RT コンポーネント操作マニュアル

圧力計測と音声出力を組み合わせた生活を彩るアプリケーション群の開発

2019 年 10 月 29 日

芝浦工業大学付属高等学校

白井哲平

更新情報

・2019 年 10 月 29 日　初版
・2019 年 11 月 25 日　7 ページ「Flag_builder」を修正
・2019 年 12 月 5 日　7 ページ「Flag_builder」を修正
・2019 年 12 月 6 日　9 ページ「DX ライブラリの準備」を修正

目次

1　まえがき

1.1　開発コンセプト

近年，情報産業や電子機器の発達により生活は便利になったが，生活は無機的になった．多くの事が自動化され，今後も自らアクションを起こす機会は失われていく．そこで我々は自らが起こしたアクションが何かに反映されるアプリケーションを製作し，生活に彩りを添えることを試みる．

Nintendo Switch のヒットや音楽ゲームに根強い人気があるように，自身が起こしたアクションが何かに反映されるものは人々に親しまれている．また最近では，ASMR(Autonomous Sensory Meridian Response)の流行や Bluetooth イヤホンが普及し，音も身近なものになっている．我々はこの 2 つの要素を組み合わせ，生活を豊かに彩ることができないかと考えた．

日常生活では，ドアノブに手をかけた時，ペンを握った時，歩くだけでも圧力は発生する．そこで我々は，圧力の観点から人のアクションを感知し，音声出力に反映するアプリケーションを製作することとした．

1.2　概要

我々は圧力センサーが圧力を感知すると音声を再生するアプリケーション群を製作した．それらの基本要素となる，「圧力センサーの測定値から力を求め，力の値が一定値を超える度に 1 回音声を再生する」までの処理をコンポーネント群として実装した．

1.3　開発環境

OS: Windows 10
RT ミドルウェア: OpenRTM-aist 1.2.0-RELESE (C++版)
開発環境: Visual Studio Community 2017
CMake: CMake 3.14.0
Python: Python 3.7.2

2　ハードウェア

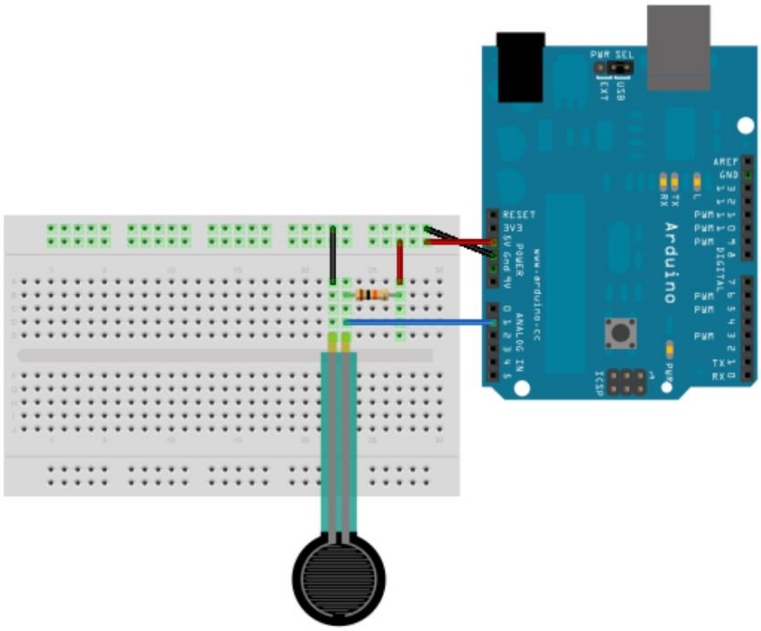
本節では，アプリケーションに必要なハードウェアと回路，設定方法を示す.

2.1　用意するもの

- Force Sensing Resistor (FSR)
- Arduino
- PC
- ブレッドボード
- 抵抗

2.2　配線

Arduino の 5V の電源とアナログピンを使い，下図(Fig.1)のように配線を組む.



(Fig.1　6 節[3]より引用)

2.3　プログラムの書き込み

Arduino IDE を用いて Arduino にスケッチを書き込む.

