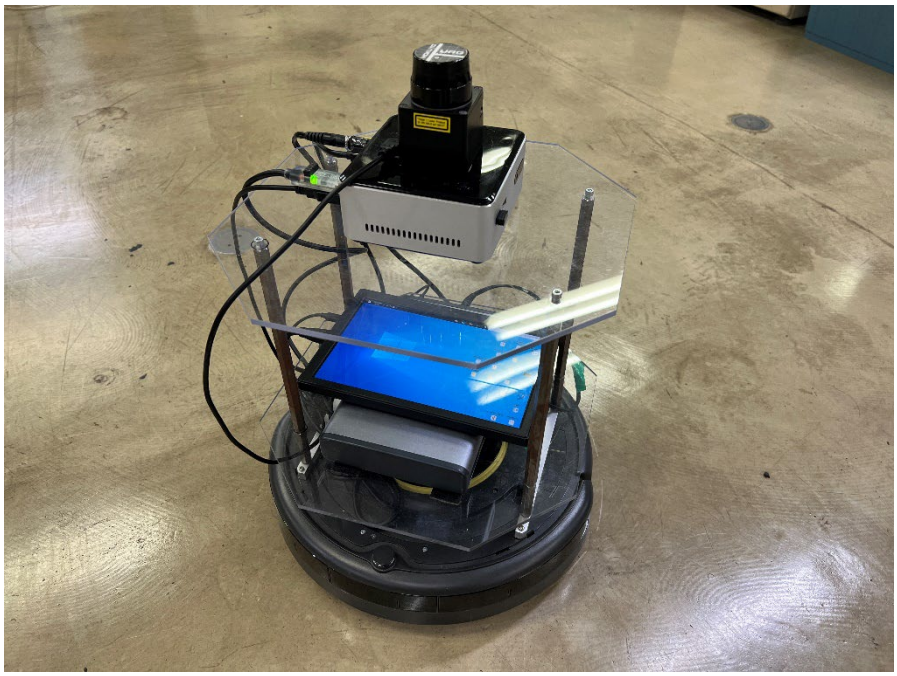
Growing Neural Gas を用いた
未知環境ナビゲーションシステム
のコンポーネントの構築



岡山大学工学部機械システム系学科システム工学コース
適応学習システム制御学研究室

藤井雄基，張家祺，古田優泰，室本達也，戸田雄一郎，松野隆幸

目次

1. はじめに 2
1.1. 目的と概要 2
1.2. 概要 2
2. ハードウェア 3
2.1. 使用するハードウェア 3
2.2. ハードウェアの作製 3
3. ソフトウェア 4
3.1. 開発環境 4
3.2. 再利用した RTC の概要 4
3.3. 開発した RTC の概要 6
4. 利用手順 11
4.1. RTC のダウンロードと共通する準備 11
4.2. Visual Studio の操作 17
4.3. 各種 RTC の起動と Configuration ポートの設定と注意 19
4.4. RTC の Activate 化 28
5. 動作例 31
5.1. 実際の動作 31
6. 参考文献 33

1. はじめに

1.1. 目的と概要

移動ロボットを自律行動させる場合，ロボット自身が自己位置推定および障害物の認識，目的地までの行動計画を行う必要があり，各タスクに応じた作業空間の環境地図を必要な粒度で作成しなければならない．本プロジェクトでは，高精度な自己位置推定のために用いる粒度の細かい占有格子地図から，経路計画に適したグラフ構造により環境を抽象化して表現した地図を作成し，Growing Neural Gas に基づいた地図構築を行う RTC を開発した．本プロジェクトでは，GNG により位置情報と占有度情報を含んだトポロジカルマップを構築し，経路コスト算出を行う RTC 開発を行い，実機を用いた未知環境における検証実験について提案手法の有効性を示す．

1.2. 概要

本プロジェクトでは，北陽電機株式会社が提供している UrgRTC のみ既存の RTC を用いており，他の全ての RTC である，LocalizationRTC，GNG_componentRTC，PathPlanCompRTC，Destination_genRTC，FuzzyControlRTC，Roomba_OutPortRTC の開発を行った．基本的な流れとしては，LocalizationRTC において，占有格子空間地図の生成及び自己位置推定を行った後，GNG_componentRTC においてトポロジカルマップを生成し，PathPlanCompRTC において経路計画を行う流れとなっている．また，経路計画のための目的地の生成は，Destination_genRTC において表示した占有格子空間地図をクリックすることによって指定可能となっている．また，生成された経路に対して，FuzzyControlRTC においてファジィ制御と多目的行動調停を用いて最終的な制御量を決定し，Roomba_OutPortRTC へ入力することで，ロボットを動かす流れとなっている．

2. ハードウェア

2.1. 使用するハードウェア

次の表 1 は，各種使用したハードウェアである．

表 1 使用したハードウェア

ハードウェア名	メーカー	型番
SLAMTEC 360 Degree Laser Range Scanner（以下，LiDAR と表記）	北陽電機株式会社社	URG-04LX-UG01
R NUC キット（以下，PC と表記）	Intel	Model: D54250WYKH
ディスプレイ	-	-
Roomba（以下，ルンバと表記）	iRobot	iRobot Create2

2.2. ハードウェアの作製

前項 2.1. で示した各種ハードウェアをスチール製の枠組みやネジ，アクリル板を用いて図 1 のように固定した．

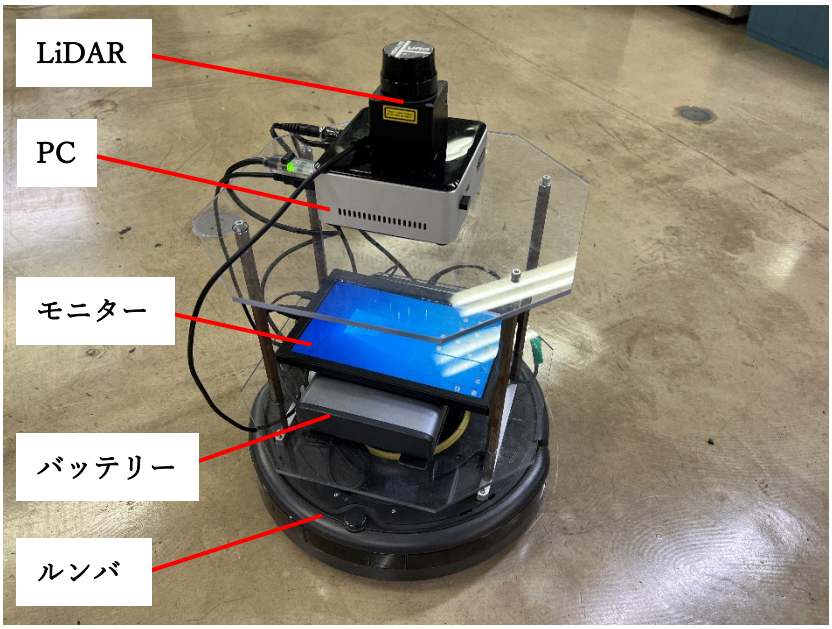


図 1 実際に作成したハードウェア

3. ソフトウェア

3.1. 開発環境

次の表 2 は，本プロジェクトで使用した PC 上に構築した開発環境を示す．
また，本プロジェクトで再利用または開発した RTC はすべて C++ で開発した．

表 2 使用したソフトウェアの情報

種別	ソフトウェア名	バージョン情報
OS	Windows 10 Pro	21H2 19044.2251
RT ミドルウェア	OpenRTM-aist	2.0.0 x86
コンパイラに依存しないビルド自動化のためのフリーソフトウェア	CMake	3.23.2
統合開発環境	Visual Studio Community 2019	16.11.20

3.2. 再利用した RTC の概要

本項では，再利用した RTC の概要について述べる．本プロジェクトで再利用した RTC は UrgRTC のみである．次の表 3 のダウンロードページからダウンロードすることが出来る．この RTC を利用するにあたって，別途 urglib-master をダウンロードし，UrgRTC-master ディレクトリに追加する必要がある．

表 3 再利用した RTC の URL

ページ種別	URL
紹介ページ	https://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID130
ダウンロードページ	https://www.openrtm.org/openrtm/ja/project/Hokuyo_URG_RTC_by_SSR

UrgRTC は，二次元平面上を LiDAR が一回転したとき，LiDAR の正面を 0 度とし，反時計回りの順に距離データを配列へ入れ，この配列を OutPort の range から出力する．Configuration ポートの port_name で LiDAR を接続した PC のポート番号を指定することができる．

次の図 2 および表 4 に UrgRTC の仕様を示す．

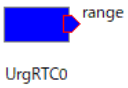


図 2　[System Editor]上に表示される UrgRTC

表 4　UrgRTC の仕様

ポート種別	ポート名	型	デフォルト値	制約	Widget
OutPort	range	RangeData	-	-	-
Configuration	port_name	string	¥¥.¥COM7	-	text
	baudrate	int	115200	-	text
	debug	int	0	-	text
	encoding	spin	2	{twochar:2, threechar:3}	text
	geometry_x	double	0	-	text
	geometry_y	double	0	-	text
	geometry_z	double	0	-	text

GitHub の GNG-Unk-Env-Navi_RTC の main ブランチ上に, UrgRTC と urglib のサブモジュールが追加されており，これを用いて本 RTC 群を動作できる．これらは GNG-Unk-Env-Navi_RTC の main ブランチとは別途 zip ファイルをダウンロードする必要があり，GNG-Unk-Env-Navi_RTC/UrgRTC, GNG-Unk-Env-Navi_RTC/UrgRTC/urglib にある各サブモジュールを[Code]，[Download ZIP]を順にクリックし，ダウンロードする必要がある．

