MQTT broker 間のデータ受け渡しによる IoT システムの実現

IMPLEMENT OF IOT SYSTEM FOR DELIVERY DATA BETWEEN TWO TYPES MQTT BROKER

石塚規友紀#, A), 福井達 B), 清道明男 C)
Miyuki Ishizuka #, A), Toru Fukui B), Akio Kiyomichi C)
A) Hitachi Zosen Corporation
B) RIKEN
C) JASRI

Abstract

Use of IoT technology is becoming general as the way to resolve various problems of the industrial world - for example, manpower shortage and technological succession etc. It is useful for equipment maintenance and a bad forecast to analyze the data collected from sensors and operation information on equipment in real time. One of the communication standards used in that case is MQTT. We made the data acquisition system which acquires temperature-humidity data periodically and made them move on armadillo with the sensors. We made them as edge devices and installed those in an electron accelerator. Further, we placed MQTT broker on VME for data collections. We acquired data from the MQTT broker, and made the process stocked in a data base. The purpose of this composition is to improve management of a network which consists of a lot of sensors. It's because they became able even to assume from sensors one of equipment including MQTT broker and deal with data. In SPring-8, the control system have been changed to new one from last year. The new system could combine the process with new framework and achieve collection of temperature-humidity data. The practical use is continued without trouble up to now. As well as the explanation about the system we made. This article introduces the implement for which a delivery of data by shared memory pass was used. We are considering separating data delivery processing from data acquisition processing as future's development.

1. はじめに

インダストリー4.0 にて掲げられているスマートファクトリは、ビッグデータを利用した成長戦略の一つである。人口減少に伴う労働力確保の一案として IoT を利用することは、既に様々な業界において検討され、実用化されつつある。例えば工場運営や作業の機械化を目的としたシステムの構築が考えられるだろう。IoT の利用は、IoT 対応機器による測定データの収集はもちろん、機器の死活監視にも対応可能である。

加速器施設では各種様々な機器が使用されており、日々多くのデータを扱い蓄積していく点で、産業界に限らず、IoT の利用が有益と考えられる。たとえば温度測定のような環境データや運転前のみ一時的に使用する機器の監視のように、堅実な構成よりも特定の機能を簡便に達成できることを優先したい場合もある。こうした機器の場合、必要最低限の仕様さえ満たすことができれば、より簡易的なセットアップを使用できることがメリットとなるだろう。

本稿では、SPring-8 における IoT 化の例として採用した、温湿度データの収集に関する仕様と今後の展開について報告する。

なお IoT に利用可能なプロトコル、システム構成は複数存在するが、今回はその中でも MQTT プロトコルを使用した例について述べることとする。

miyuki_ishizuka@hitachizosen.co.jp

2. SPring-8 のデータ収集概要

SPring-8 の制御系フレームワーク概要図を Fig. 1 に示す。SPring-8 の制御系は所内で独自に製作したフレームワーク構成を持つ。

まず上位系としてユーザが機器を操作したり、機器の状態・測定値を表示したりする GUI から、中継サーバ(Message Server; MS)へ命令文を送る。中継サーバは各機器上で動作しているプロセス(EM ; Equipment Manager)に命令文を送り、EM は命令分を機器に届ける。機器が取得した測定値は逆の順番をたどり、GUI へと渡される。

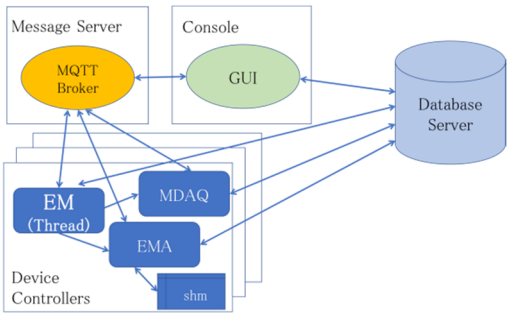

Figure 1: Data acquisition layout.

