博士論文

高速応答行動可能なロボットのための
体内分散通信制御系に関する研究

白井 拓磨

目 次

第 1 章 序論 9
1.1 本研究の背景 . . . 11
1.2 既存研究における課題と本研究の目的 . . . 12
1.3 本論文の構成 . . . 13

第 2 章 高速応答行動を可能とする体内分散通信制御系の設計要件と評価項目 15
2.1 はじめに . . . 17
2.2 ヒューマノイドロボットにおける体内制御システムの枠組み . . . 17
2.2.1 分散型制御システム . . . 18
2.2.2 実行系システムと認識・計画系システム . . . 19
2.2.3 体内システムと体外システム . . . 21
2.3 ヒューマノイドロボットにおける体内制御システム設計要件 . . . 22
2.3.1 安全規格と RAMS における設計要件分類 . . . 22
2.3.2 設計要件カテゴリ分類 . . . 23
2.3.3 評価項目 . . . 24
2.4 人体における高速応答行動の実現機構 . . . 26
2.4.1 人体の生体筋の特性とその利用 . . . 26
2.4.2 反射弓と相反性神経支配による高応答性神経系 . . . 28
2.5 分散型制御システムのためのノード間時刻同期 . . . 30
2.5.1 ノード間時刻同期 . . . 30
2.5.2 Network Time Protocol . . . 30
2.5.3 IEEE1588 . . . 30
2.5.4 EtherCAT での時刻同期 . . . 30
2.5.5 グローバル時刻同期 . . . 31
2.6 高速応答行動のための実時間システム . . . 31
2.6.1 実時間スケジューリング . . . 33
2.6.2 遅延の制御システムに対する影響 . . . 35
2.6.3 制御周期の制御性能に対する影響の評価 . . . 35
2.6.4 実時間性の制御性能に対する影響の評価 . . . 36
2.7 体内制御システムにおけるハードウェア制約 . . . 39
2.7.1 電力消費の改善 . . . 39
2.7.2 モジュール化 . . . 40
2.7.3 小パッケージ化 . . . 40
2.8 分散型制御システムのための通信系構成 . . . 41
2.8.1 ヒューマノイドロボット体内における通信系の制約 . . . 41
2.8.2 OSI 参照モデルと対比したシステム間通信モデル . . . 41
2.8.3 同期・非同期通信 . . . 43
2.8.4 接続トポロジ構成 . . . 44
2.8.5 実時間データフロー制御 . . . 44
2.8.6 通信品質とデータ符号化とデータレート . . . 45
2.9 高速応答を実現するための体内分散制御システム構成 . . . 47
2.9.1 演算負荷と応答性に応じたサブシステム分割 . . . 47
2.9.2 設計要件に基づいた分散ノード計算機アーキテクチャ . . . 48
2.9.3 体内分散制御システム構成の将来展望 . . . 50
2.10 本章のまとめ . . . 50

第 3 章 分散型センサを接続可能な実時間ベクトル制御モータドライバの開発 51
3.1 分散型制御システムにおける組み込みプロセッサ . . . 53
3.1.1 既存 CPU アーキテクチャでの制御器の実装 . . . 53
3.1.2 組込みプロセッサにおけるロボット制御演算性能 . . . 53
3.1.3 既存組込み用 CPU アーキテクチャにおける実時間プロセッシング . . . 54

3.2 Dependable Responsive Multi-Threaded Processor (D-RMTP) の概要 54
3.2.1 RMTPU における実時間プロセッシング 55
3.2.2 Responsive Task 56
3.2.3 20mmSiP を搭載した拡張コネクタ付き基板 56
3.3 RMTP-02D 基板の開発 58
3.3.1 ハードウェアロジックによる主要機能の実装 58
3.3.2 電流制御器とベクトル制御器 59
3.3.3 光伝送モジュール 59
3.3.4 マイコン用バスインターフェース 62
3.4 分散型センサデバイスインターフェース 62
3.4.1 衝撃センサ 63
3.4.2 6 軸力センサ 63
3.5 D-RMTP 及び RMTP-02D 基板による実時間モータ制御の実験 67
3.5.1 単軸トルクフィードバック制御による実時間モータ制御実験 69
3.5.2 Responsive Task による実時間モータ制御の性能検証 70
3.6 本章のまとめ 76

第 4 章 マルチプロトコル対応多ノード間低遅延体内分散通信系の開発 77
4.1 はじめに 79
4.2 従来ロボットにおける体内通信系の構成 79
4.2.1 HRP3L-JSK における体内通信系の構成 79
4.2.2 腱駆動ヒューマノイドにおける体内通信系の構成 79
4.2.3 その他のロボットにおける体内通信系の構成 81
4.2.4 既存通信規格と提案通信方式の比較 82
4.2.5 従来のロボット体内通信系における課題 84
4.3 低遅延体内分散通信系のための通信リンク設計 86
4.3.1 物理メディア層 86
4.3.2 物理層 87
4.3.3 データリンク層 88
4.3.4 プロトコル優先度に基づいた MAC 方式による低遅延通信 90
4.4 低遅延体内分散通信系のプロトコルブロック設計 96
4.4.1 プロトコルブロック基本構成 96
4.4.2 基本パケット構成 99
4.4.3 ルーティングテーブル初期化プロトコルブロック 99
4.4.4 グローバルクロック同期プロトコルブロック 100
4.4.5 Light RPC 型プロトコルブロック 105
4.4.6 分散共有メモリプロトコルブロック 107
4.4.7 コネクション型プロトコルブロック 111
4.5 低遅延体内分散通信系を利用したアプリケーション層設計 113
4.5.1 Light RPC 型プロトコルによるループバック遅延測定 113
4.5.2 パブリッシュ・サブスクライブモデル型通信 114
4.5.3 分散型共有メモリプロトコルによるトピック通信の検証 115
4.6 *Responsive Link* による分散制御実時間通信 121
4.6.1 冗長ネットワーク構成による耐障害性の検証 121
4.7 本章のまとめ 123

第 5 章 ヒューマノイドロボットにおける高速応答動作を実現する実行系システムの開発 125
5.1 はじめに 127
5.2 大出力モータドライバを持つ HRP3L-JSK と従来型上位システム構成 127
5.2.1 HRP3L-JSK 127
5.2.2 従来型上位システム構成 130

5.2.3 従来型システム構成における課題 . . . 131
5.3 即応的反射行動アーキテクチャ . . . 131
5.3.1 即応的反射行動 . . . 131
5.3.2 即応的反射行動アーキテクチャに基づいた高速応答行動 . . . 132
5.3.3 3 階層制御システム . . . 132
5.4 中間層制御システム . . . 135
5.4.1 中間層制御システムの透過性 . . . 135
5.4.2 中間層制御システム構成 . . . 136
5.4.3 ハードウェア構成 . . . 137
5.4.4 分散システム用ミドルウェアによるサブシステム間通信 . . . 138
5.4.5 EtherCAT による上位層との同期通信 . . . 139
5.4.6 EtherCAT、アクティブ光ファイバーリンクブリッジの構成 . . . 140
5.4.7 データロガー機能 . . . 141
5.4.8 上位システムバックアップ . . . 142
5.4.9 従来型システムとの比較 . . . 142
5.5 中間層制御システム構成モジュール . . . 144
5.5.1 サーボ制御モジュール . . . 144
5.5.2 一般化慣性行列計算モジュール . . . 145
5.5.3 参照トルク生成モジュール . . . 146
5.5.4 制御モードセレクタモジュール . . . 149
5.5.5 衝突検知モジュール . . . 149
5.5.6 モジュール群の実行時間計測 . . . 150
5.6 サブシステム間データ通信系の実時間性検証 . . . 150
5.6.1 OpenRTM データポート . . . 151
5.6.2 ROS トピック通信 (TCPROS) . . . 151
5.6.3 EtherCAT . . . 152
5.6.4 3 経路による目標関節角度指令値の計測結果 . . . 152
5.7 中間制御層の分散マルチノード構成 . . . 156
5.7.1 分散ノード群でのグローバルクロックの利用 . . . 156
5.7.2 分散マルチノード構成時のサブシステム間接続トポロジ構成 . . . 156
5.8 上位層制御システム . . . 158
5.9 実時間歩行軌道生成 . . . 158
5.9.1 倒立振子モデルベース重心軌道生成 . . . 159
5.9.2 足部面内 ZMP 修正許容領域に応じた目標 ZMP 軌道修正 . . . 161
5.9.3 多リンクモデルによる ZMP 誤差補償 . . . 161
5.9.4 遊脚接地高修正制御 . . . 163
5.10 絶対座標基準位置姿勢推定器 . . . 164
5.11 姿勢安定化制御 . . . 166
5.11.1 床反力フィードバック型姿勢安定化制御 . . . 167
5.11.2 トルクベーススタビライザ . . . 167
5.12 即応的反射行動制御系全体構成 . . . 169
5.13 脚型ロボットにおける体内制御システム構成例 . . . 170
5.13.1 ライントポロジ接続構成 . . . 170
5.13.2 リングトポロジ接続構成 . . . 172
5.13.3 ツリー(スター) トポロジ接続構成 . . . 172
5.14 本章のまとめ . . . 176

第 6 章 ヒューマノイドロボットにおける衝撃外乱に対する高速応答行動実験 177
6.1 中間層制御システムを利用した腕ロボットの動作実験 179
6.1.1 実験構成 179
6.1.2 スイング動作時の衝突実験 181
6.1.3 スイング動作実験における衝突検出の検証 181
6.1.4 ステップ動作実験 189
6.1.5 制御周期高速化に伴う衝撃応答性能への影響の検証 198
6.2 脚型ロボットによる身体運動計画系の実験 202
6.2.1 実験構成 202
6.2.2 歩行実験 204
6.2.3 物体踏破実験 208
6.3 脚型ロボットによる即応的反射行動の統合実験 212
6.3.1 トルク制御状態での立位バランス実験 212
6.3.2 フットステップ修正による不整路面歩行における転倒防止実験 216
6.3.3 遊脚外乱に対する緩和動作の定量的評価 217
6.3.4 足払い外乱に対する緩和動作と転倒防止動作実験 222
6.4 本章のまとめ 226

第 7 章 結論 227
7.1 本研究の結論 229
7.1.1 実時間プロセッサによる分散モータ制御モジュール 229
7.1.2 マルチプロトコル対応分散ノード間低遅延通信系 230
7.1.3 即応的反射行動アーキテクチャに基づいた実行系システム 231
7.1.4 本研究の意義 232
7.2 今後の展開 232

付 録 A ロボット諸元 235
A.1 A0-B spec 237
A.2 L1 240

付 録 B ロボット体内ノイズと通信系への影響 243
B.1 ロボット体内のノイズと通信システムへの影響 245
B.1.1 同相ノイズの差動信号への影響 246
B.1.2 フェライトコア・フェライトシースケーブルによる同相ノイズ除去 247

付 録 C 高速動作のためのセンサ信号処理 251
C.1 高速関節動作のための関節速度推定器 253
C.2 衝撃検出センサの実時間データ処理 260
C.2.1 DC 成分ウォッシュアウトフィルタ 260
C.2.2 衝撃加速度による制御パラメータの調整 260
C.3 関節トルク推定モジュールとトルク制御の定性的評価 260

謝辞 264

参考文献 268

発表文献 278

第1章

序論

1.1 本研究の背景

従来ロボットは、主に大規模な製造・組立ラインにおける自動化に利用するといった業種において利用され、生産性向上を発展をターゲットとして発展をとげてきた。しかし、ロボットシステムの発展に伴って自動化可能な作業内容が多種に渡るようになるにつれ、より広範な分野においてロボットを導入することで経済成長につなげようという機運が高まってきている。例えば、ロボットによるプラントやインフラの整備・点検の高度化、効率化を進めることや、高所作業、高電圧・高圧・高温環境での人手による作業の安全性の確保のためのアバターロボットシステムの開発が必要と考えられる。特に全国で橋梁等の老朽化が深刻化しており、早期に修繕箇所を発見及び、即時性が要求される災害現場での情報収取、探索、救助活動等に利用可能なロボットの開発は社会的な課題となっている。このようにこれまでは大手製造業の生産ラインという限定的な範囲のみで実用化されていたロボットが、いよいよ身近なものとして利用されることが社会的に期待されるようになってきた。

とりわけ 2014 年に「ロボット新戦略 [1]」として政府主導でロボット産業を成長戦略として採り上げ、「ロボット革命」のキーワードのもと従来の製造業におけるロボット利用に留まらず、製造・サービス・インフラ等の幅広い業種においてロボットの利活用を行うための提言がとりまとめられている。

ロボット革命では成長戦略として目標を以下のように掲げている ([1] からの引用)。

1. 世界のロボットイノベーション拠点– ロボット創出力の抜本的強化

   日本のロボットを徹底的に強化し、社会変革に繋がるロボットを次々と創出する拠点とするべく、産学官の連携やユーザーとメーカーのマッチング等の機会を増やしイノベーションを誘発させていく体制の構築や、人材育成、次世代技術開発、国際展開を見据えた規格化・標準化等を推進する。

2. 世界一のロボット利活用社会– ショーケース(ロボットがある日常の実現

   中堅・中小を含めたものづくり、サービス、介護・医療、インフラ・災害対応・建設、農業など幅広い分野で、真に使えるロボットを創り活かすために、ロボットの開発、導入を戦略的に進めるとともに、その前提となるロボットを活かすための環境整備を実施する。

3. 世界をリードするロボット新時代への戦略

   IoT の下でデジタルデータが高度に活用されるデータ駆動型社会においては、あらゆるモノがネットワークを介して結びつき、日常的にビッグデータが生み出される。さらにそのデータ自体が付加価値の源泉となる。こうした社会の到来によるロボット新時代を見据えた戦略を構築する。そのためには、ロボットが相互に接続しデータを自律的に蓄積・活用することを前提としたビジネスを推進するためのルールや国際標準の獲得を進めることが必要である。その際には、ロボット新時代の可能性を生かしていく上で基盤となるセキュリティや安全に係るルール・標準化などが不可欠である。

以上のような提言について、ロボットの開発者目線で捉えた場合、

1. ロボットハードウェア・ソフトウェアの標準規格化及び実用化を意識した手法・機構の研究開発を促進すること (1,2,3)

2. 幅広い分野で実用に耐えるために、より安全性・信頼性・利便性の高いロボットシステムを実現すること (2)

3. 従来の枠組みだけでなく、Internet of Things(IoT) に代表される高度なネットワーク・ビッグデータといった新たな社会基盤を前提としたロボット応用の提案すること (3)

が重要であると考えられる。

これまで等身大ヒューマノイドのようなロボットによる実環境における自律的な作業の実現を目指した研究の多くでは、産業用ロボットの技術をベースとしたロボットをプラットフォームに、認識・計画系のシステムを構築してきた。多くは基幹となる制御用の汎用計算機が体内にあり、拡張型のインターフェースカードを経由してアクチュエータやセンサと接続され, 認識のような多大な計算処理能力を要するタスクについてはロボット体外に設置された計算機を使用するリモートブレイン方式 [2][3] が採られてきた。このような構成はロボット開発を従来の計算機プログラム開発と同じ枠組みで行うことができ、また計算機資源

の制約を受けにくいため手軽に高い性能を引き出すことができる反面、安定したネットワーク接続・電力供給のためにロボットはケーブルによって主制御計算機や電源装置といった周辺設備に接続されていなければならなかった。つまり、ロボットが自由に行動可能な範囲は電源ケーブルが届き、なおかつ主制御計算機との間のネットワークが安定して提供される範囲に限定されるということであり、ロボットが自律的に行動しタスクをこなす上で大きな制約となっている。

ところが近年は組込みシステムの性能の向上やネットワーク技術・ネットワークサービスの品質向上に伴い、

- 小規模な組込みシステムでもマルチコア・マルチプロセスが利用可能となり、従来よりも計算処理能力・電力効率が向上した。
- 高精細な視覚センサによる周辺環境のリアルタイムな認識が可能な視覚モジュールが登場するようになった。
- 無線ネットワーク経由でクラウド上の膨大な量の知識・記憶（ビッグデータ）あるいはサービスへのアクセスが可能となった。

といった変化が生じており、ロボット体内の制御システムとクラウドコンピューティングによって従来の枠組みの延長上で開発を行いつつも、ロボットの自律的な行動が行いやすくなってきている。

以上のように、ロボットが実用的な水準で自律的に行動が可能な技術が社会的に要求されるという変化の中で、実際にロボットが従来のシステムを拡張する形で従来の制約を取り払った自律性を確保するための技術的な環境が整いつつある。

1.2 既存研究における課題と本研究の目的

これまで等身大ヒューマノイドロボットの実応用を目指す研究の中で、実環境下での継続的な作業を目指したロボット開発を行ってきたが、多岐に渡る要因によって作業の失敗や行動不能状態に陥るという致命的な問題が発生し、継続的な作業の実現には課題が多い状況である。

特に、実環境で動作する上で人や物との接触・衝突、ロボット自身の転倒といった怪我やダメージを与えてしまうことにつながりうる失敗は完全に回避されなければならない。また、転倒や衝突によるダメージが遠因となって、体内制御システムに致命的な障害が生じるといった例も数多くみられる。表 1.1 にこれまでに見られた主な致命的な障害の発生要因とその内容を示す。致命的な障害はハードウェア異常によって発生するケースが多く、また発生要因は衝撃・振動による機械的なダメージの蓄積に起因すると考えられる例が多数である。例えば熱による障害の発生も、遠因として衝撃・振動によるヒートシンクの取り付け不良、空冷ファンの破損等が考えられ、衝撃・振動を抑制することで、致命的障害につながる要因の除去ができると言える。

従って、人の行動と同様、事前に予期可能な事象に対して転倒や衝突といった失敗を回避するのは当然であるが、現実には予期不可能な事象に対しても失敗を回避、あるいは失敗の程度を極力小さくする行動がとられなければならない。そのためにも、体内制御システムは**現在の自機と周辺環境の状態をセンサからの感覚入力として知覚し、即座に動作として反映可能な頑健なアーキテクチャ**になっている必要があると考えられる。具体的には、ヒューマノイドロボットの身体を構成する関節機構の制御にしなやかさを持たせることで、外乱入力を吸収・緩和することを目指す。

また、失敗を完全に回避することは現実的には困難であると考えられ、場合によっては転倒等によるタスク失敗が発生することはあらかじめ想定されている必要がある。この場合に重要となるのは失敗から速やかに復帰し、作業を継続することが要求される。現在研究等で使用されるヒューマノイドロボットでは一度転倒すると、最悪の場合上述した障害要因によってシステムダウン等致命的な障害が発生し、失敗からの復帰が困難になるという問題が挙げられる。このような致命的障害の発生を抑制するための方策として、体内制御システムは制御則により失敗回避・抑制を行うだけでなく、転倒衝撃等に対する**機械的・電気的な頑健性・堅牢性**、あるいは**ソフトウェア的頑健性**を有する必要があると考える。

本研究では、これらの実作業継続を可能とするための要件を満たす体内制御システムの構成方法について述べ、行動実現について検証を行う。

表 1.1: Main fatal failure causes on a humanoid robot control system and their details.

	Fatal failure cause	details
1	Disability of Control raw and calculator	Fall down / Collision
2	Mechanical shock / vibration	Interface failure Mechanical damage Electrical short / Break of wire
3	Heat	Processor hung-up Break of ROM
4	Electrical short / Over voltage	Break of circuit Disorder of sensor output
5	Slidings / Repeated flexing	Break of wire / Voltage drop Wearing of sheathing
6	Electrical noise	Worth of sensor output S/N ratio Communication trouble
7	Voltage drop	System down Motion failure by loss of power

1.3 本論文の構成

本論文は全7 章から構成される．以下に各章の概要を述べる．

第1 章「 序論」では，本研究の背景と目的，及び本論文の構成について述べた．

第2 章「 高速応答行動を可能とする体内分散通信制御系の設計要件と評価項目」では，高速応答行動を実現する上での従来の体内システムの問題点について整理し、実時間性· 分散型構成をキーワードにロボット全身のアーキテクチャ設計論について述べる。

第3 章「 分散型センサを接続可能な実時間ベクトル制御モータドライバの開発」では，高速応答行動を実現する上で重要となる分散制御モジュールの設計及びアーキテクチャについて述べ、ノードでの実時間プロセスによるアクチュエータ制御について述べる。また、センサネットワークを構成する上で必要となる分散型センサの接続インターフェース構成についても述べる。

第4 章「 マルチプロトコル対応多ノード間低遅延体内分散通信系の開発」では，ヒューマノイドロボットの体内通信系として、同期型· 非同期型通信を柱とするマルチプロトコルに対応可能な通信リンクの実装方法、並びにハードウェアロジックによるプロトコル処理を実現するアーキテクチャ設計手法について論じる。また、実装した通信機構についてシミュレーション· 実験によってその性能を検証する。

第5 章「 ヒューマノイドロボットにおける高速応答動作を実現する実行系システムの開発」では，本研究が目的とする外乱に対する高速な応答を実現するための体内制御システム構成について述べる。反射的フィードバックループを構成するための即応的反射行動アーキテクチャを導入し、それを構成する中間層制御システムの構成方法· 実装について述べ、さらに上位層制御システムを含めた全体システム構成について論じる。また、オンラインでの動作修正としてオンライン歩行軌道生成を例に脚型ロボットのための上位層制御システムの構成について述べていく。

第6 章「 ヒューマノイドロボットにおける衝撃外乱に対する高速応答行動実験」では，本研究で提案する高速応答行動を実現する即応的反射行動アーキテクチャを有する体内制御システムについて、実ロボットで動作失敗につながりうる外乱入力に対してどのように振る舞うか実験· 検証を行う。

第7 章「 結論」では，本論文において得られた知見，主張点についてまとめ，最後に今後の課題，展望について述べていく．

第2章

高速応答行動を可能とする体内分散通信制御系の設計要件と評価項目

2.1 はじめに

ヒューマノイドロボットが作業継続が困難となる要因の一つとして、予期せぬ外乱入力によって計画された身体運動軌道から外れることで、作業中タスクの致命的な失敗につながるケースがある。本研究で言及する事前の予期が困難な外乱入力とは、例えば

- 不整地路面の移動によるつまずき、及び転倒
- 人とロボットの衝突
- ロボットのアームが物体と接触
- ハンドル操作中の急なキックバック
- 電動ドリル操作中の引っかかりによる急な反動
- ドア開平中に他の人もドア開閉を行う
- 双腕でのマニピュレーション中にかかる内力のバランシング

といった不意に起こり得る行動失敗ケースについてである。このようなケースの発生時に即座に回避行動をとり、復帰することで致命的な失敗を避けることが可能な制御システムの構成方法について論じるのが本論文の目的となる。

ただし、これらの行動失敗ケースについて全てを完全に防ぐことは人間でも失敗することがある点から考えて現実的ではない。万が一転倒等の失敗状態に陥ったとしても、速やかに障害から復帰して作業を継続できれば大抵のタスクにおいては十分と考えられる。失敗状態からの復帰における課題としては、転倒や衝突時に身体機構に物理的な破損が生じている場合や、制御システムが衝撃や熱· 振動によってダウンしてしまうケースである。

本研究で目指す高速応答行動とは、こうした即座の回避行動であったり、あるいは過大な負荷による破損、システムダウンを防ぐために、外乱入力に対して即応的に制御へ反映させることでその影響を軽減することを狙う手法である。上述の課題を踏まえて、本章ではヒューマノイドロボットが実環境下において継続的に作業を行うことを可能とするためには体内制御システムが如何なる機能を持つべきであるか、その構成方法について論じる。

2.3 節にて、本研究で構成する体内分散通信制御システムについて、その設計指針· 設計要件及び評価項目を与えるために、従来のヒューマノイドロボットの運用において見られた障害の発生要因について整理し、また既存の設計指針規格を参照し比較を行った。

2.4 節では、人体あるいは生体が不意の外乱に対して体のしなやかさ及び反射行動を利用して、衝撃を吸収することに着目し、ロボットの制御システムにおいてもその行動を実現するための機構構成について述べる。

さらに、体内分散通信システムの構成において、性能の決定因子として「 時刻同期性」「 実時間性」「 ハードウェア構成」、「 通信系構成」を挙げ、それぞれについて、2.5 節、2.6 節、2.7 節、2.8 節にて設計要件の定義を行った。通信系構成については、ヒューマノイドロボット制御システムにおいて必要となる通信モデルのメッセージング形態として同期形態と接続形態に着目した二種類の型があることを示し、さらに通信のハードウェアロジック化による実時間高速処理によってそれらを低遅延で実現することが可能となることについて論じていく。

2.2 ヒューマノイドロボットにおける体内制御システムの枠組み

ロボット制御システムの基本構成要素

本節では、ロボットの制御システムの基本的な構成要素について述べる。図 2.1 に示すのは、最も簡単なロボットの制御システムの構成で制御器とプラントから成っており、制御器はフィードバック入力 y を元に出力 u を決定する。実際には、図 2.2 に示すように制御器はプラントに対して作用するアクチュエータとプラントの状態を測定するセンサに各々接続されており、センサからの入力とアクチュエータへの出力を通してロボットの行動を起こし、ロボットの周辺環境に対して作用を行う。

連続･ 離散変換とネットワーク遅延の導入

図 2.2 では制御器とプラント中にあるアクチュエータ、センサとは直接接続されており、入出力 u,y はそのままの形でやり取りされているが、現実にはロボットの制御器は計算機システムで構成され離散システムとなっている。そのため、センサ情報は Analog to Digital Converter(ADC)、アクチュエータ指令は Digital to Analog Converter(DAC) によってアナログ･ デジタル変換が行われる (図 2.3)。これによって入出力 u,y は u_d,y_d と離散データとして制御器側では扱われる。

また、制御器･ アクチュエータ･ センサのそれぞれの間は情報伝達のためにネットワークを介して接続される。ネットワークを介することで、情報伝達には遅延 τ が生じるためセンサから制御器への入力値 y_d は $\hat{y}_d$、制御器からアクチュエータへの指令出力値 u は $\hat{u}$ と時間遅れが生じた値となって伝達される (図 2.4)。遅延にはネットワーク遅延の他、アナログ･ デジタル変換等のデータ処理による遅延も含まれる。

2.2.1 分散型制御システム

以上の構成モデルではコントローラは1 つとして扱われており、ロボットの各種アクチュエータ･ センサの情報を集約して管理を行う構成となっている。次に、コントローラを複数に分割しそれらをネットワーク

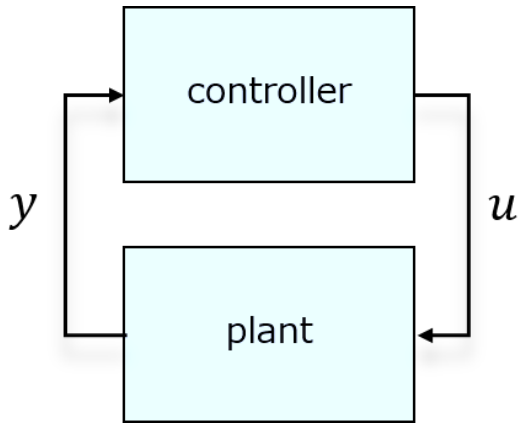


図 2.1: Simplest robot control model with controller and plant.

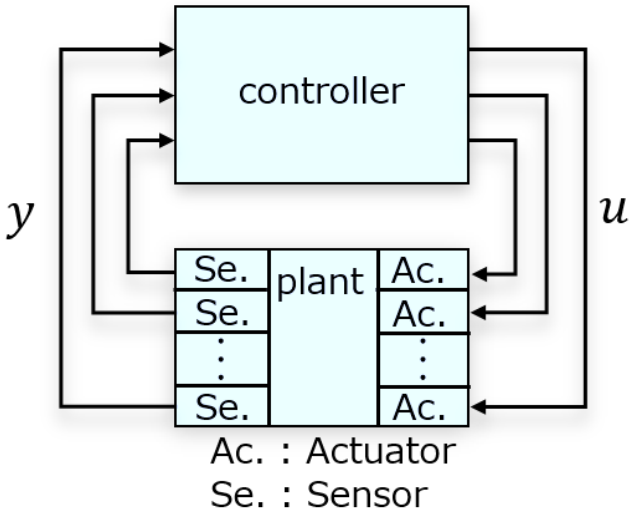


図 2.2: Robot control model with actuator and sensor.

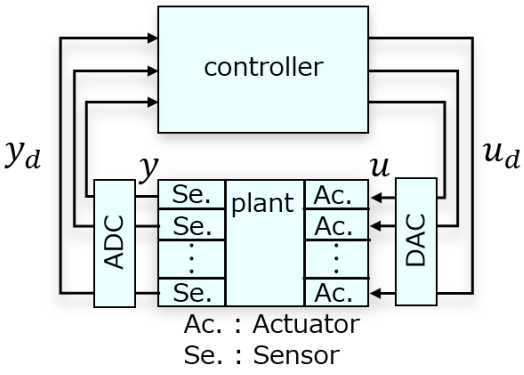


図 2.3: Robot control model with Analog Digital converter.

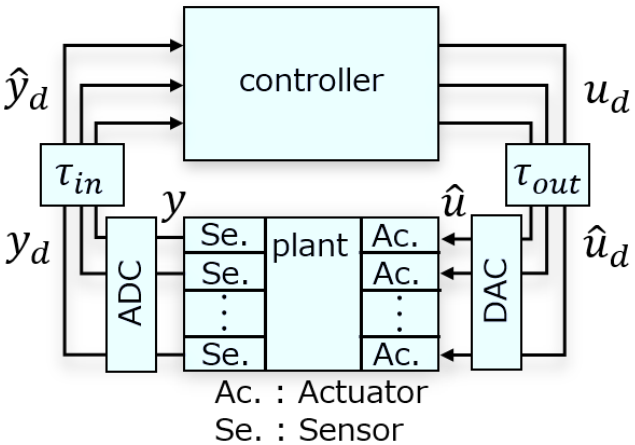


図 2.4: Robot control model with Network latency.

によって相互接続された**分散型制御システム**の構成について考える。図 2.5 に、主となる制御器及び、アクチュエータ・センサと対になった副制御器を含むノードがネットワークによって相互に接続されたロボットの分散型制御システムの構成を示す。

図 2.6 に、実際に分散型制御システムがロボットの体内システムとして構成されている例を示す。

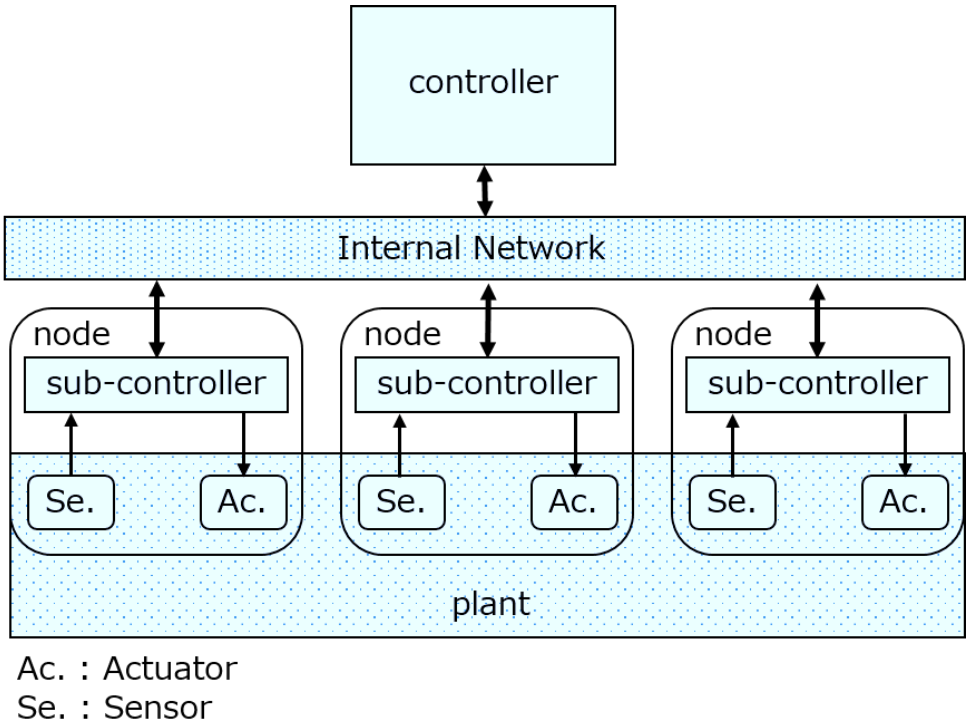


図 2.5: Distributed robot control model.

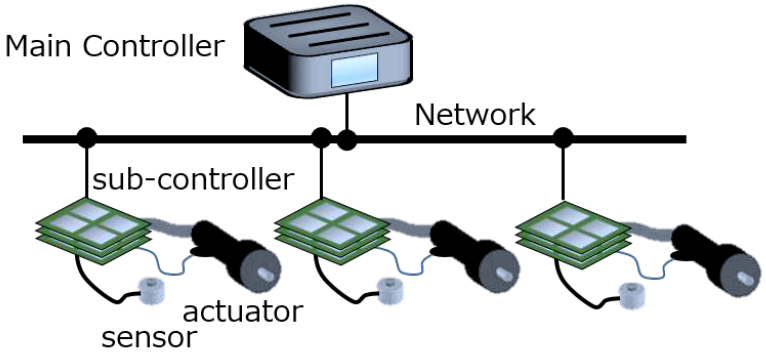


図 2.6: Example of distributed robot control model.

以上のようにロボット制御システムは、制御器・アクチュエータ・センサの3 要素によって構成され、連続・離散系の複合系であり、また入出力について遅延を考慮する必要があるモデルを対象としている。

次に制御器内部の制御システム構成について述べる。

2.2.2 実行系システムと認識・計画系システム

図 2.7 に示すようにロボットの制御システムでは、大きく分けて

- 認識・計画系

- 実行系

の2 つの処理に分類できる [4]。それぞれの処理について概要を述べる。

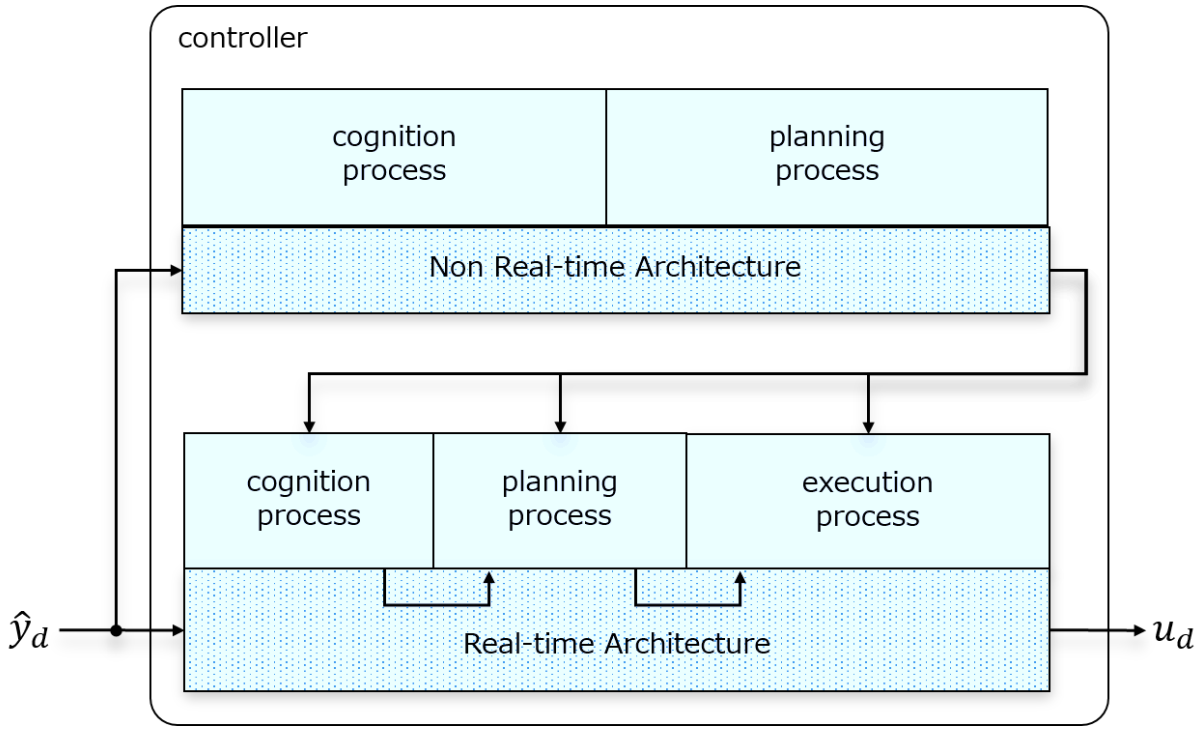


図 2.7: Architecture of processes for cognition and planning and for execution inside the robot controller.

実行系処理

実行系処理では、認識・計画系が生成した動作軌道を目標軌道としてそれに追従するようにロボットの全身のアクチュエータ制御を行う。

例えば歩行動作の場合、実行系は認識・計画系が生成した歩行軌道に追従するように、実際の関節軌道を生成してアクチュエータ制御を行う。

実行系はロボットの姿勢・運動状態・環境からの反力といったダイナミクスを直接扱うために、センサフィードバック情報を即座にアクチュエータ指令値に反映して出力できるよう極めて応答性の高いシステムとして動作しなければならない(実時間システム)。そのため、実行系は短い時間での周期タスクとして処理されるよう制御器内でも低遅延な処理が可能となるようなハードウェア・OS 上で構成されるのが一般的である。このような実時間システムについては、詳しくはこの後の 2.6 節で述べる。

実行系処理において扱う主なセンサデータ種類と、そのデータ量の実際に使用されている例を図 2.8 に示す。

認識・計画系処理

認識処理では、力センサや姿勢センサ・視覚センサの情報を元にロボット自身の状態あるいは周辺環境の現在の状態を認識する。計画処理では、認識された現在の状態を元にロボットがタスク遂行を行うための行動についてアクチュエータに対する動作軌道を生成し、発行を行う。また、現在状態だけでなく、記憶された過去の状態や知識に基づく将来の状態予測・推定も行うことで生成動作軌道の品質を高める。

例えば歩行動作の場合、認識系において歩行軌道上の障害物の認識、自己位置の推定等を行い、その結果に基づいて計画系が歩行軌道を生成する。

認識・計画系処理についても周辺状態を即座に認識して動作に反映させる実時間システムとして機能しなければならないが、実行系処理と比較して演算負荷の高い処理となっており、多少の応答の遅れが即座にタ

Sensor	Data Cycle(Hz)	Data Size / hour
Encoder	1000	7.2MB
ABS encoder	1000	7.2MB
Velocity	1000	7.2MB
Error	1000	7.2MB
Torque	1000	7.2MB
Temperature	100	0.72MB
Iq	100	0.72MB
Id	100	0.72MB
Servo Voltage	100	0.72MB
Board Voltage	100	0.72MB
Output	100	0.72MB
Output Limit	100	0.72MB
Estimated temperature	100	0.72MB
Link error count	100	0.72MB
Hale sensor status	100	0.72MB
Control status	1000	7.2MB

Command	Data Cycle(Hz)	Data Size / hour
Target angle	1000	7.2MB
Target velocity	1000	7.2MB
Target torque	1000	7.2MB
P gain	100	0.72MB
D gain	100	0.72MB
Control Mode	100	0.72MB
CC P gain	100	0.72MB
Current limit	100	0.72MB
Temperature limit	100	0.72MB
Stiffness	100	0.72MB

Sensor	Data Cycle(Hz)	Data Size / hour
Force Sensor *4	1000	172.8MB
IMU*1	1000	62MB

図 2.8: Data sizes related with execution system.

スク実行の失敗につながるわけではない場合には、実時間性よりも演算速度・演算精度に重点を置いたハードウェア・ＯＳ上で構成される。当然、応答の遅れが許されない認識処理・計画処理については、実時間性を重視したアーキテクチャ上に構成されることが望ましいため、演算性能重視のアーキテクチャと実時間性重視のアーキテクチャの2 層構成で認識・計画系処理を実装することも考えられる。

認識・計画系処理において扱う主なセンサデータ種類と、そのデータ量の実際に使用されている例を図 2.9 に示す。

Sensor	Capture Data	Data Cycle(Hz)	Data Size / hour
Tactile	3D pressure 10 ~ 1000 points	100~1000	20MB ~ 2GB ~ 20GB
Vision	stereo VGA 640 * 580 * 2	60fps~120fps	457GB ~ 914GB
Laser Range	depth image	7.5fps	600 MB
Multisense	Stereo 2048 * 1088	30fps	675 GB
Microphone	Sournd 32Ch	*	13 GB
Point cloud	3D point data with color 1 million point ~ 100 million point	*	15GB ~ 1500GB / Room

図 2.9: Data sizes related with cognition and planning system.

2.2.3 体内システムと体外システム

従来のヒューマノイドロボットでは、実行系処理と低レイヤの認識・計画系はロボット体内の主となる汎用計算機において目標関節角軌道列の生成・発行までを行い、さらに低レイヤにあるアクチュエータ・センサの制御を行うモータ制御システム層において、上層から発行された関節角指令値に追従するようフィードバック制御を行う。また、モータ制御システム層はセンサ情報を制御周期ごとに主制御計算機に送信する。

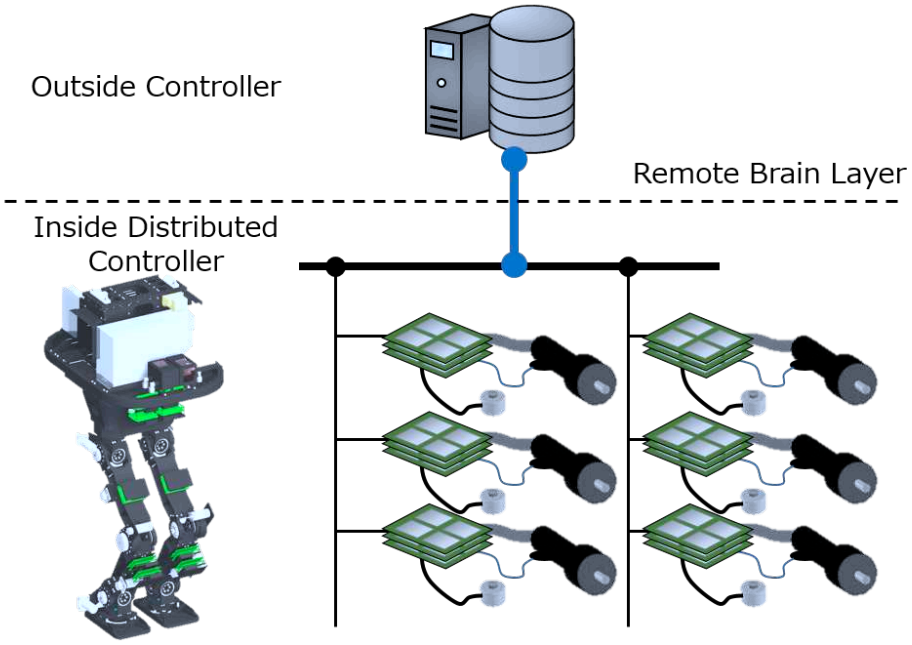


図 2.10: Remote brain system by outside computer.

