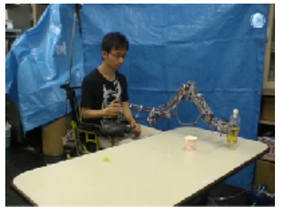
ロボットミドルウェアでロボットシステムを作る

株式会社SUGAR SWEET ROBOTICS

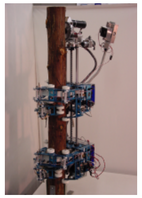
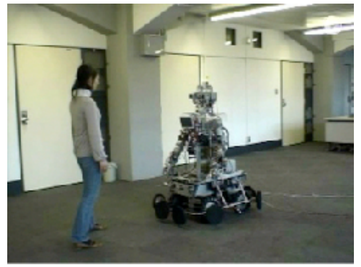
菅　佑樹

自己紹介

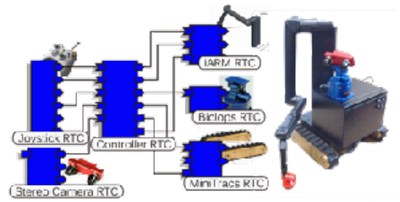
- 菅　佑樹 (Yuki Suga)
  - 職歴
    - 2012-現在 株式会社SUGAR SWEET ROBOTICS代表取締役
    - 2012-現在 早稲田大学表現工学科尾形研研究員
    - 2010-2012 株式会社リバスト
    - 2007-2010 早稲田大学総合機械工学科助手 (菅野研)
  - 学歴
    - 2008 博士 (工学)
    - 2002-2004 早稲田大学理工学研究科機械工学専攻
    - 1998-2002 早稲田大学理工学部機械工学科
    - 1995-1998 早稲田大学附属高等学院
  - 趣味
    - 子育て
    - 文房具，万年筆収集
    - インラインスケート，ヨーヨー

