

Processing実習

宮本 信彦

国立研究開発法人産業技術総合研究所
インダストリアルCPS研究センター



資料

- 「WEBページ」フォルダのHTMLファイルを開く
 - Processing 活用事例 _ OpenRTM-aist.html
- もしくは以下のリンク
 - <https://openrtm.org/openrtm/ja/node/7232>



Processing 活用事例

👍 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

- Processingとは?
- 実習概要
- Processing開発環境の起動
- OpenRTM-aist Processing用ライブラリのインストール
- graficaのインストール
- プログラミング
- RTシステムの構築、動作確認

Processingとは?

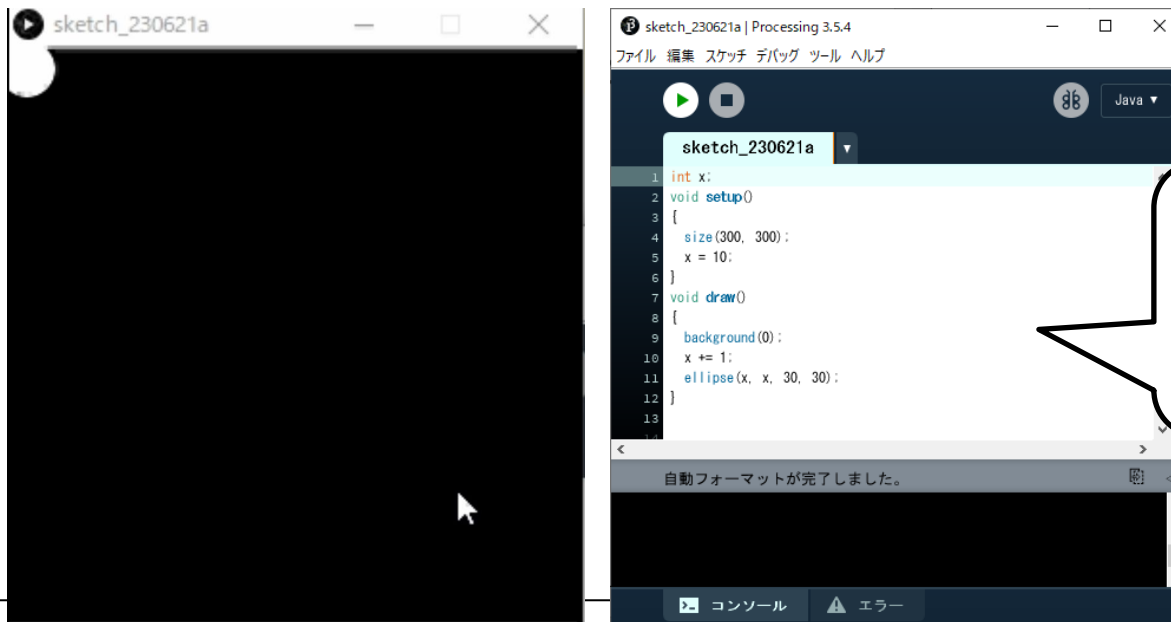
Processingはオープンソースのプログラミング言語で、以下の特長があることから初心者向けであるとされています。

