メガローバーVer.3.0
ROS 制御モード

取扱説明書
（2021.12.17）

Vstone®
ヴイストン株式会社

このたびは、メガローバーVer.3.0、メガローバーVer.3.0 屋外対応版をお買い求めいただき、誠にありがとうございます。本ドキュメントをよくお読みいただき、操作を行ってください。

目次

1　はじめに / 注意事項

本書は、メガローバーVer.3.0、メガローバーVer.3.0 屋外対応版を ROS で制御する際の取扱説明書兼チュートリアルです。メガローバー本体についての説明書である「メガローバーVer.3.0 取扱説明書」または「メガローバーVer.3.0 屋外対応版 取扱説明書」も併せてご確認いただき、注意事項を守り、安全に十分配慮してご使用ください。

本製品の使用にあたっては下記注意事項に従い、正しくご使用ください。

○　本製品を無許可で複製、再配布、再販することはできません。ただし、著作権法で認められた範囲における複製については許可されます。
○　本製品の対応環境は、ネイティブの Ubuntu 16.04 と ROS Kinetic または、Ubuntu 18.04 と ROS Melodic です。それ以外の環境での使用についてはサポート対象外となる他、それによって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。
○　本製品の制御を仮想環境上の ROS から行われた場合、タイムラグ等により安全な制御が行えない可能性があります。そのため、仮想環境からの制御は非推奨とさせていただいており、その使用によって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。
○　本製品の使用にあたっては、本製品に含まれない公開ライブラリを多数使用する必要がございます。本製品に含まれないライブラリについてはサポート対象外となる他、その使用によって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。
○　本製品を使用中にハードウェアに強い振動や衝撃が加わると、暴走に繋がる可能性がございます。衝突などによる強い振動や衝撃を加えないでください。
○　本製品を使用する PC はお客様にてご準備ください。Ubuntu のデバイスドライバの対応状況等により、一部の機能が正常に動作しない可能性がございますが、デバイス固有の問題についてはサポート対象外となる他、それによって生じたいかなる損害についても、製造元および販売元は何らの責任を負いません。

2 ROS について

2.1 ROS の概要

ROS（Robot Operation System）は、OSRF（Open Source Robotics Foundation）によって開発・メンテナンスされているロボット用のミドルウェアです。分散処理が求められる複雑なロボットシステムを制御できる性能を備えており、世界中の研究者や開発者が作成した豊富なライブラリを使用することができます。ロボット制御システムの作成を効率化できることから、人型ロボットから車両型ロボット、水上・水中ロボットやドローンに至るまで、幅広い分野で活用されています。

ROS の特徴のひとつが、BSD ライセンスに基づくオープンソースとして公開されており、誰でも開発に参加し貢献できることです。ROS には強力な開発者コミュニティが存在し、誰でも使用可能な 5000 以上のライブラリのほとんどは、OSRF ではなくコミュニティによって開発・メンテナンスされています。

ROS 本体が BSD ライセンスによって提供されているため、ROS を用いて開発した成果物は、商用利用することが可能です。ROS を用いて動作する様々なロボットが発売されており、メガローバーもそのひとつです。ただし、ライブラリによっては BSD ではないライセンスによって提供されているものも存在するため、商用利用ではご注意ください。

2.2 ROS が提供する機能

ROS によって提供される主要な機能について、説明します。

- メッセージ通信

  ROS を用いて構成されるロボットシステムは分散処理が基本となっており、ユーザは「ノード」と呼ばれるプログラムを複数立ち上げることでシステムを作成します。例えば、ゲームパッドで操作できる台車ロボットであれば、「ゲームパッドの入力値を取得するノード」、「入力値に従って移動指令を出すノード」、「移動指令をもとにモータを回転させるノード」などが必要となります。当然、ノード間で情報を通信する仕組みが求められます。各ノード間で、センサ入力値の情報やカメラの映像、制御指令値といったデータをやり取りするために、ROS では Pub/Sub 方式によるメッセージ通信が提供されています。開発者はわずか数行のコードにより、任意の情報をパブリッシュ（配信）したり、サブスクライブ（購読）したりすることができます。メッセージには、構造体のような型が定められているため、ROS ノード同士であれば互換性を気にする必要はありません。また、型は自作することもできます。

- パッケージ管理

  ROS のライブラリやプログラムは、パッケージという単位で管理されています。パッケージの中にはノードやその設定ファイル、起動スクリプトなどが含まれており、ユーザは使用したい機能を持つパッケージをインターネット上からダウンロードし、ローカル環境に組み込むことができます。パッケージの追加や削除といった操作は非常に簡単に行う

事ができます。また、ユーザが独自の制御プログラムを開発する際には、まずパッケージを作成し、その中で開発を行う事になります。

- デバイスドライバ

  ROS では様々なデバイスのドライバがパッケージの形式で提供されています。対応しているデバイスであれば、パッケージを導入し、デバイスを接続するだけで使用することができます。

- ハードウェア抽象化

  ROS による制御システムは複数のノードによる分散処理によって動作します。これにより、ハードウェアが異なるロボットでも、上流の制御システムは同じものが使用できます。例えば未知環境の地図を作成する SLAM 機能を提供するパッケージ「gmapping」では、周囲の障害物の情報をレーザー光を用いた測距センサである LRF で取得することになっています。この LRF から情報を取得する部分は、LRF の種類に応じて異なるパッケージが提供されているため、ユーザは使用したい LRF を自由に選択することが可能です。そして開発者は、デバイス毎の違いを意識しなくても汎用的に使用可能な制御システムを作成することができます。

- ライブラリ

  ROS では、5000 を超える公開ライブラリを使用することができます。それらはデバイスドライバのようなものから、経路計画や動作生成といったものまで様々です。

- 視覚化ツール

  ROS には、いくつかの視覚化ツールが存在しており、ロボットの操縦 UI やデバッグ等のために活用されています。中でも「Rviz」は強力なツールです。3D 表示機能を持つこのツールは、実に多くのパッケージで情報を表示するために使われています。表示情報のカスタマイズが容易ですので、ユーザオリジナルの UI を作成することも容易です。

2.3 ROS に関する情報の集め方

ROS を用いた開発を行う際には、使用するパッケージの情報など、様々な情報が必要になります。ROS やその開発に関する情報は書籍から集めることもできますが、ここではインターネットから情報を集める際に参考になるサイトをいくつかご紹介します。

- ROS Wiki - http://wiki.ros.org/

  ROS の公式 Wiki です。ROS のインストール方法からチュートリアル、各公開パッケージの情報まで、様々な情報が公開されています。ただ、パッケージの情報などが更新されないまま古くなっていることもありますので、ご注意ください。

- ROS Wiki(ja) - http://wiki.ros.org/ja

  ROSWiki の日本語訳版です。現在も有志による精力的な翻訳作業が行われていますが、古い情報も多いので、英語版とあわせて確認した方がよいでしょう。

- ROS Answers - https://answers.ros.org/questions/

  ROS の Q&A フォーラムです。パッケージを使用した際のエラーの解消法など、様々な情報が蓄えられています。

3　メガローバーのセットアップ

メガローバー本体側のセットアップを行います。必ず「メガローバーVer.3.0 取扱説明書」または「メガローバーVer.3.0 屋外対応版 取扱説明書」をご確認いただいたうえで作業を行ってください。

メガローバーを ROS で制御するためには、メガローバーの制御基板である VS-WRC058 を、USB 有線シリアルまたは、Wi-Fi を用いて PC 等の ROS が動作するデバイスと接続する必要があります。以下の手順に従って、メガローバー側のセットアップを行ってください。

① 「メガローバーVer.3.0 取扱説明書」に従い、メガローバーの制御基板 VS-WRC058 を Arduino IDE で開発できるよう、環境をセットアップしてください。

② Arduino IDE メニューのファイル＞スケッチ例＞vs_wrc058_megarover から、ご利用環境に合わせて megarover3_common または megarover3_outdoor を開いてください。

③ スケッチ内の指示に従い、「WRC_WIRELESS_MODE」,「WRC_ROS_SERIAL_MODE」を適切に設定してください。また Wi-Fi を用いて接続する場合、メガローバーを接続するデバイスの IP アドレスおよび接続ポートを ROS 接続設定の項で設定してください。接続ポートは通常 11411 です。

④ setup()関数内のモータ制御パラメータ std_mortor_param[]について、必要に応じ各パラメータを適切に設定してください。

⑤ 更新したスケッチをメガローバーに書き込んでください。

スケッチ書き込み後、メガローバーが VS-C3 を使って動作することを確認してください。

※注意：Wi-Fi を用いて接続する場合、ROS との通信が確立するまで、Xbox コントローラを使った動作がスムーズに動作しません。ROS ライブラリの仕様です。

4 ROS のインストール

ROS Kinect または ROS Melodic を Ubuntu にインストールする方法を説明します。ここで述べる手順は、ROS Wiki で紹介されているインストール方法から一部を抜粋したものです。より詳しい情報が欲しい方は、下記ページを確認してください。

ROS Kinetic の Ubuntu へのインストール - http://wiki.ros.org/ja/kinetic/Installation/Ubuntu

ROS Melodic の Ubuntu へのインストール - http://wiki.ros.org/melodic/Installation/Ubuntu

ROS のファイルはインターネットから取得するため、インターネット接続が必要です。PC をインターネットに接続してください。ネットワークの接続制限があったり、プロキシが設定されている環境では、ファイルのダウンロードに失敗することがあります。

