Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

WoT アーキテクチャにおけるサービス連携のための一貫性制御機能の実装

水野翔輝† 平尾龍也† 清水宏康† 藤井章博†

概要：近年,農業分野やスマートホームなどの様々な分野で IoT(Internet of Things)が導入されつつある.例えば,様々なデバイスやシステムを,ネットワークを介して連携させることで,スマートホーム分野などでのオートメションやコンテキストアウェアなサービスの提供などが期待されている.しかし,現在の IoT 向けサービスは,専用のアプリケーションを利用して構築されるのが 一般的である.そのため,異なるシステム間での相互運用を行わせるのには,データフォーマットやプロトコルの統一化,トランザクション制御など様々な問題 がある.本研究では,これらの問題に対して,Semantic Web や RESTful API などの Web 技術を IoT に適用する WoT (Web of Things) アーキテクチャにより,異なるシステム間でのデータ表現の標準化と,トランザクションの一貫性制御等の同期機構を実装した.

キーワード：Internet of things, REST アークテクチャ, セマンティック Web, Web of Things

Distributed consensus platform service integration over WoT Architecture

Shoki Mizuno †, Tatsuya Hirao †, Hiroyasu Shimizu †, Akihiro Fujii †

Abstract: In recent years, IoT (Internet of Things) is being introduced in various fields. For example, in smart home and agriculture field, IoT provide automation and context aware service that provides service appropriate for user's situation. However many of these systems use a dedicated application that almost incompatible with other systems and variable devices, it is so difficult to integration with each other and extensions. In this paper, we propose a method to construct a platform that enables various devices to collaborate and cooperate with each other, using WoT(Web of Things) architecture that apply Web technology such as Semantic Web and RESTful API to IoT.

Keywords: Internet of Things, REST architecture, Semantic Web, Web of Things

１．はじめに

Internet of Things(IoT)とは,様々な Things(モノ)がインターネットにつながり,それらが互いに通信を行うことによりサービスを実現する環境を指す言葉である.近年,インターネット網の充実やコンピュータの演算処理能力の充実により,IoT を謳う種々の製品が発売されている.たとえばスマートホームを意識して,ネットワーク接続・制御を可能とする家電製品が販売されている.また,農業分野でも,センサデバイスから取得できる気温・湿度・土壌の状態をネットワーク越しに管理・閲覧できるシステムが導入されている.このように,様々な分野で IoT が広がりを見せつつある.しかし,これらのシステムは専用のアプリケーションにおいてのみの動作を前提とするものや,他社のシステムやデバイスとの互換性がないものも多い.こうした現状から,「システムの孤立化(サイロ化)」が重要な課題の一つとなっている[1].

IoT の「サイロ化」問題に対し,W3C は Web of Things (WoT)を提唱し問題の解決をはかっている.Web of Things とは,Internet of Things のデバイスやシステム間の通信を,Web 技術を用いて実現することである.WoT では,デバイス間やシステム間における通信を HTTP などの Web プロトコルを用いて通信を行う.また,各自のデバイスやシステムが保持する機能を,REST 形式の API などで公開する.WoT において,IoT 機器は HTTP を用いて REST という統一インターフェースにアクセスを行うため,他種多様な IoT デバイス同士が統一に通信することが可能である.これにより,サイロ化の問題を解決することが可能となる.一方で,IoT に求められる要素のひとつとして,IoT 機器同士が協調し動作することが挙げられる.複数の IoT 機器が協調して動作するには,IoT 機器同士が,互いの情報を同期して保持することが必要である.

WoT は,IoT 機器が通信を行う際の統一インターフェースを提供するが,同期を行う機構についての言及はされていない.現在では,管理サーバをクラウド上などに設置

† 法政大学大学院理工学研究科応用情報工学専攻
† Graduate school of science and engineering, Hosei University

Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

することによりこれを実現している.一方,中央集中のシステムでは中央の管理サーバにかかる負荷や,ネットワーク帯域の混雑といったことが問題となることが懸念されており,その対策として,システム末端に配置されるIoT 機器による分散自律処理が求められている.ここでいう分散自立処理とは,IoT 機器が環境から得られた情報を基に,IoT 機器自身が状況を判断し動作する機構を指す.

