RT コンポーネント操作マニュアル

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

2014 年 12 月 1 日版

芝浦工業大学大学院

ロボティクスシステムデザイン研究室

土屋彩茜　立川将　佐々木毅

更新履歴

2013 年 11 月 28 日版　第 1 版.

2013 年 11 月 29 日版　誤字等の修正、および第 2 章、第 3 章 4 節の修正。

2013 年 12 月 6 日版　誤字等の修正、および第 4 章コミュニティの編集。

2014 年 12 月 1 日版　コンポーネント修正・追加による第 2 章の変更。第 6 章にコンポーネントの多重起動に関する項目を追加。所属変更による問い合わせ先の変更およびコミュニティの QR コードの追加。全体的な誤字等の修正および編集。

目次

1. 本提案の概要

1.1 開発の背景

近年メディアアートを目にする機会が増えています。興味を持つ人が増えている中、RTやプログラミングについて知識が必要なことから、制作の敷居は高いものとなっています。そこでメディアアーティストが簡単にRTを利用し、表現を制作することが出来る環境の実現が望まれます。このような背景から、環境実現のための事例として実際に制作を行い、メディアアート表現のためのコンポーネント開発を行うことといたしました。

コンポーネントの設計指針等につきましては、

> 土屋 彩茜, 立川 将, 佐々木 毅（芝浦工大）
> “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”,
> 第13回計測自動制御学会システムインテグレーション部門講演会, 2013.

に詳細がありますので、そちらもご参照いただけましたら幸いです。

1.2 開発事例-MovicCube-

「身近なものとRT(Robot Technology)の組み合わせで新しいものを作る」というコンセプトに基づき、様々な動作と表現を組み合わせた「動作するサイコロ-Movic Cube-」の実現を目指します。本稿では「光る」という動作と表現を組み合わせてMovicCubeの制作を行いました。

この制作を通し、メディアアート的表現を制作する上で発生する技術的知識の不足を解消する手段として、RTコンポーネントを使用することを提案しております。また実機制作とシステム（表現）制作を別々に行う場合も想定し、Android端末を用いた表現コンポーネントのみでテストを行う方法についても紹介いたします。

1.3 開発環境

本コンポーネント群はWindowsにて動作確認を行いました。開発環境は以下の通りです。

- Windows 7（32bit版,64bit版）
- RTミドルウェア：OpenRTM-aist-1.1.0-RELEASE（C++版）
- コンパイラ：Microsoft　Visual　C++ 2010　Express（SP1）
- Eclipse：Eclipse 3.4.2 + OpenRTM Eclipse tools 1.1.0-RC3
  Eclipse 3.8.1 + OpenRTM Eclipse tools 1.1.0-RC4
- CMake：CMake 2.8.8

1.4 開発したコンポーネント群

MovicCube のための基本的な表現を提供する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。

・サイコロ計算コンポーネント（DiceCalculation）
・サイコロ色表現コンポーネント（DiceColorPattern）2014/12/1:修正
・3 軸加速度調整コンポーネント（Calibration_forAcceleration3D）2014/12/1:修正
（旧：3 軸加速度接続コンポーネント（Acceleration3DConnectOne））
・RGB 変換コンポーネント（convRGBColor）2014/12/1:修正
（旧：RGB 接続コンポーネント（RGBConnectOne））
・RGB 表示コンポーネント（RGBTester）

それぞれのコンポーネントの詳細は 2 章を参照してください。
また、メディアアート的表現として、以下のコンポーネント群を追加開発しました。

・サイコロ記憶コンポーネント（DiceMemory）2014/12/1:追加
・サイコロソートコンポーネント（DiceSort）2014/12/1:追加

1.5 利用したコンポーネント群

Arduino との接続には、RTno ライブラリと RTnoProxy コンポーネントを利用しています。詳細については（http://ysuga.net/rtc/124）をご覧ください。

また、Movic Cube のプロトタイプの動作には、2013 年度の RT ミドルウェアコンテストで発表された以下のコンポーネントを利用しています。

立川 将, 大野 祥平, 佐々木 毅（芝浦工大）
“RTM on Android を用いた Android 用マルチセンサコンポーネント群“
第 13 回計測自動制御学会システムインテグレーション部門講演会, 2013.

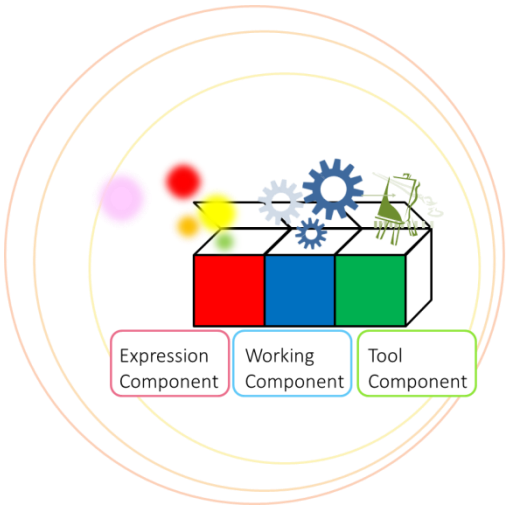
・端末加速度取得コンポーネントアプリケーション(GetAcceleration)
・三軸加速度フィルタコンポーネント(AccelerationFilter)

