講習会テキスト第2部 Linux 版

目次

1.	17	よじめに	. 2
	1.	OpenCV とは	.2
	2.	作成する RT コンポーネント	.2
2.	c	vFlip 関数の RT コンポーネント化	. 2
	1.	cvFlip 関数について	.2
	2.	コンポーネントの仕様	.2
	3.	Flip コンポーネントの雛型の生成	.3
	4.	ヘッダ、ソースの編集	13
	5.	CMake によるビルドに必要なファイルの生成	15
	6.	ビルド	16
	7.	コンポーネント起動	16
	8.	コンポーネントの接続	17
3.	F	lip コンポーネントの全ソース	21
	1.	Flip コンポーネントソースファイル (Flip.cpp)	21
	2.	Flip コンポーネントのヘッダファイル (Flip.h)	21
	3.	Flip コンポーネントの全ソースコード	21

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 Linux 編) http://www.openrtm.org/openrtm/ja/node/430 (2016/1/8 アクセス)

1. はじめに

1. OpenCV とは

画像処理・画像解析および機械学習等の機能を持つ C/C++、Java、Python、MATLAB 用ライ ブラリです。

2. 作成する RT コンポーネント

Flip コンポーネント: OpenCV ライブラリが提供する様々な画像処理関数のうち、cvFlip() 関数 を用いて画像の反転を行う RT コンポーネント

2. cvFlip 関数の RT コンポーネント化

OpenCVの cvFlip 関数を使用して、入力された画像を左右または上下に反転して出力するコンポー ネントを作成します。 作成手順としては 1)コンポーネントの仕様を決定 2)RTCBuilder を用いたソースコードのひな形の作成 3)アクティビティ処理の実装 4)コンポーネントの動作確認 になります。

1. cvFlip 関数について

cvFlip 関数は、OpenCV で標準的に用いられている関数です。入力された画像データを反転さ せて出力する機能があります。反転させる軸は垂直軸、水平軸、両軸と三種類あり引数で設定 することが出来ます。

2. コンポーネントの仕様

これから作成するコンポーネントを Flip コンポーネントという名称にします。

このコンポーネントの動作としては画像データを入力ポート(InPort)から受け取り反転処理した画像データを出力ポート(OutPort)へ出力します。

それぞれのポートの名前を入力ポート(InPort)名:originalImage,出力ポート(OutPort) 名:flippedImage とします。

これらのコンポーネントのデータポートは画像の入出力に Camera Image 型を使用しています。

また、画像を反転させる方向は、左右反転、上下反転、上下左右反転の3通りが有ります。こ れを実行時に指定できるように、RT コンポーネントのコンフィギュレーション機能を使用して 指定できるようにします。パラメータ名は flipMode という名前にします。

flipMode は cvFlip 関数の仕様に合わせて、型は int 型とし上下反転、左右反転、上下左右反転

それぞれに0,1,-1を割り当てることにします。



以上から Flip コンポーネントの仕様をまとめると下記の様になります。

コンポーネント名称	Flip
InPort	
ポート名	originalImage
型	CameraImage
意味	入力画像
OutPort	
ポート名	flippedImage
型	CameraImage
意味	反転された画像
Configuration	
パラメータ名	flipMode
型	int
意味	反転モード
	上下反転:0
	左右反転:1
	上下左右反転:-1

3. Flip コンポーネントの雛型の生成

Flip コンポーネントの雛型の生成方法を説明します。

1. RTCBuilder の起動

OpenRTP を起動させると作成物を保存するディレクトリを指定します。ここでは下記ディレクトリに保存します。

[/home/<ユーザ名>/rtcws]

最初に起動したとき下記画面がでます。この画面は使用しないので左上の×ボタンを押します。



×ボタンを押すと下記画面が表示されます。右上の[Other...]をクリックしてください。



GVS Rep	ository Explor	ing	
Debug		-	
🐉 Java (de	fault)		
🔊 Java Bro	owsing		
🖁 Java Typ	e Hierarchy		
🕪 Plug-in 🕻	Development		
눱 Resourc	e		
रू RT Syste	em Editor		
🚰 RTC Buil	der		
⁰ Team Sy	nchronizing		

下記ウィンドウが出ますので[RTC Builder]を選択します。

「RTC Builder」を選択することで、RTCBuilder が起動します。メニューバーに「カナヅチ と RT」の RTCBuilder のアイコンが現れれば完了です。