ヒューマノイドのように関節自由度が多く、歩行によって移動ができる状態だと、同じタスクを解くための行動においても解が無数に存在し計算負荷は非常に大きくなる。これを短時間で演算するには高い性能を持つ計算機が必要となるが、これらは小さいロボットの体内に搭載できるほどの規模ではなかった。

そのため、体外に設置された大規模計算機と体内の小規模計算機でシステムを分割するリモートブレイン方式 [2][3] が採られてきた (図 2.10)。

近年は、体内に収められる規模の計算機でも身体制御を行うのに十分な演算能力が得られるようになってきたが、認識・計画処理を行うにはまだ不十分な性能であるため、簡単な動作については体内システムで自律的に行いつつ、全体の動作計画については引き続きリモートブレインで行われている (図 2.11)。

2.3 ヒューマノイドロボットにおける体内制御システム設計要件

本節では、制御システムの設計において指標となる設計要件をまとめ、以降の章において提案するシステムの評価指標として利用する。本研究の目的となる、実環境下において継続的な動作を可能にする制御システムプラットフォームを実現するためには、システム設計要件に実環境を想定した項目を挙げる必要があり、研究の枠を越えて製品開発レベルにおいても要求されるものと考える。

2.3.1 安全規格と RAMS における設計要件分類

鉄道システム分野におけるシステムの開発・評価の方針を定めた規格として RAMS 規格 (IEC62278[1][5]) が活用されている。RAMS 規格は、信頼性 (Reliability)、可用性 (Availability)、保守性 (Maintainability)、安全性 (Safety) の頭文字を冠した規格であり、安全性についての規定のみならず、実運用面におけるシステム開発・評価の指針を示している点で他の安全規格と一線を画す規格となっている。信頼性はシステムの故障率、障害発生頻度等で評価される指標であり、可用性(アベイラビリティ) は、システムの稼働時間で評価される指標となる。鉄道システムにおいては、システムの障害や保守によってダウンする時間が長くなるほど運航に支障をきたす範囲が広くなるため重要視されている項目となっている。保守性は、システムダウンに要する時間やコストによって評価される指標である。RAMS 規格における安全性に関しては、リス

[1]https://webstore.iec.ch/publication/6747

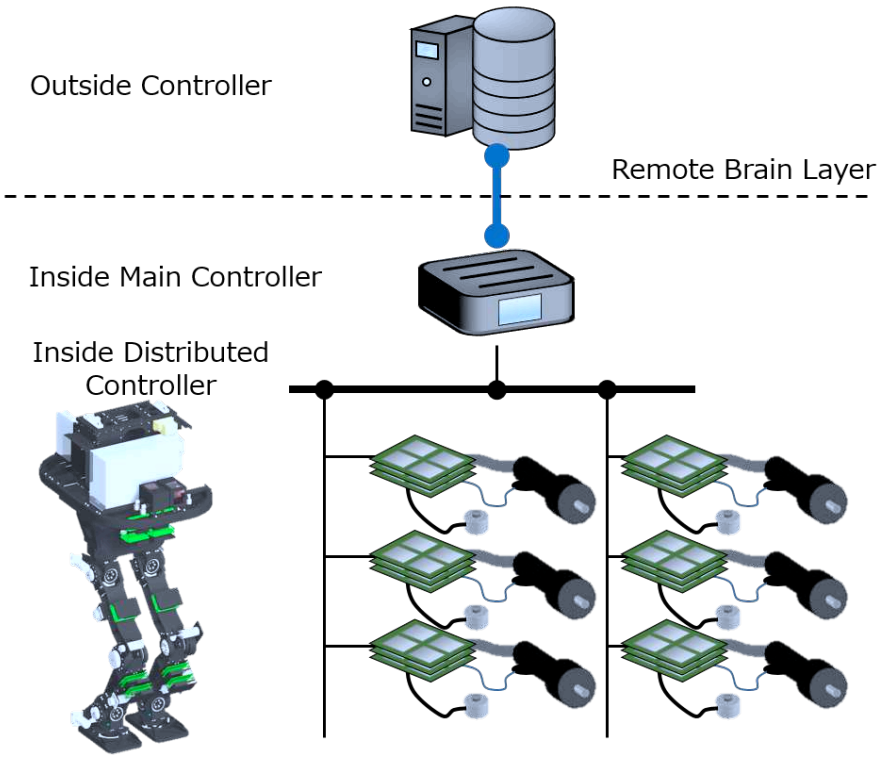


図 2.11: Body control system by using inside computer, and outside computer.

クの検証方法や設計から運用に至るまでの実行内容· 検証方法について規定されており、体系的に安全性が保障される内容となっている。

上記は全て鉄道システムを念頭に置いた規格であり、細部においてヒューマノイドロボットの設計要件に落とし込むには不適当な項目（保守によるダウン時間等）が存在しており、また、ヒューマノイドロボット固有の課題について設計要件に加えることが望ましいと考えられる。安全性に関しては、ロボット分野においても特に産業用カテゴリにおいて国際規格 ISO10218 やさらに協業ロボットでは ISO/TS15066[2][6] にて規定が為されている。従来は機械的リミットやアクチュエータの出力の上限によって安全性を担保する方法が主であったが、力制御や周辺状況認識· 緊急停止による安全策が施されている場合に、人間が近くに存在する場合でも作業を認められるようになっている。ただし、具体的な方策や評価指標について事細かに規定されているわけではないため、開発者側で設計要件に落とし込む必要がある。また、あくまでも産業用ロボットを念頭に置いているため、自立移動が可能で体内システムで完結させる必要性のあるヒューマノイドロボットにおいては、他にも検討される項目が存在すると考えられる。

2.3.2 設計要件カテゴリ 分類

前節で述べた鉄道システムにおける RAMS 規格、並びにこれまで培われてきたヒューマノイドロボットの運用実績を参考に、ヒューマノイドロボットのための本研究における独自の設計要件を用途によって大まかに以下の5 つのカテゴリ分けを行うこととした。

- 性能
  構成されたシステムによって達成されるロボットの動作が要求される品質を保有することを示す。
- 信頼性
  システムが不具合なく稼働されることを示す。
- 設計/運用
  システムの開発時や運用において問題が生じないことを示す。

[2]https://www.iso.org/standard/62996.html

カテゴリ	評価項目	要件	番号
性能	制御性能 (performance)	演算能力、実時間性、応答性が優れている	①
	連続稼働時間 (uptime)	低消費電力、高放熱能力で長時間稼働できる	②
	安全性 (safety)	異常時に即座に停止または正常状態に復帰する	③
信頼性	頑健性 (robustness)	耐熱、耐振動、耐衝撃、耐障害、耐ノイズ	④
	耐久性/堅牢性 (durability)	挿抜回数、繰り返し曲げ回数、強度、故障頻度	⑤
設計/運用	パッケージサイズ (size)	小サイズ、軽量、レイアウト自由度	⑥
	入手性 (availability)	コスト・納期・ロット数・販売期間	⑦
	保守性 (maintainability)	不具合を適切に修正することができる	⑧
	容易性 (facility)	少工数でシステムを構築できる	⑨
展開	継続性 (continuity)	更新時に機能を完全に引継ぎできる	⑩
	拡張性 (extensibility)	システム規模の拡大への対応	⑪
	柔軟性 (flexibility)	システム構成の変更自由度	⑫
	互換性 (compatibility)	更新・未更新部が混在しても機能を保てる	⑬
テスト	可観測性 (observability)	動作状態を観測・記録できる	⑭
	検証可能性 (testability)	仕様通りに機能しているか確認できる	⑮

図 2.12: Design requirements for robot control system.

- 展開
  システムを他のロボットへと適用範囲を拡大する際に問題が生じないことを示す。
- テスト
  システムに実装した機能について、正しく動作していることを検証することが可能であることを示す。

2.3.3 評価項目

設計要件として設定した各カテゴリごとに評価項目について説明を述べていく。一覧として図 2.12 に表形式でまとめている。

性能カテゴリについては「制御性能」「連続稼働時間」「安全性」を項目に挙げている。「制御性能」は演算能力、実時間性、応答性といった、要求される制御機能を実現するために不可欠な項目であり、構成されるＣＰＵの処理能力によっておおまかに決定される。分散型制御系においては複数あるノード群が全体としてどのような制御演算を達成することが可能であるかによって評価されると言える。「連続稼働時間」は、ロボットがどれだけの時間単独で要求される作業を実行可能であるかを示し、バッテリーサイズやＣＰＵ群の消費電力によって決定される。また、熱によるシステムダウンしないために放熱能力が消費電力を上回っている、耐熱性能が高いことを要求する項目である。「安全性」については、システムに異常が発生したことを検知することが可能である能力、異常が検知された場合に即座に復帰することが可能な能力あるいは故障を防ぐように停止することが可能な能力を有していることを示す項目である。

信頼性カテゴリについては「頑健性」「耐久性/堅牢性」を挙げている。「頑健性」は衝撃や振動、熱、電磁ノイズに対して、物理的にシステムが耐えられることを示す項目である。また、「耐久性/堅牢性」の項目についてはコネクタの挿抜回数やケーブルの繰り返し曲げ、緩みといった経年劣化によって不具合が生じ

る部位について、どの程度の耐用年数を期待できるかを示す項目である。耐用年数が極端に低い場合には、部品点数の多いロボットにおいては、慢性的に不具合が生じる原因となる。

設計/運用カテゴリについては「パッケージサイズ」「入手性」「保守性」「容易性」を挙げている。「パッケージサイズ」はシステムを構成する上で必要となる部品の空間占有体積や重量を示す。当然、パッケージサイズは小さく、軽量である方がロボットの設計自由度が増大するため有利となる。また、同じ空間占有体積であったとしても、レイアウト自由度がある場合にはトータルのパッケージサイズや設計の容易性が向上するため有利となる。「入手性」は、システムを構成する部品が低コストであるか、納期やロット数において問題がないかを示す項目である。特殊なインターフェース部品等を用いる場合、それらを採用できる計算機の構成は限られてしまうため、必然的にコストや納期が増大してしまう。そのため、汎用的で選択肢の多いシステム構成となるよう設計されていることが望ましい。「保守性」は、システムに不具合が発見された場合に、それが容易に解消することが可能であることを示す項目である。不具合の原因が特定が困難であるシステムや不具合の解消に工数が必要となる場合には保守性が損なわれていると評価される。「容易性」は少工数でシステムを構成可能であることを示す項目である。分散型制御システムでは接続されるノード数が多くなるため、それぞれについて設定が必要となるが、それらについて一括で設定可能であったり、自動で判定される場合には容易性が高いと評価される。

展開カテゴリについては「継続性」「拡張性」「柔軟性」「互換性」を挙げている。「継続性」は、システムの更新によってそれまで使用していた機能が損なわれないことを保証することを示す項目である。インターフェースやデバイスドライバについて、システムを載せるデバイスを変更しても同様に利用が可能なことが望ましい。これらが対応されていない場合には、利用したいセンサデバイスであったり、機能が制限されてしまうため評価が低くなる[1]。「拡張性」は関節数やリンク数の多いロボットにシステムを移植する際に、既存の制御用ノードの追加と一部の設定変更のみで規模の拡大を達成できることを示す項目である。制御対象となるアクチュエータやセンサ数の増大に対して、システム側で受け入れらるるデバイス数を超えてしまう場合には拡張性が低く、システムの展開が困難となってしまうため、評価が低くなる。また、各種インターフェースのセンサデバイスに対してシステムに容易に接続できる場合にも拡張性は高いと評価可能である。「柔軟性」はシステム構成の変更を容易にできることを示す項目である。上記の拡張性にて述べたノード数の変更といった規模の変更に加え、ノードの接続順番やノードを構成するデバイスの変更に対して、システム側での設定の変更が最小限に留まる場合に柔軟性が高いと評価される。「互換性」はノードを構成するデバイスの更新に対して、システムの未更新部が混在しても引き続き利用が可能なことを示す項目である。互換性が無い場合には、制御ノードデバイスの新機種への更新に際して、関係するノードの制御ソフトウェアやインターフェース種類も合わせて更新されなければならず、システムの展開性を損なっていると評価される[1]。

テストカテゴリについては「可観測性」「検証可能性」を挙げている。「可観測性」は分散型制御システム内で実行されている処理についてトレースが可能であることを示す項目である。特に不具合が生じた際には、原因を特定する必要があるが処理のログであったり、リアルタイムでの処理結果のモニタリングが行えないとデバッグは困難を極めることになる。「検証可能性」は実装されたある処理について、ロジックが正しく動作しているかテストパターンと比較することが可能であることを示す項目である。可観測性で述べたトレースの可不可であったり、テスト用のシミュレーション環境が提供されていることで検証可能であると評価される。

以上に挙げた項目について要件を満足するよう設計を進めていく。ある考案手法について、上記要件について検証を行い、ある評価項目において容認できない欠点が考えられる場合には、その手法の採用は不可とし、逆に既存手法と比較して総合的に優れていると考えられる場合には、その手法を採用することで設計を進めた。現実には全てにおいて要件を満足する解法は中々存在しない。そこで本研究では上記項目について特に優先度の高い項目を設定し、それらを優先的に満足する方法を採るよう設計を進めた。優先度の高い項目として以下を設定した。

1. 制御性能
   演算性能、実時間性、応答性、安全性

2. 信頼性

[1]実例として、上位層制御システムの Linux OS を更新したところ、同じ計算機であるにも関わらず制御演算プログラムが以前の倍以上の時間がかかるようになってしまった。これは、更新によって CPU のパワーセーブモードへの自動切換えモジュールがインストールされてしまったためであった。この場合は、このモジュールが原因と特定できたため無効にするのみであるが、現実には多岐にわたる要因を確認する必要が生じ、非常に工数を要してしまう。

耐熱、耐振動、耐衝撃、耐障害、耐ノイズ、故障頻度

3. 設計/運用
   小サイズ、軽量

以上は致命的な障害に繋がる要因、タスク実行能力の確保において重要要因となっているため、優先度を高く設定することは妥当であると思われる。

2.4 人体における高速応答行動の実現機構

2.4.1 人体の生体筋の特性とその利用

現在のヒューマノイドロボット研究では、人間と同等の運動能力が一つの達成目標になっている。現状のヒューマノイドロボットは、単純な関節速度と関節トルクを見たときの出力においては、人間と同等かそれ以上を発揮することが可能である。浦田らは人間の筋特性モデルである Dennis らの式に基づいて、等身大ヒューマノイドロボットにおいては静的な最大出力関節トルクが 200Nm 以上、最大関節角速度が 800-1000[deg/sec] 以上必要であることに着眼し、同等の出力特性発揮可能なヒューマノイドロボット用の大出力モータ駆動系を実現している [7, 8]。

しかし、人間はそれと同時に衝撃入力時にも力を緩和できるしなやかさを備えている。これは、アクチュエータである筋肉が機械的には出力発生器であると同時に粘弾性要素としての性質も兼ね備えているからであり、これにより衝撃入力を機械的に緩和することが可能となっている。また、反射神経系によるフィードバックによって筋の緊張状態を短い時間で衝撃入力に対して応答させて抑制指令を伝達することで、筋を弛緩させていることも衝撃入力緩和に貢献している。

対してロボットの場合、たわみによる制御追従性能の悪化を防ぐために関節は高精度に動作するよう高剛性な構造となっており、衝撃入力を緩和できるほどの粘弾性を備えていない。上述の大出力モータ駆動系では減速比 240:1 以上の比較的高い減速機を使用しているため、バックドライバビリティは低い構成となっている。関節角度規範の制御則を用いる限りにおいては減速比の高さは問題とならないが、衝撃吸収のためのしなやかさを求める場合にはこの関節の固さは問題となる。センサフィードバックによる力制御によって衝撃入力緩和動作を行うことも考えられるが、衝撃入力のようなインパルス的入力に対しては制御周期が間に合わない。そのため、十分な撃力の緩和を行うことができず、関節の減速機や骨格にダメージを与えてしまう。これが、ロボットの行動において安全性・信頼性を低下させている要因の一つとなっている。

本研究の目的の一つは、従来しなやかさを備えていなかった大出力モータ駆動系において、その出力性能を維持しながら同時にしなやかな動作を実現することとなる。表 2.1 に、関節の出力特性について人間と既存研究における性能、本研究で目標とする性能を示す。

表 2.1: Comparison of joint torque performance between human and robots.

	human[7, 8]	prior research[7]	our target
Max. Torque [Nm]	Over 200	Over 200	Over 200
Max. Speed [deg/sec]	Over 800	Over 800	Over 800
Flexibility	Inherent viscoelasticity	Limited (Utilized by force feedback)	Inherent backdrivability (Utilized by torque control)

現状の大出力モータ駆動系の性能を維持しつつ、人間のようなしなやかな身体能力を持たせるためのアイデアとして、

1. ロボットの関節や骨格に、人間のような粘弾性特性を機械的に挿入する
2. 制御周期の大幅な向上によって衝撃入力に対して高速にフィードバック応答可能なアーキテクチャを採用する

3. 関節制御則のトルク制御化によって、物理的な作用による高速応答を行う

ことが考えられる。(1)(2)(3) とも多くの研究例があり、これらの手法の混合によって、より人間のような身体能力をロボットに持たせられると考えられる。表 2.2 に各手法の比較をまとめた。

(1) の場合、粘弾性要素の挿入によって関節の位置追従精度は従来より悪化してしまう。そのため、粘弾性を考慮した制御系の設計によってロボットの動作に影響が出ないようにする必要がある。また、機械的に追加の機構が入るために、ロボットの重量増加・外形の誇大化・機械的信頼性の悪化・コストの上昇といった課題についてもロボットの設計時に考慮された上で採用される必要がある。

(2) の場合、制御周期・アーキテクチャの応答性能および演算性能が重要となる。実際には、制御周期には限界があるために、(1) のように機械的に粘弾性要素を挿入することは不可欠と考えられる。また、力フィードバックでは力センサの精度が、外乱衝撃入力に対する応答性を決定づけるが、一般には力センサの出力には無視できないノイズが重畳しており、フィルタ処理は不可欠となっている。フィルタ処理による応答性の低下は不可避であり、そのために対応できる衝撃入力に上限ができることになってしまう。高精度な力センサを導入することで応答性の向上を試みることも可能であるが、その場合にはセンサ自体の機械的サイズ・重量の増大を伴うことが一般的である。また一般に高精度な力センサは、その精密さ故に衝撃や過負荷に対して非常に弱いという特性があり、結果的に機械的信頼性の低下に繋がってしまうという問題が生じる。しかし、制御側での衝撃緩和能力を高めることによって機械的粘弾性要素への要求仕様が弱まるために、上に挙げた (1) の課題を効果的に解決できる可能性がある。

(3) の場合、アクチュエータのバックドライバビリティが確保されている場合には有効であると考えられる。また、バックドライバビリティが低い場合においても、アクチュエータモデルによる補償トルクの重畳によってバックドライバビリティを改善することは可能である。ただし、外力によって本来計画されていた関節軌道から大きく外れてしまうことをトルク制御は許容してしまうため、別途軌道の修正則を適用する必要が生じる。制御則としてアクチュエータのバックドライバビリティを補償することで、より効果的に外乱に対して応答が可能になると考えられる。また、トルク制御は出力が線形な電気モータを使用する場合には、センサフィードバックに依存しないフィードフォワード制御のみである程度の目標値追従性が実現可能であるため、上記2 つの方法と比較して新たな機構の追加による重量増・外形誇大化・機械的信頼性低下といった問題を回避することが可能である点において有利であると考える。特に関節のトルクセンサレスで運用が可能となると、大幅に重量・サイズ・配線数を低減することが可能となり、信頼性の向上に大きく寄与すると考えられる。

表 2.2: Comparison of methods to achive flexible joint.

Method	Mechanical	Force Feedback	Feed-Forward Torque Control
Mechanical stiffness	yes	no	yes
Weight	no	no	yes
Size	no	no	yes
Reliability	no	no	yes
Controllability	no	yes	no
Responsiveness	yes	no	yes
Necessary Equipment	Spring Damper	Force/Torque sensor	Backdrivable reducer

また、上記のどの手法においても制御系設計時には筋の粘弾性特性の制御による模倣、反射神経系のフィードバック回路の模倣を行うことで、バイオメカニクス的知見を元にした撃力緩和をロボットのアーキテクチャ設計に反映させることは合理的であると考えられる。以下では生体筋の機械的特性と反射神経系のメカニズムについて述べ、ロボットのアーキテクチャ設計のヒントとしていく。

生体筋の特性

- 筋長-張力関係

生体筋運動特性は筋長に依存して大きく異なり，その様子を図 2.13 に示す (from [9, p.86])．なお，図中の生体長とは関節可動範囲の中央値における筋長を表す．図 2.13 の A は静止状態における筋の筋長-張力関係を示しており，生体長よりも伸長される場合にはしだいに張力が増加する傾向を示すハードスプリングと同様の特性となっている．また，B は収縮時の各筋長における筋張力を示しており，張力は生体長において極大値をとり，筋長変化に伴って張力は減少していく．この場合には伸長時には A で示された筋の有するハードスプリング特性によってある程度伸長した後，再び張力は増加傾向を示す．

- **筋速度-力関係**
  生体筋の運動特性は等張力収縮・等速度収縮時における筋に働く負荷張力の大きさによって筋長変化速度にも影響を与える (図 2.14)．等張力収縮時の筋の収縮速度は負荷の増大に伴って双曲線的に減少するが，これは低速度領域において負荷の増大に応じて筋の粘性特性が増大すると読み替えることもできる．以上の関係は Hill の特性式によって以下のように定性的に表されることが知られている [10].

$$(P + a)(v + b) = b(P_0 + a) \tag{2.1}$$

この式は等張力収縮における負荷 P と最大筋張力 P_0 の釣り合いを表しているが，筋の収縮速度 v によって発生する粘性摩擦との釣り合いを示すものでもある．高負荷状態で低速領域では相対的に粘性係数が増大することがこの式からもわかる．

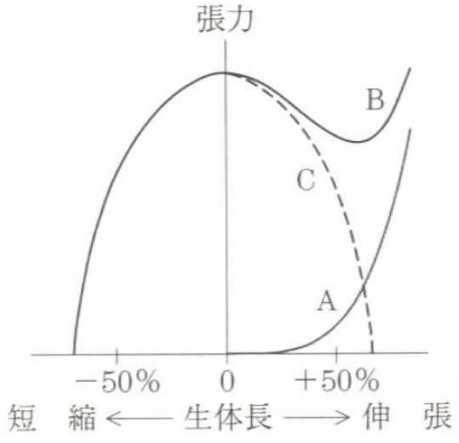


図 2.13: Relation of Muscle Length and Tension.(from [9, p.86-Fig6.5])

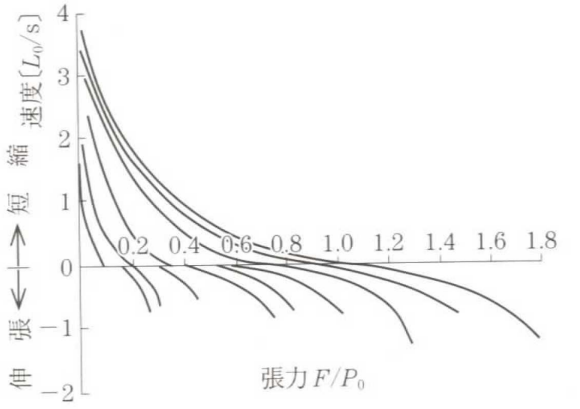


図 2.14: Relation of Muscle Speed and Tension.(from [9, p.90-Fig6.9])

これらの筋の筋長-張力特性及び筋速度-張力特性は筋繊維の構造に由来している．筋繊維はミオシンとアクチンによる張力発生器であり，同時に粘弾性要素でもあるため，一般にバネとダンパによる機械要素でモデル化することが可能である [9][11]．モデル中のバネの弾性係数に非線形性を持たせることによって生体筋のハードスプリング特性を出すことも可能である．

2.4.2 反射弓と相反性神経支配による高応答性神経系

末梢神経系

生体筋の駆動は図 2.15 に示されるフィードバック型神経接続によって実現されている．各々の働きについて簡単に述べていく．

- α 運動ニューロン
  上位中枢からの刺激あるいは筋紡錘からの刺激に応じて α 運動繊維へとインパルスを送る．

- γ 運動ニューロン
  筋紡錘の感度調節機能を実現するニューロンである。

- I_a 求心性繊維
  筋長変化に比例して活性かすることから、速度センサ入力としての機能を持つ。

- I_b 求心性繊維
  筋張力に応じて活性化することから、筋の張力センサとしての働きを持つ。

- 介在ニューロン
  神経繊維からの出力と運動ニューロンへの入力の間に介在して，神経パルス信号を中継するニューロンである．この介在ニューロンによる神経系の複雑なネットワーク構成によって，ある行動を実現するための反射回路やその回路のパラメータ調節機能が構成されている [11].

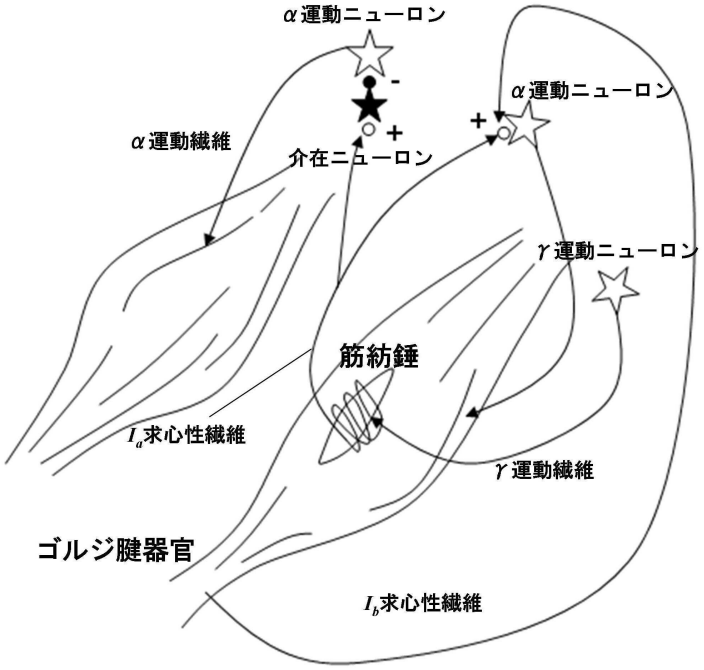


図 2.15: Structure of muscle and spinal cord.

反射弓と相反性神経支配の最短ループバック構成

反射弓は末梢神経を含む神経系の中で最下層に位置する神経系統であり、「筋紡錘，I_a 求心性繊維，脊髄，α 運動繊維，筋肉」の最短経路で構成される [9][12]。生体において拮抗筋の動作が互いに干渉し合わないように協調させることが重要であるが、反射弓によって構成される相反性神経支配 [13] という筋指令抑制機構がこれを実現している。さらにこの反射弓は上位神経系からの抑制化・活性化によるパラメータ調整を受けることによって、現在の動作目的に適した筋の運動特性を高い精度で実現することが可能となっている。

以上のような反射弓という神経系の最短経路のループバック回路の構成によって、電子回路による情報伝達速度に比べてはるかに低速な神経伝達機能でも高精度な身体運動を実現するためのフィードバック制御が構成されている。

本研究でも人間の反射弓のような短経路フィードバック制御をロボットの制御システムに組み込むことによって、特に実時間制約が要求される応答について応答性の向上を目指す。

2.5 分散型制御システムのためのノード間時刻同期

2.5.1 ノード間時刻同期

分散制御系では各ノードがそれぞれ非同期に動作を行い、データ取得やモータ制御の実行を行う。しかしロボットにおいて全身の各関節が協調して精密な動作を行う場合、データ取得タイミングを全ノードで同期させることで、制御器の性能を向上させることが可能となる。例えば、2 足歩行型のロボットにおいて足先の6 軸力センサの情報を解析する場合に、右足と左足のセンサ入力が同期していないと、位相遅れの大きさに応じた ZMP 値の誤差がたとえセンサ値自体が高精度で計測できていた場合でも発生してしまうことになる。従って、分散制御系においてセンサデータの計測を高精度で同期させ、時刻(タイムスタンプ) 付きで収集されることが重要となる。

一般に組込みシステムでは基準となる一定周波数のクロックを水晶発振子やオシレータ等から入力して、カウントアップすることで時間計測を行う。あるいは、リアルタイムクロック機能を持ったチップ等により日時付きで時刻を取得する。しかし高精度な時刻を取得しようとした場合、

- 同一周波数の発振子でも実クロックには個体差がある。
- 温度条件による周波数のドリフトが生じる。
- 分散システムではノード間でクロックの周波数· 位相が異なる。
- ほとんどのリアルタイムクロックは精度が秒単位である。

といった問題があるため、基準となる時刻あるいはクロックへ同期するための仕組みが必要となる。一般的に使用される時刻同期手法を次に挙げる。

2.5.2 Network Time Protocol

ネットワーク越しの時刻同期手法として一般的に使用されているのは Network Time Protocol(NTP) がある [14]。NTP はマスターとスレーブの2 ノード間でパケットを授受し合い、パケットに付与したタイムスタンプの差分からネットワークを介することによる遅延誤差を考慮したマスター時刻にスレーブ時刻を同期させる手法である。

実装が簡単で、ハードコーディングレベルで実現可能であるが、ネットワーク遅延は送受信の間一定であるという仮定での手法であるため、ネットワーク遅延が状況に刻々と変化する通信系の場合には同期精度は悪化する。

2.5.3 IEEE1588

また、NTP より高精度に分散ノード間で同期を行うための手法 (Precision Time Protocol(PTP)) が IEEE1588 にて提唱されており、分散ノード間で高精度で同期された制御を行う研究が行われている [15][16]。IEEE1588 の時刻同期精度はマイクロ秒から最大ナノ秒オーダを実現しており、クロック単位での同期が可能となっている。Best Master Clock(BMC) アルゴリズムによって実装されており、基本的には NTP と同様にマスターノードとの間のタイムスタンプ差分を元のクロック誤差を刻々と修正していく。

本研究では限られた FPGA 資源で実装を行うことや、NTP で十分な同期精度を実現可能であると考え、IEEE1588 については未実装としている。

2.5.4 EtherCAT での時刻同期

通信規格自体で時刻同期の機構を提供している例としてここでは EtherCAT を挙げる [17]。EtherCAT では IEEE1588 ベースの分散クロック (Distributed Clock) 方式を採用しており、時刻基準となるクロックマスターノードがネットワーク内に接続され、他のスレーブノードはクロックマスターノードから分配されてくる基準時刻を参照することでローカルクロックをクロックマスターに同期させる。クロック参照時には、上り 下りのフレームのクロックの差から通信遅延による誤差を計測し、誤差補償を行うことによって

1μsec 以下の精度が実現されている。10 ノード 程度の規模であれば、ノード 間の誤差を 10nsec に抑えられるという 報告もある [18][19]。これらは実用的には 1msec 周期での通信に利用される。

なお EtherCAT ではあくまでも、通信は EtherCAT Master が管理を行うため、クロック分配についても EtherCAT Master が制御を行い、クロックマスターノード は被制御器として振る舞う。EtherCAT Master はクロック同期用のコマンドを送信することで時刻同期を制御する。

2.5.5 グローバル時刻同期

あるロボット の体内システムだけで完結している場合には、各体内ノード の時刻が同期さえされていれば制御のための基準時刻としては十分である。しかし、センサ取得時刻を後で参照したい場合や体外のシステムで利用される場合には、センサデータに付加するタイムスタンプがグローバルな世界標準時と同期していることが望ましい。従って、体内システムで時刻同期の基準となるマスターノード の時刻はさらにグローバルな基準時刻に同期させておく仕組みが必要となる。

グローバル時刻への同期を行う場合、インターネット 経由で NTP サーバを利用するのが一般的で簡単な方法である。その他に、GPS 信号を利用するという 手法があり、信号を受信可能な環境であればインターネット に接続されていなくてもグローバル時刻に同期できるため、ロボット の自立性を高めることが可能となる。

2.6 高速応答行動のための実時間システム

ここまで述べてきたようにロボット の制御器における情報処理系では、センサ入力から得られる状態を動作へと即座に反映することが要求される。工学的に「 **実時間**」という 言葉は「 ある処理を定められた締め切り 時間までに終えること」と定義される。例えば実時間プロセスであれば、そのプロセスは一連の処理をあらかじめ設定された時刻までに終えなければならない (図 2.16)。物理的な時間に制約されているという点で実時間プロセスは通常のプロセスと異なる。特に産業用のプロセッサや通信リンクで実時間性 (リアルタイム性) を実現した、と表現されている場合、応答時間が決定的 (Deterministic) という 意味での実時間性が保証されるようなアーキテクチャとなっている。

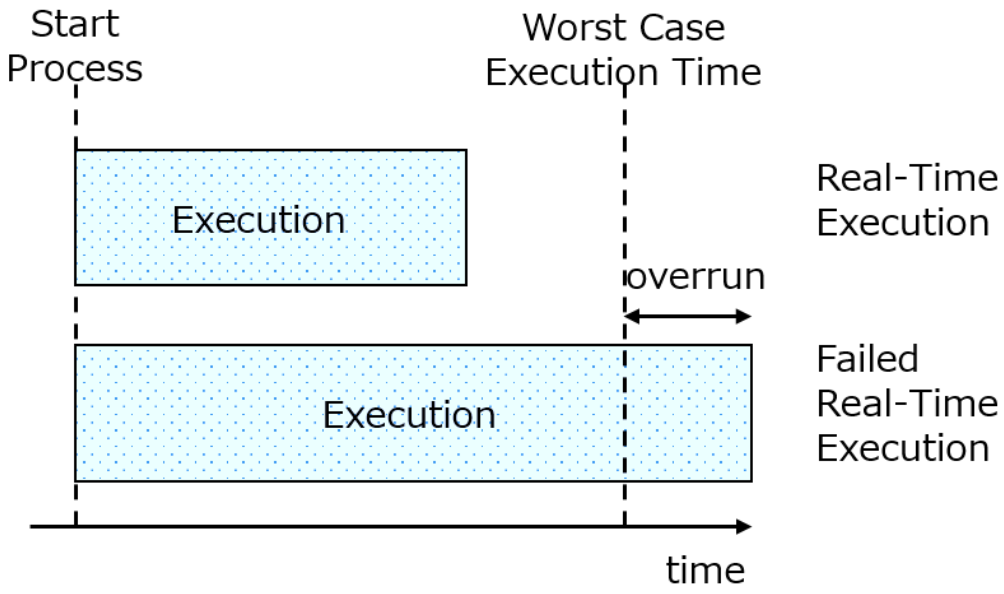


図 2.16: Real-time process execution. Upper case is a real-time process because the process ends befoce worst case execution time. Lower case is not a real-time process because the process overruns the worst case execution time.

締め切り時間は「最悪実行時間」として表現される。単純にはプロセスが処理を終えるまでの時間は、処理速度と処理量だけで決定される時間によって確定されるため、最悪実行時間がその時間を上回っていればその処理は実時間であると言うことができる。しかし現実的には処理速度・処理量が一意に決定されるとは限らないため、あらかじめ処理時間を事前に確定させた上で最悪実行時間を設定することは不可能である。実時間処理を実現する上で障害となる要素は以下が考えられる。

1. 条件分岐による処理量の変化
   条件分岐によってその後の処理量に差が出る場合に、必要となる処理時間は異なってくる。従って、条件分岐がある場合には最悪実行時間は、処理時間が最大となる分岐に合わせて設定される必要がある。

2. 処理量に対する演算性能の不足
   画像処理等の負荷の高い処理では、単純に処理量が多い。そのため、たとえハードウェアの最大性能で演算を行えたとしても長い処理時間を要してしまうような処理では、目標とする最悪実行時間を満たせないケースも出てくる。その場合には、ハードウェア側での演算性能の向上、処理アルゴリズムの改良による処理量の削減が必要となる。

3. ネットワークの遅延
   分散型システムで他ノードとのデータのやり取りを行うプロセスの場合、ネットワーク経由でのデータリードのタイミングはネットワーク構成によっては不確定であり、遅延を生じる。

4. リクエスト応答待ち
   上記のネットワークの遅延と同様、プロセッサから利用可能なハードウェア資源について、アクセス要求を開始して即座に応答を得られるのが理想的である。しかし、演算器や物理メモリやストレージ、CPU バス、I/O へのアクセスはデバイスそのものの応答遅延、他プロセスからのアクセス要求との衝突によって、即座に応答が得られるわけではない。また、メモリのページング処理による待ちのようにＯＳ側の処理待ち時間も存在する。リクエストの応答があるまでプロセスは待ち状態になり、実時間制約を満たせない場合が出てくる。ネットワーク越しにリクエストを送信し、アクノレッジが返るまで待つ場合はネットワーク遅延の影響が含まれるため、さらに実時間性の悪化を招く。

5. タスク数に対するハードウェア数の不足
   制御システムが動作する計算機上では、単一のプロセスが走る場合にはそのプロセスが演算器や I/O といったハードウェア資源を常時使用することが可能である。しかし実際にはシステム内で走るプロセスは複数存在し、お互いに独立して動作をするため、全てのプロセスが同時にハードウェア上で動作することは不可能である。Operating System(OS) はプロセスの動作を管理し、ハードウェア資源にアクセスするプロセスの割り振りを行うことで、有限な個数のハードウェア資源に対して複数のプロセスでの共有を可能としている。このような仕組みのため、プロセスは実際には物理的なプロセッサの上ではなく、ＯＳから割り当てられた仮想的なプロセッサの上で動作を行う。ＯＳは後述するコンテキストスイッチによって、物理プロセッサを割り当てるプロセスあるいはスレッドの切替を行う。物理プロセッサを割り当てられていない間は、プロセスは一切処理を行うことができないため、演算速度自体は十分に高速でも実時間制約を満たせない場合が出てくる (図 2.17)。

実時間性の実現において解決すべき技術的な問題は、「処理能力が有限なハードウェア資源を、複数のタスク間で時間軸・空間軸についてそれぞれどのように分配するか」という問題に言い換えることができると考える。

リアルタイムＯＳやリアルタイムプロセッサはこのような問題を解決するために開発された仕組みを有している。リアルタイムＯＳはプロセスへの処理の割り振りにおいて、実時間スケジューリングアルゴリズムを適用することで、プロセスが最悪実行時間以内に処理が割り振られるよう管理を行う。また、リアルタイムプロセッサではキャッシュメモリや外部メモリを低遅延性を優先した構成としたり、専用のコンテキスト保持機構を搭載することで、リクエスト応答待ち等による応答時間の揺らぎを削減することで、プロセス処理時間を Deterministic にすることを目指している。

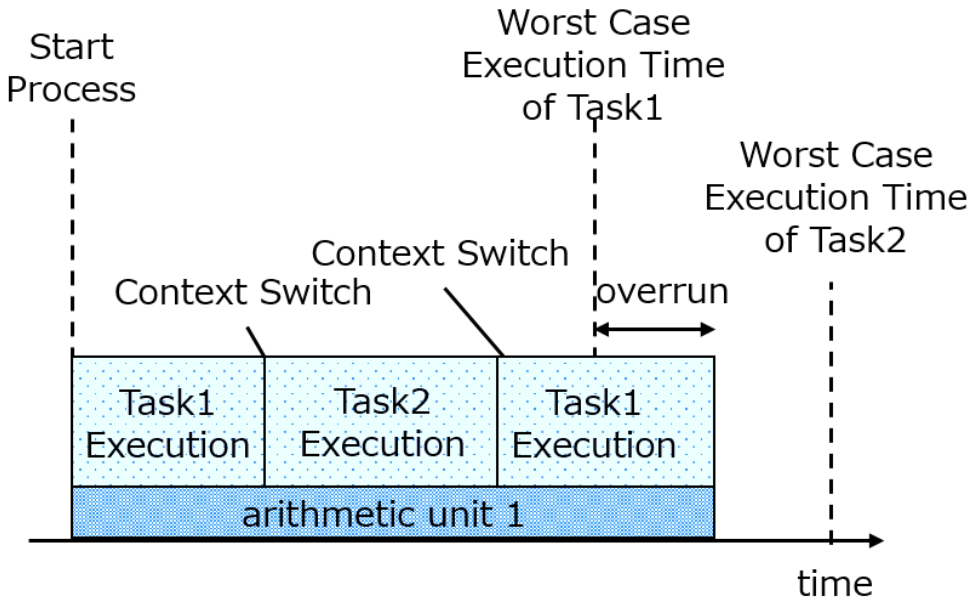


図 2.17: Context switch between Task1 and Task2. Task1 and Task2 request to use arithmetic unit 1 at the same time. Due to the context switch, Task1 stops executing, and during Task2 is running, Task1 cannot execute process. After next context switch, Task1 restarts its execution but overruns its WCET.