これらのコンポーネントの詳細については、開発者のサポートページ（http://www.openrtm.org/openrtm/ja/project/contest2013_1B3-3）を参照してください。

2. コンポーネントの説明

2.1 コンポーネントの構成

本提案ではコンポーネントを以下の構成に分けて開発いたしました。



動作コンポーネント群は、メディアアートの動作を決めるコンポーネントです。実機に影響する部分を Configuration とすることで、実機とシステムの作業を分けて行うことができます。また、他の RT コンポーネントを利用することで、動作を増やすことができます。

次に表現コンポーネント群は、メディアアート的な表現に関わるコンポーネントです。表現のパラメータを Configuration とすることで、プログラムへの知識が少ないアーティストでも表現を変更することができます。また、コンポーネントをつなげることで表現を複雑化することや、独自の表現をコンポーネント化し共有することができます。

最後にツールコンポーネントは、開発支援ツールとしてのコンポーネントや、構成拡張のためのコンポーネントとなっています。制作目的に合わせて使用することで、制作の利便化やアート自体の拡張ができます。

以上 3 つの構成のコンポーネントの組み合わせによってメディアアートの制作を行います。

2.2 動作コンポーネント

メディアアートの動作を決めるコンポーネントです。実機に影響する部分を Configuration として設計しました。今回開発したコンポーネントの中では以下のものを動作コンポーネントとしています。

・サイコロ計算コンポーネント（DiceCalculation）
・3 軸加速度調整コンポーネント（Calibration_forAcceleration3D）2014/12/1:修正
・RGB 変換コンポーネント（convRGBColor）2014/12/1:修正
・サイコロソートコンポーネント（DiceSort）2014/12/1:追加

本章では、これらのコンポーネントについて説明します。

2.2.1 サイコロ計算コンポーネント（DiceCalculation）

DiceCalculation は、正六面体のサイコロの出目を算出するコンポーネントです。InPort の DiceAcceleration からサイコロ内の 3 軸加速度センサ値を読み込み、サイコロが静止した時、サイコロの上面にあたる目の数を OutPort の DiceNumber から出力します。

Configuration では静止判定と上面判定のための閾値を変更することができます。静止状態の判定は、現在の加速度と重力加速度を比較することで行っており、

①各軸加速度値の前回の加速度値との差が SensorThreshold の値より低い

②3 軸の加速度の合計値と重力加速度の差が SensorThreshold の値より低い

状態をいいます。この状態の時間が、StopTimeThreshold で設定した時間続いた場合、サイコロが止まって目を判定できるとみなします。

上面の判定は SlopeThreshold で設定した角度をサイコロの傾き誤差とし、この範囲未満であれば上面が決定されます。

アクティブ後、InPort の DiceAcceleration からセンサの値が入力される度に静止状態の判定に入ります。静止状態と判定された場合は上面の判定を行い、その結果上面が決定されたときは、その面の値を出目として OutPort の DiceNumber から出力します。判定ができない場合は-1 を OutPort から出力します。その後、静止状態が解除されるまでは出力を行いません。解除されている間は 0 を出力します。

また、Configuration の PlusX～MinusZ で各面の目の値を設定できます。x 軸の加速度が+1g になる面の目の値を PlusX に、-1g になる面の目の値を MinusX に、といったように各パラメータを設定してください。デフォルト値は+X 軸方向に 1、+Y 軸方向に 2、+Z 軸方向に 3 を設置し、それぞれ対面との和が 7 になるようにして設定してあります。

・InPort

名称	型	説明
DiceAcceleration	TimedAcceleration3D	3 軸の加速度値。単位はm/s^2。

・OutPort

名称	型	説明
DiceNumber	TimedShort	サイコロの出目。サイコロの静止状態が確認され、上面が決定する場合、上面にある目の値を出力する。上面が決定出来ない場合は-1 を出力する。静止していない時は 0 を出力する。

・Configuration

名称	型	デフォルト値	説明
PlusX	int	1	+X 軸方向にある面に対応する目の値。
PlusY	int	2	+Y 軸方向にある面に対応する目の値。
PlusZ	int	3	+Z 軸方向にある面に対応する目の値。
MinusX	int	6	-X 軸方向にある面に対応する目の値。
MinusY	int	5	-Y 軸方向にある面に対応する目の値。
MinusZ	int	4	-Z 軸方向にある面に対応する目の値。
SensorThreshold	double	0.15	静止していると判断するための閾値。重力加速度と入力加速度を比較し、差がこの値より小さい場合、静止状態とみなす。単位はm/s^2。
StopTimeThreshold	double	1.0	サイコロの目の判定を開始するために必要な静止状態の時間。 単位は秒。
SlopeThreshold	int	15	上面を特定するための閾値。この傾きの角度分を誤差とする。単位は度。 範囲は$0 \leq \mathrm{SlopeThreshold} < 45$。

