Vol.2013-GN-86 No.40
Vol.2013-CDS-6 No.40
2013/1/17

ホームゲートウェイ向け機器管理制御フレームワークの開発

宮田克也[†1,a)]　寺岡秀敏[†1,b)]　関原拓也[†2,c)]　芳野明彦[†2,d)]

概要: 各家庭には様々な機器があり，それらをホームゲートウェイ上で管理・制御する多様なアプリケーションの登場が期待される．しかし，各機器のネットワークへの接続方法，通信プロトコルが様々である等，制御方法が統一されていないことが課題である．本稿では，ホームゲートウェイ向けに，多様な機器を統一的に管理・制御するための機器管理制御フレームワークを提案する．本フレームワークは，OSGi 上に各アプリケーションが共通で必要とする機能を取り込み，機器制御用の共通 I/F，使い易い機器選択機能，登録すべき機器情報項目を取得するための共通 I/F を備えることで，アプリケーション開発量を削減できる．また，本フレームワークを利用して，試作 HEMS 環境を構築し，本フレームワークの有用性をアプリケーションの開発生産性の観点で評価した．

キーワード: OSGi，ネットワーク家電，ホームネットワークシステム，マルチデバイスシステム，HEMS

Development of HGW software framework for managing and controlling multi-devices

KATSUYA MIYATA[†1,a)]　HIDETOSHI TERAOKA[†1,b)]
TAKUYA SEKIHARA[†2,c)]　AKIHIKO YOSHINO[†2,d)]

Abstract: Various applications are expected to be proposed for a home network system (HNS). One of main subjects in developing applications which work on a home-gateway (HGW) in the HNS is that the applications must be available in various home environments, where a lot of types of devices are connected to the HGW and various connection methods and connection protocols are used. This paper proposes a method that constructs an OSGi-based software framework for managing and controlling multi-devices. Since the framework which works on the HGW includes common functions for applications, and provides common APIs for controlling devices, device selection features that is easy to use, and common APIs for acquiring the list of device properties be registered, then it can reduce development cost for the applications. We have implemented the proposed method, and deployed the experimental HEMS. We have also evaluated the effectiveness of the framework.

Keywords: OSGi, networked appliances, home network system, multi-device system, HEMS

1. はじめに

家電やセンサ等の宅内機器とネットワークを連携することで，機器単体では実現できない高付加価値サービスをユーザに提供するホームネットワークシステム（HNS）が注目を集めている．HNS では，宅内に配置されたホームゲートウェイ（HGW）が，宅内機器を制御したり，外部サーバと情報交換したりする．宅内消費電力の見える化や省電力制御を行うホーム・エネルギー・マネジメント・システム（HEMS），快適生活を支援するホームオートメーション，防犯やヘルスケアとしての見守りシステムといった多様な応用が考えられる．

しかしながら，HNS 向けアプリケーション開発の最大の課題は，制御対象機器の制御方法が統一されていないことである．通信プロトコルや伝送メディアは機器の種類によっても異なり，同種の機器であってもベンダ毎に異なる場合がある．また，家庭毎に使用する機器が異なり，長期間で見た場合には使用機器が追加／変更されることもある．そのため，統一的な方法で機器制御でき，機器構成の変化にも柔軟に対応できるシステムが望まれている．

一方で，様々デバイスやサービスに対応するためのホーム ICT 実行基盤として OSGi[1]が注目されている．OSGi は Java ベースのソフトウェアモジュール（バンドル）の動的更新を実現する基盤システムであり，ソフトウェアの部品化を効率的に実現することができる．家庭毎に使用する機器や，利用するサービスが異なる場合でも，必要なバンドルの組み合わせで適切なシステムを構築することが可能となる．

そこで本研究では，HNS において，アプリケーションの開発効率の向上を図るため，多様な機器を統一的に管理・制御するための機器管理制御フレームワークを OSGi 上で開発し，その評価を行った．

2. 先行研究

HNS の一般的な構成を図 1 に示す[2]．宅内機器を最終的に制御するのは宅内の HGW である．HGW 内の制御ロジックのみで宅内機器を制御する場合と，外部サーバや外部

†1 (株)日立製作所　横浜研究所
Hitachi Ltd.　Yokohama Research Laboratory
†2 (株)日立ソリューションズ
Hitachi Solutions, Ltd.
a) katsuya.miyata.yn@hitachi.com
b) hidetoshi.teraoka.rf@hitachi.com
c) takuya.sekihara.ew@hitachi-solutions.com
d) akihiko.yoshino.ev@hitachi-solutions.com

Vol.2013-GN-86 No.40
Vol.2013-CDS-6 No.40
2013/1/17

機器との情報交換に基づいて制御する場合とがある．

HNS において，機器種別，ネットワーク接続形態，通信プロトコルが異なる機器を統一的な方法で制御するという観点で，これまで様々な研究が行われてきた．例えば，アプリケーション開発負担を低減することを目的とし，HGW と外部機器間の I/F を共通化する研究[3]や，HGW 内部のアプリケーションと宅内機器制御モジュール間の I/F を共通化する研究[4]が行われている．

しかし，従来研究では，多数の宅内機器の中から制御対象機器を柔軟に選択する方法や，機器種別，ネットワーク接続形態，通信プロトコルが異なる機器の機器情報登録を簡易化する方法について十分な検討がなされていない．

