RT_System_Integration_Framework
操作マニュアル

名城大学メカトロニクス工学科

ロボットシステムデザイン研究室

2023 年 12 月 14 日

目次

目次

1.　はじめに

1.1. 目的

本書の目的は RT System Integration Framewor を用いたシステムの実行および操作方法を解説することである，RTM と ROS を使った移動ロボットのサンプルのシステムを 2 種類用意し，それらを用いて説明を行う．

1.2. 本書を読むにあたって

本書は RT ミドルウェアに関する基礎知識を有した利用者を対象としている．また，各 RTC の詳細な仕様等については同ファイル内の仕様解説マニュアルを参考にすること．

1.3. RT System Integration Framework について

RT System Integration Framework は異なるミドルウェア間でのソフトウェアリソースの相互運用を可能にする仕組みを導入したソフトウェアフレームワークである．本書では，図 1 のように ROS と RTM を使ったシステムを対象としている．RTC や RT システムの管理・運用フレームワークである wasanbon を基盤とし，RTM と ROS の相互運用が可能な形に機能拡張を行った．本フレームワークではシステム構築の工程を必要なモジュールの収集，システム構築，システム運用に分け，ミドルウェアの違いを意識させない運用を実現している．

システム構成としては，

・Systemoperate.py…システムに必要なモジュールの収集，システム構築，システムの起動を行う．

・systemconfig.yml…必要なモジュールやコマンドを記載. Systemoperate.py を起動したら読み込まれる．

・GUI.py…Systemoperate.py の起動するための GUI ファイル．

・Dockerfile…Ubuntu20.04，ROS noetic，RTM2.0，wasanbon(RTM2.0 対応)がインストールされる．

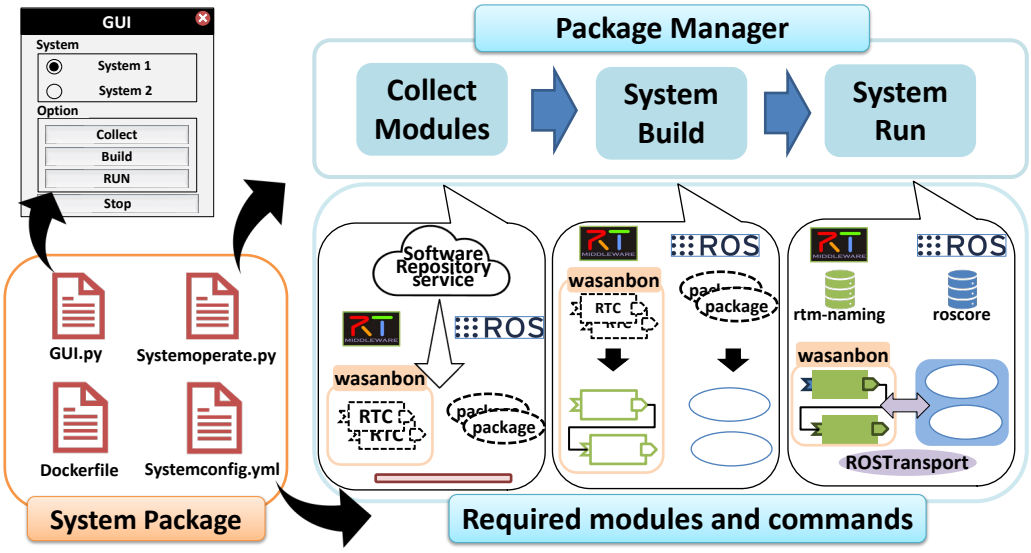


図 1 RT System Integration Framework の概要図

1.4. wasanbon について

wasanbon は株式会社 SUGAR SWEET ROBOTICS らが開発した RT システムを管理・運用を行うフレームワークである．GitHub からの RTC の収集から RT システムの実行までの工程を CUI で簡便に操作できる．RT System Integration Framework はこの wasanbon をベースとしている．以下に参考ページを掲載する．

wasanbon の紹介：http://wasanbon.org/

本書で使用する wasanbon の GitHub ページ: https://github.com/rsdlab/wasanbon

本書で使用する wasanbon のインストール方法: OpenRTM-aist-2.0.0 への Wasanbon のインストール

1.5. 動作環境

Docker 環境を利用しているため，インストール先は問わないが，以下の動作環境で操作した場合を示している．

OS	Ubuntu20.04
RT ミドルウェア	OpenRTM-aist-2.0.1
ROS	ROS noetic
Docker	version 24.0.5

1.6. 開発環境

OS	Ubuntu20.04
RT ミドルウェア	OpenRTM-aist-2.0.1
言語	C++,Python
Docker	version 24.0.5

2.　操作の準備

2.1 環境構築

Docker の環境構築を行う．

```
$ sudo apt install docker.io
```

本パッケージを GitHub からダウンロードする

```
$ git clone https://github.com/KatoMisa/RT_System_Integration_Framework
```

