

ROS 2講習 第一回

4S 野口 史遠

参考文献:ROS2とPythonで作って学ぶAIロボット入門

ROS 2の基礎知識

ROS 2は多くの実行中のプログラム間で通信してロボットを動かす。このプログラムのことを **ノード (node)** という

つまりロボットを動かすにはROS 2の通信を理解しないといけない。

通信方法

ros2の通信方法には

トピック通信 **サービス通信** **アクション通信** **パラメーター通信**

の4つがある。

1. トピック (Topic)

概要

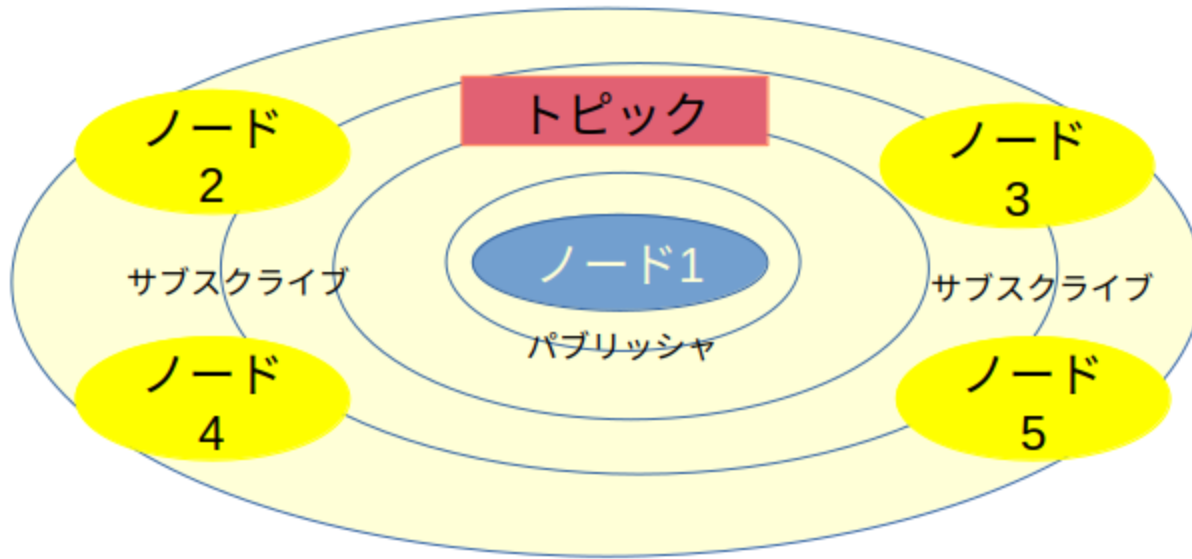
- 1方向・非同期通信

用途

多くのノードに同じデータを送る場合

センサーデータの配信・受信

カメラ, LIDARとか



イメージ：テレビ

テレビ局に該当するデータの送り手を **パブリッシャ(publisher)**
視聴者に該当する受け手を **サブスクライバ(subscriber)** という
番組のチャンネルが **トピック** といいデータを **メッセージ** と呼ぶ

2. サービス (Service)

