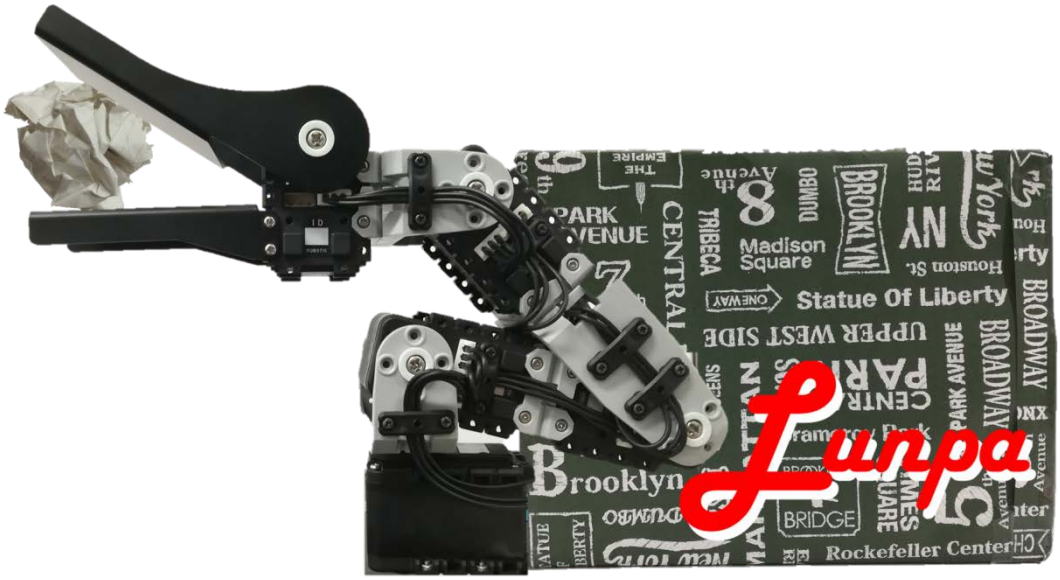
2018 年 10 月 23 日



マニュアル

芝浦工業大学　デザイン工学部　デザイン工学科
エンジニアリングデザイン領域
メカトロニクス・組み込みソフトウェア分野

プロジェクト演習 6

B 班 – 1 班 Suica

CY15232　伴 晃一

CY16153　長倉 朱里

CY16173　八木橋 晃一

CY16174　山木 淳弘

目次

1. lunpa 概要

1.1 開発概要

今回の演習において、RT ミドルウェアを用いた「身近にあるもののロボット化」という製作課題が与えられた。成果物については本体、ソフトウェア、チラシ、マニュアル、プレゼンを通して評価される。

私たちは「ゴミを投げて外れた際にゴミ箱に捨てる機能」と「入った際には音が鳴り、快感を得られる機能」の 2 つの機能を持った装置、通称 lunpa を製作した。製作目的は、人がゴミを投げて外した際に、それを拾いなおしてくれるロボットがあれば、気軽に投げることができると考えたからである。加えて、投げてゴミ箱に入った際にその快感をより味わいたかったからである。

1.2 機能

lunpa の機能としては、ゴミを投げた際にゴミ箱に入れば効果音がなり、外れた際にはアームでゴミを拾いゴミ箱に捨てる機能である。ただし、ここでのゴミとはカメラに映る背景色と色の異なる静止した物体のことを指す。

