情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.10　No.1　40–49 (Feb. 2020)

コンシューマ・システム論文

環境情報に基づいて異種規格のIoTデバイス連携をサポートするiHAC Hub

林 宏輔[1,a)]　鈴木 秀和[1,b)]

受付日 2019年7月8日, 採録日 2019年11月13日

概要：通信プロトコルの違いを意識することなくスマート家電を直感的に制御することができるiHAC（intuitive Home Appliance Control）システムが提案されている．また，iHACシステムの応用例として，センサノードが計測した温度や湿度などの環境情報に基づいて機器連携を行うことが提案されているが，その実現方法は十分に検討されていない．本論文では，環境情報をリアルタイムに取得する機能をiHACフレームワークに追加し，ECAルールに基づいて異種通信規格のIoTデバイス連携を実現する宅内用デバイスiHAC Hubを提案する．iHAC Hubのプロトタイプを開発し，環境センサとECHONET Lite機器の連携動作検証および既存機器連携サービスとの性能評価実験を行った．その結果，提案システムは既存サービスよりも短時間で環境情報に基づくスマート家電機器の制御が可能であることを確認した．また，既存機器連携サービスと比較して，複数条件に基づいた機器連携を簡単かつ柔軟に実現できることを確認した．

キーワード：IoT，機器連携，ECHONET Lite

iHAC Hub Supporting Heterogeneous IoT Device Cooperation Based on Environment Information

Kosuke Hayashi[1,a)]　Hidekazu Suzuki[1,b)]

Received: July 8, 2019, Accepted: November 13, 2019

Abstract: An intuitive Home Appliances Control (iHAC) system has been proposed that can intuitively control smart home appliances without being aware of differences in communication protocols. Further, as an application example of the iHAC system, it has been proposed to perform device cooperation based on environmental information such as temperature and humidity measured by sensors. However, the realization method has not been sufficiently studied. In this paper, we propose an in-home device iHAC Hub that adds the function of acquiring environmental information in real time to the iHAC framework, and realizes IoT device coordination with different communication standards based on an ECA rule. We developed a prototype of the iHAC Hub, and conducted experiments on the verification of the coordinated operation of environmental sensors and ECHONET Lite devices and the performance evaluation of the existing device coordinated service. As a result, we confirmed that the proposed system can control smart home appliances based on environmental information in a shorter time than existing services. We also confirmed that device linkage based on multiple conditions can be implemented easily and flexibly compared to existing device linkage service.

Keywords: IoT, ECHONET Lite, device cooperation

1. はじめに

モノ，ヒト，サービス，情報などあらゆる“モノ”をインターネットなどのネットワークに接続することを意味する，IoT（Internet of Things）という言葉が浸透しつつある．

[1] 名城大学大学院理工学研究科
Graduate School of Science and Technology, Meijo University, Nagoya, Aichi 468–8502, Japan
[a)] kosuke.hayashi@ucl.meijo-u.ac.jp
[b)] hsuzuki@meijo-u.ac.jp

© 2020 Information Processing Society of Japan

その中でもインターネットに接続可能な機器である IoT デバイスの数は 2015 年時では 154 億個であり，2025 年までに約 754 億個まで規模が拡大すると予想されている [1]．また，IoT の普及により AI スピーカやスマートフォンをコアとして情報家電やスマートメータ，自動車などの家庭のモノがインターネットに接続されるスマートホームを利用した新たなサービスの提供に向けた取り組みも進んでいる [2]．それにともない，家庭内で複数のスマート家電どうしを連携して動作させたり，温度や湿度といった環境情報をセンシングする IoT デバイスを用いることにより，環境情報に基づいた機器制御を行うことが可能となると考えられる．

IoT デバイスが採用している代表的な通信プロトコルとして，スマート家電では DLNA（Digital Living Network Alliance）[3] や ECHONET Lite [4] など，センサデバイスでは ZigBee [5] や Z-Wave [6]，6LoWPAN（IPv6 Low-power Wireless Personal Area Network）[7]，Thread [8] などがある．DLNA は主にテレビや Blu-ray レコーダなどといった AV 機器に対応しており，ECHONET Lite はエアコンや照明機器などの白物家電を中心に対応した通信プロトコルである．また，ZigBee や Z-Wave などのセンサデバイス用通信プロトコルは温度センサや湿度センサなどに対応している．現在のホームネットワークはこれら異なる規格のスマート家電が混在する環境となっており，ユーザは制御したい IoT デバイスに合わせてアプリケーションや操作方法を変更しなければならない．

そこで梅山らは通信プロトコルの違いを意識することなく，スマート家電を直感的に制御することができる iHAC（intuitive Home Appliance Control）システムを提案している [9]．iHAC システムは，機器の登録や探索に関わる API が定義された iHAC フレームワークを導入し，この API がそれぞれの機器に対応した通信処理部の API を呼び出すことにより，プロトコルの違いを抽象化している．また，文献 [10] では iHAC システムを応用例として，IoT デバイスの動作条件や制御内容などをまとめたルールに基づいた機器連携を可能にしている．しかし，iHAC システムを導入した iPhone や iPad などの操作端末を常時起動させ，つねに環境情報をセンシングしなければならないことやルールの記述方法が十分に検討されていないという課題がある．

そこで本論文では，iHAC システムを拡張し，環境情報をリアルタイムに取得する機能を有する宅内用デバイス iHAC Hub を提案する．また，柔軟な機器連携ルールを記述するために，ECA ルール [11] に基づいたデータ構造を定義する．IoT デバイスがセンシングした環境情報の取得と機器制御を iHAC Hub が行うことにより，既存の iHAC システムの課題を解決し，環境情報に基づく異種規格の IoT デバイス連携を実現する．

