次世代ロボット知能化技術開発プロジェクト

施設内生活支援ロボット知能の研究開発

音声合成モジュール(SpeechSynthesizer)

マニュアル

Ver. 2.2

2012 年 1 月

九州工業大学

改変履歴

Ver.	改版日	改変内容
1.0	2010/3	新規作成
1.1	2010/11	ドキュメントの改訂
2.0	2011/12/21	OpenRTM-1.0.0対応、音声対話共通API対応、モジュール名変更
2.1	2012/1/4	64bit 対応の説明を修正
2.2	2012/1/24	ドキュメントの不要な部分を削除

1. はじめに

1.1. 本文書について

本書は、「次世代ロボット知能化技術開発プロジェクト」の「施設内生活支援ロボット知能研究開発」において構築した音声認識モジュールについてのマニュアルです。本書は RT ミドルウェア（以下 RTM）、RT コンポーネント（以下 RTC）を用いたロボットシステム開発者を対象に記述されており、RTM、RTC や関連ツールに関する一般的な知識を持つことを前提とします。

OpenRTM-aist official webside: http://www.openrtm.org/openrtm/ja/

1.2. 対象プラットホーム

本モジュールは、以下の環境で動作確認しています。

- Windows XP professional 32bit
- Windows Vista Business 32bit/64bit
- Windows 7 Professional/Enterprise 32bit/64bit

1.3. OpenRTM-aist-C++ 関係

以下のソフトウェアをインストールしてください。

- Microsoft Visual Studio 2008 Professional Edition/Express Edition
- OpenRTM-aist-1.0.0-RELEASE_vc9_100212.msi

2. 音声合成モジュール(SpeechSynthesizer)

音声合成エンジンとして AquesTalk を使用する音声合成モジュールのための RTC(Provider)です。AquesTalker 自体は、Win32,Linux などをサポートしていますが、ここでは Win32 版を前提としています。

モジュールのディレクトリは SpeechSynthesizer となります。

このモジュールは、入力された発話用文字列を音声合成エンジンである AquesTalk に渡して音声サウンドデータを合成し、OpenHRI の PortAudioOutput モジュール等を利用して出力します。

2.1. インストール

RTC 再利用センター登録の zip ファイルをダウンロードしてください。

音声合成モジュール: SpeechSynthesizer.2.0.zip

以下の手順で eclipse にプロジェクトをインポートします。

1. eclipse を起動する。
2. ファイルメニューから「インポート」を選択し、「一般」の「既存のプロジェクトをワークスペースへ」を選ぶ。
3. 次に進み、「アーカイブ・ファイルの選択」の欄にチェックし、「参照」で SpeechSynthesizer.2.0.zip を選択する。
4. 「完了」を選択する。
5. 環境に合わせて、SpeechSynthesizer / rtm_config.vsprops ファイルを調整する。
   環境に合わせて、SpeechSynthesizer / user_config.vsprops ファイルを調整する。
   詳細については OpenRTM のドキュメントを参照してください。
6. Microsoft Visual Studio より次のソリューションファイルを読み込ませる。
   SpeechSynthesizer_vc9.sln

2.2. ビルド

1. ビルドには、Microsoft Visual Studio 2008、OpenRTM の環境(OpenRTM C++、OmniORB など)が必要となります。
2. SpeechSynthesizer のソリューションのビルドを実行して下さい。
3. ビルドが完成したら、
   SpeechSynthesizer/SpeechSynthesizer/Debug,Release
   SpeechSynthesizer/SpeechSynthesizerComp/Debug,Release
   に実行ファイルや DLL ファイルが作成されます。
4. モジュールを動作させる対象のディレクトリには、AquesTalk の DLL ファイルなどをコピーします。必要なファイルについては、SpeechSynthesizer / components を参照して下さい。

なお、AquesTalker / components に、Win32 向けにビルド済みのファイルを格納しています。

2.3. 起動

起動時の設定ファイルである SpeechSynthesizer / rtc.conf の内容を確認して下さい。

- corba.nameservers: CORBA ネームサーバのホスト名とポート番号

  例: corba.nameservers: localhost:5005

次にコマンドプロンプトを開き、SpeechSynthesizer に移動します。

先にビルドした SpeechSynthesizerComp.exe を実行します。

```
C:${SRP_HOME}/SpeechSynthesizer> components¥SpeechSynthesizerComp.exe
```

