

Zumo ロボットシステム, コンポーネント群 利用マニュアル

SI 2016 「Zumo と Raspberry Pi を用いた教育用ロボット環境」

2017 年 12 月 02 日第 3 版
甲南大学知能情報学部

目次

目次	2
1.	まえがき 4
2.	Raspberry Pi の設定方法..... 5
2.1.	用意するもの..... 5
2.2.	作業手順 5
2.2.1.	OS のイメージファイルのダウンロード..... 5
2.2.2.	Win32 Disk Imager のインストール..... 5
2.2.3.	micro SD カードへのイメージファイルの書き込み (Raspberry Pi 2, Raspberry Pi A+の場合) 6
2.2.4.	Raspberry Pi の起動と RTM, RTShell のインストール..... 6
3.	Zumo 32U4 Robot の設定..... 8
3.1.	用意するもの..... 8
3.2.	作業手順 8
3.2.1.	PC へ Arduino Software のインストール 8
3.2.2.	PC へ Zumo 用デバイスドライバのインストール 8
3.2.3.	Arduino 開発環境への Zumo 用設定の追加..... 9
4.	Zumo と Raspberry Pi の組み立て手順 11
4.1.	用意するもの..... 11
4.2.	作業手順 11
4.2.1.	Zumo 側の準備 11
4.2.2.	RaspberryPi ユニットの取り付け 11
5.	Zumo 用プログラムの書き込み 14
5.1.	用意するもの..... 14
5.2.	作業手順 14
6.	作例の動かし方 15
6.1.	用意するもの..... 15
6.2.	作業手順 15
6.2.1.	システムの起動..... 15
6.2.2.	Zumo へのプログラムの書き込み..... 16
6.2.3.	Name サーバーの起動..... 16
6.2.4.	Raspberry Pi の IP アドレスの取得 16
6.2.5.	RT コンポーネントのダウンロード..... 16
6.2.6.	RT コンポーネントの起動..... 16
6.2.7.	コンポーネントの接続・実行..... 16
7.	タクトイルスイッチによるコンポーネント群の一括管理..... 18

7.1	ユニバーサル基板の実装	18
7.1.1.	必要なもの.....	18
7.1.2.	回路図.....	18
7.1.3.	実装例.....	18
7.2.	ユニバーサル基板の取り付け	19
7.3.	管理用スクリプトの配置	19
8.	コンポーネントの説明	20
9.	付録：Raspberry Pi A+を用いたシステム構築.....	26
9.1.	はじめに	26
9.2.	用意するもの.....	26
9.3.	電圧変換モジュールの作成.....	26
9.4.	治具のノイズ対策処理	27
9.5.	治具の取り付け	28
9.6.	電圧変換モジュールの取り付け.....	29
9.7.	Raspberry Pi の取り付け.....	30
9.8.	ケーブルの接続	30
10.	更新情報	33

1. まえがき

小型台車ロボット Zumo 32U4 と Raspberry Pi を組み合わせた教育用ロボットを制御するための 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, 安価かつ小型で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.
- 最小構成で Raspberry Pi を搭載した卓上の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.
- コンポーネント群は OS のプラットフォームに依存しないため, 柔軟性が高い教育用ロボットシステムを構築できます.
- Raspberry Pi を用いた移動ロボット構築例を作るための治具の CAD データ, 3D プリント用データを公開しています.
- 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 Zumo コンポーネントに対応する Zumo ロボット側の制御プログラムを公開しています.
- 本ロボットシステムの RT コンポーネントは, 2015 年に応募した XML で動作する対話ロボットシステムのロボットベース部を改良した, 現行の小型漫才ロボット(こちら)の制御に用いられています.
- 2016 年に応募したシステムに改良を加え, 約一年の間デモンストレーションの場などで実際に稼働し, システムの実用性を示しています.

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

以下の OS のインストールは, Raspberry Pi 2, Raspberry Pi A+の場合について説明している. Raspberry Pi 3 では, 2016 年 12 月標準のインストールバージョン (jessie)を使用する.

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 wheezy (Raspberry Pi 用 OS) のイメージファイル (約 1 GB) をダウンロードする¹. そして, Windows のコンテキストメニューの「すべて展開...」その ZIP ファイルを展開する².

<http://ftp.jaist.ac.jp/pub/raspberrypi/raspbian/images/raspbian-2015-05-07/2015-05-05-raspbian-wheezy.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/

¹ Raspbian の最新版は jessie であるが, Raspberry Pi 2 では wheezy を使用している. Raspberry Pi 3 では jessie を用いてシステムを構築している.

² 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.

2.2.3. micro SD カードへのイメージファイルの書き込み (Raspberry Pi 2, Raspberry Pi A+の場合)

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 2015-05-05-raspbian-wheezy.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。

