mrubyによる組込み開発のメリット

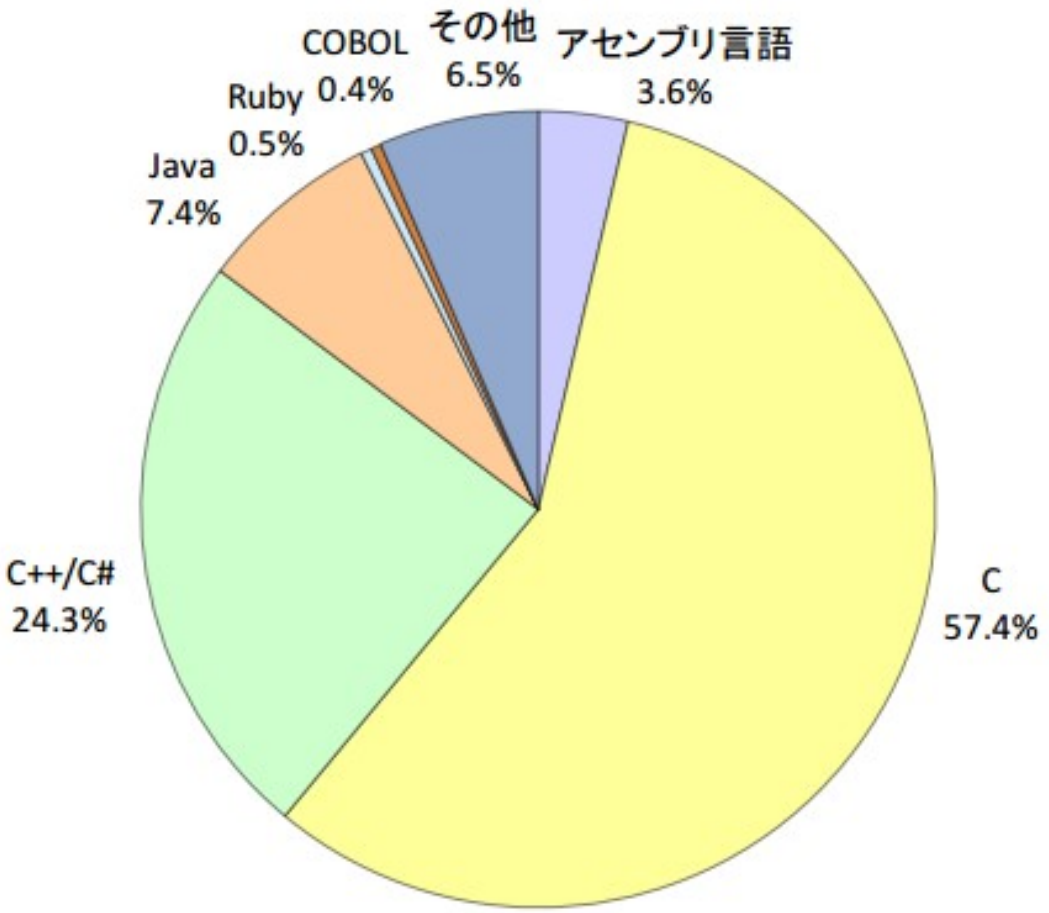
九州工業大学
田中 和明
軽量Rubyフォーラム

自己紹介

- 九州工業大学 情報工学部
  - 教育, 研究
- 軽量Rubyフォーラム
- Ruby Association

組み込み開発の現状

- 開発の約8割は C/C++ を使っている
  - これまでのコード
  - ハードウェアのアクセス



2011年版 組込みソフトウェア産業実態調査
（経済産業省）

Rubyを組み込み開発へ・・・

- 平成22年度 経済産業省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」

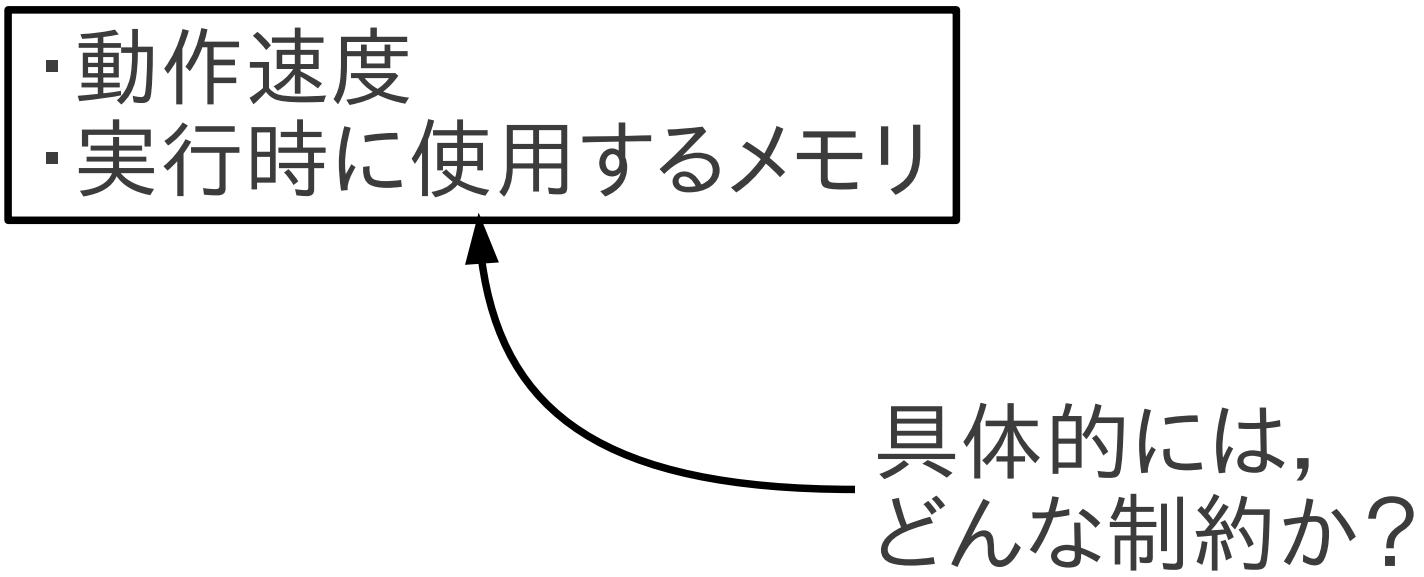
- Rubyを軽量化（スリム化）し，
組込み開発で利用できるようにする

C / Ruby

	C/C++	Ruby	軽量Ruby
実行時	実行速度が速い 少ないメモリで 動作する	実行速度が遅い 多くのメモリを 必要とする	少ないメモリで 動作する
開発時	開発コストが 大きい	開発効率が高い	開発効率が高い
安全性	検証が難しい	読みやすいコード 検証しやすい	読みやすいコード 検証しやすい

mrubyの開発

- 従来のRubyの特徴を活かしつつ，
  組込み開発で要求される制約を満足する



組込みソフトの要求

- 動作速度
  - 多くの場合，「高速」に動作することより，
    リアルタイム性（動作時間の保証）が望まれる
- 使用するメモリ
  - 実行時に使えるメモリが少ない
    Rubyは一般に多くのメモリが必要

