マルチホップネットワークでの動的センシングレートと省エネルギー経路選択を統合したデータ収集の可能時間の延長

大沢 恭平[1]　串田 高幸[1]

概要：マルチホップセンサーネットワークにおいて，バッテリー駆動ノードによる継続的なデータ収集では，一定のセンシングレートによるバッテリー消費の増加とデータ収集可能時間の短縮が課題となっている．特に農業分野の環境モニタリングでは，環境変化に応じた適応的なセンシング制御が求められる．本稿では，マルチホップセンサーネットワークにおけるデータ収集の可能時間の延長を目的として，気象予測と実測値の差から得られる指数平滑スコア (sErr) をもとにセンシングレートを切り替える．この手法はOpenWeatherAPI の気温予測データと実測値との差分を EWMA と標準偏差で標準化し，レート判定スコアを算出する．ESP32 ノードとホスティングサービスの Render 上のサーバーで実装し，OpenWeatherAPI の予報データをもちいた実験を 2025 年 10 月 13 日〜15 日と 2025 年 10 月 17 日〜20 日の 2 回行った結果，直近 24 時間のデータポイントを利用することで，固定 48 個のデータポイントを使用する場合と比較してセンシング回数を 2505 回から 1802 回まで 703 回削減し，同一バッテリーでの稼働時間を 2750[min] から 4273[min] まで約 1.55 倍に延伸できることを確認した．これにより，外部電源に頼らない広域センシングにおいても，高頻度監視と省電力運用を両立できる可能性を示した．

1. はじめに

背景

マルチホップネットワーク（Multi-hop Network）は，複数のノードが中継機能を持ち，直接通信できないノード間でも複数ホップを経由してデータを伝送する分散ネットワークアーキテクチャである [1]．各ノードは，データの送信元，中継点，受信先の役割を動的に切り替えながら，ネットワーク全体での効率的なデータ伝送を実現する．

マルチホップネットワークの特徴として，以下の点があげられる：

- 拡張性: ノード数の増加に対応したネットワーク規模の拡大
- 冗長性: 複数の経路による通信の信頼性向上
- 効率性: 直接通信が困難な長距離でも中継により通信可能

これらの特性により，マルチホップネットワークは従来の単一ホップ通信では実現困難な，広域かつ柔軟なデータ伝送を可能にする．一方で，複数のノードを経由してデータを伝送することによる遅延の増加，経路制御の複雑化，電力消費の増大の課題もある [2].

マルチホップ通信におけるエネルギー消費の特徴

- 中継負荷: 他ノードのデータ転送による追加的な通信処理
- ルーティング制御: 経路探索と維持のための制御メッセージ処理
- 負荷集中: 特定ノードへの通信負荷の集中による早期故障
- ホップ数依存: 経路のホップ数に比例した総エネルギー消費

バッテリー制約がもたらす影響センサーネットワークは複数の小型 IoT デバイスのノードで構成され，それぞれのノードはバッテリーで動作する．ノードの小型化にともない，バッテリーサイズも制限されるため，バッテリー消費量の削減と効率化は重要な課題である．

- ネットワーク分断: 中継ノードの故障による通信経路の喪失
- 負荷の再分配: 故障ノードの負荷が他のノードに集中
- 経路の動的変更: 故障ノードを回避した新たな経路の探索

マルチホップネットワークはシングルホップ通信と比較して広範囲のセンシングに適している．遠隔地にあるゲートウェイやサーバーに対してセンシングデータを送信と

[1] 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

保存するために複数のノードを経由する．短距離のマルチホップを行うことで，ノード間の伝送距離が短縮され，バッテリー消費量が削減できる．したがって，マルチホップネットワークはバッテリー消費量の削減に効果的な手段である [3].

マルチホップネットワークはヘルスケア，軍事，農業で活用される．農業環境モニタリングにおいて，マルチホップネットワークは特に重要な役割を果たす [1]．農地でのモニタリングでは，センサーノードは気温，土壌湿度，作物の監視画像データを収集する．これらは農地の環境整備，作物の成長監視，作物の病害虫の発生予測に使用される [4]．大規模農地では，直接サーバーに接続できない遠隔地のセンサーノードが多数存在し，これらのノードがマルチホップ通信によりデータを集約と伝送する必要がある．

課題

課題は，一定のセンシングレートでセンシングを行い続けることによるバッテリー消費量の増加によって，センシング可能時間が減少することである．センシングを一定レートで行っても，間隔が短いとデータの変化量が乏しいことがある．例えば気温は短期的には微小な変化にとどまり，急激な変化は稀である．従来型の多くのセンシングシステムはイベントの有無に関わらず一定のセンシングレートでデータを取得するため，時系列データとしてのデータの整合性は取れても余分なバッテリーの消費や，ストレージへの圧迫が生じる [5]．また，データの中継を行うクラスターヘッドノードはデータの中継によってクラスターメンバーよりも多くのバッテリーを消費する．マルチホップネットワークでは，トポロジーの上位ノードが停止すると，下位ノードはデータをサーバーに送信できなくなる．このような下位ノード (クラスターメンバー) は孤立ノードと呼ばれる．図 1 は孤立ノード発生の例である．CM1-1，CM1-2，CM2-1，CM2-2 はそれぞれ CH1 と CH2 を介してサーバーまでデータを送信する．しかし，バッテリー枯渇により CH2 が動作を停止すると，CM2-1，CM2-2 はバッテリーが残っていてもサーバーにデータを送信できない．この時，CM2-1，CM2-2 は孤立ノードとなる．孤立ノードになったノードのバッテリーを有効活用できていない．

