CONFERENCE DIGEST

ロボティクス・メカトロニクス講演会2009

2009 JSME Conference on Robotics and Mechatronics

ROBOMEC 2009 in FUKUOKA

豊かな暮らしを創生するロボティクス・メカトロニクス

Robotics and Mechatronics for Creating an Affluent Society

太宰府天満宮本殿

Sun. 24th ~ Tue. 26th May, 2009

Fukuoka International Congress Center
Fukuoka, Japan

主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

The Japan Society of Mechanical Engineers, Robotics and Mechatronics Division

組込み Linux のための RT ミドルウエアと開発環境

RT-Middleware for Embedded Linux Systems and Development Environment

○正 安藤 慶昭 (産総研)　正 神徳 徹雄 (産総研)　学 佐々木毅 (東大)　正 橋本 秀紀 (東大)

Noriaki ANDO, National Institute of Advanced Industrial Science and Technology, n-ando@aist.go.jp
Tetsuo KOTOKU, National Institute of Advanced Industrial Science and Technology
Takeshi SASAKI, University of Tokyo, Hideki HASHIMOTO, University of Tokyo

Conventional RT-Components (RTC) require PC-level computational capacity and memory. To utilize RTC on embedded CPU based devices with other generic RTCs, the RTC-Lite component, which depends on a full-spec proxy component running on a PC, is necessary. For the module independency and autonomy, RT-Middleware should support embedded devices. In this paper, efficient RT-Component development and deployment process for the embedded Linux based system is proposed. Experimental results and an application for proposed embedded RTC are also shown.

Key Words: RT-Middleware, emmbedded system, Linux

1. はじめに

従来の RT ミドルウエア (RTM): OpenRTM-aist は PC 上の Linux または Windows OS 上で動作することを前提としており [1]，組込み機器に搭載する際には RTC-Lite と呼ばれる専用のコンポーネントフレームワークと，PC 上のプロキシコンポーネントが必要である [2].

しかしながら，モジュール化の本来の目的は，機能要素の独立性，自立性を高めることでモジュールの再利用性を向上させる点にある．したがって，組込み機器単位で独立した RT コンポーネントがそのまま動作することが望ましい．

本稿では，Linux を搭載可能な組込み CPU ボード上に RT ミドルウエアおよび RT コンポーネントを搭載するとともに，開発，配置を容易に行う方法を提案する．

2. 組込みシステム

組込みシステムはその特性から，開発，運用において汎用システムとは異なる様々な制限が課せられる．ロボット用ミドルウエアは以下に示す制約や問題点を解決し，効率的な開発を行える基盤を提供することも重要である．

2.1 ミドルウエアに対する要求

RT ミドルウエアに対して要求される要件を以下に挙げる．

2.1.1 開発の効率化

組込みソフトウエアの開発は，実行環境と開発環境が異なるクロス開発が一般的である．開発環境から CPU ボード上のフラッシュメモリ等への書込みには数十秒から数分かかることがあるため，コンパイルと実行のサイクルをいかに早く行うかが，開発効率向上に大きく寄与する．

2.1.2 設置の効率化

設置 (デプロイメント) とは開発済みのソフトウエアを実際の機器に置き，実行可能にすることである．その方法には，ネットワーク経由のもの，シリアル通信によるもの等がある．シリアル通信経由での書込みでは，機器を設置場所から取り外したり，書込み器を機器に接続する必要があるなど，プログラムの設置・更新に手間がかかることが多いため，より効率的な設置方法が求められる．

2.1.3 設定の容易さ

設置したソフトウエアは，動作させる環境に応じて各種パラメータを設定する必要がある．組込み機器は一般にモニタやキーボードを持たないことが多いため，これらの機器がなくても効率的に設定が行える方法が求められる．

2.2 ターゲット CPU ボード

本稿で使用する CPU ボード (Armadillo240) の外観を図 1 に，仕様を表 1 に示す．Armadillo240 は アットマークテクノ社から販売されている，ARM9 が搭載された Linux を実行可能な小型 CPU ボードである．

