

Conductor System

環境構築・利用マニュアル

目次

1. はじめに.....	2
2. 動作検証環境	2
2.1. 検証用 PC 環境	2
2.2. ソフトウェア環境	2
2.3. 使用可能ハードウェア	3
2.3.1. センサ	3
2.3.2. 指揮棒.....	3
3. 動作環境構築	4
3.1. Windows10 上の OpenRTM-aist 環境構築	4
3.2. Raspbian Jessie 上の OpenRTM-aist の環境構築.....	4
3.2.1. Raspbian のインストール	4
3.2.2. Raspbian の初期設定	5
3.2.3. Raspbian の初期アップデート	5
3.2.4. OpenRTM-aist の環境構築	6
3.3. iPhone Simulator の環境構築	6
4. Conductor System の利用方法	7
4.1. Conductor System RT コンポーネントの概要	7
4.2. Conductor System RT コンポーネント群の詳細	8
4.3. Conductor System 群のダウンロード・コンパイル方法	12
4.4. Conductor System の起動方法.....	15
4.5. Conductor System の利用方法.....	16

1. はじめに

本ドキュメントは，指揮について知識がないユーザや周りに楽器を演奏できる人がいないような状況においても指揮者のように曲をコントロールでき，指揮者の気分を感じることができることを目指して身近にあるようなマイコンやセンサを利用して作成した Conductor System の RT コンポーネント群の動作環境構築および利用方法について述べる．

作成した Conductor System は，OpenRTM-aist 上で動作するシステムであり，既存で公開されている MIDI ファイルを読み込み，曲を流すことができるコンポーネントを再利用することで，指揮者のように指揮棒を振ることで，曲のボリュームを変更することができる．

2. 動作検証環境

本ドキュメント作成にあたり，以下に示す検証環境を用いた．

2.1. 検証用 PC 環境

PC	MacBook Pro
	Raspberry Pi 3
OS	Windows10 (VirtualBox)
	macOS High Sierra 10.13.2
	Raspbian Jessie

2.2. ソフトウェア環境

ソフトウェア名	バージョン	備考
OpenRTM-aist C++	1.1.0-RELEASE	ミドルウェア
OpenRTM-aist python	1.1.0-RELEASE(32bit 用)	ミドルウェア
Python	2.7.10(32bit 用)	OpenRTM-aist python の動作に必要
PyYAML	3.11(32bit 用)	OpenRTM-aist python の動作に必要
RTSystemEditorRCP		RT システム構築に利用．
CMake	2.8.8	RT コンポーネントのコンパイル環境構築に必要．
Doxygen	1.8.1	RT コンポーネントに関するドキュメント生成に必要．
Xcode	9.2	iPhone7 simulator の動作に必要

2.3. 使用可能ハードウェア

2.3.1. センサ

本ドキュメントでは、3 軸加速度センサ LSM6DS33 から I2C 通信を介してデータ取得が可能である。

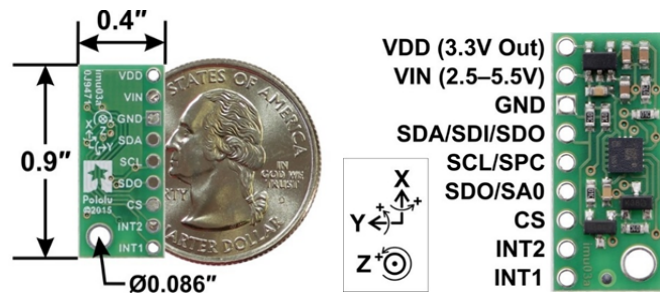


Fig.1 3 軸加速度センサ LSM6DS33

2.3.2. 指揮棒

本ドキュメントでは、3 軸加速度センサ LSM6DS33 を搭載した指揮棒を利用し棒を振る際の加速度データを取得する。センサデータは有線接続により Raspberry Pi 3 へ送られる。双方への接続状態を以下の Fig.2 に示す。

