Vol.2013-SE-180 No.4
Vol.2013-EMB-29 No.4
2013/5/27

発展的組込みソフトウェア作成教材の開発

谷川郁太[†1]　福山祐哉[†2]　渡辺晴美[†1]

本稿では，組込みソフトウェア教育のためのソフトウェア開発事例について紹介する．教材の要件は，学生が容易にプログラム可能であること，様々な開発環境や通信方式に対応できることである．さらに教育システムは様々な用途で用いられることから発展性が求められる．これらの要件を満たすために，本開発では２つの工夫を施した．まず，学生は C 言語風のスクリプトを記述し，C#の.net フレームワークで動作するシステムを開発した．また，様々な開発環境や通信方式に耐え，発展性を加味するために，通信プログラムの継承機能とプラグイン機能を用意した．

A Development of Evolutional Education Software for Embedded System

IKUTA TANIGAWA[†1]　YUYA FUKUYAMA[†2]　HARUMI WATANABE[†1]

The article presents a case study of developing software for embedded software education. It is requirements of educational software that learning is easy. Additionally, handling multiple development environments and communication protocols are required. Since educational software is used in various situations, evolutional structure of software is desired. To overcome these requirements, we provide two main features. First, students can describe the script program such as C language, and the program is executed on the .net framework of C#. Second, in order to handle various development environments, communication protocols and evolve the system, communication function is implemented with inheritance and plug-in.

1. はじめに

組込みソフトウェア開発には，ソフトウェア・ハードウェアの両方の知識を必要とする．各々の分野が未習熟の初学者が段階的に学ぶことができ，高い学習意欲を持てる教材を提供することは容易ではない．

組込みソフトウェア開発の教材には，様々な題材があり，それぞれコンテストが開催されている[1]．その多くはハードウェアを中心としたものが多い．ET ロボコンとして知られている LEGO を利用した ET ソフトウェアデザインロボットコンテストでは，ソフトウェアのモデルについて競う[2]．LEGO のシステムは，ハードウェアに関する知識が乏しい状況でも動作させることが可能である．ハードウェアについても学びやすくすること，実践的な要求分析を可能とするために，飛行船を題材としたコンテストが開催されてきた[3].

LEGO や飛行船は親しみやすい教材ではあるが，日常で利用する組込みシステムではない．実践的なソフトウェア開発を体得するには，学習意欲を向上させるために，日常的に利用している題材が望まれる．家電製品を教材として用いた産学連携の PBL 事例として扇風機がある[4]. 扇風機の事例では，産業界の開発方法と同じ形式で行われているため，初学者には難しい．

扇風機をはじめ多くの家電製品は，センサの値に応じて何がしかの処理を加えモータを動作させる機構を持つものが多いため，ロボットを初学者用として位置付ける教材が多い．ロボットの開発を行う際には，開発の初期段階で，ハードウェアの動作特性を調べる必要がある．この段階では，ロボットを動作させるのに十分なソフトウェアを組込むことができないので，ラジコン操作により動作を目視で確認するとともに，データを収集する．

以上から，(1) 段階的に学べること，(2) 日常的に用いられること，(3) ラジコン等が可能な通信機能を搭載できることの 3 点を考慮し，我々は，掃除機システム[5]のための組込み制御通信ソフトウェアを開発した．本ソフトウェアは，学習者は PC 上で掃除機システムの振る舞いに関するプログラムを記述し，その内容を，システムに無線で指令を出し，動作させるソフトウェアである．

上記(1)の段階的な学習を可能とするために，ハードウェア依存部分を隠蔽し，C 言語風にシステムの振る舞いをスクリプトで記述することができるようにしている．C 言語風であること，PC 上で動作していること，ハードウェア知識を必要としないことから，大学初年次の教育を受けた段階で利用することが可能である．スクリプト言語が教育に有益であることは知られている[6]．上記(2)に関して，掃除機を題材としていることで，学生が日常的な製品を作成しているという実感が持てるようにしている．

