社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

SNSを介した情報家電・センサへの遠隔アクセスミドルウェア

大野　淳司†　　玉井　森彦†　　安本　慶一†

† 奈良先端科学技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
E-mail: †{atsushi-o,morihi-t,yasumoto}@is.naist.jp

あらまし　本稿では，種々のネット対応デバイスを様々なアプリケーションから統一的に利用するためのインタフェースを提供することを目的に，これらデバイスをSNSを介して容易に制御・監視できるようにするミドルウェアを提案する．本ミドルウェアでは，多種多様なデバイスに対応するために，ホームネットワークで利用されている各種プロトコル（UPnP, DLNA, ECHONET，IrDA）とのゲートウェイ機能を実現する．また，多種多様なアプリケーションに対応するために，本ミドルウェアの外部との通信をSNSを介した独自言語に基づく通信プロトコルに統一する．この独自言語によって，ホームネットワーク上の全てのデバイスの操作とセンサデータの取得を外部から容易かつ柔軟に実行できるようにする．提案するミドルウェアを実装し評価実験を行った結果，遠隔操作の応答性能が6秒以内程度と実運用に十分耐えうること，また，様々なアプリケーションを少ない工数で開発可能であることを確かめた．
キーワード　SNS，情報家電，遠隔操作，ミドルウェア，M2M

Middleware for Remotely Accessing Appliances and Sensors through SNS

Atsushi OHNO†, Morihiko TAMAI†, and Keiichi YASUMOTO†

† Graduate School of Information Science, Nara Institute of SCience and Technology　Takayama 8916–5, Ikoma, Nara, 630–0192 Japan
E-mail: †{atsushi-o,morihi-t,yasumoto}@is.naist.jp

Abstract　In this paper, we propose middleware which operates networked devices via SNS. The goal of the middleware is to provide a uniform and flexible interface to networked devices so that they can be easily operated from various applications. To adapt to various devices, the proposed middleware provides a gateway among various protocols used on the home network (UPnP, DLNA, ECHONET, IrDA, etc). To adapt to various applications, the middleware utilizes a text-based language for communication between the middleware and the applications via SNS. The language enables easy and flexible operations of devices on home networks. We have implemented the proposed middleware and confirmed that it achieves the practical response time within about 6 seconds in operating devices, and that various applications can be developed using the middleware with reasonably small man-hour.
Key words　SNS, information appliances, remote operation, middleware, M2M

1. はじめに

近年，インターネットに接続し，遠隔から制御・監視ができるデバイスやシステムの製品開発が盛んに行われている．しかし，このようなネット接続機能が提供されているデバイスはまだ少数派であり普及するには至っていない．また，ネットワーク経由で操作できる機能の範囲が限定されているなどの問題がある．さらに，操作のためのインタフェースもそれぞれ異なっており，ユーザは操作したいデバイス毎にアプリケーションを切り替える必要がある．このような問題に対して，様々なデバイスにおける実装やプロトコルの差異を吸収し，一つのインタフェースによる操作を可能とするゲートウェイシステムが幾つか提案されている [1]〜[3]．しかし，これらの既存システムは，(i) そのゲートウェイシステムを利用可能なアプリケーションが限定されている，(ii) イベント通知機能を持たない，などの問題を持ち，これらを全て解決したシステムは提案されていない．また，操作以外の実装例としては，ホームネットワーク上の家電の使用ログを監視し，異常を検出する見守りアプリケーション [4], [5] や，ホームオートメーションや分散監視アプリケーション向けのセンサネットワークゲートウェイ [6] などが挙げられるが，これらを一度に動作させたい場合はそれぞれのミドルウェアや機器を同時に導入する必要があり，導入コストが高くなるとともに，同時使用が考慮されていない場合は思わぬ不具合が発生する可能性がある．これらのことから，多種多様な

デバイス・アプリケーションに対応できる柔軟性を持った，家庭内のデバイス制御・監視ミドルウェアが必要であると考えられる．

本稿では，家庭内のデバイスを容易にネットワークへ接続し，それらの様々なアプリケーションへの統一的なインタフェースを提供するミドルウェアを提案する．多種多様なデバイスに対応するためには，これらが使用するネットワークやプロトコルの差異を吸収する必要がある．一方で，多種多様なアプリケーションへ対応できる能力も求められており，そのため，可能な限りデバイスを本来の通信手段・プロトコルを用いて操作する際の機能性・応答性を保つ必要がある．提案ミドルウェアでは，ユーザがホームネットワーク外から統一的なインタフェースで全てのデバイスにアクセスする手段として，Twitter などの SNS を用いることとする．また，プロトコルとして人が直接理解可能なテキストベースの独自言語を用いる．ミドルウェアとホームネットワーク内のデバイスとの通信は，UPnP（Universal Plug and Play）や DLNA，ECHONET，IrDA など，それぞれのデバイス本来の通信プロトコルを利用する．

本研究では，提案ミドルウェアの典型的な利用シナリオとして**家電の遠隔操作**と**見守り**，**家電使用状況データの収集**の 3 つのアプリケーションを想定し，これらのアプリケーションを実現するために必要な，(i) デバイスへのコマンドの送信・データの受信，(ii) 条件を満たした時のアクションの指定，(iii) ユーザへのデバイスの操作権限の付与，(iv) 要求に応じたセンサ情報の送信，等の機能をミドルウェアに組込む．

提案ミドルウェアを実装し，基本性能を調べる実験と，3 種類の利用シナリオに対応するアプリケーションの開発実験を行った．結果，レスポンスタイムに関しては 6 秒程度以内と実用十分な応答性能を持つこと，アプリケーション開発に関しては，種類を問わず十分少ない工数での開発が可能であることが分かった．