3.3. 開発した RTC の概要

本項では，本プロジェクトで開発した RTC について述べる．本プロジェクトでは，LocalizationRTC，Destination_genRTC，GNG_componentRTC，PathPlanCompRTC，FuzzyControlRTC，Roomba_OutPortRTC の開発を行った．これらの RTC は次の URL 先からダウンロードすることができ，GNG-Unk-Env-Navi_RTC-main.zip ファイルに格納されている．

URL: https://www.openrtm.org/openrtm/ja/project/growing-neural-gas-and-unknown-environment-navigation

① LocalizationRTC

LocalizationRTC は，UrgRTC から受け取った距離データをもとに占有格子空間地図の生成及び自己位置推定を行う．

InPort の range は，LiDAR が一回転する間に取得した距離データの配列を InPort から受け取っている．

OutPort の omap は，この LocalizationRTC で生成した占有格子空間地図を二次元配列から一次元配列に変換して出力する．robot_position は，この LocalizationRTC で推定した自己位置を xy 座標とロボットの進行方向として出力する．

Configuration ポートの MAPRATE は生成する占有格子空間地図の画像データにおける 1 ピクセル当たりの地上の縮尺を示し，GANM は自己位置推定を行うための遺伝的アルゴリズムの個体数，T2 は一度の占有格子空間地図生成における突然変異の回数を指定することができる．

次の図 3 および表 5 に LocalizationRTC の仕様を示す．

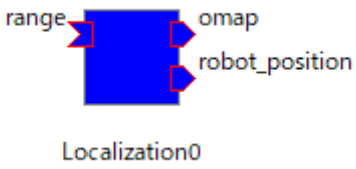


図 3 [System Editor]上に表示される LocalizationRTC

表 5　LocalizationRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	range	RangeData	-	-
OutPort	omap	TimedShortSeq	-	-
	robot_position	TimedPose2D	-	-
Configuration	MAPRATE	int	50	text
	GANM	int	600	text
	T2	int	2400	text

② Destination_genRTC

Destination_genRTC は，InPort の omap から受け取った占有格子空間地図を一次元配列から二次元配列へ復元し，この地図を OpenCV を用いて Map ウィンドウに表示する．この Map ウィンドウ上の任意の地点をマウスでクリックすることで，クリックした任意の目標位置を OutPort の dest_gen から xy 座標として出力する．

次の図 4 および表 6 に Destination_genRTC の仕様を示す．

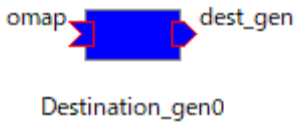


図 4　[System Editor]上に表示される Destination_genRTC

表 6　Destination_genRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	omap	TimedShortSeq	-	-
OutPort	dest_gen	TimedPose2D	-	-

③ GNG_componentRTC

GNG_componentRTC は，InPort の input_vec から受け取った占有格子空間地図を一次元配列から二次元配列へ復元し，この地図を用いて GNG を行うことでトポロジカルマップを生成し，OutPort の topomap から出力する．また，このトポロジカルマップを OpenCV を用いて gng ウィンドウに表示する．

次の図 5 および表 7 に GNG_componentRTC の仕様を示す．

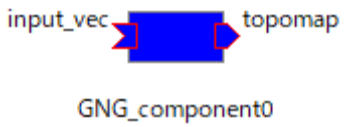


図 5 [System Editor]上に表示される GNG_componentRTC

表 7 GNG_componentRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	input_vec	TimedShortSeq	-	-
OutPort	topomap	Gng_path	-	-

④ PathPlanCompRTC

PathPlanCompRTC は，InPort の Goal で目標位置，Position で自己位置，TopoMap でトポロジカルマップを受け取り，これらを用いて経路計画を行い，受け取ったトポロジカルマップに，OpenCV を用いて輪郭ノードと経路計画，自己位置，目標位置を GNG_PathPlan ウィンドウに表示する．選択した経路計画の内，自己位置から最も近いノードの xy 座標を OutPort の Direction から出力する．

次の図 6 および表 8 に PathPlanCompRTC の仕様を示す．

図 6 [System Editor]上に表示される PathPlanCompRTC

表 8　PathPlanCompRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	Goal	TimedPose2D	-	-
	Position	TimedPose2D	-	-
	TopoMap	Gng_path	-	-
OutPort	Direction	TimedPose2D	-	-

⑤　FuzzyControlRTC

FuzzyControlRTC は，InPort の obstacleDistance は，LiDAR が一回転する間に取得した距離データの配列，localTarget は PathPlanCompRTC の Direction，RobotPosition は自己位置とロボットの進行方向，GoalPosition は目標位置を InPort から受け取り，これらをもとにファジィ制御と多目的行動調停を用いて最終的な制御量を決定し，ロボットの左右のタイヤの移動速度を OutPort の wheelLeft, wheelRight から出力している．

次の図 7 および表 9 に FuzzyControlRTC の仕様を示す．

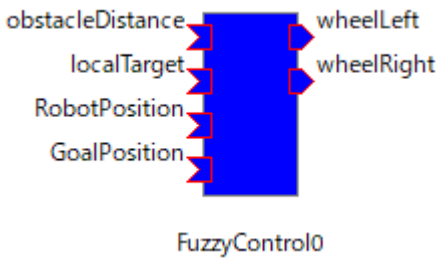


図 7　[System Editor]上に表示される FuzzyControlRTC

表 9　FuzzyControlRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	obstacleDistance	RangeData	-	-
	localTarget	TimedPose2D	-	-
	RobotPosition	TimedPose2D	-	-
	GoalPosition	TimedPose2D	-	-
OutPort	wheelLeft	TimedVelocity2D	-	-
	wheelRight	TimedVelocity2D	-	-

⑥ Roomba_OutPortRTC

Roomba_OutPortRTC は，ロボットの左右のタイヤの移動速度を InPort の wheel_l, wheel_r から受け取り，このそれぞれの移動速度をもとにルンバを制御している．Configuration ポートの port_name でルンバを接続した PC のポート番号を指定することができる．

次の図 8 および表 10 に Roomba_OutPortRTC の仕様を示す．

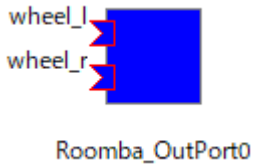


図 8 [System Editor]上に表示される Roomba_OutPortRTC

表 10 Roomba_OutPortRTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	wheel_l	TimedVelocity2D	-	-
	wheel_r	TimedVelocity2D	-	-
Configuration	port_name	string	COM6	text

4. 利用手順

4.1. RTC のダウンロードと共通する準備

ここでは，本プロジェクトで作成した RTC の利用手順の内，RTC のダウンロードと各 RTC の初期の共通する手順を解説する．

① PC にディスプレイ，バッテリー，LiDAR，ルンバを適切なポートに接続し，バッテリー，PC，ルンバの順に起動する．

② 本プロジェクトページから GNG-Unk-Env-Navi_RTC-main.zip ファイルをダウンロードし，図 9 のように解凍後の GNG-Unk-Env-Navi_RTC-main.zip フォルダー内にある全てのフォルダーを自身の OpenRTM-aist で使用している workspace 内に貼り付ける．（以下では，自身の workspace を”C:/Users/[UserName]/Desktop/workspace”にあるものと仮定し説明する．）

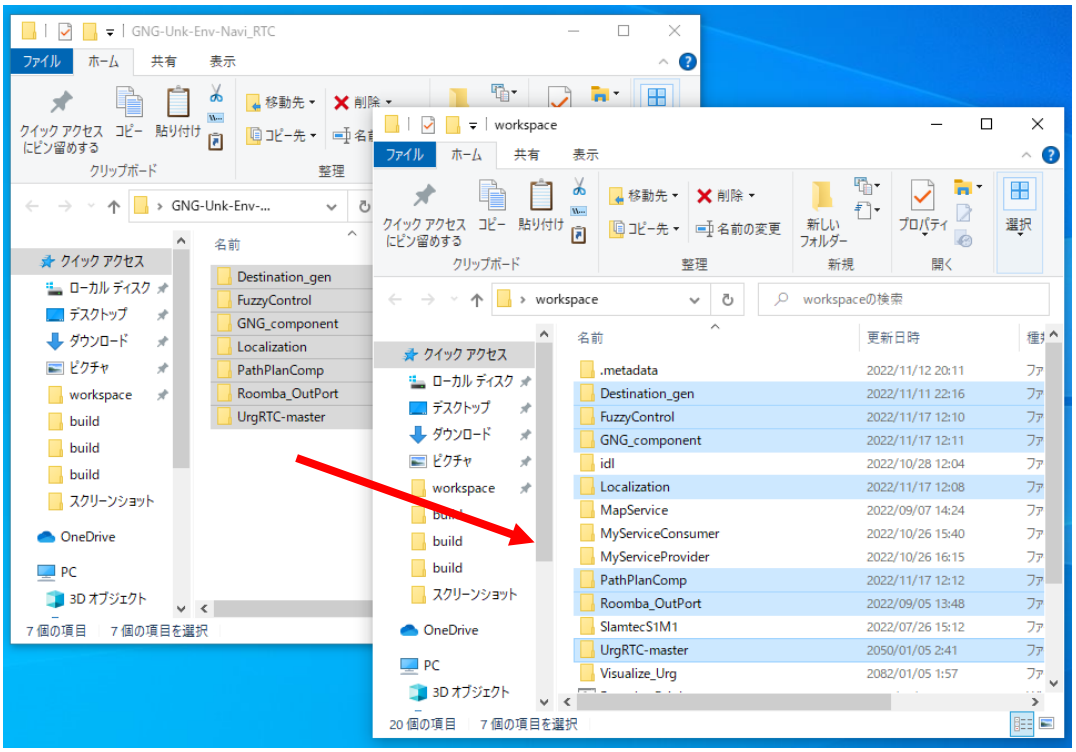


図 9　ダウンロードしたプロジェクトファイルを自身の workspace 内にドラッグ & ドロップ

③　図 10 のように[デスクトップ]から[スタート]ボタンを選択し，表示された項目の中から，[CMake]フォルダー内の[CMake (cmake-gui)]をクリックし，図 11 のようなアプリを起動させる．

