2020年9月23日(水)
SICE2020 RTミドルウェア講習会

第１部：OpenRTM-aistおよび RTコンポーネントプログラミングの概要

国立研究開発法人産業技術総合研究所
インダストリアルCPS研究センター
ソフトウェアプラットフォーム研究チーム長
安藤　慶昭

はじめに

- RTミドルウエアの概要
  - 基本概念
- OpenRTM-aist-1.2の新機能と開発ロードマップ
- ROSとの比較、動向
- プラットフォームロボットプロジェクト
- RTMコミュニティー活動
- まとめ

RTミドルウェアとは？

RTとは?

- RT = Robot Technology　　cf. IT
  - ≠Real-time
  - 単体のロボットだけでなく、さまざまなロボット技術に基づく機能要素をも含む (センサ、アクチュエータ, 制御スキーム、アルゴリズム、etc….)

産総研版RTミドルウエア

OpenRTM-aist

- RT-Middleware （RTM)
  - RT要素のインテグレーションのためのミドルウエア
- RT-Component （RTC)
  - RT-Middlewareにおけるソフトウエアの基本単位

ロボットミドルウエアについて

- ロボットシステム構築を効率化するための共通機能を提供する基盤ソフトウエア
  - 「ロボットOS」と呼ばれることもある
  - インターフェース・プロトコルの共通化、標準化
  - 例として
    - モジュール化・コンポーネント化フレームワークを提供
    - モジュール間の通信をサポート
    - パラメータの設定、配置、起動、モジュールの複合化（結合）機能を提供
    - 抽象化により、OSや言語間連携・相互運用を実現
- 2000年ごろから開発が活発化
  - 世界各国で様々なミドルウエアが開発・公開されている

従来のシステムでは…

Controller

Controller software

Robot Arm
Control software

Robot Arm1

互換性のあるインターフェース同士は接続可能

従来のシステムでは…

Controller

Controller software

Humanoid's Arm Control software

Robot Arm2

Robot Arm Control software

Robot Arm2

ロボットによって、インターフェースは色々
互換性が無ければつながらない

RTミドルウエアでは…

RTミドルウエアは別々に作られた
ソフトウエアモジュール同士を繋ぐ
ための共通インターフェースを提供

Controller software

Controller

Arm A
Control software

compatible
arm interfaces

Arm B
Control software

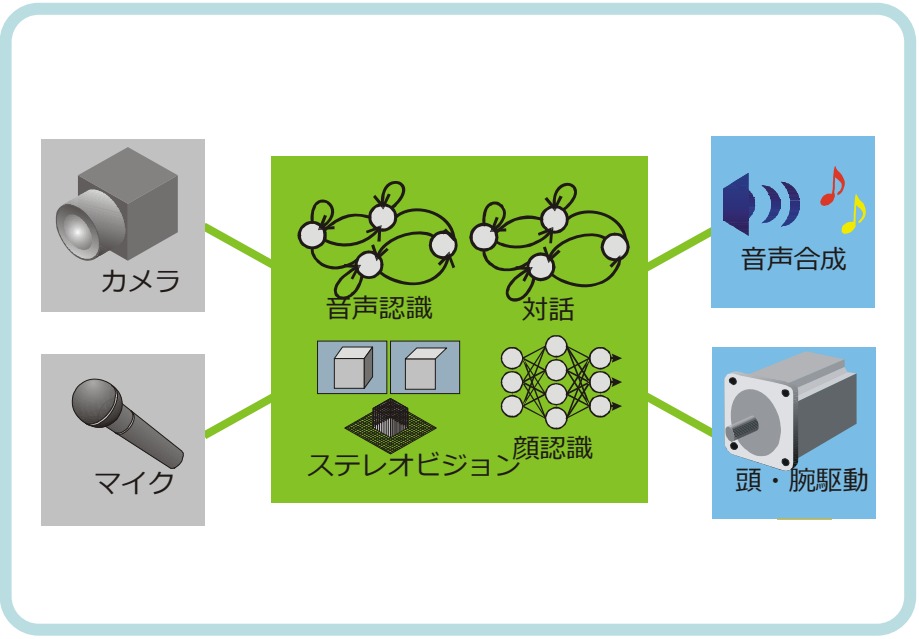
Robot Arm2

Robot Arm1

ソフトウエアの再利用性の向上
RTシステム構築が容易になる

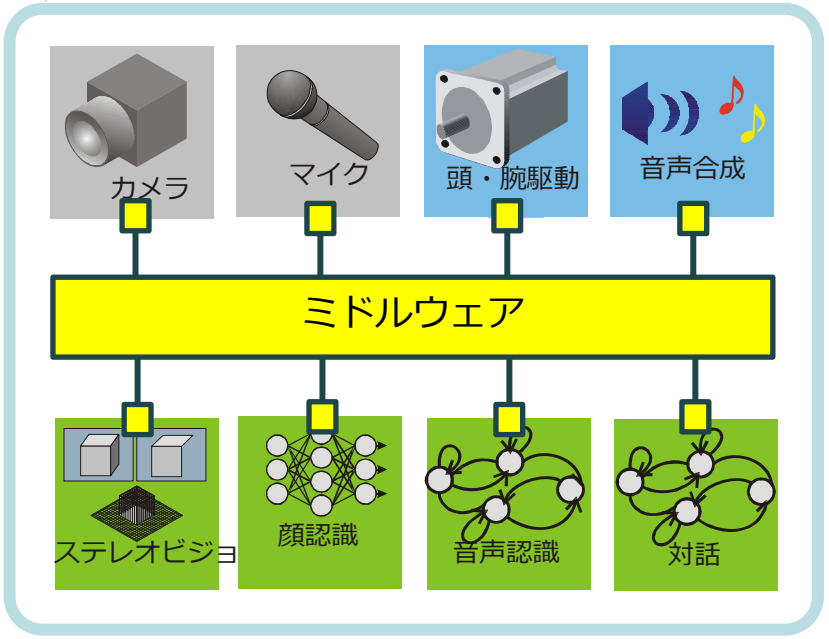
ロボットソフトウェア開発の方向

従来型開発



- ✓ 様々な機能を融合的に設計
- ✓ 実行時の効率は高いが、柔軟性に欠ける
- ✓ システムが複雑化してくると開発が困難に

コンポーネント指向開発



- ✓ 大規模複雑な機能の分割・統合
- ✓ 開発・保守効率化（機能の再利用等）
- ✓ システムの柔軟性向上

モジュール化のメリット

- 再利用性の向上
  - 同じコンポーネントをいろいろなシステムに使いまわせる
- 選択肢の多様化
  - 同じ機能を持つ複数のモジュールを試すことができる
- 柔軟性の向上
  - モジュール接続構成かえるだけで様々なシステムを構築できる
- 信頼性の向上
  - モジュール単位でテスト可能なため信頼性が向上する
- 堅牢性の向上
  - システムがモジュールで分割されているので、一つの問題が全体に波及しにくい

