オブジェクト指向を用いた人間型ロボット制御用ドライバの作成 *

荒　牧　重　登**
松　尾　　英***
白　井　　亨****
長　井　達一郎**

Development of Object-oriented Control Drivers for Humanoid Robot

Shigeto ARAMAKI **, Akira MATSUO ***, Toru SHIRAI **** and Tatsuichiro NAGAI **

It has been recently requested to make a humanoid robot with high performance. A complex regulating system is necessary to make the robot high performance. When a complex system for the robot is developed, the drivers are requested that is used easily for programming. The existing drivers of the humanoid robot in our laboratory were inefficiency in the programming. We have improved the existing drivers by using object-oriented. We can develop an excellent system in recycling and in the readability of program source.

Key Words : Humanoid Robot, Driver, Object-oriented, Multi-modal

1. はじめに

最近では，高齢者介護用ロボットや商業施設の案内用ロボットなどが試験的に取り入れられるなど，本格実用に向けて研究が進められている．人間が環境の変化に対応できるのは五感によって外界の情報を収集・処理しているためであるが，現在のロボットが持つ外界認識機能はセンサ等が高価であることや技術的な問題から人間と比べて圧倒的に劣っている点が多い．高価であることは不可避なため，ロボットには少ない外界情報から環境を認識し，動作を決定する機能を実現しなければならない．だが，そのようなシステムは規模が大きく複雑になってくるため，研究・開発が難航しているのが実情である．また，外界認識ができ汎用的な動作が可能な機構のロボットが望まれている．そのような機構のロボットは一般的に多関節機構に限られてくるため，１つの部位の制御にも複雑な処理が必要になる．汎用動作には複数の部位間の協調制御が不可欠だが，複雑な処理同士を組み合わせることになるため，制御システムも複雑化してしまう．そのため，上述のような認識システムを構築するための基盤を整えるだけでも大きな障害となっている．

当研究室では，ロボットのマルチモーダルなインターフェースのための知識表現の研究 [１] [２] [３] を行ってきた．これら提案された技術を実際のロボットに適用し実証実験を行い，更なる改善が求められている．現在までに，複数の研究でいくつものシステムが構築されてきたが，ロボットの視覚 (カメラ)，首，腕等を協調制御させるシステムプログラムは複雑で再利用性と可読性に乏しかった．そこで，本論文ではより複雑なシステムの構築のために，まずロボットのドライバ群を改良し，再利用性と可読性の向上を目指す．

* 平成 22 年２月８日受付
** 電子情報工学科
*** 東芝情報システム株式会社
**** NEC 通信システム株式会社

1.1. ドライバ群の改良方針

本研究で作成するドライバの改良方針を述べる．

⑴ **ドライバ群のクラス化**：ロボットの既存のドライバ群は，パラメータの多い構造体を使用しており，ロボットと制御側 PC との送受信において非効率であった．そこで，この構造体をオブジェクト指向を用いてクラス化することで，構造体を意識せずプログラミングを可能にし，さらに送受信では変更パラメータのみ通信を行えるように改良を行う．

⑵ **各機能のモジュール化**：ドライバ群は視覚，首，腕の各モジュールに分ける．このように機能ごとに分離することによりシステム全体の複雑さを軽減する．

2. 人間型ロボットの構成

当研究室にある研究用ヒューマノイドロボットは，三井造船株式会社製のロボット ”MES-HR”[4] である．MES-HR には，各部位を駆動するための CPU が搭載されており，OS はリアルタイム OS の VxWorks(Wind River Systems, Inc.)[5][6] である．この VxWorks 上にロボットを駆動するドライバとドライバ群に指示を出す命令生成部 (PC 側) と通信するためのネットワーク通信プログラムが実行される．これらを駆動制御部と呼ぶ．

2.1. 基本構成

MES-HR の外観を図 1 に示す．MES-HR には人間における「目・首・腕・腰・足・触覚」に相当する機構として，それぞれ「カメラ・首・腕・腰・車輪・近接センサ・タッチセンサ」が実装されている．図 2 に内部の構成を示す．各部位にはそれぞれ独立したモータ制御用 CPU ボードが搭載されており，動作制御には外部からプログラムをダウンロードして実行する．この構成により各 CPU ボードが担当する区分が小さくなるため，各部位それぞれに生じる負荷を軽減できるという利点を持つ．また同時に，開発するプログラムはそれぞれの部位の制御に専念できるため，開発者側へ生じる負担も少ない．それぞれの CPU ボードは，MES-HR 内部の LAN で接続されており，TCP/IP や UDP/IP を用いた通信を行うこともできる．

2.2. VxWorks

MES-HR の各 CPU ボード上には，リアルタイム OS である VxWorks が搭載されている．この OS 上で制御用プログラムを実行することで MES-HR の操作を行うことができる．また，VxWorks 内の機能をプログラム内から呼び出すことで汎用性と安全性を確保した高度なシステムを構築することが可能である．以下に今回の研究に関連のある各機能の概要を示す．

