UR10eをROS2で利用するための環境設定

作成日：2026年8月19日（水）
作成部：和歌山県工業技術センター、ものづくり支援部

本資料の構成

本資料はUniversal Robots製のUR10eをROS2を用いて利用したいと考えている方を主な対象としています。
本資料は以下の章立てで構成されています。

また、ROS2をあまり知らない方に対しても資料の序盤で簡単にどのようなことができるのかについて説明していますので、そちらもご覧いただければと思います。

1 はじめに

和歌山県工業技術センターでは協働ロボットの一つであるUR10eを保有しています。
こちらはROS2から操作が可能です。ROS2はロボット制御用のミドルウェアとして無料で利用可能で、世界中で研究・開発目的に広く利用されています。今回はUR10eをROS2を通して制御するためのドライバを導入したため、その手順を公開します。

システム構成

システムの構成について説明します。
システムは協働ロボットであるUR10e（以下、UR）とURを外部制御するホストPCの 2 つから構成されます。
今回はホストPCにJetson Orin Nano（以下、Jetson）を利用します。

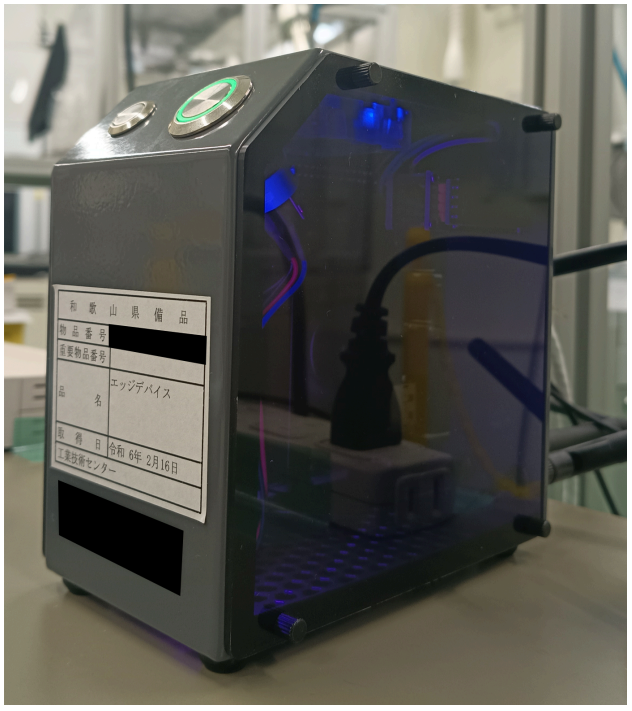
機器の種類

ロボット	Universal Robots製	UR10e
ホストPC	Nvidia製	Jetson Orin Nano

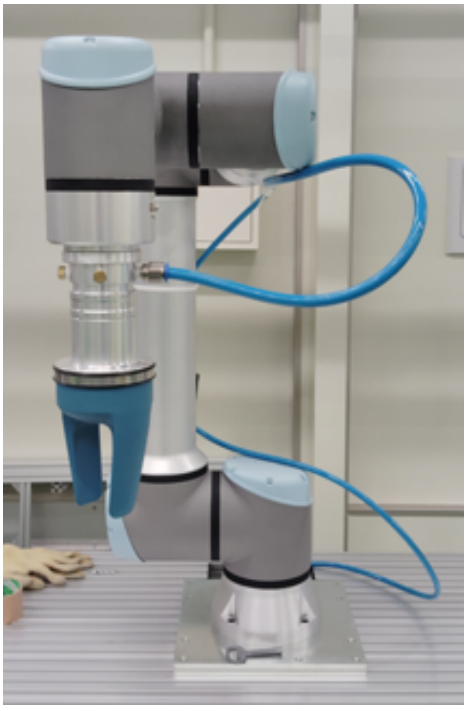
OS、バージョン

OS	Ubuntu 22.04
ROS2	Humble

JetsonとURはLANケーブルを用いて、直接もしくはハブを通してEthernet接続をします。



Jetson Orin Nano



UR10e

2 ROS2のインストール

ROS2について

ROSは Robot Operating System の頭文字をとったものです。
OSと名前に入っていますが、WindowsやUbuntuのようなOSではなく、それらの上で動作するミドルウェアとしての役割を果たします。ROS2はロボットアプリケーションの開発に便利な通信システム・ライブラリ・ツール群を提供しており、ロボット開発のハードルが下がります。代表的なものとして移動ロボットの自律移動を支えるSLAMおよびNavigation、ロボットアームの軌道計算のためのパッケージ等が利用可能です。
一般的にROS2を利用するメリットとして取り上げられるのは通信システムとしての機能です。通常のロボットはそれぞれメーカ独自のプログラミング環境を持っており、他のロボットやセンサと連携するためにはそれぞれのプログラミング環境が持つインターフェースに合わせてプログラムを作成する必要があり、複雑になりがちです。しかしROS2は各ロボット・センサのインターフェースを抽象化し、ROS2基盤上で同一の通信方式で連携が可能になるため、異なるメーカのロボット・センサを同様に扱うことができます。これにより、複数台のロボットの連携や開発途中でのシステムの変更などが行いやすくなります。
さらにROS2ではプログラミング言語に Python / C++ が主に使用されます。このことから独自で作成したPythonプログラムを用いてロボットを制御する、というようなことが非常に容易にでき、ロボットの利活用の幅が広がります。
従来はROSが用いられていましたが、現在はその後継であるROS2に移行が進んでいます。この移行は主に通信方式を刷新することでリアルタイム性を向上させ、ROS2の商用・製品利用を促進することを目的としています。

ROSについてより詳しく知りたい方は、以下の公式サイトによる説明をご参照ください。
About ROS

バージョンの確認

ROS2は基本的にUbuntu上で動作しますが、各Ubuntuのバージョンに対応した異なるROS2のバージョンが用意されています。
そのため、それぞれに適したバージョンのROS2をインストールする必要があります。
インストールするROS2のバージョンを決めるため、まずUbuntuのバージョンを以下のコマンドで確認します。

```
lsb_release -a
```

以下の出力が得られました。

```
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.5 LTS
