組み込み Linux を使った Web アプリケーションの開発

近畿職業能力開発大学校　　滝　本　雄　一

Development of the Web application using Embedded Linux

Yuuichi TAKIMOTO

要約　家電や電話、時計や音楽プレーヤ等、身の回りのあらゆるエレクトロニクス機器がネットワークで接続され、誰でもどこでも意識することなく簡単に操作し、いつ何時でも世界中の情報を入手できる利便性を享受できる社会、「ユビキタス社会」が到来しようとしている。この社会を支えるキーテクノロジーが「組み込みシステム」である。

組み込みシステムによって実現されるユビキタス・ネットワークは、携帯電話に代表される Web クライアントが身の回りにあるだけでなく、従来はオフィスの片隅にあった Web サーバが家電製品等さまざまな機器に組み込まれ、リアルタイムに双方向通信を行い監視操作が可能になる。

このような環境下では、従来と異なる新しい Web アプリケーションが構築可能であると考え、能力開発においても組み込みシステムを応用した Web アプリケーション構築教材が必要であると考える。

本報では、組み込み用 OS として近年脚光を浴びている組み込み Linux について、その導入手法と Web 経由で機器の遠隔操作を行う簡易模擬システムの開発事例について報告する。

1　はじめに

コンピュータといえば一般的にはパソコンを指すほど、会社や家庭にパソコンが普及しているが、パソコンよりも更に普及しているのが、組み込み（embedded）システムである。

組み込みシステムとは、パソコン、ワークステーション、サーバ、及びこれらをベースにしたシステムのように、ソフトウエアによってさまざまな用途をこなすシステムと異なり、ROM もしくはフラッシュ ROM にプログラムが組まれ、CPU によって動作することができる、用途が限定された比較的小規模なシステムのことを指す。

実際の応用例は広く、POS 機器、カーエレクトロニクス等制御機器のみならず、携帯電話、PDA、ゲーム機等、情報家電と呼ばれる装置も組み込みシステムである。

従来は、機器の動作を制御するための比較的地味な存在であったが、近年この組み込みシステムが脚光を浴びている。その背景は、携帯電話に代表されるように、組み込み用マイコンの高機能化により、小型装置でもインターネット接続が可能になったことがあげられる。

また Java に代表されるように、アプリケーションも Web ベースのものが現れてきて、機器が超小型サーバの機能を有すれば、従来のパソコンベースのシステムも組み込みシステムに置き換えることができるようになる。

論文受付日　H15. 11. 10

本報は、エンタープライズ用途 OS として実績があり、近年組み込みシステム用としても注目されている Linux について、実際にシステム構築を行った３例の実践報告を行うものである。

ここで報告する内容は、制御用ボードマイコンを用いて Web 経由で機器の遠隔操作を行う、簡易模擬システムを試作した概要と、その構築までのプロセスの報告である。

II 組み込みシステムにおける OS の役割と現状

先ず初めに、組み込みシステムにおける OS の役割について考察してみるが、その前になぜ OS が必要かを考えてみる。

システムに OS を組み込んでいない場合、その一番のデメリットは、システムのタスクスケジューリング（いわゆるカーネル）やファイル・通信処理等の上位インタフェース（いわゆるミドルウエア）を、全て一から開発しなければならないということである。このことは、小型のシステムや軽負荷のシステムでは問題にならないが、近年のように高機能なミドルウエア、例えば TCP/IP ネットワークや USB 等パソコン用インタフェースが機能として要求されている場合は、OS の必要性が増してくる。

組み込みシステムに OS を採用する場合の必要な機能を大別すると、①ハードウエアを制御するカーネル部分、②外部インタフェースを制御するミドルウエア部分、③開発ツール、となるが、組み込みシステムのハードウエアは、パソコンの場合のように統一されたアーキテクチャが無い。よってカーネル部は専用部分となるために、OS を採用する最大のメリットは、ミドルウエアと開発ツールということになる。特にこの部分は、アプリケーションのクロスプラットホーム化を行う上で重要な部分である。

