自由課題

自由課題で使用する機材：RaspberryPi と Web カメラ

概要

Raspberry Pi に接続された WEB カメラから画像データを取得し、OutPort で出力するコンポーネントを作成します。自由課題では、ここまで行ってきた課題を理解していることの確認と他の機器で使用するコンポーネントの作成の仕方を学びます。これができれば、自分でコンポーネントを作成できるようになります。みなさん、チャレンジしてみてください。

内容

この講習会テキストは下記ページを参考にしています。
・チュートリアル（画像処理コンポーネントの作成 Windows 編）
http://www.openrtm.org/openrtm/ja/node/5022 （2016/7/28 アクセス）

※文中の「x.y」や「x.y.z」の表記は使用環境の OpenRTM-aist のバージョンに読み替えてください。

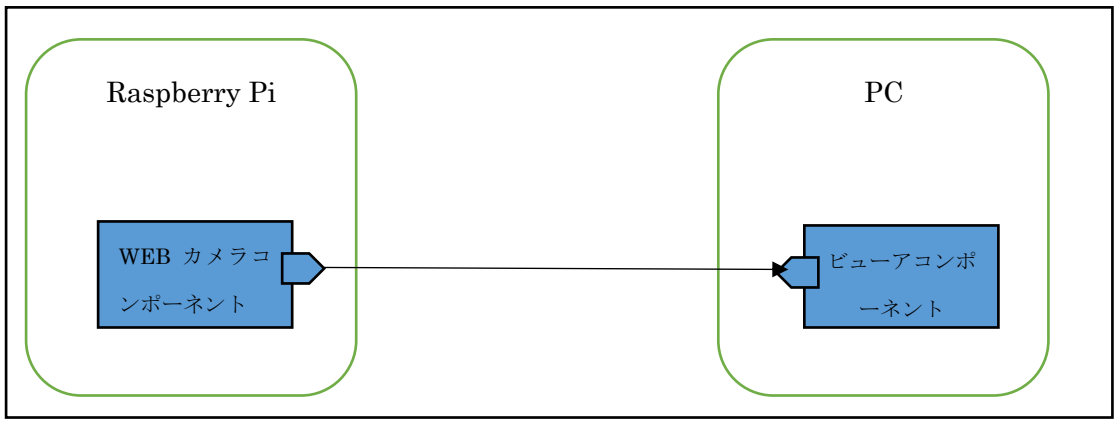
1. コンポーネントの仕様

① 作成するコンポーネントについて

作成するコンポーネントは Raspberry　Pi に接続された WEB カメラから画像を取得し OutPort で画像データを出力するコンポーネントです。

コンポーネントの動作は、OpenCV を使用し WEB カメラから画像を取得し画像データを OutPort(CameraImage 型)から出力します。
さらに外部パラメータを使用し、取得する画像の縦幅、横幅を変更することが出来ます。
従って、このコンポーネントは OutPort(CameraImage 型)を一つ持ち、コンフィギュレーションで画像サイズを変更するパラメータを持ちます。

このコンポーネントは Raspbery Pi 上で起動しデータを取得します。取得したデータは PC で起動されたビューアコンポーネントに送られ PC 側で表示されます。従ってシステムのイメージは下記の様になります。



② 仕様まとめ

以上のことをまとめるとコンポーネントは以下の様になります。

コンポーネント名称	RaspWEBCamera	
OutPort		
ポート名	WebCameraImageOut	
型	CameraImage	
Configuration		
パラメータ名	height	width
型	int	int

この仕様を元にコンポーネントを作成していきます。

2. コンポーネントの雛型の生成

コンポーネントの雛型の生成のために RTCBuilder を起動してください。

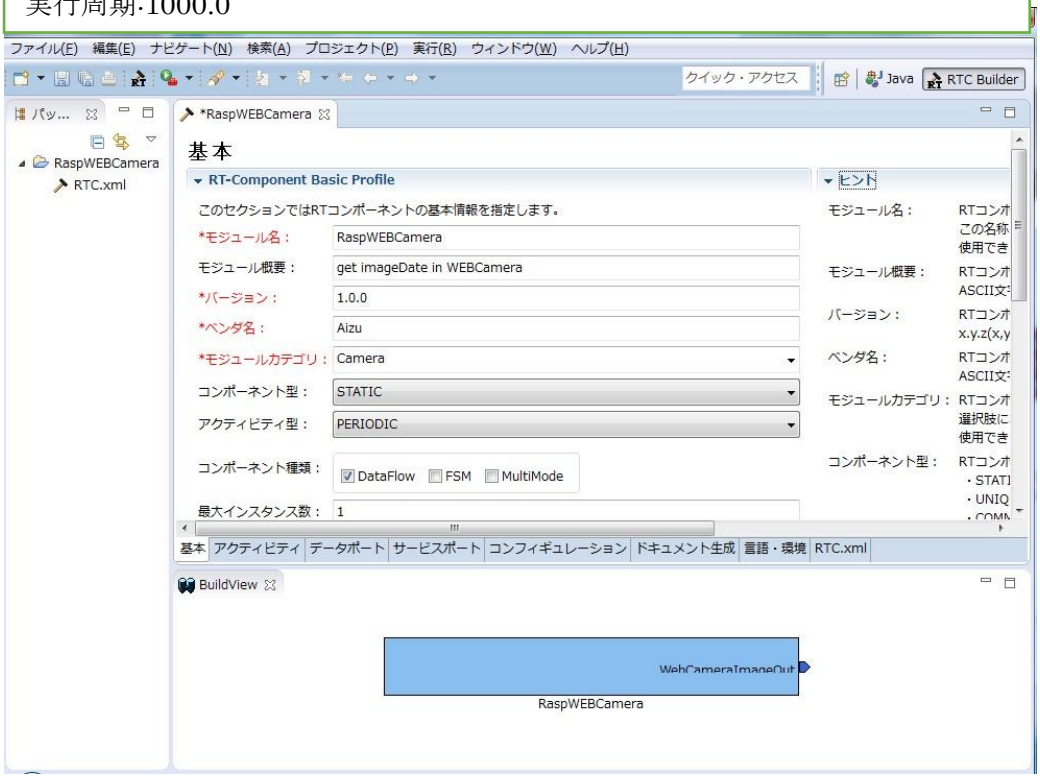
① 新規プロジェクト

新規プロジェクトを作成します。プロジェクト名は[RaspWEBCamera]とします。
手順に関しては、第二部の Flip コンポーネント作成を参考にしてください。