各章の概要

第 2 章では本稿の関連研究を記述する．第 3 章では本稿の提案手法を記述する．第 4 章では提案手法の実装を記述する．第 5 章では提案手法の評価実験を記述する．第 6 章では提案手法の議論を記述する．第 7 章では本稿のまとめを記述する．

2. 関連研究

LEACH(Low Energy Adaptive Clustering Hierarchy protocol) はクラスターヘッドの選択に確率的なアプローチを採用し，各ラウンドごとにクラスターヘッドが変更される [6]．ノードが交代でクラスターヘッドになることで消費電力量をノード間で均一にしている．クラスターヘッドは TDMA(Time Division Multiple Access) を使用してクラスターメンバーにセンシングデータの送信権限を与える．ノードは指定されたタイミングでセンシングデータを送信するため，他のノードとの電波干渉や競合が抑えられるというメリットがある．ただし，ノード間で同期をとる必要がある．LEACH は全てのノードがサーバーと通信できることを前提としている．実際にはノードの通信範囲は限られているため，マルチホップ通信を必要とする広域での通信には適していない．また，LEACH はセンシングエリア内に多数のノードが配置されている場合に最も有効な手段である．それゆえ，ノード数が少数である場合には効果を発揮しない [7,8].

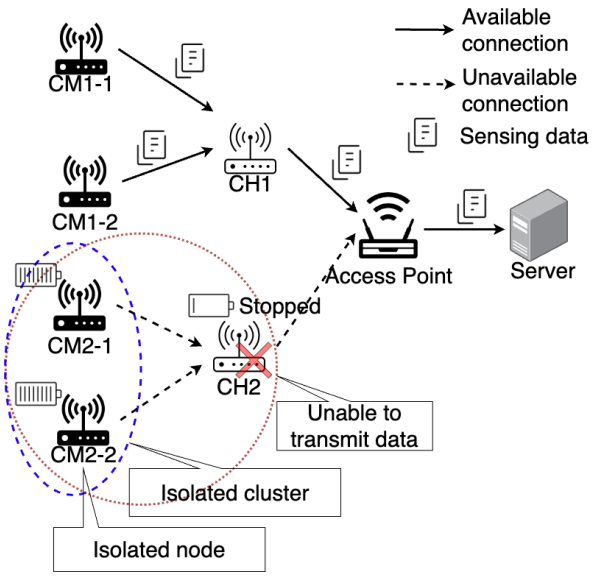

図 1 孤立ノード発生の例

HEED はエネルギー効率の高いクラスターヘッドを選出するアルゴリズムである．各ノードは自身の残存エネルギー量と隣接ノードとの通信距離からクラスターヘッドとしての適正を計算する．各ノードは計算によって求めた値が閾値よりも高い場合，クラスターヘッドを担う．この手法ではクラスターヘッド選出のための隣接ノードとの通信がオーバーヘッドとなるため，ノード数の多い環境では大きな通信コストがかかる．また，センシングエリア内でクラスターサイズを均等に分割できるという条件がある [9–11].

ICIC アルゴリズムでは，マルチホップネットワークにおいて最も伝送距離の短い経路を選択してデータを転送することで消費電力の削減を行う．この手法では，データ転送レートはが考慮されていない [12].

3. 提案

提案方式

本稿では動的なセンシングレートの変更と，省エネルギー経路選択によってバッテリー消費の削減とデータ収集の可能時間を延長することを目的とする．

なお，本提案方式では以下の項目を前提条件として設定する．

- リンクは固定であり，変更されることはない．この前提条件により，各ノードは接続可能なノードのリストをあらかじめ保持している．
- センシングレートは全ノードで共通である．
- サーバーは 1 台である．

提案方式のセンシングレート決定処理を Listing 1 に示す．

Listing 1 センシングレート決定アルゴリズムの疑似コード

```
procedure UpdateControlRate(observedTemp, forecastTemp,
      timestamp):
    sampleBuffer.insert(observedTemp, timestamp)
    sampleBuffer.evictExpired()
    meta.refresh(sampleBuffer.count, sampleBuffer.
     window, sampleBuffer.age)

    absErr <- abs(observedTemp - forecastTemp)
    ewma <- alpha * absErr + (1 - alpha) * state.
     prevEwma
    sigmaDay <- StdDev(sampleBuffer.values)
    smoothErr <- beta * absErr + (1 - beta) * state.
     prevSmoothErr
    // smooths system error trend
    r <- NormalizeEnergy(ewma, sigmaDay, smoothErr)
    sErr <- exp(-r)

    if meta.count < minSamples or not GateSatisfied(
     meta):
        return HoldPreviousRate(timestamp)

    if sErr < highThreshold:
        rate <- HIGH          // escalate or hold
    else if sErr < mediumThreshold:
        rate <- MEDIUM        // de-escalate or hold
    else:
        rate <- LOW           // de-escalate to save
     energy

    LogRateChange(timestamp, rate, state.prevRate, meta
     .count)
    ExportCsv(timestamp, rate, meta.count, sErr, ewma,
     sigmaDay)

    state.prevEwma <- ewma
    state.prevSmoothErr <- smoothErr
    state.prevRate <- rate
    return rate
```

