6W-04

iHAC Hubにおける IoT デバイス連携のレシピ解析を行う ECA ルールエンジンの実装

佐藤 里菜子 †1　林 宏輔 †2　鈴木 秀和 †1
†1 名城大学理工学部　†2 名城大学大学院理工学研究科

1 はじめに

情報家電の通信プロトコルの違いにより，ユーザは操作を変えなければならない．この課題を解決するため，通信規格の違いを意識せずにスマート家電を直感的に制御することが可能な iHAC（intuitive Home Appliance Control）システム [1] や，環境情報に基づいた機器連携を実現するホームデバイス iHAC Hub[2] が提案されている．

本稿では iHAC システムにおいて ECA ルール [3] に則って記述した家電連携に関するレシピの解析と，レシピに従って機器制御を行う ECA ルールエンジン（レシピエンジン）を提案する．

2 iHAC Hub の概要

iHAC Hub は機器の登録や探索を行う API を定義した iHAC フレームワークの機能と環境情報の取得などを行う各種 API を持つデバイスであり，環境情報をリアルタイムに収集することで機器連携を可能にしている．機器連携の内容は温度や湿度などの環境情報の閾値，連携させる機器の動作内容などをレシピとして記述し，このレシピに基づいて iHAC Hub は環境情報の取得や機器連携などの処理を行う．

レシピに基づいた処理を行うためにレシピを解析する必要があるが，既存の iHAC Hub には解析を行うレシピエンジンが未実装である．

3 提案システム

3.1 構成

図 1 に提案システムの概要を示す．レシピエンジンは解析 API，環境情報 API，機器状態 API の 3 つの API で構成される．解析 API はレシピの解析を行い，そのレシピのトリガの種類によってその後の処理を決定する．環境情報 API は環境情報がトリガとなる場合に呼び出され，環境情報の取得・確認を行うバックグランドプロセスの呼び出しと受信した環境情報の値がレシピ条件を満たすかどうかの確認を行う．機器状態 API は機器状態がトリガとなる場合に呼び出され，機器状態の確認とその結果がレシピ条件を満たすかどうかの確認を行う．

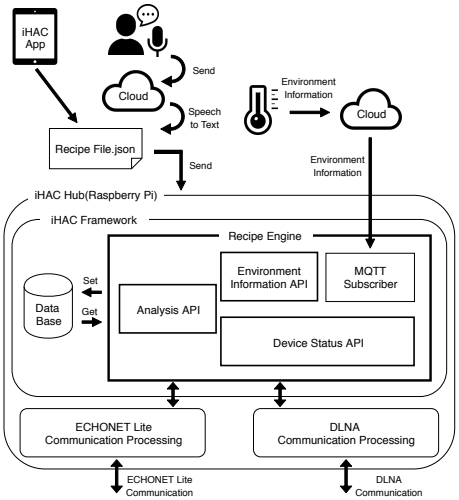

図 1　提案システムの概要

```
{
  "recipeType":1,
  "event":[
    {"number":1, "topic":"living/temperature", "type":"receive"}
  ],
  "condition":[
    {"number":1, "operator":">", "value":30}
  ],
  "action":[
    {"controlDeviceId":1, "controlDeviceStatus":"powerOn"}
  ]
}
```

図 2　ECA ルールに基づいて記述したレシピの例

家電の連携動作を記述したレシピは ECA ルールに基づいて記述し，JSON（JavaScript Object Notation）形式で作成する．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るルールであり，ルール実行のトリガとなる Event，ルール実行の際に確認される条件 Condition，実行される処理 Action から構成される．例えば，「リビングの温度が 30 度より高いとき，リビングのエアコンの冷房を入れる．」というレシピは図 2 のように記述される．

3.2 動作

iHAC Hub 内でのレシピ解析とその結果に基づいたレシピ実行の流れについて説明する．ユーザは iHAC シス

Implementation of ECA Rule Engine to Analyze Recipe of IoT Device Cooperation in iHAC Hub
Rinako Satoi[†1], Kosuke Hayashi[†2] and Hidekazu Suzuki[†1]
[†1] Faculty of Science and Technology, Meijo University
[†2] Graduate School of Science and Technology, Meijo University

テムを導入した操作端末から UI を操作，またはスマホやタブレットなどの端末に対して発話することでレシピを作成する．発話によってレシピを作成する場合，音声認識サービスを用いて音声のテキスト化を行い，文章中から必要な単語を抽出する必要がある．レシピファイルには，環境情報の値がトリガとなる場合か機器の状態がトリガとなる場合かを判断するレシピタイプやレシピの動作条件，連携する機器の動作内容が記述される．

