情報処理学会論文誌 Vol.60 No.2 633–641 (Feb. 2019)

組込みソフトウェア技術者育成のための開発工程を重視したプログラミング教育

舘 伸幸[1,a)] 山崎 進[2] 香山 瑞恵[3]

受付日 2018年5月24日, 採録日 2018年11月7日

概要：プログラミング言語は学んでも，実際の課題をどのように設計すればよいのか困っている学生は少なくない．いわゆる，実践力が不足しているケースである．この問題に対し，要件分析からテストまでの開発工程を順に体験することで，組込みソフトウェア開発のしかたを学ぶ教材を作成した．この教材は，特に信頼性や安全性が求められる組込みシステム用のソフトウェア開発技術者育成を主たる目的とし，各開発工程の責務の理解と遵守についても扱うものである．本稿では，作成した教材の内容と試行結果を報告し，改善への考察を述べる．

キーワード：組込みソフトウェア，プログラミング教育，実践力，教材開発，開発工程

Programming Education Emphasizing the Development Process for Training Embedded Software Engineers

Nobuyuki Tachi[1,a)] Susumu Yamazaki[2] Mizue Kayama[3]

Received: May 24, 2018, Accepted: November 7, 2018

Abstract: Many students do not understand how to design actual problems even though they learn programming languages. It is a case where so-called practical skills are lacking. In response to this problem, we created teaching materials to learn how to develop embedded software by experiencing the development process from requirement analysis to testing in order. The main purpose of this educational material is to train software development engineers for embedded systems which require reliability and safety, as well as dealing with the understanding and compliance of responsibilities of each development process. In this paper, we report on the content of the teaching materials we created and the trial result, and discuss the consideration for improvement.

Keywords: embedded system, programing education, practice ability, teaching material development, development process

1. はじめに

組込みシステムのソフトウェア（以下，組込みソフトウェア）は，わが国の産業構造にとって重要な技術領域である．その開発範囲は，アプリケーションからミドルウェアやデバイスドライバまで広範囲である．また場合によっては，電気回路や機械機構に関連する知識も要求される．渡辺らは，大学学部における組込みソフトウェア教育に関し，同様の性質をあげてその困難さを述べている [1]．

さらに組込みソフトウェアは，自動車や医療機器のように信頼性や安全性を強く求められることが多い．このため，単に最終成果物が動くだけでなく，その開発プロセスの品質も強く要求される．

本研究では，このような技術領域に立ち向かえる技術者を育成するために，組込みソフトウェアをどのように開発すればよいかという，基本の型の 1 つを学ぶための大学学

[1] 名古屋大学
Nagoya University, Nagoya, Aichi 464–8601, Japan
[2] 北九州市立大学
The University of Kitakyushu, Kitakyushu, Fukuoka 808–0135, Japan
[3] 信州大学
Shinshu University, Nagano 380–0928, Japan
[a)] n-tachi@nagoya-u.jp

部向け教材の開発と試行を行うこととした．

なお，本研究は実践的情報教育協働ネットワーク enPiT [2] の一環として実施した．組込み分野を担当していることから，教材として組込み開発をテーマにしている．

2. 教育のポイントと関連研究

2.1 教育のポイント

本研究で試作した教材は，開発プロセスに従って「きちんと作る」ことを学ぶことを最重要視している．そのためのしかけとして，一般的な授業の演習テーマと異なり，要求仕様をあえてあいまいな内容としている．これにより，要求仕様を斜め読みしてとりあえずコードを書くようなやり方が通用しないことを体験させると同時に，要求仕様を分析する，そして設計するといった，開発プロセスに従った作り方への導入を狙っている．

結果，学生らによる自由記述の学習記録には，仕様分析や設計の重要性についての言及が多く現れ，それらには演習実施日のプロセス内容が反映されており，一定の理解が得られたと考えられる．その他演習結果の詳細については 5 章で述べる．

2.2 関連研究

開発プロセスを学習に取り入れている報告には，次のようなものがある．たとえば松澤らは，ソフトウェア開発 PBL において，反復プロセスを導入する試みを実施しており，開発の進め方に関する学生のポジティブな反応が紹介されている [3]．沢田らも，ソフトウェア開発実習において開発プロセスを強く意識させるカリキュラムを用いた報告を行っている [4]．海外では，たとえば Lydia らによる，PBL にプロジェクトマネジメントを組み合わせて，開発演習にマネージャやステークホルダが登場する実践的な学習の取り組みが報告されている [5]．

これらに特徴的なことは，ほとんどが PBL として実施している点である．PBL は課題解決型のチーム学習であり，その学習効果については，前述の論文以外にも多くの肯定的報告がある．一方で PBL は，参加するための最低限のスキルが要求される学習法でもある．

無手勝流でも課題解決に至ることも可能なため，今回のような基本の型を学習するというような目的には使いにくい．また，チームでの学習であるため，最低限のスキルに満たない人はうまく学べない場合がある．これについてはたとえば河西らは実験結果から，「知識が乏しく自己学習習慣も身に付いていない学生の場合，能動的な学習や参加が求められる PBL のような授業にうまく対応できず，苦手意識や劣等感を抱きやすい」という仮説を述べている [6]．また松浦は，PBL の効果を述べながらも，前提知識の教育などの授業設計が重要である旨述べている [7]．