Listing 1 は，新規サンプルをバッファへ投入してメタ情報を更新し，絶対誤差を起点に EWMA や日次分散を逐次計算して正規化誤差比 r とスコア s_{Err} を導出する手続を示している．ゲート条件を満たさない場合は既存レートを保持し，満たした場合のみ閾値比較で HIGH/MEDIUM/LOW を切り替えることで，図示したフローの「計測→誤差解析→レート決定→ログ／ CSV 出力」を擬似コード化している．

1 行目: ‘UpdateControlRate‘が観測温度，予測温度，時刻を引数に受け取ることを宣言している．

2 行目: 新しい観測温度データ (サンプル) を ‘sampleBuffer‘へ挿入し，直近データを蓄積する．

3 行目: 有効期限切れの古いサンプルをバッファから除去し，統計ウィンドウを維持する．

4 行目: サンプル数，時間窓，経過時間といったメタ情報を再計算して最新化する．

5 行目: 読みやすさのために誤差計算との区切りを入れている．

6 行目: 予測値と実測値の差の絶対値を ‘absErr‘として計算し，誤差指標の基礎とする．

7 行目: 直前の EWMA と今回の絶対誤差を係数 ‘alpha‘で重み付けし，平滑化済み誤差を更新する．

8 行目: バッファ内サンプルの標準偏差 ‘sigmaDay‘を求め，日内変動幅を把握する．

9 行目: 別係数 ‘beta‘でシステム誤差を平滑化し，緩やかな傾向変化を抽出する．

11 行目: EWMA，標準偏差，平滑誤差を入力に正規化エネルギー比 ‘r‘を計算する．

12 行目: ‘r‘から指数関数でスコア ‘sErr‘を算出し，後段の閾値判定に供する．

13 行目: 判定ブロックの前に空行を挿入して可読性を高めている．

14 行目: サンプル数が閾値未満かゲート条件が偽であれば過去レートを維持してリターンする．

15 行目: レート決定部分との区切りを示す空行である．

16 行目: スコアが高閾値未満なら HIGH レートを選択し，高頻度センシングを継続する．

17 行目: 高閾値以上，中閾値未満の範囲では MEDIUM レートへ緩やかに遷移する．

18 行目: それ以外のケースでは LOW レートを設定し，消費電力を優先する．

19 行目: ログ処理との区切りとなる空行である．

20 行目: 現在の判定結果と過去レート，サンプル数を記録し，トレース性を確保する．

21 行目: CSV 出力を通じて時刻，レート，誤差統計を外部解析向けに保存する．

22 行目: 状態更新ブロックとの区切りを担う空行である．
23 行目: 次回計算用に最新 EWMA を保持し，再帰的な平滑化を成立させる．
24 行目: 最新の平滑誤差を保存し，システム誤差のトレンドを継続して追跡する．
25 行目: 決定した制御レートを状態として格納し，ヒステリシス判定に活用する．
26 行目: 今回算出したレートを呼び出し元へ返却し，処理を終了する．

本稿の提案方式を 2 つのセクションに分けて説明する．

センシングレート変更

本セクションではセンシングレート変更手法について述べる．本提案手法では，天気予報と実測値にもとづいてセンシングレートを決定と変更するセンシングレート変更手法を用いる．

本提案手法は，観測環境の変動と予測精度に応じてセンシングレートを動的に調整し，異常時の高頻度監視と平常時の省電力運用を両立する．サーバー側で計算した次周期の推奨間隔をデバイスに応答として返却し，デバイス側ではその間隔でセンシングを行う．サーバー側で気象予報と実測値の乖離にもとづく誤差指標 sErr と前回レートを入力として，ヒステリシス付きの決定則により HIGH，MEDIUM，LOW のいずれかを選択する．基本決定は閾値比較で行い，境界付近の振動を避けるために降格側の閾値を高めに設定する．

レート決定には共有分析モジュールが用いる誤差解析アルゴリズムを採用する．予測温度を forecastC，観測温度を observedC とすると，絶対誤差は式 1 で定義する．式 1 は，予測温度と実測温度の差を絶対値で捉え，局所的な予測誤差の大きさをスカラー量として抽出する．

$$|e| = |\text{forecastC} - \text{observedC}| \tag{1}$$

= 式 2 は，過去 24 時間の観測値に対する標準偏差を計算し，日次温度変動（環境ノイズ）のスケールを推定する．

$$\sigma_{\text{day}} = \sqrt{\sum_{i}(t_i - \mu)^2/(n-1)} \tag{2}$$

式 3 は，最新誤差と履歴平均を指数移動平均で合成し，係数 (α) で反応速度を調整する．

$$\text{EWMA}_{\text{new}} = \alpha|e| + (1-\alpha)\text{EWMA}_{\text{old}} \tag{3}$$

