複数ロボットのための協調制御 RTC
ユーザーマニュアル_Ver1.0

2018/10/31

芝浦工業大学大学院

理工学研究科　機械工学専攻

知能機械システム研究室

内藤　佑太，松日楽　信人

内容

1. 概要

1.1 はじめに

近年，人が立ち入るには危険な場所において，遠隔でロボットが作業することが求められている．しかし，遠隔操作には遠隔通信で行うための特別な機材・システム構築及び各ロボットの制御プログラムの調整が必要となる．そこで，ロボットサービス用の共通通信規格である RSNP(Robot Service Network Protocol)[1]とロボットシステムの開発に標準化されたソフトウェアプラットフォームである RT ミドルウェア[2]を用いることで比較的容易に遠隔操作システムの開発が可能である．また，遠隔で作業を行うためには，遠隔地で移動するための移動台車や実際に作業を行うためのロボットアーム，遠隔地の状況を把握するためのカメラなどが必要となり，これらを統合したシステムを一から構築するには，期間及びコストがかかる．しかし，既存のロボットアームや移動台車などの遠隔操作システムを合体させることで，遠隔地の状況に合わせてシステムを組み合わせ臨機応変に対応できると考える．先行研究では，各システムが独立しており，またそれらを合体させることで一つの遠隔操作システムとしても利用可能なクライアント協調型遠隔操作システムの構築を行った[3]．しかし，各ロボット単体では達成できるタスクに限界があるため複数のロボットの協調制御が必要になる．加えて，各ロボットに操作デバイスを用意しては操作者の負担につながる．

そこで，本コンテストでは複数ロボットの遠隔操作において一つの操作装置の操作の仕方の違いから自動で制御するロボットを切り替える協調制御のための RT コンポーネントの開発を実施する．今回は，作業ロボットとして基礎的な移動台車とロボットアームの 2 台に限定して協調制御を行う．

協調制御は以下の方針で行う

・操縦装置の入力値から操作するロボットを決定

・測域センサにより検出した障害物の位置により各ロボットの指令値を補正

1.2 開発・動作環境

PC(OS：64bit 版 Windows10), OpenRTM-aist1.1.2(C++, 32bit 版), Visual C++ 2013, CMake3.5.2, Python2.7.10, PyYAML3.10, doxygen1.8.1 で開発を行った．修正 BSD ライセンスを適用する．

1.3 リンク

芝浦工業大学　知能機械システム研究室　RT ミドルウェア

http://www.sic.shibaura-it.ac.jp/~matsuhir/index.html

1.4　入力装置

図 1 に操作者側(以降マスタ側)で使用する操縦装置を示す. 操縦装置は Novint 社の Falcon, 3 自由度のゲームコントローラーであり，X,Y,Z の 3 軸の位置及び速度の計測やモータのフィードバック機能による反力提示が可能である.

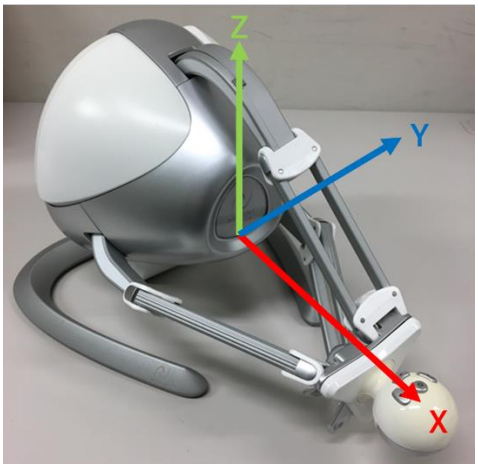

図 1　3 自由度操縦装置

1.5　移動作業ロボット

図 2 に本システムで使用する遠隔地側(以降スレーブ側)の移動作業ロボットを示す．移動作業ロボットは，移動台車がイクシスリサーチ社の iWs09，4 軸のロボットアームが Robotis 社の MikataArm および WEB カメラがあり，以上の 4 つから構成されている．移動台車前方には，北陽電機社の LRF(測域センサ)が搭載されており，障害物回避が可能である.

