

データ解析による生産設備の保全に関する研究（第1報）

生産技術部

中野太郎 大坪昭文

製造業において、生産設備の摩耗や疲労・劣化による不具合は避けられない現象である。従来、熟練技術者が装置の稼働音を聞く、あるいは掌で振動を感じるといった官能評価によって機器類の不調を捉え維持管理を行っていたが、信頼性・再現性に課題があった。また、少子高齢化に伴う人手不足や技能伝承問題が深刻化し、属人的な保全活動の維持が困難になってきており、IoT や AI などを活用した設備保全の省力化や自動化が急務となっている。本研究では、加速度センサやマイク・電流計等の IoT センサを用いて収集した生産設備の稼働データに基づき、異常状態の兆候を検知し、装置の故障・劣化予測を行う解析技術の研究・開発に取り組む。本年度は、設備状態の分析における情報処理の高度化を図るため、従来の環境データに加え、振動や音響などの時系列データの収集に対応した汎用的な計測用エッジデバイスの試作開発を行い、生産現場での各種センサデータの収集および送信システムを効率的に構築できることを確認した。

1. はじめに

製造業において、主要な装置に故障が発生すると、修理のために生産ラインを長時間停止せざるを得ず、納期の遅延や生産計画の変更等、企業の生産活動に多大な影響を及ぼす。一方、生産設備の摩耗、疲労および劣化による不具合は避けられない現象であるため、従来は、熟練技術者による装置の稼働音の聴診や振動の触診等の官能評価によって、機器類の状態監視および維持管理が実施されてきた。しかしながら、これらの手法には信頼性および再現性の確保に課題がある。また、少子高齢化に伴う人手不足や技能伝承問題が深刻化し、属人的な保全活動の維持が困難になってきており、IoT (Internet of Things) や AI (Artificial Intelligence) などを活用した自動化や設備保全が急務とされている。

本研究では、加速度センサ、マイクロフォンおよび電流センサ等の汎用 IoT センサにより収集された生産設備の稼働データに基づき、異常状態の兆候を検知し、装置の故障および劣化を予測する解析技術の開発を行う。本年度は、解析対象となる時系列データを収集するためのエッジデバイスの設計および実装に取り組んだ。

2. データ収集システムの構築

2.1 時系列データ収集システム

本システムでは、監視対象となる設備機器にセンサ類（加速度センサやマイクロフォンなど）を設置

し、エッジデバイスを介してデータの収集および転送処理を実行する自動計測機能を実装する。その際、一定周期でサンプリングされた加速度や音響などの時系列データは、温度、湿度、圧力といった静的な環境データと比較して、データ量が顕著に大きくなる傾向がある。このような大容量データを無線通信で収集する場合、高速通信が可能で、多様なデータ形式を取り扱うことができる Wi-Fi を用いたデータ転送が有効である¹⁾。

2.2 計測データの形式と通信プロトコル

センサなどで計測したデータを Web 経由で保存するデータ蓄積システムの概要を図1に示す。工場内の設備機器に設置されたエッジデバイスから、Wi-Fi 通信を介して効率的にデータを収集するために、Linux 系サーバ上に、HTTP プロトコルを基盤とした Web API (Application Programming Interface) を設計・実装した。Web API とは、サーバ上の特定のエンド

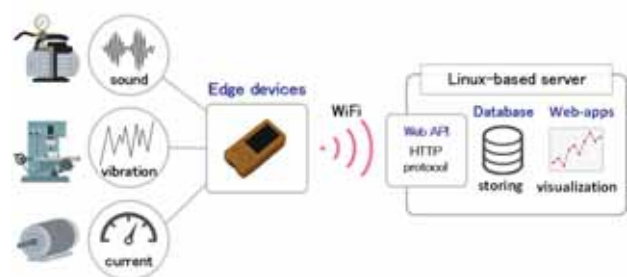


図1 データ収集システムの概略図

ポイントに対して、データの取得 (GET) や作成・更新 (POST) などのリクエストメソッドを定義したものであり、JavaScript Object Notation (JSON) や Extensible Markup Language (XML) 形式などを用いて、データを体系的に送受信するインターフェース仕様である。

センサで収集したデータの処理を Web API を介してサーバに委譲する際、時系列データ配列、各種識別子、および関連するパラメータをオブジェクト内に構造化して格納し、HTTP POST 通信によって送信する。サーバ側では、受信した構造化データを、データベースへの格納やファイル保存、解析処理など、各種の処理要件に応じて活用することが可能である。データの構造化形式としては JSON を採用した。JSON はシンプルな構造を有し、多くのプログラミング言語でサポートされているため、異なるシステム間でのデータ交換が容易である。さらに、JSON はスキーマ (データ構造) が固定されていないため、将来的なメタデータや付加的な情報の拡張性を確保し、新規要素を容易に追加可能な柔軟な設計を実現できる。