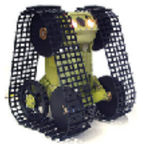


早稲田時代

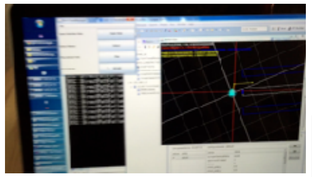





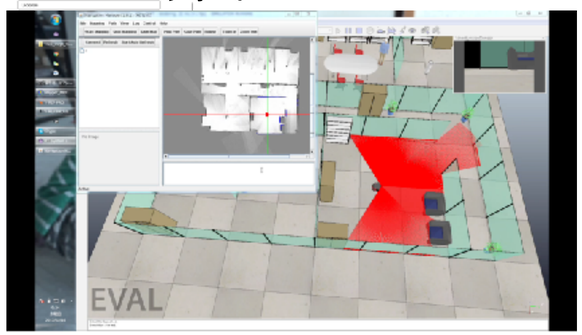



リバスト時代

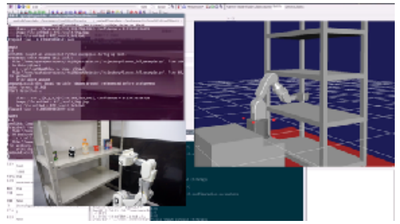





SSR時代



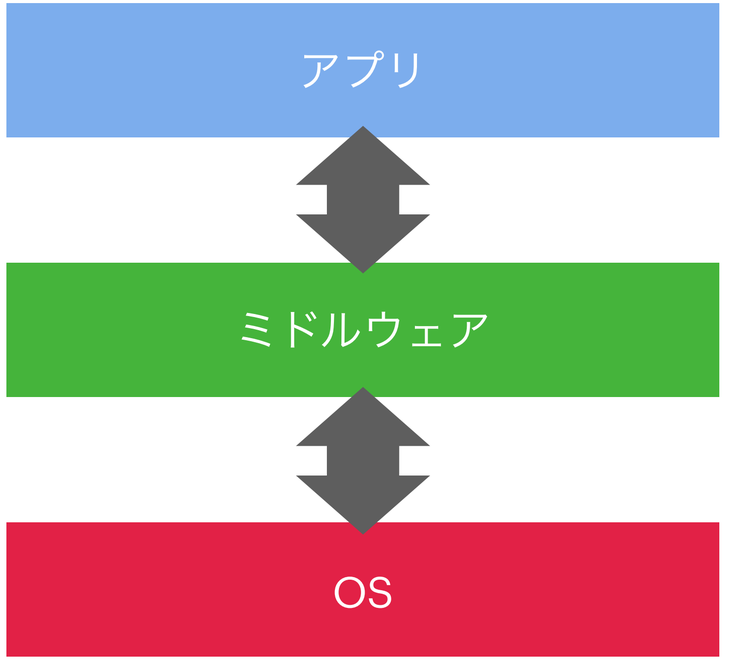


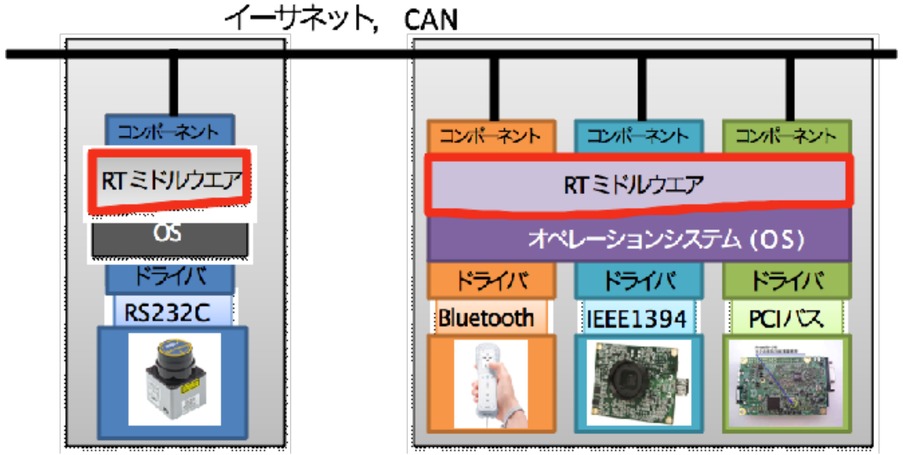


ロボットシステムを
ロボットミドルウェアで作る

ロボット用ミドルウェア

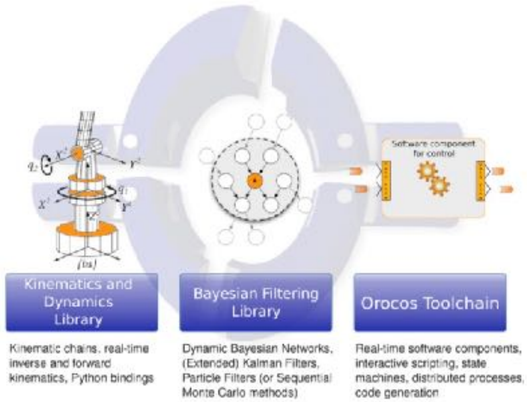
- 最近はROSやOpenRTM-aistに代表される「ロボット用ミドルウェア」が便利
  - そもそもミドルウエアとはソフトウエア間の通信を補完する通信プロトコルおよびライブラリ・ツール群
    - OSをラッピングして，OS間の差を吸収
    - OSの機能や定型的な手続きを簡単化
    - 異なる言語で開発しても一つのサービスに統合が可能
    - 例： Unityなどのゲームミドルウェア
  - 分散システムの文脈では，特にネットワーク越しに通信をした時に，ホストとなるサーバーのOSや実装をラッピングして，プラットフォームによらない通信を提供するためのライブラリ群のこと
    - 例：DBMS (Oracle, MySQLなど)
    - 通信を標準化して，モジュールのソフトウェアを使いやすくする

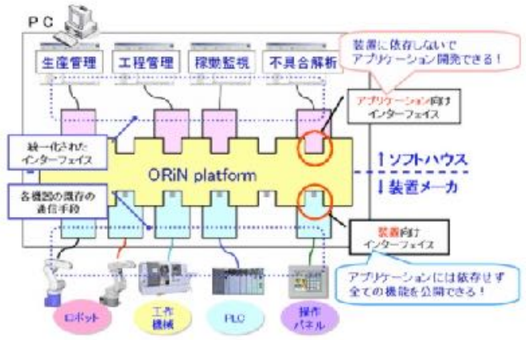


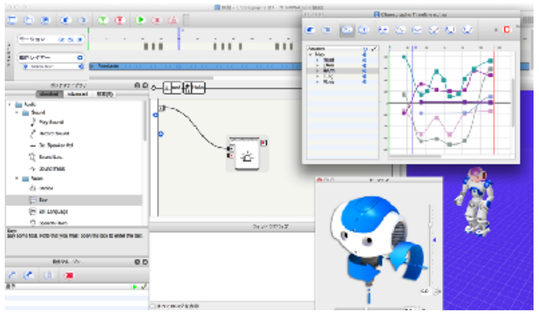


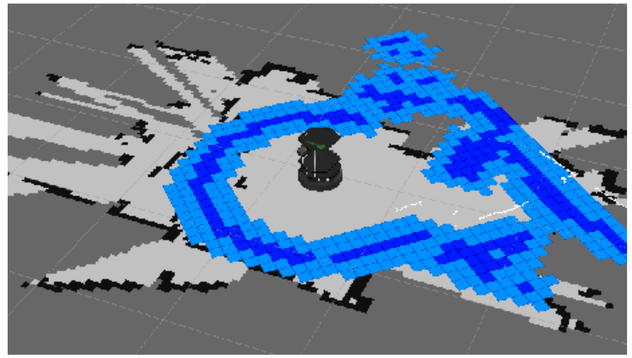
ロボットミドルウェア

- Microsoft Robotics Developer Studio
  - Kinectでのモデリング
  - 動力学シミュレータ
- YARP
  - 赤ちゃんロボットiCub．人工知能研究分野
- OROCOS
  - ヨーロッパ
  - コンポーネントモデル
- OpROS
  - コンポーネントモデル
  - 韓国製
- ORiN
  - DENSOウェーブの産業用ロボット
- ROS
  - Robot Operating System
  - もっとも使われているロボット用ミドルウェア
  - DARPA Robotics Challengeでも採用
- RT-middleware
  - 国際団体OMGで規格化された規格
  - OpenRTM-aistなど，多くの実装がある
- naoqi
  - フランスのAldebaran Robotics社が開発
  - NAO用だが，SoftBankのPepperにも採用
- NVIDIA Issac SDK
  - NVIDIAが開発
  - Protobufで通信．コンポーネントモデル



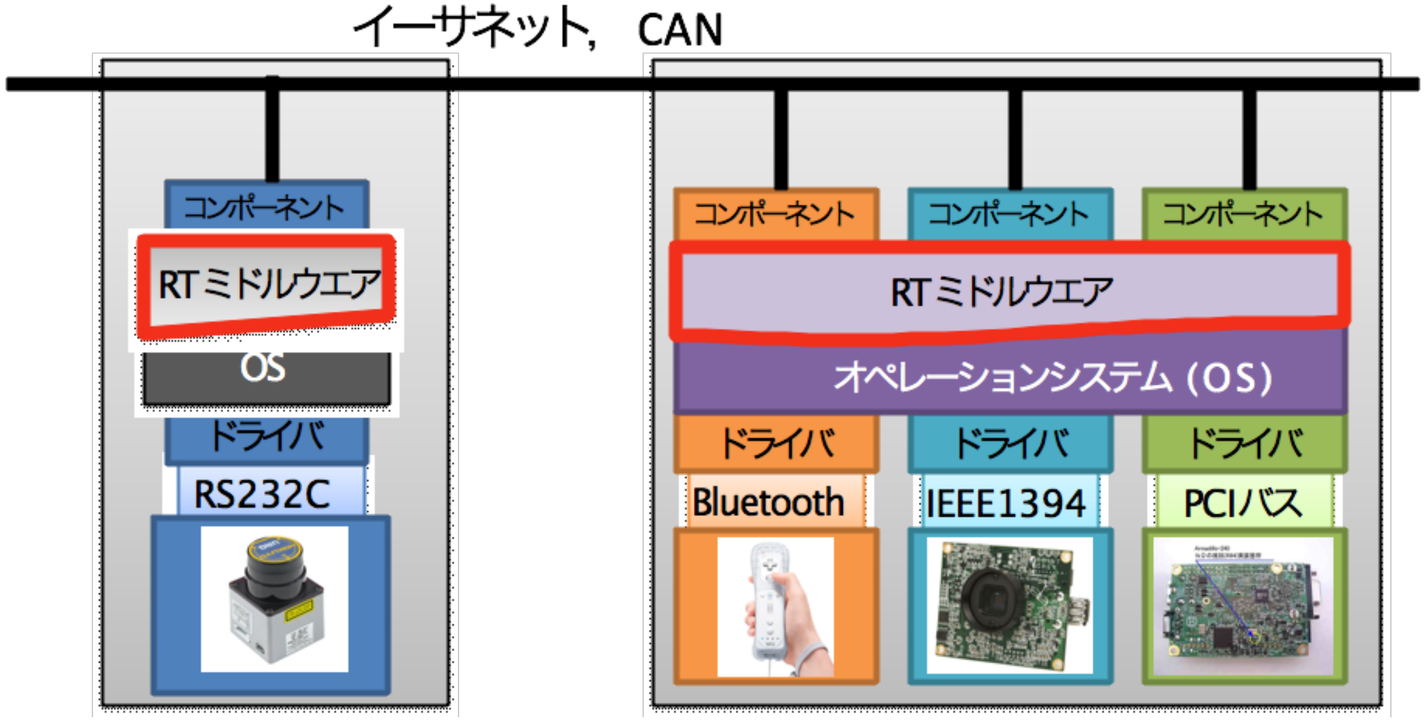






ミドルウェアの役割

- ネットワーク越しのリソースの利用促進
  - 通信を標準化することで，OSがLinuxでもWindowsでも関係なくなる
  - 同じ種類のデバイスであればデータを標準化して入れ替えられるようになる

