エネルギーハーベスティング型アプリケーション設計を最適化する電力収支シミュレータ

飯塚 達哉[1]　成末 義哲[1,2]　川原 圭博[1]　浅見 徹[1]

概要：センサネットワークにエネルギーハーベスティングを利用することは，アプリケーションの稼働時間の制限を取り除き，電池交換コスト，メンテナンスコストを著しく小さくするために，非常に有意義である．しかし，エネルギーハーベスティングを利用したアプリケーション (EH アプリケーション) において，発電電力量と消費電力量のバランスが上手く取れるように適切に設計を施すのは未だに難しい．設計を補助するツールとして，センサネットワークの動作を推定する多くのシミュレータが提案されてきたが，アプリケーションの動作が外部環境に大きく依存する場合においては，動作および消費電力の正確な推定が上手くできていなかった．しかし，アプリケーションの消費電力が５種類のタスクのみに依存するように限定して近似することで，アプリケーションの消費電力を予測は簡単になり，高い精度と忠実性を保持してのエネルギー収支のシミュレーションを可能にする．本稿ではこれを利用して，EH アプリケーションにおけるノード単位での正確なエネルギー収支のシミュレーションを可能にする計算モデルを提案した．そして，その計算モデルに基づいてユーザが想定する設計におけるエネルギー収支の時間推移を可視化することで，設計の支援を行うシミュレータツール PLEH (PLanning simulation tool for Energy Harvesting applications) を実装し，その有用性を検証した．

Simulation of Energy Transition to Optimize Design of Energy Harvesting Applications

TATSUYA IIZUKA[1]　YOSHIAKI NARUSUE[1,2]　YOSHIHIRO KAWAHARA[1]　TOHRU ASAMI[1]

1. はじめに

Arduino や Raspberry Pi などのオープンソースハードウェアの登場によって，マイコンを用いたアプリケーションの開発の敷居が下がり，開発の場が活性化することで，IoT 化を促進している．しかし，アプリケーションの数が増えるにつれて，電池交換やメンテナンスに要するコストが膨大になっており，IoT 化への障壁となっている．そこで，環境からのエネルギーを電力に変換して稼働するエネルギーハーベスティングの技術は，アプリケーションは理論的には永久な動作を可能とするため注目されている．

しかし，エネルギーハーベスティングを利用したアプリケーション (EH アプリケーション) を適切に設計することは一般的に難しい．適切な設計とは，EH アプリケーションが停止することなく動作し，かつ電池が常に満充電されているときに生じる無駄な発電電力が生じないような設計である．これを達成するためには，消費電力と発電電力の正確な予測や見積もりが必要となるが，両者はアプリケーションの構成とアプリケーションを取り巻く環境から大きな影響を受ける．例えば，アプリケーションがセンサを間欠駆動させる場合はセンサの駆動間隔を長くすれば消費電力は小さくなったり，太陽電池を用いて発電する場合は太陽電池をより日差しの強い場所に設置すれば発電電力は大きくなったりする．消費電力や発電電力は様々な要因で変化するため見積もりが難しく，その見積もりを高い精度で行えない者が EH アプリケーションを設計すれば，電力不足で全く稼働しなくなることも起こり得る．一時的に上手

[1] 東京大学　大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[2] 日本学術振興会特別研究員 DC
Research Fellow of Japan Society for the Promotion of Science

く稼働したとしても，天候の移り変わりなどの外環境の変化によって，設計者の想定外の時刻にアプリケーションが停止する可能性もある．

そこで本稿では，EH アプリケーションの設置場所の環境と設計情報に関する情報が与えられた時に，その動作とエネルギー推移をシミュレートするツールである PLEH (PLanning simulation tool for Energy Harvesting applications) を実装した．高い精度での見積もりを行えない者は PLEH を用いてシミュレーションを繰り返し試行錯誤することで，より安定して効率的に稼働するような設計が可能となる．一方で，EH アプリケーションの製作に多くの経験を持ち，高い精度で消費電力と発電電力の見積もりを行える者に対しても PLEH は有用である可能性は高い．そのような設計者の多くは EH アプリケーションが停止することを確実に回避するために，発電電力の見積もりは実際より小さく，消費電力の見積もりは実際より大きくなっている．これは EH アプリケーションが停止しないための設計指針として間違ってはいないが，最適な設計と比較すると，エネルギー利用効率は低く，ハードウェアコストは大きくなる．発電電力と消費電力を正確に見積もった上で，アプリケーションの停止を確実に回避できるような設計は PLEH の実行を繰り返すことで得られるだろう．

本稿では，PLEH を実装するために，EH アプリケーションの動作とエネルギー収支のシミュレーションが可能な計算モデルを提案した．計算モデルを構築するにあたり，設置場所の環境を表す環境モデルと EH アプリケーション設計情報を表すアプリケーションモデルの 2 つを用意した．EH アプリケーションの動作及びエネルギー推移に大きく影響を与える要素として，環境モデルではエネルギーモデル，物理モデル，イベントモデルの 3 つの要素より構成されるモデルを考え，アプリケーションモデルでは動作をタスクの実行の流れとして表し，タスクの種類としてセンシングタスク，駆動系タスク，無線通信タスク，モード切り替えタスク，エネルギー調整タスクの 5 種類のタスクを用意した．このように，EH アプリケーションに関する設置環境とアプリケーションの構成の情報を簡潔に表すモデルを用意することで，必要最低限の入力情報からエネルギー収支の推定を可能にした．そして最後に，アプリケーションの動作とエネルギー推移の時間変化を可視化することで，ユーザに有意義な情報を与えられ，設計の手助けとなることを検証した．

2. 関連研究

エネルギーハーベスティングを利用したワイヤレスセンサネットワーク (WSN) のシステムデザインやエネルギーマネジメントアルゴリズムに関する多くの研究が行われてきているが，提案したシステムの評価に必要なアプリケーションのセンサネットワークの設置には大きな設置コストが生じる．WSN の効率的な運用システムの開発やネットワークプロトコルの研究では，その設置コストはセンサネットワークの規模が大きくなるほどに増加するために非常に大きな問題であった．そのため提案手法の有用性を低コストに検証するツールとして，多くのエネルギーハーベスティング型 WSN のシミュレータが作成されてきた [1], [2], [3], [4], [5]．GreenCsatalia は，既存の WSN シミュレータ Castalia を，複数の発電素子，複数のストレージを持ったワイヤレスセンサネットワークにも適応できるようにしたシミュレータである．WSNsim[4] は，EH アプリケーションをシミュレーションする上で，考慮しなくてはならない多くの要因を示し，理想的なシミュレータのモデルを提唱したが，現在利用できる状況にない．また，TinyOS[6] を用いたアプリケーションを対象とした有名なシミュレータとして TOSSIM がある [7]．これは消費電力をシミュレートできるように拡張されているが，エネルギーハーベスティングに対応しておらず，また外部環境による消費電力の変化も考慮できていない [8]．ekho[9] は，発電素子から得られる電力量は負荷電圧によって変化するため，時々刻々の IV 曲線を記録し，再現することで，実験室においてエネルギー環境を再現できるエミュレータである．