概要

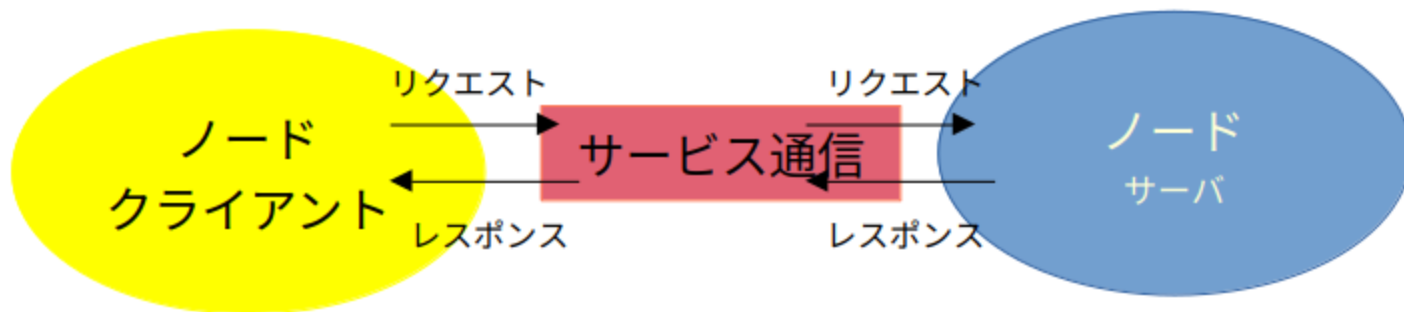
- 双方向・同期通信

用途

すぐに終了するタスク

電源の起動確認，モード切替

カメラ，LIDARなどのセンサーの動作モード切替



イメージ：Google検索

データの送り手を クライアント (client)

そのデータを処理して応答する側を サーバ(server) という。

クライアントからサーバに仕事を依頼することを リクエスト ，そのデータを リクエストメッセージ ，サーバが結果をクライアントに返すことを レスポンス ，そのデータを レスポンスメッセージ と呼ぶ

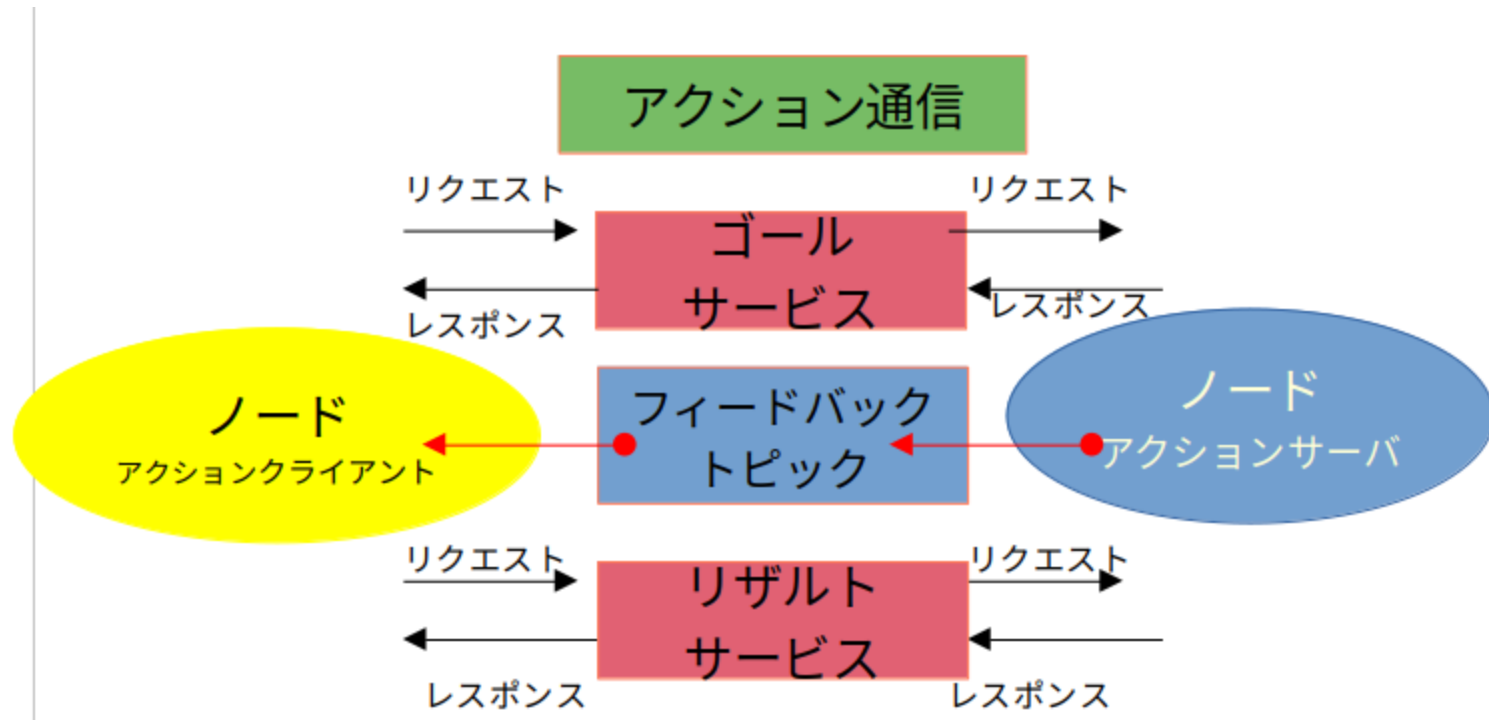
3. アクション (Action)