① 端末（シェル）を立ち上げる

ランチャー上部の「コンピュータの検索」から“端末”を検索するか、ショートカットキー“Ctrl+Alt+T”を使用して、新しい端末（シェル）を起動します。

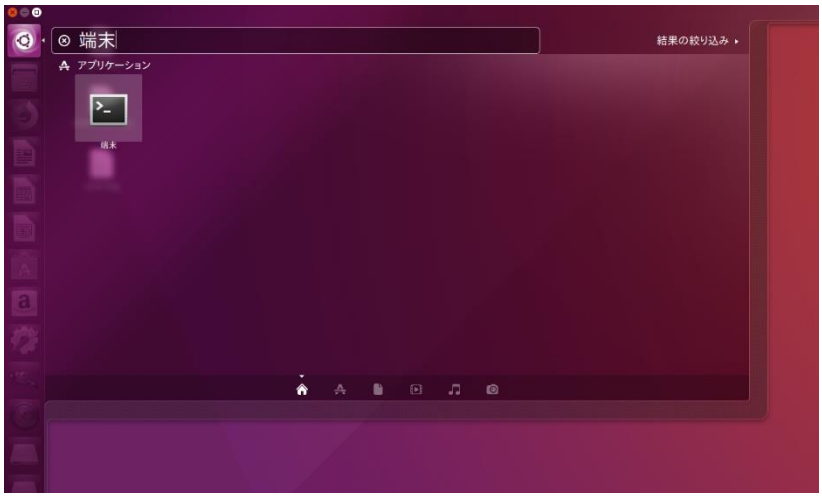

図 4-1 コンピュータの検索

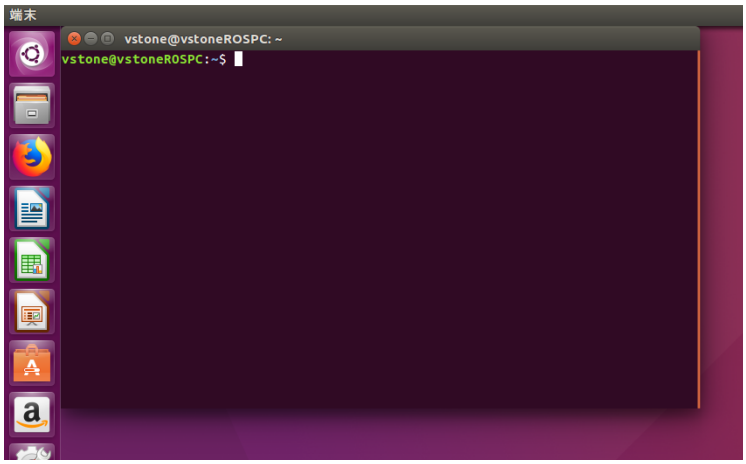

図 4-2 端末（シェル）

② sources.list を設定する

以下の青枠内のコマンドを端末にコピー&ペーストし、エンターキーを押して実行してください。コマンドラインへの貼り付けは右クリックまたは“Ctrl+Shift+V”で行えます。ひとつの青枠内にひとつのコマンドを書いていますので、複数行に分かれていてもまとめてコピーしてください。コピー&ペーストできない場合は手で入力してください。

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

③ curl のインストール

```
sudo apt install curl
```

④ 鍵を設定する

```
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

⑤ インストール

先にパッケージインデックスをアップデートしておきます。

```
sudo apt update
```

ROS Kinect をインストールします。ネットワーク環境によってはかなり時間がかかります。

```
sudo apt install ros-kinetic-desktop-full
```

ROS Melodic をインストールする場合は、以下のコマンドを使用してください。

```
sudo apt install ros-melodic-desktop-full
```

ROS のライブラリをバイナリでインストールする場合、上記のように「ros-バージョン-パッケージ名」で指定します。以下、バージョンに相当する箇所は、kinetic または melodic をご利用の環境に合わせて指定してください。

⑥ rosdep の初期化

以下の 2 つのコマンドを順に実行してください。

```
sudo rosdep init
```

```
rosdep update
```

⑦　環境設定

パスの設定を行います。Kinetic の場合、以下の 2 つのコマンドを順に実行してください。

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

```
source ~/.bashrc
```

Melodic の場合、コマンドは以下のようになります。

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

```
source ~/.bashrc
```

⑧　パッケージ構築のための依存ツールのセットアップ

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

以上で、ROS Kinect 本体のインストールは完了です。ROS の基幹プログラムを立ち上げるコマンド“roscore”を使って、起動することを確認しておきましょう。

```
roscore
```

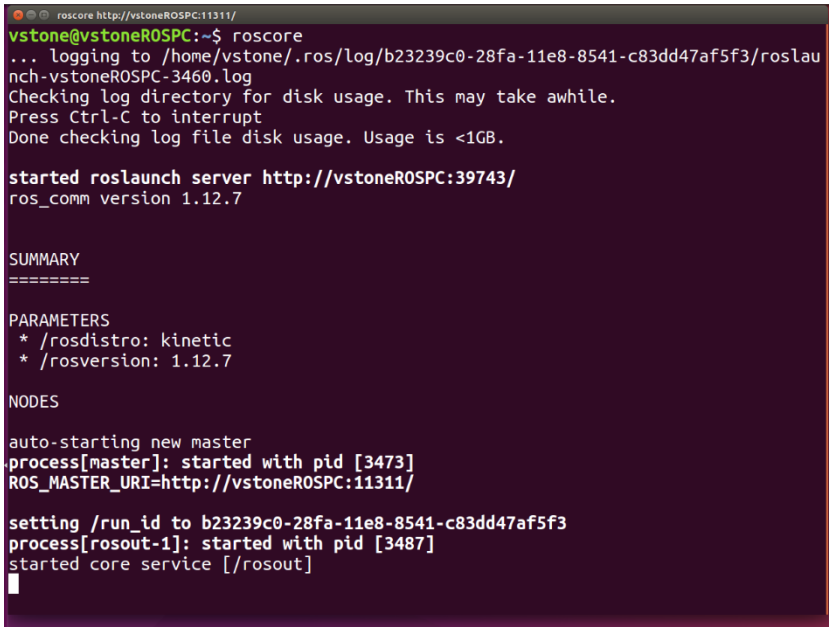

図 4-3 roscore 起動時のシェル

上のようなメッセージが表示されれば、ROS のインストールは正常に行われています。

roscore は、ROS の各サービスを提供する基幹プログラムです。ROS を使用する際には、必ずこの roscore を起動しておく必要があります。

5　メガローバー用サンプルパッケージのセットアップ

メガローバーを ROS で制御するサンプルパッケージのセットアップを行います。下記の手順に従って作業を行ってください。

5.1 catkin ワークスペースの作成

ROS のビルドシステムである catkin のためのワークスペースを作成します。この作業は、ROS Wiki に掲載されているチュートリアル「ROS 環境のインストールとセットアップ」（http://wiki.ros.org/ja/ROS/Tutorials/InstallingandConfiguringROSEnvironment）にも記載されています。

端末を起動して、以下のコマンドを順番に実行してください。

・フォルダの作成

```
mkdir -p ~/catkin_ws/src
```

・作成した src フォルダに移動

```
cd ~/catkin_ws/src
```

・ワークスペース作成

```
catkin_init_workspace
```

これでワークスペース「catkin_ws」ができあがりました。ワークスペースの src フォルダ内にパッケージフォルダを配置していくことができます。作成したばかりの src フォルダは空ですが、~/catkin_ws で以下のコマンドを実行することで、ワークスペースをビルドすることができます。

・(src フォルダに居る場合) ~/catkin_ws に移動

```
cd ~/catkin_ws
```

・ワークスペースをビルド

```
catkin_make
```

このワークスペース内で作成したパッケージを動作させるためには、ワークスペースをオーバーレイする必要があります。オーバーレイについては ROS Wiki にある catkin のチュートリアル「workspace_overlaying」(http://wiki.ros.org/catkin/Tutorials/workspace_overlaying)を確認してください。オーバーレイを行うためには、~/catkin_ws で次のコマンドを実行します。

```
source devel/setup.bash
```

ただしこの状態では、端末を新しく起動するたびにオーバーレイの作業を行う必要があります。端末起動時に自動的にオーバーレイが実行されるようにするためには、以下のコマンドを用いて設定を.bashrc に書き込みます。

```
echo "source /home/ユーザ名/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

以上で、ワークスペースの作成は完了です。

5.2 サンプルパッケージのダウンロードとビルド

GitHub で公開しているサンプルパッケージをダウンロードしビルドします。以下のコマンドを順番に実行してください。

・~/catkin_ws/src フォルダに移動します

```
cd ~/catkin_ws/src
```

・GitHub よりサンプルパッケージのソースコードをクローンします

```
git clone https://github.com/vstoneofficial/megarover3_ros.git
```

・ビルドします、表示が[100%]になればビルド完了です。

```
cd ~/catkin_ws
```

```
catkin_make
```

ROS の公開ライブラリの多くは GitHub でソースコードを公開しています。GitHub で公開されているものについては、同じ手順でワークスペースに追加、ビルドして使用することができます。

5.3 サンプルパッケージの内容について

メガローバーの ROS サンプルパッケージ「megarover3_ros」の内容について簡単に説明します。

- ノード

  megarover3_ros には、「joycon」と「pub_odom」という 2 つのノードが存在します。「joycon」は、ゲームパッドなどのジョイスティックを使ってメガローバーを走行させられるノードです。「pub_odom」は、メガローバーのホイールエンコーダ値を基にオドメトリを出力するノードです。

  それぞれのソースコードは megarover3_ros/src に存在します。ソースコードを書き換えてビルドすれば、機能を変更することができます。

- launch ファイル

  launch フォルダ内に存在する拡張子が「launch」のファイルは、ROS で使用できる非常に便利なファイルです。ROS で制御システムを構築するためには数多くのノードを起動する必要がありますが、それをひとつひとつ手動でやるのは大変です。launch ファイルを使えば複数のノードを同時に起動することができます。

  他にも、パラメータの設定や remap 機能など、便利な機能を使用することができますので、積極的に利用しましょう。

  megarover3_ros の launch フォルダ内には、サンプルプログラムを実行するための複数の launch ファイルが格納されています。各 launch ファイルについては、7 章以降で説明します。

- configuration ファイル

  configuration_files フォルダ内に存在する、拡張子が「yaml」や「lua」のファイルです。サンプルプログラム実行時に使用する様々なパラメータ等について記載された設定ファイルです。

- map ファイル

  navigation サンプルで使用する map ファイルを格納するフォルダです。初期状態では参考用の map ファイルが入っています。

- CMakeLists.txt と package.xml

  CMakeLists.txt と package.xml は ROS パッケージに必須な、catkin の設定ファイルです。詳しくは下記のサイトを参照してください。

  CMakeLusts.txt: http://wiki.ros.org/catkin/CMakeLists.txt
  Package.xml: http://wiki.ros.org/catkin/package.xml

5.4 パラメータについて

ROS を用いて SLAM や自律移動を行う場合、車両の大きさや、旋回中心とセンサとの位置関係といったパラメータは大変重要です。これらのパラメータが誤って設定されていた場合は、良好な動作結果が得られない可能性があります。

