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

センサーロボット②

（第３回/第４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３回授業日（だいかいじゅぎょうび） 2024年　　月　　日

第４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11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

センサーロボット②

第3回

センサーで工夫しよう

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。

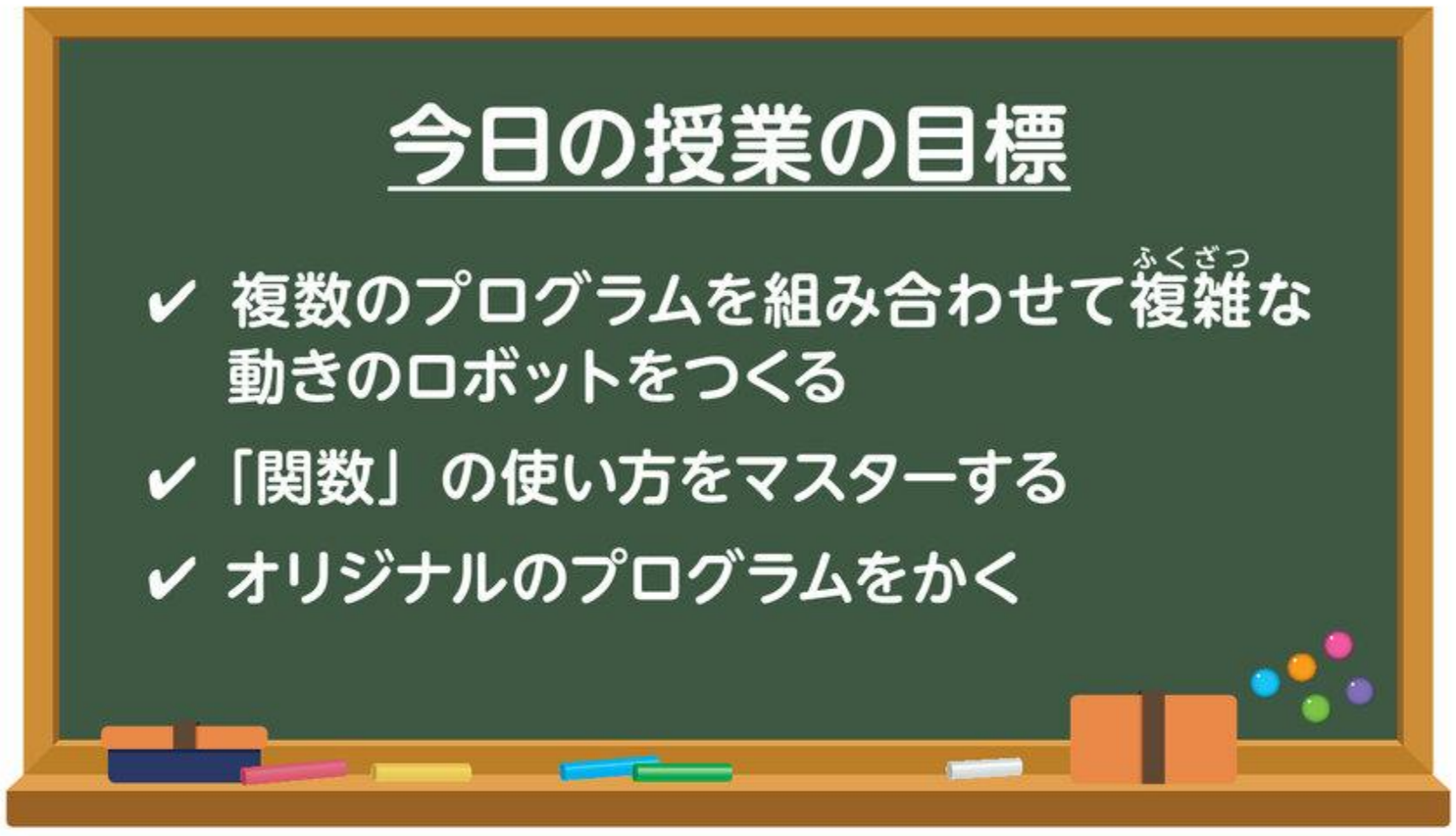
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

　（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）

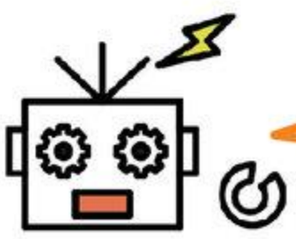
目安時間は授業時間 120 分のうち、休憩 10 分程度取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0．センサーで工夫しよう（目安 5 分）

0.0. 「センサーで工夫しよう」でやること



今回はパーツや機能をさらに追加して、よりカシコイ動きをするロボットにしましょう。
前回までは「超音波距離センサー」「カラーセンサー」を使っていましたが、さらに「スピーカー」も追加します。そして「7 セグメント LED」に関しても、新しい表示方法を取り入れて使ってみましょう。
使うパーツや機能を増やすと、その分高性能なロボットをつくることができます。たとえば前回は、モーターを利用することによってセンサーの機能を向上させました。センサーだけでは決まった方向の情報しか読み取れませんでしたが、グルグル動き回ることによって読み取れる情報を増やしました。そうすることで 360 度全方位にセンサーを設置するのと同じだけの効果が得られたわけです。
とはいえ、機能を増やそうとすると、どんどんプログラムが複雑になっていきます。そこで役に立つのが、今回新しく登場する「関数」という命令です。これを使ってプログラムを見やすく簡単にしていきますよ。
さまざまな機能が追加されると、よりたくさんの情報を人間に伝えることもできます。オリジナルロボットの構想もふくらませながら進めていきましょう！



かっこいいオリジナルロボットをつくるためにあらゆるパーツや機能を使いたおそう！

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	スピーカー 1
			

図 0-0　必要なもの

図 0-1 のようにこれから組みかえをします。超音波距離(ちょうおんぱきょり)センサーは前方に２つ固定します。カラーセンサーは上向きに固定します。

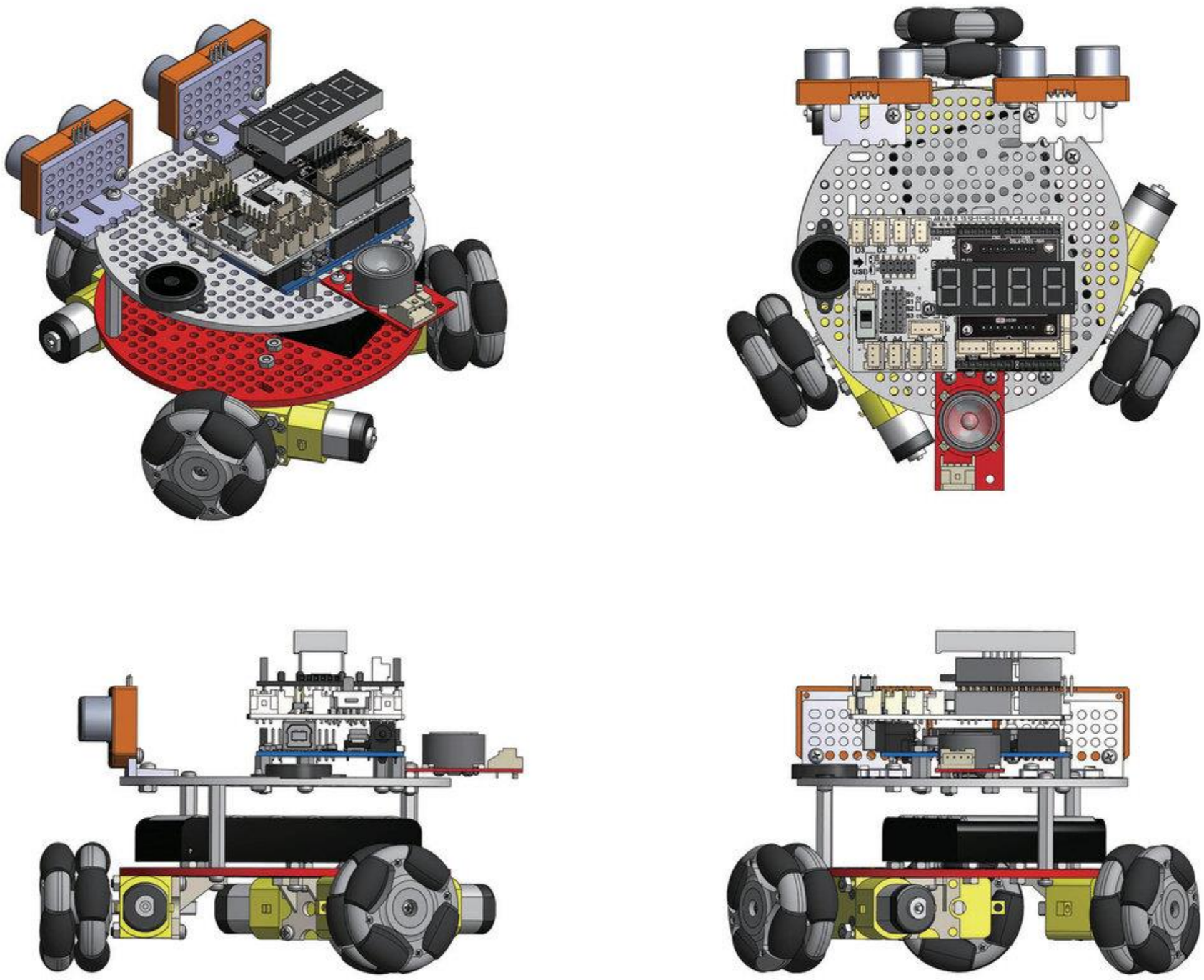


図 0-1　完成イメージ

0.2.　組みかえとセンサーの配線

各部、センサーの取り付けは以下を参考にしてください。なお、ネジ穴の位置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。また、パーツの取り付けに使用するネジも表記した長さのものでなくとも固定できれば、問題ありません。
配線が多くなるので、混線しないように注意しましょう。

