

RTミドルウェア入門

株式会社SUGAR SWEET ROBOTICS

早稲田大学ヒューマノイド研究所 招聘研究員

芝浦工業大学SIT研究所 客員研究員

東京大学情報理工学系研究科大学発新産業創出拠点

プロジェクト (JST-START) 特任研究員

菅 佑樹

コンテンツ

- 自己紹介
 - ロボット屋がミドルウェアを使う経緯
- 海外の動向
 - 海外でのロボット用基盤ソフトウェアの現状
- RTM応用事例集
 - NEDOプロジェクト等でのRTMの利用
- 普及活動
 - RTミドルウェアサマーキャンプなど
- 今後の展望



自己紹介

自己紹介

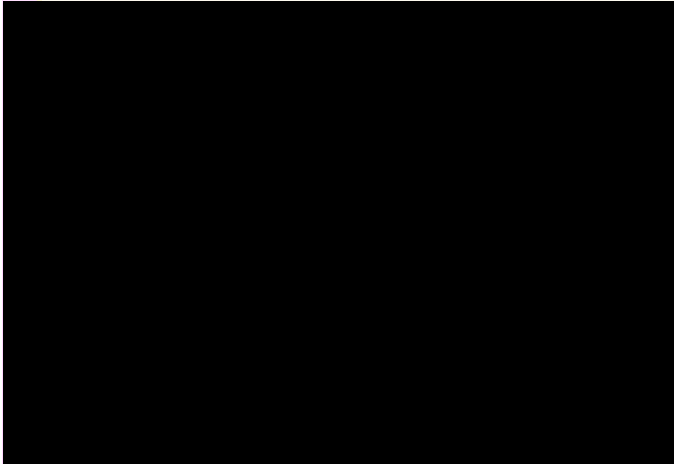
- 菅 佑樹 (Yuki Suga)
 - 2012～ 株式会社SUGAR SWEET ROBOTICS代表取締役
 - 2010～2012 株式会社リバスト
 - 2007～2010 早稲田大学総合機械工学科助手



<http://ysuga.net>

<http://sugarsweetrobotics.com>

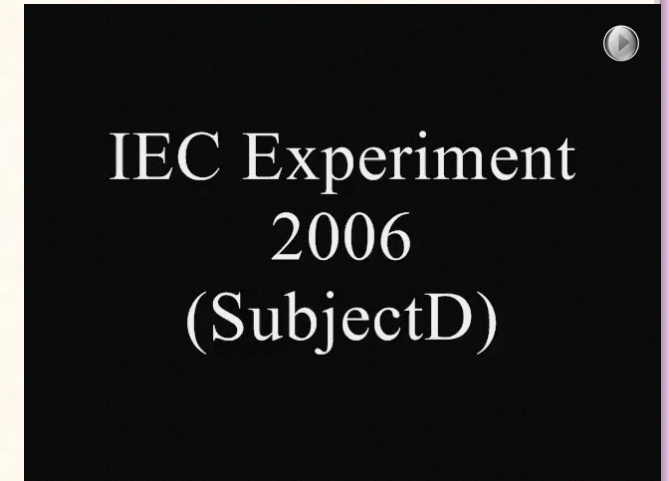
- 2007～2010 早稲田大学総合機械工学科助手
<http://www.sugano.mech.waseda.ac.jp>



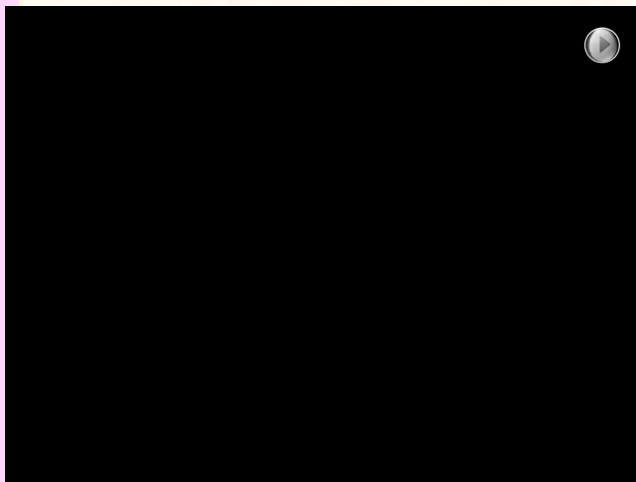
林業機械支援システム
岐阜県・早稲田大学WABOT-HOUSE
研究所による



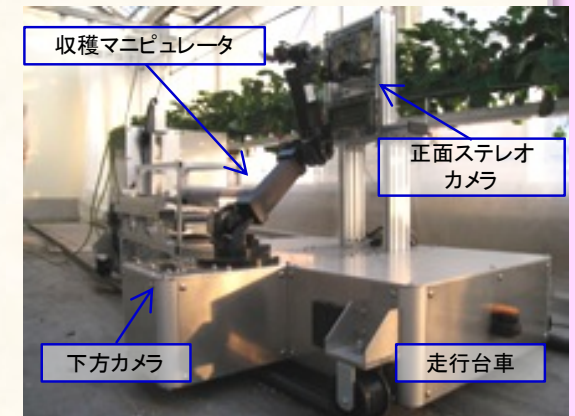
学習適応する
コミュニケーション
ロボット



IEC Experiment
2006
(SubjectD)



車いす搭載型
ロボットアーム



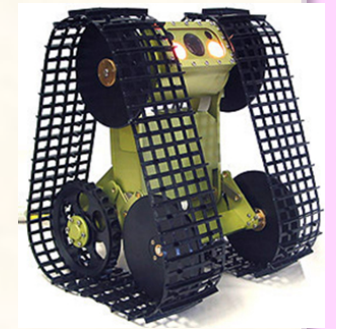
いちご収穫ロボット
前川製作所との共同研究



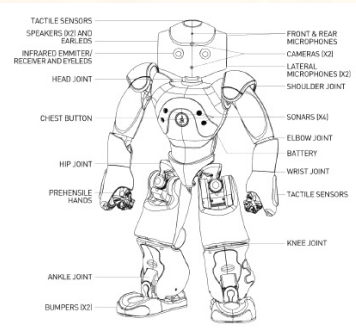
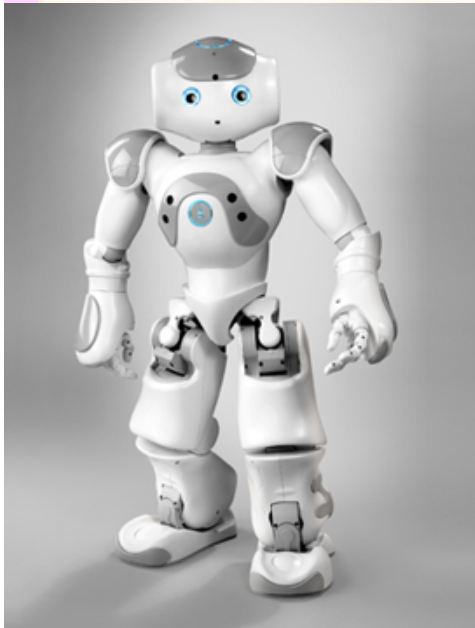
スイス Neuronics社
Katana ロボットアーム



アメリカ Mobile Robot社
移動台車 Pioneerシリーズ



2010年
危機管理産業展
出展ロボット



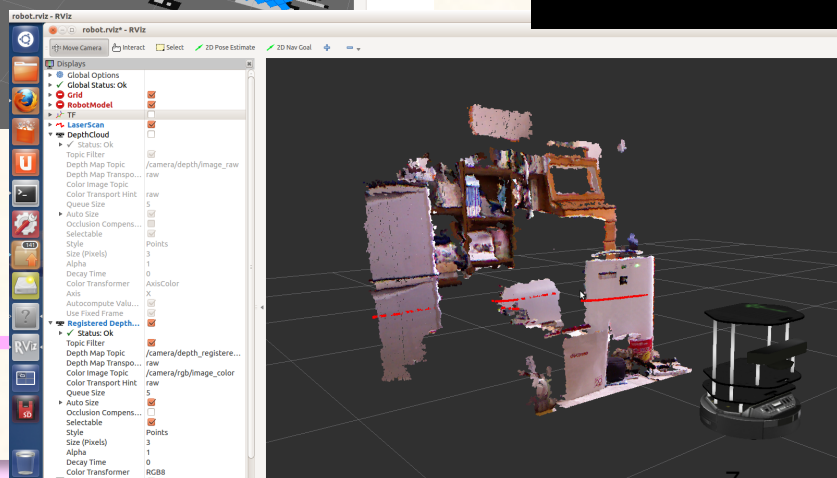
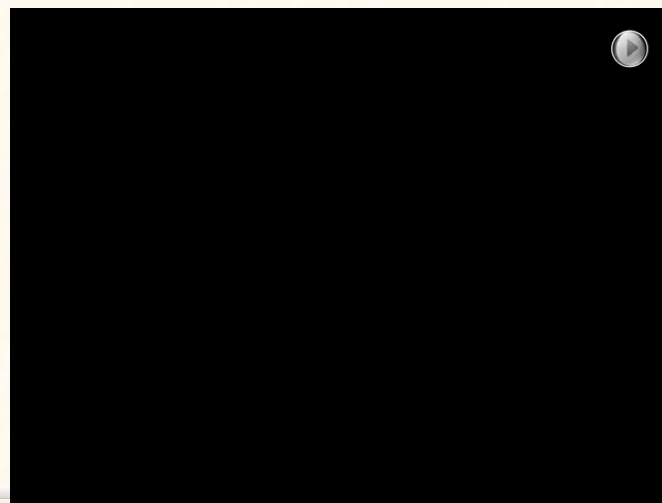
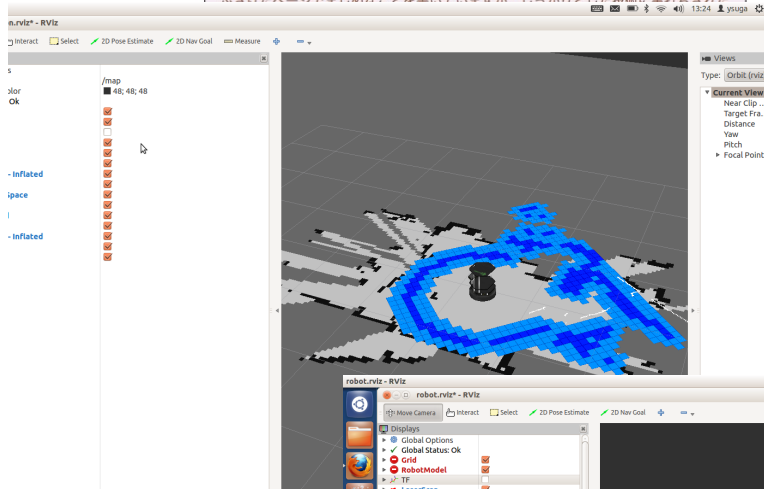
フランス Aldebaran Robotics社
NAO

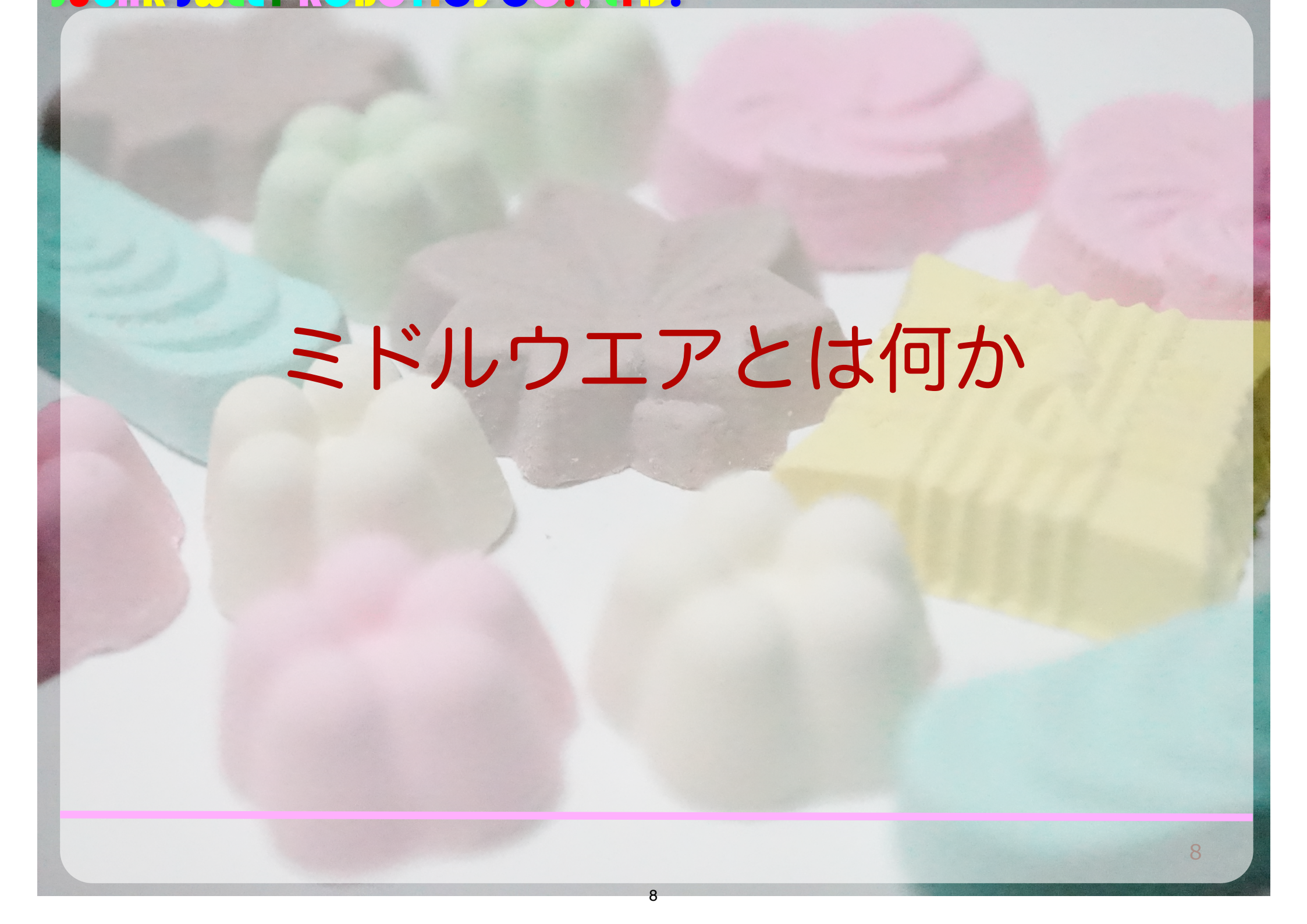


カナダ Inuktun社
探索ロボット



- 2012～ 株式会社SUGAR SWEET ROBOTICS
 - ロボットの受託開発・コンサルティング
 - RTミドルウェアに関する開発
 - ロボット用ミドルウェアに関する講演

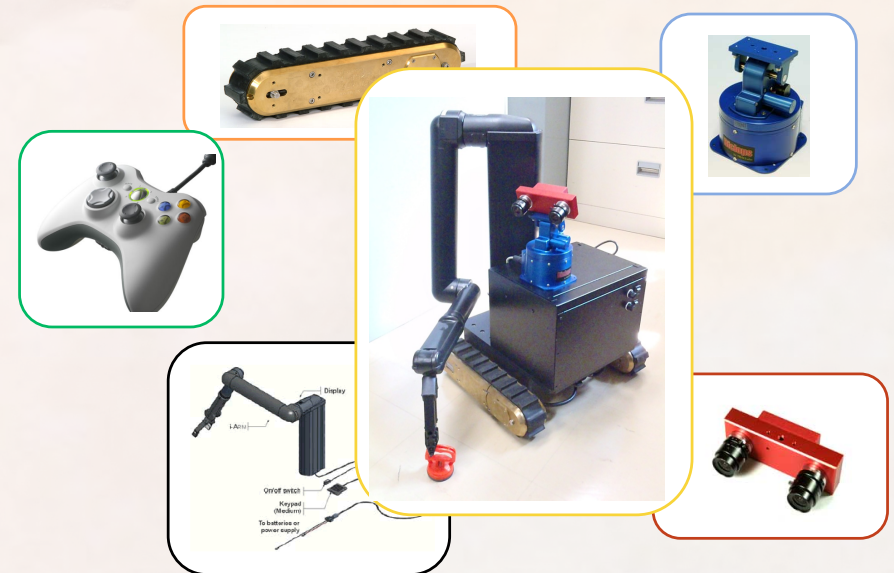
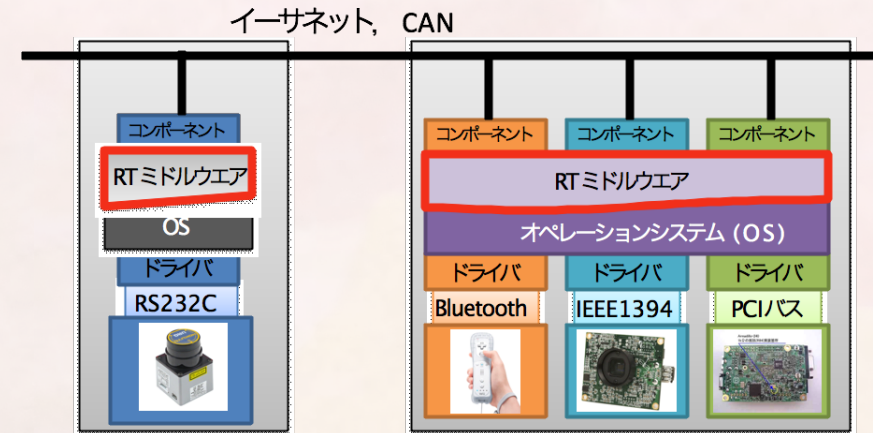




ミドルエアとは何か

ミドルウェアとは

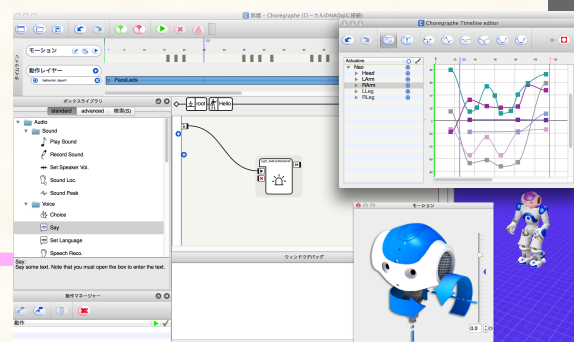
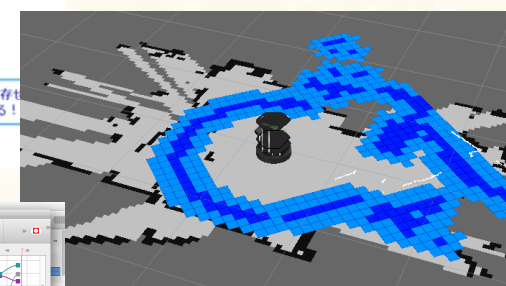
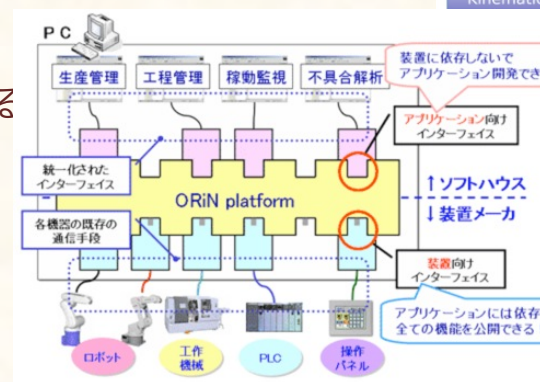
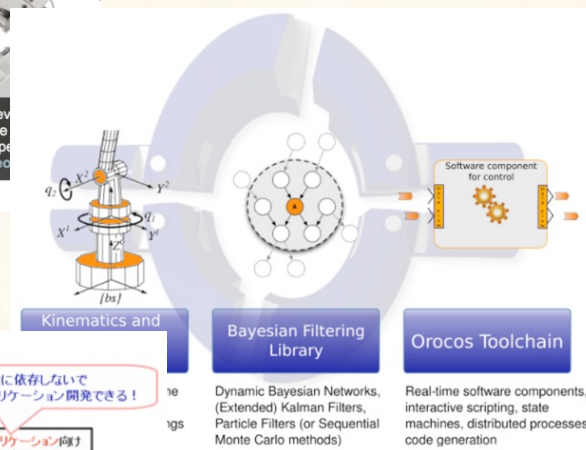
- ミドルウェアとは
 - ソフトウェア間の通信を補完する通信プロトコルおよびライブラリ・ツール群
 - OSとの通信をラッピングして、OSの違いを吸収する
 - 異なる言語で開発しても通信が可能
 - 例：DBMS (Oracle, MySQLなど)
- ロボット用ミドルウェアとは
 - ロボット用ソフトウェア間の通信を補助するライブラリ・ツール群
 - OSをラッピングしてOSの違いを吸収
 - ネットワーク経由で遠隔から通信可能（分散システム構築）
 - 複数の言語で開発可能

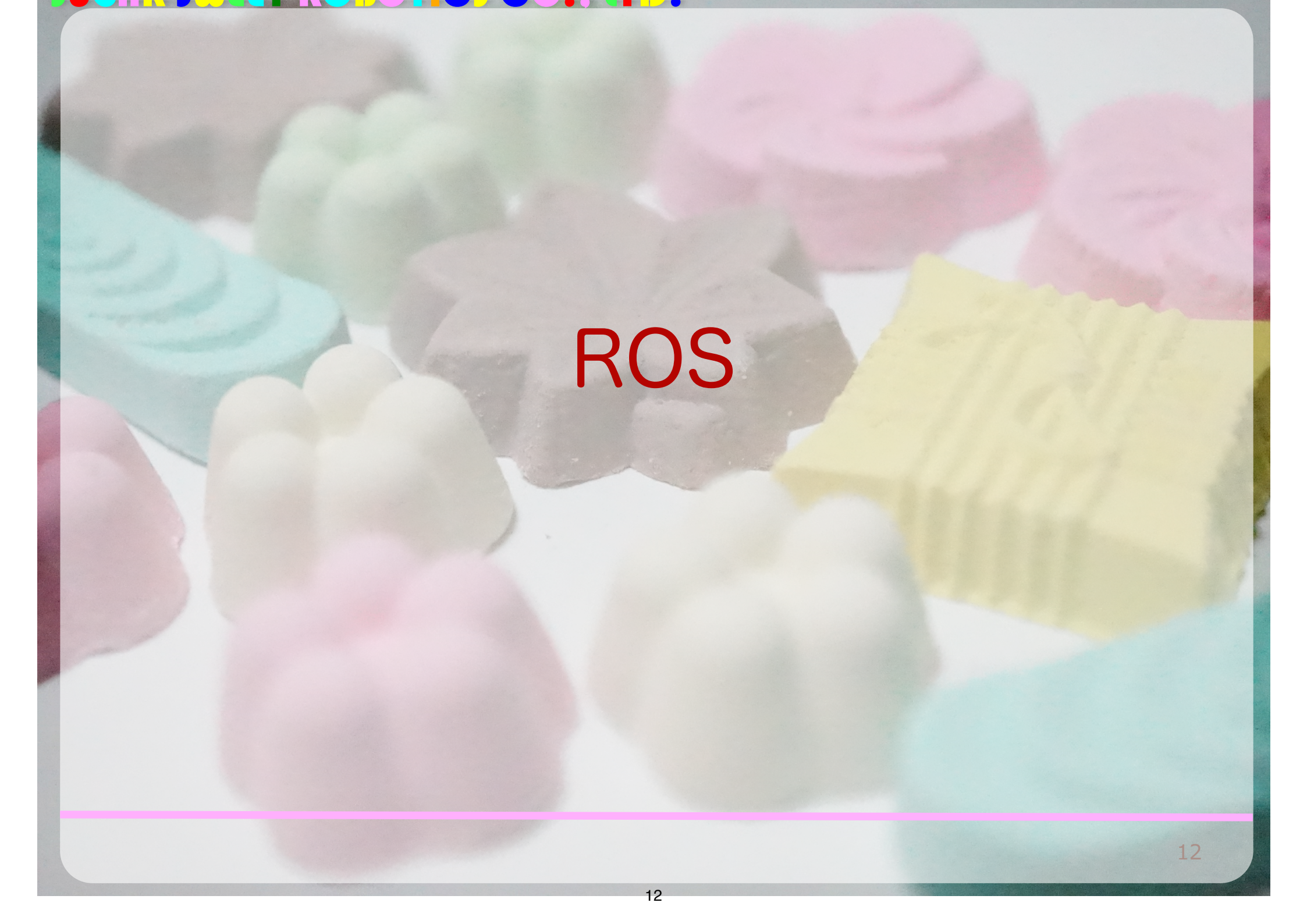




ロボット用 基盤ソフトウェア技術の動向

- 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
 - コンポーネントモデル
 - ロボット知能化のためのライブラリ
- OpROS
 - コンポーネントモデル
 - 韓国製
- ORiN
 - DENSOの産業用ロボットに広く使われている
- ROS
 - Robot Operating System
 - 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

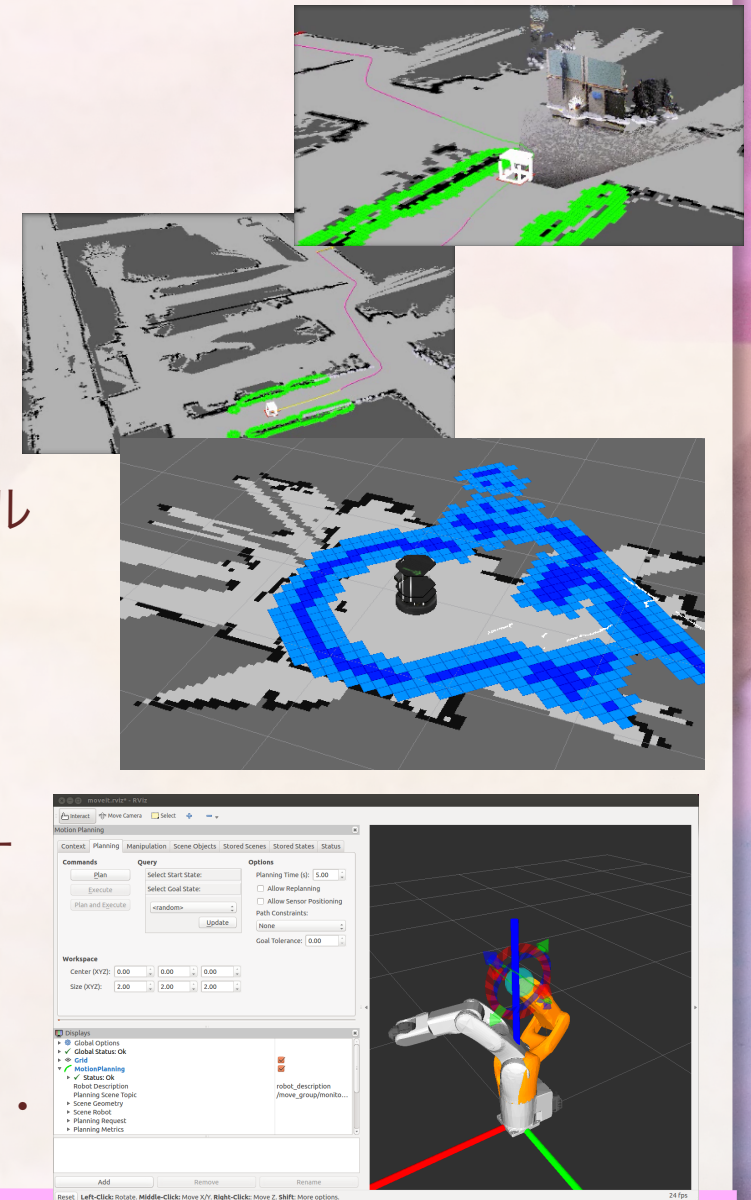




ROS

ROS

- ROS (Robot Operating System) とはいえOSではない
- Ubuntu Linuxに特化しており， Unixコマンドラインツールに慣れたユーザは使いやすい
 - 基本的にオープンソース文化
 - コードが読める人はどんどん開発していきける
- 通信のためのミドルウェアライブラリが基本でシンプル
- コミュニティが盛んになっている
 - パッケージ管理・ビルド・テストツールが充実
 - ロボットを智能化させるツール群も増えている
 - 論文発表に合わせてROSのパッケージをリリースする研究も増えてきた
- 品質は各個に保証される
 - ドキュメントと実装の同期が取れてないものも・・・

