再利用性の高いヒューマノイドロボット制御用オープンソースソフトウェアの開発

○高橋直樹（千葉工大），林原靖男（千葉工大）

Development of a high compatible open source software for controlling humanoid robots

○Naoki TAKAHASHI and Yasuo HAYASHIBARA (Chiba Institute of Technology)

Abstract: We are developing a software that controls humanoid robots emphasizing. We are aiming to provide it as an open source software. The developed software is designed to be layered and modulized. Therefore, it is applicable to various hardware by replacing the module. Furthermore, many algorithms may also be applied. In this paper, we introduce the developed software, and confirm that it can control a real robot.

１．はじめに

本研究室では，ロボカップなどを対象としたヒューマノイドロボットの研究・開発を行っている [1] ．その中で，ヒューマノイドのハードウェアを公開して，データを自由に利用できるようにしている[2]．これにより，ハードウェアの設計にかかる負担を減らして，ヒューマノイドの研究を加速させることを意図している．一方でソフトウェアにおいても，歩行制御などを容易に検証できる環境があることが望まれる．入手性の良いヒューマノイドロボットで動作を確認できる公開ソフトウェアとしては，**Table 1** に示すようなものが挙げられる[3]～[6]．ただし，これらのソフトウェアは特定のハードウェアに向けて開発されていることが多く，さらに十分にモジュール化されていないために独自のアルゴリズムを検証しづらいなどの問題がある．ヒューマノイドのハードウェアは未だ研究・開発が盛んに行われており，さまざまなハードウェアに対応できる仕組みを備えていることが望まれると考える．

本研究では，ヒューマノイドロボットを制御するソフトウェアにおいて，再利用性などを重視したソフトウェアを開発して，オープンソースとして提供することを目的とする．

Table 1 Comparison of control software for humanoid robots

	Target H/W	Scalability
ROBOTIS[3][4]	DARwIn-OP	△
igus[5]	NimbRo-OP/2/2X	○
B-Human[6]	NAO	△

２．歩行制御のソフトウェアの構成

ヒューマノイドを歩行制御するソフトウェアの構成例を **Fig. 1** に示す．図に示すように，ソフトウェアは複数のモジュールから構成される．上位層はコマンドや状態を送受信する層であり，下位層はハードウェアを駆動する層となる．例えば，上位層から歩行のコマンドを受信すると，歩行パターンを生成し，算出された角度に従いモータを制御する．さらに，ロボットの状態を計測・推定し，他のモジュールに伝達する．これらの構成は，多くのヒューマノイド用のソフトウェアで採用されており，本システムもこれを参考に開発している．

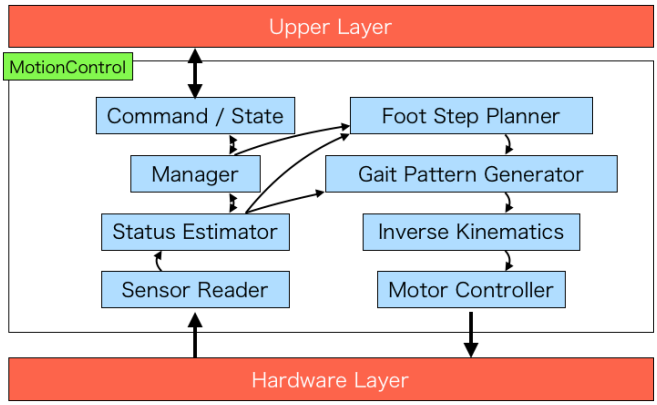

Fig. 1 An example of a software structure for controlling humanoid robots

３．ソフトウェア開発の指針

本研究ではハードウェアへの依存を低減して，再利用性の高いソフトウェアの開発を試みている．再利用性を高めるために，以下の指針に基づいてソフトウェアを開発した．

1) 機能ごとに階層化され，入れ替えの容易なソフトウェアの構造を採用する．

SY0010/19/0000 – 0659

2) デバイスを駆動する機能の多くをベースとなるクラスが提供することで，固有の処理を追加することにより利用できる環境を提供する．
3) マルチプラットフォームで動作するサンプルを提供する．
4) 実装方法に関するドキュメントを提供する．

また，ソフトウェア開発を支援する機能として，以下のような機能も提供する．

1) ソースコード追加時に Makefile を自動生成する機能
2) JSON ファイルによるデバイスの選択機能
3) デバイスの単体テスト機能
4) ダングリングポインタ防止機能

これらの機能は，追加のモジュールを開発するにあたり自動的に提供される機能であり，これにより開発の効率を上げることが可能と考えられる．

4．開発した制御ソフトウェアの構成

開発した制御ソフトウェアの構成を **Fig. 2** に示す．階層的に構成されており，ハードウェアやアルゴリズムの変更を容易にできるようにしている．各モジュールは異なるスレッドで実行され，計算機のリソースを効率よく利用するように配慮している．各階層の役割を以下に示す．

1) RobotStatus

モジュール間で共有するデータを管理する．例えば IMU の計測値やサーボモータの角度などは，この層に送られて保存される．保存されたデータは必要に応じて各モジュールに提供される．

