ETロボコン向けTOPPERS活用セミナー

mruby on EV3RT+TECS を使ったアプリ開発環境

小南靖雄（TOPPERS個人会員）

2018年6月11日

TOPPERS

目次

- EV3に対してmrubyで操作できること
- mruby on EV3RT+TECSの仕組み
- mruby on EV3RT+TECSのリアルタイム性能
- 環境構築
- mrubyのビルド
- EV3Way向けとETロボコン向け設定
- 二つの開発方法
- 開発方法(SD)
  - コンパイル手順
  - アプリケーションの起動方法
- ev3way_sample
- エラー対処方法
- 付録

TOPPERS

EV3に対してmrubyで操作できること(クラス一覧)

- Balancer
- Battery
- Button
- LCD
- LED
- Speaker, ・RTOS , ・Parameter

クラスメソッドとして
呼び出す

- Motor
- ColorSensor
- GyroSensor
- UltrasonicSensor
- TouchSensor

インスタンス化（new）が必要
及びポート番号の指定が必要

TOPPERS

EV3に対してmrubyで操作できること(クラスメソッド)

Balancer	ETロボコン向けバランス制御（4ミリ秒周期）
Battery	バッテリの電流、電圧取得
Button	ボタンの押下状態取得
LCD	LCD表示
LED	LEDライトのカラー設定
Speaker	音量を調整、トーン出力
RTOS	自タスクを指定された時間遅延、時刻の参照、自タスクを起床待ち状態にする
Parameter	各種センサ、各モータのポート番号記録、記録したポート番号取得

TOPPERS

EV3に対してmrubyで操作できること(インスタンスメソッド)

Motor	モータのスピード設定、パワー取得、モーター停止、モータを指定角度回転、モータの角位置取得、モータの角位置をゼロにリセット
ColorSensor	カラーセンサで反射光の強さを測定、環境光の強さを測定、RGB値を測定、カラーを識別、指定色であるか識別
GyroSensor	ジャイロセンサで角速度、角位置を測定、角位置をゼロにリセット、キャリブレーション複数回測定した値の平均値取得
UltrasonicSensor	超音波センサで距離測定、超音波信号検出
TouchSensor	タッチセンサの状態検出

TOPPERS

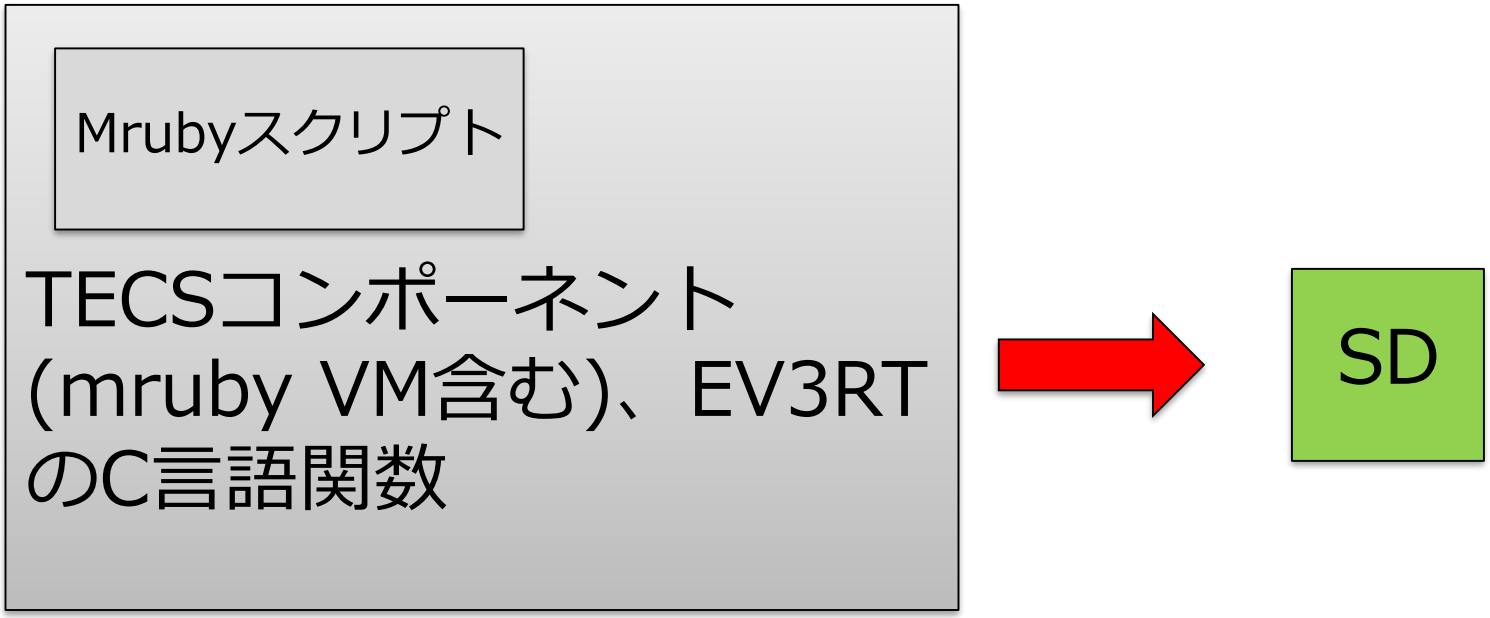
mruby on EV3RT+TECSの仕組み

- EV3RTの**スタンドアローン形式**として動作
- EV3RTで提供されているC言語ライブラリ、mrubyのVMをTECSコンポーネント化(TECSジェネレータがコンポーネント部分のC言語コードを自動生成)
- TECSコンポーネントをmrubyのオブジェクトとするC言語コードを自動生成(TECSジェネレータのMrubyBridge プラグイン)
  - mrbgemが内部で行うことと同等な処理をする
  - オブジェクトのメソッドの実体はC言語関数

