

2. 概要

2.1 RT ミドルウェアとは？

RT ミドルウェアとは，Robot Technology Middleware の略称で，ロボットシステムを構築するためのソフトウェアプラットフォーム（ソフトウェアの土台となる部分．OS のようなもの）です．

RT ミドルウェア上では RTC (Robot Technology Component の略称) を使用することができます．RTC とは，ロボットに関わる機能をモジュール化したソフトウェアであり，モータを動かす RTC，カメラを制御する RTC，ロボットのナビゲーションができる RTC などたくさんの種類があります．既存の RTC は再利用することは容易であり，また，RT ミドルウェア上のロボットシステムに RTC を追加したり除去したりすることも容易であるため，拡張性の向上が実現できます．

よって，RT ミドルウェア上で RTC を使用することで，ロボットシステムを低コストかつ効率的に構築できるようになります．

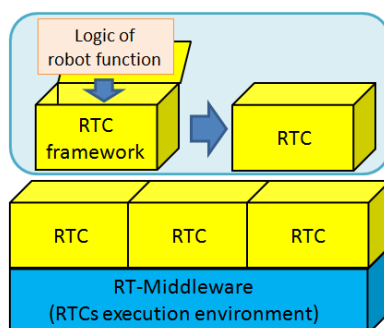


図 1 RT ミドルウェアと RTC の説明

2.2 RTC とは？

RTC とは前述のとおり，ロボットに関わる機能をモジュール化したソフトウェアです．

RTC は型, 名称, データ長などが定められた入力ポート (In Port) と出力ポート (Out Port) を設定することができ，図のように入力ポートと出力ポートを接続することで RTC 同士の通信を行います．他に「サービスポート」というポートもありますが，本学習環境で用いていないため解説を省略します．

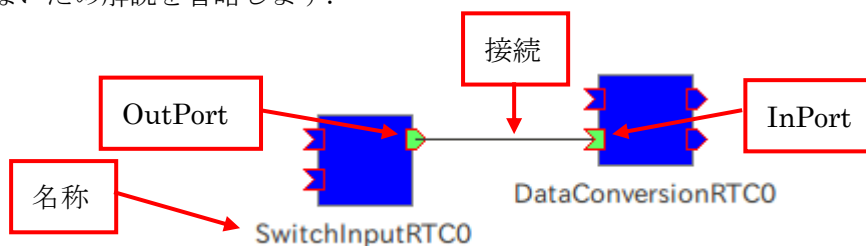


図 2 RTC の接続の説明

また、「**コンフィグレーション**」と呼ばれる、パラメータを動的に変更できる機能も有しています。一度出来上がった RTC でもコンフィグレーションによりパラメータ変更することで、自分の環境に応じた RTC をある程度作成することが可能となります。このことにより、RTC の再利用性や拡張性がより向上しています。

2.3 OpenRTM-aist とは？

OpenRTM-aist とは、**RT ミドルウェアの実装のひとつ**です。

(OpenRTM-aist の公式 Web サイト：<http://www.openrtm.org/openrtm/ja/>)

特長として以下があげられます。

- ① フリーウェア
- ② ネットワーク透過性、OS 非依存性、言語非依存性が重視されているため、CORBA を用いて実装
- ③ RT System Editor や RTC Builder など各種ツール群により効率的に開発可能であり、GUI のツールも多く直感的に使用可能
- ④ **動力学シミュレータ OpenHRP3** を利用することが可能

RT ミドルウェアや RTC, OpenRTM-aist の説明に関しては、OpenRTM-aist の公式 Web サイト左のナビゲーション内にある「ドキュメント」にて解説されております。より詳しく知りたい方は、そちらをご覧ください。

2.4 OpenHRP3 とは？

OpenHRP3 は、コントローラを RTC として開発することができる動力学シミュレータです。OpenHRP3 の利用には、VRML ファイル (**.wrl) を使用して表したモデルと RTC として開発したコントローラを用意する必要があります。

(OpenHRP3 の公式 Web サイト：<http://www.openrtp.jp/openhrp3/jp/index.html>)

OpenHRP3 を用いることで、シミュレーション内のロボットの関節角度や関節トルク、ワールド座標系における位置姿勢、距離データ、カメラ画像などを取得することができるようになります。また、それを出力し他の RTC に利用することができます。

