Vol.2019-MBL-92 No.21
Vol.2019-CDS-26 No.21
2019/8/30

iHAC Hubとスマートスピーカを用いたM2Mデバイス連携手法の提案

林 宏輔[1,a)]　鈴木 秀和[1,b)]

概要：音声対話型の AI アシスタントを搭載した Amazon Echo や Google Home 等のスマートスピーカとそれらに対応した IoT（Internet of Things）デバイスが数多く登場している．筆者らは通信プロトコルで制御可能な IoT デバイスの連係動作を実現する iHAC Hub を提案してきた．本稿では，iHAC Hub とスマートスピーカを用いた M2M（Machine-to-Machine）デバイス連携手法について提案する．提案手法では，環境情報をセンシングする IoT デバイスとスマートスピーカ対応 IoT デバイスを連携動作させるためのフレームワークを提供する．iHAC Hub はクラウドサービスを利用して IoT デバイスを制御するための音声命令を生成し，これをスマートスピーカに向けて再生することにより，人手を介さずにスマートスピーカ対応 IoT デバイスの制御を実現する．

Proposal of Cooperation Method of M2M Devices Using iHAC Hub and Smart Speaker

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

1. はじめに

音声対話型の AI（Artificial Intelligence）アシスタントを搭載した Amazon Echo [1] や Google Home [2] などのスマートスピーカの登場により，ユーザはスマートスピーカに話しかけることで機器を制御することが可能になった．世界のスマートスピーカの市場規模は今後も拡大していくと予想されており，人間がこれまで身体的に感じ取っていた温度や湿度，照度や空気の品質などの環境情報をもとにリモコンや専用のアプリを用いて機器を制御するのではなく，声で機器を制御する生活が浸透すると予想される [3]．

さらに，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，インターネットに接続可能な機器である IoT デバイスが普及している．それに伴い，IoT デバイスを用いることで人間がこれまで身体的に感じ取っていた温度や湿度などの環境情報をデジタルデータとして取得および蓄積し様々な IT サービスに活用されるようになった [4,5]．

しかし，スマートスピーカーを利用して IoT デバイスを制御するソリューションは，ユーザが制御命令を発声し，スマートスピーカが音声認識することにより動作を実行する仕組みである．これは，人がリモコンで操作し，受信側装置が信号パターンを認識して所定の動作を行う従来の仕組みと基本的に同じである．そのため，デバイス同士が自律的に交渉して動作する M2M（Machine-to-Machine）システムの実現には至っていない．

一方，筆者らは iHAC Hub と呼ぶ宅内に設置する IoT デバイスを導入することにより，温度センサや湿度センサ，照度センサなどの多種多様な IoT デバイスがセンシングした環境情報に基づいてスマート家電を連携するシステムを提案している [6]．しかし，iHAC Hub が制御可能なスマート家電は ECHONET Lite [7] や DLNA（Digital Living Network Alliance） [8] など通信プロトコルに対応

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

しなければならない．市販されているスマートスピーカ対応デバイスの多くはこれらのプロトコルを搭載していないため，iHAC Hub で直接制御することができない．iHAC Hub がスマートスピーカ対応の IoT デバイスを制御することが可能になれば，ホームオートメーションの幅が広がり，ユーザの QoL（Quality of Life）のをさらに向上することができる．

そこで本稿では，iHAC Hub とスマートスピーカを連携させることにより，人間が介在しない M2M デバイス連携手法を提案する．既存の iHAC フレームワークに，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し，スマートスピーカに向けて再生する機能を追加することで，スマートスピーカ対応デバイスの制御を実現する．

以下，2 章で従来の iHAC Hub と市販されているスマートスピーカについて示し，3 章で提案システムについて述べる．4 章では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，5 章ではプロトタイプシステムを用いた評価について述べ，6 章でまとめる．

