XMLを用いた遠隔機器制御支援システム

増渕 敬[†] 並木 美太郎[†]

近年，世界的なコンピュータネットワークの普及や通信機器の低価格化により，場所を問わずに特定のコンピュータに容易にアクセスできる環境が整備されつつある．しかし，現状では家電製品をはじめとするコンピュータ以外の機器へのネットワーク接続は遅れている．本研究では，対象となる機器のネットワーク機能の有無にかかわらず，任意の機器を共通のプラットフォームで操作することのできる環境を提案する．操作に必要なすべての情報を XML 文書を用いて構造化・抽象化し，汎用的なシステムを目標とする．本システムを用い，制御サーバを作成・配置することによってネットワークからの機器操作を容易に行うことができる．

Network Appliance Control Systems with XML

Takashi Masubuchi[†] and Mitaro Namiki[†]

In recent years, the spread of global computer networks and low pricing of communication devices have enabled access to a remote computer from anywhere. However, in the present condition, apparatus other than most computers, like home electronics devices, does not have a network function. In this research, the environment where arbitrary equipments can be operated on a common platform, irrespective of the existence of a network function, is proposed. Information required for operation is structuralized and abstracted using an XML document, and aims at the general-purpose system. By making this environment, it becomes possible to perform operation from a network easily by arranging a control server.

1. はじめに

現在，コンピュータネットワークは世界各地に張りめぐらされている．パーソナルコンピュータをみると，インターネットに接続する環境さえあれば，場所を問わずに特定の計算機へアクセスすることができる．しかしながら，家電製品などコンピュータ以外へのネットワーク接続は遅れているのが現状である．一部では，情報家電としてサーバ機能を持たせ，IP プロトコルを搭載してネットワークに接続して利用できる製品もあるが，互換性やコストなどの問題により，大半の機器がスタンドアロンで運用されている．これらの理由としては，組込み機器としてネットワークインタフェースやプロトコルを実装するためのコストが必要であることや，各社で仕様が共通化されていないことなどがあげられる．

そこで本論文では，計算機や IP を搭載した家電機器などのネットワーク機器をはじめ，ネットワークに対応していない家電製品や機器に対してもネットワーク経由での制御を可能にするための環境を設計し，操作環境を提供するサーバの構築について述べる．

† 東京農工大学工学研究科情報コミュニケーション工学専攻
Graduate School of Engineering, Tokyo University of Agriculture and Technology

2. システムの目標と概要

2.1 システムの目標

本節では，システムの設計目標を満たすために必要となる事項について述べる．

(1) サービス統合型プラットフォームの実現

本システムでは，独立した機器がそれぞれ提供するサービスを仮想的にまとめ，一連のまとまったサービスとして提供する統合型プラットフォームを実現する．たとえば，リモコンを動作させての家電製品の遠隔操作や，TCP/IP プロトコルで接続されている機器からの情報取得を，Web 上で行える 1 つのサービスとしてサーバが提供する．これにより，機能を提供するサーバにアクセスするだけで，接続された複数の機器を同じプラットフォーム上でアクセスおよび管理することが可能になる．

(2) IP ネットワークを用いて，インターネットからの容易なアクセスを可能にする

サーバの接続には IP ネットワークを使用する．これにより，ネットワークが利用できる場所すべてから

のアクセスが可能になる．特に，WWW にアクセス可能な携帯電話など移動体通信からのアクセスができることで，利便性が大幅に向上する．操作系は Web をベースとし，ネットワークブラウジングが可能な機器であればクライアントとなりアクセスすることができる設計を行う．

(3) **ターゲット機器のパラメータや処理の定義と記述を可能にする**

既存の制御システムは，設計時に設定された機能だけが利用でき，それ以外の用途に使うのことは困難である．また，対象となるハードウェアに多種の機能を組み込むことは，開発・生産のコスト増大につながるほか，すべての機器を同じ制御規格で揃えなくてはならず，環境を構築することは難しい．そこで本システムでは，拡張ポートや赤外線受光部などの外部インタフェースや TCP/IP を持っている機器であれば，それらを通し，機器に手を加えることなく制御機能を加えることができることを 1 つの目標にする．

制御サーバには，どのように外部から機器に働きかければ操作ができるのかといった情報を記述しておくことで，利用できる機能をシステムの利用者が定義できるという特長を持たせる．その情報の内容としては，機器に固有な定数・パラメータといったデータと，外部からの通信方法の 2 つが必要となる．記述の自由度をできるだけ多くとり，拡張性を高める目的から，データは XML 文書にて記述，通信方法に関しては制御スクリプト言語を設計すると同時に Java 言語によるプログラムとしても記述できるようにした．XML を用いることで，構造的なデータ記述ができ，システムが利用するデータはもちろん，ユーザインタフェースも同じ様式で記述ができるといった利点が得られる．

2.2 既存の制御システムとの比較

Jini[1] は，パソコンや周辺機器をはじめ，電化製品に至るまで様々な機器をネットワークを通じて相互に接続するための技術仕様である．Java 技術を基盤としているため，特定の OS や MPU に依存せず，対応機器は，ネットワークにつなぐだけで複雑な設定をすることなくすぐに機能する．Jini は，機器がネットワークに接続されると，discoveryAPI により，サービス内容を登録する lookup サービスが検索され，joinAPI によってサービスに関するオブジェクトが登録される．クライアントは lookup サービスを通してサービスオブジェクトを取得し，取得したオブジェクトを用いて対象機器と直接通信を行えるという特徴がある．

