4tronix Pi2Go 移動台車ロボット
RT コンポーネント・サブシステム群
利用マニュアル

自己完結性を有する小型移動ロボット環境を用いた実演システムの開発

2017 年 12 月 13 日　第 3 版 甲南大学知能情報学部

目次

1. まえがき

小型移動ロボット Pi2Go と Raspberry Pi を組み合わせた自己完結性を有するロボットを制御するため RT コンポーネント群を開発しました．これらのコンポーネント群は Raspberry Pi 単体でも動作し，再利用性と拡張性の高いシステムを提供します．

このコンポーネント群は以下の特徴を持っています．

- ロボットに搭載されたマイクロコントローラ Raspberry Pi のみで，移動ロボットを制御することができます．
- Raspberry Pi を搭載した移動ロボットシステムを構築できるため，RT ミドルウェアのソフトウェア資産を生かし，容易に拡張性の高いロボットシステムを作ることができます．
- 本コンポーネント群は DUALSHOCK 3 で制御可能で，Pi2Go 搭載の Raspberry Pi に接続することも可能であるため，外部の PC を必要とせずにロボットを操縦することができます．
- 標準的な移動ロボット制御コンポーネントに対応できるように，Raspberry Pi, Windows 側で動作する Pi2Go コンポーネントに対応する Pi2Go ロボット側の制御プログラムを公開しています．

2. Raspberry Pi の設定方法

2.1. 用意するもの

- Raspberry Pi 3 Model B
- micro SD カード（8 GB 以上）
- WiFi 子機（本資料では Buffalo WLI-UC-GNM2 を使用する）
  - WiFi 付きの Raspberry Pi を使用する場合には不要
- USB ハブ
- USB ケーブル（A－micro B）
- PC
  - 内蔵もしくは外付けの SD カードスロットが必要
  - 本資料では Windows 7 機を使用する
- ディスプレイ（HDMI から信号を入力できるケーブルが必要）
- キーボード
- マウス

2.2. 作業手順

2.2.1. OS のイメージファイルのダウンロード

PC にて以下の URL にアクセスし，Raspbian jessie（Raspberry Pi 用 OS）のイメージファイル（約 1.5 GB）をダウンロードする．そして，Windows のコンテキストメニューの「すべて展開…」で，その ZIP ファイルを展開する[1]．
http://ftp.jaist.ac.jp/pub/raspberrypi/raspbian/images/raspbian-2017-07-05/2017-07-05-raspbian-jessie.zip

2.2.2. Win32 Disk Imager のインストール

PC にて以下の URL などから Win32 Disk Imager のインストーラー（本資料執筆時は Win32DiskImager-0.9.5-install.exe）を入手し，インストールする．
https://osdn.net/projects/sfnet_win32diskimager/

2.2.3. micro SD カードへのイメージファイルの書き込み

まず，SD カードを SD カードスロットにセットする．PC に SD カードスロットがない場合には，外付け SD カードスロットを用いる．次に，Win32 Disk Imager を起動し，イメージファイルとして 2017-07-05-raspbian-jessie.img を指定する（図 1）．そして，Write ボタンをクリックし，SD カードへの書き込みを行う．Write Successful のダイアログ（図 2）が表示されたら，Win32 Disk Imager を終了する．

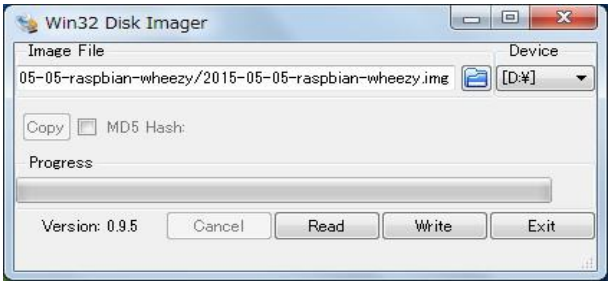


図 1：Win32 Disk Imager

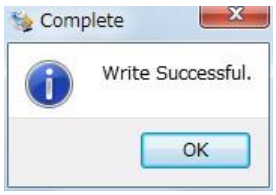


図 2：書き込み終了を示すダイアログ

[1] インストールされているソフトウェアによってはメニューが異なる場合がある．

2.2.4. Raspberry Pi の起動と RTM のインストール

2.2.3 にて作成した micro SD カードを Raspberry Pi の micro SD カードスロットに挿入する．Raspberry Pi にディスプレイ，キーボード，マウス，WiFi 子機（必要な場合）を接続した上で，USB ケーブルから電源を投入する．初回起動時には，必要に応じて言語やキーボードに関する設定を行う．

