Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

Squareknot:コントローラとアクチュエータのためのフレームワークの実装

氏 縄 武 尊[†1] 林 原 尚 浩[†1]

ロボットなどのアクチュエータを任意のコントローラで操作するためには，それらを接続するためのプログラミングが必要となる．しかしながら，それぞれのモジュール化を行わずにコードを記述すると，理解が困難で拡張が容易で無く，また柔軟でないプログラムとなってしまう．本研究では，コントローラでの操作とアクチュエータの動作を独立して記述し，それらの動作を柔軟に結びつけることが出来る MAC(Mapping between Actuators and Controllers) モデルを提案し，それに基づいたフレームワーク SquareKnot を開発した．

Squareknot:Framework implementation for Controllers and Actuators

Takeru Ujinawa [†1] and Naohiro Hayashibara [†1]

To operate an actuator (e.g., robots) with a certain controller, programming for connecting between them is required. However, programming without modularization makes difficult to understand and extend it. It also leads to lose flexibility. In this paper, we show an implementation of programming framework for connecting between controllers and actuators based on MAC (Mapping between Actuators and Controllers) model.

†1 京都産業大学コンピュータ理工学部
Faculty of Computer Science and Engineering, Kyoto Sangyo University

1. はじめに

1.1 背 景

近年，Wi-Fi や Bluetooth 等の通信を用いて操作や管理を行うことのできる家電，玩具等が多く開発されている．

中でも，プログラミング教育用ロボット LegoMindstorms[1] や家庭用掃除機 Roomba[2]，iPhone や Android 用ラジコン AR.Drone[3] 等，ユーザがプログラムを書くことにより手軽に操作することができるロボット（アクチュエータ）が増加している．

また同じく，Wii Remote Controller や Wii Balance Board[4]，PlayStation[5] のコントローラ，Kinect[6] 等のゲーム機器のコントローラや従来の携帯電話，スマートフォン等，ユーザがプログラムを書くことにより手軽に入力できるコントローラも増加している．

それによりユーザは，Wi-Fi や Bluetooth 等，共通のインターフェースを持つコントローラを用いて自分が操作したいアクチュエータを動かすといったプログラムを記述することが可能になった．

1.2 問 題 点

前節で述べたように，コントローラを用いて共通のインターフェースを持つアクチュエータを動作させるプログラムを記述する場合，以下のような問題が生じることある．

- 既存のライブラリは導入コストが高い（インストール作業，初期設定など）．
- 新たなコントローラやアクチュエータを追加することが容易ではない．
- 仕様の異なるコントローラとアクチュエータに関してプログラムを行うため，プログラムが複雑になる．

1.3 研究の目的

そこで本研究では，以下の特徴を持つコントローラとアクチュエータのためのフレームワーク Squareknot を提案する．

- 導入容易性：簡易な初期設定，GUI によるユーザ支援．
- 柔軟性：コントローラの操作とアクチュエータの動作を自由に対応付けできる．
- 拡張性：新しいコントローラやアクチュエータ用のコードを容易に追加できる．
- 可読性：それぞれのコードが独立している．

このフレームワークは，コントローラの操作とアクチュエータの動作をそれぞれの別々のプログラムとして独立に記述を行い，それぞれの持つ操作・動作を柔軟に対応付けて接続を可能にする MAC モデルに基づいている．このモデルにより，コントローラの操作とアクチュ

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

エータの動作のためのソースコード記述を完全に独立させることができるため，個々の処理が 1 つのソースコードに混在することがない．そのため，どの操作・動作がどこに記述されているのか探すことが容易になり，読みにくいコードとなる事を防ぐことができる．

また，操作と動作をマッピングするという手法により，コントローラの操作とアクチュエータの動作の対応を容易に切り替えることができる．そして，新たなコントローラやアクチュエータを追加したい際も，それぞれのソースコードを記述するだけで拡張できる．

2. 関 連 研 究

本研究では，コントローラからロボットを操作するためのモデルの提案とフレームワークの開発を行なっている．関連研究としては，以下のようなものが上げられる

2.1 ROS（Robot Operating System）

ROS は，Willow Garage 社によってオープンソースで提供されており，ロボットを動作させる上で必要なよく使われる関数，プロセス間通信，パッケージ管理などの機能を有するロボット用のオペレーティングシステムである[7]．

非常に多くのロボットを対象としていて，さらにシミュレータなども実装されている．また，それらの対象を操作するためのコマンドなども多く提供されている．

しかしそれら多くの機能を保持しているため，ROS 自体のインストールに手間がかかってしまうことや，必要なツール群の中で何を使って良いのかの判断が難しい等の問題点がある．