また，Jini と同様の技術として，HAVi[2] があげられる．HAVi は，IEEE1394 を用いて AV 機器を中心とする家電製品どうしの相互接続を実現するための，API とプロトコルを定めた規格である．

インプローブネットワークスの Cafemoon@HOME[3] は，Jini や HAVi，Bluetooth といったあらかじめ存在するプラットフォームを用いて制御を実現する．対象機器に関する情報は，ホームサーバにデバイスドライバという形で存在し，サーバは対象機器のデバイスドライバを通し，各プラットフォームのネイティブドライバに制御命令を発行する．これにより，対象機器のプラットフォームを問わずに，ドライバを用意することで一貫した操作体系を提供することができる．

Jini の問題は，接続される機器には Jini のサービスオブジェクトが組み込まれている必要があり，対応していない機器は利用ができない点である．同様に HAVi においても，機器に HAVi に対応したネットワーク機能が搭載されていなければならない．一方，Cafemoon@HOME は，ドライバが Java 言語で記述されており，特定の機器に対するドライバを利用者が作るのが容易ではないこと，稼働のためには制御のために Cafemoon@HOME サーバのほか，別途ゲートウェイサーバが必要である．これに対し本システムでは，ネットワークに対応していない機器においても，処理を記述するためのスクリプト言語を用意して処理内容を容易に記述することができるのに加え，高度な処理に対しては Java クラスを呼び出すためのフレームワークを提供する．また，サーバ上での機器制御に加え，ゲートウェイ機能を持たせることもできる．

一方，制御以外のサービス分野では Web サービスが主なものとしてあげられる．Web サービスの記述言語としては，SOAP，WSDL[4),7)] といったものが存在する．インタフェースが分かっていないサービスを正しく呼び出すためには，与えるパラメータとその内容などを取得しなくてはならない．そのため，クライアントはサービスの内容を WSDL を用いて取得する．詳細なやりとりの内容を理解したクライアントは，実際に通信を行うインタフェースを作成し，SOAP により通信を行う．この方式に比べ，本システムでは，サービスの内容は一括管理されているため，クライアントがアクセスのつど，対象機器のインタフェースを取得する必要がない点などが利点としてあげられる．

その他，電話回線を用いて制御を行う試みとしては，富士通研究所の MARON-1[5]，ダイヤルトーンを用いた計算機アクセスシステム[6] といった研究があげられる．また，NTT コムウェアの L-BOX[8] といった情報家電向けのサーバが存在するが，本システムには，これらの装置のように対象機器と機能を限定せず，一

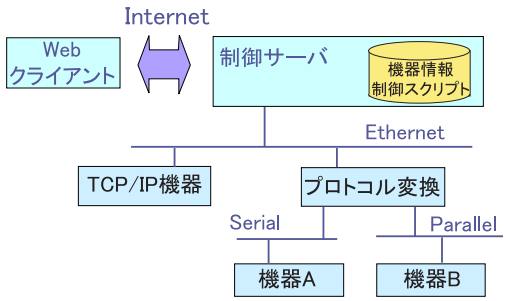

図 1 システム構成図
Fig. 1 The system structure.

貫した記述方法で複数の操作方法を持つ機器の定義ができるという利点がある．

3. システム構成

柔軟なシステムを構築するために，クライアントが直接個々の機器に接続して操作を行うのではなく，対象をサーバで一括して管理する．そこで本システムでは，機器群を管理するサーバをシステムの中心とし，その下にツリー状に機器を接続していく形をとる．ユーザインタフェースの生成，個々の機器への対応を 1 つのサーバが行うことにより，機種依存性の少ないシステム構築が可能となる．

3.1 システム構成

システム構成図を**図 1** に示す．一般利用者（システムを利用して実際に機器の操作を行うエンドユーザ）はブラウザを用い，Web サーバに対してネットワーク経由でアクセスを行う．このとき，利用者へは操作画面が提供され，必要な動作を選択してサーバに送信する．

リクエストを受け取ると，サーバは対象機器を操作するために必要な情報を機器定義文書より取り出し，定義された通信手順によって実際に機器と通信を行う．このとき，対象がネットワーク対応機器であれば，TCP/UDP プロトコルでそのまま通信を行い，対応していない機器の場合は，TCP とシリアルのように，プロトコル変換を行う機器を経由し，目的のインタフェースに適合する信号で通信を行う．機器からの応答があった場合は，一般利用者およびシステム構築者は応答を Web サーバを通して受け取る．

また，このほかにも対応したプロトコル変換のためのインタフェースを用意できれば，今回の設計で考慮したもの以外の専用プロトコルを持つネットワーク機器の操作も可能である．

3.2 機器記述とモジュール構成

現在あるシステムの問題として，サービスの記述が使用する機器に依存しているという点がある．そこで，本システムでは様々な属性を持っている機器を，システム構築者が一貫した方法で記述できるようにする．

まず，記述の前提として，サーバと対象機器間の通信内容および手順が一定ではないことがあげられる．このため，任意の機器と通信をさせるためには，対象が何であるか，どのようなコマンドを用いて通信を行えばよいかといった事項が重要になる．そこで，このために必要な記述を以下のように分類する．

(1) 機器の特性に関する記述

任意の機器を操作するために，まず操作対象となる機器に固有なパラメータを記述する必要がある．例として，対象が Web 対応機器であれば TCP/IP でリクエストを送信するためのコマンドやファイル名が，非ネットワーク機器であれば，プロトコル変換を行う装置に命令を送信することになるが，その際のコマンドなどを定義する必要がある．例をあげると，以下のようなものが機器に固有なパラメータとして考えられる．

