1

はじめての MicroPython

この章では MicroPython の概要と最もシンプルな使い方を説明します。

1.1 MicroPython の概要

MicroPython（マイクロパイソン）は、マイクロコンピューターや組み込み機器で使うことを目的に作成された Python のサブセットです。

◆ MicroPython の特徴

MicroPython には、次のような特徴があります。

- MicroPython は、広く普及している汎用プログラミング言語である Python のサブセットです。MicroPython は Python と高い互換性があり、MicroPython のほとんどの部分は Python と同じなので、Python の豊富な情報や多くのライブラリなどの多くを使うことができます（Python のものすべてを使えるわけではありません）。
- MicroPython は効率的に実装されていて、リソースが十分でない環境でも快適に動作するように最適化されています。MicroPython は、わずか 256k のコードスペースと 16k の RAM で動作するほどコンパクトです。
- MicroPython はインタープリタ言語なので、言語の習得や実行が容易です。また、プロトタイプを容易に作成することができます。
- MicroPython はマイクロコンピューターや組み込み機器のハードウェアに近い制御を容易に行うことができます。
- MicroPython ではプログラマーは原則としてメモリー管理を意識する必要がありません。一方、マイコンや組み込み機器の制御などに良く使われるアセンブリ言語（アセンブラ）や C 言語、C++ などでは、プログラマーがメモリーを管理しなければなりません。
- MicroPython のプログラムを実行するために、OS を必要としません。マイコンに MicroPython をサポートするファームウェアーをインストールするだけで、MicroPython のプログラムを実行することができます。なお、Windows や Linux のような OS 上で実行することも可能です。

◆ MicroPython の実行環境

MicroPython は、コンパクトな電子回路基板から PC まで、さまざまな場所で実行することができます。MicroPython のプログラムを実行できる固有プラットフォームとして公式ドキュメン

トには次のようなものが紹介されています。

- pyboard
- ESP8266
- ESP32
- Raspberry Pi RP2xxx
- NXP i.MXRT 10xx
- WiPy/CC3200
- UNIX
- Windows
- Zephyr
- Renesas RA
- SAMD21/SAMD51

MicroPython のプログラムは、PC からマイコン（マイクロコントローラや組み込みシステム）にケーブルで接続して簡単に転送し、マイコンで実行できます。

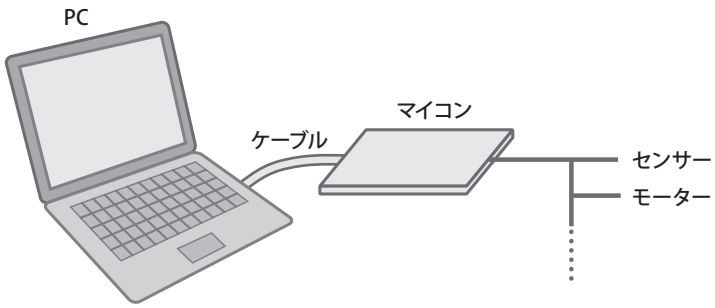

図1.1●開発環境とマイコンの接続

◆ Web エミュレーター

MicroPython のプログラムを実行できる固有プラットフォームのリストの先頭には、micropython.org が提供する pyboard があります。pyboard を含むマルチプラットフォームの Web エミュレーターとして micropython.org から unicorn が次のサイトで提供されています。

https://micropython.org/unicorn/

このエミュレーターは、PyBoard と、ボードに搭載されている LED、およびそれに接続された LED やサーボモータなどもエミュレートすることができます。

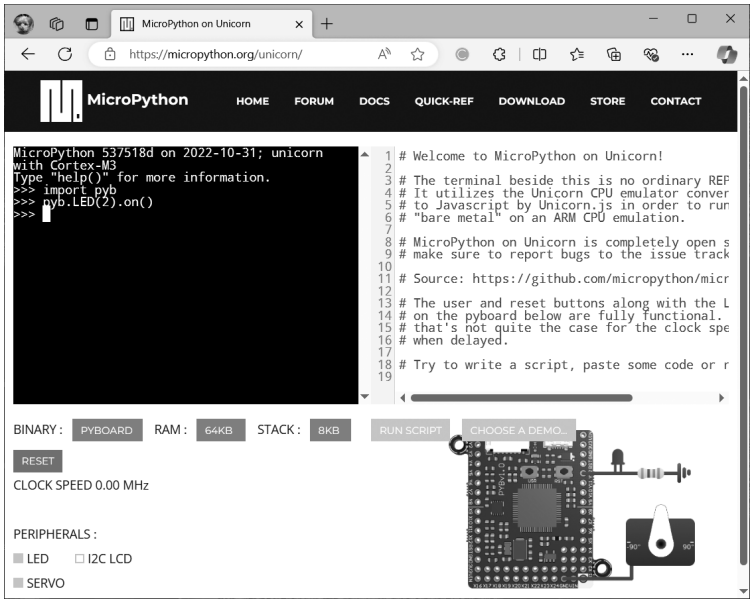

図1.2●Webエミュレーター

このエミュレーターを使って MicroPython の多くのプログラムを実行できます（すべてのプログラムを実行できるわけではありません）。

ページの左上のフィールドには、対話的にプログラムを実行する領域があります。ここに表示されているプロンプト「>>>」に対して MicroPython のコードを入力すると、そのコードを実行することができます。