2. 既存システム

2.1 iHAC Hub

iHAC Hub とは，規格の違いを意識することなく機器を直感的に制御できるシステムである iHAC（intuitive Home Appliance Control）システム [9] を応用し，温度や湿度などの環境情報に基づいた機器連携を実現する宅内用デバイスである．図 1 に iHAC Hub の構成を示す．iHAC Hub は，従来の iHAC システムから UI（User Interface）部を除いた iHAC フレームワーク部と各規格の通信処理部から構成される．発生する事象，ルールの発火条件，実行される操作を ECA ルール [10] に基づいて記述したレシピを定義し，iHAC フレームワーク内のレシピエンジンが解析し，機器連携のトリガとなる環境情報の取得や機器の状態を取得しレシピを満たしたか判断する．レシピの動作条件を満たした場合，各種通信処理部を呼び出すことで環境情報に基づいた機器連携が可能になる．例えば，IoT センサが計測した温度や湿度などの環境情報がある閾値を超えた場合に ECHONET Lite 対応エアコンを稼働させるように，人が制御に介在しない M2M システムを実現することができる．

しかし，既存の iHAC Hub はスマート家電向け通信プロトコルを搭載していないスマートスピーカ対応 IoT デバイスを制御することはできない．

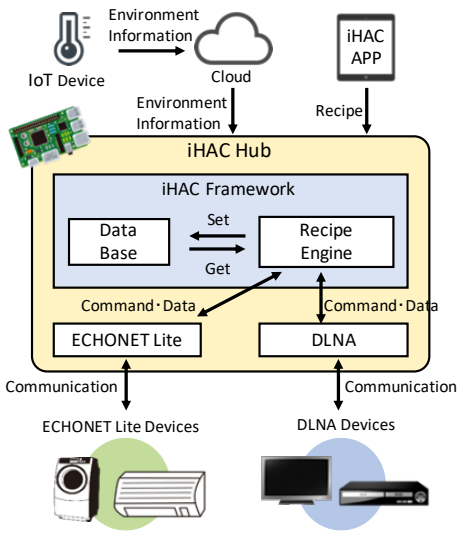

図 1 iHAC Hub の構成
Fig. 1 Configuration of iHAC Hub .

2.2 市販のスマートスピーカ

スマートスピーカとは，対話型の音声操作に対応した AI アシスタント機能を搭載した IoT デバイスであり，ユーザが話した音声を認識することにより，音楽の再生や検索，天気やニュースを読み上げたりするサービスを提供する．代表的なスマートスピーカとして，Amazon Echo や Google Home，LINE Clover などが安価に販売されている．また，スマートスピーカに搭載された AI アシスタントに対応したスマート家電や IoT デバイスも数多く販売されており，これらのデバイスの電源 ON/OFF やカーテンの開閉などの制御を音声操作で行うことが可能である[*1]．

図 2 にスマートスピーカを用いた音声による IoT デバイスの操作処理の流れを示す．まず，ユーザがスマートスピーカに対して “OK，Google” や “Alexa” などそれぞれのスマートスピーカに対応したウェイクワードと呼ばれる単語を発声し，その後に行いたい処理内容（例えば「Alexa，エアコンを 24 ℃に設定して」のように）を発話する．スマートスピーカはウェイクワードを認識するとユーザの発話した音声の録音を開始し，ユーザの質問やリクエストが

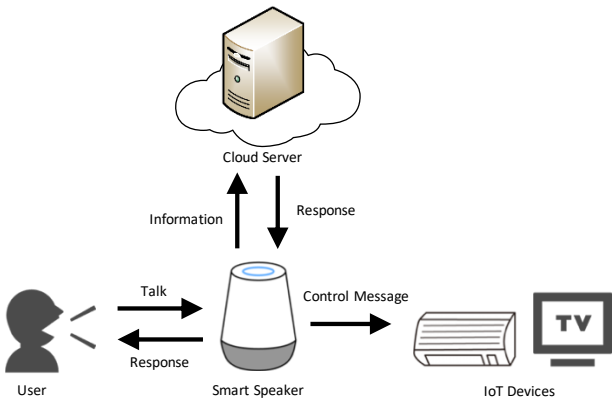

図 2 スマートスピーカを利用した IT サービスの概要
Fig. 2 Overview of the IT service using smart speaker.