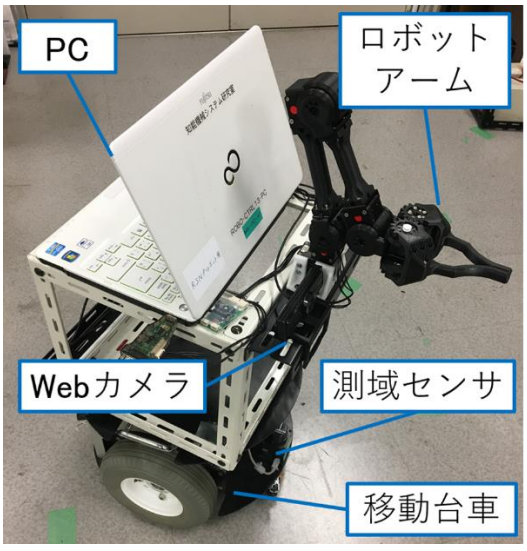

図 2　移動作業ロボット

1.6　クライアント協調型フレキシブル遠隔操作システムの構成

図 3 にクライアント協調型フレキシブル遠隔操作システムの概要を示す．各ロボットの制御プログラムや RSNP のクライアントは RT コンポーネント[3]から構成されている．ピックアンドプレイスなどの作業内容によっては移動台車とロボットアームが一体となって動作させる必要があり，その際は測域センサのデータからロボット周辺の障害物の位置によって開発する協調制御 RTC で操作するロボットを切り替える協調制御を行う．本コンポーネントは，TCP/IP 通信などの無線 LAN 環境下での使用も可能である．

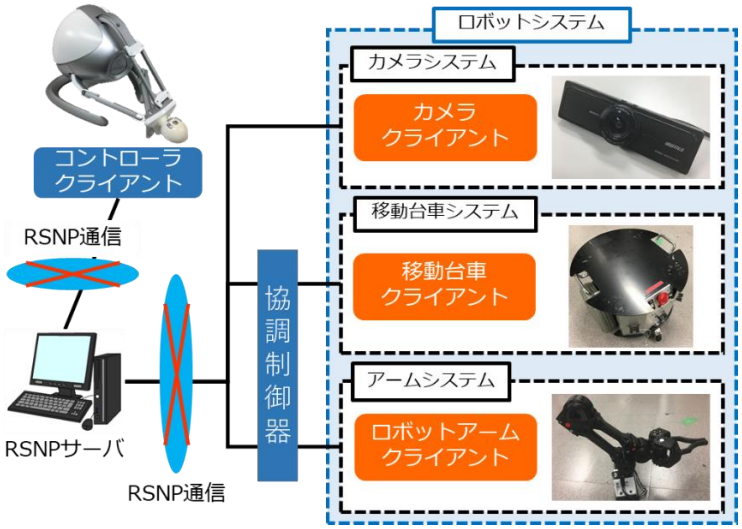

図 3　各クライアント協調時のシステム概要

また図 4 のように各クライアントでロボット単体での遠隔操作システムとして使用することが可能である．組み合わせてロボット一体型の遠隔操作システムとしても扱うこともできる．