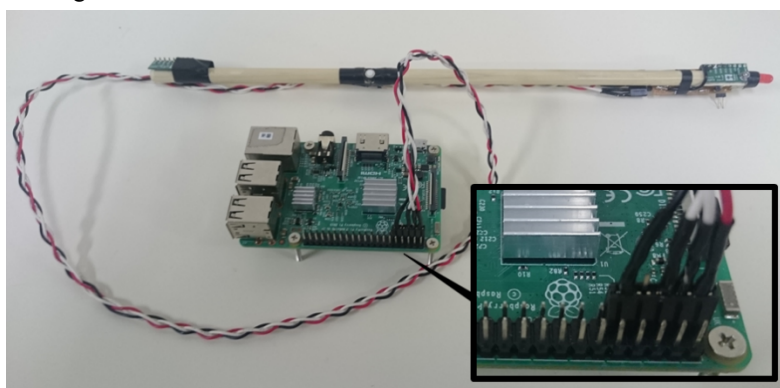


Fig.2 指揮棒・Raspberry Pi 3 の構成例

3. 動作環境構築

3.1. Windows10 上の OpenRTM-aist 環境構築

ここでは、楽器演奏のための MIDI コンポーネントを Windows10 上で動作させるための環境構築方法について述べる。

OpenRTM-aist-Python-1.1.0-RELEASE のインストールを行う。下記の URL から 32bit 用の Python2.7.10 と PyYAML-3.11, OpenRTM-aist-Python1.1.0 のインストーラーをダウンロードし実行する。Python は OpenRTM-aist や PyYAML をインストールする前にインストールする必要がある。

① **Python2.7.10(32bit):**

<https://www.python.org/ftp/python/2.7.10/python-2.7.10.msi>

② **PyYAML-3.11(32bit):**

<http://pyyaml.org/download/pyyaml/PyYAML-3.11.win32-py2.7.exe>

③ **OpenRTM-aist-Python-1.1.0(32bit):**

http://openrtm.org/pub/Windows/OpenRTM-aist/python/OpenRTM-aist-Python_1.1.0-RELEASE_x86.msi

OpenRTM-aist の開発支援ツールである RTSystemEditorRCP を動作させるために、下記の URL から OpenRTP の「全部入りパッケージ」をインストールする。また、RTSystemEditorRCP の動作には OpenRTM-aist-C++が必要であるので、以下の URL からインストーラーをダウンロードしてインストールを行う。

④ **「全部入りパッケージ」**

<http://openrtm.org/pub/openrtp/packages/1.1.0.rc5v20151111/eclipse381-openrtp110rc5v20151111-ja-win32.zip>

⑤ **OpenRTM-aist-C++ 1.1.0-RELEASE**

http://www.openrtm.org/pub/Windows/OpenRTM-aist/cxx/1.1/OpenRTM-aist-1.1.0-RELEASE_vc10.msi

3.2. Raspbian Jessie 上の OpenRTM-aist の環境構築

ここでは、Conductor System を Raspbian Jessie 上で動作させるために環境構築について述べる。

3.2.1. Raspbian のインストール

下記に示す方法は、Raspbian の OS を書き込むための一つの方法であるので、やりにくいのであれば別の手段をとっても構わない。

① 下記のサイトから Raspbian jessi をパソコンにダウンロードする。

<https://downloads.raspberrypi.org/raspbian/images/raspbian-2017-07-05/2017-07-05-raspbian-jessie.zip>

- ② SD カードに Raspbian の OS を書き込むため、Etcher という書き込みソフトを下記のソフトからダウンロードし、インストールする。

<https://etcher.io>

- ③ Etcher を用いて、SD カードに Raspbian の OS を書き込む。

3.2.2. Raspbian の初期設定

Raspberry Pi の初期設定を行うために、コマンドプロンプトで下記のコマンドを入力する。その後、設定を選択していく。

```
$ sudo raspi-config
```