Release:        22.04
Codename:       jammy
```

この結果からUbuntu 22.04がインストールされていることが分かります。
これに対応したROS2のバージョンはHumbleであるため、これをインストールします。

ROS2 Humbleをインストールする

インストール手順は公式サイトで紹介されているものに従います。
ROS2 Humble Install Official Document

ロケール設定

ROS2のインストールにはUTF8のサポートが必要になるため、ロケールの設定を行います。
公式では英語になっていますが、日本語に変更します。

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen ja_JP ja_JP.UTF-8
sudo update-locale LC_ALL=ja_JP.UTF-8 LANG=ja_JP.UTF-8
export LANG=ja_JP.UTF-8

locale  # verify settings
```

最後のコマンドの出力でUTF8になっていれば成功です。

リポジトリ登録

apt経由でROS2をインストールする場合、ROS2のaptリポジトリを追加する必要があります。
まず、Ubuntu Universe repositoryを有効化します。

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

ros2-apt-sourceをインストールします。
これでJetsonにROS2リポジトリが設定されます。

```
sudo apt update && sudo apt install curl -y
export ROS_APT_SOURCE_VERSION=$(curl -s https://api.github.com/repos/ros-
infrastructure/ros-apt-source/releases/latest | grep -F "tag_name" | awk -
F'"' '{print $4}')
curl -L -o /tmp/ros2-apt-source.deb "https://github.com/ros-
infrastructure/ros-apt-
source/releases/download/${ROS_APT_SOURCE_VERSION}/ros2-apt-
source_${ROS_APT_SOURCE_VERSION}.$(. /etc/os-release && echo
```

```
${UBUNTU_CODENAME:-${VERSION_CODENAME}})_all.deb"
sudo dpkg -i /tmp/ros2-apt-source.deb
```

ROS2のインストール

リポジトリの設定が完了したので、ROS2をインストールすることができます。

```
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-desktop
```

環境設定

ROS2のインストールが完了しました。
`ros2` コマンドを使えるようにするために、セットアップ用スクリプトを読み込んで環境を設定します。
これをしないと、 `ros2` コマンドが使えません。

```
source /opt/ros/humble/setup.bash
```

これはROS2利用時に毎回実行する必要があるため、一般的には `.bashrc` の一番下に上の行を追記します。

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

これで端末が起動するたびにROS2の環境が設定されるため、毎回手動で環境を設定しなくてもROS2を利用することができるようになります。

3 ロボットのROS2用ドライバのインストール

JetsonにURのROS2ドライバをインストールしていきます。
ドライバは公式から公開されています。
以下が公式のgithubリポジトリです。
Universal_Robots_ROS2_Driver

今回はこちらで指示されている通り、aptを用いたインストールを行います。

