RT コンポーネント操作マニュアル

RTM on Android を用いた Android 用 マルチセンサコンポーネント群

2013 年 12 月 07 日版

芝浦工業大学　大学院理工学研究科
電気電子情報工学専攻

立川 将

更新履歴：

2013/11/27　　第一版

2013/11/28　　第二版（端末用マルチセンサ取得コンポーネントアプリケーションを追加）

2013/12/04　　第三版（端末用 GPS 取得コンポーネントアプリケーションを追加）

2013/12/07　　第四版（コンポーネント群の利用例を追加）

目次

1　本コンポーネント群の概要

1.1 開発の背景

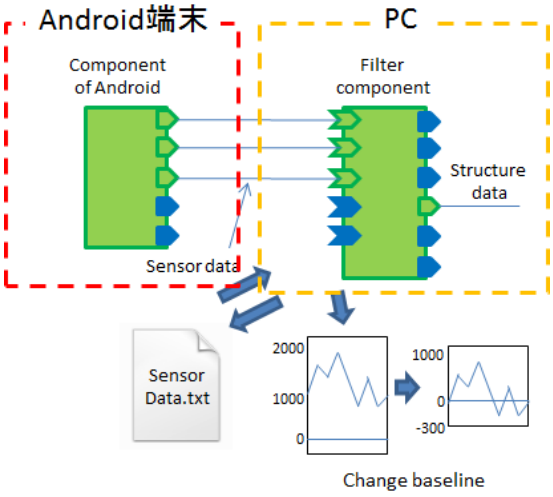
生産、研究、教育、生活、様々な場面での物事への自動化や機能付与にはセンサを使わなければ成立しないものが大半であり、人間には不可能の精度を実現するために、センサはとても重要な電子部品です。しかし、その重要さ故に同種類のセンサでも性能・価格・強度などは様々で、その多様さ故にどのセンサを使用していいのか迷うことも多いです。さらにはセンサの実装方法も様々で、時には実装するまでに想像以上に時間がかかってしまうこともあります。

このような問題を解決するため、我々はスマートフォンに搭載されているセンサ群を利用することはできないかと考えました。センサ技術及びマイクロコンピュータの性能の向上により、スマートフォンの高性能化は年々進んでおり、機種によっては市販で手に入る物よりもはるかに高い精度を持つセンサを搭載しているものもあります。そこで、RT(Robot Technology)と、RT を Android 端末上で使用できるアプリケーションとして(株)セック社のリリースした RTM on Android を利用することで、これらセンサ群の情報を簡単に取得するための開発支援コンポーネント群を開発する事といたしました。

1.2 マルチセンサコンポーネント群の機能と用途

本コンポーネント群は、Android 端末からセンサの値を取得し、その情報を様々な出力形式に変換を行います。センサ値に対する主要な型は所持しているため、従来のコンポーネントにもご利用いただけます。また、センサより取得した情報のノイズ情報をフィルタリングすることを目的としたコンポーネントには、以下の機能があります。

・取得した情報を元に、センサの情報として適した形式への変換
・取得したセンサ情報を外部テキストに出力及び、外部テキストの読み込み
・センサ情報に対する暴れ値の丸め処理
・基準値 0 の変更



1.3 開発環境

本コンポーネント群は Windows にて動作確認を行っています。

環境は以下の通りです。

・OS：Windows7 Home Premium Service Pack 1 (32bit 及び 64bit)

・RT ミドルウェア：OpenRTM-aist-1.1.0-RC1(Java)

・Tools：OpenRTP 1.1.0-RC4(全部入りパッケージ Eclipse-3.8 ベース)

+ Eclipse Java EE IDE for Web Developers.(Version: 2.0.0.)

+ Android Development Toolkit(Version: 22.2.1.v)

+ Android SDK Tools, revision 7

・Java 実行環境：Oracle Java SE Version 7 Update45

また、開発したコンポーネントのうち、Android 端末上で使用するコンポーネントアプリケーション全てに依存ライブラリとして RTM.jar を利用しています。

・RTM on Android-1.0.2

1.4 開発コンポーネント群

センサ値取得用コンポーネント Android アプリケーションと、他コンポーネントとのスムーズな接続を目的とするセンサ情報フィルタ用コンポーネントとして、以下のコンポーネント群を開発しました。

- RTM on Android を利用した端末センサ値取得用コンポーネント(.apk)
  - 端末加速度取得コンポーネントアプリケーション(GetAcceleration)
    - 端末の三軸それぞれの加速度を取得するコンポーネント
      - TYPE_ACCELEROMETER
  - 端末線形加速度取得コンポーネントアプリケーション(GetLinearAcceleration)
    - 重力加速度を除いた三軸それぞれの加速度値を取得するコンポーネント
      - TYPE_LINEAR_ACCELERATION
  - 端末ジャイロ取得コンポーネントアプリケーション(GetGyroscope)
    - 端末のピッチングの角速度を取得するコンポーネント
      - TYPE_GYROSCOPE
  - 端末重力加速度取得コンポーネントアプリケーション(GetGravity)
    - 重力加速度の三軸それぞれの値を取得するコンポーネント
      - TYPE_GRAVITY
  - 端末磁気取得コンポーネントアプリケーション(GetMagneticField)
    - 端末にかかる磁気を三軸それぞれで取得するコンポーネント
      - TYPE_MAGNETIC_FIELD
  - 端末傾き取得コンポーネントアプリケーション(GetOrientation)
    - 端末のピッチングの回転角度を取得するコンポーネント

- 端末回転ベクトル取得コンポーネントアプリケーション(GetRotationVector)
  - 端末の回転ベクトル及び、余弦値、センサ精度を取得するコンポーネント
    - TYPE_ROTATION_VECTOR
- 端末照度取得コンポーネントアプリケーション(GetLight)
  - 端末の表(メインモニター側)への明るさを取得するコンポーネント
    - TYPE_LIGHT
- 端末圧力取得コンポーネントアプリケーション(GetPressure)
  - 端末にかかる空気圧を取得するコンポーネント
    - TYPE_PRESSURE
- 端末近接値取得コンポーネントアプリケーション(GetProximity)
  - 端末の表への近接値を取得するコンポーネント
    - TYPE_PROXIMITY
- 端末温度取得コンポーネントアプリケーション(GetAmbientTemperature)
  - 端末周辺の温度を取得するコンポーネント
    - TYPE_AMBIENT_TEMPERATURE
- 端末マルチセンサ取得コンポーネントアプリケーション(GetSensors)
  - 上記センサの全てを取得することが出来るコンポーネント
- 端末 GPS 取得コンポーネントアプリケーション(GetGPS)
  - 端末の GPS を取得するコンポーネント

- センサ情報フィルタ用コンポーネント
  - 三軸加速度フィルタコンポーネント(AccelerationFilter)
    - 三軸加速度値を元に Acceleration3D 及び 2D を生成し出力する
  - 三軸線形加速度フィルタコンポーネント(LinearAccelerationFilter)
    - 三軸加速度値を元に Acceleration3D 及び 2D を生成し出力する
  - 三軸ジャイロフィルタコンポーネント(GyroscopeFilter)
    - 三軸回転速度を元に AngularVelocity3D を生成し出力する
  - 三軸重力加速度フィルタコンポーネント(GravityFilter)
    - 三軸重力加速度値を元に Acceleration3D を生成し出力する
  - 三軸磁気フィルタコンポーネント(MagneticFieldFilter)
    - 三軸磁気値を元に Vector3D を生成し出力する
  - 三軸傾きフィルタコンポーネント(OrientationFilter)
    - ピッチングの角度を元に Orientation3D を生成し出力する
  - 三軸回転ベクトルフィルタコンポーネント(RotationVectorFilter)
    - ピッチングのベクトルを元に Vector3D を生成し出力する
  - 照度フィルタコンポーネント(LightFilter)

- 圧力フィルタコンポーネント(PressureFilter)
- 近接値フィルタコンポーネント(ProximityFilter)
- 周辺温度フィルタコンポーネント(AmbientTemperatureFilter)
- GPS フィルタコンポーネント(GPSFilter)

それぞれのコンポーネントの詳細については 2 章を、使用方法については 3 章を参照してください。

2 コンポーネントの説明

2.1 Android 端末用コンポーネントアプリケーション

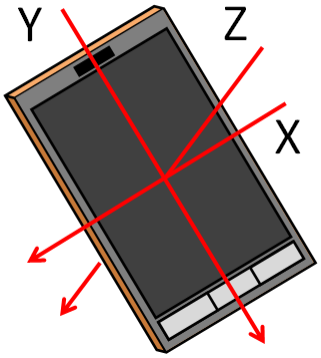
Android 端末用コンポーネント群は、Android 上で実行できるように.apk という Android 端末上で実行可能のアプリケーションにエクスポートしてあります。これらのアプリケーションに対するキーストアの有効期限は 2013 年より 25 年となっています。

2.1.1 端末加速度取得コンポーネントアプリケーション(GetAcceleration)

GetAcceleration は、アクティブ時に Android 端末上の三軸加速度センサの値を Acceleration3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoubleX	TimedDouble	三軸加速度センサの X 軸の加速度。単位[m/s^2] X 軸は端末の右側から左側へ向けて正方向
OutDoubleY	TimedDouble	三軸加速度センサの Y 軸の加速度。単位[m/s^2] Y 軸は端末の上部から下部へ向けて正方向
OutDoubleZ	TimedDouble	三軸加速度センサの Z 軸の加速度。単位[m/s^2] Z 軸は端末の正面から裏へ向けて正方向

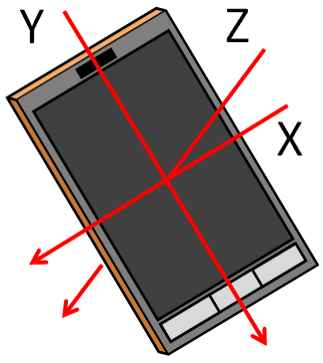
- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.2 端末線形加速度取得コンポーネントアプリケーション(GetLinearAcceleration)

GetLinearAcceleration は、アクティブ時に Android 端末上の三軸線形加速度センサの値を Acceleration3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し

- OutPort

名称	型	説明
OutDoubleX	TimedDouble	三軸線形加速度センサの X 軸の加速度[m/s^2] X 軸は端末の右側から左側へ向けて正方向
OutDoubleY	TimedDouble	三軸線形加速度センサの Y 軸の加速度[m/s^2] Y 軸は端末の上部から下部へ向けて正方向
OutDoubleZ	TimedDouble	三軸線形加速度センサの Z 軸の加速度[m/s^2] Z 軸は端末の正面から裏へ向けて正方向

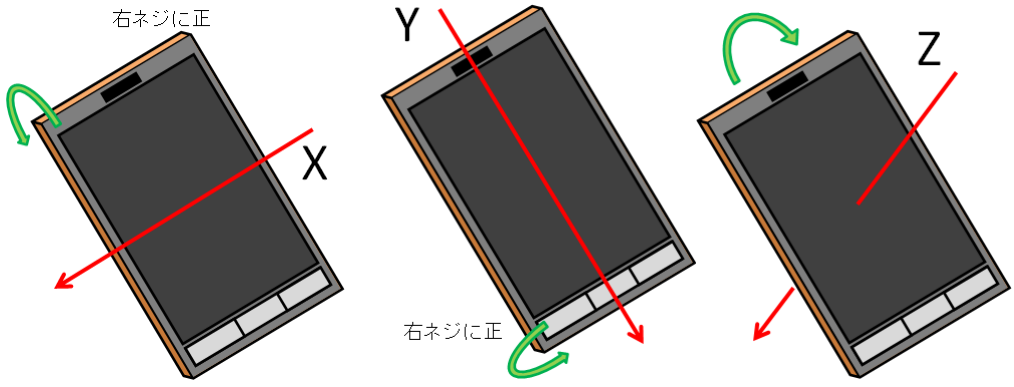
- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.3 端末ジャイロ取得コンポーネントアプリケーション(GetGyroscope)

GetGyroscope は、アクティブ時に Android 端末上の三軸ジャイロセンサの値を AngleVelocity3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort

➢ 無し

- OutPort

名称	型	説明
OutDoublePitch	TimedDouble	ジャイロセンサの X 軸回りの回転速度［m/s］ X 軸は端末の右側から左側へ向けて正方向
OutDoubleRoll	TimedDouble	ジャイロセンサの Y 軸回りの回転速度［m/s］ Y 軸は端末の上側から下側へ向けて正方向
OutDoubleYaw	TimedDouble	ジャイロセンサの Z 軸回りの回転速度[m/s] Z 軸は端末の正面から裏へ向けて正方向