- 視覚的な表現が他の言語と比較して簡単である。(グラフや図形のアニメーションやインタラクション等)
- 開発環境の導入が簡単

実習概要

プログラミング言語Processing

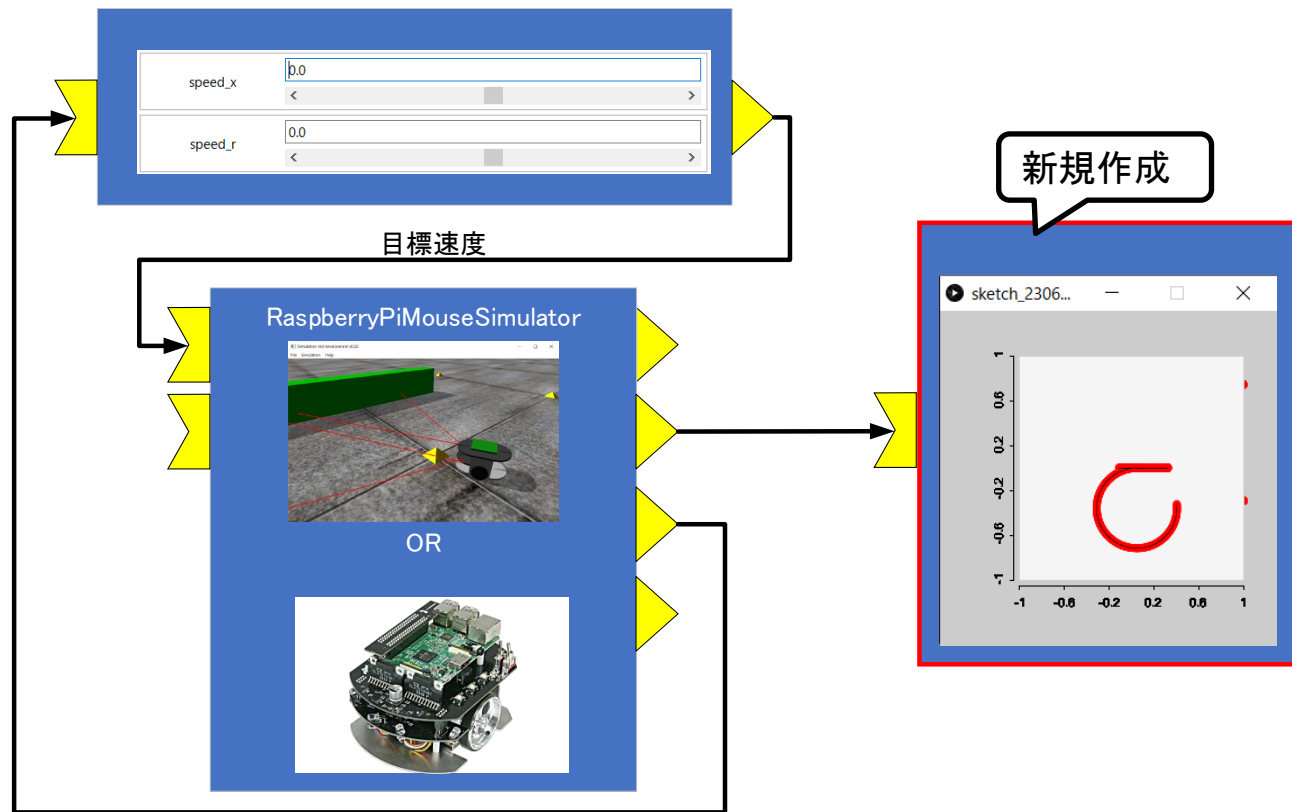
- 以下の特長を持つプログラミング言語
 - 視覚的な表現が他の言語と比較して簡単であり、初心者向け
 - グラフや図形のアニメーション
 - インタクション等