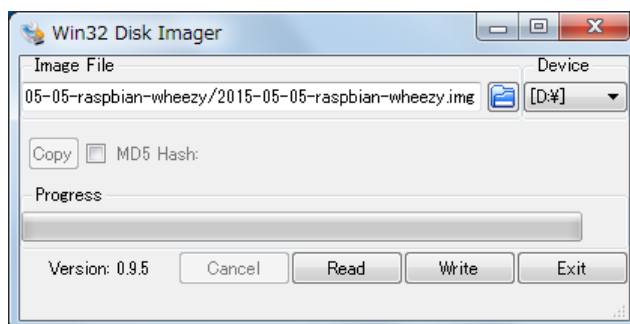


図 1 : Win32 Disk Imager

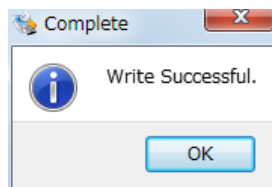


図 2 : 書き込み終了を示すダイアログ

2.2.4. Raspberry Pi の起動と RTM, RTShell 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、コマンドプロンプトが表示されたら、startx コマンドにより GUI 環境を起動する。次に、WiFi の設定を行う。ネットワークに接続したら、以下の Web ページに従って、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、2016 年 10 月 18 日現在、上記のページには誤りがある。/etc/apt/sources.list の編集後、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
```

続けて RTShell のインストールを行う。

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

以上で Raspberry Pi の設定は完了である。

3. Zumo 32U4 Robot の設定

3.1. 用意するもの

- Zumo 32U4 Robot
- USB ケーブル (A－micro B)
- PC

3.2. 作業手順

3.2.1. PC へ Arduino Software のインストール

PC にて以下の URL にアクセスし、最新の Arduino Software をダウンロードしインストールする。執筆時点のバージョンは 1.6.12 である。

<https://www.arduino.cc/en/Main/Software>

3.2.2. PC へ Zumo 用デバイスドライバのインストール

PC にて以下の URL にアクセスし、a-star-2.0.0.zip をダウンロードし展開する。次に、展開したフォルダーの drivers 内にある a-star.inf をコンテキストメニューからインストールする (図 3)。

https://www.pololu.com/file/download/a-star-2.0.0.zip?file_id=0J743

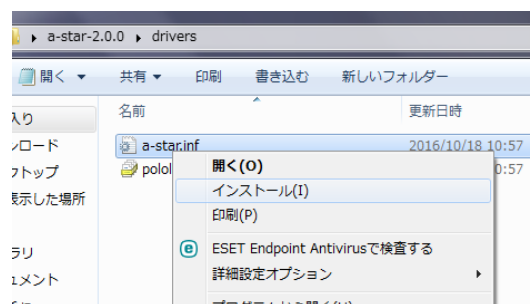


図 3：ドライバーのインストール

さらに、a-star-2.0.0¥add-on フォルダ内 polou フォルダ全体を C:¥Program Files (x86) ¥Arduino¥hardware の中にコピーする (図 4 図)。

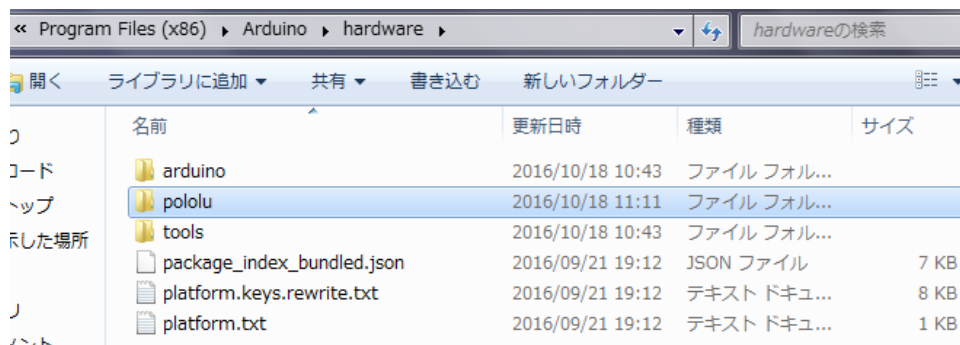


図 4 : Polou フォルダのコピー

3.2.3. Arduino 開発環境への Zumo 用設定の追加

PC と Zumo を A-Micoro B USB ケーブルで接続し、PC にて Arduino 開発環境を起動する。起動後、ツール→ボードから Polou A-Star 32U4 を選択する (図 5)。

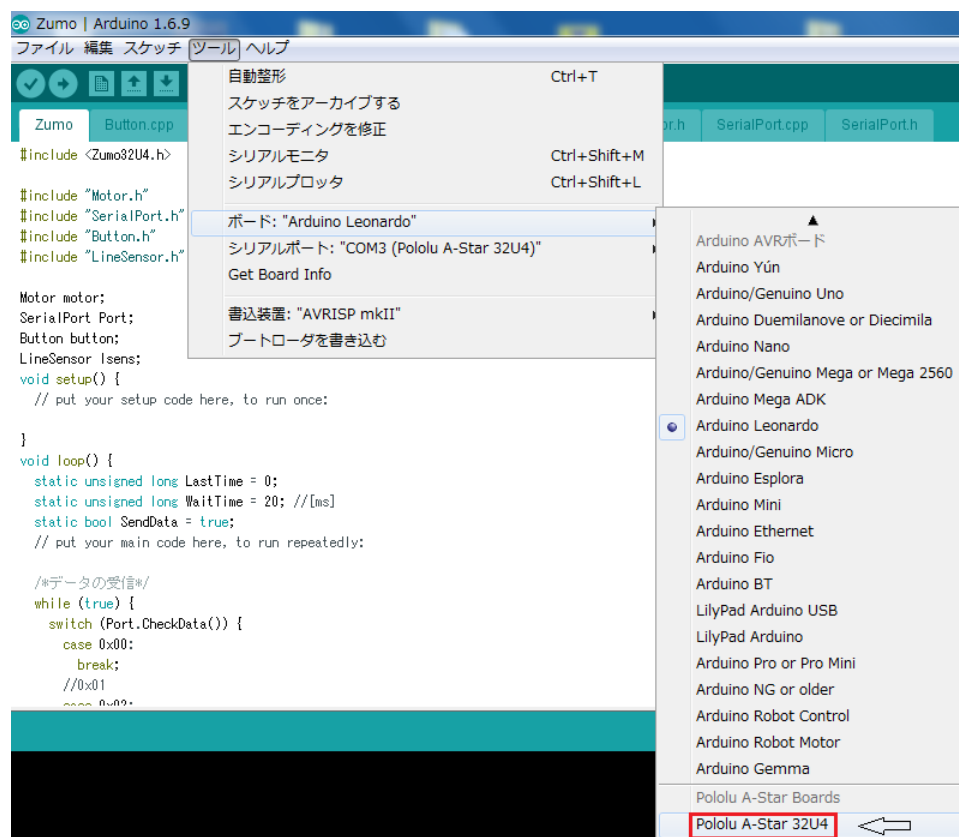


図 5 : ボードの選択

最後に、デバイスマネージャーのポート（COM と LPT）に Pololu A-star 32U4 が表示されていることを確認する（図 6）。

