一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

状態遷移図ベースのコード生成とログ可視化機能を備えたセンサネットワーク実機検証基盤の開発

辻村篤志† 小林侑生† 不破泰† アサノデービッド†

† 信州大学 〒380-0928 長野県長野市若里 4-17-1
E-mail: †24w6047a@shinshu-u.ac.jp

あらまし センサネットワークの開発は幅広い専門知識と多大な工数を要する。先行研究では、センサネットワークの動作を表す状態遷移図からコードを生成し、汎用ハード上で動作する環境が構築されてきた。本研究ではその環境を拡張し、その上で実用的な無線通信プロトコルを実装・検証した。その過程で汎用ハード上でのプロトコルの動作が確認しづらいことを課題として認識し、デバッグ用ログ出力とそれを可視化・ステップ実行できる環境を開発しデバッグ効率の向上を図った。

キーワード 無線センサーネットワーク、無線通信プロトコル、状態遷移図、コード生成、ログ可視化

Atsushi TSUJIMURA†, Yu KOBAYASHI†, Yasushi FUWA†, and David ASANO†

† Shinshu University, 4–17–1 Wakasato, Nagano-shi, Nagano 380–8553 Japan
E-mail: †24w6047a@shinshu-u.ac.jp

Abstract The development of sensor networks requires extensive expertise and significant effort. Previous research has generated code from state transition diagrams representing the behavior of sensor networks, establishing an environment that operates on general-purpose hardware. This study expands that environment and implements and verifies practical wireless communication protocols. During this process, we recognized the difficulty in confirming the operation of protocols on general-purpose hardware as a challenge and developed an environment for debugging log output and visualizing and step-executing it to improve debugging efficiency.

Key words wireless sensor networks, wireless communication protocols, state transition diagrams, code generation, log visualization

1. はじめに

1.1 背　景

近年、無線センサネットワーク（Wireless Sensor Network：WSN）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、WSN を含む IoT デバイスの数は 2020 年時点で約 130 億台に達し、年率 20 %で増加しており、2025 年には 300 億台弱に到達すると予測されている [1]. また、MDPI の報告によれば、WSN ノードの数は 2012 年の約 4.5 億台から 2022 年には約 20 億台へと拡大しており [2]、その研究および実用の両面での発展が顕著である。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設計やデータ処理アルゴリズムの実装が求められ、開発者には高度な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。

1.2 従来研究

従来研究では、無線通信プロトコルの状態遷移図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実装・検証できる環境が構築されていた（旭ら [**?**, asahi]。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、デバッグ方法はシリアル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。

1.3 目　的

本研究の目的は、従来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、状態遷移図から生成したコードの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、教育や応用実証に適した支援環境を提供することを目指す。

2. 提案システム

2.1 システム構成

本研究で開発したシステムの全体構成を図 1 に示す。本システムは、状態遷移図を入力として C++ プログラムを自動生成し、PlatformIO 上でビルドを行い、汎用ハードウェア上で実行することでセンサネットワークのプロトコルの動作検証を可能とする。さらに、通信動作の状態遷移や変数値変化の過程をログとして記録し、ブラウザ上で状態遷移の可視化およびステップ実行を行える環境を備えている。これにより、プロトコル設計から実機検証、デバッグまでを一貫して支援することが可能となる。

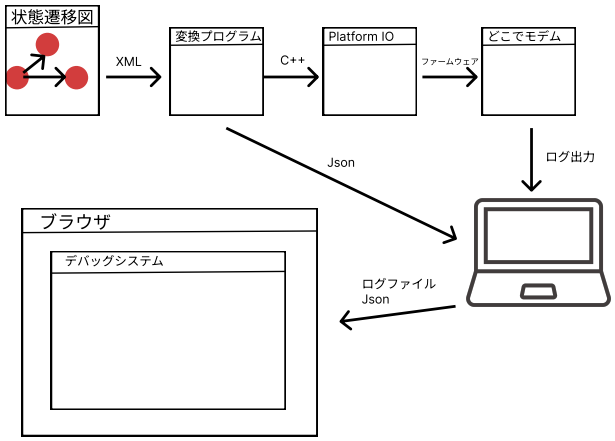

図 1 提案システムの全体構成
Fig. 1 Block diagram of the proposed system.

