組込み用 Linux ボード，BeagleBone Black における RT ミドルウェアの動作と開発支援ツール

○村上青児(筑波大学/産総研)　安藤慶昭(産総研)　関山守(産総研)
鍛冶良作(産総研)　谷川民生(産総研)　神徳徹雄(産総研)

1. はじめに

現在の組込みシステムの開発は専門性が求められるため開発の敷居が一般的に高い．これに対し，ロボットの要素技術をより開発者が使いやすいと感じる仕組みを作り，ロボット開発を行う人口を増やすことでロボット産業を活発にすることを目指す．

本論文では，(独)産業技術総合研究所で開発が進められている RT ミドルウェア(RT-Middleware:RTM)を利用し，組込み機器向けの開発を効率的に行う開発環境の構築を目指す．最近発売された Texas Instrument 社の BeagleBone Black を例に取り，開発フローを検討，問題点を抽出した．同ボード上での RT ミドルウェアの動作を検証するとともに，今回は組込み機器等ヘッドレスシステム(キーボード，マウス，モニタがないシステム．) の開発・運用時に問題となる点を解決する方法を提案する．

2. RT ミドルウェアと小型 Linux ボード

RT ミドルウェアは分散されたロボット機能要素を統合・管理するための開発支援ツールである[1]．分散されたロボット要素のモジュールを RT コンポーネント(RT-Component:RTC)と呼び，コンポーネントはネットワークを通して管理することができる．このような開発者支援ツールを用いることで，大幅に開発者の敷居が下がると期待される．現在 RTM は多様なプラットフォームで動作するように開発され，組込み機器などでも多数動作実績が報告されている[2,3]．

これまで，小型 Linux ボードにおいての RT ミドルウェア動作実績は，Armadillo や RaspberryPi の報告がある[4,5]．このような組込み型の小型 Linux ボードは，センサモジュールや，カメラからの出力をオンボードで処理することができ，さらにネットワークに容易に接続できるというメリットがある．また，ロボット技術を扱うためには組込み機器の専門知識が必要とされるが，このような汎用 OS を使っていることで，組込みの専門知識が無い人も取り込むことが出来ると期待できる．そのため，今後ますます組込み Linux ボードの使用が拡大すると期待される．

図 1 BeagleBone Black

本論文では最近発売された組込み小型 Linux ボード，BeagleBone Black において RT ミドルウェアを導入し評価を行った．図 1 に示すものが BeagleBone Black である．BeagleBone Black はプロセッサアーキテクチャが Coretex-A8，動作周波数 1GHz，RAM 512MB を持ち，$45 ドルと安価に手に入るため，コスト面においても開発の敷居を下げることに貢献すると考えられる．今回はユーザーの敷居を低くし開発に集中できるように，工場出荷状態でインストールされている Angstrom Linux を利用することを前提とした．

3. 小型組込み Linux ボードにおける開発

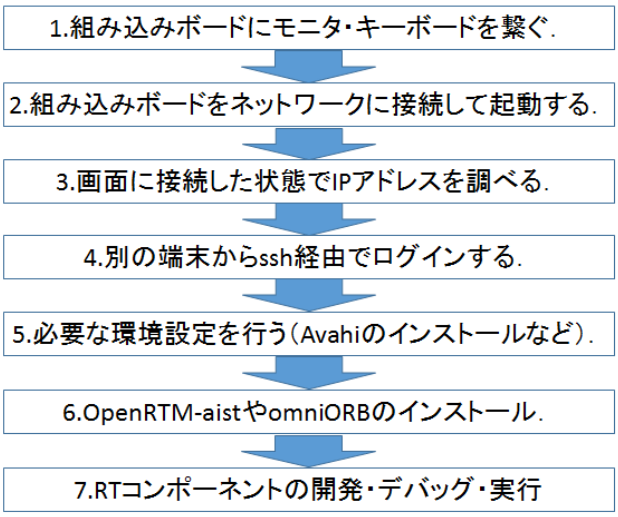

図 2 通常の組込みボードにおける開発手順

通常，組込み Linux ボードを開発する際には図 2 のような手順を踏む．このような開発手順は煩雑で時間もかかるため，これを解決する方法について考える．

3.1 開発効率化に関する考察

ユーザーの敷居を下げるという観点から，ネットワークにBeagleBone Blackを接続したその場から，RTミドルウェアの実行環境をセットアップできる必要がある．上述の開発プロセスにおいて，とくに煩雑である部分は主に以下の3つである．