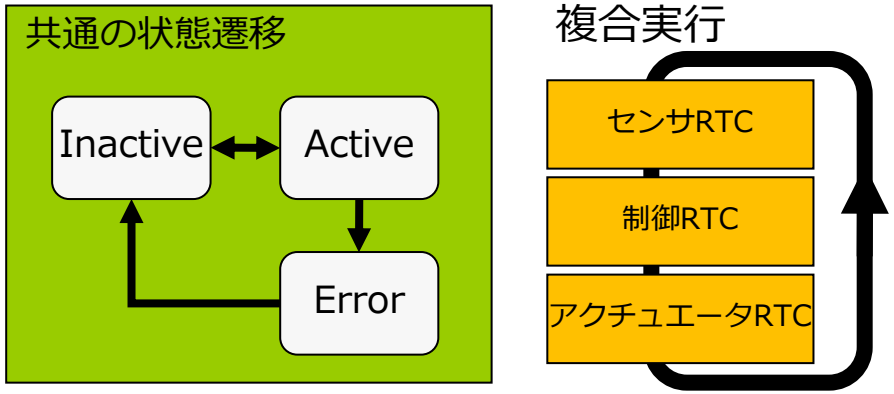
RTコンポーネント化のメリット

モジュール化のメリットに加えて

- ソフトウエアパターンを提供
  - ロボットに特有のソフトウエアパターンを提供することで、体系的なシステム構築が可能
- フレームワークの提供
  - フレームワークが提供されているので、コアのロジックに集中できる
- 分散ミドルウエア
  - ロボット体内LANやネットワークロボットなど、分散システムを容易に構築可能

