ナノローバー
ROS 制御モード

取扱説明書
（2019.11.20）

Vstone®
ヴイストン株式会社

このたびは、ナノローバー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 2
2　ROS について 3
2.1　ROS の概要 3
2.2　ROS が提供する機能 3
2.3　ROS に関する情報の集め方 5
3　ナノローバーのセットアップ 6
4　ROS のインストール 7
5　ナノローバー用サンプルパッケージのセットアップ 11
5.1　catkin ワークスペースの作成 11
5.2　サンプルパッケージのダウンロードとビルド 12
5.3　サンプルパッケージの内容について 13
6　ナノローバーと ROS の接続 14
6.1　rosserial のインストール 14
6.2　ナノローバーと ROS の接続 15
6.3　ナノローバー用メッセージの仕様 19
7　ゲームパッド操作サンプル 21
8　マウス（タッチパッド）操作サンプル 26
9　動作確認バージョンについて 27

1　はじめに / 注意事項

本書は、ナノローバー を ROS で制御する際の取扱説明書兼チュートリアルです。ナノローバー本体の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、ナノ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　ナノローバーのセットアップ

ナノローバー本体側のセットアップを行います。必ず「ナノローバー 取扱説明書」および「ナノローバー　取扱説明書　~ユーザープログラム編~」をご確認いただいたうえで作業を行ってください。

ナノローバーを ROS で制御するためには、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、ナノローバー側のセットアップを行ってください。

① 「ナノローバー 取扱説明書 ~ユーザープログラム編~」に従い、ナノローバーを 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_wrc021_nanorover＞nanorover_common を開いてください。

③ スケッチ内の指示に従い、「WRC021_WIRELESS_MODE」,「WRC021_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、ナノ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 69 行目以降の 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ナノローバーに書き込んでください。

4 ROS のインストール

ROS 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

Ubuntu 16.04 をお使いの場合は ROS Kinetic を、Ubuntu 18.04 をご使用の場合は ROS Melodic をインストールする必要があります。

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。

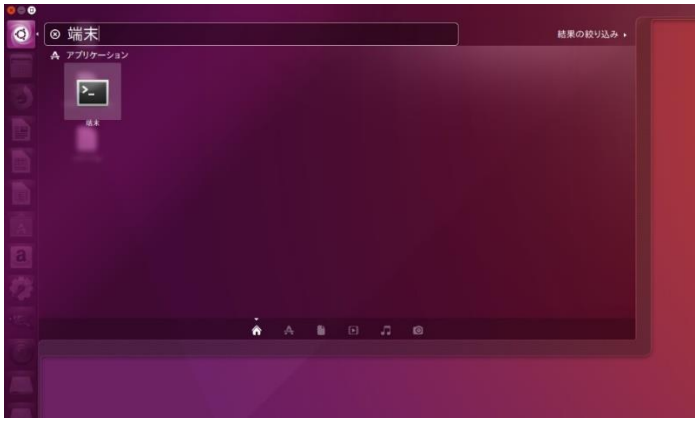

図 4-1 コンピュータの検索

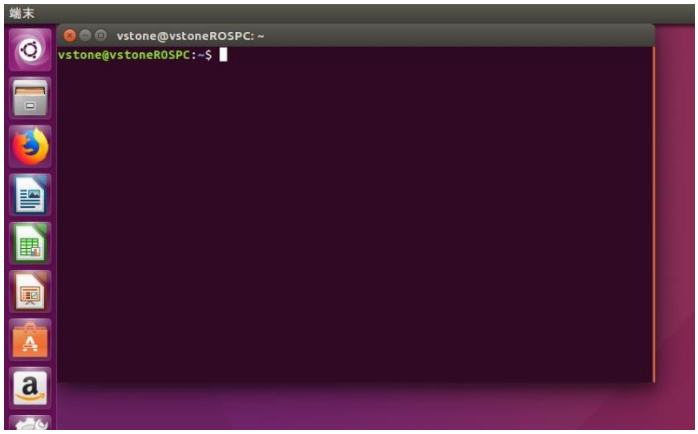

図 4-2 端末（シェル）

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 鍵を設定する

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

④ 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。

⑤ 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
```

⑥　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
```

⑦　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 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
```

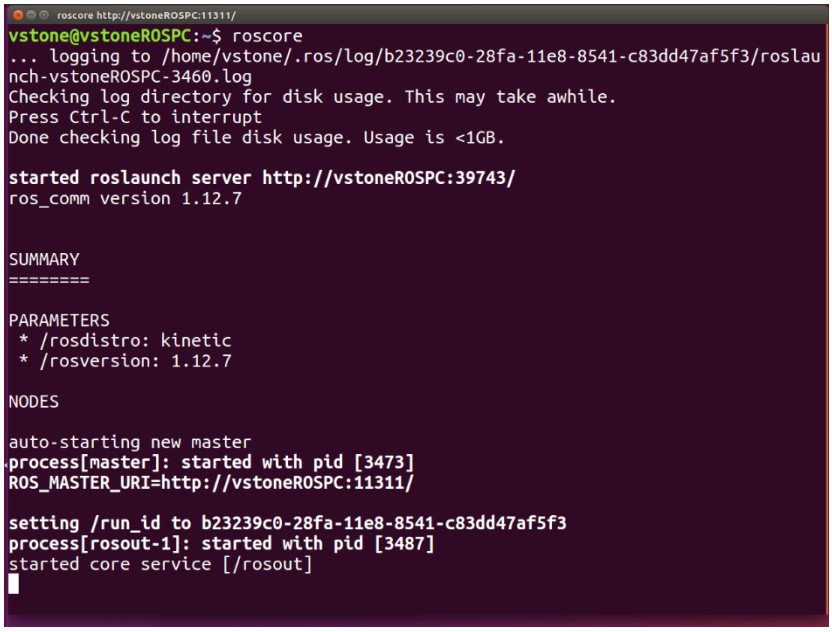

図 4-3 roscore 起動時のシェル

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　ナノローバー用サンプルパッケージのセットアップ

ナノローバーを 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/NanoRover
```

