ETロボコン向けTOPPERS活用セミナー

EV3RTの概要

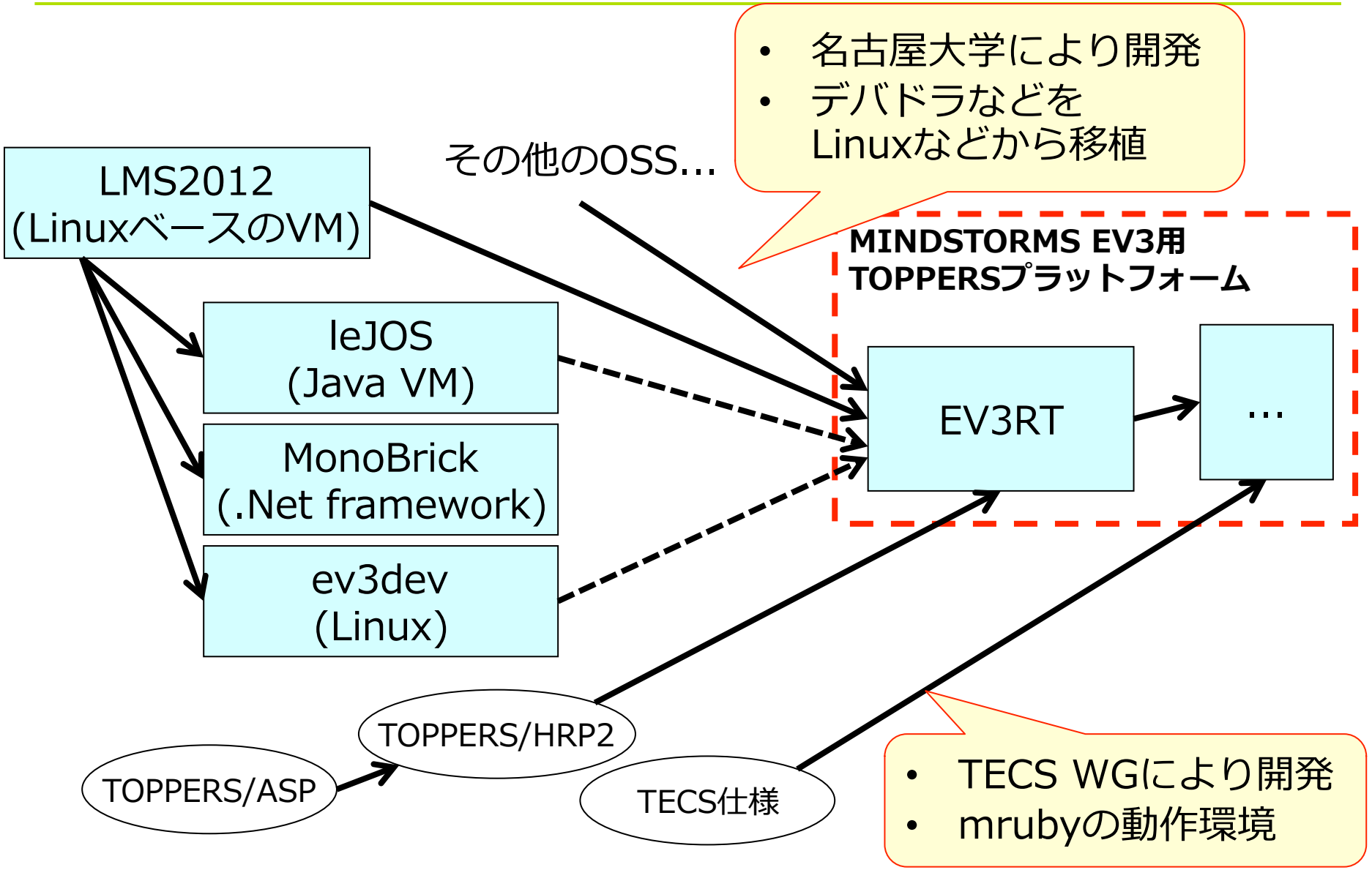
2015年6月20日(土)