ミドルウエアと開発ツールがパソコン用 OS と互換性があれば、幅広い組み込みシステムにおいてネットワーク機能等高機能化が可能になるが、国内で設計される組み込みシステム用 OS で有名なのが、パソコン用 OS とは異なる ITRON という OS である。（社）トロン協会による RTOS の利用動向に関するアンケート調査結果（2000年11月）によると、約50%が ITRON 系、約20%が OS 無し、その他専用 OS（VxWorks 等）数種類がいずれも10%以下である。

このアンケート調査結果からもわかるように、国内では多くの場合 μITRON が使われていた。この要因は、性能的な側面も重要な部分を占めているが、ターゲットとする組み込みプロセッサが多種多様で、ミドルウエアと開発ツールの統合環境が、μITRON ベースのプロセッサメーカ製が多いという側面も大きく作用していると考えられる。

この μITRON というのは、厳密にいうと OS というよりも API（システムコール）の標準仕様のことで、パソコン用 OS の様な高い互換性は無く、世の中に溢れている便利なツールやアプリケーションを活用する等ユーザの利便性が少ない。

以上のことを分析すると、マイクロプロセッサの相違が、広く普及しているパソコン用 OS 互換のミドルウエアや開発ツールの組み込みシステムへの採用を妨げる要因となっていると思われる。

ところが、同アンケート調査によると、近年使用プロセッサが高性能なものに絞られてきているようであり、約36%が日立 SuperH（以下 SH）で、約16%が x86互換である。

以上の２つのプロセッサをサポートする OS（API や開発環境）が存在し、入手が容易であれば、組み込みシステムへのパソコン用 OS の導入が進むと考えられる。

III 組み込みシステム用 OS としての Linux

前述のように、近年導入が進んでいる、高機能組み込みプロセッサをサポートするパソコン用と互換性のある OS が存在すれば、通信等高機能なミドルウエアや開発ツール等がパソコン用のものを流用できる。候補としては、マイクロソフト社製 OS があげられるが、同社 OS は GUI に特徴があり、要求されるハードウエアが限定される。

近年、もうひとつの有力候補として注目されているのが Linux である。このような Unix 系 OS は、従来からパソコン用、特にエンタープライズ用として多く採用されているが、オープンソースのため、さまざまなハードウエアに移植されており、GNU/Linux on SuperH Project により SH マイコンにも移植された。産業用としての実績はまだ少ないが、その活躍する範囲が急速に広がっている。セットトップ・ボックスやホーム・ゲートウエイ、携帯型情報端末（PDA）などはもちろん、もっと小さな腕時計まで Linux で動かす試みが出てきているほどである。（図１）

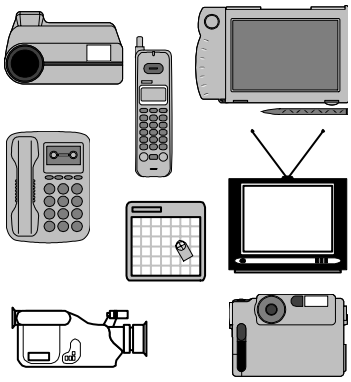

図１　組み込みシステム応用分野

組み込みシステムの最大のターゲットは携帯電話である。この分野を制する OS が事実上の標準になることが想像できる。現状では、μITRON ベースのメーカ製独自 OS が多いようだが、今日のように日進月歩に技術革新が進み、組み込みシステムでもインフラのブロードバンド化に伴うマルチメディアデータの取り扱いが必須になる等、複雑かつ高機能化するシステムに対応するには、独自 OS では限界がある。

携帯電話に代表されるような組み込みシステムが、パソコンのような急速な高機能化を果たすためには、ミドルウエアと開発環境の標準化が不可欠である。現状を分析すると、マイクロソフト社製 OS か Linux 等 Unix 系 OS しか残らないと思われる。

Linux の場合、最大の特徴はオープンソースソフトウエアであるという点である。この特徴は、さまざまなハードウエアに合わせて改良することが容易で、自由にコントロールすることが可能であるという優位性を持つ。反面マイクロソフト社製 OS と比較すると、ミドルウエアや開発環境が複数存在し、互換性が薄いということもいえる。ただ、豊富なミドルウエアや開発環境が容易に入手可能な点やロイヤリティフリーの点は、導入に際する敷居を低くする意味で大きなアドバンテージがあり、組み込みシステムにおける Linux の採用が急速に増えると見込まれる。

