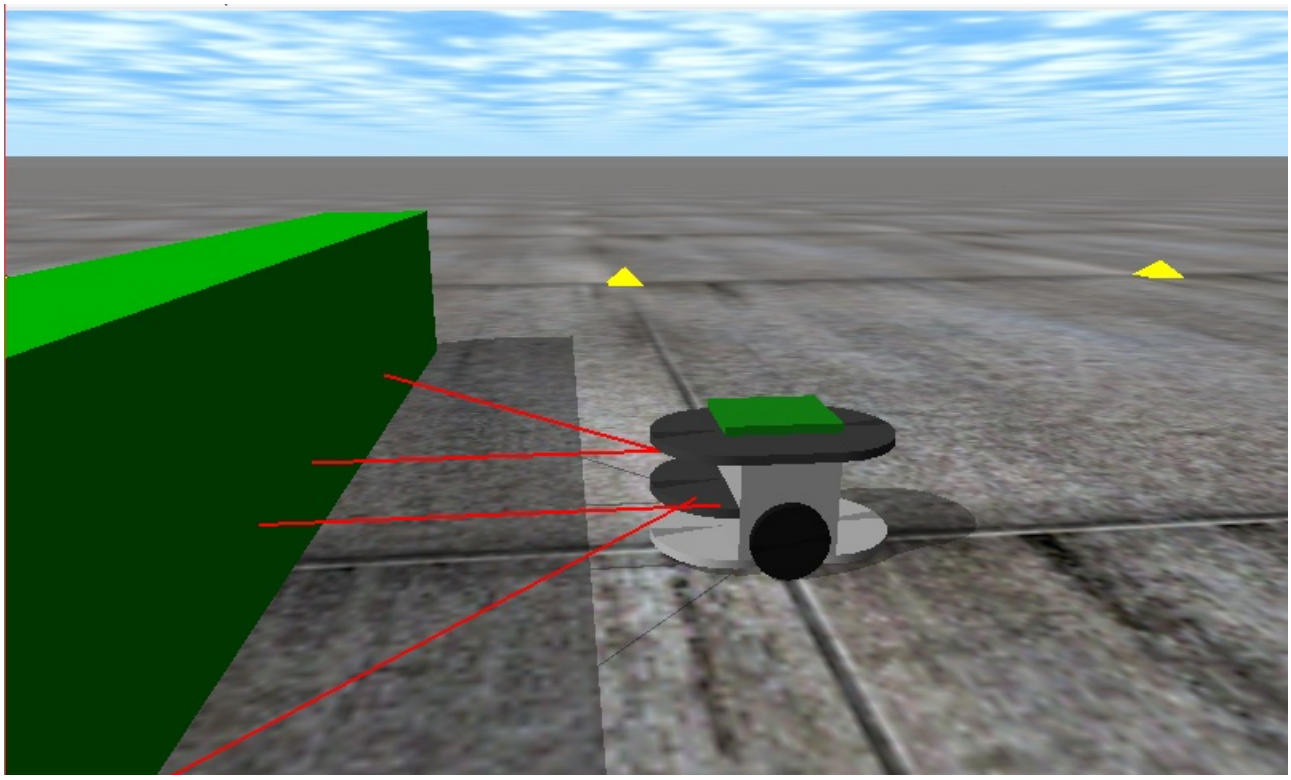
チュートリアル(Raspberry Pi Mouse、Python、Ubuntu、強化月間用)

いいね! Facebookに登録して、友達の「いいね!」を見てみましょう。

Table of contents

はじめに

このページではシミュレーター上の Raspberry Pi マウスを操作するためのコンポーネントの作成手順を説明します。

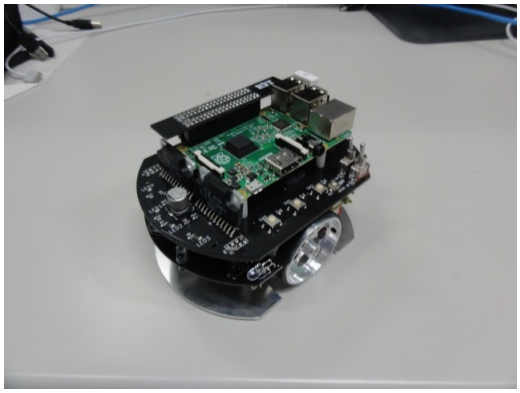


シミュレーター

- RaspberryPiMouseSimulatorコンポーネント

シミュレーターは Open Dynamics Engine(ODE) という物理演算エンジンと ODE 付属の描画ライブラリ(drawstuff)を使用して開発しています。OpenGL が動作すれば動くので、大抵の環境で動作するはずです。

以下の Raspberry Piマウス というロボットのシミュレーションができます。



シミュレーター上の Raspberry Pi マウスの動力学計算、接触応答だけではなく、距離センサーのデータも現実のロボットに近い値を再現するようにしています。

作成する RTコンポーネント

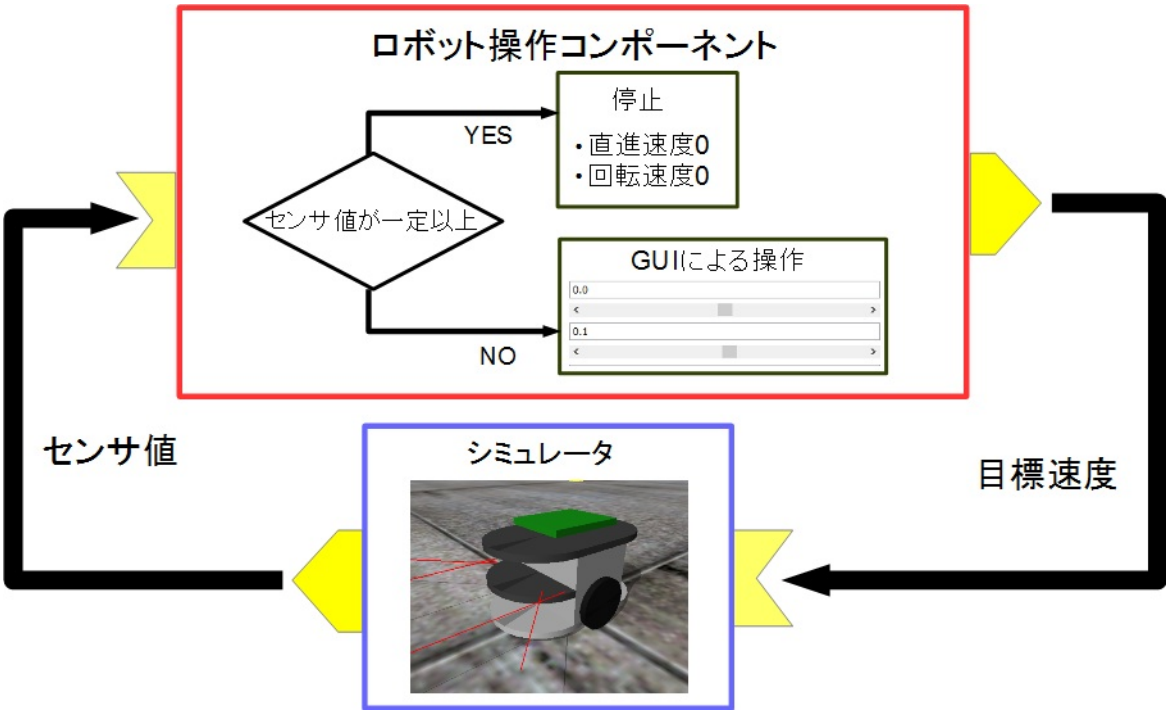
- RobotController コンポーネント

RaspberryPiMouseSimulator コンポーネントと接続してシミュレーター上のロボットを操作するためのコンポーネントです。

RobotController コンポーネントの作成

GUI(スライダー)によりシミュレーター上のロボットの操作を行い、センサー値が一定以上の時には自動的に停止するコンポー

ネントの作成を行います。



作成手順

作成手順は以下の通りです。

- 開発環境の確認
- コンポーネントの仕様を決める
- RTC Builder によるソースコードのひな形の作成
- ソースコードの編集
- コンポーネントの動作確認

動作環境・開発環境

Linux (ここでは Ubuntu 16.04 を仮定) 上に開発環境を構築します。

OpenRTM-aistのインストール

インストールスクリプトでインストールします。

```
$ wget http://svn.openrtm.org/OpenRTM-aist/trunk/OpenRTM-aist/build/pkg_install_ubuntu.sh
$ sudo sh pkg_install_ubuntu.sh -lpython
```

OpenRTP のインストール

こちらのURL から Linux版の OpenRTP (コンポーネント開発ツール、システム開発ツール統合環境) をダウンロード、インストールします。OepnRTP の実行には Java も必要となりますので default-jre パッケージをインストールします。

```
$ apt-get install default-jre
$ wget http://openrtm.org/pub/openrtp/packages/1.1.2.v20160526/eclipse442-openrtp112v20160526-ja-linux-gtk-x86_64.tar.gz
```

```
$ tar xvzf eclipse442-openrtp112v20160526-ja-linux-gtk-x86_64.tar.gz
```

eclipse起動後、RTSystemEditor でネームサーバに接続できない場合があります。その場合、/etc/hosts の localhost の行に自ホスト名を追記してください。

```
$ hostname
ubuntu1404 ← ホスト名は ubuntu1404
$ sudo vi /etc/hosts
```

```
127.0.0.1       localhost
を以下のように変更
127.0.0.1       localhost ubuntu1404
```