2.6.1 実時間スケジューリング

前節にて言及したように、ＯＳ により提供されるマルチスレッド機能ではソフトウェアによって、プロセスやスレッドの切替 (コンテキストスイッチ) を行う (図 2.17)。コンテキストスイッチは、物理プロセッサの持つ資源の仮想プロセッサへの割り当てを入れ替える処理であり、プロセッサ内レジスタの状態の一時退避と切替先プロセスのレジスタ状態の復帰が主な内容となる。

コンテキストスイッチは OS カーネルのソフトウェア管理によるメモリへのストア・ロードを行う方法と、プロセッサのアーキテクチャ自体にコンテキストスイッチの支援用ハードウェアを実装しておく方法が考えられている [20]。コンテキストスイッチが行われるタイミングは、スケジューラ管理によるスイッチ、割込み信号によるスイッチ、I/O アクセス API コールのタイミングで行われる。特にスケジューラによるコンテキストスイッチはマルチスレッドＯＳ では重要な実装の一つであり、複数タスクの並行処理時の応答性に関わってくる。コンテキストスイッチ周期は短いほどタスク処理の応答性は向上するが、コンテキストスイッチは一般に重い処理となっており、高頻度でのコンテキストスイッチの発生はシステム全体のスループットパフォーマンスを低下させる。

このコンテキストスイッチ処理の負荷は、ＯＳ のスケジューラ、コンテキストスイッチ処理の実装、プロセッサのアーキテクチャにも依存するが、一般的な Linux カーネルの場合例外処理ハンドラ等の呼び出しのオーバヘッドが重畳され約 7000〜8000 クロック程度のオーバヘッドを生じる。2GHz 程度のクロック周波数のプロセッサでは約 $4\mu sec$ ほどのオーバヘッドとなり、100MHz 程度の組込み向けプロセッサの場合は約 $80\mu sec$ ほどのオーバヘッドがコンテキストスイッチだけで発生してしまう計算となる。

応答性を要求されるシステムにおいて、割込みやプロセスを切り替える処理だけで $100\mu sec$ に近い遅延が発生するのは許容できない。また通常の Linux カーネルでは、応答性を重視した実時間実装の設計とはなっていないため実際には数百 μsec から数 $msec$ 程度の遅延が生じる。従って、応答性を要求されるシステムの開発においては、コンテキストスイッチのオーバヘッドが小さく、応答性能の高い実時間ＯＳ・プロセッサアーキテクチャが採用される。

実時間ＯＳ として、Linux カーネルベースの ART-Linux、RT-Linux、Xenomai や、TRON 系の T-Kernel、μT-Kernel 等多くの実装が公開されている。これらの実装では最悪実行時間と現在のタスク・プロセスの実行時間に応じてスケジューリングを行う実時間スケジューラが実装されている。表 2.3 に一般的な Linus ＯＳ で使用可能なスケジューリングポリシを示す。この表に示すように、通常のカーネル実装ではあくまで

も優先度に基づいて実行順序を制御する機能が実装されているのみであり、タスク・プロセスの最悪実行時間と現在の実行時間によって優先度を動的に変更するような実時間スケジューリングは実装されていない。

表 2.3: 一般的なＯＳ のスケジューラポリシ

スケジューラポリシ	概要
時分割	スレッド生成時のデフォルトの時分割スケジューリング。動的優先度である nice 値を設定することで、デフォルトの時分割スケジューリングポリシのスレッド間においても、物理プロセッサへの割り当て頻度を上下させることが可能となっている。
ファーストイン・ファーストアウト (FIFO)	デフォルトの時分割ポリシよりも優先度が高く設定されている。同じ優先度のスレッド間では、スレッドが発行された順に FIFO に積まれ、他に優先度の高いタスクが無い場合には順に実行される。スレッドは、スリープや高優先度のスレッドが発行されたことによるコンテキストスイッチ等で一時的に処理を停止する際には FIFO に再度積まれ、次の実行順番を待つ。
ラウンドロビン	FIFO ポリシの拡張版となっており、FIFO から発行されたスレッドは最大時間単位の間だけ実行した後、FIFO に再度積まれる。同優先度スレッド間で均等にプロセッサが割り振られるポリシとなっている。
バッチ処理タスク	デフォルトの時分割スケジューリングにおいて nice 値による優先度変更を受け付けないポリシとなっている。
アイドルタスク	最も優先度が低く、他のスレッドが存在しないときに実行される

また、Linux カーネルの応答性を高めるために、スケジューラ実行周期を $1msec$ と通常のカーネルの設定値よりも短くした low-latency カーネルも提供されている。スケジューラ自体はオリジナルと差分が無いため、厳密な最悪実行時間に応じたスケジューリングではないが、多くの用途において実用に耐える性能を有している。このように、実時間ＯＳ の実装は多く提供されているが、

- 対応ハードウェアプラットフォームが限定的
- 開発速度は Linux カーネル本流と比較して遅い
- 応答性とスループットのトレードオフの存在
- 実時間アプリケーションの他ＯＳ やハードウェアプラットフォームとの互換性

といった課題もあり、システムのハードウェアアップデート停滞の要因の一つとなっている。ロボットシステムでは、最新規格のハードウェアプラットフォーム・ＯＳ カーネルバージョンを要求するデバイスやソフトウェア迅速に採用したいという要求がある一方で、基幹となる制御システムは上述の問題により簡単には更新ができないため、ロボット全体としての開発停滞につながり、システム全体の陳腐化を招いてしまう。従って、ロボット制御システムの設計では、システム全体としては非実時間系と実時間系でハードウェアプラットフォームごとに分割された分散型システムとすることでこれらの問題を回避することが解決策として考えられる (図 2.7)。多くのハードウェアプラットフォームでは Ethernet のような汎用インターフェースをサポートしているため、分散型システムを構成することは可能となる。

また、ＯＳ によるマルチスレッド管理に頼らず、ベアメタル実装によってタイマ割込み等を直接扱うことで複数の処理を高い実時間性で実現するという組込みシステムの開発方法も採られることがある。しかし、ＯＳ レスでのプログラム開発は、モータ制御と同時にセンサ処理や通信処理、状態監視等の多種多様なタ

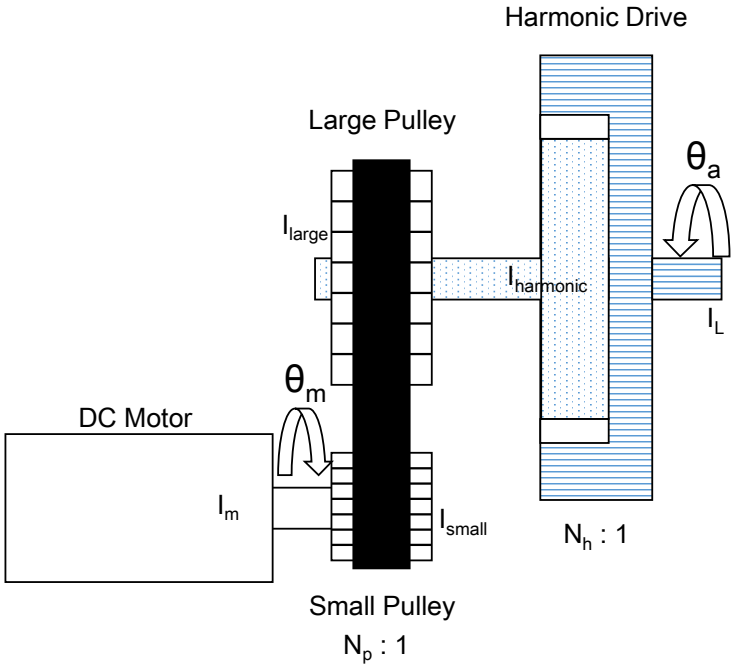


図 2.18: Actuator Model

スクを同時に行う必要があるロボットシステムにおいては、非常に開発難易度が上がってしまうため現実的とは言えない。

実時間ＯＳによる手法による問題への別のアプローチとして、プロセッサのアーキテクチャで実時間実行のサポート機能を提供する方法がある。ハードウェア実装により最適化されたスレッドスケジューリングによって、ソフトウェアによる実行では達成できない品質での低遅延性、実時間性を実現することが可能となる。本研究では山崎らの提案する実時間プロセッサ [21] を用いることで、目的であるロボット体内システムにおける高速応答が可能な組込みシステムが実装できないか検証していく。詳細については、第 3 章にて述べる。

2.6.2 遅延の制御システムに対する影響

分散型制御システムにおいて、ネットワークを介在させることによって生じる遅延や制約が制御ループのパフォーマンスに影響を与えることが、Ray[22] によって指摘された。

平井ら [23] や森 [24]、PooGyeon[25] らは不定な状態フィードバック遅延がシステムの安定性に与える影響について、システムが漸近安定となるための条件式について安定化マージンに応じた数種類の定式化を行っている。また、遅延の存在によってシステムの安定領域は狭まること、安定領域の求解において遅延を考慮することで安定領域解を拡大することが可能となることが示されている。逆に遅延を考慮しない安定領域についても求解することが可能であるが、遅延を考慮した場合と比較してその解領域は狭まることとなる。離散システムとなる計算機による制御ループの安定性についても制御周期と関係し、ネットワークの低遅延化・制御周期の高速化は制御系の安定化によってループゲインを上げて応答性を改善するための重要なパラメータとなる。

2.6.3 制御周期の制御性能に対する影響の評価

DC モータダイナミクスモデルを以下で表す。

$$I_m\ddot{\theta} = K_t i + \tau_{in} \tag{2.2}$$

$$L_m\dot{i} + K_v^{-1}\dot{\theta} + Ri = V_{in} \tag{2.3}$$

モータロータ角度を θ[rad]、モータ電流を i[A]、モータ駆動電圧を V_{in}[V]、モータ軸トルクを τ_{in}[Nm] としている。また、モータの電気・機械パラメータを表 2.4 に示す。一般にモータドライバでは PWM によるスイッチングによって入力電圧 V_{in}[V] を制御する。特に電流制御器が実装されている場合には、目標電流 i^{ref}[A] に i が追従するように V_{in} が調整される。電流制御による目標値への追従が十分に早いと仮定すると、式 2.2 は

$$I_m\ddot{\theta} = K_t i^{ref} + \tau_{in} \tag{2.4}$$

となり、ダイナミクスモデルを入力と出力の間が線形なモデルとして定式化され、制御において有利となる。実際のアクチュエータモデルでは、さらに減速機がモデル化される必要がある。本研究で用いるアクチュエータは図 2.18 のように、モータ出力をプーリ減速機によって減速し、さらにハーモニックドライブ減速機によって減速してロボットのリンクを駆動する出力となる2 段階の減速機構となっている。アクチュエータ出力軸回転角を θ_a とすると、

$$\begin{aligned}\Big(I_L + (I_{harmonic} + I_{large})N_h^2 + (I_{small} + I_m)(N_h N_p)^2\Big)\ddot{\theta}_a = {} & N_h N_p K_t i^{ref} \\ & + \tau_{in} \\ & - C_v\dot{\theta}_a \\ & - C_c\mathrm{sign}(\dot{\theta}_a)\end{aligned} \tag{2.5}$$

と定式化される。各パラメータの公称値を表 2.5 に示す。なお右辺後ろ2 項はそれぞれハーモニックドライブ減速機の粘性摩擦項とクーロン摩擦項となっている。このモデルでは省略しているが、ハーモニックドライブやプーリの剛性を考慮した弾性項も追加することが可能である。関節角度 θ_a をエンコーダ等により直接観測可能な場合には、弾性項を付加することで関節軸トルクの推定が容易となり、制御性能が向上することが知られている。本研究では、このエンコーダが搭載されていないため、弾性項のモデル化は簡単のために省略した。

図 2.19 には、i から θ_a への伝達関数と i から $\tau_{out} = \tau_{in}$ への伝達関数のそれぞれの PI 制御時のステップ応答とボード線図を示す。関節角度制御については制御周期を 1kHz と 5kHz、トルク制御については制御周期を 1kHz、5kHz、100kHz としてモデル化を行っている。この図は MATLAB[3]にて離散時間 PID コントローラのモデル化を行い、stepplot 及び bodeplot 関数を用いて得ている。また、PID 制御器のパラメータチューニングは立ち上がり時間を重視しながら、オーバーシュート量が少なくなるように調整した。トルク制御器については実質 PI 制御となっている。図より関節角度に比べて関節トルクの制御帯域の方が広く、ベースとなる制御則としてトルク制御の応答性が優れていることが確認される。また、関節角度制御では 1kHz でも 5kHz でも Step 応答波形はほぼ変化していないが、トルク制御では 1kHz から 100kHz まで制御周期の増加とともに、ステップ応答の追従性が大幅に改善されているのが確認できる。これは、ロボットをトルク制御ベースで駆動する上で、制御周期の改善は効果が見込まれることを意味する。100kHz まで向上させればほぼ完全追従と見なすことができるが、本研究で使用する組込み用プロセッサで制御を回すには性能が不足してしまうそのため、本研究では現実的な 5kHz でのトルク制御が可能となるよう実装を進めていく。将来的には制御ロジックが固まれば、FPGA 等でハードウェアロジック化することで 100kHz での運用は可能になると思われる。

なお、現実にはさらにモデル化誤差の問題、弾性項の存在がある他、後述する制御タイミング Jitter の問題によって実性能はさらに異なることに留意する必要がある。これらの要因によって、制御周期を単純に向上させたとしても制御ゲインが期待値ほどには上げられず、上述のモデルほどの差異は生じない。逆にこれらの問題を考慮した制御系とすることで関節角度制御においても制御周期向上の効果を得られると考えられる。

2.6.4 実時間性の制御性能に対する影響の評価

現在ロボットの制御系の実行周期として 1msec 周期 (1000Hz) を採用するのが一般的である。これは従来の組込みプロセッサでデッドラインミスを起こさない範囲で高速な数値で、かつ関節角度サーボ制御を十分な精度で行うことができる値である。しかし、アクチュエータの大出力化、高速化、安全性能への要求によ

[3]https://www.mathworks.com/products/matlab.html

表 2.4: DC motor parameter

I_m	ロータ慣性モーメント	$3.33 * 10^{-5}$[kgm^2]
L_m	端子間インダクタンス	$3.86 * 10^{-5}$[H]
R	端子間抵抗	0.21[Ω]
K_t	トルク定数	$20.5 * 10^{-3}$[Nm/A]
K_v	回転数定数	48.78[rad/sec/V]

表 2.5: Actuator parameter

N_h	ハーモニックドライブ減速比	160
N_p	プーリ減速比	1.5
I_L	リンク慣性モーメント	[kgm^2]
$I_{harmonic}$	ハーモニックドライブ慣性モーメント	$9.40 * 10^{-6}$[kgm^2]
I_{large}	大プーリ慣性モーメント	$1.20 * 10^{-5}$[kgm^2]
I_{small}	小プーリ慣性モーメント	$1.86 * 10^{-6}$[kgm^2]
I_m	ロータ慣性モーメント	$3.33 * 10^{-6}$[kgm^2]
K_t	トルク定数	$20.5 * 10^{-3}$[Nm/A]
C_v	ハーモニックドライブ粘性摩擦係数	6-15[Nm/(rad/sec)]
C_c	ハーモニックドライブクーロン摩擦係数	5-10[Nm]

り、従来よりさらに高精度、高応答性を実現することが求められている。一般に制御周期の高速化は制御性能の向上に有効とされるが、ハードウェア制約が厳しい組込みシステムにおいてはハードウェアリソースの使用率上昇に伴う周期タスクのデッドラインミスやタイミング jitter の存在によって逆に制御性能が悪化してしまうという問題がある。

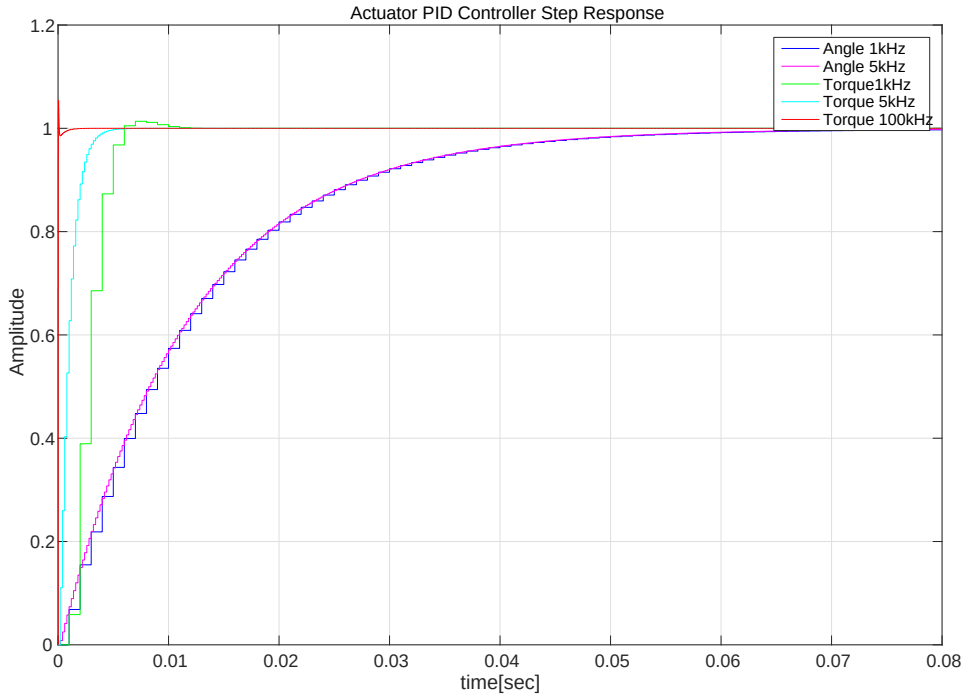
jitter が制御システムに与える影響については, サンプリングタイミングジッタによるセンサノイズ、制御タイミングジッタによるフィードバック系の性能低下が挙げられる。図 2.20 に jitter の存在によって周期実行タスクが影響を受ける仕組みの模式図を示す。実時間周期タスクでは基本的にリリース時刻からデッドラインまでの間でタスクが完了するようにスケジュールされる。すると図のように本来想定しているサンプリングタイミングからずれた時刻に信号のサンプリングを行ってしまう (サンプリングタイミングジッタ) ことで、計測された信号は真の信号に対して誤差が乗じた波形として観測されてしまう。同様に制御出力についても真の出力波形に対して誤差が乗じてしまう。

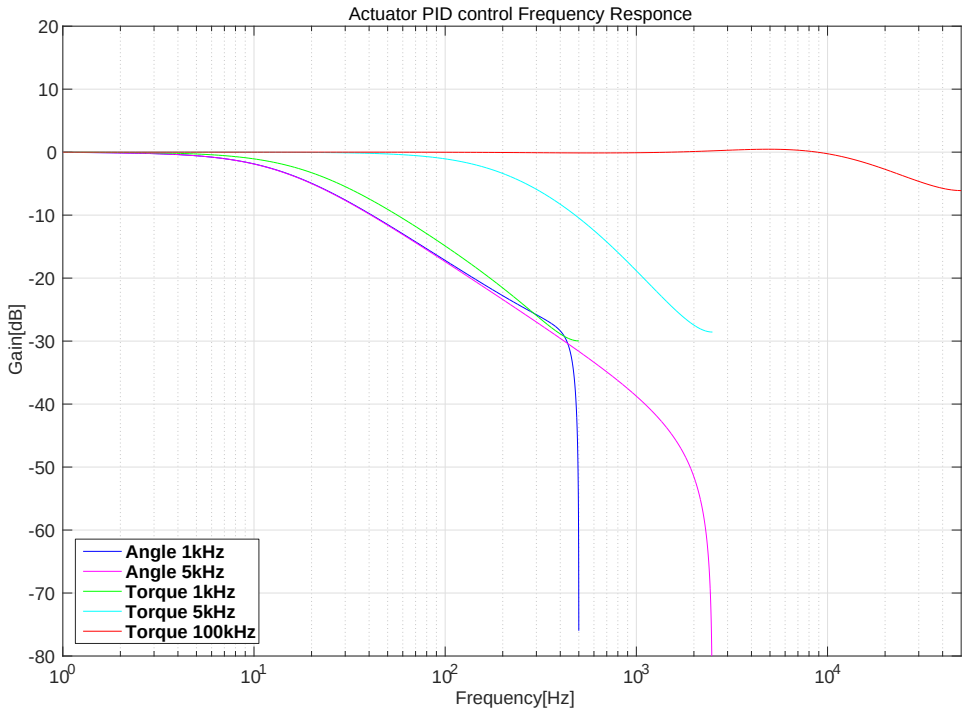
サンプリングタイミングジッタによって生じるセンサ値に乗じるノイズについては多くの研究が成されており、Da ら [26] は入力信号及びタイミングジッタのパワー P_x, P_τ について Signal-to-Noise Ratio(SNR) が以下のようになることを示した。

$$\mathrm{SNR} = 10 \cdot \log_{10}\left(\frac{P_x}{-\ddot{P}_x P_\tau}\right)\mathrm{dB} \tag{2.6}$$

タイミングジッタのパワー P_τ は jitter の分布について正規分布を仮定した場合に、その平均値 μ_τ と標準偏差 σ_τ を用いて $P_\tau = \sigma_\tau^2 + \mu_\tau^2$ と表される。従って、センサ計測値の SNR を向上させるためにはジッタのバラつき範囲を抑える必要がある。ロボット制御では、同時に多数のセンサ計測値を取得し処理する必要があるが、これらのタイミングジッタをソフトウェアスケジューリングで高精度にそろえることは困難であり、外部回路等で別途処理する必要がある。

また、Skaf ら [27] は Linear Quadratic Regulater(LQR) によるフィードバック制御系における制御タイミングジッタの影響について評価を行っている。LQR はロボットの制御システム内でも適用例の多い基本的な制御器であり、この制御器の理論性能上限の決定因子を知ることは重要事項である。Skaf らの研究では制御周期 T に対する jitter の割合 $\frac{\Delta}{T}$ が LQR の評価値の上限を制約することを示している。最大 20%のタイミングジッタが生じるシステム上における LQR ではノミナルシステムの評価値に対しておよそ 5%悪化





(b) Bode plot

図 2.19: Step and Bode plot of Actuator PID controller with angle feedback or torque feedback.

し、最大 50%の場合には 25%以上悪化することが報告されている。従ってアプリケーションにもよるがタイミングジッタは理想的には 0、最悪でも 20%程度に抑えておくのが望ましいと考えられる。従来のロボットシステムの制御周期 1–2[msec] は jitter 量 200–400[μsec] 程度存在するシステムにおいても十分なマージンがあると言えるが、100–200[μsec] 周期の制御タスクを実現するには jitter 量が 20–40[μsec] 程度に抑えられることが望ましい。特に分散型制御システムにおいては、コマンドやセンサデータの通信にかかるオーバヘッドによってネットワーク遅延が生じるため、これにより制御タイミングジッタが生じ、フィードバック制御の性能に対して同様に悪影響を与えることが知られている [28]。

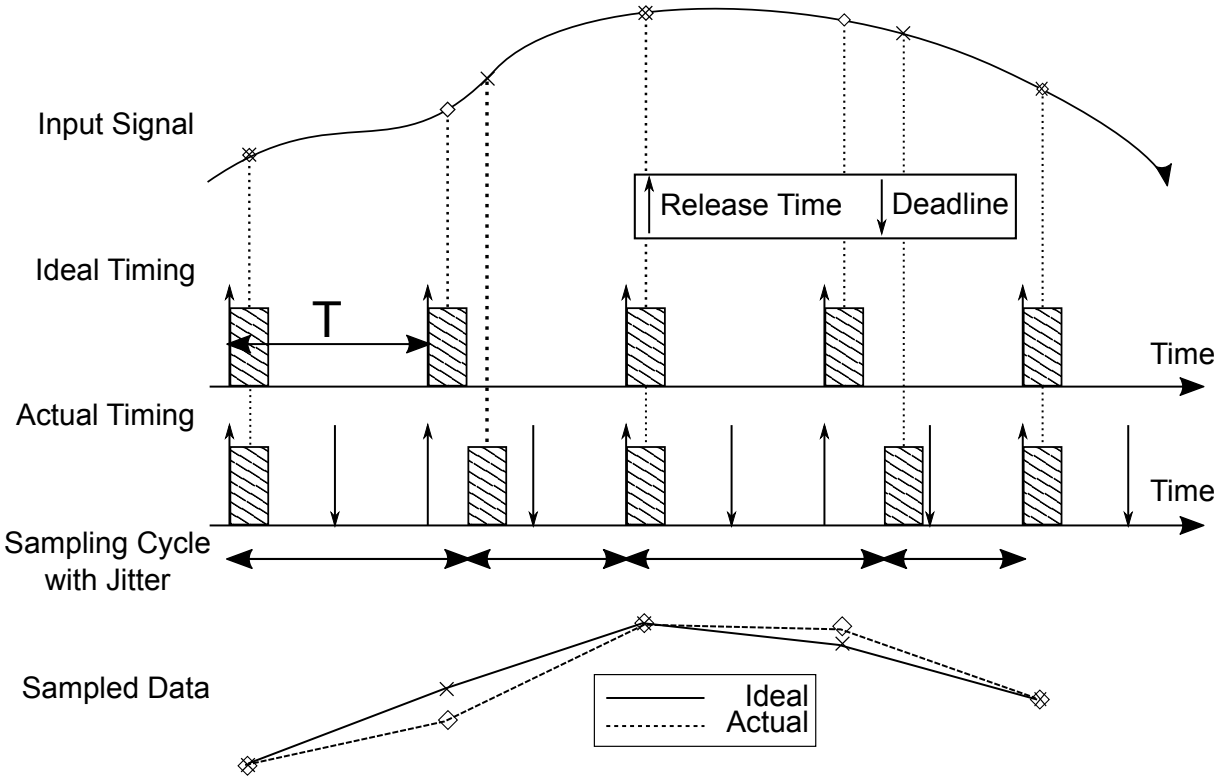


図 2.20: Effect of jitter on cyclic real-time task.

2.7 体内制御システムにおけるハードウェア制約

等身大ヒューマノイドのようなサイズ帯のロボットでは、外装や骨格、リンク機構、アクチュエータ等の機械部品が空間を占める割合が大きくなるため、必然的に電装系による制御システムに割り当てられる空間は極めて強い制約が課せられる。トータルパッケージとしての完成度が最終的にロボットのパフォーマンスを決定するため、自動車と同様に総合技術としての開発能力が要求される技術分野であると言える。その中で特にロボット分野では情報技術の重要性が高い分野であり、ロボット研究においても主要な関心の一つはシステム制御・人工知能などの情報処理技術であり、近年でもロボット分野で発展してきた技術が自動車開発の分野に採り入れられるという流れが加速している情勢である。ロボット体内システムは単に情報処理能力が仕様を満たすのみではなく、ロボットのトータルパッケージの最適化を見据えた構成とすることが重要であると考える。以下では、ロボットのパッケージデザインに影響を与える制御システムのハードウェア要素について述べる。

2.7.1 電力消費の改善

プロセッサは数十 W から数百 W の電力を消費するが、それらは熱として放出されなければならない。プロセッサの耐温度性能はおおよそ 85 ℃から産業・自動車用途で 125 ℃程度である。ロボットでは周辺環境によらすこの動作温度を超えないよう放熱設計がなされていなければならない。

この放熱方法がロボットに限らず、様々な実装で問題となってくる。汎用計算機では CPU や GPU 等のプロセッサの放熱には放熱用のフィンを持ったヒートシンクとファンによる空冷によって行われる。このヒートシンクは放熱量および周辺環境によって必要外形サイズがおおよそ決まり、体内の空間制約があるロボットにおいては事前に考慮されなければならない。また、ロボットのように振動や衝突による加速度が生じる系においては、ヒートシンクの固定方法が甘い場合に外れてしまう可能性も考慮しなければならない。水冷システムによる冷却も可能であるが、システム全体としてはラジエータやウォーターポンプによって大規模化してしまう。特に産業用途のように信頼性が要求される分野では、水冷やファンによる強制空冷はシ

ステム停止時に放熱性能が激減してしまうため、ヒートシンクによる自然放熱だけで十分な性能を出せるよう設計されるものがある。

プロセッサの消費電力はコア電圧 V[V] 及びクロック周波数 f[Hz] と関係があり、

$$P_d = \alpha f C V^2 + \alpha f I V \tag{2.7}$$

で表される。α はスイッチング率、C[F] は CMOS 負荷容量、I[A] は貫通電流である。ただし、P_d は CMOS ゲートのスイッチングに伴う消費電力であり、これとは別にリーク電流によるスイッチングとは独立した消費電力も存在する。

式 2.7 で示される通り、プロセッサの消費電力はクロック周波数に比例する。そのため、バッテリで動作する自律型ロボットのような電力量に制約のあるシステムでは、単純にクロックの速いプロセッサを搭載するという選択はできず、必要演算性能と電力系統のバランスをとる必要がある。近年では、定電圧コア技術や半導体プロセスの向上、クロックゲーティング等の技術によって電力あたりの演算性能は高まっているため、ロボットの低レイヤのプロセスを実行するには十分な性能となっている。しかしながら、実行周期の速い低レイヤプロセスと演算負荷の大きい認識系･ 計画系プロセスが混在したシステムでは、低レイヤに要求される実時間性能を維持するためには十分な演算性能余力が必要となってくる。そのため、演算パフォーマンスに特化した電力消費量の大きい計算機構成が採用される傾向が有り、必ずしもバッテリによる動作時間を延長する構成とはなっていない。

また、HRP3L-JSK(5.2 節) の場合、歩行時のサーボ系統の消費電力はおおよそ 200W から 800W 程度である。制御システムの消費電力は 160W から 240W ほどある。HRP3L-JSK は体外に主制御計算機を置いているが、その消費電力約 100W も合わせると制御システムの消費電力は全体の消費電力のおよそ 30 %から 50 %を占めている計算になる。

2.7.2 モジュール化

モジュール化の概念は、工業製品等で組み付け･ メンテナンス作業の効率化のために一般的に採り入れられている合理化のためのアイデアである。あるコンポーネントの着脱時に、他のコンポーネントの着脱を行う必要なく作業が行えるように設計されていることで実現できるため信頼性の向上にもつながる。分散型体内制御システムでは多数のノードが搭載される。その数に応じた電力配線や通信配線を通すことになるため、ノードへの配線アクセスは取り回しが簡単になるよう合理的にコネクタ類は配置されていなければならない。配線の数にもよるが、同一方向からケーブルの抜き差しができる構造になっているとモジュール性が高くなり、組み付け時に作業負担を軽減することができる。また、分散システムでは必然的に配線の分岐を用意しなければならず、基板側で配線分岐のサポートが成されているとさらに作業負担の軽減ができる。また基板のロボットへの取り付け方法についても、片方向からのアクセスだけで着脱ができるようねじ穴や、工具パスを設定しておくことや、複数基板をお互いに組みにした上でまとめてロボットに取り付けられるとさらに作業効率は上がる。

2.7.3 小パッケージ化

ロボット体内に分散配置されたモータドライバ基板についても、パッケージサイズを小さくすることでロボット全身のプロポーション設計の最適化に貢献が可能である。基板サイズは搭載される回路規模できまるが、近年では System on Chip(SoC) パッケージ化技術によって、1 つの素子に多くの機能を盛り込んだ製品が登場しているため、それらを積極採用することで従来よりもパッケージサイズを小さくすることが可能となる。また電子回路基板では、電力関係の素子が占める面積は1 割～5 割程度と無視できないほどに大きい場合がある。これについても、SoC 技術によって単電源で動作可能な素子が多くなってきているため、基板内電力系統の統一化によって小パッケージ化を進めることができる。また、コネクタは物理的強度を確保する都合から小サイズ化が難しい構成要素であり、基板内に占める面積は他素子の小型化に伴って、相対的に大きくなってきている。特に小さなコネクタは、保証挿抜回数が少なく、物理的な引張り力に対する強度も弱いため振動のあるロボット体内の通信系には不向きである。そのため、搭載インターフェースの小線規格への置換や、ポート数の削減等で対応するのが現実的である。

2.8 分散型制御システムのための通信系構成

分散型制御システムのための通信系構成について、まずその制約条件について述べ、通信モデルについて検討を行う。通信モデルを元に接続形態とメッセージング形態について、ロボット体内の分散型制御システムとして適した通信系構成と絡めて述べる。

2.8.1 ヒューマノイドロボット体内における通信系の制約

分散型制御システムの通信系を設計する上での検討項目について述べる。ロボットの体内において、多ノード間通信を構成する上での制約として、

1. アクチュエータ制御データ・センサデータの転送に低遅延性を確保する。
2. プロセッサ処理能力やメモリサイズが限定的な組込み系でも扱える。
3. 狭小な身体構造に収まるよう配線は必要最小源に留める。
4. 身体の構成に合わせて、通信系の接続構成も変更ができる。
5. 関節の繰り返し動作に耐えられるよう、繰り返し曲げ耐性の高いケーブルで構成する。
6. モータサーボノイズに対する耐性が高い物理層を利用する。
7. 工事現場等屋外作業を想定する場合には、防水性・防塵性を備える。
8. 長時間バッテリー運用のために可能な限り省電力での駆動が望ましい。
9. 多種多用なサードパーティ製デバイスを接続できる (I^2C や USB とブリッジ接続できる)。

が挙げられる。空間的制約が大きいのがロボット体内システムにおける特徴となっている。ロボット体内におけるノイズの影響については B.1 節にて測定を行い、その対策についても検討を行っている。

また、

1. ノード間距離は最大で 1m 程度。
2. 最大ノード数は 100 程度。
3. ノード数・接続構成の動的な変更は無い。
4. データフローの変動は小さい。

といった点については、汎用的なシステムと比較して有利となる。通信リンクのプロトコル、ハードウェア実装の両面から適切なシステム構成とすることが望ましい。

2.8.2 OSI 参照モデルと対比したシステム間通信モデル

通信モデルは様々な形態が提案・開発されており、幾つかのモデルの特徴を組み合わせて規格化されることもあるため、体系的に分類することは難しい。その中で代表的なモデルとして、ISO で策定された通信機能を表 2.6 に示すように7 階層に分割した OSI 参照モデル（ ISO/IEC 7498[29][4]）がある。

Ethernet では、このモデルに合わせて各層ごとにプロトコルが提供されており、必要な情報・データが各層でカプセル化されて、パケットが生成される。インターネットでは第3 、第4 階層のネットワーク層・トランスポート層に TCP/IP を用いることが一般的となっており、これを利用して HTTP や SSH、SMTP 等の各種通信モデルが実現されている。これらのプロトコルは汎用的であり、ライブラリも充実している反面、多様なネットワーク構成、ハードウェア構成に対応するためにロボット体内で使用するにはパケットフォーマットやプロトコル処理に冗長な部分が生じてしまう。その結果、1kHz 以上の制御周期を実現でき

[4]https://www.iso.org/standard/20269.html

表 2.6: Layers of OSI reference model.

Layer 1	Physical Layer
Layer 2	Data Link Layer
Layer 3	Network Layer
Layer 4	Transport Layer
Layer 5	Session Layer
Layer 6	Presentation Layer
Layer 7	Application Layer

る通信低遅延性能や体内に収めるハードウェアのパッケージサイズが要求されるスペックを満たすまでには至っていない。

そうすると、ロボット体内の分散型制御システムに特化した通信系を構成するためのより単純化された通信モデルを作り出す必要がある。産業用 Ethernet 等はそうした要求から生み出された通信系であり、TCP/IP に頼らない通信によってオーバヘッドを削減することで、通信遅延の低減につなげている。本研究でも 2.8.1 節にて述べた、ロボット体内における通信系の制約条件と照らし合わせながら、分散型制御システムのための通信系の構成について検討をすることが望ましいと考える。

まず、物理層に関連する制約条件として「ノイズ耐性」「パッケージサイズ」「ケーブル」「省電力」が挙げられる。これらの信頼性に関する対策方法として、電磁ノイズの影響を受けない光通信による実装が一般的である。通常は光ファイバ通信網の構成には専用の光電変換モジュールが必要となり、パッケージサイズやケーブル取り回しに制約が生じていたが、近年は**光アクティブコネクタ**が開発され、組込み用途で利用可能となっており、ロボット体内用物理層に利用できないかと考えられる。また、光通信化は副次的に通信速度の大幅な向上が可能となり、従来の通信速度では困難であった通信手法を実用可能なレベルで実装できるのではないかと期待される。この光アクティブコネクタを用いた通信システム物理層の開発について、第 3 章及び第 4 章にてさらに詳細を述べる。

データリンク層、ネットワーク層、トランスポート層では、「接続構成」、「接続ノード数」や「接続トポロジ」、「ホットプラグ」が対応する。ロボット体内通信系ではノード間接続は静的に決定され、トポロジも単純なツリー型やライン型の組合せに対応されればで十分である。従って、IP のような複雑なネットワーク構成のための仕組みは省略され、これらの層構成におけるオーバーヘッドを削減することが可能となる。次に、分散型制御システムの場合、各ノード間で対称に行われることが望ましい。つまり、サーバクライアント型通信において、ある特定のサーバノードとクライアントノードに分かれるのではなく、どのノードもサーバノードでありクライアントノードにもなれることが要求される。これはどのノードの通信モジュールにおいてもデータリンク層からトランスポート層までは同一又は互換のものが用いられていることで達成される。また、通常は IP のプロトコルスタックはソフトウェア処理であるが、大幅に機能を削減されることで、**ハードウェアロジックのみでの実時間高速処理**が可能にならないかと考えられる。つまり、通常ソフトウェアによって多種多様なプロトコルに対応する部分について、ヒューマノイドロボット制御用に固定されたプロトコル処理に特化することで、オーバヘッドを削減する方法である。従って、データリンク層・ネットワーク層・トランスポート層は分散型制御システム上のハードウェアロジックでの実装を前提とした簡易な通信モデルを採用することを考える。第 4 章にて、詳細を述べていく。以上の実装方法を汎用的な Ethernet と比較すると、図 2.21 に示す通りになる。

セッション層からアプリケーション層においても可能な範囲でハードウェアロジック化による実時間高速処理性能を実現することで通信システムで一貫した低遅延性能を提供できるのではないかと考えられる。ロボットの体内で要求されるデータ通信では、コアとなる制御用データの通信においてはセッションの確立を必要としない。これは、ノードの接続構成が静的に決定することが可能であるためである。従って、セッション層は省略される。また、最終的にノード間で要求されるメッセージング形態としては、同期通信による制御レジスタへの読み書き(push, pull) あるいは割込みフラグを設定する Remote Procedure Call (RPC) 型通信が必要となる。また、目標関節角度等の毎制御周期送受信される制御コマンドを非同期でのデータストリーム型通信によって送受信する必要がある。制御周期を高速化し、送受信データ量が増大するにつれてデータストリーム処理は負荷が増大するため、ハードウェアロジックによる実時間での高速処理が

		Ethernet	Our Method
Layer 1	Physical	**Hardware (NIC)**	**Hardware (Opt. Fiber)**
Layer 2	Data Link		**Hardware (FPGA Logic)**
Layer 3	Network	Software	
Layer 4	Transport		
Layer 5	Session		-
Layer 6	Presentation		
Layer 7	Application		**Hardware (FPGA Logic) & Software**

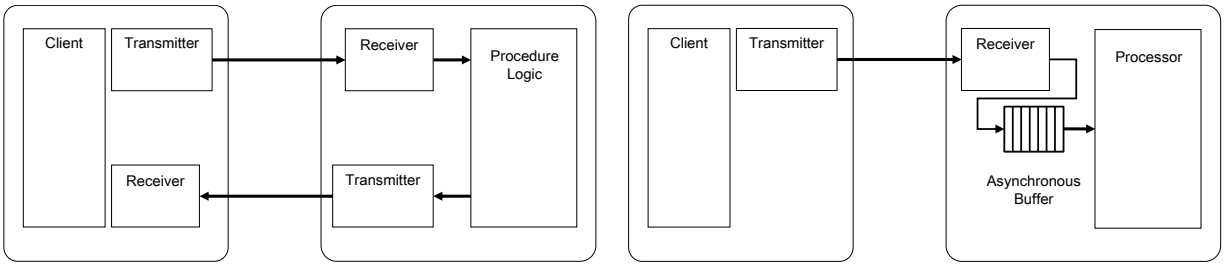
図 2.21: Comparison of communication system implementation between Ethernet and our method.

実現されれば、非力な組込み系においても実用化が可能となる。RPC 型通信とデータストリーム型通信の 2 種のメッセージングモデルを実現するプロトコルまでが、完全にハードウェアロジックによって実装される構成となる通信モデルを採用することを考える。この二種の通信モデルについて、表 2.7 に実装される機能と同期形態、接続形態についての仕様を示す。実際の実装については第 4 章にて、詳細を述べていく。

表 2.7: Messaging Types to use for communication model.

Messaging Type	Function	Synchronous	Connection Type
RPC	Push	Sync.・Async.	one-to-one
	Pull	Sync.	one-to-one
	Interrupt	Sync.・Async.	one-to-one
Data Stream	Message Queue	Async.	one-to-many many-to-many

2.8.3 同期・非同期通信



(a) Sync. (b) Async.

図 2.22: Synchronous communication model and Asynchronous communication model.

通信における同期は一般には、送信側が通信を開始してから返信が来るまで処理を待つことを指す。この例では返信が返ってくるリクエスト通信に限られるが、本研究では通信による処理の発生という観点に着目して「通信により運ばれたデータに基づいた処理が受信側で即座に発生する場合を**同期通信**、受信側の任意のタイミングで処理が発生する場合を**非同期通信**」と定義することにする。同期通信及び非同期通信

のモデル図を図 2.22 に示す。なお、物理層の文脈においては同期通信とは送信側と受信側でクロックが互いに同期していることを指すが本研究ではこの意味で用いることはない。

同期通信は、例えば即座に処理が行われることが望ましい制御目標値や、非常停止コマンド等を送信する場合に適当である。同期通信を実時間で処理するために通信系に組み込まれた処理ロジックが備えられている必要があると考える (図 2.22a)。本研究で用いるモータドライバにはハードウェアロジックとして実装された電流制御器とベクトル制御器が搭載されているため、プロセッサでの処理を待たずに即座に目標電流値指令を出力に反映することが可能である。このような専用ロジックとの通信においては同期通信は応答性の向上を達成する上で非常に優れていることがわかる。pull 型通信についても、専用ロジックで即座にレスポンスが返せる場合には、即値を取得することが可能であるため同期通信は有効である。反対に、受信側の制御サイクルに合わせて読み込みを行いたいセンサーデータ処理等は非同期通信により運ばれるのが望ましい。多種あるセンサーデータの受信ごとに処理を行っていては、処理の負荷が増大してしまうためである。特にプロセッサへの割込み処理を要する場合には実時間制御を達成する上で大きな障害となってしまう。また、非同期通信の場合には受信したデータが処理されるまでの間保持しておくための非同期バッファを備えておく必要がある (図 2.22b)。以上のような視点に基づいて、本研究ではロボットの体内通信系に RPC 型通信モデルとデータストリーム型通信モデルの2 つを基本モデルとして採用することにした。

2.8.4 接続トポロジ構成

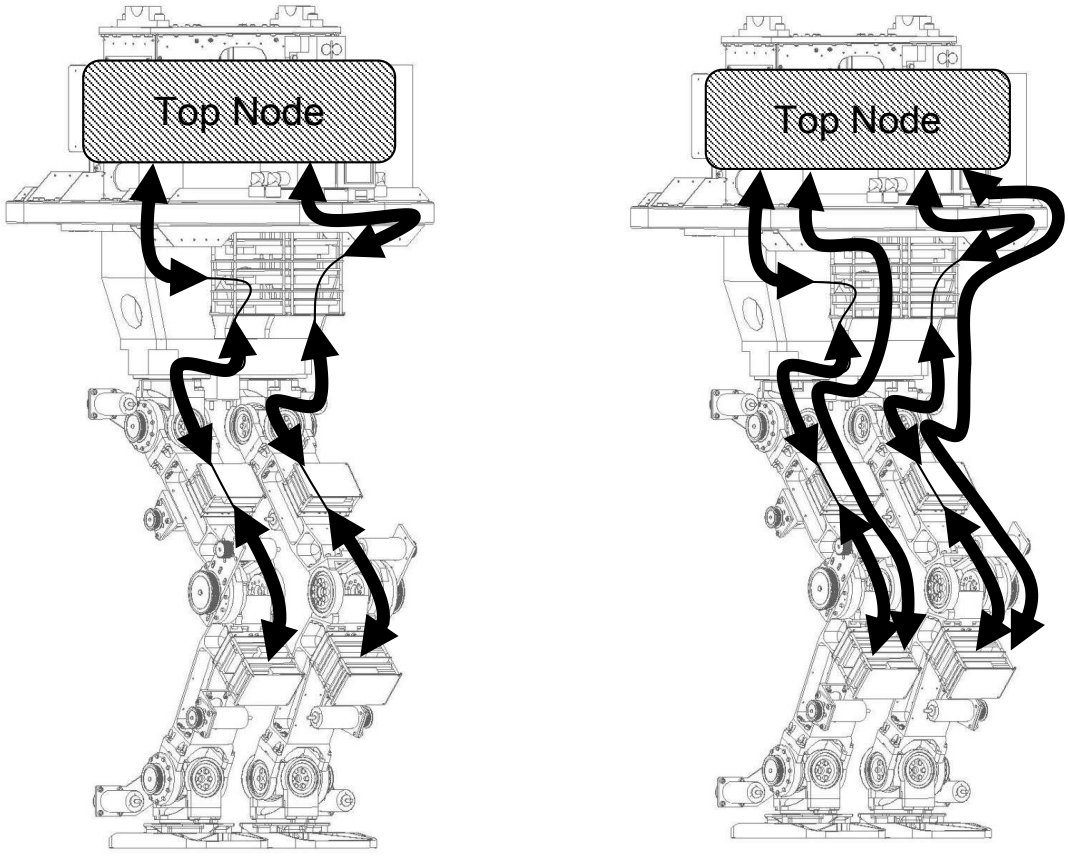
通信系の設計において、ネットワークの接続構成(トポロジ) はシステム全体の性能を決める上での重要な要素である。2 つのノード間のみで1 対1 の通信で完結する場合には接続構成は1 通りであるため物理構成やプロトコルもシンプルに構成される。しかし現実には分散型制御システムでは2 つ以上のマルチノード構成となり1 対1 、1 対多、多対多の接続に対応した構成にする必要がある。ノード間を通信リンクで接続する構成パターン(接続トポロジ) については、理想的には任意のトポロジ(フリートポロジ) で構成できることが望ましいが、物理構成やプロトコルが複雑化するため、データ転送にかかるオーバヘッドの増大によって通信遅延を招いてしまい、特にハードウェア資源の乏しく、実時間性能を要求されやすい組込みシステム等で簡単に使用することはできない。

トポロジ構成は、ネットワークの信頼性指標の一つである冗長性を高める上で考慮される必要がある。ロボット体内においては、体幹や頭に置かれた主計算機やハブをルートとしたツリートポロジやライントポロジ (図 2.23a) が採用されることが多い。これは、身体構造自体がそのような物理的構造になっていることから配線長の最適化において合理的であるが、一か所の断線による通信エラーによって、そこから先のノードは一切通信ができなくなる。ノード間の通信において、複数の経路でデータ伝送ができる冗長なトポロジ構成が可能な通信リンクは、信頼性の点で有利であると考えられる。ロボットの身体構造と親和性の高い、ツリー・ライントポロジを冗長化するためには、単純な方法として図 2.23b のようにリング化することが挙げられる。要求される信頼性とコスト制約に応じて、ツリー・ライントポロジとライン・リングトポロジを選択できると可用性が高まると言える。

2.8.5 実時間データフロー制御

通信リンクにはデータ転送帯域やリンク遅延といったハードウェア制約が存在する。また、多ノード間でデータ転送を行う場合に、データリンク層に相当するレイヤーにおいてデータフロー制御について取り決めておかなければ、正しくメッセージを転送することは難しい。特に実時間性が要求されるデータについては、データの転送タイミングが制御可能であることが望ましい。従って、通信系全体としてデータの送受信が管理されるための仕組みが必要となる。通信系の制御においてもプロセッサにおけるプロセスと同様にスケジューリングによって管理することが考えられ、山崎らは通信モデルにプロセッサにおけるスケジューリングアルゴリズムを適用することを考えた *Responsive Link*[30] を提唱している。実時間OS においては、スレッドの優先度と最悪実行時間をパラメータとして、物理プロセッサへの時間軸方向への割り振りについて調整された。通信系においても、転送するデータパケットの優先度、最悪転送遅延時間に応じて、通信リンク使用権を付与するという問題として、同様の手法が活用できると考えられる。

まずは、時分割ポリシに対応するタイムトリガー型 [31]、イベントトリガー型 [32] が考えられており、通信系の実時間性や低遅延性を実現するために産業用通信リンクを始めとした様々な規格において、各自の



(a) Tree and Line topology connection.