2.2.2 3 軸加速度調整コンポーネント（Calibration_forAcceleration3D）

Calibration_forAcceleration3D は、InPort から入力されたセンサ値に対し、Configuration で設定したキャリブレーションパラメータでキャリブレーションを行い、OutPort から結果を出力するコンポーネントです。InPort と OutPort は Acceleration3D 型と TimedDoubleSeq 型があり、使用するポートの型を Configuration で指定します。

キャリブレーションの方法としては、入力値に対し推定値を Ax+B の線形の形で表現し、A,B がそれぞれ params_x,y,z の値となっています。さらに、params_theta,psi,phi では傾きを指定することができ、上記パラメータによる推定値に対して、回転した値を結果として出力します。また、デフォルトでは、入力値がそのまま出力されるため、型の変更のみとしても利用できます。

・InPort

名称	型	説明
Acceleration3DIn	Acceleration3D	Acceleration3D 型のセンサ値を取得するポート。単位はm/s^2。
TimedDoubleSeqIn	TimedDoubleSeq	TimedDoubleSeq 型のセンサ値を取得するポート。配列の 0 番目から順に x 軸、y 軸、z 軸。単位はm/s^2。

・OutPort

名称	型	説明
Acceleration3DOut	Acceleration3D	Acceleration3D 型のセンサ値を出力するポート。単位はm/s^2。
TimedDoubleSeqOut	TimedDoubleSeq	TimedDoubleSeq 型のセンサ値を出力するポート。配列の 0 番目から順に x 軸、y 軸、z 軸。単位はm/s^2。

・Configuration

名称	型	デフォルト値	説明
params_x	std::vector <double>	1.0,0	x 軸に対するパラメータ。 x 軸のセンサ値を Ax+B の線形で表し、(A,B)の順で設定する。
params_y	std::vector <double>	1.0,0	y 軸に対するパラメータ。 y 軸のセンサ値を Ay+B の線形で表し、(A,B)の順で設定する。
params_z	std::vector <double>	1.0,0	z 軸に対するパラメータ。 z 軸のセンサ値を Az+B の線形で表し、(A,B)の順で設定する。
params_theta	double	0	x 軸の角度に対するパラメータ。
params_psi	double	0	y 軸の角度に対するパラメータ。
params_phi	double	0	z 軸の角度に対するパラメータ。
InPortSelect	string	TimeDoubleSeqIn	入力ポートの型を選択する。 radio ボタンで選択できる。
OutPortSelect	string	Accelaration3DOut	出力ポートの型を選択する。 radio ボタンで選択できる。

2.2.3 RGB 変換コンポーネント（convRGBColor）

RGBConnectOne は、InPort の値を読み込んで、0~255 の範囲のデータに変換し、OutPort から出力するコンポーネントです。InPort、OutPort は TimedDoubleSeq 型、TimedShortSeq 型、TimedRGBColour 型があり、使用するポートの型を Configuration で指定します。

Configuration の MaxData で MinData は入力される値の範囲を入力します。この範囲外の値はその時の最大・最小値と同値とします。MaxData を 0、MinData を 255 というように値を反対にすることで、RGB 値を反転させて出力することもできます。また、デフォルトでは、入力値がそのまま出力されるため、型の変更のみとしても利用できます。

・InPort

名称	型	説明
TimedDoubleSeqIn	TimedDoubleSeq	TimedDoubleSeq 型のデータを取得するポート。配列の 0 番目から順に R、G、B。
TimedShortSeqIn	TimedShortSeq	TimedShortSeq 型のデータを取得するポート。配列の 0 番目から順に R、G、B。
TimedRGBColourIn	TimedRGBColour	TimedRGBColour 型のデータを取得するポート。

・OutPort

名称	型	説明
TimedDoubleSeqOut	TimedDoubleSeq	TimedDoubleSeq 型のデータを出力するポート。配列の 0 番目から順に R、G、B。
TimedShortSeqOut	TimedShortSeq	TimedShortSeq 型のデータを出力するポート。配列の 0 番目から順に R、G、B。
TimedRGBColourOut	TimedRGBColour	TimedRGBColour 型のデータを出力するポート。

・Configuration

名称	型	デフォルト値	説明
MaxData	double	255	入力データの最大値を指定する。
MinData	double	0	入力データの最小値を指定する。
InPortSelect	string	TimeShortSeqIn	入力ポートの型を選択する。 radio ボタンで選択できる。
OutPortSelect	string	TimedRGBColourOut	出力ポートの型を選択する。 radio ボタンで選択できる。

2.2.4 サイコロソートコンポーネント（DiceSort）

DiceSort は、サイコロの出目を複数取得し、ソート結果を配列として OutPort の SortDiceNumbers から出力するコンポーネントです。InPort 数は、サイコロの数に応じて Configuration で変更することができます。また、ソート順を選択することもできます。

・InPort

名称	型	説明
InDiceNumber	DynamicInPort<TimedShort>	サイコロの出目を取得するポート。1～6 がそれぞれサイコロの目に対応している。それ以外の値が来たときは待機状態となる。

