ゴミ捨て動作により映像が変化するディスプレイ付きゴミ箱の開発

芝浦工業大学付属高等学校

久保田浩平

内容

1　はじめに 2

1.1 概要 2

1.2 開発環境 2

2　ハードウェア 3

3　ソフトウェア 4

3.1 RTnoProxy(電圧計測用コンポーネント) 4

3.2　distance 5

3.3　music 6

3.4 Create Tapioca 7

4　本コンポーネントを利用するまでの手順 8

4.1　コンポーネントのビルド 8

4.2　music の準備 [5] 8

4.3 Create Tapioca の準備 12

4.4 コンポーネント接続 14

5 システム利用 15

参考文献 16

1　はじめに

1.1 概要

我々はゴミ箱にディスプレイを取り付け，人がゴミを捨てるとそれに反応して映像が変化するゴミ箱を制作した．映像は物理エンジンで作り，入れたゴミをタピオカに変換して落ちてくるようなものにした．

1.2 開発環境

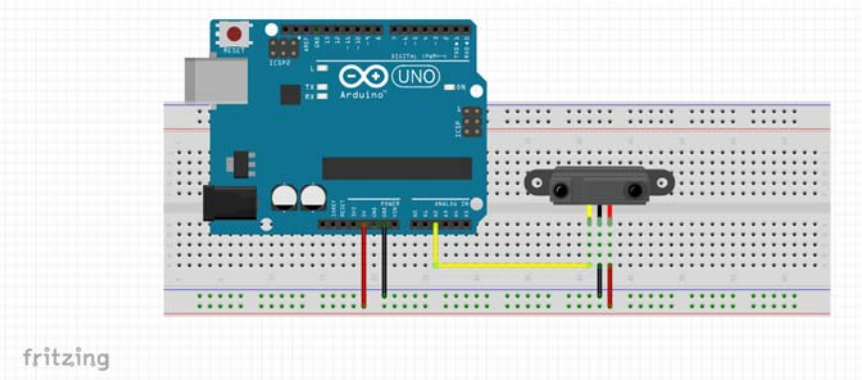
本 RTC の開発環境を以下に示す

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

今回の制作に使用した部品をまとめた．下記は距離センサーの配線図である．

部品名	型番	個数
距離センサー	GP2Y0A21	1
Arduino	UNO	1



また，今回我々はゴミ箱を作成した．以下はその材料である．実際にソフトのみを起動する場合は必要ない．

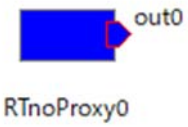
材料名	個数
プラスチック段ボール　縦 180mm 横 90 ㎜	10
塩ビ板 透明	5
スプレー(黒 装飾用)	1
カラーボール(黒 装飾用)	20

3　ソフトウェア

本システムは，既存のコンポーネントである「RTnoProxy [1] (電圧計測用コンポーネント)」，「DataProcessing [2] (計測値処理用コンポーネント)」と，新たに製作した「distance」，「music」，「CreateTapioca」の 5 つの RTC から構成される．

3.1 RTnoProxy(電圧計測用コンポーネント)

RTnoProxy は Arduino 用のライブラリ「RTno」[1]を用いて他のコンポーネントと通信するコンポーネントである．今回は距離センサーのデータを TimedDouble 型で出力するために用いる．



・InPort

なし

・OutPort

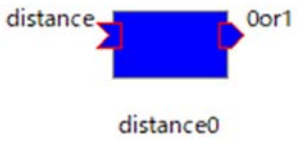
名称	データ型	説明
out0	TimedDouble	電圧のアナログデータ

・Configuration 変数

なし

3.2 distance

距離が変化したことを検出するコンポーネントである．InPort から値を受け取り，初回の値と現在の値の差の絶対値が Configuration 変数の『kyori』で決めた値より大きければ 1，小さければ 0 を出力する．



InPort

名称	データ型	説明
distance	TimedDouble	現在のセンサーからの距離.

・OutPort

名称	データ型	説明
0or1	TimedLong	0 か 1 を出力する.

・Configuration 変数

名称	データ型	デフォルト値	説明
kyori	int	5	基準の距離.

