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-2②

第4回

カラーセンサーロボット

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。

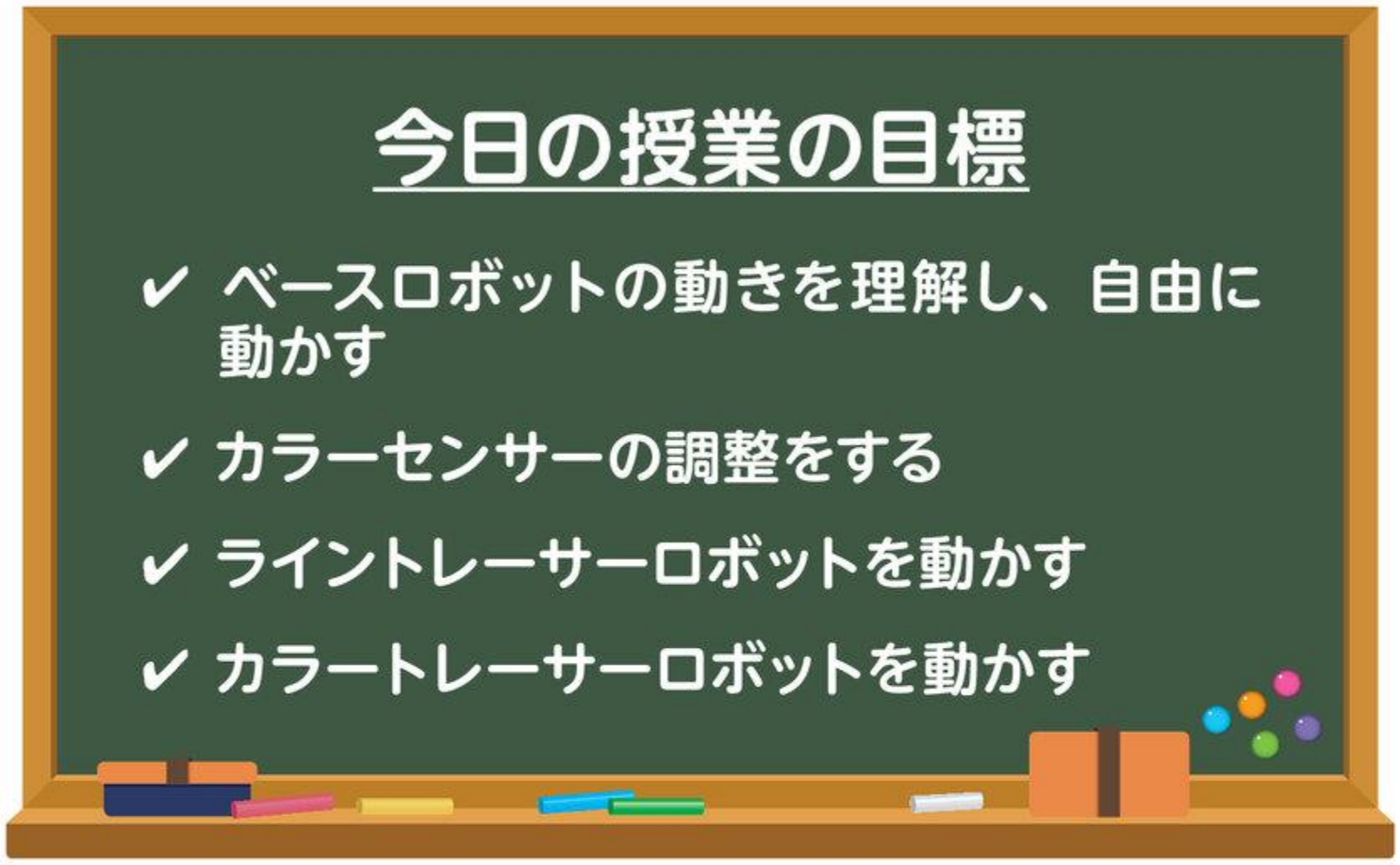
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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. カラーセンサーロボット（目安 10 分）

0.0. 「カラーセンサーロボット」でやること



今回の授業では、前回作ったベースロボットを自動で動けるように改造します！
まずは、モーターの動きをよく見て、「どう回転したらどう動くのか？」といった各モーターとロボット本体の動きの関係を理解します。
次に、カラーセンサーの調整を行ってから、ライントレーサーを作ります。知っている人も多いかもしれませんが、ライントレーサーとは、床にかかれた黒いラインを伝って動く、ロボット技術の基礎（きそ）になるロボットです。カラーセンサーを取り付けて、動きを考えるプログラムを作れば完成です！
「感じて」、「考えて」、「動く」ロボットの要素を学んでいきましょう！

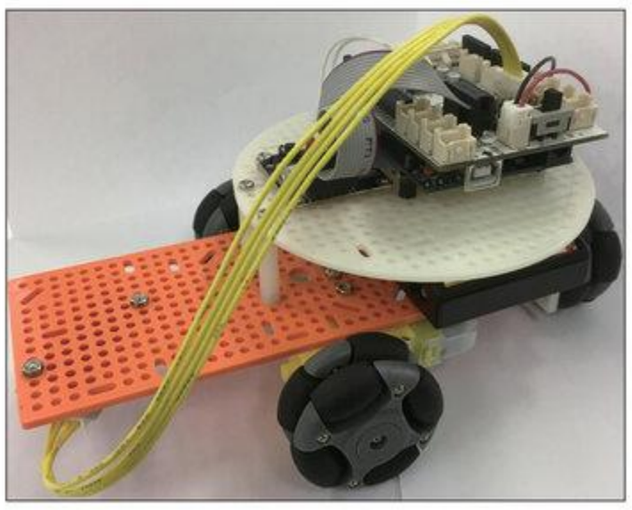
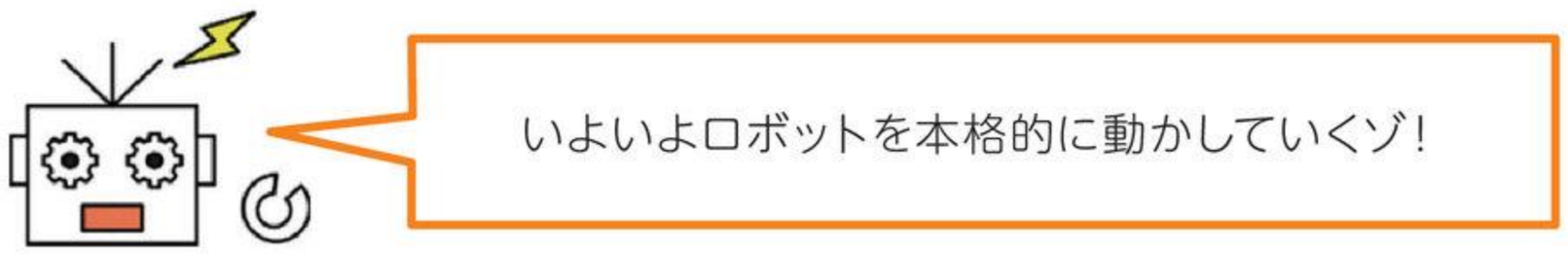


