K-009

ロボットを用いた組込みシステム学習支援環境の開発

Development of Educational Support Environment of Embedded System with Robot

川上亮太郎† 田中祐樹† 早川栄一‡
Ryotaro Kawakami† Yuki Tanaka† Eiichi Hayakawa‡

1. はじめに

近年，様々な場面で組込みシステムが用いられるようになってきており，そのスキルを持った技術者の需要が高まっている．しかし，情報処理[1]によれば日本における組込みソフトウェア技術者のスキル標準があると考えているのは 20%に満たず，企業ではスキル習得を重要視している．また，学校教育の中でも組込みシステムやリアルタイムソフトウェアの学習が重要視されており，講義や演習に取り入れられているが，効果的な結果を生んでいない．その理由としては次の 3 点があげられる：(1) 他の学習対象と比較して内部や結果が見えにくく，学習者が興味を持ちにくい，(2)広範囲に渡る知識が必要であり，講義や演習の時間内に内容全体を消化しきれない，(3)学習に向いた開発環境がサポートされていない．

既存の学習では brickOS[2]，legOS[3]などが多く用いられているが，上述した問題が解決されていない．

本研究ではそれらの問題を解決することを目的とする．

2. 設計方針

上述した問題を解決するため，本研究では次の方針を挙げる．

(1) 学習向きのロボットプラットフォームの提供
(2) 小規模なソフトウェア教材の提供
(3) ロボット向けのデバッグ環境の提供
(4) オブジェクト指向環境でのアプリケーション作成の支援

3. 教材を用いた学習

本システムを用いた場合のコースウェアを図 1 に示す．

本システムでは C 言語や Java 言語の知識，センサ工学，ロボット工学の知識を前提としている．その上で，本システムを用いた学習では次の段階を通してシステム全体を学習していく．

(1) 制御プログラミング

オペレーティングシステム(以下 OS)および開発環境のサービスを用いたロボット制御プログラミングや各種センサ，ハードウェアの操作を学ぶ．

(2) 組込みシステム

組込みシステム学習では OS をはじめとしたシステムソフトウェアがどのように動作しているかを学習する．ここでは，本システムについてコードリーディングを行ったのちにリアルタイム機能の追加を行う．

(3) 高度な制御プログラミング学習

これまでの学習結果を活かした高度な自律制御アプリケーションの作成を行う．大規模なプログラミングを容易にするために，オブジェクト指向言語を用いたプログラミング環境を利用する．

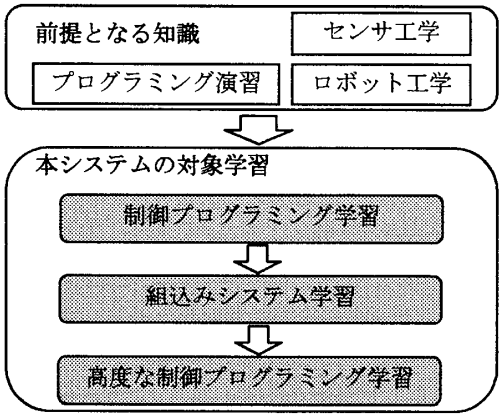

図 1 コースウェア

このようにカリキュラムや，学習者の興味や意欲，能力にあわせた課題の設定や実現が可能になる．

4. 全体構成

全体構成を図 2 に示す．我々が試作したロボットを図 3 に示す．

組込みボードは，教材として複数の実装学習に利用されることを考慮し，組込みシステムとしては処理能力が高く，メモリ容量の大きいもの，かつ OS の講義で使用されることを考慮し，仮想記憶が使用可能なもの，また，無線 LAN が使用できるものを選択した．

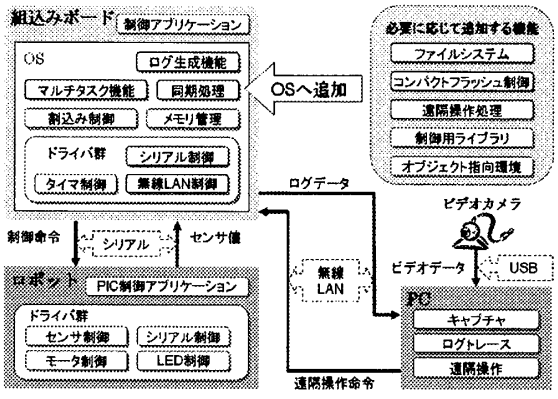

図 2 全体構成

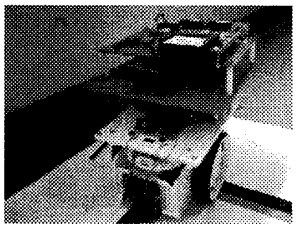

図 3 試作ロボット

† 拓殖大学大学院工学研究科，
Graduate school of Engineering, Takushoku University
‡ 拓殖大学情報工学科，
Faculty of Engineering, Takushoku University