本研究の目的は,W3C の提案する WoT アーキテクチャにおいて,分散自律処理を実現するためのシステムの構築を行うことである.本研究では,分散自律処理を実現するためのアプローチとして,IoT 機器間における同期処理を,REST 形式の Web API として提供することを提案する.本研究における提案システムの機能構成は２項目で成り立っている.1 つ目は,異なる種類のデバイスが,互いが持つデータを共通に認識するための機能である.これは,W3C では Thing Description と表現されている機能である.本研究では,Web に意味を付与する試みである Linked Data (RDF) を用いてこれを実現する.２つ目は,異なる種類のデバイス間において,デバイスの動作状況といった情報を分散同期するための機構である.本論文では IoT 機器間での同期を実現するために,IoT 機器に実装する同期モジュール機構の実装と,IoT 機器間のメッセージ仲介を行う中央サーバの実装を行う.また,提案手法を用いたプロトタイプシステムの実装を行い,提案するシステムが一貫性を保った同期処理が可能であることを示す.

２. W3C の Web of Things アーキテクチャ

W3C は Internet of Things の通信を統一する規格として,Web of Things(WoT)を提案している W3C の提案する WoT アーキテクチャでは,Web で用いられているプロトコルやアーキテクチャを IoT に適用することにより,IoT のシステム間の互換性を確保し,アプリケーション設計がより簡単になることが期待されている.

W3C の WoT アーキテクチャにおける IoT 機器は,主として WoT Server 機能(デバイスのコントロール)と WoT Client 機能(WoT Server へのアクセス)に分類することが可能である.これは,Web におけるサーバ・クライアントの通信モデルと同様である.WoT Server 機能は,IoT 機器の操作を,Web API として提供する機能である.例えば,電灯の ON/OFF といった機能や,気温の取得といった機能がこれに当たる.WoT Client 機能は,WoT Server が Web API として提供している機能や情報を利用するための,Web クライアント機能である.また,Server 機能と Client 機能を有する仮想デバイスを,WoT Servient と呼ぶ.WoT Servient は,Legacy な IoT 機器とのプロトコル変換といった機能や,スマートホームであればインターネットに対するゲートウェイとして機能する.また,WoT Servient は,IoT におけるデバイス間の協調動作の際の,コミュニケーションの担当やスイッチの ON/OFF といった操作を決定するエージェントとしての役割を有する.例えば,図の例[1]では,WoT Servient1 が温度センサであり,Servient2 の空調を動作させるとする.servient1 は自身の温度センサによって取得された温度値が閾値を超えた場合,Servient2 の WoT Server に対し温度を変化させる要求を,Servient1 の Web Client 機能を用いて実行する.

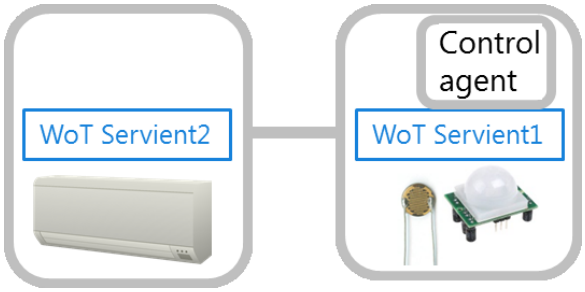

図 1. Thing-toThing Control (W3C WoT)

一方で,W3C の WoT アーキテクチャでは,デバイス間通信におけるメッセージの順序といった事象については触れられていない.一貫性が必要になる処理とは,動作の順序に意味が伴う場合である.本研究では,W3C の WoT アーキテクチャにおいて,WoT Servient に対して一貫性を保った同期機能を,REST 形式の API として実装することを行った.

３. セマンティック Web を用いたデバイス情報の定義

IoT 機器によって得られた情報を利用する際には,得られたデータを構造化し格納する必要がある.一方で,システムごとにフォーマットやラベルを独自に定義し,情報を表現した場合にはシステム間の連携が難しく「システムの孤立化(サイロ化)」に繋がってしまう.そのため,本研究ではデータ表現の標準化に対する試みとして,セマンティック Web を用いて,構造化データである RDF として定義を行った.セマンティック Web は,データ表現に用いられるオントロジーが同一であれば,フォーマットが異なるデータであっても統一的に解釈できるという特徴がある.これにより,「サイロ化」に陥ることがない,各種デバイスに関する情報のデータ表現の統一化を試みた.