図0-0　ライントレーサーロボットの完成図



0.1. 必要なもの

前回作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。

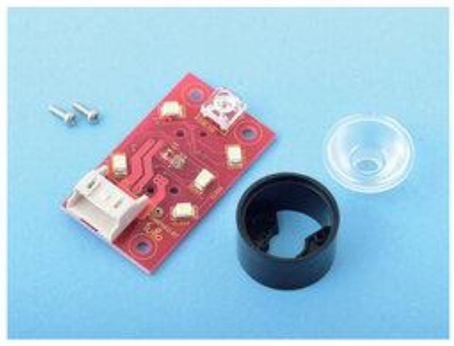
ラジオペンチ 1	ドライバー 1	USBケーブル 1	センサー L字ステイ 1
			
スピーカー 1	**カラーセンサー 1**	**センサーケーブル 1**	**M3L8ネジ 2**
			
M3ナット 6	**M3L10ネジ 2**		
			

図0-1　必要なもの

1．ライントレーサーロボット（目安 60 分）

1.0.　カラーセンサーの取り付け

では、ライントレーサーロボットを作りましょう。まずは、ベースロボットにカラーセンサーを取り付けます。

＜組み立て手順①＞

図1-0のように、カラーセンサーにネジどめをしておきます。使用するパーツは、M3L10ネジ（×2）とM3ナット（×2）です。

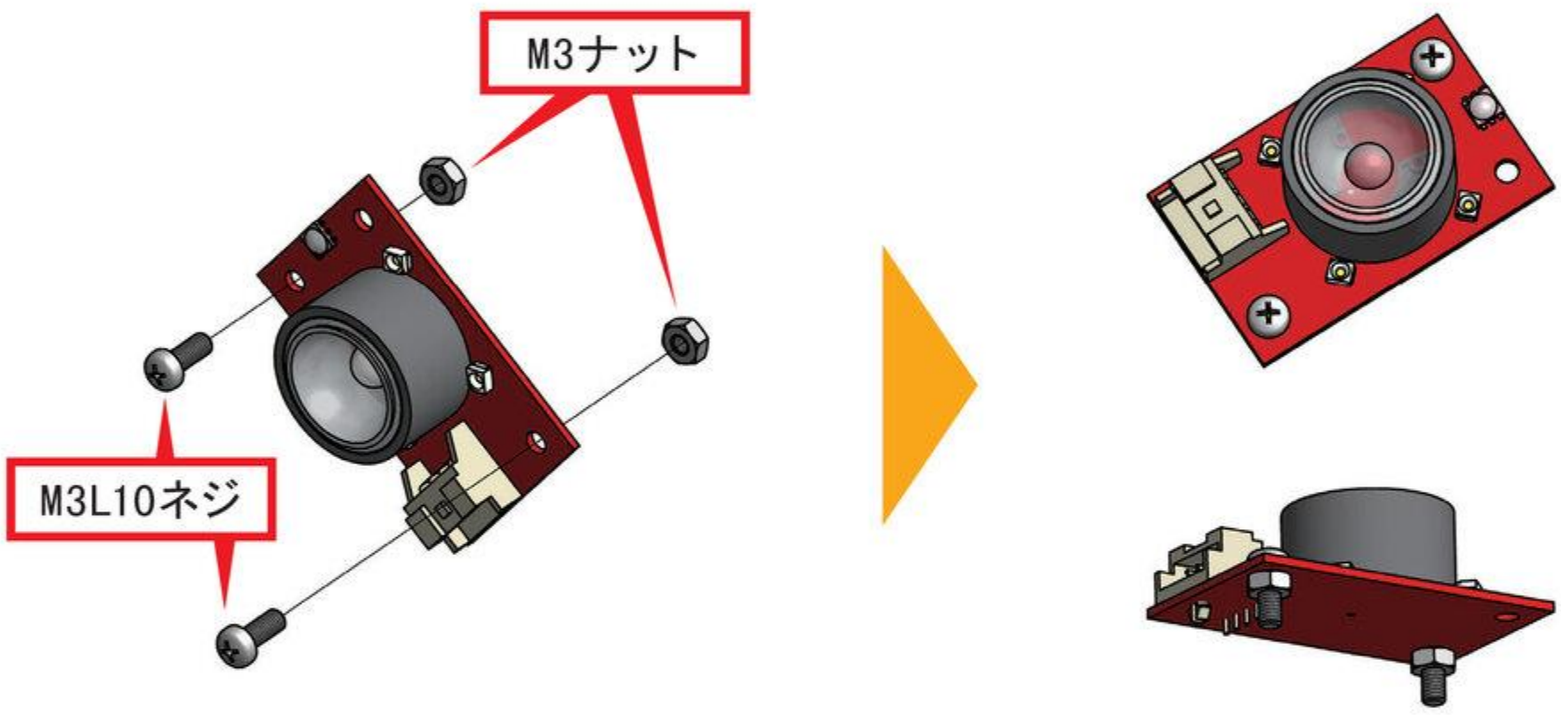


図1-0　カラーセンサーへのネジどめ

＜組み立て手順②＞

ネジどめをしたカラーセンサーを、M3ナット（×2）を使ってベースロボットの前部に裏(うら)向きに取り付けます。取り付け位置は図と多少ズレていても大丈夫です。自分なりに一番良いと思う場所を探して取り付けてください。