動作速度

- Rubyは開発しやすさを重視している
  ↓
- プログラムの実行に手間がかかる
  （Rubyの柔軟性を提供するため）
  ↓
- Rubyプログラムの実行速度が遅くなってしまう

リアルタイム性

- Rubyでは，メモリは自動で管理されている
（Cでは手動なので，メモリ管理も手間がかかる）
↓
- 不要になったメモリを自動で回収する仕組み
↓
- 実行時に自動回収（GC）が行われるため，
Rubyプログラムの実行が一時停止する

使用するメモリ

- Rubyは開発しやすさを重視している
  ↓
- プログラムを実行時に解読して, 処理する
  (プログラムを書いて, そのまま実行できる)
  ↓
- Rubyの解読プログラムが実行時に必要
  (これがメモリを大量に消費する)

mruby（軽量Ruby）

- リアルタイム性をある程度実現する
- 実行時に必要なメモリを減らす

さらに，

- 組込みソフトの要求を満たす
  - さまざまなハードウェアをサポートする
  - OSやファイルシステムも異なる（無い場合も）

Rubyプログラム
実行の仕組み

- Rubyプログラムの実行


```
a = b + c
if a > 5 then
  puts “OK”
end
```

Rubyプログラム

```
(= a (+ b c) )
(if (> a 5)
    (puts “OK” )
    nil)
```

実行環境に
依存しない

```
OP_SEND R2  :b
OP_LOADSELF R3
OP_SEND R3  :c
OP_ADD  R2  :+  1
OP_MOVE R1  R2
....
```