そこで本研究では，あえて PBL の形態とせず，講義＋演習という形で，前提知識として開発の基本の型を学ぶところを志向した．

3. 教材設計

3.1 方針

組込みソフトウェア技術者としての社会における実践力の要素として，次の 4 つをテーマとして教材を設計することとした．

(1) 製品技術 Product

ハードウェアの制御技術や，動作制約に関する技術を学ぶ．

(2) 開発工程を進める能力 Process

要件を起点として，プログラムの作り方を学ぶ．

(3) プロジェクトで活動する能力 Project

有期性のある作業として，中間および最終の目標を達成することを学ぶ．

(4) 専門職としての技術者の行動規範 Professionalism

不明点や疑問点について能動的に行動する，周囲と協働する，考え抜くといったことを学ぶ．

(1) は，組込みソフトウェア開発に求められる特徴的な技術である．文字どおり機器に組み込んで周辺のハードウェアと連携して機能を実現するプログラムであるため，ハードウェアを利用，制御する技術が必須であることから [8]，テーマの 1 つとしている．また，動作順序，動作優先度，並行動作といった，システム全体や個々の機能の動作制約について考慮し対応する技術も，このテーマに含まれる [9]．以下，プロダクトと称する．

(2) は，特に重視している項目である．2.2 節で述べた関連研究以外でも，ソフトウェア教育において開発プロセスを導入することが重要であることが報告されている [10], [11]．また，実践的な情報教育を受講した卒業生に対する調査では，70.4%の人が，「システム開発手法や開発プロセス」の知識・経験が企業での実務に役立っていると感じていると回答している [12]．以下，プロセスと称する．

(3) については，2012 年から実施した大学院生向け enPiT [13] での教育の取り組みにおいて，学生にスケジュール管理を学ばせた実績からその有用性を認識できたことにより，テーマに取り入れている [14]．以下，プロジェクトと称する．

(4) は，経済産業省の提唱する社会人基礎力 [15] を念頭に，テーマとした．以下，プロフェッショナリズムと称する．

3.2 教材内容の概説

教材内容を教材設計マニュアル [16] を参考に解説する．

3.2.1 課題

学習に使用する開発課題は，キッチンタイマのソフトウェア開発である．30 秒または 60 秒の時間を計測する機

能を持ち，操作は 2 つのタクトスイッチを用い，動作状態は 3 つの LED で表示する．制御プログラムは C 言語で記述を行い，そのプログラム規模は，150〜200 ライン程度を想定している．仕様の詳細については 4.2 節で述べる．

3.2.2 学習体制

教員 1 名に学生数十名の演習授業を想定している．また，今回の試行では，学生 10 名あたり 1 名程度の TA を配置した．開発環境は 1 名に 1 台で，個別学習である．周囲の学生どうし相談することや，TA または教師に質問することは，積極的に推奨としている．また，インターネットなどによる調査も許可している．これらは，プロフェッショナリズムの行動目標（後述）を涵養することを目的としている．

3.2.3 前提条件

学習対象者は，情報技術系の大学学部生 3 年生としている．しかし，次に述べる前提条件を満たしていれば，特に制限を設ける必要はない．授業受講の前提条件は次のとおりとした．

- C 言語によるプログラミングについて学習済みであり，文法を知っていて，100 行程度のプログラムを自力で書くことができること．
- 開発環境（Arduino [17]）の使い方，UML [18] によるモデリング，組込み分野に特化したプログラム開発の経験については，必要な条件としない．

3.2.4 行動目標

前章で述べた 4 つのテーマに次の行動目標を定めた．

- プロダクト

次の 5 つの製品技術を学習し獲得する．

  - Arduino 開発環境の使い方
  - LED の制御方法
  - スイッチの特性と使い方
  - タイマの使い方
  - 複数のタスクを同時実行させる方法

- プロセス

プログラムを作る基本的な手順としての開発工程（要件分析，基本設計，詳細設計，実装，テスト）に従って作業を進められる．

- プロジェクト

開発工程ごとに締切りを意識して作業を進められる．

- プロフェッショナリズム

次のような行動ができる．

  - 不明点を自力で調査できる．
  - 周囲の学生と相談できる．
  - 作業報告ができる．

また，開発工程を学習する前段階として開発環境の準備および学習を学習項目としている．この内容に対する行動目標は，自主学習テキストにある 6 つの演習課題を解決できることとした．演習課題の詳細は 4.1 節で述べる．

表 1 機能チェック項目
Table 1 Function check items.

No.	チェック内容
1	LED0 が 1 秒周期で点滅する
2	LED3 が動作中パターンで点滅する
3	LED3 が 0.25 秒で 15 秒間点滅する
4	SW0 でタイマがスタートする
5	SW1 で LED1 の点灯と延長の設定を制御できる
6	LED0 と LED3 が同時点滅する
7	カウントアップで LED3 が終了の点滅をする
8	停止中以外に SW1 を受け付けない
9	カウント中に SW0 で停止できる
10	終了点滅中に SW0 で停止できる
11	停止状態に戻ったら時間延長をクリアする
12	時間を正確に測れる（誤差 1 秒以内）
13	スイッチのチャタリング対策ができている

3.2.5 評価条件

各開発工程への評価条件は，3.2.4 項の行動目標への意識を持つことと，課題への前向きな取り組み姿勢，および中間成果物や最終成果物の提出とした．なお，演習授業中には，周囲の学生・TA・教員との相談を許可した．

3.2.6 合格基準

各種図表などの中間成果物と，最終プログラムの機能チェック項目（**表 1**）に対する網羅度を総合的に判断する．

3.3 指導方略

本教材は，3.1 節に述べたように，プロセスに則って開発を進めることを重要視している．そこで，授業全体を大きく「開発環境の学習」「要件分析」「基本設計」「詳細設計・実装・単体テスト」「システムテスト」の 5 つの開発工程に区切って，順を追って進める形とした．以下，工程ごとの指導方略を述べる．

