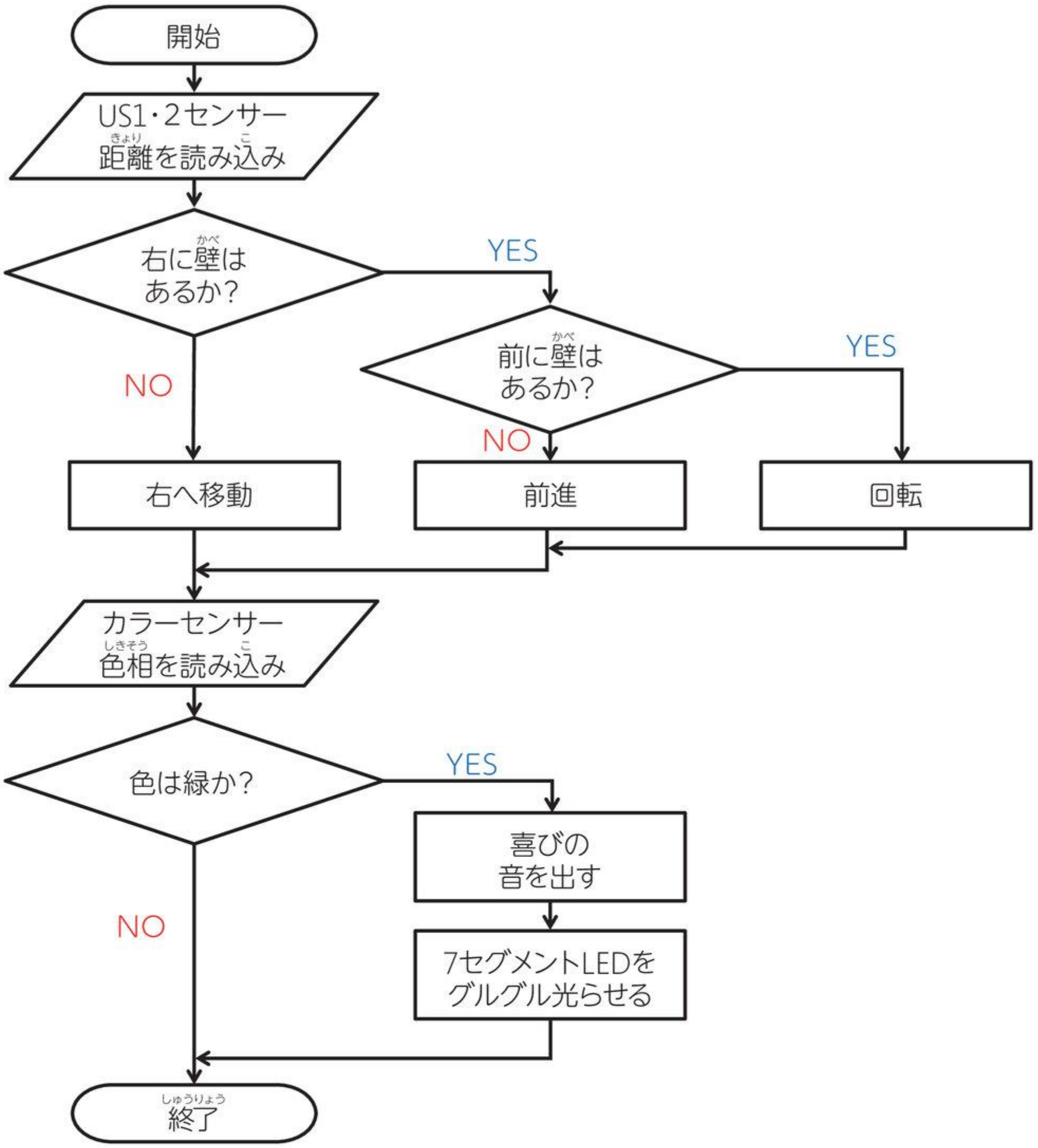


図 1-1 「迷路ぬけロボット」フローチャート

1.4. ステップ④ プログラムを完成させる

フローチャートが完成したら、プログラム化していきましょう。
まずは、以下のプログラムを実行します。

プログラムの書き込み
RoboticsProfessorCourse2 > CombRobot4 > Case1

実行結果：ロボットの前方および右に障害物があると反時計回りに回転する。同時に、７セグメント LED に前方の障害物までの距離を表示する。

「パターン 1」にあたる場合ですね。このプログラムをベースに、ほかのパターンのときの動作を追加して、迷路をぬけるまでの処理を完成させましょう。

やってみよう！

プログラム「Case1」をかきかえ、「右手法」をつかって迷路を進む動作を完成させよう！

ヒント

フローチャートを見れば、何通りの動きが必要かすぐわかるね！　ちなみに、「今どのパターンの動きをしているか」を 7 セグメント LED に表示させるようにすると、正しく動作しているか確認しやすくなるよ！　以下の黄色の部分を追加してみよう！

```
if (dist1 <= 6 && dist2 <= 10){   // もし前方の壁まで6センチ以下かつ、
                                     右の壁まで10センチ以下なら
    lc.setDec(0, 1);
    omniBot.move(0, 0, -20);      // ロボットは左回転
    while (ussRead(US1) < 20 || ussRead(US2) > 10)
                                  // 前方が20センチ以上開けていて右
                                     10センチの範囲に壁が見えるまで
                                     回転を継続
    delay(600);                   // ***調整箇所*** ロボットが正面を
                                     向くように値を調整します。
}
```

lc.setDec(0, x); という命令をかくことで、7 セグメント LED に x の部分の数字を表示することができるよ！
ただし、このときはプログラムの終わりにある lc.setDec(0, dist1); の部分は削除しておこう。

講

解答例は以下のプログラムです。
・前方に壁がなく、右に壁があるとき
　RoboticsProfessorCourse2 > CombRobot4 > Case2
・（前方の壁の有無に関係なく）右に壁がないとき
　RoboticsProfessorCourse2 > CombRobot4 > Case3
・すべての動きを合体させた完成版プログラム
　RoboticsProfessorCourse2 > CombRobot4 > CaseAll
完成版プログラムの解答例は巻末にも記載します。

プログラムの修正だけで迷路の脱出がうまくいかない場合には、**図 1-2** のように超音波距離センサーの取り付け角度を変化させることでロボットの動きを調整できます。ソフトウェアとハードウェアの両方をうまく調整してみましょう！