OpenHRP3 は、

<http://www.openrtp.jp/openhrp3/jp/install.html>

にて動作確認が行われております。OS さえあれば、このページのインストール手順に従うことで、無料でシミュレーション環境や RT ミドルウェア開発環境を構築することができます。

現在は Ver.3.1.1 で、RT ミドルウェアは OpenRTM-aist-1.0.0-RELEASE を使用しています。

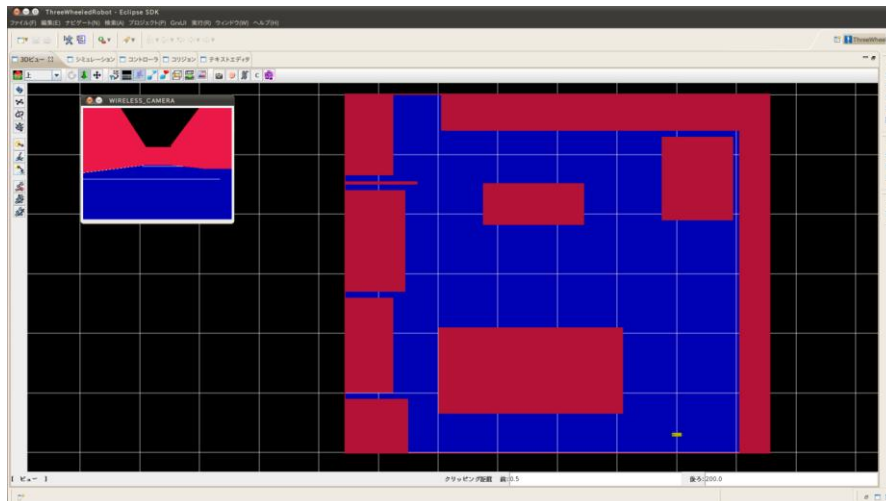


図3 OpenHRP3 実行中の例

(左：ロボットモデル前方のカメラ画像，右：シミュレーション環境)

2.5 RT ミドルウェア学習環境の概要

RT ミドルウェア学習環境とは、安価かつ入手容易で、多人数向けかつ初心者向けの、RT ミドルウェアをより実践的に学習するための環境です。また、学習環境構築のためにリファレンスロボット上で動作する RTC 群を再利用することで、該当 RTC 群の有用性を示すことを目指しました。

そのために、RT ミドルウェア学習環境は、以下の3つを目標にして作られました。

- ① 安価な移動ロボットを開発し使用すること
- ② 移動機能がリファレンスロボット（高機能なロボット）と同等であること
- ③ 配布や入手が容易であること

そのために、安価で入手容易な3輪移動ロボットを開発し、リファレンスロボットで使われている「来訪者受付システム」という RTC 群の移動機能に関する部分を再利用することで、RT ミドルウェア学習環境を作成しました。図2が再利用した「来訪者受付システム」の移動機能に関する RTC 群を用いた RT ミドルウェア実行環境であり、図4が今回作成した RT ミドルウェア学習環境です。

(来訪者受付システム：http://210.154.184.16/pukiwiki/?SYS_001)