EH アプリケーションまたは WSN アプリケーションのために，多くのシミュレータやエミュレータが開発されてきたが，それらのほとんどはユーザに専門的な知識を要求していた．そのため，自身が作成したアプリケーションにエネルギーハーベスティングの技術を適応し長時間稼働させたいという欲求がある場合でさえ，エネルギーハーベスティングに関する知識を多く持たないユーザのアプリケーション設計支援を行うツールは存在しなかった．

3. システム設計

3.1 PLEH の特徴

外環境にインタラクティブに動作するアプリケーションの場合，外環境をモデル化しなくてはアプリケーションの動作をシミュレーションできない．今までのほとんどのシミュレータは EH アプリケーションの動作に影響を与える外環境を無視していため，より忠実なアプリケーションの動作を再現できなかった．例えば，図 1 のようなフローチャートでプログラムが表現できる自動ヒータシステムをシミュレートしたい場合，今までのシミュレータでは不可能であった．なぜなら，フローチャートにはセンシングした部屋の温度によって，その後ヒータを駆動させるかどうかが決定され，このヒータを駆動するというタスクがいつ実行されるかによってアプリケーションの消費電力は大きく変化するためである．このようなアプリケーションにおいて消費電力をより正確にシミュレートするためには，時間経過による部屋の温度変化とヒータによって上昇する温

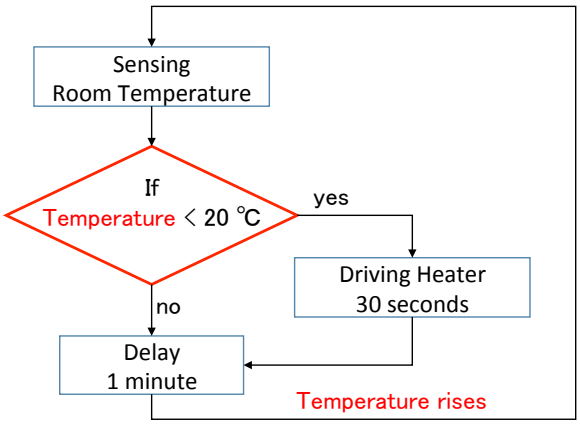


図 1　自動室温ヒータシステムのプログラムフローチャート

度を考慮して計算しなくてはいけなかった．

そこで本稿では，温度などのアプリケーションの動作に影響を与える外部環境の情報をモデル化した．そしてシミュレーション対象とするアプリケーションの消費電力に影響を与えるタスクの種類を限定し近似すれば，アプリの総消費電力の計算ははそれらのタスクの実行タイミングをシミュレートすることで可能となる．これによって，タスクの実行に影響を与える外環境の記述ができればアプリケーションの動作および消費電力の計算が可能となる．

3.2　対象アプリケーションの具体例と構造

PLEH の対象とするアプリケーションは，センサネットワークを構成する各ノードである．例えば一つの例として，図 2[a] の CES2015 にて Parrot が発表した Parrot Pot という製品がある．このような自動で植物を育てるアプリケーションは，土壌の乾燥を検知したときにポンプを駆動して水やりを実行したり，土壌に関するデータをセンシングして，データをサーバに送信したりする．このアプリケーションは実行するべき複数のタスクを持っていて，どのタイミングでどのタスクを実行するかはマイコンによって制御される．マイコンは稼働しているだけでも電力を消費するので，このアプリケーション全体で消費される電力は，マイコンの待機電力とタスクの消費電力に大別される．これはマイコンベースのアプリケーションに一般的に言えることである．一方で，センサネットワークノードがエネルギーハーベスティングを利用して稼働する場合，EH アプリケーション特有のソフトウェア制御がなされているものも多い．それは，変化する発電電力量と，予測が難しい消費電力のバランスを取る操作であるエネルギーアウェアオペレーションである．EH アプリケーションは発電電力に比べて消費電力が大きい場合，エネルギー不足になって停止してしまう．これを避け，パフォーマンスを向上させるために，エネルギー状況に応じて消費電力を動的に変化させる手法がよく用いられる．例えば，図 2[b] は EH アプリケーションを記述するのに特化した言語である Eon を用

[a]Parrot Pot[10]

[b] 亀の自動位置追従センサ

図 2　EH アプリケーションの具体例

いて記述された，亀の位置を追従するためアプリケーションである．甲羅に張り付けた太陽電池から得られるエネルギーを利用して GPS データを取得し続ける．このアプリケーションの特徴は，現在の電池残量と太陽電池の発電量を考慮して，GPS データを取得する間隔を変化させることである．電池が満充電されて発電エネルギーを効率よく利用できない，もしくは電池残量がなくなり停止してしまう，といういずれの状態にも陥らないように，GPS の起動間隔を調整している．このようなデューティサイクルの動的制御は EH アプリケーションにおいてはよく用いられており，消費電力を大きく変化させる．

例に挙げたような EH アプリケーションのエネルギー収支を推定するために，計算モデルを構築した．EH アプリケーションの動作及びエネルギー収支に影響を与える全ての要素は，アプリケーションを取り巻く外環境とアプリケーションの構成の 2 つに分類できると考えられ，前者を表すモデルを環境モデル，後者を表すモデルをアプリケーションモデルと呼ぶことにする．計算モデルを構築するにあたって，環境モデルやアプリケーションモデルに取り入り得る変数は数多く存在する．当然，より多くの要素を取り入れれば，より高精度のシミュレーションが可能になるが，その一方でユーザに多くの入力変数を要求することになる．本稿では，このシミュレーションの精度と入力の複雑性のトレードオフを考慮し，最少の入力で最大の精度を達成できるような EH アプリケーションの動作及びエネルギー推移を計算するモデルを構築することを試みた．提案するモデルの概念図を図 3 に示す．

環境モデルにおいて，アプリケーションの発電電力に影響を与える要素として，太陽光発電であれば日射量であったり，振動発電であれば振動量であったりなどの発電源のエネルギー密度が考えられる．この発電量の推定にエネルギー密度を利用するために，エネルギーモデルを環境モデルの一部として抽出した．また，アプリケーションが電力を消費するのは後述するタスクの実行が行われた時であり，タスクの実行時刻を変化させる環境の要素はアプリケーションの消費電力に影響を与え得る．このようなタスクの実行時刻に影響を与える環境モデルの要素として，物理モデルとイベントモデルを用意した．

