RTミドルウエア
コンテスト2007

プレゼンテーション資料集

http://www.is.aist.go.jp/rt/RTMcontest

SICE システムインテグレーション部門講演会（SI2007）

2007年12月22日

広島国際大学 広島キャンパス ＆ RCC文化センター

2008.02.16版

測域センサデバイスの RTM化

上村 聡文

報告内容

- URG データ取得コンポーネント作成について
- OpenRTM-aist を使った感想

参加の動機

- つくばチャレンジにて作成した制御ライブラリを、RTM コンポーネント化しよう！
- まずは、URG データの取得を行うコンポーネントを作成しよう！

作成、修正したライブラリ

- 走行制御 (SH7045 によるモータ制御)
- 測域センサ (Top-URG, URG-04LX)
- ジョイスティック (USB, Wii)
- ジャイロ (秋月の方位計)
- GPS
- 経路追従

- ソースコード sourceforge.jp 公開中
- まずは、URG まわりの RTM 化

作成したURGコンポーネント概要

- OpenRTM-aist を利用
- つくばチャレンジで作成したライブラリをRTMコントローラ化
- URGシリーズの SCIP2.0 対応
- センサからの距離データを、CSV 形式文字列として、常時出力する

今思えば出力は、TimedLongSeq にするべきだったかと

インストール方法

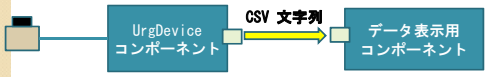
- Linux 環境であれば、configure && make でコンパイルが可能

コンパイル例

```
% tar zxvf rtm_urg-0.0.1.tar.gz
% cd rtm_urg-0.0.1/
% ./configure
% make
```

使用例

- 使用例（もしかすると、はげしく異なります）
  - 受信データのコンソール出力コンポートと接続しての表示用コンポーネントで、出力を確認



※ サービスポートの使用は断念。
IDL ファイル内で、struct の使い方がよくわからず。

OpenRTM-aist についての感想

- いろいろと、行き詰った
  - CORBA についての理解で行き詰る
  - IDL ファイルで struct を使おうとして行き詰る
- 提案
  - CORBA 初心者への、サイトの紹介とか
  - もっと、サンプル実装を
  - 作ったコンポーネントの公開場所など
  - コンパイルが遅い…

今後の予定

- 引き続き、自作ライブラリの RTM 化を行いたい
- 参考資料の充実を期待する
- 今後、問題は ML にて相談したい

屋外自律移動ロボットにおけるGPSコンポーネント

芝浦工業大学
○佐藤大介，　田中基雅，　水川真

発表内容

- はじめに
- コンポーネントのシステム設計
- ナビゲーション機能概要
- 実証実験

はじめに

屋外サービスロボット


ロボハイター
（富士重工株式会社）


田植えロボット
(中央農業総合研究センター)

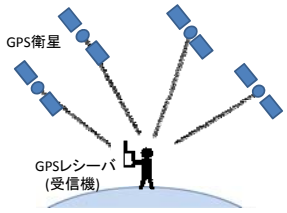

ムジロー・リグリオ
（株式会社テムザック）

独自の基本設計に基づいたナビゲーションシステムが適用されている

再利用可能なナビゲーションシステムが必要

はじめに

GPS (Global Positioning System)



・広い測位範囲
・絶対座標を取得可能
・簡易な装置で利用可能

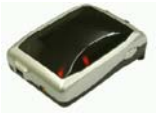
RTミドルウェアを用いて，GPSによるナビゲーションをRTコンポーネント化した

コンポーネントの設計

GPSレシーバとの接続

出力する信号は同様だが，
情報へのアクセスの方法は異なる


RS232C


Bluetooth


SDカード型

GPSレシーバとの接続機能とGPS情報処理部分を
分けてコンポーネント化した

コンポーネントの設計

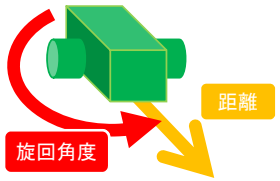
GPSレシーバとの接続

通信コンポーネントの変更で
異なるGPSレシーバに対応可能

コンポーネントの設計

出力する情報

距離・角度をナビゲーション情報として出力

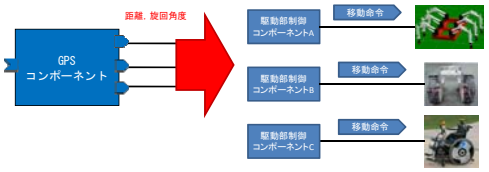


GPSに関する知識の無い開発者でも利用可能

コンポーネントの設計

出力する情報

距離・角度をナビゲーション情報として出力

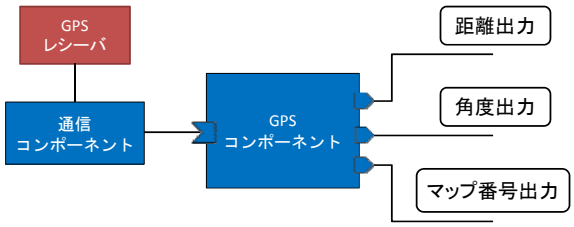


距離・角度は多くのロボットで利用可能な制御量

コンポーネントの設計

システム構成

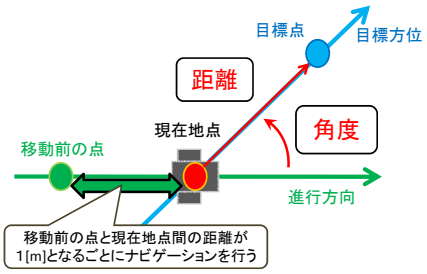
GPSからの情報を入力とし目的地へのナビゲーションを行う
コンポーネント



ナビゲーション機能概要

ナビゲーション手法

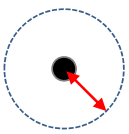
移動前の点・現在位置・目標点の3点の関係を用いる



ナビゲーション機能概要

マップデータによる経路設定

緯度・経度・次点への切り替え範囲の3つのデータを
目標点とする

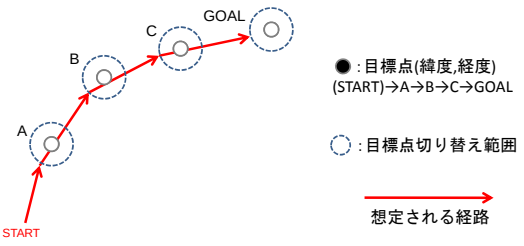


●:座標値(緯度,経度)　○:目標点切り替え範囲(半径で指定)

ナビゲーション機能概要

マップデータによる経路設定

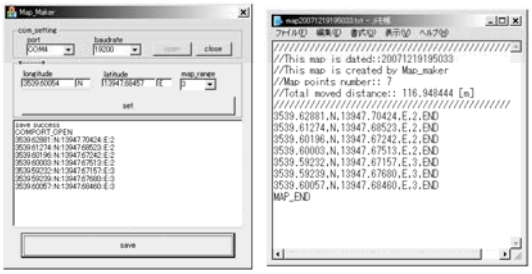
経路上に設定した目標点の点列をマップデータとして用いる



ナビゲーション機能概要

マップデータ作成ツール

「Map_Maker」を用いて容易にマップデータを作成可能



実証実験

内容

・ナビゲーションアルゴリズムの検証実験

・GPSコンポーネントの動作検証

・GPSコンポーネントのロボットへの適用実験

実証実験

GPSナビゲーションアルゴリズムの検証

本研究室で開発した屋外走行用ロボットPAR-NE07へ，GPSコンポーネントと同様のアルゴリズムを適用した．

実証実験

GPSナビゲーションアルゴリズムの検証

A100 Smart Antenna(Hemisphere)

・MSAS補正による高い精度(精度60cm)

・高速な出力周期[10Hz]