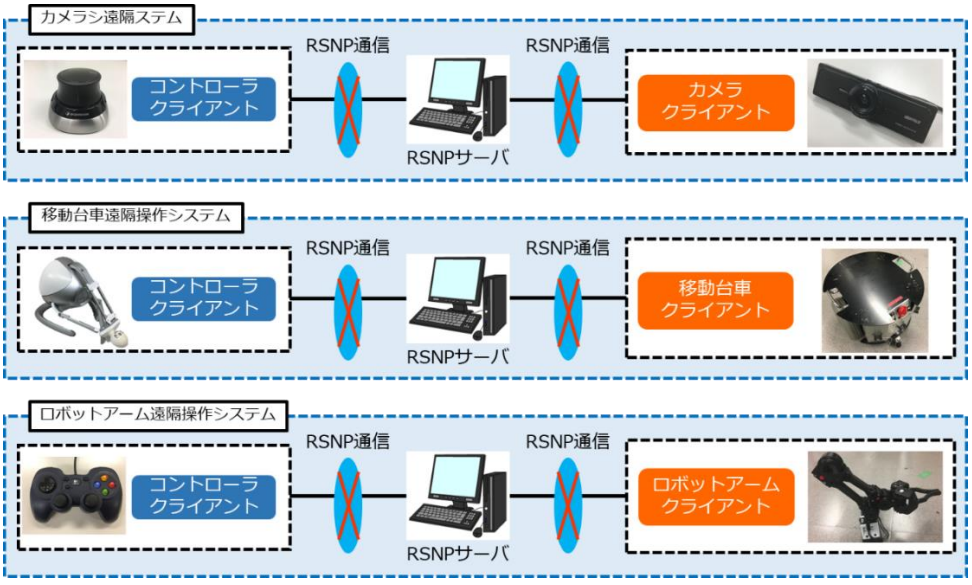

図 4　各クライアント単体でのシステム概要

1.7　RT システム構成

図 5 と図 6 に既存の RTC 群を使用したクライアント協調型フレキシブル遠隔操作システムの構成図を示す．システムは操作者が操作装置を扱うマスタ側と遠隔地でロボットを制御するスレーブ側に分かれている．

マスタ側

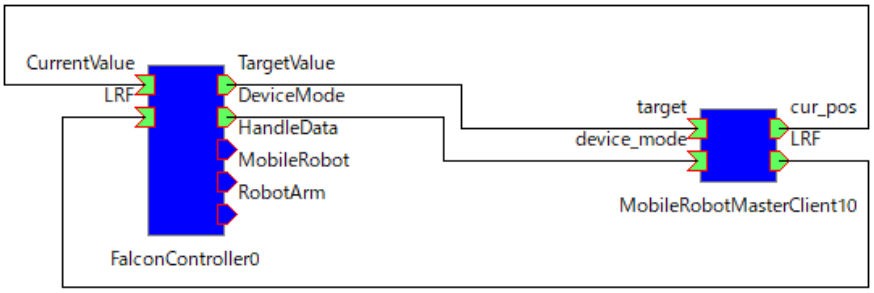

図 5　RT システム構成(マスタ側)

表 1　RTC 概要(マスタ側)

RTC 名称	用途
MobileRobotMasterClient1()	RSPN サーバーとの通信及びデータの送受信用
FalconController()	操作装置入力用

マスタ側は 2 つの RT コンポーネントから構成されている．2 つのコンポーネントは先行研究で開発された RTC を再利用したものである．FalconController で操作装置のグリップ部の 3 自由度の位置(X[m]，Y[m]，Z[m])とボタンの入力値を読み取り，その値からロボットアームと移動台車への指令値を生成する．そして，各ロボットの指令値と操作装置の入力値を MobileRobotMasterClient から RSNP サーバーに送信する．

スレーブ側

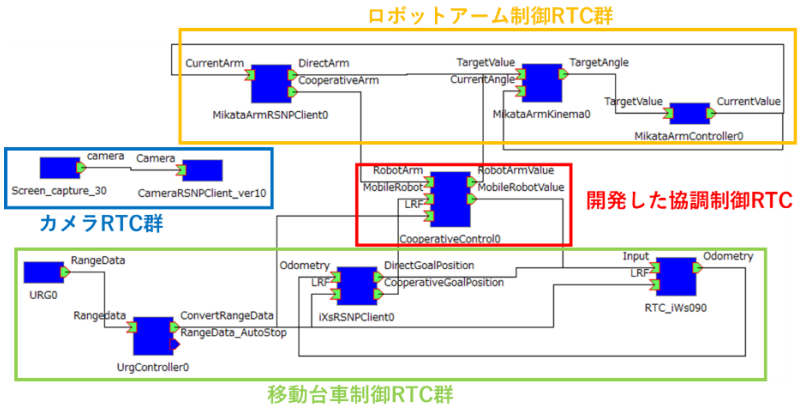

図 6　RT システム概要(スレーブ側)

表 2　RTC 概要(スレーブ側)

RTC 名称	用途
URG()	測域センサからのデータ取得
UrgController()	測域センサからのデータ補正用
iXsRSNPClient()	RSNP サーバーとの通信及びデータの送受信用
RTC_iWs09()	移動台車の制御用
MikataArmRSNPClient()	RSNP サーバーとの通信及びデータの送受信用
MikataArmKinema()	ロボットアームの指令値生成用
MikataArmController()	ロボットアームの制御用
CameraRSNPClient()	RSNP サーバーとの通信及びデータの送受信用
Screen_capture()	画像取得用
CooperativeController()	協調制御用

