mruby Bytecode Loader Using Bluetooth in Multi-VM Environment（Bluetooth を用いたマルチ VM 対応 mruby バイトコードローダ）

学籍番号：09C12174　潮 研究室　山本 拓朗

1　まえがき

近年，組込みシステムは複雑化・大規模化しているため，組込みソフトウェアの生産性が問題になっている．

組込みソフトウェア開発の生産性の向上を目的として，mruby（軽量 Ruby）を適用させたコンポーネントベース開発が可能なフレームワークである mruby on TECS を提案してきた [1].

現状の mruby on TECS では，プラットフォームに mruby バイトコードを組み込んでいるため，mruby プログラムを修正する度にコンパイル・リンクし直す必要がある．さらに，マルチ VM を提供しているが，複数の mruby プログラムを効率よく並行動作させるには開発者がリアルタイム OS の機能を熟知している必要がある．

本研究では，mruby on TECS の拡張として，mruby アプリケーションのバイトコードを Bluetooth で転送することで開発効率を向上させる．さらに，複数の mruby プログラムを協調動作できるフレームワークを提案する．

2　mruby on TECS

mruby on TECS は，組み込み向けの高生産性なスクリプト言語のひとつである mruby (軽量 Ruby) と，組込みシステムに適したコンポーネントシステムである TECS (TOPPERS Embedded Component System) を組み合わせた，組込みソフトウェアのコンポーネントベース開発が可能なフレーム―ワークである．

スクリプト言語は生産性が高い反面，C 言語に比べると実行速度が遅いので，組込みシステムに適用するのは難しい. mruby on TECS では，mruby-TECS Bridge によって mruby プログラムから C 言語レベルの関数を呼ぶことが可能になっている. mruby on TECS は，mruby に比べて，アプリケーションを約 100 倍速く実行できる．

現状の mruby on TECS では，mruby アプリケーションを修正する度にホスト・ターゲットデバイス間で SD カードを抜き差し，再度 OS を起動する必要がある. その上，複数の mruby アプリケーションを並列実行させる場合，開発者がタスクを待ち状態へ遷移させるような OS の機能を呼び出さなければならない．

3　提案フレームワーク

提案フレームワークは，mruby on TECS をベースにして Bluetooth を用いた mruby バイトコードローダと実用的なマルチタスク処理の実装を行った．システムモデルを図 1 に示す．

提案フレームワークでは，はじめの 1 度のみ mruby アプリケーション部分を除いたプラットフォームをコンパイル・リンクする．ホスト側では，mruby アプリケーション (.rb) をバイトコード (.mrb) にコンパイルし，Bluetooth を通してターゲットデバイスにバイトコードを送信する．ターゲットデバイス側では，RiteVM に実装されたローダが送られたファイルを受信し，すでにリンクされている mruby ライブラリと合わせてアプリケーションを実行する．この過程で開発することで，SD カードを抜き差しする手間や OS を起動し直す時間が省けるので，作業効率を上げることができる．

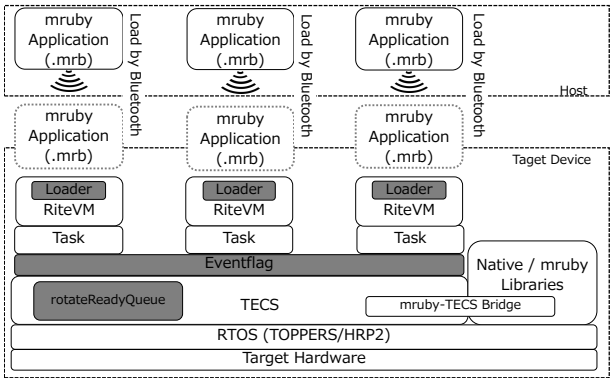

図 1　System Model

さらに，現状の mruby on TECS では使用するのが難しかったマルチタスク処理を，μITRON [2] のサービスコールである *rotateReadyQueue* を周期ハンドラで周期的に呼び出すことで，開発者にとって使いやすい設計にした. *rotateReadyQueue* は，同じ優先度のタスクの実行順序を切り替える機能である. 複数タスク間の同期にはイベントフラグ処理を適用し，すべてのタスクが同時に起動するように実装した. イベントフラグのセットパターンや待ちパターンは，各コンポーネントの属性として定義する. これによって，VM の数に関わらず，同じ.c ファイルを利用することが可能になる．

提案フレームワークでは，RiteVM や周期ハンドラ，イベントフラグはすべて TECS のコンポーネントとして実装されているので，開発者の必要に応じて機能を付けたり外したりすることも容易に実現できる．

4　結論

mruby on TECS の機能として，Bluetooth を用いた mruby バイトコードローダと実用的なマルチタスク処理を提案した. 実験評価では，ローダによる作業効率の向上やオーバーヘッドの少ないマルチタスク設計の有用性を示した．

参考文献

[1] Azumi, T. and Nagahara, Y. and Oyama, H. and Nishio, N., “mruby on TECS: Component-Based Framework for Running Script Program,” Proceedings of the 18th IEEE International Symposium on Real-Time Distributed Computing (ISORC), pp.252-259, 2015.

[2] Hiroaki Takada and Ken Sakamura, “μITRON for Small-Scale Embedded Systems”, IEEE Micro, vol.15, no.6, pp.46-54, 1995.