名古屋大学　石川拓也

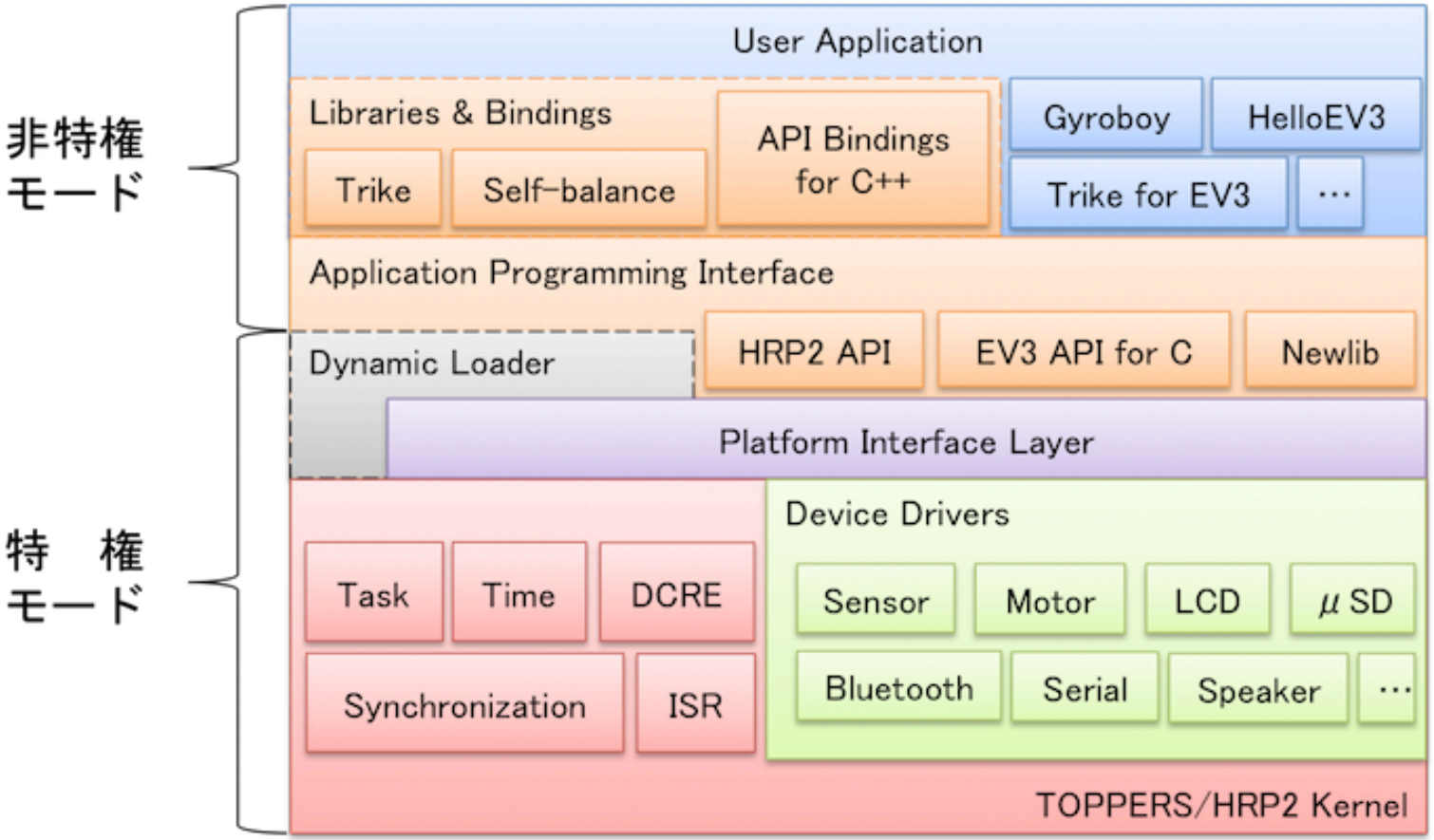
EV3RT

- Real-Time platform for EV3
  - http://dev.toppers.jp/trac_user/ev3pf/wiki/WhatsEV3RT
- TOPPERS/HRP2カーネルをベースとしたプラットフォーム
  - メモリ保護機能を提供
- CやC++で開発が可能
  - TOPPERS OS/newlib/EV3用のAPIを利用可能
- その他の利点
  - 起動時間が早い（約5秒．LMS2012では30秒以上）
  - メモリ消費量が少ない（LMS2012の約10%）
  - ダイナミックローディング機能（OSを再起動せずにアプリケーションを更新可能）
- ETロボコン2015での公式プラットフォーム

