Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

Node-RED による HEMS のための複数スマート家電規格の統合

山中悠輝[†1]　加藤丈和[†1]

概要：本稿では、複数のスマート家電規格に対応し、相互に連携する HEMS のためのフローベースアーキテクチャと，Node-RED による実装について述べる．近年，スマート家電の通信規格は乱立状態で，また，アプリケーションの違いから取得できる状態や制御項目も規格ごとに異なっている．さらには家庭内にはネットワーク対応していない家電も数多い．そこで本研究では，HEMS を中心とするスマートハウスアプリケーションを対象に，家電に対する共通の状態取得，制御のモデル（家電モデル）を定義した上で，異なる規格のスマート家電やアプリケーションの間で，相互に連携するためのフレームワークを開発した．さらにスマートタップやスマート分電盤，スマートスイッチなどの組み合わせを見かけ上 1 個の家電とみなすことで，ネットワーク対応しないレガシー家電も含めたスマートホームアプリケーションの開発基盤を構築した．

キーワード：Node-RED，HEMS，スマート家電，スマートハウス

1. はじめに

現在，国内外の家電市場においてスマート家電が急速に普及しつつある．ここでいうスマート家電とは，ネットワークに対応し，スマートホンやスマートスピーカーから遠隔で制御や情報取得が可能な家電のことを言う. 近年では，Apple HomeKit [1]や Amazon Echo[2], Google Home [3]などのクラウドプラットホームやサービスメーカが主導するスマート家電規格が注目されている．しかし，単にスマート家電と言っても様々な規格が乱立している状態であり，個々のスマート家電メーカが対応規格を増やさなければ相互に連携することは困難である．

スマート家電，スマートホームと関連の深いシステムとして Home Energy Management System（HEMS）[4]がある．HEMS は家全体の消費エネルギーの最適化を図るシステムの総称であり，家庭内の家電や電気設備の電力エネルギーを一括管理し，電力消費量を最適化して無駄な電力を削減したり，設定したしきい値以下に制御することを目的としたシステムである．HEMS で連携できる家電はネットワーク対応できるスマート家電に限られ，さらに HEMS を販売するメーカの通信規格に限られる場合が多い．

本研究の目的は，近年普及してきた複数のスマート家電規格を統合して相互に連携しながら，HEMS のように家全体を一括管理するアプリケーションを構築するためのフレームワークを開発することである．

HEMS のような家全体の家電を一括管理するシステムのポイントは，家庭内のすべての家電を対象とすること，ユーザ自身による直接的な家電操作や，家電自身の自律的動作によって個々の家電は非同期に状態を変えること，家庭内には複数のスマート家電規格が存在し，またどのスマート家電規格にも対応しないレガシー家電が存在すること等が挙げられるが，本研究では特に，家電規格の混在と，家電動作の非同期性に着目する．

我々はすでに，ECHONET-Lite[7]を中心とした家電ネットワーク規格を相互連携し，HEMS を実現する Energy On Demand(EoD)システム[6]を提案している．このシステムでは，OSGi[13]という JavaVM 上にバンドルと呼ばれる動作中に更新可能なモジュールを追加できるフレームワークを使用して，個々のスマート家電規格をデバイスバンドルとして登録することにより複数規格への対応を行った. また，個々の家電の状態を管理するために, 自律的な動作単位（ここでいう動作単位は一般的にはスレッドやプロセス）を持つオブジェクトであるエージェントモデルを導入し，個々の家電に対応する家電エージェントによって家電ごとの非同期の状態変化に対応した．

しかし，個々の家電に対応する動作単位をもつ家電エージェントのモデルでは，家電の数だけのエージェントが必要であり，また個々の家電エージェントと実際の家電との間のメッセージ（ネットワーク経由の状態取得や制御コマンド）が非同期の場合に，さらに各エージェントが複数スレッドを持つなどの非同期通信に対応させる必要があり，一般的な家庭内のホームゲートウェイで動作させるには負荷が大きいという問題点があった．他にもユーザとシステムのやり取りに関してはアドホックに家電エージェントにメッセージを送ることで実現していたため，最近のスマートスピーカなどのアシスタントの機能に対応させることも困難であった．

そこで，本研究では，システムの非同期性の本質は，家電単位ではなく，UI と家電，HEMS の間を飛び交うメッセージの非同期性にあると考え，新たにメッセージフローベースの HEMS 向けアーキテクチャを提案し，フローベースプログラミングを簡易に行える Node-RED[12]による実装を示す．