TOPPERS

mruby on EV3RT+TECSの仕組み

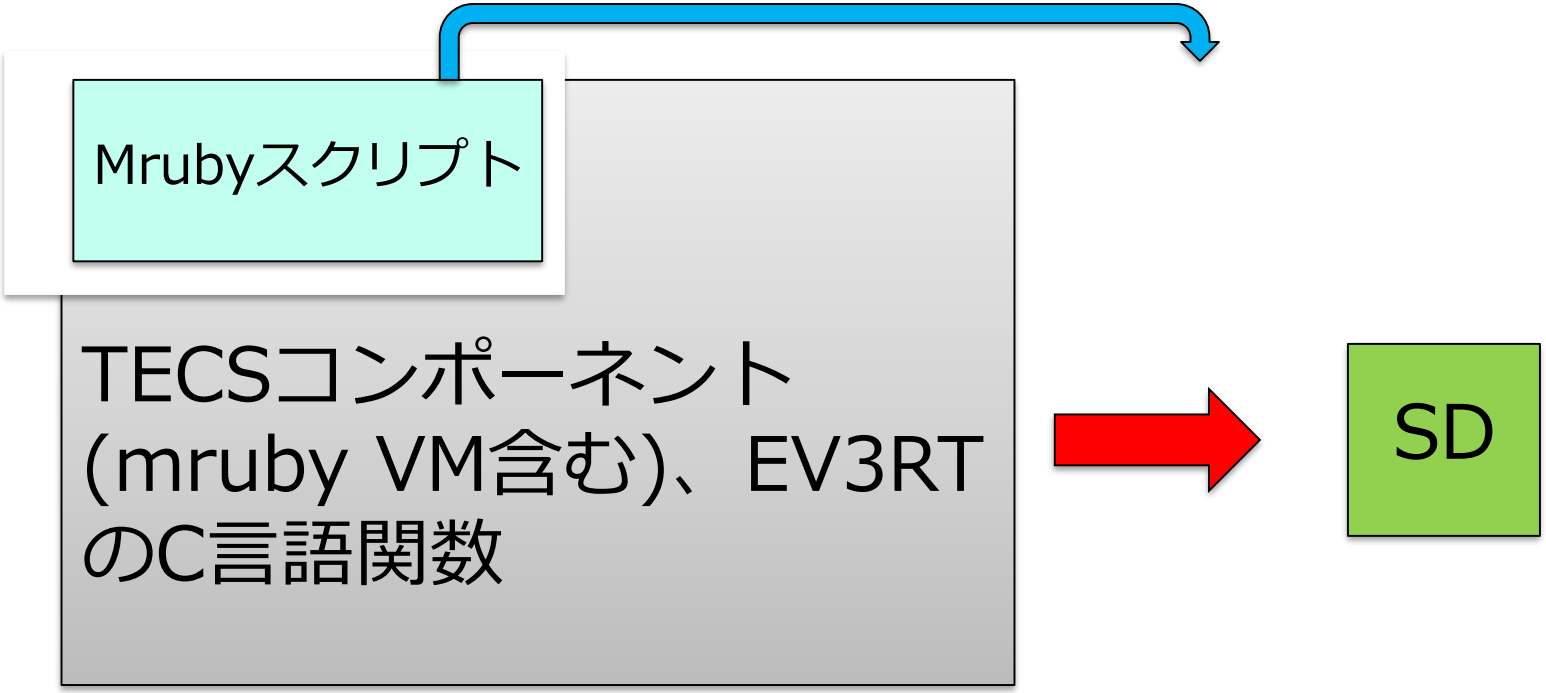
- SD
  - mrbcでスクリプトをコンパイルしてC言語の配列として出力。スクリプトを含むもの全体をCコンパイラでコンパイル。**一つのイメージとしてSDカードに書き込む。**