- Web サーバへ送信するリクエスト文
- プロトコル変換装置に対するコマンド
- 機器からのリプライより結果を生成するための，四則演算や文字列連結などの数式

ここでは通信文を構成する要素や，機器ごとに異なる応答内容の取り出し方法といった，機器に固有な内容を定義し，実際の操作手順については次の (2) で記述する．

(2) 機器の操作手順に関する記述

サーバに接続された機器を操作する場合，対象機器が HTTP によるネットワーク接続であればプロトコルに沿った送受信を，シリアル・パラレルなどであればポートの入出力命令といったように，その通信内容や入出力の手順が機器によって異なる．そこで，どのパラメータを用いて，どのような順序で機器に対して操作をするかという具体的な手順を記述する必要がある．この記述内容には，以下のようなものがあげられる．

- 機器に固有なパラメータを変数に読み出す．
- 変数どうしの演算（数値の場合），連結（文字列の場合）を行う．
- 変数の内容を対象機器へと送信する．
- 受信動作をし，結果を生成するために受け取った数値や文字列を変数に格納する．
- 変数の内容をユーザへと返す．

これらの処理手順を記述することで，サーバと対象機器との間で，どのパラメータを用いてどのような手順で通信を行うのかを定義できる．この処理手順は，シ

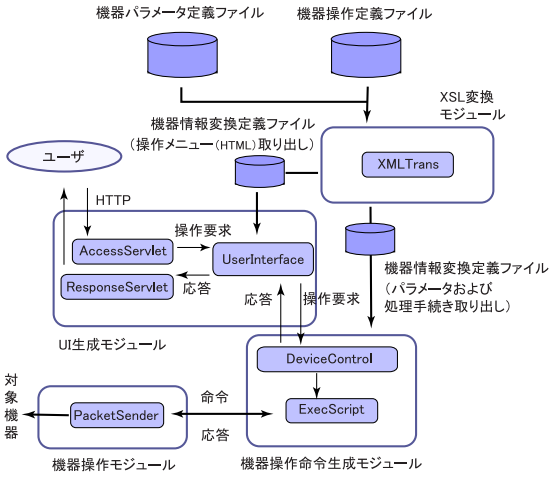

図 2 モジュール構成
Fig. 2 The module structure.

ステム構築者が容易に記述できるようスクリプト言語を用いて記述する．

(3) **ユーザインタフェースに関する記述**

システムを利用するユーザは，Web を経由して本システムにアクセスするため，HTTP を用いたユーザインタフェース（以下，UI）を作成する必要がある．しかし，対象となる機器の特性はそれぞれ異なるため，すべての機器に適合する UI をあらかじめ用意することはできない．そこで，操作画面と結果表示画面を動的に生成することを考える．そのためには，XML で記述された定義文書より，そのつど一般利用者へ提供する HTML を構成する必要がある．そのため，一般利用者へ返す画面の構成を定義ファイルの中に記述する．

これら (1)〜(3) の記述は，定義ファイルとして格納される．定義ファイルは XML で記述されるため，システムで利用するためには，要求に応じて変換を行わなくてはならない．また，操作のつど UI を生成する必要があること，作られた操作命令を実際に機器へと送る信号に変換する必要があることなどから，以下の 4 つのモジュールでシステムを構成する（**図 2**）．

(1) **UI 生成モジュール**
(2) **XSL 変換モジュール**
(3) **機器操作命令生成モジュール**
(4) **機器操作モジュール**

UI 生成モジュールは，操作画面と結果画面の提供を行う．各モジュールが機器に関するデータを要求したときは，XSL 変換モジュールが，XML で書かれた機器定義文書を一定の形式に変換して要求元に渡す．このときの変換規則を，機器情報変換定義と呼び，XSL 変換による変換を行うためのスタイルシートとして記述を行う．通信命令生成モジュールは，ユーザインタフェース生成モジュールより操作リクエストを受け取ると，機器定義を問い合わせ，操作手順に従い命令を生成する．最終的に，生成された操作命令は機器操作モジュールによって実際のプロトコルに変換され，機器へと送受信される．

前記の (1)〜(3) を記述することで，機器の処理，通信や入出力，UI の生成を行う．これらの記述は図 2 中の機器パラメータ定義ファイル，機器操作定義ファイルに分けて格納される．各定義ファイルはそれぞれ一定の構造を持ち，記述の容易性の観点から，これらは 1 つにまとめずに別々のファイルに記述した．通信手順に依存するものは，(2) と (3) と考え，機器操作定義ファイルに記述する．また，(1) のパラメータ（通信内容）については，通信手順とは別々に定義する必要があるために，機器パラメータ定義ファイルに格納する．そして，これらの情報を各モジュールへ受け渡す形式に変換するための規則を，機器情報変換定義ファイルに記述する．これらのファイルについて，次章で詳細を述べる．

4. XML による機器と処理記述

4.1 記述ファイルの概要

(1) **機器パラメータ定義ファイル**

機器パラメータ定義ファイルには，IP アドレスやポートなどの機器に関する情報と，3.2 節にて示した対象機器に固有なパラメータを記述する．このファイルには，同じタグ構造で複数の機器パラメータを記述することができる．パラメータの記述方法は，パラメータ名をタグとして記述することでパラメータ名とパラメータ内容を同時に複数記述できるようにし，次項目で述べる機器操作定義において任意のパラメータを取り出せる設計とした．

