太陽光発電を用いたIoTデバイスの動的な省電力手法

杉本 一彦[1,a)]　串田 高幸[1]

概要：外部電源に接続されていない IoT デバイスでは現状, 1 次電池を用いて駆動しているのが一般的である．しかし 1 次電池は使い切りの目的が前提となっており再充電が行えず電池残量がなくなった際は電池交換をしなければならず人的資源の確保やユーザへの負荷が大きくなってしまう．その問題を解決するために本研究では太陽光発電と 2 次電池を用いた IoT デバイスの電力管理手法を提案する．この手法では, 発電量や 2 次電池の残量, デバイスの消費電力から動作を継続的に持続させるためにスリープやネットワークへの通信を自動的に制御する．また太陽光パネルを用いることで発生するユーザリクエストの処理に関する問題についても改善の提案を行い, その合意手法を定める事で消費電力を下げることへと繋げる事ができた．そして今後の発電量を無料で利用できる OpenWeatherAPI の天気予報データを元に予測する手法についても提案した．その結果, ユーザへ精度の高い今後のデータの送信可否確率を提示することができ, ユーザとのリクエスト数の調整や, 基礎実験としての消費電力制御が従来のシステムに比べ精度の高い自動化をすることが可能となった．

1. はじめに

昨今，インターネットに接続された多数のマシン (Internet of Things) は，日常生活をより便利にし，有益なサービスを提供するのに役立つ膨大な量のデータを生成している [1].

背景

この論文では, 自然エネルギーの中の 1 つである太陽光へ焦点を当て, IoT におけるエネルギーを根本から解決することを目的としている．太陽光発電は, 基本的にその発電量は時間や場所の周りの環境に依存をする．エネルギー源の性質, エネルギーがランダムな時間に任意の量で変化するため, 収集されたエネルギー量は予測ができない可能性が高い [2]．太陽光発電を利用する上で自らがどれほどの電力を消費しているのか，どのくらい発電しているのか，残りのバッテリー残量はどのくらいなのか, その変化に合わせて対応する必要がある．

課題

課題として 2 つ述べる．1 点目は IoT デバイスの電力供給についてである．IoT デバイスを動作されるためには原則として, コンセントを含んだ AC 電源へ接続されている必要がある [3]．AC 電源へ接続することは, IoT デバイスに対して安定して継続した電力を常に供給することが可能であるからである．IoT デバイスは安定した電力供給がなされない限り, 動作することができない．また, 大半の IoT デバイスは常にネットワークに接続されているために, 消費電力も小さいものにはならない．例として, 一般的なマイクロコンピュータとしてあげられる Wifi 接続時の RaspberryPi は平均消費電力が約 2.6W であり, マイクロコントローラーである ESP32 の Wifi 接続時の平均消費電力は約 1.0W である [4]．これは約 5000mAh の容量があるリチウムイオン二次電池を用いて駆動させた場合, RaspberryPi であれば約 10 時間, ESP32 であれば約 25 時間で容量を完全に使い切る結果となる．

IoT デバイスは小型な装置であるが, 消費電力は決して小さいものではないため, もし AC 電源に接続されていない場合に AC 電源の代用として安定した電力を供給できる装置が限られてくることになる．また, AC 電源は屋内であれば大半が接続可能な範囲になるのが一般的であるが, 屋外は AC 電源への接続ができない範囲がほとんどである．屋外に IoT デバイスを設置する場合, AC 電源を設置地点まで敷設をする必要があり, IoT デバイス 1 つを設置するコストとしては非常に大きなものとなってしまう．IoT の発展に伴ってより多様なデータが必要になってきており, センサーを設置するための IoT デバイスの役割は大きなものとなってきている．場所をなるべく選ばすに自由に IoT デバイスを設置し, センサーデータの収集を助長するため

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[a)] C0117164

にもこの電力供給の問題は大きなものとなっている [5].

また AC 電源の代用として大容量なバッテリーの設置がある．しかし, 大容量の 2 次電池を取り付けたとしてもいずれ電池は底をつき, IoT デバイスは停止してしまう．その際の電池交換は人の手で行う必要があるが, IoT デバイスの設置場所が人が容易に立ち入れる場所ではない海上や山奥となると, この電池交換にもかなりのコストと負荷がかかることになり, コストパフォーマンスという面からしてもデメリットが目立ってしまう．

そこで AC 電源の接続なしに安定して IoT デバイスを動作させようとすることを目的として本論文にて太陽光パネルを用いた IoT デバイスの駆動手法を提案する．

しかし太陽光パネルを用いることに関しても問題を孕んでいる．消費電力の制御についてである．太陽光パネルからの発電量は不規則であり, この発電量は電力管理において無視できないものである．太陽光から発電される電力量にシステムの利用できるエネルギー量は完全に依存するため, いくら発電されているのか監視をする必要性がある．またシステムに必要な量を常に提供されることは望まれない．これは太陽光を用いるという特性上, 太陽光が太陽光パネルに照射している期間しか発電することができないからである．そのため夜間は当然発電することは叶わず, 更には日中であっても天気や曇の陰りによって大きく発電量は左右される [6].

また電力管理は太陽光発電のみならず, 2 次電池 (バッテリー) や消費電力を考慮する必要もある．太陽光パネルからどれだけの電力が生み出され, その電力をどのように効率よく消費し, 2 次電池のバッテリー残量をどのように保持をするのかの対応がなされない限り, 太陽光パネルを用いた IoT デバイスのシステム管理はできないためこれは大きな課題となる [7].

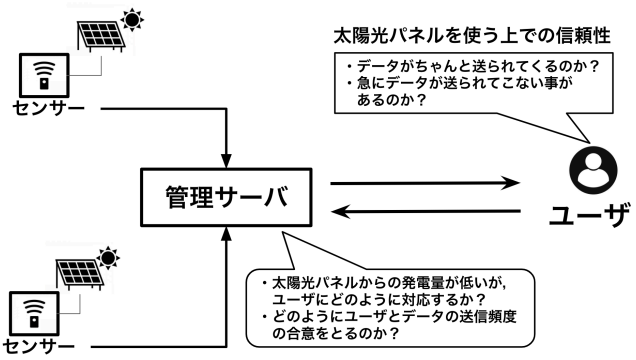

図 1 ユーザインタラクションの課題

2 点目は, ユーザリクエストに対するレスポンスについてである．図 1 はユーザインタラクションについての課題を示している．図 1 では, センサーと管理サーバは Wifi や Bluetooth を含めてネットワークや近距離無線通信で互いに接続されている．またセンサーには小型の IoT デバイスが設置されており, センサーデータは IoT デバイスを介してサーバに送られる．また電力供給は AC 電源ではなく, 太陽光パネルへ接続されており 2 次電池に蓄電された後, IoT デバイスへ電力供給される．ユーザはセンサーデータを集約している管理サーバが持っているウェブサイトへアクセスすることで, センサーデータへアクセスすることやセンサーデータの取得のリクエストをすることができる．

そして, ここで課題となっていることが図 1 にも記述があるように, ユーザ視点から”太陽光パネルを使う上での信頼性”である．これは太陽光発電で IoT デバイスを動作させているため, ユーザからは「IoT デバイスが電池不足でデータが送られてこない可能性」や「急にデータが送られなくなる可能性」といったユーザにとって不安要素となる点がいくつか存在する．これはユーザにとってはサービスを利用する上で, センサーデータが得られない可能性があるという事象は大きな障壁となる．

また管理サーバ側にとっても, 太陽光発電での上記の課題は利用する上でも大きなデメリットになる．図 1 にある通り, 太陽光パネルからの発電量が低いのは認識しているが, ユーザに対する対応についてや, 初期の段階でデータの送信頻度を含めたリクエストの調整をユーザと行う事が非常に大きな障壁になる．

また, IoT デバイスがスリープ中の場合はサーバとの接続が取れておらず, サーバがユーザーよりリクエストを受け取ったとしても即時に IoT デバイスにリクエストを送信することができない．また, 太陽光発電に電力供給を頼っている場合, 1 点目の課題のように消費電力の制御がなされていないと電力不足で IoT デバイスが停止してしまう．そのためリクエストが処理されずリジェクトやエラーをユーザに返してしまう可能性がある．これはユーザへの意図とは反している処理結果となってしまうため, ユーザ側のシステムでエラーやデータ処理ができないといった問題が発生してしまう危険性もある．しかし, 消費電力を制御することで必ずしも太陽光発電の課題を解決することには至らない．ユーザリクエストの量が単一時間当たりに多数である場合, 供給量が消費量を総合的に上回ってしまうと最終的に電力不足で IoT デバイスが停止することになる．そのため, 消費電力を下げるために一時的にユーザリクエストの見送りをする場合も否めない．その際にどのようにユーザへインタラクションを行い, ユーザから合意を得られるかが課題となる．

各章の概要

2 章では本研究に関連した先行研究について述べる．3 章では本研究の提案について述べる．4 章では, 提案した手法について実機を用いた実装について述べる．5 章では実装と評価について述べ，6 章ではこの論文で述べてきたことに対する議論を行う．7 章では今後の研究の課題と方

