Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

PicoGW: 商用利用に適したオープンソースの汎用デバイスゲートウェイ

大和田　茂[1,a)]　杉村 博[2,b)]

概要：IoT サービスのエッジ側ハブとして使われることを想定した,Node.js ベースのオープンソースデバイスゲートウェイ「PicoGW（ピコゲートウェイ）」の開発と商用利用について報告する. 本ゲートウェイは商用利用の促進のために, インストールや機能拡張・フロントエンドの改変が容易な作りになっており, 長期安定稼働にも適した仕組みを持たせた. 様々な既存クラウドサービスとの連携を想定し, 複数の API エンドポイントも実装している. また, ゲートウェイは多企業の接点となるためにベンダーロックインが生じやすいが, 本ゲートウェイは神奈川工科大学よりオープンソースで公開されているために, そういった懸念が少ない.2019 年には本ゲートウェイを既存のクラウドサービスと連携させた, 工場向け EMS(FEMS) サービスが商品化された.

キーワード：BEMS, FEMS, HEMS, BAS, FA, HA, ゲートウェイ, IoT, デバイス, オープンソース, ECHONET Lite, modbus, API

PicoGW: An Open Source, General Purpose Device Gateway Suitable for Commercial Use

Shigeru Owada[1,a)]　Hiroshi Sugimura[2,b)]

Received: January 1, 2021, Accepted: xx xx, xxxx

Abstract: We report on the development and commercial use of PicoGW (Pico Gateway), a Node.js-based open-source device gateway designed to be used as an edge-side hub for IoT services. In order to promote commercial use of the gateway, it is designed to be easy to install, extend functions, and modify the front end, and has a mechanism suitable for long-term stable operation. It is also implemented with multiple API endpoints for integration with various existing cloud services. In 2019, we are planning to develop a factory EMS (FEMS) services have been commercialized.

Keywords: FEMS, FA, Gateway, IoT, Devices, Open source, ECHONET Lite, modbus, API

1. 背景

IoT がバズワードから一般的な用語へと変化して久しく, 家庭向け, 事業向け問わず, 身近で当たり前の技術として認知されるようになってきている. 他方, 当初より懸念されていた, 様々な事業分野を対象とした様々なメーカーの機器やサービスが相互依存的に参加するシステムの動作保証や, プロトコルの乱立・排他性の問題は依然として残っていて, 広範な普及を妨げている面がある.

この多メーカーの機器が同時に接続される, エッジ側のハブはデバイスゲートウェイ, または IoT ゲートウェイなどと呼ばれる（以下 GW）. 本稿で紹介する PicoGW はこの GW の一種である. 使いやすく, 安定で, システムの目的に合わせたカスタマイズも容易な GW の存在は,BEMS・FEMS・HEMS・BAS・FA・HA の別なく IoT システムの運用コストを下げ, クラウドとエッジの整合性を高める上

[1] 株式会社ソニーコンピュータサイエンス研究所
Sony CSL, Shinagawa-ku, Tokyo 141–0022, Japan
[2] 神奈川工科大学
Kanagawa Institute of Technology
[a)] owd@hoikutech.com
[b)] sugimura@he.kanagawa-it.ac.jp

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

で大変重要である. ここには, 多業種・多企業が参加することで出現する問題がある. 一つには, すべての業者がすべての技術に精通しているわけではないということである. その企業間のドメインの違いは, 製品としてのサポート範囲の切り分けを難しくする. しかし, それらのサービスは相互的に依存しており, ユーザーからすれば一つのシステムとして見えているので, 問題が生じた時に誰にサポートを求めたらよいのかわからなくなる. 実際のところ, 相互の実装が分からないままに問題点の抽出や解決策を考えるのは難しいのである. また, この GW のハブとしての重要性はベンダーロックインの可能性を招く. すなわち, この部品をコントロールする企業が他社のビジネスを操作することが可能になる. 例えばクラウドサービスに強い競争力を持っている企業が GW も提供し, かつその機能を自社のビジネスに有利なデバイスのみしか繋がないように制約することができる. すると, デバイスメーカーはそのクラウド企業に従属的な立場となり, 自由競争が阻害され, 最終的に顧客利益を損なうこととなる.

このゲートウェイのもたらす課題について, オープンソースを用いるアプローチを示すことが本稿の主題である. オープンソースの歴史や有用性については良く知られている通りである.Linux を筆頭として, 我々の ICT 生活を支える大黒柱の一つがオープンソースである[*1]. オープンソースソフトウェアのメリットの中で我々が注目しているものを以下に列挙する.