スレーブ側は 10 個の RT コンポーネントから構成されている．CooperativeController 以外は先行研究で開発された RTC を再利用したものである．主にロボットアームを制御するロボットアーム制御 RTC 群，移動台車を制御する移動台車制御 RTC 群，カメラ画像を取得するカメラ RTC 群からなっている．各 RSNP クライアント RT コンポーネントで RSNP サーバーとの通信やマスタ側とのデータの送受信を行う．ロボットアーム制御 RTC 群では，MikataArmRSNPClient が RSNP サーバーからロボットアームの指令値(手先の位置 X[m], Y[m], Z[m], Pitch[rad])を受け取り，MikataArmKinema を通して各関節の目標角度を算出し，MikataArmController でロボットアームを制御する．移動台車制御 RTC 群では，iXsRSNPClient が RSNP サーバーから移動台車の指令値(直進速度[m/s]，旋回速度[rad/s])を受け取り，RTC_iWs09 で移動台車を制御する．カメラ制御 RTC 群では，Screen_capture で取得した画

像を CameraRSNPClient で RSNP サーバーに送り Web ブラウザ上に表示する．

2.　複数ロボットのための協調制御 RTC

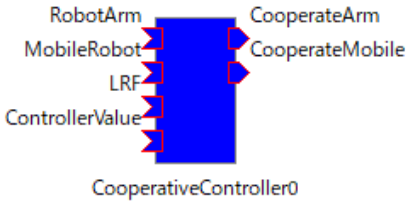

図 7　協調制御 RTC

開発した本 RTC は，操作装置の入力値からロボットアームか移動台車どちらの制御を行うかを決定し，測域センサのデータから障害物の検出した位置によってロボットアームと移動台車の指令値に補正を行う．図 8 に開発した協調制御 RTC の接続イメージを示す．InPort には，ロボットアーム及び移動台車への指令値，そ操作装置の入力値，測域センサのデータを送信するポートを接続する．本稿で使用するロボットアームは 4 自由度，操作装置は 3 自由度の操作装置であるが，Configuration の値より操作装置とロボットアームの自由度は変更可能である．操縦装置の自由度は最大で 3 自由度，ロボットアームの自由度は最大でハンド含めて 7 自由度である．操縦装置は本稿で使用している 3 自由度の装置だけでなく，2 自由度であるジョイスティックなどの操縦装置でも使用可能である．OutPort には，ロボットアームと移動台車を制御する RT コンポーネントで指令値を受信するポートを接続して使用する．表 3，4，5 に InPort，OutPort，Configuration の概要を示す．

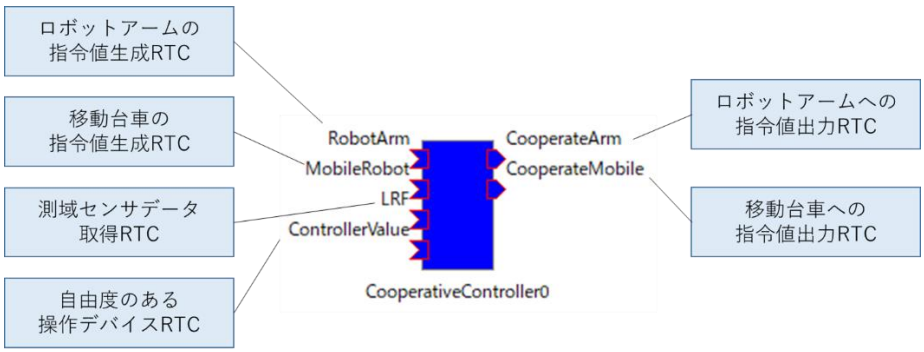

図 8　協調制御 RTC の接続イメージ

表 3　協調制御 RTC 概要(InPort)

InPort		
名称	データ型	機能
		データの例
RobotArm	TimedDoubleSeq	ロボットアームの目標手先位置[m]及び目標姿勢[rad]，ハンドの目標開閉角度[rad]を取得．自由度はハンドを含めて最大で 7.
		RobotArm.data[0] = 手先目標位置 X RobotArm.data[1] = 手先目標位置 Y RobotArm.data[2] = 手先目標位置 Z RobotArm.data[3] = 手先目標姿勢 Roll RobotArm.data[4] = 手先目標姿勢 Pitch RobotArm.data[5] = 手先目標姿勢 Yaw RobotArm.data[6] = ハンドの開閉角度
MobileRobot	TimedVelocity2D	移動台車への直進速度[m/s]及び旋回角速度[rad/s]を取得．TimedVelocity2D.data.vy は使用しない.
		TimedVelocity2D.data.vx = 直進速度 TimedVelocity2D.data.va = 旋回角速度
LRF	RangeData	測域センサのレーザーn 本の各長さ[mm]を取得
		LRF.ranges.data[0] = レーザーの長さ 〜 LRF.ranges.data[n] = レーザーの長さ
ControlleValue	TimedDoubleSeq	操作装置の入力値を取得．操縦装置の自由度は最大で 3 自由
		ControlleValue.data[0] = X 軸方向の入力[m] ControlleValue.data[1] = Y 軸方向の入力[m] ControlleValue.data[2] = Z 軸方向の入力[m]

