社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－CE－98 (19)
2009／2／28

情報系工学科におけるロボットを用いた組込みシステム教育の実践

早川 栄一　高橋 丈博　青嶌 健一
拓殖大学　工学部　情報工学科

{hayakawa, take, kaoshima}@cs.takushoku-u.ac.jp

本報告では，拓殖大学工学部情報工学科において行ってきたロボットに用いた組込みシステム教育の実践例を紹介する．情報系学科において，制御をはじめとした組込みシステムに関する教育は重要度を増しているが，その一方でハードウェアに対する知識不足，プログラミングの難しさから積極的に取り組ませるのが難しいという問題がある．本学では，1997年より学部3年生の実験において，Mindstormsなどのロボットを用い，学生に興味を持たせることによって，主体的に課題に取り組めるカリキュラムを運営してきた．また，ここ2年間は，組込みシステムのプログラミングに加えて，ソフトウェア工学的な要素も取り入れてきた．学生からの評価では，この実験は，おおむね高評価であった．その一方で，実践における問題も生じてきている．この問題についても合わせて報告する．

Experiment on Embedded System Education using Robot in Computer Science Course

Eiichi Hayakawa, Takehiro Takahashi, Kenichi Aoshima
Department of Computer Science, Faculty of Engineering, Takushoku University

This paper describes the experiment of embedded system education with robot in Takushoku University. Although increasing the importance of embedded system education in computer science course, it is difficult to build the embedded system course for the lack of knowledge about hardware or the difficulty of embedded programming. In our department the robot experiment has been built at the 3rd grade of undergraduate since 1997. Additionally software engineering factors are added in our experiment. The experiment is almost highly evaluated by students. On the other hand, problems are found in the experiment. The problems are also described in this paper.

1．はじめに

学習のモチベーションを持たせつつ，必要な知識を習得させることは，学習カリキュラムの設計にとって非常に重要なことである．近年，強い動機を持たずに専門学科に入学してくる学生の増加に伴って，彼らの意欲を維持させつつ，主体的にテーマに取り組ませるというのは，スタッフにとっての大きな課題となっている．特に，システムプログラミングなどの理解が難しい科目では，興味を持たせて取り組ませることは難しい．

さらに，チームワークによるコミュニケーション能力の育成，文書化，ソフトウェア工学などの主題については，社会的な要求がある．これらの要求を満たす科目の一つに，実験科目がある．実験科目では，演習という形態を取ることから，より主体的に取り組ませることが期待できる．その一方で，スタッフ数や授業時間の問題から，学力の分散の大きい学生群に対して一つのテーマを与えざるを得ないという問題があり，テーマの設定は難しい問題となっている．

拓殖大学工学部情報工学科において，著者らは3年次の情報系実験において，1997年度よりLEGO Mindstorms（以下，Mindstorms）含むロボットを用いた組込みシステムに関する実験を継続して行っている．この実験では，組込みシステムを対象として上記の問題の解決を目指す科目の一つとして設定し，運用してきた．

本報告では，情報系工学科においてロボットを使った組込みシステム教育の実践例を紹介するとともに，そこで発生した問題などについても述べる．

2．実験の経緯

本実験は，学部3年生の実験カリキュラムである情報工学実験 II[1]の1テーマとして設置したものである．この実験では，学生の希望によってテーマを選択し，半期（週1回，12週）を2回行うものである．

3年次の実験では技術の習得に加えて，ある程度学生に自由を与え，裁量を与えるテーマを多く設定している．4年次の卒業研究では学生自身の自主的な取組み姿勢がもっとも重要であり，本実験はその前段階であるという位置づけである．その分，教員側はテーマの設定および指導に工夫が必要となる．

1997年に本実験を再設計する機会があったときに，実験テーマに対する要求事項として，次の項目が挙がった．

- それまでに培った知識を用いて，実験に自

主的に取り組めること．特に，2 年生までに習得したプログラミングおよび専門知識を応用可能であること．
- 半期程度で形になること．学生が中途半端で終わるのではなく，どのようなものであれ，最後まで形にできるものであること．
- 学生の自主性をある程度活かせるテーマであること．

このような要求に基づき，1998 年より教育用ロボットを用いたロボット作成および簡単な制御プログラミングの実験のテーマとして「自走ロボット」を立ち上げた．この目的としては，制御プログラミングの基礎技術の習得，およびプログラミングがあまり得意ではない学生でも，興味を持って取り組ませたいという 2 点である．