- **相互接続上の問題解決が容易**
  デバイスやクラウド接続のハブとなる GW がオープンソースとなることによって, そこに繋がる機器やサービスを提供する複数のメーカーのエンジニアすべてに対して, その実装が明らかになる. また, そのソースコードを改変することも可能になるので, 問題解決が促進される.
- **ベンダーロックインの回避**
  オープンソースを用いることでそこに導入される価値観が多様化し, ベンダーロックインへの動機がある企業に対抗することができ, 多様で自由な IoT システムが構築できる. このメリットは, ソフトウェアが企業体でなく中立な組織から公開されることでより強化される.
- **ソフトウェアの継続的な入手可能性, 不変性**
  GW が特定企業からリリースされていると, 一方的な仕様変更や, 時として公開中止になるリスクがある. しかし, 公開リポジトリをベースに開発され, 履歴も含めていつでもアクセスできるオープンソースは, ユーザーが納得している限りずっと使い続けることができる.

一方で, オープンソースに付随する以下のような問題点は, 我々が提案するオープンソース GW にも存在している.

- **サポートや動作保証を行う明確な主体が存在しない**
  オープンソースで最もよく使われ,PicoGW でも採用しているライセンスは MIT ライセンスである [1]. このライセンスは著作権表示と免責事項しかないため, このソフトウェアから損害を被った場合にも, 誰にもその責を問うことができない. オープンソースの品質を判断する基準は基本的に動作実績しかないが, これは保証をもたらすものではない.
- **ドキュメンテーションが不足しがちである**
  オープンソースは個人, または小さなチームの, 自らの必要性・内的動機によって開発されていることが多い. 自分の開発物について自分自身は熟知しているので, ドキュメンテーションが疎かになる傾向にある.
- **開発やメンテナンスが, 開発者の動機ベースになる可能性がある**
  前項で書いたように, オープンソースの開発動機は開発者の内的なものであることが多いためにバグフィックスや利用者からの質問への回答などが遅れる可能性がある. そして, 開発者の動機が消失すればメンテナンスされなくなる可能性が高い. このデメリットは, メリットの最後に書いた「ソフトウェアの継続的な入手可能性, 不変性」と相反しない. メリットに書いたことは, 同じソフトウェアを使い続けられる可能性に関することで, ここで述べたいのはバグの解消や時代の変化に追従し続けるメンテナンスに関することである.

このような, 商用ゲートウェイにない利点や, 独特な欠点を持ち合わせているオープンソースの国内における実績はまだ大きくない. 海外に OpenHAB[2] などオープンソース GW が存在しており,ECHONET Lite 対応なども全くないわけではないが, 全体的に見れば日本国内での利用実績は少なく, 日本語リソースも不足している. 国内では,AiSEG[3] や NextDrive Cube[4] などの商用ゲートウェイが大多数となっているのが現状である.

我々は商用目的に際して課題になりそうな点を明らかにしつつ要件を定義し, それに沿った形で新たにでオープンソースの GW を開発し,PicoGW(ピコゲートウェイ) の名で神奈川工科大学より 2017 年から公開を行っている [5]. PicoGW は 2019 年に商品化され, 紡績工場等において電力デマンド対応 IoT システムの基盤として利用されており, 安定稼働の実績を積み上げている [6]. 本稿では,PicoGW の開発コンセプトや商品化における課題, 今後の展望についても述べる.

2. 既存システム

商用 GW は HMS Networks による Intesis[7] や

[*1] 無料で提供されるフリーソフトウェアとソースコードが開示されるオープンソースソフトウェアは本来異なる概念だが, 本稿では簡単化のために無料かつオープンソースなソフトウェアを「オープンソースソフトウェア」と呼ぶ.

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

bus[8] がよく知られており, 国内でもたけびし社のデバイスゲートウェイ [9] や,HEMS 目的であればパナソニック社の AiSEG や [3],GW 単体でも NextDrive Cube[4] など多数販売されているが, そういった商用 GW の課題については前章で述べたとおりである. 本章では,PicoGW に至るまでの文脈や課題意識を明らかにするために,ECHONET Lite 登場前後より現在までの, オープン志向の GW の歴史をたどる.

2.1 D-HEMS

大和ハウス工業株式会社 (以下大和ハウス) は 2009 年にアプリケーション開発基盤として REST ベースの「住宅 API[10]」の仕様を公開し, それをホストする商用 GW である「D-HEMS」をリリースした. この API は Sony CSL の「萌家電」をはじめいくつかの Web アプリケーション開発を触発し [11], またハッカソンやコンテストを通じて一般のアプリケーション開発も行われた [12] が, これをフル活用するにはスマート家電を備えた住宅が必要であり, その市場の盛り上がりのタイミングに合わなかったためにコンテンツ開発会社の関心を引くに至らなかった.

2.2 Kadecot

住宅 API/D-HEMS はクロスドメインアクセスが実装されていないなど技術的な課題をいくつか抱えていたが, オープンソースではなかったためにこれを修正することが困難であった. そこで著者らはオープンソースの GW として「Kadecot（カデコ）」を開発し,2012 年に公開した [13].Kadecot は Android アプリケーションでありながら REST,WAMP(Web Application Messaging Protocol[14]) の API エンドポイントを持つ完全な GW として動作した. この Android 版は, 実行端末の差異の影響を受けやすかったり,OS バージョンアップに伴うメンテナンスコストの増大といった困難を抱えていたためプロジェクト終了となったが, 同様の形態でのデバイス GW というコンセプトは NTT ドコモが開発する「DeviceConnect[15]」に継承されている. また, μ ITRON 上で動作する互換実装である「μ Kadecot」もリリースされており, 独自の展開を見せた [16].