EV3用プラットフォーム



EV3RTのアーキテクチャ



- HRPカーネルの機能を利用し，アプリケーションを非特権モードで動作させ，カーネルやデバイスドライバなどを特権モードで動作させることで，アプリケーションの不具合を検出しやすくしている

EV3RTのインストール

- 開発環境をホストPCにインストール
  - http://dev.toppers.jp/trac_user/ev3pf/wiki/DevEnv の「開発環境(クロスコンパイラ，ツール)のインストール」を参考に
- パッケージをダウンロード
  - http://dev.toppers.jp/trac_user/ev3pf/wiki/Download から β5(ev3rt-beta5-release.zip)を取得
- パッケージを解凍
  - $ unzip ev3rt-beta5-release.zip
- カーネルソースコードを解凍
  - $ cd ev3rt-beta5-release
  - $ tar xvf hrp2.tar.xz

EV3RTパッケージのフォルダ構成

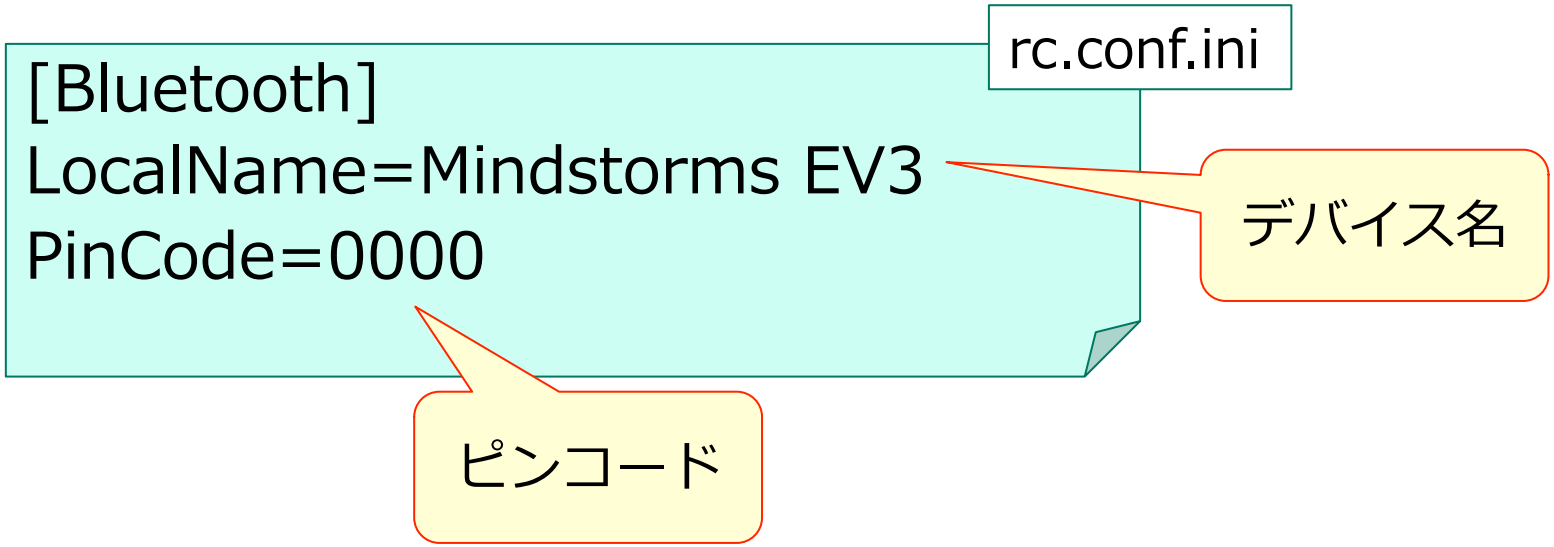
- Changelog.txt…リリースノート
- EV3RT_CPP_API_Reference…EV3RTのC++ APIリファレンス
- EV3RT_C_API_Reference…EV3RTのC APIリファレンス
  - index.htmlを開くと閲覧可能
- ngki_spec-170.pdf…TOPPERSカーネルの仕様書（Ver.1.7.0）
- sdcard…SDカードに保存するファイルのサンプル
  - EV3RTはカーネルやアプリケーションのイメージファイルをSDカードに保存し，EV3本体に挿入することで使用
- hrp2…EV3RTのソースコード本体
  - HRP2カーネル
  - デバイスドライバやミドルウェア
  - アプリケーションのワークスペース