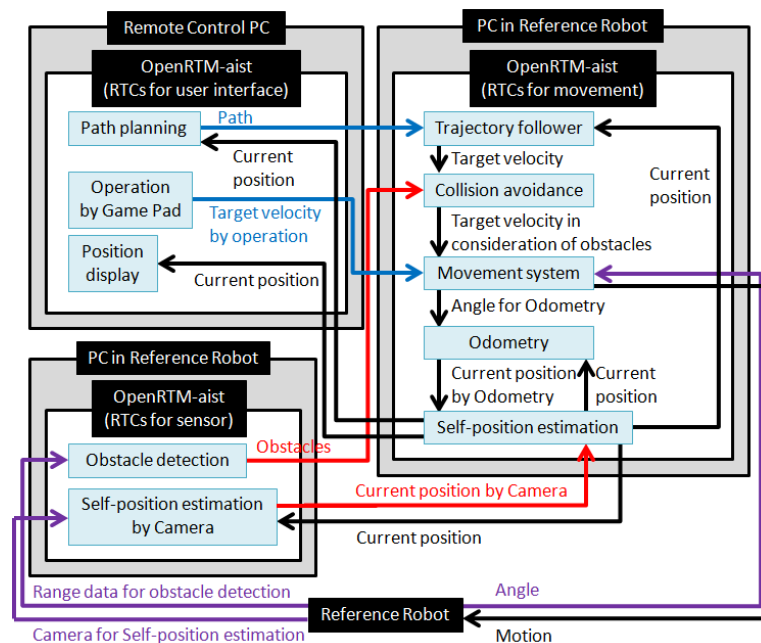


図4 来訪者受付システムの移動機能に関する RTC 群を用いた RT ミドルウェア実行環境

ユーザインタフェースに関する RTC からの出力を青色，センサ類に関する RTC からの出力を赤色，移動に関する RTC のからの出力を黒色，リファレンスロボットからの出力を紫色で表示しました。

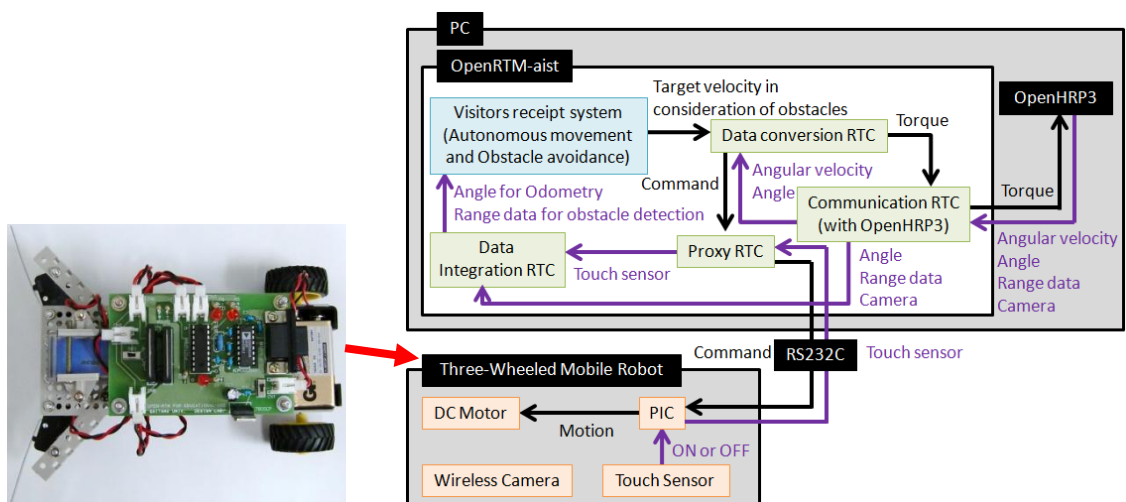


図5 RT ミドルウェア学習環境

図5にある来訪者受付システム（自律移動と障害物回避）とは，図4に示す RT ミドルウェア実行環境を再利用したものです．図5左は，開発した3輪移動ロボットです．3輪移動ロボットや OpenHRP3 から取得している情報は紫色で表示しました。

図4においてリファレンスロボットから取得していた情報の中には，測域センサからの

距離データやエンコーダ付きモータから算出したロボット後輪の回転角度など 3 輪移動ロボットでは取得できない情報もあります。

そこで、図 5 にもあるように動力学シミュレータ **OpenHRP3** を用いて、シミュレーションにより距離データや車輪回転角度などを取得できるようにします。

この学習環境により、3 輪移動ロボットは自律移動と障害物回避ができるようになります。

2.6 再利用する RTC

2.6.1 来訪者受付システムの移動機能に関する RTC 群

来訪者受付システムの移動機能に関する RTC 群は、自律移動と障害物回避、また、操作により移動させることができます。この内、本学習環境では、自律移動と障害物回避の部分を再利用しています。

来訪者受付システムは、

http://210.154.184.16/pukiwiki/?SYS_001

にて公開されております。本学習環境では Ver 1.0 を用いているため、

http://210.154.184.16/pukiwiki/?SYS_001_V100

のソース（RTC 群など）やドキュメント、参考資料などを使用しています。（RTC 群の実際の使用方法については 3.3 章で詳しく解説します）

上記 URL で公開されている来訪者受付システム内、本学習環境では移動機能に関する RTC 群を用います。その中で、本学習環境で使用している 12 個の RTC を紹介します。RTC の詳細は上記 URL にて公開されているドキュメント、参考資料を参照してください。

