IoT 機器向けスクリプト言語の消費電力に関する評価

2020SC074 佐竹笠輔
指導教員：本田晋也

1　はじめに

IoT 技術の発展とともに，様々な機器がネットワークに接続するようになっている．これにより，多くの利用者の利便性向上やコストの削減，データ活用等による新たなビジネスの機会創出等が行われている．IoT 技術の例として，スマートロッカーやコネクテッドカー，スマートシティ，スマート工場等がある [1].

これまでの組込みシステムは，C 言語で開発されていることが一般的である．IoT システムは，WEB 系のシステムと連携することから，開発も組込みシステムの知識が少ない WEB 系の技術者が開発できることが要求される．その為，WEB 系の開発で広く用いられている Python を使用できると開発を容易にすることができる．

組込みシステム開発で Python が使える環境として，MicroPython がある．MicroPython を使用することで，Python の高い生産性とハードウェアの抽象化によるポータビリティで開発効率の向上や IoT 技術開発者の増加が期待できる．Python はインタープリタ方式であるため，一般的に実行時間がコンパイル方式よりもかかる．

バッテリー駆動による IoT システムでは，消費電力を抑えることが要求される．消費電力量を最小限に抑えることで，駆動時間を伸ばすことができる．例えば，センサネットワークで 1 時間に 1 回センサー値をクラウドに送る機器の場合について．消費電力を抑えるためには，マイクロコントローラーの各種低消費電力機能をプログラムから制御する必要がある [2].

現状の課題として，C 言語で低消費電力機能を実現することは可能であるが，MicroPython により同様の機能を実現可能か不明である．そこで，本研究では，MicroPython と C 言語のそれぞれのライブラリを使用した場合の消費電力の制御結果を比較し，消費電力が削減可能か評価することにする．また，上記で示した MicroPyrton の優位性が C 言語とどれくらい差があるか検討する．

2　背景技術

2.1　MicroPython

MicroPython はスクリプト言語の一種である．インタープリター型であり，実行結果を迅速に確認できコードの変更等が容易である．しかし，コンパイル型に比べて処理速度が遅くなる．上記のメリットが大きく活用され，開発効率の向上や開発領域の拡大に貢献している．MicroPython は Python3 と高い互換性を持つスクリプト言語で，マイクロコンピュータ上での動作を最適化されたものである．

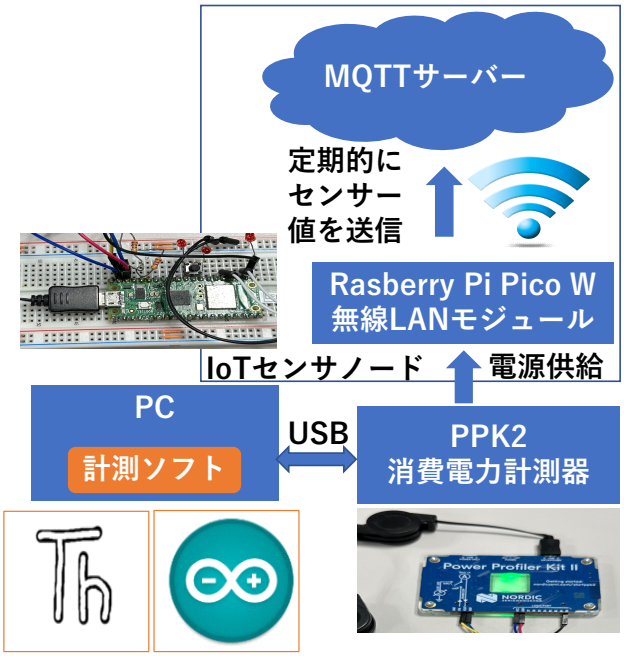

図 1　評価環境の構成図

2.2　Raspberry Pi Pico W

Rasberry Pi Pico W（以下，Pico W）は，センサーデータの取得や IoT デバイス，ロボティクスなど 電子工作に特化したマイクロコントーラである．C 言語及び MicroPython でプログラムを記述し，実行できる．C 言語向けには SDK（Pico C SDK）が提供されている．また，PicoW は，無線チップを搭載し WLAN 機能を使用でき，IoT デバイスの試作や開発に適している．

Pico W では，低消費電力の機能を実現することができる．C 言語と Micropython それれぞれで使用可能な機能を表 2 に示す．低消費電力の機能として，動作周波数の変更・動作電圧の変更・スリープモード・Wi-Fi の OFF 等が存在する．動作周波数は，125MHz が標準となっている．しかし，クロック周波数を 20MHz まで下げることが可能である．動作電圧は，C 言語の場合のみ変更可能である．動作電圧は，1.1V，1.05V，1V，0.95V から選択できる．スリープモードは，sleep，deepsleep，lightsleep の 3 種類が用意されている．deepsleep は，リセットが必要なため再開までに時間を要する．Wi-Fi を OFF にすることで，Wi-Fi を ON にしていた時と比較すると消費電力を抑えることができる．

3　評価環境

IoT 通信を模擬するシステム（IoT センサーノード）を C 言語と MicroPython で実現し，それらに対して各種の低消費電力機構を適用する．

表 1　C 言語と Micropython の低消費電力の機能比較

低消費電力の機能	C 言語	MicroPython
動作周波数の変更	○	○
動作電圧の変更	○	×
スリープモード	○	○
Wi-Fi OFF	○	○
UART の無効化	○	○
SPI の無効化	○	○
I2C の無効化	○	○

