XMLを用いた遠隔機器制御支援システムとDBMSやICカードの連携

増渕　敬†　並木　美太郎†
櫻田　武嗣††　萩原　洋一††

現在，ネットワークを利用した通信・制御が容易に行える環境が整備されつつある．しかし，機器同士の互換性や，コストの問題により，大半の機器はネットワークに接続されることなく運用されている．本研究は，制御対象となる機器のプラットフォームやインタフェースに関係なく，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しているものである．本論文では，システムが持つ XML データベースと，外部の RDBMS との連携の必要性について述べ，データ管理方法の設計と実装の手法について述べる．また，認証のために非接触型 IC カードとの協調動作を設計および実装し，既存のシステムの枠組みの中でシステム拡張および外部との連携が可能であることを確認した．

Cooperation of Network Appliance Control System with XML, and DBMS or IC Card

Takashi Masubuchi,† Mitaro Namiki,† Yoichi Hagiwara†† and Takeshi Sakurada ††

In recent years, the environment for access to a remote computer from anywhere, is being improved. However, due to the problem about compatibility or costing, mostly apparatus does not have a network function such as DBMS abd RFID devices. In this research, we proposing the environment where arbitrary equipments can be operated on a common platform, irrespective of the existence of a network function. In this paper, the necessity for cooperation with the XML database, and external RDBMS is described, The data management method and technique of mounting is designed. Moreover, cooperation with a contactless IC card was designed and mounted for authentication, and it confirmed that system extension and cooperation with the external function were possible in the framework of the existing system.

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．

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について述べた上で，データベースの必要性と外部とのデータの連携方法，認証のための非接触型 IC カードとの連携のための手法について述べる．

† 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
†† 東京農工大学 総合情報メディアセンター
General Information Media Center,
Tokyo University of Agriculture and Technology

2. システムの概要

本節では，”XML を用いた遠隔機器制御支援システム[10)]”の概要について述べる．本システムは，任意の機器をネットワーク経由で制御することができる環境をめざし，次のような目標で設計された．

2.1 システムの目標

(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，プロトコルやインタフェースが異なる機器に対して，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，複数の機器を一つのサーバ，同じプラットフォーム上で制御を行うことができる環境をを実現する．

(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とで，ネットワークが利用できるすべての場所からのアクセスを可能とする．

(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ことは困難であるほか，規格を揃えなければいかないといった問題がある．そのため，本システムでは，機器の外部インタフェースを利用し，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目標にする．

制御サーバには，操作対象の機器に対してどのような通信文を生成したら操作ができるのかといった情報を記述しておくことで，機器を仮想化し，利用者が操作内容を定義できるという特長を持たせる．

2.2　システムとモジュール構成

対象となる機器をサーバで一括して管理するため，クライアントは個々の機器に直接接続して操作を行うのではなく，サーバがすべての操作内容を生成して通信を行う方式をとる．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に配置し，その下にツリー状に機器を接続していく形をとる．制御系の基幹には TCP/IP を利用し，ネットワークに対応していない機器はプロトコル変換装置を介してシステムに接続する．

システム構成図を図 1 に示す．エンドユーザはブラウザを用いてサーバに操作リクエストを送信する．サーバは操作リクエストを受け取ると，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，操作を行うために必要な通信手順を生成し，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，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を出力する．