1.3 使用の流れ

lunpa を使うにあたって簡単に使用の流れを説明する。

1)ゴミ箱に向かってゴミを投げる。

2-1)ゴミ箱にゴミが入った場合

2-1-1）lunpa から音が鳴る。

2-2)ゴミ箱にゴミが入らなかった場合

2-2-1)バケットが開いた状態でアームが初期位置からゴミの位置まで移動する。

2-2-2)バケットが閉じ、ゴミを掴む。

2-2-3)アームがゴミ箱へゴミを運び、バケットが開きゴミを離す。

2. コンポーネント説明

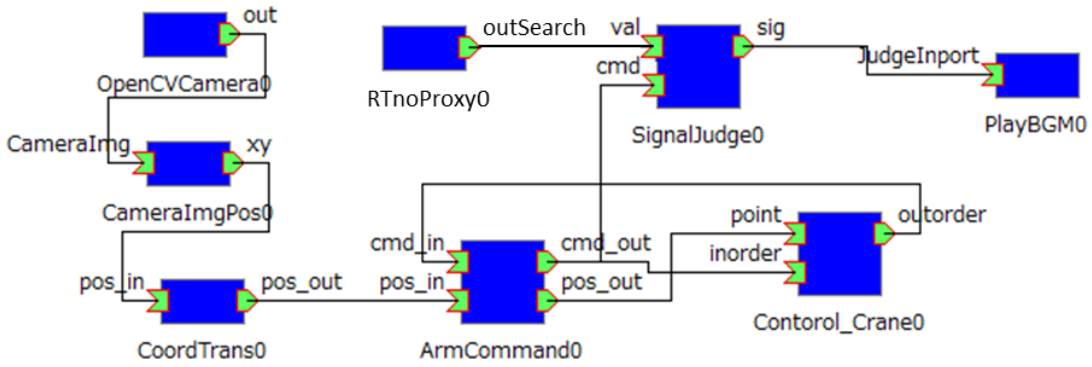


図 1　使用したコンポーネントをつなげた図

基本的な操作を提供するものとして以下のコンポーネント群を開発・利用した。
本章では各コンポーネントの詳細な説明をする。

＊色の識別/座標の特定コンポーネント(CameraImgPos)
カメラ画像から対象物の重心座標をピクセル座標で出力するコンポーネント

＊アーム座標変換コンポーネント(CoordTrans)
入力された座標の単位をピクセルからｃｍに変換し、アームを中心とした座標に変換するコンポーネント

＊アーム命令コンポーネント(ArmCommand)
アームの動きを指定するコンポーネント

＊アーム制御コンポーネント(Contorol_Crane)
指定された動きを実行するコンポーネント

＊センサーコンポーネント(RTnoProxy)
センサーを用いて物体との距離を測定するコンポーネント

＊信号判定コンポーネント(SignalJudge)
人とアームのどちらが捨てたかを判定するコンポーネント

＊スピーカーコンポーネント(PlayBgm)
音楽ファイルの再生または停止をするコンポーネント

2.1　色の識別/座標の特定コンポーネント(CameraImgPos)

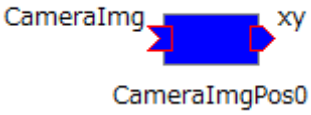


図 2　CameraImgPosComp

CameraImgPos は、カメラ画像を処理し対象物の重心座標を出力する RT-Component である。CameraImgPos では、USBCamera から入力されたカラー画像に対して、グレースケール化と 2 値化の処理を施し、対象物の重心座標(画像のピクセル座標)を計算し出力する。

以下詳細な処理の内容を記述する。

まず、OpenCV2.4.9 ライブラリ付属の OpenCVCamera コンポーネントから、パソコンに接続された USBCamera の CameraImage 型のカラー画像を、CameraImgPos コンポーネントに入力する。入力されたカラー画像を CameraImage 型から Mat 型に変換する処理を行った後、cvtColor 関数を用いてカラー画像からグレースケール画像へ変換する。**(文献 1)**グレースケール化の後、threshold 関数を用いて 2 値化処理を施す。通常この関数では閾値処理を設定するが、幅広く活用できるよう大津のアルゴリズム(引数 thresh において THRESH_OTSU を適用)を用いて毎回最適な閾値を決定している。**(文献 1)**また、この 2 値化された画像では、対象物は白に、その他は黒に処理される。2 値化後 Moments 関数を用いて、画像をもとにピクセル座標として重心(x,y)を求め**(文献 2)**、求めた重心座標(x,y)を TimedPoint2D 型として出力する。また、処理後のキャプチャとして**表 1** の通り 3 つ用意した。

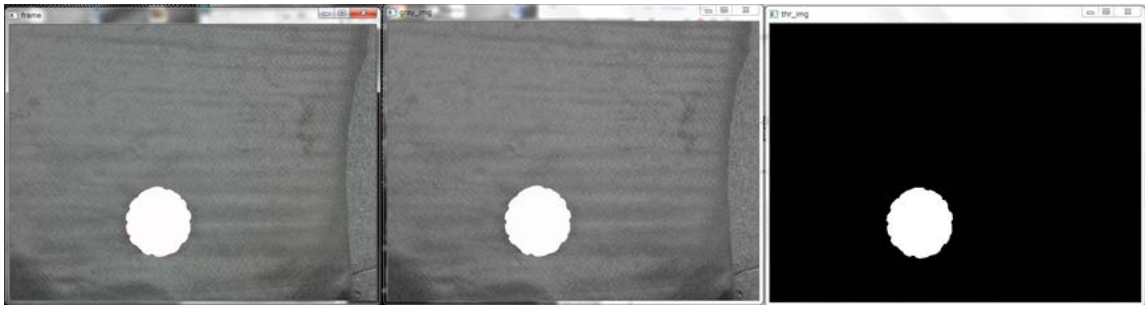
表 1　CameraImgPos キャプチャ

キャプチャ順	キャプチャウインドウ名	中身
i	frame	処理前の入力されたカラー画像
ii	gray_img	グレースケール化された画像
iii	thr_img	2 値化された画像