2. 関 連 研 究

遠隔地からデバイスを制御・監視する手段がいくつか提案されてきている．本章では，様々なデバイスをインターネットに接続するゲートウェイに関する研究，デバイスを SNS と接続する研究に分けて，関連研究を概説する．

2.1 デバイスをインターネットに繋げるゲートウェイの例

Khiyal らは，携帯電話の SMS（Short Message Service）を用いた HACS（Home Appliance Control System）と呼ばれるデバイスの遠隔制御・警報システムを提案した [7]．このシステムは，システムソフトウェアが動作しているパソコンと，SMS の送受信機能をシステムに提供する GSM モデム，そして SMS を実際に送受信するモバイルデバイスで構成されている．HACS はユーザに家電の遠隔制御機能と，不審者を発見した際の警報機能を提供する．ユーザとの対話は SMS を通じて行われ，システムは発信元の電話番号を用いて認証を行うことで，セキュリティを確保しつつテキストによるデバイス制御を実現している．しかしインタフェースが SMS に固定されており，また警報の発報条件は遠隔地から設定や変更ができない等柔軟性に欠ける設計となっているため，セキュリティは堅牢であるが，本研究の目的の一つである柔軟なアプリケーションへの対応を実現することは難しい．

三次らは UPnP や ZigBee で通信を行う家電やセンサデバイスを CoAP（Constrained Application Protocol）を用いて接続するゲートウェイを提案した [2]．CoAP は現在標準化策定中の，リソースに制約のあるインターネットデバイスでの使用を想定した Web 転送プロトコルである [8]．ユーザは遠隔地のサーバ等からゲートウェイに HTTP リクエストを送ることでセンサデバイスの監視や操作が可能である．しかし，このシステムは条件を設定しての通知が不可能なため，センサの監視を行いたい場合，アプリケーション側でセンサデータのポーリングを行う必要がある．また，認証機能を持たないなど，セキュリティ面が考慮されていない．

幾つかの企業から，自宅の家電やセンサデバイスをインターネットと接続するゲートウェイが発表され，実際に運用されている．パナソニックは，インターネットを介した対応機器の遠隔制御や，自動制御による節電を実現する AiSEG [9] を販売しているが，制御のできる範囲が限られている等の課題が存在する [10]．

2.2 デバイスと SNS を組み合わせた例

近年，ソーシャル・ネットワーキング・サービス（SNS）の普及により，デバイスを SNS に接続し，SNS を介してデバイスの状態を閲覧したり制御する手法が幾つか提案されている．

Kamilaris らは，Facebook をインフラとして利用し，スマートホームと Web を統合する手法を提案した [11]．この手法では，スマートホームのインターフェースとして Facebook アプリケーションが実装されており，これを介して情報の閲覧や制御が可能である．また，FacebookAPI を利用したアクセス制限も実装されており，セキュリティも確保されている．しかし，厳格なアクセス制限のため情報の拡散は困難であり，また，複数の個所に散在する情報を容易に収集することが難しい．

米澤らは，ワイヤレスセンサネットワーク（WSN）のセンサ情報に基づいたイベントを Twitter を介して容易に定義，共有するプラットフォームを提案した [12]．このプラットフォームでは，収集されたセンサ情報の状態に対して予め定義されたイベントが発生した時，自動的に Twitter へイベントが発生した旨を投稿し，共有する．また，イベントの定義を Twitter の投稿により行うことも可能である．これにより，ユーザは WSN 内にある監視対象の動向をシームレスかつ容易に定義，共有することが可能である．しかし，共有する情報は定義されたイベントのみであり，またデバイスの制御には対応していない．

Demirbas らは，Twitter をセンサやスマートフォンデバイス向けのオープンな投稿・購読インフラとして利用することにより，クラウドを介したセンシングとその活用を行うシステムを提案した [13]．このシステムにより，ユーザが Twitter にクエリを投稿することで，特定地域の現在の天気などについてクラウドソーシングを行ったり，デバイスが Twitter へ自動的に投稿するセンサ情報をユーザからのリクエストを受けて都度収集，解析することで，ユーザに結果を低遅延で返答する事が可

能である．しかし，このシステムはセキュリティを考慮しておらず，結果の信頼性を担保することが難しい．

また，近年では Twitter に投稿する機能を有した様々なセンサや家電が発売されている [14] が，これらは基本的に通知のためだけに使用され，公開された情報を利用したアプリケーションなどの提案はされていない．

3. SNS を介したデバイス遠隔操作ミドルウェア

本章では，提案するミドルウェアの目的および利用シナリオと，それを実現するための課題や必要な要件を述べた後，課題解決に向けたアイディアの詳細について述べる．

3.1 ミドルウェアの目的と利用シナリオ

本ミドルウェアは，以下の 3 つの利用シナリオ（アプリケーション）を想定している．

a） 家電の遠隔操作

近年，ネットワークに接続可能な家電や情報機器が普及しており，それらの機器をネットワーク経由で制御する枠組みとして UPnP や DLNA，ECHONET などが整備されつつある．しかし，これらの枠組みの間には互換性はなく，それぞれ専用のアプリケーションやインタフェースを介して操作を行う必要がある．また，対応機器は限られており，すべての機器がこれらの枠組みを使用して制御が行えるわけではない．例えば外出先から帰宅前に室温をチェックし，予めエアコンの電源を入れておくことや，録画予約忘れを思い出した時に外出先からレコーダを操作して録画予約や録画操作をすることなどが一元的に行えると便利である [1]．

