ESP32のDeepsleepからの復帰時間の規則性によるデータ収集の遅延の削減

井出 佑[1]　大沢 恭平[2]　串田 高幸[1]

概要：IoT は環境モニタリングで使用される．特に森林火災の検知では，被害を最小限に抑えるために火災の早期の検知をする必要がある．IoT 機器は，インフラが整っていない屋外で使用する場合，バッテリー駆動で使用される．この時，省電力化の手法として一部の機能を停止させるものがある．IoT 機器には ESP32 があり，省電力化モードとして Deepsleep がある．Deepsleep から復帰するのに 1 秒から 2 秒程度かかる．課題は，Deepsleep からの復帰に遅延が発生することでセンシングが遅れることである．提案では，センシングを行う ESP32 の Deepsleep からの復帰にかかる時間の変動パターンを調べ，それに合わせて次回の Deepsleep の時間を変動させる手法を提案した．基礎実験では ESP32 ごとの Deepsleep からの復帰にかかる時間を計測する実験と，サイクルの長さを変えた場合の Deespsleep からの復帰にかかる時間を計測する実験を行った．実験では時間の計測を 180 回行い，これを各 ESP32 で 5 回ずつ行った．3 台の ESP32 の Deepsleep からの復帰にかかる時間を計測する実験では，3 台の ESP32 で同じ周期，変動の幅，パターンで復帰にかかる時間が変動することが分かった．周期に関しては，3 台のどの ESP32 でも復帰回数が 62 回で最初に測った復帰時間まで戻って来ることが分かった．サイクルの長さを変えた場合の Deespsleep からの復帰にかかる時間を計測する実験では，サイクルの長さに関係なく，復帰した回数で復帰にかかる時間が変動することが分かった．

1. はじめに

背景

Internet of Things(以降 IoT とする) は医療，農業，スマートシティ，製造業，防災の分野で活用される．環境モニタリングでは，IoT 機器は物理的に離れた距離で分散配置して使用される [1, 2]．IoT 機器とは，マイクロコントローラユニット (以降 MCU とする) とセンサーモジュールから構成される，通信機能を持ったセンシングデバイスである [3, 4].

防災分野では，洪水，地震，津波，地滑り，森林火災の検知のための環境モニタリングで使用される [1, 5]．森林火災の検知は，初期段階の火災を検知し消火し被害を最小限に抑えることが目的である [6]．森林火災は，表面火災が最初に発生し，表面火災が樹冠に上昇して葉と枝に引火することで樹冠火災が発生する [7]．表面火災とは，地表の可燃物が燃焼する火災である [8]．樹冠火災とは，木や低木の上部が燃焼する火災である [8]．表面火災から約 10 秒から 40 秒の時間をかけて樹冠火災に遷移する [9]．樹冠火災は表面火災に比べて広がりが速く，火柱と熱量，燃焼高度が増加するため消火が困難になり，人的被害と森林破壊の規模が大きくなる [10]．このことから，表面火災の段階で検知して消火することが被害を抑えるために必要である．森林火災の検知では温度データを 12 秒から 3600 秒の間隔で収集する [11].

IoT 機器をインフラが整っていない屋外で使用する場合，電源の確保が難しいためバッテリー駆動で使用される [12]．また，屋外での使用ではアクセスが困難でコストがかかることがあり，頻繁なバッテリーの交換が困難になる場合がある [13]．このことから，運用コストを削減するためにバッテリーの交換頻度を減らし，長時間稼働させるために省電力化する必要がある．省電力化の手法として，IoT 機器の一部の機能を使用する場合を除いて，停止させるものがある [14]．この時，機能の起動と停止は定期的に行うため，起動しておく時間と停止しておく時間のサイクルを決めて制御される．この時，機能を起動したまま，次の処理を待つことをアイドル状態と呼ぶ [15]．一部の機能を使用する時以外は停止させておく省電力化の手法は，森林火災の検知にも使用される [16].

IoT 機器には ESP32 がある．ESP32 とは，Espressif

[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
[2] 東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻 クラウド・分散システム研究室
〒 192–0982 東京都八王子市片倉町 1404-1

Systems 社が開発した，低コストかつ低電力で，Wi-Fi と Bluetooth の 2 種類の無線機能を内蔵する MCU である [17]．ESP32 には省電力化モードとして Deepsleep がある．Deepsleep とは ESP32 の RTC モジュール以外の全てのモジュールが停止する状態である [18]．Deepsleep を使用する時，Deepsleep からネットワーク機能をオンにするまでに約 1〜2 秒かかる．本稿では，Deepsleep からのネットワーク機能をオンにするまでにかかる時間を復帰時間とする．また，Deepsleep の状態を維持する時間をスリープ時間とする．