TOPPERS

mruby on EV3RT+TECSの仕組み

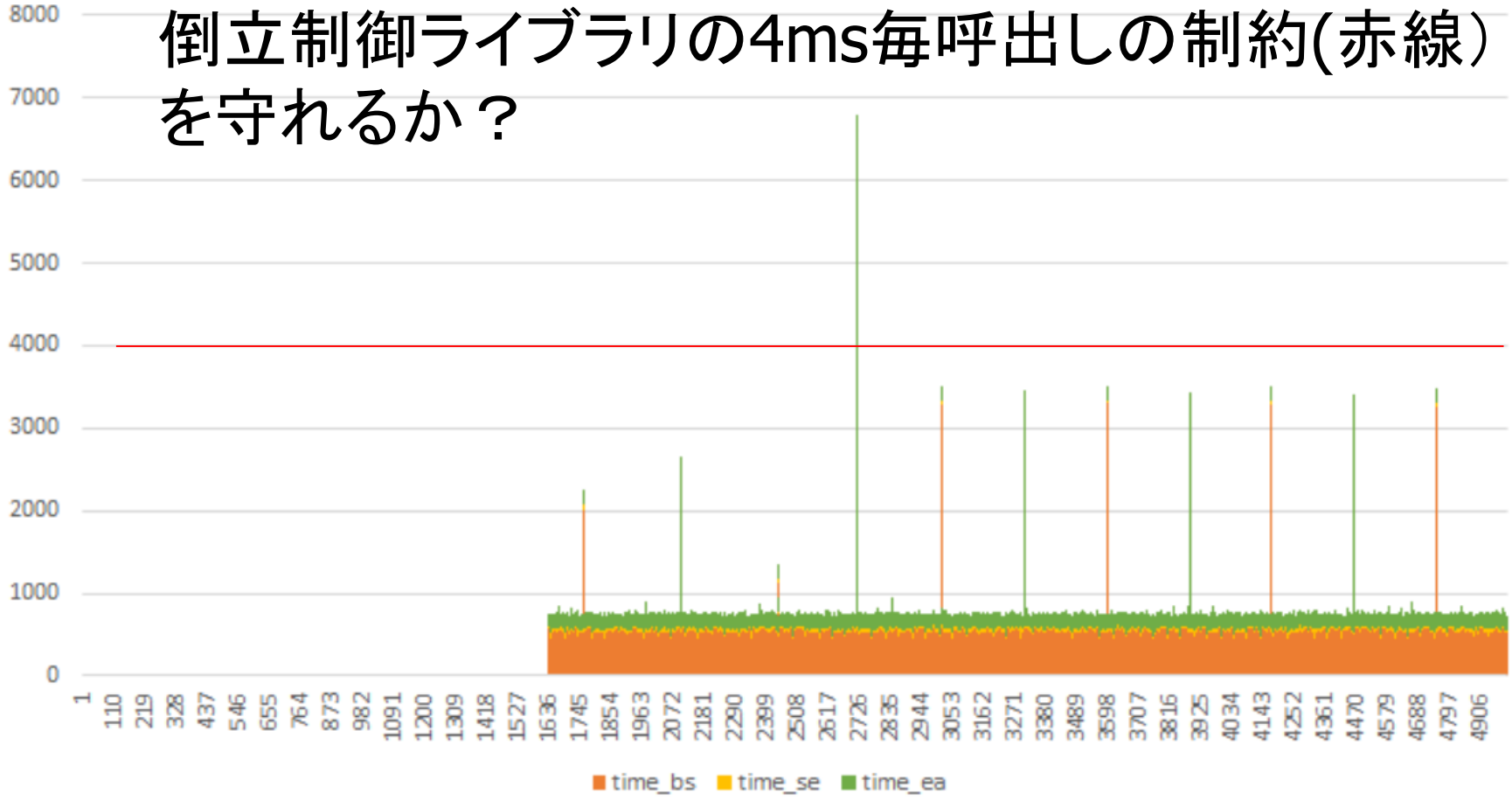
- Bluetooth
  - mrbcでスクリプトをコンパイルしてmrbファイルとして出力。**スクリプトを含まないものを一つのイメージにとしてSDカードに書き込む**。SDカードから起動して、Bluetoothを介して**mrbファイルを転送**して、スクリプトを実行。



TOPPERS

mruby on EV3RT+TECSのリアルタイム性能

倒立制御ライブラリの4ms毎呼出しの制約(赤線）を守れるか？



TOPPERS

環境構築

- パッケージをダウンロードして解凍-
https://www.toppers.jp/download.cgi/mruby-on-ev3rt+tecs_package-beta2.1.0.tar.gz
- Windows7、 Windows８、Windows８.1、Windows10
- Cygwinインストール
  - Ruby、GNU Make、bison
- クロスコンパイラ
  - arm-none-eabi-gcc.exe (GNU Tools for ARM Embedded Processors 6-2017-q1-update) 6.3.1 20170215 (release) [ARM/embedded-6-branch revision 245512]
  - https://developer.arm.com/-/media/Files/downloads/gnu-rm/6_1-2017q1/gcc-arm-none-eabi-6-2017-q1-update-win32.exe
- mkimage
  - Windows用バイナリはパッケージに同梱
- Cygwin及びクロスコンパイラのインストールはEV3RTの開発環境構築を参照してください。
  - http://dev.toppers.jp/trac_user/ev3pf/wiki/DevEnvWin

TOPPERS

環境構築：ディレクトリ構造

- bin
  - Windows向け開発環境のバイナリ
  - mkimageを含む
- doc
  - mrubyリファレンス
  - ビルド手順
  - サンプルプログラムの説明
- hr-tecs
  - TOPPERS/HRP2及びEV3プラットフォーム
  - サンプルプログラム（hr-tecs/workspace/mruby_app）
- mruby
  - mruby ver1.2.0
  - build_config.rbでEV3用のクロスコンパイルを指定

TOPPERS

環境構築：パスの通し方

- クロスコンパイラをインストールしたディレクトリにPATHを通す
  - C:¥Program Files (x86)¥GNU Tools ARM Embedded¥6 2017-q1-update¥bin

    ※フォルダ名は、クロスコンパイラのバージョンごとに変わります。
- arm-none-eabi-gccへパスが通っているかを確認

```
$ arm-none-eabi-gcc --version
arm-none-eabi-gcc.exe (GNU Tools for ARM Embedded Processors 6-2017-q1-update) 6.3.1 20170215
 (release) [ARM/embedded-6-branch revision 245512]
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

- コンパイラバージョンが表示されない場合はPATHを確認

TOPPERS

mrubyのビルド

- mrubyのビルド（host用コンパイル→ARM用クロスコンパイル）
- パッケージを展開したディレクトリで
  - $ cd mruby
  - $ make
  - mrubyディレクトリでmakeを実行すると、mrubyがビルドされる。

※ビルドには、ruby及びbisonのインストールが必要
下記の出力がされればビルド成功