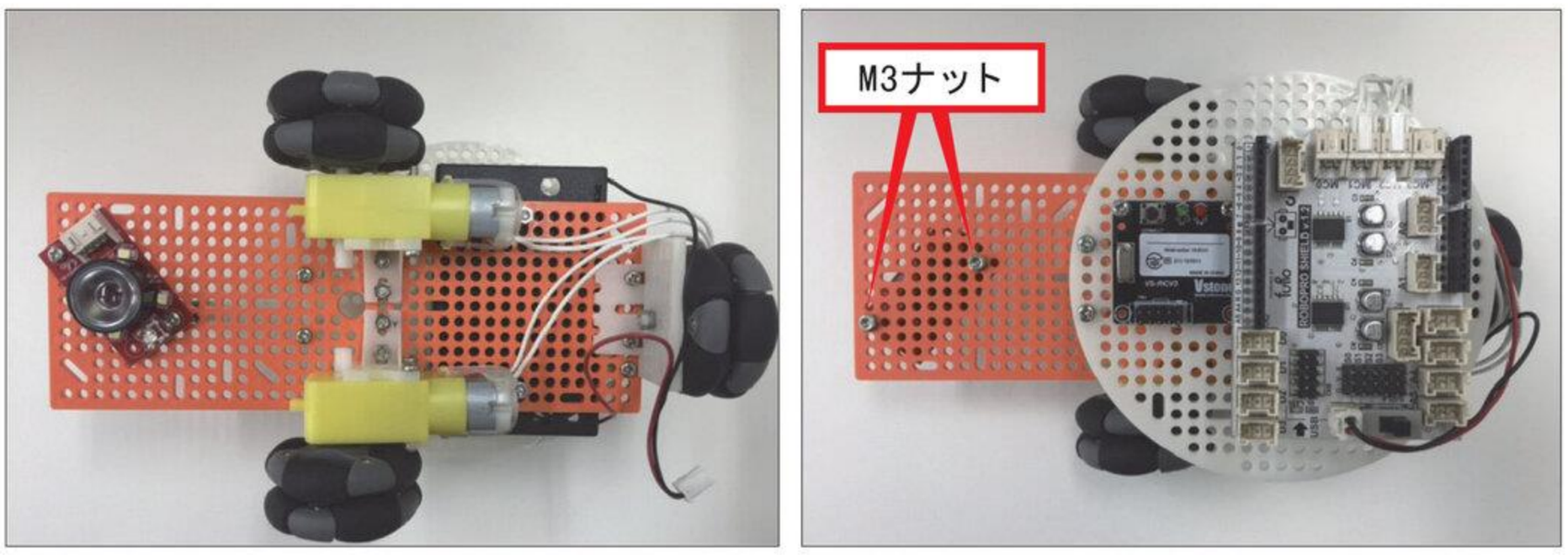


図1-1　カラーセンサーの取り付け

＜組み立て手順③＞

カラーセンサーを、センサーケーブルを使って、**図1-2**のようにロボプロシールドの IIC に接続します。

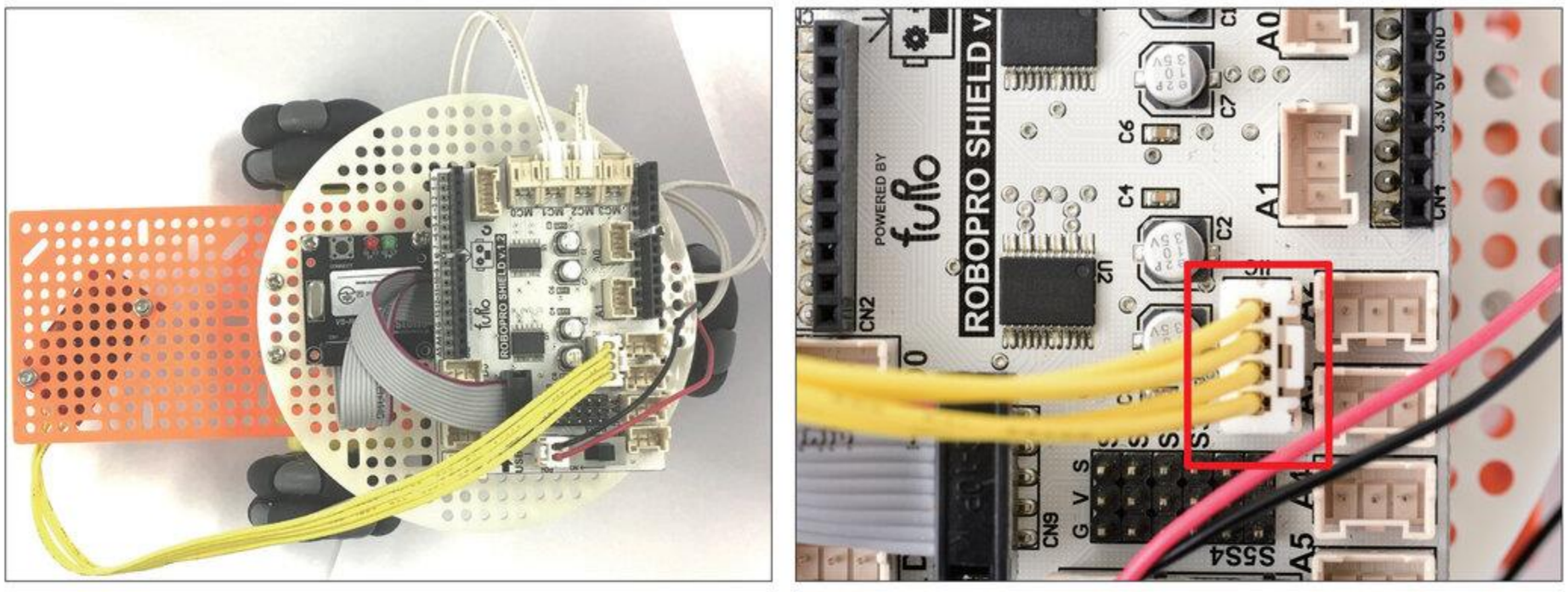


図1-2　カラーセンサーとロボプロシールドの配線

1.1. モーターとロボットの動きの関係

1）モーターの回転方向とロボット本体の動き

前回も少しふれましたが、ロボット本体を直進させるにはどうしたらよいでしょうか？　以下のプログラムを改造して、直進できるようにしましょう。

プログラムの書き込み
RoboticsProfessorCourse1 ＞ MagicItemB4 ＞ MoveTest

プログラムをそのまま実行すると、ロボットはその場で回転してしまいます。中身を確認すると以下のようになっています。

プログラム「MoveTest」より抜粋（ばっすい）

```
void loop(){
  mc1.rotate(50);// 50の速度で正方向に回す！（最大255）
  mc2.rotate(50);// 50の速度で正方向に回す！（最大255）
}
```

MC1 と MC2 に接続した各モーターの値（上記の黄色の部分）を変えてみましょう。そして、前進・後退のときに各モーターの値が、「＋」になるのか「－」になるのかを、**表1-0**に書いておきましょう。

