LEGO SPIKE Prime向けソフトウェアプラットフォーム SPIKE-RT の紹介

2022/06/25

名古屋大学

朱　義文

発表内容

- Lego SPIKE Prime
- 既存事例:Pybricks
- SPIKE-RT 概要
- 内部実装

SPIKE-RT

- LEGO SPIKE Prime 向けの RTOS ベースのSPF.
- EV3 の廃版に伴う, EV3RT の 後継プロジェクト.
- 開発中.

LEGO® Education SPIKE™ Prime

小学校高学年から中高生向けの STEAM学習セット
([1]より引用)

ETロボコンでは今年度より使用可能.

画像は, [1] より引用

- SPIKE Prime Hub
  - プログラム可能
  - 6 つ I/O ポート (PUPデバイス接続用)
  - 32 MB の外部記憶領域(フラッシュメモリ)
  - 5x5 LED マトリックス表示器
  - 加速度センサとジャイロスコープセンサ
  - 3 つの制御用ボタン(1つは ライト を含む)
  - スピーカ
  - USB
  - Bluetooth
- PUP(Powered Up) デバイス
  - モータ
  - カラーセンサ
  - 超音波センサ
  - フォースセンサ

[1] https://education.lego.com/ja-jp/products/-spike-/45678#spike%E3%83%97%E3%83%A9%E3%82%A4%E3%83%A0

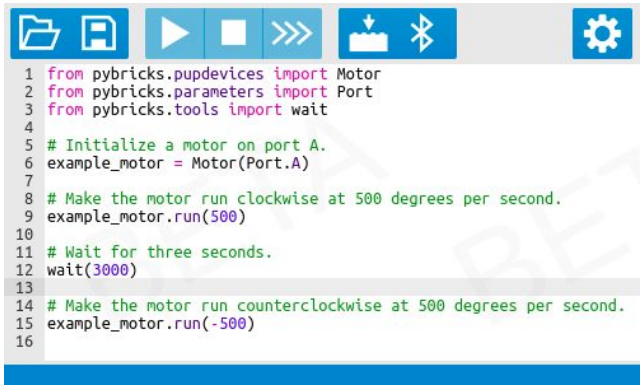
SPIKE Prime Hub の詳細

- SoC：STM32F413VG
  - Arm Cortex-M4
  - 最大 100 MHz
  - RAM : 320 KB
  - ROM : 1 MB （うち32KBはBootloaderが使用）
  - USB DFU による書き込み
- 公式ファームウェア
  - MicroPythonによるプログラミング環境を提供.
  - 公式の仕様書[2]には以下のようにある.
    - > Embedded MicroPython operating system
  - RTOS を載せてないように見える.

[2]
https://education.lego.com/ja-jp/product-resources/spike-prime/%E3%83%80%E3%82%A6%E3%83%B3%E3%83%AD%E3%83%BC%E3%83%89/%E6%8A%80%E8%A1%93%E8%A6%81%E4%BB%B6

Pybricks

- LEGOのコンピュータ（Technic/Boost/City Hub, EV3, SPIKE Prime ）向けOSSのSPF.
- MicroPythonによるプログラミング環境.
- ブラウザからのコーディングし，Bluetooth でロード.
- SPIKE Prime Hub は，まだベータ版.