```
==================================================
      Config Name: ARM
 Output Directory: build/ARM
    Included Gems:
              mruby-print - standard print/puts/p
              mruby-toplevel-ext - toplevel object (main) methods extension
              mruby-compiler - mruby compiler library
==================================================
```


TOPPERS

EV3Way向けとETロボコン向け設定

- 各種センサ、各モータのポートへの接続
  - EV3WayとETロボコンでは異なるポートに接続する。
- ポートの接続の割り当てを記録するクラスParameterを用いて、そこに記録したポート番号を利用する。
- Makefileの変数RUNNING_BODYにev3wayまたはetroboを指定することで、適切なポートの接続が記録される。
  - hr-tecs/workspace/sd/Makefile
  - hr-tecs/workspace/Bluetooth/Makefile

```
#RUNNING_BODY = ev3way
RUNNING_BODY = etrobo
```

EV3Way向けとETロボコン向け設定

	EV3Way向	ETロボコン向
タッチセンサポート	:port_1	:port_1
カラーセンサポート	:port_2	:port_3
ジャイロセンサポート	:port_3	:port_4
超音波センサポート	:port_4	:port_2
尻尾モータポート	:port_a	:port_a
右モータポート	:port_b	:port_b
左モータポート	:port_c	:port_c

ETロボコン2018向けデカタイヤ対応

- ETロボコン2018で採用されたタイヤについて
  - サイズが大きいため、EV3WayやETロボコン2017以前の倒立制御ライブラリのままでは倒立制御が難しくなった。
- 倒立制御ライブラリのパラメータ変更
  - 倒立制御ライブラリのパラメータをデカタイヤ用に変更。
  - Makefileの変数RUNNING_BODY_TIREにnormal(EV3WayやETロボコン2017以前のタイヤの場合)またはbig(ETロボコン2018のタイヤの場合）を指定することで、適切なパラメータ値が設定される。
    - hr-tecs/workspace/sd/Makefile
    - hr-tecs/workspace/Bluetooth/Makefile

```
#RUNNING_BODY_TIRE = normal
RUNNING_BODY_TIRE = big
```

  - 参考
    - 2018年走行体EV3way(二輪倒立)について 前編 - Qiita

TOPPERS

ETロボコン2018向けデカタイヤ対応

- バックラッシュキャンセル処理の追加
  - タイヤが大きくなり、バックラッシュの影響が増えた。
  - 今回、バックラッシュキャンセルを、モータエンコーダの実測値からバックラッシュの半分の値を引いてから、C言語の倒立制御ライブラリを呼び出すことで実現した。
  - このためmrubyのクラスメソッドBalancer.controlの最後に引数backlashhalfを追加した。
    - Balancer.control(forward, turn, gyro, gyro_offset, angle_l, angle_r, battery, backlashhalf)
  - ノーマルタイヤの場合は引数Backlashhalfを0にする。この場合バックラッシュキャンセル処理自体を行わない。
  - サンプルプログラムではBacklashhalfの値を4としているが、個体差などもあるため、調整する必要があるかもしれない。
  - 参考
    - 2018年走行体EV3way(二輪倒立)について 後編 - Qiita

TOPPERS

Makefileに追加した変数の値の確認

- Makefileに追加した変数RUNNING_BODY,RUNNING_BODY_TIREの値を確認するには、makeにターゲットshow-configを指定する
  - hr-tecs/workspace/sd/Makefile
  - hr-tecs/workspace/Bluetooth/Makefile

```
$ make show-config
RUNNING_BODY=etrobo
RUNNING_BODY_TIRE=big
```

二つの開発方法

- SD
  - アプリケーションのバイナリファイルもまとめてSDカードへコピーし，起動する．（完成時はこちらを推奨）

  ○：アプリケーションもSDカードに組み込んでいるため，安定する

  △：PCとEV3をUSBケーブルで接続すると、SDカードを差したまま読み込み／書き込みが可能になる。
- Bluetooth
  - mrubyのバイトコードをホストPC (開発マシン)からBluetoothを用いて転送し，起動する．

  ○：SDカードの抜き差しが不要になり，開発が効率的

  ×：Bluetooth機能のあるPCや機器が必要

  × ：現在、同一バイト―コードを連続して2回転送しないと、バイトコードの実行が開始しない。

  × ：バイトコード実行からバイトコード転送を繰り返して、正常に実行される場合もあるし、転送ができない場合、バイトコード実行の段階で電源断になる場合もある。

TOPPERS

SD

SD：コンパイル手順

- EV3RT+TECS・EV3ドライバ・ Mrubyスクリプトのビルド準備
  - SDカードへのコピー準備
    - デフォルトでは、cygwinが使用されていることを想定の上で、Eドライブにコピーされる（SDカードを挿入したEV3とPCをUSBケーブル接続すると、SDカードを抜き差しせずにコピー出来る）。
    - ドライブを変更するには、サンプルコードのMakefileを編集。
      hr-tecs/workspace/sd/Makefile
    - Makefile内を "SD_DIR" で検索して、変数にSDカードドライブを指定
    - SDカードドライブのドライブレターに合わせてください
    - ここで指定したディレクトリにEV3用イメージファイルがコピーされます