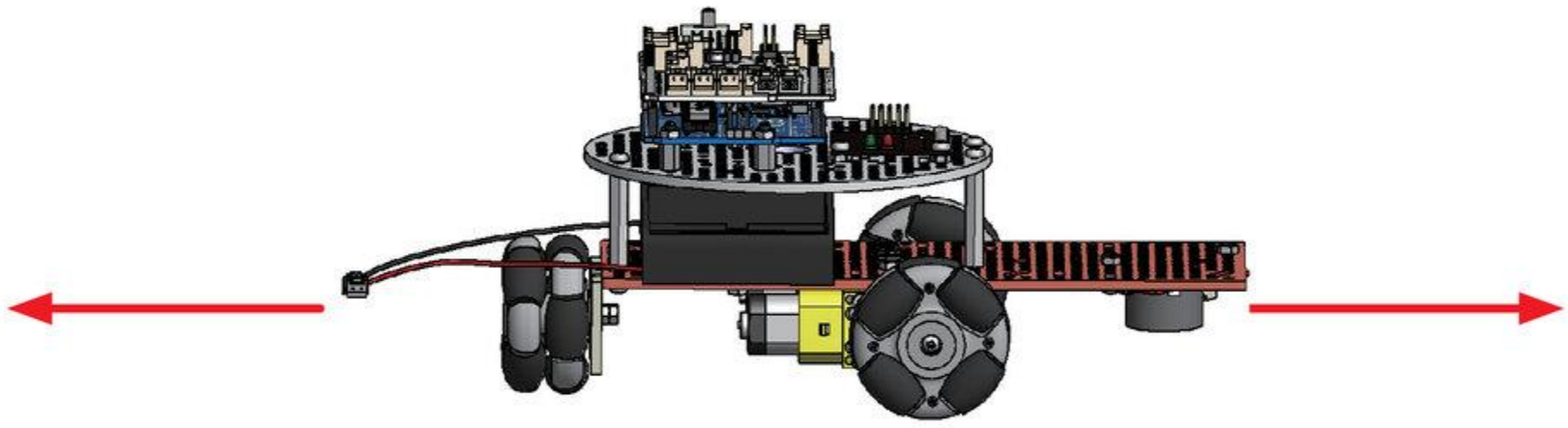


図1-3　まっすぐに走るロボット

表1-0　モーターの値とロボットの動き

	MC1	MC2
前進	－	＋
後退	＋	－
時計回りに回転	＋	＋
反時計回りに回転	－	－

同じように、回転もさせてみて、「＋」と「－」を**表1-0**に書き入れておきましょう。
ここでは、モーターの回転方向とロボット本体の動きの関係性をつかんでおきましょうね。

2）さまざまな動きをさせる

では、基本の動きが理解できたところで、今度はそれらを利用してさまざまな動きを実現してみましょう。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 > MagicItemB4 > MoveTest2

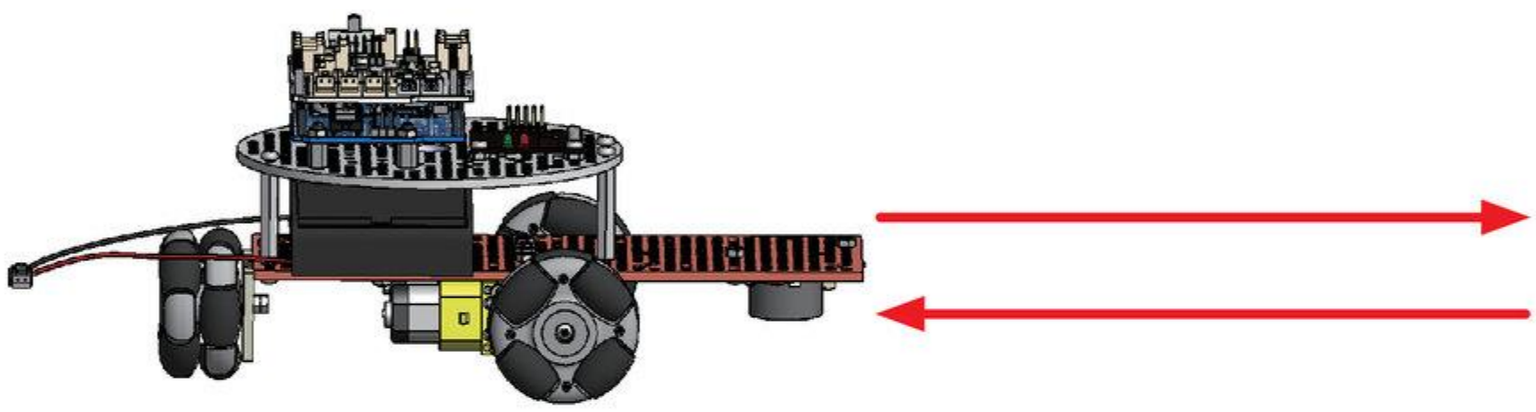
実行結果：ロボットが直進と回転を交互に繰り返す。

やってみよう！

プログラム「MoveTest2」を書きかえ、ロボットが前進したあと、同じ時間だけ後退してもとの場所に帰ってくるのをくり返すようにしてみよう！

ヒント

書きかえる前の「MoveTest2」は、「2秒間前進したあと、1秒間回転する」という動きになっているよ！



ロボットの動かし方には慣れてきましたか？

講

解答例は以下の通りです。

```
void loop(){
    mc1.rotate(-50);
    mc2.rotate(50);
    delay(2000);
    mc1.rotate(50);
    mc2.rotate(-50);
    delay(2000);
}
```

時間の指定には、delay(); という命令が使われています。

命　　令　「delay」
実 行 内 容 ：直前の命令を、指定した時間だけ続ける
使　い　方 ：mc.rotate(100);
　　delay(2000);　// モーターを回転させる命令を2秒間続ける

時間の単位は、「秒」を1000分割した「ミリ秒」です。1000ミリ秒で1秒になるので、2秒間動作させたいときは 2000 という値を書く必要があります。

ステップアップ

プログラム「MoveTest2」を書きかえ、ロボットが以下のような図形をえがくようにしよう！　図形の大きさは自由だよ！

① 円
② 8の字
③ 正方形

講

解答プログラムはそれぞれ以下となります。
なお、床の材質や広さなどに合わせて数値の調整が必要です。
円：RoboticsProfessorCourse1 ＞ MagicItemB4 ＞ challenge1
8 の字：RoboticsProfessorCourse1 ＞ MagicItemB4 ＞ challenge2
正方形：RoboticsProfessorCourse1 ＞ MagicItemB4 ＞ challenge3
解答例は巻末に記載します。

1.2. カラーセンサーの調整

では、ロボットの動きがわかったところで、いよいよライントレーサーを動かしていきたいのですが、その前にカラーセンサーの調整を今一度しておきましょう。スピーカーを D2 に取り付けて、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 > MagicItemB4 > ColorSensor2

実行結果：カラーセンサーの読み取ったものが黒（暗い）と判定されると、スピーカーから音が出て、マイコンボード上のLEDが点灯する。

「もしカラーセンサーが黒を検知（けんち）したら、スピーカーから音を出す」ということで、第1回などでも登場した「if文」が使われています。

プログラム「ColorSensor2」より抜粋（ばっすい）

```
if(isBlack){
    digitalWrite(Led, HIGH);
    tone(Speaker, NOTE_F2, 50);
}
else {
    digitalWrite(Led, LOW);
}
```

黄色の部分が黒を検知（けんち）したときの動き、緑色の部分がそうでない、つまり白を検知したときの動きです。

digitalWrite という命令は、HIGH と書かれていたらLEDを点灯、LOW と書かれていたらLEDを消灯させます。

if文の( )内の条件は isBlack とだけ書かれています。つまり、このif文は「もし isBlack なら…」という意味になります。 isBlack とはどのような条件なのでしょうか。