向性について述べる．

2. 関連研究

国連気候変動会議（締約国会議 21）より，様々な再生可能エネルギープラントが従来の化石燃料ベースの発電プラントから世界的に開発されてきた．しかしそのような再生可能エネルギー源は計画されたスケジュールで動作することは困難であり，天候などの予測できない環境条件のため出力が不安定になる．継続的な発電状況の情報を収集・分析・対応することで，より安定的に発電システムを管理することができ、蓄積されたデータは、将来の発電予測や最適なメンテナンスに有利になる．上記を考慮したオープン IoT プラットフォームを備えた LoRa ベースの再生可能エネルギー監視システムを提案する [8]．提案された監視システムは、実装の容易さ、開発コストの削減、アプリケーションの多様性により、将来のエネルギー IoT システムに適用できるとしている．また、低電力長距離ネットワークをサポートする LoRa は、Telco の基地局なしの低コストソリューションを通じて適用される．実装では Arduino や RaspberryPi を用いてオープン IoT プラットフォームを利用して実用性を示している．

太陽光発電を用いた IoT によるアプローチでは，公共の天気予報に基づく IoT デバイスでの太陽エネルギー予測がある [9]．リソースに制約のある IoT デバイスにてパフォーマンスを低下させることなく動作させるために、太陽光発電は IoT の多くのシナリオで重要であり，太陽エネルギーの予測は資源の効率的な管理と利用に必要としている．機械学習はすでに大規模発電所の太陽光発電を予測するために使用されているが，簡単に入手できる公共の気象データに基づいてさまざまな機械学習方法を使用する方法について調べている．

IoT デバイスのエネルギー問題に関するアプローチでは，エネルギーハーベスティングセンシングでバッテリーを再考する論文がある [10]．非充電式バッテリーを用いて IoT デバイスを最初は動作させていた．しかし合理的なサイズの一次電池で達成可能なセンサノードの寿命が短いため、動作を維持するためにノードのために頻繁に電池を交換する必要がある．その解決策としてエネルギーハーベスティングに研究者は目を付けた．バッテリー予測問題については、固定ラウンドロビンアクセス制御ポリシーを採用し、RL ベースのアルゴリズムを開発して、エネルギー源に関するモデル知識がなくても予測損失（エラー）を最小限に抑える [11]．

IoT デバイスの省電力化について, 処理の増加による消費電力増加を抑えるため処理の分散手法を述べている [12]．太陽光発電で稼働する場合において, 消費電力が増加した場合には発電量で対処ができないため負荷を分散させることが重要であり, 本研究と合わせて利用することができる．

バッテリー依存を抑えるためにソーラーハーベスティング (太陽光発電) を用いて駆動させることを目的としている論文である [13]．太陽光発電を用いたものであるが, 恒久的な動作時間の延長ではなくあくまで動作時間の延長としており, 長期間動作としての電力管理は提案されていない．また発電量の変化には言及しているが IoT デバイスが停止した場合について考慮されていない．対して本研究ではエンドユーザに対しても焦点を当てている．

3. 提案

本研究では, IoT デバイスを太陽光発電と 2 次電池で動作させるために太陽光パネルの発電量や 2 次電池の残量を自動的に測定し, さらに WeatherAPI から取得できる天気予報を用いた動的な動作スケジューリングによる消費電力制御手法と, この消費電力の制御手法に伴って発生する課題解決のためにユーザへのインタラクションを改善する手法を提案する．

3.1 システム/ハードウェア構成

まず全体のシステムとハードウェアの構成を図 2 へ示す．なおより詳細な内容は 3.1.2 以降へ記載している．まずハードウェアの全体の構成では要素は 3 つある．「ユーザ」「サーバ」,「IoT デバイス」である．ユーザはデベロッパーではなくシステムを利用するエンドユーザのことを意味している．サーバと IoT デバイスはネットワークで接続されており, HTTP 通信にてデータの送受信を行っている．なお, サーバと IoT デバイスは常に通信を行っているわけではなく, むしろ接続をしていない時間の方が多い．これは IoT デバイス側で消費電力を抑えるために通信を極力行わないからであり, データの送受信やスリープ時間やユーザからのデータ送信リクエストの更新を行わない限り接続はされていない．

IoT デバイスは太陽光パネルや 2 次電池を含んだモジュールから電力を供給されている．このモジュールは物理的な電池回路で接続されている．

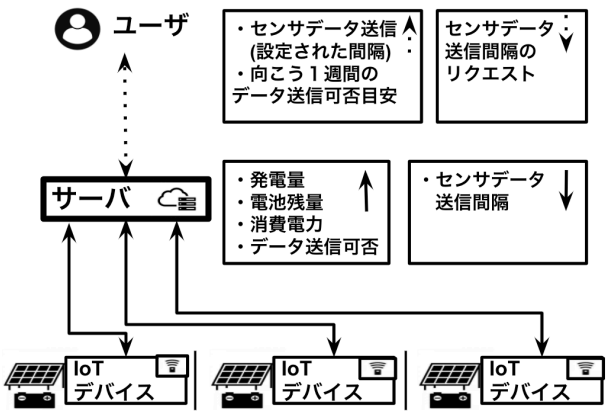

図 2　全体アーキテクチャ図

図 2 ではユーザとサーバ, サーバと IoT デバイスのアップストリームとダウンストリームがそれぞれ矢印を用いて示されている．ユーザとサーバの繋がりは波線にて, サーバと IoT デバイスとの繋がりは実線にて表されている．

まずユーザとサーバ間では, ユーザからサーバへセンサーデータの送信間隔をサーバへリクエスト (送信) する．サーバからユーザへはユーザから設定されたタイミングで指定されたセンサーのデータを送信する．またそれに合わせて, WeatherAPI から取得された天気予報のデータを元に 5 日後までの送信ができる可能性を確率にした送信可否の目安というのものを合わせてユーザに送信する．

そしてサーバから IoT デバイスへはユーザから設定されたセンサーデータの送信間隔を反映する．そして IoT デバイスはその設定に合わせてサーバにセンサーデータの送信を実施する．また, ユーザのリクエストで送信するセンサーデータとは別に, 一定の間隔でサーバへ太陽光パネルの発電量, 2 次電池の残量 (電源電圧), IoT デバイスの消費電力 (現在の CPU 周波数, 動作モード, スリープ時間, データシートからの定数的な消費電力量) が送信され, サーバのデータベースへ格納されていく．

次にシステム構成図を図 3 に示す．図 3 はサーバと IoT デバイス内のシステム要素が示されており, 動作の一連の流れとデータのやりとりが表されている．まずユーザはサーバ側が設けているブラウザから IoT デバイスからデータの送信する間隔の設定を行う．そして, サーバ側で保存をした後 IoT デバイスへとその間隔が送信されデータの送信処理が始まる．また IoT デバイスは単一ではなく, 複数個存在しており, それはサーバの消費電力計算で処理されている．ここでは, IoT デバイスの消費電力, 太陽光パネルからの発電量, 電池残量を IoT デバイスごとに管理をしている．サーバと IoT デバイス内で行う詳細の処理については, 下記に示す．

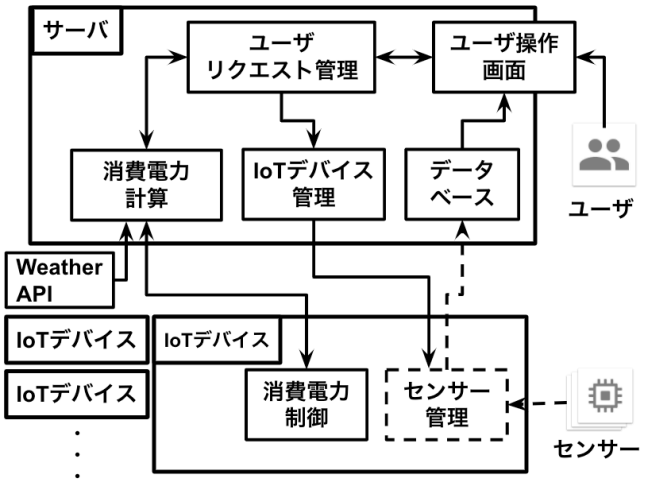

図 3 ソフトウェア構成図

3.1.1 サーバ

サーバ側の処理は大きく 2 つに分けられる．「消費電力制御のための計算処理」と「ユーザリクエスト管理」である．消費電力制御のための計算処理は図 3 ではサーバ内の”消費電力計算”と”IoT デバイス管理”に当たる．ここでは, IoT デバイスから送られてくる太陽光パネルでの発電量と 2 次電池の電源電圧, IoT デバイスの消費電力と WeatherAPI から取得できる天気予報を元に今後, IoT デバイスがどれほどの間動作できるのか期間を計算し, その期間に合わせて IoT デバイスの消費電力の制御を行う．この具体的な計算方法と消費電力に制御方法については 3.1.2 にて記述する．