3　コンポーネントの説明

我々が製作したアプリケーションを構成するコンポーネントは 4 つである．そのうち「RTnoProxy」「DataProcessing」は既存のコンポーネントを使用している．本節では，「RTnoProxy」「Flag_builder」「sound_outer」の仕様を示す．

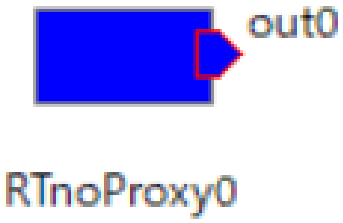
3.1　RTnoProxy

概要

RT コンポーネントと Arduino（もしくは Arduino コンパチのデバイス）との間の通信を簡単化する．

RTno パッケージは Arduino 用のライブラリ「RTno」と，それと通信する「RTnoProxy」という RT コンポーネントのセット．もし，Arduino のテンプレートに従ってプログラムをすれば，RTno は RTnoProxy を通して，他の RT コンポーネントと通信出来る．

本アプリケーションでは，Arduino のアナログピンを使用し，圧力センサーにかかる電圧を得るために用いている



・InPort

なし

・OutPort

名称	データ型	説明
out	TimedDouble	電圧のアナログデータ

・Configuration 変数

なし

本アプリケーションでは使用する圧力センサーの数に応じてスケッチを変更し，OutPort の数を増やした．

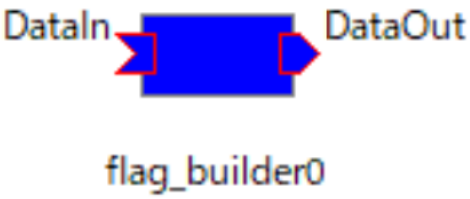
3.2　Flag_builder

概要

InPort から数値を受け取り，その数値が閾値を超える度に OutPort から値を出力する．InPort の値が閾値を超えたときの値の上昇度(傾き)の大小によって OutPort の値が変動するようになっている．この上昇度(傾き)を三角関数のタンジェント[tan]とし，これをサイン[sin]に変換した値を OutPort から出力する．

変換時には三角関数の公式から導出した下記の式を利用した．

$$\sin = \sqrt{1 - \frac{1}{\tan^2 + 1}}$$



flag_builder0

・InPort

名称	データ型	説明
DataIn	TimedDouble	数値を受け取る．

・OutPort

名称	データ型	説明
DataOut	TimedDouble	閾値を超えた後の dt 秒間における値の上昇度[tan]を[sin]に変換した 0～1 の値を出力する．

・Configuration 変数

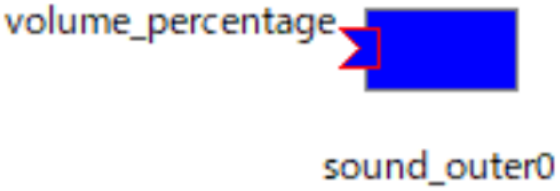
名称	データ型	デフォルト値	説明
Flag	TimedDouble	5000	閾値の役割をする．
dt	TimedDouble	0.5	何秒間の InPort の値の上昇度を測るかを決める．

使用時は「Flag」と「dt」を任意の値に変更する．

3.3　sound_outer

概要

DX ライブラリを使用した音声出力，同時に音量調節を行う．InPort から最大音量に対する音量を割合で取得することで音量を調節する．コンフィギュレーションで再生する音楽ファイルを変更できる．



・InPort

名称	データ型	説明
volume_percentage	TimedDouble	0～1.0 までの値を最大音量に対する割合として取得する．

・OutPort

なし

・Configuration 変数

名称	データ型	デフォルト値	説明
Sound_File	string	hirate_uti.mp3	再生する音楽ファイルを指定する．

音楽ファイルは「sound_outer ファイル」→「build ファイル」→「src ファイル」→「Debug ファイル」内に入れる．

音楽ファイルを「Debug ファイル」に保存したらデフォルト値に設定されている「hirate_uti.mp3」を消し，音楽ファイルのファイル名を入力する．

4　基本要素の動かし方

我々が製作したアプリケーションは基本的に「RTnoProxy」「DataProcessing」「Flag_builder」「sound_outer」，この 4 つのコンポーネントで構成されている．

4.1　作業手順

4.1.1　ハードウェアの準備

2 節に従いハードウェアの準備を行う．

4.1.2　OpenRTM-aist 及び Name サーバの起動

4.1.3　RT コンポーネントのダウンロード・起動

下記の 4 つのコンポーネントをダウンロードする．(3)(4)は本プロジェクトのホームページから，(1)(2)は 5 節の参考文献に掲載したサイトからダウンロードする．

(1) RTnoProxy　(http://ysuga.net/?p=124)
(2) DataProcessing　(http://openrtm.sakura.ne.jp/cgi-bin/wiki/wiki.cgi/2010/2B21)
(3) Flag_builder
(4) sound_outer