RTコンポーネントの主な機能

アクティビティ・実行コンテキスト

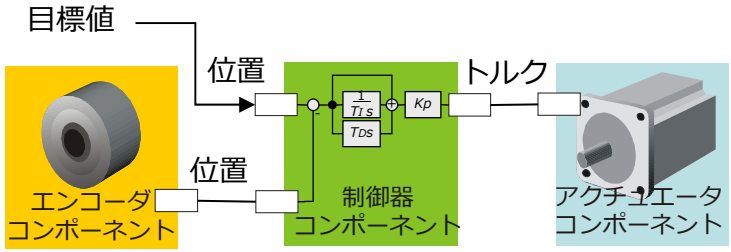


ライフサイクルの管理・コアロジックの実行

データポート

- データ指向ポート
- 連続的なデータの送受信
- 動的な接続・切断

サーボの例

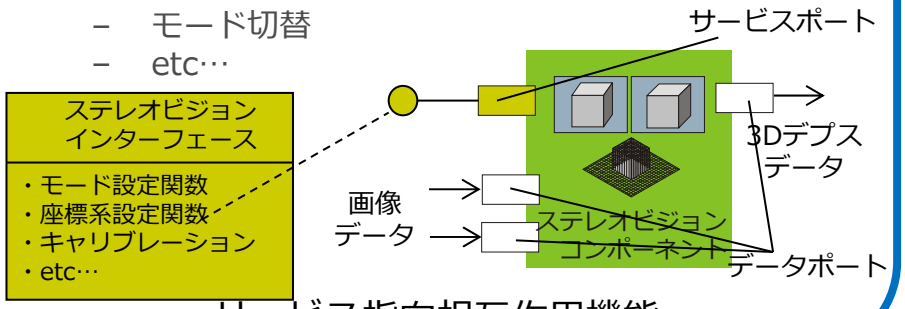


データ指向通信機能

サービスポート

- 定義可能なインターフェースを持つ
- 内部の詳細な機能にアクセス
  - パラメータ取得・設定
  - モード切替
  - etc…

ステレオビジョンの例

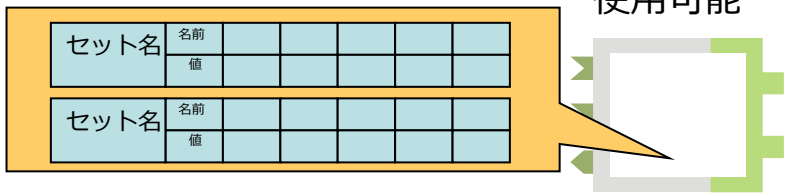


サービス指向相互作用機能

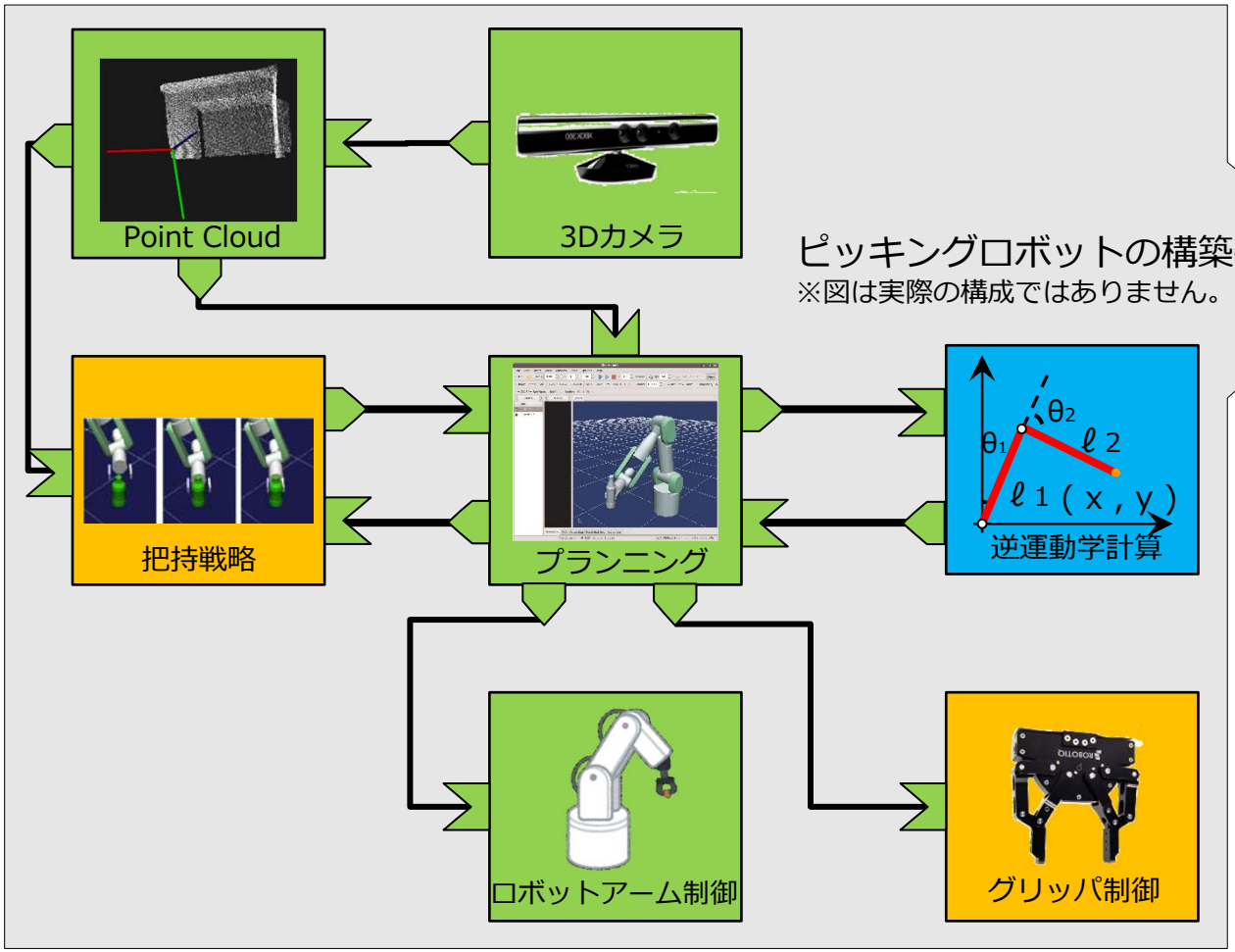
コンフィギュレーション

- パラメータを保持する仕組み
- いくつかのセットを保持可能
- 実行時に動的に変更可能

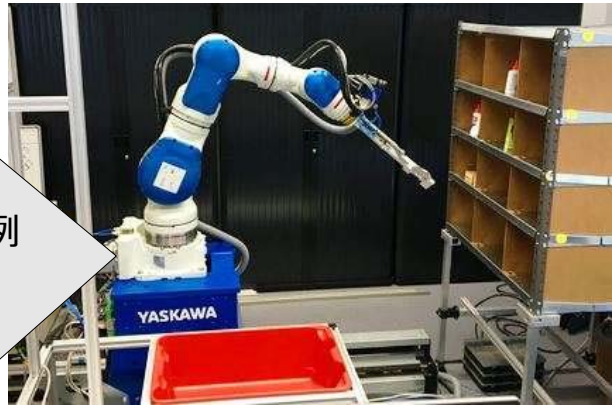
複数のセットを
動作時に
切り替えて
使用可能



ミドルウェアを利用した開発の利点



ピッキングロボットの構築例
※図は実際の構成ではありません。



ミドルウェアを利用すると、既存のモジュールが利用できる

開発するときに新規に作らなければならない部分は少なくて済む

RTミドルウエアによる分散システム

RTMにより、ネットワーク上に分散するRTCをOS・言語の壁を越えて接続することができる。

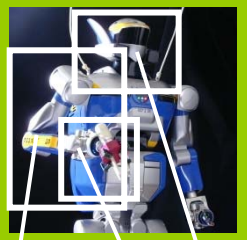
ロボットA

RTC (C++) | RTC (C++)
RTM
VxWorks

ロボットB

RTC (C++) | RTC (C++)
RTM
QNX

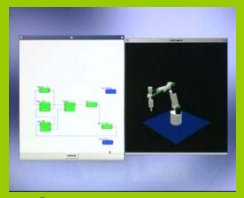
ロボットC



RTC (C++) | RTC (C++) | RTC (Python)
RTM
Real-time Linux

ネットワーク

Linux
RTM
RTC (Java) | RTC (Python)



アプリケーション

Windows
RTM
RTC (C++) | RTC (Java)

操作デバイス

uITRON
RTM
RTC (C++) | RTC (C++)

