ヒューマンアカデミーロボット教室

第3回　ロボプロ全国大会

「ライントレース部門競技ガイド」

2019 年 8 月 29 日

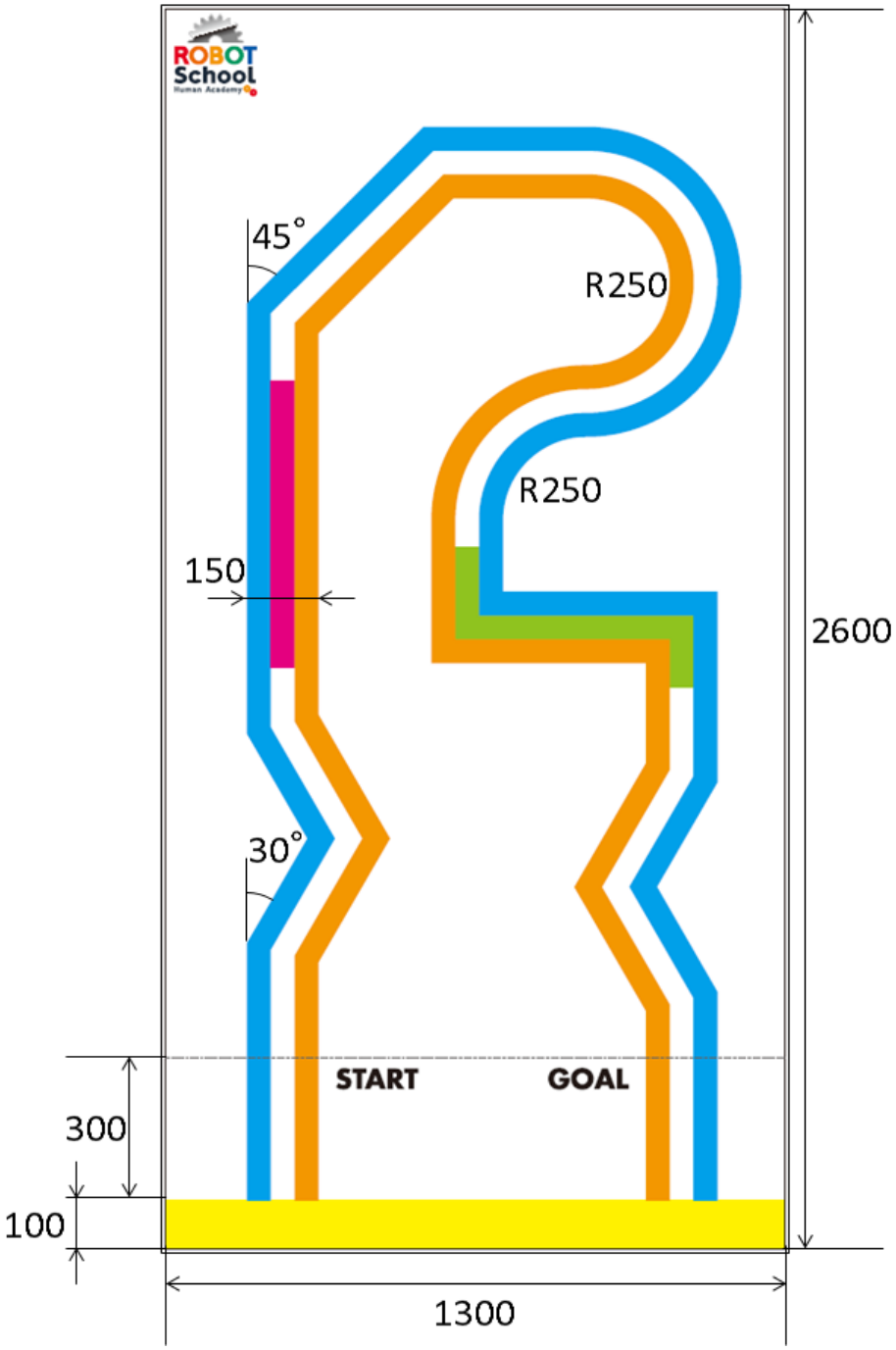
ヒューマンアカデミー株式会社

ロボット教室本部

『ライントレース部門 』競技ガイド

1. ライントレース部門昨年度コース

【サイズ】2600㎜×1300㎜
【ライン幅】150㎜（単色 50㎜）
【マットカラー】ホワイト
【ラインカラー】オレンジ、グリーン、マゼンタ、ブルー、イエロー（計 5 色）