課題

課題は約 1〜2 秒の復帰時間が発生することで，センシングが遅れることである．森林火災の検知のユースケースでは，センシングが遅れることで火災の検知が遅れ，消防への通報が遅れる．通報が遅れることで樹冠火災が発生し，火災の被害の規模が拡大する．図 1に，Deepsleep からの復帰に時間がかかることで火災の検知が遅れ消防の消火活動が遅れることを示す．

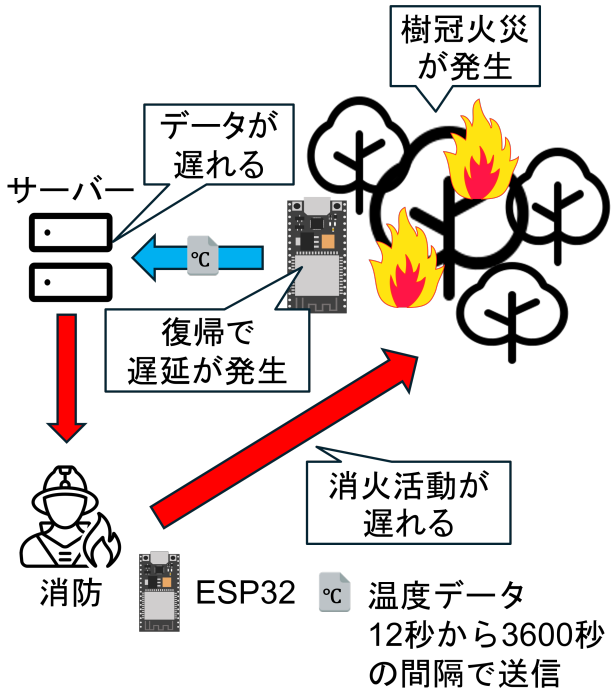

図 1: Deepsleep からの復帰に時間がかかることで消防の消火活動が遅れる例

各章の概要

第 2 章では関連研究について議論する．第 3 章では，課題に対しての提案方式について説明する．第 4 章では，実装したソフトウェアについて説明する．第 5 章では，評価実験について説明する．第 6 章では，提案手法についての議論をする．第 7 章では，本稿のまとめを行う．

2. 関連研究

指数平滑法を適用して通信のデータ量を予測し，残留エネルギーからエンドノードのデューティサイクルを調整する提案を行っている研究がある [19]．これにより，ノード間の電力消費のバランスを取り，パケット遅延を抑えて，電力効率を改善できる．この研究では，スリープ時間の決定時に復帰時間を含めて計算していないため，本稿の課題の解決は行えていない．

強化学習を使用した各ノードのクラスタの選択による動的クラスタリングと，適応型スリープスケジューリングにより，エネルギー消費とネットワーク寿命，データ回復精度を既存の手法よりも改善した研究がある [20]．本稿は，センシングの遅延を課題としているため，この提案は適していない．

Public Goods Game 理論をもとに，各タスクの優先順位で実行するタイミングを調整することにより，デューティサイクルを調整するリアルタイム OS スケジューラを提案した研究がある [21]．この提案では優先度の低いタスクに対して，バッテリー残量が少ない時は実行を遅らせるといったことを行う．この研究は，バッテリー駆動ノードの寿命延長を目的としているため，本稿の課題解決には適していない．

3. 提案

提案方式

本稿では，ESP32 の復帰時間によるセンシングの遅延を削減することを目的とする．提案として，復帰時間の変化パターンからスリープ時間を決定する手法を提案する．この手法では，各 ESP32 で復帰時間を 124 回計測して保存する．次に 124 回分のデータを 62 回ずつに分割し，同じ順番の復帰時間の誤差を計算し，誤差の最大値を保存する．そして，124 回計測した後の最初の Deepsleep からの復帰を 1 回目の復帰として，保存した復帰時間のデータの 1 回目を次の復帰時間とする．その後は Deepsleep の入るたびに復帰時間のデータを順番に取り出して次の復帰時間とする．この時，124 回目まで達した場合は，また 1 回目に戻る．そして，設定されているスリープ時間と，次の復帰時間に復帰時間の誤差の最大値を加算した値との差を次のスリープ時間とする．本提案には，事前準備と実行フェーズがある．事前準備と実行フェーズについて項目を分けて説明する．

事前準備

まず，復帰時間の測定を行う．Deepsleep からの復帰を 124 回繰り返して復帰時間を CSV ファイルに保存する．そして，62 回ずつに分割して，同じ順番の値の誤差を出し，誤差の最大値を CSV ファイルに保存する．誤差を E，分割したそれぞれの復帰時間のデータを R1 と R2，復帰時間

