洞窟の 3 次元点群を用いた図面化システムの RT コンポーネント開発

藤井雄基，戸田雄一郎，松野隆幸
岡山大学 適応学習システム制御学研究室

目次

1. はじめに

1.1. 目的

洞窟での測量は，地理学や地質学的な観点から重要なタスクの一つであり，センサ技術を利用したデジタル化が望まれている一分野である．特に，3 次元距離計測は洞窟内の地図構築や洞窟の形状把握が可能となるため，小型化された計測システムと解析技術の発展が望まれている．そこで本研究では，昨年作成した測量システムを拡張し，洞窟で計測された 3 次元点群データをもとに図面化するシステムを RT コンポーネント （RTC） により開発することで，容易に洞窟の図面化が可能となるシステム化を目指す．

1.2. 概要

本プロジェクトでは，すべてのコンポーネントである，EtheURG, MeasurementSystem RTC, Registration RTC, Analyses RTC, PointCloud_Viewer RTC, PointCloud_Reader RTC, FPS RTC, WallDTC RTC, Contour RTC, MapViewer RTC の RTC 開発を行った．基本的な流れとしては，EtheURG RTC 及び，MeasurementSystem RTC により，3 次元点群を計測し，その結果を可視化用コンポーネントである PointCloud_Viewer RTC で可視化する流れとなっている．また，複数の場所で計測された点群データは Registration RTC により一つの点群データとして統合され，洞窟全体の地図として可視化できるような構成となっている．さらに，統合された点群データは Analyses RTC で Growing Neural Gas (GNG) を用いた解析が行われ，その解析結果も PointCloud_Viewer RTC で可視化可能である [1].

これらの機能を拡張し， Registration RTC で統合された点群を FPS RTC で粗密に関わらず均等にダウンサンプリングを行う．このデータを WallDTC RTC と Contour RTC に送り，WallDTC RTC で洞壁のクラスタを生成し，Contour RTC でエレベーションマップを生成する．これらを MapViewer RTC で統合して可視化する構成となっている．さらに，PointCloud_Reader RTC で既に生成されている点群データをファイルから読み込むことで，この大規模な RT システムの途中から処理することが出来るようになっている．

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
Laser Range Scanner （以下，LiDAR と表記）	北陽電機株式会社社	UTM-30LX-EW
サーボモータ	ROBOTIS	Dynamixel XL430-W250-T
ノート PC （以下，PC と表記）	Panasonic	CF-SX2JDRYS

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金属板とネジを用いて三脚に図 1 のように固定した．

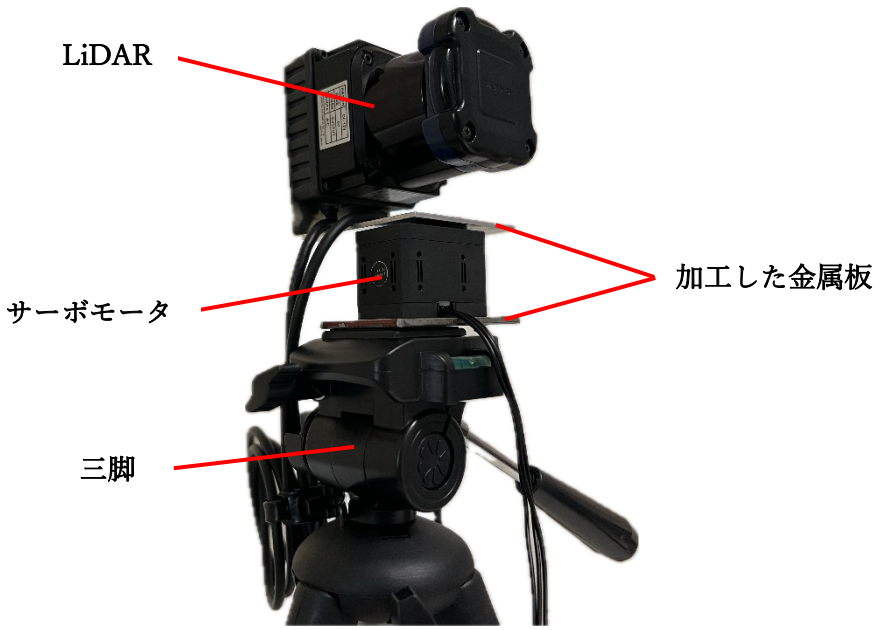

図 1 実際に作成したハードウェア

3. ソフトウェア

3.1. 開発環境

本プロジェクトで開発した EtheURG RTC は一部 C 言語で作成されたライブラリを用いて C++ で開発した．その他の RTC はすべて C++ で開発した．

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．以下の項目は各 RTC で利用する各ライブラリの境構築方法である．

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 11 Pro	23H2
RT ミドルウェア	OpenRTM-aist	2.0.1 x86_64
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	3.27.7
統合開発環境	Visual Studio Community 2022	17.11.5
パッケージ管理システム	chocolatey	2.2.2
ライブラリ	urg_library DynamixelSDK PCL All-in-one-installer Eigen	2023 年 10 月現在 3.7.31 1.13.1 3.4.0

① urg_library

urg_library は，北陽電機社製の LiDAR を利用するために必要なライブラリである．本プロジェクトでは EtheURG RTC で使用している． [2, 3]

表 3 は利用した公式 HP とライブラリのリンクである. 本 RTC を利用するためにはライブラリを表 4 にあるクローン場所の階層に GitHub からクローンし，表 4 にある階層のソリューションファイルでビルドする必要がある． [2]

表 3 urg_library に関する URL

ページ種別	URL
HP	https://sourceforge.net/p/urgnetwork/wiki/top_jp/
ライブラリのコンパイル方法	https://urgnetwork.sourceforge.net/html_ja/usage_windows_vcproj_page.html
GitHub	https://github.com/UrgNetwork/urg_library.git

表 4 urg_library 利用のための準備に使うファイル階層の一覧

階層種別	階層
クローン場所	C:¥src¥urg_library
ソリューションファイル	C:¥src¥urg_library¥current¥vs2019¥c¥urg.sln

② DynamixelSDK

DynamixelSDK は，ROBOTIS 社製のサーボモータを利用するために必要なライブラリである．本プロジェクトでは MeasurementSystem RTC で使用している．Dynamixel を利用するには，電源(コンセントまたはバッテリー)，電源や TTL I/F の信号を分配する機器が必要である．これらの接続方法は表 5 の DYNAMIXEL basic tutorial を参考にしてもらいたい．[4, 5]

表 5 は，参考サイトと GitHub のリンクである．本 RTC を利用するには，表 5 の GitHub のリンクから表 6 のクローン場所である C ドライブ直下へ，表 7 のコマンドを使ってライブラリをクローンする．クローン後，表 6 にある C++ のソリューションファイルでビルドする必要がある．また，表 8 の環境変数をシステム環境変数に設定する．[4, 6]

表 5 DynamixelSDK に関する URL

ページ種別	URL
DYNAMIXEL SDK	https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/
DYNAMIXEL basic tutorial	https://www.besttechnology.co.jp/modules/knowledge/?DYNAMIXEL%20basic%20tutorial
GitHub	https://github.com/ROBOTIS-GIT/DynamixelSDK.git

表 6 DynamixelSDK 利用のための準備に使うファイル階層の一覧

階層種別	階層
クローン場所	C:¥DynamixelSDK
ソリューションファイル	C:¥DynamixelSDK¥c++¥build¥win64¥dxl_x64_cpp.sln

表 7 DynamixelSDK のインストールに使うコマンド

インストールソフト	コマンド
DynamixelSDK	$ git clone https://github.com/ROBOTIS-GIT/DynamixelSDK.git

表 8 DynamixelSDK の設定に必要な環境変数の一覧

環境変数名	階層
path	C:¥DynamixelSDK¥c++¥build¥win64¥output

③　Point Cloud Library

Point Cloud Library（PCL）は，点群を処理する一般的なライブラリである．また，「3.2. 再利用を検討した RTC の概要」において後述する GitHub で公開されている idl を利用するために必要なライブラリである．本プロジェクトでは MeasurementSystem RTC, Registration RTC, Analyses RTC, PointCloud_Viewer RTC, PointCloud_Reader RTC, FPS RTC, WallDTC RTC, Contour RTC, MapViewer RTC で使用している．ここでは，PCL の All-in-one-installer を使ってインストールする． [7, 8, 9]

表 9 の GitHub にある「PCL-1.13.1-AllInOne-msvc2022-win64.exe」をダウンロードし，ダウンロード後にこの exe ファイルを実行することで PCL をインストールする．詳細な設定方法は表 9 の金子邦彦研究室を参考にしてもらいたい． [7, 9]

また，上記サイトを参考に表 10 に示した環境変数をシステム環境変数に設定する． [9]

表 9　PCL に関する URL

ページ種別	URL
金子邦彦研究室	https://www.kkaneko.jp/db/win/libpcl.html
GitHub	https://github.com/PointCloudLibrary/pcl/releases

表 10　PCL の設定に必要な環境変数の一覧

環境変数名	階層
PKG_CONFIG_PATH	C:¥Program Files¥PCL 1.13.1¥lib¥pkgconfig
PCL_ROOT	C:¥Program Files¥PCL 1.13.1
OPENNI2_INCLUDE64	C:¥Program Files¥OpenNI2¥include¥
OPENNI2_LIB64	C:¥Program Files¥OpenNI2¥lib¥
OPENNI2_REDIST64	C:¥Program Files¥OpenNI2¥Redist¥
Boost_ROOT	C:¥Program Files¥PCL 1.13.1¥3rdParty¥Boost
EIGEN_ROOT	C:¥Program Files¥PCL 1.13.1¥3rdParty¥Eigen¥eigen3¥Eigen
FLANN_ROOT	C:¥Program Files¥PCL 1.13.1¥3rdParty¥FLANN
path	C:¥Program Files¥PCL 1.13.1¥bin C:¥Program Files¥OpenNI2¥Redist C:¥Program Files¥PCL 1.13.1¥3rdParty¥VTK¥bin

④　Eigen

Eigen は, 線形代数ライブラリである. 本プロジェクトでは MeasurementSystem RTC で使用している．Point Cloud Library の All-in-one-installer を使ってインストールできるが, これだと Eigen の CMake ファイルが正しくインストールできない．そのため，別の方法でインストールする必要がある．ここでは，chocolatey を使ってインストールした.

まず，chocolatey をインストールするために PowerShell を管理者で起動し，表 11 の Installing Chocolatey で Chocolatey をインストールするコマンドを入手する．入手したコマンドを PowerShell に入力し実行すると Chocolatey がインストールされる． [10]

続けて，PowerShell で表 12 のコマンドを実行し，Eigen をインストールする． [11]

その他に clapack を利用しているが，これは Analyses RTC に埋め込んでいる.

表 11　chocolatey に関する URL

ページ種別	URL
Installing Chocolatey	https://chocolatey.org/install

表 12　Eigen のインストールに使うコマンド

インストールソフト	コマンド
Eigen	> choco install eigen

3.2. 再利用を検討した RTC の概要

本プロジェクトでは UrgRTC や Top-URG RTC などのネット上に公開されている既存の RTC の利用を考えたが，使用デバイスの接続方法はイーサネット接続であり，対応している RTC のビルドを行うために古い開発環境を利用する必要があるため新規に EtheURG RTC を開発した．［12, 13, 14, 15, 8, 16, 17, 18］

本プロジェクトでは，EtheURG RTC を除くすべてのコンポーネントのデータポートにおいて，大きい点群データを数多く送受信する必要がある．そこで，複数の実装例のある pointcloud.idl の PointCloud 型（PointCloudTypes::PointCoud）を使用した． Point-cloud.idl は，Geoffrey Biggs (gbiggs) 氏の RT-Components for the Point Cloud Library に含まれているものである．［19］

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，EtheURG RTC, MeasurementSystem RTC, Registration RTC, Analyses RTC, PointCloud_Viewer RTC, PointCloud_Reader RTC, FPS RTC, WallDTC RTC, Contour RTC, MapViewer RTC の開発を行った．これらの RTC は表 13 の URL 先からダウンロードすることができ， Mapping-by-3D-CaveSurvey-main.zip ファイルに格納されている．

表 13　本プロジェクトの GitHub リポジトリの URL

ページ種別	URL
GitHub	https://github.com/yukimeat1999/Mapping-by-3D-CaveSurvey.git

① EtheURG RTC

EtheURG RTC 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． [3]

Configuration ポートの connect_flag で接続方法を選択することで，シリアル接続とイーサネット接続いずれかの接続方法で LiDAR と接続することができる．シリアル接続の場合は serial_port_name と serial_baud_rate を設定し，イーサネット接続の場合は ethe_IP_add と ethe_port を設定する必要がある．シリアル接続やイーサネット接続時に必要な COM 番号の調べ方や IP アドレスの設定方法などは表 3 の HP を参考にしてもらいたい． [2]

次の図 2 および表 14 に EtheURG RTC の仕様を示す．

図 2 [System Editor]上に表示される EtheURG RTC

表 14 EtheURG RTC 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	connect_flag	string	Ethe	(Serial, Ethe)	radio
	serial_port_name	string	¥¥.¥COM1	-	text
	serial_baud_rate	int	115200	-	text
	ethe_IP_add	string	192.168.0.10	-	text
	ethe_port	int	10940	-	text
	geometry_x	double	0	-	text
	geometry_y	double	0	-	text
	geometry_z	double	0	-	text
	geometry_roll	double	0	-	text
	geometry_pitch	double	0	-	text
	geometry_yaw	double	0	-	text