以下，2 章で従来の iHAC システムと機器連携に関わる既存研究を示し，3 章で提案システムについて述べる．4 章で提案システムのプロトタイプ実装および動作検証について述べ，5 章でプロトタイプの評価を行い，6 章でまとめる．

2. 既存システム

2.1 iHAC システム

iHAC システムは，規格の違いを意識することなくスマート機器を直感的に制御できるシステムである．**図 1** に iHAC システムの構成を示す．iHAC システムは，操作端末とスマート家電から構成され，操作端末にインストールされる iHAC アプリケーションは，UI（User Interface）部，iHAC フレームワーク部および各規格の通信処理部から構成される．

UI 部は iHAC フレームワーク部から取得するスマート家電の情報や操作メニューを表示する．ユーザはスマート家電の規格はいっさい意識することなく，操作したい機器を選択することができる．

iHAC フレームワーク部はスマート家電の通信規格の違いを抽象化するレイヤであり，UI 部からの要求に従って各規格の通信プロトコルの通信処理部に命令を行う．発見した機器名，機器を一意に識別する識別子，部屋名などの位置情報，エアコンや照明器具などの機器の種類を示すデバイスタイプをデータベースに記録するとともに，異なる規格のスマート家電の情報を統合して UI 部へ提示する．

各規格の通信処理部は，iHAC フレームワーク部からの命令に従って，機器の探索や命令を行う．各規格の通信処理部は独立して実装することが可能で，ECHONET Lite や DLNA 以外の通信プロトコルに対応した処理を追加することにより，iHAC システムがサポートする対象機器を柔軟に拡張することが可能である．

iHAC システムの応用例として，特定の条件を満たした場合に自動的に機器を連携する手法が提案されている [10]．

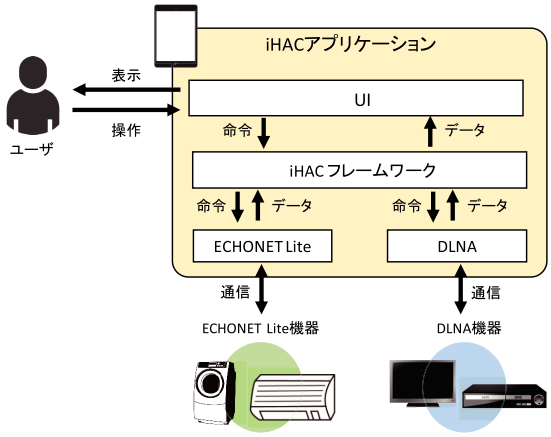

図 1　iHAC システムの構成
Fig. 1　Configuration of iHAC system.

© 2020 Information Processing Society of Japan

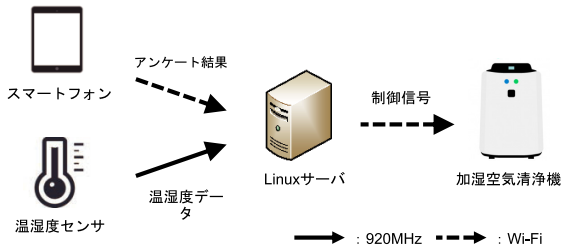

図 2　快適嗜好性を考慮した加湿空気清浄機の自動制御手法
Fig. 2　Automatic control method of humidifier and air cleaner considered with indoor comfort preference.

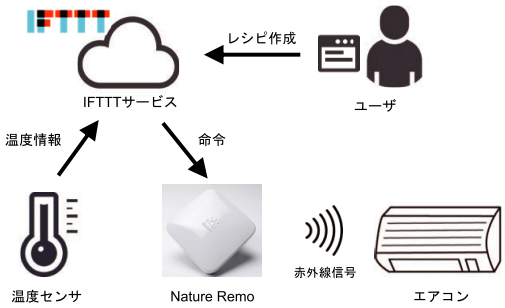

図 3　IFTTT を利用した機器連携の例
Fig. 3　Example of device cooperation using IFTTT.

ある状態をつねにモニタリングし，あらかじめ定めた条件を満たしたとき，機器制御を実行する．たとえば，「リビングの温度が 30°C 以上になったらエアコンの電源を ON にする」といった環境情報に基づいた機器連携が可能となる．

しかし，上記のような機器連携を実現する場合，iHAC アプリケーションを常時起動させ，温度や湿度をつねにセンシングし続けなければならない．iHAC アプリケーションは iOS 向けの実装しかなく，iPhone や iPad などの操作端末を常時起動するという用途は向いていないことや，動作条件や制御内容を示すルールの記述方法など，実現方法について十分に検討されていない．

2.2　室内温度の快適嗜好性を考慮した加湿空気清浄機の自動制御手法

環境情報に基づいた機器連携として，室内温度の快適嗜好性を考慮した加湿空気清浄機の自動制御手法が提案されている [12]．**図 2** に示すように，宅内に設置された温湿度センサは 1 分ごとに室内の温度と湿度を計測し，Linux サーバに転送する．ユーザはスマートフォンを用いて年齢，性別，温湿度の快適嗜好性に関する Web アンケートに回答し，その結果を Linux サーバにへ送信する．温湿度の快適嗜好性は，現在の室内環境をどのように感じたかを温度，湿度，快適性の 3 種類に対してそれぞれ 7 段階評価で回答してもらうことにより，暑がり，どちらでもない，寒がりの 3 種類に分類される．Linux サーバは受信したユーザの温湿度の快適嗜好性から温度や湿度の快適域を定義し，この快適域を満たすように温湿度センサによって収集された温度と湿度の値を元に加湿空気清浄機の制御を行う．