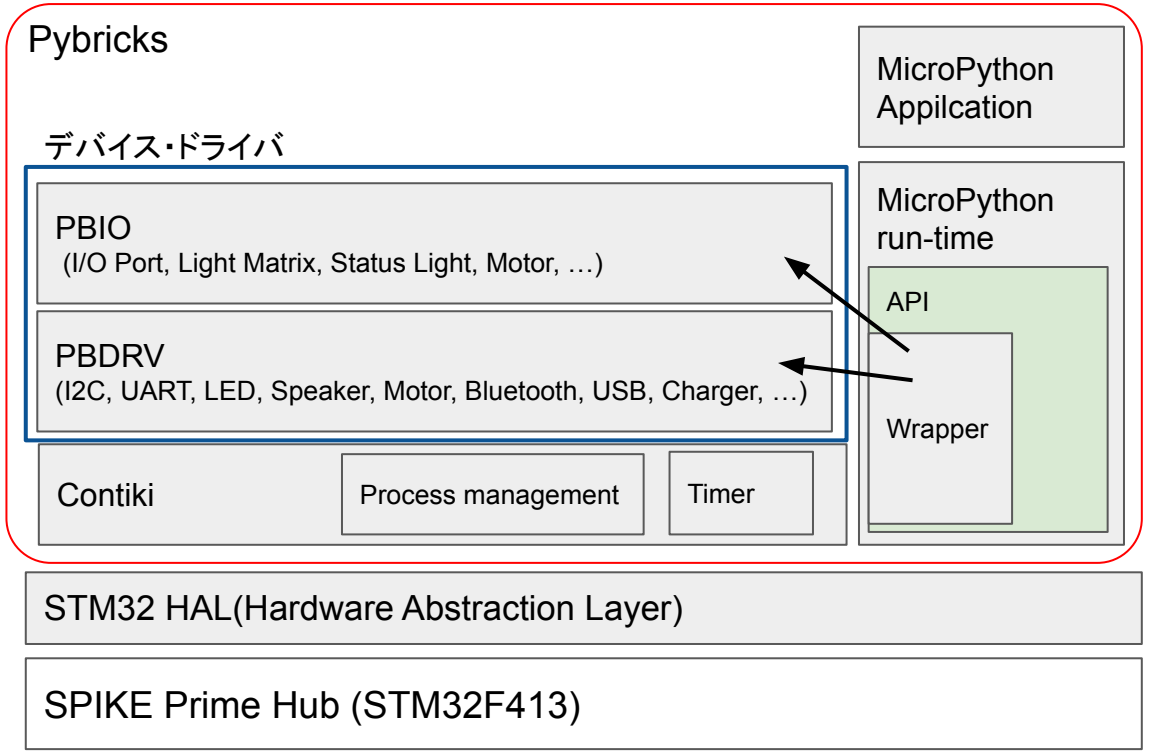


```
1  from pybricks.pupdevices import Motor
2  from pybricks.parameters import Port
3  from pybricks.tools import wait
4
5  # Initialize a motor on port A.
6  example_motor = Motor(Port.A)
7
8  # Make the motor run clockwise at 500 degrees per second.
9  example_motor.run(500)
10
11 # Wait for three seconds.
12 wait(3000)
13
14 # Make the motor run counterclockwise at 500 degrees per second.
15 example_motor.run(-500)
16
```

画像は，[3] より引用

[3] https://pybricks.com/

Pybricks の構成



- Contiki
  マルチタスク機能.
- 10 以上のプロセス
  デバイス・ドライバなどのうち非同期的な処理をプロセスとして実装

※Contikiにおける処理単位をプロセスと呼ぶ

Contiki

- ほぼC 言語のみで実装されたOS.
- 移植性・軽量性に優れる.
- 省電力無線通信を標準で想定．IoT向け.
- マルチタスク機能
  - Contikiにおける処理単位をプロセスと呼ぶ.
  - 協調的マルチタスク（ノンプリエンプティブ・マルチタスク）
    - プロセスがreturn文により自主的に実行権を移譲することを想定.
    - 関数先頭のswitch文とgoto 文の組み合わせにより中断箇所から実行再開.
      - スタック（→局所変数）は保存されない.
  - FCFS(FIFO)スケジューリング
    - 優先度の概念が無い.

- Pybricksでは，MicroPythonランタイムが暇な時などにContikiを明示的に呼び出す.
- RTOS ではない.

[4] https://github.com/contiki-os/contiki

SPIKE-RT

目標：RTOS によるプログラミング環境の提供

特徴

- TOPPERS/ASP3 RTOS
- C 言語プログラミング環境
- MIT ライセンス

API

- ASP3 API
- C 標準ライブラリ（Newlib）
- SPIKE API （デバイス制御API）

SPIKE API（予定）

- ハブ本体の機能
  - マトリックスLED
  - 単一 LED
  - ボタン
  - 加速度センサ/ジャイロスコープ
  - スピーカ
- PUPデバイス
  - カラーセンサ
  - 超音波センサ
  - フォースセンサ
  - モータ