- ① Timezone の変更
「4 Localisation Options」 → 「I2 Change Timezone」 → 「Asia」 → 「Tokyo」
- ② キーボードレイアウトの変更
「4 Localisation Options」 → 「I3 Change Keyboard Layout」 → 「Generic 105-key (Intl) PC」 → 「Other」 → 「Japanese」 → 「Japanese」 → 「The default for the keyboard layout」 → 「No compose key」
- ③ WiFi の変更
「4 Localisation Options」 → 「I4 Change Wi-Fi Country」 → 「JP Japan」
- ④ I2C の有効化
「5 Interfacing Options」 → 「P5 I2C」 → 「Yes」
- ⑤ SD カードの有効領域拡大（初期設定では一部しか使えないため）
「7 Advanced Options」 → 「A1 Expand Filesystem」
- ⑥ 設定終了後、再起動を行う。
「Finish」 → 「Yes」

3.2.3. Raspbian の初期アップデート

デスクトップ右上の WiFi のアイコンを選択肢 WiFi に接続をしておく。下記のコマンドを実行し、raspbian のアップデートを行う。

```
$ sudo apt-get update  
$ sudo apt-get upgrade
```

3.2.4. OpenRTM-aist の環境構築

OpenRTM の「Ubuntu/Debian へのインストール」のページへ移動する.

<http://openrtm.org/openrtm/ja/node/1182>

「一括インストールスクリプト(Debian)」を右クリックし、シェルスクリプトをダウンロードして、実行する

http://svn.openrtm.org/OpenRTM-aist/trunk/OpenRTM-aist/build/pkg_install_debian.sh

```
$ cd
$ cd Download
$ sudo sh pkg_install_debian.sh
```

OpenRTM python 版を下記のコマンドを実行しインストールする.

```
$ sudo apt-get update
$ sudo apt-get install python
$ sudo apt-get install python-omniorb-omg omniidl-python
```

3.3. iPhone Simulator の環境構築

ここでは、iPhone Simulator を動作させるための環境構築について述べる. iPhone Simulator はアプリ開発ツールである Xcode 実行すれば起動できる. そのため Mac の App Store から Xcode をインストールする. また、App Store には App ID が必要なため、持っていない場合は下記の URL から App ID を取得する.

<https://appleid.apple.com/account#!&page=create>

4. Conductor System の利用方法

Conductor System では MIDI コンポーネントから読み込まれた音楽の選択、音量調整の機能を提供する。以下にこれらに利用する RT コンポーネントやアプリの準備から利用方法までを示す。

4.1. Conductor System RT コンポーネントの概要

下記に Conductor System RT コンポーネントおよび既存の RT コンポーネントである MIDI コンポーネントを接続した様子を Fig. 3 に示す。

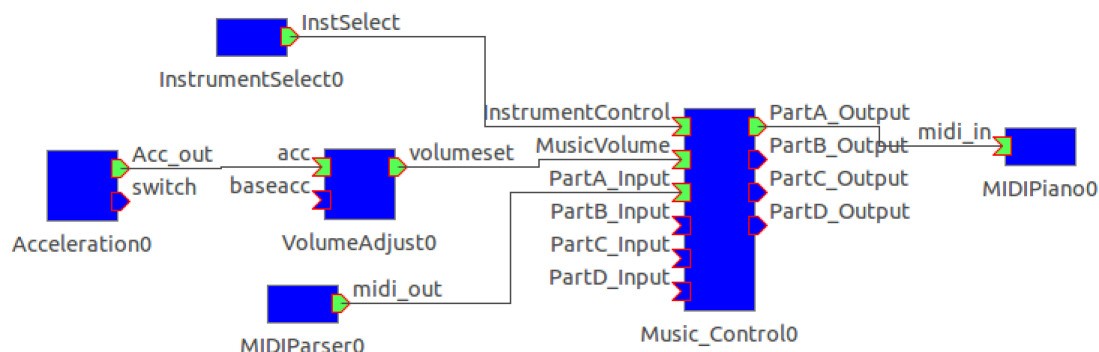


Fig. 3 指揮演奏のための RT コンポーネントの接続図

上記のシステムでは、以下のような特徴を持つ


- 4つまでの MIDI ファイルの読み込み、再生が可能
- LSM6DS33 からの加速度情報により、音量調整
- iPhone による再生音楽の選択

4.2. Conductor System RT コンポーネント群の詳細

ここでは、Conductor System に用いる各 RT コンポーネントの仕様を示す。


◆ MIDIParser

SMF 形式(.midi)のファイル中の音楽データを MIDI メッセージとして出力する RT コンポーネント

RTC の名称		
MIDIParser	 midi_out MIDIParser0	
出力ポート		
名称	データ型	データ
midi_out	MIDI::MIDIMessage	MIDI ファイルからの出力
主なコンフィグレーションパラメータ		
名称	機能	
midi_file	読み込む MIDI ファイルの選択	

