◆ Rubyの応用 ◆

6 Rubyを使った組込みソフト開発

― mrubyによる組込みシステム開発―

田中和明（九州工業大学）

組込みシステム開発の課題

組込みシステムとは，家電製品や自動車，産業用機器などの制御に利用されるコンピュータシステムの総称であり，ソフトウェアによりハードウェアを操作することが，組込みシステムの目的である．近年，製品に対して，多くの複雑な機能が要求されてきている．従来の機械式制御や電気式制御では，複雑な機能の実装が困難であったり，コスト面から現実的でないこともある．現在では，さまざまな機能をソフトウェアで実現するのが一般的で，これを組込みソフトウェアと呼んでいる．

たとえば，十数年前の自動車に搭載されていたコンピュータ（車載コンピュータ）のソフトウェアは一部の機能の実装にのみ利用され，そのコードは数十万行程度であった．2015年現在では，車載コンピュータのコード規模は数千万行から1億行とも言われている．

組込みソフトウェアが大規模になることは，ソフトウェア開発のコストが増大することを意味する．しかし現実問題として，先の車載コンピュータの例にあるようにコード規模が100倍になったとしても，100倍のコストがかけられるわけではない．ここには，ソフトウェアの生産性を高める工学的手法が採用されてきた．たとえば，アセンブラで記述されていたコードをC言語で記述する，スクリプトを使ってコードを自動生成するなどの手法で，近年はモデルベース開発も採用されてきている．

一方で，ソフトウェアのメンテナンス面での課題もある．Webシステムや PC上で動作するシステムであれば，ネットワークを使ったソフトウェアの自動更新などの手法によって，ソフトウェアの不具合を改修できる．組込みシステムは，ネットワークに接続されていない，または，制限されたネットワーク（たとえば工場内LANなどの閉じたネットワーク）で運用されることも多く，ソフトウェアの改修には相当のコストが発生する．しかも，組込みシステムは運用期間が長く，長期間にわたって機能を維持しなくてはならない．

mrubyの開発

◆ mruby（軽量 Ruby）

組込みシステムの開発を効率良く行え，ソフトウェアの改修が容易で，しかも，長期間にわたるソフトウェアのメンテナンスが可能であるような開発手法を提案できないかと考えた．とはいえ，このような都合の良い開発手法が簡単に提案できることはなく，開発手法に代えて，開発言語（プログラミング言語）に着目し，要件を以下のように整理した．

- 組込みシステムに利用できる
- ソフトウェアを効率良く開発できる
- ソフトウェアの改修が容易である
- ソフトウェアの保守が容易である

経済産業省の2010年度地域イノベーション創出研究開発事業「軽量 Rubyを用いた組込みプラットフォームの研究・開発」において，福岡CSK，九州工業大学，東芝情報システム，福岡県が共同で，この要件を満足するようなプログラム言語を開発した．

この事業では，プログラム言語 Rubyの軽量化によって組込みソフトの開発環境を開発した．すでに，Webアプリケーション分野では Rubyと Ruby on

Rails が広く利用されてきており，Ruby を使うことでソフトウェアを効率良く開発できることが知られていた．この Ruby のメリットを組込みソフトの開発に適用したものが軽量 Ruby である．

当該事業の終了後，軽量 Ruby はオープンソースソフトウェア mruby として公開された．mruby という名称は，embedded ruby（もしくは embeddable ruby）から生まれた．実は，HTML に埋め込まれた Ruby が eRuby（拡張子 erb）として Web アプリケーションで利用されており，eRuby ではなく mruby となっている．なお，minimal ruby や machine ruby という意味合いも含んでいる．

◆ mruby の実装

mruby の開発で最も重視したのは，「組込みシステムに利用できる」ことである．組込みシステムのハードウェアは，消費電力，熱，コストの要件を満たすため，使用できる資源に制限がある．特に，メモリに対する制限は厳しく，少ないメモリで動作することを目標の 1 つとした．また，組込みシステムではさまざまな実行環境が想定され，OS やプロセッサへの依存を少なくすることも望まれる．

Ruby はインタプリタ方式のプログラム言語であり，実行時にプログラムソースコードの解析を行うが，このことは実行環境に多くの資源を必要とする．そのため，Ruby をそのまま組込みソフトウェアの開発言語として採用することはできない．

望まれているのは「実行時に必要な資源」を少なくすることである．そこで，mruby ではコンパイラ方式を採用した．コンパイラ方式には，C 言語のようにネイティブコードを生成するものと，Java のように中間コードを生成して，その中間コードを仮想計算機（VM）で実行するものがある．mruby は後者の方式を採用した．mruby の実行を**図 -1** に示す．