配布しております megarover3_ros はメガローバーVer.3.0 を基準にパラメータが設定されています。よって、お客様によってセンサ配置や向きの変更といった改造を行われた場合、パラメータの調整を行っていただく必要があります。

パラメータは環境によって最適値が変化するものが多いため、お客様の環境に合わせて調整を行ってください。なおパラメータの調整についてはサポート対象外となります。

車両サイズおよび LRF と旋回中心の位置関係について記載があるのは以下のファイルです。

- configuration_files/megarover_costmap_common_params.yaml
- launch/gmapping.launch
- launch/megarover_move_base_dwa.launch
- launch/megarover_move_base.launch

6 メガローバーと ROS の接続

メガローバーと ROS を接続し、メッセージ通信が行えるようにします。以下の手順に従って、作業を行ってください。

6.1 rosserial のインストール

メガローバーの制御基板と PC は USB-シリアル通信で接続します。rosserial は USB-シリアル通信で接続したデバイスと ROS との通信をサポートするライブラリです。rosserial を使用することで、メガローバーは ROS のメッセージ通信に対応することができます。

以下のコマンドを実行して rosserial をインストールしてください。ROS バージョンには、kinetic または melodic を入力してください。

```
sudo apt install ros-ROS バージョン-rosserial
```

もしくは、rosserial をソースからビルドしてインストールすることもできます。

```
cd ~/catkin_ws/src
```

```
git clone https://github.com/ros-drivers/rosserial.git
```

```
cd ~/catkin_ws
```

```
catkin_make
```

6.2 urg_node のインストール

LRF オプションとして北陽電機社製レーザーレンジファインダ URG-04LX-UG01 を搭載されている場合は、本項に従ってセットアップを行ってください。LRF オプション(TG30)として YDLiDAR TG30 を搭載されている場合は、ydlidar_ros のインストールの項に従ってセットアップを行ってください。

URG-04LX-UG01 を使用するためには、urg_node パッケージをインストールする必要があります。urg_node は、SCIP 規格で通信を行う LRF のデータを ROS で活用するためのデバイスドライバの役割を果たします。

以下のコマンドを実行して urg_node をインストールしてください。

```
sudo apt install ros-ROS バージョン-urg-node
```

または、6.1 項と同様の手順で、以下のコマンドによりソースをクローンすることで、ソースか

らビルドしてインストールすることもできます。

```
git clone https://github.com/ros-drivers/urg_node.git
```

6.3 ydlidar_ros のインストール

LRF オプション(TG30)として YDLiDAR TG30 を搭載されている場合は、本項に従ってセットアップを行ってください。LRF オプションとして北陽電機社製レーザーレンジファインダ URG-04LX-UG01 を搭載されている場合は、urg_node のインストールの項に従ってセットアップを行ってください。

TG30 を使用するため位は、ydlidar_ros パッケージをインストールする必要があります。ydlidar_ros は、YDLiDAR ブランドの各 LiDAR を ROS で活用するためのデバイスドライバの役割を果たします。

以下のコマンドで ydlidar をソースからビルドしてインストールしてください。

```
cd ~/catkin_ws/src
```

```
git clone https://github.com/YDLIDAR/ydlidar_ros
```

```
cd ~/catkin_ws
```

```
catkin_make
```

なお、使用する YDLiDAR の数が 1 個の場合は、ydlidar_ros に付属する以下のスクリプトを実行することで TG30 のデバイスアドレスを固定することができます。
シェルを新しく起動し、以下のコマンドを実行してください。

```
roscd ydlidar_ros/startup
```

```
sudo chmod 777 ./*
```

```
sudo sh initenv.sh
```

6.4 メガローバーと ROS の接続

作業を行う前に、メガローバーを十分充電しておいてください。メガローバーと PC を USB ケーブルもしくは Wi-Fi で接続した後に、rosserial を使って通信を行います。以下の手順に従って作業を行ってください。なおこの作業は、メガローバーと ROS を接続するたびに行う必要があります。

〔制御基板と ROS の接続〕

メガローバーの制御基板 VS-WRC058 と ROS を接続します。有線シリアルを使用する場合は手順①から、Wi-Fi を使用する場合は手順③から実施してください。

① VS-WRC058 のデバイス名の確認(有線シリアル使用時)

有線シリアル使用時には VS-WRC058 のデバイス名を設定する必要があります。Wi-Fi で接続する場合は手順③に進んでください。

本体前部に取り付けられたメガローバーの制御基板 VS-WRC058 と PC を USB ケーブルで接続します。続けて、端末で以下のコマンドを実行し、VS-WRC058 のデバイス名を確認します。

```
dmesg
```

```
654] usb 1-1: new full-speed USB device number 5 using xhci_hcd
759] usb 1-1: New USB device found, idVendor=0403, idProduct=6015
762] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
764] usb 1-1: Product: FT231X USB UART
766] usb 1-1: Manufacturer: FTDI
767] usb 1-1: SerialNumber: DO00B2HY
502] usbcore: registered new interface driver usbserial
516] usbcore: registered new interface driver usbserial_generic
526] usbserial: USB Serial support registered for generic
418] usbcore: registered new interface driver ftdi_sio
605] usbserial: USB Serial support registered for FTDI USB Serial Devi

556] ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
838] usb 1-1: Detected FT-X
655] usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
oneROSPC:~$
```

図 6-1　制御基板のデバイス名確認

図 6-2 がコマンドの応答例です。4 行目の記述に「FT231X USB UART」とあります。これは VS-WRC058 上の USB シリアル変換チップが認識されたことを示しています。VS-WRC058 の名称は応答に現れませんのでご注意ください。14 行目の記述のうち「ttyUSB0」が VS-WRC058 のデバイス名となります。

デバイス名は接続する PC や、接続する順番などによって変化します。誤ったデバイス名で操作すると、誤動作の原因となりますのでご注意ください。

② デバイスの権限の設定（有線シリアル使用時）

LRF と制御基板を接続しただけでは、権限が不足しているため ROS からアクセスすることができません。次のコマンドでパーミッション設定を変更し、アクセスできるようにします。なお、Wi-Fi 接続で制御する場合は、この手順は不要です。手順③に進んでください。

```
sudo chmod 666 /dev/VS-WRC058 のデバイス名
```

本書の例では、VS-WRC058 のデバイス名が「ttyUSB0」でしたので、コマンドは次のようになります。

```
sudo chmod 666 /dev/ttyUSB0
```

③ rosserial の実行

rosserial を実行し、メガローバーと通信を行います。roscore を起動してから、別の端末（シェル）で以下のコマンドを実行してください。

・有線シリアル接続の場合

デバイス名には、先ほど確認した VS-WRC058 のデバイス名が入ります。

```
rosrun rosserial_python serial_node.py _port:=/dev/デバイス名
_baud:=115200
```

ここで“_baud:=115200”は、通信速度「ボーレート」を表しています。単位は[bit/sec]です。数字が大きいほど早い通信速度となります。ボーレートに設定できる値には 9600, 19200, 115200 など何種類かありますが、実際に使用可能かどうかは PC 環境によって異なるので調べてください。また、ボーレートを変更する際には、メガローバーのスケッチもそれに合わせて変更する必要があります。

・Wi-Fi 接続の場合

IP アドレスの設定等は不要です。メガローバーが起動した状態で、以下のコマンドを実行してください。

```
rosrun rosserial_python serial_node.py tcp
```

```
vstone@vstoneROSPC:~$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM1
 _baud:=115200
[INFO] [1521452138.004318]: ROS Serial Python Node
[INFO] [1521452138.025052]: Connecting to /dev/ttyACM1 at 115200 baud
[INFO] [1521452140.148024]: Note: publish buffer size is 512 bytes
[INFO] [1521452140.149393]: Setup publisher on rover_sensor [std_msgs/Int16Multi
Array]
[INFO] [1521452140.155315]: Setup publisher on rover_odo [geometry_msgs/Twist]
[INFO] [1521452140.160309]: Note: subscribe buffer size is 512 bytes
[INFO] [1521452140.161070]: Setup subscriber on rover_twist [geometry_msgs/Twist
]
```

図 6-2　rosserial の応答（成功）

接続が成功した際の応答を図 6-3 に示します。これと同じ応答が得られれば、メガローバーと ROS の接続は成功です。

次に、エラーが発生したときの応答例を示します。

```
vstone@vstoneROSPC:~$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM1
 _baud:=115200
[INFO] [1521456232.941900]: ROS Serial Python Node
[INFO] [1521456232.959108]: Connecting to /dev/ttyACM1 at 115200 baud
[ERROR] [1521456232.962662]: Error opening serial: [Errno 13] could not open por
t /dev/ttyACM1: [Errno 13] Permission denied: '/dev/ttyACM1'
```

図 6-3　rosserial の応答（エラー1）

「Permission denied」は、パーミッションの設定が間違っていることを示しています。もう一度手順②でデバイス名を確認し、手順③を実施してください。

```
vstone@vstoneROSPC:~$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM1
 _baud:=115200
[INFO] [1521456279.475981]: ROS Serial Python Node
[INFO] [1521456279.495616]: Connecting to /dev/ttyACM1 at 115200 baud
[ERROR] [1521456279.498873]: Error opening serial: [Errno 16] could not open por
t /dev/ttyACM1: [Errno 16] Device or resource busy: '/dev/ttyACM1'
```

図 6-4　rosserial の応答（エラー2）

「Device or resource busy」は rosserial 以外のプロセスがデバイスにアクセスしている等により接続できないことを示しています。しばらくそのまま放置しておくことで、接続できることがあります。

```
vstone@vstoneROSPC:~$ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM1
 _baud:=115200
[INFO] [1521451981.255419]: ROS Serial Python Node
[INFO] [1521451981.265973]: Connecting to /dev/ttyACM1 at 115200 baud
[INFO] [1521451983.383432]: Note: publish buffer size is 512 bytes
[INFO] [1521451983.384752]: Setup publisher on rover_sensor [std_msgs/Int16Multi
Array]
[INFO] [1521451983.398196]: Setup publisher on rover_odo [geometry_msgs/Twist]
[INFO] [1521451983.400647]: wrong checksum for topic id and msg
```