本構成の特徴はメッセージング方式により機器の操作・機器からのデータ取得を実現する点である。命令文を伝達していくことで必要なデータを得る。

本フレームワークは、加速器運転に必要な電磁石・真

空・モニター系から、環境測定用の機器のデータまで幅広く取りまとめ、管理している。

さらに今年度からは中継サーバとして MQTT ブローカーを使用している。[1]そこで、次に MQTT プロトコルについて説明する。

3. MQTT プロトコル

MQTT(Message Queueing Telemetry Transport)とは、TCP/IP ネットワークで利用できる通信プロトコルの一つである。

MQTT はブローカーと呼ばれる中継サーバと、データを送受信するクライアントから構成される。ブローカーへデータを送信するクライアントを publisher, ブローカーからデータを受信するクライアントを subscriber と呼び、1 対 1、1 対多、多対多の通信が可能である。(Fig. 2)

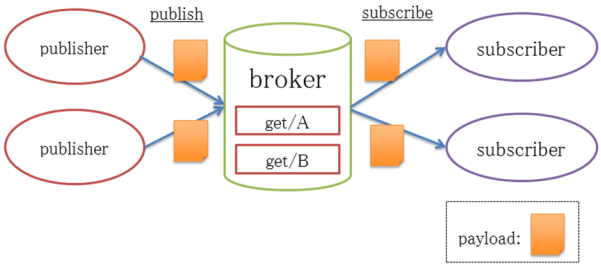

Figure 2: Schematic diagram of MQTT.

MQTT の特徴の一つは、publisher と subscriber の動作が非同期である点である。publisher/subscriber はそれぞれ独立に、互いの状態に依らずブローカーへのデータ送信およびブローカーからのデータ受信を行う。ブローカーは受信したデータを保持しているため、publisher からのデータ送信から subscriber によるデータ受信の間に時間差があっても良い。両者はそれぞれのタイミングでデータを送受信することができる。

その他の特徴として、送受信データとは別に topic と呼ばれるタグを使用する点が挙げられる。送受信データは payload として受け渡しされるが、1 つの payload に対して topic を 1 つ付加する。

Topic は"/"区切りの階層構造を持つ。例を Fig. 3 に示す。subscriber はデータ受信時に取得データの topic を指定する。その際、ワイルドカードも使用可能である。

"/"区切りの階層構造を持つこの書式は、SPring-8 におけるメッセージングと書式が似ている。そのため、SPring-8 制御系更新に伴い、主要な仕様を残しつつ更新を行うことが可能な MQTT を採用した。

```
a.get/
b.get/srmonthermo-A-01
c.get/srmonthermo-A-01/temperature
```

Figure 3: Hierarchical structure of MQTT topic.

4. 温湿度データ収集のシステム構成

4.1 温湿度データ

温湿度データとしては収納部内の一か所につき温度(temperature)・湿度(humidity)・露点温度(dew point)の測定値を、蓄積リングに沿って計 27 か所、それぞれ 10 秒周期で収集している。これは SPring-8 蓄積リングの BPM バランス変動が湿度に相関して変動することが分かり、そのためリング各所に配備してデータ収集することを目的としているためである。つまり BPM のケーブルが経年劣化による放射線損傷等を受けた場合に、収納部内の湿度の変化によってケーブルが水分を吸収してしまう。その結果誘電率が変化し、ケーブルの周波数特性が変化して BPM の 4 電極におけるバランスが崩れてしまうことが確認されている。

測定用の機器には Armadillo を使用した。Power-over-Ether(PoE)対応拡張基板に温湿度センサーを専用ハーネスで接続した Armadillo-410 搭載ユニットを製作し、現場に取り付けている。Armadillo の詳細な仕様は Table 1 にて後述する。

