中級者向け講習会課題 2-1

会津大学　ＲＴミドルウェア講習会

タッチセンサーで障害物を検知し、それを避けて進むプログラムを作成する

目次

※ 文中の「x.y」や「x.y.z」の表記は使用環境の OpenRTM-aist のバージョンに読み替えてください。

当ドキュメントは下記ページを参考にしています。

・移動ロボット Kobuki の制御

http://www.openrtm.org/openrtm/ja/content/raspberrypi_kobuki_control

(2016/1/20 アクセス)

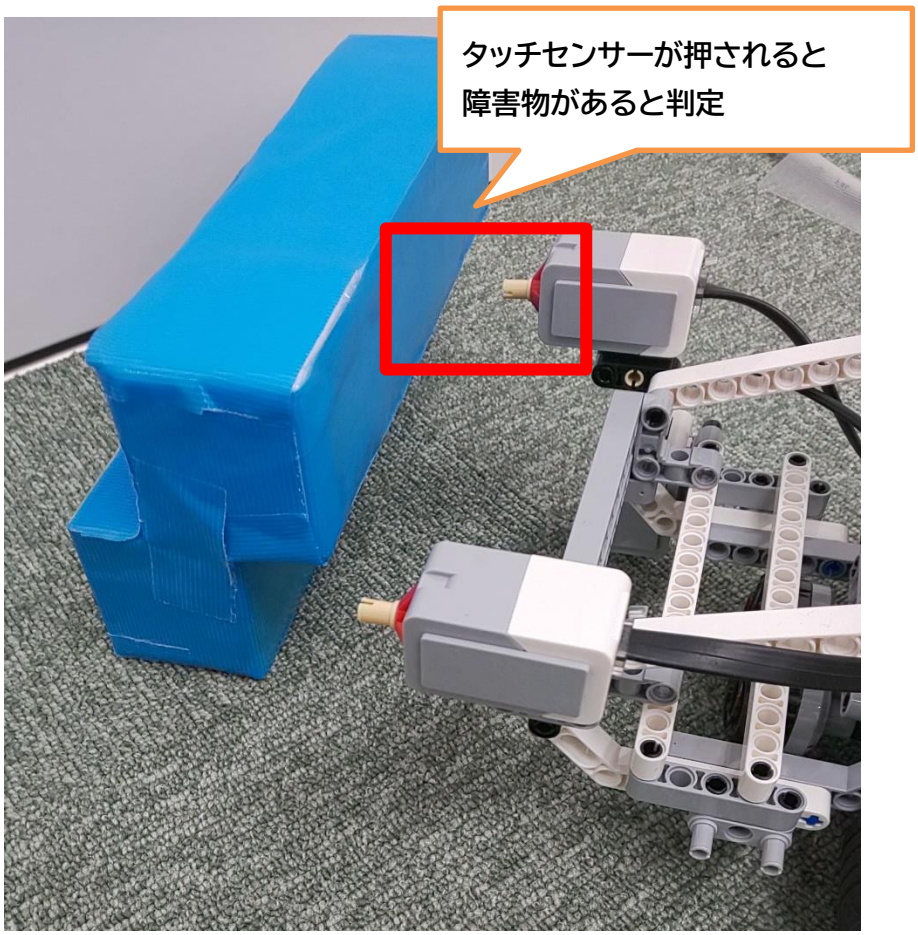
・LEGO Mindstorms EV3 活用事例

http://www.openrtm.org/openrtm/ja/casestudy/lego_mindstorm_ev3 （2016/1/20 アクセス）

1 システム概要

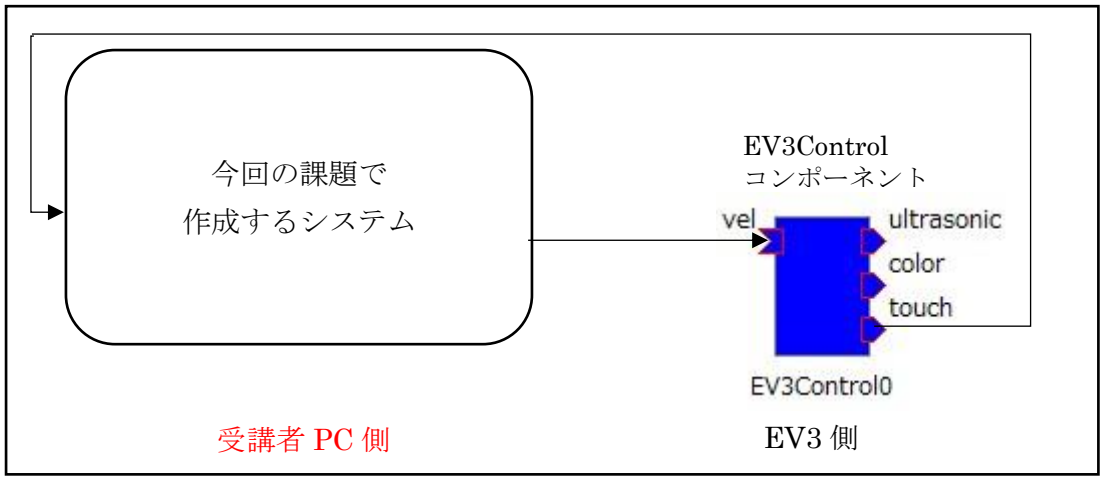
1.1 システム概要

EV3 に接続されたタッチセンサーで障害物を検知し、後退した後に旋回して障害物を避けて進むプログラムを作成してください。



1.2 コンポーネント概要

下記 EV3Control コンポーネントは、EV3 を制御するための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 からの入力値で制御します。今回の課題では、センサーにて取得した値を受け取り、その値を基に EV3 の速度を決定して出力を行います。



1.2.1 入出力ポート

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

ポート種別	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

1.2.2 タッチセンサーのデータ型

EV3Control のタッチセンサーのデータ型は以下のようになります。

【RTC.TimedBooleanSeq】

型	変数名	意味
sequence<boolean>	Data	True(1)か False(0)が格納されている配列データ
RTC.Time	Tm	タイムスタンプ※今回は使用しません