図 6-5　rosserial の応答（エラー3）

「wrong checksum for topic id and msg」は、メガローバーから受信したメッセージのチェックサムが一致しなかったときに生じるエラーです。**このエラーは発生したタイミングにより行うべき対処が異なります。**

- **通信開始時に発生したとき**
  メガローバーとの通信がうまく確立されていません。もう一度 rosserial を実行しなおしてください。ボーレートの値に問題がある可能性もあります。
- **メガローバーを動作させている最中に発生したとき**
  通信中の一部のビットが欠落したことで発生したものと考えられます。メガローバーが正常に動作しているようであればそのまま操作を継続して問題ありません。ボーレートを変更することで改善する可能性があります。

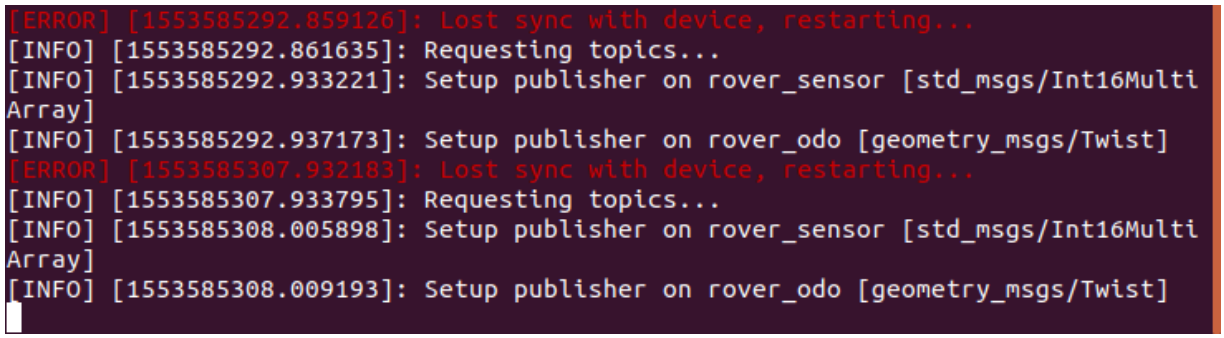

図 6-6　rosserial の応答（エラー4）

「Lost sync with device, restarting…」は、メガローバーと ROS との通信がタイムアウトした際に発生します。タイムアウト時間は初期設定では 15 秒となっています。**このエラーはメッセージの内容により対処が異なります。**

- **Lost sync with device に続けて、Setup publisher が流れる場合**
  図 6-6 のように、Lost sync with device に続けて Setup publisher が表示される場合は、エラーではなく、メガローバーと ROS は正常に通信できている可能性が高いです。エラーを無視して作業を続けて頂いて構いません。
  rosserial のタイムアウトカウントは、メガローバー側で spinOnce()関数が呼び出されることでリセットされます。メガローバーでは仕様により、ROS からメッセージが配信されたときにのみ spinOnce()関数が呼び出されるため、メガローバーに速度指令値を送信しない状態が 15 秒以上続くと、タイムアウトが発生します。

- **Lost Sync with device のみが流れる場合**
  何らかの理由で接続に失敗しています。もう一度、接続手順を最初から実施指定ください。

〔LRF と ROS の接続〕

メガローバーに搭載されている LRF と ROS を接続します。LRF が搭載されていない場合、LRF を使用しないタスクの場合は、この手順は不要です。

メガローバーVer.3.0 にオプションで搭載できる北陽電機製レーザーレンジファインダ URG-04LX-UG01 または YDLiDAR TG30 を例に、LRF の接続について説明します。なお、YDLiDAR TG30 を一つしか使用せず、かつセットアップ時に initenv.sh を実行されている場合は、以下の手順①,②は不要です。

① LRF の接続とデバイス名の確認

本体取り付けられた LRF と PC を、USB ケーブルで接続します。続けて端末で以下のコマンドを実行し、接続した LRF のデバイス名を確認します。

```
dmesg
```

```
2.620611] usb 1-2: new full-speed USB device number 7 using xhci_hcd
2.788631] usb 1-2: device descriptor read/64, error -71
3.079721] usb 1-2: New USB device found, idVendor=15d1, idProduct=0000
3.079735] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
3.079744] usb 1-2: Product: URG-Series USB Driver
3.079752] usb 1-2: Manufacturer: Hokuyo Data Flex for USB
3.081432] cdc_acm 1-2:1.0: ttyACM0: USB ACM device
e@vstoneROSPC:~$
```

図 6-7　LRF のデバイス名確認(URG)

図 6-7 がコマンドの応答例です。6 行目の記述に「Hokuyo Data Flex for USB」とありますので、URG-04LX-UG01 が認識されたことが分かります。7 行目の記述のうち「ttyACM0」が URG-04LX-UG01 のデバイス名となります。

YDLiDAR TG30 の場合は、以下の図 6-8 のようになります。この場合は「ttyUSB1」がデバイス名です。デバイスファイルパスは「/dev/ttyUSB1」となります。

```
[ 8926.107342] usb 1-1: Manufacturer: Silicon Labs
[ 8926.107344] usb 1-1: SerialNumber: 0001
[ 8926.125057] usbcore: registered new interface driver cp210x
[ 8926.125083] usbserial: USB Serial support registered for cp210x
[ 8926.125138] cp210x 1-1:1.0: cp210x converter detected
[ 8926.127057] usb 1-1: cp210x converter now attached to ttyUSB1
```

図 6-8　LRF のデバイス名確認(TG30)

デバイス名は接続する PC や、接続する順番などによって変化します。誤ったデバイス名で操作すると、誤動作の原因となりますのでご注意ください。

② デバイスの権限の設定

LRF と制御基板を接続しただけでは、権限が不足しているため ROS からアクセスすることができません。次のコマンドでパーミッション設定を変更し、アクセスできるようにします。

```
sudo chmod 666 /dev/LRF のデバイス名
```

本書の例では、URG-04LX-UG01 のデバイス名が「ttyACM0」でしたので、その場合のコマンドは次のようになります。

```
sudo chmod 666 /dev/ttyACM0
```

③ （任意）LRF の動作確認

接続した LRF の信号が正しく受信できているか確認を行います。なお、接続設定自体は手順②までで完了していますので、この手順は任意で実施してください。

“A”「URG-04LX-UG01 の信号を読み込む方法」、“B”「YDLiDAR TG30 の信号を読み込む方法」、“C”「読み込んだ LRF の信号を表示して確認する方法」の三つについて説明します。ローバーに搭載されているセンサに合わせた方法（A または B）で信号を読み込めるようにしていただいた後、信号を表示して確認する方法（C）をお試しください。

A〔URG-04LX-UG01 の信号を読み込む方法〕

urg_node は、SCIP 規格で通信を行う LRF のデータを ROS で活用するためのデバイスドライバの役割を担います。本製品に付属の LRF を使用したサンプルプログラムでは、launch ファイル呼び出し時に、urg_node を起動するよう記載しています。

roscore が起動した状態で、新たな端末で以下のコマンドを実行し urg_node を起動します。

```
rosrun urg_node urg_node _serial_port:="/dev/ URG-04LX-UG01 のデバイス名
```

本書の例では、URG-04LX-UG01 のデバイス名が「ttyACM0」でしたので、コマンドは次のようになります。

```
rosrun urg_node urg_node _serial_port:="/dev/ttyACM0"
```

B〔YDLiDAR TG30 の信号を読み込む方法〕

ydlidar_ros は、YDLiDAR シリーズのデータを ROS で活用するためのデバイスドライ

バの役割を担うパッケージです。本製品に付属の LRF を使用したサンプルプログラムでは、launch ファイル呼び出し時に ydlidar_ros を起動するよう記載しています。

では実際に、ydlidar_ros を起動してみましょう。

YDLiDAR の環境セットアップ時に initenv.sh を実行されている場合は、roscore を起動した状態で、新たな端末で以下のコマンドを実行してください。

```
roslaunch ydlidar_ros TG.launch
```

YDLiDAR の環境セットアップ時に initecv.sh を実行されていない場合は、launch ファイルの設定を書き換える必要があります。catkin_ws 内にある ydlidar_ros のディレクトリを開き、内部にある TG.launch ファイルを開きましょう。

```
roscd ydlidar_ros/launch
```

```
gedit TG.launch
```

オープンするデバイスファイルを指定するパラメータ”port”の値を、手順①で確認した YDLiDAR のデバイスファイルのパスに変更してください。デバイスファイルパスが「/dev/ttyUSB1」の場合は以下のようになります。

```
<param name=”port” type=”string” value=“/dev/ttyUSB1” />
```

変更を保存したら、以下のコマンドで launch ファイルを起動しましょう。

```
roslaunch ydlidar_ros TG.launch
```

C〔読み込んだ LRF の信号を表示して確認する方法〕

続いて、Rviz を用いて LRF からの信号が正常に受信できているかを確認します。新しい端末で以下のコマンドを実行し Rviz を起動します。

```
rviz
```

Rviz が起動したら、下図の赤丸で示した「Fixed Frame」の設定値を「laser」または「laser_frame」としてください。設定値の部分をクリックすることで入力できるようになります。なお、利用状況等によっては Rviz の表示内容が異なることがあります。