2.2 RTM（Robot Technology Middleware）

RTM とは，ロボット等に備え付けられたセンサー等のモジュールを組み合わせることで，システムを構築するというソフトウェアプラットフォームである．

OpenRTM-aist[8] は，その実装の一つであり，オープンソースで提供されている．実装は，センサ，カメラ，モータ等のモジュールをポートと呼ばれるインターフェースでつなぎ，制御アルゴリズム等を書き加えることで，新たな合成モジュールを生成し，それを RT コンポーネントと呼び，そうして作られた複数の RT コンポーネントを用いてロボットを制御するという仕組みである．

これを利用することにより，機器との接続、状態遷移の監視等が視覚的に確認しやすくなるなどの利点がある．

またこの実装は C++,Java,Python 等の多言語での記述に対応しているため，可用性にも富んでいる．

2.3 考　　察

これらの研究は，利用するための下準備や利用できるコマンド数等が多く，また新たな機器を追加する際にも多くのコンポーネントを作成しなければならないなど，導入が容易では無い部分があるといった問題があるが，本研究で提案している Squareknot では，既に準備されている場所に，自分の作成したクラスを追加するだけでコントローラの追加が可能になるため，導入が容易となる．

また，動作を切り替える際には，やはりそれぞれのソースコードを改変する必要があり，改変が困難になる可能性が高い．その点においても Squareknot が優れていると言える．

3. MAC モデル

本研究で提案する MAC モデルとは，何らかのコントローラとアクチュエータが存在し，それらの操作や動作がプログラムによって記述可能な場合，それらの機器の操作と動作をマッピングすることにより，コントローラの操作それぞれによるアクチュエータの動作を動的に決定・変更できるようにすることが可能となるモデルである．

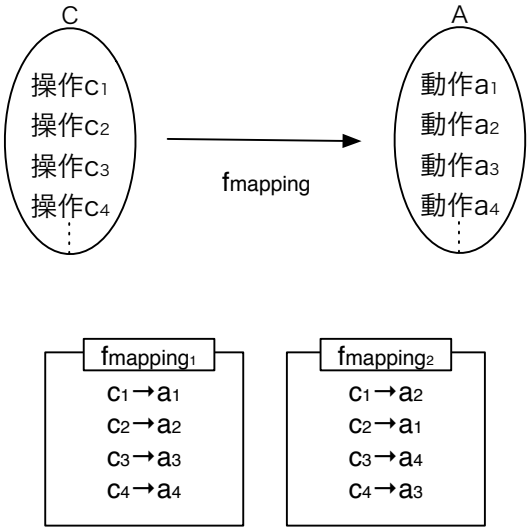

図 1 MAC モデル 1

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

このモデルでは，コントローラを，複数の操作 $c_1, c_2, c_3, \cdots, c_n$ を持つ操作集合 C，アクチュエータを，複数の動作 $a_1, a_2, a_3, \cdots, a_n$ を持つ動作集合 A と定義する．

コントローラの操作 c_i でアクチュエータの持つ動作 a_i を実行したい場合，それらを対応付けなければならない．この対応付けを行うのが写像 $f_{mapping}$ であり，次のように定義できる．

$$f_{mapping}(C) = \{a \mid a = f_{mapping}(c), c \in C, a \in A\}$$

$f_{mapping}$ によってコントローラの操作とアクチュエータの動作の柔軟な対応付けを実現することができる．また，コントローラの複数の操作が，アクチュエータ側の 1 つの動作に対してマッピングされていても良い．

4. 研究内容

本研究では任意のコントローラとアクチュエータを接続し動作させるためのフレームワーク Squareknot を提案する．プログラミング言語は，Java 言語を利用．コントローラ及びロボットとの通信インターフェースは Bluetooth と Ethernet を仮定する．

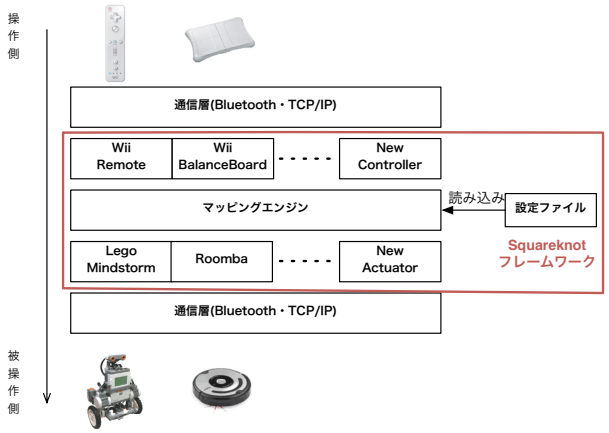

図 2 Squareknot のアーキテクチャ
Fig. 2 Architecture of Squareknot

Squareknot フレームワークは，図 2 のように，コントローラクラス群とアクチュエータクラス群，設定ファイルそしてマッピングエンジンから成っている．それらの役割は以下の通りである．