提案するシステムは，複数のスマート家電規格の間で相

†1 静岡理工科大学
Shizuoka Institute of Science and Technology

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

互に連携するだけでなく，ネットワーク対応していないレガシー家電についてもスマートタップやスマート分電盤とスマートリモコンの機能を組み合わせて仮想的にスマート家電として扱う．また，Apple Home アプリや, Amazon Echo などのスマートスピーカと連携して動作させることができる．

以下，2 章では 3 層レイヤによる HEMS のためのメッセージフローアーキテクチャについて説明し，3 章ではレガシーな家電と各種電力センサとの連携について，4 章にて実際に Node-RED を用いて作成したフローと実験結果を説明する．

2. HEMS のためのメッセージフローベースアーキテクチャ

フローベースアーキテクチャは，処理を行うノードと，ノード間を流れるメッセージフローから構成される．ノード間のメッセージはそれぞれ非同期に処理される．本研究では，家電やユーザとのコマンドのやり取りや，システム内部の情報のやり取りをメッセージとして，そのフローを制御することで，非同期性をもった HEMS システムを構築する．

2.1 アーキテクチャの概要

現在主流となっている，Apple HomeKit や Google Home, Amazon Echo などのスマートホームアプリケーションは，スマートホンやスマートスピーカからアプリの画面や音声認識機能などをつかってシステムに指示を出すユーザインターフェース部分と，スマート家電に制御コマンドの送信や状態取得を行うデバイス操作部分からなっている．本研究では，ユーザインターフェース部分を上位層，デバイス操作部分を下位層として，さらに複数のスマート家電規格の統合と相互連携を実現するために，標準的な家電のモデルを定義し，家電相互の連携をスマート家電規格の種類にかかわらずに行うための中間層を加えた 3 層のアーキテクチャを提案する(図 1).

上位層はユーザインタフェースアプリケーションの役目を持ち，中間層の家電の情報を提示しつつ，UI からの指示を中間層に伝える．下位層はスマート家電との通信の役目を持ち，実際のスマート家電規格に合わせた通信をおこなって家電を操作する．

上位層では独自のユーザインタフェースアプリケーションだけでなく，Apple Home, Amazon Echo などの既存のアプリケーションとのブリッジとしても機能する. この場合，中間層の家電モデルをそれぞれのアプリケーションの規格に対応する家電に見せかけ，アプリケーションからの指示を家電モデルにつたえるプロトコル変換の役目をもつ．新しいアプリケーションに対応したい場合は上位層のプロトコル変換部を追加すればよい．

一般的なスマート家電の場合，アプリケーション・ユーザインターフェスと家電コントロールという 2 層で構成されることが多いのに対して，3 層構造としたことでアプリケーションが対応していない規格のスマート家電とも連携できるだけでなく，Amazon Echo で指示した内容を Apple Home に反映させるといったアプリケーション同士の連携も可能となる．以下，提案アーキテクチャのコアとなる中間層と，アプリケーションや家電とのブリッジの役目を果たす上位層，下位層について説明する．

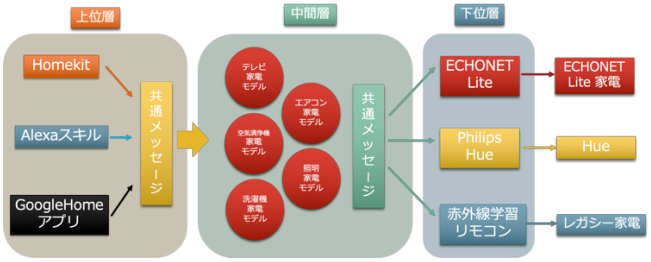

図 1　提案システム概要

2.2 中間層

中間層は，家電の状態管理と，家電への制御コマンド送信の標準的なやり取りを定義する層である．中間層への荷電状態問い合わせや中間層からの実際の家電へのメッセージ送信は，アプリケーションやスマート家電規格によらない共通メッセージを使用する．

2.2.1 家電モデルの定義

メーカーやモデルの違いによって生じる制御項目と状態項目の違いを統一するために，家電の種類毎に共通で，状態や消費電力値を格納できる家電モデルを定義した．家電モデルの具体的な例と電力値データの取得方法，その働きなどを次に述べる．

家電モデルの例

家電モデルには全ての家電に共通する最低限必要な状態・制御項目と，家電の種類毎の状態・制御項目を 1 つにしてまとめることにした．ここで，エアコンの家電モデルの例を図 2 に示す.