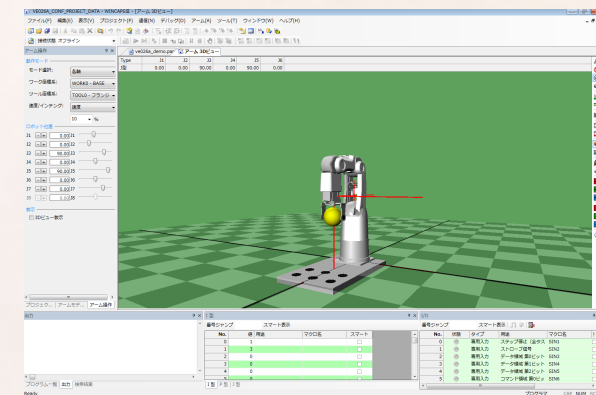
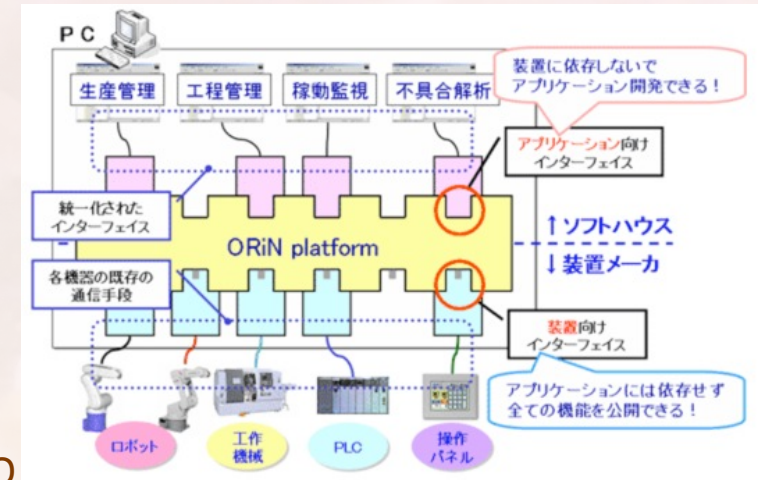


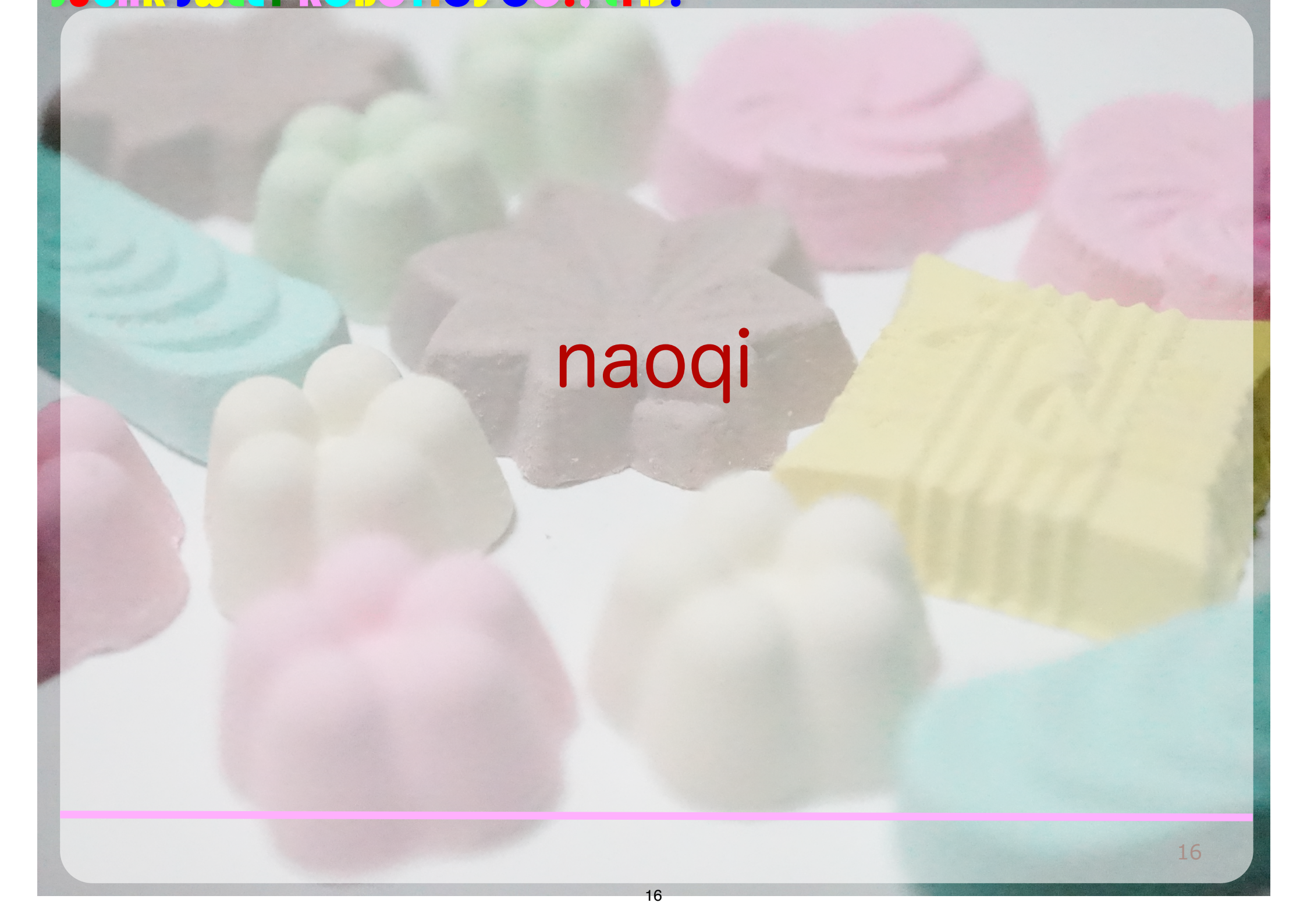


ORiN

ORiN (オンライン)

- 工場の監視などの工業応用が目的
 - 企業が中心となってORiN評議会を設立して開発
- DENSOウェーブが自社のロボット用の標準プラットフォームとして採用しメンテナンス
 - 信頼性が高く保証されている
- 基本的にはWindows
 - .NET対応 (VC, VB, C#で開発可能)
 - すべてのDENSOロボットがORiNで動いており, DENSO以外のロボットにも広がっている
 - FA用機器のORiNノード対応化が進んでいる
- 欧州で利用され始めており, 他の規格との連携も盛んになっている
 - EtherCAT (TwinCAT3) リアルタイムフィールドバス対応
- オープンソースコミュニティとの連携も開始
 - ROS, RTM連携

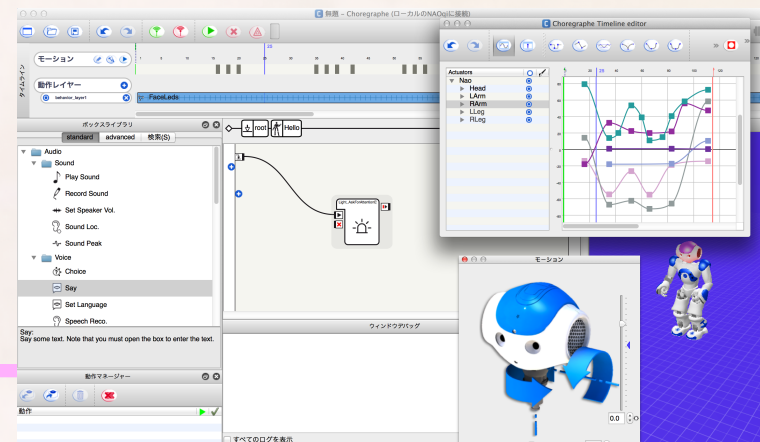
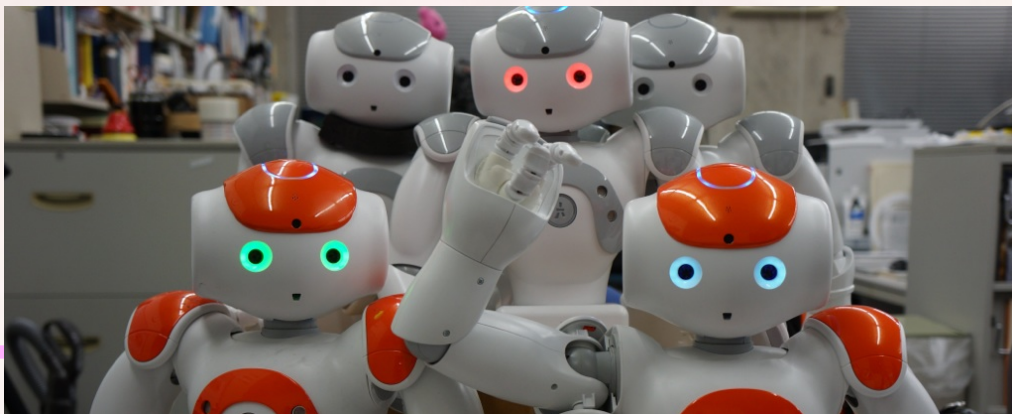


A collection of various colorful candies, including pink, green, brown, yellow, and light blue ones, some with intricate shapes like stars and flowers. The word "naoqi" is written in red in the center.

naoqi

naoqi (ナオキー)

- フランスのAldebaran Robotics社が開発
- 小型ヒューマノイドロボットNAOのためのミドルウェア
- 分散アーキテクチャ, マルチ言語, マルチプラットフォーム
- 機能単位はModuleと呼ぶ
- ModuleをつなぐChoregrapheというビジュアルプログラミング環境が充実
- 最近ではSoftBank社のPepperでも採用されたミドルウェア

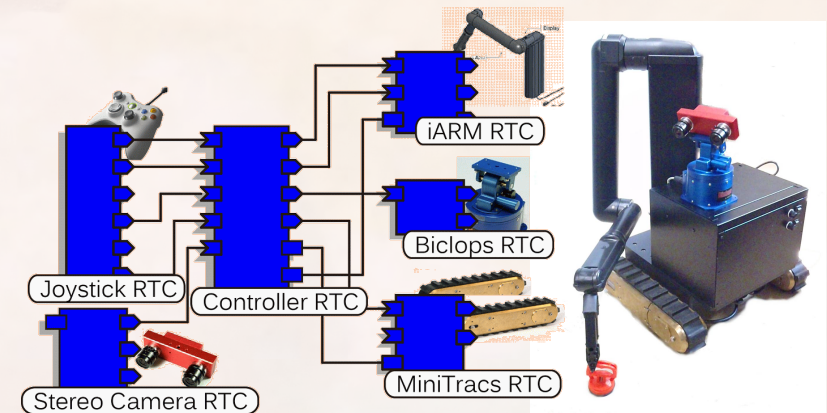
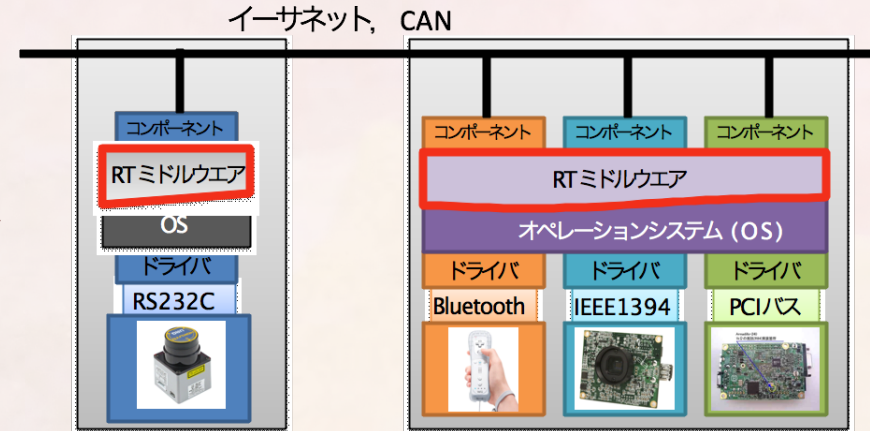




RT-middleware

RTミドルウェアとは

- RT <> Real Time. RT==Robot Technology
- ロボット技術(RT)要素のソフトウェアをモジュール化するための規格
 - RT要素 (=アクチュエータ, センサ, インターフェース, ソフトウェア) をRTコンポーネント (RTC) と呼ぶ
 - RTCの組み合わせでロボットを作る
 - RTCをどう作るか? という規格
 - RTC規格は, CORBAやUMLの規格化を行う
OMG(Object Management Group)に採択された国際標準規格
- 言語やOSなどのプラットフォームによらない形で規格を提供
 - 規格なので, 実装をプラットフォームに合わせて作れる
 - 対応OSが多い, 対応言語が多い
 - OS無しの組み込み対応が可能
 - 実時間OSにも対応
 - 異なるRTミドルウェア間でのブリッジ開発が用意



RTミドルウェアとは

- あくまでも規格なので、実装がたくさんある
 - **OpenRTM-aist** . . . 産総研が開発. CORBAを利用. C++, Python, Java対応. Win, Linux, Macに対応
 - **OpenRTM.NET** . . . 株式会社セックが開発. .NET framework対応
 - **H-RTM** . . . Honda Research Instituteが開発. OMGのRTCモデルにFSMを拡張
 - **RTM Safety** . . . 株式会社セック開発. 国際規格IEC 61508 SIL3の機能安全認証を取ったRTミドルウェア
 - **miniRTC, microRTC** . . . 組み込み用軽量RTM. CANやZigBeeでの通信に対応
 - **RTM on Android** . . . Android上で動作するRTM. OpenRTM-aistと互換
 - **RTC-CANopen** . . . 芝浦工業大学水川研究室で開発. CANopen対応機器をRTCに半自動変換
- 分散アーキテクチャ, マルチ言語, マルチプラットフォーム

ロボット用ミドルウェアに 共通していること

- ロボット自体は要素技術のシステムインテグレーション
- モジュール化・分散システム
 - ハードおよびソフトの拡張性の担保
 - マルチプラットフォーム (OS, 言語)

ハードおよびソフトの抽象化と
インターフェースの規格化・共通化

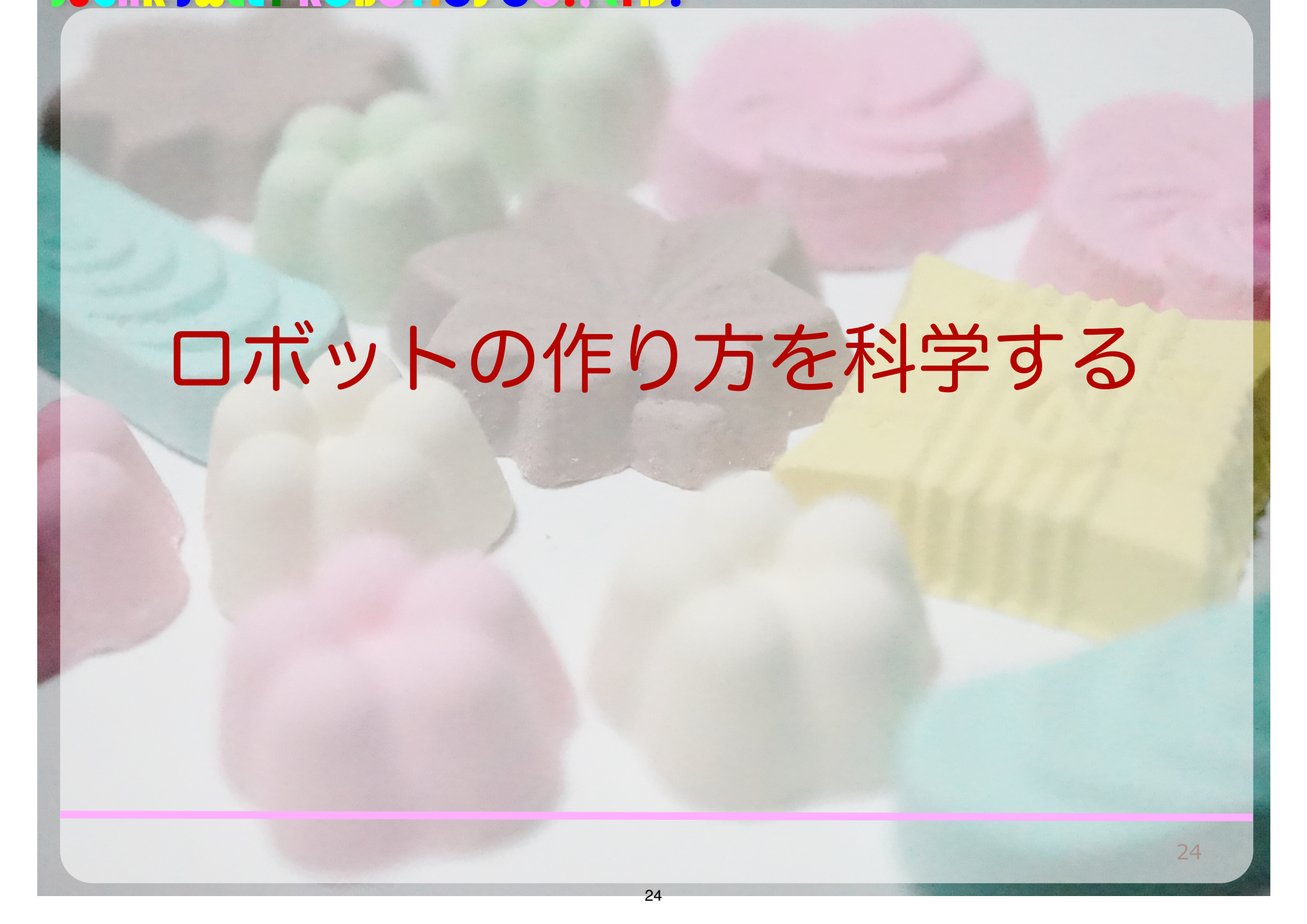
Slur側のメリット・デメリット

- すぐに試せて、試したモジュールをそのまま再利用が可能
- 基本的にデータのやり取りなので、通信の仕様（型、データの意味や単位）さえ分かれば簡単に使える
- ほとんどがフリーかつオープンソースであるため、ミドルウェア自体をカスタマイズすることも可能（ライセンスに注意）
- ネットワークを隠ぺいするので、分散システムが容易に開発できる
- ソフトウェアのオーバーヘッドは存在する
- 初期導入時の時間的コスト
- システムのチューニング作業は不可欠

モジュール開発者側としてのメリット

- ユーザ向けのソフトウェア・インターフェースが決定できる
- 同じミドルウェア利用者には簡単に試してもらえる
- ソフトウェアのドキュメントを簡潔にできる
- 新規参入しやすい

- 初期導入時のコスト
- モジュール開発自体のコスト
 - ドライバ, ライブラリに加えてミドルウェア対応モジュールの開発も必要だが, Exampleにも出来る
- 置き換えも容易 (メリット? デメリット?)



ロボットの作り方を科学する

あるロボット屋の日常

- あるロボット屋が、IT関連企業のA社の技術開発部の人に呼ばれました
- A社「ロボットを使って、サーバー内のマシンの状態監視や温度計測などがしたいなあ...」
- ロボ屋「出来ますよ、ロボットならね」
 - ロボットは、サーバー内の温度を計測できて、カメラでマシンの状態ランプの色や点灯箇所が監視できる
 - 温度センサ、カラーカメラ、パンチルト機能
 - サーバルーム内を障害物を回避しながら自律的に移動できる
 - レーザーレンジセンサ、マップ作成、自己位置同定、パスプランニングと障害物回避
 - 遠隔操作、ブラウザでインターフェースを

- 希望を要求に変える, 要求を仕様に変える.
 - より具体的な技術者が分かる言葉に直す
 - 本当にそれが企画者の希望に合っているかを検証しながら作業を進めなければ行けない
 - この段階で, つくる側の希望が入りすぎるのは良くない仕事
- 分野を超えてコミュニケーションをする技術が必要
 - コミュニケーション力という曖昧なものではない
 - 言葉を定義する能力, 言葉の定義の齟齬に気づく能力
 - モデルの力
 - 図化することで問題を整理しやすくする
 - SysML