そしてユーザリクエスト管理については, 図 3 中では”ユーザ操作画面”, ”ユーザリクエスト管理”に当たる．ユーザはサーバへリクエストの送信をすることになるが, その操作はサーバが用意しているウェブにて行う．それが”ユーザ操作画面”である．ユーザはこのウェブにアクセスすることでデータの取得やリクエストの設定が可能である．そして, ユーザ操作画面から入力されたリクエストは”ユーザリクエスト管理”へと送られる．ここではユーザリクエストの保存と管理を行っており, 上記の消費電力計算や IoT デバイス管理へと反映される．またこのリクエストの IoT デバイスへの反映は”IoT デバイス管理”にて実行される．IoT デバイス管理は複数の IoT デバイスと接続されており, ここで各々の IoT デバイスへのユーザリクエストの設定や管理をしている．

3.1.2 IoT デバイス

IoT デバイス内の”消費電力制御”はサーバ内の”消費電力計算”から指示を受けた, スリープ時間の設定や動作スケジュールの変更を IoT デバイスへ実行するソフトウェアである．ここでの詳細についても 3.1.2 で記述する．また, ”センサー管理”はセンサーデータの取得と, 取得したデータをサーバへ送信する役割を持っている．これは既存のシステムにも存在しているものである．

3.2 消費電力制御

電力管理として太陽光パネルからの発電量を監視し, それに合わせた動作を決定することが重要なポイントとなる．太陽光パネルで発電された電力は一度 2 次電池へ貯められ IoT デバイスへと送られる．監視する電力は太陽光パネルでの発電量と併せて 2 次電池から供給される電圧である．電力の監視は, IoT デバイスがスリープではないアクティブ (CPU への電力供給がなされている) な状態でなければならず, また電力を測定するためにも電力を消費するため, 常時監視することは望ましくない．そのため測定するタイミングはアクティブ中のみとし，間隔は 1 分間隔とした．スリープ中にデータ取得を行わないのは, センサーからデータを取得するためにスリープから起き上がる必要

があるが, この際に電力を大きく消費してしまう．そのためスリープは極力長期間になるように設定をする．なおスリープ中に発電された電力量については, スリープ前とスリープ後に取得する 2 次電池の出力電圧から求め出す．なおスリープ中の発電量の過程については不明であるが, スリープ前とスリープ後に測定されたデータから発電された量は推定で求め出す事が可能である．

2 次電池の電源電圧から残りの残量を求め出す手法はいくつか存在しているが, 電圧が実際の静電容量と比例するものではないため定数を与えて測定することはできない．そのため本研究では, 2 次電池の電圧の降下特性を回帰によって求め出すことで実際の残量に近い数値を求め出すことができた．これは 2 次電池を最大まで充電した後, 同じ負荷をかけながら放電を続け, IoT デバイスが起動できる最低電圧まで下げ, 時間と降下する電圧の関係をグラフに表すことができる．そのグラフに対して近似をしていく事で $R^2 = 0.99$ ほどの回帰関数を作る事ができる．その関数を用いることで電源電圧から残量を求め出す事が可能である．ここでの具体的な手法や計算については第 4 章にて示す．

太陽光パネルからの発電量は分単位の間隔で変化するため, ワット分 (Wm) で計算を行う．これに合わせて, 2 次電池の残量についても, 消費電力についても同様にワット分で計算を行うものとする．今後の発電量の計算の目安として, WeatherAPI を用いるがまず取得できる天気予報の情報の一部を下記に示す．

- 天気予報日時
- 天気 (晴れ, 曇り, 雨, 雪)
- 気温
- 湿度
- 気圧
- 雲量

上記の情報で今後の発電量を予想する上で重要になってくる要素は 3 つあり, 「天気予報日時」と「天気」, 「雲量」である．太陽光パネルからの発電量のデータは IoT デバイスが一定の間隔で取得するが, 同時にサーバ側でも WeatherAPI から常に上記のデータを取得している．図 3 での消費電力計算では, WeatherAPI のデータと実際の太陽光パネルの発電量のデータに関係性をつける処理をする．そのようにすることで, 雲量や天気を元に発電量を目安として求め出す事が可能となる．

そして, 残りの動作可能時間については上記で求めた太陽光パネルでの発電量, 2 次電池の残量, IoT デバイスの消費電力の 3 つの要素を基に求め出す．

まず 2 次電池の残量は先述した回帰関数から求め出す事が可能となる．ここでの回帰関数を $f(y)$ とする．$f(y)$ での y は現在の 2 次電池の電源電圧とする．$f(y)$ は縦軸が電圧, 横軸が時間の関数である．$f(y)$ に現在の電圧を y として代入をして,

$$\frac{f(y)}{maxtime[h]}$$

とすることで, 残りの 2 次電池の残量比率を出す事ができる．なお $maxtime[h]$ は先述した 2 次電池を最大まで充電した後, IoT デバイスが起動できる最低電圧まで下がるまでの経過時間のことを指す．また 2 次電池の最大容量を $Y[Am]$ とする．基本的に 2 次電池の容量はアンペア時 (Ah) であるため, 下記にて単位換算を行う．

$$Y[Am] = Y[Ah] \times 24[hour(1day)]$$

この場合, 2 次電池の残量を $remaningpower[Am]$ の計算式は以下のようになる．

$$remainingpower[Am] = \frac{Y[Am] \times f(y)}{maxtime[h]}$$

また, IoT デバイスの消費電力を $consamption[Am]$ 動作可能時間を $remainingtime[m]$ とすると, 計算式は以下のようになる．

$$remainingtime[m] = \frac{remainingpower[Am]}{consamption[Am]}$$

上記の計算手法については第 6 章にて実験結果として後述する．

太陽光パネルからの発電量予測については, WeatherAPI から取得できる雲量の数値を用いて予測を行う．太陽光パネルの発電量は直射日光が当たるか当たらないかで大きく変化する [14]．そのため, 天気の分類である「晴れ」や「曇り」では実際の空に存在する雲の量に明確性がなく, 直射日光がどれほどの割合で遮られてしまうかが不鮮明になってしまうため, 天気よりもより雲の状況がわかる雲量の数値を用いる．WeatherAPI から取れる雲量の数値は 0(快晴)〜100(そら一面に雲がかかっている状態) まであり, この数値を 5 ごとに区切り, 太陽光パネルからの発電量とで比較をしていく．太陽光パネルからの発電量は短時間でデータ取得を行うが, 1 時間ごとに区切り各々のデータの平均値をその 1 時間の発電量とする．そして, その 1 時間の発電量と WeatherAPI から取得した雲量と紐付けを行う．例として, 雲量 10 である時の 1 時間の発電量と平均が $300[mAh]$ であれば, ”雲量:10 = 発電量:300” という様に紐付けを行う．この雲量と発電量の関係を関数として扱う．

表 3.2 は WeatherAPI から取得できるデータの例をリストとして表示されている．

表 3.2 は表 3.2 の雲量に対して発電量のデータを紐付けた際にどのようになるかを示したものとなっている．表 3.2 中の雲量が 88, 7 となっている列があるが, 5 で割り切れない数値である場合は数値の繰り上げを行い, 88 であれば 90, 7 であれば 10 といった様に次に高い雲量データに当てはめた発電量を用いるものとする．

表 1　WeatherAPI からのデータ取得の例

日時	天気	雲量
2021-01-17 12:00:00	曇り	95
2021-01-17 13:00:00	曇り	88
2021-01-17 14:00:00	晴れ	10
2021-01-17 15:00:00	晴れ	10
2021-01-17 16:00:00	晴れ	7

表 2　WeatherAPI からのデータ取得の例

日時	天気	雲量	発電量
2021-01-17 12:00:00	曇り	95	雲量 95 に紐付けられている発電量
2021-01-17 13:00:00	曇り	88	雲量 90 に紐付けられている発電量
2021-01-17 14:00:00	晴れ	10	雲量 10 に紐付けられている発電量
2021-01-17 15:00:00	晴れ	10	雲量 10 に紐付けられている発電量
2021-01-17 16:00:00	晴れ	7	雲量 10 に紐付けられている発電量

発電量はワット時 (Wh) で計算するため, この場合に限っては消費電力や 2 次電池の残量も同様にワット時 (Wh) で計算を行う．1 時間当たりの消費電力を $consamption[Wh]$ とし, 発電量を $genaratedpower[Wh]$, 2 次電池の残量を $remainingpower[Wh]$ とすると, 下記の計算式で 1 時間ごとの 2 次電池残量が求め出せる．

$$remainingpower[Wh] = generatedpower[Wh] - consamption[Wh]$$

同様の処理を各 1 時間ごとに得られる WeatherAPI のデータに当てはめていき, 今後の発電量の予測を行う．

3.3　ユーザインタラクション

IoT デバイスからのデータを送信する処理はユーザが要求する時間や時間間隔によって決定される．ユーザが要求する内容を本研究ではユーザリクエストと示している．ユーザリクエストは直接 IoT デバイスに設定はなされず, 一度 IoT デバイスを管理しているサーバへ設定される．その後, IoT デバイスにデータ送信処理を行う時間や時間間隔を設定する．ユーザが求めているセンサーデータは設定された時間を迎えると自動的にインターネットを介してデータベースへと保存される．データベースへ保存された後, ユーザへデータの送信が行われる．これが通常のデータの流れになる．