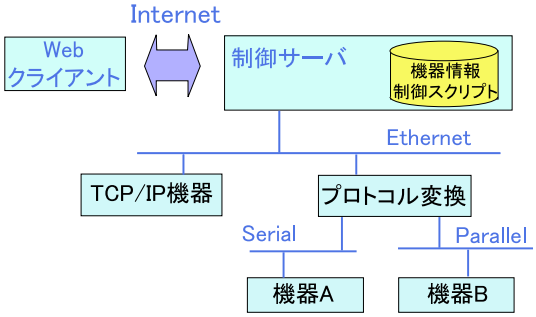

図 1　システム構成図

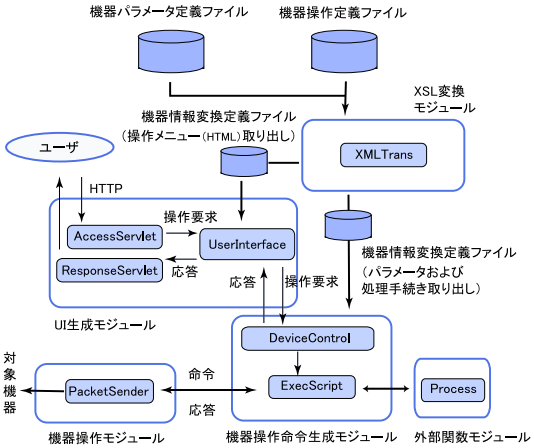

図 2　モジュール構成

システムのモジュール構成は，図 2 のようになっている．

UI 生成モジュールは，一番ユーザ側に位置し，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，機器へと送信される．

2.3　機器の記述方法

対象となる機器を仮想化する方法について述べる．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挙げられる．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し，ファイル化する．

(1)　機器パラメータ定義ファイル

任意の機器を操作するために，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．　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，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．すべての情報はこのファイルに記述され，次項目で述べる機

器操作定義において任意のデータを取り出せる設計とした．

(2) **機器操作定義ファイル**

サーバに接続された機器を操作する場合，対象機器の持つインタフェースによって，その通信内容や入出力の手順が異なるため，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通信を行うかという具体的な手順を記述する必要がある． 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 を含むデータの処理方法を記述する．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，本ファイル中にプログラムを埋め込み，実行できる設計とした．

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．以上の二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．

（記述方法の詳細については，”XML を用いた遠隔機器制御支援システム[10]”を参照）

3. データベースとの連携方法

本システムでは，機器からの応答を記録する手段として、サーバ内にある XML データベースと，サーバ外にあるリレーショナルデータベース外部データベースが利用できる．本章では，この二つのデータベースとシステムとの連携方法について述べる．

3.1 本システムにおけるデータベース

本システムでは，機器との通信結果については，操作リクエストを発行したクライアントのブラウザに表示される．しかし，応答の内容をデータとして再利用する場面も考えられるため，機器からのデータをサーバに保存することを考えなくてはいけない．そのため，サーバにはデータベースを持たせ，機器から得たデータはそこへストアされる設計とした．このデータベースは，実装が容易なこと，システムが扱う文書が全て XML で記述されており親和性があることなどから，XML 形式のデータベース（以下，XML-DB）を利用した．図 3 は，図 2 の UI 生成部分にデータベースを加えて拡大したものである．応答がユーザへ返されるのと平行して，XML-DB へのストアも可能となっている．

3.2 データの指定

本システムの基幹は TCP/IP であり，通常，機器

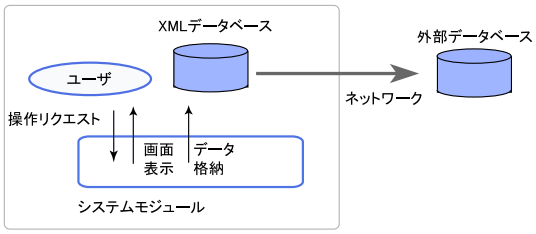

図 3 データベースとの関係

からの応答はデータグラムの数値列で送られてくる．しかし，この数値列が文字列表現なのか数値なのか，データ列のどの部分をユーザに提供すべきなのかは機器によって異なるため，あらかじめ範囲と型を明示しておかないと，ユーザ側へ提供することはできない．そこで，機器操作定義ファイルに，応答データを指定する部分を設定し，データの保存を可能にした．データ指定の方法を，図 4 に示す．

```
<Data Save="false" Start="0" End="8"
Type="str" Label="Test_string" />
<Data Save="true" Start="55" End="0"
Type="bin" Label="Test_image" />
```

図 4 応答データの定義例

表 1 Data セクションの属性

Save	True	このデータを保存する
	False	このデータを保存しない
Start	数値	データの開始位置をバイトで指定
End	数値	データの終了位置をバイトで指定
Type	str	データを文字列として定義
	int	データを 10 進数値として定義
	hex	データを 16 進数値として定義
	bin	データをバイナリとして定義
Label		データ名を半角英数で指定