3.3.1 開発環境の学習

社会での開発では，開発環境の構築と学習は重要な作業工程である．また，学習対象である情報技術系の大学学部 3 年生は，その多くがスイッチや LED といったハードウェア部品の振舞いや使い方について経験が少ない．そこで本教材では，開発環境や使用するハードウェアの学習に関する専用テキストを用意し，自己学習させることとした．詳細については 4.1 節で述べる．

3.3.2 要件分析

要件分析は開発工程の最上流であり，ここでの不具合は製品の品質に大きな影響を及ぼす．たとえば飯泉らの調査では，組込みソフトウェア開発で作りこまれた不具合のおよそ 40%が要求仕様にかかわるものとされている [19]．したがって，実践力という観点において重要な学習課題であるが，一般に大学の演習授業における課題では，分かりやすく完成されており，学習項目として扱われないことが多い．

本教材では，あいまいな記述や矛盾を含んだ要求仕様を意図的に使用することで，要件分析の重要性を学ばせるこ

とを目的としている．詳細については 4.2 節で述べる．

学習の進め方としては，最初に，学生に製品のイメージ図を書かせる．これにより，要件に含まれるランプやスイッチなどのアイテムとそれらの役割の理解を促す．

次に各人で要件を再度詳細に読むよう指示する．この際，ラインマーカなどで記述を塗りつぶしながら読むことを求める．そして，疑問に感じたことなどを記入させる．

そのうえで学生には，隣同士のペアで不明点や疑問点について話し合わせる．さらに，複数ペアでの 4〜6 名で話し合わせる．これにより学生は，自分が気づかなかったことに他人が気づいて，より多くの不明点や疑問点を抽出できるという，レビューの効果も実体験することが期待できる．

最後に，グループごとに順に疑問点を発表させ，それに対して教師が顧客役となって回答を出す．学生はそれをメモして手元の要件を完成させる．

3.3.3 基本設計

基本設計では，組込みソフトウェアの 3 つの技術要素に関し，「機能」に対する「振舞い」と「構造」について学ばせることを目的としている．基本設計での記法には，UML [18] を使用した．

学習の進め方として，最初に，システムの振舞いを理解させるために，状態遷移図を書かせる．ほとんどの学生が書き方を知らないので，信号機など身近な例を使って状態について演習し，次に課題の仕様について順を追って穴埋め式に状態遷移図を記述させる．状態遷移図を書いていく過程で，要件の新たな不明点が発見できるようにテキストを設計してあり，学生は要件分析としても有効であることを学ぶ．状態遷移図が書けたら，それをもとに状態遷移表を作成し，状態遷移図の記述漏れを防ぐことも学ぶ．

次に，ユースケース図やコンテキスト図を説明し，静的モデルを導き出すまでを講義する．この段階で学生は，どのようなソフトウェア部品（以下，部品）を作れば，目的のソフトウェアを作成できるか，すなわちプログラムの構造を理解することができる．

3.3.4 詳細設計・実装・単体テスト

詳細設計では，組込みソフトウェアの特徴であるリアルタイム性と同時実行を学ばせることを主たる目的としている．これには，シーケンス図とフロー図を組み合わせた，シーケンス・フロー図という表記法を考案して使用した．詳細については 4.4 節で述べる．

学習の進め方としては，最初に，時間制御が必要でかつ最も単純な動作を行う LED0（表 1 機能チェック項目 No.1）について，シーケンス・フロー図から実際の処理を行う部品の作り方を説明し，実装のしかたを例示し，実行させる．このとき，作成した部品を呼び出すダミーの部品を作ることで，部品単体でのテストをする方法も示す．この練習を経たのち，他の部品について自力での設計を求める．

3.3.5 システムテスト

システムテストでは，最後に検証をするということを学ぶことを目的としている．本来であれば単体テストの次に結合テストがあるが，課題のプログラムは小規模のため，本教材では省略することとし，ここではシステムテストのみを扱う．また，授業時間上の制約から，異常系動作については省略し，正常系のみの簡易チェックとしている．

学習の進め方は，表 1 のチェック項目に従ってセルフチェックを行い，次に隣席など別の学生のチェックを受け，最後に教師による合否判定を受ける．

4. 教材の詳細

前章で述べた教材に関して，開発環境（ハードウェアおよびテキスト），要求仕様，基本設計における状態遷移図，詳細設計におけるシーケンス・フロー図の詳細を述べる．

4.1 開発環境

開発環境のハードウェアには，Arduino Leonardo と専用シールドの組合せを用意した（**図 1**）．Arduino [17] を選んだのは，Windows でも Mac でもほぼ同一の環境で開発可能であること，開発環境のインストールに手間がかからないこと，初心者でも習得が簡単であることが理由である．

Arduino の入出力コネクタに挿して使う回路基板をシールドという．今回の教材では専用のものを作成した．搭載部品は，LED4 個，タクトスイッチ 2 個，DIP スイッチ 4 個，7 セグ LED4 桁，スピーカ 1 個，WiFi モジュール 1 個，I2C コネクタである．本稿の教材テキストでは，このうち LED3 個とタクトスイッチ 2 個のみを使用している．

開発環境のテキストは全 54 ページで，教材の開発テーマに準じた 6 つの演習問題を含んでいる．これら演習問題は，本教材での開発課題で使用する製品技術に対応しており，事前学習や演習途中での参考書の役割を狙っている．各演習の内容は次のとおりである．

演習 1　サンプルコードをもとに LED の点滅を行う．
演習 2　入出力定義のヘッダーファイルを完成させる．
演習 3　2 つの LED を異なるタイミング（同期）で点灯させる．シリアルモニタでデバッグする．

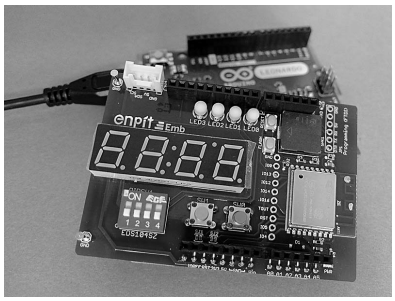

図 1　Arduino Leonardo + 専用シールド
Fig. 1　Arduino Leonardo + exclusive shield.