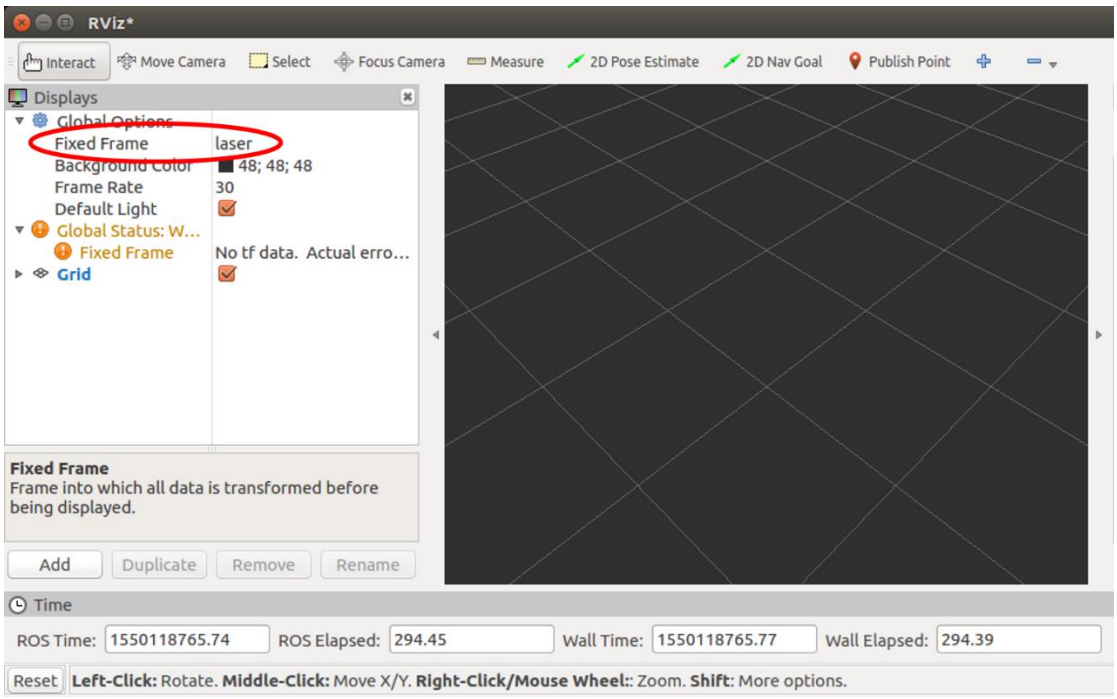

図 6-9　Rviz 起動画面

続いて左下の Add ボタンを押し、現れた選択画面から、「Laser Scan」を選択して OK ボタンを押して下さい。

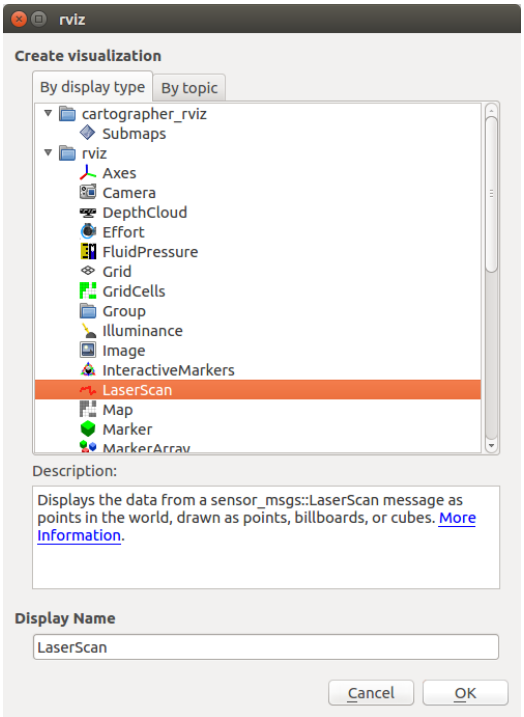

図 6-10　Create visualization 画面

Displays に Laser Scan が追加されたら、左端の三角印を押して詳細を開き、「Topic」の設定値を「/scan」に変更します。その後、下図のように LRF の捉えた障害物が点群として表示されることを確認してください。点群の見た目は異なる場合があります。

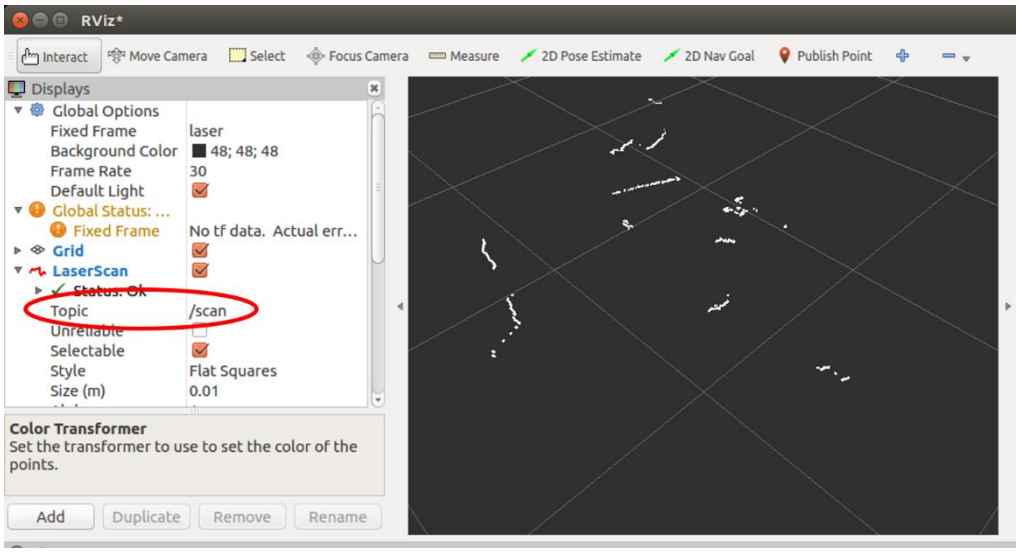

図 6-11　Rviz で LRF の測距情報を確認

6.5 メガローバー用メッセージの仕様

メガローバーは 1 つのメッセージをサブスクライブ（購読）し、2 つのメッセージをパブリッシュ（配信）しています。ここでは、各メッセージの仕様について解説します。

〔サブスクライブ〕

- /rover_twist

“/rover_twist”は、メガローバーへの移動指令を記載するメッセージです。メガローバーはこのメッセージをサブスクライブし、記載されている並進移動量[m/s]および旋回量[rad/s]の指令値に従って動作します。

表 6-1 に、“/rover_twist”の詳細を示します。メガローバーの座標系は、前方向を x 軸の正方向、左方向を y 軸の正方向にとる右手系で定義しています。メガローバーは並行二輪移動台車ですので、x 軸方向への移動と z 軸回りでの旋回が行えます。

よって、“linear.x”に並進移動量[m/s]を入力し、“angular.z”に旋回量[rad/s]を入力してパブリッシュすることで、メガローバーをコントロールすることができます。

表 6-1　/rover_twist の詳細

メッセージ名	/rover_tsist
型	geometry_msgs/Twist
内容	linear: 　x:　　//　x 軸方向の並進移動量指令値（float64） 　y:　　//　不使用 　z:　　//　不使用 angular: 　x:　　//　不使用 　y:　　//　不使用 　z:　　//　z 軸周りの旋回量指令値（float64）

〔パブリッシュ〕

- /rover_sensor

“/rover_sensor”は、メガローバーのオプション品であるバンパーセンサの入力値および現在のバッテリー電圧を記録したメッセージです。

表 6-2 に“/rover_sensor”の詳細を示します。ユーザは“/rover_sensor”をサブスクライブすることで、これらの値を利用した制御プログラムを作成することができます。

表 6-2　/rover_sensor の詳細

メッセージ名	/rover_sensor
型	std_msgs/Int16MultiArray
内容	layout: 　dim: [] 　data_offset: 0 data: [s0]　//　VS-WRC058 の MU16_M_DI data: [s1]　//　バッテリー電圧

- /rover_odo

“/rover_odo”は、メガローバーの現在速度と旋回量を記録しているメッセージです。これらの値はエンコーダ値をもとに算出されています。本製品のサンプルパッケージに含まれるノード「pub_odom」は、“/rover_odo”をサブスクライブして、メガローバーの現在位置と姿勢を示すオドメトリ情報を計算して配信しています。

表 6-3 に“/rover_odo”の詳細を示します。“/rover_odo”は、“rover_twist”と同じ、geometry_msgs/Twist 型のメッセージです。Twist 型は、移動量を表すためによく使われています。ROS には、データの種類に応じた沢山の型が用意されており、例えば「pub_odom」が配信するオドメトリ情報の型は nav_msgs/Odometry 型です。

表 6-3　/rover_odo の詳細

メッセージ名	/rover_odo
型	geometry_msgs/Twist
内容	linear: 　x:　　//　x 軸方向の並進移動量（float64） 　y:　　//　不使用 　z:　　//　不使用 angular: 　x:　　//　不使用 　y:　　//　不使用 　z:　　//　z 軸周りの旋回量（float64）

7　ゲームパッド操作サンプル

ゲームパッド操作サンプルは、市販のゲームパッドを用いてメガローバーを走行させることができるサンプルプログラムです。本章では、その使用方法を説明します。

使用するゲームパッドは、アナログスティック入力機能を有する USB 接続のものを、お客様にてご用意ください。本製品に付属しております VS-C3 は使用できません。また、ゲームパッドの種類によっては、デバイスドライバが非対応で動作しない可能性もございます。

① ゲームパッド入力パッケージ「Joy」のインストール

ROS でゲームパッドの入力を取得するためのパッケージ「Joy」をインストールします。端末を起動し、以下のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-joy ros-ROS バージョン-joystick-drivers
```

② ゲームパッドの接続とデバイスファイルのパスの確認

PC とゲームパッドは未接続の状態で端末を起動し、以下のコマンドを順に実行してください。

ゲームパッドのデバイスファイルが生成されるフォルダに移動します。

```
cd /dev/input
```

フォルダの中身を確認します。

```
ls
```

```
vstone@vstoneROSPC:~$ cd /dev/input/
vstone@vstoneROSPC:/dev/input$ ls
by-id    event0  event10  event12  event3  event5  event7  event9  mouse0
by-path  event1  event11  event2   event4  event6  event8  mice
```

図 7-1　ls の応答（接続前）

PC の USB ポートにゲームパッドを接続し、再度フォルダの中身を確認します。

```
ls
```

```
vstone@vstoneROSPC:/dev/input$ ls
by-id    event0  event10  event12  event2  event4  event6  event8  js0   mouse0
by-path  event1  event11  event13  event3  event5  event7  event9  mice
```

図 7-2　ls の応答（接続後）

図 7-1 と図 7-2 を比較すると、ゲームパッド接続後に“js0”が増えていることが分かります。これがゲームパッドのデバイスファイルです。よって、ファイルのパスは“/dev/input/js0”と

なります。デバイスファイル名は、機種やゲームパッドを接続するタイミングによって変化しますので、都度確認してください。

③　デバイスファイルパスをパラメータとして設定

手順①で確認したゲームパッドのデバイスファイルパスを、ゲームパッド操作サンプルの launch ファイル「joycon.launch」を使ってパラメータとして設定します。次の手順に沿って作業を実施してください。

端末を開き、gedit を使って joycon.launch を開きます。