記述法は表 1 の通りである．結果の表示と同時に，Save 属性にて”true”を指定されたデータは，システム内のデータベースに一時的に保存され，”false”を指定された場合は画面表示のみ行う．

記述形式は画面表示と同じくテキストとし，Data タグの Label 属性で指定したデータ名と，通信を行った日時が付加される（図 5）．なお，データがバイナリ形式の場合，テキスト形式である XML-DB に書くことができないため，Base64 によるエンコードをしたのちに記録される．

3.3 外部データベースの利用

前節で示したデータベースは，XML 文書で簡易的に記述されるため，大がかりな RDB 等を実装する必

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<Database>
    <Data Time="195128" Label="PICNIC1">
        00
    </Data>
    <Data Time="195128" Label="PICNIC2">
        05
    </Data>
    <Data Time="195128" Label="PICNIC3">
        3F
    </Data>
</Database>
```

図 5　一時データベースの記述例

要がなく，必要最低限の情報を保持するのに適している．反面，システムにはデータベースを管理するための機構が備わっておらず，XML 記述という条件により，テキスト形式以外での保存が行えないといった問題点がある．

一方，一般的に用いられている RDBMS を用いれば，上記の問題は解決されるが，システムに大きく手を加えなければならない点や，サーバ自体が分散環境で用いられていた場合に，データを統合させることができないといった問題点が存在する．そこで，図 3 にて示した，システムの外にあるデータベースに，本システムが持つ XML データベースの内容を転送させる機構が必要となる．

3.4　外部データベースへのエクスポート

本節では，システム内の XML-DB に蓄積されたデータを，外部の RDB 等に転送するための手法について述べる．

通常の操作によって機器から取得したデータは，図 5 にて示した形式で，機器ごとのファイルに保存されている．このデータベースは，通信を行うたびに新しいデータが追加されていく．これは，前に述べた理由により，一時的なデータベースとして考え，ユーザの操作によって外部に転送する必要がある．

転送のためには，まず XML-DB よりデータを取り出す作業が必要となる．次に，取り出したデータからクエリを生成し，対象データベースへのストアを実行させなければならない．後者は，任意のプログラムからアクセス用のライブラリを用いてデータをストアさせることが可能であるが，まずこの二点を結びつける必要がある．

3.5　実現させる手法

XML-DB からのデータの取り出し方法は，機器定義ファイルからデータを取り出す方法と同じく，XSL 変換を用いる．次にデータ転送用のプログラムであるが，エクスポート先のデータベースは，リレーショナルデータベースやディレクトリサービス等，様々な形態が考えられるため，特定のものに限定してしまうことは得策とはいえない．そこで，XSL 変換とクエリ生成，プログラムの実行を一度に行う方式を採った．

図 7 は，これを実現するための XSL スタイルシートである．これを用いて XML-DB を変換すると，図 8 が出力され，これを直接実行することで，データの転送が実現する．

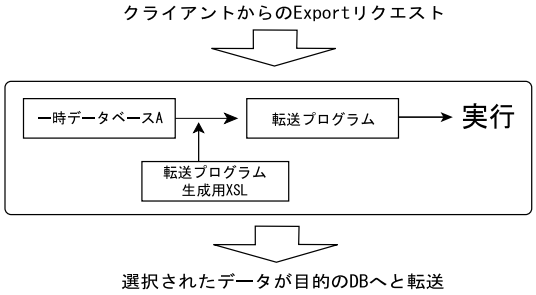

図 6　エクスポート時におけるデータの流れ

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org
/1999/XSL/Transform">

<xsl:template match="/">
#!/usr/bin/perl
use DBI;

$dsn = 'DBI:mysql:Test_Table';
$user = 'user';
$password = 'pass';
<xsl:apply-templates select="Database"/>
</xsl:template>

<xsl:template match="Database">
$dbh = DBI-<xsl:text disable-output-escaping="yes">&gt;
</xsl:text>connect($dsn, $user, $password);
<xsl:apply-templates select="Data"/>
$sth-<xsl:text disable-output-escaping="yes">&gt;
</xsl:text>finish;
$dbh-<xsl:text disable-output-escaping="yes">&gt;
</xsl:text>disconnect;
</xsl:template>

<xsl:template match="Data">
$sth = $dbh-<xsl:text disable-output-escaping="yes">&gt;
</xsl:text>
prepare("INSERT table SET Time=
<xsl:value-of select="@Time"/>,Label=
<xsl:value-of select="@Label"/>,Data=
<xsl:value-of select="."/>");
$sth-<xsl:text disable-output-escaping="yes">&gt;
</xsl:text>execute;
</xsl:template>

</xsl:stylesheet>
```