IV　組み込み Linux の現状

これまで、今後の組み込みシステムには高機能化が不可欠である点を明らかにしてきたが、制御用として要求される機能は、リアルタイム処理である。ここで言うリアルタイム処理とは、マルチタスク処理も含むが、本質的な意味は、アプリケーションがハードウエアを忠実にコントロールできるかという点である。つまり、マイクロプロセッサの割り込み処理をコントロールできるかどうかということになる。

この点を考えると、μITRON 等市販の専用 OS か、従来からあるシングルタスク OS であるパソコン用の DOS しか選択肢が無く、現状多く普及している GUI ベースのパソコン OS は選択できなくなる。なぜなら、現在普及しているパソコン用 OS は、システムの安定性の関係でカーネルがすべてのハードウエアを制御しており、アプリケーション側からハードウエアを制御することを許可していないからである。

通常の Linux カーネルも同様で、厳密な割り込み処理に対する応答時間を保証できていないため、リアルタイム処理は難しい。しかし、前述のようにオープンソースのため、目的に合わせて改良することが許されており、実際、ハードウエアリアルタイム処理を実現するために拡張された、RTLinux という OS が存在する。又、無駄なプロセスを外し、プロセス優先度を工夫するなど調整を行うことにより、数十 ms 程度の割り込み処理なら追従できると見込まれる。

組み込み Linux の現状分析をまとめると、メリットとしては、

①オープンソースでありシステムにあわせて改良できる
②マルチメディアデータ等重要性が増しているファイルシステムユーティリティが豊富
③ネットワークユーティリティが豊富
④開発環境ツールが豊富で入手が容易

という点であり、デメリットとしては、

①システムリソース、特にメモリを必要とする
②システムブート／シャットダウンに時間がかかる
③通常の Linux カーネルはハードウエアリアルタイム処理を保証していない
④組み込みマイコンエンジニアは初期導入に対して、より情報技術の知識が要求される

という点である。

組み込み Linux の他の商用 RTOS との性能比較は不明だが、オープンソースであり、ハードウエアの性能を引き出すという意味では、十分に組み込みシステムに採用するメリットがある。

V　組み込み Linux アプリケーション例（システム構成）

ここでは、簡易模擬システム試作例として、企業人スクールで教材として取り上げた、組み込み Linux を制御装置兼マイクロサーバとする、Web を使った

遠隔入出力制御装置の構築例を３例示す。

３例とも、共通仕様は卓上での実習用システムとし、システム構成は、

①制御装置兼マイクロサーバ
②簡易入出力装置
③操作用クライアント

となる。（図２）

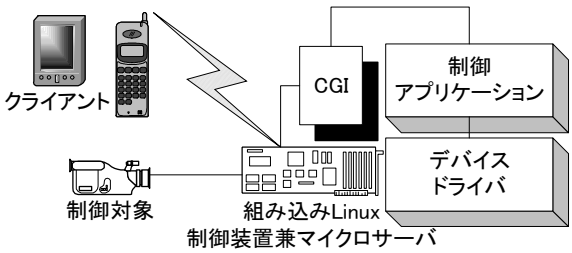

図２　組み込み Linux アプリケーション構築実習用システム

ネットワークはイーサネットでプロトコルは TCP-IP、サービスは HTTP 及び FTP を基本とする。なお、３例ともシステムは模擬システムのため、ネットワーク間にはゲートウエイ等を介さず、セキュリティも考慮していない。

システムの動作は以下の通りである。

①インターネット接続可能な Web ブラウザを搭載したクライアントから、制御装置兼マイクロサーバに対し制御指令を HTTP リクエストとして送る
②制御装置兼マイクロサーバは、クライアントからの HTTP リクエストを CGI もしくは Java で処理し、制御用アプリケーションに指示を送る
③制御用アプリケーションは、制御用モジュール（デバイスドライバ）とシステムコールを介して通信を行い、ハードウエアをコントロールする
④制御装置兼マイクロサーバはハードウエアの制御結果を HTTP レスポンスとしてクライアントに返す

以上のような動作により、Web を介して制御装置をコントロールすることが可能である。

このシステムは、卓上の実習用システムであるが、応用として、生産ライン監視システムや Web カメラ遠隔監視システム等が考えられ、実際にホームセキュリティ装置等実用化されているものもある。

