

目次

第 1 章 環境構築	1
1.1 SOLID の環境構築	1
1.1.1 SOLID-OS の導入	1
1.1.2 SOLID-IDE の導入	2
1.2 Raspberry Pi OS の設定	4
1.3 ROS の導入	7
1.4 Rosmaster X3	8
1.4.1 ROS ロボットコントロールボード	9
参考文献	13

第1章 環境構築

1.1 SOLID の環境構築

SOLID for Raspberry Pi の環境を構築するにはターゲットである Raspberry Pi への SOLID-OS の導入と Windows11 上に開発環境である SOLID-IDE の導入を行う必要がある。

1.1.1 SOLID-OS の導入

Raspberry Pi で SOLID-OS を動作させるため、以下の手順で導入する [?]

手順 1-1 イメージファイルのダウンロード

GitHub 上の京都マイクロコンピュータ社の solid-rapi4 ダウンロードページ¹から SOLID-RPi4B-1.2.1-raspios-bullseye-lite.img.xz をダウンロードする。

手順 1-2 イメージファイルの選択

Raspberry Pi Imager で OS の選択画面より「カスタムイメージを使う」を選択し、ダウンロードしたイメージファイルを設定する。

手順 1-3 OS の初期設定

ウインドウ右下の設定ボタンより、詳細な設定を開く。図 1.1 に示すように、以下の項目を設定する。

- ホスト名：接続先の指定で使用する。
- SSH を有効化する：SSH で接続するため有効化する。
- ユーザ名とパスワードを設定する：設定することで初回起動時の設定を省略する。
- Wi-Fi を設定する：Wi-Fi 経由で接続する場合設定する。
- ロケールを設定する：使用するキーボードに合わせてキーボードレイアウトを変更する。

¹<https://github.com/KyotoMicrocomputer/solid-rapi4/blob/main/doc/download.md>

手順 1-4 イメージファイルの書込み

書き込むボタンをクリックし、SD カードへの書込みを開始する。

手順 1-5 起動

イメージファイルを書き込んだ SD カードを Raspberry Pi の SD カードスロットに挿入し、電源を接続する。2 回の再起動ののち、Raspberry Pi OS Lite 版の CLI でログインプロンプトが表示される。

1.1.2 SOLID-IDE の導入

開発環境である SOLID-IDE の導入と設定を以下の手順で行う [?]

手順 2-1 インストーラのダウンロード

GitHub 上の京都マイクロコンピュータ社の solid-rapi4 リポジトリ²から SOLID-RPI4B-IDE-1.0.5.exe をダウンロードする。

手順 2-2 SOLID-IDE インストール

インストーラを起動し、以下の手順でインストールを行う。

1. Visual Studio Shell 2013 (Isolated) のインストールが要求されるのでインストール画面へ進む。
2. ライセンスへの同意をしインストールを行う。
3. SOLID インストーラ画面にて「ライセンス使用許諾に同意します。」にチェックを入れ、インストールボタンをクリックする。
4. PARTNER-IDE インストーラ画面にて指示に従って SOLID-IDE のインストールを進める。
5. PARTNER-Jet2 インストーラ画面にて指示に従って PARTNER Debugger のインストールを進める。
6. exeGCC インストーラ画面にて指示に従って exeGCC のインストールを進める。

手順 2-3 Visual Studio 2013 のアップデート

Visual Studio 2013 のアップデートが要求されるので以下の手順でアップデートを行う。

²<https://github.com/KyotoMicrocomputer/solid-rapi4/blob/main/doc/download.md>

1. Visual Studio 2013 Update 5 のダウンロードページ³を開き、Microsoft アカウントでログインする。
2. Visual Studio 2013 Update 5 の DVD (ISO ファイル) を選択し、ダウンロードする。
3. ライセンスに同意し、INSTALL をクリックする。

手順 2-4 ライセンスの有効化

SOLID-IDE から SOLID-OS に接続するためにはライセンスの有効化を行う必要がある。

1. SSH 鍵ペアの生成

ホスト PC で以下のコマンドを実行し、公開鍵と秘密鍵を生成する。メッセージが表示されるので Enter を押す。

```
> ssh-keygen -t ed25519
```