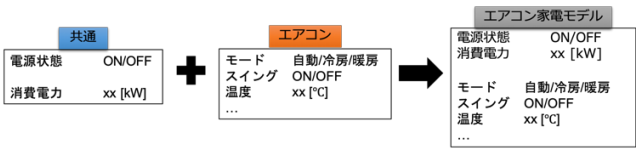

図 2 エアコンの場合の家電モデル

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

図 3 Node-RED ダッシュボードによる独自インタフェース

2.2.2 中間層における家電モデルの働き

中間層では，家電モデルは定期的に消費電力や動作モードなどの状態問い合わせのメッセージを発行して，下位層を経由してスマート家電に送信し，スマート家電からの返答を受け取ると，家電の状態をアップデートする．上位層のアプリケーションから状態問い合わせに対しては，家電モデルで管理している状態を返答する．また上位層からの操作要求を家電モデルが受け取ると，家電制御メッセージを下位層を経由してスマート家電に送信する．つまり中間層を経由したメッセージフローによって，上位層と下位層はお互いにメッセージの送信先の通信規格によらずにやり取りが可能となる．

また，中間層における家電モデルは，対応するスマート家電への状態問い合わせだけでなく，独立したスマートタップなどの電力センサからの消費電力値データや温度照度センサなどの家電に関連する情報も一緒に格納する．つまり，通信デバイスと一対一対応である必要はなく，複数のセンサや制御機器を対応付けて，一つの家電の状態管理や制御を行うことができる．この点については次章のレガシー家電対応で詳細に述べる．

2.3 上位層

上位層はアプリケーションとユーザインタフェースの層で，主にアプリケーションから操作要求メッセージを共通コマンドに変換し，中間層に流すブリッジとして機能する．現在，Apple 社製の HomeKit[1]と，Amazon.com 社製の Amazon Echo とそのシリーズ機[2]に搭載されているスマートホームスキル[4]に対応するノードと，独自インタフェースノードを実装している．

上位層の機能の一つは，HomeKit や Amazon Echo から，中間層で管理する家電モデルをそれぞれのアプリケーションに対応する家電であるようにエミュレートすることであり，既存の HomeKit アプリケーション，Amazon Echo のスマートスピーカからの操作をシステムに伝えるだけでなく，Amazon Echo でおこなった操作の結果を HomeKit に反映させるなどの連携が実現できる．

2.3.1 HomeKit[1]

HomeKit は Apple 社製の iOS 機器に標準的にインストールされているスマートホームアプリケーションである．アプリケーション画面及び同社製の AI アシスタントである Siri に話しかけることでも家電を操作するコマンドを出力可能である．提案システムでは，上位層ノードがこれらのコマンドをスマート家電の代わりに受け取り，また家電の状態を HomeKit に送信する．

2.3.2 スマートホームスキル[4]

スマートホームスキルは Amazon.com 社が開発したスマートスピーカーの Amazon Echo シリーズに搭載されているスキルの 1 つである．こちらもアプリケーションからでも Amazon Echo に話かけることでも家電を操作するコマンドを出力し，上位層のノードが家電の代わりにやり取りを行う．本稿では Amazon Echo dot を使用した(図 4).

図 4 Amazon Echo dot

2.3.3 ウェブアプリケーションによる独自インターフェース

個々の家電モデルの状態を視覚的にモニタするための独自インタフェースとして，家電モデルの状態を表示，操作，設定する機能を作成した(図 3)．このインタフェースは Node-RED dashboard[14]によるウェブアプリケーションとして実装しており，ウェブブラウザから各家電や家全体の消費電力の確認や個々の家電への操作要求の送信を行う．また，後で述べる HEMS 機能のパラメータ（消費電力しきい値や家電の優先順位など）の設定を行えるようにした．