2.2 ファームウェア生成部

ファームウェア生成部では，無線通信プロトコルの動作を Astah Professional 上の状態遷移図として設計し，XML 形式で出力する。次に，変換プログラム（Python）を用いて状態遷移図の XML を C++ プログラムへ変換し，PlatformIO 環境でビルドすることでファームウェアを生成する。このファームウェアは 2.3 節で述べる汎用ハードウェア上で実行される。

先行研究（小林ら [4]）では，状態遷移図の各要素（通信状態・遷移・初期値設定・処理内容など）を XML ファイルから抽出し，C++ の ‘switch-case‘構造に変換するアルゴリズムが実装されている。本研究におけるファームウェア生成部は，この変換処理を利用しており，状態遷移図で設計されたプロトコル仕様を汎用ハードウェア上で動作するプログラムとして自動生成するものである。

図 2 に状態遷移図の例を，プログラム 1 に変換後の C++ プログラムを示す。状態遷移図で定義された「待機（LISTEN）」「受信（RECEIVE）」「送信（TRANSMIT）」の各状態と遷移関係が，‘switch-case‘構造によって対応づけられており，設計段階の論理構造がソースコードとして正しく反映されていることが確認できる。

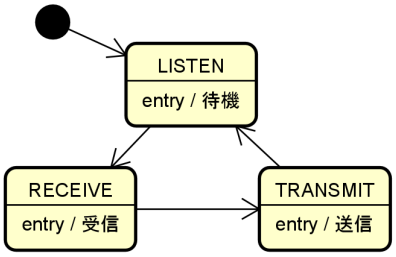

図 2 状態遷移図の例
Fig. 2 State transition diagram.

プログラム 1 変換された C++ プログラム

```
1  typedef enum { LISTEN, RECEIVE, TRANSMIT } NodeState
       ;
2
3  void protocol_main() {
4    switch (state) {
5      case LISTEN: updateState(RECEIVE); break;
6      case RECEIVE: updateState(TRANSMIT); break;
7      case TRANSMIT: updateState(LISTEN); break;
8    }
9  }
```

2.3 実行環境 (ハードウェア)

本システムにおける実行環境を図 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モデム）を使用した。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，実際に電波飛ばしながら通信プロトコルの検証を行うことができる。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，複数ノード間での無線通信プロトコルの正確な動作検証を可能としている。

2.4 デバッグシステム部

本節では，通信プロトコルの動作を可視化し，逐次的なデバッグを可能とするデバッグシステムについて述べる。本シス

図 3　どこでモデム (左) と GPS モジュール (右)
Fig. 3　Block diagram of the proposed system.

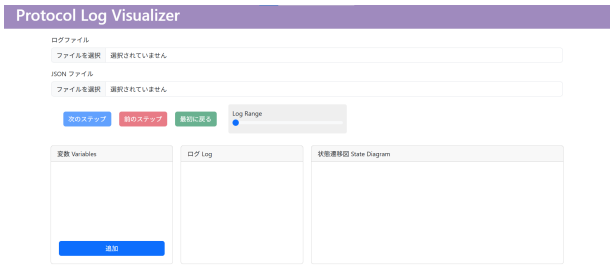

図 4　ログ可視化・ステップ実行環境の UI
Fig. 4　User interface of the log visualization and step execution environment.