上記の変数の data は boolean 型の配列です。配列の長さ（要素数）は 2 で 0 番目（先頭）の要素に左のタッチセンサーの値が入り、1 番目に右のタッチ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://tmp.openrtm.org/doc/idl/1.1/idlreference_ja/annotated.html

1.2.3 サンプルソース

タッチセンサーの値取得のサンプルソース（Python）

```
        # タッチセンサーの値が更新されている場合
        if self._touchIn.isNew():
                # タッチセンサーの値を読み込み
                self._d_touch = self._touchIn.read()
```

タッチセンサーの値比較のサンプルソース（Python）

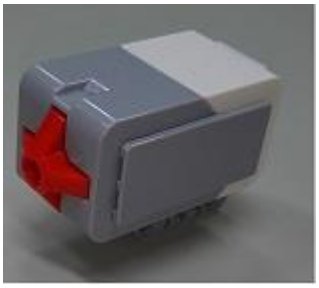
```
        # タッチセンサーの右押されている場合
        if self._d_touch.data[1]==1:
                # ここに処理追加

        # タッチセンサーの左押されている場合
        if self._d_touch.data[0]==1:
                # ここに処理追加

        # タッチセンサーのどちらかが押されている場合
        if self._d_touch.data[0]==1 OR self._d_touch.data[1]==1:
                # ここに処理追加
```

「def __init__(self, manager):」内を以下のように修正します。

```
修正前：self._d_vel = RTC.TimedVelocity2D(*vel_arg)
修正後：self._d_vel = RTC.TimedVelocity2D(RTC.Time(0, 0), RTC.Velocity2D(0.0, 0.0,
0.0))
```

2 タッチセンサーの仕様

2.1 タッチセンサーについて



タッチセンサーとは、ボタンが押されたかどうかを検知することが出来るセンサーです。EV3 ので使用しているタッチセンサーは左図となります。このタッチセンサーを EV3 に接続することによりタッチセンサーを使用することが出来るようになります。