```
#
# SDのドライブ文字を指定
SD_DIR = /cygdrive/e/
```

ここの名前をSDカードのドライブ名に変更

TOPPERS

コンパイル手順

- EV3RT+TECS・EV3ドライバ・Mrubyスクリプトのビルド
  - パッケージを展開してディレクトリで
  - $ cd hr-tecs/workspace/sd/
  - $ make tecs
    - GNU Makeがtecsgenを実行してくれます
  - $ make depend
    - ファイルの依存関係を抽出します
    - ヘッダファイルなど、読み込まれるファイルを更新していなければ、実行する必要ありません。
  - $ make mrb
    - make tecs, make dependは最初に1回実行すればよい
    - Mrubyスクリプト(*.rb)を修正したり、MakefileでMrubyスクリプト名を変更した場合、make mrbを実行すると、確実にmrbcによるバイトコードコンパイルを行った後、全体のビルドを行う。

コンパイル手順：mrubyバイトコード

- Makefileを編集する
  (hr-tecs/workspace/sd/Makefile)

```
#  mrubyのアプリケーションファイル名
#APP_NAME = battery_sample.rb
APP_NAME = button_sample.rb
#APP_NAME = color_sample.rb
#APP_NAME = color_sample2.rb
#APP_NAME = ev3way_sample.rb
#APP_NAME = gyro_sample.rb
#APP_NAME = lcd_sample.rb
#APP_NAME = lcd_sample2.rb
#APP_NAME = lcd_sample3.rb
#APP_NAME = led_sample.rb
#APP_NAME = motor_sample.rb
#APP_NAME = motor_sample2.rb
#APP_NAME = rtos_sample.rb
#APP_NAME = speaker_sample.rb
#APP_NAME = speaker_sample2.rb
#APP_NAME = touch_sample.rb
#APP_NAME = ultrasonic_sample.rb
```

実行したいアプリケーション
を一つ選ぶ

TOPPERS

コンパイル手順：①make tecs

```
$ make tecs
  TECSGEN tEV3Sample.cdl
../../tecsgen/tecsgen/tecsgen.exe -k euc --cpp="arm-none-eabi-gcc -E"
```

```
"TACP_KERNEL"
"TACP_KERNEL"
"TACP_KERNEL"
"TACP_KERNEL"
acv = ["TACP_KERNEL", "TACP_KERNEL", "TACP_KERNEL", "TACP_KERNEL"]
===== end check regions HRP2Kernel =====
===== end tKernel plugin =====
touch tecs.timestamp
```

tecs.timestampが出力されれば成功

コンパイル手順：②make depend

```
$ make depend
if ! [ -f Makefile.depend ]; then \
        rm -f kernel_cfg.timestamp kernel_cfg.h kernel_cfg.c kernel_mem2.c ; \
        rm -f cfg1_out.c cfg1_out.o cfg1_out cfg1_out.syms cfg1_out.srec; \
        rm -f makeoffset.s offset.h; \
fi
rm -f Makefile.depend
  CFG[1]  cfg1_out.c
  CC      cfg1_out.c
  LINK    cfg1_out
  NM      cfg1_out.syms
  OBJCOPY cfg1_out.srec
  CFG[2]  kernel_cfg.timestamp

touch -r kernel_cfg.c kernel_cfg.timestamp
  CFG[3]  offset.h

Generating Makefile.depend.
```

コンパイル手順：③make mrb

```
$ make mrb
```

```
../../../bin/mkimage.exe -A arm -O linux -T kernel -
 "hrp2 kernel" -d hrp2.bin uImage
Image Name:   hrp2 kernel
Created:      Mon May 09 16:40:50 2016
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    1096112 Bytes = 1070.42 kB = 1.05 MB
Load Address: c0008000
Entry Point:  c0008000
chmod +x uImage
cp uImage /cygdrive/h/
```

このような出力が出れば成功

下記のようなエラーになる場合は、SD_DIRの指定が間違っている
もしくは、SDカードを認識していない（ささってない）

```
cp uImage /cygdrive/h/
cp: 通常ファイル `/cygdrive/h/' を作成できません: No such file or directory
Makefile:550: ターゲット 'uImage' のレシピで失敗しました
make: *** [uImage] エラー 1
```

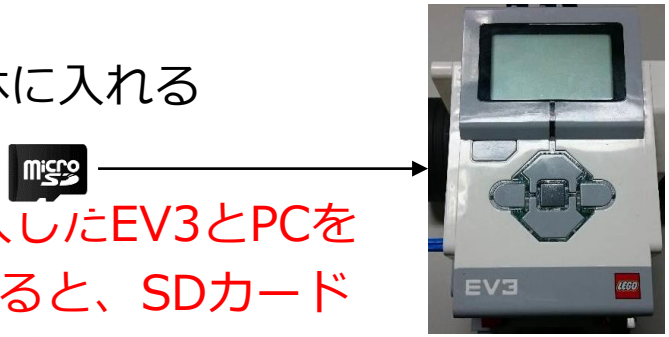

TOPPERS

コンパイル手順：uImage

- ビルドの確認
  - mkimageが生成され、uImageがコピーされていれば、ビルド成功です