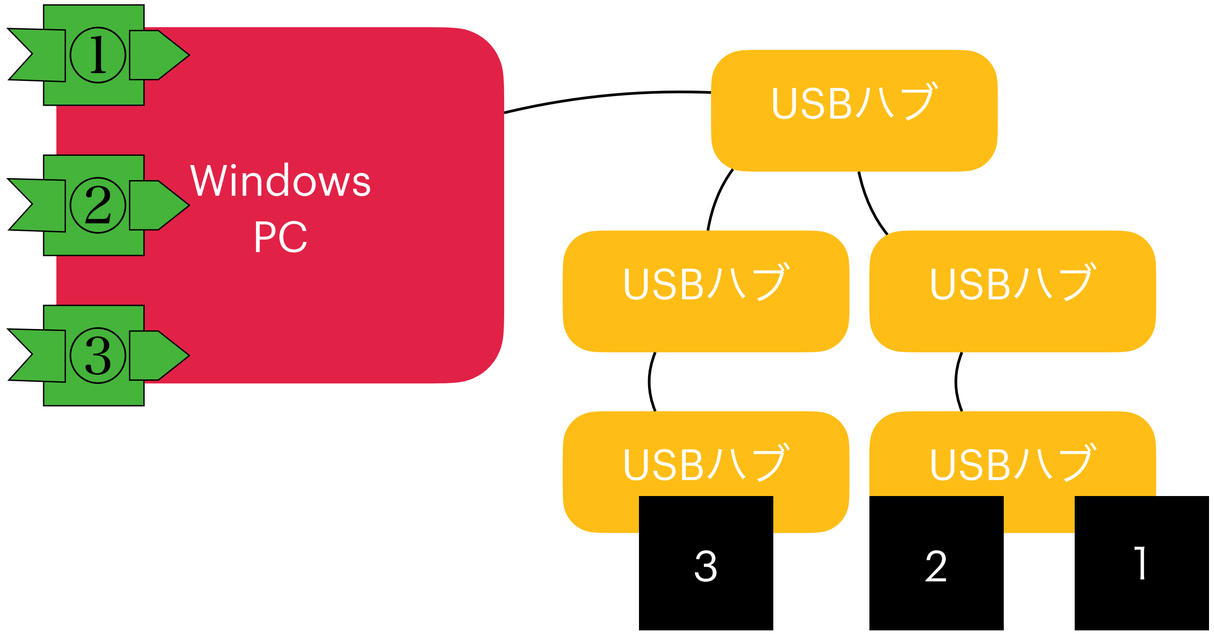


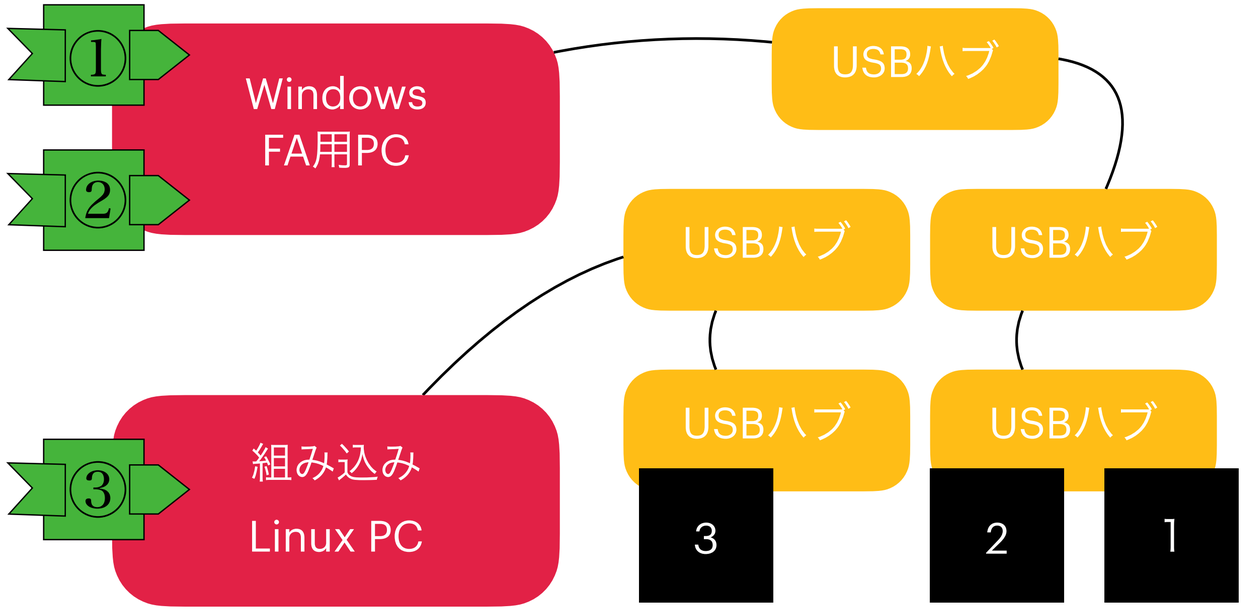
例えばこんな時ありがたかった

論理設計と物理設計の分離

- 論理設計・・・欲しい機能をモジュールとしてまとめる設計
- 物理設計・・・モジュールの配置．分散システム上のどのマシンにリソースを配置するか
- テスト環境ではうまく動いたが，本番環境で動作しなかったので配置を変更した
  - 本番用の工業用PCがUSBハブ認識数に制限があったので，分散システムにした
    - RTCをLinux用に変更して組み込み