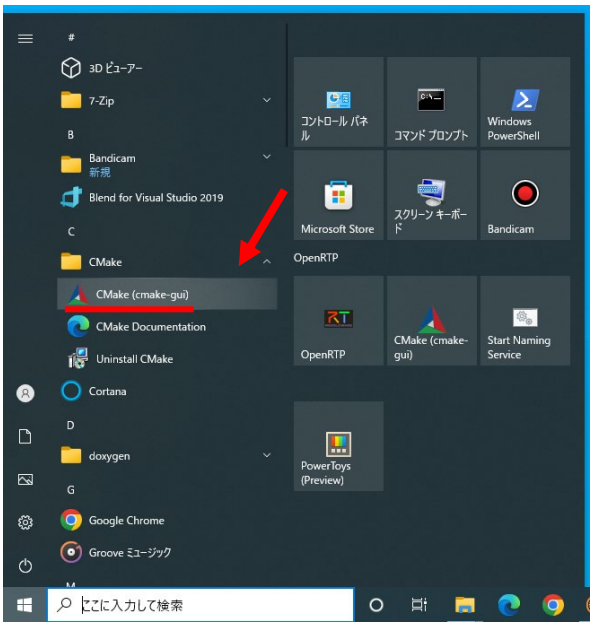


図 10　CMake の起動方法

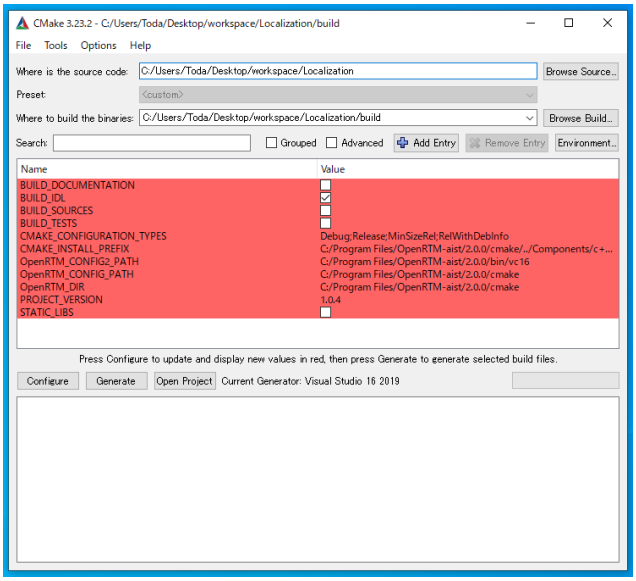


図 11　起動した CMake

ここからは LocalizationRTC を例に解説する．

④ 図 12 のように[Where is the source code]の項目の[Browse Source...]をクリックし，表示されたウィンドウ上で，自身の workspace 内の各 RTC のプロジェクトフォルダーを選択し，[フォルダーの選択]をクリックする．ここでは次のディレクトリを開く．
”C:/Users/[UserName]/Desktop/workspace/Localization”

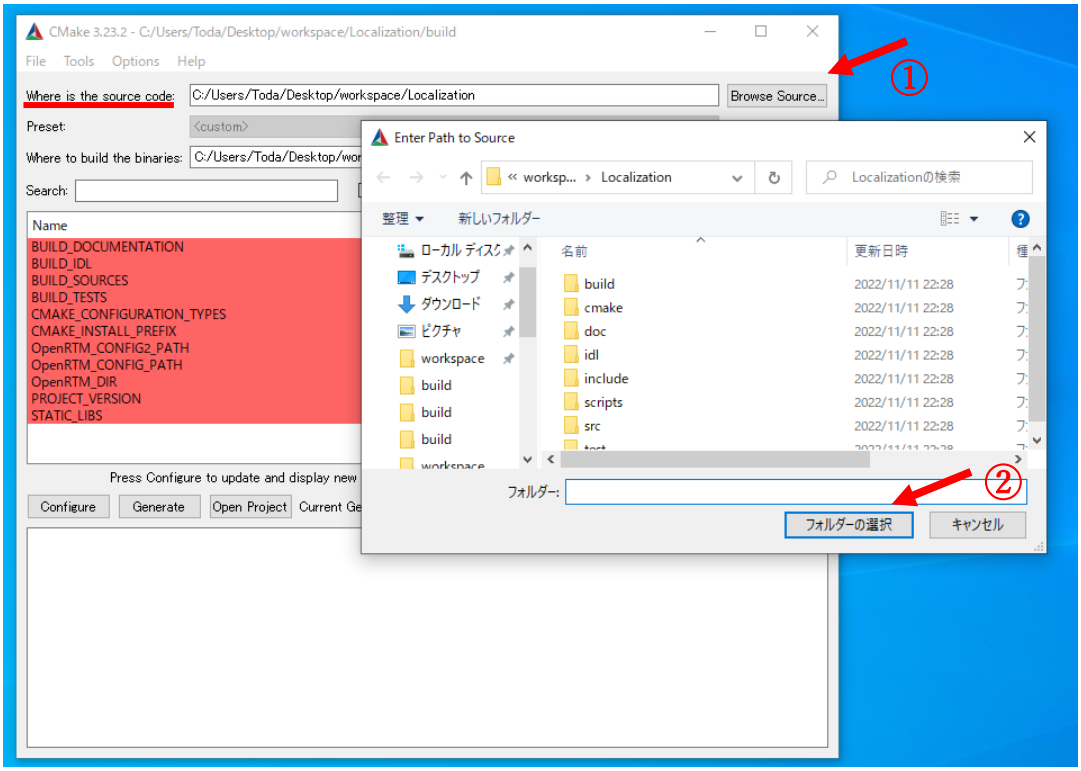


図 12　[Where is the source code]の項目の[Browse Source...]をクリックした状態

⑤　図 13 のように[Where to build the binaries]の項目の[Browse Source...]をクリックし，表示されたウィンドウ上で，自身の workspace 内の各 RTC のプロジェクトフォルダーを選択し，プロジェクトフォルダー内で[Ctrl]+[Shift]+N を押下し，[新しいフォルダー]が作成されるため，名前を[build]と変更する．この[build]フォルダーを選択し，[フォルダーの選択]をクリックする．ここでは次のディレクトリを開く．
”C:/Users/[UserName]/Desktop/workspace/Localization/build”

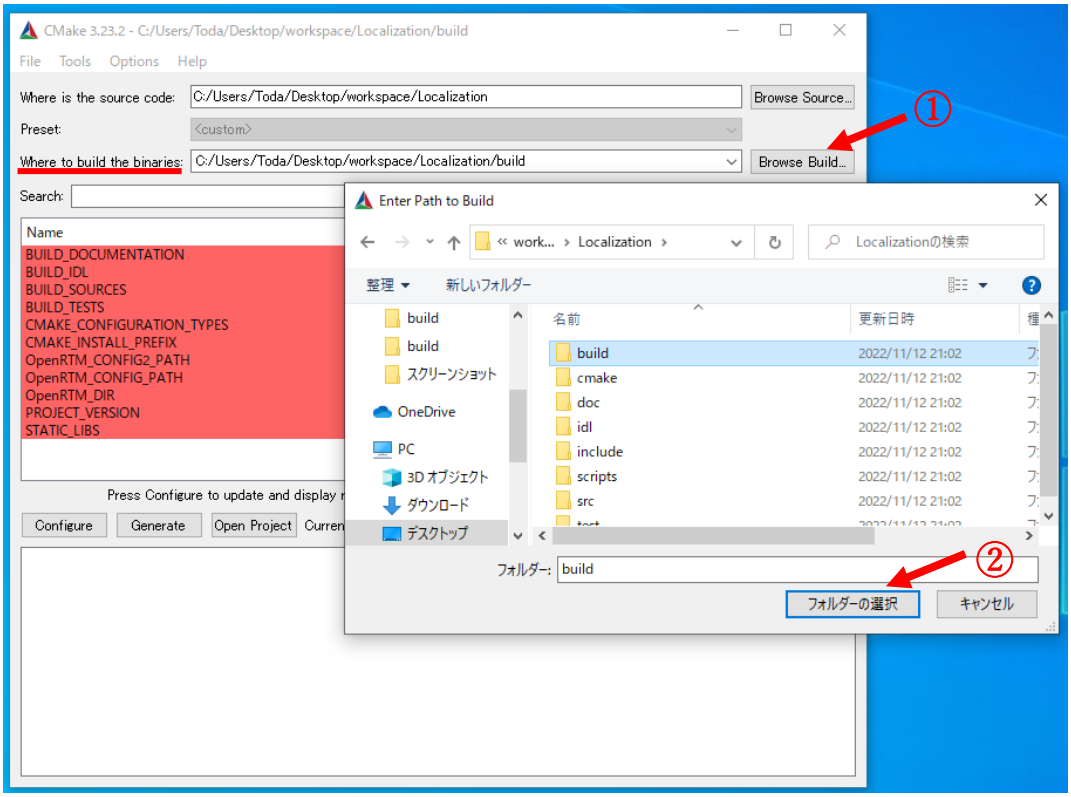


図 13　[Where to build the binaries]の項目の[Browse Source...]をクリックした状態

※　[Where to build the binaries]のディレクトリを指定後，[Where is the source code]が変更されていることがあるため，指定したディレクトリが正しいかどうか確認をする必要がある．