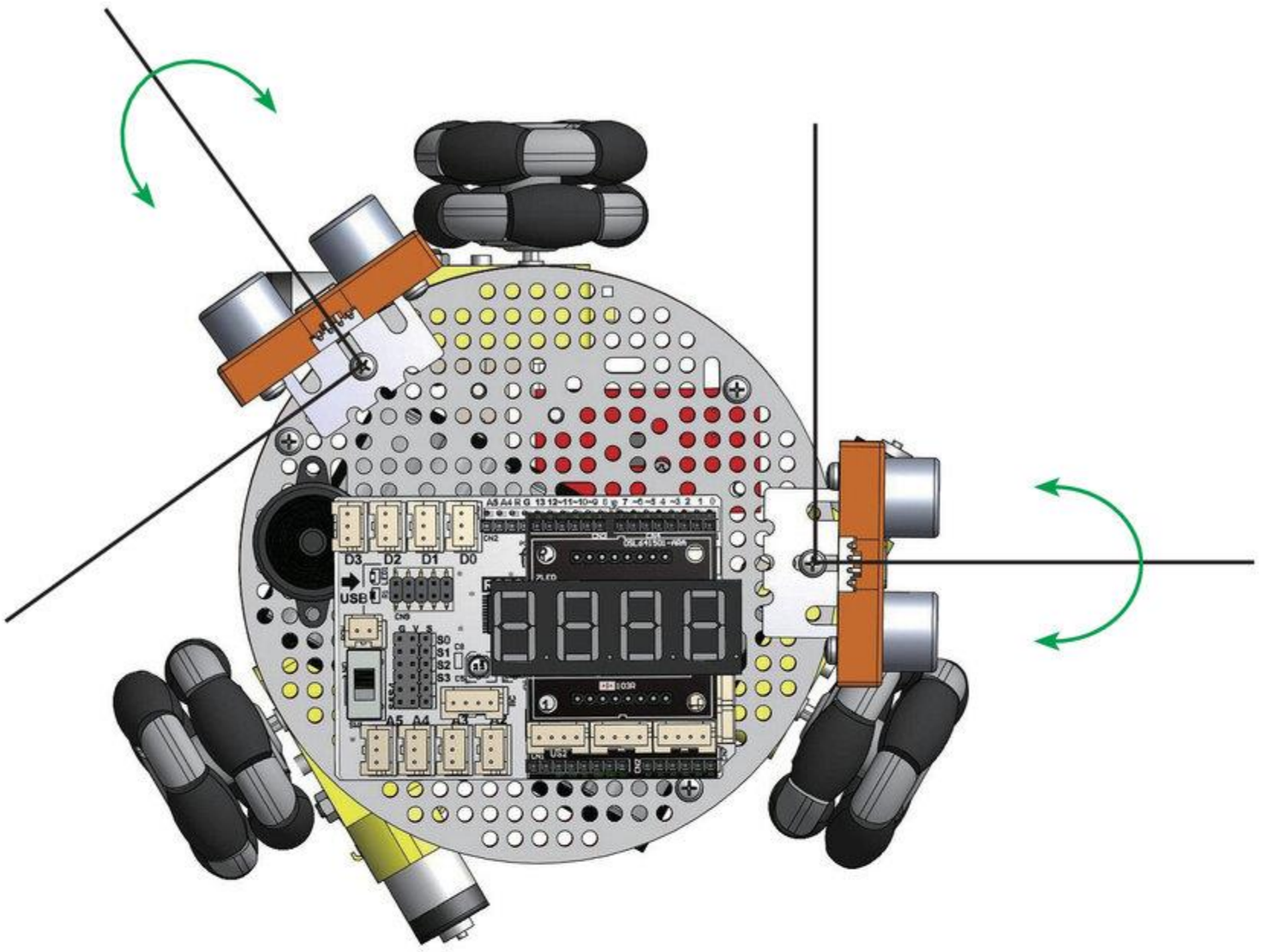


図 1-2　ハードウェア（超音波距離センサーの向き）の調整

1.5.　ステップ⑤ ゴール処理のプログラム

1）プログラムの実行

迷路が脱出できるようになったら、ゴール時に実行する、歓喜のパフォーマンスをつくりましょう！　前回つくった「暗がりが好きなロボット」のプログラムを一部使います。では、以下のプログラムを実行してみましょう。

プログラムの書き込み
RoboticsProfessorCourse2 ＞ CombRobot4 ＞ GoalFunc1

実行結果：緑色を検知すると音が鳴り、7 セグメント LED の周りをルーレットのように光がまわる。

内容は前回のプログラムとほとんど同じですが、暗がりをゴールとするのではなく、緑色のゴール(巻末のゴールシート)を使うように変更してあります。動作環境によってゴール判定値(h 値)の微調整が必要なので、ゴールシートの上にロボットをのせたときだけ、ゴールパフォーマンスが実行されるように if 文の条件を調整しましょう！

2）関数のおさらい

ところで、このプログラムでは、前回学んだ「関数」を使って、メインのループの外にゴールパフォーマンスの処理がかかれていることに気がついたでしょうか？

プログラム「GoalFunc1」より抜粋

```
if(h >= 100 && h <= 140){        // 緑(H=120)を検出
    // ゴールの処理
    goal();
}

(中略)

//------------------- オリジナル関数
void goal(){                     // 名前は好きに決めることができる(※すでに用意されている名前でなければ)
    lc.clearDisplay(0);          // 表示クリア
    tone1.play_rtttl(sound2);    // sound2の曲を流す
    for(int i = 0; i < 12; i++){
        lc.setLed(0, rx[i], ry[i], HIGH);
        delay(100);
        lc.setLed(0, rx[i], ry[i], LOW);
    }
}
```

ここで少しだけ関数のおさらいをしましょう。

POINT

関数のポイント

・ある決まった処理を行う命令のかたまり。
・関数名の最後に（）がつく。
・命令の内容はメインのループの外にかかれている。
・命令の集まりを新しいオリジナルの命令としてつくることができる。

