iHAC Hub を用いた
異種 IoT サービス連携システムに関する研究

183426014　林 宏輔
鈴木研究室

1. はじめに

ネットワークを通じて操作が可能なスマート家電や，温度や湿度などの環境情報をセンシングする IoT デバイスを活用したサービスが普及しつつある．しかしながら，スマート家電や IoT デバイスに採用されている通信規格は複数存在するため，それらを連携することは困難である．

本研究では，通信規格の違いを意識することなく，機器を直感的に制御することができる iHAC（intuitive Home Appliance Control）システム [1] を拡張し，機器連携をサポートする宅内用デバイス iHAC Hub を提案する．iHAC Hub がユーザが作成した機器連携のルールに基づき，宅内の IoT デバイスから環境情報を取得し機器制御を行うことで異種 IoT サービス連携を実現することができる．

iHAC Hub のプロトタイプを実装し，既存の機器連携サービスとの性能評価実験を行い，既存サービスより短時間で機器連携できること，また，適切なパラメータを用いて生成した音声命令を用いることで，高い認識率でスマートスピーカ対応デバイスを制御できることを確認した．

2. iHAC システム

iHAC システムは，通信規格の違いを意識することなく，直感的に家電を制御することを目標としたシステムである．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 部，機器の登録や探索に関わる API が定義された iHAC フレームワーク部および各規格の通信処理部から構成される．iHAC フレームワーク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，通信規格の違いを抽象化する．また，文献 [2] では iHAC システムの応用例として，レシピと呼ばれる 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く機器連携が可能である．

しかしながら，iHAC システムを用いた機器連携を実現するためには，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，環境情報や機器状態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．

3. 提案システム

3.1 概要

iHAC システムを用いて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，機器の状態や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，ECA ルール [3] を導入しレシピを表現することで，柔軟かつ，汎用性のある機器連携を実現する．

3.2 構成

図 1 に提案システムの構成を示す．温度や湿度などの環境情報を計測する IoT デバイスおよび ECHONET Lite や DLNA などの通信規格に対応した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一切行わない．また，IoT デバイスが計測