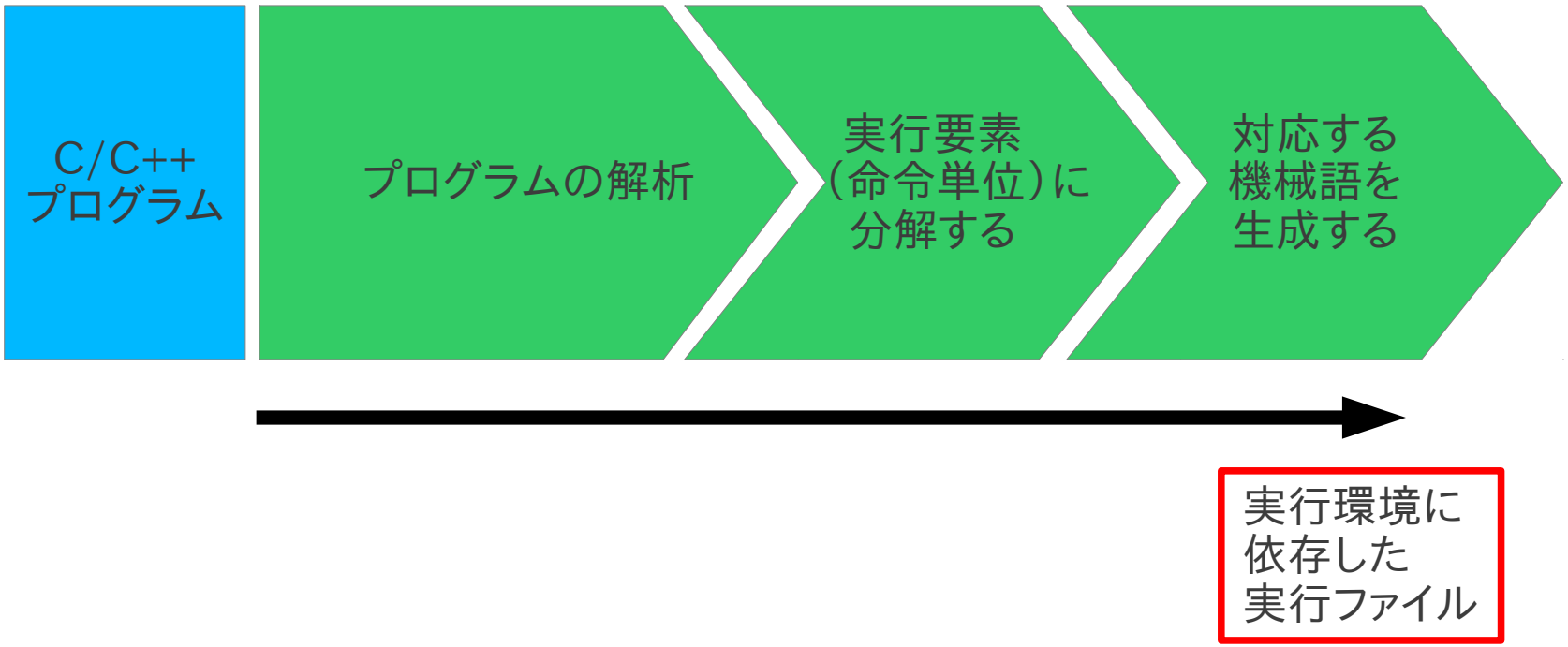
実行環境に
依存しない

```
movl  -8(%ebp), %eax
movl  -12(%ebp), %edx
addl  %edx, %eax
movl  %eax, -4(%ebp)
...
```

