


i

More than 1 year has passed since last update.

Pythonその2 Advent Calendar 2019 Day 2

@moomooya

posted at 2019-12-03 updated at 2020-10-26

ギリギリまでお手軽にMicroPythonでIoTやるための手引書

 Python, Arduino, IoT, micropython, ESP32

この記事はPythonその2 Advent Calendar 2019の2日目です。1日目は@ssh-22さんのre.subを使った高度な文字列置換でした。

Arduino, Raspberry Pi の誕生によって急速にエコシステムが整ってきているIoT界限ですが、Raspberry Pi は Python で実装していけるもののちょっとした電子制御には Arduino の方が圧倒的に手軽に利用できます。

とはいえ Arduino 言語（C/C++もどき）ではなく、使い慣れた Python で書きたいという方もいらっしゃるのではないのでしょうか。本記事では組み込み用 Python である MicroPython を利用した開発を金銭的な意味でも手軽に始める手引を書いていきたいと思います。

用意するもの

- Arduinoではなく ESP32開発ボード
- サンハヤト社 ニューブレッドボード SAD-101

作業環境はmacOS + Python3です。

ESP32開発ボード

最近のArduino界限ではArduinoと同じ開発環境が使えて、Wifi/Bluetoothが使えるESP32が主流になっています（正確にはArduinoではないけど大体同じように使える）。

定番のArduion UNOよりも概ね高機能なのに安いESP32-DevKitCおよび互換品がよく使われています。

急ぐなら1,480円の秋月電子 ESP32-DevKitCとか、2,200円のスイッチサイエンスESPr Developer 32なりを調達しましょう。

ただ個人的にはaliexpressなどの海外通販をおすすめします。今回は436円のこちらを調達しました。お値段1/3です。ただし届くまで3週間位かかりました。

1,500円程度で買えるなら国内で購入してもいいのですが、電子工作は部品を壊したり、なにか作るのに使ったら使い回すのが面倒だったりするので、急がなければ1/3の値段で3つ買っておく方がおすすめです。

海外通販でクレジットカード決済をしたくない方にはpaypalが使えるbanggoodなどのほかサイトも有るのですが、国内購入と価格差が少ないので国内で買ってしまったほうがいいかもしれません。それでもRaspberry Piに比べれば半額以下です。

（追記）と思ったらAmazonで安く売ってますね。2個でも2,000円以下なのでもうこのへんでいいんじゃないかな。

サンハヤト社 ニューブレッドボード SAD-101

ESP32-DevKitCはピン列の間の幅が少し広く、安く売っている両サイド5穴ずつのブレッドボードだと差し込み穴がたりません。

ニューブレッドボードシリーズであれば両サイド6穴ずつなので使いやすいです。今回は一番シンプルな SAD-101 を用意しました。

ちょっと高級なブレッドボードなのでAmazonで518円です。

MicroPythonのインストール

あとは日本語訳されているドキュメントに従ってESP32にMicroPythonをセットアップしていきましょう。

通常ESP32-DevKitC購入時にはArduino CoreというArduino互換ファームウェアが書き込み済みです。

このままではArduino言語でしか実装できないのでMicroPythonのファームウェアを書き込み直します。

ファームウェアのダウンロード

Firmware for ESP32 boards

ESP32用のMicroPythonでは「Wifiが使えるがBluetoothが使えないファーム」「Bluetoothは使えるがWifiが使えないファーム」のどちらかを選択しなければなりません。

今回は「Wifiが使えるファーム」の安定ビルドを選んでみます。

Firmware for ESP32 boards

The following files are daily firmware for ESP32-based boards, with separate firmware for boards with and without external SPIRAM. Non-SPIRAM firmware will work on any board, whereas SPIRAM enabled firmware will only work on boards with 4MiB of external pSRAM.