OS が起動し， WiFi の設定を行う．ネットワークに接続したら，以下の Web ページを参照し，コマンドを入力･実行し，Raspberry Pi 用の RTM をインストールする．

http://openrtm.org/openrtm/ja/content/raspberrypi_openrtm_installation

ただし，2017 年 10 月 30 日現在，上記のページには誤りがある．

wheezy を jessie に変更し，deb http://openrtm.org/pub/Linux/raspbian/ jessie main を新しい行に付け加える．/etc/apt/sources.list の編集は，vi エディタを用いる．以下に vi エディタで用いる vi コマンドを示す.

・入力
- `i`（カーソルの左に文字を入力する）
- `I`（カーソル行の行頭に文字を入力する）
- `o`（改行）

・削除
- `x`（1 文字削除）

・カーソル移動
- `h`（左），`l`（右），`k`（上），`j`（下）

・ファイル操作
- `ZZ` （保存して終了）
- `w + Enter`（上書き保存）
- `w` ファイル名 `+ Enter`（名前をつけて保存）
- `:wq + Enter`（保存して終了）
- `:q! + Enter` （保存せずに終了）
- `:q + Enter`（終了）

apt-get により関連ファイルのインストールを行う部分で，python-omniorb-omg と omniidl-python もインストールする必要がある．以下では，sudo を使って実行する例を示している．赤字の部分が上記のページに記載されていない部分である．

```
$ sudo apt-get update
$ sudo apt-get -y --force-yes install gcc g++ make uuid-dev
$ sudo apt-get -y --force-yes install libomniorb4-dev omniidl omniorb-nameserver
$ sudo apt-get -y --force-yes install openrtm-aist openrtm-aist-dev openrtm-aist-example
$ sudo apt-get -y --force-yes install openrtm-aist-python openrtm-aist-python-example
```

```
$ sudo apt-get install python-omniorb-omg omniidl-python
```

2.2.5. RTShell のインストール

RTShell は以下のコマンドを入力･実行する．

```
$ sudo pip install rtshell
$ sudo rtshell_post_install
```

2.2.6. 管理用スクリプトの配置

GitHub より ComponentConrol.py をダウンロードし，/home/pi/ に配置する．その後，/etc/ に ComponentControl.py を起動するためのスクリプトである run_pi2go.sh を配置する．初期設定では /Desktop/RTC/ にあるコンポーネントを指定しているため，各自のダウンロードしたディレクトリに合わせて ComponentControl.py を修正する．

次に， Raspberry Pi の起動時に ComponentControl.py が作動する環境を整備する．Raspberry Pi の /etc/rc.local を編集する．下図のように fi と exit0 の間に

```
sh run_pi2go.sh &
```

と書き込む．このとき，&を忘れると正常に動作しないため忘れずに入力すること．

変更を適用するため再起動を行い， ps コマンドなどで ComponentControl.py が動いているか確認を行う．

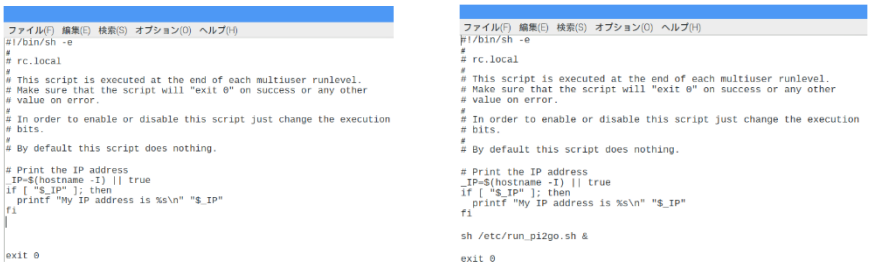


図 21：rc.local の変更前(左)，変更後

2.3. 管理用スクリプト群の解説

管理用スクリプトである ComponentControl.py は Python を用いて実装されている．

スクリプト上では以下のライブラリを用いている．

- os
- subprocess
- RPi.GPIO

- rtshell

os ライブラリ， subprocess ライブラリは Python 上でシェルを動かすためのライブラリである． RPi.GPIO は Raspberry Pi の GPIO を扱うためのライブラリである． RTShell は RT ミドルウェアの CUI ツールである．