のデータの順番を i とした時，式 (3) で表される．

$$E = \sqrt{(R1(i) - R2(i))^2} \qquad (0)$$

順番 i の範囲は 1 から 62 である．順番 i の範囲すべての誤差 E を計算し，最大値を CSV に保存する．また，次の復帰時間の復帰時間のデータでの順番を CSV ファイルに保存する．実行フェーズの最初の復帰時間は復帰時間のデータの 1 番目の復帰時間であるため，1 を CSV ファイルに保存する．事前準備で保存した CSV ファイルは全てセンシング用の ESP32 に保存する．図 2に取得した復帰時間のデータを分割してから誤差の最大値を保存するまでの流れを示す．

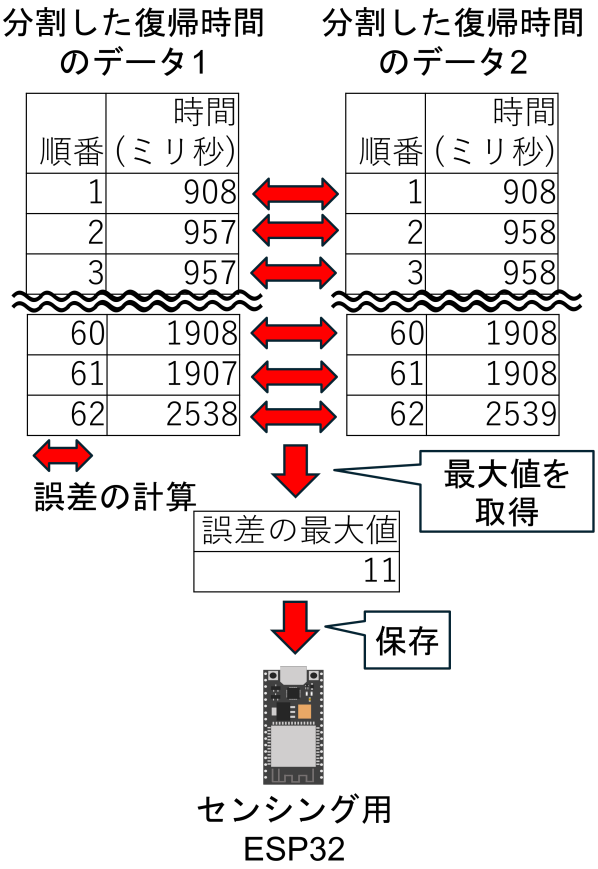

図 2: 復帰時間のデータを分割してから誤差の最大値を保存するまでの流れ

実行フェーズ

図 7に保存されている順番をもとに，復帰時間のデータから次の復帰時間を取得し次のスリープ時間を計算するまでの流れを示す．実行フェーズでは，センシングと Deepsleep を繰り返す．Deepsleep に入る前に，保存されている順番の復帰時間のデータの値を調べ，これを次の復帰時間とする．最後に，設定してあるセンシングの間隔と，次の復帰時間に誤差の最大値を加算した値の差を次のスリープ時間として Deepsleep に入る．図 7ではセンシングの間隔を，背景で述べた森林火災の検知で使用されるセンシングの間隔のうちの最も短い 1200 ミリ秒を，取り上げて計算を行っている．誤差の最大値を計算に含む理由は，復帰時間が誤差の分だけ増加して，センシングが遅れることを防ぐためである．Deepsleep に入る前に，保存されている順番を 1 加算した値に更新する．もし，保存されている順番が 124 以上の場合は値を 1 にする．