アプリケーションモデルにおいて，PLEH はアプリケーションの動作をタスクの実行の流れとして表現する．アプリケーションが消費する電力を外部モジュールの駆動によ

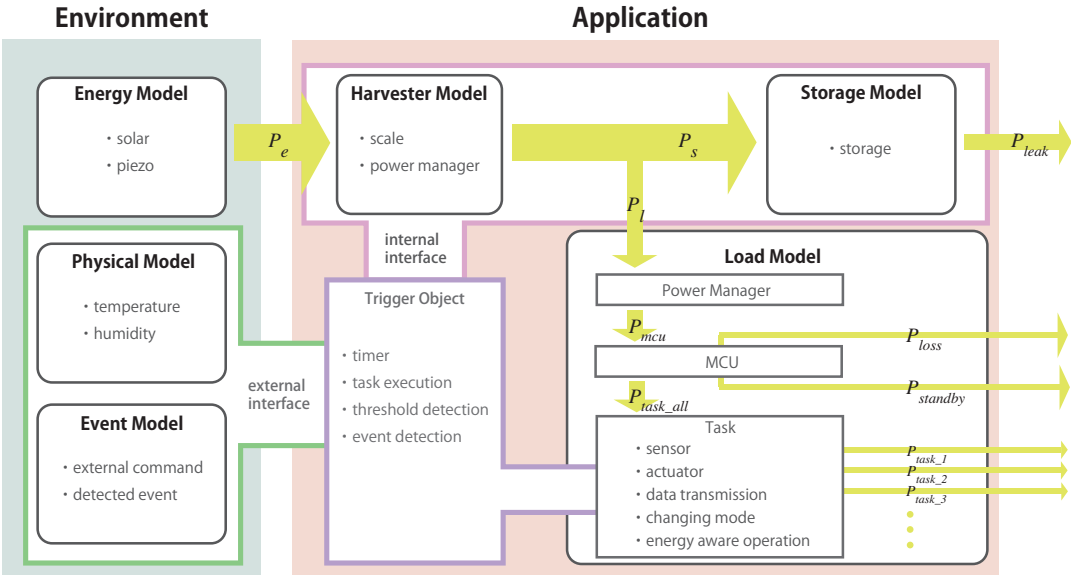


図 3　提案する EH アプリケーションのエネルギー収支モデル

り消費する電力，マイコン本体の待機電力，電圧変換で無駄になる電力の 3 つに分類し，アプリケーションの全体の消費電力をこれら 3 つの電力の和と考えた．この時，外部モジュールの消費電力はセンサやアクチュエータの駆動時間によって変化し，この動作の記述としてセンシングタスクと駆動系タスクの 2 つ，待機電力は無線通信を行う時やスリープモードへ切り替える時に変化し，この動作の記述として無線通信タスクとモード切り替えタスクの 2 つを用意した．さらに，例示したカメの自動位置追従センサで用いられているような発電電力量や電池残量に応じて外部モジュールの駆動間隔を調整する操作は消費電力を変化させるので，この動作の記述としてエネルギー調整タスクを用意し，合計で 5 種類のタスクを用意することで，EH アプリケーションの動作及びエネルギー収支を少ない入力情報でかつ高い精度で計算できると考えた．

以下では，提案する環境モデル，アプリケーションモデルの詳細をそれぞれ節 3.3 と節 3.4 に記す．そしてその後に，提案したモデルの妥当性を論ずる．

3.3　環境モデル

環境モデルは，EH アプリケーションを取り巻く環境を記述可能な形式にモデル化したものである．これは，同一のアプリケーションでも，設置される場所によってエネルギー推移が変化していくことを再現できなくてはならない．このエネルギー推移に影響を与える外部環境として，エネルギーモデル，物理モデル，イベントモデルの 3 つを用意した．

(i) エネルギーモデル

外環境のエネルギー密度は，EH アプリケーションの発電量に大きく影響を与える．これは，同一の発電素子であろうと設置場所のエネルギー源の大きさ (例えば太陽電池であれば日射量であり，振動発電であれば振動量) が異なれば発電量は変化し，EH アプリケーションのエネルギー推移も変化するからである．このモデルでは，各時刻におけるエネルギー密度 P_e が定義される．

(ii) 物理モデル

EH アプリケーションの動作に影響を与えるものに，センシングする物理量がある．図 1 のように，センサ値によって直後に実行されるタスクが決定される場合，このセンシングされる物理量の変化も考慮されなくてはならない．この物理モデルは，物理量が他の分野で何かしらの知見が得られている場合に特に有用である．例えば温度や湿度に関しては気象学を用いてある程度の予測が可能であろう．また，後に説明する駆動系タスクによって物理量に変化がもたらされる場合もある．これらを考慮して，このモデルでは，各時刻における物理量の変位量が定義される．

(iii) イベントモデル

物理量と同様に，EH アプリケーションの動作に影響を与えるものとしてアプリケーションが検知するべきイベントが存在する．例えば，人の存在を検知した時に照明を光らせるアプリケーションを作成するとき，正確な消費電力の計算を行うためには，人が存在する時刻が記述されなくてはならない．人の存在を物理モデルとして表現することは難しいため，検知するべきイベントとして物理モデルを一般化したモデルである．イベントの発生の有無を 1 もしくは 0 で表現する．このモデルでは，各時刻におけるイベントの発生の有無が定義される．

3.4 アプリケーションモデル

アプリケーションモデルは，作成するアプリケーションのハードウェアおよびソフトウェアの情報を記述可能な形式にモデル化したものである．ここで対象とする EH アプリケーションは，発電量を決定するハーベスタモデル，ストレージの性質を決定するストレージモデル，消費電力を決定する負荷モデルの 3 モデルから構成されると考えることができる．以下では，それぞれのモデルの構造とエネルギー推移の計算式に関して詳細に述べる．

(i) ハーベスタモデル

ハーベスタモデルでは，エネルギーモデルで定義された外部環境情報から発電量を定めるモデルである．発電量は外部環境のエネルギー密度と環境発電素子の規模によって定まる．しかし，実際の EH アプリケーションを稼働させるためには，発電素子の電圧の昇降圧が必要となってくるので，パワーマネージャによるエネルギー損失が生じる．この損失を表現するために効率 η_{h} を用意した．プロパティとして，ハーベスタの規模 $scale(0 < scale)$ ，効率 $\eta_{\mathrm{h}}(0 < \eta_{\mathrm{h}} < 1)$ が定義される．このとき，発電量 P_{h} は以下の式で表せる．

$$P_{\mathrm{h}} = P_{\mathrm{e}} \cdot scale \cdot \eta_{\mathrm{h}} \tag{1}$$

(ii) ストレージモデル

ストレージモデルとは，EH アプリケーションがハーベスタから発電したエネルギーをバッファとして 蓄えるストレージを表現している.EH アプリケーションにおける理想的 なストレージは，容量に制限がなく，充電効率が 100 % であり，漏れ電流がゼロである. しかし，現実 のストレージは容量に制限があり，充電効率が 100 % 未満であるため充電時に損失が生じ，長期間放 置しておけば漏れ電流によって，電池残量が減少していく. これらの特性のうち，充電効率や漏れ電流 は電池の種類毎にほぼ決まっている．これらを考慮して，ストレージの容量，初期エネルギー残量，充電効率，漏れ電流がプロパティとして定義される．ストレージに供給される電力 P_{s} は，ハーベスタモデルで決定された発電電力 P_{h} と，後述する負荷モデルで決定される負荷消費電力 P_{l} を用いて