図 -1 の手順の中で，mruby コンパイラによるバイトコードの生成は開発環境で行う．生成されたバイトコードを実行環境に転送する．一般にバイトコードの転送は JTAG ケーブルやメモリカードを介して行われる．バイトコードは実行環境にある VM により実行される．実行環境で使用できる資源には制限があるため，この VM はコンパクトに実装されている．VM は C 言語で記述されており，ソースコードの行数は 25,000 行程度で，Ruby のソースコードの 230,000 行程度と比べて十分に小さい．また，実行時に VM が必要とするメモリは 400KB 程度である．

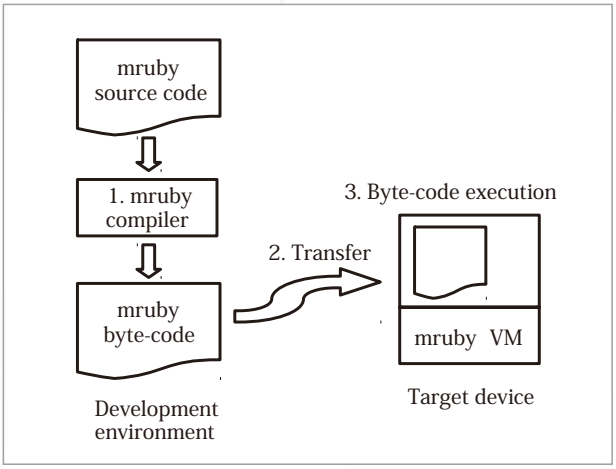

図 -1　mruby プログラムの実行

VM によるバイトコードの実行では，新しい実行環境への対応を簡単化できる．もし，コンパイラがネイティブコードを生成した場合，そのコードの実行はその実行環境に依存してしまう．実行環境（主に小型マイコン）は変化が激しく，新機能を搭載したマイコンが次々にリリースされている．そのため，組込みソフトウェアは常に新しい実行環境へ対応していく必要がある．mruby が VM による実行を採用したことで，新しい実行環境に対応した VM を作成するだけで，アプリケーション（mruby で記述され，コンパイラで生成されたバイトコード）はまったく変更することなく，新しい実行環境で動作する．

◆ mruby のリアルタイム性

組込みシステムでは，リアルタイム性が要求される．リアルタイム性とは，ある入力を受けてから出力する処理に，時間の制約が設けられる性質のことである．リアルタイム性は，必ずしも実行時間（実行速度）を問うものではなく，実行にかかる時間の

制約を満たすことが可能かどうかを問う．一般には，ある処理の実行時間が正確に見積もれるかどうかが問題となる．

Ruby および mruby は，メモリ管理にガーベージコレクション（GC）を採用している．GC は実行中に参照されなくなったメモリを回収する機構で，Java や Ruby のほか，多くのプログラミング言語で採用されている．一般的な GC は Mark&Sweep 方式であり，メモリを再帰的に探索してメモリに印をつけ（Mark），すべてのメモリの探索が完了したら，印が付いていないメモリ（＝参照されていないメモリ）を回収する（Sweep）．Mark&Sweep 方式の GC のパフォーマンスは高いが，GC 実行中は VM の実行が完全に停止する．

mruby の VM は，インクリメンタル GC を採用している．インクリメンタル GC は，GC 実行時の停止時間を細分化し，VM による mruby プログラムの実行と GC を並行して進める．GC を細分化することによるオーバヘッドは生じるが，細分化した GC の処理量は見積もれるため，ある程度のリアルタイム性を確保できる．その反面，オーバヘッドが大きいことと，細分化した GC の処理量に起因する実行時間のゆらぎが生じるため，厳密なタイミングを保証することはできない．

一方で，頻繁に GC が行われることから，実行時に必要なメモリは少なくなる．**図 -2** に Mark&Sweep 方式とインクリメンタル方式で使用するメモリ量のイメージを示す．Mark&Sweep 方式では，GC を実行するまでに多くのメモリが消費され，GC により不要なメモリが回収される．インクリメンタル方式では頻繁に GC が行われるため，不要なメモリがメモリに存在しつづける時間が短い．結果として，mruby の VM が使用するメモリは少なくなる．

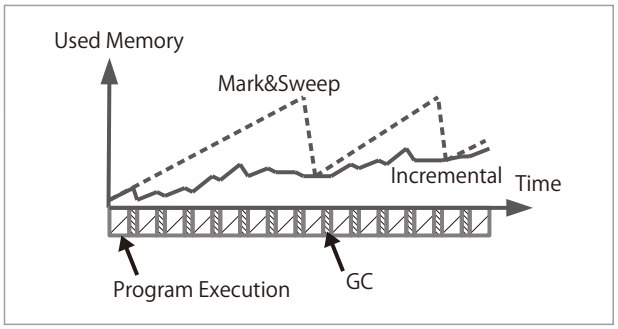

図 -2　GC による実行と使用メモリのイメージ

◆ VM の機能拡張

mruby プログラムは，開発環境でのコンパイルによりバイトコードに変換され，そのバイトコードが実行環境の VM により逐次実行される．したがって，同一のバイトコードを異なる実行環境の VM で動作させることができる．VM 本体は実行環境に依存するプログラムであり，VM が実行するバイトコードは実行環境に依存しない．

mruby バイトコードから環境に依存するメソッドを呼び出す際には，そのメソッドに対応する C 言語で実装された関数が呼び出される．VM には，C 言語の関数をラッピングして mruby のメソッドとして呼び出せるようにするラッパー（wapper）が記述されている．このラッパーこそが実行環境に依存する個所であり，ラッパーを適宜用意することでさまざまな実行環境に対応できる．