しかし，このシステムでは上記以外の IoT デバイスの連携は考慮されておらず，汎用性に欠けている．また，快適嗜好性によって定められる温度や湿度の快適域の値をユーザが調節することができないなど，機器連携設定の柔軟性に課題がある．

2.3　IFTTT

Web サービスどうしの連携サービスとして，IFTTT（If *This* Then *That*）[13] がある．このサービスを応用することにより，たとえば**図 3** のようにリモコンの赤外線信号を登録しスマートフォンなどから家電製品を制御することが可能なスマート家電制御装置 Nature Remo [14] と宅内に設置した温度センサを連携させることができる．「温度センサの値が 30°C 以上を検出した場合，Nature Remo を用いてエアコンを起動する」というレシピを作成することにより，環境情報に基づく機器連携が可能となる．

しかし，IFTTT では通常単一の条件や動作しか設定できない．そのため，温度と湿度のように複数の条件を満たした場合や，制御対象機器が複数台あるような場合は，Webhook を利用した独自のアプレットを作成しなければならず，一般のユーザが利用するのは困難である．また，宅内に設置するスマート家電制御装置が IFTTT 対応でなければならず，スマート家電制御装置が異種規格の IoT デバイズの制御に対応している事例は筆者らが調べた限り存在しない．

2.4　ホームネットワークにおけるデバイス間連携に基づく新サービス構築手法

文献 [15] では，iHAC システムと同様に規格の違いを吸収する統合ミドルウェアを搭載し，スマート家電，組み込みデバイス，Web サービスとの連携を可能とする手法を提案している．このシステムは，汎用インターフェース，機能処理レイヤー，統合ミドルウェア，各通信処理部により構成される（**図 4**）．汎用インターフェースは，ユーザが操作する UI に相当し，制御する機器の持つ機能を Drag&Drop で関連付けることにより，機器連携のルールを作成する．機能処理レイヤーは，制御する機器の機能の抽象度を上げる．たとえば，ECHONET Lite のエアコンクラスには，“冷房モード時温度設定値” や “暖房モード時温度設定値” などの機能があり，この機能をそのままユーザに表示すると利用が困難になるため，“設定温度の変更” のように機能を集約し，ユーザが利用しやすくする．統合ミドルウェアは，通信規格の違いを吸収する．これにより，ユーザは規格の違いを意識することなく，機器を操作することができる．

しかし，複数条件に基づく機器連携を実現するルールの

© 2020 Information Processing Society of Japan

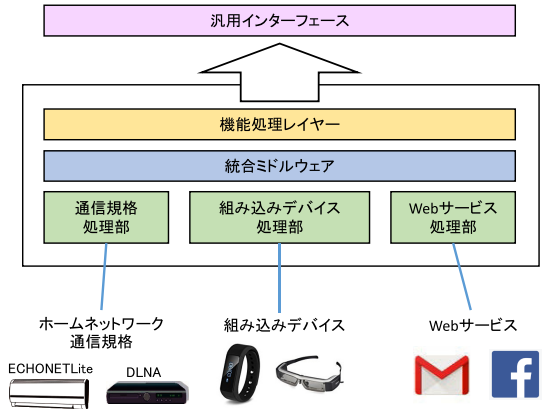

図 4　ホームネットワークにおけるデバイス間連携に基づく新サービス構築手法のシステム構成

Fig. 4　Configuration of the method of constructing new services by cooperation between devices in home network.

定義については検討されていない．

2.5　オントロジー駆動型アダプテーション層

文献 [16] では，ICT とスマートホーム技術を利用して高齢者の自立を支援し，生活の質を向上させる AAL（Ambient Assisted Living）プラットフォーム [17] と ECHONET Lite デバイスを相互運用を実現するために，オントロジー駆動型アダプテーション層が提案されている．ECHONET Lite オントロジーと AAL のオープンプラットフォームである universAAL のミドルウェアを搭載したサービスゲートウェイを仲介することにより，規格の異なる AAL 対応デバイスと ECHONET Lite デバイスを連携させることを実現している．

ただし，異なる規格のデバイスを連携させる際の動作条件やルールの定義については議論されていない．

2.6　IoT サービス連携アーキテクチャ

文献 [18] では，既存の IoT デバイス，サービス，アプリケーションが垂直統合で開発されており，今後は IoT サービス間の相互運用性が重要になることを述べている．そこでフォグコンピューティングの概念を導入し，IoT アプリケーションが複数の IoT サービスに効率良くアクセスできるアーキテクチャを提案している．ホームネットワークとインターネットの境界エリアにフォグサーバを導入し，IoT デバイスがセンシングしたデータなどを蓄積する種々のクラウドと IoT アプリケーションの間に介在することにより，IoT サービスの違いを抽象化している．

しかし，この提案アーキテクチャを用いて IoT デバイス連携を行う場合，IoT デバイスがフォグサーバに実装した専用の API をコールする必要があること，またフォグサーバを実現するにはホームネットワークにおけるブロードバンドルータに機能実装する必要があるため，一般のユーザが利用することは難しい．

また，文献 [19] においても，個別に開発されている IoT アプリケーションをシームレスに統合または相互作用に対処するオーケストレーションプラットフォームが存在しないことを指摘しており，IoT サービスがイベント発生時にリアルタイムで動的に応答できるようにするために，イベント駆動型サービス指向アジャイル IoT サービス通信およびオーケストレーションメカニズムを提案している．この提案アーキテクチャでは，デバイスアクセス管理，リソース管理，統合メッセージ通信インフラおよびサービスオーケストレーションエンジンなど，複数のコンポーネントを接続する構成となっており，デバイスのプロトコルスタックを自動的に構成したり，分散オーバレイネットワークを構築したりするなど，広域かつ大規模な IoT システムを想定している．そのため，本研究が対象としている一般のユーザが市販の宅内 IoT デバイスやスマート家電の連携を実現することには適していない．