本研究では,RDF において用いられる語彙として,W3C の SSN(Semantic Sensor Network)オントロジー[2]や M3 オントロジー[3]など,IoT 分野での利用を目的とした主要なオントロジーを用いて,デバイス情報の表現を行った.RDF での表現に用いたオントロジーを表１に,語彙やリソースの意味を表２に示す.

表１. デバイス情報の RDF 表現に用いたオントロジー

オントロジー	接頭辞	ネームスペース
RDF	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
RDF Schema	rdfs	http://www.w3.org/2000/01/rdf-schema#
SSN ontology	ssn	http://purl.oclc.org/NET/ssnx/ssn#
M3	m3	http://sensormeasurement.appspot.com/m3#
schema.org	shema	http://schema.org/
Basic Geo (WGS84 lat/long)	geo	http://www.w3.org/2003/01/geo/wgs84_pos#

Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

表２．デバイス情報の RDF 表現に用いた語彙

URI	接頭辞表記	意味
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	rdf:type	リソースタイプ
http://www.w3.org/1999/02/22-rdf-syntax-ns#label	rdf:label	リソースラベル
http://www.w3.org/2003/01/geo/wgs84_pos#lat	geo:lat	緯度
http://www.w3.org/2003/01/geo/wgs84_pos#long	geo:long	経度
http://schema.org/location	schema:location	場所
http://schema.org/Place	schema:Place	場所
http://purl.oclc.org/NET/ssnx/ssn#observes	ssn:observes	観測事象
http://purl.oclc.org/NET/ssnx/ssn#FeatureOfInterest	ssn:FeatureOfInterest	特徴
http://purl.oclc.org/NET/ssnx/ssn#observedProperty	ssn:observedProperty	観測事象の特徴
http://sensormeasurement.appspot.com/m3#produces	m3:produces	生成データ
http://sensormeasurement.appspot.com/m3#hasUnit	m3:hasUnit	単位
http://sensormeasurement.appspot.com/m3#hasDateTime	m3:hasDateTime	日時
http://sensormeasurement.appspot.com/m3#hasValue	m3:hasValue	値
http://sensormeasurement.appspot.com/m3#hasName	m3:hasName	名前
http://sensormeasurement.appspot.com/m3#AirThermometer	m3:AirThermometer	空気温度計
http://sensormeasurement.appspot.com/m3#AirThermometer	m3:SmartHome	スマートホーム

表１，表２に示したオントロジーと語彙を用いて，デバイス情報を表現するリソースを定義した．デバイス情報は，①デバイスの名称や設置場所を表現する「デバイスに関する情報」，②センサの観測事象やアクチュエータの機能を表現する「機能に関する情報」，③データの単位や観測時刻を表現する「データに関する情報」，④それら３つのリソースの関連を表現する「出力に関する情報」の４つにリソースに分けて定義した．それぞれのリソースの URI と意味を表３に示す．また，それらのリソース表現を用いて，デバイス情報を RDF で表現したものを図２に示す．

表３．デバイス情報を表現するリソース

URI	接頭辞表記	意味
http://localhost:8080/device/resource/device{デバイスID}}	device:device{デバイスID}	「デバイスに関する情報」を表現するリソース
http://localhost:8080/device/resource/device{デバイスID}}_observation	device:device{デバイスID}_observation	「機能に関する情報」を表現するリソース
http://localhost:8080/device/resource/device{デバイスID}}_measurement{データID}	device:device{デバイスID}_measurement{データID}	「データに関する情報」を表現するリソース
http://localhost:8080/device/resource/device{デバイスID}}_output{出力ID}	device:device{デバイスID}_output{出力ID}	「出力に関する情報」を表現するリソース

上記の表３ようにデバイス情報を複数のリソースを分けて定義することで，より複雑なデバイスの表現や詳細の記述が可能となる．また，表１～表３のオントロジーや①～④として定義した４つのリソースを用いて，デバイス情報を RDF で表現したものを図２に示す．