2. 公開鍵の転送

生成した id_ed25519.pub を SCP コマンドを用いて Raspberry Pi へ転送する。ここで、ユーザ名 pi、ホスト名 raspberrypi は 4.1.1 項で設定したユーザ名とホスト名にする。

```
> cd C:\Users\<Username>\.ssh  
> scp id_ed25519.pub pi@raspberrypi:
```

3. 公開鍵の書き込み

Raspberry Pi へログインし、転送した id_ed25519.pub を authorized_keys ファイルに追加する。

```
> ssh pi@raspberrypi  
  
$ mkdir .ssh  
$ chmod 700 .ssh  
$ cat id_ed25519.pub >> .ssh/authorized_keys  
$ chmod 600 .ssh/authorized_keys  
$ rm id_ed25519.pub
```

³https://my.visualstudio.com/Downloads?q=visual%20studio%202013&wt.mc_id=o~msft~vscom~older-downloads

4. ライセンスの取得

SOLID-IDE 内の Raspberry Pi Selector からライセンスを取得する。SOLID-IDE 起動時に自動で Raspberry Pi Selector が起動する。また、画面上部のライセンスタブから起動できる。

Raspberry Pi が接続された状態で、新規をクリックし以下の項目を設定する。以後、接続先を選択し、右上の接続をクリックすることで接続する。

- 表示名：任意の名前
- ホスト名：接続先のホスト名、または IP アドレス
- ユーザ名：ログインに使用するユーザ名

1.2 Raspberry Pi OS の設定

SOLID の提供する Raspberry Pi OS は Lite 版であり設定が必要である。以下の手順で設定を行う。

手順1 ネットワークへの接続

外部からパッケージをダウンロードするため、インターネットへの接続が必要である。学内ネットワークへ接続する手段を以下に示す。

1. ネットワーク設定ファイルの編集

設定ファイルを2つ、以下の内容に編集する。identity にはユーザ名、password にはパスワードを設定する。

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=JP

network={
    ssid="eduroam"
    scan_ssid=1
    key_mgmt=WPA-EAP
    eap=PEAP
    identity="xxxxx"
    password="yyyyy"
    phase2="MSCHAPV2"
}
```

```
$ sudo nano /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
# source /etc/network/interfaces.d/*
allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

2. 設定の反映

以下のコマンドを実行しネットワーク設定を反映させる。

```
$ sudo killall wpa_supplicant
$ rfkill unblock wifi
$ sudo ip link set wlan0 up
$ sudo wpa_supplicant -i wlan0 -c /etc/wpa_supplicant/
wpa_supplicant.conf -B
$ history -c
```

手順2 GUI のインストール

Lite 版であるが ROS の視覚化ツール等を利用するため、GUIが必要となる。
以下のコマンドを実行し、ウインドウシステムである Xorg Display Server と Raspberry Pi Desktop GUI のインストールを行う。

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install -y xserver-xorg raspberrypi-ui-mods
$ sudo apt-get install --no-install-recommends xinit
```

手順3 VNC のインストール

リモートデスクトップを実現するため、VNC サーバの以下の手順でインストールと設定を行う。

1. VNC サーバのインストール

VNC サーバとして RealVNC サーバをインストールする。

```
$ sudo apt install realvnc-vnc-server
```

2. VNC の有効化

raspi-config にて VNC を有効化する。

```
$ sudo raspi-config  
  
3 Interface Options  
I2 SSH  
<Yes>
```

3. パスワード認証の設定

設定ファイルを編集しパスワード認証を行う設定する。

```
$ sudo nano /etc/vnc/config.d/common  
  
Authentication=VncAuth
```

以下のコマンドで任意のパスワードを入力しパスワードの登録を行う。

```
$ sudo vncpasswd -service
```

4. GUI 起動の設定

GUIで起動するための設定を行う。HDMI を接続していない状態でも起動されるように起動設定ファイル/boot/config.txt にて hdmi_force_hotplug=1 のコメントを外す。