$$P_{\mathrm{s}} = P_{\mathrm{h}} - P_{\mathrm{l}} \tag{2}$$

と表される．P_{s} は負の値も取り得り，充電するときは充電効率が 100%ではないため，場合分けをしなくてはならない．そこで，以下のように定義される正規化線形関数 $\varphi(x)$ を用いる．

$$\varphi(x) = max(x, 0) \tag{3}$$

P_{s} が求まったとき，ストレージに蓄えられるエネルギー量 S は，充電効率 $\eta_{\mathrm{s}}(0 < \eta_{\mathrm{s}} < 1)$，漏れ電力 P_{leak}，現在の電池残量値 S_0 を用いて，以下の式で表せる．

$$S = S_0 + \eta_{\mathrm{s}} \int_0^T \varphi(P_{\mathrm{s}})dt - \int_0^T \varphi(-P_{\mathrm{s}})dt - \int_0^T P_{\mathrm{leak}}dt \tag{4}$$

(iii) 負荷モデル

負荷モデルは，式 (2) で求まるストレージのエネルギー変化量 P_{s} を決定するために必要な P_{l} を決定するためのモデルである．アプリケーションの消費電力は，マイコン本体が消費する待機電力と，アクチュエータやセンサなどの外部モジュールを利用することによって消費される電力に大別できる．本モデルでは，後者を各タスクの実行による消費電力と考え，センシングタスク，駆動系タスク，データ送信タスク，モード切替タスク，エネルギーアウェア調整タスクの 5 種類でアプリケーションの消費電力を近似計算できると考えた．以下，P_{l} の計算法について述べる．EH アプリケーションを設計する際に，発電素子やストレージから電力が供給されるとき，ほとんど全てのアプリケーションには，DCDC コンバータなどの電圧変換モジュール Power Manager が必要となる．この変換の際の損失を考慮するために，$\eta_{\mathrm{l}}(0 < \eta_{\mathrm{l}} < 1)$ を用意した．これを用いて，負荷に供給される電力 P_{l} は，マイコンに供給される電量 P_{mcu} を用いて，次式で求まる．

$$P_{\mathrm{mcu}} = \eta_{\mathrm{l}} \cdot P_{\mathrm{l}} \tag{5}$$

そして, マイコンに供給される電力は, 待機電力 P_{standby} とタスクの実行で消費される電力 $P_{\mathrm{task\_all}}$，そして無駄になる電力 P_{loss} に分類でき，それらの和で求まる．

$$P_{\mathrm{mcu}} = P_{\mathrm{standby}} + P_{\mathrm{task\_all}} + P_{\mathrm{loss}} \tag{6}$$

P_{loss} は，Arduino や Raspberry Pi などのマイコンボードを使用する場合，電源駆動部にレギュレータが接続されている．レギュレータがスイッチングレギュレータであれば，電力は保存されると近似してもよいが，リニアレギュレータが使用されている場合，Power Manager の出力電圧によって無駄になる電力 P_{loss} は大きく変化し，それは次のように表せる．

$$P_{\mathrm{loss}} = \varphi(V_{\mathrm{in}} - V_{\mathrm{mcu}}) \cdot I_{\mathrm{mcu}} \tag{7}$$

電源駆動部分にスイッチングレギュレータが使用されている場合や，そもそもレギュレータが内蔵されていない場合は $P_{\mathrm{loss}} = 0$ として考える．なお，V_{in} はマイコンに供給される電圧 (Power Manager の出力電圧)，V_{mcu} は，マイコンに内蔵されたレギュレータの出力電圧，I_{mcu} はマイコンに流れる電流を表す．I_{mcu} はタスクの消費電流と待機電流の和で求まる．P_{standby} は，マイコンの種類と状態によって決定されるものである．状態とは，多くのマイコンに備わっているスリープモードであったり，データを送信している時の

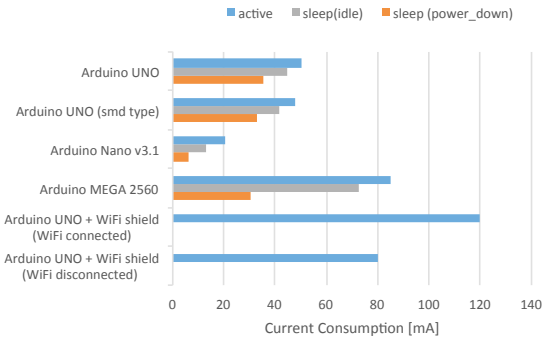


図 4 各種マイコンの待機電力：Arduino UNO， mega，mini，Raspberry Pi の各アクティブ．スリープモードでの消費電力，WiFi シールド接続時の WiFI 接続 or 非接続時の待機電力比較化

状態を表し，待機電力は変化する，マイコンの種類と状態を変化させて計測した待機電力を図 4 に示す．これらの値は，種類とモードによって依存するものと近似できる．図 4 に示した値を P_{standby} として用いることで消費電力の計算が可能となる．あとは，全てのタスクによる消費電力 $P_{\mathrm{task\_all}}$ が求まれば，負荷電力 P_{l} の計算が可能となる．

・タスクモデル

本モデルでは，アプリケーションの動作を再現するプログラムを，タスクの実行の流れとして表現している．そこで，EH アプリケーションのエネルギーに影響を与えるタスクを 5 種類 (センシングタスク，駆動系タスク，無線通信タスク，モード切り替えタスク，エネルギーアウェアオペレーション) に限定した．アプリケーションでプログラムされるソフトウェアを表現するために，各タスクには実行されるタイミングを決定するトリガオブジェクトが記述される．トリガオブジェクトが，内部もしくは外部の状態によって実行タイミングを決められるように，インターフェースを用意した．内部インターフェースは，タイマーや電池残量，発電量の情報を提供し，外部インターフェースはセンシングの値やイベントモデルの情報をトリガオブジェクトに提供する．インターフェースから提供された値を用いて，トリガオブジェクトがタスクの実行の有無を決定する．このような 5 種類のタスクとトリガオブジェクト，そしてインターフェースを用意することで，多くのアプリケーションのソフトウェア制御の表現を可能にする．消費電力を伴うタスクとして，センシングタスク，駆動系タスク，無線通信タスクがあり．センシングタスク，駆動系タスクは，ユーザに電圧値と電流値を記述させ，タスクの消費電力を，電圧と電流の積として求める．

$$P_{\mathrm{task}} = \begin{cases} V \cdot I & (when\ task\ is\ running) \\ 0 & (otherwise) \end{cases} \tag{8}$$