⑥　図 14 のように[メニューバー]の[File]をクリックし，[Delete Cache]をクリックする．

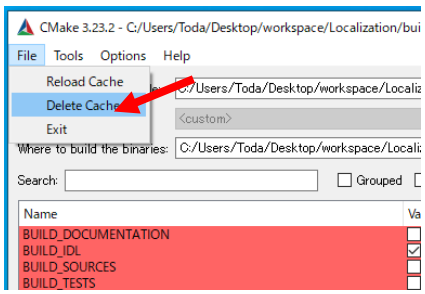


図 14　[Delete Cache]の場所

⑦　図 15 のように[CMake]ウィンドウ画面下方の[Configure]をクリックし，図 16 のように[Specify the generator for this project]の項目から，自身の PC で有効な[Visual Studio]を選択し，[Finish]をクリックする．
ここでは，[Visual Studio 2019]を選択する．

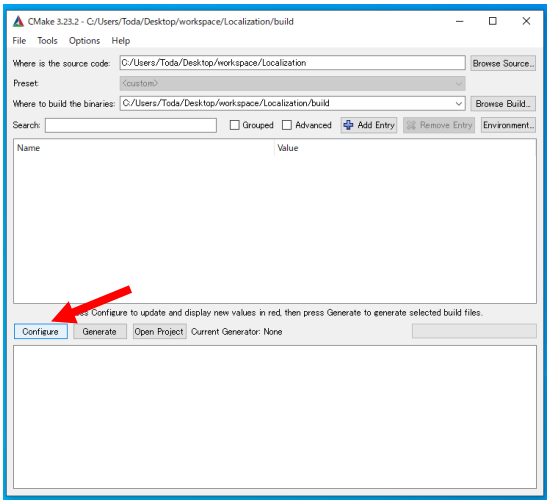


図 15　[Configure] ボタンの位置

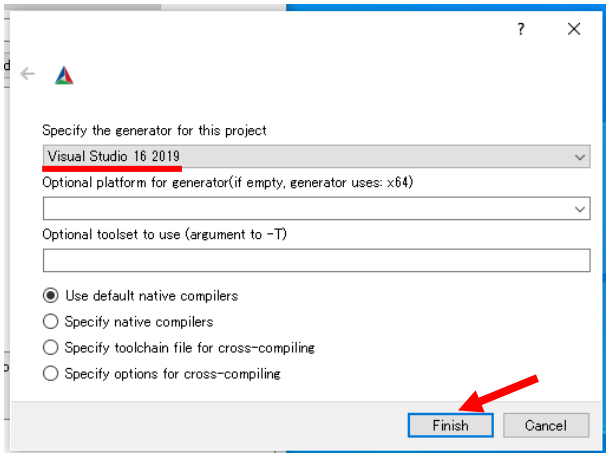


図 16　[Specify the generator for this project]の項目から，自身の PC で有効な[Visual Studio]を選択する

⑧　図 17 のようにウィンドウ下部のログに[Configuring done]と表示されたことを確認し，[Generate]を選択する．

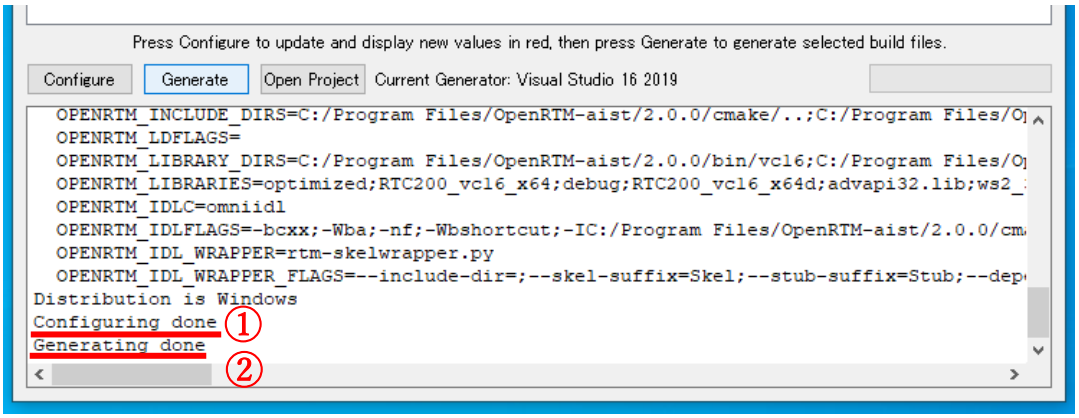


図 17　ウィンドウ下部のログに[Configuring done]と表示された状態

⑨　図 17 のように，同じくウィンドウ下部のログに[Generating done]が表示されたことを確認し，他の各 RTC の CMake を行う．すべての RTC の CMake が終了したことを確認後，ウィンドウ右上の[×]を選択し，[CMake (cmake-gui)]を閉じる．

4.2. Visual Studio の操作

ここからは LocalizationRTC を例に解説する.

① 図 18 のように自身のワークスペース内の各 RTC のプロジェクトファイル内の[build]を開き，[プロジェクト名].sln ファイルをダブルクリックして[Visual Studio 2019]（CMake を行った際に選択した Visual Studio のバージョンを選択する.）で開く．ここでは次のディレクトリのファイルを開く.
”C:/Users/[UserName]/Desktop/workspace/[プロジェクト名]/build/[プロジェクト名].sln”

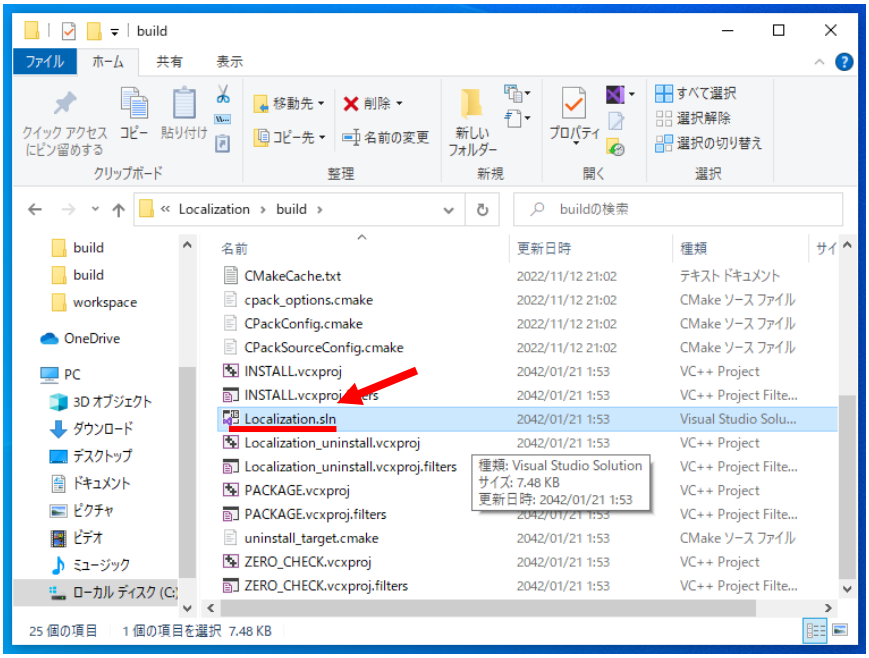


図 18　プロジェクトファイル内の[build]にある[プロジェクト名].sln ファイル

② 図 19 のように Visual Studio の画面下部のエラー一覧にエラーが表示されていないことを確認し，図 20 のように [メニューバー]の[ビルド]の中にある[ソリューションのリビルド]をクリックする.

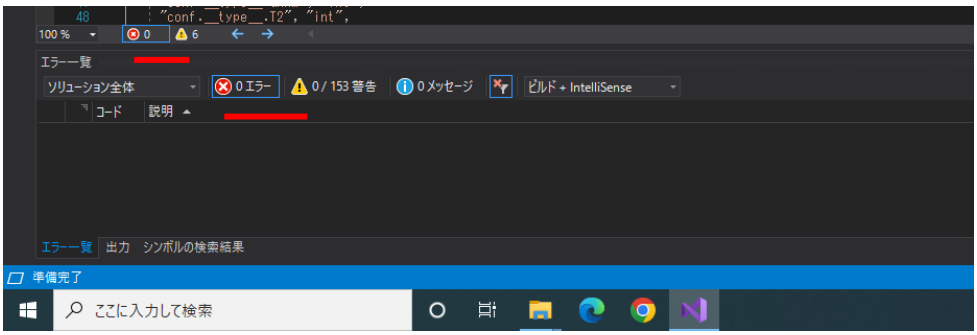


図 19　画面下部のエラー一覧にエラーが表示されていない状態

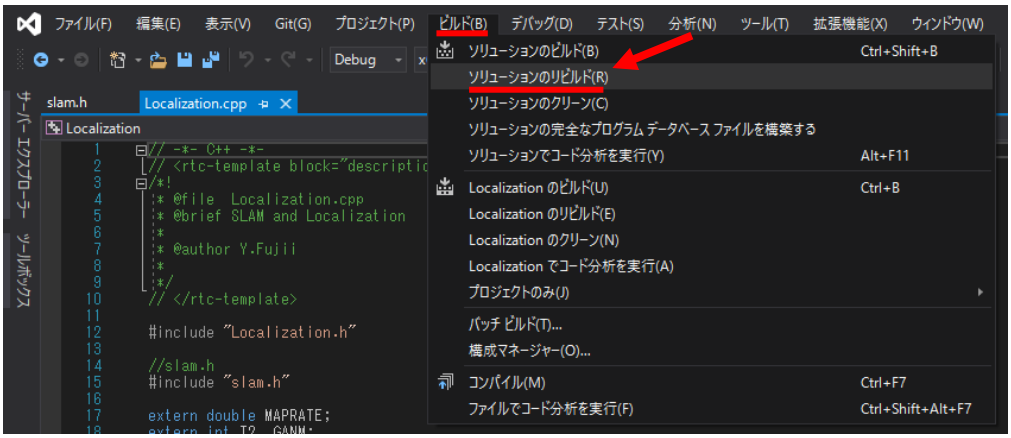


図 20　[メニューバー]の[ビルド]の中にある[ソリューションのリビルド]