2. 新規プロジェクトの作成

画面上部のメニューから[File]-[New]-[Project...]を選択



「New Project」画面において、「Other」-「RTC Builder」を選択し、「Next >」をクリック

😣 💷 New Project
Select a wizard
Wizards:
type filter text
🕨 🗁 General
CVS
Eclipse Modeling Framework
🕨 🗁 Java
🕨 🗁 Plug-in Development
🕨 🗁 Xcore
🔻 🗁 Other
nt RTC Builder na her state and stat
🕨 🗁 Examples
? < Back Next > Cancel Finish

「Project name」欄に作成するプロジェクト名 (ここでは Flip) を入力して「Finish」をクリックします。

😣 🗉 RT-Componen	t Builder Project			
Project name: Flip				
👿 Use default locat	ion			
Location: /home/ev	3/rtcws/Flip			Browse
?	< Back	Next >	Cancel	Finish

下記画面の様にパッケージエクスプローラ内にプロジェクトが追加されれば完了です。



3. RTC プロファイルエディタの起動

基本的に RTC.xml が生成された時点で、このプロジェクトに関連付けられているワー クスペースとして RTCBuilder のエディタが開くはずです。

もし開かない場合は、「カナヅチと RT」の RTCBuilder のアイコンを押下します。



 プロファイル情報入力とコードの生成 一番左の「Basic」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:Flip
モジュール概要:Flip component
バージョン:1.0.0
ベンダ名:Aizu
モジュールカテゴリ:Category
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、「Activity」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。 Flip コンポーネントでは、onActivated0,onDeactivated0,onExecute0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。

File Edit Source Refactor	Navigate Search Proje	ect Run Window Help		💷 👣 🕪)) 6
😣 🗐 🗊 RTC Builder - Flip/	RTC.xml - Eclipse SDK			
] 🖬 🔹 📾 📾 🖢] 🚓]	\$* • ○ • ♀ •] @	• ≁ ▼] ☆ ▼ ☆ ▼ ☆ ▼ ☆ ▼	1	⊸ ″
📙 Package Ex 🛛 🗖 🗖	א ∗Flip א			- 0
□ 🛓 マ	Con	nponent Action concerning the ExecutionContext's st	artup and shutdown	o
🔻 🚔 Flip	onStartup	onShutdown		o
		Component Action in the alive state		o
	onActivated	onDeactivated	onAborting	q
	GILITOI	Dataflow Component Action		
	onExecute	onStateUpdate	onRateChanged	c
		FSM Component Action		c
	onAction			c
		Mode Component Action		c
	onModeChanged			
	 Documentation 			P F
	This section specifies If the action above is	a short description of each action. selected, each document can be described.		P
	Activity name :	onActivated	10 0	
	Basic Activity Data Po	rts Service Ports Configuration Documentation La	nguage and Environment	RTC.xml
	BuildView 🕱			
		Elin		
□°	🛱 BuildView 🏼	Flin		

最終的に下図の様になります。

onInitialize	onFinalize			
Co	omponent Action concerning the ExecutionContext's sta	rtup and shutdown	i.	
onStartup	onShutdown			
	Component Action in the alive state			
onActivated	Activated onDeactivated onAborting			
onError	onReset			
	Dataflow Component Action			
onExecute	onStateUpdate	onRateChanged		
	FSM Component Action			
onAction				
	Mode Component Action			
onModeChanged				
- Documentation				
This section specific If the action above i	es a short description of each action. is selected, each document can be described.			
Activity name : OnExecute 💿				

「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。[Add]ボタンを押して新しいデータポートを追加します。