ここで重要になるのは太陽光発電を用いるためにユーザに利用するシステムに対して信頼性を持ってもらわなければならないという点である．つまり, ユーザからは「発電量が少ない時にデータが急に送られてこなくなるのではないか」や「どうやってユーザからのリクエストの守る保証があるのか」といった信頼性を問われることである．システムを利用するユーザの最大の目的は「リクエストしたデータの取得」である．ユーザに対してシステム側からリクエストに対してしっかりとレスポンスを返すという確証を与え, かつ発電量の問題をユーザへ同意してもらい, その上でシステムを利用してもらう必要がある．

そのために本研究では, ユーザに対して 3.1.2 にて提案した天気予報を用いて, 5 日後までのデータが送る事ができるか否かを確率的に示したものをユーザへ提示することで解決する．

ユーザは, まずリクエストを受け付けるサーバに対して, どのセンサーからどの頻度でデータを送信するかを設定する．サーバはそのリクエストを受け取った後, 3.1.2 で提案した手法を用いて IoT デバイスの現在の動作可能時間にユーザリクエストを反映して計算をし, 設定されたリクエストで IoT デバイスがどれほどの確率でセンサーデータを送信できるかを 5 日後先まで計算し, ユーザへ提示する．

表 3　リクエスト

センサー	BME280
送信頻度	10 分毎に 1 回

表 4　レスポンス

センサーデータ	
BME280-温度	12.3 ℃
BME280-湿度	23.4 %
BME280-気圧	1034.1 hP
データ送信可能確率	
1 日後	100%
2 日後	100%
3 日後	90%
4 日後	80%
5 日後	70%

表 3.3 はユーザのリクエスト内容の例である．表 3.3 の場合ではユーザはセンサーデータの送信頻度を 10 分毎に 1 回としている．そのため, ユーザの元へサーバは 10 分に 1 回のペースでセンサーデータを送信することになる．なお, センサーとして選択している BME280 は温度, 湿度, 気圧の 3 つのデータを同時に取得可能なセンサーモジュールである．

表 3.3 は表 3.3 のリクエストの内容に対するサーバからのレスポンスの例である．レスポンスは大きくセンサーデータとデータ送信可能確率に分けられる．まずリクエストの直接的なレスポンスであるセンサーデータはユーザが定めた 10 分に 1 回送信するというリクエストに合わせてユーザへデータが送信される．BME280 を例として用いているため, 今回届くセンサーデータの内容は温度と湿度, 気圧の 3 つになる．そして, データ送信可能確率はセンサーデータの情報とは別にデータが今後送信できるかどうかの確率を示している．この確率はサーバ側で自動的に集約される各 IoT デバイスの発電量や 2 次電池のレベル, 天気予報を考慮し自動的に数値が設定される．

3.3 ではデータ送信可能確率を 100% ～ 0%まで 25%区

表 5　データ送信確率

データ送信可能確率	意味
100%	必ず送信する事が可能 (発電量が 0 であっても動作可能)
100% ～ 75%	送信できる可能性が非常に高い
75% ～ 50%	天気予報通りの発電量が得られなかった場合, 電力不足になる可能性がある
50% ～ 25%	送信できる可能性が低い (予想発電量より高い発電量でなければ 電力不足で停止する可能性がある)
25% ～ 0%	リクエスト数を下げなければ 電力不足で停止する可能性が非常に高い
0%	リクエスト数を下げなければ 電力不足で停止する

切りにして目安となる意味の説明をしている．それぞれの数値は WeatherAPI から取得された天気予報に基づいて, 3.1.1 で述べた手法を用いて発電量を予め求め出し, その上でユーザのリクエストを満たせるかを示したものである．100%は万が一発電量 0 になったとしても残っている 2 次電池の電力で動作可能であるという意味であり, 逆に 0%はもし発電できる最大量を充電できたとしても残っている 2 次電池の残量が少ないため, ユーザのリクエストを達成する事ができない事を示している．

なぜこのように確率的な表記になるかを説明する．WeatherAPI で取得できる天気予報を用いて発電量の算出を行うことは 3.1.1 で既に述べている．しかし, あくまで予想であり, 厳密に実際に発電される量を特定することが難しい．

図 3.3 は雲量によって発電される量にばらつきがあることを示した箱髭図である．後述にある本研究で用いた太陽光パネルでの雲量を区別して 1 時間測定した結果を分析した．これは雲量 0%, 30%, 40%, 60% , 90%, 100%と区別されている．例として, 雲量が 0%だとしても最大値と最小値には約 300mW ものの差が空いている．しかし第一四分数と第三四分数は非常に近い数値となっている．雲量 30%は最大値と最小値は雲量 0%と同様に約 300mW ものの差が出ており, また第一四分数と第三四分数にも約 200mW ほどの差が出ている．

このように雲量から特定するに当たって, 厳密な発電量を取得することができないため, 例えば雲量 30%だとしても考えられる最大値から万が一の最小値まで計算を行う必要性がある．そのため, 最大値で発電されれば IoT デバイスに支障が出ない場合でも, 万が一最小値での値での発電量になった場合, ユーザのリクエストを 100%満たせなくなる可能性が出てくる場合も想定される．

このようにユーザに対してより分かりやすい形にしてレスポンスを返す事で柔軟にリクエストの変更や調整を行う事が可能である．また, ある一定の数値を万が一下回った

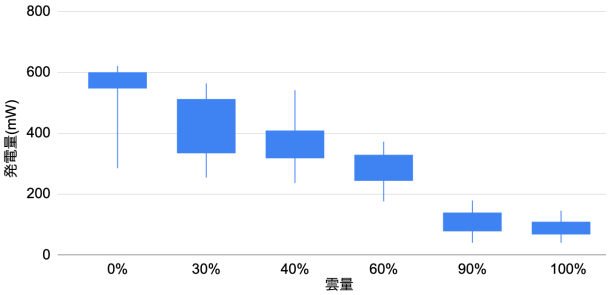

図 4　雲量による発電量のばらつき

場合に自動的にリクエストを調整することも可能である．

表 6　リクエスト調整前

センサー	
リクエスト数	毎時 100 回
日数	確率
1 日後	100%
2 日後	80%
3 日後	70%
4 日後	50%
5 日後	40%

表 7　リクエスト調整後

センサー	
リクエスト数	毎時 80 回
日数	確率
1 日後	100%
2 日後	90%
3 日後	80%
4 日後	60%
5 日後	50%

表 3.3 と表 3.3 はユーザリクエストを自動的に調整する前と後とを比較した表である．この場合, 例えば 50%を下回った場合に自動的に 50%以上へなるようにリクエストを変更する調整を行っている．これはシステムが自動的に毎時 100 回設定されていたリクエストを毎時 80 回に落とす事で解決する．ユーザが 1 つ 1 つセンサーの監視をする必要がなくなり, より負担を減らせることに繋がる．

このようにユーザへインタラクションをすることで, 太陽光発電で動作する IoT デバイスであったとしてもユーザから柔軟にリクエストの調整や, 今後の見立てが分かることで, より高い信頼性を得られることに繋がる．

4.　実装と実験環境

4.1　実装

ここでは提案も基にした実装について記述する．全体の構成図を図 5 に示す．

サーバは Ubuntu で構築されており, ユーザとのやり取りや, ESP32 の管理, WeatherAPI からのデータ取得の処理

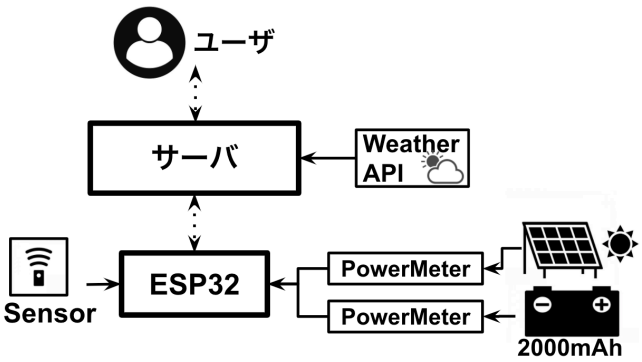

図 5　全体の実装構成図

を行う．またサーバ内にデータベースとして MongoDB を構築しており, 取得したセンサーデータや, 天気予報, ユーザからのリクエストを格納している．IoT デバイスとして本論文では ESP32 を用いる．ESP32 では, 物理的な電子回路でセンサーやバッテリー, 太陽光パネルと接続されている．太陽光パネルから発電された電力が電力計と太陽光パネルからの発電量は電力計にて発電量の監視を行っている．電力系モジュールセンサーとして INA226PRC を用いる．これは省電力かつ低抵抗で回路に影響を与えず 0.1mA の精度にて計測ができるため採用しています．また太陽光パネルからの発電量は電圧, 電流ともに大きく変動するため電圧レギュレータで一定の電圧に降圧をする．なお今回は IoT デバイスに ESP32 を用いるため定格電圧である 4.5V〜5.0V へ電圧を一定にする．そのため 2 次電池の定格電圧も 5V のものを扱うものとする．そして 2 次電池にて貯められた電力は DC/DC コンバータにて昇降圧をし, 電圧を一定に保つ．2 次電池はバッテリー残量により出力電力が変化する．電池が満たされている際は定格電圧で動作を続けるが, 残量が低下していくと共に次第に出力電力も下がっていく性質があるため, 2 次電池からの出力電圧は定格電圧よりも高いものを利用する．

