第1章 MicroPython高速化の基本原理を体感

マイコン用Python高速化の方法

井田 健太

ここでは，マイコンでの定番Python環境MicroPythonにC言語モジュールを追加して，処理性能UP方法の研究を行ってみます（**写真1**）．行列乗算処理が50～100倍速くなりました．

マイコン用定番実装 MicroPythonとは

● よく定番マイコンの標準装備になっている

組み込みマイコン環境向けのPython実装にMicroPythonがあります．Python 3.4の文法と一部の標準ライブラリ，さらにPython3.5で導入されたType Hintsなどの一部の新しいバージョンの機能を実装しています．

もともとはSTM32F4（STマイクロエレクトロニクス）を使ったPyboardというマイコン・ボード向けの実装でした．現在ではnRF51，ESP8266，ESP32，PIC24といったマイコンから，UNIX，Windows，ZephyrなどのOS向けの実装が標準で用意されていて，マイコンで定番のPython環境になっています．

文法としてはPython3.4と同等ですが，リソースの制約が大きいマイコン向けということで，幾つか挙動が違うところや制限があります．このため，標準のPython実装であるCPythonの3.4で動作するプログラムをそのまま動作させられるわけではありません．

MicroPythonの便利なところ

● その1：少ない記述で表現できる

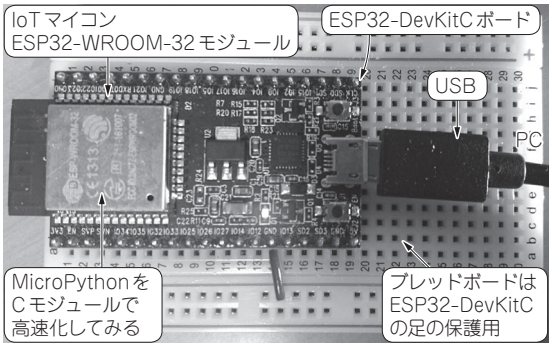
C/C++言語と比較して記述量を減らせる場合が多いです．

● その2：対話型の環境が用意されている

MicroPythonでは，スクリプトを実行する他に，対話型の環境でMicroPythonのプログラムの入力と実行と結果の表示を行えます．この環境をREPL（Read Eval Print Loop）環境と言います．

REPL環境を使えば，例えばI²C接続のセンサの接続確認を手動で行って，どうやって使うのかをある程度把握しつつ，実際に動かすためのスクリプトを記述するといったことが可能で，実験のときに非常に便利です．

また，スクリプト実行中にエラーが発生して実行が停止した場合，自動的にREPL環境に入ります．REPL環境では直前までスクリプトを実行していた状態が保持されているため，変数の内容を表示するなどしてエラーの原因を調べることができます．



（a）定番IoTマイコンESP32ボード

（b）自作Cモジュールで行列乗算100倍高速化

写真1　ここでは定番IoTマイコンESP32のMicroPythonにC言語モジュールを追加して50～100倍高速処理に