2025年度
修士論文

SN開発におけるデバッグシステムの開発

指導教員　　アサノ デービッド　　教授
不破 泰　　肩書き

2025年1月30日提出

信州大学大学院 総合理工学研究科 工学専攻 情報数理・融合システム分野
24W6047A
辻村篤志

あらまし

あらましを書く。

目 次

図 目 次

表 目 次

第 1 章

はじめに

はじめにを書く。

1.1 研究背景

近年、無線センサネットワーク（Wireless Sensor Network：WSN）やIoT（Internet of Things）は、環境モニタリングやスマートシティ、インフラ監視、農業、ヘルスケアなど、さまざまな分野での応用が期待されている。実際，IoT 全体の接続デバイス数は急速に増加している。IoT Analytics によれば，2023 年時点で約 166 億台，2024 年には約 188 億台に達し，2030 年には約 400 億台に到達すると予測されている [1]。また，WSN の普及も市場規模の面から拡大を続けており，Grand View Research の報告では，産業用 WSN（Industrial WSN）の市場は 2023 年に約 51.9 億ドルと推定され，2024 年から 2030 年にかけて年平均 12.1%の成長が見込まれている [2]。

しかし、これらのシステムを構築するためには、複雑な通信プロトコルの設計やデータ処理の設計および実装が求められ、開発者には幅広い専門知識と多大な工数が必要となる。さらに、無線通信環境の構築やマイコン設定など導入時のハードルも高く、これらの研究や開発の多くはシミュレーション環境やエミュレータ上での検証にとどまることが多い。しかし、シミュレーション環境では実機特有の挙動や外部環境の影響を完全に再現することが難しく、実機での検証が不可欠である場合も多く、実機でのプロトコル検証を容易にするための支援環境の整備が求められている。

1.2　当研究室における先行研究

当研究室の先行研究では、状態遷移図上で無線通信プロトコルの動作を定義し、そこから自動生成したプログラムを汎用ハード上で動作させ、プロトコルを実機で動作検証できる環境が構築されていた（旭ら [3]、小林ら [4]）。

1.3　先行研究の課題

このシステムにより基本的な通信プロトコルの動作検証が可能となったが、デバッグ方法はコンソールへのログ出力に依存しており、通信処理が高速に進むため逐次的な状態遷移を追跡することが難しかった。その結果、複雑な動作を含むプロトコルに対しては、異常動作の原因を把握しづらく、開発効率や教育的利用の観点では十分でない点が課題として残されていた。

1.4　本研究の目的

本研究の目的は、先行研究で課題となっていたデバッグ効率の低さを改善し、実機でのプロトコル検証をより効果的に行える環境を実現することである。そのために、無線通信デバイス上で実行された動作をログとして出力し、それを可視化・ステップ実行できる仕組みを開発した。これにより、プロトコルの動作を逐次的に把握でき、異常動作の原因究明を容易にするとともに、教育やチーム開発、応用実証に適した支援環境を提供することを目指す。

1.5　本論文の構成

本論文は全 7 章から構成されている。第 1 章では、本研究の背景として当研究室におけるこれまでの研究の流れを整理し、先行研究で顕在化した課題を明らかにした上で、本研究の目的を述べる。第 2 章では、当研究室における先行研究システムについて、その構成や実現されてきた機能を整理するとともに、運用およびデバッグの観点から課題をまとめる。第 3 章では、先行研究の課題を踏まえ、本研究で提案するシステムの概要と設計方針について述べる。ここでは、全体構成および各コンポーネントの役割を中心に説

明する。第4章では、提案システムの設計および実装について述べ、特にデバッグ支援を目的としたログ取得方式や状態遷移の可視化手法について詳述する。第5章では、提案システムの操作例を示し、実際の利用を通してどのような情報が得られるかを説明する。第6章では、従来手法との比較を通じて提案システムの有効性を評価し、その結果について考察を行う。最後に第7章では、本研究のまとめを行うとともに、今後の課題および展望について述べる。