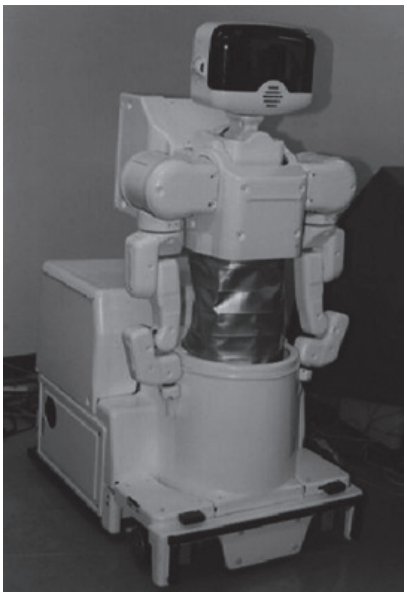


図 1 MES-HR

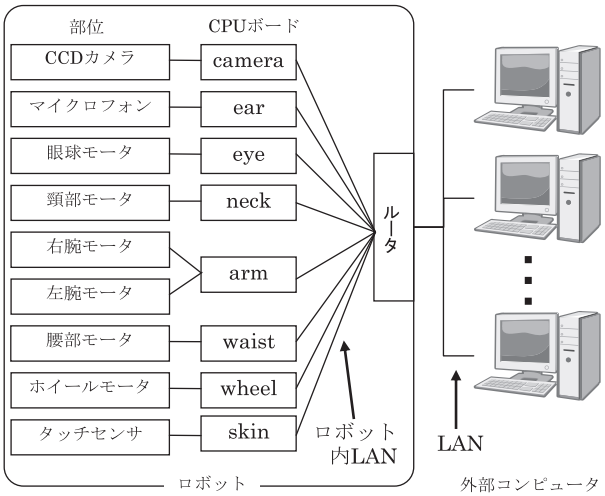


図 2 ロボットの制御ネットワーク構成

①マルチタスキング
優先度別のタスク生成によって複雑な並行処理を可能にする．

②プリエンプティブ優先度スケジューリング
タスクの優先度に応じて CPU を割り当てるため，優先度が高いタスクほど早く処理が終了する．

③タスク間通信
同期制御や排他制御にはセマフォ，値のやり取りにはメッセージキューが用意されている．

④ C/C++ 開発サポート
プログラムの開発には C/C++ 言語を利用することができる．

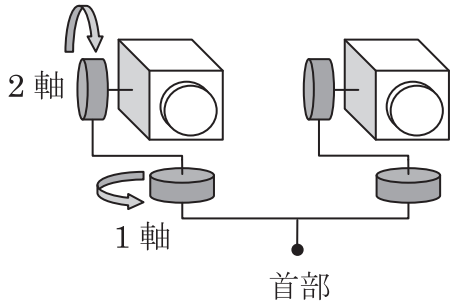


図 3 視覚部機構

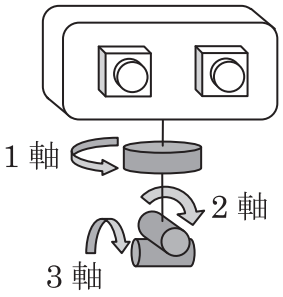


図 4 首部構成

2.3. 視覚部構成

視覚部には左右それぞれに CCD カメラが搭載されている．カメラの種類については以下の通りとなっている．

① 中心視野カメラ（左右 2 台）：SONY 製　FCB-IX47A
有効画素数　38 万画素　768(h)×494(v)
② 周辺視野カメラ（左右 2 台）：東芝製　IK-M43H
有効画素数　38 万画素　768(h)×494(v)

①はズームやオートフォーカス等多様な機能を備えており，汎用性に富んでいる．②は視野角が広く，一度に得られる映像範囲が広い．

注視制御によって広範な視野を確保できるため，①のカメラを用いる．カメラ映像は外部の PC へ映像ケーブルを直接接続し，そこで画像処理を行う．

視覚部の駆動は 2 軸構成になっており，1 軸が左右旋回，2 軸が上下傾斜になっている．図 3 に構成イメージを示す．

また，表 1 に駆動方向と範囲を示す．ハードリミット[☆1]，ソフトリミット[☆2] は原点位置を基点としたものであるため，原点の設定法によって変化する．尚，視覚部においては外装衝突が生じないため，内部に実装されている自動原点検出機能を用いている．

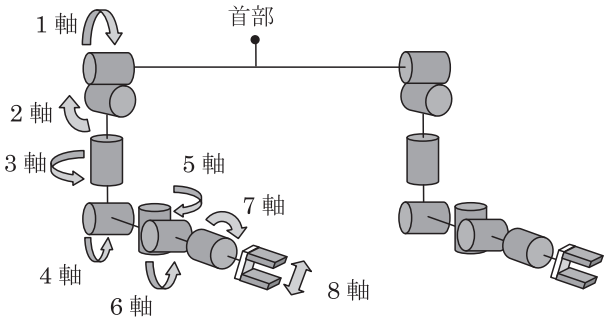


図 5 腕部構成

2.4. 首部構成

首部は 3 軸構成になっており，1 軸が左右旋回（Yaw），2 軸が左右傾斜（Roll），3 軸が前後傾斜（Pitch）を行う．図 4 に構成を示す．

また，表 2 に駆動方向と範囲を示す．ハードリミット，ソフトリミットは原点位置を基点としたものであ

表 1 視覚部の駆動範囲と方向

	駆動方向	範囲	ハードリミット	ソフトリミット	単位
1 軸	旋回	90	−45 ～ 45	−45 ～ 45	度
2 軸	上下	90	−45 ～ 45	−45 ～ 45	度

表 2 首部の駆動範囲と方向

	駆動方向	範囲	ハードリミット	ソフトリミット	単位
1 軸	旋回	330	−165 ～ 165	−160 ～ 160	度
2 軸	左右	92	−46 ～ 46	−43 ～ 43	度
3 軸	前後	102	−38 ～ 64	−35 ～ 61	度

☆1 機構的な可動限界
☆2 ソフトウェア上で設定している限界値

表 3 腕部の駆動範囲と方向

	駆動方向	範囲	ハードリミット	ソフトリミット	単位
1 軸	肩前後	234	−52 ～ 182	−50 ～ 180	度
2 軸	肩左右	127	−15 ～ 112	−13 ～ 110	度
3 軸	上腕捻り	184	−92 ～ 92	−90 ～ 90	度
4 軸	肘前後	131	−83 ～ 48	−80 ～ 45	度
5 軸	手首左右	122	−91 ～ 31	−90 ～ 30	度
6 軸	手首前後	186	−93 ～ 93	−90 ～ 90	度
7 軸	手首回転	326	−163 ～ 163	−160 ～ 160	度
8 軸	指開閉	60	0 ～ 60	0 ～ 60	mm