本研究では，OSGi 搭載 HGW において，アプリケーションと宅内機器制御モジュール間 I/F の共通化のみに留まらず，制御機器選択や機器情報登録の観点でのアプリケーションの開発効率向上に寄与する機器管理制御フレームワークを提案する（図 1）．

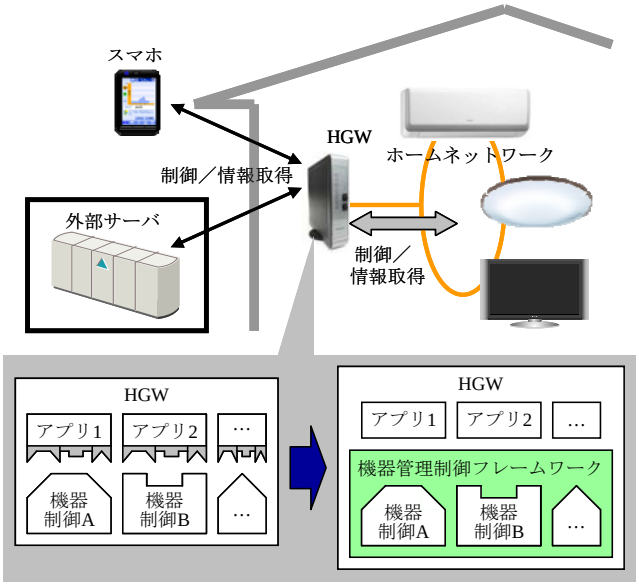

図 1　HNS 構成図

Figure 1　General architecture of HNS

3. 機器管理制御フレームワークの検討

3.1 要件定義

本稿では，機器管理制御フレームワークの要件として，以下の 3 つを定義した．

要件 R1：機器に対して制御を行う場合，接続形態，通信プロトコルによらず，同じ API で制御できること．

要件 R2：任意の制御対象機器を柔軟に選択できること．

要件 R3：異なる種別の機器を含む複数の機器情報を統一的に管理できること．

既に述べたとおり，各家庭で利用機器は異なり，機器によって通信プロトコルが異なる．また，機器の接続形態も複雑である．例えば，2 つ以上のネットワークがプロトコル変換アダプタを介して接続される場合もあれば，複数の機器を中継器で一旦集約して HGW と接続される場合もある．このような状況であっても，例えば，「宅内の消費電力が所定値を超えた場合に，全エアコンを停止する」というアプリケーションは，どの家庭のどのエアコンに対しても同様に動作する必要がある．要件 R1 を満たすことで，これを実現できる．

アプリケーションによって，制御対象機器は様々である．例えば，節電目的で機器の電源をオフする場合であっても，機器単体制御，「全エアコン」のような機種指定制御，「全リビング機器」のような設置場所指定制御，「宅内の全機器」のような全機器制御等が考えられる．このように柔軟に制御対象機器を選択できることが望ましい．要件 R2 を満たすことで，これを実現できる．

機器の種別／接続形態／通信プロトコルの多様性のため，一つの機器が保持すべき情報の多様化は避けられない．例えば，使用するアドレスは通信プロトコルによって異なるし，中継機を介して接続されていれば中継器の動作パラメータ情報が必要となる．そのため，少なくとも一度は，各機器情報を機器管理制御フレームワークに対して入力する必要がある．要件 R1 と同様の観点で，機器登録処理においても，機器の種別／接続形態／通信プロトコルが異なったとしても，統一的な方法で行えることが望ましい．要件 R3 を満たすことで，これを実現できる．

3.2 基本方針

OSGi の標準サービスの一つに Device Access サービス[5]がある．これは，例えば，機器が動的に追加された場合に，その機器のデバイスドライバを動的にネットワークからダウンロードするといった機能の実現を支援する．今回の機器管理制御フレームワーク開発においては，多種多様な機器と，実際に制御コマンド等を扱う制御モジュールとの対応付けが必要である．よって，Device Access サービスを利用するのが効率的であると考えた．

Device Access サービスは，物理的な機器を表すデバイスクラスと，最適デバイスドライバの検索機能の一部を担うドライバクラスとの対応付けを行う．基本的な動作の流れは次のとおりである．デバイスクラスのオブジェクトがサービス登録[6]されたことを検出した場合，その時点で登録されているドライバサービス（サービス登録されたドライバクラスのオブジェクト，クラスとサービスの関係については以下でも同様）について適合判定処理を行う．これにより最適なドライバサービスが確定した場合，そのドライバサービスのドライバ確定処理を呼ぶ．ドライバ確定処理の詳細は利用者が実装する．

今回，ドライバ確定処理の処理内容を 3.4 節にて検討した．また，3.5 節にてデバイスクラスを利用する実現方法を検討した．

3.3 制御インタフェースの抽象化の検討

(1) 機器制御 API の基本クラス構成

要件 R1 に対し, 機器制御 API を機器制御 I/F クラス(Java の interface class)と, 機器制御実装クラス(Java の implement class)とに切り分けた（図 2）. 具体例として，エアコン制御の場合，機器制御 I/F クラスでは電源 ON/OFF，動作モード変更，設定温度変更という抽象的な I/F を規定した．これにより，接続形態や通信プロトコルに非依存となる．一方, 機器制御実装クラスは, Echonet Lite[7]対応エアコン用，メーカ独自 I/F 対応エアコン用といった単位で別クラスとした．各クラスで通信プロトコル等の違いを考慮し，適切な制御コマンドを生成，送信する処理を実現する．