- コントローラクラス群：コントローラの操作をメソッドに持ったクラスの集まり．
- アクチュエータクラス群：アクチュエータの動作をメソッドに持ったクラスの集まり．
- 設定ファイル：操作と動作の対応付けを記述したファイル．
- マッピングエンジン：コントローラとアクチュエータを繋ぐ部分．

現時点の Squareknot フレームワークにおいては，図 2 の赤枠内に示されている通り，コントローラとして，任天堂の Wii Remote Controller，Wii Balance Board[4] に対応するコントローラクラスを実装し，ロボットは LEGO 社の LegoMindstorm[1] のためのクラスとデバックのためにコンソールに出力を行うクラスを実装している．

4.1 実装詳細

Squareknot フレームワークを実装するにあたって使用したライブラリを表 1 に示す．

ここでは先ほど記述した，Squareknot フレームワークの重要な 4 つの構成要素について説明する．

ライブラリ名	使用箇所
Apache Commons Configuration[9]	設定ファイル
Bluecove[10]	Bluetooth 通信
WiiRemoteJ[11]	WiiRemote コントローラの使用
Lejos[12]	LegoMindstorms の使用

表 1 使用ライブラリ
Table 1 Related libraries

コントローラクラス群

Squareknot フレームワーク内には，コントローラが持つべき必要最低限の操作を記述してもらうための Controller インターフェースがあり，ボタンベースのコントローラであれば，表 2 が示すような処理が記述されている．ユーザはこのコントローラインターフェースを用いてコントローラクラスを実装する．

アクチュエータクラス群

アクチュエータにおいても，アクチュエータにあるべき動作を記述してもらうための Actuator インターフェースがあり，二輪駆動のロボットのためのクラスであれば，表 3 が示すような処理が記述されている．ユーザはこのアクチュエータインターフェース用いてアクチュエータクラスを実装する．

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

内容	メソッド名
上ボタンを押す	upButton
下ボタンを押す	downButton
左ボタンを押す	leftButton
右ボタンを押す	rightButton
機器との接続を行う処理	connect
機器との接続解除を行う処理	disconnect

表 2 コントローラの操作とメソッドの対応関係
Table 2 Controller

内容	メソッド名
前進を行う処理	gain
後退を行う処理	back
左に方向転換を行う処理	rotateLeft
右に方向転換を行う処理	rotateRight
機器との接続を行う処理	connect
機器との接続解除を行う処理	disconnect

表 3 アクチュエータの動作とメソッドの対応関係
Table 3 Actuator

設定ファイル

コントローラに記述されている操作とアクチュエータに記述されている動作同士をつなぎ合わせるための設定ファイルである．このファイルには図 3 のように，コントローラ側の操作指示が書かれたメソッドのメソッド名とアクチュエータ側の動作指示が書かれたメソッドのメソッド名のペアをコロン（:）で区切って列挙する．それにより操作と動作をマッピングし，この記述を変更することで動作変更が可能になる．

```
upButtont : gain
downButton : back
leftButton : rotateLeft
rightButton : rotateRight
```

図 3 設定ファイル例
Fig. 3 Configuration file

マッピングエンジン

```
Mapping aMapping = new Mapping();

aMapping.addController("ControllerClassName");
aMapping.addActuator("ActuatorClassName");
```

図 4 実行方法
Fig. 4 Squareknot

マッピングエンジンでは，主にコントローラの操作からアクチュエータの動作をマッピングし指示を送る処理を行なっている．図 4 に示すように実行を行うことで Mapping が可能になる．

4.2 使 用 方 法

Squareknot では，図 4 のように，マッピング用のインスタンスを生成し，ユーザに実装してもらったコントローラクラスとアクチュエータクラスをマッピングクラスに与えることで，それぞれのクラスを実行し接続が行われる．