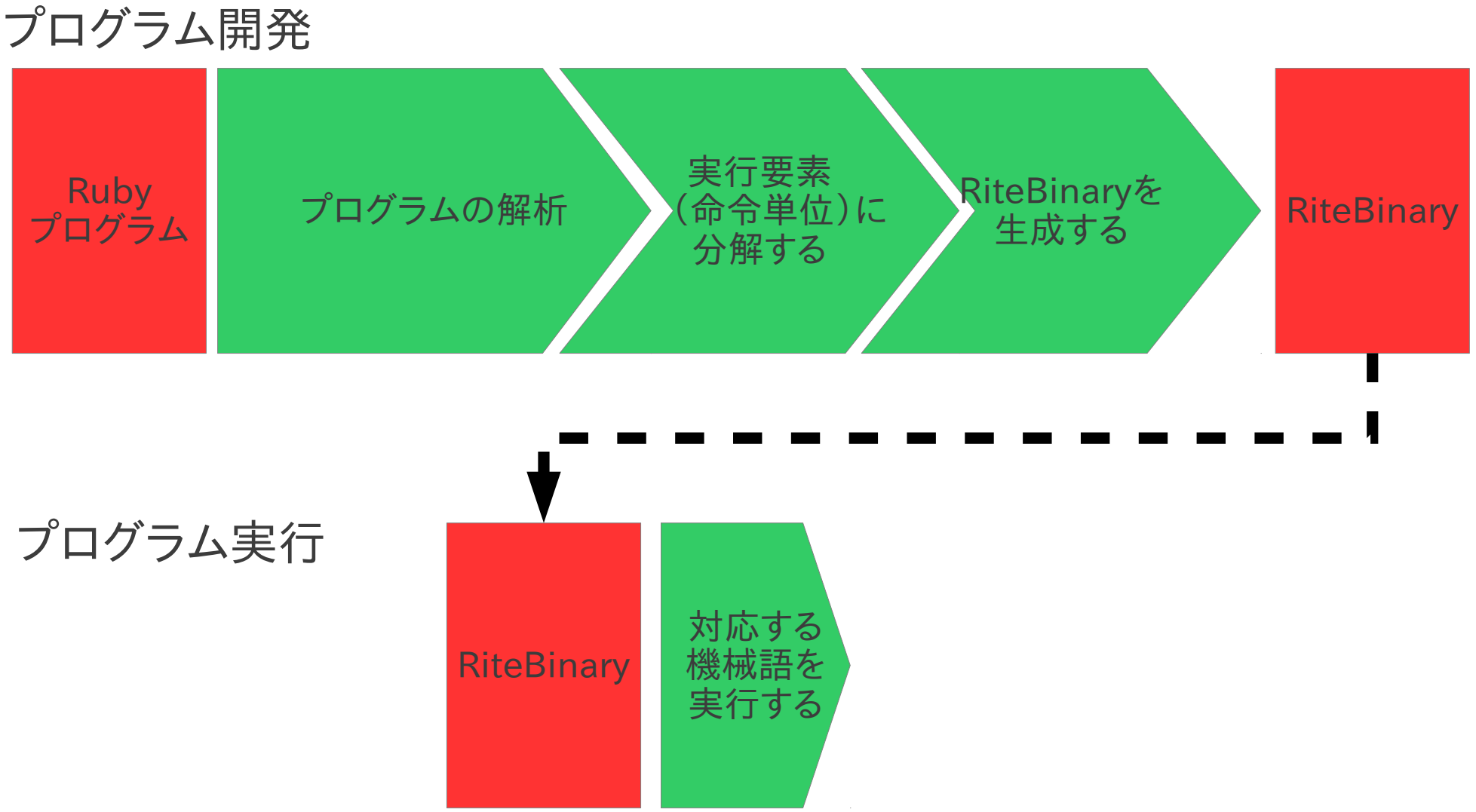
実行環境に
依存した機械語

（参考）C言語では...

