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

1. はじめに 3

1.1. 目的 3
1.2. 本書を読むにあたって 3
1.3. RT System Integration Framework について 3
1.4. wasanbon について 4
1.5. 動作環境 4
1.6. 開発環境 4

2. 操作の準備 4

2.1. Ubuntu20.04 の環境がある場合 5
2.1.1. 環境構築 5
2.2. Ubuntu20.04 の環境がない場合 5
2.2.1. 環境構築 5
2.2.2. Docker コンテナの内容 6
2.3. システム操作用の GUI について 6

3. システム適用事例 1 7

3.1. サンプルシステム概要 7
3.2. Ubuntu20.04 の環境がある場合 7
3.3. Ubuntu20.04 の環境がない場合 8

4. システム適用事例 1 9

4.1. サンプルシステム概要 9
4.2. Ubuntu20.04 の環境がある場合 9
4.3. Ubuntu20.04 の環境がない場合 11

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 RTM を ROS のシステムをモジュールの収集から一括起動まで行うことを可能にする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 のネイティブな環境がない場合に使用する．Ubuntu20.04，ROS noetic，RTM2.0，wasanbon(RTM2.0 対応)がインストールされる．

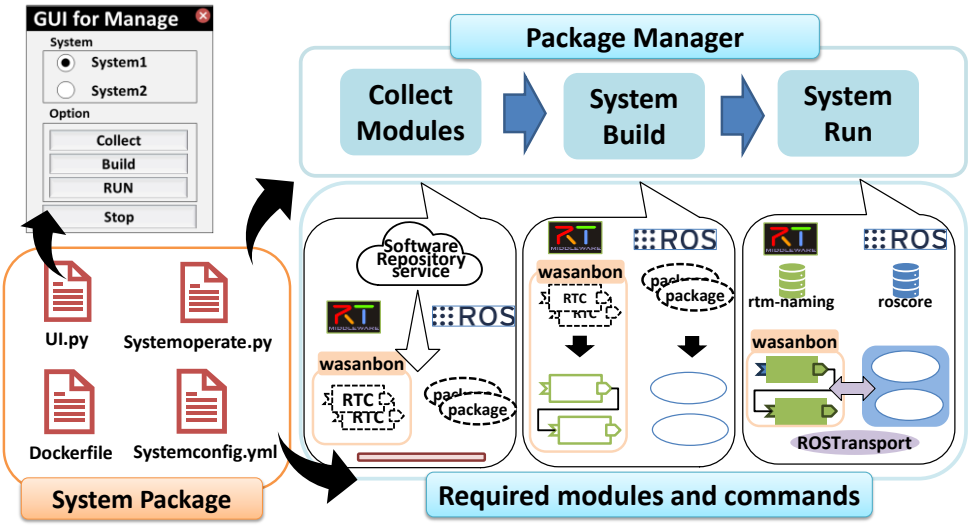


図 1 RT System Integration Framework の概要図

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

wasanbon のインストール方法: https://wwwms.meijo-u.ac.jp/kohara/openrtm-aist-200_wasanbon

RT System Integration Framework は，図 2 のように wasanbon による RT システムの操作及び ROS パッケージの操作を行う大きなフレームワークである．

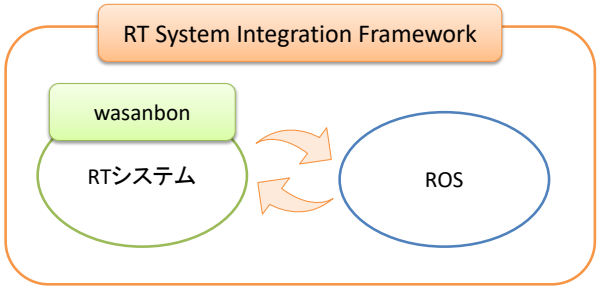


図 2　RT System Integration フレームワークの操作範囲の簡略図

1.5. 動作環境

以下に推奨する動作環境を示す．付属の Docker 環境を利用すれば，インストール先は問わない．

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
ROS	ROS noetic
言語	Python
Docker	version 24.0.5

2.　操作の準備

以下では，Ubuntu20.04 の環境がある場合に直接システムを導入する方法と Ubuntu20.04 の環境がない

場合に Docker を使用してシステムを導入する方法を説明する．ここでは，ワークスペースを下記に作成することとしている．Docker コンテナのユーザ名は rsdlab としている．

```
$ /home/<ユーザ名>
```