を使用し，初回は $|e|$ で初期化する．

式 4 は，平滑化誤差を標準偏差または下限値で正規化し，日次変動と比較した相対的な誤差の大きさを示す．

$$r = \text{EWMA}/\max(\sigma_{\text{day}}, \sigma_{\text{floor}}) \tag{4}$$

誤差スコア $s_{\text{Err}} = \exp(-r)$ を得る．このスコアが $s_{\text{Err}} < 0.45$ であれば HIGH，$0.45 \le s_{\text{Err}} < 0.70$ であれば MEDIUM，$0.70 \le s_{\text{Err}}$ であれば LOW を選択する．降格時にはヒステリシスを適用し，HIGH→MEDIUM には $s_{\text{Err}} > 0.55$，MEDIUM→LOW には $s_{\text{Err}} \ge 0.80$ を要求する．

省エネルギー経路選択

本セクションでは，省エネルギー経路選択手法について述べる．本提案手法では，各ノード間での 1 回の通信にかかる消費エネルギー量の総和を通信コストとして，各ノードが負担する通信コストが全ノードで平滑化されるように経路選択を行う．

図 2 は，マルチホップネットワークのトポロジー例を示している．目標ノードを Node4 とした場合，距離の近い Node2，Node3 の 2 つのノードを介して Node4 へデータを送信する経路 (Route1) と，距離が離れているが直接 Node4 へデータを送信する経路 (Route2) が存在する．この時，Route1 の通信コストは $C_1 + C_2 + C_3$，Route2 の通信コストは C_4 である．

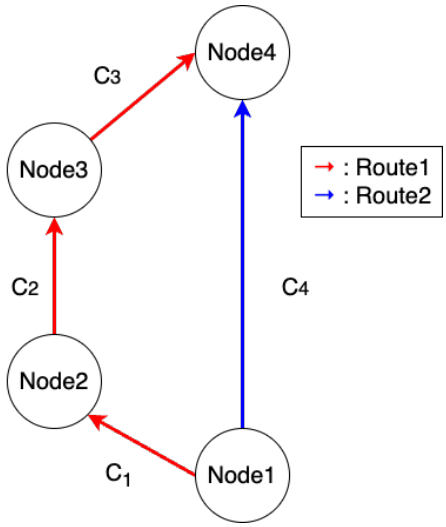

図 2 トポロジー例

解析に使用する関数

3.0.1 エネルギー消費モデル

ノード i の総エネルギー消費量 E_i は以下の式 5 で表される．

$$E_i = E_{sense} \cdot N_{sense} + E_{tx} \cdot N_{tx} + E_{rx} \cdot N_{rx} + E_{sleep} \cdot T_{sleep} \tag{5}$$

ここで，E_{sense} はセンシング 1 回あたりのエネルギー消費量，N_{sense} はセンシング回数，E_{tx} と E_{rx} はそれぞれ送信と受信 1 回あたりのエネルギー消費量，N_{tx} と N_{rx} は送信と受信回数，E_{sleep} はスリープ時の単位時間あたりのエネルギー消費量，T_{sleep} は総スリープ時間である．

動的センシングレート $R(t)$ を導入することで，センシ

ング回数は式 6 で表される．

$$N_{sense} = \int_0^T R(t)dt \tag{6}$$

となり，固定レート R_{fixed} と比較して，以下の条件を満たすとき動的センシングレートの方が省エネルギー効果が得られる．式 7 は，動的レートの積算回数が固定レート運用より少ない場合に省エネルギー効果が得られる条件を示す．

$$\int_0^T R(t)dt < R_{fixed} \cdot T \tag{7}$$

3.0.2 最適経路選択の定式化

ノード i からサーバーまでの経路 P_i におけるエネルギーコスト $C(P_i)$ は式 8 で表される．式 8 は，経路 (P_i) を構成する各ホップの送受信エネルギーを積算し，ルート別の総コストを定義する．

$$C(P_i) = \sum_{j \in P_i} (E_{tx,j} + E_{rx,j}) \cdot R(t) \cdot T \tag{8}$$

全ノードのエネルギー消費を平準化する目的関数は式 9 である．

$$\min \max_{i \in N} C(P_i) \tag{9}$$

ユースケース・シナリオ

本稿の提案方式は農地での環境モニタリングに適用することを想定している．農地では，作物の成長監視，病害虫の発生監視，品質管理，熱害防止，霜害防止を目的とし，気温，湿度，土壌湿度，監視画像データを収集する．

実際の IoT デバイスのデータ収集の例を以下に示す．ブドウ農園では短期間の気候変化に対応し，適切な防除を行う必要がある．図 3 は本稿での農園でのセンシング例を示している．

図 3 のように，本システムは農地に配置された IoT デバイスが環境データを収集し，マルチホップネットワークを利用して複数のノードを介しながらサーバーへデータを送信する．その後，サーバーに蓄積されたデータはユーザーによって可視化され，農業生産の管理や農作物の品質管理に活用される．