概要

- 双方向・同期通信，サービス通信より高性能

用途

ナビゲーションなどタスク終了までに時間がかかり，途中経過も知りたい複雑なタスクのとき



サービスとトピックの両方を使用している

4. パラメータ (Parameter)

概要

- ノードの振る舞いを変えるパラメータ専用通信

用途

ロボットの動作などを実行中に変えたいとき

コマンド

ROS 1 はLinux環境でしか使用できなかったため,ROS2もLinux文化の影響が強く GUIではなく、コマンドを使用するCUIアプリを使用することが一般的

コマンドが使えないとROS 2を使うことは非っ常にキビシー

ROS2ではコマンドとしてros2コマンドを使う
どんなのがあるか調べてみよう

```
ros2 --help
```

```
ros2 サブコマンド --help
```

サブコマンド	内容
action	アクション通信関連
bag	Rocbag関連.rosbagはセンサ情報を記録，再生する強力なツール
launch	ローチンファイルを実行. ローチンファイルは複数ののードやその設定などをまとめて起動するために使うファイル
node	ノード関連
pkg	パッケージ関連
run	パッケージの実行可能ファイルを実行
service	サーブス通信関連

初めてのROS2プログラミング

プログラムの流れ

1. ワークスペースの作成：作業スペースを作る，パッケージはこの中に
2. パッケージの作成：ROS2はパッケージ単位でプログラムを作る
3. ソースコードの作成：ノードのソースコードを作る
4. ビルド：ノードの実行ファイルを作る
5. 設定ファイルの反映：ノードが実行できるように環境を整える
6. ノードの実行

1.ワークスペースの作成

ROS2ではプログラムをパッケージと呼ばれる単位でつくります．はじめにパッケージを保存するための作業用ディレクトリを作らなければなりません．それがワークスペースです．1つのワークスペースに何個でもパッケージを作れます．

また，自作パッケージを使うにはROSシステムのワークスペース環境（アンダーレイ）と自作ワークスペース環境(オーバーレイ)を整えなければならない．

- 設定ファイルの実行

`source` コマンドでアンダーレイの設定ファイルを実行

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

- ワークスペース用のディレクトリ作成

`mkdir` コマンドでディレクトリを作成

```
mkdir -p ~/hazimete/src
```

2. パッケージの作成

パッケージは自作したROS2コードの入れ物．パッケージを使うことでかんたんにビルでできたり一般公開できるようになる

ファイル構成

- setup.py: パッケージのインストール法が書かれたファイル
- setup.cfg: パッケージ実行ファイルがある場合に必要なファイル (ros2 run)
- package.xml: パッケージに関する情報
- パッケージ名: パッケージと同じ名前のディレクトリ

- パッケージの作成

`cd` コマンドでディレクトリを移動する

```
cd ~/hazimete/src
```

つぎに `ros2 pkg create` コマンドでパッケージを作成

```
ros2 pkg create --build-type ament_python hazimetepkg
```

なお、

```
ros2 pkg create --build-type ament_python --node-name hazimetenode hazimetepkg
```