図 6 はサーバと IoT デバイスの詳細な構成を示した図である．

Ubuntu のバージョンは 18.04 である．ユーザはサーバ内に設けられたブラウザにアクセスしリクエストを送れるようになっている．ブラウザで入力されたリクエストはサーバ内のリクエスト処理へ送られ, ここでリクエストの保存と ESP32 へのリクエストの設定が行われる．ESP32 のセンサー管理では接続されているセンサーの管理とセンサーデータの取得, 送信を行っている．センサーには BME280 を用いる．BME280 は I2C にて接続可能なモジュールであり, 温度, 湿度, 気圧の 3 種類のデータを取得できるセンサーである．本実験ではユーザからのリクエストを BME280 のデータの取得としている．ユーザが定められたタイミングで BME280 のセンサーデータをユーザへ送信するようになる．

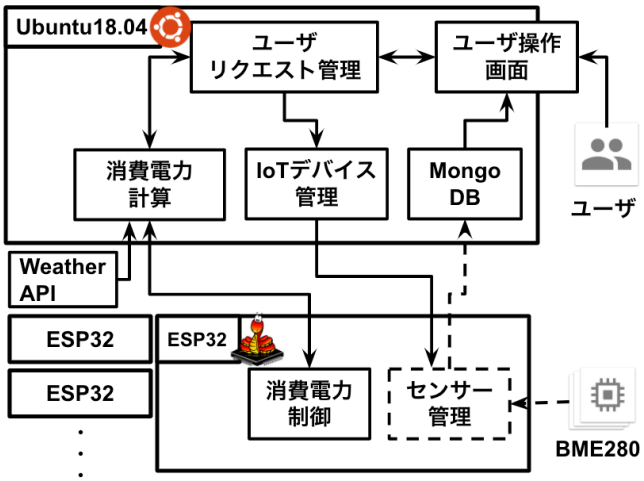

図 6　実装の詳細構成図

4.2　処理の手順

ユーザのリクエストの具体的な処理の手順を記す．まず全体の処理構成図を図 7 へ示す．処理手順は①〜⑥まである．詳細な説明は図 8〜図 10 へ示す．

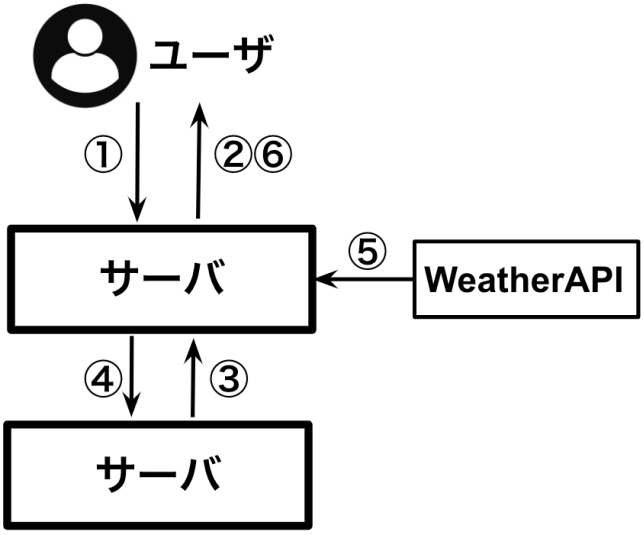

図 7　全体の処理手順構成

処理手順①, ②は 8 へ示す．まずサーバ側からユーザに対してリクエストをするよう受付処理をする．その後, ユーザは専用のブラウザからリクエストを設定する．ここでのリクエストというのは先述しているが, センサーデータの送信頻度の設定である．例として,「1 分間に 1 回」,「10 分間に 1 回」と言ったようなものである．ユーザから受け取ったリクエストはサーバに構築されている MongoDB へ格納される．

処理手順③, ④は図 9 へ示す．②にてユーザからのリクエストをサーバに設定したが, その設定を③にて ESP32 へ反映する．ESP32 は常時サーバに接続されているわけではないため, ESP32 へ設定が反映されるのは最短で ESP32 がネットワークに接続しサーバとコネクトが取れた際とな

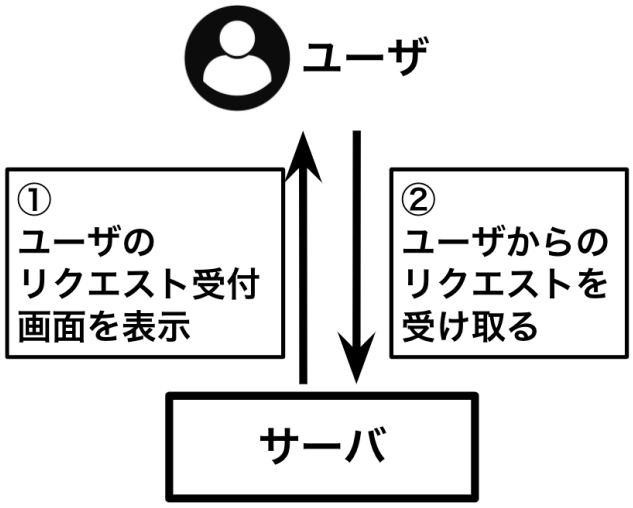

図 8　処理手順①, ②

る．またその際, 同時に④を実行する．④ではリクエストされたデータと共に, 太陽光パネルの発電量, バッテリーの残量をサーバに送信する．このようにすることでサーバ側で ESP32 の電源状況をそれぞれ把握することができる．

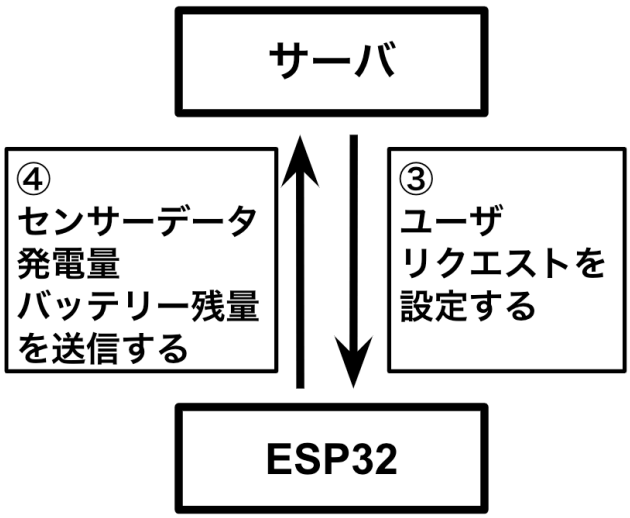

図 9　処理手順③, ④

処理手順⑤, ⑥は図 10 へ示す．サーバは④は ESP32 から電源状況について太陽光パネルの発電量とバッテリー残量のデータを受け取っている．⑤にて, Open Weather API を用いて向こう 1 週間の天気予報の取得を行う．この天気予報では向こう 5 日までは 3 時間ごと, 5 日〜7 日までは 2 4 時間ごとの天気 (晴れ，曇り，雨，雪) と雲量 (0%〜100%) のデータを取得する．この天気予報と ESP32 の電源状況のデータを紐付けする．そして⑥である向こう 1 週間のセンサーデータが送ることができる確率をユーザに表示する．

4.3　実験環境

ここでは実験した際のハードウェアとソフトウェアの環境や構成について記述する．

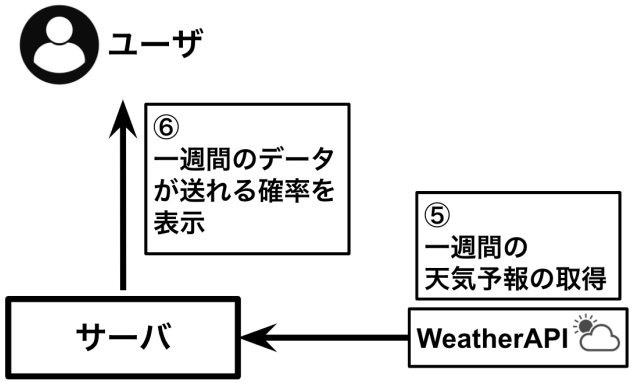

図 10　処理手順⑤, ⑥

4.3.1　ハードウェア

- IoT デバイス：ESP-WROOM-32
- 太陽光パネル：SY-M5W-12,
  寸法：188 × 251 × 17mm
- 電力計センサーモジュール：INA226PRC
- センサーモジュール：BME280
- 2 次電池：リチウムイオン電池 2000mAh