ソフトウェアの点では、Armadillo 上でデータ取得のためのプログラムが動作し、データの取得を行っている。ただしその他の機器上では EM が起動している一方、本機器ではより単純にデータ収集のみを行うプログラムが動作している点が異なる。各 Armadillo は制御用のネットワークに接続されている。

4.2 ハードウェア構成

Armadillo-410(atmark techno 社製[2])の詳細な仕様を Table 1 に示す。

Table 1: Specification of Armdillo-410

項目	詳細
型番	A4110-U00Z
プロセッサ	NXP セミコンダクターズ (旧 Freescale)製 i.MX257
CPU コア	ARM926EJ-S(コアクロック:400MHz, バスクロック:133MHz)
RAM	128MB(LPDDR SDRAM)
ROM	32MB(NOR 型)
電源電圧	DC 3.1~5.25V
消費電力	1.2W(Typ.),待機時 0.9W(Typ.)
動作温度範囲	-10~50℃
外形サイズ	50×40mm
標準 OS	Linux (カーネル 2.6/3.14)

Armadillo は ARM コアのアプリケーションプロセッサを搭載した IoT 向けの組み込み CPU ボードである。実機として使用しているのは、I2C2 インタフェースにハーネスにて温湿度センサーSHT21/SHT25(SENSIRION 社製, Table 2 参照)を接続したユニットである。ネットワークケーブルのみの接続で受電できるため、電源を確保しにくい

加速器トンネル内への配備が容易に可能である。IoT 化にあたり、ハードウェアは以前から使用しているセンサー付きの本 Armadillo ユニットをそのまま再利用した。

その他変更点として、MQTT ブローカーを稼働するための VME を 1 台、新規に用意した。この VME は 27 台の Armadillo から送信されるデータを取りまとめる役割として利用する。OS として Ubuntu を使用しており、MQTT ブローカーとしては mosquitto[3]を採用した。

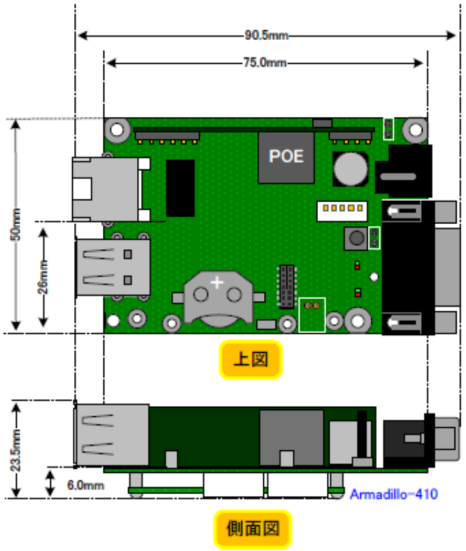

Figure 4: Aspect of Armadillo-410 unit.

Table 2: Specification of SHT21/SHT25

項目	詳細
サイズ	3×3×1.1mm^3
出力	デジタルインタフェース(I2C), PWM,SDM
電源電圧	2.1~3.6V
消費電力	3.2μW(8 ビット、毎秒 1 測定時)
動作湿度範囲	0~100%
動作温度範囲	-40~125℃
湿度応答時間	8s(τ 63%)

5. ブローカー間のプロセス

5.1 ソフトウェア構成とその目的

次にソフトウェア構成について、概要図を Fig. 5 に示す。今回の IoT 化においては、EM を一つにまとめる改修を加えた。これは、27 台の Armadillo から収集するデータを一括で扱えるようにするためである。全体のシステムでは共通用のブローカー(eMQTT[4])があるため、EM はそちらへデータを送信する。

また、その他の変更点としてデータ収集を行うまでに 2 つのブローカーを経由するように改修した。1 台は VME 上に置き、もう 1 台はその他の機器も含めて共通に使用するために用意されているものである。

まず Armadillo 起動時にプログラムが自動的に起動し、データ取得を始める。そして周期的に温度・湿度を測定し、値を取得する。この値はデータ取得が行われるたびに信号ごとの payload に ASCII で書き込まれ、VME 上の MQTT ブローカーへ送信する。この処理が Armadillo 計 27 台分繰り返し行われる。