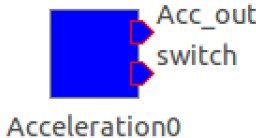
◆ MIDIPiano

MIDI::Message 型のデータを受けてソフトウェアシンセサイザを制御する RT コンポーネント

RTC の名称		
MIDI Piano	 MIDI Piano0	
入力ポート		
名称	データ型	データ
midi_in	MIDI::MIDIMessage	MIDI メッセージ
主なコンフィグレーションパラメータ		
名称	機能	
channel	MIDI メッセージのチャンネルの設定	
tone	ソフトウェアシンセサイザの音色の設定	
delay_time	MIDI メッセージの遅延をキャンセルするための待ち時間の設定	
device_name	ソフトウェアシンセサイザの名前の指定	

◆ Acceleration

LSM6DS33 から加速度を取得する RT コンポーネント

RTC の名称		
Acceleration		
出力ポート		
名称	データ型	データ
Acc_out	TimedDouble	加速度センサからの出力
switch	TimedBoolean	スイッチからの出力
主なコンフィグレーションパラメータ		
名称	機能	
port_name	COM ポートの指定	
baudrate	ボーレートを指定	
debug	Debug モードの ON/OFF	


◆ VolumeAdjust

合成加速度によって音量設定値を出力する RT コンポーネント。加速度の基準値を入れることによって、ユーザにあったキャリブレーションの設定が可能となる。

RTC の名称		
VolumeAdjust		
入力ポート		
名称	データ型	データ
acc	TimedDouble	合計加速度
baseacc	TimedDouble	基準加速度
出力ポート		
名称	データ型	データ
volumeset	TimedDouble	音量設定値
主なコンフィグレーションパラメータ		
名称	機能	
base	音量の基準値の設定	

◆ InstrumentSelect

iPhone と通信し，iPhone アプリの値を出力する RT コンポーネント

RTC の名称		
InstrumentSelect		
出力ポート		
名称	データ型	データ
InstSelect	Conductor::CoutrolMusic	オドメトリによる現在の推定自己位置
主なコンフィグレーションパラメータ		
名称	機能	
portnumber	TCP/IP 通信のポート番号設定	
timeout_sec	通信待ち時間の設定	

◆ Music_Control

ConductorSystem 統合コンポーネント

RTC の名称		
Music_Control	<div><div>InstrumentControl</div><div>MusicVolume</div><div>PartA_Input</div><div>PartB_Input</div><div>PartC_Input</div><div>PartD_Input</div><div>PartA_Output</div><div>PartB_Output</div><div>PartC_Output</div><div>PartD_Output</div><div>Music_Control0</div></div>	
入力ポート		
名称	データ型	データ
InstrumentControl	Conductor::ControlMusic	楽器コントロール
MusicVolume	TimedDouble	音量調整
PartA_Input	MIDI::MIDIMessage	楽器 A の MIDI データ
PartB_Input	MIDI::MIDIMessage	楽器 B の MIDI データ
PartC_Input	MIDI::MIDIMessage	楽器 C の MIDI データ
PartD_Input	MIDI::MIDIMessage	楽器 D の MIDI データ
出力ポート		
名称	データ型	データ
PartA_Input	MIDI::MIDIMessage	楽器 A の MIDI データ
PartB_Input	MIDI::MIDIMessage	楽器 B の MIDI データ
PartC_Input	MIDI::MIDIMessage	楽器 C の MIDI データ
PartD_Input	MIDI::MIDIMessage	楽器 D の MIDI データ
主なコンフィグレーションパラメータ		
名称	機能	
debug	デバックモード	

4.3. Conductor System 群のダウンロード・コンパイル方法

Conductor System に利用する RT コンポーネントやアプリのダウンロード・コンパイル手順について示す.