```
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-ur
```

【注意】
arm64のアーキテクチャを使用している場合、ドライバパッケージのバージョンが2.14.0であることを確認してください。

4 PolyScopeの更新

このドライバはUniversal_Robots_Client_Libraryを使用しています。
Universal_Robots_Client_Libraryを使うためにはPolyScopeのバージョンが5.9.4以上であることが必要条件です。
Universal_Robots_Client_Libraryの必要条件

現在インストールされているバージョンを確認したところ、5.6.0であったため更新を行います。

PolyScopeのバージョン確認方法

1. 右上のハンバーガーを押します。
2. バージョン情報、を押します。
3. URSoftwareの後に続く数字がバージョン情報なのでそれを読み取ります。
   ※ 今回はURSoftware 5.6.0.90886 (Nov 15 2019) と書かれており、PolyScopeのバージョンは5.6.0ということがわかります。

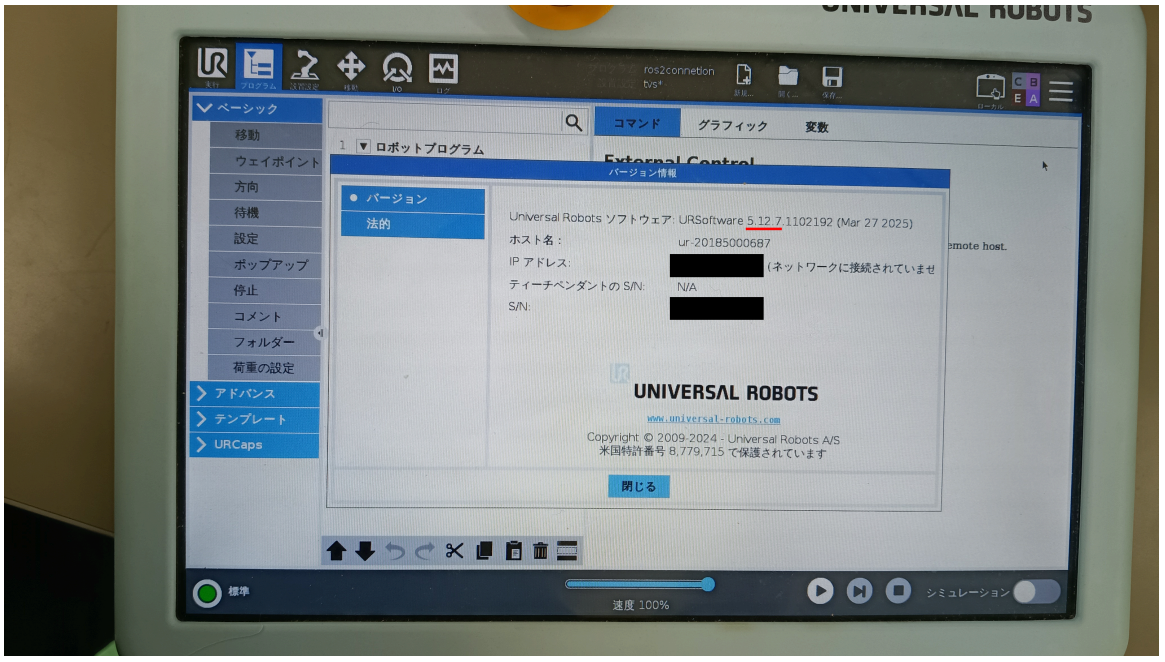
PolyScopeの更新手順

バージョンを5.12.7に更新します。

手順

1. Polyscope Software Update - SW 5.12.7 LTS - e-Series にアクセスし、`ursys-CB5.0-full-5.12.7-build11.0.2192.urup` をダウンロードします。
2. ダウンロードしたファイルを空のUSBメモリに入れます。
   ※USBメモリは8GB以上、32GB以下が好ましいです。
3. USBファイルをティーチングペンダントに挿します。
4. ティーチングペンダントからインストールボタンを押して更新します。
5. 再起動するまで待ちます。
6. 再起動後、ロボットの起動ボタンを押すとファームウェアが更新されます。

完了後バージョンが変更されたことを確認します。



PolyScopeのバージョン確認画面です。

参考：[How to Update Polyscope 5]

5 UR10eの外部制御ライブラリの導入

URCap External Controlの導入

URCap（URの拡張プログラム）の一つである External Control を導入する必要があります。

手順