組込みシステム技術に関わる技術者は今後増える傾向にあり，それに伴ってソフトウェア技術者のニーズも高くなっていく．これまで本学科内にはこのようなカリキュラムがなかったことから，3 年次の実験のテーマの一つとして設定した．

3．自走式ロボットを用いた実験

この実験では，systemload 社製の CZR720-JRP[2]を用いて行った（図 1）．ロボット，制御ボードの作成，および付属の C 言語を用いて，センサ入力，モータ出力，ライントレースによる制御プログラミングを学ぶことをテーマとした．このロボットでは，デバイスとしてフォトセンサとスイッチ，LED，およびモータがついており，Z80 互換の CPU で動作をする．

課題としては，黒い走行用のボード上に白いテープを複数本張り，それをランドマークとしてランドマークの変化に対して，ロボットがコースを変更するプログラミングを課題として提示した[3]．割込み関連の課題も考えたが，アセンブリ言語を用いた実習は難しく，C 言語から割込みを適切に扱う方法がないという問題もあり，行わなかった．

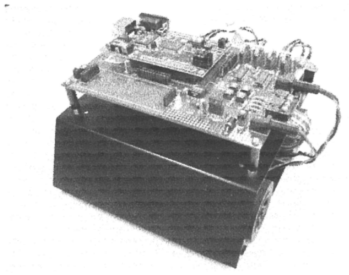

図 1 CZR720-JRP

このテーマを数年間行っている中で，次の問題が浮上してきた．

1. プログラミング環境が貧弱であった．付属のＣコンパイラは，構造体の機能がない，配列操作に制限があるなど，機能が制約されているもので，学部 3 年生が用いるものとしては不十分であった．
2. 学生が自主的に取り組んでいない．ロボットハードウェアが固定化されていることから，学生はハードウェアそのものにあまり興味を持てない．
3. ロボット自体のコストが高く，実験希望者の受け入れ人数を増やすことが難しい．
4. 半田付けによるハードウェア作成から，ロボットの調整，ソフトウェアの開発まで，幅広い分野が対象となり，対応できるスタッフや TA の数が少ない．

このような問題に対して，まずコンパイラをフリーソフトウェアの SDCC に変更した．しかし，採用時点で SDCC もバグが多く，実験環境として稼働させるのは難しいという問題があった．最終的には，AKI-H8 および gcc を採用した制御ボードを再設計し，変更することにより，1 の問題を解決することができた．その一方で，2ー4 の問題を解決することはできず，新しい環境が必要となった．

4．LEGO Mindstroms RCX を用いた実験

4．1　LEGO Mindstorms RCX の採用

3．のシステムの問題点を，解決するシステムについて検討を行った結果，2001 年度から，LEGO 社の Mindstorms RCX（以下 RCX）を用いた実験環境の構築および，実験カリキュラムを新設した．これは，次の四つの要求を満たすものである．

1. 学習者が魅力を感じるデバイスの提供
2. 情報系学科の 3 年生のスキルに見合ったプログラミング環境の提供
3. 一人一台以上利用可能な低コストの環境
4. 教授側での管理コストの低減

デバイスの制御を学習する教材ロボットとしては，ロボット自体にもある程度の自由度を持たせ，デバイスとしての面白さを提供する必要があった（これにより対象の難しさを乗り越えて，自主的に課題に取り組むことを期待できた）．また，その一方でトイではなく，学部 3 年生が持っている，もしくは持つべきスキルレベルに見合った環境がフリーソフトウェアとして提供可能なものである必要があった．さらに，故障が少なく，低価格で購入できなければならなかった．

このような要求に対して検討した結果，RCX を採用した実験を計画した．この章では，この実験について述べる．

4．2　開発環境

2001 年からの実験では，RCX 上で動作する基

盤として，Markus L. Noga が開発した legOS[6]（現在では brickOS[7]というプロジェクトになっている）および gcc の開発環境を採用した．2001 年当時，RCX 上でプログラムを実行する環境としては，次のものが存在していた．

- RIS(Robotics Invention System)[4]
- NQC(not quite C)[5]
- legOS(brickOS)
- leJOS[8]

RIS は，ビジュアルプログラミング環境を提供しており，マウスで部品を選択，接続していくことで，ロボットのプログラミングが可能になるものであり，ほとんどプログラミングの前提知識は不要である．また NQC は，C 言語に近いが独自の文法を持つプログラミング言語および環境であり，RIS で行える操作がほぼ利用可能である．