(2) **機器操作定義ファイル**

機器操作定義ファイルには，機器に対する通信手順ないしは入出力処理および UI を記述する．たとえば，ネットワークに接続された機器については，コマンド送信と結果の受け取りなどの通信手順を．サーバに直接接続された機器については操作の処理手順を本ファイルに記述する．本システムでは，記述を容易にするため，独自のスクリプト言語を設計した．本スクリプト言語では，機器へのコマンド送受信手順を単純なコマンドで記述できる．本スクリプト言語で対応できない複雑な処理については，本ファイル中に Java 言語などにより記述された処理プログラムを埋め込み，一貫した管理を行える設計とした．

UI については，Web を基本としている．Web 画面のデザインおよび機器から受け渡されるデータを本ファイルで定義する．

(3) **機器情報変換定義ファイル**

機器パラメータ定義ファイルと機器操作定義ファイルによって，機器の処理手順およびユーザインタフェースを同一ファイルで一貫した記述と管理ができるようになる．ただし，このままでは各モジュールで解釈実行できる形式にならないため，本定義ファイルで，機器パラメータファイルと機器操作定義ファイル中の各記述を取り出し，変換することが必要となる．本機器情報変換定義ファイルでは，以下の 2 つを記述する．

(a) HTML への変換規則

ユーザへ提供する機器操作メニューを具体的に HTML に変換する，ユーザインタフェースの生成規則を記述する．

(b) パラメータおよび処理手続き取り出しの規則

実行に必要なパラメータとスクリプト言語ないしは処理手続きを取り出す規則を記述し，XML による機器定義をモジュールが理解できる文字列に変換する．このように XSL 変換を用いることで XML で表現された機器定義文書から HTML を構成することが可能となり，サーバが読むデータと，利用者に提供されるデータを同じフレームワークで扱うことができる．

4.2 機器パラメータ定義ファイル

機器パラメータ定義ファイルは 3.2 節に従い，個々の機器に関するデータだけが記述されたファイルであり，XML の記述形式に従って要素はタグで囲まれる．XML としての基本的な形式を**図 3** に示す（UML の図において，菱形は集約を，数値は互いの多重度を表す．XML の構造を表すため，階層は集約にて，属する要素の数を多重度で表した）．本ファイルは通信に必要なホストや名称などの基本データのほか，ユーザによる定義が可能な機器パラメータで構成されている．

トップレベルには <Parameters> タグを配置し，その下にその機器が持っている操作定義を Http のように半角英数字で記述する．また，機器 ID は対象機器に一意なものを半角英数字で与える．ホスト情報は，アドレスを指定する <Destination> タグとポートを指定する <Port> タグから成っている．機器の名称は，機器 ID に対応するものを文字列で書き，ユーザへと提供する際に使われる（例：赤外線リモコン）．

一方，<DeviceData> タグは機器に固有なパラメータを記述する部分となり，このタグで囲んだものはすべて機器パラメータとなる．例として**図 4** のような記述を行った場合，変数 TestValue には数値 0x10 が，

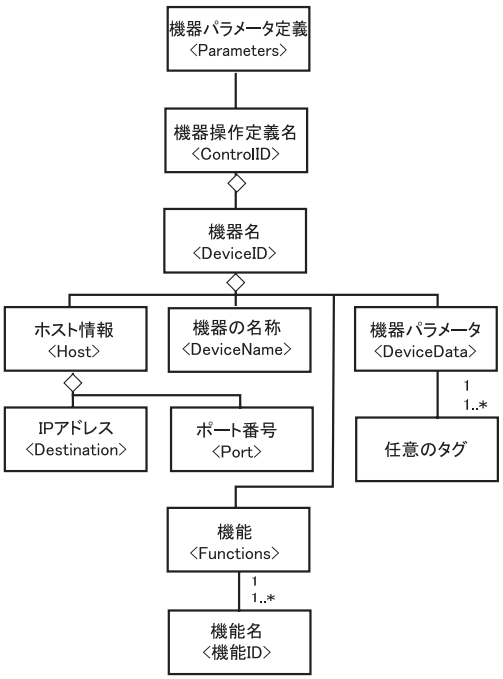

図 3 機器パラメータ定義ファイルの記述形式
Fig. 3 UML definition of device parameter definition file.

```
<DeviceData>
    <TestValue>0x10</TestValue>
    <TestString>XYZ</TestString>
</DeviceData>
```

図 4 DeviceData の記述例
Fig. 4 An example of device data description.

TestString には XYZ という文字列を与えることができる．このように，機器パラメータ記述部分ではタグでパラメータ名とその値を対応付けることができる．記述に許されている形式は，文字列と数値（10 進，16 進）であり，パラメータの個数に上限はない．

<Functions> タグには利用できる機能の一覧を，一意の機能 ID とともに記述する．なお，機能の具体的な内容は，機器定義に書かれたデータをどのように使って通信を行うかで決まるため，機器パラメータ定義ファイルには記述せず，後述の機器操作定義ファイルに記述する．

4.3 機器操作定義ファイル

機器操作定義ファイルの記述形式は**図 5** のようになっており，すべての機能に共通の手続きを記述できるのに加え，機能ごとに異なる処理を記述することができる．パラメータ定義と操作定義は対をなすため，内容はそれぞれに対応したものでなければならない．

手続きの部分には制御スクリプトを記述することが

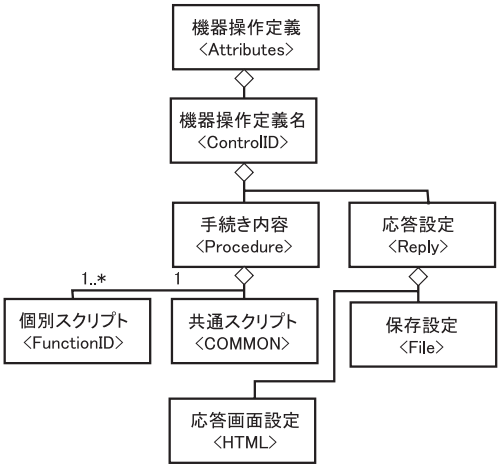

図 5 機器操作定義ファイルの記述形式
Fig. 5 UML definition of device control definition file.

