Armadillo-640
開発基礎セミナー

第6部 拡張I/Fを使った開発の流れ（その2）
（アプリケーション編）

株式会社アットマークテクノ
http://www.atmark-techno.com/

目次

Armadilloにおける開発の要素

- **ハードウェア**
  - 拡張I/Fを使った拡張基板 → **GPIO接続のSWとLED**
    I2C接続の加速度センサー
- **カーネル/デバイスツリー**
  - デバイスドライバの適用 → **I2Cドライバの有効化確認**
  - 拡張I/Fの端子機能の設定 → **I2C端子の有効化**
- **アプリケーション**
  - 既存のパッケージのインストール → **C言語ビルド環境**
    Pythonパッケージ
  - 独自のアプリケーションの作成
    → **SWからデータを取得してLEDへ出力するプログラム**
    照度センサーからデータを取得するプログラム

Armadilloにおける開発の要素

- ハードウェア
  - 拡張I/Fを使った拡張基板 → GPIO接続のSWとLED
    I2C接続の加速度センサー
- カーネル/デバイスツリー
  - デバイスドライバの適用 → I2Cドライバの有効化確認
  - 拡張I/Fの端子機能の設定 → I2C端子の有効化
- **アプリケーション**
  - 既存のパッケージのインストール → C言語ビルド環境
    Pythonパッケージ
  - 独自のアプリケーションの作成
    → SWからデータを取得してLEDへ出力するプログラム
    照度センサーからデータを取得するプログラム

アプリケーション

SWの入力でLEDが点灯/消灯するアプリケーション

トグルSWの入力を取得してLEDが点灯/消灯するアプリケーション1

- Armadillo-640のGPIOはGPIOクラスディレクトリ以下のファイルによって制御を行うことが出来ます。
- まずは今回使用するピンのGPIO番号を確認します。

用途	ピン番号	ピン名	GPIO番号
スイッチ	13	GPIO3_IO23	87
LED	28	GPIO4_IO09	105

- GPIO番号が確認出来たら、使用するピンのGPIOクラスディレクトリを作成します。
- GPIOクラスディレクトリは、/sys/class/gpio/export に GPIO番号を書き込むことで作成できます。

```
root@armadillo:~# echo 87 > /sys/class/gpio/export
root@armadillo:~# echo 105 > /sys/class/gpio/export
```

トグルSWの入力を取得してLEDが点灯/消灯するアプリケーション1

- GPIO ディレクトリ以下の direction ファイルへ以下の値を書き込み、入出力方向を指定します。

設定	説明
high	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。 出力レベルは HIGH レベルになります。
out	入出力方向を OUTPUT に設定します。出力レベルの取得/設定を行うことができます。 出力レベルは LOW レベルになります。
Low	out を設定した場合と同じです。
in	入出力方向を INPUT に設定します。 入力レベルの取得を行うことができますが設定はできません。

- 今回、SWに接続する端子を”in”に、LEDに接続する端子を”out”に設定する為、以下のコマンドを実行します。

```
root@armadillo:~# echo in > /sys/class/gpio/gpio87/direction
root@armadillo:~# echo out > /sys/class/gpio/gpio105/direction
```

トグルSWの入力を取得してLEDが点灯/消灯するアプリケーション1

- “sw_led.sh”というファイルを作成します。

```
root@armadillo:~# vi sw_led.sh
```

- 下記のスクリプトを書いて保存します。

```
#!/bin/bash

LED=105
SW=87


while true
do
 data=`cat /sys/class/gpio/gpio$SW/value`
 if [ "$data" -eq "1" ] ; then
  echo "ON"
  echo high > /sys/class/gpio/gpio$LED/direction
  sleep 1
 else
  echo "OFF"
  echo out > /sys/class/gpio/gpio$LED/direction
  sleep 1
 fi
done
```

1秒ごとに、SWの状態を確認し、
ONの場合はLED点灯、
OFFの場合はLEDを消灯させる

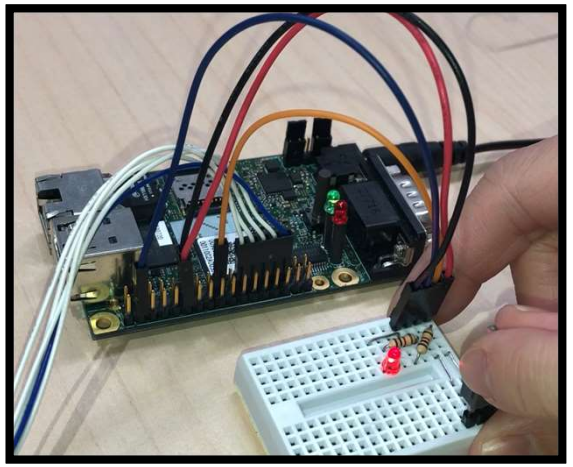
トグルSWの入力を取得してLEDが点灯/消灯するアプリケーション1

- 作成したシェルスクリプトはそのままでは実行権限がありません。
- 下記コマンドを実行して、実行権限を付与します。

```
root@armadillo:~# chmod +x sw_led.sh
```