サーバールームのマシン
を遠隔地から監視する

マシンの状態はLEDランプ
の色と点灯箇所で分かる

サーバールームは10m X
10mで、通路は1090mm幅

マシンはラックに入っ
ていて、高さ2mまで10台以
上が重なっている

通路の床はグレーチング
の箇所とコンクリートの
箇所がある

メンテナンスのためにケー
ブルや段ボールが通路に
おいてあることがある

グレーチングの箇所と他
の箇所との境目には最大
5mmの段差がある

データベース上に
位置温度計測情報を記入

データベース上に
目標移動軌跡を記入する
とロボットが追従

パンチルト機能付き
カラーカメラ

ロボットの全幅は800mm
程度

高さ2mまで500mm
於きにロボットの直上3点
を温度計測

グレーチングでの走行可

障害物回避
衝突を赤外線センサ
で非接触検知

5mmの段差乗り越え

サーバールームのマシン
を遠隔地から監視する

パンチルト機能付き
カラーカメラ



ロボットの全幅は800mm
程度

高さ2mまで500mm
於きにロボットの直上3点
を温度計測



グレーチングでの走行可

障害物回避
衝突を赤外線センサ
で非接触検知



5mmの段差乗り越え

サーバールームのマシン
を遠隔地から監視する



すでにロボットは組み合わせて作る時代

サーバールームのマシン
を遠隔地から監視する



カメラ画像取得
パンチルト動作制御
ソフトウェア

USB温度計温度取得
ソフトウェア



レーザ距離計情報取得
ソフトウェア

ロボット台車運転制御
ソフトウェア



ソフトウェアも組み合わせで作りましょう

カメラ画像取得
パンチルト動作制御
ソフトウェア

ムのマシン
監視する

USB温度計温度取得
ソフトウェア



マップ管理モジュール

マップ情報

ランドマーク
情報

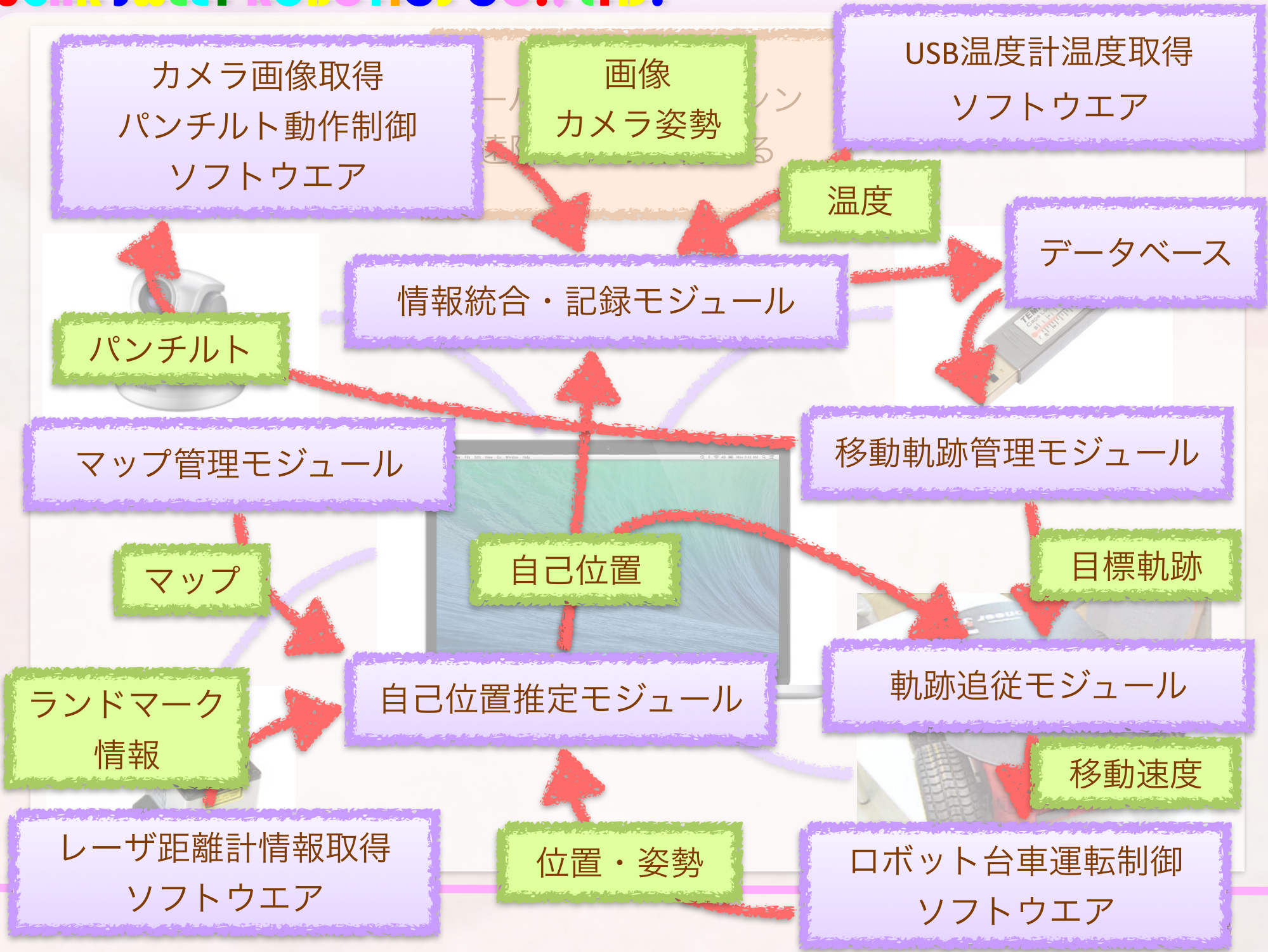
自己位置推定モジュール

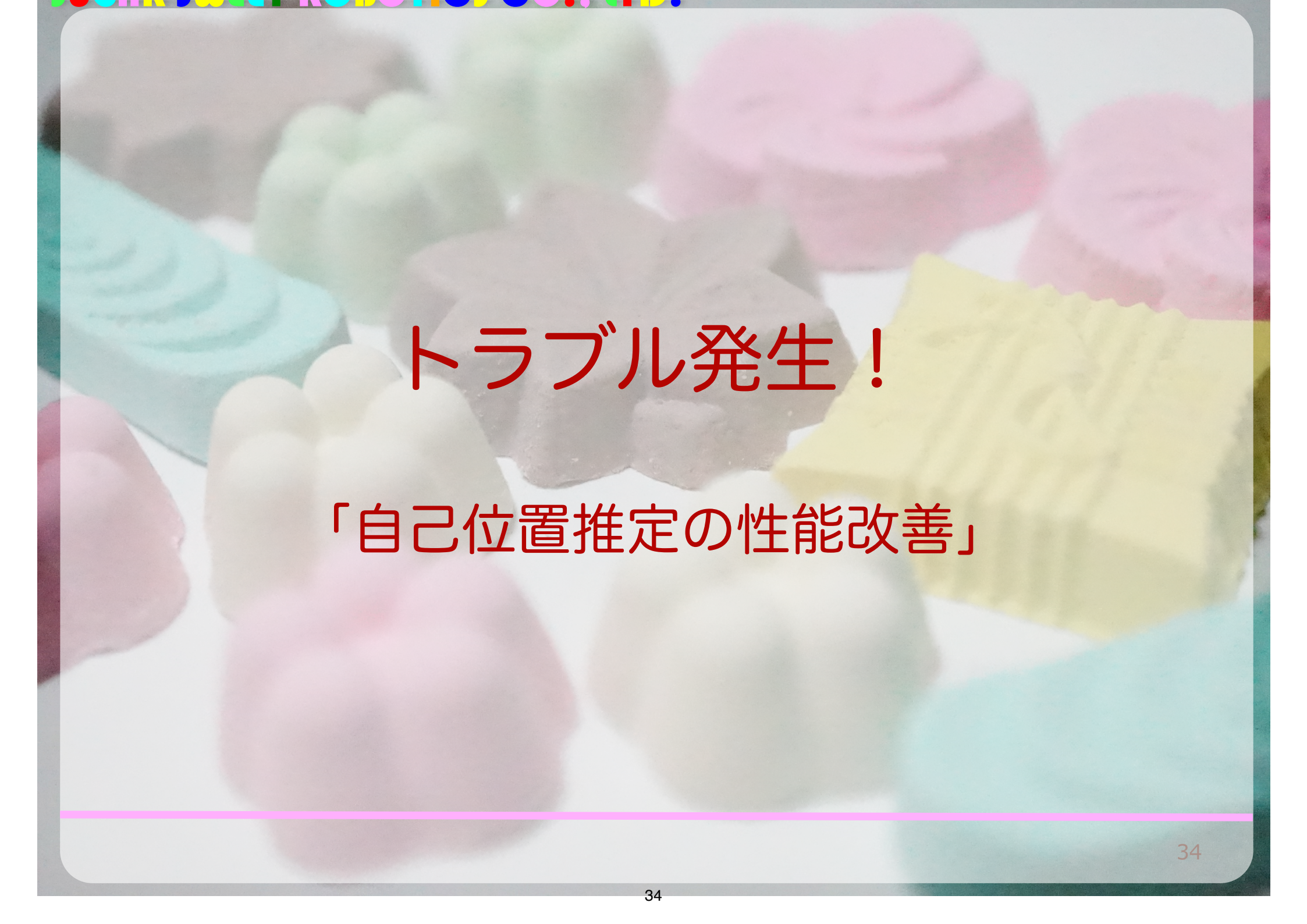
移動距離
オドメトリ



レーザ距離計情報取得
ソフトウェア

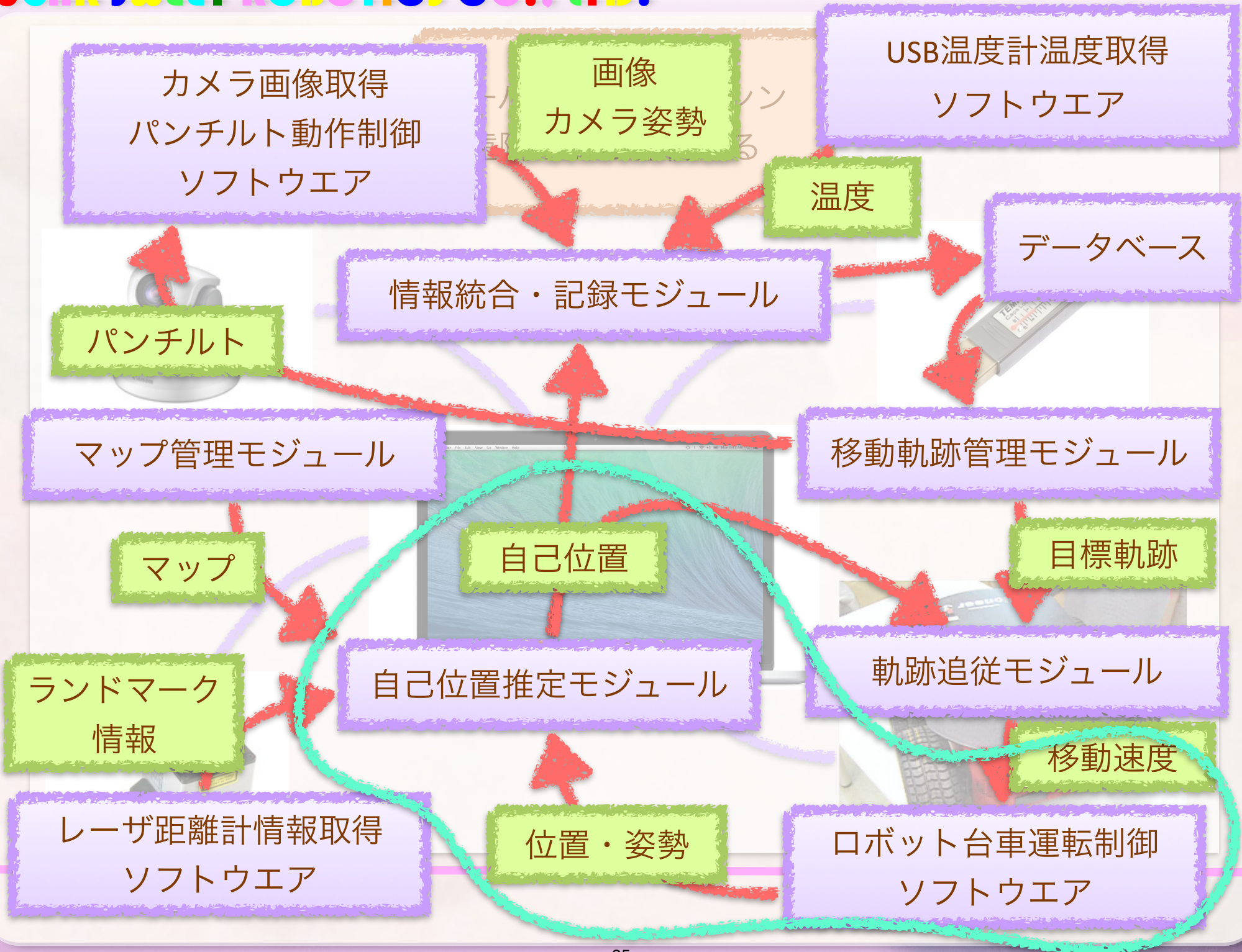
ロボット台車運転制御
ソフトウェア





トラブル発生！

「自己位置推定の性能改善」



自己位置

サーバールームのマシ
を遠隔地から監視する

軌跡追従モジュール

自己位置推定モジュール

移動速度

マップ

位置・姿勢

ロボット台車運転制御
ソフトウェア

カルマンフィルタ

位置・姿勢

位置・姿勢

加速度・角速度

姿勢予測モジュール

IMUモジュール

モジュール化のメリット

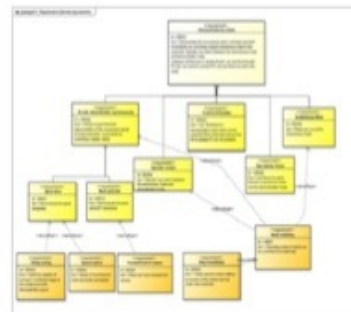
- 各モジュールの再利用性の向上
- モジュールの再利用継続による信頼性向上
- インターフェースの共通化によりモジュールの可換性向上
- システムのカスタマイズ性の向上

SysML

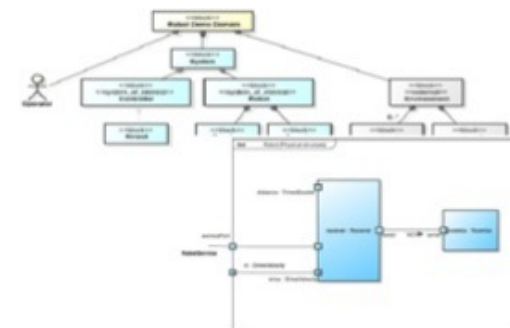
- 上記のような仕様策定過程を可視化する共通言語

ChangeVision SysMLの4本柱

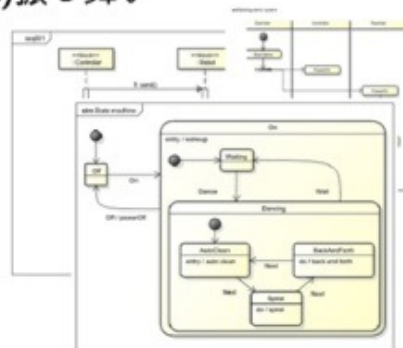
(a)要求



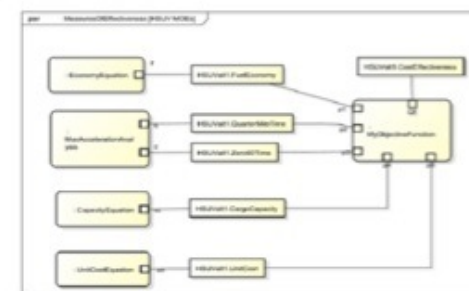
(b)構造



(c)振る舞い

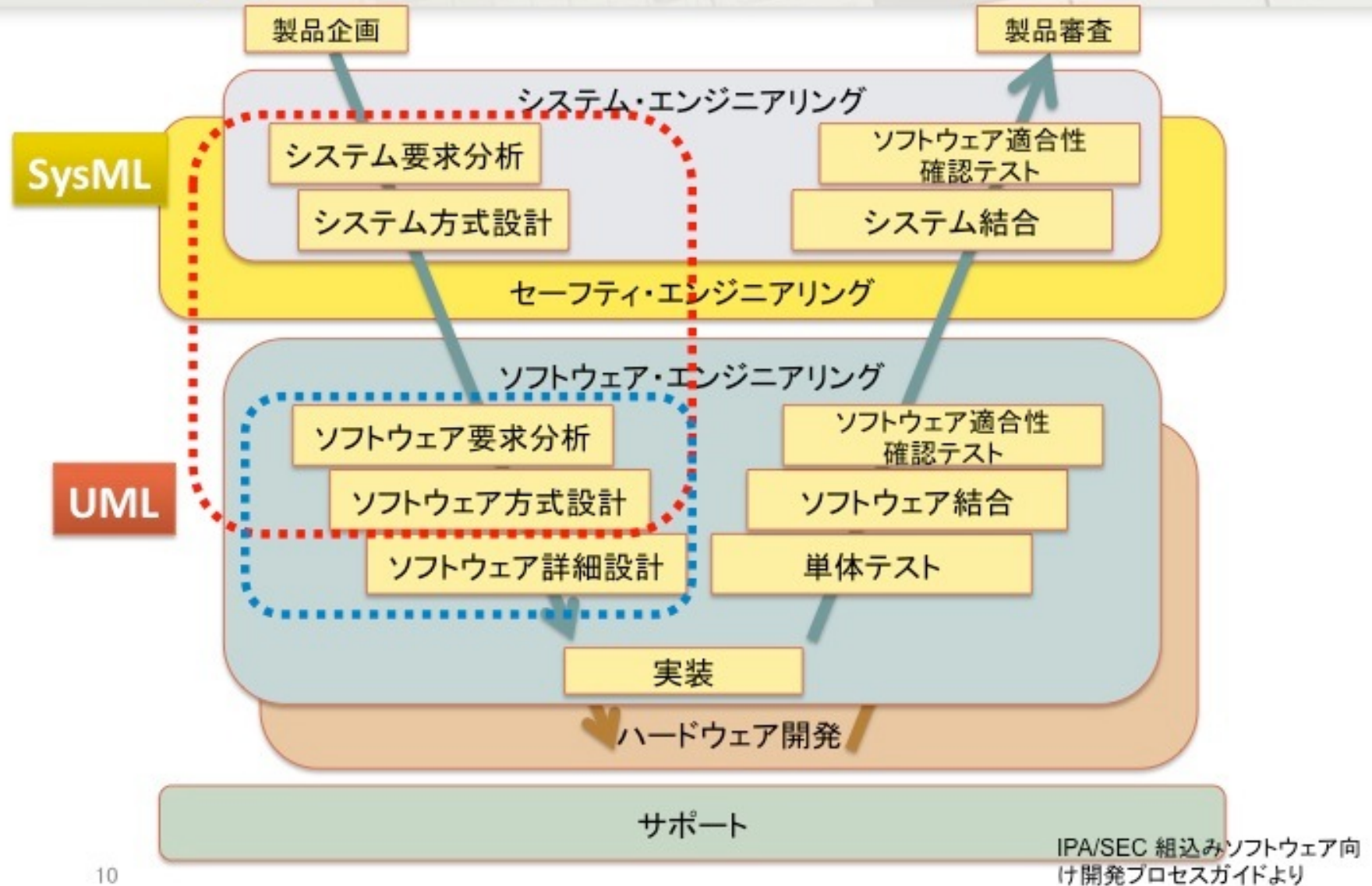


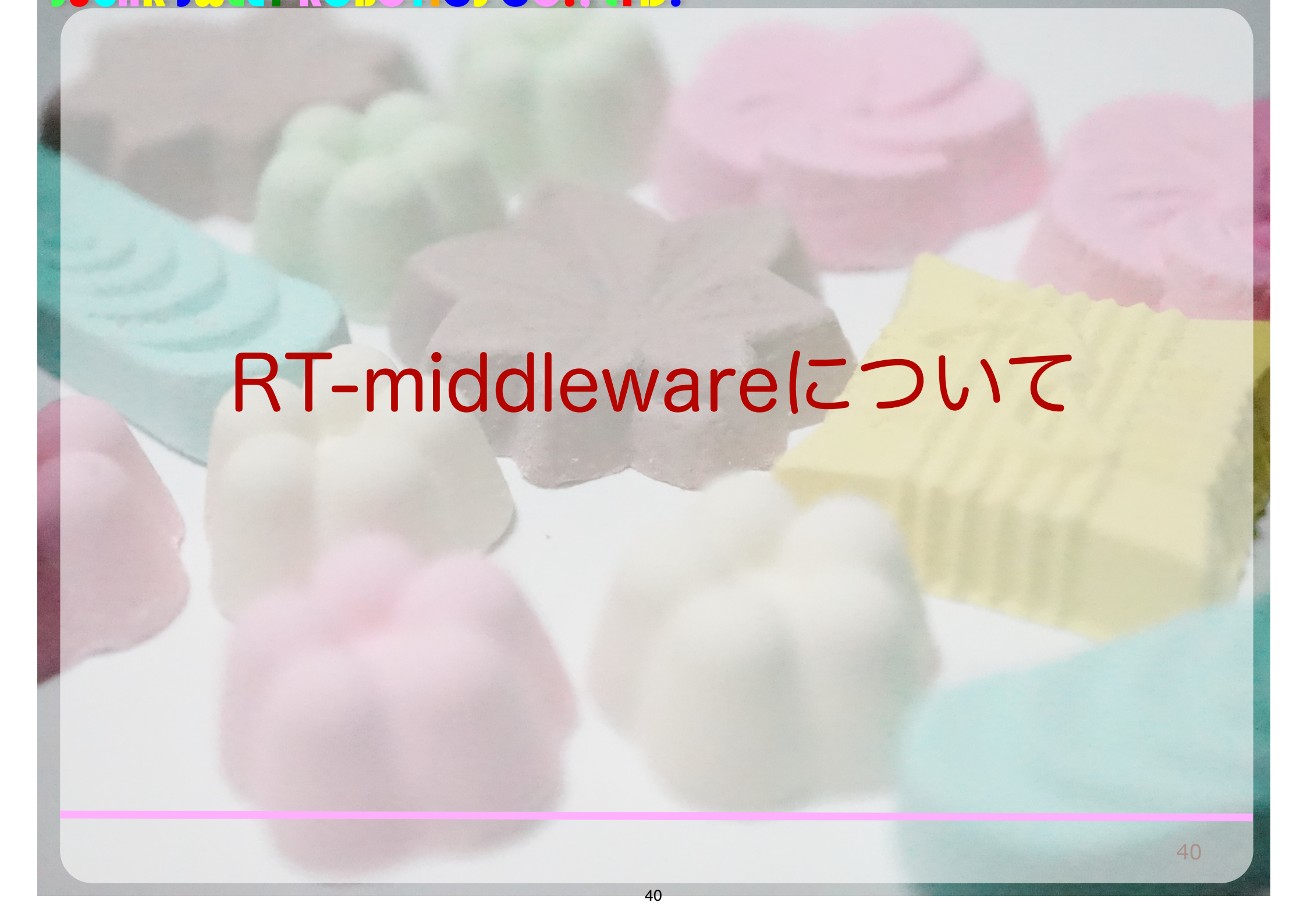
(d)パラメトリック



ロボペディアより転載

UMLとSysMLの主な対象範囲

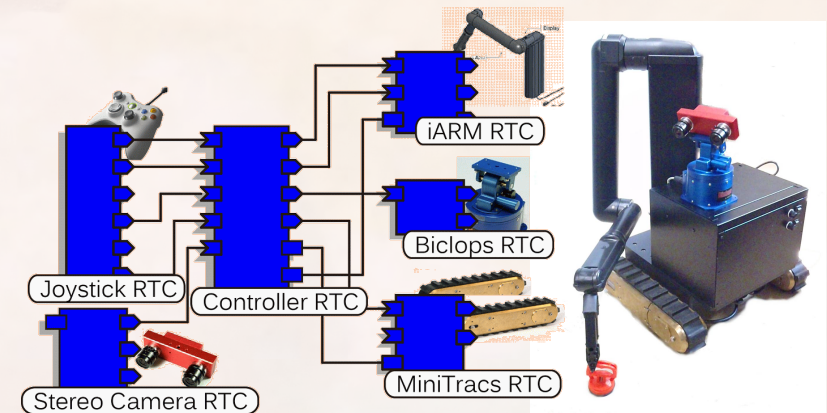




RT-middlewareについて

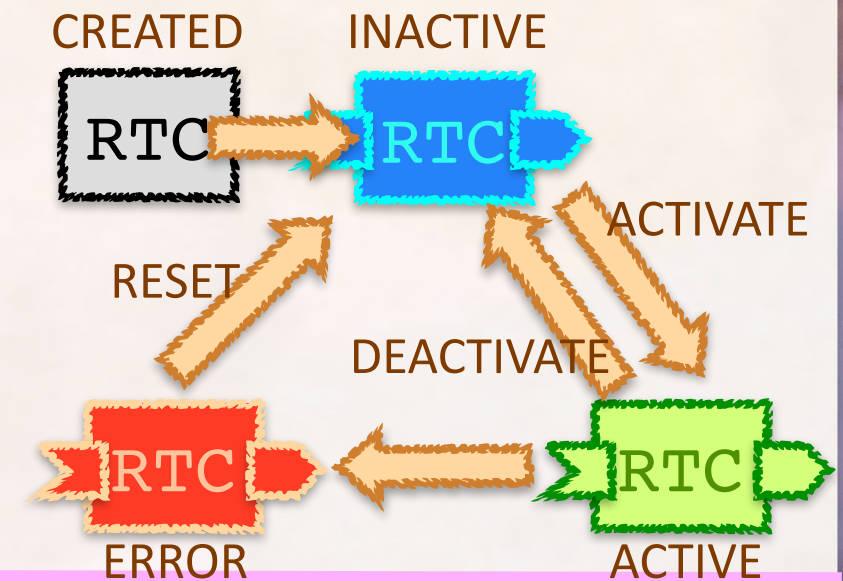
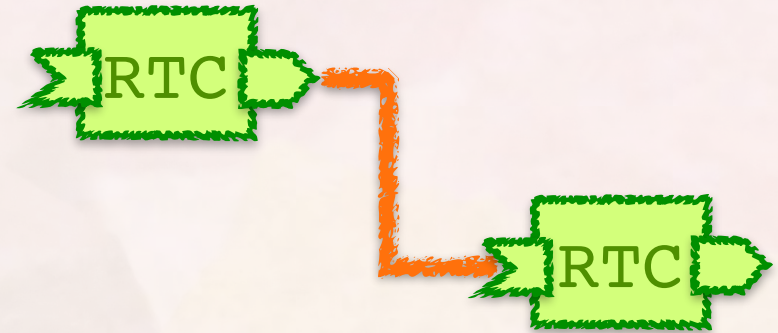
RTミドルウェアとは

- RT <> Real Time. RT==Robot Technology
- ロボット技術(RT)要素のソフトウェアをモジュール化するための規格
 - RT要素 (=アクチュエータ, センサ, インターフェース, ソフトウェア) をRTコンポーネント (RTC) と呼ぶ
 - RTCをどう作るか? という規格
 - RTC規格は, CORBAやUMLの規格化を行うOMG(Object Management Group)に採択された国際標準規格
- 言語やOSなどのプラットフォームによらない形で規格を提供
 - 規格なので, 実装をプラットフォームに合わせて作れる
 - 対応OSが多い, 対応言語が多い
 - OS無しの組み込み対応が可能
 - 実時間OSにも対応
 - 異なるRTミドルウェア間でのブリッジ開発が用意



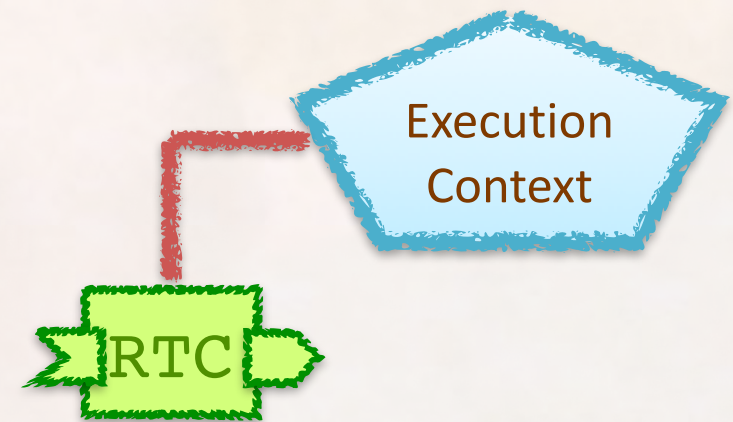
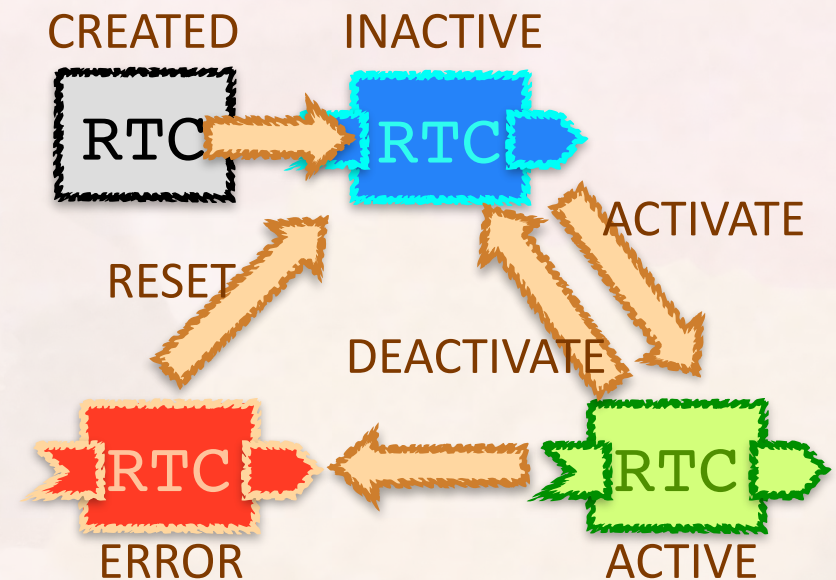
RTC規格の特徴

- RTCはポートを持つ
 - ポートはインターフェースを持ち、同じ型のインターフェースならば接続が出来て通信出来る
- RTCは状態マシンを持つ
 - RTCは作成されると、直ちにCREATE状態になり、そこからINACTIVE状態に遷移する
 - 各状態遷移に対応するコールバック関数がある
 - RTCがACTIVE状態に励起されると、周期的にon_executeを実行する
 - RTCはdeactivateされると、INACTIVE状態に戻る
- RTCはコンフィグレーションを持つ
 - RTCの起動時もしくは、実行中に変更が可能



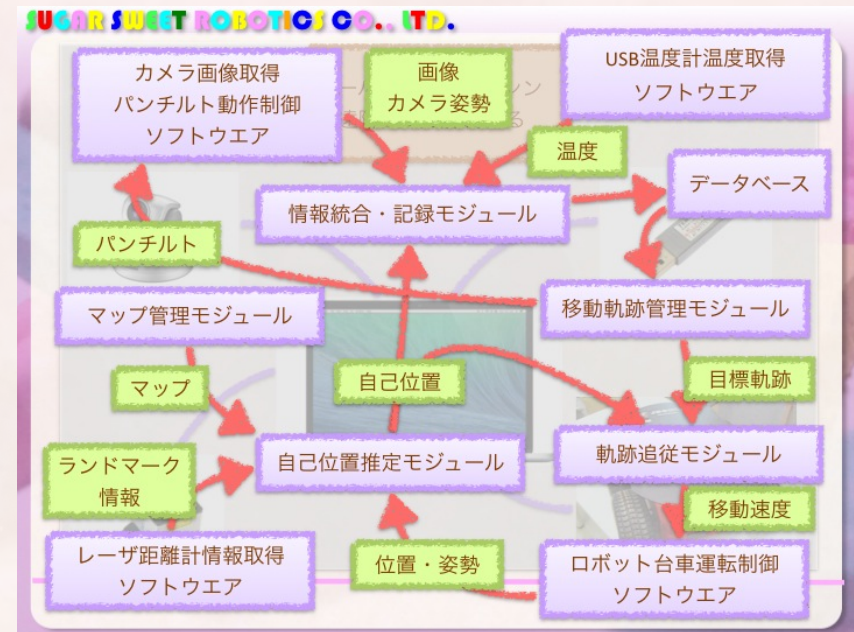
RTC規格の特徴

- RTCには状態マシンを管理する「実行コンテキスト (Execution Context)」がある
- ECには種類がある
 - 周期実行型EC
 - イベント駆動型EC
- ECは、RTCの状態がACTIVEの場合、`on_execute`を周期的に呼ぶ
- ECはRTCの起動中も変更可能
 - リアルタイムEC
 - シミュレータ同期EC



モデルベース開発

- 分散オブジェクト的なモデリングをサポート
- モデリング言語SysMLと連携して，RTCのコードやシステムプロファイルを自動生成
- ロボット開発の要求分析からサポートし，文書作成と開発の初期段階までを一本化



これもモデル



HRP-4: Kawada/AIST



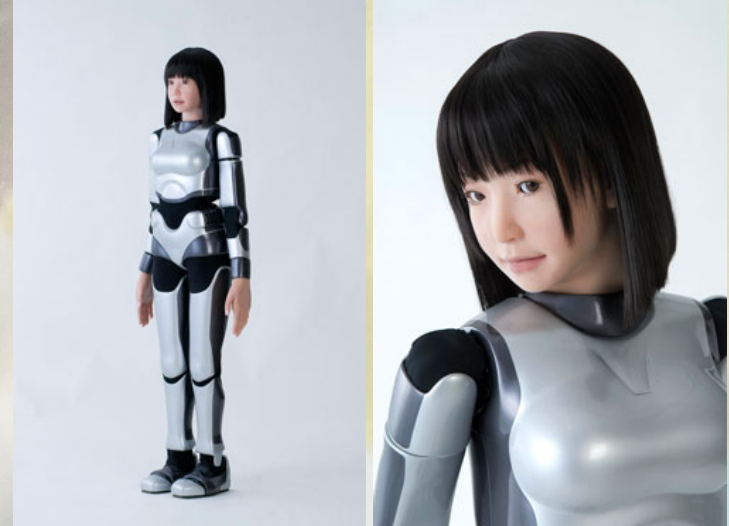
OROCHI: mayekawa



TAIZOU: General Robotics Inc.