```
../../../bin/mkimage.exe -A arm -O linux -T kernel -
 "hrp2 kernel" -d hrp2.bin uImage
Image Name:   hrp2 kernel
Created:      Mon May 09 16:40:50 2016
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    1096112 Bytes = 1070.42 kB = 1.05 MB
Load Address: c0008000
Entry Point:  c0008000
chmod +x uImage
cp uImage /cygdrive/h/
```

  - SDカードのルートディレクトリに、uImageというファイルができます
  - SDカードをEV3本体に入れる



  - 予めSDカードを挿入したEV3とPCを
  - USBケーブル接続すると、SDカード
  - を抜き差しせずにコピーできる。

TOPPERS

アプリケーションの起動方法

- SDカード入れたEV3の電源を入れる

中央（Enter）ボタンで電源オン

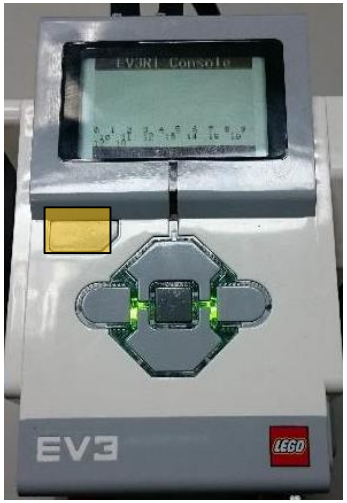
OS起動中

LEDが赤の間はOSの起動中

中央（Enter）ボタンでmrubyプログラム開始

LEDが緑になればOS起動完了

アプリケーションの実行



戻る（Back）ボタン長押しでスタート画面へ

戻る(Back)ボタン、右ボタン、左ボタンの同時押しで電源断。

TOPPERS

コンパイル手順：サンプルプログラムの変更の場合

サンプルプログラムの変更後はmake mrbだけで良い
（make tecs,　make dependは一度だけ）

```
#  mrubyのアプリケーションファイル名
#APP_NAME = battery_sample.rb
APP_NAME = button_sample.rb
#APP_NAME = color_sample.rb
#APP_NAME = color_sample2.rb
#APP_NAME = ev3way_sample.rb
```

```
#  mrubyのアプリケーションファイル名
#APP_NAME = battery_sample.rb
#APP_NAME = button_sample.rb
APP_NAME = color_sample.rb
#APP_NAME = color_sample2.rb
#APP_NAME = ev3way_sample.rb
```

サンプルプログラムの変更

make mrb

ev3way_sample

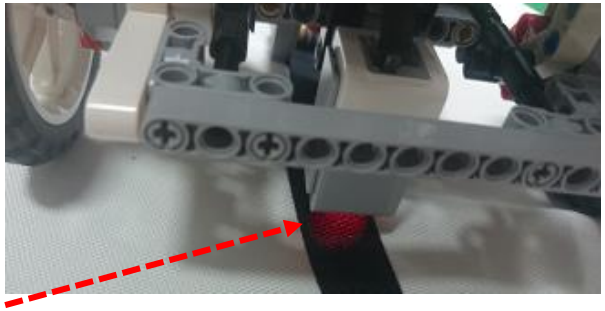
- ETロボコン用のサンプル
  - 倒立制御しながら、ライントレースを行う

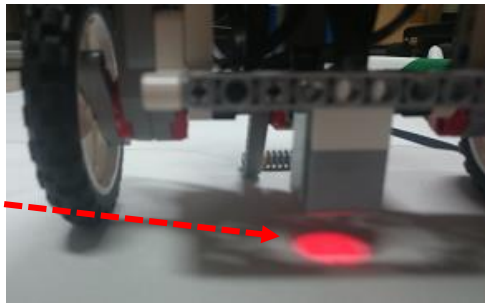
- タッチセンサ（Parameter.touch_sensor_port）
- カラーセンサ（Parameter.color_sensor_port ）
- ジャイロセンサ（Parameter.gyro_sensor_port ）
- 超音波センサ（Parameter.sonar_sensor_port ）

- しっぽモータ（Parameter.tail_motor_port ）
- 右モータ（Parameter.right_motor_port）
- 左モータ（Parameter.left_motor_port）

ev3way_sample

- 操作手順
  - 電源を入れる
  - 黒色のライン上にカラーセンサを移動
  - タッチセンサを押す：黒色の値を取得
  - 白色の上にカラーセンサを移動
  - タッチセンサを押す：白色の値を取得
    ：しっぽを下ろす
  - ライン上移動
  - タッチセンサを押す：ライントレーススタート





ev3way_sample.rb：初期化

```
begin
  LCD.puts "ev3way_sample.rb"
  LCD.puts "--- mruby version ---"
  Speaker.volume = 1
  forward = turn = 0
  #各オブジェクトを生成・初期化する
  @sonar = UltrasonicSensor.new(SONAR_SENSOR)
  @color = ColorSensor.new(COLOR_SENSOR)
  @color.reflect
  @touch = TouchSensor.new(TOUCH_SENSOR)
  @gyro  = GyroSensor.new(GYRO_SENSOR)

  motor_l = Motor.new(LEFT_MOTOR)
  motor_r = Motor.new(RIGHT_MOTOR)
  @motor_t = Motor.new(TAIL_MOTOR)
  @motor_t.reset_count
  #各オブジェクトを生成・初期化する
  LED.color = :orange
…
```

ひとつしかないもの（ポート番号指定不要）は、クラスメソッドとして直接呼び出す

ポート番号を指定して初期化（インスタンス化）

TOPPERS

ev3way_sample.rb：黒色、白色の取得

```
#黒・白色のキャリブレーション
  black_value = color_calibration
  LCD.puts "black::#{black_value}"
  white_value = color_calibration
  LCD.puts "white::#{white_value}"

  threshold = ((black_value + white_value) / 2).round
  #  スタート待機
   LCD.puts "Ready to start"
```

ライントレースの基準値を計算

```
def color_calibration(n=10)
  while true
    break if @touch.pressed?
    RTOS.delay(10)
  end
  col = 0
  n.times { col += @color.reflect}
  col = (col / n).round
  Speaker.tone(:a4, 200)
  RTOS.delay(500)
  col
end
```

タッチセンサが押されるまで待つ

カラーセンサn回取得し、平均値を取得

TOPPERS

ev3way_sample.rb：スタート準備