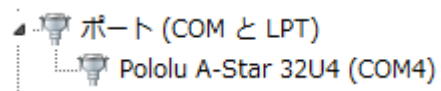


図 6 : COM ポートの確認

以上で Zumo 32U4 Robot の設定は完了である。

参考 : <https://www.pololu.com/docs/0J61/6.1>

4. Zumo と Raspberry Pi の組み立て手順

4.1. 用意するもの

- Raspberry Pi 3 Model B
- Zumo 32U4 Robot
- 3D プリンタで出力した治具 A-1, A-2, A-3, A-4 各 1 個 (CAD データおよび 3D プリンタ用データが本プロジェクトのホームページにて公開されている)
- M3×15 ネジ 8 本
- M2.6×15 ネジ 4 本
- M2×25 ネジ 14 本
- M3 ナット 8 個
- M2.6 ナット 4 個
- M2 ナット 14 個 (うち 4 個は Zumo に付いているものを使用する)
- スペーサー (M2.6 ネジ用, 高さ 3 mm) 4 個 (3D プリンタ用データが本プロジェクトのホームページにて公開されている)

4.2. 作業手順

4.2.1. Zumo 側の準備

Zumo についているブレード, 赤外線センサアレイ, 液晶ディスプレイを取り外す.

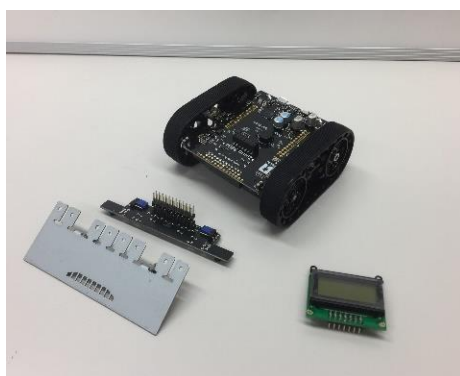


図 7 : Zumo のブレード, 赤外線センサアレイ, 液晶ディスプレイを取り外した状態

4.2.2. RaspberryPi ユニットの取り付け

治具 A-1, A-2 を Zumo の基板上にネジで固定する. 図 8 左のように先に治具 A-1 を基板に載せ, その上に図 8 右のように治具 A-2 を載せてネジで固定する. ブレードが固定されていた 2 か所 (図 9) では治具から Zumo に向かってネジを通し, それ以外の 6 か所では Zumo の電池ボックスの内側 `sudo apt-get -y --force-yes install openrtm-aist openrtm-aist-dev`

openrtm-aist-example から治具に向かってネジを通す。



図 8 : 治具 A-1 (左図), 治具 A-2 (右図) の取り付け

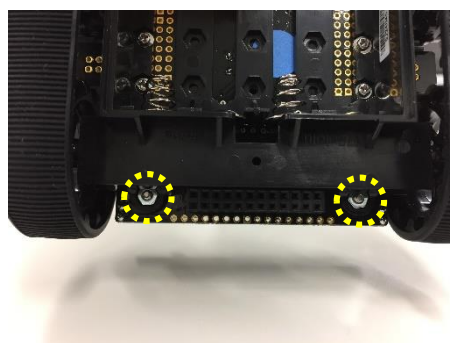


図 9 : 治具側からネジを挿入する部分

ネジの取り付け完了後, 赤外線センサアレイを元の位置のピンコネクタに挿す。

治具 A-3 に Raspberry Pi を取り付ける。

この際, 治具 A-3 と Raspberry Pi の間にスペーサーを取り付ける。



図 10 : 治具 A-3 へ Raspberry Pi を取り付けた様子

治具 A-4 にユニバーサル基板用のネジを天板の裏側から取り付ける。
治具 A-3 に治具 A-4 を取り付ける。
図 11 のように治具 A-3 から治具 A-4 に向かってネジを通す。

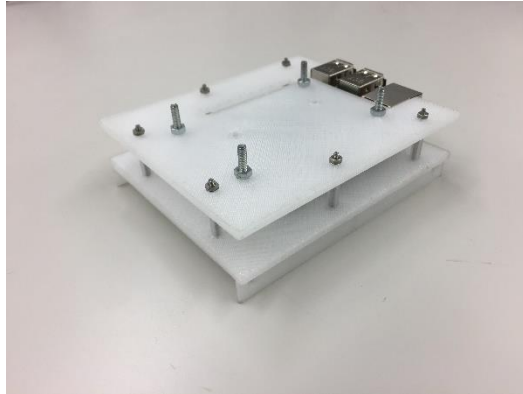


図 11 : 治具 A-3 に治具 A-4 を取り付けた様子

治具 A-2 に治具 A-3 を取り付ける。

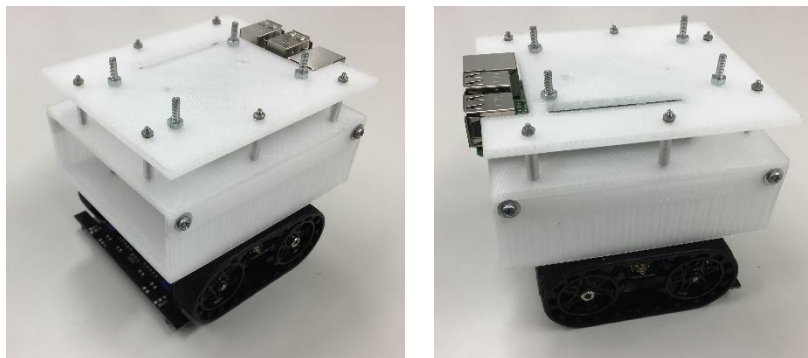


図 12 : 治具 A-2 に治具 A-3 を取り付けた様子

以上で組み立ては完了である。

5. Zumo 用プログラムの書き込み

5.1. 用意するもの

- Zumo 32U4 Robot
- USB ケーブル (A-micro B)
- PC (3 節の設定を済ませたもの)