1. BeagleBone Blackにモニタ・キーボードを接続しIPアドレスを確認する手順
2. OpenRTM-aistと依存ライブラリをコンパイルする手順
3. 開発したRTコンポーネントをsshログインして起動する手順

BeagleBone BlackはDHCPでヘッドレス状態でも自動的にIPアドレスを取得しネットワークに接続するが，PCなどからログインし開発するには結局BeagleBone Blackにモニタ・キーボードを接続してIPアドレスを確認する必要がある．また，BeagleBone Blackは比較的高速な組込みボードではあるが，PCと比較すれば低速であり，OpenRTM-aistや依存ライブラリ(omniORB等)のコンパイルには数時間を要する．また，RTコンポーネント開発後に実際に運用する際にはやはりRTコンポーネントをsshログインして起動する必要がある．今回はこれらの問題を解決し図3の開発フローを実現することを目指す．

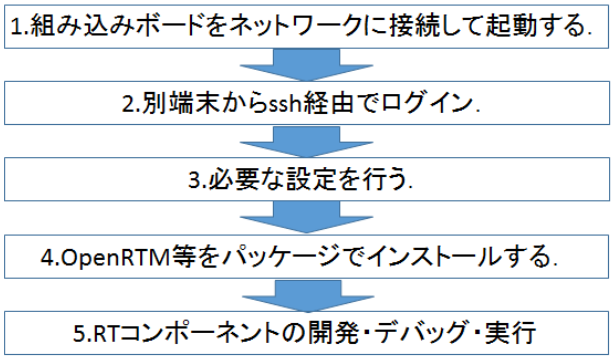

図 3 目標とするワークフロー
手順 2~5 までを自動で行う

4. RT コンポーネント開発・運用環境の構築

4.1 BeagleBone Black 探索ツール

BeagleBone Black にモニタ・キーボードを接続して IP アドレスを確認しなければならない問題を解決するために，ネットワーク上に存在する BeagleBone Black を MAC アドレス頼りに IP アドレスを返すツール:BeagleBone Finder （以下 BBF）を作成した．

BBF を使って BeagleBone Black の MAC アドレスを調べるために表示させたものが図 4 である．MAC アドレスの先頭 24bit，つまり BeagleBone Black では図 4 の示すとおり，C8:A0:30 がベンダーID であるので，ネットワーク上にこのアドレスがあれば BeagleBone Black だと認識できる．

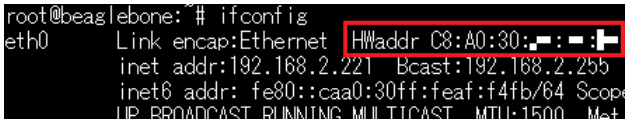

図 4 BeagleBone Black における MAC アドレス

次に BeagleBone Black の先頭 3 つの MAC アドレスを頼りにネットワーク上を探したものが図 4 である．このプログラムは ARP プロトコルを通してネットワーク上の MAC アドレスと IP アドレスを取得し，先頭の MAC アドレスを比較することで目的のデバイスの IP アドレスを特定している．このプログラムを提供することで，ユーザーは BeagleBone Black をモニタに接続することなく，手元のパソコンから ssh 経由でログインすることが出来るため，目的であるヘドレスシステムへの第一歩となる．

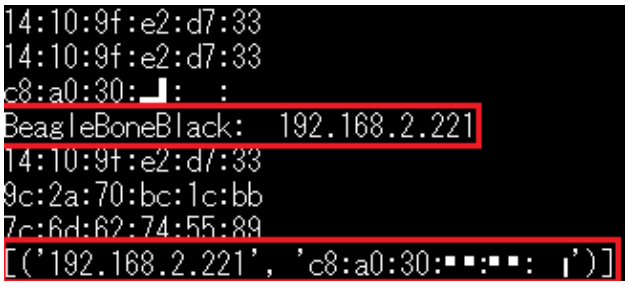

図 5 BeagleBone Black の IP アドレス取得

4.2 opkg 形式によるパッケージの提供

BeagleBone Black 上での，RT ミドルウェアの実行例は過去に報告がなかったため，今回はネットワーク上から必要なソースファイルをダウンロードしコンパイルを行った．このように，使用している組込みボードがあまりメジャーでない場合，ウェブ上にバイナリの実行ファイルが無いため，ソースファイルからのコンパイルを行う必要がある．BeagleBone Black の場合，RT ミドルウェアで必要な omniORB をソースからコンパイルした際約 2 時間，RT ミドルウェアのコンパイルに約 2 時間を要した．しかし，この問題を解消するために，Angstrom Linux ディストリビューションで使われているパッケージ形式，

opkg に準拠した omniORB のパッケージ OpenRTM-aist-1.1 のパッケージを作成した．パッケージをあらかじめ作成し提供することで，RT ミドルウェアの導入にかかる時間を大幅に短縮した．