2.4 下位層

下位層では中間層からの共通コマンドを家電の規格に沿った形式に再変換して家電へ実際にコマンドを送信する．また，スマートタップやスマートリモコンなどの家電以外

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

のデバイスとのインタフェースも下位層のノードとして実装する．

現在は，ECHONET Lite[7]対応ノード，Philips 社製の Hue[10]対応ノード, 学習リモコンに対応するノードをそれぞれ実装している．

2.4.1 ECHONET Lite[7]

ECHONET Lite は日本の家電メーカーが中心となってエコーネットコンソーシアムが策定した通信プロトコルである．ECHONET Lite 対応家電を動かすためには ECHONET Lite フレームと呼ばれる専用の電文を家電とコントローラー間で相互に送り合うことで家電の状態の取得や家電の操作が可能になる．ECHONET Lite 対応ノードは，コントローラーとして実装しており，中間層との間で通信プロトコルの変換を行う．今回使用した ECHONET Lite 対応家電のリストを表 1 に示す．

表 1　ECHONET Lite 対応家電

名称	型番	メーカー
オーブンレンジ	AX-XW300-R	SHARP
エアコン	AY-G22S	SHARP
エアコン	CS-227CFR	Panasonic
空気清浄機	KI-EX100	SHARP
洗濯機	TW-Z96X1	TOSHIBA
照明	LEDH-LT2	TOSHIBA
冷蔵庫	GR-G51FXV	TOSHIBA
ロボットクリーナー	VC-RCX1	TOSHIBA

ECHONET Lite は家電だけでなく，スマート分電盤やスマートタップのための規格もあり，本研究ではレガシー家電対応の消費電力取得にも用いている．

2.4.2 Philips Hue

Hue は Philips 社製のコネクテッド照明である(図 5). Hue は Zigbee ベースの独自プロトコルで照明の制御を行うが，仕様が公開されている REST インタフェースを実装した Hue ブリッジを経由してアクセスでき，Hue 対応ノードはこれに基づく REST クライアントとして実装している．

図 5　Philips Hue と Hue ブリッジ

2.4.3 赤外線学習リモコン

赤外線学習リモコンは，予め家電の操作項目ごとに赤外線信号を学習しておき，操作内容に対応する信号を送信することで, 様々な家電を操作することのできる装置である．本研究では，ネットワーク対応していないレガシー家電の制御のために使用した．

赤外線学習リモコンノードは，事前に対象家電の操作信号の対応表をもっておき，中間層からの制御メッセージに従って，対応する信号を送信することで家電を操作する．表 2 に使用した赤外線学習リモコンの種類と図 6 に外観を示す．

表 2　赤外線学習リモコンの詳細

	名称	型番	メーカー
赤外線学習リモコン	eRemote	RJ-3	リンクジャパン

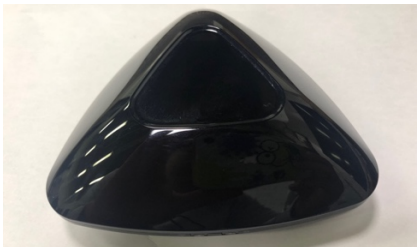

図 6 赤外線学習リモコンの外観

3.　レガシー家電への対応

家庭内にはネットワーク対応していない家電も多い．また，ネットワーク対応の家電でもリモコンのように操作だけ受け付けて，消費電力などの情報取得には対応していない家電や，逆に状態取得だけでネットワーク操作ができない家電なども存在する．本研究では，このような家電をレガシー家電とよび，仮想的にスマート家電のように見せかけることでシステムに対応させる．

2.2.2 で述べたように，提案システムでは，家電モデルに対応する下位層のノードとの関係は 1 対 1 に限らず，規格も異なる複数デバイスと 1 個の家電モデルを対応付けることができる．この機能を利用して，家電の状態取得はスマートタップから，家電の制御は学習型リモコンから行うことで，スマート家電のように扱うことができる．

3.1　消費電力値データの取得

消費電力値データの取得にはまずスマート家電に既に備わっている計測機能を使い，その機能がスマート家電に備わっていなければスマートタップかスマート分電盤を使用する．表 3，図 7 に実験で使用したスマートタップとスマート分電盤を示す．スマートタップはコンセントとコンセントプラグの間に差し込んで使用することで消費電力値を取得することができる．スマート分電盤は分電盤の分岐回路の単位の電力を取得できる．

本研究では，エアコンなどの消費電力が大きく，個別配線されている機器はスマート分電盤から，消費電力が小さい機器はスマートタップから電力を取得した．

表 3 使用したスマートタップとスマート分電盤

	名称	型番	メーカー
スマートタップ	WeMo insight switch	F7C029fc	Belkin
スマート分電盤	住宅用分電盤EN3Y	5160-33F	河村電器産業

