2025年度 株式会社XACK寄付講座

ROS2を使いロボットカーを動かして自動運転の基礎技術を学ぼう！

第1回

2025年05月07日

講座の流れ

- 本講座の各回の内容は下記の通りです。

日時	内容
5/07(水) 15:10~16:50 講義	環境構築、課題説明
5/14(水) 15:10~16:50 演習	課題実現に向けた実装
5/21(水) 15:10~16:50 演習	課題実現に向けた実装
5/28(水) 15:10~16:50 講義	中間発表会
6/04(水) 15:10~16:50 演習	課題実現に向けた実装
6/11(水) 15:10~16:50 演習	課題実現に向けた実装
6/18(水) 15:10~16:50 講義	最終発表会、講評

- 普段は東京にいます。
- 授業時間外に質問がある場合は yajima@xack.co.jp までメールをください。

本日の流れ

- ROS2の簡単な説明
- 講座で使用するロボットカーの説明
- 課題の説明
- ロボットカーの開発・動作環境のセットアップ
- ROS2のプログラムの解説

ROS2とは

- ロボット開発のためのオープンソースのソフトウェアフレームワーク
- ロボット毎の差異を制御要素(ROS2ではトピックという)毎に抽象化し同一プログラムで多種のロボットを操作可能な仕組みを提供
- 通信機能を内包し遠隔でロボットを操作可能な仕組みを提供
- 通信方式にPublisher/Subscriberモデルを採用し多数のロボットからの情報収集/操作可能な仕組みを提供