② MeasurementSystem RTC

MeasurementSystem RTC は, InPort の range から受け取った各ステップにおける距離データをもとに点群データを作成し，サーボモータである Dynamixel を制御することで LiDAR を 180 度回転させ，3 次元点群データを構築する．作成された点群データを .ply ファイルとして保存し，作成された点群データを OutPort の new_PointCloud から出力する．

Configuration ポートの DEVICE_NAME と BAUDRATE に使用するサーボモータの COM 番号とボードレートを調べて設定を行う. encoder_resolution_ はサーボモータの分解能の総ステップ数を設定する．

計測が一度完了すると，"[INFO] Measurement is completed, please Deactivate." のメッセージとともに警告音が周期的に発せられるため，指示通りに Deactivate 化を行う．この後に Activate 化すると，新しい計測が開始される．

次の図 3 および表 15 に MeasurementSystem RTC の仕様を示す．

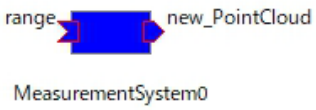

図 3 [System Editor]上に表示される MeasurementSystem RTC

表 15 MeasurementSystem RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	range	RangeData	-	-
OutPort	new_PointCloud	PointCloud	-	-
Configuration	DEVICE_NAME	string	¥¥.¥COM1	text
	BAUDRATE	int	57600	text
	encoder_resolution_	int	4096	text

③ Registration RTC

Registration RTC は，MeasurementSystem RTC から受け取った複数回計測した点群データをもとに，データの特徴量を抽出することで点群データの位置合わせを行い，点群データの結合を行う．

InPort の new_PointCloud は，新しく計測された点群データを OutPort から受け取っている．

OutPort の merge_PointCloud は, Registration RTC の InPort の new_PointCloud で受け取った各点群データを結合させることで, 生成した点群データを出力する. Localization は，この Registration RTC で推定した，最後に計測した点群データの原点を自己位置として xyz 座標を出力する．

次の図 4 および表 16 に Registration RTC の仕様を示す．

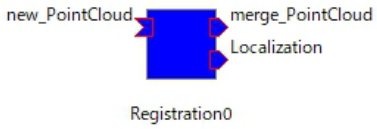

図 4 [System Editor]上に表示される Registration RTC

表 16 Registration RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	new_PointCloud	PointCloud	-	-
OutPort	merge_PointCloud	PointCloud	-	-
	Localization	TimedPose3D	-	-

④ Analyses RTC

Analyses RTC は，InPort の merge_PointCloud から受け取った点群データを用いて GNG による解析処理を行う．この処理結果の GNG のクラスタを OutPort の analyses_Cluster から出力する．ここでは，洞壁の表面粗さを認識し表示する．

次の図 5 および表 17 に Analyses RTC の仕様を示す．

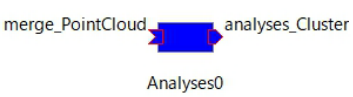

図 5 [System Editor]上に表示される Analyses RTC

表 17　Analyses RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	merge_PointCloud	PointCloud	-	-
OutPort	analyses_Cluster	ClusterData	-	-

⑤　PointCloud_Viewer RTC

PointCloud_Viewer RTC は，InPort の new_PointCloud，merge_PointCloud から受け取った点群データと，analyses_Cluster から受け取ったクラスタ，Localization から受け取った位置情報を各種ウィンドウで表示させる．merge_PointCloud を表示させるときは Localization から受け取った位置座標に赤い球として表示させる．

Configuration ポートの DataLoadOption で点群データの表示方法を選択できる．表 18 の InPort の上から順番に点群データを表示する場合は ”One_at_a_Time” を，3 つ同時に表示させる場合は ”SameTime” を，ファイルから読み込んで一つの点群データを表示させる場合は ”File” を選択する．ファイルから読み込んで点群データを表示する場合は，”FILE_NAME” に .ply 形式のファイル名を指定し，コンポーネントの起動用 .exe ファイルと同じフォルダー内に表示させるファイルを保存する．

次の図 6 および表 18 に PointCloud_Viewer RTC の仕様を示す．

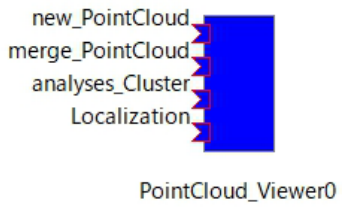

図 6　[System Editor]上に表示される PointCloud_ViewerRTC

表 18　PointCloud_Viewer RTC の仕様

ポート種別	ポート名	型	デフォルト値	制約	Widget
InPort	new_PointCloud	PointCloud	-	-	-
	merge_PointCloud	PointCloud	-	-	-
	analyses_Cluster	ClusterData	-	-	-
	Localization	TimedPose3D	-	-	-
Configu-ration	DataLoad Option	string	SameTime	(One_at_a_Time, SameTime, File)	radio
	FILE_NAME	string	Your_PointCloud_File.ply	-	text

⑥　PointCloud_Reader RTC

PointCloud_Reader RTC は，Configuration の FILE_NAME をもとに，PointCloud_Reader RTC の .exe ファイルがあるフォルダー内から FILE_NAME を持つ .ply 形式の点群データを読み込む. 読み込んだ点群データを OutPort の File_PointCloud から出力する.

次の図 3 および表 15 に PointCloud_Reader RTC の仕様を示す.

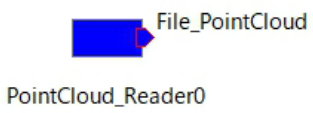

図 7　[System Editor]上に表示される PointCloud_Reader RTC

表 19　PointCloud_Reader RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
OutPort	File_PointCloud	PointCloud	-	-
Configuration	FILE_NAME	string	Your_PointCloud_File.ply	text

⑦ FPS RTC

FPS RTC は，InPort の PCD から受け取った点群データをもとに，Farthest Point Sampling (FPS) を行うことで点群の粗密に関係なく均等にダウンサンプリングを行う．作成された点群データを OutPort の DownPCD から出力する．

Configuration ポートの FPS_Max でダウンサンプリング後の点群の点数を指定することが出来る．処理には時間がかかるため，コンソール画面にダウンサンプリングの進捗率を可視化している．

次の図 3 および表 15 に FPS RTC の仕様を示す．

図 8 [System Editor]上に表示される FPS RTC

表 20 FPS RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	PCD	PointCloud	-	-
OutPort	DownPCD	PointCloud	-	-
Configuration	FPS_Max	int	10000	text

⑧ WallDTC RTC

WallDTC RTC は，InPort の PCD から点群データを受け取る．受け取った点群データをもとに，鉛直（z 軸）方向上から見た洞壁をクラスタとして生成し，作成されたクラスタを OutPort の PlanWall から出力する．

受取った点群データの z 軸の値を 0 にして，鉛直（z 軸）方向上から見た平坦な 2D データとする．この生成された平坦な点群データを用いて，α-shape（α 値を用いた concave hull）を適用し，点群の枠のクラスタを生成する．生成された枠の途切れている部分を接続することで，洞窟全体のクラスタを生成することが出来る．

Configuration ポートの MaxEdgeLength は，α-shape を行う際のノード間の最大距離を設定する．MaxConnDis は途切れているクラスタを接続する際のノード間の最大距離を設定する．Alpha は，α-shape を行う際の α 値を設定することが出来る．

次の図 3 および表 15 に WallDTC RTC の仕様を示す．

図 9　[System Editor]上に表示される WallDTC RTC

表 21　WallDTC RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	PCD	PointCloud	-	-
OutPort	PlanWall	ClusterData	-	-
Configuration	MaxEdgeLength	float	5	text
	MaxConnDis	float	10	text
	Alpha	float	0.5	text

⑨　Contour RTC

Contour RTC は，InPort の PCD から受け取った点群データをもとに洞床の高低差をエレベーションマップで表現し，作成された点群データを OutPort の Contour から出力する．

内部では，受け取った点群データをボクセルグリッドにかけて，同一 xy 座標のグリッドにおいて，一番低い箇所を抽出する．ここで，抽出したグリッドの中で z 軸の値が最も低い場所と最も高い場所の z 軸の値をもとに疑似カラーでエレベーションマップの高低差を表現する．各グリッドの座標とそれに対応する疑似カラーの RGB 情報を点群として出力する．

Configuration ポートの GridSize でボクセルグリッドのグリッドサイズを設定できる．

次の図 3 および表 15 に Contour RTC の仕様を示す．

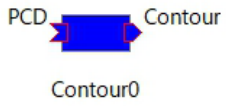

図 10　[System Editor]上に表示される Contour RTC

表 22　Contour RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	PCD	PointCloud	-	-
OutPort	Contour	PointCloud	-	-
Configuration	GridSize	float	0.5	text

⑩ MapViewer RTC

MapViewer RTC は，InPort の PlanWall からクラスタを受け取り，Contour からエレベーションマップのグリッド情報を点群として受け取る．これらを統合して可視化する．

Configuration ポートの DataLoadOption は（One_at_a_Time, SameTime）を選択することができ，いずれか片方のみを可視化するか，両方を統合して可視化するかを選択できる．ここで，SameTime を選択するとさらにビュアーを操作することができ，すでに各 InPort からデータを受け取った状態で操作できる．Switching では（EMap, EMap_PlanWall, PlanWall）を選択する事が出来る．ここで任意の要素を選択し，現在表示されているビュアーを閉じると，選択された要素をビュアーに可視化することが出来る．選択されている要素を表示させるビュアーのループから抜けるには，SameTimeViewerClosed で true を選択し，現在表示されているビュアーを閉じると，可視化処理が終了する．

次の図 3 および表 15 に MapViewer RTC の仕様を示す．

図 11 [System Editor]上に表示される MapViewer RTC

表 23 MapViewer RTC の仕様

ポート種別	ポート名	型	デフォルト値	Widget
InPort	PlanWall	ClusterData	-	-
	Contour	PointCloud		
Configuration	DataLoadOption	string	SameTime	radio
	SameTimeViewerClosed	string	false	radio
	Switching	string	EMap_PlanWall	radio

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． [20, 21]

① PC に LiDAR を cat 5e の LAN ケーブルで接続し，サーボモータを USB ポートに接続する．それぞれ電源等を繋ぎ，PC，LiDAR，サーボモータを起動する．

② 本プロジェクトページから Mapping-by-3D-CaveSurvey-main.zip ファイルをダウンロードし，図 12 のように解凍後の Mapping-by-3D-CaveSurvey-main 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．）

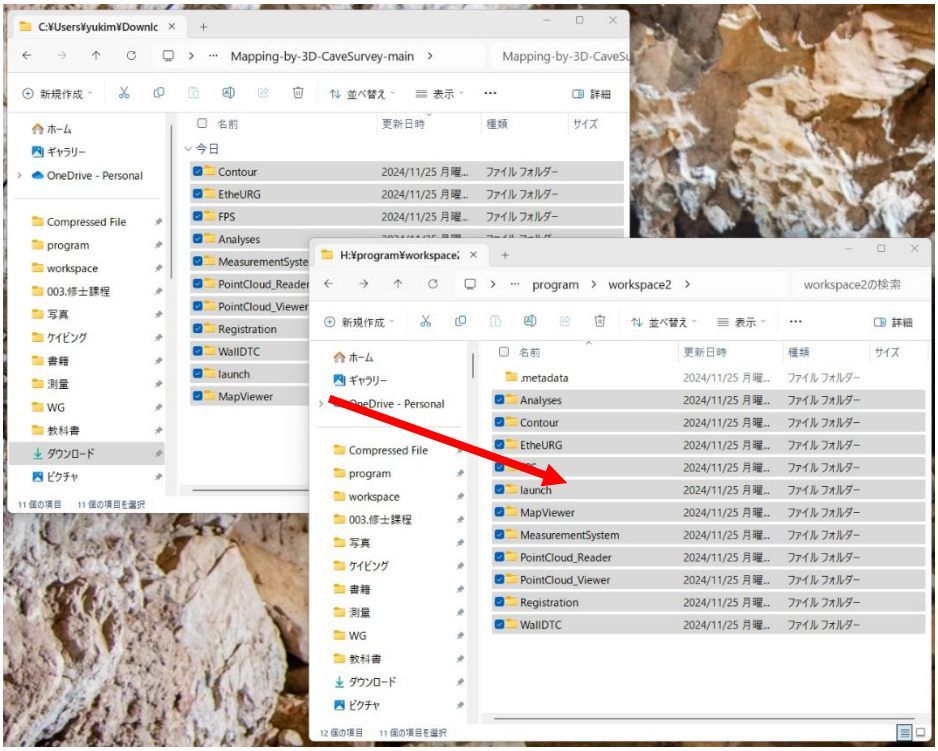

図 12 ダウンロードしたプロジェクトファイルを自身の workspace 内にドラッグ & ドロップ

③　図 13 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 14 のようなアプリを起動させる．

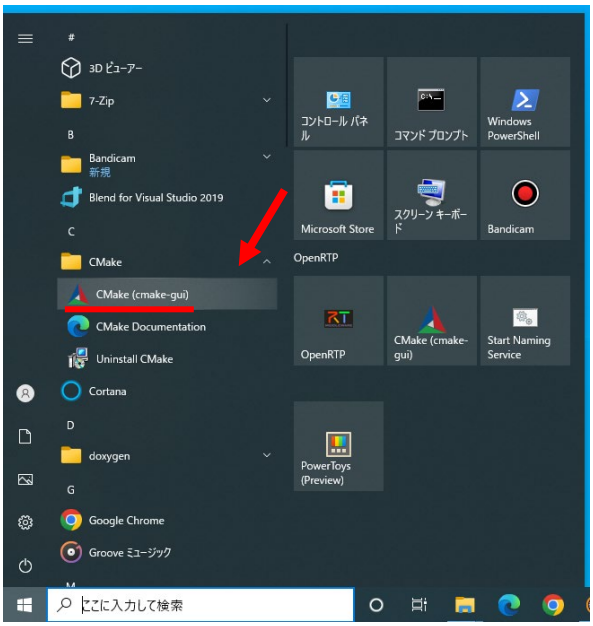

図 13　CMake の起動方法

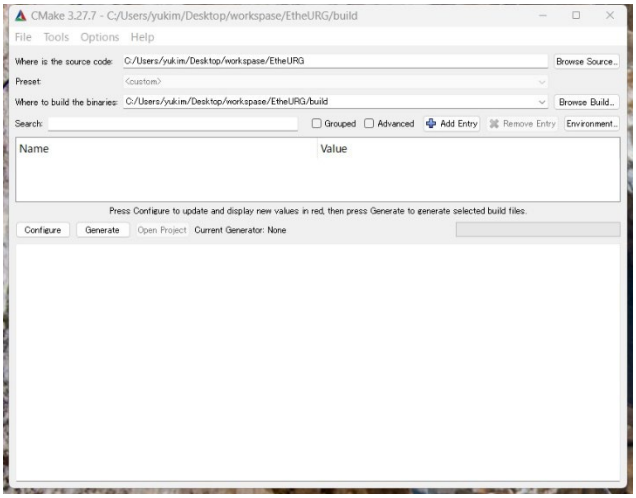

図 14　起動した CMake

ここからは EtheURG RTC を例に解説する．

④ 図 15 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/EtheURG”