著者らは次に, より安定なプラットフォームを求めて,Linux 上で動作する Kadecot-JS を開発し公開した [17]. このバージョンは Node.js により実装され,Android 版より安定に実行でき, 開発やメンテナンスコストも大幅に下がっていたが, 用いた WAMP ルータ（Crossbar.io）の影響でインストールの手間と時間が大きく, 実行時のメモリ使用量も大きかった. 加えて当時,WAMP 自体があまり普及していないことも一つの懸念点であった.

Kadecot 系列の GW のもう一つの大きな問題としては, オープンソースとはいっても Sony CSL という企業からリリースされていたことも挙げられる. 競合企業からすれば, その機能が魅力的に見えたとしてもその利用は回避しようと考えるかもしれないし, 事実, そのリリース元の企業がこのオープンソースをいきなりクローズにしたり, ビジネス上の動機から機能を制約するようなことがないとは誰も言いきれないのである.

2.3 海外の主要なオープンソース GW

海外を見れば, オープンソース GW の実装は多い. 本稿ですべての GW の紹介をすることは無理なので, メジャーなものから三点紹介する.

openHAB は Java/OSGi で実装されたオープンソースのデバイス GW で,2010 年より Apache 2 ライセンスでリリースされている [2]. Raspberry pi 等も視野に入れた広範な開発コミュニティにサポートされており,ECHONET Lite のモジュールをリリースしているユーザーもいるが [18], 国内での使用実績は大きいとはいえない. また, メモリ使用量について我々が調査したところ, 最低でも 300MB を消費していた. これは安定時 50MB 未満という PicoGW のメモリ消費量に比べるとかなり大きい.

Home Assistant は Python で実装されたオープンソースのデバイス GW であり,EPL2（Eclipse Public License）にて 2013 年からリリースされている [19].1700 以上ものプラグインがリリースされており (2020/11 現在),ECHONET Lite 用のプラグインも存在しているが [20],openHAB 同様, 国内の使用実績はあまり大きいとは言えない.Home Assistant が一番に想定しているプラットフォームは Raspberry pi であり,OS イメージが提供されている.OS イメージからのインストールは確実だが手間がかかる.Python3.8 以上が動作すれば virtualenv を用いたインストールもできるので我々が試した結果, メモリ消費量も 100MB 前後であり, 比較的省メモリで動作するようである. ただし Home Assistent は UI も作りこまれた, これ単体でサービスを提供できるトータルなシステムであり, 商用システムの部品として組みこむような用途にはあまり適していないように思われる.

Domoticz は C++で実装されたオープンソースのデバイス GW である [21]. インストールは容易であるが, プラットフォームとして Raspberry pi が筆頭に挙げられていたり,GPLv3 でリリースされているなど, 一層 DIY ユーザーが念頭に考えられているようであり, 商用用途に適するとは言えない. また, 他のオープンソース同様, 国内での実績に乏しく, 日本語ドキュメントも存在しない.

3. 商用利用に適するための要件

PicoGW は, 商用利用に適するオープンソース GW をめざして開発された. そのための機能要件として, 以下の項目を考慮した.

- **R1: インストールと基本的機能の利用が容易なこと**

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

GW のユーザーとして想定されるのは, デバイスメーカー, クラウド企業,Web 開発企業など様々であり, 技術的スキルも様々であると考えられる. インストールや基本的な機能利用の技術的ハードルが低いことは最も重要な要件である.

- **R2: 一通りのドキュメントが存在すること**
  オープンソース利用において頻出する問題の一つがドキュメント不足である. 特に, ネット上に解説記事が増えるまでの間, アーリーアダプターが使いこなせる程度にはドキュメントが存在する必要がある.
- **R3: 長時間安定に実行できること**
  GW はインフラとして長時間利用されるソフトウェアである. また, オープンソースの品質を信じる一番の根拠は利用実績であることから, 開発初期から安定に動作し, 問題発生時にも動き続けるような仕組みが必要である.
- **R4: 同じ API を長期間使い続けられること**
  住宅やビル・工場などにおける設備は納品後は変更なく長時間稼働し続けることが期待されるのに対し, ソフトウェアはとにかく早くリリースして, だんだんに改善していくという開発モデルになっており, これが設備の一部として IoT システムを構築していくうえでの一つの障害になっている. 一度利用できた API は, 半永久的に使い続けられることが望ましい.
- **R5: 改変・機能追加が容易であること**
  オープンソースを企業が利用する場合, そのブランドイメージや既存の商品群との整合性を維持するために改変したい, あるいは単に足りない機能を追加したいと考えるのは自然なことである. このような場合に, 変更を加える手間が大きいと利用をためらうだろう.
- **R6: クラウドとの接続が容易であること**
  現代の GW であれば, クラウドとの親和性が高いこともまた重要である. そして,R4 とも関連して, クラウド技術の変化に影響されてすぐに使えなくなる可能性は（完全に予測することは不可能であるものの）最小限にとどめておきたい. すでに枯れていて, 今後も継続することが予想できる技術を用いてクラウドと接続することが重要である.
- **R7: 中立的な主体から公開されること**
  企業には通常ライバル企業が存在する. そのライバルを利する, あるいはライバルに依存するような活動を行うことは難しい. 企業間政治の影響を受けにくい主体から公開されることも重要である.