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

図 7　スマートタップとスマート分電盤

3.2　動作状態の取得

アプリケーションによっては消費電力だけでなく，動作状態も必要な場合が多いが，スマートタップやスマート分電盤では消費電力しか取得することができない．そこで，本研究では，あらかじめ動作状態に対応する消費電力を調べておき，得られた消費電力値から動作状態を判定することにした．消費電力から動作状態を決めるのは機器によっては難しいが，最低限 ON 状態か OFF 状態かは簡単なしきい値によって決定できる．

3.3 家電の制御

レガシー家電の制御には，主に赤外線学習リモコンを用いた．現在家庭用で赤外線リモコンに対応する機器は多い．

さらに，赤外線リモコンにも対応しない機器に関しては，スマートタップのスイッチ機能をつかって，ON，OFF 制御のみ対応させた．現在照明向けのスマートスイッチやコンセント型スマートタップなど様々な製品が販売されており，学習型リモコンとスマートタップで家庭内のほとんどの機器を制御可能となる．最後に赤外線リモコンにも対応せずコンセントから直接電源入りきりできない機器については，制御不能機器として消費電力と動作状態のモニタリングだけ行った．

4. HEMS アプリケーションの実装

提案システムをつかって，HEMS としてエネルギーマネジメントを行うアプリケーションを作成した．HEMS アプリケーション機能は，上位層のノードの一種として実装した．

HEMS 機能では，中間層へ問い合わせメッセージを送信し，家電モデルに問い合わせることで各家電の消費電力を取得する．また，家全体の消費電力状態に応じて中間層を経由して家電を制御するメッセージを送信することで，マネージメントを実現する(図 8).

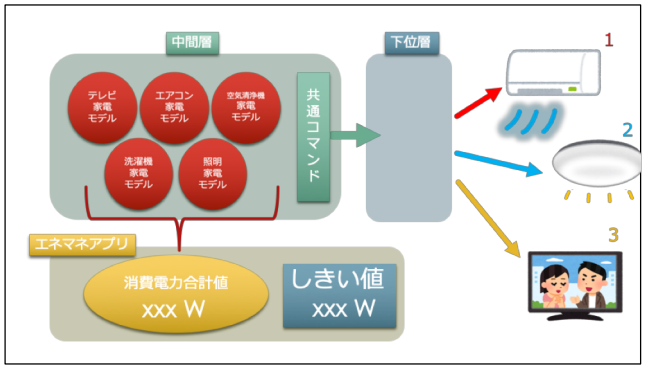

図 8　エネルギーマネジメントのフロー

本研究では，事前設定した家電優先度と家全体の消費電力上限値にしたがって家電を制御する簡易的な HEMS 機能を実装した．以下にその内容を示す．

(1) **消費電力上限値の設定**

2.3.3 で述べた独自インタフェースを使って，家全体の総消費電力値の上限値を設定する．

(2) **家電の優先度の設定**

同様に独自インタフェースを使って，制御される家電優先度を設定する．

これらの設定を行った上で，通常通り家電を使用する．この際，HomeKit や Amazon Echo から家電を操作しても，直接家電を操作しても良い．

(3) **消費電力合計値としきい値の比較**

家電を 1 つ制御したところでその瞬間の総消費電力値を取得し，設定したしきい値を上回っていたら(4)に進む，しきい値を下回っていたらそこでフローを終了させる．

(4) **家電の出力制御**

優先度の高い家電から出力を抑えるコマンドを出すように中間層にメッセージを送り，(3)に戻る．

ここで実装した HEMS 機能は提案システムの動作を確認するための非常に単純なものであるが，今後 EnergyOnDemand や，人物行動推定に基づく高度は HEMS 機能も実装していく．

5. Node-RED[12]をつかった実装と実験結果

ここまではシステムのデザインについて述べてきた．ここからは実際に Node-RED 上に提案システムを実装した内容と，フローの動作状態について述べる．

5.1 Node-RED による 3 層フローの実装

Node-RED は，図 9～図 12 に示すように処理を記述するノードと，ノードとノードをつなぐリンクによってビジュアルにプログラムを開発する JavaScript ベースの開発環境である．Node-RED のリンクはメッセージフローを表しており，各ノードは，リンクで表現されるメッセージを入力として処理をおこない，処理結果を新たなメッセージとして別のノードに送信する．外部からのネットワークを経由したメッセージの受信や他のノードからのメッセージ受信，タイマーなどをトリガーとしてノードの処理が非同期に実行されるため，今回のようなフローベースの非同期処理に向いており，また Node.js の非同期通信を利用して実装されているため高い非同期処理性能を引き継いでいる．