- コンパイルにより機械語の実行プログラムが生成される



mrubyの場合


プログラム開発
Ruby
プログラム
プログラムの解析
実行要素
(命令単位)に
分解する
RiteBinaryを
生成する
RiteBinary
プログラム実行
RiteBinary
対応する
機械語を
実行する

mrubyを使った開発スタイル

- VMがRiteBinaryを実行する

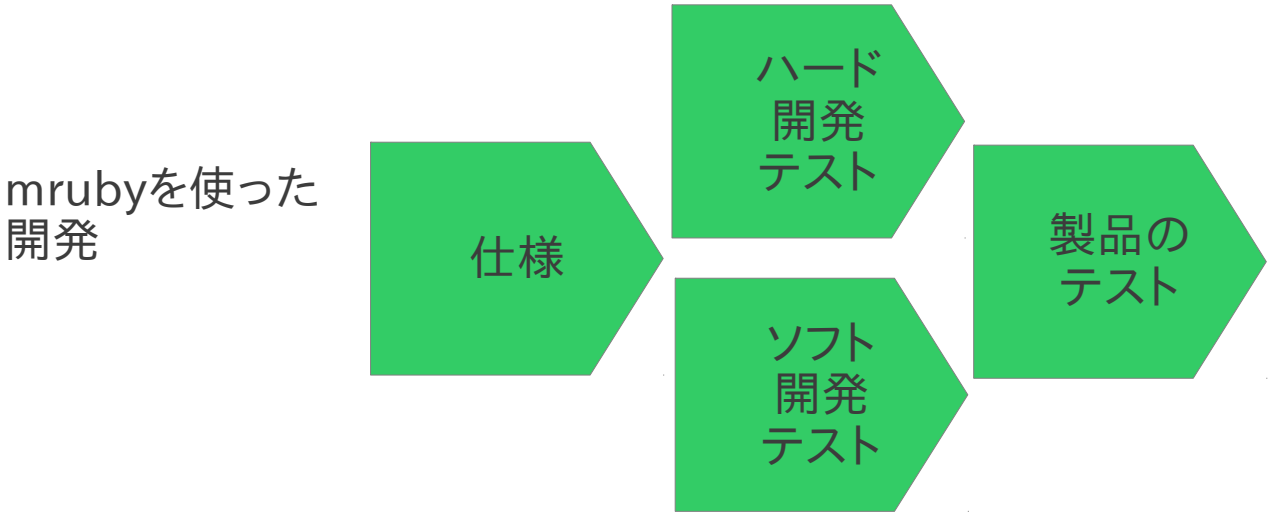
  ||

- VMがあれば, どの環境であっても mrubyプログラムを動かせる!
- 開発環境でソフトウェア開発, 同時にテスト

  ↓

- ターゲットデバイスで動作させる

開発効率



製品がなくても
ソフトウェアを
テストできる

ソフトウェアテストで発見した
仕様の不具合を
ハードウェア開発に
フィードバックできる

開発の具体例(プログラミング)

- Rubyを使うと,動作を記述しやすい

```
def setup()
   gr_pinMode($PIN_LED0, $OUTPUT)
end

def loop()
   gr_digitalWrite($PIN_LED0, 1)
   gr_delay(100)
   gr_digitalWrite($PIN_LED0, 0)
   gr_delay(100)
end
```

LEDの出力モードに設定する

LEDを点灯させる

100msウェイト

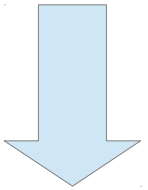
LEDを消灯させる

ILC社 EAPL-Trainer mrubyのサンプルプログラム

開発の具体例(コンパイル)

```
def setup()
   gr_pinMode($PIN_LED0, $OUTPUT)
end

def loop()
   gr_digitalWrite($PIN_LED0, 1)
   gr_delay(100)
   gr_digitalWrite($PIN_LED0, 0)
   gr_delay(100)
end
```



mrubyコンパイラ (mrbc)

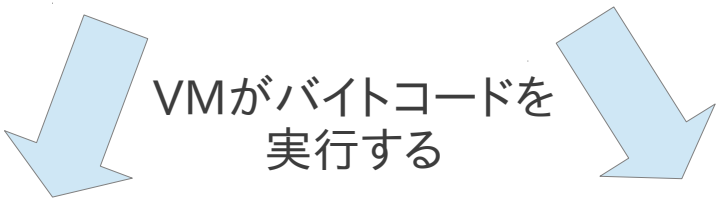
```
000 OP_LOADI      R1    10
001 OP_SETGLOBAL  :$PIN_LED0  R1
002 OP_LOADI      R1    1
003 OP_SETGLOBAL  :$OUTPUT    R1
004 OP_TCLASS     R1
005 OP_LAMBDA     R2    I(+1) 1
006 OP_METHOD     R1    :setup
007 OP_TCLASS     R1
008 OP_LAMBDA     R2    I(+2) 1
009 OP_METHOD     R1    :loop
010 OP_LOADNIL    R1
011 OP_STOP
...
```

