

SI2024\_RT ミドルウェアコンテスト

複数台ロボットのリアルタイム状態監視を可能にする

柔軟なモニタリングサービスの提案

大竹 徹 （名城大学）

大原 賢一（名城大学）

# 目次

## 1. 概要

### 1.1. 製作システムについて

### 1.2. テーマ選定における背景

## 2. 構成

### 2.1. システム構成図

### 2.2. SysML による要求図,

### 2.3. SysML ブロック定義図

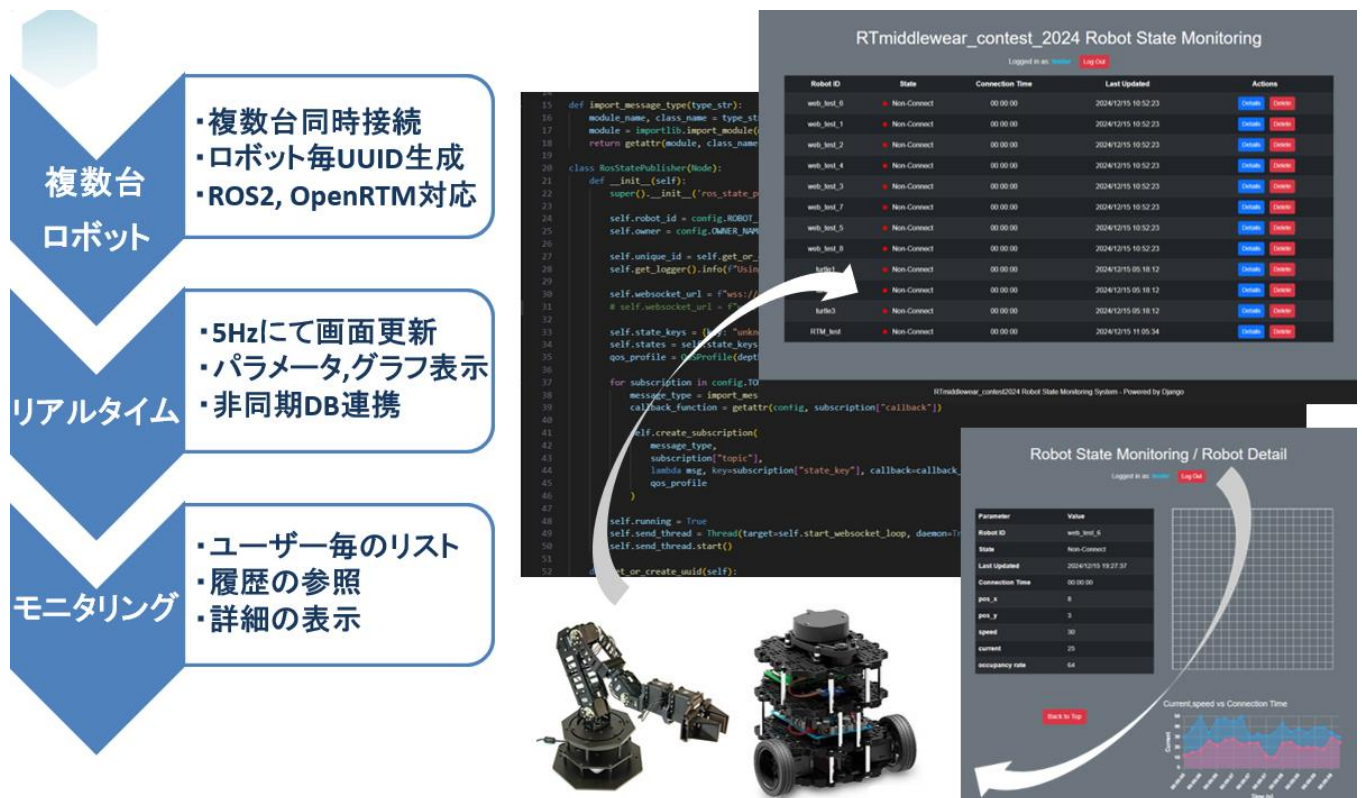
## 3. 開発環境

## 4. 利用手順

## 5. まとめ

# 1. 概要

## 1.1 製作システムについて



本プロジェクトにて

ロボットの状態監視リアルタイムモニタリングサービスを作成した。

本サービスは、稼働中のロボットの状態を取得し、websocket 通信にて送信する、情報送信モジュールと、エンドポイント先の WEB サービスによって構成されている。本サービスにて実現したことについて以下に示す。