2.3.1. 多重起動の防止法

スイッチによる管理で困難なものは，コンポーネントなどプロセスが複数立ち上がってしまうことである．この場合，正常に停止の指示を受け付けず暴走してしまう．
この解決のため Python の subprocess ライブラリを用いる．

subprocess ライブラリの call クラスは実行したシェルコマンドの返り値を 0 か 1 で出力する。その特性を活かし，以下のコードを使うことで特定のプロセスが走っているかどうかの判断ができる．

```
ret = subprocess.call(“ps ax | grep (調べたいプロセス名) | grep -v grep”)
```

ret の中身には調べたいプロセス名があれば 1 が、なければ 0 が代入される．
これを用い多重起動の防止が成功した．

2.3.2. 改良版管理用スクリプトについて

先述した管理用スクリプトでは，RT コンポーネントの起動や RT コンポーネント間の結線など使用者にあわせて書き換えてもらう必要があった．しかし，この状態では使用者が自分の環境に合わせて動かすためには rtshell の知識があることが前提であり，且つ多くの行数の訂正が必要であったため，とても利便性や可搬性に富んだものとは言い難いものであった．

そこで，使用者はコンポーネントの起動スクリプトの作成と，結線情報やコンフィグレーションパラメータの変更の情報が保存されている XML 形式のファイルや使用する計算機の IP アドレスを管理用スクリプトの引数とすることで管理用スクリプトに関して書き換えるコードの行数を最大で 0 行とし，非常に簡易的に使用できる管理用スクリプトを作成した．

2.3.3. 事前準備

事前にコンポーネント群を立ち上げ rtshell もしくは RTSystemEditor をもちいて結線し，コンフィグレーションパラメータを適宜変更しておく．

2.3.4. 改良版管理用スクリプトの配置

配置の仕方は 7.4 と同じく管理用プログラムを立ち上げるシェルスクリプトとコンポーネント群を起動するシェルスクリプトをそれぞれ `rc.local` に記述する．

またコンポーネント群を起動するシェルスクリプトでは管理用プログラムでネームサーバが立ち上がった後に各コンポーネント群を起動する必要があるため最初に `sleep` を `15` 秒入れることが望ましい．

以上で， Raspberry Pi の設定は完了である．

3. Pi2Go と Raspberry Pi の組み立て手順

3.1. 用意するもの

- Raspberry Pi 3 Model B（2 節で設定した Raspberry Pi）　　1 台
- micro SD カード（8GB 以上）
- タイヤ　　2 個
- モータ　　2 個
- ライントレース基板　1 個
- メイン基板　1 個
- 26 ピン エクステンド ヘッダ　1 個
- 9 ピン エクステンド ヘッダ　1 個
- ボール キャスタ　　1 個
- 電池ボックス　　1 個
- モータマウンタ　　4 個
- サポートプレート　　2 個
- 電池ボックス固定用ネジ　　2 個
- M3 ネジ　　4 本
- 27mm 六角ポスト　　2 個
- 25mm ネジ　4 本
- ナット　　4 個
- 16mm 六角ポスト　　4 個
- 5mm 六角ポストネジ 8 個
- 4mm ネジ　8 個