1）カラーセンサーとスピーカー

カラーセンサーは、M3L10 ネジ（× 2）と M3 ナット（× 4）を使用しています。
M3L10 ネジをカラーセンサーの穴に通して、M3 ナットで固定します。そのうえで、円形ボードの穴にネジを通して、M3 ナットで固定します（ダブルナット方式）。
スピーカーは、M3L8 ネジと M3 ナットを使用して固定します。

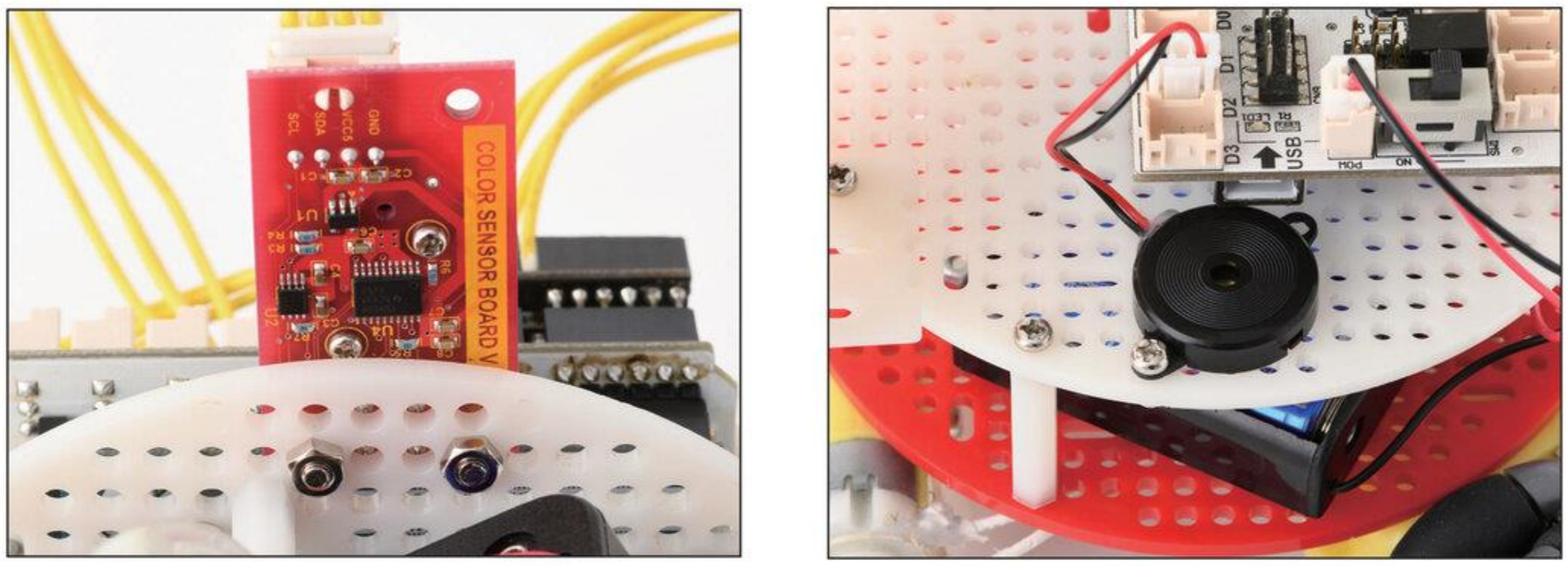


図 0-2　カラーセンサーとスピーカーの取り付け

2）超音波距離センサー

超音波距離センサーは、M3L8 ネジ（× 2）、M3 ナット（× 2）を使用しています。なお各センサーは、ネジ 1 本ずつで固定しているので、グラグラしないようにしっかりとネジを締めましょう。

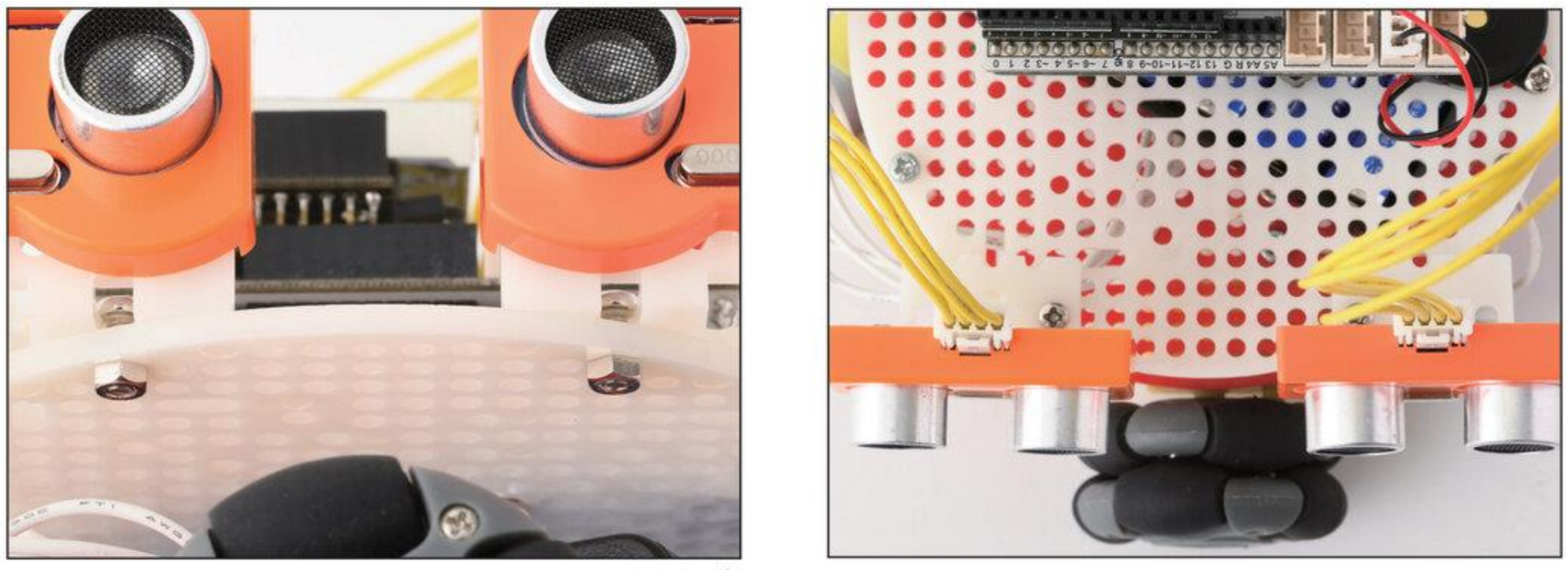


図 0-3　超音波距離センサーの取り付け

組みかえを終えたら、以下の POINT と写真を参考にして、センサーの配線をしましょう。なお、コントローラーは使用しません。ほかにも不要なものがついている場合は、電力を消費するので外しておきましょう。

POINT

- 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（ちょうおんぱきょり）センサー（右）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（左）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。