Python用エディタのインストール

PyDev等、Python用のエディタをインストールしてください。

RaspberryPiMouseSimulator コンポーネント

シミュレーターコンポーネントについては手動でビルドを行います。以下のコマンドを入力してください。

```
$ wget https://raw.githubusercontent.com/Nobu19800/RTM_Tutorial_2017/master/install_raspimouse_simulator.sh
$ sudo sh install_raspimouse_simulator.sh
```

インターネットに接続できない環境で講習会を実施している場合がありますので、その場合は配布の USBメモリー内のスクリプトを起動してください。

```
$ sudo sh install_raspimouse_simulator_usb.sh
```

コンポーネントの仕様

RobotController は目標速度を出力するアウトポート、センサー値を入力するインポート、目標速度や停止するセンサー値を設定するコンフィギュレーションパラメーターを持っています。

コンポーネント名称
RobotController
InPort
ポート名
in
型
TimedShortSeq

説明
センサー値
OutPort
ポート名
out
型
TimedVelocity2D
説明
目標速度
Configuration
パラメーター名
speed_x
型
double
デフォルト値
0.0
制約
-1.0<x<1.0
Widget
slider
Step
0.01
説明
直進速度の設定
Configuration
パラメーター名
speed_r
型
double
デフォルト値
0.0
制約
-2.0<x<2.0
Widget
slider
Step
0.01
説明
回転速度の設定
Configuration
パラメーター名

stop_d

型
int
デフォルト値
30
説明
停止するセンサー値の設定

TimedVelocity2D 型について

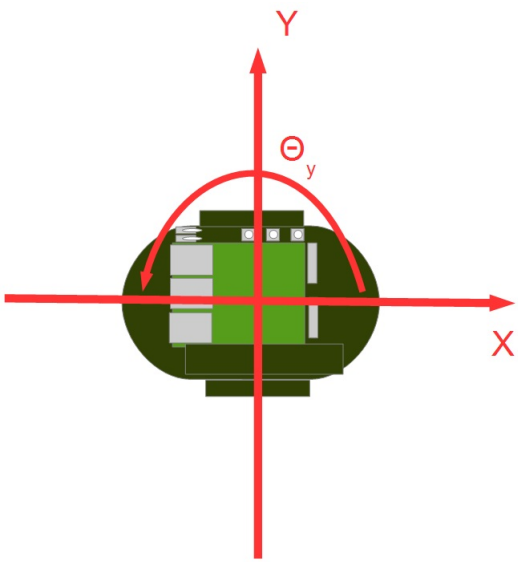
2次元平面上の移動ロボットの移動速度を格納するデータ型である TimedVelocity2D 型を使用します。

```
struct Velocity2D
{
  /// Velocity along the x axis in metres per second.
  double vx;
  /// Velocity along the y axis in metres per second.
  double vy;
  /// Yaw velocity in radians per second.
  double va;
};


struct TimedVelocity2D
{
  Time tm;
  Velocity2D data;
};
```

このデータ型にはX軸方向の速度**vx**、Y軸方向の速度**vy**、Z軸周りの回転速度**va**が格納できます。

vx、**vy**、**va**はロボット中心座標系での速度を表しています。



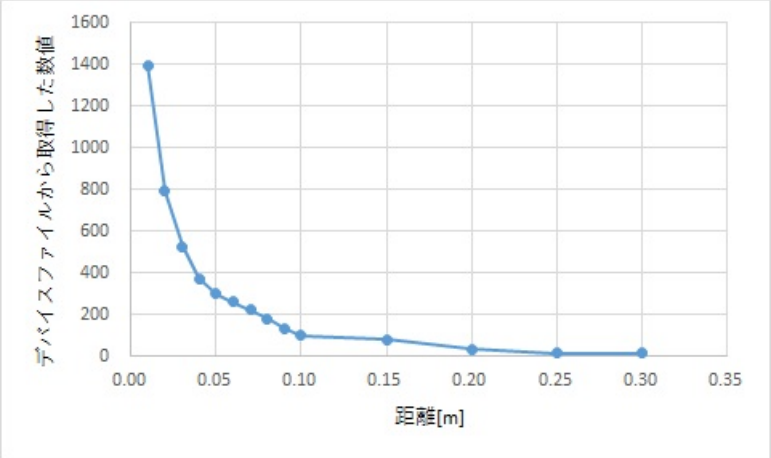
vxはX方向の速度、**vy**はY方向の速度、**va**はZ軸周りの角速度です。

Raspberry Piマウスのように2個の車輪が左右に取り付けられているロボットの場合、横滑りしないと仮定すると**vy**は0になります。

直進速度**vx**、回転速度**va**を指定することでロボットの操作を行います。

距離センサーのデータについて

Raspberry Pi マウスの距離センサーのデータは物体との距離が近づくほど大きな値を出力するようになっています。



デバイスファイルから取得した数値	
実際の距離[m]	
1394	
0.01	
792	
0.02	
525	
0.03	
373	
0.04	
299	
0.05	
260	
0.06	
222	
0.07	
181	
0.08	
135	
0.09	
100	

0.10
81
0.15
36
0.20
17
0.25
16
0.30

シミュレーターではこの値を再現して出力しています。RobotController コンポーネントではこの値が一定以上の時に自動的に停止する処理を実装します。

RobotController コンポーネントのひな型の生成

RobotController コンポーネントの雛型の生成は、RTCBuilder を用いて行います。

RTCBuilder の起動

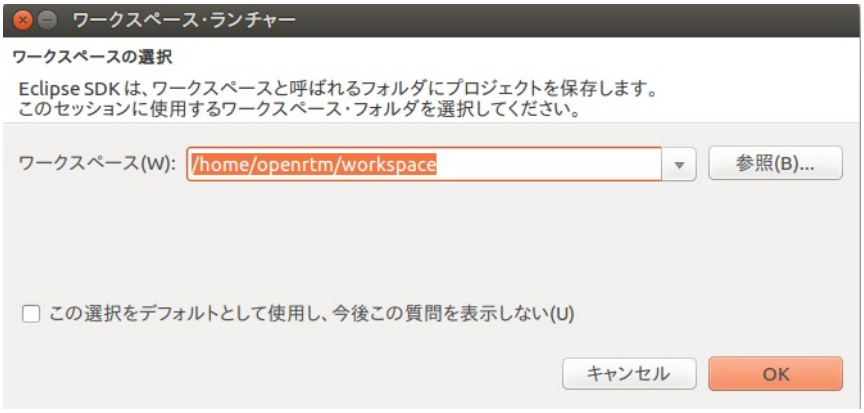
Eclipse では、各種作業を行うフォルダーを「ワークスペース」(Work Space)とよび、原則としてすべての生成物はこのフォルダーの下に保存されます。ワークスペースはアクセスできるフォルダーであれば、どこに作っても構いませんが、このチュートリアルでは以下のワークスペースを仮定します。

- /home/ユーザー名/workspace

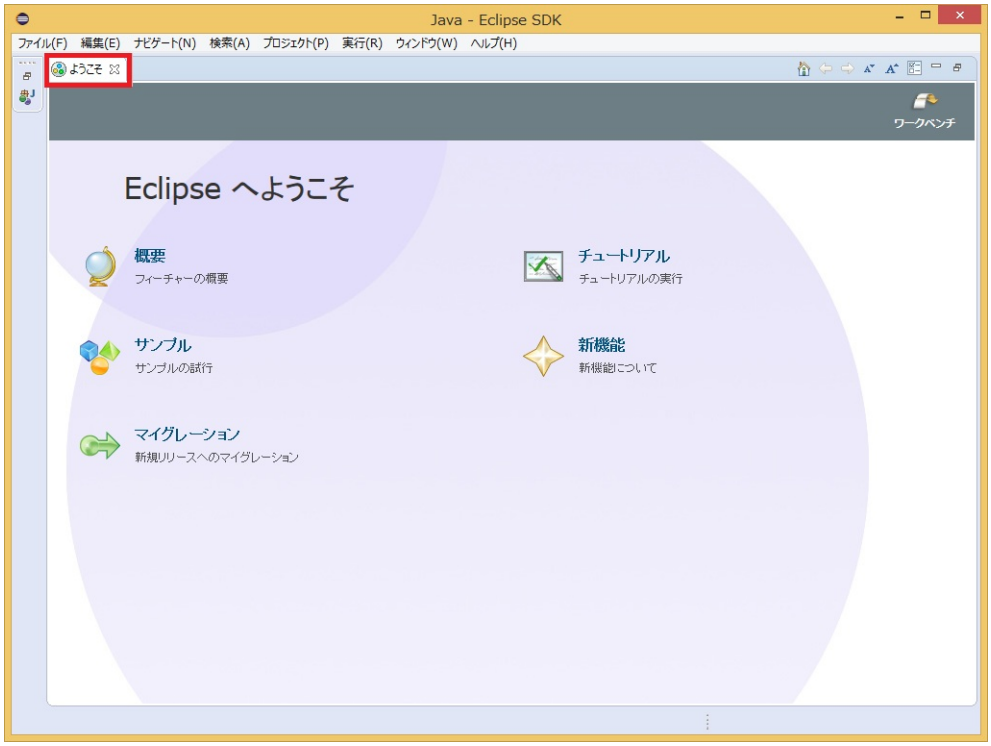
まずは Eclipse を起動します。OpenRTP を展開したディレクトリーに移動して以下のコマンドを入力します。

```
$ ./openrtp
```