プログラムの中で関数（たとえば `goal()` ）がよび出されると、プログラムは、メインのループの外にかかれた関数にジャンプして内容を実行し、終わると再びよび出された行へともどる動きをするのでしたね。関数の名前は好きに決めることができるので、ここでは `goal()` という名前にしてみました。

3）プログラムの改造

さて、関数について思い出したところで、このゴールパフォーマンス用の関数 goal() を、もう少し高機能にしていきます。

やってみよう！

プログラム「GoalFunc1」をかきかえ、7 セグメントの光が 1 周ではなく 3 周になるようにしてみよう！

ヒント

「1 周する命令を 3 回くり返す」という考え方ができるといいね！　for 文を使ってみよう！

for 文を使った解答例は以下の通りです。

```
void goal(){                          // 名前は好きに決めることができる(※すでに用意
                                         されている名前でなければ)
    lc.clearDisplay(0);               // 表示クリア
    tone1.play_rtttl(sound2);         // sound2の曲を流す
    for(int j = 0; j < 3; j++){
        for(int i = 0; i < 12; i++){
            lc.setLed(0, rx[i], ry[i], HIGH);
            delay(100);
            lc.setLed(0, rx[i], ry[i], LOW);
        }
    }
}
```

変数 j を 0,1,2 と増加させ、そのたびに 7 セグメント LED の光を 1 周させています。for 文を「3 回くり返す」という処理のために使っていますね。

終わりの条件 j < 3 の部分をかきかえれば光が何周するかも指定できますが、「関数」の新たな機能について学ぶため、以下のプログラムを見てみましょう。

プログラムの書き込み

RoboticsProfessorCourse2 > CombRobot4 > GoalFunc2

プログラム「GoalFunc2」より抜粋

```
if(h >= 100 && h <= 140){          // 緑(H=120)を検出
    // ゴールの処理
    goal(3);
}

(中略)

void goal(int lnum){               // 名前は好きに決めることができる(※すでに
                                      用意されている名前でなければ)
    lc.clearDisplay(0);            // 表示クリア

    tone1.play_rtttl(sound2);      // sound2の曲を流す

    for(int j = 0; j < lnum; j++){ // 好きな回数をまわす
        for(int i = 0; i < 12; i++){
            lc.setLed(0, rx[i], ry[i], HIGH);
            delay(100);
            lc.setLed(0, rx[i], ry[i], LOW);
        }
    }
}
```

実行結果：「GoalFunc1」と同じように、7 セグメント LED の光が 3 周する。

for 文の終わりの条件が `j < 3` から `j < lnum` という、変数を用いた式にかわっていますね。

この変数 `lnum` というのがどこで宣言されているかというと、`void goal(int lnum)` の部分です。 `goal()` の `( )` の中に変数が入っているわけですね。

つまり「 `goal();` の ( ) の中に変数が入ります、その変数は 7 セグメント LED の周回数に使われます」と宣言しているのです。あとは、goal 関数を使うときに、何周させたいかの数字を一緒にかくだけで済みます。

実際、`void loop()` 内で goal の関数を使っているところを見てみると、`goal(3);` となっていますね。こうかくだけで、光を 3 周させるという命令に早変わりするのです。

このように、関数には好きな値を指定できる機能があります。指定する数値のことを「引数」といいます。

実は、みなさんも今までにたくさん引数を使ってきました。

たとえば、ギアドモーターを回転させるとき、`mc1.rotate(100);` のようにかいていましたよね。

これは `rotate()` という関数に `100` という引数を入れて、`mc1` と名付けたモーターに実行させるという意味です。

`rotate()` という関数はライブラリファイル内で宣言されています。

ライブラリファイル「RPlib」より抜粋

```
void RPmotor::rotate(int velo)
```

少しいつもとちがうかき方が混じっていてわかりにくいですが、やはり変数 velo を使う、と宣言(せんげん)しています。これで、( ) 内にかかれた数値を引数(ひきすう)として、モーターの回転速度の指定に使えるようになっているのです。

ちなみに、迷路ぬけロボットにも使われている omniBot.move(0, 0, -20); というのも move() という関数を利用しています。ただ、よく見ると今回は引数(ひきすう)が 3 つあります。宣言(せんげん)を見てみましょう。