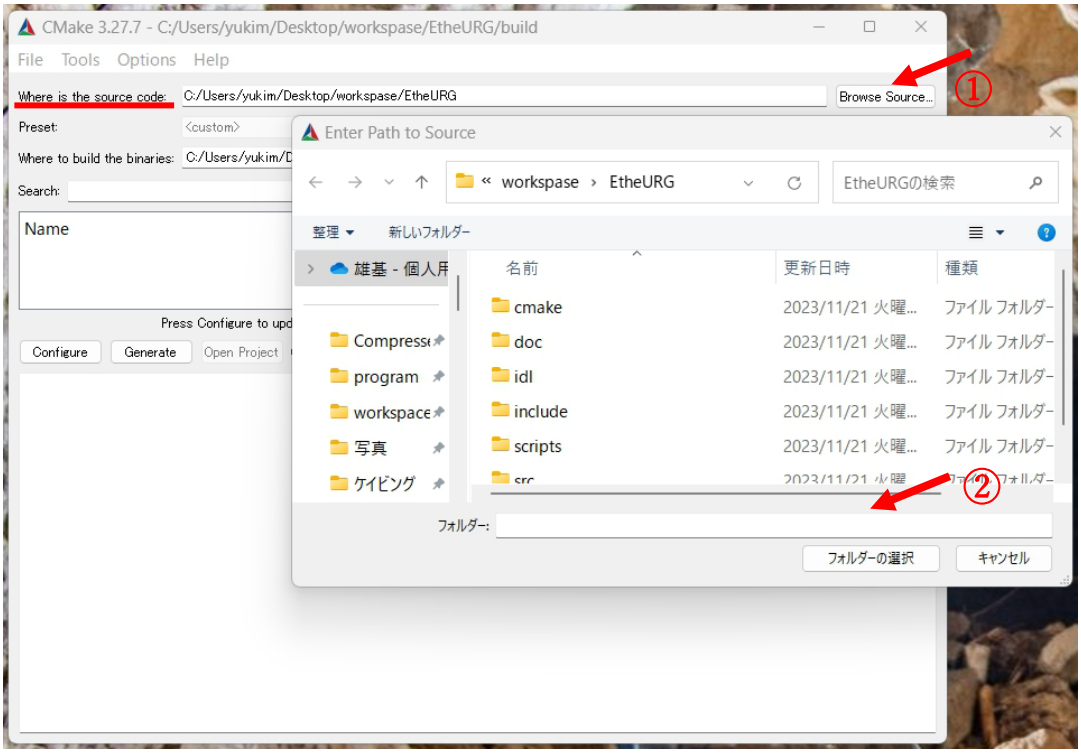

図 15　[Where is the source code]の項目の[Browse Source...]をクリックした状態

⑤　図 16 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/EtheURG/build”

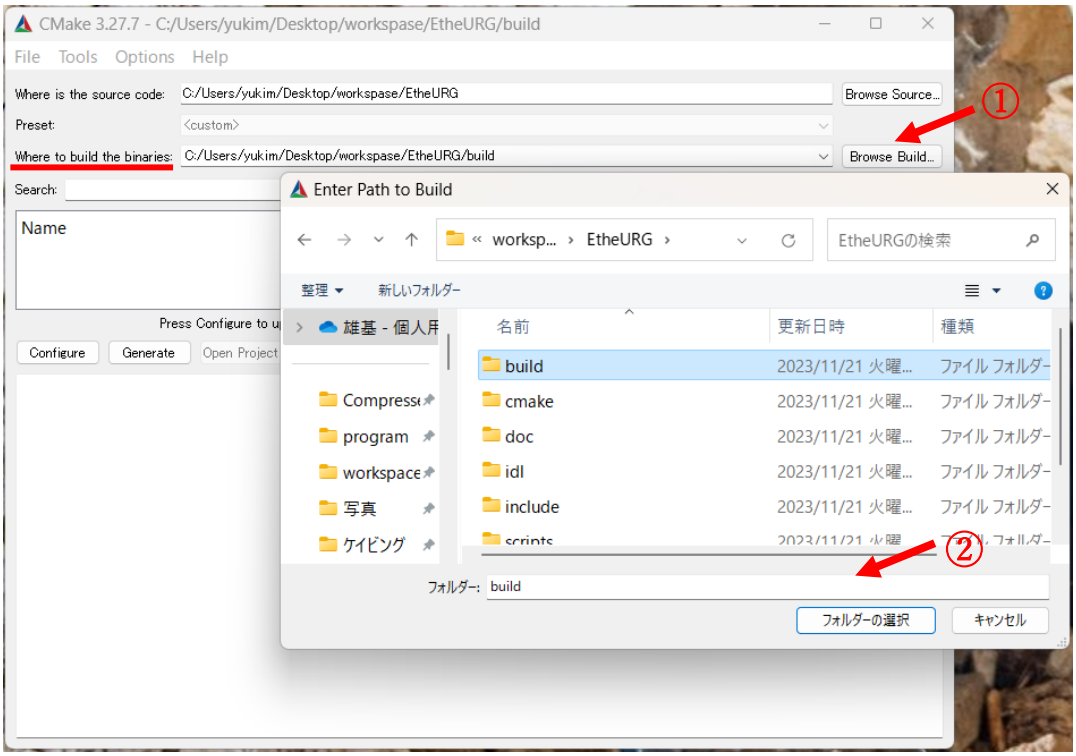

図 16　[Where to build the binaries]の項目の[Browse Source...]をクリックした状態

※　[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 17 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．

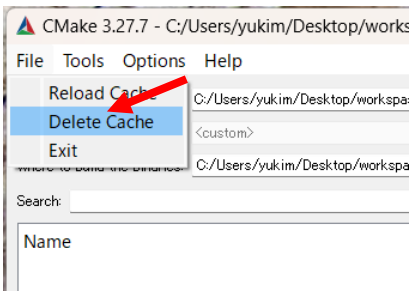

図 17　[Delete Cache]の場所

⑦　図 18 のように[CMake]ウィンドウ画面下方の[Configure]をクリックし，図 19 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 2022]を選択する．

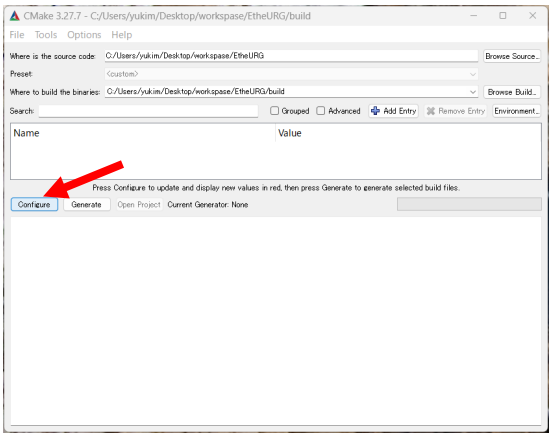

図 18　[Configure] ボタンの位置

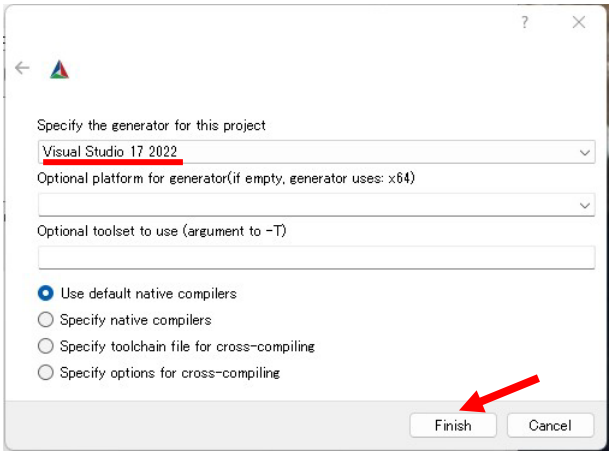

図 19　[Specify the generator for this project]の項目から，自身の PC で有効な[Visual Studio]を選択する

⑧　図 20 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．

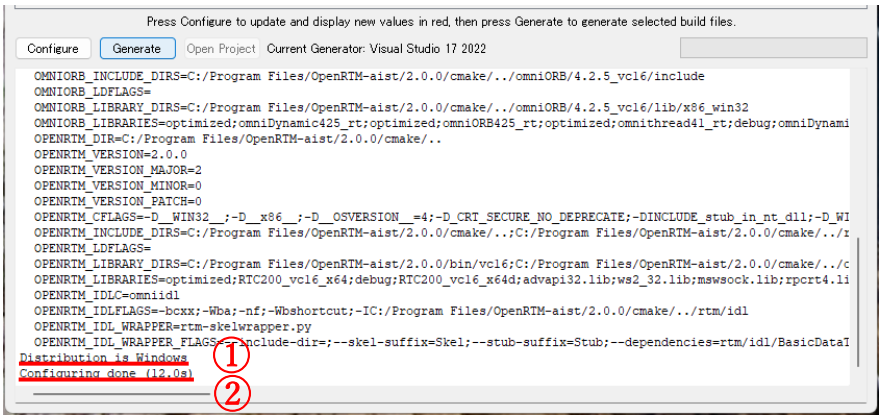

図 20　ウィンドウ下部のログに[Configuring done]と表示された状態

⑨　図 20 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 EtheURG を例に解説する． [20, 21]

① 図 21 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 2022]（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”

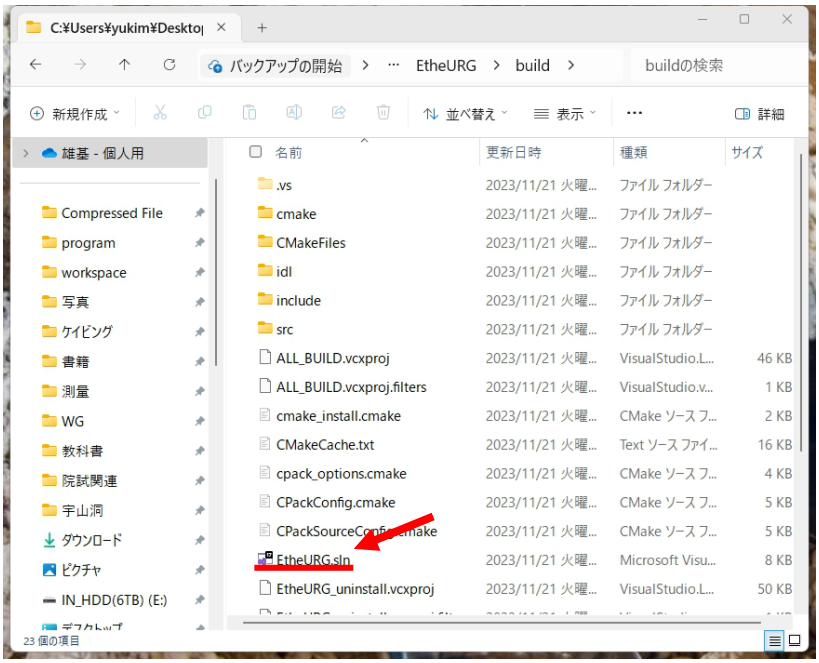

図 21　プロジェクトファイル内の[build]にある[プロジェクト名].sln ファイル

② 図 22 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 23 のように[メニューバー]の[ビルド]の中にある[ソリューションのビルド]をクリックする．