VI　組み込み Linux アプリケーション例（ハード構成）

実際に構築した３例の基本システムは同じであるが、大きな違いは、組み込み Linux を導入する制御装置兼マイクロサーバに異なる３種類の CPU を使用している点である。

これは、組み込みシステムに OS を使用しない場合の CPU の相違、つまり開発ターゲットのプラットフォームの相違というのは、開発する上で非常に大きな問題となるが、組み込み Linux を採用するにあたり、どれだけ開発上の利点があるか、もしくは不具合があるかを考察するためである。

以下、３例をモデルケース１、２、３、と呼び、そのハードウエア仕様は以下の通り。

①モデルケース１（図３）
x86互換 CPU を採用した AT 互換ボードコンピュータ　ADVANTEC（PCM5820）
PC104仕様拡張ボード　ADVANTEC（PCM3730）
自作簡易入出力装置
ロボットアーム　共立電子（MR－999）

②モデルケース２（図４）
SH３（SH7708R）を採用した教育用組み込みマイコンボード　有限会社りぬくす工房（CAT68701SBC-SH３）
SUNLIKE 製液晶モジュール（SC1602B）搭載自作簡易入出力装置

③モデルケース３
プロセッサコア ARM720T 採用の Linux 対応小型マイコンボード　梅澤無線電機株式会社（HT1070）
PC104仕様拡張ボード　SUNLIKE 製液晶モジュール SC1602B 搭載　梅澤無線電機株式会社（HT1071－U00）
ロボットアーム　共立電子（MR－999）

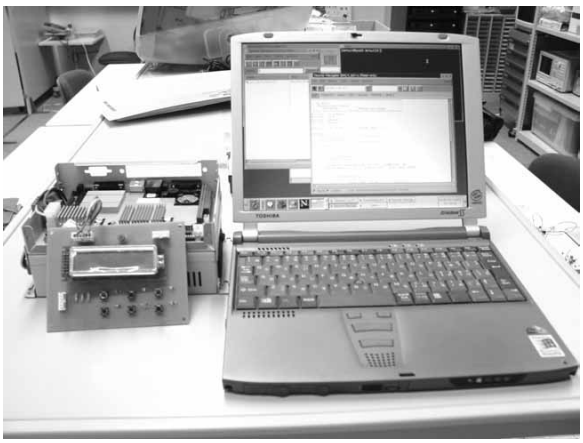

図３　マイコンボードと開発システム

SH、ARMともに組み込みに適する、小型省電力32ビットRISCマイコンである。互いに用途に応じて数種類存在し、それぞれ特徴的な機能を備えているが、今回使用した装置はOSを搭載することを前提にした、性能や基本構成が比較的似ている装置を選定した。

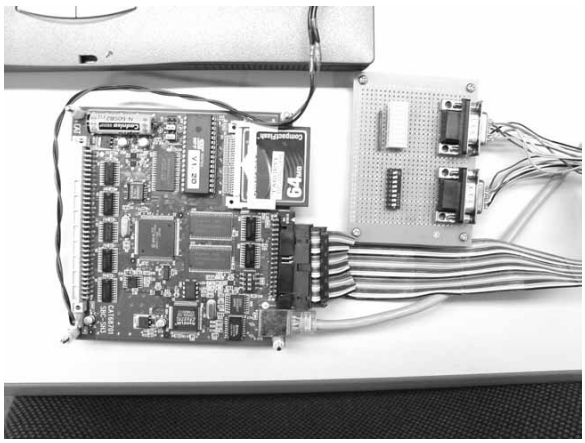

図4　マイコンボード実習用システム

Ⅶ　組み込みLinuxアプリケーション例（詳細）

ソフトウエアの構成は、大きく分けて、
①制御装置兼マイクロサーバ用システム（以下ターゲットと呼ぶ）
②操作用（クライアント）システム
③開発環境システム（以後母艦と呼ぶ）
の3つに分類できる。

ターゲットシステムソフトウエアは、交換可能なコンパクトフラッシュROMディスク（以後CFディスクと呼ぶ）上に構築する。

母艦とターゲットはCPUが異なるため、ソフトウエアの開発は母艦により行い、開発したソフトをターゲットに転送してデバック等確認を行う、いわゆるクロス開発という手法をとる。