最初にワークスペースの場所を尋ねられますので、上記のワークスペースを指定してください。



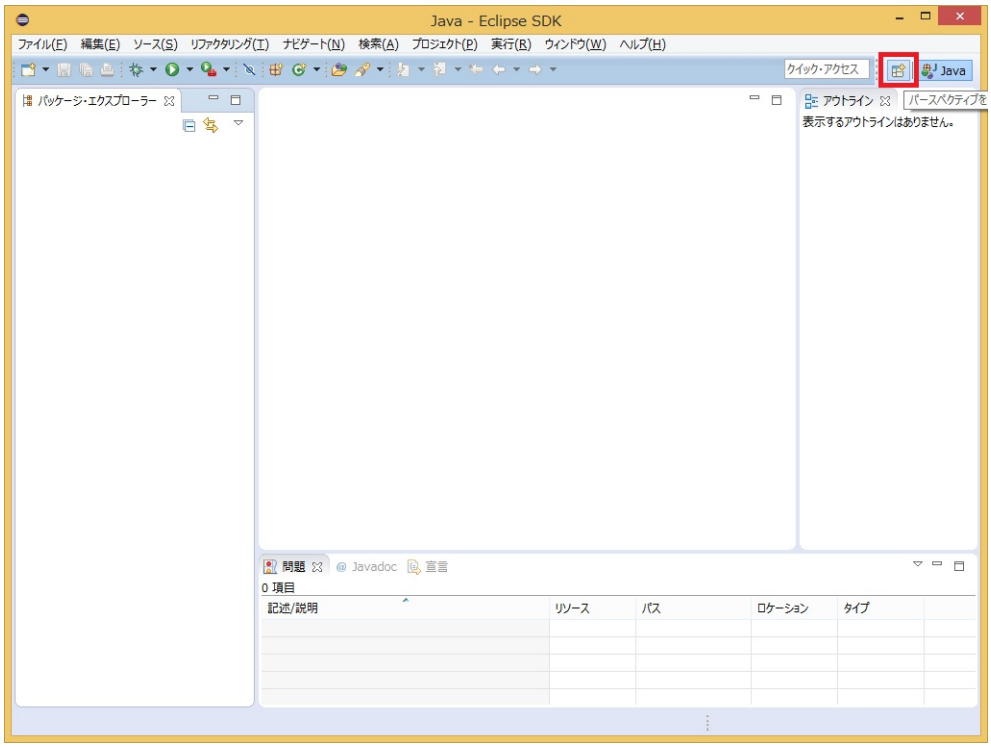
すると、以下のような Welcome ページが表示されます。



Eclipse の初期起動時の画面

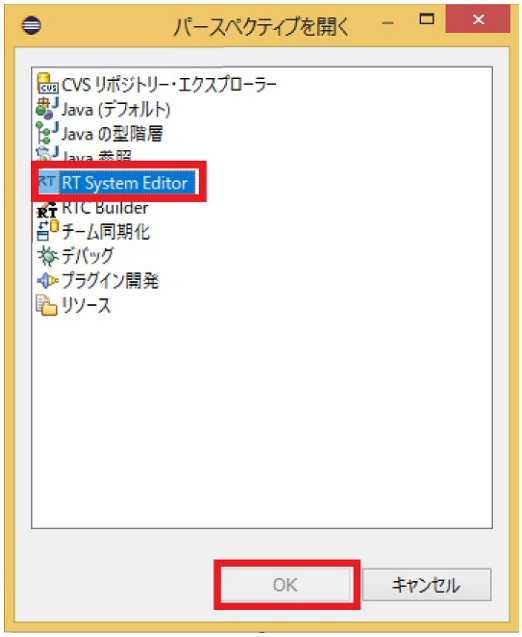
Welcome ページはいまは必要ないので左上の「×」ボタンをクリックして閉じてください。

右上の [Open Perspective] ボタンをクリックしてください。



パースペクティブの切り替え

「RTC Builder」を選択することで、RTCBuilderが起動します。メニューバーに「カナヅチとRT」の RTCBuilder のアイコンが現れます。

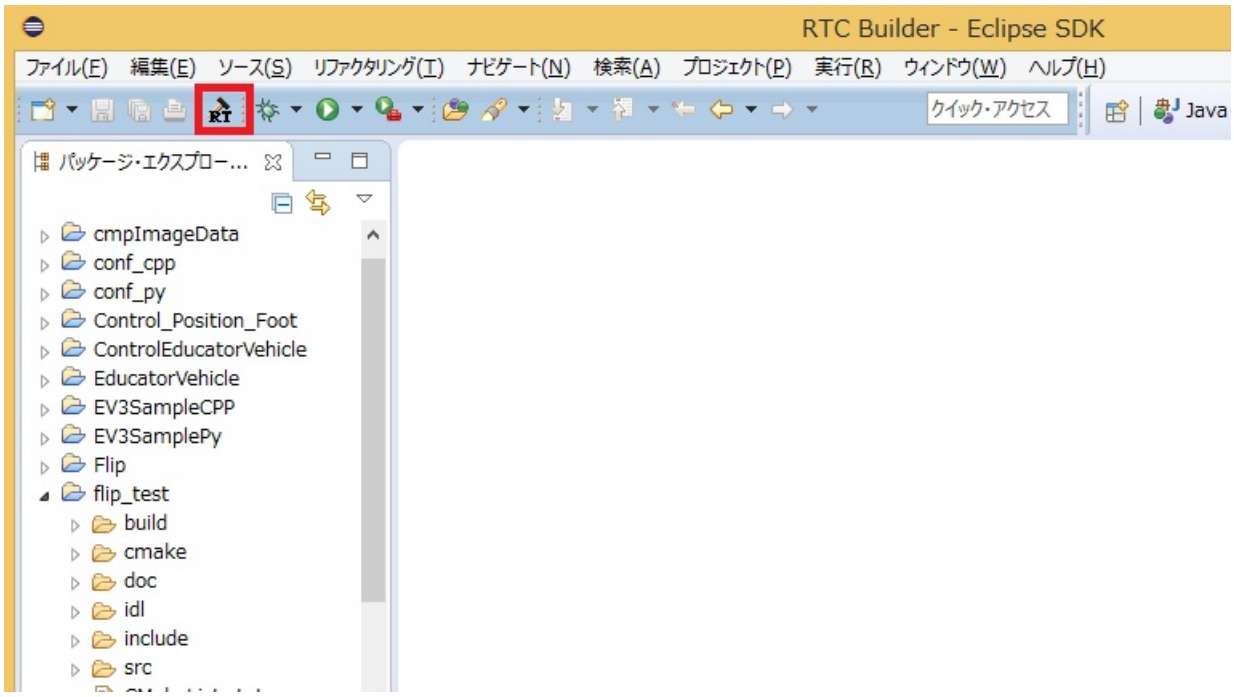


パースペクティブの選択

新規プロジェクトの作成

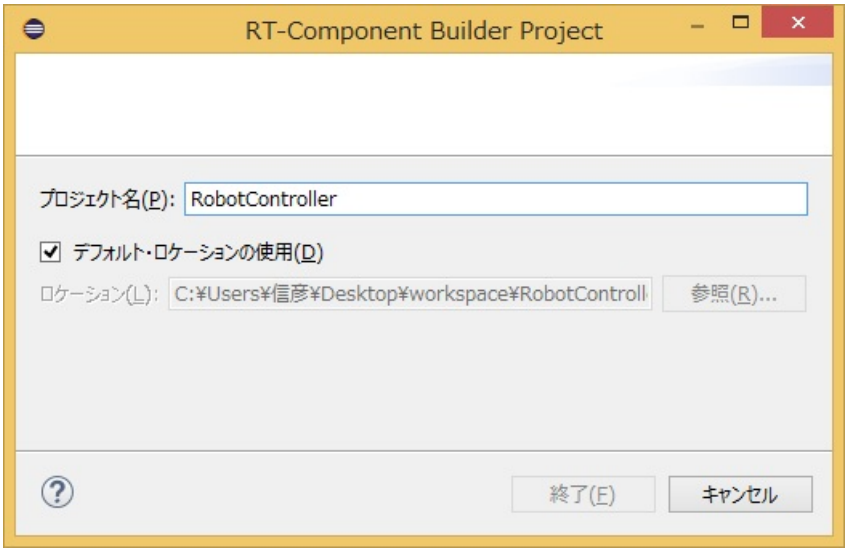
RobotController コンポーネントを作成するために、RTC Builder で新規プロジェクトを作成する必要があります。

左上の [Open New RTCBuilder Editor] のアイコンをクリックしてください。

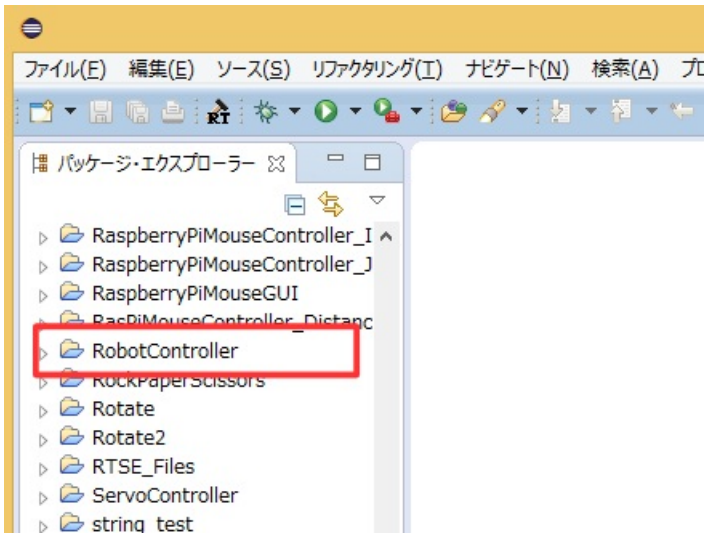


RTC Builder 用プロジェクトの作成

「プロジェクト名」欄に作成するプロジェクト名 (ここでは **RobotController**) を入力して [終了] をクリックします。



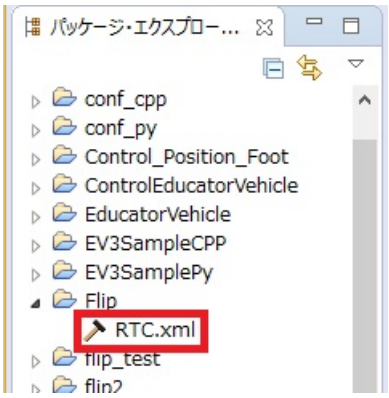
指定した名称のプロジェクトが生成され、パッケージエクスプローラ内に追加されます。