- Windows10 上のコンポーネント

- ① MIDI ファイル読み込み コンポーネント

下記の URL から MIDIParser をダウンロードして, デスクトップ上に解凍する.

<https://github.com/HiroakiMatsuda/MIDIParser>

- ② MIDI 出力 コンポーネント

下記の URL から MIDIParser をダウンロードして, デスクトップ上に解凍する.

<https://github.com/HiroakiMatsuda/MIDIPiano>

MIDI コンポーネントの 1 つである MIDIPiano は MIDIIO ライブラリに依存しているため必要なライブラリをダウンロードする. 下記の URL から MIDIIO ライブラリ 1.0 (MIDIOLib1.0.zip) をダウンロードする.

<http://openmidiproject.osdn.jp/MIDIIOLibrary.html>

ダウンロードした MIDIOLib1.0.zip を適当な場所に解凍し, MIDIOLib1.0¥Release¥に格納されている MIDIO.dll をコピーする. コピーした MIDIO.dll を MIDIPiano フォルダ内にある pymidiio フォルダの直下に配置する.

※ 複数起動する場合は, MIDIParser-master フォルダ内と MIDIPiano フォルダ内の rtc.conf を下記のように変更する.

Rtc.conf ファイルの 218 行目

naming.formats: %h.host/%n.rtc

↓

naming.formats: %h.host/%n/%p.rtc

- Raspberry Pi 上のコンポーネント

下記のコマンドを実行し, Conductor System コンポーネントをダウンロードする.

```
$ cd
$ cd workspace
$ git clone https://github.com/
```

① 加速度取得 コンポーネント

上記でダウンロードした Conductor System 内の Acceleration が加速度取得のコンポーネントである。python のコンポーネントなので、コンパイルは必要ない。

② 音量調整 コンポーネント

上記でダウンロードした Conductor System 内の VolumeAdjust が音量設定値を出力するコンポーネントである。下記のコマンドを実行し cmake まで行う。

```
$ cd
$ cd workspace/ConductorSystem/RTC/VolumeAdjust
$ mkdir build
$ cd build
$ cmake ..
```

ここで、Raspberry Pi ではポート設定などの **idl ファイルが関係づけられていない**。そのため make を行う前にコンポーネントに必要なファイルをコピーする必要がある。

ConductorSystem/idl/BasicData フォルダ内の全てのファイルを下記の場所にコピーする。

workspace/ConductorSystem/RTC/VolumeAdjust/build/idl

そして、make をしてコンパイルを実行する。

```
$ make
```

③ 音楽管理マネージャー コンポーネント

上記でダウンロードした Conductor System 内の Music_Control が音楽管理マネージャーである。下記のコマンドを実行し cmake まで行う。

```
$ cd
$ cd workspace/ConductorSystem/RTC/Music_Control
$ mkdir build
$ cd build
$ cmake ..
```

ここで、Raspberry Pi ではポート設定などの **idl ファイルが関係づけられていない**。そのため make を行う前にコンポーネントに必要なファイルをコピーする必要がある。

ConductorSystem/idl/BasicData と ConductorSystem/idl/ConductorData,

ConductorSystem/idl/MIDIData フォルダ内の全てのファイルを下記の場所にコピーする。

workspace/ConductorSystem/RTC/Music_Control /build/idl

そして、make をしてコンパイルを実行する。

```
$ make
```

④ iPhone との通信 コンポーネント

上記でダウンロードした Conductor System 内の InstrumentSelect が iPhone との通信コンポーネントである。下記のコマンドを実行し cmake まで行う。

```
$ cd
$ cd workspace/ConductorSystem/RTC/InstrumentSelect
$ mkdir build
$ cd build
$ cmake ..
```

ここで、Raspberry Pi ではポート設定などの **idl ファイルが関係づけられていない**。そのため make を行う前にコンポーネントに必要なファイルをコピーする必要がある。