- 例えば、動画の図形を動かすアニメーションを作成する場合、12行のコードで記述できる。
- 実行する場合は統合開発環境の左上の実行ボタンを押す。

実習の概要

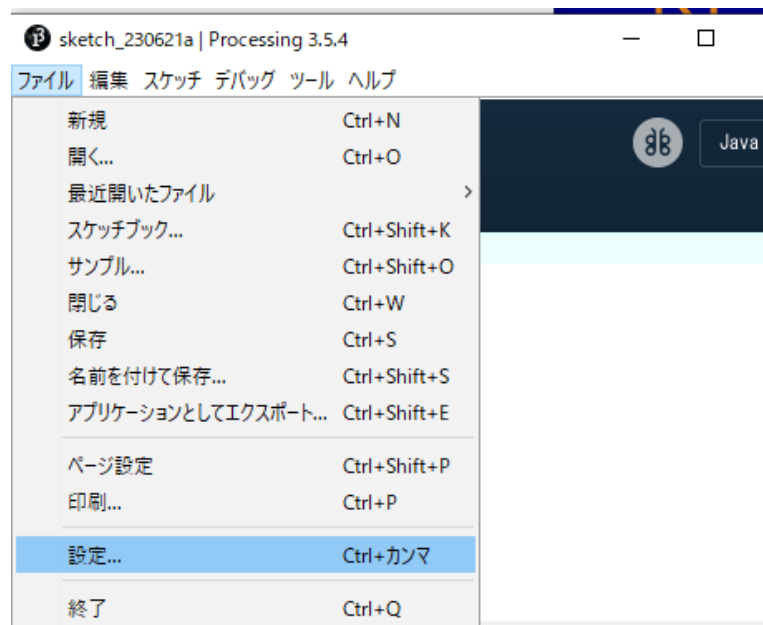
- Raspberry Piマウスの移動経路をグラフに描画するシステムの作成



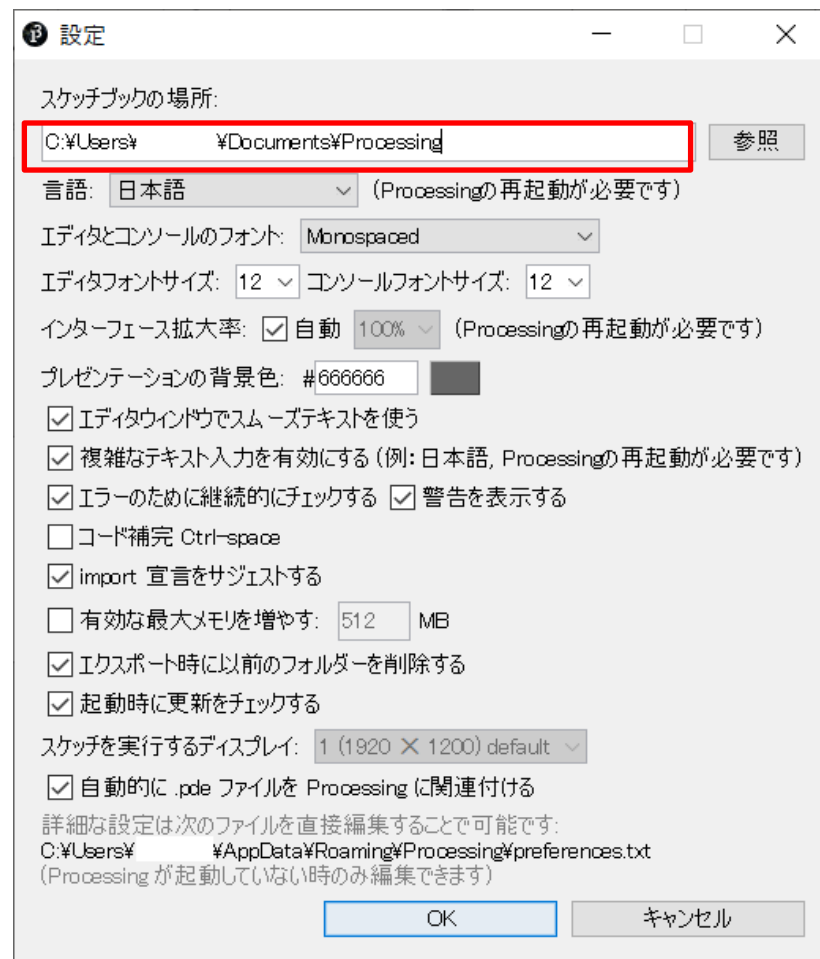
Processing開発環境の起動

- **Windows: processing.exe**を実行する
 - USBメモリのProcessing¥processing-3.5.4-windows64フォルダ内
 - Javaのバージョンの問題で、OpenRTM-aistはProcessing 4.0以降では現状は利用不可
- **Ubuntu: processing**を実行する
 - USBメモリのProcessing¥processing-3.5.4-linux64フォルダ内