HIRO: Kawada/GRX

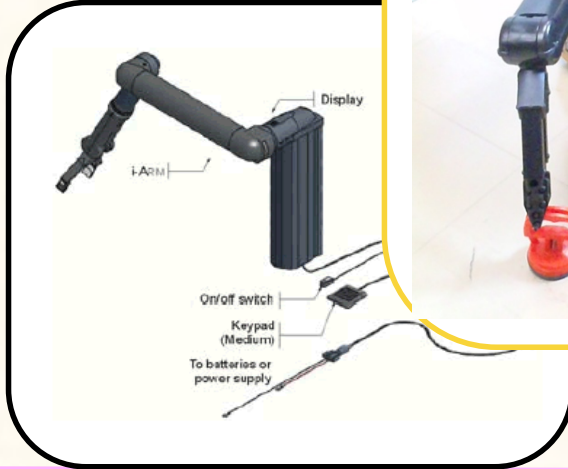
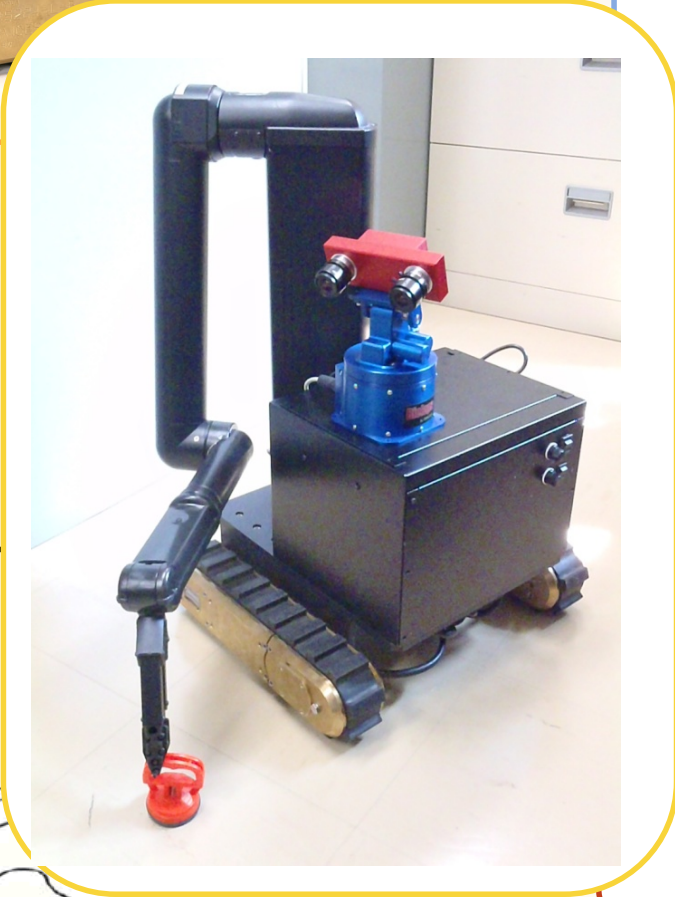


HRP-4C: Kawada/AIST

RTミドルウェア応用の具体例



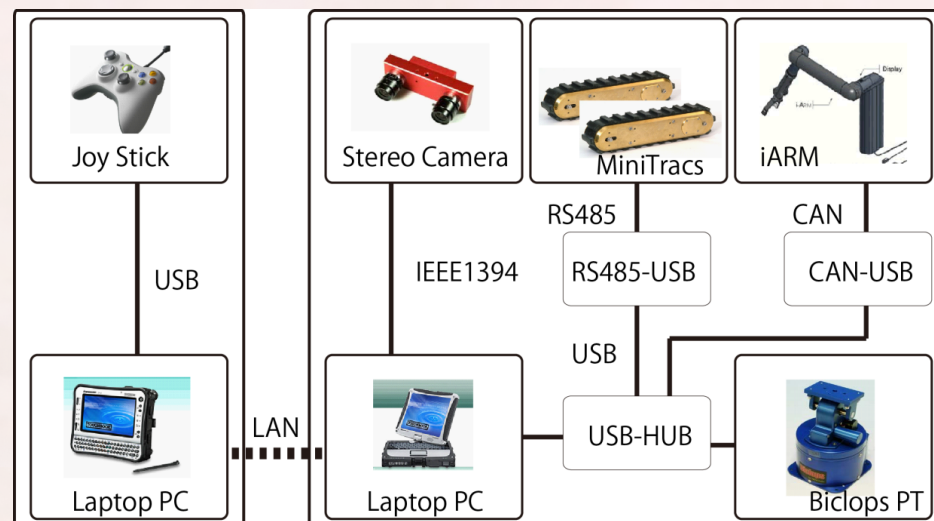
会社のロボットをRTC化してみた



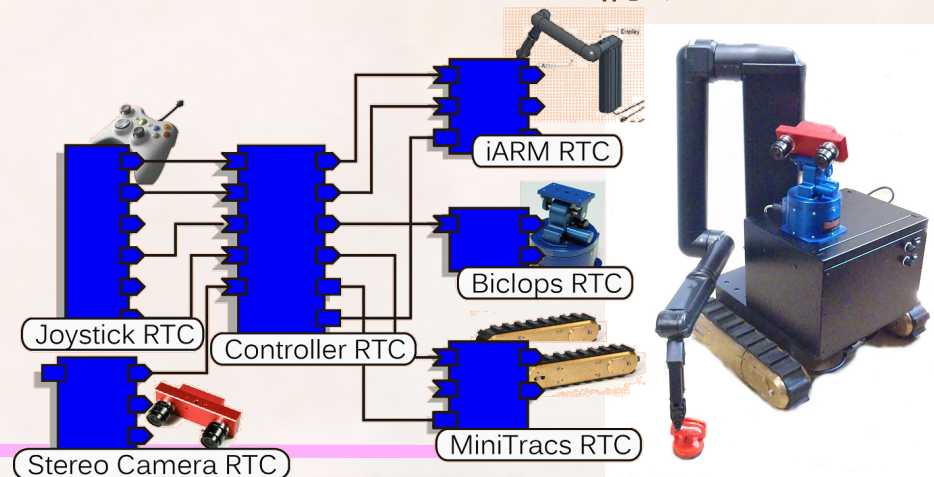
RTC化したロボット要素を結合


- 遠隔地にクローラで移動し、パンチルト動作可能なステレオカメラで確認してアームで対象物体をハンドリングする
- すべてが異なるAPIを利用
- PCとの接続方法も異なる
- RTCでラッピングして置くことで、再利用性が向上
- 遠隔操作が簡単に構築できる

ハードウェアの構成



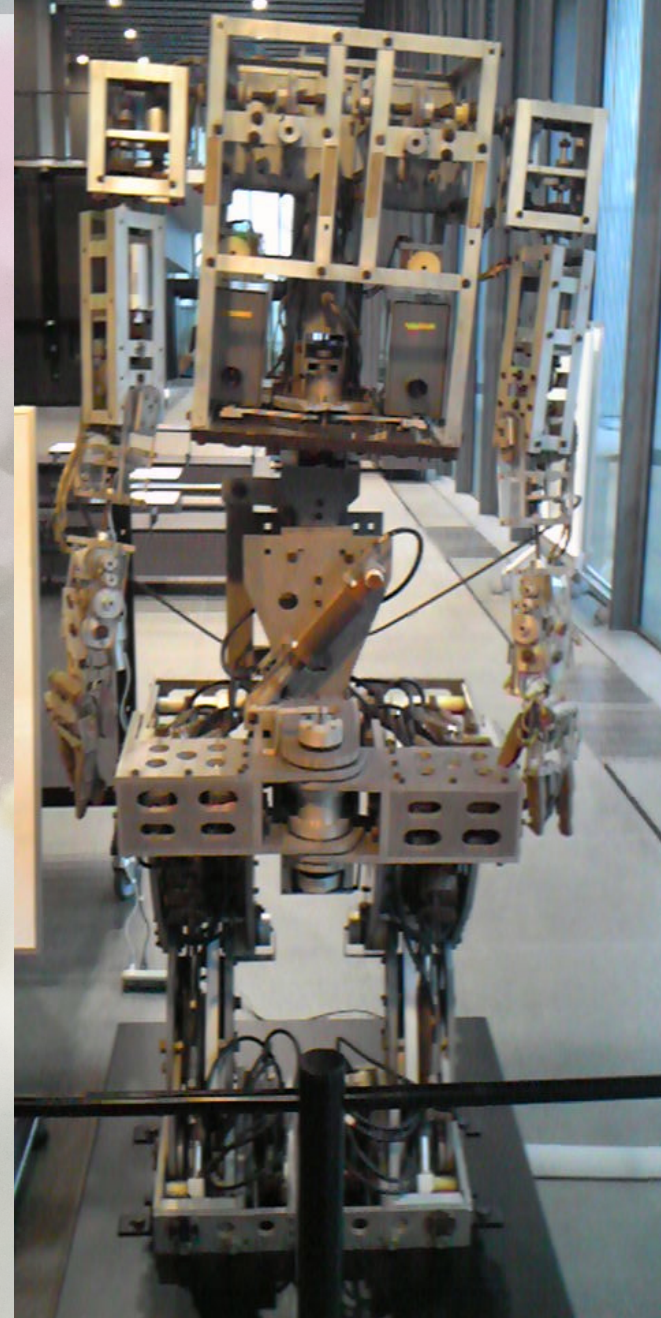
RTシステムの構成





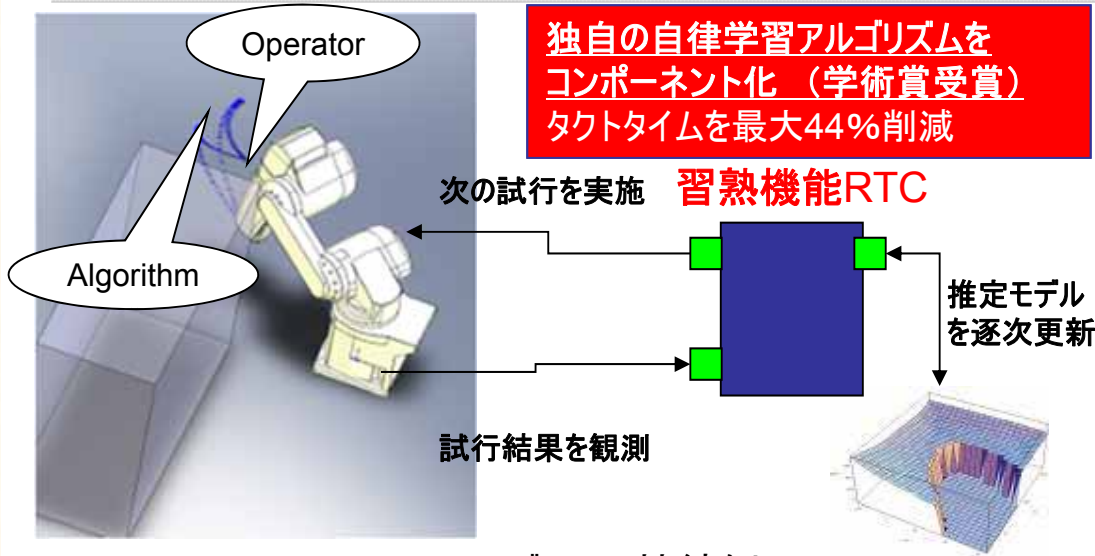
**2010年
危機管理産業展
出展ロボット**

RTミドルウェア 応用事例



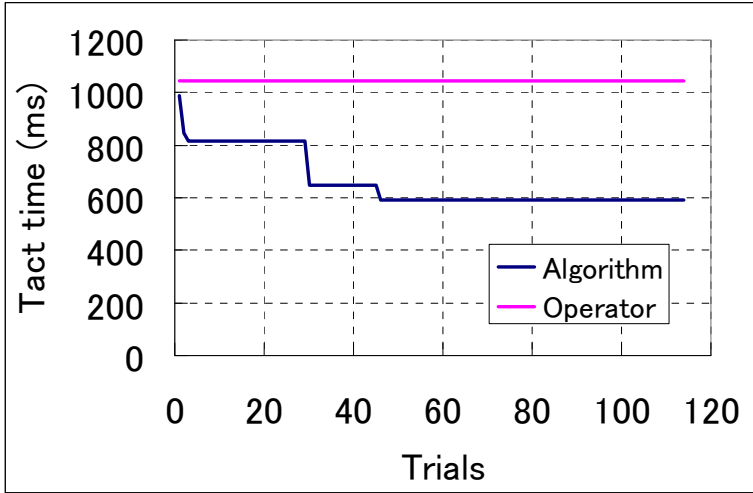


主要開発成果： 知能モジュール(1/2)



独自の自律学習アルゴリズムを
コンポーネント化 (学術賞受賞)
タクトタイムを最大44%削減

習熟機能RTC

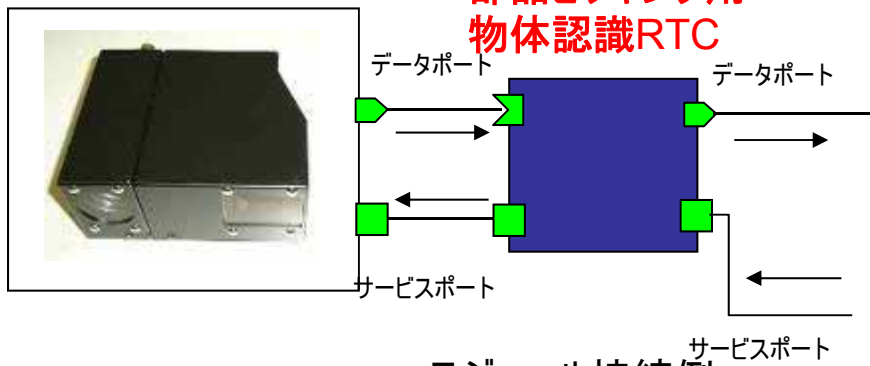


ロボット自らが動作時間を習熟する例

・再利用率が高い モジュール接続例

高精度・高速・コンパクトな3D認識システムをコンポーネント化(学術賞受賞)
13種以上の部品を, 0.4秒で認識

小型3次元センサー



ばら積み部品の把持点認識処理例

・商用性が高い モジュール接続例

主要開発成果： 知能モジュール(2/2)

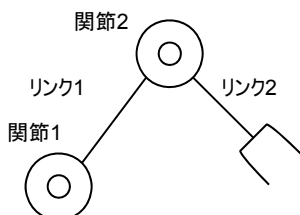
関大

時変系メカの制振制御を実現する指令値整形アルゴリズムをコンポーネント化
 静定時間1.61秒→0.92秒でタクト短縮

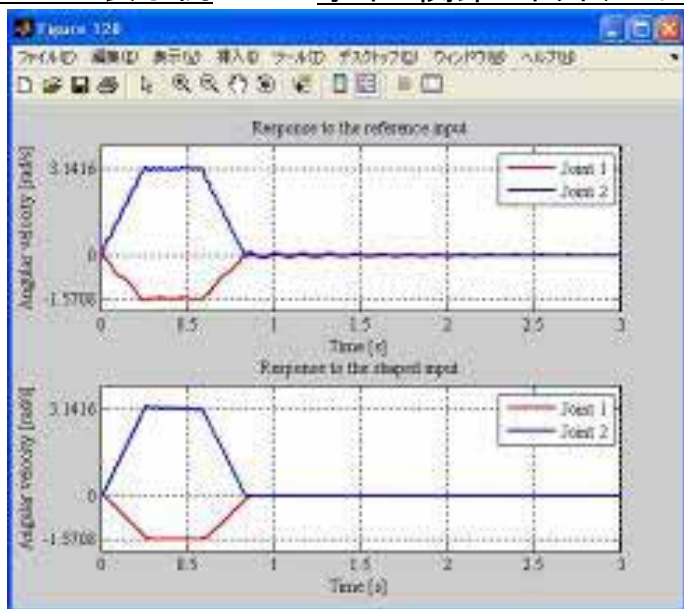
振動抑制RTC



モジュール表示例



水平2関節ロボットアーム



台形則加減速パターン入力応答(上)と整形後パターン入力応答(下)

・再利用率が高い

神大

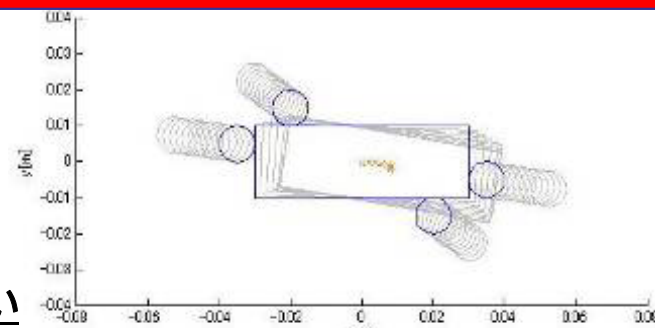
物品把持時の過渡的な物理現象を解析するアルゴリズムをコンポーネント化 (学術賞受賞)
 3本指, 4本指で3次元物体の把持現象を模擬

ハンドライブラリRTC



モジュール表示例

・再利用率が高い



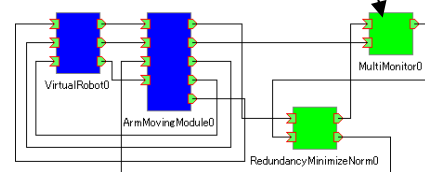
指と把持対象物の運動の軌跡のシミュレーション例

京大

インタフェース設計理論から導出されたアルゴリズムによるGUIをコンポーネント化
 教示作業時間を1/3以下に短縮

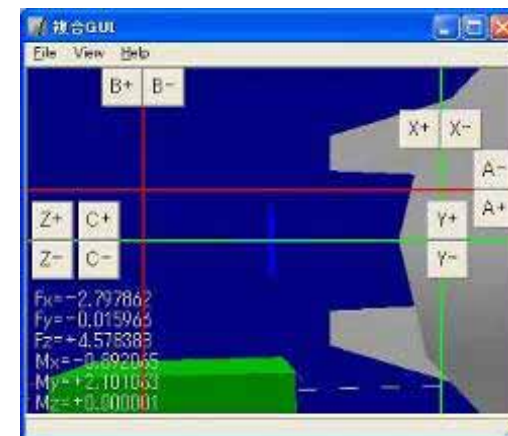
複合情報GUI RTC

本モジュール



モジュール表示例

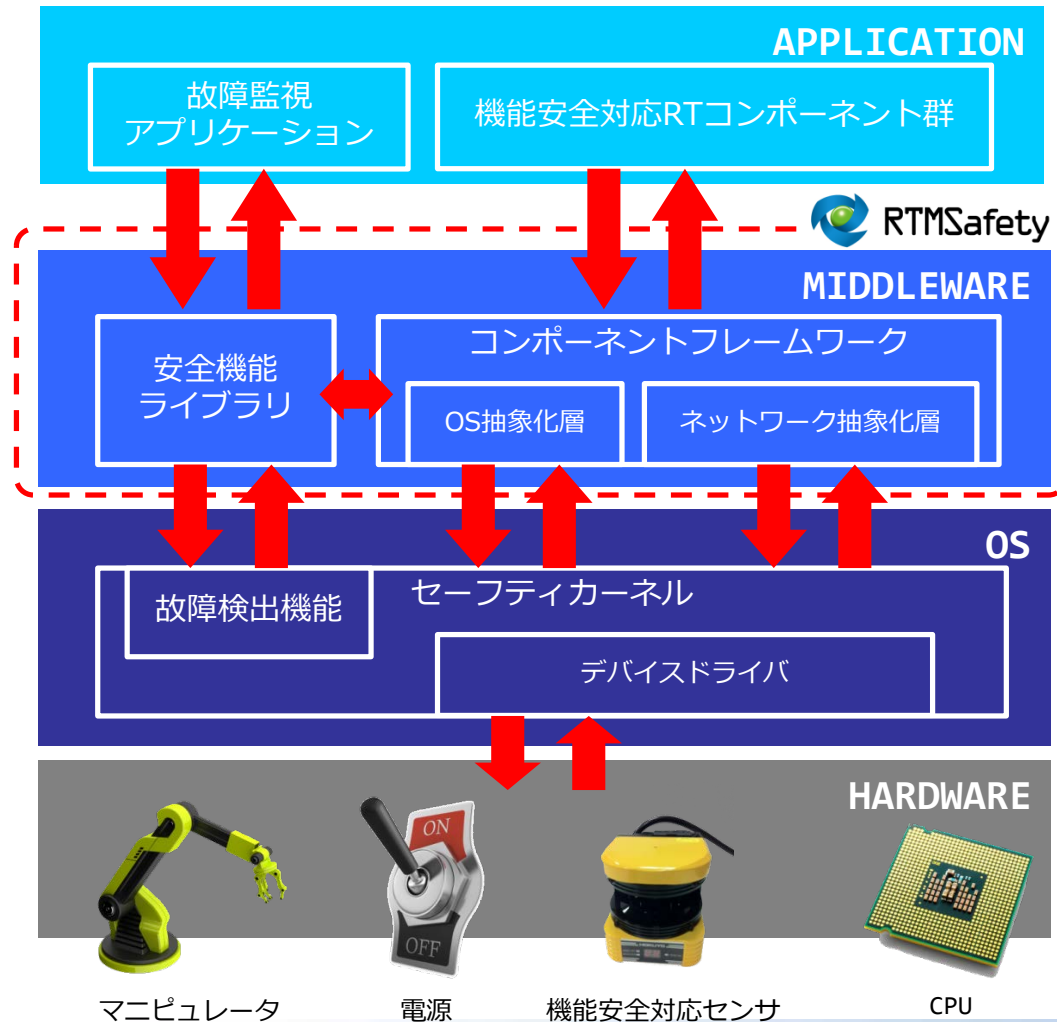
・商用性が高い



RTMSafetyの構成とメリット



RTMSafety™



•機能安全対応のRTコンポーネントを再利用することで開発を効率化

ミドルウェア層まで機能安全に対応していると...