ConductorSystem/idl/ConductorData フォルダ内の全てのファイルを下記の場所にコピーする。

workspace/ConductorSystem/RTC/InstrumentSelect/build/idl

そして、make をしてコンパイルを実行する。

```
$ make
```

⑤ Raspberry Pi 上の全コンポーネント

Raspberry Pi で実行したコンポーネントの状態を別のパソコンの eclipse や RTSystemEditorRCP で確認するとき、Raspberry Pi のパソコン名が表示されずにコンポーネントが確認できないことがある。そのため、各コンポーネントの rtc.conf を変更し、コンポーネントの起動を IP アドレスの直下に変更する。

rtc.conf ファイルの 218 行目

```
# naming.formats: %h.host/%n.rtc
```

↓

```
naming.formats: %n.rtc
```

● iPhone Simulator のアプリ

Mac 上に下記の URL から ConductorSystem をダウンロードし、Instrument_Selection をデスクトップ上に移動する。

<https://github.com/>

4.4. Conductor System の起動方法

指揮者体験を行うためには、4.3 においてダウンロード・コンパイルしたコンポーネントやアプリを起動する必要がある。

- Windows10 上のコンポーネント・プログラム

- ① Start Python Naming Service を実行する
- ② RTSystemEditorRCP を実行する
- ③ MIDIParser-master フォルダ内の MIDIParser.py を実行する
再生する音楽を変える場合は、MIDIParser コンポーネントのコンフィグレーションで
ある midi_file のパラメータを再生したい mid ファイルに変更をする。
- ④ MIDIPiano-master フォルダ内の MIDIPiano フォルダ内にある MIDIPiano.py を実行する。

- Raspberry Pi 上のコンポーネント

RT コンポーネント毎にターミナルを起動する。

- ① 加速度取得コンポーネントを下記のコマンドにより実行する。

```
$ cd
$ cd workspace/ConductorSystem/RTC/Acceleration
$ python Acceleration.py
```

- ② 音量設定値を出力するコンポーネントを下記のコマンドにより実行する。

```
$ cd
$ cd workspace/ConductorSystem/RTC/VolumeAdjust/build/src
$ ./VolumeAdjustComp
```

- ③ 音楽管理マネージャーコンポーネントを下記のコマンドにより実行する。

```
$ cd
$ cd workspace/ConductorSystem/RTC/MusicControl/build/src
$ ./MusicControlComp
```

- ④ iPhone との通信コンポーネントを下記のコマンドにより実行する。


```
$ cd
$ cd workspace/ConductorSystem/RTC/InstrumentSelect/build/src
$ ./InstrumentSelectComp
```

※Raspberry Pi の IP アドレスを利用するので、下記のコマンドにより IP アドレスを確認しておく。

```
$ ifconfig
```

● iPhone Simulator

ダウンロードした Instrument_Selection フォルダ内の Instrument_Selection.xcodeproj を開くことによって、楽器選択アプリを Xcode で開く。Xcode の左上にある iPhone のアイコン(set the activate schene)から iOS Simulator を iPhone7 に選択し、左上にあるスタートボタン（ビルドボタン）から実行する。

4.5. Conductor System の利用方法

RTSystemEditorRCP において、指揮者体験を行うための RT システム(複数の RT コンポーネントを組み合わせで構築されるシステム)を構築する。以下に手順を示す。

- ① 「Open New System Editor」をクリックし、システムエディタを立ち上げる。
- ② ネームサーバの追加を選択し、上記で調べた Raspberry Pi のネームサーバを追加する。
- ③ ネームサーバに登録されている RT コンポーネントをすべて、システムエディタ上にドラック&ドロップする。
- ④ 配置した RT コンポーネントを Fig.4 のように結線する。複数の音楽を流す際は Fig.5 のように結線する。
- ⑤ InstrumentSelect を選択し、ポート番号を確認しておく。
- ⑥ Fig.6 の左に示す iPhone アプリの IP アドレスに Raspberry Pi の IP アドレスを、ポート番号に InstrumentSelect で調べたポート番号を入力する。
- ⑦ ここまで設定したら、ジョイスティック、移動ロボット、レーザレンジファインダが PC につながっていることを確認し、「All Activate」をクリックする。
- ⑧ 赤くなるコンポーネントがなければ正常に起動が完了している。
- ⑨ その後、Fig.6 の左に示す iPhone アプリの Connect_Btn を押すことにより、iPhone アプリと Raspberry Pi と通信が可能となり、Fig.6 の右の図に移る。