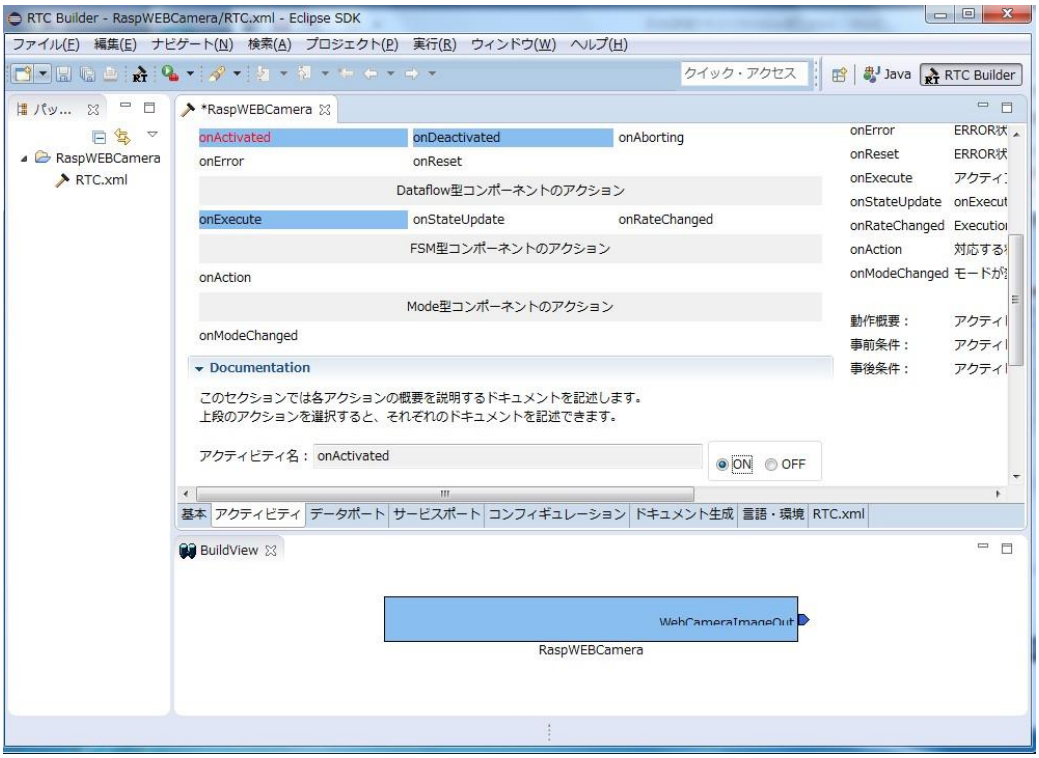
② プロファイル情報入力とコードの生成

一番左の「基本」タブを選択し、基本情報を設定します。コンポーネントの名前や概要などを記入します。ラベルが赤文字の項目は必須項目です。その他はデフォルトのままで大丈夫です。

モジュール名: RaspWEBCamera
モジュール概要: get imageDate in WEBCamera
バージョン:1.0.0
ベンダ名:Aizu
モジュールカテゴリ: Camera
コンポーネント型:STATIC
アクティビティ型:PERIODIC
コンポーネント種類:DataFlowComponent
最大インスタンス数:1
実行型:PeriodicExecutionContext
実行周期:1000.0

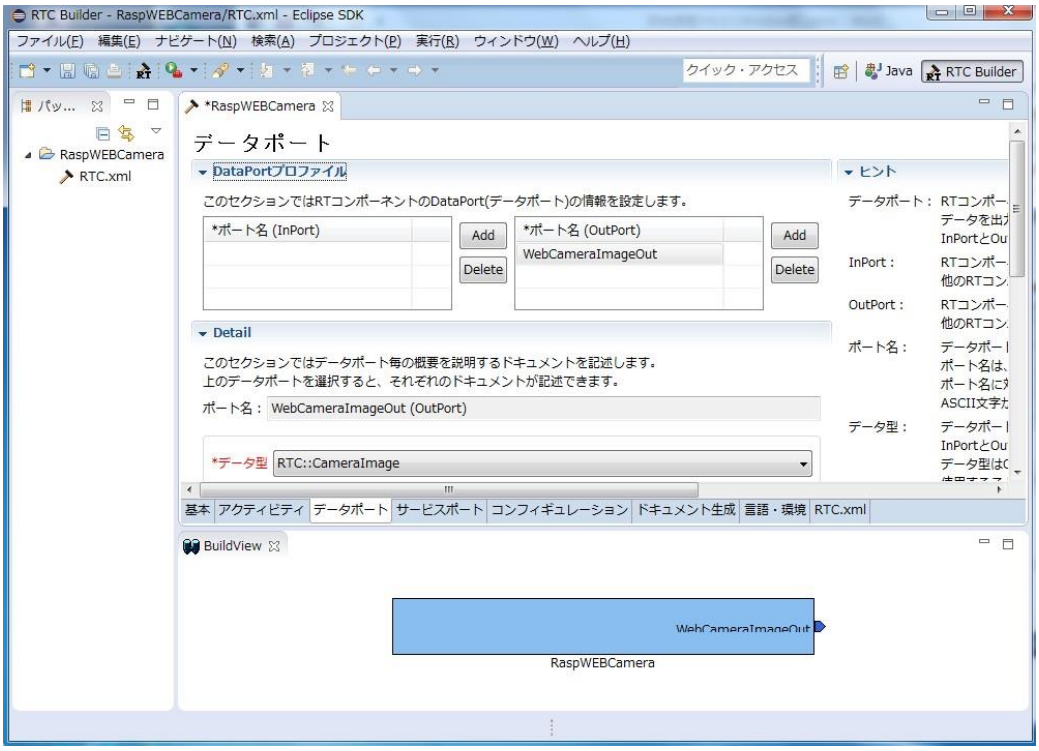


次に、「アクティビティ」タブを選択し、使用するアクションコールバックを指定します。WEB カメラコンポーネントでは、onActivated(),onDeactivated(),onExecute()コールバックを使用します。下図のように赤枠の onAtivated をクリック後に赤枠のラジオボタンにて"on"にチェックを入れます。onDeactivated,onExecute についても同様の手順を行います。



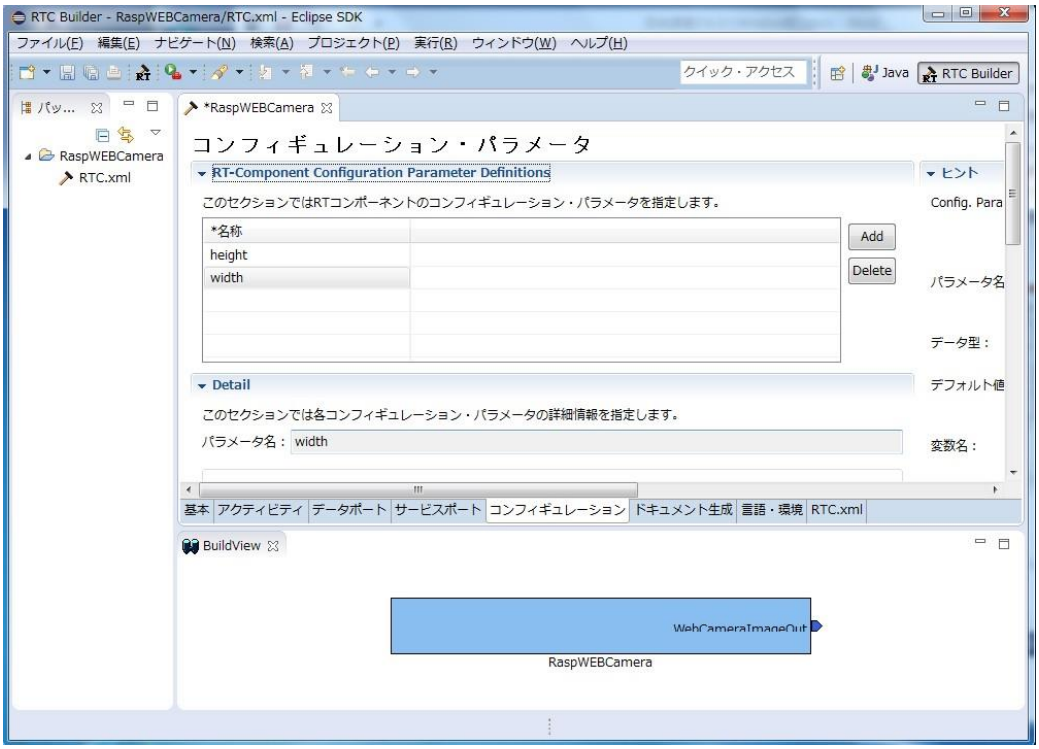
「データポート」タブを選択し、データポートの情報を入力します。 先ほど決めた仕様を元に以下のように入力します。

・OutPort
ポート名: WebCameraImageOut
データ型: RTC::CameraImage
変数名: WebCameraImage
表示位置: right