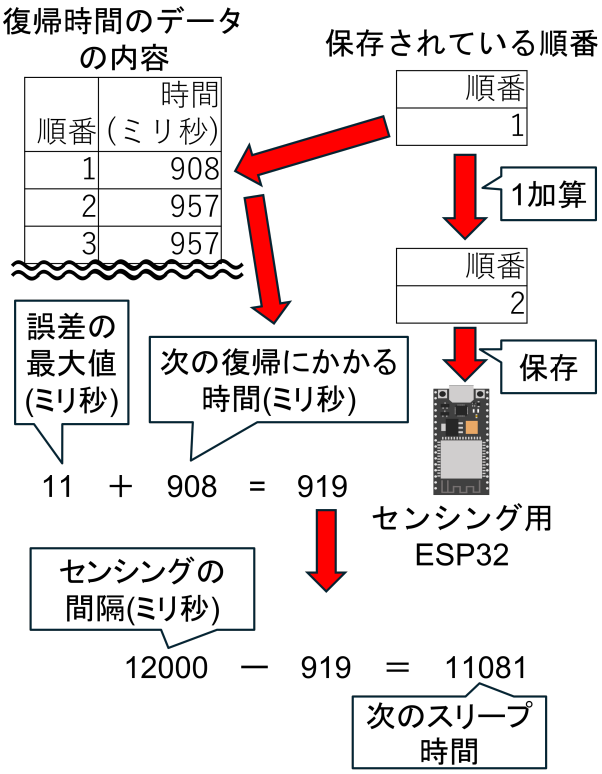

図 3: 保存されている順番の取得から次のスリープ時間を計算するまでの流れ

ユースケース・シナリオ

本稿のユースケースとして，森林火災の検知をセンシングノードを使用して行う場合を想定する．図 4にユースケース・シナリオに本稿の提案を適用した場合を示す．

収集するデータは温度データであり，収集間隔は 12 秒から 3600 秒の間で変動する [11]．取得したデータはサーバーに送られて，火災の検知の判断に使用される．センシングの後は次のセンシングタイミングまで Deepsleep を行う．この場合のスリープ時間には復帰時間を反映していないため，センシングに遅延が発生し火災の検知が遅れてしまう．提案を適用することで，復帰時間を反映したスリープ時間にすることができる．これにより，復帰時間による火災の検知の遅延を削減することができる．

4. 実装

ESP32 を 2 台用意して，それぞれに提案手法をもとに作

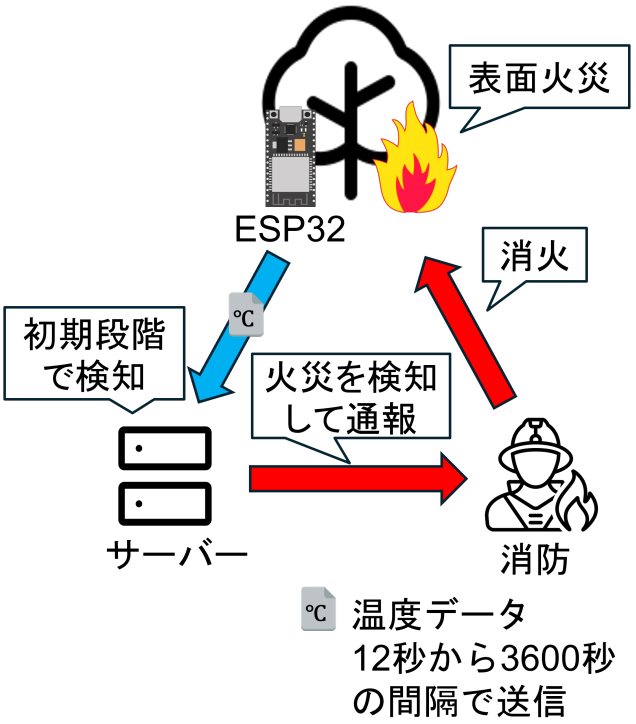

図 4: 提案を適用した場合のユースケース・シナリオ

成したソフトウェアを実装する．それぞれ計測ノードとセンシングノードとする．計測ノードは事前準備で各 ESP32 の復帰時間を計測する．センシングノードは提案手法によりスリープ時間を決定し，Deepsleep とセンシングを繰り返す．事前準備で計測ノードの GPIO4 と GPIO19 はそれぞれ，センシングノードの同じ GPIO にジャンプワイヤーで接続する．実装について事前準備と実行フェーズに分けて説明する．

4.1 事前準備

事前準備を計測フェーズと設定フェーズに分けて説明する．

4.1.1 計測フェーズ

図 5に，計測フェーズの計測ノードとセンシングノードの動作の流れを示す．最初にセンシングノードは Deepsleep に入る．次に，計測ノードが GPIO4 を High にすると，センシングノードが Deepsleep から復帰する．計測ノードは GPIO4 を High にした時間を記録しておく．復帰したセンシングノードは初期化処理を行う．ネットワーク機能を起動した後，センシングノードは GPIO19 を High にする．その後，5 秒間待機した後に GPIO19 を Low にして再び Deepsleep に入る．計測ノードは GPIO19 が High になった時間を記録する．そして，GPIO4 を High にした時間から GPIO19 が High になった時間までに経過した時間を復帰時間として，return.csv に保存する．その後，GPIO4 を Low にした後に 10 秒間待機して，再び GPIO4 を High にする．これらの処理を 124 回行う．