テムは，通信プロトコルの状態遷移および変数変化をログとして記録し，ブラウザ上でそのログを可視化・ステップ実行できる環境を提供する。これにより，通信動作の流れを直感的に把握でき，異常動作の原因究明や設計検証を効率化することが可能となる。

2.4.1　UI　構　成

図 4 に可視化画面の構成を示す。画面上部にログファイル（テキスト）および状態遷移図ファイル（JSON）のアップロード領域を配置し，中央にステップ実行用のボタン群（次，前，最初に戻る）とスライダを設ける。下部は「変数追跡」「ログ表示」「状態遷移図描画」の 3 領域で構成し，処理の流れと内部状態を統合的に把握できる設計とした。

2.4.2　ログ出力形式

状態遷移や変数を逐次ログとして出力し、それを可視化・解析するために，ログの出力形式を統一した。具体的には，状態遷移は

```
STATE:LISTEN
STATE:RECEIVE
```

のように 1 行 1 状態で出力し，変数は

```
[VAR:current_slot=3]
[VAR:has_own_slot=true]
```

のように `[VAR:<name>=<value>]` 形式で出力する。値は数値・文字列の双方を許容し，可視化側で時間順に解析する。単純で曖昧さの少ない表記とすることで，正規表現に基づく軽量なパーサで高速に処理できる。

さらに，ログ出力の形式にばらつきが生じないよう，C++ のマクロを用いてログ出力関数を定義している。このマクロは，変数名と値を自動的に取得して `[VAR:<name>=<value>]` 形式で出力するため，開発者は変数名を明示する必要がない。例えば，ユーザは `PRINT_VAR_LOG(変数)` のように記述するだけで統一フォーマットのログを生成できる。この仕組みにより，全ての出力が一定形式で記録され，可視化システム側で確実に解析可能となる。

2.4.3　可視化・再現機構

本システムでは，記録されたログファイルと，状態遷移図の構造を記述した JSON ファイルをもとに状態の遷移を再現する。具体的には，JSON ファイルから状態名および遷移関係を抽出し，それらを Mermaid.js のフォーマットへ変換して描画処理に渡す。ログの進行に合わせて該当ノードをハイライトすることで，通信処理の時系列的な動作を視覚的に確認できる。

Mermaid.js は，テキストベースでフローチャートや状態遷移図を描画できる軽量な JavaScript ライブラリであり，HTML 上での埋め込みや動的描画に適している。本研究では，ブラウザ環境で動作し，スタイルの変更が可能であり，および構文が簡潔で自動生成との親和性が高い点から採用した。これにより，状態遷移図を表す JSON データから自動的に Mermaid 記法を生成し，`stateDiagram-v2` 形式で描画することで，設計段階の論理構造をそのままの形で可視化できる。

なお，可視化に用いる JSON ファイルは，2.2 節で述べたファームウェア生成部と同一の Python プログラムによって自動的に生成される。Astah Professional から出力された状態遷移図の XML を解析し，C++ コード生成と同時に，状態遷移関係を JSON 形式で出力している。この仕組みにより，設計情報と可視化データの不整合を防ぎ，設計・実装・可視化の一貫性を確保している。

プログラム 2 に、生成される JSON の例を示す。この JSON は，状態リストと遷移リストの 2 つの要素で構成されており，状態名およびそれらを結ぶ遷移関係を保持する。

ここで，`states` 配列は通信状態の一覧を表し，`transitions` 配列は状態間の遷移関係を示す。`from` および `to` がそれぞれ遷移元と遷移先の状態を表している。このデータ構造をもとに Mermaid.js 形式のテキストを自動生成し，ブラウザ上で状態遷移図を描画している。

変換後の Mermaid.js フォーマットをプログラム 3 に示す。このように，シンプルな JSON 構造を中間表現として用いることで，XML から生成された設計情報を軽量に扱うことができ，状態遷移構造の可視化を容易に実現している。

2.4.4　ステップ実行および変数追跡

本システムでは，通信プロトコルの動作を 1 ステップ（1 状態遷移）単位で再現できるステップ実行機能を備えている。画面上部に配置された「次のステップ」「前のステップ」「最初に