表 4　協調制御 RTC 概要(OutPort)

OutPort		
名称	データ型	機能
CooperativeArm	TimedDoubleSeq	補正したロボットアームの手先の目標位置[m] 及び姿勢[rad]，ハンドの開閉角度[rad]を出力
		CooperativeArm.data[0] = 手先補正位置 X CooperativeArm.data[1] = 手先補正位置 Y CooperativeArm.data[2] = 手先補正位置 Z CooperativeArm.data[3] = 手先補正姿勢 Roll CooperativeArm.data[4] = 手先補正姿勢 Pitch CooperativeArm.data[5] = 手先補正姿勢 Yaw CooperativeArm.data[6] = ハンドの補正開閉角度
CooperativeMobile	TimedVelocity2D	補正した移動台車への直進速度[m/s]及び 旋回角度[rad]を出力
		TimedVelocity2D.data.vx = 直進速度 TimedVelocity2D.data.va = 旋回角速度

表 5　協調制御 RTC 概要(Configuration)

Configulation		
名称	データ型	機能
inputDOF	int	操作装置の自由度
armDOF	int	ロボットアームの自由度
offset	double	制御ロボット切り替え時の入力値のオフセット

ピックアンドプレイスなどの位置決め精度が求められる作業とそうでない作業では，操作者の各ロボットの操作量に違いが発生する．移動台車の場合，目標位置周辺への移動のみで高い精度を求められることは少ないが，ロボットアームの場合は実際に作業を行う場面が多く，物体の把持など高い精度が求められる．そのため実際に作業を行う際，ロボットアームと移動台車の操作では，慎重に動作させる必要があるロボットアームの方が移動台車に比べて操作の変化量が小さくなると考えられる．図 9 に熟練者がロボットアームと移動台車を操作した際の，操作装置のグリップ部の位置の変化つまり速度を示す．図からも変化の傾きに差があることが分かる．

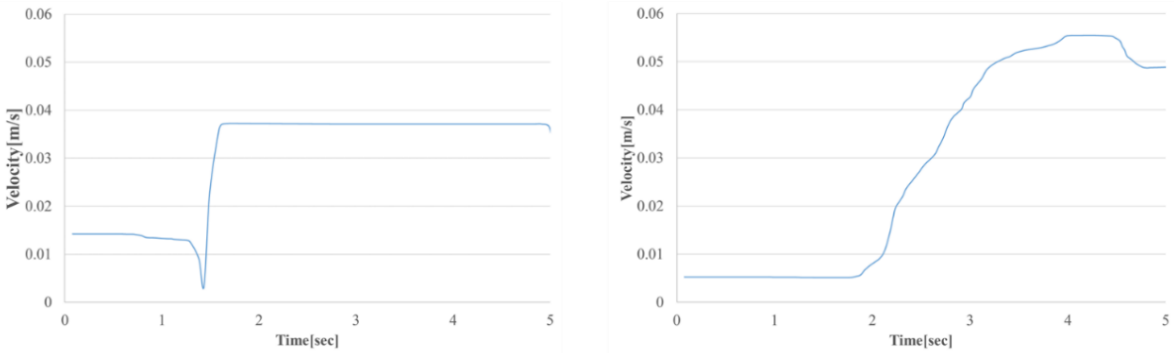

図 9　移動台車操作時(左)とロボットアームの操作時(右)の操作装置の値

そこで，本 RTC では操作装置より受信したハンドル部の 3 軸の位置(X[m]，Y[m]，Z[m])の変化量，つまり速度を求め，ローパスフィルタとして移動平均を行った速度から最小二乗法で逐次傾きを求める．そして，評価関数として利用するシグモイド関数に代入して得た値から制御するロボットを決定する．以下に示すシグモイド関数は a>0 に対して a→∞のときステップ関数に近づく．この性質を利用することで操作装置の入力から，操作量の変化の小さいロボットアームとそうでない移動台車を判別し制御するロボットを決定する．