・屋外での使用を前提とした設計で雨や振動に強い

実証実験

GPSナビゲーションアルゴリズムの検証

屋外実環境での実験

・つくば国際会議場前の遊歩道

・落ち葉，路肩などがロボット走行の障害となる．

・樹木によりGPS電波の受信が困難

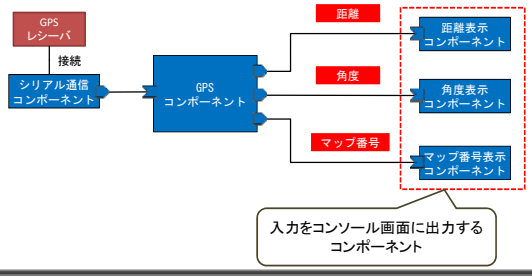
実証実験

GPSナビゲーションアルゴリズムの検証

実証実験

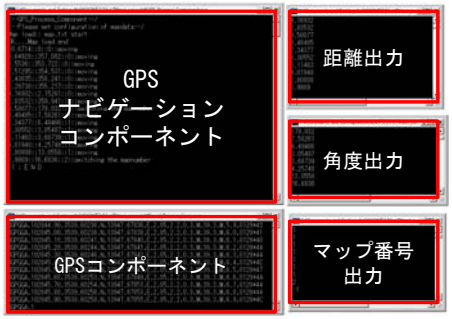
GPSコンポーネントの動作確認

GPSコンポーネントの動作を確認するための出力表示システムを構築した



実証実験

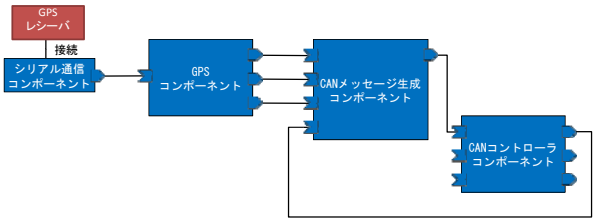
GPSコンポーネントの動作確認



実証実験

GPSコンポーネントのロボットへの適用

GPSコンポーネントとCANコンポーネントを用いて
駆動部制御のシステムを構築した



実証実験

GPSコンポーネントのロボットへの適用

GPS・ロボットにとって理想的な環境での実験

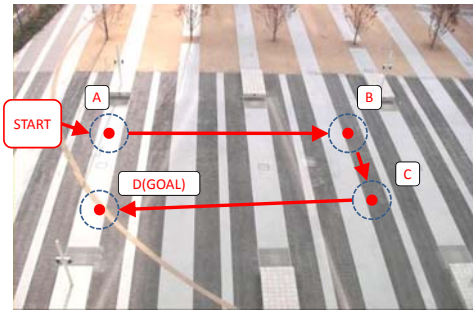
・場所は本学豊洲校舎前の広場

・路面は比較的平坦な石畳

・障害物の無い経路を設定した

実証実験

GPSコンポーネントのロボットへの適用



まとめ

公開内容

・コンポーネントのソースコード

・GPSコンポーネントマニュアル

・「Map_Maker」実行ファイル

まとめ

・GPSコンポーネントの開発背景について述べた

・GPSコンポーネントのシステム構成と機能概要について述べた

・GPSコンポーネントの実証実験を行い，実際にロボットへ適用可能であることを示した

屋外自律移動ロボットにおけるGPSコンポーネント

芝浦工業大学
○佐藤大介， 田中基雅， 水川真

分散制御ロボットにおける CANコンポーネント

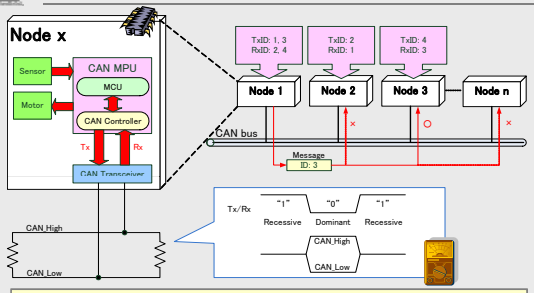
芝浦工業大学　水川研究室

○ 三浦俊宏 (Toshihiro MIURA)
水川真 (Makoto MIZUKAWA)

目次

1. CANとは

1. CANとは



通信方式採用によるメリット

- ✓ 配線本数の削減
- ✓ コントローラ(Node)の小型化
- ✓ 設計自由度の向上
- ✓ 電気的信頼性の向上
- ✓ データ通信以外への用途拡大

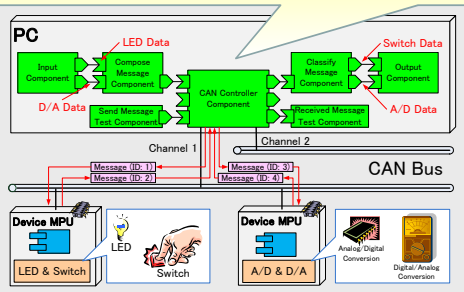
2. 目的

CAN通信技術の共有・蓄積への貢献を目的とし実装を行った.

3. CAN Componentsの概要

CAN Componentsとは？

CAN通信機能要素をRT-Component化したCAN Controller Componentとこれの動作確認及び評価を行うために作成した他のRT-Componentを含めたもの.



4.1. CAN Controller Componentの仕様

- 概要 -
2つのCANバスにCANメッセージを送受信することが可能で動的に設定できる受信メッセージのフィルタ機能を実装している.

使用機器
CAN通信機能要素をRT-Component化するにあたり，ベクター・ジャパン社製XLファミリー製品を使用しCAN Controller Componentの実装を行った.

CANcardXL

CANcardXL の特徴
> PCカード (Type II)
> 64MHzの32ビット マイクロコントローラ
> 2つの完全に独立したチャンネル
> CAN 2.0BおよびLIN
> CANcab/LINcabでのバス トランシーバ
> プラグ&プレイ

4.1. CAN Controller Componentの仕様

- Portの仕様 -
各InPort, OutPortは，CANバスに送受信するメッセージデータを入出力としており，このデータ長はCANメッセージ長を示すDLCの値によって変わる.

Port の仕様

ポート種別	ポート名称	データ型	説明
InPort	TxChannel1	TimedOctetSeq	入力されたデータをCANバスのチャンネル1に送信する.
	TxChannel2	TimedOctetSeq	入力されたデータをCANバスのチャンネル2に送信する.
	TxChannelAll	TimedOctetSeq	入力されたデータをCANバスのチャンネル1と2に送信する.
OutPort	RxChannel1	TimedOctetSeq	CANバスのチャンネル1で受信したデータを出力する.
	RxChannel2	TimedOctetSeq	CANバスのチャンネル2で受信したデータを出力する.
	RxChannelAll	TimedOctetSeq	CANバスのチャンネル1及び2で受信したデータを出力する.

InPort の仕様 (DLC=8 の場合)

データ名称	Bit数			
	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]
DLC	data[4]			
D0	data[5]			
D1	data[6]			
D2	data[7]			
D3	data[8]			
D4	data[9]			
D5	data[10]			
D6	data[11]			
D7	data[12]			

OutPort の仕様 (DLC=8 の場合)

データ名称	Bit数							
	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
	data[19]	data[18]	data[17]	data[16]	data[15]	data[14]		

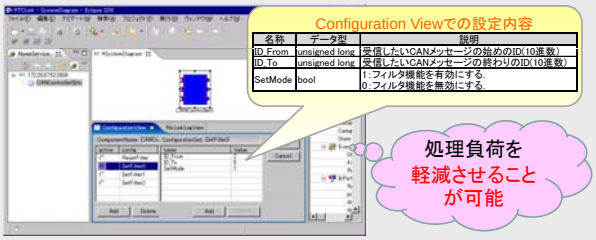
InPort の仕様 (DLC=4 の場合)

データ名称	Bit数			
	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]
DLC	data[4]			
D0	data[5]			
D1	data[6]			
D2	data[7]			
D3	data[8]			

って変化する

4.1. CAN Controller Componentの仕様

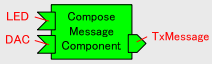
- 受信メッセージのフィルタ機能 -
RTCLink上で受信したいCANメッセージのIDを動的に設定することが可能となっている.



Configuration Viewでの設定内容

名称	データ型	説明
ID_From	unsigned long	受信したいCANメッセージの始めのID(10進数)
ID_To	unsigned long	受信したいCANメッセージの終わりのID(10進数)
SetMode	bool	1:フィルタ機能を有効にする. 0:フィルタ機能を無効にする.

4.2. Compose Message Componentsの仕様



- 概要 -
入力されたデータに対し，あらかじめ設定しておいたID及びDLCの設定を行いメッセージデータの生成を行う.

TxMessageの仕様 (DLC=8 の場合)

データ名称	Bit数			
	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]
DLC	data[4]			
D0	data[5]			
D1	data[6]			
D2	data[7]			
D3	data[8]			
D4	data[9]			
D5	data[10]			
D6	data[11]			
D7	data[12]			

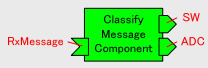
- プログラム -

```
RTC::ReturnCode_t ComposeMessage::onExecute(RTC::UniqueId ec_id)
{
                :
                :
  // ID=1, DLC=1のCANメッセージを生成してTxMessageから出力したい場合
  // (LEDのデータからCANメッセージを生成して出力する場合)
  m_LEDIn.read();
  CANTxMessageFrame CANTxMessageTmp;
  memset(&CANTxMessageTmp, 0, sizeof(CANTxMessageTmp));// 0クリア

  CANTxMessageTmp.ID = 1;          // IDの設定
  CANTxMessageTmp.DLC = 1;         // DLCの設定
  CANTxMessageTmp.Data[0].ByteAccess = m_LED.data; // Dataの設定

  m_TxMessage = ComposeTxMessage(CANTxMessageTmp); // 生成
  m_TxMessageOut.write();          // LEDのデータを出力
                :
                :
  return RTC::RTC_OK;
}
```


4.3. Classify Message Componentsの仕様



- 概要 -
CAN Controller Componentから出力されたメッセージデータを振り分けて対応するOutPortへ出力する.

RxMessageの仕様 (DLC=8 の場合)

データ名称	Bit数							
	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]				
DLC	data[4]							
D0	data[5]							
D1	data[6]							
D2	data[7]							
D3	data[8]							
D4	data[9]							
D5	data[10]							
D6	data[11]							
D7	data[12]							
MessageType	data[13]							
TimeStamp	data[20]	data[19]	data[18]	data[17]	data[16]	data[15]	data[14]	

データ名称	Bit数							
	1bit	1bit	1bit	1bit	1bit	1bit	1bit	1bit
Message Type			Tx/Rx	Channel種別				

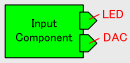
データ名称	説明
Tx/Rx	"1"の場合は送信、"0"の場合は受信したメッセージである.
Channel種別	"0"の場合は"CANcardXL Channel1"からのメッセージ. "1"の場合は"CANcardXL Channel2"からもメッセージ. "2"の場合は"Virtual Channel1"からのメッセージ. "3"の場合は"Virtual Channel2"からのメッセージ (※これはCANcardXLだけを使用した場合)

- プログラム -

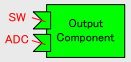
```
RTC::ReturnCode_t ClassifyMessage::onExecute(RTC::UniqueId ec_id)
{
                :
  // 受信したCANメッセージを振り分ける
  // (スイッチのデータを振り分けて出力する場合)
  CANRxMessageFrame CANRxMessageTmp;
  memset(&CANRxMessageTmp, 0, sizeof(CANRxMessageTmp));
  m_RxMessageIn.read();            // 受信したデータの読み込み
  CANRxMessageTmp = ClassifyRxMessage(m_RxMessage);  // 生成

  switch (CANRxMessageTmp.ID) // IDごとに分ける
  {
    case 2:                        // ID=2の場合
      m_SW.data = CANRxMessageTmp.Data[0].ByteAccess; // データの設定
      m_SWOut.write();             // スイッチのデータを出力
      break;
                :
  }
                :
  return RTC::RTC_OK;
}
```

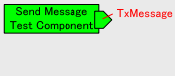

4.4. その他のRT-Compoentの仕様



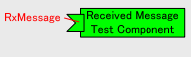
Input Component
コンソール画面で入力したLED及びD/A変換の指令値を出力する.



