

mruby-on-ev3rt+tecs_package ビルド手順

安積卓也（大阪大学）
長谷川涼（大阪大学）
山本拓朗（大阪大学）
小南靖雄（TOPPERS個人会員）

最終更新日：2018/ 5/16

目次

- 環境構築
- mrubyのビルド
- 開発方法
 - Bluetooth
 - コンパイル手順
 - アプリケーションの起動方法
 - SD
 - コンパイル手順
 - アプリケーションの起動方法
- エラー対処方法
- 付録

環境構築

- Windows7、 Windows 8、 Windows 8.1、 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

環境構築

- Tera Term
 - Bluetoothを用いてアプリケーションを使用する場合に使用
- (Bluetooth Stack for Windows by Toshiba)
- (Microsoft標準ドライバ)

Bluetoothの接続手順

- 対応機器およびBluetooth接続手順はEV3RTの開発環境構築を参照してください。
 - Windows
 - http://dev.toppers.jp/trac_user/ev3pf/wiki/BluetoothWin

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

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

mrubyのビルド

- mrubyのビルド（host用コンパイラ→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
=====
```

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

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

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](#)

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

● 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。

● 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

Bluetooth : 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できる) 。
 - 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/bluetooth/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/
```

この名前をSDカードのドライブ名に変更

Bluetooth : 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たディレクトリで
 - `$ cd hr-tecs/workspace/bluetooth/`
 - `$ make tecs`
 - GNU Makeがtecsgenを実行します
 - genディレクトリにTECS用ソースコードが自動生成されます。
 - `$ make depend`
 - ソースの依存関係を調べる
 - `$ make mrb`
 - Bluetoothローダー付きMruby VM作成
 - `$ cd ../mruby_app`
 - サンプルMrubyスクリプトのディレクトリに移動
 - `$ make`
 - mrubyのコンパイラでバイトコード作成。

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 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
```

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

- 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
- 差しせずにファイルをコピー出来る。



コンパイル手順：mrubyバイトコード

- 作業ディレクトリに移動する

```
$ cd ../mruby_app/
```

- cf. workspace以下のディレクトリ構造
workspace /mruby_app
 /bluetooth
 /sd

コンパイル手順：mrubyバイトコード

- Makefileを編集する
(hr-tecs/workspace/mruby_app/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

コンパイル手順 : mrubyバイトコード

- コンパイルする

