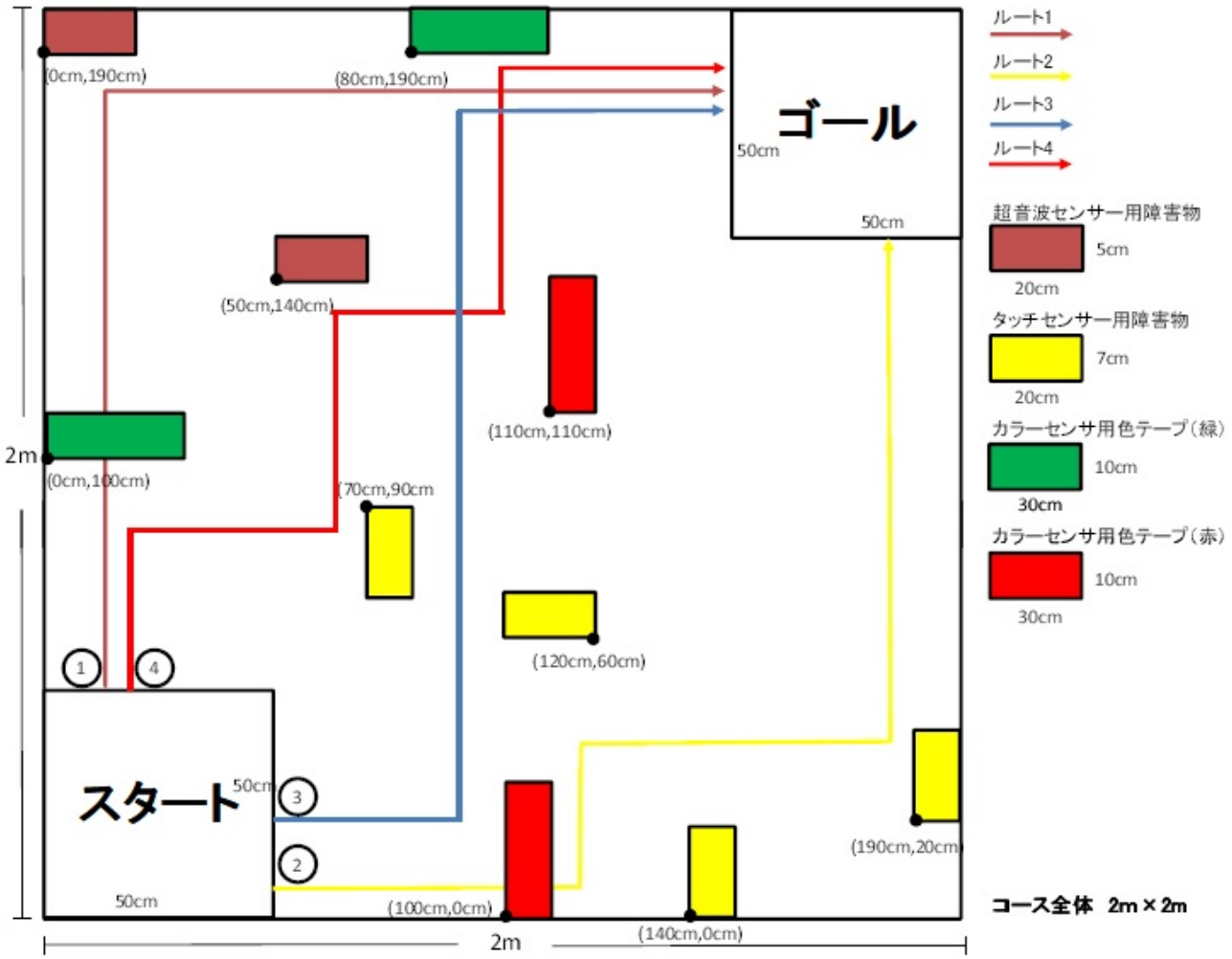
中級者向け講習会課題 3

カラーセンサーで障害物を検知し、それを
避けて進むシステムを作成する

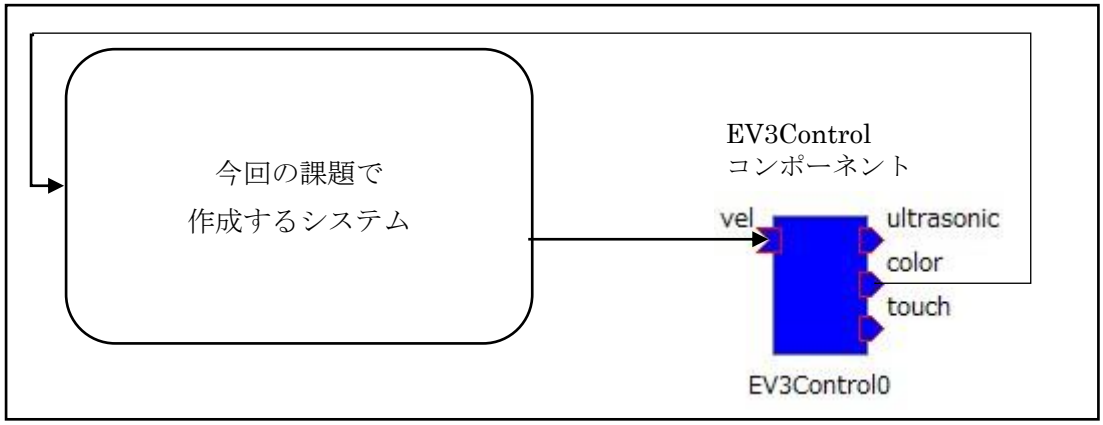
内容

1. システム概要

コース上の配置された障害物を EV3 に接続されたカラーセンサーで検知し、少し下がり旋回して障害物を避けて進むシステムを作成してください。通るルートは下記図のルート③です。



下記 EV3Control コンポーネントは EV3 制御用コンポーネントです。機能は、接続されたセンサーの値を各 OutPort から出力し、EV3 のモータを Inport からの入力の値で制御します。センサーの値を受け取りその値を元に EV3 の速度を決定して出力するのが今回の課題です。



EV3Control コンポーネントの InPort と OutPort は以下の内容になります。

ポート種類	ポート名	データの型	説明
InPort	vel	RTC::TimedVelocity2D	EV3 の速度の値
OutPort	ultrasonic	RTC::RangeData	超音波センサーの値
	color	RTC::TimedString	カラーセンサーの値
	touch	RTC::TimedBooleanSeq	タッチセンサーの値

EV3Control のカラーセンサーのデータ型は以下のようになります。

```
struct TimedString
{
   Time tm;
   string data;
};
```

上記の変数の data にカラーセンサーの値が入ります。