Output Component
入力されたデータをコンソール画面に出力する.



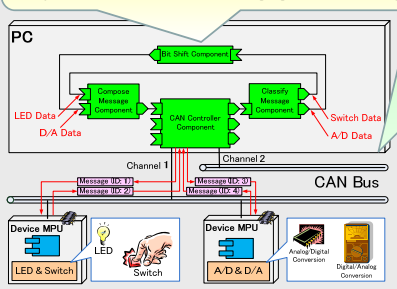
Send Message Test Component
コンソール画面で入力したID及びDLC，データからCAN Controller Componentに出力するメッセージデータの生成を行い出力する.



Received Message Test Component
CAN Controller Componentで出力されたメッセージデータをコンソール画面に出力する.

5. 動作確認

Bit Shift Component
PSoCは，A/D変換を12[Bit]でD/A変換を9[Bit]で行っている．そのため，"Bit Shift Component"では入力されたデータを3[Bit]シフトさせ（下位3[Bit]切捨て）出力している．



使用機器
◆ ノートPC (W2)
Pentium M : 900[MHz]
Memory : 512[MB]

◆ CANcardXL

◆ PSoC CAN
PSoC : CY8C29466
CAN Controller Chip : MCP2515

6. 使用例

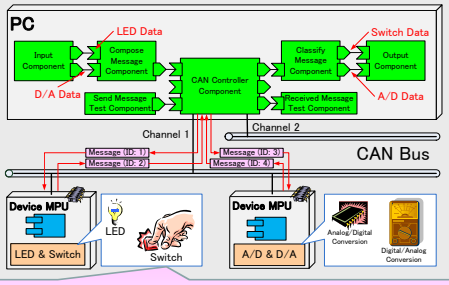

PAR-NE07

PAR-NE07 (Physical Agent Robot for Natural Environment)
屋外環境でロボットを運用するための技術獲得を目的に本研究室で開発を行い『つくばチャレンジ』に参加したロボット．
同研究室からコンテストに参加している『屋外ナビゲーションにおけるGPSコンポーネント』のRT-Componentと組み合わせることで目的地まで自律でロボットが動く．


クリスマスツリー

クリスマスツリー
クリスマスツリーをCAN通信で制御している．
LED1個ずつを周期的に制御し光の強度を変化させている．

7. 公開資料関連



公開資料
✓ CAN Componentsのソースコード ・・・ Visual Studio 2005
✓ ユーザーマニュアル ・・・ CAN Componentsの仕様・使用方法等を記載
✓ 概要資料 ・・・ CAN Componentsの概要について記載

8. ライセンス・環境について

ライセンス等ついて
CANインターフェイスを制御するAPIのソースコード等を除き，CAN Componentの著作権は，芝浦工業大学水川研究室に帰属します．
そして，APIのソースコード等の著作権は開発元の"Vector Informatik"にあります．

開発環境
- Windows XP
- Microsoft Visual Studio 2005 (VC++8.0)
- RT-Middleware (OpenRTM-aist-0.4.0)
- CANcardXL（ベクター・ジャパン社製XLファミリー製品）

動作確認環境
- Windows XP (Visual Studio 2005でコンパイル)
- RT-Middlware (OpenRTM-aist-0.4.0)
- CANcardXL（ベクター・ジャパン社製XLファミリー製品）
- CANboradXL（ベクター・ジャパン社製XLファミリー製品）

9. まとめ

◆ CAN通信機能要素をRT-Component化するにあたり，ベクター・ジャパン社製XLファミリー製品を使用しRT-Component化した．

◆ CAN Componentsの仕様・構成について述べた．

◆ CAN ComponentsのCompose Message ComponentとClassify Message Componentを作りかえることによって容易にDevice MPUの変更・機能拡張などに対応することが可能である．

◆ 以下の資料を提供することを述べた．
✓ ソースコード ・・・ Visual Studio 2005のプロジェクトで提供
✓ ユーザーマニュアル ・・・ 仕様及び使用方法等を記載
✓ 概要資料 ・・・ CAN Componentsの概要について記載

分散制御ロボットにおけるCANコンポーネント

芝浦工業大学　水川研究室

○ 三浦俊宏 (Toshihiro MIURA)
水川真 (Makoto MIZUKAWA)

PSoC CAN

PSoC CAN

PSoC CANのコンセプト

✓回路を学生が簡単に作成することが可能
✓部品は低価格であり入手性が良いものを使用
✓簡単にCAN通信をすることが可能
✓PSoCを使用しているため容易にセンサを使用したノードを作成することが可能

PSoC CANの仕様

PSoC：CY8C29466(他のPSoCにも変更可能)
CAN Controller Chip：MPC2515
CAN Transceiver Chip：PCA82C250
入力電圧：5.2～12[V]

PSoCとは

PSoCはプログラマブルアナログ・デジタル混載マイコンである.
各機能はモジュールの組み合わせにより実現可能, 動的に再構成することも可能

PSoC CAN開発について

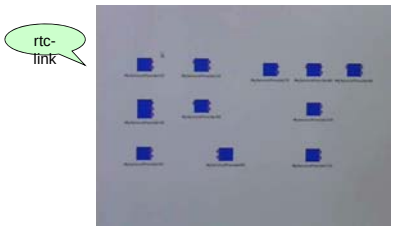
芝浦工業大学　水川研究室
指導教員:水川　真
ソフト開発:九十九里　洋介
回路設計:高木　和貴

RTコンポーネント用 コネクタモジュールの開発

菅原　隆行（筑波大）

モジュール開発背景　1/2

- RTコンポーネントの接続方法，操作方法
  『rtc-link上で操作』



モジュール開発背景　2/2

- 任意のポートを一度に接続したい
  - 現在は手動で接続
- RTコンポーネントの接続時間を短縮したい
  - デバッグ時，コンポーネントを起動する度に接続しなおすのは，時間のロス

RTミドルウェアをより使いやすいプラットフォームとするためにも，
rtc-link以外の方法でもコンポーネントを接続するモジュールがあると便利

rtc-linkと共存

開発目的

- RTコンポーネントの操作性を向上させる『RTコンポーネント用コネクタモジュール』の作成

“コネクタモジュール”コンセプト

- CORBA，pythonを意識せずに作成可能
  - 潜在ユーザに対するRTミドルウェアへの招待
- 編集するソースファイルの明確化
  - ソースファイル1つだけで作成可能
- スクリプト言語にて実装
  - ユーザがコンポーネントの接続を管理可能

今回提供するモジュール

- マニュアルを日本語と英語で提供
  - 国内 / 海外ユーザの獲得
- ライセンスフリー
  - 使用の際，ライセンスは不要.
- ソースファイルの提供
  - tar.gzファイルにてマニュアルと共に提供

開発環境

- OS：Vine3.2 (kernel 2.4.31-0v11.8smp)
  ※ Windows上でも動作確認済
- CPU：Intel Pentium D (2.8GHz)
- Memory：2.0GB
- RT-Middlewareのバージョン：
  OpenRTM-aist.0.4.1-RELEASE
- Pythonのバージョン： python02.3.4-0v18.1
- omniORBpyのバージョン：omniORBpy-2.7-1

モジュールの使用例

SimpleIOのポートを接続し，
各コンポーネントをActivateする場合

モジュール使用方法 1/4

A. ネーミングサービスの入力
B. コンポーネントリストの入力
C. 各ポートへ変数の割り当て
D. ポートの接続
E. 各コンポーネントへ
変数の割り当て
F. コンポーネントのActivate

モジュール使用方法 2/4

A. ネーミングサービスの入力

```
# Please write your naming service
#========================================
a.Connect2Nameservice("ufrg01.a02.aist.go.jp:2809")
#========================================
```

起動するネーミングサービス

B. コンポーネントリストの入力

```
# Please write your component lists in "comp_lists"
#========================================
comp_lists = { 1:'ConsoleIn0', 2:'ConsoleOut0' }
#========================================
```

起動するコンポーネント

モジュール使用方法 3/4

C. 各ポートへ変数の割り当て

```
# Please ready the connect
#========================================
con_in  = Lists[0] # con_in = Outport of 'ConsoleIn'
con_out = Lists[1] # con_out= Inport  of 'ConsoleOut'
```

変数の割り当て

D. ポートの接続

```
# Connnect
#========================================
a.PortConnect(con_in[0],con_out[0]);
```

接続する2つのポートを入力

モジュール使用方法 4/4

E. 各コンポーネントへ変数の割り当て

```
# Please ready the activate
#========================================
act_in  = Activity[0] # act_in  = ConsoleIn
act_out = Activity[1] # act_out = ConsoleOut
```

変数の割り当て

F. コンポーネントのActivate

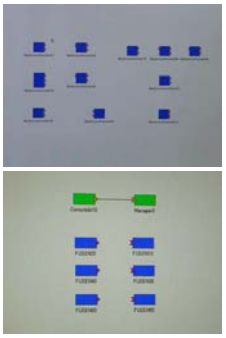
```
# Activate
#========================================
act_in[0].activate_component(Components[0])
act_out[0].activate_component(Components[1])
```