本アプリケーションでは，異なる枠組みのデバイスを一貫して扱うための柔軟なインタフェースを提供する必要がある．例えば外出先から帰宅前に室温をチェックし，予めエアコンの電源を入れておくことや，録画予約忘れを思い出した時に外出先からレコーダを操作して録画予約や録画操作をすることなどを，一つのインタフェースから行うことができるようにする．そのためには，ユーザが SNS やメール等の，自身の一番使いやすい通信手段を選択し，インタフェースとして利用できることが望ましい．また，ネットワーク接続に対応していない従来型家電も制御できるようにするため，赤外線経由のデバイス制御機能も必要である．

家電の遠隔操作は，家人や，親類，その他指定した人のみが可能である必要がある．また，そのような人であっても，例えば小さな子どもがストーブの操作をしようとしている場合や，自宅に誰もいない状況で鍵を解錠しようとしている場合など，誤操作の危険性は常に存在する．本アプリケーションでは，そのような危険性を低減する仕組みの導入も必要である．

b） 見　守　り

近年，一人暮らしのお年寄りが誰にも看取られることなく亡くなる孤独死が社会問題となっている [15]．この問題に対して，公的機関による日常的な見守りなど様々な対策が実際に講じられているが，時間や範囲，頻度に限りがあり，十分ではない．ネットワークに接続された既存のデバイスを利用して，継続的かつ即時性のある見守りアプリケーションが実現できれば，孤独死の抑制に非常に効果があると考えられる．

本アプリケーションは，継続的かつ即時性のある見守りを実現するために，家の各所に設置してあるセンサの情報や，テレビやエアコン等のデバイスの動作状況を随時収集する必要がある．これにより，例えば室温が異常に高い（または低い）かつ人が在室している時にエアコンの動作を確認できない場合は，在室している人に異常がある可能性を検出できる．そのため，本アプリケーションの実現のためには，収集したセンサの値が異常かどうかを自動的に判断し，何らかのアクションを起こす機能が必要となる．柔軟性を持たせるために，異常を判定する条件や，その時実行するアクションを自由に設定・変更できるインタフェースも必要である．また，異常時以外でも見守りが実現できるように，ユーザ側からのクエリを受けて随時家電使用状況を返信できることが望ましい．

c） 効果的な節電のための家電使用状況の収集

東北地方太平洋沖地震以来，全国的な電力不足が続き，節電の必要性とそれに対するモチベーションが高まっている．また，ユビキタスコンピューティング技術を用いて，コンテキストの変化に合わせてデバイスを省エネ制御する方法が幾つか提案されてきたが，自動制御によって削減できる電力量は限定的であるため，省エネの為にユーザの自制を効果的に促す方法が求められている [16]．しかしながら，何の評価対象もない状態では節電に対するモチベーションを保つことは難しい．現在および過去の消費電力を可視化するシステムが多数提案されている [17], [18] が，得られるのは自宅の消費電力の値だけであり，比較対象は過去の自分の行動のみである．自分だけでなく，自分とつながりのある他者との比較が可能なアプリケーションが実現出来れば，より効果的な評価と競争によるモチベーションの維持，更には向上が見込める．また，収集したデバイスの使用パターンなどのビッグデータを解析することで，節電だけではなく快適性を向上させる提案（新たな家電の設置，リプレース等）やデバイス自身の性能の向上等に活用できる可能性も考えられる．

本アプリケーションを実現するためには，センサデータを，要求に応じたフォーマット・タイミングで送信する機能が必要である．アプリケーションによって，センサデータの取得間隔や，必要な統計値の種類は異なる．そのため，取得間隔と統計値の種類が外部から指定でき，かつそれに基づいて期間毎の統計値をホームネットワーク内で集計し，結果だけを送る機能が必要となる．また，ユーザのプライバシを保護する仕組みも同時に必要となる．

3.2 システム要件と基本方針

以上より，本ミドルウェアの目的を達成するための要件として，以下の 5 つを挙げる．

(1) 多種多様なデバイスと通信を行う
(2) 容易かつ柔軟に外部からアクセスできる
(3) 少ないコストで実装・利用できる
(4) デバイス本来の機能が制限されない
(5) 多種多様なアプリケーションへ応用できる能力を持つ

これらの要件を満たすための基本方針として，(1) の要件に

ついては，(i) ホームネットワークに接続されているデバイスについては，ホームネットワークを通じてそれぞれのプロトコルで通信を行い，(ii) ホームネットワークへ接続されていないデバイスについては，赤外線通信と無線通信のゲートウェイを用いて，間接的な制御を実現する．(2), (3) の要件を満たすため，本ミドルウェアでは様々な通信方式に共通のインタフェースを定義し，そのインタフェースを実装する形で各通信方式を使ったコマンドの送受信を実現する．これにより，メッセージの受信と受信の部分のみを実装するだけで容易に新しい通信方式を追加することができる．また，(4), (5) の要件を満たすために，テキストベースの独自言語を提案する．提案言語は，デバイス本来の通信媒体・コマンドを介したものと同等の操作性を実現しつつ，上記の利用シナリオで示したアプリケーションの容易な実現を可能とするものである．

上記の利用シナリオから得られた，それぞれのアプリケーションを実現する上での要件は以下である．

(5a)　デバイスの状態変化に応じたアクションの設定

(5b)　任意のユーザによる必要な粒度でのセンサデータの要求

(5a) の要件を満たすために，提案ミドルウェアはデバイスのデータが更新される毎にチェックを行い，設定された条件に合致した場合は任意のコマンドを実行するようにする．提案言語には，条件とその時の動作を設定するためのコマンドを含める．また，(5b) の要件を満たすために，提案言語にセンサデータ提供要求コマンドを含める．また，提案ミドルウェアは要求を受け，センサデータを収集し，指定された形式に合わせてデータを処理した結果をサブスクライバに向けて送信する．