OpenRTM で使用される変数の情報は以下のページに記載されています。

http://openrtm.org/doc/idl/1.1/idlreference_ja/annotated.html

カラーセンサーの値取得のサンプルソース（C++）

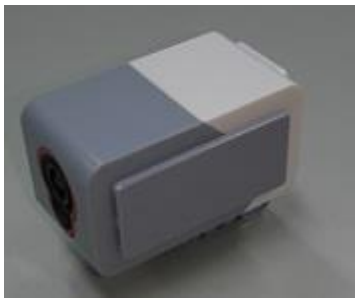
```
std::string color;

// カラーセンサーの値が更新された場合
if (m_colorIn.isNew())
{
        // カラーセンサーの値 読み込み
        m_colorIn.read();
        // カラーセンサーの値 取得
        color = m_color.data;
}
```

カラーセンサーの値取得のサンプルソース（Python）

```
# カラーセンサーの値が更新されている場合
if self._colorIn.isNew():
        # カラーセンサーの値 読み込み
        self._d_color = self._colorIn.read()
        # カラーセンサーの値 取得
        color_val = self._d_color.data
```

2. カラーセンサーの仕様

① カラーセンサーについて



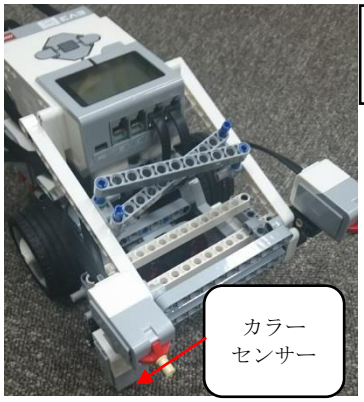
カラーセンサーとは異なる色を検知することが出来るセンサーです。EV3 のカラーセンサーは左の写真の物になります。このカラーセンサーを EV3 に接続することによりカラーセンサーを使用することが出来る様になります。