4.3.2　ソフトウェア

- ESP32：micropython1.3
- Management Server：Ubuntu18.04LTS
- Database：MongoDB4.5

本研究では, マイクロコントローラとして ESP32-WROOM-32 を利用した．ファームウェアとして micropython1.3 をインストールしている．micropython は Python3 というプログラミング言語をベースに開発されたものであり, マイクロコントローラーという制限された環境で動作することを目的として最適化されたインタプリターである．今回は汎用性が高い Python が利用できる観点から micropython にて実装を行う．サーバには Linux ディストリビューションの 1 つである Ubuntu を用いており, その中のデータベースとして MongoDB を用いている．本実験では複雑な構造のデータを扱うものはなく, 単純な数値データのみであるため NoSQL である MongoDB にて実装を行う．太陽光パネルは最大出力 5W の比較的大きがあるものを用いる．しかし先述している様に最大出力がでることは殆どなく, ESP32 を動作させることに対しては実際の発電量から適切なものであると考える．

5.　評価と分析

5.1　検証方法

本研究での検証は, 実際に屋外に太陽光パネル, リチウムイオン電池, ESP32 を取り付けた装置を設置し実測を行った．その様子は図 11 へ示す．

基礎実験では, 太陽光パネルの発電量やリチウムイオン電池の電圧の減衰特性や ESP32 の消費電力の実測を行い結

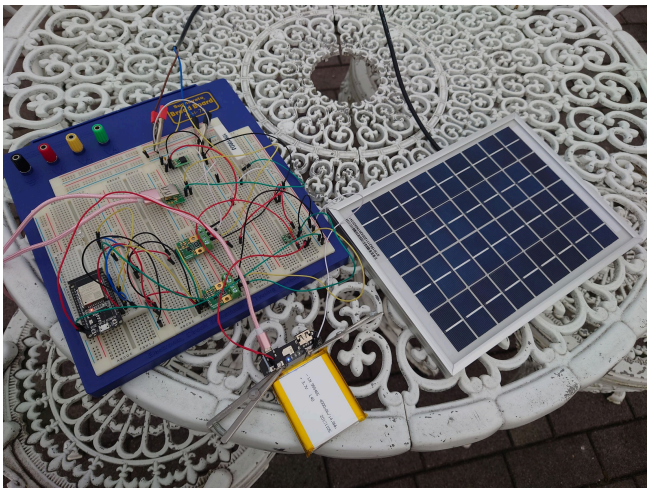

図 11　屋外での実験の様子

果を示す．太陽光パネルの発電量は晴れの日, 曇りの日, 直射日光が当たるか否かでの発電量の変化, 日没時ので発電量の変化の測定, 検証を行う．リチウムイオン電池の電圧の減衰に関しては, 実際に常に一定の負荷をかけ同一の電力を消費させることでその変化や関係性を検証する．ESP32 の消費電力は Wifi 接続時, Wifi 否接続時, Deepsleep 時, データ送信時, 起動時と分けて測定を行い検証を行う．

屋外での実測値とサーバで取得された OpenWeatherAPI からのデータを紐付けを実際に行う．そしてユーザへ第 3 章で述べた様に 5 日後までのデータ送信可否確率を算出し, 実際にその数値通りに稼働させることができたのか検証を行う．

5.2　評価・検証

ここでは検証の前段階である基礎実験について記述する．基礎実験では太陽光パネルの発電量と, ESP32 の消費電力について説明する．

5.2.1　太陽光パネルの発電量

実験で使用する太陽光パネルの寸法は 188 × 251 × 17mm である．図 12 と図 13 では発電量を示したグラフとなる．縦軸の電力の単位は mW である．図 12 は雲量 30%の晴れた日の 11:30〜13:30 での太陽光パネルの発電量をグラフで表している．この時間帯では 200mW〜500mW の間で発電量が変動している．発電量が 400mW〜500mW となる時間帯では太陽光が直接太陽光パネルに当たっている状態であるが, 200wW〜250mW になる時間では雲によって太陽光が遮断され陰りになっている．雲の陰りになることで直射日光である場合と比べて約 50%ほど発電量が減衰する．

図 13 では, 雲量 0%の晴れた日の 11:30〜17:00 での太陽光パネルの発電量の電圧をグラフで表している．なお日付は 12 月 22 日と 12 月 23 日である．図 13 と比べ, 雲の陰りがなかったため大きな発電量の差がない．16:00 以降で

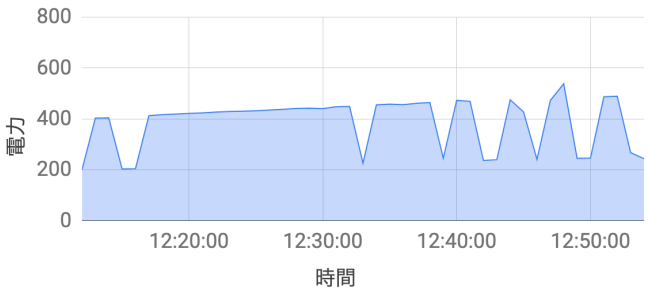

図 12　太陽光パネルでの発電量 (晴れの日，時間:11:30〜13:30)

は日没のため発電量が激しく減衰している．太陽光が直接当たり，かつ雲の陰りがない状態でこの太陽光パネルでは 500mW〜600mW ほどの発電能力がある．また図 13 では, 雲量 0%の晴れた日の 9:40〜17:00 での太陽光パネルの発電量をグラフで表している．図 13 と比べ, 雲の陰りがなかったため大きな発電量の差がない．16:00 以降では日没のため発電量が激しく減衰している．太陽光が直接当たり，かつ雲の陰りがない状態でこの太陽光パネルでは 500mW〜600mW ほどの発電能力がある．また日没前の 16:00 まではあまり発電量の減衰は見られなかった．太陽光パネルの発電量は日の出から徐々に発電量が上がっていき, 太陽が最も高度が高くなる 12 時が一番発電量が高くなり, 日没にかけて徐々に発電量が減衰していくという，12 時を中心に線形的に発電量が繊維するのが一般的である．しかし, 本基礎実験ではそのような変化は見られず太陽光が太陽光パネルに直射しているか否かで発電量が大きく変化していることが読み取れた．

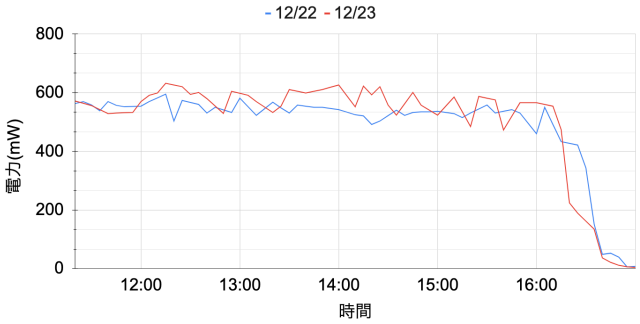

図 13　太陽光パネルでの発電量 (晴れの日，時間:11:20〜17:00)

また図 14 はある曇りの日の発電量のグラフである．8:50 からデータ計測を初めて 17:00 にて終了した．図 14 のグラフで特徴的な点は, まず 11:00 頃に一瞬ではあるが, 2 回ほど発電量が大幅に伸びた場面がある．この時の天気は雲量 90%の曇りであったがその時間だけ雲の切れ間から太陽光が直接太陽光パネルへ当たったのである．その変化の差は一目瞭然であり, 数値としては約 2.4 倍の増加である．同じような傾向が見られたのがもう 1 点有り, それが 16:00 直前である．16:00 直前にも一瞬だけ発電量が増加している時間があるが, これも 11:00 頃と同様にその時間だけ太

陽光が直接太陽光パネルに当たっていた．16:00 直前も直射日光が当たったことで発電量が約 2.5 倍増加した．

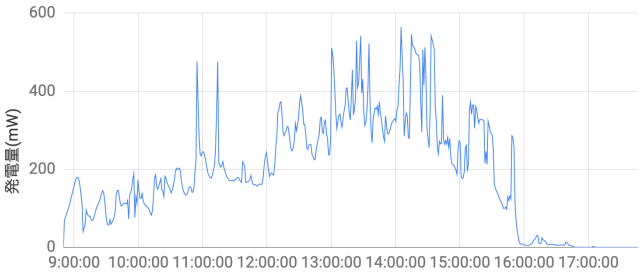

図 14　太陽光パネルでの発電量 (曇りの日，時間:8:50〜17:00)