4.3 BeagleBone Black，プログラムの自動起動．

RTミドルウェアの導入まで非常に容易に行えるようになり，あとはRTコンポーネントのソースコードBeagleBone Black上にコピーしコンパイルするだけでRTコンポーネントを動作させることができる．ここでさらに，実運用時にsshでリモートからログインせずにRTコンポーネントを起動することができれば，機器に組み込んだ際に電源を投入するだけで他のシステムと連携して動作を開始することができる．このために，RTコンポーネントを自動起動する仕組みを構築した．

Angstrom Linux は従来の SysV 系 init スクリプトと systemd を併用したシステムサービス起動機構を持っている．さらに，デフォルト状態では従来のいわゆる rc.local スクリプトが存在しない．今回は，init.d に rc.local をコールする仕組みを独自で作成するとともに，rc.local から RT コンポーネント起動スクリプトを呼び出す仕組みを作成した．具体例を次に示す．

5 BeagleBone Black で Kobuki の操作を行う

移動ロボット制御RTコンポーネントを例にとり，RTコンポーネントの自動起動の仕組みを構築する．対象とする移動ロボットはYujin Robotics社のKobukiとした．Kobukiは実験用の移動ロボットプラットフォームであり，PCなどからシリアルポートとして認識されるUSBから制御することが可能である．ロボットからは5V,12V等の電源が供給されており，BeagleBone Blackなどに電源供給可能である．

Kobukiを制御するRTコンポーネント（KobukiAIST RTC）を作成し，BeagleBone Black上にてコンパイル，インストールを行った．さらに，自動起動スクリプトをrc.localから起動するように設定した．この起動するクリプトは先頭のRTコンポーネントの実行ファイル名を変更することにより，他のコンポーネントの起動にも容易に利用することができる．

このスクリプトでは，BeagleBone Black 上で CORBAネームサービスを起動し，さらにKobukiAIST RTCを起動しこのネームサーバ上に登録させている．外部からはBeagleBone Blackのホスト名・IPアドレスをavahi経由で取得，このネームサーバへ接続することにより起動したKobukiAIST RTCの参照を取得することができる．

以上により，Kobukiの電源を投入後BeagleBone Blackが自動的に起動することにより，ネームサーバ・KobukiAIST RTCが自動起動する．以上によりBeagleBone Black にログインしRTCを起動することなくRTコンポーネント経由でロボットを操作でき，完全なヘッドレスシステムとして運用可能となった．