•汎用的なアプリケーションフレームワークや共通の安全機能を利用することで開発効率の向上が見込める

•新規開発部分を少なくすることで、安全の積み上げが容易になり、機能安全認証にかかるコストも低減できる

- 株式会社セック (ロボペディアより)

ロボットシステムの活用事例 ～RT技術の住宅への適用～

インテリジェント空調
システム



パワーアシスト
ウィンドウ



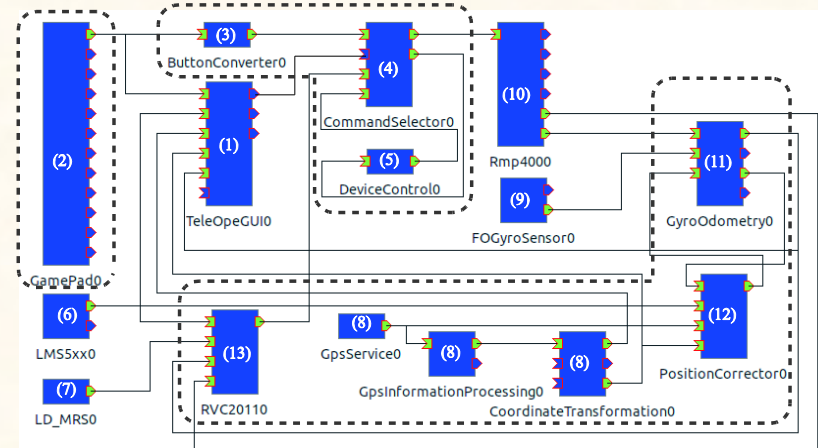
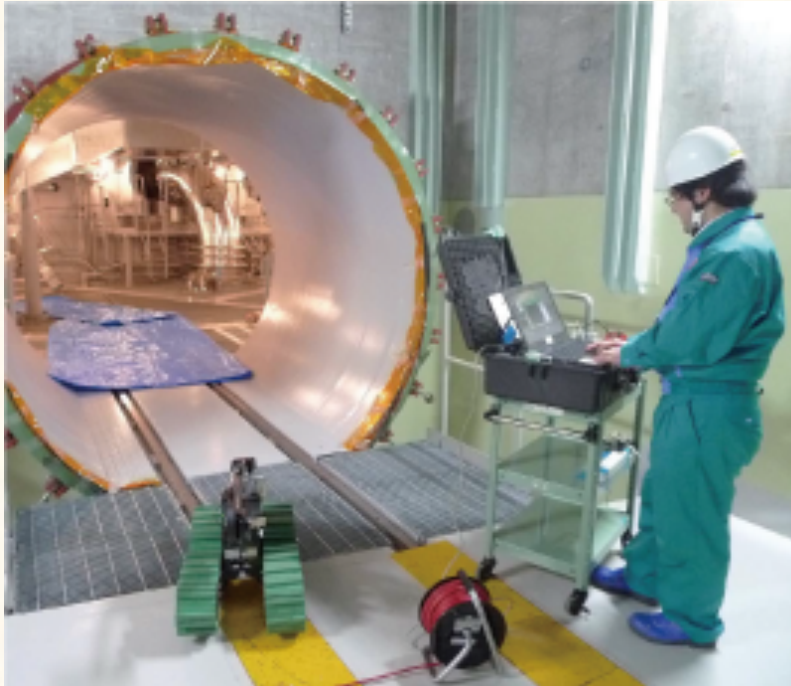
セキュリティ
システム



プラグアンド
プレイ



• 京都大学 松野研究室 (ロボペディアより)



RTミドルウェア対応させた災害対応ロボット
(上左下左: KOHGA3、上右: MATOI、下右: MATOIのRTM構成)

ROSモジュール自動相互運用RTC

ROSモジュールをRTMから利用する (東大)

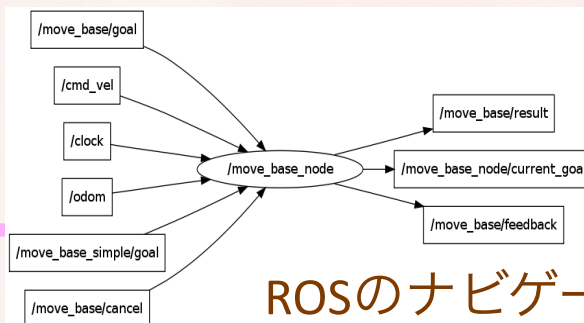
- 稼動しているROSモジュールから自動的にRTMコンポーネントを作成するプログラム

```
$ roslaunch move_base_stage_tutorial robot.launch
```

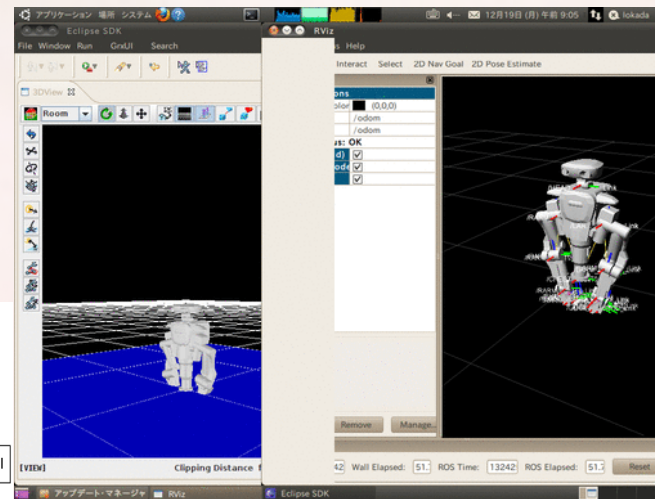
```
$ roslaunch rosnode_rtc stage_sample.launch
```

- 稼動しているROSモジュールの入出力メッセージを解析
- 対応するOpenRTM用のIDLファイルを出力
- 生成されたIDLファイルを利用してRTMコンポーネントを生成. ROSメッセージとRTMデータポートの相互変換を提供

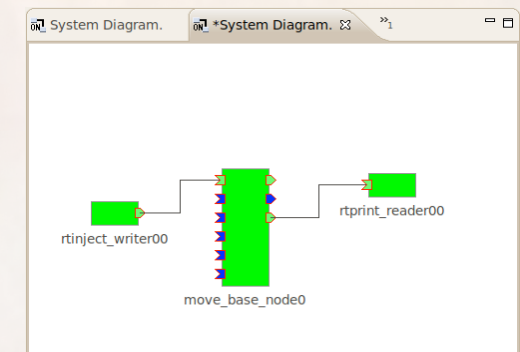
- ROSで記述された3000のパッケージを全てRTMから利用するための基盤を構築



ROSのナビゲーションノード



自動生成されたROSナビゲーション機能を提供するコンポーネント



RTM普及活動



RTM講習会

- 2013年7月24日 早稲田大学理工学部にて





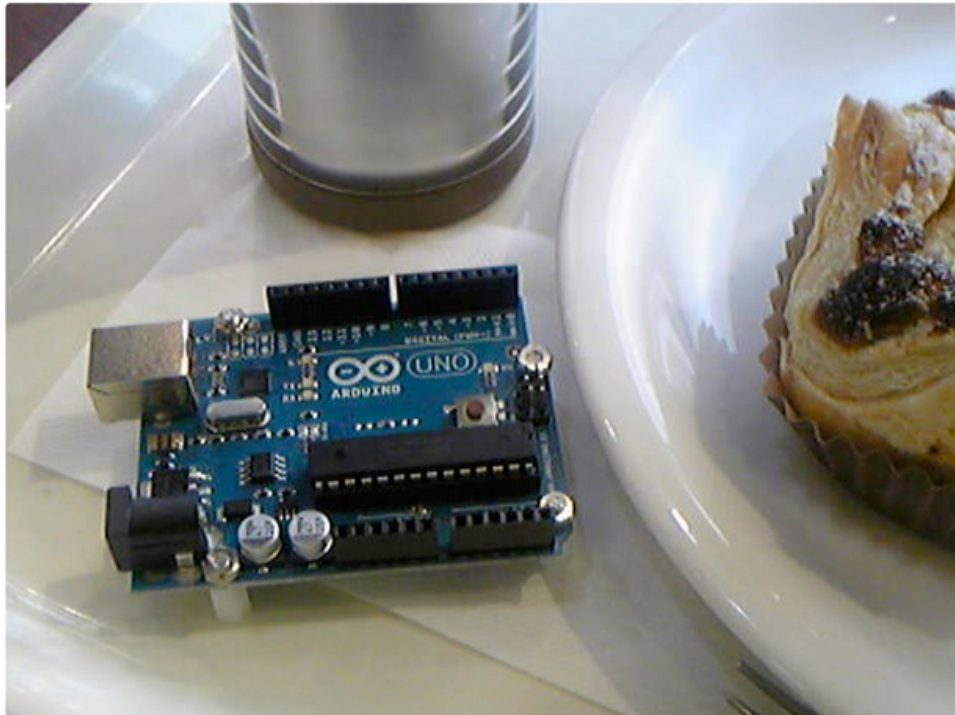
RTMコンテスト

[共同主催] [ロボットビジネス推進協議会](#)

[共同共催] (社) [計測自動制御学会](#) [システムインテグレーション部門](#)

[共同共催] (独) [産業技術総合研究所](#) [知能システム研究部門](#)

[協賛] 冠賞を提供いただく団体、個人など (詳細は[表彰\(協賛\)ページ](#)参照)



RTno (RT-middleware + arduino)

[Tweet](#) 3 [Like](#) 4 [+1](#)

組み込みマイコン用RTC開発キット「RTno（アールティーン）」を開発しました。

ROBOMECH2011での発表資料：

[原稿 \(PDF\)](#)

[ポスター \(PDF\)](#)

SI2011での発表資料：

[スライド \(SlideShare\)](#)

ニュース：

LANGUAGE

English

日本語

カテゴリー

[ROS](#)

[RTコンポーネント集](#)

[RTミドルウェア](#)

[RTM入門](#)

[RTC開発](#)

[RTM応用](#)

[インストール](#)

[ツール開発](#)

[基礎知識](#)

[YARP](#)

[ブログ](#)

[ロボット](#)

最近の投稿

[YARPとiCub SIMをUbuntu 12.04](#)

[\(presice\) にインストール](#)

[RTミドルウェア サマーキャンプの感想](#)

[ROBOTECH2012でのRTミドルウェア](#)

[講習会の感想](#)

移動ロボットのソフトウェア開発のための屋内環境シミュレータRTC

Submitted by shigemura on Wed, 2011-10-26 17:15

概要

環境シミュレータは屋内環境、そこで行動するロボット、そして環境内の人の動きを再現します。このシミュレータはセンサ出力や人物検出結果、地図などさまざまな情報を出力します。また、ロボット制御命令（指定する速度）を入力することによってシミュレータ上の仮想移動ロボットを制御することができます。

このRTコンポーネントを使用することによって実際に実験することが難しい多数の人が行動する環境での試験を行うことができるようになります。また開発したアルゴリズムのそのような環境での動作を安全に検証することができます。

図の橙色の丸がロボットを、緑の丸が人を表しています。

OS: Windows

言語: C++

OpenRTM ver.: 1.0

Average:

☆☆☆☆☆

Your rating: None Average: 5 (1 vote)



RT-Component RTM contest 2011 NEDO知能化プロジェクト 移動知能コンポーネント

[Login or register to post comments](#) [Read more](#)

RTno (汎用マイコンボードarduinoでRTコンポーネント対応デバイスを作るためのライブラリ)

Submitted by ysuga on Sat, 2011-07-30 11:01

RTno (アールティーノ) は、RTコンポーネントとarduino (もしくはarduinoコンパチのデバイス) との間の通信を簡便化します。

下図をご覧ください。RTnoパッケージはarduino用のライブラリ「RTno」と、それと通信する「RTnoProxy」というRTコンポーネントのセットです。もし、arduinoのテンプレートに従ってプログラムをすれば、RTnoはRTnoProxyを通して、他のRTコンポーネントと通信出来ます。

OS: Windows, Linux, MacOSX

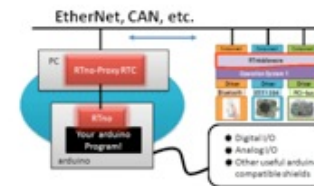
言語: C++

OpenRTM ver.: 1.0, 1.1

Average:

☆☆☆☆☆

Your rating: None Average: 5 (1 vote)



system development ツール RTM contest 2011 RTno

[Login or register to post comments](#)

RTミドルウェアの学習を目的とした安価で入手容易なロボット上での実行環境

Submitted by s08tm049 on Wed, 2011-10-26 17:19

概要

- RTミドルウェアの導入教育を目的とした安価&入手容易&多人数向け&初心者向けのRTミドルウェア学習環境
- 開発した移動ロボットは自律移動を実現
- 移動能力に関しては、リファレンスハードウェアを使用した場合と同等の学習環境を実現

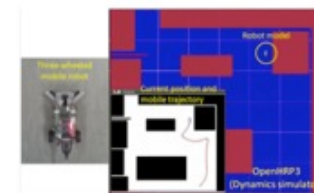
OS: Linux

言語: C++

OpenRTM ver.: 1.0

Average:

☆☆☆☆☆



計測自動制御学会学会RTミドルウェア賞（最優秀賞）（副賞10万円）（1件）

奨励賞（賞品協賛）（提供製品）（2件）

- maxon X drives 賞【提供：[マクソンジャパン株式会社](#)】
- ウィン電子工業賞【提供：[有限会社ウィン電子工業](#)】

奨励賞（団体協賛）（副賞2万円）（12件）

- チェンジビジョン賞【提供：[株式会社チェンジビジョン](#)】
- トヨタ自動車（株）パートナーロボット賞【提供：[トヨタ自動車株式会社](#)】
- NTTデータ賞【提供：[株式会社NTTデータ](#)】
- グローバルアシスト賞【提供：[株式会社グローバルアシスト](#)】
- RTイノベーション賞【提供：[株式会社グローバルアシスト](#)】
- ヴィストン ロボットショップ賞【提供：[ヴィストン株式会社](#)】
- ロボットサービスイニシアチブ(RSi)賞【提供：[ロボットサービスイニシアチブ\(RSi\)](#)】
- アドイン賞【提供：[株式会社アドイン研究所](#)】
- PiRT-Unit賞【提供：[有限会社ウィン電子工業](#)】
- SUGAR SWEET ROBOTICS賞【提供：[株式会社SUGAR SWEET ROBOTICS](#)】
- ベストコンセプト賞【提供：[ロボットビジネス推進協議会](#)】
- 日本ロボット工業会賞【提供：[一般社団法人日本ロボット工業会](#)】



2013年

RTMサマーキャンプ

2011年8月29日 (月) ~9月2日 (金)

2012年7月30日 (月) ~8月3日 (金)

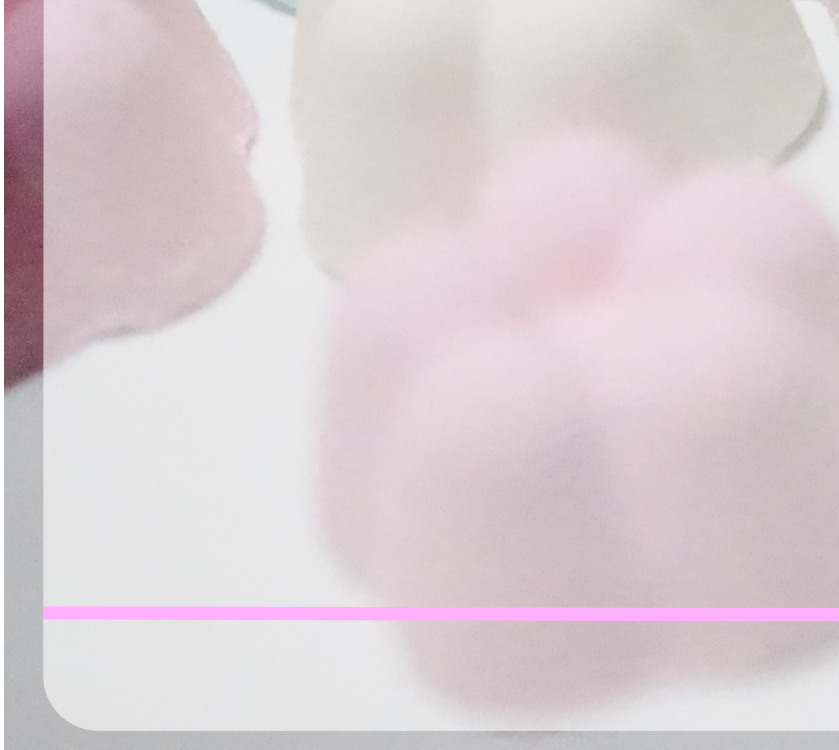
2012年7月29日



2011年



2012年



グループワーク化によるメリット

- 上流からのシステム設計の教育的意義
- システムを分割して開発
 - RTMの意義を体感
- 責任感の創出



「ロボペディア」

The screenshot shows the Robopedia website interface. At the top, there's a navigation menu with links for 'トップページ', 'ニュース', 'ソリューション', 'シーズ技術', 'セミナー・ワークショップ', 'お問い合わせ', and '索引'. Below the menu is a search bar and a calendar for September 2013. The main content area includes a 'ランダムピックアップ' section with several featured articles, such as 'RTM・ROSの相互運用技術' and '災害対応無人化システム研究開発プロジェクト'. The footer contains a link to the '旧ロボペディアサイト'.

- 災害対応ロボットや，ロボット開発を行う地域的取り組み，RTミドルウェアなどの基盤技術について紹介
- 2012-2013年のNEDO特別講座（ロボット技術経営）で実施
- 続きはWebで。

共通インターフェース仕様の策定

同等の機能を有するロボット要素間のインターフェースの共通化



OpenRTM-aist
The power to connect

ホーム ダウンロード ドキュメント コミュニティ 研究・開発 プロジェクト ハードウェア

Google Translate
言語を選択
Powered by Google 翻訳

ナビゲーション

- ホーム
- ダウンロード
- ドキュメント
- コミュニティ
- 研究・開発
- プロジェクト
- ハードウェア
- Pukiwikiマニュアル

リンク

- OpenHRP3
動力学シミュレータ
- OpenHRI
対話制御コンポーネント群
- OpenRTP
統合開発プラットフォーム
- OpenINVENT
移動ロボット用コンポーネント群

Extended RT-Middleware

ホーム >> プロジェクト >> 仕様・文書等 >> 共通 I / F 仕様書について

共通 I / F 仕様書について

共通 I / F 仕様書について

投稿者: ogasawara 投稿日時: 金, 2011-08-05 17:01

いいね! 「いいね!」と言っている友達はまだいません。

問合せ (メールアドレス): ogasawar.R80@live.jp

はじめに

近年、ロボットの開発を効率化するためコンポーネントベースのミドルウェア開発が盛んになっている。コンポーネントベースのミドルウェア開発において、インターフェースの共通化は、コンポーネントの相互接続性や相互運用性を確保するうえで非常に重要である。このような背景に基づき、各機能単位に、その機能に関わるコンポーネントの共通規格を定義する。

移動ロボット

概要)

移動ロボットを構成する標準的な機能モジュール群についてモジュール構成と各モジュールの規格を定義する。

- ・共通 I / F 仕様書
- ・標準モジュール
- ・情報共有サイト

仕様準拠モジュール

- 産総研が取りまとめた基本的RTC群 (移動機能)
- 人追従機能RTCモジュール群 (オープンソース)
- RTC-CANopenリファレンスロボット
- リファレンスハードウェア移動制御モジュール



OpenRTM-aist入門

インストール (Windows)

- まず, UACをOFFにしましょう.
- OpenRTM-aist バージョン1.1 C++言語版 (32bit版)
 - <http://www.openrtm.org/openrtm/ja/content/110-release>
 - JDK7 . . . [jdk-7u4-windows-i586.exe](#) (32bit版)
 - Python 2.6 . . . [python-2.6.6.msi](#) (32bit)
 - PyYAML . . . [PyYAML-3.10.win32-py2.6.exe](#)
 - OpenRTM-aist . . . [OpenRTM-aist-1.1.0-RELEASE_vc10.msi](#) (32bit)
 - GUIツール入りeclipse . . . [eclipse342_rtmtools110-rc2_win32_ja.zip](#)
- Doxygen
 - <http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc>
 - Doxygen win32 . . . [doxygen-1.8.1.1-setup.exe](#)
- CMake 2.8
 - <http://www.cmake.org/cmake/resources/software.html>
 - CMake 2.8 . . . [cmake-2.8.8-win32-x86.exe](#)
- Visual C++ 2010 (必ず32bit版)
 - 必ずSP1を入れること

動作確認 (Windows)

- ネームサービスの起動
 - 「スタートメニュー」 > 「OpenRTM-aist 1.1」 > 「C++」 > 「tools」 > 「Start Naming Service」
- RTCの起動 (ConsoleIn, ConsoleOut)
 - 「スタートメニュー」 > 「OpenRTM-aist 1.1」 > 「C++」 > 「components」 > 「examples」 > 「ConsleInComp.exe」 もしくは 「ConsoleOutComp.exe」
- GUIツールの起動
 - eclipse***.zipを展開
 - Eclipseフォルダ内のeclipse.exeを起動
 - ワークスペースはデフォルトでOK
 - 通常は, /Users/\$USER_NAME/workspace

Macの場合

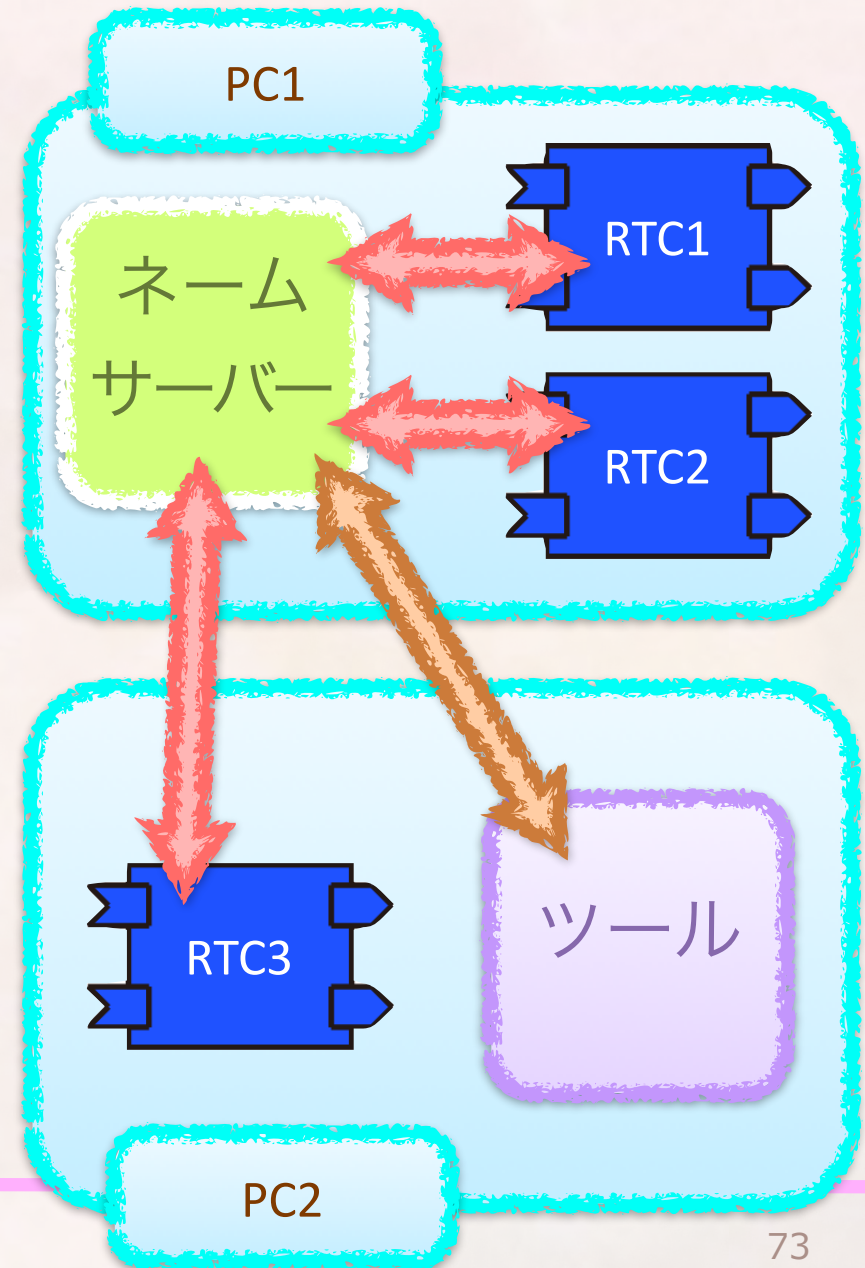
- OSXは10.9.3+Xcodeは5.1 (これ以外の環境では, 自分でビルドする必要があります)
- C++版インストーラ
 - <http://sugarsweetrobotics.com/pub/Darwin/OpenRTM-aist/cxx/1.1/OpenRTM-aist-1.1-cxx-osx-10.9.dmg>
 - \$HOME/.bash_profileに以下の記述を追加する必要がある.
 - RTM_ROOT=/usr/local/include/openrtm-1.1
 - PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
- これ以外に, 以下のツールが必要です (自分でビルドする場合も必要)
 - Xcodeコマンドラインツール
 - <https://daw.apple.com/cgi-bin/WebObjects/DSEAuthWeb.woa/wa/login?&appldKey=891bd3417a7776362562d2197f89480a8547b108fd934911bcbea0110d07f757&path=%2F%2Fdownloads%2Findex.action> (Apple Developerにサインイン必要)
 - cmake <http://www.cmake.org/files/v2.8/cmake-2.8.10.2-Darwin64-universal.dmg>
 - doxygen <http://sugarsweetrobotics.com/pub/Darwin/doxygen/Doxygen-1.8.3.1.dmg>
 - pyyaml <http://sugarsweetrobotics.com/pub/Darwin/libs/PyYAML-3.10.tar.gz>
 - pkg-config <http://sugarsweetrobotics.com/pub/Darwin/pkgconfig/PkgConfig.dmg>
 - Sun JDK (ターミナルでjavaと打つと, インストールされていなければ公式サイトに飛びます)
- また, ツールとして, RTシステム全部入りのEclipse (Mac版) をOpenRTM-aistのページからダウンロードする必要があります.
 - http://openrtm.org/pub/openrtp/packages/1.1.0.rc4v20130216/eclipse381-openrtp110rc4v20130216-macosx-cocoa-x86_64.tar.gz

Macの場合

- ターミナルからNameServer起動
 - `$ rtm-naming`
- ターミナルからConsoleIn起動
 - `$ cd /usr/local/share/openrtm-1.1/examples/`
 - `$./ConsoleInComp`
- ターミナルからConsoleOut起動
 - `$ cd /usr/local/share/openrtm-1.1/examples/`
 - `$./ConsoleOutComp`
- ターミナルからEclipseを起動
 - Downloadsにダウンロードして、そのまま展開してeclipseというディレクトリが出来た、と仮定します.
 - `$ cd $HOME/Downloads/eclipse`
 - `$ cd Eclipse.app/Contents/MacOS`
 - `$./eclipse`

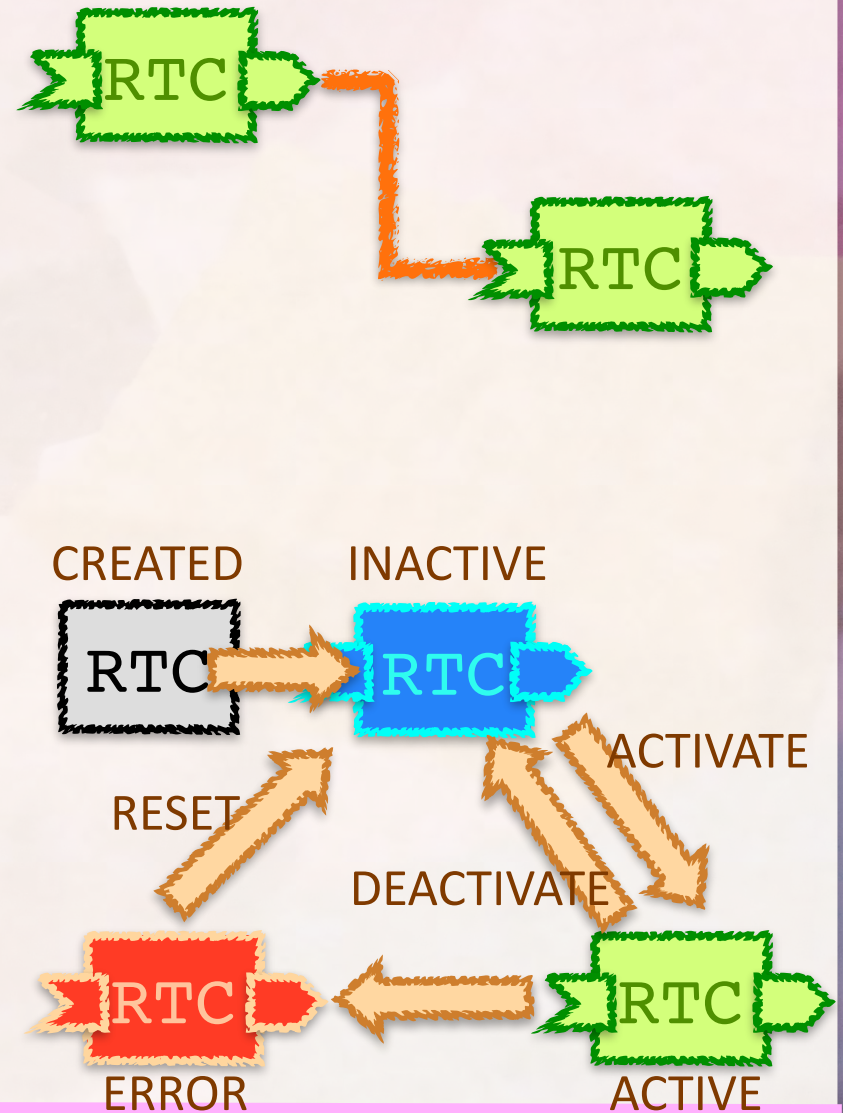
ネームサービスとは

- 実行中のRTCの管理
 - OpenRTM-aistを使う場合, RTCを使ったシステムでは最低1つ必要
 - 異なるホストのRTCも登録可能
 - 複数のネームサービスを併用可能