```
$ sudo nano /boot/config.txt  
  
hdmi_force_hotplug=1
```

また、raspi-config にて GUI 起動に変更する。

```
$ sudo raspi-config  
  
1 System Options  
S5 Boot / Auto Login  
B3 Desktop もしくは B4 Desktop Autologin
```

すべての設定後、再起動せずシャットダウンを行う。

```
$ sudo shutdown -h now
```

手順4 画面サーバの起動設定の初期化

ディスプレイに接続した状態で GUI を起動すると、画面サーバの起動でエラーが発生する。一度イメージファイルを作成し、再度書き込むことで画面サーバの起動設定を初期化する。

1. バックアップツールである Win32 Disk Imager をダウンロードページ⁴からダウンロードする。
2. Win32 Disk Imager を起動する。SD カードを挿入し、Device から SD カードのドライブを選択する。
3. Image File に拡張子を .img として任意のファイル名を指定する。
4. 画面下部の Read ボタンをクリックしイメージファイルを作成する。
5. 4.1.1 項と同様の手順で、作成したイメージファイルを SD カードへ書き込む。

手順5 画面サイズの変更

SD カードを挿入し、ディスプレイに接続した状態で電源に接続すると GUI 画面で起動される。しかし、画面サイズが小さいため起動設定ファイルを編集しサイズを変更する。

```
$ sudo nano /boot/config.txt  
  
hdmi_group=2  
hdmi_mode=16
```

1.3 ROS の導入

Raspberry Pi OS への ROS の導入を行う。表??より、Raspberry Pi OS(Debian11) は Tier 3 サポートであり、Raspberry Pi OS で ROS を使用する場合、Docker コンテナでの動作が推奨されている。しかし Docker を用いた場合、ホストマシンとの通信が難しいため本研究では Raspberry Pi OS 向け ROS2 Humble パッケージを導入する。

以下の手順で導入を行う。

手順1 パッケージのインストール

以下のコマンドを実行し、パッケージをローカルに落としてからインストールを行う。

```
$ wget https://s3.ap-northeast-1.wasabisys.com/download-  
raw/dpkg/ros2-desktop/debian/bullseye/ros-humble-  
desktop-0.3.1_arm64.deb  
$ sudo apt install -y ./ros-humble-desktop-0.3.1_arm64.deb
```

[手順2 ビルドツールのインストール

ROS で使用するビルドツールをインストールする。

⁴<https://sourceforge.net/projects/win32diskimager/>


```
$ sudo pip install vcstool colcon-common-extensions
```

手順3 apt リポジトリの GPG キーの更新

ROS の apt リポジトリの GPG キーを以下のコマンドを実行し更新する。

```
$ sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
```

手順4 環境変数の登録

ROS は実行する際に ROS 環境変数が登録されている必要がある。以下のコマンドを実行し、ターミナル起動時に ROS 環境変数を自動的に読み込むよう設定する。

```
$ echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc  
$ source ~/.bashrc
```

本環境では推奨環境ではない Raspberry Pi OS に ROS を導入しているため、通常の方法ではビルド時に不具合が生じる。本環境での ROS パッケージのビルド方法を以下に示す。

- C++言語で作成された ROS ノードを含む ROS パッケージ

c++言語で作成した ROS ノードのビルドでは、ROS の CMake ベースのビルドシステムである ament_cmake が存在していないというエラーが発生する。手動での環境変数の設定等を行ったがエラーは解消できなかった。そのため本研究では Python によってノードを作成する。

- Python 言語で作成された ROS ノードを含む ROS パッケージ

Python 言語で作成した ROS ノードのビルドでは、通常のビルド方法では場合パーミッションエラーが発生する。そのため sudo コマンドをつけた以下のコマンドでビルドを行う必要がある。xxxxxx はビルドするパッケージ名である。