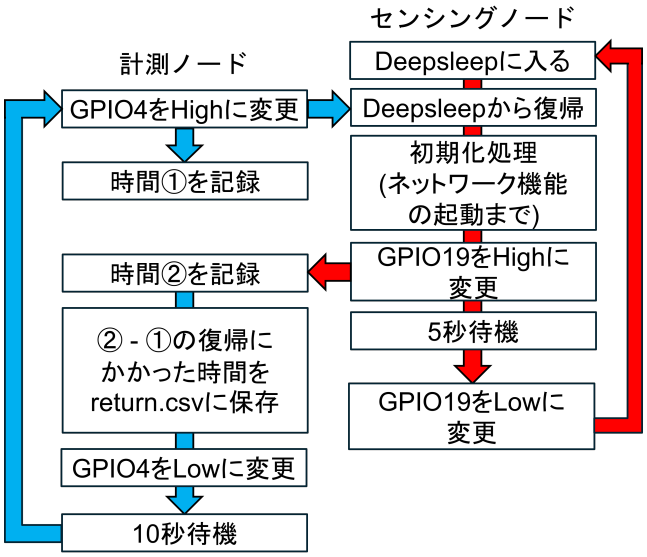

図 5: 計測フェーズの動作の流れ

4.1.2 設定フェーズ

計測フェーズの後に設定フェーズに入る．図 6に設定フェーズの計測ノードとセンシングノードの動作の流れを示す．

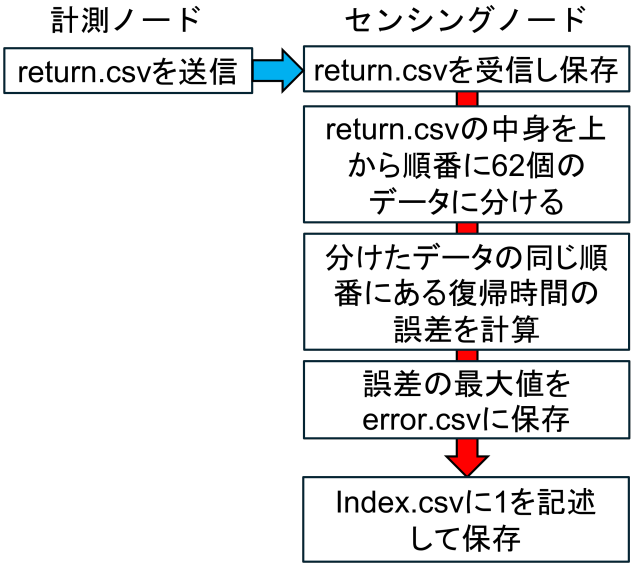

図 6: 設定フェーズの動作の流れ

計測フェーズ終了後，計測ノードは return.csv をセンシングノードに送信する．センシングノードは，return.csv のデータを 62 個ずつに分ける．分けた後のデータの同じ順番の値の誤差を求め，誤差の最大値を error.csv に保存する．最後に 1 を index.csv に保存する．index.csv の数字は実行フェーズで次の復帰時間を return.csv から取得する際に，値を指定して取得する時に使用する．1 周期が 62 回で

あり，125 回目の復帰時間は復帰時間のデータの 1 番目の復帰時間と同様になる．そのため index.csv には 1 を記述して保存する．

4.2 実行フェーズ

事前準備が終了した後，実行フェーズに入る．図 7に実行フェーズのセンシングノードの動作の流れを示す．センシングノードは起動した後，センシングを行いセンサーデータをサーバーに送信する．次に，index.csv から順番を取得する．そして，pattern.csv の順番の値を取得し，error.csv の値と足し合わせる．センシングの間隔と足し合わせた値の差を次のスリープ時間とする．最後に，順番に 1 加算した値を index.csv に保存し，決定したスリープ時間だけ Deepsleep に入る．順番に 1 を加算した値が 124 を超過した場合，順番を 1 にする．Deepsleep から復帰した後，センシングから同じ処理を繰り返す．

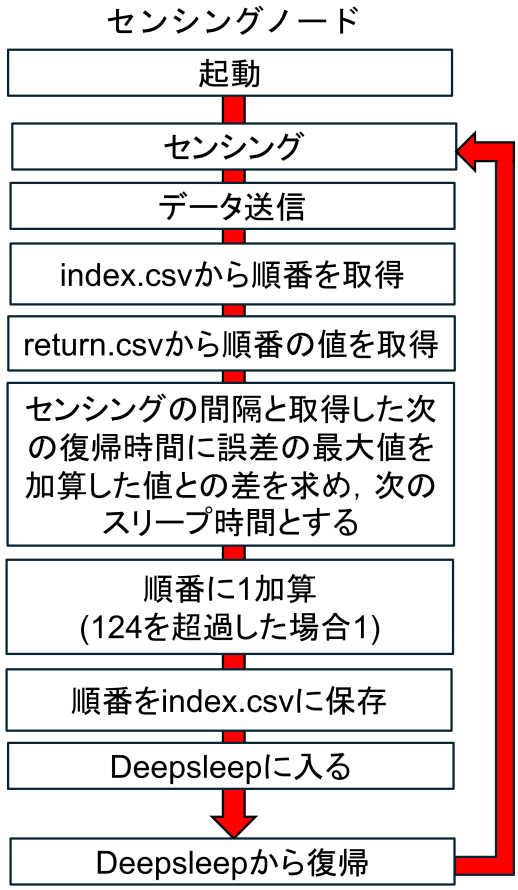

図 7: 実行フェーズの動作の流れ

5. 評価実験

評価実験では，3 台の ESP32 を用意して評価実験を行う．提案を適用した ESP32 と提案の事前準備に使用する ESP32 を 1 台ずつ用意し，もう 1 台は提案を適用しない ESP32 を用意する．提案を適用する前と後で，単位時間あたりに取得できたデータ量が，本来取得できるはずのデータ量とどれだけ差があるかで評価を行う．