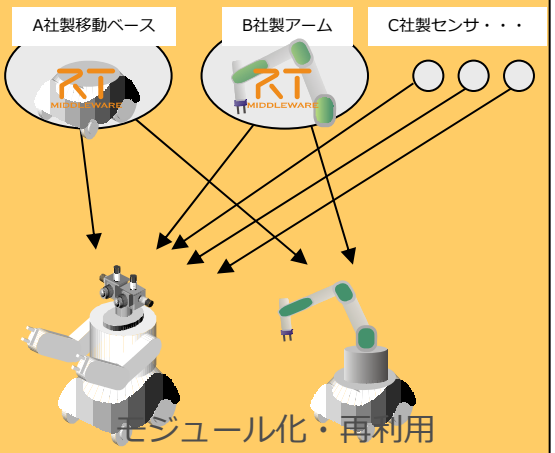
センサ

RTC同士の接続は、プログラム実行中に動的に行うことが出来る。

RTミドルウエアの目的

モジュール化による問題解決

コストの問題

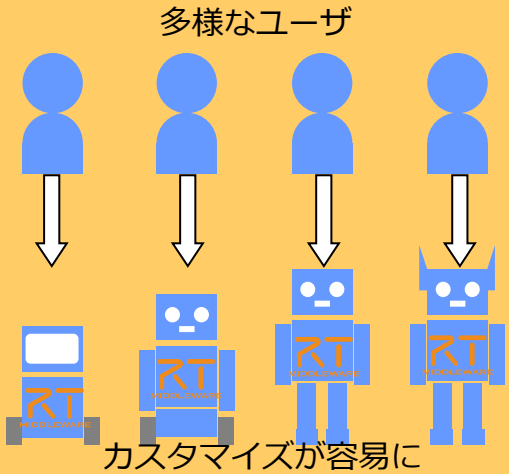


ロボットの低コスト化

技術の問題

最新技術を利用可能

ニーズの問題



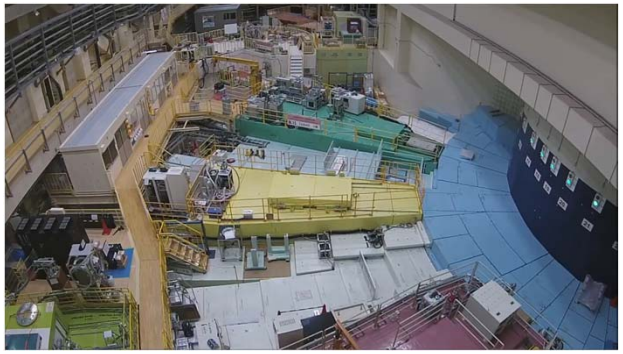
多様なニーズに対応

ロボットシステムインテグレーションによるイノベーション

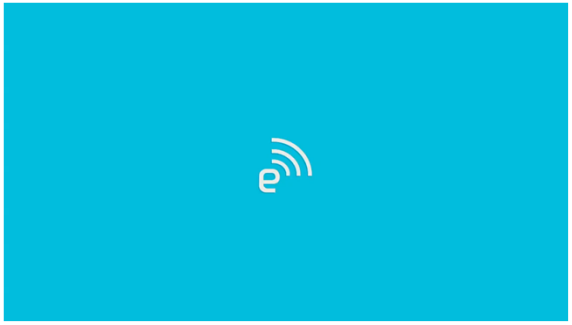
実用例・製品化例

HRP series: KAWADA and AIST

S-ONE : SCHAFT



DAQ-Middleware: KEK/J-PARC
KEK: High Energy Accelerator Research Organization
J-PARC: Japan Proton Accelerator Research Complex



HIRO, NEXTAGE open: Kawada Robotics

THK
The Mark of Linear Motion

THK: SIGNAS system

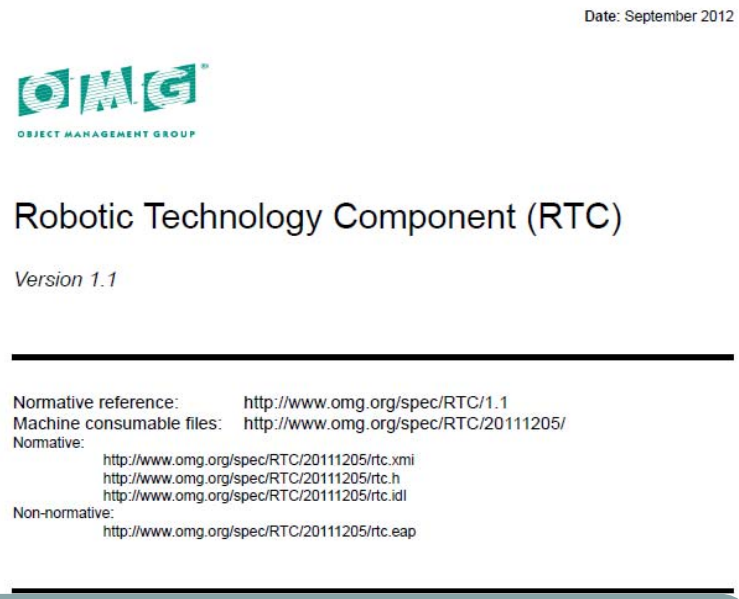
TOYOTA L&F : Air-T

VSTONE's education robots

OROCHI（RT corp.）

Robot operation simulator: NEDO

RTミドルウェアは国際標準


Date: September 2012

OMG
OBJECT MANAGEMENT GROUP

Robotic Technology Component (RTC)

Version 1.1

Normative reference: http://www.omg.org/spec/RTC/1.1
Machine consumable files: http://www.omg.org/spec/RTC/20111205/
Normative:
http://www.omg.org/spec/RTC/20111205/rtc.xmi
http://www.omg.org/spec/RTC/20111205/rtc.h
http://www.omg.org/spec/RTC/20111205/rtc.idl
Non-normative:
http://www.omg.org/spec/RTC/20111205/rtc.eap

標準化履歴
- 2005年9月
  Request for Proposal 発行(標準化開始)
- 2006年9月
  OMGで承認、事実上の国際標準獲得
- 2008年4月
  OMG RTC標準仕様 ver.1.0公式リリース
- 2012年9月
  ver. 1.1改定
- 2015年9月
  FSM4RTC(FSM型RTCとデータポート標準) 採択

OMG国際標準

標準化組織で手続きに沿って策定
→ 1組織では勝手に改変できない安心感
→ 多くの互換実装ができつつある
→ 競争と相互運用性が促進される

RTミドルウエア互換実装は10種類以上

名称	ベンダ	特徴	互換性
OpenRTM-aist	産総研	NEDO PJで開発。参照実装。	---
HRTM	ホンダ	アシモはHRTMへ移行中	◎
OpenRTM.NET	セック	.NET(C#,VB,C++/CLI, F#, etc..)	◎
RTM on Android	セック	Android版RTミドルウエア	◎
RTC-Lite	産総研	PIC, dsPIC上の実装	○
Mini/MicorRTC	SEC	NEDOオープンイノベーションPJで開発	○
RTMSafety	SEC/AIST	NEDO知能化PJで開発・機能安全認証取得	○
RTC CANOpen	SIT, CiA	CAN業界RTM標準	○
PALRO	富士ソフト	小型ヒューマノイドのためのC++ PSM 実装	×
OPRoS	ETRI	韓国国家プロジェクトでの実装	×
GostaiRTC	GOSTAI, THALES	ロボット言語上で動作するC++ PSM 実装	×

特定のベンダが撤退しても
ユーザは使い続けることが可能

ROSとの比較、動向

ROSとRTM

ROSの特長

- UNIX文化に根差した設計思想
  - Windows等Linux以外は原則サポートしない
  - ROS2ではLinux以外もサポート
- 独自のパッケージ管理システムを持つ
- ノード（コンポーネント）の質、量ともに豊富
  - OSRFが直接品質を管理しているノードが多数
- ユーザ数が多い
- メーリングリストなどの議論がオープンで活発
  - コアライブラリの仕様が固定しにくい
- 英語のドキュメントが豊富

OpenRTMの特長

- 対応OS・言語の種類が多い
  - Windows、UNIX、uITRON、T-Kernel、VxWorks、QNX
  - Windowsでネイティブ動作する
- 日本国内がメイン
  - 日本語ドキュメント・ML・講習会
  - ユーザ数が少ない
- GUIツールがあるので初心者向き
  - コマンドラインツールの種類は少ない
  - rtshell
- 仕様が標準化されている
  - OMGに参加すればだれでも変更可
  - サードパーティー実装が作りやすい
  - すでに10程度の実装あり
- コンポーネントモデルが明確
  - オブジェクト指向、UML・SysMLとの相性が良い
  - モデルベース開発
- IEC61508機能安全認証取得
  - RTMSafety