さらに(3)に関して本教材は，掃除機以外のロボットに対しても，最小限の変更でラジコン動作およびデータ収集を可能なシステムを提供している．以上，(1)のために学生が記述した C 言語風のプログラムは，C#のフレームワークで動作させる．また，(3)の開発環境や通信方式に耐え，発展

†1 東海大学大学院情報通信学研究科
Tokai University Graduate School of Information and Telecommunication Engineering
†2 東海大学情報通信学部
Tokai University School of Information and Telecommunication Engineering

Vol.2013-SE-180 No.4
Vol.2013-EMB-29 No.4
2013/5/27

性を加味するために，通信プログラムの継承機能とプラグイン機能を用意した．

本稿で紹介するソフトウェアは，現在，大学の授業で利用している．今後，様々な教育イベントで用い，評価，洗練を行っていく予定である．

以下，2 章では，紹介する教材の概要について，3 章ではそれを可能とするシステムの構造について述べる．

2. 組込み制御通信ソフトウェア

本章では教材である組込み制御通信ソフトウェアの構成や通信プログラムの作成方法について述べる．

2.1 教材の構成

本教材は，学生が通信プログラムを作成するためのエディタと，作成したプログラムを実行させるための実行ソフトによって成り立っている．実行ソフトは作成したプログラムを読み込み，プログラムに記述された振る舞いを基に組込み機器と通信を行う．

図 1 に教材利用の流れを示す．初めに，エディタを用いてスクリプトや各種設定ファイル，操作画面のデザインなどを編集することで，通信プログラムを作成している．次に，実行ソフトから作成プログラムを読込み，その内容を解釈することで，読み込んだプログラムの通りに通信を実行する．このような流れで，プログラムを作成，実行する．

図 2 に教材の構成を示す．教材は実行ソフトとエディタの他に，通信プログラムを置くためのフォルダがあり，そこにあるプログラムをエディタから編集，あるいは実行ソフトから実行する．さらにプラグイン機能にも対応しており，通信プログラム中で使えるライブラリを追加することも可能である．

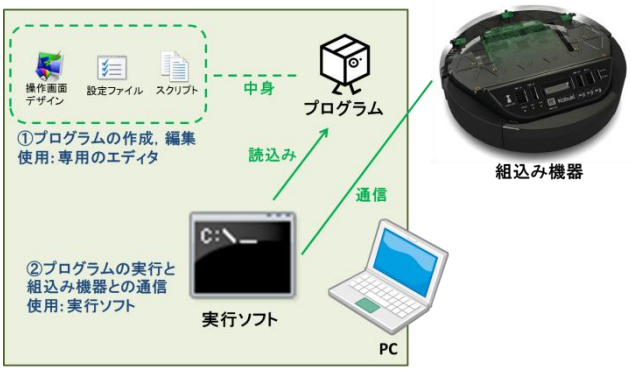

図 1 教材利用の流れ

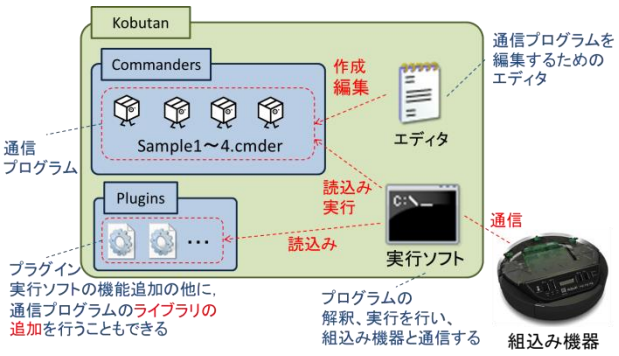

図 2 教材の構成

2.2 通信プログラムの作成

通信プログラムは，プログラムの振る舞いを記述するスクリプトや各種設定(データ送信の周期や使用するプログラミング言語など)を行うファイル，操作画面のデザインなどによって構成されている．この中で特に大事なのが，通信の振る舞いを記述するスクリプトである．本節では，掃除機システムを動かすためのプログラム例と用意されているコマンド，受信データ，関数の一覧を表 1 に示す．