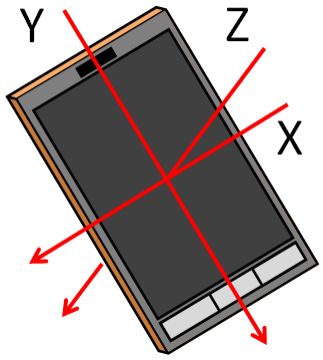
- Configuration 変数
  - ➢ 無し
- サービスポート
  - ➢ 無し

2.1.4 端末重力取得コンポーネントアプリケーション(GetGravity)

GetGravity は、アクティブ時に Android 端末上の重力センサの値を Acceleration3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoubleX	TimedDouble	三軸重力センサの X 軸方向に働く重力[m/s^2] X 軸は端末の右側から左側へ向けて正方向
OutDoubleY	TimedDouble	三軸重力センサの Y 軸方向に働く重力[m/s^2] Y 軸は端末の上部から下部へ向けて正方向
OutDoubleZ	TimedDouble	三軸重力センサの Z 軸方向に働く重力[m/s^2] Z 軸は端末の正面から裏へ向けて正方向

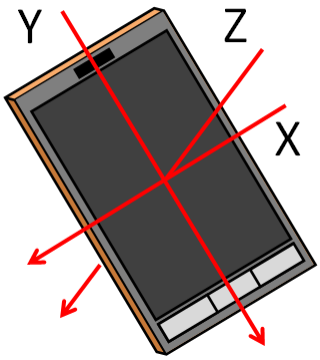
- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.5 端末磁気取得コンポーネントアプリケーション(GetMagneticField)

GetMagneticField は、アクティブ時に Android 端末上の三軸磁気センサの値を Vector3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoubleX	TimedDouble	三軸磁気センサの X 軸方向にかかる電磁強度 [uT] X 軸は端末の右側から左側へ向けて正方向
OutDoubleY	TimedDouble	三軸磁気センサの Y 軸方向にかかる電磁強度 [uT] Y 軸は端末の上部から下部へ向けて正方向
OutDoubleZ	TimedDouble	三軸磁気センサの Z 軸方向にかかる電磁強度 [uT] Z 軸は端末の正面から裏へ向けて正方向

- Configuration 変数
  - 無し
- サービスポート
  - 無し

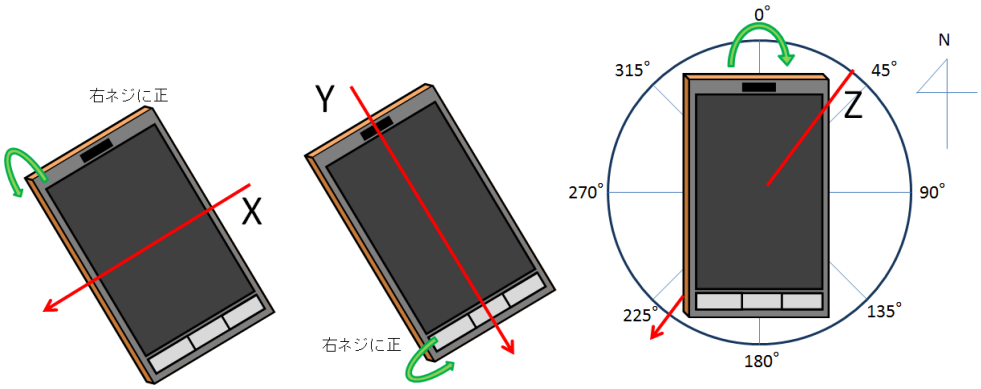
2.1.6 端末傾き取得コンポーネントアプリケーション(GetOrientation)

GetOrientation は、アクティブ時に Android 端末上の傾きセンサの値を Orientation3D の元となる Double 型で取得し、方位角、傾斜角、回転角の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を方位、傾斜、回転それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

基本情報取得には傾きセンサ(TYPE_ORIENTATION)を使用していますが、搭載していない機種の場合には、加速度センサ・磁気センサの情報を取得し、センサ情報を元に回転行列を作成し傾きを取得します。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoubleAzimuth	TimedDouble	傾きセンサの方位角[deg] 方位は端末の Z 軸回り
OutDoublePitch	TimedDouble	傾きセンサの傾斜角[deg] 傾斜角は端末の X 軸回り
OutDoubleRoll	TimedDouble	傾きセンサの回転角[deg] 回転角は端末の Y 軸回り

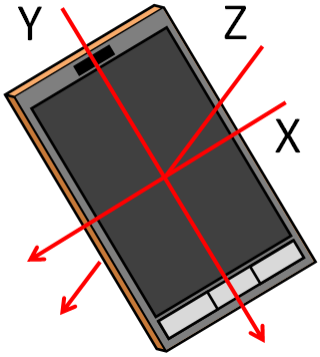
- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.7 端末回転ベクトル取得コンポーネントアプリケーション(GetRotationVector)

GetRotationVector は、アクティブ時に Android 端末上の回転ベクトルセンサの値を Vector3D の元となる Double 型で取得し、X 軸 Y 軸 Z 軸の情報をそれぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を X 軸 Y 軸 Z 軸それぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

座標軸



※座標軸の設定は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoubleX	TimedDouble	回転ベクトルセンサの X 軸方向のベクトル
OutDoubleY	TimedDouble	回転ベクトルセンサの Y 軸方向のベクトル
OutDoubleZ	TimedDouble	回転ベクトルセンサの Z 軸方向のベクトル
OutDoubleCos	TimedDouble	回転ベクトルセンサの余弦値
OutDoubleAccuracy	TimedDouble	回転ベクトルセンサの精度

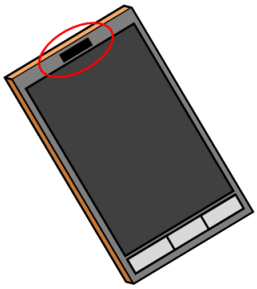
- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.8 端末照度取得コンポーネントアプリケーション(GetLight)

GetLight は、アクティブ時に Android 端末上の照度センサの値を Float 型で取得し出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

センサ位置



※照度センサの位置は機種によって違う場合がございます

- InPort
  - 無し
- OutPort

名称	型	説明
OutDouble	TimedDouble	照度センサから取得した明るさ[Lux] 照度センサは正面上部にあることが多い

- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.9 端末圧力取得コンポーネントアプリケーション(GetPressure)

GetPressure は、アクティブ時に Android 端末上の圧力センサの値を Float 型で取得し出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoublet	TimedDouble	圧力センサから取得した気圧[hPa]

- Configuration 変数
  - 無し
- サービスポート

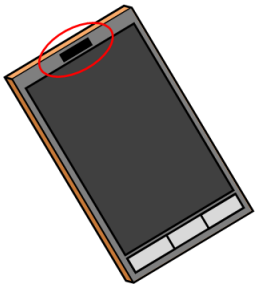
➢ 無し

2.1.10 端末近接値取得コンポーネントアプリケーション(GetProximity)

GetProximity は、アクティブ時に Android 端末上の近接センサの値を Float 型で取得し出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリスト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

センサ位置



※照度センサの位置は機種によって違う場合がございます

- InPort
  - ➢ 無し
- OutPort

名称	型	説明
OutDouble	TimedDouble	近接センサから取得した距離[cm] 近接センサは正面上部にあることが多い

- Configuration 変数
  - ➢ 無し
- サービスポート
  - ➢ 無し

2.1.11 端末温度取得コンポーネントアプリケーション(GetAmbientTemperature)

GetAmbientTemperature は、アクティブ時に Android 端末上の照度センサの値を Float 型で取得し出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するためのリス

ト登録を行い、アクティブ化されるとタイムスタンプを押し、センサの情報を OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

- InPort
  - 無し
- OutPort

名称	型	説明
OutDoublet	TimedDouble	温度センサから取得した周辺温度

- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.12 端末マルチセンサ取得コンポーネントアプリケーション(GetSensors)

本コンポーネントは、他のセンサ取得用アプリケーションとは大きく違います。

~~RTMonAndroid には、コンポーネントアプリケーションがバックグラウンド実行になってしまうと、OutPort の更新が行われなくなる（センサの更新が行われない）というデメリットがあります。つまり、他のセンサ取得用コンポーネントと併用して使うことができない、ということになります。~~（現在は、バックグラウンド時でも情報の取得が可能です）

そこで、本コンポーネントでは、アプリケーションの機動と共に、その端末で使用可能なセンサを一覧として表示し、ユーザへ使用したいセンサの選択を促します。（複数選択可能です）選択後に決定を行い、ネームサーバとの接続を行うと、「選択されたセンサ用の OutPort のみを所持する」コンポーネントを作成することができます。

アクティブ化されるとタイムスタンプを押し、センサの情報を OutPort に格納し、他のコンポーネントへ送ります。センサの値は SensorEventListener で常時確認され、センサの値に変化があった場合はイベント処理によりそのセンサ値を更新します。本アプリケーションでのセンサの確認周期は SENSOR_DELAY_GAME(20ms)で行っています。

本コンポーネントの詳しい操作方法は 3.

- InPort
  - 無し
- OutPort
  - 選択したセンサ用の取得コンポーネントアプリケーションと同じものになります。
  - 複数選択された場合には、本マニュアルの章順と同じ並び順で、OutPortg が作成されます。

- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.1.13 端末 GPS 取得コンポーネントアプリケーション(GetGPS)(new)

GetGPS は、アクティブ時に Android 端末上の GPS の値を GPSData の元となる Latitude・Longitude・Altitude・Error・Heading・Speed・GPSTime の情報を取得し、それぞれバラバラに出力するコンポーネントです。

本コンポーネントは、アプリケーションを起動するとセンサ情報を取得するため `LocationEventListener` に登録を行い、アクティブ化されるとタイムスタンプを押し、GPS の `Location` 上の情報をそれぞれ別々の OutPort に格納し、他のコンポーネントへ送ります。センサの値は `LocationEventListener` で常時確認され、GPS の値に変化があった場合はイベント処理によりその GPS の値を更新します。

- InPort
  - 無し
- OutPort

名称	型	説明
DoubleLatitude	TimedDouble	GPS の緯度情報 測定できていない場合は 0.0 を出力 ※Time 情報は RTC の Time を使用
DoubleLongitude	TimedDouble	GPS の経度情報 測定できていない場合は 0.0 を出力 ※Time 情報は GPSTime を使用
DoubleAltitude	TimedDouble	GPS の標高情報 測定できていない場合は 0.0 を出力 ※Time 情報は RTC の Time を使用
DoubleError	TimedDouble	GPS の水平方向の精度情報 測定できていない場合は 0.0 を出力 ※Time 情報は RTC の Time を使用
DoubleHeading	TimedDouble	GSP の傾き情報 測定できていない場合は 0.0 を出力 ※Time 情報は RTC の Time を使用
DoubleSpeed	TimedDouble	GPS の速度情報 測定できていない場合は 0.0 を出力 ※Time 情報は RTC の Time を使用

- Configuration 変数
  - 無し
- サービスポート
  - 無し

2.2 センサ情報フィルタ用コンポーネント

フィルタ用コンポーネントにはデータ型の変換の他に以下の機能が搭載されています。

・InPort より取得した情報を外部テキストフォルダに書き出す。
・指定されたテキストファイルを読み込み、擬似的に InPort の入力にする。
・丸め範囲を元に、暴れ値の丸め処理を行う。
・ユーザーによって指定されたタイミングの出力値を基準値にして、その後は変更後の基準値との差分を出力する。

外部テキストを生成してデータを書き出すか、外部テキストからデータを読み込むかは、どちらか一方しか行われません。条件の分岐は、onActivated 時に Configuration の ImportTextName にテキストファイル指定されているかどうかで行われます。

ImportTextName が null か空の文字列か初期値の場合には、InPort の情報をテキストファイルに書き出します(以下 Write モード)。なお、ImportTextName は String 型で、初期値には初期値であることを明確にするために String データの”null”という文字列が入っています。

Write モードで書き出すテキストのファイル名は Get[センサ名]_[onActivate()された時間].txt となります。[センサ名]はフィルタの名前に依存し、[時間]部は、_[年_月(0×)_日(0×)_時間(24)_分].txt となります。月・日・時・分が一桁の場合は、ゼロパティングを行い名前によるソートがしやすいように変換しています。実際に書き込まれる String の内容は各フィルタの説明で改めて説明を行います。