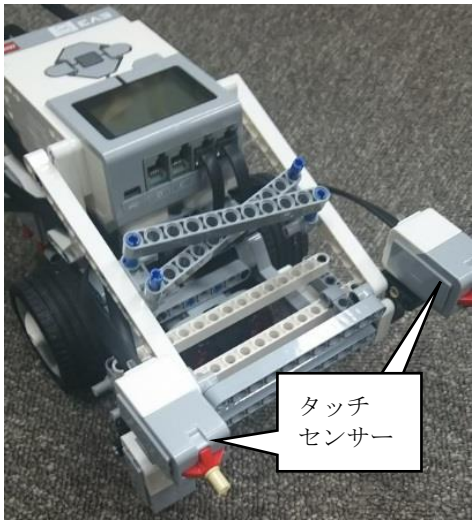
2.2 タッチセンサーの性能

センサー状態	値
Released	0
Press	1

2.3 タッチセンサーの値の取り方

EV3 の起動方法については「EV3 起動終了手順」を参照

EV3 に接続されたタッチセンサーはどのように値を取っているのか確認してみましょう。



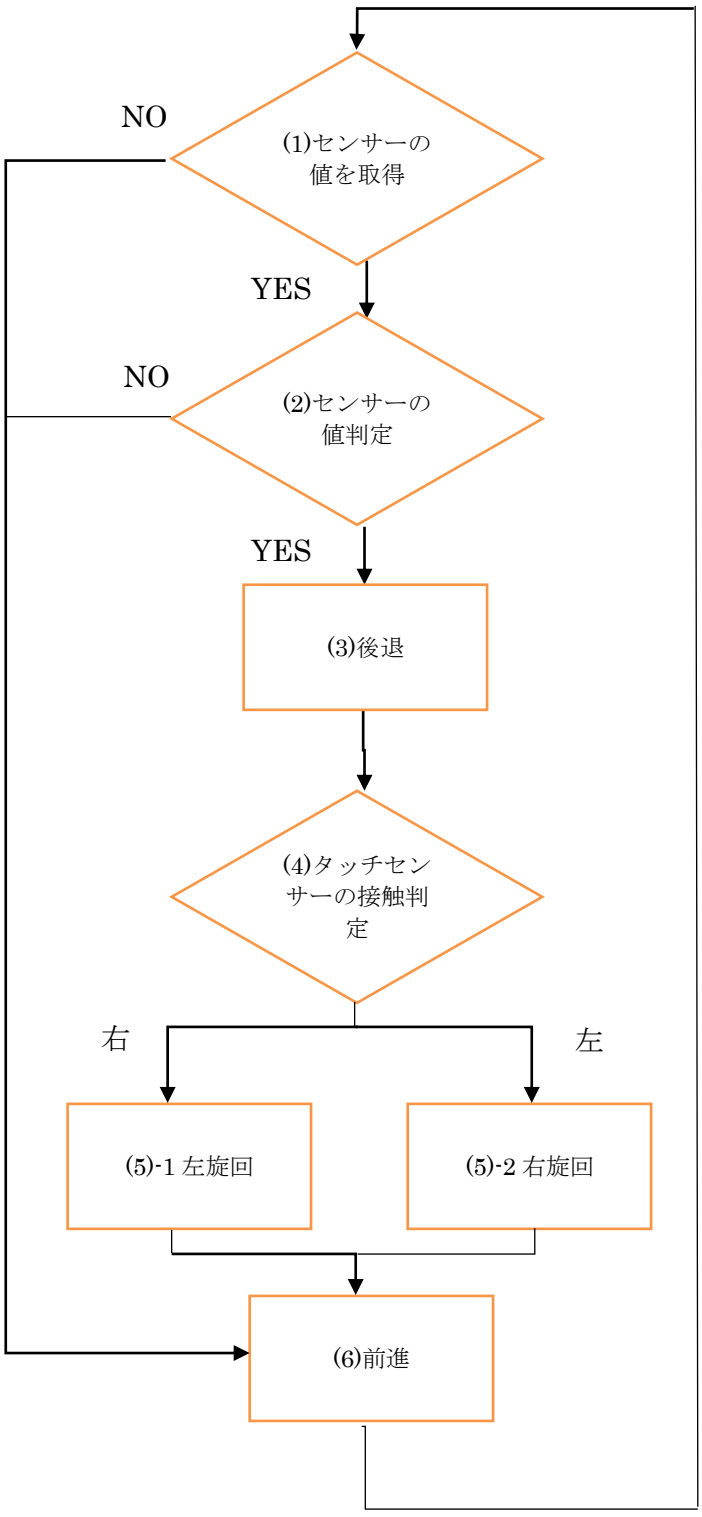
3 速度の与え方

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

4.1 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ようになります。



(1) センサーの値を取得できたか判定。NO なら(6)前進の処理に移行。YES なら(2)の判定に移行。
(2) 値が規定値か判定。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。後退処理終了後に(4)タッチセンサーの接触判定に移行。
(4) 接触したセンサーが左右のどちらかを判定。右の場合(5)-1 左旋回の処理に移行。左の場合(5)-2 右旋回の処理に移行。
(5) -1 左旋回の処理実行。旋回処理終了後に(6)前進の処理に移行。
-2 右旋回の処理実行。旋回処理終了後に(6)前進の処理に移行。
(6) 前進の処理実行。前進処理終了後(1)の判定に戻る。

