Raspberry Pi を用いた環境モニタリングシステムとその長寿命化

藤澤 功[1] Erick Chandra[2] 出口利憲[3] 田島孝治[3]

概要： 本稿では、環境モニタリングシステムを Raspberry Pi と呼ばれるシングルボードコンピュータを用いて実現した結果について述べる。この方法により、測定対象のセンサデータだけではなく画像も取得可能である。しかし、電力の供給が困難な場所では、測定を行うデバイスはバッテリー駆動が必要である。そのような場合は消費電力の問題で長期運用は困難である。そこで、本稿ではデバイスを間欠動作させるハードウェアを試作し解決を図った。結果、一日 16 回モニタリングを行う場合、モバイルバッテリーで 3 ヶ月以上動作可能であることが明らかになった。

A Trial Product of Environment Monitoring System with Raspberry Pi and To Achieve the Long Lifetime.

FUJISAWA TSUTOMU[1] Erick Chandra[2]
DEGUCHI TOSHINORI[3] TAJIMA KOJI[3]

1. 背景

近年、世界の自然災害による被害額は増加傾向にあり、有効な防災対策は各国の重要課題となっている。日本において最も被害が大きいのは、不定期に発生する大地震であるが、世界の自然災害の被害件数・被害額が最も大きいのは風水害である[1]。風水害とは主に、洪水害・高潮害・土砂災害・風害などを指す。

風水害が発生することを事前に察知する仕組みとして、カメラやセンサを用いて災害が発生しやすい場所を監視する手法がある。日本において、大きな河川や土砂災害が起こりうる場所においては、対象を監視するデバイスが国・自治体によって設置されている。一方で、個人であっても任意の場所に監視デバイスを設置することが可能なサービスも存在する。デバイスの多くは組み込み機器として設計され、運用に月額数千～数万円、設置コストは数万～数十万円と様々である。しかし、土砂災害が予想される危険箇所は日本全土に 50 万箇所も存在し[2]、全てを網羅するには監視デバイスの単価・運用コストを下げる必要がある。また、設置場所によっては電源供給が困難である場合もあり、バッテリー駆動を求められる場合もある。

一方、”Internet of Things”(IoT)と呼ばれる概念に注目が集まっている。この概念は、インターネットに多様かつ多数の物が接続されて、それらの物から送信され、又はそれらの物に送信される大量の情報の活用に関する技術[3]と定義される。この流行によりインターネットに様々な物を接続するための環境が整備されつつあり、個人においても多様で、安価な IoT デバイスを製作することが可能になりつつある。

2. 目的

本稿では、Raspberry Pi による環境モニタリングシステムを構築した結果について述べる。さらに、このシステムの長期運用、長寿命化を行うためのハードウェアを開発する。この環境モニタリングシステムは、電源供給が不可能な場合を想定する。このため、モバイルバッテリーを電源として、間欠的に測定を行うように設計する。具体的には、数ヶ月単位で動作し続けられるようにすることを目標とする。同時に、モニタリングシステム全体のコストをなるべく抑えるようにする。

システムは測定したデータ（各センサの値と写真）を管理するソフトウェアと、実際に測定を行うハードウェアによって構成する。本システムにおけるハードウェアをモニタリングデバイスと呼ぶ。

今回は、モニタリングデバイスの中心として Raspberry Pi を用いる。この理由は次の二つである。

(A) Linux を搭載するシングルボードコンピュータであることから、他の組み込み機器と同様な機能を実現可能であり、拡張性に優れている。

(B) 安価であることから個人で製作・設置するモニタリングデバイスとして適している。

しかし、バッテリーで長期運用する場合は稼働可能時間が課題となる。5V 印加時の Raspberry Pi の消費電流は数百 mA であり、数十 Wh 程度のモバイルバッテリーでは数十

1 先端融合開発専攻、岐阜工業高等専門学校専攻科
2 Institut Teknologi Bandung, Sekolah Teknik Elektro dan Informatika (STEI)
3 電気情報工学科、岐阜工業高等専門学校