るため，原点の設定法によって変化する．尚，首部の 3 軸は三次元座標計測の正確性向上のため，マニュアルに記されているリミットとは異なる設定を行っている．

本研究では Yaw 軸，Pitch 軸だけでほとんどの範囲を捉えられるため，Roll 軸は使用していない．

2.5. 腕部構成

腕部は 8 軸構成になっており，1 軸が肩を中心とした前後傾斜，2 軸が肩を中心とした左右傾斜，3 軸が上腕部における捻り回転，4 軸が肘関節の前後傾斜，5 軸が手首の左右傾斜, 6 軸が手首の前後傾斜, 7 軸が手首回転, 8 軸が指の開閉である．構成イメージを図 5 に示す．

また，表 3 に駆動方向と範囲を示す．ハードリミット，ソフトリミットは原点位置を基点としたものであるため，原点の設定法によって変化する．

2.6. 機構上の問題点

腕部においては 7 軸–8 軸間干渉問題というものが存在する．これは，7 軸モータと 8 軸モータが部分的に連動していることによって生じている．そのため，7 軸回転時に生じる 8 軸への影響増分を補正する必要がある．本システム内で用いた式を以下の (1) に示す．

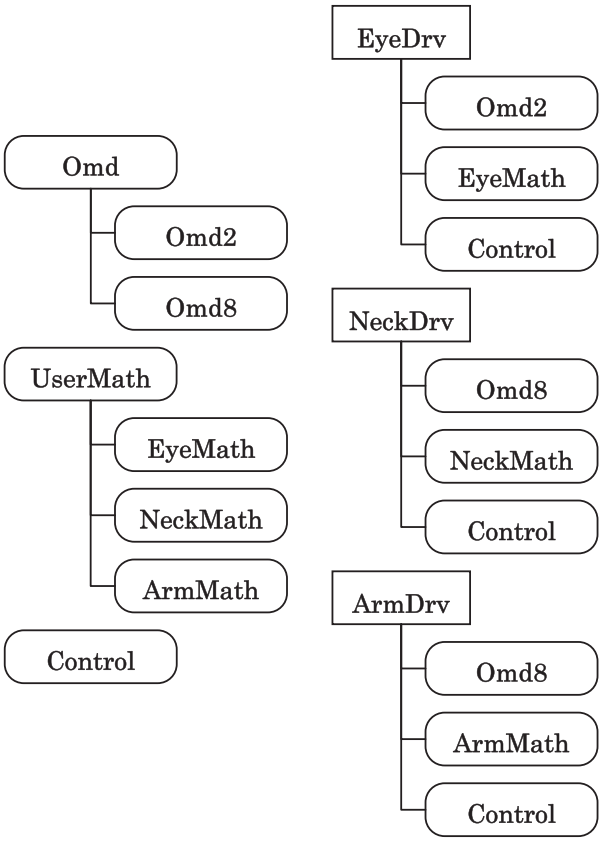
$$W8 = R \times \theta 7 \quad \cdots\cdots \quad (1)$$

θ 7 : 7 軸回転角(r)
W8 : 8 軸補正幅(mm)
R : 回転－直動変換係数(mm/r)

ただし，補正によって 8 軸が駆動限界を超えてしまうような場合は，7 軸がその位置で固定されてしまう．そのため，動作時には 8 軸の動作後状態を予測し，駆動限界を超えるようであれば影響増分の符号反転を行う必要がある．

3. ロボットドライバ

MES-HR は原点からの絶対位置をエンコーダによって指定することで制御する．このときのエンコーダとは各軸の回転量に相当するわけだが，人間が扱うのは容易でない．このようなハードウェアに依存した値を直接扱



(a) 継承関係　　(b) 所属関係

図 6 クラス構成

うようなプログラムは，開発時に大きな混乱を招く．そのため，人間が扱いやすい値（例えば角度など）に変換するプログラムを用いることで MES-HR の操作を簡略化する．

今回，既存ドライバの不備を解決するために，視覚・首・腕用ドライバを再構築した．以下に大まかな不備の修正点を記す．

① 処理とデータの一体化
　既存ドライバは構造体を引数とした関数で使用していたが，クラス化によって構造体を意識せず扱えるようにした．

② 抽象化
　関数名を抽象化し，内部がどのように動作しているかを利用側は気にしなくて良くなった．

③ 送信の一括化
　既存ドライバでは抽象化した場合の送信頻度が高く非効率的であった．新規ドライバでは変更点のみに焦点をあて，送信を管理することで効率性を高めた．

再構築の際にはオブジェクト指向を適用し，再利用性と可読性の向上を図るとともに利便性を追求した．ドライバ全体の大まかな構成を図 6 に示す．

以下，それぞれのクラスの機能について解説する．

3.1. 軸共通基底クラス：Omd（Object Motor Driver）

図 7 Omd クラス構成

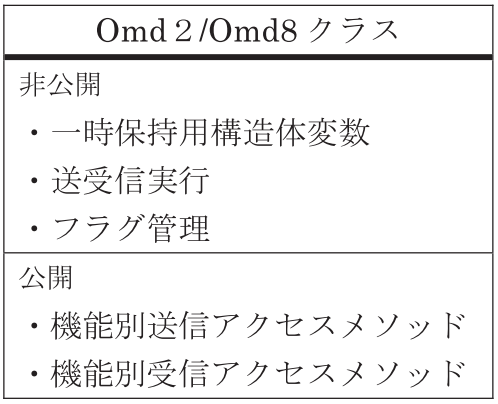


図 8 Omd2、Omd8 クラス構成

MES-HR とのグループ単位の送受信を行う機能を実装．このクラスではグループの意味に関係なく，ただ送受信することのみを行う．ここでの送受信とは特定メモリアドレスへの読み書きを指すため，非公開な値としてメモリアドレスを内部に持っている．図 7 にクラス構成を示す．