具体的な IoT デバイスを用いたセンシングソリューションの例として，ワイン用ブドウの品種であるカベルネ・ソーヴィニョンの栽培農地があげられる．ブドウ農園では，短時間での天気の変化に迅速に対応し，防除対策を適切に行うことが求められる．特に，2~3 時間での気温変化はブドウの成長に大きな影響を及ぼす病原菌の発生を招く可能性があるため，リアルタイムでの環境監視とデータ解析が重要である．

温度変化の時間スケールに関して，作物の生育環境では，

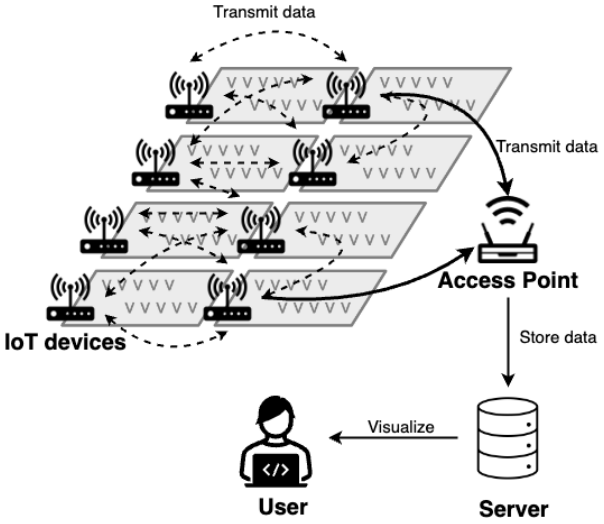

図 3 農園でのセンシング例

数分～数十分単位で変化することもあれば，数時間単位で緩やかに変わることもある．どの変化を捉えるかによって，センシングレートが決定される．例えば，霜発生や冷え込みでは短時間の変動がカギになるため，5～10 分の短いレートを選ぶこともある．一方，日中の緩やかな変化だけを目的とするなら 1 時間毎でも十分とされる．

商業リンゴ園では，果実表面温度と周辺気象をリアルタイムで監視するために，各ノードは 5 分ごとにデータを取得する．果実温度は短期間で急激に上昇するため，高頻度計測で日焼け発生前に迅速な冷却を行う必要がある．日焼けは日中の数十分単位で発生するため，5 分という高頻度が正当化されている [13].

4. 実装

以下に提案手法で実装が必要な主要コンポーネントと機能を示す．

Server(Render)

- Backend
  - ESP32 からの温度データ取得
  - データベースへの温度データの保存
  - センシングレートの決定
- Frontend
  - 実測データ，予測データのグラフ表示
  - csv データのダウンロード
- Cron Job
  - OpenWeather API からの気象予測データの定期取得
  - OpenWeather API からの過去の気象データの定期取得

Sensor Node

- センサーデータの取得
- センサーデータの送信
- センシングレート変更

システム構成の概要を図 4 に示す．ESP32 はサーバー

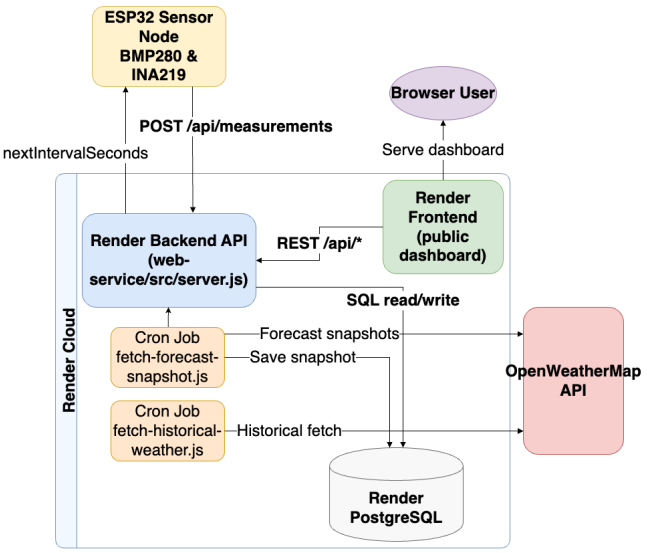

図 4　システム概要図

からセンシングレートを受信後，Wi-Fi の停止処理を行い，ディープスリープ状態に入る．基本経路は machine.deepsleep(nextIntervalSeconds*1000) によるディープスリープであり，アイドル電流を最小化できる．直近で採用したセンシングレートは RTC メモリへ保存し，電源断や初期化直後に Wi-Fi 接続が遅延しても前回のセンシングレートで動作を継続できる．

バックエンドはセンシングデータを受信後, OpenWeather API から取得した予測気温と実測値の誤差解析とセンシングレート決定を行う．決定されたセンシングレートを ESP32 へ返す．レートの決定はイベントログとして JSON 形式で記録する．

5.　評価実験

実験環境

センシングレート変更の実験では，OpenWeatherMAP の Weather API から 3 時間毎に取得した東京都八王子市の天気予報を用いて，センシングレートを変更した．実験期間は 2025 年 10 月 13 日〜2025 年 10 月 15 日と 2025 年 10 月 17 日〜2025 年 10 月 20 日の 2 回実施した．使用機器，環境は以下の通りである．