ライブラリファイル「RPomniDirect」より抜粋(ばっすい)

```
void RPomniDirect::move(float vx, float vy, float  w)
```

今までとちがい、整数しか入れられない int 型ではなく、小数も入れられる float 型で変数を宣言(せんげん)しています。そして、変数が vx , vy , w と 3 つ宣言(せんげん)されていますね。このように、カンマ（,）で区切ることで引数(ひきすう)をいくつも使うことができるようになります。さっそく確認してみましょう。

ステップアップ

プログラム「GoalFunc2」をかきかえ、goal 関数を「喜びの音を 2 回鳴らし、LED の光を 3 周させる」という関数にしてみよう！

ヒント

LED の周回数は変数 lnum を引数にして指定できるから、音を鳴らす回数も同じようにしてみよう！

講

解答例プログラムは以下となります。
RoboticsProfessorCourse2 > CombRobot4 > GoalFunc3

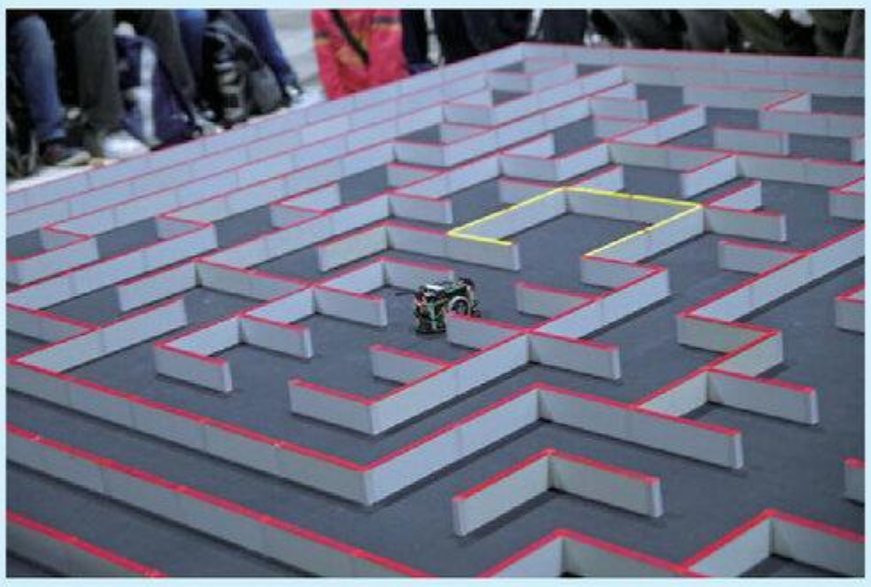
goal(3,2); のように引数を 2 つ定義します（数字はかえられます）。そのうえで、以下のように「オリジナル関数」の変数を増やします（名前はかえられます）。
void goal(int lnum, int snum)