生成したプロジェクト内には、デフォルト値が設定された RTC プロファイル XML(RTC.xml) が自動的に生成されます。

RTC プロファイルエディタの起動

RTC.xmlが生成された時点で、このプロジェクトに関連付けられているワークスペースとして RTCBuilder のエディタが開くはずです。もし起動しない場合はパッケージエクスプローラーの RTC.xml をダブルクリックしてください。

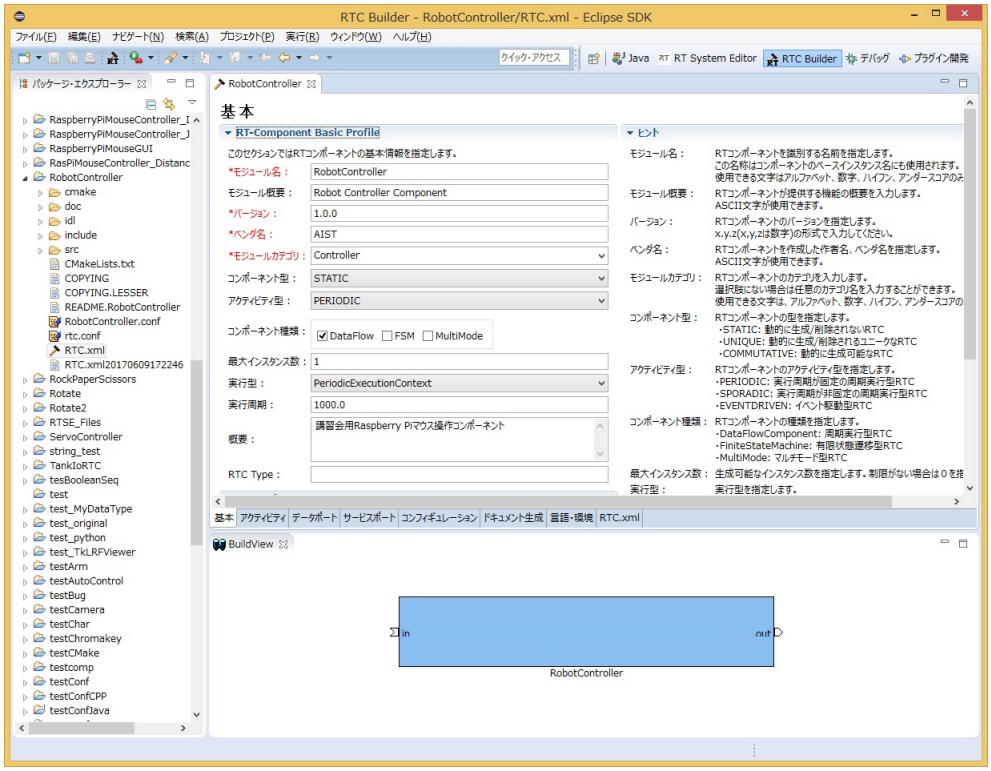


プロファイル情報入力とコードの生成

まず、いちばん左の「基本」タブを選択し、基本情報を入力します。先ほど決めた RobotController コンポーネントの仕様(名前)の他に、概要やバージョン等を入力してください。ラベルが赤字の項目は必須項目です。その他はデフォルトで構いません。

- モジュール名: RobotController

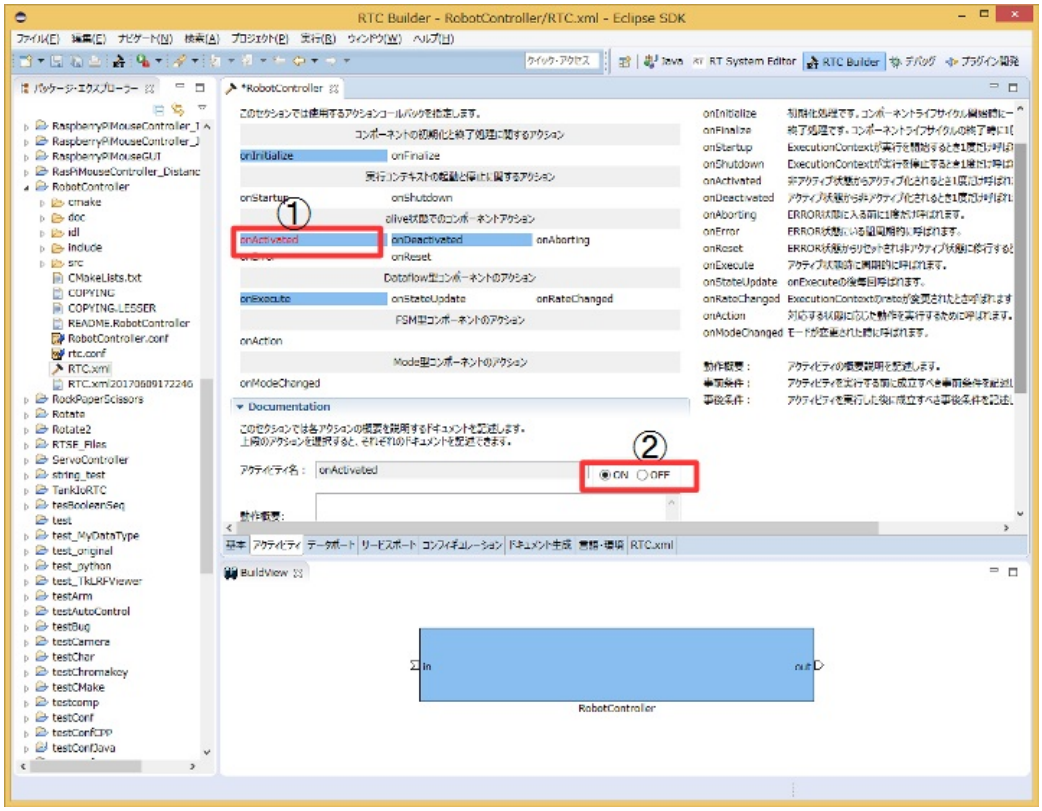
- モジュール概要: 任意(Robot Controller component)
- バージョン: 任意(1.0.0)
- ベンダ名: 任意
- モジュールカテゴリ: 任意(Controller)



基本情報の入力

次に、「アクティビティ」タブを選択し、使用するアクションコールバックを指定します。

RobotController コンポーネントでは、onActivated()、onDeactivated()、onExecute() コールバックを使用します。下図のように①の onAtivated をクリック後に②のラジオボタンにて [ON] にチェックを入れます。onDeactivated、onExecute についても同様の手順を行います。

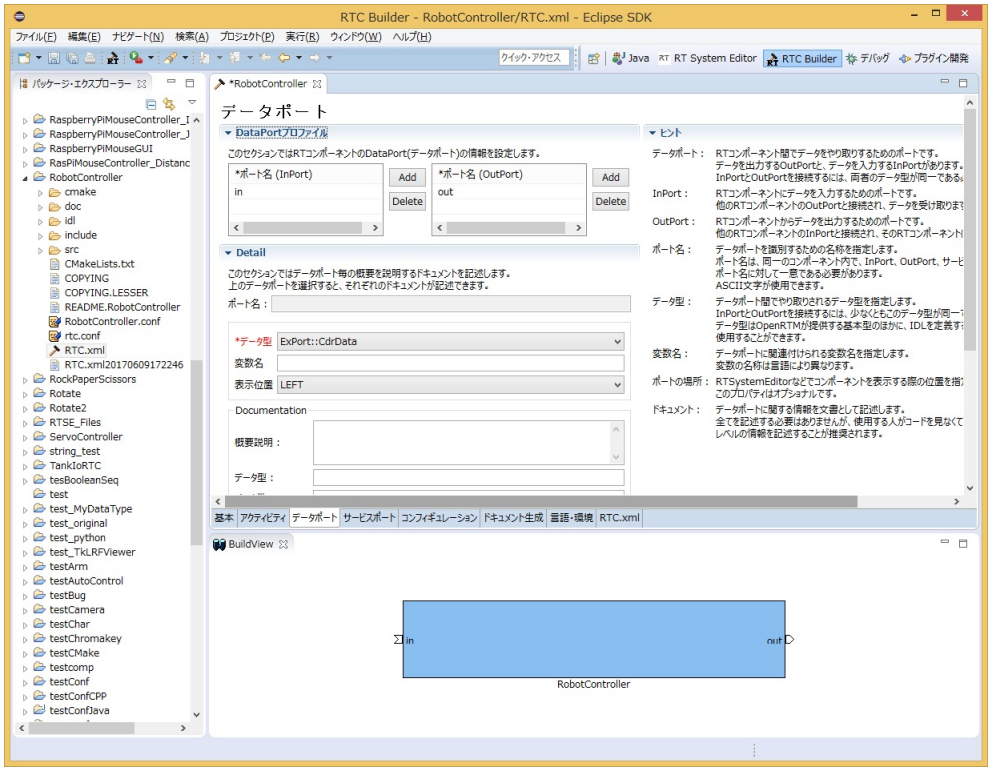


アクティビティコールバックの選択

さらに、「データポート」タブを選択し、データポートの情報を入力します。先ほど決めた仕様を元に以下のように入力します。なお、変数名や表示位置はオプションで、そのままで結構です。

