Vol.2019-SLDM-189 No.5
Vol.2019-EMB52 No.5
2019/11/13

Raspberry Pi における OpenCV プログラムの消費電力測定

角田和正[1]　谷口一徹[2]　冨山宏之[1]

概要：Raspberry Pi は，教育用途だけでなく，IoT デバイスのプロセッサとしても広く利用されている．IoT デバイスは，低い消費電力で動作することが要求される．本研究では，OpenCV プログラムを対象として，Raspberry Pi の実行時間と消費エネルギーを測定する．OpenCV の実行方式や Raspberry Pi の動作周波数を様々に変更し，実行時間と消費エネルギーの測定と解析を行う．

キーワード：Raspberry Pi，OpenCV，消費エネルギー

1. はじめに

近年，組込みシステムや IoT デバイスにおいて，コンピュータビジョンなどの計算負荷の高い処理が実行されることが多い．そのため，組込みシステムのさらなる性能の向上が求められている．それと同時に，低消費電力化／低消費エネルギー化も求められている．Raspberry Pi は，教育用コンピュータとしてだけでなく，組込みシステムや IoT デバイスのプロセッサとしても広く実用化されている．本論文は，Raspberry Pi 上でコンピュータビジョン・プログラムを実行した際の性能と消費電力／エネルギーについて扱う．

Raspberry Pi の消費電力を測定あるいは解析する過去の研究として，文献[1][2][3][4]などがある．文献[1]では，Raspberry Pi の消費電力を，PC やスマートフォンなどと比較している．文献[2]では，センサネットワークのノードとして Raspberry Pi を使用し，その消費電力を測定している．文献[3]では，ワイヤレスセンサネットワークのゲートウェイとして Raspberry Pi を使用し，その CPU やメモリの周波数を変えながら，消費電力を測定している．文献[4]では，無限ループのプログラムにより CPU に負荷を与え，消費電力を測定し，回帰分析により消費電力モデルを構築している．

本研究では，コンピュータビジョン処理を対象として，Raspbery Pi の実行時間と消費エネルギーを測定する．具体的には，コンピュータビジョンのライブラリ OpenCV を用いて書かれた円検出，Kmeans，人検出の 3 種類のプログラムを，Raspberry Pi 上で実行する．OpenCV はマルチコア CPU 上での並列処理をサポートしており，OpenCV をビルドする際にカスタマイズすることができる．本研究では，OpenMP による並列実行，pthread による並列実行，および，Single-thread 実行の 3 種類を試す．さらに，CPU の動作周波数を 400MHz から 1.2GHz まで変化させる．上述のように，OpenCV や Raspberry Pi のコンフィギュレーションを変更し，実行時間と消費エネルギーを測定，解析する．

本論文の構成は以下の通りである．まず 2 章で実験環境を説明する．3 章で，CPU の動作周波数を変化させた際の実行時間と消費エネルギーを測定する．4 章では，複数の OpenCV プログラムを実行させ，実行時間と消費エネルギーを測定する．5 章で考察を行い，6 章でまとめを行う．

2. 実験環境

2.1 機材とソフトウェア

本研究では Raspberry Pi 3 Model B を使用した．Raspberry Pi 3 Model B には，最大 1.2GHz で動作する ARM Cortex-A53 が 4 コアと 1GB の SDRAM が搭載されている．消費エネルギーの測定には，KKmoon 社の電圧電流テスタ UM24C を用いた．

使用した OpenCV のバージョンは 3.4 であり，

- OpenMP
- pthread
- Single-thread

の 3 種類のコンフィギュレーションでビルドした．ビルドには，GCC 7.4 を用いた．OpenCV のサンプルプログラム集から，以下のプログラムを実行した．

- ハフ変換による円検出
- Kmeans 法によるクラスタリング
- Haar-like 特徴を利用した分類器カスケードによる顔検出

Raspberry Pi 上では Ubuntu 18.04 を動作させ，Governor を Performance に設定することにより，CPU クロック周波数を固定した．周波数は，400MHz，800MHz，1.2GHz の 3 種類を試した．

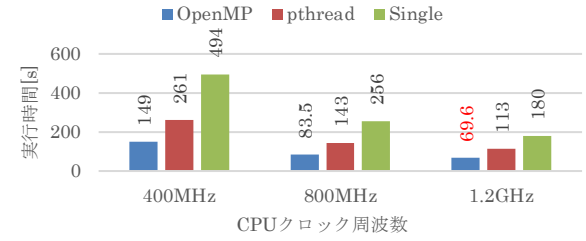
2.2 プロファイリング

予備実験として，3 つのプログラムのプロファイリングを行った．CPU 周波数を 1.2GHz に固定し，3 種類の方法（OpenMP，pthread，Single-thread）でビルドされた OpenCV