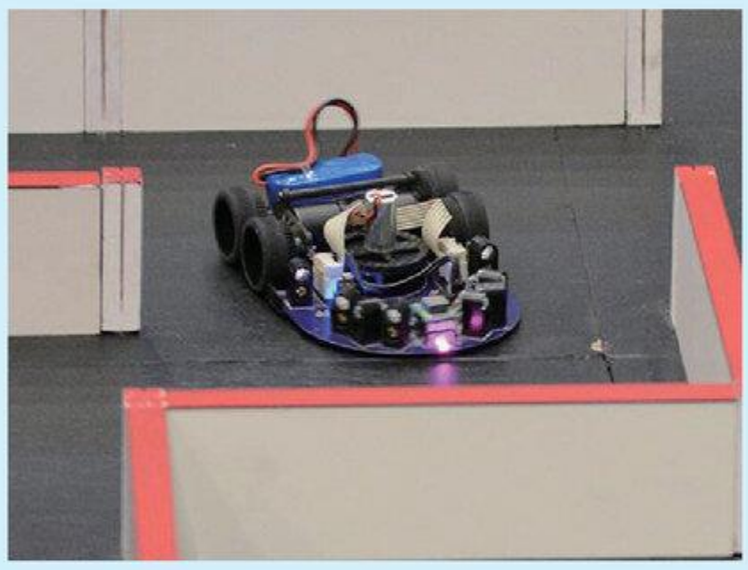
解答例は巻末にも記載します。

2. マウスオムニロボット（目安 20 分）

コラム　マイクロマウス競技とは

マイクロマウス競技は、ロボットが迷路を走りぬける速さと知能を競う競技です。主に小型のロボットが使用され、競技に出場するロボットは「マイクロマウス」とよばれます。日本で競技が始まったのは 1980 年からです。ロボット競技としては長い歴史を持っていますが、技術の競技ですから、毎年常に最先端（さいせんたん）の技術で競っています。
コースは、縦横それぞれ 16 区画、なんと合計 256 もの区画があります。ちなみに、マイクロマウス競技は、１走目で迷路の地図をつくり（探索（たんさく）走行）、２走目はものすごい速さでゴールを目指します。つまり真の競争は２走目なのです！





2.0.　プログラムをつなぎあわせる

さて、ここまではロボットに搭載（とうさい）する各機能や動作を分割して説明してきましたが、最終的にはコラムで紹介した「マイクロマウス」のようなロボットを目指していましたね。最後に全てを合体して「マウスオムニロボット」を完成させます。

ステップアップ

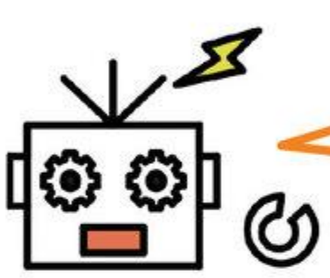
今回学んできたプログラムをうまく合体させ、「迷路ぬけロボット」を完成させよう！

講　解答例は以下のプログラムです。
RoboticsProfessorCourse2 > CombRobot4 > OmniMouse

マウスオムニロボットが完成したら、最速の迷路脱出ロボットを目指してください！
ロボットの走行速度や、曲がり方、センサーの反応距離（きょり）などを調整するとタイムを短縮できます。改良のパターンは複数あるので、好きに試してみましょう。

3. まとめ（目安５分）

今回は、迷路を脱出するロボットをつくってみました。迷路だけでなく、段ボール箱等の障害物の周りや教室の壁に沿っても走れます。友達と競争してみても良いかもしれませんね。
次回からはまた新しいセンサーを使っていろいろとできることを増やしていきましょう。