また、右上のフィールドにプログラムコードを入力して、［RUN SCRIPT］を実行することでスクリプト（一連の MicroPython プログラムコード）を実行することができます（ここには最初は # で始まるコメントが入力されていますが、コメントなので削除してかまいません）。

ページの左下には、LED やサーボモータ（SERVO）などを表示するかどうかを指示するチェックボックスが表示されています。

ページの右下には pyboard の図が表示されており、チェックボックスをチェックした場合は LED やサーボモータ（SERVO）などの状態が表示されます。

Web エミュレーターはデモ用途のものなので、すべての機能をサポートしているわけではありません。たとえば、入力、ファイル入出力、マルチスレッドなどのプログラムは意図したように実行できない場合があります。

◆ Raspberry Pi Pico と Pico W

本書では、MicroPython のプログラムを実行するプラットフォームとして、Raspberry Pi（ラズベリー・パイ）Pico や WiFi 機能を備えた Raspberry Pi Pico W（以下 Pico/Pico W）を使う方法も説明します。

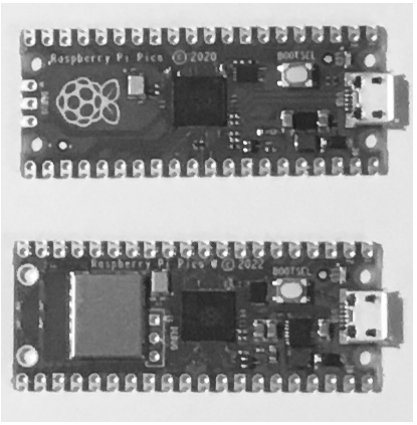

図1.3●Raspberry Pi Pico（上）とPico W（下）

実際の装置でプログラムが動くところを見てみたい場合には Pico/Pico W を入手して試してみるのも良いでしょう。

◆ Linux 上の micropython

Linux に micropython をインストールすると、MicroPython のプログラムの一部を実行することができます。本書の多くはプログラミング言語としてのMicroPythonに焦点を当てているので、本書の多くのコードを Linux 上の micropython で実行することができます。

2.2 節「文字と名前」の「文字」でも説明しますが、環境によっては日本語がサポートされていません。日本語を使うとエラーになる場合や、日本語の表示が意図通りにできない環境では、英数文字を使ってください。

1.2　MicroPython のライブラリ

MicroPython では、次の 2 種類のモジュールが提供されます。

- Python 標準ライブラリのサブセット組込みモジュール
- MicroPython 固有のモジュール (byp、machine、bluetooth など)

なお、特定のシステム上の MicroPython の特定のビルドでは、リソースやその他の制限のため、一部の機能（モジュール全体、クラスや機能）が使えない可能性があります。

特定の環境で利用可能なモジュールは、help('modules') で調べることができる場合があります。次の例は pyboard のエミュレーターで「help('modules')」を実行した例です。

```
>>> help('modules')
__main__          gc                pyb               ustruct
builtins          machine           uarray            usys
cmath             math              ucollections      utime
framebuf          micropython       uio
Plus any modules on the filesystem
>>>
```

この例の場合、たとえば thread や _thread はサポートされていないのでマルチスレッドで並列処理を行うことはできません。

◆ Python 標準ライブラリのサブセットモジュール

Python 標準ライブラリは、Python モジュールにある機能のサブセットを実装します。また、MicroPython 特有の拡張機能を提供しているものもあります（array、os）。

表1.1●Python 標準ライブラリに準拠するモジュール

モジュール	説明
array	数値データの配列
asyncio	非同期 I/O スケジューラ
binascii	バイナリ /ASCII 変換
builtins	組込みの関数と例外
cmath	複素数のための数学関数
collections	コレクションとコンテナのデータ型
errno	システムエラーコード
gc	ガベージコレクションの制御
gzip	gzip 圧縮器 & 展開器
hashlib	ハッシュ化アルゴリズム
heapq	ヒープキューアルゴリズム
io	入出力ストリーム
json	JSON のエンコードとデコード
math	数学関数
os	基本的な「オペレーティングシステム」サービス
platform	基盤プラットフォームの識別データへのアクセス
random	乱数の生成
re	簡素な正規表現
select	一連のストリームのイベント待機
socket	ソケットモジュール
ssl	SSL/TLS モジュール
struct	基本データ型のパックとアンパック
sys	システム固有関数
time	時間関連の関数
zlib	zlib 圧縮器 & 展開器
_thread	マルチスレッドサポート