・OutPort

名称	型	説明
SortDiceNumbers	TimedShortSeq	ソートされた順に出目の値を出力するポート。配列の 0 番目を含めた偶数番目にはサイコロのポート情報が入り、奇数番目には出目が入る。

・Configuration

名称	型	デフォルト値	説明
AddPort	int	1	InPort の数を指定する。サイコロの実機の数に合わせて指定する。範囲は$x \geq 1$。
SortBase	int	0	ソート順を選択する。radio ボタンで選択できる。 0：大きい・早い順にソート 1：小さい・遅い順にソート
SortObject	int	0	ソート条件を選択する。radio ボタンで選択できる。 0：出目 1：入力のあった時間

2.3　表現コンポーネント

メディアアート的な表現に関わるコンポーネントです。表現のパラメータを Configuration として設計しました。今回開発したコンポーネントの中では以下のものを表現コンポーネントとしています。

・サイコロ色表現コンポーネント（DiceColorPattern）2014/12/1:修正
・サイコロ記憶コンポーネント（DiceMemory）2014/12/1:追加

本章では、これらのコンポーネントについて説明します。

2.3.1 サイコロ色表現コンポーネント（DiceColorPattern）

DiceColorPattern は、サイコロの出目に応じて LED の色を変化させるための表現コンポーネントです。InPort の DiceNumber からサイコロの出目を読み込み、その目に応じた LED の色を OutPort の DiceColorRGB から出力します。Configuration では、各目に対して好きなように色を設定することで、表現を決めることが出来ます。Dice1 はサイコロの目の 1、Dice2 はサイコロの目の 2 といったように名前と目の数字が対応しています。サイコロが動いている間や出目が取得できない場合は LED を消灯させます。

・InPort

名称	型	説明
DiceNumber	TimedShort	サイコロの出目を取得するポート。 1～6 がそれぞれサイコロの目に対応しており、 それ以外の値が来た場合、LED を消灯させる。

・OutPort

名称	型	説明
DiceColorRGB	TimedRGBColour	LED の色を出力するポート。

・Configuration

名称	型	デフォルト値	説明
Dice1	vector<int>	255,0,255	サイコロの目が 1 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は magenta。
Dice2	vector<int>	255,0,0	サイコロの目が 2 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は red。
Dice3	vector<int>	255,255,0	サイコロの目が 3 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は yellow。

Dice4	vector<int>	0,255,0	サイコロの目が 4 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は green。
Dice5	vector<int>	0,0,255	サイコロの目が 5 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は blue。
Dice6	vector<int>	255,255,255	サイコロの目が 6 の時の LED の色(RGB)。 各項 0～255 の値をとる。デフォルト値は white。

2.3.2 サイコロ記憶コンポーネント（DiceMemory）

DiceMemory は、出目を記憶して順に出力するコンポーネントです。出目は InPort の DiceNumber から入力され、OutPort の DiceMemory から出力します。Configuration で指定された数の目を記憶すると、動いて次に静止したときに記憶の再生を始めます。再生が完了したら消灯し、次に動き始めるまで待機します。Configuration では、再生時の時間間隔とループの ON/OFF を切り替えることが出来ます。TimeInterval では出力間隔を指定できるので、それにより記憶再生の時間間隔が変更できます。LoopMode ではループの ON/OFF が切り替え出来ます。ON の場合は、出力し終わった後に消灯せず、また初めから出力します。ループ中に Configuration が切り替わった場合は、OFF の時と同様にその回の出力が終わった段階で消灯し、待機します。

・InPort

名称	型	説明
DiceNumber	TimedShort	記憶対象サイコロの出目。

・OutPort

名称	型	説明
DiceMemory	TimedShort	記憶した結果のサイコロの出目。記憶順に出力される。 出力間隔は Configuration の TimeInterval で指定される。

・Configuration

名称	型	デフォルト値	説明
TimeInterval	double	2.0	出目を出力する際の時間間隔。単位は秒。
LoopMode	int	1	出力がループするかどうかの変更。radio ボタンで選択できる。 0：ON。ループする。 1：OFF。ループしない。
DiceStore	int	5	出目の記録数。範囲は x>0。

2.4 ツールコンポーネント

開発支援ツールとしてのコンポーネントや、構成拡張のためのコンポーネントとなっています。今回開発したコンポーネントの中では以下のものをツールコンポーネントとしています。

・RGB 表示コンポーネント（RGBTester）

本章では、これらのコンポーネントについて説明します。

2.4.1 RGB 確認コンポーネント（RGBTester）

RGBTester は、InPort の InRGBColour から入力された RGB の色を、RGB 値とともにウィンドウに表示するコンポーネントです。Configuration では表示ウィンドウのサイズを変更することが出来ます。