3. 計測用エッジデバイスの開発

3.1 計測デバイスの仕様

IoT デバイスに適用される主な要件は 6A (Anything, Anytime, Anyone, Anyplace, Any service, Any network) と呼ばれており、ワイヤレス環境における信頼性の高い遠隔通信とデータ転送を保証する効果的な接続性を提供することにある^{2,3)}。本検討では、設備状態監視における情報処理の高度化を目的として、振動、音響等の生波形データを収集するデバイスの開発に取り組んだ。計測用エッジデバイスの開発における主要な仕様は以下の通りである。

- ・計測信号の状態を現場で視覚的に確認できること
- ・複数センサから時系列データを収集し、API サーバに送信可能なこと
- ・安価で入手が容易な機器構成とし、拡張性と汎用性を確保すること

これらの機能を効率的に実装するために、エッジデバイスのベースとなる小型のマイコンモジュールとして、M5Stack 社製「M5StickC Plus」を採用した⁴⁾。M5StickC Plus は、安価でありながら高性能かつ低消費電力のマイコンチップ「ESP32」(Espressif Systems 社製) を搭載し、無線通信機能 (WiFi および Bluetooth) を備えている。また、Grove インターフェース (汎用コネクタ) を搭載するなど拡張性も

高く、スマートホームデバイスや環境モニタリングシステムなど、さまざまな IoT アプリケーションの開発に広く利用されている⁵⁾。

3.2 振動センサおよび音響センサ

振動の計測センサには、M5Stick に内蔵された慣性計測装置 (IMU ; Inertial Measurement Unit) を利用した。IMU から取得した加速度値をマイコン制御により定期的にサンプリングし、振動に関わる指標値とする。

音響データの収集に関しては、小型のエレクトレットコンデンサマイク (ECM) による計測を試みたが、各マイクの感度に大きな個体差が存在し、ばらつきが問題となった。感度調整により一定程度の誤差には対応可能であったものの、設置時の手間や設置後の感度変動の可能性、さらに感度差が大きすぎて調整では対処できないなどの課題が残った。そこで、より安定した音響収録を実現するために、個体差の影響が少ないデジタルマイクロフォン (SPM1423) の適用を検討した。

SPM1423 は音響信号を PDM (Pulse Density Modulation) 方式でデジタル化する MEMS マイクロフォンであり、I2S (Inter-IC Sound) 規格のシリアル通信でデジタル信号を伝送する。I2S による通信には、M5StickC Plus (ESP32) に内蔵された PDM モジュール⁶⁾ を利用した。このモジュールを用いて伝送した PDM 信号を PCM (Pulse Code Modulation) 信号に変換処理することで、最大約 50 kHz までの音響信号が受信可能である。SPM1423 を用いて実験的に計測した音響データ (正弦波) を図 2 に示す。音響計測時と無音時の SN 比を用いて ECM との性能を比較した結果、ECM が 10.4db であったのに対し、SPM1423 は 34.8db と、品質面においても安定していることが確認されたため、本検討における音響計測用センサとして採用した。最終的に選定した計測デバイスの機器構成を表 1 に示す。

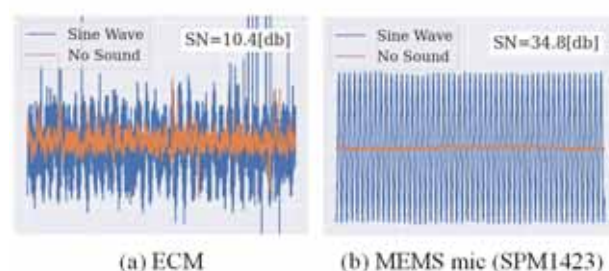


図 2 SN 比によるマイク性能の比較

表1 計測エッジデバイスの構成機器

Type	Part number
Device controller	ESP32
M5StickC Plus	Espressif Systems
Accelerometer	MPU6886
Inertial Measurement Unit	InvenSense
Microphone	SPM1423
Pulse Density Modulation	SiSonic

表2 マルチコアへのタスクの割り当て

Core 0 : 時系列データ収集タスク
<ul style="list-style-type: none"> ・定期的にセンサからデータを収集 ・データをバッファに保存 ・必要に応じたデータの前処理(センタリング等)
Core 1 : データ送信および描画タスク
<ul style="list-style-type: none"> ・バッファに蓄積されたデータを定期的にPOST ・送信エラーの検出と再送処理 ・簡易的なデータの統計処理(FFT等) ・液晶画面表示内容の更新

4. 計測制御プログラミング

4.1 開発環境

ESP32 のプログラム開発言語には、C 言語、MicroPython, Lua, JavaScript など用途に応じた多様な選択が可能であるが、本検討ではソフトウェア開発の拡張ライブラリが充実している Arduino プラットフォームを利用する。Arduino は C++を基盤とした言語構文が採用されており、初心者でも比較的容易に習得可能である。また、オープンソースコミュニティやサードパーティによって開発された多数のライブラリが存在し、デジタル I/O, シリアル通信などの基本機能に加え、センサ操作やネットワーク通信など、特定の用途に特化したライブラリも多岐にわたって公開されている。

