blinky_ota について

blinky_otaは OTA (Over The Air) でプログラムの更新を行うことができます。

その為には、`idf.py menuconfig`での設定と、1度だけ有線接続での書き込みが必要となります。

詳細は以下を参照ください。

---

1. menuconfig

blinky_otaには`idf.py menuconfig`で設定していただく項目があります。

```
cd blinky_ota
idf.py menuconfig
```

`Blinky Configuration`を選択してください。

ここには、5つの設定項目があります。

全Blinky共通の設定が4つ、個々のBlinky毎に設定する項目が1つあります。

1-1. 全Blinky共通の設定

まずは、全Blinkyで共通の設定項目が4つあります。

- **WiFi SSID**

  使用する2.4GHz帯無線LANルータのSSIDを入力してください。

- **WiFi Password**

  上記SSIDに対応するパスワードを入力してください。

- **HTTP Server IP**

  HTTP Server として使用する PC の IPアドレスを入力してください。

- **HTTP Server Port**

  HTTP Server で使用するポート番号を入力してください。

  特に指定がなければデフォルト設定のままで問題ありません。

1-2. 個々のBlinkyでの設定

個々のBlinkyで設定する項目が1つあります。

- **Individual number of blinky**

  個体番号を入力してください。1以上の整数を想定しています。

2. 有線接続での書き込み

blinky_otaは OTA(Over The Air) に対応したプログラムですが、
最初に 有線で接続して書き込みを行う必要があります。

```
idf.py build
idf.py -p PORT -b 115200 erase_flash flash
```

ここで、`PORT`は`ls /dev/cu*`で出てきたESP32のポートを指定してください。

また、通常の書き込みと異なり、`erase_flash`が必要ですのでご注意ください。

3. 無線接続での書き込み

上記の設定、書き込みが完了していれば、次からはOTAでの書き込みが可能となります。

3-1. ターゲットの設定

書き込みを行うBlinkyを指定する為に、target.txtを編集します。

target.txtはblinky_otaフォルダの直下に配置されています。

target.txtを開くと以下のような構成になっています

```
add
1-50,60,70,80,90
remove
5-10,20,30
```

`add`の次の行が、書き込み対象となる個体番号の指定

`remove`の次の行が、`add`の中から排除する個体番号の指定となります。

個体番号の指定は`1-50`のようにすると1番から50番までを対象とします。

また、`,`で区切ることでいくつでも数字を指定することができます。

上記の例では実際に書き込み対象となる個体番号は

> 1から4, 11から19, 21から29, 31から50, 60, 70, 80, 90

となります。

また、target.txtが存在しない場合には、全部のBlinkyが書き込み対象となります。

3-2. ビルド

プログラムのビルドおよびハッシュ値の作成、
target.txtからターゲットとする個体番号の計算を行います。

```
idf.py app
python hash.py
python constraint.py
```

上記の3つのコマンドをまとめた `build.sh` スクリプトも用意しています。
`build.sh` もblinky_otaフォルダ直下に配置していますので、macでしたら

```
./build.sh
```

を実行していただければ、3つのコマンドが実行されます。

以下、3つのコマンドの簡単な解説となります。

3-2-1. `idf.py app`

blinkyプログラムのビルドを行います。

buildフォルダ直下にblinky_ota.binファイルが作成されます。

このbinファイルが、実際に書き込まれるファイルとなります。

3-2-2. `python hash.py`

build/blinky_ota.binファイルからハッシュ値を計算します。

計算した値はbuildフォルダ直下にhash.txtとして保存されます。

このハッシュ値は、書き込み時にハードウェアが持っているハッシュ値と比較し、同じ値であればbinファイルに更新がなかったと判断し、書き込みを行いません。

3-2-3. `python constraint.py`

target.txtから、書き込み対象の個体番号を計算します。

計算した値はbuildフォルダ直下にconstraint.binとして保存されます。

書き込み時にこのデータを見て、ハードウェアの個体番号がこのデータに含まれていれば書き込みを行います。

また、target.txtが無い場合には、buildフォルダのconstraint.binを削除し、全部のBlinkyを書き込み対象とします。

3-3. HTTP Server

サーバーを起動し、Blinkyデバイスへ無線経由で書き込みを行えるようにします。

pyhon 2系の場合

```
cd build
python -m SimpleHTTPServer 8070
```

python 3系の場合

```
cd build
python -m http.server 8070
```

pcは、***1. menuconfig*** で設定したWiFiへ接続し、ポート番号も設定したHTTP Server Port を指定してください。

3-4. Blinkyデバイスの起動

Blinkyデバイスの電源をオン、またはリセットすると以下の手順で書き込みを試みます。

- **WiFi接続**

  WiFi接続に成功すると、赤色のLEDが点灯します。

  WiFi接続できなかった場合は、Blinkyの動作に切り替わります。

- **サーバー（PC）への接続**

  サーバーへの接続に成功すると、赤色と緑色のLEDが点灯します。

  サーバーへ接続できなかった場合、Blinkyの動作に切り替わります。

  （ここでサーバーに接続できない場合、接続失敗と判断するのに少し時間がかかります。）

- **個体番号のチェック**

  constraint.binから、ターゲットの個体番号リストを取得します。

  取得した番号リスト内に自分の個体番号が含まれていない場合、Blinkyの動作に切り替わります。

- **ハッシュ値の取得**

  ハッシュ値を取得し、Blinkyデバイスに保存されているハッシュ値と比較します。
  もし、ハッシュ値が更新されていれば、デバイスのハッシュ値を更新します。
  更新されていなかった場合、Blinkyの動作に切り替わります。

- **書き込み**

  これまでのハッシュ値、個体番号のチェックを通過したのでデータの更新を試みます。

  blinky_ota.binのダウンロード、書き込みを行います。

  ダウンロード中はBlinkyデバイスのLEDが時計回りに点灯します。

  時計回りにLEDが光っていれば、データの更新が行われていると判断してください。

  無事書き込みが終了すると、プログラムが再起動します。

  再起動後、ハッシュ値の比較が行われ、同じハッシュ値になっているのでBlinkyの動作に切り替わります。