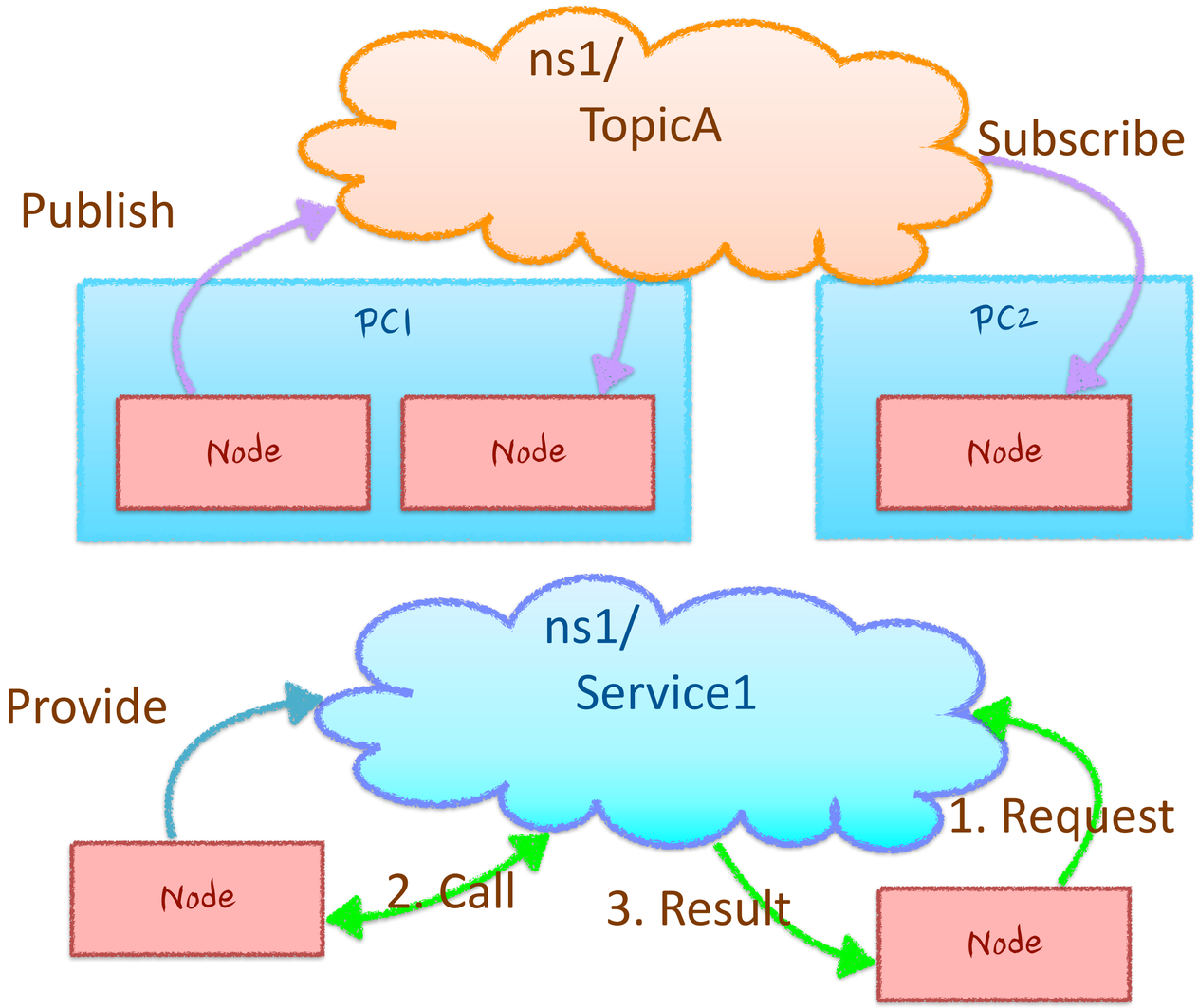
      Linux上で動作させた





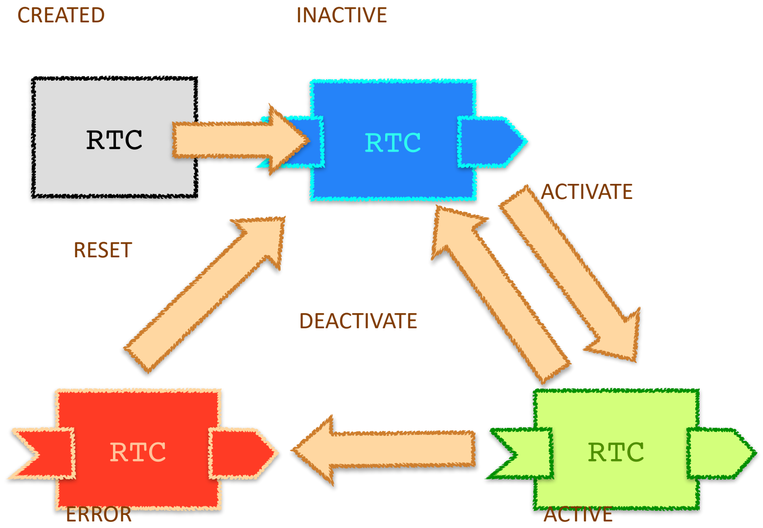
ROS

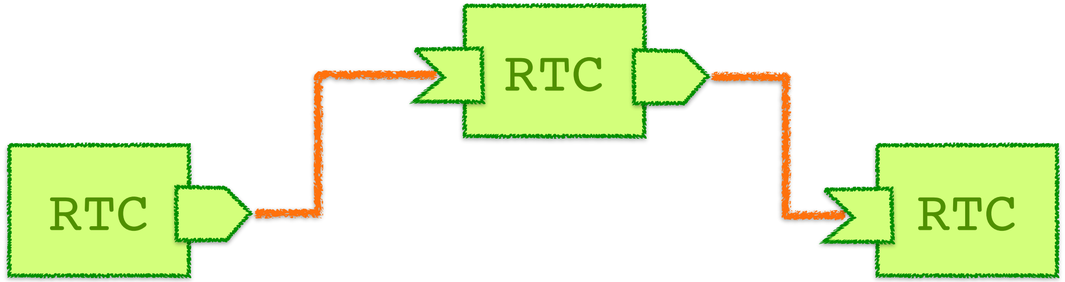
- 機能単位＝Node
- データフロー型通信＝Topic
- 遠隔手続き型通信＝Service
- 機能単位の調整＝Parameter



OpenRTM-aist

- 機能単位＝RT Component
  - 機能単位に状態マシンがある
    - INACTIVE-ACTIVE-ERROR
- データフロー型通信= Data Port
- 遠隔手続き型通信＝Service Port
- 機能単位の調整＝Configuration