5.1.1　３層レイヤーの実装

フローの全体図をレイヤごとに，上位層を図 9，中間層を図 10，下位層を図 11 にそれぞれ示す．

図 9 に示す上位層では，おおきく HomeKit 対応ノードと Alexa 対応ノードを実装している．また図からは省略

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

しているが，Node-RED ダッシュボードを利用した独自インタフェースもこの層に実装している．

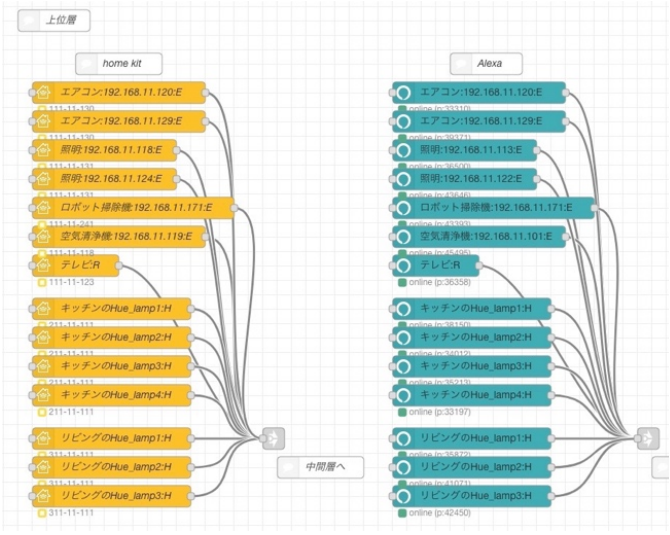

図 9　上位層の実装

図 10 に示す中間層では，家電モデルの管理をおこなっており，共通メッセージでうけとった操作指示のメッセージの下位層への送信や，下位層からうけとったメッセージにもとづく各家電の状態を更新をおこなっている．フロー図には記述していないが，家電状態はコンテキストと呼ばれるノード間で共有できるメモリ領域に保存している．

図 10　中間層の実装

図 11 に示す下位層では，ECHONET lite, Hue，赤外線学習リモコンに対応するノードを実装している．

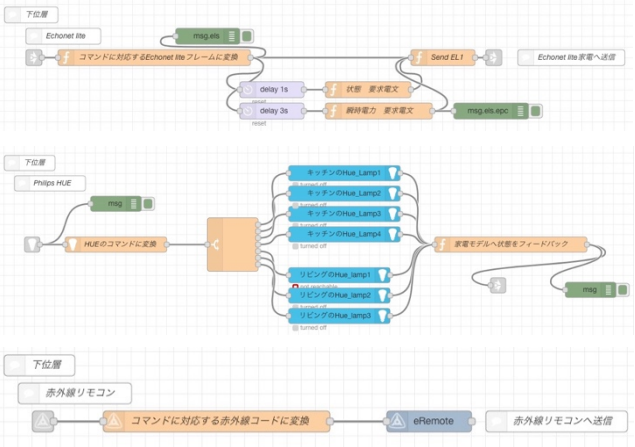

図 11 下位層の実装（上から，：ECHONET Lite, Hue，赤外線学習リモコン）

5.1.2 家電モデルインスタンスの作成

Node-RED 上での家電モデルは，システム起動時にスマート家電へネットワーク接続の有無を確認するコマンドを送り，返答があった家電に対してそのインスタンスを生成した．その際，個別の家電を区別するための ID を生成する．ECHONET Lite の場合は，IP アドレスと家電の種別を表すグループ ID, クラス ID とインスタンス ID の組み合わせを ID とした．またスマートタップやスマート分電盤で状態管理を行っているレガシー家電は，これらのデバイスからの返答に対してインスタンスを生成し，家電の種別は事前に対応表を作成しておくこととした．