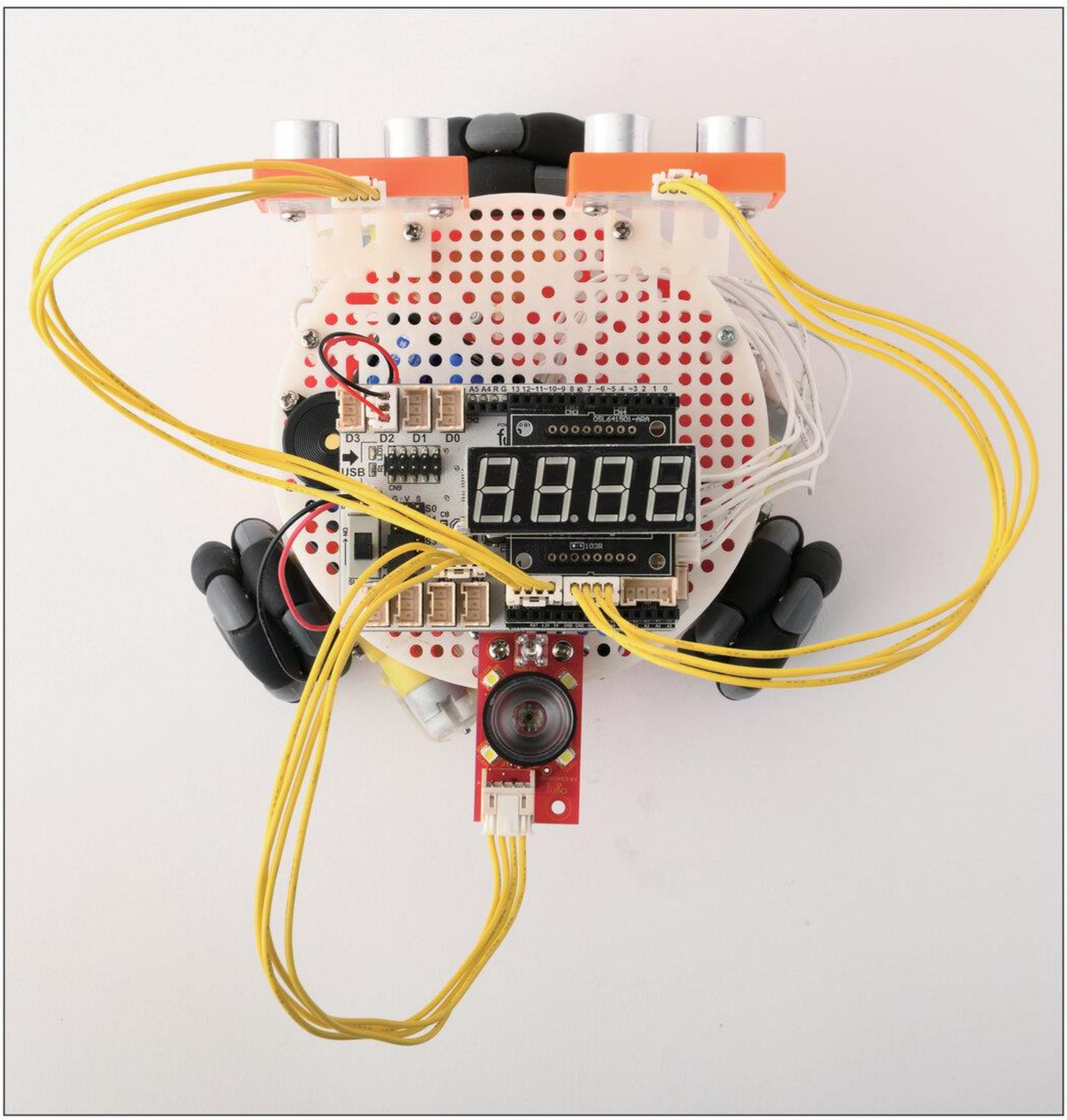


図 0-4　センサーロボット

1．暗がりが好きなロボット（目安 70 分）

1.0. 「暗がりが好きなロボット」とは

ここでは、「暗がりが好きなロボット」をつくります！
部屋の中を、障害物を避けながら動き回って、いすや机の下など影のある場所を見つけたら、その場で落ち着きます。そして、落ち着いたときには、7 セグメント LED の光をルーレットのようにグルグル回し、スピーカーから「喜びの音」を出します。また明るくなると、落ち着かないので再び動き回ります。
今回つくるロボットは、そんな動きをするちょっぴり内気でかわいらしいロボットです。

図 1-0　暗がりが好きなロボットのイメージ

まずは、このロボットに搭載したい機能を 1 つずつあげていきましょう。

POINT

① カラーセンサーで周囲の明るさを検知する。
② もし周囲が暗かったら、スピーカーから「喜びの音」を出す。
③「喜びの音」とあわせて、7 セグメント LED をルーレットのようにグルグル光らせる。
④ 周囲が暗くない場合、超音波距離センサーで障害物を検知し、避けながら動き回る。

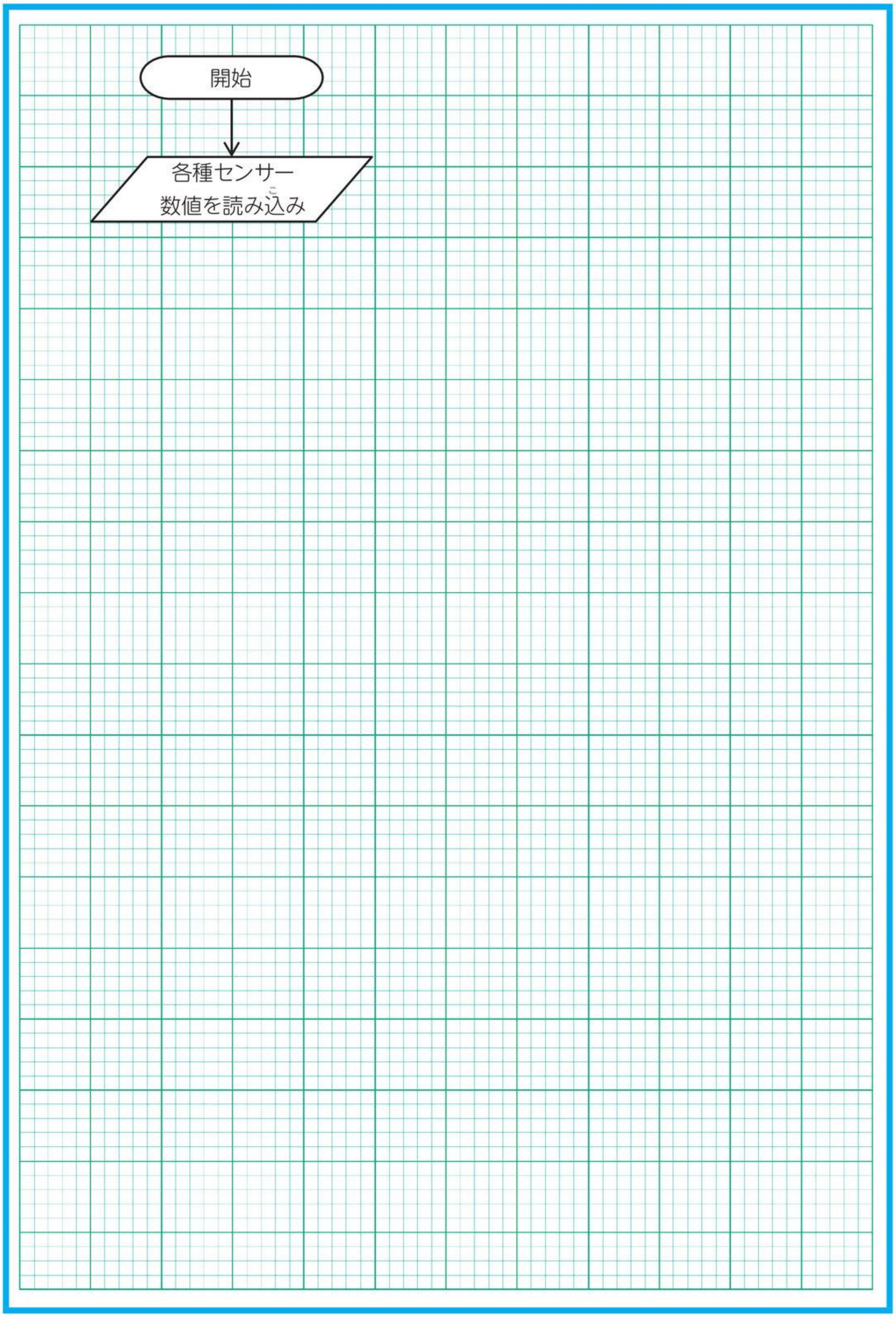
1.1. 搭載する機能をフローチャートで整理する

このようにたくさんの機能を搭載したロボットをつくるときには、前回学んだ「フローチャート」をかいてみるのが非常に効果的です。
以下の 3 種類の if をうまく組み合わせて、「暗がりが好きなロボット」のフローチャートを考えていきましょう。

図 1-1　フローチャートに使用する条件分岐ブロック

やってみよう！

図 1-1 の条件分岐をつかって、「暗がりが好きなロボット」のフローチャートをかいてみよう！
超音波距離センサー、カラーセンサーの読み取りブロックまではかいておくから、残りを完成させてみてね。


開始
各種センサー
数値を読み込み

解答例は以下の通りです。形が違っていても、同じ流れになっていれば構いません。

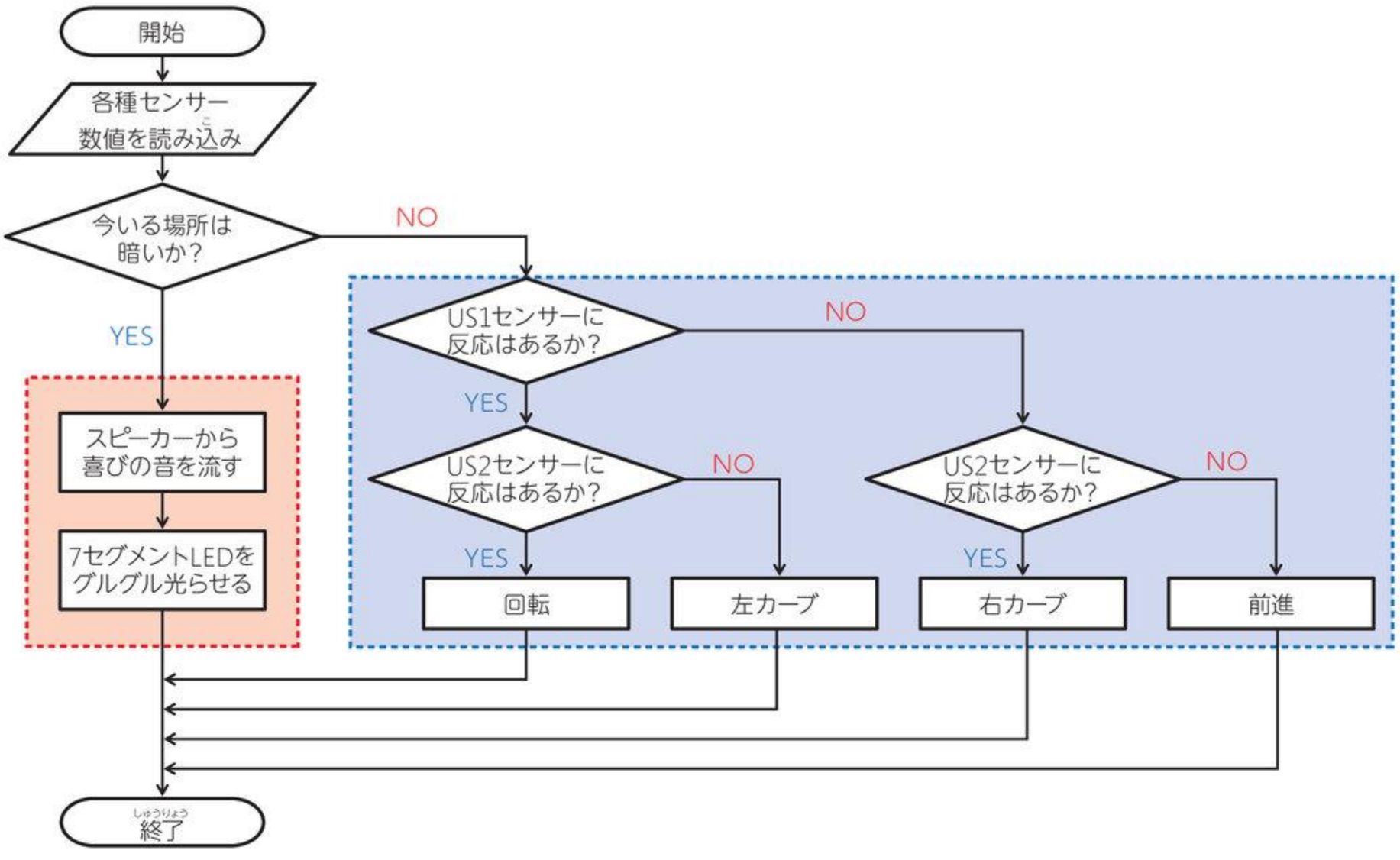


図 1-2　暗がりが好きなロボット

分岐が多く、かなり複雑なフローチャートになってしまいました。しかし、よく考えるとこのロボットの機能は大まかにいえば「暗がりで落ち着く」と「障害物を避けて動き回る」の 2 つだったはずです。それをふまえてみると、たった今かいたフローチャートも以下のようにまとめられるはずです。