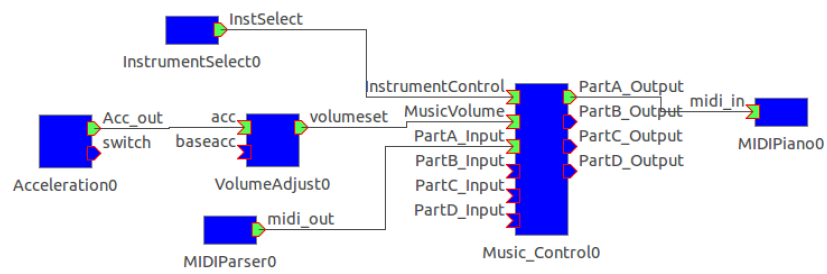


Fig.4 構築する RT システム例 1

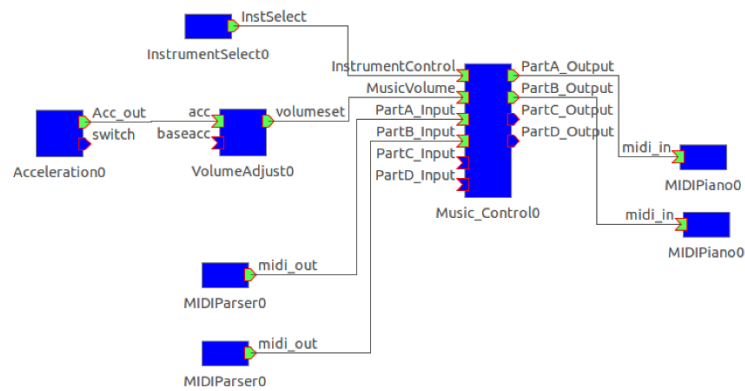


Fig.5 構築する RT システム例 2

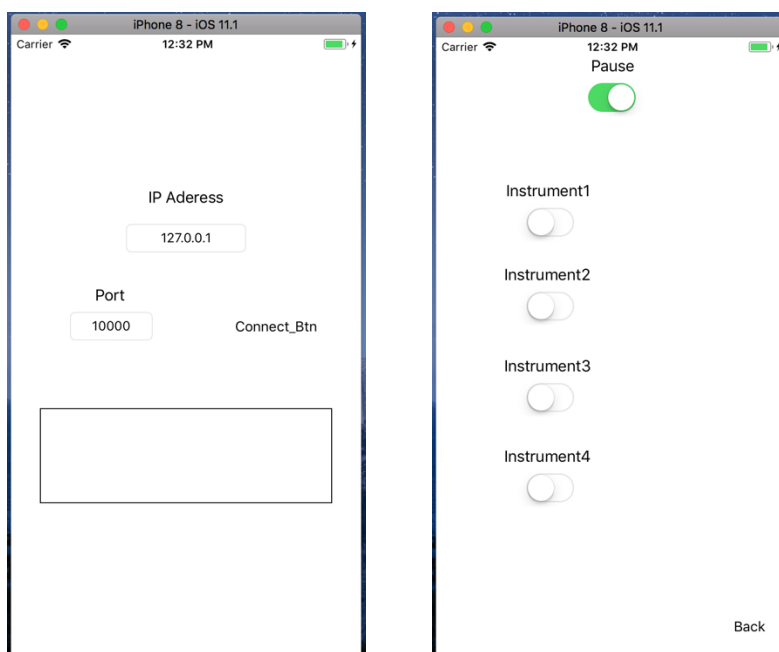


Fig. 6 iPhone アプリ

RTSystemEditorRCP で「All Activate」を実行したと同時に、音楽が再生される。最初に再生する音量は最大音量の 50%であり、指揮棒を振ることにより音量の変化が確認できる。また、Fig.4 の右側に示す iPhone アプリの Instrument ボタンを選択することにより、それぞれ選択した音楽の音量を消音 ON/OFF にすることが確認できる。