図 3：Pi2Go の全体部品

図 4：使用するネジ

3.2. 作業手順

3.2.1. フロントキャスタとライントレース基板の取り付け

27mm 六角ポスト 2 本を M3 ネジ 2 個を使って，メイン基板の写真 （図 5，6） で示す位置に裏から，ねじ止めする．

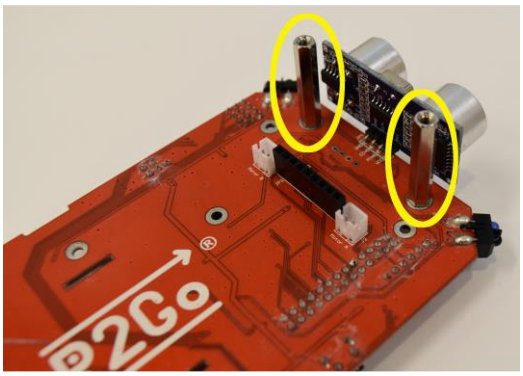


図 5：27mm 六角ポスト

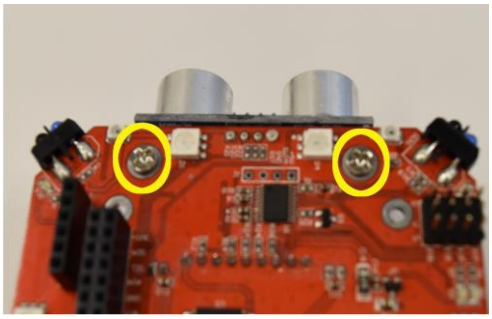


図 6：ネジ止め

9 ピンエクステンドヘッダのピン側を十分注意してメイン基板の写真（図 7）の位置のソケットに装着する．その上に，ライントレース基板のピンを写真（図 8）の向きに挿入し装着する（ピンは完全に置くまではささらない）．ライントレース基板上にボールキャスタを M3 ネジ 2 個でねじ止めする．

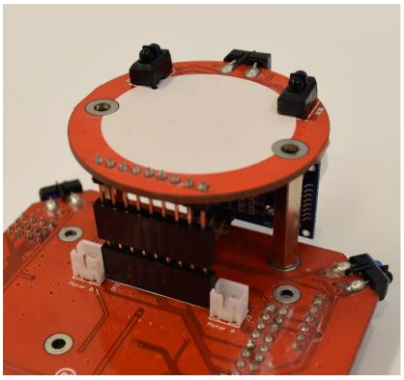


図 7：ライントレース基板

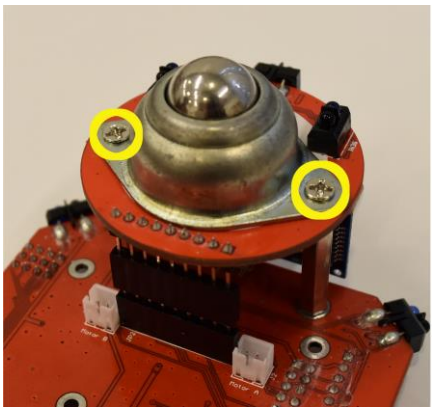


図 8：M3 ネジ

3.2.2. Raspberry Pi 固定支柱の取り付け

5mm 六角ポストネジ 4 個をメイン基板の写真（図 9）の裏側から 4mm ネジを使って，4 か所にねじ止めする．

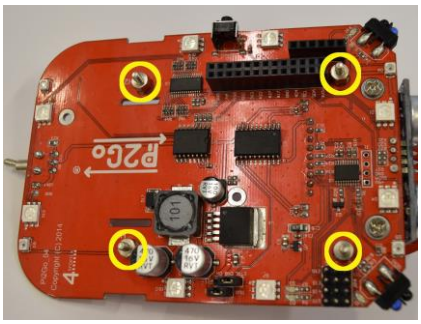


図 9：5mm 六角ポストネジ

取り付けた 5mm 六角ポストネジ 4 個の先にそれぞれ 16mm 六角ポストを取り付ける（図 10）.

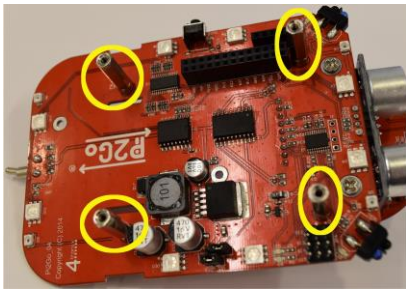


図 10：16mm 六角ポスト

3.2.3. モータの取り付け

3.2.2.とは反対面にモータをモータマウンタ（T 字），25mm ネジ，ナットを使って固定する．モータマウンタのうち，内側は基板の裏側から通す．モータは軸が外側に向く方向に取り付ける（図 11）.