- InPort Profile:
  - ポート名: in
  - データ型: TimedShortSeq

- OutPort Profile:
  - ポート名: out
  - データ型: TimedVelocity2D

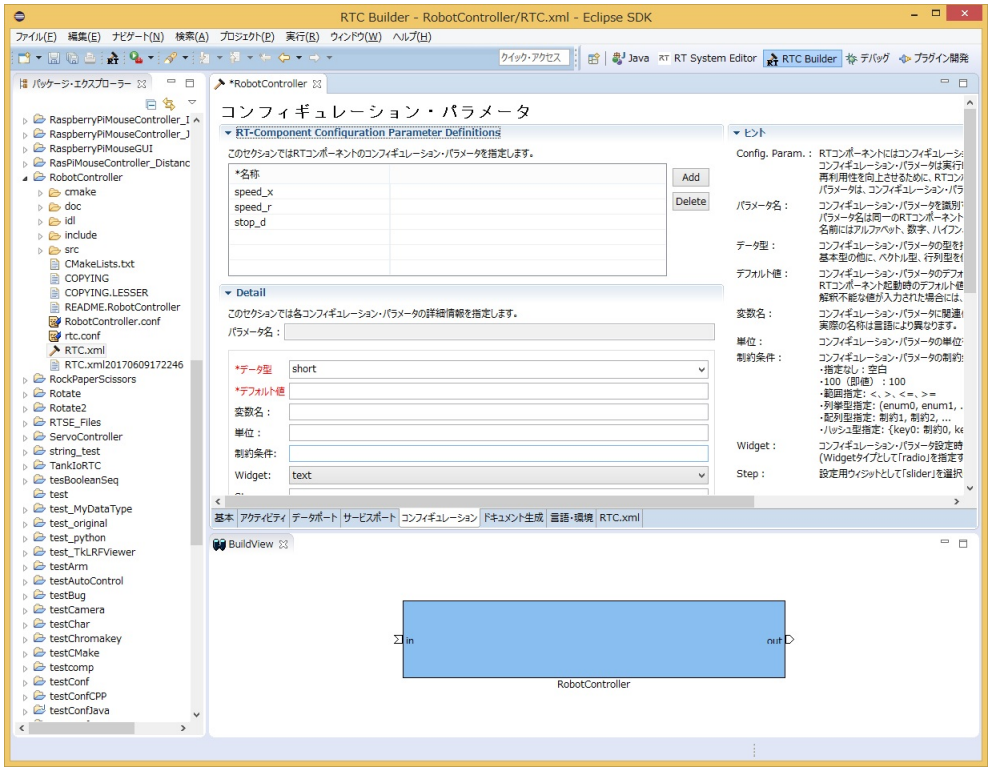


データポート情報の入力

次に、「コンフィギュレーション」タブを選択し、先ほど決めた仕様を元に、Configuration の情報を入力します。制約条件および Widget とは、RTSystemEditor でコンポーネントのコンフィギュレーションパラメーターを表示する際に、スライダー、スピンボタン、ラジオボタンなど、GUI で値の変更を行うためのものです。

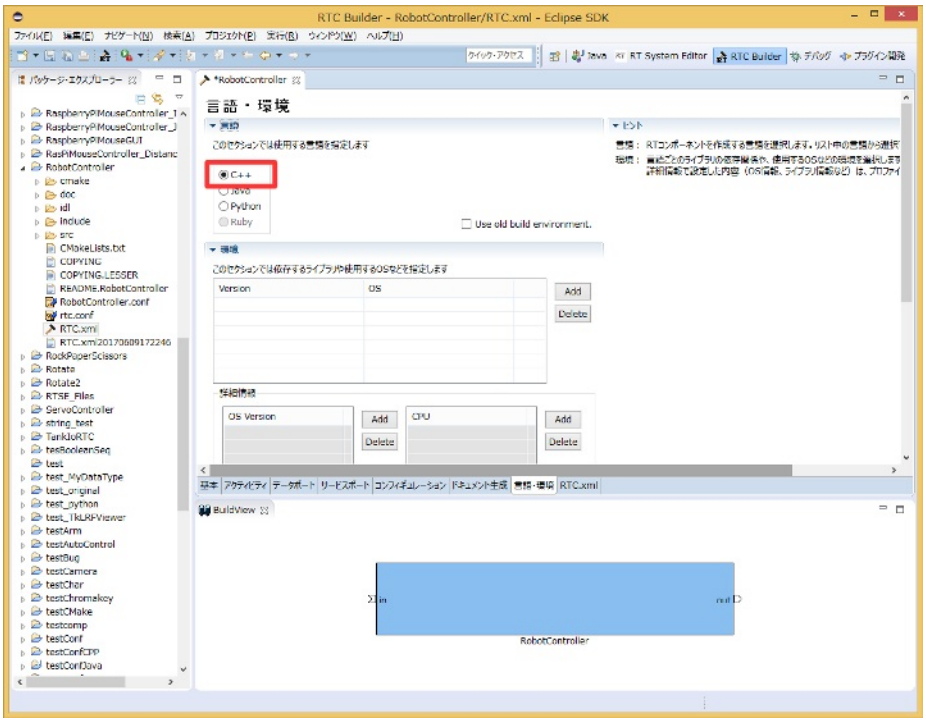
直進速度 speed_x、回転速度 speed_r はスライダーのより操作できるようにします。

- speed_x
  - 名称: speed_x
  - データ型: double
  - デフォルト値: 0.0
  - 制約条件: -1.0<x<1.0
  - Widget: slider
  - Step: 0.01
- speed_r
  - 名称: speed_r
  - データ型: double
  - デフォルト値: 0.0
  - 制約条件: -2.0<x<2.0
  - Widget: slider
  - Step: 0.01
- stop_d
  - 名称: stop_d
  - データ型: int
  - デフォルト値: 30
  - Widget: text



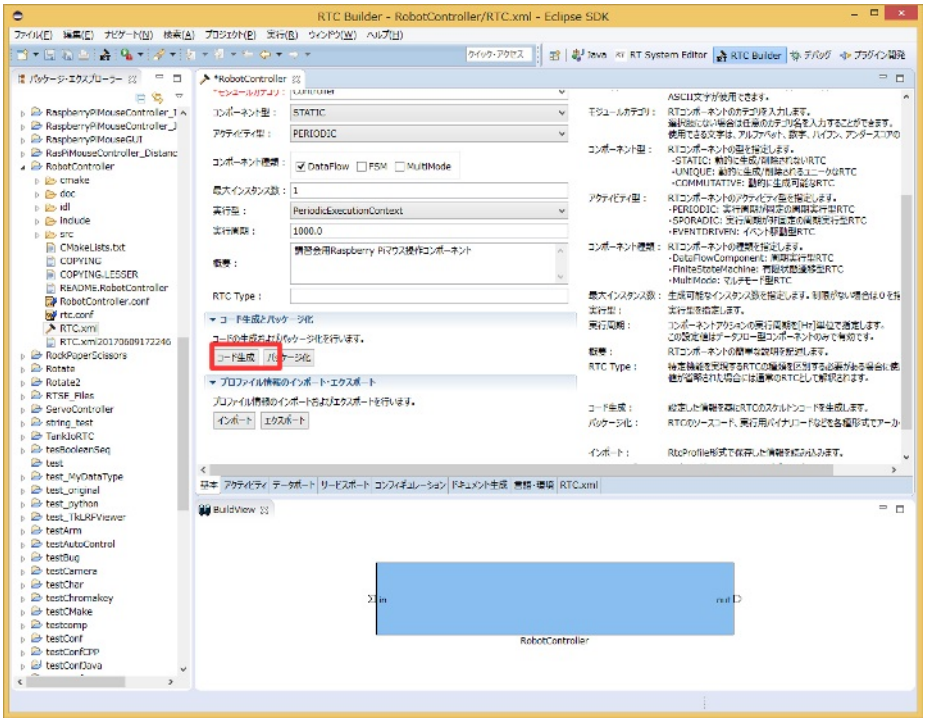
コンフィグレーション情報の入力

次に、「言語・環境」タブを選択し、プログラミング言語を選択します。ここでは、C++(言語) を選択します。なお、言語・環境はデフォルト等が設定されておらず、指定し忘れるとコード生成時にエラーになりますので、必ず言語の指定を行うようにしてください。