そして，$P_{\mathrm{task\_all}}$ は各タスクの消費電力 $P_{\mathrm{task\_i}}$ の和として表せる．

$$P_{\mathrm{task\_all}} = \sum_{i=0}^{n-1} P_{\mathrm{task\_i}} \tag{9}$$

EH アプリケーションが実行するタスクとして以下の異なる性質をもつ 5 種類のタスクが存在し，それぞれの記述の形式は異なる．これは消費電力を正確に計算する際に必要となる分類であり，かつ十分に多くのアプリケーションの動作を表現できるものと考えた．それぞれについて，詳細に述べる．

センシングタスク

外部環境をセンサを用いてセンシングするタスクを表す．このセンシングタスクを実行することで，物理モデルで定義した物理量を取得できる．

駆動系タスク

駆動系タスクはモータやスピーカー，LED の点灯など電力を消費して外部環境に出力するタスクで ある．比較的大きな電力を消費するタスクであることが想定される．物理モデルで定義した物理量を変化させることができる．また，このタスクの消費電流がマイコンボードの許容電流量を超えた場合を想定して，このタスクの電力供給源をマイコンか Power Manager のいずれとするか指定できるようになっている．

無線通信タスク

無線通信タスクは，電波を発してネットワークに接続されることにより，データを送受信するタスクである．EH アプリケーションは，IoT デバイスとして利用されることの効果が大きく，イン ターネットに接続されることが前提にされていることが多いため，このタスクを用意した．

モード切り替えタスク

モード切り替えタスクは，MCU 本体の状態を変化させるタスクである．上述したように，MCU の待機電力はモードによって変化する．このモードを変化させるタスクを用意した．このタスクによって生じる消費電力は他のタスクに比べて小さいので無視する．

エネルギー調整タスク

エネルギー調整タスクとは，EH アプリケーションではよく用いられるエネルギーマネジメントを 行うためのタスクである．消費電力を伴うタスクであるセンシングタスク，駆動系タスク，無線通信タスクのプロパティを変更し，対象タスクの消費電力を変更できる．このタスクも消費電力を無視できる．

以上の 5 種類のタスクの流れで EH アプリケーションのプログラムを表現でき，かつエネルギー推移の近似計算が可能になった．次に，タスクそれぞれに記述されるトリガオブジェクトを説明する．

トリガオブジェクト

トリガオブジェクトでは，タスクが実行される条件を記述する．その条件として，4 種類 (タイマー，他のタスクの実行，閾値検知，イベントの検知) を用意した．タイマーは，周期的もしくは特定時刻でのタスクの実行を記述できる．他のタスクの実行は，他のタスクの実行が開始，もしくは完了した時に開始される．閾値検知は，インターフェースから提供される値 (物理量もしくは電力，電池残量) が設定した閾値条件を満たしているか確認する．イベント検知は，インターフェースから提供されるイベントの発生の有無の情報をもとに，イベントが発生しているときに実行する．

3.5 提案モデルの妥当性

提案したモデルは，EH アプリケーションの情報を抽象化しているため，エネルギー収支の計算において誤差が生じる．本節では，抽象化によって生じる精度誤差について評価し，その精度とモデル変数の増加による複雑性がトレードオフの関係にあるという観点から，提案モデルの妥当性を検討する．

3.5.1 マイコンの消費電力

提案したモデルでは，消費電力が大きく影響を受け得る変数としてマイコンへの供給電圧に注目し，ユーザが入力する変数として用意した．実際には Arduino UNO に供給する電圧としては 7 V 以上 12 V 以下の範囲が推奨されているため，固定されていない．式 (7) によると，パワーマネージャの供給電圧が変化すると P_{loss} が変化し，式 (6) より，マイコンに供給される電力 P_{mcu} も変化する．ここでは，実際に P_{mcu} が式 (7)，(6) によって高精度に求まるのかを検証する．

式 (6) では，P_{mcu} を決定するために，P_{standby} と P_{loss} を求める必要がある．Arduino UNO，WiFi シールドともに，7 V 以上での駆動が推奨されているので，マイコンの待機電力を 7 V で駆動した時の，空のプログラムを走らせた時の消費電流から算出した．消費電流は，79.8 mA となり，$P_{\mathrm{standby}} = 7 \times 79.8 = 558.6\mathrm{mW}$ となった．P_{loss} は，MCU が電源駆動に使用されているレギュレータの種類で求め方が変わる．Arduino UNO，WiFi シールドの設計回路図は公開されており [11], [12]，それによれば，Arduino UNO はリニアレギュレータ (MC33269D-5.0)，WiFi シールドはスイッチングレギュレータ (LM2736) が内蔵されている．そのため，WiFi シールドによる P_{loss} は無視して考えられるので，Arudino UNO について考える．Arduino UNO 単体に対して空のプログラムを実行させた時の消費電流 I_{arduino} は，7 V 駆動時で，$I_{\mathrm{arduino}} = 50.5$ mA であった．これらの値より，

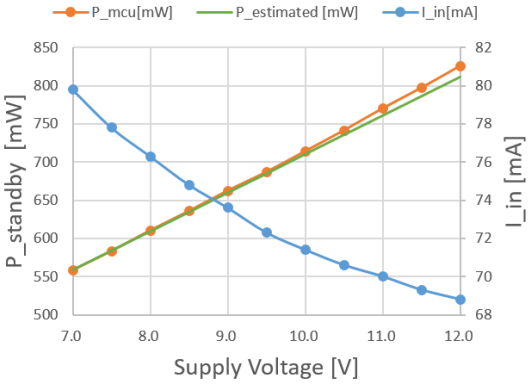


図 5 マイコンの消費電力の実測値と推定値

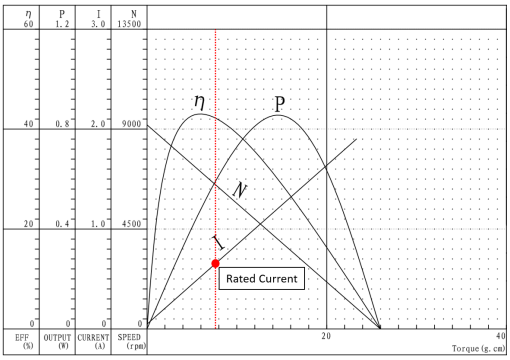


図 6 DC モータの負荷トルクの変化による効率，回転速度，電流，機械出力の変化

$$P_{\mathrm{mcu}} = 558.6 + (V_{\mathrm{in}} - 7) \times 50.5\ mW \tag{10}$$

となる．この推定値と，Arduino UNO に WiFi シールドを接続したモジュールに対して，電源電圧を 7 V から 12 V の範囲で変化させた時の消費電流 I_{arduino} と，それから算出した消費電力を比較したものを図 5 に示す．確かに，高い精度で一致していることが分かる．