3.　提案システム

3.1　概要

本論文では，文献 [10] で提案したアプローチをベースとし，その実現方法および仕様を明確化するとともに，2 章で示した既存システムの課題を解決する異種規格 IoT デバイスの連携システムを実現する．iHAC システムを用いて環境情報に基づいた機器連携を実現する場合，iHAC アプリケーションと同等の機能を有したデバイスを常時起動させ，IoT デバイスがセンシングした環境情報を取得する必要がある．そこで，従来の iHAC アプリケーションから UI 部を除いた iHAC フレームワークと通信処理部から構成されるように設計した iHAC Hub と呼ぶ宅内用デバイスを新たに定義する．

また，高い汎用性や柔軟性を実現する機器連携ルールを記述するために，ECA ルールを導入する．これにより，複数の動作条件や複数台の機器制御，さらには対象機器あたりに複数の制御を同時に実行するルールを記述できるようにする（詳細は 3.2 節にて記述する）．なお，機器連携ルールと ECA ルールの混同を避けるため，以後，機器連携ルールを「レシピ」として定義する[*1]．

3.2　レシピ仕様

iHAC Hub で使用するレシピの記述方法は ECA ルールに則った ECA タグを用いて JSON（JavaScript Object Notation）形式で記述する [20]．ECA ルールとは，アクティブデータベースにおいて自動的に実行する処理を定義するために用いられるルールであり，ルール実行のトリガとなる「Event」，ルールが実行された際に確認される条件「Condition」，条件を満たした際に実行される処理

[*1] IFTTT では Web サービスの連携ルールをレシピと命名しており，本研究でもこの用語を採用した．

© 2020 Information Processing Society of Japan

表 1　レシピの記述で使用する ECA タグ仕様
Table 1　ECA tag specifications used in recipe description.

ECA	タグ	概要
	recipeType	レシピ実行のトリガとなる種類（1：機器状態，2：環境情報）
Event	event	レシピ実行のトリガとなるイベントを記述する配列
	number	condition 部分の要素と対応づけるための番号
	triggerDeviceId	状態を確認したい機器の ID
	fetch	トリガとなる情報の取得方法（MQTT，REST API）
	provider	環境情報を提供するプロバイダの URL
	mqttTopic	MQTT を用いて取得する環境情報のトピック名
Condition	condition	レシピ実行時に確認される条件を記述する配列
	number	event 部分の要素と対応づけるための番号
	triggerDeviceStatus	レシピ実行条件となる機器の状態
	triggerDeviceStatusValue	レシピ実行条件となる機器状態の値
	operator	レシピ実行条件の比較演算子（`$eq`，`$ne`，`$gt`，`$lt`，`$gte`，`$lte`）
	value	レシピ実行条件となる閾値
Action	action	条件を満たした場合に実行するアクションを記述する配列
	controlDeviceId	制御対象機器 ID
	controlDevieStatus	制御内容
	controlDeviceStatusValue	制御内容で使用する値

「Action」から構成される．

表 1 にレシピ記述のために定義した ECA タグ仕様を示す．これらの ECA タグを組み合わせてルールを記述することができるため，柔軟に条件を設定することができる．

3.2.1　Event 情報

Event 情報には number タグ, triggerDeviceId タグ, fetch タグ，mqttTopic タグに必要な情報を記述する．number タグには Event 情報と Condition 情報に記述する内容を対応付けるための番号を記載する．これにより，複数のアクションとコンディションを容易に対応付けることができる．

recipeType タグが 1 の場合，どの機器の状態をトリガとするか指定するために，iHAC フレームワークのデータベースで管理されている機器 ID [*2] を triggerDeviceId タグに指定する．recipeType タグが 2 の場合，環境情報を取得するために，fetch タグと provider タグでデータの取得方法とプロバイダを示す URL を指定する．たとえば，MQTT（Message Queuing Telemetry Transport）[21] を用いて情報を取得する場合，Provider には Broker の URL を，また IoT デバイスやクラウドサービスとして REST API が公開されている場合はその URL を記述する．mqttTopic タグは環境情報を MQTT で取得する場合に使用し，「`living/temperature`」などの環境情報を取得するためのトピック名を記述する．

3.2.2　Condition 情報

Condition 情報には number タグ，triggerDeviceStatus タグ，triggerDeviceStatuValue タグ，operator タグ，value タグに必要な情報を記述する．recipeType タグが 1 の場合，triggerDeviceStatus タグには「`Power`」などトリガとなる機器の状態を，triggerDeviceStatuValue タグには「OFF」など機器状態を示す値を記述する [*3]．recipeType タグが 2 の場合，operator タグにはレシピ実行の条件に使用する比較演算子を，value タグにはレシピ実行の条件となる環境情報の閾値を記述する．

3.2.3　Action 情報

Action 情報には controlDeviceId タグ，controlDeviceStatus タグに必要な情報を記述する．controlDeviceId タグには制御したい機器を一意に判別可能な ID を記述する．controlDeviceStatus タグには controlDeviceId タグで指定された機器の制御内容を記述する．controlDeviceStatus タグに記述された制御内容が温度設定や照度設定などの数値を用いる制御である場合，controlDeviceStatusValue タグを使用し，このタグにその値を記述する．

3.2.4　レシピ形式

図 5，**図 6** に機器状態および環境情報をトリガとするレシピフォーマット例を示す．Event，Condition，Action の内容はオブジェクトの配列形式で記述する．これにより，オブジェクトを複数記述するだけで，複数条件を指定したり，複数の機器制御や対象機器に対して複数の制御命令を行うことができる．なお，複数条件を指定する場合，Condition 部分において論理演算子 [*4] を用いて記述する．