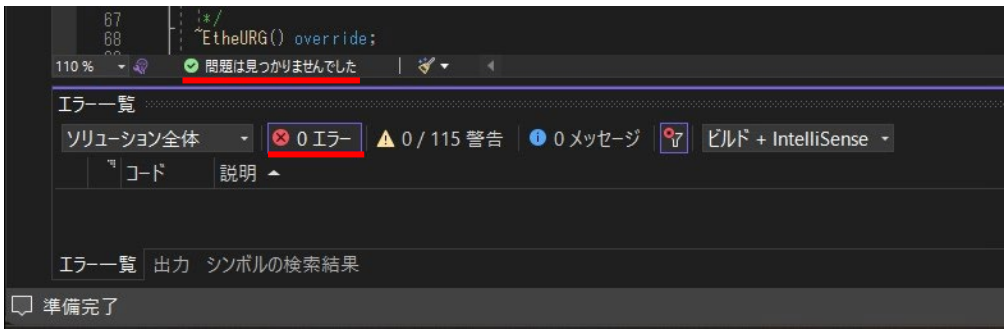

図 22　画面下部のエラー一覧にエラーが表示されていない状態

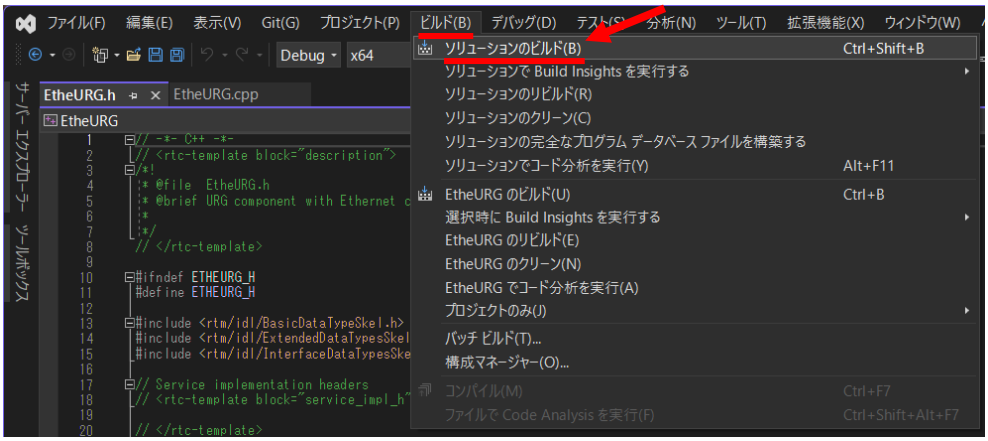

図 23　[メニューバー]の[ビルド]の中にある[ソリューションのビルド]

③　図 24 のように Visual Studio の[ステータスバー]に[ビルド正常終了]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．

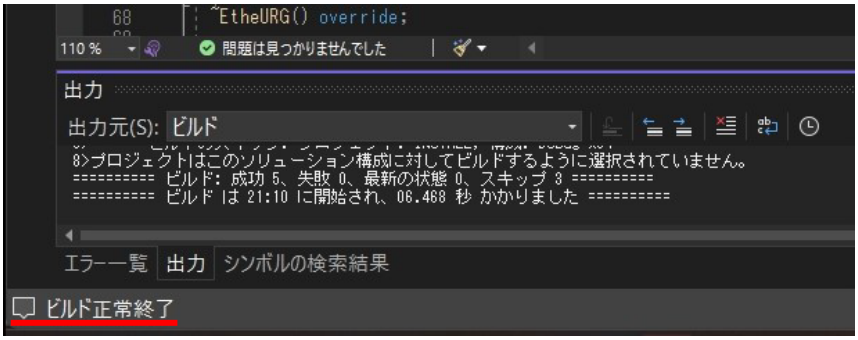

図 24　[ステータスバー]に[ビルド正常終了]と，表示された状態

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

ここからは 各種 RTC の起動と Configuration ポートの設定と注意すべき事項について解説する． [20, 21]

① 図 25 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．

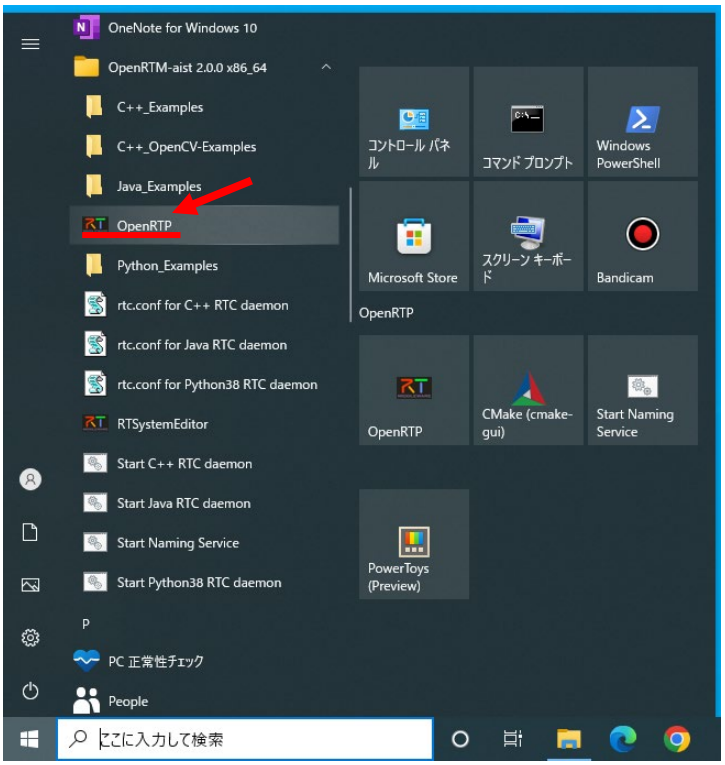

図 25　[OpenRTP]の起動方法

② 図 26 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．

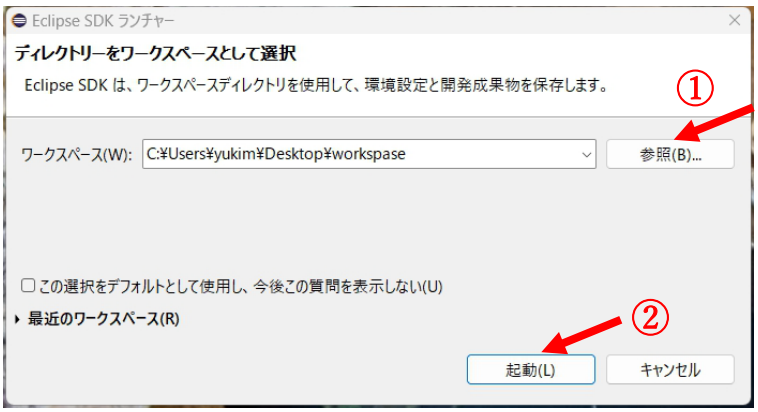

図 26　[Eclipse SDK ランチャー]ウィンドウの[ワークスペース]を選択する画面

③　図 27 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.

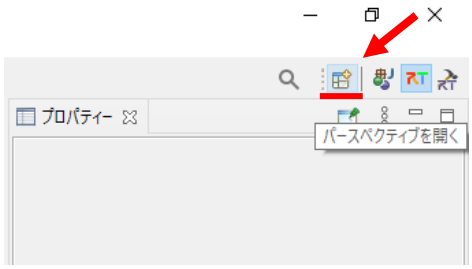

図 27　[パースペクティブを開く]の場所

④　図 28 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.

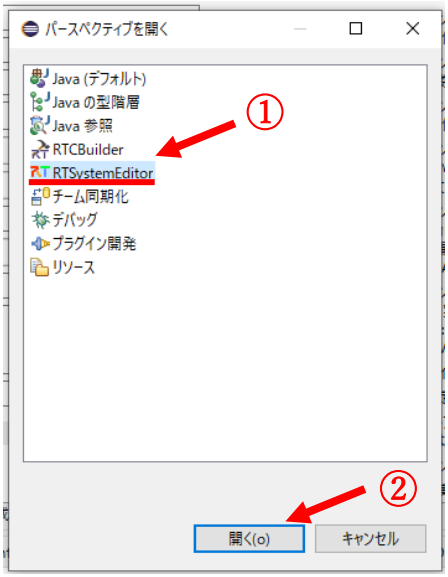

図 28　[パースペクティブを開く]ウィンドウの[RTSystemEditor]の場所

⑤　図 29 のように[ツールバー]の中から，[Open New System Editor]をクリック.

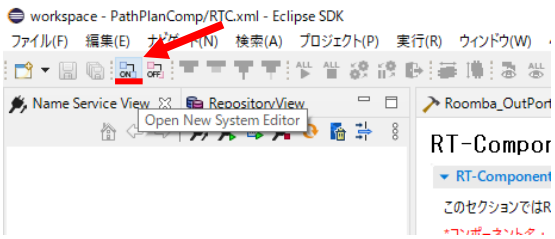

図 29　[ツールバー]内の[Open New System Editor]の場所

⑥　図 30 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.

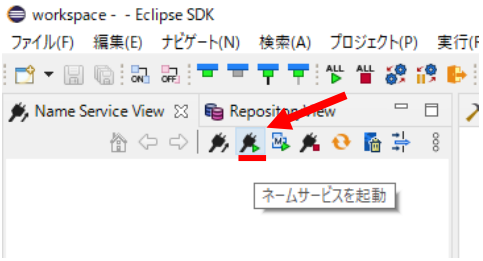

図 30　[Name Service View]の[ツールバー]内の[ネームサービスを起動]の場所

⑦　図 31 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.

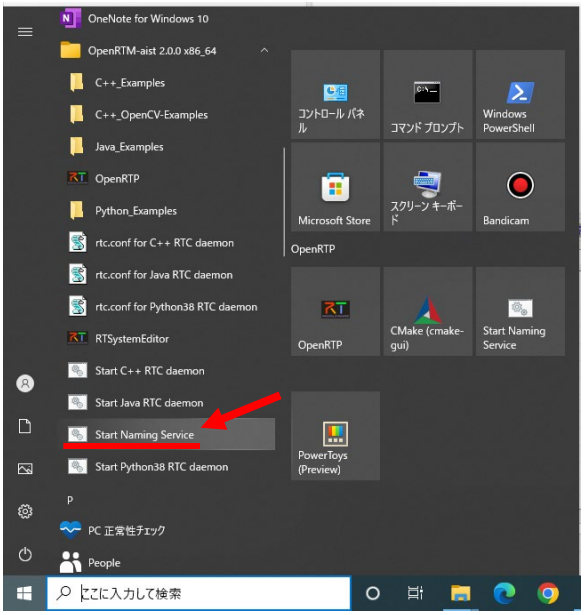

図 31　[Start Naming Service]の起動方法

⑧　図 32 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.

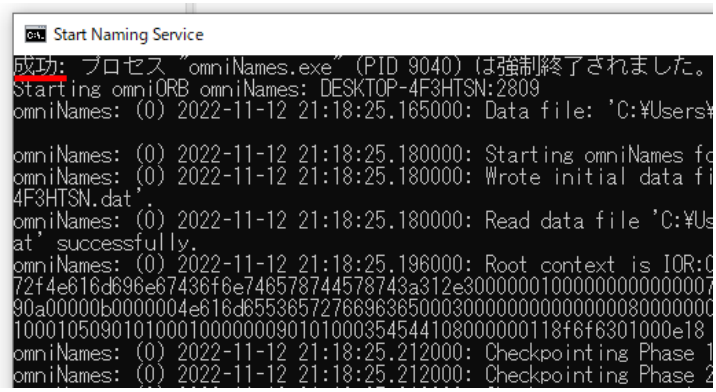

図 32　[Start Naming Service]のコンソール左上に[成功]と表示された状態

⑨　図 33 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"

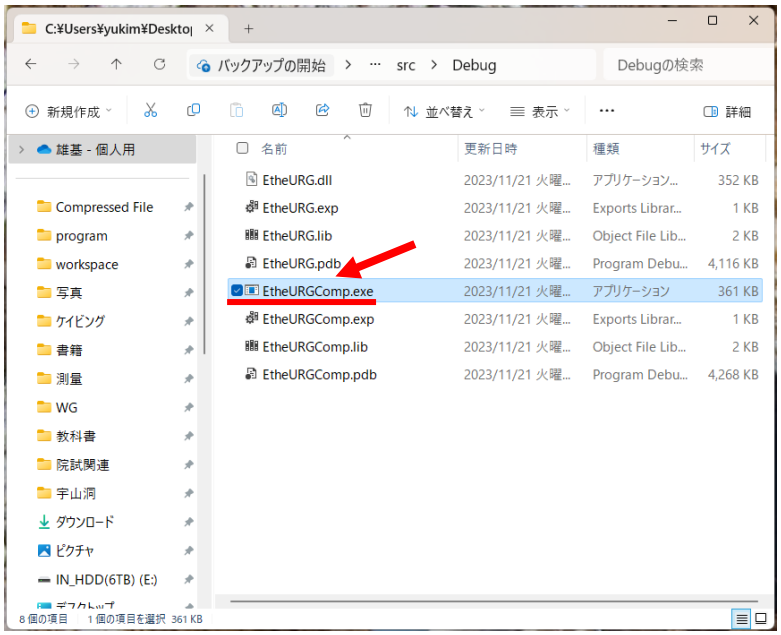

図 33　"[プロジェクト名]Comp.exe" ファイル

⑩ 図 34 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 35 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．