基礎実験

基礎実験では，ESP32 ごとの復帰時間の計測と，サイクルの長さを変更した場合の復帰時間の計測を行った．それぞれ，1 回の実験での時間の計測を 180 回行い，これを 5 回ずつ行った．

実験環境

基礎実験で使用した実験環境について記載する．実験で使用するノードには ESP32 を使用した．ESP32 には MicroPython ファームウェアの ESP32_GENERIC-20241129-v1.24.1 を実装し，MicroPython の開発環境を構築した．ノードには計測ノードとスリープノードがある．計測ノードはスリープノードの復帰時間の計測を行う．スリープノードは Deepsleep とアイドル状態を任意の時間だけ維持して，切り替えるサイクルを繰り返す．アイドル状態では，ネットワーク機能が起動している．実験ごとの条件について以下に記す．

ESP32 ごとの復帰時間の計測

- 計測ノード：1 台
- スリープノード：3 台
- サイクル 1：Deepsleep 5 秒，アイドル状態 5 秒

スリープノードはそれぞれ ESP32_A，ESP32_B，ESP32_C とした．

サイクルの長さを変更した場合の復帰時間の計測

- 計測ノード：1 台
- スリープノード：1 台
- サイクル 1：Deepsleep 5 秒，アイドル状態 5 秒
- サイクル 2：Deepsleep 20 秒，アイドル状態 10 秒

計測ノード

計測ノードは最初に GPIO4 を High にし，GPIO19 が High になるのを待つ．GPIO4 を High にした時点から，GPIO19 が High になるまでの時間を測り，それをスリープノードの復帰時間として，CSV ファイルに保存する．GPIO19 が High になった後，GPIO4 を Low にして，設定した時間が経過した後に，再び GPIO4 を High にする．

スリープノード

スリープノードは最初に Deepsleep に入り，GPIO4 が High になった場合に Deepsleep から復帰する．ファームウェアのブートシーケンスとネットワーク機能の起動が終了した後，GPIO19 を High にする．そして，設定した時間だけアイドル状態を維持した後，GPIO19 を Low にして再び Deepsleep に入る．

実験結果と分析

ESP32 ごとの復帰時間の計測

この実験では各 ESP32 の復帰時間の計測を行った．図 8に計測した結果の折れ線グラフを示す．

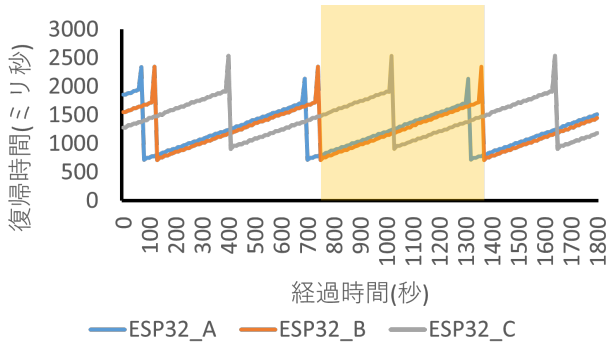

図 8: 機器ごとの復帰時間の結果

縦軸は復帰にかかった時間を示しており，単位はミリ秒である．また，横軸は実験開始から終了までに経過した時間である．それぞれのグラフを見ると，平行にズレてはいるものの，全てのスリープノードである一定のところまで時間が長くなった後に，急激に時間が短くなるといったパターンで復帰にかかった時間が遷移している．復帰にかかった時間が短くなった地点から，次に時間が短くなった地点の直前までを 1 周期とする．図 8の ESP32_B で示すと黄色く塗りつぶされた範囲を 1 周期とする．1 周するまでのデータ数を数えると，全ての ESP32 の全ての周期のデータ数が 62 個であった．また，各 ESP32 の 1 周期の復帰にかかった時間のデータに関して，最初と最後の 4 回分のデータを図 9に示す．

	ESP32_A	ESP32_B	ESP32_C
順番	時間(ミリ秒)	時間(ミリ秒)	時間(ミリ秒)
1	718	708	908
2	748	758	957
3	748	758	957
4	777	788	987
121	1668	1678	1877
122	1698	1708	1908
123	1697	1718	1907
124	2137	2348	2538

図 9: 各 ESP32 の 1 周期の最初と最後の 4 回分の復帰にかかった時間のデータ

機器ごとに復帰にかかった時間の変動の幅にほとんど差がなく，異なる周期の同じ順番のデータの誤差範囲は± 11 ミリ秒以内だった．1 周期の内，最初と最後の値以外は，± 10 以内の値が続けて出てきた後に，一つ前の値に 20 から 40 加算された値が出てくるといったことを繰り返している．図 9の ESP32_A を例にすると，748 ミリ秒が 2 回出てきた後に 777 が 2 回出てきている．この時，777 と 748 の差は 29 である．このように，同じ数が出てきた後に，前の値に 20 から 40 を加算した値が次に出てくることを繰り返している．これらのことから，どの ESP32 も復帰時間は，同様の周期，変動幅，パターンで変化する事が分かった．