・InPort

名称	型	説明
InRGBColour	TimedRGBColour	ウィンドウに表示する色（RGB）。

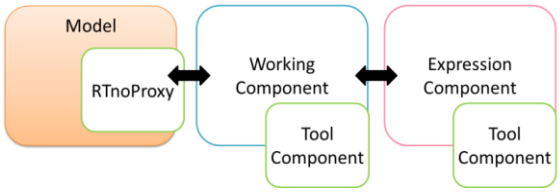
・OutPort

無し

・Configuration

名称	型	デフォルト値	説明
Width	int	500	色を表示するウィンドウの幅。
Height	int	100	色を表示するウィンドウの高さ。

3. メディアアートのシステム構成



メディアアートのシステム構成として、動作コンポーネントと表現コンポーネントを接続するという形が基本構成となります。

実機と繋がるところが動作コンポーネントとなり、実機に応じて Configuration を設定することで、動作を実現します。Movic Cube を例とすると、加速度センサから出目をみる。LED の RGB 値を指定する。といった動作が実現されます。

さらに、動作コンポーネントに表現コンポーネントを接続することで、動作に表現を加えることができます。Movic Cube を例とすると、出目に応じて RGB が変わる。などといった表現が加わります。

これらによって構築されたシステムを、ツールコンポーネントや他のコンポーネントによって拡張していきます。今回は RTnoProxy によって Arduino を用いた実機とコンポーネントが繋ぐことができます。

また、それ以外にも動作コンポーネントや表現コンポーネントの拡張に使えるものをツールコンポーネントと呼んでいます。Movic Cube を例とすると、表現のテストツールとして RGB を表示させる。といったものです。

このように、動作コンポーネント＋表現コンポーネントという基本構成に対し、ツールコンポーネントで拡張していくことで、変更や別のシステム構成を考えることができると考えます。

4. コンポーネントの使用方法

本提案では基本構成の動作コンポーネントと表現コンポーネントを接続することで、メディアアートのシステムの実現を目指します。本章では以下のコンポーネント群を用いて、Movic Cube を例として使用方法の説明を行います。

・サイコロ計算コンポーネント（DiceCalculation）
・サイコロ色表現コンポーネント（DiceColorPattern）
・3 軸加速度調整コンポーネント（Calibration_forAcceleration3D）
・RGB 変換コンポーネント（convRGBColor）
・RTnoProxy コンポーネント

4.1 RTSystemeEditor を用いたシステム構築の手順

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

1. ネームサーバを起動する
2. RTSystemEditor を起動する
3. ネームサーバへ接続する
4. コンポーネントを起動する
5. システムを構築し、実行する

これらの手順はどのようなシステムでも共通です。RTSystemEditor の詳しい操作方法は OpenRTM-aist のホームページ（http://www.openrtm.org/）を参照してください。以下の節では手順 5 に関して説明します。

4.2 Arduino との接続方法（RTnoProxy の使用手順）

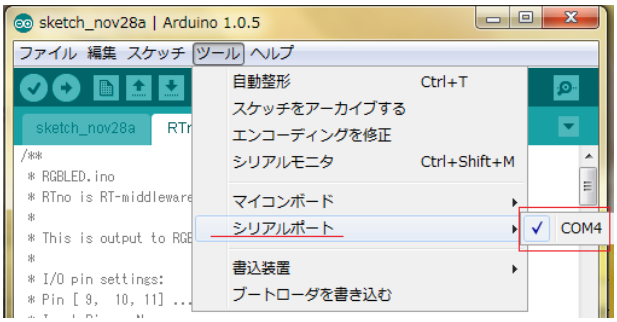
まず初めに実機との通信部分についての説明を行います。Movic Cube の実機部分では Arduino（アルディーノ）というオープンソースの電子プロトタイピングプラットフォームを使用しています。この Arduino と RT コンポーネントとの通信を簡略化するためのライブラリとして RTno があり、これを利用して他の RT コンポーネントとの通信を可能にするコンポーネントが RTnoProxy です。

この章では、Movic Cube を使用するために必要な手順のみを説明します。したがって、この手順の前に Arduino-1.0 のインストールと RTno および RTnoProxy のダウンロードと展開を行ってください。この方法については、以下を参照してください。

・Arduino 公式サイト：http://www.arduino.cc/
・RTno・RTnoProxy 開発者のページ：http://ysuga.sakura.ne.jp/robot/rtm/rtc/rtno

準備ができたら Arduino を USB ケーブルで接続し、Arduino 1.0.5 を起動します。その後ソースコードを書きます。RTnoProxy のポートは Arduino のソースコードで記述されたものが生成されるため、このソースコード上で宣言します。今回は InPort 名を in0、OutPort 名を out0 とし、型は両方とも TimedDoubleSeq としています。サンプルコードや変数等の詳細は開発者のサポートページを参照してください。今回の例は RTno_for_RGBLED.ino というファイルで GitHub においてあるため、そちらを参照してください。

ソースコードが書けたら、検証・書き込みを行います。この書き込みの際、必ずシリアルポートを確認してください。



確認後、シリアルポートの設定を行います。RTnoProxy.conf というファイルを開き、先ほど確認したポート番号に合わせて設定してください。

```
##########################################
# comport
##########################################
conf.default.comport:¥¥¥¥.¥¥COM4
#conf.default.comport:/dev/ttyACM0
#conf.default.comport:/dev/cu.usbmodem1411
```