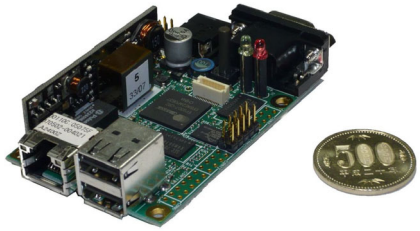

Fig.1 Target CPU board (Armadillo 210)

Table 1 Target CPU (Armadillo240) specification

Armadillo 240	
プロセッサ	EP9307 (Cirrus Logic 社製)
CPU コア	ARM920T (200MHz)
SDRAM	64MB
フラッシュメモリ	8MB
Ethernet	10BASE-T/100BASE-TX
シリアルポート	2 ポート
汎用入出力 (GPIO)	16 ビット + SW 入力 ×1
USB	2.0 (12Mbps) 2ch
基板サイズ	75.0 × 50.0〔mm〕

3. 開発プロセスの効率化

以上を踏まえて，図 2 に示す開発プロセスを定義した．

USB メモリに実行プログラムを配置することで上述した開発の効率化及び設置の効率化を実現する．変更の必要ない Linux システム本体のみ，書込みに時間のかかるフラッシュメモリ上に置き，開発時に頻繁に変更されるプログラムを USB メモリに配置することで，開発サイクルを効率化するとともに，システム設置の柔軟性が向上した．また，RTM のネットワークに関する設定は USB メモリ上の設定ファイルに，RTC の設定は Configuration 機能を利用することで，設定は容易に変更可能である．

図 2 に示すように，Linux 上で一旦 OpenRTM-aist をクロスコンパイルすることで容易に開発環境を構築することができる．また，RTC の開発自体も通常とほとんど変わらず，コンポーネントのビルド時に make コマンドに対してクロスコンパイラを以下のように指定するだけでビルド可能である．

```
$ CXX=arm-linux-gnu-g++ make -f Makefile
```

上述したように，RT ミドルウエアおよび今回提案した開発プロセスを用いることで，環境の差異をほとんど意識する

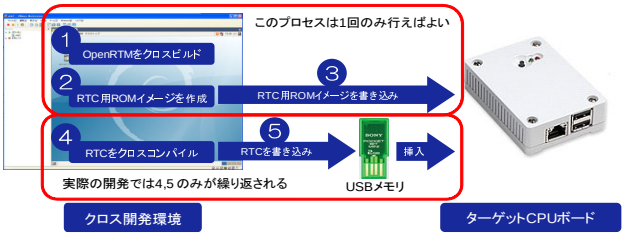

Fig.2 RTC Development process for embedded Linux

ことなく RT コンポーネントの開発を効率的に行うことができる．

4. 評価および応用例

4.1 分散 LRF ユニット

当該 CPU ボード上に，LRF (LASER range finder) センサとして北陽電機製レーザ測域センサ (URG-04LX) を搭載しユニット化したセンサデバイスを製作した (図 3 左)．CPU ボードはハブから給電可能な PoE (Power Over Ether) 電源を備えており，CPU，センサともに PoE 電源から給電可能なため，設置の自由度が高い．

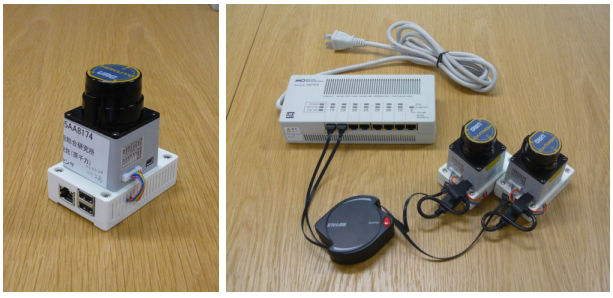

Fig.3 Distributed LRF sensor unit(left). PoE (Power over ether) hub and sensor units(right).