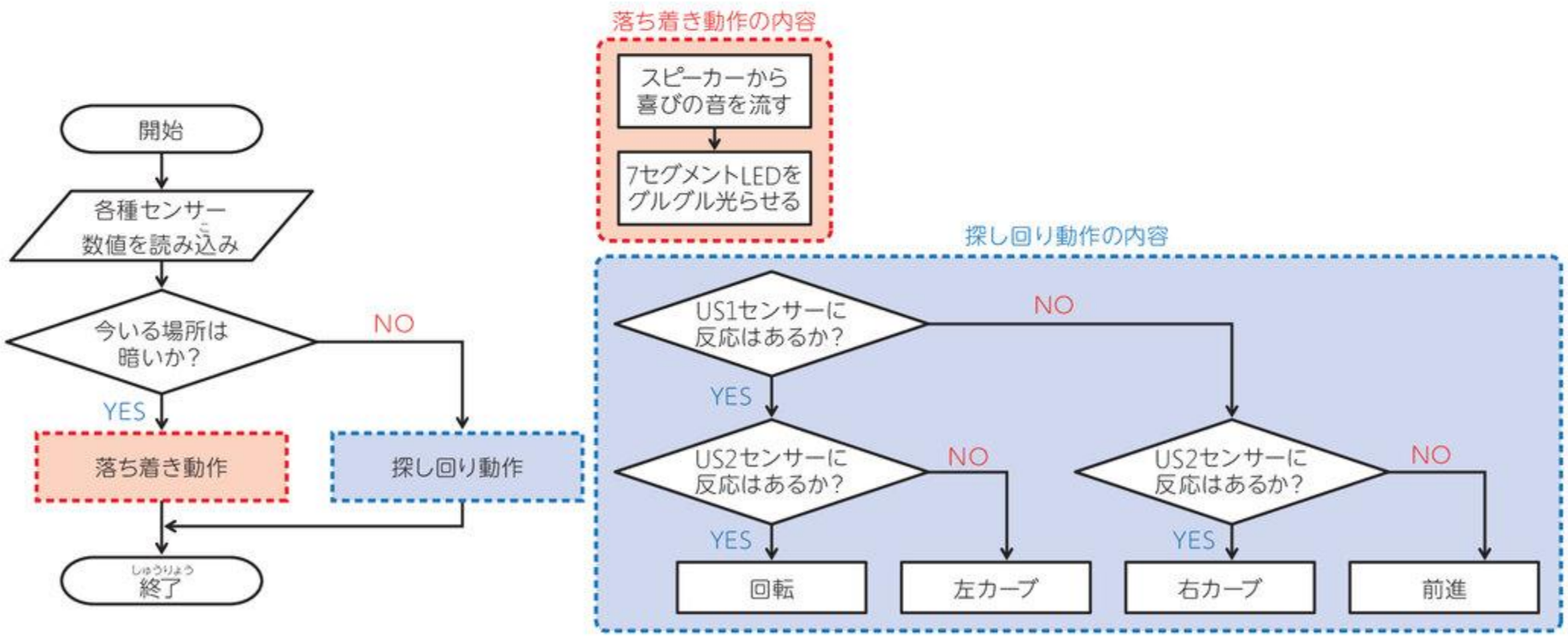


図 1-3　暗がりが好きなロボット（シンプルバージョン）

図 1-2 と**図 1-3** のちがいがわかりますか？
図 1-2 の赤枠･青枠部分を、それぞれひとくくりにしたものが**図 1-3** です。ブロックの数はかわっていませんが、**図 1-3** のほうがループ全体の流れはわかりやすくなったように感じませんか。

実は、Arduino のプログラムには**図 1-3** のように、複数の処理をまとめて１つの処理としてかくという機能があります。この機能を「関数」とよぶのですが､今回は「暗がりが好きなロボット」のプログラムをシンプルに直すことで「関数」の使い方をマスターすることを目標にします。まずは、いつも通り必要な機能を１つずつつくっていきましょう。

1.2. ①カラーセンサーで光の強さを検知する

ここでは、カラーセンサーで光の強さを検知します。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
RoboticsProfessorCourse2 > CombRobot3 > ColorClearTest

実行結果：カラーセンサーが読み取った光の強さの値が 7 セグメント LED に表示される。

この機能は、以前ライントレーサーで使いましたね。白と黒で反射する色の明るさを判定して、色を判別していました。ここでは純粋（じゅんすい）に、明るい場所と暗い場所を見分けます。
さて、この後の準備として、明るい場所と暗い場所の２つの値を取っておきましょう。ここでロボットがどのような状況を明るい（あるいは暗い）と感じているかを知るために、光の強さをはかっておくわけです。

やってみよう！

ロボットを実際に動かす場所の光の強さをはかってみよう。はかったら以下にメモをしておこう。測定するときは、測定する場所にしばらく静止させておいてね。

明るい場所の数値：

例: 机の上（蛍光灯の真下）では 3000 以上。ただし少し動くと、2000 以下になる。

暗い場所の数値：

例：机の上で蛍光灯の真下からずれると 2000 以下。物の影だと、1000 以下になる。

講　解答例の数値は参考数値です。環境や、照明の明るさで大きくかわるので、実際の数値を記入してください。

1.3. ②スピーカーから「喜びの音」を出す

まず、暗い場所で落ち着いたときに出す音のプログラムをテストします。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
RoboticsProfessorCourse2 > CombRobot3 > ToneTest

実行結果：スピーカーから音が出る。
プログラム「ToneTest」を見てみると、ライブラリ「Tone」を使って、自分でつくった楽譜（がくふ）の音を出しています。1年目の不思議アイテムの授業でも勉強しましたね。

1.4. ③７セグメントＬＥＤをルーレットのようにグルグル光らせる

次に、7 セグメント LED の動作テストをします。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
RoboticsProfessorCourse2 > CombRobot3 > _7segRoulette2

実行結果：7 セグメント LED の周りをルーレットのように光が回る。
プログラム中では、以下の部分で 7 セグメント LED を点滅（てんめつ）させています。 HIGH は点灯、LOW は消灯の命令です。配列を使ってプログラムされています。

```
for(int i = 0; i < 12; i++){          // for文、11以下の場合に値は 1 ずつ大きくなる
    lc.setLed(0, rx[i], ry[i], HIGH);
```

lc.setLed(0, a, b, HIGH); という命令は「7 セグメント LED の、a ケタ目の b 番目のセグメントを点灯させる」という意味です。

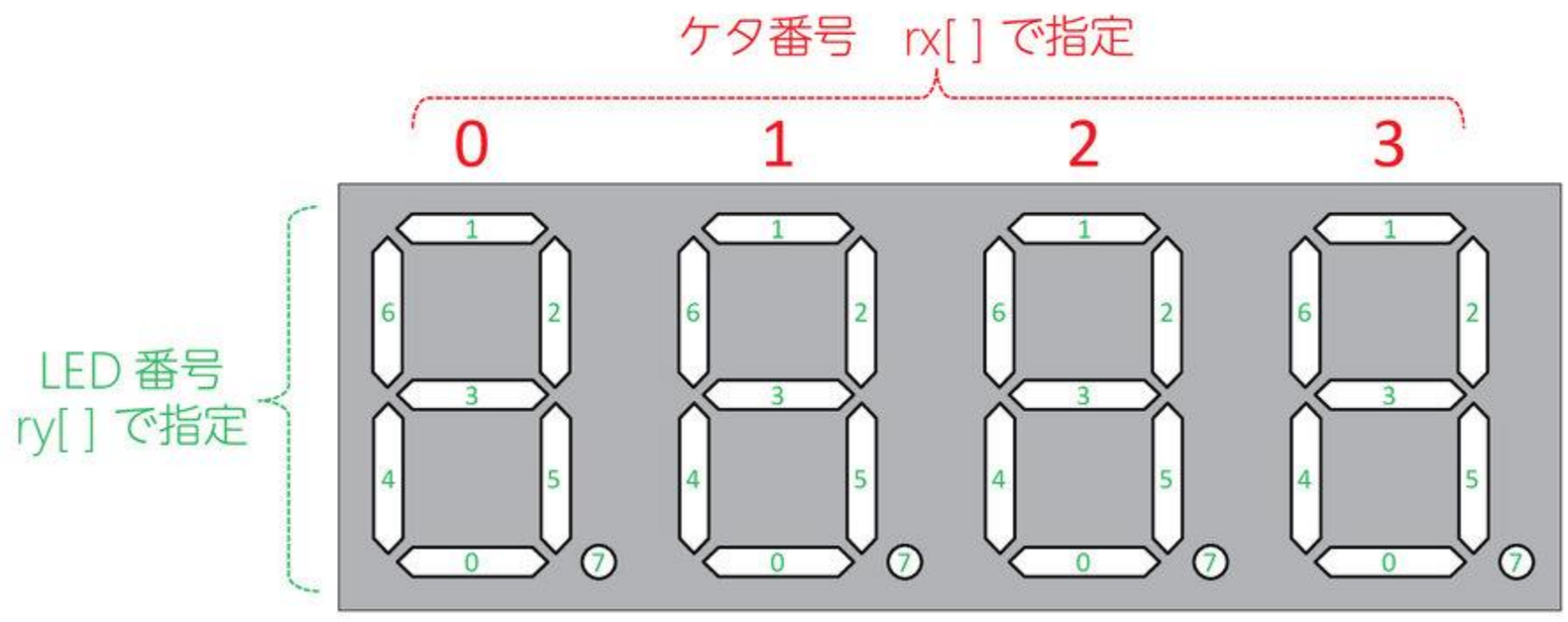


図 1-4　7 セグメント LED のケタ番号と LED 番号

for 文によって変数 i が 0 から 11 まで増加していくので、lc.setLed(0, rx[0], ry[0], HIGH); から lc.setLed(0, rx[11], ry[11], HIGH); まで、LED の点灯命令を 12 回くり返すことになります。**図 1-4** を見ると、変数 i が 0 のときは lc.setLed(0, 0, 1, HIGH); となればよいことがわかります。つまり、rx[0] は 0, ry[0] は 1 に置きかわることになります。