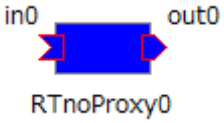
これで必要な設定は終了です。その後は以下の手順でコンポーネントの起動を行います。これら手順についても開発者のサポートページを参照してください。

1. ネームサーバの起動
2. rtc.conf でネームサーバの IP アドレスの設定
3. RTnoProxyComp.exe の起動

無事に起動すると以下のようなコンソールが表示されます。

```
Opening SerialPort(¥¥.¥COM4)..... OK.
Waiting for Startup the arduino...
3.....
2....
1..
Go!
Starting up onInitialize sequence.
-RTnoProtocol::getRTnoProfile() called.
-Parsing RTnoProfile.
-Success.
RTno Status == 73
Execution Context Type == 33
--ProxySynchronousExecutionContext detected!
onInitialized OK.
```

この表示が出たら、ネームサーバに RTnoProxy が追加されています。今回 RTnoProxy のポートは InPort、OutPort が一つずつなので、配置すると以下の図のようになります。

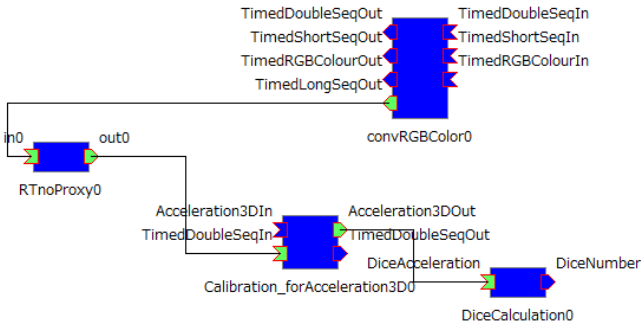


4.3 システム構築

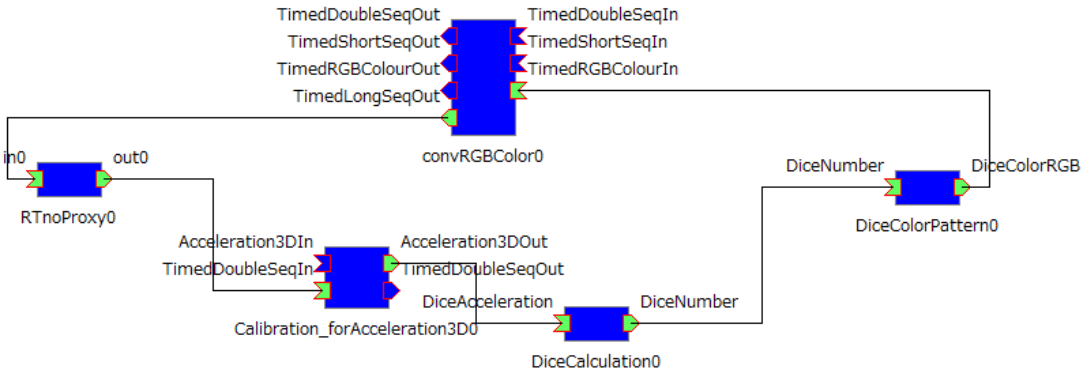
次に、システムの構築手順を説明します。まず、動作コンポーネントの接続を説明します。今回用いるコンポーネントの内、以下のコンポーネントが動作コンポーネントとなっています。

・サイコロ計算コンポーネント（DiceCalculation）
・3 軸加速度調整コンポーネント（Calibration_forAcceleration3D）
・RGB 変換コンポーネント（convRGBColor）

これらと RTnoProxy を接続した図は以下のようになります。



次に表現コンポーネントの接続を行います。今回用いるコンポーネントの内、サイコロ色表現コンポーネント（DiceColorPattern）が表現コンポーネントとなっています。これを他のコンポーネントに接続した図が以下のようになります。

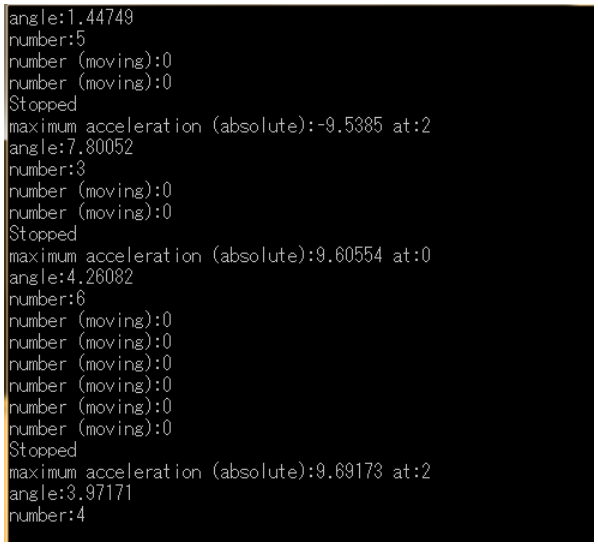


この状態でコンポーネントの接続は完了です。次に Configuration の設定についてです。実機を適切に動作させるためには、その実機に応じて動作コンポーネントの Configuration を設定する必要があります。