起動するコンポーネント

あくまで一例
拡張可能

モジュールの活用例

- 多数のコンポーネント接続の煩雑さを解消
- 動的なコンポーネント接続（Plug & Play）
- デバッグ時，コンポーネント接続時間を短縮

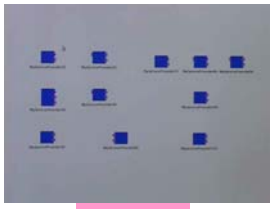


RTミドルウェアコンテスト2007

例：接続時間の比較

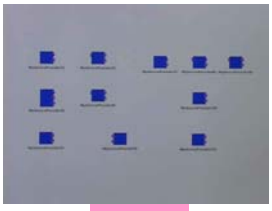
11個のコンポーネント接続の場合・・・

通常



約43 [s]

コネクタモジュール使用時



約3 [s]

RTミドルウェアコンテスト2007

コネクタモジュールのメリット

- コンポーネントの接続時間を短縮
- 多様なコンポーネントへ対応可能
- 各ユーザによる機能の拡張が可能

RTミドルウェアコンテスト2007

RTコンポーネント用コネクタモジュール

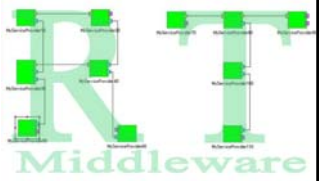
開発者：菅原　隆行（筑波大学大学院）

概要：

RTC-Linkを用いずにコンポーネント間を接続し，かつアクティビティを変更可能な，RTコンポーネント用のコネクタモジュール

特徴：

- pythonやCORBAを意識せずに，任意のコネクタモジュールを作成可能
- 1ファイルで任意のコネクタモジュールを作成可能
- モジュールのソースコードを公開



連絡先：

菅原　隆行
(t-sugawara@aist.go.jp)
(http://www.is.aist.go.jp/ufrg/index.htm)

RTミドルウェアコンテスト2007

RTコンポーネント用コネクタモジュールの開発

菅原　隆行（筑波大）

RTミドルウェアコンテスト2007

CLUEリーダコンポーネント

大原　賢一

本コンポーネントのコンセプト

- ロボットが作業をするためには、以下の情報が必要
  - 作業対象に関する知識
  - 作業対象の位置・姿勢（画像内）

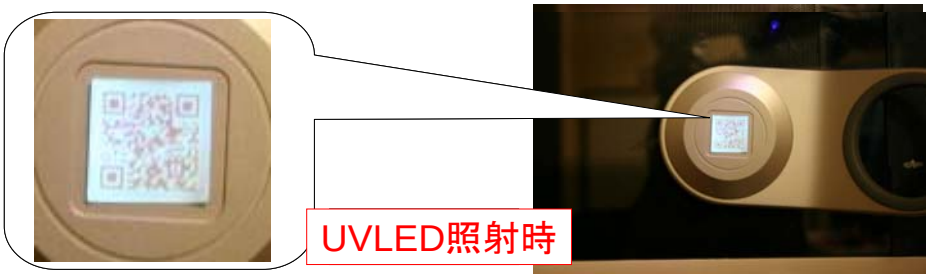
簡便な方法で上記の情報を獲得したい！

知識保存可能なマークから上記の情報を獲得し、利用可能とするコンポーネント群

CLUEとは？

- 環境に溶け込むようなロボット用のマーク
  - ロボットへの知識提供
  - ロボットへのガイド的な機能
  - 人間生活環境に浸透したマーク

CLUE　(Coded Landmark for Ubiquitous Environment)



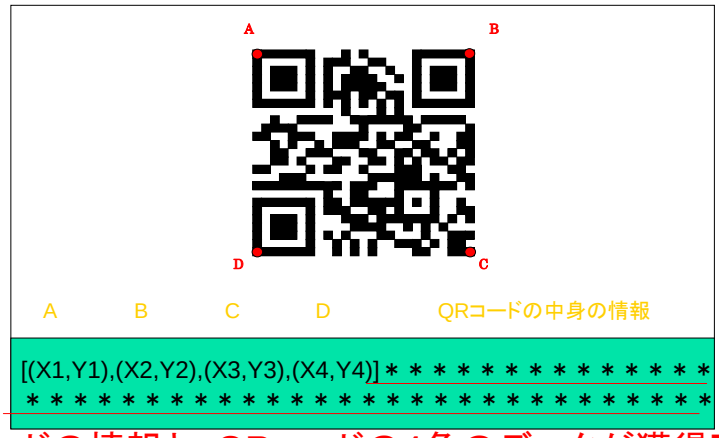
CLUEリーダ

本コンポーネントではCLUEとして、
QRコードを取り上げる。

- QRコードからの情報を獲得
- QRコード4角のデータを出力可能（標準対応）
- カメラは2種類選択可能
- ハードウェアエンコーディング。
- 環境に合わせてエンコードの前段階のパラメータを調整可能

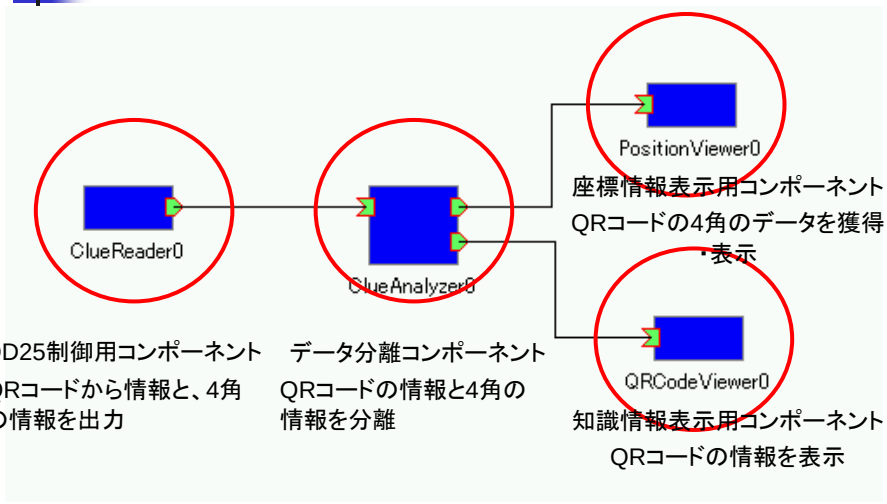
DENSO WAVE社製
QD25

CLUEリーダの特長



QRコードの情報と、QRコードの4角のデータが獲得可能
QRコードの情報のみの情報も獲得可能

CLUEReaderコンポーネント群



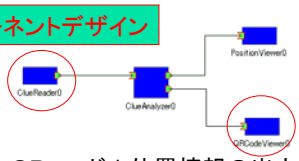
ClueReaderコンポーネント群の特長

- QRコードのみの読み込み
- QRコードの読み込み＋QRコードの4角の位置

双方を簡単な切り替えで利用可能である。
コンポーネントの再利用性を考慮にいれた
データポートの設計

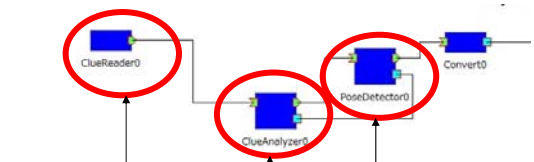
共通的に利用できるコンポーネントデザイン

QRコードのみの出力時



QRコード＋位置情報の出力時

ClueReaderコンポーネント群の利用例



QRコードリーダからの情報の読み出し

マークの位置・姿勢情報の生成

知識情報と位置情報の分離

ロボットの把持作業時の位置修正情報を
本コンポーネント群を応用して獲得している。

動作環境

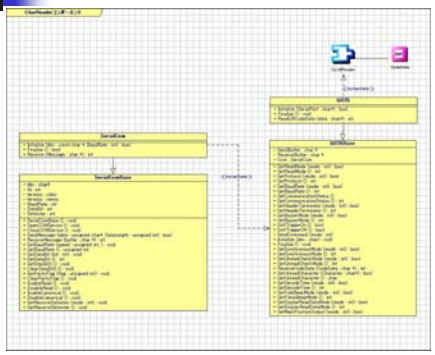
OS(Distribution)	Vine Linux 4.1
Kernel	2.6.16-0vl76.3
GCC	gcc-3.3.6-0vl7
RT Middleware Version	0.4.1
Programming Language	C++

まとめ

- 本コンポーネント群はCLUEを用いるためのコンポーネント群について示した。
- 簡易な方法によりCLUEの貼られた対象とCLUEの四角の情報から得られる相対関係を利用することで、ハンドアイシステムなどでの利用が可能である。
- 今後も後段に続くコンポーネントを適宜公開する

本コンポーネントは，文部科学省の平成18年度科学技術振興調整費による「科学技術連携施策群の効果的・効率的な推進 環境と作業構造のユニバーサルデザイン」における研究成果物である.