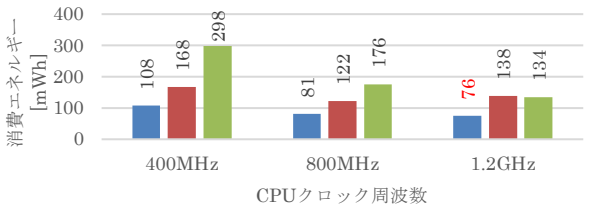
1 立命館大学
Ritsumeikan University
2 大阪大学
Osaka University

表 1. Perf によるプロファイリング結果

	円検出			Kmeans			人検出		
	OpenMP	pthread	Single	OpenMP	pthread	Single	OpenMP	pthread	Single
CPU 時間（秒）	261.3	414.2	182.8	60.3	71.8	15.9	175.6	140.2	199.9
実行サイクル数（M）	305,493	438,507	217,119	72,345	80,104	19,007	206,476	152,051	235,254
実行命令数（M）	104,567	159,691	99,285	49,065	54,771	13,475	116,385	86,133	141,737
IPC	0.34	0.36	0.46	0.68	0.68	0.71	0.56	0.57	0.60
分岐数	12,887	19,553	11,967	5,772	5,475	1,352	6,359	4,622	7,556
分岐予測ミス率	0.44	0.30	0.40	0.78	2.53	2.54	1.76	1.26	1.24
ページフォールト	27,526	26,813	15,206	35,755	111,566	28,578	851,240	504,667	508,497
コンテキストスイッチ	2,036	2,684	15,177	754	1,286	1,376	66,703	12,102	18,502
CPU マイグレーション	10	95	27	8	94	4	17,441	1,726	176

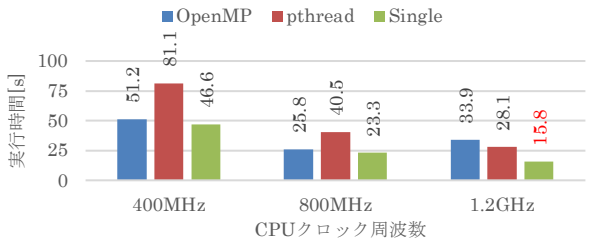


(a) 実行時間

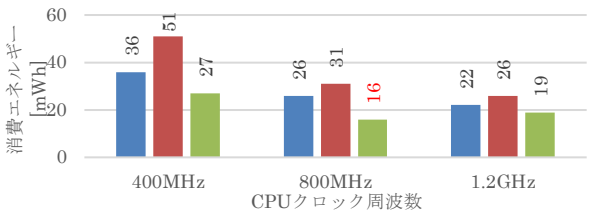


(b) 消費エネルギー

図 1. 円検出プログラム



(a) 実行時間



(b) 消費エネルギー

図 2. Kmeans プログラム

ライブラリを用いて，3 つのプログラムを Raspberry Pi 上で実行した．その際，perf コマンドにより，CPU のパフォーマンスカウンタの値を取得した．

プロファイリング結果を表 1 に示す．実験で使用した Raspberry Pi 3B は 4 コアを有しているため，表中の CPU 時間と，実際の経過時間は大きく異なる．粗い近似ではあるが，表中の CPU 時間をコア数 4 で割ると，経過時間を得ることができる．表 1 を見ると，人検出プログラムのページフォールト数が，他のプログラムと比較して非常に多い．人検出は，ワーキングセットが大きいことが分かる．