3.3 music

DX ライブラリ[3]を用いて，InPort から受け取ったデータが 0 から 1 になったときに決まった音声（タピオカが液体に落ちる音）を出力する．

InPort

名称	データ型	説明
1or0	TimedLong	0 と 1 の入力を受け取る．

・OutPort

なし

・Configuration 変数

なし

・依存ライブラリ

DX ライブラリ https://dxlib.xsrv.jp

3.4 Create Tapioca

このコンポーネントは InPort から受け取ったデータが 0 から 1 になったときにディスプレイ上でタピオカを追加する．また，出力されたタピオカの数をカウントし，40 個タピオカを出すと容器内をリセットするようになっている．物理エンジンには BulletPhysics[4]を用いている．

InPort

名称	データ型	説明
addTapi	TimedLong	受け取る数字が 0 から 1 になったときにタピオカを出力する．

・OutPort

なし

・Configuration 変数

なし

・依存ライブラリ

Bullet Physics SDK https://pybullet.org/wordpress/

4　本コンポーネントを利用するまでの手順

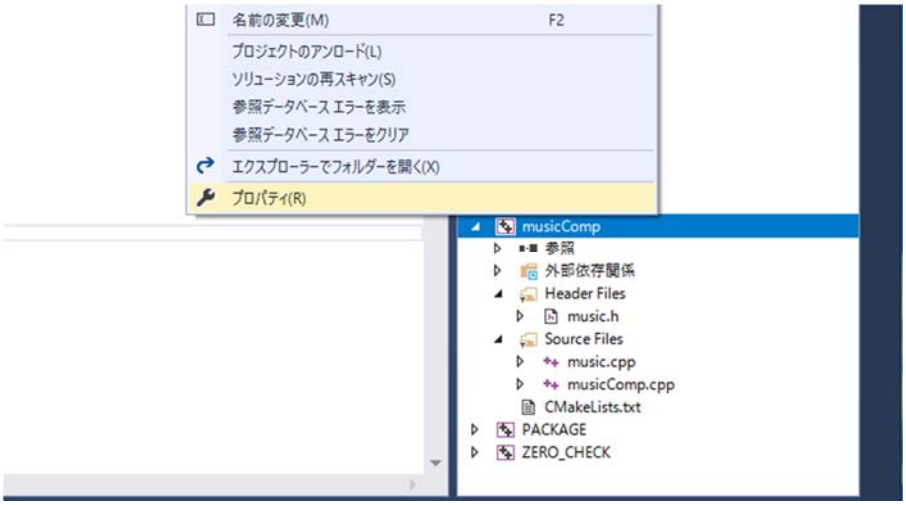
4.1　コンポーネントのビルド

『Create Tapioca』以外のダウンロードしたコンポーネントをビルドする．今回は CMake と Visual Studio を用いた．

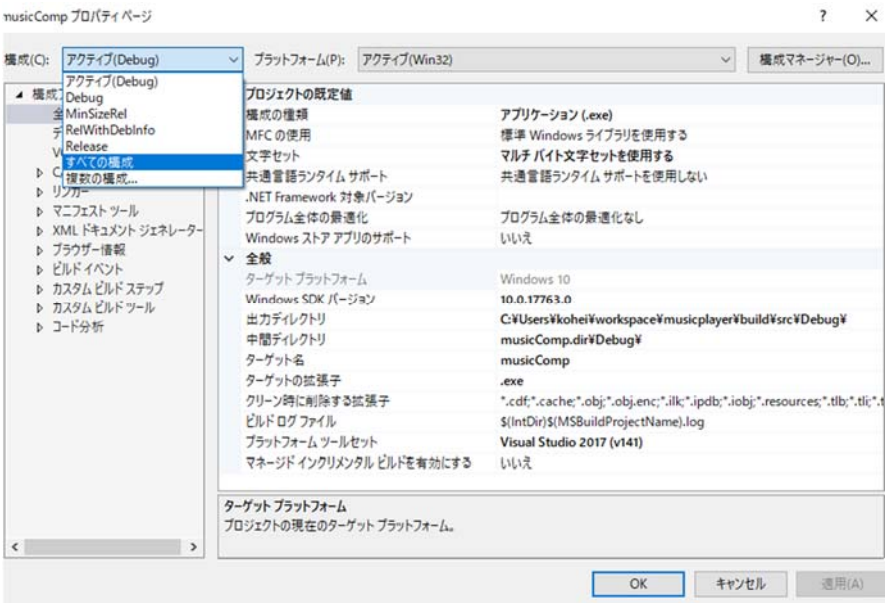
4.2　music の準備［5］

[1] build フォルダの中にある『music.sln』を開く

[2] 『musicComp』を右クリックし，プロパティを押す．

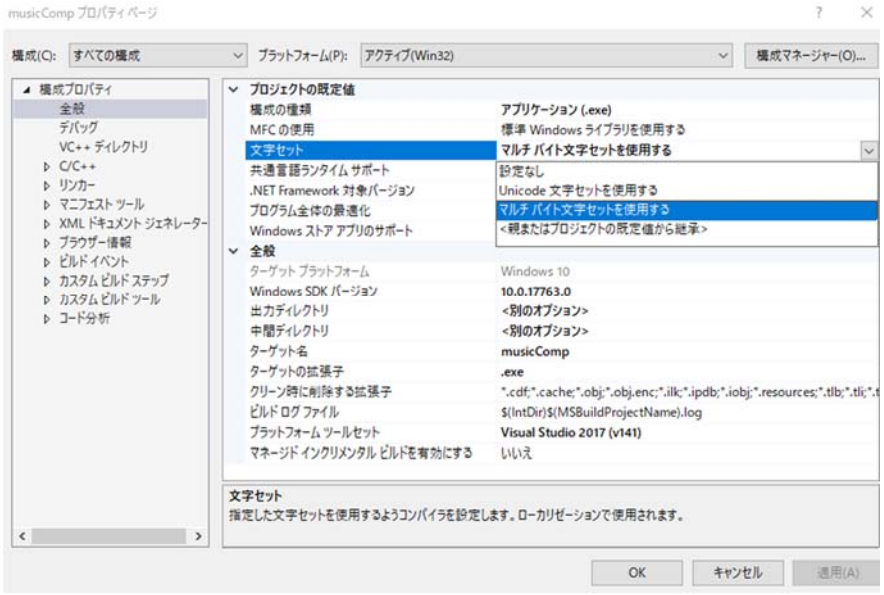


[3] ダイアログの左上にある『構成(C):』と書かれている項目を『アクティブ(Debug)』から『すべての構成』に変更する．



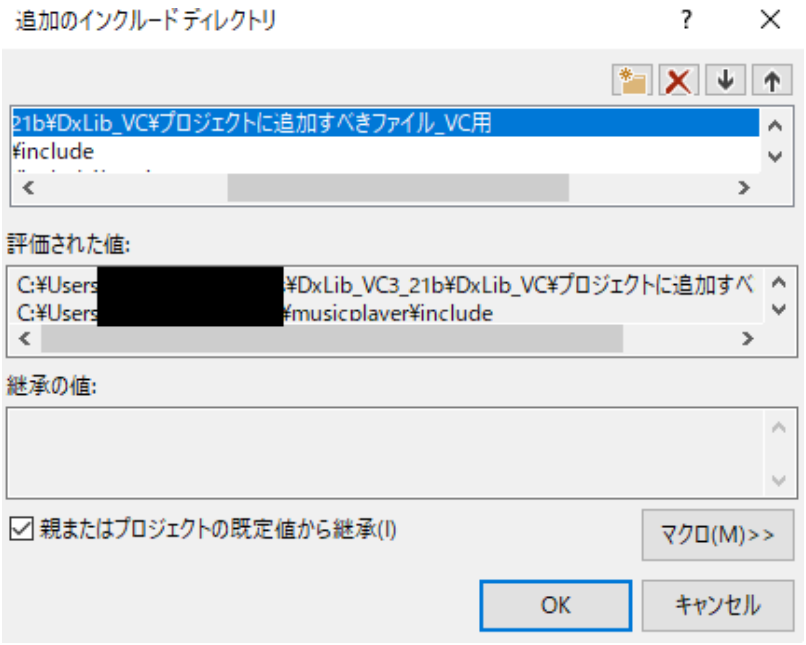
[4] ①ダイアログの左側のリストから『構成プロパティ』→『全般』を選ぶ．

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.