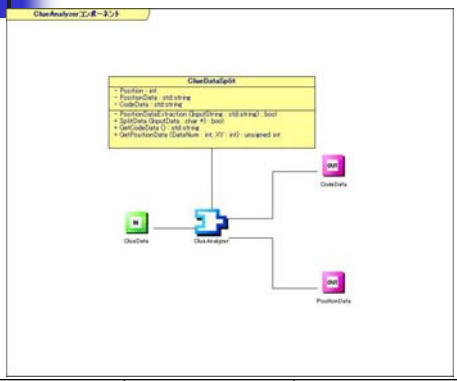
ClueReaderコンポーネント



- QD25を制御し、QRコードの情報と位置・姿勢を出力
- シリアル通信クラスとQD25制御用クラスから構成
- シリアル通信クラスを入れ替えることで、Windowsでの利用も可能と考える。

データポート	ポートネーム	型	機能
OutPort	CodeData	TimedCharSeq	ClueReaderからの出力 QRコードの情報のみ or QRコードの情報＋QRコードの4角の座標

ClueAnalyzerコンポーネント



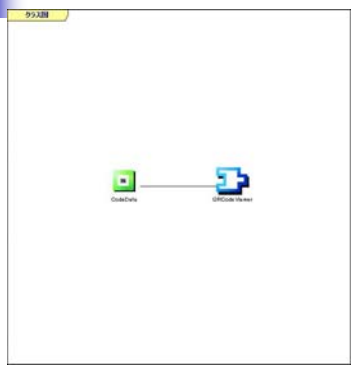
- ClueReaderコンポーネントから得られるQRコード情報とQRコードの4角のデータを分離し、それぞれの利用を可能とするコンポーネント

（ブリッジ的なコンポーネント）

- QRコード情報のみしか扱わない。

データポート	ポートネーム	型	機能
InPort	ClueData	TimedCharSeq	ClueReaderコンポーネントのOutPortと接続
OutPort	CodeData	TimedCharSeq	QRコードの中身の情報を出力
OutPort	PositionData	TimedShortSeq	QRコードの4角の座標を連続で出力

QRCodeViewerコンポーネント



•QRコード情報を入力として受け取り、表示するコンポーネント。

コンポーネントの機能拡張方法

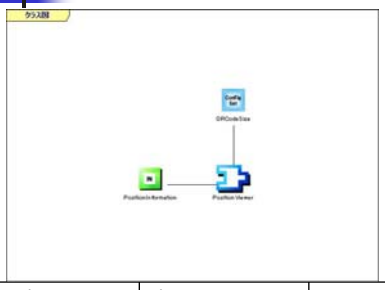
•QRコード情報のDBへの登録や問い合わせ

•QRコード情報を抽出し、後段のコンポーネントで利用

アプリケーションに合わせた使い方

データポート	ポートネーム	型	機能
InPort	CodeData	TimedCharSeq	QRコード情報の入力

PositionViewerコンポーネント



•QRコードの4角の位置を表示するコンポーネント

コンポーネントの機能拡張方法

•カメラパラメータを用いることで、カメラ座標系->実世界座標系への変換が可能

•上記の機能を持たせることで、ハンドアイシステムなどによるロボット制御への適用が可能

データポート	ポートネーム	型	機能
InPort	CodeData	TimedShortSeq	QRコードの4角の情報の入力 4点のX,Yデータ計8個のデータが入力

ConfigurationSet	
Default	0
Mark1	15
MARK2	20
MARK3	50

OpenCV を使った画像処理コンポーネントの作成例

○田窪　朋仁（大阪大学）

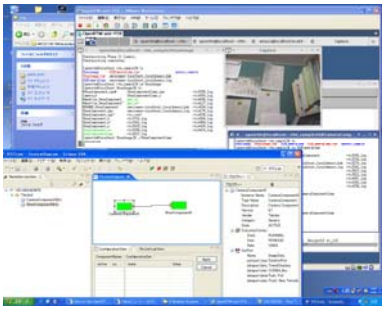
発表概要
- ・初心者のRTM導入例（開発環境紹介）
- ・OpenCVライブラリのコンポーネント化
- ・コンポーネント郡を組み合わる（実演）

初心者のRTM導入例

- 超初心者のRTミドルウェア導入例
  手軽にRTMの魅力を体験できる例題を提供
  - OpenCVライブラリを使ったUSBカメラ画像取得と映像表示
  - http://www-arailab.sys.es.osaka-u.ac.jp/~takubo/howto_rtminst.html

開発環境：
- ・産総研オフィシャルHP提供VMWareイメージ FedraCore6
- ・OpenCVをインストール
- ・USBカメラ

初心者のRTM導入例



Windows上での実行例

コンポーネントの構成

- USBカメラコンポーネント
  - 出力ポート１：TimedCharSeq でRGBカラー画像
  - 画像をキャプチャし、送信する機能のみ
- 画像表示コンポーネント
  - 入力ポート１：TimedCharSeq でRGBカラー画像
  - 画像を受け取りウィンドウに表示する機能のみ

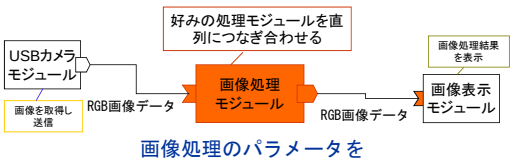
OpenCVライブラリのコンポーネント化

１３種類のコンポーネントを作成

- **カメラキャリブレーション**：カメラの内部パラメータ計算
- **背景差分：モジュール**を起動した時を基準とした差分
- **フレーム差分**：フレーム間差分により動体を表示
- **テンプレートマッチング**：テンプレートに近い画像を探索
- **オプティカルフロー**：区間画像の動きをベクトルで示す
- **ハフ変換**：画像内の直線らしき場所を検出
- **閾値処理**：カラー画像をある輝度値で２値化する.
- **回転**：画像の回転だけ．あまり使い道はない.
- **膨張と拡大**：モフォロジー処理を行える.
- **平滑化**：雑音の除去．このあとに他の処理をするといいことがあるかもしれません．エッジは甘くなる.
- **エッジ画像**：カラー画像を入れるとモノクロのエッジ画像になります.
- **グレイ画像**：カラー画像をグレイ画像に変換します.
- **輪郭抽出**：閾値処理後の画像を入れると輪郭を抽出してくれます.

OpenCVライブラリのコンポーネント化

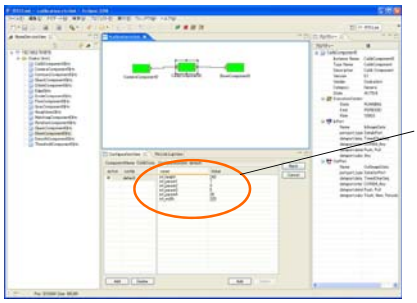
- 各モジュールのデータ送受信は画像情報のみ
  - 出力ポート１：TimedCharSeq でRGBカラー画像
  - 入力ポート１：TimedCharSeq でRGBカラー画像



画像処理のパラメータを
RTCLinkのConfiguration View
で修正可能にする

OpenCVライブラリのコンポーネント化

- RTCLinkの表示例



コンポーネント郡を組み合わる

- 実演？！

まとめ

- 特徴：
  - 広く利用されている画像処理ライブラリOpenCVをRTMコンポーネントにすることで作成例を学ぶ.
  - 複数の実用性のある画像処理を簡単に連結できる.
  - 主要な処理パラメータをRTCLinkから調整できるようにすることでライブラリの試用ができる.
  - VMWareイメージにより仮想環境を配布．すぐに試せる.

RTミドルウェアコンテスト

複数CPUのための共有メモリコンポーネント

中央大学大学院　理工学研究科
電気電子情報通信工学専攻
○　小島　隆史

HMSL

はじめに

HMSL

開発背景

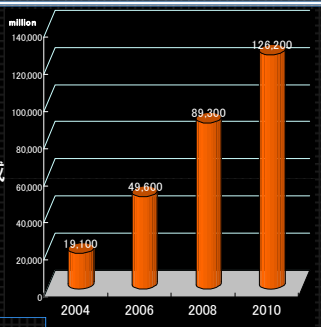
ロボット市場は今後拡大

☆AIBOの成功と撤退

開発コスト・運用コスト削減

↓

ビジネスとしての成立



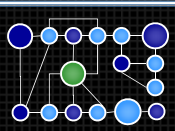
RTミドルウェアの利用

◆モジュール化　◆分散オブジェクト

モジュール化の問題点

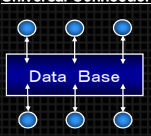
モジュール数の粒度を細かくする

・・・モジュール間の構成が複雑化



これらを引き起こしている1つの原因はデータの受け渡し

Universal Connection



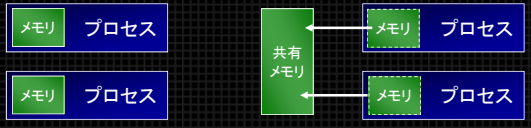
○　データの共有化
○　オーバーヘッドが少ない
○　モジュール同期が容易化
○　データの一括管理
×　モジュール量∝トラフィック