$$y = \frac{1}{1 + e^{-ax}}$$

更に，移動台車に搭載されている測域センサにより前方 180 度範囲で障害物を検出し，検出した障害物の位置によって移動台車とロボットアームの制御に補正を行う．図 10 に示す②の範囲に障害物がある場合は，ロボットアームの手先位置の 2 軸(X，Z)のみの動作と移動台車の旋回のみを行い，①の範囲に障害物がある場合はロボットアームのみの制御に切り替わる．①と②の範囲に障害物を検出しない場合は，移動台車のみの制御を行う．以上の制御により一つの操縦装置で移動台車とロボットアームの協調制御を行い，移動台車が障害物に接触することなく操作することが可能である．なお，①と②はロボットアームの動作範囲から決定したものである．

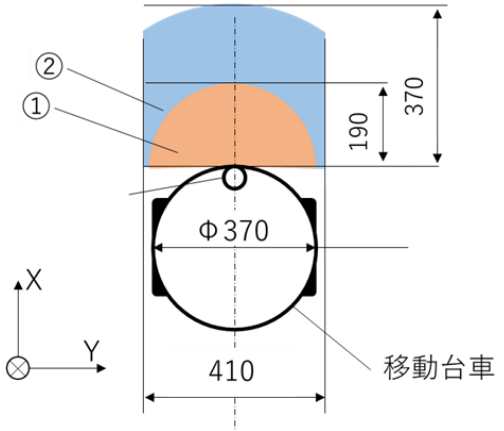

図 10　測域センサの障害物検出範囲

表 6　測域センサによる補正内容

障害物の位置	指令値の補正内容
①	移動台車の指令値を全て 0 に補正
②	移動台車の直進成分とロボットアームの Y 軸方向の指令値を 0 に補正
①と②以外	ロボットアームの指令値を全て 0 に補正

3　利用方法

3.1　用意するハードウェア

以下に本稿で使用するハードウェア一覧を示す.

表 7　用意するハードウェア

名称	数量
操作装置	1
移動作業ロボット	1
Web カメラ	1
測域センサ	1
ノート PC	2
サーバーPC	1
モバイルルーター	1
バッテリー	1

3.2　動作環境

以下に本稿で使用する遠隔操作システムの動作環境を示す.

サーバー

・サーバーPC:OS Windows7

マスタ側クライアント

・クライアント PC:OpenRTM-aist1.1.2 がインストールされており使用できる OS

・実行環境:www ブラウザ Google chrome(HTML5 対応)

スレーブ側クライアント

・クライアント PC:OpenRTM-aist1.1.2 がインストールされており使用できる OS

3.3　利用手順

①PC の起動

マスタ側とスレーブ側，RSNP サーバーに使用する PC の電源を入れる

②ロボットの電源

移動台車の電源を入れ，Web カメラ，測域センサ，ロボットアームの各 USB を制御用 PC に接続する．

③RSNP サーバーの起動

Eclipse を起動して RSNP サーバーのウィンドウから使用するサーバーの名前を選択し，「サーバーを起動」ボタンをクリックする．

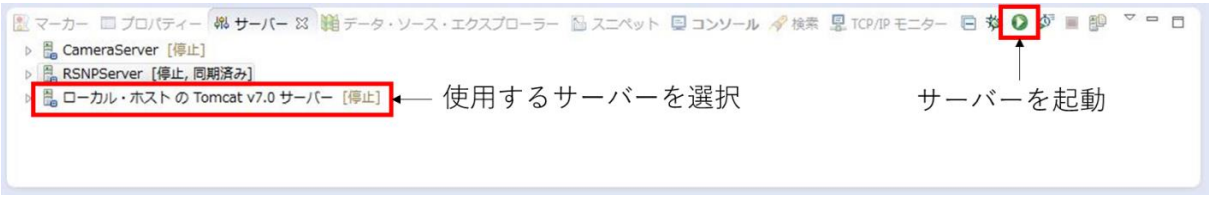

図 11　サーバーウィンドウ

④クライアントの準備

マスタ側とスレーブ側の PC で StartNamingService を起動する．そして，Eclipse を起動してネームサーバーを localhost に接続する．マスタ側の PC では以下の RT コンポーネントを起動する．