トライアルエラーで迷路脱出できたかな？

≪次回必要なもの≫

次回は、エンコーダー基板の使い方を学びます。今回製作したセンサーロボットに加えて、以下のものを準備してきてください。

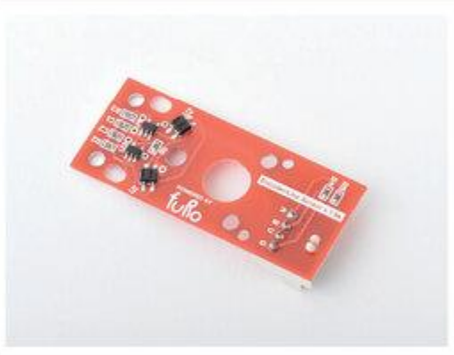
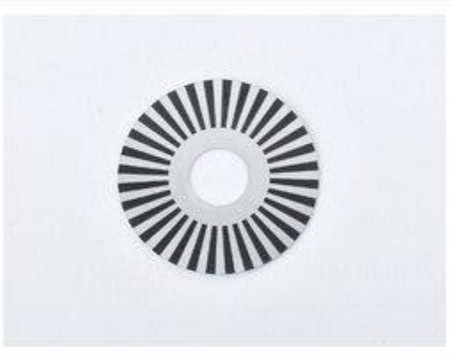
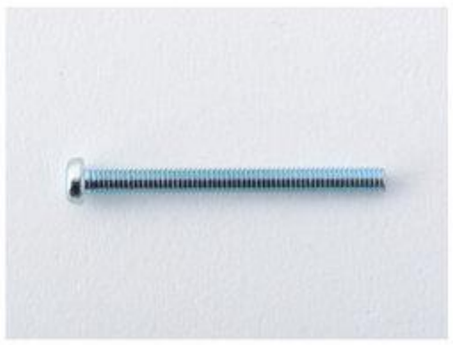
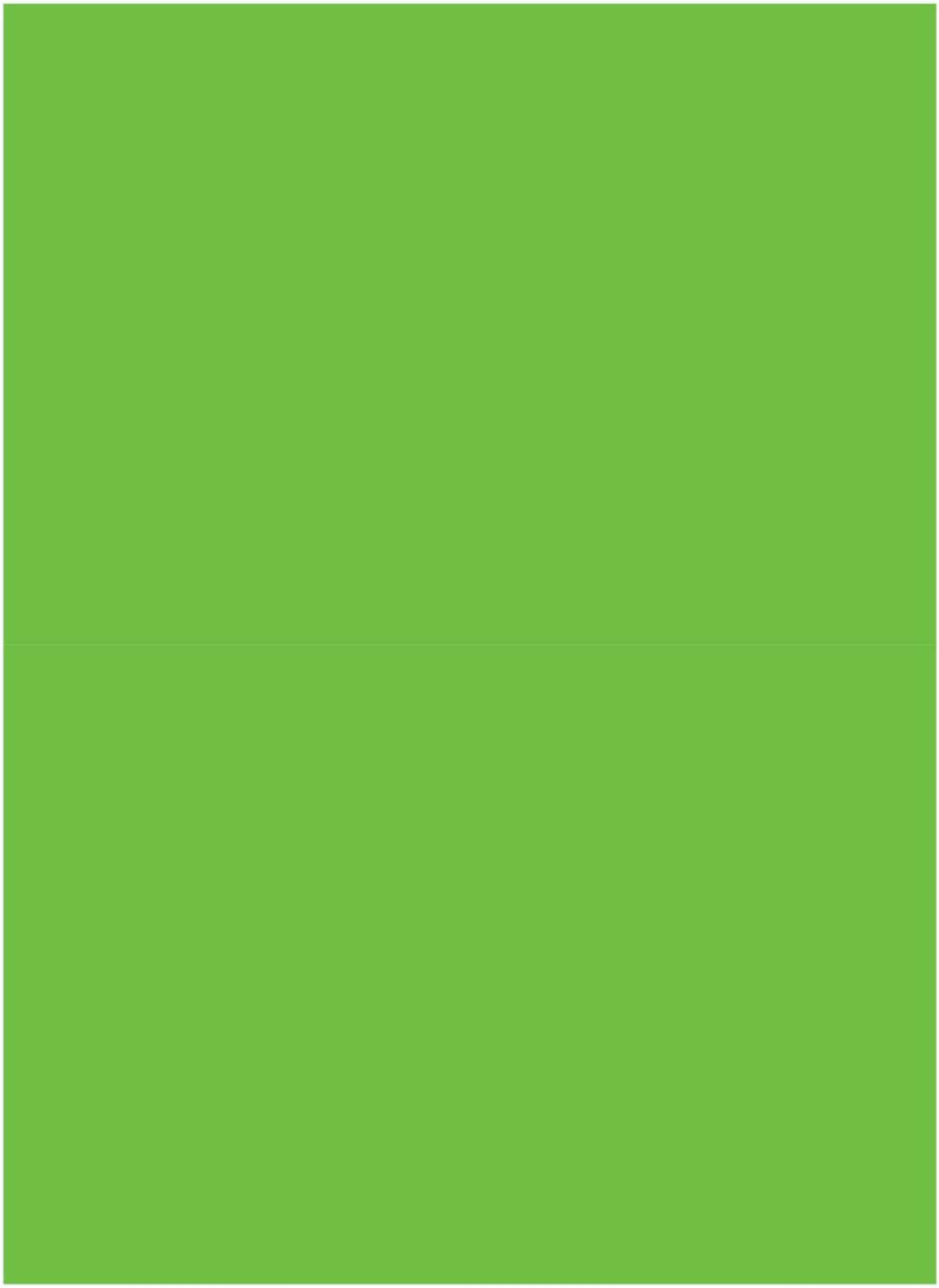
ラジオペンチ 1	ドライバー 1	USB ケーブル 1	タッチセンサー 1
			