講　読み込んだプログラムのままでは、環境によっては黒をうまく検知できないため、この後調整します。

大まかに説明すると、カラーセンサーが黒かどうかを判定するときには、検知した様々な値のうち「clear値」というものを使っています。
clear値は色が黒っぽいほど小さい値に、白っぽいほど大きい値になります。ですから「clear値が●●未満のときは黒！　それ以外は白！」と事前に決めておくことで、ロボットに白黒を判定させることができるのです。

プログラム「ColorSensor2」より抜粋

```
bool isBlack = ColorSensor.colorIsBlack(1000);
```

( ) 内の値が、白と黒の境目になるclear値です。この境目の値を「閾値」とよびます。
もしもclear値が閾値未満なら isBlack 、つまり「黒モード」をオンにする、という命令だとイメージしてください。つまり if(isBlack) というif文は「もし黒モードがオンなら…」という条件になるわけです。

( ) の中の値を書きかえることで、白と黒の境目を自由に変更できます。実際にロボットに黒い紙をかざし、きちんと反応するような値を探しておきましょう。

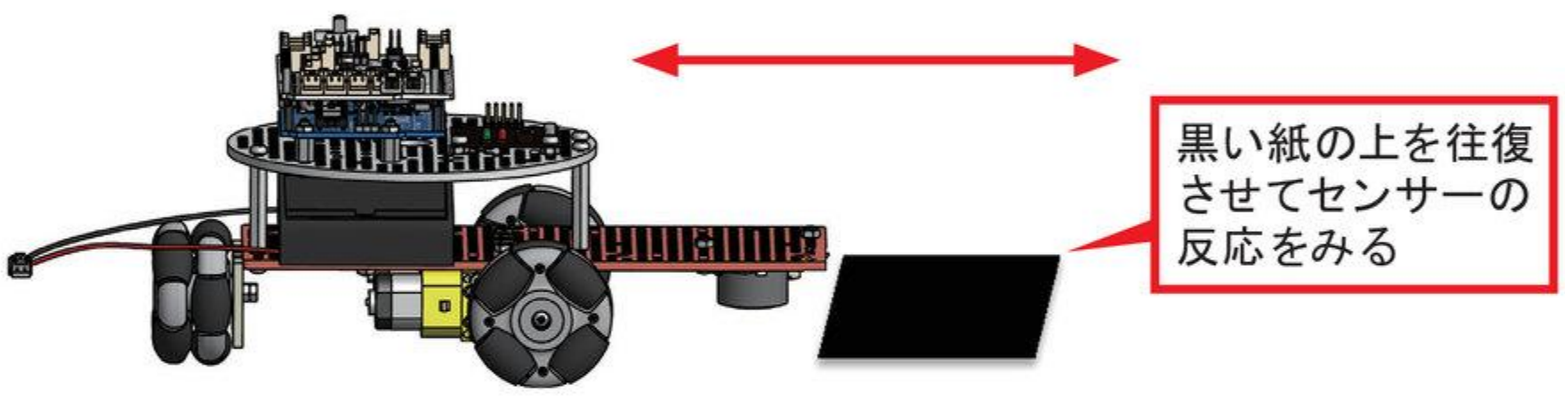


図1-4　カラーセンサーの調整方法

調整が済んだら、忘れないように書いておきましょう。

カラーセンサーの調整値は、

これで準備は終了です！

1.3. ライントレーサーロボットを動かす

それでは、これまでの内容をまとめてライントレーサーロボットを動かしてみましょう。以下のプログラムを実行してください。ライントレース用のコースは巻末のものを使ってください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemB4 > Tracer

やってみよう！

プログラム「Tracer」は、黒を検知（けんち）したときとそうでないとき、それぞれどのような動きをするようになっているかな？
プログラムを読みとり、下の解答らんにまとめよう！

黒のときは…　マイコンボードの LED を点灯し、スピーカーから音を出し、左旋回する。

それ以外は…　マイコンボードの LED を消灯し、右旋回する。

ヒント
実際にロボットを動かさなくても、プログラムを読めばわかるんじゃないかな？

コラム　黒検知とフラグ

プログラム「ColorSensor2」や「Tracer」では、前のページでも説明したとおり isBlack という条件を使って動きを分けていました。 isBlack のことを「黒モード」とよんでいましたが、興味のある人はこの機能についてもうすこし理解を深めてみましょう。

第2回で、赤・緑・青の3色のLEDの明るさを決めるのに red 、green 、blue などの「変数」を使ったことを覚えているでしょうか。変数 red は、ボタンを押した回数によって0、1、2、…といった数字に置きかわりましたね。
実は、isBlack も red などと同じ「変数」です。ただし、変数 red が「0」や「30」、「100」などさまざまな整数に置きかわれるのに対し、isBlack は「true」と「false」という2つの文字列のどちらかにしか置きかわれません。「true」は英語で「正しい、合っている」、「false」は「まちがっている、合っていない」という意味です。

変数 isBlack は、もしclear値が閾値未満であれば true 、そうでなければ false に置きかわります。
if(isBlack) というif文は正確には「もし変数 isBlack が true なら…」という条件だったのです。

このように「●●か、そうでないか」の2択で判別をするのであれば、true と false にしか置きかわらない変数を使った方が、コンピューターも処理が楽にすみます。このような変数の種類を「ブーリアン型」といいます。

ちなみに、ブーリアン型の変数が true になることを「フラグが立つ」などと表現することもあります。
もしカラーセンサーが黒を検知して変数 isBlack が true になったら、「黒フラグが立った」などと言えるわけですね。

2．カラートレーサーロボット（目安 30 分）

2.0.　カラーセンサーの準備

続いて、色に反応して動くロボットを製作します。第2回の巻末の付録の色見本を用意してください。赤、青、緑のページに反応して向かうロボットを作ってみましょう。

図 2-0　色に反応するロボット

それでは、ベースロボットのカラーセンサーの位置を変えましょう。**図2-1**が完成図となります。ロボットに下向きに付けたカラーセンサーを取り外し、センサーケーブルも抜いておきましょう。