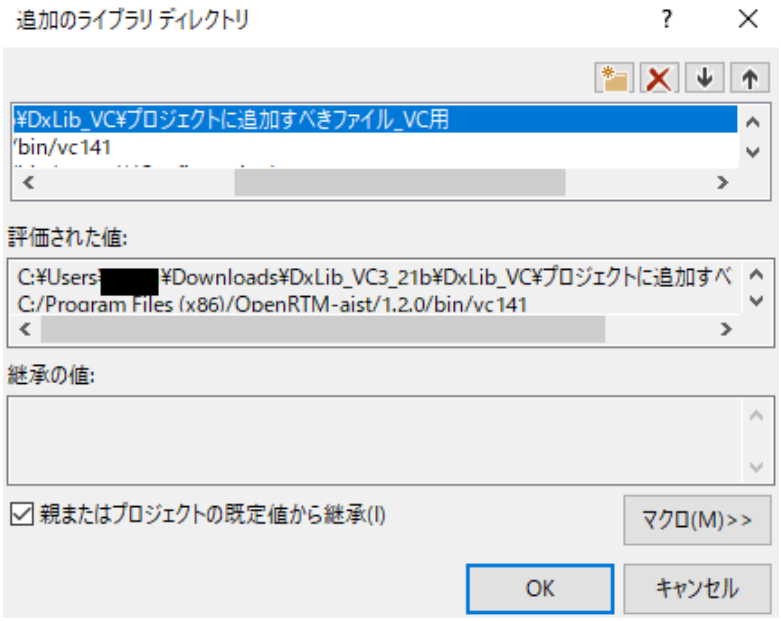
[5] ①左側のリストから『構成プロパティ』→『C／C＋＋』→『全般』を選ぶ.

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「https://dxlib.xsrv.jp/dxuse.html」からダウンロードできる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.

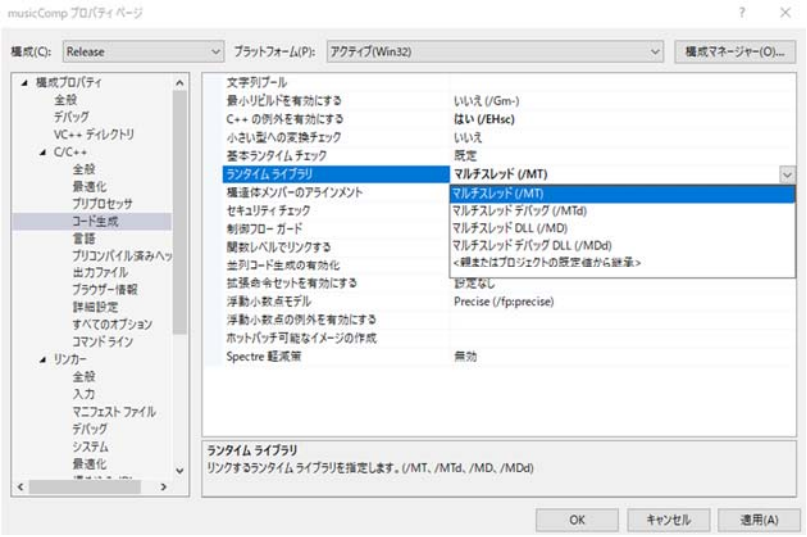
②右側に表示されている『追加のライブラリディレクトリ』の項目に[5]②と同じＤＸライブラリの『プロジェクトに追加すべきファイル_VC 用』フォルダのパスを入力する.



[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.