3.2. 特定軸用クラス：Omd2，Omd8

Omd クラスを継承し，2 軸機構用の Omd2，8 軸機構用の Omd8 となる．各種グループの意味づけを行い，またそれぞれの値ごとにアクセスメソッドを持つことで操作性を向上している．送受信そのものはグループ別変

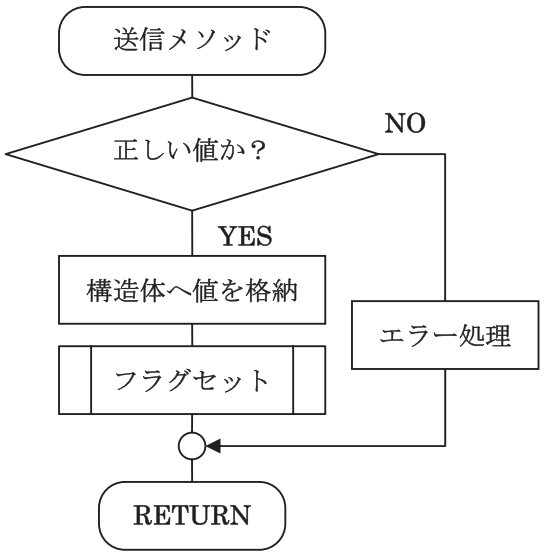


図 9 送信メソッド

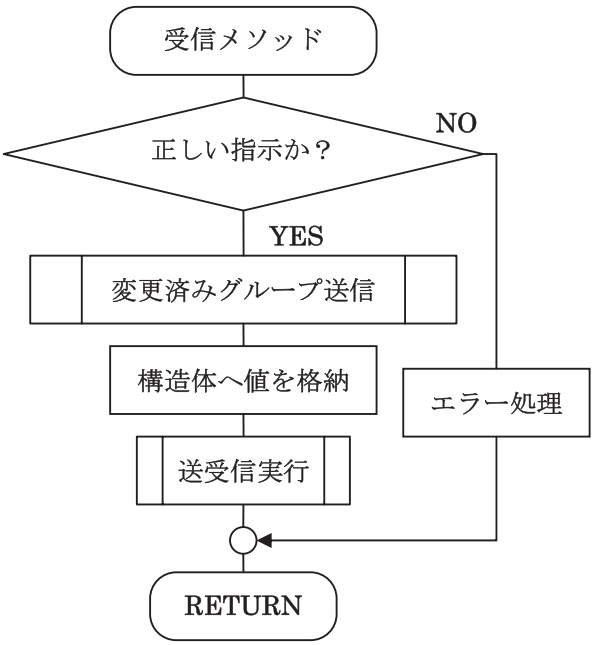


図 10 受信メソッド

更フラグにて管理するため，メソッド実行ごとの送受信実行を防ぎ，オーバヘッドの軽減を行った．図 8 に Omd2/Omd8 クラスの構成を示す．

機能別のアクセスメソッドを実行する際，内部ではその種類に応じてフラグ管理・送受信を行う．基本的に，係数設定等の送信メソッドはフラグ管理を実行する．位置指令系の送信メソッド及び受信メソッドは，フラグ状態に応じた送受信を一括して行った後に自身を実行することとなる．以下にそれぞれを説明する．

通常の送信メソッドでは正常値を受け取った後，一時格納用の構造体へ値を格納してフラグをセットする．このフラグは所属するグループに変更があったことを示すものであり，後に記述する一括送信の際に参照される．受け取った値が異常値の場合はエラー処理を行う．フローチャートを図 9 に示す．

受信メソッドもしくは位置指令系送信メソッドなどは状態の正確性が求められる．そのため，正しい指示であったときはまずフラグがセットされているグループの送信を順次行う．その後，構造体へ値を移して自身の属するグループの送受信を行うことでデータを取得する．フローチャートを図 10 に示す．

3.3. 数学計算クラス：UserMath，EyeMath，NeckMath，ArmMath

MES-HR の操作にはエンコーダ値を用いるなど，設計時には考慮しにくい値が多い．よって，このクラスでは各種機構別にエンコーダ⇔角度変換を行うようにしている．

UserMath クラス
非公開 ・円周率等グローバルな係数
公開 ・角度⇔ラジアン変換関数等

図 11 UserMath クラス構成

EyeMath/NeckMath/ArmMath クラス
非公開 ・ハードウェア依存係数 （減速比、駆動限界等）
公開 ・エンコーダ⇔角度変換関数 ・順運動学、逆運動学 （ArmMath のみ）

図 12 EyeMath、NeckMath、ArmMath クラス構成

UserMath クラスは全てに共通する数学的な処理を実装した．また，各部位専用数学計算クラスは，UserMath クラスを継承して部位特化を行っている．EyeMath は視覚部専用，NeckMath は首部専用，ArmMath は腕部専用である．これらはそれぞれの部位特有の計算を実装している．

UserMath クラス構成を図 11 に，EyeMath/NeckMath/ArmMath クラス構成を図 12 に，それぞれ示す．

EyeMath では軸のエンコーダと角度の対応の関係から，角度の算出は式 (2) を用いる．

$$Angle = Enc \div 100 \quad \cdots\cdots (2)$$

Angle：軸角度
Enc：エンコーダ値

同様に NeckMath と ArmMath では式 (3) によって角度を算出している．一部駆動軸では 3 次減速比が存在しないため，その場合は 1 を割り当てている．

$$Angle = \frac{Enc \times ER \times 360}{R_1 \times R_2 \times R_3 \times EP} \quad \cdots\cdots (3)$$