4. PicoGW

本稿で紹介する PicoGW は, 過去の我々の開発経験を踏まえつつその問題点を克服するものとして, 前章で示したような要件を念頭にスクラッチから開発された. 目指すところは住宅 API の時代から大きくは変わっていないが, 商用化に耐える性質を持つよう細かく検討を行った。全体アーキテクチャを図 1 に示した。

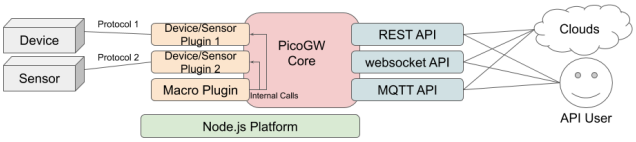

図 1 PicoGW アーキテクチャ
Fig. 1 The Architecture of PicoGW

4.1 開発言語の選択

新たな GW を開発するにあたって最初の選択肢は, どの言語を用いるかという点である. 現在サーバープログラミング言語としてメジャーなものは,Java, PHP, JavaScript(Node.js) ,Ruby ,Scala, Go などがあるだろう. 特に GW 開発においては,Java ベースの OSGi(Open Service Gateway Initiative) フレームワークが良く使われている. 動的な機能の追加・削除ができることが大きな利点であり, 実際 openHAB など多くの GW で採用され, 広い実績を獲得している. しかし, 我々は Java/OSGi を採用せず,Node.js を選択した. その理由は, 以下の通りである.

インストールの容易さ

Node.js のパッケージは npm というリポジトリに集積されており, システムに npm コマンドさえインストールされていれば基本的に一行コマンドでのインストールが可能である.PicoGW のインストールは, 以下のようになる.

npm i -g picogw

このインストールの容易さは要件 R1 に対応している.

また,Node.js がサポートする OS の多様さを反映して,PicoGW も Linux, Windows, MacOS, Android（Termux というシェルアプリ上で動作させる）で動作する.FreeBSD でも動作したという報告がある. この点も R1 にマッチしていると言えるだろう.

言語の習得の容易さ・継続性

IEEE Spectrum によれば,2020 年現在人気のある言語は上から順に Python, Java, C, C++ JavaScript である [22]. また,JavaScript は習得が容易な言語として言及されることが多い [23]. さらに,JavaScript は Web ブラウザの中で動作する, 歴史の長い言語であり, 今後も Web ブラウザが続く限り（互換性のためにも）存続する可能性が高いと考えられる. ユーザーが多く, 習得が容易で, 継続性が見込める JavaScript は要件 R4 にマッチしている.

実行時のメモリ使用量

OSGi の代表的な実装の一つである Apache Felix をインストールした直後（Gogo シェルのみが使える状態）のメモリ使用量は, 我々が調べたところ 90MB 弱であった.

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

ここに Web サーバやプロトコルスタックを足していくと,OpenHAB のように容易に数百 MB となってしまうと想像される. PicoGW の安定時の実行時使用メモリは 40MB を上回る程度である (図 2).

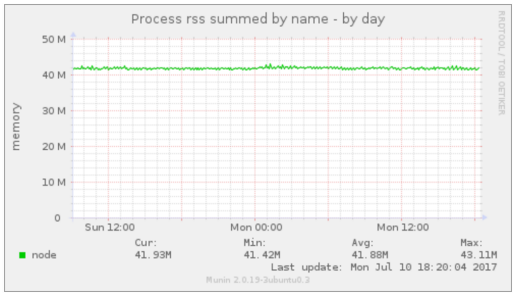

図 2 PicoGW 実行時メモリ使用量
Fig. 2 Memory usage during PicoGW execution

このメモリ消費量が少ないという利点は, ローパワーなデバイスでの実行を可能とし, 一時的なメモリ使用量増加にも対応できるという意味で, 長時間の安定実行にも寄与する. これは要件 R3 に対応するものである.

一方で,Node.js を利用する場合の欠点も存在する. 一つには, プラグイン追加を行ったときにサーバーの再起動が必要になる点である. もう一つには,JavaScript は動的型付け言語であるため, 実行時にならないと出現しないバグ混入の可能性が増大する. これは長期間の安定実行を求めるという R3 にマッチしない特徴であるが, 我々はメリットの方が大きいと考え,Node.js を選択している.