この中で，RIS は，情報系学科の 3 年生で用いるには，非力である．また，NQC は RCX ファームウェアの機能をすべて利用でき，安定していて，トラブルが少ないという利点があるが，言語仕様が従来のものと異なり，新しい言語を習得するコストが高いこと，また，環境自体のオーバーヘッドが大きく制御に向かないことを懸念して，採用は見送った．また，Java が実行可能な leJOS を採用し，Java で実験を行うことも検討したが，NQC と同様に言語習得のオーバーヘッドが大きいことや，OS が初期バージョンであり，教材としての採用は難しいと判断した．

legOS は，オープンソースで提供されている環境である．小型の OS であり，スレッドを提供している．また，センサ類については API がすでに実装されている．また，プログラムサイズが小さく，軽量であること，OS とアプリケーションを別々にロードできる，という利点がある．
本実験では，ロボット側の制御はこの OS 上でのプログラムとして実現するようにした．

また，開発環境は gcc をベースとした Linux 環境上で行うようにした．これは，(1)本学のプログラミング演習の環境上で動作することから，スキルを応用できること，(2)スレッドなどの講義の知識を応用できること，(3)センサ API が整備されていて，RCX で利用できるほとんどのデバイスを，API から操作することができること，(4)アプリケーションを別にロード可能で，転送時間が非常に短く，試行錯誤が容易，(5)OS 自体が軽量であり，アプリケーションの反応がよく，細かいレベルでの制御が可能である，の 5 点からである．

4．3　走行用コース

ロボット実験では，走行環境が結果に与える影響が大きいことから，コースなどの実走環境は重要な問題となる．あまりにもその変化が大きいと，設計やデバッグが難しくなり，実験テーマとして成立しにくくなってしまう．そこで，180cm×180cm の大きさの実験用のコースを作成した（図 2）．このコースは，周囲を黒く塗ってあり，照明や床の状態の影響が出にくいように工夫がしてある．これによって，環境の影響を避け，動作の再現性を高めるようにする．

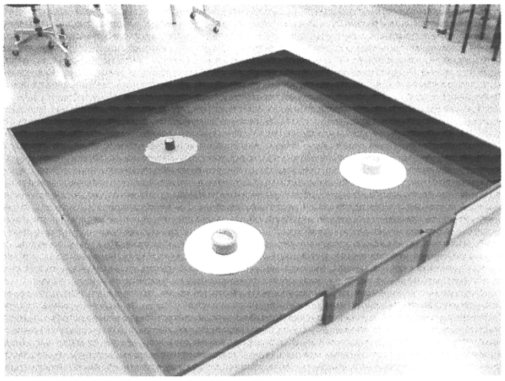

図 2　走行用コース

4．4　実験カリキュラム

実験カリキュラムを表 1 に示す．

表 1　実験カリキュラム

回数	内容
第 1 週	ガイダンス，部品整理
第 2 週	マニュアルにある Pathfinder ロボットの作成と基本的な動作の取得
第 3 週～第 4 週	スレッドプログラミングの基本と各種デバイスの操作
第 5 週～第 6 週	ライントレースの実現
	最終課題の提示とそれに向けたロボット設計（中間レポート提出）
第 7 週～第 9 週	最終課題に向けたロボット，プログラム開発
第 10 週	プレゼンテーション準備
第 11 週	発表会，走行会
第 12 週	報告書作成，提出

自走ロボット実験は，毎週金曜日の午後 3 コマ（90 分×3 限分）に開講していて，トータルで 12 週である．この実験での学習目標としては，制御システムのもっとも簡単な形を学習することである．基礎的な制御の知識の習得と，ロボットデバイスの構築，そして，その装置に対する制御の手法を学ぶことを目標としている．

カリキュラムの構成としては，環境に慣れるための基本的な操作への習熟，テーマ発表，設計，発表，報告書提出という手順を踏む．最初の段階では，ロボットに触れたことのない学生も多いことから，まず単純なロボットを組み立てさせて，LEGO の構造を把握する．また，サンプルプログラムを与えて，それを改良させることで，実行環境の操作についても慣れさせる．

legOS の API などに慣れた段階で，最終課題のルールを与える．それに対して，どのような設計，方向で望むかを簡単にレポートとして提出させて，学生に自分のアイデアをまとめさせ，方向を意識させる．

後半では，ロボットの実装とデバッグ，そして

発表会のためのプレゼンテーションの準備を行う．プレゼンテーションは，全員の前で一人当たり５分程度行う．そこでは，ハードウェアおよびソフトウェアの両方について自分がどのような工夫をしたのかという点を中心に発表してもらう．発表会後に，実際に走行会を行い，その成果を確認する（ちなみに，上位入賞者には個人的に賞品を用意した）．