※単位の記載がないものはすべて ㎜ です。
※上記のコースは昨年度のものとなります。

2. 競技ルール

コース内に描かれたライン上を自律走行させて、ゴールまでのスピードを競います。
大会本番では、昨年度のコースを改変したものを出題します。大会当日は 10：15 頃に入場後、本番用コースで 40 分ほど走行・調整時間が与えられます。
大会当日は、予選（タイムトライアル）の後に決勝トーナメントが行われます。予選でのタイム（完走できなかった場合は走行距離等）に応じて決勝トーナメントの組み合わせを決め、勝ち抜き戦により 1～3 位を決定します。
決勝トーナメントでは一回戦ごとに 3 分程度の調整時間を確保する予定です。その他、細かなルールは以下に記載します。

■ルール■
① 予選で使用したロボットを本選までに改良することができます。
② レース毎の乾電池の交換は自由とします。
③ ロボットのサイズは直径 30cm の円の中におさまるように設計してください。接地部分以外のパーツやケーブル類も含みます。
④ スタート時はロボットの先端がスタートラインより後方になるように設置します。
⑤ スタートの操作はロボプロシールドのスイッチまたはタッチセンサーを使用してください。
⑥ ロボットの動作開始後にロボットに触れることはできません。
⑦ ライン上から完全に脱線したり、コース上で停止したりした場合、失敗とみなし終了となります。
⑧ ゴール判定はロボット全体がゴールエリアに入った時点になります。ゴールエリア以外の場所に落下物（ねじ、ナット類を含む）があった場合、ロボット全体がゴールエリアに入ったと見なされません。
⑨ その組の全てのロボットが脱線・停止などでゴール条件を満たさない場合は、再度競技を行います。
ただし、再試合でも決着がつかない場合は、より遠くまで走行した方、もしくは審査員による判定とします。
⑩ 大会開催中に第三者の助言や手助けを受けることは禁止とします。
⑪ 別紙「禁止規定」に抵触するものを使用した場合は失格となります。
⑫ 大会当日は、実際に競技を行うコースを開放し、試合前にスタッフ管理の元で、練習走行ができます。

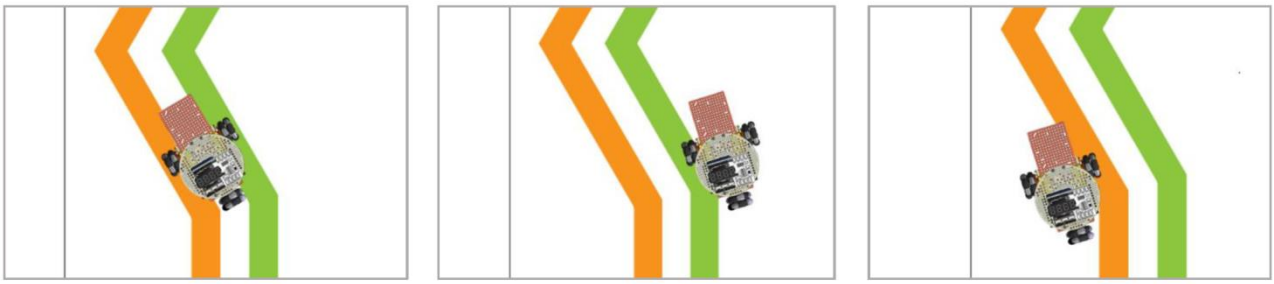
■全国大会本選ルール
① ロボットの動作開始後にロボットに触れる、またはコントローラー等で遠隔操作をすることはできません。
② 動作開始の操作をタッチセンサー、コントローラー等で行うことは可とします。直接電源スイッチでスタート操作をしても構いません。なお、スタート時にロボット自体に力をかけるような行為は禁止です。
③ コースは前回同様、オレンジ、グリーン、マゼンタ、ブルー、イエローが配置されています。
④ 検出するラインの色や検出後の分岐は任意で決めることができます。すべての色を検出しなくても構いません。