```
$ make
```

- ソースファイル (.rb) をコンパイルすると,
バイトコード (.mrb) が生成される
 - ex.
 - ソースファイル (motor_sample.rb)
 - バイトコード (motor_sample.mrb)

アプリケーションの起動方法：起動①

- 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

スタート画面



LEDが赤
の間はOS
の起動中



LEDが緑になればOS起動完了



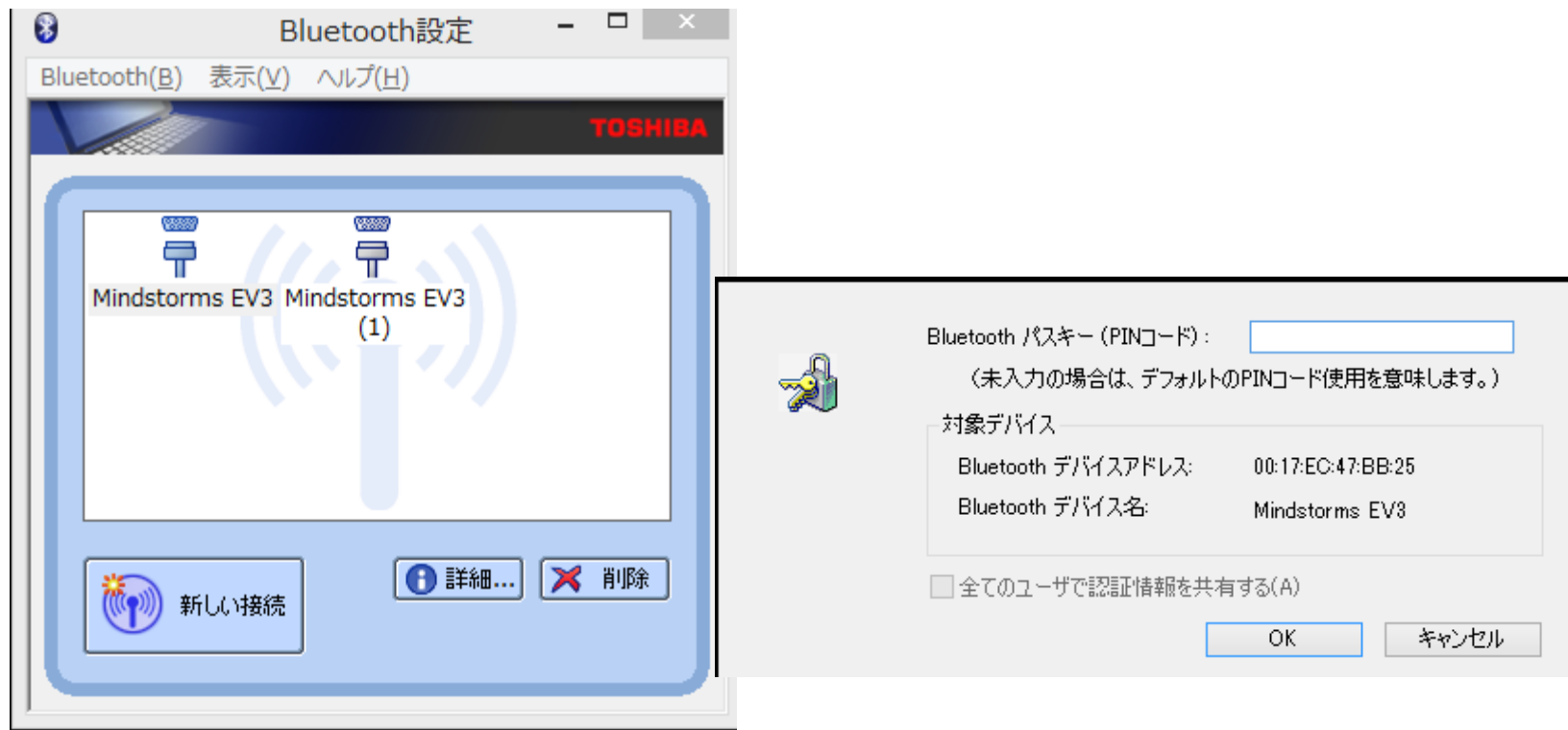
Bluetoothの
ペアリングを行う
(次ページ)

Load App → 受信待ち

Shutdown → 電源オフ

アプリケーションの起動方法：起動②

- EV3とホストPCをBluetoothで接続する
 - Mindstorms EV3を選択し，ペアリングする
 - (PINコードのデフォルト値は “0000”)



アプリケーションの起動方法：起動②

- ペアリングが完了した状態でLoad App