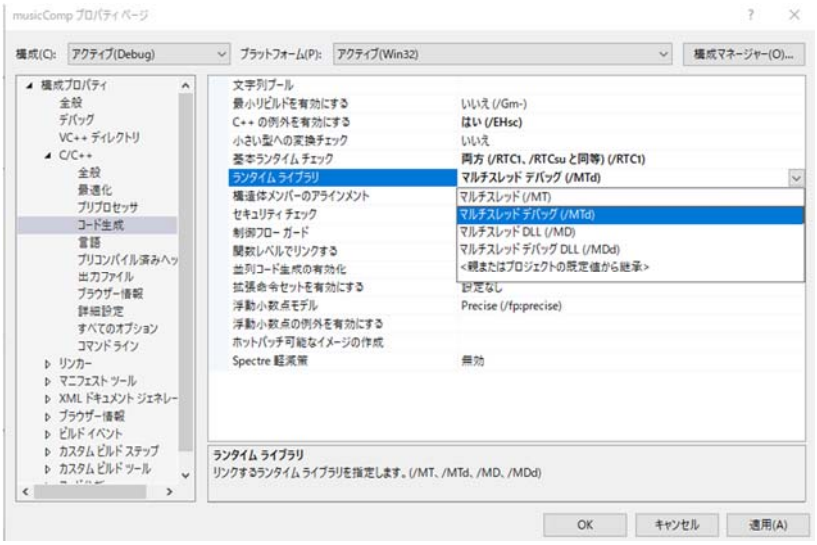
②左側のリストから『構成プロパティ』→『Ｃ／Ｃ＋＋』→『コード生成』を選ぶ.

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.



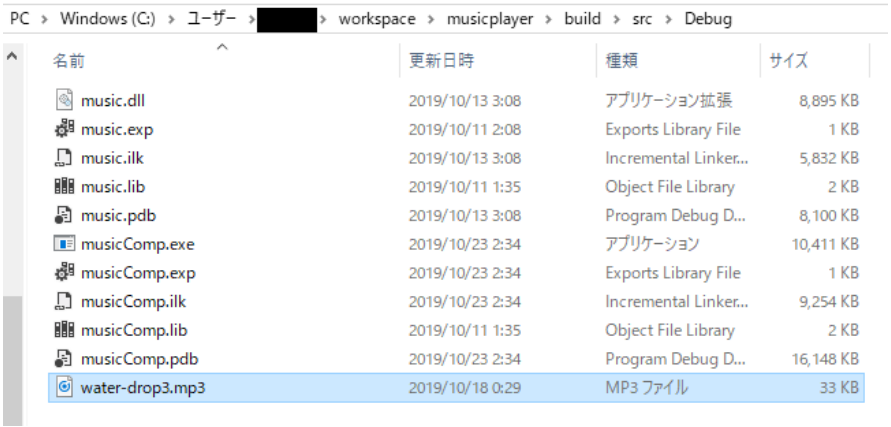
[9] ダイアログの下の方にある『ＯＫ』を押してダイアログを閉じる

[10]『music』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.

[13] 「https://soundeffect-lab.info/sound/various/」から『水滴 3』をダウンロードし，そのまま『build フォルダ』→『src フォルダ』→『Debag フォルダ』内にいれる.



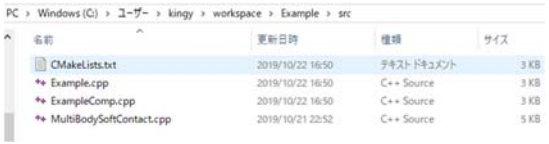
4.3 Create Tapioca の準備

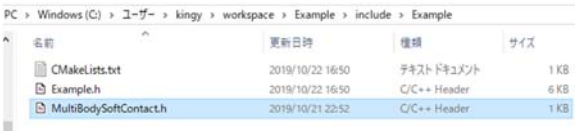
[1] 物理エンジン Bullet Physics を PC にて以下の URL にアクセスし、Bullet Physics(Source code)の zip ファイル(約 1 GB)をダウンロードする。ダウンロード後、zip ファイルをすべて展開する。

https://github.com/bulletphysics/bullet3/releases

[2] RTP 内でコンポーネントを作成後、build ファイルを作成する。

[3] Bullet Physics の example ファイル内にある MultiBodySoftContact から MultiBodySoftContact.cpp を Create Tapioca の src ファイルに、MultiBodySoftContact.h を include の中のファイル内にコピーする。