- ESP-WROOM-32 1 台
- INA219 1 台
- 通信規格: Wi-Fi
- 送信データ (約 210[byte])
  - deviceId
  - temperature
  - timestamp
  - voltage
  - current
  - power
- センシングレートマッピング
  - HIGH: 1[min]
  - MEDIUM: 5[min]
  - LOW: 10[min]
- センシングレート決定のための sErr 閾値
  - HIGH: $s_{\mathrm{Err}} < 0.45$
  - MEDIUM: $0.45 \leq s_{\mathrm{Err}} < 0.70$
  - LOW: $0.70 \leq s_{\mathrm{Err}}$

実験結果と分析

評価では，センシング回数と稼働時間の比較を行う．

実験 1: 2025 年 10 月 13 日〜2025 年 10 月 15 日

図 5 は 2025 年 10 月 13 日〜15 日の ESP32 による実測温度，OpenWeatherAPI が提供する実測温度，予測温度の比較を示している．

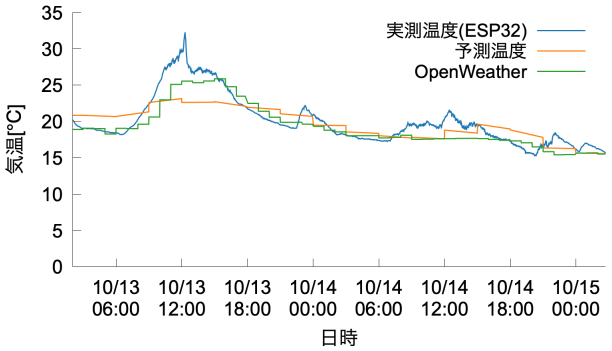

図 5　2025 年 10 月 15 日の実測データ，OpenWeatherAPI データの比較

図 5 は横軸が 2025 年 10 月 13 日〜15 日の時刻，縦軸が気温 [℃] であり，ESP32 実測・OpenWeather 実測・予測の 3 系列を同一時間軸で比較する．

この実験では，標準偏差の算出時のサンプル数を直近 48 個のデータを用いて次回センシングレートの決定を行なった．その結果，図 6 と図 7 に示すように，誤差が 0 に近づかない限り sErr 値が上昇せず，結果として全体的な sErr 値が低くなり，HIGH でのセンシングが多くなった．詳細なそれぞれのセンシングレートの出現回数は表 1 に示す．

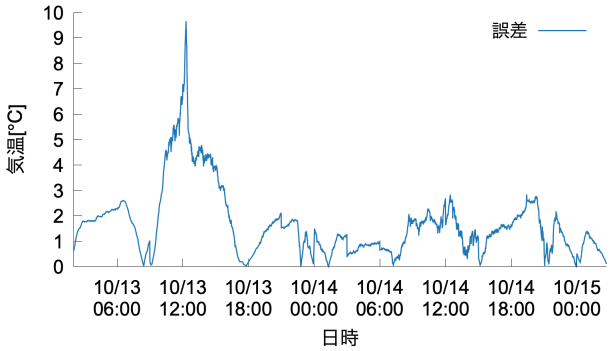

図 6　2025 年 10 月 15 日の気温データと予測の誤差

図 6 は横軸が同期間のタイムスタンプ，縦軸が予測と実測の絶対誤差 [℃] で，時間経過に伴う誤差の大きさを示す．

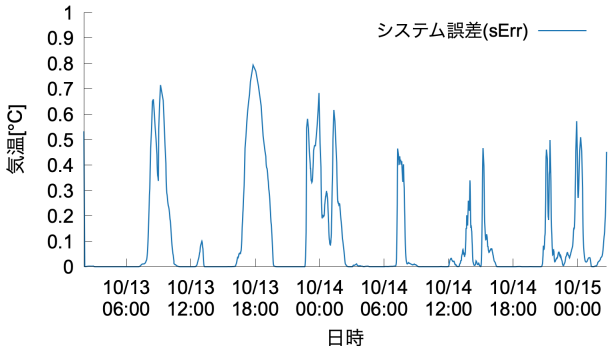

図 7　2025 年 10 月 15 日の気温データでの sErr

図 7 は横軸が 2025 年 10 月 13 日〜15 日の測定時刻，縦軸が誤差スコア s_{Err} で，誤差指標の推移とレート判定域を視覚化している．

表 1 は，2025 年 10 月 13 日〜15 日の実験におけるセンシングレート出現回数の内訳であり，HIGH が全体の大半を占め（2450 回），MEDIUM が 50 回，LOW が 5 回にとどまった結果，標準偏差計算が 48 サンプル程度と小さかったため誤差指標が敏感に反応し続け，総センシング回数は 2505 回に達したことを示している．

表 1　センシング種別 内訳

HIGH	2450 回
MEDIUM	50 回
Low	5 回
Total	2505 回

実験 2: 2025 年 10 月 17 日〜2025 年 10 月 20 日

図 8 は 2025 年 10 月 17 日〜20 日の ESP32 による実測温度，OpenWeatherAPI が提供する実測温度，予測温度の比較を示している．