母艦とターゲット間のLinuxの種類の相違（ディストリビューション、カーネル等）は、CGI、Java等ユーザアプリケーションの場合それほど問題にならないが、ハードウエアを取り扱うデバイスドライバは問題が発生する場合が多い。よって母艦とターゲットは同じ仕様にする。具体的には、GNUラインセンスである点とパッケージ管理が容易な点より、「Debian GNU/Linux2.2」を採用した。

操作用システム（クライアント）は特に規定はしていないが、NTT DoCoMo,IncのiモードからのHTMLができることを前提にしており、cHTMLリクエストを処理できるようにしている。

1.モデルケース1

1.1　ターゲットシステムの構築

モデルケース1のターゲットはx86互換である。よって、LinuxをCFディスクに組み込むだけならパソコンのシステムとあまり相違がない。よって、カーネルにリアルタイム処理のパッチをあてたRTLinuxの導入を試みた。

本ケースの試みは、RTLinuxと他のRTOSとの性能比較ではなく、システム導入からアプリケーション構築までのプロセスを検証するものである。

はじめに、母艦としてノートパソコンを用意し、この母艦にターゲットイメージを構築する。
①PCMCIAカード内蔵ノートパソコンに「Debian GNU/Linux2.2」を導入する。このときパッケージの選択に「カーネルソース」「PCMCIAモジュール」を必ず導入しておく。又母艦は、汎用シリアルポートと、イーサネットポートを有している必要がある。
②母艦にCFディスクを挿入し、フォーマットした上でマウントする
③Linuxのソースディレクトリに RTLinuxのパッチをあてたソース（rtlinux−2.0−prepatched.tgz、rtlinux−2.0.tgz）を展開し、リンクを張りなおす
④Debianパッケージを利用し、必要なアプリケーションをCFディスクのマウント先（／mnt）上に構築する。方法としては、chroot等を使ってターゲットの仮想環境を構築する手法を用いる。なお、ターゲットシステム（PCM5820）向けに、あらかじめ構築されたベースイメージが公開されている場合はそれを利用する方法がある
⑤カーネルをターゲットに合わせてコンパイルし、できたカーネルイメージとモジュールをCFディスクのマウント先（／mnt）にコピーする
⑥ブートローダをCFディスクのマウント先（／mnt）にコピーし、設定ファイルを修正する

以上でCFディスクにRTLinuxカーネルのDebianが実装されたので、これをボードコンピュータに挿すと、このボード上でRTLinuxが実行可能になる。

実装した主なアプリケーションは、Httpd、Ftpクライアント、サーバ、Telnetクライアント、サーバ、Perl、等である。

後述するオリジナルの制御用アプリケーションは、

RTLinux 対応のデバイスドライバとして構築し、自作の入出力装置とシリアルポートを制御する。

1.2 開発環境の構築

前述のプロセスで、母艦は「Debian GNU/Linux 2.2」ベースの RTLinux ソースファイルが展開されている。この状態で、母艦に汎用 Linux 開発環境を導入し、構築したアプリケーションを FTP でターゲットに転送する。デバック環境は、ICE 等を使わないソフトウエアデバックを用いる

主な開発手法は以下の通りである。

①フリーの開発環境である「SourceNavigator」をインストールする

②任意のディレクトリに、プログラムソース・make ファイル・SourceNavigator 用プロジェクトファイルを作成する

③SourceNavigator を使って、ソースのコンパイル・デバッグを行う

④母艦でコンパイルした実行ファイルを、ターゲットであるボードコンピュータへ FTP 転送する

⑤ターゲット上で実行テストを行う

2.モデルケース２　SH３マイコンを使う

モデルケース１では、ターゲットに x86互換マイコンを搭載し組み込み用に特化した小型ボードコンピュータを使用した。このボードコンピュータは組み込み用に考慮されているので、消費電力等十分組み込みシステムに耐えうるハードウエアではあるが、基本的に AT 互換であるので、例えば VGA コントローラやキーボードコントローラ等、無駄なハードウエアが存在する。ここでは、より従来のマイコンに近く、組み込みシステムに多くの実績がある SH マイコンベースのボードを採用したい。