(b) Ring topology connection.

図 2.23: Comparison of traditional line topology connection and ring topology.

実装が成されている。

本研究では、低遅延性や実時間性を備えた通信リンクをロボットの体内分散型制御システムのネットワークに適用することを目指す上で、2.8.1 節にて述べた制約であるロボット体内ではツリートポロジやラントポロジ、リングトポロジといった単純な接続構成で十分な点を踏まえて、優先度重みづけによるデータフロー制御を実装する。詳細については、第 4 章にて述べる。

2.8.6 通信品質とデータ符号化とデータレート

通信品質評価指標

通信品質の評価指標として、以下の3 項目を挙げる。

- スループット性能
- 遅延性能
- データ損失率

スループット性能は、ある通信系が一定の時間内にどの程度の量のデータを送受信可能かを示す指標となり、Byte per sec 単位で一般に示される。遅延性能は、送信されたデータが受信側で受け取るまでにかかる時間を指す。データ損失率は、ある単位当たりの送信されたデータについてそれが正しく受信される確率で示す。

これらを決定づける要因として、通信系に接続されているノード数、トラフィック量、データレート、ビット損失率 (Bit Error Rate, BER)、ノイズ耐性、エンコード・デコード処理遅延、ネットワークトポロジ、フロー制御、輻輳制御、プロトコルが考えられる。

通常インターネットのような汎用的なネットワークでは、まずデータが正確に送受信されることを優先するため、例えば TCP を用いたネットワークによって通信の信頼性を向上させる。これは、コネクション型通信によるエラー発生時の再送をプロトコルレベルでサポートし、さらにフロー制御、輻輳制御によってネットワーク側の受付可能バッファサイズ(受信ウィンドウサイズ) を教えあうことで、ネットワーク側トラフィックを考慮したデータ送信量に制御することで、通信全体でのデータ損失率を低減し、信頼性の高い確実な通信を実現している。このような手法は、インターネットのようにネットワークの中継点の構成やトラフィック情報が送信側・受信側では関知できないようなネットワークでは重要な方式となる。しかしながら、データの確実性を重視するための処理内容がオーバーヘッドとして重畳してしまうため、スループット性能や遅延性能についてはネットワークが持っている本来の性能に至らないという問題が生じる。このような一般的なインターネット回線では、回線の帯域や混雑具合によってパケットの転送にかかる時間は際限なく遅延し得る**ベストエフォート方式**で提供される。しかしながら、ロボットの制御では、遅延時間は実時間性の観点から最も要求の高い項目であり、汎用的な通信回線では達成不可能なことが分かる。

専用線等によりスループット性能、遅延性能を保証した通信系をベストエフォート方式に対して**ギャランティ方式**と一般に呼ぶ。ロボットの体内通信系を構成する場合、ネットワークを構成する中継ノードの状態については、事前に設計時点で考慮することが可能であるため、フロー制御や輻輳制御を大幅に簡略化することが可能であると考えられる。このように、特殊な用途に限定した通信系では、スループット性能や遅延性能についても最悪値を保証するために通信系の品質 (Quality of Service、QoS) を定義し、それを実現するためのハードウェア構成、プロトコルを採用することが必要である。

なお、インターネット回線のようなベストエフォート方式のネットワークにて、ある程度の実時間性を実現するための手法として、動画のストリーミング等で用いられる Multiple Desctiption Coding (MDC) がある。MDC はビットストーリームを多重化し、多数の冗長経路を経由するよう送信される。多重化されたパケットで送られるため、一部のパケットロストやデータ破損が発生した場合でも残りのパケットからある程度の品質のデータを復元することが可能となっている。これによって、受信側はネットワーク品質に応じた平均的なデータ品質のサービスを受けることができる。動画配信サービスのように経由ネットワーク品質が不確定な場合でも、サービス受益者側で一定の品質の動画を視聴することが可能となる。これはデータ損失率を犠牲に、実時間性・データ到達性を向上させていると考えることもできる。MDC については、ヒューマノイドロボットの体内通信システムにおいては利用シーンは少ないと思われる。ただ、体外との通信においてはヒューマノイドロボットは基本的には無線通信系となるため、ベストエフォート方式にならざるを得ない。そのため、体外との実時間性要求のあるデータ(例えば、テレメトリデータやカメラ画像のストリーミング) のやり取りについて、MDC が有効ではないかと考えている。

前方誤り訂正符号

コネクション型プロトコルによるエラー時の再送に依存しない形でのデータ損失の抑制方法として、**前方誤り訂正符号** (FEC) による通信が用いられる。この FEC は、エラー時の再送を行わない実時間通信系や BER の大きい無線通信系において通信品質の向上を目的に用いられる。

簡易な FEC として、ハミング符号化が用いられる。演算は比較的軽量であるが、多ビットの誤り時のエラー訂正能力については低いため、BER が低いネットワークでの限定的な使用のみとなっている。BER の悪い通信では、よりエラー訂正能力の高い Reed-Solomon ブロック符号やターボ符号、低密度パリティ検査符号 (Low Density Parity Check Code、LDPC) が用いられる。これらはシャノン限界に近いエラー訂正能力を持つため、非常に信頼性の高い通信品質を提供することが可能となるが、デコード処理が複雑であるためどうしても処理によるオーバーヘッドが大きくなってしまうという問題が生じる。オーバーヘッドによる遅延が問題となりにくい、3G/4G 等の無線通信系で用いられている。本研究の対象とするヒューマノイドロボットにおいても、5.2.2 節にて述べる先行研究で使用されていた JAXON や HRP3L-JSK で使用されてきた通信系では、分散制御基板間を RS422 ベースのディジーチェーン接続によって構成されており、Reed-Solomon ブロック符号を用いることでロボット体内の高強度ノイズ環境内においても高い通信信頼性を確保していた。しかしながら、Reed-Solomon ブロック符号のエンコード・デコードは各ノード間でデータを転送するたびに行われ、そのたびに $10\mu sec$ の遅延が生じていた。接続デバイス数が増加するに

つれ、レイテンシが積み重なっていくためシステム応答速度の悪化につながる。10kHz 100μsec 周期でのデータサンプリングを行い、衝撃応答性能の向上を目指す本システムにおいては 10μsec の遅延は許容できない。従って、根本的な解決として、通信系の BER が簡易なハミング符号で実用レベルで運用可能な程度まで改善されることが望ましいと考える。なお、10GigE, 100GigE では LDPC をベースとしたより低遅延でエラー訂正能力の高い手法が研究されているが、本研究のターゲットを超える領域となるため比較対象とはしない。

通信媒体とビット損失率

通信媒体の信頼性を評価する上で重要となるビット損失率は、一般的な所のイーサネットの 1000BASE-T 等の規格である IEEE802.3[33] では、BER について 1e-10 以下の品質を要求している。このように概ね 1e-9 から 1e-10 以下の BER を品質として確保している場合には、十分に信頼できる通信を行うことが可能である。光ファイバ通信系は Rjpp(Random Jitter Peak-Peak) が BER 10e-12 相当を公称値としており、非常に信頼の高い通信を行うことが可能となっている。シングルワイヤの簡易な通信系では、電磁ノイズ環境下では BER が 10e-6 から 10e-1 程度と大幅に悪化することも報告されている (Choi ら、[34])。また筆者らが I^2C や CAN 等の汎用的なバス通信規格について、ロボット体内のノイズ環境下で実際に使用した際に数百フレームに一度はエラーが生じることが確認され、実用レベルには及ばないことが分かった。CAN については本来ノイズに強い規格であるはずであるが、ロボット体内の 30dBw を超える強度の電磁ノイズ下ではコモンモードノイズによって通信不良を引き起こしていることが確認され、アイソレータを使用することでコモンモードノイズの影響を低減し、ある程度改善された。前述の FEC はこのような BER の悪い簡易通信媒体において実質 BER を 1e-9 以下に引き上げるものである。

2.9 高速応答を実現するための体内分散制御システム構成

2.9.1 演算負荷と応答性に応じたサブシステム分割

前節までの論旨を踏まえ、本研究で提案する体内分散制御システムの各ノード構成について検討していく。

2.3 節にて論じたように、ロボットが外乱を知覚しアクチュエータ出力に反映するまでの遅延時間を 100μ 秒オーダまで低減することで応答性を高めることが可能となる。しかしながら、2.6 節にて述べたように汎用的な計算器システムでは μ 秒オーダでの実時間実行能力を持たせるためには OS の介在による遅延の存在を極力排除する必要がある。一方でロボット制御では運動学計算や認識・計画計算等の処理それぞれにおいて高い演算能力を要求するため、x86 アーキテクチャに代表されるクロック周波数の高いプロセッサを採用する必要があるが、これらの利用において OS の介在は事実上不可欠である。リアルタイム OS 等により μ 秒オーダでの応答を実現することも可能ではあるが、演算能力とのトレードオフになるため、応答性は限定的となってしまう。以上の問題から、ヒューマノイドロボットの制御系においては主に演算性能に特化した上位システムと応答性に特化した下位組込みシステムとして実装される分散制御システムであることを 2.2 節にて述べた。

本研究では、この上位下位の階層構造の中間にもう一層中間的な性能の制御システムを組み込んだ 3 階層制御システムとして体内分散制御システム全体の構成を第 5 章にて行う。これは、近年の組込み用プロセッサの性能向上に伴って、スマートフォンに代表されるような中規模計算器システムを小サイズパッケージに収めることが容易となり、同時にインターフェース処理を FPGA 等によりプログラマブルなハードウェアによって行うことが可能となったことが背景としてある。この中間層システムの存在によって、システムは従来組込みプロセッサでは困難であったロボットの運動学計算に基づくフィードバック制御を μ 秒オーダで実行することが可能となる。この 3 階層制御システム構成について表 2.8 に示す。この構成では、従来システムにおいて課題であったセンサ用インターフェースの集約化問題も解決するために、下位層にて直接各種センサを扱うためのセンサドライバを備えることを検討する。これは、従来体内に物理的に分散していた各種センサのインターフェースが中央の上位層計算機に集まっていたために、配線が複雑化していた問題であり、各下位層モータドライバがデバイスインターフェースを備えることでこの配線問題を解決するものである。

表 2.8: Explanation of three sub-layer control system.

Layer	CPU Frequency	Computer Load	Realtime Response	main component
Upper	Over 3.0GHz	High	1-100[msec]	x86 Architecture
Middle	200MHZ-1.0GHz	Middle	100-1000[μsec]	FPGA and ARM Cortex-A series equivalent processor
Lower	Under 100MHz	Low	1nsec-100μsec	FPGA and embedded processor, Sensor Drivers

2.9.2 設計要件に基づいた分散ノード計算機アーキテクチャ

上位層を構成する計算機アーキテクチャは現状ではデスクトップ向けの x86 系 CPU にオプションで GPGPU を搭載する形が基本となる。据え置き型のロボットにおいては、サーバー向けのメニーコアアーキテクチャ製品を採用する場合も見られるが、消費電力やパッケージサイズの観点からヒューマノイドロボットにおいては採用することは困難である。

下位層の分散ノードにおいては、基板全体のトータルパッケージサイズの小ささが要求仕様として大きい。そのため主要構成要素となるプロセッサ部は System on Chip(SoC) のような小パッケージ製品から選定することが望ましいと考える。下位層の計算機アーキテクチャは (i) 組込み用 SoC のみ、(ii) 組込み用 SoC+FPGA のヘテロジニアス構成、(iii)FPGA のみの3 パターンが考えられる。ただし、2.8 節にて述べた通信系の制約によって、本研究では **FPGA による独自通信機構**の実装を行う(第 4 章)。これは、汎用組込みプロセッサに一般的に搭載されているインターフェース規格では上記通信系制約を満足することは困難であるからである。そのため、(ii) または (iii) の構成が選択肢となるが、本研究では様々な用途への拡張性を考慮して (ii) のヘテロジニアス構成をモータドライバ基本アーキテクチャとして採用し、第 3 章にて開発を行う。

仮に組込みプロセッサのみで下位層のモータドライバノードを構成する (i) の場合、以下のような点が利点となる。

- 標準的なソフトウェア開発環境が利用可能で開発難度は低い
- ソフトウェアによる柔軟で効率的な開発が可能
- SoC 製品の採用で1 チップのみで必要機能を構築可能
- FPGA より CPU クロック周波数を高速化可能
- 低価格で実装が可能
- 内蔵 RAM が FPGA より大きい SoC が選択可能

一方で欠点としては、

- 必要インターフェースが制約となり製品選択肢が限定的
- 機能拡張性は限定的
  - 処理頻度の増大とともに応答性低減
  - インターフェース種別やポート数に制約
- シングルコアプロセッサでは割込み頻度の高いモータ制御の高速化に限界

が挙げられる。インターフェースに関係する制約は特に多数のセンサデバイスを接続する必要があるヒューマノイドロボットの制御システムにおいては著しく問題となる。

FPGA のみの構成とした (iii) の場合、以下のような点が利点となる。

- 処理を完全並列化することで、応答を高速化・Deterministic にすることが可能
- ソフトウェア CPU コアを合成することでソフトウェア的処理に対応可能
  - マルチコア構成を採ることも可能（ OS レベルの対応は限定的）
  - 割込み信号の割り振りを任意に設定可能（ 割込み処理の自由度が高い）
- インターフェース種別・ポート数・チャンネル数を任意に設計可能
- チップ間データ転送に伴うオーバヘッド無し
- 低消費電力

一方で欠点は以下が挙げられる。

- ソフトウェア開発難度が高い
- 最大 CPU クロック周波数は 100-400MHz
- 内蔵 RAM は高々数十 KByte 程度
- 十分な性能の FPGA は高価格になりがち
- USB や CAN 等の電気的信号が特殊なインターフェースは専用 PHY が別途必要

本研究で開発するモータドライバの基本アーキテクチャとなる (ii) のヘテロジニアス構成の場合、以下の点が利点として考えられる。

- ソフトウェアとハードウェアによる柔軟な開発が可能
- SoC インターフェースの制約が緩和されるためプロセッサ部の性能の選択肢が増える
- FPGA による拡張性の高いインターフェースをソフトウェアで利用可能
  - 組込み向け SoC で採用例の少ない高速シリアルインターフェースを利用可能
  - センサ関連処理を FPGA 側で並列処理させることでプロセッサ側負荷の低減が可能
- 電流制御等の固定化された処理については FPGA で処理することでプロセッサでは困難な高制御周期にも完全並列処理で対応可能
- SoC 側の RAM を FPGA 側から利用可能
- FPGA 規模を抑えられるため比較的低価格

一方欠点としては、以下の点が挙げられる。

- チップ間データ転送にオーバヘッドが存在
  - PCI Express 等の高速インターコネクト規格の採用でオーバヘッドを低減
  - DMA 転送の活用によりオーバヘッドを低減
- マルチコア構成は限定的
- 低消費電力化に限界
- 必要基板面積が増大

ヘテロジニアス構成の欠点の中で物理的な設計要件に関わる基板面積増大に関しては、下位モータドライバの構成において看過できない課題であるが、基板実装としてスタック式を採用する等の工夫によって占有体積に対する基板面積を拡大する等の工夫によって許容範囲に抑えることができれば、実用上は問題が無いと考える。

本研究では設計要件の中で可能な限りで小パッケージ化を進めつつ、一方で拡張性や過去のシステムとの互換性を維持するためにヘテロジニアス構成を採用した。しかし、将来的にモータドライバにて処理するソフトウェアが固まり、ハードウェアロジックとして最適化して実装する段階を進めた場合には、完全に FPGA のみあるいは専用 ASIC の開発によって下位層モータドライバノードを最小パッケージ構成とすることを目指す。

2.9.3 体内分散制御システム構成の将来展望

前節までの議論は現状の組込み製品とヒューマノイドロボットに要求される制御性能に基づいたものである。ここでは、将来的にさらに組込みシステム製品の性能が高まった場合にヒューマノイドロボットのシステム構成としてどのような構成が考えられるか論じていく。2.9.1 節にて論じたように、従来のヒューマノイドロボット制御システム構成は多くの場合、中央の演算性能の高い計算機において全ての身体運動を含めた制御指令を決定し、分散ノードにてローカルなアクチュエータ制御とセンサ処理を行うに留まっていた。しかし、ロボット制御演算自体の処理内容の増大と高速応答性の要求はさらに高まることが予想される。演算処理の増大に対しては近年はメニーコアアーキテクチャや GPGPU のような多数の演算器を並列に実行するための機構が AI 開発の加速に伴って多数発表されている。特に Deep Learning に代表されるデータ学習の活用をターゲットとした畳み込み演算の高速化に特化した専用アーキテクチャが開発されており、ヒューマノイドロボット用システムへの適用も今後進むものと思われる。これらは上位層システムにおける認識・計画・学習系処理において効果的であると考えられるが、制御応答性が要求されるバランス関連や力制御関連の制御系演算においてもメニーコア化の恩恵はあると考えられる。例えば実時間要求の高い処理については、専用コアを割り当てることによって OS のタスクスケジューリングによる割込みの影響を排除することが可能となる点が挙げられる。現状のシステムでは 10 コアにも満たない数であるため、演算負荷の高い上位層において専用コアを割り当てるのは全体の演算能力の低下に繋がるが、十分なコア数が確保できる場合には専用コア割り当て手法は効果的である。

ただし、メモリバスや各種インターフェースについてはメニーコアアーキテクチャにおいても限定的であるため、システム構成によってはボトルネックとなる可能性がある。実際、GPGPU 等を利用したアプリケーションにおいてもメモリ転送はボトルネックとなるため、実時間制御のような高頻度でメモリや I/O へのアクセスが発生する処理は実時間処理に特化したアーキテクチャにて行うのが理想的であると言える。また、メニーコアアーキテクチャにおいて各コア構成は対称であるため、ソフトウェア開発的には効率的であるが、消費電力の観点からは無駄の生じやすい構成であると言える。従って、上位層においてメニーコアアーキテクチャの採用が一般的となった場合においても、中間層に位置するサブシステムを構成し、インターフェース処理の応答性要求が高く、演算負荷の比較的小さい運動学レベルの制御演算を行う構成は有効であると考えられる。

中間層サブシステムの構成についても、将来的にはプロセッサのマルチコア化が予測される。中間層ノードを複数並列に接続するマルチノード構成とすることで、ロボットの身体構成に合わせてシステム側を拡張することが可能となるため、本研究において開発する中間層ノードもマルチノード化に対応する。このマルチノード構成では、例えば一つのノードに一つの四肢を担当させるような使い方をする場合には、処理内容は各ノードにおいて対称であり、ノード間で共有する必要のある情報も少ない。しかし、歩行軌道生成や姿勢安定化制御等を中間層サブシステムにて行うために制御情報をノード間で情報を共有する必要がある場合には、GPGPU におけるメモリ転送ボトルネックと同様に通信速度がボトルネックとなる可能性が存在する。そのため、本研究の時点では体内分散通信系はノード間で対称な構成であるが、将来的には通信系に制御情報共有専用線を拡張する等の非対称性を持たせることで通信のボトルネック問題に対応する必要が生じると考える。

2.10 本章のまとめ

本章ではロボットの制御システムの一般的な構成論について述べ、プロセッサ性能の向上やロボットの多自由度化、ロボットのトータルパッケージデザインの必要性を背景に、体内制御システムにおいても合理的な実装手法を採り入れることについて示した。また、実時間性・低遅延性がロボットの動作における信頼性の向上に必要であり、上位計算機システムではこうした実時間性を高めるためのシステムデザインが発展してきていることについて述べた。実時間性・低遅延性は体内の下層制御システムのハードウェアデザインと密接に関係してくることから、体内制御システムのコントローラ・通信リンクについて実時間・低遅延を意識した設計を採り入れることについて述べた。

第3章

分散型センサを接続可能な実時間ベクトル制御モータドライバの開発

3.1 分散型制御システムにおける組み込みプロセッサ

3.1.1 既存 CPU アーキテクチャでの制御器の実装

従来ロボットの主制御器として使用されている x86 アーキテクチャをベースにした汎用 CPU 及び OS では、豊富なライブラリ群やデバイスドライバがあることによる実装の効率化、非常に高い CPU の演算能力やメモリ、ストレージサイズといった潤沢な計算機資源による演算制約の低さによって計算負荷の高い高度な制御演算でも実現することが可能となっている。逆にこれらの高い演算性能を十分に活かすためには、演算規模に見合った大規模なメモリやストレージ、十分な品質の電源と冷却機能の実装が不可欠となるため、システムの規模は一般に拡大しやすい傾向にある。市販されていて入手性の高いマザーボードでは Intel の NUC が 10cm 角サイズと、ヒューマノイドロボットに搭載するのに有利なパッケージデザインとなっている。また、汎用 CPU を搭載した計算機は元々組み込み用途に向けたデザインでは無いため、ロボット体内のような高温･ノイズ環境下での動作、ロボットの歩行時の振動･衝撃等にも特別な対策を講じる必要がある。産業用に素子の耐熱性や消費電力等が組込み用途に合わせた高水準な CPU パッケージも存在し、Computer on Module(COM) といった計算機システムとしての機能を一つのモジュールにパッケージングするというアイデアのもと、COM Express を始めとした規格が登場している。USB コネクタやメモリ･ PCI スロットは強い振動･衝撃に対して十分に接続性を保証できる構造にはなっていない。そのため、歩行や転倒時に最悪システムのシャットダウンを引き起こしてしまう恐れがある。コネクタの形状やサイズ、向きに関してもロボットへの搭載に適するようデザインされているわけではないため配線の取り回しの最適化が困難であり、サイズ･重量の制約が厳しいロボットへの搭載には適さない。表 3.1 に汎用計算機と組込み計算機とのロボットへのハードウェア的な実装上の比較を示す。

表 3.1: ハードウェア実装の観点から見た汎用計算機と組込み計算機の比較

計算機	実装サイズ	消費電力	I/O	実装自由度
汎用	大規模	大 (40W〜1kW)	PCI/Ether/USB	市販製品をほぼ流用 または チップセット制約の 下で設計
組込み	小規模	小 (50mW〜5W)	UART/USB/Ether GPIO/Parallel Bus	ロボットに適した パッケージで設計可 能

ロボットの制御の観点では、汎用の計算機では一般にモータやセンサを扱うためのインターフェースは搭載されていないため、拡張デバイスによって追加する必要がある。Ether,USB あたりが一般に搭載されているインターフェースであり、これらのインターフェースを備えたスレーブ型モータドライバ基板･センサ基板と接続することでロボット制御システムを構成する。また、大規模なロボットシステムでは PCI/PCIe に拡張基板を追加することによって直接モータドライバや ADC,DAC をパッケージ内に搭載することを行う。

3.1.2 組込みプロセッサにおけるロボット制御演算性能

古くから、簡単なモータ制御やセンサ情報処理、通信処理については組込み向けのプロセッサであるマイコンで行われてきた。マイコンはクロック周波数が汎用計算機向けのプロセッサと比較して小さく、大規模な処理は全て汎用のＰＣ側で行うのが当たり前であった。

しかし近年はスマートフォンに代表されるような、組み込み用途のプロセッサにおいてもクロック数は 1GHz 以上で動作し、メモリサイズも 1GB 以上を搭載しつつもバッテリ駆動によって 10 時間以上動作し続けることが可能なほどに低消費電力な実装になることが当たり前の状況となってきている。組込み用途でマルチコアプロセッサが登場し、実用レベルで軽量な組込み用ＯＳの実装も多数登場しているため、マルチスレッドが簡単に利用可能な環境になりつつある。

ロボット制御においてはマルチコアプロセッサ、マルチプロセッサ環境においては、例えば順運動学計算や逆運動学計算、関節角軌道の補間処理等で関節ごとにプロセッサを割り当てて並列計算を行うことで演算処理のスループットを向上させる手法は数多く研究されている。組込みプロセッサの処理能力の向上を背景に、分散型制御システムではこうしたマルチプロセッサ環境をネットワークによってシームレスに接続することで、従来は組込みシステムでは不可能であった処理をロボットの体内システムにおいて実現できるのではないかと期待されている。

分散システムとしての機能向上を目指した組込みプロセッサとして RMTP を次節で採り上げ、本研究において目指す自律型体内分散制御システムの各ノードにおけるコアコントローラとして活用することを考える。

3.1.3 既存組込み用 CPU アーキテクチャにおける実時間プロセッシング

ARM Cortex-R 等の実時間プロセッシングを特徴とした組込み向けＣＰＵアーキテクチャが実時間組込みシステム用に展開されている。これらのアーキテクチャで使われている “実時間性” はあるタスクの**実行時間が確定的 (Deterministic)** であることを意味している [35]。そのためにキャッシュや MMU による物理アドレスと仮想アドレスの変化といった応答までのクロックサイクルが不確定になるユニットは排除されている [36]。また低レイテンシでのメモリアクセスを実現するための Tightly Coupled Memory(TCM) という特別なメモリユニットを搭載している [35]。割込みについても、低レイテンシな割込み機構により Deterministic なタスク処理を実現している。また、μT-Kernel 等の組込み向け実時間ＯＳを利用することで、ベアメタル実装と比較してスループット性能は落ちるが、簡単に実時間タスクを開発することが可能であり、多くの組込みシステムで採用されている。

3.2 Dependable Responsive Multi-Threaded Processor (D-RMTP) の概要

Dependable Responsive Multi-Threaded Processor(D-RMTP) は山崎・水頭ら [37, 21] によって開発された分散型実時間システムを実現することを目的に設計された組み込み用プロセッサである。特徴としては実時間処理のためのマルチスレッド制御機能と、さらに分散ノード間での通信に実時間性を実現する *Responsive Link* を内蔵していることにある。D-RMTP チップ内のコアプロセッサとなる Responsive Multithreaded Processing Unit(RMT PU) は8 つの論理コアを内蔵しており、8 つのスレッドを並列的に同時実行可能なアーキテクチャとなっている。8 つのスレッドはそれぞれ並列に命令発行ユニットにて各スレッドの命令をデコードし、解釈結果に応じて命令演算ユニットへの演算命令を供給(dispatch) する。命令演算ユニットは表 3.2 に示すような個数用意されており、複数のスレッドから同時に同一演算ユニットへの実行命令が供給された場合競合が発生するが、RMT PU ではスレッドに設定した優先度によって、実時間要求の高い処理に対して優先的に演算ユニット資源を割り当てて命令発行(issue) することが可能となっている。演算器は 8 スレッド全てに同時に割り当てるほどの数は用意されていないため、優先度に応じて使用権を割り当てられるとはいえ浮動小数点演算のような処理の長い演算についてはそもそも使用するタスク数を限定しておくことでパフォーマンスを維持することに留意しておく必要があると考えられる。また多数の処理を並行に走らせる場合、タスク切替時に生じるコンテキストスイッチにおいてレジスタ等の情報を低遅延で退避するための専用のコンテキストキャッシュ機構を備えている。これらの特徴により、従来ソフトウェアレベルで制御されていたマルチスレッドの実時間実行について大幅にオーバヘッドが削減されるため、スレッドの短時間周期実行、低ジッタ特性といった高品質な実時間性能を得ることが可能となっている。図 3.2 に、RMT PU における優先度付きの8 スレッド並列実行の例を示す。また周辺機能として,DDR SDRAM I/Fs, DMAC や UART, パルスカウンタ等のインターフェース機能も LSI 内に組み込まれており、System-on-a-Chip (SoC) として 1 チップでロボットのセンシング・アクチュエータ制御が実現できるようデザインされている。

以上の優先度付きスレッド実行機構の処理は全てハードウェアレベルで処理される。従来の X86 アーキテクチャの計算機や ARM アーキテクチャのマイコンではこうしたマルチスレッド実行機能や優先度に応じたスレッドの割り当てはソフトウェアレベルで管理されている。

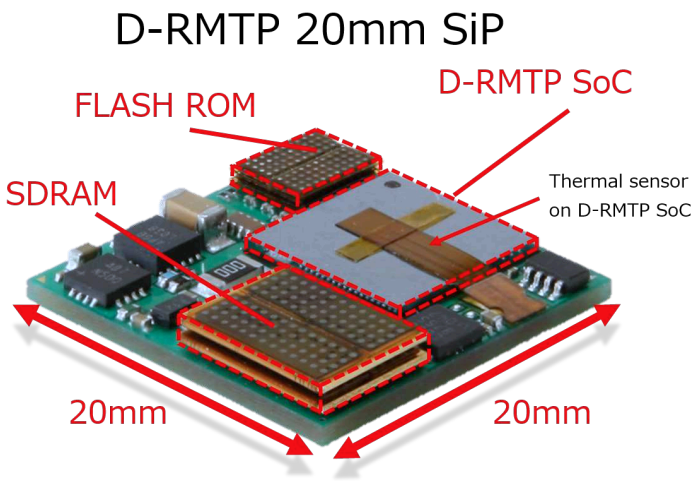


図 3.1: Dependable Responsive Multi-threaded Processor(D-RMTP) (photograph from [21])

表 3.2: RMTPU の持つ演算ユニット (from [38])

ユニット名	個数
整数演算器	4
整数演算器 (divider)	1
浮動小数点演算器	2
浮動小数点演算器 (divider)	1
64bit 整数演算器	1
整数ベクトル演算器	1
浮動小数点ベクトル演算器	1
分岐ユニット	2
メモリアクセスユニット	1
同期ユニット	1

3.2.1 RMTPU における実時間プロセッシング

上述の通り RMT PU ではコンテキストキャッシュを持ち、通常の CPU アーキテクチャでは大きなオーバヘッドとなるコンテキストスイッチによる遅延を大幅に短縮している。また、8 スレッドの同時実行が可能なためスケジューラによるコンテキストスイッチや割込みによるスレッドの一時停止を行わずに、処理を続行させることも可能となっている。そのため、モータ制御タスクやセンサデータ通信タスクといった実時間性の要求が高いタスクについてはスケジューラを関与させず、スレッドを一つ占有させてしまうという実装も可能である。

D-RMTP ではこのような実装を Responsive Task[39, 40] と呼んでおり、ユーザが使用するための API が OS[41] によって提供されている。

また、実際にはプロセッサ上で独立したタスクとして処理したいタスクはロボットシステムでは、補間計算、エラー判定処理、モータ温度推定、外乱推定、等数多く存在するため、有限のスレッドを割り当てるためにスケジューラによって実行タイミングを管理される必要がある。スケジューラによるタスクへのスレッド割り当てでは、一般的なＯＳのスケジューラのように優先度に応じて順に割り当てる固定優先度方式の他、最悪実行時間ベースでスケジューリングされる Earliest Deadline First(EDF) 方式 [42]、レートモノトニック方式 [43] が提供されている。

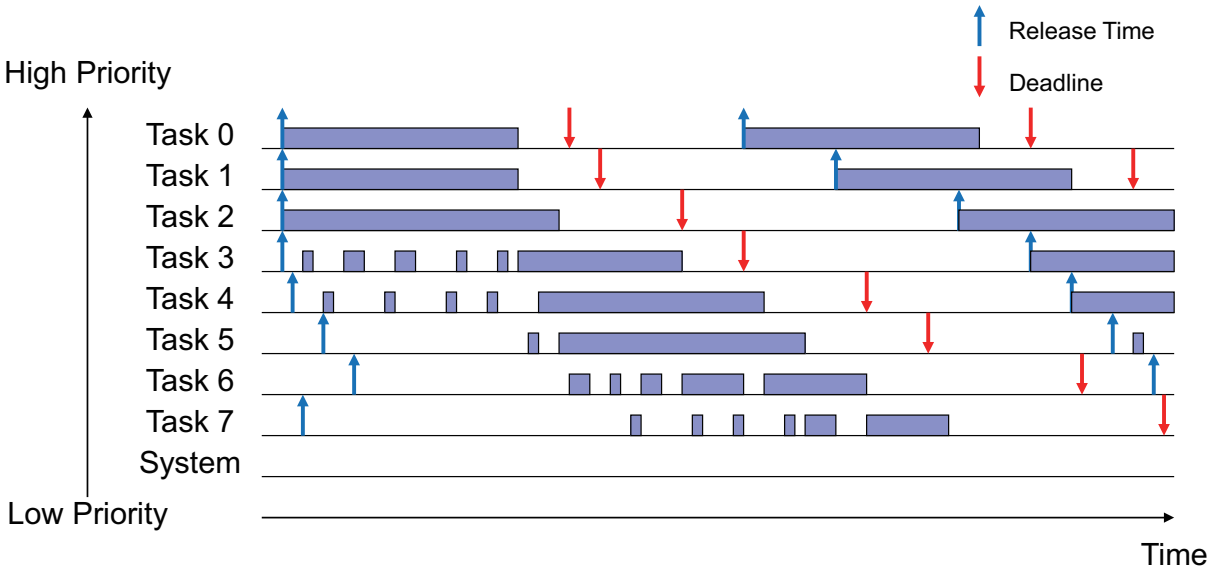


図 3.2: Real-time execution on RMT PU(from [37]).

3.2.2 Responsive Task

Responsive Task[39, 40] は Watanabe、Mizotani らによって提案された低遅延実時間実行機構である。Responsive Task では RMT PU 内の 8 つの論理コアの内の1 つを、1 タスクで占有させ、論理コア内に備えられたハードウェアタイマ割込みベースでの周期実行を行う (図 3.3)。これは通常のソフトウェアベースによるリアルタイムスケジューラの使用によって発生する、実行タスクのリリース、スケジューラ処理、次タスクへのコンテキストスイッチといったオーバヘッドを完全に削減する。これにより、Responsive Task では通常のソフトウェアスケジューラによる実時間マルチタスク実行と比較して短い周期で、低ジッタ性能の実現が可能となる。なお本研究では、ソフトウェアスケジューラによる実時間マルチタスク実行機構を特に Real-Time Task と呼称する。

Watanabe らによれば Responsive Task のオーバヘッドはタスク数に依存せず 1μsec であるのに対して、Real-Time Task ではタスク数に依存して 13μsec 以上かかる。また、Responsive Task の release jitter は、簡易な演算タスクで平均 0.04%, 最大 1.8%、配列操作による多数のメモリアクセスを伴うタスクで平均 0.4%、最大 3.2%と 報告されており、10μsec 単位での周期タスク実行が可能となる (図 3.4)。そのため μsec の精度を 必要とする電流制御や衝撃センサ応答処理のような極めて実時間性の要求が高い処理についても、プロセッサ上で走らせるプログラムから制御が可能になると考えられる。

なお、Responsive Task は論理コアを占有するため、並列実行可能なタスク数は最大でも物理的な論理コア数に制限されてしまう。また、演算ユニット資源数は限られているため、Responsive Task であってもスレッド間で競合が発生した場合にはスレッド優先度に応じて資源を割り当てることになるため、より実時間性の要求の高いタスクに高優先度を与える必要がある。

3.2.3 20mmSiP を搭載した拡張コネクタ付き基板

D-RMTP では電源や SDRAM,FLASH-ROM 等の組込みシステムとして必要になる周辺デバイスを 1 つのパッケージにまとめた System in Package(SiP) が開発されている。さらに D-RMTP SiP を様々な追加デバイスと接続して機能拡張を行えるようスタッカブルな構成として、D-RMTP 基板が開発された (図

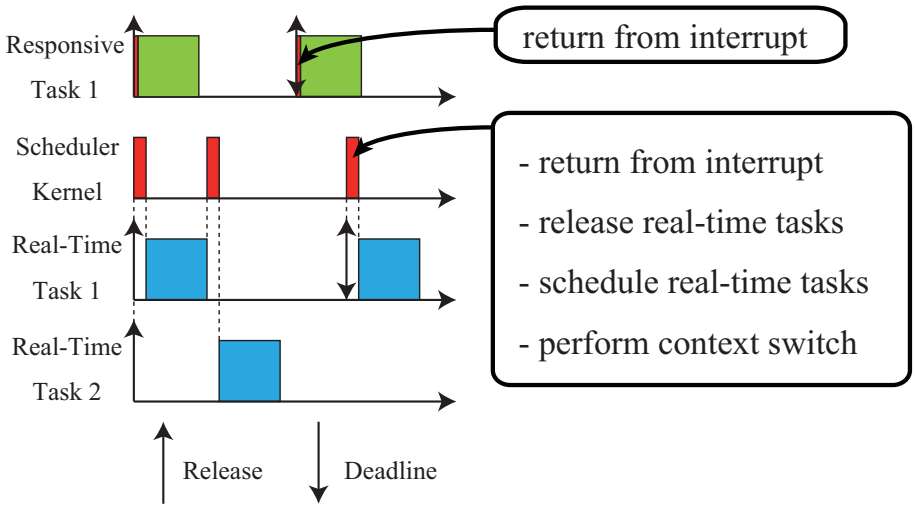


図 3.3: Comparison of Responsive Task and Real-Time Task. Responsive Task occupies one core, and immediately wakes up on interrupt. Real-Time Task is managed by scheduler, and is dispatched after scheduler process (from [7]).

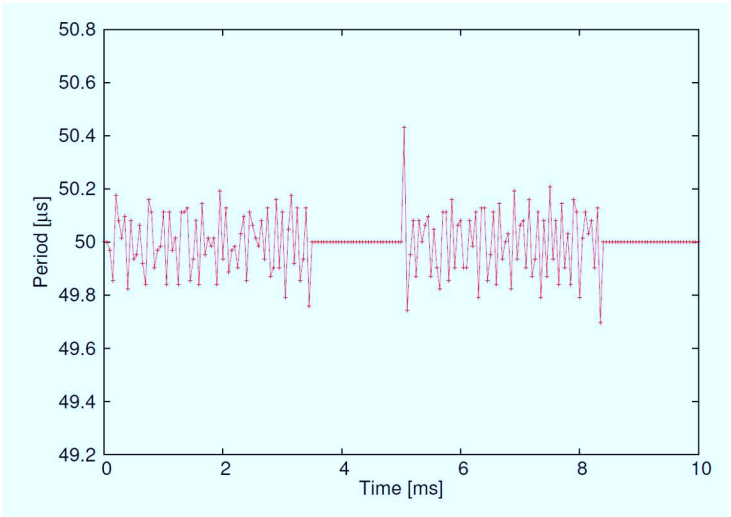


図 3.4: Jitter on execution time is lower than 1μsec(This figure is from [40]).