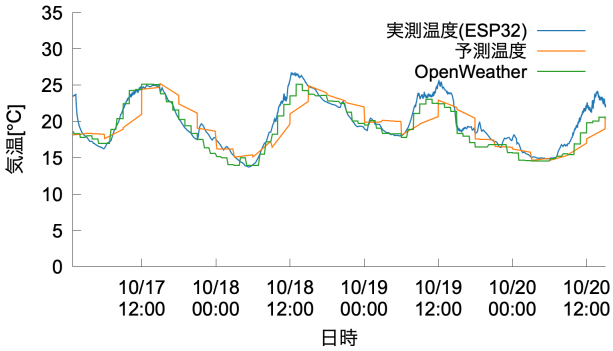

図 8　2025 年 10 月 20 日の実測データ，OpenWeatherAPI データの比較

図 8 は横軸が 2025 年 10 月 17 日〜20 日の時刻，縦軸が気温 [℃] であり，ESP32 実測・OpenWeather 実測・予測を日付の進行に沿って比較する．

この実験では，標準偏差の算出時のサンプル数を直近 24 時間のデータを用いて次回センシングレートの決定を行なった．実験 1 で行なった実験より標準偏差を算出するためのサンプル数が大幅に増加する．24 時間中に取得したデータポイントを使用するため，計算に使用されるサンプルの個数は変動する．その結果，図 9 と図 10 に示すように，誤差の変化にやや鈍感になる変わりに MEDIUM や LOW での運用回数が増加し，全体的なセンシング回数を減らすことができた．詳細なそれぞれのセンシングレートの出現回数は表 1 に示す．

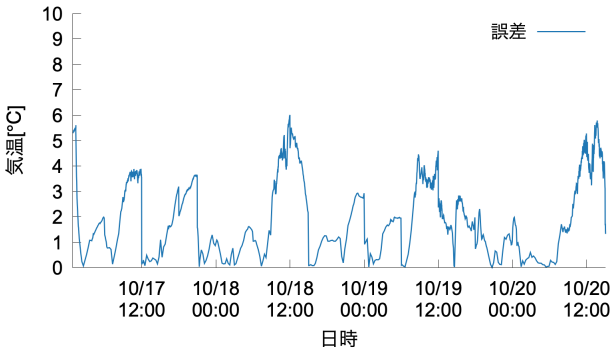

図 9　2025 年 10 月 20 日の気温データと予測の誤差

図 9 は横軸が同期間のタイムスタンプ，縦軸が絶対誤差 [℃] で，24 時間分のサンプルを反映した誤差振る舞いを示す．

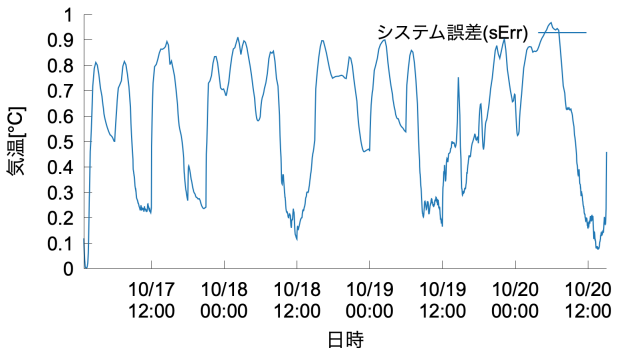

図 10 2025 年 10 月 20 日の気温データでの sErr

表 2 は，2025 年 10 月 17 日〜20 日の実験で算出したセンシングレート別の発生回数をまとめており，HIGH が 1383 回まで減少し，MEDIUM が 260 回，LOW が 159 回へ増加したことで，総センシング回数が 1802 回に削減され，誤差評価に 24 時間分のデータを用いたことでレート切替が落ち着き省電力運用に寄与したことを示している．

表 2 センシング種別 内訳

HIGH	1383 回
MEDIUM	260 回
Low	159 回
Total	1802 回

実験 1 での ESP32 の総稼働時間は 2750[min]，実験 2 での ESP32 の総稼働時間は 4273[min] であり，実験 2 の方が約 1.55 倍長く稼働した．また，センシング回数は 703 回分削減された．

6. 議論

本稿で提案した動的センシングレート変更手法について，実験結果から得られた知見をもとに議論する．

センシングレート変更による省エネルギー効果について，2025 年 7 月 4 日の実験では，従来の 15 分固定間隔によるセンシング（96 回）と比較して，提案手法では 54 回のセンシングに削減された．これは 42 回の削減効果を示しており，バッテリー消費量の大幅な改善が期待できる．特に，環境変化が少ない時間帯に Ultra Low（60 分間隔）レートを適用することで，不要なデータ収集を効果的に抑制できることが確認された．

センシングレート変更による省エネルギー効果について，行なった 2 回の実験では，データポイントの数が多いほど実測温度と予測温度の誤差に対して敏感に反応せず低レートでのセンシングが継続できることが確認された．

ただし，むやみにセンシングレートを下げることが必ずしも良いとは限らない．作物の防除に必要な適切なアラートを出すためには，必要なタイミングで正確な値を取得することが重要である．センシングレートを下げることによる省電力化と指定温度を取得する可能性の向上はトレードオフの関係にある．したがって，実際にアラートを出すべき気温に達したタイミングを適切に検出できたかを評価することで提案の有用性をさらに高められる．