その後，発表会での結果を踏まえて，工夫した点および，それがどのような結果につながったかの考察などを報告書としてまとめて提出してもらう．

４．５　運用体制

本実験に取り組む学生は，用意した機材の都合で最大で 10 人程度である．また，これと平行して従来の自走ロボット実験も併設して行っていた．これら二つを合わせて最大で 20 人程度の学生が，半期のテーマとして取り組んでいた．担当教員は３名が，両方を日にちに応じて分担するという柔軟な形をとっている．また，TA については，それぞれのテーマごとに各１名を割り当てていた．TA については，開発環境の設定，説明から，プログラムのデバッグの面倒まで，多くの仕事を行ってもらった．

４．６　最終課題の方針

著者らが最終課題を決める上で設定した方針は，(1)学生が構造やプログラムの工夫をする余地があること，(2)結果が量として出ること，の 2 点である．

CZR720-JRP を用いた実験では，レベル別にコースの走行を分け，その到達度によって，実験のレベルを分けていた．このような方法は，学生にとっては自分の意欲との戦いとなる．このような自分の到達度による評価手法は，彼らの意欲の刺激にならないと考えた．そこで，本実験では他人との競争を取り入れ，結果が量的に明らかになるテーマ設定を基本方針とした．

ここでは，2003 年度以降に行った課題を紹介する（2002 年度の課題については文献[9]を参照）．

４．７　2003 年度の課題

結果が量的に分かるようにするために，次のようなルールを設けた．

- 図 2 のコースを使用し，円形の的にロボットを使ってボールを入れる．
- 球は金属球と発砲スチロールの球の 2 種類を用意する．どちらであっても一つ１点とする．
- 一回 90 秒で何点取れるかを競う．
- 勝負は 3 回行い，3 回の合計点を得点とする．
- 勝負中ロボットが停止したら，ギブアップ宣言をすることで終了できる．
- スタート地点は三つ作る．
- 白色の円の的を二つ，青色の円の的を一つ置く．白色の円の的は 1 個 1 点，青色の円の的は 1 個 4 点とする．的の回りには，いずれも 10cm 幅の認識用の円がある．白色の的は直径 10cm であり，青色の的は直径 3cm である．
- ロボットにいくつ球を乗せるかは設計者の自由とする．
- 障害物を 2 箇所に置く．どこにおくかは実験当日に決める．

前年度まではロボット同士の対戦型の勝負を行わせていたが，対戦型では偶然の要素が大きすぎて，実験テーマとしては不適切であった．上記テーマ設定にすることで対戦時の相手の動作や，初期の位置に依存した勝負結果を回避可能になった．また，障害物の場所については不確定性を残し，学生がどのような場合でも動作できるようになる考察の必要性を促すようにした（障害物については，不確定性が高すぎることから，次年度以降採用しなかった）．

また，コースの中間でプレマッチを行うようにし，その結果を中間報告として提出させて，自分のシステムの問題点を把握させるようにした．さらに，発表後に人気投票を行い，どれが一番多く得点を得るかを予想させた．これは，実際の作成者の目から見た評価を知りたかったからである．

４．８　実験結果

学生が開発したロボットの例を図 3 に示す．

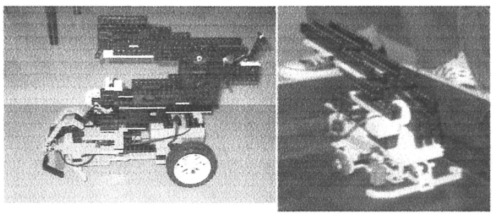

図３　学生が作成したロボット

このルールでは，どのように球をロボット内に保持するか，また地面の色をどのように判定して動作を変えるか，さらに，壁や障害物に当たったときに，どのように動作を変えるかという点がポイントとなる．多くのロボットでは，傾斜をつけたレールや筒のような機構により，ロボット内部にボールを保持していた．的に来ると，球をとめていたストッパをモータで回転させることによって，球を的に落とす動作をしていた．

優勝した学生のロボットでは，傾斜した筒によってボールをできる限り確保している．その一方

で，重力に任せて的に入れると，入らなかった場合でも全部球が出てしまう．そこで，ピストン機構によって，球を押し出すようにして，一つずつ入れるようにしている．これによって，的に球を入れすぎることなく，複数の的に入れていた．