次に順次，状態や消費電力値を要求するコマンドを送り(図 12)，その応答データを家電モデルへ格納した．

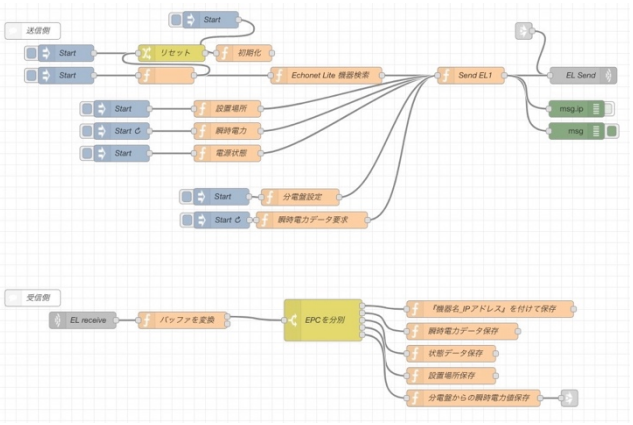

図 12　ECHONET Lite 用の家電モデルインスタンス作成の実装

このフローを経てコンテキストに保存された家電モデルの一部を図 13 に示す．

図 13 作成した家電モデルのインスタンス

5.1.3 HEMS 機能の実装

HEMS 機能の実装を図 16 に示す．エネルギーマネジメントのパラメータである総消費電力上限と家電の優先度は，2.3.3 に示した独自インタフェースから設定し，コンテキストに保存したものを取得した．定期的に中間層のコンテキストに保存されている家電モデルの各インスタンスを参照し，総消費電力を算出し，上限値を超過していれば，家電の優先順位に従って出力を抑えるためのコマンドを灰色の矢印から中間層へ送信する．

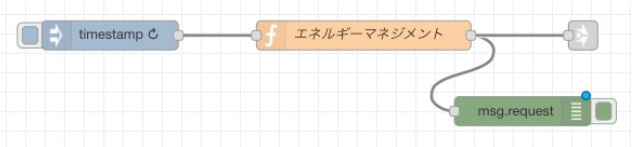

図 14 HEMS の実装

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

5.2 実験結果

5.2.1 既存のインタフェースアプリケーションとの連携のメッセージ

上位層を HomeKit として ECHONET Lite 対応エアコンの電源を ON にした時の上位層，中間層，下位層におけるメッセージを図 15〜図 17 にそれぞれ示す．さらにエアコンの状態を変化させた時，エアコンの電力計測機能からの電力値データ取得のタイミングを各メッセージのログに基づいてタイムライン上に可視化した結果を表 4 に示す．

図 15 HomeKit からのメッセージ

図 16 中間層の共通メッセージ

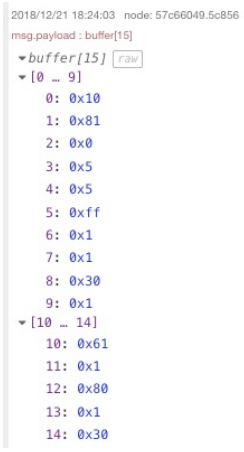

図 17 下位層からの ECHONET Lite 用メッセージ

表 4 電力値データの取得タイミング

時間 [s]	UI	家電モデル	下位層
0	照明ON	照明ON	照明 ON
1			
2			
3		消費電力 45W	消費電力 45W
4	照明 OFF	照明OFF	照明 OFF
5			
6			
7		消費電力 2W	消費電力 2W

5.2.2 デバイスと家電が 1 対 1 でない場合のメッセージ

家電操作と電力データ取得が別々のデバイスで行われる例として，分電盤に接続したエアコンの状態を変化させた時と電力値データを非同期的に取得した時のメッセージのタイムラインを表 5 に示す．

表 5 非同期的な電力値データ取得

時間[s]	UI	家電モデル	下位層	分電盤応答データ
0	エアコンON	エアコンON	エアコンON	0W
4		0W		91W
8		91W		218W
12		218W		225W
16		225W		230W
20		230W		228W
24	エアコンOFF	228W	エアコンOFF	229W
28		エアコンOFF		119W
32		229W		22W
36		119W		16W
40		22W		11W
44		16W		7W
		11W		
		7W		

5.2.3 HEMS アプリケーション実行時のメッセージ

HEMS フローを用いてエネルギーマネジメントを行った．例として 2 つの照明（リビング照明とキッチン照明）と空気清浄機を用意した．キッチン照明と空気清浄機の出力を

Vol.2019-GN-106 No.3
Vol.2019-CDS-24 No.3
Vol.2019-DCC-21 No.3
2019/1/24

最大にし，リビング照明を OFF にした時の総消費電力値 98W から電力上限値を 110W と設定した．この後リビング照明の出力を最大にして電力上限値を超えた時に行われたエネルギーマネジメントによる家電コントロールのタイミングとそれによって変化する総消費電力値の変化をまとめた（表 6，図 18）．