できる．<Procedure> タグで囲んだ部分は制御スクリプトと見なされ，独自に設計した制御用の言語で記述する．制御スクリプトによる処理の内容は，対象機器に共通したものは <Common> タグ内に，選択した機能に特有なものは，< 機能 ID> をタグとしてその中に記述できる．これにより，通信方法の定義および実行は共通のスクリプトで行い，そのために必要なデータの収集は個別スクリプトで行うといった動作が可能となる．また，応答結果の扱いは <Reply> タグ内に記述する．<File> には属性 type で結果を保存するか否かを（0 で非保存，1 で保存），属性 name でファイル名を指定する．<HTML> タグ内には，ユーザへと送る HTML を指定し，この内容が結果画面として表示される．この際，%スクリプト変数名%という記述を入れることで，スクリプト内の変数を HTML 内に埋め込むことができる．また，リプライが画像などの場合は一度保存をする必要があるが，取得日時をファイル名として保存され，予約語%FileName%にて指定することができる．

図 6 に掲載した命令は，システムが標準として用意したスクリプト言語の命令である．変数の型は，文字列 (string)，数値 (int)，数値配列 (int[])，実数 (real)，式 (expr) の 5 種類が存在し，変数宣言を行う．expr 型は，式を保持し，Eval 関数で評価するために設けた．機器に固有な値を計算する必要があるときは，記述した式を評価し，結果を得ることができる．式は，次の 3 つの演算を行うことができる．

(1) 代入等号で結ぶことにより，変数どうしの代入が可能
(2) 四則演算等号で結ぶことにより，加減乗除が可能

```
int Variable, int[] Variable, string Variable,
  expr Variable, real Variable
  指定された型で変数を宣言する
ReadValue(Data:int or string,Tag)
  Tag で指定された機器パラメータを，Data で指定された変数に格納する
Send(Data:int[] ,Protocol TCP or UDP)
  指定された変数をその変数の型でデバイスに送信する
Eval(Data:expr, Result:int or real)
  指定された数式の評価を行い，結果を Result に格納する
Receive
(Data:int[],Protocol TCP or UDP,Timeout:int)
  デバイスからの応答を受信し，指定された変数に応答を格納する
Copy(Data:  int[],Position:int,Length:int)
  指定された配列の一部を切り出したものを返す
Return(Data:  int[] or int or string)
  指定された変数をユーザインタフェース生成モジュールに返す
```

図 6 制御スクリプト命令一覧
Fig. 6 Commands of the script language.

```
四則演算例
tmp = (Variable1 * 2) + Variable2;
文字列連結例
send := "GET /" .  file .  " HTTP/1.0%#%#";
                    （%は復帰，#は改行を表す）
```

図 7 式の記述例
Fig. 7 Examples of expression in the language.

(3) 文字列連結代入演算子 (:=) と連結演算子 (.) を用いることで文字列の連結が可能

通常の代入のほか，四則演算では加減乗除が，文字列連結では，単純な文字列どうしの連結のほか，数値配列への代入も可能である（**図 7**）．また，式として変数に記述することができるものも，以上に示したものと同様である．

送受信は，TCP/UDP にて行うため，送受信関数に渡すデータ形式は数値配列とした．基本的な手順は，機器パラメータを ReadValue 関数で読み込み，必要に応じて式で演算を行い，最終的に連結式によって数値配列へと代入する．なお，数値配列への変換時は，対象が数値であるか文字列であるかを判断し，適切な置換えを行う．

機器からの応答を受信する際も，受信したデータ列を数値配列として記録する．応答結果を加工する必要があるときは，Copy 関数によって一部を切り出し，演算をすることができる．最終的に，結果は Return 関

```
<External Senddata= VariableNameList,
Recvdata= VariableName>
    <![CDATA[
        Java Program Here
    ]]>>
</External>
```

図 8 外部関数の定義
Fig. 8 XML tags for external functions.

数によりユーザインタフェース生成モジュールへと渡され，スクリプトは終了する．

また，<Procedure> タグ内では本スクリプト言語以外の言語で記述された処理内容も外部関数として呼び出せるようにした．外部関数は Java 言語を用いて直接スクリプト内に記述できるようにし，制御スクリプト中の任意の場所から，本システムのスクリプト言語では記述できない複雑な処理も実行できる．

図 8 は外部関数の定義である．関数部分は予約語である <External> タグで囲まれ，関数に受け渡す変数名を引数に記述する．これを制御スクリプトの任意の部分に挿入することで，スクリプトインタプリタはスクリプトの処理から外部関数の実行を開始し（クラスファイルが存在しなければコンパイルを行う），戻り値を再び制御スクリプトに渡すことができるようになる．引数として，属性 Senddata の値に関数に渡すスクリプト変数をカンマで区切り，Recvdata の値に，戻り値を格納するスクリプト変数を記述する．クラスは，<External> タグが読み込まれた時点で実行され，実行が終了すると Recvdata で指定されたスクリプト変数に結果が収められる．

<External> タグ内に記述されたものは独立したモジュールとして動作するので，XSL 変換を行ってデータを取り出し，あるいは制御スクリプトを使わずに外部との通信を行うといった，OS 系のシステムを直接実行させることが可能となる．また，この部分は自由に拡張が行えるため，システムの内部にとどまらず外部，たとえば取得したデータの保存を行い，sendmail などの MTA により結果をメールで送信するなどといったことも可能になる．