[*1] LinkJapan 社の IoT スマートカーテン eCurtain https://linkjapan.co.jp/product/ecurtain/，富士通社のエアコン nocria AS-X22J https://www.fujitsu-general.com/jp/products/aircon/2019/lineup/nocria-x/asx22j.html など

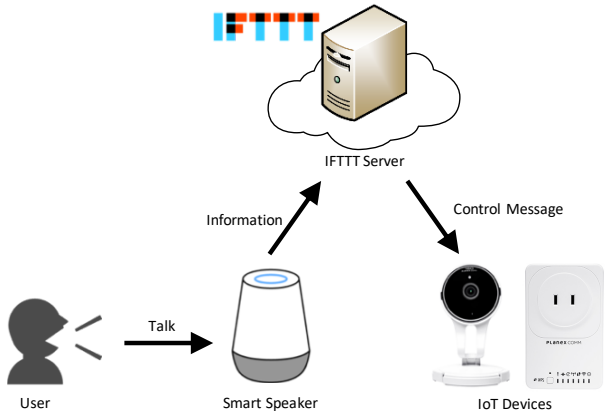

図 3　プラネックス・IFTTT・サービスを用いた機器連携の概要
Fig. 3　Overview of device cooperation method using Planex-IFTTT-Service.

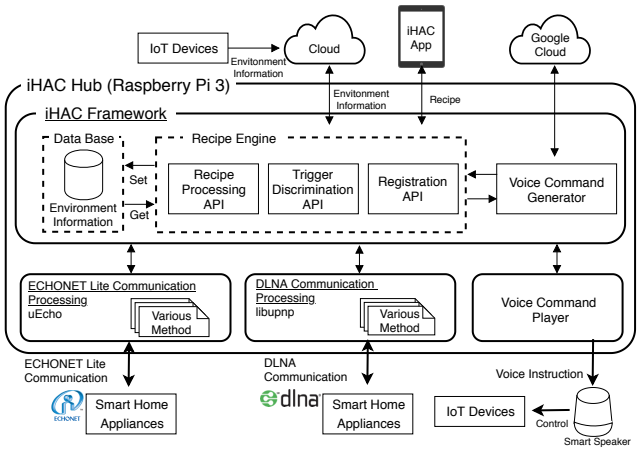

図 4　提案システムの概要
Fig. 4　Overview of the proposed system.

クラウドで処理されるまでの間の音声に限って，クラウドにストリーミングされる．クラウドの AI によって発話内容が音声認識され，言語化される．その後，ジョブの生成・管理が行われる．ここでは言語化された命令（例えば「エアコン」「24 ℃」などのキーワード）を処理することにより，機器の制御や音楽の再生など行うための命令を IoT デバイスに送信することにより，IoT デバイスの制御を実現している．

しかし，スマートスピーカとそれに対応した IoT デバイスから構成される現在の IoT システムは人間が制御命令を発話する必要があり，デバイスが自律的に協調動作する真の M2M システムの実現までには至っていない．

2.3　スマートスピーカによる機器連携サービス

スマートスピーカを用いた機器連携サービスの例として，プラネックス・IFTTT・サービスが存在する [11]．このサービスでは，異なるアプリや Web サービス同士を連携することができる IFTTT を用いて，スマートスピーカに話しかけることでプラネックス社の機器を制御することが可能となる．例えば，**図 3** のようにスマートスピーカに向けて「OK Google，見守りをスタートして」などの発話を行うことにより，IFTTT サーバからプラネックス社のスマートカメラに対して制御命令を送信し，見守りモードでカメラを起動するといった機器連携が可能となる．