- (1) PositionInput : 目的地入力
- (2) PathPlanning : 経路計画
- (3) Navigation : 経路走行
- (4) PathFollower : 軌跡追従
- (5) Urg_to_Obstacles : 障害物検知
- (6) ObstacleMonitor : 障害物モニタ
- (7) CollisionDetection : 衝突判定
- (8) ObstacleAvoidance : 回避行動
- (9) SwitchInputRTC : 自律と操作の入れ替え（本学習環境では操作による移動は未実装）
- (10) Odometry : オドメトリ推定
- (11) LocalizeCenter : 自己位置姿勢推定
- (12) DispPosition : 位置表示

なお、(5)～(8)の障害物回避に関する RTC 群は現在のバージョンでは、正常に障害物回避を行うことができません。よって、パラメータなどの設定によっては使用しても支障が

ない場合もありますが、基本的には使用しないことを推奨します。

2.6.2 Proxy RTC

ProxyRTC は、RTC 群と 3 輪移動ロボットとで RS232C 通信を可能とする RTC です。

ProxyRTC は、「ysuga.net」

<http://ysuga.net/>

にて公開されている「5.1COM ポートにアクセスする RTC 作成」

http://ysuga.net/robot/rtm/practice/com_port

を参考に作成しました。

本学習環境では、上記 URL の SerialPortRTC に多少の修正を加えたものを、ProxyRTC と呼んでいます。SerialPortRTC を修正して ProxyRTC を作成する方法については、3.3.1 章で詳しく解説します。

ProxyRTC の開発環境を表 1、仕様を表 2-A~D に示します。

表 1 ProxyRTC の開発環境

OS	Ubuntu 10.04 LTS
RT ミドルウェア	OpenRTM-aist-1.0.0-RELEASE
開発言語	C++
コンパイラ	gcc 4.4.3

表 2-A ProxyRTC の動作条件

実行周期	100.0[Hz]
------	-----------

表 2-B ProxyRTC のデータポート (InPort)

名称	型	データ長	説明
in	RTC::TimedOctetSeq	4	3 輪移動ロボットへの指令値 (RS232C 通信で送るデータ)

表 2-C ProxyRTC のデータポート (OutPort)

名称	型	データ長	説明
out	RTC::TimedOctetSeq	12	3 輪移動ロボットから受け取った タッチセンサの ON/OFF のデータ (RS232C 通信で受け取るデータ)

表 2-D ProxyRTC のコンフィグレーション

名称	型	デフォルト値	説明
debug	int	debug	未使用
filename	std::string	/dev/tty/USB0	例 : "/dev/ttyUSB0", "COM0"
baudrate	int	9600	例 : 9600, 115200

2.7 自作する RTC

2.7.1 Data Conversion RTC

DataConversionRTC は、入力された目標並進速度と目標旋回速度を、3 輪移動ロボットへ渡す動作指令値と、**OpenHRP3** のロボットモデルへ渡すトルクに変換する RTC です。

DataConversionRTC は、来訪者受付システムの移動機能に関する RTC 群の中のひとつであるシミュレーション用の RTC 「MotorControl」の、目標速度から **OpenHRP3** のロボットモデルに与えるトルクを算出するアルゴリズムを再利用しています。

MotorControl とは、来訪者受付システム Ver 1.0

http://210.154.184.16/pukiwiki/?SYS_001_V100

にある、Download の【ソース】の RH 関連（ファイル名 RH_10.zip）

をダウンロードし、展開した中の、

RH/RemotePC/src/comp/ForSimulation/MotorControl

のことです。

また、移動ロボット用コンポーネント群である **OpenINVENT** にて **MotorControl** の説明がされている word ファイルなどがダウンロードできますので、参考にしてください。

（ **OpenINVENT** の WEB ページ : <http://www.openrtp.jp/INVENT/> ）

DataConversionRTC の開発環境を表 3、仕様を表 4-A~D に示します。

表 3 DataConversionRTC の開発環境

OS	Ubuntu 10.04 LTS
RT ミドルウェア	OpenRTM-aist-1.0.0-RELEASE
開発言語	C++
コンパイラ	gcc 4.4.3