ROS1→ROS2へ

- NASAの仕事を請け負った時に、独自形式のROS messageはNASAでは使えないから、プロトタイプをROSで実装後にすべて作り直した
- NASAでは何らかの標準に準拠したものでないと使えない。その時は結局DDSを使用した。
- それ以外にもROS1では、1ノード1プロセス、コンポーネントモデルがないので、モデルベース開発にならない、ROS masterがSPOFになっているなど不都合な点が多々ある
- それ故、ROS2ではこれらの問題点を克服するため全く新しい実装にする予定。

ROS2

- 最新版：2016年12月 Beta版リリース
- 標準ミドルウェアの利用
  - 自前主義からの脱却（NASAや商用システムでは何らかの標準準拠が求められるため）
  - 通信部分はOMG標準のDDS（Data Distribution Service、OMG規格）を採用
- コンポーネントモデルを導入することにした
  - RTMのように組み込み、性能を意識したアーキテクチャへ
- 対応OSの拡大
  - これまでは、ある特定のLinux（Ubuntu Linux）のみ
  - ROS2では、Windows、Macにも対応
  - ただし、リアルタイムOS（VxWorks、QNX等）への対応はなし

OMGによる通信規格DDSを採用

航空・軍事・医療・鉄道などで実績のある通信ミドルウエア標準

http://design.ros2.org/articles/ros_middleware_interface.html

RTミドルウェアとコミュニティー

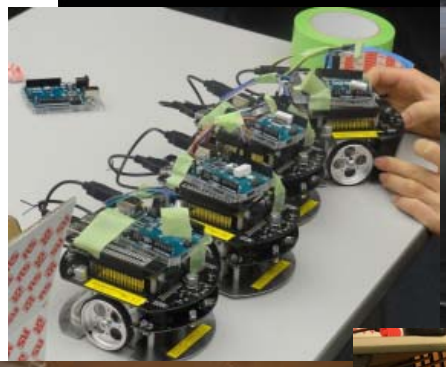
プロジェクトページ

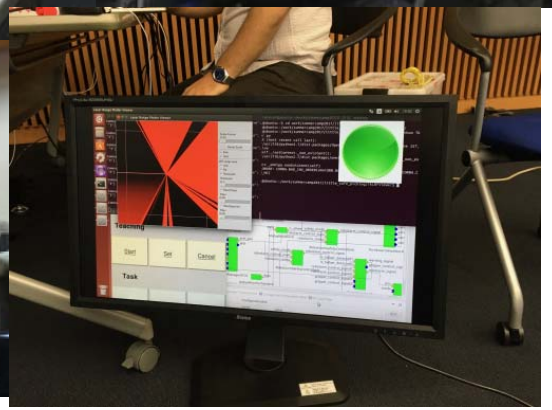
- ユーザが自分の作品を登録
- 他のユーザの作ったRTCを探すことができる

タイプ	登録数
RTコンポーネント群	405
RTミドルウエア	14
ツール	19
仕様・文書	4
ハードウエア	28

サマーキャンプ

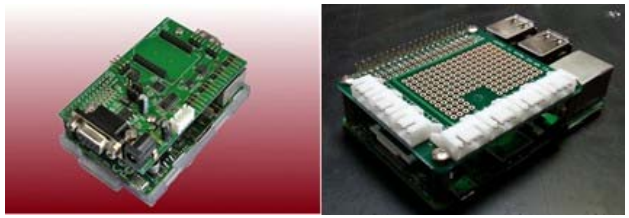
- 毎年夏に1週間開催
- 今年：8月24日～8月28日
  - オンラインで実施
- 参加人数：11名
- 場所：産総研つくばセンター
  - Online (Zoom)
- 座学と実習を1週間行い、最後にそれぞれが成果を発表
- 産総研内のさくら館に宿泊しながら夜通し？コーディングを行う！





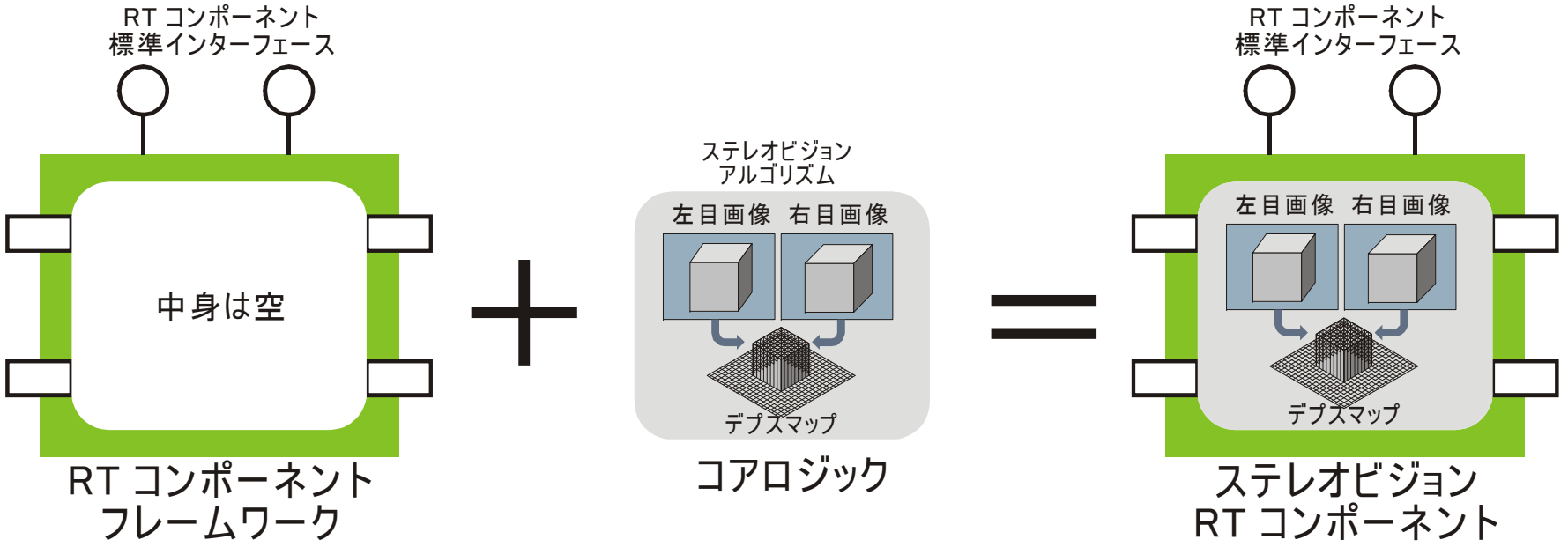
RTミドルウエアコンテスト

- SICE SI （計測自動制御学会 システムインテグレーション部門講演会 ）のセッションとして開催
  - 各種奨励賞・審査基準開示:6月頃
  - エントリー〆切：9月23日(SI2020締切)
  - 講演原稿〆切：10月26日(SI予稿締切)
  - ソフトウエア登録：10月中旬
  - オンライン審査：11月下旬～
  - 発表・授賞式：12月ごろ
- 2019年度実績
  - 応募数：11件
  - 計測自動制御学会RTミドルウエア賞 （副賞10万円）
  - 奨励賞（賞品協賛）：2件
  - 奨励賞（団体協賛）：9件
  - 奨励賞（個人協賛）：10件
- 詳細はWebページ： openrtm.org
  - コミュニティー→イベント をご覧ください