これは、2 値化での大津のアルゴリズムにおいて、画像の閾値が毎度調整されるために、思い通りの処理が行かない可能性があるため、トラブルシューティングとして用意した次第である。ここで以下**図 3-図 5** においていくつかの例を紹介する。なおここでの図番号(i),(ii),(iii)は**表 1** の項目”キャプチャ順”に準拠する。

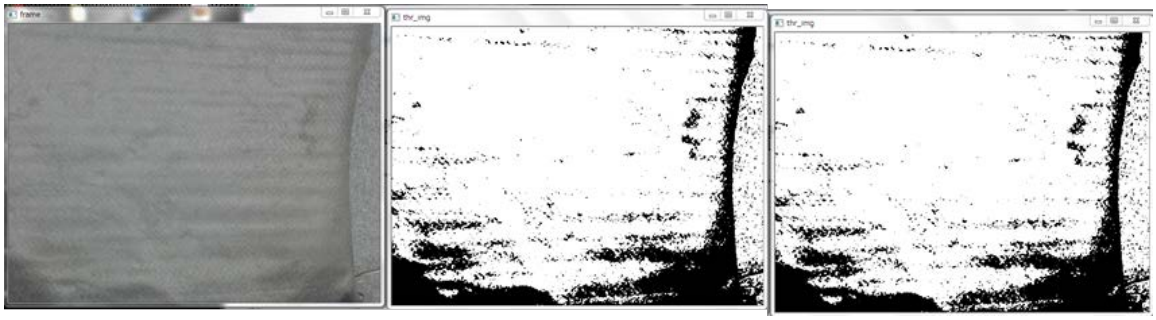
(i)　　(ii)　　(iii)

図 3　カラー物体を読み込んだときのキャプチャ



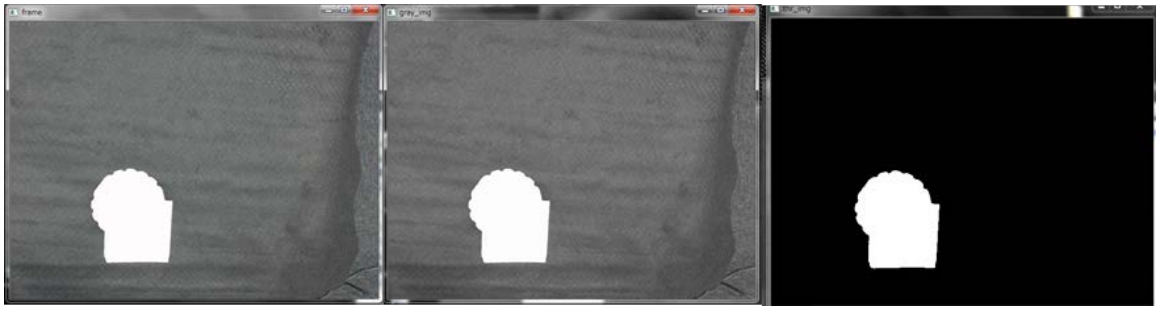
(i)　　(ii)　　(iii)

図 4　ゴミ(パターン 1)のキャプチャ



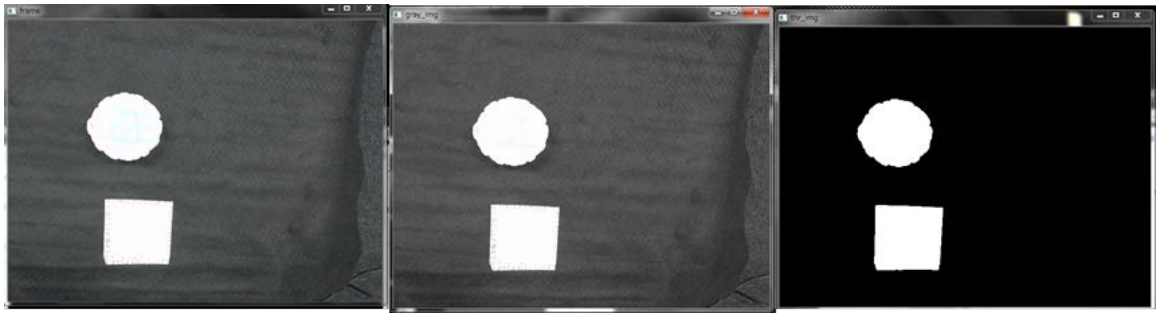
(i)　　(ii)　　(iii)

図 5　物体がなくなったときのキャプチャ



(i)　　(ii)　　(iii)

図 6　ゴミ(パターン 2)のキャプチャ



(i)　　(ii)　　(iii)