しかしながら，プラネックス・IFTTT・サービスでは，機器連携のトリガとして人間が制御命令を発話しなければならず，複数台の機器を制御することは出来ない．

3.　提案システム

3.1　概要

iHAC Hub を用いてスマートスピーカ対応 IoT デバイスの機器連携を実現するためには，iHAC Hub とスマートスピーカとの間で情報交換するためのインタフェースを定義する必要がある．そこで，既存の iHAC フレームワークを拡張し，クライドサービスを利用して音声命令を生成し，再生する機能を新たに追加する．これにより，ある IoT デバイスの状態やセンシングした環境情報をトリガとして生成した音声命令を iHAC Hub が人間の代わりに発声し，スマートスピーカに IoT デバイスの制御を依頼することにより，M2M デバイス連携を実現する．

3.2　システム構成

図 4 に提案システムの概要を示す．従来の iHAC フレームワーク内にスマートスピーカに向けて IoT デバイスを制御するための音声命令を生成し再生する機能を追加する．

環境情報は IoT デバイスによりセンシングされ，クラウドに蓄積される．蓄積された環境情報を iHAC Hub が取得して処理を行うことにより，環境情報に基づいて様々な通信規格の機器連携を行う．

iHAC Hub で使用するレシピの記述方法は ECA ルールに則ったキーを用いて JSON（JavaScript Object Notation）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 Action から構成される．

3.3　システム動作

iHAC Hub を用いたスマートスピーカ対応デバイス連携の流れについて**図 5** を用いて説明する．まず，ユーザが iHAC システムを導入した iPad などの操作端末から UI を操作し，**図 6** のようにスマートスピーカ対応デバイスを用いた機器連携を行うレシピを作成する．作成したレシピの実行を選択すると，操作端末から iHAC Hub に ECA ルールに則り記述されたレシピ情報を送信する．

レシピ情報を受信した iHAC Hub はレシピエンジンを

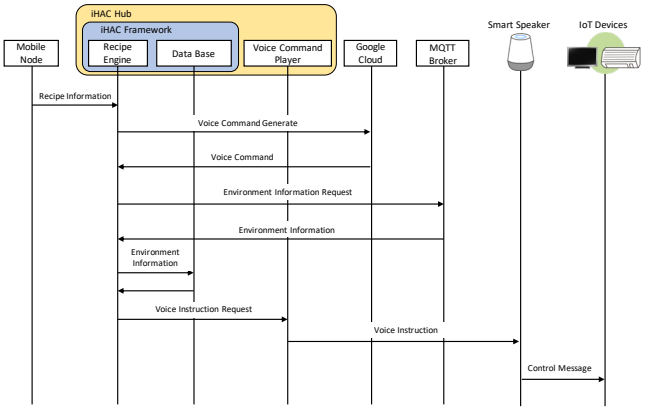

図 5 機器制御シーケンス
Fig. 5 Device control sequence.

```
{
  "recipeType":2,
  "event": [
    { "number": 1, "fetch": "MQTT", "provider": "m13.cloudmqtt.com",
      "mqttTopic": "living/temperature" }
  ],
  "condition": [
    { "number": 1, "operator": ">=", "value": 30.0 }
  ],
  "action": [
    { "controlDeviceId": 1, "controlMethod": "VoiceInstruction",
      "utterance": "Turn on the electric fan." }
  ]
}
```

図 6 ECA ルールに基づいたレシピ例
Fig. 6 Example of recipe based on ECA rule

起動し，解析用スレッドを稼働させレシピを解析する．レシピの解析が終了した後，レシピエンジンはクラウドサービスを用いてスマートスピーカ対応デバイスを連携動作させるための音声を生成し，“fetch キー” で指定された方法に従って環境情報を取得するためのバックグラウンドプロセスを稼働させる．バックグラウンドプロセスによって iHAC Hub は Subscriber とし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レシピエンジンによって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 は動作条件を満たすまでクラウドから環境情報を取得し，レシピ実行の判断を繰り返し行う．動作条件を満たした場合，スマートスピーカに対して機器連携を促す音声命令を再生し，機器連携を行う．