2.1. Ubuntu20.04 の環境がある場合

2.1.1. 環境構築

本パッケージを GitHub からダウンロードする．（ここでは github ディレクトリ上に git clone する）

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

本パッケージ内の system ディレクトリをホーム上に移す

```
$ cd ~/RT_System_Integration_Framework
$ cp -rp system/ ~/
```

コピーした system ディレクトリに移動する．

```
$ cd system
```

2.2. Ubuntu20.04 の環境がない場合

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

```

```

2.2.1. 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
$ cd RT_System_Integration_Framework
```

コード編集アプリを用いて 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 system
$ gedit Systemoprerate.py
[WAS] #subprocess.run(“catkin_make”)
            subprocess.call([ “catkin”, “build” ])
 [IS]   subprocess.run(“catkin_make”)
       #subprocess.call([ “catkin”, “build” ])
```

同封されている Docker file をコンパイルする(イメージ名は問わない)

```
$ docker build -t [image 名:タグ名]
```

Docker コンテナを起動(コンテナ名は問わない)

```
$ 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]
```

※コンテナから出る方法(コンテナ内では#で示す)

```
# exit
```

※起動したコンテナに再び入る方法

```
$ docker start -i [コンテナ名]
```

2.2.2.　Docker コンテナの内容

付属の Dockerfile をビルド，起動すると，Ubuntu20.04 の環境に ROS noetic，RTM2.0.1 及び wasanbon といったプラットフォームがインストールされた環境を使用可能である．

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 3(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)に概略的なフロー図を示す．

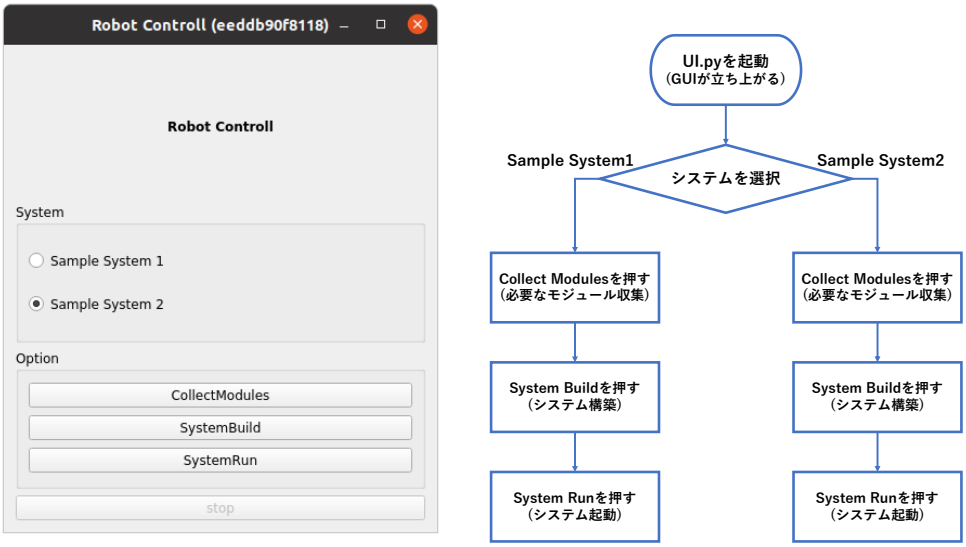


図 3　操作用の GUI

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 ・ROS と RTM の通信を行うためのファイルが移動
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. Ubuntu20.04 の環境がある場合

3.2.1.　システム操作用の GUI アプリの起動

ホーム上の system ディレクトリに移動する．

```
$ cd catkin_ws
```

GUI を起動して System1 を選択(デフォルトは Sample System1)

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

3.2.2. [Collect Modules]を押す

エラーが起きずに， workspace 上に SFML2-4-2 ディレクトリが作られ，SFML と表示されたら次へ進む.

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

3.2.3. [System Build]を押す

3 つの.so ファイルが表示されていたら次へ進む.

```
SFMLJoyStick2.so SFMLJoyStickToVelocity2.so TestSerializer.so
```

3.2.4. 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～
```

コード編集アプリを用いて 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
$ gedit seed_r7_bringup.launch
[WAS] <param name="port_lower" value="/dev/aero_lower"/>
 [IS]  <param name="port_lower" value="/dev/serial/by-id/usb～>
```

3.2.5. [System Run]を押す

Roscore が起動したタブが新しく表示された後，元のターミナルで 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．(～:ユーザ名)

(一度立ち上げていたら(y/N)を聞かれる場合があるため，N と選択)