HMSL

Universal　Connection実装にむけて

共有メモリを使って実現を図る

共有メモリ：　CPUがメモリアドレス空間に確保する共有のメモリ領域



問題点　複数CPUでは、共有メモリの認識を行うことが不可能

↓

複数CPU間でも使える共有メモリモジュール

HMSL

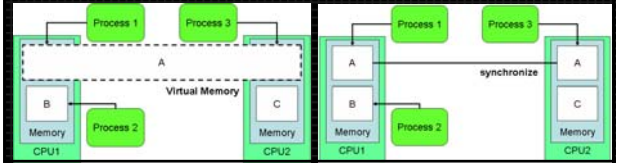
複数CPUのための共有メモリコンポーネント

● 複数CPU間を意識せずに使える共有メモリ機能

異なるCPU間で共有メモリを同期させる

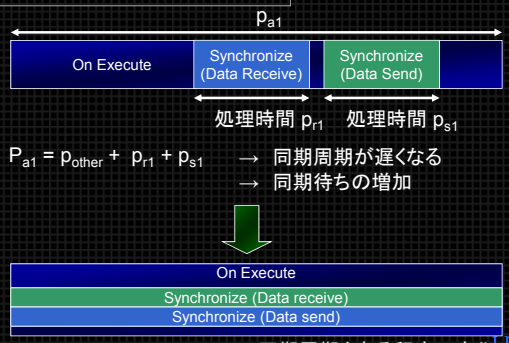
↓　仮想的に機能を実現

同期が限りなく高速におこなえれば望む機能を実現できる



同期を高速に行うために

RTコンポーネントの機能を拡張する



$P_{a1} = p_{other} + p_{r1} + p_{s1}$ → 同期周期が遅くなる
→ 同期待ちの増加

同期周期をある程度一定化

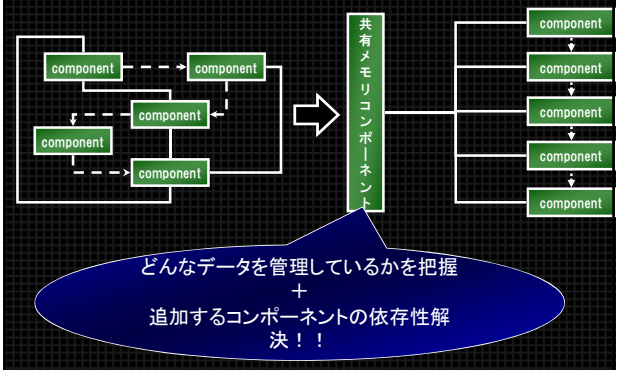
HMSL

コンポーネントの概要

～コンポーネントの導入イメージ
～コンポーネントの仕様

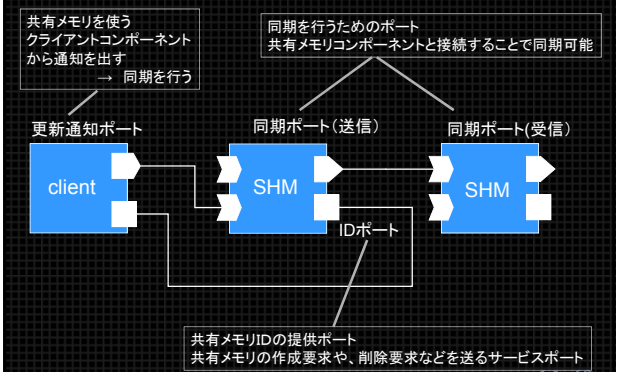
HMSL

コンポーネントの導入イメージ



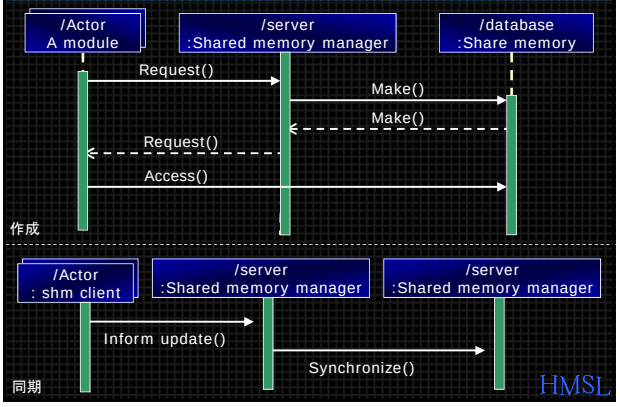
HMSL

コンポーネントの概要（2）



HMSL

シーケンス図

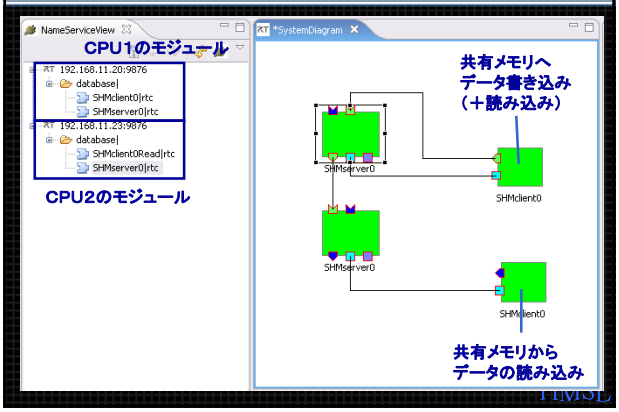


HMSL

動作実験と応用例

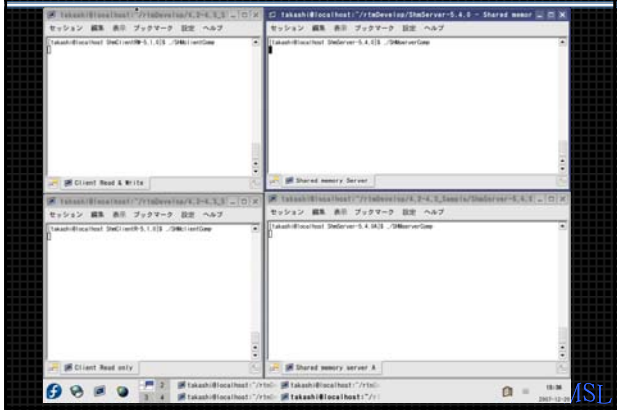
HMSL

共有メモリコンポーネントの同期実験（1）



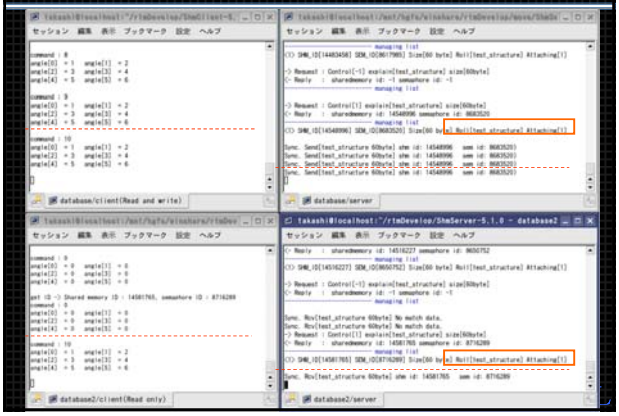
HMSL

movie

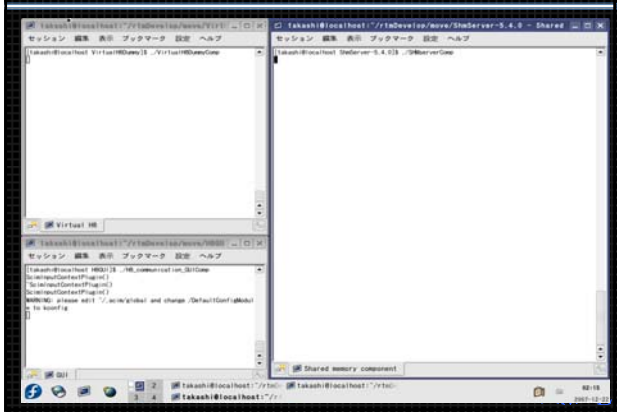


HMSL

データベースコンポーネントの同期実験の解説



コンポーネントを介した簡単なシステム



まとめと今後の課題

- まとめ
  - コンポーネントにより、コンポーネント間の依存関係を低減できた
  - 複数のCPU間で共有メモリが利用可能になった
- 今後の課題
  - モジュールの同期機能の検証と検討
    （過負荷時の挙動・同期の保証性）

HMSL

END

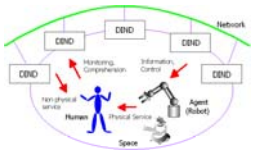
HMSL

分散レーザレンジファインダのキャリブレーション支援コンポーネント群

佐々木毅
東京大学生産技術研究所
橋本研究室

開発の背景

- 知能化空間
  - センサやアクチュエータを空間に分散配置し、ネットワーク化
- センサのキャリブレーションが必要
  - 各センサの座標系（ローカル座標系）において得られたデータを知能化空間の座標系（ワールド座標系）へ変換
  - 多数のセンサのキャリブレーションには多くの手間を要する



- 分散センサのキャリブレーション支援コンポーネント群を開発する

開発の背景

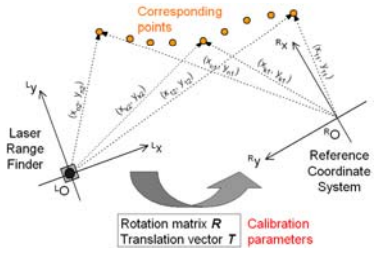
- レーザレンジファインダ(LRF)は主要なセンサの1つとして期待される
  - 設置が比較的容易
  - 知能化空間の基本機能の1つである物体トラッキングにおいても対象に特別なタグを持たせる必要がない
  - 空間の地図も獲得可能
- 本コンポーネント群では特にLRFの位置・姿勢のキャリブレーションに着目

開発環境

- OS: Ubuntu Linux 7.0.4
- RTミドルウェア: OpenRTM-aist-0.4.1-RELEASE
- コンパイラ: gcc 4.1.2
- CORBA : omniORB 4.0.7
- ACE : ACE 5.4.7-12
- Eclipse : Eclipse 3.2
- Java実行環境: Sun Java 1.5.0-11-1

LRFの位置・姿勢のキャリブレーション

- 環境中の2点以上の点について、それらのローカル座標系とワールド座標系の位置が既知ならばその対応から計算可能



開発したコンポーネント群

- LRFコンポーネント (LRFComponent)
  - 北陽電機（株）のLRF (URG-04LX)をRTコンポーネント化※
- 移動体トラッキングコンポーネント (SimpleTracker)
  - LRFのスキャンデータから移動物体の位置を出力
- キャリブレーションコンポーネント (LRFCalibration)
  - 同一物体の2つの座標系での位置を入力として受け取り、その対応点から2つの座標系の位置・姿勢の関係を出力
- 座標変換コンポーネント (CoordTrans2D)
  - キャリブレーションパラメータに基づいてセンサ座標系から基準座標系への座標変換を実行
- 入力コンポーネント (ConsoleIn2)
  - コンソールから入力した値を順に2つのOutPortlに出力

※デバイス処理には商品付属のサンプルプログラムの一部を使用

LRFCalibrationの使用例

- キャリブレーションオブジェクトを用いた絶対的な位置・姿勢の手動キャリブレーション（※）
- 移動物体を用いたLRF間の相対的な位置・姿勢の自動キャリブレーション（※）
- 移動ロボットを用いた絶対的な位置・姿勢の自動キャリブレーション
- 天井カメラの位置・姿勢のキャリブレーション