Program your board using the esptool.py program, found [here](#). If you are putting MicroPython on your board for the first time then you should first erase the entire flash using:

```
esptool.py --chip esp32 --port /dev/ttyUSB0 erase_flash
```

From then on program the firmware starting at address 0x1000:

```
esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 460800 write_flash -z 0x1000  
esp32-20190125-v1.10.bin
```

Firmware built with ESP-IDF v3.x, with support for LAN and PPP but no bluetooth:

- GENERIC : [esp32-idf3-20191202-v1.11-611-g7f24c2977.bin](#)
 - GENERIC : [esp32-idf3-20190529-v1.11.bin](#)
 - GENERIC : [esp32-idf3-20190125-v1.10.bin](#)
 - GENERIC : [esp32-idf3-20180511-v1.9.4.bin](#)
 - GENERIC-SPIRAM : [esp32spiram-idf3-20191202-v1.11-611-g7f24c2977.bin](#)
 - GENERIC-SPIRAM : [esp32spiram-idf3-20190529-v1.11.bin](#)
 - GENERIC-SPIRAM : [esp32spiram-idf3-20190125-v1.10.bin](#)
 - TinyPICO : [tinypico-idf3-20191202-v1.11-611-g7f24c2977.bin](#)
- 

Firmware built with ESP-IDF v4.x, with support for bluetooth but no LAN or PPP:

- GENERIC : [esp32-idf4-20191202-v1.11-611-g7f24c2977.bin](#)
- GENERIC-SPIRAM : [esp32spiram-idf4-20191202-v1.11-611-g7f24c2977.bin](#)
- TinyPICO : [tinypico-idf4-20191202-v1.11-611-g7f24c2977.bin](#)

ドライバのインストール

ESP32-DevKitCにUSB接続するためにSilicon Labsのドライバをインストールします。

CP210x USB - UART ブリッジ VCP ドライバ

今回はmacOS用ドライバをインストールしました。ESP32をUSBで接続すると、おそらく `/dev/cu.SLAB_USBtoUART` にESP32が認識されるはずです。

書き込み済みファームウェアの初期化

```
$ pip install esptool
```

```
$ esptool.py --port /dev/cu.SLAB_USBtoUART erase_flash
```

実行するとこんな結果が出ると思います。



```
$ esptool.py --port /dev/cu.SLAB_USBtoUART erase_flash
esptool.py v2.8
Serial port /dev/cu.SLAB_USBtoUART
Connecting....._
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: XX:XX:XX:XX:XX:XX
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 8.4s
Hard resetting via RTS pin...
```

ファームウェアの書き込み

```
$ esptool.py --chip esp32 --port /dev/cu.SLAB_USBtoUART write_flash -z 0x1000 esp32-idf3-20190529-v1.11.bin
```

実行するとこんな結果が出ると思います。

```
$ esptool.py --chip esp32 --port /dev/cu.SLAB_USBtoUART write_flash -z 0x1000 esp32-idf3-20190529-v1.11.bin
esptool.py v2.8
Serial port /dev/cu.SLAB_USBtoUART
Connecting....._____
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: XX:XX:XX:XX:XX:XX
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 1146864 bytes to 717504...
Wrote 1146864 bytes (717504 compressed) at 0x00001000 in 63.5 seconds (effective 144.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

これで MicroPython のセットアップができました。
早速Pythonで動かしてみようと思います。

Pythonの実行

MicroPython REPLの利用

まずはREPLプロンプトにアクセスしてみます。

```
$ screen /dev/tty.SLAB_USBtoUART 115200
```

おもむろに `help()` `[enter]` と入力して以下のように出力されれば成功です。

```
Welcome to MicroPython on the ESP32!