プログラミング言語の選択

最後に、「基本」タブにある [コード生成] ボタンをクリックし、コンポーネントの雛型を生成します。

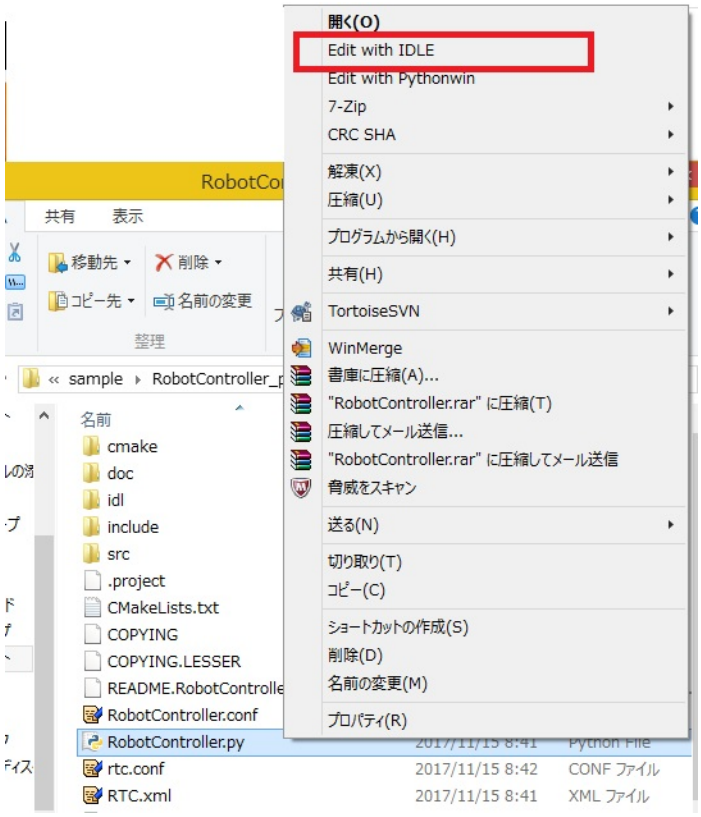


雛型の生成(Generate)

※ 生成されるコード群は、eclipse起動時に指定したワークスペースフォルダーの中に生成されます。現在のワークスペースは、[ファイル] > [ワークスペースの切り替え...] で確認することができます。

ソースコードの編集

<ワークスペースディレクトリー>/RobotController/RobotController.pyをPython用エディタで開いて編集してください。



変数初期化部分の修正

OpenRTM-aist 1.1.2のRTC Builderを使用している場合は、変数初期化部分を修正する必要があります。(OpenRTM-aist 1.2.0では修正される予定です)

まずは、init関数のself._d_in変数初期化部分を修正してください。

```
def __init__(self, manager):
    OpenRTM_aist.DataFlowComponentBase.__init__(self, manager)

    #in_arg = [None] * ((len(RTC._d_TimedShortSeq) - 4) / 2)    ←削除
    #self._d_in = RTC.TimedShortSeq(*in_arg)    ←削除
    #以下の行を追加
    self._d_in = RTC.TimedShortSeq(RTC.Time(0,0),[])
```

次にself._d_out 変数初期化部分を修正してください。

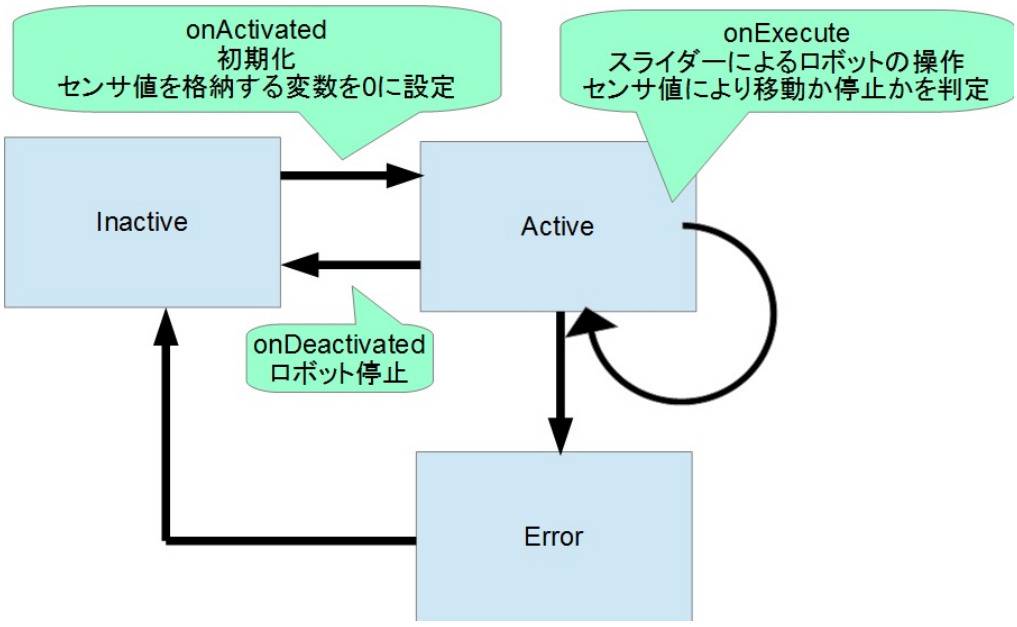
```
    #out_arg = [None] * ((len(RTC._d_TimedVelocity2D) - 4) / 2)    ←削除
    #self._d_out = RTC.TimedVelocity2D(*out_arg)    ←削除
    #以下の行を追加
    self._d_out = RTC.TimedVelocity2D(RTC.Time(0,0),RTC.Velocity2D(0.0,0.0,0.0))
```

これで完了です。

アクティビティ処理の実装

RobotController コンポーネントでは、コンフィギュレーションパラメーター(speed_x、speed_y)をスライダーで操作しその値を目標速度としてアウトポート(out)から出力します。インポート(in) から入力された値を変数に格納して、その値が一定以上の場合は停止するようにします。

onActivated()、onExecute()、onDeactivated() での処理内容を下図に示します。



アクティビティ処理の概要

下記のように、onActivated()、onDeactivated()、onExecute() を実装します。

```
def onActivated(self, ec_id):
    #センサー値初期化
    self.sensor_data = [0,0,0,0]
    return RTC.RTC_OK
```

```
def onDeactivated(self, ec_id):
    #ロボットを停止する
    self._d_out.data.vx = 0
    self._d_out.data.va = 0
    self._outOut.write()
    return RTC.RTC_OK
```

```
def onExecute(self, ec_id):
    #入力データの存在確認
    if self._inIn.isNew():
        data = self._inIn.read()
        #この時点で入力データがm_inに格納される
        #入力データを別変数に格納
        self.sensor_data = data.data[:]
    #前進するときのみ停止するかを判定
    if self._speed_x[0] > 0:
        for d in self.sensor_data:
            #センサ値が設定値以上か判定
            if d > self._stop_d[0]:
                #センサ値が設定値以上の場合は停止
                self._d_out.data.vx = 0
                self._d_out.data.va = 0
                self._outOut.write()
                return RTC.RTC_OK

    #設定値以上の値のセンサが無い場合はコンフィギュレーションパラメータの値で操作
    self._d_out.data.vx = self._speed_x[0]
    self._d_out.data.va = self._speed_r[0]
    self._outOut.write()

    return RTC.RTC_OK
```

RobotController コンポーネントの動作確認

作成した RobotController をシミュレーターコンポーネントと接続して動作確認を行います。

以下より RaspberryPiMouseSimulator コンポーネントをダウンロードしてください。

- RTM_Tutorial_2017

インターネットに接続できない環境で講習会を実施している場合がありますので、その場合は配布のUSBメモリーに入れてあります。

NameService の起動

コンポーネントの参照を登録するためのネームサービスを起動します。

```
$ rtm-naming
```

RobotController コンポーネントの起動

RobotController コンポーネントを起動します。

RobotController¥build¥srcフォルダーの RobotControllerComp ファイルを実行してください。

```
$ RobotControllerComp
```