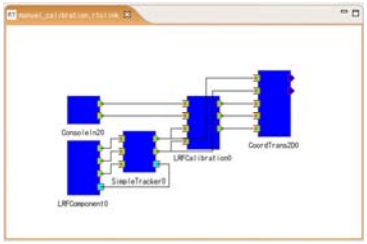
- （※）は本コンポーネント群のみで実現可能な機能

キャリブレーションオブジェクトを用いた絶対的な位置・姿勢の手動キャリブレーション

- レーザレンジファインダで検知しやすい物体を環境内の既知の位置に順番に置き、その位置情報に基づいてレーザレンジファインダの絶対的な位置・姿勢を計算
- 高精度なキャリブレーション
- 使用するコンポーネント
  - LRFComponent …… 1つ
  - SimpleTracker …… 1つ
  - LRFCalibration …… 1つ
  - ConsoleIn2 …… 1つ
  - CoordTrans2D …… 1つ

システム構築

- コンポーネントの接続



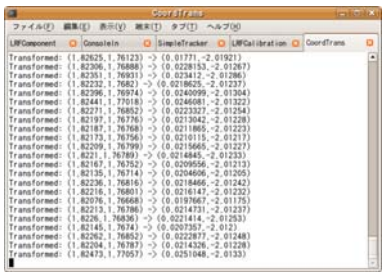
- パラメータ設定
- アクティブ化

キャリブレーション

- キャリブレーションオブジェクト（LRFで検知しやすい物体）を既知の位置に配置
- ConsoleIn2のコンソールからキャリブレーションオブジェクトの位置の座標を入力

結果の確認

- 物体を既知の位置に置き、結果を確認
  - (0, -2.0) に物体を置いた場合

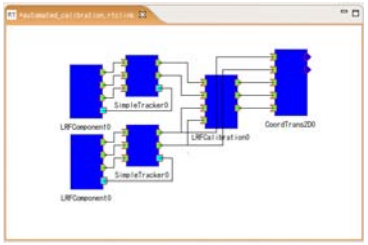


移動物体を用いたLRF間の相対的な位置・姿勢の自動キャリブレーション

- 2台のレーザレンジファインダの観測領域の重なりを利用し、一方のレーザレンジファインダを基準とした相対的な位置・姿勢を獲得
- 手動キャリブレーションと比べると精度は劣るが、簡単にキャリブレーションが可能
- 使用するコンポーネント
  - LRFComponent …… 2つ
  - SimpleTracker …… 2つ
  - LRFCalibration …… 1つ
  - CoordTrans2D …… 1つ

システム構築

- コンポーネントの接続



- パラメータ設定
- アクティブ化

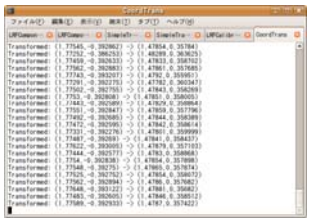
キャリブレーション

- 2台のLRFの観測領域の重なり部分で、人間やロボットなどを移動させる

結果の確認

- 物体を適当な位置に置き結果を比較

基準となるLRFでの物体の位置



キャリブレーションを行ったLRFでの物体の位置

その他の使用例

- 移動ロボットを用いた絶対的な位置・姿勢の自動キャリブレーション
  - 移動物体を用いた自動キャリブレーションでは、移動物体の絶対的な位置情報がわからないため、キャリブレーションも相対的
  - 基準となるLRFの代わりに移動ロボットの自己位置推定モジュールから得られる位置情報を利用すれば、絶対的な位置・姿勢を計算可能
- 天井カメラの位置・姿勢のキャリブレーション
  - LRFCalibrationはLRFのキャリブレーション用モジュールとして開発されたが、2次元平面上での位置を獲得するセンサであれば本コンポーネント群を同じように適用可能

まとめ

- 分散レーザレンジファインダ（LRF）のキャリブレーション支援コンポーネント群の開発
  - 各機能要素のコンポーネント化
    - LRF(北陽電機URG04-LX)
    - 移動物体トラッキング
    - キャリブレーション
- コンポーネント群の使用例の提案と動作確認
  - キャリブレーションオブジェクトを用いた絶対的な位置・姿勢の手動キャリブレーション
  - 移動物体を用いたLRF間の相対的な位置・姿勢の自動キャリブレーション

Thank you for your attention!

知的制御システム 橋本研究室
Intelligent Control System Laboratory – Hashimoto Lab.
http://dfs.iis.u-tokyo.ac.jp/

謝辞

- レーザレンジファインダコンポーネント (LRFComponent) の開発に当たりましては、東京大学生産技術研究所橋本研究室の川路浩平氏、Drazen Brscic氏、佐々木毅が作成したC++ UrgLaserクラスのコードの一部を利用しております。川路浩平氏、Drazen Brscic氏の両名に感謝申し上げます。

ライセンス（公開条件）について

- 著作権の放棄はしませんが、非商用利用であれば自由にご利用ください
- LRFコンポーネント (LRFComponent)について
  - URG-04LX付属のサンプルコードの一部を使用しているため、UrgLaserクラスはライブラリとして提供
  - 追記：成果発表会の会場にて北陽電機様よりソースコード公開の許可をいただきました

参考資料
各コンポーネントのインタフェース概説

（詳細はマニュアルを参照）

LRFComponent

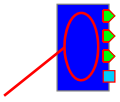
- 北陽電機（株）のLRF (URG-04LX)をRTコンポーネント化

OutPort

名称	型	説明
ScanData	TimedShortSeq	センサが獲得した距離データ [mm]
StartPoint	TimedShort	スキャン開始点（ステップ）
EndPoint	TimedShort	スキャン終了点（ステップ）

LRFComponent

- 北陽電機（株）のLRF (URG-04LX)をRTコンポーネント化

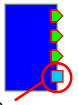


Configuration変数

名称	型	デフォルト値	説明
DeviceName	char*	“/dev/ttyACM0”	デバイスが接続されたポートの名前
ScanRate	int	19200	通信速度 [bps]
ScanStart	int	0	スキャン開始点（ステップ）
ScanEnd	int	768	スキャン終了点（ステップ）
ScanStep	int	1	ScanStartとScanEndの間のステップ数

LRFComponent

- 北陽電機（株）のLRF (URG-04LX)をRTコンポーネント化



サービスポート (provider)

関数名	引数	戻り値の型	説明
getMaxRange	なし	short	センサの最大計測可能距離[mm]を返す
getMinAngle	なし	short	最小ステップに対応する角度[deg]を返す
getMaxAngle	なし	short	最大ステップに対応する角度[deg]を返す
getMaxStep	なし	short	最大ステップを返す

SimpleTracker

- LRFのスキャンデータから移動物体の位置を出力

InPort

名称	型	説明
ScanData	TimedShortSeq	LRFComponent が出力した距離データ [mm]
StartPoint	TimedShort	スキャン開始点（ステップ）
EndPoint	TimedShort	スキャン終了点（ステップ）

SimpleTracker

- LRFのスキャンデータから移動物体の位置を出力

OutPort

名称	型	説明
X	TimedDouble	観測領域内で最大の大きさをもつ移動物体の位置のx座標 [m]
Y	TimedDouble	観測領域内で最大の大きさをもつ移動物体の位置のy座標 [m]

SimpleTracker

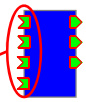
- LRFのスキャンデータから移動物体の位置を出力

サービスポート (consumer)

提供される関数についてはレーザレンジファインダコンポーネント (LRFComponent) のサービスポートを参照

LRFCalibration

- 同一物体の2つの座標系での位置を入力として受け取り、その対応点から2つの座標系の位置・姿勢の関係を出力

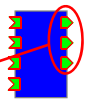


InPort

名称	型	説明
GlobalX	TimedDouble	物体の基準となる座標系での位置のx座標 [m]
GlobalY	TimedDouble	物体の基準となる座標系での位置のy座標 [m]
LocalX	TimedDouble	物体のセンサ座標系での位置のx座標 [m]
LocalY	TimedDouble	物体のセンサ座標系での位置のy座標 [m]

LRFCalibration

- 同一物体の2つの座標系での位置を入力として受け取り、その対応点から2つの座標系の位置・姿勢の関係を出力



OutPort

名称	型	説明
Tx	TimedDouble	基準座標系に対するセンサ座標系のx方向の並進量 [m]
Ty	TimedDouble	基準座標系に対するセンサ座標系のy方向の並進量 [m]
Theta	TimedDouble	基準座標系に対するセンサ座標系の回転角度 [rad]

LRFCalibration

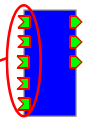
- 同一物体の2つの座標系での位置を入力として受け取り、その対応点から2つの座標系の位置・姿勢の関係を出力

Configuration変数

名称	型	デフォルト値	説明
data_num	int	0	キャリブレーションに用いる対応点の数

CoordTrans2D

- キャリブレーションパラメータに基づいてセンサ座標系から基準座標系への座標変換を実行

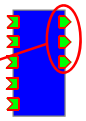


InPort

名称	型	説明
Xin	TimedDouble	センサ座標系での位置のx座標 [m]
Yin	TimedDouble	センサ座標系での位置のy座標 [m]
Tx	TimedDouble	基準座標系に対するセンサ座標系のx方向の並進量 [m]
Ty	TimedDouble	基準座標系に対するセンサ座標系のy方向の並進量 [m]
Theta	TimedDouble	基準座標系に対するセンサ座標系の回転角度 [rad]

