

# 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<sup>†</sup>      小林侑生<sup>†</sup>      不破泰<sup>†</sup>      アサノデービッド<sup>†</sup>

Atsushi TSUJIMURA<sup>†</sup>, Yu KOBAYASHI<sup>†</sup>, Yasushi FUWA<sup>†</sup>, and David ASANO<sup>†</sup>

**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

## 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。そのため、システム開発の効率化や開発者支援を目的とした研究が進められている。

### 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、CSMAおよびTDMAベースのMAC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）。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、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

<sup>†</sup> 信州大学, 長野県  
Shinshu University, 4-17-1 Wakasato, Nagano-shi, Nagano 380-8553 Japan  
DOI:10.14923/transfunj.??????????

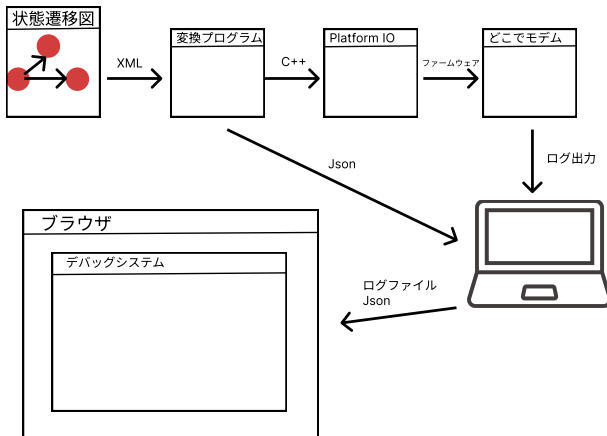


図1 提案システムの全体構成  
Fig. 1 Block diagram of the proposed system.

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protocols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、protocol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。

## 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protocolの動作をAstar 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、それぞれをリスト化したうえで、通信状態間の関係を解析し、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protocol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++コードを示す。状態遷移図で定義された「待機(LISTEN)」「受信(RECEIVE)」「送信(TRANSMIT)」

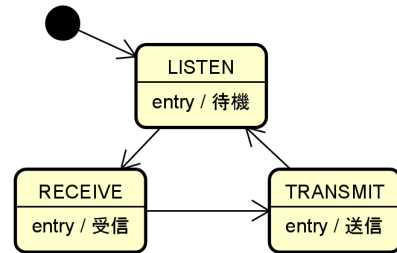


図2 状態遷移図の例  
Fig. 2 State transition diagram.