演習 4　スイッチ入力プログラムを作成する．
演習 5　チャタリングを考慮したスイッチ入力プログラムを作る（サンプルあり）．
演習 6　タイマ割込みによる LED 点滅プログラムを作る．

4.2　要求仕様

学生に示すあいまいさや矛盾を含む仕様は以下である．

- 概要
  - タイマを起動して，設定した時間が経過したことを知らせる
  - 設定時間は 30 秒刻みで設定できる
  - スイッチと LED の使い方
    - ・LED0　：電源が ON であることを表示
    - ・LED1　：SW1 が押されたことを表示
    - ・LED3　：タイマが動いていることを表示
    - ・SW0　：タイマを起動・停止
    - ・SW1　：設定時間を 30 秒延長
- 外部仕様
  - 電源が ON の間（プログラムが動いている間），LED0 の点灯・消灯を 1 秒間隔で繰り返す
  - SW0 を押すと，設定時間を 30 秒にして，タイマを起動する
  - タイマが動作中に SW0 を押すと，タイマを停止する
  - タイマが停止中に SW1 を押すと，LED1 を点灯させ，設定時間を 30 秒延長する
  - タイマが動いている間は，5 秒間隔で，LED3 を 2 回点滅させる．LED3 の 2 回点滅は，点灯・消灯を 0.25 秒間隔で 2 回繰り返すことで行う
  - 設定時間が経過すると，LED3 を 15 秒間点滅させた後，消灯する．LED3 の点滅は，点灯・消灯を 0.25 秒間隔で繰り返すことで行う

学生が気づくべきあいまいな記述は，後半の外部仕様に含まれており，それらを修正・追記したものは次のとおりである（太字が修正または追記．取り消し線は削除を表す）．

- 外部仕様
  - 電源が ON の間（プログラムが動いている間），LED0 の点灯・消灯を 1 秒**周期**で繰り返す．**（0.5 秒点灯，0.5 秒消灯）**
  - **電源を ON 後の起動状態を停止状態とする**
  - **タイマ計測時間の初期値は，30 秒とする**
  - SW0 を押すと，~~設定時間を 30 秒にして~~，タイマを起動する
  - タイマが動作中に SW0 を押すと，停止状態に戻る
  - タイマが停止中に SW1 を押すと，LED1 を点灯させ，設定時間を 30 秒延長する
  - **停止中以外に SW1 を押しても無視する**
  - **停止状態かつ LED1 点灯中に SW1 を押すと，LED1 を消灯し，設定時間を 30 秒に戻す**
  - 設定時間が経過すると**アラーム状態**となり，LED3 を 15 秒間点滅させた後，消灯する．LED3 の点滅は，点灯・消灯を 0.25 秒間隔**（0.25 秒点灯，0.25 秒消灯）**で繰り返すことで行う
  - **アラーム中に SW0 を押すと，停止状態に戻る**
  - **アラーム終了時には，停止状態に戻る**
  - **タイマ動作中やアラーム中から停止状態に戻るときは，LED3 の点滅は停止して LED3 を消灯する．また，時間延長設定は取り消し，LED1 を消灯する．また，タイマカウントは 0 に戻す**
  - タイマが動いている間は，**下図のように**，5 秒間隔で，LED3 を 2 回点滅させる．LED3 の 2 回点滅は，点灯・消灯を 0.25 秒間隔で 2 回繰り返すことで行う
  - **各時間の誤差は 1%以内とする**

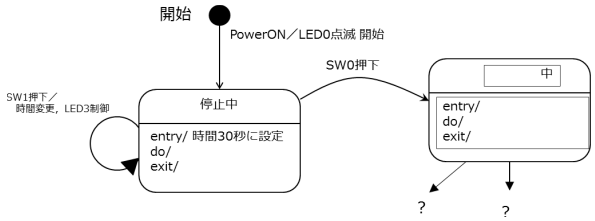

図 2　状態遷移図 1
Fig. 2　State diagram 1.

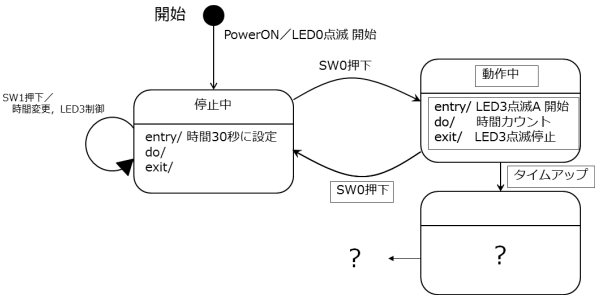

図 3　状態遷移図 2
Fig. 3　State diagram 2.