```
gedit ~/catkin_ws/src/megarover3_ros/launch/joycon.launch
```

次の行を探してください。

```
<param name="dev" type="string" value="/dev/input/js0" />
```

launch ファイルでは、param 要素を用いることでパラメータを設定することができます。name 属性がパラメータ名、value 属性がパラメータ値です。“value=”以下を先ほど取得したゲームパッドのデバイスファイルパスに書き換えてください。書き換えが完了したら保存してください。

④　移動量の指令値の確認

メガローバーへの移動量指令値が正しく出力されるか確認します。**安全のため、メガローバーと PC が接続されていない状態で作業を行ってください。**新しい端末で次のコマンドを実行し、joycon.launch を呼び出します。

```
roslaunch megarover3_ros joycon.launch
```

メガローバーへの移動指令が正確に出力されているか確認してみましょう。新しく端末をたち上げ、次のコマンドを実行してください。

```
rostopic echo /rover_twist
```

“rostopic echo メッセージ名”は、ノード間でやり取りされているメッセージを見ることができるコマンドです。コマンドと書きましたが、実際にはこれも ROS のノードのひとつとして実装されています。ここでは、メッセージ名に“/rover_twist”を指定して、メガローバーに送信されている移動速度[m/s]と、旋回量[rad/s]を確認します。

ゲームパッド操作サンプルにはセーフティーボタンが実装されており、デフォルトではゲームパッドの 3 番ボタンを押下している間のみ指令値が送信されます。ゲームパッドの 3 番ボタンを押下しながらアナログスティックを動かし、rostopic echo の応答を確認しましょう。

```
---
linear:
  x: 0.196164146066
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.234151661396
---
```

図 7-3　rostopic echo の応答（/rover_twist）

アナログスティックを動かすことで、図 7-3 のように linear.x および angular.z の値が変化すれば、指令値は正しく出力されています。linear は並進移動量を、angular は旋回量を表しており、x,y,z はそれぞれ座標軸です。

「何も値が出てこない場合」や「片方の値しか変化しない場合」は、手順④の操作方法のカスタマイズが必要です。あるいは、値は両方とも変化したが、ボタンやアナログスティックの割り当てが不自然で操作しづらいときなども、操作方法のカスタマイズを行ってください。

指令値が正しく出力されていて、操作のしやすさにも問題がない場合は、手順⑤に進んでください。

⑤　操作方法、パラメータのカスタマイズ

joycon.launch を編集することで、操作方法や最高速度などを変更することができます。次のコマンドで joycon.launch を開いてください。

```
gedit ~/catkin_ws/src/megarover3_ros/launch/joycon.launch
```

以下に示す joycon の node 要素内にある各 param 要素の value 属性値を編集します。

```
<!-- joycon -->
<node pkg=”megarover3_ros” type=”joycon” name=”joycon” respawn=”true” >
  <param name=”axis_linear” value=”1” />
  <param name=”axis_angular” value=“0” />
  <param name=”scale_linear” value=”0.6” />
  <param name=”scale_angular” value=”0.8” />
  <param name=”safety_button” value=”2” />
</node>
```

各パラメータの意味を次表に示します。

表 7-1　joycon のパラメータ一覧

パラメータ名	説明	初期値
axis_linear	前後の移動速度[m/s]を決めるアナログスティックの番号 /joy メッセージの axes 配列の要素番号から選択	1
axis_angular	旋回量[rad/s]を決めるアナログスティックの番号 /joy メッセージの axes 配列の要素番号から選択	0
scale_linear	移動量のゲイン（1.4 までに留めること） 移動量＝scale_linear×アナログスティック入力値	0.6
scale_angular	旋回量のゲイン（3.0 程度までに留めること） 旋回量=scale_angular×アナログスティック入力値	0.8
safety_button	セーフティーボタンの番号 /joy メッセージの button 配列の要素番号から選択	2

“axis_linear_x/y”、“axis_angular”、“safety_button”を編集することで操作方法を変更できます。**安全のため、メガローバーと PC が接続されていない状態で**以下のコマンドを実行してください。

```
roslaunch megarover3_ros joycon.launch
```

ゲームパッドからの入力値を確認します。

```
rostopic echo /joy
```

```
---
header:
  seq: 55
  stamp:
    secs: 1521521432
    nsecs:   9187362
  frame_id: ''
axes: [0.07760214805603027, 0.3597537875175476, 0.0, 0.0, 0.0, 0.0]
buttons: [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
---
```

図 7-4　rostopic echo の応答（/joy）

図 7-4 に、ゲームパッドの 3 番ボタンを押下しアナログスティックを倒したときの“rostopic echo /joy”の応答を示します。アナログスティックの入力値は axes 配列に、ボタンの入力値は buttons 配列に格納されます。どの要素にどのスティック、ボタンの値が格納されるかはゲームパッドによって異なります。ゲームパッドを操作して適当なパラメータを設定してください。

“scale_linear”、“scale_angular”を編集することで、ゲームパッドの操作量に対する応答を変化させることができます。メガローバーが発揮できる速度には限界がありますので、大きな値を設定すると操作性が失われ、大変危険です。表 7-1 を参考に慎重に変更を行ってください。

⑥ ゲームパッド操作サンプルの実行

手順②、③で書き換えた launch ファイルを呼び出し、メガローバーをゲームパッドで操作します。メガローバーが壁などと衝突しないよう十分なスペースを確保してください。また、PC はメガローバーにしっかりと固定し、落下などしないようご注意ください。ケーブルが絡まったりすることがないように、ケーブルの状態を確認しつつ実施してください。

メガローバーの電源が OFF になっていることを確認し、6.3 項に従って PC とメガローバーを接続します。接続が正常に行われていることを確認したら、新しい端末で次のコマンドを実行し、joycon.launch を呼び出します。

```
roslaunch megarover3_ros joycon.launch
```

ゲームパッドを操作していない状態でメガローバーの電源を入れます。ゲームパッドをゆっくりと操作し、メガローバーが走行することを確認します。アナログスティックを急激に操作すると危険ですのでおやめください。

8　マウス（タッチパッド）操作サンプル

マウス（タッチパッド）操作サンプルは、マウスまたはタッチパッドを使い、メガローバーを走行させることができるサンプルです。本章ではその使用方法を説明します。

① mouse_teleop パッケージのインストール

マウス（タッチパッド）操作サンプルの実行には mouse_teleop パッケージのインストールが必要です。端末を立ち上げ、以下のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-mouse-teleop
```

② マウス（タッチパッド）操作サンプルの実行

launch ファイルを用いてサンプルを起動します。メガローバーが壁などと衝突しないよう十分なスペースを確保してください。また、PC はメガローバーにしっかりと固定し、落下などしないようご注意ください。ケーブルが絡まったりすることがないように、ケーブルの状態を確認しつつ実施してください。

安全のため、メガローバーと PC が接続されていない状態で mousectrl.launch を呼び出します。roscore を起動していない場合は別端末にて先に起動しておいてください。

```
roslaunch megarover3_ros mousectrl.launch
```

mousectrl.launch を呼び出すと、図 8-1 のようなウィンドウが画面に現れます。白いエリアをクリックし、上下方向にドラッグすると移動量 v が、左右方向にドラッグすると旋回量 w が出力されます。

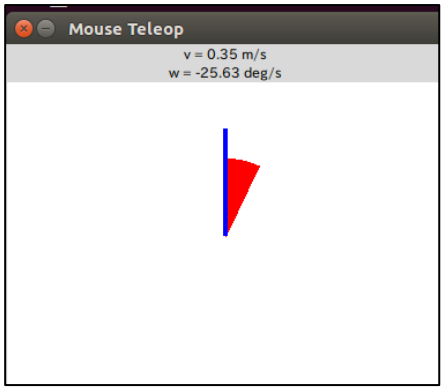

図 8-1　Mouse Tekeop の GUI

メガローバーの電源が OFF になっていることを確認し、6.3 項に従って PC とメガローバーを接続します。その後メガローバーの電源を ON にしてマウスを操作すると、メガローバーが走行します。

9　SLAM（gmapping）サンプル

SLAM(Simultaneous Localization and Mapping)は、ロボットの自己位置推定と環境地図作成を同時に行う手法です。ロボット掃除機などの自律移動ロボットでよく用いられており、現在も盛んに研究されている高度な技術です。

ROS には、この SLAM 手法を実装したパッケージが存在しており、誰でも簡単に SLAM を利用することが可能となっています。ROS で SLAM を実装したパッケージはいくつか存在しますが、ここでは「gmapping」を利用して、SLAM を実行してみましょう。

SLAM（gmapping）サンプルは、メガローバー前部に取り付けられた LRF で取得した情報をもとに、自己位置の推定と周囲の環境地図の作成を行います。メガローバーの移動は手動で行う必要がありますので、ゲームパッド操作サンプルまたはマウス（タッチパッド）操作サンプルを一緒に使用します。

LRF(Laser Rangefinder）は、周囲に存在する物体の位置を検出することができるセンサです。レーザー光を発射し、その反射を見て物体の有無を確認しています。視野が広く、計測精度が高いことから、自律移動台車ロボットなどでよく使用されています。

「gmapping」は ROS でメジャーな SLAM パッケージのひとつです。LRF のデータと、車輪の回転数から移動量を計測するオドメトリの情報を用いて SLAM を行います。「gmapping」を用いて作成した地図の例を図 9-1 に示します。

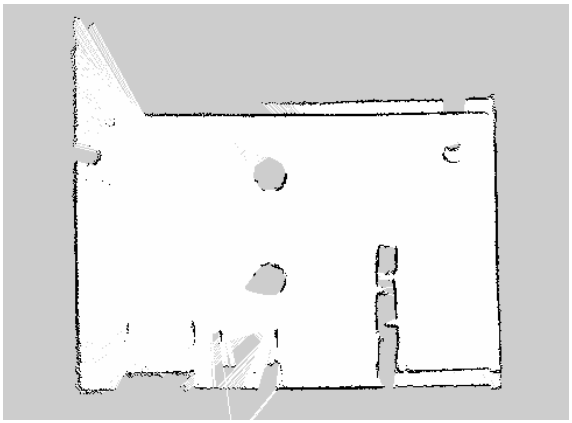

図 9-1　gmapping の地図作例

図中の黒い部分は障害物が存在する通行不可能な箇所、白い領域は通行可能な場所、灰色は未知領域です。地図はグリッド状になっており、各グリッドには障害物が存在する確率（占有確率）を示す 0～100 の値もしくは、未知領域を示す-1 の値が与えられます。この占有確率を用いた地図の表現方法は、ROS の 2D マップ表現方法として標準化されています。gmapping は絶対に障害物が存在しない 0 もしくは、絶対に障害物が存在する 100 の 2 種の値しか出力しません。