OpenRTM-aist Processing用ライブラリのインストール



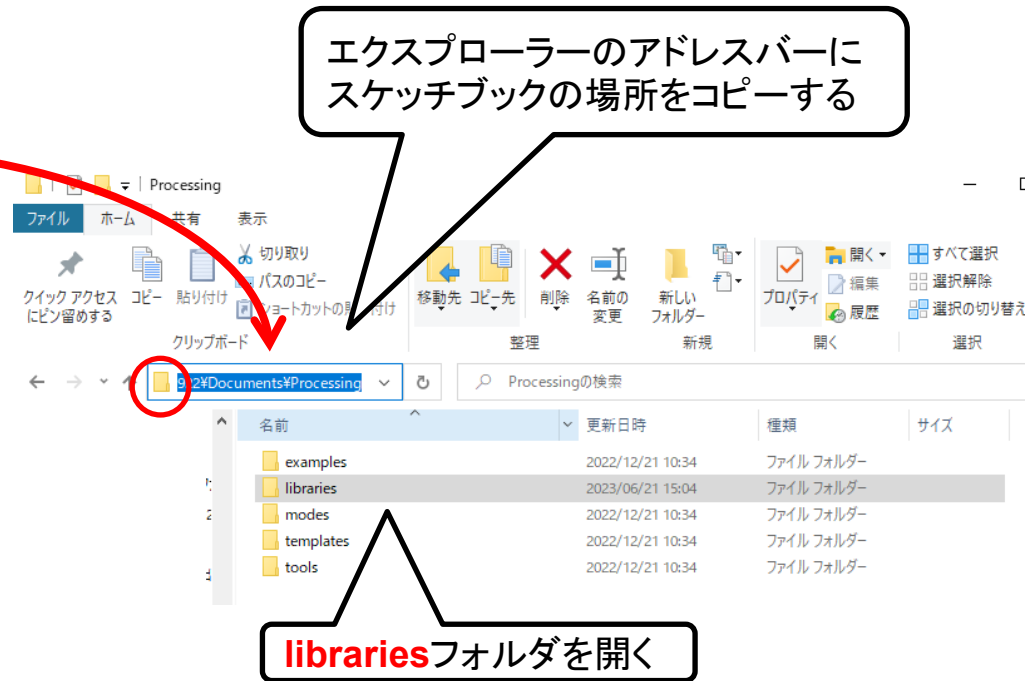
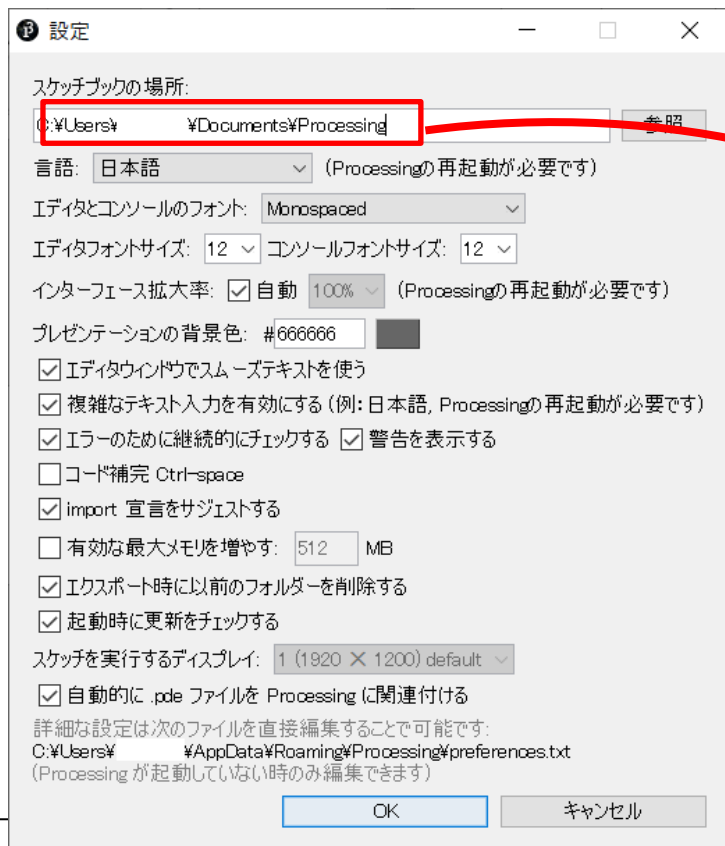
Processing開発環境で、「ファイル」→「設定」をクリックすると、設定画面が表示される