第 2 章

先行研究の提案システム

2.1　システムの概要

先行研究では、無線通信プロトコルの設計・実装・動作検証を支援するシステムが提案されてきた。このシステムは主に次の二つのコンポーネントから構成されている。第一に、UML の状態遷移図を用いてプロトコルの動作を定義し、そこからコードを自動生成するコンポーネントである。第二に、生成されたコードを実行するための環境（ハードウェア）を提供するコンポーネントである。これにより、状態遷移図で定義されたプロトコルが実機上で動作し、その動作を検証できる環境が提供されてきた。システムの概要を図 2.1 に示す。

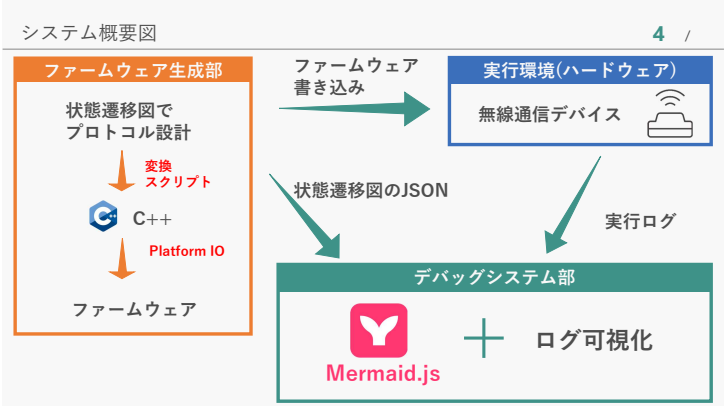

図 2.1: 先行研究システムの概要図

2.2　実現事項

先行研究システムでは、以下の主要な機能が実現されている。

- 状態遷移図からのコード自動生成：UML の状態遷移図を用いてプロトコルの動作を定義し、そこから C 言語コードを自動生成する機能が提供されている。これにより、プロトコル設計者は視覚的に動作を定義でき、手動でのコーディング作業を削減できる。
- 実機上での動作検証：生成されたコードは、無線モデム搭載のマイコンを組み合わせたハードウェア上で実行される。これにより、シミュレーション環境では捉えきれない実機特有の挙動を検証できる。

2.3　課題

先行研究システムには以下の課題が存在する。

- 状態遷移図の記法の制約：先行研究における状態遷移図からのコード自動生成は、特定の記法に依存しており、複雑なプロトコル動作を表現する際に制約がある。これにより、設計者が意図する動作を完全に反映できない場合がある。
- デバッグ効率の低さ：デバッグ方法がコンソールへのログ 出力に依存しており、通信処理が高速に進むため逐次的な状態遷移を追跡することが難しい。これにより、複雑な動作を含むプロトコルに対しては、異常動作の原因を把握しづらい。

これらの課題を解決するために、本研究ではデバッグ効率の向上に焦点を当て、実機でのプロトコル検証をより効果的に行える環境を提案する。なお、状態遷移図からのコード生成部については、[5] で改善が図られているため、そちらを参照されたい。

第 3 章

提案システムの概要と設計方針

本研究では，先行研究システムにおいて課題となっていたデバッグ効率の低さを改善することを目的として，新たなデバッグ支援システムを提案する。本章では，提案システムの概要と設計方針について述べる。はじめに，本研究で対象とする課題を整理し，次に設計方針およびシステム全体の構成について説明する。

3.1　設計方針

先行研究システムでは，状態遷移図を用いて通信プロトコルを設計し，実機上で動作検証を行うことが可能であった。一方で，デバッグ方法は主に図 3.1 のように、コンソールへのログ出力に依存しており，状態遷移の流れや内部動作を逐次的に把握することが難しいという課題があった。