4.3　状態遷移図

UML での記述法を説明したあと，交差点の信号機を例に演習を行う．次に本教材の課題の状態遷移図を順次作成していく．最初に**図 2** の状態まで例示し，右の状態の四角の中や，「？」で示したイベントのとび先について考えさせる．

最初のあいまいな仕様では，キッチンタイマの動作終了後の動作について記述されていない．これは要件分析でも学生が見落とすことがある．しかし，状態遷移図を**図 3** まで書き進むと，学生は右下の状態からの移動先が不明であることに気づく．このことから，状態遷移図を書くことで仕様の不備を発見できることも学べる．

4.4　シーケンス・フロー図

複数の部品が時間の経過に従って連動して動作することを学ばせるのは難しい．UML にはシーケンス図という部品間での情報の授受を表記する方法がある．しかし，組込みソフトウェアで重要な情報授受のタイミングや，それにともなう処理内容まで表すことは得意ではない．

そこで，本研究ではシーケンス図と，処理フローを組み合わせた図を考案した．これをシーケンス・フロー図と呼称している．**図 4** に LED0 の制御例を示す．各軸は構成するソフトウェア部品の処理フローであり，下方向が時間経過を示す．横方向は要求または応答を示す．これにより，部品間のタイミングと部品での処理フローを同時に設計できる．

また，図 4 における初期化や時間通知といった部品間の要求は，それを受け取る部品には受け口が必要であり，すなわちそれがその部品に必要な公開関数を意味する．そこ

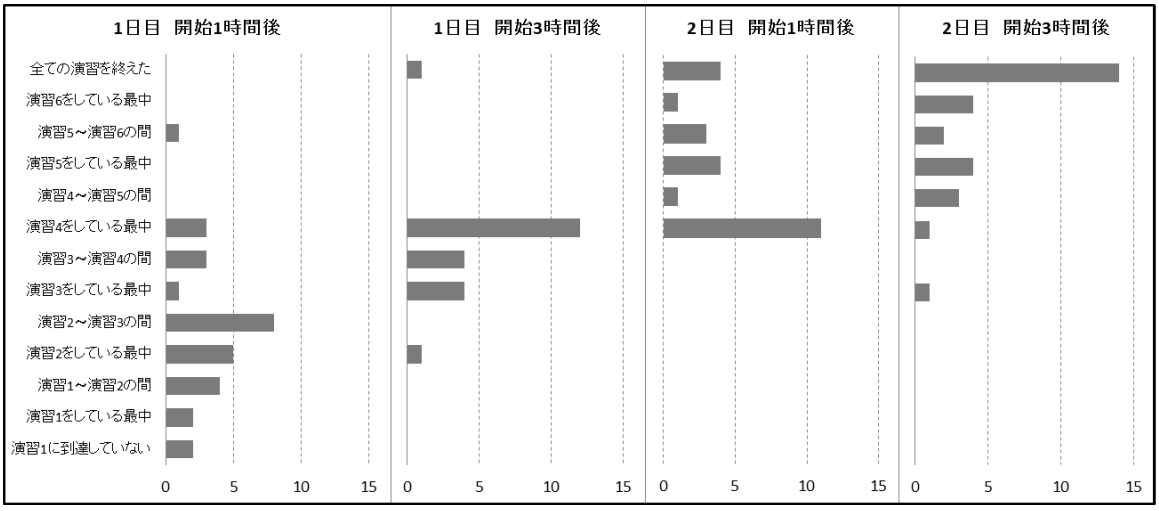

図 5 開発環境学習の進捗状況
Fig. 5 Progress of learning development environment.

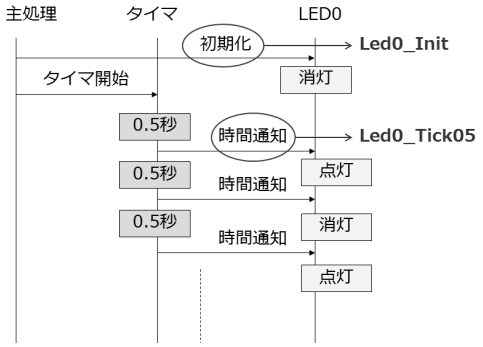

図 4 シーケンス・フロー図
Fig. 4 Sequence flow chart.

で，図 4 の Led0_Init や Led0_Tick05 のように，それらに名前を付けることにより，学生は必要な関数のフレームを理解することができる．

この後，設計の実装例を示して動作を確認させる．そして，LED3 についても同様に設計，実装，単体テストを自力で行うよう求める．

5. 試行結果と考察

本教材の試行は，学部 3 年生 61 名を前半組 30 名と後半組 31 名とに分け，2 ターン実施した．前半は 2017 年 4 月 25 日から 6 月 6 日までの 6 回，後半は 2017 年 6 月 20 日から 7 月 25 日の 6 回とし，各回 3 時限 × 90 分であった．学生には，各回 3 時限目終了時に学習記録としての質問紙調査を実施した．実験計画は，作成した教材を前半組で実施し（試行 1 回目），結果をもとに指導方略の修正をしたのち後半組で試行 2 回目を実施することとした．

5.1 開発環境学習

3.3.1 項で述べた開発環境の学習に，前半組では 1 日 3 時限（270 分）を割り当てた．その結果，6 項目ある演習問題のうち，最後の演習 6 まで終了したのは 30 名中 4 名で，大半の学生が演習 4 の途中か終了という状態だった．そこで，後半組では 90 分増やして 360 分を割り当て，進捗状況を計測した．結果を**図 5** に示す．横軸は人数で，縦軸は学習中の演習課題である．上に行くほど学習が進んでいることを示す．

表 2 最終結果
Table 2 Final result.

