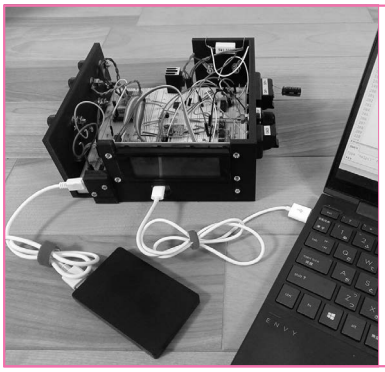

第3章 カリカリ高速チューンMicroPython×My実験ステーション自動計測システム

ラズパイPico制御！自動計測プログラムの制作

加藤 忠 Tadashi Kato

第1章，第2章で作成した測定・実験用のアナログ回路は，マイコンの制御がなければ機能しません．現代のシステムは，フロントエンドに最小限のアナログ回路を置き，大多数の処理をディジタル信号処理で行うのが主流になっています．

もはやマイコン（またはFPGA）は避けて通れません．本章では，初心者におすすめのMicroPythonによるマイコン開発手法について紹介します．自作した測定器を，ラズパイPico（以降Pico）とMicroPythonで制御していきます．また，回路とソフトウェアと連携した実用的な応用測定を紹介します．

マイコン・プログラミングにはMicroPythonがオススメ

● C言語によるマイコン開発のデメリット

マイコンをはじめるに当たって，最初の関門となるのがプログラミング言語のC言語です．現代的なプログラミング言語に比べると，独特なポインタの概念，面倒なメモリ管理など，難解で機能面でも貧弱です．

次に立ちはだかる壁は，開発環境の複雑さです．C言語ではソースコードをコンパイルする必要があります．開発環境の構築やアップデートの管理，コンパイルのため複雑な設定ファイルの記述は，初心者にとって大きな障壁です．マイコンごとに異なるSDK（Software Development Kit）を理解し，それを使ってようやくマイコン操作ができるようになります．

● 対応マイコンが増えてきている「MicroPython」

マイコンの性能が飛躍的に向上したことで，Pythonのような処理負荷の高い実行環境もマイコンにも搭載できるようになりました．クラウドファンディングから始まったMicroPythonの開発は，当初STM32F4マイコンに向けに進められていましたが，次第に対応マイコンが増え，ESP32，nRF，ラズベリー・パイPico（以下Pico）でも利用できるようになりました．今回作成した実験ステーションではPicoを使用しています．

パソコン版のPythonに比べて一部機能が制限されているものの，使い勝手はほとんど変わりません．Python言語であり，習得も容易でしょう．以下のような利点により，開発効率の向上が期待されます．まさに初心者に（上級者にも）うってつけです．

- 非コンパイル言語なので，開発環境の構築が不要
- ペリフェラル[注1]制御用のライブラリが同梱されている
- 異なるマイコン間でも，ソースコード・レベルではほぼ互換性がある
- 対話型プロンプト（REPL）[注2]で1行ずつ即時実行できる
- スクリプト単位での実行や，マイコン起動時の自動実行が可能

● 万能じゃない？ MicroPythonの弱点

以下の2点は，C言語開発に比べて大きく劣ります．

①コンパイル済みのマシン語ではないため，実行速度が大きく劣ります．大量の演算処理をリアルタイムでこなすような，CPU負荷の高いアプリケーションには不向きです．

②ペリフェラル制御用のライブラリは，汎用的な機能しか実装されていません．

● MicroPythonの弱点を克服するテクニック

今回のような測定器制御では，ペリフェラルとの高速信号伝達や，高度なペリフェラル機能を駆使する必要があります．たとえば，アナログ電圧測定器のオシロスコープ機能では，数百kSpsの高速データをサンプリングします．これは，前述の弱点にとってまさに逆風です．

しかしご安心ください．これらの弱点を克服するための「3大マル秘テクニック」があります．これらを

注1：ペリフェラルとは，I²CやSPIなどのマイコンの周辺機能のこと
注2：REPL（Read - Eval - Print Loop）とは，入力したコードをすぐに実行・確認できる開発スタイルのこと