SDカードに置くファイルのサンプル

- sdcardフォルダの中身
- uImage…EV3RTのカーネル，動的ローダのイメージファイル(EV3起動時にここから実行を始める)
- ev3rt…EV3RTの使用するファイル
  - apps…アプリケーションのロードイメージを置くフォルダ
  - etc/rc.conf.ini…Bluetoothデバイスの設定ファイル
  - res…サンプルアプリケーション(ファイルI/O)で使用しているフォルダ
    - /ev3rt/resというパスでアクセスできる

Bluetoothの設定ファイル

- ホストPCから見えるBluetoothデバイスの名前やペアリング時のピンコードを設定



EV3RTのフォルダ構成

- hrp2フォルダの中身
- EV3RTのサンプルアプリケーションおよびアプリケーションのワークスペースは workspace
- cfg はTOPPERS新世代カーネル用コンフィギュレータのフォルダ
  - cfg は EV3RTでのビルド時に必要なツール (静的APIのため)
  - Windows以外の環境ではcfgのバイナリを入れ替える必要がある
    - http://www.toppers.jp/cfg-download.html から環境にあったバイナリをダウンロードし，cfg/cfg/ に置く
- configureはアプリケーションのMakefileをテンプレートから生成するユーティリティ
- HRP2カーネルのソースコードは，arch / include / extension / kernel / library / pdic / syssvc / target
  - デバイスドライバなどは target/ev3_gcc/ のさらに下

アプリケーションのビルド

- EV3RTのworkspaceフォルダに移動してアプリケーションをビルド

```
$ cd ev3rt-beta5-release/hrp2/workspace/
```

- スタンドアローン形式のモジュールをビルドする場合は make app=<フォルダ名>

```
$ make app=helloev3
```

- 動的ローディング形式のモジュールをビルドする場合は make mod=<フォルダ名>

```
$ make mod=helloev3
```

スタンドアローン形式

- EV3RTとアプリケーションを一つのモジュールにリンクしてEV3RT起動とともにアプリケーションを実行開始する形式
- EV3RTとアプリケーションが一つのuImageファイルとなるため，SDカードのuImageファイルを置き換えることでアプリケーションを書き換える
  - SDカードのトップフォルダにuImageを置けばよい
  - EV3が起動すると，EV3のメモリに書き込まれているブートローダubootが起動し，SDカードにあるuImageファイルをSDRAMに展開してEV3RTを起動

動的ローディング形式

- EV3RTの提供する動的ローダにより，アプリケーションをEV3RTの起動後にロードする形式
  - アプリケーションモジュール(デフォルト名はapp)と，動的ローダを含んだEV3RTのモジュールを別々にビルドする
  - 動的ローダはスタンドアローン形式のEV3RTアプリケーションの一種
- 動的ロード用のアプリケーションモジュールはSDカードの/ev3rt/appsフォルダに置く
  - EV3RT(動的ローダ)起動中にBluetooth経由でアプリケーションモジュールを転送することも可能
    - 参考：http://dev.toppers.jp/trac_user/ev3pf/wiki/SampleProgram#Bluetoothまたはシリアルケーブルによるアプリケーションのロード方法

