4WDS ローバーX40A/X120A
ROS 2 制御モード
取扱説明書

(2025.03.12)

Vstone®
ヴイストン株式会社

目次

1. はじめに/注意事項

本書は、4WDS ローバーX40A および 4WDS ローバーX120A を ROS 2 で制御する際の取扱説明書です。4WDS ローバー本体についての説明書である「4WDS ローバーX40A 取扱説明書」または「4WDS ローバーX120A 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 22.04 と ROS 2 Humble 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 2 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ライブラリ等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等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. 動作確認環境

本製品は以下の環境にて動作確認を実施しております。

○ROS 関連ソフトウェア

ROS 2	ROS 2 Humble
micro_ros_setup	3.1.0 Humble
micro-ROS-Agent	3.0.5

○Arduino 関連ソフトウェア

Arduino IDE 1.x	1.8.19
Arduino IDE 2.x	2.0.4
esp32-arduino	2.0.13
NimBLE-Arduino	1.4.1
micro_ros_arduino	v2.0.5-humble

○ROS 2 実行環境

OS	Ubuntu 22.04
ROS	ROS 2 Humble
CPU	Intel® Core™ Ultra 5 125H Processor
RAM	32GB
ストレージ	M.2 SSD 256GB
グラフィック	Intel® Arc™ GPU

（※）上記条件を満たしていても、相性などにより正常に動作しない場合があります。

（※）ROS PC オプション（UM350/UM590/UM760）をお使いの場合、Arduino IDE を用いて基板にスケッチを書き込むことはできません。

3. 4WDS ローバーのセットアップ

4WDS ローバーを ROS 2 で制御するためには、対応するスケッチへの書き換えが必要となります。以下の手順で作業を実施してください。

① ライブラリのバージョン変更

「4WDS ローバーX40A 取扱説明書」または「4WDS ローバーX120A 取扱説明書」の記述に従い、ローバーの制御基板にスケッチを書き込むための Arduino IDE 環境をセットアップしてください。

esp32 ボードライブラリのバージョンは 2.0.13 を、NimBLE-Arduino のバージョンは 1.4.1 を使用してください。

② Vstone 版 micro_ros_arduino ライブラリの導入

Arduino 環境で micro-ROS を使用するためのライブラリを導入します。Vstone 製の各種ロボットで使用する専用メッセージを加えた Vstone 版 micro_ros_arduino をご利用ください。以下のページにアクセスし、ページ内の説明に従いライブラリを導入してください。

GitHub vstoneofficial/micro_ros_arduino：
https://github.com/vstoneofficial/micro_ros_arduino

③ スケッチの編集

4WDS ローバーX40A をご利用の場合は、メニューのファイル＞スケッチ例＞vs_wrc058_fwdsrover から、fwdsrover_x40a_ros2.ino を選択し開いてください。4WDS ローバーX120A をご利用の場合は、fwdsrover_x120a_ros2.ino を開いてください。

スケッチに記載の説明に従い、各種定義を設定してください。

Wi-Fi で接続する場合は、SSID やパスワード、micro-ROS Agent を実行するデバイスの IP アドレス、ポート番号も設定してください。

④ スケッチの書き込み

ローバーと PC を接続し、スケッチを書き込んでください。

ROS 2 用のスケッチでも、付属コントローラーを用いた操作を実施することはできます。ただし、ROS 2 が接続されていない場合、接続処理のため走行中動作が不安定になることがございますのでご注意ください。

4. ROS 2 のセットアップ

下記の公式ドキュメントを参考に、ROS 2 Humble をお使いの環境にセットアップしてください。

ROS 2 Documentation: Humble documentation：
https://docs.ros.org/en/humble/Installation.html

5. micro-ROS Agent のセットアップ

4WDS ローバーは micro-ROS を利用することで ROS 2 に対応します。micro-ROS はマイコンレベルで動作している機器を ROS 2 の通信ネットワークに参加させるための仕組みです。詳しくは下記の micro-ROS のページをご覧ください。

micro-ROS | ROS 2 for microcontrollers：
https://micro.ros.org/

4WDS ローバーの制御基板に搭載されているマイコン ESP32 と、ROS 2 の DDS 通信を中継する機能を持つアプリケーション「micro-ROS Agent」をお使いの PC 等にインストールします。以下の手順で作業を実施してください。

シェルを新しく起動してください。
ワークスペースを作成し、GitHub から micro_ros_setup をクローンします。

```
mkdir -p ~/uros_ws/src
```

```
cd ~/uros_ws/src
```

```
git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro_ros_setup.git
```

```
cd ~/uros_ws
```

```
rosdep update && rosdep install --from-paths src --ignore-src -y
```

ビルドし、パスを通します。

```
colcon build
```

```
source install/local_setup.bash
```

同じシェルで micro-ROS Agent のセットアップを行います。

```
ros2 run micro_ros_setup create_agent_ws.sh
```

```
ros2 run micro_ros_setup build_agent.sh
```

```
source install/local_setup.bash
```

sh ファイルを実行時に以下のような Warning が出ることがありますが、問題ありません。

```
CMake Warning (dev) at /usr/share/cmake-
3.22/Modules/FindPackageHandleStandardArgs.cmake:438 (message):
  The package name passed to `find_package_handle_standard_args` (tinyxml2)
  does not match the name of the calling package (TinyXML2).  This can lead
  to problems in calling code that expects `find_package` result variables
  (e.g., `_FOUND`) to follow a certain pattern.
Call Stack (most recent call first):
  cmake/modules/FindTinyXML2.cmake:40 (find_package_handle_standard_args)
  /opt/ros/humble/share/fastrtps/cmake/fastrtps-config.cmake:51 (find_package)
  CMakeLists.txt:153 (find_package)
```