SPIKE API

超音波センサの例

- デバイスポインタを渡してAPIをを実行
- mm 単位

```
pup_device_t *eyes;
int32_t distance;

// ポート番号から制御に用いるデバイスポインタの取得
eyes = pup_ultrasonic_sensor_get_device(PBIO_PORT_ID_A);

// 距離[mm] の取得
distance = pup_ultrasonic_sensor_distance(eyes);

printf("Distance : %d mm\n", distance);
```

SPIKE C API (SPIKE-RT)

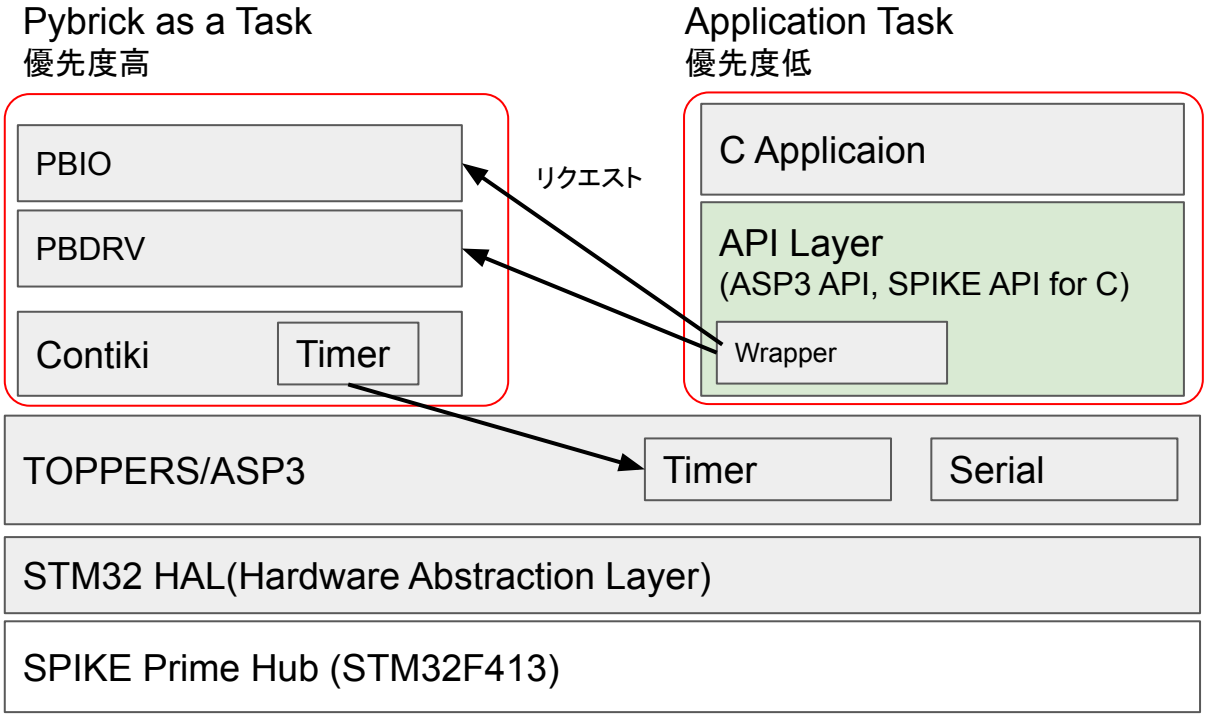
```
int16_t distance;

// 距離[cm] の取得
distance = ev3_ultrasonic_sensor_get_distance(EV3_PORT_1);

printf("Distance: %-3d cm", distance);
```

EV3 C API (EV3RT)

SPIKE-RTの構成


Pybrick as a Task
優先度高
Application Task
優先度低
PBIO
PBDRV
Contiki
Timer
リクエスト
C Applicaion
API Layer
(ASP3 API, SPIKE API for C)
Wrapper
TOPPERS/ASP3
Timer
Serial
STM32 HAL(Hardware Abstraction Layer)
SPIKE Prime Hub (STM32F413)

デバイス・ドライバの実装方針

一般に思いつくアイデア

- 0から実装する.
- Pybricks のデバイス・ドライバを分離し，ASP3のタスクとして移植する.
  - 10 以上のプロセス

→ 実装・維持コストが大きい

デバイス・ドライバの実装方針

一般に思いつくアイデア

- 0から実装する.
- Pybricks のデバイス・ドライバを分離し，ASP3のタスクとして移植する.
  - 10 以上のプロセス

→ 実装・維持コストが大きい

Contiki を移植し, Pybricks 全体 を1つのタスクとして動作させる.