また，的にまっすぐにロボットが入るように，動作調整を行っていた．センサをロボットの両側につけておき，片方が反応したら，両方のセンサが反応するまで旋廻し，その後２回後退することで，よりまっすぐに入るようにしている．

また，面白い工夫としては，(1)球を乗せる部分にキャタピラを使い，重力によって全部球が落ちてしまうことを避けて，何度でもチャレンジできるようにしたもの，(2)球をたくさん保持するために，階段状にする．(3)途中で空回りして落ちない場合があることから，両方をキャタピラではさんで制御することで，安定して落とせるようにしたもの，(4)金属球と発泡スチロール球とを組み合わせて乗せることで，球が詰まらないようにしたもの，などがあった．

５．グループワークによるロボット実験

2007年度より実験のカリキュラムが変わり，6週間でロボット実験を行うことになった．時間の短縮により，プレテストを行うことが難しくなった．また，学生のプログラミング能力の低下や，学科の標準的な教育言語がJavaに変更されたことから，従来のlegOSを使った実験が難しくなっていた．

さらに，著者らは本実験にコミュニケーション能力の強化や，仕様書や報告書の作成，レビューやテストなどの，ソフトウェア工学的な要素の実践を取り入れたいと考え，これらを盛り込んだ実験計画に変更した．本章ではこれについて述べる．

５．１　ロボット実験のカリキュラム

基本的な方向は，４章で述べた実験とほぼ同じである．異なる点は次のとおりである．

- ３人１チームのグループ構成とした．グループは機械的に割り振った．
- 言語環境は，2007年度がNQC+BrickCC，2008年度がleJOS+Eclipseとした．2007年度は実験の準備の関係でNQCとなった．2008年度はJavaのサブセットが動作するleJOSを用いた．

実験のカリキュラムは次のとおりである．

1. ガイダンス，チーム分け，簡単な制御プログラミング
2. 実験提案書提出，ロボット開発
3. レビュー，ロボット開発
4. テスト，ロボット開発
5. プレゼンテーション準備，ロボット開発
6. 発表会，走行会

５．２　ソフトウェア工学的な側面

専門学科でも，ソフトウェア工学を実践する機会はほとんどない．授業としてはあっても，それを実際の開発で用いることはほとんどない．演習などがほとんど個人での提出になっていることや，まとまったテーマで長い期間である程度の規模のソフトウェアを作成することが卒論までないからである．

この実験では，ソフトウェア工学の側面のうち，仕様書，報告書，レビュー，テストといった側面を採り入れた．直接意識させるより，実験の中の必須要素として行わせるようにした．文献[10]に書かれている次のフォームを改訂して使用した．

- チームシート
- 作業報告書
- 障害対処票
- レビュー報告書
- テスト報告書
- 相互レビューシート

チームシートは，名前付けや目標を明確にさせるために用いた．また，各回の終わりに作業報告書を提出させ，スタッフが毎回チェック，添削を行い，返却した．記述が不足している点については，再提出も行わせている．相互レビューシートは，実験がすべて終わった段階で，各自の実験への貢献度，長所を記述させた．記述後お互いに見せ合うことで，チームに対する貢献度，他人から見た自分の評価を確認させている．

チーム内でリーダ，ハードウェア担当，ソフトウェア担当，文書管理担当というように，責任者を決めて（3人なので，担当以外のことをやらないというわけではない），担当についての工程管理を行わせた．文書類については，紙ファイルに綴じさせて，最後に提出させ，採点の対象とした．

５．３　実験結果

これらの実験で作成されたロボットの例について図４に示す．

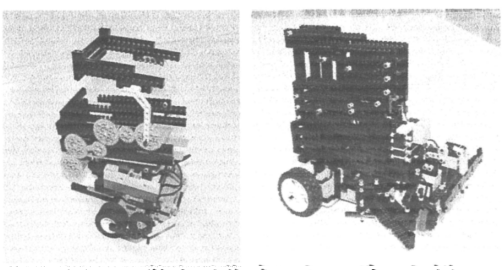

図４　学生が作成したロボット例

レポートやノウハウが継承されているわけではないが，基本的に構造上の違いは少なかった．

また，開発環境や言語の違いから，混乱が生じるのではないかという懸念があったが，大きな問

題は生じなかった．NQC が単純な言語であったことや，Java に習熟している学生が増えたことが大きな理由であろうと思われる．

5．4　実験アンケートによる評価

2008 年度の学生を対象にして，アンケート評価を行った．アンケートの項目は次の九つである．評価は 4 段階で行った．

1. 実験全体は面白かったか？
2. Java プログラミングの理解に有効だったか？
3. ロボットの制御方法の理解に有効だったか？
4. グループ作業の方法の理解に有効だったか？
5. 文書作成の方法の理解に有効だったか？
6. テストやレビューなどの理解に有効だったか？
7. プレゼンテーションの理解に有効だったか？
8. Mindstorms は教材として有効だったか？
9. 言語としての Java の利用は有効だったか？