表 4-A DataConversionRTC の動作条件

実行周期	1000000[Hz]
備考	CommunicationRTC(withOpenHRP3) に RTC としての処理を委ねている (詳細は 2.7.2 章にて説明)

表 4 - B DataConversionRTC のデータポート (InPort)

名称	型	データ長	説明
model_angle	RTC::TimedDoubleSeq	2	ロボットモデルの 左右後輪の回転角度[rad] (タイムスタンプも使用します)
target_velocity	IIS::TimedVelocity2D	1	目標並進速度[m/s]と 目標旋回速度[rad/s]

表 4 - C DataConversionRTC のデータポート (OutPort)

名称	型	データ長	説明
model_torque	RTC::TimedDoubleSeq	2	ロボットモデルの左右後輪に 与えるトルク
robot_command	RTC::TimedOctetSeq	4	3 輪移動ロボットへの指令値

表 4 - D DataConversionRTC のコンフィグレーション

名称	型	デフォルト値	説明
PGainL	double	5.0	左車輪モータの P ゲイン
DGainL	double	2.0	左車輪モータの D ゲイン
PGainR	double	5.0	右車輪モータの P ゲイン
DGainR	double	2.0	右車輪モータの D ゲイン
leftWheelID	int	1	左車輪の ID 番号
rightWheelID	int	0	右車輪の ID 番号
radiusOfLeftWheel	double	0.018	左車輪の半径
radiusOfRightWheel	double	0.018	右車輪の半径
lengthOfAxle	double	0.078	左右車輪間の長さ
radiusOfBodyArea	double	0.2	ロボット全体を円で 囲んだ場合の半径 安全を考慮したエリア定義用
torqueSensitivity	double	4.6	ロボットモデルへ与えるトルク の伝わりやすさ (実験値). 大きいほど伝わりやすくなる.
torqueConstant	double	2.0	ロボットモデルへ与えるトルク を定数倍する値 (実験値). 大きいほどロボットモデルへ 与えるトルクが大きくなる

【注意点①】

DataConversionRTC の In Port にある IIS::TimedVelocity2D という型は来訪者受付システムにて使われている型で, intellirobot.idl 内で次のように定義されています.

```
struct TimedVelocity2D {  
    RTC::Time tm;  
    sequence<long> id;  
    RTC::Velocity2D data;  
    sequence<double> error;  
};
```

DataConversionRTC では,

m_target_velocity.data.vx : 目標並進速度[m/s]

m_target_velocity.data.vy : 未使用

m_target_velocity.data.va : 目標旋回速度[rad/s]

として扱っております.

詳しい IIS::TimedVelocity2D の仕様は, 3.4.1 章でダウンロードする RH 関連参考資料.zip 内にある, 移動 SWG 共通 IF 案 (移動 SWG 共通 IF 案 101008.pdf) をご覧ください.

【注意点②】

コンフィグレーションの torqueSensitivity と torqueConstant は, OpenHRP3 のロボットモデルと現実の移動ロボットの動きが一致するよう調整した値です. 再利用元である MotorControl のアルゴリズムでは, このコンフィグレーションは存在しません. つまり, 両方のコンフィグレーションとも 1.0 であることが望ましいです. もし実機とシミュレーションのロボットモデルの動きが一致しなければ, この値を, 実験を重ね調整してください.

【注意点③】

実機をより速く動作させたい場合は, DataConversionRTC.cpp 内の 275 行目~321 行目, つまり, 「実機の動作設定 開始」から「実機の動作設定 終了」までを修正し再コンパイルしてください. 指令値に関しては, 2.8.2 章で詳しく解説します.

2.7.2 Communication RTC (with OpenHRP3)

CommunicationRTC (withOpenHRP3) は, OpenHRP3 のロボットモデルと通信するための RTC です. 入力される情報は OpenHRP3 のロボットモデルの左右後輪に与えるトルクです. 出力する情報は, ロボットモデルの車輪回転角度, シミュレーション上の測域センサによる距離データ, ロボットの前方のカメラ画像です.