9.1 SLAM（gmapping）サンプルの実行

SLAM（gmapping）サンプルを実行する手順を説明します。以下の手順に従って作業を行ってください。

① gmapping のインストール

slam-gmapping パッケージをインストールします。次のコマンドを実行してください。

```
sudo apt install ros-ROS バージョン-slam-gmapping
```

② メガローバーのセットアップ

メガローバーの電源が OFF になっていることを確認し、6.3 項に従いメガローバーと ROS を接続してください。

続いて、メガローバーを手動で操作できるようにします。ゲームパッドを使用する場合は 7 章の、マウスまたはタッチパッドを使用する場合は 8 章の手順に従い、各サンプルを使用して、メガローバーを操作できることを確認してください。

③ launch ファイルの修正

新しい端末で次のコマンドを実行し、gmapping.launch を開いてください。

```
gedit ~/catkin_ws/src/megarover3_ros/launch/gmapping.launch
```

URG-04LX-UG01 を使用する場合、次の項目を確認し、default 属性値を手順②で確認した LRF のデバイスファイルパスに変更して、保存してください。

```
<!-- LRF のデバイスファイルパス  -->
<arg name="port_urg" default="/dev/ttyACM0" />
```

YDLiDAR TG30 を使用する場合、次の項目を確認し、default 属性値を手順②で確認した LRF のデバイスファイルパスに変更して、保存してください。

```
<!-- YDLiDAR デバイスドライバノード  -->
<param name="port" type=”string” default="/dev/ydlidar " />
```

④ SLAM の実施

新しい端末でコマンドを実行し、SLAM（gmappig）サンプルを起動します。このとき引数を与えることで、使用する台車、センサの種類に合わせたパラメータ設定や、ノードの起動が行えます。

launch ファイルの起動に引数を与える場合、launch ファイル名の後に「引数名:=引数」という形で追記します。与えられる引数は、あらかじめ launch ファイル内で定義されている物のみです。

引数は以下の 2 種類が設定可能です。

○ rover_type

rover_type:=mega3 で設定することで、メガローバーVer.3.0 用のパラメータが適用されます。

rover_type:=outdoor で設定することで、メガローバーVer.3.0 屋外対応版用のパラメータが適用されます。

○ lrf

lrf:=urg で設定することで、URG-04LX-UG01 用のパラメータが適用され、urg_node が呼び出されます。

lrf:=tg30 で設定することで、YDLiDAR TG30 用のパラメータが適用され、ydlidar_ros が呼び出されます。

例えば、メガローバーVer.3.0 で URG-04LX-UG01 を使用している場合は、launch のコマンドは以下のようになります。

```
roslaunch megarover3_ros gmapping.launch rover_type:=mega3 lrf:=urg
```

メガローバーVer.3.0 屋外対応版で YDLiDAR TG30 を使用している場合は以下のようになります。

```
roslaunch megarover3_ros gmapping.launch rover_type:=outdoor lrf:=tg30
```

新しく端末を立て、ご使用の環境にあった引数を設定し、gmapping.launch ファイルを実行してください。

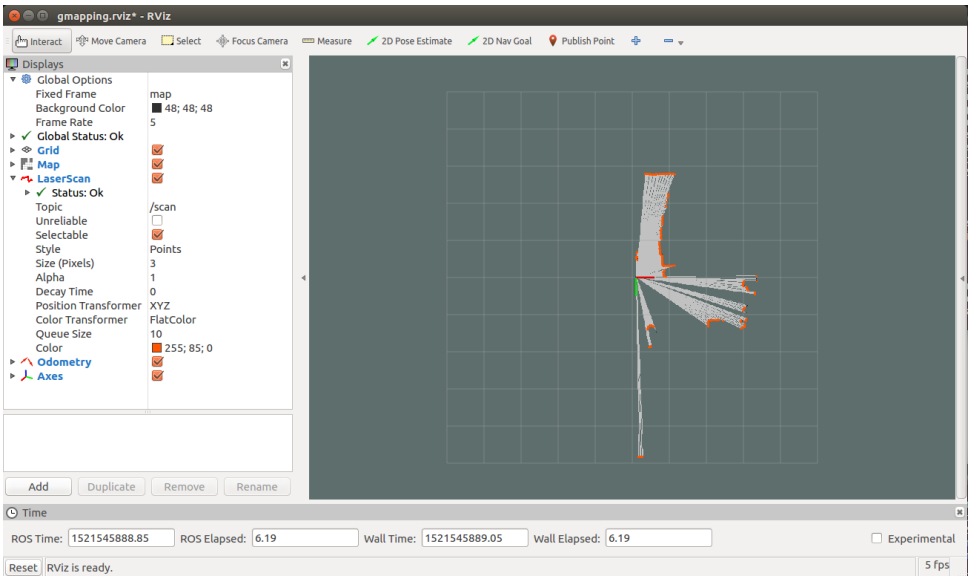

図 9-2　Rviz による gmapping の表示

サンプルが起動すると、図 9-2 に示すように Rviz が起動します。Rviz 上には LRF が捉えている障害物と、生成される地図がリアルタイムで表示されます。画面が表示されなかったり、LRF のデータが表示されなかったりした場合は、起動中に何らかのエラーが発生した可能性があります。一度プログラムを終了し、再度実行しなおしてください。

地図が生成され始めたら、メガローバーをゆっくりと移動させていきましょう。移動した範囲の地図が徐々に作成されていくはずです。急発進や急停止、急旋回などを行うと地図が乱れやすいので注意してください。

9.2 map の保存

作成した map を保存する方法を説明します。map の保存には map_server パッケージの map_saver ノードを使用します。次のコマンドを実行することで、ホームフォルダに地図の画像ファイルとデータファイルが保存されます。ファイル名は適宜設定してください。

```
rosrun map_server map_saver -f ファイル名
```

map_server または map_saver が見つからないというエラーが出てマップが保存できない場合、必要なパッケージが不足しています。11.1 節を参考に、navigation スタックをインストールし、map_server をインストールしてください。

10 navigation サンプル

ロボットの自律移動を行うためには、自己位置推定や移動経路のプランニングといった技術が必要です。ROS では navigation スタックという形でこれらの機能が提供されています。スタックとは、複数のパッケージを束ねたパッケージ群のことです。ここでは、SLAM（gmapping）サンプルを用いて作成した地図を使って、メガローバーを目標地点まで自律的に移動させてみましょう。

10.1navigation スタックのインストール

navigation スタックをインストールします。navigation スタックに属する各パッケージは、以下のコマンドを実行することで一括でインストールすることができます。

```
sudo apt install ros-ROS バージョン-navigation
```

10.2地図の作成と launch ファイルの編集

まず、ロボットが移動する環境の地図を作成します。自律移動を行うためには、できるだけ正確な地図が必要です。また、launch ファイルを編集し、作成した地図が読み込まれるようにします。

① 地図の作成

9.1 項の手順に従い gmapping を実行し、navigation で移動させたい領域の地図を作成してください。本来障害物が無いはずの場所に障害物があると判定された場合は、再び同じ個所を走行することで地図が正しく修正されることがあります。

② 地図の保存

9.2 項の手順に従い、map_saver を用いて地図を保存してください。地図が保存されていることを確認したら、SLAM（gmapping）サンプルは終了させてください。

③ 地図ファイルの移動

保存した pgm ファイルと yaml ファイルを以下のアドレスに移動させてください。

```
~/catkin_ws/src/megarover3_ros/map
```

④ launch ファイルの編集

megarover_move_base_dwa.launch を編集し、作成した地図が読み込まれるようにします。以下のコマンドで launch ファイルを開いてください。

```
gedit
~/catkin_ws/src/megarover3_ros/launch/megarover_move_base_dwa.launch
```

下記の行を探し、“ファイル名.yaml”を、先ほど保存，移動させた地図ファイルのファイル名に合わせて設定します。

```
<arg name=”map_file” default=”$(find megarover3_ros)/map/ファイル名.yaml”
```

10.3navigation サンプルの実行

それでは navigation サンプルを実行して、メガローバーに自律移動をさせてみましょう。なお、環境によっては上手く自律移動できないことがあります。その場合は、各機能のパラメータを調整いただくことで、改善する可能性があります。サンプルプログラムではオドメトリの影響が小さくなるようパラメータを設定しています。

平面で、オドメトリの誤差原因となるタイヤの滑りが少なく、分かれ道や柱の凹凸など、分かりやすい特徴が存在する通路といった環境でお試しください。LRF では捉えられない高さに障害物があったり、イスやテーブルの脚といった小さな障害物が多く存在する環境では正常に動作させることが難しくなります。

以下の手順に従って作業を行ってください。11.1 項を実施した後、メガローバーと ROS との接続を解除している場合は、6.3 項にもとづいて接続作業を行ってください。

① launch ファイルのセッティング

新しい端末で次のコマンドを実行し、megarover_move_base_dwa.launch を開いてください。

```
gedit ~/catkin_ws/src/megarover3_ros/launch/megarover_move_base_dwa.launch
```

URG-04LX-UG01 を使用する場合、次の項目を確認し、default 属性値を手順②で確認した LRF のデバイスファイルパスに変更して、保存してください。

```
<!-- LRF のデバイスファイルパス  -->
<arg name="port_urg" default="/dev/ttyACM0" />
```

YDLiDAR TG30 を使用する場合、次の項目を確認し、default 属性値を手順②で確認し

た LRF のデバイスファイルパスに変更して、保存してください。

```
<!-- YDLiDAR デバイスドライバノード  -->
<param name="port" type=”string” default="/dev/ydlidar " />
```

② launch ファイルの呼び出し

メガローバーと ROS が接続された状態で、以下のコマンドを実行して launch ファイルを呼び出します。このとき、使用環境に合わせて引数を適切に設定してください。引数の詳細は 9.1 項の手順④をご参照ください。