- 複数台の同時接続
- ROS2, OpenRTM への対応
- 5Hz での画面動的更新
- ロボットオーナー毎のリストの表示
- 詳細ページでのデータの参照
- 送信データを利用したデモ

## 1.2 テーマ選定における背景

近年、ロボット技術は様々な分野で普及している。  
複数台のロボットが稼働する環境下において、タスク管理、連携や保全等にてロボットの状態をリアルタイムで監視する技術が不可欠となっている。  
しかしながら、既存のツールは、カスタマイズ性や、複数台のロボットを一元的に管理するための機能にまだ課題がある。

そこで、本プロジェクトにおいて、導入、設定の複雑さやロボットミドルウェアの互換等の問題に取り組み、より簡便で、気軽に使用できるツールを製作した。

## 2. 構成

### 2.1. システム構成図

本プロジェクトにおけるシステム構成図を以下に示す。

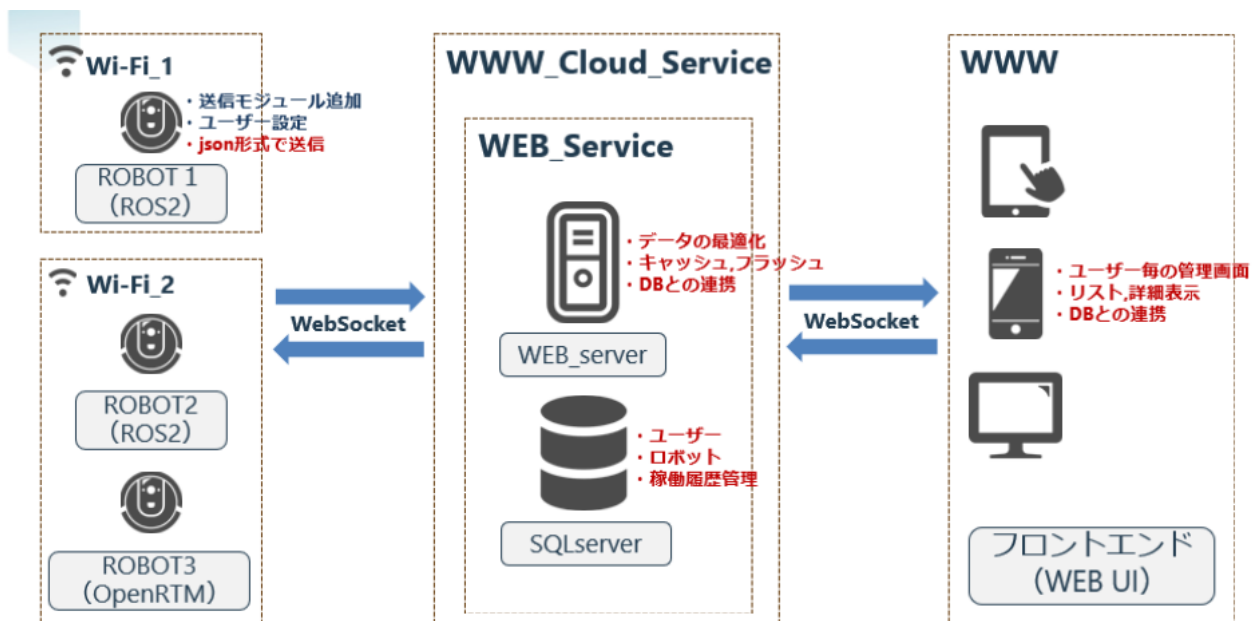


図 1.システム構成図

本システムにおいての特徴としては、

- ・ 導入,設定はロボット側にて最小限の操作
  - ・ ミドルウェア問わず管理
  - ・ フロントエンドのデバイス問わず表示
- が挙げられる。

## 2.2. SysML 要求図

システム構成における要求図を下に示す.