2.2.1 プログラムの例

リスト 1 に学生が記すプログラムの例を示す．このプログラムは掃除機システムを 1 秒間前進させ，その後 2.5 秒間左旋回して，1 秒間前進するプログラムである．このようにスクリプトは C 言語とほぼ同じ形式で書くことができる．

プログラムの流れとしては，初めのステップで Forward コマンドを実行し，掃除機システムに前進するように指示を出している．次のステップで Sleep コマンドを呼び出し，次のステップまで 1 秒間待つ．このように，しばらく待つコマンドを入れないと，最初から最後の行まで一瞬で到達し，そのまま終了してしまって動作確認ができないからである．次に Stop コマンドを呼び出して，前進を終了する．以降も同様に旋回や前進のコマンドを出した後，Sleep コマンドにより指定した時間待つことで，その処理を指定した時間行わせている．最後に Exit コマンドを呼び出してプログラムを終了している．

このプログラムでは，掃除機システムを簡単に動かすためのフレームワークとライブラリを使用している．これによって，ComMain で Forward などのコマンドを呼び出すだけで，簡単にプログラミングできる．フレームワークやライブラリの詳細は 3.2.2 で述べる．

Vol.2013-SE-180 No.4
Vol.2013-EMB-29 No.4
2013/5/27

リスト 1 学生が記すプログラムの例

```
// エントリポイント
void ComMain()
{
    // 1秒間 前進
    Forward();          // 前進
    Sleep(1000);        // 次ステップまで1秒待つ
    Stop();             // ストップ
    // 2.5秒間 左旋回
    TurnLeft();         // 左旋回
    Sleep(2500);        // 次ステップまで2.5秒待つ
    Stop();             // ストップ
    // 1秒間 前進
    Forward();          // 前進
    Sleep(1000);        // 次ステップまで1秒待つ
    Stop();             // ストップ

    // 終了
    Exit();
}
```

2.2.2 コマンド・関数一覧

表1にコマンドの一覧，表2に受信データの一覧，表3に関数の一覧をそれぞれ示す．ここでいうコマンドとは，掃除機システムの動きを制御するための関数であり，受信データはデータ受信の度に内容が更新される変数である．学生はこれらの用意されたライブラリを用いて通信プログラムを作成する．

表1 コマンド一覧

分類	定義	概要
移動コマンド	void Forward()	まっすぐ前進
	void Backward()	まっすぐ後進
	void TurnLeft()	その場で左に旋回
	void TurnRight()	その場で右に旋回
	void Stop()	その場に停止する
設定コマンド	void SetSpeed(int speed)	直進速度の設定 (0～100) [10mm/s]
	void SetTurnSpeed(int speed)	旋回速度の設定 (0～100)
	void SetLEDColor (int num, char color)	LEDの色設定 num : 色設定するLEDの番号(1,2) color : 'N'(無し), 'R'(赤), 'G'(緑), 'Y'(黄)
システムコマンド	void Sleep(int ms)	指定された時間(ミリ秒) 次のステップに移るのを待つ
	void Exit()	プログラムを終了する このコマンドが呼ばれず最終行に到達すると初めの行に戻り再び実行される

表2 受信データ一覧

分類	定義	概要
バンパ	int RightBumper	右バンパが押されているか
	int LeftBumper	左バンパが押されているか
	int CentralBumper	中央バンパが押されているか
ボタン	int Button1	ボタン1が押されているか
	int Button2	ボタン2が押されているか
	int Button3	ボタン3が押されているか
ホイール	int RightWheelDrop	右ホイールが浮いているか
	int LeftWheelDrop	左ホイールが浮いているか
エンコーダ	ushort RightEncoder	右モータのエンコーダ値
	ushort LeftEncoder	左モータのエンコーダ値