図 15 は図 14 のグラフに同時間帯の OpenWeatherAPI から取得した雲量の数値を当てはめたグラフになる．赤い線で示しているのが雲量であり最大値は 100%，最小値は 0%となっている．測定開始の 8:50〜正午頃まで常に雲量は 90%であったが, 13:00 前〜14:00 前までの約 1 時間と 14:00 過ぎ〜15:00 前当たりまでは雲量が　40%と 45%まで下がっていることが分かる．またこの時に発電量が増加しており, 雲量が低下することで発電量が増加することが見て取れる．

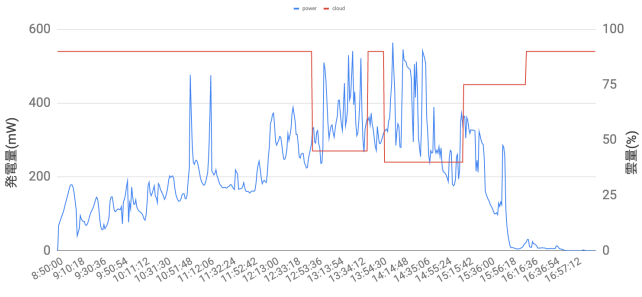

図 15　太陽光パネルでの発電量と OpenWeatherAPI の雲量の数値との比較

5.2.2　リチウムイオン電池

本研究では 4000mAh のリチウムイオン電池を利用する．図 16 はリチウムイオン電池を最大まで充電した後, 100mWh の電力を消費し続けた際の電圧の変化をグラフにしたものである．このリチウムイオン電池の最大電圧は 4.05[V] であり，ESP32 が動作することの出来る最低電圧は 3.4[V] であった．図 16 からは時間が経つにつれて, 現象はしているが, 局所的にグラフの傾きが変化している．この変化を回帰関数として定義すれば現在の電圧からどれほど電圧が残っているかが分かるようになる．

6.　分析

6.1　評価・分析

基礎実験から得られたデータを基に評価と分析を記す．

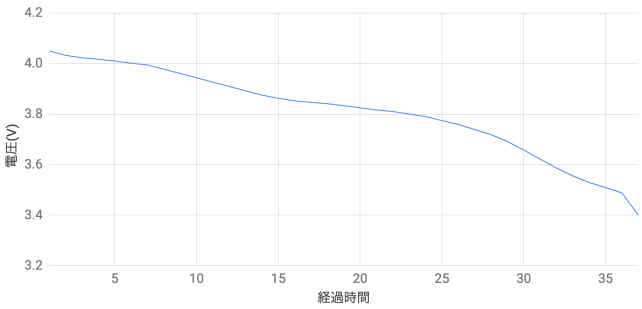

図 16　リチウムイオン電池の電圧の減衰

6.1.1　太陽光パネルの発電量

太陽光パネルの発電量のデータの分析を行う．図 12 では直射日光が当たる際の発電量と雲によって直射日光が遮られてしまい太陽光パネルが影になった際の発電量のデータが取得されている．図 12 のデータを基に, 本来雲によって太陽光が遮られなかった場合と雲に陰りがあった場合どれほど発電量に影響するのかを分析する．図 17 は図 12 のグラフに本来取得できたはずの電力量を赤色で重ねたグラフである．元の電力量が濃く表されており, 雲に隠れて本来得るべき電力が得られなかった部分が薄く表されている．この薄い部分を損失分と考える．損失した電力量は 2129.9mW である．つまり, 全体の約 11.5%もの電力が雲によって失われてしまった．また時間に換算すると, 全体の計測時間が 40 分であり, 雲がかかっていた時間は 10 分であるため, 全体の 25%が雲にかかっていたこととなる．また, 単一時間に着目すると約 45%もの電力の損失となる．また雲がかかった際の発電量は約 200mW〜220mW となり, ESP32 の Wifi 接続時の消費電力とほぼ同等の電力量となる．ESP32 は太陽光発電ができない夜間も同様に稼働する必要があり, 日中にて夜間消費する電力以上の電力を確保する必要があるため直射日光が当たっている方がとても望ましいことが分かる．

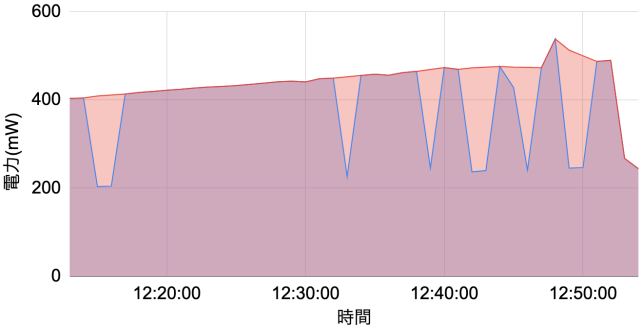

図 17　太陽光が遮られた際の損失発電量

また図 13 のグラフでは, 12 月 22 日と 12 月 23 日の 11:30〜17:00 までの太陽光パネルから取得できる電圧の値であるが，どちらとも雲量 0%の晴れの日であり, グラフの形がほとんど変化してないことである．なお, 太陽光パネ

ルの設定場所や設置角度は全て同じ条件としている．図 13 より, 雲量 0%である晴れの日 (快晴) である場合は 1 日の発電量がほぼ同じ状態になることが分かる．そのため発電量のモデルが 1 つ存在することで天気予報を用いてある程度発電量の目安を立てることが可能である．

また第三章の図 4 では, 雲量によって変化するばらつき具合を示した．雲量が 0%～60%までは最小値から最大値までの幅が約 200mW～300mW ほども開いているのは直射日光が当たるか否かで非常に発電量が変化するからであると考える．雲量というものが厳密に雲の量の平均値ではないため, 直射日光が遮られる時間が雲量のパーセンテージにはなりえない．そのため, 基礎的な雲量と太陽光発電量を紐付けたデータが必要になっている．よりデータを大きくしていくことで, 最大値と最小値は小さくならないが第一四分数と第三四分数の値が収束していくと考えられるため, 今後もより多くのデータの収集を行う必要がある．また機械学習との組み合わせでより効率よくデータの紐付けができるとも考えられる．

6.1.2 リチウムイオン電池の電圧の減衰特性

図 16 にてリチウムイオン電池の電圧の減衰の関係をグラフにして示したが, ここでは実際に回帰関数 $f(x)$ を求め出す．このグラフを回帰関数とした場合 $f(x) = 4.06 - 0.0184x + 1.86 \times 10^{-3}x^2 + 1.32 \times 10^{-4}x - 3 - 6.98 \times 10^{-5}x^4 + 6.85 \times 10^{-6}x^5 - 2.94 \times 10^{-7}x^6 + 5.91 \times 10^{-9}x^7 - 4.55 \times 10^{-11}x^8$ の様な方程式となった．また $f(x)$ は $R^2 = 0.999$ である．

図 18 は上記の回帰関数を図 16 のグラフに重ねたグラフを表している．赤い線が回帰関数であり, 非常に図 16 に近いものになっていることが分かる．この回帰関数を用いることで, リチウムイオン電池の残量の算出が電源電圧から高い精度で求め出す事が可能である．