4.1.4　DX ライブラリの準備

DX ライブラリ置き場(https://dxlib.xsrv.jp/)から DX ライブラリをダウンロードする．

CMake から準備する場合

[1]　src フォルダ内「CMakeLists.txt」に以下の赤文字を書き加える

```
set(comp_srcs sound_outer.cpp )
set(standalone_srcs sound_outerComp.cpp)
set(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE} /MT")
set(CMAKE_CXX_FLAGS_DEBUG "${CMAKE_CXX_FLAGS_DEBUG} /MTd")

if(${OPENRTM_VERSION_MAJOR} LESS 2)
  set(OPENRTM_CFLAGS ${OPENRTM_CFLAGS} ${OMNIORB_CFLAGS})
  set(OPENRTM_INCLUDE_DIRS ${OPENRTM_INCLUDE_DIRS} ${OMNIORB_INCLUDE_DIRS})
  set(OPENRTM_LIBRARY_DIRS ${OPENRTM_LIBRARY_DIRS} ${OMNIORB_LIBRARY_DIRS})
endif()

if (DEFINED OPENRTM_INCLUDE_DIRS)
  string(REGEX REPLACE "-I" ";"
    OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")
  string(REGEX REPLACE " ;" ";"
    OPENRTM_INCLUDE_DIRS "${OPENRTM_INCLUDE_DIRS}")
endif (DEFINED OPENRTM_INCLUDE_DIRS)

if (DEFINED OPENRTM_LIBRARY_DIRS)
  string(REGEX REPLACE "-L" ";"
    OPENRTM_LIBRARY_DIRS "${OPENRTM_LIBRARY_DIRS}")
  string(REGEX REPLACE " ;" ";"
    OPENRTM_LIBRARY_DIRS "${OPENRTM_LIBRARY_DIRS}")
endif (DEFINED OPENRTM_LIBRARY_DIRS)

if (DEFINED OPENRTM_LIBRARIES)
  string(REGEX REPLACE "-l" ";"
    OPENRTM_LIBRARIES "${OPENRTM_LIBRARIES}")
  string(REGEX REPLACE " ;" ";"
    OPENRTM_LIBRARIES "${OPENRTM_LIBRARIES}")
endif (DEFINED OPENRTM_LIBRARIES)

include_directories(${PROJECT_SOURCE_DIR}/include)
include_directories(${PROJECT_SOURCE_DIR}/include/${PROJECT_NAME})
include_directories(${PROJECT_BINARY_DIR})
include_directories(${PROJECT_BINARY_DIR}/idl)
include_directories(${OPENRTM_INCLUDE_DIRS})
include_directories("C:/…/DxLib_VC3_20f/DxLib_VC/プロジェクトに追加すべきファイル_VC 用")
add_definitions(${OPENRTM_CFLAGS})
add_definitions(-D_MBCS)
```

```
MAP_ADD_STR(comp_hdrs "../" comp_headers)

link_directories(${OPENRTM_LIBRARY_DIRS} "C:/…/DxLib_VC3_20f/DxLib_VC/プロジェクトに追加すべきファイル_VC 用")

add_library(${PROJECT_NAME} ${LIB_TYPE} ${comp_srcs}
  ${comp_headers} ${ALL_IDL_SRCS})
set_target_properties(${PROJECT_NAME} PROPERTIES PREFIX "")
set_source_files_properties(${ALL_IDL_SRCS} PROPERTIES GENERATED 1)
if(NOT TARGET ALL_IDL_TGT)
  add_custom_target(ALL_IDL_TGT)
endif(NOT TARGET ALL_IDL_TGT)
add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES})

add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
  ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
add_dependencies(${PROJECT_NAME}Comp ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES})

install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}Comp
    EXPORT ${PROJECT_NAME}
    RUNTIME DESTINATION ${INSTALL_PREFIX} COMPONENT component
    LIBRARY DESTINATION ${INSTALL_PREFIX} COMPONENT component
    ARCHIVE DESTINATION ${INSTALL_PREFIX} COMPONENT component)

install(FILES ${PROJECT_SOURCE_DIR}/RTC.xml DESTINATION ${INSTALL_PREFIX}
        COMPONENT component)
```

ただし include_directories と link_directories の赤文字部分については DX ライブラリフォルダ内「プロジェクトに追加すべきファイル__VC 用」ファイルへのパスを入力してください.

[2]　書き終えたら CMake する.

[3]　「sound_outer.sln」を開き，ビルドを行う.