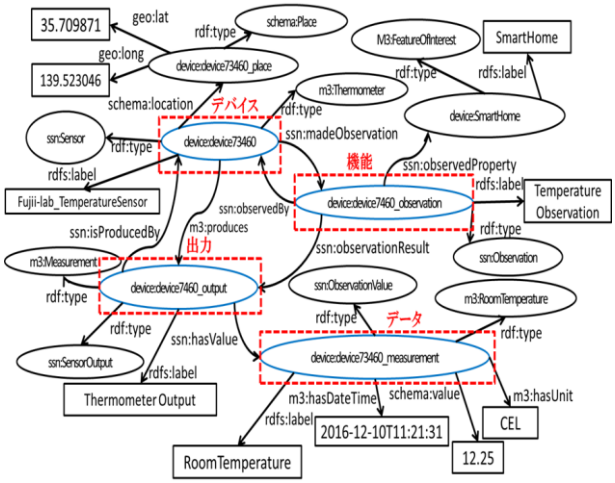

図 2．デバイス情報を表現した RDF

図２では，主に「デバイス」，デバイスの「機能」，「出力」，「データ」に関する情報を別々のリソースとして定義し，それらを関連付けることでデバイスを表現している．「デバイス」に関する情報とは，デバイス設置場所の緯度・経度などの地理情報や，デバイス名などである．また「機能」では，観測対象事象などの情報を定義しており，「データ」では，単位，値，観測時刻などの情報を定義している．さらに「出力」では「デバイス」，「機能」，「データ」との関係を定義することで，”どのデバイスのどの機能から出力されたデータか”を表現している．また，これらのリソースの関係を全て関連付けることでデバイス情報を表現している．また，図２のモデルに対して推論を行うことで「device:73460」という URI で示されるセンサは，大気温度という事象を観測するデバイスであり，デバイス名は Fujii-lab_TemperatureSensor であることがわかる．本研究では，各デバイス間でデバイスの種類や名前，センサデバイスが取得したデータなどのデバイス情報の共有に上述の RDF を用いた．

４． REST アーキテクチャによる同期機構

システムを構成する各デバイス上のプロセスで事象（イベント）が発生し，これをシステム全体で共通の順序関係で事象を認識させる方法について述べる．本研究では，RESTful API を用いて事象認識の全順序関係を満たすための機構を Web リソースとして提供する．利用する同期アルゴリズムについては藤井らのアルゴリズム[12]を利用した．デバイスは，提案システムにおいて RESTful API にアクセスすることにより，全順序関係を満たした情報の同期を行う．

はじめに，提案システムが提供する RESTful API のリソースについての説明を行う．提案システムの RESTful API が提供する URI リソースを表４に示す．

Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

表４．同期 API が提供するリソース

/event/{event_id}/{myid}/{value}	POST	イベントの発生を通知
/receive	GET	イベントの発生状況を取得
/receive/{event_id}	GET	イベントの内容を取得
/ok./{event_id}/{myid}	GET	イベントの認知状況を取得
/ok./{event_id}/{myid}	POST	イベントの認知を通知
/event	DELETE	イベントの削除

表４の URI 中の”中かっこ”は変数を表す,それらの変数の意味を表５に示す.

表５．同期 API の URI 変数

変数	説明
myid	各プロセスに割り当てられている一意な番号
event_id	事象を認識するための番号。現在時刻を用いる。
value	イベントの内容

デバイス上の各プロセスは,表４に示した API が提供するリソースを利用して,以下の(1)～(6)のいずれかの処理を行う.

(1) イベントの発生を通知

イベントの発生を通知とは,自身のデバイス上のプロセスの状態の変化や,データ更新などが発生した場合に,他の全てのプロセスに対して,その事象(イベント)の発生と内容の通知を行うことを指す.事象の発生通知を行うには,/event/{event_id}/{myid}/{value}という URI で示されるリソースに対してHTTP POSTを行う.このとき,レスポンスとして”Failure”または”Success”が返される.”Failure”は,既に他のプロセスの事象が発生している状態であり,排他的制御により事象の通知が失敗したことを表す.そのため,複数の事象の通知が並列に進行することはない.また,”Success”は事象の通知が成功したことを表す.