さらに光を移動させる場合は、**図 1-5** のように rx[1] が 1, ry[1] が 1 に置きかわればいいですね。

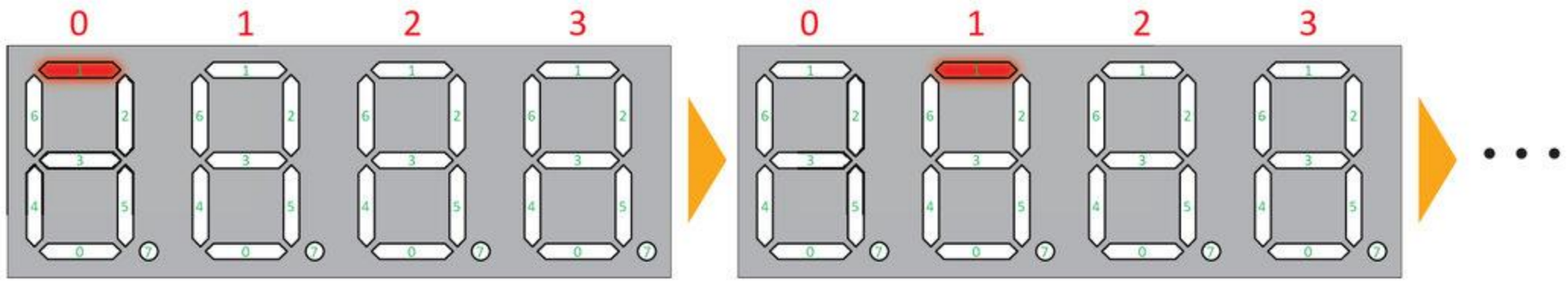


図 1-5　7 セグメント LED をグルグル光らせるパターン

やってみよう！

下の表を完成させて、7 セグメント LED の光を 1 周させるためにはどのセグメントを光らせればいいかをまとめてみよう！

	変数 i の値											
	0	1	2	3	4	5	6	7	8	9	10	11
rx[i] の値	0	1	2	3	3	3	3	2	1	0	0	0
ry[i] の値	1	1	1	1	2	5	0	0	0	0	4	6

プログラム「_7segRoulette2」を見てみると、rx[i] や ry[i] の値の指定は以下のようにかかれています。

```
int rx[]      = {0, 1, 2, 3, 3, 3, 3, 2, 1, 0, 0, 0};
int ry[]      = {1, 1, 1, 1, 2, 5, 0, 0, 0, 0, 4, 6};
```

上で完成させた表と見比べると、数字の並びが同じになっていることがよくわかりますね。rx[i] や ry[i] の値は規則的に変化していくわけではないので、かわりに 1 ずつ増加していく変数 i を使って for 文にしているわけです。ほかのテーマですでに学んでいる人もいるかもしれませんが、「配列」を利用しています。

やってみよう！

プログラム「_7segRoulette2」を変更（へんこう）して、ルーレットを逆回りにしてみよう。

講　解答例は巻末に記載します。

1.5. ①～③のプログラムを合体させる

それでは、①～③のプログラムを合体していきます。その前にプログラムの全体像を見ておきましょう。
まずは、②のプログラムです。

プログラム「ToneTest」より抜粋

```
#include <RPlib.h>
#include <Tone.h>

Tone tone1;

//曲データ1
char *sound1 = "M1A:d=4,o=5,b=120:16g,16c6,16g5,16c6,16g5,16c6";
//曲データ2
char *sound2 = "M2A:d=4,o=6,b=180:16g,16c7,16f7,16p,16c7,16d7,16g7,16p,
16g,16c7,16d7,16f7,16g7,16d7,16c7,16d7";
```

```
void setup(){
    tone1.begin(D2);              //D2にスピーカーを接続
}
```

```
void loop(){
    tone1.play_rtttl(sound1);   //sound1の曲を流す
    tone1.play_rtttl(sound2);   //sound2の曲を流す
}
```

紫色の枠内の #include で Arduino のライブラリから Tone.h という音を使えるプログラムを取り込んでいます。これにより、「sound1」と「sound2」という音楽データ（楽譜）をつくりました。
次に、水色の枠内で D2 のスピーカーを使えるように初期化 setup() をしています。
橙色の枠内のループ内で「sound1」と「sound2」の音楽データ（楽譜）を順番に実行していきます。

なお、「暗がりが好きなロボット」の音楽データは１つでいいので、今回は「sound2」を選択します。

次に、③のプログラムを見ていきましょう。

プログラム「_7segRoulette2」より抜粋

```
#include <LedControl.h>

LedControl lc = LedControl(11, 13, 1);
                                        // 7セグメントLEDを使うためのオマジナイ
```

```
void setup(){
    lc.shutdown(0, false);          // 7セグメントLEDをリセット
    lc.setIntensity(0, 8);          // 色の濃さ
    lc.clearDisplay(0);             // 表示クリア
}
```

```
void loop(){
    int rx[] = {0, 1, 2, 3, 3, 3, 3, 2, 1, 0, 0, 0};        // けた
    int ry[] = {1, 1, 1, 1, 2, 5, 0, 0, 0, 0, 4, 6};        // LED番号

    for(int i = 0; i < 12; i++){  // for文、11以下の場合に値は1ずつ大きくなる
        lc.setLed(0, rx[i], ry[i], HIGH);
        delay(100);
        lc.setLed(0, rx[i], ry[i], LOW);
    }
}
```

紫色の枠内の #include で Arduino のライブラリから LedControl.h という 7 セグメント LED を使えるプログラムを取り込みます。
水色の枠内で 7 セグメントの初期化 setup() をしています。
橙色の枠内のループ内は先ほども勉強した、LED を点滅する命令です。

やってみよう！

②と③のプログラムが整理できたところで、①に合体させていこう。
「ColorClearTest」のプログラムにかき加えてね。

ヒント
紫色、水色、橙色それぞれの枠同士で合体させよう！

講　できる生徒には一人でチャレンジさせてください。プログラムの入力が難しい場合は、できるだけ解答プログラムを読み解かせましょう。以下が解答例になります。
RoboticsProfessorCourse2 ＞ CombRobot 3 ＞ GloomyStep1

1.6.　関数を使う

1）プログラムをシンプルにする

さて、「暗がりが好きなロボット」のプログラムのうち、暗がりを見つけたときの反応は完成しました。障害物を避ける動作はまだ組み込んでいないので、下の図のフローチャートのような流れになっています。

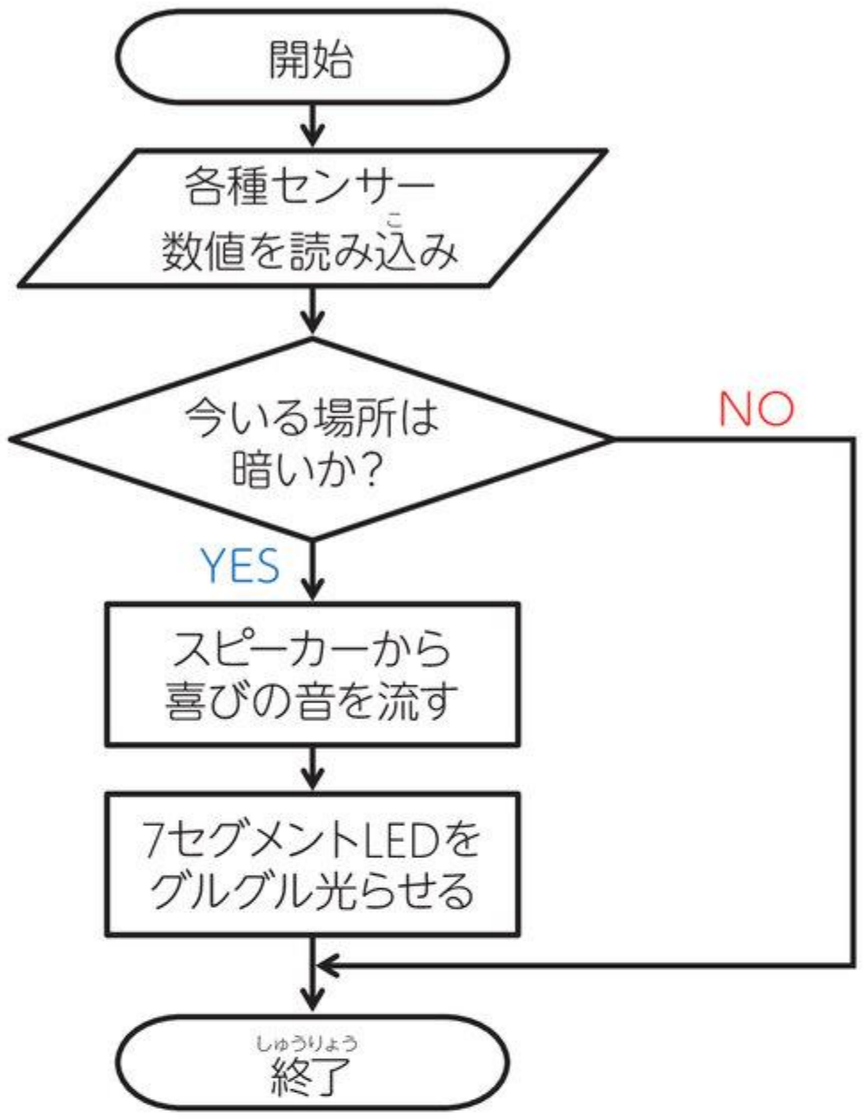


図 1-6　暗がりが好きなロボット未完成

しかし今回ははじめに説明したとおり、よりシンプルでわかりやすくするため「関数」を使ってさらにプログラムを改良していきます。
具体的には、下の図のフローチャートのようになるようプログラムをかきかえていきます。