◆ MicroPython 固有のライブラリ

MicroPython 処理系に固有の機能を提供するライブラリとして、例えば以下のような機能があります。

表1.2●MicroPython 固有のモジュール

モジュール	説明
bluetooth	低レベル Bluetooth
btree	単純な BTree データベース
cryptolib	暗号化アルゴリズム
deflate	DEFLATE 圧縮と展開
framebuf	フレームバッファの操作
machine	ハードウェア関連の関数
micropython	MicroPython 内部のアクセスと制御
neopixel	WS2812 / NeoPixel LED の制御
network	ネットワーク設定
openamp	標準非対称マルチプロセッシング (AMP) の提供
uctypes	構造化手法でのバイナリデータアクセス
vfs	仮想ファイルシステム制御

さらに、マイコン固有のライブラリを使うことができる場合があります。
たとえば、pyboard の場合、次のようなモジュールを使うことができます。

表1.3●pyboard固有のモジュール

モジュール	説明
pyb	pyboard 関連の関数
stm	STM32 MCU に固有の機能
lcd160cr	LCD160CR ディスプレイの制御

特定のボード（プラットフォーム）で使用できるライブラリについては、その ボードのドキュメントを参照してください。

1.3　MicroPython の使い方

ここでは MicroPython を使い始めるために必要なことを説明します。

最初に簡単なプログラムを動かしてみて、MicroPython のプログラムとは何かということを理解しましょう。MicroPython の実行環境については付録 B「実行環境」も参照してください。

◆ REPL の起動

対話的に MicroPython のプログラムを実行するために、REPL（Read Evaluate Print Loop）のプロンプトを表示する必要あります。いずれかの環境で REPL のプロンプト「>>>」を表示してください。

unicorn を使う場合は、ウェブブラウザで、次のアドレスを開きます。

```
https://micropython.org/unicorn/
```

エミュレーターが表示されます。

ページの左上のフィールドにプロンプト「>>>」が表示されて、MicroPython のコードを入力して実行することができるようになります。

unicorn 以外の環境で実行したいときには、付録 B「実行環境」を参照してください。

MicroPython が起動すると、MicroPython のメッセージと REPL のプロンプト「>>>」が表示されます。これが MicroPython のインタープリタのプロンプトです。

```
MicroPython 537518d on 2022-10-31; unicorn with Cortex-M3
Type "help()" for more information.
>>>
```

これは unicorn で MicroPython 537518d の場合の例です。バージョン番号やそのあとの情報は、この例と違っていても構いません。

Raspberry Pi Pico または Raspberry Pi Pico W と、Thonny を使っている場合は、付録 B「実行環境」に従って Thonny を起動すると Thonny の左下のシェルペインにたとえば次のように表示されます（バージョン番号や日付情報などは、この例と違っていても構いません）。

```
MicroPython v1.23.0 on 2024-06-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

また、たとえば、Linux で micropython を起動したなら、次のように表示されるでしょう（バージョン番号や日付情報などは、この例と違っていても構いません）。

```
$ micropython
MicroPython v1.19 on 2022-06-23; linux [GCC 9.4.0] version
Use Ctrl-D to exit, Ctrl-E for paste mode
>>>
```

いずれにしても、MicroPython の REPL のプロンプト「>>>」が表示されれば、MicroPython のインタープリタが起動したことがわかります。

◆ REPL

プロンプト「>>>」が表示されている環境を **REPL** といい、MicroPython の命令や式などを読み込んで、その結果が必要に応じて出力されます。このようなコードを読み込んでは実行するシステムをインタープリタといいます。プロンプト「>>>」が表示されている環境をインタラクティブシェル（対話型シェル）ともいいます。

インタープリタは「解釈して実行するもの」という意味、インタラクティブシェルは「対話型でユーザーからの操作を受け付けて結果や情報を表示するもの」という意味があります。

OS 上で MicroPython を使っているときには、OS（コマンドウィンドウ、ターミナルウィンドウなど）のプロンプトである「>」や「#」、「$」などと、MicroPython のインタープリタを起動すると表示される REPL のプロンプト「>>>」を使います。この 2 種類のプロンプトは役割が異なるので区別してください。