本システムは応用例として，ロボサッカーを想定している．サッカーでは外部状況を判断するためにセンサ，移動のためにモータが用いられる．これを用いることで組込みシステムの基本を学ぶことができる．本システムでは,タッチセンサ，赤外線センサ，可視光センサ，モータ，LED，スイッチ，ブザーを装備している．

5. システムの設計

5.1 OS

教育用組込みOSについて述べる．

(1) リアルタイムスケジューラ

リアルタイムスケジューラは組込み OS では必須の機能である．本システムでは基本的な優先度付きスケジューリング，EDF(Earliest Deadline First)スケジューリングをサポートする．スケジュール可能性判定についてはシステム内部で行わない．また，デッドラインミスへの対応,プライオリティインバージョンへの対応も実装しない．それによりコードを小さくする．

(2) デバイスドライバ

本システムで扱うデバイスはタイマ，シリアル通信，センサである．

タイマはスケジューリング時のトリガ，時間の計測など，重要な要素である．

シリアル通信は通信インタフェースとして単純である．また，現在のロボットの構成上必要である．受信だけをバッファリングする，オーバーランなどのエラー処理を省略することで単純なコードとした．

センサについてはドライバボード上の PIC コントローラにドライバのコードを搭載することでロボットの構成を単純化し，ロボットの試作を容易にした．OS 側ではインタフェースを切り分け，センサボードをアップグレードした場合もコードの変更が少なくてすむようにした．

(3) 割込み処理

割込み処理は，ハードウェアの割込みベクタを仮想化した形で提供する．これは，OS 自身の移植性を向上させるとともに，ネットワーク装置などのデバイスの割込み処理を，OS のコードから隠すために用いる．学習者は分かりにくい割込み処理の詳細に触れずに，割込みを用いたプログラムを読んで理解することができるようになる．

(4) 起動環境

フラッシュ上にはネットワークおよびシリアル通信を通したブートプログラムだけを用意し，OS のコードは無線 LAN から転送可能にした．これによりフラッシュ ROM の書換え制限を気にせずに，OS の機能を試すことが可能である．

これらの機能はコードリーディングの妨げにならないようにコードサイズに制限をつけて実装を行う．ここで述べた機能以外は必要な場合に起動時に組み込む．

5.2 デバッグ環境

本システムではログ解析アプリケーションを用いてログ解析を支援する手段を学習者へ提供する．

学習者にとっては外部動作からプログラムの振舞いを知る手段が必要となるが，ロボットでは装置の動作から内部動作がどのようになっているかを知らなければならない．また，教授者は学習者全員の動作をチェックすることは難しい．

これに対して，本デバッグ環境では，動作中の内部状態をログとして残し，デバッグ時の参考データとして学習者へ提供する．ログからの特定の時間におけるプログラム内で実行中の関数名，スレッドID，ロボットに搭載されたセンサの値が取得できる．この情報は組込み OS 自体にも適用できるので，OS の内部までのログトレースが可能である．さらに，実行時の動作の動画をログと同期して学習者および教授者へ提供する．学習者は再現性の低いバグを何度でも見ることが可能になり，プログラムと動作との関係を理解することできる．また,教授者は，動作と合わせてプログラムをチェックできることから，再現性がないタイプのバグに対しての対処がやりやすくなる．

図4に実行画面を示す．

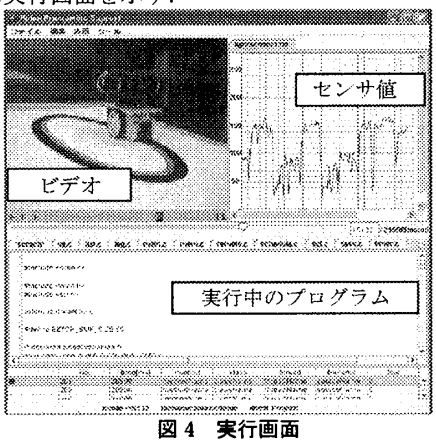

図4 実行画面

6. 実現

開発した実行環境および，OS のステップ数を表1に示す．

表1 開発した OS のステップ数

機能	ステップ数
マルチタスク	374
割込み	688
ドライバ群	681
ロボット制御用 API	233
合計	1976

表1において，学習者がコードリーディングの対象は OS カーネルであり，ライブラリ，ドライバ群は対象としていない．これまでの演習の経験から，学部の学習者が半期でコードを読み理解できる上限は 2000 行程度であることを考えると適切なサイズであることを示している．

7. 今後の課題

本稿では従来の組込みシステム学習における問題点について述べ，それらを解決するための環境と教材の構成について述べた．現在はプロトタイプの作成を行っている．今後はコードの不要箇所の削除,ドキュメントの作成をおこない，システムの評価をとる．

参考文献

[1] 情報処理学会，"情報処理 Vol.46 No.2"，feb.，2005
[2] brickOS, http://brickos.sourceforge.net/
[3] legOS, http://www.noga.de/legOS/