ROS2を使わない場合

サーバーPG1

サーバーPG2

遠隔制御したければ自力で実装する必要がある

ロボットA用PG

ロボットA用ドライバー/API

ロボットB用PG
API呼出

ロボットB用ドライバー/API

ロボットC用PG

ロボットC用ドライバー/API

ロボットA

API呼出はロボット毎に異なる。通信にも影響するかは自力実装次第

ロボットB

ロボットC

ROS2を使う場合

サーバーPG1

ROS2

サーバーPG2

ROS2

通信はROS2が行う

ロボットA用PG

ROS2

ロボットA用ドライバー/API

ロボットB用PG

ROS2

API呼出

ロボットB用ドライバー/API

ロボットC用PG

ROS2

ロボットC用ドライバー/API

ロボットA

ロボットB

トピックの読書
(publish/subscribe)は
全プログラムで共通か
つ一度で全PGに届く

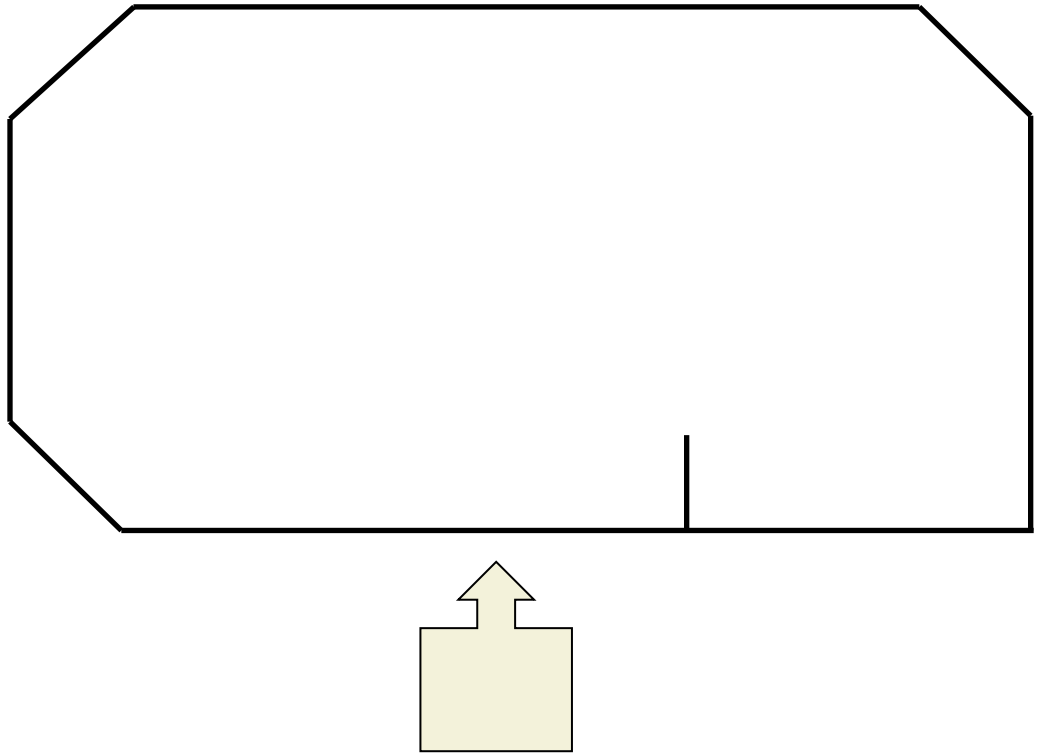
ロボットC

本講座で使用するロボットカー

- Raspbot V2
  - Raspberry Pi 5(Linux)
  - ROS2環境構築済み
- 各種センサー搭載

課題内容

- 線の外側からスタートし、線の上を一周し横線に到達したら停止するROS2を使ったプログラムを作成する



課題内容

- 中間発表会までにすること
  - 下記を盛り込んだ発表資料の作成
    - 中間発表時点のプログラムの作成状況と今後の見込みの説明
    - 中間発表時点のプログラムのデモ(見せられるものがあれば)
- 最終発表会までにすること
  - プログラムの完成
  - 下記を盛り込んだ発表資料の作成
    - 作成したプログラムの説明
      - どのような方法で線を検出するか
      - どのような方法で進む方向を決定するか
      - どのような方法で止まる場所を検出するか　など
- 提出物
  - 発表資料
  - プログラムのソースコード

ロボットカーの開発・動作環境のセットアップ

大まかな流れ

1. PCに必要なアプリケーションのインストール
2. ロボットカーに接続
3. ROS2を使用しない構成でロボットカーを操作
4. ROS2を使用する構成でロボットカーを操作

ロボットカーの開発・動作環境のセットアップ

1. 必要なアプリケーションのインストール
   - ロボットカーとはWi-Fiで直接接続します
   - そのため、ロボットカーと接続する前に必要なアプリケーションをインストールする必要があります
   1. SSHクライアント

      最近のWindowsだとデフォルトでインストールされています。

      コマンドプロンプトでsshと叩いてusageが出ればインストールされています。

```
C:¥Users¥yajima>ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
  ...
```

      インストールされていない場合、設定から「システム」>「オプション機能」>「機能を表示」を選択し、「OpenSSHクライアント」にチェックを入れてインストールできます。

      ※TeraTerm、PuTTY、RLoginなど、普段使用している端末エミュレーターがあればそれを使用しても構いません。

ロボットカーの開発・動作環境のセットアップ

1. 必要なアプリケーションのインストール
    2. VNCビューワー

       ロボットカーのデスクトップに接続する場合は必要です。インストールせずとも開発はできます。

       インストールする場合は「VNC Viewer」「TightVNC」「TigerVNC」などで検索してダウンロードしてください。

ロボットカーの開発・動作環境のセットアップ

ロボットカーの詳細な仕様や操作方法は下記の公式ページにまとまっています。

http://www.yahboom.net/study/RASPBOT-V2

2. ロボットカーに接続

    1. ロボットカーの電源ON

        ロボットカーの背面にあるスイッチを右に入れることで電源が付きます。

        起動するとOLEDディスプレイにIPアドレスが表示され、音が鳴ります。

    2. Wi-Fi接続

        ロボットカーはWi-Fiのアクセスポイントとして動作します。このアクセスポイントに接続することでロボットカーとPCが通信できるようになります。

        SSID: Raspbot数字(数字は1~6、ロボットカーについている付箋の番号)

        パスフレーズ: 12345678

ロボットカーの開発・動作環境のセットアップ

本資料およびソースコードは下記からダウンロード可能です。

https://github.com/TakashiYajima/2025kit

2. ロボットカーに接続

    3. SSH接続

    OLEDディスプレイに表示されたIPアドレス(デフォルト192.168.1.11)に対してSSH接続します。

    ユーザー名: pi

    パスワード: yahboom

```
C:¥Users¥yajima>ssh pi@192.168.1.11
pi@192.168.1.11's password:
  ...
pi@yahboom:~ $
```

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

サーバープログラムはなし

ここの部分を作る

ロボットA用PG	ロボットB用PG API呼出	ロボットC用PG
ロボットA用ドライバー/API	ロボットB用ドライバー/API	ロボットC用ドライバー/API

ロボットA

ロボットB

ロボットC

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

1. Dockerコンテナを起動

Raspbot V2ではロボットカーを操作するドライバーやROS2環境をDockerコンテナというアプリケーションとその動作環境を一緒にパッケージ化する技術を用いて提供しています。Dockerコンテナを動作させる環境(SSHでログインしたプロンプトがpi@yahboomになっている環境)のことはDockerホストと呼称します。まずはDockerコンテナを起動します。

```
pi@yahboom:~ $ ./docker_ros2.sh
  ...
root@yahboom:/#
```

Dockerコンテナは動作環境をパッケージ化して提供するため、その環境内で行った変更は、次回起動時に失われてしまいます。Raspbot V2が提供するDockerコンテナ環境では、/root/tempというパスのみ、永続的に変更を保持する場所として設定されています。