姿勢検出シールド 1	エンコーダー基板 2	エンコーダーディスク 2	エンコーダー用シール 2
			
M2.3L25 ネジ 2			
			

図 3-0　次回必要なもの

講

○　以下の内容を振り返りながら生徒の理解を確認していきます。
・マイクロマウスタイプのロボットをつくる
・オリジナル命令「関数」をつくってみる

○　次回のテーマは「エンコーダーを使う」であることを告知します。



ゴールシート（H 値 120 度）

P.13　やってみよう！　解答例（CaseAll）

```
void loop(){
    int dist1 = ussRead(US1);              // センサーUS1検知
    int dist2 = ussRead(US2);              // センサーUS2検知

    if (dist1 <= 6 && dist2 <= 10){        // 前方の壁まで6センチ以下、右の壁まで
                                              10センチ以下のとき
        lc.setDec(0, 1);                   // ケースナンバーを表示
        omniBot.move(0, 0, -20);           // 左回転
        while (ussRead(US1) < 20 || ussRead(US2) > 10)
                                           // 前方が20センチ以上開けていて右10セ
                                              ンチの範囲に壁が見えるまで回転を継続
        delay(600);                        // ***調整箇所*** ロボットが正面を向く
                                              ように値を調整します。
}
else if (dist1 > 4 && dist2 <= 12){// 前が4センチより開けていて、右側の壁
                                              まで12センチ以下のとき
    lc.setDec(0, 2);                       // ケースナンバーを表示
    // 右の壁との距離を8センチに保って走ることを考えます
    if (dist2 == 8){                       // (右の壁まで12センチ以下で、)距離が8
                                              センチのとき
        omniBot.move(0, 20, 0);            // 直進
    }
    else if (dist2 > 8){                   // (右の壁まで12センチ以下で、)8センチ
                                              より大きいとき
        omniBot.move(0, 20, 3);            // 右カーブしつつ前進(壁との距離を近づ
                                              ける)
    }
    else{                                  // (右の壁まで12センチ以下で、)8センチ
                                              未満のとき
        omniBot.move(0, 20, -3);           // 左カーブしつつ前進(壁との距離を遠ざ
                                              ける)
    }
}
else{
    lc.setDec(0, 3);                       // ケースナンバーを表示
    omniBot.move(0, 20, 10);               // 右側の壁を探して右カーブ走行
    delay(400);
}
delay(10);
}
```

P.19　ステップアップ　解答例（GoalFunc3）

```
void loop(){
    // カラーセンサー処理----------------------------------
    if (ColorSensor.colorRead()){                    //カラーセンサーの読み込み
        word clear = ColorSensor.colorClear();       //カラーセンサーの数値初期化

        word red   = ColorSensor.colorRed();         //カラーセンサーの赤要素検出
        word green = ColorSensor.colorGreen();       //カラーセンサーの緑要素検出
        word blue  = ColorSensor.colorBlue();        //カラーセンサーの青要素検出

        ColorSensor.convertHSV(red, green, blue); //RGBデータをHSVデータに変
                                                    更する
        word h = ColorSensor.convertH();
        word s = ColorSensor.convertS();
        word v = ColorSensor.convertV();
        // ---------------------------------------

        lc.setDec(0, h);                             //H値を７セグメントLEDで表示
                                                       する


        if(h >= 120 && h < -140){                    //緑(H=120)を検出
            // ゴールの処理
            goal(3, 2);
        }
    }
    delay(100);
}

// ------------------- オリジナル関数
void goal(int lnum, int snum){                       //名前は好きに決めることができ
                                                       る(※すでに用意されている名
                                                       前でなければ)
    for(int k = 0; k < snum; k++){                   //好きな回数で鳴らす
        tone1.play_rtttl(sound2);                    //sound2の曲を流す
    }

    lc.clearDisplay(0);                              //表示クリア

    for(int j = 0; j < lnum; j++){                   //好きな回数をまわす
        for(int i = 0; i < 12; i++){
            lc.setLed(0, rx[i], ry[i], HIGH);
            delay(100);
            lc.setLed(0, rx[i], ry[i], LOW);
        }
    }
}
```