以上の 2 項目による定義ファイルにより，対象機器との通信をテキストで記述することが可能となる．

4.4 機器情報変換定義ファイル

本システムでは「機器リストの取得」と「機器パラメータ・操作定義の取得」の 2 種類の変換を定義している．

例として，機器リストの取得時には，HTML 形式で受け渡しを行い，受け取ったモジュールはそれを用いて操作画面を構成する（**図 9** A）．また，ユーザからの要求が来て，操作のために機器パラメータ・操作定義を取得する必要があるときは，機器パラメータと操作定義を列挙した文字列を受け取り（図 9 B），中から必要なものだけを取り出し，処理を行う．

```
(A) 機器名の取得時（温度センサに対する数値取得）
<Form name="ANALOG_SENSOR"
method="get" action="AccessServlet">
<input type=RadioButton name="Temperature"
value="GET_TEMP">
...
</Form>
(B) 操作内容の取得時（温度センサのパラメータ）
-
Temperature（機器 ID）
Constant:  0.48828125  （変数と内容）
Picnic_Pin:99  （変数と内容）
-
Humidity（機器 ID）
...
```

図 9 変換出力形式の例
Fig. 9 Examples for output data conversions.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<Http>
    <KX-HCM1>
        <DeviceInfo>
          <Attr>HTTP</Attr>
          <DeviceName>Web カメラ </DeviceName>
          <DeviceID>WebCamera</DeviceID>
          <Host>
            <Destination>192.168.5.71</Destination>
            <Port>80</Port>
          </Host>
        </DeviceInfo>
        <DeviceData>
          <Command>GET</Command>
          <FileName>SnapshotJPEG</FileName>
          <Command1>Resolution</Command1>
          <ResQvga>320x240</ResQvga>
          <ResVga>640x480</ResVga>
        </DeviceData>
        <Functions>
            <Get_qvga> 画像取得 (QVGA)</GET_qvga>
            <Get_vga> 画像取得 (VGA)</GET_vga>
        </Functions>
    </KX-HCM1>
</Http>
```

図 10 機器パラメータ定義ファイル
Fig. 10 An example of device parameter definition file.

4.5 記 述 例

実際の記述を行ったものを**図 10**，**11**，**12** に示す．ここでは，Web に対応したネットワークカメラ（Panasonic KX-HCM1）から画像を得るための定義を作成した．ネットワークに対応したもの以外の機器については，5 章の試作システムで述べる．

このカメラは，Web サーバを搭載し，HTTP による

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<Attributes>
    <Http>
        <Reply>
            <Html>
               送信が完了しました <br/>
                <img src="%FileName%">
            </Html>
            <File type="1" name="hoge.jpg"/>
        </Reply>
        <Procedure>
            <Common>
              int[] send,recv;
              string com,file,com1,res;
              ReadValue(com,Command);
              ReadValue(file,FileName);
              ReadValue(com1,Command1);
          </Common>
          <Get_qvga>
              ReadValue(res,ResQvga);
          </Get_qvga>
          <Get_vga>
              ReadValue(res,ResVga);
          </Get_vga>
          <Common>
              send := com." /".file."?".com1."=".
                              res." HTTP/1.1%#%#";
              Send(send,TCP);
              Receive(recv,TCP,3000);
              Return(recv);
          </Common>
        </Procedure>
    </Http>
</Attributes>
```