Angle：軸角度　　R_1：1 次減速比
Enc：エンコーダ値　　R_2：2 次減速比
ER：エンコーダ分周率　　R_3：3 次減速比
EP：エンコーダパルス数

尚，腕 8 軸においては指開閉に相当するため，角度指示ではない．そのためエンコーダ値から開閉幅を算出するための式 (4) を用いる．

$$W8 = \frac{Enc \times ER \times R8}{R_1 \times R_2 \times EP} \quad \cdots\cdots (4)$$

W8：8 軸開閉幅　　R_1：1 次減速比
Enc：エンコーダ値　　R_2：2 次減速比
ER：エンコーダ分周率　　R8：回転－直動変換係数
EP：エンコーダパルス数

また，動作指示の際は角度・幅で指定し，内部で逆変換を行って MES-HR へと送信している．

その他に，ArmMath では左右の腕における駆動方向の違いを吸収するために角度の符号反転によって左右対称に扱えるようになっている．

Control クラス
非公開 ・動作権セマフォ ・割り込み権セマフォ
公開 ・権限取得 ・権限返却 ・継続許可問い合わせ

図 13 Control クラス構成

3.4. 権限管理クラス：Control

MES-HR はシステム側から見ると共有リソースとも見て取れる．よって，マルチタスクによる処理を行う際には排他制御が必要となる．このクラスでは動作権限と割り込み権限の二つの権限を発行することで割り込み禁止型実行モードと割り込み許容型実行モードを実現している．尚，動作命令の実行は全て FIFO 形式となっている．Control クラス構成を図 12 に示す．

また，やや複雑な機能となる権限取得のフローチャートを図 14 に示す．

割り込み権は「最優先であるかどうか」を表し，動作権は「実行中かどうか」を表している．これらを VxWorks の機能であるセマフォを用いてそれぞれ実装することで，競合を回避しつつ FIFO 順実行を可能にしている．

EyeDrv/NeckDrv/ArmDrv クラス
非公開 ・Omd 系クラス ・Math 系クラス ・Control クラス
公開 ・原点検出/破棄関数 ・絶対指定動作関数 ・相対指定動作関数

図 15 EyeDrv、NeckDrv、ArmDrv クラス構成

3.5. 部位特化ドライバ：EyeDrv，NeckDrv，ArmDrv

前述のクラスを包含し，組み合わせることで利用頻度の高い定型処理を実装している．代表的なものとして原点検出，フィードバック制御等があるが，それらは次節以降で説明する．部位特化ドライバには、視覚部用の EyeDrv, 首部用の NeckDrv, 腕部用の ArmDrv があり，図 15 にクラス構成を示す．