今回は RGB 変換コンポーネント（convRGBColor）と 3 軸加速度調整コンポーネント

（Calibration_forAcceleration3D）はデフォルトで利用します。実機との接続、キャリブレーション等必要な場合は変更してください。ここでは、サイコロ計算コンポーネント（DiceCalculation）の Configuration について説明します。

初めは Configuration を変更せずアクティブ化します。この時、DiceCalculation のコンソールは以下のようになります。



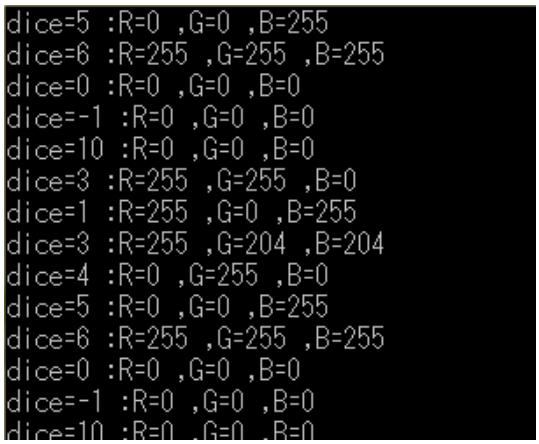
この判定を見ながら、DiceCalculation の Configuration を変更します。以下の基準を参考にしてください。

小	Configuration	大
停止判定が厳しい	SensorThreshold	停止判定が緩い
判定時間短い	StopTime Threshold	判定時間長い
水平垂直	SlopeThreshold	傾きに寛容

DiceCalculation の Configuration が適切に設定できているかどうかは、実機の動作で判断します。判断の基準は実現したい表現によって様々ありますが、6 面全ての出力結果が出れば基本動作としては問題ありません。

次に表現コンポーネントの Configuration について説明します。表現コンポーネントの Configuration では実機の表現部分を変更します。今回はサイコロ色表現コンポーネント（DiceColorPattern）を例に説明します。

現在は、デフォルトのまま実行を行っているため、出目に応じて赤、青、緑、ピンク、黄色、白に光ります。この色は DiceColorPattern の Configuration によって変更できるため、各目の数値を好きな RGB 値に設定してください。このとき DiceColorPattern のコンソールは以下のようになっているため、Configuration の値が正しく設定されているか確認できます。



後は、実現したい表現を目指して、実機の様子を見て細かい調整を行ってください。

以上で、システム構築は完了です。

5. プロトタイピングテストでの接続例

動作コンポーネントと表現コンポーネントは、実機とやり取りしている情報さえ分かれば開発することが可能です。したがって、実機がなくてもテストが可能です。また、特定の実機以外でも、ポートの型さえ合えば他のコンポーネントと接続できるため、別実機でのテストやプロトタイピングも可能となります。

今回は、別実機として加速度センサを持つ Andoroid 端末と、テスト用ツールコンポーネントとして開発した RGB 表示コンポーネント（RGBTester）を利用した場合の接続例を示します。使用するコンポーネントは以下の通りです。

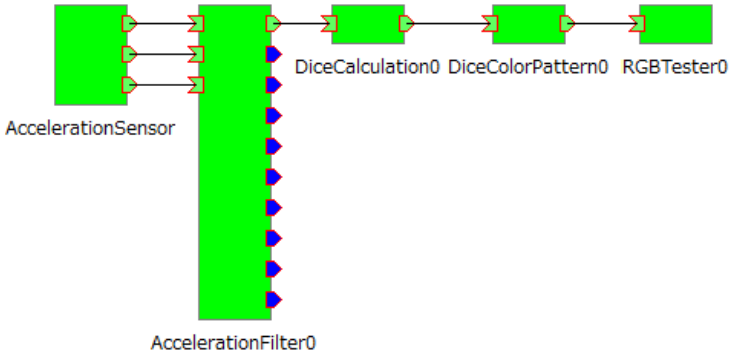
・サイコロ計算コンポーネント（DiceCalculation）
・サイコロ色表現コンポーネント（DiceColorPattern）
・RGB 表示コンポーネント（RGBTester）

また、Andoroid 端末から加速度センサを取得するため、“RTM on Android を用いた Android 用マルチセンサコンポーネント群”の中から以下のコンポーネントを利用しています。

・端末加速度取得コンポーネントアプリケーション(GetAcceleration)
・三軸加速度フィルタコンポーネント(AccelerationFilter)

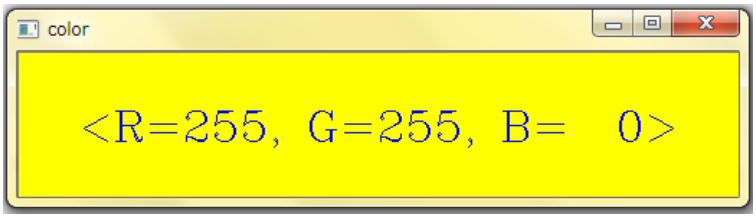
こちらのコンポーネントについては、開発者のサポートページ（http://www.openrtm.org/openrtm/ja/project/contest2013_1B3-3）にあるマニュアルを参照して起動を行ってください。