③　図 21 のように Visual Studio の［ステータスバー］に［リビルドがすべて正常に終了しました。］と，表示されたことを確認し，ウィンドウ右上の［×］をクリックし，Visual Studio を終了する．他の RTC も同じようにリビルドを行う．

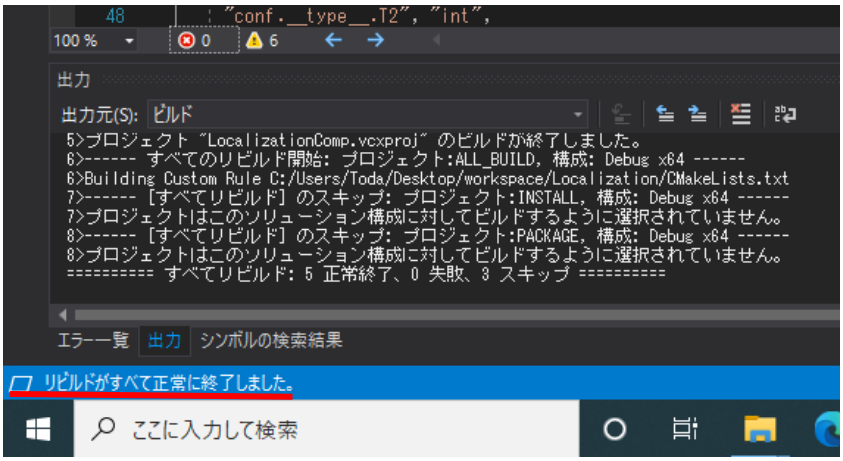


図 21　［ステータスバー］に［リビルドがすべて正常に終了しました。］と，表示された状態

4.3. 各種 RTC の起動と Configuration ポートの設定と注意

① 図 22 のように[デスクトップ]から[スタート]ボタンを選択し，表示された項目の中から，[OpenRTM-aist 2.0.0 x86_64]フォルダー内の[OpenRTP]をクリックし，アプリを起動させる．

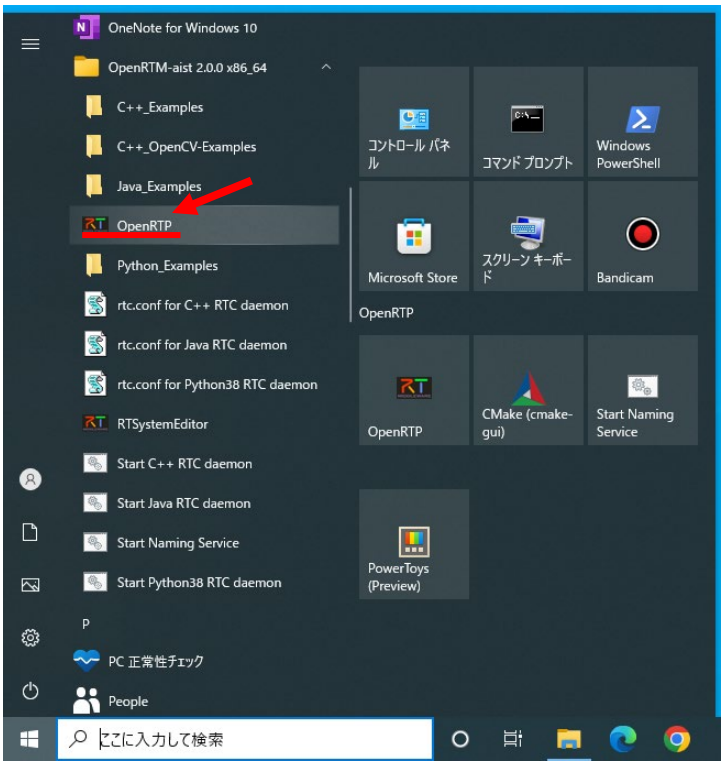


図 22 [OpenRTP]の起動方法

② 図 23 のように[Eclipse SDK ランチャー]が起動するため，[ワークスペース]の項目の[参照]をクリックし，自身の workspace を選択し，[起動]をクリックする．

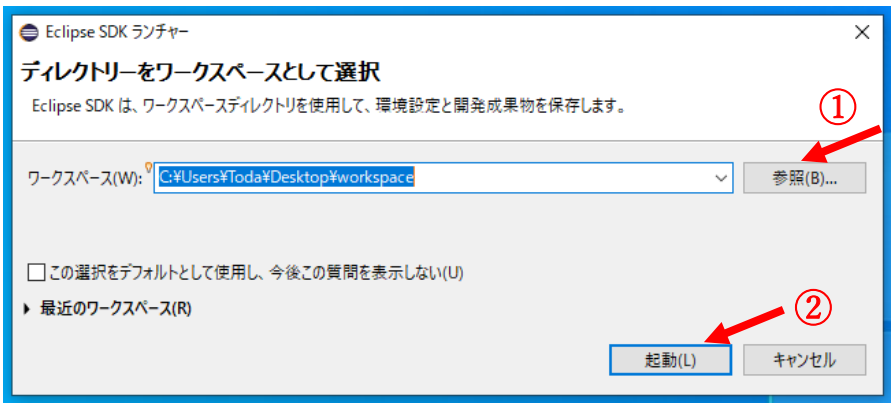


図 23 [Eclipse SDK ランチャー]ウィンドウの[ワークスペース]を選択する画面

③　図 24 のように OpenRTP の[メニューバー]の右側にある[パースペクティブを開く]をクリック.

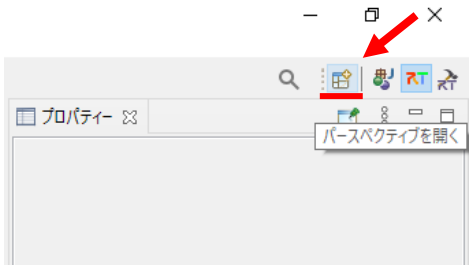


図 24　[パースペクティブを開く]の場所

④　図 25 のように表示されたウィンドウの選択項目の中から[RTSystemEditor]を選択し，クリックする.

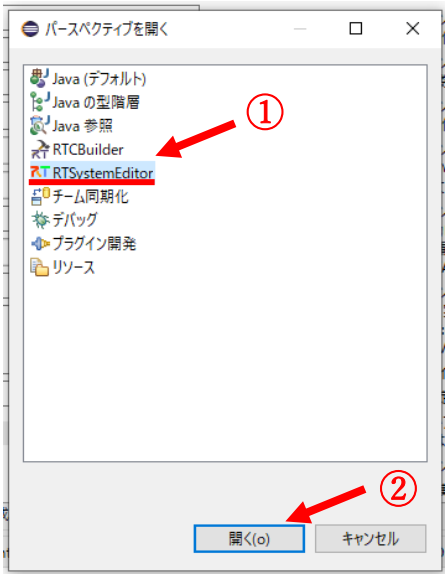


図 25　[パースペクティブを開く]ウィンドウの[RTSystemEditor]の場所

⑤　図 26 のように[ツールバー]の中から，[Open New System Editor]をクリック.

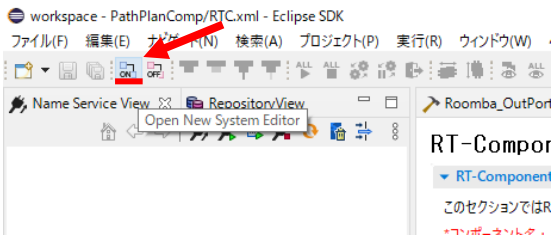


図 26　[ツールバー]内の[Open New System Editor]の場所

⑥　図 27 のように[Name Service View]の[ツールバー]から[ネームサービスを起動]をクリック.

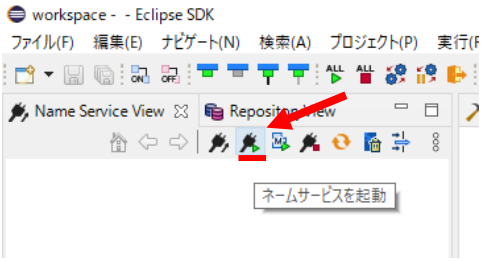


図 27　[Name Service View]の[ツールバー]内の[ネームサービスを起動]の場所

⑦　図 28 のように[デスクトップ]から[スタート]ボタンを選択し，表示された項目の中から，[OpenRTM-aist 2.0.0 x86_64]フォルダー内の[Start Naming Service]をクリックし，アプリを起動させる.

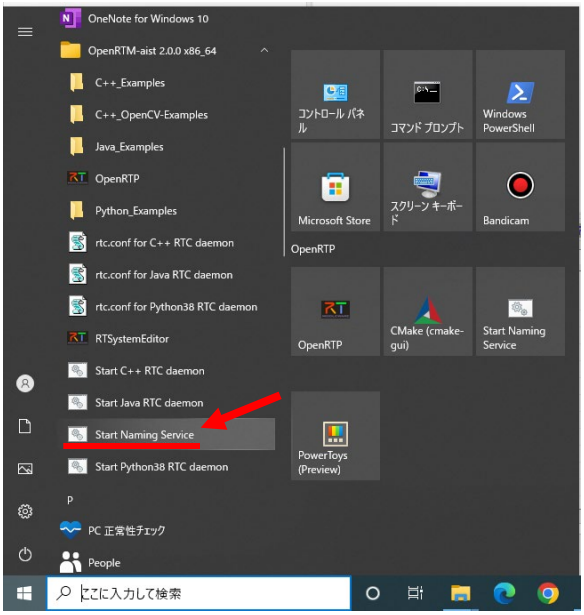


図 28　[Start Naming Service]の起動方法

⑧　図 29 のように，表示された[Start Naming Service]のコンソール左上に[成功]と表示されていることを確認し，コンソールウィンドウを最小にする.