とするとノードも作れる。

3. ソースコードの作成

現在の中身

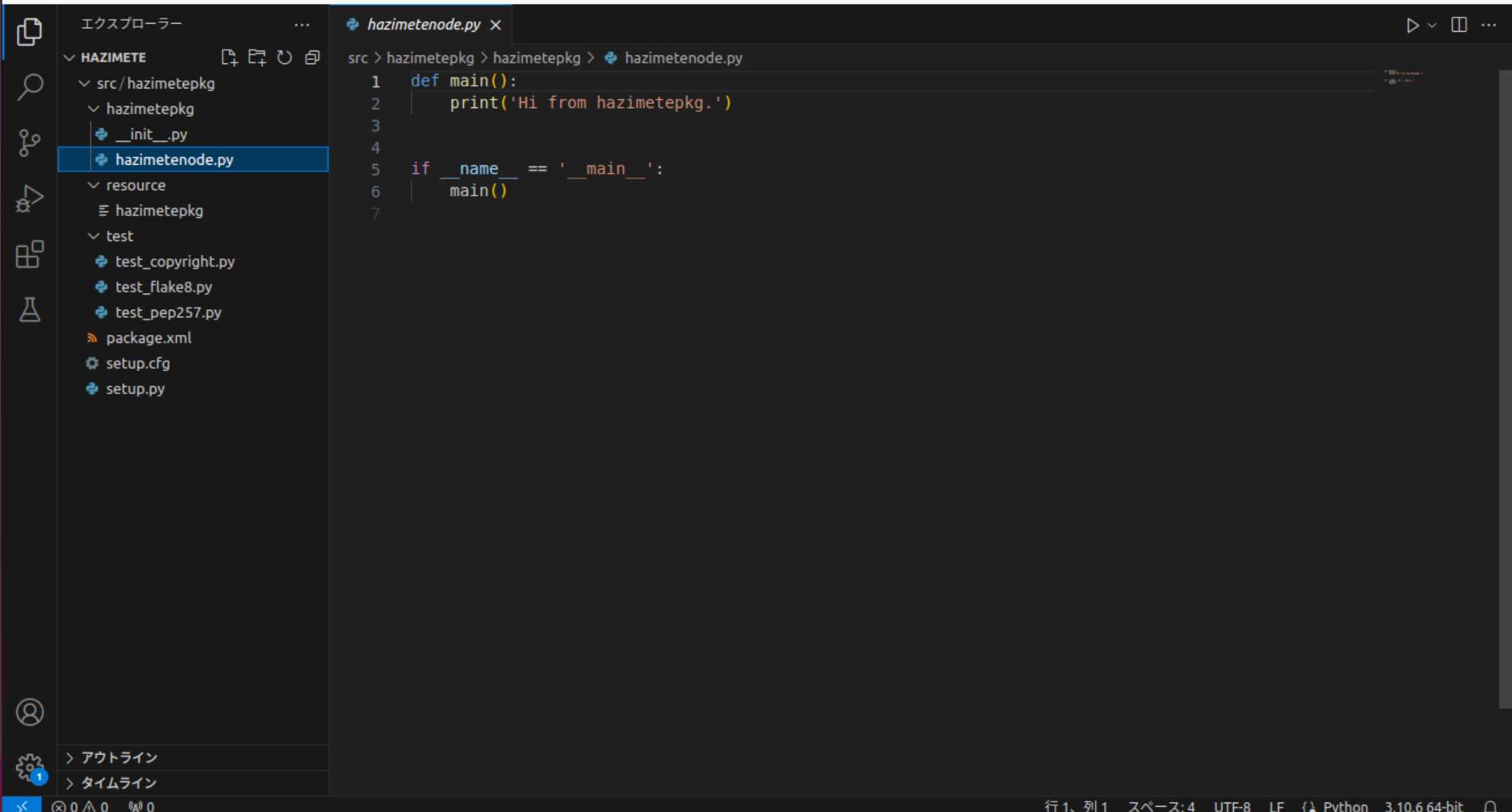
```
altair@altair-ThinkPad-L380:~/hazimete/src$ exa -T
.
├── hazimetepkg
│   ├── hazimetepkg
│   │   ├── __init__.py
│   │   └── hazimetenode.py
│   ├── package.xml
│   ├── resource
│   │   └── hazimetepkg
│   ├── setup.cfg
│   ├── setup.py
│   └── test
│       ├── test_copyright.py
│       ├── test_flake8.py
│       └── test_pep257.py
```