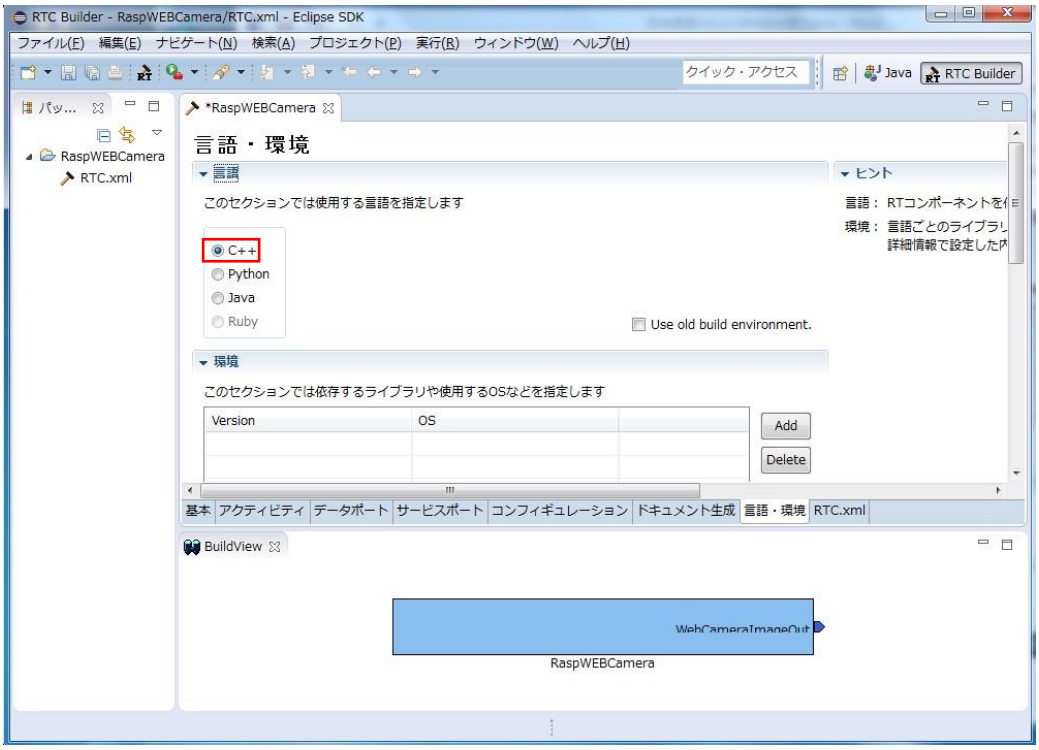


「コンフィギュレーション」タブを選択し、先ほど決めた仕様を元に、Configuration の情報を入力します。制約条件および Widget とは、RTSystemEditor でコンポーネントのコンフィギュレーションパラメータを表示する際に GUI で値の変更を行うための形式を表すものです。
ここでは、カメラのサイズをコンフィギュレーションで操作出来る様にするので、[height]と[width]を設定します。

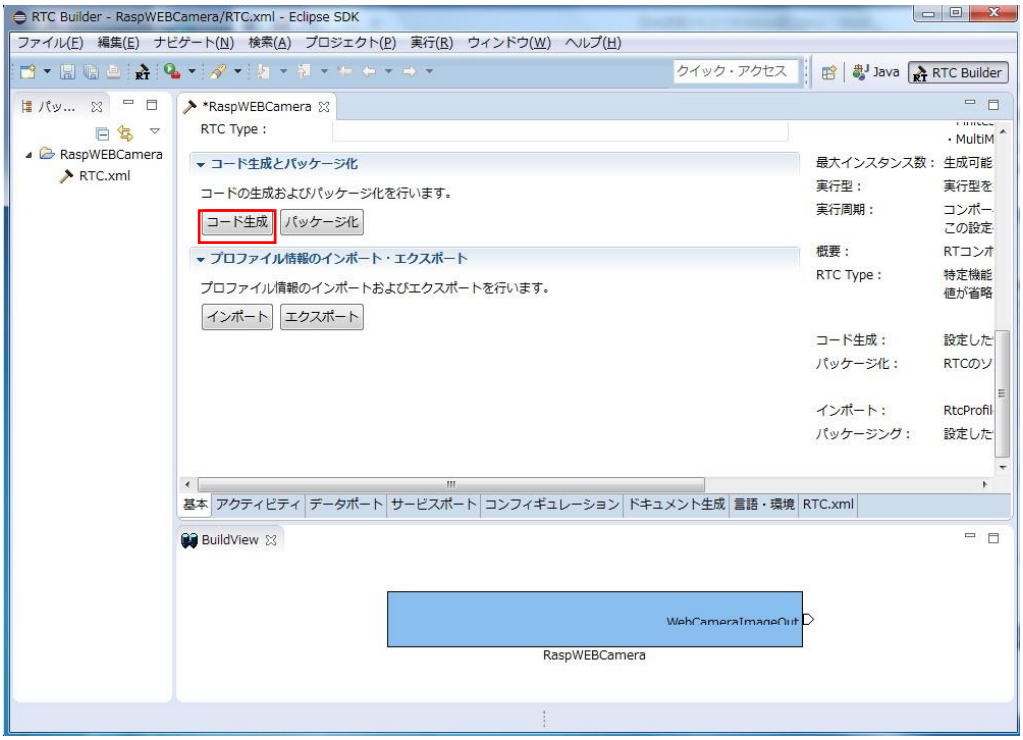
名称: height データ型: int デフォルト値: 240 変数名: height 制約条件: なし Widget: text	名称: width データ型: int デフォルト値: 320 変数名: width 制約条件: なし Widget: text



「言語・環境」タブを選択し、プログラミング言語を選択します。 ここでは、C++(言語)を選択します。言語・環境はデフォルトでは設定されていないので、指定し忘れるとコード生成時にエラーになりますので、 必ず言語の指定を行うようにしてください。

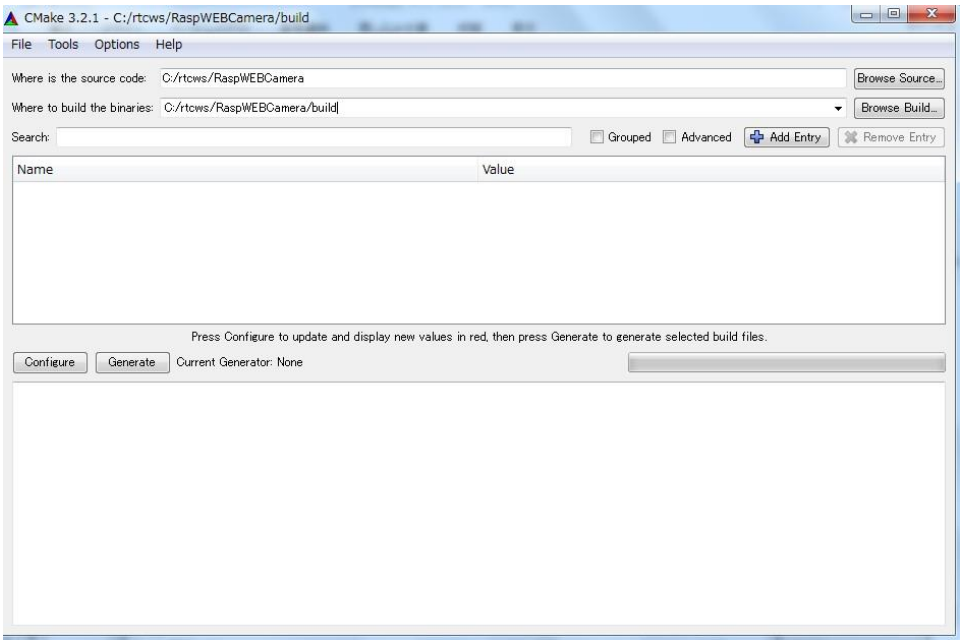


全ての設定が完了しましたら、「基本」タブに戻りコード生成ボタンをクリックします。問題がなければコンポーネントの雛型が生成されます。



③ 仮ビルド

ここまでの作業で WEB カメラコンポーネントの雛型が生成されました。
次の作業として CMake を利用してビルド環境の Configure を行います。
スタートメニューなどから CMake (cmake-gui)を起動します。



画面上部に以下のようなテキストボックスがあります。

・Where is the source code

・Where to build the binaries

「Where is the source code」に CMakeList.txt が有る場所、「Where to build the binaries」にビルドディレクトリを指定します。