使用するコンポーネントを以下の図のように接続します。



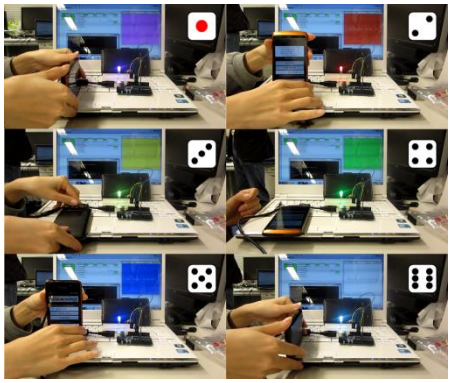
これにより、Android 端末から取得した加速度センサ値に対して 1~6 の目を算出し、それに応じた RGB を出力することができます。サイコロ計算コンポーネント（DiceCalculation）とサイコロ色表現コンポーネン（DiceColorPattern）の Configuration の設定方法については前章を参考にしてください。

出力結果は、RGB 表示コンポーネント（RGBTester）によって視覚的に確認できます。RGBTester の Configuration は表示ウィンドウのサイズを指定するものなので、必要があれば変更してください。デフォルトでの表示結果は以下のようになります。



このように、設定した RGB が表示されたら成功です。

次にこのシステムをプロトタイプに適応させます。プロトタイプとしては、Andoroid 端末の加速度センサと Arduino にフルカラーLED を接続したものを使用します。コンポーネントは RGBTester を前章の手順で起動した RTnoProxy に変更します。すると、以下のように、Android 端末の傾きに応じて LED の色が変更していることが確認されます。



6. 複数の実機を扱う方法

実機が複数に増えた場合も、一つの場合と同様に動作コンポーネントと表現コンポーネントの接続でシステムを構築することができます。また、必要があればコンポーネントを多重起動してあげることで、実機ひとつひとつに対してばらばらに動作や表現を設定することができます。今回は Movic Cube を 3 つ使って、それぞれのサイコロの出た目を覚えておき、全部の目が決定したら目が大きい順に、目の数に応じて LED を光らせていくシステムの例を示します。

まず、Movic Cube が 3 つあるので、4 章 2 節で説明した構成をそのまま 3 つ用意します。コンポーネントを多重起動するためには、各コンポーネントの rtc.conf を編集する必要があります。まず、実行ファイルと同じディレクトリに、rtc.conf の名前を変えて 3 つ（実機の個数分）用意します。その rtc.conf のなかを、それぞれ以下の赤文字の部分が別の名前になるように書き換えてください。

corba.nameservers:127.0.0.1
naming.formats: %h.host_cxt/DiceCalculation1.rtc

これにより、同じコンポーネントを複数立ち上げることができます。

次に、複数の実機が連携して動作・表現を行えるようにするため、複数の実機に対応した動作コンポーネント、表現コンポーネントを接続します。今回の例では、出た目をソートする動作コンポーネントの DiceSort、出た目を覚えて光らせるという表現コンポーネントの DiceMemory を接続します。これにより、複数の実機を扱うことができます。

7. コミュニティ

現在 OpenRTM-aist の公式 Web ページでは、プロジェクトとしてユーザが作成した様々なコンポーネントやツールを登録・利用することが出来ます。このように表現をコンポーネント化し、紹介・共有することが出来るという利点をメディアアートの分野で適応することが出来れば、制作プラットフォームとして RT ミドルウェアの使用者層が増えると考えております。また、RT 技術との連携も取れるため、共同開発のきっかけになり、お互いに感化し合える環境作りにも繋がると考えられます。このようなコミュニティ環境を提案するために、ブログ形式にて随時表現収集とコンポーネント化を行ってまいります。

ブログはこちら（http://rtmediaart.blog.fc2.com/）です。また、Twitter（@MediART2013：https://twitter.com/MediART2013）にて更新情報などを呟いているので、ぜひ興味が出た方はフォローしてください。現在ブログにて“サイコロ＋光る→Movic Cube”ということでプロトタイプの公開と表現案の募集を行っております。ぜひコメントなどでご提案・ご意見よろしくお願いいたします。

8. お問い合わせ

MovicCube および本コンポーネント群につきましては、発案に対し随時開発・公開をしているため、まだまだ展開・改善の余地があるものと考えております。提案、要望、バグ報告、マニュアル記述の不備等に関しましては下記までご連絡ください。

【問い合わせ先】

〒108-8548

東京都港区芝浦 3-9-14,611

芝浦工業大学大学院電気電子情報工学専攻

ロボティクスシステムデザイン研究室

土屋彩茜

Email:ma14076@shibaura-it.ac.jp

Blog：MediA-RT

URL：http://rtmediaart.blog.fc2.com/

Twitter:：@MediART2013

URL：https://twitter.com/MediART2013