図 7　エクスポート変換規則

3.6　システムへの実装

この方法を実現させるためには，前節で挙げたように，XML 文書を XSL によって変換する動作と，ファ

```
<?xml version="1.0" encoding="UTF-8"?>（出力時に削除）
#!/usr/bin/perl
use DBI;

$dsn = 'DBI:mysql:Test_Table';
$user = 'user';
$password = 'pass';
$dbh = DBI->connect($dsn, $user, $password);

$sth = $dbh->prepare("INSERT table SET
        Time=20040107195128,Label=PICNIC1,Data=00");
$sth->execute;
$sth->finish;
$dbh->disconnect;
```

図 8　エクスポート変換例

イルを書き込み，必要であればコンパイルして実行させる動作が必要となる．本システムのモジュール構成は図 2 のようになっているが，このうち，前者に関しては XSL 変換モジュールに，後者は外部関数モジュールにデータを渡すことで実行させることができる．

実際の実装では，メニュー画面に外部データベースへのエクスポート項目を追加し，これらのモジュールを呼び出すための命令を追加したのみで，新たな処理のためのモジュールを作成することなくこの機能が実現した．

この設計方式をとることで，既存のモジュールを再利用でき，コードを増やすことなくデータベースに関する機能の追加ができるといった利点に繋がる．また，RDB でなく，ディレクトリサービスといった種類の異なるデータベースに転送を行いたいときは，変換定義ファイルを記述して追加するだけで対応が可能となるという利点がある．

4. Felica との連携

4.1 目　　的

FeliCa（フェリカ）は，ソニーが開発した非接触 IC カードの技術方式である．通信には 13MHz 帯の電波を用い，カード側は IC とコイルで構成され，無電源でおよそ 2 キロバイトの情報を収めることができるという特徴を持っている．また，セキュリティ評価基準の国際標準である ISO/IEC 15408 EAL4 の認定を受けており，変造や改竄に対して非常に強い．

本章では，このカードを認証の目的で利用することを考え，これまでに設計した枠組みの中で FeliCa システムを利用することができないかについて考察する．

4.2 FeliCa カードフォーマット

FeliCa は，一枚のカードの中に複数のデータを格納することができ，各々のデータには個別のアクセス権を設定することができる．ファイルシステムは,「エリア（フォルダに相当）」と「サービス」によって階層状に構成されており，サービスにてデータへのアクセス権を設定できる．データはエリアの下に配置されるが，それぞれのデータにはサービスを付加する必要がある．サービスは，データへのアクセス権（リード・ライトの権限）を設定でき，サービスキーによって認証をかけることができ，一回の相互認証で複数のファイルをオープンすることができるという特徴を持っている．

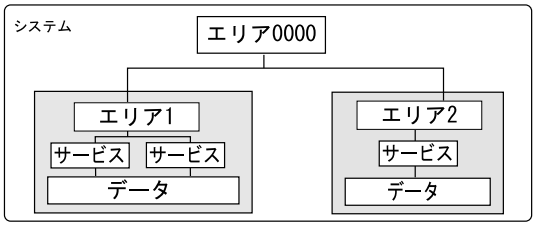

図 9　カードシステム

4.3 カードへのアクセス

FeliCa の SDK は，カードリーダおよび Win32 のダイナミックリンクライブラリ（以下 DLL）で提供されている．この DLL は，カードにアクセスするためのライブラリと，その下位に位置する，カードリーダにアクセスするためのライブラリで構成されており，アプリケーション側からは，カードリーダの存在を気にすることなくカードにアクセスすることができる．カードにアクセスをするためには，アプリケーションから DLL を通して関数を呼び出す必要があるため，本システムから FeliCa システムを利用するためには，Windows 上で動くアクセス用プログラムを経由しなければならない．