- 実は少ない修正で実現可能.
- Pybricks 本家のバグ修正・機能追加への対応が容易に.

参考：EV3RTでは，Linuxのデバイス・ドライバをそれぞれ移植

Pybricks 全体のASP3タスクへの移植

- Contiki の移植
  ASP3 API に置き換える
  - タイマ機能
    - タイマハンドラを周期ハンドラとして登録
  - Contiki 自体のスリープ・復帰
    - slp_tsk(), wup_tsk() に

- 初期化コードの修正
- メインコードの修正
  - Contiki のみを呼び出す．MicroPython は呼び出さない．

SPIKE API実装

- SPIKE API
  - Pybricks の デバイス・ドライバに対して，リクエストを送る.
- Pybricks におけるMicroPython向けAPI実装を移植.
  ≒ MicroPython Run-time 依存の処理の置き換え.
  - C 言語向けに引数処理を削除.
  - リトライによる待ちの処理をdly_tsk()に置き換え.
- 超音波センサ用APIのコード量
  - PUPデバイスに共通：約220行
  - 超音波センサに固有：約70行
    - 公開用ヘッダファイルを除いた.
  - 容易に実装可能！

Unity によるテスト

組込み/C言語向けテストフレームワーク Unityを用いた SPIKE APIのテスト
コードを修正するたびに全テストを実行

目的

- API の使い方の例示
- 非同期処理などで失敗していないかの最低限の確認

※ 厳密な動作の保証を目的としていない

```
TEST(UltrasonicSensor, distance)
{
  pup_device_t *eyes;
  int32_t distance;

  eyes = pup_ultrasonic_sensor_get_device(
                PBIO_PORT_ID_TEST_ULTRASONIC_SENSOR);
  // eyes != NULL か？
  TEST_ASSERT_NOT_NULL(eyes);

  distance = pup_ultrasonic_sensor_distance(eyes);
  // distance > 0 か？
  TEST_ASSERT_GREATER_THAN(0, distance);
}
```

pup_ultrasonic_sensor_distance() の実際のテストコード

[5] http://www.throwtheswitch.org/unity/

デバイス対応状況

ハブ本体の機能	動作（※1）	API対応
マトリックスLED	O	X
単一 LED	O	X
ボタン	O	X
加速度センサ/ ジャイロスコープ	X	X
スピーカ	△（※2）	X
外部フラッシュメモリ	X	X
USBシリアル	O	X
Bluetooth	O	X

PUPデバイス	動作（※1）	API対応
カラーセンサ	O	△（※3）
超音波センサ	O	O
フォースセンサ	O	X
モータ	O	X

※1 ... Pybricks の関数を呼び出すことで動作する場合を含む
※2 … 動作確認にはしてないが動作することが期待される
※3 … 研究室内の他の学生が実装中

ほとんどが動作確認済み！

現状のファームウェアのサイズ

ROM ： 150KB / 1MB

RAM ： 40KB / 320KB

EV3RTとの比較

	EV3RT	SPIKE-RT
OS	TOPPERS/HRP3	TOPPERS/ASP3
デバドラ	ev3devを参考に移植	Pybricksをタスク化して再利用
API	● HRP3 API ● EV3 API for C/C++ ● Newlib	● ASP3 API ● SPIKE API for C ● Newlib
アプリダウンロード方法	● USBケーブル ● SD ● OTA	USBケーブル（USB DFU）
プラットフォームのテスト	printf, try & error	printf Unity（テストツール）

今後の展開

- 残りの API の実装
- 年内にはコードを公開（？）
- 時期・規模は不明だが，チーム化して開発することを検討中
- 最悪応答性評価によるリアルタイム性の評価

以上