■コースの改変ポイント■
① ラインの色の種類は昨年度と同色とし、オレンジ、グリーン、マゼンタ、ブルー、イエローの計 5 色となります。
今年度はコースシートの周囲を囲むようにイエローの直線を配置しますので、落下防止の緊急停止分岐を組み込むことを推奨します。なお、イエローは前述以外の部分に配置されることはありません。
② イエロー以外の色の組合せ（配置）は変更します。
③ ラインは直線と曲線で構成されます。幅は 1 色あたり 50mm で固定です。
④ 直線の交わるところは、なす角が 90 度未満になる事はありません。
⑤ 最小曲線半径は昨年度と同様 250mm とします。

3. ライントレースの定義

ライントレースは「ロボットのホイール（または脚）がラインに触れた状態で走行していること」と定めます。ロボット全体がラインから離れた場合、脱線と見なし失格になります。ただし、例外として、ラインから瞬間的に脱線して走行し、再び脱線した位置に復帰した場合のみ競技続行が認められます。

OK ライントレースしている状態＝ロボットがライン上に触れた状態



NG ライントレースしていない状態＝ロボットが脱線している状態

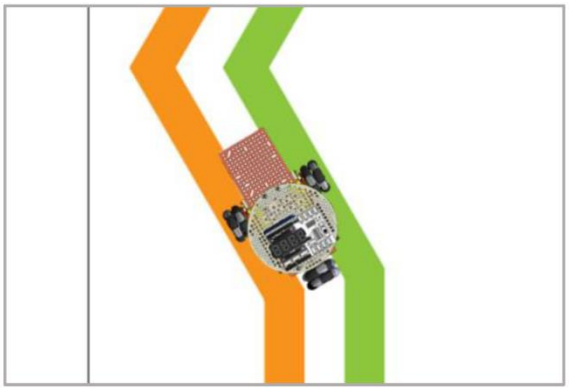
4. ライントレース攻略法

使用するロボットは製作規定の範囲内で自由に選択することができますが、カラーセンサーの使用は必須です。また、ロボットを走行させる処理手順に制限はありません。
ここでは、処理手順とプログラムの一例をあげて説明します。

１）処理手順例
ロボットは、カラーセンサーでラインの色相を読み取り、色相の値によってロボットを制御します。

※本番用コースは当日に発表されるため、以下の説明部分と本番のコースの色の配置は異なります。

【POINT1】
例えば、下図のようなコース（オレンジ、ホワイト、グリーンの三色の組み合わせ）があったとして、線の中間を走行したい場合は、次のような処理手順が考えられます。
・オレンジの線上では、右カーブをする
・グリーンの線上では、左カーブをする
・それ以外は、直進する



【POINT2】
コースのところどころに、中間にコースの両脇のラインとは別の色が入っている部分があります。
たとえば下図のようになっている場合、次のような処理手順が考えられます。
・ブルーの線上では、加速する
・マゼンタの線上では、減速する