(2) イベントの発生状況の取得

イベントの発生状況の取得とは,システム全体で事象が発生しているプロセスの存在の有無,および存在する場合はその事象を一意に識別する ID を取得するものである.事象の発生状況を取得するには”/receive”という URI で示されるリソースに対して HTTP GET を行う.このとき,レスポンスとして”NONE”または”{event_id}”が返される.”NONE”は現在発生中の事象は存在しないことを表す.”{event_id}”は現在発生中の事象の ID を表し,(1)の”事象の発生通知”を行った際に用いた変数である.また,(1)で示したように複数の事象が並列に発生することはない.そのため,複数の”{event_id}”がレスポンスとして返されることはない.

(3) イベントの内容の取得

イベントの内容の取得とは,現在発生している事象の内容を取得するものである.事象の内容を取得するには”/receive/{event_id}”という URI で示されるリソースに対して HTTP GET を行う.このとき,レスポンスとして”Failuer”または”{value}”が返される.ここで返される{value}は,(1)の事象の発生通知を行った際に用いた変数である.また,事象の内容の取得に用いる URI に含む変数{event_id}は,(2)で返されるレスポンスの内容である.

(4) イベントの認知を通知

イベントの認知を通知とは,現在発生している事象の内容取得が完了し,それを他の全てのプロセスに対して通知を行うこと指す.事象の認知を通知するには”/ok/{event_id}/{my_id}”で示されるリソースに対して,HTTP POST を行う.このとき,レスポンスとして”Success”または”Failure”が返される.レスポンスの”Success”は,通知が完了したことを表す.また,”Failure”はリクエストで指定した{event_id}で示される事象が存在しないことを表す.

(5)イベントの認知状況を取得

イベントの認知状況の取得とは,(4)の”事象の認知を通知”を行ったプロセスの{myid}の一覧を取得するものである.事象の認知状況を取得するには”/ok/{event_id}/”という URI で示されるリソースに対して HTTP GET を行う.このとき,レスポンスとして”Failure”または１つ以上の”{my_id}”を返す.ここで返される”Failure”は,”{event_id}”で示される現在発生中の事象が存在しないことを表す.また,レスポンスに含まれる”{my_id}”は,URI で変数として指定した”{event_id}”で示される事象の発生を認知しているプロセスの ID を表す.

(6)イベントの削除

イベントの削除とは,(1)において”Success”が返された事象に対して,その事象の通知処理を終了するものである.例えば,プロセス全体で現在発生中の事象の認識が完了し,全てのデバイス上のプロセスで更新処理が完了した場合に実行される.事象の削除を行うには”/event”に対して HTTP DELETE を行う.このとき,(1)の”事象の発生通知”を実行し,事象を発生させたプロセスが”事象の削除”を行うものとする.

システムを構成する各サブシステム上の各プロセスは,自身の状態及び(1)～(6)の結果から,自身の状態を遷移させる.これを繰り返すことで各プロセスは共通の順序で事象を認識する.各デバイスの状態遷移を図 3 に示す.

Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

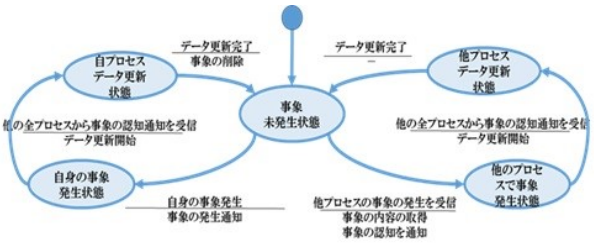

図 3. デバイスの状態遷移

各プロセスは図 3 のように,状態を遷移させることで全順序関係を満たした事象の認識を行う.また(1)で述べたようにシステム全体で複数の事象の通知処理が並列に進行することはなく,他の全てのプロセスが事象を認知するのを待って次の事象の通知処理に移る.これにより,全てのプロセスで全順序性を保証した事象の認識を行う.

5. 実装

本研究における提案手法を用いたシステムが,順序性を保った同期が可能であるかを検証するために,提案手法を用いたプラットフォームを構築した.また,動作の検証と実利用を想定し,ユーザからのデバイス情報の利用や,他の Web アプリケーションとの連携が行えるユーザインタフェースも併せて実装した.