表 6 エネルギーマネジメントのタイムライン

時間[s]	UI	エネマネ	空気清浄機	キッチン照明	リビング照明
0	リビング照明ON				ON
2		合計143W			45W
4					
6					
8					
10		キッチン照明50%		明るさ50%	
12					
14		合計123W		21W	
16					
18					
20		空気清浄機30%	風量30%		
22					
24		合計94W	28W		
26					
28					
30		完了			

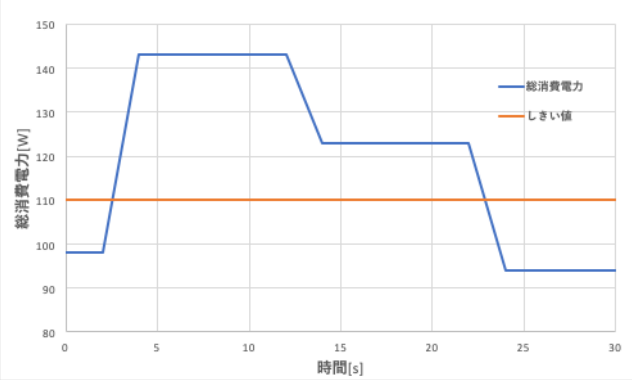

図 18 エネルギーマネジメントによる総消費電力の変化

6. まとめ

本研究では，非同期のメッセージ通信に基づく，HEMS のためのフローベースアーキテクチャを提案し，Node-RED によって実装した結果を示した．アプリケーション，ユーザインタフェースの上位層，家電モデルを管理する中間層，家電やセンサなどのデバイスとの通信を行う下位層の 3 層構造によって複数のスマート家電アプリケーションや，規格の異なるスマート家電との間で相互に連携する様子を示した．また簡易的な HEMS 機能を実装し，そのメッセージフローを示した．

今後の課題としては，レガシー家電に対するセンサデバイスと家電モデルの対応付はプログラムの構造に埋め込まれており，この設定を変えるにはプログラムそのものを変更する必要があり，これを設定ファイルや UI による設定など簡単に設定する方法が必要である．また，HEMS 機能は上限値と固定優先度による簡易的なものとなっており，より高機能な HEMS 機能の実装を行っていく予定である．最後に，現在のフレームワークでは家電単位の管理を家全体でフラットにおこなっており，部屋ごとの管理や，ユーザごとの設定など異なる単位の管理方法についても検討する予定である．

謝辞

本研究は JSPS 科研費基盤研究(B) JP17H01922 の助成を受けたものである．

参考文献

[1]Apple Inc., HomeKit, https://developer.apple.com/homekit/
[2]Amazon.com Inc., Amazon Echo Series, https://developer.amazon.com/echo
[3]Google LLC, Google Home, https://store.google.com/product/google_home
[4]Amazom.com Inc.., Amazon Skills Kit, available form https://developer.amazon.com/alexa-skills-kit
[5]住環境計画研究所:平成 17 年度一般家庭における HEMS 導入実証試験による省エネルギー効果の評価解析成果報 告書，技術報告，新エネルギー・産業技術総合開発機構 (2006).
[6]加藤丈和・湯浅 健史・松山隆司「オンデマンド型電力制御システム」串田高幸（編）『情報処理学会論文誌』(情報処理学会）第 J94-B 巻 10 号 1232-1245 頁.
[7] ECHONET コンソーシアム:ECHONET Lite 規格，(オンライン)，https://ECHONET.jp/product/ECHONET-lite/
[8]ZigBee Alliance: ZigBee RF4CE Overview (online), available from ⟨http://www.zigbee.org/Specifications/ZigBeeRF4CE/Overview.aspx⟩ (accessed 2012-05-29).
[9] ZigBee Alliance: ZigBee Smart Energy Profile (online), available from ⟨http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Overview.aspx⟩ (accessed 2012-05- 29).
[10]Philips Hue, https://developers.meethue.com.
[11] Belkin, Wemo SDK, http://developers.belkin.com/wemo/sdk
[12] Node-RED, https://nodered.org
[13] OSGi Alliance: OSGi Alliance Home Page (online), available from ⟨http://www.osgi.org/Main/HomePage⟩ (accessed 2012-05-08).
[14] Node-RED dashboard, https://flows.nodered.org/node/node-red-dashboard