## 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 }
```

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。

## 2.3 実行環境(ハードウェア)

次に、実行環境について述べる。本システムでは、SAMD21マイコンを搭載した無線モデム(株式会社サーキットデザイン製どこでもモデム)を用い、生成したファームウェアを書き込み実行する。このモデムは429 MHz帯で動作する汎用的な無線モデムであり、技術基準適合証明を取得しているため、実際に電波を送受信しながら通信protocolの検証を行うことができる。また、429 MHz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、将来的には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、本モデム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IPPS信号による高精度な時刻同期を実現し、複



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.

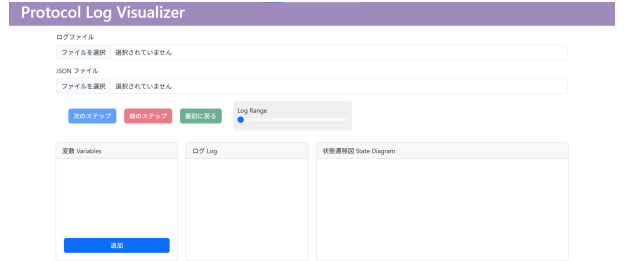


図4 ログ可視化・ステップ実行環境のUI

Fig. 4 User interface of the log visualization and step execution environment.

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、逐次の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。提案システムは、通信処理中に出力されるログ情報を活用し、ブラウザ上で状態遷移の再現・変数追跡を行うことで、開発者が通信の挙動を直感的に理解できる環境を提供する。

### 2.4.1 UI 構成

図4に本システムの可視化画面を示す。画面上部にはログファイルおよび状態遷移図ファイルのアップロード領域があり、中央にはステップ実行ボタンとスライダーが配置されている。下部には、変数追跡領域・ログ表示領域・状態遷移図描画領域が横並びに配置されており、通信処理の進行状況を統合的に把握できる構成となっている。

### 2.4.2 ログ出力

通信処理中の各状態遷移および変数の変化は、ファームウェア内で逐次的にログとして出力され、テキストファイルとして保存される。ログは識別タグを付与したシンプルな形式を採用しており、各行において状態または変数の情報を明確に区別できる。これにより、後述の可視化システムにおいて状態遷移と変数変化を容易に解析できる。ログファイルの一例を以下に示す。

### 2.4.3 可視化・再現機構

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ファイルをブラウザ上でアップロードすることで、Mermaid.jsを用いた状態遷移の可視化および再現を行う。Mermaid.jsはテキストベースで状態遷移図を描画できるJavaScriptライブラリであり、ログの進行に合わせて図中の状態ノードが順次ハイライトされる。これにより、通信処理の進行状況を視覚的に確認できる。状態遷移図のJSONファイルは、以下のような形式で各状態と遷移関係を定義している。

### 2.4.4 ステップ実行及び変数追跡

可視化画面には、ログ再現を操作するための「次のステップ」「前のステップ」「最初に戻る」ボタン、および任意位置に移動可能なスライダーが実装されている。これにより、通信動作の経過を1ステップずつ追跡しながら確認することができる。また、特定の変数を選択してその値の変化を随時表示する機能を備えており、状態遷移と変数変化の対応関係を同時に把握できる。これらの機能により、異常動作が発生した際の原因追跡を効率的に行える。

### 2.4.5 リアルタイム可視化

さらに、本システムの拡張として、MQTTおよびSocket通信を利用したリアルタイム可視化機能の試作を行った。これにより、実機から送信されるログデータを逐次受信し、ブラウザ上で即時に状態遷移を反映することが可能となった。通信プロトコルのリアルタイム動作を視覚的に確認できる点で有用であるが、通信周期が短い場合は状態遷移が高速に変化するため、視認性に課題が残る。

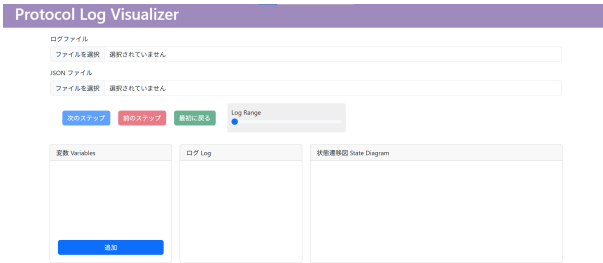


図5 ログ可視化・ステップ実行環境のUI

Fig. 5 User interface of the log visualization and step execution environment.

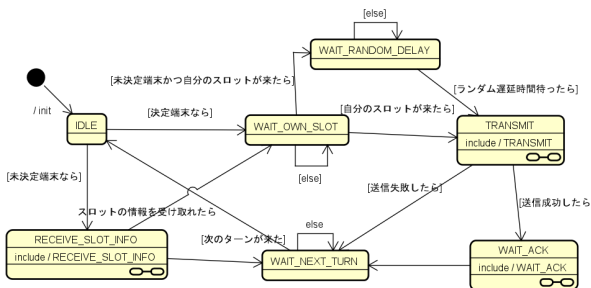


図6 開発した無線通信プロトコルの状態遷移図. ※実装上の詳細条件や内部処理は簡略化して記載している.

Fig. 6 State transition diagram of the developed wireless communication protocol.

### 3. プロトコル実装と評価

本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コードを用いて無線通信プロトコルを実装し、複数端末間での通信を行った。その際に出力されるログをブラウザに取り込み可視化することで、プロトコルの状態遷移を逐次的に追跡できることを確認した。

従来はコンソール出力のみに依存していたため、高速に進行する通信処理の挙動を逐一把握することは困難であった。本システムではログを視覚的に再現し、ステップ実行形式で確認できるため、デバッグ効率の向上に寄与することを示した。

### 4. 考 察

本研究で開発したログ可視化機能により、通信処理中の状態遷移を逐次的に追跡できることを確認した。これにより、従来はコンソール出力のみに依存していたため把握が困難であったプロトコルの挙動を直感的に理解でき、デバッグ効率の向上に有効であると考え

```
STATE:IDLE
current_slot = 0
STATE:RECEIVE_SLOT_INFO
STATE:CHECK_RECEIVED_PACKET
STATE:WAIT_OWN_SLOT
current_slot = 1
STATE:WAIT_OWN_SLOT
STATE:WAIT_OWN_SLOT
current_slot = 2
STATE:WAIT_RANDOM_DELAY
STATE:CREATE_PACKET
STATE:TRANSMIT
.
.
.
```

図7 従来のコンソールによるデバッグ方法

Fig. 7 Example of the old console-based debugging method.

られる。

特に、従来のコンソールログによる確認方法では、高速に進行する通信処理の流れを逐一追跡することが難しく、異常動作の原因把握に時間を要していた。これに対し、本研究の可視化環境では状態遷移を図として再現できるため、通信処理の理解や他者への説明が容易になり、教育的利用や共同開発の場においても有効性が高いと考えられる。(従来のコンソールでのデバッグと、比較した画像添付)

一方で、本研究では通信性能の定量的な評価や大規模ネットワークでの検証は行っておらず、開発環境の有効性を示すには今後さらなる検証が必要である。また、ログ可視化のリアルタイム性についても限定的であり、実時間での挙動確認に向けた拡張が今後の課

### 5. まとめ、今後の展望

本研究では、状態遷移図から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検証可能とするシステムを開発した。さらに、通信処理の挙動をログとして記録し、ブラウザ上で可視化・ステップ実行できる環境を実装することで、プロトコルの状態遷移を逐次的に追跡できることを確認した。その結果、従来のコンソール出力のみの手法と比べてデバッグ効率の向上に有効であることを示した。

今後の課題としては、定量的な評価による有効性の測定、複数端末を用いた大規模環境での検証、およびリアルタイムでのログ可視化機能の強化が挙げられる。

また、登山者見守りシステムへの応用や教育現場での利用など、社会的実装を意識した展開も検討していく予定である。(ほかのセンサ導入とかも)

## 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]
- [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]

## 付 録

1.

**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