3.5)。本研究ではこの D-RMTP 基板とスタック接続可能なモータ制御基板を開発し、D-RMTP の実時間プロセッシング機能と組み合わせることで、従来にない応答性の高いモータ制御モジュールを目指していく。表 3.3 にこの基板の主要スペックを記載する。

表 3.3: Specifications of D-RMTP 20mm SiP Board

Input Clock	50MHz
CPU Clock	45MHz (Configurable by software)
SRAM	64kB
SDRAM	64MB
Size	39mmx26mm

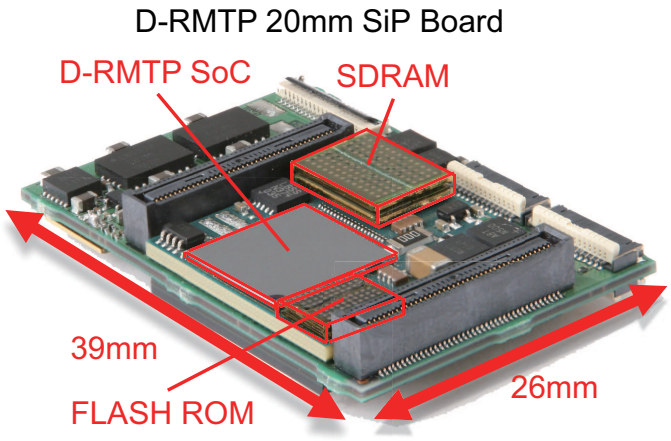


図 3.5: Dependable Responsive Multi-threaded Processor(D-RMTP) (photograph from [21])

3.3 RMTP-02D 基板の開発

本基板は浦田らによるヒューマノイドロボットのための大出力モータドライバ [45, 7, 46, 47] をベースに、実時間通信のための光通信メディア及び D-RMTP SiP 基板を搭載できるようリデザインしたものである。図 3.6 に RMTP-02D 基板の外観を示す。図 3.6 中に示したインターフェースは、表 3.4 に対応しており、従来の基板と互換性を保つためのインターフェースと新機軸のデバイスを接続するためのインターフェースを全て搭載するために、従来の 1.5mm ピッチ系のコネクタから、1.25mm 径ピッチのコネクタに変更している。

表 3.4: RMTP-02D 基板搭載インターフェース

Number	interface	application
1	Differential pulse input 3CH	Encoder
2	Differential pulse input 3CH	Hall Sensor
3	RS422 full duplex port 2CH	Communication bus Sensor interface
4	GPIO 1CH	Global reset switch, etc
5	Auxury I/O 3CH	I^2C,SPI
6	RS485 full duplex port 1CH	Absolute encoder, etc
7	BtoB Connector	Connect to FET board
8	Active optical port	High speed communication

3.3.1 ハードウェアロジックによる主要機能の実装

RMTP-02D 基板には FPGA をモータドライバのメインの処理デバイスとして搭載している。主に

- モータのベクトル制御回路
- 通信系のデータリンク・ネットワーク回路
- センサデバイス制御用回路
- トルク推定器 (C.3 節)

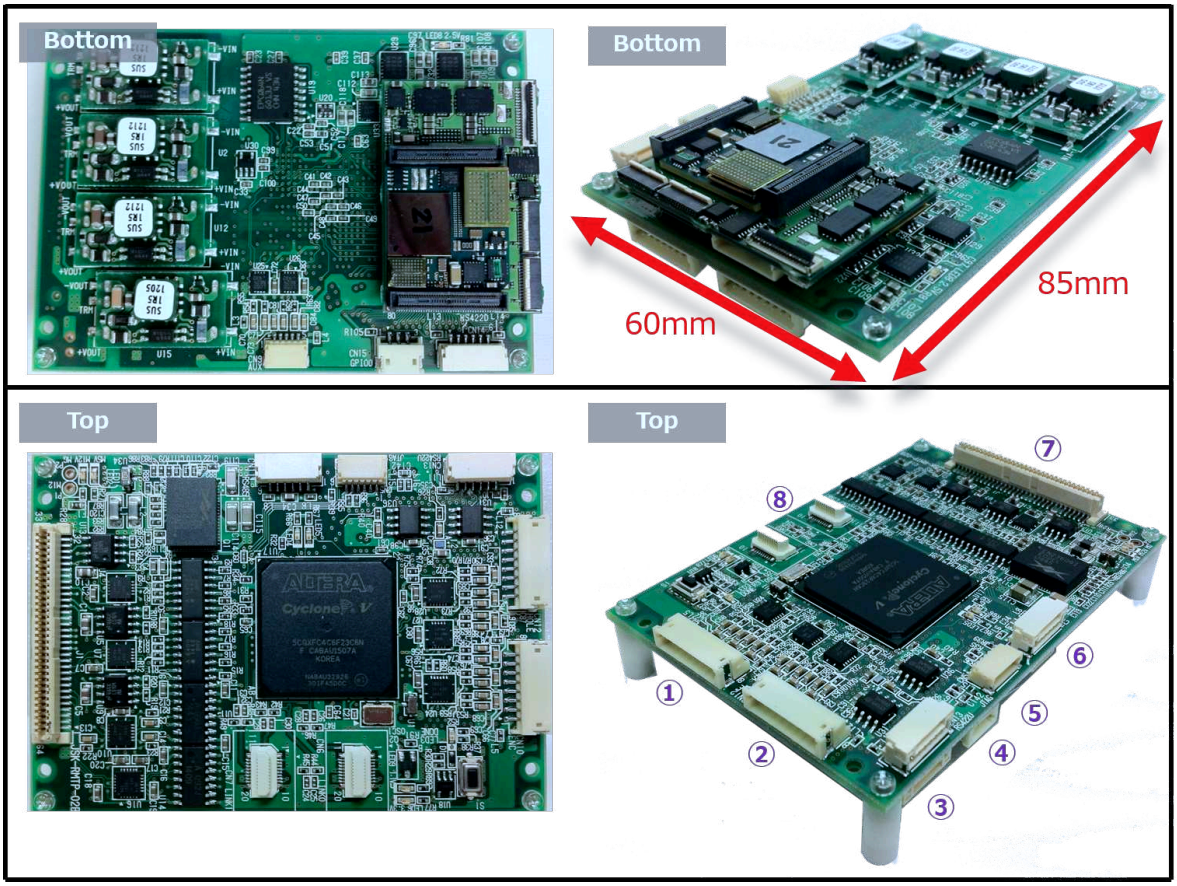


図 3.6: RMTP-02D Motor Driver Board

- マイコン用バスインターフェース回路

をハードウェアロジックとして実装している。FPGA 上のハードウェアロジックで実装することにより、高い実時間性と並列実行能力を付加することが可能となる。近年 FPGA は高速トランシーバ回路をハードマクロとして内蔵している製品が出てきており、RMTP-02D 基板でも通信系の物理層をハードマクロによって一部実装を行っている。

3.3.2 電流制御器とベクトル制御器

RMTP-02D 基板には浦田ら [48] によって開発された FPGA 論理回路実装の PI 電流制御器、ベクトル制御器と FET 基板を搭載している。ブラシレスモータ駆動用の三相 PWM 生成器と接続されており、FET 基板に実装されている各相の電流計測用センサからのフィードバックを電流制御器、ベクトル制御器に入力している。ベクトル制御演算は非常に速い制御周期を要求されつつも、座標変換操作といった演算が必要となるため、組込み用CPUでは比較的負荷の高い演算である。この実装では電流制御器とベクトル制御器は 38.4kHz で周期実行され、モータ制御の一連のフィードバックループの中で最も速い制御ループとなっている。FET 基板はモータで人と同様のトルク特性と関節速度特性を両立させるために、80 V 電圧駆動で最大 200 A のドライブ能力を持った設計となっている。図 3.7 に浦田らの FET 基板と組み合わせた RMTP-02D 基板を示す。

3.3.3 光伝送モジュール

高速な通信速度やノイズへの耐性、ノード間の絶縁性、長距離伝送が要求される通信では、光ファイバーケーブルによる光伝送が採用されるのが一般的である。ロボットでの使用例としては、HRP3L-JSK [46] で

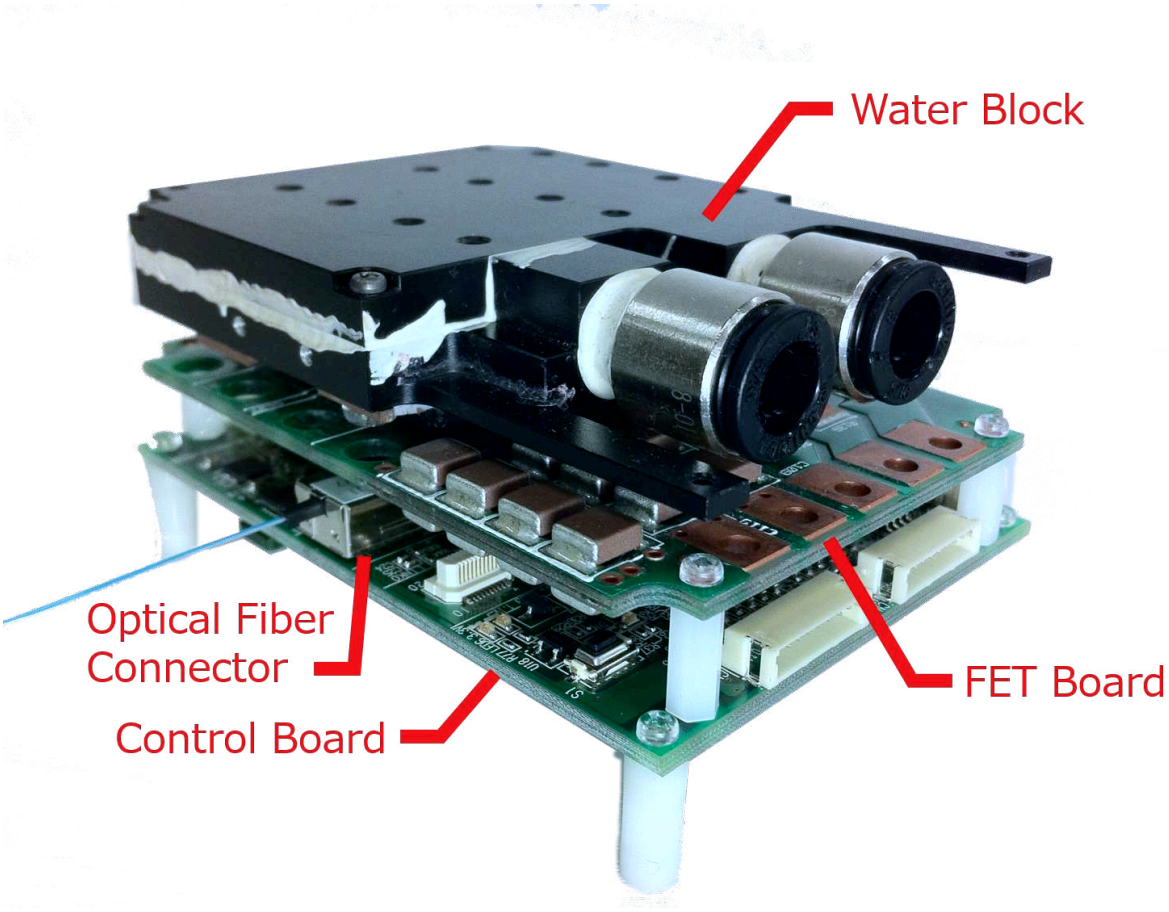


図 3.7: RMTP02D Motor Driver Board assembled with FET board and water block module.

は光ファイバー接続の PCI バスエクステンダによって、ロボット体外に置かれた主制御計算機と体内に搭載された分散型制御基板用インターフェースを接続している。PCI バスの要求する伝送帯域を持ちつつ、計算機とロボット間を絶縁しながらロボットが歩き回れるだけのケーブル長を実現することができている。

しかし、メタル電線による接続と比較して、

- 光電変換モジュールを基板側に搭載しなければならないことによるモジュールサイズの誇大化。
- 光ファイバーケーブルが曲げに弱く、狭い空間でのケーブルの取り回しが困難。
- コストが高い。
- ケーブルの延長・短縮加工が困難なため、機器内の配線長を最適化しにくい。

といった課題もあり、特にロボット体内で使用する通信システムに採用されるまでには至っていなかった。

近年、光アクティブコネクタが組込み用途向けに開発され、製品として登場し始めた。光アクティブコネクタは従来基板側に搭載しなければならなかった光電変換モジュールがコネクタとして一体化され、光ファイバケーブルと合わせて一つのモジュールとして構成された通信用ケーブルである。従来品と比較して、

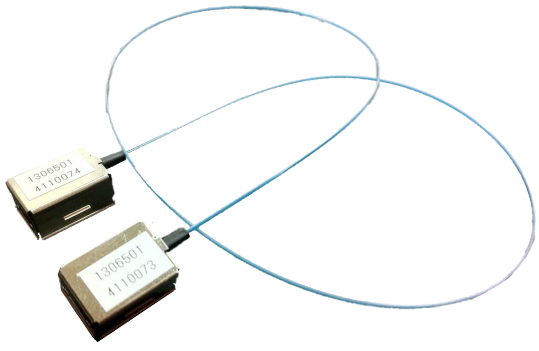
- モジュールが小型化され、サイズ要求の厳しい組込み用途でも使用可能。
- 光電変換素子はケーブル側についているため、基板設計時には従来の高速差動信号インターフェースと同様に扱うことが可能。
- 従来の光ファイバケーブルよりも細径化されており、最小曲げ半径も大幅に向上している。

といった利点が得られるようになっている。

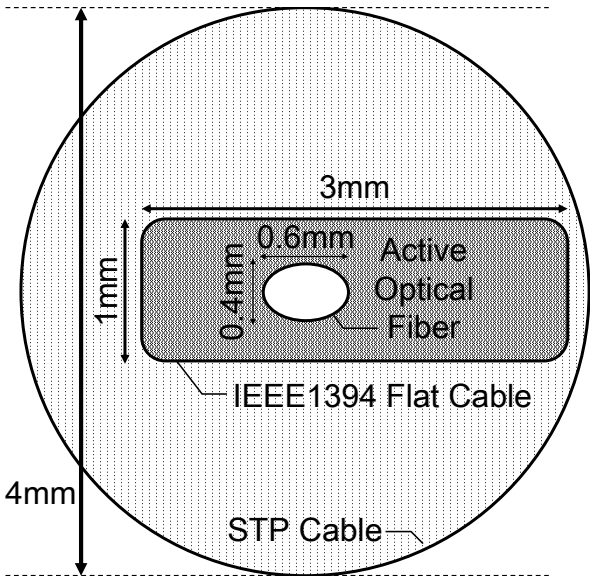
本研究では、光アクティブコネクタとして表 3.5 に示す Panasonic のモジュール (図 3.8a) を使用して、体内分散制御システムのための高速通信機構を実現する。本モジュールは全二重で最大 6Gbps という通信性能を有するだけでなく、約 0.5mm 角というファイバ径も特筆すべき利点となっている。図 3.8b に既存ロボットの体内通信系で主に使用してきたケーブル径を示すが、市販 STP ケーブルで最小径クラスの ϕ4[mm] のケーブル及び IEEE1394 フラットケーブルの 1[mm]×3[mm] サイズと比較して、圧倒的に細いファイバ径であることが分かる。また、細線同軸ケーブルも同程度のケーブル径ではあるが、双方向高速通信を実現するためには複数線を束ねる必要が生じるため、光アクティブコネクタのファイバ径には及ばない。このファイバ径では、等身大サイズのヒューマノイドロボットでは体内の配線において、空間制約を考慮する必要が事実上無くなり、自由に配線を行うことが可能となる。ファイバ最小曲げ半径 15mm という特徴も利用して、例えば関節を跨ぐ部位の配線において、多重にファイバを巻くことで、関節駆動にともなう繰り返し曲げによるファイバにかかる応力を低減することが可能となり、ファイバの断線等に伴う障害のリスクを低減する構成が実現される。図 3.9 に RMTP-02D 基板同士を光アクティブコネクタで接続した様子を示す。

表 3.5: 光アクティブコネクタの仕様

製造	Panasonic
型番	AYG4V13065
伝送速度	20Mbps〜6Gbps
伝送ポート	Tx,Rx 全二重 1Ch
ファイバ長	50mm,300mm,1000mm
ファイバ径	0.4mm × 0.6mm (2 心)
ファイバ最小曲げ半径	15mm
コネクタサイズ	12.7mm × 9.1mm × 5.9mm
光変換モジュール駆動用電圧	3.3V



(a) Photograph of a module.



(b) Comparison of active optical fiber and traditional thin communication cable.

図 3.8: Active optical connector

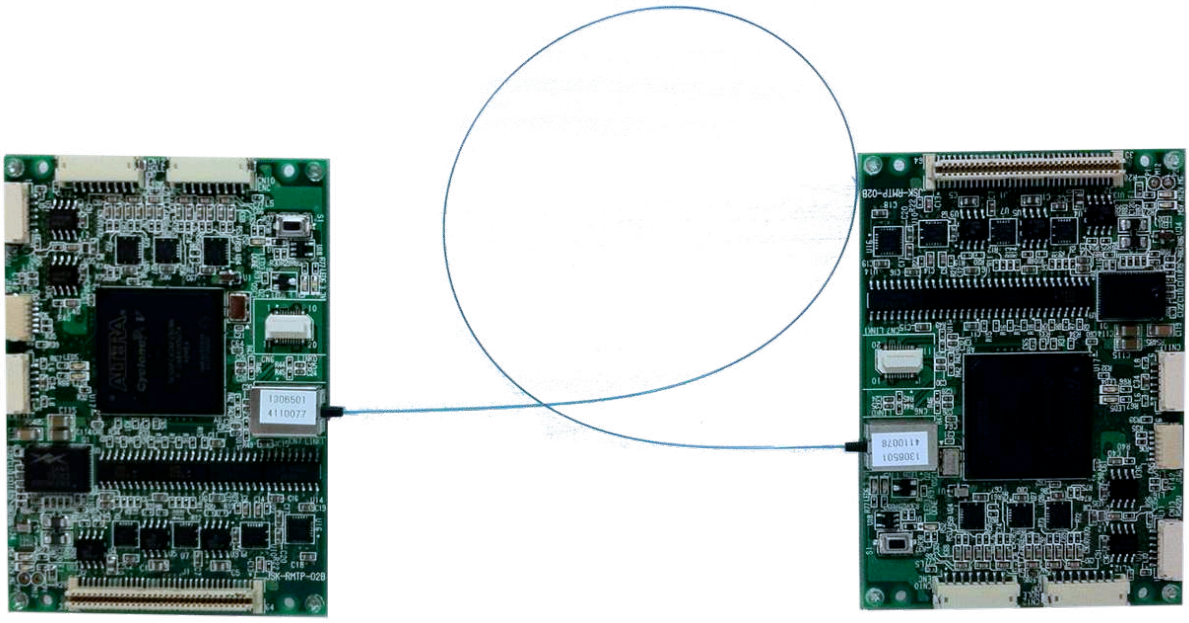


図 3.9: RMTP02D Motor Driver Boards connected by active optical cable.

3.3.4 マイコン用バスインターフェース

組み込みＣＰＵとＦＰＧＡや周辺デバイスとの間のデータ通信ではパラレルバス形式でのデータ転送が高速で一般的である。しかし、モータ制御周期を短縮していくと、このデータバス上の転送でもオーバーヘッドの問題が生じるようになり、実時間処理に影響を及ぼすようになる。データバスはデバイスやスレッド間で競合が起こり得るため、競合による待ち時間が存在することや、バス権獲得に処理時間が必要となることによるオーバーヘッドが重畳されるため多データを転送する場合は、単なるメモリアクセスと比較して遅い処理となる。しかし、データバスで接続された先にＡＤＣの変換データやモータドライバ・通信機構等のデバイスの制御レジスタが存在するため、高頻度でのアクセスが必要であり、データバス転送を高速化することが求められる。データバス転送の高速化の手法として、ＤＭＡによるメモリ間転送、バースト転送による連続データ転送等の技術が組み込みシステムでは使用される。これらの手法はデータバス転送の無駄時間を削減し、高速転送を可能とするが、データや制御レジスタへのランダムアクセス性を低下させる。そのため、ＦＰＧＡ上のデータアドレスマップを高速転送を想定して最適化した状態で配置する必要がある。

通常のバスアクセスでは、1 データの転送に 4 サイクル+ バス調停のためのオーバヘッドが生じ、スレッドはこの間応答待ちとなる。一方、DMA による転送では N データの転送に 4 サイクル+ N サイクル+ バス調停のためのオーバヘッドが生じる。また、スレッド自体はバスの応答を待たずに先の処理に進むことができるため、スレッドの実行時間を大幅に改善することが可能となる。

3.4 分散型センサデバイスインターフェース

RMTP-02D 基板では ADC デバイスを搭載していないため、直接アナログ入力によるセンシングは行わない。本基板ではロボット制御に必要なセンサデバイスは全てシリアルインターフェースを介して、外付けの拡張基板・デバイスを利用する形式としている。RMTP-02D 基板上には汎用の全二重 RS422 インターフェース及び 3 線 GPIO 入出力を利用した I^2C,SPI インターフェースを搭載しており、センサデバイス側のインターフェース, 要求データレートに応じて使い分けることが可能となっている。

図 3.10: Shock Detection Module

3.4.1 衝撃センサ

衝撃センサは加速度センサの一種であるが、特にインパルス入力的な加速度に対しても十分に計測可能なのように広い加速度計測レンジと周波数応答性を備えたセンサである。

本研究では永松らによって開発された衝撃検出モジュールを使用する [8]。ANALOG DEVICES 社製の ADXL326 および ADXL377 を使用することで、±16[G] 及び ±200[G] の2 種類の計測レンジに対応してデータを取得することが可能となっている。計測軸は X,Y,Z の3 軸を全て計測しており、サンプリングレートは通常利用時に約 20k[SPS] 程度得ることができる。衝撃検出モジュールのインターフェースには絶縁型 I^2C が採用されており、電磁ノイズの影響による計測不良を防いでいる。また、衝撃検出モジュール化による副次的な効果として、ロボットの骨格への取り付けが強固になるよう、取り付け点を6 ヶ所備え、また取り付け面に面接触の状態で固定できるよう基板背面に部品実装は行っていない。このように、骨格とモジュール間の固定を剛に行うことによって骨格への衝撃入力に対してより高速に応答させることが可能となっている。

衝撃センサはロボット体内の数あるセンサの中でも、特に外部入力に対する感度と応答性が非常に優れているため、ロボットの高速応答動作の実現において重要なセンサとなる。衝撃検出の感度は、骨格を爪先でつつくような非常に小さい入力から、着地衝撃のような大きな入力まで高精度で検知することが可能である。

衝撃検出モジュールのサンプリングレートは、制御周期の 1k〜10k[Hz] と比較して十分に高速であるため, フィルタリングによってより計測品質を高める余地がある。

3.4.2 6 軸力センサ

6 軸力センサは、並進3 軸方向の力とモーメント 三軸の計6 軸の入力を計測するためのセンサである。ヒューマノイドロボットにおいては足先や手先に搭載し、姿勢制御や手先力制御における重要なフィードバックを与えるセンサである (図 3.11)。6 軸力センサは軸間のひずみ干渉を抑えながら、高い定格入力を実現するために機械的な構造が非常に複雑になっている。そのため外形サイズは最大定格によってほぼ決定してしまい、ロボットの構造設計においてレイアウト制約の一つになる。センサインターフェースには、アナログ信号出力の製品とデジタル信号出力の製品がそれぞれ選択できる場合がほとんどである。PC との接続用に専用のインターフェースカードが合わせて販売されているが、本研究では体内システムに直接計測値を取り込むために、デジタル信号出力を RMTP-02D 基板に入力し、専用の6 軸力センサ制御ロジックを FPGA 内に実装した図 3.12 。この制御ロジックは CPU バスを通じて組込みシステムや通信システムから直接アクセスすることが可能となっている。

ロボット体内ではモータサーボノイズの影響が無視できないため、アナログ信号出力は計測精度の点で不利であると考えられる。ただし、デジタル信号出力製品では計測回路を内蔵するため外形サイズがアナログ信号出力製品と比較して大きい傾向がある。計測回路が外付けの場合もあるが、アナログ信号線がセンサ筐体外に出てきてしまう点に留意しなければならない。また、6 軸力センサのサンプリングレートは多くの製品でおよそ 1kHz から 2kHz が一般的である。これは現在のロボットの 1kHz 制御システムにおいては十分なレートであるが、10kHz 制御システムを実現していく場合にはスペックとして不十分になって

くる。高速周期でのサンプリングが可能な6 軸力センサはまだ製品として広まっていないため、実現するためにはアナログ出力製品について独自に ADC モジュールを開発する必要があると考えられる。

図 3.11: Six axis force sensor(表 3.6).

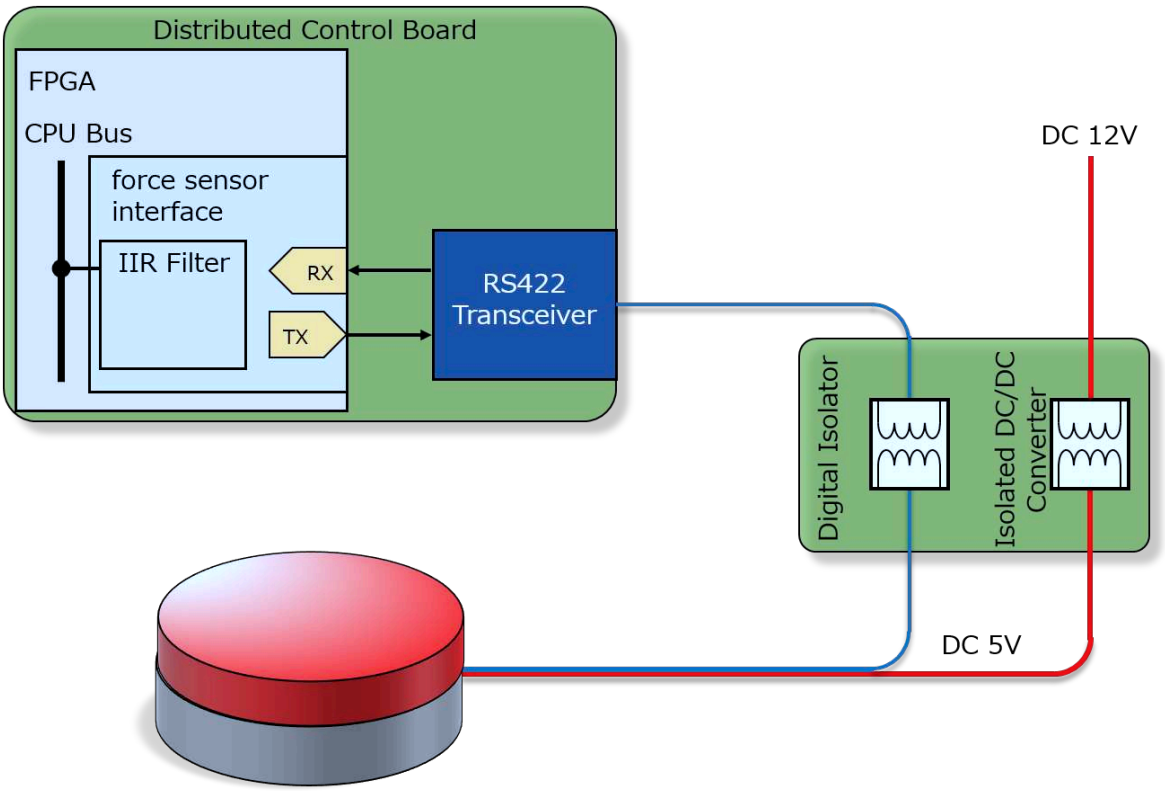


図 3.12: 6axis force sensor interface with isolator board.

また、ロボットの骨格とボルトによって剛結されるため、金属製のセンサ筐体は骨格と同電位に落ちる。そのため、ロボット骨格を経由してモータや制御基板からのノイズがセンサ筐体に侵入してくるので、センサ内の基板の筐体に対する電気的なシールドの構造が6 軸力センサの測定値に大きな影響を及ぼす。実際

に、図 3.13 にモータのサーボノイズによってロボット骨格に生じる電位の振れと、対策を施さない場合に 6 軸力センサがノイズから受ける影響を示す。

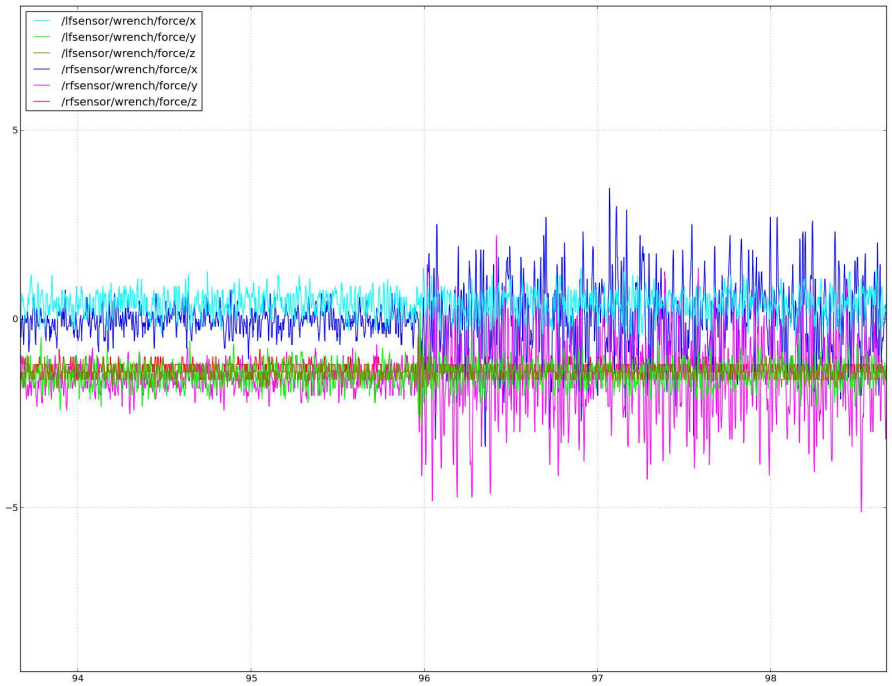


図 3.13: Influence on sensor measurements by motor servo noise.

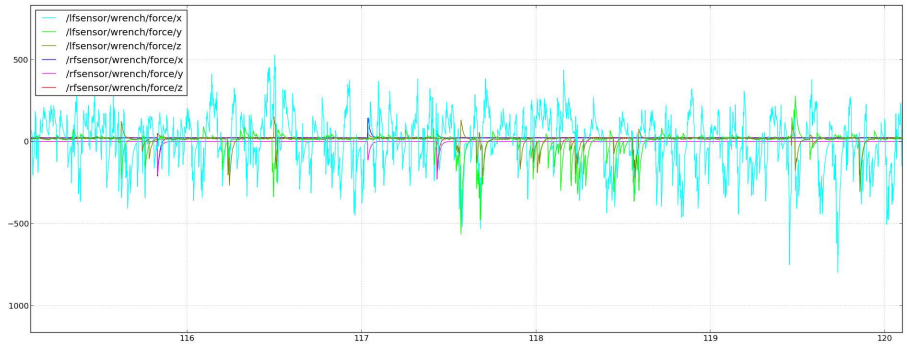


図 3.14: Comparing the effect of isolator.

本来、ロボットの骨格自体を電気的に接地させることでこのような筐体の電位の揺れを抑えるのが一般的である。しかし、自立型のヒューマノイドロボットでは接地のためのケーブルを接続することが不可能であり、骨格は電気的に浮いた状態であったり人と同様に骨格は帯電していたりすることが考えられる。体内電源の-極側に骨格の電位を落とすことも考えられるが、モータサーボ電源系は非常にノイズレベルの高い系統となっており、やはり骨格を電気的に安定に保つには困難である。このような点から、筐体が骨格と剛結される6 軸力センサでは、内部の測定回路の筐体に対する絶縁性がヒューマノイドロボットにおいては重要な要素となる。あるいは、フィルタリングによるノイズの除去も考えられ、実際に6 軸力センサ製品には内蔵の回路においてあらかじめフィルタ処理を設定可能なものも多く存在する。ただし、過度なフィルタリングは計測値に位相遅れを生じさせるため、姿勢制御のようなセンサの応答性が制御性能に直結する用途では、カットオフ周波数が 500Hz 程度以上で実際には運用される。

本研究では、こうしたヒューマノイドロボットに6 軸力センサを搭載するにあたり、センサインターフェースを絶縁化して対処を行った (図 3.15)。6 軸力センサに供給する電源及び通信信号を絶縁化することで、

ロボット体内の電源系統に印加されているノイズが直接センサ内に侵入するのを抑制している。また、使用した6 軸力センサのスペックを表 3.6 に示す。

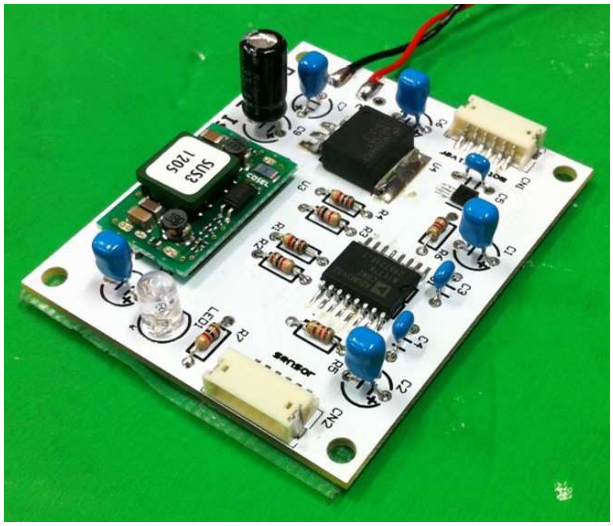


図 3.15: RS422 isolator board.

表 3.6: 6 軸力センサ

製造元	Wacoh Tech[1]
型式	WEF-6A1000-80-40-RCXTi2
インターフェース	全二重 RS422
定格荷重 (Fx,Fy,Fz)	2000[N]
定格荷重 (Mx,My)	80[Nm]
定格荷重 (Mz)	40[Nm]
検出感度 (Fx,Fy,Fz)	3.275[LSB/N]
検出感度 (Mx,My)	81.88[LSB/Nm]
検出感度 (Mz)	163.75[LSB/Nm]
最大サンプリング周波数	2k[SPS]
応答周波数	1kHz

[1]http://www.wacoh-tech.com/

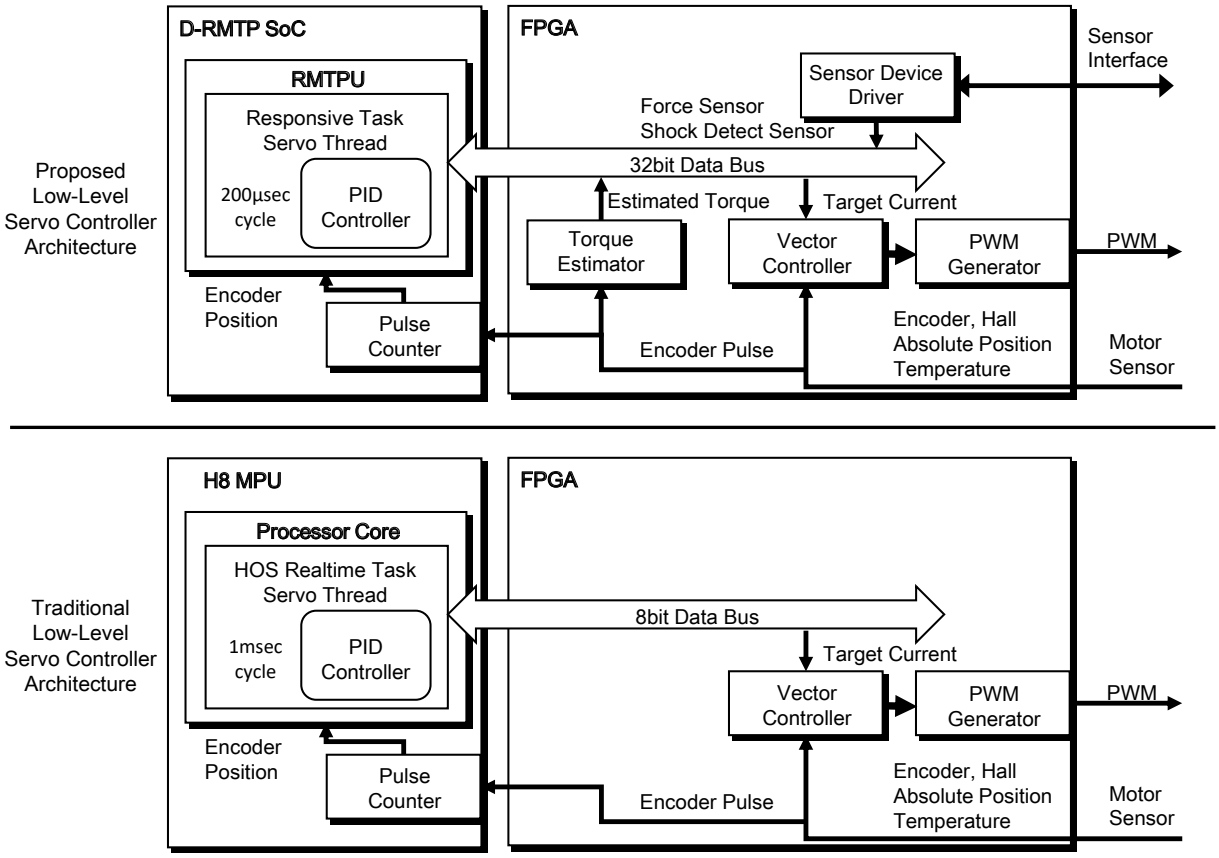


図 3.16: Sensor data and Control data flow in RMTP-02D board and a traditional H8 based board.

3.5 D-RMTP 及び RMTP-02D 基板による実時間モータ制御の実験

RMTP-02D 基板に搭載された D-RMTP 基板にてモータの制御タスクを実時間プロセスとして実装する。まずセンサデータと制御コマンドの基板上での流れは次の通りになる (図 3.16)。D-RMTP 上のサーボ制御タスクは、

1. エンコーダパルスカウンタからのモータポジション情報の更新。
2. モータポジションの変分からモータ回転速度の計算。
3. データバス経由で FPGA 上のセンサデータを取得。
4. サーボ計算の実行。
5. 演算結果であるベクトル制御器用目標電流値をデータバス経由で FPGA に書き込み。
6. 上位システムとの通信用データをデータバス経由で FPGA 上レジスタ・メモリに書き込み。
7. スレッドをスリープさせ、次周期を待つ。

という手順で処理が行われる。(2) 及び (5) のデータバスリード・ライトはデータ数が多いと、CPU からは重い処理となり、短周期での実時間実行に影響を与える。そのため、DMAC による高速なデータ転送処理を行うよう FPGA 上のレジスタ・メモリのアドレス配置を最適化している。

以上の処理内容で、モータ制御タスクは 200μsec=5kHz の制御周期を実現している。従来の 1msec 周期の制御周期よりも 5 倍の高速化が達成できており、力制御の応答性の向上が期待される。

次に実時間サーボ制御タスクの実装について述べる。図 3.17 に示すように、D-RMTP 上での実時間タスクは RTServoTask クラスによって管理を行っており、タスクの実行周期や実行時間の計測処理、実際にサーボ演算を行う Local Motor Controller クラスの登録、初期化を行っている。Local Motor Controller に

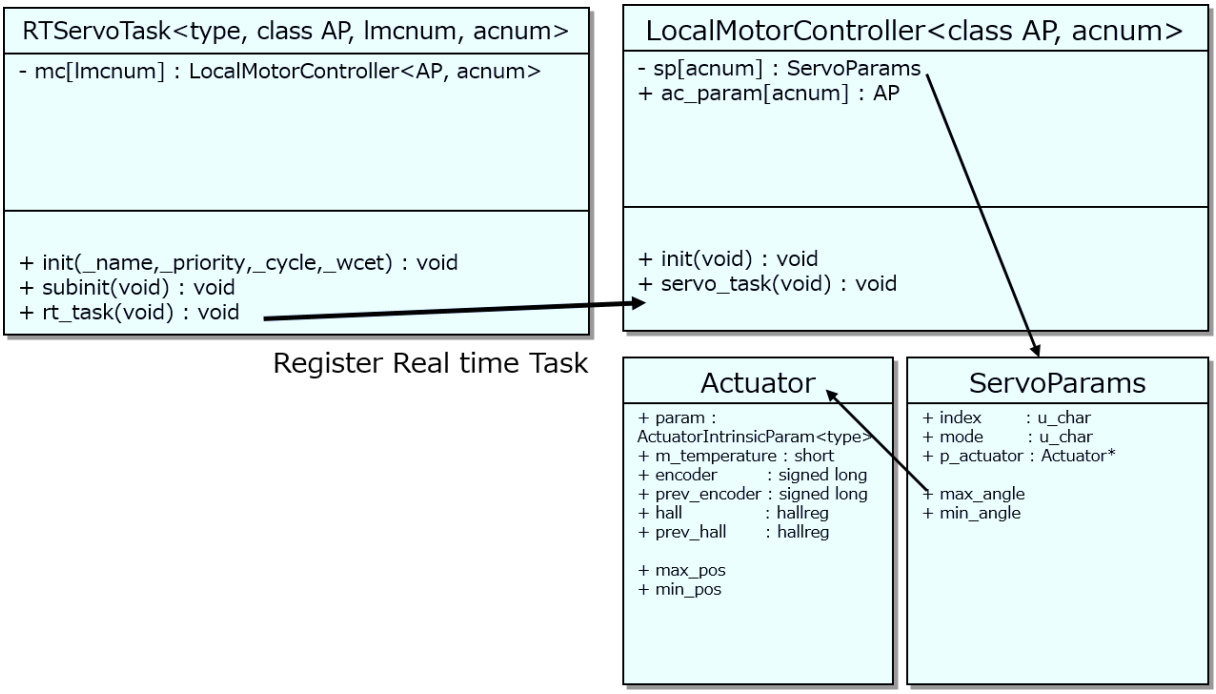


図 3.17: Real-time servo control task on D-RMTP.

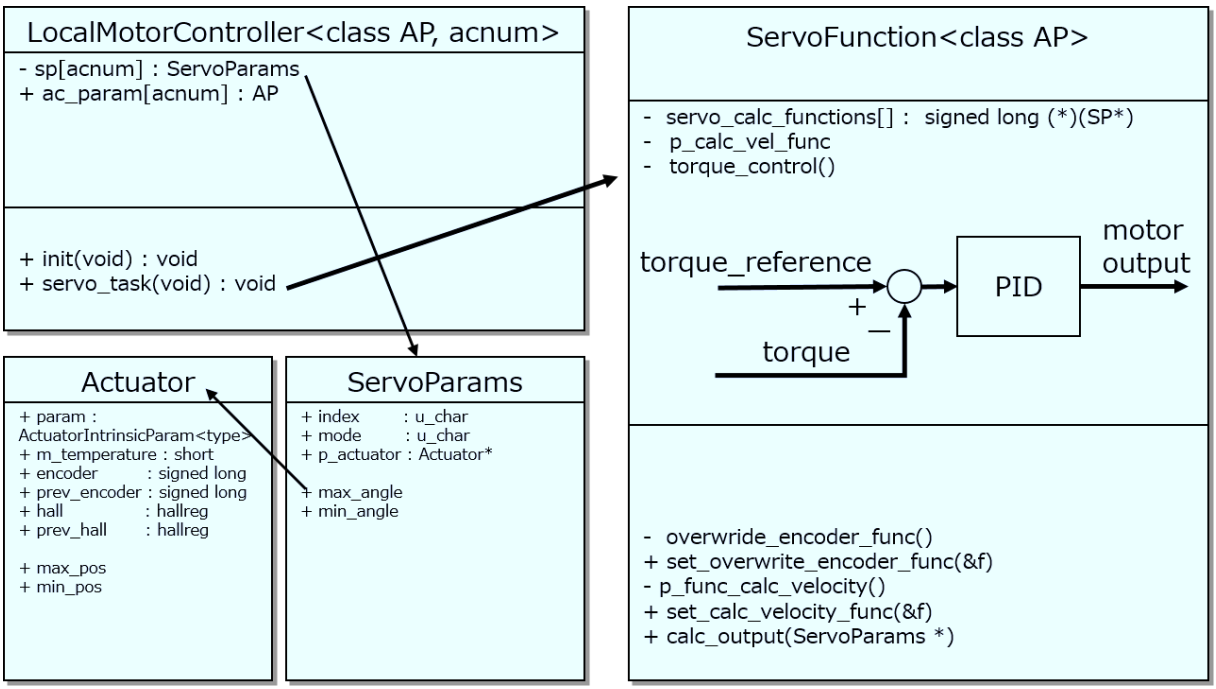


図 3.18: PID torque controller called from local servo controller.

はサーボ制御で必要となる基本パラメータ、アクチュエータの物理的スペック情報を保持しており、初期化時にデフォルト値または上位層から送信されてきたパラメータを適用する。

Local Motor Controller は周期実行時にサーボ演算処理を現在設定されている制御モードに応じて、ServoFunction クラスから選択して実行を行う (図 3.18)。ServoFunction クラスには、位置制御、力制御、剛性制御等の基本モータ制御関数を登録している。

3.5.1 単軸トルクフィードバック制御による実時間モータ制御実験

RMTP-02D 基板に実装したモータ制御システムについて、トルクフィードバックに基づく単軸でのトルク制御を実行することで、実時間プロセッサの動作検証を行う。

実機には A0-B spec のエンドエフェクタ軸を用いて、手先に搭載している 6 軸力センサの Z 軸モーメントの計測結果を関節トルクとして RMTP-02D 基板で直接取得することで、トルク制御系を構成する (図 3.19)。トルク制御として次の PID 制御器を含むサーボ制御タスクを実時間プロセスとして、D-RMTP 上で起動している。また、計測データを HRP3L-JSK で使用している従来通信リンクによって主制御計算機に送り、1msec 周期でのログを記録した。トルク目標値は0 Nm を設定し、外部トルク入力に対してなじむ状態とした上で、外乱を手で加えた。

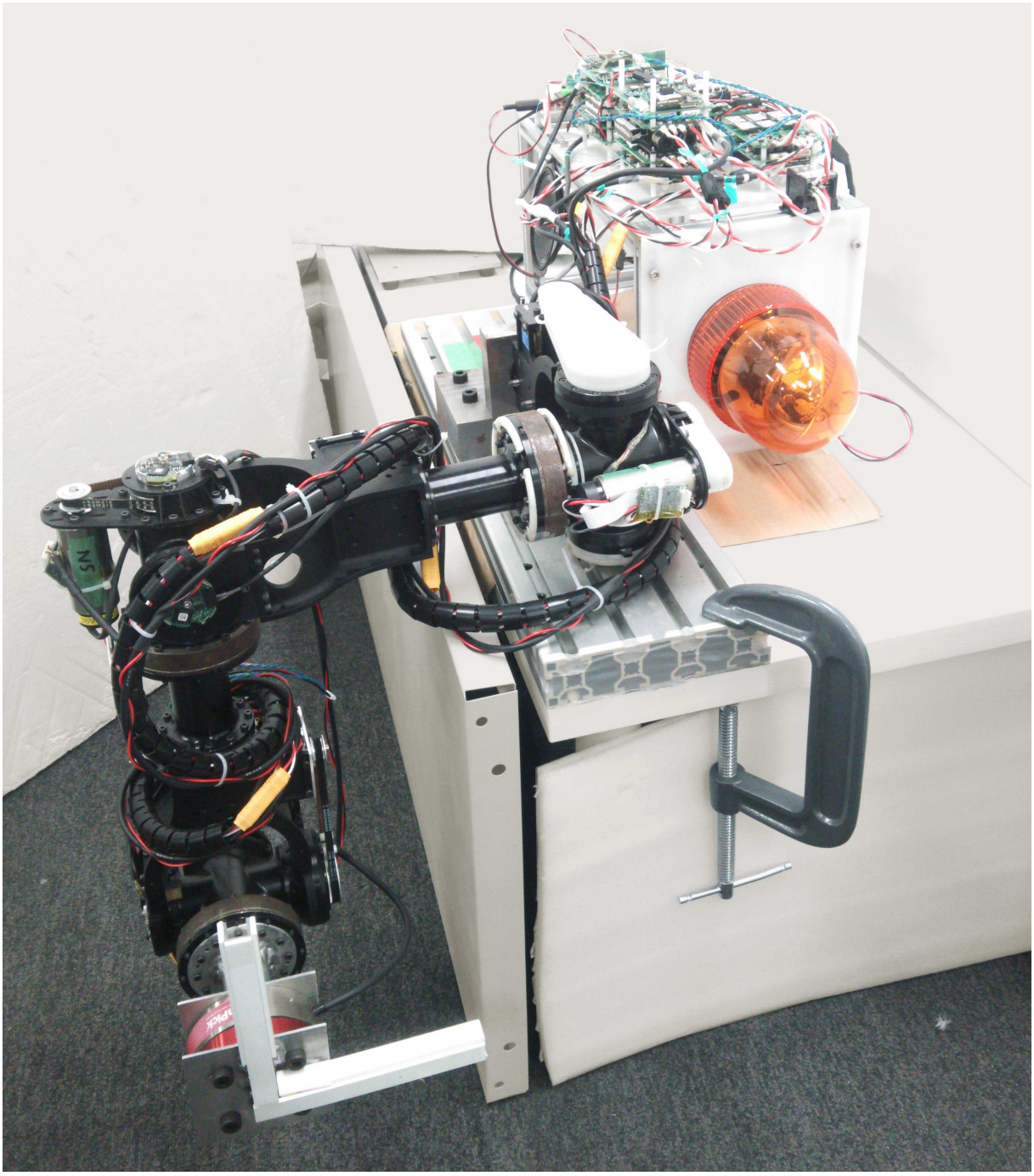


図 3.19: Experiment of single motor control unit on A0-B spec.