中央 (Enter) ボタンで
開始



Bluetooth Loaderの
受信待ち



戻る (Back) ボタン
長押しでスタート画面

アプリケーションの起動方法：起動③

- ホスト側からmrubyバイトコードを転送する
 - ①Tera Termを起動して，転送する方法
 - ②Cygwinのbash上から，コマンドで転送する方法

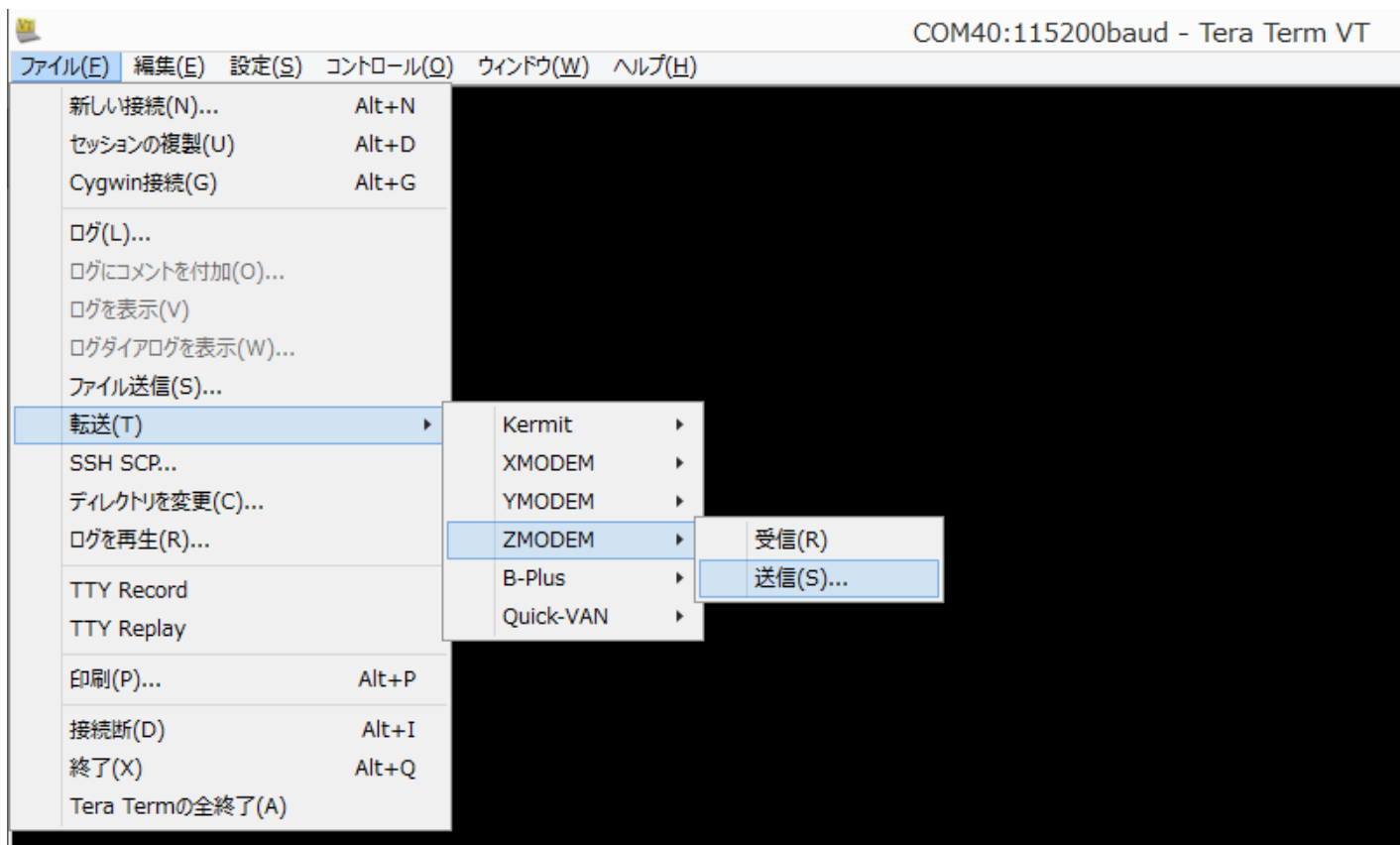
バイトコード転送 : Tera Term

- Tera Termの設定

- ポート : BluetoothのCOM番号
- ボーレート : 115200

- 転送する

- ファイル→転送→ZMODEM→送信
- 転送するバイトコードを選択



バイトコード転送 : Tera Term

- Tera Termのデフォルト値設定
- メニューからいちいち設定するのではなく、デフォルト値として設定しておきたければ、TERATERM.INIを変更します。
 - C:¥Program Files (x86)¥teraterm¥TERATERM.INI