3.3　システム構成

図 7 に提案システムの構成を示す．温度や湿度などの環

[*2] iHAC Hub は定期的に宅内の IoT デバイスやスマート家電を探索しており，発見した機器をデータベースに追加して管理する．

[*3] これらのタグで指定できる値は，ECHONET Lite や DLNA（UPnP）などのプロトコルを用いて取得可能な IoT デバイスやスマート家電の稼働状況，動作モード，設定値などが該当する．

[*4] `$and`，`$or`，`$nor`，`$not` の各論理演算子をサポート．

© 2020 Information Processing Society of Japan

```
{
  "recipeType": 1,
  "event": [
    {
      "number": i,
      "triggerDeviceId": deviceId,
    }
  ],
  "condition": [
    {
      "number": i,
      "triggerDeviceStatus": "status",
      "triggerDeviceStatusValue": "value"
    }
  ],
  "action": [
    {
      "controlDeviceId": deviceId,
      "controlDeviceStatus": "status"
    },
    {
      "controlDeviceId": deviceId,
      "controlDeviceStatus": "status",
      "controlDeviceStatusValue": "value"
    }
  ]
}
```

図 5　機器状態をトリガとするレシピフォーマット例
Fig. 5　Example of recipe format triggered by device status.

```
{
  "recipeType": 2,
  "event": [
    {
      "number": i,
      "fetch": "mqtt", "provider": "brokerUrl",
      "mqttTopic": "topicName"
    }
    {
      "number": j,
      "fetch": "rest", "provider": "apiUrl"
    }
  ],
  "condition": [
    "$and": [
      {
        "number": i,
        "operator": "operator", "value": value
      },
      {
        "number": j,
        "operator": "operator", "value": value
      }
    ]
  ],
  "action": [
    {
      "controlDeviceId": deviceId,
      "controlDeviceStatus": "status"
    }
  ]
}
```

図 6　環境情報をトリガとするレシピフォーマット例
Fig. 6　Example of recipe format triggered by environment information.

境情報を計測する IoT デバイスおよび ECHONET Lite や DLNA 準拠のスマート家電は宅内に設置されている市販のデバイスを想定し，提案システムに関わる機能追加はいっさい行わない．また，IoT デバイスがセンシングした環境情報はクラウドへアップされ，各メーカが提供する API などを利用してセンシングデータが取得できるものとする[*5]．

[*5] IoT デバイスから直接センシングデータを取得できる API などが公開されている場合は，クラウドを経由せず iHAC Hub が直接取得してもよい．

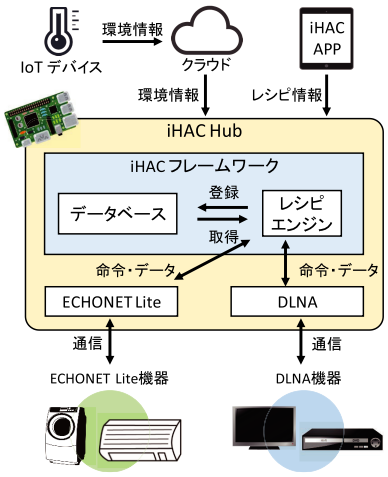

図 7　提案システムの構成
Fig. 7　Configuration of the proposed system.

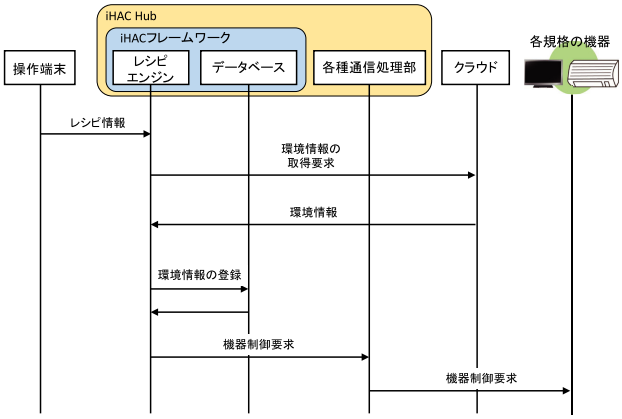

図 8　機器制御シーケンス
Fig. 8　Device control sequence.

新たに定義する iHAC Hub は環境センシングをする IoT デバイスと各種スマート家電を連携するハブ的なデバイスである．iHAC Hub に実装する iHAC フレームワークには，3.2 節で定義した ECA ルールにのっとり記述されたレシピを解析するレシピエンジンを追加する．なお，レシピは従来の iHAC アプリケーションを導入した操作端末で作成および設定し，iHAC Hub へ送信するものとする．

3.4　システム動作

iHAC Hub を用いた環境情報に基づく機器連携の流れを図 8 を用いて説明する．iHAC Hub は受信したレシピ情報をレシピエンジンで解析し，レシピの Event 情報で指定された方法に従って，クラウドから環境情報の取得を開始する．取得した環境情報は iHAC フレームワーク内のデータベースに蓄積され，iHAC Hub はレシピの Condition 情報に記載された動作条件を満たしているか確認する．動作条件を満たした場合，レシピの Action 情報で指定された機器制御を行うために，iHAC フレームワークは ECHONET Lite や DLNA などの通信処理部に命令を行い，スマート