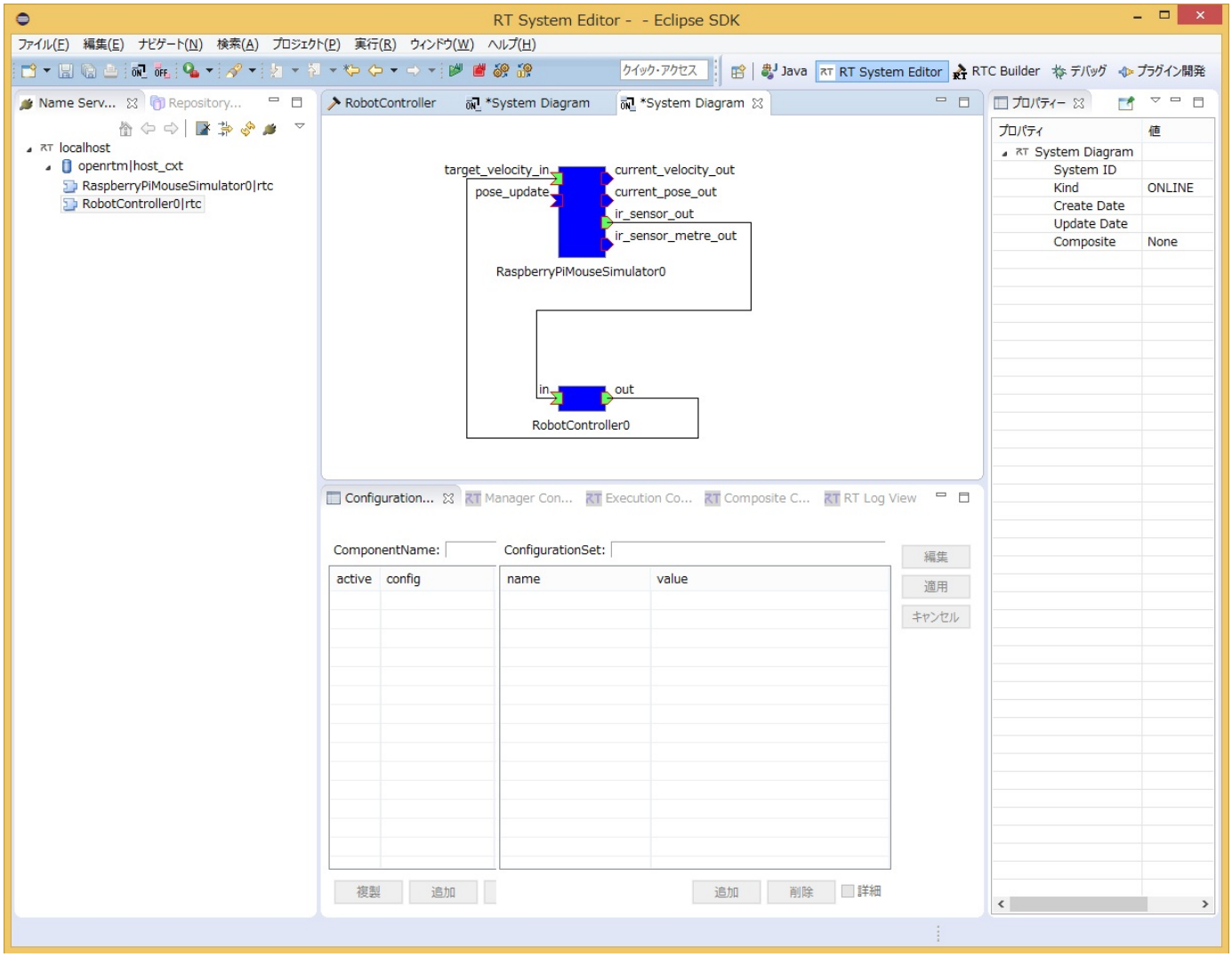
シミュレーターコンポーネントの起動

RaspberryPiMouseSimulator コンポーネントをインストールしたディレクトリーに移動後、下記のコマンドにて起動できます。

```
$ src/RaspberryPiMouseSimulatorComp
```

コンポーネントの接続

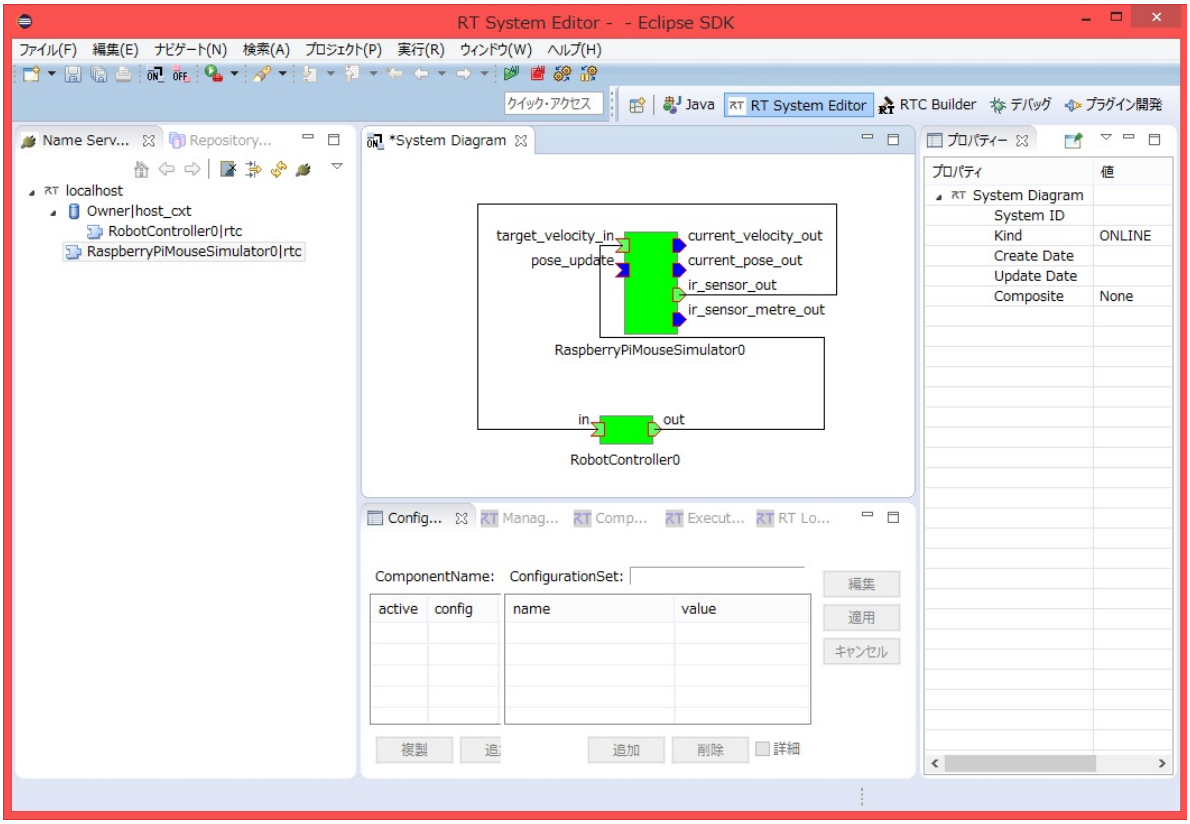
下図のように、RTSystemEditorにて RobotController コンポーネント、RaspberryPiMouseSimulator コンポーネントを接続します。



コンポーネントの接続

コンポーネントの Activate

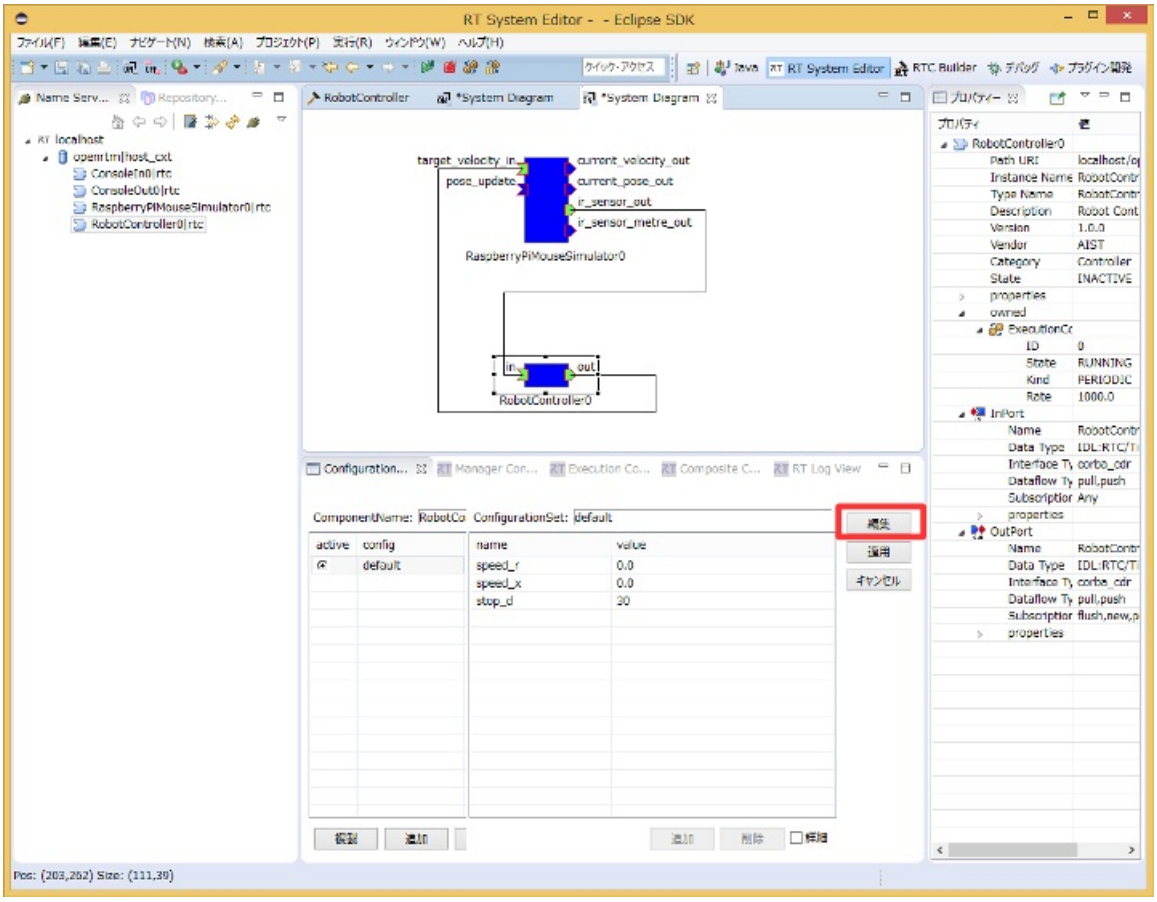
RTSystemEditor の上部にあります [All Activate] というアイコンをクリックし、全てのコンポーネントをアクティブ化します。正常にアクティベートされた場合、下図のように黄緑色でコンポーネントが表示されます。