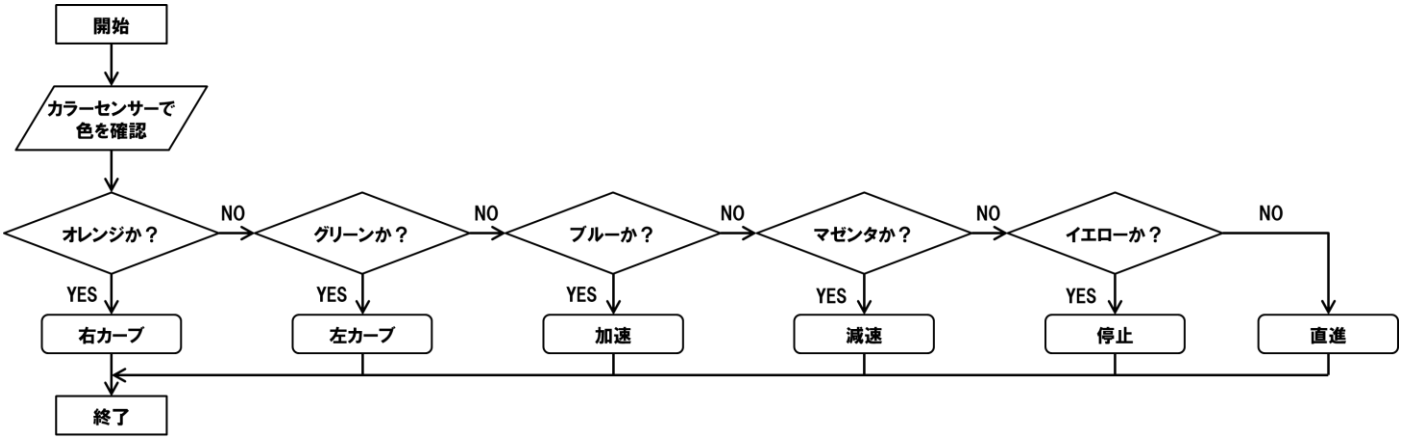
なお、直線部分と曲がり角部分に同じ色を配置することはありません。

【POINT3】

ゴール判定はロボット全体がゴールエリアに入った時点になりますので停止は必須ではありませんが、ステージからの転落等を防ぐ場合には、次のような処理手順が考えられます。