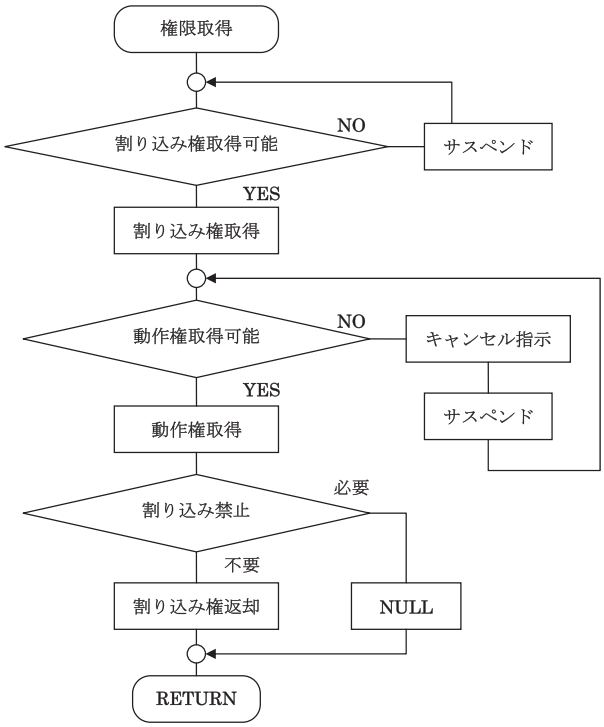


図 14 権限取得

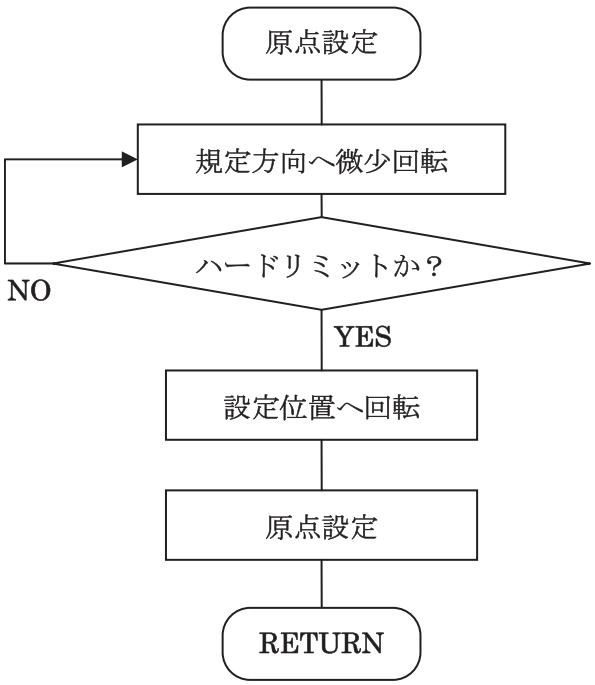


図 16 原点設定の流れ

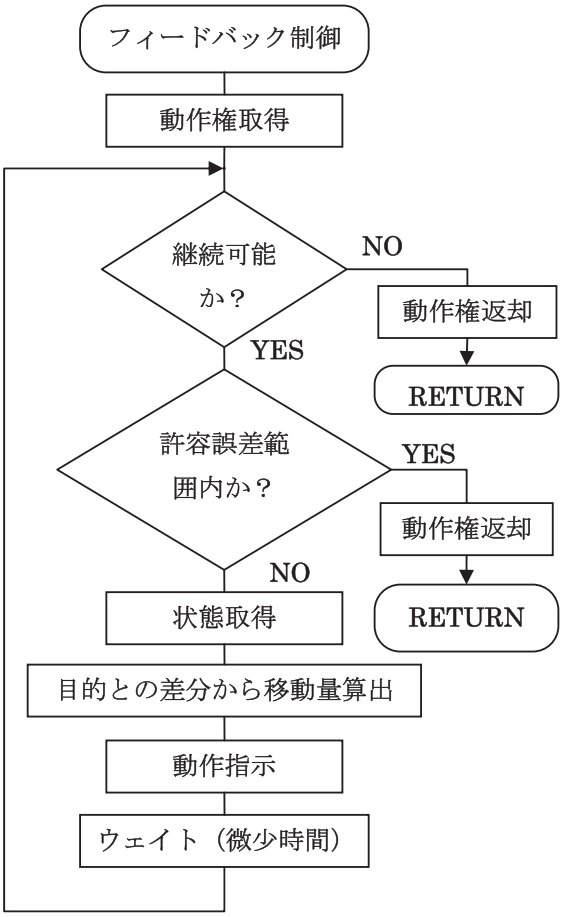


図 17 フィードバック制御のフローチャート

3.6. 原点検出

近年のロボットの多くは，あらゆる操作を行う基点となる原点情報を電源 ON/OFF に関係なく保持している．しかし，今回用いた MES-HR は電源 OFF 時に原点情報を保持することができないため，起動後に原点を認識させる必要がある．

原点の検出方法としては，ハードウェア内部に用意されている自動検出機能を用いる方法と現在状態を原点とする方法とがある．ただし，自動検出機能はサンプル程度のものであり，外装による衝突が考慮されていない．よって，今回は現在状態を原点とする方法とハードリミットを組み合わせて原点設定機能を作成した．大まかな流れを図 16 に示す．

原点設定は内部で原点位置の検出を行うと同時に軸の固定を行っている．固定を行うことで意図しない回転を防ぐことができる．これらを一括して行うことを「原点設定」と呼ぶ．また，検出と固定は任意に設定・解除することができる．

3.7. フィードバック制御

各部位動作制御時においては，移動後の状態と目的の状態とを照らし合わせて到着判定を行っている．到着していないと判断した場合は，さらに動作を行うことで誤差を極力減らしている．図 17 に処理の流れを示す．

フィードバック制御における一回のループ処理時間はウェイトを含めて約 0.1[sec] となっている．これは

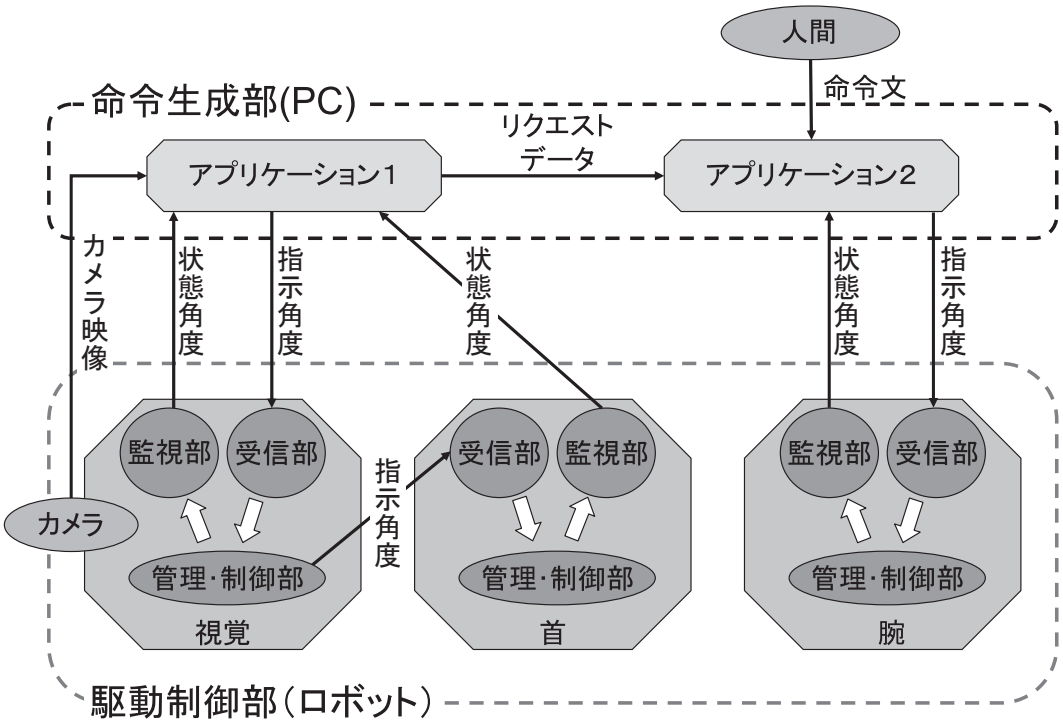


図 18 システム構成図（例）

CPU の性能と軸の駆動速度に関係し，それらを考慮して決定している．CPU の性能が向上した場合はウェイトを大きく，軸の駆動速度を上げた場合はウェイトを小さくするなど状況に応じた設定変更が必要である．