図 7　ゴミ(パターン 3)のキャプチャ

図 3 は実装に用いた対象物ではなく、カラー物体を読み込んだときの処理である。図 3-(i)ではカメラに映った画像、つまり私達が目にするままの状態が入力されていることが見て取れる。図 3-(ii)では入力された画像がグレースケール化され、図 3-(iii)では 2 値化されたことがわかる。

図 4-7 までは実装に用いた対象物の例である。まず**図 4** だが、これは球形の物体を想定した処理である。この場合図 4-(iii)の白い部分の重心(x,y)を出力する。次に**図 5** であるが、これは物体を置かず背景だけを映したものである。**図 4** と背景は全く変わっていないものの、大津のアルゴリズムに基づいた 2 値化処理によって、画像内の相対閾値となっているため、背景のフェルト地の折り目の光の反射が入ってしまっている。したがってこのような状態を回避するために、今回の背景の場合は次の**図 8** のようなもの(以下、目印)を背景上に設置すると良いだろう。

図 8　背景バグを防ぐための対策(目印)

これについては、次項記述の 2.2 座標変換コンポーネントに互換性がある。目印はコンフィギュレーション変数における設定に用いる。

図 6-7 ではその目印と物体を置いたパターンである。**図 6** は、目印と物体が重なった図形の重心(x, y)を出力する。一方、**図 7** は目印と物体が離れているが、この場合にはそれぞれの重心を繋いだ線分の中心座標(x, y)が出力される。

以上を踏まえてユーザー用トラブルシューティングとしては次の**表 2** の通りである。

表 2　CameraImgPos トラブルシューティング

トラブル内容	原因	解決策
物体を物体として認識できない	・目印がない ・目印が背景に対して小さすぎる ・背景が安定していない ・カメラの影が映り込んでいる ・照明の位置	・目印の改善 ・背景の素材改善 ・照明の位置を真上のなるべく高い位置に変える、もしくはなくす
物体から少し離れたところの座標が出力される	・目印と物体が離れている	・目印と物体それぞれの重心を繋いだ線分の中心座標が出力される仕様のため、気にする必要はない
物体の重心とはまるで思えないところの座標が出力される	・背景が反転している ・背景以外何も置かれていない	・目印の改善

2.1.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.1.2

2.1.2　依存ライブラリ

・OpenRTM-aist-1.1.2

・OpenCV2.4.9

2.1.3　データポート

・入力

表 3　CameraImgPos 入力ポート

名称	型	説明
CameraImg	CameraImage	USBCamera からの入力画像。

・出力

表 4　CameraImgPos 出力ポート

名称	型	説明	出力範囲
xy	TimedPoint2D	ゴミの重心座標(x, y)が、ピクセル座標で出力される。	0≦x≦340 0≦y≦250

・コンフィギュレーション変数

なし

2.2　座標変換コンポーネント(CoordTrans)

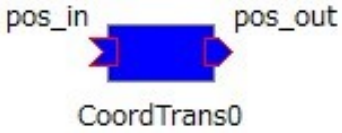


図 9　CoordTransComp

カメライメージ内の位置を表す座標を受け取ると、アーム制御に利用できる実際の長さに対応した座標に変換して出力する。入力される座標は、検出精度を上げるための目印と目標物の座標の平均となる。平均の座標を目標物の座標に変換するため、目印の座標をパラメータとして設定する必要がある。

2.2.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.0.0

2.2.2 依存ライブラリ

・OpenRTM-aist-1.1.0

・OpenCV2.4.9

2.2.3　データポート

・入力

表 5　CoordTrans 入力ポート

名称	型	説明	入力範囲
pos_in	RTC::TimedPoint2D	カメライメージ内の位置を表すピクセル座標を受け取る。	特になし

・出力

表 6　CoordTrans 出力ポート

名称	型	説明	出力範囲
pos_out	RTC::TimedPoint3D	アームの根元にあたる部分を原点とした実際の座標を渡す。単位は cm とする。	特になし

・コンフィギュレーション変数

表 7　CoordTrans コンフィギュレーション変数

名称	型	説明	入力範囲
arm_x	double	出力する座標の原点 x 成分を設定するパラメータ。出力する際に 0 にしたい点のピクセル座標 x 成分を入力する。	特になし
arm_y	double	出力する座標の原点 y 成分を設定するパラメータ。出力する際に 0 にしたい点のピクセル座標 y 成分を入力する。(今回は仕様上カメライメージ内にアームが映ってはいけない。そのため arm_y を測定する際は、アームの根元の正面部分に目印を置き、その目印のピクセル座標を入力する。実際のアームの位置との差はプログラム内で修正を行っている。)	特になし
mark_x	double	目印の座標 x 成分のパラメータ。目印のピクセル座標 x 成分を入力する。	特になし
mark_y	double	目印の座標 y 成分のパラメータ。目印のピクセル座標 y 成分を入力する。	特になし
mark_cm	double	アームの y 座標成分から目印の y 座標成分までの長さを入力する。長さは cm とする。	特になし