CMakeList.txt はデフォルトでは<ワークス ペースディレクトリ>/ RaspWEBCamera になります。

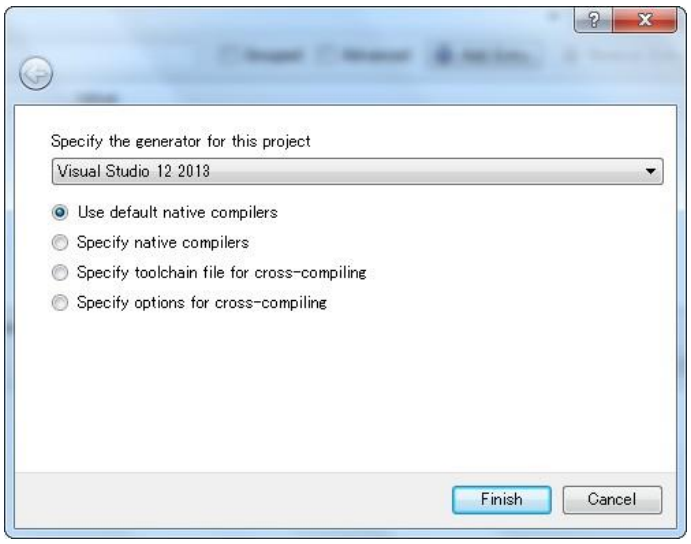
ビルドディレクトリとは、ビルドするためのプロジェクトファイル やオブジェクトファイル、バイナリを格納する場所のことです。場所は任意ですが、この場合 <ワークスペースディレクトリ>/RaspWEBCamera/build のように分かりやすい名前をつけた RaspWEBCamera のサブディレクトリを指定することをお勧めします。

ディレクトリは自動で作成されるので指定前に作成する必要はありません。

今回は以下の様になるはずです。

Where is the source code	C:¥rtcws¥RaspWEBCamera
Where to build the binaries	C:¥rtcws¥RaspWEBCamera¥build

指定したら、下の Configure ボタンを押します。すると下図のようなダイアログが表示されますので、生成したいプロジェクトの種類を指定します。

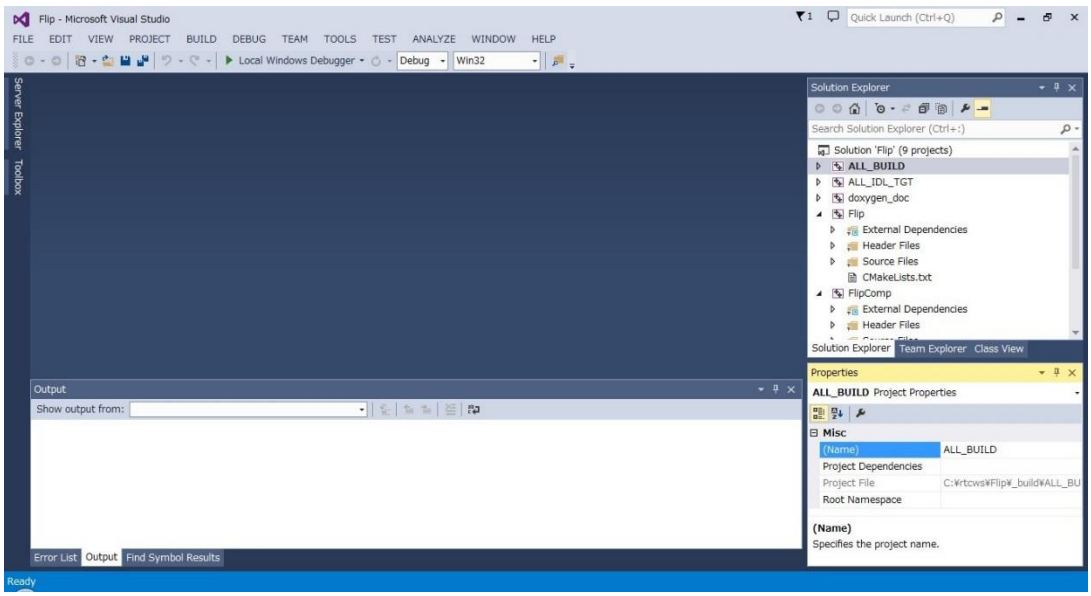


Visual Studio バージョン	32/64 bit	生成したいプロジェクトの種類
Visual Studio 2013	32 bit	Visual Studio　12 2013
	64 bit	Visual Studio　12 2013 Win 64
Visual Studio 2015	32 bit	Visual Studio　14 2015
	64 bit	Visual Studio　14 2015 Win 64

ダイアログで Finish を押すと Configure が始まります。問題がなければ下部のログウインドウに Configuring done と出力されますので、続けて Generate ボタンを押します。Generating done と出ればプロジェクトファイル・ソリューションファイル等の出力が完了します。

次に先ほど指定した build ディレクトリの中の RaspWEBCamera.sln をダブルクリックして Visual Studio 2013 を起動します。

起動後、ソリューションエクスプローラーの ALL_BUILD を右クリックしビルドを選択してビルドします。特に問題がなければ正常にビルドが終了します。



3. ヘッダ、ソースの編集

① ヘッダファイル（RaspWEBCamera.h）の編集

OpenCV のライブラリを使用するため、OpenCV のインクルードファイルをインクルードします。下記内容をインクルードしている所に追加してください。

※ソースコードは一行で表示するためにフォントを小さくしています。

```
//OpenCV 用インクルードファイルのインクルード
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc_c.h>
#include <opencv2/imgproc/imgproc.hpp>
```

画像の保存用にメンバー変数を追加します。下記内容を class の private:内(// <rtc-template block="private_attribute">の下)に追加してください。

```
int width;
int height;
cv::VideoCapture cap;
cv::Mat mat_image;
```

② ソースファイル（RaspWEBCamera.cpp）の編集

下記のように、onActivated(),onDeactivated(),onExecute()を実装します。

onActivated()

```
RTC::ReturnCode_t RaspWEBCamera::onActivated(RTC::UniqueId ec_id)
{
	std::cout << "Active" << std::endl;
	// カメラ接続
	cap.open(0);
	if (cap.isOpened())std::cout << "USB Camera Connect" << std::endl;
	else std::cout << "USB Camera UnConnect" << std::endl;
	// 画面サイズ設定
	cap.set(CV_CAP_PROP_FRAME_WIDTH, m_width);
	cap.set(CV_CAP_PROP_FRAME_HEIGHT, m_height);

	return RTC::RTC_OK;
}
```

onDeactivated()

```
RTC::ReturnCode_t RaspWEBCamera::onDeactivated(RTC::UniqueId ec_id)
{
	std::cout << "Deactive" << std::endl;
	// カメラ用メモリの開放
	cap.release();
	return RTC::RTC_OK;
}
```

onExecute()

```
RTC::ReturnCode_t RaspWEBCamera::onExecute(RTC::UniqueId ec_id)
{
	if (cap.isOpened()){
		if (width != m_width || height != m_height)
		{
			// 画面サイズ再設定
			cap.set(CV_CAP_PROP_FRAME_WIDTH, m_width);
			cap.set(CV_CAP_PROP_FRAME_HEIGHT, m_height);

			std::cout << "Width:" << m_width << " Height:" << m_height << std::endl;
		}
		//画像取得
		cap.read(mat_image);

		IplImage frame = mat_image;

		int len = frame.nChannels * frame.width * frame.height;
		// 画面サイズ情報を入れる
		m_WebCameraImage.pixels.length(len);
		m_WebCameraImage.width = frame.width;
		m_WebCameraImage.height = frame.height;
		//画像データを OutPort にコピー
		memcpy((void *)&(m_WebCameraImage.pixels[0]), frame.imageData, len);
		//画像データ出力
		m_WebCameraImageOut.write();

		//コンフィギレーションの値を保存
		height = m_height;
		width = m_width;
	}
	return RTC::RTC_OK;
}
```

4. CMake によるビルドに必要なファイルの生成

C:¥rtcws¥RaspWEBCamera¥src¥CMakeLists.txt を編集します。

このコンポーネントでは OpenCV を利用していますので、OpenCV のヘッダのインクルードパス、ライブラリやライブラリサーチパスを与えてやる必要が有ります。以下の 2 点を追加・変更するだけで OpenCV のライブラリがリンクされ使えるようになります。