3.5.2 駆動系タスクの消費電力

提案したモデルにおいて，駆動系タスクの消費電流は一定値としての入力に限定しており，エネルギー推移の計算を行う際にも駆動系タスクの消費電力は常に一定として近似している．ここでは，駆動系タスクの消費電力を一定として近似することによる誤差について検討する．

駆動系タスクは，モータやポンプ，スピーカ，ヒータ，LED など，人間が認識できるものを外部環境に出力するものを対象としている．ここでは，駆動系タスクの代表である DC モータをとりあげる．評価するにあたって，入手が容易な 1.5 V 駆動 DC モータである FA-130RA2270 を用いた．そのデータシートを図 6 に示す．

この図は，モータにかかる外部トルクの大きさによって，モータの効率，回転速度，電流，機械出力の変化を表している．なお，駆動電圧は 1.5 V で一定である．データシートより定格電流は 0.66 A と記述されており，図中の赤点に相当する．この DC モータではこれ以上の負荷をかける

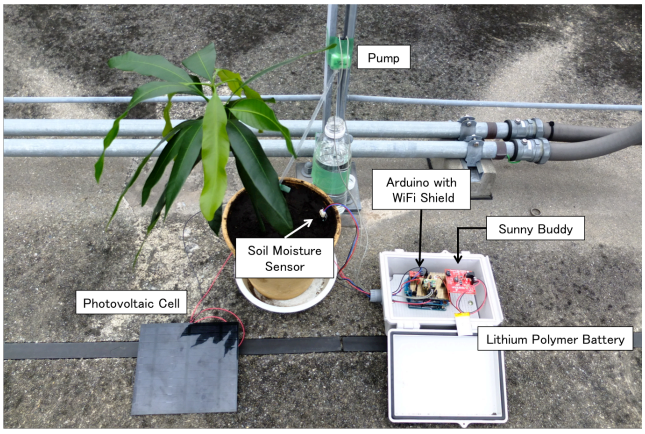


図 7　水遣り機の外観

ことは推奨されていない．しかし，無負荷時の電流は 0.2 A であり，停動電流 (モータの回転速度がゼロになるときの電流) は 2.2 A であるため，DC モータの負荷の大きさの変化によって，電流は 0.2 A から 2.2 A まで変化し，大きな誤差の要因となる．より高精度なシミュレーションを行うためには，モータの負荷の大きさを外部環境として記述させるべきであるが，それはユーザの負担を増やす．つまりは，シミュレータへの入力情報量と計算精度のトレードオフの問題になる．提案したモデルはは，それぞれのアクチュエータは定格電流で動作すると仮定することで，計算の精度は低くなるものの，ユーザに上記のようなデータシートからトルクと電流の関係図を読み取らせる負担をなくした．

4.　実装

上述したデザインの実装について述べる．まず，提案したモデルで定義される変数のうち，時事刻々のエネルギー残量 S の変化量に関係する変数 $(P_{\mathrm{h}}, P_{\mathrm{standby}}, P_{\mathrm{tasks}}, P_{\mathrm{mcu}}, P_{\mathrm{mcu}}, P_{\mathrm{loss}}, P_{\mathrm{l}}, P_{\mathrm{s}})$ を内蔵する Model オブジェクトを生成し初期化する．以降は，式 (1),(2),(5),(6),(8),(9) より，$P_{\mathrm{h}}, P_{\mathrm{standby}}, P_{\mathrm{tasks}}, P_{\mathrm{mcu}}, P_{\mathrm{mcu}}, P_{\mathrm{loss}}, P_{\mathrm{l}}, P_{\mathrm{s}}$ を更新していく．そして，エネルギー推移については，式 (4) より，計算する．ここで dt は，トリガーオブジェクトによって次にタスクの実行状態が切り替わる時刻が求まるので，最も近くに切り替わる時刻までの時間として求める．

このように計算を行うプログラムをブラウザ上で実行できるように javascript を用いて実装した．入力変数用のオブジェクトを用意したので，ユーザは対応するその入力オブジェクトのプロパティに，設計する EH アプリケーションの情報を記述することでシミュレーションが可能となる．現時点では，グラフィカルユーザインターフェース (GUI) は用意していない．今後，オブジェクト変数を簡単に記述できるような GUI を用意する予定である．

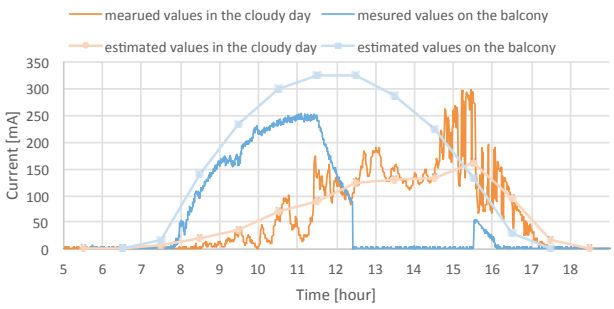


図 8　モデルによって予測された発電電力と実測値

5.　評価

本章では，PLEH を有用性の観点から評価した．評価するにあたって自動水やり機を作成したので，その外観を図 7 に示す．

5.1　発電量の推定精度

太陽光発電を環境モデルとして適応するには，ユーザは太陽光パネルに与えられる時々刻々の日射量を推定しなくてはならない．日本の幾つかの地点における，1 時間で得られる日射量の合計値のデータは気象庁のウェブサイト [13] より入手することができる．現時点で PLEH では，このデータをオブジェクトに入力した時にエネルギー密度 P_e に換算してシミュレーションを実行するインターフェースを用意されている．この節では，この気象庁から容易に入手可能なデータを用いた発電量の推定が，どれほどの精度であるかを検証する．発電量を計測するにあたって，リチウムイオンポリマー電池の太陽光充電モジュールである Sunny Buddy [14] によって発電された電流値を 30 秒間隔で測定した．発電電流から日射量を推定するにあたり，電池電圧を 3.6 V，Sunny Buddy の発電効率を公称値である 0.8 とした．過去の研究では，晴れの日でかつ見晴らしの良い場所における，日射量から推定された発電電流値は高い精度であることが示されている．ここでは提案したモデルによる精度を検証するために，曇りの日に見晴らしの良い場所での測定と晴れの日に見晴らしの悪いバルコニーでの測定を行った．晴れの日のバルコニーでの実測と測定値の比較は図 8 の 2 本の赤線に示した．この二つの値はおおむね一致するものの，実測値は大きく変動し，1 時間おきの日射量の合計値からでは，この変動の大きさを推定することが難しいことがわかった．曇りの日の見晴らしの良い場所での実測と推定地の比較は図 8 の 2 本の青線に示した．午前 10 時頃までは高い精度で推定できていることが分かるが，それ以降は誤差が大きくなり，午前 11 時 30 分を過ぎた辺りから実測される発電量は急激に小さくなっていることが分かる．これはバルコニーが南東の方角を向いているため，太陽が南の方角に移動すると直射日光が太陽光パ