デバイス端末は,小型コンピュータボード Raspberry pi を採用した.サーバは,OS に Ubuntu Linux,アプリケーションコンテナとして Tomcat を,RDF データベースに Virtuoso を用いて機能を実装した.図 4 にシステム構成図を示す.本システムは,各デバイスの持つデータを共有するための機能である「デバイス同期・制御機能」,ユーザや Web アプリケーションに対してデバイス情報を提供する「Web サービス機能」からなる.

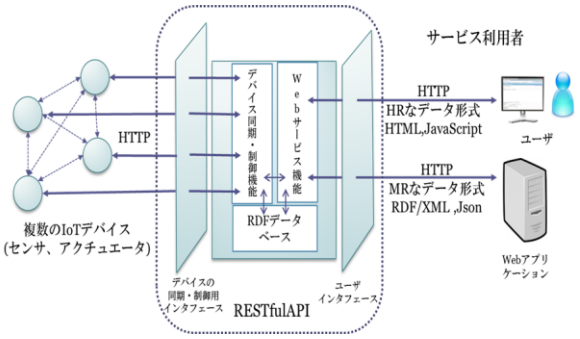

図 4. システムの構成図

(I)デバイスの同期・制御機能

デバイスの同期・制御機能は,各デバイス間の状態やデータの共有を行う機能を提供する.デバイスの状態や観測データは,RDF として各デバイスに格納され,本機能を用いて全てのデバイスで共有される.前述した REST アーキテクチャによる同期機構の実装である.

(II)Web サービス機能

Web サービス機能では,ユーザや Web アプリケーションに対して,各デバイスの設置場所や機能,デバイスが保持するデータなどの情報を提供する.各デバイスに関するデータは,ユーザと Web アプリケーションは,ユーザインタフェースに対して HTTP リクエストを送信することで,各種デバイス情報の参照やデータ取得が行える.表 6 にユーザインタフェースの URI と意味を示す.

表 6. ユーザインタフェースの URI

URI	説明
/device/resource	全デバイスに関するインデックス
/device/resource/device{デバイス ID}}	デバイスの設置場所、名前
/device/resource/device{デバイス ID}}_observation	デバイスの機能詳細
/device/resource/device/{デバイス ID}_output	観測データなどのインデックス
/device/resource/device{デバイス ID}}_measurement{データ番号}	値、単位、観測日時などのデータ詳細

ユーザインタフェースでは,ユーザに対しては Web ブラウザを通して人間が理解しやすい情報形式(HR:Human Readable)であるページとして情報提供する.またそれに対して,Web アプリケーションに対しては,機械が理解可能(MR:Machine Readable)な JSON-LD や RDF/XML,CSV 形式での情報提供を行う.MR/HR なデータ形式の判別は,HTTP の Accept ヘッダにて行う.

6. 実行結果

(a) デバイスのデータ同期

温度センサを実装した 3 台のデバイスにて,各デバイスの種類や観測データの同期を行わせた結果,あるデバイスの観測データの更新が発生した場合に,全てのデバイスで共通の順序でデータが更新されることを確認した.デバイスの動作管理画面を以下の図 5 に示す.

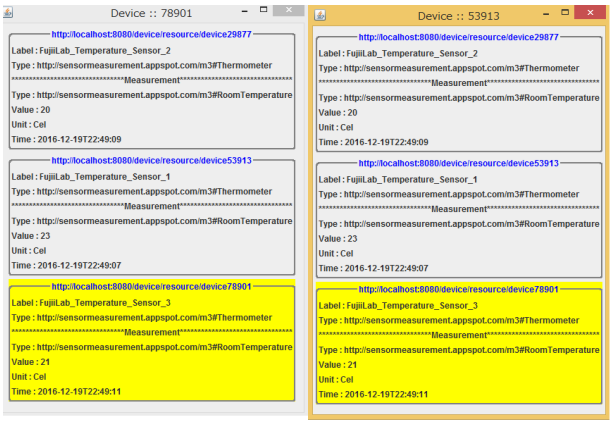

図 5. デバイスの動作管理画面

図内の各ウィンドウはそれぞれが接続されているデバイスが保持している情報を表示する.また,ウィンドウ内の各枠には,接続されている各デバイスの種類や,デバイスによって観測されたデータの内容を表示している.また,

Vol.2017-DPS-171 No.24
Vol.2017-MBL-83 No.24
Vol.2017-ITS-69 No.24
2017/6/2

色付きの枠はデバイス情報の更新があったことを示している.図 5 で確認できるように,各デバイスにおいて共通な順序で情報が更新されていることが確認できた.