・イエローの線上では、停止する

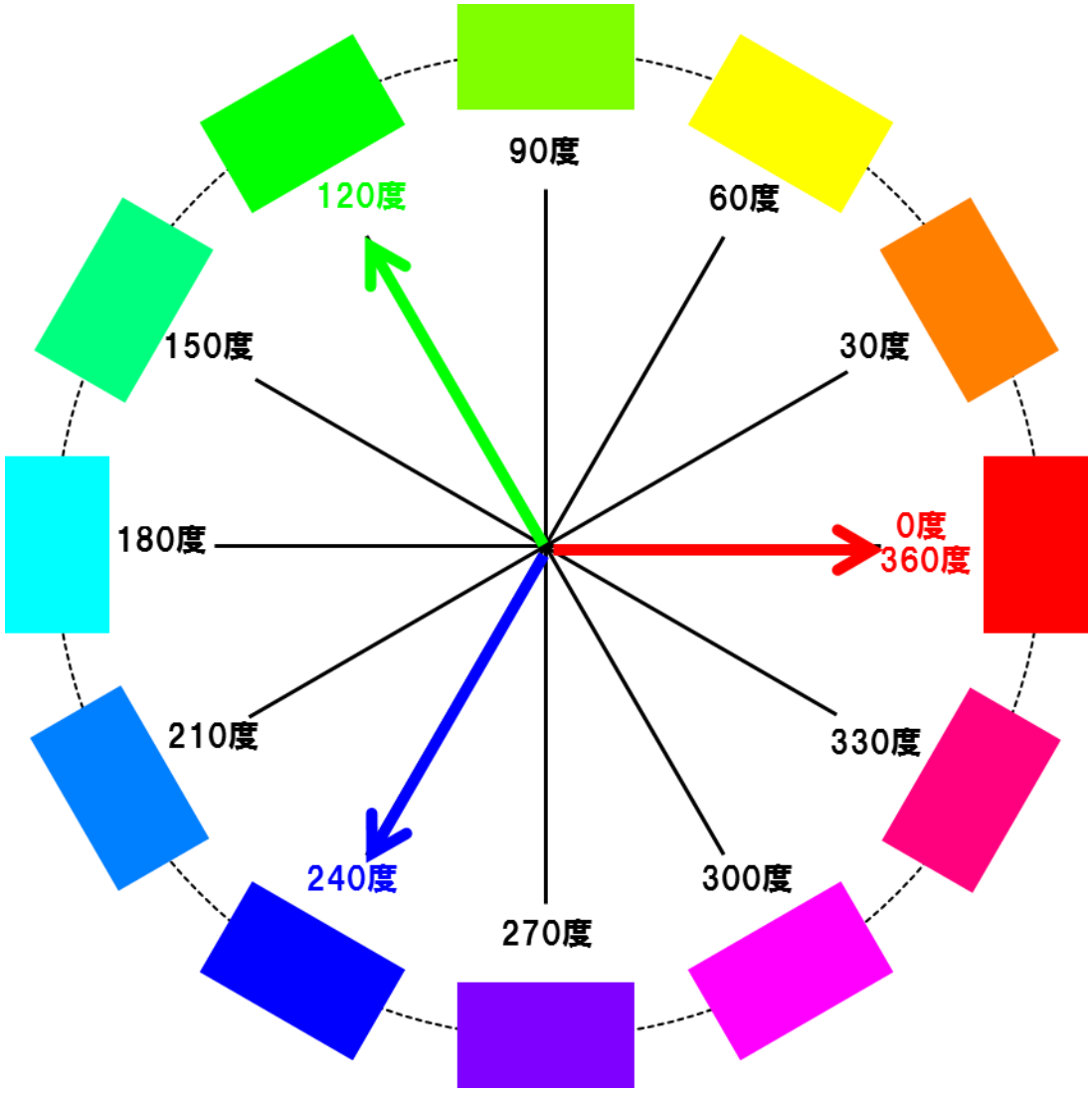
POINT1～3 をフローチャートにまとめると、以下のような流れでプログラムを作成すればよいことがわかります。



ただし上図はあくまで動作の一例です。参加者の方針や創意工夫によって、さまざまな処理が考えられます。
各場面での動作が上図と異なっていたり、反応する色数が上図より少なくても問題ありません。

２）プログラム例

ロボットに色検知をさせるためには、「色相環」という考え方を用います。
色の三原色である赤、緑、青の 3 色をベースとして、色を円形に配置して角度で表せるようにしたものです。
赤を 0 度（または 360 度）、緑を 120 度、青を 240 度の場所に配置し、原色を混ぜてさらに 9 色を追加し全 12 色とすると下図のような色相環になります。たとえば赤と緑を同量ずつ混ぜると黄色になるため、黄色は 0 度と 120 度のちょうど中間に配置されています。



「もしオレンジなら右カーブする」というプログラムを作成する場合、色相環を用いると色を数値で表すことができるため、「もし色相が 30 なら右カーブする」と書けばよいことになります。
ただしカラーセンサーの読みとる値は若干上下するので、多少のずれを許容するため「もし色相が 20 以上かつ 40 以下なら右カーブする」などとして、前後に多少の「遊び」を設けるとよいでしょう。