CommunicationRTC (withOpenHRP3) は, DataConversionRTC のディレクトリ内に置くスクリプトファイルを起動することで, DataConversionRTC とともに実行されます.

このスクリプトにより、DataConversionRTC としての処理を OpenHRP3 側の ControllerBridge に委ねることができます。DataConversionRTC など CommunicationRTC (withOpenHRP3) と接続しているすべての RTC が、OpenHRP3 でのシミュレーションのスタート・ストップに連動して実行されるようになります。実際にスクリプトを作成する方法については、3.3.3 章で詳しく解説します。

スクリプトの詳細については、OpenHRP3 の公式 Web サイトにある解説ページを参考にしてください。

(スクリプト解説ページ: http://www.openrtp.jp/openhrp3/jp/controller_bridge.html)

CommunicationRTC (withOpenHRP3) の開発環境を表 5、仕様を表 6-A~D に示します。

表 5 CommunicationRTC (withOpenHRP3) の開発環境

OS	Ubuntu 10.04 LTS
RT ミドルウェア	OpenRTM-aist-1.0.0-RELEASE

表 6-A CommunicationRTC (withOpenHRP3) の動作条件

実行周期	1000000[Hz]
実行コンテキスト	SynchExtTriggerEC
備考	OpenHRP3 の ControllerBridge に処理を委ねている

表 6-B CommunicationRTC (withOpenHRP3) のデータポート (InPort)

名称	型	データ長	説明
torque	RTC::TimedDoubleSeq	2	ロボットモデルの左右後輪に与えるトルク

表 6-C CommunicationRTC (withOpenHRP3) のデータポート (OutPort)

名称	型	データ長	説明
angle	RTC::TimedDoubleSeq	2	ロボットモデルの左右後輪の回転角度[rad]
image	RTC::TimedLongSeq	ピクセル数	ロボットモデル前方のカメラのカラー画像
range_sensor	RTC::TimedDoubleSeq	センサ出力数	測域センサから得られた距離データ[m]

表 6-D CommunicationRTC (withOpenHRP3) のコンフィグレーション

名称	型	デフォルト値	説明
—	—	—	コンフィグレーションなし

2.7.3 Data Integration RTC

DataIntegrationRTC は、**CommunicationRTC (withOpenHRP3)** と **ProxyRTC** の出力を統合して、データを来訪者受付システムの移動機能に関する **RTC** 群へ渡す **RTC** です。

入力される情報は、**CommunicationRTC (withOpenHRP3)** から出力された車輪回転角度、前方カメラのカラー画像、シミュレーション上の測域センサによる距離データと、**ProxyRTC** から出力されたタッチセンサの ON/OFF のデータです。出力する情報は、データの意味（角度、カラー画像、距離データ、タッチセンサ ON/OFF）を変えずに、それらのデータに単位変換などの多少の修正を施したものとなっております。

DataIntegrationRTC の開発環境を表 7、仕様を表 8－A～D に示します。

表 7 **DataIntegrationRTC** の開発環境

OS	Ubuntu 10.04 LTS
RT ミドルウェア	OpenRTM-aist-1.0.0-RELEASE
開発言語	C++
コンパイラ	gcc 4.4.3

表 8－A **DataIntegrationRTC** の動作条件

実行周期	1000000[Hz]
------	-------------

表 8－B **DataIntegrationRTC** のデータポート (InPort)

名称	型	データ長	説明
in_angle	RTC::TimedDoubleSeq	2	ロボットモデルの 左右後輪の回転角度[rad]
in_color_image	RTC::TimedLongSeq	ピクセル数	ロボットモデル前方の カメラのカラー画像
in_range_sensor	RTC::TimedDoubleSeq	センサ出力数	距離センサから得られた 距離データ[m]
in_touch_sensor	RTC::TimedOctetSeq	12	タッチセンサの ON/OFF のデータ

表 8 - C DataIntegrationRTC のデータポート (OutPort)

名称	型	データ長	説明
out_angle	RTC::TimedDoubleSeq	2	ロボットモデルの 左右後輪の回転角度[rad]
out_color_image	RTC::TimedLongSeq	ピクセル数	ロボットモデル前方の カメラのカラー画像
out_range_sensor	RTC::TimedDoubleSeq	センサ出力数	測域センサから得られた 距離データ[mm]
out_touch_sensor	RTC::TimedState	1	タッチセンサの ON/OFF のデータ