4.2 API エンドポイント

PicoGW の基本機能は, 様々なデバイスに統合的な API エンドポイントを通じてアクセスすることである.PicoGW ではその方式として,HTTP REST, Websocket, MQTT を提供している. どれも使いやすく広範な利用実績のあるプロトコルであり, 要件 R1 に対応する. 特に HTTP REST と Websocket は JavaScript と同様, ブラウザがある限り継続することが期待されるプロトコルである (要件 R4 にマッチする）. また,MQTT プロトコルは AWS はじめ現在の多くの商用クラウドが API のインターフェースとして用意しているものである. これは要件 R6 に適合している.

マクロ機能

また,PicoGW は外部にクライアントを用意しなくても,GW 内で任意のプログラムを走らせることができるマクロ機能が存在する. マクロはサンドボックス内に用意された JavaScript の処理系であり,PicoGW の Web フロントエンドでプログラムを記述できる (図 3).

マクロ内から外部のパッケージを呼び出すことは禁止されており, セキュリティ上の懸念は少ない. マクロ内で可能なのは以下の操作である.

- JavaScript 言語の基本機能

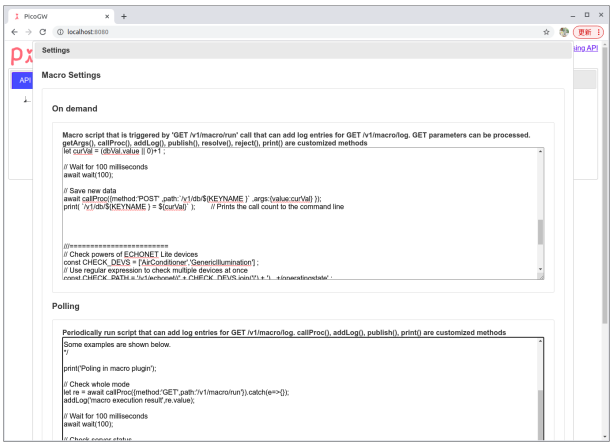

図 3 マクロプラグインの設定画面
Fig. 3 Macro Plugin Settings

- 他のプラグインにアクセスし, その機能を呼び出す
- 自分自身が API を実装し, 他のプラグインやユーザーに対して特定の機能を提供する

マクロの実行タイミングは, 直接 API としてコールされたときと, 設定により定時実行される場合である. 実行中に Websocket や MQTT で接続しているクライアントに対して通知を送信することも可能である. 単に機器の定時ログをとる目的や, 機器同士の通信だけを目的に GW を利用する場合には, マクロ機能だけで十分である. このマクロ機能も GW の機能拡張を大幅に容易にしており, 要件 R1 と R5 を満たしている.

4.3 安定実行のための工夫

サーバーは高可用性が必要である. そのための対策の一つが, クラッシュ時の自動再起動である. 再起動時に問題が残存していれば再びクラッシュする可能性もあるが, そうでない場合は自動再起動はサービス中断を最小限に抑えるうえで有効な対策の一つである. PicoGW は forever というデーモン化パッケージを用いてメインのロジックをラップすることにより, クラッシュ時に自動的に再起動を行う.

もう一つの対策が, ファイルシステムへの書き込みをコントロールすることである. バグもなく正常稼働していたはずのシステムが長期運用でクラッシュする大きな原因の一つが,SD カードなどのフラッシュメモリが, 書き換えを重ねることにより劣化し, 不良セクタが出現することである.PicoGW ではこの対策として, プロセス内でディスクへの書き込みが必要になった場合は必ず専用オブジェクトを介するようにし, 実体は特定のディレクトリ内（デフォルトでは ~/.picogw）に集積することとした. これにより, サーバーマシンを構築する際に, この書き込みディレクトリだけ耐用回数の多いファイルシステムにマウントするか, あるいは overlayfs 等を用いて,RAM ディスク上に配置するように設定しておくと, ファイルシステムの寿命を大幅に長くすることができる. この機能は要件 R3 にマッチして

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

いる.

この, ディスク書き込み時の特定方式の利用は, プラグイン開発者に守って欲しい一つのルールである. このようなルールは他にもあるが, 特に重要なのは API の上位互換を保つということである. 上位互換が保てないような改変を API に施したい場合は, プラグイン名を変更して, 元の API も維持するように努めることを要求している. これは要件 R4 を満たすことを目的としている.

4.4 機能拡張 (プラグイン)

PicoGW の機能を拡張するにはマクロを用いるのが最も簡単だが, プラグインを開発するのも難しくはない. プラグインも通常の Node.js のプログラムで, 個別に npm パッケージとして公開しておけば, 本体の PicoGW とは独立に npm でインストールでき,PicoGW の実行時には自動的に認識されて利用できるようになる. この際, 以下のような制約がある.