・MobileRobotMasterClient　→　MobileRootMasterClient.bat
・FalconController　→　FalconController.exe

スレーブ側の PC では開発した協調制御 RT コンポーネントを含む以下の RT コンポーネントを起動する．

・URG　→　urgcomp.exe
・UrgController　→　UrgControllerComp.exe
・iXsRSNPClient　→　iXsRSNPClient.bat
・RTC_iWs09　→　RTC_iWs09.exe
・MikataArmRSNPClient　→　MikataArmRSNPClient.bat
・MikataArmKinema　→　MikataArmKinemaComp.exe
・MikataArmController　→　mikataarmcontrollercomp.exe
・CameraRSNPClient　→　CameraRSNPClient.bat
・Screen_capture　→　Screen_capture.bat
・CooperativeController　→　cooperativecontroller.exe(今回開発した協調制御 RTC)

⑤SystemEditor を完成させる

③で起動した RT コンポーネントを System Diagram 上にドラッグ＆ドロップし，図 5 と図 6 のように各 RT コンポーネントを接続する．

⑥コンフィギュレーション値の設定

開発した CooperativeController の inputDOF と armDOF は，操作装置とロボットアーム(ハンドも含む)の自由度の設定である．本システムでは，３自由度の操作装置に 5 自由度のロボットアーム(ハンドの開閉も含む自由度)を使用するが，接続する操作装置とロボットアームに合わせて変更する．また，各 RSNP クライアントのコンフィギュレーション値は RSNP サーバーとなる PC に合わせて変更する．EndPoint は以下のようになっている．[IP]にはサーバーとなる PC の IP アドレスを入力し，[Port]にはサーバー設定時に定めた Port を入力する．

http:// [IP][Port]/Falcon_master_slave/services

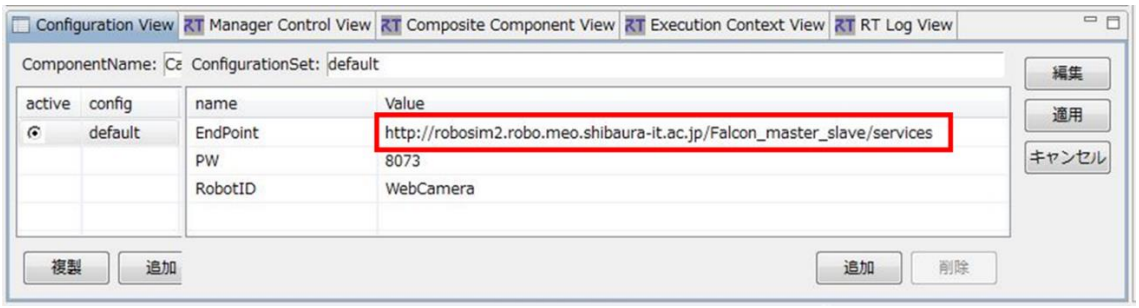

図 12　コンフィギュレーション値設定画面

⑦システムの起動

マスタ側とスレーブ側の RT コンポーネントを Activate することで，クライアントがサーバーに接続され遠隔操作が可能となる．

3.4　ビルドの手順

①ファイルのダウンロード

下記 URL より協調制御 RTC のフォルダをダウンロードし, 任意のディレクトリに配置する.

https://github.com/YutaNaito/CooperativeController

②CMake

CMake を開き，ダウンロードしたファルダから「CMakeLists.txt」を CMake のウィンドウにドラッグ＆ドロップする. そして,「Where to build the binaries」のディレクトリに「/build」を加える.「Configure」,「Generate」の順にそれぞれ成功したらクリックして実行していく．開発環境の選択では Visual Studio 12 2013 を指定する.