CoordTrans2D

- キャリブレーションパラメータに基づいてセンサ座標系から基準座標系への座標変換を実行

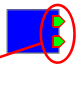


OutPort

名称	型	説明
Xout	TimedDouble	基準座標系に変換した位置のx座標 [m]
Yout	TimedDouble	基準座標系に変換した位置のy座標 [m]

ConsoleIn2

- コンソールから入力した値を順に2つのOutPortに出力



OutPort

名称	型	説明
out	TimedDouble	コンソールから最初に入力された数値
out2	TimedDouble	コンソールから2番目に入力された数値

第8回
計測自動制御学会 (SICE)
システムインテグレーション部門講演会

開発成果プレゼンテーション（2007112014）

CANコネクタ

名城大学
○ 福森　聡哲
小阪　正朋
大道　武生

1 背景

機械やロボットの機能の高度化

- 配線の数が増加
- 機器の着脱や保全に支障
- 小型化の妨げ

インテリジェントコネクタを用いた省配線システム

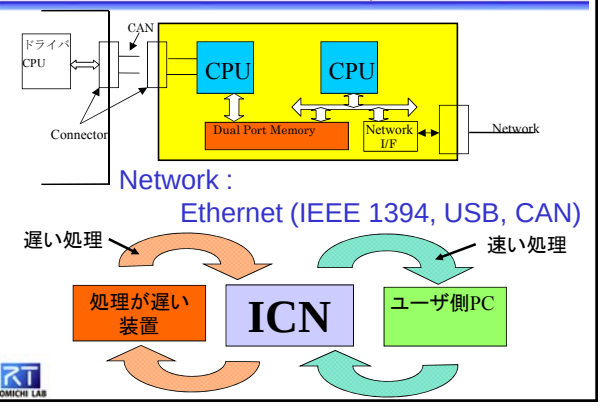
（独）中小企業基盤整備機構
“戦略的基盤技術力強化プロジェクト―ネットワークプラグインアクチュエータの開発”プロジェクト

2.1 インテリジェントコネクタの特徴

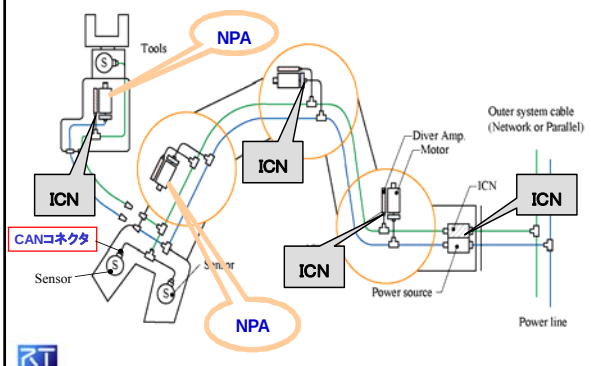
- 省配線
- ネットワークの選択が可能
  （CAN, Ethernet,IEEE1394, USB etc）
- アクチュエータやセンサ数の変更が容易
- 異種ネットワークとの接続

ICN を用いた省配線システムの一翼を担うCANコネクタの開発成果を述べる.

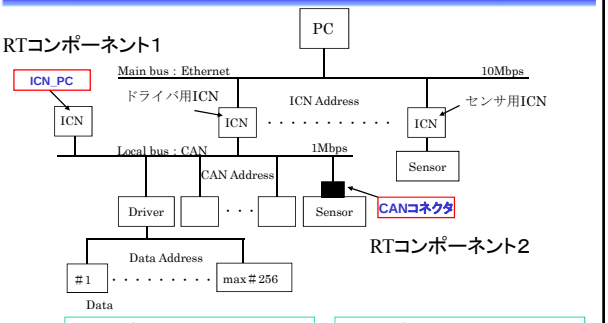
2.2 インテリジェントコネクタ（ICN）



2.3 インテリジェントコネクタのコンセプト



2.4 インテリジェントコネクタの省配線システム



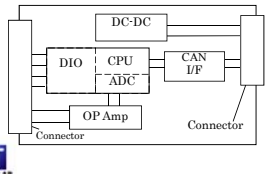
3.1 CANコネクタ

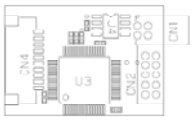
●仕様

- DIO(16bit)を用いたデジタルセンサとの接続
- A/D変換器(分解能10bit×1ch)を用いたアナログセンサとの接続
- オペアンプを用いた差動増幅回路を内蔵
- ICNとのCAN通信

基板サイズ 22×30×5mm

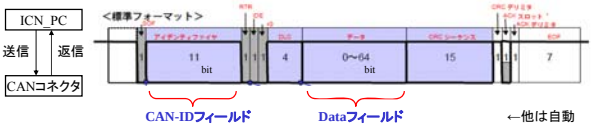
●ハードウェア





3.2 CAN-IDフィールドの構成

CAN通信における標準フォーマットのデータフレームを用いる



＜CAN-IDフィールド＞

CAN-IDのビット											
10	9	8	7	6	5	4	3	2	1	0	備考
0	CANコネクタのMAC-ID										送信
1	CANコネクタのMAC-ID										返信

MAC(Media Access Control):装置を識別する固有番号

3.3 Dataフィールドの構成

2つまで組み合わせできる

＜データ送信モード＞

MPF	Address	Data
1Byte	1Byte	2Byte

＜データアドレス送信モード＞

MPF	Address	Address	Address
1Byte	1Byte	1Byte	1Byte

＜データ返信モード＞

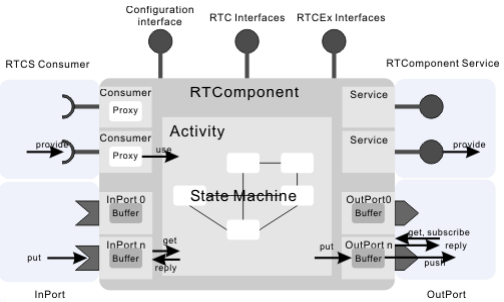
MPF	Address	Data
1Byte	1Byte	2Byte

MPF(Message Property Frame):送信データの意味付けを行う

4.1 コントローラ

●RTコンポーネント(CANコネクタ)

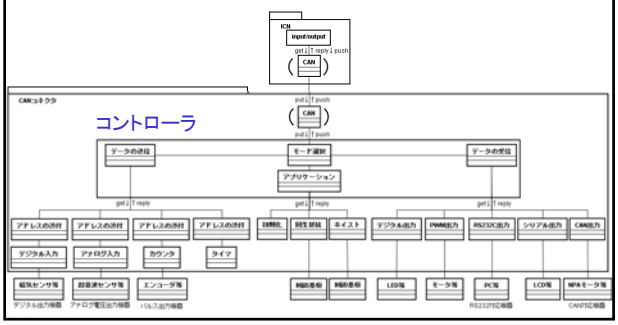
① CAN通信よりデータを受信する(get)
② 受信データの処理(解釈と実行)を行いCAN通信よりデータを返信する(push)



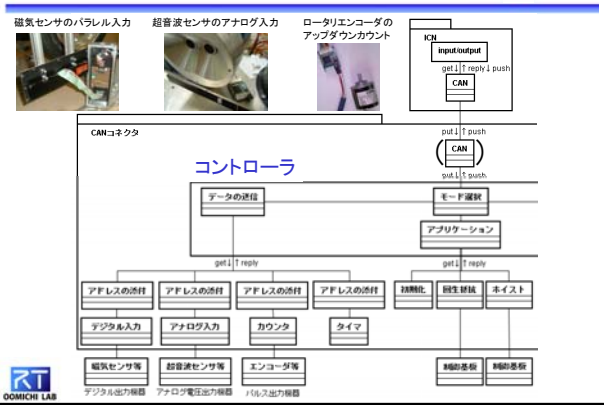
4.2 機能モジュール

●RTコンポーネント(CANコネクタ)

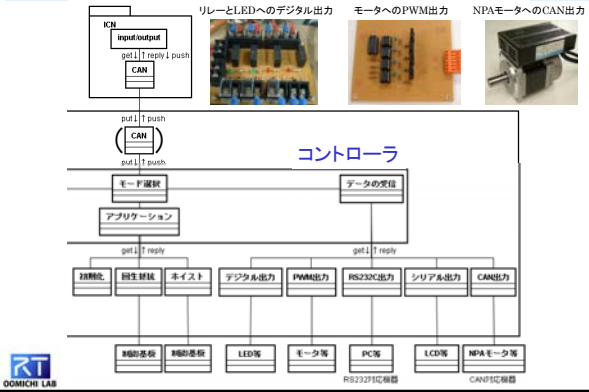
① CAN通信よりデータを受信する(get)
② 受信データの処理(解釈と実行)を行いCAN通信よりデータを返信する(push)



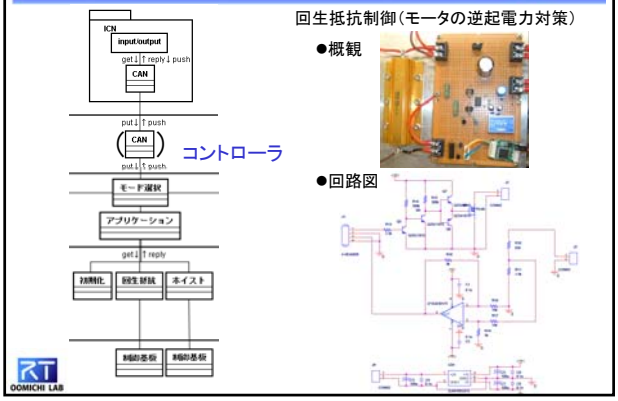
4.3.1 入力



4.3.2 出力



4.3.3 アプリケーション



5. 開発成果

ロボットに実装可能・導入容易な

CANコネクタを開発した

OOMICHI LAB

ご清聴ありがとうございました

OOMICHI LAB