3.3　操作性と応用性を併せ持つテキストベースの独自言語

前述したように，本ミドルウェアの実現のためには，デバイス本来の通信媒体・コマンドを介したものと同等の操作性を実現しつつ，多種多様なアプリケーションの実現を可能とするテキストベースの独自言語が必要となる．表 1 に，本節で用いる記号について示す．

表 1　記　号　表

記号	書式		意味
Room			部屋の名前
User			ユーザ名
Device			デバイス名
DeviceID	日本語	*Room* の *Device*	デバイスを一意に特定する識別子
	英語	*Room*'s *Device*	
Sensor			一般的なセンサ名
SensorID	*DeviceID*:*Sensor*		センサを一意に特定する識別子
Command			コマンド名
Role			権限
Term			条件文
Timespan			時間間隔
Security			セキュリティオプション
Function			統計関数

システム要件より，提案言語は以下の機能を持つ．

- デバイスの操作の実行
- センサデータの取得
- デバイスの状態変化に応じたアクションの設定
- 第三者からのセンサデータの提供要求
- ユーザ権限の変更

表 2 に，提案言語のコマンド一覧を示す．

表 2　コマンド一覧

	操作内容	コマンド	
		日本語	英語
手動遠隔操作	デバイスでコマンドを実行	*DeviceID* を *Command*	*Command DeviceID*
	デバイスの状態取得	*DeviceID* の状態？	*DeviceID*'s state?
	デバイスで実行できるコマンド一覧	*DeviceID* のコマンド一覧？	*DeviceID*'s command list?
	指定された場所にあるデバイス一覧	*Room* のデバイス一覧？	*Room*'s device list?
	場所一覧	部屋一覧？	Room list?
	別名を設定	*DeviceID* を *Command* = *Alias*	*Command Device* = *Alias*
	現在のセンサ値取得	*SensorID*？	*SensorID*?
	デバイスのセンサー覧	*DeviceID* のセンサー覧?	*DeviceID*'s sensor list?
自動遠隔操作	条件を満たした時にコマンドを実行	*Term* のとき *DeviceID* を *Command*	*Command DeviceID* when *Term*
	条件を満たした時に通知	*Term* のとき通知	Notify me when *Term*
	センサデータの提供要求	*Sensor* を *Timespan* 毎に購読（*Security*）	Subscribe *Sensor* every *Timespan*
	センサデータの統計値提供要求	*Function* を *Timespan* 毎に購読 (*Security*)	Subscribe *Function* every *Timespan*
セキュリティ	任意のユーザの権限を昇格	*User* の権限を *Role* に変更	Change power of *User* to *Role*
	任意のユーザにデバイス操作許可	*User* に *DeviceID* の操作を許可	Allows to operate *DeviceID* to *User*

UPnP デバイス等，ホームネットワークに接続されたデバイスに最初から設定されているデバイス名やコマンド名は，製品の型番や，その製品独特の識別子であり，ユーザがそのまま利用するのには適さない．提案言語では，部屋名やデバイス名，コマンド名は自由に命名できるようにすることで，定型文ながら，ユーザにとってわかりやすい文となるように工夫した．一方，部屋やデバイス，コマンドの名前のユーザによる命名により見た目の分かりやすさは向上するが，これらを全てユーザが記憶しておくことは依然難しい．この問題を解決するために，提案言語には部屋の一覧，部屋にあるデバイスの一覧，デバイスで実行できるコマンドの一覧，デバイスのセンサ一覧をそれぞれ参照できる機能を設け，ユーザの問い合わせに応じてそれらのリストを返すようにした．また，コマンドの実行には部屋名とデバイス名，コマンド名を列挙する必要があり，ユーザがコマンドを手入力する際は，入力にかかるコストや誤入力が問題となる．そのため，提案言語には別名（エイリアス）を設定する機能を持たせ，別名によって簡単にコマンドを実行できるようにした．

表 3　統計関数一覧

関数	意味	関数	意味
SUM	合計	AVERAGE	平均
MAX	最大	MIN	最小
VARP	分散	VAR	不偏分散
SKEW	修正された歪度	KURT	修正された尖度
FREQUENCY	累積度数		

センサの購読は，センサ値をそのまま取得するものと，統計値を計算した上で返すものの 2 つの手段を用意した．統計値を計算するための関数を容易に定義できるように，一般的に統計処理に使用されているいくつかの関数を用意した．定義済みの統計関数の一覧を表 3 に示す．また，それぞれの手段ではデータの取得間隔（*Timespan*）及びセキュリティオプション（*Security*）が指定できる．セキュリティオプションでは，例えば位置情報の有無や粒度などが指定でき，このオプションにしたがって提供するデータの k 匿名化が行われる．

3.4 コマンドの使用例

提案言語を使用したデバイス操作を具体的な例を用いて説明する．デバイスやセンサの配置状況を図 1 に示す．

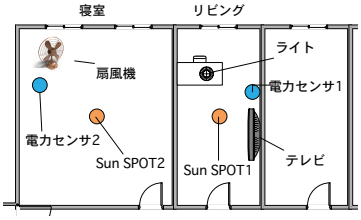

図 1　デバイスの配置

表 4　デバイス一覧

部屋	デバイス名	センサ	コマンド
リビング	Sun SPOT1	気温 照度	なし
	電力センサ 1	電力	なし
	ライト	なし	点灯/消灯
	テレビ	なし	ON/OFF 音量 UP/音量 DOWN
寝室	Sun SPOT2	気温 照度	なし
	電力センサ 2	電力	なし
	扇風機	なし	ON/OFF