そこで，ソケットを用いてこのアクセス用のプログラムと本システムが通信できる設計にし，TCP/IP 経由でカードにアクセスできる仕組みを考えた．システムには，単純なデータの読み出しのほか，あらかじめ用意されたデータとカード内容との比較を可能にし，認証にも用いることができる設計とした．

4.4 設　　計

4.4.1 アクセスの手順

カードからデータを読み込むためには，決まった手順に従って DLL の関数を呼び出す必要がある．初期状態からデータを読み出すために必要な関数呼び出しの手順は以下の通りである．

- ライブラリの初期化（InitializeLibrary）
  利用するライブラリを初期化する．
- カードリーダ・ライタのオープン（OpenReader-WriterAuto）

カードリーダ・ライタをオープンする．

- アクセスキーの作成（MakeAccessKeys）
  カード認証に用いる鍵を作成する．
- ポーリングと認証（PollingAndMutualAuthentication）
  カードを捕捉し，アクセスキーを用いて認証を行う．
- データの読み出し（ReadBlock）
  認証済みのカードから読み込みを行う．

カードへのアクセスを行うプログラムでは，上記の関数を実装し，IP ネットワーク経由でリクエストを受け取り，DLL 経由でカードとの通信を代行する．

4.4.2 プロトコルの設計

カードへのアクセスを透過的に行うためには，前節で述べた関数をネットワーク経由で実行できる必要がある．本来ならば，一つの関数あたりに一つのコマンドを割り当て，一つずつ手順を踏みながらクライアントが命令を発するという方法が普通であるが，FeliCa の場合，前節で述べた関数は必ず順番に実行する必要があり，前の関数で生成した構造体へのポインタを次の関数に渡すという方式となっているため，本システムのスクリプト変数に容易に置き換えることができない．そこで，関数呼び出しの手順は固定とし，最低限必要となる，認証に関するパラメータのみをアクセスプログラムに渡すこととする．それに加え，本節で設計するプロトコルに沿って送受信を行うスクリプトを記述することで，カードとの通信が可能となる．

FeliCa では，認証のために，システムへのアクセスを許可するシステムキー，エリアへのアクセスを許可するエリアキー，サービスへのアクセスを許可するサービスキーを必要とする．そこで，この三つをパラメータとして渡すと，カードへのアクセスを行って結果を返す設計とする．プロトコルは表 2, 表 3 のようになる．

CreateSession	00
ReadFromCard	01 + SystemArea キー (16byte) + サービスキー（サービス数× 8byte）
CloseSession	02

表 2 送信文

ReadData	00 + カードから読み込まれたバイト列
Authentication	01 + 認証の可否 (0:失敗,1:成功)

表 3 受信文

アクセスモジュールは，CreateSession コマンドにより一対一での通信を開始する．ReadFromCard コマンドが発行されると，システムキー (8byte)・Area キー (8byte)・サービスキー (サービス数× 8byte) を受け取り，認証を行ってデータの読みとりを行う．返信の方法は，単純にバイト列を返信する ReadData と，カード内容との比較を行って認証結果を通知する Authentication の二通りを用意した．どちらが用いられるかはアクセスモジュールの設計による．そして，CloseSession コマンドにより，ライブラリを解放してリーダ・ライタをクローズ，接続を切断する．

4.4.3 アクセスモジュールの作成

カードとの通信には VisualBasic 用の DLL を使用したため，それを利用するプログラムも VisualBasic で作成する必要がある．基本的な機能は，前節で示したプロトコルとカードとの橋渡しであるが，認証機能を利用する場合，パスワードを保持してカードの内容と一致したら真を返すような機構が必要である．この場合，認証結果が真であれば処理を続行し，偽であれば中断するといったスクリプト記述をすることで，認証つきの制御システムを構築することができる．