RTC開発の実際

フレームワークとコアロジック



RTCフレームワーク＋コアロジック＝RTコンポーネント

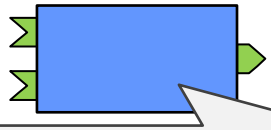
モデルに基づくコード生成

雛型にコアロジックを埋め込む

生成されたクラスの特定の関数に、必要な処理を実装する

コンポーネント仕様

name:	MyComp
category:	temp.sensor device
description:	temp. sensor RTC
comp_type:	STATIC
act_type:	PERIODIC
InPorts:	mode:TimedBool
OutPorts:	temp: TimedDouble



同一のRTC仕様からは言語が異なっていても、同じ(コンポーネントモデルの)RTCが生成される

RTCBuilder (Template code generator)

C++ backend

Java backend

Python backend

RTC source for C++

```
class MyComp
 : public DataflowComponent {
  public:
    virutal onExecute(ec_id);
     :
  private:
     TimedBool m_mo
     TimedDouble m_t
};
```

RTC source for Java

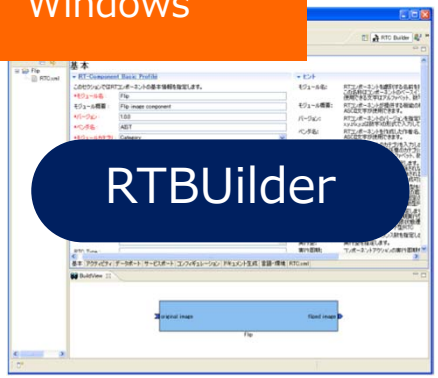
```
import RTC.DataFlowComponent;
public class MyCompImpl
extends DataFlowComponent
{
  public ConsoleInImpl(mgr)
  {
  }
  :
};
```

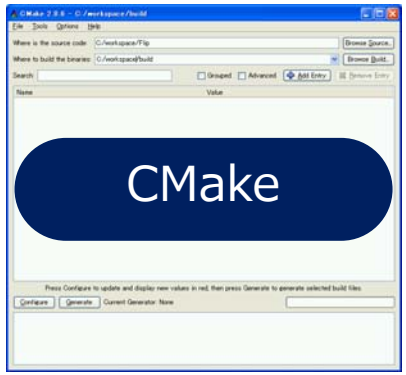
RTC source for Python

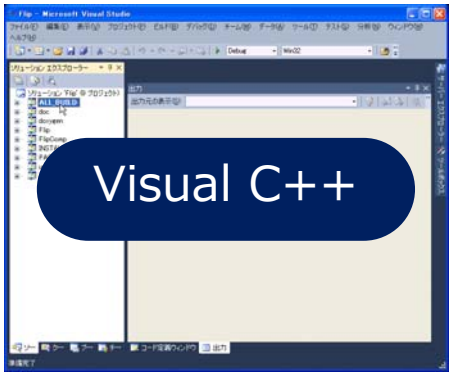
```
#/usr/bin/env python
import RTC
class MyComp(
          DataFlowComponent):
  def __init__(self, manage
    :
  def onExecute(self,
    :
```