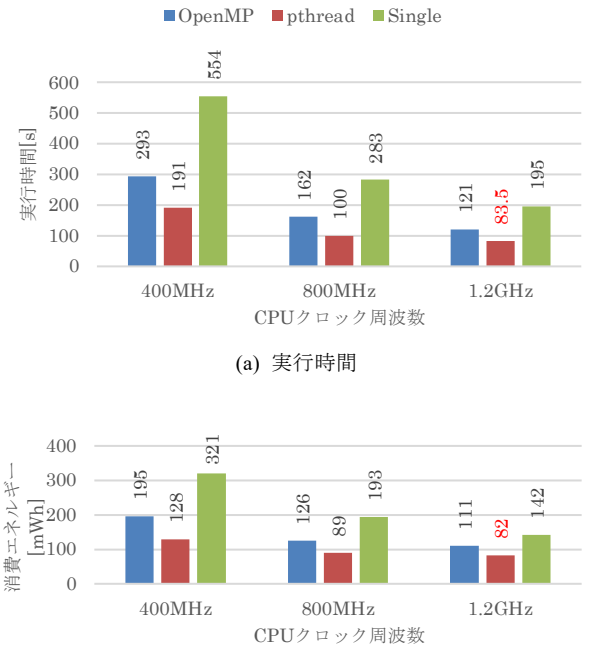
3. 実行時間と消費エネルギーの結果

3 つの OpenCV プログラムを，動作周波数を変更させて Raspberry Pi 上で実行した．

円検出プログラムの結果を図 1 に示す．図 1 (a)が実行時間を表し，図 1 (b)が消費エネルギーを表している．実行時間，消費エネルギーともに，クロック周波数が 1.2GHz で，OpenMP により並列実行した場合が最良となっている．pthread 実行の実行時間は Single-thread 実行より短いものの，コア数ほどの効果が得られていない．その結果，pthread 実行の消費エネルギーは，Single-thread 実行の場合よりも悪くなっている．これは，今回使用した Raspberry Pi と OS の環境では，CPU コア単位での電力制御が行われておらず，アイドル状態の CPU コアも少なからぬ電力を消費しているためであると考えられる．

Kmeans プログラムの結果を図 2 に示す．実行時間は Single-thread 実行の 1.2GHz が最短，消費エネルギーは Single-thread 実行の 800MHz が最小となっている．実行時に与えたデータセットが比較的小さく，OpenMP や pthread による並列化の恩恵を得られていない．図 2 (b)の Signle-

Vol.2019-SLDM-189 No.5
Vol.2019-EMB52 No.5
2019/11/13



(a) 実行時間

(b) 消費エネルギー

図 3. 人検出プログラム

thread 実行において, 800MHz のときの消費エネルギーが, 1.2GHz のときの消費エネルギーよりも小さくなっている. 周波数の低下に伴う内部電源電圧の低下が, 消費電力および消費エネルギーの削減に寄与していると考えられる.

図 3 に人検出プログラムの結果を示す. 今回は, 1.2GHz における pthread 実行が, 実行時間と消費エネルギーの両面で最良となった.

4. 複数のプログラムを実行したときの結果

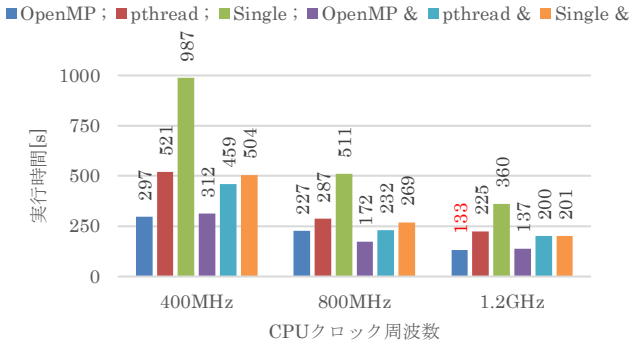
3 章の実験では, 各 OpenCV プログラムを単体で実行した. 本章では, 各 OpenCV プログラムを複数回実行する実験を行う. その際, 以下の 2 通りの実行を行う.

- 同一プログラムを複数回逐次的に実行する
  (bash 上で, セミコロン";"により並べて書いて実行)
- 複数の同一プログラムを同時に実行する
  (bash 上で, アンド記号"&"により並べて書いて実行)

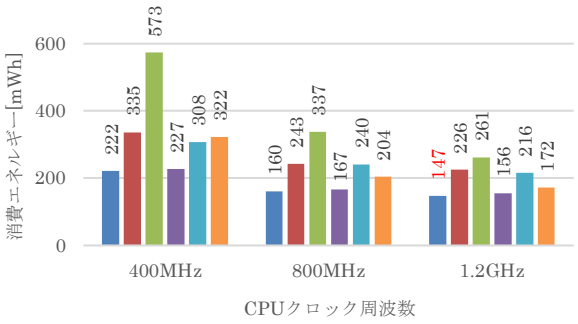
4.1 2 つのプログラムの実行

図 4 に, 2 つの円検出プログラムを実行したときの実行時間と消費エネルギーを示す. 単体で実行した時 (図 1) と同様, 1.2GHz で OpenMP 実行したときが, 実行時間と消費エネルギーの両面で最良だった. 逐次実行と同時実行を比較した場合, 逐次実行の方が僅かに優れていた.