サイクルの長さを変更した場合の復帰時間の計測

この実験では，サイクルの長さを変更した時の復帰時間の計測を行った．図 10に同一の ESP32 でサイクルの長さを変化させた場合の復帰にかかった時間の最初と最後の 4 回分のデータを表で示す．

	Deepsleep5秒 アイドル状態5秒	Deeplseep20秒 アイドル状態10秒
順番	時間(ミリ秒)	時間(ミリ秒)
1	708	718
2	738	748
3	748	748
4	778	778
121	1668	1667
122	1698	1698
123	1698	1698
124	2148	2148

図 10: サイクルの長さを変化させた場合の復帰にかかった時間の最初と最後の 4 回分のデータ

1 周期の長さは 62 個と同じだった．1 周期の同じ順番の値の誤差範囲は± 11 ミリ秒だった．図を例にすると，図の一番上にあるデータの 708 と 718 の差は 10 だった．このような差の計算をすべてのデータで行った結果，差の範囲が± 11 秒であった．これらのことから，復帰時間は，サイクルの長さに関係なく復帰した回数で変動することが分かった．

6. 議論

本稿の基礎実験では，復帰時間が約 1 秒から 2 秒あることを示した．本提案は，復帰時間の原因を解決することで，復帰時間を削減できる．考えられる原因として，実行されるプログラムとファームウェアによるブートシーケンスがある．原因を調べるために，まずは復帰に必要な処理に含まれるブートシーケンスと実行されるプログラムの部

分を分けて，かかった時間を測る．復帰時間は，GPIO4 が High になったところから，ネットワーク機能を起動し GPIO19 を High にするまでの時間である．このことから，GPIO4 の信号を受信してからプログラムを実行するまでの時間と，プログラムを実行してから GPIO19 を High にするまでの時間を分けて測る．これにより復帰に必要な処理のうち，ブートシーケンスの処理の部分のみと，実行されるプログラムの処理の部分のみの処理にかかる時間を計測することができる．ファームウェアによるブートシーケンスで時間がかかっている場合，ファームウェアを使用しない C 言語を使用することでこの問題を解決できる．

7. おわりに

本稿では，Deepsleep からの復帰に遅延が発生することでセンシングが遅れることを課題とした．この課題は森林火災の検知が目的のユースケースでは問題となる．提案では，センシングを行う ESP32 の復帰時間の変動パターンを調べ，それに合わせて次回の Deepsleep の時間を変動させる手法を提案した．基礎実験では ESP32 ごとに復帰時間を計測する実験と，サイクルの長さを変えた場合の復帰時間を計測する実験を行った．3 台の ESP32 の復帰時間を計測する実験では，3 台の ESP32 の復帰時間の変動の周期が復帰の回数で 62 回だと分かり，どの ESP32 も同じ周期，変動の幅，パターンで復帰にかかる時間が変動することが分かった．サイクルの長さを変えた場合の復帰時間を計測する実験では，サイクルの長さに関係なく，復帰の回数が 62 回を 1 周期として，復帰にかかる時間が変動することが分かった．

参考文献

[1] Chataut, R., Phoummalayvane, A. and Akl, R.: Unleashing the Power of IoT: A Comprehensive Review of IoT Applications and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0, *Sensors*, Vol. 23, No. 16 (online), DOI: 10.3390/s23167194 (2023).

[2] Esposito, M., Palma, L., Belli, A., Sabbatini, L. and Pierleoni, P.: Recent Advances in Internet of Things Solutions for Early Warning Systems: A Review, *Sensors*, Vol. 22, No. 6 (online), DOI: 10.3390/s22062124 (2022).

[3] Gulati, K., Kumar Boddu, R. S., Kapila, D., Bangare, S. L., Chandnani, N. and Saravanan, G.: A review paper on wireless sensor network techniques in Internet of Things (IoT), *Materials Today: Proceedings*, Vol. 51, pp. 161–165 (online), DOI: https://doi.org/10.1016/j.matpr.2021.05.067 (2022). CMAE'21.

[4] Silverio-Fernández, M., Renukappa, S. and Suresh, S.: What is a smart device?-a conceptualisation within the paradigm of the internet of things, *Visualization in Engineering*, Vol. 6, No. 1, pp. 1–10 (2018).