```
[Tera Term]
ComPort=4
BaudRate=115200
```

バイトコード転送：Cygwin上から転送

- Tera Term に PATH を通す
 - C:¥Program Files (x86)¥teraterm
- Tera Term にパスが通ってるか確認
 - ttpmacro.exe が実行できる
（“コマンドが見つかりません”にならなければOK）

バイトコード転送：Cygwin上から転送

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

```
# コマンドから転送する場合, TRANSFER = true とする
# デフォルトでは, TRANSFER = となっている
TRANSFER = true

# COM番号
COM_PORT = 40
# ボーレート
BAUDRATE = 115200

# TTLファイル名
ZMODEM_TTL_TMPL = ../common/zmodemsend_template.ttl
ZMODEM_TTL = zmodemsend.ttl
```

Cygwin上から転送する場合,
TRANSFER = true とし, COM番号とボーレートを設定する

バイトコード転送：Cygwin上から転送

- mrubyコンパイル&転送

```
$ make
```

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/
```

この名前をSDカードのドライブ名に変更

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

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
```

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。

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。

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
#APP_NAME = ev3way_sample2.rb
```

サンプルプログラムの
変更



make mrb

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（又は充電電池）を外して電源断にします。

付録

mruby gems

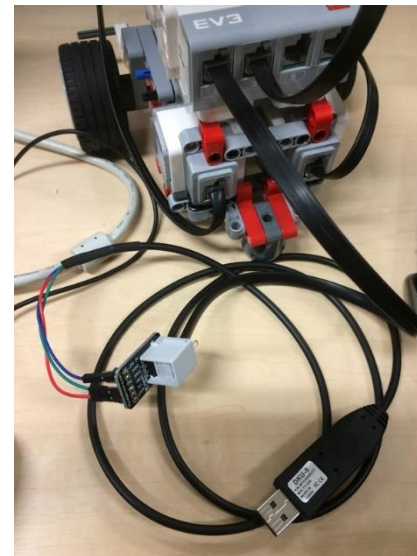
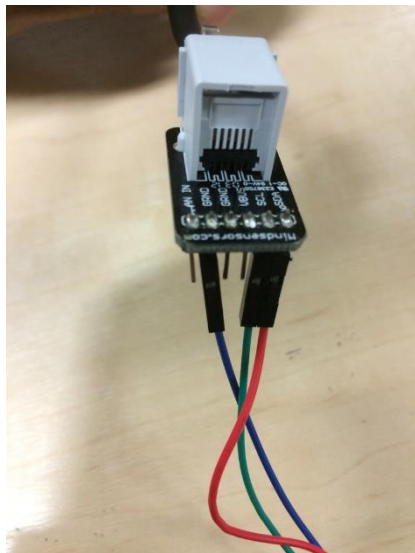
- 以下の特徴を持つ、mrubyのライブラリ
 - Cやmrubyのソースコードを、mruby VMから実行する関数として組込める
 - 通常ビルド時に、Webからライブラリを追加できる
- 配布パッケージではmruby-printを追加済み
 - EV3シリアル出力をリンクし、print, p, putsなどを実現しています