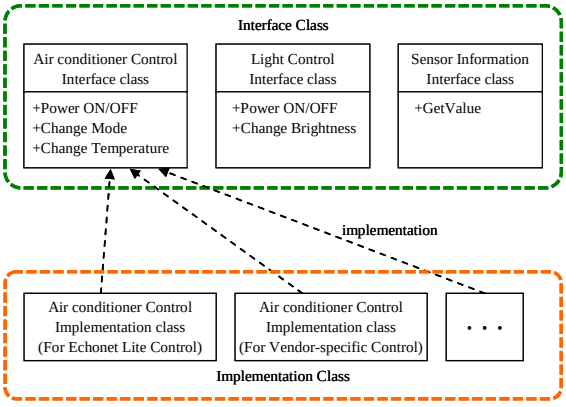

図 2　機器制御用 I/F クラスと実装クラスの関係図

Figure 2　Interface and Implementation class for device control

(2) 多段接続機器の機器制御実装クラスの実装方法

中継器を介して多段接続された機器について，機器制御実装クラスの実装方法を検討した．

（方法 1） 親機／子機を個別に表すクラスを設け，それらを組み合わせて一つの機器制御実装クラスとして表す方法（図 3）

個々の中継機に対応するクラスを生成する方法である．各中継機の実装クラスは，自分が受信できるプロトコル，アドレス，親機 ID 等の情報を含む．

HGW 上のアプリケーションは，末端の制御対象機器の制御メソッドを呼ぶ（処理 1）．そのメソッド実体である機器制御実装クラスの中で，プロトコル，アドレス，制御内容に応じたコマンド（指示）を生成する（処理 2）．そのコマンドを親機の機器制御実装クラスに渡す（処理 3）．制御対象機器の親機に当たる中継機クラスは，自分のプロトコル，アドレス，子機から渡されたコマンドに応じて，プロトコル変換を行い，新たなコマンドを生成する（処理 4）．これを繰り返すことで（処理 5，6），HGW が最終的に送信すべきコマンドを生成する（処理 7）．

以上のように，複数の中継機クラスを連動させることで，一つの機器制御実装クラスとしての役割をなす．

本方法の長所は，各中継機クラスを一度用意すれば，その組み合わせを変えても使用できるという拡張性が高い点である．逆に，短所は実装が複雑な点である．

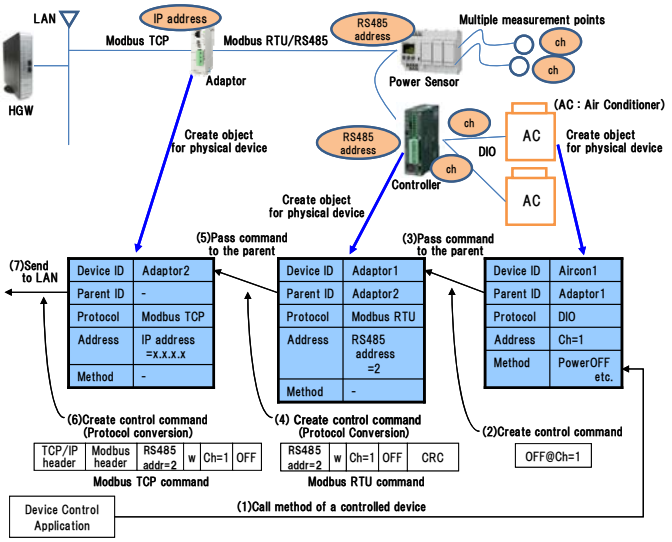

図 3　多段接続機器への制御コマンド生成方法（案 1）

Figure 3　Control command creation method for hierarchically connected device (1)

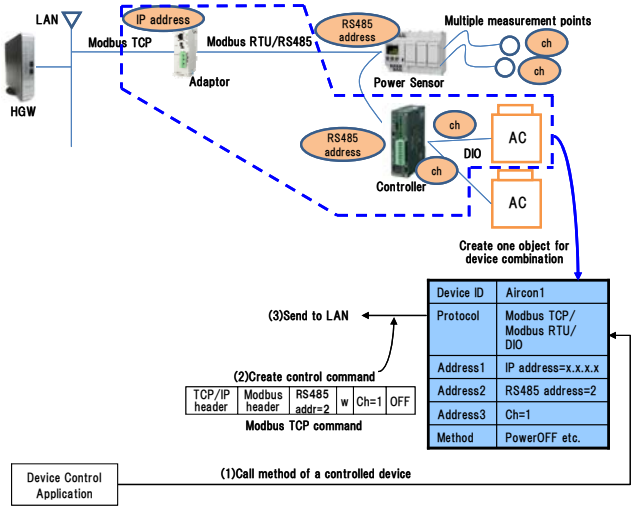

図 4　多段接続機器への制御コマンド生成方法（案 2）

Figure 4　Control command creation method for hierarchically connected device (2)

（方法 2） 親機／子機の関係を含めて一つの機器制御実装クラスとして表す方法（図 4）

HGW と末端の制御対象機器との間のつながりを，中継器も含めて一括りにして，一つの機器制御実装クラスとして実装する方法である．そのために，本機器制御実装クラスには，途中で経由するプロトコル，階層構造の中の各アドレス等の，最終的な制御コマンドを生成するために必要な情報を全て含める．

HGW 上のアプリケーションは，末端の制御対象機器のメソッドを呼ぶ（処理 1）．そのメソッド実体である機器制御実装クラスは，プロトコルやアドレスの階層関係を考慮

して，HGW が最終的に送信すべきコマンドを一度に生成する（処理 2）. HGW はそのコマンドを送信する（処理 3）.