以上の処理により，他の IoT デバイスが計測した環境情報に基づいてスマートスピーカ対応 IoT デバイスを制御することが可能となる．

4. 実装および動作検証

4.1 実装

スマートスピーカ対応デバイスの連携を実現するために，Raspberry Pi 3 を用いて iHAC Hub のプロトタイプシステムを実装した．iHAC フレームワークを拡張し，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し，再生する機能を追加した．音声命令を生成するためのクラウドサービスは Google が提供している音声変換サービスである Google Cloud Text-to-Speech [12] を使用し，音声命令を生成する機能と，レシピの動作条件を満たした際に生成した音声命令を再生する機能を Python で実装した．なお，iHAC フレームワークは C 言語で開発されているため，C 言語から Python の関数を呼び出すようにした．

図 7 動作検証で使用した機器の一覧
Fig. 7 Devices used for verification.

レシピエンジンにより稼働され，MQTT を用いて環境情報を取得するバックグラウンドプロセスには C 言語で MQTT が使用できるライブラリである Mosquitto [13] を使用した．稼働したバックグラウンドプロセスは IoT デバイスによってセンシングされた環境情報が蓄積されるクラウドである CloudMQTT [14] へ使用するユーザ名とパスワード，受信したいトピック名などを送信し，Subscriber として環境情報の取得を行う．CloudMQTT から環境情報を取得する度に，取得した時刻を測定し MQTT で使用したトピック名と環境情報，時刻をレシピエンジンへ返しレシピの動作条件を満たしたか判断するようにした．

4.2 動作検証

スマートスピーカ対応デバイスの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 7** に動作検証で使用した機器の一覧を示す．自作した温度センサと iHAC Hub，Amazon Echo と Google Home の 2 種類のスマートスピーカ，両方のモデルをサポートする TP-Link 社の Wi-Fi スマートプラグである HS 105 を同一ネットワークに接続し動作検証を行う．なお，Wi-Fi スマートタップには通信機能を持たない扇風機を接続している．また，温度センサは Raspberry Pi 3 で作成し，10 秒ごとに室内の温度をセンシングし，センシングした温度を「`living/temperature`」というトピック名で CloudMQTT へ送信を行うようにプログラミングした．

動作検証の内容としては，ユーザが iHAC システムを導入した操作端末から UI を操作し，図 6 に示した「リビング

の温度が 30 ℃以上になったスマートスピーカに向けて “扇風機をつけて” と音声命令を再生し，扇風機をつける」というレシピを作成し，環境情報に基づいてスマートスピーカ対応機器の連携が可能か確認する．まず，自作した温度センサから CloudMQTT へ取得した温度の送信を開始する．その後，動作検証を行う iHAC Hub のプログラムを実行する．プログラムを実行すると，iHAC Hub が iHAC システムを導入した操作端末からレシピ情報を取得し，レシピエンジンによってレシピを解析できたものとし，「扇風機をつけて」という音声命令を Google Cloud Text-to-Speech API を用いて生成し，保存する．その後，環境情報をクラウドから取得するバックグラウンドプロセスを稼働する．バックグラウンドプロセスによってリビングの温度を取得するために「`living/temperature`」というトピック名を用いて CloudMQTT へアクセスし，Subscriber として温度の取得を開始する．温度センサーに対してドライヤーで熱風を当て，取得した温度が 30 ℃以上を計測した時，iHAC Hub がスマートスピーカに対して音声命令を再生して機器連携が行われるか確認した．

動作検証の結果，温度センサによってセンシングされた温度が 30 ℃以上となった際，iHAC Hub から Amazon Echo および Google Home に対して機器制御を促す音声命令が再生され，スマートタップが ON になり扇風機が起動した．この結果，提案手法により環境情報に基づいてスマートスピーカ対応デバイスの連携制御が可能であり，人が制御命令を行うことなくデバイス同士が自律的に連動する M2M システムを実現できることを確認した．