本研究では，この課題を踏まえ，実機上で実行される通信プロトコルの内部動作をより直感的に把握できるデバッグ支援環境の構築を設計方針とした。具体的には，プロトコルの動作を状態遷移として捉え，その実行状況をログとして取得・可視化することで，動作の流れを段階的に確認できる仕組みを目指す。

また，本研究のシステムは，先行研究で構築されたプロトコル設計および実機検証の枠組みを活用することを前提とし，既存システムとの親和性を保ちながら改善を行うことを重視した。これにより，研究室内での継続的な利用や拡張が容易な構成とすることを設計上の方針とした。

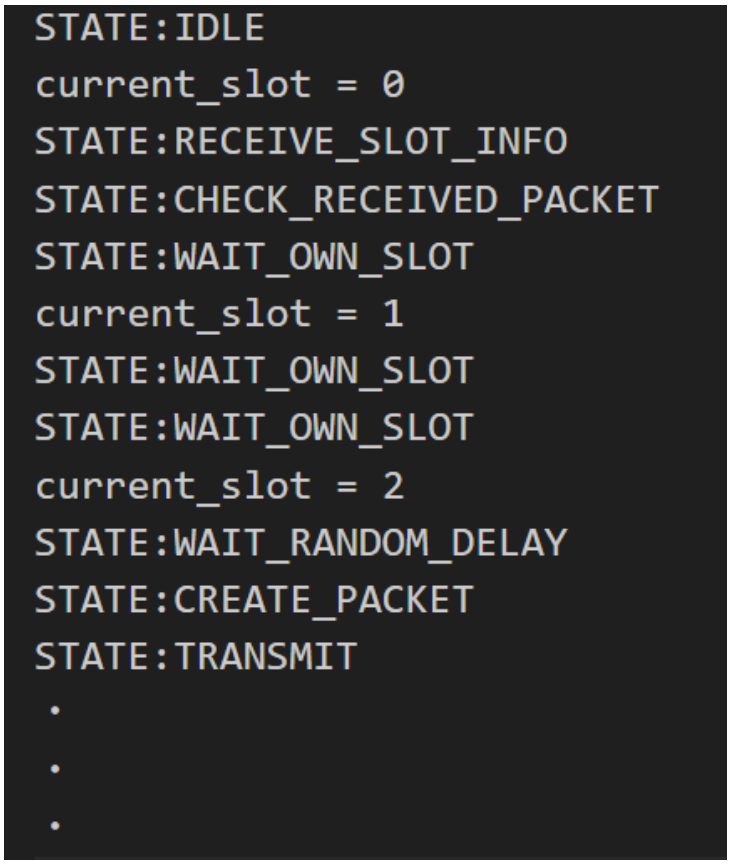

図 3.1: 従来のデバッグ方法

3.2　システムの構成

システム全体の構成を図 3.2 に示す。

提案システムは，先行研究システムに対してデバッグ支援機能を付加する形で構成されている。全体としては，通信プロトコルを実行する組込みシステムと，その動作を外部から確認するためのデバッグ支援システムからなる。

組込みシステム側では，状態遷移に基づいて通信プロトコルが実行され，その実行過程に関する情報がログとして出力される。一方，デバッグ支援システム側では，出力されたログを収集し，状態遷移の流れや実行状況を可視化する役割を担う。