図 11 機器操作定義ファイル
Fig. 11 An example of device control definition file.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <xsl:apply-templates select="Attributes"/>
</xsl:template>
<xsl:template match="Attributes">
    <xsl:for-each select="./*">
        <xsl:value-of select="name()"/>
        <xsl:apply-templates select="./Reply"/>
        <xsl:apply-templates select="./Procedure"/>
        end
    </xsl:for-each>
</xsl:template>
<xsl:template match="Reply">
        html
        <xsl:value-of select="./Html"/>
        type
        <xsl:value-of select="./File/@type"/>
        name
        <xsl:value-of select="./File/@name"/>
</xsl:template>
<xsl:template match="Procedure">
    <xsl:for-each select="./*">
    <xsl:value-of select="name()"/>
    <xsl:value-of select="."/>
   -
    </xsl:for-each>
   =
</xsl:template>
</xsl:stylesheet>
```

図 12 機器情報変換定義ファイル例
（機器操作定義を取得する変換定義）
Fig. 12 An example of device data conversion definition file.

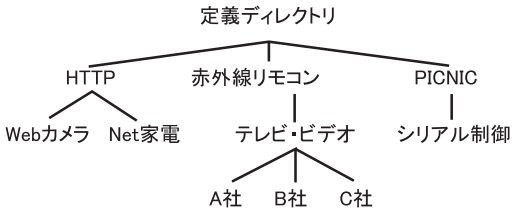

図 13 定義のディレクトリ分類例
Fig. 13 An example of device categories.

リクエストを送信することで画像を取り出すことができる．Web サーバには，SnapshotJPEG というファイルがあり，これに対して画像サイズ（Resolution），品質（Quality）を引数として与えることで現在の画像が JPEG で返される．このため，ファイル名と引数の名前，引数の内容を機器パラメータとして図 10 のように定義し，操作定義にて HTTP のコマンドである “GET” や “HTTP/1.1” を加えて送受信を行う（図 11）．

4.6 定義ファイルの入手方法

新しく機器を追加する場合，その機器向けに設計された機器パラメータ定義ファイルと，機器操作定義ファイルをサーバに記録する必要がある．しかし，通信プロトコルや入出力手順への理解があるユーザならまだしも，一般の利用者にとってファイルを作成するのは容易な作業ではない．そこで，開発側からこれらのファイルを提供する仕組みを考える必要がある．この仕組みの 1 つに，対応するファイルをディレクトリとしてオンラインで公開する方法がある．

定義ファイルをオンラインで配布する場合，同じ家電機器でも，テレビ，エアコンといったカテゴリ，メーカーによる分類，操作法による分類など無数の属性分けができるため，どのようにディレクトリを区分するかが重要になってくる．今回，トップレベルでの分類として「操作方法」を配置し，その下にカテゴリごとに文書の場所（URI）の記述を配置する構造とした（**図 13**）．

定義の配布に HTTP を用いると，ディレクトリサーバが，それぞれの定義を置いているベンダの URI を認識して管理することで，ユーザから要求が来た場合に，何の定義がどのサーバにあるといった情報を返し，システムはそれを基に実際の内容を取得するといった動作ができると考えられる．

このディレクトリサービスへの HTTP を経由した

表 1 試作に用いたシステム
Table 1 A prototype system.

サーバマシン	PlatHome 社 OpenBlockSS
Java	J2SE 1.3.1
サーブレットコンテナ	Tomcat 3.3.1
XML 処理系	Sun JAXP1.1

アクセス方法自体を機器定義文書に記述することにより，ファイル書き込みなどの一部機能は外部プログラムで記述する必要は出るが，新規機器に適合する定義文書をベンダより取得し，システムが持つ機器定義リストに自動的に取り込むといった動作が可能になる．なお，機器 ID については一意性を保証しなくてはならない．そのため，名前空間をベンダごとの URI と組み合わせることで機器 ID を一意に定めることができる．

5. 試作システム

5.1 ハードウェア構成

サーバには，Linux を搭載している小型のマシンを選んだ．常時接続を考え，小型化と低消費電力化を図った，10 cm 角程度のいわゆるマイクロサーバは各社から製品が発表されているが，開発が容易なこと，ある程度の性能が確保できることから，Plat'Home 社のマイクロ Linux サーバ OpenBlockSS（PowerPC 200 MHz，64 MB RAM）を使用した．このサーバには Ethernet の 10/100 BASE ポートが 2 つ，RS232C ポートが 1 つ存在する．そこで入出力インタフェースを Ethernet とし，その他のインタフェースを要求する機器ついては，必要に応じてプロトコル変換を行う構造とする．

プロトコルの変換には，Tristate 社のネットワークインタフェースカード PICNIC を用いた．PICNIC は TCPwUDP/IP からパラレル・シリアル入出力を制御できるカードであり，一定のプロトコルに従って UDP パケットを送信することで，シリアル・パラレルポートの出力を操作でき，同時にポートに入力されている内容を取得できる．これを経由してサーバと機器を接続することで，サーバから非ネットワーク機器の操作が可能となる．構成については**表 1** のとおりである．

5.2 機器定義の試作と評価

4.5 節で示した図 10〜12 の Web カメラの動作画面を**図 14** に示す．正しく画像の取得が行えていることが確認でき，サーバと直接ネットワークによる接続を行う機器において，リクエスト文字列を組み合わせて通信文を生成し，HTTP などのプロトコルを操作手順として記述することによって機器定義が可能であるということを確認することができた．同じく Web へのアクセス方法を提供する WSDL と比較すると，本システムの方法によりサービスを簡単に記述することができるほか，HTTP を用いた Web サービスにとどまらず，ネットワークによる通信全般を定義することができるという利点がある．

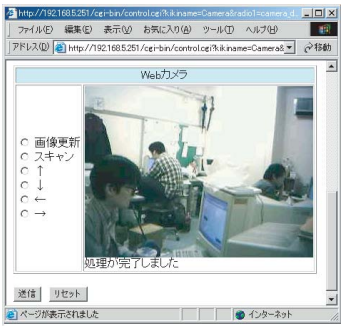

図 14 Web カメラの操作画面例
Fig. 14 Sample WWW page of controller for WWW cameras.