ロボットミドルウェアの特徴

- ROS, OpenRTM-aistはコンポーネント指向型ミドルウェア
  - 機能単位： RTComponent，Node
  - 通信機能を提供
    - メッセージ指向型： DataPort, Topic
    - 遠隔呼び出し型： ServicePort, Service, Action
    - 機能を実行時に調整するための仕組み： Configuration, Parameter
  - 異なるAPIをカプセル化
    - 共通の通信規格でデータを送受信可能に
    - 複雑なAPIやデータの集まりを隠蔽し，共通化

ロボットシステムをミドルウェアで組む目的

- コストを下げること
  - 時間的・金銭的コスト
    - 機能の再利用による開発工数削減
    - 将来のカスタマイズ工数を削減
    - 複数プラットフォーム対応のための工数削減
- コストが下がらなければミドルウェアなんか使わなくていい
  - 以下の条件のANDならミドルウェアの利用は全く進めない！
    - 一個しか作らないし，今後自分も含め，他の誰も使わない特殊な環境，ハードウェア
    - 一人ぼっちで開発する.
    - 誰にも知られない，使われたくない，使われない.

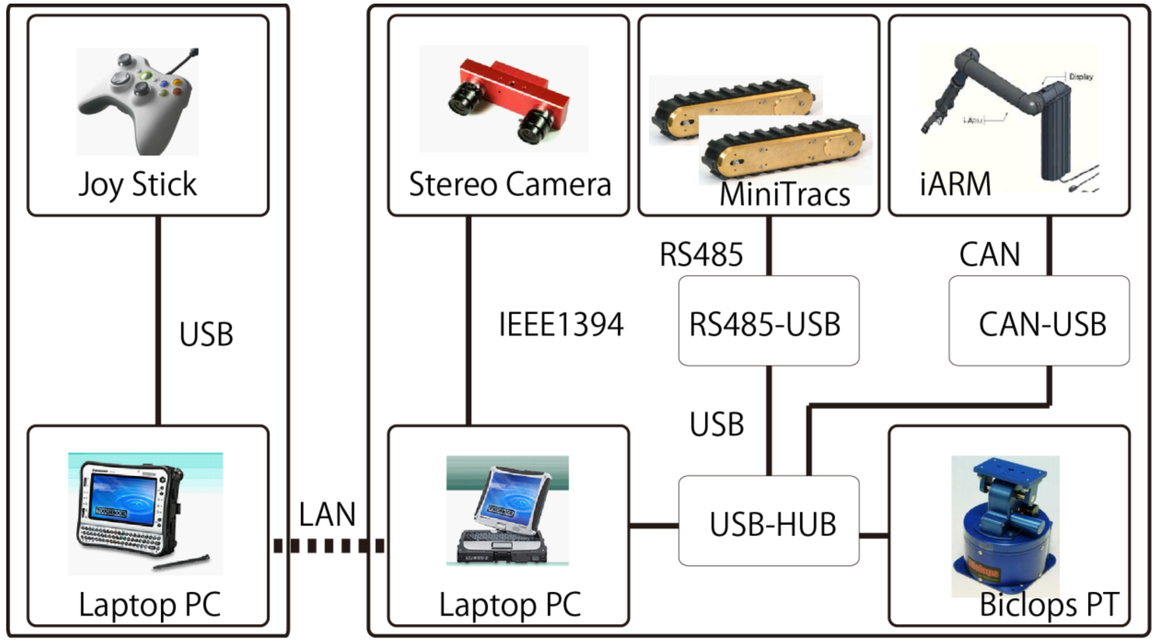
ロボットミドルウェアでシステムを組む

- コストが下がるポイント
  - 多人数で並行作業，外注，共同研究
  - 他人の機能要素やツールを再利用
  - 論文の最新技術をフォロー
  - 他人が運用する
- コストがかかるポイント
  - ミドルウェア自体の学習コスト．人間の調達コスト（調達難易度高）
  - オーバーヘッド，無駄処理
  - 分散システムの管理運用コスト

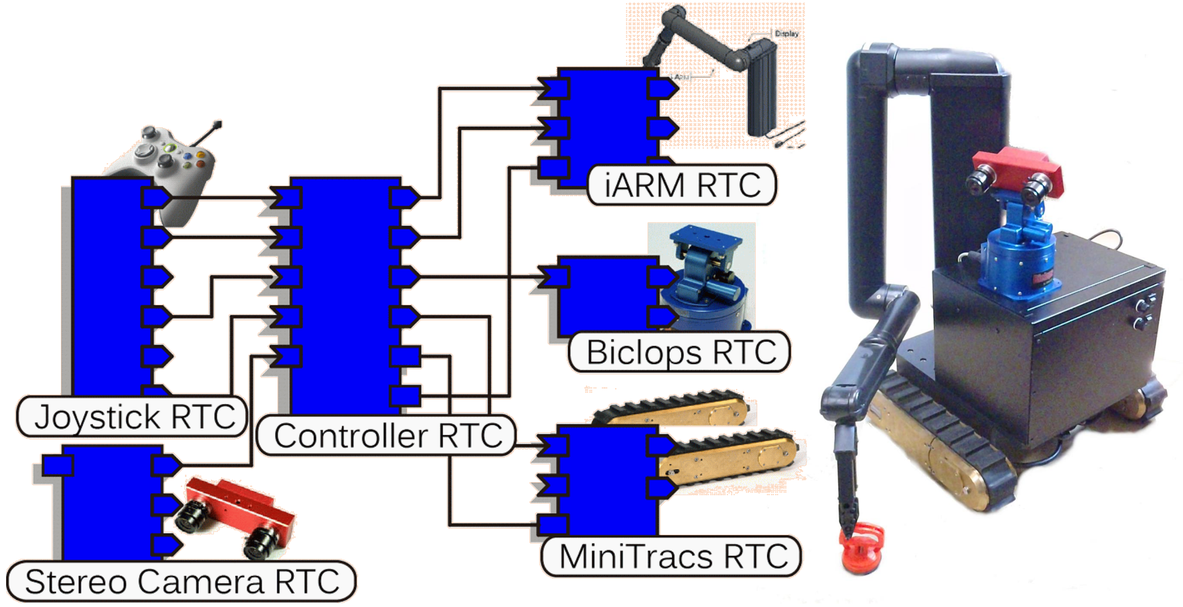
コンポーネント化と再利用

- システムの統合
  - すべてが異なるAPIを利用するロボット製品)
    - カナダInuktun製クローラ
    - オランダED社製アーム
    - アメリカTRACLabs製パンチルト雲台
    - アメリカVIDERE Design製ステレオカメラ
  - PCとの通信ラインの接続方法も異なる
  - RTコンポーネントでAPIをラッピングして置くことで，ソフトウェアの再利用性が向上
  - ネットワークで隔てられたPC間の通信が容易
  - 異なる言語で書かれたRTCでも通信可能
- モジュールの分割とインターフェース (論理設計)と実際のハードウェアの接続 (物理設計)を分離することがミドルウェアのメリット