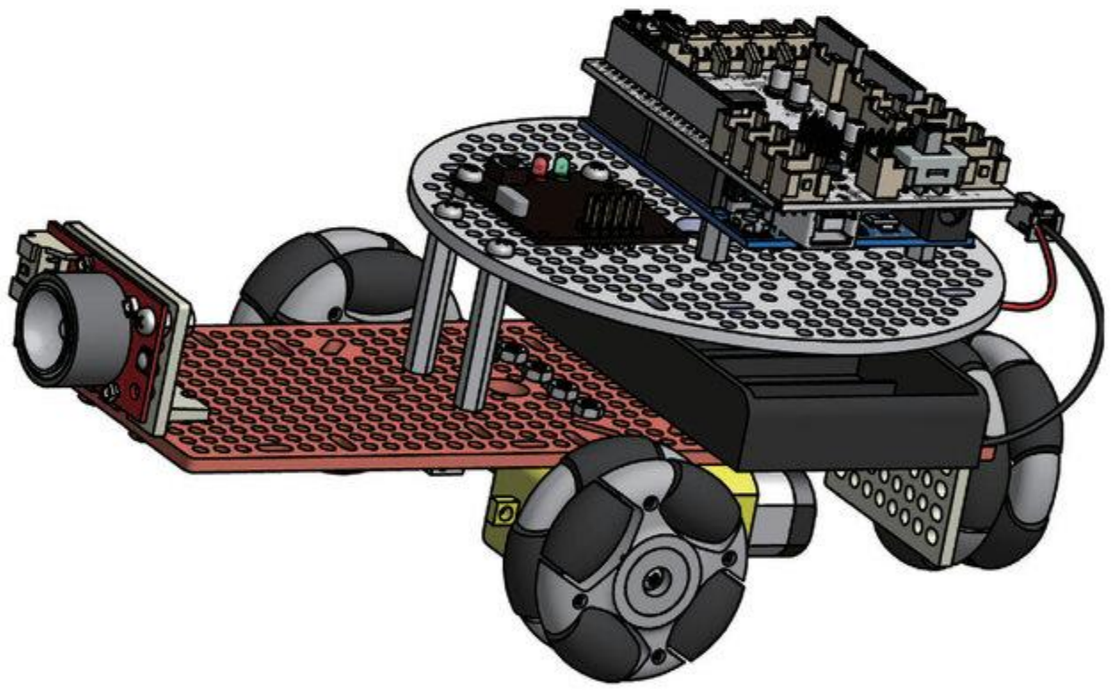


図 2-1　カラートレーサーロボットの完成イメージ（未配線）

＜組み立て手順①＞

図2-2のように、カラーセンサーをセンサー L字ステイに取り付けます。使用するパーツは、全部でM3L10ネジ（×2）とM3ナット（×4）です。

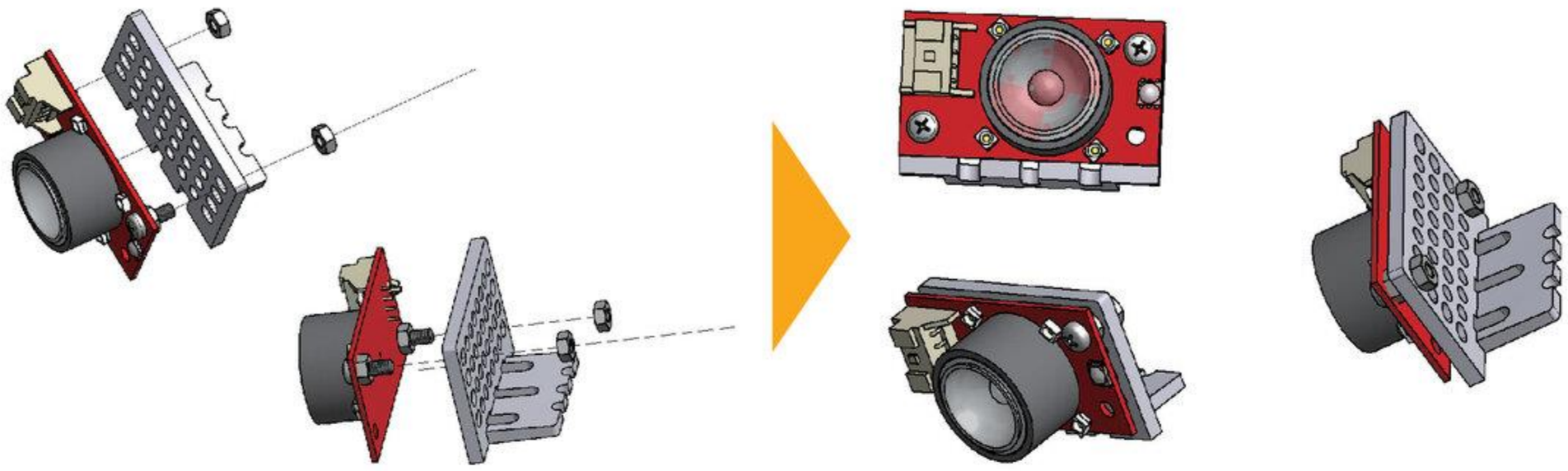


図2-2　カラーセンサーのセンサー L字ステイへの取り付け

＜組み立て手順②＞

センサー L 字ステイに取り付けたカラーセンサーを、ベースロボット本体に組み付けます。**図2-3**にしたがってM3L8ネジ（×2）とM3ナット（×2）で取り付けてください。