といったようになります。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。

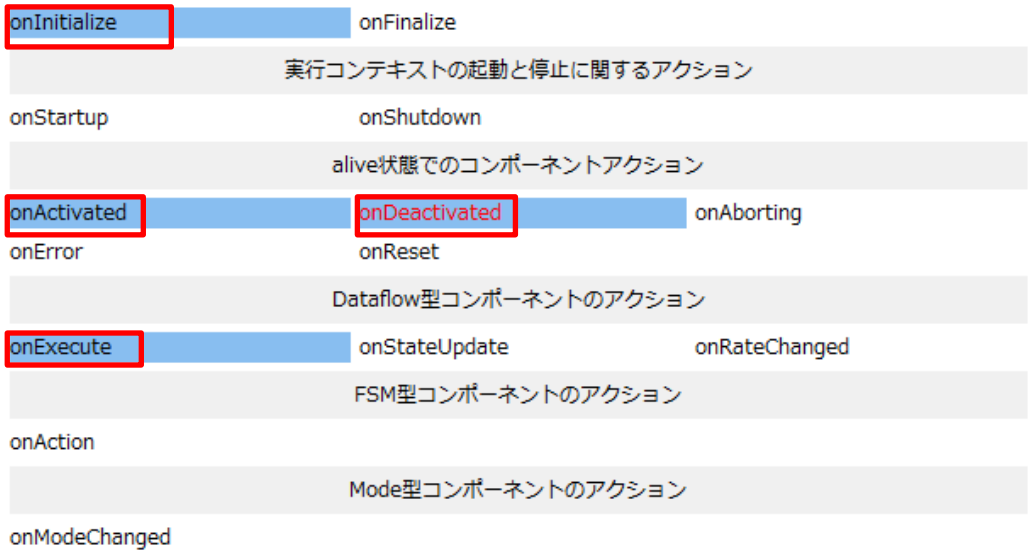
4.2 雛型の値

RTCBuilder で作成する雛形の例を以下に記載します。

4.2.1.1 基本

モジュール名：Subject2
モジュール概要：Subject2 component
バージョン：1.0.0
ベンダ名：Aizu
モジュールカテゴリ：Category
コンポーネント型：STATIC
アクティビティ型：PERIODIC
コンポーネント種類：DataFlowComponent
最大インスタンス数：1
実行型：PeriodicExecutionContext
実行周期：1000.

4.2.1.2 アクティビティ



4.2.1.3 データポート

・InPort
ポート名: touch
データ型: RTC::TimedBooleanSeq
変数名: touch
表示位置: left

・OutPort
ポート名: vel
データ型: RTC::TimedVelocity2D
変数名: vel
表示位置: right

4.2.1.4　コンフィギュレーション

なし

4.2.1.5　言語・環境

“Python”を選択

4.3 障害物を検知して旋回するサンプルソース

タッチセンサーを使用したサンプルソースを以下に載せます。穴埋め式になっていますので赤文字の所を自分で埋めてみてください。

4.3.1 サンプルソース（Python）

```
def onExecute(self, ec_id):

        # タッチセンサーの値が更新されている場合
        if self._#InPort のポート名#In.isNew():
                # タッチセンサーの値取得
                self._d_# InPort の変数名# = self._ # InPort ののポート名#In.read()
                # タッチセンサーの何れかの値が 1 の場合
                If#左右のタッチセンサーどちらかが押されている時#:
                        # 後退指定
                        self._d_vel.data.vx = -0.04
                        self._d_vel.data.va = 0.0
                        self._d_vel.data.vy = 0.0
                        self._velOut.write()
                        # Sleep
                        time.sleep(2);

                        # 右のタッチセンサーの値が 1 の場合
                        if #右のタッチセンサーが押されている時# == 1:
                                # 左旋回指定
                                self._d_vel.data.vx = 0.0
                                self._d_vel.data.va = 0.5
                                self._d_vel.data.vy = 0.0
                                self._velOut.write()
                                time.sleep(2)
                        else:
                                # 右旋回指定
                                self._d_vel.data.vx = 0.0
                                self._d_vel.data.va = -0.5
                                self._d_vel.data.vy = 0.0
                                self._velOut.write()
                                time.sleep(2)
        # 前進指定
        self._d_vel.data.vx = 0.04
        self._d_vel.data.va = 0.0
        self._d_vel.data.vy = 0.0
        self._velOut.write()

        return RTC.RTC_OK
```

「def __init__(self, manager):」内を以下のように修正します。

```
修正前：self._d_vel = RTC.TimedVelocity2D(*vel_arg)
修正後：self._d_vel = RTC.TimedVelocity2D(RTC.Time(0, 0), RTC.Velocity2D(0.0, 0.0,
0.0))
```