表3 関数一覧

分類	定義	概要
出力関数	void OutputString(string str)	渡された文字列を出力
	void OutputInteger(int i)	渡された整数を出力
	void OutputFloat(double f)	渡された浮動小数を出力
数学関連	double Sin(double a)	角度a(ラジアン)のsinを返す
	double Cos(double d)	角度d(ラジアン)のcosを返す
	double Tan(double a)	角度a(ラジアン)のtanを返す
乱数	int GetRandomValue()	0～2,147,483,647の範囲の乱数を返す

2.3 通信プログラムの実行

通信プログラムを実行ソフト上で動かしている様子を図 3 に示す．上の「コマンダウィンドウ」が実行するプログラムを選択するためのウィンドウである．ウィンドウの左側に実行するプログラムを選ぶスペースがあり，選択したプログラムの情報がウィンドウの右側に表示される．実行するプログラムを選択し，ウィンドウの下側にあるインスタンス名,ポート名を設定してボタンをクリックすると，選択されたプログラムのインスタンスが生成され，生成されたインスタンス固有のウィンドウが新たに表示される．以後，このウィンドウを「インスタンスウィンドウ」と呼ぶこととする．

インスタンスウィンドウはそれぞれのインスタンス固有のものとなっており，このウィンドウを用いることによって，インスタンスに各種操作を行う．ウィンドウの下側にある「通信開始」ボタンをクリックすることで，組込み機器との通信を開始する． 2.2 節に示したプログラムはこのボタンをクリックした時点で，動作を開始するようになっている.プログラムが Exit()まで到達するか,「通信終了」ボタンがクリックされることで，この通信プログラムは終了する．

インスタンスウィンドウには，上記の他にもいくつかの操作が用意されており，ウィンドウの左側から各種表示したい画面の名前を選ぶことで，右側に選んだ画面が表示される．画面は元から用意されているものの他に，プログラム作成者によって追加された画面も含まれる．図 4 に実行中のプログラムのプロパティ値を変更する画面を示す．これにより，パラメータを変更することでプログラムの挙動がどのように変わるのかを実際に動かしながら確認することが可能となる．