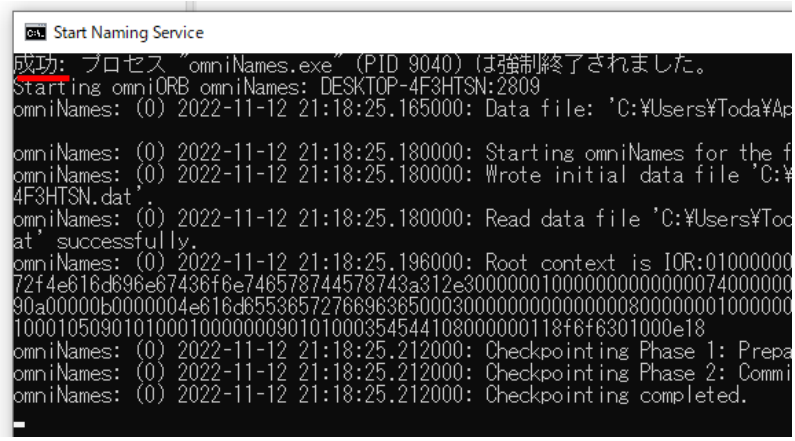


図 29　[Start Naming Service]のコンソール左上に[成功]と表示された状態

⑨　図 30 のようにエクスプローラーで, 本プロジェクトでダウンロードしたすべての RTC プロジェクトファイル内にある次の階層の "[プロジェクト名]Comp.exe" ファイルをそれぞれダブルクリックし，起動させる.

"C:/Users/[UserName]/Desktop/workspace/[プロジェクト名]/build/src/Debug/[プロジェクト名]Comp.exe"

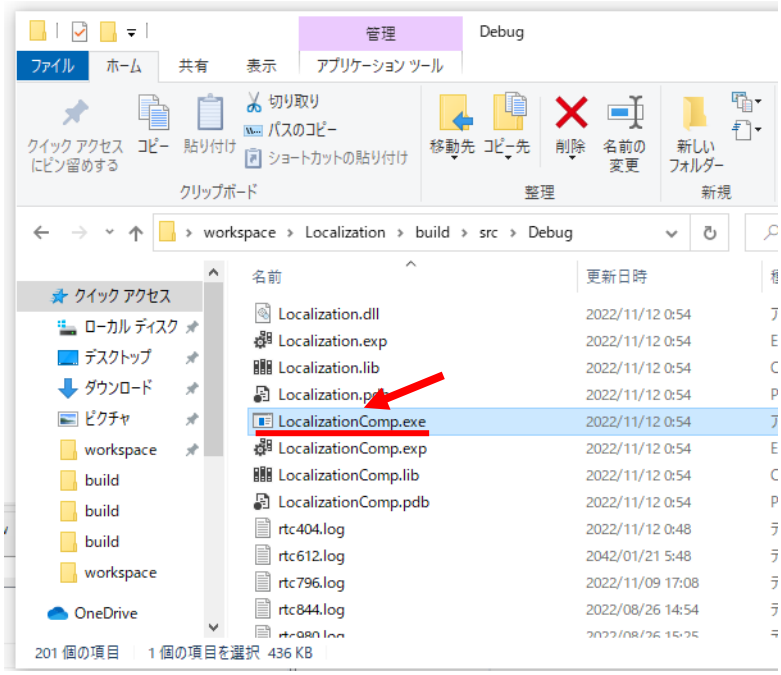


図 30　"[プロジェクト名]Comp.exe" ファイル

⑩ 図 31 のように OpenRTP の[Name Service View]内にある[localhost]の左にある[>]をクリックし，[[デスクトップ名]|host_cxt]の左にある[>]をクリックし，図 32 のように表示されたすべての[[プロジェクト名]0|rtc]を[System Editor]上にドラッグ & ドロップする．

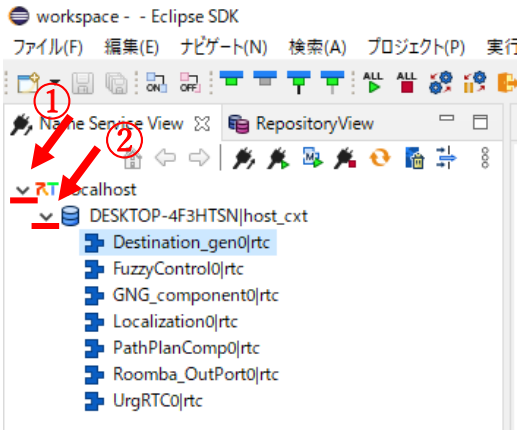


図 31　[Name Service View]内にある[localhost]と[[デスクトップ名]|host_cxt]を開いた状態

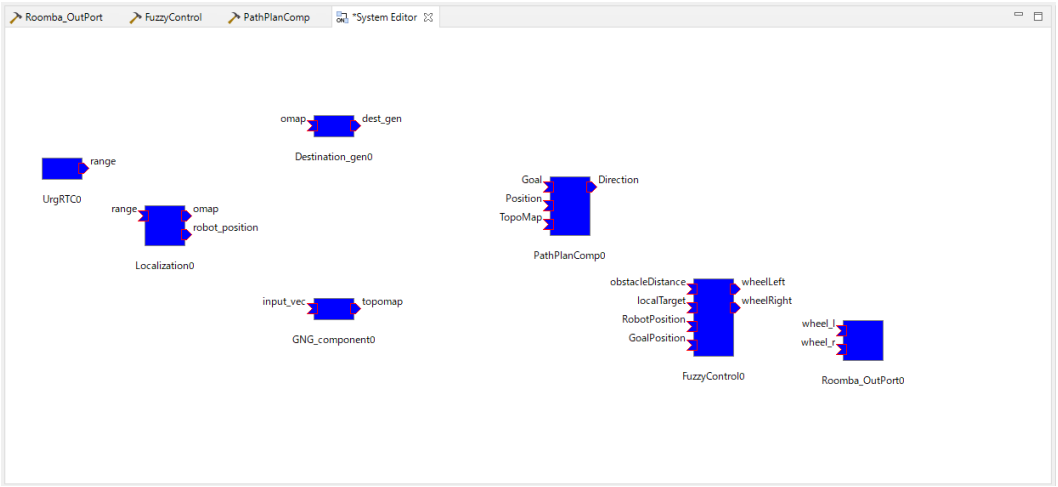


図 32　[System Editor]上に配置された[[プロジェクト名]0|rtc]の配置例

⑪ 図 33 および図 34 のように[System Editor]上で，[UrgRTC0]と[Localization0]，[Roomba_OutPort0]は画面中央下部の[Configuration View]で Configuration ポートのデータを変更できる．基本的に変更しないが，[UrgRTC0]と[Roomba_OutPort0]の[port_name]を適切なポート名に変更する必要がある．