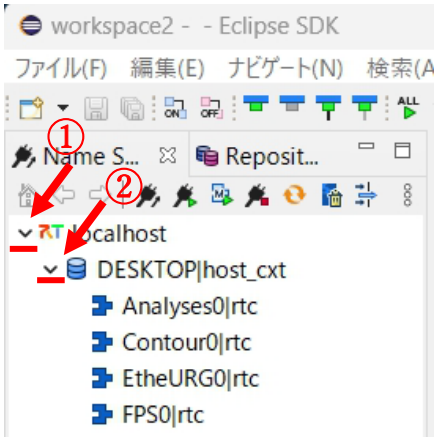

図 34 [Name Service View]内にある[localhost]と[[デスクトップ名]|host_cxt]を開いた状態

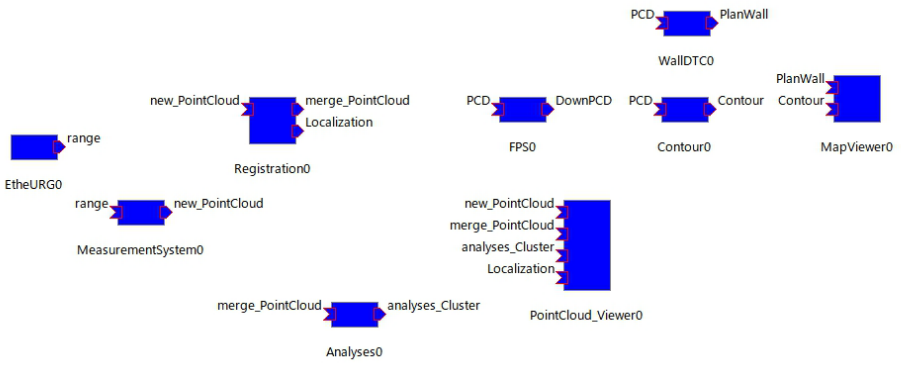

図 35 [System Editor]上に配置された[[プロジェクト名]0|rtc]の配置例

⑪ 図 36 から図 44 のように一部の RTC は，[System Editor]上で各 RTC を選択した上で，画面中央下部の[Configuration View]でにおいて Configuration ポートのデータを変更することができる．基本的に変更はしないが，「3.3 開発した RTC の概要」の各項目を参考に[EtheURG0]と[MeasurementSystem0]において，使用するデバイスに対する適切な接続方法にポート名等を変更する必要がある．