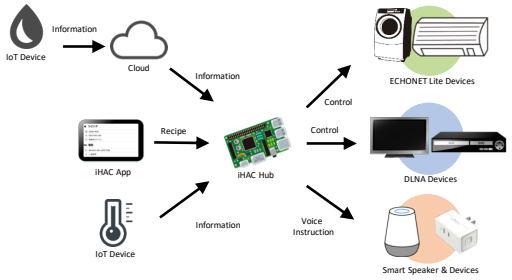

図 1: 提案システムの概要

```
{ "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 2: ECA ルールに基づいたレシピ例

した環境情報はクラウドへアップロードされ，各メーカが提供する API などから取得できるもの，あるいはクラウドを経由せず iHAC Hub が直接取得できるものとする．

新たに定義する iHAC Hub は環境情報をセンシングをする 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，ECA ルールに則り記述されたレシピを解析し処理するレシピエンジン，環境情報を取得するプロセスを追加する．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．

3.3 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 形式で記述する．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，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 “Action” それぞれに対応したタグを用いてルールを表現する．例えば，「リビングで熱中症の危険性のある温湿度になった場合，エアコンを起動し，照明の色を変更する」というレシピは図 2 のように表現され，複数の動作条件や複数台の機器制御を行うといったルールを表現することが可能となる．

3.4 動作シーケンス

iHAC Hub を用いた環境情報に基づく機器連携の流れを図 3 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，スマートスピーカ対応デバイスを制御する場合は，音声命令の生成を行う．その後，レ

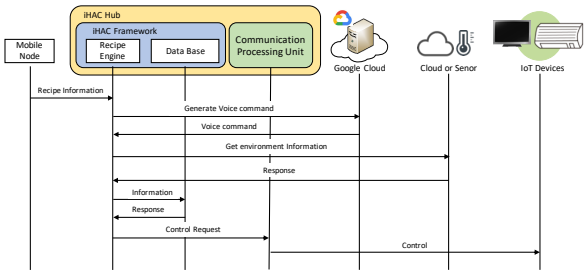

図 3: 機器制御シーケンス

表 1: 処理時間の測定結果

処理時間 [ms]	RTT	サーバ処理	機器制御	合計
既存サービス	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

シピの Event 情報で指定された方法に従って，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の通信処理部を呼び出し，スマート家電と直接通信し制御する，あるいは音声命令を再生し，音声命令を認識したスマートスピーカによって機器制御を行う．

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．

4. 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，レシピエンジン，環境情報を取得するプロセス，データベース，スマートスピーカ対応デバイスを制御するための音声命令を生成する機能を実装した．なお，音声命令を生成する際，Google Cloud Text-to-Speech を使用した．また，通信処理部では，ECHONET Lite 機器と DLNA 機器を制御する機能，レシピエンジンによって生成された音声命令を再生する機能を実装した．なお，ECHONET Lite ライブラリとして uEcho，DLNA ライブラリとして libupnp を使用した．

5. 評価

5.1 既存サービスとの処理時間の比較

IoT デバイスがレシピで設定された動作条件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提案システムと既存サービスで処理時間の比較を行った．異なる Web サービスを連携することができる IFTTT を用いて，リモコンの赤外線信号を送信し機器制御を行う Nature Remo と温度センサを連携させた既存サービスと提案システムに対して，機器制御するまでに要した時間をそれぞれ 60fps でビデオ撮影して計測した．なお，性能評価を行なったネットワーク構成は図 4 の通りであり，試行回数はそれぞれ 10 回とし，その平均を算出した．

表 1 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，既存サービスでは，2,665.6[ms] の時間を要した．提案システムの処理時間は既存サービスの処理時間の 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．

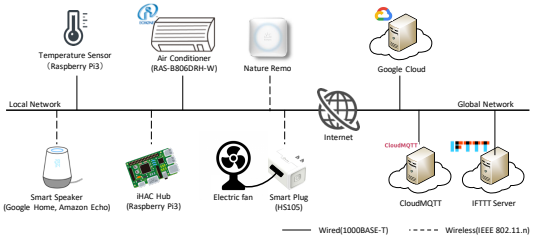

図 4: 動作検証用ネットワーク構成

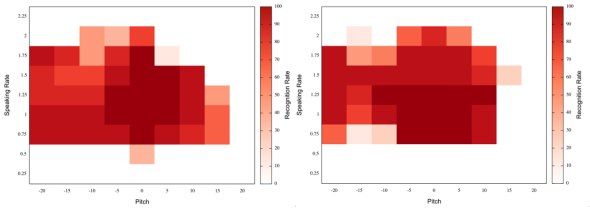

図 5: Amazon Echo の認識率

図 6: Google Home の認識率

5.2 スマートスピーカの認識率の比較

人が発話によって機器を制御する場合，その日の体調などによって認識率にばらつきが生じる．したがって，提案システムにおいて生成された音声命令が高い認識率があるのか，また再現性があるのか市販のスマートスピーカを用いて認識率の比較を行った．iHAC Hub がスマートスピーカ対応機器であるスマートプラグを制御するため，Google Cloud Text-to-Speech で音声命令を生成する際，話す速度を表す speakingRate と声のピッチを表す pitch の値を変化させ音声命令を生成した．生成した音声命令を市販のスマートスピーカである Amazon Echo と Google Home に対して 10 回ずつ再生し，認識率の比較を行った．

両スマートスピーカの認識率をヒートマップ にて表現した結果を図 5，図 6 に示す．図より，speakingRate が 1.0，1.25 かつ pitch が-5，0．5，または speakingRate が 0.75 かつ pitch が 0 に設定した際，両スマートスピーカでの認識率が 100%となった．したがって，これらの設定を用いて音声命令を生成することで，実用上問題ない精度で機器連携が可能であると考えられる．

6. まとめ

本研究では iHAC システムを応用し，機器連携をサポートする宅内用デバイス iHAC Hub を用いた異種 IoT サービス連携システムを提案した．提案手法により，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機器連携が可能であり，ECA ルールを導入しレシピを作成することで，複数の動作条件に基づく制御や，複数台の機器制御など，柔軟な機器連携を実現することができる．また，比較評価の結果，提案システムは既存サービスより短時間で制御できること，スマートスピーカ対応デバイスを高い認識率で制御可能であることを確認した．

参考文献

[1] 梅山．他：情報処理学会論文誌　コンシューマ・デバイス＆システム, Vol. 6, No. 1, pp. 84–93, 2016.

[2] 江崎．他：第 79 回情報処理学会全国大会講演論文集, Vol. 2017, No. 3, pp. 65–66, 2017.

[3] Klaus R. Dittrich et al.: Rules in Database Systems, Vol. 985, pp. 3–20, 1995.