ハードウエアの構成（物理設計・配置）



RTシステムの構成（論理設計）



インテグレーションのコスト舐めんな

- これまでのサマーキャンプとか，学生プロジェクトを見る限り
  - 役割分担まではできる．SysMLっぽいものを書いたりできる
  - 進捗ミーティングでいくら『インテグレーション舐めんな』といってもやらない
  - 動かしてみると検討不足（よくあるのはシステムの開始や終了部分）でインターフェースを見直したり，パラメータチューニングが面倒でデモに間に合わない
- OpenRTM-aistやROSはインテグレーションしながら組めるのよ．やって！お願いだから．みんなの苦しむ姿が見たくないの．

短期間でのシステム構築の手順

- SysML等で設計図を書く
- スケルトンコードを書く
  - RTCBuilder
  - ROSなら必要なパッケージとNodeのメインコードを全て書く
- 再利用するモジュールはダウンロードしてビルドしてみる
  - ここでつまづくのが多い．中身空っぽのStubコンポーネントを作っておいてもいい．僕なら作る．
    - ロボットハードのStubコードはハード依存が取れてハード抜きで開発を進めたり，テストしやすくなる
- とりあえずビルドして接続されることを確認する
  - RTSystemEditorなら簡単．SystemProfileを保存する
  - ROSはlaunchを書いてrqtなどで確認．launchも保存する
- ここで全部githubに上げる
- 中身は空っぽだが，入力された値を表示するコードくらいは書いておくといい
- 一つ一つのモジュールを実装して，出力を確認しながら進める

Stubコンポーネント

- ハードなしでもシステム開発できるようにする
  - シミュレータと繋げるにはシミュレータ環境を作るのが面倒
  - コンポーネントだけで小さなシミュレーションを可能にしておく
    - 例：移動ロボット
      - 速度指令を時間積分するだけのコンポーネント
    - 例：アームロボット
      - 関節角度指令に対して，指令速度で数値を変更するだけ
    - 逐次ログを取っておく (csv) →Excelとか Matplotlibで可視化

それぞれのモジュールの実装方法

- ROSでもROS2でもRTCでも，やることは同じ
  - ミドルウェア相互に移植しやすく書くべし
    - ミドルウェアに依存しない
    - いざとなればミドルウェアを外して単一のアプリに変更できる
  - 内部に機能を実装するクラスを作り，ROSのノードのコードや，RTCのコードにはデータ型の変換コードのみを書くべき
    - ご参考： UrgRTC (拙作) https://github.com/sugarsweetrobotics/UrgRTC/blob/master/src/UrgRTC.cpp

モジュールを単体でテストする方法

- データを突っ込んでみる
  - OpenRTM-aist ・・・ rtinject (rtshell)
  - ROS・・・ rostopic pub
- 出てくるデータをprintしてみる
  - OpenRTM-aist ・・・ reprint (rtshell)
  - ROS・・・rostopic echo
- ログを取ってみる．再生してみる
  - OpenRTM-aist ・・・ rtlog (rtshell)
  - ROS・・・ rosbag

例えば・・・

・出来上がったコンポーネントに対して，rtprintを準備してからrtinjectでデータを送信するbatスクリプトを用意する

・RTCをコンパイルしてVisual Studioでデバッグしながら，上記のbatスクリプトを実行すればデバッグが簡単 (RTSEでつなぎながらテストすると大変)

例えば・・・

・実験環境でLiDARとロボットのオドメトリをrtlogコマンドで一気に収集

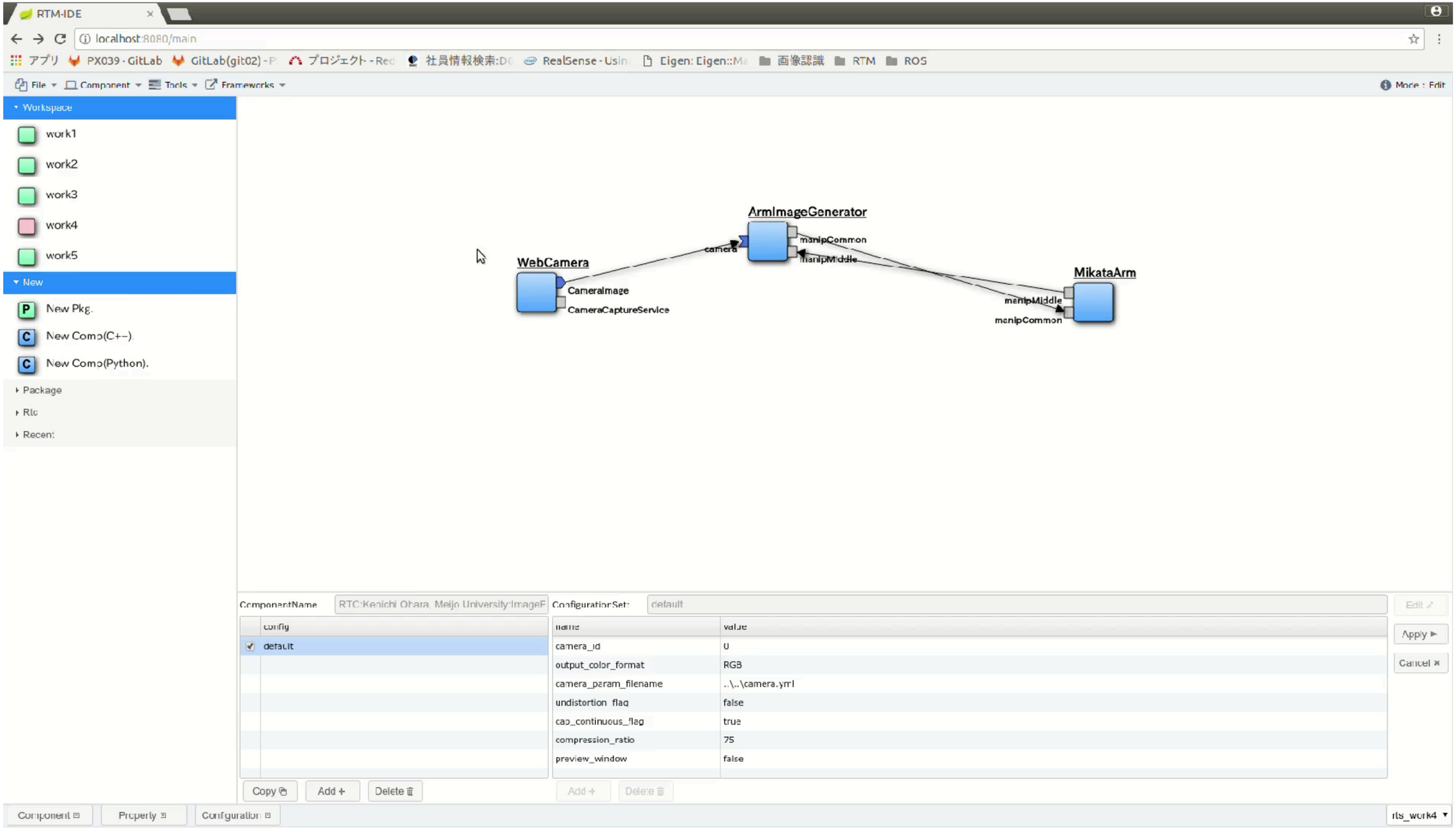
・帰宅してオドメトリとLiDAR情報でSLAMしてみる．パラメータを調整しながら完成するマップを確認できる