ソリューションファイルから準備する場合

[1]　CMake を行い「sound_outer.sln」を開く.

[2]　「sound_outerComp」を右クリックし，プロパティを押す.

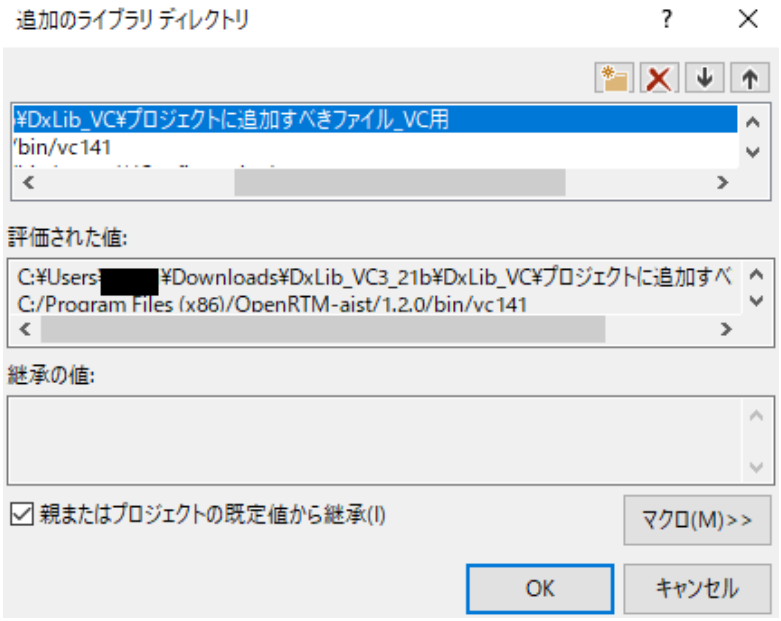
[3] ダイアログの左上にある「構成(C):」と書かれている項目を「アクティブ(Debug)」から「すべての構成」に変更する.

[4] ①ダイアログの左側のリストから「構成プロパティ」→「全般」を選ぶ.
②ダイアログ右側に表示されている「文字セット」の項目を「マルチ バイト文字セットを使用する」に変更する.

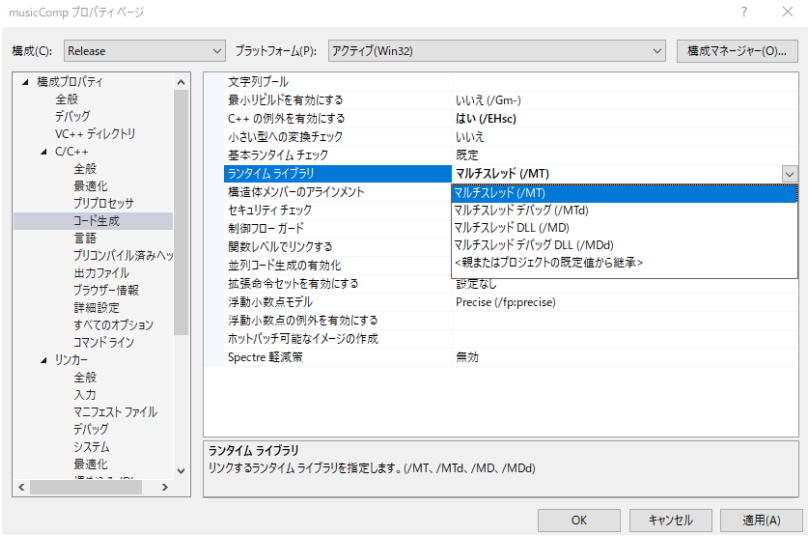
[5] ①左側のリストから「構成プロパティ」→「Ｃ／Ｃ＋＋」→「全般」を選ぶ.
②右側に表示されている「追加のインクルードディレクトリ」の項目に，ＤＸライブラリのパッケージ内に入っている「プロジェクトに追加すべきファイル_VC 用」フォルダのパスを入力する.

[6] ①左側のリストから「構成プロパティ」→「リンカー」→「全般」を選ぶ.

②右側に表示されている「追加のライブラリディレクトリ」の項目に[5]②と同じＤＸライブラリの「プロジェクトに追加すべきファイル_VC 用」フォルダのパスを入力する.