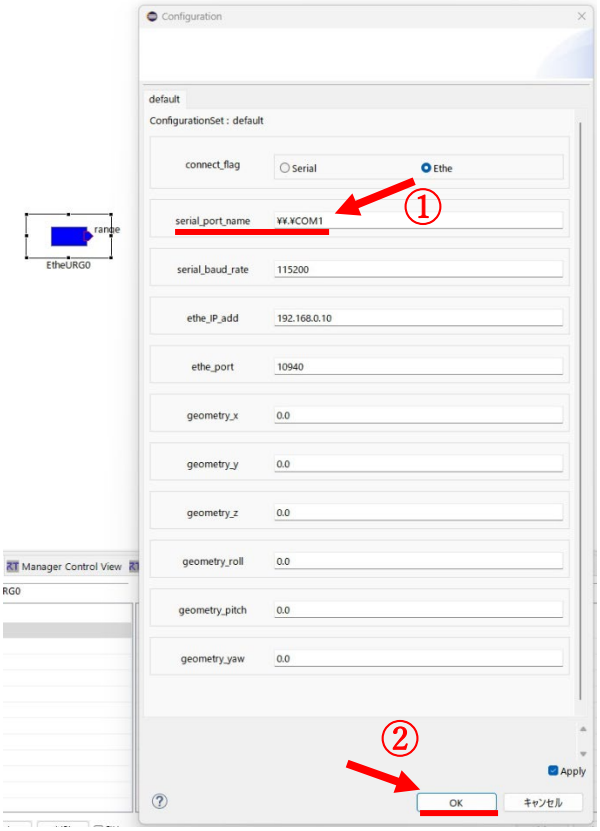

図 36 ”EtheURG0” の Configuration ポート変更画面

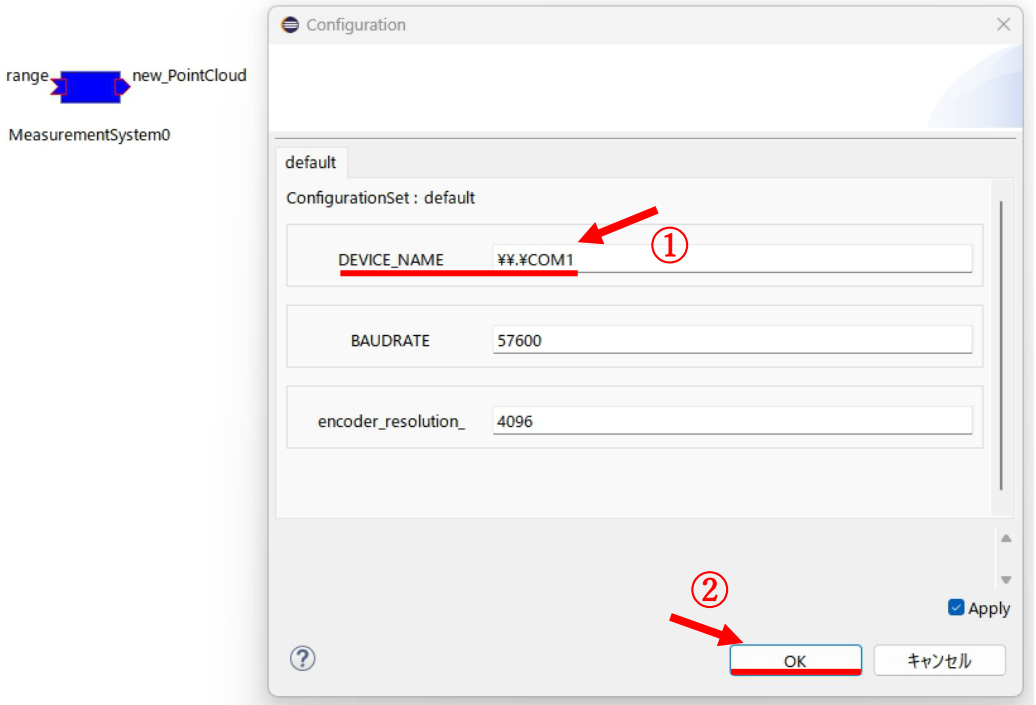

図 37 ”MeasurementSystem0” の Configuration ポート変更画面

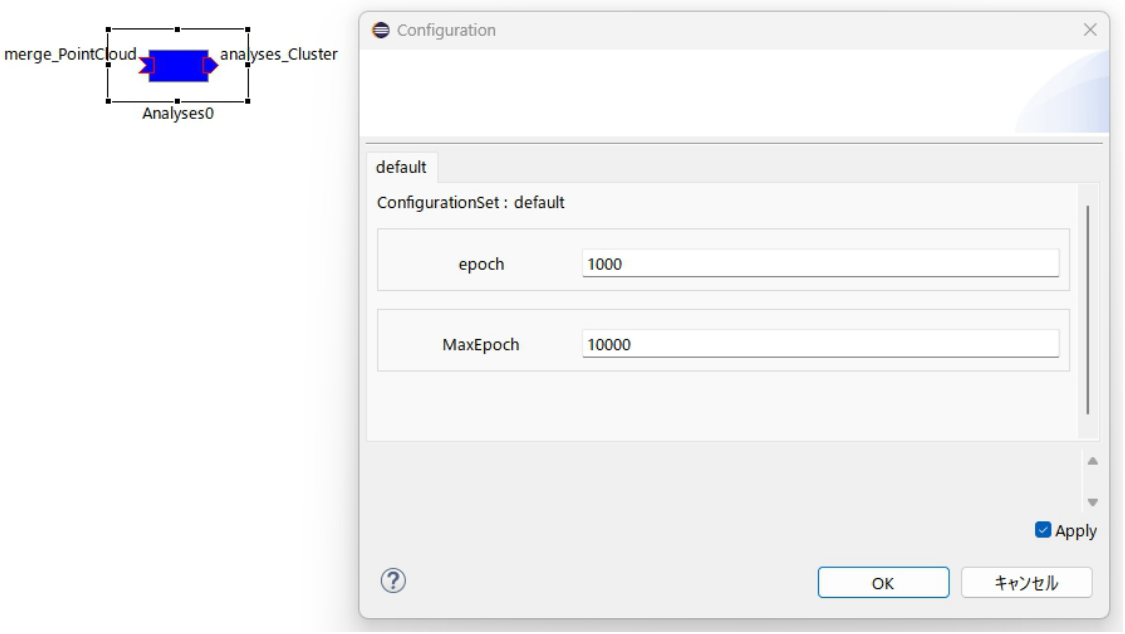

図 38 "Analyses0" の Configuration ポート変更画面

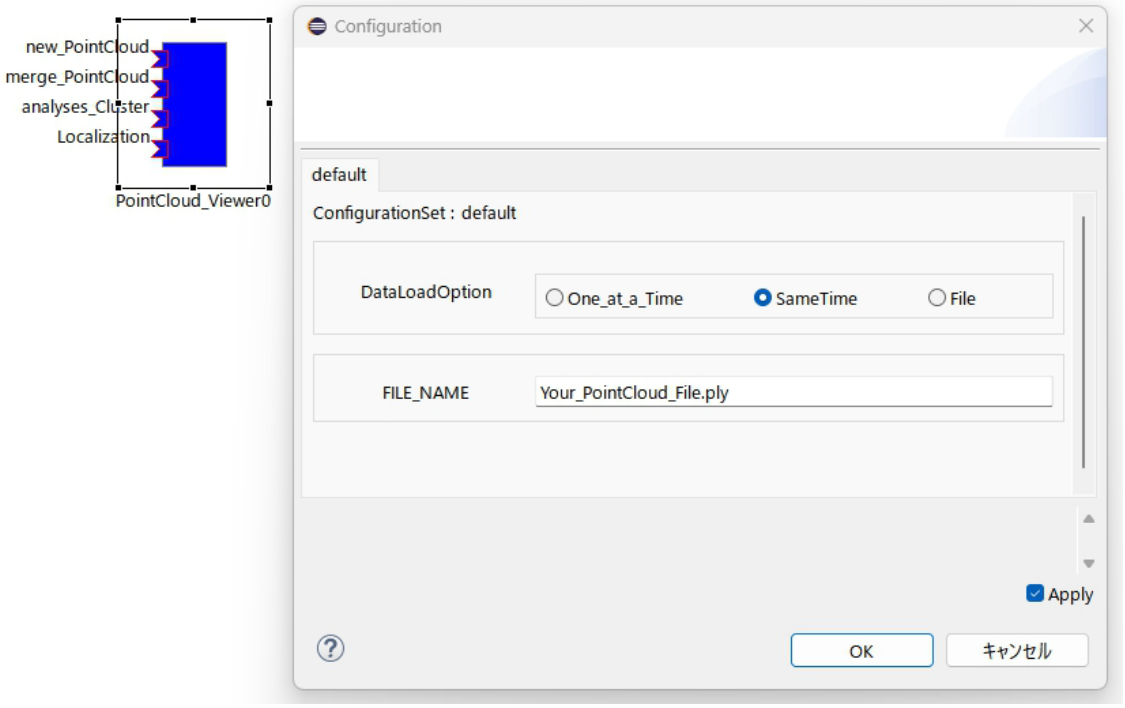

図 39 "PointCloud_Viewer0" の Configuration ポート変更画面

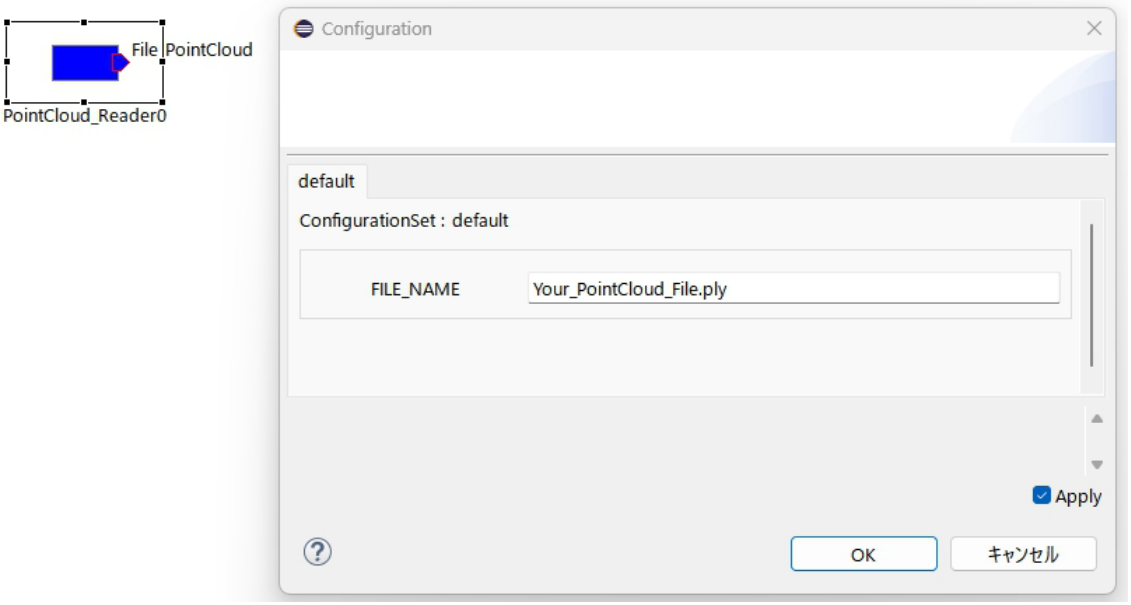

図 40 ”PointCloud_Reader0” の Configuration ポート変更画面

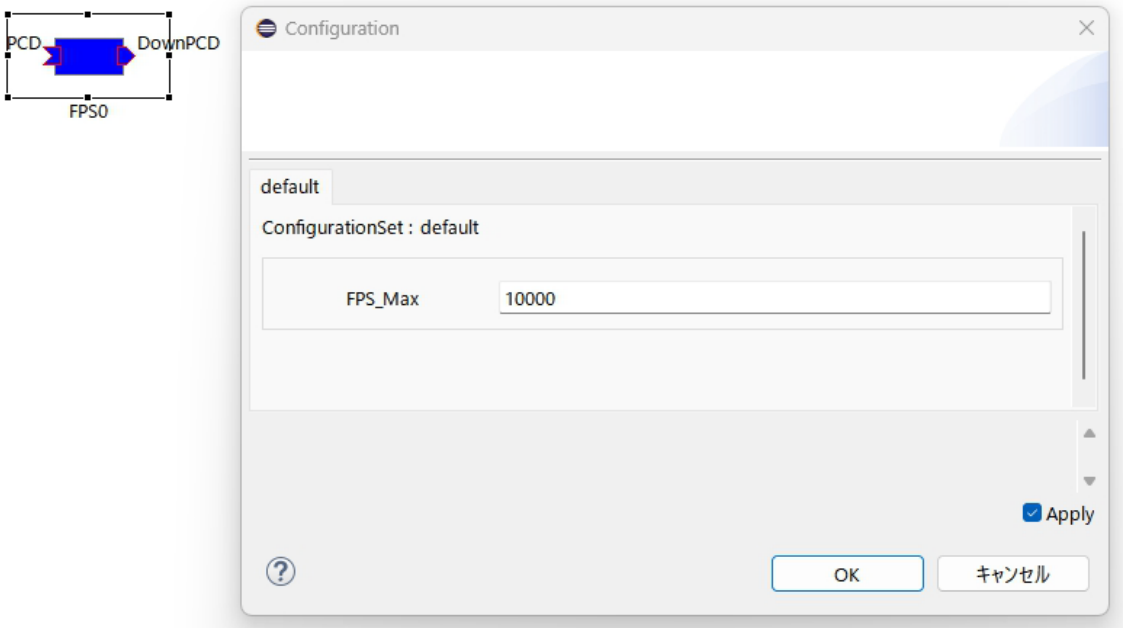

図 41 ”FPS0” の Configuration ポート変更画面

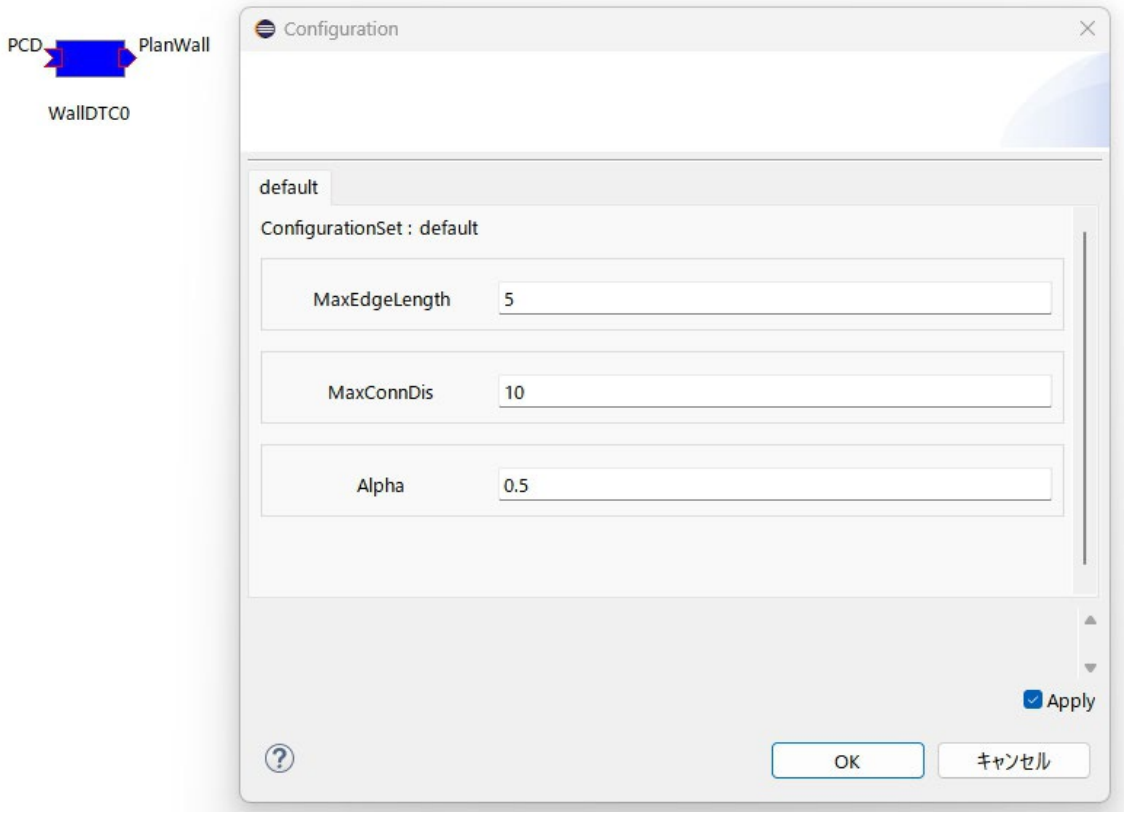

図 42 "WallDTC0" の Configuration ポート変更画面

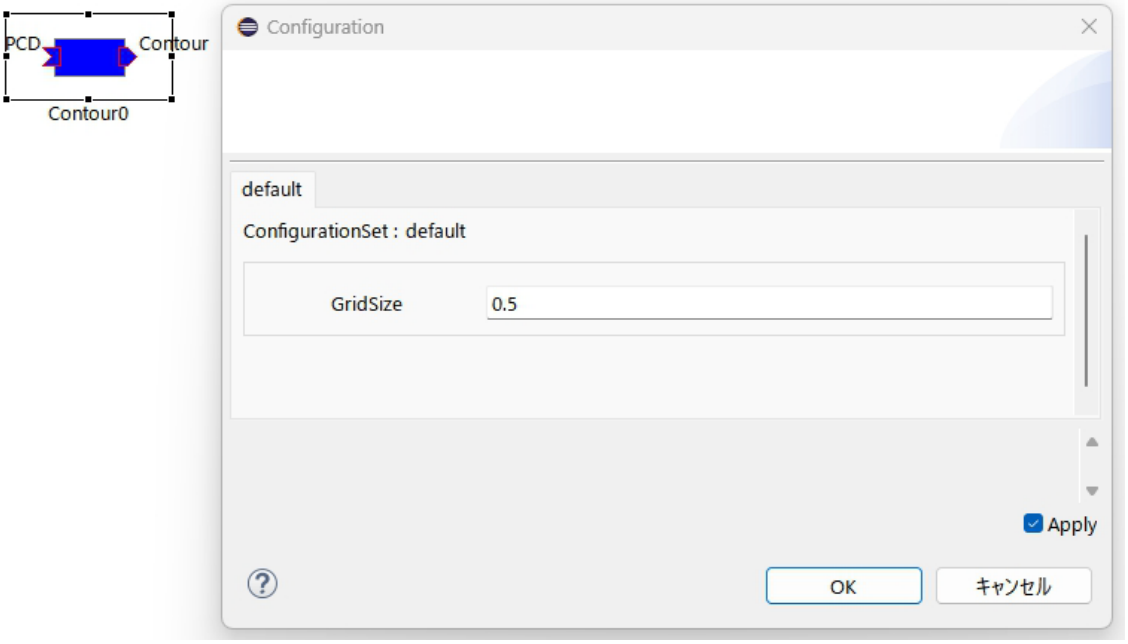

図 43 "Contour0" の Configuration ポート変更画面

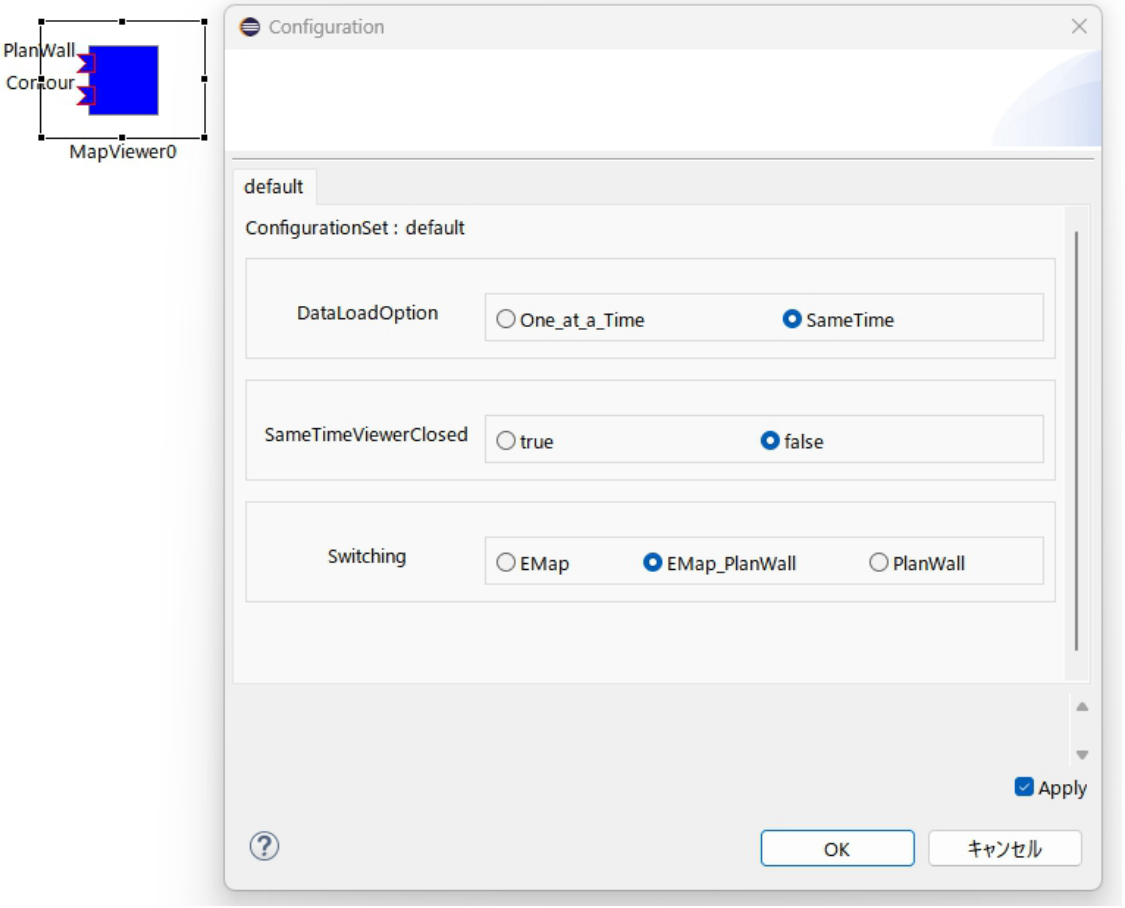

図 44 ”MapViewer0” の Configuration ポート変更画面

⑫　図 45 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．

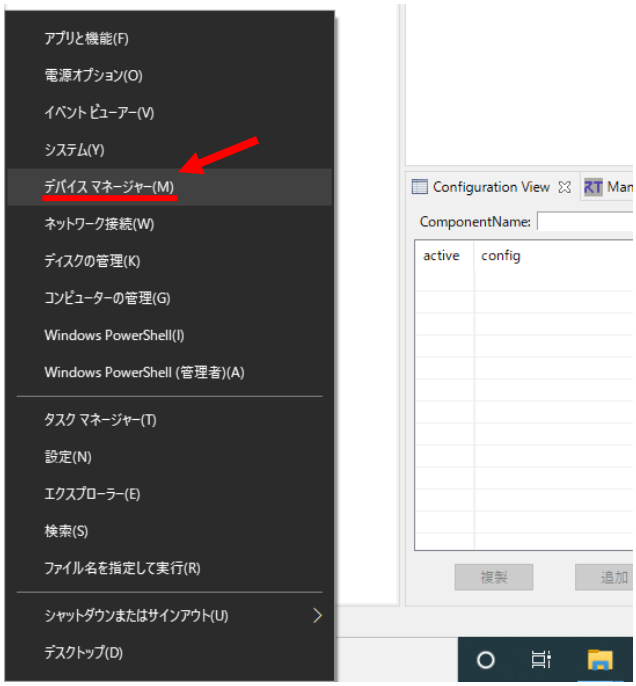

図 45　[スタート]ボタン上で[右クリック]し，表示させた[デバイスマネージャー]の項目

⑬　図 46 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，[EtheURG0]または[MeasurementSystem0]のポート名を設定する箇所に ”¥¥.¥COM7” を記述し，[適用]をクリックする．

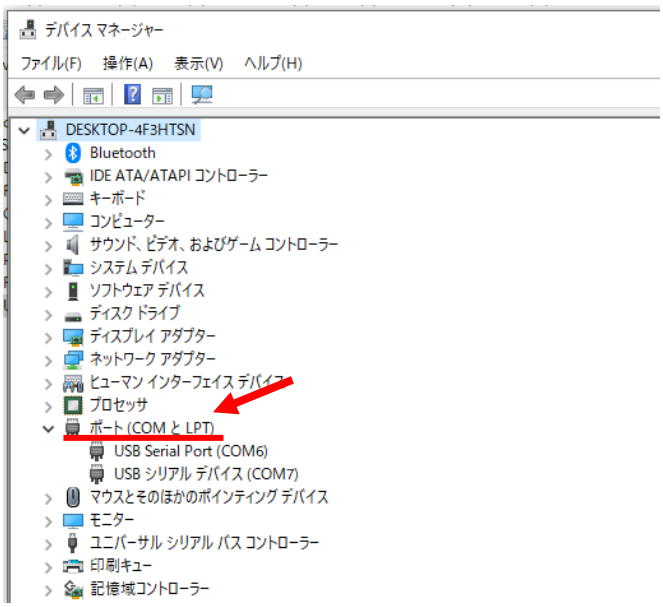

図 46　デバイスマネージャーウィンドウの[ポート(COM と LPT)]