ここからは、使いやすいvscodeを使います

```
cd ~/hazimete
code .
```

とするとvscodeが立ち上がります。 `.` はカントディレクトリ

ファイル 編集 選択 表示 移動 実行 ターミナル ヘルプ



The image shows the Visual Studio Code interface. On the left, the File Explorer sidebar is open, showing a project structure. The file 'hazimetenode.py' is selected. The main editor area displays the code for 'hazimetenode.py'.

File Explorer (Left):

- HAZIMETE
 - src/hazimetepkg
 - hazimetepkg
 - __init__.py
 - hazimetenode.py**
 - resource
 - hazimetepkg
 - test
 - test_copyright.py
 - test_flake8.py
 - test_pep257.py
 - package.xml
 - setup.cfg
 - setup.py

Editor (Right):

```
src > hazimetepkg > hazimetepkg > hazimetenode.py
1 def main():
2     print('Hi from hazimetepkg.')
3
4
5 if __name__ == '__main__':
6     main()
7
```

Bottom Status Bar:

行 1、列 1 スペース: 4 UTF-8 LF Python 3.10.6 64-bit

- package.xmlの編集

コメントアウトが変更する場所

```
<name>hazimetepkg</name> # 1.パッケージ名  
<version>0.0.0</version> # 2.パッケージのバージョン  
<description>TODO: Package description</description> # 3.パッケージの説明  
<maintainer email="Altairu@github.com">altair</maintainer> #4.保守者のメールアドレス  
<license>TODO: License declaration</license> # 5. パッケージのライセンス
```

- setup.pyの編集

setup.pyを変更しないとノードを実行できません

コメントアウトしている7つを変更しなければならない。

一般公開しない場合は7つ目のみ要変更

```

from setuptools import find_packages, setup

package_name = 'hazimetepkg' # 1.パッケージ名

setup(
    name=package_name,
    version='0.0.0', # 2.パッケージのバージョン
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='altair', #3.保守者
    maintainer_email='Altairu@github.com', #4.保守者のメールアドレス
    description='TODO: Package description', #5.パッケージの説明
    license='TODO: License declaration', #6. パッケージのライセンス
    tests_require=['pytest'],
    entry_points={ #7.エントリポイント
        'console_scripts': [
            #ノード名=パッケージ名.ノード名:main
            'hazimetenode = hazimetepkg.hazimetenode:main'#pythonファイルごとに必要
        ],
    },
)

```

エントリポイントを自動生成したければ `--node-name` のオプションをつければよし

- ソースコード作成

本来ならここでソースコードを作るが自動生成されたものを使ってみましょう
hazimetenode.pyは自動生成されたプログラムです.

```
def main():  
    print('Hi from hazimetepkg.')
```

if __name__ == '__main__':
 main()

4.ビルド

パッケージを作るためにビルドしないといけない

ROS 2 ではビルドシステムとして `ament` を使い，ビルドコマンドとして `colcon` を使う

ちなみにワークスペース名直下のディレクトリしかビルドできない

```
cd ~/hazimete  
colcon build
```

5. 設定ファイルの反映

パッケージののーどを実行する前に，ROS2の設定ファイルをsourceコマンドで反映させる

- アンダーレイ設定ファイルの反映

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

- オーバーレイ設定ファイルの反映

自作ワークスペースの設定を反映させる．これで作成したワークスペースの場所がわかるようになりノードを実行できる

```
source ~/hazimete/install/setup.bash
```

6. ノードの実行

```
ros2 run hazimetestpkg hazimetenode
```

```
altair@altair-ThinkPad-L380:~/hazimete$ ros2 run hazimetestpkg hazimetenode  
Hi from hazimetestpkg.  
altair@altair-ThinkPad-L380:~/hazimete$
```


次はROS2プログラムの作り方

前提レベル：python 3 のプログラムがかけます。（クラス）