4.2 性能評価

性能評価を行うため，上述の LRF センサユニット上の RTC から，センサデータを PC 上の RTC 転送するときにかかる時間を計測した LRF センサは 1 回のスキャンで最大 768 点のデータを取得可能であるため，データ要素数を 1, 10, 100, 768 とした．計測された転送時間を表 2 に示す．

Table 2 Data transfer time (Armadillo240 to PC)

要素数	最小 [ms]	最大 [ms]	平均 [ms]	標準偏差 [ms]
1	1.69	1.75	1.72	0.0157
10	1.74	1.86	1.78	0.0405
100	1.78	1.88	1.82	0.0345
768	1.97	2.08	2.04	0.0407

LRF センサの仕様上のスキャン周期は 100[ms] であるので，データ転送時間は 2% 程度と，許容可能な範囲である．

4.3 システム構成例

分散センサの計測に基づき人間等にサービスを提供する知能化空間:インテリジェントスペースが提案されている．インテリジェントスペース内の物体トラッキングのため，複数のレーザレンジセンサによる計測手法を提案した [3]．

複数のセンサを用いる場合は，設置の際にセンサ位置のキャリブレーションが必要となるが，[3] では，図 4 に示すように，環境中の 2 点以上の点について，対応関係からセンサ間の相対位置を計算し，自動的にキャリブレーションを行っている．

自動キャリブレーションシステムは図 5 上部に示すように，LRFComponent (×2), SimpleTracker (×2), LRFCalibration (×1), CoordTrans2D (×1) の 6 個のコンポーネントから構成される．図 5 下に示すように，人物の位置を表す二つの点 (赤・緑) がキャリブレーションが行われる (左図から右図) に従って近づいていることが分かる．

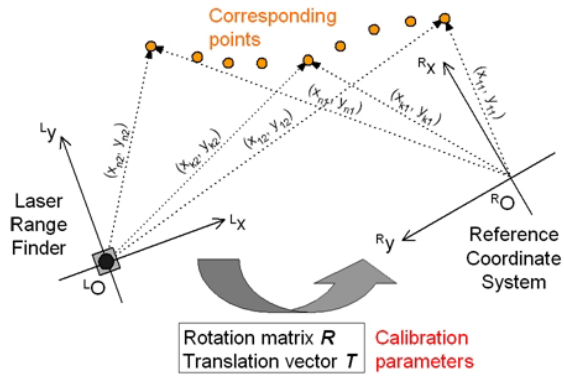

Fig.4 The LRF automatic calibration algorithm. Relative

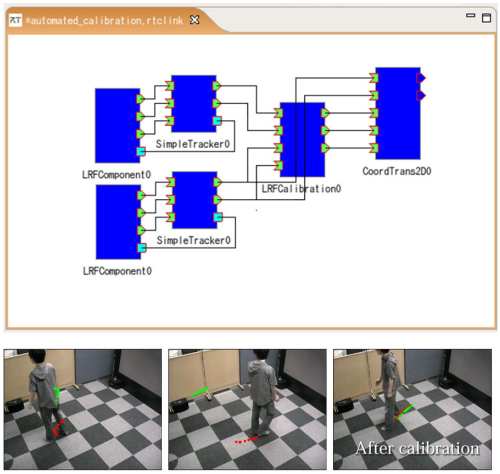

Fig.5 LRF automatic calibration and tracking system based on OpenRTM-aist.

5. おわりに

本稿では，組込みシステムにおける RTC 開発効率化について議論し，組込み RTC 開発環境を提案した．RT ミドルウエアの特徴を生かし，開発サイクルの効率化とシステム設置の柔軟性向上を実現した．また，PoE を利用した配置の自由度の高い RTC センサユニットを製作し，これを分散 LRF による人トラッキングシステムに応用し，組込み Linux ベースの RTC が有効に動作することを示した．

謝辞

本研究の一部は平成 20 年度，NEDO 次世代ロボット知能化技術開発プロジェクト (平成 19〜23 年度) の補助を受けた．

[1] OpenRTM-aist Web page, http://www.openrtm.org

[2] 安藤 慶昭, 鈴木 喬, 大原 賢一, 大場 光太郎, 谷江 和雄, "組込機器のための軽量 RT コンポーネント：RTComponent-Lite", 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 2005 (SI2005), p.3C2-2, 2005.12, 熊本

[3] Takeshi Sasaki, Hideki Hashimoto, "Hierarchical Framework for Implementation of Intelligent Space", Proceedings of the 33th Annual Conference of the IEEE Industrial Electronics Society (IECON '07), pp.28-33, 2007.11