・find_package(OpenCV REQUIRED)を追加

・最初の target_link_libraries に ${OpenCV_LIBS} を追加

・target_link_libraries は 2 ヶ所あります。

・追加するときは${OpenCV_LIBS}の前に半角スペースを入れてください。

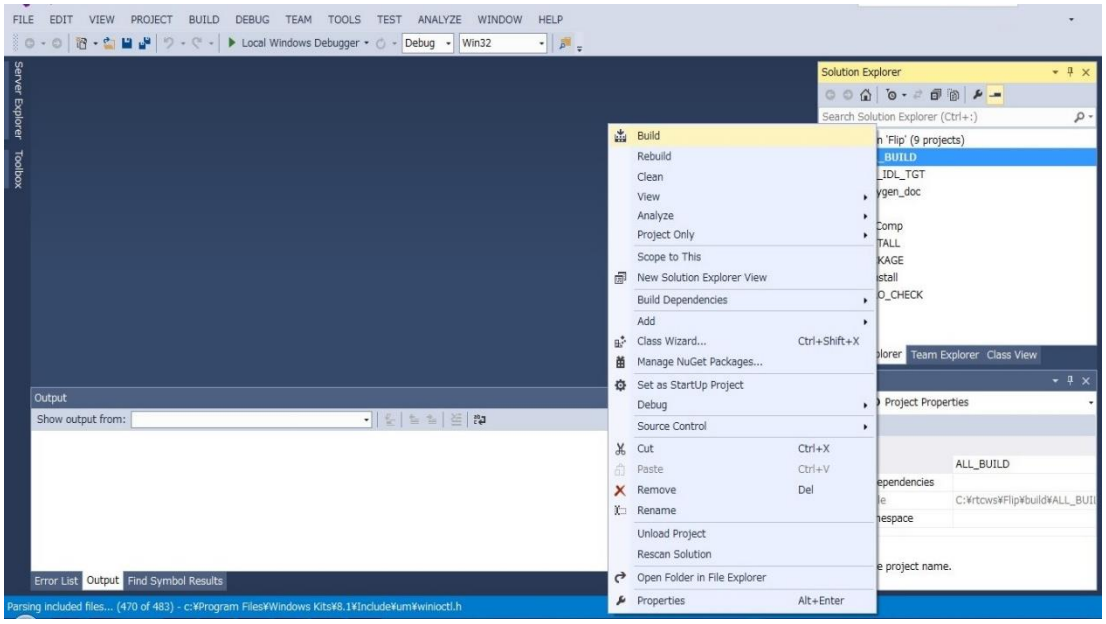
```
set(comp_srcs RaspWEBCamera.cpp )
set(standalone_srcs RaspWEBCameraComp.cpp)

find_package(OpenCV REQUIRED) ←この行を追加
  : 中略
add_dependencies(${PROJECT_NAME} ALL_IDL_TGT)
target_link_libraries(${PROJECT_NAME} ${OPENRTM_LIBRARIES} ${OpenCV_LIBS}) ←OepnCV_LIBS を追加
  : 中略
add_executable(${PROJECT_NAME}Comp ${standalone_srcs}
  ${comp_srcs} ${comp_headers} ${ALL_IDL_SRCS})
target_link_libraries(${PROJECT_NAME}Comp ${OPENRTM_LIBRARIES} ${OpenCV_LIBS}) ←OepnCV_LIBS
を追加
```

5. Visual Studio でビルド

CMakeList.txt を編集したので、再度 CMake GUI で Configure および Generate を行います。CMake の Generate が正常に終了した事を確認し、RaspWEBCamera.sln ファイルをダブルクリックし、Visual C++ 2013 を起動します。

Visual C++ 2013 の起動後、下図のように右クリックでコンポーネントのビルドを行います。



6. コンポーネントの動作確認

コンポーネントが仕様通りに動くか確認をします。そのために PC 内で WEB カメラコンポーネントとビューアコンポーネントを起動し実際に WEB カメラから画像を取得します。

① NameService の起動

コンポーネントの参照を登録するためのネームサービスを起動します。

[スタート]メニューから[すべてのプログラム]→[OpenRTM-aist x.y.z]→ [tools]→[Start Naming Service]をクリックして下さい。

※Windows8 の場合下記パスを参考になります。

C:¥ProgramData¥Microsoft¥Windows¥Start Menu¥Programs¥OpenRTM-aist x.y.z ¥Tools

※[Start Naming Service]をクリックしても omniNames が起動されない場合は、フルコンピュータ名が 14 文字以内に設定されているかを確認してください。

② WEB カメラコンポーネントの起動

C:¥rtcws¥RaspWEBCamera¥build¥src¥Debug の RaspWEBCameraComp.exe をダブルクリックで起動させます。

③ ビューアコンポーネントの起動

InPort で受け取った画像を画面に表示する CameraViewerComp を起動します。

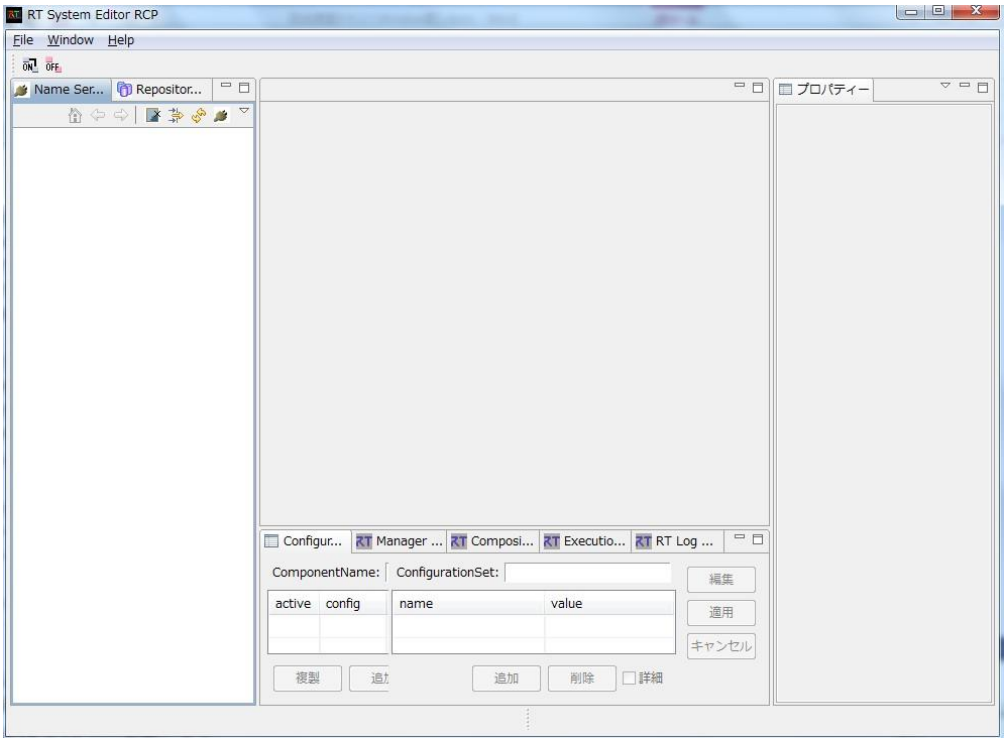
[スタート]メニューから[すべてのプログラム]→[OpenRTM-aist x.y.z]→ [C++]→[Components]→[OpenCV-Examples]
内にあるのでダブルクリックで起動してください。