そこで以降の作業では/root/temp配下に移動し、その下にプログラムを作成していきます。

```
root@yahboom:/# cd /root/temp
root@yahboom:~/temp#
```

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

2. ROS2を使用せず直接APIを呼び出すプログラムを作成

下記コードを記載したファイルを/root/temp配下に作成します。

```
#!/usr/bin/env python3

from Raspbot_Lib import Raspbot
import time

car = Raspbot()
car.Ctrl_WQ2812_brightness_ALL( 255, 255, 255)
time.sleep(1)
car.Ctrl_WQ2812_brightness_ALL( 0, 0, 0)
```

このコードはRGBバーを白色に光らせた後、1秒待機し、RGBバーを消灯するプログラムです。

Ctrl_WQ2812_brightness_ALLはRGBバーの色を制御するAPIです。

その他のAPIは http://www.yahboom.net/study/RASPBOT-V2 の3.Car basic courseに記載されています。

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

2. ROS2を使用せず直接APIを呼び出すプログラムを作成

Raspbot V2上にファイルを作成する方法はいくつかあります。好きな方法を選択してください。

1. Dockerコンテナ上で直接ファイルを作成・編集する。

Dockerコンテナにはnanoやvimなどのテキストエディターがインストールされています。これらを使用してDockerコンテナ上で直接ファイルを編集可能です。

```
root@yahboom:~/temp# nano ファイルパス
または
root@yahboom:~/temp# vim ファイルパス
```

2. Dockerホスト上でコンテナの/root/tempへマッピングしているファイルを作成・編集する。

Dockerコンテナの/root/tempはDockerホスト上の/home/pi/tempにマッピングされており、このファイルを編集することでDockerコンテナからもアクセス可能です。

```
pi@yahboom:~ $ nano /home/pi/temp/ファイルパス
または
pi@yahboom:~ $ vim /home/pi/temp/ファイルパス
```

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作
    2. ROS2を使用せず直接APIを呼び出すプログラムを作成
        3. Jupyter Labを使用してブラウザーからファイルを作成・編集する。

Raspbot V2にはJupyter Labという統合開発環境がインストールされています。これを使用してDockerホスト上のファイルを編集することが可能です。

・「http://IPアドレス:8888/」にアクセスする。パスワードはSSH接続と同じものです。

・最初に表示されている場所は/home/piなので、そこからtempディレクトリーに入ることで/home/pi/temp配下のファイルを編集可能です。

4. PCで編集したファイルを転送する。

SSHはファイルの転送が可能です。これによりPC上でファイルを編集し、それをロボットカーに転送することでプログラムを配置することが可能です。

```
C:¥Users¥yajima>scp Windows上のファイルパス pi@192.168.1.11:/home/pi/temp/Dockerホスト上のファイルパス
pi@192.168.1.11's password:
```

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

2. ROS2を使用せず直接APIを呼び出すプログラムを作成

Dockerコンテナ上でファイルを作成した場合、rootユーザー(特権ユーザー)権限でファイルが作成されます。一方、Dockerホストにはpiユーザー(一般ユーザー)権限でログインしています。このため、Dockerホスト上でDockerコンテナ上作成したファイルを編集しようとすると権限がないため失敗します。

これを解消する場合は、Dockerコンテナ上でファイルの所有者またはパーミッションを変更してください。

```
root@yahboom:~/temp# chown 1000:1000 ファイルパス
または
root@yahboom:~/temp# chmod o+w ファイルパス
```

ロボットカーの開発・動作環境のセットアップ

3. ROS2を使用しない構成でロボットカーを操作

3. 作成したプログラムを実行する

Dockerコンテナ上でプログラムを実行します。

```
root@yahboom:~/temp# python3 ファイルパス
```

RGBバーが白色に点灯し、その後消灯すれば成功です。

ロボットカーの開発・動作環境のセットアップ

4. ROS2を使用する構成でロボットカーを操作

ここの部分を作る。
本講座内ではこのPGは
ロボットカー内で動作さ
せる(余裕がある人は
PCで動かしてもよい)

サーバーPG
ROS2

ロボットA用PG
ROS2
ロボットA用ドライバー/API

ロボットB用PG
ROS2
API呼出
ロボットB用ドライバー/API

ロボットC用PG
ROS2
ロボットC用ドライバー/API

ロボットA

ここの部分は
Raspbot V2が提
供するものを使う

ロボットB

ロボットC

ロボットカーの開発・動作環境のセットアップ

4. ROS2を使用する構成でロボットカーを操作

1. ROS2ワークスペース、パッケージの作成

ROS2を使用するために、新しいプログラムを作成する際には最初にDockerコンテナ上でワークスペースおよびパッケージの作成を行う必要があります。ここではワークスペースをsample_work_space、パッケージ名をpackage_sample_py、ノード名をsampleとしてパッケージ作成の例を示します。

```
root@yahboom:/# cd /root/temp
root@yahboom:~/temp# mkdir -p sample_work_space/src
root@yahboom:~/temp# cd sample_work_space
root@yahboom:~/temp/sample_work_space# colcon build
root@yahboom:~/temp/sample_work_space# cd src
root@yahboom:~/temp/sample_work_space/src# ros2 pkg create package_sample_py --build-type ament_python
  --dependencies rclpy --node-name sample
```