2.3　アーム命令コンポーネント(ArmCommand)

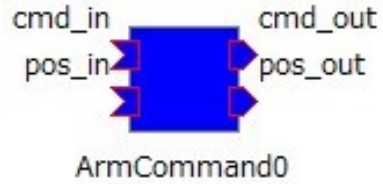


図 10　ArmCommandComp

アーム制御に用いる座標を受け取ると、アームを動かすために対応した命令と座標を出力する。また、アーム制御コンポーネントから直前に行った処理を表す文字を受け取り、次に行う処理に対応した命令を出力する。

2.3.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.0.0

2.3.2 依存ライブラリ

・OpenRTM-aist-1.1.0

・OpenCV2.4.9

2.3.3 データポート

・入力

表 8 ArmCommand 入力ポート

名称	型	説明	入力範囲
cmd_in	RTC::TimedChar	アーム制御コンポーネントが直前に行った処理の命令を受け取る。	‘m’, ‘c’, ’o’, ’t’, ’r’のいずれかの文字
pos_in	RTC::TimedPoint3D	アーム制御に用いるアームの根元を原点とした実際の座標を受け取る。	使用するアームの対応可能な範囲

・出力

表 9 ArmCommand 出力ポート

名称	型	説明	出力範囲
cmd_out	RTC::TimedChar	アーム制御コンポーネントが次に行う処理の命令を渡す。	‘m’, ‘c’, ’o’, ’t’, ’r’のいずれかの文字
pos_out	RTC::TimedPoint3D	アーム制御に用いるアームの根元を原点とした実際の座標を渡す。	使用するアームの対応可能な範囲

・コンフィギュレーション変数

表 10 ArmCommand コンフィギュレーション変数

名称	型	説明	入力範囲
tx	double	命令 t で移動する座標の x 成分	使用するアームの対応可能な範囲
ty	double	命令 t で移動する座標の y 成分	使用するアームの対応可能な範囲
tz	double	命令 t で移動する座標の z 成分	使用するアームの対応可能な範囲

2.4　アーム制御コンポーネント(Contorol_Crane)

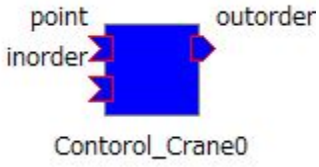


図 11　Control_CraneComp

このコンポーネントはアームに対してバケットの開閉、三次元空間における任意座標へバケットが鉛直下向きの状態となるように移動が行える。入力コンポーネントから受け取る座標は初期状態のアームの根元を原点とし、アームから見てそれぞれ、右向きに x 軸、前方に y 軸、鉛直上向きに z 軸をとり、単位は小数点を含む c m である。ただし、ここでのアームの根元は台座を含めないものとし、関節間の長さは根元側から順に 8.2cm、9.5cm、12.5cm とする。また、もう一方の入力コンポーネントからアームが行う動作を指定する文字が入力されるが、実行可能な命令は表 9 に示した通りである。表 9 に示した文字以外が入力された場合は標準出力で表１２に示したエラー文が出力される。コンフィギュレーション変数はアームが接続されているポート番号であり、ユーザーはコンポーネントをアクティベイトする前に、デバイスマネージャーのポートなどからポート番号を確認し、入力する必要がある。コンポーネントの内部動作としては、アームの土台の回転角度をθ_0とし、根元から順に関節の角度をそれぞれθ_1、θ_2、θ_3とすると、角度の対応は表１０に示すようになっており、θ_0は正の x 軸上を 0 度、θ_1は x 軸の正の方向から見て、根元を通る水平線の y 軸の正の側を 0 度、θ_2とθ_3はそれぞれの根元側のアームの延長線上を 0 度する。ただし、arm.servo[4].write.Angle はバケットの開閉角度に対応しており、150 は閉じた状態であり、数値が大きくなるにつれ開かれる。また、プログラムではアームの角度に換算する工程は最後に行っており、xy 平面上の目的地までの距離(d)は三平方の定理を用いて求めている。lunpa においては重心座標を受け取っているので、掴みたいものが大きい場合は使用者が d の計算結果を、第１、第２象限の場合は小さく、それ以外の場合大きくしなくてはならない場合がある。角度計算については、表１１に示す通り、土台が３６０度回転出来ないため、まず入力ポートから得た座標が x y 座標面において第何象限にあるのか判定し、第１、第２象限にある場合はアームの正面に、第３、第４象限にある場合はアームの真後ろに来るようθ_0を決める。次に余弦定理と三角関数の関係からθ_2を決める。この時、目的の座標が第１、第２象限にあるか第３、第４象限にあるかで$\sin\theta_2$の値は変わるので、注意が必要である。θ_1は余弦定理と三角関数の関係に加えて atan2 を用いて出している。最後にθ_3はバケットが掴む角度(θ)からθ_1とθ_2の値を引いて算出している。図 6 は各変数とアームの模式図である。