集計結果を図 5 に示す．系列 4 がもっとも評価が高く，系列 1 がもっとも低い．

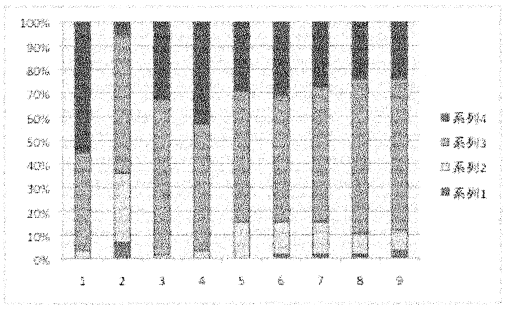

図 5　アンケート結果

図 5 から分かるとおり，総じて学生の評価は高い．面白さや制御方法の学習，グループの作業については特に高いスコアである．その一方で Java の理解，文書作成，プレゼンテーションでは不満も見受けられる．Java については，この学年では初めて触れた学生が多かったこと，文書作成については，添削を行ったが，それでも重要性を理解するところまでいかなかったと思われる．プレゼンテーションについては，発表が一度であり，修正ができない点が理解につながらなかったものと思われる．

6．考察

本章では，実験に対する考察について述べる．

(1)ロボットハードウェア開発

彼らが，もっとも熱中して行ったのが，ハードウェアの作成である．多くの学生が複数の構想を持って，取り組んでいる．

Mindstorms では，ロボットの強度を確保しつつ作成するのが難しい．特に，ファームウェアの暴走やリセット，電池交換などでは，RCX の本体を取り出して，ケースを開けなければならない．この部分の保守性まで考えて作成したものは多くなく，テスト中のちょっとした衝突により，ばらばらになっている例をよく見かけた．また，センサ類の感度のばらつきがかなりあり，影響を受けにくい取り付け方を工夫する必要があった．

また，部品数が足りないという不満は常にあった．

(2)ソフトウェア開発

ほとんどの学生にとってスレッドプログラミングやイベントドリブンプログラミングは初めての経験であり，優先度の設定や，デバッグなどが難しいと感じている例があった．このような問題は，分かりにくく，聞かれた側でも原因追求を一緒になって行わなければならない．教員および経験のある TA(システムプログラミングを研究している大学院生を割り当てた)を採用して，解決にあたった．

また，PC 上のプログラムのように，コンパイル→実行ではなく，ロボットへの転送がある点，さらに，通常利用可能なライブラリ群が使えない点，そして make を用いたコンパイルなど，それまで経験しない手法を使っていることもあったが，学生はさほど苦労せずにこのような開発環境に順応していた．2007 年度以降は統合環境を用いたことにより，これらの問題はほぼ解消された．

また，従来のプログラミングに慣れている学生からは，printf などを用いた内部状態の表示ができないことや，センサ入力のばらつきなどによって，意図通りの動作をしないことに対する苦労があった．

(3)プログラムサイズ

平均的には約 140 行，最大でも 300 行程度と作成したプログラムサイズは，小さなものにとどまっている．また，優勝者とプログラムサイズとの間に関係はなく，プログラム開発能力のある学生のロボットが，「強い」ロボットであるとは一概に言えない．この部分は，多くの学生がロボットハードウェアの改良に熱中していて，ソフトウェア作成の時間がほとんど足りなくなってしまったのが理由であろう．

また，本来であればハードウェアの変更に伴い，ソフトウェアも変更しなければならないが，これに気づくのが最終テスト段階に近くなってから，ということが多く，ほとんどが時間切れのまま望むことになっていた．

本来は，モジュール化を含むソフトウェアの再設計を行うべきだが，6ー12 回程度では難しい．動作させることが第一になってしまい(走行会があることから余計にそうなる)，関数化，モジュール化を含むきれいな設計，実装はあまり行われていない．最終的なレポート提出時に，コードの

見直しやコメント付けを再度行わせる指示を出すことしかできなかった.