OpenRTM-aist

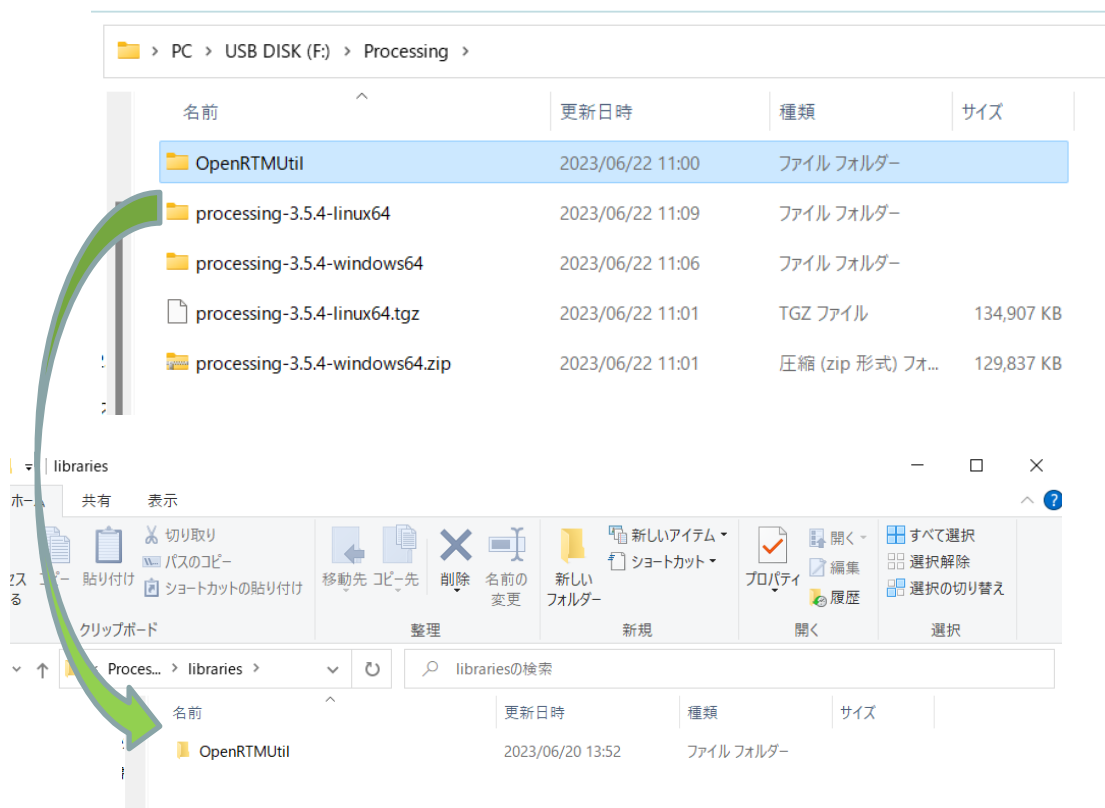
Processing用ライブラリのインストール

- スケッチブックの場所の **libraries** フォルダをエクスプローラーで開く



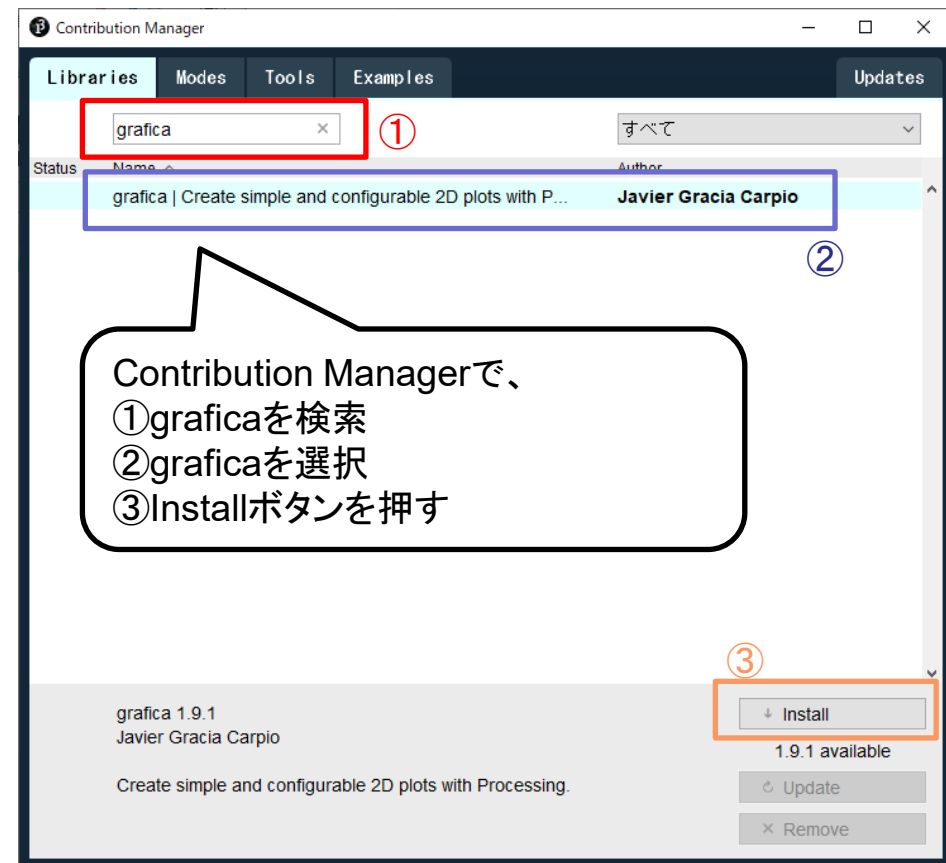
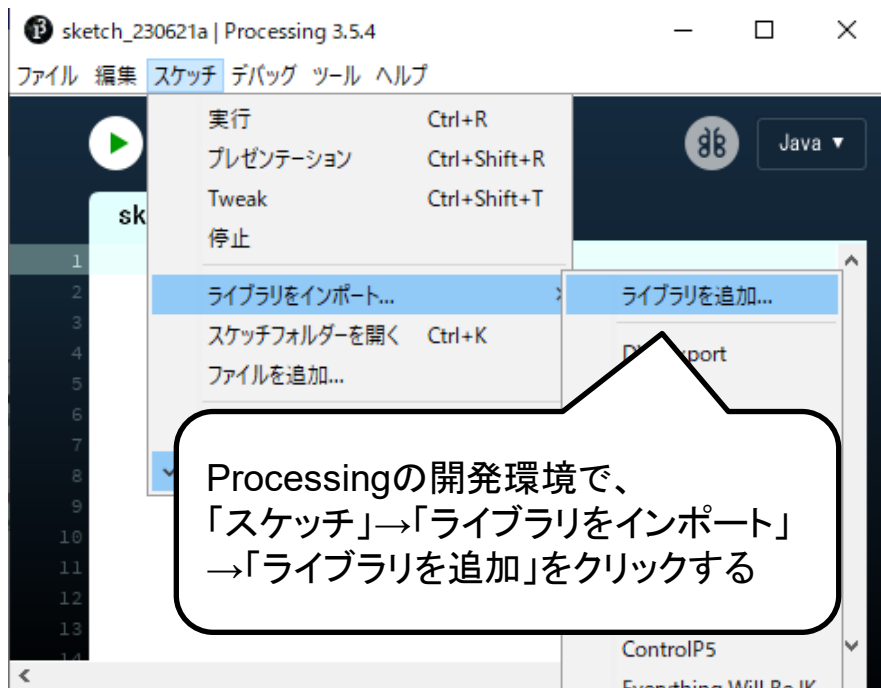
OpenRTM-aist Processing用ライブラリのインストール