ラッパーが記述されるのは mruby の VM 本体であり，容易に機能拡張できるようにするため，mrbgems というパッケージ管理システムを持っている（Ruby の RubyGems に相当する）．たとえば，時刻に関するライブラリは OS や RTC（real-time clock）に依存するため，`mruby-time` パッケージとして提供されている．

このパッケージを使用したい場合，VM を再構成（ビルド）する．ビルドに必要な設定は，**図 -3**（1 行目）のように，ビルド設定ファイル `build_config.rb` に行を追加するだけである．同様に，GitHub で公開されているパッケージであれば，図 -3（2 行目）のように記述できる．このような VM を構成するパッケージを柔軟に変更できる仕組みにより，実行環境に依存する部分を明確に切り分けることができ，さまざまな環境に対応した VM をビルドする際にとても役立つ．

```
conf.gem 'mrbgems/mruby-time'
conf.gem :github =>  'username/repositry'
```

図 -3　build_config.rb

```
led_port = GPIO::PORT01
loop do
  digitalWrite led_port, GPIO::HIGH
  delay 500
  digitalWrite led_port, GPIO::LOW
  delay 500
end
```

図 -4　mrbgem：GPIO

◆ mruby プログラムの実行

VM をビルドする際に，環境に依存した部分を切り分けることができる．そこで，開発環境用のパッケージを作成しておけば，開発環境で作成した mruby プログラムを，そのまま開発環境で実行させることができる．

Arduino に似せた汎用 I/O（GPIO）クラスを試作し，それをパッケージとして利用できるようにした．このパッケージを使うことで，たとえば**図 -4** のような mruby プログラムを作成できる．プログラム中の `digitalWrite` メソッドはディジタル出力，`delay` メソッドはタイマーによる一定時間のウェイトである．これらのメソッドは，環境に依存するライブラリで実装されており，このパッケージを新規に作成するだけで，VM を異なる環境へ移植できる．

汎用 PC 用の GPIO クラスでは，`digitalWrite` メソッドはログメッセージを画面に出力し，`delay` メソッドはログメッセージを出力した上でウェイトする．一方，マイコンボード（今回使用したのは Raspberry Pi）用の GPIO クラスでは，`digitalWrite` メソッドは Raspberry Pi 上の I/O ピンにディジタル値を出力し，`delay` メソッドはウェイトする．

図 -4 のプログラムから mruby コンパイラにより作成されたバイトコードは，開発環境の汎用 PC 上で動かすことができ，出力されるログメッセージと動作タイミングを見ることでプログラムの挙動を確認できる．次に，このバイトコードを Raspberry Pi の SD カードに書き込み，あらかじめ Raspberry Pi 用にビルドされている VM で実行させると，期待通りの出力を行うことができた．

今後の展望

mruby を使うことで，組込みソフトウェアの開発効率が高まると期待している．Ruby/mruby のプログラムは可読性が高く，アジャイル開発でもよく利用される．このような開発スタイルを組込みシステム開発に導入できると考える．すなわち，mruby により組込みソフトウェアを短時間でリリースし，そのテストと評価を再び実装に反映させることで，機能を少しずつ実装していくことができる．これにより，手戻りの発生を回避できる．

多くの場合，実行環境に依存するライブラリがすでに整備されていて，新規開発すべき個所は実行環境に依存しないことも多い．たとえば，ユーザインタフェースの開発や，制御シーケンスの実装などでは，必要な GUI 部品 ☆1 や制御インタフェースは開発済みで，その組合せと実行の順序を開発するというような場面を想定している．

mruby を組込みソフトウェアの開発の一部に利用することで，組込みシステム開発の効率化と競争力向上に寄与できることを期待している．最後に，mruby の情報については，NPO 法人軽量 Ruby フォーラム ☆2 で提供しているので活用されたい．

（2015 年 8 月 3 日受付）

☆1 Graphical User Interface，操作画面中のボタンなど，ユーザインタフェースを構成するソフトウェア部品．
☆2 http://forum.mruby.org/

田中和明（正会員） kazuaki@mse.kyutech.ac.jp

博士（情報工学）．ロボット制御やセンサデータの処理に関する研究から，現在は組込みソフトウェア開発に Ruby を利用する試みに取り組んでいる．Ruby Association 理事，軽量 Ruby フォーラム理事．