生成されたバイトコードは
デバイス非依存
(実際にはバイナリファイル)

開発の具体例(実行)

```
000 OP_LOADI      R1    10
001 OP_SETGLOBAL  :$PIN_LED0  R1
002 OP_LOADI      R1    1
003 OP_SETGLOBAL  :$OUTPUT    R1
004 OP_TCLASS     R1
005 OP_LAMBDA     R2    I(+1) 1
006 OP_METHOD     R1    :setup
007 OP_TCLASS     R1
008 OP_LAMBDA     R2    I(+2) 1
009 OP_METHOD     R1    :loop
010 OP_LOADNIL    R1
011 OP_STOP
...
```

RiteBinary



PC用のVM

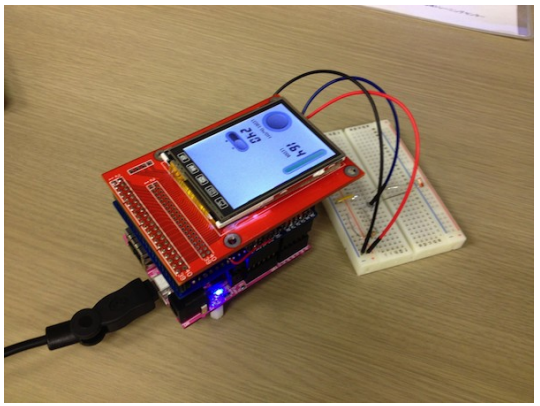
マイコン用のVM

クロス開発が不要になる

- mrubyコンパイラ
  - Rubyプログラム→RiteBinary（バイトコード）
  - バイトコードはどこで作成しても同じ
- mruby VM
  - 実行環境に合わせたVMを準備しておく
  - VMの作成にはクロス開発が必要

教育用教材

組込み業界GUIライブラリ
トップシェアのILC社による
軽量Ruby（mruby）教育教材



■全体構成

マイコンボードとタッチパネル付き液晶を同梱しています。mrubyプログラムから使えるグラフィック表示機能を内蔵しているので液晶画面とタッチパネルを活用したmrubyプログラムを作ることができます。

MobiRuby

軽量Ruby（mruby）による
モバイルアプリケーション開発

```ruby
# UIAlertView demo
class Cocoa::MyAlertView < Cocoa::UIAlertView
    define C::Void, :didPresentAlertView, Cocoa::Object do
        p "MyAlertView::didPresentAlertView"
    end

    define C::Void, :alertView, Cocoa::Object, :clickedButtonAtIndex, C::Int do
 |me, index|
        if index.to_i == 1
            app = Cocoa::UIApplication._sharedApplication
            url = Cocoa::NSURL._URLWithString("http://mobiruby.org")
            app._openURL url
        end
    end
end

alert = Cocoa::MyAlertView._alloc._initWithTitle "Hello",
  :message, "I am MobiRuby",
  :delegate, nil,
  :cancelButtonTitle, "I know!",
  :otherButtonTitles, "What's?", nil
alert._setDelegate alert
alert._show
```