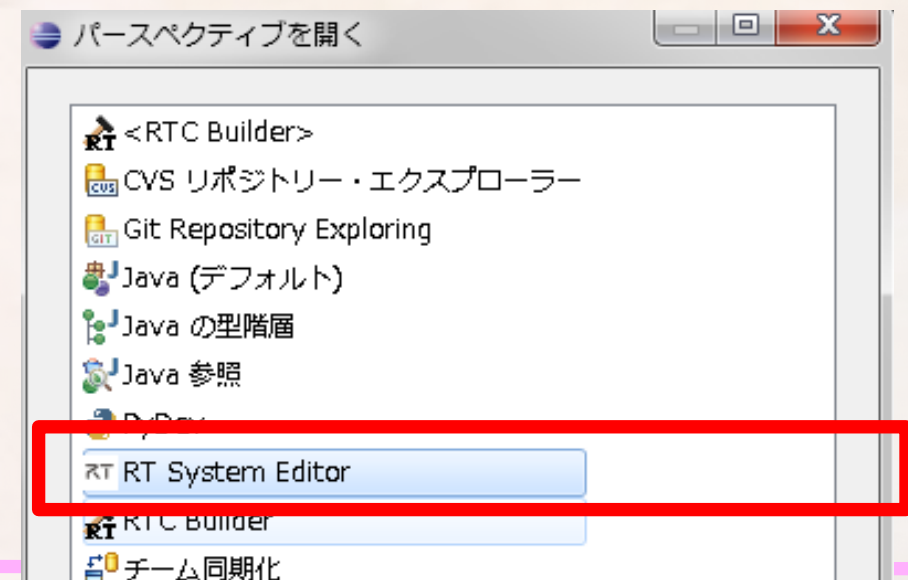
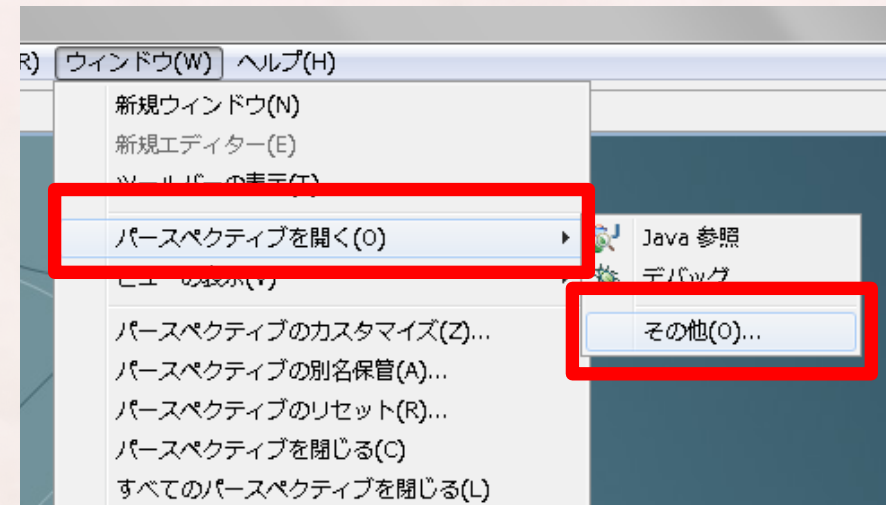
ツールの役割

- ツールの種類
 - RT System Editor
 - Eclipseのプラグイン
 - GUI
 - rtshell
 - コマンドからRTCを制御
 - CUI. スクリプト化が可能=自動化
 - 自作ツール
 - ツール作成自体も容易
- ツールの役割
 - RTC間の接続
 - RTCのコンフィグレーションの変更
 - RTCの状態の変更
 - 実行コンテキストの制御



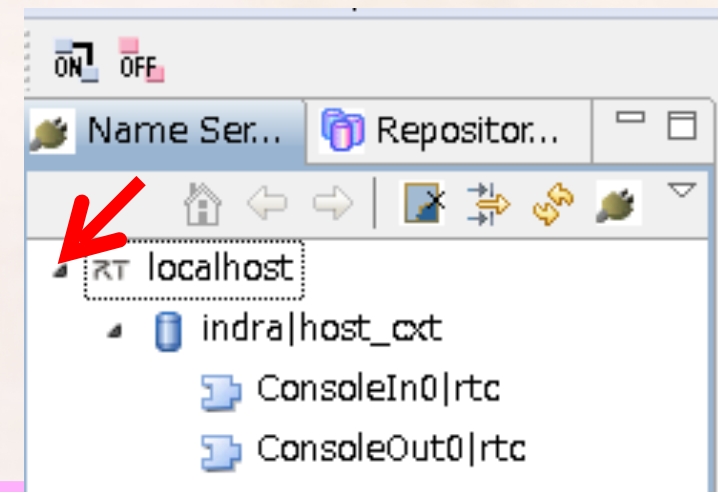
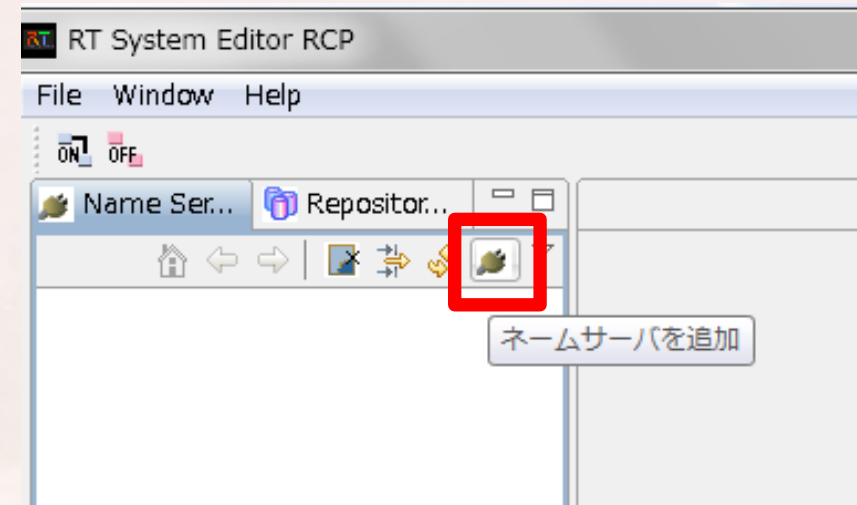
RT System Editorの使い方

- パースペクティブを「RT System Editor」に変更
 - パースペクティブとはEclipseの作業画面のレイアウトタイプ
 - メニュー>「ウィンドウ」>「パースペクティブを開く」>「その他」
 - 「RT System Editor」を選択



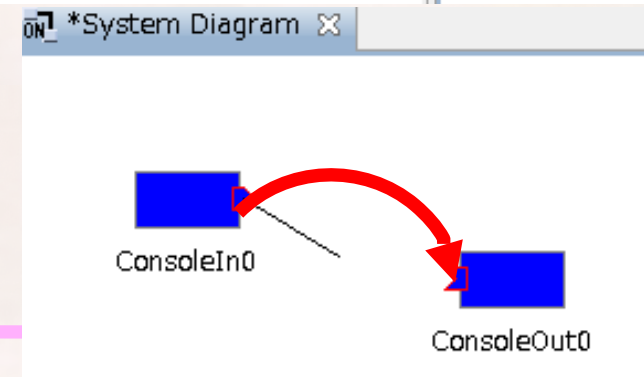
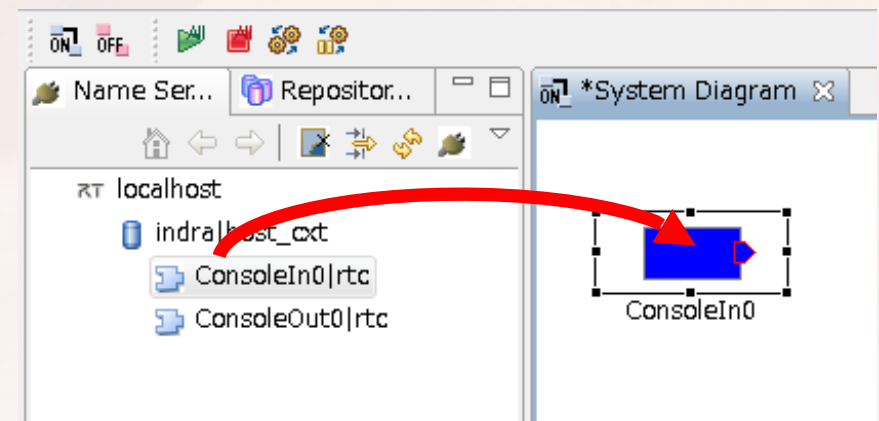
RT System Editorの使い方

- 所望のネームサーバが見つからない場合は追加処理
 - ネームサーバとは実行中のRTCの管理を行うサーバ
- localhost (自分自身)
- デフォルトでlocalhostを検索し、ネームサーバが発見されれば追加される
- Macでは不具合があるので、ネームサーバを起動する前にRTSEを起動した方がいい
- ネームサーバに起動したRTCが登録されていれば成功
 - ConsoleInは
 - 「/localhost/indra.host_cxt/
ConsoleIn0.rtc」という名前でネームサーバに登録された
 - 名前の登録ルールを変更することも簡単
 - rtc.conf



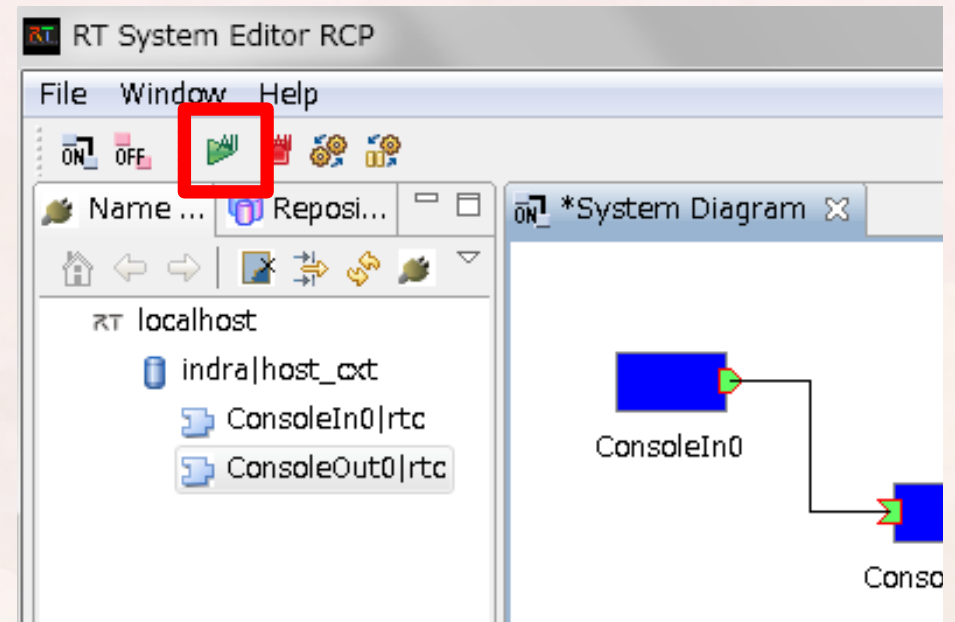
RT System Editorの使い方

- System Editorを開いてRTシステムを編集する
 - メインビューに空の「System Diagram」が表示される
- ネームサービスビューからドラッグ&ドロップ
 - 利用するRTCをすべて表示
- ポートとポートをドラッグ&ドロップで接続
 - 表示されるダイアログはOK



RT System Editorの使い方

- すべてのRTCをACTIVATE
- ConsoleInのウィンドウに「Please Input number」と表示
 - 適当な数字を入れると ConsoleOut側に表示
 - RTC間の通信が行われたことが分かる
- DEACTIVATEする場合は、指令を送ってからConsoleInに数字を送る必要がある
 - (scanf入力待ちになっている)



OpenRTM-aist学習用台車シミュレータ

<http://ysuga.net/?p=133>

- Loader.batを実行

- シミュレータ
- 仮想ジョイスティック
- コントローラ

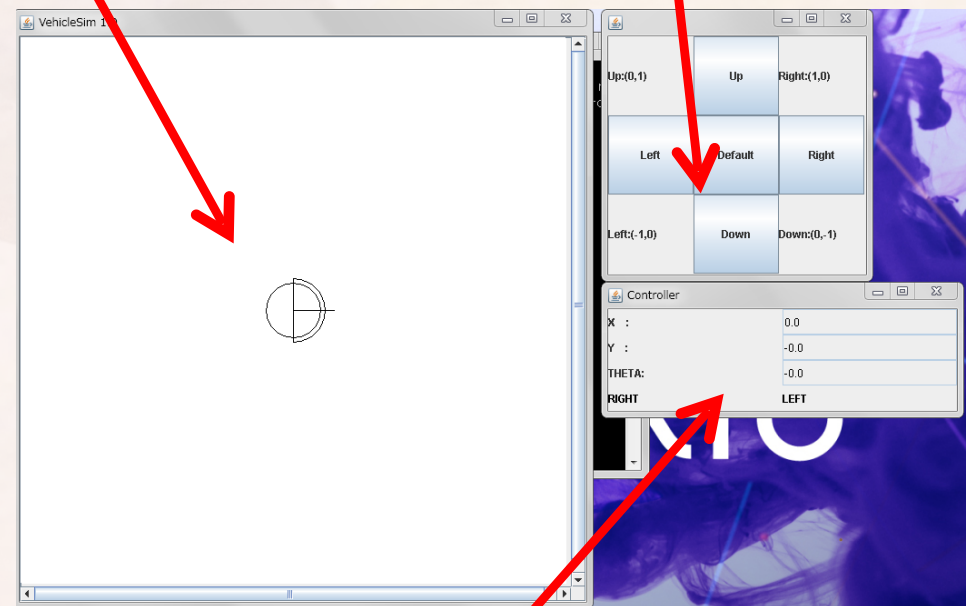
- すべて接続してACTIVATE

- ジョイスティックで操作

- コントローラGUIで位置や接触センサを確認

シミュレータ

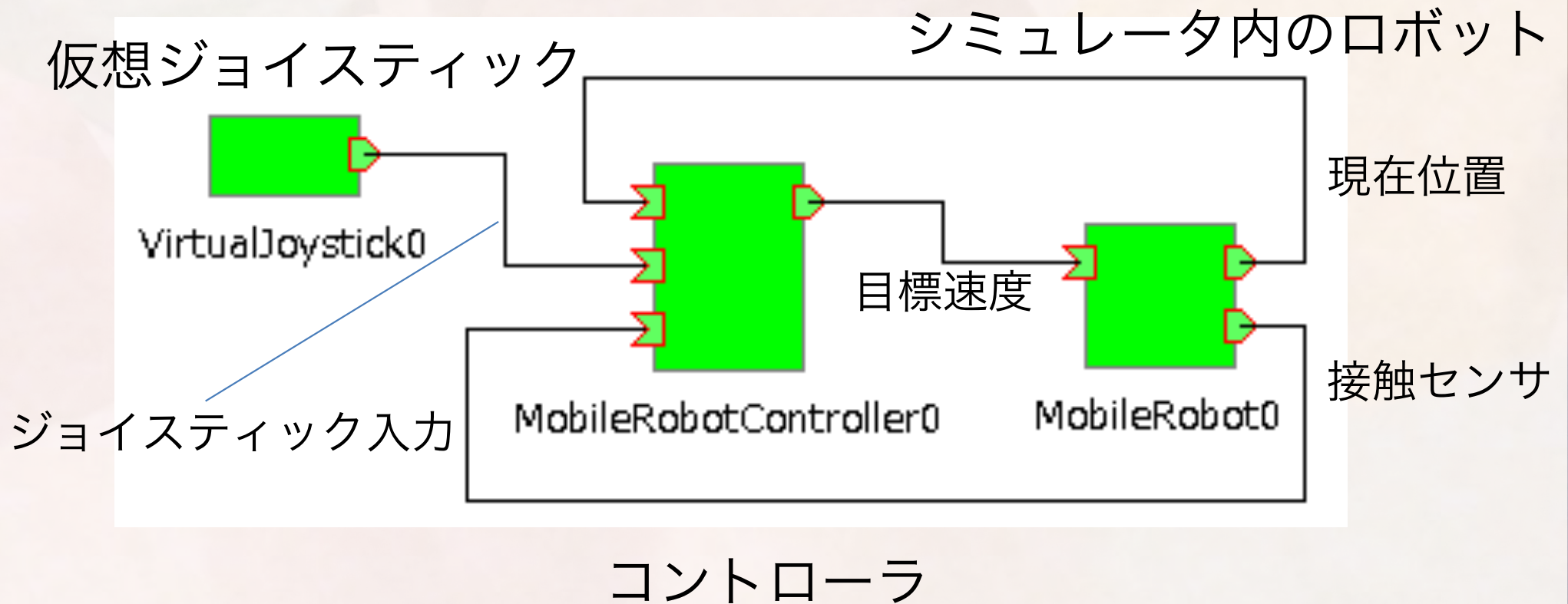
仮想ジョイスティック



コントローラGUI

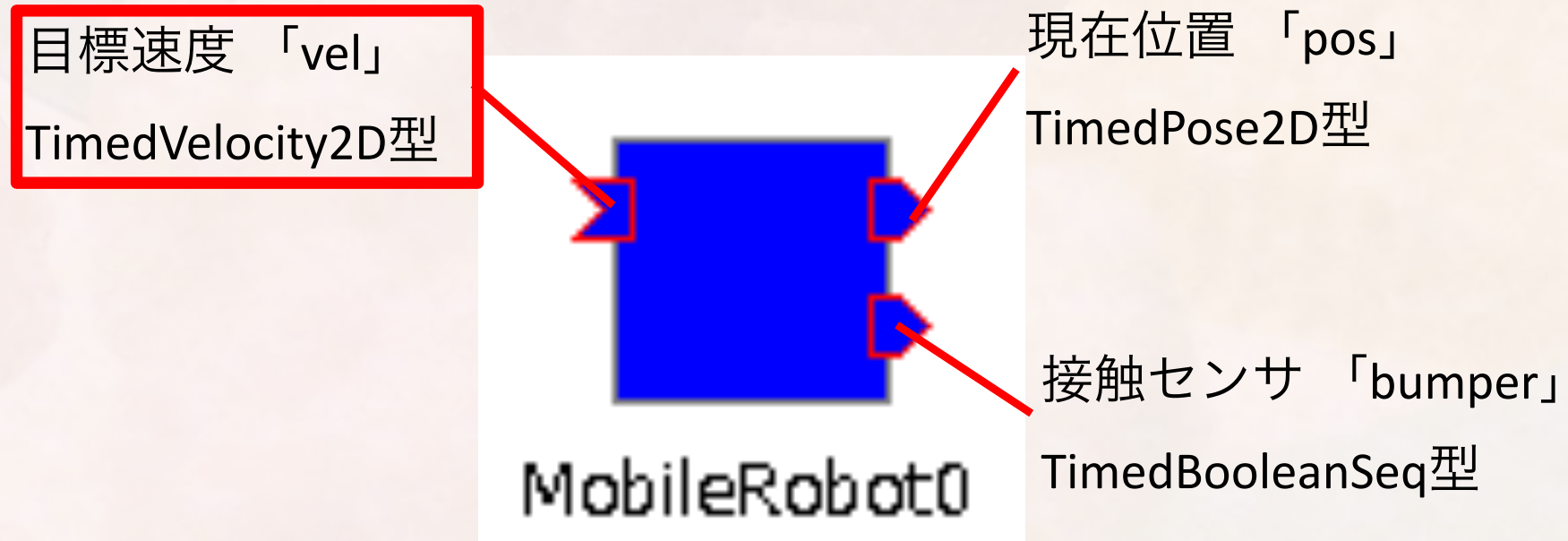
台車シミュレータの中身

- RT System Editorで表示



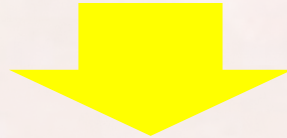
実習1. 台車に指令を送ってみよう

- 台車に直進と回転の指令を同時に送り、その場でぐるぐる回る動作をさせてみよう



RTCプログラミングの流れ

- RTC Builderによるスケルトンコードの生成



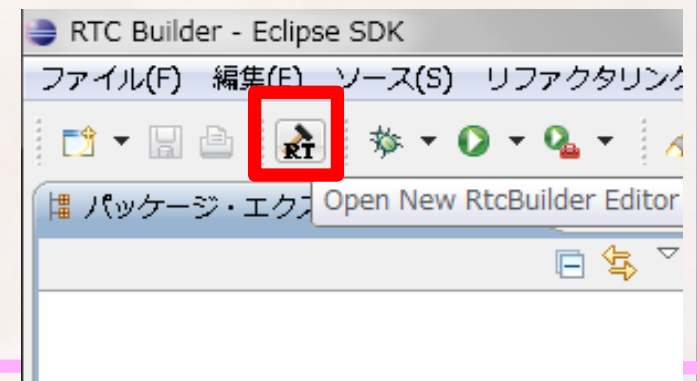
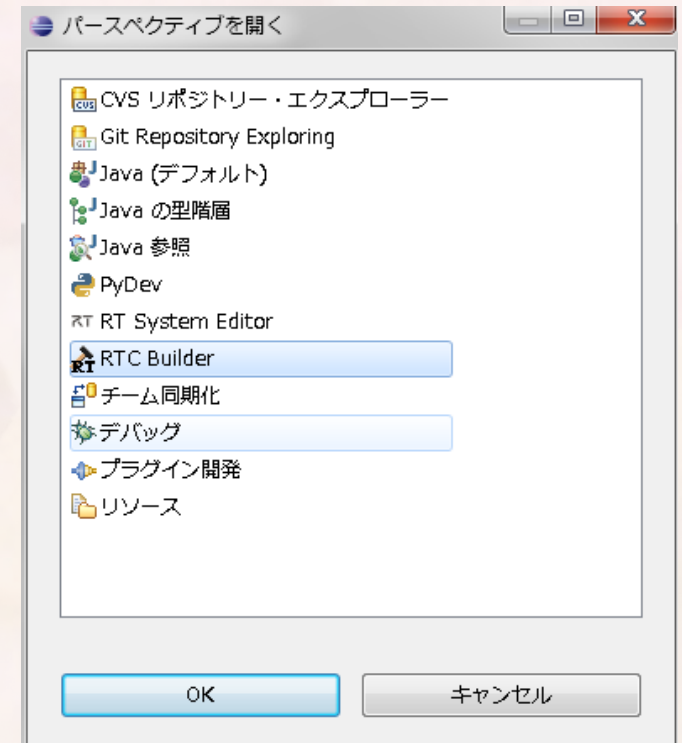
- スケルトンコードの特定のイベントハンドラに独自のコードを追加
 - on_initialized CREATED->INACTIVE
 - on_activated INACTIVE->ACTIVE
 - on_deactivated ACTIVE->INACTIVE
 - on_execute ACTIVE状態で周期的に呼ばれる



- コンパイルし実行

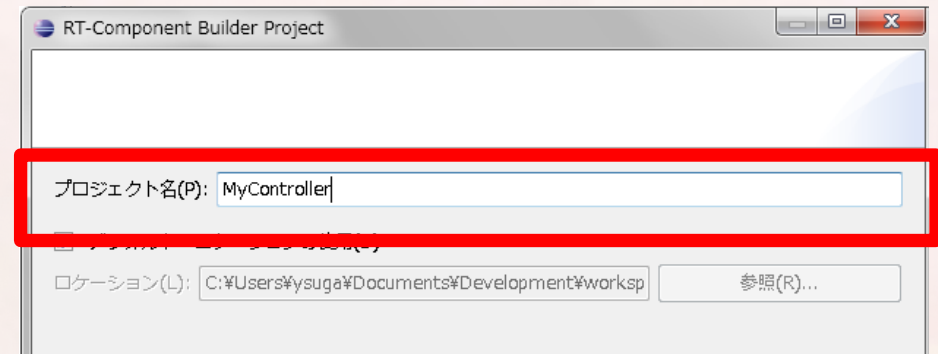
RTC Builder

- パースペクティブをRTC Builderに変更
 - Builder Editorで, RTCの骨格 (スケルトン) コードを作成できる
- Builder Editorを開く
 - RTCが持つべき「イベントハンドラ」が実装されているので, それを編集して自分のコードを差し込む



RTC Builder

- プロジェクト名の入力
– MyController
- 画面下部のタブを切り替えながら必要な情報を入力



RTC Builder

- 基本タブ

基本

RT-Component Basic Profile

このセクションではRTコンポーネントの基本情報を指定します

***モジュール名:** MyController

モジュール概要: ModuleDescription

***バージョン:** 1.0.0

***ベンダ名:** ysuga_net

***モジュールカテゴリ:** Example

コンポーネント型: STATIC

アクティビティ型: PERIODIC

コンポーネント種類: DataFlow FSM MultiMode

最大インスタンス数: 1

実行型: PeriodicExecutionContext

実行周期: 1000.0

ヒント

モジュール名: RTコンポ
この名称
使用でき

モジュール概要: RTコンポ
ASCII文

バージョン: RTコンポ
x.y.z(x,y)

ベンダ名: RTコンポ
ASCII文

モジュールカテゴリ: RTコンポ
選択肢に
使用でき

コンポーネント型: RTコンポ
・STAT
・UNIQ
・COMI

アクティビティ型: RTコンポ
・PERIC
・SPOR
・EVEN

RTC Builder

- アクティビティタブ

アクティビティ

▼ アクティビティ

このセクションでは使用するアクションコールバックを指定します。

コンポーネントの初期化と終了処理に関するアクション

onInitialize onFinalize

実行コンテキストの起動と停止に関するアクション

onStartup onShutdown

alive状態でのコンポーネントアクション

onActivated onDeactivated onAborting

onError onReset

Dataflow型コンポーネントのアクション

onExecute onStateUpdate onRateChanged

FSM型コンポーネントのアクション

onAction

Mode型コンポーネントのアクション

onModeChanged

▼ Documentation

このセクションでは各アクションの概要を説明するドキュメントを記述します。
上段のアクションを選択すると、それぞれのドキュメントを記述できます。

▼ ヒント

onInitialize	初期化処
onFinalize	終了処理
onStartup	Executi
onShutdown	Executi
onActivated	非アクテ
onDeactivated	アクティ
onAborting	ERROR状
onError	ERROR状
onReset	ERROR状
onExecute	アクティ
onStateUpdate	onExecu
onRateChanged	Executi
onAction	対応する
onModeChanged	モードが