Copyright (c) 2017 by the Information Processing Society of Japan

時間しか稼働できない。よって、5V 印加時に消費電流が数十 μA~数 mA である PIC と電子回路により Raspberry Pi の電源を制御する。具体的には、測定後に電源を完全に落とし、一定時間後に再び起動する間欠動作を実現可能なハードウェアを開発することで解決する。

3. システム概要

本稿で開発するシステム概要を図 1 に示す。このシステムは、Raspberry Pi と Arduino によって対象となる環境の状態を測定・撮影し、そのデータをインターネット上のクラウドストレージやクラウドデータベースに保存する。その後、インターネット上のサーバがその保存したデータを取得し、Web サイトとして環境の状態をユーザに提供する。

情報を提供するソフトウェアの特徴を以下に示す。

(1) 現在のモニタリングデバイスの状態を確認可能である。

(2) 取得した環境のデータをグラフ化し、視覚的に分かりやすく情報を提供できる。

(3) 撮影した写真を、日付や時間ごとに確認可能である。

測定するモニタリングデバイスの特徴を以下に示す。

(A) Arduino にセンサ類を接続し、環境の状態を測定する。

(B) Raspberry Pi Camera により、周囲の環境を撮影する。

(C) Raspberry Pi をインターネットに接続し、Arduino から送られてきた測定データや撮影データをインターネット上のクラウドストレージやクラウドデータベースに保存する。

(D) Raspberry Pi の電源を PIC で制御し、間欠動作を行うことで、消費電力を低減する。

(E) 間欠動作のスリープ中であっても、外部入力により強制的に起動し、データを収集する。

(F) 他のモニタリングデバイスと比較して小型であり、個人でも設置可能な価格である。

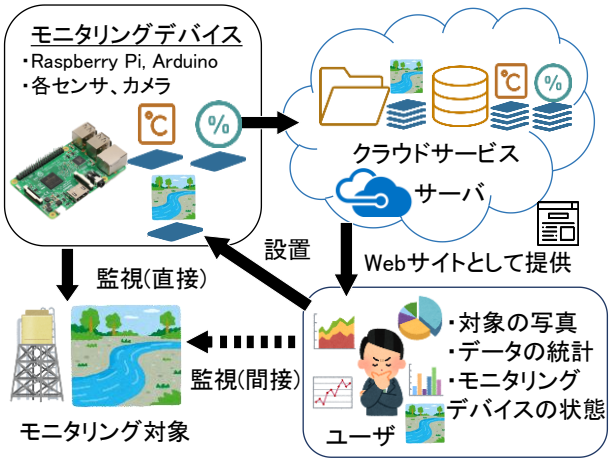

図 1. システム概要

4. 設計

4.1 データ管理用システム

データ管理用システムは、収集したデータを閲覧可能な Web ベースのシステムである。モニタリングデバイスから送信されたデータのうち、写真はクラウドストレージに、各センサによる測定値はクラウドデータベースに保存する。加えて Web サーバを構築し、ここで専用のプログラムを動作させる。ユーザからサーバにアクセスがあると、このプログラムがクラウドサービスにアクセスし、データを取得する。その後、ユーザ視覚的に分かりやすい Web サイトとして表示する。また、現在のモニタリングデバイスの状態もこのプログラムを用いて確認可能とする。

4.2 モニタリングデバイス

環境の状態を測定することが可能なデバイスとする。デバイスのハードウェア構成を図 2 に示す。モニタリングデバイスは主に三つのデバイスから構成される。一つ目は、環境の状態を各センサで取得した後、Raspberry Pi に送信する Arduino である。二つ目は、写真を撮影し、その写真と Arduino から送られてきたデータをクラウドサービスに保存する Raspberry Pi である。三つ目は、Raspberry Pi と Arduino の電源をリレー素子によって制御する PIC とそれに付随する電子回路である。これによって制御される Raspberry Pi の電源は、大容量のモバイルバッテリーである。この回路を電源制御回路と呼び、Raspberry Pi に載せることが可能である。一方、Arduino 及びセンサ類を配置した基盤は、Raspberry Pi とは独立させている。