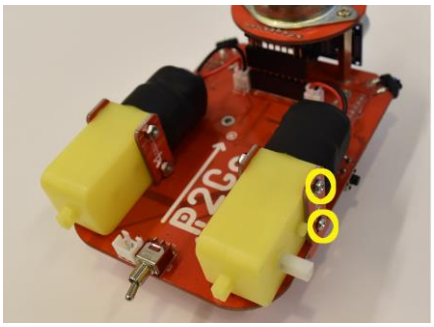


図 11：25mm ネジとナット

モータから出ているコネクタをメイン基板上のソケットに装着する．モータの軸にタイヤを差し込んで取り付ける（図 12）.

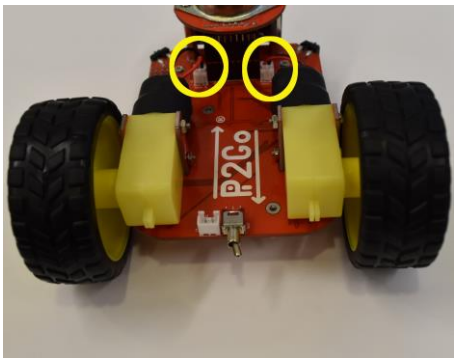


図 12：コネクタの装着

3.2.4. Raspberry Pi の取り付け

26 ピンエクステンドヘッダをメイン基板の 26 ピンソケットに装着する（図 13）.

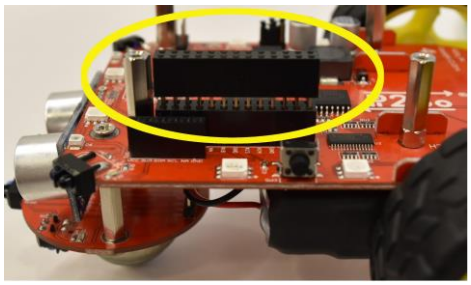


図 13：エクステンドヘッダの装着

写真（図 14）のとおり，Raspberry Pi の GPIO ピンヘッダの左 26 ピン分がささるように取り付ける.

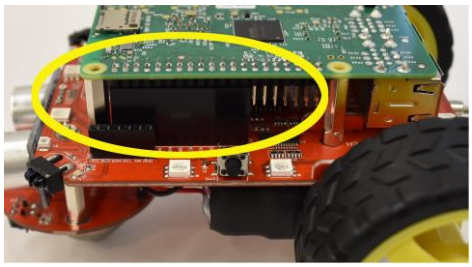


図 14：GPIO ピンヘッダの装着

3.2.5. 電池ボックスの取り付け

5mm 六角ポストネジを 4 か所に取り付ける（図 15）.

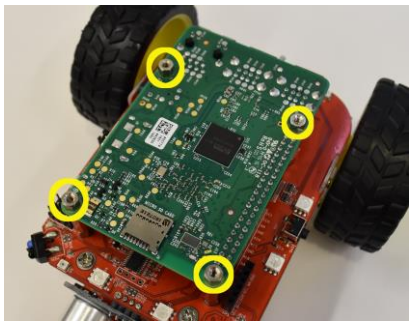


図 15：六角ポストネジ取り付け

写真（図 16）を参考にサポートプレート 2 個を 4mm ネジで固定する．サポートプレートには表裏があり，凹んでいる面を上にくるようにする．また，サポートプレートのネジ穴位置が写真（図 16）と同じようにする.

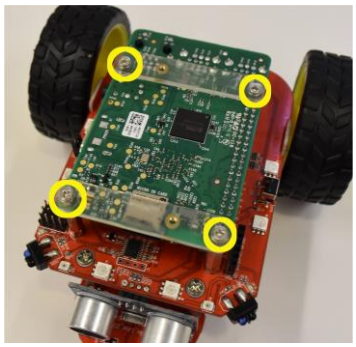


図 16：4mm ネジ

電池ボックスをネジで 2 か所を固定する（図 17）.