(b) ユーザインタフェース

Web ブラウザを利用して,サーバ上のユーザインタフェースにリクエストを送信し,デバイスの種類や観測データなどの情報がページとしての表示されることを確認した.表示画面を図 6 に示す.デバイスの設置場所は,Google Maps を利用した地図上にマッピングされ,観測データのグラフとデバイスの種類などの情報はポップアップウィンドウ内に表示される.

図 6. ユーザインタフェース

7. 考察・結論

IoT の普及にともない,Web 技術を用いたデバイスやシステム間の相互接続を標準化する試みとして,Web of Things は提案されてきた.本研究では,WoT アーキテクチャにおいてデバイスが分散に存在し自律的に協調動作をする分散自律処理を実現するために,RESTful API として同期機能をモジュール化し提供する提案を行い,プロトタイプの実装を行った.

REST アーキテクチャによる同期機構については,HTTP や RESTful Web API などの代表的な Web 技術を用いているため,異種システムやデバイス間での相互接続性が高い.そのため,IoT における課題である「サイロ化」に陥ることなく,同期機能の提供が可能である.したがって,提案システムを用いることによりデバイス間の同期を用いたサービスの構築が可能であると考える.

構築したシステムのサービスの連携性については,セマンティック Web 技術を用いてデバイス情報を表現することで,他のシステムとのデータ表現の統一化を図る一例を示した.

一方で,本論文では提案システムの同期機構を用いた際に管理サーバや各デバイスにかかる負荷の測定といった,パフォーマンスについての検証には至っていない.提案システムの実利用に先立っては,システムを用いることによって生じる負荷の計測や,ネットワークの帯域の使用率といったパフォーマンスの計測と,最適化が必要である.サービスの連携性については,セマンティック Web を IoT 分野へ応用を行うプロジェクトは複数存在する,が,利用可能なサービスとして公開されているものは確認できなかった.そのため,本研究では他のシステムとのサービス連携についての実検証には至っていない.今後は,IoT 分野や W3C で策定中である WoT 仕様に関する動向の調査を行った上で,他のシステムとの連携性などの詳細な検証を行っていきたい.

また,本研究における提案システムを,ホームオートメーションや農業分野の圃場管理といった実環境に適用することにより,実利用の際の評価も必要であると考える.

参考文献

[1] W3C, “WEB OF THINGS ”, https://www.w3.org/WoT/, 2017/2/15 閲覧
[2] 藤井章博, 中村眞, 根本義章,” 連続同期を効率良く処理する分散型同期制御方式”,電子情報通信学会論文誌, Vol.J81-D-I, No.3, pp.244-252
[3] W3C,” OWL-Semantic Web Standards”, https://www.w3.org/OWL/,2017/2/15 閲覧
[4] W3C Semantic Sensor Network Incubator Group(SSN- XG), “Semantic Sensor Network Ontology”, http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[5] schema.org, schema.org, http://schema.org, 2017/2/15 閲覧
[6] Gyrard et al, Enrich machine-to-machine data with semantic web technologies for cross-domain applications, WF-IOT 2014
[7] 清水宏泰「Semantic REST による RDF データの活用」情報処理学会論文誌,第 55 巻,2 号,2014 年
[8] A.Fujii, H. Shimizu, RDFa Parser for Hybrid Resources ,Semantic Technology LNCS8943,Springer
[9] 江上周作,「BOM エージェントの実現に向けた LOD の構築」,電子情報通信学会論文誌 D Vol.J98 D/ pp.992-1004 (2015)
[10] Cisco Systems, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, 2017/2/15 閲覧
[11] 長野伸一, Linked Data とセンサネットワーク, 人工知能学会誌 27(2), 200-206, 2012-03-01
[12] Distributed Synchronization over RESTful Web API Daiki Murayama, Shouki Mizuno, Akihiro Fujii NBiS-2015 The 18-th International Conference on Network-Based Information Systems 2015/09/04