4.3 モニタリングサイクル

モニタリングデバイスは、以下のサイクルで動作する。

(A) Raspberry Pi を起動する。

(B) Arduino とカメラによりデータを収集する。

(C) クラウドサービスにデータを送信する。

(D) Raspberry Pi を停止する。

この(A)~(D)を 1 サイクルとし、モニタリングサイクルと呼ぶ。このモニタリングサイクルの実行を電源制御回路で制御する。

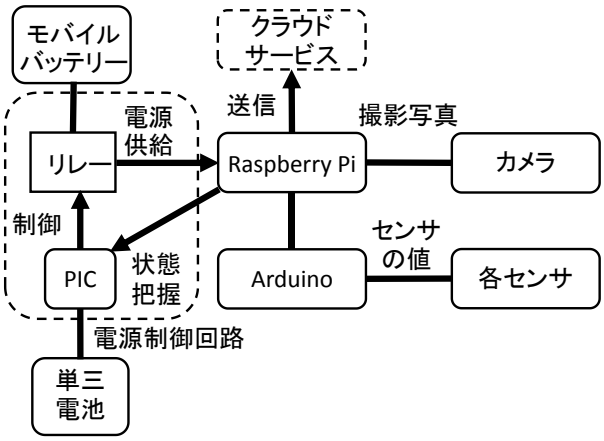

図 2. モニタリングデバイスの構成

Copyright (c) 2017 by the Information Processing Society of Japan

4.4 電源制御回路による間欠動作回路

電源制御回路は、モバイルバッテリーから Raspberry Pi に流れる電流を制御する。PIC の GPIO は 25 mA 程度の電流しか出力できないため、Raspberry Pi を起動するための電源として適していない。Raspberry Pi を使用するには数百 mA が必要である。よって、モバイルバッテリーを電源制御回路に接続し、Raspberry Pi の電源として利用できるようにする。ただし、電源制御回路の電源はモバイルバッテリーではなく、単三電池を用いる。

電源制御回路は以下のようなフローで動作する。

(P-1) 単三電池からの電源供給により、PIC が起動する。

(P-2) PIC の GPIO ピンの一つを High にし、リレー素子の FET を動作させ、Raspberry Pi を起動する。

(P-3) Raspberry Pi の GPIO ピンの一つを PIC に接続し、Raspberry Pi の状態を取得する。

(P-4) Raspberry Pi が終了するまで待機する。

(P-5) 終了後、PIC の GPIO ピンを Low にし、FET により Raspberry Pi の電源を完全に遮断する。

(P-6) 定められていた時間をカウントし、時間が経つと(P-2)に戻る。

この電源制御回路の動作を実現するために、Raspberry Pi は以下のようなフローで動作する。

(R-1) Raspberry Pi は起動後、GPIO ピンの状態を High になるようにプログラミングしておく。

(R-2) データの送信を終了した後、シャットダウンコマンドにより終了待機モードに移行する。このとき、GPIO ピンの状態は Low になる。

終了待機モードとは、Linux の halt コマンドを実行した後の状態に似ている。Raspberry Pi は、シャットダウンコマンド実行後、OS の機能は停止するが電源は完全に遮断していないため、電力を消費し続けている状態である。完全に電源を遮断するには電源ケーブルを抜くか、今回の手法のようにリレー素子で電源を遮断しなければならない。

5. 実装

5.1 データ管理用システム

データ管理用システムとして、Web 上で動作するシステムを構築した。そのインターフェースを図 3 に示す。データ管理用システムは、Raspberry Pi から送られ、クラウドサービスに保存されたデータを取得し、表示する。このシステムでは以下の内容を確認できる。