尚，フィードバック制御中はスレッドが拘束されてしまう．そのため，動作権限を利用することで他のスレッドからの停止，割り込みなどが可能となっている．

4. 駆動制御部の作成

4.1. システム構成

MES-HR を動かすためのシステムは，PC 側に用意される命令生成部と，ロボット側に用意されているドライバ群を含む駆動制御部（視覚：EyeModule，首：NeckModule，腕：ArmModule）で構成される．今回作成したのは駆動制御部である．命令生成部と駆動制御部間の通信関係，及びやり取りするデータの流れの例を図 18 に示す．

今回作成したドライバを含む駆動制御部は，命令生成部により各関節等の指示角度を受信しロボットを動作させ，また現在のロボットの状態を命令生成部に送信する．駆動制御部はロボットの各部位ごとのモジュールに分かれている．

各モジュール間の通信には高速性を優先するために UDP/IP 通信を利用している．それぞれのプログラムで出入りする情報を明確にし，責任の分解を行うことで移植性を高めている．そのため，一部のプログラムを差し替えても正確な動作が行われていればシステム全体としては変わらず稼動することが可能となっている．

4.2. モジュール

MES-HR の各部位ごとのドライバを含む制御プログラム（モジュール）を説明する．今回は視覚，首，腕のモジュールを作成した．

4.2.1. 構成

ドライバと通信機能を組み合わせ，視覚・首・腕ごとにそれぞれ EyeModule，NeckModule，ArmModule を作成した．共通する大まかな構成は図 19 のようになる．

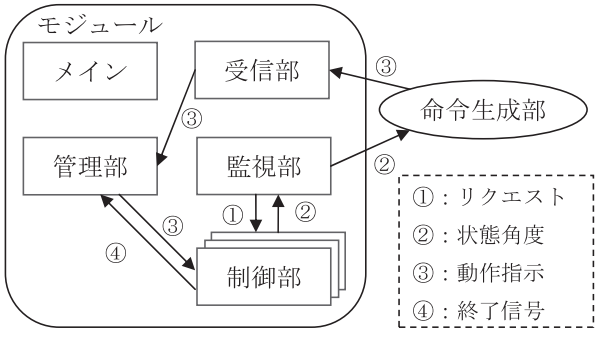


図 19 モジュール内部構成

モジュールはメイン，受信部，管理部，監視部，制御部と分かれ，マルチスレッドプログラミングによって構築している．以下，それぞれの説明を行う．

メイン

コンソール画面から利用する部分にあたる．実行開始時に全スレッドを生成し，メニューに応じて動作命令を管理部へ送る．

受信部

モジュール外部（本システムでは命令生成部）から命令を受信するスレッド．受信機能のみだが，この機能をスレッド化することで受信待ちの影響を外部へ及ぼさないようにしている．受け取った命令は管理部へと送られる．このとき，外部形式の並びになっている命令列を内部形式へと並び替える作業も同時に行っている．

制御部

スレッド間通信を利用し，FIFO 順に命令を実行するスレッド（スレッド間通信に関しては次節参照）．MES-HR の操作は，このスレッドでのみ行うようにすることで安全性を確保している．本スレッドは同時に複数生成されており，フィードバック制御によるスレッド拘束中であっても別スレッドが状態取得機能や割り込み機能を発揮することができる構成になっている．

管理部

送られてきた命令の実行タイミングを管理するスレッド．そのため管理部にて命令をストックし，タイミングを合わせて制御部に送信する．

監視部

MES-HR の状態監視を行うスレッド．微少時間毎に制御部へとリクエストを送り，受け取った値を外部が扱える形式に並べて送信する．

4.2.2. スレッド間通信

各スレッド間では VxWorks に備えられたメッセージキューを複数用いて通信を行っている．キューを通じたスレッド間通信の流れは図 20 のようになる．

キュー A はメイン・受信部から管理部への動作命令送信に用いる．キュー A には同時に 20 個まで命令をストックでき，管理部は実行可能になったタイミングでキュー A から命令を取り出し，キュー B へ入れる．キュー B に入っている命令は実行可能な制御部が先着順に取り出し，実行していく．動作命令は図 21 の形式のものを用いている．

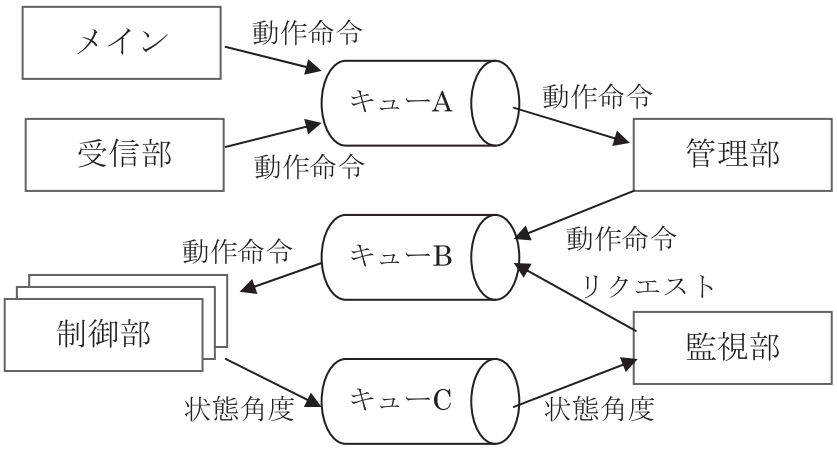


図 20 メッセージキューを利用したスレッド間通信

受信部にて外部から受け取った外部形式の命令も，管理部へ送る際に上記の形式へ置き換えられる．動作モードには実行する動作種類，軸角度指定には動作目標となる位置指令，制御キーには実行順を指定する．