実験で使用した天気予報データ（OpenWeatherMAP Weather API）は実際の農業現場での活用を想定した現実的なデータソースである．しかし，天気予報の精度に依存する部分があり，予報と実測値の乖離が大きい場合には適切なセンシングレート制御が困難になる可能性がある．

7. おわりに

本稿では，マルチホップセンサーネットワークにおけるデータ収集の可能時間の延長を目的として，気象予測と実測値の誤差にもとづく動的センシングレート制御と通信コストの平準化を図る省エネルギー経路選択を組み合わせ，マルチホップネットワークセンサーネットワークのデータ取得が可能な時間を伸ばす手法を提案した．ESP32 ノードと Render サーバーで構成したシステムを用い，OpenWeatherMAP の予報を取り込んだ長期運用実験を実施した．提案手法の sErr 算出に使用するデータポイントの個数を 48 個の固定値から過去 24 時間分に変更することで，センシング回数を 703 回削減しながら稼働時間を約 1.55 倍へと延伸でき，高頻度監視と省電力の両立が可能であることを示した．

今後は，ユースケースをもとにした気温の検出可/不可の評価，省エネルギー経路選択の実ネットワーク実証，複数ノードによるホップ数とリンク品質変動下での制御安定性の評価を進める．

参考文献

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y. and Cayirci, E.: A survey on sensor networks, *IEEE Communications Magazine*, Vol. 40, No. 8, pp. 102–114 (online), DOI: 10.1109/MCOM.2002.1024422 (2002).

[2] Conti, M. and Giordano, S.: Mobile ad hoc networking: milestones, challenges, and new research directions, *IEEE Communications Magazine*, Vol. 52, No. 1, pp. 85–96 (online), DOI: 10.1109/MCOM.2014.6710069 (2014).

[3] Pešović, U. M., Mohorko, J. J., Benkič, K. and Čučej, Ž. F.: Single-hop vs. Multi-hop–Energy efficiency analysis in wireless sensor networks, *18th telecommunications forum, TELFOR* (2010).

[4] Arora, S., Nijhawan, G., Verma, G. and Patel, R. J.: A systematic survey on various energy harvesting systems for WSN applications, *2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, pp. 1–5 (online), DOI: 10.1109/ICIERA53202.2021.9726530 (2021).

[5] Daupayev, N., Engel, C. and Hirsch, S.: Two-to-One Trigger Mechanism for Event-Based Environmental Sensing, *Sensors*, Vol. 25, No. 13 (online), DOI: 10.3390/s25134107 (2025).

[6] Omari, M. and Laroui, S.: Simulation, comparison and

analysis of Wireless Sensor Networks protocols: LEACH, LEACH-C, LEACH-1R, and HEED, *2015 4th International Conference on Electrical Engineering (ICEE)*, pp. 1–5 (online), DOI: 10.1109/INTEE.2015.7416826 (2015).

[7] Song, L., Song, Q., Ye, J. and Chen, Y.: A Hierarchical Topology Control Algorithm for WSN, Considering Node Residual Energy and Lightening Cluster Head Burden Based on Affinity Propagation, *Sensors*, Vol. 19, No. 13 (online), DOI: 10.3390/s19132925 (2019).

[8] Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A. and Chandrakasan, A.: Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks, *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, New York, NY, USA, Association for Computing Machinery, p. 272–287 (online), DOI: 10.1145/381677.381703 (2001).

[9] Leu, J.-S., Chiang, T.-H., Yu, M.-C. and Su, K.-W.: Energy Efficient Clustering Scheme for Prolonging the Lifetime of Wireless Sensor Network With Isolated Nodes, *IEEE Communications Letters*, Vol. 19, No. 2, pp. 259–262 (online), DOI: 10.1109/LCOMM.2014.2379715 (2015).

[10] Omari, M. and Laroui, S.: Simulation, comparison and analysis of Wireless Sensor Networks protocols: LEACH, LEACH-C, LEACH-1R, and HEED, *2015 4th International Conference on Electrical Engineering (ICEE)*, pp. 1–5 (online), DOI: 10.1109/INTEE.2015.7416826 (2015).

[11] Priyadarshi, R., Singh, L., Randheer and Singh, A.: A Novel HEED Protocol for Wireless Sensor Networks, *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 296–300 (online), DOI: 10.1109/SPIN.2018.8474286 (2018).

[12] Shah, I. K., Maity, T., Dohare, Y. S., Tyagi, D., Rathore, D. and Yadav, D. S.: ICIC: A Dual Mode Intra-Cluster and Inter-Cluster Energy Minimization Approach for Multihop WSN, *IEEE Access*, Vol. 10, pp. 70581–70594 (online), DOI: 10.1109/ACCESS.2022.3188684 (2022).

[13] Ranjan, R., Khot, L. R., Peters, R. T., Salazar-Gutierrez, M. R. and Shi, G.: In-field crop physiology sensing aided real-time apple fruit surface temperature monitoring for sunburn prediction, *Computers and Electronics in Agriculture*, Vol. 175, p. 105558 (online), DOI: https://doi.org/10.1016/j.compag.2020.105558 (2020).