続いて、VME 上のデータを全体用の MQTT ブローカーへ送信する。これは VME 上で起動するプロセス(Fig. 5 における EM)が行う。

EM はその他の機器と同様、既定の手順によって製作するため、SPring-8 制御系全体の信号命名規則に従う topic を保持している。この topic に基づく信号名のデータを送信することになる。送信されるデータは他の機器からのデータと同様に収集され、専用の管理 WEB ページから随時、値を確認することができる。

5.2 ブローカー間のデータ送受信

2 つのブローカーを使用した構成によるデータ収集においては、使用するライブラリが懸念事項であった。両ブローカーに対する topic は異なっており、さらに同期・非同期のアクセスを行う予定であったため、当初両方向へのデータ送受信に同一のライブラリ(paho[5]ライブラリ)を使用できるのではないかと考えた。しかし試験の結果、mosquitto への同期アクセスと eMQTT への非同期アクセスの混在が困難であると判明した。そのため 2 種類の別ライブラリを使用することにし、EM から eMQTT への非同期アクセス用ライブラリとして mosquitto API を使用した。一般的な MQTT プロトコルによる通信と同様に、クライアントの生成・コネクションの接続・topic の登録・データの受信と送信という処理を EM 内で両ブローカーに対して行うよう、製作した。

ブローカーのアドレスや収集周期、信号名は設定ファイルから設定する。このファイルは SPring-8 制御系共通のフォーマットで書かれたファイルである。EM 製作時には必ず用意するファイルであり、信号名は全体用ブローカーへの送信の際の topic としても使用する。

さらにこのファイルとは別に、topic の一覧を記載したファイルも用意した。これは Armadillo から VME 上のブローカーへ送信しているデータの topic 一覧である。前述の信号名とも対応しているものの、別の topic 名を付与している。なお Armadillo から VME への送信で使用する topic 名は、全体用ブローカーへの送信の際の topic と対応するように命名したため、信号名を把握できるようになっている。

EM は、topic 一覧ファイルに記載されている topic を持つ信号をブローカーから受信する。callback 関数を使用して次々と値を受信するようにした。このとき、データは payload に格納されているため、payload からデータ部分を取出して取得する。取得したデータは全体用ブローカーへの送信のため改めて topic をつけ、送信する。

ただしその際は、データは payload ではなく、topic の一部として全体用ブローカーに送信する。これは SPring-8 制御系のフレームワーク規則に従う書式であり、今回に依らずその他の機器についても同様に、データや応答が topic の末尾部分に置かれる。

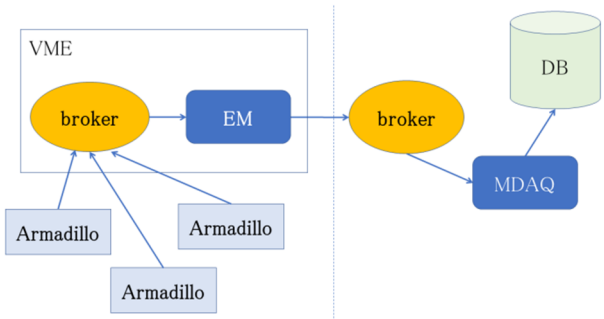

Figure 5: Software layout.