・デバイスの状態（動作中かスリープ中かを表す）

・デバイスの現在の温度・湿度・照度（動作中のみ）

・センサデータのグラフ

・撮影された写真とその日付

5.2 モニタリングデバイス

実際に Raspberry Pi に載せる基盤を実装した様子を図 4 に示す。電源制御回路を実装したユニバーサル基盤は Raspberry Pi に載せることが可能である。

5.3 コスト

モニタリングデバイスに構成された部品から、必要なコストを試算した。結果を表 1 に示す。表を見ると、総費用は約 16,000 円で、今回製作した Raspberry Pi に載せる基板にかかるコストは約 1,000 円である。大部分を占める費用は、デバイスとモバイルバッテリーである。今回の実装では、インターネットに接続するデバイスを用いなかったため、そのコストを考慮していない。データを管理ソフトウェアのサーバ、クラウドストレージ・データベースは無料のものを使用したためコストには入れていない。しかし、規模によっては別途費用が必要な場合も考えられる。

6. 評価

6.1 電源制御回路の消費電力・消費電力量

電源制御回路の消費電流を電流計で測定した。その結果を表 2 に示す。このとき、PIC は単三電池 3 本、電圧 4.5V で動作させた。その結果をもとに、電源制御回路の消費電力を計算した結果を表 3 に示す。測定は、基板を実装した状態で電源制御回路に流れる電流を測定した。しかし、実装した基板の部品ごとの電流を計測するのは困難であったため、電源から流れる電流を測定し、回路全体の電流のみを測定した。ここでは、電池からの電流を測定した。表 2 より、電源制御回路の消費電流は約 300~500 μA であった。また、リレー素子がオンのときとオフのときを比較すると、約 170μA の差があった。これは、FET に対するピンから出力される電流の差であると考えられる。

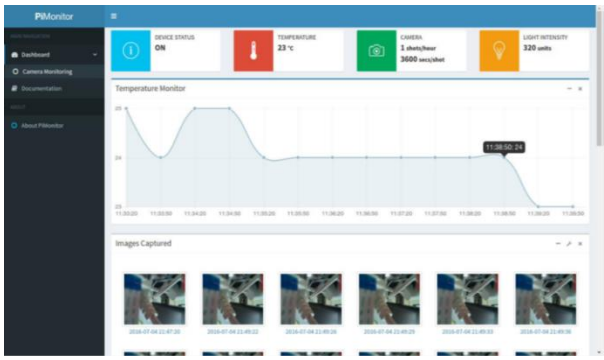

図 3. データ管理用システム

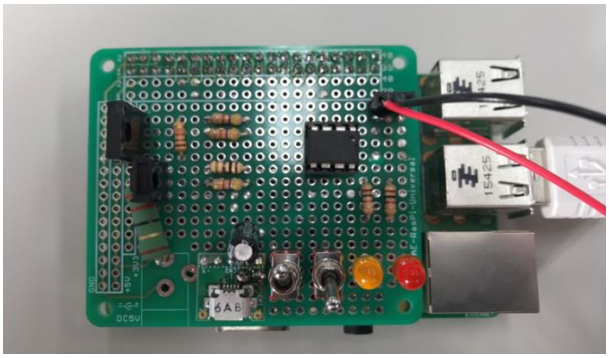

図 4. 電源制御回路を搭載した Raspberry Pi

Copyright (c) 2017 by the Information Processing Society of Japan

ここで、モニタリングサイクルを一定間隔で一日 10 回実行し、1 回あたり 120 秒で終了する間欠動作を考える。この間欠動作のスリープ状態（リレー素子がオフ）の時間は、以下のように求められる。

$$24[\mathrm{h}]\times 3{,}600[\mathrm{s}] - 1\text{ 日の起動回数}[\text{回}]\times\text{起動時間}[\mathrm{s}] \quad (1)$$