色相の読み取りは、以下のプログラムを使用してください。

RoboticsProfessorCourse3 ＞ InfraRed6 ＞ ColorSensorTest

プログラムを実行し、カラーセンサーを色にかざすと色相が表示されます。なお、マトリクス LED には一の位の値は表示されません。(240 度の場合は「24」と表示されます。プログラムに書く際には 10 倍した値を用いてください)
※マイコンボード、ロボプロシールド、マトリクス LED シールド、マトリクス LED、カラーセンサーを使用します。
※カラーセンサーは、センサーケーブルでロボプロシールドの IIC に接続します。

```
#include <ColorSensor.h>
#include <Sprite.h>        // Sprite before Matrix
#include <Matrix.h>
#include <RPomniDirect.h>
#include <RPlib.h>

void setup(){
    ColorSensor.begin(1000);
    myMatrix.clear();        //マトリクスLEDの表示をクリアする
    pinMode(D1,INPUT_PULLUP);
    // 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)
}

void loop(){
    int h = getHue();        //色相値を変数に入れる
    myMatrix.putd2(0,0,h / 10);        //LEDに色相/10を表示する
}

//-----------------------------
// 色相を返す
//-----------------------------
int getHue(){
    ColorSensor.colorRead();                //色センサーの読み込み
    word clear = ColorSensor.colorClear();  //色センサーの数値初期化

    word red   = ColorSensor.colorRed();    //色センサーの赤要素検出
    word green = ColorSensor.colorGreen();  //色センサーの緑要素検出
    word blue  = ColorSensor.colorBlue();   //色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();
    return h;
}
```

カラーセンサーは部屋の明るさや床の材質などに影響を受けるため、教室や自宅での練習時と大会当日の会場では、同じ色を用いても読みとる値が大きく異なる場合があります。必ず当日に色相をはかり、値にあわせてその場でプログラムを変更できるよう練習しておきましょう。

色相によって条件分岐するプログラムは、以下のプログラムをベースにすることを推奨いたします。

RoboticsProfessorCourse1 ＞ MagicItemB4 ＞ ColorTracer

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

//おまじない(ピン接続設定)
RPmotor mc1(MC1);               //MC1につながっているモーターを指定する
RPmotor mc2(MC2);               //MC2につながっているモーターを指定する

void setup(){
    ColorSensor.begin(1000, 0); // カラーセンサーの積分時間を指定 x10[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
    ColorSensor.led(3);
}


void loop(){
    if (ColorSensor.colorRead()){
        word clear = ColorSensor.colorClear();

        word red = ColorSensor.colorRed();
        word green = ColorSensor.colorGreen();
        word blue = ColorSensor.colorBlue();

        ColorSensor.convertHSV(red, green, blue);
        word h = ColorSensor.convertH();
        word s = ColorSensor.convertS();
        word v = ColorSensor.convertV();