3.3　各コンポーネントの概要

提案システムは，主に以下のコンポーネントから構成されている。

- 状態遷移図からコードを自動生成するコンポーネント

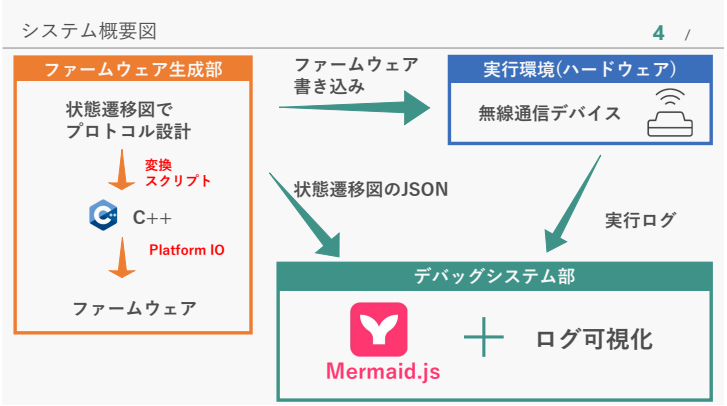

図 3.2: 提案システムの概要図

- 通信プロトコルを実行する組込みシステム
- デバッグ支援システム

3.3.1　コード生成コンポーネント

はじめに，状態遷移図からコードを自動生成するコンポーネントである。このコンポーネントは，UML の状態遷移図を解析し，通信プロトコルの動作を表現する C++コードを生成する役割を担う。これにより，設計者は視覚的にプロトコルの動作を定義でき，手動でのコーディング作業を削減できる。- 状態遷移図の構造を解析し、存在する状態や遷移、イベント、アクションを抽出する。- 抽出した情報に基づき、C++コードを生成する。- 生成されたコードは、switch-case 文を用いて状態遷移を実装している。- 生成されたコードは、組込みシステム上で実行される。

3.3.2　組込みシステムコンポーネント

本システムにおける実行環境を図 3.3 に示す。本研究では，SAMD21 マイコンと無線通信モジュールを一体化した無線通信デバイス (株式会社サーキットデザイン製どこでもモデム) に GPS モジュールを組み合わせて、一つの無線通信デバイスとして使用した。この無線通信モジュールは 429MHz 帯で動作する汎用的なものであり，技術基準適合証明 (技適) を取得しているため，実際に電波を飛ばしながら通信プロトコルの検証を行うこ

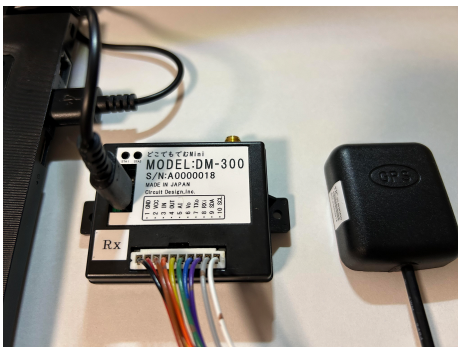

図 3.3: どこでモデム (左) と GPS モジュール (右)

とができる。また，429MHz 帯は中山間地域において回折性能に優れており，低消費電力で長距離通信が可能な LPWA（Low Power Wide Area）通信に適している。この特性を活かすことで，登山者見守りシステムなどへの応用も期待できる。さらに，本デバイスはマイコンと通信モジュールが一体化しているため，外部配線や追加ハードウェアを必要とせず，開発や実験環境の構築を容易に行うことが可能である。加えて，GPS モジュールを併用することで，位置情報の取得および 1PPS 信号による高精度な時刻同期を実現し，複数ノード間での時刻同期による無線通信プロトコルの実現を可能としている。

3.3.3　デバッグ支援システムコンポーネント

最後に，本研究の中核であるデバッグ支援システムである。このコンポーネントは，組込みシステムから出力されるログを収集し，状態遷移や内部動作を可視化する役割を担う。これにより，従来は把握が困難であったプロトコルの動作過程を確認できる。このコンポーネントは、次章で詳述する。

第 4 章

設計及び実装

本章では、提案システムの設計および実装について述べる。特に、デバッグ支援システムの設計方針、ログ取得方式、および状態遷移可視化の手法について詳述する。

4.1 使用ソフトウェア、ツール

使用ソフトウェア、ツールについて説明する。- astah Professional: UML 設計ツール、状態遷移図の作成に使用- Visual Studio Code: コード編集、開発環境として使用- PlatformIO: 組込みシステムの開発環境、コードのビルドに使用 [6] - Node Package Manager (npm): デバッグ支援システムの開発に使用、フロントエンド