- プログラム (本プロジェクトの web ページよりダウンロードする)
 - Zumo 後方の USB ポートを経由して Zumo を制御する場合 : Zumo-USB フォルダ (通常はこちらを使用する)
 - Raspberry Pi Model A+から GPIO シリアル通信を用いて Zumo を制御する場合 (付録を参照) : Zumo-GPIO フォルダ

5.2. 作業手順

まず, PC と Zumo を USB ケーブルで接続する. 次に, Zumo にプログラムを書き込む. 以下, Zumo 後方の USB ポートを経由して Zumo を制御する場合を例に説明する. PC にて Zumo-USB.ino をダブルクリックして, Arduino software を起動する (Raspberry Pi Model A+などで GPIO シリアル通信によって制御する場合には Zumo-GPIO.ino をダブルクリックする). そして, メニューバーの直下にある矢印アイコンをクリックすることによって, プログラムが検証, コンパイルされ, Zumo に書き込まれる (図 13).

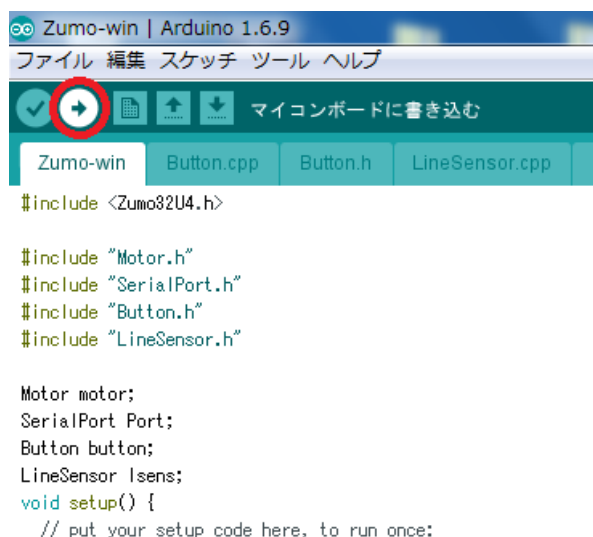


図 13 : Zumo へのプログラムの書き込み

以上で Zumo 用プログラムの書き込みは完了である.

6. 作例の動かし方

本節ではゲームパッドを用いて Zumo 32U4 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。

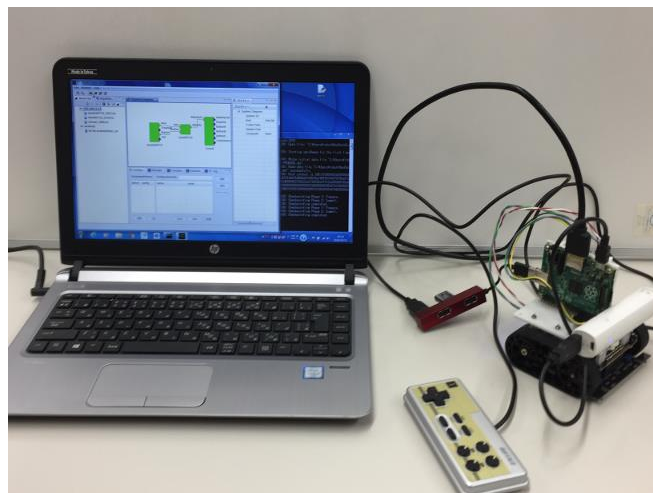


図 14：ゲームパッドによる Zumo の制御（Raspberry Pi A+での構成）

6.1. 用意するもの

- Zumo 32U4 に Raspberry Pi 3 Model B を接続したシステム（5 節までで設定したもの）³
- Raspberry Pi 3 Model B に接続するディスプレイ，キーボード，マウス
- ゲームパッド（本資料では Buffalo BGCFC801 を使用）
- 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）

6.2. 作業手順

6.2.1. システムの起動

Zumo および Raspberry Pi 3 Model B を起動する。Raspberry Pi 3 Model B にはディスプレイ，キーボード，マウス，ゲームパッドを接続しておく。

³ 2016 年 10 月 31 日現在，Raspberry Pi A+はこの処理に対して非力であることが分かっている。

2016 年 12 月 10 日現在では，Raspberry Pi 3 で Zumo 後方の USB ポートを経由したシリアル通信を用いることを標準とする。

6.2.2. Zumo へのプログラムの書き込み

5 節の記述に従って、Zumo_USB.ino を Zumo に書き込む。

6.2.3. Name サーバーの起動

2.2.4 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⁴。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。

6.2.4. 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。

6.2.5. 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) Zumo.py