⑭　図 47 および表 24 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．

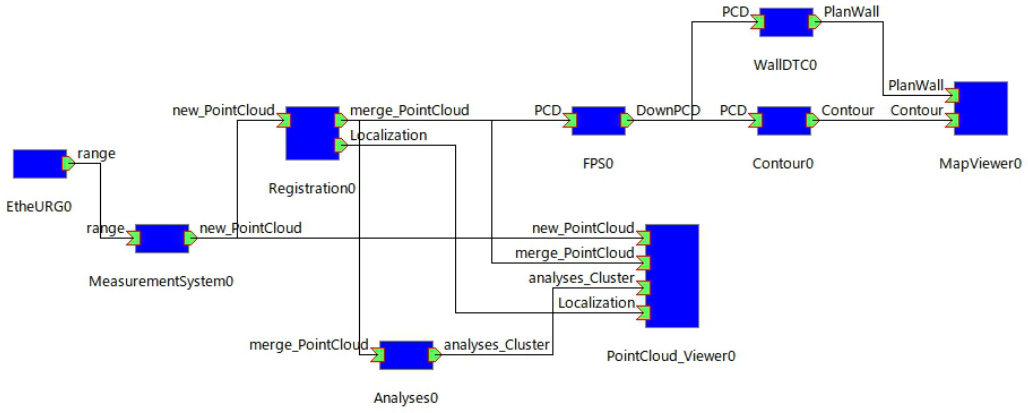

図 47　各ポートを実際に接続した状態

表 24　各種 RTC におけるポートの接続先

出力 RTC	OutPort 名	入力 RTC	InPort 名
EtheURG	range	MeasurementSystem	range
MeasurementSystem	new_PointCloud	Registration	new_PointCloud
		PointCloud_Viewer	new_PointCloud
Registration	merge_PointCloud	Analyses	merge_PointCloud
		PointCloud_Viewer	merge_PointCloud
		FPS	PCD
	Localization	PointCloud_Viewer	Localization
Analyses	analyses_ Cluster	PointCloud_Viewer	analyses_ Cluster
FPS	DownPCD	WallDTC	PCD
		Contour	PCD
WallDTC	PlanWall	MapViewer	PlanWall
Contour	Contour	MapViewer	Contour

4.4. RTC の Activate 化

ここからは RTC の Activate 化について解説する． [20, 21]

① 図 48 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 49 のようにすべての RTC が有効になる．

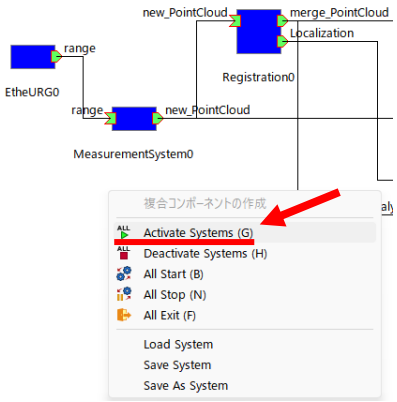

図 48 [System Editor]上の空白部分で[右クリック]し，表示させた[ALL Activate Systems]

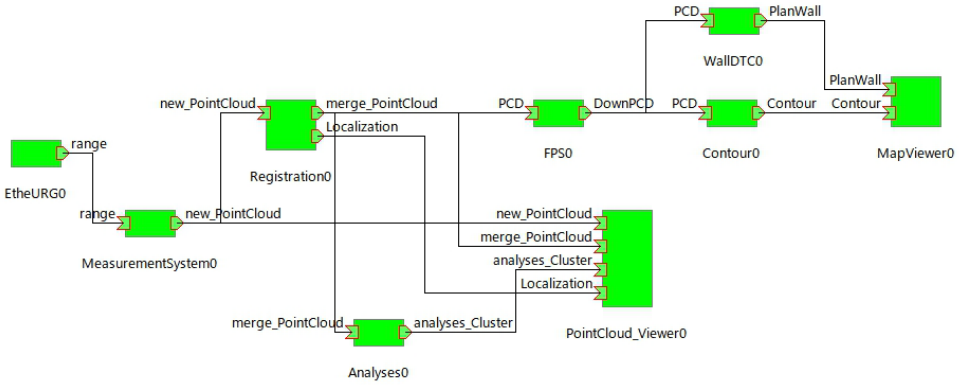

図 49 すべての RTC が有効となった状態

② 図 49 のように MeasurementSystem RTC を[Activate]することで LiDAR による計測が開始される．また，計測終了後に図 50 のようにコメントが表示されるため，この指示の通り MeasurementSystem RTC を[Deactivate]することで計測が 1 回終了する．再度計測する前に PointCloud_Viewer RTC を[Deactivate]することで，これらの RTC においてプログラムの実行が一度終了し，計測と可視化を再度実行する準備ができる．

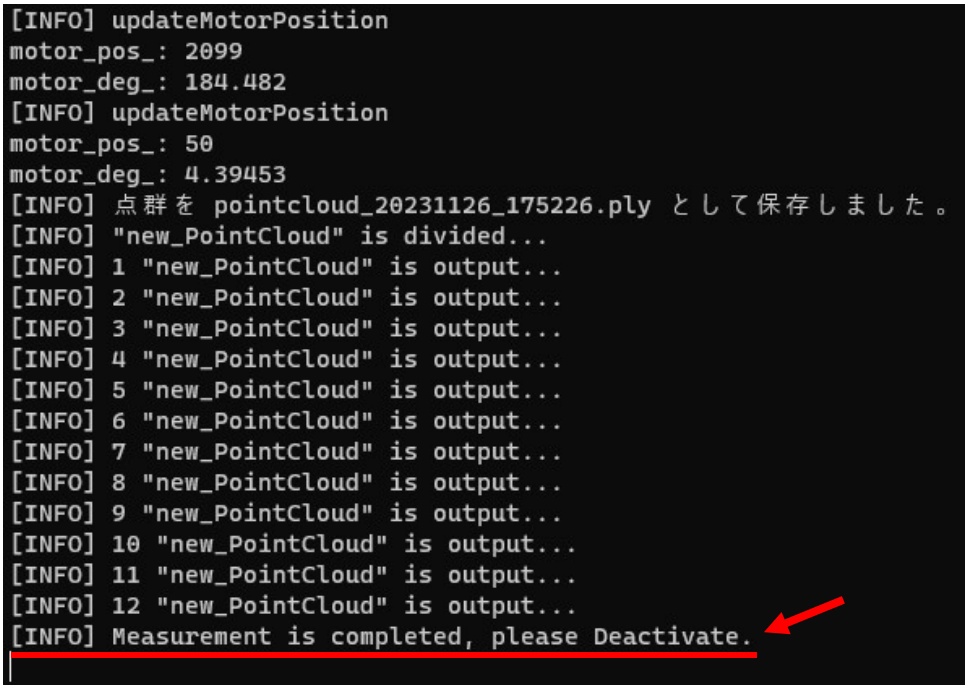

図 50　"MeasurementSystem0"のコンソールに表示される[Deactivate]化の指示

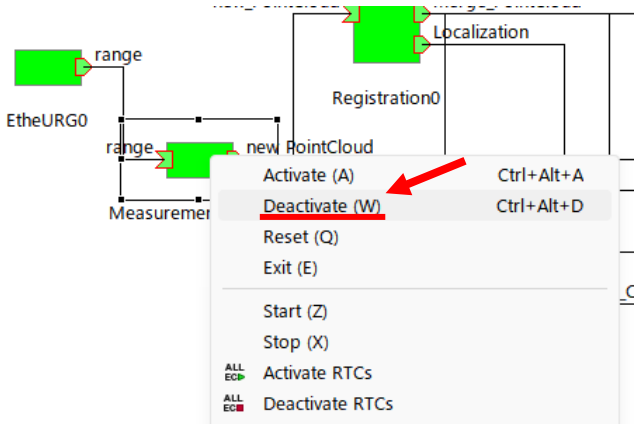

図 51　"MeasurementSystem0"の[Deactivate]

③　図 52 のように MeasurementSystem RTC を再度[Activate]することで LiDAR による 2 回目の計測が開始される．この様に②と③を繰り返すことで何度も計測することができる．また，MeasurementSystem RTC と PointCloud_Viewer RTC の RTC において[ALL Activate Systems]することで，各処理が新たに実行される．この時，PointCloud_Viewer RTC の Viewer ウィンドウを閉じている必要がある．Viewer ウィンドウを閉じていないと新しい点群データを PointCloud_Viewer が受け取れられないため，新しい点群データを Viewer ウィンドウに正しく表示できない．

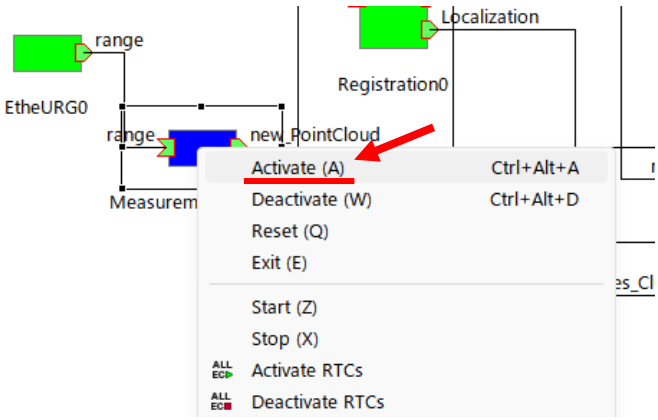

図 52　"MeasurementSystem0"の[Active]

④　図 53 のように[System Editor]上の空白部分で[右クリック]し，[All Exit]をクリックすると，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．この時，PointCloud_Viewer RTC のウィンドウを閉じている必要がある．

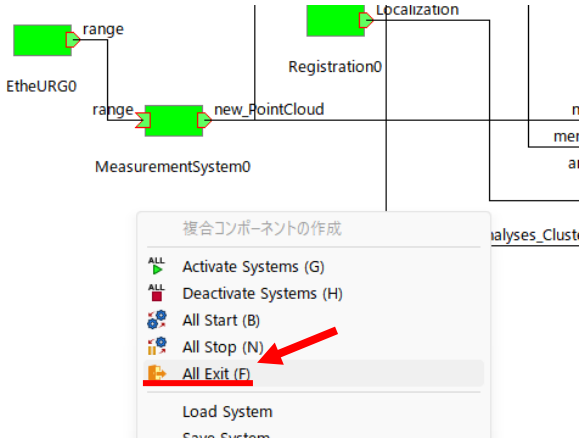

図 53　[System Editor]上の空白部分で[右クリック]し，表示させた[All Exit]

4.5. バッチファイルの利用

本プロジェクトでは各 RTC を起動するために launch ファイル内にバッチファイルを作成した．各バッチファイルにおいて[CaveScan_*.bat]はすべての RTC について処理を行い，[CaveScanOnly_*.bat]は EtheURG RTC と MeasurementSystem RTC のみの処理を行う．また，[*_Startup.bat]は各 RTC の起動とポートの接続を行う．[*_Exit.bat]は各 RTC を[Deactivate]し，各ポートの接続を切断し，各 RTC を終了する．

各バッチファイルを起動して処理がうまく動作しない場合は，OpenRTM の[System Editor]において，各 RTC のホスト名が PC のコンピュータ名(Windows のシステム設定にあるホスト名)と大文字と小文字が一致しないために発生するバグである．この場合は各バッチファイルの 22 行目のコメントアウトを外して[System Editor]上で確認できるホスト名を設定する．

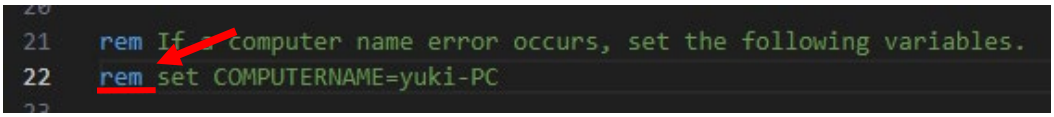

図 54　コンピュータ名の設定をコメントアウトしている状態

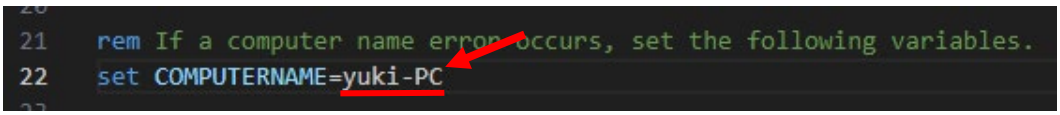

図 55　コンピュータ名の設定を有効にしている状態

5. 動作例

5.1. 実際の動作

① [EtheURG0]を［Activate］化し，他の RTC を［ALL Activate Systems］すると，MeasurementSystem RTC において計測が開始され，計測地点における三次元点群データが保存され，OutPort の new_PointCloud から出力する．このデータを受け取った PointCloud_Viewer RTC において，ウィンドウが表示され，三次元点群データが表示される．