図 5 に, 2 つの Kmeans プログラムを実行したときの結果を示す. 単体で実行した (図 2 参照) と同様, 1.2GHz で Single-thread 実行したときが, 実行時間と消費エネルギーの両面で最良だった. Signle-thread 実行の場合は, 2 つのプログラムを逐次的に実行するよりも, 同時に実行したほうが優れている. 2 つのプログラムを同時に実行することに

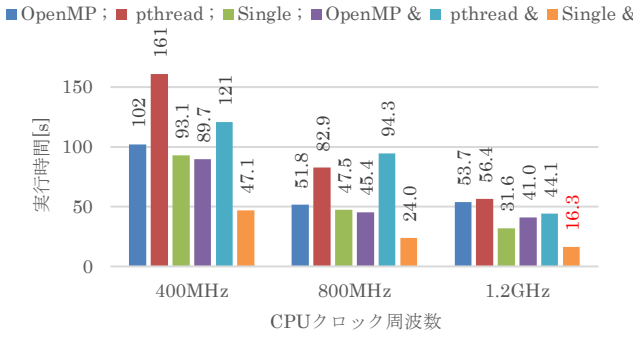


(a) 実行時間

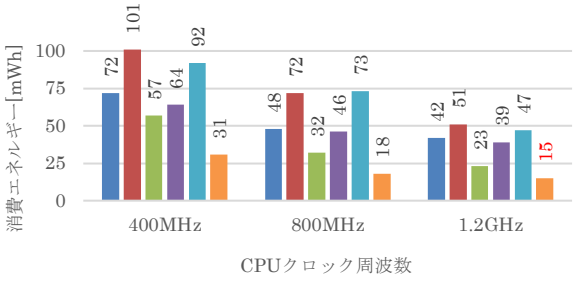


(b) 消費エネルギー

図 4. 2 つの円検出プログラム

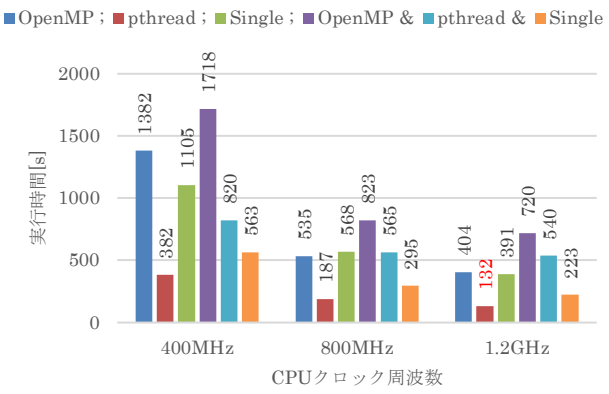


(a) 実行時間

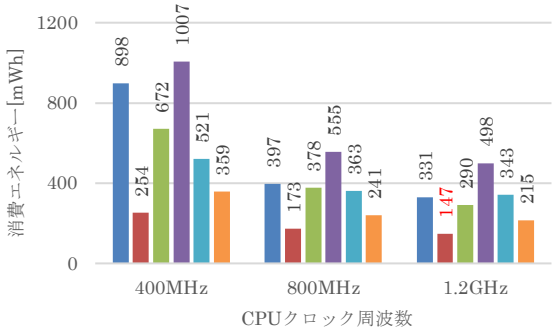


(b) 消費エネルギー

図 5. 2 つの Kmeans プログラム

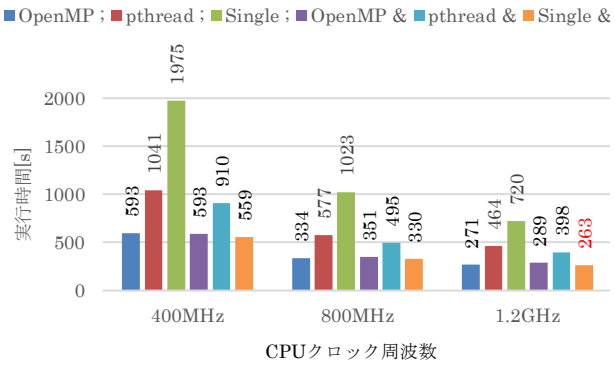


(a) 実行時間

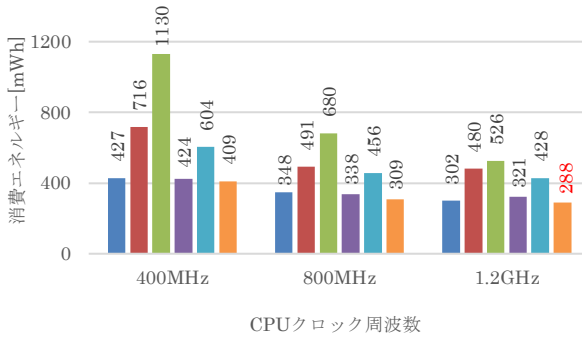


(b) 消費エネルギー

図 6. 2 つの人検出プログラム



(a) 実行時間



(b) 消費エネルギー

図 7. 4 つの円検出プログラム

より，マルチコアによる並列処理が行われ，実行時間が短縮する．