図 3.20 から図 3.22 にトルク制御時のエンコーダ、トルクセンサ値、制御器出力のログデータを示す。トルクセンサ値はオフセット値が 0.2Nm ほどのっているが、外部からの入力に対してなじむことでほぼ一定値を保てている。グラフの 3 秒から 8 秒の間は、振幅は関節角 10 度、周期5 Hz ほどの回転運動を手で関節軸に加えている期間である。この期間トルクセンサ値は ±0.1Nm ほどのトルク目標値との誤差が出てしまっている。モータやリンク慣性等を考慮していない制御系であり、周波数応答性については課題が残る。0.1Nm の誤差は手で入力している感触としても、抵抗を少し感じる大きさである。

また、グラフ後半は 30 度ほど連続して動かしている期間であるが、こちらについてはトルク追従性能を高く発揮している。

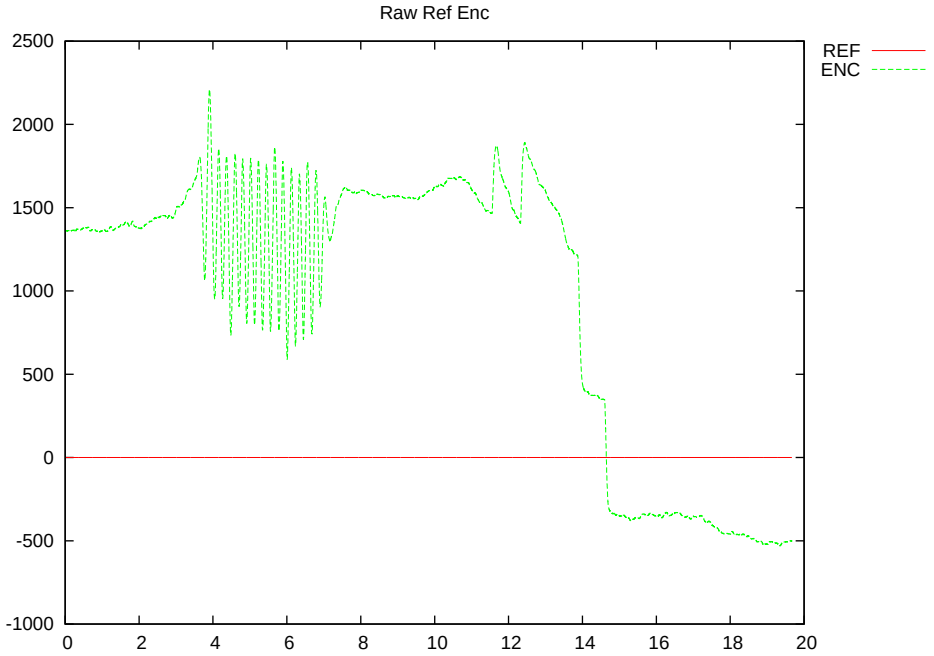


図 3.20: Encoder data.

従来構成のモータ制御基板との比較

本実験でのトルク制御の性能の比較対象として、H8 マイコンをコントローラに用いた従来型基板にて同様の構成によって関節トルク制御を行い、トルク追従性の比較を試みた。従来使用してきた H8 マイコンではモータサーボ制御周期は 1msec であり、制御モードに RMTP-02D 基板で実験したタスクと同様の PID ベースのトルクフィードバック制御を実装し、比較を行った。なお、単に制御周期が異なるだけでなく、プロセッサのアーキテクチャが根本的に異なることによるプログラムの厳密な処理ルーチンの差異、従来型の基板を使用するため FPGA 側回路のロジックのバージョン違いや電源系統の差異等があるため、制御周期に違いを示すための比較とすることはできないが、システム全体としてどのように改善されたかを示す指標として考えることはできると思われる。

図 3.23 に、従来基板システムでのトルクフィードバック制御時のログデータを示す。電源システムの差異によるトルク信号へ重畳するノイズの差や制御周期の違いにより、外部入力に対して十分になじむまで制御ゲインを挙げることが難しく、グラフのように非常に大きいトルク誤差が生じている。

3.5.2 Responsive Task による実時間モータ制御の性能検証

3.2.2 節にて述べた低ジッタ性能を持つ Responsive Task による実時間モータ制御タスクによる性能を検証する。ここでは単軸試験機によるステップ応答軌道を取得した。図 3.24 に実験に使用した単軸試験機を示す。モータドライバには RMTP-02D 基板と D-RMTP が使用されており、モータは HRP3L-JSK で使用

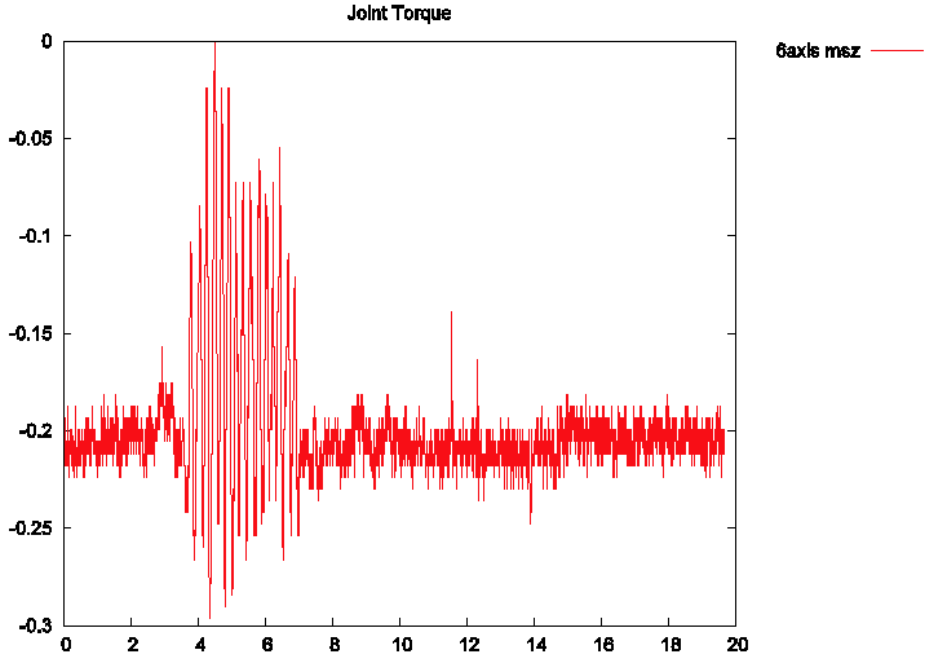


図 3.21: Joint Torque data(Nm).

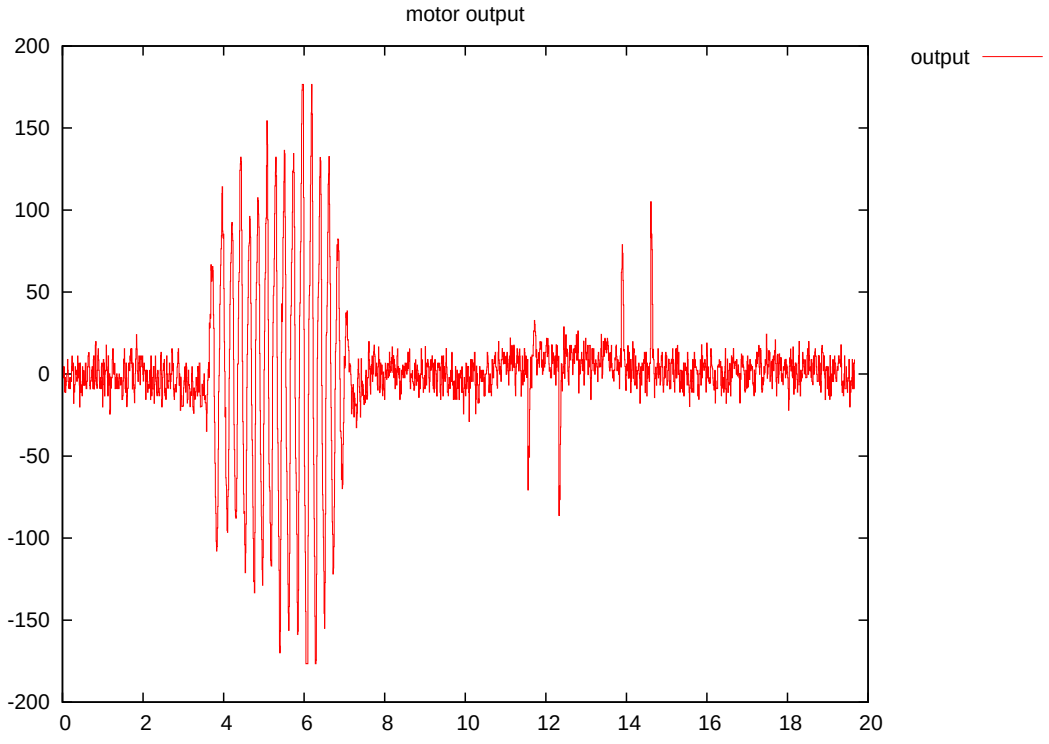


図 3.22: Controller output data(Low data).

されているものと同一である。3.75:1 の低減速比なプーリ減速機を介して棒リンクを回転するシンプルな構造となっており、減速機での損失やリンク剛性に起因する振動等の影響を無視することができる。目標関節角度は、0.100[second] の間に 360[degree] 回転する軌道を与えた。この目標関節軌道はメイン制御 PC 内のロボット制御層関節角度軌道補間器にて躍度最小補間によって 500Hz で中間目標関節角度が生成され、実時間インターフェースを介してモータドライバへ毎周期送信される。なお、この回転角度は実際にロボット

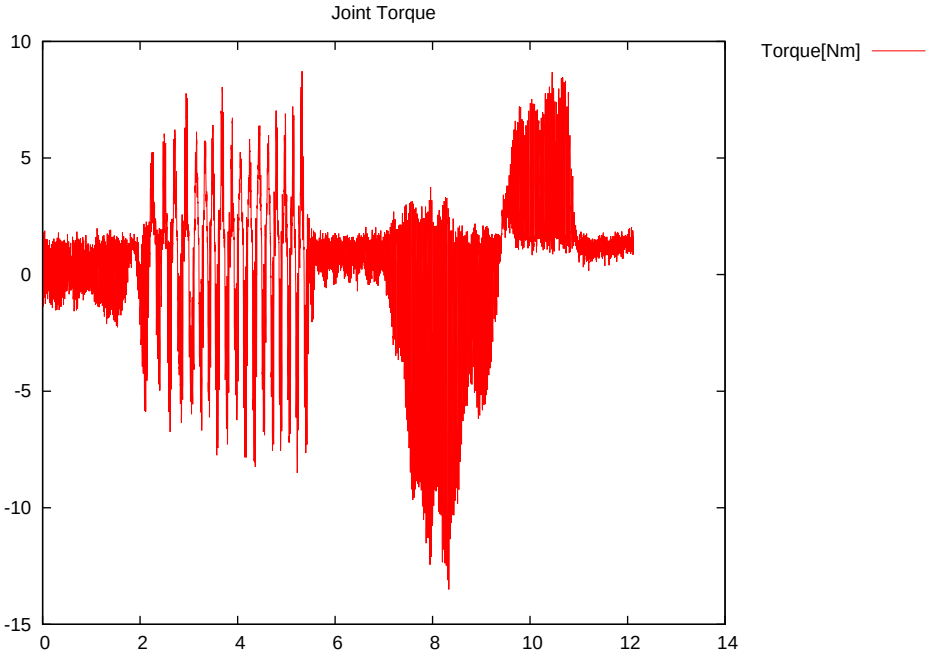


図 3.23: Joint Torque data on previous control board.(Nm)

で使用される減速比 240:1 換算では約 5.6° に相当し、最大関節角速度はモータドライバの通常モードでの最大速度の約 1/10 であるため、モータドライバのスペックから見て十分余裕のある軌道である。

RMTP-02D 基板上に接続された D-RMTP が目標関節角度への追従を行うサーボ制御演算を行い、モータドライバ内のベクトル制御器へ目標電流値を出力する。サーボ制御は関節角度エラー値を入力とし、目標電流値を出力とする PD 制御器で実装されている。サーボ制御タスクはクロック 45MHz の D-RMTP 上で実行時間が平均 31.2μsec 要する。

実験条件としてサーボ制御タスクの実行周期、並びにそのタスクの実時間実行機構 (Responsive Task or Real-Time Task) について以下の組み合わせで応答軌道を記録し、比較を行った。Real-Time Task では最大実行周期のタスク以上の周期でスケジューラが呼び出される必要があるため、実行タスクの周期を上げるとそれに比してスケジューラオーバヘッドが大きくなる。そのため、本実験環境での Real-Time Task によるサーボ制御タスクでは 2000Hz が実用的な最大周期となる。なお、実験条件によっては PD 制御が発振してしまうため、P ゲイン、D ゲインは各条件でチューニングを行った。ここではゲインは目標値への到達までのレイテンシを短縮しつつ、静的状態で発振しないようにチューニングを行っている。また、D 成分演算時に用いるローパスフィルタは時定数が $Ts = 0.001$[sec] となるよう各制御周期で係数を調整している。実際に実験で使用したゲイン値についても表 3.7 に示す。また、サーボ制御タスク以外にスレッド状態監視タスク、センサデータ処理タスクが並列に実行された。

表 3.7: Configurations of Experiment

Label	Execution Mechanism	Period	P Gain	D Gain
A	Responsive Task	1000Hz	1200	1000
B	Responsive Task	5000Hz	2000	1000
C	Real-Time Task	1000Hz	960	1000
D	Real-Time Task	2000Hz	880	1000

実行結果のロギングはモータドライバ基板とメイン制御 PC の間に位置するインターフェースブリッジ基板 (5.4 節) にて、D-RMTP 内サーボタスクと同期して実行されるモニタリング、データロギングアプリケーションを実装して行った。従来システムでは、通信帯域やメイン制御 PC で走る OS の制約等により下

図 3.24: Test instrument composed of a motor and 3.75:1 reduction pulley with single bar link.

位コントローラ内の制御量のログを取ることは困難であったため、中間層インターフェースブリッジ基板の役割は重要と言える。本実験では A,B,C,D の各条件について 10 回の試行を行った。

また、制御の正確性、精度を示す指標として、それぞれ目標角度に対する関節角度誤差、試行間のバラつきを計測結果から計算する。

図 3.26 に実験によって得られた、関節角度制御のステップ応答軌道を示す。その際に撮影された実験の様子を図 3.25 に示す。各軌道は目標関節角度軌道が時刻 0.100[sec] に立ち上がり始めるように時間軸のオフセットを合わせている。また、図 3.27a は図 3.26 の立ち上がり付近を、図 3.27b には整定領域付近の拡大図を示す。また、図 3.28a は目標関節軌道と実関節軌道の誤差角度、図 3.28b にはサーボ制御タスク内で関節角度の差分から計算された関節角速度軌道を示す。

表 3.8 に、各実験条件での 10 回試行時のオーバーシュート量の最悪値、及び立ち上がり時の最大誤差値を示す。実験条件 B による軌道ではオーバシュート量が 4.94°、立ち上がり時誤差値が-26.92°となっており、他の条件と比較して最もオーバシュート量、立ち上がり時誤差値を抑えることができている。

各実験条件について 10 回試行した関節角度軌道のバラつきについて評価を行う。軌道間のバラつきの指標としては様々なものがあるが、この実験では試行間でほぼ同一の軌道を描いていることから軌道間のマンハッタン距離をバラつきの指標とする。マンハッタン距離は軌道の形状違いだけでなく、位相ずれについても数値に反映されるためここでのバラつきの指標として十分と考えられる。表 3.9 に各実験条件の平均軌道を計算し、その平均軌道からの各軌道のマンハッタン距離を計算し、さらにその平均を計算したものを示す。A,B,C の各条件については各軌道間のバラつきは 1° 未満に収まっているが、D のみ大きくバラついていくことがわかる。同様にサーボ制御タスク内で計算された関節角速度軌道の平均マンハッタン距離を表 3.10 に示す。関節角度軌道の場合と同様、A,B,C の各条件についてはバラつきは約 20[deg/sec] に収まっているのに対して、D のみ 1042[deg/sec] と試行間で計測値が大きくばらついていることがわかる。

表 3.8: Worst case error of joint angle trajectory for each trials

	A	B	C	D
overshoot[deg]	9.69	4.94	6.57	13.05
max error[deg]	-42.52	-26.92	-53.32	-46.08

表 3.9: Similarity of joint angle trajectory for each trials

Similarity (avg. of manhattan distance)

A:1000Hz(resp)	B:5000Hz(resp)	C:1000Hz(RT)	D:2000Hz(RT)
0.2008	0.0639	0.0873	1.7911

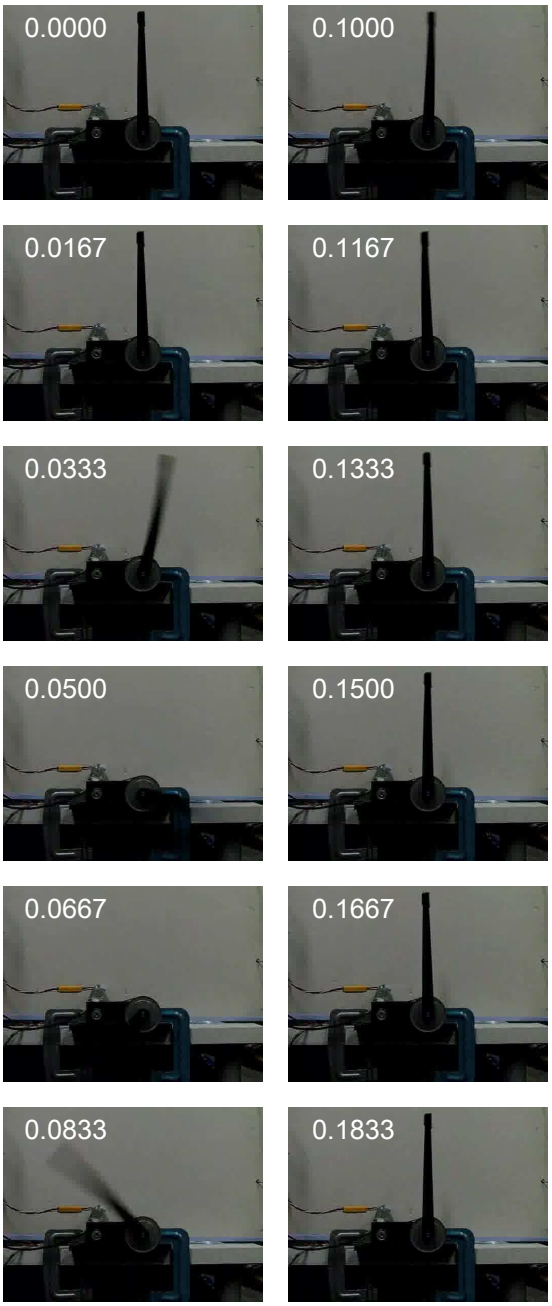


図 3.25: Photographs taken with a 300fps camera.

表 3.10: Similarity of joint angle velocity trajectory for each trials

Similarity (avg. of manhattan distance)			
A:1000Hz(resp)	B:5000Hz(resp)	C:1000Hz(RT)	D:2000Hz(RT)
26.1	23.5	21.5	1042.0

制御の正確性、試行間の繰り返し精度の高さ共に制御周期が最も早い実験条件 B が、他の実験条件と比較して明らかに有意な差で優れていることが確認された。立ち上がり時誤差値は関節角度誤差増加時に指令電流値を素早く増大させることが重要であるため、単純に P ゲインの値が支配的と考えられる。P ゲインを増大させても発振しないよう、制御周期の高速化が有効であることが確認できる。

また、実験条件 D では 10 回の試行間のバラつきが関節角度に換算して平均して約 1.8 ° 程度となってお

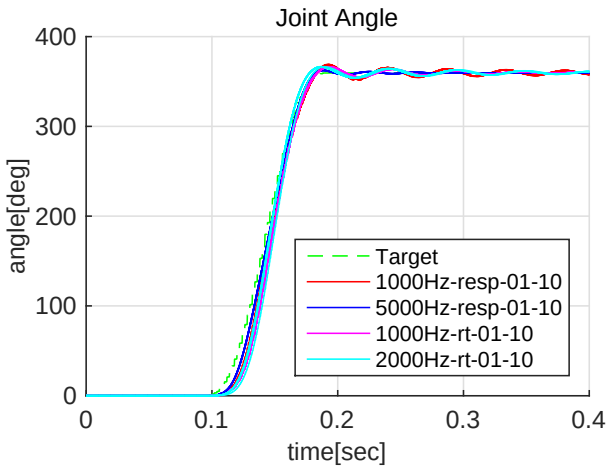
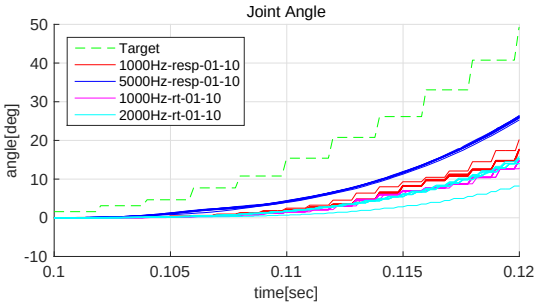
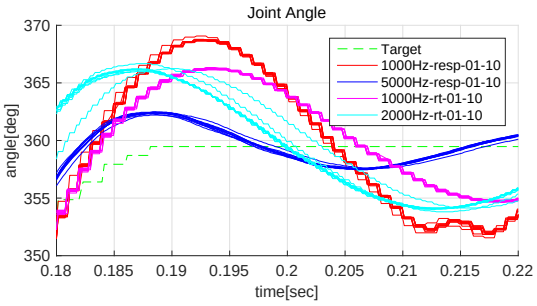


図 3.26: Trajectories of joint angle.

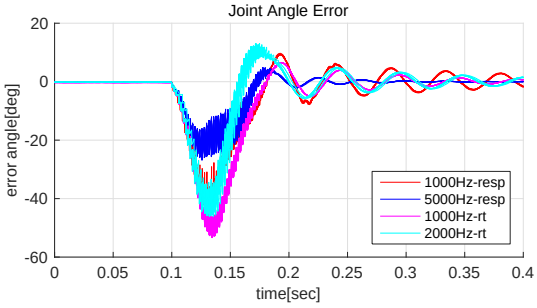


(a) leading edge area

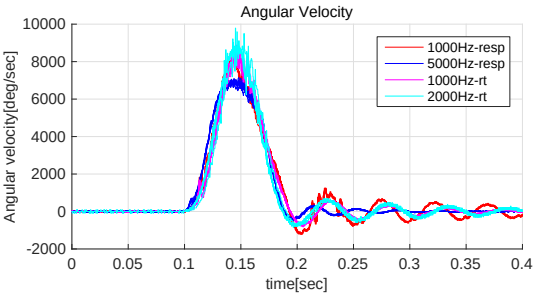


(b) Ssettling area

図 3.27: Enlarged trajectories of joint angle.



(a) Error



(b) Velocity

図 3.28: Tajectories of joint angle error and velocity.

り、他条件と比較して突出して大きくなっている。実験条件 D の Real-Time Task での 2000Hz ではスケジューラの呼び出しが頻繁に発生するため、スケジューラ自体のオーバヘッドの大きさによってサーボ制御タスクの実行タイミングジッタが相対的に大きくなっていると考えられる。従って、実験条件 C と比較して制御周期が高速化することで P ゲインを向上させることができているが、実時間タスクにも関わらずサーボ制御の繰り返し精度が悪化していると考えられる。

Responsive Task は 5000Hz での実行時でも高い繰り返し精度が保たれており、なおかつ制御周期の高速化によって制御の正確性も向上できることが実証された。

3.6 本章のまとめ

本章では、体内分散制御システムの末端ノードとなるモータドライバについて、本研究における設計要件を満足するよう既存の大出力モータドライバをベースに、山崎らによって開発された実時間プロセッサ D-RMTP をモータ制御用マイコンに用いることが可能な RMTP-02D 基板の設計・製作を行った。

この制御基板では、実時間プロセッサ D-RMTP の提供する低遅延周期実行スレッド機能を活用した Responsive Task により、従来の組込み環境構成では難しかった低ジッタ 200μsec 周期でのモータサーボ制御タスクを実現し、ステップ応答実験にて高い制御追従性能と繰り返し精度を有することを実証した。

また、この制御基板では FPGA ハードウェアロジックで実装された専用センサドライバを搭載することで、6 軸力センサや関節トルク推定等の各種ヒューマノイドロボット制御で必要となるデバイス・計測値を体内分散通信系に直接接続可能である点が特色として挙げられる。これによって、並列実行にて高精度・高周期でのセンサデータの計測及びそれらを体内ネットワークに直接送信することが可能となり、システム全体での取り扱いデータフローを増加させることが可能となった。また配線コストの低減によって身体ハードウェア構成の小型化・軽量化・レイアウト自由度の向上につながり、身体運動能力の向上を実現可能な手法となっている。

第4章

マルチプロトコル対応多ノード間低遅延体内分散通信系の開発

4.1 はじめに

本章ではヒューマノイドロボットの体内分散制御用ノードを相互接続する分散通信系の構成方法について述べていく。

最初に 4.2 節では筆者が携わってきた HRP3L-JSK[50, 46] と腱志郎 [21] の体内通信系の構成、及びその他一般に利用される体内バス用通信規格について解説を行い、2.3 節及び 2.8 節にて設定した通信系設計要件と比較してどのような点で課題を抱えているのか明らかにする。

次に 4.3 節にて、本研究で提案する分散通信系を構成する通信リンク部の設計について述べる。第 3 章にて述べた通り、RMTP-02D 基板にはノード間の高速シリアル通信を可能にするギガビット帯の性能を持つ光アクティブ型インターフェースを搭載している。耐環境性能の向上、ケーブル構成の最適化、データレート・実時間・低遅延性能の向上を実現する上で適したハードウェア構成となっている。本研究で使用する光アクティブコネクタの仕様については第 3 章にて掲載したが、4.3 節では実際にノード間通信を行うために実装したシステム側のハードウェアデザイン、パケットのデータフロー制御手法について述べる。

4.4 節では、2.8.2 節にて導入した RPC 型通信とデータストリーム型通信をハードウェアロジックレベルで低遅延に実現するためのプロトコル処理ブロックの実装方法とその利用方法について述べる。

その次に 4.5 節では、設計された通信プロトコルについて、実際にロボットで使用するアプリケーションデザインを示し、実験・シミュレーションにてその通信性能の検証を行う。

最後に 3.2 節にて導入した山崎らの D-RMTP の機能の一つである *Responsive Link*[30] の構成について述べ、その評価を行う。

4.2 従来ロボットにおける体内通信系の構成

4.2.1 HRP3L-JSK における体内通信系の構成

HRP3L-JSK では、主制御計算機において歩行用関節角軌道が生成され、制御周期 1msec の実時間ロボットインターフェースプログラムを通して、体内に搭載された大出力モータ制御基板に関節角指令値が送られる。本論文ではこの従来型の体内通信リンクを slave link と呼称する。ノード間の通信リンク物理層には全二重 RS422 が用いられており、データレートは 20Mbps に設定されている。接続トポロジはマスタを中心としたライントポロジで各ノードはデイジーチェーン接続される (図 4.1)。また、データリンク層にはサーボノイズによって発生する通信エラーを訂正するためにブロック符号化手法としてリードソロモン符号エンコーダ・デコーダが備えられている。この符号化によるオーバヘッドは約 10μsec となっており、1msec での制御周期に間に合う設計とされている。マスタ主導での通信リンク管理手法であり、基本的にスレーブ間通信はサポートされておらず、隣接ノードとの協調制御用パケット (発行タイミングはマスタからのコマンドで決定) の送信のみに限定されている。つまり通信モデルとしては、RPC 型通信による同期通信が主となり、マスタースレーブによる1 対多接続構成となっている。接続トポロジは物理的にはライントポロジとなっているが、論理的にはリングトポロジ接続であり冗長構成が可能となっている。

図 4.2 に各制御ノードにおける FPGA 内部の slave link の PHY/MAC 構成及び、MPU とのデータ通信のためのバス接続アーキテクチャを示す。マスターから転送されてきたパケットは自ノードデバイスI D 宛であれば取得し、そうでなければ次ノードへと即座に転送する。図 4.3 に slave link で使用しているパケットのデータ構造を示す。パケット受信ブロックでは、コマンドを解釈しデータライトコマンドであればパケット内のデータを指定アドレスの内部メモリあるいはレジスタに書き込む。リード命令であれば、指定アドレスの内部メモリあるいはレジスタのデータを応答パケットとしてマスタに送り返す。これらの処理は全て FPGA 内の論理回路で実装されており、高い実時間性で処理される。これによって通信リンクの使用率を高い状態で保てるため、マスターインターフェース1 チャンネルあたりに6 ノード接続しても、1msec 周期の通信を実時間で処理するのに十分な性能を実現している。なお上記処理に最適化された設計でパケット処理ブロックの実装されている都合上、可変長パケットやスレーブ間通信には制約があり、利用可能なプロトコルはマスタースレーブ型の push・pull リクエストに限定されてしまう。

4.2.2 腱駆動ヒューマノイドにおける体内通信系の構成

分散型制御システムの別の実装例として腱駆動ヒューマノイド「 腱志郎」[21] について述べる。

腱志郎は全身で 105 個の腱駆動アクチュエータを持ち、それらを駆動するためのモータドライバとモータ制御基板が搭載されている。このモータ制御基板は、頭の中に搭載されている全身運動の筋長軌道指令を生成する主制御計算機と USB によって接続されている。USB を体内通信系に使用することで、USB の特徴である多ノード接続性と汎用インターフェースとしての可用性を得ることで、柔軟なシステム設計を

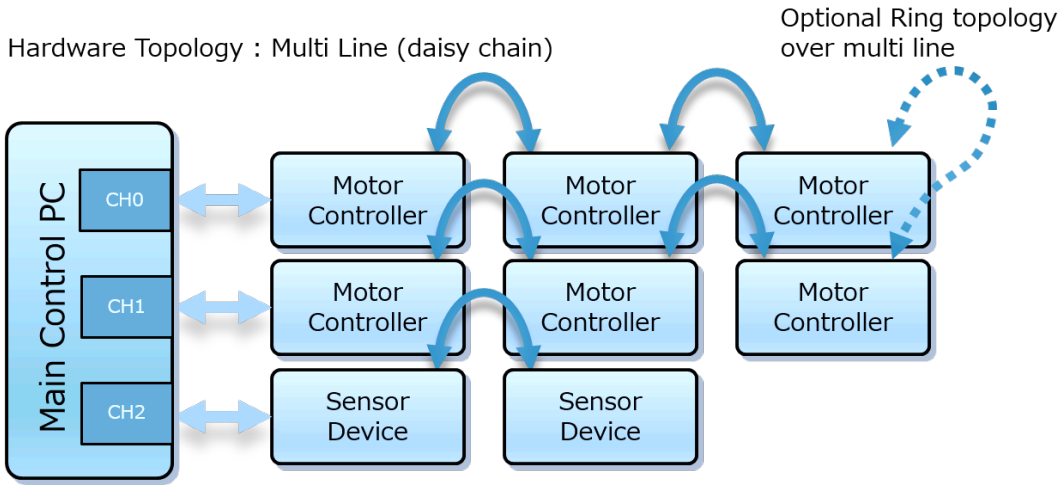


図 4.1: Former hardware topology of robot network.

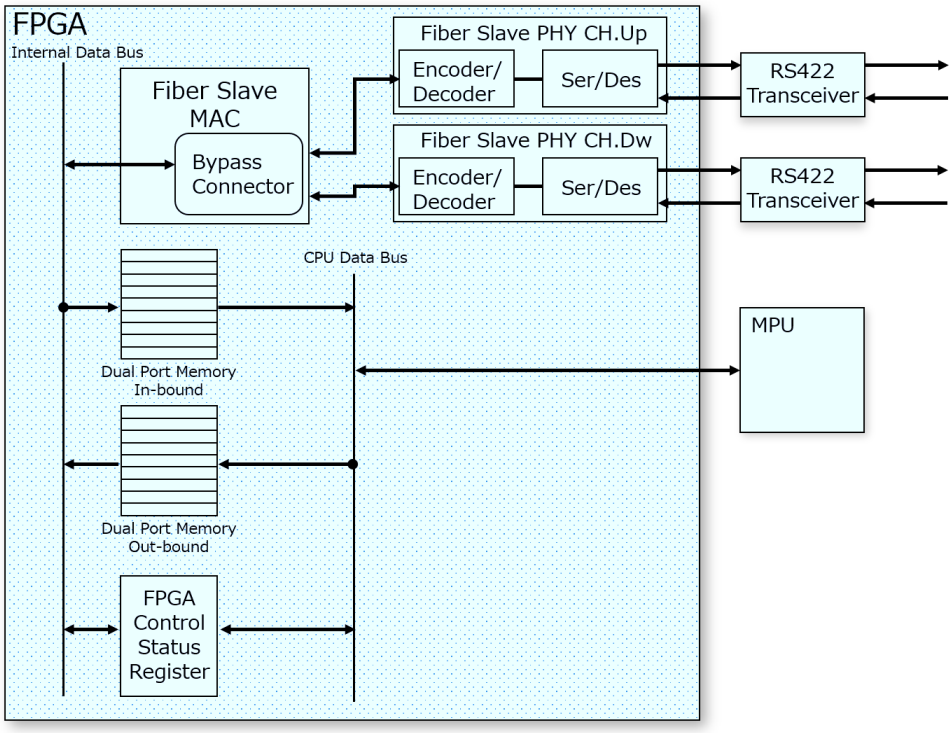


図 4.2: Fiber slave link architecture inside FPGA.

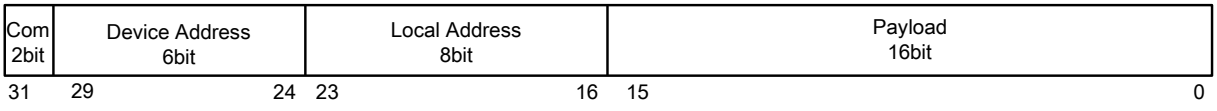


図 4.3: Packet data structure of slave link

可能としている。しかし、USB は規格上転送タイミングを制御することは困難であり、非実時間転送である。特に多ノード接続した場合には最低でも数 msec 周期を要してしまうため、力制御のようなタスクにおいてはネットワーク越しでのフィードバック制御が困難である。また、USB はマスタースレーブ型の通信構成のため、スレーブ間でのデータ通信を行うためにはマスターを経由させる必要があるため、さらにネットワーク遅延は大きくなる。腱駆動機構の特徴である拮抗筋の制御においては、拮抗筋のペアの制御器間で協調した制御を行うことでガタのない滑らかな動作が可能となるが、スレーブ間通信のできない通信リンク制約は、拮抗筋制御の点で不満のある構成となる。

また、制御基板とモータドライバはさらに RS485 をベースにした独自実装の実時間マスター・スレーブ間シリアル通信 (図 4.5) が実装されており、PWM 指令値の送信とアクチュエータセンサデータの取得を行っている。データレートは 3Mbps で、同期方式は調歩同期式、半二重通信という構成のため転送可能なデータ量は限られており、1msec 周期の制御を行う際の接続可能なデバイス数は4 ノードが限界である。

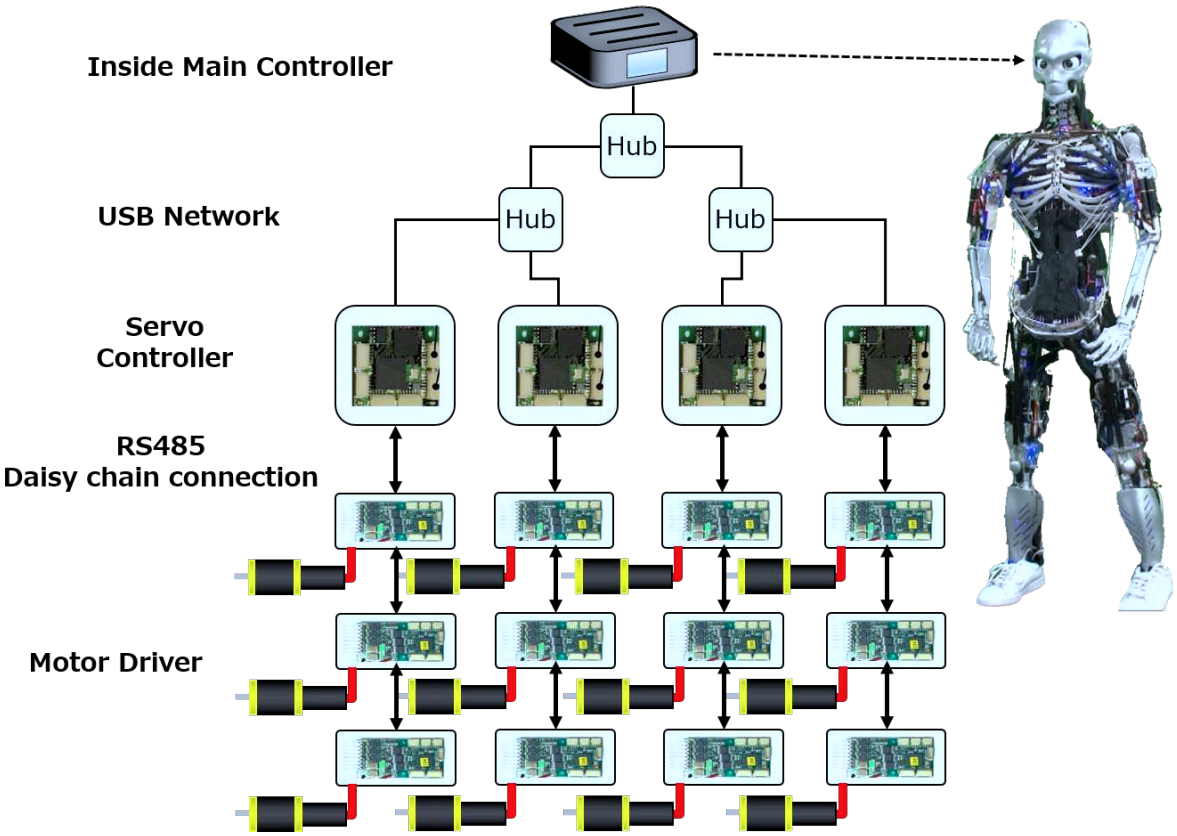


図 4.4: Control system architecture in Kenshiro.

4.2.3 その他のロボットにおける体内通信系の構成

上記に挙げたロボット以外にも多くの研究所・企業においてロボットが開発されている。体内システムの通信系として採用されているのが公開されている例では、CAN、EtherCAT、Space Wire、SERCOS、PROFINET 等が使用されている (表 4.1)。HRP3L-JSK のように独自の体内通信規格を使用するというケースは、Nagakubo らの ETL-Humanoid[52, 53] において採用例があるが、プロトコルまで含めた詳細について報告されている例は少ない。この例に限らず、体内のシステム構成、通信系の構成についてはスペックとして公開されていないケースがほとんどであり、メイン計算機のプロセッサやマイコンのスペックについての情報が公開されている程度である。これらは体内システムの構成については非公開扱いであるというわけではなく、ロボットの設計においては機械的な機構・アクチュエータの設計や制御アルゴリズムに主眼が置かれている場合が多く、それらを制御するシステムについては既存のシステムの流用であり関心が薄いため、あまり細かく言及されないということであると思われる。

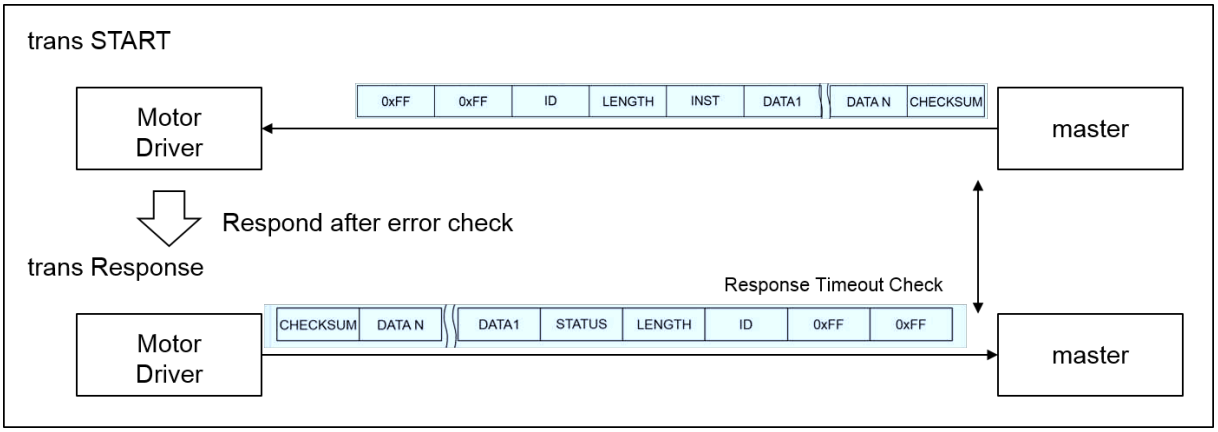


図 4.5: Transaction protocol between servo controller and motor driver.