② Registration RTC において，各計測地点における点群データを位置合わせし，統合した点群データを OutPort の merge_PointCloud から出力する．このデータを受け取った PointCloud_Viewer RTC において，ウィンドウが表示され，統合後の点群データが表示され，最後に計測した地点の位置座標が赤い球として表示される．

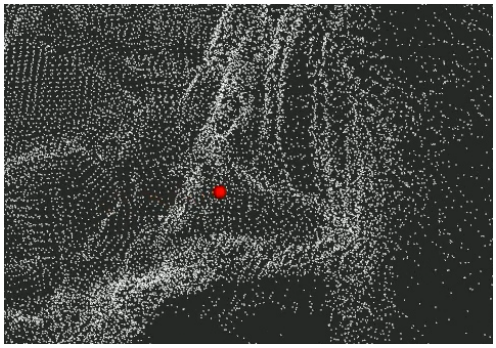

図 56　merge_PointCloud における Localization の表示例

③ Analyses RTC において，Registration RTC から受け取った InPort の merge_PointCloud をもとに GNG の処理を行い，その結果を OutPort の analyses_Cluster から出力する．このデータを受け取った PointCloud_Viewer RTC において，ウィンドウが表示され，クラスタが表示される．ここでは空間における表面の粗さを認識し，表面粗さが大きい部分が密に表示される． [1]

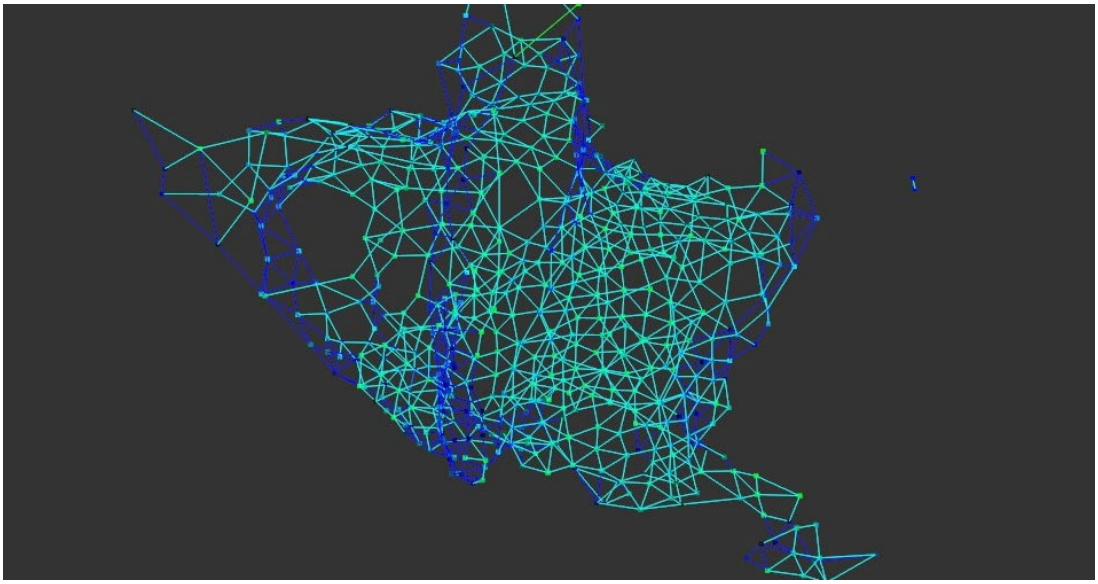

図 57　analyses_PointCloud の表示例

④　PointCloud_Viewer RTC の Configuration ポートの DataLoadOption で点群データの表示方法で ”One_at_a_Time” を選択すると, 上記①から③の様にウィンドウが一つずつ表示される．”SameTime” を選択すると，一つのウィンドウに 3 つ同時に表示される．図 58，図 59 は”SameTime” を選択した時のウィンドウである．一度目の計測時のみ図 58 のように new_PointCloud のみが表示される．

図 58　一度目の計測の時に表示される new_PointCloud

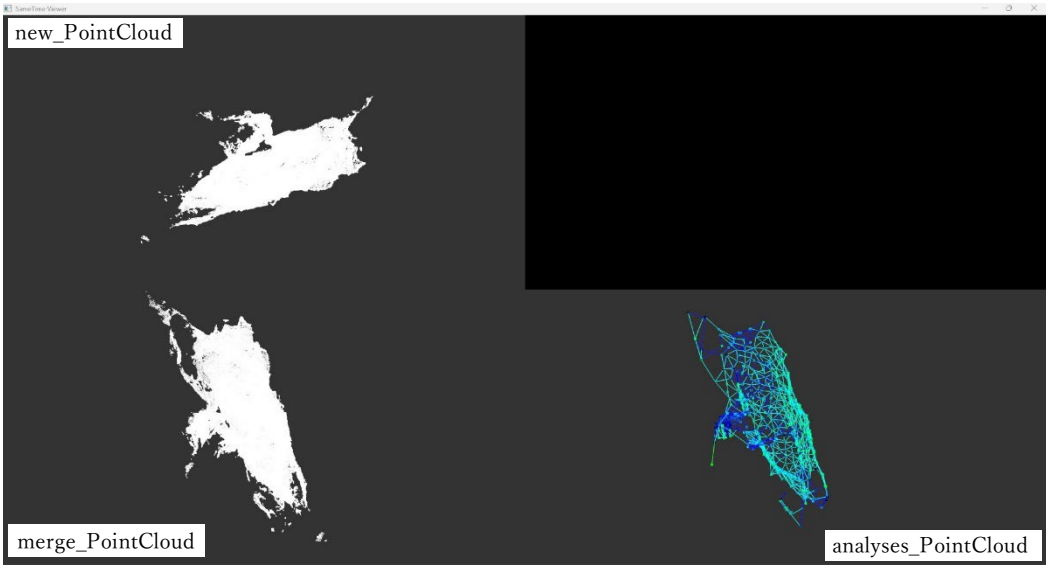

図 59　二度目以降の計測時に表示される各種点群データ

⑤ FPS RTC では点群の粗密度に関わらず満遍なく点をサンプリングすることで，ダウンサンプリングを行っている．処理に時間がかかるため，コンソールにプログレス表示がある．

⑥ MapViewer RTC で表示されるウィンドウでは，図 60 の様に WallDTC RTC と Contour RTC から受け取った洞壁のクラスタと洞床の高低差を表すエレベーションマップを可視化することができる．MapViewer RTC の Configuration ポートの DataLoadOption で点群データの表示方法で ”One_at_a_Time” を選択すると，WallDTC RTC と Contour RTC のデータを可視化するウィンドウが一つずつ表示される．”SameTime” を選択すると二種類のデータを統合して一つのウィンドウで可視化することができる．この時，Switching で選択した内容のデータを次のウィンドウに表示させることが出来る．現在開いているウィンドウを閉じると Configuration ポートを再読み込みする．SameTimeViewerClosed を ”true” にするとウィンドウを閉じても，次のウィンドウが表示されなくなり，処理が終了する．

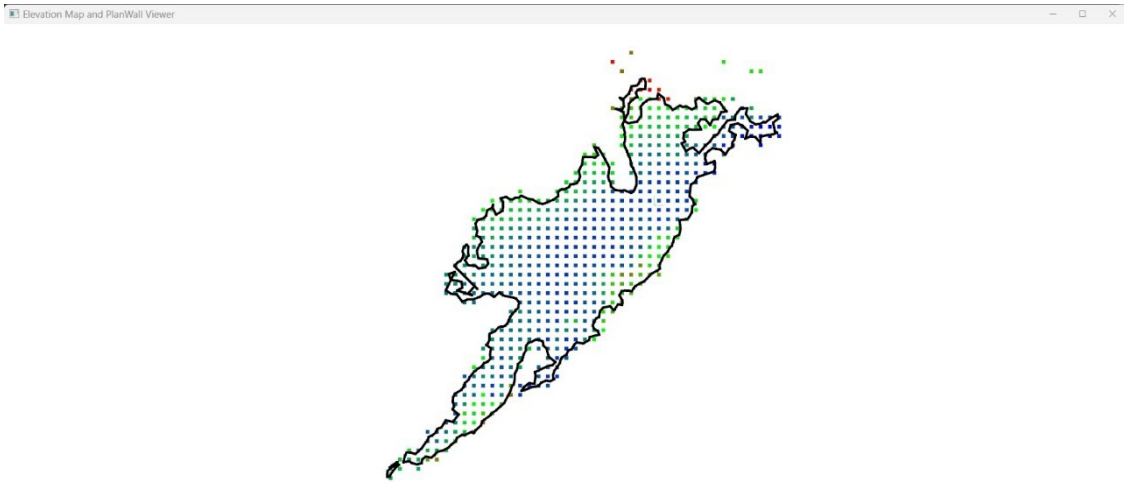

図 60　洞壁のクラスタとエレベーションマップを統合して表示させたウィンドウ

6. 参考文献

[1] A. W. H. M. K. O. T. M. M. M. Yuichiro Toda, “Growing Neural Gas with Different Topologies for 3D Space Perception,” *Applied Sciences,* 第 巻 Vol. 12, 第 No. 3, p. 1705, 2022.

[2] 北陽電機株式会社, “URG Network Wiki,” SourceForge, 25 8 2020. [オンライン]. Available: https://sourceforge.net/p/urgnetwork/wiki/top_jp/. [アクセス日: 25 11 2023].

[3] 北陽電機株式会社, “製品詳細 UTM-30LX-EW,” HOKUYO, 2020. [オンライン]. Available: https://www.hokuyo-aut.co.jp/search/single.php?serial=146. [アクセス日: 25 11 2023].

[4] ROBOTIS, “ DynamixelSDK, ” ROBOTIS, 2023. [オンライン]. Available: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/. [アクセス日: 25 11 2023].

[5] 株式会社ベストテクノロジー, “DYNAMIXEL basic tutorial,” 株式会社ベストテクノロジー, 7 9 2023. [オンライン]. Available: https://www.besttechnology.co.jp/modules/knowledge/?DYNAMIXEL%20basic%20tutorial. [アクセス日: 25 11 2023].

[6] honghyun79, “ROBOTIS-GIT/DynamixelSDK,” GitHub, 28 6 2023. [オンライン]. Available: https://github.com/ROBOTIS-GIT/DynamixelSDK.git. [アクセス日: 25 11 2023].

[7] kunaltyagi, “PointCloudLibrary/PCL,” GitHub, 10 5 2023. [オンライン]. Available: https://github.com/PointCloudLibrary/pcl/releases. [アクセス日: 25 11 2023].

[8] nakaoka, “PointCloud の IDL について,” CHOREONOID, 18 8 2018. [オンライン]. Available: https://discourse.choreonoid.org/t/pointcloud-idl/99. [アクセス日: 25 11 2023].

[9] 金子邦彦, “Windows で libPCL 1.9.1, Boost, Eigen, OpenNI 2.2, NITE2 のインストール,” 金子邦彦研究室, 2 2 2023. [オンライン]. Available: https://www.kkaneko.jp/db/win/libpcl.html. [アクセス日: 25 11 2023].

[10] ともすた, “Chocolatey で、まっさら Windows に一気にソフトをインストール,” ともすた, 4 6 2019. [オンライン]. Available: https://tomosta.jp/2019/06/chocolatey/. [アクセス日: 25 11 2023].

[11 lebarsfa, “ Eigen 3.4.0, ” chocolatey, 12 5 2022. [オンライン]. Available:

] https://community.chocolatey.org/packages/eigen. [アクセス日: 25 11 2023].

[12] gbiggs, “RTC:HokuyoAIST,” OpenRTM-aist, 27 6 2011. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/project/rtchokuyoaist. [アクセス日: 25 11 2023].

[13] ogasawara, “Top-URG RTC （北陽電機社製：UTM-30LX）,” OpenRTM-aist, 2 2 2012. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID131. [アクセス日: 25 11 2023].

[14] ysuga, “北陽電機 URG センサ RTC,” OpenRTM-aist, 30 11 2014. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/project/Hokuyo_URG_RTC_by_SSR. [アクセス日: 25 11 2023].

[15] 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）, “Classic-URG RTC （北陽電機社製：URG-04LX）,” 11 11 2022. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID130.

[16] G. Biggs, “Flexiport data communications library,” Flexiport v2.0.0 documentation, 2011. [オンライン]. Available: https://gbiggs.github.io/flexiport/. [アクセス日: 25 11 2023].

[17] G. Biggs, “HokuyoAIST range sensor driver,” HokuyoAIST 3.0.0 documentation, 2011. [オンライン]. Available: https://gbiggs.github.io/hokuyoaist/. [アクセス日: 25 11 2023].

[18] G. Biggs, “RTC:HokuyoAIST - 日本語,” RTCHokuyoAIST v3.0.0 documentation, 2011. [オンライン]. Available: https://gbiggs.github.io/rtchokuyoaist/index_j.html. [アクセス日: 25 11 2023].

[19] G. Biggs, “gbiggs/rtcpcl: RT-Components for the Point Cloud Library,” GitHub, 25 4 2014. [オンライン]. Available: https://github.com/gbiggs/rtcpcl.git. [アクセス日: 25 11 2023].

[20] 産業技術総合研究所, “OpenRTM-aist を 10 分で始めよう！,” 11 11 2022. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/doc/installation/lets_start.

[21] 産業技術総合研究所, “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),” 11 11 2022. [オンライン]. Available: https://www.openrtm.org/openrtm/ja/node/6057.

洞窟の 3 次元点群を用いた図面化システムの RT コンポーネント開発

2024 年 12 月 1 日　初　版第 1 刷発行

編　集　藤井雄基
発行元　岡山大学