表 8 - D DataIntegrationRTC のコンフィグレーション

名称	型	デフォルト値	説明
—	—	—	コンフィグレーションなし

2.8 3 輪移動ロボット

2.8.1 どんなロボットか

RT ミドルウェアを利用でき、安価であるため 20 人から 30 人程度の多人数向けの RT ミドルウェア学習にも使え、移動に関する最低限の機能を実装しているロボットとして、3 輪移動ロボット (3988 円) が開発されました。

完成したロボットの写真を図 6 に示します。

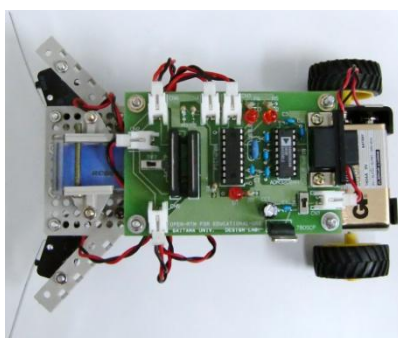


図 6 3 輪移動ロボット

選定した主な部品などを表 9 に示します。

表 9 選定した主な部品など

マイコン	PIC16F88
センサ	タッチセンサ用マイクロスイッチ (オプション：ワイヤレスカメラ)
アクチュエータ	DC モータ
外部との通信方法	RS232C

マイコンとして PIC16F88 を用いた理由は、PIC は安価であるためと開発環境を無料で利用できるためです。また、開発のための資料が豊富であり、独自に機能を追加・拡張することが容易であるためでもあります。

センサとしては、タッチセンサ用マイクロスイッチを使用します。これにより、マイクロスイッチの ON/OFF で容易に接触検知が行えるようになります。また、オプションとして、ワイヤレスカメラを実装できるようにしています。

アクチュエータとして DC モータを使用します。工作部品として広く使用されており、安価かつ入手容易で PIC による制御もモータドライバを介することで容易となるためです。

外部との通信に RS232C 通信を利用する理由は、使用する PIC16F88 は RS232C 通信を使用できる機能が備わっており、RS232C 通信ケーブルを用いることで容易にデータの送受信が行えるためです。

製作するために必要な詳細な部品表、ガーバデータなどは 3.9 章で詳しく解説します。

2.8.2 ロボットが受け取れる情報

PWM により左右モータを制御する際のデューティ比を、55 通りに分類した指令値の内 1 つを受け取ることができます。その指令値を PIC プログラム内での定義に従って変換することで左右モータの動作決定を行うことができます。

受け取れる指令値の具体的な仕様について解説します。指令値は、「速度を決定する文字データ」+「動作を決定する文字データ」+「o」+「k」という文字配列データとなるよう PIC プログラム内で定義されています。これにより、左右の DC モータの回転方向および 5 段階の速度の制御を行うことができますようになります。表 10-A に速度を決定する文字データを、表 10-B に動作を決定する文字データを示します。

表 10-A 速度を決定する文字データ

速度	受け取るデータ
最大速度	1
高速	2
通常速度	3
低速	4
最低速度	5

表 1 0－B 動作を決定する文字データ

動作	受け取るデータ
前進	a
後退	b
右折	c
急な右折	d
左折	e
急な左折	f
右後退	g
急な右後退	h
左後退	i
急な左後退	g
停止	k

表 1 0－A と表 1 0－B に示す文字データと「o」と「k」, つまり最大速度の前進を表す「1aok」, 最低速度の後退を表す「5bok」, 停止を表す「1kok」などを文字配列データとして受け取ることができます.

2.8.3 ロボットが送り出せる情報

ロボット前面に取り付けたタッチセンサ用マイクロスイッチの ON/OFF, つまりタッチセンサの接触の有無を送り出すことができます.

送り出せる情報を表 1 1 に示します.

表 1 1 送り出せるタッチセンサ情報

タッチセンサの状態	送り出すデータ
ON	1
OFF	0

表 1 1 に示す文字データを, タッチセンサの反応の有無として外部に送信することができます.