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

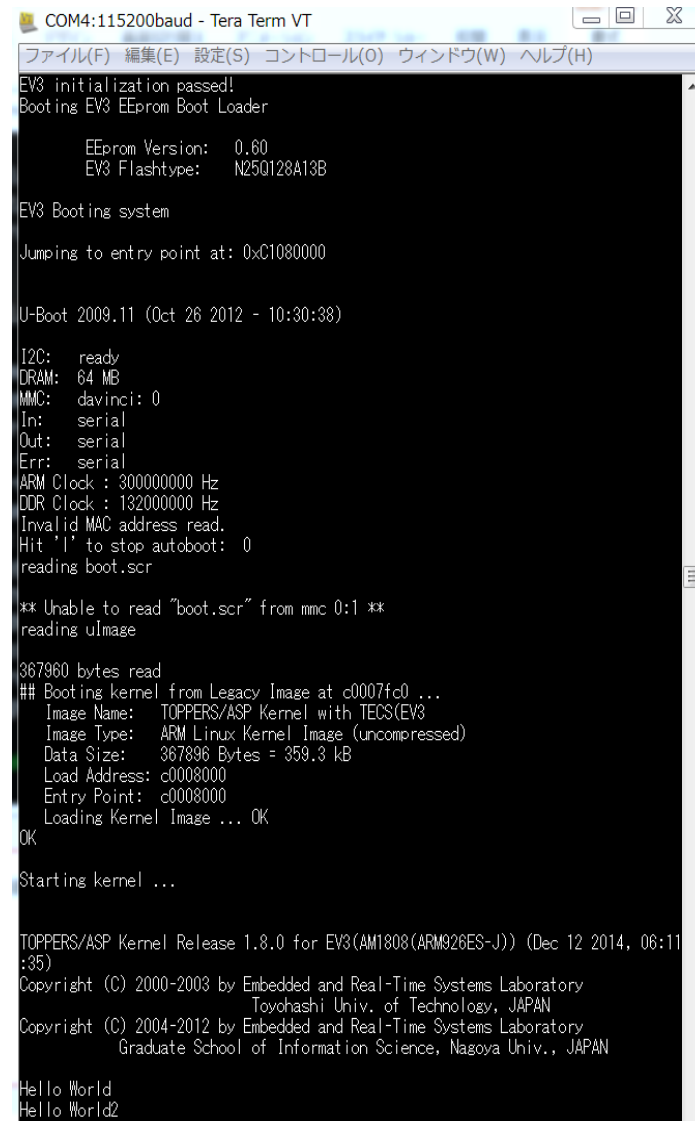
- EV3RTの機能として、設定ファイルでSDカードの /ev3rt/etc/rc.conf.iniで動作ログの出力先（デバッグポート）を指定できる。

```
[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



```
COM4:115200baud - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
EV3 initialization passed!
Booting EV3 EEprom Boot Loader

        EEprom Version: 0.60
        EV3 Flashtype: N25Q128A13B

EV3 Booting system
Jumping to entry point at: 0xC1080000

U-Boot 2009.11 (Oct 26 2012 - 10:30:38)

I2C: ready
DRAM: 64 MB
MMC: davinci: 0
In: serial
Out: serial
Err: serial
ARM Clock : 300000000 Hz
DDR Clock : 132000000 Hz
Invalid MAC address read.
Hit 'l' to stop autoboot: 0
reading boot.scr

** Unable to read "boot.scr" from mmc 0:1 **
reading uImage

367960 bytes read
## Booting kernel from Legacy Image at c0007fc0 ...
Image Name: TOPPERS/ASP Kernel with TECS(EV3)
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 367896 Bytes = 359.3 kB
Load Address: c0008000
Entry Point: c0008000
Loading Kernel Image ... OK

Starting kernel ...

TOPPERS/ASP Kernel Release 1.8.0 for EV3(AM1808(ARM926ES-J)) (Dec 12 2014, 06:11:35)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2004-2012 by Embedded and Real-Time Systems Laboratory
Graduate School of Information Science, Nagoya Univ., JAPAN