• InPort
ポート名:originalImage
データ型: RTC::CameraImage
変数名:originalImage
表示位置:LEFT

・OutPort ポート名: flippedImage データ型: RTC::CameraImage 変数名: flippedImage 表示位置: RIGHT

File Edit Navigate	Search Project Run Window Help	💷 t _i	●)) 6:27 PM 🔱
] 🗈 🔻 🔛 🔞 🗁	A] Q ▼] ペ ▼] ઙી ▼ δl ▼ ♡ ◇ ▼ ○ ▼	1	÷ "
1 Pa 🛛 🗖	> *Flip 🛙		- 8
□ 🔄 マ	Data Ports		
🕨 🗁 Flip	▼ DataPort Profile	* Hint	
	This section defines RT-Component's DataPort Profile information.	Data Ports :	A port to send and
	*Port Name (DataInPort) Add *Port Name (DataOutPort) Add		There are an OutP
	2) originalImage FlippedImage		
	Detet	InPort :	A port to input da It is connected wit
	✓ Detail		
	This section specifies a short description of each Data Port.	OutPort :	A port to output d It is connected wit
	If data port above is selected, each port can be described.		
	Port name : originalimage (InPort)	Port name :	Specifies the port The port name nee
			OutPort and Servi
	*Data Type RTC::Cameralmage 🗘		ASCII is available.
	Var Name originalImage		
	Disp. Position LEFT \$	Data type :	Specifies a data ty To connect an InPo
	Documentation		at least this data t
			provided by OpenF
	Port description :	Variable pame :	Specifies the varia
	Basic Activity Data Ports Service Ports Configuration Documentation Language and Environment	RTC.xml	specifies the varial
	BuildView 8		- 0
		_	
	гар		
1 =*	1		
] "			

「Configuration」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、flipModeの値は先ほど仕様を決めたときに、-1,0,1の3つの値のみ取るこ とにしたので、ラジオボタンを使用することにします。[Add]ボタンを押して新しいコ ンフィギュレーションを追加します。

名称: flipMode
データ型: int
デフォルト値: 1
変数名:flipMode
制約条件: (-1, 0, 1)
Widget: radio

File Edit Navigate	Search Project Run Window Help	🎞 🗊 🗊 💷 💷	мψ
] 🗈 🔹 🖶 ち	▲] Q ▼] ダ ▼] 원 ▼ 한 ▼ ♥ ♥ ♥ ♥ ♥	Ľ	- "
🗎 Pa 🕺 🗖 🗖	א Flip צ		
□ 🔄 🍷	RT-Component Configuration Parameter		
🕨 🗁 Flip	RT-Component Configuration Parameter Definitions	★ Hint	
	(7) his section defines RT-Component Configuration Parameter.	Config. Param :	A para
	*Name Add		The Co
	flipMode		In ord
	Delec		the pa in RT-
		Parameter name :	Specil The p
	- Detail		Only a
	This section specifies each Configuration Parameter description.	Data type :	Specif
	Parameter name : flipMode	bata type .	Vector
		Default value :	Specif
	Type Inc V	Deraute value.	The de
	*Default Value 1		but als
	Variable name : flipMode	Variable name :	Specif
	Unit:		Actua
	Constraint : (-1,0,1)	Unit :	Specif
	Basic Activity Data Ports Service Ports Configuration Documentation Language and Environment RTC xml	Constraint :	Specif
	Σ originalImage FlippedImage		
	Flip		
	1		
j ∎°			

「Language and Environment」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。

File Edit Navigate	Search Project Run Window Help	💷 👣 🗤) 6:30 PM 🔱
] 🖬 🔻 🔛 🕼 🗁 [🏦] 💁 후] 🖋 후] 원 후 행 후 🏷 수 후 수 후	🖹 🔻 "
😫 Pa 😫 🗖 🗖	א +Flip מ	- 0
E 🔄 🎽	Language and Environment	
🕨 🗁 Flip	✓ Language	- Hint
	This section defines a language that is used. C++ Java Python	Language: Selects a programming language Environment: Selects the dependency in librar such as OS that is used. The set contents (Oses and Libra is saved only within Profile.
	Ruby Use old build environment.	
	✓ Environment	
	This section defines depending libraries and Oses etc that are used.	
	Version OS Add Delete	
	Detail information	
	OS Version Add CPU Add	
	Basic Activity Data Ports Service Ports Configuration Documentation Language and Er	vironment RTC.xml
		1
	Σoriginalimage	flinnedImage
	Flip	
] 0*]

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。

File Edit Navigate	Search Project Run Window H	elp	📖 👣 📢)) 6:3	зрм 🕁
] 🗈 🕶 🔡 🖷 🗎] 🚓] 💁 🔻] 🖋 🔻] 🖄 💌 🖗	* 🐎 🗇 * 🗘 *	B	~ ″
1 Pa 🕱 🗖 🗖	⊁ *Flip ☎			- 0
⊑ 🔄 ▽	Execution type :	PeriodicExecutionContext	÷	Only alph
🕨 🗁 Flip	Execution rate :	0.0	Component type :	Specifies
	Abstract :			 STATIC UNIQU COMM
	RTC Type :		Component's activity type :	Specifies
	Code Generation and Package	ging		
	Generates code and Packaging.			
	Code Generation Packaging		Component kind :	Specifies
	Import/Export Profile inform	nation		·DataFlov ·FiniteSta
	Imports/Exports Profile inform	ation		∙MultiMo
	Import		Number of maximum instance :	Specifies Specify '(
			Execution type :	Specifies
			Execution rate :	Specifies This confi when the
			Abstract :	Specifies
	Basic Activity Data Ports Service	Ports Configuration Documentation Language and En	vironment RTC.xml	
	📦 BuildView 🛿			- 0
	Σ	orininalImage		
		Flip		
[]] ∎¢			1	
1 -				