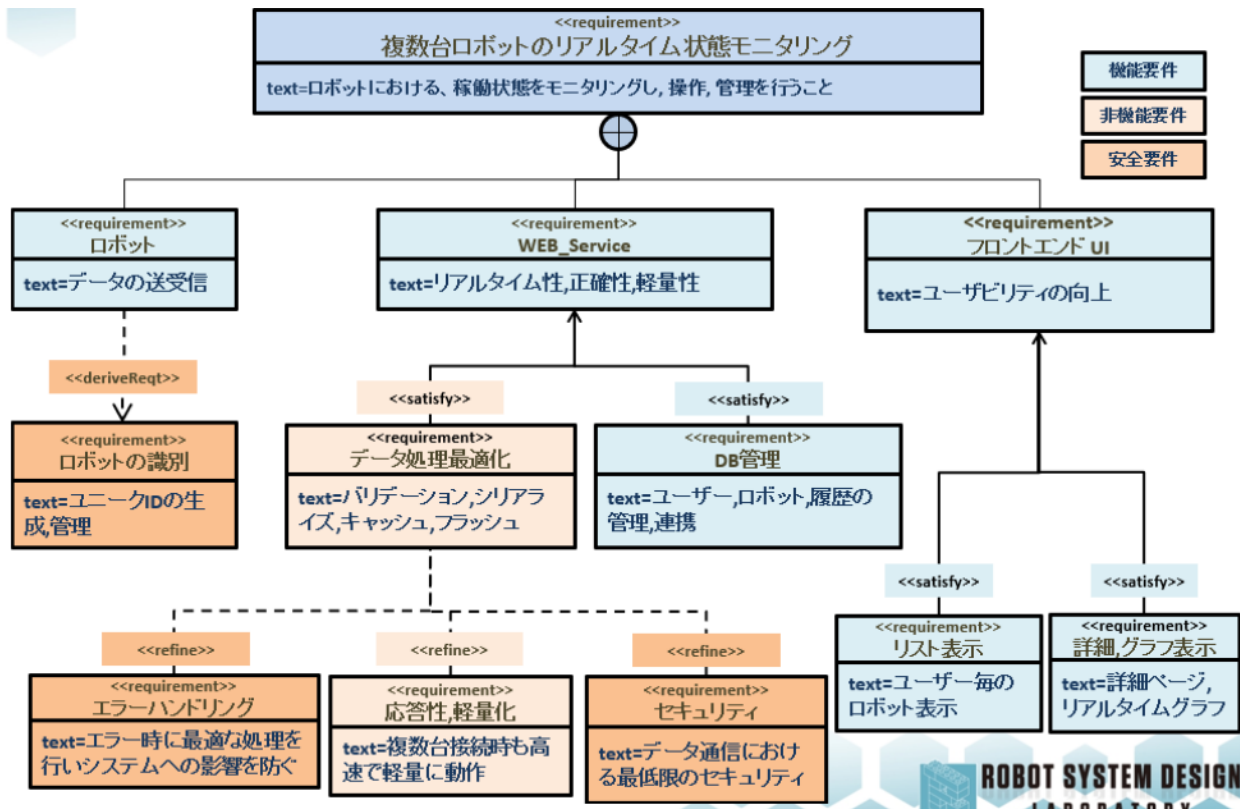


図 2.SysML 要求図

モニタリングシステムとしての最低限の機能を保証しつつ,WEB サービスとして軽量で,非同期における高速な画面描写更新,セキュリティについての要求も必要である.