- USBメモリのProcessing¥OpenRTMUtilフォルダをlibrariesフォルダ内にコピーする



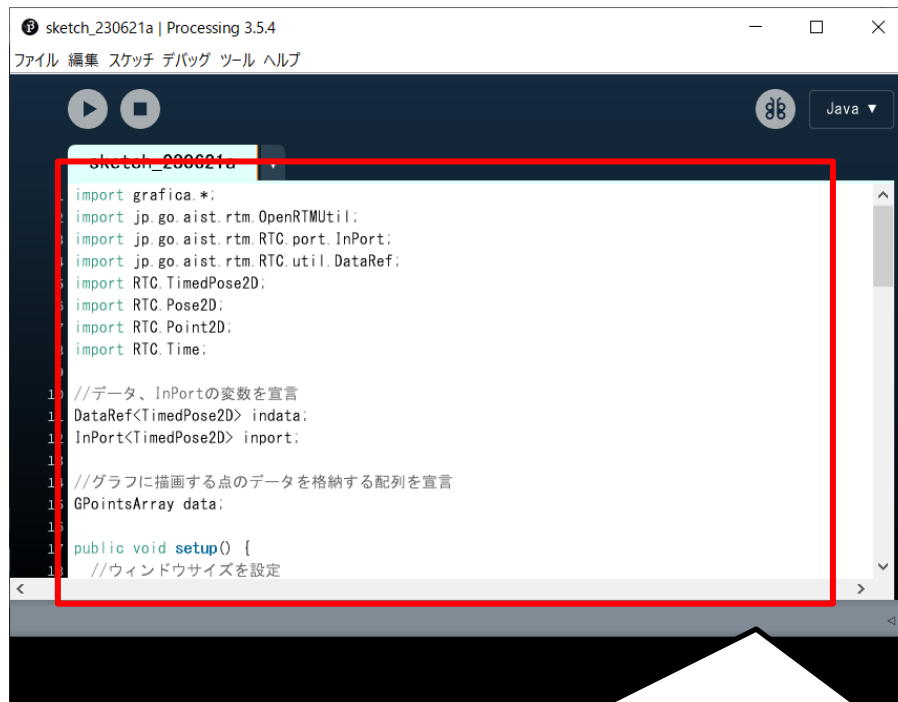
graficaのインストール

- グラフ描画用ライブラリのgraficaをインストールする



プログラミング

- Processing開発環境でコードを編集する



```
import grafica.*;
import jp.go.aist.rtm.OpenRTMUtil;
import jp.go.aist.rtm.RTC.port.InPort;
import jp.go.aist.rtm.RTC.util.DataRef;
import RTC.TimedPose2D;
import RTC.Pose2D;
import RTC.Point2D;
import RTC.Time;
```

```
//データ、InPortの変数を宣言
DataRef<TimedPose2D> indata;
InPort<TimedPose2D> inport;
```

```
//グラフに描画する点のデータを格納する配列を宣言
GPointsArray data;
```