また，各デバイスのコマンドとセンサー覧を表 4 に示す．以下で，各コマンドカテゴリ毎のコマンド例を挙げる．

(a) 手動遠隔操作コマンド

想定環境において，寝室にある扇風機を ON にするには，以下のコマンドを送信する．

> 寝室の扇風機を ON

コマンドを送信すると，そのコマンドの実行の成否が返される．このとき，権限の制限により管理者の確認が必要な場合は，管理者へ確認を求めるメッセージが送信される．上記のコマンドをユーザ *child* が送信し，かつそのコマンドが確認が必要であった場合の，管理者へ送信されるメッセージ例は以下である．

> child が”寝室の扇風機を ON” しようとしています．よろしいですか？

このメッセージに対し，管理者は”はい” 又は”いいえ” と返信し，コマンドの実行を制御することが可能である．

また，センサの値を参照したい場合は，デバイス ID の末尾にセンサ名を付与する．例えば，寝室の照度を取得したい場合のコマンドは以下のようになる．

> 寝室の SunSPOT2:照度？

記憶間違えやコマンドのタイプミス等で，合致するデバイスやセンサ，コマンドが見つからなかった場合は，候補のリストを返信する．例えば，上記のライト点灯コマンドにおいて，”寝室”を”ベッドルーム”と間違えた場合は，以下の様なメッセージが返る．

> 部屋が見つかりません: 次から選んでください（寝室，リビング）

(b) 自動遠隔操作コマンド

例えば，寝室の気温が 40 度以上になった時に通知をするように設定したい場合は以下の様なコマンドを送信する．

> 寝室の SunSPOT2:温度 >40 のとき通知

登録後，条件が満たされた時は次のような通知が送信される．

> 寝室の SunSPOT2:温度 >40 となりました

(c) セキュリティコマンド

例えば，ユーザ *child* がリビングのライトを操作することを許可したい場合は以下の様なコマンドを送信する．

> child にリビングのライトの操作を許可

4. ミドルウェアの設計と実装

本章では，前章で提案した基本方針，アイディアに基づいた家庭内デバイスの遠隔制御・監視ミドルウェアの設計と実装について述べる．

4.1 システム構成

提案ミドルウェアを使用したシステムの全体概要を図 2 に示す．システムは，ホームネットワークに接続されたネットワーク対応家電デバイスとホームサーバ，赤外線対応家電デバイス，無線センサノードから構成される．デバイスの制御や状態の参照，無線センサノードの情報参照を行う提案ミドルウェアは，ホームサーバで実行される．(図 2 参照) ミドルウェアは，各デ

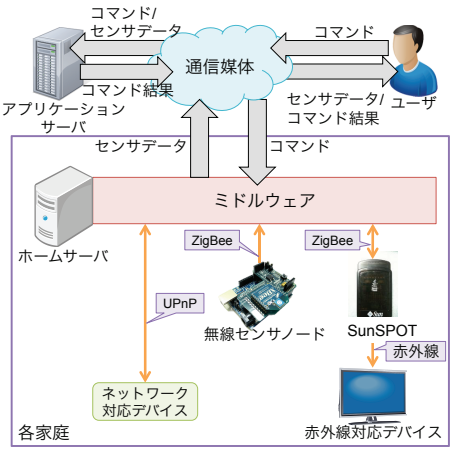

図 2　ミドルウェアの動作環境

バイスとユーザの選択した通信媒体からの情報を随時取得・格納し，必要に応じて通信媒体へのデバイスデータの投稿や，家電デバイスの制御を行う．また，外部に設置されたアプリケーションサーバが通信媒体を介して公開された情報を収集・蓄積し，様々なサービスに応用する．通信媒体はメールや Web ページ等，様々なものが考えられるが，今回はソーシャル・ネットワーキング・サービス（SNS）の 1 つである Twitter を用いる．これにより，(i) よりリアルタイムなイベント通知を受けられる，(ii) 緩やかな社会的つながりを利用したアプリケーションが実現できる，(iii) 情報拡散・収集が容易に可能である，といったメリットが享受できる．

4.2 家電デバイス・センサノードとの通信

提案システムは，ホームネットワークに接続されたデバイスの監視や制御を行うために UPnP を利用する．ミドルウェアは UPnP 対応デバイスと通信し，家電の使用状況の監視や制御を行う．また，UPnP に対応していないデバイスについても，そのデバイスが使用する通信プロトコルと UPnP のゲートウェイ機能を持つプログラムを作成し，UPnP を介して通信を行う．

ホームネットワークに対応していないデバイスの制御を行うために，赤外線の送受信機能を有した Sun SPOT [19], [20] を家の各所に配置して UPnP を介したデバイスの赤外線制御を行う．Sun SPOT のホストとデバイスは IEEE802.15.4 (ZigBee) を使用して通信しており，一つのホストに対して複数のデバイスが接続可能である．ホストとなる PC には UPnP を実装した Sun SPOT 通信プログラムが動作しており，UPnP を介して Sun SPOT に任意のコマンドを送信できる．