今回の仮定の下で計算すると、

$$24[\mathrm{h}]\times 3{,}600[\mathrm{s}] - 10[\text{回}]\times 120[\mathrm{s}] = 8{,}640[\mathrm{s}] \quad (2)$$

となる。電源制御回路は 8,640 秒のスリープ状態を終えると、リレー素子をオンにし、120 秒のモニタリングサイクルに移行する。これより、モニタリング回数 10 回においての一日あたりの消費電力量を計算した。その結果を表 4 に示す。モニタリングサイクル 1 回と、スリープ時間を含めた電源制御回路の消費電力量は 12.531J であった。

6.2 Raspberry Pi と Arduino の消費電力・消費電力量

Raspberry Pi の消費電流をテスターで測定した。その結果を表 5 に示す。このとき、モバイルバッテリーの出力電圧である 5.0V で動作させた。ここでは、モバイルバッテリーからの電流を測定した。その結果をもとに、Raspberry Pi の消費電力を計算した結果を表 6 に示す。Raspberry Pi の OS 起動時は、演算数の増減が激しく、電流の増減が激しいため正しく測定できない。測定中見られた電流のおおよその平均値として 300 mA とした。表 5 より、カメラを使用して撮影すると、消費電流が約 100 mA 上昇する。撮影を終えると、通常時の消費電流に戻る。シャットダウンコマンドにより終了すると、すぐに消費電流は低下し約 80 mA となった。

ここで、モニタリングサイクル 120 秒の間に Raspberry Pi が遷移した状態ごとの時間を考える。電源を入れて起動にかかる時間と、シャットダウンコマンドを打ってから終了待機モードへ遷移するまでの時間は 10 秒であった。通常モードにおいてカメラを 5 秒だけ使うと仮定し、インターネットとの通信を確立・送受信する時間を 95 秒と仮定する。これより、モニタリングサイクル 1 回の Raspberry Pi と Arduino の消費電力量を算出した。これを表 7 に示す。それぞれ電力量を計算した結果、モニタリングサイクル 1 回の消費電力量は、175.2 J であった。Arduino の電源は、Raspberry Pi から供給されているため、Raspberry Pi の消費電力に含有されている。実際には 20 mA ほど Arduino に電流が流れている。

6.3 運用可能日数の試算

算出した電力量からモニタリングデバイスの運用可能日数を試算する。ここでは、電池は 1 本の容量を 1.5V/2,000mAh とした単三電池、モバイルバッテリーは 74Wh(3.7V/20,000mAh 程度)の容量を持ったものを使用すると仮定する。すると、電池は 3 本で 43,200J、モバイルバッテリーは 266,400J の電力量を持つ。

電池の電力量をP_C[J]、一日に行うモニタリングサイクルの回数を N[回/日]とし、モニタリング一回の電源制御回路

表 1　コスト計算

デバイス	値段[円]
PIC12F683	120
MOSFET 2SK2232	100
トランジスタ 2SC3422	40
ユニバーサル基板	100
抵抗、コンデンサ、LED、DIP キット等	300
Raspberry Pi 3	5,000
Raspberry Pi Camera	3,000
Arduino Nano	3,000
単三電池 4 本 1.5V	300
モバイルバッテリー 74Wh (3.7V/20000mAh)	4,000
合計	15,960

表 2　電源制御回路の消費電流 I [μA] (電圧=4.5V)

リレーの状態	1 回目	2 回目	3 回目	平均値
ON	485	483	489	486
OFF	315	312	319	315

表 3　電源制御回路の消費電力

リレーの状態	電圧[V]	電流[μA]	電力[W]
ON	4.5	483	0.00219
OFF	4.5	312	0.00142

表 4　電源制御回路の一日あたりの消費電力量

リレーの状態	電力[W]	稼働時間[s]	電力量[J]
ON	0.00219	120	0.262
OFF	0.00142	8,640	12.269
合計		8,760	12.531

表 5　Raspberry Pi の消費電流 I [mA] (電圧=5.0V)