コンポーネント作成の流れ

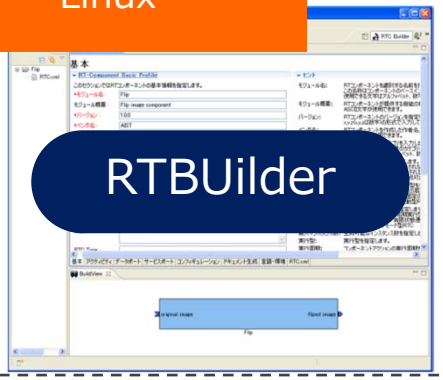
Windows

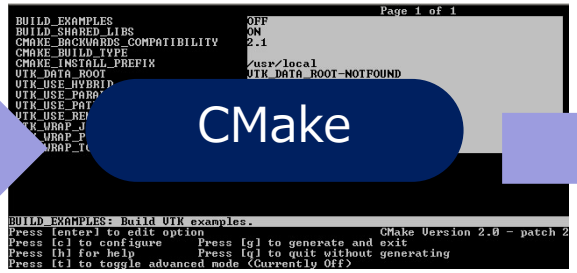


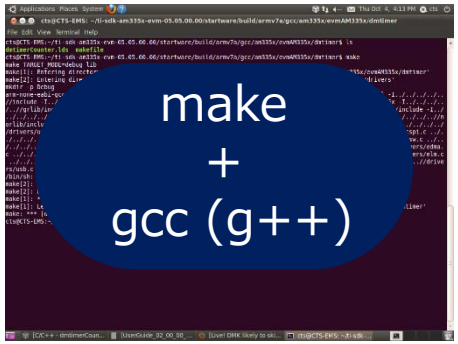




Linux







コンポーネントの
仕様の入力

VCプロジェクトファイル
またはMakefileの生成

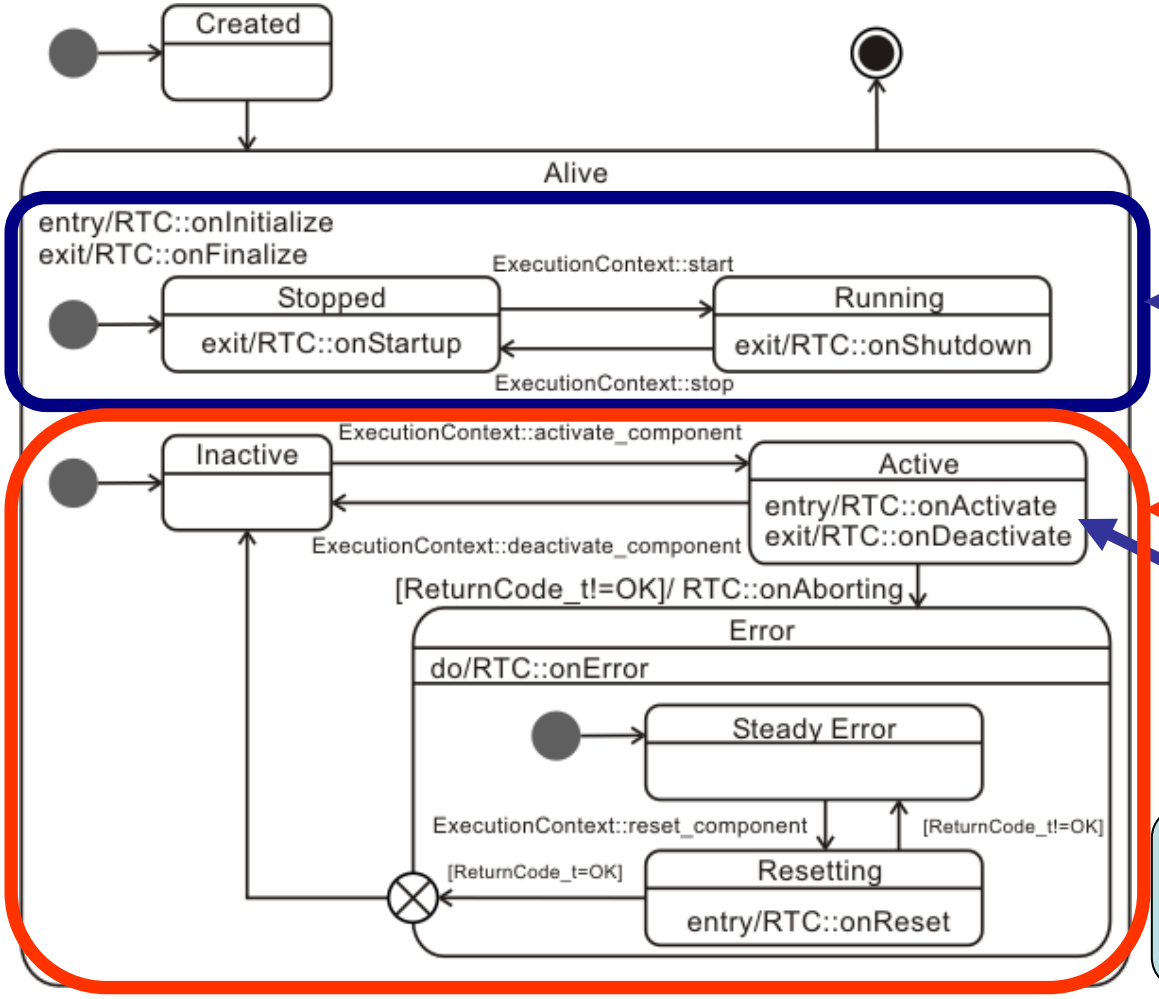
実装およびコンパイル
実行ファイルの生成

途中まで流れは同じ、コンパイラが異なる

CMake

- 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
- 様々なOS上の様々な開発環境用ビルドファイルを生成することができる
  - Linux では Makefileを生成
  - Windows ではVC(Visual C++)のプロジェクトファイルを生成
- 最近のオープンソースソフトウェアではCMakeでビルドするようになっているものが多数。

コンポーネント内の状態遷移



ユーザがあまり
意識しなくてよい部分

コンポーネント開発時に
必要な部分

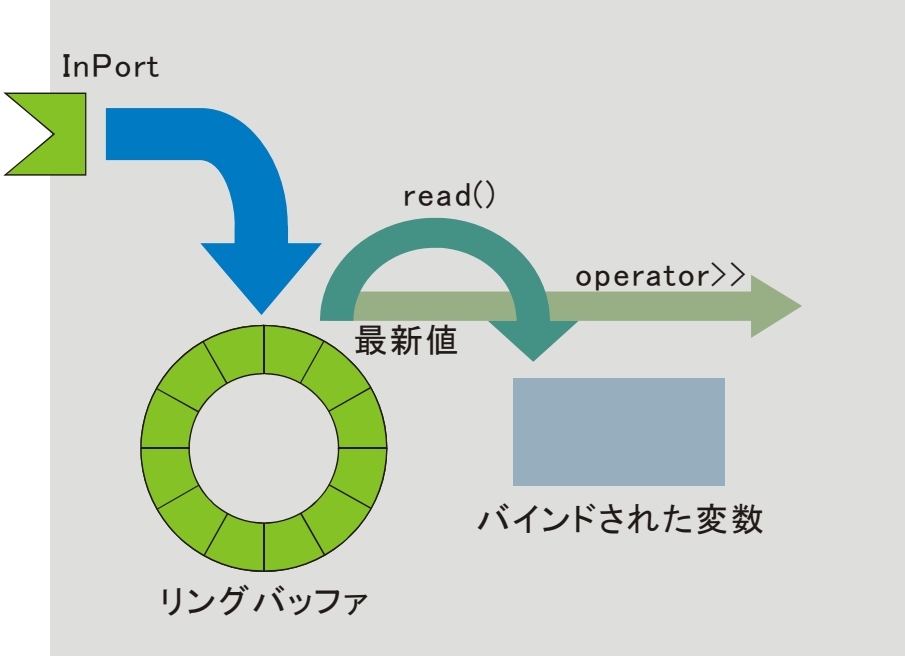
ActiveDo/RTC::onExecuteはここに入る
（DataFlow型のコンポーネントのとき）

アクティビティ

コールバック関数	処理
onInitialize	初期化処理
onActivated	アクティブ化されるとき1度だけ呼ばれる
onExecute	アクティブ状態時に周期的に呼ばれる
onDeactivated	非アクティブ化されるとき1度だけ呼ばれる
onAborting	ERROR状態に入る前に1度だけ呼ばれる
onReset	resetされる時に1度だけ呼ばれる
onError	ERROR状態のときに周期的に呼ばれる
onFinalize	終了時に1度だけ呼ばれる
onStateUpdate	onExecuteの後毎回呼ばれる
onRateChanged	ExecutionContextのrateが変更されたとき1度だけ呼ばれる
onStartup	ExecutionContextが実行を開始するとき1度だけ呼ばれる
onShutdown	ExecutionContextが実行を停止するとき1度だけ呼ばれる