© 2020 Information Processing Society of Japan

家電と通信を行う．

以上の処理により，iHAC Hub は IoT デバイスが計測した環境情報に基づいた機器連携が可能となる．

recipeType が 2，fetch が MQTT のとき，iHAC Hub は MQTT Subscriber としてクラウド（provider タグで指定された Broker）からリアルタイムに環境情報の取得を開始する．取得した環境情報はデータベースに登録され，レシピ情報に記載された機器の動作条件を満たしているか比較する．iHAC Hub は環境情報の取得処理を繰り返し，動作条件を満たした場合にアクションに記載された機器の制御を行う．

4. 実装・動作検証

4.1 実装

iHAC Hub による IoT デバイス連携を実証するため，Raspberry Pi 3 を用いて iHAC Hub のプロトタイプを実装した．従来の iHAC フレームワークは Objective C++ で実装されていたため，Linux OS で動作するよう C 言語により再設計し，今回はその一部機能を実装した [22]．

温度センサおよび湿度センサは Raspberry Pi 3 で開発し，10 秒ごとに室内の温度と湿度をセンシングし，MQTT によりクラウドサービス CloudMQTT [23] へ送信する Publisher とした．なお，MQTT のトピック名はそれぞれ「`living/temperature`」と「`living/humidity`」とした．

4.1.1 iHAC フレームワークの拡張

iHAC フレームワーク部に実装するレシピエンジンは動作検証を行うための必要最低限の機能として，レシピ解析機能の一部，MQTT を用いて環境情報を取得する Subscriber プロセス，取得した環境情報をデータベースに登録する機能，データベースを実装した．

JSON 形式のレシピ情報を解析するために，JSON パーサである Jansson [24] を利用した．fetch タグに `mqtt` が指定されていた場合，MQTT Subscriber プロセスを起動する．

MQTT Subscriber プロセスには，C 言語で MQTT が使用できるライブラリである Mosquitto [25] を使用した．Subscriber プロセスは Broker の役割を果たす CloudMQTT へ受信したいトピック名などを送信し，環境情報の取得を開始する．CloudMQTT から環境情報を取得する度に，時刻，MQTT で使用したトピック名と環境情報をレシピエンジンへ返すようにした．

データベースには SQLite3 を用いて，取得した環境情報を登録する環境情報データベースを実装した．環境情報データベースには，環境情報を取得した IoT デバイスが設置されている部屋 ID，および Subscriber プロセスからわたされる環境情報などを登録する．

以上の実装により，今回のプロトタイプ実装では次の機能をサポートした．

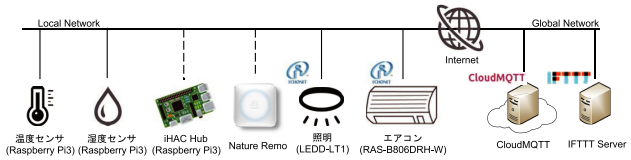

図 9　動作検証および性能評価用ネットワーク構成

Fig. 9　Network configuration for operation verification and performance evaluation.

```
{
  "recipeType":2,
  "event": [
    {
      "number": 1, "fetch": "MQTT",
      "provider": "m13.cloudmqtt.com",
      "mqttTopic": "living/temperature"
    },
    {
      "number": 2, "fetch": "MQTT",
      "provider": "m13.cloudmqtt.com",
      "mqttTopic": "living/humidity"
    }
  ],
  "condition": [
    "$and": [
      {
        "number": 1,
        "operator": "$gte", "value": 30.0
      },
      {
        "number": 2,
        "operator": "$gte", "value": 32.8
      }
    ]
  ],
  "action": [
    {
      "controlDeviceId": 1,
      "controlDeviceStatus": "Power",
      "controlDeviceStatusValue": "On"
    },
    {
      "controlDeviceId": 2,
      "controlDeviceStatus": "lampColor"
    }
  ]
}
```

図 10　ECA ルールに基づいたレシピ例

Fig. 10　Example of recipe based on ECA rule.

表 2　処理時間の測定結果

Table 2　Measurement result of processing time.

処理時間 [ms]	RTT	サーバ処理	機器制御	合計
既存システム	111.6	1,889.4	664.6	2,665.6
提案システム	178.8	141.1	780.9	1,100.8

- ECHONET Lite 対応スマート家電の電源 On/Off を検知して，他の ECHONET Lite スマート家電を制御
- MQTT で温湿度情報を取得し，閾値を超えた場合，ECHONET Lite スマート家電を制御

4.1.2 通信処理部

本論文では iHAC Hub における通信処理部として uEcho [26] ライブラリを使用して ECHONET Lite モジュールを実装した．ECHONET Lite モジュールは，レシピエンジンから制御したい機器の IP アドレス，オブジェクトコード，ECHONET Lite プロパティ，処理内容などを受け取ると，ECHONET Lite 機器を制御するためのコントローラを作成し，ネットワーク内に存在する ECHONET

© 2020 Information Processing Society of Japan

表 3　既存研究との比較
Table 3　Comparison with existing researches.

	汎用性	柔軟性	市販デバイスのサポート	利用の簡便性
快適嗜好性による自動制御手法 [12]	×	×	○	○
IFTTT [13]	△	○	△	△
新サービス構築手法 [15]	○	×	○	○
オントロジー駆動型アダプテーション層 [16]	△	△	○	○
IoT サービス連携アーキテクチャ [18], [19]	○	○	×	×
iHAC Hub	○	○	○	○

Lite 対応家電の探索を開始する．レシピエンジンから取得した IP アドレス，オブジェクトコードと一致する機器が見つかった場合，機器制御を行うための電文を作成し送信することで制御を行う．