本方法の長所は実装が比較的容易であること，短所は，親機／子機の組み合わせが一つでも変わった場合に新たな機器制御実装クラスの作成が必要な点である.

上記二つの方法を比較した場合，当面は親機／子機の組み合わせを自由に変えて使用することが少ない点を考慮し，方法 2 を採用することとした．なお，将来的に親機／子機の組み合わせパターンが増加し，方法 1 の方が有利になった時には，上記機器制御実装クラスの部分だけの修正で済むと考える.

(3)機器制御 I/F クラスの階層化

機器制御 I/F クラスを定義するに当たり，どの単位で機器制御 I/F を共通化すべきかを検討した.

基本的には機器の種類毎（エアコン，照明，テレビ，冷蔵庫等）に制御できる内容が異なる．よって，その単位で機器制御 I/F クラスを用意した．しかし，電源 ON/OFF 制御はほぼ全ての機器で実行可能と考えられる．よって，電源 ON/OFF 制御を行うメソッドを含む基本機器制御 I/F クラスを定義し，それを継承して（Java の extend），各機器用の制御 I/F クラスを定義した.

同じ種類の機器については，例えば，普及機と高級機とで機能に違いがあり, 両者の機器制御 I/F クラスは異なる．しかし，基本的には高級機は普及機の機能を含むと考えられる．よって，普及機用の機器制御 I/F クラスを継承して高級機用の機器制御 I/F クラスを定義した．これにより，高級機で追加された機能についての機器制御 I/F だけを記述すれば良い.

3.4 制御対象機器の選択方法の検討

要件 R2 に対して，3.3 節で検討した機器制御 I/F クラスを OSGi サービスとして登録することによって，機器制御 I/F サービスとして利用できるようにした.

具体的な処理の流れは以下のとおりである．ある機器が機器管理制御フレームワークに登録された場合（機器登録処理は 3.5 節参照），3.2 節で述べたように Device Access サービスが，その機器を表すデバイスサービスに最適なドライバサービスを検索し，ドライバ確定処理を行う．この処理の中で，その機器の制御機能を表す機器制御実装クラスのオブジェクトを生成する．それを，OSGi が提供するサービス登録機能を利用し，機器制御 I/F サービスとして登録することとした．その際，そのオブジェクトがサポートする全ての機器制御 I/F クラスをサービス登録することとした（図 5）．例えば，温度センサ付き高級エアコン用の機器制御実装クラスは，基本機器制御 I/F クラス，基本エアコン制御 I/F クラス，高級エアコン制御 I/F クラス，センサ情報 I/F クラスの全ての実装クラスであると言える．よって，このオブジェクトが生成された場合，上記全ての機器制御 I/F クラスをサービス登録する.

これにより，ある機器制御 I/F サービスを取得したいアプリケーションは，OSGi が提供するサービス取得機能[6]を利用する際に，所望の機器制御 I/F サービス名を渡すだけで済む.

この時，指定した機器制御 I/F サービスとして複数機器分が登録されていれば, それらを全て取得できる. よって，例えば, 基本エアコン制御 I/F サービスを取得することで，全エアコンの設定温度を一斉に 1 度下げるという制御ができる. また, 基本機器制御 I/F サービスを取得することで，全機器（エアコン，照明，テレビ，冷蔵庫等）を一斉に電源 OFF するという制御もできる.

また，より柔軟に制御対象機器の選択を行うために，機器制御 I/F クラスをサービス登録する際に，デバイスサービス情報をサービスプロパティとして登録するようにした. デバイスサービス情報は，Device Access サービスのドライバ確定処理にてドライバサービスに渡されるので，これを利用する．OSGi のサービス取得機能には強力な検索機能が備わっており，複数の条件を含む検索式で表せる．上記サービスプロパティに機器 ID，機器名称，設置場所，プロトコル種別等を含めることで，サービス取得時には，それらを使った検索式を指定するだけで簡単に所望の機器制御 I/F サービスを取得できる．例えば，設定場所を指定した機器制御や，機器 ID を指定した機器制御を簡単に行える.

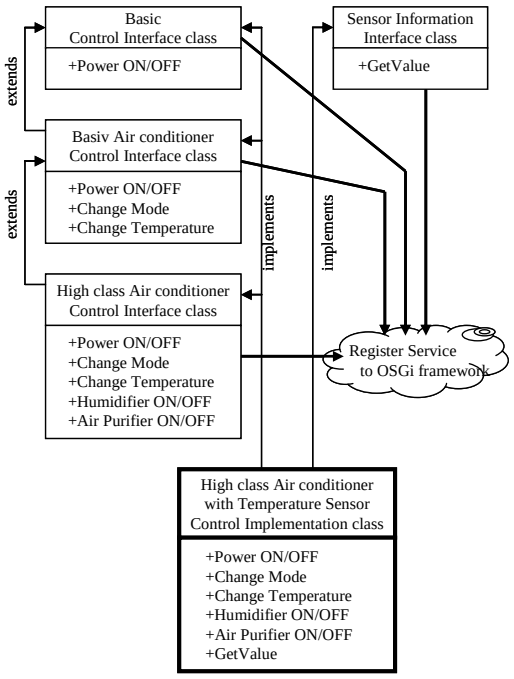

図 5　機器制御 I/F クラスの階層化

Figure 5　Hierarchical design of control interface class

Vol.2013-GN-86 No.40
Vol.2013-CDS-6 No.40
2013/1/17

3.5 機器情報登録方法の検討

要件 R3 に対して，機器 ID，機器名称，設置場所，プロトコル種別といった機器情報を表し，各情報へのアクセス API 機能を持つ機器情報クラスと，機器情報クラスの登録／削除／一覧管理を行う機器構成管理クラスを定義した．