状態・モード	1 回目	2 回目	3 回目	平均値
起動中	平均して 300 程度			300.0
通常	307.0	308.6	308.0	308.0
通常＋カメラ	406.0	408.0	407.0	407.0
終了待機	76.0	75.0	76.0	75.7

表 6　Raspberry Pi の消費電力

状態・モード	電圧[V]	電流[mA]	電力[W]
起動中	5.0	300	1.500
通常	5.0	307	1.539
通常＋カメラ	5.0	407	2.035
終了待機	5.0	75.7	0.378

表 7　Raspberry Pi のモニタリングサイクル 1 回における消費電力量

状態・モード	電力[W]	時間[s]	電力量[J]
起動中	1.500	10	15.00
通常	1.539	95	146.2
通常＋カメラ	2.035	5	10.18
終了待機	0.378	10	3.783
合計		120	175.2

Copyright (c) 2017 by the Information Processing Society of Japan

の電力量をP_1[J]、間欠動作のスリープ一回の電力量をP_2[J]とすると、電源制御回路の運用可能日数は以下の式で表される。

$$\text{電源制御回路の運用可能日数} = \frac{P_C}{(P_1 + P_2) \times N} \ [\text{日}] \qquad (3)$$

また、モバイルバッテリーの電力量をP_B[J]とし、Raspberry Pi のモニタリング一回の電力量をP_3[J]とすると、Raspberry Pi の運用可能日数は以下の式で表される。

$$\text{Raspberry Pi の運用可能日数} = \frac{P_B}{P_3 \times N} \ [\text{日}] \qquad (4)$$

(3),(4)式をもとに一日に行うモニタリングの回数を変更したときの､デバイスごとでの運用可能日数を図 5 に示す。グラフから、電源制御回路は起動回数を多くしても約 1 年の動作が見込まれる。それに比べ、Raspberry Pi などは起動回数を多くすると反比例的に運用可能日数が減少する。図 5 のグラフの交点より、最大動作日数は、一日 4 回モニタリングサイクルを実行した時の約 1 年間である。今回の目的は数ヶ月の運用であったため､目標を 3 ヶ月と仮定する。一日 16 回モニタリングサイクルを行うと 95 日間運用可能であり、目標を達成可能である。

6.4 間欠動作性能の比較

間欠動作を行うにあたっての性能を比較する。6.3 と同等の仮定で、一日 16 回のモニタリングサイクルを実行するとするとき、モニタリングデバイスの運用可能日数を試算した。以下の場合において、それぞれ試算を行った。

(A) 間欠動作を行わない場合
(B) 間欠動作をソフトウェア制御で行った場合
(C) 間欠動作を電源制御回路で行った場合

Raspberry Pi には、終了待機モード状態で特定の GPIO ピンに外部から入力を入れることで、ソフトウェア上でリセット割り込みが発生する。すると、本体をリセットされ本体が再起動し、終了待機モードから通常モードへ移行する。これらにより、ソフトウェアによる間欠動作を実装可能である。この間欠動作は電源を完全に切らないのが特徴である。これと今回実装した電源制御回路による間欠動作の性能比較を行う。また、間欠動作自体の比較対象として、モニタリングサイクルを終えても通常モードで待機する、間欠動作を行わない場合も想定した。

一日 16 回のモニタリングサイクルを実行するときにおける、Raspberry Pi の運用可能日数を図 6 のグラフに示す。グラフより、ソフトウェアと電源制御回路での間欠動作の差は約 80 日であり、電源制御回路の有用性が認められる。この差の原因は、ソフトウェアで間欠動作を行う場合は、モニタリングサイクルを終え終了待機モードに移行しても、約 80mA の電流が流れ、電力を消費し続けるからである。それと比較して電源制御回路は完全に電源を落とすため、電力を消費しない。よって、メインバッテリーの使用可能時間を大きく延ばすことが可能である。