4.2.4 既存通信規格と提案通信方式の比較

本節では、既存通信規格として特にヒューマノイドロボットの体内通信系として使用されている規格と本研究で提案する光アクティブコネクタと FPGA によるマルチプロトコル実装を利用した通信方式について比較を行い、本方式の位置づけを明確にする。

まずロボット体内における通信系に要求される要素として任意ノード間の通信における遅延性能が重要となる。遅延性能は通信系のデータレートとプロトコル方式、データリンク方式等の要因が影響をするため、これら要素全てにおいて、本研究で要求される通信遅延性能、通信周期、通信データ量を満足する方式となっている必要がある。

例としてここではロボットのモータドライバノード 1 つにつき毎周期 64 バイトのデータを送信するとし、ロボット 1 台に 32 ノード搭載されていることを仮定する。簡単な計算として表 4.2 に示すビットレートの通信系において、2048 バイト (=64*32) を完全に送信するのにかかる遅延時間を計算したものを表 4.2 の下段に示す。この表に示されるように、100Mbps 程度では仮にビットレートを完全にペイロードに割り当てることが可能だとしてもデータの送信に 100μsec 以上かかる計算となり、本研究の要求仕様である制御周期 200μsec やジッタ値 20μsec を満たすのはシステム全体では困難となる数字であることが分かる。制御周期に対して通信遅延が十分に小さい 20%以下にしようとすると、データレートで 400Mbps 以上が必要となる計算である。実時間性を考慮するとデータレートには余裕が必要となるため、ビットレート上は 1000Mbps 以上あることが望ましいと考えられる。本研究で採用する物理層は 2500Mbps 以上のビットレートを実現することが可能となり、ロボット体内のデータ通信量に対しても十分に低遅延で通信を完了させる性能があると言える。

以上を元に通信系のビットレートについて、図 4.6 にて比較を行っている。物理層方式やパケット構造によって多少の差異が出るがビットレートが高い通信系は基本的にデータレートも高くなると考えて問題ない。汎用的なマイコン等で使用でき、採用例も多い CAN や UART、SPI といったペリフェラルデバイス用通信規格等は通信データ量が、ヒューマノイドロボットの体内制御系で要求されるデータ量を想定しておらず、ビットレートも低いため本研究において要求を満たす規格ではないことが分かる。これらは1 トランザクションのみに限定すれば 100μsec 以内の通信も不可能ではないが、システム全体として実時間性を保証するのは困難である。同様に実時間性能の高いことで知られる EtherCAT 等の産業用 Ethernet 規格においても、ベースとなる規格が 100BASE-TX であるため、ノード数と通信データ量、制御周期の高速化に対してビットレートの点において性能不足となることが分かる。実際、本研究では汎用計算機と組込み系との間のインターフェースに EtherCAT を採用することを第 5 章にて述べるが、ノード数が多くなる全身型ヒューマノイドにおいてはビットレートの低さがボトルネックとなって制御周期を高められないことが報告されている。ギガビットイーサをベースとした産業用 Ethernet 規格が登場しコモディティ化した場合には、ヒューマノイドロボットの体内通信系として候補になり得ると考えられる。また、USB は USB3.0 以降の Super-Speed モードでは 5Gbps 以上のビットレートが利用可能となるが、安定性の問題や通信方式としての応答性の問題、あくまでもマスタースレーブ通信であるため、任意ノード間通信では遅延が増大するという問題があり、ヒューマノイドの体内分散系の通信系に採用するには技術的な課題があると考える。

表 4.1: その他ロボットにおける体内通信系

通信規格	概要	採用例
CAN	自動車で一般的に使用されている通信規格で、サポートしているマイコンも多数販売されている。バストポロジ接続で最大データレートは 1Mbps となる。	ARMAR-4[54]、モータドライバと主制御計算機の接続に使用。
EtherCAT	ハードウェア構成はイーサネットベースの実装となっており、産業用ロボットのコントローラモジュール間の接続等に採用されている。任意トポロジ構成が可能で、データレートは 100Mbps となる。	産業用ロボット、PR2, Justin[55] のベースリンク通信に使用。
Spacewire[56, 57]	衛星機用に開発された通信規格であり、非実時間通信であるが低遅延で、データレートは 1Gbps となる。	Justin、DLR Hand[58] に使用される。
PROFINET	Ethernet 技術ベースの実装となっており、実時間· 非実時間通信の混在した通信を行う。Isochronous Real-Time(IRT) と呼ばれる実時間通信品質を設定している。帯域の時分割によって実時間通信と非実時間通信を両立している。専用 ASIC によって実現される。	産業用ロボット
FPGA による独自実装	本研究や 4.2.1 節にて述べたのと同様に FPGA を利用した独自のプロトコル構成によるノード間通信の構成であり、通信遅延の低減を狙ったものである。	ETL-Humanoid[52, 53]

表 4.2: Bit-rate configrations and latency.

Bit Rate (Mbps)	1	100	400	1000	2500
Latency (μsec)	16384	163	41	16	6.6

あるいは、汎用的なマイコン等で通信処理を行う場合、通信頻度の増大に伴ってプロセッサ負荷が増大するために、本来マイコンで処理したい演算処理の実時間性能が損なわれるという問題がある。そのため、計算資源の少ない組込み向けプロセッサではそもそも高速通信インターフェースが必要とされず、処理性能に見合った速度の汎用的なインターフェースのみ搭載されていると考えられる。

以上の問題点の多くは、当該通信規格がターゲットとしているアプリケーションに対して、ヒューマノイドロボットの通信系への要求が低遅延性能について特に厳しいことにあると考える。従って、汎用的な通信規格ではなく FPGA 等を利用して専用通信リンクを採用することが合理的であると考えることが可能となる。従来は 1000Mbps を超えるような高速通信インターフェースについては、高度な開発能力· 環境が要求された。そのため、UART やソースシンクロナス通信等の限定的な速度の通信機構か、あるいは専用 PHY

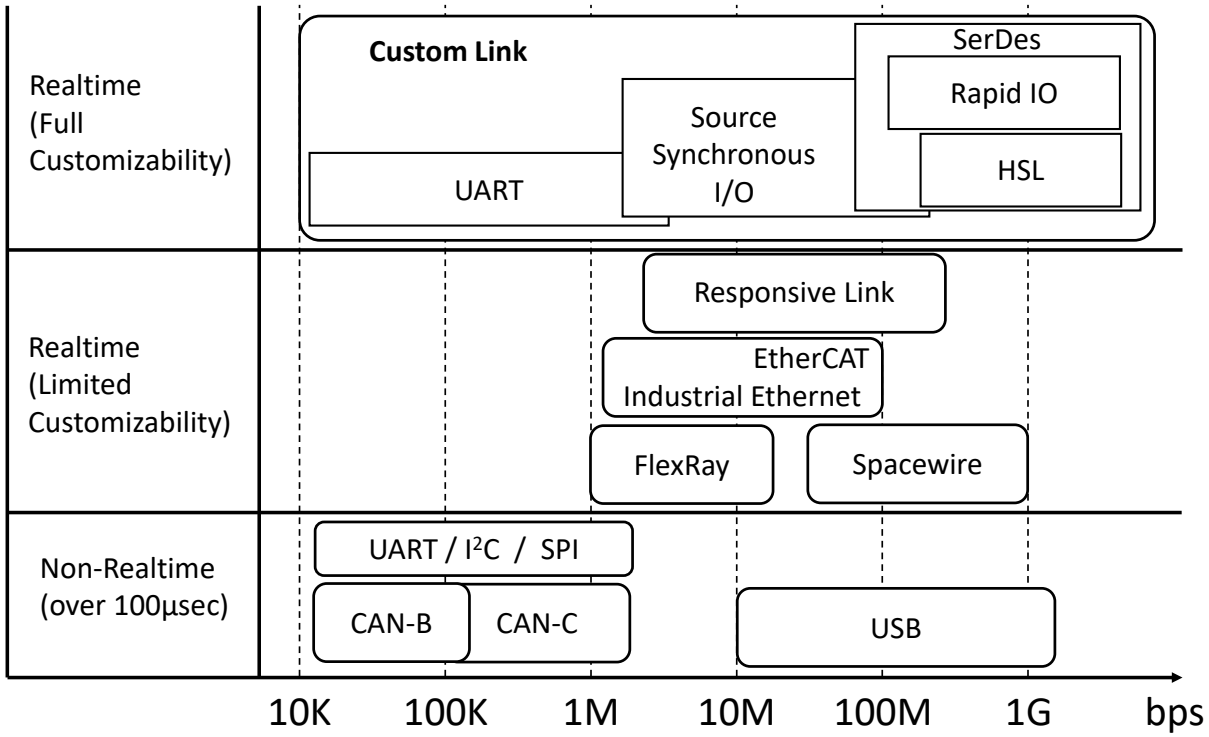


図 4.6: Comparison of major communication links used for humanoid's internal system.

を別途実装する必要があった。高速トランシーバ PHY はデータセンター・サーバ等の高トラフィックシステムに適用されることが前提となっている場合が多く、ロボットの体内システムに採用するにはコストや消費電力・パッケージサイズの観点から適合するものを選定するのは困難であった。現在では FPGA に搭載された高速トランシーバ PHY 等を利用することによって、研究室レベルにおいても比較的簡易に高速通信系の実装を行うことが可能な環境となっている。また、FPGA だけでなくハイエンド DSP にも Rapid IO[1]と呼ばれる高速シリアル通信インターフェースが搭載されるようになってきており、これらを利用してユーザ側で高速通信系を実装することが可能である。本研究で提案する通信リンク（ High Speed Link, HSL）も FPGA の高速トランシーバを物理層として、さらに FPGA ハードウェアロジックによってデータリンク層以降の処理も実装したものとなる。FPGA で通信系を構成すると、通信に関連する処理についても FPGA 側でハードウェアロジック処理を行うことでプロセッサ自体の負荷の増大を抑制するという効果も見込まれる。2.8.2 節にて述べた2 つの通信モデルは、このような通信処理と専用ハードウェアロジック処理の関連付けにおいて必要となる要件となる。

4.2.5 従来のロボット体内通信系における課題

従来使用されてきた体内通信機構においても、ロボットの制御を実現することは可能であった。しかしながら、本研究のように制御周期を向上させ、応答性を向上させることを目指していく場合には、分散ノード間でのデータの授受の頻度は必然的に増大する。また、2.8.1 節にて述べたように、ロボットにおいては電磁ノイズ、熱、衝撃、振動等への耐環境性能や接続構成の柔軟性が求められる。以下ような点で標準化された通信規格では本研究で要求される仕様を満たすことができず、独自の通信規格を用いる動機となっている。

データレート

データレートは図 2.8 にて示したデータを用いると、全二重型通信系で1 ノードあたり約 168kbps 必要となる。符号化やヘッダに必要な帯域を考慮すると約 612kbps となり、12 ノードで 7.3Mbps、40 ノードで 25Mbps 程度のデータレートが要求される。自動車向けの CAN ではデータレートが不足する計算となる。

[1]http://www.rapidio.org/

これは従来型のシステムでの数値であり、制御周期を 5kHz まで向上し、送受信されるデータ長や種類を増加された場合、約 1.6Mbps 程度まで増大すると想定する。1 ロボットあたりに 12 ノードのモータドライバがあるとすれば、約 20Mbps であり、40 ノードならば 65Mbps まで増大する。符号化やヘッダに必要な帯域も考慮すると、およそ 73-245Mbps のデータレートが必要になる計算となる。実時間性や低遅延性を保証するためには帯域に余裕が必要となるため、実際にはこの数値より大きいデータレートが適正な数値となる。必要データレートは通信系に接続されるノード数によってスケーラブルとなるが、上記の計算より実用的には 100Mbps 以上は最低でも必要な値となる。一方、産業用 Ethernet では 100BASE-TX が標準となっている場合が多いため、データレートは 100Mbps となってしまう。そのため、他ノード接続された場合に帯域不足となり、実時間性能に影響すると考えられる。データ転送量を分散させるために、通信系1 チャンネルあたりに接続するノード数をデータレートに適した数に制限した上で、多チャンネル化して対応されてきた。この方法の場合、チャンネル数に応じたインターフェースコネクタや配線数の増加、チャンネル間通信のサポートといった課題が生じる。

実時間性能、低遅延性能

実時間性能、低遅延性能については規格によって異なるが、Ethernet ベースの規格の場合 MAC フレームの最大長は 1500Byte=1.5kByte. 100Mbps で 120usec の時間がかかる。そのため、例えば PROFINET では伝送周期は最小単位を 250usec としており、本研究で要求される 5kHz の制御周期に適用するには不十分と言える。フレームサイズは実際には 1.5kByte も必要無い場合が多いため、現実にはさらに細かい時間粒度での伝送周期は設定可能であるはずだが、ライブラリ等では対応していない、或いは想定していないということも多い。

耐環境性能

耐環境性能という観点では、コネクタの耐環境性能が重要となる。汎用の USB コネクタや Ethernet 用の RJ45 型コネクタでは耐衝撃· 耐振動· 耐防水という点で信頼性が劣るため、産業用グレードのコネクタが使用できるようにインターフェースを選定する必要がある。電磁ノイズについては、メタルワイヤを使用する場合にはどの規格においても問題となりうる。Ethernet は耐ノイズ性能の高い物理層を持っており、信頼性が高いと言える。ただし、耐ノイズ性能の高いケーブルは次に述べるケーブル構成の点において問題が生じる。

ケーブル構成

ロボットの体内通信系に使用するケーブルでは、屈曲性能や摺動耐久性、細さが重要となる。屈曲性能では耐屈曲回数や最小曲げ半径が小さいことが要求される。また、摺動耐久性ではケーブルを覆うシースの素材や厚みが要素となる。産業用にこれらの性能を高めたケーブルは生産されているが、主に大型の産業用ロボット用に設計されているため、ケーブル太さが本研究で対象とするヒューマノイドロボットの体内に配線するには太すぎるという問題がある。特に Ethernet 用ケーブルはツイストペアケーブル4 組8 本が必要となり、シールド等と合わせると最も細いケーブルでも ϕ6mm 程度はあり、空間制約の大きいロボット体内では不適である。Ethernet 用ケーブルは硬く、最小曲げ半径が大きいものが多いことも問題となる。

接続構成

産業用 Ethernet ではディジーチェーン接続やスイッチングハブ等で柔軟な接続構成が可能になっている場合が多く見られる。ロボットではディジーチェーン接続によるライントポロジが構成できれば十分であるため、スイッチングハブ等の補助的なネットワークデバイスが必要無い規格であることが望ましい。USB や Spacewire は多元接続のためにはハブによる接続が前提となるため、この点で不利となる。

4.3 低遅延体内分散通信系のための通信リンク設計

4.3.1 物理メディア層

RMTP-02D 基板に搭載した光アクティブ型インターフェースは、光ファイバ内を通す光を伝送媒体として通信を行う (図 4.8)。実際の物理インターフェースとなるトランシーバは FPGA 側に内蔵されており、光アクティブコネクタの光電変換モジュールと FPGA 側トランシーバ間は AC カップリングされた 1.5V PCML[2]信号によって伝送されている。光アクティブコネクタは原理的に電磁ノイズが通信に混入しない仕組みであるが、厳密にはこの光電変換素子とトランシーバ間でノイズが混入する可能性がある。そのため、配線パターンはGNDガードを施した上で最短配線長となるようコネクタと FPGA の配置に配慮を行っている。また、光アクティブコネクタ自体も周辺ノイズを拾わないよう、内部変化回路はスカート構造の金属筐体でシールド処理を施されている (図 4.7)。

表 4.3 にその他の基本仕様を示す。光アクティブコネクタは1 本のケーブル内に2 本のファイバーを通している構造となっており、TxRx の全二重通信が可能となっている。RMTP-02D 基板ではこれを2 チャンネル分搭載しており、デイジーチェーン接続によって複数ノードでライントポロジを構成可能となっている。また、ケーブル自体のデータ転送可能なビットレートは最大 6Gbps であるが、RMTP-02D 基板で採用している FPGA 側のトランシーバの最大ビットレートが 3.125Gbps のため、こちらが設計上の最大ビットレートとなる。このような高速信号では配線のインピーダンスマッチングは不可欠であり、RMTP-02D 基板では 100 Ωで設計されている。実際に使用時のビットレートは内部トランシーバの設定によって可変となっており、RMTP-02D 基板では FPGA に入力しているクロックから PLL によって生成しやすい 2.50Gbps に設定を行っている。

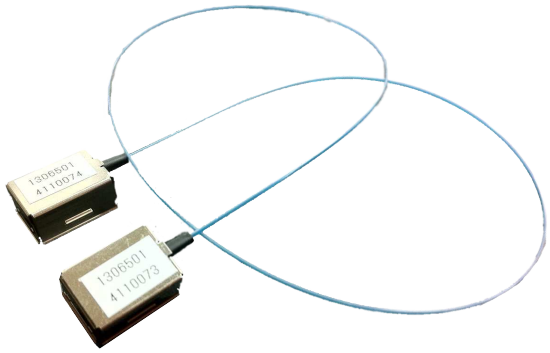


図 4.7: Active optical connector module

表 4.3: RMTP-02D 基板における光アクティブインターフェースの実装

# of channels	2
Data flow	Full duplex
Data Rate	2.5Gbps (3.125Gbps MAX)
FPGA I/O	AC-coupling 1.5V PCML

[2]Pseudo Current Mode Logic

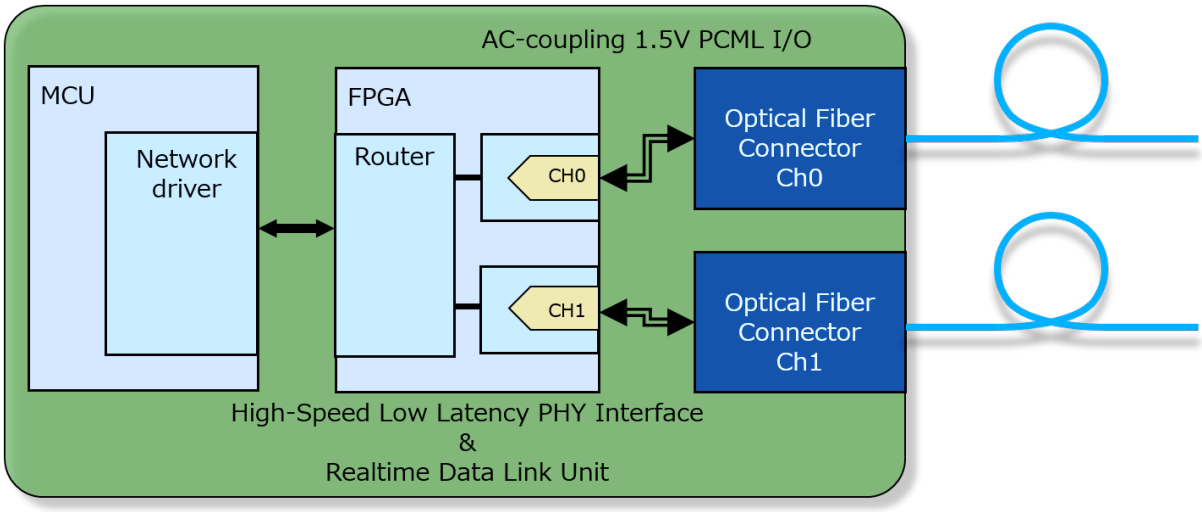


図 4.8: Active optical interface implementation on RMTP-02D Board.

4.3.2 物理層

物理層ではトランシーバとしての機能を提供しており、RMTP-02D 基板では FPGA 内蔵の高速トランシーバ PHY を使用して物理層を構成している。

トランシーバ PHY では論理回路側で処理されているワードデータについて、シリアライズ前のエンコーディング・デコーディング処理 (Physical Coding Sublayer, PCS) を行い、シリアライザ・デシリアライザによる電気信号レベルのシリアルデータとの間の変換 (Physical Medium Attachment, PMA) を行うという 2 つの処理を主に行う [59]。トランスミッタとレシーバそれぞれの PHY 内部構成を図 4.9 及び図 4.10 に示す。

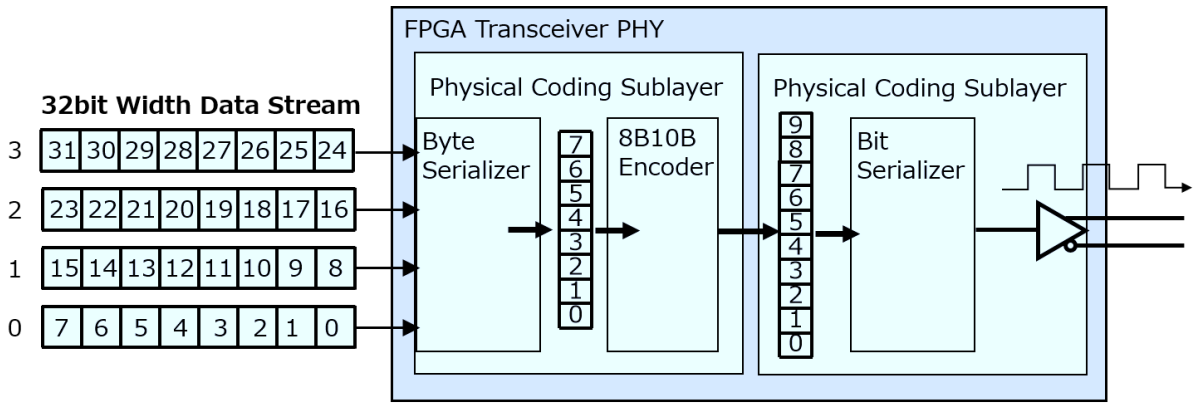


図 4.9: Internal architecture of transmitter PHY.

トランスミッタ側 PCS では上位の層から入力されてくる送信ワードデータストリームについて、

- ワードデータをシリアルバイトデータに変換
- 8B10B 符号エンコード
- エラー検出・訂正用符号化

を行い、PMA でシリアライズするための前処理を行う。

ワードデータ・シリアルバイトデータ変換処理は、FPGA 内部回路の動作周波数とトランシーバ PHY の動作周波数の差を埋めるためのモジュールである。2.5Gbps のビットレートを最大限使用する場合、単純

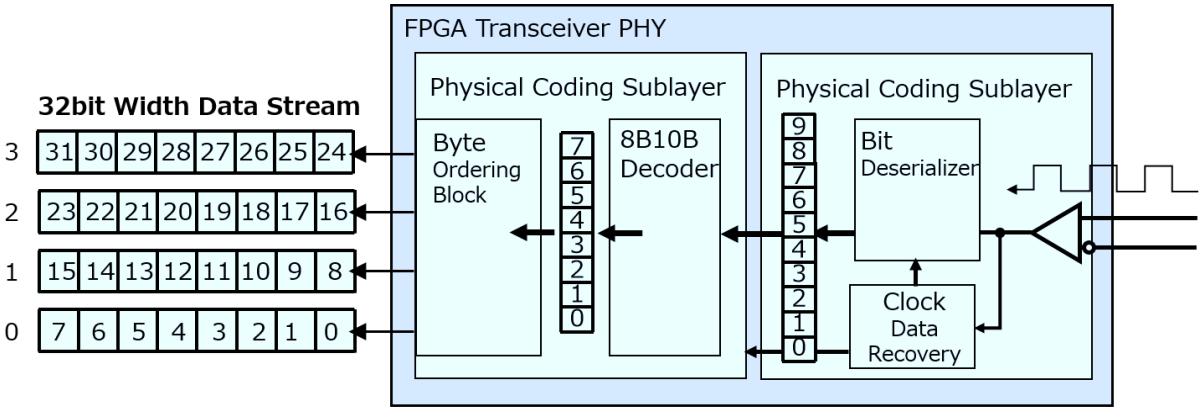


図 4.10: Internal architecture of receiver PHY.

計算で 312.5MByte/sec のデータレートで FPGA 内部のユーザロジックとトランシーバ PHY の間でデータ転送されなければならない。8bit 長のデータ幅で転送される場合、312.5MByte/sec の転送速度を達成するにはロジックの動作周波数は 312.5MHz が要求される。しかし、現在の FPGA のアーキテクチャでは内部ロジックの動作周波数は最大でも 300MHz 付近であり、データ通信ロジックのような規模のロジック回路をこの最大周波数で駆動することは現実的ではない。これが、32bit 長のデータ幅で転送される場合には 78.125MHz まで低減され、実用的な動作周波数となる。32bit 長データは PCS に渡された後、8B10B エンコードのためにバイト単位にシリアライズされる。

8B10B エンコーディングは 8bit 長のバイトデータを 10bit 長の冗長データに変換することで、転送データ内に送信クロックを重畳させる処理である。また、冗長な符号化によって転送制御用信号を転送データとは独立に設定することが可能となる。シリアル通信では受信側でどのようにビット受信用のクロックを生成するかが重要となる。UART 等の低速なシリアル通信で一般的に使用される調歩同期通信では、トランスミッタとレシーバで送受信のビットレートの設定をあらかじめそろえてあれば、同期用クロックを分配することなくビット信号に同期して受信することが可能である。しかし、信頼性の高い UART を実現するには、トランシーバ側の駆動クロックをビットレートの 4 倍から 16 倍程度は確保する必要があり、現実には数十 Mbps 程度のビットレートが限界である。SPI 等のクロック分配配線を持つ同期通信方式の場合でも、高速信号ではクロック信号線とデータ信号線の間に生じるスキューによって、信頼性の高い通信を行うことが困難となる。一方、8B10B エンコーディング等によって高速シリアル通信ではシリアル信号列自体にクロックが埋め込まれているため、レシーバ側で PLL によって元のクロックを復元することが可能である。このように受信用クロックを生成する手法はクロックデータリカバリ (CDR) と呼ばれ、一般に高速シリアル通信用トランシーバで使用されている。FPGA 内蔵のトランシーバ PHY を利用する利点として、これらのリカバリークロック用の配線が用意されており、貴重なクロック用内部配線を余計に消費しないということが挙げられる。

また、エラー検出･ エラー訂正用符号化もこの層にて行うことが可能となる。上位レイヤにおいてあらかじめ符号化しておくことも可能であるが、エラー発生場所はおおよそ物理メディア層であるため、この層においてエラー訂正･ 検出を行うのが最も最速である。

RMTP-02D 基板におけるトランシーバ PHY のコンフィギュレーションを表 4.4 に示す。

4.3.3 データリンク層

物理層インターフェース

物理層の機能により、上位層はワードデータストリームを物理層に流し込むことでデータ送信を行い、受信側ではワードデータストリームが流れ込む形となる。単純なデータストリームではデータ構造の境界が不明確であり、データ通信を行うことができない。データリンク下位層では、データストリームにデータリンクプロトコルのための制御構文を付け加えることによって、

- 通信リンクのコネクションの確立

表 4.4: RMTP-02D 基板におけるトランシーバ PHY コンフィギュレーション

Transmitter Configuration			
PCS Configuration		PMA Configuration	
PCS interface width	32bit	PLL type	CMU
FIFO mode	low latency	Data rate	2500Mbps
Use Byte serializer	True	Reference clock frequency	100.0MHz
Use 8B10B encoder	True	Local clock division factor	1
Receiver Configuration			
PCS Configuration		PMA Configuration	
PCS interface width	32bit	CDR reference clock frequency	100.0MHz
FIFO mode	low latency	Data rate	2500Mbps
Use Byte Ordering	False		
Use Byte deserializer	True		
Use 8B10B decoder	True		

- フロー制御
- 送受信データのパケット[3]化

を行う。これにより上位のプロトコル層やアプリケーション層からは送受信データはパケットの形でやり取りできるようになるため、用途に応じた適切なプロトコルの設計を容易に行うことができる。図 4.11 に示すように、Start of Packet 信号と End of Packet 信号によってデータストリームのパケットの区分けが可能なため、上位プロトコルはわざわざデータストリーム中からパケットの取り出しのための規約を設計する必要はない。

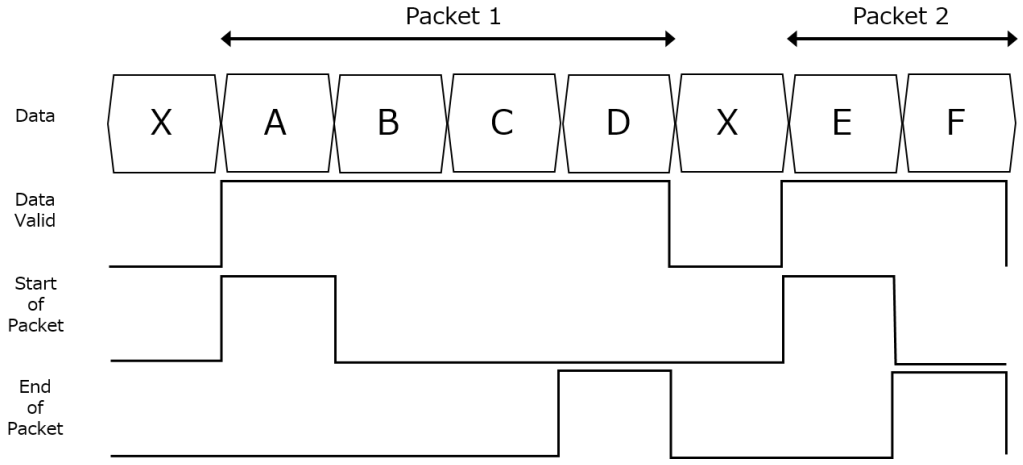


図 4.11: Packet data separation by start of packet signal and end of packet signal.

RMTP-02D 基板では下位データリンク層の実装として、IF.HOTARU[60] IP を用いた。IF.HOTARU は高速光伝送におけるインターフェースを簡易に扱えるように設計されており、転送レイテンシは小さくや FPGA 上の回路規模も約 1000LE 程度と非常に優れたパッケージとなっている。自動での通信リンクのコネクションも行えるため、簡単にホットプラグ機能も実装可能である。

図 4.12 に、RMTP-02D 基板の物理層及びデータリンク下位層の構成図を示す。IF.HOTARU IP は送受信チャンネルごとの一つ実装されており、それぞれトランスミッタ PHY とレシーバ PHY の間でワード

[3]通常、フレームとパケットはどの層でのデータ単位となるかで使い分けされるが、本研究ではどのプロトコルにおいてもデータはカプセル化されずに送受信さるため、特に使い分けは意識せず、全てパケットと呼称する。

データストリームをやり取りする。また、4 つのトランシーバ PHY はそれぞれ独立したクロック系統で駆動されており、そのクロックに同期してデータストリームのシンク・ソースを行っている。これら 4 つのデータストリームをユーザクロックで駆動される FPGA 内部回路の間でやり取りするために、図 4.13 に示すように非同期 FIFO による位相差吸収を行っている。

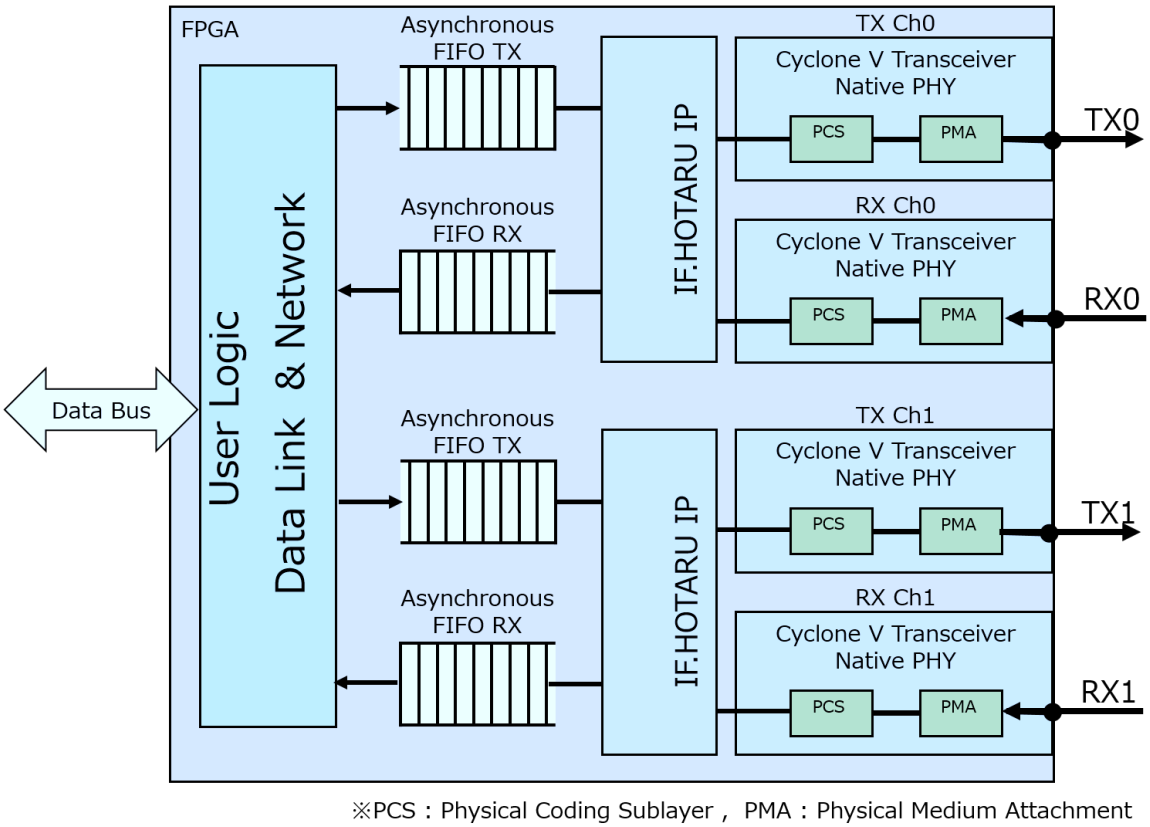


図 4.12: Data link layer architecture in RMTP-02D 基板.

4.3.4 プロトコル優先度に基づいた MAC 方式による低遅延通信

通信リンクの1 物理メディア上を伝送できるデータは通常同時に1 つまでであり、同一のケーブル上を複数のノードから同時にデータ送信を行うことはできない。そのため、複数ノード間でのデータ通信や、複数データを送受信する場合には通信リンクの物理層へのアクセス権を調停するための仕組みが必要となる。この仕組みは Media Access Control(MAC) と呼ばれ、OSI 参照モデルにおいては物理層の一つ上の第2 層に位置するデータリンク層の役割に相当する。MAC には大きく分けると2 通りの手法が考えられており、

1. アクセス要求の衝突時に動的にアクセス権を調停
2. アクセス権の割り振りを事前に決定

の2 つとなる。(1) は通信要求発生時に即座にアクセス要求が発生するイベントトリガー的手法で、アクセス権の競合を前提としている。(2) はアクセス権はシステム全体で管理されており、自ノードのアクセス権が回ってくるまでは待ちが発生するが、確定的なタイミングで通信が行われる。アクセス権を管理するマスターとアクセス権を与えられるスレーブの関係にあたり、アクセス権の競合を発生させない前提となっている。実時間性の観点からは、アクセス権管理手法の方が通信までの時間を事前に予測することができる点で有利であり、周期的に送受信されるデータについては高い転送効率を実現できると考えられている [61]。しかし、マスター・スレーブの関係が生じることによりロボットの分散型システムの観点からは、1. 即時性の要求されるデータ送信に遅延が生じる、2. 非周期的なイベントトリガータイプのデータの送受信につい

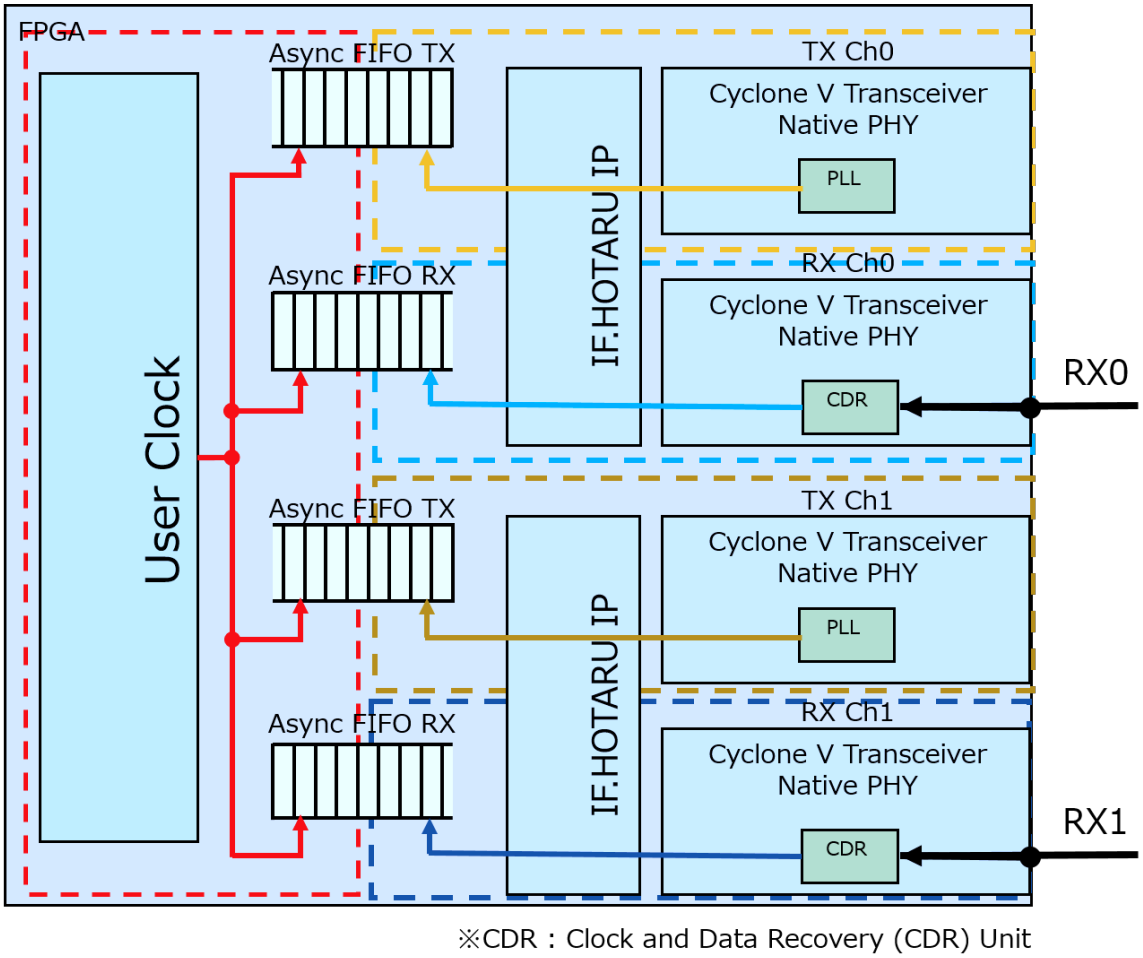


図 4.13: Clock domain of data link layer in RMTP-02D 基板.

てもあらかじめスケジュールされてしまうため、リンクの効率が悪化する、3. ノードの追加・転送データの追加でアクセス権管理マスターの再スケジューリングの発生による拡張性の悪化、といった欠点も考えられる。データの転送に実時間性を優先するか、低遅延性を優先するかでどちらの手法を採るか決定される。

広く知られている MAC 方式としては次の方式が挙げられる。

- CSMA/CD(Carrier Sense Multiple Access / Collision Detectionr)
  Ethernet のような1 系統の通信リンクを複数のノードで共有している通信規格で用いられる MAC 方式で、各ノードは転送開始時に通信リンク上で衝突が発生していないかを監視し、衝突が発生した場合にはランダム時間の待ち状態の後、転送の再開を試みる方式である。ノード数が多くなるにつれて、衝突確率が上がり転送効率が激減するため、近年の Ethernet ではほぼ使用されていない。

- CSMA/CA(Carrier Sense Multiple Access / Collision Avoidance)
  CSMA/CD と違い、通信リンクが空き状態であることを確認した後、ランダム時間だけ待ち状態を経てから転送を試みる方式である。ノード間での競合発生確率を下げる方法であるが、完全に競合が回避されるわけではない。

- CSMA/BA(Carrier Sense Multiple Access / Bitwise Arbitration)
  CAN-bus で使用されている MAC 方式であり、CSMA/CD と同様に衝突検知を行う。ただし各ノードの送信データの上位ビットは優先度が符号化された形式となっており、アクセス衝突時には優先度の低いノードは次ビットから送信を停止し、優先度の高いノードは引き続きデータ転送を行う。

- TDMA(Time Division Multiple Access)
  TDMA ではネットワーク中のノードは同期されたグローバルクロックを持つ。このグローバルクロッ

クを基準に、時分割されたスロットに対して各ノードのアクセス権が付与されることによってデータ送信の調停を行う。グローバルクロックのノード間同期のためのプロトコルが必要となる。CAN をベースに TDMA プロトコルを実行可能にした TTCAN や、自動車の制御システム用実時間通信として開発された FlexRay が挙げられる。

- Token Bus
  トークン・リングに代表される MAC 方式の一種である。アクセス権はトークンを所持しているノード・パケットに与えれられ、トークンを順次受け渡していくことで調停を行う。通信リンクアクセス時の衝突は発生しないため、転送効率を高めることができる。トークンの受け渡しのアルゴリズムによって転送開始タイミングが左右され、トークンが回ってくるまでは優先度の高いデータであっても転送を開始することができない。

- bus request/ bus arbitration
  ノード間通信とは少し異なるが、CPU バスのようなデバイス間の通信において、バスに接続されたノードはバスマスターとバスリクエスト信号· バスアクノレッジ信号で接続されている。バスマスターはバスリクエスト信号を元に、ノードに対してアクセス権の付与をバスアクノレッジ信号によて伝送することで調停を行う。

- Master/Slave
  ネットワークの転送は全てマスターで管理される方式である。マスターはネットワーク中のノードに対して逐一コマンドを送信することによって、アクセス権を付与する。CPU バスマスター方式とは異なり、マスターにスレーブ側からの転送要求を伝送する手段が無いため、送信タイミングはマスター側が全て決定する。

ロボット体内の通信リンクの場合、データ転送粒度は制御周期に実時間で同期した 1kHz から 10kHz 程度と細かい区切りとなる。また、一つのパケットあたりのデータペイロード長は多くの場合数バイト程度であるが、種々のタスクからのデータ転送に対応が可能なよう、可変長パケットでのデータ転送がサポートされていることが望ましい。また数 k バイト単位での大容量転送はビジョンセンサ等のデータに限定される。ヒューマノイドのような等身大ロボットの体内に搭載できるノード数は高々100 ノードくらいである。ロボットの分散制御系のための通信リンクとして考えた場合、MAC でのアクセス権の調停が低遅延で実行され、転送要求の生じたデータは即座に転送開始され、次の制御周期以内にデータ転送が完了していることが求められる。また、可変長パケットの転送によって不確定時間の待ち時間が入ってしまうことは実時間性の面から好ましくない。低遅延性と実時間性の両立を目指した MAC 方式として、優先度付き並列キューを以下のように設計した。

優先度付き並列キュー

MAC に求められる要件として、1. 低遅延性、2. 実時間性については前述した。低遅延性では、データ転送準備の完了したパケットについては即座にデータストリームとして送られることが望ましい。一方、実時間性については、データ転送の開始から完了までが確定的な時間で見積もることができ、最悪実行時間内に収めることが求められる。

低遅延性を阻害する要因としては、先行して転送開始したパケットの転送終了待ちが考えられる。これに対して、*Responsive Link* の設計において採り入れられていたパケットの優先度に応じて転送の順序を決定するという方針を採用することを考える。低遅延性を優先するパケットについては高い優先度を付与することによって、先行するパケットが転送途中であっても、転送切替によって即座に転送を開始できるようにスイッチを構成する。

また実時間性を阻害する要因としては、可変長パケットの存在、及び、イベントトリガー的データ転送の割込みの存在によって転送待ち時間に不確定性が出てしまうことがある。これについても同様に実時間性を要求するパケットについては可変長パケットよりも高い優先度を付与することによって、待ち時間の不確定性を解消することを考える。

以上をまとめると優先度分けとして次のように割り振ることができる。

1. 低遅延· 実時間 (ハードリアルタイム)· 固定長パケット

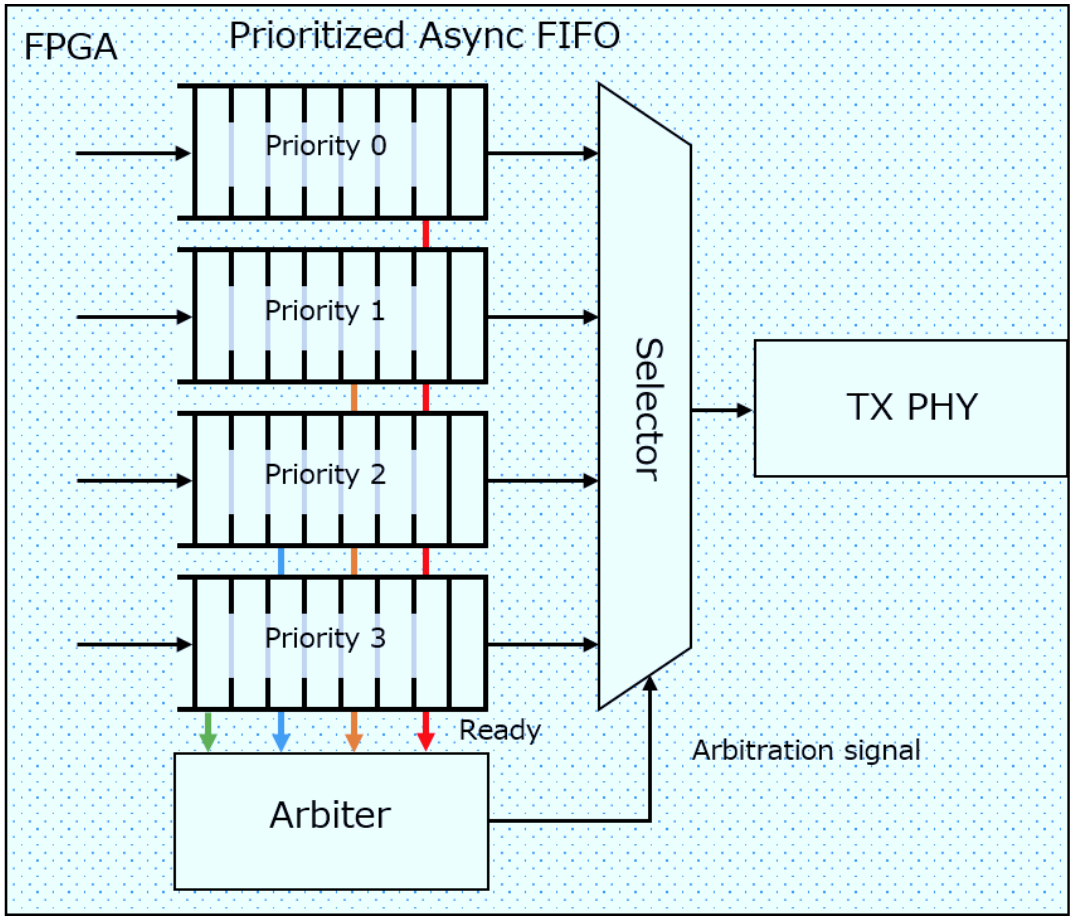


図 4.14: Transmitter Side MAC layer of RMTP-02D board.