- (2) JoyVelRTC.py
- (3) JoystickRTC.py

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 Zumo.py &
$ sudo python JoyVelRTC.py &
$ sudo python JoystickRTC.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、それらを図 15 のように配置、接続する。

次に、Zumo コンポーネントのコンフィギュレーションの Port の値を /dev/ttyACM0 に変更する (図 16)。これは接続するシリアルポートの設定である。

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、ゲームパッドで Zumo を制御することができるようになる。

⁴ 参考: http://openrtm.org/openrtm/ja/content/raspberrypi_openrtm_installation

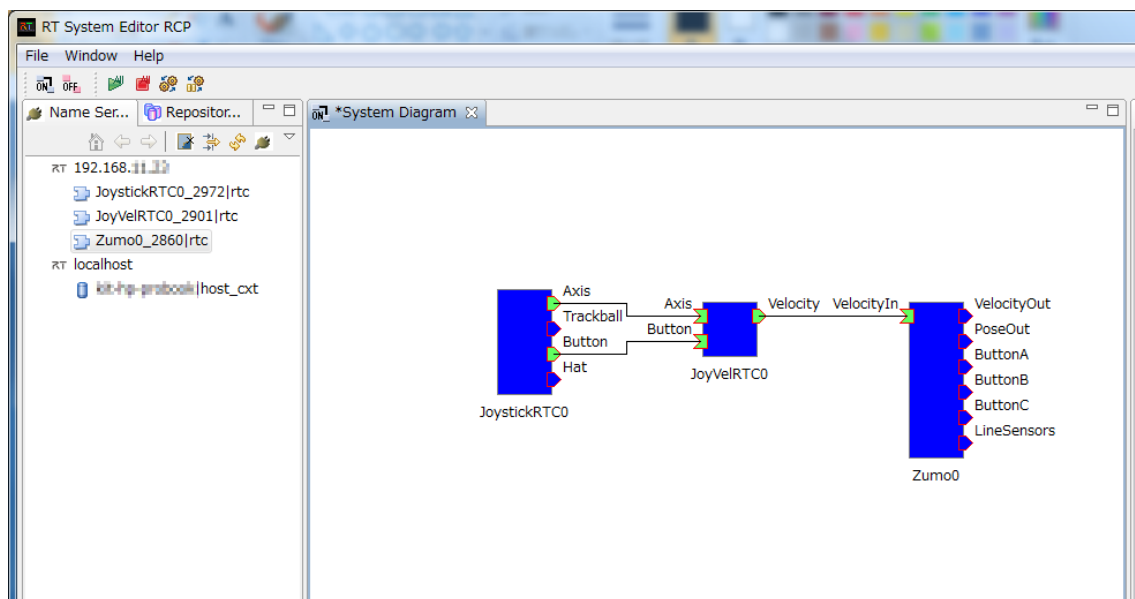


図 15 : コンポーネントの接続

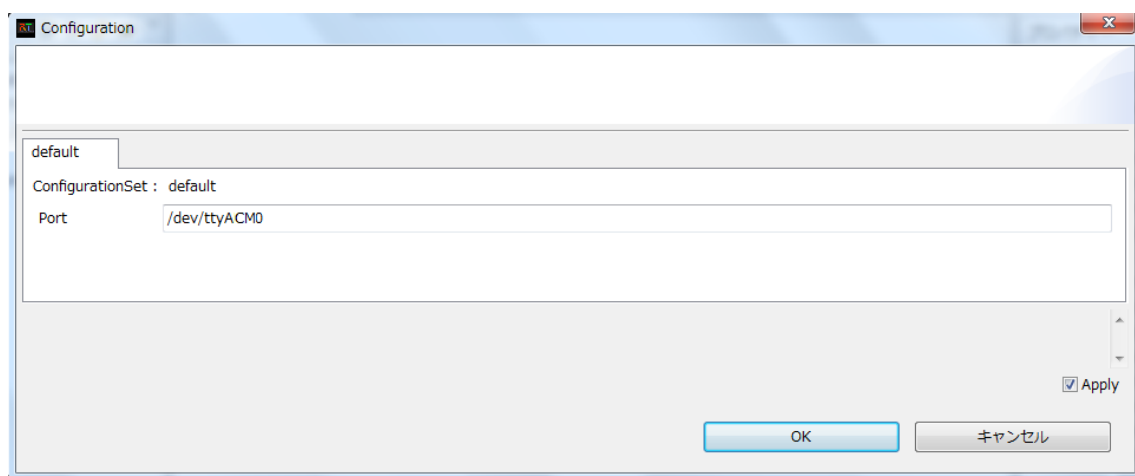


図 16 : Zumo コンポーネントのコンフィギュレーション（シリアルポートの設定） : Raspberry Pi A+の場合

7. タクタイルスイッチによるコンポーネント群の一括管理

本節ではタクタイルスイッチを用いたコンポーネント群の一括管理を行う方法を説明する。

7.1 ユニバーサル基板の実装

7.1.1. 必要なもの

- ユニバーサル基盤 1 枚
- タクタイルスイッチ 1 個
- 抵抗 22k Ω 3 個
- LED 1 個
- [MI2CLCD-01 モジュール](#) 1 個
- 各配線 必要分

7.1.2. 回路図

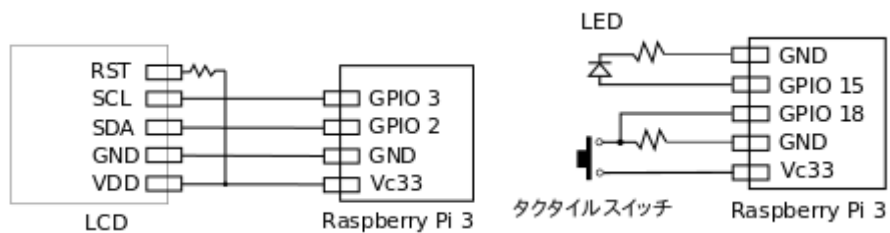


図 17 : 回路図

7.1.3. 実装例

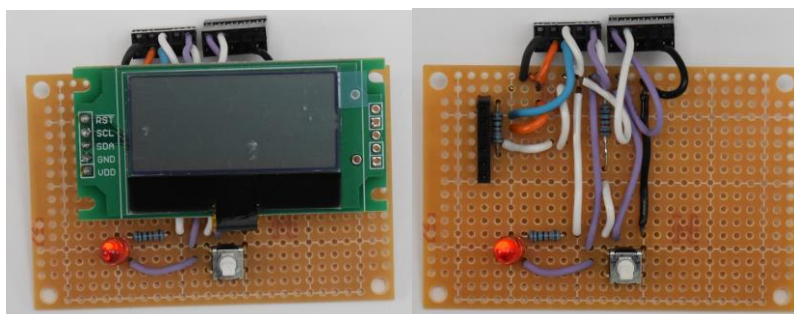


図 18 : 実装例 表面 LCD 有り(左), 表面 LCD 無し(右)

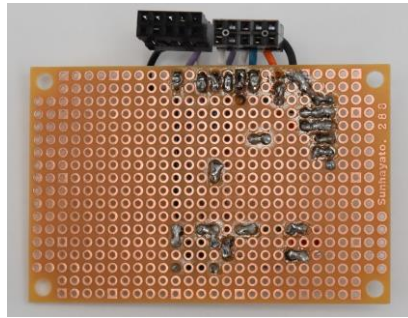


図 19：実装例 裏面

7.2. ユニバーサル基板の取り付け

治具 A-4 へ完成したユニバーサル基板を取り付ける。

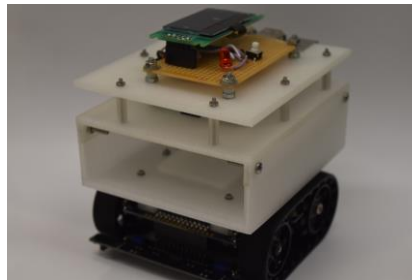


図 20：基板の取り付け

7.3. 管理用スクリプトの配置

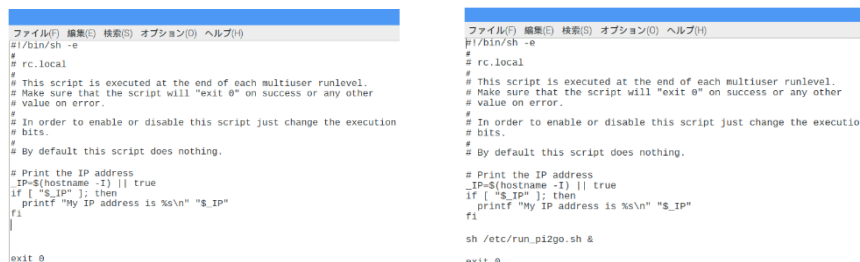
GitHub より `ComponentControl.py` をダウンロードし、`/home/pi/`に配置する。その後、`/etc/`に `ComponentControl.py` を起動するためのスクリプトである `runZumo.sh` を配置する。初期設定では `/Desktop/RTC/`にあるコンポーネントを指定しているため、各自のダウンロードしたディレクトリに合わせて `ComponentControl.py` を修正する。

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 runZumo.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。