プログラム 2　状態遷移図を表す JSON ファイルの例

```
{
  "states": [
    "LISTEN",
    "RECEIVE",
    "TRANSMIT",
    "WAIT_ACK"
    ],
  "transitions": [
    {"from": "LISTEN", "to": "RECEIVE"},
    {"from": "RECEIVE", "to": "TRANSMIT"},
    {"from": "TRANSMIT", "to": "WAIT_ACK"},
    {"from": "WAIT_ACK", "to": "LISTEN"}
    ]
}
```

プログラム 3　生成された Mermaid.js フォーマットの例

```
stateDiagram-v2
[*] --> LISTEN
LISTEN --> RECEIVE
  RECEIVE --> TRANSMIT
  TRANSMIT --> WAIT_ACK
  WAIT_ACK --> LISTEN
```

戻る」ボタンにより，状態遷移を順方向または逆方向に逐次移動でき，通信処理の流れを段階的に確認することが可能である。さらに，スライダを用いることで任意のステップ位置へ直接移動でき，確認したい箇所を即座に調査できる柔軟な操作性を実現している。

状態遷移の再生にあたっては，ログの各行を 1 状態単位として扱い，時系列に沿って状態情報を更新する。ログ再生に合わせて，中央のログ表示領域，右側の状態遷移図，および左側の変数表示領域が同期して更新されるよう設計されている。これにより，ある状態における変数値や，その状態から次への遷移を同時に確認できる。また，状態遷移図上では現在の状態ノードがハイライト表示され，通信動作の進行位置を視覚的に把握しやすい構成となっている。

変数追跡機能では，ログ出力された任意の変数を選択し，その値の変化を時系列に追跡できる。追跡対象の変数は複数選択可能であり，状態変化とあわせて並行して複数の変数を観察することで，通信プロトコルの内部状態を多角的に解析できる。これらの機能により，特定条件下での変数の変化や分岐挙動を直感的に検証でき，デバッグおよび設計検証の効率向上に寄与する。

2.4.5　リアルタイム可視化

本システムの拡張として，実機で動作するファームウェアのログをリアルタイムに取得・可視化する機能を試作した。図 6 にその構成を示す。

デバイス上で動作するファームウェアは，通信動作に応じて状態遷移ログをシリアルポートへ逐次出力する。Python ス

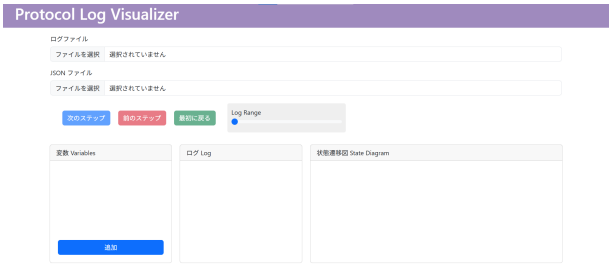

図 5　ステップ実行時の状態ハイライト例

Fig. 5　Example of state highlighting during step execution.

クリプトがこのシリアルポートを常時監視し，取得したログデータを MQTT（Message Queuing Telemetry Transport）ブローカーを介してバックエンドサーバへ送信する。MQTT は軽量な Pub/Sub 型通信方式であり，組込みデバイスとサーバ間のリアルタイム通信に適している。

バックエンドでは，Node.js 上で動作するサーバが MQTT ブローカーからログデータを受信し，Socket.IO を用いてブラウザと双方向通信を行う。これにより，取得された状態遷移情報はフロントエンドへ逐次転送され，ブラウザ上の Mermaid.js による状態遷移図に即時反映される。通信遅延は数百ミリ秒程度に抑えられ，状態の変化をほぼリアルタイムに確認できることを確認した。

この構成により，従来の「ログ取得→解析→可視化」といった事後的なデバッグ手法に対し，動作中のシステム挙動を即時に観測できる「実時間デバッグ」が可能となった。これにより，通信プロトコルの動作確認やノード間同期の評価を効率化できるだけでなく，将来的には実験中の異常検知や教育用デモ環境への応用も期待される。