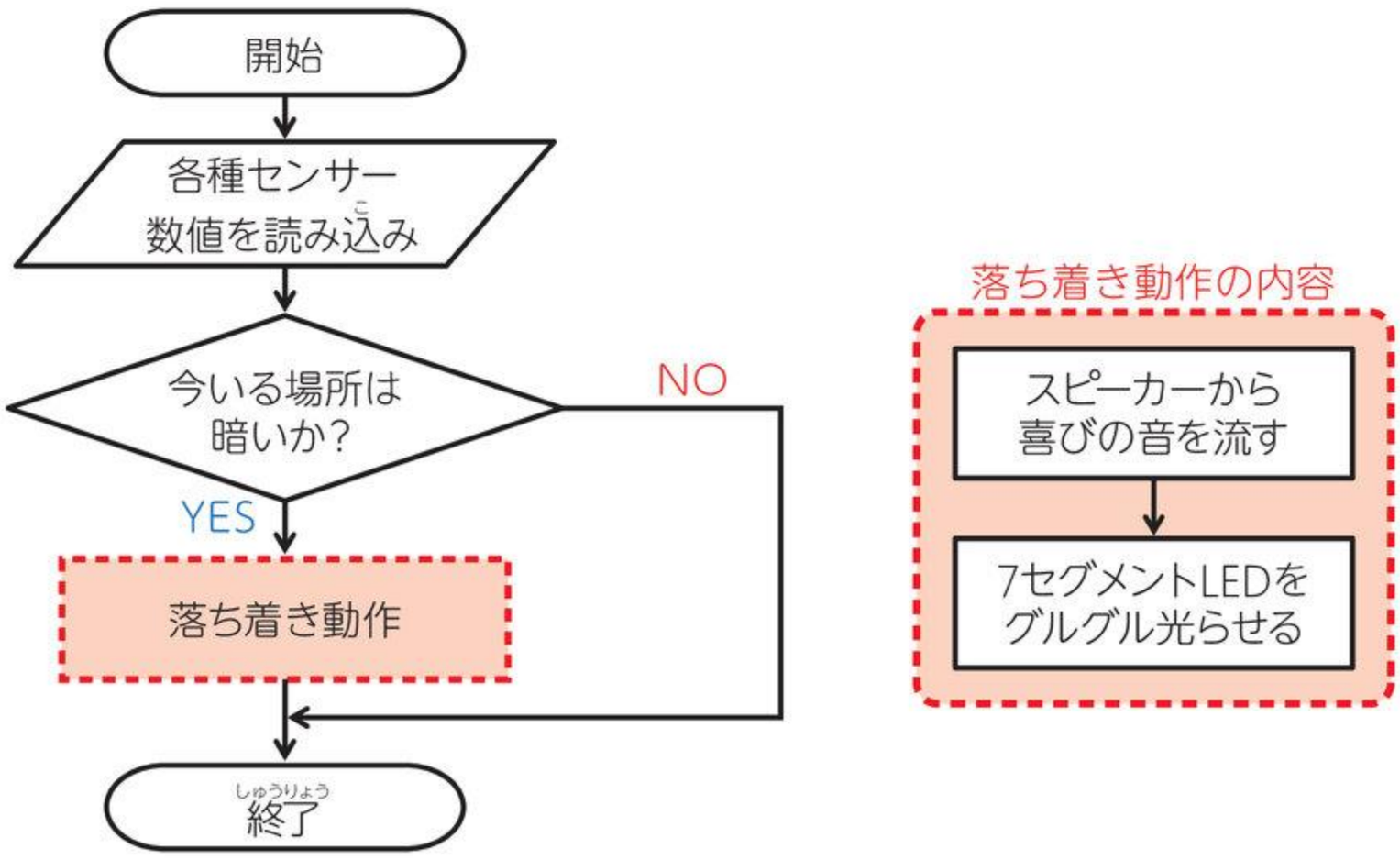


図 1-7　暗がりが好きなロボット未完成　シンプルバージョン

テキストでは解答例である「GloomyStep1」を使用しますが、先ほど皆さんが自分でつくったプログラムを使ってももちろん構いません。

やってみよう！

以下のプログラムのうち、**図 1-7** の「落ち着き動作」（赤枠の部分）にあたる部分はどこかな？　四角で囲んでみよう！

プログラム「GloomyStep1」より抜粋（ばっすい）

```
#include <ColorSensor.h>
#include <LedControl.h>
#include <RPlib.h>
#include <Tone.h>
Tone tone1;

// 曲データ2
char *sound2  = "M2A:d=4,o=6,b=180:16g,16c7,16f7,16p,16c7,16d7,16g7,
                 16p,16g,16c7,16d7,16f7,16g7,16d7,16c7,16d7";
int rx[]      = {0, 1, 2, 3, 3, 3, 3, 2, 1, 0, 0, 0};
int ry[]      = {1, 1, 1, 1, 2, 5, 0, 0, 0, 0, 4, 6};

LedControl lc = LedControl(11, 13, 1);    // ７セグメントLEDを使うため
                                             のオマジナイ


void setup(){
    tone1.begin(D2);                      // D2にスピーカーを接続

    lc.shutdown(0, false);                // ７セグメントLEDをリセット
    lc.setIntensity(0, 8);                // 色の濃さ
    lc.clearDisplay(0);                   // 表示クリア
    ColorSensor.begin(4000, 0);           // カラーセンサーの積分時間を
                                             指定 x0.1[ms]　値が大きいほ
                                             どセンサー値の精度が良くな
                                             るが、そのぶん時間がかかる
                                             6000で600[ms] 最大 6144



}

int clear;
int i;

void loop(){
    if (ColorSensor.colorRead()){
        clear = ColorSensor.colorClear(); // 輝度を取り込む
    }
    if (clear < 500){                     // 計測したデータ
        // 陰気スポット発見
        lc.clearDisplay(0);               // 表示クリア
        tone1.play_rtttl(sound2);         // sound2の曲を流す
        for (i = 0; i < 12; i++){
            lc.setLed(0, rx[i], ry[i], HIGH);
            delay(100);
            lc.setLed(0, rx[i], ry[i], LOW);
        }
    }
    delay(100);

}
```

今、四角で囲んだ部分をまとめればいいことになります。 void loop() 内を以下のようにしてみましょう。

```
void loop(){
    if (ColorSensor.colorRead()){
        clear = ColorSensor.colorClear();   // 輝度を取り込む
    }
    if (clear < 500){                       // 計測したデータ
        // 陰気スポット発見
        calmfunc();
    }
    delay(100);
}
```

落ち着き動作全体を calmfunc(); という１つの処理にまとめました。この calmfunc(); という命令はあらかじめ Arduino やサンプルプログラムに組み込まれていたものではなく、今この場でつくったオリジナルの命令です。

豆知識

「calm」は「落ち着き、安らぎ」という意味の英単語で、「func」は「function（関数）」を略したものです。

あとは、この落ち着き動作 calmfunc() が具体的にどんな動きでできているのかを、別のところで説明してあげれば完成です。
void setup() や void loop() 内ではない場所に、以下の文を追加します。

```
void calmfunc(){
    lc.clearDisplay(0);
    tone1.play_rtttl(sound2);
    for(i = 0; i < 12; i++){
        lc.setLed(0, rx[i], ry[i], HIGH);
        delay(100);
        lc.setLed(0, rx[i], ry[i], LOW);
    }
}
```

{} 内は先ほど四角で囲んだ部分と全く同じです。これで、calmfunc(); とかくだけで「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 つの動作をさせられるようになりました。

このように、複数の動作や処理をまとめて１つの命令にまとめたものをプログラミングの世界では「関数」とよぶのです。

POINT

関数のポイント

・ある決まった処理(しょり)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ループの外にかかれる。
・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。

2）関数を使うメリット

さて、先ほど「関数は、プログラム全体をスッキリとさせる」という話をしましたが、「新しいプログラムのほうが行数が増えて複雑(ふくざつ)になっているのでは？」と、思った人もいるのではないでしょうか？　確かに、全体の行数だけで比べると新しいプログラムのほうが増えていますね。しかし、メインループの中身だけで見るとどうでしょうか？

```
void loop(){
    if (ColorSensor.colorRead()){
        clear = ColorSensor.colorClear(); // 輝度を取り込む
    }
    if (clear < 500){                     // 計測したデータ
        // 陰気スポット発見
        calmfunc();
    }
    delay(100);
}
```

センサーで明るさを見て

暗かったら、落ち着きをアピール

音を鳴らして LED を光らせるという動作を、calmfunc(); という、一つの関数（命令）で表しているととらえると、メインループのプログラムの構造はとてもスッキリして見えませんか？実際のところ、関数として用意されているものは、使い方さえ理解していれば、その中身がどうかかれているかは気にせずに使えることがほとんどです。逆に言えば、使い方を示した扱(あつか)いやすい関数をたくさん用意しておくと、メインプログラムはとてもシンプルにかけるようになります。