```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

exit 0
```

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

sh /etc/run_pi2go.sh &

exit 0
```

図 21 : rc.local の変更前(左), 変更後

7.4. 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。

7.4.1. rtshell によるコンポーネント群の管理

本節では rtshell の解説を行う。

- rtcon

rtcon はコンポーネント同士の結線を行うコマンドである。

使用法 :

\$rtcon (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc/(データポート名) (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc/(データポート名)

- rtconf

rtconf は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変更するコマンドである。

使用法 :

\$rtconf (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc set (コンフィギュレーションパラメータの名称) (変更後の値)

- **rtact**

rtact はコンポーネントをアクティブ状態にするためのコマンドである。

使用法：

\$rtact (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc

- **rtdeact**

rtdeact はコンポーネントをデアクティブ状態にするためのコマンドである。

使用法：

\$rtdeact (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc

- **rtexit**

rtexit はコンポーネントを削除するためのコマンドである。

使用法：

\$rtdeact (IP アドレス)/(PC ネーム).host_cxt/(コンポーネント名).rtc

他にも **rtshell** のコマンドがあるが、それらは[こちら](#)で適宜確認してほしい。

7.4.2. 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。

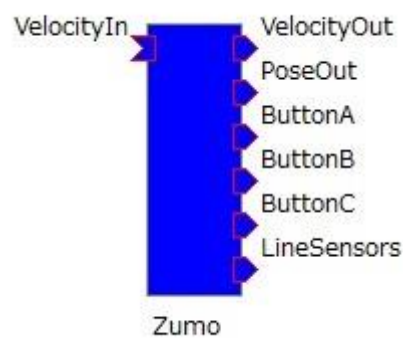
これを用い多重起動の防止が成功した。

8. コンポーネントの説明

Zumo

概要

速度と角速度の入力に応じて Zumo を移動させる。速度と角速度，オドメトリ，Zumo 本体のボタンの On/Off，赤外線センサレイ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
VelocityIn	InPort	RTC.TimedVelocity2D	速度情報を入力する。
VelocityOut	OutPort	RTC.TimedVelocity2D	速度情報を出力する。
PoseOut	OutPort	RTC.TimedPose2D	起動直後からのオドメトリを出力する。
ButtonA	OutPort	RTC.TimedBoolean	ButtonA の On/Off を出力する。
ButtonB	OutPort	RTC.TimedBoolean	ButtonB の On/Off を出力する。
ButtonC	OutPort	RTC.TimedBoolean	ButtonC の On/Off を出力する。
LineSensors	OutPort	RTC.TimedULongSeq	ラインセンサデータを出力する。

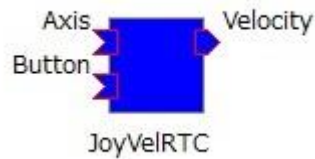
コンフィギュレーションパラメータ

名称	データ型	初期値	説明
Port	文字列	COM1	使用するポート名

JoyVelRTC

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。
Velocity	OutPort	RTC.TimedVelocity2D	速度情報を出力する。

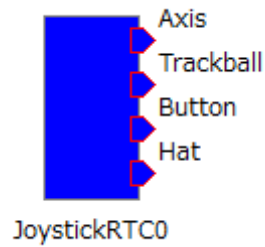
コンフィギュレーションパラメータ

名称	データ型	初期値	説明
VXGain	浮動小数	-1.0	ジョイスティックからの入力に対する速度の重み
VAGain	浮動小数	1.0	ジョイスティックからの入力に対する角速度の重み
PivotButtonN	整数	0	超信地旋回に切り替えるボタンの指定
VXAxisN	整数	1	速度に用いる軸番号
VAAxisN	整数	0	角速度に用いる軸番号

JoystickRTC

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
Axis	OutPort	RTC.TimedDoubleSeq	座標軸を出力する.
Trackball	OutPort	RTC.TimedDoubleSeq	トラックボールの情報を出力する.
Button	OutPort	RTC.TimedBooleanSeq	ボタンの情報を出力する.
Hat	OutPort	RTC.TimedShortSeq	ハットスイッチの情報を出力する.

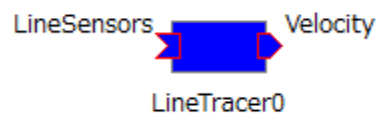
コンフィギュレーションパラメータ

名称	データ型	初期値	説明
JoyStick	文字列	一つ目に見つかったジョイスティック	使用するジョイスティック

LineTracer

概要

ライントレースを行うときの速度指令値を生成するコンポーネントである。ラインセンサの値を読み取り、センサの値に対して重み付け和を求めることで、**Zumo** ロボットの速度指令値を生成する。



データポート

名称	フローポート	データ型	説明
LineSensors	InPort	RTC.TimedULongSeq	ラインセンサの値
Velocity	OutPort	RTC.TimedVelocity2D	制御速度

コンフィギュレーションパラメータ

名称	データ型	初期値	説明
VX_Gain	文字列	[-0.005,0.05,0.1,0.05,-0.005]	速度の係数の配列
VA_Gain	文字列	[5.0,3.0,0.0,-3.0,-5.0]	角速度の係数の配列

9. 付録：Raspberry Pi A+を用いたシステム構築

9.1. はじめに

Raspberry Pi A+では、USB ポートが少なく、Zumo ロボットとは GPIO 接続を用いて、シリアル通信を行う必要がある。そのため、電圧変換ボードの取り付けを行う。また、Zumo 側から見た通信経路が異なるため、Zumo 用プログラムについても、GPIO 経由での通信プログラム群を使用する。

9.2. 用意するもの

- Raspberry Pi A+
- Zumo 32U4 Robot
- 2 電源 8 ビット幅双方向ロジックレベル変換モジュール（秋月電気通商販売）1 個
- ケーブル（見分けのつくように別々の色を使うことが望ましい）
- ユニバーサル基板（46 mm × 35 mm）1 枚
- ケーブル A, B 各 1 本
- ピンヘッダ 2.54 mm 間隔 1 列×4 ピン L 型 2 個
- QI コネクタ ハウジング 1×4 3 個
- QI コネクタ ハウジング 2×5 1 個
- QI コネクタ用ピンメス 16 個
- M3×15 ネジ 2 本
- M2.6×15 ネジ 4 本
- M2×25 ネジ 8 本
- M3 ナット 2 個
- M2.6 ナット 4 個
- M2 ナット 8 個（うち 4 個は Zumo に付いているものを使用する）
- スペーサー（M2.6 ネジ用、高さ 3 mm） 6 個（3D プリンタ用データが本プロジェクトのホームページにて公開されている）
- 3D プリンタで出力した治具 B-1, B-2 各 1 個（CAD データおよび 3D プリンタ用データが本プロジェクトのホームページにて公開されている）

9.3. 電圧変換モジュールの作成

電圧変換モジュールの回路図および作成例を図 22, 図 23 に示す。この例では、ピンヘッダとロジックレベル変換モジュールをユニバーサル基板の表裏になるように取り付けている。