### 2.3. SysML ブロック定義図

要求図の機能を実現するため具体的なデータの流を、ブロック定義図として以下に示す。

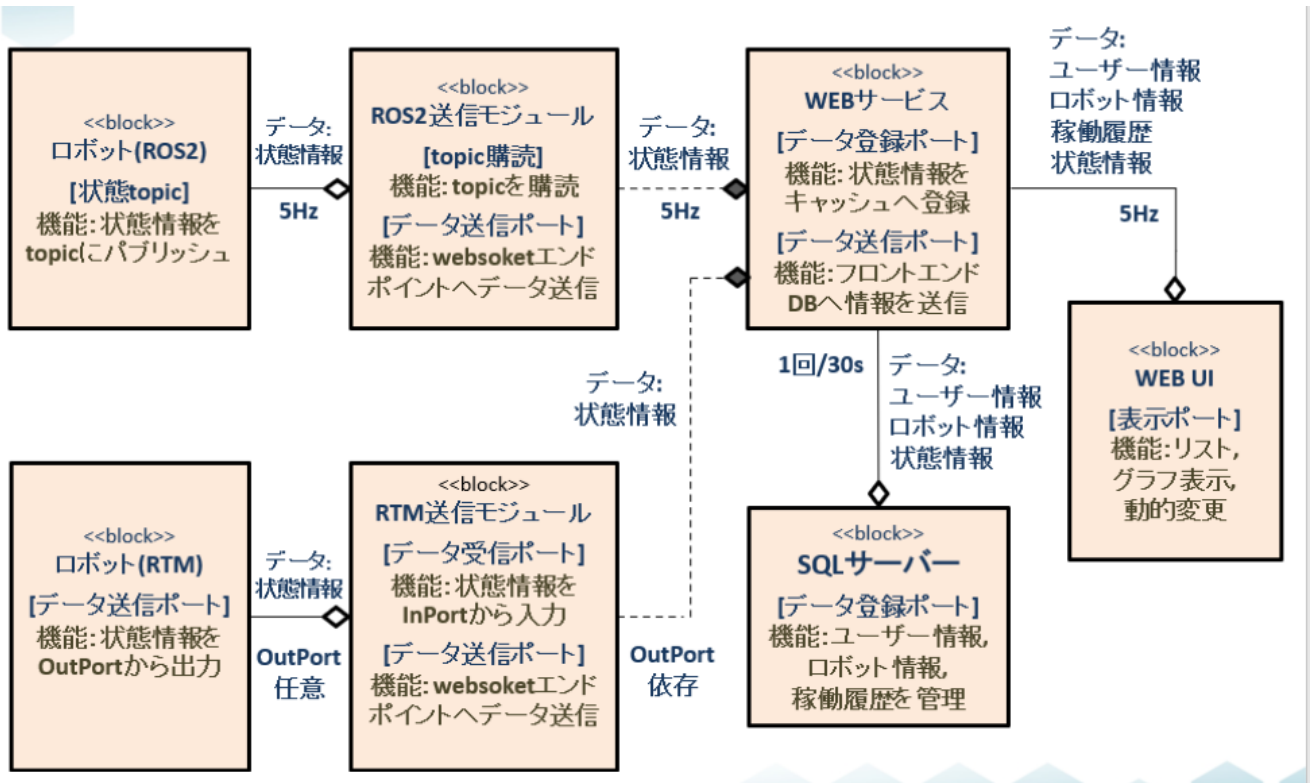


図 3. SysML 図

本プロジェクトはこのブロック定義図を元に製作した。

### 3. 開発環境

本プロジェクトにおける開発環境を以下に示す.

OS : ubuntu22.04

Middlewear : ROS2\_ Humble, OpenRTM-aist-Python2.0.2

言語 : Python3.10

WEB サービス製作フレームワーク: Django 5.1.3

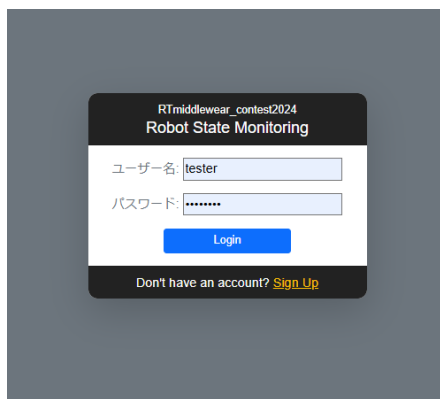
SQL サーバー: PostgreSQL

キャッシュサーバー : redis

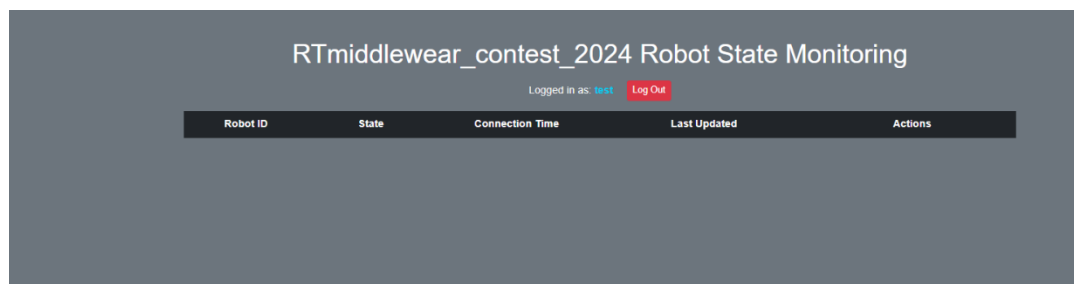
Cloud サービス: AWS (VPC, EC2, RDS)

### 4. 利用手順

- ① 製作 WEB サービス URL : <https://monitoring.ddns.net/>へアクセス
- ② ログインページにてサインアップをクリック
- ③ サインアップページにてユーザーの登録を行う。



その後,空のリストが表示されたトップページへ遷移する.



Robot ID	State	Connection Time	Last Updated	Actions
----------	-------	-----------------	--------------	---------

#### ④ 状態送信モジュールのダウンロード

本リポジトリソースコードにて状態送信モジュールをダウンロード.