スライド10、11、12ページに記載のソースコードを入力する。
USBメモリ内のsample¥drawGraph¥drawGraph.pdeの中身を
コピーしてもいいです。

プログラミング

```
//グラフに描画する点のデータを格納する配列を宣言
GPointsArray data;

public void setup() {
    //ウィンドウサイズを設定
    size(300, 300);

    //RTCを"drawGraph"というインスタンス名で生成
    OpenRTMUtil util = new OpenRTMUtil();
    util.createComponent("drawGraph");
    //データの初期化
    TimedPose2D val = new TimedPose2D();
    val.tm = new Time();
    val.data = new Pose2D();
    val.data.position = new Point2D();
    indata = new DataRef<TimedPose2D>(val);
    //InPortを"pose"という名前で生成
    inport = util.addInPort("pose", indata);

    //配列dataの初期化
    data = new GPointsArray();
}
```

Processingでsetup関数は実行開始時に1度だけ呼ばれる。
setup関数内でRTCの生成を行っている。

プログラミング

```
int count = 0;
public void draw() {

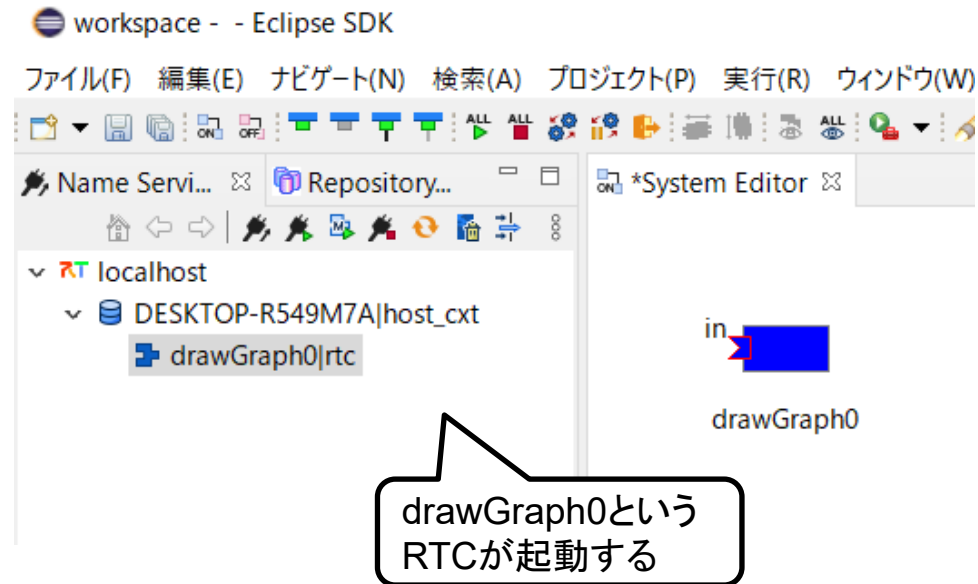
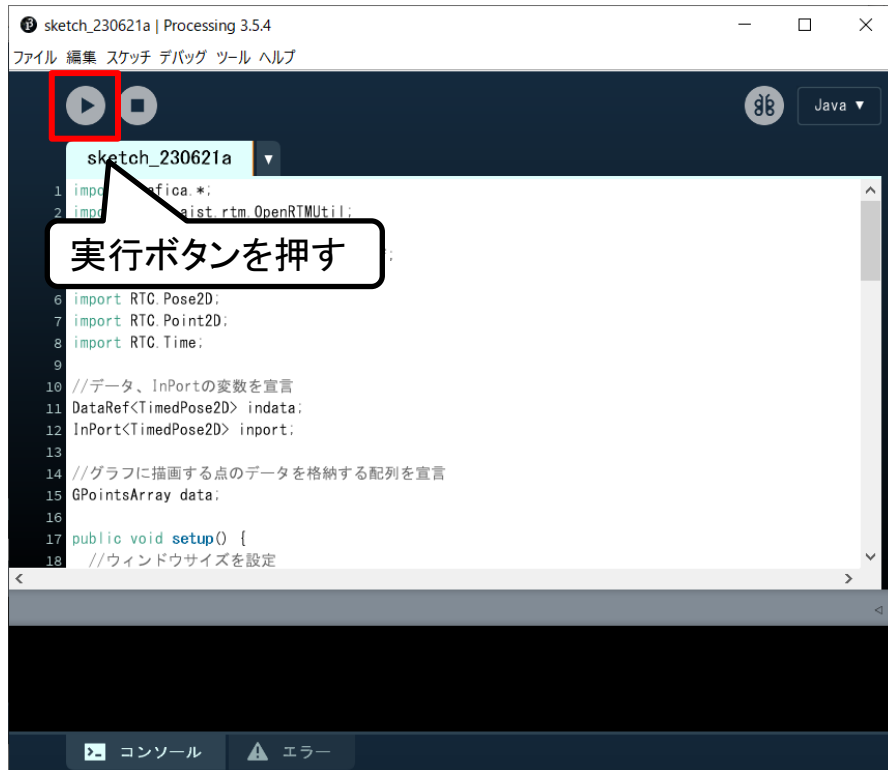
    //InPortでデータを受信した時の処理
    if (inport.isNew())
    {
        //受信データの読み込み
        inport.read();
        //配列dataに取得した位置を追加する
        data.add((float)indata.v.data.position.x,
        (float)indata.v.data.position.y);

        //配列の大きさが1000を超えた場合、
        //古いデータは捨てる
        if (data.getNPoints() > 1000)
        {
            data.remove(0);
        }
    }
}
```