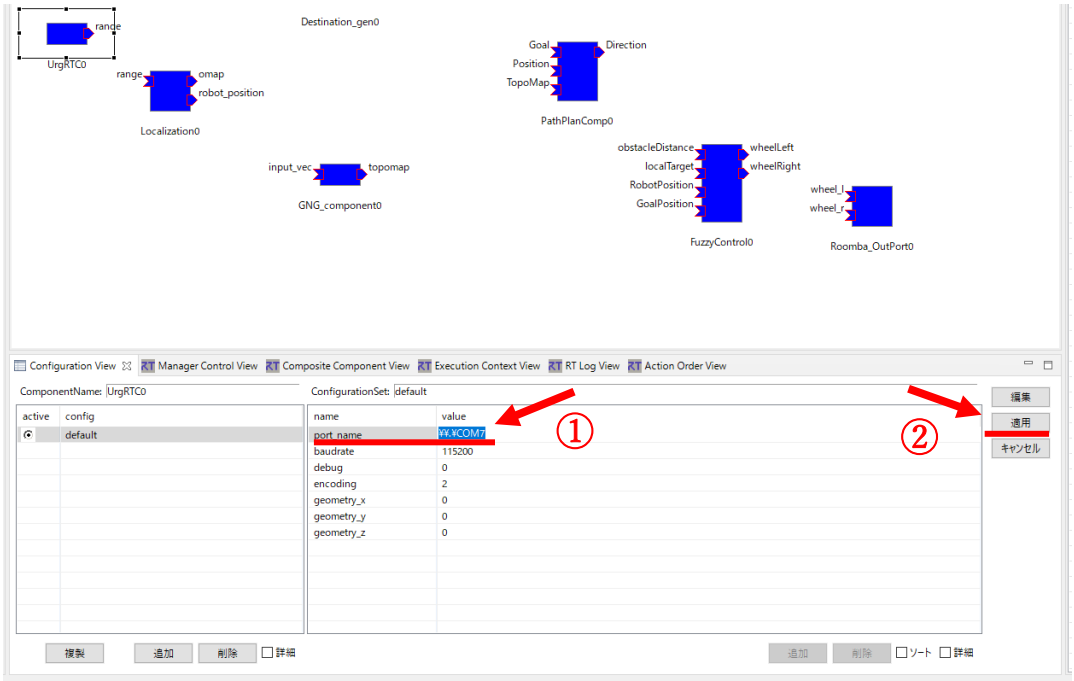


図 33 ”UrgRTC0” の Configuration ポート変更画面

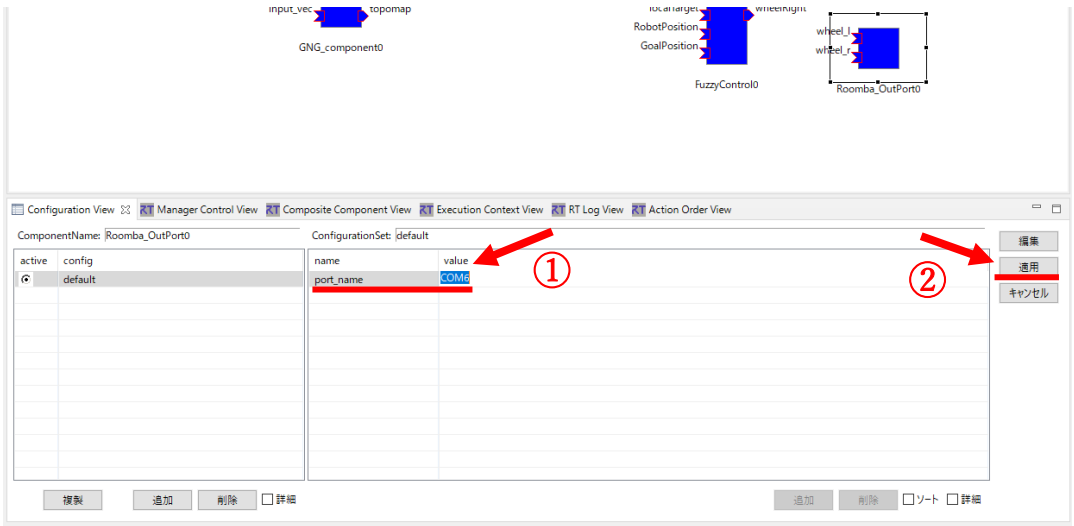


図 34 ”Roomba_OutPort0” の Configuration ポート変更画面

⑫　図 35 のように[デスクトップ]から[スタート]ボタン上で[右クリック]し，表示された項目の中から，[デバイスマネージャー]をクリックし，アプリを起動させる．

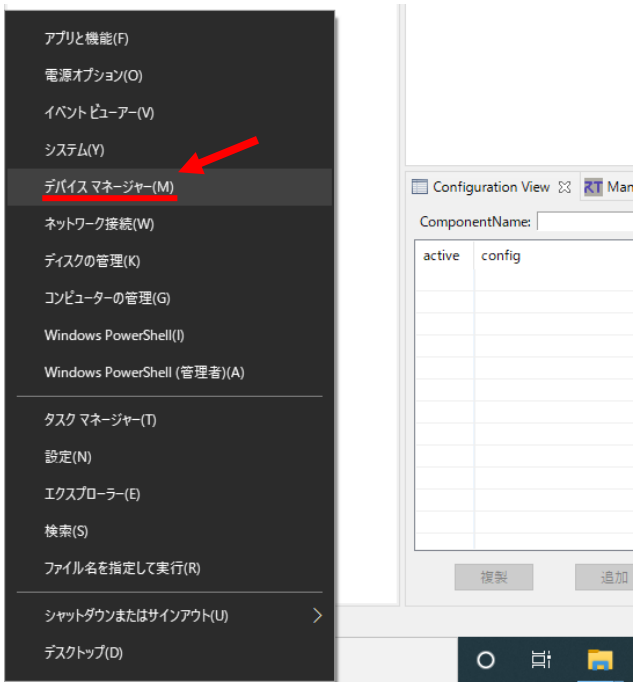


図 35　[スタート]ボタン上で[右クリック]し，表示させた[デバイスマネージャー]の項目

⑬　図 36 のように表示されたウィンドウの[ポート(COM と LPT)]内の[USB Serial Port (COM7)]をもとに，[UrgRTC0]の[port_name]に ”¥¥.¥COM7” を記述し，[USB シリアル デバイス (COM6)]をもとに，[Roomba_OutPort0]の[port_name]に ”COM6” を記述し，[適用]をクリックする．

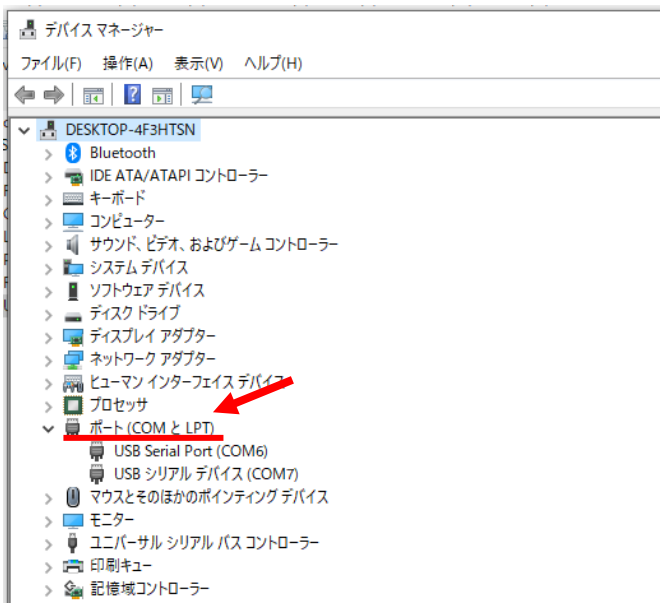


図 36　デバイスマネージャーウィンドウの[ポート(COM と LPT)]

⑭　図 37 および表 11 をもとに，各ポートをドラッグ ＆ ドロップで繋ぎ，表示されたウィンドウの[OK]をクリックする．

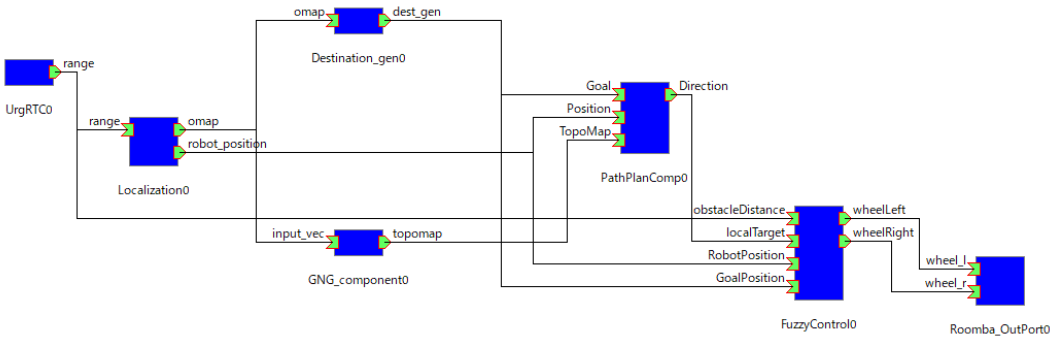


図 37　各ポートを実際に接続した状態

表 11　各種 RTC におけるポートの接続先

出力 RTC	OutPort 名	入力 RTC	InPort 名
UrgRTC	range	Localization	range
		FuzzyControl	obstacleDistance
Localization	omap	Destination_gen	omap
		GNG_component	input_vec
	robot_position	PathPlanComp	Position
		FuzzyControl	RobotPosition
Destination_gen	dest_gen	PathPlanComp	Goal
		FuzzyControl	GoalPosition
GNG_component	topomap	PathPlanComp	TopoMap
PathPlanComp	Direction	FuzzyControl	localTarget
FuzzyControl	wheelLeft	Roomba_OutPort	wheel_l
FuzzyControl	wheelRight	Roomba_OutPort	wheel_r

4.4. RTC の Activate 化

① 図 38 のように［UrgRTC0]を[右クリック]し，[Activate]をクリックする．

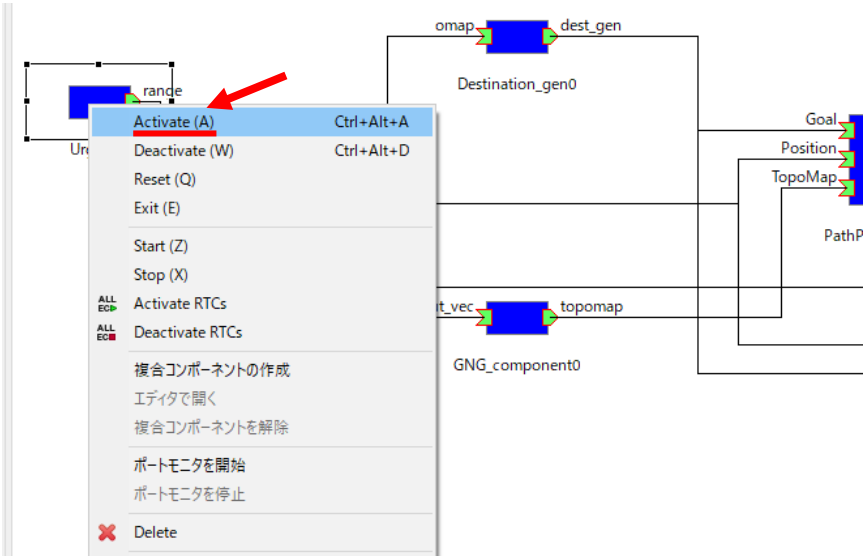


図 38 ”UrgRTC0” の[Activate]

② 図 39 の[進行情報]ウィンドウが表示され，これが閉じたことを確認した後に，図 40 のように[System Editor]上の空白部分で[右クリック]し，[ALL Activate Systems]をクリックし，図 41 のようにすべての RTC が有効になり，OpenCV のウィンドウが 3 つ表示される．