作成されたレシピファイルは iHAC Hub へ送信される．レシピを受信した iHAC Hub は解析用のスレッドを立て，レシピエンジンでレシピの解析を行う．まず，レシピファイルに記述されたレシピタイプを確認し，トリガが環境情報の場合か機器状態の場合かを判断する．トリガが環境情報の場合は環境情報 API を呼び出す．環境情報の値を取得するためにスレッドを立て，クラウド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し，取得した値がレシピの Condition 部分に記載されたレシピ実行の条件を満たすまで確認を行う．条件を満たした場合は各通信規格の通信処理部を呼び出し，機器制御を行う．トリガが機器状態の場合は機器状態 API を呼び出し，現在の機器の状態を確認する．確認した機器状態がレシピ条件を満たさない場合は機器制御を行わず終了し，条件を満たす場合はトリガが環境情報の場合と同様にして各通信規格の通信処理部を呼び出し，機器制御を行う．

4 実装と評価

4.1 実装

ECA ルールによって記述されたレシピを解析するレシピエンジンのプロトタイプを Raspberry Pi3 に実装した．レシピエンジンは C 言語を用いて実装し，JSON 形式のレシピファイルをパースするためのパーサとして Jansson[4] を，ECHONET Lite 機器を制御するライブラリとして uEcho[5] を使用した．

4.2 動作検証

実装したレシピエンジンについて，「リビングの照明が切れたとき，寝室の照明をつける」というレシピを用いて動作検証を行った．検証では，レシピエンジンがレシピファイルを解析することでトリガが機器状態であると判断し，機器状態に基づいた機器制御が可能かを確認する．リビングの照明と寝室の照明にはそれぞれ TOSHIBA の LEDD-LT1，LEDH-LT4 を使用した．

動作検証の結果，リビングの照明が切れている場合，寝室の照明がつき，レシピに記載された内容の機器連携が正しく実行されたことを確認した．

4.3 評価

レシピエンジンによるレシピ解析が実際にレシピを実行する上で問題ないか，iHAC Hub がレシピを受信してから機器制御を行うまでの時間のうちレシピ解析に要した時間の割合を求め評価する．図 3 に測定環境のネットワーク構成を示す．iHAC Hub がレシピを受け取ってから機器制御を終えるまでの時間をレシピ実行時間，受け取ったレシピファイルを解析 API が受信してから機器状態 API を呼び出すまでの時間をレシピ解析時間として，それぞれの時間を計測する．試行回数は 100 回とし，それぞれの平均時間とレシピ実行時間におけるレシピ解析時間の割合を求める．

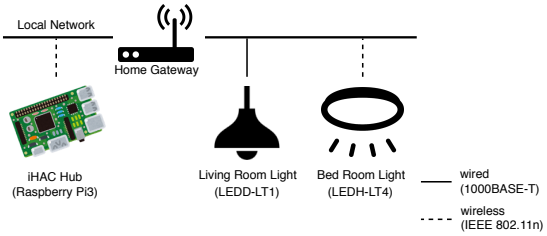

図 3　測定環境のネットワーク構成

表 1　測定結果

	処理時間 [ms]
最大レシピ解析時間	7.962
平均レシピ解析時間	1.108
最小レシピ解析時間	8.070
平均レシピ実行時間	1.302×10^3

表 1 に測定結果を示す．レシピ実行には平均で 1.302[s]，レシピ解析には平均で 1.108[ms] の時間を要した．また，レシピ解析時間の分散は 7.113×10^{-7} であった．レシピ実行時間におけるレシピ解析時間の割合は 0.085% であることから，レシピ解析はレシピ実行処理自体に大きな影響を与えないことがわかり，レシピエンジンによるレシピ解析は実用上問題がないといえる．

5 まとめ

本稿では iHAC Hub における ECA ルールを解析するレシピエンジンの設計と実装を行った．作成したレシピエンジンの動作検証を行った結果，レシピ内容に基づいた処理を正しく行えることを確認した．また，レシピ実行において実用上問題ない時間で解析が可能であることがわかった．

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] 林．他：DICOMO2018 論文集, pp. 283–290 (2018).
[3] Klaus R. Dittrich et al.: Rules in Database Systems, Vol. 985, pp. 3–20 (1995).
[4] Jansson：http://www.digip.org/jansson/
[5] GitHub-cybergarage/uecho：https://github.com/cybergarage/uecho