Processingでdraw関数は一定間隔で呼ばれる。
InPortの読み込み処理とグラフの描画更新を行う。

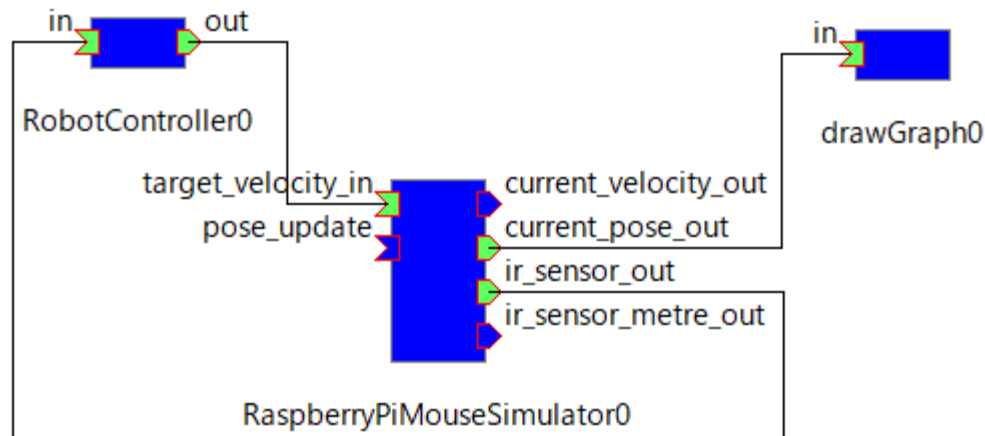
```
//グラフをウィンドウの(0,0)から(300,300)の範囲
//に描画する
GPlot plot = new GPlot(this, 0, 0, 300,
300);
//グラフの縦軸、横軸の上限、下限を設定する
plot.setXLim(-1.0, 1.0);
plot.setYLim(-1.0, 1.0);
plot.setFixedXLim(true);
plot.setFixedYLim(true);
//配列dataをグラフに設定する
plot.addPoints(data);
//グラフの描画を開始する
plot.beginDraw();
//グラフに外枠、座標、折れ線、縦軸、横軸を
//描画する
plot.drawBox();
plot.drawPoints();
plot.drawLines();
plot.drawXAxis();
plot.drawYAxis();
//グラフの描画を終了する
plot.endDraw();
}
```

RTシステム構築



RTシステム構築

- 以下のようにポートを接続して、RTCをアクティブ化する
 - シミュレータ(RaspberryPiMouseSimulator0)、実機(RaspberryPiMouseRTC0)のどちらでも可



RTシステム実行