5. 仮ビルド

ここまでの作業で Flip コンポーネントの雛型が生成されました。中身の実装は出来ていま せんがこの状態でコンパイルは実行できます。

Linux 環境の場合 Flip コンポーネントのソースが生成されたディレクトリで下記コマンド をうちます。

\$ cd rtcws/Flip
\$ mkdir build
\$ cd build
\$ cmake
\$ make

これで Configure およびビルドが完了します。問題なく出来ることを確認してください。

4. ヘッダ、ソースの編集

1. アクティビティ処理の実装

Flip コンポーネントでは、InPort から受け取った画像を画像保存用バッファに保存し、その保存した画像を OpenCV の cvFlip0関数にて変換します。その後、変換された画像を OutPort から送信します。

onActivated0,onExecute0,onDeactivated0での処理内容は下記図になります。



Flip コンポーネントの処理の流れは以下の図になります。



2. ヘッダファイル (Flip.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。

//OpenCV 用インクルードファイルのインクルード #include<cv.h> #include<cxcore.h> #include<highgui.h>

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:の中(// <rtc-template block="private_attribute">の下)に追加してください。

IplImage* m_imageBuff; IplImage* m_flipImageBuff;

3. ソースファイル (Flip.cpp) の編集

下記のように、onActivated0,onDeactivated0,onExecute0を実装します。

```
onActivated()
```

```
RTC::ReturnCode_t Flip::onActivated(RTC::UniqueId ec_id)
{
    // イメージ用メモリの初期化
    m_imageBuff = NULL;
    m_flipImageBuff = NULL;
    // OutPortの画面サイズの初期化
    m_flippedImage.width = 0;
    m_flippedImage.height = 0;
    return RTC::RTC_OK;
}
```

onDeactivated()

```
RTC::ReturnCode_t Flip::onDeactivated(RTC::UniqueId ec_id)
{
    if(m_imageBuff != NULL)
    {
        // イメージ用メモリの解放
        cvReleaseImage(&m_imageBuff);
        cvReleaseImage(&m_flipImageBuff);
    }
    return RTC::RTC_OK;
}
```

onExecute()

```
RTC::ReturnCode_t Flip::onExecute(RTC::UniqueId ec_id)
 // 新しいデータのチェック
 if (m_originalImageIn.isNew()) {
   // InPort データの読み込み
   m_originalImageIn.read();
   // InPort と OutPort の画面サイズ処理およびイメージ用メモリの確保
  if( m_originalImage.width != m_flippedImage.width || m_originalImage.height != m_flippedImage.height)
  m_flippedImage.width = m_originalImage.width;
  m_flippedImage.height = m_originalImage.height;
   // InPort のイメージサイズが変更された場合
   if(m_imageBuff != NULL)
     ł
       cvReleaseImage(&m_imageBuff);
       cvReleaseImage(&m_flipImageBuff);
     }
   // イメージ用メモリの確保
   m_imageBuff = cvCreateImage(cvSize(m_originalImage.width, m_originalImage.height), IPL_DEPTH_8U, 3);
   m_flipImageBuff = cvCreateImage(cvSize(m_originalImage.width, m_originalImage.height), IPL_DEPTH_8U, 3);
     }
   // InPortの画像データを IplImageの imageData にコピー
   memcpy(m_imageBuff->imageData,(void *)&(m_originalImage.pixels[0]),m_originalImage.pixels.length());
   // InPort からの画像データを反転する。 m_flipMode 0: X 軸周り, 1: Y 軸周り, -1: 両方の軸周り
   cvFlip(m_imageBuff, m_flipImageBuff, m_flipMode);
   // 画像データのサイズ取得
   int len = m flipImageBuff->nChannels * m flipImageBuff->width * m flipImageBuff->height;
   m_flippedImage.pixels.length(len);
   // 反転した画像データを OutPort にコピー
   memcpy((void *)&(m_flippedImage.pixels[0]),m_flipImageBuff->imageData,len);
   // 反転した画像データを OutPort から出力する。
   m_flippedImageOut.write();
 }
 return RTC::RTC_OK;
```