5. 評価

5.1 評価方法

Google Cloud Text-to-Speech の API を用いて生成された音声命令が，市販のスマートスピーカで正しく認識し，スマートスピーカ対応機器を制御可能か評価する．Google Clout Text-to-Speech の API を用いて音声命令する際には input，languageCode，name，ssmlGender，speakingRate，pitch のパラメータを設定できる．input には生成したい音声命令の内容を，languageCode にはどの言語で音声命令を生成するのかを言語コードと国コードの組み合わせで記述する．name では声色や訛りを，ssmlGender には男性か女性どちらの性別の音声命令を生成するか設定することが可能である．また，speakingRate は 0.25 から 4.00 の範囲で話す速度を，pitch は-20 から 20 の範囲で話すピッチを調整できる．本評価実験においては，**表 1** のような設定で speakingRate と pitch をそれぞれ変化させて生成した音声ファイルを Amazon Echo と Google Home の 2 種類のスマートスピーカに対して 10 回ずつ再生し，正しく機器制御できたか認識率の比較を行った．

表 1 Google Cloud Text-to-Speech の設定値
Table 1 Setting value of Google Cloud Text-to-Speech

Parameter	Value
input	Alexa, Turn on the electric fan.
	OK Google, Turn on the electric fan.
languageCode	en-US
name	en-US-Wavenet-A
ssmlGender	MALE
speakingRate	0.25–4.00 の間で 0.25 刻みで変化
pitch	-20–20 の間で 5 刻みで変化

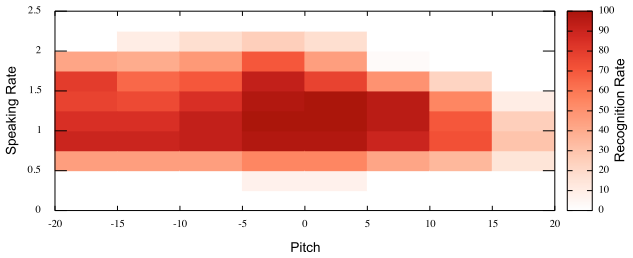

図 8 Amazon Echo の認識率
Fig. 8 Recognition rate of Amazon Echo.

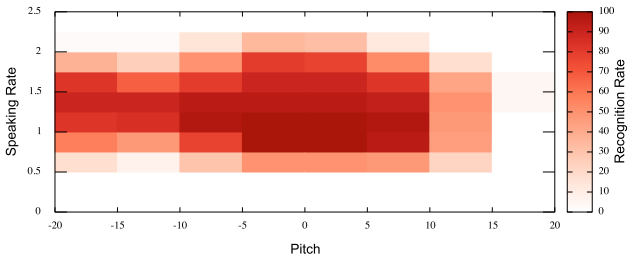

図 9 Google Home の認識率
Fig. 9 Recognition rate of Google Home.

5.2 結果

評価実験の結果を**図 8**，**図 9** に示す．まず，Amazon Echo の認識率について述べる．pitch の値が 0 の場合，speakingRate が 0.75，1，1.25，1.5，1.75 に設定した際，認識率は 100%となり，speakingRate の値が 1.0 の場合，pitch は-5，0，5 に設定した際，認識率が 100%となった．また，pitch が± 20 に近づくにつれ認識率は低下し，speakingRate が 0.25，および 2.25 以上の値を設定した場合，pitch の値を変更しても認識率は 0%のままであった．次に，Google Home の認識率について述べる．pitch の値が 0 の場合，speakingRate が 0.75，1，1.25 に設定した際，認識率が 100%となり，speakingRate の値が 1.0 の場合，pitch は-5，0，5 に設定した際，認識率が 100%となった．また，Amazon Echo の結果と同様に，pitch が± 20 に近づくにつれ認識率は低下し，speakingRate が 0.25，および 2.25 以上の値に設定した場合，pitch の値を変更しても認識率は 0%のままであった．

表 2　既存研究との比較
Table 2　Comparison with existing researches.