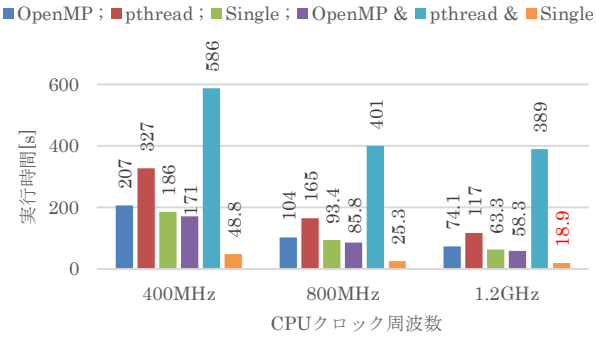
図 6 に，2 つの人検出プログラムを実行したときの結果を示す．1.2GHz で pthread プログラムを逐次実行したときが，実行時間と消費エネルギーの両面で最良となった．

4.2 4 つのプログラムの実行

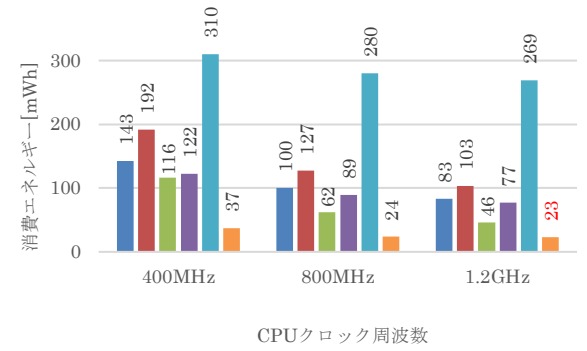
本節では，4 つのプログラムを逐次的に，または，同時に実行した実験を示す．

図 7 に，4 つの円検出プログラムを実行したときの実行時間と消費エネルギーを示す．2 つの円検出プログラムを実行したとき（図 4 参照）は 1.2GHz で OpenMP プログラムを逐次的に実行した場合が最良だったが，今回は，1.2GHz で Single-thread プログラムを同時に実行した場合が，実行時間と消費エネルギーの両面で最良となった．実験で使用した Raspberry Pi は 4 つのコアを有しているため，4 つの Single-thread プログラムが 4 つのコア上で並列に実行されている．

図 8 に，4 つの Kmeans プログラムを実行したときの結果を示す．円検出プログラムの場合と同様，1.2GHz で Single-thread プログラムを同時に実行した場合が，実行時間と消費エネルギーの両面で最良となった．一方，pthread プログラムを同時に実行した場合に，実行時間と消費エネルギーが大幅に悪化している．