講

前ページまでの内容をまとめたプログラムは以下となります。
RoboticsProfessorCourse2 ＞ CombRobot 3 ＞ GloomyStep2
修正が難しい生徒がいたり、時間がない場合は、上記のプログラムを開き解説を進めてください。

３）これまでに使っていた関数

実はこの関数、みなさんはこれまでに何度も使ってきていました。たとえば、rotate(); や、move(); や、digitalWrite(); などです。これらは全て「関数」です。いま挙げたものには最後に () がついていますよね！ この () が関数であることの印でした。では、これらの関数の中身は、どこにかかれているのでしょうか？

実は、これらの関数の中身はみなさんがかきかえているプログラムとは別のファイルにかかれていたのです。今まではふれずにきましたが、プログラムをよく見ると、はじめの方に #include<> といった記述を見つけることができます。これは、別のファイルに関数の中身がかいてあることを示しています（ライブラリという、プログラムファイルから取り込んでいます）。

処理のまとまりを関数として準備しておくことで、複雑なプログラムもとてもシンプルに圧縮してかくことができます。そうすることで、プログラムをとてもわかりやすくすることができるのです！

４）関数の便利な使い方の例

ここで、関数の便利な使い方の例を紹介します。

それは、何度もくり返し使う処理のまとまりを一つの関数にしておく方法です。処理を一つの関数にしておけば、何度も同じ処理内容をかく必要がなくなり、一行で実行できるようになります。「同じ内容のくり返しならプログラムをコピー & ペーストすればいいや。」と思う人もいるかもしれませんが、とてもわかりにくいプログラムになるので、プログラマーとしてはさけた方がよい方法です。むしろ関数は、「for 文」などと組み合わせると、更にスッキリとしたプログラムがかけます。たとえば、「5 回落ち着く」プログラムは次のようになります。

```
// 5回落ち着く
for(i = 0; i < 5; i++){
    calmfunc();
}
```

プログラムを見やすくするということは、とても大切です。複雑なプログラムであればあるほど、全体を把握しやすくかいておくことが重要なのです。特に、プログラミングにおいて大半の時間を占めるデバッグ作業（プログラムがうまく動かなかったとき、問題のある箇所を探し、修正する作業）の効率が段ちがいに良くなります。

1.7.　プログラムの完成

さて、最初の目的に戻って、「暗がりが好きなロボット」をつくりましょう。

POINT

① カラーセンサーで周囲の明るさを検知する。
② もし周囲が暗かったら、スピーカーから「喜びの音」を出す。
③「喜びの音」とあわせて、7 セグメント LED をルーレットのようにグルグル光らせる。
④ 周囲が暗くない場合、超音波距離センサーで障害物を検知し、避けながら動き回る。

①～③は先ほどつくりました。
あとは、④を合体するだけですね。

ステップアップ

先ほど「落ち着き動作」を関数化した自作プログラム、またはプログラム「GloomyStep2」をかきかえ、「暗がりが好きなロボット」を完成させよう！　「探し回り動作」は関数を使ってまとめてみてね。

ヒント

超音波距離センサーの数値を読み取り、値に応じて if で分岐すればいいよね。
超音波距離センサーの数値を読み取るには、あらかじめ変数 d1 と d2 を宣言したうえで void loop() 内に以下の文を追加しよう。
こうすると、変数 d1 と d2 に、US1 と US2 それぞれのセンサーの値が入るようになるよ。あとは、**図 1-3** を参考にプログラムをつくっていこう！

```
d1 = ussRead(US1);
d2 = ussRead(US2);
```

講　解答プログラムは以下となります。
RoboticsProfessorCourse2 ＞ CombRobot3 ＞ GloomyStep4
解答例は巻末にも記載します。

ここからはオリジナルロボットをつくってみましょう。

チャレンジ課題

プログラムを「かき直して」オリジナルロボットにしてみよう！
以下のプログラムをアレンジしてみよう。アレンジの例をいくつか紹介するね。

プログラムの書き込み

RoboticsProfessorCourse2 > CombRobot 3 > GloomyStep4

ヒント

以下の { } 内の値を変更して、7 セグメント LED の動きをかえてみよう！

```
int rx[]={0, 1, 2, 3, 3, 3, 3, 2, 1, 0, 0, 0};
int ry[]={1, 1, 1, 1, 2, 5, 0, 0, 0, 0, 4, 6};
```

ヒント

以下の値を大きくして、少し暗くなるだけで、止まるようにしてみよう！

```
if(clear < 500)
```

ヒント

以下の値を変更してロボットの動きをかえてみよう。
緑色のラインは超音波距離センサーの値、黄色のラインはオムニホイールの動きだよ。

```
    if(d1 < 15 && d2 < 15){
        omniBot.move(0, 0, 20);      // 回転
        delay(3000);
    }
    else if(d1 < 15){
        omniBot.move(0, 20, -20);    // 左旋回
        delay(500);
    }
    else if(d2 < 15){
        omniBot.move(0, 20, 20);     // 右旋回
        delay(500);
    }
    else{
        omniBot.move(0, 20, 0);      // 前進
    }
```

講　今回のテキストの復習をしながら、アレンジのヒントを見つけさせましょう。

2. 好きな色を探し回るロボット（目安 30 分）

2.0. 「好きな色を探し回るロボット」とは

続いてのロボットは、「暗がりが好きなロボット」のセンサーの取り付け位置とプログラムを少し変更（へんこう）してつくります。部屋の中をウロウロしながら、プログラムで指定した色の紙の上で停止し、LED が光って音が出ます。LED の光り方は先ほどとは変更（へんこう）してみましょう。

POINT

床（机）に置いた色紙が読み取れるように、カラーセンサーを取り外し、下向きに取り付け直します。なおネジ位置や使用するネジのサイズなどは多少アレンジしても構いません。

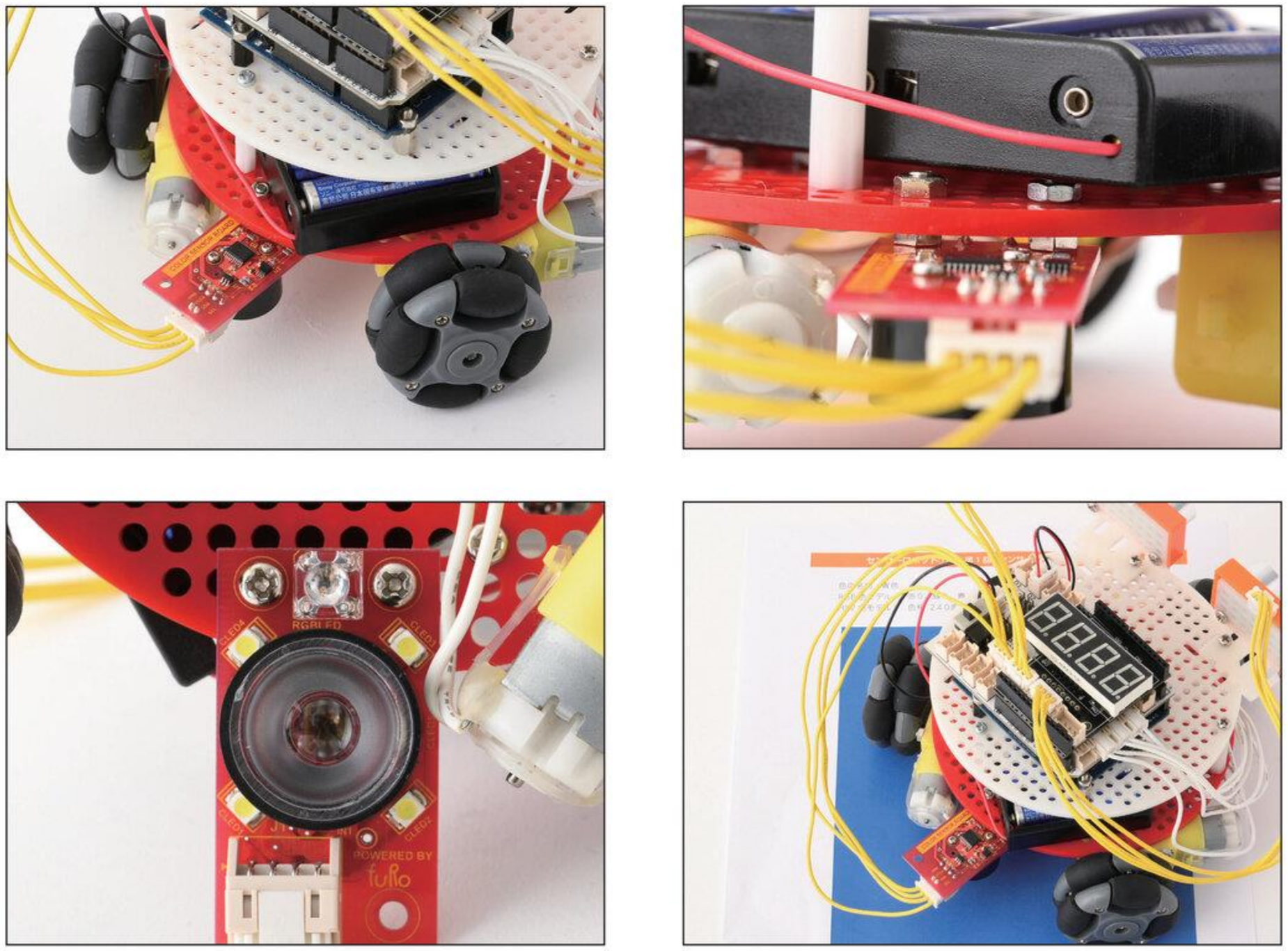


図 2-0　カラーセンサーの取り付け直し

2.1. 停止させる色の色相をはかる

続いて、以下のプログラムを使って、停止させる色の色相をはかります。ここで計測したデータをもとに先ほどのプログラムの色相の値の部分をかきかえれば、「好きな色を探し回るロボット」の完成です。第１回の色見本を使い、**図 2-1** のようにはかります。
では、以下のプログラムを実行して、ロボットを色見本の上に置いてみましょう。