次に，IP 以外のインタフェースを持った機器に関する記述性を評価するため，赤外線リモコンについて記述を行った．赤外線リモコンは，シリアル信号を入力すると赤外線を発光する装置を作成した[10]．この装置は，家電協会で定められている赤外線信号のパラメータと，送信するデータをシリアルポートから入力すると，信号を生成して赤外 LED から出力する．機器のパラメータとしては，「受信したデータをシリアルから出力」という PICNIC の制御命令，メーカーごとに異なるヘッダ情報と，操作内容に相当する情報を記述し，それを “PICNIC のシリアル経由で赤外線パラメータを与える” という操作定義と組み合わせた．この結果，正しく目的の家電製品が動作することを確認でき，非ネットワーク機器に対しては，パラメータとして，プロトコル変換装置へのコマンドと，機器へ送るデータを記述することで操作が行えることが確認できた．これを機器制御規格である Jini と比較すると，対象機器に外部インタフェースがあれば，ソフトウェアの組込みなど，機器自体に手を加えることなく制御機能が加えられるという利点がある．

本システムでは，任意のデータを組み合わせて通信を行えるという特長があるため，以上に示した試作のほか，有効な記述が考えられるものとして，Web サービスの操作があげられる．入力と出力の形式があらかじめ定まっている場合，カメラの記述例と同じように引数を機器パラメータとして記述することでサービス

の操作が可能である．また，このように，TCP/UDPの複数個のコマンドを組み合わせることで命令文が生成でき，一定の形式に従った応答を返すものについては，本システムの記述法で定義することができる．

試作システムに関しては，いずれも所定の結果が得られた．実行時にはXMLの変換処理を含め15秒から30秒程度の時間がかかったが，これは製作に用いたマイクロサーバのCPUが200 MHz，RAMが64 MBという比較的制約のある環境にJ2SDKを搭載して実行したことが原因であり，一般的なPentiumクラスのPC（Celeron 300 MHz 128 MBのPCにて試験）であれば，送信から応答まで5秒以下で実行することができた．今後，組込みシステム，ハードウェアの高性能化により，解決は容易になると考えられる．

記述量については，Webカメラの画像取得操作を例にとると，機器パラメータ定義が約30行，機器操作定義が約35行で記述が可能という結果となった．画像取得に加え，カメラの視点移動などの新たな操作コマンドの追加を行うときは，追加すべき定義量は約10行となり，記述の容易性が確認できた．

一方，本システムを利用するうえでの問題点に，非ネットワーク機器の動作確認が難しいという点がある．例をあげると，赤外線を発光したという結果は受けることができても，実際に家電製品が作動したかは確認することができない．確認をするためには，フィードバックのためのノードを追加する必要があり，被制御機器に手を加えずに操作が可能という特長と引き換えとなっている．

また，もう1つの問題点としてあげられるのが，現状ではユーザからのリクエスト（イベント）に対してサーバが応答（アクション）を返して動作を完了する方式となっているため，ユーザが要求したときにしかデータを送受信することができない点である．また，制御スクリプトには，送信文を構成するのに最低限必要な命令しか実装されていないため，下記のように機器の応答によって動作を変えたり，条件を設定して繰り返し処理を行うといったことはスクリプトだけでは実現できない．

- 機器からの応答を待ち，応答が正しくなければ信号を再送する．
- 防犯センサが反応していない限りループし，感知したらカメラの映像を携帯電話に送信する．

しかし，このような複雑な記述は，スクリプト中に外部関数を記述することで可能となる．

定義に関しての問題は，機器パラメータを自由に記述し，制御スクリプト側でそれを組み合わせるという方法をとっているため，記述に関しての規則が機器別に確立されていない点があげられる．誤ったものが書かれる可能性も十分にあるため，スキーマなどを用いて明確にする必要がある．

6. おわりに

IPネットワークを通した機器制御を実現し，接続された機器を一括管理するシステムの設計と実現を行い，リモコンによる家電制御などの試作システムでの評価を行った．情報家電が普及していくであろう今後，メーカーによって完全に統一した操作系が提供されることはないと考えられる．また，常時接続が一般化しつつある今日，外部から様々な機器に対して自由な操作が行えるフレームワークの構築というのは，十分に意義のあることと考える．

参考文献

1) NTTデータJava研究会，荒川弘熙：Jiniって何だ? — Javaがもたらす近未来のネットワーク技術テクノロジーを知る，（株）カットシステム (1999).
2) Dutta-Roy, A.: Network for Homes, IEEE Spectrum, Vol.12, pp.26–33 (Dec. 1999).
3) INPROBE製品案内，CAFEMOON@HOME. http://www.inprobe.com/product/cafemoonathome.html
4) Webサービス記述言語—WSDL1.1. http://www.microsoft.com/japan/developer/workshop/xml/general/wsdl.asp
5) MARON-1，富士通研究所. http://pr.fujitsu.com/jp/news/2002/10/7.html
6) 本間一也，矢吹道朗：ダイヤルトーンを用いた計算機アクセスシステム，情報処理学会第35回プログラミングシンポジウム，pp.83–92 (1994).
7) 日本ユニテックDigitalXpress編集部：SOAP/UDDI/WSDL Webサービス技術，基礎と実践徹底解説，技術評論社 (2002).
8) L-BOXの開発，NTTコムウェア. http://www.nttcom.co.jp/comtech/tech02/tech0229.html
9) 永田智大，西尾信彦，徳田英幸：ASAMA（適応的なサービス利用機構），情報処理学会論文誌，Vol.42, No.6, pp.1557–1569 (June 2001).
10) 寒河江忠男：PIC赤外線リモコンI/Oターミナル．http://www.page.sannet.ne.jp/sagae/hard/ircon.html
11) AtMarkIT—技術者のためのXML再入門. http://www.atmarkit.co.jp/fxml/rensai/rexml01/rexml01.html
12) 増渕　敬，並木美太郎：小型サーバとXMLを

用いた遠隔機器制御支援システム，FIT 情報科学技術フォーラム投稿論文，LM-36 (2002).

(平成 14 年 12 月 21 日受付)
(平成 15 年 4 月 17 日採録)

増渕 敬（学生会員）

1979 年生．2002 年東京農工大学工学部情報コミュニケーション工学科卒業．同年，同大学大学院工学研究科情報コミュニケーション工学専攻修士課程入学．情報家電・ユビキタスネットワークに興味を持ち，ネットワークソフトウェアに関する研究に従事．

並木美太郎（正会員）

1984 年東京農工大学工学部数理情報工学科卒業．1986 年同大学大学院修士課程修了．同年 4 月（株）日立製作所基礎研究所入社．1988 年東京農工大学工学部数理情報工学科助手．1989 年同大学電子情報工学科助手．1993 年 11 月同大学電子情報工学科助教授．1998 年 4 月同大学情報コミュニケーション工学科助教授．博士（工学）．オペレーティングシステム，プログラミング言語，ウィンドウシステム等のシステムソフトウェア，並列処理，コンピュータネットワークおよび日本語情報処理の研究・開発に従事．ACM，IEEE 各会員．