(4)実験全体の感想

まず,学生が非常に主体的に興味を持って取り組んでいることが,強く印象付けられた.それまで成績がよくなく,特にプログラミングが苦手な学生であっても,集中してロボットを組み立て,プログラムを書いている.多くの学生は,実験時間外でも取り組んでいて,講義のない空き時間や土曜日なども実験室を開放していた.本学は夜の10時が門限になっているが,多くの学生は時間いっぱいまで取り組んでいる.

プログラミングという抽象的な作業については,苦手意識を持っている学生であっても,ロボットという具体的な装置を伴って教えることで,それまで持っていた苦手意識をうまく克服しているように感じる.かなり工夫したメカニズムや戦略を考案している場合があり,教える側が驚かされることもあった.

また,プレマッチの効果はかなりあった.これによって,自分たちのシステムやルールの問題点をかなり把握することができ,より改良したロボットを作成することが可能になった.アイデアのコピーを心配したが,学生は自分たちのアイデアにプライドを持っていて,安直に取り入れることはなかった.

7. 問題点

前記のとおり,本実験は学生からは,おおむね高評価を得ていたが,その一方で,実験を行う上で多くの問題点が発生していた.この章では,それらについて述べていく.

(1)デバッグ環境の不十分さ

RCX はその応用対象から,複雑なプログラミングを行うように作られていない.また,通信手段が赤外線の低速な環境しかなく,ディスプレイも5文字程度しか表示できない.そのために,リモートデバッグや内部情報の表示が難しい.このような環境下では,学習者はロボットの動作から,プログラムの動作を演繹しなければならなくなる.これがスレッドを用いたプログラム動作の分かりにくさ,システム自体のバグと融合し,バグの原因の追及をさらに困難にした.この問題は最後まで学習者について回った.

(2)教える側の負担の増大

このような学生の自主性を養う実験では,TAを含めた教える側にも,かなりの負担がかかる.すでに述べたように,ロボットプログラミングでは,通常のプログラムと比較してもバグの特定が難しい.TA 自体が必ずしもロボットプログラミングの経験者でないことから,スタッフ自身のスキルも大きな問題となった.特に,ロボットハードウェア自体の問題,ファームウェアの問題,学習者のプログラムの問題など複合的な要因の場合に,それを切り分けるのが大変であること,コースの反射や照明の違いといった物理的な要因によって動作が影響を受けることなどから,学習者がバグを再現することが難しかった.これについては,自由コメント欄でも問題として挙げている学生が多かった.

特に,本実験ではハードウェア自体にも自由度がある.学生が作成したハードウェアは,保守性や分かりやすさよりも,限界まで積み重ねられた状態で作成されていて,スタッフや TA(時には,作成者本人でさえも)にとって,どこに問題があるのかを切り分けにくいことが多々発生した.

(3)ロボット組立てへの過度の熱中

すでに述べたとおり,これまでの実験の観察では,学生の多くはロボットハードウェアの改良,アイデア出しに多くの時間を費やす傾向にあった.その結果,ソフトウェアのほうは必要最小限しか用意しないということが起きた.

これに対して,外界の状況を学習させ,それを動作に反映させるようなテーマを提示することも考えたが,このテーマを選ぶ学生は複雑なプログラミングを苦手にしている場合が多いことから,あまり複雑な課題を提示することはできず,またそこまで到達しなかった.

著者らは従来のロボットと比較して,Mindstorms を利用することでハードウェアの自由度を上げ,学生の自主的な取組みをすることを期待していたが,ここでは逆に,ハードウェアの自由度が上がったことが問題となったといえる.

これを解決するには,利用できる部品数に制限を与えるといったことが考えられるが,それは逆に(プログラミング能力は低いが)意欲のある学生をそいでしまうことにもつながる.制限をどのように設定すればよいかという点は,このような自由度のあるテーマでは,難しい問題である.

(4)ハードウェア能力の不足

学生の中には,かなり大掛かりな機構を構想してくる者がいる.しかし,RCX では入出力ポートがそれぞれ3つずつしかないことから,その部分で構想の実現に制約がかかる.意欲のある学生からはもう少し入出力ポートを使いたいという要求が多かった.また,中には RCX を2台カスケードして通信しながら動かすという応用を考えた学生もいたが,API レベルで簡易な通信手段がなく,実装が難しいことから断念せざるを得なかった.