2) CommandReader

上位層からのコマンドを受信して解釈する機能を提供する．モジュールとして独立しているため，モジュールを交換することにより，様々なプロトコルに対応可能な構造である．

3) StatusExporter

ロボットの状態を上位層に送信する機能を提供する．

4) FootStepPlanner

コマンドに従い，足を置く位置を決定する．

5) GaitPatternGenerator

足を置く位置に基づいて，歩行パターンを生成する．様々な制御の方式が提案されているため，それらを入れ替えて検証できる柔軟さが求められる．現在は，線形倒立振子モデルに基づく歩行素片を実装して検証している．

6) Kinematics

運動学・逆運動学を計算する機能を提供する．GaitPatternGenerator では多くの場合，足先の軌道が出力されるため，各関節の角度に変換することが必要となる．さまざまなハードウェアに柔軟に対応できるようにするため，モデルを URDF で記述して RBDL[7]で逆運動学を求めている．RBDL は，動力学を計算するライブラリであるが，将来的には動力学も計算結果を各モジュールに提供することを意図しているため採用している．

7) I/O

アクチュエータやセンサなどデバイスの入出力を制御する．具体的には，サーボモータとの通信や IMU の出力値の取得などを行う．

8) Tools

以上の歩行制御ソフトウェアの開発をサポートするため，ログを保存およびデータを加工するなどの機能を提供する．

なお，本ソフトウェアは現在も開発を進めており，随時拡張を行っているため，本構造が最終的な構造ではないことを述べておく．

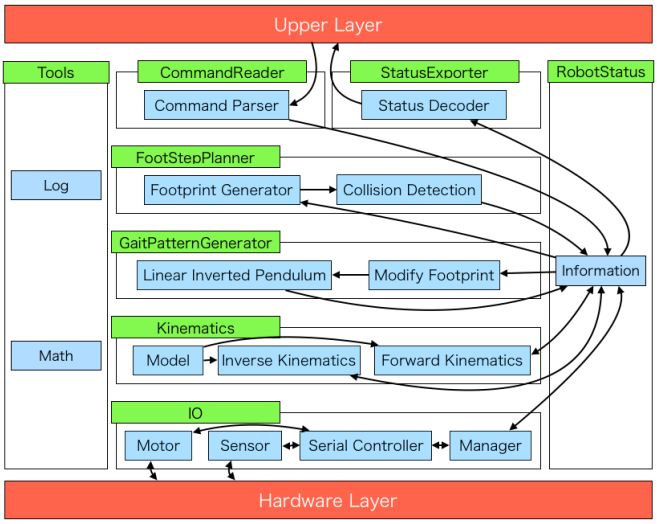

Fig. 2 The diagram of the developed control software

5．RobotStatus によるデータの管理

本システムでは，計算機のリソースを効率良く利用することを意図し，各モジュールを異なるスレッドで実行している．ただし，このようにマルチスレッドを採用したとしても，スレッド間でクリティカルセクションを共有した場合，排他制御により各スレッドの待ちが多くなり，シングルスレッドでの処理と差異がなくなるおそれがある．各スレッドの処理を調整し，クリティカルセクションへのアクセスを最小限にすることにより，各モジュールは並列に処理されるようになる．しかし，他のモジュールの挙動を考慮しながらコーディングしなければならないため，拡張性という観点ではあまり望ましい環境とはいえない．これを防ぐ方法としては，ROS のパブリッシュとサブスクライブの

ような仕組みを採用するという方法もあるが，オーバーヘッドが大きいため，10ms 程度の周期でフィードバック制御しなければならない歩行制御には適さないと考えられる．

そのため，本システムでは，各モジュールでデータを共有する仕組みとして，RobotStatus モジュールを提供している．**Fig. 3** に示すように，RobotStatus は，モジュール間で共有するデータを管理するモジュールであり，IMU やモータの角度などのセンサデータを受け取ると，可変の循環バッファにタイムスタンプとともに保存し，必要に応じてモジュールに提供する．センサデータ以外にも，歩行制御の計算結果など，モジュール間で共有するデータを保存する．これにより，他のモジュールの挙動を意識すること無く，必要なデータを参照し，モジュール固有の処理をプログラミングするだけで，アルゴリズムの検証が行えるようになる．**Fig. 4** にデータが入出力されるタイミングを示すが，各モジュールでは，処理の始めに RobotStatus モジュールからデータを受け取り，終わりにその結果を送る．RobotStatus モジュールはデータを受け取ると，各データに応じたバッファに保存し，他のモジュールからの要求に応じて，それを提供する．なお，循環バッファのサイズは自由に変更できるようになっており，過去のデータを参照することもできる．例えば，「1 秒前の IMU データを参照する」なども行えるようになっている．アルゴリズムによっては，過去のデータを要求するものがあるが，それに関しても標準で提供することができるようになっている．