4.2　動作検証

環境情報に基づいた機器連携が可能であることを確認するため，実装した iHAC Hub のプロトタイプを用いて動作検証を行った．**図 9** に大学研究室内に構築した検証用ネットワーク構成を示す．温度センサと湿度センサ，iHAC Hub，ECHONET Lite 対応エアコンと照明を同一ネットワークに接続し動作検証を行った．「リビングの温度と湿度が熱中症になる危険性がある値を超えた場合，エアコンを起動し，照明の色を電球色に変える」ことを確認するために，**図 10** に示したレシピを作成し[*6]，ドライヤーを用いて温度センサを温め，研究室を加湿して湿度を上げた．

CloudMQTT 上でモニタリングしていた温度が 30 度以上，湿度が 32.8%以上となった際，研究室内のエアコン（`controlDeviceId=1`）が起動し，照明（`controlDeviceId=2`）の色が変わった．この結果，提案システムにより環境情報に基づいた機器連携が可能であることを確認した．また，ECA ルールに基づいたレシピに従って，複数の条件，複数の機器制御に対応できることも確認した．

5.　評価

5.1　性能評価

IoT デバイスがレシピに設定されたトリガを満たす環境情報を検出してから，実際に操作対象機器を制御するまでに要した時間が実用上問題ないかを確認するために，iHAC Hub のプロトタイプと既存システムとしてあげた IFTTT と Nature Remo を用いて機器を制御するまでに要した時間をそれぞれ 60 fps でビデオ撮影して計測した．なお，性能評価を行ったネットワーク構成は図 9 と同様で，Nature Remo を他の機器と同じネットワークに接続させた．提案システムの処理時間は，温度センサが動作条件を満たした温度を取得した時刻から，iHAC Hub が制御メッセージを送信し，エアコンが起動するまでの時刻を，既存システムの処理時間は，温度センサが動作条件を満たした温度を取得した時刻から，IFTTT を用いて Nature Remo が赤外線信号を送信し，エアコンが起動するまでの時刻とする．試行回数はそれぞれ 10 回とし，その平均を算出した[*7]．

表 2 に測定結果を示す．提案システムでは，温度センサが動作条件を満たした温度を取得してからエアコンが起動するまでに 1,100.8 [ms] の時間を要した．一方，既存システムでは，2,665.6 [ms] を要した．提案システムの処理時間は既存システムの処理時間の 41.3%であるため，提案システムの処理時間は実用上問題ないといえる．

5.2　既存システムとの比較評価

表 3 に提案システムと既存システムにおける，レシピの汎用性と柔軟性，ならびに当該システム利用の簡便さと市販デバイスでの利用可能性に関する比較を示す．2 章で述べたとおり，室内温度の快適嗜好性を考慮した加湿空気清浄機の自動制御手法については，温湿度センサと加湿空気清浄機以外の機器連携を実現する手段がなく，汎用性や設定の柔軟性に欠けている．ホームネットワークにおけるデバイス間連携に基づく新サービス構築手法は，統合ミドルウェアにより様々なデバイスの制御が可能であるものの，複数条件に基づく機器連携については検討されていない．IFTTT を用いたシステムでは複数条件に基づいた機器連携が可能であるが，独自のアプレットを作成する必要があり，一般のユーザが利用するには困難である．IoT サービス連携アーキテクチャは多種多様な IoT サービスおよび IoT デバイスを柔軟に連携できる仕組みが考えられているが，まったく新しいアーキテクチャを提案しているため，現在市販されている IoT デバイスが使えなかったり，ホームネットワークへの導入面で高いハードルが存在する．

これに対して，提案手法では ECA ルールを導入したた

[*6] 本来は iHAC アプリケーションで項目を選択などすることにより JSON 形式のレシピを生成して iHAC Hub へ送信するが，今回のプロトタイプではレシピを直接コーディングし，iHAC Hub が JSON ファイルを読み込むこととした．

[*7] 本評価実験では試行回数を 1 セット 10 回とし，別の日時にもう 1 セットの計 20 回測定を行い，1 セット目と 2 セット目のデータに対して，t 検定による有意差検定を行った結果，有意水準 5%を大きく上回り（既存システム：$p = 0.881$，提案システム：$p = 0.868$），有意差が確認できなかったため，試行回数を 10 回としたデータを採用した．

© 2020 Information Processing Society of Japan

め，複数条件や複数の制御処理に対応した柔軟な機器連携レシピを作成することができる．iHAC システムの特徴である通信処理部の追加と iHAC フレームワークとの連携部分を実装することにより，様々な通信規格に対応することができる．また，既存の IoT デバイスやスマート家電の仕様を活用しているため，市販の機器をそのまま利用でき，ユーザは iHAC Hub を宅内に設置し，iPad などでレシピを作成するだけで異種規格の IoT デバイスを連携することが可能である．そのため，高い汎用性も実現しており，提案システムが優れていることが分かる．

6. まとめ

本論文では，環境情報に基づいて異種規格の IoT デバイス連携をサポートする iHAC Hub を提案した．提案システムでは，IoT デバイスがセンシングした環境情報の取得と機器制御を iHAC Hub が行う．これにより既存の iHAC システムで課題となっていた操作端末を常時起動することなく環境情報に基づく機器連携が可能となった．また，ECA ルールによるレシピの作成や MQTT などのプロトコルのサポートにより，既存システムと比較して高い汎用性と柔軟性を兼ね備えたシステムを実現することについて述べた．提案システムのプロトタイプを実装し，実環境において動作確認および性能評価実験を行った結果，実用上問題のない処理時間で温度センシングシステムとスマート家電を連携して制御できることを確認した．

今後は，iHAC アプリケーションで ECA ルールを作成する UI の改良および iHAC Hub へのレシピ送信機能などを行う．また，文献 [27] の研究成果を iHAC Hub に組み込むことにより，スマート家電だけではなく，通信機能を持たない従来の家電機器の連携もサポートする予定である．さらに，ユーザの状態，コンテキストや嗜好などに基づいた，より複雑な機器連携の実現方法についても検討していく．