1. Universal_Robots_ExternalControl_URCap から、`externalcontrol-1.0.5.urcap` をダウンロードします。
2. ダウンロードしたファイルを空のUSBメモリに入れます。
3. ティーチングペンダントにて、ロボットの設定画面から `URCaps` メニューを開きます。
4. USBメモリをティーチングペンダントに挿入します。
5. ＋ボタンを押します。
6. ファイル画面から `externalcontrol-1.0.5.urcap` を選択し、`open` ボタンを押します。
7. `Restart` ボタンを押し、ロボットを再起動します。

参考：SMC 協働ロボット用エアグリッパ RMH* シリーズ -ソフトウェア(URCap)編-

6 ネットワーク設定

JetsonとURをLANケーブルで直接つなぎます。

Jetson側の設定

IPv4 address	{host_ip}
Netmask	255.255.255.0

UR側の設定

ティーチングペンダントのIPアドレス設定

ティーチングペンダントから以下のように設定

IP address	{robot_ip}
Subnet mask	255.255.255.0
Default gateway	0.0.0.0
Preffered DNS server	0.0.0.0
Alternative DNS server	0.0.0.0

External Controlでの、ホストPCのネットワーク設定

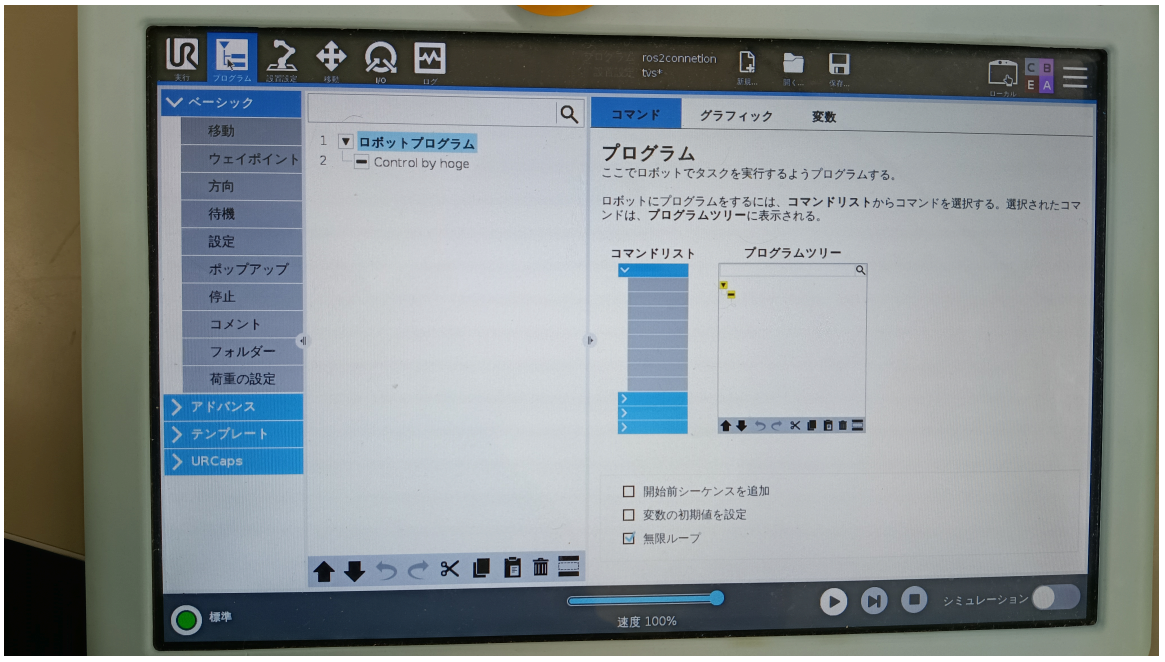
ティーチングペンダントの `設置設定 > URCaps > External Control` にて

Host IP	{host_ip}
Custom port	50002
Host name	hoge

※Host name は何でもよいです。
`{host_ip}` および `{robot_ip}` には任意のものを設定します。

External Controlを起動するスクリプトの作成

これを設定したExternal Controlを呼び出すためURスクリプトを別途作成します。
今回は `ros2connection.urp` という名前にします。
プログラムはティーチングペンダント上で作成します。
内容は URCaps から External Control を呼び出すだけです。
作成したプログラムは以下になります。



ros2connect.urpの画面です。

ここまでで一度JetsonからURに向けてpingコマンドで通信が確立しているか確認をします。