動作概要: アクティ
事前条件: アクティ
事後条件: アクティ

アクティビティ名: onExecute ON OFF

1. onExecuteを選択

2. ONにする

RTC Builder

- データポートタブ

データポート

▼ DataPortプロファイル

このセクションではRTコンポーネントのDataPort(データポート)の情報を設定します。

*ポート名 (InPort)	

Add Delete

*ポート名 (OutPort)	
velocity	

Add Delete

▼ Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。
上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: velocity (OutPort)

*データ型: RTC::TimedVelocity2D

変数名: velocity

表示位置: RIGHT

ポート名を編集

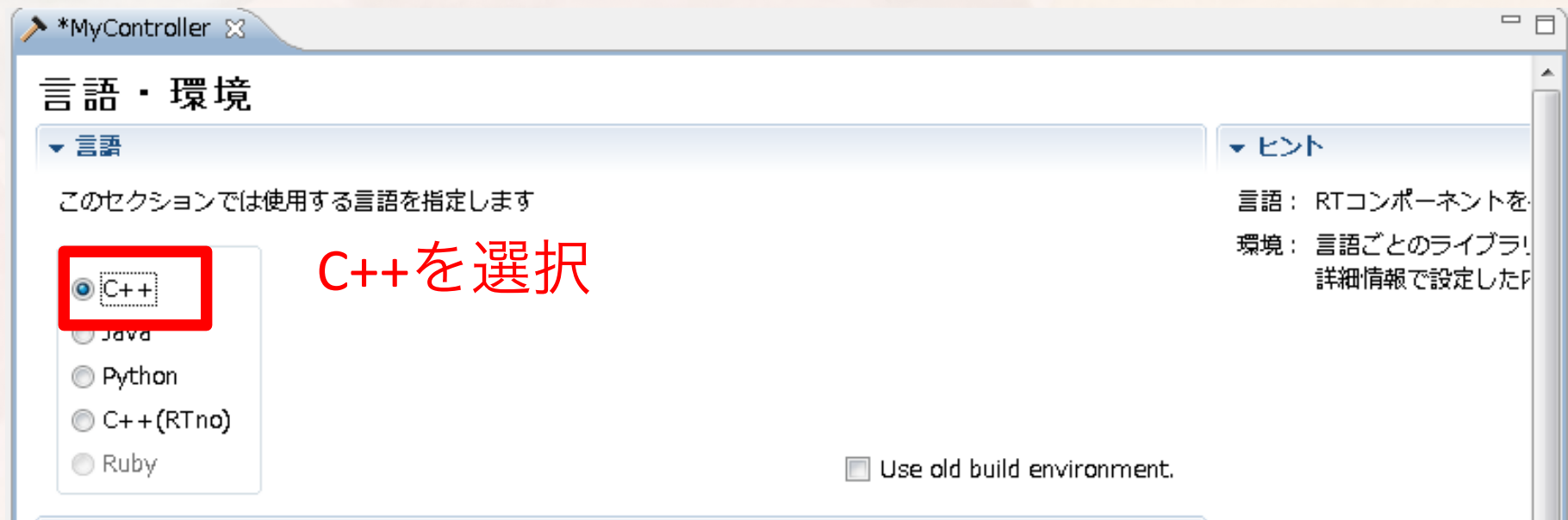
出力ポートを追加

ポートの型を変更

ポートのソースコード上の変数名を入力

RTC Builder

- 言語タブ



RTC Builder

- 基本タブに戻る

実行周期 : 1000.0

概要 :

RTC Type :

▼ コード生成とパッケージ化

コードの生成およびパッケージ化を行います。

コード生成 | パッケージ化 **コード生成**

▼ プロファイル情報のインポート・エクスポート

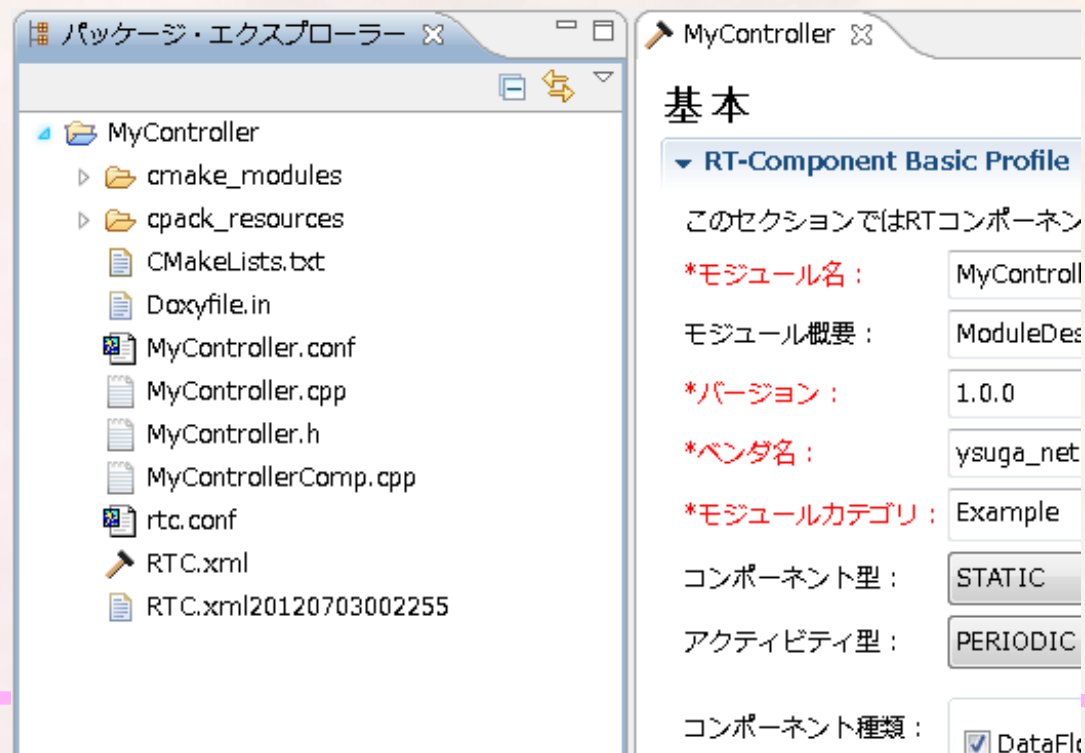
プロファイル情報のインポートおよびエクスポートを行います。

インポート | エクスポート

基本 | アクティビティ | データポート | サービスポート | コンフィギュレーション | ドキュメント生成 | 言語・環境

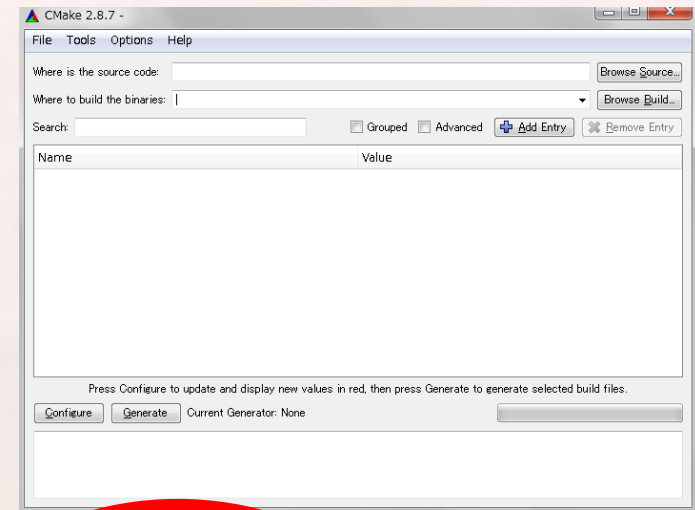
RTC Builder

- MyControllerプロジェクトにソースコードが生成される
 - Builder Editor（真ん中のウィンドウ）は閉じてよい

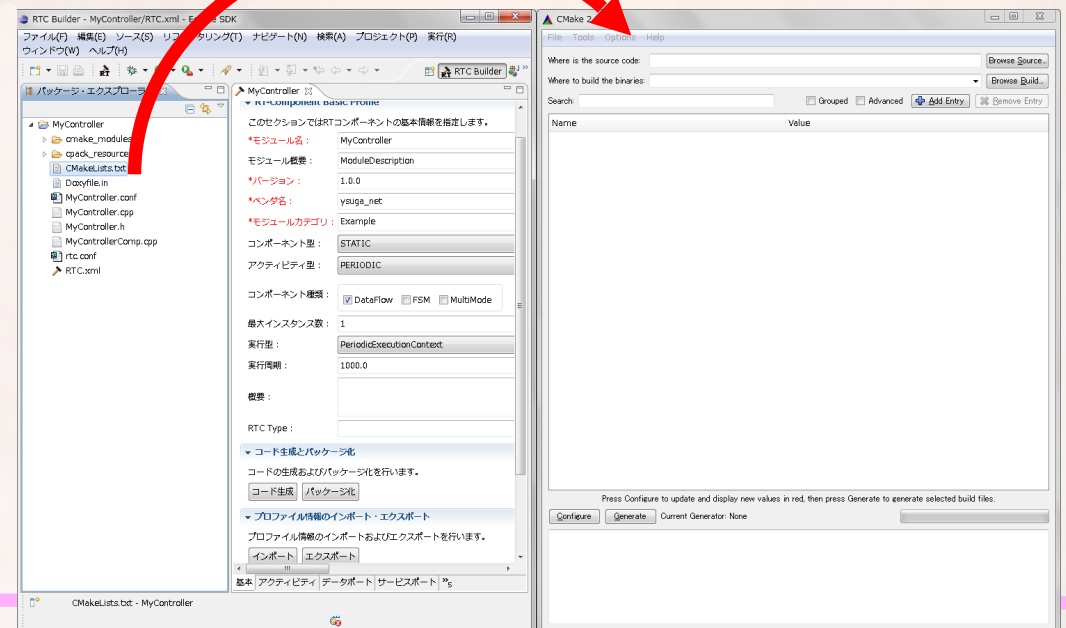


CMakeによるプロジェクト生成

- CMake2.8を起動

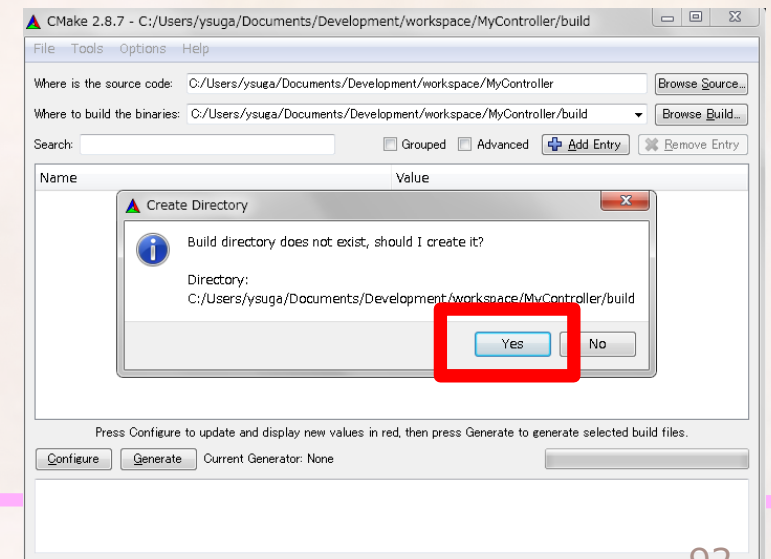
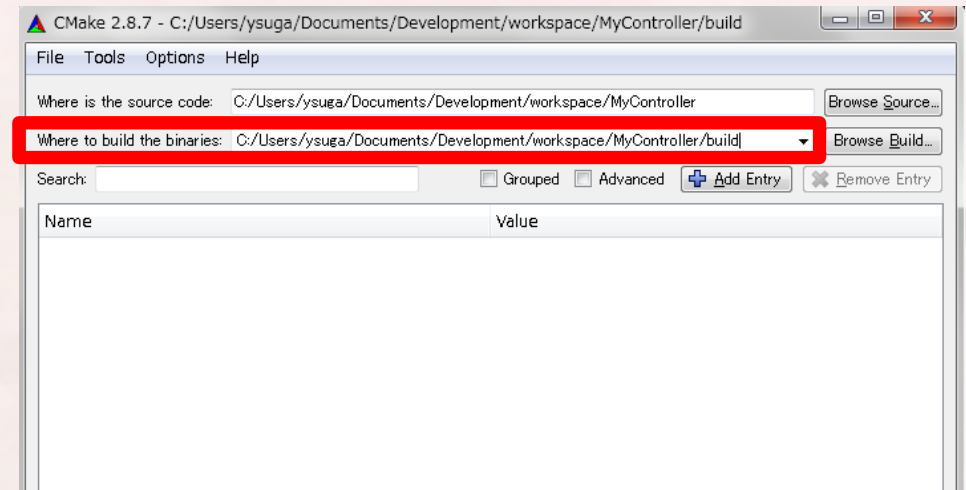


- CMakeLists.txtを
CMakeにドラッグ
& ドロップ



CMake2.8

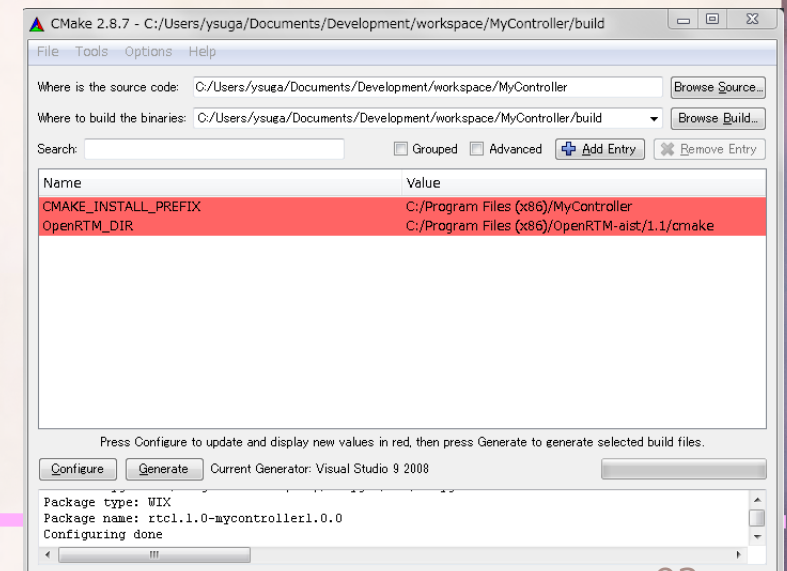
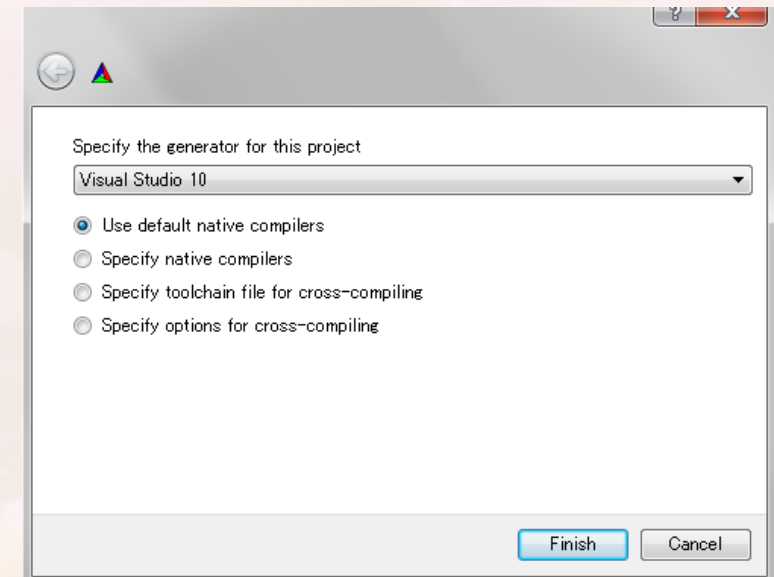
- 出力フォルダパスにbuildを追加
 - RTCのフォルダが”.../workspace/MyController”の場合, ”.../workspace/MyController/build”というフォルダを作る
- Configureを押すとフォルダ作成確認のダイアログが出るのでOK



CMake2.8

- ビルド環境を選択
 - Visual C++ 2010 . . . Visual Studio 10
 - Use default native compilersを選択

- 赤い表示が出るが恐れず「Generate」
 - Visual Studio用プロジェクトファイルが生成される



MacやLinuxなら

- 以下のコマンドを打つ
 - `cd $HOME/workspace/MyController`
 - `mkdir build`
 - `cd build`
 - `cmake ../`
 - `make` // ここでコンパイル
- ただし, Macの場合は, RTCのヘッダーファイルを書き換える必要がある
 - RTCの名前を`$ModuleName`とすると・・・
 - RTCのディレクトリ`/include/$ModuleName/$ModuleName.h`を開く

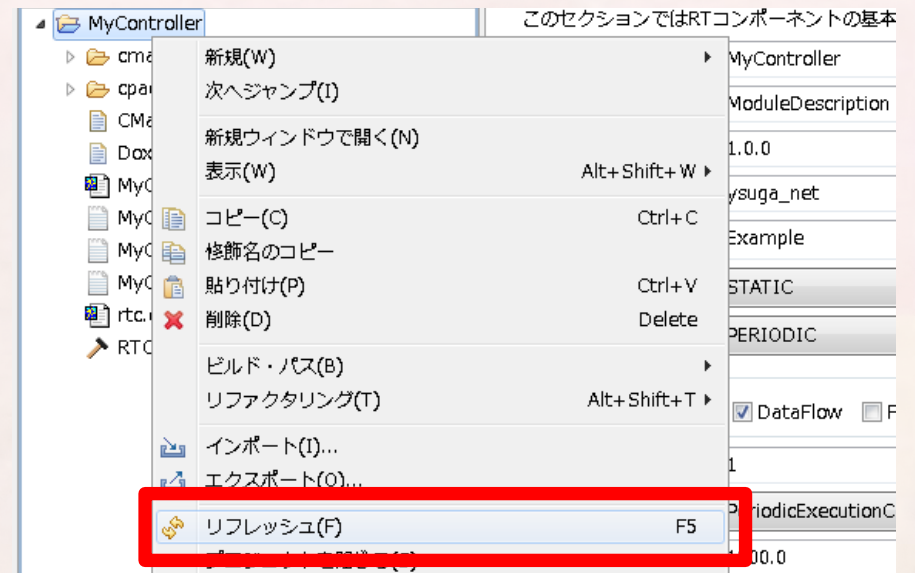
```
#include <rtm/Manager.h>
#include <rtm/DataFlowComponentBase.h>
#include <rtm/CorbaPort.h>
#include <rtm/DataInPort.h>
#include <rtm/DataOutPort.h>
#include <rtm/idl/BasicDataTypeSkel.h>
#include <rtm/idl/ExtendedDataTypesSkel.h>
#include <rtm/idl/InterfaceDataTypesSkel.h>
```



```
#include <rtm/idl/BasicDataTypeSkel.h>
#include <rtm/idl/ExtendedDataTypesSkel.h>
#include <rtm/idl/InterfaceDataTypesSkel.h>
#include <rtm/Manager.h>
#include <rtm/DataFlowComponentBase.h>
#include <rtm/CorbaPort.h>
#include <rtm/DataInPort.h>
#include <rtm/DataOutPort.h>
```

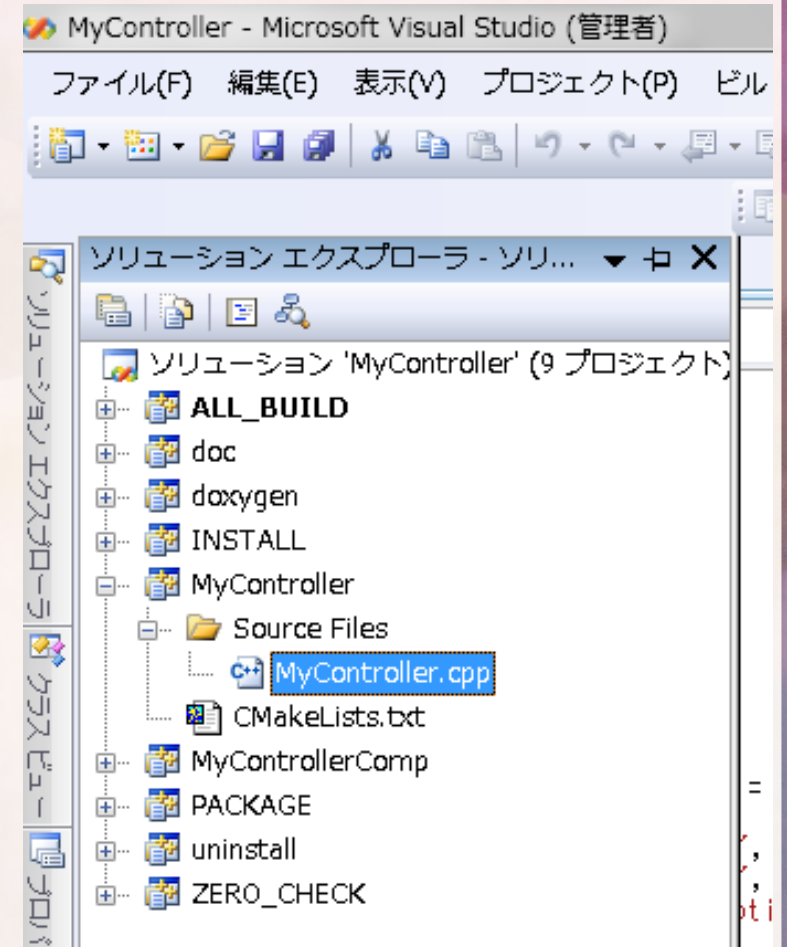

RTCBuilder

- MyControllerプロジェクトを右クリックして「リフレッシュ」
 - ファイルが加わっているのがわかる
- build/src/MyController.slnファイルをダブルクリック
 - Visual Studioが起動する



Visual Studio

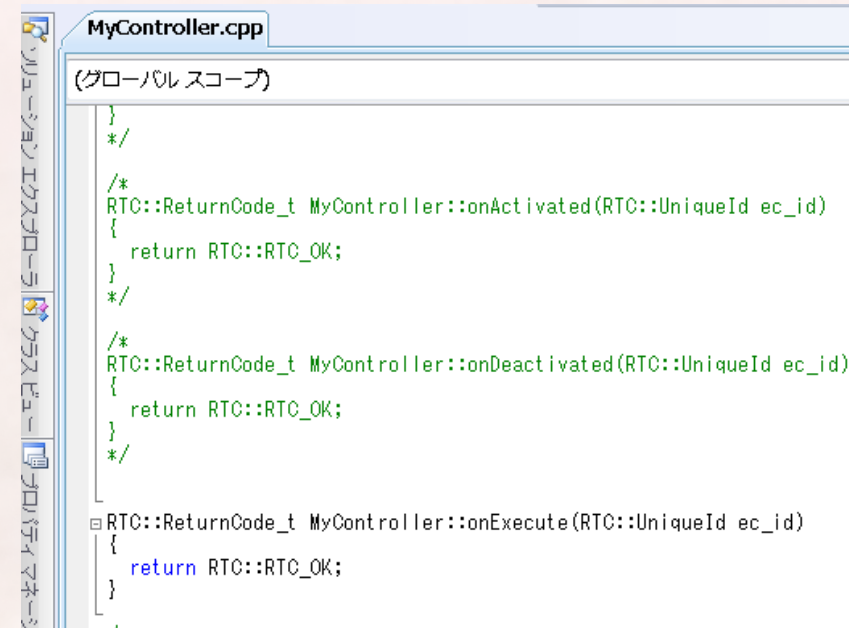
- 複数のプロジェクトが存在
 - MyController・・・RTC本体
 - MyControllerComp・・・RTCを単体のアプリケーションとして実行するためのプロジェクト
- MyControllerプロジェクト
 - MyController.cppを開く



Visual Studio

- MyController::onExecute関数

- RTCがACTIVE状態の場合に周期的に呼ばれる
- 実行周期は後で設定
- ここに周期的に呼ばれるべき制御アルゴリズム等を記述

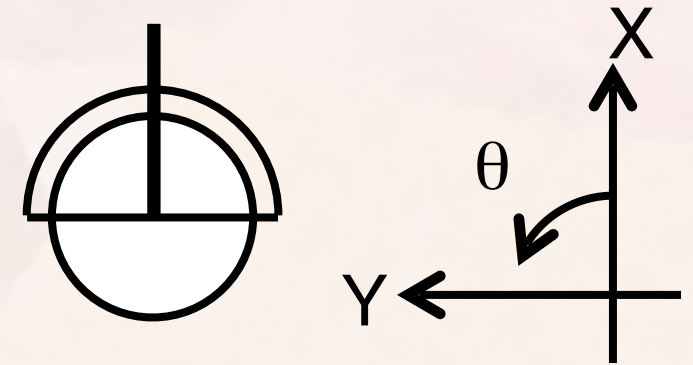


```
MyController.cpp
(グローバルスコープ)
}
*/
/*
RTC::ReturnCode_t MyController::onActivated(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}
*/
/*
RTC::ReturnCode_t MyController::onDeactivated(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}
*/
RTC::ReturnCode_t MyController::onExecute(RTC::UniqueId ec_id)
{
    return RTC::RTC_OK;
}
*/
```

```
RTC::ReturnCode_t MyController::onExecute(RTC::UniqueId ec_id)
{
    m_velocity.data.vx = 0.05; // バッファに書込む
    m_velocity.data.vy = 0;
    m_velocity.data.va = 1.0;
    m_velocityOut.write(); // データを送信
    return RTC::RTC_OK;
}
```

移動ロボットに指令を送る

- TimedVelocity2D . . . 2次元平面での速度
 - data.vx . . . X軸方向速度 [m/s]
 - data.vy . . . Y軸方向速度 [m/s]
 - data.va . . . Z軸方向速度 [rad/s]
- データ型がどんな構造体なのか知るには, IDLファイルを読む
 - 通常は, C:\Program Files (x86)\OpenRTM-aist\1.1\rtm\idlにある
 - TimedVelocity2Dは, ExtendedDatatype.idlで定義されている.