シェル起動時にワークスペースがオーバーレイされるように設定します。

```
echo "source ~/uros_ws/install/local_setup.bash" >> ~/.bashrc
```

以上で micro-ROS Agent のセットアップは完了です。

6. 4WDS ローバー用 ROS 2 パッケージのセットアップ

4WDS ローバーを ROS 2 で使用するためのサンプルが入ったパッケージ「fwdsrover_xna_ros2」のセットアップを行います。以下の手順で作業を実施してください。

シェルを新しく起動してください。
ワークスペースを作成し、GitHub から fwdsrover_xna_ros2 をクローンします。

```
mkdir -p ~/ros2_ws/src
```

```
cd ~/ros2_ws/src
```

```
git clone https://github.com/vstoneofficial/ fwdsrover_xna_ros2.git
```

```
rosdep install -r --from-paths . --ignore-src --rosdistro $ROS_DISTRO -y
```

ビルドします。

```
cd ~/ros2_ws
```

```
colcon build --symlink-install
```

シェルを起動時にワークスペースがオーバーレイされるように設定します。

```
echo "source ~/ros2_ws/install/local_setup.bash" >> ~/.bashrc
```

以上で fwdsrover_xna_ros2 のセットアップは完了です。

colcon build 時に” --symlink-install”オプションを用いることで、可能であれば、インストール時にファイルコピーではなくシンボリックリンクの作成を行います。これにより、Python ファイルや YAML ファイルなど一部のファイルでは、実行時にワークスペースの src ディレクトリ内のファイルが参照されるため、内容を書き換えた後、colcon build を行わなくても変更が動作に反映されます。

7. 4WDS ローバーと ROS 2 の接続

作業を行う前に、4WDS ローバーを十分充電しておいてください。4WDS 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、micro-ROS 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、4WDS 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〕

4WDS ローバーの制御基板 VS-WRC058 と micro-ROS Agent 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。

4WDS ローバーの制御基板 VS-WRC058 と PC を USB ケーブルで接続し、ローバーの電源を ON

にします。続けて、シェル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。

```
sudo 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
555] usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
oneROSPC:~$
```

図 7-1　制御基板のデバイス名確認

図 7-1 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 micro-ROS Agent 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　micro-ROS Agent の実行

micro-ROS Agent を実行し、4WDS ローバーと通信を行います。

・有線シリアル接続の場合

先ほど確認した VS-WRC058 のデバイス名を引数“--dev”に設定します。ここでは”/dev/ttyUSB0”だった場合の例を示します。以下のコマンドをシェルで実行してください。

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyUSB0 --baudrate 921600
```

ここで“—baudrate 921600”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、4WDS 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 アドレスの設定は不要ですが通信ポートの設定が必要となります。VS-WRC058 に書き込んだ Arduino スケッチ上で設定した通信ポートと同じ値を引数“--port”に設定してください。ここでは”8888”になっています。

4WDS 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。

```
ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888
```

8. メッセージの仕様

4WDS 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、4WDS ローバーへの移動指令を記載するメッセージです。4WDS 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 8-1 に、“/rover_twist”の詳細を示します。4WDS 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。4WDS ローバーは全方向移動台車ですので、x 軸,y 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”と“linear.y”にそれぞれ並進移動速度[m/s]を入力し、“angular.z”に旋回速度[rad/s]を入力してパブリッシュすることで、4WDS ローバーをコントロールすることができます。

表 8-1　/rover_twist の詳細

メッセージ名	/rover_twist
型	geometry_msgs/msg/Twist
内容	linear: 　x:　　//　x 軸方向の並進移動速度指令値[m/s]（float64） 　y:　　//　y 軸方向の並進移動速度指令値[m/s]（float64） 　z:　　//　不使用 angular: 　x:　　//　不使用 　y:　　//　不使用 　z:　　//　z 軸周りの旋回速度指令値[rad/s]（float64）

〔パブリッシュ〕

- /rover_sensor

“/rover_sensor”は、4WDS 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 8-2 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 8-2　/rover_sensor の詳細

メッセージ名	/rover_sensor
型	std_msgs/msg/Int16MultiArray
内容	layout: 　dim: [] 　data_offset: 0 data: [s0]　//　VS-WRC058 の MU16_IM_DI data: [s1]　//　バッテリー電圧[mV]

- /rover_odo

“/rover_odo”は、4WDS ローバーの現在速度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、4WDS 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 8-3 に“/rover_odo”の詳細を示します。“/rover_odo”は、“rover_twist”と同じ、geometry_msgs/msg/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/msg/Odometry 型です。

表 8-3　/rover_odo の詳細

メッセージ名	/rover_odo
型	geometry_msgs/msg/Twist
内容	linear: 　x:　　//　x 軸方向の並進移動速度[m/s]（float64） 　y:　　//　y 軸方向の並進移動速度[m/s]（float64） 　z:　　//　不使用 angular: 　x:　　//　不使用 　y:　　//　不使用 　z:　　//　z 軸周りの旋回速度[m/s]（float64）

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

〒555－0012　大阪市西淀川区御幣島 2-15-28

www.vstone.co.jp