本パッケージに移動する

```
$ cd RT_System_Integration_Framework
```

Dockerfile をビルドする(イメージ名は問わない)

```
$ docker build -t rtsif:1
```

Docker コンテナを起動．GUI のアプリを起動させるためにオプションを多数追加している．

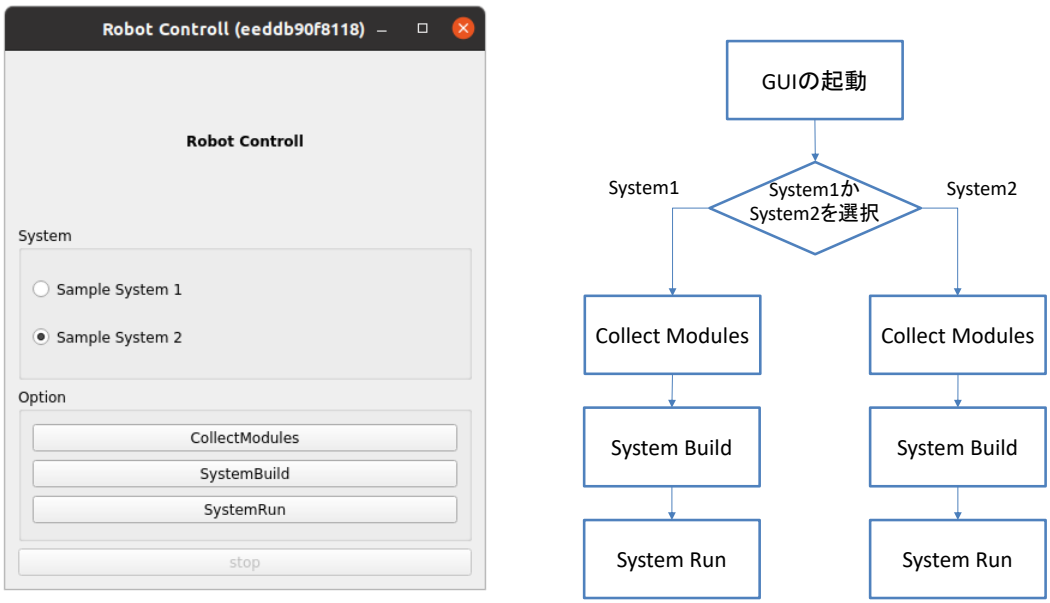
```
$ docker run -it --name [ コンテナ名 ]  --privileged    --env="DISPLAY"  ¥
    --volume="/etc/group:/etc/group:ro"    ¥
    --volume="/etc/passwd:/etc/passwd:ro"      ¥
   --volume="/etc/shadow:/etc/shadow:ro"      ¥
   --volume="/etc/sudoers.d:/etc/sudoers.d:ro"    ¥
    --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" [image 名:タグ名]
```

2.2.Docker コンテナの内容

・osrf が提供している Ubuntu20.04 の Docker イメージを使用している
・ユーザは sudo 権限を持ち，名前は rsdlab である
・ROS noetic，RTM2.0.1 および wasanbon といったミドルウェアがインストールされている
・emacs や git などの必要なライブラリはインストールされている

2.3.システム操作用の GUI について

システムに必要なモジュールの収集，システム構築，システムの起動については，図(左)のような GUI を使用する．GUI の使い方としては，Python ファイルを起動させ，[System]で選択したシステムに関するシステムの操作を行う[Option]のコマンドを選択することで，システムに必要なモジュールの収集からシステムの起動までの工程を行っていく．図(右)にフロー図を示す．



概要図　　フロー図

図 2 システム操作用の GUI について

3.　システム適用事例 1

3.1. サンプルシステム概要

サンプルシステム 1 は，RTM 側でジョイスティックの操作を行い，ROS 上で動作する THK(株)の SEED-Mover を操縦するシステムである．wasanbon のパッケージはジョイスティックを操作するためのシステムである MobileRobotControl(https://github.com/rsdlab/MobileRobotControl.git)を使用する．表にシステム 1 におけるオプションで行われるコマンドを示す．

オプション	実施事項
Collect Modules	・MobileRobotControl を動かすための SFML ライブラリのダウンロード ・MobileRobotControl パッケージ及びその RTC の収集 ・SEED-Mover を動かすための ROS パッケージの収集
System Build	・ROS のビルド ・RTC のビルド及びシステム構築
System Run	・ROS 及び RTM のネームサーバーの起動 ・RTM のシステムを起動 ・seed_r7_bringup パッケージの seed_r7_bringup.launch を起動 （SEED-Mover を起動するための Launch ファイル）

3.2.操作方法

3.1.1.システム操作用の GUI アプリの起動

Docker コンテナに入ったら，catkin make の性質上 catkin_ws 上で操作を行うため，移動する．

```
$ cd　catkin_ws
```