```
$ sudo colcon build --select-packages xxxxx
```

1.4 Rosmaster X3

本研究では Rosmaster X3 をターゲットとしてアプリケーションを作成した。後述する ROS ロボットコントロールボードや RPLiDAR を使用するためのコードであるため、Rosmaster X3 を用いなければ自律移動を行うことはできない。Rosmaster X3 は yahboom 社製のメカナム 4 輪ロボットである。Jetson NANO といった Jetson シリーズや Raspberry Pi 4B を用

いた、ROS によるロボット制御に関する教育を対象としている [1]。ロボットには LiDAR、深度カメラ、STM32 マイコンを搭載した ROS ロボットコントロールボードが搭載されている。外観を図 1.2 に、構成を図 1.3 に示す。

図 1.3 より、RaspberryPi は RPLiDAR と深度カメラ、STM32 マイコンの搭載された ROS ロボットコントロールボードと接続することで、外部センサによる環境データの取得や STM32 マイコンに接続された内部センサのデータを取得する。また、制御命令を送信することで STM32 マイコンに接続されたアクチュエータの制御を行うことが可能である。STM32 マイコンはエンコーダ、9 軸センサに接続しており、モータの回転数、加速度、ジャイロ、地軸データの取得を行うことが可能であるほか RaspberryPi からの制御命令に応じてモータの制御を行う。

1.4.1 ROS ロボットコントロールボード

ROS ロボットコントロールボードは、Rosmaster X3 に搭載されているドライバボードである。マイコンである STM32F103RCT6 が搭載されており、以下の機能を持つ。また、外観を図 1.4 に示す。

- 加速度、ジャイロ、地軸の 9 軸センサである MPU9250 の計測
- エンコーダの計測と回転数の自動カウント