モジュールの中のログをとる

- ミドルウェアが用意しているログ記録方法を使うと良い
  - printfなどでコンソールに出すと共有しにくい．ファイルに保存できれば検索可能
- OpenRTM-aist・・・RTC_DEBUG, RTC_INFOなどのコマンドがある
  - RTC_DEBUG((“DataInPort(‘in’) received data (%d)”, m_data.data));
  - RTC_INFO((“....”));
  - RTC_ERROR((“...”));
- ROS・・・roslogを使う
  - ROS_DEBUG(“Topic(‘hoge’) received data(%d)”, msg->data);
  - ROS_INFO(“...”);
  - ROS_ERROR(“...”);
- ログのコツ
  - ログメッセージは5W1Hで出すこと（Who, When, Where, What, Why, How）
  - ログレベルを考えて使う．

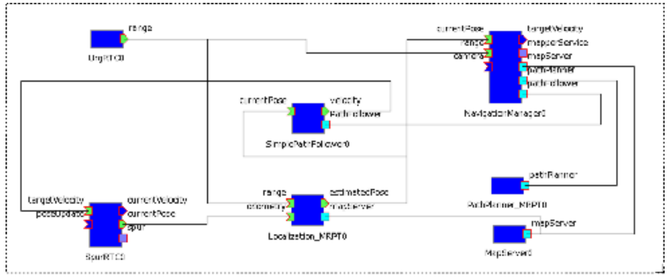
OpenRTM-aistの使えるもの

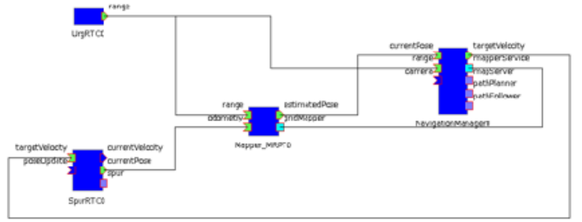
- とりあえずこのサイトは見て欲しい
  - wasanbon.org
  - 本来wasanbonはRTCのリポジトリ管理＋ビルド＋運用ツールだけど，このサイトの右上にはwasanbonで管理しているRTCが並んでいる
    - RTCBuilderが掃き出すRTC.xmlというプロファイルを使ってヘルプを自動構築した結果
    - カテゴリが階層状になってるともっといいんですけどね
  - セックさんにも使ってもらっています
    - Air-graph https://sec-airgraph.github.io/AirGraph-doc/
  - 今，Python3.8およびOpenRTM 1.2.2に対応作業中です.

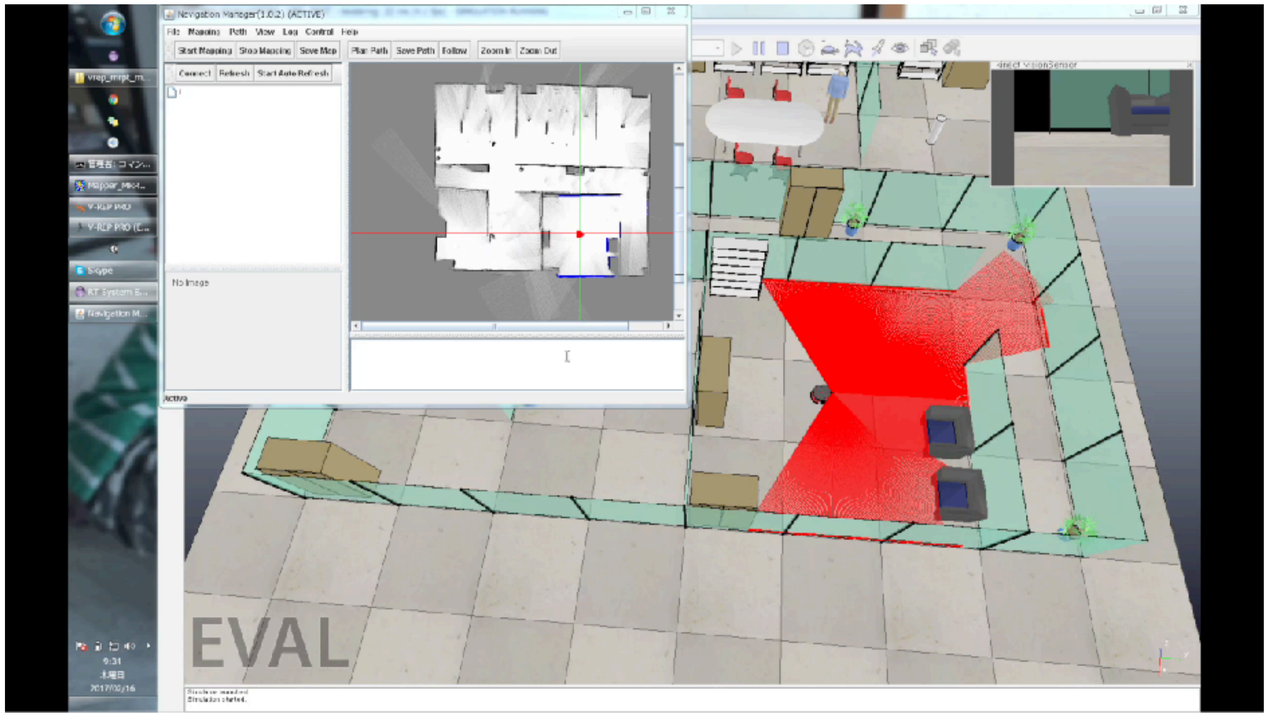

RTM-IDE
localhost:8080/main
File
Component
Tools
Frameworks
Mode : Edit
Workspace
work1
work2
work3
work4
work5
New
New Pkg.
New Comp(C++).
New Comp(Python).
Package
Rtc
Recent
ArmImageGenerator
camera
manipCommon
manipMiddle
WebCamera
CameraImage
CameraCaptureService
MikataArm
manipMiddle
manipCommon
ComponentName
RTC:Kenichi Ohara Meijo University:ImageF
ConfigurationSet:
default
Edit
config
name
value
default
camera_id
0
output_color_format
RGB
camera_param_filename
..\..\camera.yml
undistortion_flag
false
cap_continuous_flag
true
compression_ratio
75
preview_window
false
Apply
Cancel
Copy
Add +
Delete
Add +
Delete
Component
Property
Configuration
rls_work4