        if(h > 180 && h < 270){         //カラーセンサーに反応したら前進して近づく
            mc1.rotate(-50);
            mc2.rotate(50);
        }
        else{                           //カラーセンサーに反応しなかったら何もしない
            mc1.rotate(0);
            mc2.rotate(0);
        }
    }
    delay(10);
}
```

特に変更が必要な部分は以下の３つです。

① `ColorSensor.led(3);`
カラーセンサー基板に内蔵されている LED のうち、3 番のものを点灯させるという処理です。
自前の照明で地面を照らすことになるので、検知の際に周囲の照度に影響されにくくなります。
`ColorSensor.led(4);`～`ColorSensor.led(7);`までを書き足すことで、複数の LED を点灯させることもできます。
明るすぎると地面からの反射でかえって検知が上手くいかないこともあるので、適宜調整してください。

なお、前述の色相計測用プログラムとこのライントレース用プログラムで、同じ LED が点灯するよう調整する必要があります。

② `if(h > 180 && h < 270)`
変数 `h` にはカラーセンサーで計測した色相が入ります。よってこの場合は「もし色相が 180 より大きく、かつ 270 より小さいならば」という条件分岐を表しています（ `A && B` と書くことで『A かつ B の場合』という条件式になります）。
計測した色相にあわせ、この値を書きかえることになります。

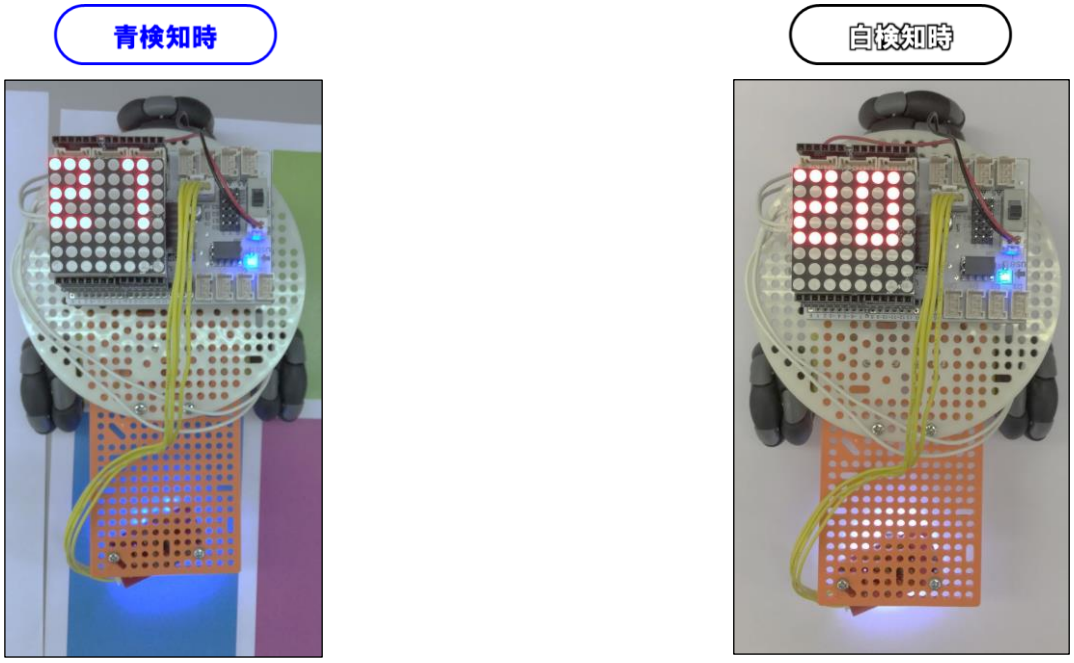
また、複数の色で分岐するため、`else if(h > ●● ～` という条件分岐を書き足していく必要があります。

③ `mc1.rotate(-50);`
`mc2.rotate(50);`
各色を検知したときの動作に関する命令文です。この場合 `MC1` モーターを逆回転、`MC2` モーターを順回転させています（このプログラムを使用する授業では `MC1` モーターは `MC2` モーターと逆向きに取り付けられているため、結果としてロボットは前進します）。

上記プログラムの①～③部以外を変更しても構いませんし、より安全性や走行性の高いプログラムを１から自作しても構いません。ただし、当日の状況に応じて自由にプログラムを書きかえられるよう、自分のプログラムがどのようなつくりになっているか、どこをどう変更するとどのような動作になるのかを、なるべく把握しておくとよいでしょう。