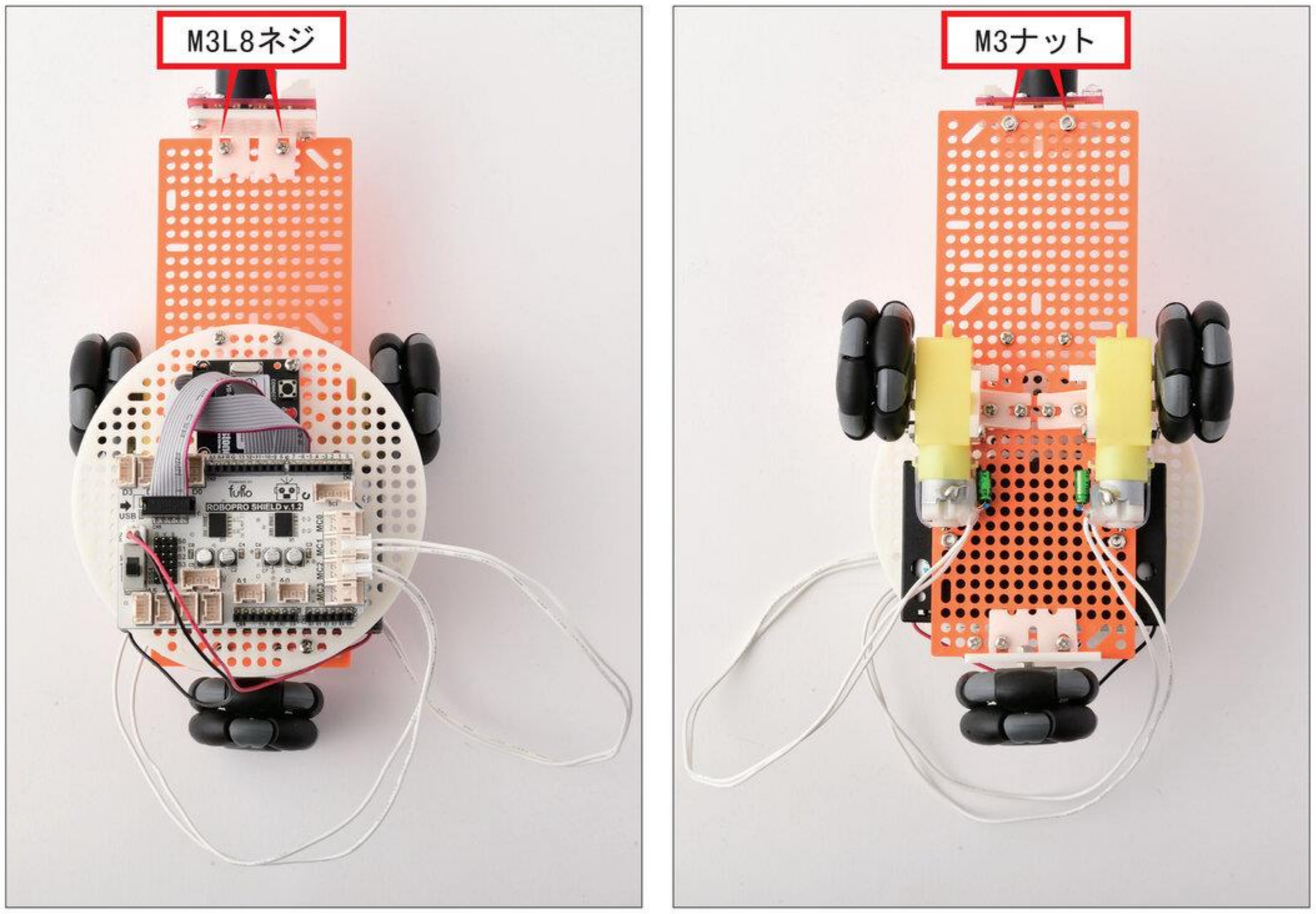


図 2-3　カラーセンサーのベースロボットへの取り付け

＜組み立て手順③＞

最後にセンサーケーブルをカラーセンサーと IIC に接続してください。これで本体は完成です。

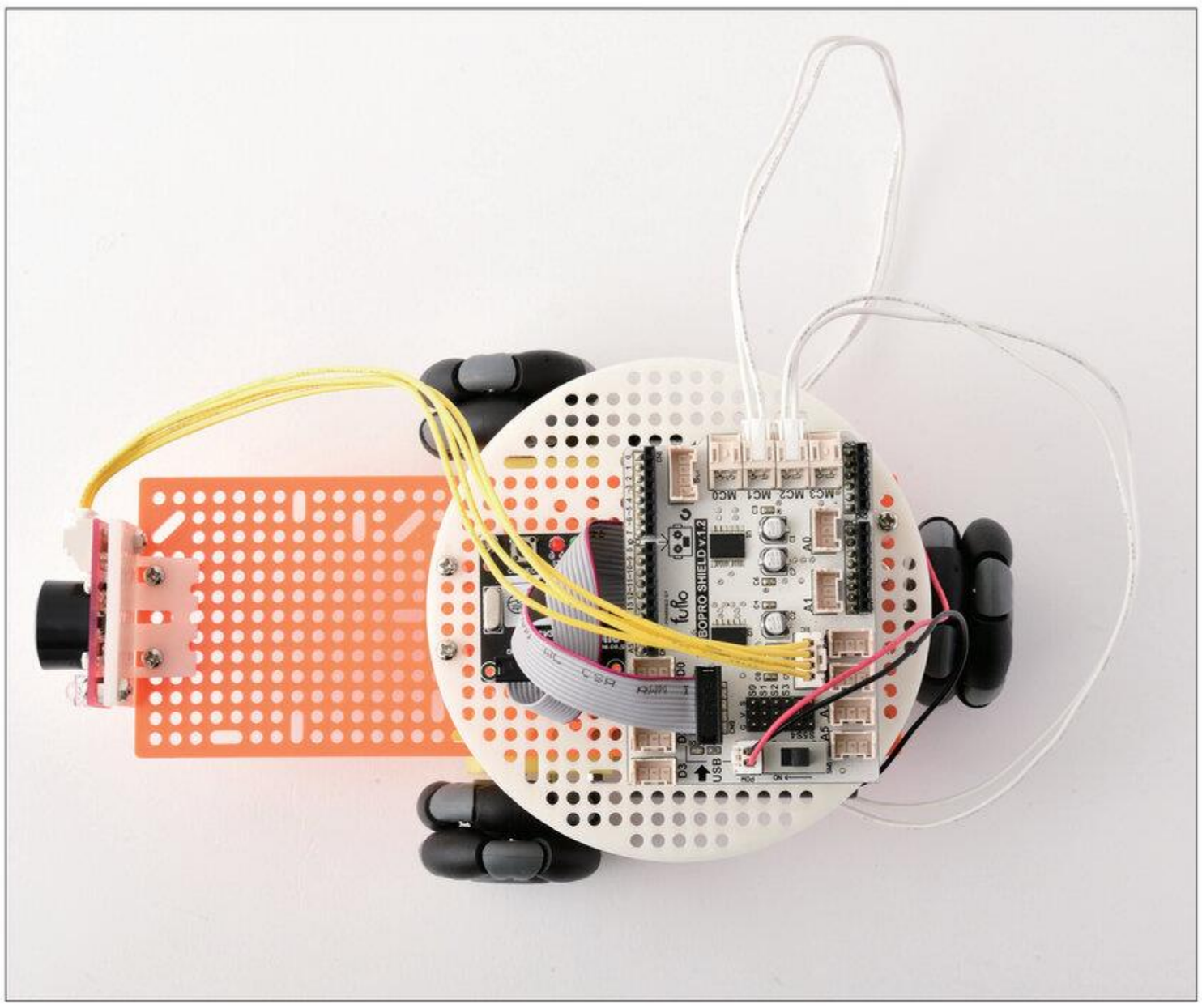


図2-4　カラートレーサーロボットの完成図

2.1. カラートレーサーロボットを動かす

それでは、カラートレーサーロボットを動かしてみましょう。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 > MagicItemB4 > ColorTracer

書き込みを終えたら、第2回の巻末の付録の色見本をロボットの前にかざしてください。どのような結果になったか、以下に記入しましょう。なるべく明るい場所で行ってください。また、センサーを天井に向けた状態では反応しないことがあります。