enzi（仮称）

- 福岡CSKが開発したmubyボード
  - http://enzi.cc/

DEMO

- サンプルプログラムの動作
  - 簡単なプログラム例
- Cプログラムとの連携
  - mrubygems

現状

- オープンソース mruby として公開中

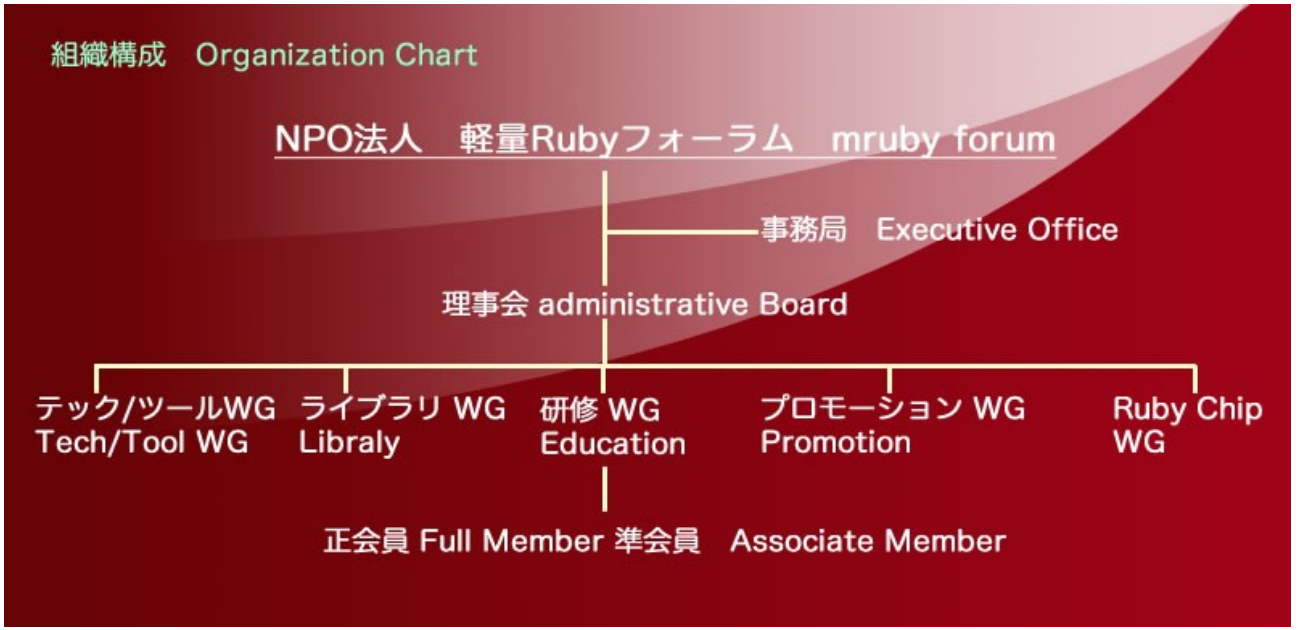
  https://github.com/mruby

  - MITライセンス
    ソースコード非公開で自由に利用できる
    著作権者名とライセンスの記載が必要

git https://github.com/mruby/mruby.git

今後の軽量Ruby（mruby）

- 特定非営利活動法人「軽量 Ruby フォーラム」
  - 成果物（軽量Ruby）の管理
  - ライブラリ等の開発支援
  - 著作権管理
  - 普及活動



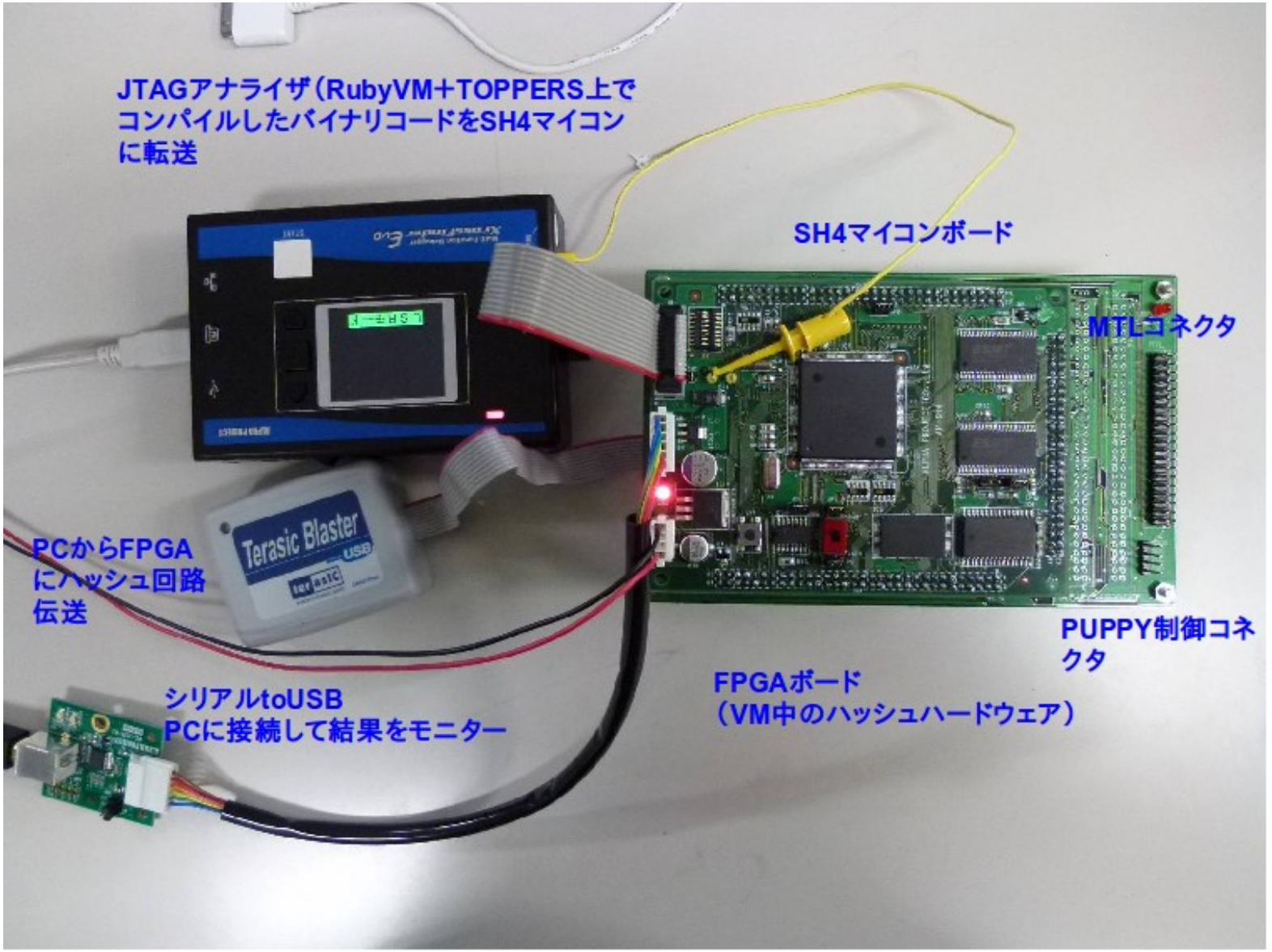
http://forum.mruby.org/

研究開発事例

- mrubyチップ

mrubyのチップ化

- LSIチップの開発を目指す
  - LSIチップ開発に必要な技術を集積する
- FPGAを使って, mruby機能の一部をハード化
  - 使用するLE（Logic Element）は
    Altera Cyclone チップの1%以下
- Rubyのプログラムの変更は不要
  - VMで自動的にハードウェアを利用する