一方で，状態遷移の周期が非常に短いプロトコルや高速に状態が変化するシステムにおいては，描画更新が処理速度に追従できず，視認性が低下する場合がある。このため，本機能は主に周期の長い通信プロトコルや教育目的の可視化において有効であり，高速プロトコルの解析にはログ記録・再生型の可視化機構（2.4 節）を用いることが望ましい。

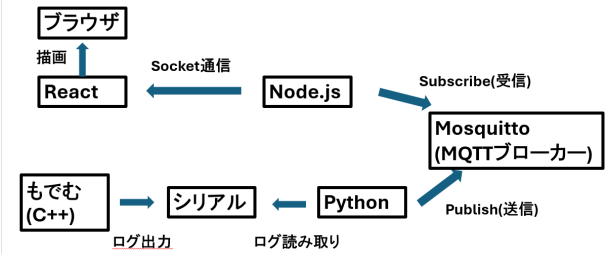

図 6　リアルタイム可視化機能の構成

Fig. 6　System structure of the real-time visualization function.

3.　プロトコル実装と検証

本章では、提案システムを用いて実際に無線通信プロトコル

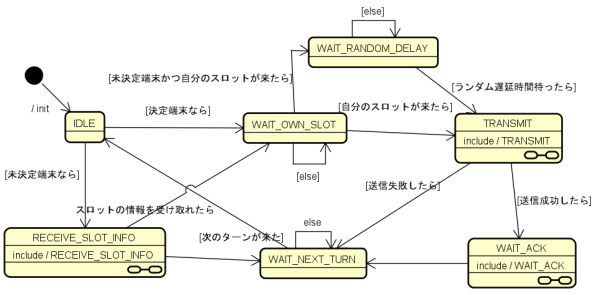

図 7　開発した CSMA/TDMA ベース無線通信プロトコルの状態遷移図（主要経路のみ）

Fig. 7　State transition diagram of the developed CSMA/TDMA-based wireless communication protocol (simplified main flow).

を開発・実装し、その有効性を検証した結果を示す。

3.1　プロトコルの実装

本節では，状態遷移図から自動生成されたファームウェアを用いて実装した無線通信プロトコルについて述べる。本プロトコルは，園田ら [**?**] による CSMA/TDMA ベースのプロトコルを参考として設計したものであり，TDMA によるスロット分割を基本構造としつつ，各端末が自律的に送信スロットを決定する点に特徴を持つ。

通信は，中継器（集約機）と複数の端末ノードとの間で行われる。通信ターン（フレーム）は複数のスロットで構成される。このプロトコルの特徴は最初のスロットにおいて中継器が全端末に対してスロットの占有状況を通知する点である。各端末はこの情報をもとに，自ノードの送信スロット候補を自律的に選択する。端末は、選択した候補スロットにおいててリクエストパケットを送信し，中継器から ACK 応答を受信した時点でスロットを確定・占有する。一方で，ACK が返ってこなかった場合や他ノードとの競合が発生した場合は，次のターンで再びスロット情報の受信を試み再試行を行う。このように，各端末が占有状況に基づいて分散的にスロットを決定することで，動的なスロット割り当てを実現している。

図 7 に本プロトコルの主要な状態遷移図を示す。本図は都合上、主要な通信状態のみを示している。アイドル状態 `IDLE` から開始し，スロット情報の受信状態（`RECEIVE_SLOT_INFO`），自分のスロットを待つ状態（`WAIT_OWN_SLOT`），ランダム遅延待ち状態（`WAIT_RANDOM_DELAY`），パケット送信状態（`TRANSMIT`），ACK の受信を待つ状態（`WAIT_ACK`），次のターンを待つ状態（`WAIT_NEXT_TURN`）を経て，再び `IDLE` に戻るサイクル構造を有する。

3.2　検証方法

提案システムの有効性を確認するため，状態遷移図から自動生成したファームウェアを用いて，CSMA/TDMA ベースの無線通信プロトコルを実機上で動作させた。

実験では，2.3 節で述べた無線通信デバイスおよび GPS モジュールを一つのノードとしてセンサネットワークを構築した。中継器ノードと複数の端末ノードにそれぞれファームウェアを書き込み，電波環境下で実際に通信を行った。