・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

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

- ノード

  nanorover_samples 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ナノローバーを走行させられるノードです。「pub_odom」は、ナノ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 nanorover_samples/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。

  nanorover_samples 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。

- 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

6 ナノローバーと ROS の接続

ナノローバーと 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

ナノ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、ナノローバーは 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 ナノローバーと ROS の接続

ナノローバーと PC を USB ケーブルまたは 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、ナノローバーと ROS を接続するたびに行う必要があります。

〔ナノローバーと ROS の接続〕

ナノローバーと ROS を接続します。

① ナノローバー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ナノローバーのデバイス名を確認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。

ナノローバーと 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、ナノローバー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

656] 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 6-1 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。これはナノローバー上の 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。ナノローバー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」がナノローバー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）

ナノローバー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/ナノローバーのデバイス名
```

本書の例では、ナノローバーのデバイス名が「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、ナノ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、先ほど確認したナノローバー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、ナノ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。ナノ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 6-2 に示します。これと同じ応答が得られれば、ナノローバーと 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、ナノ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**
  ナノ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。
- **ナノ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。ナノ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。

```
[ERROR] [1553585292.859126]: Lost sync with device, restarting...
[INFO] [1553585292.861635]: Requesting topics...
[INFO] [1553585292.933221]: Setup publisher on rover_sensor [std_msgs/Int16Multi
Array]
[INFO] [1553585292.937173]: Setup publisher on rover_odo [geometry_msgs/Twist]
[ERROR] [1553585307.932183]: Lost sync with device, restarting...
[INFO] [1553585307.933795]: Requesting topics...
[INFO] [1553585308.005898]: Setup publisher on rover_sensor [std_msgs/Int16Multi
Array]
[INFO] [1553585308.009193]: Setup publisher on rover_odo [geometry_msgs/Twist]
```

図 6-6　rosserial の応答（エラー4）

「Lost sync with device, restarting…」は、ナノローバーと 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、ナノローバーと 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、ナノローバー側で spinOnce()関数が呼び出されることでリセットされます。

- **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。

6.3 ナノローバー用メッセージの仕様

ナノローバーは 1 つのメッセージをサブスクライブ（購読）し、1 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、ナノローバーへの移動指令を記載するメッセージです。ナノ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。ナノローバーの座標系は、前方向を x 軸の正方向、右方向を y 軸の正方向にとる右手系で定義しています。ナノローバーは全方向移動台車ですので、x 軸方向への移動および y 軸方向への移動と、z 軸周りでの旋回が行えます。

よって、“linear.x”, “linear_y”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、ナノ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:　　//　y 軸方向の並進移動量指令値（float64） 　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_odo

  “/rover_odo”は、ナノ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、ナノ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 6-2　/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:　　//　y 軸方向の並進移動量（float64） 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ナノ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。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/nanorover_samples/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

ナノ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、ナノローバーと 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 nanorover_samples joycon.launch
```

ナノ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、ナノ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/nanorover_samples/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=”nanorover_samples” 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_x	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	2
scale_linear	移動量のゲイン（1.3 程度までに留めること） 移動量＝scale_linear×アナログスティック入力値	0.08
scale_angular	旋回量のゲイン（6.0 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”、“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、ナノローバーと 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 nanorover_samples 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。ナノ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、ナノローバーをゲームパッドで操作します。ナノ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 はナノ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。

6.3 項に従って PC とナノ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 nanorover_samples joycon.launch
```

ゲームパッドをゆっくりと操作し、ナノ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、ナノ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_teleope 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_teleope 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-teleope
```

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。ナノ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 はナノ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、ナノローバーと 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 nanorover_samples 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。

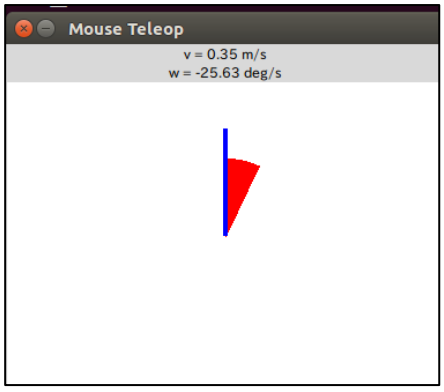

図 8-1　Mouse Tekeop の GUI

6.3 項に従って PC とナノローバーを接続します。その後ナノローバーの電源を ON にしてマウスを操作すると、ナノローバーが走行します。

9　動作確認バージョンについて

ナノ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 9-1 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

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