```
#  スタート待機
  LCD.puts "Ready to start"
  while true
    # 完全停止用角度に制御
    tail_control(TAIL_ANGLE_STAND_UP)
    RTOS.delay(10)
    #  タッチセンサが押されるまで待つ
    break if @touch.pressed?
  end
  #走行モータエンコーダーリセ
  motor_l.reset_count
  motor_r.reset_count
  # ジャイロセンサリセット
  @gyro.reset

  # LED:緑 走行状態
  LED.color = :green
```

しっぽの位置を指定された角度に保つ
（フィードバック制御）

目標値

現在の値

```
def tail_control(angle)
  pwm = ((angle - @motor_t.count) * P_GAIN).to_i
  if pwm > PWM_ABS_MAX
    pwm = PWM_ABS_MAX
  elsif pwm < -PWM_ABS_MAX
    pwm = -PWM_ABS_MAX
  end
  if pwm == 0
    @motor_t.stop(true)
  else
    @motor_t.power = pwm
  end
end
```


TOPPERS

ev3way_sample.rb：ライントレース

障害物まで一定の距離以下になると止まる

```
# main loop
  forward = turn = 0
  while true
    start = RTOS.msec
    # バランス走行用角度に制御
    tail_control(TAIL_ANGLE_DRIVE)
    # 障害物検知
    if sonar_alert
      forward = turn = 0
    else
      # Line trace
      forward = 30
      turn = @color.reflect >= threshold ? 20 : -20
    end
    …
  end
```

```
def sonar_alert
  @sonar_counter += 1
  if @sonar_counter == 10
    distance = @sonar.distance
    @sonar_alert_var = distance <=
        SONAR_ALERT_DISTANCE
        && distance >= 0
    @sonar_counter = 0
  end
  @sonar_alert_var
end
```

サンプルでは、30に固定

カラーセンサと閾値と比較し
どちらかに曲がる
ここを変更すると、
自前のライントレースが可能

TOPPERS

ev3way_sample.rb：倒立制御