- プラグインのパッケージ名に node-picogw-plugin- という接頭子をつける
- PicoGW, プラグインともにグローバルインストールされていなければならない

これらは PicoGW 本体が, システムにインストールされているパッケージの中からプラグインを効率よく発見するための手段である. これも同じく PicoGW がプラグインを見つけるための手段である. 特に二点目は強い制約だが, 対象プラットフォームの中にグローバルインストールでなければパッケージを見つけられないものがあったため, 汎用性を重視して必須とした. なお, プリインストール以外の外部パッケージを使うつもりがなければ,PicoGW をローカルインストールしても動作可能である.

プラグイン実装に当たっては API エンドポイントの種類を意識する必要がない作りになっており,HTTP REST, Websocket, MQTT, マクロ経由, どの方式であっても同じ引数で同じコールバックが呼ばれるようになっている. 詳細は Web にドキュメントを掲載した [24].

このようにして開発したプラグインを公開するのは, 通常の npm パッケージと同じであり, インストールもグローバルインストールが必要な以外は通常通りである.

この容易さは要件 R1 と R5 を満たすものであり, またプラグイン開発上の指針が,R4 の API の不変性を促すものとなっている.

4.5 ドキュメンテーション

PicoGW の日本語ドキュメンテーションは Web 上に存在する [25]. ただし, 近年のドキュメンテーションは使いながら必要なだけ知識を得られるよう工夫されているものが多いため,PicoGW でもインタラクティブなフロントエンドを持たせ, そこから API が検索できるようになっている（図 4）.

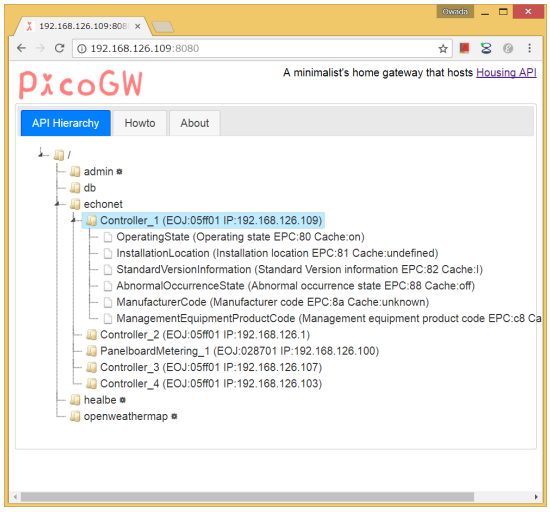

図 4 PicoGW フロントエンド
Fig. 4 The Frontend of PicoGW

また, オープンソースの課題抽出や疑問・問題点の解消にはサポートコミュニティが育つことが必要だが,PicoGW はまだその状態には至っていないと認識している.

4.6 配布主体

PicoGW は神奈川工科大学より公開されている. これにより, 要件 R7 を満たすことができた. また別のメリットとして, 神奈川工科大学の HEMS 認証支援センターに多数の情報家電が設置されているため, 開発上の効果もあった.

5. 商品化

PicoGW の最初のリリースは 2017 年の 12 月であった.2019 年に FA ソリューションを販売していた KYOSO 社による PicoGW 商品化の話が持ち上がった.KYOSO 社は工場にセンサーを設置し, 商用 GW に装着された SORACOM 社の SIM を通じて AWS に情報を蓄積し, 独自の可視化を行うパッケージ商品を販売していた. この GW を PicoGW に置き換えることで新しいセンサー機器を使うことに関心があったようであった. 特に, 紡績工場からデマンド対応を行う FEMS へのニーズが発生しており,modbus 経由で情報を取得できる安価なオムロンの CT 電力センサーを用いることができれば有益とのことであった.

そこで, 大和ハウスと SonyCSL は共同で,PicoGW 用の modbus プラグインを開発した.modbus とは RS-485 ベースの, かなり歴史がある枯れたシリアル通信プロトコルで, オムロンに限らず多数のメーカーが対応機器を販売している. このプラグインはオープンソースではなく, 独自のライセンスを用意し,KYOSO 社に対しライセンス販売を行った.PicoGW はプラグインを別パッケージとしてインストールするため, その時にユーザーに対して独自の EULA を提

Vol.2021-GN-112 No.5
Vol.2021-CDS-30 No.5
Vol.2021-DCC-27 No.5
2021/1/25

示することもでき, ライセンスの切り分けが明確であることもメリットであった.

KYOSO 社は PicoGW のマクロを記述し, 一定時間間隔で modbus より情報を取得し, それを publish するようにした.API エンドポイントとしては MQTT を用い,AWS IoT コアを用い,AWS 内の MQTT Broker と接続する.

実装を進める中で,GW の ID として SIM の IMSI を使えないかという要望があった.PicoGW は自分自身の ID として Mac アドレスを用いているが,GW マシンが故障した場合に, 機器を入れ替えても同じ SIM を使うことで ID を不変にしたいからということであった. そこで, それまで ID の変更ができなかったのを変更できるようにした.