本ケースでは、Linux をサポートしている SH マイコン搭載ボードコンピュータで、GNU/Linux on SuperH　Project のテストシステムとして利用されている、CAT68701（有限会社りぬくす工房）を使用した。

主な仕様は、①フラッシュ ROM（コンパクトフラッシュ）インタフェース、②DIO（入力11点出力14点）、③シリアルインタフェース、④10BaseT イーサネットインタフェース、となっている。最大の特徴は、BIOS も含むすべてのソフトウエアがオープンソースソフトウエアであるところである。

2.1ターゲットシステムの構築

CAT68701に実装する Linux は、Debian ベースのディストリビューションである「Debian GNU/Linux for SuperH」で、CF ディスクに実装するための手順は、モデルケース１の場合とほぼ同様である。

汎用アプリケーションの追加削除は、Debian GNU/Linux より．deb パッケージを用いてインストールする。実装した代表アプリケーションはモデルケース１とほぼ同様である。

オリジナルの制御用アプリケーションは、カーネルが RTLinux ではないので、通常のデバイスドライバ開発手法を使用して構築し、LCD（SC1602B）を実装した自作入出力装置とシリアルポートを制御する。

2.2開発環境の構築

制御用ソフトウエアの開発は、母艦を使ったクロス開発で ICE を使わないソフトウエアデバックを用いるが、CF ディスクに余裕があればセルフ開発も可能である。どちらにしてもクロスアセンブラとリンカ（binutils)、クロスコンパイラ（gcc)、デバッカ（gdb）をインストールする必要がある。

3.モデルケース３　ARM マイコン（ARM720T）を使う

ARM プロセッサは、x86互換、SH３等と異なり、SoC（システムオンチップ）内で IP コアとして実装されている。よってチップとして基板上に発見することは難しいが、携帯電話等さまざまな組み込みシステムに利用されている。

本ケースで使用したターゲットは、梅澤無線電機株式会社の HT1070とオプションボード HT1071－U00で、前述の CAT68701より実システムに採用されることを意識したマイコンボードである。

基本仕様は、CAT68701とほぼ同様で、主な仕様は、①フラッシュ ROM（コンパクトフラッシュ）インタフェース、②パラレルポート（８bit)、③シリアルインタフェース、④A/D コンバータ（10bit 8 ch）10BaseT イーサネットインタフェース、となっている。最大の特徴は、オンボードフラッシュ ROM に最小ブートシステムがあらかじめ導入されており、すぐにネットワークアプリケーションを構築することが可能である点である。

オンボード ROM システムのみでアプリケーションを構築してもよいが、ARM プロセッサは Debian GNU/Linux において、そのパッケージがリリースされていることを考慮すると、拡張性などから前例と同じく Debian　GNU/Linux2.2ベースのシステムを導入した。

ターゲットシステム導入手法および開発環境構築手法は前２ケースとほぼ同じであるが、ターゲットのベースイメージや汎用アプリケーションがメーカから提供される。

前例と同じくLCD（SC1602B）がオプションボード上に搭載しており、サンプルプログラムを参考にそのデバイスドライバプログラムを作成した。

4.制御用アプリケーションの構造

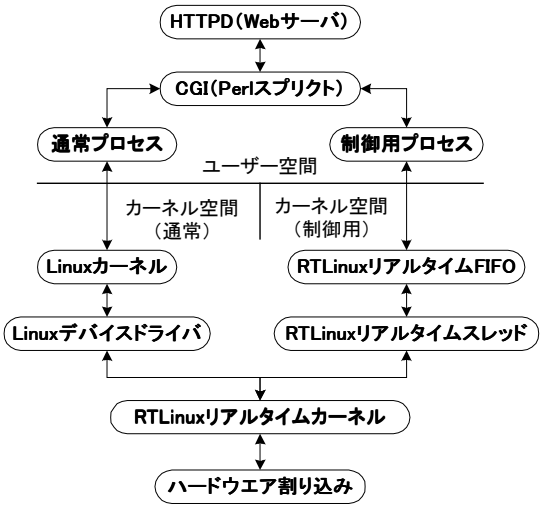

図５　制御用アプリケーションの構造

Linux等UNIX系OSでは、ユーザーアプリケーション（プロセス）が動作するユーザー空間と、ハードウエアを制御するデバイスドライバが動作するカーネル空間が分離され、保護されている。よって、Linux上で制御用アプリケーションを構築するには、カーネル空間にあるデバイスドライバとユーザー空間にある制御用プロセスの２つをペアで制作する必要がある。