Vol.2013-SE-180 No.4
Vol.2013-EMB-29 No.4
2013/5/27

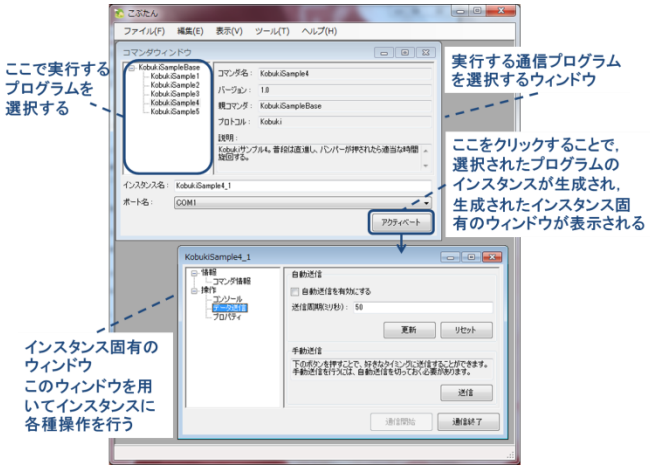

図 3 通信プログラムの実行画面

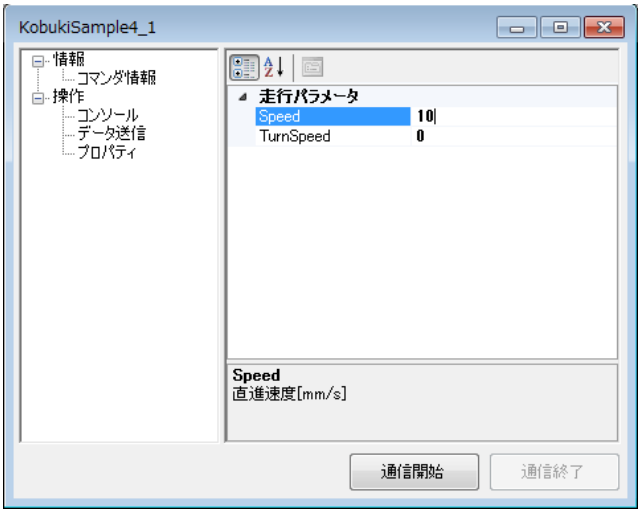

図 4 プロパティ変更画面

3. 組込み制御通信ソフトウェアの構造

本章では，C 言語風のスクリプトを利用し段階的に学べること，様々な開発実行環境および通信方式を容易に実装可能にするために施した内部構造の工夫について述べる．

3.1 共通部分と変更部分

本教材には，特定の組込み機器や通信手段などに依存せずに，あらゆる通信プログラムや組込み機器で扱うことができる箇所とそうでない箇所に分かれている．前者を共通部分とし，通信プログラムの実行ソフトや通信プログラム上で使われるフレームワーク，標準ライブラリとして実装している．後者は変更部分にあたり，通信プログラムそのものやそこで使われる追加ライブラリなど，ユーザによって作成される箇所としている．図 5 に上記の様子を示す．

このように作ることで，通信の振る舞いのように頻繁に変更される箇所を通信プログラムとして分離でき，バイナリをコンパイルし直す必要が無くなる．また，通信プログラムをスクリプトによって実装することで，開発環境を持たないユーザでも通信プログラムを作成することができる．

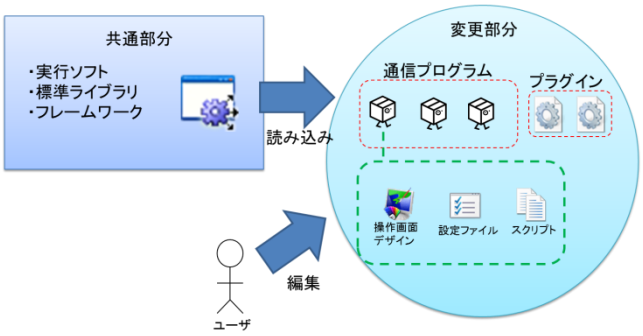

図 5 共通部分と変更部分

3.2 通信プログラム

本節では，実行ソフト上に読み込まれる通信プログラムについて述べる．

3.2.1 通信プログラムの形式

教材の上で扱う通信プログラムは 2.1 節で述べたとおり，プログラムの振る舞いを記述するスクリプトや各種設定を行うファイル，操作画面のデザインなどから構成される．

教材では，上記ファイルを Zip 圧縮によってパッケージ化し，それを一つのプログラムとして扱っている．実行ソフトでは，作成したパッケージを読み込みそこからインスタンスを生成する．組込み機器との通信は生成されたインスタンス単位で行う．これにより，同じプログラムを別々の機器に対して同時に実行するといったことが実現できる．

さらに，パッケージは他のパッケージを継承することによる差分開発も行える．これによって，教材作成者が前もって便利なメソッドやフレームワークを用意した親パッケージを作成しておき，それを教材利用者に継承させることで利用者は楽に通信プログラムを作成することができる．この話に関しては次項でさらに詳しく述べる．図 6 にパッケージの作成とそれを実行する流れを示す．

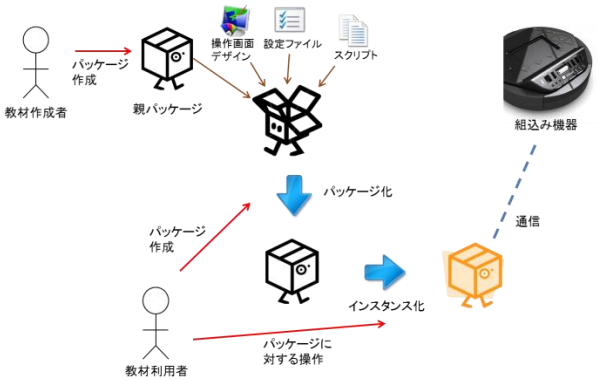

図 6 パッケージ(通信プログラム)の作成，実行

3.2.2 継承による通信プログラムの簡単化

前項の後半で通信プログラム(パッケージ)は他の通信プログラムを継承できることを示した．継承元でメソッドやフレームワークを追加することにより，細かな部分を隠して簡単に通信プログラムが実装できるようになる．