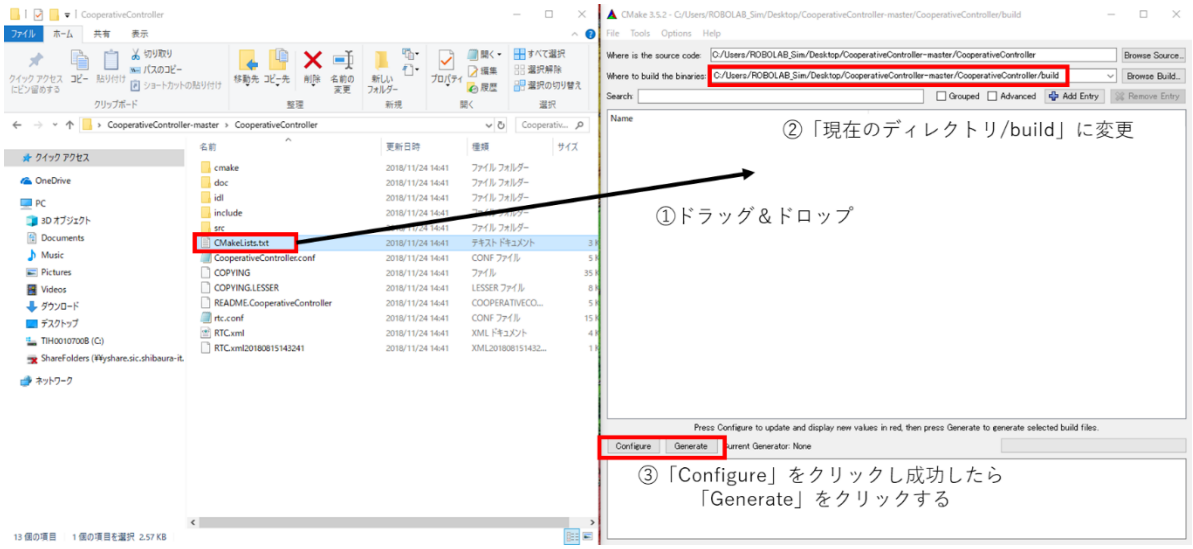

図 13　CMake の手順

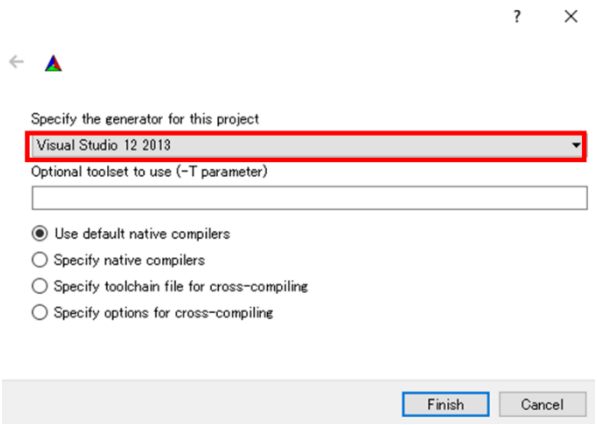

図 14　開発環境の選択

③ソリューションのビルド

②で作成された「build」直下の CooperativeController.sln を開き，「ビルド」から「ソリューションのビルド」をクリックすることで実行ファイルである「cooperativecontrollercomp.exe」が作成される．

4　使用ソフトウェア入手方法

・OpenRTM-aist1.1.2
http://openrtm.org/

・RSNP ライブラリ v2.3（RSi）
・rsnp-robot-api-2.3.0_r49.jar
・rsnp-robot-fjlib-2.3.0_r49.jar
http://robotservices.org/index.php/aboutrsnp/library/
（上記サイトを参考に書面またはメールで申し込みが必要）

・ Apache Tomcat v7.0.55
http://tomcat.apache.org/

・ Axis2 v1.4.1
http://axis.apache.org/axis2/java/core/

・ Java SE Development Kit v1.7.0_65 (JDK)
http://www.oracle.com/technetwork/opensource/index.html

・ Eclipse IDE for Java EE Developers v4.3
http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/junosr1

・ apache-mime4j-0.6.jar
http://james.apache.org/mime4j/

・commons-codec-1.3.jar
http://commons.apache.org/proper/commons-codec/

・ commons-logging-1.1.1.jar
http://commons.apache.org/proper/commons-logging/

・ httpclient-4.0.jar, httpcore-4.0.1.jar, httpmime-4.0.jar
http://hc.apache.org/downloads.cgi

・ jquery-2.1.1.min.js
http://jquery.com/download/

・ OpenRTM-aist-1.1.2.jar
http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website

5 参考文献

[1] Robot Service Network Protocol2.2 仕様書 2010

[2] OpenRTM-aist Web ページ
「http://openrtm.org/」
最終閲覧日 2018 年 8 月 25 日

[3] 内藤佑太, 松日楽信人 : ”RSNP クライアント協調型フレキシブル遠隔操作システム”, 第 36 回日本ロボット学会学術講演会 2018, 1H2-02, 2018.