家の各所に例えば温度，湿度，照度，消費電力センサ等を備えた無線センサノードを配置し，センサ値を UPnP を用いて共有する．これらのノードの構成例としては，Arduino と各種センサ，XBee シールドの構成が挙げられ，センサ値は ZigBee を使用してホスト PC へ 10 秒毎に送信される．ホスト PC では受信したセンサ値をリアルタイムに更新・共有する UPnP プログラムが動作しており，UPnP を介してリアルタイムにセンサ値を取得することが可能である．

5. 提案ミドルウェアの評価

本章では，提案ミドルウェアの汎用性および有用性を示すために，前章で述べたミドルウェアを用いた評価を行う．まず，基本性能に関する実験及び評価を行い，その後，3 種類のミドルウェア利用シナリオに基づいたアプリケーション開発実験とその評価を行った．

5.1 実 験 内 容

実験には，4. 章で述べた提案ミドルウェアの実装を使用した．本実験で使用したホームサーバの詳細を表 5 に示す．

表 5　実験サーバの詳細

CPU	Intel Core i7 2600 3.40GHz
メモリ	12GB
OS	Microsoft Windows 7 Enterprise x64
データベースサーバ	Oracle MySQL 5.5
Java	Java SE 7 1.7.0_03

Sun SPOT 通信プログラムや無線センサノード通信プログラムも同一の実験サーバ上で動作している．ミドルウェアに接続されているデバイスの一覧および配置は，それぞれ，表 4，図 1 の通りである．

また，基礎実験においてデバイスが不足したため，実験用のダミー UPnP デバイスを作成した．ダミー UPnP デバイスは，実環境のデバイスに影響を及ぼさないこと以外は，その他のデバイスと同じ動作を行うように設計した．

実験 1 として，ミドルウェア基本性能について，以下の 2 つの観点におけるレスポンスタイムの実験及び評価を行う．

実験 1-1: 外部からのコマンド実行

本実験では，遠隔手動操作時の性能を確認するため，コマンドを送信してから実際にコマンドが実行されるまでの時間を測定する．時間の計測は，コマンドが送信された瞬間から開始し，ミドルウェア上でコマンドの実行が終了した時点で停止する．また，測定は，(i) コマンドが送信されてから，そのコマンドがミドルウェアで検出されるまでの時間，(ii) コマンドがミドルウェアに検出されてから，コマンドの内容が実際に実行完了するまでの時間，の 2 つの区切りで行う．この 2 つの値の合計値がミドルウェアのレスポンスタイムとなる．送信のタイミング，対象のデバイス，実行するコマンドを毎回ランダムに変更するために，専用のコマンド発行プログラムを使用する．デバイス増加によるパフォーマンスへの影響を考察するため，登録デバイス数を 1, 10, 50, 100 と変化させ，それぞれ 100 回コマンドを送信して計測を行う．

実験 1-2: 内部の状態変化によるコマンド実行

本実験では，遠隔自動操作時の性能を確認するため，ホームネットワーク内のデバイスの状態が変化し，設定された条件を満たした時から，条件に設定されたコマンドが実行されるまでの時間を測定する．時間の計測は，設定された条件を満たした瞬間から開始し，設定されたコマンドの実行が終了した時点で停止する．また，条件を満たすタイミングをランダムにするために，デバイスは全てダミー UPnP デバイスを使用し，ランダムなタイミングで状態変化を引き起こす補助プログラムを使用する．デバイスや条件の増加によるパフォーマンスへの影響を考察するため，登録デバイス数と 1 つのセンサに登録する条件数を 1, 10, 50, 100 と変化させ，それぞれ 100 回状態変化させて計測を行う．条件はそれぞれ重複しないものを登録する．

実験 2 として，提案ミドルウェアの有用性を示すために，いくつかの利用シナリオに基づいたアプリケーション開発を実際に行い，実装にかかった時間やソースコードの行数から評価を行う．アプリケーション開発実験は，全て同じ環境，プログラミング言語，被験者で行い，実装環境は，実験サーバと同一（表 5）とし，プログラミング言語は Java を使用する．被験者は，本実験で利用する Java 言語について，簡単な GUI アプリケーションが開発できる程度の知識は有しており，Java の標準ライブラリだけでなく，インターネット上に公開されているライブラリは自由に利用できることとする．また，ソースコードの行数は操作・監視を行う主要ロジック部分の行数のみを評価対象とし，GUI 等実際の動作には影響の少ない部分の行数は含

めないものとする．

アプリケーション開発実験に使用する利用シナリオは，3. 章で述べたアプリケーションにそれぞれ 1 つずつ対応したものを使用する．すなわち，(i) 家電の遠隔操作，(ii) 見守り，(iii) 消費電力データ収集（効果的な節電のための家電使用状況の収集），の 3 種類である．以下で，それぞれの種類に対応したシナリオの詳細を述べる．

実験 2-1: 家電の遠隔操作

実験環境におけるデバイス群を 1 つの画面で同時に操作が可能なリモコンアプリケーションを開発する．ただし，全てのデバイス名及びコマンド名は開発者に既知であり，新たなデバイス及びコマンドの追加は行われないこととする．

実験 2-2: 見守り

実験環境には，各部屋に温度センサが存在している．これを使い，リビングの気温が異常に高温になった場合，火事などの異常が発生したとしてユーザに通知を行う見守りアプリケーションを開発する．

実験 2-3: 消費電力データ収集

消費電力データを定期的に収集するアプリケーションを開発する．ここで，本ミドルウェアは Twitter にデータを送信する際，*#iotc* というハッシュタグを付加しており，ユーザはハッシュタグ *#iotc* を検索することで，任意のアカウントから発せられた情報を取得できることとする．また，位置情報は要求せず，1 時間毎の消費電力の平均値のみを取得する．