一方、間欠動作を行わない場合は電力の消費が激しいため、2 日持たない計算となった。間欠動作を行わない場合は、長期運用は困難である。

6.5 稼働時間実測実験

理論的な稼働時間の試算が正しいかを確かめるために、実測による実験を行った。この実験ではモバイルバッテリーを装着したモニタリングデバイスが、一定期間でバッテリー容量を使い果たすようにモニタリング回数を調整する。その後、一定期間システムを動作させ、理論的に計算したモニタリングサイクルの合計回数と、実際に行ったモニタリングサイクルの合計回数を比較する。本来であれば、目標である 3 ヵ月を達成するために動作期間を設定するべきである。しかし、日程の関係上動作期間を約一週間とした。また、この実験では電源制御回路用の電池の稼働時間は考慮しないものとし､Raspberry Pi とモバイルバッテリーとの関係のみを調べる。使用した実験器具を次に示す。

・mA 測定用：テスター Kaise KU-2600
・μA 測定用：mA 電流計(レンジ 1 mA) YOKOGAWA.E.W
・モニタリングデバイス一式

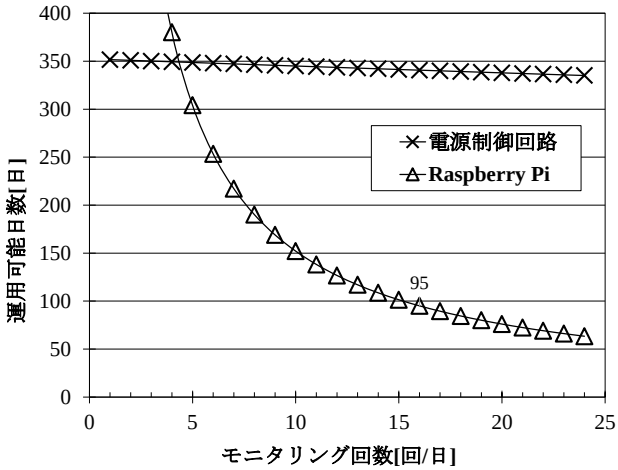

図 5. 運用可能日数

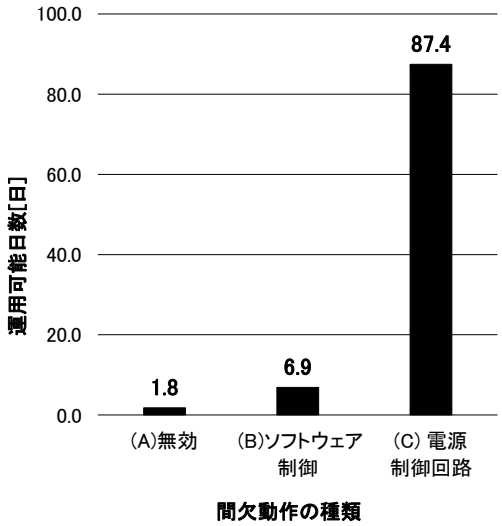

図 6. 間欠動作の性能(モニタリング：16 回/日)

Copyright (c) 2017 by the Information Processing Society of Japan

・モバイルバッテリー：

Ombar iJuice 12.4Wh 3.7V/3,350mAh リチウム電池

使用したモバイルバッテリーは、理論で仮定したモバイルバッテリーよりも容量が小さい 12.4Wh のものを使用した。このモバイルバッテリーは 44,640J の電力量を持つ。また、予備実験としてモバイルバッテリーの定格の約 9 割を使用可能であることを確認した。実験は研究室内で行い、データ転送には Wi-Fi を用いた。

12.4Wh のモバイルバッテリーの 9 割の電力量が全て使われると仮定する。(3),(4)式よりモニタリングサイクルが 120 秒であれば、一日 33 回モニタリングを行うことで、モニタリングデバイスは 6.94 日の動作が可能である。このときの間欠動作のスリープ時間は 2,498 秒である。