機器情報には，機器 ID や機器名称といった制御対象機器自体に属する情報がある．一方，HGW と当該機器との接続方法や機器制御実装クラスの実装方法に依存する情報がある．例えば，IP アドレスや RS485 局番等のアドレス情報，使用する通信プロトコルによっては書き込みデータサイズ，中継器を介して接続される場合は中継器の動作パラメータである．これらは，機器制御実装クラスにて，機器制御コマンド生成処理の中で利用されるため，機器制御実装クラスに属する情報と言える．

そこで，制御対象機器に対応する機器制御実装クラスが一意に決まった場合に，その機器が保持すべき機器情報の項目一覧（これを「ドライバ情報」と呼ぶ）を管理するクラスを設けた．本ドライバ情報クラスを，ドライバ情報 I/F クラスとドライバ情報実装クラスとに切り分けた．これにより，ドライバ情報にアクセスする API を共通化し，かつ，機器制御実装クラス毎に異なる情報を扱うことができる．

以上の，機器情報クラス，機器構成管理クラス，ドライバ情報 I/F クラスをまとめて機器構成管理バンドルとした．ドライバ情報実装クラスは，前述の機器制御実装クラスとドライバクラスとまとめてドライババンドルとした．動作概要を以下に示す（図 6）．

(i) 機器登録を行うアプリケーションは，ドライバ情報 I/F クラス中のドライバ情報取得 API を呼び，ドライバ情報を取得する（処理 1，2）．この時，登録したい機器に対応するドライババンドルを特定する情報を引数で指定する．

(ii) アプリケーションは，取得したドライバ情報に対して，必要なパラメータを設定し（処理 3），機器構成管理クラス中の機器登録 API を呼ぶ（処理 4）．

(iii) 機器登録 API の処理の中で，機器情報クラスのオブジェクトを生成する（処理 5）．機器情報クラスは，Device Access サービスが認識できるように，デバイスクラスの実装クラスとしても実装し，これを OSGi のサービス取得機能を利用してデバイスサービスとして登録する（処理 6）．

(iv) 機器構成管理バンドルに登録された機器情報を取得したいアプリケーションは，OSGi のサービス取得機能を利用して機器情報サービスを取得する．機器情報サービスが提供する機器情報取得 API を呼ぶ（処理 7）．機器情報取得 API の処理の中で，指定された機器に関する情報を返す（処理 8）．

以上により，対応する機器制御実装クラスによって異なる機器情報を登録する必要があっても，アプリケーションは処理 1，2 で取得したドライバ情報に値を設定すれば良い．よって，機器情報登録において，全ての機器を統一的に取り扱うことができる．

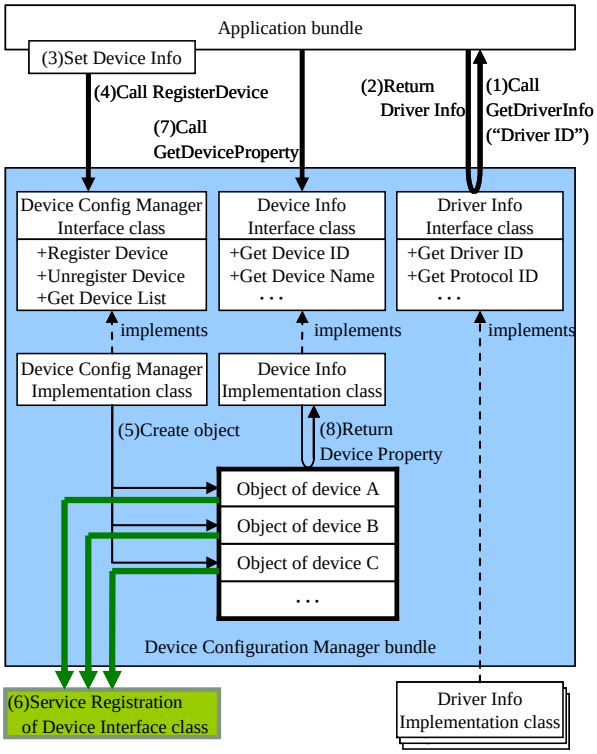

図 6　機器構成管理バンドルの構造

Figure 6　Structure of device configuration manager bundle

4. 試作システムの構築と評価

4.1 試作システム構成

今回設計した機器管理制御フレームワークの効果を確認するため，図 7 に示す試作 HEMS 環境を構築した．機器管理制御フレームワークを動作させる日立製 HGW の主要緒元を表 1 に，各制御対象機器の HGW との接続形態／通信プロトコルを表 2 に示す．

今回の試作システムで作成した機器管理制御フレームワークの全体バンドル構成を図 8 に示す．今回の制御対象機器 3 種（エアコン，照明，DTV）用と，電力センサ用の I/F バンドルを実装した．このバンドルには，各機器制御 I/F クラスが含まれる．機器ドライババンドルとしては機器種別と接続方法との組み合わせにより 8 個のバンドルを実装した．このバンドルには，ドライバクラス，機器制御実装クラス，ドライバ情報実装クラスが含まれる．通信プロトコルバンドルとしては 3 個のバンドルを実装した．このバンドルは，各通信プロトコル処理を行う機能であり，機器制御実装クラスから利用する．また，機器情報クラス，機器構成管理クラス，ドライバ情報 I/F クラスを含む機器構成管理バンドルを実装した．