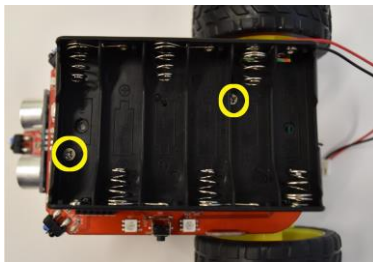


図 17：電池ボックス

電池ボックスから出ている電線をメイン基板のソケットへ接続する（図 18）. 以上で組み立ては完成となる（図 19）.

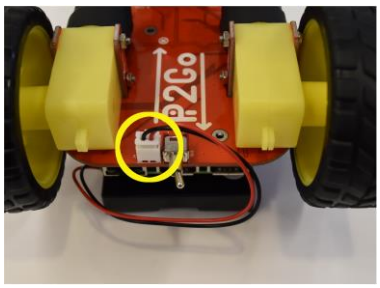


図 18：ソケット

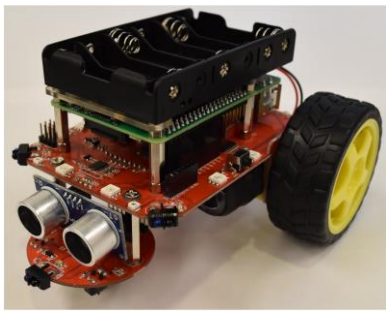


図 19： 完成図

4. Pi2Go の設定

4.1. 用意するもの

- Raspberry Pi 3 Model B（2 節で設定した Raspberry Pi）
- micro SD カード（8 GB 以上）
- USB ケーブル（A－micro B）
- HDMI ケーブル
- ディスプレイ
- マウス
- キーボード

4.2. 作業手順

4.2.1. Raspberry Pi へ必要なソフトウェア, ライブラリのインストール

Raspberry Pi を起動し, ターミナルをクリックして次のコマンドを入力・実行する.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install tightvncserver
$ sudo apt-get install xrdp
$ sudo apt-get install python-smbus python3-smbus python-dev python3-dev
$ sudo leafpad /boot/config.txt
```

4.2.2. Config.txt の編集

4.2.1.節の最後のコマンドを実行すると、テキストエディタが起動する. テキストエディタの末尾に次の行を追加する. （i2c1 の下線部は数字の 1 である）.

```
dtparam=i2c1=on
dtparam=i2c_arm=on
```

5. DUALSHOCK 3 と Raspberry Pi の接続方法

5.1. 用意するもの

- Raspberry Pi 3 Model B（2 節で設定した Raspberry Pi）
- micro SD カード（8GB 以上）
- DUALSHOCK 3 （PlayStation 3 用のコントローラ）
- ディスプレイ
- キーボード
- マウス
- USB ケーブル
- HDMI ケーブル

5.2. 作業手順

5.2.1. DUALSHOCK 3 と Raspberry Pi の有線接続

Raspberry Pi の電源を入れ、ターミナルを起動し、以下のコマンドを入力・実行する.

```
$ sudo apt-get -y install libsub-dev joystick python-
$ wget http://www.pabr.org/sixlinux/sixpair.c
$ gcc -o sixpair sixpair.c -lusb
```

次に DUALSHOCK 3 と Raspberry Pi を USB ケーブルで有線接続し、そのまま再起動する.

5.2.2. DUALSHOCK 3 の MAC アドレス

再びターミナルを起動し、以下のコマンドを入力・実行する.

```
$sudo ./sixpair
```

実行後、 DUALSHOCK 3 の MAC アドレスが表示されるので、メモしておく.

5.2.3. Bluetooth 接続の設定

DUALSHOCK 3 と Raspberry Pi 3 の USB ケーブルを抜く.（ここで抜いておかないと無線接続が失敗に終わる. 失敗した場合、 Raspberry Pi の 5.2.1 節の再起動からやり直す）.
次に以下のコマンドを入力･実行する.（ターミナルが Bluetooth モードになり，来のコマンドは通らないようになる）.

```
$ sudo bluetoothctl
```

上記のコマンドを実行後 DUALSHOCK 3 の PS ボタン（図 20）を押し、以下のコマンドを入力・実行する.（コマンド実行の際、 Raspberry Pi と DUALSHOCK 3 の MAC アドレスが表示され続けるが無視する）.

DUALSHOCK 3 の 1 の場所に赤いランプが点灯するか、どのボタンを押しても点滅しなくなれば無線接続は完了である.