その他, マクロの実行頻度を細かく調整できるようにしてほしいなどいくつか要望はあったが, 大きな変更は必要なかった.KYOSO 社は PicoGW を組み込んだパッケージを「MORAT GW」と名付けて販売を開始し [6], 紡績工場に納品し,2020 年 11 月現在でも稼働を続けている 5. 今後は紡績工場での適用拡大及び、農業 IoT におけるセンサーシステムの構築案件もあり、適用範囲が広がっているとのことである。

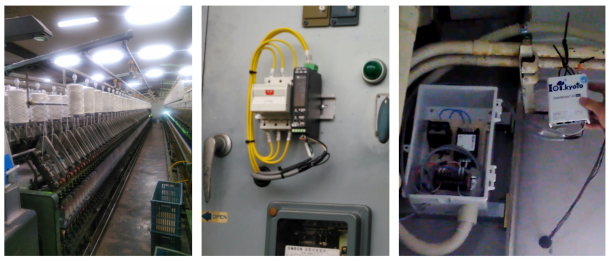

図 5 紡績工場における PicoGW の実用例
Fig. 5 An Applications of PicoGW in a Spinning Mills Factory

また,modbus プラグインを開発した一方の大和ハウスは, 住宅を製造する自社の奈良工場や九州工場（D's Smart Factory[26]）で PicoGW を利用し, 稼働させている. こちらではオムロンの電力センサに加え, 環境センサ (EnOcean) や空調制御・照明制御にも PicoGW の modbus プラグインを利用している. センサーや設備機器に元々 modbus のインターフェースがなくても変換器が入手できることがほとんどなので,modbus の有用性は非常に高いとのことである. こちらでもクラウドは AWS を利用している.

6. 商品化に関する考察

この商品化によって, 我々の開発したオープンソースの PicoGW は, ライセンス体系の違うプロプライエタリプラグインを組み合わせることにより, 商用利用が可能であることが示された. しかし, そのプロセスでいくつかの課題も明らかになっているため, それについて本章で議論する.

6.1 サポートの問題

PicoGW のほとんどの部分は MIT ライセンスでリリースされており誰からも免責されているため, 基本的にはサポートがない状態である. また, 実はプロプライエタリな modbus プラグインのライセンス契約も, 現時点では実験的なフェーズであり, ライセンサーもサポート義務を負わない旨書かれている. すなわち, 本商品化は誰もサポートしないものが販売されていることになるのである. サポートについては, ある程度の販売数が見込めた段階で再考する取り決めになっている. 現時点では今回は関係者がみな, 新たな世界に踏み込むのだという目標を共有し, それぞれ持ち出しで努力するのだという意思をベースに一時的に我慢しているのが実情であり, ビジネス上のリスクである.

6.2 CT ベースの電力センサーを用いている

よく知られているように, 電源ケーブルにクランプするタイプの電力計は大まかな使用電力はわかるが, 実際の料金計算に使えるほどの精度は得られない. 正確な値を取得するにはスマートメーターが必須である.PicoGW はデフォルトで IPv4/v6 の ECHONET Lite に対応しており, 高圧スマートメーターの値も直接読み取ることが可能である. しかし,KYOSO 社が今後 ECHONET Lite の高圧スマートメーターを使う可能性があるかと尋ねたところ, あまり前向きには考えていないとのことであった. 理由は,ECHONET Lite から値を読み取るには SMA 認証を通ったコントローラが望ましく,PicoGW は SMA 認証を通っていないのと, スマートメーターにはその値をパルス出力できるものがあり, それで読み取ったほうが低コストにシステムを組めるから, とのことであった. もちろん ECHONET Lite のスマートメーターオブジェクトには詳細で便利なプロパティが多数定義されているが, 現時点で KYOSO 社に対してアピールするものにはなっていなかったようである. ただ, スマートメーターだけを見ていればパルスカウンタとの比較になってしまうのかもしれないが, サブメーターや分電盤等, 他の ECHONET Lite 機器と組み合わせて用いるといったケースを想定すると,ECHONET Lite でまとめるということにはメリットが生まれてくるようにも思われる.

6.3 遠隔操作の機能が不十分

運用コストを下げるための機能としては, 遠隔で様々なことができるということがあるだろう. 実は遠隔からのプラグインインストールや本体のアップデート機能, 再起動などは,Sony CSL 社による実装が当初から存在していた. しかし, この機能は Firebase で実装されており,KYOSO 社がそれまでに基軸として用い, ノウハウを多々蓄積していた AWS との適合性が低く, 使われなかった. 遠隔操作に関しては, たいていはクラウドシステムに依存する部分が発生し, クラウドは互いに競合している面もあるため難しい

のかもしれないと感じた. ただ, 既存の API 体系の機能上にそういった機能があったとしたら, 例えば MQTT からコマンドを送るだけでアップデートや再起動ができたとしたら,KYOSO 社もそれを利用したかもしれない.