Hello World
Hello World2
```

mruby VMコンポーネント

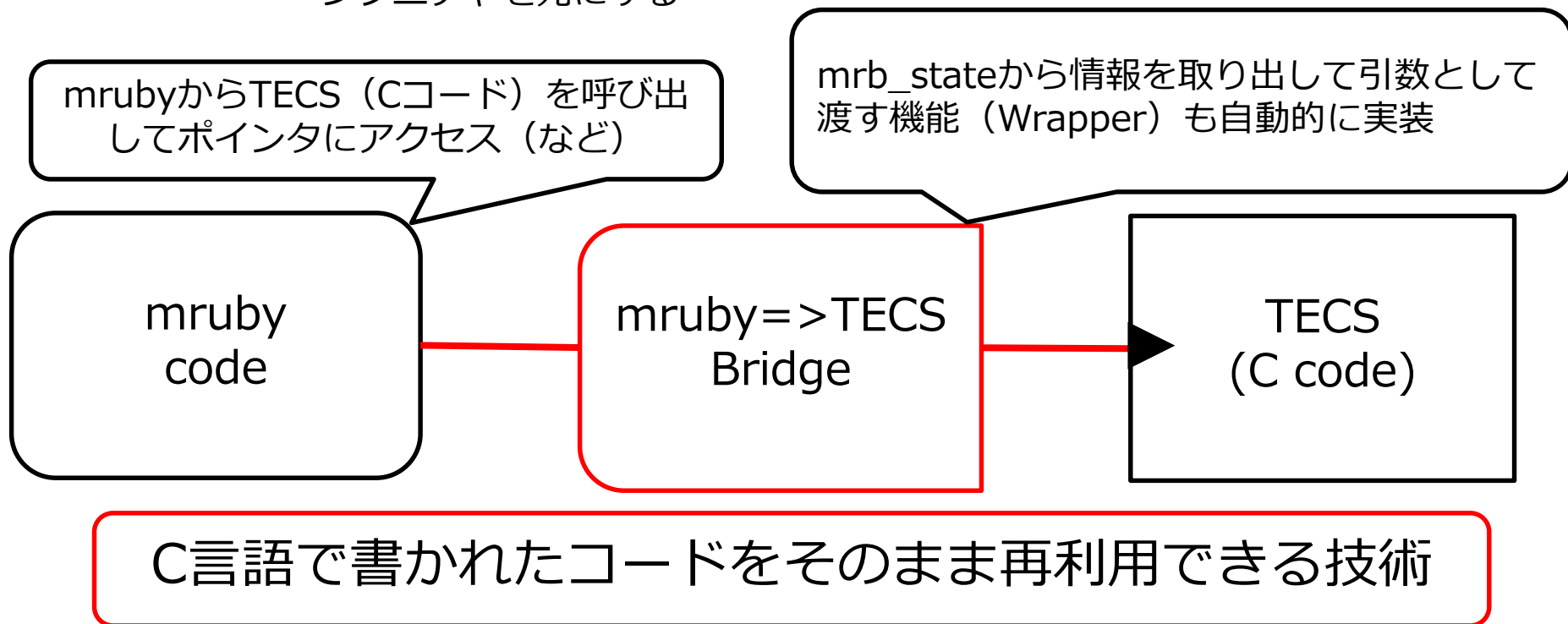
- VMはスクリプト言語のプログラムが動作する仮想機械
 - 実際には、直接スクリプト言語を実行するのではなく、mruby のソースコードをコンパイルして作られるバイトコードのプログラムを動作させます
- VMは中間役
 - VMではバイトコードを逐次実行します

mruby VM コンポーネント初期化手順

- tecsgen/tecs/mruby/nMruby_tMruby.cのeMrubyBody_main関数内で行われる
 - cell tTask MrubyTask1において、cBodyと接続される
- VMを用意（初期化）
 - mrb-open（メモリの確保）
- 中間コードのロード
 - mruby_irep（配列を読み込む）
- VMの実行
 - mrb_run

mruby TECS bridge

- mrubyからTECS（C言語）を呼び出すためのインタフェースを作成する
 - TECS側を元にmruby側に合ったインタフェースを作成
 - シグニチャを元にする



mruby TECS bridgeのCDL記述

- ブリッジセルタイプの生成
 - generate(MrubyBridgePlugin, sSpeaker, "");
- ブリッジセル本体
 - cell nMruby::tsSpeaker BridgeSpeaker {
 - cTECS = Speaker.eSpeaker;
 - };
- 組み上げ記述
 - cell nMruby::tRiteVM RiteVM{
 - mrubyFile="mrb_sample.rb";
 - //ブリッジセルの初期化と宣言
 - cInit = VM_TECSInitializer.eInitialize;
 - };

改版履歴

- 2015年6月8日 alpha 1.0.0
 - 安積卓也（大阪大学）
 - 長谷川涼（大阪大学）
- 2015年7月5日 alpha 1.0.2
 - 安積卓也（大阪大学）
 - 長谷川涼（大阪大学）
- 2016年5月10日 beta 1.0.0
 - 安積卓也（大阪大学）
 - 長谷川涼（大阪大学）
 - 山本拓朗（大阪大学）
- 2018年5月16日 beta 2.1.0
 - 安積卓也（大阪大学）
 - 長谷川涼（大阪大学）
 - 山本拓朗（大阪大学）
 - 小南靖雄（TOPPERS個人会員）