```
# main loop
  while true
    ...
    # 倒立振子制御APIを呼び出し、倒立走行するための
    # 左右モータ出力値を得る */
    pwm_l, pwm_r = Balancer.control(
      forward, turn,
      @gyro.rate, GYRO_OFFSET,
      motor_l.count, motor_r.count,
      Battery.mV, BACKLASHHALF)
    if pwm_l == 0
      motor_l.stop(true)
    else
      motor_l.power = pwm_l
    end
    if pwm_r == 0
      motor_r.stop(true)
    else
      motor_r.power = pwm_r
    end
    RTOS.delay(wait)
  end
```

バランサの返り値が2つ

C言語で実装されたバランサを呼び出す
ETロボコン2018で採用されたデカタイヤを用いる場合はBACKLASHHALF=4が適当であろう。ただし、走行体毎に調整する必要もあるだろう。
ノーマルなタイヤの場合は0にする。

ループ内を4ミリ秒周期で実行
wait= 3(ミリ秒) でdelayする
(1ミリ秒+ 3ミリ秒=4ミリ秒)
現状1ミリ秒程度で処理完了
mrubyでも十分制御可能

エラーの対処方法

mrubyプログラム起動時に電源がオフになる場合

- 物理的にささっているポートとプログラムのポートを確認
- 例：プログラムではポートAを使おうとして、物理的にはポートAになにも接続されていない場合
- スタックサイズが4096バイトの場合にこの症状が発生し、スタックサイズを増やすと発生しなくなるという報告がある。サンプルプログラムではスタックサイズに余裕を持たせて81920バイトにしている。

mrubyプログラム実行時に電源がオフに出来ない

- 戻る（BACK）ボタンの長押し、または戻る（BACK）ボタン、左ボタン、右ボタンの同時長押しをしても、電源オフに出来ない場合があります。

- この場合は、EV3充電式バッテリーまたは乾電池(又は充電池）を外して電源断にします。

パッケージ内のドキュメント

- doc/EV3RT_mruby_API_Reference.pdf
  - APIリファレンスマニュアル
- doc/mruby_on_ev3rt+tecs_buld.pdf
  - 環境設定
- doc/mruby_sample.pdf
  - サンプルプログラムの説明

問い合わせ先・リンク

- 質問などは下記のMLに
  - users@toppers.jp
- EV3RT
  - http://http://dev.toppers.jp/trac_user/ev3pf/wiki/WhatsEV3RT
- TOPPERS ETロボメーリングリスト
  - https://www.toppers.jp/etrobo.html#etrobo-ml

付録

mruby gems

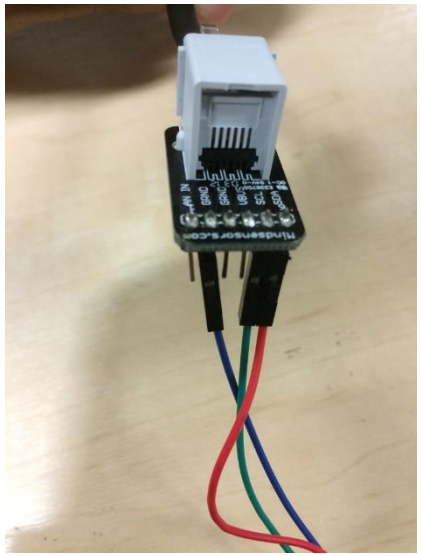
- 以下の特徴を持つ、mrubyのライブラリ
  - CやmrubyのソースコードをmrubyVMから実行する関数として組込める
  - 通常ビルド時に、Webからライブラリを追加できる
- 配布パッケージではmruby-printを追加済み
  - EV3シリアル出力をリンクし、print, p, putsなどを実現しています

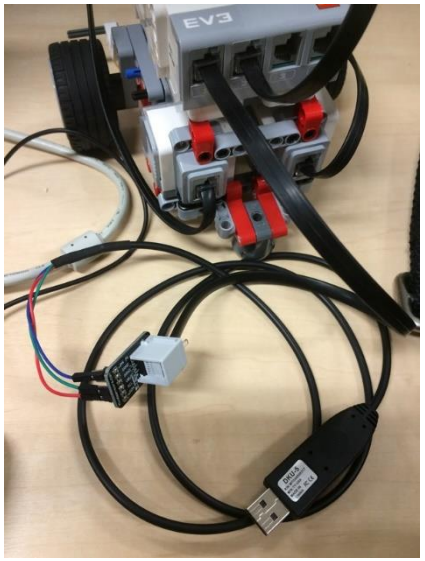
シリアル通信

- mruby の puts, print, p は、シリアル通信から出力されます。
- シリアルケーブルはEV3本体のPort1を使用します。
  - (シリアルに使用するため、現在Port1にセンサを取り付けできない制約があります)
- コンピュータへの接続にはUSB-Serial変換ケーブルおよびEV3-Serial変換コネクタが必要となります。
- EV3RTの機能として、シリアルケーブル接続によるシリアル通信とBluetooth接続によるシリアル通信を切り替え可能です(/ev3rt/etc/rc.local.ini) 。
- 以降のシリアル通信についての記述は、（物理的な接続を除いて）Bluetooth接続によるシリアル通信についても当てはまります。

シリアル通信

- 動作確認では以下のシリアルケーブルおよびコネクタを使用しました
- OLIMEX USBシリアル変換ケーブル(3 線式)
  - ※2015年6月現在、Windows 8で使用ができません (ドライバがありません)
- Breadboard Connector Kit for NXT(BCK01)





シリアル通信

- 上記シリアルケーブルおよびConnector Kitを使用する場合、それぞれ以下が対応します。
- 青(GND) - GRND (2 or 3)
- 緑(TX) - SCL (5)
- 赤(RX) - SDA (6)

/ev3rt/etc/rc.conf.iniの設定

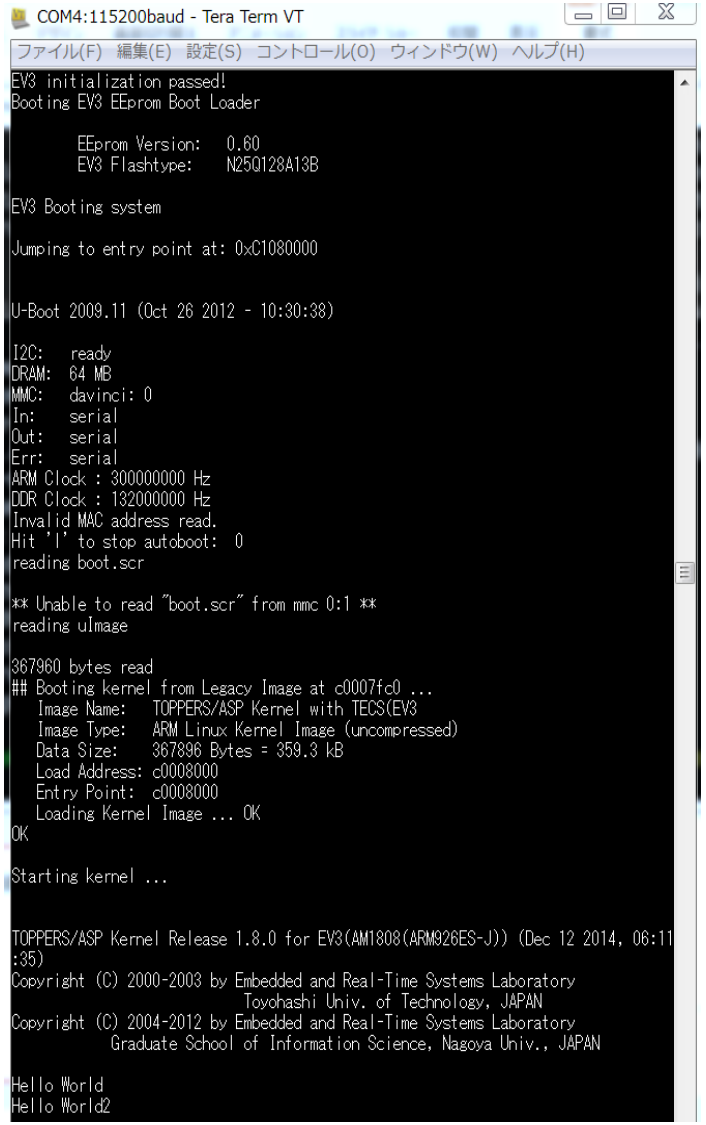
- EV3RTの機能として、設定ファイルでSDカードの/ev3rt/etc/rc.conf.iniで動作ログの出力先（デバッグポート）を指定できる。

```
[Debug]
DefaultPort=UART
#DefaultPort=BT
#DefaultPort=LCD
```

- UART -> シリアル通信
- BT -> Bluetooth接続でシリアル通信
- LCD -> LCDに出力（LCDには、LCDクラスの表示メソッド以外にも、 puts, print, p メソッドでも出力される）

Tera Termとの接続

- Tera Termの設定
  - ポート：USBのシリアルポート
  - ボーレート：115200
- PCとEV3をシリアルで接続後EV3の真ん中のボタンを押して起動
- 起動確認
  - 起動時にuImageを直接読み込んで、TOPPERS/HRP2が自動的に起動します



TOPPERS