5.2 評価結果と考察

5.2.1 実験 1: 基本性能に関する評価結果

外部からのコマンド実行を行った時（実験 1-1）のレスポンスタイムの平均値を表 6 に示す．

表 6 遠隔手動操作レスポンスタイム計測結果 （ms）

デバイス数	コマンド取得まで	コマンド実行完了まで	計	最大
1	407.85	495.50	903.35	1420
10	425.19	509.78	934.97	1692
50	418.40	513.49	931.89	1342
100	408.39	542.64	951.03	1413

計測した 2 つの時間の区切りのうち，前者にはデバイス数との相関は見られなかった．これは，遠隔地からコマンドが送信されてから，それがミドルウェアに検出されるまではインターネットを経由しているため，インターネットの帯域や遅延が最も大きな影響を与えると考えられる．また，本研究では通信媒体として Twitter を使用している為，Twitter のサーバの動作状況からも大きな影響を受けると考えられる．本実験は，Twitter のサーバが正常に動作している環境下で行われ，その結果すべての試行において投稿から 1000 ms 未満でコマンドの検出に成功した．一方，後者はデバイス数の増加に伴い，緩やかに増加する傾向が見られた．これは，デバイス数が増加するとデータベースのレコード数も増加し，それに伴いレコードの検索にかかる時間も増加したためと考えられる．以上から，外部からコマンドを実行した時のレスポンスタイムの合計は，平均で 930.75 ms となり，最大でも 1692 ms となった．これは，本研究と同様に，様々なデバイスを統合し，それらを操作するための API を外部に提供する従来手法 [21] と比較しても遜色のないレスポンスタイムであると言える．

内部の状態変化によりコマンド実行を行った時（実験 1-2 のレスポンスタイムの平均値を表 7 に示す．

表 7 遠隔自動操作レスポンスタイム計測結果 （ms）

デバイス数＼条件数	1	10	50	100	最大
1	52.38	282.07	1331.64	3107.00	5462
10	41.69	262.04	1251.97	2771.57	4781
50	42.29	289.74	1290.43	2770.42	5312
100	38.40	299.80	1306.03	2773.28	4563
最大	287	876	2543	5462	5462

表 7 の結果が示すように，デバイス数が変化した際の遠隔自動操作のレスポンスタイムの差異は 337 ms 以内に収まり，デバイス数との相関は見られない．一方，1 つのセンサに対する条件数は，その増加に伴いレスポンスタイムの明らかな増加が見られた．これは，ミドルウェアがデバイスのセンサ値の変化を検出した際に，そのセンサに関連する全ての条件文を検査する事に起因すると考えられる．すなわち，1 つのセンサに関連づいた条件数が増えれば増えるほど，そのセンサの値の変化を検出した際に検査する条件文の数が増加し，レスポンスタイムの低下につながる．さらに，条件文を検査する際に，毎回条件文を構文解析し，評価を行なっていることも，レスポンスタイム低下の原因の 1 つとして考えられる．また，実験に使用したプログラムは，動作確認のために詳細なログをコンソール及びファイルに書きだすようになっている．条件数が増えると，1 度に何個もの条件がマッチし通知コマンドが実行される為，それに伴いログも一度に大量に出力され，その結果 IO の負荷によりレスポンスタイムが低下したとも考えられる．

しかしながら，実際の利用において，1 つのセンサに 100 以上の条件文が関連付けられることは考えづらい．また，たとえ 100 個の条件文が 1 つのセンサに関連付けられていても，レスポンスタイムは遅くとも 6000 ms 程度であり，定期的にデータを取得して，値をチェックする形式の従来手法と比較すると，遥かに高速であり，有用であると言える．

5.2.2 実験 2: 有用性に関する評価結果

それぞれの利用シナリオにおける実装時間やソースコードの行数を表 8 に示す．各利用シナリオのアプリケーションは全て同一の被験者によって作成された．また，全てのアプリケーションにおいて，通信媒体である Twitter へ簡単に接続する為に，外部ライブラリとして Twitter4J(注1) を利用した．表 8 の

表 8 アプリケーション開発コストの評価結果

利用シナリオ	実装に要した時間 (分)	ソースコードの行数 (行)
家電の遠隔操作	52	35
見守り	15	22
ビッグデータ収集	20	24

(注1) : http://twitter4j.org/

```
new TwitterFactory().getInstance().updateStatus("foo");
```

図 3　Twitter4J を使用したメッセージ送信

結果が示すように，全てのシナリオにおいて実装に要した時間は 1 時間以内に収まり，また主要ロジック部のソースコードの行数も最大で 35 行という結果となった．

また，実装に要した時間が一番長かった家電の遠隔操作シナリオについても，要した時間のうちの大半は GUI 部分の実装の時間であり，実際の操作を行う処理の実装にはあまり時間を要さなかった．その他の利用シナリオにおいても，主要ロジック部分の実装に要した時間は少ない．これは，本実験で使用した通信媒体である Twitter に容易に接続するための手段がライブラリ（Twitter4J）という形で提供されていたことが一因として挙げられる．例えば，Twitter4J を使用した Twitter へのメッセージ送信の実装は，図 3 に示すコードのみである．

以上の結果より，本ミドルウェアは多種多様なアプリケーションに利用できる汎用性，及び多種多様なアプリケーションを少ないコストで実装できる有用性を持つと言える．しかしながら，本実験で使用したシナリオでは，全て決まったデバイス・センサを対象としたものであったため，送信するコマンドを一部ハードコーディングしており，それにより開発時間やソースコードの行数が削減できたとも考えられる．そのため，更に汎用的なアプリケーションを開発する際は，コマンド文の生成においてさらなる開発コストがかかると考えられる．