3.1　評価システム構成

評価システムの構成図を図 1 に示す．C 言語の場合は，pico-sdk ライブラリ，MicroPython の場合は Thonny によりプログラムを開発する．pico-sdk ライブラリのバージョンは 1.5.1 を使用した．MicroPython のバージョンは，1.21.0 を使用した．IoT センサーノードは，無線通信（Wi-Fi）を使用し Pico W から 5 分間に 1 度，1byte のデータを MQTT サーバーに送信するシステムである．

3.2　計測環境

消費電力は PPK2 で計測する．PPK2 は動的消費電流を計測する機器である．サンプリング速度は 100kS/s であり，計測の分解能は 0.2µA である．

3.3　評価システムの実現

Pico W で動作する C 言語と MicroPython で IoT センサーノードをそれぞれ実現した．電源投入後，WLAN を ON にする．WLAN がアクセスポイントに接続後，MQTT サーバーに対してダミーセンサー値を送信する．その後，Wi-Fi を OFF にし，C 言語は interval 関数，MicroPython は通常の sleep 命令, deepsleep 命令, lightsleep 命令のいずれかで指定時間まで待機する．本研究では、外部センサーは使わないものとする．

4　評価

C 言語環境の IoT センサーノードの計測結果を図 2 に示す．0 は待機フェーズ，1 は Wi-Fi 初期化・MQTT 接続フェーズ，2 は Wi-FiOFF 待機フェーズ，a はセンサー値送信フェーズの消費電流を示す．図 2 は，sleep モードでかつ表 1 の機能を 1 つも使わなかった場合の計測結果である．表 2 に，消費電力量を示す．(MicroPython-1) や (C 言語-1) は，通常の sleep モードで表 1 の低消費電力機能を使用しない場合の計測結果である．(MicroPython-2) や (C 言語-2) は低消費電力モードでかつ表 1 の機能をすべて使用する場合の計測結果である．通常の sleep では，C 言語は MicroPython に比べて消費電力量は小さくなった．表 1 低消費電力機能をすべて使用し，deepsleep モードにすることで MicroPython の消費電力量は一番小さくなった．

次に，IoT センサーノードの電池による駆動時間を評価した．評価を実施するにあたり，C 言語の通常の sleep モードと dormant モード，MicroPython で通常の sleep モードと deepsleep モードについて計測を実施した．駆動

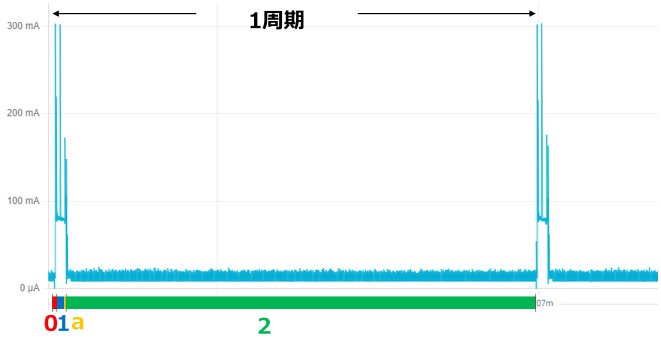

図 2　C 言語の消費電流の計測結果.

表 2　C 言語と Micropython の消費電力量

項目	休止状態	動作周波数 [MHz]	消費電力量 [Wh]
MicroPython-1	通常 sleep	125	46.6
C 言語-1	通常 sleep	125	13.2
MicroPython-2	deepsleep	62.5	1.8
C 言語-2	dormant	62.5	2.3

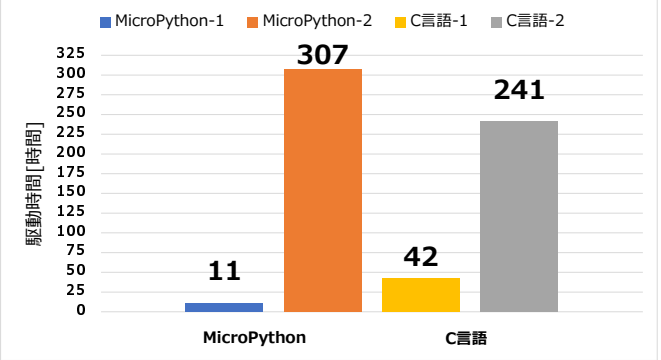

図 3　IoT センサーノードの電池による駆動時間

時間を図 3 に示す．C 言語環境では，MicroPython 環境に比べて駆動時間は大幅に長いことが分かった．しかし，(MicroPython-2) や (C 言語-2) は，Micropython 環境で C 言語環境よりも駆動時間が長くなることが分かった．

5　まとめ

本研究では，Pico W を用いた IoT システムの言語及び低消費電力機能のよる消費電力量の違いについて比較評価した．表 1 の低消費電力機能の最適な組み合わせで実装することで，MicroPython は C 言語よりも消費電力を抑えることができることが明らかになった．今後の課題は，IoT システムを Bluetooth での通信までにかかる時間や処理速度の効率性の実施を考えている．

参考文献

[1] 坂中靖志：電子情報通信学会-IEICE 会誌, Society 5.0 における IoT の役割, pp. 379, May. 2019, https://www.journal.ieice.org/bin/pdf_link.php?fname=k102_5_378&lang=J&year=2019.

[2] Peter Marwedel：Embedded System Design, 7.4.1 Dynamic Voltage and Frequency Scaling (DVFS), 7.4.2 Dynamic Power Management (DPM), pp. 373-377, 2021, https://library.oapen.org/handle/20.500.12657/46817.