表 1　評価機仕様

Table 1　Specification of HGW

Item		Description
CPU		650MHz
Memory		RAM:512MB, ROM:128MB
LAN/WAN interface	Port	RJ-45 4port(LAN), 1port(WAN)
	Supported Standard	10BASE-T/100BASE-TX /1000BASE-T
USB interface		2port
OS		Linux
Application Platform		JavaME(SuperJ Engine)/ OSGi Framework (SuperJ Engine Framework)

表 2　試作システムにおける制御対象機器一覧

Table 2　List of target devices in experimental HEMS

Device	Description
エアコン 1	・HA で制御（電源 ON/OFF のみ） ・Modbus 中継機経由で接続
エアコン 2	・独自プロトコルで制御 ・無線中継器 A 経由で接続
エアコン 3	・Echonet Lite で制御 ・試作 Echonet Lite アダプタ経由で接続
エアコン 4	・マルチ IR リモコンで制御 ・マルチ IR リモコンは LAN 及び独自プロトコルで HGW と接続
照明 1	・IR で制御 ・無線中継器 A とシリアル-IR 変換機経由で接続
照明 2	・マルチ IR リモコンで制御 ・マルチ IR リモコンは LAN 及び独自プロトコルで HGW と接続
DTV1	・独自プロトコルで制御 ・LAN 接続
DTV2	・マルチ IR リモコンで制御 ・マルチ IR リモコンは LAN 及び独自プロトコルで HGW と接続
電力量計 1	・Modbus 中継機経由で接続
電力量計 2-4	・無線中継器 B 経由で接続

4.2 要件の満足性に関する評価

今回実装した機器管理制御フレームワークが，各要件を満たすかを評価した．評価に際して，下記シナリオを実現するために，簡易アプリケーションを開発した．

(a)　集中リモコン制御

HGW に接続された宅内機器用の操作画面（GUI）を Servlet として公開する機能を持つアプリケーションを実装した．PC／タブレット／スマホ等のブラウザで上記操作画面を開き，ユーザが操作を行うことで，あらゆる接続機器の遠隔制御が可能となる．

(b)　電力ピークカット制御

電力量計の値を定期的に監視し，あらかじめ設定した最大許容値を超えた場合に，所定の機器を電源 OFF する機能を持つアプリケーションを実装した．所定機器の指定方法を変えて動作の違いを確認した．

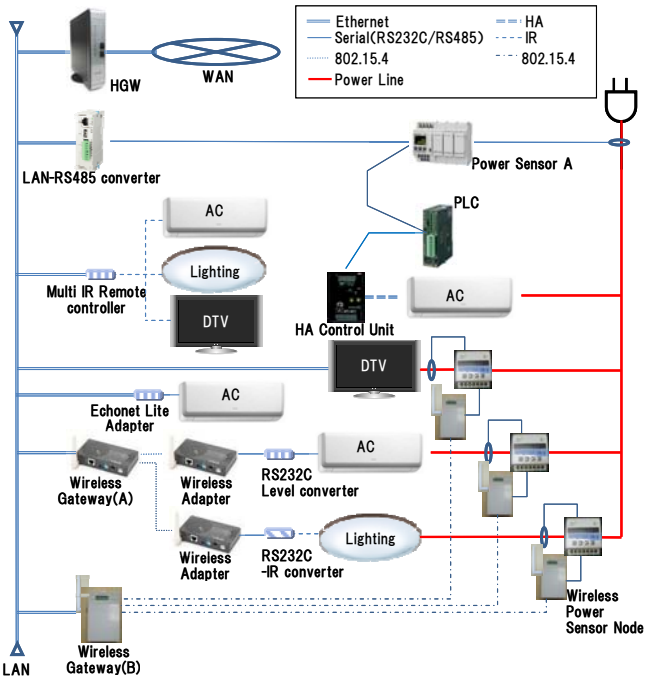

図 7　試作システム構成

Figure 7　Structure of experimental HEMS

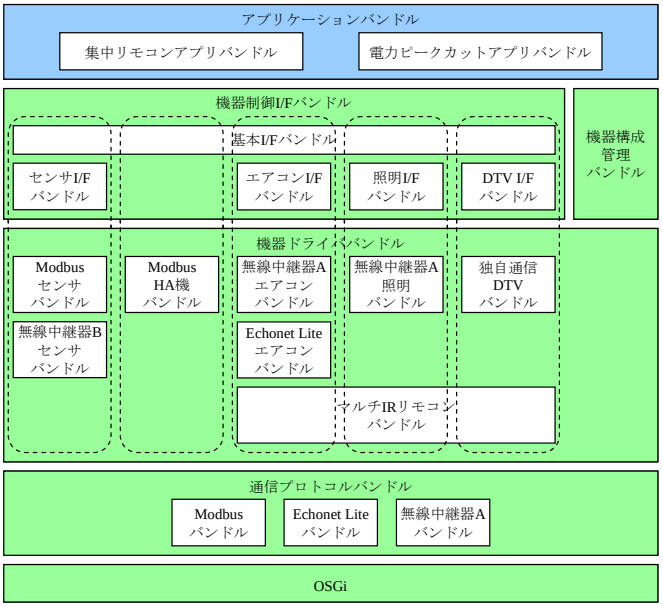

図 8　試作バンドル構成

Figure 8　Developed framework bundle structure

なお，機器一つずつの情報を GUI から入力させる機器登録アプリケーションを作成した．最初にドライババンドル種別を入力させた後，それ以外に登録すべき機器情報リストを提示し，ユーザに入力を促す．また，登録した機器情報一覧を表示する機能も持たせた．(a)，(b)のいずれの場合も，これを使って機器情報を事前に登録した．実際に登録した機器情報の一例を表 3 に示す．