6. おわりに

本稿では，様々なアプリケーションへの統一的なインタフェースを提供することを目的に，ホームネットワークに接続された情報家電やセンサを SNS を介して容易に制御・監視するミドルウェアを提案した．提案ミドルウェアでは，多種多様なデバイスに対応するために，ホームネットワークで利用される各種プロトコルとのゲートウェイ機能，多種多様なアプリケーションに対応するために，デバイスの操作やセンサ値の取得を SNS を介した独自言語に基づき行う機能を実現した．また，基本的な遠隔制御・監視に加え，見守りやビッグデータ収集といった異なるアプリケーションに対応できるよう，条件を設定しプッシュ型で情報を取得する機能，指定したデータをホームネットワーク内で統計処理した後に取得する機能も実現した．

提案ミドルウェアを実装し，基本性能の評価実験及び 3 種類の異なるアプリケーションの開発実験を行った．その結果，レスポンスタイムは，最大でも 6 秒程度となり，実運用に十分耐えうる応答性能を持つことが分かった．また，本ミドルウェアの API を用いることで，少ない工数で様々なアプリケーションを実現可能であることが分かった．

文　献

[1] 清川皓太，山本眞也，柴田直樹，安本慶一，伊藤　実，“3D 仮想空間を用いた情報家電のためのリモコンフレームワーク，” 情報処理学会論文誌，vol.52，no.2，pp.596–609，2011.

[2] J. Mitsugi, S. Yonemura, H. Hada, and T. Inaba, “Bridging UPnP and ZigBee with CoAP,” Proc. of the workshop on Internet of Things and Service Platforms (IoTSP ’11), pp.1–8, 2011.

[3] K.-S. Kim, C. Park, and J. Lee, “Internet Home Network Electrical Appliance Control on the Internet with the UPnP Expansion,” Proc. of Int’l. Conf. on Hybrid Information Technology 2006 (ICHIT ’06), pp.629–634, 2006.

[4] 新谷祐司，岡田康義，佐藤　直，“ホームネットワークにおける人間行動の機械学習に基づく異常検出，” 信学技報. ISEC（情報セキュリティ），vol.108，no.473，pp.23–28，2009.

[5] 石田和生，廣澤一輝，田村美保子，甲斐正義，“O-024 家電の利用状況モニタリングによる独居者安否見守りシステム (1)：全体概要と基本コンセプト，” 情報科学技術フォーラム講演論文集，vol.9，no.4，pp.539–540，2010.

[6] G. Song, Y. Zhou, W. Zhang, and A. Song, “A multi-interface gateway architecture for home automation networks,” IEEE Trans. on Consumer Electronics, vol.54, no.3, pp.1110–1113, 2008.

[7] M.S.H. Khiyal, A. Khan, and E. Shehzadi, “SMS based wireless home appliance control system (HACS) for automating appliances and security,” Issues in Informing Science and Information Technology, vol.6, pp.●●–●●, 2009.

[8] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, “Constrained application protocol (coap) draft-ietf-core-coap-13”. https://datatracker.ietf.org/doc/draft-ietf-core-coap/.

[9] Panasonic, “Panasonic aiseg”. http://sumai.panasonic.jp/hems/aiseg/.

[10] 井上理，“スマホでエアコン操作　パナソニック断念の不可思議，” 日本経済新聞，2012. http://www.nikkei.com/article/DGXBZO46180800V10C12A9000000/.

[11] A. Kamilaris and A. Pitsillides, “Social networking of the smart home,” Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on, pp.2632–2637, Sept. 2010.

[12] T. Yonezawa and H. Tokuda, “Twitthings: sharing, discovering and defining things’ happening using wireless sensor networks,” Proc. of Int’l. Conf. on Internet of Things 2010, pp.21–23, 2010.

[13] M. Demirbas, M.A. Bayir, C.G. Akcora, Y.S. Yilmaz, and H. Ferhatosmanoglu, “Crowd-sourced sensing and collaboration using twitter,” Proc. of 2010 IEEE Int’l. Symp. on World of Wireless Mobile and Multimedia Networks (WoWMoM 2010), pp.1–9, 2010.

[14] covia, “Wifi body scale”. http://www.covia.net/main/product-bodyscale.html.

[15] 内閣府，“平成 23 年版　高齢社会白書，” 2011. http://www8.cao.go.jp/kourei/whitepaper/w-2011/zenbun/23pdf_index.html.

[16] 安本慶一，小倉和也，山本眞也，伊藤　実，“快適度の低下を最小限に抑える省エネデバイス制御手法，” 情報処理学会研究報告，vol.2011-DPS-149，no.9，pp.1–8，2011.

[17] 株式会社エネゲート，“Smart ecowatt”. http://www.enegate.co.jp/smarteco_portal/.

[18] 埼広エンジニヤリング株式会社，“省エネルギー監視機器 ps03”. http://www.saikoh-e.co.jp/kaihatsu/kankyo03.html.

[19] Oracle, “Sunspot”. http://www.sunspotworld.com/.

[20] D. Simon and C. Cifuentes, “The squawk virtual machine: Java^{TM}on the bare metal,” Proc. of 20th annual ACM SIGPLAN Conf. on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA ’05), pp.150–151, 2005.

[21] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K.-i. Matsumoto, “Constructing home network systems and integrated services using legacy home appliances and web services,” International Journal of Web Services Research, vol.5, no.1, pp.82–98, 2008.