表 11　命令と文字の対応

文字	説明
mまたはｔ	入力コンポーネントから入力された座標への移動。アーム命令コンポーネントで判断する際に、移動動作と捨てる動作を区別するために２つ文字を使用。
o	バケットを開く。
c	バケットを閉じる。
r	あらかじめ設定されている体勢をとる。
l	標準出力画面で５００カウントされるまでの間の動きを記憶する。
p	記憶した動きを実行する。

表 12　現実の角度とアームの角度の対応

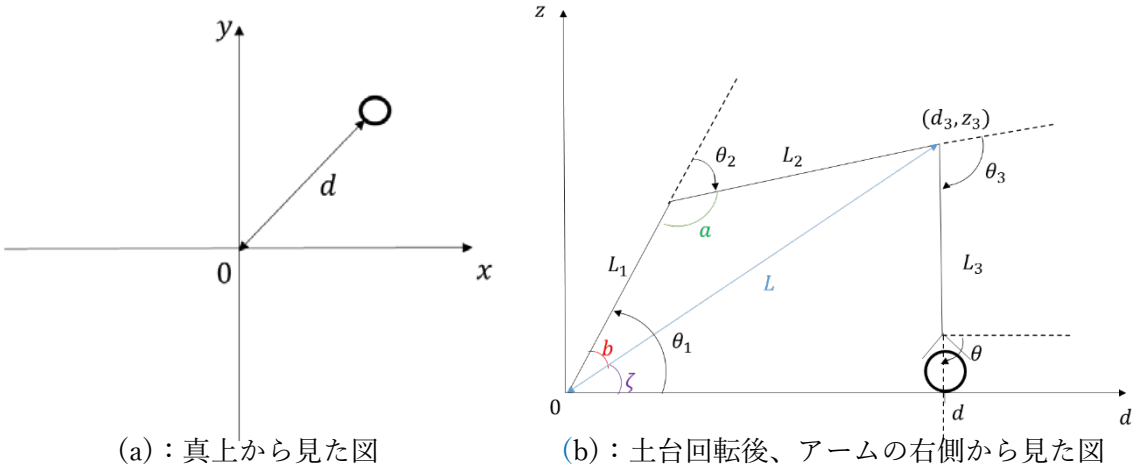
角度	0 度	90 度	180 度	−90 度	正負の基準	構造体との対応
θ_0	60	150	240	330	反時計回りを正	arm.servo[0].write.Angle
θ_1	240	150	60	270	反時計回りを正	arm.servo[1].write.Angle
θ_2	150	240	330	60	反時計回りを正	arm.servo[2].write.Angle
θ_3	150	240	330	60	反時計回りを正	arm.servo[3].write.Angle

表 13　それぞれの関節の可動範囲

可動範囲
$60 \leq arm.servo[0].write.Angle \leq 240$
$46 \leq arm.servo[1].write.Angle \leq 250$
$5 \leq arm.servo[2].write.Angle \leq 288$
$48 \leq arm.servo[3].write.Angle \leq 232$
$150 \leq arm.servo[4].write.Angle \leq 299$

表 14　エラー対応

よくあるエラーの一覧	エラーに対する対応
Error in serialWrite():device not connected: が 5 回出力される。	アームが USB ポートに接続されていないか、コンフィギュレーション変数のポート番号を間違えている可能性があります。ポート番号の確認方法は上記の説明をお読み下さい。
Error in Crane::GetStatusPacket(): invalid return value:0 0 204 204　が 1 回、Error in Crane::GetStatusPacket(): invalid return value:0 204 204 204 が 4 回出力される。	アームの電源が入っていない可能性があります。電源が入っているか今一度ご確認下さい。
計算途中の座標が可動域外です。	角度計算の過程で目標位置がアームの可動域外であることを示しています。
theta～が可動域外です: theta～の値	その角度の計算結果がアームの可動域外であることを示しています。出力された値が表１１の範囲にあるかご確認下さい。角度の対応は表１０をご確認下さい。
設定されていない文字入力です:	入力された文字が未対応です。右に出力された文字が表 9 に書かれているか今一度ご確認下さい。対応する文字を追加したい場合は、switch 文に追加して下さい。