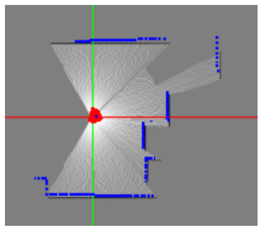
ロボットミドルウェアで構成したシステムの例
SLAMとナビゲーション

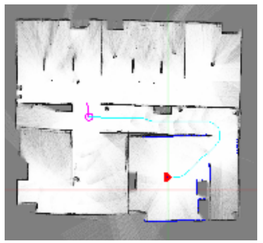
- 各種移動ロボット対応
- RTCを入れ替え可能
- SLAM（マップ生成）
- 自己位置推定（パーティクルフィルタ）
- パスプランニング

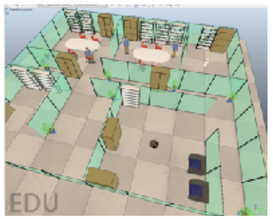












Arduino + RT-middleware = RTno

あーるてぃーの

- RTno “アールティーノ” = RT-middleware + Arduino
- ArduinoとRTミドルウエアの連携
- RTコンポーネントをArduinoで作る
- 組み込みプログラミングのみでRTCを作る

データポート構成など
どのプロファイル

コンポーネントの
自動生成 (ProxyRTC)

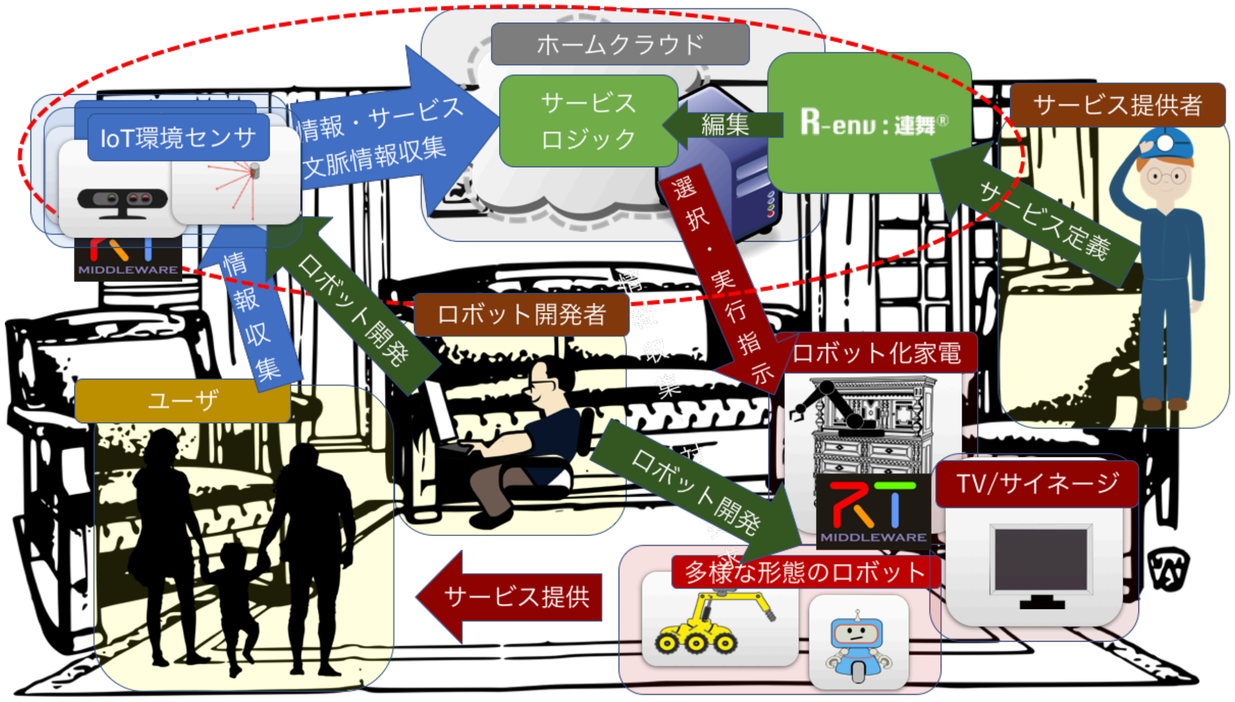
データポートの通信
状態マシンの制御

組み込みプログラム
・データポートの宣言
　・Double型出力ポートx1
　・Long型入力ポートx1
・出力ポートからA/D変換結果を出力
・入力ポートへの指令でLEDを制御

ロボットミドルウェアの応用例

ロボットミドルウェアとR-envの連携

- シナリオを編集してロボットシステムを現場に適用する「運用」と，基本的なロボットシステムを開発する「開発」の分業が目的
- ロボットプログラミングツール R-env (NTT) と OpenRTM-aist / ROS / ROS2の連携
  - R-envによるロボットサービスのシナリオ作成
  - ロボットシステムを構成するミドルウェアレイヤにR-envから指令を送ることができる
    - データ型変換
    - データフロー型送受信＋遠隔呼び出し型通信



まとめ

- さきに全体を組む．空っぽでもいい
- ロボはStubにできると良い．シミュレータが簡単ならそれでもいい
- 一つずつ実装していく．出口側のRTCで出力をログしておいて，CSVで確認するとかできる
- コンポーネントの中のコードはクラスでひとまとまりにしておきたい
  - できればそのクラスの単体テストを実装したい
- デモを一周できる最低限のコードを実装していく
  - さっさとデモだけはできるようにしたい
- デモ＋αを実装するときは，最低限デモを通した時のコードを変更しないで済むような設計がいい
  - 新しい機能は新しいコンポーネントを追加して機能追加できるようにしたい