図 7 はリスト 1 のように ComMain 上で実行できる仕組みを実現するクラス図である．通常の通信プログラムの場合は，バイナリ側の CommanderBase クラスを継承し，送受信時や接続・切断時に呼ばれるメソッドをオーバーライドすることで通信プログラムを作成する．一方，図 7 では，SampleBase という通信プログラム(クラス)で CommanderBase クラスを継承し，その上に ComMain によるフレームワークを構築している．さらなる継承先の Sample1～3 では送受信時や接続・切断時に呼ばれるメソッドを直接オーバーライドすることはせずに，ComMain のみをオーバーライドしている．これによって，通信の細かい部分が隠され，教材利用者はリスト 1 のように，ComMain に前もって用意されたメソッドを呼び出して行くことで通信プログラムを作成することができる．

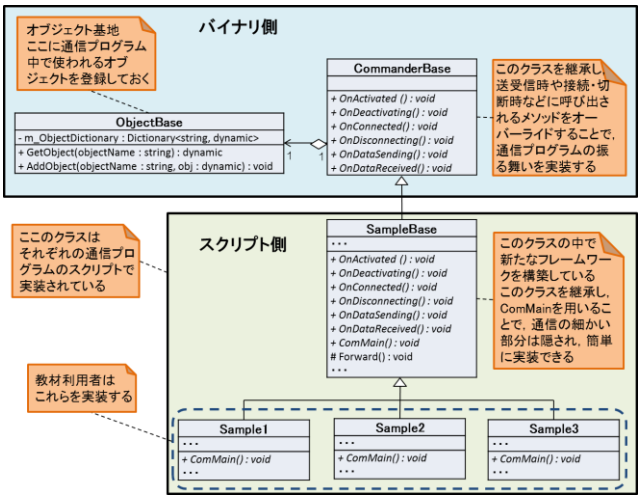

図 7 ComMain を実現するための仕組み

3.2.3 スクリプト

教材中で扱えるスクリプト言語は C#である．これは，内部で C#のコンパイラ(CSharpCodeProvider)を呼び出し，コンパイルすることで実現している．C 言語しか使ったことがない学生にとっていきなり C#を使うことは困難である．そのため，C#を C 言語に近づけるための工夫が必要となる．

C#と C 言語の違いはクラス，名前空間の有無や組込み型のサイズ，配列の作り方などである．これらの違いの中で特に大きいのがクラスや名前空間の有無である．そのため，クラス定義や先頭の using を補う機能を用意した．これによって，学生はクラス定義を気にすることなく，C 言語とほぼ同じ感覚でプログラミングすることができる．

図 8 に実際にソースコードを補完する様子を示す．ここでは，学習者が書いた C 言語とほぼ同じような形式のソースコードにクラス定義などを補ったソースコードを内部で生成し，それをコンパイラに渡すことで実行可能な形としている．