(a)：真上から見た図　　(b)：土台回転後、アームの右側から見た図

図 12　各変数とアームとの模式図

2.4.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.0.0

2.4.2　依存ライブラリ

・OpenRTM-aist-1.1.0

2.4.3　データポート

・入力

表 15　Contorol_Crane 入力ポート

名称	型	説明	入力範囲
point	TimedPoint3D	三次元における移動後のアーム先端の座標を持つ構造体。単位はｃｍ。	アームの可動域範囲の座標。可動域外の場合は、表１２に示したエラーが出力される。
inorder	TimedChar	アームが行う動作を指定する文字。	TimedChar 型であれば入力は受け付けるが、対応していない場合、表１２に示したエラーが出力される。

・出力

表 16　Control_Crane 出力ポート

名称	型	説明	出力範囲
outorder	TimedChar	アームが行った動作を表す文字。	入力ポートに入力可能な文字。

・コンフィギュレーション変数

表 17　Control_Crane コンフィギュレーション変数

名称	型	説明	入力範囲
port	String	アームが接続しているポート番号。	指定なし

2.5　センサーコンポーネント(RTnoProxy)

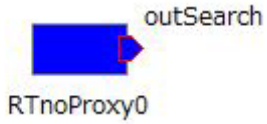


図 13　RTnoProxyComp

このコンポーネントは赤外線距離センサーで物体との距離を測定し、アナログ値をデジタル値に変換する AnalogRead 関数を使用することで、測定結果を出力するコンポーネントである。
まず、このコンポーネントを作る上では Arduino の設定をしなければなりません。最初に Arduino IDE といわれる Arduino にプログラムを書き込むソフトウェアをインストールし、バージョンも 1.0 番台のものでないと、RT ミドルウェアで使用するための RTnoProxy を使用することができません。また、使用する USB ポートを設定しなければ、動かすこともできないので注意が必要である。
プログラムの内容はセンサーから値を受け取るために AnalogRead 関数を使用し、0 番ピンからデジタル値をアナログ値として受け取り、そのアナログ値のデータを送るために、outSearch.data=distance と置くことにより、write 関数で獲得した距離データを出力にする内容となっている。

2.5.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1
・VisualC++2013
・CMake
・OpenRTP 1.0.0
・RTnoProxy
・Arduino1.0.5

2.5.2　依存ライブラリ

・OpenRTM-aist-1.1.0
・OpenCV2.4.9

2.5.3　データポート

・入力
　なし

・出力

表 18　RTnoProxy 出力ポート

名称	型	説明	出力範囲
outSearch	TimedDouble	センサーとの物体の距離を測り、値を判定するコンポーネントに送る	

・コンフィギュレーション変数

なし

2.5.4　Arduino の設定

表 19　RTnoProxy Arduino ピン

ピン番号	入力/出力	使用する関数
A0	入力	AnalogRead

2.6　センサー判定コンポーネント(SignalJudge)

図 14　SignalJudgeComp

センサーからの値を入力とし入力があったときに True の信号を出力する。このときアーム命令コンポーネントからの信号を受け取り、アームがゴミを捨てている最中は信号を送らないよう判定を行う。

2.6.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.0.0

2.6.2 依存ライブラリ

・OpenRTM-aist-1.1.0

・OpenCV2.4.9

2.6.3　データポート

・入力

表 20　SignalJudge 入力ポート

名称	型	説明	入力範囲
val	RTC::double	ゴミ箱内のセンサーが読み取ったゴミとの距離を表す値を受け取る。	特になし
cmd	RTC::TimedChar	アーム命令コンポーネントが送った命令を受け取る。	‘c’, ‘o’, ’s’, ’t’のいずれかの文字

・出力

表 21　SignalJudge 出力ポート

名称	型	説明	出力範囲
sig	RTC::Boolean	アームではない手段でゴミが入ったことを表す信号を出力する。	TRUE または FALSE

・コンフィギュレーション変数

なし

2.7　スピーカーコンポーネント(PlayBGM)

図 15　PlayBGMComp

このコンポーネントはアーム制御以外の方法でゴミがゴミ箱に入ったときに BGM を流すコンポーネントである。入力が Timed Boolean 型になっているので If 文で TRUE が送られてきたときに PlaySound 関数を使い、あらかじめ bulid の中にある debug にいれてある wav ファイルを再生されるようになっている。しかし、このままだと BGM が流れ続けるので、停止させる PlaySound 関数を追加し、BGM を止められるようにした。

2.7.1　動作確認 OS・開発環境

・Windows 7 32bit Service Pack 1

・VisualC++2013

・CMake

・OpenRTP 1.0.0

2.7.2 依存ライブラリ

・OpenRTM-aist-1.1.0

・OpenCV2.4.9

・mmsystem

・pragma comment

2.7.3 データポート

・入力

表 22　PlayBGM 入力ポート

名称	型	説明	入力範囲
Judge	TimedBoolean 型	アーム制御かそれ以外の方法でゴミが入ったかどうかを判定するコンポーネントの timed Boolean 型の出力を受け取る	TRUE または FALSE