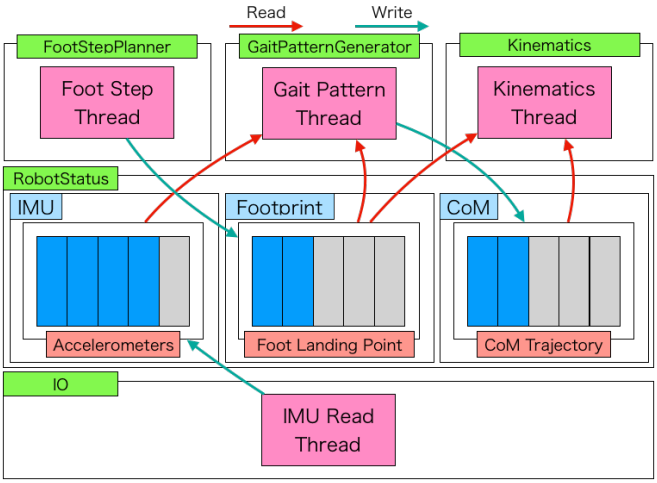

Fig. 3 Concept of RobotStatus module

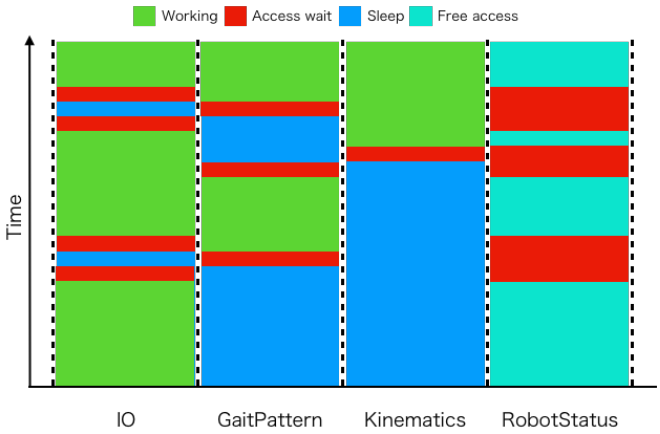

Fig. 4 An example of thread processing timing

6．実ロボットを用いた検証

これらのソフトウェアにより，実ロボットを歩行させることができるかを検証するために，ROBOTIS 社製 DARwIn-OP を使用した実験を行った．DARwIn-OP は標準で Intel 社製 Atom Z530 を搭載しているが，本実験では Ubuntu18.04 を使用するために Raspberry Pi に変更して実験を行った．**Fig. 5** に開発したソフトウェアを用いてロボットを歩行させたときの様子を示す．図に示すように，安定して歩行しており，本ソフトウェアにより実ロボットを歩行制御できることが確認された．

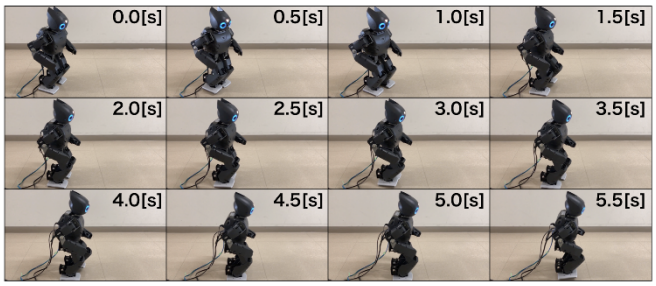

Fig. 5 Verification of propose method

7．今後の課題

今後，ROS で利用できるようにソフトウェアを拡張していく予定である．ROS のシミュレータ環境や，可視化ツールを活用することで，ソフトウェアの開発環境をさらに向上させていく予定である．

8．最後に

本稿では，ヒューマノイドロボットを歩行制御するソフトウェアに関して，再利用性などを重視したソフトウェアの指針を述べて，それに基づくソフトウェアを開発した結果に関して述べた．本プログラムは DARwIn-OP を歩行制御できることが確認されており，今後も開発を続けて行く予定である．なお，以下の URL において，本ソフトウェアを公開している．
https://github.com/NaokiTakahashi12/OpenHumanoidController

参考文献

[1] 林原靖男，南方英明，“RoboCup を通じた自律型サッカーヒューマノイドの研究・開発”，計測と制御（計測自動制御学会），Vol. 52, No.6, pp.487-494 (2013)

[2] 下吉拓明，林原靖男，“ロボカップ用オープンプラットフォームヒューマノイドロボットの開発　－第１報　メカニズムの設計と製作－”，日本機械学会ロボティクス・メカトロニクス講演会'17 予稿集, 2P1-J07 (2017)

[3] ROBOTIS-OP project https://sourceforge.net/projects/darwinop

[4] Inyong HA, et al., “Development of open humanoid platform DARwIn-OP”, SICE annual conference 2011, IEEE, pp. 2178-2181 (2011)

[5] igus Humanoid Open Platform ROS Software https://github.com/AIS-Bonn/humanoid_op_ros (2019)

[6] B-Human Code Release https://github.com/bhuman/BHumanCodeRelease (2019)

[7] Martin L. FELIS, “RBDL: an efficient rigid-body dynamics library using recursive algorithms”, Autonomous Robots, 41.2, pp. 495-511 (2017)