表 1　シミュレーション条件

環境モデル	エネルギーモデル		2 月頃の日射量を想定して，最大値 0.6 kW/m² の正弦波関数
	物理モデル		700 を初期値として，50/hour で単調減少する土壌水分量
アプリケーションモデル	ハーベスタモデル	規模	2.5 W
		効率	0.8
	ストレージモデル	容量	リチウムイオンポリマー電池 2000 mAh
		初期電池残量	0.2
		充電効率	0.99
	負荷モデル	MCU	Arduino UNO + Wi-Fi Shield
		負荷電力効率	0.9
		タスク 1	土壌水分量のセンシングを 10 秒に一回行う
		タスク 2	土壌水分量が閾値を下回るとポンプを 30 秒駆動し，土壌水分量を上昇させる
		タスク 3	タスク 2 を実行後に Wi-Fi Tx モードに移行し，ツイートを行う

ネルに到達しなくなることが原因であると考えられる．このように太陽光パネルの設置場所が発電量に大きく依存するため，気象庁のデータからの推定値では精度が不十分となってしまう．

提案したモデルにおける発電量は，時々刻々の日射量を定義可能であるため，直射日光が遮断される時間帯を考慮できるようなモデルに拡張することができる．

5.2　シミュレーション結果の有用性

環境発電源を太陽電池として動作する自動水遣り機について，シミュレーションを行った．アプリケーションの起動開始時間が午前 8 時として，表 1 に示した条件のもとでシミュレーションを実行した．

シミュレータによって，出力されたグラフを図 9 に示す．上段のグラフは，待機電力，各タスクによる消費電力，ハーベスタによる発電電力，そして電池残量を示している．図 9 の下段に現れているグラフは単調減少として定義した物理量である土壌水分量を示している．上段の図と比較すればわかるように，ポンプを稼働させるたびに土壌水分量が 200 上昇し，再び減少し始めるといった時間変位を示す．これによって，モデル化された物理現象を表現できていることがわかる．乾燥を検知して水遣りを行う EH アプリケーションの電力収支をより高い精度で表現することが可能になったと言える．

そして，EH アプリケーション設計者にとって多くの場合，最大の関心事である，「現在の設計の下でアプリケーションが停止することなく常時稼働し続けられるか」という問いに対してもこのシミュレータは答えを提供する．図 9 上段より，表 1 に示した設計と条件では，翌日の午前 4 時を過ぎた頃に電池残量が尽きてしまうことが示唆されている．このシミュレータの出力グラフをもとに，より安定して稼働する EH アプリケーション設計のためにユーザに示される改良案の一つとして，ハーベスタの規模を大きくするというものがあるだろう．よって次に，表 1 に示したシミュレーションの入力変数のうち，太陽電池のスケールのみを 2.5 W から 4 W へと変更した．シミュレーション結果のエネルギー収支グラフは図 10 となった．このシミュレーション結果より，太陽光パネルの定格出力が 4 W 以上であれば，停止することなく安定して長期間稼働するということを示している．このようなエネルギー収支の可視化は EH アプリケーション設計者に有意義な情報を与える．

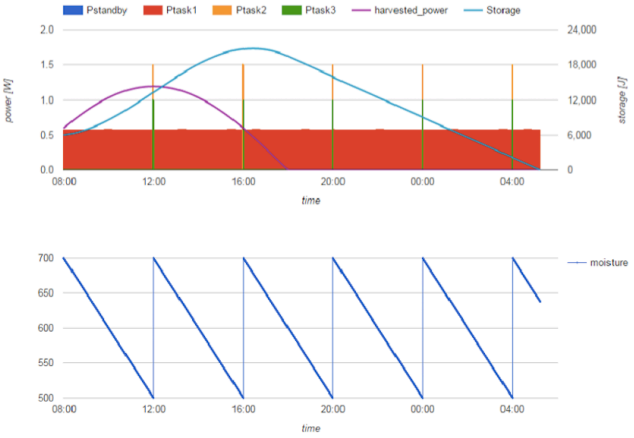


図 9　シミュレータの出力グラフ

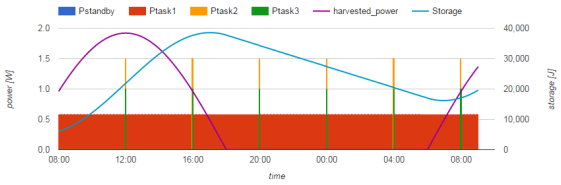


図 10　ハーベスタ規模を 4 W に変更した時のシミュレーション結果

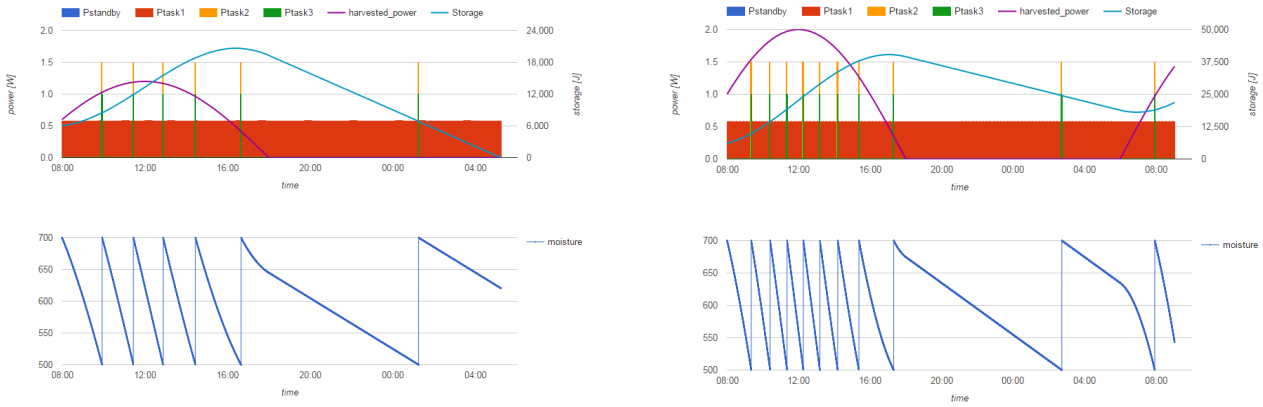
5.3　より高精度な物理モデルの再現

1 年を通じて EH アプリケーションを取り巻く環境は変化するため，安定した動作をしていた EH アプリケーションでも，季節が変化すれば動作が変化し停止してしまうこともあり得る．太陽電池を発電素子とした EH アプリケーションの場合，一見すると日射量の少ない冬で安定して動作するように設計すれば，日射量の多い夏でも安定して動作するように思えるが，そうとは言い切れない．なぜなら，季節の変化によって消費電力も変化し得るからである．例えば，乾燥を検知した時に水遣りを実行する自動水遣り機アプリケーションにおいては夏の方が土壌乾燥速度が大きいので，ポンプの駆動回数が増加すれば消費電力も増加するであろう．このような状況では，冬に安定して動作する EH 自動水やり機は夏にも安定して動作するとは断言できない．本シミュレータはこのような状況もシミュレートできる．