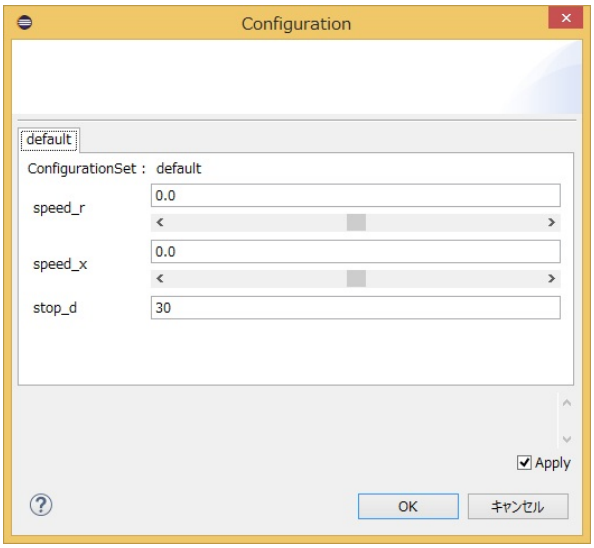
コンポーネントのアクティブ化

動作確認

下図のようにコンフィギュレーションビューの [編集] ボタンからコンフィギュレーションを変更することができます。



スライダーを操作してシミュレーター上の Raspberry Pi マウスの操作ができるかを確認してください。



コンフィギュレーションパラメーターの変更

実機での動作確認

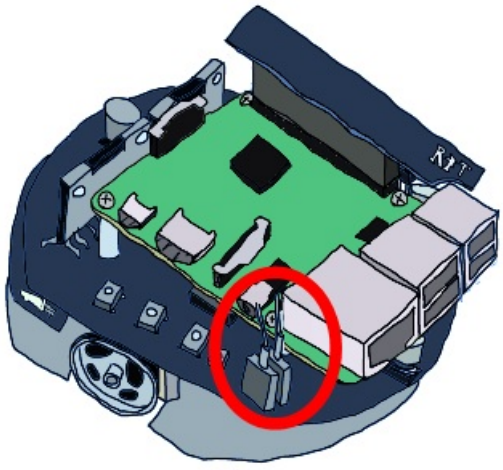
講習会で Raspberry Pi マウス実機を用意している場合は実機での動作確認が可能です。

手順は以下の通りです。

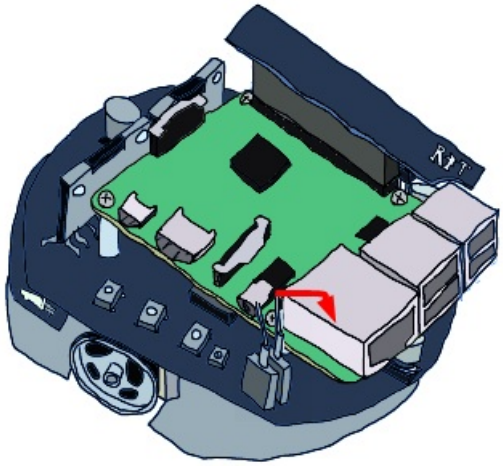
- Raspberry Pi マウスの電源を投入する
- Raspberry Pi マウスのアクセスポイントに接続
- ポートの接続
- コンポーネントのアクティブ化

電源を投入する

Raspberry PiマウスにはRaspberry Piの電源スイッチとモーターの電源スイッチの2つがあります。

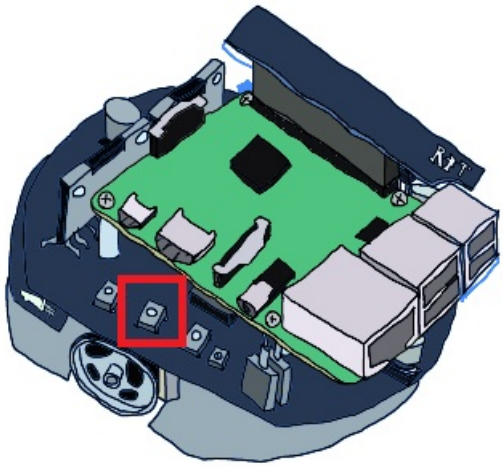


内側の電源スイッチをオンにするとRaspberry Piが起動します。



電源を切る場合

Raspberry Piの電源を切る場合は、電源スイッチから直接オフにはしないようにしてください。3つ並んだボタンの中央のボタンを数秒押すとシャットダウンが始まります。10秒程度でRaspbianのシャットダウンが終了するため、その後に電源スイッチをオフにしてください。



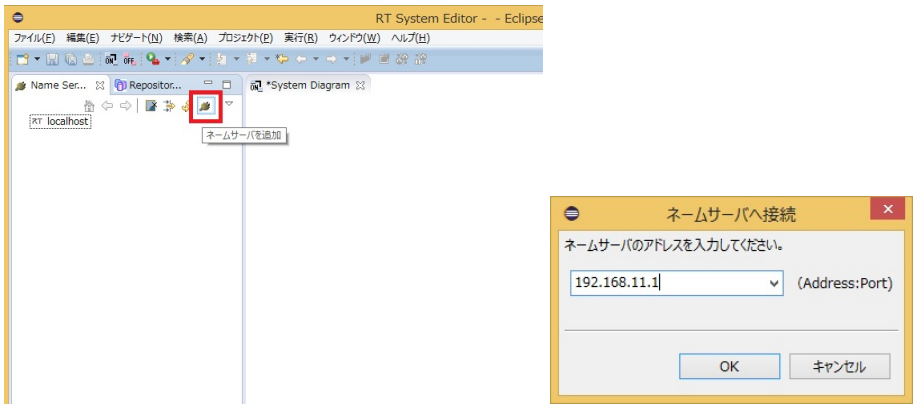
アクセスポイントに接続

SSID、パスワードは Rasoberry Pi マウスに貼り付けたシールに記載してあるので、その SSID に接続してください。

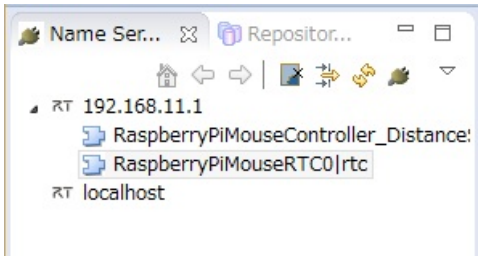
※ネットワークが切り替わった場合にネームサーバーへのコンポーネントの登録やポートの接続が失敗する場合があるのでネームサーバ、コンポーネントを一旦全て終了してください。ネットワーク切り替え後に起動した場合には問題ないので、終了させる必要はありません。

ネームサーバー追加

続いてRTシステムエディタの [ネームサーバー追加] ボタンで 192.168.11.1 を追加してください。



すると以下の2つの RTC が見えるようになります。

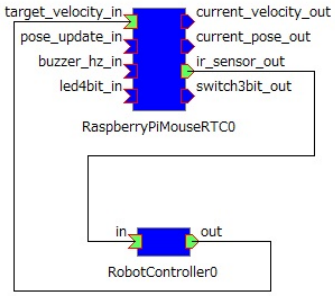


- RaspberryPiMouseRTC
- RaspberryPiMouseController_DistanceSensor

RaspberryPiMouseRTC は名城大学のロボットシステムデザイン研究室で開発されているラズパイマウス制御用の RTコンポーネントです。

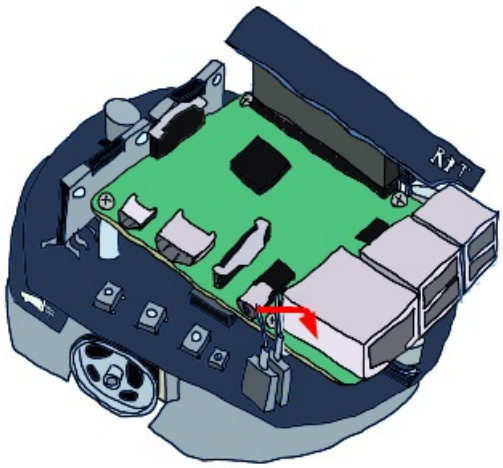
ポートの接続

RTシステムエディタで RaspberryPiMouseRTC、RobotController コンポーネントを以下のように接続します。



モーターの電源を投入する

動作の前に、モーターの電源スイッチをオンにしてください。モーターの電源はこまめに切るようにしてください。



アクティブ化

そして RTC をアクティブ化すると Raspberry Pi マウスの操作ができるようになります。

‹ チュートリアル(Raspberry Pi Mouse、Python、Windows、強化月間用)　↑ 上位　初期設定等 ›

Share / Save　印刷用ページ　コメントを投稿するには ログイン または ユーザー登録 を行ってください

最新バージョン

C++	1.1.2-RELESE
Java	1.1.2-RELESE
Python	1.1.2-RELESE
Tools	1.1.2

初めての方へ

Windows msi(インストーラ) パッケージ (サンプルの実行ができます。)

C++,Python,Java, Toolsを含む	1.1.2-RELEASE

RTコンポーネントを開発するためには開発環境のインストールが必要です。詳細はダウンロードページへ

統計

Webサイト統計

ユーザ数:	1654

プロジェクト統計

RTコンポーネント	288
RTミドルウエア	21
ツール	20
文書・仕様書	1

旧Webサイト	OpenRTM.org旧Webサイト
OpenHRP3	動力学シミュレータ
Choreonoid	モーションエディタ/シミュレータ
OpenHRI	対話制御コンポーネント群

OpenRTP	統合開発プラットフォーム
産総研RTC集	産総研が提供するRTC集
TORK	東京オープンソースロボティクス協会
DAQ-Middleware	ネットワーク分散環境でデータ収集用ソフトウェアを容易に構築するためのソフトウェア・フレームワーク
VirCA	遠隔空間同士を接続し、実験を行うことが可能な仮想空間プラットホーム

Navigation

- ホーム
- ダウンロード
- ドキュメント
- コミュニティ
- プロジェクト
- ハードウエア

News

13 Nov Linux用のOpenRTP 1.2.0 をリリー... 6 Nov 第12回AIツール入門講座(東京・新...
24 Oct Ubuntu 17.10 用パッケージ(C++,... 16 Oct Fedora のOpenRTM-aist-1.1.2 (C+...
11 Oct iREX2017講習会申し込み受け付け開始 10 Oct Debian9 のOpenRTM-aist (C++/Pyt...

About us

サイトポリシー
サービス利用規約
個人情報保護方針

ユーザー名 *

パスワード *

- アカウントの作成
- パスワードの再発行

ログイン

旧Webサイト

Copyright © 2005-2012 National Institute of Advanced Industrial Science and Technology, Japan. All Rights Reserved.