② タッチセンサーの性能

色	値
なし	0
黒	1
青	2
緑	3
黄色	4
赤	5
白	6
茶	7

③ カラーセンサーの値の取り方

※EV3 の起動方法については EV3 起動手順.docx を参照



EV3 に接続されたカラーセンサーはどの様に値を取っているのか確認してみましょう。

1)EV3 の電源を入れ、しばらくすると初期画面が表示されます。

2)その後十字キーと中央ボタンで[Device Browser]- [Sensors] - [lego-ev3-color at in1]と選択します。

※[lego-ev3-color at in1]の at in1 はポート番号をさしているので異なる場合があります。

3)選択後下ボタンを押し、[Watch values]を選択してください。

4)カラーセンサーで現在取得している値が確認できます。

5)EV3 に接続されたカラーセンサーの前に物体の色によって値が変化します。

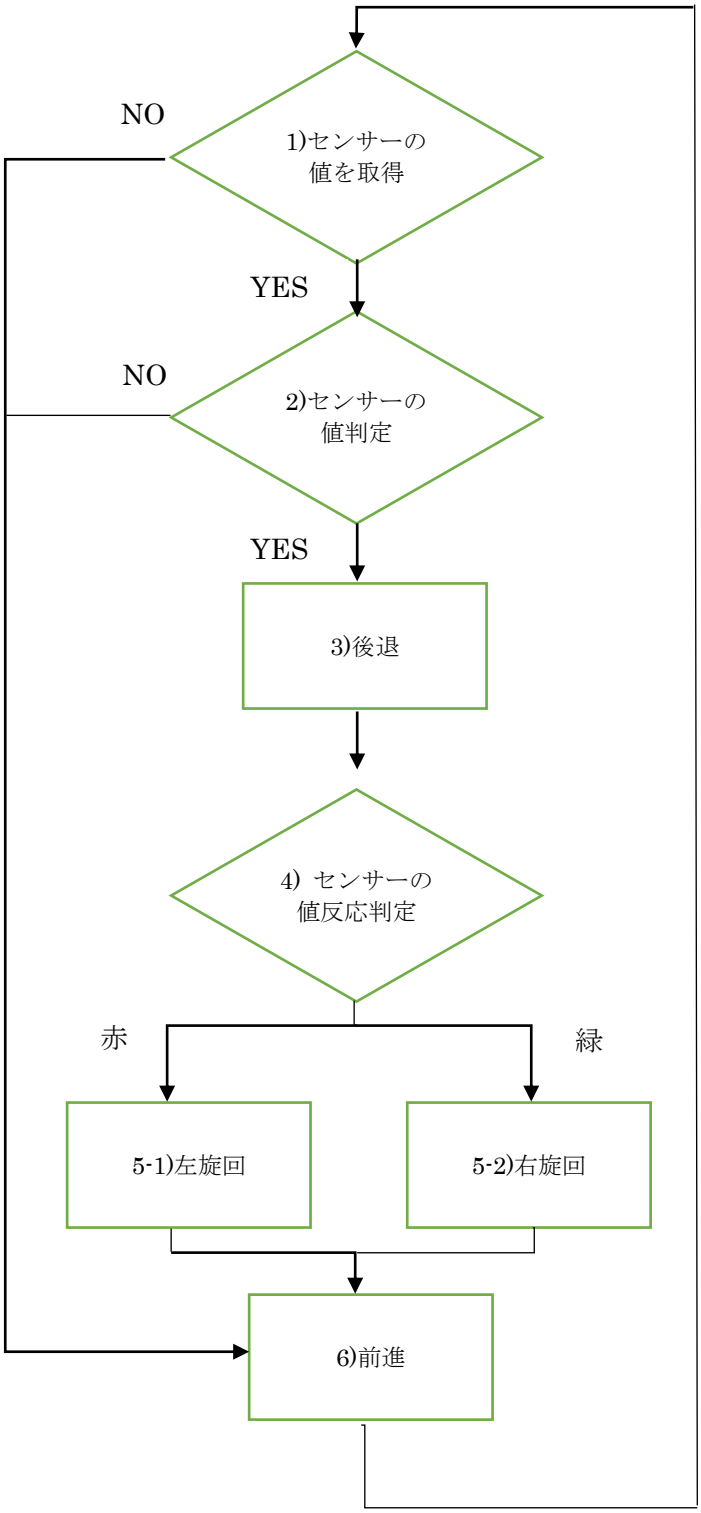
3. 速度の与え方

速度の与え方については課題 1 で説明した通りです。

4. 作成のヒント

① システムのアルゴリズム

作成するシステムの一例を示します。今回のシステムの動きをフローチャートにすると以下の様になります。



1)センサーの値の取得の判定。NO なら 6)前進の処理に移行。YES なら 2)の判定に移行。
2)センサーの値が赤か緑か判定。赤か緑なら YES、赤か緑でないなら NO。NO なら 6)前進の処理に移行。YES なら 3)後退の処理に移行。
3)後退の処理実行。終了後 4)判定に移行。
4)センサーの値が赤か緑かを判定。赤の場合 5-1)左旋回の処理に移行。緑の場合 5-2)右旋回の処理に移行。
5-1)左旋回の処理実行。終了後 6)前進の処理に移行。
5-2)右旋回の処理実行。終了後 6)前進の処理に移行。
6)前進の処理実行。終了後 1)の判定に戻る。
といった様になります。後退や旋回の処理に関しては速度を一回与えた後 sleep 関数を使用して一定時間コンポーネントを停止させる方法などがあります。