もともと土壌乾燥速度に関しては，ペンマン式より以下のように表されるという知見が得られている．

$$ET_{\mathrm{pen}} = \frac{\delta}{\delta+\gamma}\cdot\frac{S}{l} + \frac{\delta}{\delta+\gamma}\cdot f(u_2)(e_{\mathrm{sa}} + e_{\mathrm{a}}) \qquad (11)$$

ここで，ET_{pen} は，ペンマンの蒸発散位 [mm/day] であり，乾燥速度を表す．δ はある気温での法話蒸気圧曲線の



[a]2 月の日射量　　[b]8 月の日射量

図 11　日射量が考慮された土壌の乾燥速度のシミュレーション

勾配 [mbar/°C]，δ は乾湿計定数，S は純放射量，つまり日射量を表し，$f(u_2)(e_{\mathrm{sa}}+e_{\mathrm{a}})$ はダルトン型蒸発量推定式を表す．$\delta, \gamma, f(u_2)(e_{\mathrm{sa}}+e_{\mathrm{a}})$ を定数と近似した時，乾燥速度は日射量の一次式で表せる．よって，土壌水分量を M として，適当な定数 a, b を用いて，

$$-\frac{d}{dt}M = a \cdot P_e + b \tag{12}$$

と近似できる．本シミュレータは，このような物理モデルも表現でき，実際にシミュレーションを行った．シミュレーション条件は，表 1 から，物理モデルのみ変更を加えた．物理モデルの土壌水分量を式 (12) における変数を，$a = 200[1/\mathrm{hour}], \mathrm{b} = 20[1/\mathrm{hour}]$ とおいて，シミュレーションを実行した．シミュレータによって出力されるグラフを図 11[a] に示す．下段の土壌水分量のグラフを見ると，確かに発電量 (日射量) が大きい時に減少速度が大きくなっており，式 (12) で表される物理現象がより忠実に表現されていることが分かる．次に夏の 8 月ごろの日射量を想定したシミュレーション結果を図 11[b] に示す．冬と比較してポンプの駆動回数は増加しているが，より安定して動作していることがこの結果から示唆された．

このように，物理モデルが，エネルギーモデルの関数になっている場合でも PLEH は適用可能である．

6. おわりに

本稿では，エネルギーハーベスティングを利用したアプリケーションの設計に役立つであろうエネルギー収支シミュレータの作成を行った．エネルギーハーベスティングが活用されるようなアプリケーションはどのようなものであるかを定義し，多くの EH アプリケーションを表現できるようなシミュレーションモデルを提案した．そして多くの EH アプリケーションの挙動はセンシング，駆動系，無線通信，モード切り替え，エネルギー調整の 5 種類のタスクの流れで表現でき，タスク自身が保持するトリガ情報によって実行タイミングが制御されることで，それらタスクの制御の流れも，多くの想定される EH アプリケーションにおいての表現が可能となった．特に，IoT で注目される EH アプリケーションは外部環境の情報に動作が強く依存することがあるので，アプリケーションの挙動のみならず，外部情報の定義もモデルに取り入れることでより拡張性の高いモデルとなった．PLEH が出力するシミュレーション結果の精度を検証するために，アプリケーションを実際に稼働させた時のエネルギー残量の時間推移と PLEH による推定値を比較する必要があるので，今後の課題としていきたいと思う．

7. 謝辞

本研究は JST ERATO の支援を受けて行われたものである．

参考文献

[1] Benedetti, D., Petrioli, C. and Spenza, D.: GreenCastalia: an energy-harvesting-enabled framework for the castalia simulator, *Proceedings of the 1st International Workshop on Energy Neutral Sensing Systems*, ACM, p. 7 (2013).

[2] Pimentel, D., Musilek, P. and Knight, A.: Energy harvesting simulation for automatic arctic monitoring stations, *Electric Power and Energy Conference (EPEC), 2010 IEEE*, IEEE, pp. 1–6 (2010).

[3] Merrett, G. V., Weddell, A. S., Lewis, A. P., Harris, N. R., Al-Hashimi, B. M. and White, N. M.: An empirical energy model for supercapacitor powered wireless sensor nodes, *Proceedings of 17th International Conference on Computer Communications and Networks, ICCCN'08*, IEEE, pp. 1–6 (2008).

[4] Merrett, G. V., White, N. M., Harris, N. R. and Al-Hashimi, B. M.: Energy-aware simulation for wireless sensor networks, *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. SECON'09*, IEEE, pp. 1–8 (2009).

[5] Didioui, A., Bernier, C., Morche, D. and Sentieys, O.: HarvWSNet: A co-simulation framework for energy harvesting wireless sensor networks, *International Confer-*

ence on Computing, Networking and Communications (ICNC), IEEE, pp. 808–812 (2013).

[6] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E. et al.: TinyOS: An operating system for sensor networks, *Ambient intelligence*, Springer, pp. 115–148 (2005).

[7] Levis, P., Lee, N., Welsh, M. and Culler, D.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications, *Proceedings of the 1st international conference on Embedded networked sensor systems*, ACM, pp. 126–137 (2003).

[8] Shnayder, V., Hempstead, M., Chen, B.-r., Allen, G. W. and Welsh, M.: Simulating the power consumption of large-scale sensor network applications, *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM, pp. 188–200 (2004).

[9] Hester, J., Scott, T. and Sorber, J.: Ekho: Realistic and repeatable experimentation for tiny energy-harvesting sensors, *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, ACM, pp. 330–331 (2014).

[10] Parrot: Parrot @ CES2016, Parrot (online), available from ⟨http://www.parrot.com/ces/⟩ (accessed 2016-2-27).

[11] Arduino: arduino_Uno_Rev3-02-TH.sch, Arduino (online), available from ⟨https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf⟩ (accessed 2016-2-27).

[12] Arduino: ArduinoWiFiShield101.sch, Parrot (online), available from ⟨https://www.arduino.cc/en/uploads/Main/Arduino-WiFi101-schematic.pdf⟩ (accessed 2016-2-27).

[13] 気象庁 Japan MeteorologicalAgency：気象庁—過去の気象データ・ダウンロード，気象庁（オンライン），入手先 ⟨http://www.data.jma.go.jp/gmd/risk/obsdl/#⟩（参照 2016-2-27）.

[14] Sparkfun: Sunny Buddy Solar Charger V13 Hookup Guide - learn.sparkfun.com, Sparkfun (online), available from ⟨https://learn.sparkfun.com/tutorials/sunny-buddy-solar-charger-v13-hookup-guide-?_ga=1.195742955.654149371.1435909155⟩ (accessed 2016-2-27).