4.4.4 デバイス記述

以上で設計したプロトコルを実装したものを以下に示す．この例では，必要なパラメータを用いてカードとの通信と認証を行い，認証に成功していれば外部関数内で希望する処理を，失敗したらエラーを返して終了するという使い方ができる．

表 4 FeliCa による認証システムのパラメータ記述例

```
<TypeDef>
    <int>
        <Int Max="2" Min="0" />
    </int>
    <str>
        <String MaxLength="16" MinLength="16" />
    </str>
</TypeDef>

<DeviceData>
        <CreateSession Type="int">0</CreateSession>
        <ReadFromCard Type="int">1</ReadFromCard>
        <CloseSession Type="int">2</CloseSession>
        <SystemKey Type="str">0F1E2D3C4B5A6978</SystemKey>
        <Area0000Key Type="str">0F1E2D3C4B5A6978</Area0000Key>
        <ServiceKey Type="str">0123456789ABCDEF</ServiceKey>
</DeviceData>
```

5. 評価と考察

第 2 章で掲げた，外部からネットワークサーバを通して特定の機器を制御するという目標を達成したうえで，データベース，IC カードとの連携を実現することができた．

エクスポート機能については，専用のプロセスを実行する必要があったことから，XML ファイルからプ

表 5 FeliCa による認証システムの入出力記述例

```
<Procedure>
    <FeliCa>
        <Internal>
            string val1,val2,val3,val4,val5,val6;
            int[] temp;
            ReadValue(val1,CreateSession);
            ReadValue(val2,ReadFromCard);
            ReadValue(val3,CloseSession);
            ReadValue(val4,SystemKey);
            ReadValue(val5,Area0000Key);
            ReadValue(val6,ServiceKey);
            temp := val1;
            Send(temp,TCP);
            temp := val2 .  val4 .  val5 .  val6;
            Send(temp,TCP);
            temp := val3;
            Send(temp,TCP);
            Receive(temp,TCP);
        </Internal>
        <External save="test.pl" exec="perl test.pl">
            変数 temp を参照して認証の成功を判定し,
            処理を行うプログラムを記述
        </External>
    </FeliCa>
</Procedure>
```

ログラムを生成するという発想により，設計済みだった外部関数実行のためのモジュールを利用することで，既存の枠組みを用いて機能を追加することができた．これにより，システムに関連する文書をすべて XML で記述することで，サーバが持つデータに対して共通の方式で処理を行うことができるという利点を再確認することができた．

IC カードとの連携については，実際に DLL を介して認証操作を行うためのプログラムを作成し，そのうえで，システムのスクリプトで直接操作を行うことができないデバイスについては，中間にモジュールを配置して TCP/IP で通信を行うことにより，システム自体に変更を加えることなく対応することが可能だということが確認できた．システムの基幹を TCP/IP で運用しているため，ネットワーク上にある，異なったコンピュータに接続されたデバイスに対しても操作を行うことができる．

いずれも，新規機能の追加を考えた場合だが，ネットワーク上にあるデバイスには通信モジュールを追加することで，ローカルに対する操作に関しては外部関数モジュールを利用することで，システムに大きな変更を加えることなく機能の拡張が可能であることがわかった．

6. おわりに

本論文では，XML を用いた遠隔機器制御支援システムへの機能拡張として，DBMS や IC カードとの連携方法について述べた．

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) Amitava Dutta-Roy: Network for Homes, IEEE Spectrum Vol.12, pp26-33, December 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) 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
6) 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
7) 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, June, 2001.
8) AtMarkIT - 技術者のための XML 再入門 http://www.atmarkit.co.jp/fxml/rensai/rexml01/rexml01.html.
9) 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.
10) 増渕　敬, 並木美太郎: XML を用いた遠隔機器制御支援システム, 情報処理学会論文誌：コンピューティングシステム, Vol.44 No.SIG10(ACS2), pp153-163, 2003.
11) 増渕　敬, 並木美太郎: ネットワークと XML を用いた遠隔機器制御支援システムの開発, 東京農工大学工学研究科, 2002 年度卒業論文, 2002.