謝辞　本研究の一部は JSPS 科研費 15K15987 の助成を受けて実施したものである．

参考文献

[1] 総務省：平成 30 年版情報通信白書のポイント, p.7 (2018)，入手先 ⟨http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/pdf/30honpen.pdf⟩.
[2] 総務省：平成 30 年版情報通信白書のポイント，pp.67–68 (2018)，入手先 ⟨http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/pdf/30honpen.pdf⟩.
[3] Digital Living Network Alliance (DLNA), available from ⟨http://www.dlna.org/⟩.
[4] CONSORTIUM, E.: ECHONET Lite, available from ⟨https://echonet.jp/⟩.
[5] Zigbee Alliance, available from ⟨http://www.zigbee.org/⟩.
[6] Z-Wave, available from ⟨http://www.z-wave.com/⟩.
[7] Kushalnagar, N., Montenegro, G. and Schumacher, C.: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, RFC 4919 (Informational) (2007).
[8] Thread, available from ⟨https://www.threadgroup.org/⟩.
[9] 梅山莉奈，増田剛志，鈴木秀和：規格の違いを意識しない直感的家電制御システムの提案，情報処理学会論文誌コンシューマ・デバイス&システム（CDS），Vol.6, No.1, pp.84–93 (2016).
[10] 江崎敬俊，梅山莉奈，鈴木秀和：iHAC システムにおける異種プロトコル間の機器連携手法の提案，第 79 回全国大会講演論文集，Vol.2017, No.1, pp.65–66 (2017).
[11] Dittrich, K.R., Gatziu, S. and Geppert, A.: The Active Database Management System Manifesto: A Rulebase of ADBMS Features, *Rules in Database Systems*, Sellis, T.K. (Ed.), Lecture Notes in Computer Science, Vol.985, pp.3–20, Springer (1995), available from ⟨http://dblp.uni-trier.de/db/conf/rules/rules95.html#DittrichGG95⟩.
[12] 原　晃汰，松井加奈絵：室内湿度の快適嗜好性を考慮した加湿空気清浄機の自動制御，情報処理学会研究報告コンシューマ・デバイス&システム（CDS），No.10, pp.1–6 (2017).
[13] IFTTT (If This Then That), available from ⟨https://ifttt.com/⟩.
[14] Nature Remo, available from ⟨http://nature.global/⟩.
[15] 秋田浩也，川上智史，佐藤健哉：ホームネットワークにおけるデバイス間連携に基づく新サービス構築手法の検討，マルチメディア，分散協調とモバイルシンポジウム 2016 論文集，Vol.2016, pp.89–93 (2016).
[16] Pham, V.C., Lim, Y., Sgorbissa, A. and Tan, Y.: An Ontology-driven ECHONET Lite Adaptation Layer for Smart Homes, *Journal of Information Processing*, Vol.27, No.27, pp.360–368 (2019).
[17] Belbachir, A., Drobics, M. and Marschitz, W.: Ambient Assisted Living for ageing well: An overview, *Elektrotechnik und Informationstechnik*, Vol.127, No.7-8, pp.200–205 (2010).
[18] Kum, S.W., Moon, J. and Lim, T.: Design of fog computing based IoT application architecture, *2017 IEEE 7th International Conference on Consumer Electronics - Berlin* (*ICCE-Berlin*), pp.88–89 (2017).
[19] Cheng, B., Zhao, S., Niu, M. and Chen, J.: Agile IoT service communication and orchestration platform using event driven SOA paradigm, *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops* (*INFOCOM WKSHPS*), pp.1–2 (2018).
[20] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format, RFC 8259, IETF (2017).
[21] International Business Machines Corporation (IBM), Eurotech: MQTT V3.1 Protocol Specification, Technical Report (2010).
[22] 横地リー紫音，岡田真実，林　宏輔，鈴木秀和：マルチプラットフォーム対応 iHAC フレームワークの設計，第 80 回全国大会講演論文集，Vol.2018, No.3, pp.107–108 (2018).
[23] Cloud MQTT, available from ⟨https://www.cloudmqtt.com/⟩.
[24] Lehtinen, P.: Jansson — C library for working with JSON data, available from ⟨http://www.digip.org/jansson/⟩.
[25] mosquitto.org: Mosquitto, available from ⟨https://mosquitto.org/⟩.
[26] Konno, S.: uecho, available from ⟨https://github.com/cybergarage/uecho⟩.
[27] 上野大輝，岡田真実，鈴木秀和：柔軟な設定変更が可能な

© 2020 Information Processing Society of Japan

レガシー家電機器用スマート化システムの設計，第 79 回全国大会講演論文集，Vol.2017, No.1, pp.383–384 (2017).

林 宏輔　（学生会員）

2018 年名城大学理工学部情報工学科卒業．現在，同大学大学院理工学研究科情報工学専攻修士課程に在学中．ホームネットワークに関する研究に従事．学士（工学）．IEEE 会員．

鈴木 秀和　（正会員）

2004 年 3 月名城大学理工学部情報科学科卒業．2009 年 3 月同大学大学院理工学研究科電気電子・情報・材料工学専攻博士後期課程修了．2008 年 4 月日本学術振興会特別研究員．2010 年 4 月名城大学理工学部助教．2015 年 4 月より同大学理工学部准教授および東北大学電気通信研究所共同研究員を兼務．ネットワークセキュリティ，モバイルネットワーク，ホームネットワーク等の研究に従事．博士（工学）．IEEE，ACM，電子情報通信学会各会員．本会シニア会員．

© 2020 Information Processing Society of Japan