5.3 今後の改良点

一方で現在の実装では、EM は Armadillo の順番を特に気にしていない。また信号毎に取得データの保持をしていないため、データ取得の度に EM 内では毎回同じ処理が必要であり、一度に複数のデータを連続的に取得することはできない。そのため、現状より早い周期でのデータ取得が必要な場合に対応できない可能性もある。

そこで、今後の展開として次の改修を検討している。概略図を Fig. 6 に示す。

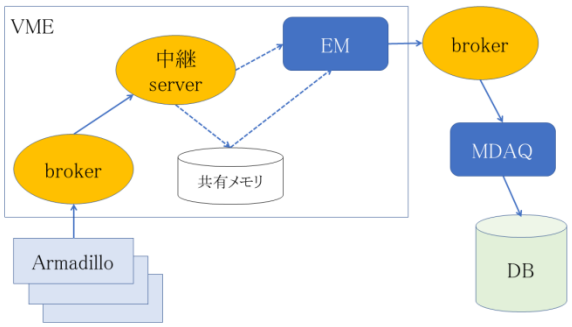

Figure 6: Next plan of software layout.

まずはデータ保持用の仕組みを取り入れることである。たとえば VME 上の MQTT ブローカーと EM の間に共有メモリを用意する。さらに MQTT ブローカーと EM の間にサーバを置き、データ取得の際には MQTT ブローカーからサーバへ、サーバから共有メモリへデータを書き込むこととし、EM はデータ取得のたびに共有メモリからデータを取り出す。

サーバを EM-MQTT ブローカー間に置くため経路は複雑になるものの、共有メモリを使用することにより複数データを保持でき、早い収集周期でのデータ収集にも対応可能になると期待できる。

また、VME 上の MQTT ブローカーと EM 間が直接通信しないため、EM は共有の MQTT ブローカーに対する送信のみを行うことになる。よって共有の MQTT ブローカーとの通信で使用しているものと同一の API(paho ライブラリ)を使用できる可能性がある。同一の API であれば既存のライブラリを使用することになり、製作の手間を減らせる上、製作にあたっての誤りも防ぐことができる。

共有メモリ以外のその他の方法としては、サーバと EM 間をメッセージングでつなぐ方法も考えられる。データの保持は中継サーバで行い、メッセージングで EM とやりとりすることにより、共有メモリを使用した場合と同様に、既存のライブラリを使用できる。その際のメッセージング方式としてはいくつか案があり、パイプや ZeroMQ を検討している。

あるいは、Armadillo からデータを受け取る中継サーバをクラウドに置き換えることも考えられる。この場合は今回のソフトウェア構成によってデータ収集を行うことに加えて、Armadillo からのデータを外部から確認することが可能になる。

6. まとめ

SPring-8 において収集している様々なデータのうち、温湿度データを対象に IoT 化を目指した実装の試作を行った。

SPring-8 では全体の制御系移行において、MQTT プロトコルを採用している。このプロトコルを使用した設計とすることにより、SPring-8 にて以前から採用してきた制御フレームワークを継承しつつ、データの重要度に合わせたデータ収集系を準備できた。送信と受信の非同期性が MQTT の特徴の一つである。これにより、双方がそれぞれのタイミングでデータのやり取りをすることができる。またデータに対応した topic を持つことで、以前のフレームワークとその思想を変えることなく使用可能である点も有用である。

試作を通して IoT 化を進める場合に必要な手順と考慮すべき注意点を把握することができた。今回は 2 種類のブローカーを使用したシステム構成を製作し、本環境においてはライブラリについての工夫が必要と分かった。対策として 2 種類のライブラリを利用している。

今後の展開として、さらに改修を加えることで、より効率的かつ製作時にわかりやすい構成にすることも可能と考えられる。今回問題となった 2 種類の API を使用する必要性もなくなり、一つの API を使用して全体を制作することもできる可能性がある。これにより新しい構成も可能になるかもしれない。

恒久的な稼働のみでなく、一時的な稼働で十分な機器のデータ収集、死活監視にも利用することができるだろう。

参考文献

[1] T. Fukui *et al.*, “Status of the Control System for the SACLA/SPring-8 Accelerator Complex”, In this Proceedings of ICALEPCS2017, Barcelona, Spain, 2017, FRAPL03.
[2] https://armadillo.atmark-techno.com/
[3] https://mosquitto.org/
[4] http://emqtt.io/
[5] https://www.eclipse.org/paho/