OS	Mac OS X 10.7
Java VM	java version 1.6.0_29
通信インターフェース	Bluetooth・Wi-Fi

表 4 動作環境
Table 4 System requirements

図 4 では，マッピングエンジンのクラス Mapping のインスタンスを作る際に，コンストラクタに引数を与えていないが，コンストラクタの引数に設定ファイルを指定，または Mapping インスタンスの setConfigure メソッドの引数に設定ファイルを指定して実行を行うことで，設定ファイルを選択することもできる．図 4 のように指定しなかった場合は，標準で図 3 のような default.config という設定ファイルが選択されるようになっている．また，このフレームワークでは，図 5 のように，GUI 的にマッピングを行うことができる．表示されているコントローラクラスとアクチュエータクラスは，提供する実行可能ファイル Squareknot.jar ファイルの中に入っているクラスと，提供したソフトウェア内の plugin ディレクトリに入れた追加のクラスである．新たなコントローラを追加したければ，plugin

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

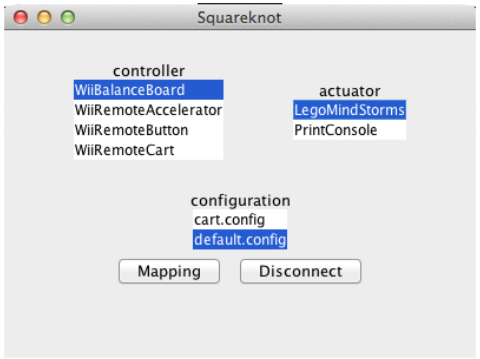

図 5 ユーザインタフェース
Fig. 5 User interface

ディレクトリの中の controller ディレクトリ内に作ったクラスを入れることで新たに表示される仕組みとなっている（図 6）.

それらのクラスが表示されたリストから，利用したいコントローラのクラスとアクチュエータのクラス，設定ファイルを選択し Mapping ボタンを押すことで，それらのクラス同士をマッピングしコントローラの操作によってアクチュエータを動作させることが可能になる.

実行時にコントローラのクラスとアクチュエータのクラスを複数選択することで，それぞれ複数機器を接続するこが可能になる．しかし設定ファイルが複数選択しているとマッピングの対応付けに異常が発生するため，設定ファイルが複数選択されている場合に関してはエラー処理が行われる.

コントローラやアクチュエータが選択されていない場合もエラー処理が行われる．設定ファイルに関しては，選択が無い場合は GUI でない場合と同様，default.config という設定ファイルが選択される.

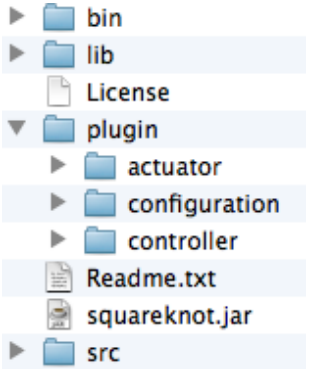

図 6 ファイル階層
Fig. 6 File hierarchy

5. まとめと考察

本論文では，コントローラとアクチュエータの接続を容易にするフレームワーク Squareknot の設計および実装について説明した.

本論文で提案している Squareknot は，導入容易性，柔軟性，拡張性，可読性の向上を目的としたフレームワークである．そこで，これらの点について，2 章で述べた関連研究と比較を行った（表 5 参照）.

研究名	導入容易性	柔軟性	拡張性	可読性
ROS	×	○	◎	△
RTM	△	△	○	○
Squareknot	○	○	○	○

表 5 関連研究の比較
Table 5 Comparison

導入容易性

関連研究で紹介したものについては，導入の際にいくつかのインストール作業を行い，最低限のプログラム等の初期設定を行なう必要がある.

ROS を導入する場合であれば，ROS 利用するまでに，複数のライブラリ（例えば wget

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

コマンドやシミュレータ等）が必要であり，それらをインストールする必要がある．よってそれらのコマンドが実行できる環境がなければ導入を行うことができない．

RTM を導入する場合であれば，公式サイトから指定されたファイルをダウンロードして開くだけでインストールは完了するのだが，ダウンロードしてきたファイルが多く，どのファイルを開くことで実行ができるのか，またどのファイルが何の役割を果たすのかがわからなくなってしまうことがある．

それに対して Squareknot は，JavaVM が導入されていれば，ソフトウェアをダウンロードするだけで，実行可能であるため，導入が容易であると言える．

柔　軟　性

Squareknot の一番の特徴がこの柔軟性であるが，コントローラのクラスが持つ操作を表すメソッドが upButton，downButton の２つ，アクチュエータのクラスが持つ動作を表すメソッドが gain，back の２つとする際，操作 upButton と，動作 gain，操作 downButton と，動作 back をマッピングしている図 7 のような記述から，操作 upButton と動作 back，操作 downButton と動作 gain をマッピングするという図 8 のような記述に変更するだけで，対応を容易に変更できる．

```
upButton : gain
downButton : back
```