	柔軟性	人間の介在
市販のスマートスピーカ	△	◯
プラネックスサービス	×	◯
提案手法	◯	×

pitch が± 20 に近づくにつれて両スマートスピーカの認識率は低下し，speakingRate が 0.25，および 2.25 以上の値では pitch の値を変更しても認識率は 0%のままであった．この要因としては，pitch を変更したことにより，それぞれのウェイクワードが聞き取りにくくなったためであると考えられる．また，speakingRate が 0.25 の時，および 2.25 以上の値を設定した際の認識率が低くなる要因としては，speakingRate が 0.25 の場合は再生された音声が遅すぎ，ウェイクワードと認識されず，speakingRate が 2.25 以上の場合は再生された音声が早すぎ，スマートスピーカが音声命令を認識できなかったためであると考えられる．

以上より，Google Cloud Text-to-Speech で生成する音声は speakingRate が 1.0，1.25 かつ pitch が-5，0，5 で設定した場合，Amazon Echo および Google Home どちらのスマートスピーカでも認識率が 100%となるため，その範囲で音声命令を生成することで実用上問題ない精度で制御可能である．

5.3　既存システムとの比較評価

表 2 に提案システムと既存システムにおける，スマートスピーカを用いた機器連携の柔軟性と人間の介在の有無に関する比較を示す．市販されているスマートスピーカを用いた連携では，定型アクションを設定することで複数台の機器を制御することが可能であるが，機器連携を行う際には人がスマートスピーカに向けて発話する必要がある．プラネックス・IFTTT・サービスでは，複数台の機器を制御することはできない．また，市販されているスマートスピーカを用いた連携と同様に，機器連携を行う際には人がスマートスピーカに向けて発話しなければならない．

これに対して，提案手法では ECA ルールを導入し，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可能であり，機器連携の柔軟性がある．また，iHAC Hub が人間の代わりに発話することにより，人を介在せず機器連携を行うことが可能である．そのため，高い柔軟性を実現し人を介在せず機器連携可能な提案システムが優れていることがわかる．

6.　まとめ

本稿では，iHAC Hub におけるスマートスピーカ対応 IoT デバイスの連携手法について提案した．提案システムでは，iHAC Hub が IoT デバイスを制御するための音声命令をクラウドサービスを利用し生成し，スマートスピーカに向けて再生することで機器連携をが可能となる．

提案システムのプロトタイプを実装して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，Amazon Echo と Google Home の両スマートスピーカに対して Google Cloud Text-to-Speech で生成する音声は speakingRate が 1.0，1.25 かつ pitch が-5，0，5 の時に認識率も 100%となるため，その範囲で音声を生成することで実用上問題ない精度で制御可能である．

今後はホームネットワークに複数台のスマートスピーカと iHAC Hub が設置された場合の M2M デバイスの分散協調システムについて検討していく予定である．

参考文献

[1] Amazon: Amazon Echo. https://www.amazon.co.jp/dp/B071ZF5KCM?tag=googhydr-22&ref=pd_sl_83zrlgu7vj_e.
[2] Google: Google Home. https://store.google.com/product/google_home?hl=ja.
[3] 総務省：令和元年度版 情報通信白書のポイント，p.61 (2019).
[4] : IFTTT. https://ifttt.com/.
[5] Eve: Evehome. https://www.evehome.com/en.
[6] Hayashi, K. and Suzuki, H.: Cooperation Between Heterogeneous IoT Devices Using iHAC Hub, *Proc. of The 37th IEEE International Conference on Consumer Electronics (ICCE 2019)*, pp. 295–296 (2019).
[7] CONSORTIUM, E.: ECHONET Lite. https://echonet.jp/.
[8] Alliance, D. L. N.: DLNA. http://www.dlna.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] 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 (1995).
[11] プラネックスコミュニケーションズ株式会社：プラネックス・IFTTT・サービス. https://www.planex.co.jp/products/ifttt/.
[12] Google: Google Cloud Text-to-Speech. https://cloud.google.com/text-to-speech/?hl=JA.
[13] mosquitto.org: Mosquitto. https://mosquitto.org/.
[14] : Cloud MQTT. https://www.cloudmqtt.com/.