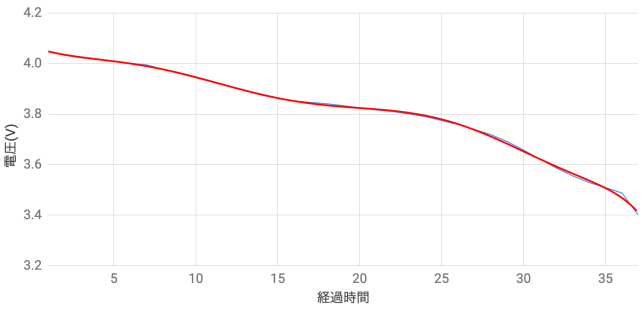

図 18 リチウムイオン電池の電圧と回帰関数

6.1.3 天気予報からの今後の動作可能時間の算出

天気予報と現在のリチウムイオン電池の残量から今後の動作可能時間の算出を行った．その結果を図 19 に示す．

図 19 より天気予報から発電量が算出され, 消費電力を考慮した上で残りの動作可能時間の算出をすることができた．

```
2021-01-18 12:00:00 予測発電量： 41100.0 動作可能時間： 35.00 時間
2021-01-18 13:00:00 予測発電量： 58800.0 動作可能時間： 49.00 時間
2021-01-18 14:00:00 予測発電量： 76500.0 動作可能時間： 64.00 時間
2021-01-18 15:00:00 予測発電量： 100500.0 動作可能時間： 84.00 時間
2021-01-18 16:00:00 予測発電量： 124500.0 動作可能時間： 104.00 時間
2021-01-18 17:00:00 予測発電量： 118500.0 動作可能時間： 99.00 時間
2021-01-18 18:00:00 予測発電量： 112500.0 動作可能時間： 94.00 時間
2021-01-18 19:00:00 予測発電量： 106500.0 動作可能時間： 89.00 時間
2021-01-18 20:00:00 予測発電量： 100500.0 動作可能時間： 84.00 時間
2021-01-18 21:00:00 予測発電量： 94500.0 動作可能時間： 79.00 時間
2021-01-18 22:00:00 予測発電量： 88500.0 動作可能時間： 74.00 時間
2021-01-18 23:00:00 予測発電量： 82500.0 動作可能時間： 69.00 時間
2021-01-19 00:00:00 予測発電量： 76500.0 動作可能時間： 64.00 時間
2021-01-19 01:00:00 予測発電量： 70500.0 動作可能時間： 59.00 時間
2021-01-19 02:00:00 予測発電量： 64500.0 動作可能時間： 54.00 時間
2021-01-19 03:00:00 予測発電量： 58500.0 動作可能時間： 49.00 時間
2021-01-19 04:00:00 予測発電量： 52500.0 動作可能時間： 44.00 時間
2021-01-19 05:00:00 予測発電量： 46500.0 動作可能時間： 39.00 時間
2021-01-19 06:00:00 予測発電量： 40500.0 動作可能時間： 34.00 時間
2021-01-19 07:00:00 予測発電量： 34500.0 動作可能時間： 29.00 時間
2021-01-19 08:00:00 予測発電量： 52500.0 動作可能時間： 44.00 時間
2021-01-19 09:00:00 予測発電量： 66300.0 動作可能時間： 56.00 時間
2021-01-19 10:00:00 予測発電量： 80100.0 動作可能時間： 67.00 時間
2021-01-19 11:00:00 予測発電量： 93900.0 動作可能時間： 79.00 時間
2021-01-19 12:00:00 予測発電量： 111900.0 動作可能時間： 94.00 時間
```

図 19 今後の動作可能時間の算出結果

7. 議論

本研究では太陽光パネルを IoT デバイスの電力供給源として消費電力の制御や発電量の天気予報からの予測や, 太陽光パネルを用いることで発生するユーザへのインタラクションの改善についての提案を行った．IoT の発展において電力供給の問題については課題であるということは様々な論文でも取り上げられ, IoT デバイスの増加に伴う消費電力の増加や, 設置場所の制限について議論がかわされている [15]．自然エネルギーである太陽光発電を IoT に活用する仕組みは先行研究でもいくつか取り上げられている [16][17][18][19]．しかし, 小型 IoT デバイスでの, より具体的な消費電力や 2 次電池の残量, 太陽光パネルの発電量の監視まで行っている論文は少ない．また，本研究では太陽光パネルを利用することで発生するユーザへのアプローチへも目を向ける事で, より具体的なサービスを構築する上で重要なユーザ目線で提案を行っている．これは他の論文でもユーザに目を向けているものは少ないと考える．

特にユーザのリクエスト数も, 電力供給を考慮する上での 1 つの要素として取り入れていることが重要な点である．

最も, 太陽光パネルの発電量に関しては自然エネルギーを活用する以上, 人間にはどうすることもできないという点がある．そのため, 太陽光パネルからの発電量は常に不規則であり, 長期的な予想をすることがそもそも困難である．そのために得られる電力が制御することができないため, 消費する電力を制御する点に常に重きをおかなければならない．ユーザのリクエスト数はまさにその消費電力を決定づける重要な要素であり, いかにこのユーザのリクエストをリクエスト通りに達成しつつ, ユーザとの合意を取りながら調整していくのかが大切になる．

その点において, 本研究での数日後までのデータが送れるかどうかを確率的に示すことはユーザにとってはとても大きな目安にすることができる．ユーザには IoT デバイス

の直接的な制御をすることができないため, ユーザのリクエスト数を通じて調整できることはメリットとなる.

本研究の太陽光発電方式に関しては, 負荷として 100 Ω の抵抗を取り付けた. これは用いた太陽光パネルにて発電される最も良い効率であったために採用したが, さらに効率をよくするために最大電力点追従制御 (MPPT) 方式を取り入れる事が必要でもある.

またリチウムイオン電池の残量の測定に関しては, 電圧では実際に残っている静電容量を測定することができない. そのためある程度の残量は取れるものの, より性格な電池残量を取ることが出来ないこととなってしまう. 2 次電池の残量の測定の際は実際に放出している静電容量で測定することで, より精度の高い電池管理をすることができ, 結果としてより効率の良い電力制御に繋がるものだと考える.

8. おわりに

最後に本研究のまとめを述べる. 本研究では, 太陽光パネルを IoT デバイスの電力供給源として消費電力の制御や発電量の天気予報からの予測や, 太陽光パネルを用いることで発生するユーザへのインタラクションの改善についての提案を行った.

これらの提案は太陽光パネルを IoT デバイスの電力供給に利用するための多くの課題の解決に繋がった. また, 発電量が一定でかつ予測ができない課題としてユーザリクエストへのアプローチについても提案を行った. 他の論文では触れられることのないユーザの視点へ目を向ける事でより具体的で実践的なサービスの構築を可能にすることが可能である.

今後の IoT において重要となるデータの多種多様化において, AC 電源が接続できない場所において自然エネルギーを利用する事で設置可能にし, 今まで取得できなかったデータを得る事ができるようになり, IoT データの拡張に貢献する事ができる.

参考文献

[1] Chi, Q., Yan, H., Zhang, C., Pang, Z. and Xu, L. D.: A Reconfigurable Smart Sensor Interface for Industrial WSN in IoT Environment, *IEEE Transactions on Industrial Informatics*, Vol. 10, No. 2, pp. 1417–1425 (2014).

[2] Chu, M., Li, H., Liao, X. and Cui, S.: Reinforcement Learning-Based Multiaccess Control and Battery Prediction With Energy Harvesting in IoT Systems, *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 2009–2020 (2019).

[3] Wan, T., Karimi, Y., Stanaćević, M. and Salman, E.: Perspective Paper—Can AC Computing Be an Alternative for Wirelessly Powered IoT Devices?, *IEEE Embedded Systems Letters*, Vol. 9, No. 1, pp. 13–16 (2017).

[4] Bekaroo, G. and Santokhee, A.: Power consumption of the Raspberry Pi: A comparative analysis, *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pp. 361–366 (2016).

[5] Myridakis, D., Spathoulas, G., Kakarountas, A., Schoinianakis, D. and Lueken, J.: Anomaly detection in IoT devices via monitoring of supply current, *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pp. 1–4 (2018).

[6] Mohammed, N. Q., Ahmed, M. S., Mohammed, M. A., Hammood, O. A., Alshara, H. A. N. and Kamil, A. A.: Comparative Analysis between Solar and Wind Turbine Energy Sources in IoT Based on Economical and Efficiency Considerations, *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 448–452 (2019).

[7] Bedi, G., Venayagamoorthy, G. K., Singh, R., Brooks, R. R. and Wang, K.: Review of Internet of Things (IoT) in Electric Power and Energy Systems, *IEEE Internet of Things Journal*, Vol. 5, No. 2, pp. 847–870 (2018).

[8] Choi, C., Jeong, J., Lee, I. and Park, W.: LoRa based renewable energy monitoring system with open IoT platform, *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–2 (2018).

[9] Kraemer, F. A., Ammar, D., Braten, A. E., Tamkittikhun, N. and Palma, D.: Solar Energy Prediction for Constrained IoT Nodes Based on Public Weather Forecasts, *Proceedings of the Seventh International Conference on the Internet of Things*, New York, NY, USA, Association for Computing Machinery (2017).

[10] Jackson, N., Adkins, J. and Dutta, P.: Reconsidering Batteries in Energy Harvesting Sensing (2018).

[11] Chu, M., Li, H., Liao, X. and Cui, S.: Reinforcement Learning-Based Multiaccess Control and Battery Prediction With Energy Harvesting in IoT Systems, *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 2009–2020 (2019).

[12] Kul, B. and Şen, M.: Energy saving IoT-based advanced load limiter, *2017 XXVI International Scientific Conference Electronics (ET)*, pp. 1–5 (2017).

[13] Gummeson, J.: Energy Harvesting is Charging Up, Vol. 22, No. 4 (2019).

[14] Wu, J. and Chou, C.: A Solar Power Generation System With a Seven-Level Inverter, *IEEE Transactions on Power Electronics*, Vol. 29, No. 7, pp. 3454–3462 (2014).

[15] : Persistent and adaptive power system for solar powered sensors of Internet of Things (IoT), *Energy Procedia*, Vol. 143, pp. 739 – 741 (2017). Leveraging Energy Technologies and Policy Options for Low Carbon Cities.

[16] Spanias, A. S.: Solar energy management as an Internet of Things (IoT) application, *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*, pp. 1–4 (2017).

[17] Shrihariprasath, B. and Rathinasabapathy, V.: A smart IoT system for monitoring solar PV power conditioning unit, *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pp. 1–5 (2016).

[18] Deshmukh, N. S. and Bhuyar, D. L.: A Smart Solar Photovoltaic Remote Monitoring and Controlling, *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 67–71 (2018).

[19] Kaur, N. and Sood, S. K.: An Energy-Efficient Architecture for the Internet of Things (IoT), *IEEE Systems Journal*, Vol. 11, No. 2, pp. 796–805 (2017).