各ノードは通信ターンごとにスロット割り当て，データ送信，ACK 受信などを実行し，その過程で状態遷移および変数の変化をログとしてシリアルポートへ出力した。取得したログを可視化ツールに読み込み，状態遷移の再現および変数追跡を行うことで，通信動作を解析した。

この検証により，提案システムがプロトコル動作を正確に再現し，通信過程を逐次的に把握できるかを評価した。

3.3　検証結果

実験の結果，提案システムを用いることで，通信動作の状態遷移および変数変化を逐次的に追跡できることを確認した。具体的には，状態遷移図上で現在の状態がハイライトされ，対応する変数値やログ行が同期的に更新されることで，通信処理の流れを直感的に把握できた。

図 8 に，従来のコンソール出力によるデバッグと提案システムによる可視化結果の比較を示す。従来手法では，ログを逐一スクロールしながら状態変化を目視で追う必要があり，複雑な条件分岐を含む通信処理の流れを理解するには時間を要した。一方，提案システムでは，状態遷移図上で動作の推移が即座に可視化されるため，処理の因果関係を容易に把握できる。特に，異常動作発生時には，該当状態と変数の不整合箇所を瞬時に特定できる点が有効であった。

また，スライダ操作によるステップ実行機能により，通信過程を 1 状態単位で再現できた。これにより，高速に進行する通信動作を段階的に検証できることを確認した。さらに，複数の変数を並行して追跡することで，変数間の関係を可視的に把握でき，それらの値の変化の関連を容易に確認できた。

これらの結果から，提案システムは通信動作の把握と異常検出を容易にし，デバッグおよび開発両面で作業効率を向上させることが確認された。

4.　評価および考察

本研究で開発したログ可視化システムを用いて，提案した CSMA/TDMA ベースのプロトコルを実機上で動作させた結果，通信動作の状態遷移および変数の変化を逐次的に追跡できることを確認した。これにより，従来のコンソール出力に依存したデバッグ手法では困難であった通信挙動の把握が，直感的かつ体系的に行えるようになった。

特に，状態遷移図上で現在の状態をハイライトし，対応する変数値やログ行を同期表示する機能により，通信処理の因果関係を容易に把握できた。これにより，異常動作発生時の前後関係を視覚的に追跡し，従来に比べて原因特定までの時間を短縮できた。また，ステップ実行機能およびスライダ操作により，通信動作を 1 状態単位で前後に移動しながら確認でき，複雑な条件分岐や再送処理を含む動作を容易に分析できた。

変数追跡機能により，複数の変数を同時に監視しながら状態変化を確認できる点も有用である。たとえば，スロット番号（`current_slot`）と ACK フラグ（`ack`）を同時に可視化することで，ACK 未受信時にどのスロットで不整合が生じたかを即座に特定できた。これにより，通信ロジックの理解とデバッグ効率の双方が向上した。