アプリケーションの実行

- (1a) スタンドアローン形式の場合はuImageをSDカードのトップに置く
- (1b) 動的ロード形式の場合はappを，Bluetooth経由もしくは直接SDカードの/ev3rt/apps/に置く
  - ファイル名をappから変更してもよい
- (2) EV3を起動(中央ボタンを押す)
- (3a) スタンドアローン形式の場合は「Run App」と画面下部に表示されているときに中央ボタンを押すとアプリケーションが起動
- (3b) 動的ロード形式の場合は「Load App」と画面下部に表示されているときに中央ボタンを押すと動的ローダが起動するので「SD card」を選択し，アプリケーションを選択して起動
  - 上下ボタンでカーソル移動，中央ボタンで決定
  - アプリケーション実行時にバックボタンを長押しするとEV3起動時のコンソールに戻る
  - EV3起動時のコンソールで右ボタンを押して「Shutdown」と画面下部に表示されているときに中央ボタンを押すと電源を切る

ビルドの仕組み

1. workspace/Makefileを使ってmake mod=test or app=test
2. workspace/test/Makefile.incをinclude
   - configureのパラメータを設定
3. workspace/OBJをmkdirしてOBJ/に移動
4. OBJ/でconfigureを実行して，Makefileを生成
   - workspace/testをパスに含めて，2の設定を使用
   - Makefileのテンプレートはtest/Makefile.appmod
   - make app=xxxの場合のテンプレートはMakefile.app
5. OBJ/で 4. で生成したMakefileを使ってmake
6. スタンドアローン版の場合は，objcopy で ELF形式のモジュールからバイナリ形式のhrp2.binを生成したあと，mkimageコマンドでhrp2.binからuImageを生成
   - ubootがロードするためのファイル形式に変換
7. workspace/にapp or uImageをコピー
   - appは動的ローディング形式のアプリケーションモジュール

新しくプロジェクトを作る

- 簡単な方法は既存のプロジェクトをコピーする

1. cp –a helloev3 new_proj
2. 不要なソースコードファイルを削除し，必要なファイルを追加
   - 注意：app.c / app.cpp / app.cfg はデフォルトで必要となるファイルなので削除しないこと！この名前を変える場合は，workspace/Makefileの"-A app"となっている場所を変更するか，new_proj/Makefile.app[mod]のAPPLNAMEを直書きすればよい
   - app.c 以外のソースコードを追加する場合は 3. でビルド対象とすればよく，app.cfg以外のcfgを追加する場合はapp.cfgからINCLUDEすればよい
3. Makefile.incでアプリケーションの設定
   - APPL_COBJS += xxx.o # gccでビルドするファイル群
   - APPL_CXXOBJS += xxx.o # g++でビルドするファイル群
   - SRCLANG := c | c++ # CのみかC++ありか
   - new_proj/Makefile.app[mod] を直接書き換えてもよい

タスクを追加する(1/2)

- cfgファイルにタスクを生成するための静的APIを追加
  - CRE_TSK(タスクID, { タスク属性，タスクに渡す引数, タスクの関数名, 優先度, スタックサイズ, スタックの先頭番地 })
    - タスクIDはシステムサービスのパラメータとなるマクロ識別子
    - タスク属性は初期状態（TA_ACT：起動，TA_NULL：休止）
    - 優先度は0に近いほど高い優先度となる
    - スタックの先頭番地はNULLを指定するとカーネルが自動的にスタック領域を確保する

app.cfg

```
DOMAIN(TDOM_APP) {
        CRE_TSK(MAIN_TASK, { TA_ACT , 0, main_task,
        8, 1024, NULL });
}
```

タスクを非特権モードで動かすための記述

タスクをカーネルに登録するための記述

タスクを追加する(2/2)

- Cファイルにタスクとして動作する関数を追加

app.c

```
void
main_task(intptr_t exinf)
{
        ...
        ext_tsk();
}
```

cfgファイルに記述したタスクの関数名と同じ名前

cfgファイルに記述したタスクに渡す引数がexinfに渡される

タスクを終了するためのAPI呼出し
※そのままmain_taskからリターンしてもext_tskにジャンプする仕組みになっているが，明示的にext_tskを呼び出すほうが正式