次に、ファイルの読み込み(以下 Read モード)の説明を行います。onActive 時に ImportTextName に入力されたファイルを読み込みます。ファイルが見つからなかった場合は、特別に回避用処理を行っていないので、Error になりますので、ご注意ください。また、読み込むファイルは、本コンポーネントから生成されたテキストファイルを読み取ることに特化しているため、違うものから生成されたセンサ情報を読み取りたい場合には、本コンポーネントの生成 String 形式に変換してお使いください。なお、Read モードを従来のコンポーネントの周期で実行すると Android 端末との接続時に比べ、圧倒的な速さでデータを読み込んでしまうため、Configuration の PitchTime で実行周期を指定することをおすすめします。PitchTime の初期値は 1000msec となっています。

暴れ値丸め処理は、Configuration の VariableToRounding の値分の配列を保持し、その

平均値を出力データにしています。初期値は 1 が入っているため、実質的には丸め処理は行われていません。

VariableToRounding の値が[従来＞変更値]の場合、差分の部分は古い情報から削り、新しい配列を作成します。[従来＜変更値]の場合、足りない分のデータは null として補い、情報数が VariableToRounding の値に達するまでは、平均を取る時の母体数は配列に入っている情報数に依存します。

基準値変更は、radio ボタンである Configuration の ChangedOfBaseline が 1 になっている時に実行されます。基準値自体の変更タイミング及びその値は、radio ボタンの 1 が押された周期の入力データを用います。radio ボタンが 1 の時には常に変更後の基準値からの偏差を出力し続けるため、再度基準値を変更したい場合は、一度 radio ボタンを 0 に戻してから再度 1 を押してください。

フィルタで扱う Configuration 変数は以下になります。

名称	型	デフォルト値	説明
ImportTextName	String	“null”	テキストの読み込みの有無、読み込むファイル名の指定に使用する。
ChangedOfBaseline	Int	0	基準値変更のタイミングを知るために使用する。0 の時は何も処理せず、1 の時には 1 に変更された周の入力データを基準値配列として保持し、基準値からの偏差を丸め処理のデータに使用します。
VariableToRounding	Int	1	暴れ値丸め処理用配列の長さを決めるための変数。配列の長さにそのまましよするため、範囲は[x>0]となる。
PitchTime	Int	1000	Read モード時の実行周期を決めるために使用する。実行周期にそのまま使用するため範囲は[x>=0]となる。0 の場はコンポーネントの実行周期に依存する。時間の単位は[msec]

2.2.1 三軸加速度フィルタコンポーネント(AccelerationFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸加速度センサに適したデータ型の TimedAcceleration3D 型と、X 軸と Y 軸のデータを基にした TimedAcceleration2D 型、また X 軸 Y 軸 Z 軸各 TimedDouble 型とそれら三つのデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型のデータの出力ポ

ートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetAcceleration_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataX	TimedDouble	GetAcceleration から入力される端末の X 軸の加速度。単位[m/s^2]
InTimedDoubleDataY	TimedDouble	GetAcceleration から入力される端末の Y 軸の加速度。単位[m/s^2]
InTimedDoubleDataZ	TimedDouble	GetAcceleration から入力される端末の Z 軸の加速度。単位[m/s^2]

- OutPort

名称	型	説明
OutTimedAcceleration3D	TimedAcceleration3D	入力データの X 軸 Y 軸 Z 軸それぞれにフィルタ処理を行い、出力結果を Acceleration3D 型にされたもの。
OutTimedAcceleration2D	TimedAcceleration2D	入力データの X 軸 Y 軸それぞれにフィルタ処理を行い、出力結果を Acceleration2D 型にされたもの。
OutTimedDoubleDataX	TimedDouble	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataY	TimedDouble	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataZ	TimedDouble	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X 軸データ DoubleSeq[1]…Y 軸データ DoubleSeq[2]…Z 軸データ
OutTimedFloatDataX	TimedFloat	入力データの X 軸情報にフィルタ処

		理を行ったデータを出力
OutTimedFloatDataY	TimedFloat	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataZ	TimedFloat	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X 軸データ FloatSeq[1]…Y 軸データ FloatSeq[2]…Z 軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.2 三軸線形加速度フィルタコンポーネント(LinearAccelerationFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸線形加速度センサに適したデータ型の TimedAcceleration3D 型と、X 軸と Y 軸のデータを基にした TimedAcceleration2D 型、また X 軸 Y 軸 Z 軸各 TimedDouble 型とそれら三つのデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型のデータの出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetLinearAcceleration_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataX	TimedDouble	GetLinearAcceleration から入力される端末の X 軸の加速度。単位[m/s^2]
InTimedDoubleDataY	TimedDouble	GetLinearAcceleration から入力される端末の Y 軸の加速度。単位[m/s^2]
InTimedDoubleDataZ	TimedDouble	GetLinearAcceleration から入力される端末の Z 軸の加速度。単位[m/s^2]

- OutPort

名称	型	説明
OutTimedAcceleration3D	TimedAcceleration3D	入力データの X 軸 Y 軸 Z 軸それぞれにフィルタ処理を行い、出力結果を Acceleration3D 型にされたもの。
OutTimedAcceleration2D	TimedAcceleration2D	入力データの X 軸 Y 軸それぞれにフィルタ処理を行い、出力結果を Acceleration2D 型にされたもの。
OutTimedDoubleDataX	TimedDouble	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataY	TimedDouble	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataZ	TimedDouble	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X 軸データ DoubleSeq[1]…Y 軸データ DoubleSeq[2]…Z 軸データ
OutTimedFloatDataX	TimedFloat	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataY	TimedFloat	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataZ	TimedFloat	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X 軸データ FloatSeq[1]…Y 軸データ FloatSeq[2]…Z 軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.3 三軸ジャイロフィルタコンポーネント(GyroscopeFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸ジャイロセンサに適したデータ型の TimedAngularVelocity3D 型と、Roll,Pitch,Yaw 各 TimedDouble 型とそれら三つのデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型の

データの出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetGyroscope_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataPitch	TimedDouble	ジャイロセンサの X 軸回りの回転速度[m/s]X 軸は端末の右部から左部へ向けて正方向
InTimedDoubleDataRoll	TimedDouble	ジャイロセンサの Y 軸回りの回転速度 [m/s] Y 軸は端末の上側から下側へ向けて正方向
InTimedDoubleDataYaw	TimedDouble	ジャイロセンサの Z 軸回りの回転速度[m/s] Z 軸は端末の正面から裏へ向けて正方向

- OutPort

名称	型	説明
OutTimedAcceleration3D	TimedAngleVelocity3D	入力データの X 軸 Y 軸 Z 軸それぞれにフィルタ処理を行い、出力結果を AngleVelocity3D 型にされたもの。
OutTimedDoubleDataPitch	TimedDouble	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataRoll	TimedDouble	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataYaw	TimedDouble	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X 軸データ DoubleSeq[1]…Y 軸データ DoubleSeq[2]…Z 軸データ
OutTimedFloatDataPitch	TimedFloat	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataRoll	TimedFloat	入力データの Y 軸情報にフィルタ処

		理を行ったデータを出力
OutTimedFloatDataYaw	TimedFloat	入力データのZ軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X軸データ FloatSeq[1]…Y軸データ FloatSeq[2]…Z軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.4 三軸重力加速度フィルタコンポーネント(GravityFilter)

入力されたTimedDouble型に対し一連のフィルタ処理を行い三軸加速度センサに適したデータ型のTimedAcceleration3D型と、X軸Y軸Z軸各TimedDouble型とそれら三つのデータを配列としたTimedDoubleSeq型及びTimedFloat型とTimedFloatSeq型のデータの出力ポートを持ちます。Configurationの ChangeOfBaselineを0、VatiableToRoundingが1の場合は、InpPortの情報をそのままOutPortより出力することができます。

書き込むテキストファイルのファイル名は
“GetGravity_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込むStringデータは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”でsplitを行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataX	TimedDouble	GetGravityから入力される端末のX軸の重力加速度。単位[m/s^2]
InTimedDoubleDataY	TimedDouble	GetGravityから入力される端末のY軸の重力加速度。単位[m/s^2]
InTimedDoubleDataZ	TimedDouble	GetGravityから入力される端末のZ軸の重力加速度。単位[m/s^2]

- OutPort

名称	型	説明
OutTimedAcceleration3D	TimedAcceleration3D	入力データのX軸Y軸Z軸それぞれ

		にフィルタ処理を行い、出力結果を Acceleration3D 型にされたもの。
OutTimedDoubleDataX	TimedDouble	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataY	TimedDouble	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataZ	TimedDouble	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X 軸データ DoubleSeq[1]…Y 軸データ DoubleSeq[2]…Z 軸データ
OutTimedFloatDataX	TimedFloat	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataY	TimedFloat	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataZ	TimedFloat	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X 軸データ FloatSeq[1]…Y 軸データ FloatSeq[2]…Z 軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.5 三軸磁気フィルタコンポーネント(MagneticFieldFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸磁気センサに適したデータ型の TimedVector3D 型と、X 軸 Y 軸 Z 軸各 TimedDouble 型とそれら三つのデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型のデータの出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetMagneticField_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換

えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataX	TimedDouble	三軸磁気センサの X 軸方向にかかる電磁強度[uT] X 軸は端末の右側から左側へ向けて正方向
InTimedDoubleDataY	TimedDouble	三軸磁気センサの Y 軸方向にかかる電磁強度[uT] Y 軸は端末の上部から下部へ向けて正方向
InTimedDoubleDataZ	TimedDouble	三軸磁気センサの Z 軸方向にかかる電磁強度[uT] Z 軸は端末の正面から裏へ向けて正方向

- OutPort

名称	型	説明
OutTimedVector3D	TimedVector3D	入力データの X 軸 Y 軸 Z 軸それぞれにフィルタ処理を行い、出力結果を Vector3D 型にされたもの。
OutTimedDoubleDataX	TimedDouble	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataY	TimedDouble	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataZ	TimedDouble	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X 軸データ DoubleSeq[1]…Y 軸データ DoubleSeq[2]…Z 軸データ
OutTimedFloatDataX	TimedFloat	入力データの X 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataY	TimedFloat	入力データの Y 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataZ	TimedFloat	入力データの Z 軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X 軸データ FloatSeq[1]…Y 軸データ

	FloatSeq[2]…Z 軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.6 三軸傾きフィルタコンポーネント(OrientationFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸傾きセンサに適したデータ型の TimedOrientation3D 型と、X 軸 Y 軸 Z 軸各 TimedDouble 型とそれら三つのデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型のデータの出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetOrientation_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

なお、入力されるデータの単位は[degree]なので、[rad]にも変換をして Orientation3D 型には[degree]と[rad]の双方のデータを出力します。

- InPort

名称	型	説明
InTimedDoubleDataYaw	TimedDouble	傾きセンサの方位角[deg] 方位は端末の Z 軸回り
InTimedDoubleDataPitch	TimedDouble	傾きセンサの傾斜角[deg] 傾斜角は端末の X 軸回り
InTimedDoubleDataRoll	TimedDouble	傾きセンサの回転角[deg] 回転角は端末の Y 軸回り

- OutPort

名称	型	説明
OutTimedOrientation Degree3D	TimedOrientation3D	入力データの X 軸 Y 軸 Z 軸それぞれにフィルタ処理を行い、出力結果を Orientation3D 型にされたもの。
OutTimedOrientation	TimedOrientation3D	入力データを[degree]から[rad]に変

Rad3D		換し Orientation3D 型に格納。
OutTimedDoubleDataRoll	TimedDouble	入力データの Roll 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataPitch	TimedDouble	入力データの Pitch 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataYaw	TimedDouble	入力データの Yaw 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…Roll データ DoubleSeq[1]…Pitch データ DoubleSeq[2]…Yaw データ
OutTimedFloatDataRoll	TimedFloat	入力データの Roll 情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataPitch	TimedFloat	入力データの Pitch 情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataYaw	TimedFloat	入力データの Yaw 情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…Roll データ FloatSeq[1]…Pitch データ FloatSeq[2]…Yaw データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.7 三軸回転ベクトルフィルタコンポーネント(RotationVectorFilter)

入力された TimedDouble 型に対し一連のフィルタ処理を行い三軸回転ベクトルセンサに適したデータ型の TimedVector3D 型に X 軸 Y 軸 Z 軸を格納し、別途に X 軸 Y 軸 Z 軸、余弦、精度情報を TimedDouble 型にして出力。また、三軸のデータを配列とした TimedDoubleSeq 型及び TimedFloat 型と TimedFloatSeq 型のデータの出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetRotationVector_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Xdata:,****,Ydata:,****,Zdata:,****,Cos:,****.Accuracy:,****,Time(sec):,****,Time(nsec):,****,¥n”

となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleDataX	TimedDouble	回転ベクトルセンサのX軸方向のベクトル
InTimedDoubleDataY	TimedDouble	回転ベクトルセンサのY軸方向のベクトル
InTimedDoubleDataZ	TimedDouble	回転ベクトルセンサのZ軸方向のベクトル
InTimedDoubleDataCos	TimedDouble	回転ベクトルセンサの余弦値
InTimedDoubleAccuracy	TimedDouble	回転ベクトルセンサの精度

- OutPort

名称	型	説明
OutTimedVector3D	TimedVector3D	入力データのX軸Y軸Z軸それぞれにフィルタ処理を行い、出力結果をVector3D型にされたもの。
OutTimedDoubleDataX	TimedDouble	入力データのX軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataY	TimedDouble	入力データのY軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataZ	TimedDouble	入力データのZ軸情報にフィルタ処理を行ったデータを出力
OutTimedDoubleDataCos	TimedDouble	入力データの余弦値情報にフィルタ処理を行ったデータを出力
OutTimedDoubleAccuracy	TimedDouble	入力データの精度情報にフィルタ処理を行ったデータを出力
OutTimedDoubleSeq	TimedDoubleSeq	DoubleSeq[0]…X軸データ DoubleSeq[1]…Y軸データ DoubleSeq[2]…Z軸データ
OutTimedFloatDataX	TimedFloat	入力データのX軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataY	TimedFloat	入力データのY軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataZ	TimedFloat	入力データのZ軸情報にフィルタ処理を行ったデータを出力
OutTimedFloatDataCos	TimedFloat	入力データの余弦値情報にフィルタ

		処理を行ったデータを出力
OutTimedFloatAccuracy	TimedFloat	入力データの精度情報にフィルタ処理を行ったデータを出力
OutTimedFloatSeq	TimedFloat	FloatSeq[0]…X 軸データ FloatSeq[1]…Y 軸データ FloatSeq[2]…Z 軸データ

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.8 照度フィルタコンポーネント(LightFilter)

照度センサには適した型などは無いため、入力された TimedDouble 型に対し一連のフィルタ処理を行い TimedDouble 型と TimedFloat 型の出力ポートへ出力します。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetLight_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Data:,****, Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleData	TimedDouble	GetLight から入力される端末の照度情報。単位[Lux]

- OutPort

名称	型	説明
OutTimedDoubleData	TimedDouble	入力照度情報にフィルタ処理を行ったデータを出力
OutTimedFloatData	TimedFloat	入力照度情報にフィルタ処理を行ったデータを出力

- Configuration 変数
  - 2.2 の表に明記

- サービスポート
  - 無し

2.2.9 圧力フィルタコンポーネント(PuressureFilter)

圧力センサには適した型などは無いため、入力された TimedDouble 型に対し一連のフィルタ処理を行い TimedDouble 型と TimedFloat 型の出力ポートへ出力します。
Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetPressure_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Data:,****, Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleData	TimedDouble	GetPressure から入力される端末の照度情報。単位[hPa]

- OutPort

名称	型	説明
OutTimedDoubleData	TimedDouble	入力照度情報にフィルタ処理を行ったデータを出力
OutTimedFloatData	TimedFloat	入力照度情報にフィルタ処理を行ったデータを出力

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.10 近接フィルタコンポーネント(ProximityFilter)

近接センサには適した型などは無いため、入力された TimedDouble 型に対し一連のフィルタ処理を行い TimedDouble 型と TimedFloat 型の出力ポートへ出力します。
Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

書き込むテキストファイルのファイル名は
“GetProximity_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Data:,****, Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleData	TimedDouble	GetProximity から入力される端末の近接情報。単位[cm]

- OutPort

名称	型	説明
OutTimedDoubleData	TimedDouble	入力近接情報にフィルタ処理を行ったデータを出力
OutTimedFloatData	TimedFloat	入力近接情報にフィルタ処理を行ったデータを出力

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.11 温度フィルタコンポーネント(AmbientTemperatureFilter)

周辺温度センサには適した型などは無いため、入力された TimedDouble 型に対し一連のフィルタ処理を行い TimedDouble 型と TimedFloat 型の出力ポートへ出力します。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。
書き込むテキストファイルのファイル名は
“GetAmbientTemperature_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”
テキストファイルへ書き込む String データは
”Data:,****, Time(sec):,****,Time(nsec):,****,¥n”
となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明
InTimedDoubleData	TimedDouble	GetAmbientTemperature から入力される端末の周辺温度情報。

- OutPort

名称	型	説明
OutTimedDoubleData	TimedDouble	入力周辺温度情報にフィルタ処理を行ったデータを出力
OutTimedFloatData	TimedFloat	入力周辺温度情報にフィルタ処理を行ったデータを出力

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

2.2.12 GPS フィルタコンポーネント(GPSFilter)(new)

入力された TimedDouble 型に対し一連のフィルタ処理を行い GPS に適したデータ型の GPSData 型と、Latitude・Longitude・Altitude の三つを配列にした TimedDoubleSeq 型及び Latitude・Longitude・Altitude・Error・Heading・Speed の TimedDouble 型の出力ポートを持ちます。Configuration の ChangeOfBaseline を 0、VatiableToRounding が 1 の場合は、InpPort の情報をそのまま OutPort より出力することができます。

丸め処理及び、基準値変更に関しては、Latitude・Longitude・Altitude のデータのみに適応されるプログラムになっています。

なお、GPSData の

書き込むテキストファイルのファイル名は

“GetGPS_YEAR_MONTH_DAY-OF-MONTH_HOUR-OF-DAY_MINUTE.txt”

テキストファイルへ書き込む String データは

"Latitudedata:","****",Longitudedata:," ****",Altitudedata:," ****",Error:," ****",Heading:," ****",Speed:," ****",Time(sec):," ****",Time(nsec):," ****",GPSTime(sec):," ****",GPSTime(nsec):," ****",¥n"

となっています。読み込む際に”,”で split を行っているため、****の部分をデータに書き換えてファイルを読み込めば、擬似的にセンサ値を作成することもできます。

- InPort

名称	型	説明

InTimedDoubleDataLatitude	TimedDouble	GPS の緯度情報
InTimedDoubleDataLongitude	TimedDouble	GPS の経度情報 ※Time 情報が GPSTime であることを想定しています
InTimedDoubleDataAltitude	TimedDouble	GPS の標高情報
InTimedDoubleError	TimedDouble	GPS の水平方向の精度情報
InTimedDoubleHeading	TimedDouble	GSP の傾き情報
InTimedDoubleSpeed	TimedDouble	GPS の速度情報

- OutPort

名称	型	説明
OutGPSData	GPSData	入力データの Latitude・Longitude・Altitude の三項目にフィルタ処理を行い、その他 Error 等を GPSData 型にまとめたもの。
OutTimedPositionSeq	TimedDoubleSeq	PositionSeq[0]…Latitude データ PositionSeq[1]…Longitude データ PositionSeq[2]…Altitude データ
OutTimedDoubleData Latitude	TimedDouble	入力データの Latitude 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleData Longitude	TimedDouble	入力データの Longitude 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleData Altitude	TimedDouble	入力データの Altitude 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleData Error	TimedDouble	入力データの Error 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleData Heading	TimedDouble	入力データの Heading 情報にフィルタ処理を行ったデータを出力
OutTimedDoubleData Speed	TimedDouble	入力データの Speed 情報にフィルタ処理を行ったデータを出力

- Configuration 変数
  - 2.2 の表に明記
- サービスポート
  - 無し

3 センサ情報取得コンポーネント群の使用方法

3.1 RTSystemEditor によるシステム構築の手順

RTSystemEditor を用いたシステム構築は通常、以下の手順で行われます。

I. ネームサーバの起動
II. RTSystemEditor の起動
III. ネームサーバへの接続
IV. コンポーネントの起動
V. システムの構築と実行

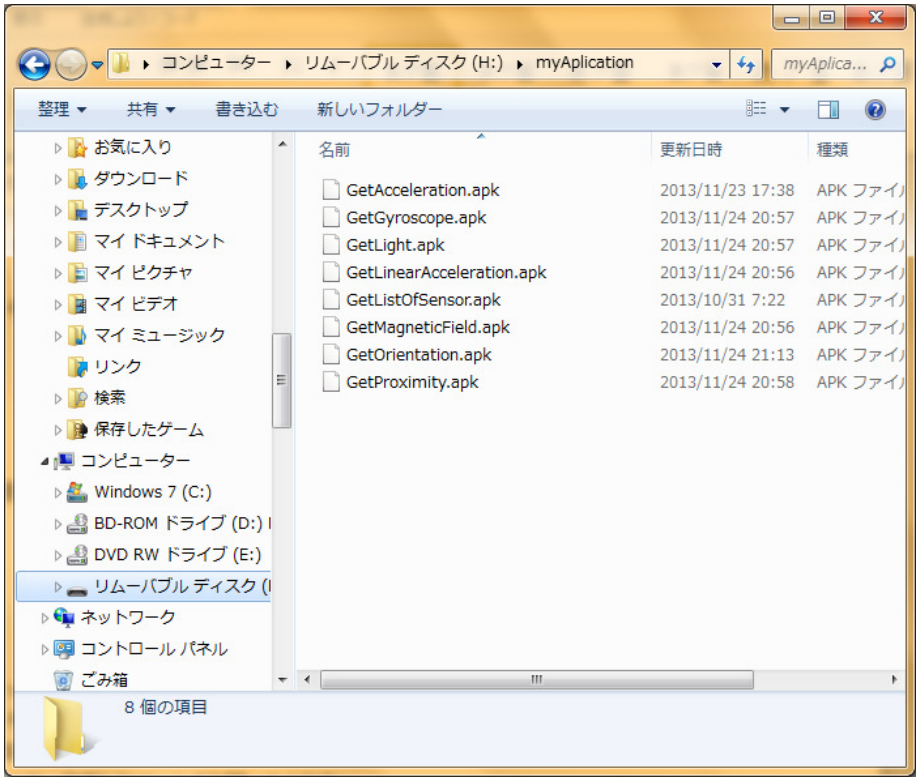
これらの手順は通常のシステムでは共通ですので、操作方法は OpenRTM-aist のホームページ(http://www.openrtm.org/)を参照してください。本コンポーネント群のフィルタコンポーネントは、通常のコンポーネントと同じ起動方法で実行準備へ移ることができます。

以下の節では、RTM on Android の起動及び、実行方法を説明します。RTM on Android はコンポーネントと同じ扱いなため、III.のネームサーバへの接続までは行っておいてください。

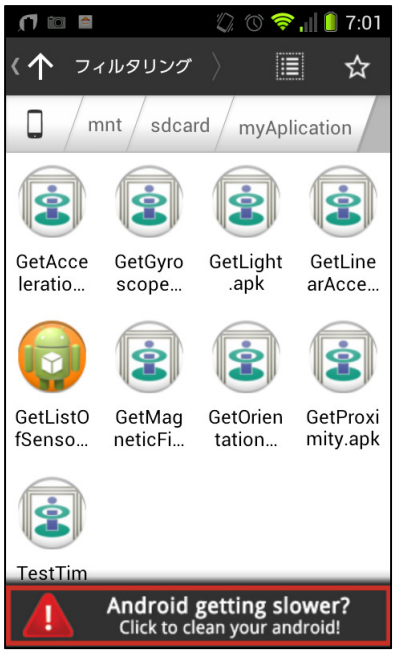
3.2 RTM on Android によるセンサ取得コンポーネントの起動と実行

※コピーの前に Android 端末の「設定」→「セキュリティ」の「提供元不明のアプリ」にチェックを入れて、インストールの許可を行ってください。また、端末がスリープ状態になると情報の取得が行えない可能性があるため、スリープまでの時間を最大にしていただくか、充電を行いながら作動できる場合には「開発者向けオプション」→「スリープモードにしない」にチェックを入れておくと効率よく作業を行えます。

手順 1 : センサ情報を取得したい Android 端末の SD カードにフォルダを作成し、そのフォルダ内にご希望のセンサ取得コンポーネントアプリケーションをコピーしてください。



手順 2：コピー後、Android 端末に SD カードを認識させ(USB モードを解除すると自動で認識します)、「アストロファイルマネージャー」などの SD カード操作用アプリケーションより、作成したフォルダを開いてください。