例えば、メガローバーVer.3.0 屋外対応版で URG-04LX-UG01 をご利用の場合は、以下のようになります。

```
roslaunch megarover3_ros megarover_move_base_dwa.launch
rover_type:=outdoor lrf:=urg
```

図 10-1 に示すように、Rviz 上に 10.2 項で作成した地図と現在の LRF が捉えた物体情報が表示されることを確認してください。

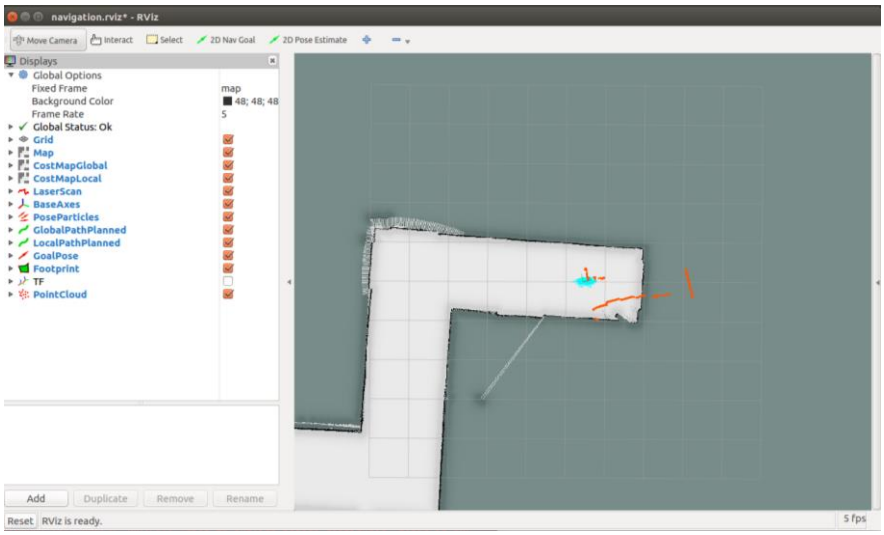

図 10-1　navigation サンプルの Rviz 画面

③ 初期位置の設定

navigation サンプルが起動出来たら、メガローバーのおおよその初期位置を教えてやる必要があります。画面上部の「2D Pose Estimate」ボタンをクリックしてから、図 10-2 に示すように、メガローバーの実際の初期位置をクリックし、ドラッグして方向を決めます。

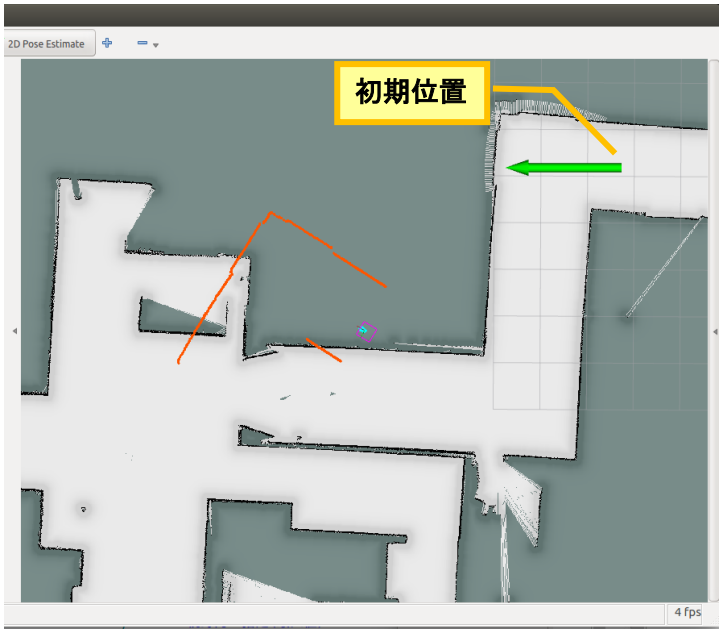

図 10-2　メガローバー初期位置の設定

図 10-3 に示すように、map の壁や障害物と、LRF の検出結果がおおよそ一致するようになるまで調整を行ってください。

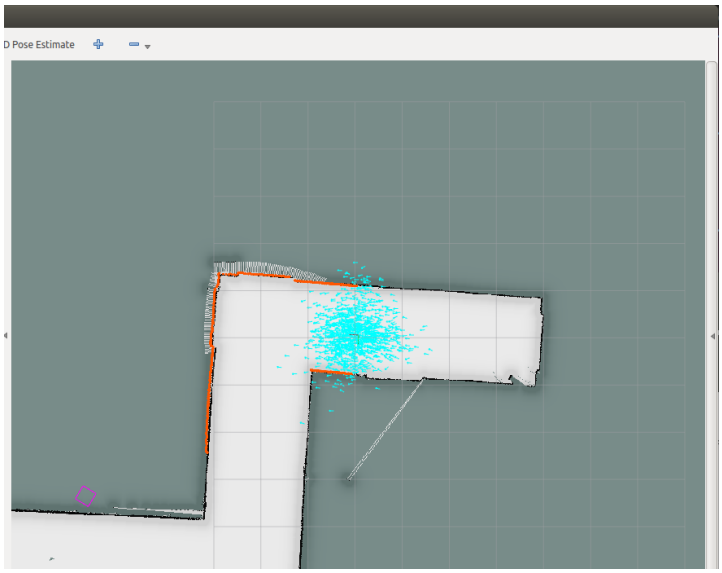

図 10-3　初期位置の設定が完了した状態

このとき、指定したメガローバーの位置周辺に青い小さな点がいくつも散らばっていることが確認できます。これらは、navigation スタック内で自己位置推定を担うノード「amcl」が計算する、メガローバーの推定自己位置です。動作開始前は自己位置が定まっていないため、青い点は大きく散らばっています。

④　目標位置の設定

初期位置の設定を完了したら、いよいよメガローバーの移動目標地点を設定します。なお目標地点を設定すると、メガローバーは即座に走行を開始しますので注意してください。

画面上部の「2D Nav Goal」ボタンをクリックしてから、図 10-4 に示すように、目標位置をクリックし、ドラッグして方向を決めてください。

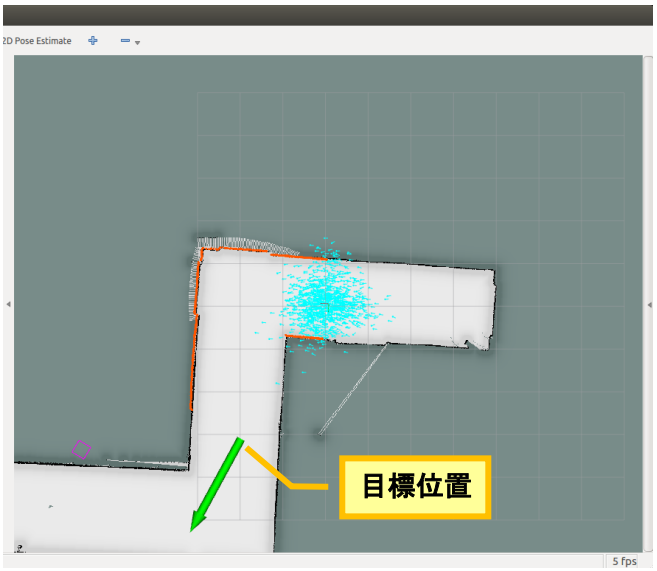

図 100-4　目標位置の設定

目標位置を設定すると、メガローバーは即座に走行を開始します。走行中の Rviz の様子を図 10-5 に示します。計画された移動経路が緑線で、目標位置姿勢が赤矢印で示されています。また、検出されている障害物の周りには色が付いた領域が存在します。これらは、観測結果をもとに数値化された衝突危険性を表すコストマップです。

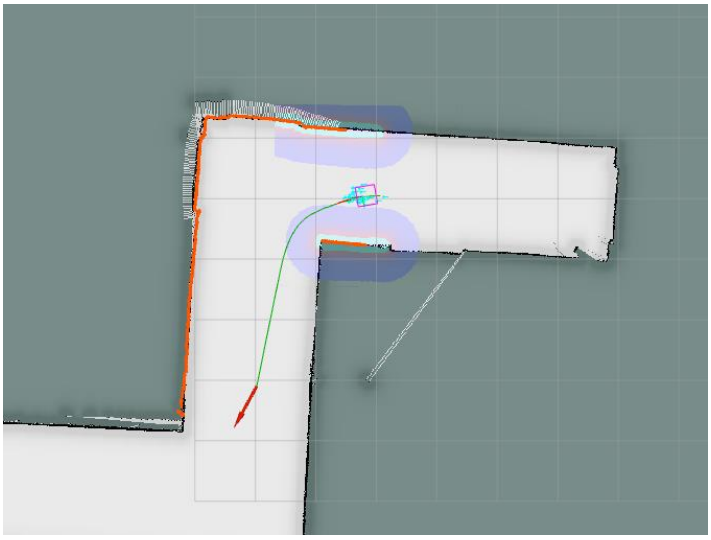

図 10-5　navigation による走行の様子

⑤　走行の終了

　メガローバーは目標位置に到達すると、自動的に停止します。そこから新たに目標位置を設定することも可能です。

11 動作確認バージョンについて

メガローバーの各サンプルが使用しているパッケージについては、以下のバージョンで動作確認を行っております。“apt”コマンドを用いてバージョン指定のオプションを付けずにバイナリを導入した場合、動作確認を行っていないバージョンが導入されシステムが正常に動作しない可能性がございます。また、Github からソースコードを取得してビルドする導入方法でも同様です。その際は、バージョン指定のオプションを用いて弊社にて動作確認を行っているバージョンのパッケージを導入してください。

表 111-1 動作確認パッケージのバージョン

パッケージ名	バージョン
rosserial	0.8.0
mouse-teleop	0.2.6
gmapping	1.3.10
navigation	1.14.4

商品に関するお問い合わせ
TEL: 06-4808-8701　　FAX: 06-4808-8702　　E-mail: infodesk@vstone.co.jp
受付時間　：9:00～18:00（土日祝日は除く）
ヴイストン株式会社　　www.vstone.co.jp
〒555－0012　大阪市西淀川区御幣島 2-15-28