(1)要件 R1

本試作システムでは，接続方法の異なるエアコンが 4 台接続されており，その内の 3 台は基本的なエアコン機能の

Vol.2013-GN-86 No.40
Vol.2013-CDS-6 No.40
2013/1/17

表 3　試作システムにおける機器情報（抜粋）

Table 3　Example of implemented device information

種別	内容
共通機器情報	・機器種別 ・ベンダ名 ・機器説明 ・ドライバ ID ・シリアル番号 ・機器名称 ・通信プロトコル ・設置場所
エアコン 2 ドライバ情報	・中継器親機 IP アドレス ・中継器子機 MAC アドレス
エアコン 4 ドライバ情報	・IP アドレス

制御が可能である（1 台は HA 制御のため電源 ON/OFF 制御しかできない）.

集中リモコン制御アプリケーションにおいて，上記 3 台のエアコンに対して，同じエアコン制御 I/F クラスを介して，電源 ON/OFF，モード変更（冷房／暖房／除湿等），設定温度変更の制御を行うことができた.

また，最終的に作り上げたのは図 8 のとおりであるが，それに至る段階では，対応機器を追加する度に，新たに機器ドライババンドルを追加することと，機器情報の登録を行うことが必要であった．しかし，アプリケーションの変更は全く不要であった．以上から，要件 R1 を満たすと考える.

(2)要件 R2

電力ピークカット制御アプリケーションにおいて，表 3 に示す共通機器情報を含む検索式を指定し，基本機器制御 I/F サービスを取得するようにした．これにより，“設置場所”や‘機器名称”で指定した機器を，想定どおりに電源 OFF 制御できることを確認した．以上から，要件 R2 を満たすと考える.

(3)要件 R3

本試作システムでは，表 3 に示すとおり，機器に応じて異なるドライバ情報を実装した．しかし，機器登録アプリケーションとしては，機器に関わらず同じドライバ情報 I/F クラスを利用することで，機器登録用 GUI を作成できた．また，登録機器情報を一覧表示する機能についても，共通の機器情報 I/F クラスを利用することで実現できた．以上から，要件 R3 を満たすと考える.

4.3 アプリケーション開発生産性に関する評価

機器管理制御フレームワークを利用した場合の，HGW 上のアプリケーションの開発生産性について評価した.

機器管理制御フレームワークを適用した場合のアプリケーション機能構成は図 9(b)のようになる．一方，今回は本フレームワークを利用しないシステムを開発していないため，図 9(a)のような機能構成を仮定した．以下に各機能の概要を記す.

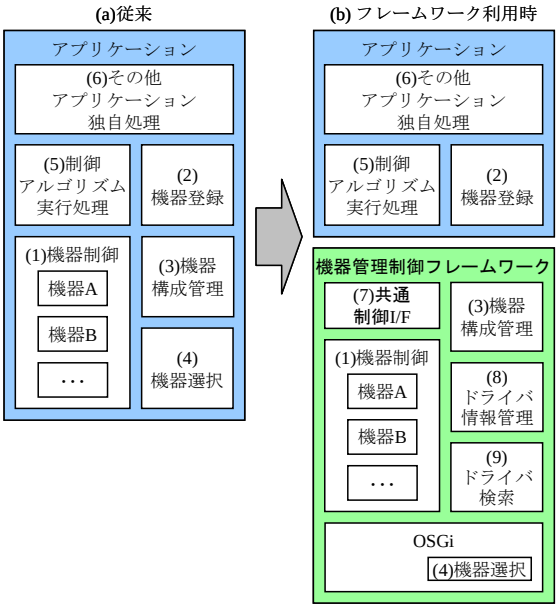

図 9　アプリケーションの機能構成の例

Figure 9　Example of structure of HGW application

① 機器制御実装クラスに相当する.
② HGW が制御可能な接続機器を登録する機能.
③ 機能②により登録された機器情報を保持する機能. 機器構成管理バンドルに相当する.
④ 機能③で保持する機器情報を参照して，所望の制御対象機器を抽出する機能．機能③の補助的機能.
⑤ 機能①と④を利用して，アプリケーションとしての機器制御アルゴリズムを実行する機能.
⑥ GUI 等の上記以外のアプリケーション機能.
⑦ 機器制御 I/F クラスに相当する.
⑧ ドライバ情報実装クラスに相当する.
⑨ 機器ドライバ検索機能. Device Access サービスを利用するためのドライバクラスに相当する.

図 9 の機能構成を踏まえ，機器管理制御フレームワーク（FW）利用時と，従来との開発規模の比較を表 4 に示す．開発規模は，従来を基準に表した.