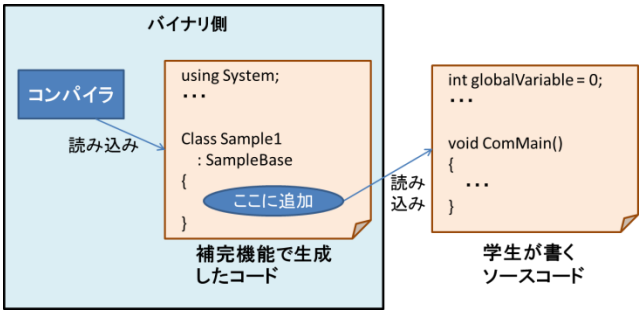

図 8 ソースコードの補完

3.3 ライブラリの追加

ライブラリの追加には 2 つの方法がある．1 つは 3.2.2 で示したように，継承元にメソッドを追加する方法である．この方法により追加したライブラリは継承先でのみ使用できるので，使用範囲を限ることができる．よって，特定の機器や環境でのみ使用されるライブラリはこの方法で追加すると良い．2.2.2 で示したライブラリは掃除機システムでしか使わないので，全てこの方法を用いている．

もう一つの方法はプラグインを用いる方法で，これによって追加されたライブラリは全ての通信プログラム中で使えるようになる．そのため，ラジコン操作用のゲームコントローラライブラリなどはこの方法を用いて追加するのが良い．図 9 に上記 2 通りの方法によって追加されたライブラリの利用可能範囲をそれぞれ示す．

上記のいずれかの方法によって追加したライブラリを使っている通信プログラムはそれらのライブラリに依存してしまっているので，別の環境で実行する場合は依存するファイルも用意する必要がある．

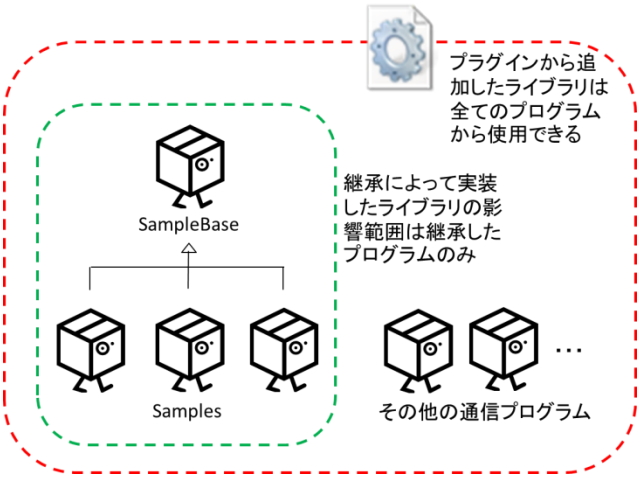

図 9 追加したライブラリの利用可能範囲

Vol.2013-SE-180 No.4
Vol.2013-EMB-29 No.4
2013/5/27

4. おわりに

本稿では，組込みソフトウェア開発のための組込み制御通信ソフトウェアについて紹介した．本ソフトウェアは，(1) 段階的に学べること，(2) 日常的に用いられること，(3) ラジコン等が可能な通信機能を搭載できることを目的として開発を行った．現在，100 名以上の学生に本教材を用いて教育を行っている．また，通信機能に関しては，Bluetooth および Zigbee の 2 種類の通信方式に適用した．

今後は，教育効果を計測し，機能追加およびユーザインタフェースの向上を図っていく予定である．

参考文献

1) 小倉信彦，渡辺晴美: ロボットコンテストを利用した組込み教育の実践，情報処理学会論文誌, Vol.49, No.10, pp.3531-3540, Oct. (2008)
2) ET ソフトウェアデザインロボットコンテスト(ET ロボコン) http://www.etrobo.jp/2013/
3) 渡辺 晴美，三輪 昌史，久住 憲嗣，小倉 信彦，紫合 治:ESS ロボットチャレンジ 2012，組込みシステムシンポジウム 2012 論文集，情報処理学会，pp. 205-208，(2012)
4) 経済産業省: 組込み技術者教育事例集, 2009 http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2009software_research/
5) Kobuki　http://kobuki.yujinrobot.com/
6) 志田 駿介，菅谷 みどり，倉光 君郎: Tiny Konoha: ET ロボコン向けのスクリプト処理系の簡素化，組込みシステムシンポジウム 2012 論文集，情報処理学会, pp. 31 – 38, (2012)