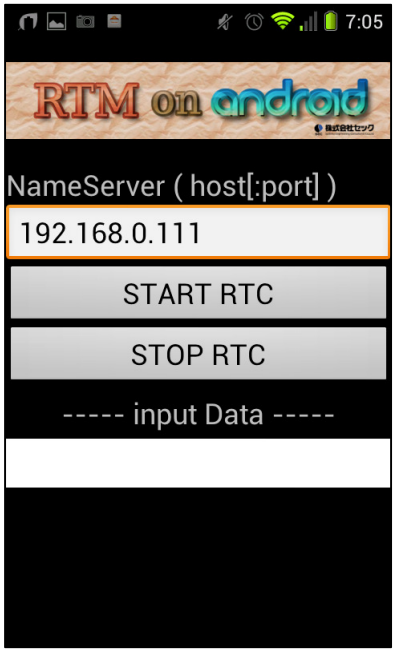


手順 3：コピーしたアプリケーションをタップすると、パッケージのインストール許可を求められますので、パッケージインストーラを選択してください。

手順 4：以下の画面に変わるので、インストールを実行してください。

手順 5：上記のインストールを終えると、ダウンロードアプリケーション一覧にアイコンが現れます。アイコンをタップして、アプリを起動してください。アプリを起動すると、下の画面が現れますので NameServer に、接続先のパソコンの IP アドレスを入力して下さ

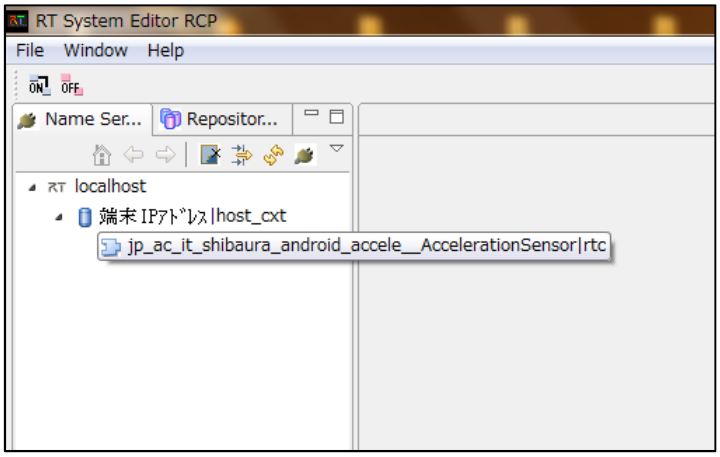
い。



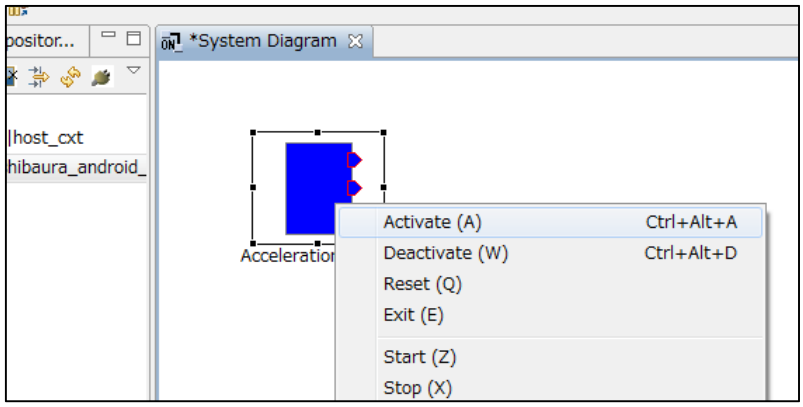
手順 6：Android 端末を、先ほど入力した IP アドレスのパソコンがあるネットワークの無線 LAN に接続し、「START RTC」をタップしてください。

これで Android 側の機動準備が整いましたので、次に、パソコン側の操作に移ります。

「START RTC」をタップして少しすると（ネットワーク強度によっては時間がかかる場合があります）host に携帯電話の IP アドレスが表示されます。このタブの中に、先ほど起動させたアプリケーションのコンポーネントが表示されていますので、従来のコンポーネントと同様に、System Diagram にドラッグアンドドロップを行ってください。（ネットワーク強度によっては時間がかかる場合があります）



ドロップ後のコンポーネントを Active にしていただくと、システムが機動します。



Android 端末の画面を確認していただき、画面下にセンサ情報が出力されていたら機動成功です。

※Android 端末のデザリング機能を用いて同じ手順で機動を行っても、実行できることが確認されています。この機能を使えば、無線環境の整っていない場所でも実行可能であるため、是非ご利用ください。また、デザリング機能を使用しての機動の場合、タイムラグが少ないため、より的確にセンサ値を取得することができます。

3.3 コンポーネント群の使用手順

動作環境：コンポーネント群は全て Java で作成されているため、Java 版の OpenRTM-aist の 1.1 をインストールしてください。1.1 のサンプルが実行可能であれば、フィルタコンポーネントを起動することができます。

環境変数：CLASSPATH　値：C¥program Files¥OpenRTM-aist¥1.1¥jar;

機動までの手順：

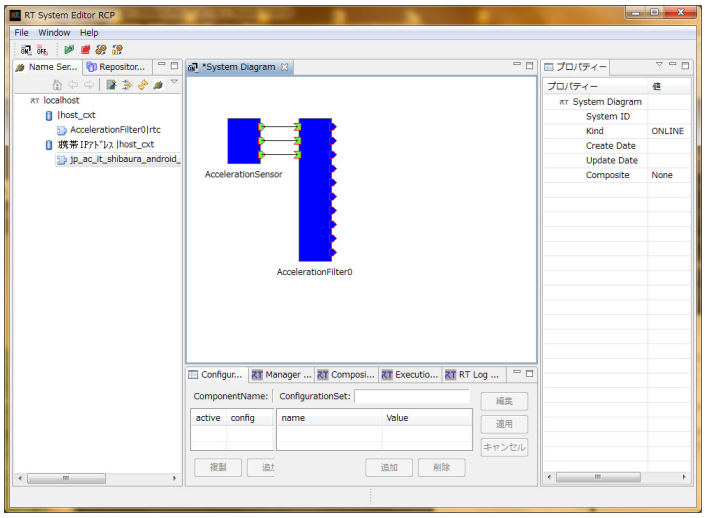
GitHub よりダウンロードして頂いた.zip を展開してください。

次に、使用したいフィルタのフォルダ内の、bin.zip をその場に展開してください。

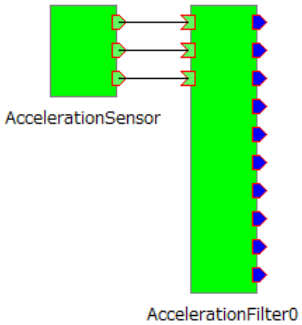
(GitHub では bin フォルダをアップロードできないため、zip 化してあります)

展開後にフィルタ側コンポーネントの bat ファイルをダブルクリックで､起動できます。

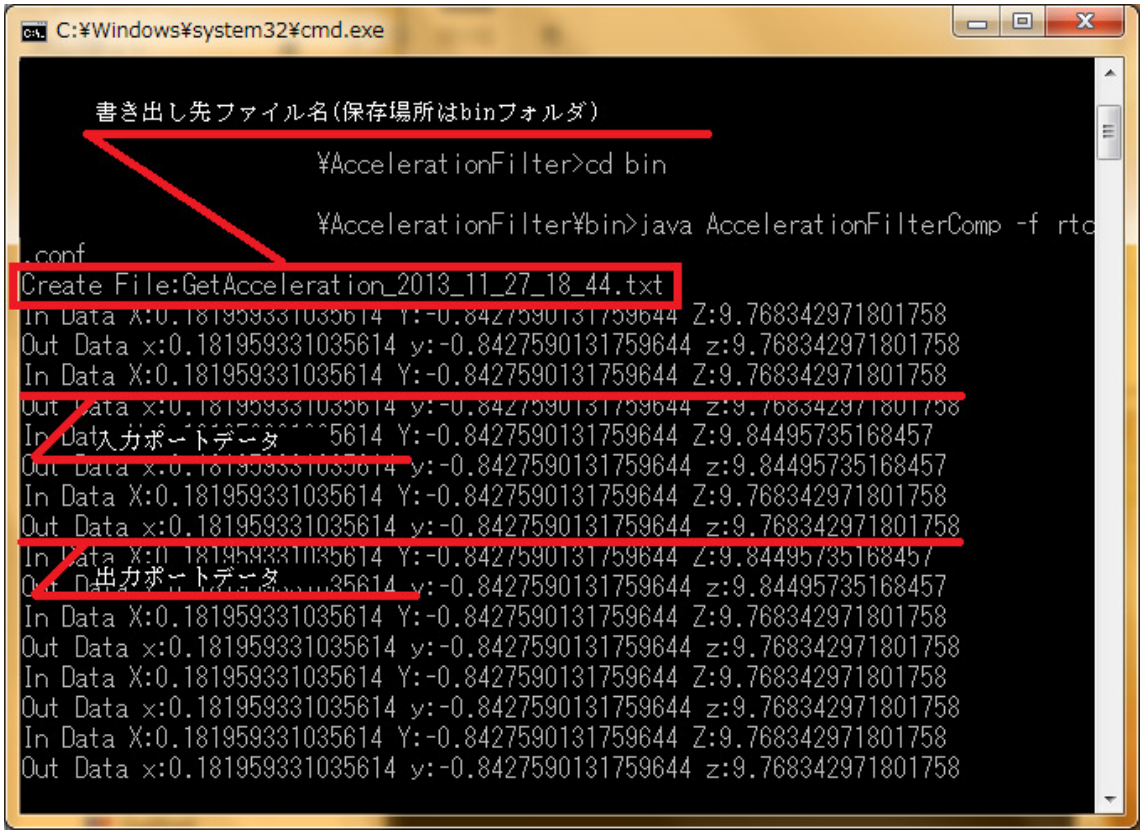
コンポーネント群としての接続図は、下の図のようになります。



何も設定を変更せずに、全コンポーネントを Active にしていただくと、端末のセンサ値をフィルタコンポーネントが取得し、OutPort への出力及び、テキストファイルへの書き出しを行います。

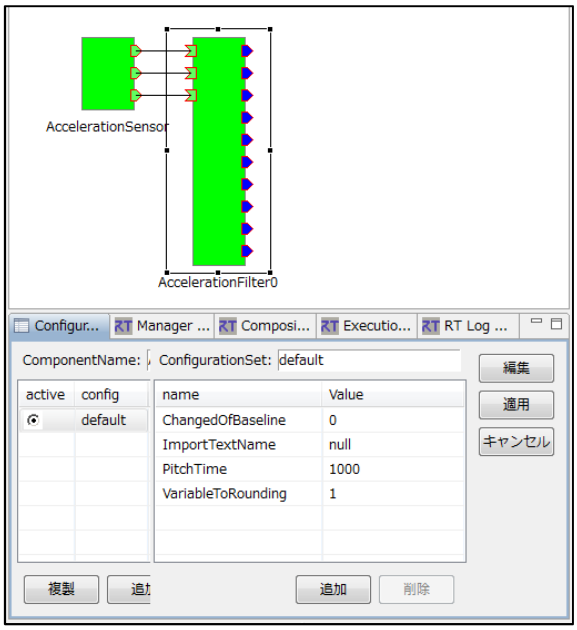


無事通信が行えていると、コンソールは下の画面のようになります。



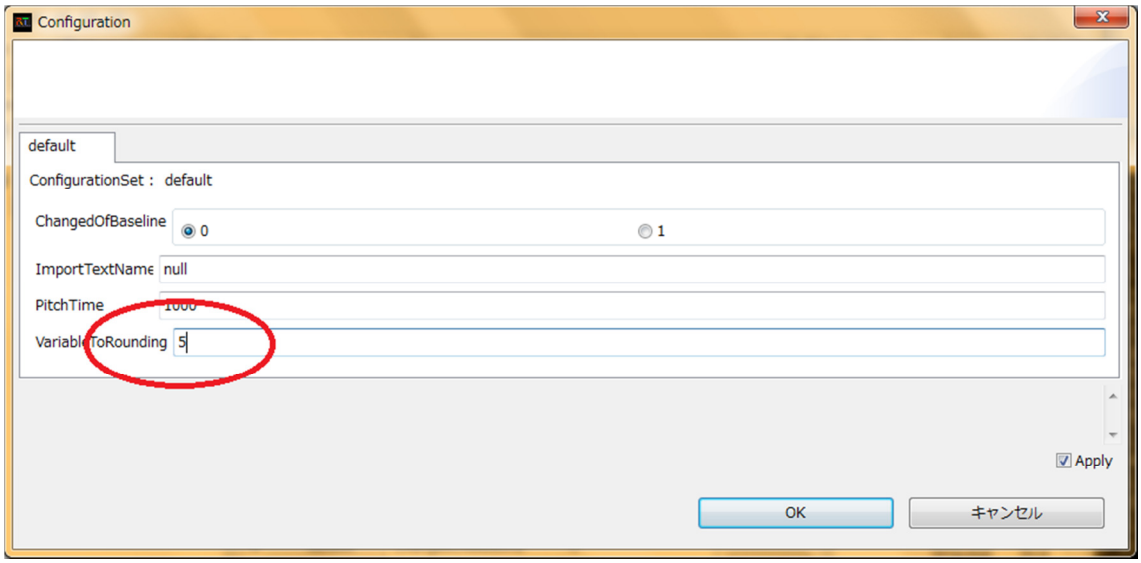
コンフィギュレーションパラメータについて：

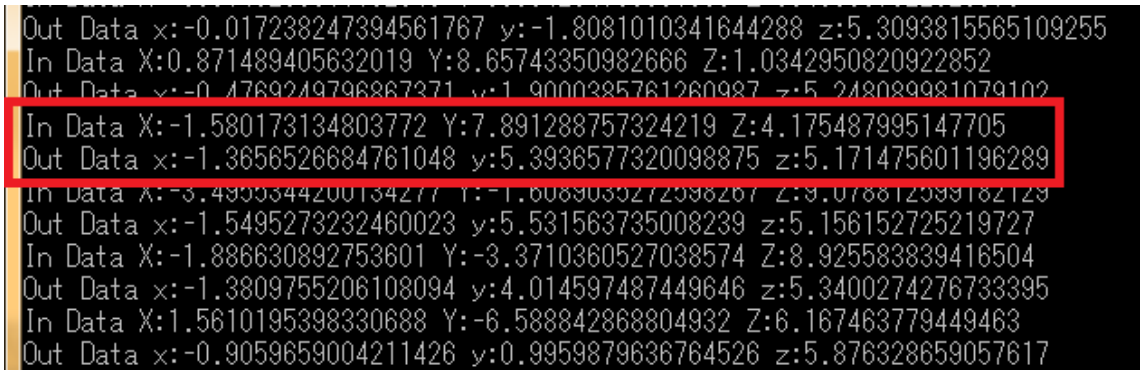
詳しい説明は 2.2 章で行っています。ここでは、簡単な操作方法及び、入力タイミングについて説明します。