状 況	前半組	後半組
部品一部のみ完成	11	2
部品ほぼ完成	8	4
最終デバッグ中	6	8
完成	5	15

31 名中 14 名（45%）の学生が，2 日間 360 分で全演習を終えている．一方で，2 日目の開始 1 時間後時点で全員が演習 4（スイッチ入力）まで進んだが，最終的には半数の学生が完了できていない．実際の開発演習でもスイッチ入力でうまくできなかったという意見が多かったことから，スイッチ入力の演習の困難度に原因があったと推測できる．スイッチ（タクトスイッチ）は，押されてない・押された・読み取られたといった 3 つの状態を持ち，さらにはチャタリングという物理特性の問題が加わるため，単純な操作の部品であるわりに設計が難しいためと考えられる．

5.2 全演習結果

前半組の最終結果を**表 2** の「前半組」列に示す．プログラムを完成できたのは 30 名中 5 名（17%）であった．最終デバッグ中が 6 名（20%），ソフトウェア部品はほぼ完成し，組み立て直前であったのが 8 名（27%）であった．これら 14 名（47%）は，さらに作業時間があれば完成に至った可能性はある．一方で，必要な部品を作りきれなかったのが 11 名と，全体の 37%にのぼった．この 11 名中 9 名

表 3 学習内容アンケート結果
Table 3 Results of questionnaire on learning contents.

日	講義・演習内容	前半組 学んだこと		前半組 難しかったこと		後半組 学んだこと		後半組 難しかったこと	
1	組込みシステム	組込みシステム	7						
	開発プロセス	開発プロセス	6						
	要件分析	要件分析の大切さ，手法	24			要件の分析の大切さ	9		
		設計の大切さ	1						
				仕様書文章の読解	10			仕様書文章の読解	6
	基本設計			状態遷移図，表の作成	10	状態遷移図，表の作成	7	状態遷移図，表の作成	4
				状態の分け方	1				
	開発環境					開発環境の使い方	14	開発環境の演習	4
2	開発環境について自己学習					開発環境について自己学習			
3	開発環境（一部の学生）	開発環境の使い方	5			開発環境の使い方	6		
	基本設計	設計図，構造	24	用語（ミリ，チャタリング）	3	フローチャートの書き方	14	フローチャートの書き方	7
		実装方法	9	例図の読み方	1			状態遷移図との関係	1
		技術Tips	7	ハードウェアの内容	2				
	詳細設計			シーケンス図の書き方	11	シーケンス図の書き方	1	具体的処理の方法	2
								プログラミング方法	2
4	詳細設計	設計と実装の関係	20	設計と実装の関係	7				
		タイマの使い方	5			タイマの使い方	12	タイマの使い方	1
		スイッチの制御方法	8	スイッチの制御方法	3			スイッチの制御方法	2
		LEDの点滅制御	8			LEDの点滅制御	8		
				同時動作	3	同時動作	10	同時動作	2
				状態遷移図やシーケンス図	3				
	実装			プログラミングの仕方	9			プログラミングの仕方	5
5	詳細設計	設計と実装の関係	4	設計と実装の関係	2				
				同時動作	3	同時動作	3		
		スイッチの制御方法	8	スイッチの制御方法	8	スイッチの制御方法	3	スイッチの制御方法	4
				タイマの使い方	2				
				LEDの点滅制御	1				
	実装			プログラミングのしかた	4	プログラミングのしかた	5	プログラミングのしかた	2
6	詳細設計	スイッチの制御方法	3	スイッチの制御方法	8			スイッチの制御方法	2
		タイマの使い方	2	タイマの使い方	2			タイマの使い方	1
		設計と実装の関係	3	設計と実装の関係	3	設計と実装の関係	1		
				LEDの点滅制御	2	LEDの点滅制御	1		
						同時処理	2	同時処理	1
	実装			プログラミングのしかた	1				
	テスト	開発環境の使い方	3			開発環境の使い方	1		

は，LED3 という特殊なパターンでのランプ点滅処理の作成時点から進めることができていなかった．

後半組では，前半組で発覚した問題であるプログラミングそのものに不慣れであるという事実に対し，2 つの対策を立てて臨んだ．1 つは，開発環境の自己学習について，学習時間を 90 分増やして 360 分としたことである（前半組は 3 時限 270 分）．その結果，5.1 節で述べたとおり半数の学生が全課題をクリアできた．もう 1 つは，処理手順をフローチャートもしくはアクティビティ図という形で，よく考えさせてからコーディングさせるという方法である．

後半組の最終結果を表 2 の「後半組」列に示す．完成までいった学生は 31 名中 15 名（48%）であり，部品完成まで行った者と最終デバッグ中だった者を含めると 27 名（87%）が，ひととおり設計に従ったコードを記述できた．

5.3 基本設計〜テストの学習記録

授業の経過について，学生による学習記録を**表 3** に示す．これは各回の終わりに，その日学んだことと分かりにくかったことについての質問紙調査をまとめたものである．

考察中の件数におけるパーセント値は，それぞれの授業回におけるアンケート回収数を母数としたものである．特に後半組で回収率が悪かったが，無記名で実施したことなどが主原因と考えられる．考察では，実際の授業観察も加味して述べる．

5.3.1 前半組

1 日目は，組込みシステムとは何かと，開発プロセスの重要性について講義を行い，要件分析の講義と演習を実施した．これに対して，学んだこととして要件分析をあげた意見が回答者 29 名中 24 件（82%）あり，この内容が印象深かったことがうかがえる．一方で状態遷移図や状態遷移表が難しかったという意見が 10 件（33%）あった．次に困難点として多かったのが仕様書文章の読解で，9 件と 30%を占めた．簡易的とはいえ技術文書であり，しかもハードウェアに関連した内容を含むので，読み慣れていないという側面も否定はできない．一方で，文章力とプログラミング能力に関連があるという研究もあり [20]，見落とせない結果の可能性がある．