図 7　設定ファイル例 1
Fig. 7　Configuration example1

```
upButton : back
upButton : gain
```

図 8　設定ファイル例 2
Fig. 8　Configuration example2

RTM においても LabView でのプログラミングとなっているため，柔軟に切り替えることが可能である．

また，ROS においても，ユーザは既に開発されている，ロボットへの指示に対してのコードを書き換えるだけで動作の変更が可能であるため，柔軟性に優れている．

拡　張　性

Squareknot では，ユーザが新たにクラスを生成し，操作対象を増やすにあたって，準備されている部分にそのクラスを配置するだけで追加が可能である．

RTM では，新たにモジュール群を組み合わせてコードを書き換える際，既存のものを利用して実行を行うことができず，新たなコードで書き換える必要がある．

ROS では，多くの機器のためのライブラリ等が既に多く提供されており，ユーザはそれらを利用することで容易に拡張ができるため，拡張性においては Squareknot より優れている．

可　読　性

Squareknot フレームワークを用いてプログラミングを行う際，各操作や動作の処理はメソッド分けされて実装を行う．よって操作や動作に関する処理を変更したい際，チェックすべき部分が明確になる．

RTM では，LabView によって記述されているため，それぞれの動作のつながりが視覚的に理解できる．そのため読みやすいコードとなっている．

ROS では，これらのように，動作は動作で記述する等といった決まりが無いため，読みづらいコードとなってしまう可能性がある．

問　題　点

現時点での Squareknot（version 0.19）では，以下のようなエラー処理が適切に行われていない．

- 指定したメソッドが無い場合の処理
- 指定したメソッドの記述が正しくない場合
- Bluetooth での接続エラーの際の処理

これらの部分を実装していく必要がある．

またこのフレームワークは http://rudds.kyoto-su.ac.jp/softwares/squareknot/でダウンロードが可能である．

今回 Squareknot は Bluetooth 対応機器での通信によるコントローラでのアクチュエータ動作を対象として制作した．

今後は，Bluetooth 通信をより簡単にするための仕組みを構築する予定である．現在同梱しているコントローラは Bluetooth 通信の中で HID プロファイル[⋆1]を利用することを予定している．またアクチュエータは SPP プロファイル[⋆2]を利用することを予定している．このように利用するプロファイルがある程度見当が付いているため，それらの仕組みを追加する予定である．

また，現段階では Squareknot フレームワークは Bluetooth 通信のみを対象としているが，今後，Bluetooth 以外の通信プロトコルによるコントローラとアクチュエータの接続

⋆1 Human Interface Device(HID) とは，キーボードやマウス等の入力機器のための Bluetooth プロファイル
⋆2 Serial Port Profile(SPP) とは，シリアルケーブル同様に通信を利用出来る Bluetooth プロファイル

Vol.2012-DPS-150 No.21
Vol.2012-CSEC-56 No.21
2012/2/29

が可能になるように機能拡張を行う．さらに，操作対象をロボットに限らず，遠隔のコンピュータなども操作できるように拡張する予定である．

参 考 文 献

1) Lego，LegoMindstorms，http://www.legoeducation.jp/mindstorms/
2) iRobot，Roomba http://www.irobot-jp.com/roomba/index.html
3) parrot，AR.Drone http://ardrone.parrot.com/parrot-ar-drone/jp/t
4) 任天堂，Wii http://www.nintendo.co.jp/wii/
5) SONY，PlayStation http://www.jp.playstation.com/
6) Microsoft，XBOX http://www.xbox.com/ja-JP
7) ROS.org，ROS，http://www.ros.org/wiki/ja.
8) OpenRTM，RT-Middleware，http://www.openrtm.org/
9) Apache，Apache Commons，http://commons.apache.org/
10) BlueCove.org，BlueCove，http://bluecove.org/
11) WiiRemoteJ，WiiRemoteJ，http://www.world-of-cha0s.hostrocket.com/WiiRemoteJ/
12) Lego，LEJOS，http://lejos.sourceforge.net/
13) YourKit，YourKit，http://www.yourkit.com/
14) KJhole.com，Kjell J Hole，http://www.kjhole.com/
15) 木村聡，Java フレームワーク開発入門，ソフトバンククリエイティブ (2010).
16) 栗田稔，数学ワンポイント双書 7「写像」，共立出版株式会社.