```
$ discoverable on
$ agent on
$ connect  DUALSHOCK 3 の MAC アドレス （例:$ connect 12:A1:12:1B:1C:AB）
$ trust  DUALSHOCK 3 の MAC アドレス （例:$ trust 12:A1:12:1B:1C:AB）
```

無線接続を解除したい場合は、次のコマンドを入力・実行する.

```
$ power off
```

Bluetooth モードを解除したい場合は次のコマンドを入力･実行する.

```
$ quit
```

5.2.4. 設定後の無線接続

上記の設定が完了している場合、電源をつけてから以下の方法で無線接続が可能である.

ターミナルを起動し，以下のコマンドを入力･実行する．実行後，DUALSHOCK 3 の PS ボタンを押す.

```
$ sudo bluetoothctl
$ power on
```

図 20:PS ボタン

6. 作例の動かし方

本節では DUALSHOCK 3 を用いて 4tronix Pi2Go を制御する方法を説明する．Raspberry Pi 上で RTShell を動かすことによって一連の処理を Raspberry Pi のみで実行することも可能だが，ここでは，PC 上の RT System Editor でコンポーネントの接続や実行を行う方法を説明する．

6.1. 用意するもの

- Pi2Go に Raspberry Pi 3 Model B を接続したシステム（〇節までで設定したもの）
- Raspberry Pi 3 Model B に接続するディスプレイ，キーボード，マウス
- PC（上記の Raspberry Pi と同じネットワークに接続されているもの）
- DUALSHOCK 3
- USB ケーブル
- 単三電池　6 本

6.2. 作業手順

6.2.1. システムの起動

Pi2Go に搭載されている Raspberry Pi を起動する．Raspberry Pi にはディスプレイ，キーボード，マウス，DUALSHOCK 3 を先に接続しておく．

6.2.2. Name サーバの起動

2 節での omniorb-nameserver のインストール時に，システム起動時に自動的に omniORB のネームサービスが起動するようになっている．しかし，起動がうまくいかない場合には rtm-naming コマンドによりネームサーバを再起動する．

6.2.3. Raspberry Pi の IP アドレスの取得

ifconfig を使うなどして Raspberry Pi の IP アドレスを取得してメモしておく．

6.2.4. RT コンポーネントのダウンロード

本プロジェクトのホームページから以下の 3 つのコンポーネントをダウンロードする．

```
(1) Dualshock3.py
(2) Pi2GoRTC.py
(3) Pi2GoVelRTC.py
```

6.2.5. DUALSHOCK 3 と Raspberry Pi 3 の無線接続

5 節の通りに無線接続を行う

6.2.6. RT コンポーネントの起動

Raspberry Pi にて上記 3 つのコンポーネントを起動する．具体的には，Raspberry Pi の端末上にて以下のコマンドを実行する．