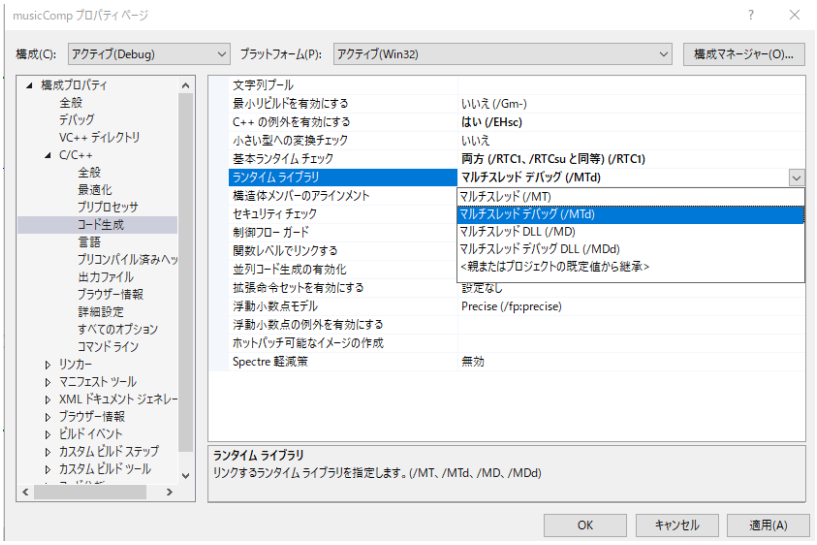
[7] ①ダイアログの左上にある「構成(C):」と書かれている項目を「すべての構成」から「Release」に変更する.
②左側のリストから「構成プロパティ」→「Ｃ／Ｃ＋＋」→「コード生成」を選ぶ.
③ダイアログ右側に表示されている「ランタイム ライブラリ」の項目を「マルチスレッド(/MT)」に変更する.



[8] ①ダイアログの左上にある「構成(C):」と書かれている項目を「Release」から「Debug」に変更する.

②ダイアログ右側に表示されている「ランタイム ライブラリ」の項目を「マルチスレッドデバッグ(/MTd)」に変更する．

[9] ダイアログの下の方にある「ＯＫ」を押してダイアログを閉じる

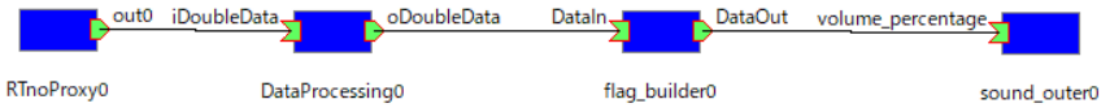


[10]「sound_outer」を右クリックし，プロパティを押す．

[11] [3]から[9]までをもう一度繰り返す．

[12] ソリューションのビルドを行う．

4.1.5　コンポーネントの接続実行



(Fig.2)