・出力

なし(実際にスピーカーからの出音を確認する。)

・コンフィギュレーション変数

なし

3. ハードウェア(lunpa 本体)説明

3.1 使用部品

・段ボール

・赤外線距離センサー(GP2Y0A21)

・ブレットボード

・Arduino

・10uF と 0.1uF のコンデンサ(各一つずつ使用)

3.2 構造

まず、段ボールで立方体のゴミ箱を制作したあと、赤外線距離センサーを取り付けるために側面部分を切り抜き、はめ込むようにした。センサーの拾える範囲が狭いことから、砂時計ように落ちてくる箇所を限定的にすることにしました。

3.3 回路図

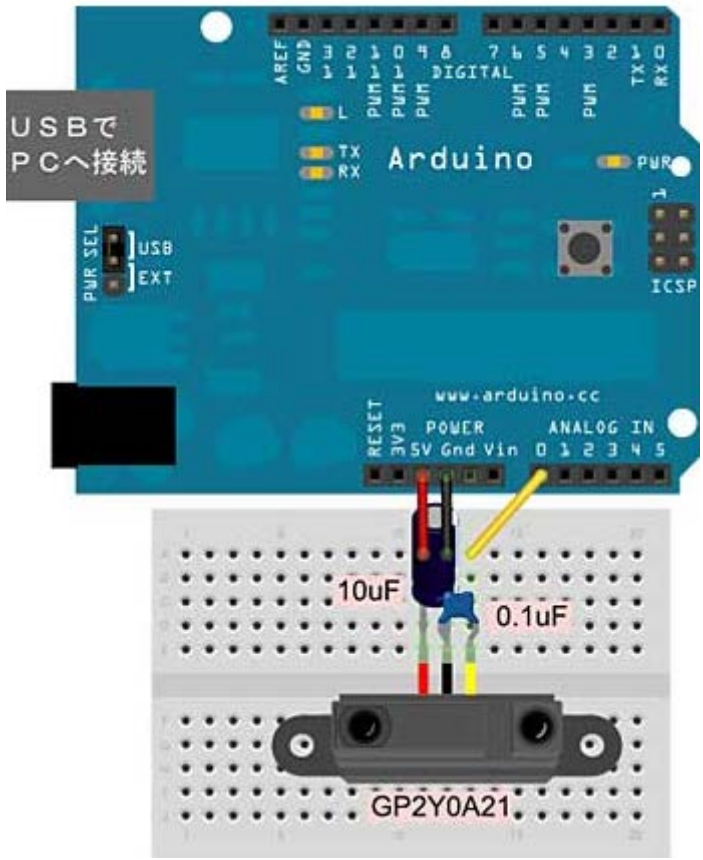


図 16　回路図

赤外線距離センサーには GP2Y0A21YK を使用した。このセンサーを使うには 10uF のコンデンサと 0.1uF のコンデンサが必要になる。これがないとセンサーが機能しない。黄色の線は 0 番ピンからデータを受け取り、黒線は grand につなぎ、赤線は 5.0V に繋ぐことで、このセンサーを使うことができる。

４． 役割分担

<コンポーネント>

・色の識別/座標の特定コンポーネント(CameraImgPos)
　長倉-100%
・アーム座標変換コンポーネント(CoordTrans)
　伴-100%
・アーム制御命令コンポーネント(ArmCommand)
　伴-100%
・アーム制御コンポーネント(Control_Crane)
　八木橋-100%
・センサーコンポーネント(RTnoProxy)
　山木-100%
・センサー判定コンポーネント(SignalJudge)
　伴-100%
・スピーカーコンポーネント(PlayBgm)
　山木-100%

<ハード製作物>

・ゴミ箱
　山木-100%

<資料作成>

・チラシ
　長倉-100%
・マニュアル(担当箇所についてはマニュアル目次(p.2-3)を参照)
　長倉-45% /伴 15%/山木-15%/八木橋-25%
・プレゼン(資料・発表)
　伴-100%

5. 参考文献

・色の識別/座標の特定コンポーネント(CameraImgPos)

1.opencv v2.1 documentation cv.画像処理とコンピュータビジョン cv::cvtColor,threshold [http://opencv.jp/opencv-2.1/cpp/miscellaneous_image_transformations.html] ,2018 年 10 月 22 日閲覧

2. OpenCV 画像解析入門　厳密な重心の求め方 2018 年 10 月 22 日閲覧 [http://www.cellstat.net/centroid/],2018 年 10 月 22 日閲覧

・センサーコンポーネント

1.Arduino 赤外線距離センサーで距離測定　https://algorithm.joho.info/arduino/distance-sensor-gp2y0a21/　2018 年 10 月 23 日閲覧