5. CMake によるビルドに必要なファイルの生成

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 Flip.cpp) set(standalone_srcs FlipComp.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 \${comp_srcs} \${comp_headers} \${ALL_IDL_SRCS}}

6. ビルド

CMakeList.txt を編集したので、再度 CMake で Configure およびビルドを行います。

\$ cd rtcws/Flip (eclipse ワークスペース下の Flip ディレクトリへ移動)
\$ rm -rf build (念のため仮ビルドで作成したディレクトリは削除)
\$ mkdir build (再度 build ディレクトリを作成)
\$ cd build
\$ cmake ..
\$ make

エラーが出た場合は CMakeLists.txt やソースコードに間違えがないか確認してください。

7. コンポーネント起動

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。

\$ rtm-naming

(y/N)と聞かれたら y を選択してください。

2. Flip コンポーネントの起動

Flip コンポーネントを起動します下記コマンドで実行します。

\$ cd rtcws/Flip/build/src (もし現在 build/src 以外にいる場合) \$./FlipComp

3. カメラコンポーネントとビューアコンポーネントの起動 USB カメラのキャプチャ画像を OutPort から出力する OpenCVCameraComp と InPort で受け取った画像を画面に表示する CameraViewerComp を起動します。

\$ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/OpenCVCameraC
omp
\$ /usr/local/share/openrtm-1.1/components/c++/opencv-rtcs/CameraViewerCo
mp

8. コンポーネントの接続

WEB カメラから画像が取得できることを確認した後、Flip コンポーネントを確認します。

1. RTSystemEditorの起動

最初に RTSystemEditor を起動します。

起動方法は RTCBuilder 画面右上の「Open Perspective」を選択し、さらに[Other...]を選 択します。「Perspective」の中から[RT System editor]を選択して起動させます。

File Edit Navigate Search Pr	oject Run Window Help	■ 1 ↓ •>))	6:36 PM	Ψ
] 🗗 🖌 📓 📾 🗋 🗟 🌆]	💁 🕶] 🛷 🕶] 🔄 👻 🖏 👻 🖘 🗢 🗢) 📂 👹 🎆 🎲	B		. "
🗯 Nam 😫 🔞 Rep 🗖 🗖	א *Flip 🕷 אין		- 0	-
▽				
🏠 🗘 🎝 📑 🎥 🤣 🎽				
▼ 127.0.0.1 Tא				
ubuntu host_cxt				
CameraViewer0 rtc				
Flip0 rtc				
OpenCVCamera0 rtc				
	Configuration View 🕱 🛛 Manager Control Vi 🚮 Composite Compo 🚮 Execution Context	T RT Log View		
	componencivame: ConrigurationSet:	Edi	t Valu	
	Copy Add Delete Add	Delete	pply	
] 0*]			

2. コンポーネントの接続

Name Service View に何も表示されていない場合は、RTSystemEditor の左側の Name Service View のコンセントアイコンをクリックし、ネームサーバへ接続します。表示され た接続ダイアログに[localhost]か[127.0.0.1]と入力します。

🖲 Conn	ect Name S	erve	r
ease enter	Name Serve	r Ado	dress.
localhost		v	(Address:Port)
	Cancel		OK
	Cancel		OK

Name Service View にリストが表示されます。

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、 SystemDiagram を開きます。

次に Name Service View から OpenCVCameraComp と CameraViewerComp ををドラッ クアンドドロップで SystemDiagram 上にコンポーネントを配置してください。

コンポーネントのデータポート同士を接続します。片方のデータポート上でドラッグする と線が伸びるので、接続したいデータポート上まで線を伸ばし接続します。 接続すると接続プロファイルが表示されるので OK をクリックします。

File Edit Navigate Search Pr	oject Run Window Help	C	1 , ())	6:36 PM	ψ
] 🗗 🖌 📓 📾 🗍 🔍 🍇]	६. र] अ र ही र ७२ ०२ र] ⊯ ≝ ३३ ३३		B	*	, "
🗯 Nam 😫 🍈 Rep 🗖 🗖	≯*Flip ๗ *System Diagram. גז			- 0	8
 ▼ AT 127.0.01 ▼ Ubuntujhost_cxt ○ GmeraViewer0]rtc ○ Flip0]rtc ○ OpenCVCamera0]rtc 	OpenCVCamera0 OpenCVCamera0 CameraViewer0 CameraViewer0 CameraViewer0	ext हो। हो।	- Log Viev		
	ComponentName : ConfigurationSet :			dit Valu	
	Copy Add Delete Add	Delet	te	Apply	
] 0°					