プログラムの書き込み
RoboticsProfessorCourse2 > CombRobot3 > ColorSensorTest

実行結果：カラーセンサーが読み込んだ色相が数値として７セグメント LED に表示される。

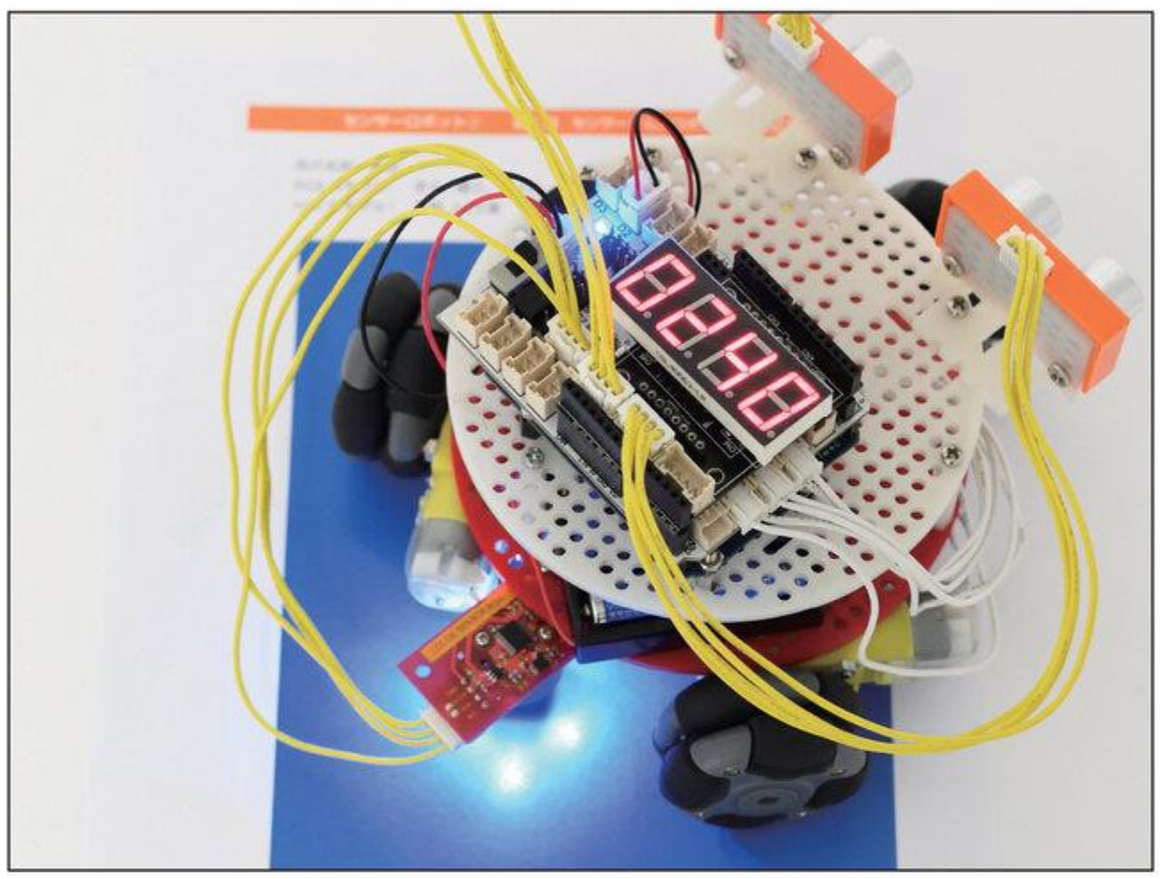


図 2-1　プログラム「ColorSensorTest」実行時の様子

2.2. 「好きな色を探し回るロボット」をつくる

部屋の中をウロウロしながら、プログラムで指定した色の紙の上で停止し、LED が光って音が出るロボットにします。
まずは、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
RoboticsProfessorCourse2 > CombRobot3 > CalmColor

実行結果：超音波距離センサーで障害物を検知して避けて進み、カラーセンサーが緑色を検知すると止まる。

やってみよう！

プログラム「ColorSensorTest」で計測した値を検出するように改造してみよう。プログラム「CalmColor」の下記の値を変更してね。

```
if(h >= 100 && h <= 140){       // もし色相が緑なら
```

ヒント

たとえば青（値 240）を読み取りたいのであれば、だいたい h > 220 & & h < 260 くらいで範囲を設定してみよう。ただし部屋の明るさなどによって数値は変動するから調整してみてね。
また、プログラム「ColorSensorTest」であらかじめ床の数値を測定しておいて、その床の色と差のある色（数値）を指定すると成功しやすくなるよ。

コラム　HSV 方式

色を「色相（Hue）」と「彩度（Saturation）」と「明度（Value・Brightness）」で表現する方式を「HSV 方式」といいます。
以前 RGB という表現方式も学びましたが、違いは表現する要素ですね。RGB は各色（Red、Green、Blue）の割合で色を表現していましたが、HSV では色相などで表現します。
今回のプログラムでは以下のように RGB を HSV（h、word s、word v）に置きかえています。

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

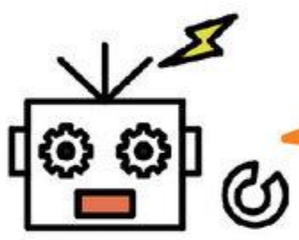
なぜわざわざ置きかえているのでしょうか？ RGB 方式で色を表現する場合は 3 色の値を入力する必要があります。しかし、HSV では以下のように「色相（Hue）」の値を入力するだけで色を指定できるため、今回の場合は便利なのです。

```
if(h >= 100 && h <= 140)
```

なお、「彩度（Saturation）」と「明度（Value・Brightness）」も word s、word v と置きかえていましたが、今回は「色相（Hue）」の値だけで色を指定できるので、プログラム内では使用していません。

3. まとめ（目安５分）

さまざまなアイテムを自由に使うことができるようになってきましたか？ センサーを工夫して使うと、いろいろなロボットの動きを再現することが可能になりましたね。
次回は、ロボットをいかに正確に動かすかがテーマです。その方法について勉強していきます。



次回もセンサーを自由自在に活用しよう！

≪次回必要なもの≫

次回は、今回使ったセンサーオムニロボットと以下のパーツを持ってきてください。
次回は迷路から脱出するロボットに挑戦します。迷路の壁（かべ）のかわりになる物（発泡スチロール製のブロックや本など）も準備してください。

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 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。
・複数のプログラムを組み合わせて複雑な動きのロボットをつくる
・「関数」の使い方をマスターする
・オリジナルのプログラムをかく

○　次回のテーマは「迷路から脱出しよう」であることを告知します。

P11　やってみよう！　解答例

```
void loop(){
    int rx[] = {0, 0, 0, 1, 2, 3, 3, 3, 3, 2, 1, 0};
    int ry[] = {6, 4, 0, 0, 0, 0, 5, 2, 1, 1, 1, 1};

    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);
    }
}
```

P.19　ステップアップ　解答例（GloomyStep4）

```
#include <RPomniDirect.h>
#include <ColorSensor.h>
#include <LedControl.h>
#include <RPlib.h>
#include <Tone.h>

Tone tone1;
RPomniDirect omniBot(1.0f, 1.0f, 1.0f, 0.0f, MC3, MC1, MC2);
                                                    // 以前の調整データを使ってね

// 曲データ2
char *sound2  = "M2A:d=4,o=6,b=180:16g,16c7,16f7,16p,16c7,16d7,16g7,16p,1
                 6g,16c7,16d7,16f7,16g7,16d7,16c7,16d7";
int rx[]      = {0, 1, 2, 3, 3, 3, 3, 2, 1, 0, 0, 0};
int ry[]      = {1, 1, 1, 1, 2, 5, 0, 0, 0, 0, 4, 6};
LedControl lc = LedControl(11, 13, 1);  // 7セグメントLEDを使うためのオマジ
                                           ナイ

void setup(){
    tone1.begin(D2);                    // D2にスピーカーを接続

    lc.shutdown(0, false);              //7セグメントLEDをリセット
    lc.setIntensity(0, 8);              // 色の濃さ
    lc.clearDisplay(0);                 // 表示クリア

    ColorSensor.begin(4000, 0);
                                        // カラーセンサーの積分時間を指定
                                          x0.1[ms]　値が大きいほどセンサー値
                                          の精度が良くなるが、そのぶん時間が
                                          かかる　6000で600[ms] 最大 6144
}

int d1 = 0;
int d2 = 0;
int clear;
```

```
void loop(){
    d1 = ussRead(US1);                          // 前センサー検出
    d2 = ussRead(US2);                          // 後ろセンサー検出

    if (ColorSensor.colorRead()){
        clear = ColorSensor.colorClear(); // 輝度を取り込む
        lc.setDec(0, clear);                    // 取り込んだ輝度をLEDに表示する
    }
    if(clear < 500){                            // もしclearが500未満(暗い場所)なら
        // 陰気スポット発見
        omniBot.move(0, 0, 0);                  // 停止
        calmfunc();                             // 自作のオリジナル命令(関数)
    }
    else{
        avoider();
    }

    delay(100);
}

//--------------------------------
//------------------ オリジナル関数
void calmfunc(){                                // 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);
    }
}

void avoider(){
    if(d1 < 15 && d2 < 15){
        omniBot.move(0, 0, 20);                 // 回転
        delay(3000);
    }
    else if(d1 < 15){
        omniBot.move(0, 20, -20);               // 左旋回
        delay(500);
    }
    else if(d2 < 15){
        omniBot.move(0, 20, 20);                // 右旋回
        delay(500);
    }
    else{
        omniBot.move(0, 20, 0);                 // 前進
    }
}
```