VariableToRounding：暴れ値丸め処理用変数

急な暴れ値が起こるような場合がある場合、その変化値を丸めることができます。常に値が大きく変動するようなセンサの処理の場合には余り有用ではありませんので、ご使用でない場合は 1 に設定してください。

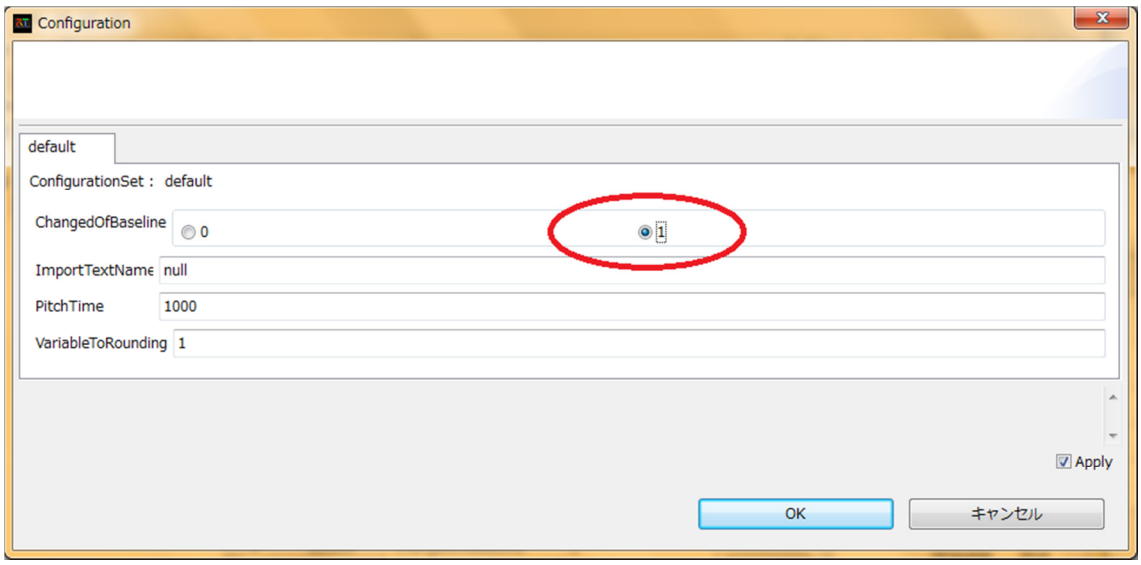


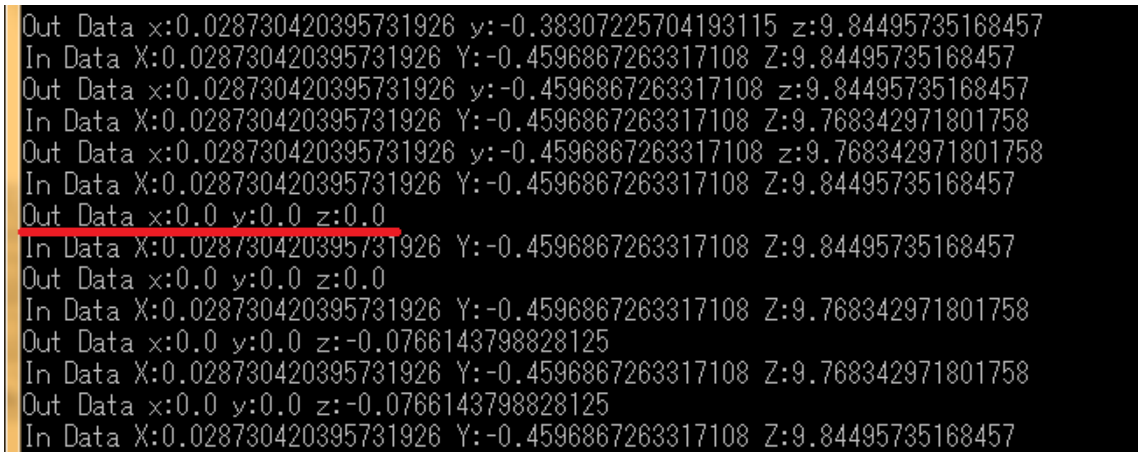


ChangedOfBaseline：基準値 0 の変更

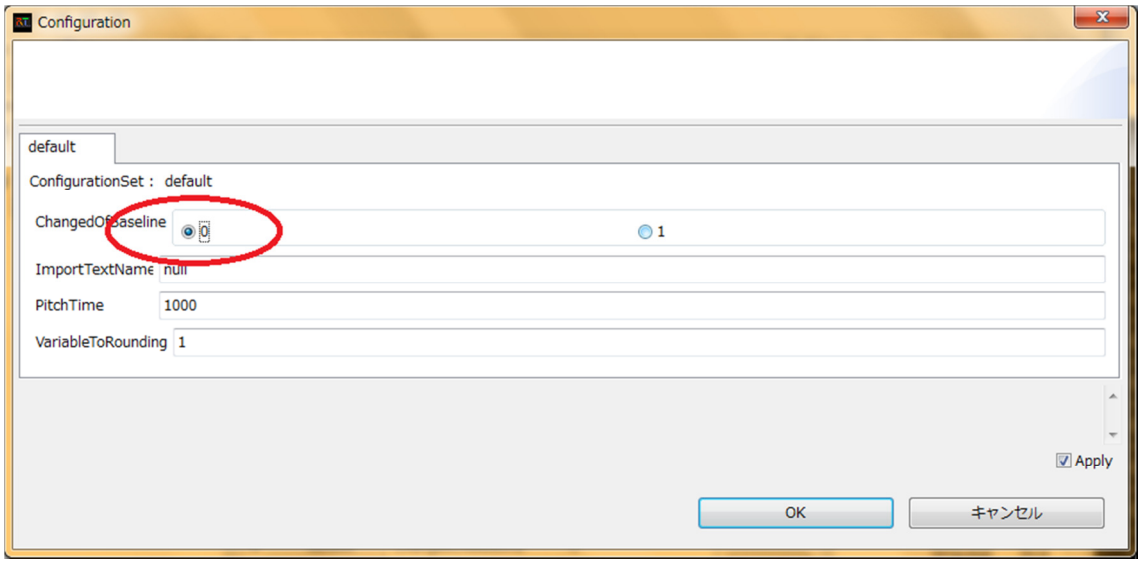
センサにとっての基準値 0 を、ラジオボタンを 1 に変更されたタイミングの値に書き換えて出力します。これは、Android 端末のキャリブレーションが正しいと思えない場合や、Android 端末を実機に組み込んでいて、初期角度に変更するのが大変である場合などに使用することが予想されます。テキストに書き込む値自体には関与しないため、テキストを読み込む処理の場合には、もう一度任意のタイミングでラジオボタンを 1 に変更してください。なお、変更された値は、もう一度 0 に戻していただくと、元に戻ります。