（ここでは、SpeechSynthesizer / components にビルド済みのファイルがコピーされているとしています。）

起動後は特にウィンドウが表示されることはありません。Initialized: Speech Synthesizer Component と表示されれば成功です。

2.4. ポートについて

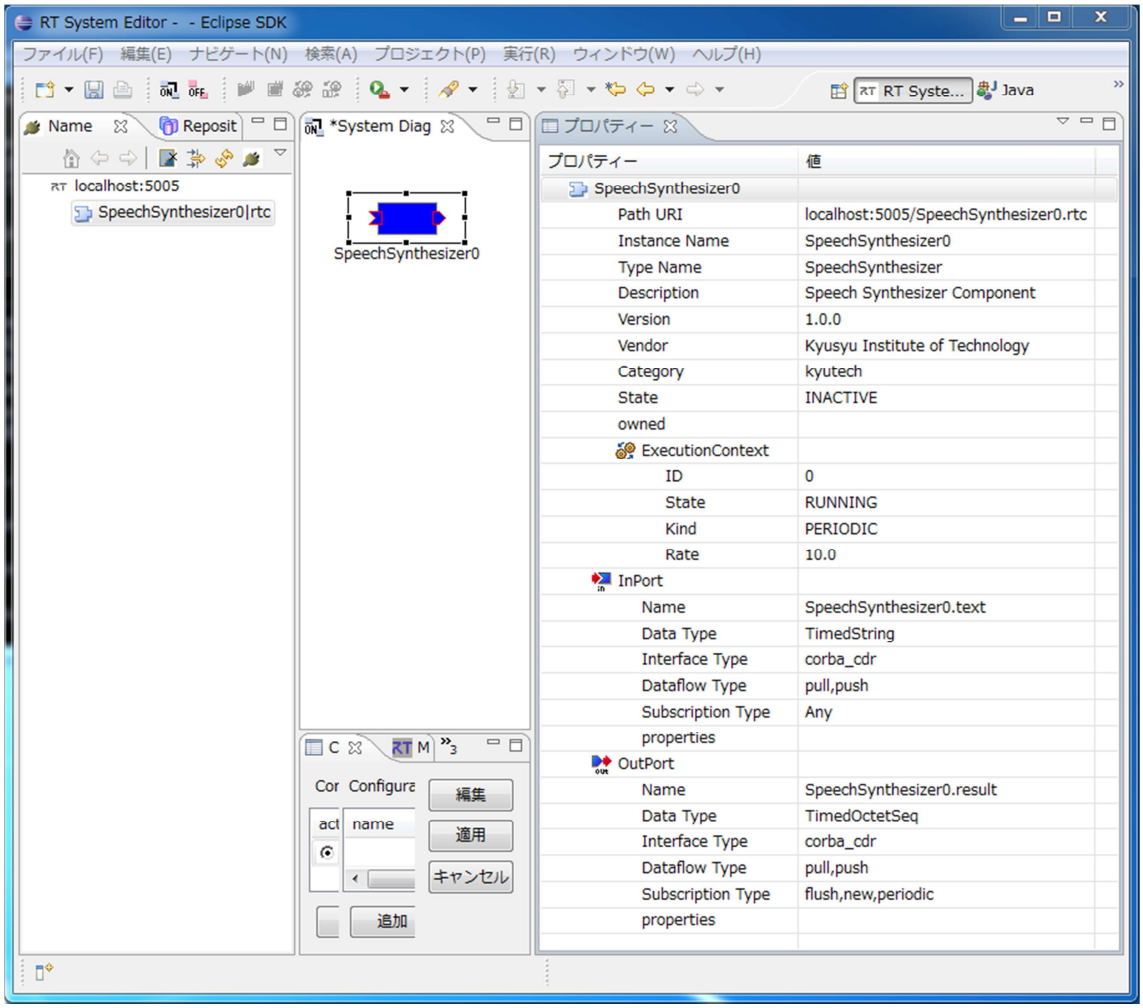
SpeechSynthesizer コンポーネントは、OpenHRI の音声合成コンポーネントである OpenJtalkRTC と入出力ポートレベルで互換があります。出力ポートに PortAudioOutput モジュール等を接続して出力します。

データポートの一覧表

入出力	ポート名	データ型	説明
入力	text	TimedString	音声合成するためのテキスト
出力	result	TimedOctetSeq	合成された音声データ

※出力音声波形は、8KHzサンプリング, 16bit,モノラル,WAVフォーマット となります。

入出力ポートのプロパティ



2.5. 動作検証

ここでは、作業計画モジュールと組み合わせた動作検証方法を紹介します。

1. スタートメニューから OpenHRI → audio → portaudiooutput を起動します。
2. portaudiooutput モジュールはデフォルトで、16KHz,16bit の WAV 形式を前提にしていますが、SpeechSynthesizerComp の出力は AquesTalk の制約上、8KHz でしか出力できません。そこで、portaudiooutput 側でサンプリングレートを 8KHz に変更します。これにより、正常な速度で発話できます。
3. SpeechSynthesizerComp を起動します。
4. SDLEngine を起動します。
5. 以下のスクリプトを SDLEngine Console で実行します。

```
env = rtc.env("localhost", 5005);
sdl_engine = rtc.local_component("SDLEngine", "SDLEngine");
env.get_handles();
env.connect(env.handles{"SDLEngine0.rtc"}.ports{"SDLEngine0.stringOut"},
            env.handles{"SpeechSynthesizer0.rtc"}.ports{"SpeechSynthesizer0.text"});
env.connect(env.handles{"SpeechSynthesizer0.rtc"}.ports{"SpeechSynthesizer0.result"},
            env.handles{"PortAudioOutput0.rtc"}.ports{"PortAudioOutput0.AudioDataIn"});
env.handles{"SDLEngine0.rtc"}.activate();
env.handles{"SpeechSynthesizer0.rtc"}.activate();
env.handles{"PortAudioOutput0.rtc"}.activate();
```

続けて、以下の行を入力すると “あいうえお” と発話します。

```
sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("あいうえお");
```

発話に使用するテキストのデータは AquesTalk のドキュメントに従って作成します。いくつか例を示します。

```
sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("こんにちわ。");
sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("かんコーヒー。");
sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("おちゃ。");

sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("も'もちゃん、フルーツジュースを/もってきて。");
sdl_engine.local_ports{"SDLEngine0.stringOut"}.put("たなかさん、かんコーヒー'でいいですか？");
```

また、<NUM VAL=数字> とすると数字として読み上げてくれます。以下に 301号室、302号室、303号室と読み上げる例です。

```
n=300;
for (i=1; i<=3; ++i) {
    sdl_engine.local_ports["SDLEngine0.stringOut"].put("<NUM VAL="+(n+i)+">ごうしつ。");
}
```

動作検証例のモジュールの接続図