接続が完了すると下記図の様になります。

3. コンポーネントの Activate

RTSystemEditorの上部にあります「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。

[] ▼ 📓 🕼 🖾] की कि⊑]	🌯 ▾] 생 ▾ 한 ▾ ♡ ▾ ♡ ▾ 🔛 👹 🏙 🍪 🏦	B	
Nam 🛛 🕅 Rep 🗖 🗖	▶ *Flip 🔐 *System Diagram. 😫		- 0
 ▼ ↓ ↓	OpenCVCamera0 CameraViewer0		
	Configuration View 회 제 Manager Control Vi 챔 Composite Compo 챔 Execution Contex	t RT Log View	
	Configuration View X II Manager Control Vi II Composite Compo II Execution Contex	tt 🕶 RT Log View) Iit Valu

4. 動作確認

ウィンドウが出てきてカメラから取得した画像が表示されることを確認してください。



確認できましたら赤色の[ALL]のボタンをクリックしてください。コンポーネントがディア クティブ状態になり、動作が終了します。

5. Flip コンポーネント動作確認

線を delete で消した後、下図の様に接続し緑の[ALL]をクリックしてください。

File Edit Navigate Search Pr	oject Run Window Help	📖 🏚 🕪) 6:36 PM 🔱
] 📬 🔻 🔛 🗟 🗁] 💀 🐜]	Q ₄ ▼] タ ▼] 約 ▼ 約 ▼ ♥ ♀ ▼ ↔ ▼] ⊮ ≝ 8	? i? 🗈 👻 "
# Nam 🕱 👘 Rep 🗖 🗖	א *Flip 💀 *System Diagram. 🛛	
▼ ▼ x7 127.0.0.1 ▼ @ ubuntu]host_cxt ③ CameraViewer0]rtc ③ Filp0]rtc ④ OpenCVCamera0]rtc	OpenCVCamera0 Flip0	age in Key_out Mouse_event Mouse_Y_pos CameraViewer0
	🔲 Configuration View 🛱 🛛 Manager Control Vi 🚺 Compos	site Compo 🔃 Execution Context 🔃 RT Log View 👘 🗖
	ComponentName : ConfigurationSet :	[Edit Valu]
	Copy Add Delete	Add Delete Apply
] 📭]

下図のようにコンフィギュレーションビューにてコンフィギュレーションを変更することができます。 Flip コンポーネントをクリックしてコンフィギュレーションビューの編集を押すと下記ダイアログが出て きます。「flipMode」を「0」や「-1」などに変更し画像が反転することを確認してください。

m Editor Eclipse SDK	C	III 🕕 🕬) 6:40 PM 🔱
] 📬 🔻 🔛 🕼 🗁] 💀 👘 [🌯 ▾] 🔗 ▾] 회 ▾ 한 수 ▾ 수 ▾] 🖬 🖉 🌮	Ē	~ "
🗯 Nam 😫 🍈 Rep 📃 🗖	א *Flip מוּ *System Diagram. צ		
~			er
🍐 (A) - 🕞 🌲 🚸 🗯			
דא ▼ 127.0.0.1			
ubuntu host_cxt			
CameraViewer0 rtc	out originalImage TippedImage in Key out		
Flip0 rtc		ent	
OpenCVCamera0 rtc	OpenCVCamera0 Flip0 Mouse_X_	pos	
	Mouse_Y_	pos	
	CameraViewer0		
	S Configuration		
	default		
	ConfigurationSet : default		
	Config ? Cancel OK	RT Log Viev	<u> </u>
	Compor	E	dit Valu
	Copy Add Delete Add De	lete	Apply
] 🗠]		

3. Flip コンポーネントの全ソース

- Flip コンポーネントソースファイル (Flip.cpp) Flip.cpp のソースコードを以下に記載します。 Flip.cpp : https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip_cpp.txt
- Flip コンポーネントのヘッダファイル (Flip.h) Flip.h のソースコードを以下に記載します。 Flip.h:https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip_h.txt
- Flip コンポーネントの全ソースコード
 Flip コンポーネントの全ソースコードを以下に添付します。
 Flip.zip: https://rtc-fukushima.jp/wp/wp-content/uploads/2016/02/Flip.zip