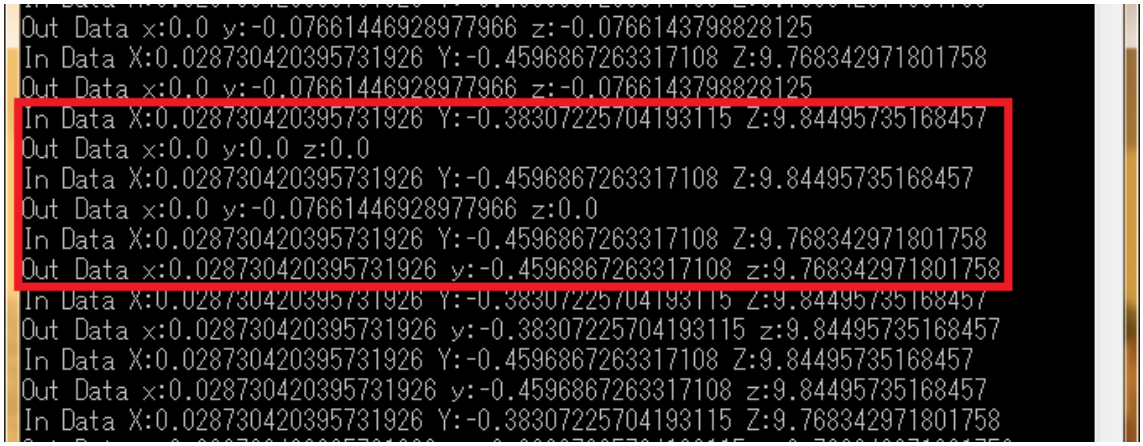
基準値 0 の変更





基準値 0 を元に戻す



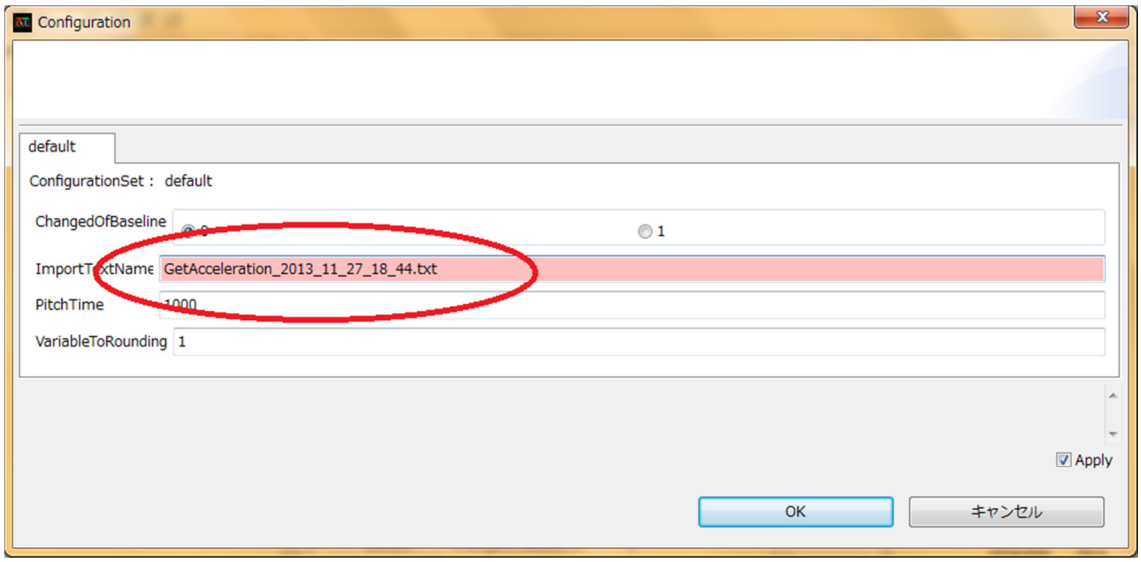


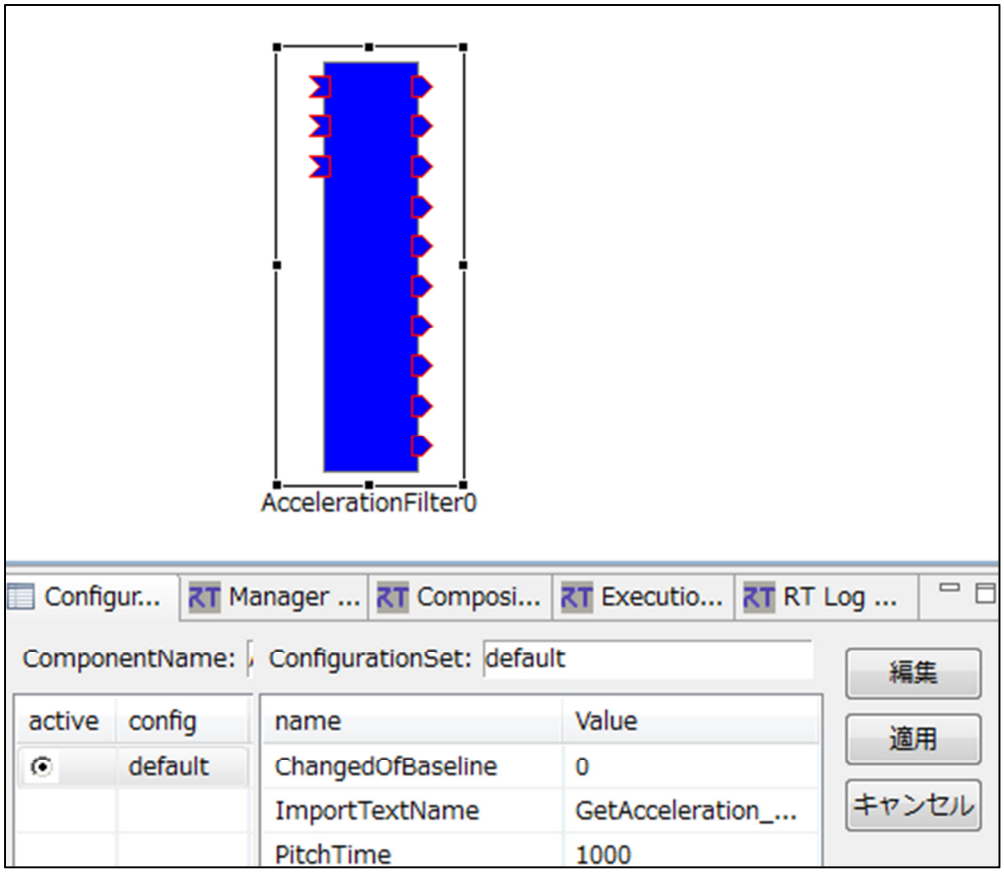
ImportTextName：フィルタにテキストファイルを読み込ませるための文字列

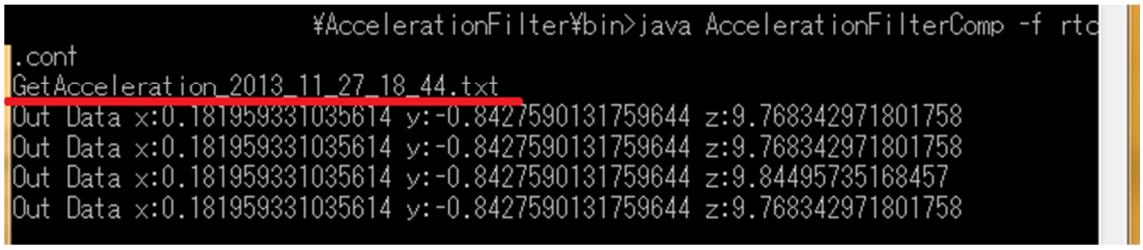
フィルタコンポーネントが自動排出するセンサ情報のテキストファイルを読み込むために使用します。ここにテキストファイル名を入力してから Active にしていただくことで、読み込みを行うことができます。

PitchTime：テキストファイルの一行を読み込む周期

既存の周期だと、テキストファイルの読み込みは一瞬で行われてしまいます。ですので、任意の読み込み時間に変更するためには、こちらの変数を使用してください。なお、単位は msec です。





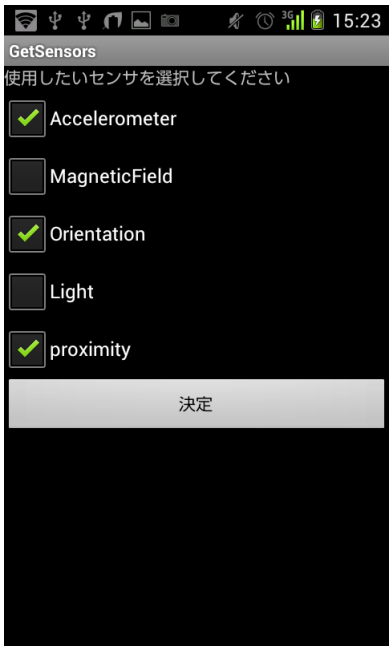


3.3.1 端末マルチセンサ取得コンポーネントアプリケーションの使用法

アプリケーションのインストールは、3.2 章の手順 4 までを参考にして、行ってください。

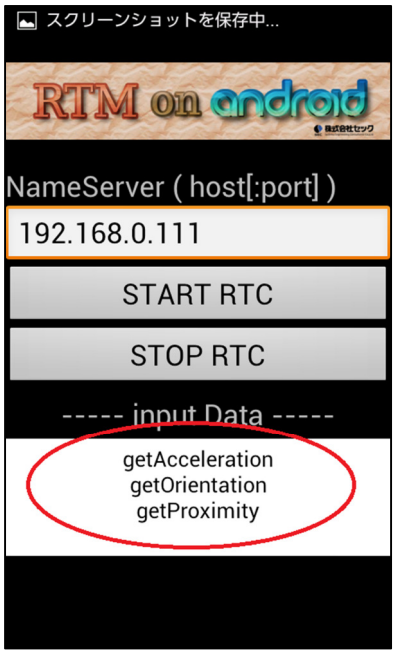
アプリケーションのインストールが終わったら、アイコンをタップして機動を行ってください。この機動処理の最中にご使用の端末に搭載されているセンサを調べ、使用可能のセンサのチェックボックス一覧を作成します。

お使いになるセンサを選択していただき、一番下に表示されている「決定」ボタンをタップしてください。(センサは複数選択することができます。)



決定を押していただくと、従来の RTMonAndroid と同様のスタート画面が表示されます。

なお、この時に画面下のテキストボックスには選択されたセンサの情報を取得できたことを示すメッセージが表示されています。

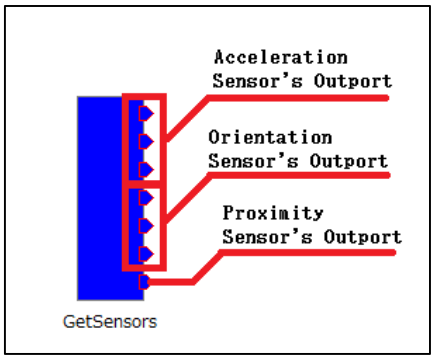


次に「START RTC」をタップして、ネームサーバーへの接続をお願いします。

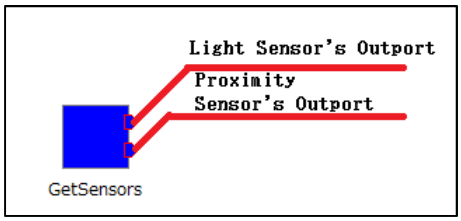
接続が完了すると、RTSystemEditor 側に、選択されたセンサに適応した OutPort を持ったコンポーネントが作成されます。

OutPort の配置の順番は、Acceleration→LinearAcceleration→Gyroscope→Gravity→MagneticField→Orientaion→RotationVector→Light→Pressure→Proximity→AmbientTemperature の順になっています。

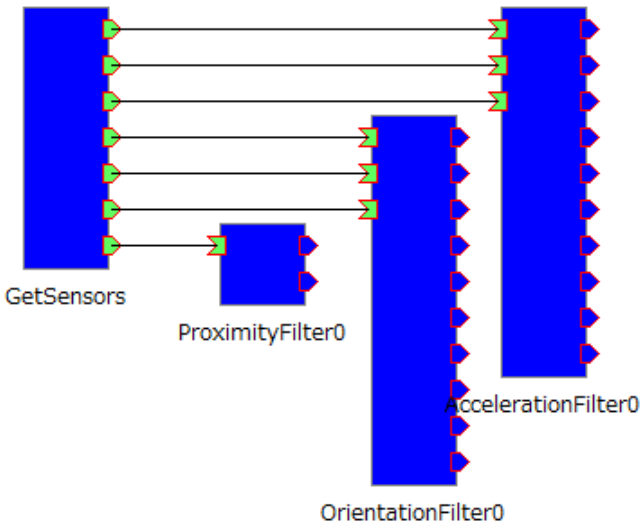
「Acceleration」「Orientation」「Proximity」を選択した場合



「Light」「Proximity」を選択した場合



あとは、このコンポーネントにフィルタ用コンポーネントとの接続を行い、Active にすれば複数センサの実行が可能になります。



4 コンポーネント群の利用例(new)

4.1 コンポーネント群利用の大まかな分類

Android 端末内のセンサ群を使用する場として、様々なシチュエーションが考えられます。

- Android 端末のセンサ情報を、外部コントローラとして使用する場合
- Android 端末を本体内部に収納し、組み込みセンサとして活用する場合
- 従来と同様の携帯端末として使用すると同時に、端末情報をアプリケーションで使用する場合

以下の節において、これらそれぞれの利用方法例をご説明いたします。

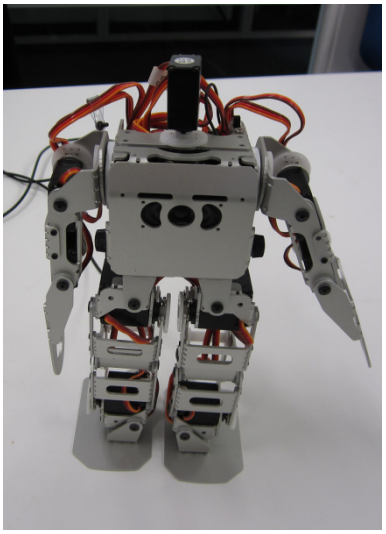
4.2 外部コントローラとして使用する場合

実行者のアクションを伴うコントローラとしての使用は、主にエンターテインメント性を高めたものに多く見られるものであり、「降る」「傾ける」などのわかりやすい動作は、対象年齢を低く設定してのロボット講習会などでも主流なセンサ情報テーマです。

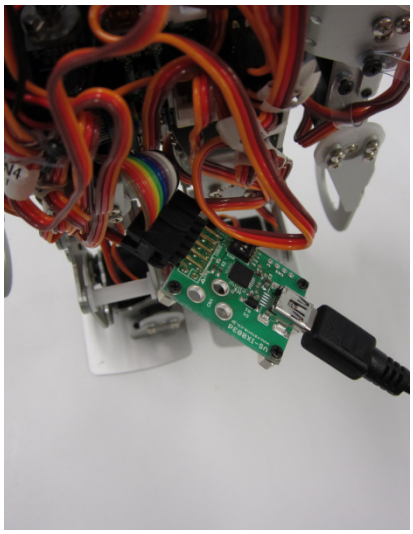
本コンポーネントを使用すれば、複雑なセンサ処理などは一切必要いため、RTM を教育の場などでも取り扱っていく上でとても有用であると考えられます。また、同じネットワークに接続してさえいれば良いため、ユーザの所持している端末に本コンポーネントアプリケーションをインストールしていただき、その場で IP アドレス接続を行えば、入れ替わり立ち代り、コントロールユーザを変更することができ、更なるエンターテインメント性を持たせることができます。

そこで、実際に Android 端末情報を用いた外部コントローラコンポーネントを作成し、実機実験を行いました。

今回コントローラの情報を実行する実機として、ヴィストン株式会社の二足歩行ロボット「Robovie-nano」及び、VS-RC003 用 シリアル通信接続ボード 「VS-IX003」を使用しています。