2 日目は開発環境に関する自己学習を実施している．3 日目は設計から実装までの講義と演習を実施した．学んだこととして設計図やソフトウェアの構造をあげた意見が 25 件中 24 件（96%）あり，学習目的はよく理解されていることが推察できる．

難しかったこととしてシーケンス・フロー図によるタイミング設計という意見が 11 件（44%）あった．これには，単に図の書き方に慣れないというものと，（時間と部品との関係を）イメージしにくいというものとがあった．

一般的な情報技術系の大学学部の演習で，タイミング設計を扱うことは少ない一方，関連業界における組込み系開発では必須技術であり，教育と社会の乖離の1つといえる．したがって，書き方も考え方も，さらに分かりやすく学習できるよう改良する必要があると考えられる．

4日目は設計内容をもとに，実際にコーディングしてソフトウェア部品を作り始めた．学んだ内容については，実際のソフトウェア部品であるLEDやスイッチ，タイマといった項目に分散したきらいはあるが，おおむね設計結果の利用と部品化というテーマは伝わっている．

一方で，設計結果を実際にどのようにコードに変換するかということでつまずく学生が多くいた．プログラミングの仕方そのものをあげたものが29件中9件（31%），設計と実装の関係をあげたものが7件（24%）あった．重複もあるものの，計16件（55%）の困難報告であり，半数の学生がここでつまずいたことがうかがえる．

内容を見てみると．「（サンプルの）スイッチ処理関数が何をしているのか分からなかった」「部品に必要なことをどこまでプログラムにすればよいか分からなかった」「ループ文の中でどのようにすればLEDの3つの動きを同時に操作できるか分からなかった」など，プログラミングに不慣れなことが要因と思われる意見が多かった．これはそのまま5日目6日目にも影響を及ぼし，先にあげたように完成に至った学生が参加者30名中5名（17%）という結果に帰結している．

5.3.2 後半組

組込みシステムの概要についての講義を一部削減したものの，要件分析にかける時間は変更していない．しかし，1日目最終時限から開発環境に関する演習をしたことから，印象の中心はそちらになってしまっている．

3日目にフローチャート作成指導を実施した．今回は，**図6**のように，基本設計で作成した状態遷移図に対応させる形で，各状態の処理フローを考えて結合させるという方略をとった．その結果，学習記録には学んだこととしてフローチャートがあがった一方で，難しかった点にもフローチャートの書き方に関するものが14件中10件（71%）あがった．実際に，書き始めた当初はほとんどの学生がまともにフローチャートを書けていなかった．たとえば処理に下からフローを入れる，条件分岐でないのに分岐しているといった具合で，処理手順以前の問題が多く見られた．

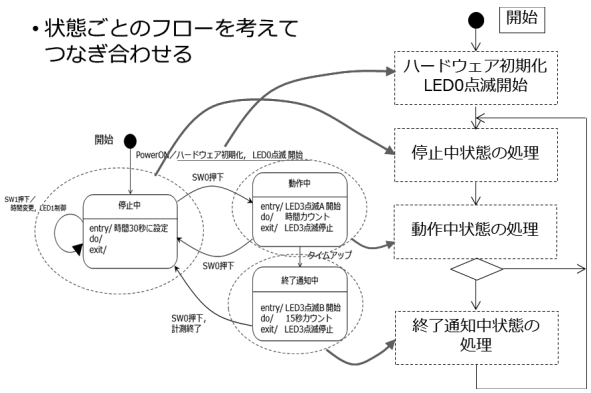

図6 状態遷移図と処理フローの対応

Fig. 6 Relation between state diagram and processing flow.

書き方については，受講学生全員に対して，例示して簡単な指導を行うことで改善がみられた．しかし次には，処理手順そのものの問題，たとえば脱出できないループであったり，重複する処理を書いていたりといった問題が多数みられることとなった．これに対して，やむをえず個別指導を行ったが，これには学生の書いたフローを瞬時に解読して問題点を指摘するスキルが求められた．つまりは指導教員のスキルに強く依存することとなり，教材設計として好ましくない．次回改版時の要改善ポイントと考えられる．

4日目以降は，各自プログラミング作業を進めた．4日目に17件中5件（29%）と目立ったプログラミングに関する困難は，5日目には13件中2件（15%）となりその後消えている．授業観察によれば，自力で書き方を調べたり，他学生に相談したりして解決できていた．

一方で，スイッチの制御方法に関する問題は，最終日である6日目になっても一定の割合で残っていた．アンケート回収率が4件と極端に低い中2件（50%）の意見があり，実際に授業観察でも完成に至らなかった学生の大半が，スイッチ制御で困っていた．このことは，先に示した開発環境の演習の進捗に現れた結果と一致する．スイッチ入力の設計には，もう少し時間をかける，解説を充実させるなどの改善が必要と考えられる．

6. おわりに

開発工程に従う組込みソフトウェア開発演習について，教材の概要と試行結果を述べた．3.2.4項で述べた行動目標に対しては，次のような所感を得た．目標とした5つのプロダクト項目に対し，「Arduino開発環境の使い方」「LEDの制御方法」は，ほぼ全員が達成できた．

「タイマの使い方」「複数のタスクを同時実行させる方法」は，基本設計後に処理フローを考えさせることで，ほぼ理解に達していると考えられる．「スイッチの特性と使い方」は，完成未達学生の主要因となっており，これに関する教材の改善が必要と考えられる．

プロセスでは，学習記録にきちんとその日のプロセス内容が反映しており，一定の理解が得られたと考えられる．

プロジェクトについては，演習ごとにこまめに作業時間設定を行ったが，学習記録にまったくキーワードが現れていないことから，理解を得られたとはいい難い．ガントチャートなどでもっと明示的にスケジュールを示し，意識づけすることが必要と考えられる．