InPort

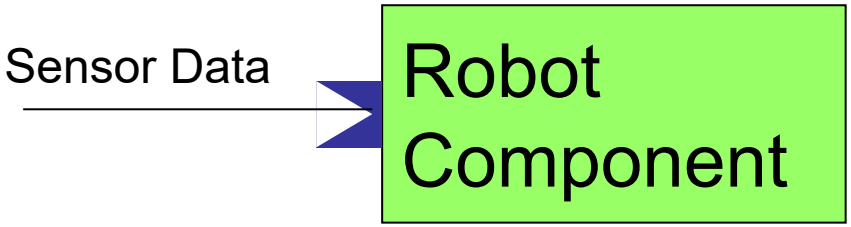
- InPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- InPortのメソッド
  - read(): InPort バッファからバインドされた変数へ最新値を読み込む
  - >> : ある変数へ最新値を読み込む



基本的にOutPortと対になる

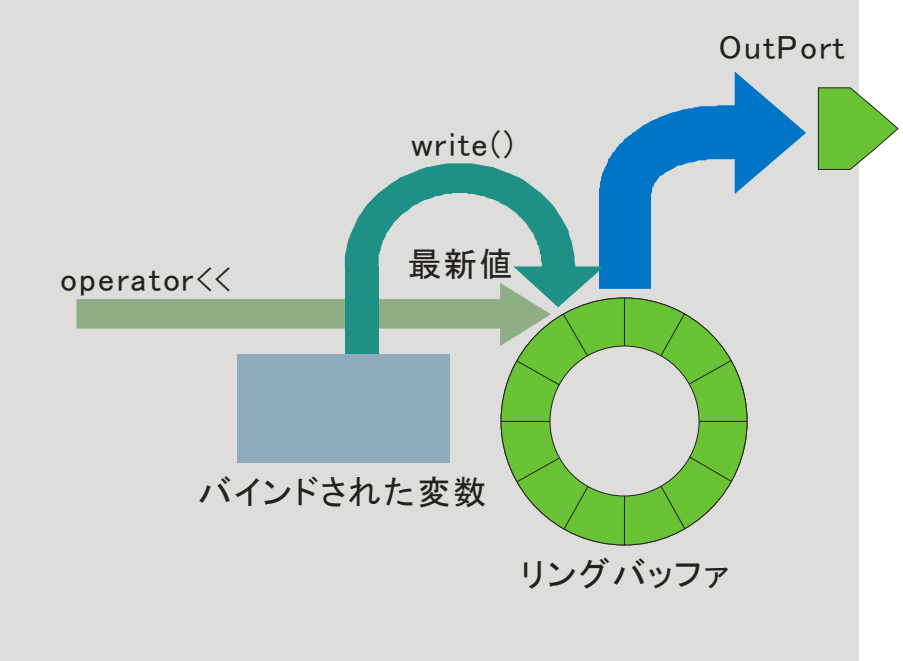
データポートの型を
同じにする必要あり

例



OutPort

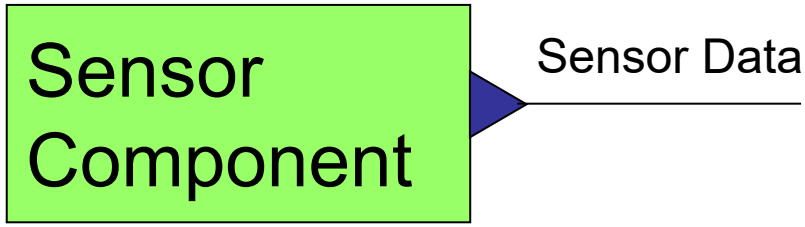
- OutPortのテンプレート第2引数：バッファ
  - ユーザ定義のバッファが利用可能
- OutPortのメソッド
  - write(): OutPort バッファへバインドされた変数の最新値として書き込む
  - >> : ある変数の内容を最新値としてリングバッファに書き込む



基本的にInPortと対になる

データポートの型を
同じにする必要あり

例



提言

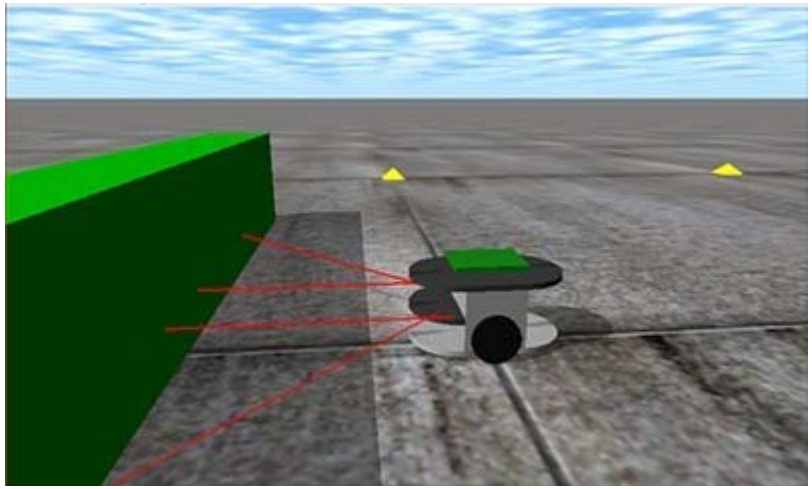
- 自前主義はやめよう！！
  - 書きたてのコードより、いろいろな人に何万回も実行されたコードのほうが動くコードである！！
  - 自分にとって本質的でない部分は任せて、本当にやりたい部分・やるべき部分のコードを書こう！！
  - 誰かがリリースしたプログラムは一度は動いたことがあるプログラムである！！
  - 人のコードを読むのが面倒だからと捨ててしまうのはもったいない！！
- オープンソースにコミットしよう！！
  - 臆せずMLやフォーラムで質問しよう！！
  - どんなに初歩的な質問でも他の人にとっては価値ある情報である。
  - 要望を積極的にあげよう！！
  - できればデバッグしてパッチを送ろう！

まとめ

- RTミドルウエアの概要
  - 基本概念
- ROSとの比較、動向
- OpenRTM-aist-1.2の新機能
- 2.0以降の開発ロードマップ
- RTMコミュニティー活動

NEXT

- Part2: RTコンポーネント開発について
  - 講師: 宮本信彦
- 開発環境の確認をお願いします。
  - OpenRTM-aist および依存ソフトウェアはインストールされていますか？
  - まだインストールしていない場合、昼休み中もスタッフがサポートします。
  - 質問がある場合は、Zoomのチャットまたは Slackで呼びかけてください。



We will create a robot controller for mobile robot and connect it to Raspberry Pi Mouse mobile robot component in the simulator.