「Robovie-nano」



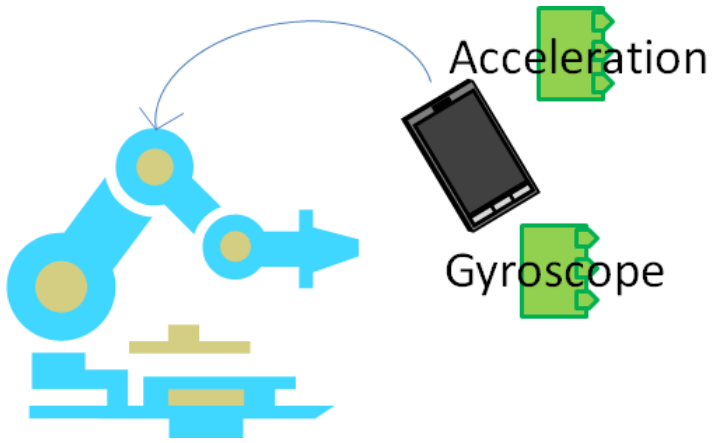
「VS-IX003」

今回の実験の使用コンポーネントは「GetOrientation」「GetLight」「OrientationFilter」「LightFilter」とロボットとシリアル通信を行い、動作を指定するコンポーネントの５つです。何も情報を与えていないときはロボットは前進し、Android 端末のピッチ角の情報が一定値を超えると、左右へのカニ歩きへと命令が変更され、照度センサで光を感知できなくなると動き止めるというプログラムを作成し、駆動を行いました。

目的通りの駆動が行えたため、外部コントローラとしては十分に役割を果たすことができることを示しました。

4.3 組み込みセンサとして活用する場合

組み込みセンサの部分を、丸々Android 端末のシステムに依存させてしまう手法です。製品として考える場合には余り良い手段ではありませんが、取り急ぎで実機を作成したい場合などには需要があると考えられます。ある一部分の加速度センサが壊れた、今とりあえず急ぎで角度情報が欲しいなど、複雑な改造を望まない場合には、本アプリケーションを実行していただき、情報が欲しい部位に取り付けていただくだけで、擬似的に組み込みセンサとして用いることができます。



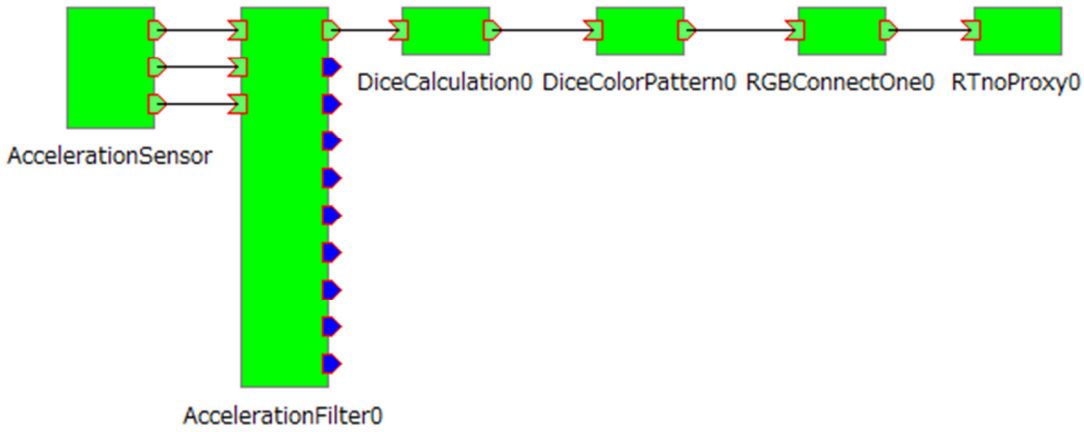
実際の実機実験としては、同コンテストに作品登録している

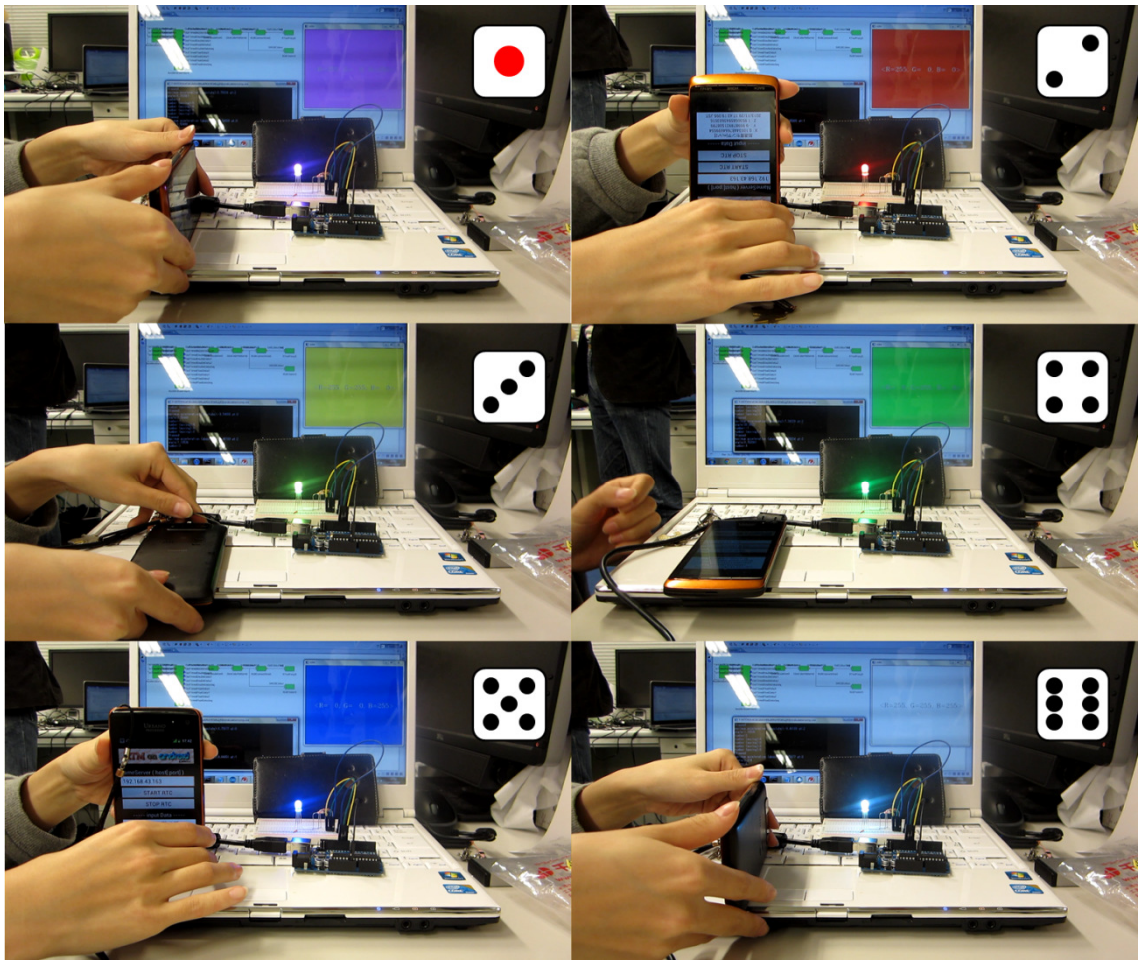
メディアアートへのＲＴミドルウェアを用いた開発手法の提案
芝浦工大 ○土屋 彩茜，立川 将，佐々木 毅

にてテスト用コンポーネントの一部として使用し、その有用性及び実用性を示しました。

実験内容としては、実機内に組み込まれている加速度センサを使用して、フルカラーLED を発行させるという一連の流れに対し、加速度センサ部分の作成が完了するまでの差し替え加速度センサとして、本コンポーネント群を使用しました。

コンポーネント図と、実験風景キャプチャーは以下のようになります。





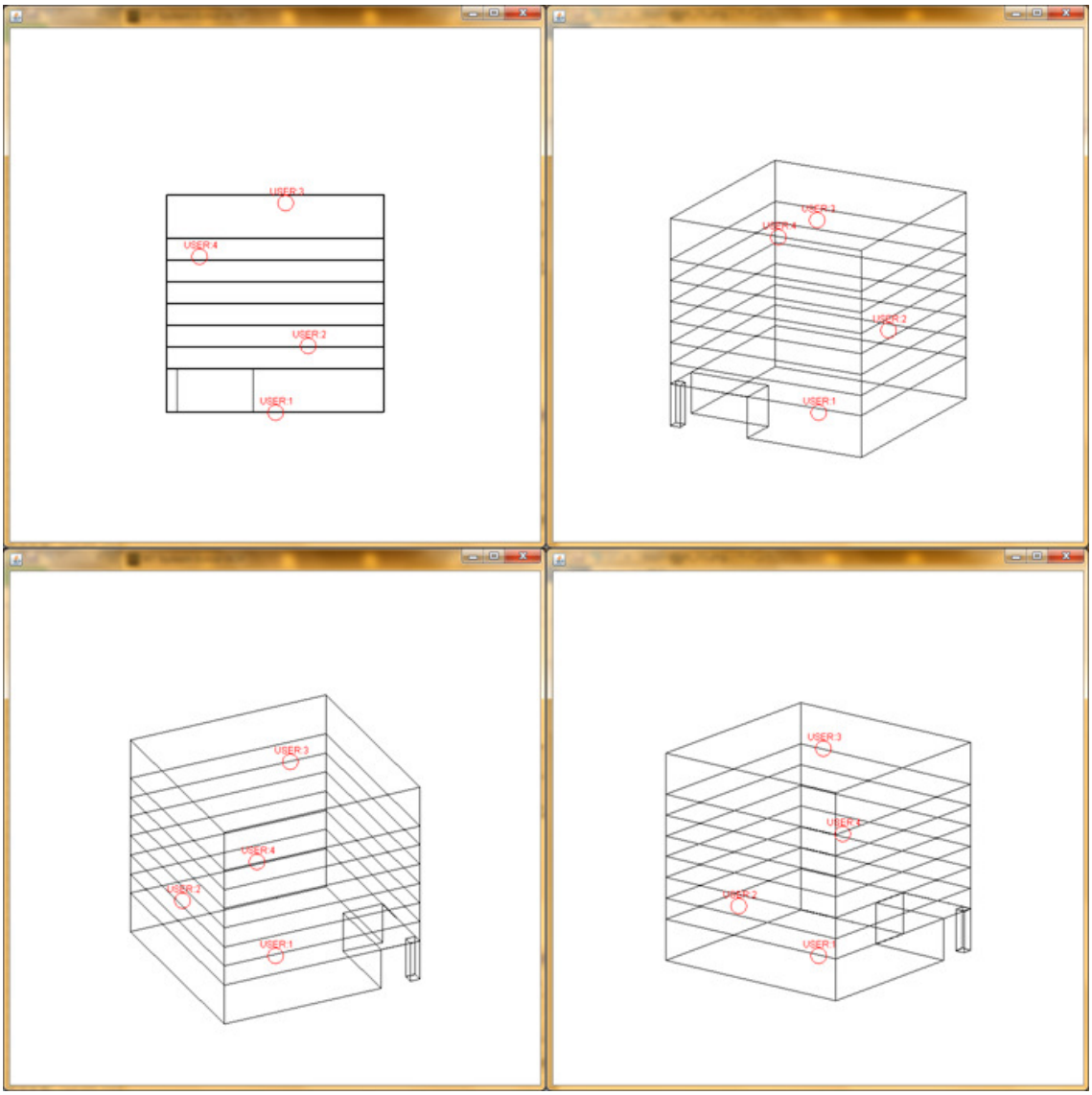
実験時の使用コンポーネントは、開発者のサポートページ
(http://www.openrtm.org/openrtm/ja/project/contest2013_1B3-4)にてダウンロードが可能です。
作品詳細についてもこちらのページをご覧下さい。

4.4 端末情報をアプリケーションで使用する場合

都市全体を同ネットワークで繋ぐなどが可能になろうとしている今、Android 端末と RT ミドルウェアを繋ぐ技術は、RT ミドルウェアを、RT の場だけでなく様々なアプリケーション上で有効活用することが出来る可能性を秘めています。その一例としては、本コンポーネントにより取得した GPS 情報を PC 上のアプリケーションで表示することで、端末所持者の居場所を、必要設備無しで管理することができるようになります。

実際に、芝浦工業大学芝浦校舎をベースに、端末の GPS 情報を表示してユーザの居場所管理を行いました。使用設備は、芝浦工業大学の校舎内に設備されている無線 LAN と、携帯端末 4 台、データ受信及び位置情報出力用 PC 1 台です。データの出力は、校舎及び位置情報を 3 D モデルに出力するコンポーネントを作成しました。

GPS 感度はオフィス街ということもありあまり良くはありませんでしたが、大まかに居場所を把握することができました。



5 お問い合わせ

本コンポーネントにつきましては、まだ改善の余地があるものと考えております。ご要望、バグ報告、マニュアルの記述の不備等に関しましては、芝浦工業大学大学院理工学研究科電気電子情報工学専攻の立川までご連絡ください。

【問合せ先】

〒108-8548

東京都港区芝浦 3-9-14

芝浦工業大学 611 室

Mail: MA13055@shibaura-it.ac.jp