このドキュメンテーションは、MicroPython の最新開発ブランチのためのものです。 リリースバージョンでは利用できない機能に言及することがあります。

特定のリリースのドキュメントをお探しの場合は、左側のドロップダウンメニューを使って、望みのバージョンを選択します。

1. ESP32 での MicroPython の始め方

MicroPython を使うと、ESP32 ボードを最大限に活用することができます。逆も同様、ESP32 チップは MicroPython を使用するための優れたプラットフォームです。このチュートリアルでは、MicroPython の設定、プロンプトの表示、WebREPLの使用、ネットワークへの接続、インターネットとの通信、ハードウェアペリフェラルの使用、およびいくつかの外部コンポーネントの制御について説明します。

始めましょう！

1.1. 必要なもの

最初に必要なのは、ESP32 チップを搭載したボードです。MicroPython ソフトウェアは ESP32 チップ自体をサポートしており、どのボードでも動作するはずです。ボードについて気にするとことは、GPIO ピンが外にどのように接続されているか、UART を PC で使用できるようにする内蔵 USB シリアルコンバータが含まれているかどうかです。

このチュートリアルではピンの名前にチップ名(例: GPIO2)を使っていますが、これが特定のボード上でどのピンに対応するかを見つけるのは簡単なはずです。

1.2. ボードの電源を入れる

ボードに USB コネクタがある場合は、PC に接続されているときに電源が供給されている可能性が非常に高いです。それ以外の場合は、直接電源を入力する必要があります。詳細については、ボードのドキュメントを参照してください。

1.3. ファームウェアの入手

まず、最新の MicroPython ファームウェアの .bin ファイルをダウンロードして、ESP32 デバイスにロードします。ファームウェアは MicroPython ダウンロードページ からダウンロードできます。ここには、3つの主要な選択肢があります:

- 安定ビルド版ファームウェア
- 日次ビルド版ファームウェア
- SPRAM サポート付き毎日ビルド版ファームウェア

MicroPython を使い始めたばかりの方には、安定版ファームウェアビルドをお勧めします。経験豊富で経験豊かな MicroPython ESP32 ユーザーで、開発が続けられている新機能をテストしたい場合は、毎日ビルド版があります。ボードが SPIRAM をサポートしているなら、標準のファームウェアか SPIRAM をサポートしているファームウェアのどちらかを使えます。後者を使うと Python オブジェクトのためにより多くの RAM が使えるようになります。

1.4. ファームウェアの配備

MicroPython ファームウェアを取得したら、ESP32 デバイスにロードする必要があります。これを行うには、主に２つのステップがあります。まず、デバイスをブートローダモードにし、次にファームウェアをコピーする必要があります。これらの手順の正確な手順は、特定のボードに大きく依存します。詳細については、そのドキュメントを参照する必要があります。

幸い、ほとんどのボードは USB コネクタ、USB シリアルコンバータを備え、DTR ピンと RTS ピンが特別な方法で配線されていて、すべてのステップを自動的に行うことができるため、ファームウェアの展開は簡単です。このような機能を備えたボードには、Espressif 社の DevKitC, PICO-KIT, WROVER-KIT dev-kits の他にも Adafruit Feather HUZZAH32, M5Stack, Wemos LOLIN32, TinyPICO ボードがあります。

最良の結果を得るために、新しい MicroPython ファームウェアをインストールする前に、まずデバイスのフラッシュ全体を消去することをお勧めします。

現在のところはファームウェアをコピーする方法として esptool.py のみをサポートしています。このツールは https://github.com/espressif/esptool/ からダウンロードするか、pip を使用してインストールしてください:

```
pip install esptool
```

1.3 で始まるバージョンは、Python 2.7 と Python 3.4 (またはそれより新しいバージョン)の両方をサポートしています。古いバージョンでも正常に動作しますが(少なくとも 1.2.1 が必要)、Python 2.7 が必要です。

esptool.py を使用すると、次のコマンドでフラッシュを消去できます:

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

消去したら、次の方法で新しいファームウェアを導入します:

```
esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000 esp32-20180511-v1.9.4.bin
```

注記:

- PC により、 "port " 指定を別のものに変更する必要があります
- フラッシュ時にエラーが発生した場合はボーレートを下げる必要があります(たとえば 115200 に下げるには、コマンドに `--baud 115200` を追加で指定します)
- 特定の FlashROM 設定を持つボードでは、フラッシュモードを変更する必要があります(たとえばコマンドに `-fm dio` を追加で指定するなど)
- ファームウェアのファイル名もダウンロードしたファイルの名前に変える必要があります

上記のコマンドがエラーなしで実行されれば、MicroPython がボードにインストールされたことになります！

1.5. シリアルプロンプト

デバイスにファームウェアをインストールしたら、ボードに応じて UART 0 (GPIO1 = TX, GPIO3 = RX)上の REPL (Python プロンプト)にアクセスできます(USB シリアルコンバータに接続されている可能性があります)。ボーレートは 115200 です。

ここまでできたならば、後は ESP8266 チュートリアルにしたがって進めることができます。これら２つの Espressif 社チップは、MicroPython を使用することに関しては非常に似ているからです。ESP8266 チュートリアルは ESP8266用 MicroPythonチュートリアル にあります(「ESP8266でのMicroPythonの使い方」の章はスキップしてください)。

1.6. インストールの問題のトラブルシューティング

フラッシュ中またはファームウェアの実行直後に問題が発生した場合は、次の推奨事項を参照してください:

- ハードウェア上の問題がないかを調べ、取り除いてください。発生しやすい問題は２つあります。電源品質が悪いことと FlashROM の寿命/欠陥です。電源と言えば、素のアンペア数だけでなく、一般的に低リップルやノイズ/EMIも重要です。最も信頼性が高く便利な電源はUSBポートです。
- 上記のフラッシュ手順は、速度と安定性の間の良好な妥協点である 460800 ボーのフラッシュ速度を使っています。ただし、モジュール/ボード、USB-UARTコンバータ、ケーブル、ホスト OS などによっては、上記ボーレートでも速すぎてエラーにつながる可能性があります。そのような場合には、より一般的な 115200 ボーレートを試してください。
- フラッシュの内容の不正(たとえば、チップ上の欠陥セクタなど)を検出するには、 上記のコマンドに `--verify` オプションを加えてください。
- ファームウェアをフラッシュしても問題が解決しない場合、問題を報告するために esptool.py プロジェクトページ https://github.com/espressif/esptool にある情報やバグトラッカーを参照してください。
- ファームウェアをフラッシュできた場合でも、 `--verify` オプションが何度試してもエラーを返す場合は、欠陥のある FlashROM チップである可能性があります。