[5] Ramadan, M. N., Basmaji, T., Gad, A., Hamdan, H., Akgün, B. T., Ali, M. A., Alkhedher, M. and Ghazal, M.: Towards early forest fire detection and prevention using AI-powered drones and the IoT, *Internet of Things*, Vol. 27, p. 101248 (online), DOI: https://doi.org/10.1016/j.iot.2024.101248 (2024).

[6] Mohapatra, A. and Trinh, T.: Early wildfire detection technologies in practice—a review, *Sustainability*, Vol. 14, No. 19, p. 12270 (2022).

[7] Xanthopoulos, G. and Athanasiou, M.: Crown fire, *Encyclopedia of wildfires and wildland-urban interface (wui) fires*, Springer, pp. 183–197 (2020).

[8] Liu, N.: Wildland surface fire spread: Mechanism transformation and behavior transition, *Fire Safety Journal*, Vol. 141, p. 103974 (online), DOI: https://doi.org/10.1016/j.firesaf.2023.103974 (2023).

[9] Cobian-Iñiguez, J., Aminfar, A. H., Saha, S., Awayan, K., Weise, D. R. and Princevac, M.: The Transition and Spread of a Chaparral Crown Fire: Insights from Laboratory Scale Wind Tunnel Experiments, *Journal of Combustion*, Vol. 2022, No. 1, p. 5630594 (online), DOI: https://doi.org/10.1155/2022/5630594 (2022).

[10] Guo, H., Kong, L., Gao, Y., Xiang, D., Li, Z., Gong, L. and Zhang, Y.: Transition from surface fire to crown fire and effects of crown height, moisture content and tree flower, *Fire Technology*, Vol. 60, No. 2, pp. 1403–1419 (2024).

[11] Ioannidis, I., Anagnostopoulos, A. and Mamalis, B.: Smart Forest Fire Monitoring and Detection System using Microservices and Container-based Virtualization, *Proceedings of the 28th Pan-Hellenic Conference on Progress in Computing and Informatics*, PCI '24, New York, NY, USA, Association for Computing Machinery, p. 370–376 (online), DOI: 10.1145/3716554.3716610 (2025).

[12] Callebaut, G., Leenders, G., Van Mulders, J., Ottoy, G., De Strycker, L. and Van der Perre, L.: The Art of Designing Remote IoT Devices—Technologies and Strategies for a Long Battery Life, *Sensors*, Vol. 21, No. 3（オンライン），DOI: 10.3390/s21030913 (2021).

[13] Alsharif, M. H., Jahid, A., Kelechi, A. H. and Kannadasan, R.: Green IoT: A Review and Future Research Directions, *Symmetry*, Vol. 15, No. 3 (online), DOI: 10.3390/sym15030757 (2023).

[14] Ciuffoletti, A.: Deep-Sleep for Stateful IoT Edge Devices, *Information*, Vol. 13, No. 3 (online), DOI: 10.3390/info13030156 (2022).

[15] Besher, K. M., Nieto-Hipolito, J. I., Buenrostro-Mariscal, R. and Ali, M. Z.: Spectrum Based Power Management for Congested IoT Networks, *Sensors*, Vol. 21, No. 8 (online), DOI: 10.3390/s21082681 (2021).

[16] Ibrahim, D.: Low-power early forest fire detection and warning system, *Indian Journal of Science and Technology*, Vol. 13, No. 3, pp. 286–298 (2020).

[17] Maier, A., Sharp, A. and Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, *2017 Internet Technologies and Applications (ITA)*, pp. 143–148 (online), DOI: 10.1109/ITECHA.2017.8101926 (2017).

[18] Wu, H., Chen, C. and Weng, K.: An Energy-Efficient Strategy for Microcontrollers, *Applied Sciences*, Vol. 11, No. 6 (online), DOI: 10.3390/app11062581 (2021).

[19] Luo, H., He, M., Ruan, Z. and Zeng, X.: Optimal sleep time controller based on traffic prediction and residual energy in duty-cycled wireless sensor networks, *International Journal of Distributed Sensor Networks*, Vol. 13, No. 12, p. 1550147717748909 (online), DOI:

10.1177/1550147717748909 (2017).
[20] El-Shenhabi, A. N., Abdelhay, E. H., Mohamed, M. A. and Moawad, I. F.: A Reinforcement Learning-Based Dynamic Clustering of Sleep Scheduling Algorithm (RLDCSSA-CDG) for Compressive Data Gathering in Wireless Sensor Networks, *Technologies*, Vol. 13, No. 1 (online), DOI: 10.3390/technologies13010025 (2025).
[21] Rodriguez-Zurrunero, R., Utrilla, R., Romero, E. and Araujo, A.: An Adaptive Scheduler for Real-Time Operating Systems to Extend WSN Nodes Lifetime, *Wireless Communications and Mobile Computing*, Vol. 2018, No. 1, p. 4185650 (online), DOI: https://doi.org/10.1155/2018/4185650 (2018).