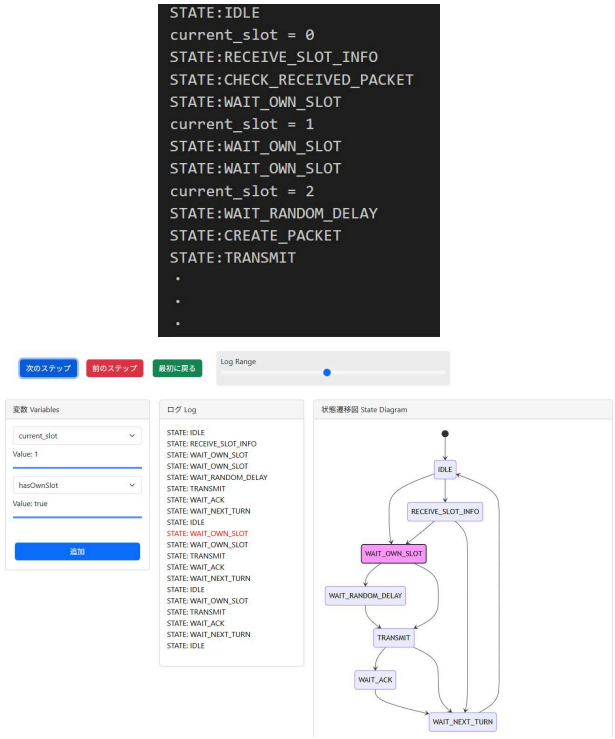

図 8　従来手法と提案システムによるデバッグ過程の比較

Fig. 8　Comparison between conventional console-based debugging and the proposed visualization system.

また，本システムは教育的利用にも適している。状態遷移図を教材として用いることで，通信プロトコルの構造と動作の関係を直感的に学習でき，設計意図の共有やアルゴリズム理解の促進にも寄与する。さらに，複数の開発者が同一ログを共有しながら解析できるため，チーム開発におけるデバッグ作業の効率化も期待できる。

一方で，現段階ではいくつかの課題も残る。第一に，リアルタイム性の向上が挙げられる。現状ではログを取得後に解析する「事後可視化型」であり，実機動作中の挙動を完全にリアルタイムで反映するには至っていない。第二に，ノード数の増加に伴うログ量の増大や描画負荷の増加に対するスケーラビリティの確保が課題である。第三に，通信効率やデバッグ時間の短縮効果など，定量的な評価を行うことで，提案システムの有効性をより客観的に示す必要がある。

以上より，提案システムは，通信プロトコルの設計・実装・解析を一貫して支援する基盤として有効であり，従来のデバッグ手法に比べて動作理解性と作業効率を大幅に向上させることが確認された。今後は，リアルタイム解析や多ノード対応を進めることで，センサネットワーク全体の挙動を俯瞰的に可視化できる環境へと発展することが期待される。

5.　まとめおよび今後の展望

本研究では，状態遷移図から自動生成したファームウェアを汎用ハードウェア上で実行し，無線通信プロトコルの動作を実機で検証可能とする環境を開発した。さらに，通信処理中に出力されるログを解析し，ブラウザ上で状態遷移および変数変化を可視化・ステップ実行できるデバッグ支援システムを実装した。

これにより，従来のコンソール出力では困難であった逐次的な動作確認を可能とし，通信動作の理解促進およびデバッグ効率の向上を実現した。また，状態遷移図ベースの設計と可視化環境の統合により，プロトコル設計から実機検証までを一貫して行える開発基盤を構築した。

今後の展望として，以下の 3 点を挙げる。

（1）　**複数ログの並行可視化：** 複数ノードのログを同時に読み込み，状態遷移を並列的に表示する機能を実装することで，センサネットワーク全体の挙動を俯瞰的に解析可能とする。

（2）　**センサネットワーク機能の拡充：** 通信制御のみならず，センサデータの取得・共有・集約を含むアプリケーション層の処理を統合し，より実運用に近いセンサネットワークを構築・検証できる環境へ発展させる。

（3）　**実利用を想定した応用実験：** 本環境を基盤として，登山者見守りシステムや環境モニタリング（水位・土砂など）などの社会実装型アプリケーションに適用し，提案手法の有効性と実用性を検証する。

これらの発展により，本システムは単なるデバッグ支援環境にとどまらず，実際のセンサネットワーク開発および運用評価を支える包括的な検証基盤として発展することが期待される。また，本研究の成果は，センサネットワーク分野における開発プロセスの効率化のみならず，通信技術教育や共同研究支援など，幅広い分野への応用可能性を有している。

文　　献

[1]　科学技術振興機構:「SIP IoE 年次報告書 2022」, 2022.
[2]　M. R. Hasan et al.: ”Recent Advancement of Data-Driven Models in Wireless Sensor Networks,” *Technologies*, vol.9, no.4, 2021.
[3]　旭健太、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-xx, 2022.
[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-xx, 2023.
[5]
[6]
[7]
[8]