制御キーの指定によって，

① 実行中の動作に割り込む

② 実行中の動作がある程度終わってから実行

③ 実行中の動作完了を待ってから実行

④ ストック分をキャンセル

⑤ ストック分と実行中の動作をキャンセル

の制御を行うことができる．

④と⑤はキュー A を一旦削除，作成することで実現している．①〜③は管理部にてウェイトを入れることで実現している．

4.2.3. 部位別処理

EyeModule に限定する機能として，動作分散が挙げられる．

EyeModule の受信部では，受信した動作命令の位置指令が一定値を超える場合に，位置指令の制限をかけている．これによって位置指令値が丸められ，カメラの駆動角度は 10° 〜 15° 前後となる．残った制限差分は NeckModule へ動作命令として送信されており，カメラの駆動だけでは捉えられない範囲を首部で補っている．これらの機能を MES-HR の駆動制御側で実装することで，命令生成部に生じる負荷を軽減することができる．簡単な処理の流れを図 22 に示す．

また，ArmModule から外部へ送信されている軸角度情報は，ArmDrv にて外部表現化された値となっている．

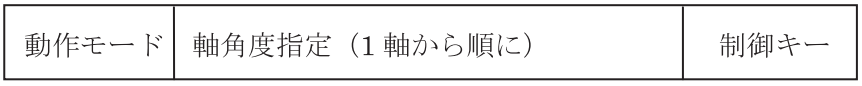


図 21 動作命令の形式

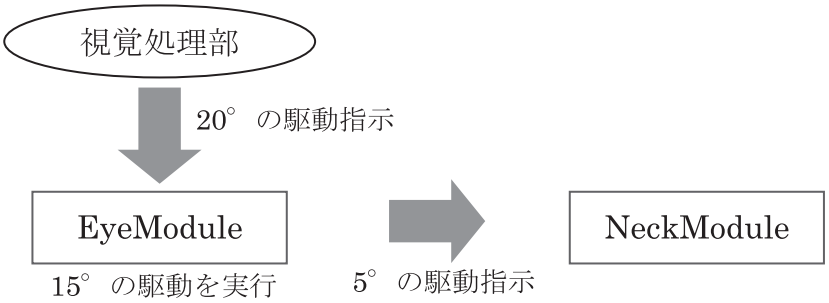


図 22 カメラと首の動作分散

5. 実験

作成した駆動制御部およびドライバ群が正常に動作するか検証を行った．

まず，命令生成部を作成し，その中の各アプリケーションから駆動制御部の各モジュールを呼び出すプログラムを作成した．既存のドライバ等を用いたプログラミングに比べ，抽象化されたドライバは非常に使いやすかった．さらに，原点の自動検出機能により毎回の原点設定のずれをかなりの精度で抑えることができた．

今回は，カメラからの情報を用いて目標位置まで腕を動かすプログラムを作成し，実験を行った．結果は通信速度や各関節の動き等が正常に行えることが確認できた．実験の様子を図 23 に示す。

6. おわりに

当研究室の人間型ロボットの既存のドライバ群を，オブジェクト指向を用いたドライバとそれらを含む駆動制御部モジュールとして改良を行った．さらに，駆動系のハードリミットを用いた原点の自動検出機能を構築したことで，毎回起動のたびに変わってしまう原点位置の正確さを向上した．また，一つの動作命令ごとに内部でフィードバック制御を行っているため，誤差を ±0.5° までに抑えることができている．しかし，今回は動作速度に関してはハードコーティングされているため，動的な変化に対応ができていない状態である．一方，システム構築にはオブジェクト指向を用いているため，改良・差し替えの容易性が以前よりも向上している．さらに，可読性が向上しているため，他人の作成したシステムでも，内容が理解しやすく，再利用性も向上したと考えられる．ドライバの追加・変更を行う場合は，ドライバクラスの仕様を変更するだけで済む．例えば，ドライバクラスにプレイバックの保存機能と再生機能を組み込み，対象ファイル名とサンプリング時間の指定のみで利用できるようにしてしまえば，実行プログラム側ではほんの数行でティーチング・プレイバックが利用可能になると考えられる．

本研究で作成したドライバは視覚，首，腕の三種類のみである．今後，より広範な作業・動作に対応するためには腰や車輪，タッチセンサを用いる必要があるため，それらの調整をしなければならない．各部位特化ドライバは，厳密な制御には対応していない．状況に合わせた各種ゲインなどの動的変更機能を加えることができれば，よりスムーズな動作が可能になると思われる．

図 23 実験の様子

参 考 文 献

[1] 弥吉孝太郎，荒牧重登，河村雅人，長井達一郎，鶴岡知昭：人間と人間型ロボットのマルチモーダルなインターフェースのための知識表現，平成 18 年度電気関係学会九州支部連合大会，p.192，2006.9.

[2] Shigeto Aramaki, Tatsuichirou Nagai, Masato Kawamura, Yasutaka Hatada, Tomoaki Tsuruoka, Human-Robot Interface by using frame like knowledge base, Proc. of IEEE Conference on Automation Science and Engineering (CASE 2007), 2007.9.

[3] Shigeto Aramaki, Tatsuichiro Nagai, Masato Kawamura, Koutarou Yayoshi, Yasutaka Hatada, Tomoaki Tsuruoka, A robot programming based on frame representation of knowledge, Proceedings of 7th IEEE International Conference on Computer and Information Technology (CIT 2007), pp.903-908, 2007.10.

[4] 三井造船株式会社，MES-HR マニュアル，1998

[5] Wind River Sytems, Inc.，VxWorks 5.4 プログラマーズガイド，1999

[6] Wind River Sytems, Inc.，VxWorks 5.4 リファレンスマニュアル，1999