7. まとめ, 今後の課題

本稿では, 商用利用に適したオープンソースのデバイスゲートウェイである PicoGW を紹介した. 商用利用を促進するためにどのような要件を定義し, 実装したか, そして, それが実際の商品化につながったことを報告し, 現状の課題についても議論した.

今後は利用者・開発者コミュニティの広がりに期待したい.

参考文献

[1] MIT: The MIT License, https://opensource.org/licenses/MIT. Accessed: 2020-12-9.
[2] openHAB Foundation: openHAB, https://www.openhab.org/ (2010). Accessed: 2020-11-21.
[3] Panasonic: AiSEG2 ラインアップ, https://www2.panasonic.biz/ls/densetsu/aiseg/aiseg2/. Accessed: 2020-11-25.
[4] Nextdrive: Cube の製品概要, https://support.nextdrive.io/ja-JP/support/solutions/articles/26000031419-cube-%E3%81%AE%E8%A3%BD%E5%93%81%E6%A6%82%E8%A6%81. Accessed: 2020-11-25.
[5] Kanagawa Institute of Technology: npm: picogw, https://www.npmjs.com/package/picogw (2017). Accessed: 2020-12-9.
[6] 株式会社 KYOSO：【新製品】クラウドネイティブなセンシングコントロール GW「MORAT GW」の販売を開始いたします, https://prtimes.jp/main/html/rd/p/000000002.000013252.html (2019). Accessed: 2020-11-25.
[7] Networks, H.: Intesis ゲートウェイ, https://www.intesis.com/ja/products/protocol-translator. Accessed: 2020-11-25.
[8] Networks, H.: Anybus Gateway 製品一覧ページ, https://www.anybus.com/ja/products/gateway-index. Accessed: 2020-11-25.
[9] 株式会社たけびし：デバイスゲートウェイ, https://www.faweb.net/product/devicegateway/. Accessed: 2020-11-25.
[10] 大和ハウス工業株式会社：住宅 API, https://www.daiwahouse.co.jp/lab/HousingAPI/. Accessed: 2020-11-21.
[11] SonyCSL: 大和ハウス工業ニュースリリース | 大和ハウス工業 × ソニー CSL スマートハウスの家電機器を、ゲーム感覚で制御する公開実験を実施, https://www.daiwahouse.co.jp/release/20110707094432.html. Accessed: 2020-11-21.
[12] 大和ハウス工業株式会社：家 CON 2015, https://www.daiwahouse.co.jp/lab/HousingAPI/iecon2015/index.html. Accessed: 2020-11-21.
[13] SonyCSL: Kadecot（カデコ）, https://www.sonycsl.co.jp/tokyo/347/ (2012). Accessed: 2020-11-21.
[14] : The Web Application Messaging Protocol, https://wamp-proto.org/. Accessed: 2020-11-21.
[15] Docomo, N. T. T.: GotAPI, https://device-webapi.org/gotapi.html. Accessed: 2020-11-21.
[16] Toppers, C. .: uKadecot – Contributed Software, https://dev.toppers.jp/trac_user/contrib/wiki/uKadecot. Accessed: 2020-11-21.
[17] SonyCSL: Kadecot-JS, https://github.com/SonyCSL/Kadecot-JS (2016). Accessed: 2020-11-21.
[18] Matsuda, K.: ECHONET Lite Binding for openHAB, https://github.com/kaz260/org.openhab.binding.echonetlite. Accessed: 2020-11-23.
[19] Home Assistant Core Team and Community: Home Assistant, https://www.home-assistant.io/ (2013). Accessed: 2020-11-23.
[20] Phillips, S.: mitsubishi_echonet - ECHONET Lite library for Home Assistant, https://github.com/scottyphillips/mitsubishi_echonet. Accessed: 2020-11-23.
[21] Domoticz: Domoticz, https://www.domoticz.com/. Accessed: 2020-11-23.
[22] Cass, S.: The top programming languages: Our latest rankings put Python on top-again - [Careers], *IEEE Spectrum*, Vol. 57, No. 8, pp. 22–22 (2020).
[23] 株式会社パソナテック：【難易度順】プログラミング言語を一覧で一挙紹介！, https://www.pasonatech.co.jp/workstyle/column/detail.html?p=2505. Accessed: 2020-11-25.
[24] Hoikutech: PicoGW プラグインの開発方法, https://hoikutech.com/nanogw/picogw-plugin-development/. Accessed: 2020-11-25.
[25] Hoikutech: NanoGW Project, https://hoikutech.com/nanogw/. Accessed: 2020-11-25.
[26] 大和ハウス工業株式会社：環境配慮型工場のショールームディーズ スマートファクトリー, https://www.daiwahouse.co.jp/business/production/factory/nara.html. Accessed: 2020-12-9.