プログラミングの統合開発環境 (IDE) としては、Microsoft 社の Visual Studio Code (VS Code) を選択した。VS Code はコード補完、デバッグ機能、ソースコード管理など、豊富な拡張機能を有し、開発の効率化に繋がる高度な開発支援環境が提供されている (図3)。

4.2 並列処理の実装

ESP32 はリアルタイムオペレーティングシステム (RTOS) である FreeRTOS⁷⁾ を搭載している。RTOS は、時間的制約のある処理を正確に実行することを

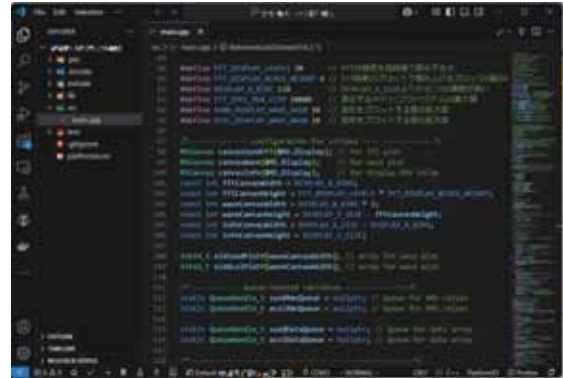


図3 制御プログラムの統合開発環境

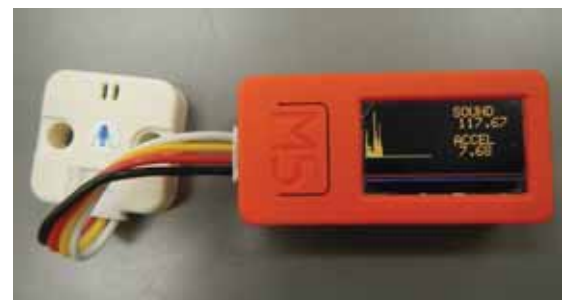


図4 開発した計測エッジデバイス

目的として設計された制御システムであり、組み込みシステムなどのリアルタイム性が要求されるアプリケーションに利用されている。ESP32 は、タイマーや通信機能など各種周辺機能を内蔵するデュアルコアの高機能集積回路であるが、FreeRTOS を介してこれらのハードウェアリソースを管理することにより、複数のプログラムの並列処理や割り込み処理の実装が容易になる。

本検討で開発する計測用エッジデバイスにおいては、計測精度の確保が最も重要である。データ収集時のリアルタイム性と要求仕様を両立するために、表2に示すように各コアに適切な処理を割り当て、ESP32 に搭載された二つの CPU を有効活用するマルチプロセス並列処理プログラムを実装した。開発した計測エッジデバイスを図4に示す。

シングルコアのチップを用いて疑似的な並列処理によってタスクを分担した場合、各タスクのトリガーが優先度によって制御されるため、予期せぬブロッキングによりデータサンプリングが妨げられる可能性がある。一方、ESP32 のようなマルチコアシステムでは、各 CPU コアにタスクを割り当てることにより、プロセスごとに高いリアルタイム性を確保することができる。

5. おわりに

生産設備の状態監視における情報処理の高度化を目的とし、振動や音響などの波形データを収集し、一括送信する計測デバイスの開発に取り組んだ。従来、振動センサ等を用いた時系列データの計測システムは高コストであることから、まず小型のIoTマイコンと低価格なセンサを組み合わせた構成により、拡張性および汎用性の確保を試みた。また、小型ディスプレイに時系列データの波形やFFT解析結果等を簡易的に表示する機能を実装し、現場における設置時および設置後のモニタリングの利便性を向上させた。さらに、複数の要求仕様に対応し、かつ時系列データの計測精度を確保するために、マルチプロセスの並列処理によるリアルタイム性の向上に取り組んだ。その結果、汎用マイクロコントローラ（ESP32）の機能特性を並列処理によって引き出し、デバイス開発に応用することで、信頼性に優れたデータ収集および送信システムを効率的に構築できることを確認した。

現在、開発したデバイスを用いて生産現場でのデータ収集を実施しており、今後は収集された生産設備の稼働データに基づいて異常状態の兆候を検知し、装置の故障および劣化を予測するための解析技術の開発に着手する予定である。

参考文献

- 1) 河野光一郎, et al. 音響データ収集による機器異常検知のためのデータ分析及び可視化手法, 電子情報通信学会技術研究報告, 2024, 123.422: IA2023-88.
- 2) Li, S., et al. The internet of things: a survey, Information systems frontiers, 2015, 17, pp.243-259.
- 3) Maier, A., et al. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, Internet Technologies and Applications, 2017, pp. 143-148.
- 4) StickC-Plus, https://docs.m5stack.com/en/core/m5stickc_plus
- 5) Foltýnek, P., et al. Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board. Measurement and Control, 52(7-8), 2019, pp. 970-984.
- 6) ESP32-I2S-Controller(I2S), https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf#i2s
- 7) FreeRTOS, <https://www.freertos.org/>