LinuxやMacなら,

`/usr/include/openrtm-1.1/rtm/idl`
もしくは,
`/usr/local/include/openrtm-1.1/rtm/idl`

IDLファイル

- オブジェクト指向言語に変換可能なインターフェース定義言語
- C++, Java, Pythonに変換
- 独自のデータ型を定義するにはIDLを書いてRTC Builderに読み込ませる

```
/*!  
 * @struct TimedVelocity2D  
 * @brief Time-stamped version of Velocity2D.  
 */  
struct TimedVelocity2D  
{  
    Time tm;  
    Velocity2D data;  
};
```

```
/*!  
 * @struct Velocity2D  
 * @brief Velocities in 2D cartesian space.  
 */  
struct Velocity2D  
{  
    /// Velocity along the x axis in metres per second.  
    double vx;  
    /// Velocity along the y axis in metres per second.  
    double vy;  
    /// Yaw velocity in radians per second.  
    double va;  
};
```

データポート関連変数の命名法則

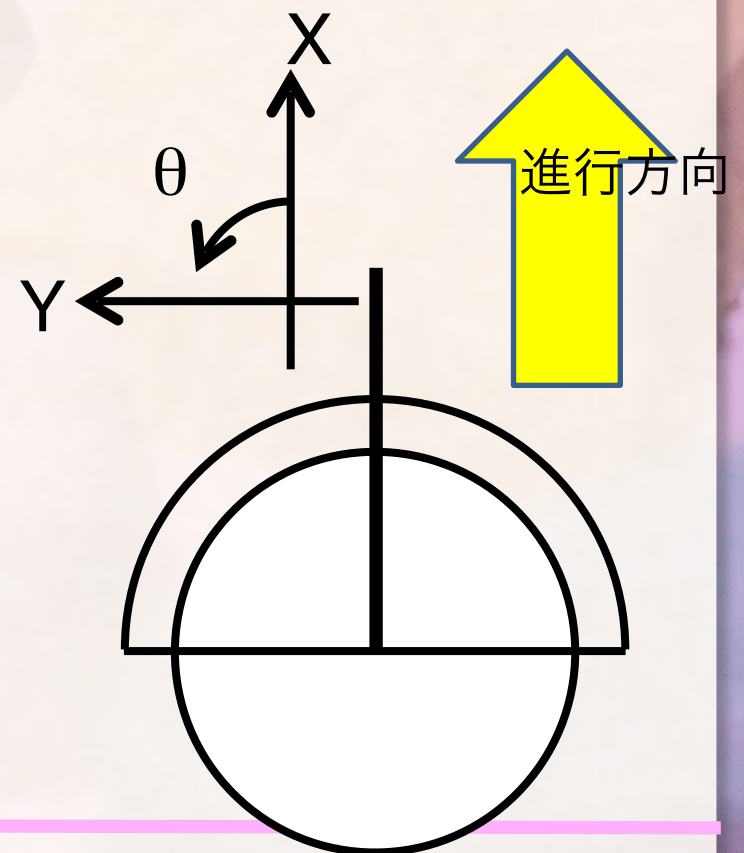
- 出力ポートの場合
 - 「変数名」 = 「example」
 - m_example
 - データポートのデータを入れるバッファ
 - m_exampleOut
 - データポート本体
- 入力ポートの場合
 - 「変数名」 = 「example」
 - m_example
 - データポートのデータを入れるバッファ
 - m_exampleIn
 - データポート本体
- 利用方法
 - 1. m_example にデータを入力
 - 2. m_exampleOut.write()
- 利用方法
 - 1. m_exampleIn.isNew()で受信確認
 - 2. m_exampleIn.read()でデータ取得
 - 2. m_exampleのデータを読み取る

実行

- ネームサービスの起動確認
- RTCの実行
 - MyControllerCompを右クリック
 - 「デバッグ→インスタンス作成」で実行
- MobileRobotSim.batを実行するとシミュレータのみ起動
- RT System Editorでの接続, ACTIVE化

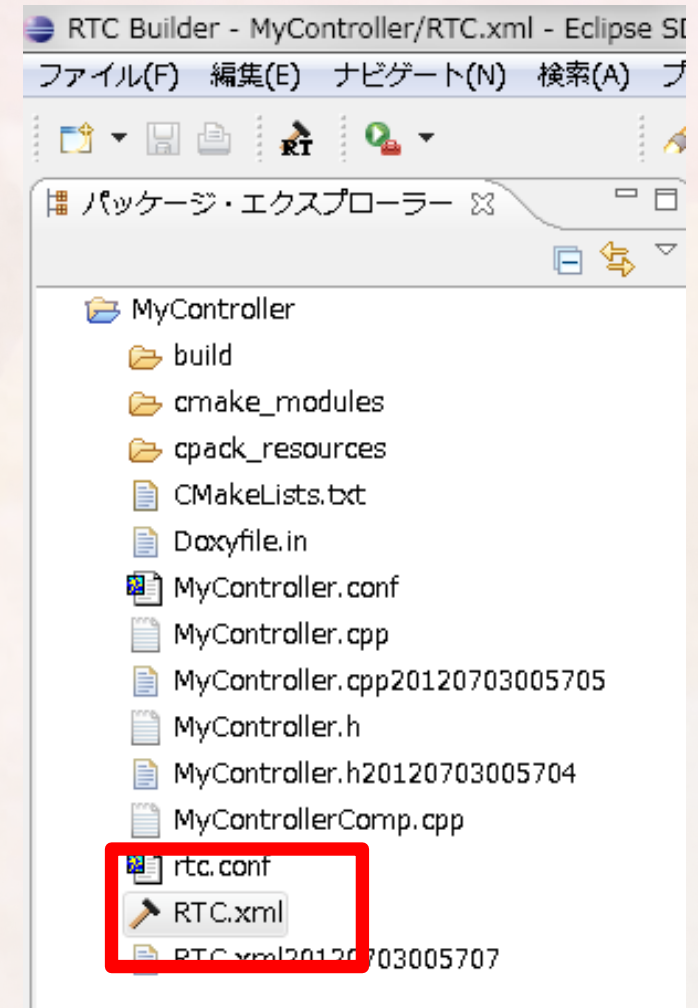
実習2. 台車の動きを調整

- コンフィグレーション機能を試す
 - コンフィグレーションとは、実行中のRTCを調整するための機能
 - 例：制御ゲインの調整
- コンフィグレーション機能を加えて実行中にMyControllerの機能を調整する



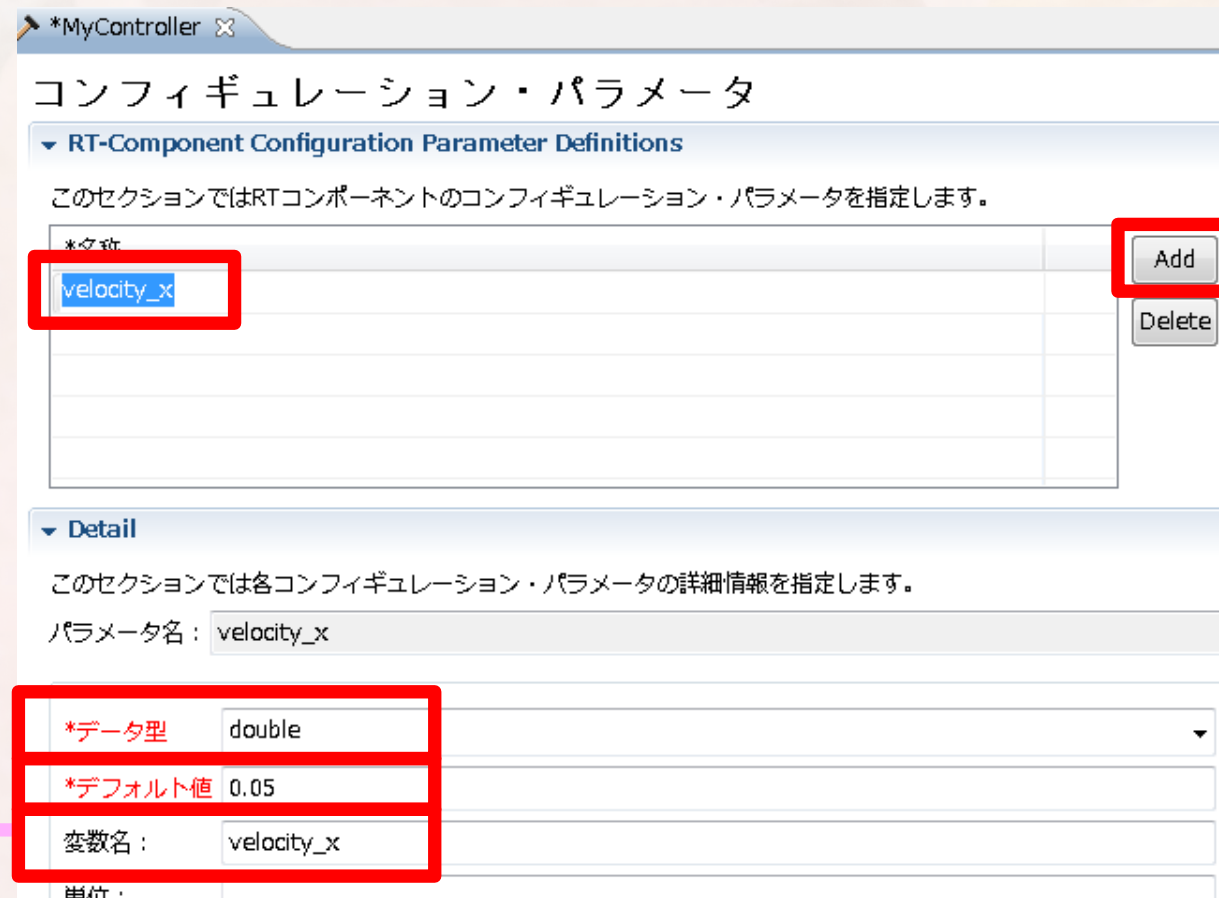
RTC Builder

- EclipseのパーспекティブをRTC Builderに再変更
- パッケージエクスプローラのRTC.xmlファイルをダブルクリックする
 - 先ほど作成した情報が記録されている



RTC Builder

- コンフィグレーションタブ
 - velocity_x というコンフィグレーションを追加



追加したコンフィグレーション

- `velocity_x` : `double`型
 - 変数名: `velocity_x`
 - デフォルト値: `0.05`
- `velocity_theta` : `double`型
 - 変数名: `velocity_theta`
 - デフォルト値: `1.0`

RTC Builder

- 比較ダイアログ

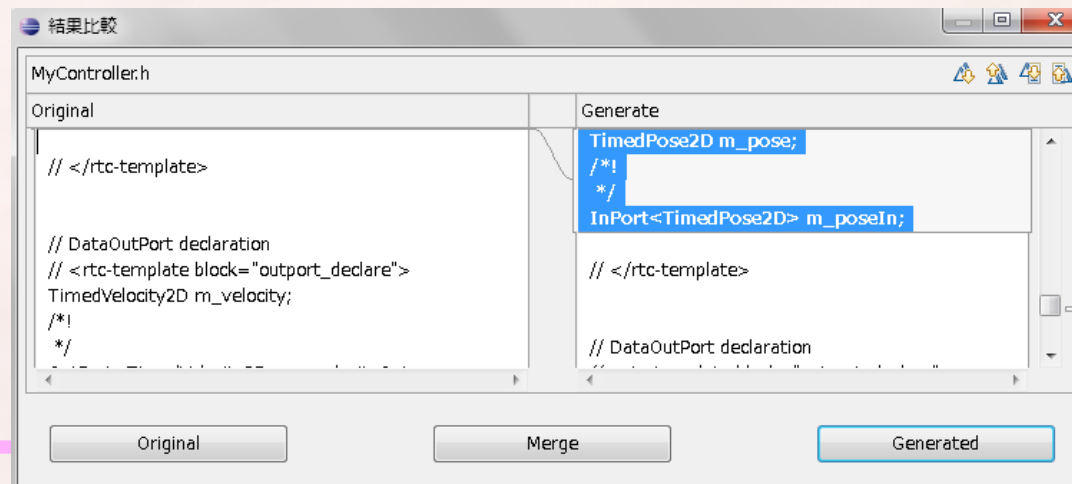
- ここでは「Generated」を選択

- 基本的に「Merge」を選択

- 新しいコードの変更点のみ反映

- Generatedは新しいコード側で上書きされるので、自分の記入したコードが消える

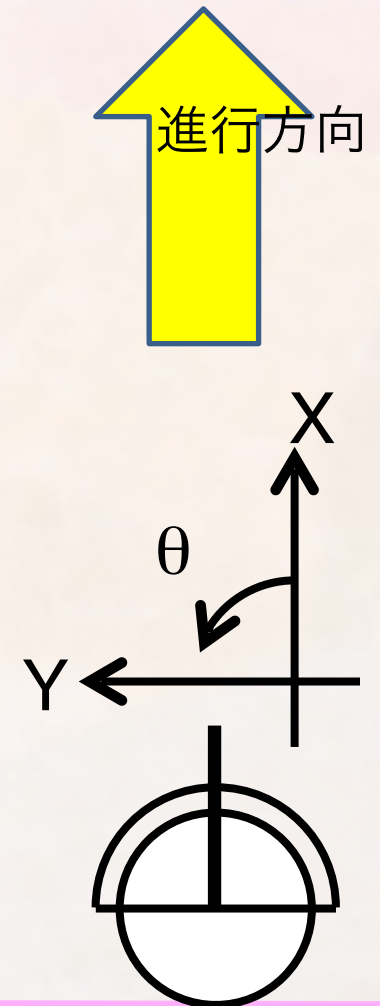
- バックアップファイルがあるので心配ありません



実習2. 台車の動きを調整

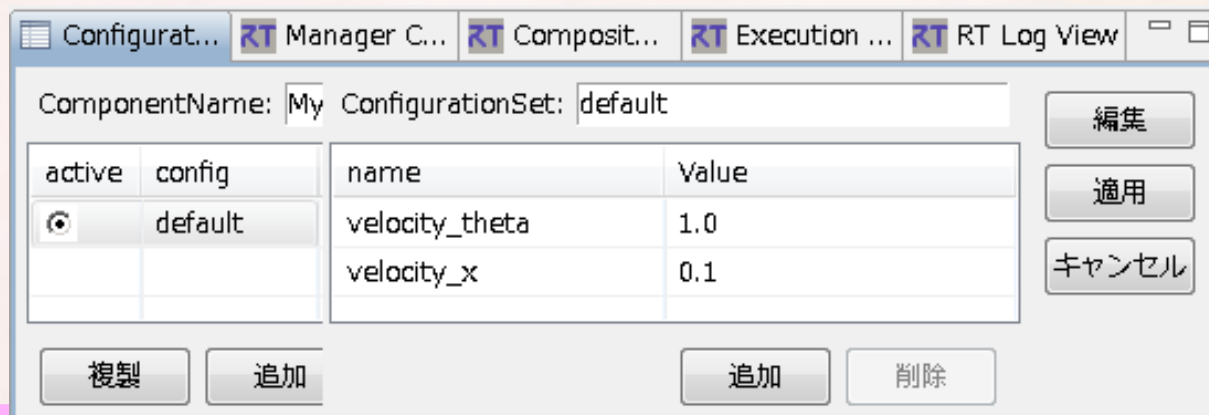
- コンフィグレーション機能を試す
 - コンフィグレーション機能を加えて実行中にMyControllerの機能を調整する

```
RTC::ReturnCode_t MyController::onExecute(RTC::Uniqueld ec_id)
{
    std::cout << "Vx=" << m_velocity_x << std::endl;
    std::cout << "Vtheta=" << m_velocity_theta << std::endl;
    m_velocity.data.vx = m_velocity_x;
    m_velocity.data.vy = 0;
    m_velocity.data.va = m_velocity_theta;
    m_velocityOut.write();
    return RTC::RTC_OK;
}
```



実行

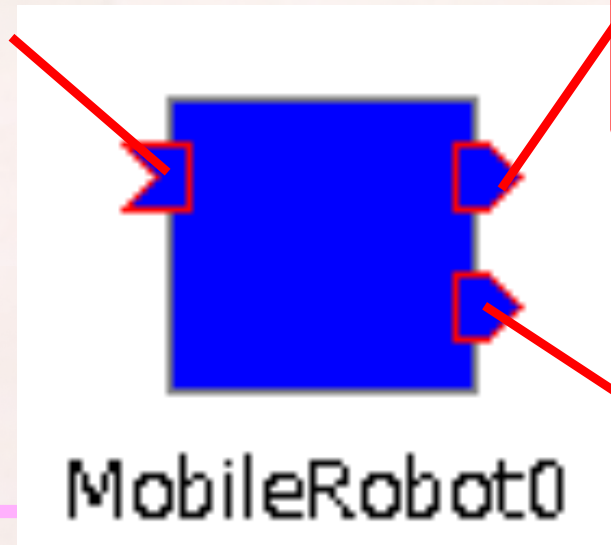
- ネームサービスの起動確認
- RTCの実行
 - MobileRobotSim.batを実行するとシミュレータのみ起動
- RT System Editorでの接続, ACTIVE化
- RT System Editor下部にConfiguration Viewに選択中のRTCのコンフィグが表示されるので変更して「適用」を選択する
(適用しないと反映されない)



実習3. 位置の取得

- 現在位置をデータポートから受け取ってコンソールに表示する

目標速度 「vel」
TimedVelocity2D型



現在位置 「pos」
TimedPose2D型

接触センサ 「bumper」
TimedBooleanSeq型

RTC Builder

- データポートタブ
 - TimedPose2D型の入力ポート「pose」を追加

*MyController

データポート

▼ DataPortプロファイル

このセクションではRTコンポーネントのDataPort(データポート)の情報を設定します。

*ポート名 (InPort)		*ポート名 (OutPort)	
pose	Add Delete	velocity	Add Delete

▼ Detail

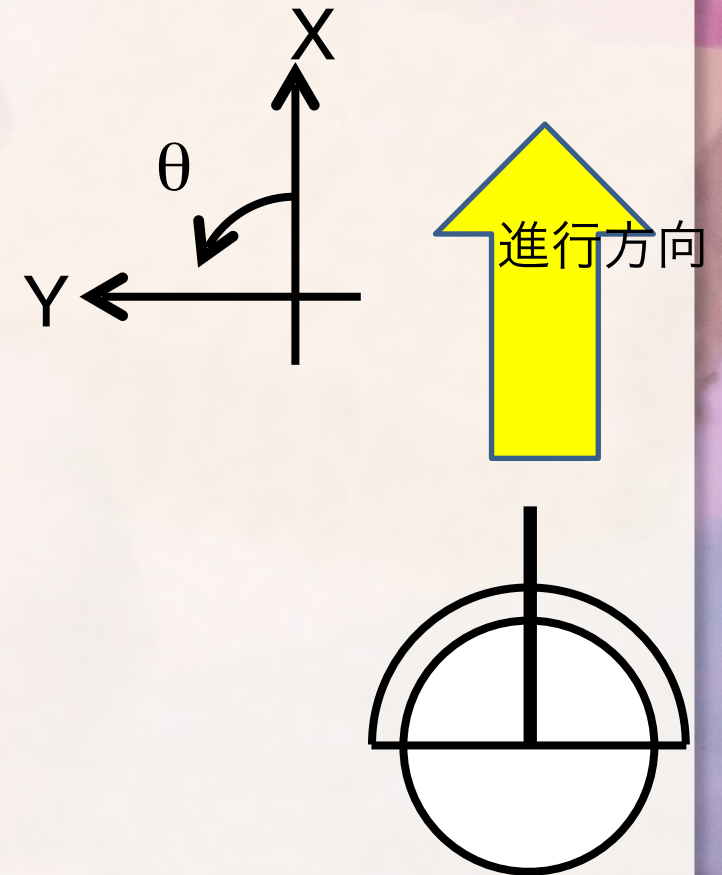
このセクションではデータポート毎の概要を説明するドキュメントを記述します。
上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名: pose (InPort)

*データ型	RTC::TimedPose2D
変数名	pose
表示位置	LEFT

2次元位置のデータ型

- TimedPose2D . . . 2次元平面での位置および姿勢
 - data.position.x . . . X軸方向変位
 - data.position.y . . . Y軸方向変位
 - data.heading . . . Z軸方向回転



Visual Studio

- 再度, onExecute関数を編集

- 入力ポートはデータが来ているか確認する処理が入る

```
RTC::ReturnCode_t MyController::onExecute(RTC::Uniqued ec_id)
{
    m_velocity.data.vx = m_velocity_x;
    m_velocity.data.vy = 0;
    m_velocity.data.va = m_velocity_theta;
    m_velocityOut.write();

    if(m_poseIn.isNew()) { // 入力ポートに入力があるか確認
        m_poseIn.read(); // 入力があるならば読み込む
        std::cout << "X = " << m_pose.data.position.x << std::endl;
        std::cout << "Y = " << m_pose.data.position.y << std::endl;
        std::cout << "Z = " << m_pose.data.heading << std::endl;
    }
    return RTC::RTC_OK;
}
```


実行

- ネームサービスの起動確認
- RTCの実行
 - MobileRobotSim.batを実行するとシミュレータのみ起動
- RT System Editorでの接続, ACTIVE化
- 実行時の周期が速すぎる？

rtc.conf

- RTCの実行時の設定ファイル

- ネームサーバのIPアドレス, ポート番号
- 実行周期, 実行コンテキストの種類
- RTCの名前付け規則
- ログの有無, ログレベル
- etc...

- rtc.confで実行周期を変更 (単位Hz)

- 右クリック→アプリケーション→テキストエディタ

```
exec_cxt.periodic.rate: 1.0
```

- rtc.confを実行ファイルと同じ場所に置いて, 実行ファイルを実行

- Visual C++でデバッグする場合は, プロジェクトファイルと同じディレクトリに置く (デフォルトでそこがカレントディレクトリになる)

実習4. 台車の接触スイッチ

- TimedBooleanSeq 真偽型 (True/False)

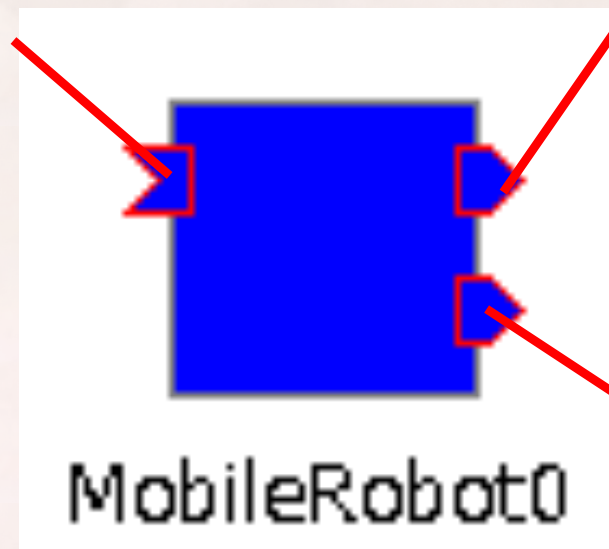
の配列

目標速度 「vel」

TimedVelocity2D型

現在位置 「pos」

TimedPose2D型



接触センサ 「bumper」

TimedBooleanSeq型

配列の0番目要素 = 右側スイッチ

配列の1番目要素 = 左側スイッチ

RTC Builderによるポートの追加

- InPort : TimedBooleanSeq型
 - ポート名: bumper
 - 変数名 : bumper

The screenshot shows the RTC Builder interface for configuring a DataPort. The window title is 'MyController' and the file is 'rtc.conf20120704012444'. The main section is 'データポート' (DataPort) with a sub-section 'DataPortプロフィール' (DataPort Profile). Below this, there is a table for adding and deleting ports. The 'InPort' table has 'pose' and 'bumper' listed. The 'OutPort' table has 'velocity' listed. Below the table, there is a 'Detail' section for the selected 'bumper (InPort)'. The data type is set to 'RTC::TimedBooleanSeq', the variable name is 'bumper', and the display position is 'LEFT'.

*ポート名 (InPort)	Add	*ポート名 (OutPort)	Add
pose		velocity	
bumper	Delete		Delete

Detail

このセクションではデータポート毎の概要を説明するドキュメントを記述します。
上のデータポートを選択すると、それぞれのドキュメントが記述できます。

ポート名 : bumper (InPort)

*データ型: RTC::TimedBooleanSeq

変数名: bumper

表示位置: LEFT

Visual Studio

- ****Seq型はdataメンバを配列のように使える**

```
RTC::ReturnCode_t MyController::onExecute(RTC::Uniqued ec_id)
{
    . . . 省略 . . .

    if(m_bumperIn.isNew()) {
        m_bumperIn.read();
        if(m_bumper.data[0] == true) {
            std::cout << "Right Bumper Hit!!" << std::endl;
        }else if(m_bumper.data[1] == true) {
            std::cout << "Left Bumper Hit!!" << std::endl;
        }
    }
    return RTC::RTC_OK;
}
```

まとめ

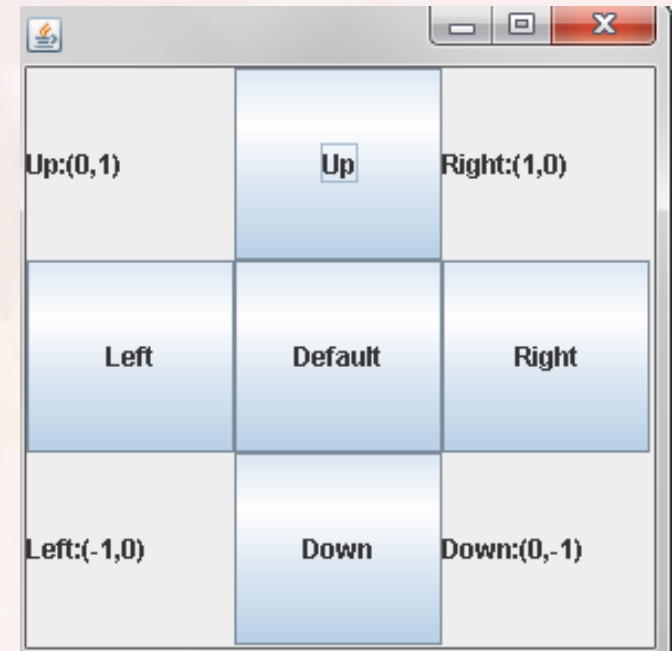
- ツールの使い方
 - RT System Editor (RTCの接続, Activate/Deactivate)
 - RTC Builder (RTCのスケルトンコード生成)
 - CMake (Visual C++用プロジェクト生成)
- コーディング方法
 - データポート入出力
 - TimedVelocity2D, TimedPose2D, TimedBooleanSeq
 - コンフィグレーション
 - rtc.confの設定

みなさんへの課題

- 接触スイッチの反応でロボットの動作を変える
 - 左右どちらかに旋回する？
- 位置の値を使って図形を描く
 - 正方形, 星型などなど
- 仮想ジョイスティックを使った操作



スティック入力「out」
TimedDoubleSeq型



各ボタンの出力値はGUIに記載されている

例： Up(0,1) → 0番目要素が0, 1番目要素が1

- お使いのロボット関連製品をRTC化する
 - onExecute以外の使えるイベントハンドラ例(RTC Builderで使えるように設定)
 - onActivated・・・Activate時に一回呼ばれる（初期化用）
 - onDeactivated・・・Deactivate時もしくはエラー状態遷移時に呼ばれる（終了処理）

ご清聴ありがとうございました

株式会社SUGAR SWEET ROBOTICS

菅 佑樹

@ysuga (RTMのこととかも呟きます)

ysuga@sugarsweetrobotics.com

<http://sugarsweetrobotics.com>

<http://ysuga.net>