３）カラーセンサーで「白」を見分ける

「白」という色は、本来色相を割り当てることができません。しかし、カラーセンサーは白色にも色相を無理やり割り当ててしまいます。結果として、コースに使われる色と白の色相が似通ってしまい、うまく分岐できなくなることがあります。

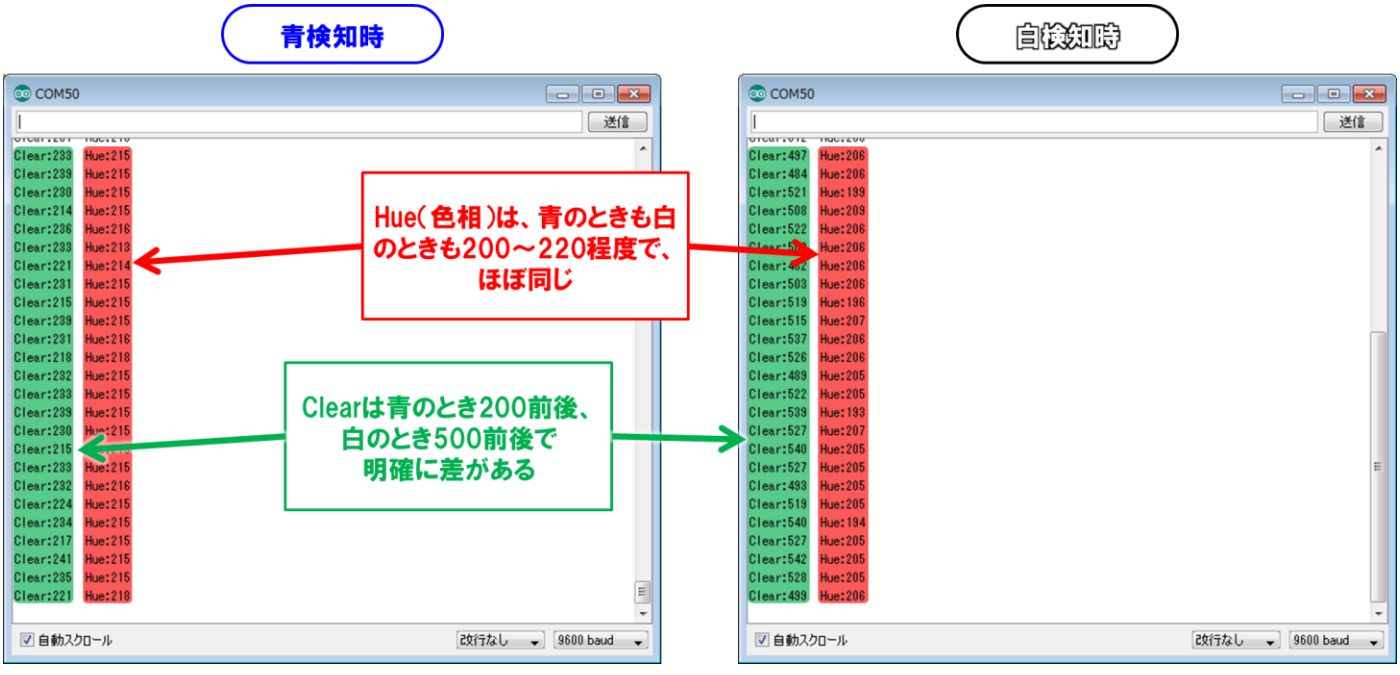


このような場合、色相以外の要素も使って条件分岐をするとうまくいくかもしれません。

プログラム「ColorCheck」を書きこみ、シリアルモニタ（画面右上の  ボタンで起動）を使うと、カラーセンサーが読みとった Hue（色相）と、Clear 値が表示されます。

Clear 値はカラーセンサーが読みとった色が白のときほど大きくなる傾向にある数値です。

色相が似通っているとしても、青と白とでは Clear 値が大きく異なるため、if 文に Clear 値の条件式を追加することでこの 2 色を区別できるようになります。なお、ライントレース参考用プログラム内では clear という変数に Clear 値が入るようになっています。



４）カラーセンサーの反応速度を上げる

「色検知は一応できているが、ロボットの反応が遅い」という場合には、以下の部分を変更してみると改善することがあります。

```
ColorSensor.begin(1000, 0); // カラーセンサーの積分時間を指定 x10[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
```

ColorSensor.begin(a,b);は、a の値でセンサーの１回あたりの検知にかける時間を指定できます。初期状態では 1000(100 ミリ秒)となっていますが、この値を増加させると時間を犠牲により正確な検知を、減少させると正確さを犠牲により高速な検知を行うようになります。a の値は 24～6144 の範囲で調整できます。

また、高速検知を行った場合、感度が足りずどの色も読み取り値に差が出ない、という問題が出やすくなりますが、b の値を 0～3 の範囲で増加させていくことである程度の改善が見込めます。a を小さくしたらあわせて b は大きくする、という方針をお勧めします。あとは、自分のロボットや作戦に合った a, b の値の組み合わせを探してみてください。

なお、こちらも色相チェック用プログラムと走行用プログラムとで、同じ値を使うようにしましょう。