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.

```
Starting omniORB omniNames: ~ :2809
～のパスワード:
```

3.3. Ubuntu20.04 の環境がない場合

3.3.1. 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 を起動して System1 を選択

```
$ python3　../system/UI.py
```

以下は 3.2.2 以降と同様である．但し，コード編集アプリは emacs のみインストールされている．

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 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	・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） ・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. Ubuntu20.04 の環境がある場合

4.2.1.　システム操作用の GUI アプリの起動

ホーム上の system ディレクトリに移動する．

```
$ cd catkin_ws
```

GUI を起動して System2 を選択(図 3 参照)

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

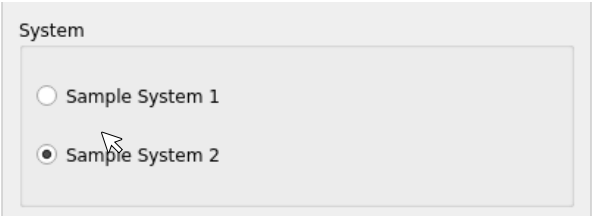


図 4　System の変更方法

4.2.2.　[Collect Modules]を押す

エラーが起きずに，“HEAD is now at e2d40c2~”と表示されていたら次へ進む．

4.2.3.　[System Build]を押す

RTC のビルドが 2 つとも SUCCESS になったら次へ進む．

Build RTC(dest_gui)　SUCCESS Build RTC(relay_gui)　SUCCESS

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

roscore が起動したタブが新しく表示された後，元のターミナルで 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．(~:ユーザ名)
(一度立ち上げていたら(y/N)を聞かれる場合があるため，N と選択)
全てのシステムが立ち上がり図 5 のような GUI が表示される．いずれか選択したら選択した目的地に応じた場所へ SEED-Mover が動作する．

Starting omniORB omniNames: ~ :2809 ~のパスワード:

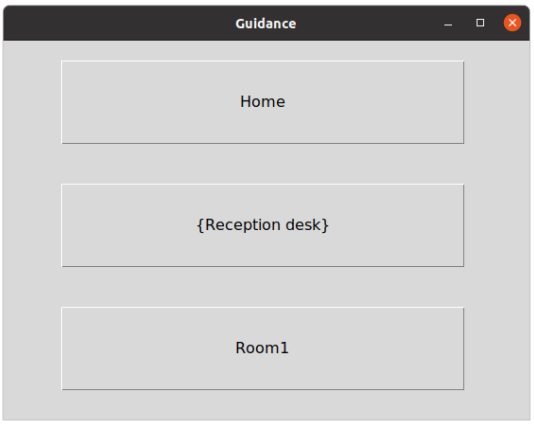
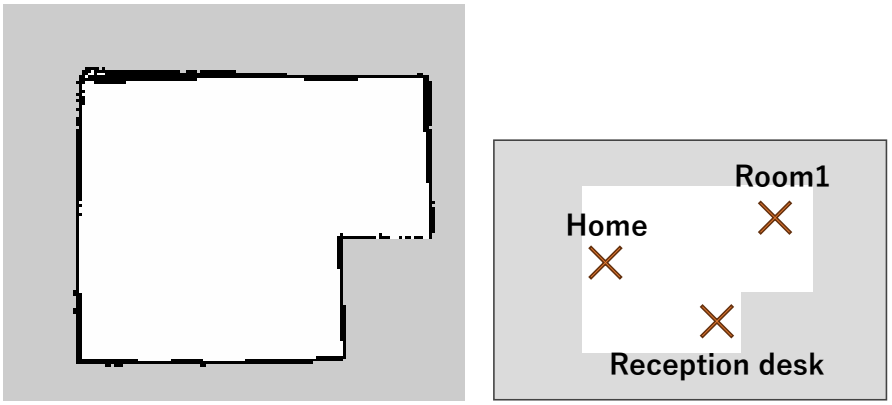


図 5　目的地を選択する GUI

Rviz で確認する場合はターミナルを立ち上げて以下のコマンドを実行する

```
$ rviz rviz -d `rospack find seed_r7_bringup`/rviz.rviz
```

図 6 に今回使用している地図データ(map.pgm)と図 4 の目的地に応じた地図上での場所の概略図を示す.



map.pgm　　目的地

図 6　使用している地図データ及び目的地

4.3. Ubuntu20.04 の環境がない場合

4.3.1.　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 を起動して System1 を選択

```
$ python3　../system/UI.py
```

以下は 4.2.2 以降と同様である．但し，コード編集アプリは emacs のみインストールされている.