④ WEB カメラの接続

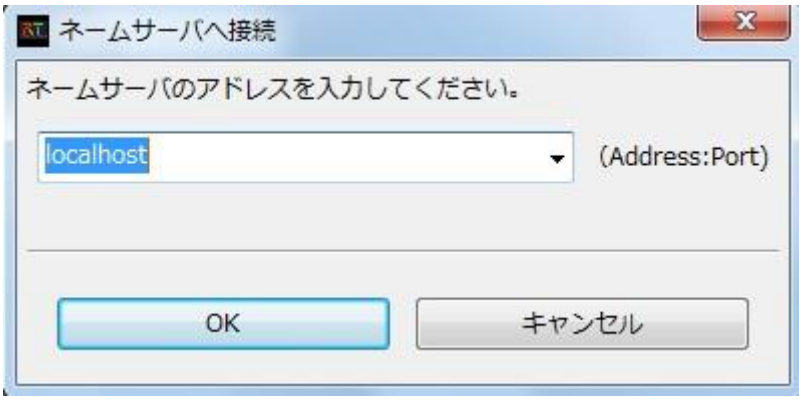
WEB カメラを PC に接続してください。

⑤ コンポーネントの接続

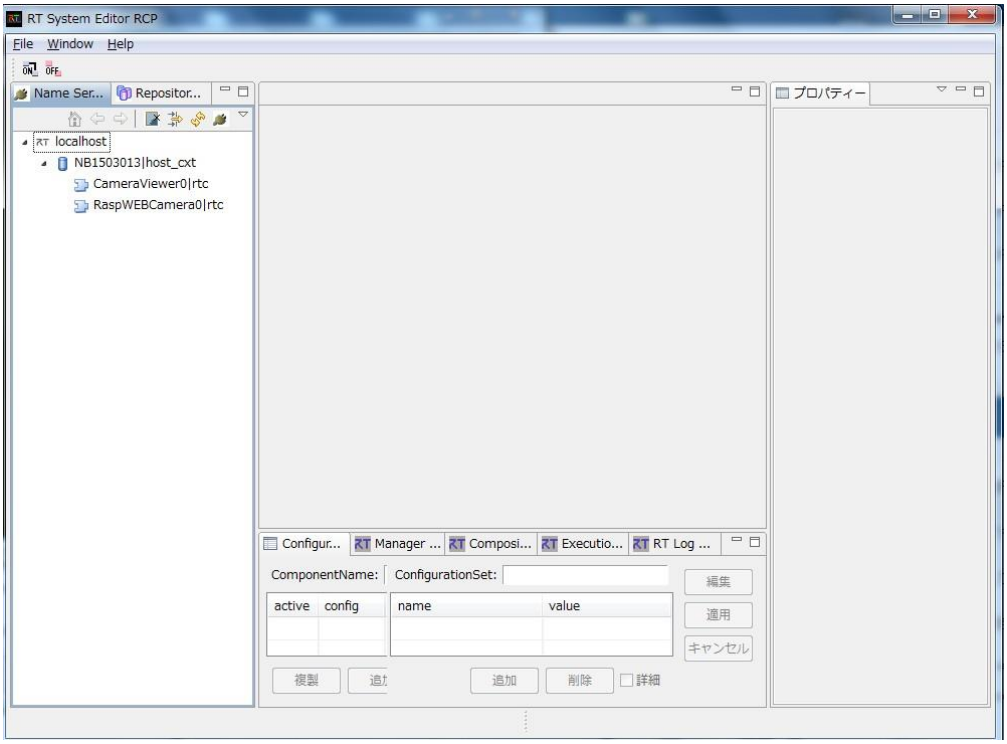
RTSystemEditor の左側の Name Service View のコンセントアイコンをクリックし、ネームサーバに接続します。



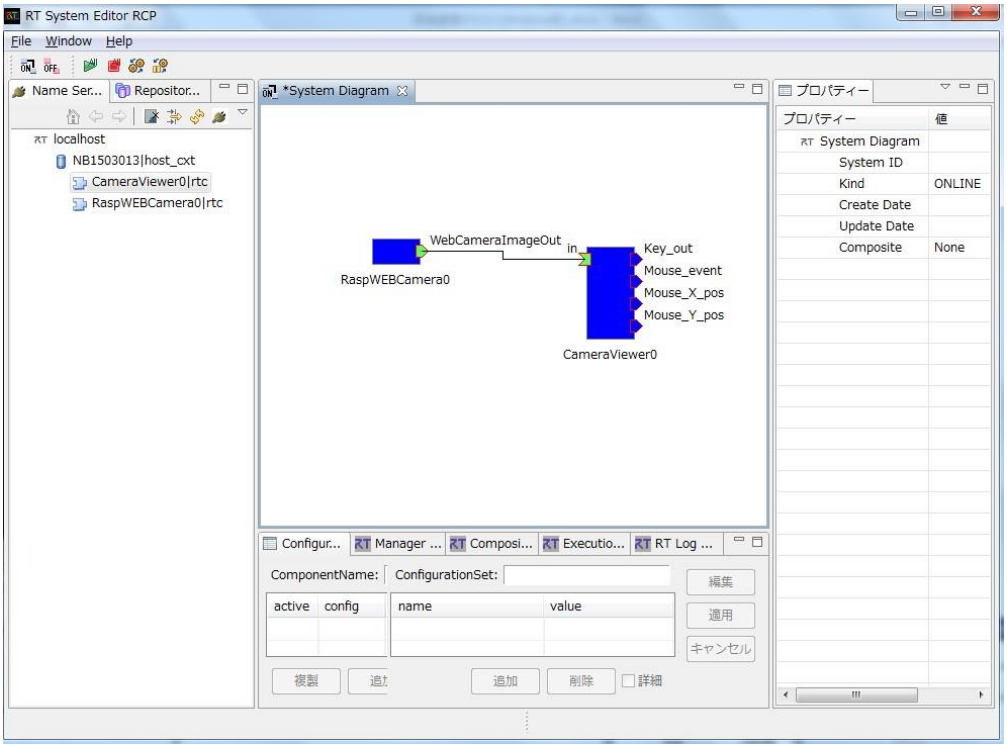
自ホストのネームサーバに接続します。接続ダイアログに localhost と入力します。



下記の様に表示されるのでその後、メニューバーの online エディタアイコン(ON と書かれたアイコン)をクリックし、SystemEditor を開きます。

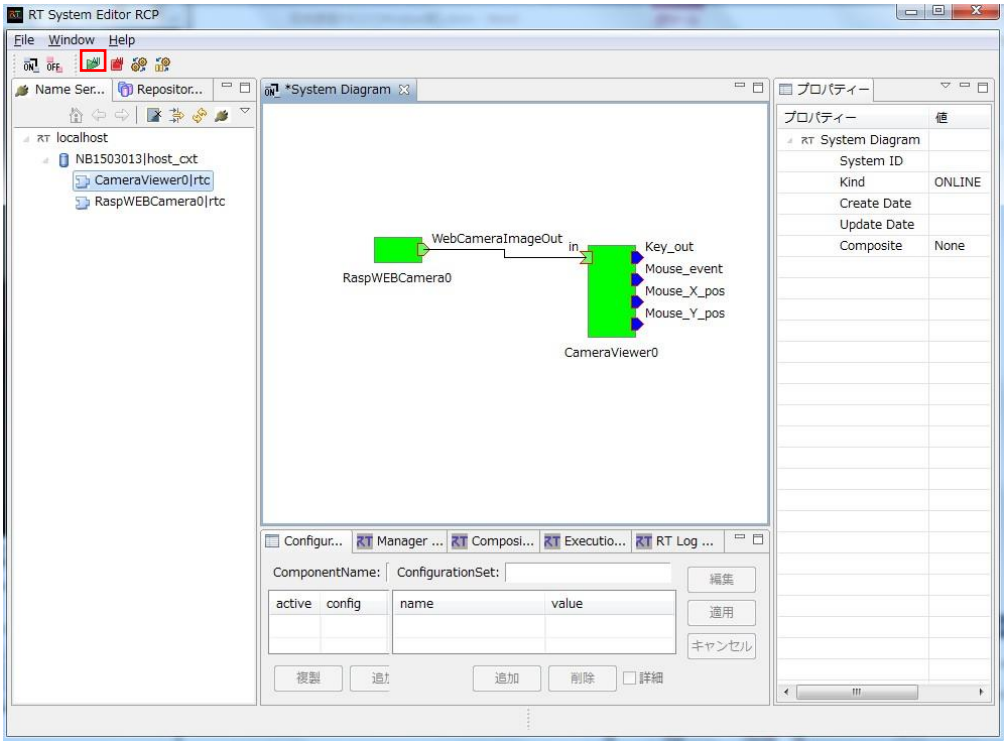


左側の Name Service View から各コンポーネントをドラックアンドドロップで SystemEditor 上にコンポーネントを配置します。そしてデータポートを下記図の様に接続します。