4. ROS2を使用する構成でロボットカーを操作

2. プログラムの作成

パッケージの作成が成功していれば、
/root/temp/sample_sork_space/src/package_sample_py/ package _sample_py/sample.py というファイルができています。このファイルを編集してプログラムを作成します。

ROS2のプログラムについては後述します。ここでは https://github.com/TakashiYajima/2025kit/blob/main/src/light.py を記述したと仮定して進めます。

ロボットカーの開発・動作環境のセットアップ

4. ROS2を使用する構成でロボットカーを操作

3. ビルド

ROS2を使用する場合、ソースコードを修正しただけでは実行することができず、ビルドを行う必要があります。

```
root@yahboom:/# cd /root/temp/sample_work_space
root@yahboom:~/temp/sample_work_space# colcon build
```

4. ロボットカー側プログラムを起動する

Raspbot V2が提供するロボットカーのAPIを実行するプログラムを実行します。自身が作成したプログラムも動作させたいため、コマンドプロンプトをもう一つ起動します。ROS2を動作させるDockerコンテナは起動済みであるため、再度起動はせずそのコンテナに接続します。

```
C:¥Users¥yajima>ssh pi@192.168.1.11
pi@yahboom:~ $ docker ps
CONTAINER ID   IMAGE                                    ...
59b5186dde8f   yahboomtechnology/ros-humble:0.1.0   ...
pi@yahboom:~ $ docker exec -it 59b5186dde8f /bin/bash # 上記コンテナIDを指定して起動済みコンテナに接続する
root@yahboom:/# ros2 launch yahboomcar_bringup bringup.launch.py
```

ロボットカーの開発・動作環境のセットアップ

4. ROS2を使用する構成でロボットカーを操作

5. 実行

作成したプログラムを実行します。

```
root@yahboom:~/temp/sample_work_space# source install/setup/bash
root@yahboom:~/temp/sample_work_space# ros2 run package_sample_py sample
※終了するにはCtrl+Cを押します。
```

1秒間隔でRGBバーが点滅したら成功です。

ROS2のプログラム

本講座のROS2サンプルプログラムは下記の構成をしています。

```
クラスのインポート

ノードクラスの定義

ROS2の初期化および実行処理
```

1. クラスのインポート

   サンプルプログラムで使用するクラス類をインポートしています。このコードをベースにする場合必要に応じて追加でインポートする必要があるかもしれません。

ROS2のプログラム

2. ROS2の初期化および実行処理

下記を行っています。

1. Python用ROS2の初期化
2. ノードクラスのインスタンス作成
3. ROS2の実行
4. ノードの破壊
5. Python用ROS2の終了

このコードをベースにする場合、この箇所はおそらく修正の必要はありません。

ROS2のプログラム

2. ROS2の初期化および実行処理

下記を行っています。

1. Python用ROS2の初期化
2. ノードクラスのインスタンス作成
3. ROS2の実行
4. ノードの破壊
5. Python用ROS2の終了

このコードをベースにする場合、この箇所はおそらく修正の必要はありません。

ROS2のプログラム

3. ノードクラスの定義

ロボットカーに指示を出すプログラムの場合、下記を行っています。

1. Publisherの作成
2. タイマーイベントの登録
3. 状態の初期化

ロボットカーのセンサーを読むプログラムの場合、下記を行っています。

1. Subscriberの作成
2. センサー読み込み時の処理定義

サンプルプログラムではそれぞれPublisherまたはSubscriberを1つずつしか作成していませんが、複数のPublisherおよびSubscriberを作成することが可能です。

ROS2のプログラム

3. ノードクラスの定義

1. Publisherの作成

Publisherはロボットカーに対して指示を出すことができます。

create_publisher(型, トピック名, QoS)で作成可能です。利用可能な型およびトピック名はhttp://www.yahboom.net/study/RASPBOT-V2 の9.1 Robot information releaseに記載されています。

2. タイマーの作成

create_timer(間隔, コールバック関数)で作成可能です。指定した間隔毎にコールバック関数が呼び出されます。

3. Subscriberの作成

Subscriberはロボットカーのセンサー情報を読み取ることができます。

create_subscription(型, トピック名, コールバック関数, QoS)で作成可能です。利用可能な型およびトピック名はPublisherを参照してください。センサーから値を読み取れたときにコールバック関数が呼び出されます。読み取った値は第2引数に格納されています。

https://xack.co.jp