青の紙をかざすと、反応して前進する。

やってみよう！

1．プログラム「ColorTracer」の以下の部分を変えてみよう。

```
if(h > 180 && h < 270)
```

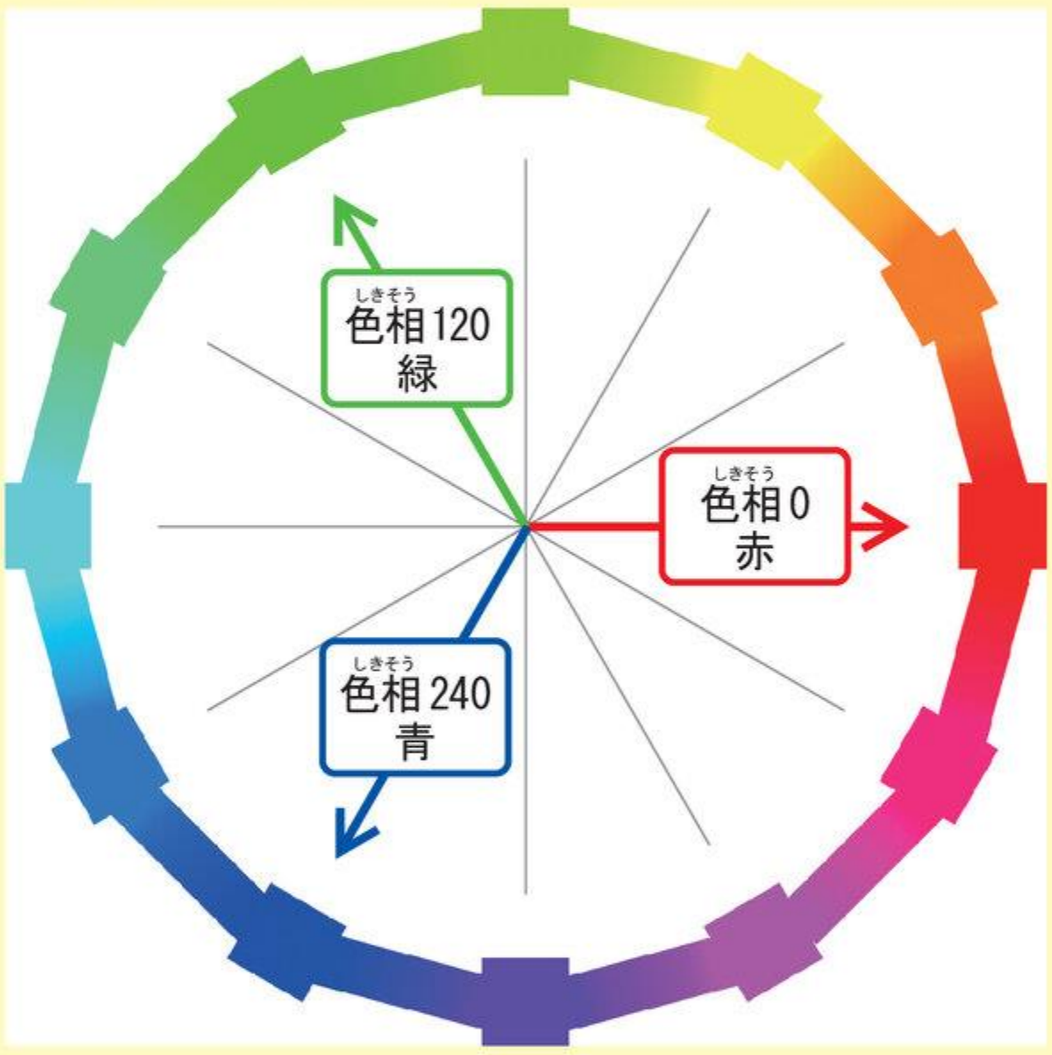
▼

```
if(h > 0 && h < 90 || h > 270 && h < 360)
```

どうなったかな？

ヒント

第2回で学んだ「色相」を覚えているかな？　変数 h は、カラーセンサーがはかった色相の値が入るようになっているよ！　つまり、変更前は「色相が180より大きく270より小さいとき」、変更後は「色相が0より大きく90より小さい、または270より大きく360より小さいとき」という条件になるよ！



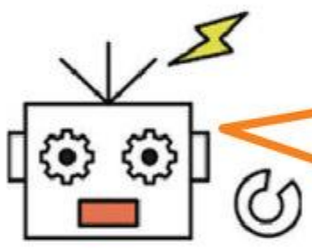
赤の紙をかざすと、反応して前進する。

2．さらにプログラムを書きかえ、反応する色を緑にかえてみよう！

講　緑の色相は 120 なので、120 を含む範囲を条件式に入れます。
やや余裕を持たせて `if(h > 90 && h < 180)` などとするとより確実に緑を検知できますが、同時に緑以外の色に誤反応する可能性も増します。周囲の環境によっても反応する色は変化するので、必要に応じ値を微調整してください。

３．まとめ（目安５分）

今回は、カラーセンサーを使って、「感じて」、「考えて」、「動く」ロボットを作ってみました。
ライントレーサーロボットは、調整次第で、とても速く動くロボットにもすることができるため、専用のコンテストがあるほどです。
ロボットは、ハードウエア（ロボット本体）とソフトウエア（プログラム）の両方を改良して、性能を上げていくことができます。どちらか片方を工夫するだけでも良いのですが、両方を調整することで、さらに精度の高いロボットへと進化できます。今後、学習が進むにつれて、さらに高度な調整方法もわかっていくので、さらにスキルアップを目指しましょう。
次回は、超音波距離（ちょうおんぱきょり）センサーを使って、同じように自動で動くロボットを製作してみましょう。



ラインという名のコードをたどり、ゴールという名の約束の地まで駆け抜けヨ！
さあ行こうか……。スピードの向こう側へ！！

講

○　以下の授業の目標を再確認します。
- ベースロボットの動きを理解し、自由に動かす
- カラーセンサーの調整をする
- ライントレーサーロボットを動かす
- カラートレーサーロボットを動かす

○　次回テーマは「ウルトラソニックロボット」であることを告知します。

<次回必要なもの>

次回は、今回使ったベースロボットと、以下のパーツを持ってきてください。
なお、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3-0　次回必要なもの

P.7　ステップアップ　解答例　円 (challenge1)

```
void loop(){
    mc1.rotate(-90);      // 90の速度で負方向に回す！ (最大255)
    mc2.rotate(50);       // 50の速度で正方向に回す！ (最大255)
}
```

P.7　ステップアップ　解答例　8の字 (challenge2)

```
void loop(){
    mc1.rotate(-100);     // 100の速度で負方向に回す！ (最大255)
    mc2.rotate(60);       // 60の速度で正方向に回す！ (最大255)
    delay(10000);
    mc1.rotate(-60);      // 60の速度で負方向に回す！ (最大255)
    mc2.rotate(100);      // 100の速度で正方向に回す！ (最大255)
    delay(10000);
}
```

P.7　ステップアップ　解答例　正方形 (challenge3)

```
void loop(){
    // まっすぐ走る
    mc1.rotate(-60);  // 60の速度で負方向に回す！ (最大255)
    mc2.rotate(60);   // 60の速度で正方向に回す！ (最大255)
    delay(5000);      // 5秒間待つ
    //直角に旋回
    mc1.rotate(-60);  // 60の速度で負方向に回す！ (最大255)
    mc2.rotate(-60);  // 60の速度で負方向に回す！ (最大255)
    delay(700);       // 0.7秒間待つ(ロボットによって要調整)
}
```