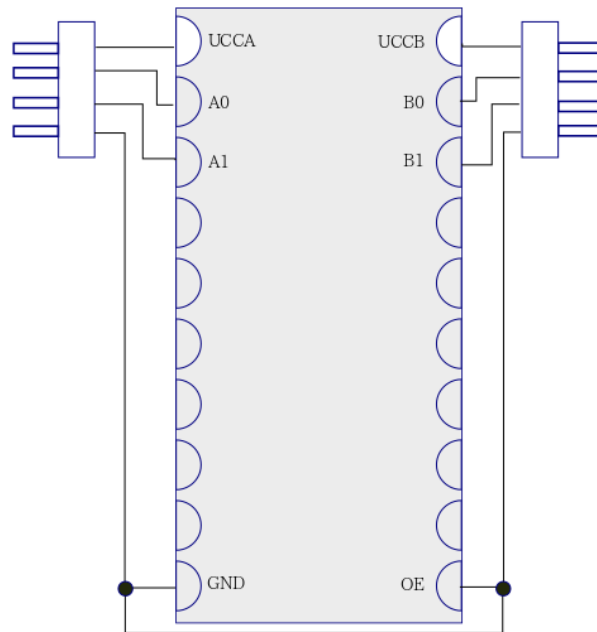


図 22 : 電圧変換モジュールの回路図

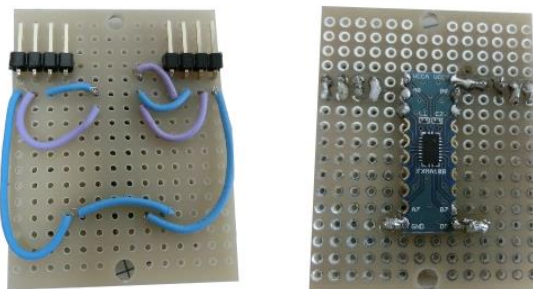


図 23 : 電圧変換モジュールの作成例（表裏）

9.4. 治具のノイズ対策処理

治具 B-2 の 2 面（図 24 に示す二箇所）にノイズ対策のアルミ箔テープを貼る．

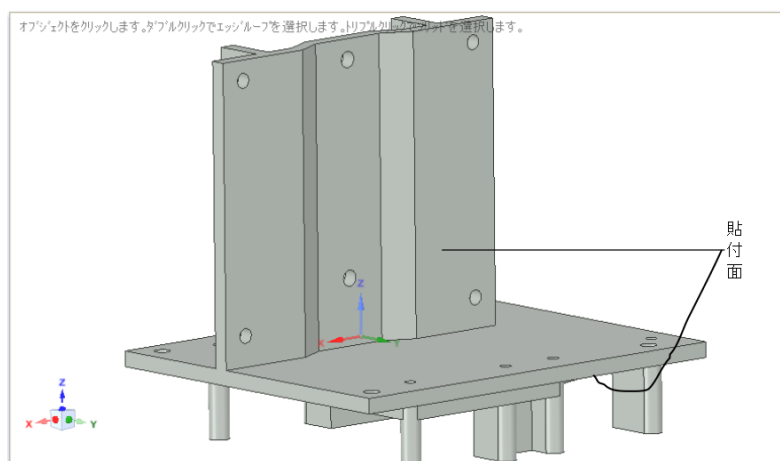


図 24 : アルミ箔テープによりノイズ対策を施した治具 B-2

9.5. 治具の取り付け

Zumo の液晶ディスプレイが接続されていたピンコネクタにピンヘッダ 2.54 mm 間隔 1 列×4 ピン L 型を挿し、ケーブル A を接続する (図 20 左)。この部分の回路図を図 25 右に示す。



図 25 : Zumo へのケーブル接続

次に、治具 B-1, B-2 を Zumo の基板上にネジで固定する。図 26 左のように先に治具 B-1 を基板に載せ、その上に図 26 右のように治具 B-2 を載せてネジで固定する。ブレードが固定されていた 2 か所 (図 27) では治具から Zumo に向かってネジを通し、それ以外の 6 か所では Zumo の電池ボックスの内側から治具に向かってネジを通す。

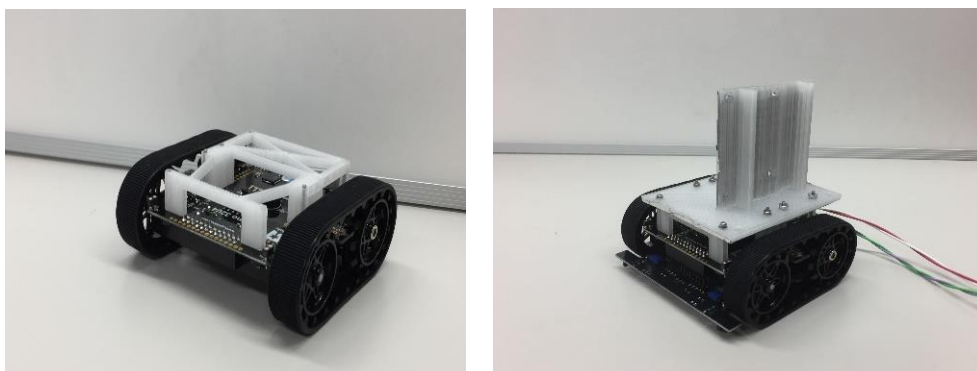


図 26：治具 B-1（左図），治具 B-2（右図）の取り付け

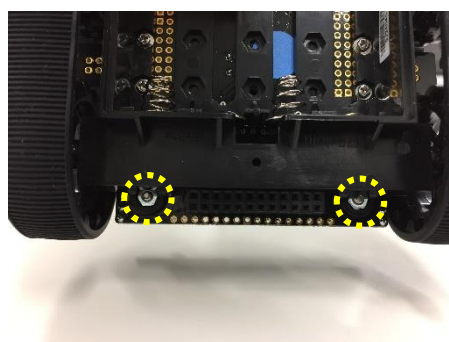


図 27：治具側からネジを挿入する部分

ネジの取り付け完了後，赤外線センサアレイを元の位置のピンコネクタに挿す．

9.6. 電圧変換モジュールの取り付け

電圧変換モジュールを治具 B-2 のアルミ箔テープを貼っていない面に取り付ける（図 28）．電圧変換モジュールの端子を取り付けた面が外側になるようにネジで固定する．

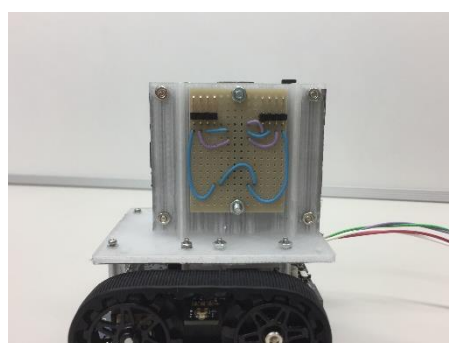


図 28：電圧変換モジュールを治具 B-2 に取り付けた様子

9.7. Raspberry Pi の取り付け

治具 B-2 のアルミ箔テープを貼った面に Raspberry Pi A+ を取り付ける (図 29).
Raspberry Pi と台座の間に必ずスペーサーを挟みネジで固定する.