For generic online docs please visit http://docs.micropython.org/

For access to the hardware use the 'machine' module:

import machine
pin12 = machine.Pin(12, machine.Pin.OUT)
pin12.value(1)
pin13 = machine.Pin(13, machine.Pin.IN, machine.Pin.PULL_UP)
print(pin13.value())
i2c = machine.I2C(scl=machine.Pin(21), sda=machine.Pin(22))
i2c.scan()
i2c.writeto(addr, b'1234')
i2c.readfrom(addr, 4)

Basic WiFi configuration:

import network
sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
sta_if.scan()                                # Scan for available access points
sta_if.connect("<AP_name>", "<password>") # Connect to an AP
sta_if.isconnected()                        # Check for successful connection

Control commands:
CTRL-A      -- on a blank line, enter raw REPL mode
CTRL-B      -- on a blank line, enter normal REPL mode
CTRL-C      -- interrupt a running program
CTRL-D      -- on a blank line, do a soft reset of the board
CTRL-E      -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)
For a list of available modules, type help('modules')
>>>
```

`print()` も返ってきます。

```
>>> print("hello")
hello
```

`screen` コマンドは `ctrl + a + k` で終了確認のプロンプトが出るので `y` で終了します。

ソースコードの転送

REPLでコードを書いていくのは辛いのでソースコードファイルを転送できるようにします。
まずはツールのインストール。

```
$ pip install adafruit-ampy
```

転送済みのファイルを確認します。

```
$ ampy -p /dev/tty.SLAB_USBtoUART ls
/boot.py
```

`boot.py` だけが転送されている事がわかります。

内容も確認してみます。

```
$ $ ampy -p /dev/tty.SLAB_USBtoUART get /boot.py
# This file is executed on every boot (including wake-boot from deepsleep)
#import esp
#esp.osdebug(None)
#import webrepl
#webrepl.start()
```

全行コメントアウトされている事と、 `boot.py` に書かれた内容はボードがリセットされたときの最初の1度だけ実行されるこ
とがわかります。

`boot.py` 実行後に（存在すれば）メイン処理として `main.py` が実行されます。以下のようなコードを `main.py` として転
送してLチカしてみます。

```
import machine
import time

pin13 = machine.Pin(13, machine.Pin.OUT)

while True:
    pin13.on()
    time.sleep_ms(500)
    pin13.off()
    time.sleep_ms(500)
```

転送は以下のコマンドで行います。

```
$ ampy -p /dev/tty.SLAB_USBtoUART put main.py
```

これで500ms間隔でD13ピンに接続されたLEDが点灯すれば成功です。

まとめ

MicroPythonを知った当初はファームウェアの書き換えが必要など、敷居が高いように思えましたが実際にやってみると慣れないツールを使う局面はあるもののそれほど難しいこともなくPythonコードの転送と実行まで実現できました。今回の手順は1度やってしまえばあとはソースコードを転送するだけなのでPythonに慣れた方はESP32をMicroPythonで使うというのもかなりアリな選択肢ではないでしょうか。

今回は行っていないですが、ESP32の通信機能を利用してWifi APへの接続も以下のような感じで簡単に出来るようなので夢が広がります。

```
import network
sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
sta_if.scan() # Scan for available access points
sta_if.connect("<AP_name>", "<password>") # Connect to an AP
sta_if.isconnected() # Check for successful connection
```

そしてなにか面白いものを作ったらぜひIoTLTで発表すると良いと思います。

ちなみに冒頭でも書きましたが届くまで待つことが出来るなら436円でこれだけ遊べるのです。控えめに言って最高じゃないですか？

明日は@TsuMakotoさんのAmazon PersonalizeをPython SDKで入門です。

Register as a new user and use Qiita more conveniently

- 1. You get articles that match your needs
- 2. You can efficiently read back useful information

[What you can do with signing up](#)

Sign up

Login

Comments

No comments

Sign up for free and join this conversation.

Sign Up

If you already have a Qiita account [Login](#)

How developers code is here.

© 2011-2023 Qiita Inc.




Guide & Help

- About
- Terms
- Privacy
- Guideline
- Design Guideline
- Feedback
- Help
- Advertisement

Contents

- Release Note
- Official Event
- Official Column
- Opportunities
- Advent Calendar
- Qiita Award
- API

SNS

-  @Qiita
-  @qiita_milestone
-  @qiitapoi
-  @Qiita

Our service

- Qiita Team
- Qiita Jobs
- Qiita Zine
- Official Shop

Company

- About Us
- Careers
- Qiita Blog