```
$ cd (上記ファイルのあるフォルダー)
$ sudo python Dualshock3.py&
$ sudo python Pi2GoRTC.py&
$ sudo python P2GoVelRTC.py&
```

6.2.7. コンポーネントの接続・実行

PC で RT System Editor を起動し，「name server の追加」で Raspberry Pi の IP アドレスを追加する．すると，上記 3 つのプロセスが表示されるので，それらを写真（図 21） のように配置，接続する． Pi2Go の電池ボックスに電池単 3 電池（6 個）を正しくセットし，Pi2Go 本体の電源を ON にした後 30 秒待つ．以上の設定が済んだら，アクティベートを行う．これによって，DUALSHOCK 3 で Pi2Go の制御が可能となる．

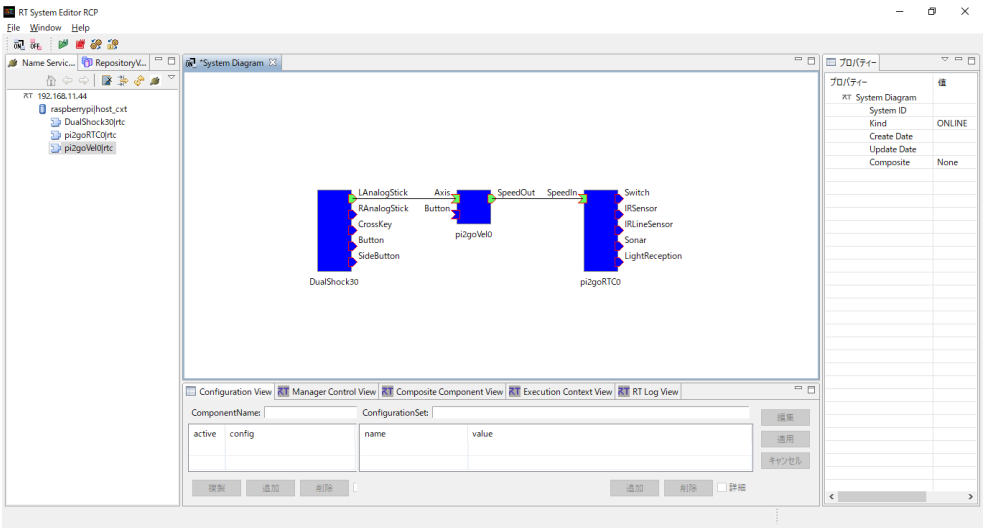


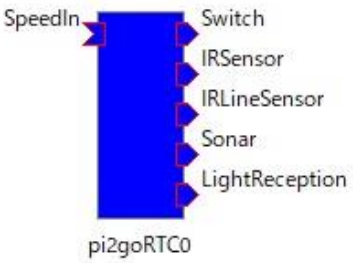
図 21：コンポーネントの接続

7. コンポーネントの説明

pi2goRTC

概要

速度と角速度の入力に応じて pi2go を移動させる． Pi2Go 本体のボタンの On/Off，赤外線障害物センサ，ライントレースセンサ，音波距離センサ，受光センサのデータを出力する．



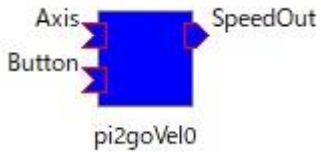
データポート

名称	フローポート	データ型	説明
speedIn	InPort	RTC.TimedVelocity2D	速度情報を入力する．
Switch	OutPort	RTC.TimedBooleanSeq	スイッチの On/Off を出力する．
IRSensor	OutPort	RTC.TimedBooleanSeq	赤外障害物センサの情報を出力する．
IRLineSensor	OutPort	RTC.TimedBooleanSeq	ライントレースセンサの情報を出力する．
Sonar	OutPort	RTC.TimedDoubleSeq	音波距離センサの情報を出力する．
LightReception	OutPort	RTC.TimedDoubleSeq	受光センサの情報を出力する．

pi2goVelRTC

概要

ジョイスティックから入力されたボタン情報を速度情報に変換し出力する.



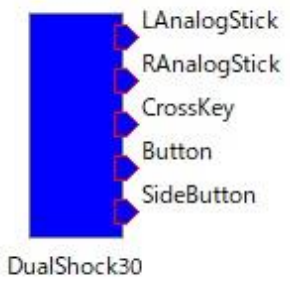
データポート

名称	フローポート	データ型	説明
Axis	InPort	RTC.TimedDoubleSeq	ジョイスティックへの入力情報を入力する.
Button	InPort	RTC.TimedBooleanSeq	ジョイスティックへのボタンの入力情報を入力する.
SpeedOut	OutPort	RTC.TimedVelocity2D	速度情報を出力する.

DUALSHOCK3

概要

ジョイスティックやゲームパッドへの入力を出力する.



データポート

名称	フローポート	データ型	説明
LAnalogStick	OutPort	RTC.TimedDoubleSeq	左スティックの情報を出力する.
RAnalogStick	OutPort	RTC.TimedDoubleSeq	右スティックの情報を出力する.
CrossKey	OutPort	RTC.TimedBooleanSeq	十字ボタンの情報を出力する.
Button	OutPort	RTC.TimedBooleanSeq	ボタンの情報を出力する.
SideButton	OutPort	RTC.TimedBooleanSeq	側面ボタンの情報を出力する.

8. 更新情報

- 2017 年 10 月 31 日　初版
- 2017 年 12 月 6 日　第 2 版
- 2017 年 12 月 18 日　第 3 版