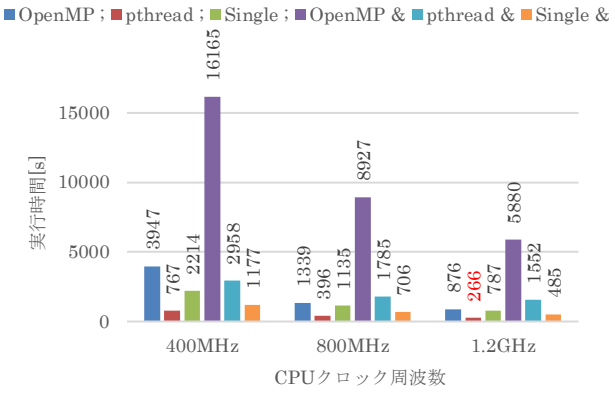
(a) 実行時間



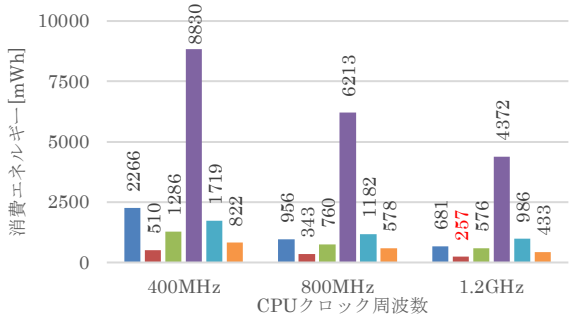
(b) 消費エネルギー

図 8. 4 つの Kmeans プログラム

Vol.2019-SLDM-189 No.5
Vol.2019-EMB52 No.5
2019/11/13



(a) 実行時間



(b) 消費エネルギー

図 9. 4 つの人検出プログラム

図 9 に，4 つの人検出プログラムを実行したときの結果を示す．2 つの人検出プログラムを実行したとき（図 8 参照）と同様，1.2GHz で pthread プログラムを逐次実行したときが, 実行時間と消費エネルギーの両面で最良となった．一方，OpenMP プログラムを同時に実行すると，実行時間が大幅に悪化した．2.2 節の表 1 で説明した通り，人検出プログラムは多くのメモリを使用する．そのため，複数の人検出プログラムを同時に，かつ，各プログラムを OpenMP でデータ並列に実行すると，メモリが不足し，ページフォールトが頻発していると考えられる．

5. 考察

OpenCV は，OpenMP，pthread，CUDA など，複数の並列ライブラリをサポートしており，ビルドする際に選択することができる．本実験では，GPU は使用せず，OpenMP によるマルチコア並列実行，pthread によるマルチコア並列実行, および, シングルスレッド実行を試行した. その結果，実行する OpenCV プログラムによって，あるいは，プログラムに与えるデータによって，実行時間が最短となる並列ライブラリが異なった．また，同時に実行されるプログラムの数や特性によっても，最適な並列ライブラリが異なった．さらに，複数の OpenCV プログラムを動作させる際には，同時並行的に実行させることが必ずしも最適とは限らず，逐次的に実行させることが良い場合も存在した．以上のことから，あらかじめ異なる並列ライブラリを有効化してビルドされた OpenCV ライブラリを複数用意しておき，OpenCV プログラム（アプリケーション）に応じてリンクする OpenCV ライブラリを使い分けることが有効だと考えられる．さらに，OpenCV プログラムをビルド（コンパイル）する際にも，異なる OpenCV ライブラリを用いて複数のバイナリを生成し，実行時に使い分けることが有効であると考えられる．

実行時間と消費エネルギーは，多くの場合で，同じ傾向を示した．つまり，実行時間の短縮が，消費エネルギーの削減につながる．動作周波数に関しては，ほとんどの場合で, 最高動作周波数（今回は 1.2GHz）で動作させることが, 実行時間と消費エネルギーの両面で最良であった．

6. おわりに

本論文は，Raspberry Pi 上で OpenCV プログラムを実行させたときの実行時間と消費エネルギーを評価した．OpenCV をビルドする際の並列ライブラリ，動作周波数，動作させるプログラム数などを変化させた．実験の結果，OpenCV プログラムや実行時の状況に応じて，OpenCV の並列ライブラリを切り替えることの有効性が示唆された．

今後は，OpenCV の各関数について，その特性と，適した並列ライブラリとの相関を詳しく調査する予定である．

謝辞

本研究は一部，キオクシア株式会社（旧社名 東芝メモリ株式会社）の支援による．

参考文献

[1] G. Bekarro, A. Santokhee, “Power Consumption of the RaspberryPi: A Comparative Analysis,” *International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies*, 2016.

[2] C. Cabaccan, F. Reidj, G. Cruz, “Power Characterization of Raspberry Pi Agricultural Sensor Nodes Using Arduino Based Voltmeter,” *International Conference on Computer and Communication Systems*, 2018.

[3] F. Astudillo-Salinas, D. Barrera-Salamea, “Minimizing the Power Consumption in Raspberry Pi to Use as a Remote WSN Gateway,” *Latin-American Conference on Communications*, 2016.

[4] F. Kaup, P. Gottschling, D. Hausheer, “PowerPi: Measuring and Modeling the Power Consumption of the Raspberry Pi,” *Conference on Local Computer Networks*, 2014.