プロフェッショナリズムとしてあげた，「不明点を自力で調査できる」「周囲の学生と相談できる」は，授業観察上

はよくできていた．また，「最後までやりきる」については，特に前半組についてプログラム完成に至らなかった学生が多かったものの，前半組後半組とも最後まで完成を目指して作業に集中できていた．このことは，今回の評価条件である，自力での調査を目的としてインターネットを使うことができること，周囲の学生と相談可能とするといった施策が有効であったためと考えられる．また，早期に課題を完成した学生には追加課題を与えており，これもほぼ全員最後まで緊張を持って取り組むことができていた．

現在は，本稿であげた問題点を改良した教材での授業実践を進めている．今後，この実践での教育効果を検証する．

謝辞 本教材で使用したキッチンタイマの仕様は，森孝夫氏（デンソー株式会社）の考案による．利用を快諾いただき，謹んで感謝の意を表する．

参考文献

[1] 渡辺晴美，吉田正廣：大学学部における組込みソフトウェア教育事例，研究報告組込みシステム，(EMB)，Vol.2011-EMB-20, No.50, pp.1–6 (2011).
[2] 成長分野を支える情報技術人材の育成拠点の形成組込みシステム分野，入手先 ⟨http://emb.enpit.jp/enpit2/⟩（参照 2018-05-01）.
[3] 松澤芳昭，塩見彰睦，秡川友宏，酒井三四郎：ソフトウェア開発の教員主導型 PBL における反復プロセスと EVM 導入の効果，情報処理学会研究報告，IPSJ SIG Technical Report, Vol.2009-CE-99, No.9 (2009).
[4] 沢田篤史，小林隆志，金子伸幸，中道 上，大久保弘崇，山本晋一郎：飛行船制御を題材としたプロジェクト型ソフトウェア開発実習，情報処理学会論文誌，Vol.50, No.11, pp.2677–2689 (2009).
[5] Fioravanti, M.L., Sena, B., Paschoal, L.N., Silva, L.R., Allian, A.P., Nakagawa, E.Y., Souza, S.R.S., Isotani, S. and Barbosa, E.F.: Integrating Project Based Learning and Project Management for Software Engineering Teaching: An Experience Report, *SIGCSE'18* (2018).
[6] 河西理恵，丸山仁司：PBL の学習効果と学生因子の関係について，理学療法科学，Vol.25, No.2, pp.203–208 (2010).
[7] 松浦佐江子：実践的ソフトウェア開発実習によるソフトウェア工学教育，情報処理学会論文誌，Vol.48, No.8, pp.2578–2595 (2007).
[8] 平山雅之：組込みソフトウェアの現状，情報処理，Vol.6, No.45, pp.677–681 (2004).
[9] 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター：組込みソフトウェア向け開発プロセスガイド，30, 36, 81 (2007).
[10] 安永 航，大場みち子，奥野 拓，伊藤 恵，山口 琢：PBL を対象としたインフォーマルラーニング環境の構築，情報処理学会研究報告，IPSJ SIG Technical Report, Vol.2013-CE-121, No.10 (2013).
[11] 舘 伸幸，山本雅基，吉田則裕，高嶋博之，海上智昭，安藤友樹，松原 豊，本田晋也，高田広章：OJL による実践的組込みシステム教育，コンピュータソフトウェア，Print ISSN 0289-6540 (2015).
[12] IPA：IT 白書 2011 (2011-5-20).
[13] 成長分野を支える情報技術人材の育成拠点の形成—組込み分野，名古屋大学事業，入手先 ⟨http://emb.enpit.jp/ojl/index.html⟩（参照 2018-09-13）.
[14] 舘 伸幸，山本雅基，高嶋博之，松原 豊，本田晋也，高田広章：enPiT-Emb（名古屋大学）OJL による実践的組込みシステム教育，信学技報，Vol.113, No.498, pp.181–186 (2014).
[15] 経済産業省：社会人基礎力，入手先 ⟨http://www.meti.go.jp/policy/kisoryoku/⟩（参照 2018-05-01）.
[16] 鈴木克明：教材設計マニュアル，北大路書房 (2002).
[17] Arduino, available from ⟨https://www.arduino.cc/⟩ (accessed 2018-05-01).
[18] ABOUT THE UNIFIED MODELING LANGUAGE, SPECIFICATION VERSION 2.5.1, available from ⟨https://www.omg.org/spec/UML/About-UML/⟩ (accessed 2018-05-01).
[19] 飯泉紀子，田所孝文，大立 薫，平島直人，井上博史：組み込みソフトウェア開発における品質向上への取り組み，日本科学技術連盟 SQiP Library，入手先 ⟨https://www.juse.or.jp/sqip/workshop/report/attachs/2008/3-A-report.pdf⟩（参照 2018-05-01）.
[20] 大場みち子，伊藤 恵，下郡啓夫，薦田憲久：論理的な文章作成力とプログラミング力との関係の分析，研究報告コンピュータと教育（CE），Vol.2015-CE-132, No.27, pp.1–5 (2015).

舘 伸幸 （正会員）

名古屋大学大学院情報学研究科附属組込みシステム研究センター研究員．元 NEC マイクロシステム所属．組込みソフトウェア技術者．

山崎 進 （正会員）

北九州市立大学情報メディア工学科情報工学専攻コンピュータシステムコース准教授．博士（理学）．

香山 瑞恵 （正会員）

信州大学工学部教授．博士（工学）．電気通信大学大学院助手等を経て，2014 年より現職．研究テーマは学習支援工学，教育工学，知識処理．