(5)転送の干渉,失敗

RCX はその設計から,多くの RCX を近い場所で使うことを想定していない.そのために,実験

室内で場所が近いと，赤外線通信時にRCX同士が干渉することが多く，転送に失敗することがある（NXTではUSBで転送が可能であるが，BlueToothを用いた場合，台数が多いと同様の問題が起きる可能性がある）．

また，使用したファームウェアやドライバが安定せず，異常な値を返してくることから，センサの入力が信頼できない状態が発生するという問題があった．

(6)部品群の管理

全体としては細かい部品が多く，部品群の管理が大変であった．本学科では，ケースを部品収納に使っていたが，全部で１２個もあり，その管理には非常に手間がかかった．その一方で，感想にもあるとおり，学生は部品を常に要求している状態であり，より効率的な管理が必要とされる．

(7)プログラミングモデルの理解

2008年度のJavaにおいては，ボタンやセンサの制御ではListenerの理解が不可欠である．ところが，この手の非同期的なプログラムは学生にとっては理解しにくい．ほとんどのプログラムをListenerの中に記述してしまうチームがあった．そのために，応答性が悪くなり，ランタイムがエラーメッセージを出力して止まってしまうこともあった．デバッグシステムを持たないことから，このようになった場合に，学生自体で対処することが難しい．

(8)チーム作業への不満

2007年度からのチーム分けで，責任者を決めたが，責任者「だけ」がその作業に関われるというチームが続出してしまい，責任者という役名に縛られる傾向があった．また，チーム構築については，ランダムに割り振ったが，チームによっては，通常の友人関係を持込み，コミュニケーションがとれず崩壊しかかっていることもあった．

(9)老朽化

RCX自体は7年間利用している．7年目ともなると，ハードウェアの故障が目立ってきて，転送できない，実行できないなどのエラーが多発した．自由コメント欄でも，故障やエラーに対するクレームがかなりあった．RCX自体は元々の環境ではなく，legOSやleJOSといった特殊な環境下で動作させている．ファームウェア自体も入れ替えていることから，故障が発生した場合に，原因の特定が困難であった．

8．おわりに

本稿では，拓殖大学工学部情報工学科において行われてきた，ロボットを使った組込みシステム教育の実践例について述べてきた．

本実験が，学生にとって，情報工学の学習の中でどのような効果を挙げたかは，量的な評価はまだ行っていない．しかし，学生のレポートを読む限り，満足度は高い．特に，プログラミングそのものに苦手意識を持っていた学生が，ロボットとその制御という分かりやすいゴールに対して，かなり自主性を持って取り組んだ点は評価できる．通常の講義に対しては受身の姿勢で臨むことが多いのに対して，彼らが自分から積極的に取り組んでいたという点で，このような実験の意味は十分あったものと思われる．

その一方で，第7章にあるような多くの問題点がある．特に，ロボットハードウェアへの過度の熱中は，システム教育という点では問題があり，テーマの設定および，実験の進行に対してさらなる工夫が必要になるものと思われる．また，課題の盛り込み過ぎによる，学生の飽和も懸念される．全体に，実験の主題を再構築すべき時期に来ているものと思われる．

今後は，初年次へのロボットを用いたプログラミングの導入教育への適用，LEGO Mindstorms NXT[11]によるより安定した環境の導入，さらに，使いやすいデバッグ環境[12]の導入などが考えられる．

最後に，本実験の運用に当たり多くのご協力をしていただいた本学科の教員ならびTA諸君に感謝する．

参考文献

[1] 拓殖大学工学部情報工学科，情報工学実験IIテキスト，1999
[2] SYSTEMLOAD,CZR720-JRP, http://www.systemload.co.jp/
[3] 高橋，早川，拓殖大学工学部情報工学科における自走ロボットによる実験，私立大学情報教育協会，2000, http://www.juce.jp/LINK/journal/0004/04_04.html
[4] LEGO, Robotics Invention System, 1999
[5] Dave Baum, NQC Programmer's Guide, 1999
[6] Markus L. Noga, legOS, 1999
[7] brickOS, http://birckos.sourceforge.net/
[8] leJOS, http://lejos.sourceforge.net/
[9] 早川，高橋，拓殖大学工学部情報工学科における LEGO MindStorms を用いた情報処理教育の実践例，人工知能学会誌，pp.522-531, Vol.21 No.5 2006
[10] 鶴保，駒谷，ずっと受けたかったソフトウェアエンジニアリングの授業，翔泳社，2006
[11] MindstormsNXT, http://mindstorms.lego.com/, 2006
[12]田中，大角，川上，西野，早川：ロボットを用いた組込みシステム学習支援環境の構築，情報処理学会組込みシステム研究グループ合同研究会，pp.79-86，2006