```
ping {robot_ip}
```

7 URドライバの起動

URとネットワーク接続ができていることが確認できれば、ドライバの起動に移ります。
ティーチングペンダントにて、URが `Remote Control` モードではなく、 `Local Control` モードに切り替わっていることを確認してください。（ティーチングペンダント右上）

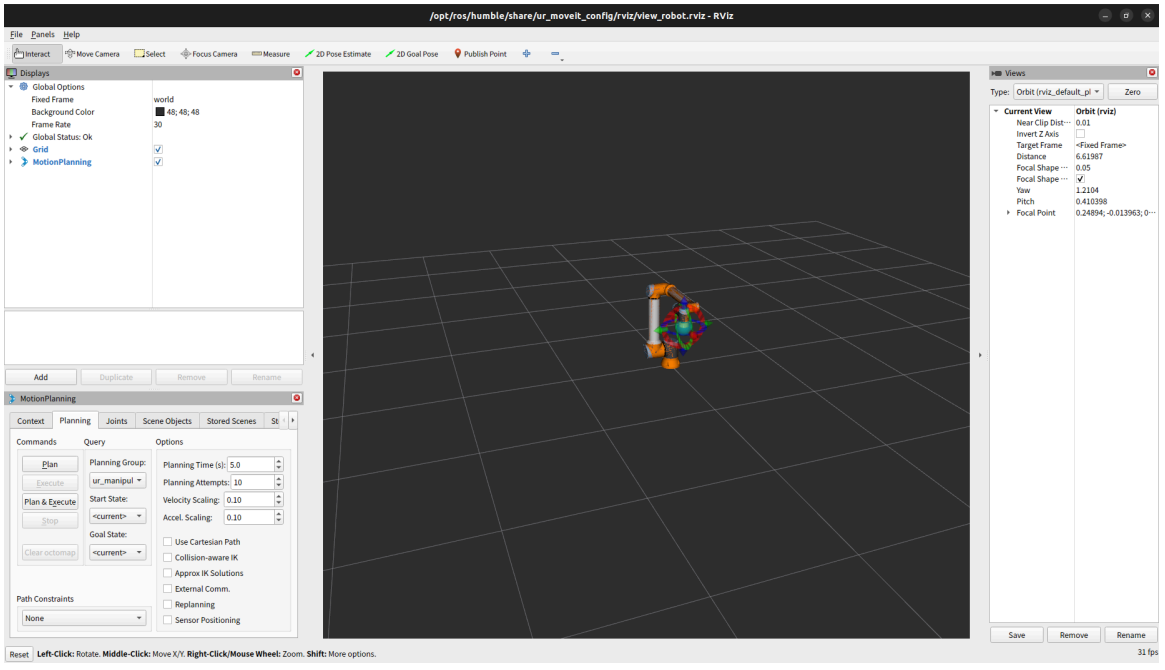
起動手順

1. ur_ros2_driverの起動（PC側、Terminal 1）

```
ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur10e robot_ip:=
{robot_ip}
```

2. `ros2connectionurp` の起動（ロボット側、ティーチングペンダント）
3. ros2のMoveItの起動（PC側、Terminal 2）

```
ros2 launch ur_moveit_config ur_moveit.launch.py ur_type:=ur10e
launch_rviz:=true
```



MoveItを起動した際のRvizのGUI画面です。

上はMoveItを起動した際のRvizのGUI画面です。
マウスでアーム先端の球をクリック&ドラッグすることで、アームの到達先を設定することができます。
到達先を設定し、左下のMotionPlannningと書かれている枠内にて `Plan & Execute` を実行するとロボットが目的地に向かって動きます。
ロボットが無事動けばドライバの導入が完了になります。

8 まとめ

以上でUR10eをROS2で利用するための環境構築の手順の説明とさせていただきます。
今後は外部のセンサやプログラムと連携させ、自動化タスクに取り組んでいくことを目指します。

「実際にROS2を用いてロボットを動かしたい」、「ROS2を用いてセンサを利用してみたい」など、本記事の内容に興味を持ってくださった場合は **和歌山県工業技術センター ものづくり支援部** までご連絡くださいますよう、よろしくお願いいたします。