GUI を起動

```
$ python3 .~/system/UI.py
```

3.1.2. [Collect Modules]を押す

エラーが起きずに，最後の workspace 上に SFML2-4-2 ディレクトリが作られるところまでできていたら次へ進む．

```
# ls ~/workspace
MobileRobotControl　SFML2.4.2　SFML2.4.2-linux-gcc-64-bit..tar.gz
```

3.1.3. [System Build]を押す

RTC のビルドが 2 つとも success, 及び TestSerializer.so が移動したことが表示されていたら次へ進む．

3.1.4.ロボットと接続する

ジョイスティックのレシーバーを PC に挿しておく．PC に SEED-Mover の USB を挿し，以下のコマンドを打って“usb-~”が表示され，SEED-Mover と接続されているかの確認する．

```
#　ls /dev/serial/by-id/
```

権限を与える．

```
# sudo chmod 666 /dev/serial/by-id/usb～
```

Emacs を用いて seed_r7_bringup.launch の[WAS]の該当部分を[IS]になるように編集して保存する

```
# cd catkin_ws/src/seed_r7_ros_pkg/seed_r7_bringup/launch
# emacs seed_r7_bringup.launch
[WAS] <param name="port_lower" value="/dev/aero_lower"/>
[IS]<param name="port_lower" value="/dev/serial/by-id/usb～>
```

3.1.5. [System Run]を押す

RTM のネームサーバーが立ち上がるときにパスワードを入力する必要があるので，入力する．

全てのシステムが立ち上がったらジョイスティックを操作すれば SEED-Mover が動作する．

4. システム適用事例 1

4.1. サンプルシステム概要

サンプルシステム 2 は，RTM 側で指定した目的地へ，ROS 上で動作する THK(株)の SEED-Mover のナビゲーションをするシステムである．地図データ及び 3 つのウェイポイントを事前に用意しており，目的地を指定すると，対応した地点まで移動する．wasanbon のパッケージは目的地を ROS ノードに送信するためのシステムである Destinaion_gui(https://github.com/KatoMisa/Destinaion_gui.git)を使用する．表にシステム 2 におけるオプションで行われるコマンドを示す．

オプション	実施事項
Collect Modules	・Destinaion_gui 及びその RTC の収集 ・SEED-Mover を動かすための ROS パッケージの収集 ・ナビゲーションを行うための ROS パッケージのダウンロード ・必要な地図データの移行
System Build	・ROS のビルド ・RTC のビルド及びシステム構築
System Run	・ROS 及び RTM のネームサーバーの起動 ・RTM のシステムを起動 ・seed_r7_bringup パッケージの seed_r7_bringup.launch を起動 (SEED-Mover を起動するための Launch ファイル) ・seed_r7_navigation wheel_with_static_map.launch (地図データの反映及び SEED-Mover を動かすための Launch ファイル) ・rviz (ロボットなどの情報を可視化するツール) ・Movetest.py の起動 (ナビゲーションを行うためのスクリプト)

4.2.操作方法

4.2.1 システム操作用の GUI アプリの起動

Docker コンテナに入ったら，catkin make の性質上 catkin_ws 上で操作を行うため，移動する．

```
$ cd　catkin_ws
```

GUI を起動

```
$ python3 .~/system/UI.py
```

4.2.2 [Collect Modules]を押す

エラーが起きていなかったら次へ進む．

4.2.3 [System Build]を押す

RTC のビルドが 2 つとも success になっていたら次へ進む．

4.2.4 ロボットと接続する(3.1.4 を参照)

4.2.5 [System Run]を押す

RTM のネームサーバーが立ち上がるときにパスワードを入力する必要があるので，入力する．

全てのシステムが立ち上がったら，GUI が立ち上がり目的地を指定したら SEED-Mover がその地点までナビゲーションを行う．図 3 に目的地を選択する GUI を示す．

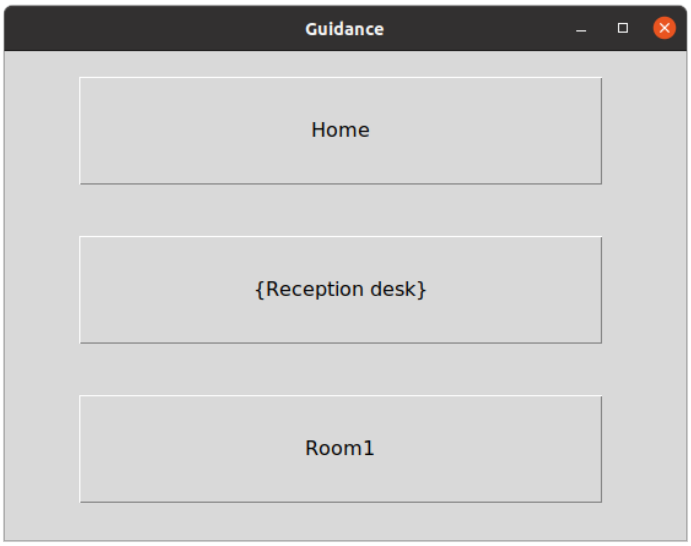


図 3　目的地を選択するための GUI