[4] src の CmakeList.txt に以下のオレンジ部分を書き換える。

```
set(comp_srcs Tapioca.cpp MultiBodySoftContact.cpp)
set(standalone_srcs TapiocaComp.cpp MultiBodySoftContact.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 "\\" "/"
    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 "\\" "/"
    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 "\\" "/"
    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:/Users/enshu/Downloads/bullet3-2.88/bullet3-2.88/src")

add_definitions(${OPENRTM_CFLAGS} -DBT_THREADSAFE=1 -
DBT_USE_DOUBLE_PRECISION)

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

```
link_directories(${OPENRTM_LIBRARY_DIRS} "C:/Users/enshu/Downloads/bullet3-2.88/bullet3-2.88/bin")
add_library(${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} glu32.lib; opengl32.lib;
optimized Bullet3Common_vs2010_x64_release.lib;
optimized BulletCollision_vs2010_x64_release.lib;
optimized BulletDynamics_vs2010_x64_release.lib;
optimized LinearMath_vs2010_x64_release.lib;
optimized OpenGL_Window_vs2010_x64_release.lib;
optimized BulletExampleBrowserLib_vs2010_x64_release.lib;
debug Bullet3Common_vs2010_x64_debug.lib;
debug BulletCollision_vs2010_x64_debug.lib;
debug BulletDynamics_vs2010_x64_debug.lib;
debug LinearMath_vs2010_x64_debug.lib;
debug OpenGL_Window_vs2010_x64_debug.lib;
debug BulletExampleBrowserLib_vs2010_x64_debug.lib; ${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 glu32.lib; opengl32.lib;
optimized Bullet3Common_vs2010_x64_release.lib;
optimized BulletCollision_vs2010_x64_release.lib;
optimized BulletDynamics_vs2010_x64_release.lib;
optimized LinearMath_vs2010_x64_release.lib;
optimized OpenGL_Window_vs2010_x64_release.lib;
optimized BulletExampleBrowserLib_vs2010_x64_release.lib;
debug Bullet3Common_vs2010_x64_debug.lib;
debug BulletCollision_vs2010_x64_debug.lib;
debug BulletDynamics_vs2010_x64_debug.lib;
debug LinearMath_vs2010_x64_debug.lib;
debug OpenGL_Window_vs2010_x64_debug.lib;
debug BulletExampleBrowserLib_vs2010_x64_debug.lib; ${OPENRTM_LIBRARIES})
install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}Comp
  EXPORT ${PROJECT_NAME}
  RUNTIME DESTINATION ${INSTALL_PREFIX} COMPONENT
  LIBRARY DESTINATION ${INSTALL_PREFIX} COMPONENT
  ARCHIVE DESTINATION ${INSTALL_PREFIX} COMPONENT component)
install(FILES ${PROJECT_SOURCE_DIR}/RTC.xml DESTINATION ${INSTALL_PREFIX}
    COMPONENT component)
```

[5] include の CmakeList.txt に以下のオレンジ部分を書き換える。書き換えたら Cmake する。