制御用のデバイスドライバは、モジュールと呼ばれるダイナミックリンク方式で実装する。更にデバイスファイルとデバイス番号をカーネルに登録することにより、カーネルから制御用デバイスドライバをコントロールすることが可能になる。ユーザー空間にある制御用プロセスは、システムコールを使い、カーネル空間にある制御用デバイスドライバとの通信を行いながら全体の制御動作を行う。

以上が、一般的なLinuxからハードウエアを制御する方法であるが、モデルケース１のターゲットである制御装置兼マイクロサーバには、ただのLinuxではなくRTLinuxが実装されている。

一般的なLinuxでは、ユーザープロセスとデバイスドライバはシステムコールを介して通信できるが、デバイスの制御権はカーネルにあり、ユーザープロセス側にはハードウエアを制御する権限が与えられていない。ところが、ハードウエアリアルタイム処理のパッチをあてたRTLinuxでは、ハードウエアの制御権を独自のリアルタイムカーネルで直接処理している。

よって、このリアルタイムカーネル用に制作したデバイスドライバ（リアルタイムスレッド）では、Linuxカーネルに依存せずにハードウエアを制御でき、ユーザープロセス間との通信は、専用のリアルタイムFIFOを介することにより、ユーザー空間からハードウエアを制御することが可能になっている。（図５）

３ケースとも、ターゲットにはHttpdとして簡易Webサーバ「Boa」を実装しており、index.htmlを／var/wwwに置き、リクエスト処理を行うCGIをPerlで記述し／usr/lib-cgi/binに置く。

CGIと制御用アプリケーションのインタフェースは、システムコールオペレーションと中間ファイルによるファイルIOストリームを用いる。

モデルケース１での制御用アプリケーションは、
①PC104仕様拡張ボードに接続した自作簡易入出力装置を制御するRTLinuxリアルタイムモジュール
②ロボットアームのリモートI/Oボード制御用シリアル通信モジュール
③PerlスクリプトでHttpリクエストを処理するCGI、④CGIとデバイスドライバとのインタフェースプログラム
により構成される。

モデルケース２と３での制御アプリケーションは、ケース１と同様であるが、LCDを制御するを追加した。

Ⅷ　組み込みLinuxアプリケーション例（Java応用）

前述の３ケースにおける実習用システムでは、CGIをベースにHTTPプロトコル通信により、Webアプリケーション制御システムを構築している。

このシステムでも、実システムに十分応用可能であるが、クライアントからのHTTPリクエストがない限り、制御動作を行うことができないという欠点がある。又、制御装置側もCGIによるスプリクト処理を行っており、制御状態やクライアントの状態により任意に変化したHTTPレスポンスを返すことが困難である。つまり、本ケースにおける実習システムは、状況に応じたインタラクティブなシステムとはいい難

い。

そこで、モデルケース１とモデルケース３における実習システムを、Java ベースのシステムへ改良することを試みた。Java は、JavaVM 上ではクロスプラットホームで動作するシステムであり、本ケースのような組み込みシステムには最も適したものである。

ターゲットシステム上で Java が動作すれば、クライアントから HTTP リクエストを送らなくても制御装置側からクライアント側に動的レスポンスを送ることが可能になる。同様に、操作用クライアント上で JavaVM が動作すれば、HTTP リクエスト送信処理を自動化できる。特にクライアントに JavaVM を搭載した携帯電話を対象とすると、本例における実習システムはより実用的なシステムになる。