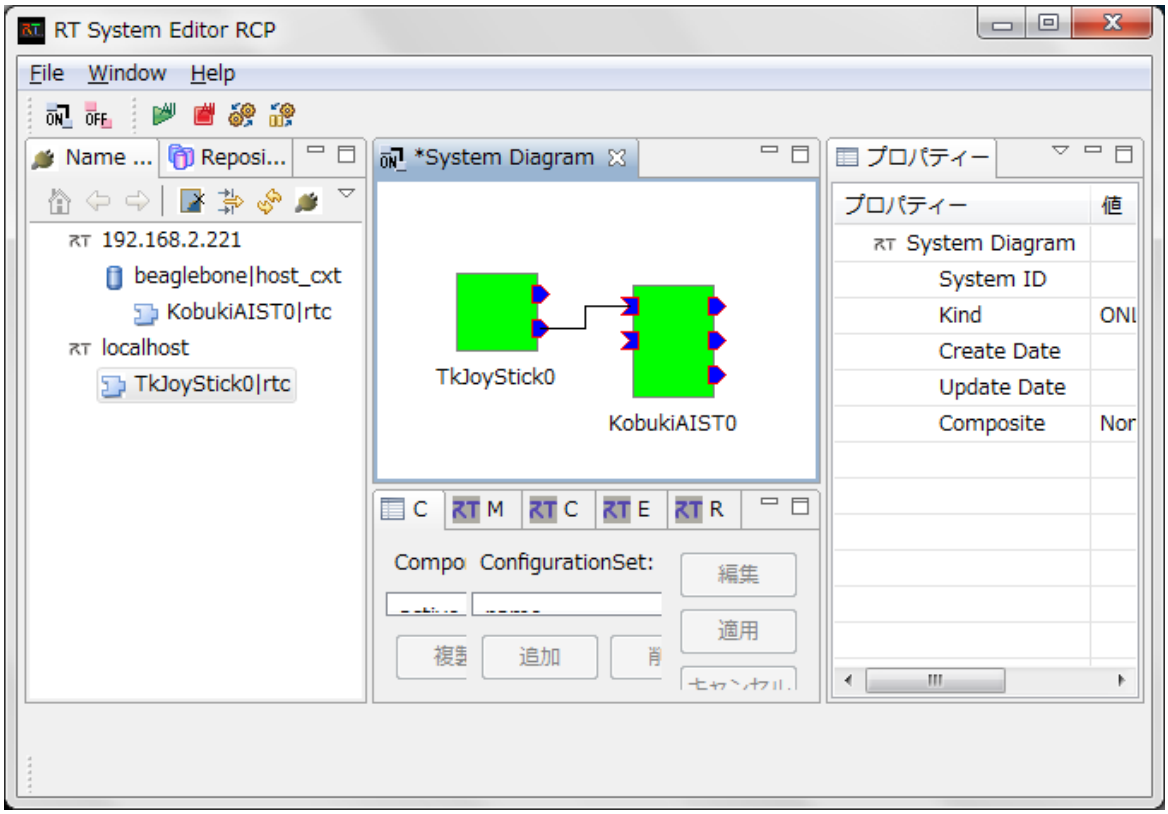

図 6 RT システムエディタ上で接続されたジョイスティック RTC と KobukiAIST RTC

5.1　Kobuki制御実験

BeagleBone Blackで自動起動したKobukiコンポーネントと，Windows上で起動させたGUIJoyStickコンポーネントをWindows上のシステムエディタで接続，起動した（図6）．KobukiとBeagleBone Blackを運用した際の構成は図7になる．実際にKobukiをGUIジョイスティックの操作で走行させた．またKobukiコンポーネントからは，Kobukiに実装されているLEDを制御することが出来るためその動作確認を行った(図8)．以上がBeagleBone Blackを使ったRTミドルウェアの動作検証である．

図7　GUIジョイスティックによるKobukiの操作実験．

図 8　BeagleBoneBlack で動いている RT コンポーネントから，Kobuki 上の LED の制御を行った

6　終わりに

本論文では，最近発売された BeagleBone Black において，RT ミドルウェアの開発における問題点を洗い出した．そこで浮き彫りになった，組込み型の小型 Linux ボードにおけるヘッドレスシステムの開発における煩雑さや，敷居の高さを解決するいくつかの方法を提案した．実際に BeagleBone Finder，OpenRTM-aist などの opkg 形式のパッケージの作成，RT コンポーネントの自動起動を行うスクリプトの作成を行うなどしていくつかの問題を改善した．今回解決できたのは RT ミドルウェアの開発環境の導入までであり，今後開発者が BeagleBone Black の汎用 IO ピン等を通してセンサ等入出力機器を容易に扱える仕組みを整備する予定である．このような仕組みを作ることで，小型組込み Linux の機能を最大限に活用できると考えている．今後 RT ミドルウェアとの更なる連携を図ることでこのような小型の組込み Linux デバイスの開発の敷居を下げ，ロボット技術を応用できるようなソフトウェア的支援環境を構築する．本稿に関するチュートリアルを OpenRTM-aist の Web に今後掲載する予定である．

参　考　文　献

[1] Noriaki Ando, et al.:”RT-Middleware: Distributed Component Middleware for RT (Robot Technology)”:IROS 2005, pp.3933-3938

[2] 鈴木 喬, 他 "ＲＴミドルウェアを適用したロボット機能要素の分散制御", SI2012 p.1B3-6, 2006.12

[3] Noriaki ANDO, et al.: "RTC-Lite: Lightweight RT-Component for Distributed Embedded Systems", SICE Journal of Control. Measurement, and System Integration.Vol.2, No.6, pp.328-333, 2009

[4] 安藤 慶昭, "組込み機器用 RT コンポーネント開発環境 ATDE for OpenRTM-aist", SI2012 pp.543-547, 2012.12

[5] Raspberry Pi + OpenRTM-aist 活用事例 http://www.openrtm.org/openrtm/ja/content/raspberrypi-openrtm-tutorial