- sw_led.shを実行すると、コンソールにSWの状態が出力され、ONの場合はLEDが点灯し、OFFの場合はLEDが消灯します。

```
root@armadillo:~# ./sw_led.sh
OFF
OFF
OFF
ON
ON
ON
OFF
```



アプリケーション

照度センサーのデータを取得するアプリケーション

I2C接続の照度センサー(TSL2561)からデータを取得するアプリケーション1

- Linuxではハードウェア制御は、主にデバイスファイル（※）を介して行います。

※）ドライバによっては、クラス等を介して制御するものもあります。

※）既存のライブラリが用意されているものもあります。

- ここでは、サンプルアプリケーション（Python3）で、照度センサー（TSL2561）から、データを取得してみます。

I2C接続の照度センサー(TSL2561)からデータを取得するアプリケーション2

- 下記コマンドを実行して、python3とpython3-pipをインストールします。

```
root@armadillo:~# apt-get install python3
root@armadillo:~# apt-get install python3-pip
```

- pipコマンドを使用して、tsl2561というライブラリをインストールします。

```
root@armadillo:~# pip3 install tsl2561
```

I2C接続の照度センサー(TSL2561)からデータを取得するアプリケーション3

- “lux.py”というファイルを作成します。

```
root@armadillo:~# vi lux.py
```

- 下記の内容のファイルを作成します。

```
tsl2561 import TSL2561
import time

if __name__ == "__main__":
while True:
  tsl = TSL2561(debug=True)
  print(tsl.lux())
  time.sleep(1)
```

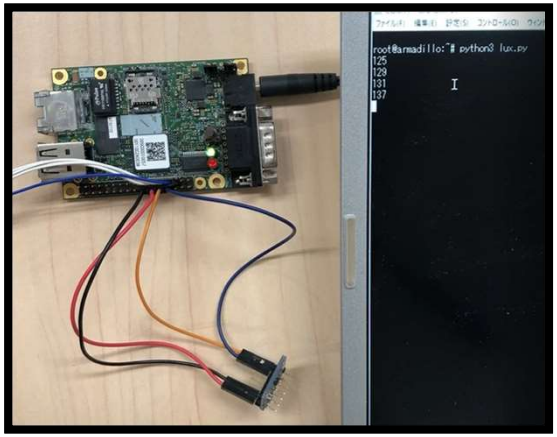
I2C接続の初期化を行い
取得したいデータを指定

I2C接続の照度センサー(TSL2561)からデータを取得するアプリケーション4

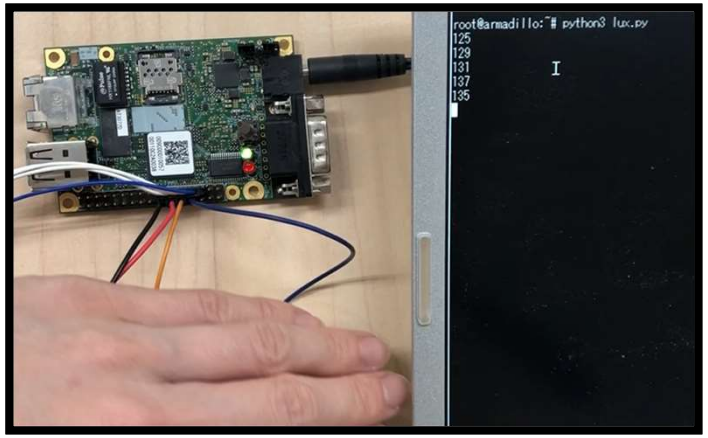
- サンプルアプリケーション（sens_adxl345）を実行すると、照度センサーから取得したデータがコンソールに出力されます。

```
root@armadillo:~# python3 lux.py
127
126
127
30
11
（略）
```

※Ctrl＋Cで終了できます。



手で覆うと
照度が低くなる



参考）アプリケーションの実例

- ここでは、I2C接続の照度センサー（TSL2561）を python3のプログラムで動作させる一例を説明しました。

- 他のデバイスや通信方法、プログラム等については、 Armadillo 実践開発ガイド（第3部）、Howto、ブログ、フォーラム にて、実例の情報が得られます。

参考）アプリケーションの自動起動

- 電源をONにすると、アプリケーションが自動起動する方法は、下記のブログが活用できます。

Armadillo-X1, Armadillo-IoT G3/G3L: systemdでアプリを自動起動する方法
(その１)
https://armadillo.atmark-techno.com/blog/6938/2865
(その２)～起動順序を設定する～
https://armadillo.atmark-techno.com/blog/6938/2898
(その３)～並列処理の実現～
https://armadillo.atmark-techno.com/blog/7370/2919