4.2 各コンポーネントの接続

各コンポーネントの接続について説明する。- 状態遷移図-¿コード生成コンポーネント: Python スクリプトを実行し、作成した astah の状態遷移図ファイルを選択、C++コードを生成、この時状態遷移図の JSON ファイルも生成- コード生成コンポーネント-¿組込みシステム：生成された C++コードを PlatformIO でビルドし、無線通信デバイスに書き込み- 組込みシステム-¿デバッグ支援システム: 無線通信デバイスからシリアル通信でログを送信（シリアルモニタで確認可能）- コード生成コンポーネント-¿デバッグ支援システム: 状態遷移図の JSON ファイルをデバッグ支援システムにアップロードし、状態遷移の可視化に使用

4.3　デバッグシステムの設計

- ブラウザベースのフロントエンドアプリケーションとして実装- React.js を使用して UI、ロジックを構築- ユーザが状態遷移図の JSON ファイルをアップロードし、組込みシステムからのログをリアルタイムで受信- Mermaid.js を使用して状態遷移図を描画 [7][8] - ログから状態遷移イベントを解析し、状態遷移図上で現在の状態をハイライト表示- ステップ実行機能を実装し、ユーザが一つずつ状態遷移を確認可能- 複数の変数を同時に監視できるウォッチ機能を実装- スライダーを用いて任意の地点のステップにジャンプ可能

4.4　ログ取得方式の設計

ログ取得方式の設計を書く。

4.5　状態遷移可視化の設計

状態遷移可視化の設計を書く。

4.6　実装概要

実装概要を書く。

第 5 章

操作例

5.1　操作例の概要

本章の目的想定する利用シナリオ。

5.2　操作例

操作の流れ（簡潔に）操作により得られるログ・可視化結果。

5.3　デバッグ支援としての有効性

操作例から確認できる利点先行研究との違い。

第 6 章

評価と考察

6.1　評価方針

評価の目的評価観点の整理

6.2　評価方法

評価環境の説明評価手順の説明

6.3　評価結果と考察

得られた結果従来手法との比較有効性に関する考察

第 7 章

まとめと今後の展望

7.1　まとめ

本研究の成果と意義を総括する。

7.2　今後の展望

今後の課題と研究の方向性について述べる。

参考文献

[1] IoT Analytics: ”State of IoT 2024: Number of Connected IoT Devices Grows 16% to 16.6 Billion”, 2024.

[2] Grand View Research: ”Industrial Wireless Sensor Network Market Size, Share & Trends Analysis Report, 2024–2030”, 2024.

[3] 旭 健汰, アサノ デービッド, 不破 泰:“無線モデムを用いたセンサネットワーク MAC プロトコル検証システムの開発”，信学技報, NS2023-147, pp.120–125, Dec. 2023.

[4] 小林 遼, アサノ デービッド, 不破 泰:“センサーネットワーク検証システムにおける遷移図を用いた MAC プロトコル設計環境の開発”，信学技報, NS2023-148, pp.126–131, Dec. 2023.

[5] 小林 侑生, アサノ デービッド, 不破 泰：

[6] PlatformIO Labs, “What is PlatformIO?”, https://docs.platformio.org/en/latest/what-is-platformio.html, 参照 Oct. 15, 2025.

[7] Mermaid.js, “Overview,” https://mermaid.js.org/intro/, 参照 Oct. 15, 2025.

[8] Mermaid.js, “State Diagram Syntax (stateDiagram-v2),” https://mermaid.js.org/syntax/stateDiagram.html, 参照 Oct. 15, 2025.

[9] Eclipse Foundation, “MQTT: Message Queuing Telemetry Transport,” https://mqtt.org/, 参照 Oct. 15, 2025.

[10] 園田 継一郎, 不破 泰, アサノ デービッド, ”無線センサーネットワークの端末・中継機における送信タイミング決定時間短縮方法の検討, ” 信学技報, NS2022-138, Dec. 2022.