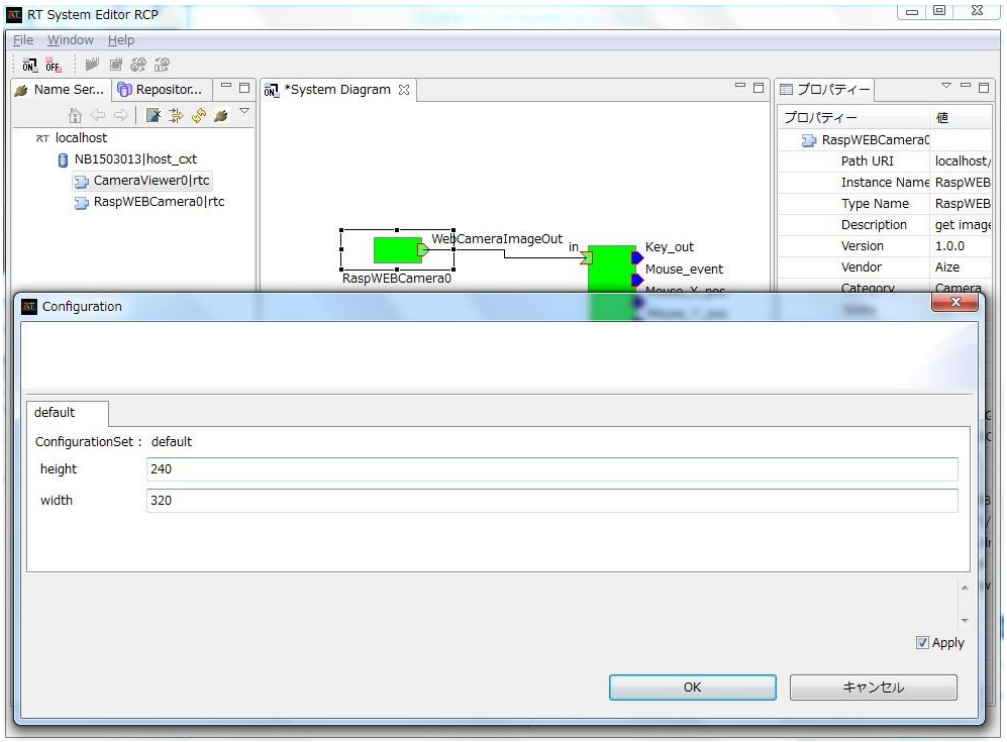
⑥ コンポーネントの Activate

RTSystemEditor の上部にある緑の「ALL」というアイコンをクリックし、 全てのコンポーネントをアクティブにします。正常にアクティブになると、下図のように黄緑色でコンポーネントが表示されます。



⑦ 動作確認

System Diagram 上の RaspWEBCamera0 をクリックするとコンフィギュレーションビューが表示されます。編集のボタンを押すと height と width が表示されますので変更して、画像のサイズが変更されるのを確認してください。



確認を終えたら、RTSystemEditor の上部にあります赤色の「ALL」というアイコンをクリックし、コンポーネントをディアクティブ状態にします。その後、Name Service View のコンポーネント一覧から RaspWEBCamera0 を右クリックで選択し、Exit でコンポーネントを削除します。その後 WEB カメラを PC から取り外し、使用する Raspberry Pi に接続してください。

7. コンポーネントを Raspberry Pi へコピー

作成したコンポーネントを Raspberry Pi にコピーする方法を記載します。前段階として Raspberry Pi に Tera Term でアクセスしてください。

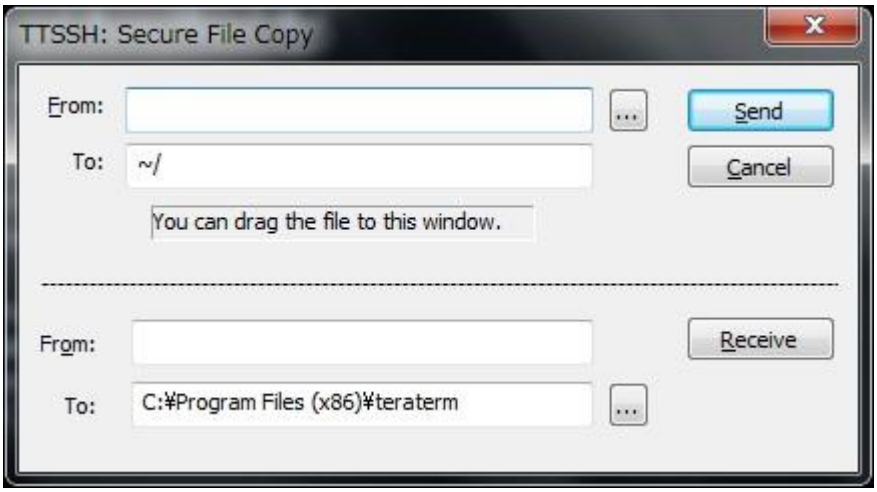
① build 以下を削除

Raspberry Pi にコンポーネントをコピーするときは CMake で作成した build 以下を削除します。これは、容量を減らすためと Windows 上で作成された build 以下は Raspberry Pi 上では使用できないためです。

② フォルダを zip で圧縮し Raspberry Pi へコピーする。

RaspWEBCamera のフォルダを zip で圧縮してください。今回コピーするには Tera Term の「SSH SCP …」を使用します。この機能で送れるファイルは 1 回につき 1 ファイルなのでフォルダを送るには圧縮して 1 ファイルにする必要があります。

次に Tera Term の「ファイルメニュー」→「SSH SCP …」を選択します。



上の方の From にさきほど圧縮した WEB カメラコンポーネントを選択し、Send ボタンをクリックします。

8. Raspberry Pi 上でコンポーネントをビルド

Raspberry Pi 上で使用できるように再ビルドを行います。

① コピーされたファイルを解凍する。
以下のコマンドを入力し、圧縮ファイルの解凍を行います。

```
$ unzip RaspWEBCamera.zip
```

unzip：圧縮ファイルを復元する。

② コンポーネントをビルドする。
以下のコマンドで Raspberry Pi 上で使用できる様にビルドします。

```
$ cd RaspWEBCamera
$ mkdir build
$ cd build
$ cmake ../
$ make
```

cd：カレントディレクトリを変更する。
mkdir：ディレクトリを作成する。
cmake：プログラムをコンパイルするための Makefile を生成する。

make：プログラムをコンパイルする。

③ NameServer とコンポーネント起動。
以下のコマンドで NameServer とコンポーネントを起動します。

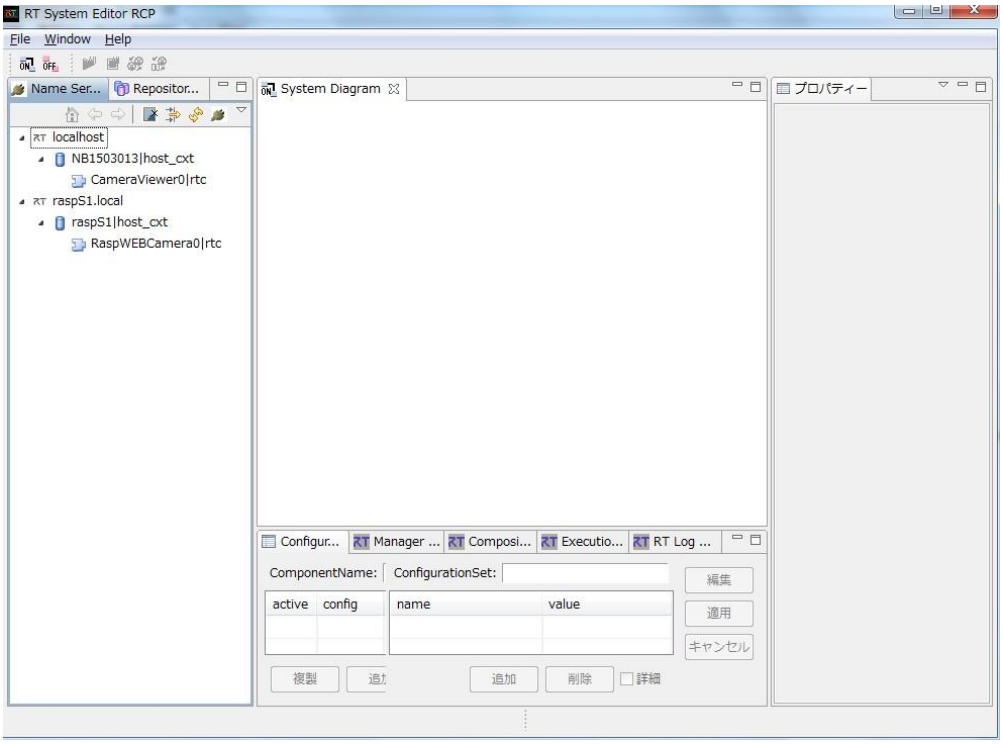
```
$ rtm-naming
$ cd ~/RaspWEBCamera/build/src/
$ ./RaspWEBCameraComp
```