コンポーネントを上図(Fig.2)のように配置，接続する．

4.1.6　コンポーネントの設定

4.1.5(Fig.1)の「DataProcessing」では「RTnoProxy」から出力される電圧の値を力の値に変換するための式を入力する．

2 節の通りの配線をし，抵抗が 1kΩ、電源電圧（Arduino から回路に供給される電圧）が 5V の場合

[1] Constantlist に「1:5」と入力する．（「:」の前が抵抗値［kΩ］，後が電源電圧値[V]）

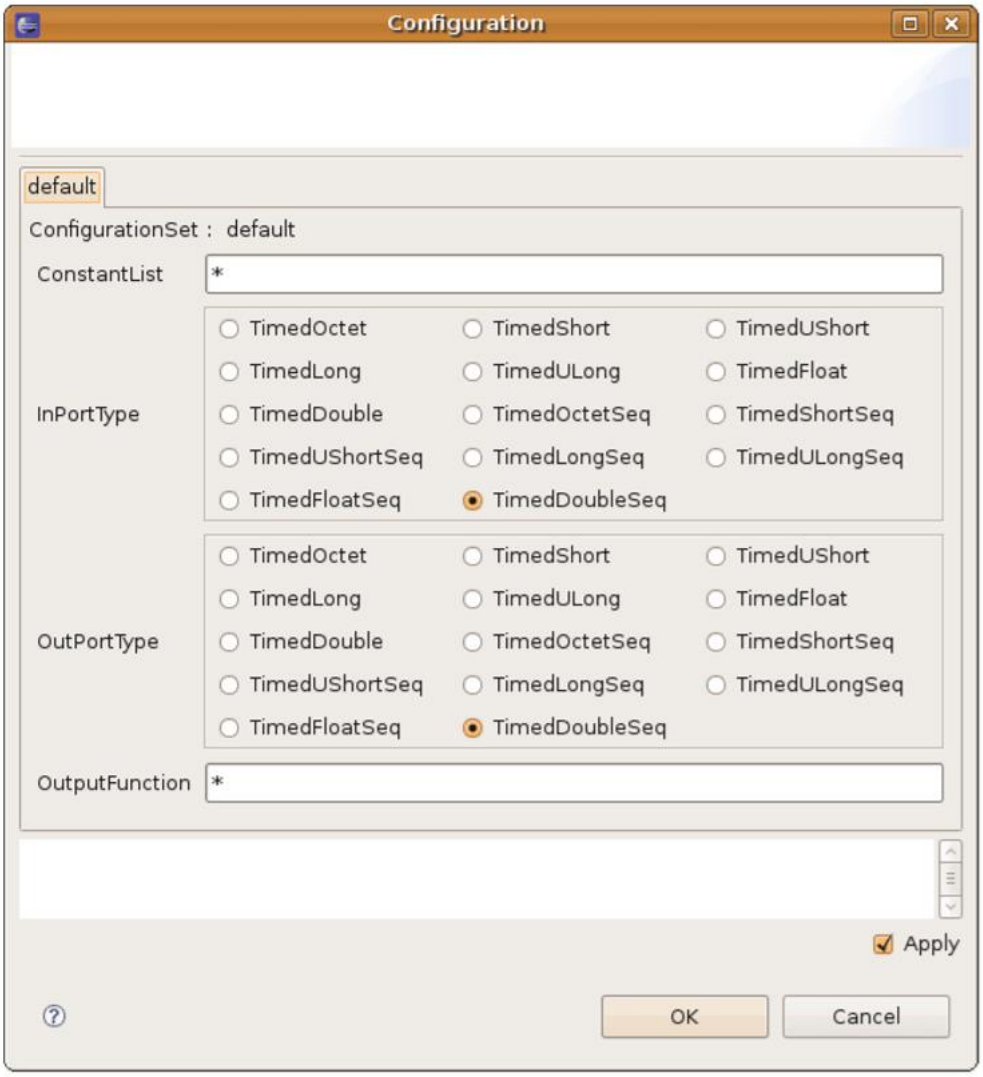
[2] InPortType のデータ型を「TimedDouble」にする．

[3] OutPortType のデータ型を「TimedDouble」にする．

[4] OutputFunction に下記の式を入力する．

「1278.5*(x/(C(0)*(C(1)-x)))^(1.326)」

[5] OK をクリックする



また, アプリケーションに応じて「Flag_builder」「sound_outer」のコンフィギュレーション変数の設定が必要（仕様は前節）.
※詳しい「RTnoProxy」「DataProcessing」の仕様は 6 節のリンクでホームページにアクセスしマニュアルを参照してください.

5　応用例

5.1　ストラックアウトクイズ

本校の文化祭で公開した体験型のゲーム．これは野球のストラックアウトにクイズを追加したゲームである．的の裏に圧力センサーを設置しボールが衝突すると正解と不正解の時で違う音声を出力する．このアプリケーションでは前節で述べたコンポーネント以外に，正解を判別するコンポーネントとクイズを表示し正解番号を出力するコンポーネントを製作した．

ボールが圧力センサーに力を加える時間が一瞬で，センサーが反応しない場合があったため，展示時は手で押してもらう形になった．

5.2　足音

靴の裏に圧力センサーを取り付け，歩くと効果音を流す．

5.3　水槽の減少を知らせるアプリ

魚を飼育するための水槽などの水は，蒸発によって少しずつ水が減っていく．そこで圧力センサーを水槽の下に設置し，水の減少をセンサーに加わる力が減少したことから感知し，音を鳴らす．

5.4　伝言メッセージ

玄関のマットに圧力センサーを設置する．誰かが家に帰り，マットを踏むと録音したメッセージが再生される．

6　参考文献

[1]菅佑樹：“RT コンポーネント対応デバイスを開発するためのマイコン用ライブラリ&ツール「RTno」の開発”，第 12 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, pp.816-818 (2011)
(http://ysuga.net/?p=124)

[2]佐々木毅，橋本秀紀：“汎用データ処理のための演算コンポーネント”，第 11 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, pp.1063-1064 (2010)
(http://openrtm.sakura.ne.jp/cgi-bin/wiki/wiki.cgi/2010/2B21)

[3]圧力センサーで遊ぼう
(http://jkoba.net/prototyping/arduino/pressure_practice.html)