- 4 チャンネルのエンコーダ付きモータの制御、サーボモータの制御
- ブザー、LED の制御
- 12V 駆動、他モジュールへの電源分配

通信は USB シリアルで行われ、STM32 マイコンへの送信データとしてモータの PWM 値、ブザーの ON/OFF、LED の配色等の制御データが送信可能である。また、制御データの送信後、一定間隔でエンコーダから計算された各モータの回転数と速度データ、9 軸センサの計測値が STM32 マイコンより送信される。

詳細な設定 X

カスタマイズオプション このセッションでのみ有効にする ▼

☒ ホスト名: solid .local

☒ SSHを有効化する

☒ パスワード認証を使う

☐ 公開鍵認証のみを許可する

ユーザーtestのためのauthorized_keys

☒ ユーザー名とパスワードを設定する

ユーザー名 test

パスワード: ●●●●

☐ Wi-Fiを設定する

SSID: eduroam

☐ ステルスSSID

パスワード:

☒ パスワードを見る

Wifiを使う国: GB ▼

☒ ロケール設定をする

タイムゾーン: Asia/Tokyo ▼

キーボードレイアウト: jp ▼

永続的な設定

図 1.1: Raspberry Pi Imager の設定

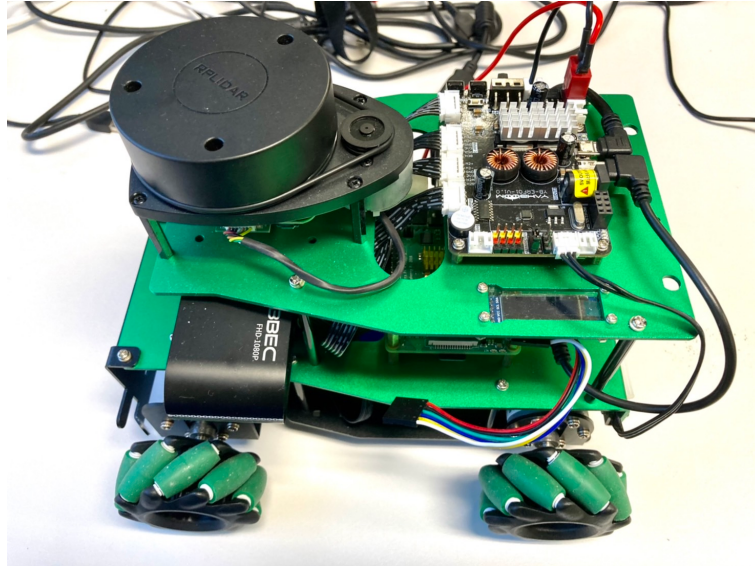


図 1.2: Rosmaster X3 の外観

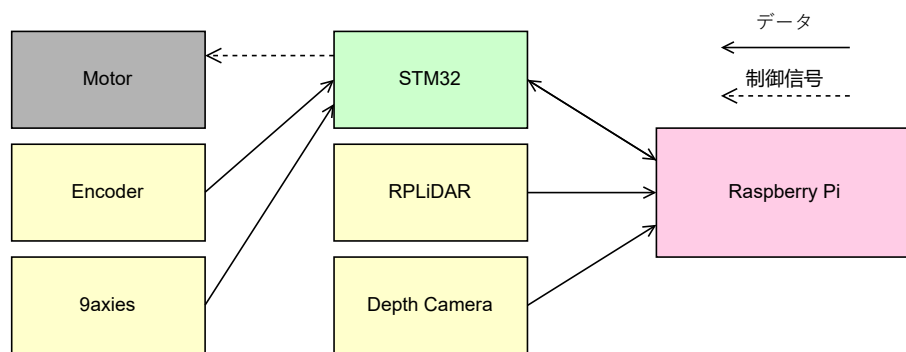


図 1.3: Rosmaster X3 の構成

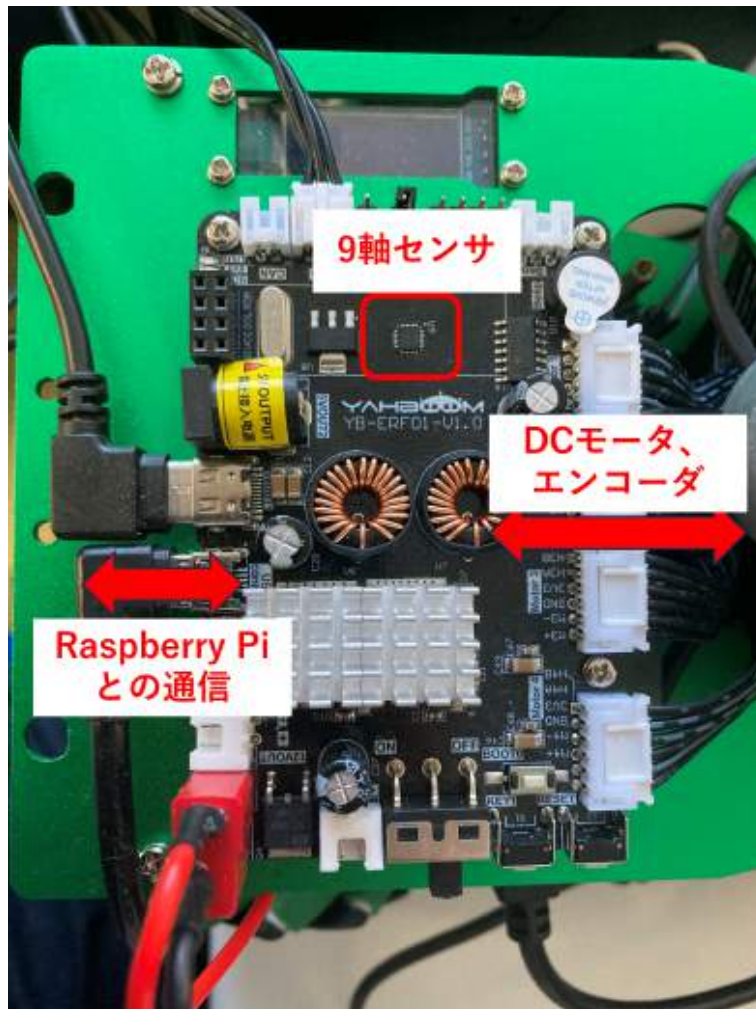


図 1.4: ROS ロボットコントロールボードの外観

参考文献

- [1] ROSMASTER X3, <http://www.yahboom.net/study/ROSMaster-X3>, 最終閲覧
日:2023/12/15