(1) 共通機能のフレームワーク化による効果

従来のアプリケーション開発において最も負担が重かったのは機能①だと考えられる．S1 は，制御対象機器と，その接続形態との組み合わせ数に合わせて増加する部分である．今回の実績では一つの組み合わせ毎に 500～2000 ステップ程度であった．一つの家庭では，機器の種類や接続形態の種類は数個であっても，あらゆる家庭で利用可能なアプリケーションを開発するには，対応すべき種類数は増加する．しかも，全てのアプリケーションで同様の機能を実装する必要があった．この部分をフレームワークで担うことで，アプリケーションの開発工数を大幅に削減するこ

表 4　開発規模の比較
Table 4　Comparison of development scales

	従来		FW 利用時			
	アプリ	規模	アプリ	規模	FW	規模
①	○	S1	×	0	○	S1
②	○	S2	○	S2-A2	×	0
③	○	S3	×	0	○	S3+A3
④	○	S4	×	0	○	0
⑤	○	S5	○	S5-A5	×	0
⑥	○	S6	○	S6	×	0
⑦	×	0	×	0	○	S7
⑧	×	0	×	0	○	S8
⑨	×	0	×	0	○	S9

とができる．他に機能③，④についてもアプリケーションでの開発が不要となる．なお，S3 については，今回の開発実績から約 400 ステップ，S4 については概算で数百ステップと考える．

(2) 機器制御 I/F の共通化による効果

機能⑤では，従来は機器毎の独自 I/F を利用して機器制御する必要があった．本フレームワークを利用することで，機器毎の違いを意識せずに機器制御可能となる．その効果は，例えば，プロトコルに応じて制御コマンド ID やパラメータを単純変換する処理が不要となったとすれば，数十ステップ程度が削減可能となる．プロトコルによって関数の呼び出しシーケンスを使い分けていた処理が不要となれば，数百ステップ程度が削減可能と考える．この値が A5 に相当する．

(3) 共通機能のフレームワーク化による効果

機能②では，従来は機器の接続形態や通信プロトコルに応じて，機器情報として登録すべき項目を意識する必要があった．本フレームワークを利用することで，それが不要となる．その効果は，例えば，単純な項目毎条件分岐処理が不要になったとすれば，百ステップ程度が削減可能と考える．この値が A2 に相当する．

(4) 1 アプリケーション当たりの合計削減量

(1), (2), (3)より，アプリケーション一つ当たりの開発削減量は S1+S3+S4+A2+A5 となる．1 種類の機器を制御する場合は 1～3k ステップ程度削減できる．制御対象機器数が増える程，削減効果を大きくなる．

(5)フレームワーク開発のオーバーヘッドに関する考察

一方で，本フレームワークは，従来アプリケーションが担っていた機能①，③，④を含む．しかしながら，機能④の実現に当たっては OSGi の仕組みを利用することで，新規開発はほぼ 0 であると見なせる．また，機能⑦，⑧，⑨が新たに必要となった．今回の開発実績から，S7 は 20～50 ステップ，S8 は 50 ステップ程度，S9 は 50 ステップ程度であった．

以上より，アプリケーション開発の N 倍化による開発工数削減効果が最も見込めない N=1 の場合，アプリケーション及びフレームワークの合計の開発削減量は A2+A5+S4-S7-S8-S9 となる．つまり，この場合でも，A2+A5+S4>S7+S8+S9≒150 であれば本フレームワークを利用した方が開発生産性の点で有利であると言える．

5. おわりに

本研究では，HNS で利用される HGW 用アプリケーションの開発効率の向上を図るため，多様な機器を統一的に管理・制御するための機器管理制御フレームワークを OSGi 上で開発し，その評価を行った．

開発に当たっては，アプリケーションが宅内機器を制御するための共通 I/F を設けるだけでなく，OSGi のサービス検索機能を利用して制御対象機器を柔軟に選択する機能と，機器情報として登録すべき項目一覧を取得する共通 I/F を利用して，様々な機器情報の登録処理を統一的に行える機能を設けた．

また，開発した機器管理制御フレームワークを利用して，試作 HEMS システムを構築した．多種多様な機器を制御するアプリケーション開発において，制御対象機器の種別や接続形態を意識せずに実装できることを確認した．一つの制御機器を扱うアプリケーションであれば 1～3k ステップ程度削減できる見込みを得た．

今後は，最大収容可能機器数や処理速度等の性能評価を行う必要がある．また，クラウドサービスとの連携を想定した，HGW 外部からの制御機能の開発や，制御対象機器の拡大を検討して行きたい．

参考文献

1) OSGi Alliance, http://www.osgi.org/
2) 丹康雄「ホームネットワーク(OSGi, ECHONET)モデルに基づく家庭内エネルギーマネジメント」，情報処理学会 Vol.51 No.8, P959-965, 2010 年 8 月
3) 大和ハウス工業株式会社「平成 21 年度スマートハウス実証プロジェクト報告書 第 2 章 テーマ 2-1:マッシュアップを促進するホームサーバ向け統合 API の開発実証およびテーマ 3-1：マルチベンダによる家電・設備機器統合コントロールシステムの開発」，2010 年 3 月
4) 福岡祐介他「動的バインディング機構を用いたマルチベンダホームネットワークシステムの一実現手法」，信学技報 Vol.107 No.525, P295-300, 2008 年 3 月
5) OSGi Alliance「OSGi Service Platform Release4 Device Access Specification Version1.1」http://www.osgi.org/
6) OSGi Alliance「OSGi Service Platform Core Specification」http://www.osgi.org/
7) ECHONET コンソーシアム ECHONET Lite 規格書 Ver1.01（日本語版）http://www.echonet.gr.jp/spec/spec_v101_lite.htm