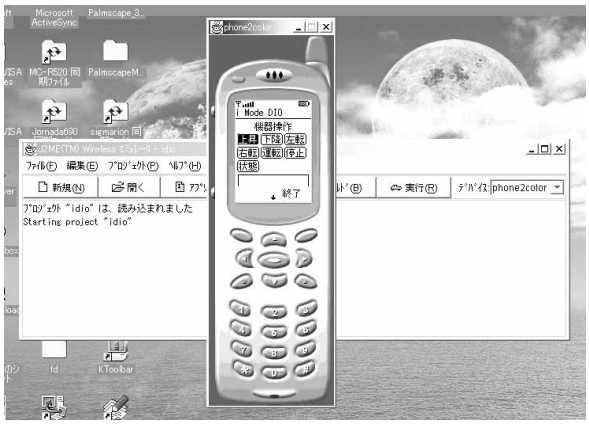

図６　i アプリ開発環境 DoJa

具体的には、ターゲットシステムの JavaVM には、オープンソースで移植も可能な Kaffe、コンパイラには、Kaffe に対応した Jikes を導入した。クライアントは、本物の携帯電話を実習用機器にするのは難しいため、NTT　DoCoMo,Inc 用の Java 動作環境「i アプリ」のフリーの Java 開発環境である J2ME Wireless SDK for the DoJa（以下 DoJa）と、そのシミュレータを使って、本例の実習システムにおけるタイマイベントによる動的 HTTP リクエスト処理の検証を行った。（図６）

IX　まとめ

今回のモデルケース試作において得た知識をまとめると、以下の通りである。

①Linux を組み込みシステムに導入する際に要求されるハードウエアリソースの制限は、導入するパッケージを必要最低限にする等工夫によって、十分クリアできる。具体的には、市販の32ビット組み込み CPU ボードマイコン（MMU 搭載、システムクロック100M、システム RAM32M、フラッシュメモリ64M）相当の性能ならば、組み込みシステムに Linux を導入可能である。

②リアルタイム処理性能については、μITRON 等の制御用 RTOS と根本的に成り立ちが異なるため、単純に比較はできないが、他のパソコン用 OS より組み込みシステムに向いている。理由としては、オープンソースで用途に合わせて改良できる点と、Linux カーネルの設計思想が軽く早くすることに主眼をおいているため、組み込みシステムにおいても対応可能な点である。

③組み込みシステムに Linux を導入するかどうかは、リアルタイム性能と汎用性、拡張性、高機能性とのトレードオフの問題である。筆者は μs 単位のリアルタイム性能を要求するアプリケーションは、組み込みシステム全体の割合からするとそれほど多くないと考える。よって、他の RTOS と組み込み Linux は棲み分けが可能と思われる。

④組み込み Linux を用いたシステム開発は、プラットホームに対する意識を少なくすることができる

⑤RTLinux でのデバイスドライバは、通常カーネルのドライバと構築方法は大きく異なるが、可読性がよい。又、プラットホームの違いによるデバイスドライバのソースレベルでの互換性は少ない。

⑥Linux は、もともと大規模システムにも対応可能なため、組み込み制御用途に対しては、全体としてオーバーヘッドがかかり、システムが重くなる。特に JavaVM は起動時のオーバーヘッドが大きく、アプリケーション起動に数秒単位の時間がかかる。

X　おわりに

組み込みシステムによって実現されるユビキタス・ネットワークの基幹技術は、従来の情報技術から、より電子技術に近い領域へシフトされる。

つまり、従来は組み込みシステムといえば電子技術で、Web アプリケーションといえば情報技術という棲み分けであったが、これらの垣根が低くなり、相互のトレードオフが重要になってくる。

「組み込み Linux を使った Web アプリケーションの開発」という技術的題材は、家電ロボット等ホームオートメーション化のキーテクノロジーになるのは明白である。又能力開発の上でも、電子技術と情報技術とのトレードオフの問題を解決する重要なテーマとなる。

［参考文献］

⑴ 日本エンベデッド・リナックス・コンソーシアム
http : //www.emblix.org/

⑵ ITRONProject　http : //tron.um.u-tokyo.ac.jp/TRON/ITRON/home-j.html

⑶ 有限会社りぬくす工房
http : //www.si-linux.com

⑷ Armadillo Official Site
http : //armadillo.atmark-techno.com/

⑸ Software　Design、CQ 出 版 社、2002年10月、p. 158－174

⑹ NTT ドコモ i アプリ技術資料
http : //www.nttdocomo.co.jp/mc-user/i/java/index.html