JTAGアナライザ（RubyVM+TOPPERS上で
コンパイルしたバイナリコードをSH4マイコン
に転送
SH4マイコンボード
MTLコネクタ
Terasic Blaster
USB
PCからFPGA
にハッシュ回路
伝送
PUPPY制御コネ
クタ
FPGAボード
（VM中のハッシュハードウェア）
シリアルtoUSB
PCに接続して結果をモニター

mrubyのボトルネックはどこ？

```
class ClassA
  def func
    puts ”Here!”
  end
end

class ClassB < ClassA
end

x = ClassB.new
x.func
```

1. 変数xの中から関数funcを探す
→ 失敗する
2. クラスClassBの中から関数funcを探す
→ 失敗する
3. クラスClassAの中から関数funcを探す
→ 発見する，関数を呼び出す

Rubyの関数呼び出しは，関数を「探す」必要がある
・・ 高速に探索する必要がある
・・・ ハッシュ表を使う
・・・・ ハッシュ関数が頻繁に呼ばれる

ハード化による性能向上

- フィボナッチ数列の計算
  - 関数の再帰呼び出し
  - 約 8%の性能向上
- 文字列処理
  - 文字列処理の関数の多くはRubyで書かれている
  - 約20%の性能向上

$$\text{性能向上(\%)} = \frac{\text{ハードウェアによる短縮時間}}{\text{ソフトウェアでの実行時間}}$$