```
set(hdrs Tapioca.h MultiBodySoftContact.h PARENT_SCOPE)
```

[6] build フォルダ内の『CreateTapioca.sln』を開き，ビルドを行う．

4.4 コンポーネント接続

[1] PC に配線が終わった Arduino を接続する

[2] ネームサーバーを実行する.

[3] eclipse を起動する.

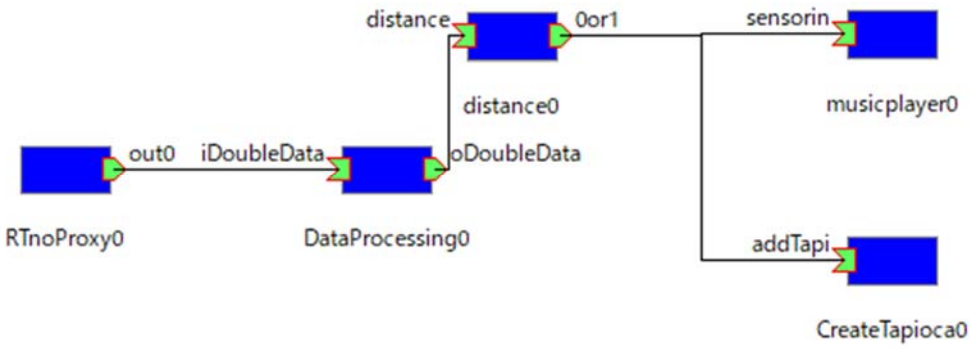
[4] eclipse が立ち上がったら，『ネームサーバーの追加』から localhost を追加する.

[5] 各コンポーネントのファイルの build フォルダ→src フォルダ→Debug フォルダから RTC を起動する. 『Create Tapioca』だけは build フォルダ→src フォルダ→Release フォルダから RTC を起動する.

[6] eclipse の Open New System Editor をクリックし，コンポーネントをシステムダイアグラムに表示する.

[7] 「DataProcessing」の Configuration 変数を変更する. InPortType，OutPortType を TimedDouble にする. OutputFunction に『27.68*x^(-1.204)』を入力する. 変更が終わったら当コンポーネントを Activate する.

[8] 以下の図のようにポートを接続する.



5 システム利用

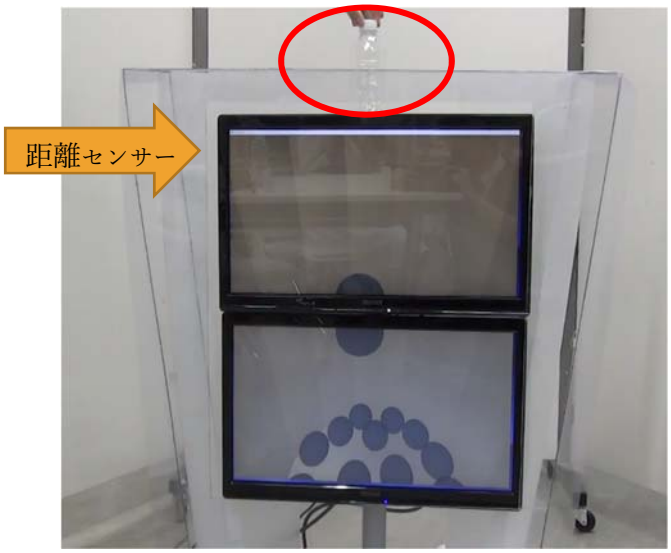
ゴミ箱の実際の使用方法について示す

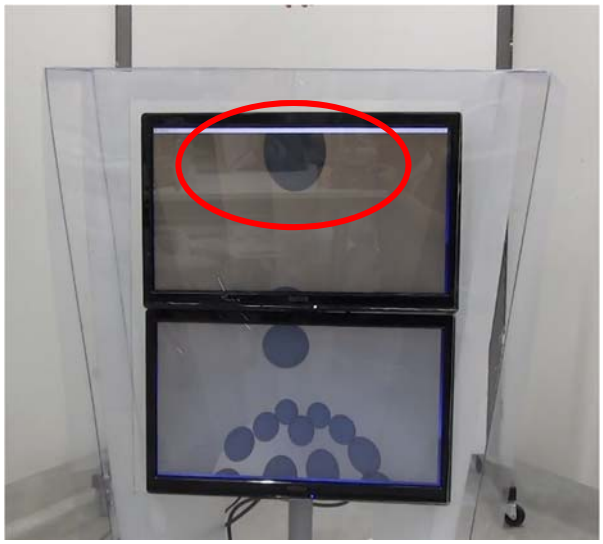
[1] ゴミ箱に距離センサーを取り付ける．このとき，ゴミによってセンサーの読み取る値が元々読み取る値と異なるように注意して設置する．

[2] パソコンとディスプレイを接続する．

[3] コンポーネントを全て Activate する．

実際にこのようにごみを入れると，





新たなタピオカが生成される．

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)　https://www.openrtm.org/rt/RTMcontest/2011/abstract2011-RC.pdf

[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?page=FrontPage

[3] “DX ライブラリ置き場 HOME” https://dxlib.xsrv.jp

[4] “Bullet Real-Time Physics Simulation” https://pybullet.org/wordpress/

[5] “Visual Studio Community 2017 を使用した場合の DX ライブラリの使い方” https://dxlib.xsrv.jp/use/dxuse_vscom2017.html