エラーが出ずに起動出来たら完了です。

9. コンポーネントの接続

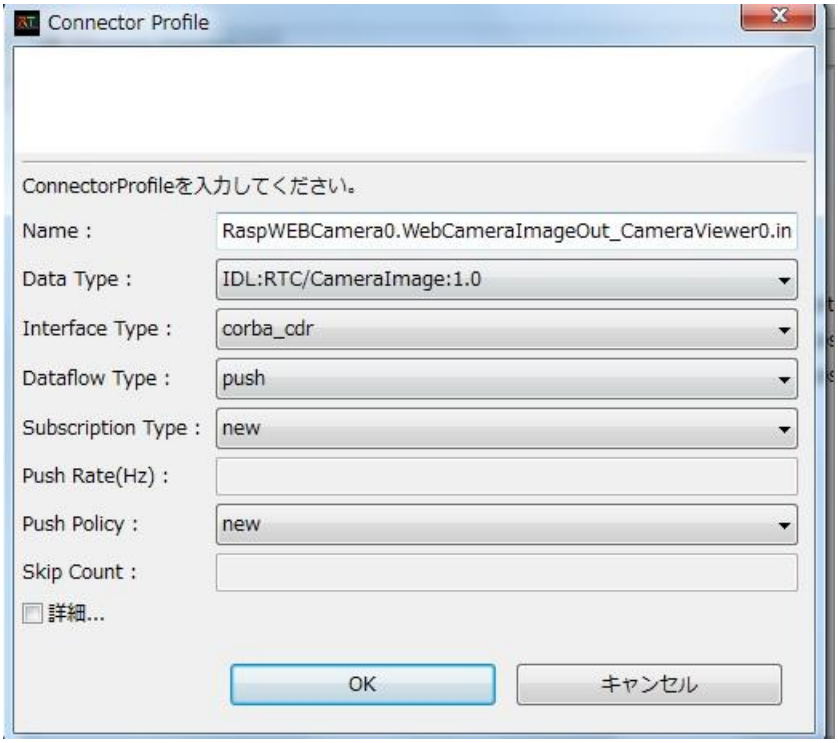
PC 側、Raspberry Pi 側で NameServer とコンポーネントが起動したので RTSystemEditor でコンポーネント同士を接続します。

① PC で RTSystemEditor を使用し各コンポーネントを Active にする。
RTSystemEditor の Name Service View の接続アイコンをクリックし Raspberry Pi のホスト名+.local 、または、Raspberry Pi の IP アドレスをダイアログに入力します。すると Name Service View に RaspWEBCamera0 が表示されます。



Name Service View から各コンポーネントを SystemEditor 上にドラッグアンドドロップし、データポートを接続します。その際に Connector Profile の設定はデフォルトでなく下

記画像の様に、Subscription Type と Push Policy を new に変更して接続してください。



データポートの接続が完了したら、「ALL」という緑のアイコンをクリックし全てのコンポーネントをアクティブにしてください。
Raspberry Pi に接続した WEB カメラから画像を取得出来たら完了です。

10. 他コンポーネントとの接続

WEB カメラコンポーネントとビューアコンポーネントの間にサンプルコンポーネントを挿むことによって取得した画像に色々な変化をもたらすことが出来ます。

ここでは一例を紹介します。

サンプルコンポーネントは下記場所にあります。

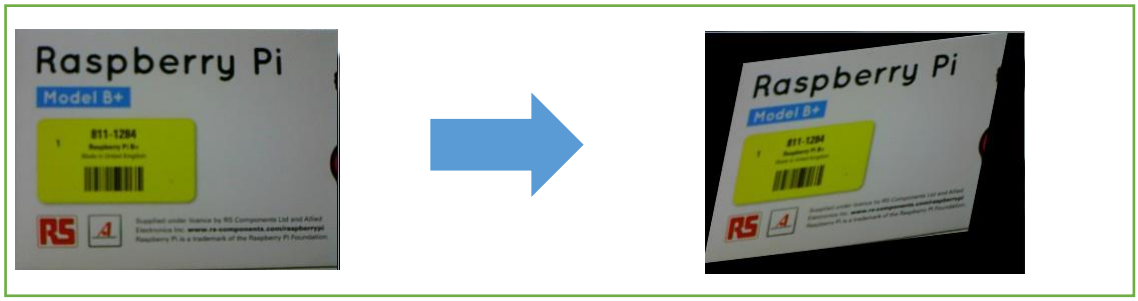
[スタート]メニューから[すべてのプログラム]→[OpenRTM-aist x.y.z]→ [C++]→[Components]→[OpenCV-Examples]

・FlipComp

入力された画像を反転して出力します。反転の種類は左右反転、上下反転、上下左右反転の3種類あります。どれにするかはコンフィギュレーションで決定します。

・AffinComp

入力された画像にアフィー変換をかけ平行四辺形にして出力します。



11. RTC-Library-FUKUSHIMA

① RTC-Library-FUKUSHIMA について

RTC-Library-FUKUSHIMA とはロボット産業振興のために作成された RTC ソフトウェアライブラリーです。

主にコンポーネントの登録やダウンロードしての再利用などが出来ます。

② コンポーネントをアップロード

RTC-Library-FUKUSHIMA へのコンポーネントのアップロードの仕方を説明します。

① ログイン

RTC-Library-FUKUSHIMA へは下記 URL でアクセスします。

RTC-Library-FUKUSHIMA：http://192.168.11.101/

・ログイン ID

ID: guestuser

PW：notrtclibguest

今回の講習会では本番の環境を使わずにローカルの環境を使用します。

サイトにアクセス出来たらサイト上部のログインをクリックしてください。
ログイン画面に移行しユーザー名またはメールアドレスとパスワードを入力する欄がありますので入力してログインをしてください。

② コンポーネントのアップロード手順
コンポーネントをアップロードするにはログイン後、トップページから「ライブラリ」を選択し、「コンポーネント登録/パッケージ登録」のタブを選択します。そして「コンポーネント登録」を選択します。

下記登録画面に遷移したことを確認してください。下記画面で登録を行います。

今回は下記項目を登録します。

RTC.xml ファイル読み込み
WEB カメラコンポーネントで作成された RTC.xml を指定します。指定後、「RTC.xml ファイル読み込み」のボタンを押してください。
RTCBuilder で設定したコンポーネントの情報が登録されます。

コンポーネント登録情報入力

・コンポーネント名：RaspWEBCamera
・概要: get imageDate in WEBCamera
・カテゴリ：カメラ
・タグ：C++、OpenCV、画像処理
・ファイルアップロード：コンポーネントを zip に圧縮してアップロードします。その際、build 以下は削除か退避しておいてください。
・同意する：チェックを入れてください。
・私はロボットではありません：チェックを入れてください。※ローカル環境ではなし

入力が終わりましたら、「確認」のボタンを押し登録情報確認ページに遷移してください。