図 29 : 治具 B-2 へ Raspberry Pi を取り付けた様子

9.8. ケーブルの接続

まず, 電源変換モジュールの左側のコネクタにケーブル A (Zumo に繋がっている) を接続する. 次に, 電源変換モジュールの右側のコネクタにケーブル B を接続し, ケーブル B のもう一方を Raspberry Pi A+ のピンヘッダに接続する. 図 25 に接続した様子を示す. 図 31 は電圧変換モジュールと Zumo および Raspberry Pi A+ との接続関係を図解したものである.

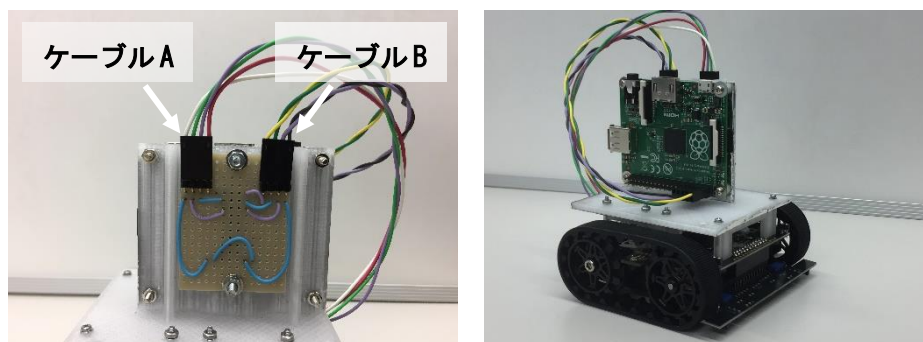
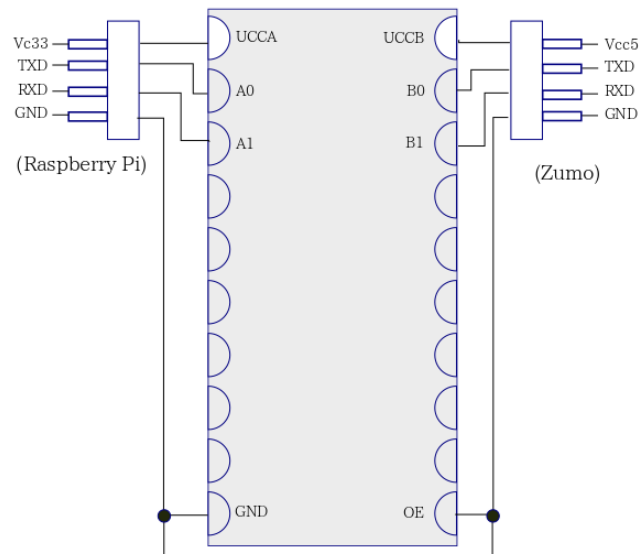
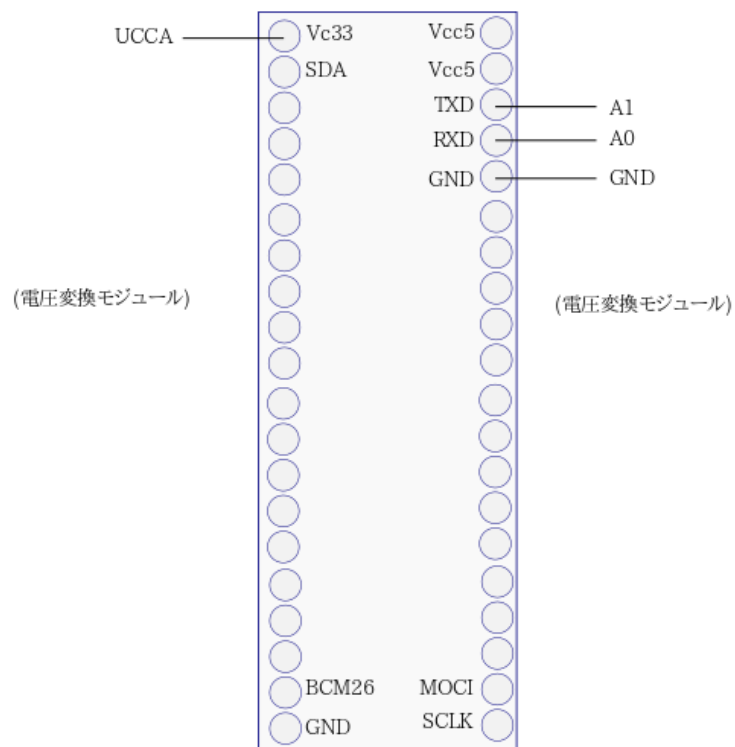


図 30 : Zumo と Raspberry Pi A+ の接続



(a) 電圧変換モジュール



(b) Raspberry Pi A+

図 31 : 電圧変換モジュールと Zumo および Raspberry Pi A+との接続

以上で組み立ては完了である.

9.9. シリアル通信の設定

Raspberry Pi でシリアル通信を可能にするために設定を行う⁵。まず、Raspberry Pi にログインする。そして、コマンドプロンプトが表示されたら、`startx` コマンドにより GUI 環境にする。次に、端末を起動し、以下の作業を行う。

(1) /boot/cmdline.txt のバックアップ

```
$ sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt
```

(2) /boot/cmdline.txt の編集

以下のコマンドは `nano` というエディタの使用を想定しているが、好みに応じて `vi` を使用してもよいし、`gedit` をインストールして使用してもよい。

```
$ sudo nano /boot/cmdline.txt
```

変更前：

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200  
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait
```

変更後：

```
dwc_otg.lpm_enable=0 rpitestmode=1 console=tty1 root=/dev/mmcblk0p2  
rootfstype=ext4 rootwait
```

(3) OS の再起動

(4) /etc/inittab のバックアップ

```
$ sudo cp /etc/inittab /etc/inittab_backup.txt
```

(5) /etc/inittab の編集

```
$ sudo nano /etc/inittab
```

変更前：

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

変更後：

```
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

⁵ 参考：<http://nanicananica.blog.fc2.com/blog-entry-14.html>

(6) Raspberry Pi のシャットダウン

以上で Raspberry Pi の設定は完了である.

10. 更新情報

- 2016 年 10 月 31 日 初版
- 2016 年 12 月 10 日 Raspberry Pi 3 を標準とするシステム構成に改訂
- 2017 年 12 月 2 日 タクタイルスイッチによる管理を中心としたシステム構成に改訂