実験を行い、稼働時間を測定した。理論的なモニタリング回数及び運用日数と、実測値を表 8 に示す。理論的なモニタリング回数が 229 回に対し、実測では 202 回となった。これは、理論に対して 88.2%に当たる。完全に理論通りにならなかった理由を考える。ここで、大きな理由としては Raspberry Pi の Wi-Fi 通信による消費電力の増加がある。6.2 の測定では通信は行っておらずスタンドアロンであったため、通常時の消費電流に誤差が生じたと考えられる。

7. まとめ

7.1 成果

本稿では、Raspberry Pi を用いた環境モニタリングシステムの試作を行った結果についてまとめた。想定するモニタリング環境は、電力の供給が困難である場合を想定した。そのため、測定を行うデバイスはモバイルバッテリーで動作する。この長寿命化のために、一日数回起動する間欠動作を行うように制御するハードウェアを試作し、数ヶ月単位で動作できるようにした。システムの特徴としてセンサのデータだけではなく、写真も取得できる。

システムは主にデータを管理するソフトウェア、間欠動作を行うハードウェアより構成される。本稿ではシステムの試作と、電源制御用ハードウェアの評価についてまとめた。Raspberry Pi はシャットダウンしても電源を完全に切らない限り数十 mA の電流が流れる。このままでは数ヶ月単位の動作が不可能なため、リレー素子より電源が完全に切れるようにした。これによってモニタリングを行わない時は電力を消費しない。代わりに、PIC がリレー素子を制御する電源制御回路が電力を消費する。しかし、電源制御回路の消費電流は数百 μA であるため、電源制御回路は一年程度動作が可能である。これにより、理論上数ヶ月単位の動作を可能とした。実際に、理論値で一週間程度システムが動作するように設定し実測での実験を行った。実際の結果と理論値と比較した結果、理論値の約 9 割の動作時間となり、おおよそ理論通りにモニタリングデバイスが動作することを確認できた。

表 8　実測実験結果

	モニタリング回数[回]	運用日数[日]
理論	229	6.94
実測	202	6.12

7.2 今後の課題

間欠動作では、実測での実験の動作日数が完全に理論通りにはならなかった。これは、主に Wi-Fi 通信での消費電力の増加や、回路の不具合などが原因と考えられる。今後はその点を考慮し、実測値に近い理論値になるように条件を追加していくことを課題とする。また、使用した Raspberry Pi 3 はモニタリングデバイスの中心としてはやや性能過多であった。よって、デバイスをより小型で安価である Raspberry Pi Zero に変更することで、消費電力を削減し運用可能日数を増加できると考えている。その環境での実験も今後の課題とする。加えて、コストも削減可能である。現在の試算ではさらに Arduino の廉価版を用いるなどにより、全体でのコストは 10,000 円まで削減可能である。

間欠動作以外では、外部との通信が挙げられる。インターネットとの通信を行う環境がなければシステムは動作しない。モバイルルーターを数ヶ月単位で動作させるには、他のバッテリーが必要である。この問題を解決する方法として、IoT デバイス用の低電力通信デバイスを利用する方法がある。システムを完成させるには、外部との通信を確立し、必要となる実地テストなどを経て改善をしていく必要がある。システム面では、Web サイトのインターフェースの改善をはじめ、災害が起こりそうな場合にスリープ中であっても、外部からの指令で強制起動しモニタリングを行うなど、システム全体の向上も今後の課題である。

参考文献

[1] 日本の防災対策への国際的な評価: 株式会社三菱総合研究所
http://www.mri.co.jp/opinion/column/trend/trend_20150512_1.html
(accessed 2017.1).

[2] 暮らしに役立つ情報: 政府広報オンライン
http://www.gov-online.go.jp/useful/article/201106/2.html
(accessed 2017.1).

[3] 国立研究開発法人情報通信研究機構法第五条第二項一号

Copyright (c) 2017 by the Information Processing Society of Japan