ROS2 :

[https://github.com/otto0521/RTMcontest\\_2024\\_RTMS\\_scripts/tree/main/ro  
s2/scripts](https://github.com/otto0521/RTMcontest_2024_RTMS_scripts/tree/main/ro<br/>s2/scripts)

RTM :

[https://github.com/otto0521/RTMcontest\\_2024\\_RTMS\\_scripts/tree/main/rt  
m/WebSocketDataSender](https://github.com/otto0521/RTMcontest_2024_RTMS_scripts/tree/main/rt<br/>m/WebSocketDataSender)

#### ⑤ コンフィグファイル設定

例 : ROS2 用モジュール

config.py  
ros2\_robot\_state\_publisher.py

config.py を開くと以下の内容が表示される.

ここで、オーナー名 (WEB サービスにて作成したユーザー名)

ロボット ID (任意) , 購読したいトピック, メッセージ型, 関数の設定を行う.

```
1
2 # ロボット設定
3 ROBOT_ID = "default" # ロボットIDを定義してください
4 OWNER_NAME = "default" # オーナー名を定義してください ※WEBサービスのサインアップ名と一致している必要があります
5
6 # 購読トピック設定
7 TOPIC_SUBSCRIPTIONS = [
8     {
9         "topic": "/turtle1/pose",
10        "message_type": "turtlesim.msg.Pose",
11        "callback": "update",
12        "state_key": "pos_x"
13    },
14
15    # {
16    #     "topic": "トピック名",
17    #     "message_type": "メッセージ型",
18    #     "callback": "コールバック関数",
19    #     "state_key": "状態キー"
20    # },
21 ]
22
23 # ユーザー定義のコールバック関数
24 def update(states, msg):
25     states["pos_x"] = msg.x
26     states["pos_y"] = msg.y
27     states["theta"] = msg.theta
28
29 # def update_state2(states, msg):
30
31 #     states["pos_y"] = msg.y
32
```

例として turtlesim の座標, 姿勢を設定



## ⑥ ロボットの起動, スクリプトの起動

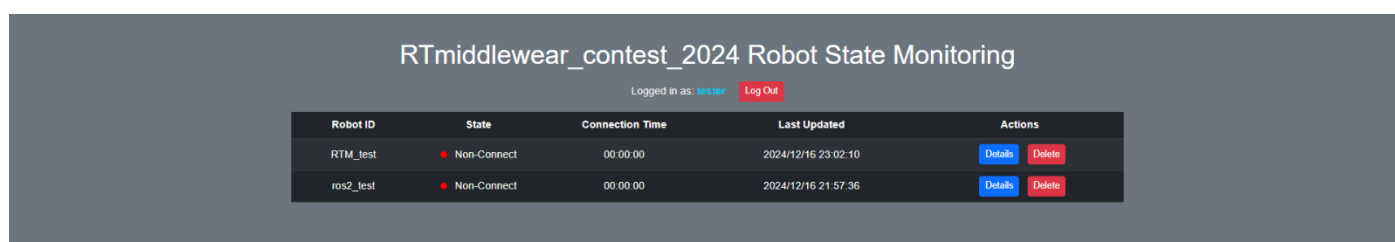
対象のロボットを起動し,topic が参照できる環境下にて  
ros2\_robot\_state\_publisher.py を起動.

コマンド : python3 ros2\_robot\_state\_publisher.py

(lunch ファイルから起動しても OK)

RTM の場合,ワークスペース内にて config.py 設定後, WebSocketSender.py を  
起動すれば, エディターにてコンポーネントとして利用することができる.

## ⑦ Top ページリロード後にてリストに追加される.



RTmiddlewear\_contest\_2024 Robot State Monitoring

Logged in as: [tester](#) [Log Out](#)

Robot ID	State	Connection Time	Last Updated	Actions
RTM_test	Non-Connect	00:00:00	2024/12/16 23:02:10	<a href="#">Details</a> <a href="#">Delete</a>
ros2_test	Non-Connect	00:00:00	2024/12/16 21:57:36	<a href="#">Details</a> <a href="#">Delete</a>

## ⑧ 詳細ページ表示

Details ボタンを押すと, ロボット個別のページに遷移し,送信した topic 情報  
やインポートしたデータが受け取られていることが確認できる.

## 5. まとめ

本プロジェクトでは複数台のロボット状態をリアルタイムにてモニタリングすることができる WEB サービスを製作した.

サービスサーバーに Json 形式にてデータを送信することによって,ミドルウェアに関わらず状態情報の取得,管理をすることが出来るようになった.

また,websocket 通信による高頻度の通信,コンシューマーでのキャッシュ処理,適切なデータベースでのアクセスを考慮し,軽量に,高速に動作するシステムを構築した.

しかしながら,本プロジェクトではデータの取得止まりであり,サービスとしてまだ価値のないものである.今後は機能の拡張に専念し,便利なツールを目指して製作を継続していく.