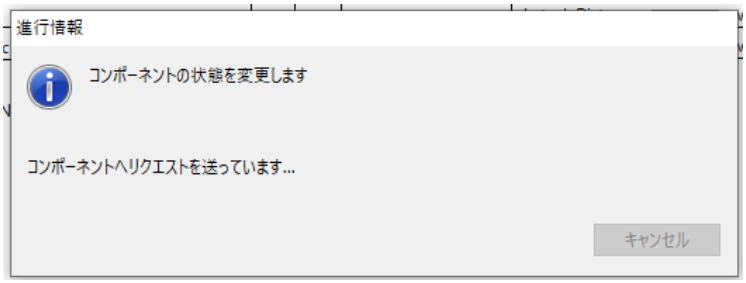


図 39 [進行情報]ウィンドウ

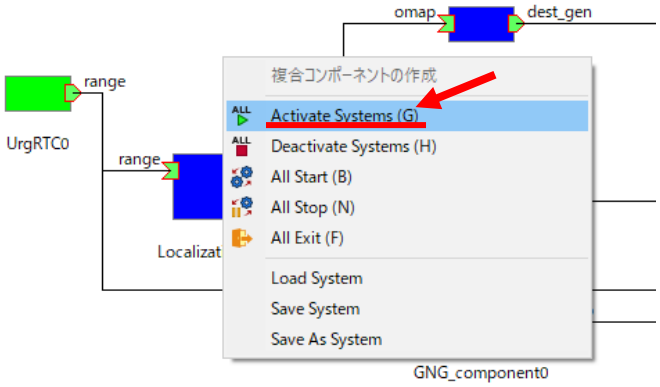


図 40　[System Editor]上の空白部分で[右クリック]し，表示させた[ALL Activate Systems]

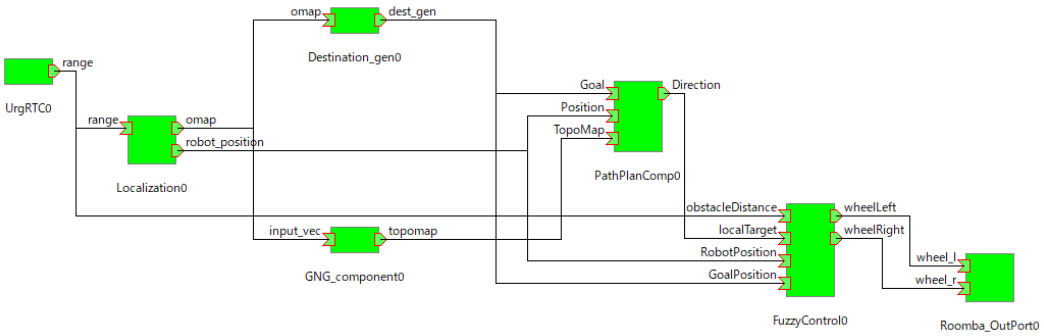


図 41　すべての RTC が有効となった状態

③　図 42 のように Destination_genRTC の表示する[Map]ウィンドウに占有格子空間地図が表示され，この地図内をクリックすることで目的地が設定され，ルンバが移動を開始する．

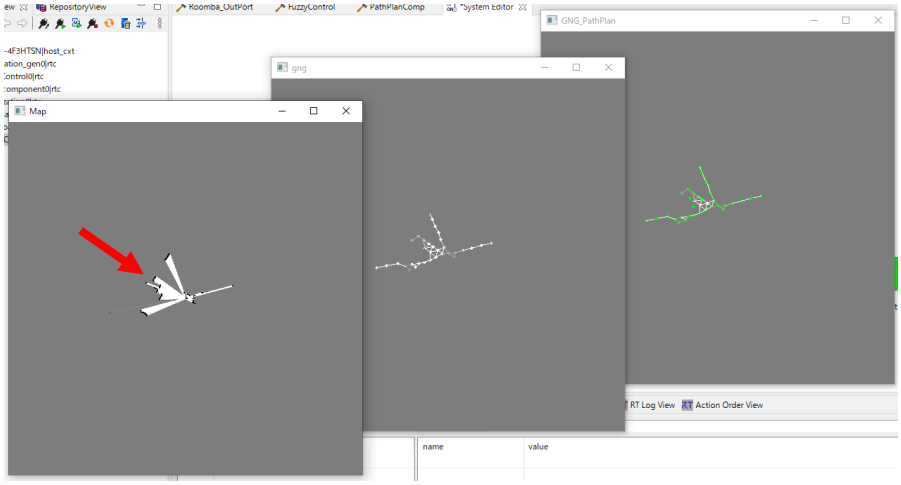


図 42　すべての RTC を[ALL Activate Systems]することで表示される 3 つのウィンドウ

④　図 43 のように[System Editor]上の空白部分で[右クリック]し，[All Exit]をクリックし，すべての RTC が終了し，[Start Naming Service]のコンソールを除く，開いていたコンソールと OpenCV のウィンドウがすべて閉じられ，[Name Service View]および[System Editor]上の RTC の表示が消える．

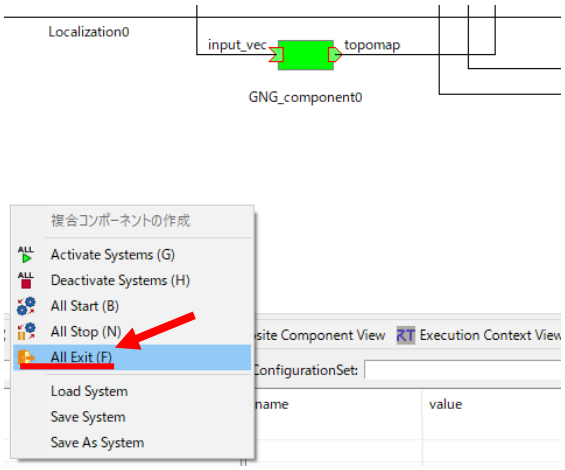


図 43　[System Editor]上の空白部分で[右クリック]し，表示させた[All Exit]

5. 動作例

5.1. 実際の動作

① [UrgRTC0]を Activate 化し，ほかの RTC を ALL Activate Systems すると，Map ウィンドウが表示され，図 44 のような占有格子空間地図が表示される．表示された地図の白色の部分が LiDAR で計測した移動可能領域であり，黒色の部分が LiDAR で計測した障害物，灰色の部分が未計測の未知環境領域である．

図 44　Map ウィンドウに表示された占有格子空間地図

② 図 45 のような gng ウィンドウには幾何的な環境地図の構造を保持しつつ占有情報も含んだトポロジカルマップが表示され，図 46 のような GNG_PathPlan ウィンドウには gng ウィンドウに表示された地図と自己位置推定の結果をもとに，経路を計画し，目標追従や障害物回避の制御を行った結果を表示する．gng ウィンドウと GNG_PathPlan ウィンドウにおいて，白色の丸い部分がノードであり，それらを繋ぐ線がエッジである．GNG_PathPlan ウィンドウにおける小さい緑色の丸は移動可能領域の輪郭ノードを示す．

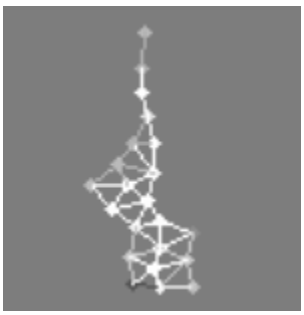


図 45　gng ウィンドウ

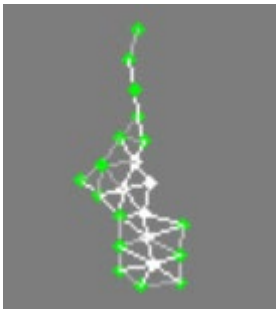


図 46　GNG_PathPlan ウィンドウ

③ Map ウィンドウの占有格子空間地図上をクリックすると目的地が設定され，ルンバが移動を開始する．図 47 のように GNG_PathPlan ウィンドウの水色の丸い部分がルンバの現在位置を示し，赤色の丸い部分が次に移動する場所を示し，赤い線が経路計画を示し，緑色の丸い部分が目的地に最も近い部分を示す．また，目的地に灰色の未知環境領域を指定していた場合，目的地に最も近いノードを目的地として経路計画を行う．

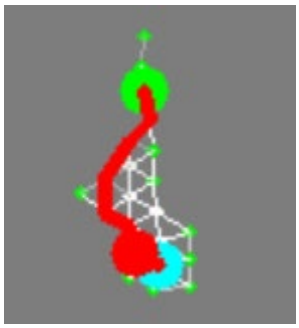


図 47　経路計画を行う GNG_PathPlan ウィンドウ

6. 参考文献

ysuga. (2022 年 11 月 11 日). 北陽電機 URG センサ RTC. 参照先: OpenRTM-aist: https://www.openrtm.org/openrtm/ja/project/Hokuyo_URG_RTC_by_SSR

株式会社 セック 開発本部 第四開発部（ＲＴミドルウェア担当）. (2022 年 11 月 11 日). Classic-URG RTC （北陽電機社製：URG-04LX）. 参照先: OpenRTM-aist: https://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID130

産業技術総合研究所. (2022 年 11 月 11 日). OpenRTM-aist を 10 分で始めよう！ 参照先: OpenRTM-aist: https://www.openrtm.org/openrtm/ja/doc/installation/lets_start

産業技術総合研究所. (2022 年 11 月 11 日). 画像処理コンポーネントの作成（Windows 8.1、OpenRTM-aist-1.1.2-RELEASE、OpenRTP-1.1.2、CMake-3.5.2、VS2015). 参照先: OpenRTM-aist: https://www.openrtm.org/openrtm/ja/node/6057

Growing Neural Gas を用いた未知環境ナビゲーションシステムのコンポーネントの構築

2022 年 11 月 16 日　初　版第 1 刷発行
2022 年 12 月 10 日　第 2 版第 1 刷発行

編　集　藤井雄基
発行元　岡山大学工学部機械システム系学科システム工学コース
　　適応学習システム制御学研究室