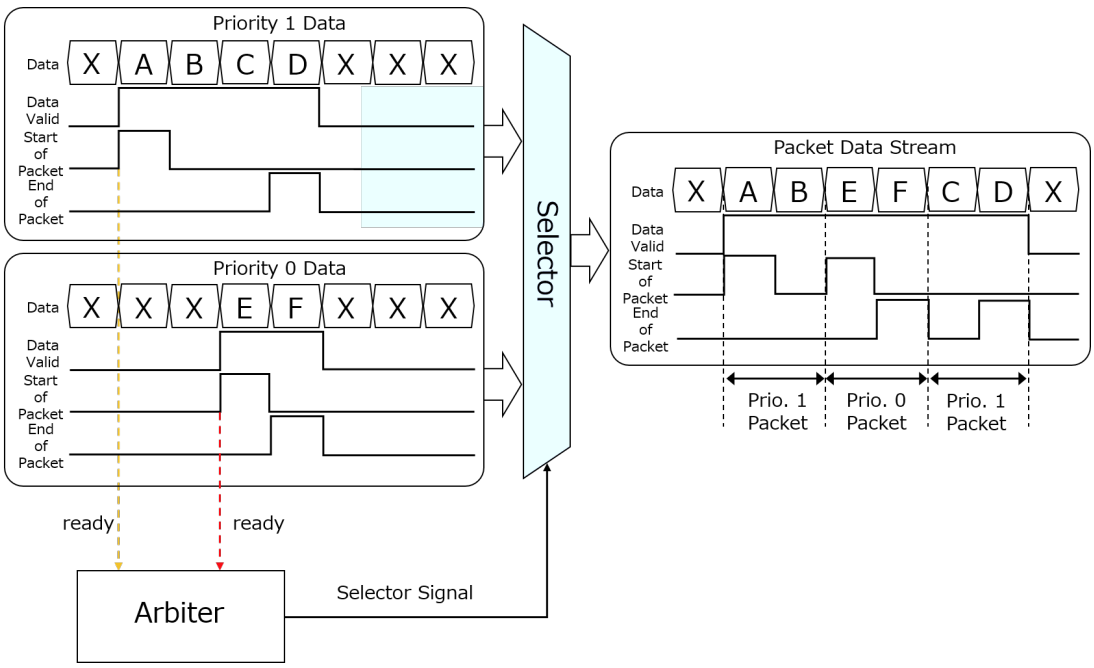


図 4.15: Data stream arbitration mechanism by prioritized queue array.

2. 非低遅延・実時間 (ハードリアルタイム)・固定長パケット
3. 非低遅延・実時間 (ソフトリアルタイム)・可変長パケット
4. 非低遅延・非実時間・可変長パケット

実時間性のために、上位優先度パケットは固定長パケットに限定される。低遅延性が要求されるパケットについては最高優先度として、即座に転送が行われる。また、遅延性能についての要求は低いが、実時間性が要求される場合には固定長パケットでの転送を次点優先度で行う。可変長パケットは実時間性への影響が大きいため、低優先度として設定している。

以上を元に、優先度による MAC モジュールの設計を行う。図 4.14 にトランスミッタ側優先度付き並列キューによる MAC の構成を示す。MAC は四本の非同期 FIFO によってパケットのキューを構成している。また、各キューからは送信開始要求信号が出ており、アービタへと入力される。各キューには優先度 (priority) 0 ～3 を割り振っており、優先度0 が最高優先度に設定している。アービタは、送信開始要求に応じてキューセレクタを切り替えることによってどのキューからのパケットをデータリンク下位層へデータストリームとして送信するか決定する。アービタは毎クロック事に送信開始要求を監視しており、現在転送中のパケットがいる状態においても、高優先度のキューからの転送開始要求が発行された場合には即座にスイッチの切替を行うことで転送パケットの途中割込みを遅延無く実現している (図 4.15)。また、レシーバ側において途中割込みパケットの復元を行うためのキューインデックス情報がパケットのヘッダに付与される。

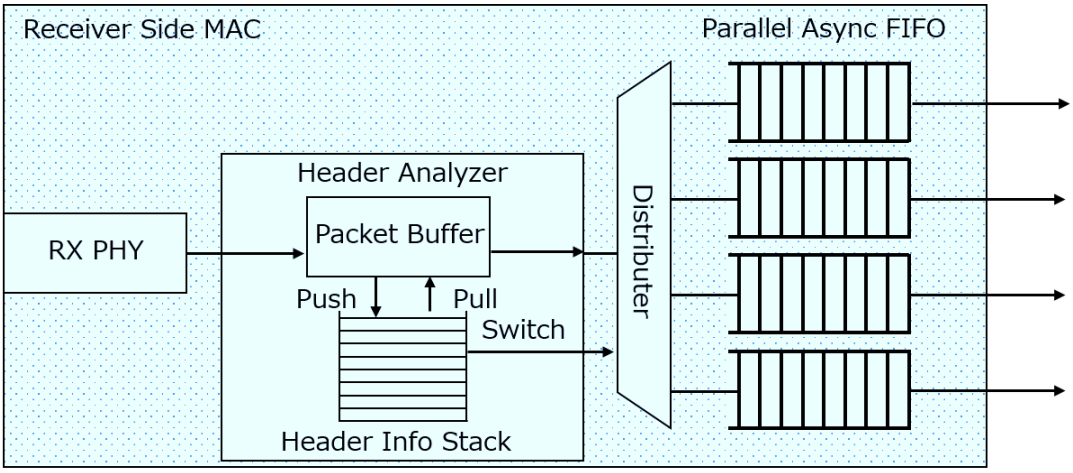


図 4.16: Receiver Side MAC layer of RMTP-02D board.

次にレシーバ側における並列キューの構成を図 4.16 に示す。レシーバ側にもトランスミッタ側と同様に 4 本のキューを用意してあり、送信時に付加されたキューインデックスに応じてディストリビュータによって遅延無しでプッシュされる。ディストリビュータによる振り分けは、直前のヘッダアナライザモジュールにおいてパケットヘッダ中のインデックス情報を解析することにより行われる。転送割込みによるパケットの分断によって、プッシュするキューのインデックス情報を失わないために、パケットのヘッダ情報はスタックバッファに一時保存される。割込みパケット受信時に先のパケットのヘッダ情報はスタックに積まれ、割込みパケット転送完了と同時にスタックから取り出されることで、シームレスなパケット転送を可能としている。

以上、RMTP-02D 基板上に構成した通信リンク系の物理層およびデータリンク層について述べた。ここまでの構成を図 4.17 に示す。優先度付きキューによる MAC を構成したことによって、物理・データリンク層は上位層からは4 本の独立した通信リンクが各チャンネルに存在しているかのように扱うことが可能となっている。従って、上位のモジュールにおいては他のプロトコルに配慮せずに自由なセッション・トランザクションを行う構成で設計を進めることが可能となる。

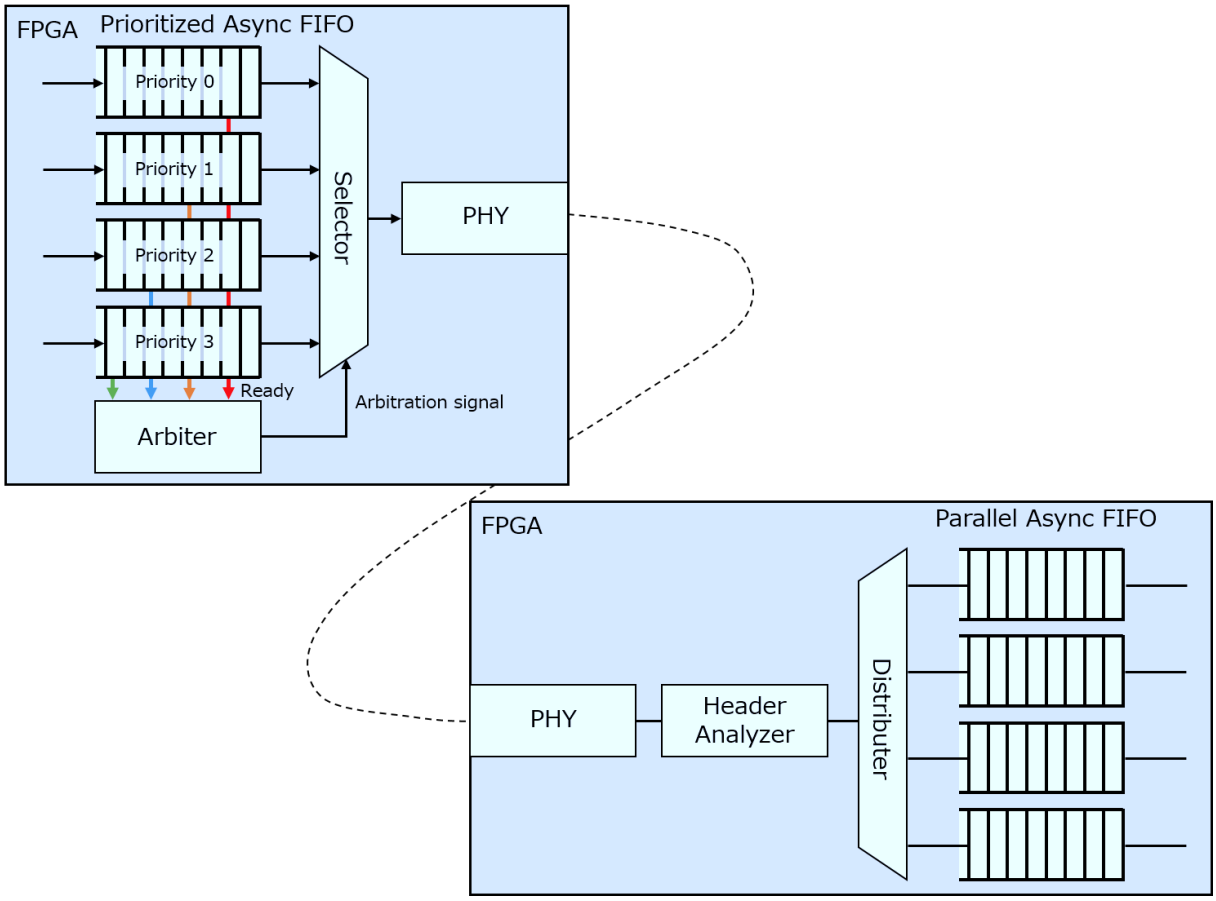


図 4.17: PHY and MAC layer of RMTP-02D board.

4.4 低遅延体内分散通信系のプロトコルブロック設計

4.4.1 プロトコルブロック基本構成

プロトコルブロックでは、ノード上に構成されるサブシステムの処理で要求されるデータ転送・コマンド転送を、転送プロトコルに従って処理するための論理回路が実装される。プロトコル規約に従ったデータ処理は逐次処理や条件分岐を伴うこともあるため、プログラムとして MPU 側で処理を行うことで実装の手間を削減することも考えられる。しかし、MPU がパケットデータを直接処理を行うのは、パケットデータのメモリ上への展開にオーバヘッドが生じる問題や、論理回路での実装で得られる実時間性・低遅延性には及ばないといった観点により、プロトコルによるパケットデータ処理についても論理回路によって実装することを目指す。

本研究ではロボット体内通信系で使用する通信モデルとして、RPC 型及びデータストリーム型を採用することを 2.8.2 節にて述べた。これらを低遅延なハードウェアロジックとして実装するために、シンプルな通信プロトコルで記述する必要が生じる。

RPC 型通信モデルを実現するプロトコル

別ノードの制御レジスタの読み書きや割込みトリガーのために、同期型通信を行う RPC 型通信モデルが採用される。多くの通信規格では、マスタースレーブ通信のような、マスターとなるノードとその他スレーブノードの間で非対称に行われる実装であった。しかし、分散型制御システムを構成する上で、任意の2ノード間で対称に RPC 型通信を行えうことで反射制御系を組み込むといったことが実現可能となる。CAN や I^2C 等のバス共有型の通信の場合、任意2 ノード間での RPC 型通信を行うことが可能であるが、実時間性の観点でバス共有型はデータ衝突の可能性があるため不利である。

本研究では上記の点を踏まえ、リング型論理トポロジ接続された固定長軽量パケットによる **Light RPC 型プロトコル**を採用することとする。リング型論理トポロジは、接続された全ノードを物理的な接続トポロジに依らず一筆書きでたどり、リング型トポロジと同様にパケットが流される構成であり、slave link や EtherCAT 等の実時間通信規格においても利用される構成である。このトポロジでは、データフロー方向が常に一方向なため、ルーティング処理による遅延やデータフローの偏りが生じない。そのため、通信にかかる時間や遅延時間が決定的となり、実時間性が要求される同期型通信として望ましい挙動となっている。リング型論理トポロジ構成とすることで、ノード数が多くなる場合には遅延時間が比例して大きくなるが、光通信化によって通信にかかる遅延は無視できる範囲で低減されているため、実用的な構成となっている。このことは 4.4.5 節にて詳細を述べる。

また、固定長の軽量パケットを使用することでさらに通信時間や遅延時間は決定的となる。RPC 型通信では push、pull、interrupt の3 種類の同期処理を行うことを述べたが、これらの処理に限定されていることでパケットフォーマットを最小限に構成することが可能となり、通信負荷の小さい固定長軽量パケットを実現できている。

データストリーム型通信モデルを実現するプロトコル

多ノード間で多種のデータについてデータストリーム型通信を構成する方法として、本研究では**分散共有メモリモデル**を採用する。データストリーム型通信では非同期でのデータの送信となるため、受信側では受信バッファを備えておく必要がある。一般的には、受信側ではこの受信バッファから適宜受信データを取り出し、パケットの解釈を行ってデータを取り出す処理を行う。この処理はプロトコルスタックソフトウェアの処理となるため、プロセッサの負荷となる。

ところが、ロボットの制御においてデータストリームで受信させたい制御データは多くの場合、即値として必要となる型であると考えられる。従って、最新時刻の受信データより古い場合には破棄されてしまっても問題が生じない。そのため、バッファ長は1 で十分と考えられる。分散共有メモリモデルを採用した場合、メモリの各アドレスが長さ1 のバッファに対応すると見なすことができる。そうすると、各アドレスに対してデータ型を各ノードにて対応させることで、アドレスサイズに対応した数のデータストリームが構成される。また、分散共有メモリへの書き込みは他の全てのノードへと伝播するため、1 対多通信をモデル化する上でもプロトコル処理としては単純化される。この分散共有メモリ処理はハードウェアロジックとし

て十分に小さいサイズで構成可能である。ただし、分散共有メモリであるため不必要なノードに対しても同様にデータストリーム通信が構成されてしまうため、通信系の効率は低下してしまう。本研究の場合は、光通信化によるデータレートの大幅な向上が副次的に達成されているため、通信系の効率低下を補うことが可能である。また、通信系の効率を考える場合には、体内の主要部分ごとにサブネットマスクを導入することで分散共有メモリパケットの伝播範囲を制限することで達成できると考えられる。

分散共有メモリによる複数制御器間での通信の特徴としては、

- 制御器間のプロセスはデータ転送処理を同期させる必要がなく、通信終了待ちによる実行時間のジッタが小さいため、短い最悪実行時間に対しても実時間プロセスのオーバーランを起こしにくい。
- プログラム側でメモリ上に展開する構造体の定義を変更することによって、柔軟に転送データを変更することが可能である。
- 複数ノードからの同アドレスに対する同時書き込みは、分散共有メモリプロトコルの転送タイミングによってメモリ間で書き込まれるデータが異なるハザードが起こる可能性があるため、書き込みブロックのための排他制御処理が必要となる。
- 通信エラーが発生していてもメモリリード自体は行えるため、信頼性を確保するためには別途ハンドシェイクプロトコル等を実装しておく必要がある。

上記の特徴から、分散型システムの制御器内の非同期プロセス間でのデータストリーム型メッセージ転送手法として有効であると考えられる。

通信系を構成するプロトコルブロック

上記2 つのプロトコルに加えて、分散型制御システムを構成上で有用と考えた3 つのプロトコルを合わせて5 つのプロトコルを、FPGA 内のハードウェアロジックとしてプロトコルブロック化することを考える。

1. ルーティングテーブル初期化プロトコル
   パケットの振り分け先を決定するために必要なルーティングテーブルを動的に決定する(オプション)。
2. グローバルクロック同期プロトコル
   分散ノード間で時刻を高精度で同期させる。
3. Light RPC 型プロトコル
   Push、Pull 型の同期通信を行う。
4. 分散共有メモリプロトコル
   分散ノード間で共有メモリを構成する。通信はメモリへのデータコピーとなる非同期通信となる。
5. コネクション型プロトコル
   セッションを張ることで柔軟に通信アプリケーションに対応する。一般的なソフトウェアによる通信処理に対応する。

各プロトコルの詳細な実装については 4.4.3 節以降にて順に述べていく。

要求通信品質に基づいた優先度付き並列キューへのプロトコル割り当て

それぞれのプロトコルに要求される通信品質要素(優先度、通信周期・頻度、パケットサイズ・パケットフロー) を表 4.5 に示す。さらにそれぞれのプロトコルをどの優先度付き並列キューに割り振ったか、そのインデックスを合わせて掲載する。

- ルーティングテーブル初期化プロトコル
  このプロトコルについては、初期化時のみ実行され、その間は他のプロトコルによる通信も行われる必要が無いため、どの並列キューへ割り振っても影響はない。

- グローバルクロック同期プロトコル (NTP)
  このプロトコルは、その性質から実時間要求が極めて高い。従って、最優先で送信をする必要があるため、最大優先度の並列キューに割り振っている。この通信の頻度は制御系の通信と比較して低くパケットサイズ・フローも最小限に抑えられているため、制御系通信を阻害する恐れは無いと言える。

- Light RPC 型プロトコル
  このプロトコルは制御コマンド等の同期通信による送受信に用いられ、ハードリアルタイム通信用途にも用いることを想定しているため優先度は高い。パケットフローはノード数に比例して多くなるが、パケットサイズは固定サイズで小さいため、他の通信の阻害する時間は長くはならない。そのため、NTP の次の優先度の並列キューに割り振っている。

- 分散共有メモリプロトコル
  このプロトコルでは、データは制御周期ごとに各ノードからブロードキャストされるため、データフローが非常に多くなっている。また、バースト長設定によってパケットサイズは可変長となっているため、実時間性を要する通信を妨げないために優先度は Light RPC 型よりも低い並列キューに割り当てた。最悪実行時間をオーバーした場合に致命的な失敗につながるようなハードリアルタイム通信については同期通信で実行されることが理想的である。従って、非同期通信である分散共有メモリプロトコルではソフトリアルタイム通信をサポートし、優先度を一段下げることは妥当である。

- コネクション型プロトコル
  このプロトコルでは、汎用的な通信アプリケーションをサポートするために用意されている。そのため、実時間性能については問われず、優先度が最も低い並列キューに割り当てる。実時間性能とのトレードオフによって、パケットサイズについては可変で最大 4kByte 長まで利用が可能となっている。

表 4.5: Communication quality requirements for each protorol.

Protocol	Priority	Cycle	Packet Size (Flow)	Priority Queue Index
Rooting Table Init.	Low (Not RT)	Only at initialize	24Byte (Low)	0
NTP	Very High (Hard RT)	40msec	24Byte (Low)	0
Light RPC	High (Hard RT)	every Control Cycle (over 1kHz)	16Byte (Many)	1
Distributed-SHM	Middle (Soft RT)	every Control Cycle (over 1kHz)	24-156Byte (Stream)	2
Connection	Low (Not RT)	depends on application	MAX 4096Byte (depends on application)	3

プロトコルブロックデータアクセス方法

以上のプロトコルを実行する各プロトコルブロックは、それぞれ独立してペリフェラルデバイスとして FPGA チップ内のインターコネクト上に接続される形で構成する。主となる制御プロセスが実行される MPU からは、インターコネクトを介したバスアクセスによってそれぞれのプロトコルを制御する。

MPU からのアクセスはメモリマップド I/O として扱うことが多く、頻繁にアクセスをするレジスタ・メモリのアドレスは連続でかつアラインされたアドレス空間となるよう設計を行うことで、Direct Memory Access Controller (DMAC) による高効率なデータ転送を行うことが可能となる。そのため、連続アドレス

空間のアクセスで一連のプロトコル処理、データ送受信処理を行えるように各プロトコルブロックのデータアクセス方法を工夫しておく。

4.4.2 基本パケット構成

図 4.18 に本研究で実装した通信リンクで使用しているパケット構造を示す。パケットは 32bit 長のワードを1 単位として、32 × N[bit] 長の可変長パケットを基本構造としている。その内第1 ワード目に各プロトコルで共通のヘッダ情報を載せている。

1. キューインデックス：2bit
   前述した MAC においてどのインデックスの並列キューを使用するか。
2. トランシーバチャンネル：2bit
   送信元のトランシーバチャンネル情報。
3. パケット長：10bit
   ヘッダを含めたパケットの全長をワード長で示す。最大 1024 ワードのパケットの転送が可能。受信側のプロトコルブロックでトランザクションの終了判定に使用。
4. 送信先デバイス ID：7bit
   送信先のデバイスの ID。ネットワークルーティングに使用。
5. 送信元デバイス ID：7bit
   送信元のデバイス ID。ネットワークルーティングに使用。
6. パケットプロトコル型：4bit
   パケットデータ部のプロトコル型を示す。プロトコルごとに定数が設定され、プロトコルスイッチでのルーティング等に使用。

また、パケットの終端ワードは前方誤り検出用の XOR チェックサムとなっており、パケット各ワードの Exclusive-OR を採ることで簡単なエラー検出を行えるようになっている。CRC 等の誤り検出･訂正用符号と比較して信頼性は低いが、小規模な回路のみで遅延無しで計算可能である。実際の運用上の誤り発生頻度と、論理回路実装面積･速度とのトレードオフを考慮することでどのような誤り検出･訂正を行うか検討される必要がある。

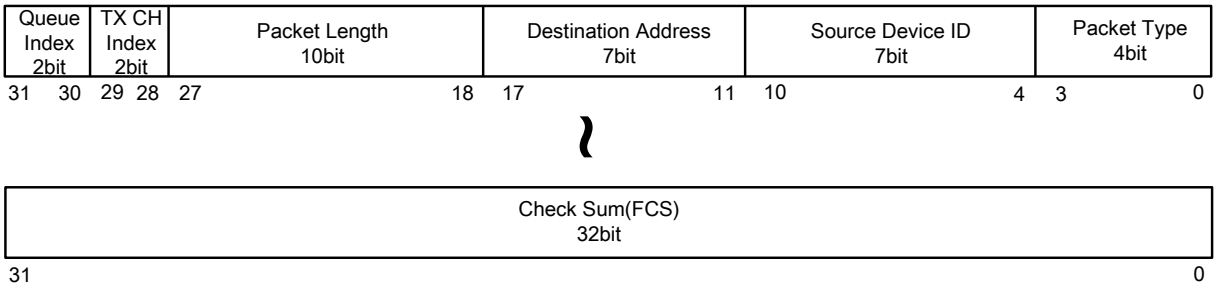


図 4.18: Definition of packet header.

4.4.3 ルーティングテーブル初期化プロトコルブロック

多ノード間で通信を行う場合、ネットワーク内でのパケットルーティングを行う必要がある。ルーティングのためには、各ノードがネットワーク構成についての情報を持っている必要がある。あらかじめ静的に実ネットワーク構成を ROM に焼きこんでおく方法が一つであるが、著しく拡張性に欠けるため、動的にネットワーク構成について確認しルーティング情報を獲得することが望ましい。

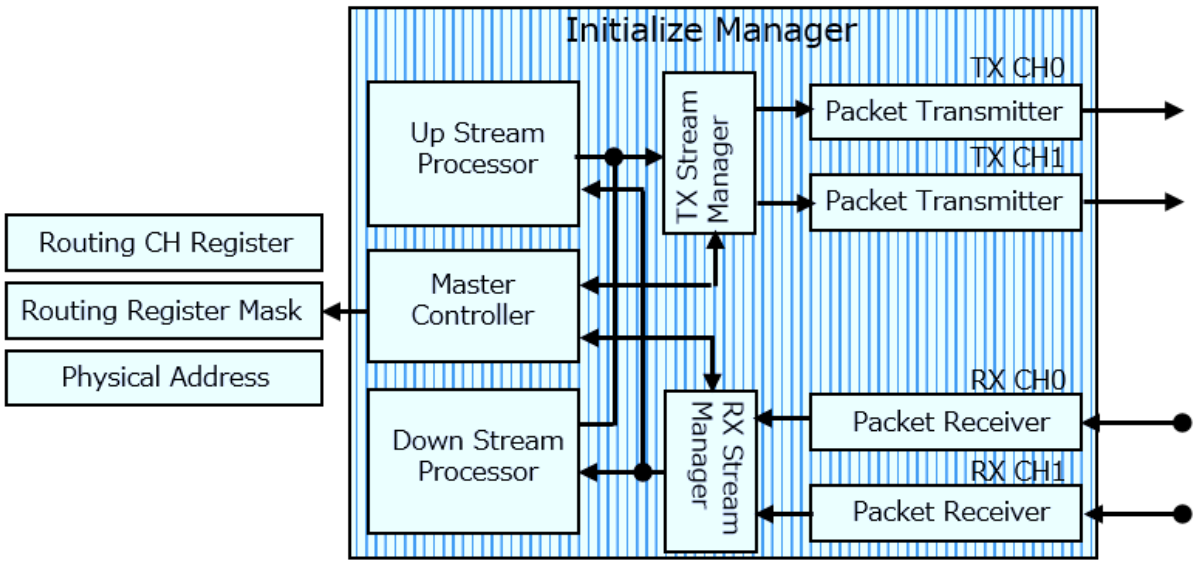


図 4.19: Schematics of Routing Table Initializer Protocol Block.

ロボットの体内通信系程度の小さいネットワーク規模であれば、シンプルでネットワーク構築速度が速いアルゴリズムが適切と考えられる。体内通信システム上のノード数は高々100 程度であることを考慮すると、全ノードへのルーティング情報をテーブルとして各ノードに持たせ、パケットルーティング時にはそのテーブルを参照させるという方法で十分であると考えられる。

従って、RMTP-02D 基板の通信リンクでは電源投入後にルーティングテーブル初期化プロトコルをまず実行し、ルーティングテーブルが準備された時点でネットワークは利用可能状態として、その後の各種プロトコルの実行状態に移行する。ルーティングテーブルの初期化プロトコルとしては、スパニングツリーの構成による物理トポロジチェックアルゴリズムを採用する。ルーティングテーブル初期化モジュールの構成を図 4.19 に示す。なお、RMTP-02D 基板では通信リンクのチャンネル数は2 つであるため、これらのみでネットワークを構成した場合にはライントポロジで固定されてしまうため、厳密にはスパニングツリープロトコルである必要はない。他基板への拡張性を考慮してスパニングツリーアルゴリズムとしている。

なお、このプロトコルは起動時のみに動作すれば十分であり、またロジックが比較的複雑であるため FPGA での実装では回路面積を多く消費してしまう。さらにツリー型トポロジを採用しない場合には、ルーティングテーブルがそもそも不要である。そのため、プロトコルブロックとしてハードウェアロジックとして実装するのではなく、ソフトウェアプロトコルスタックとして実装し、内蔵の MPU からコネクションプロトコルを通して実行するという実装方法や、このプロトコルブロックを省略するといった手法を採ることも、実際のハードウェア資源との兼ね合いで採用してもよい。

4.4.4 グローバルクロック同期プロトコルブロック

分散ノード間では、電源投入後からクロックをカウントすることで時間計測を行うことができる。この各ノード上で計測されている、システム起動からの時間を**ローカルクロック**と定義する。このローカルクロックは、電源投入タイミングの違い、クロックの周波数精度の違いによってノード間で異なる値となる。

しかし、計測データの取得時刻をログとして記録しておいて、後でノード間のデータ比較を行う場合や、ノード間で同期してデータ取得を行う、といったことを行う場合に、分散システム内で共通の時刻を参照できると都合が良い。

従って、グローバルクロックの基準となるマスターノードを一つ設定し、他のノードはマスターノードをクロック同期プロトコルサーバとしてクロック同期を行い、全てのノードでグローバルクロックを共有している状態を作る。

クロック同期プロトコルとして、RMTP-02D 基板では NTP[14] を採用した。最優先度キューを使用するデータストリームとしてクロック同期パケットの送受信を行うことによって、パケットの通信遅延は小さ

く、ジッタの小さい実時間通信として扱えるため、数クロック未満でのノード間同期クロックを取得可能である。

図 4.20 及び図 4.21 にクロック同期のための NTP パケットの構成を示す。図 4.20 の NTP リクエストパケットには、このパケットの送信時刻 t^s_{req} を埋め込んで送信を行う。図 4.21 に示す NTP リクエストパケットの 4,5 ワード目は応答パケットと通信遅延をそろえるためのダミーデータを挿入したものである。NTP 応答パケットには、このダミーデータ部分にサーバ側の送信時刻におけるグローバルクロック t^m_{global} が挿入されてから返送される。本来 NTP の応答パケットには、NTP リクエストパケットの受信時刻 t^m_{req} 及び、NTP 応答パケットの送信時刻 t^m_{ack} を含めることになっているが、最優先度パケットとして通信リンクを使用できるように設定したことによって、サーバ側での処理によるジッタの発生は0 に抑えることが可能なため、送信時刻 t^m_{ack} のみの返送で十分である。通信リンクの伝送による遅延も、パケットサイズをリクエスト・応答で同一の長さにしたことにより1 クロック以内に抑えることができている。クライアント側では、NTP 応答パケット受信時刻 t^s_{ack} を記録した後、以下の式でグローバルクロック t_{global} とローカルクロック t_{lobal} の差分 t_{offset} を計算する。

$$t_{offset} = t^m_{global} - \frac{t^s_{ack} + t^s_{req}}{2} \tag{4.1}$$

$$t_{global} = t_{local} + t_{offset} \tag{4.2}$$

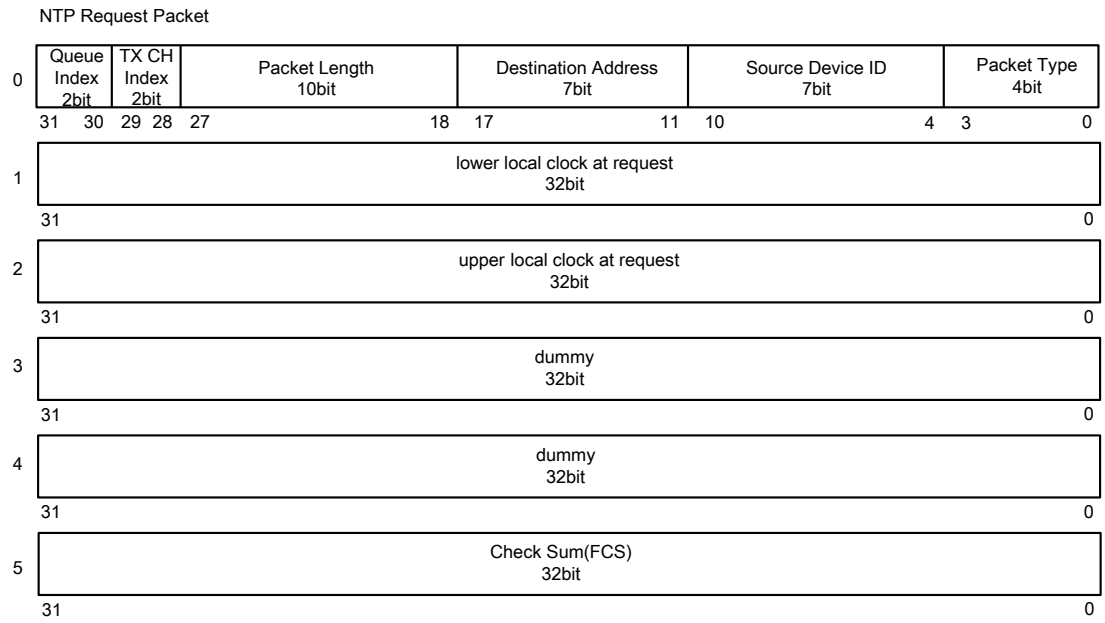


図 4.20: Packet structure of NTP request packet.

以上の構成によって、NTP によるグローバルクロック同期精度としてノード間で1 クロック以内の精度を実現している。モジュールの駆動クロックは 100MHz で設計しているため、時間にして 10nsec 未満の精度であり、IEEE1588 プロトコルによる EtherCAT 等でのクロック同期精度と比較しても十分に高い水準を達成している。RMTP-02D 基板で使用した発振器の周波数安定性は公称値で 25PPM である。周波数は 100MHz であるため、

$$100\mathrm{M} * \frac{1}{25\mathrm{PPM}} = 40[\mathrm{msec}] \tag{4.3}$$

より、40msec で 1 クロック分 (=10nsec) 時刻誤差が生じる計算となる。従って、NTP による同期は 40msec 周期で実行すれば十分であると分かる。これは制御周期 1kHz(=1msec) や 5khz(200μsec) と比較して十分に通信頻度が低くなるため、通信優先度を高くしても問題が生じないことを裏付ける。

クロック同期プロトコルブロックの論理回路構成を図 4.22 に示す。

なお、インターネット上の NTP サーバを利用するなどの方法で世界標準時を基準としたシステム内グローバルクロックを作成することが考えられる。インターネット接続が可能な上位のシステムが一般的な NTP によって世界標準時に同期した上で、グローバルクロック同期プロトコルのクロックマスターに設定

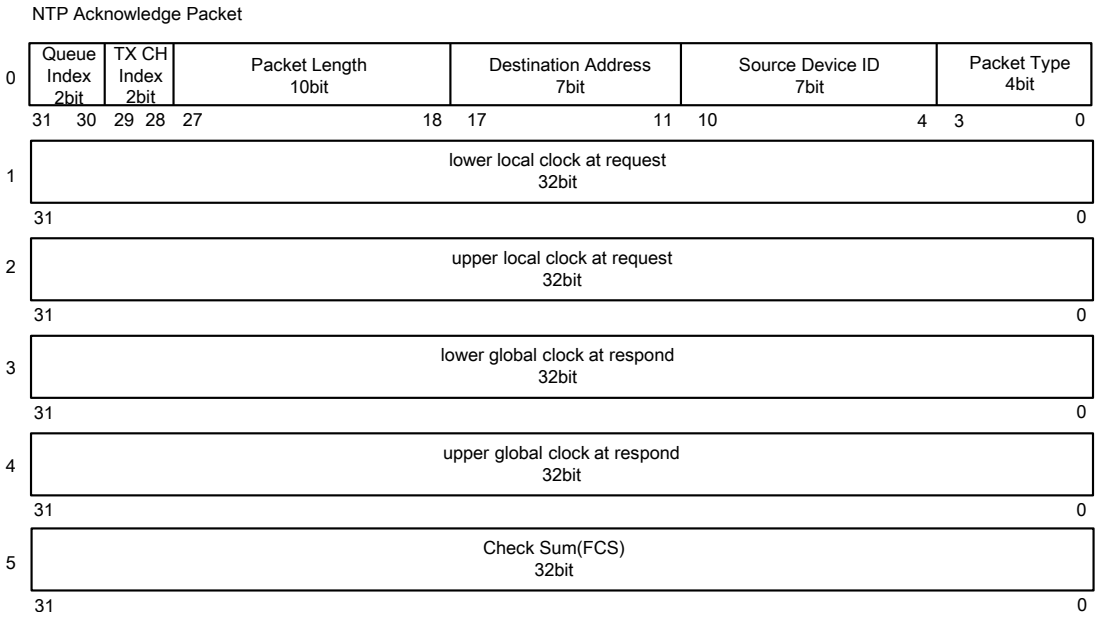


図 4.21: Packet structure of NTP acknowledge packet.

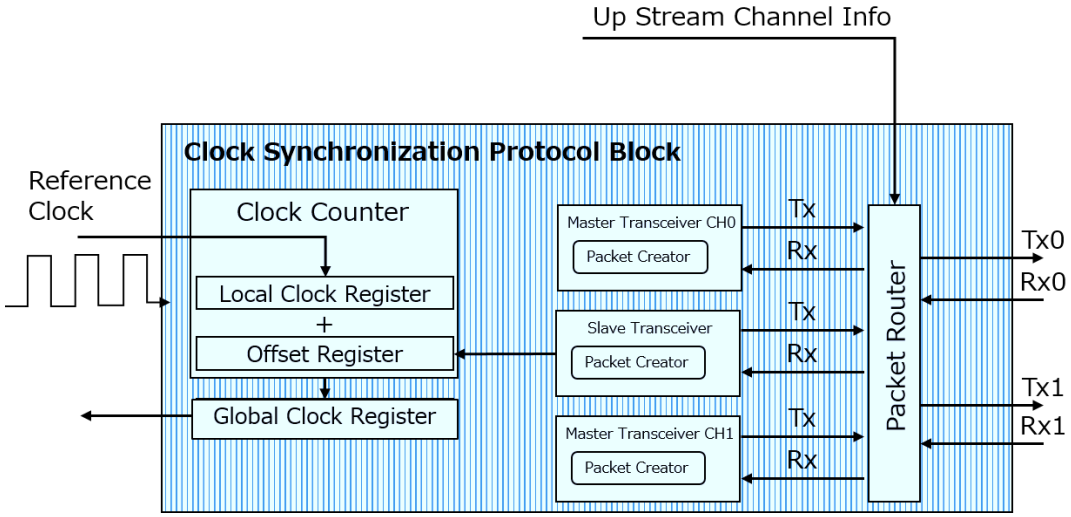
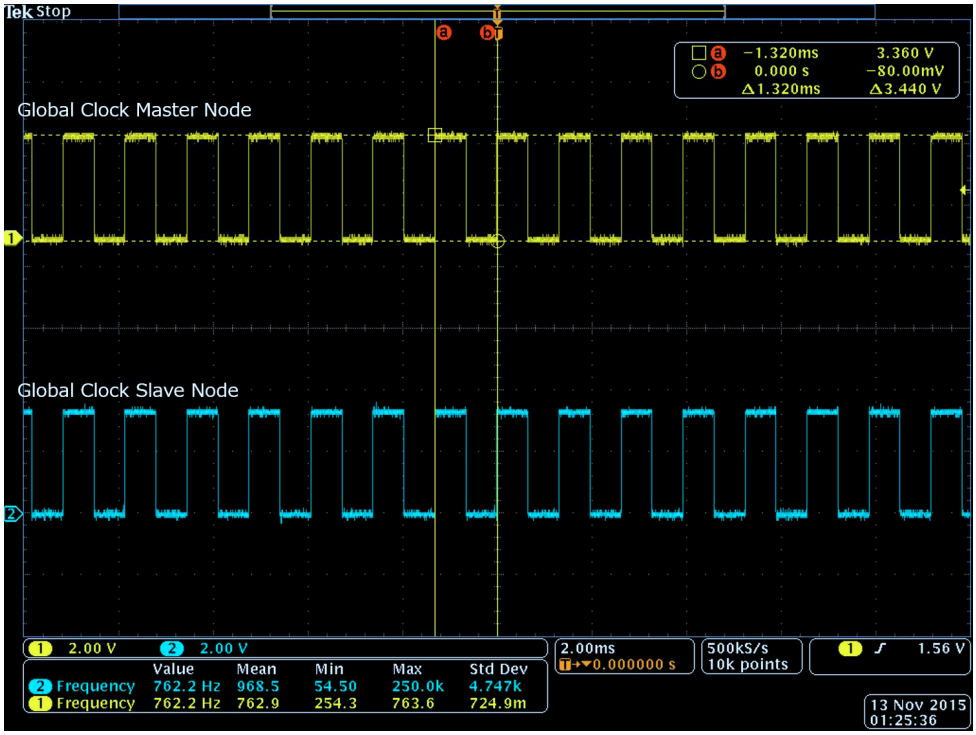


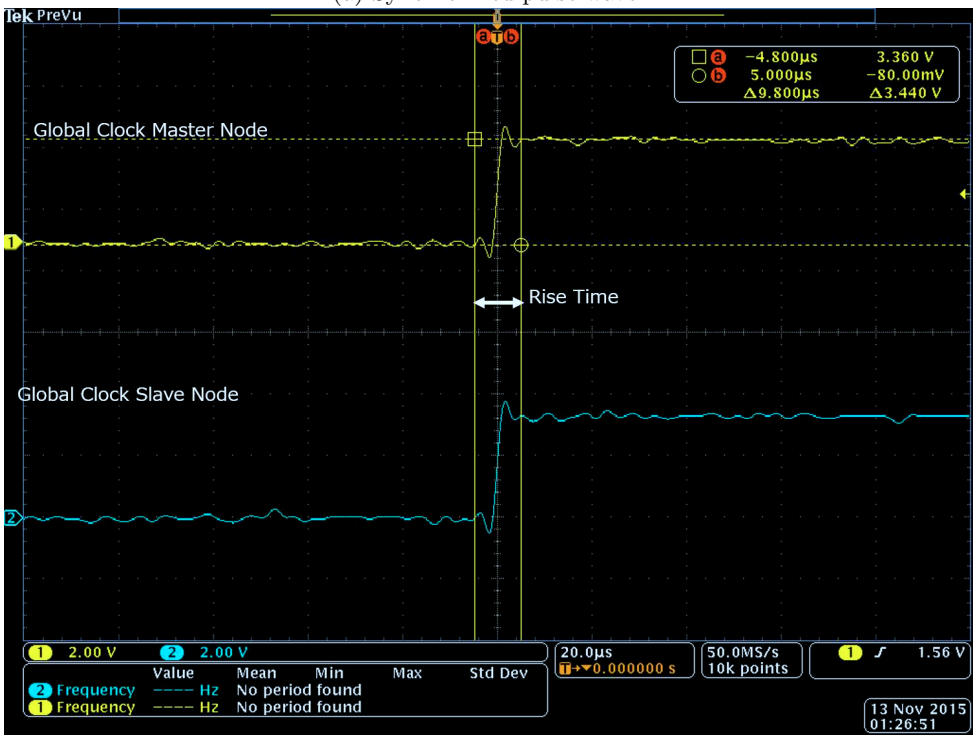
図 4.22: Architecture of Clock Synchronization Protocol Block in RMTP-02D board.

すれことが考えられる。また、GPS 信号を利用することで高精度な標準時刻を得られることも知られており、本研究の段階では未実装となっているがこれらをシステムに組み込むことでマスタークロックを複数のロボットで同期させることが可能となり、ロボットの知覚情報を共有する上で有用であると考えられる。

図 4.23a に今回実装した NTP モジュールによるグローバルクロック同期による、ノード間で同期したパルス波形生成のオシロスコープ波形を示す。グローバルクロックマスターノードの出力波形とスレーブノードの出力波形は同期がとられており、位相が揃っていることが確認される。また、図 4.23b に図 4.23a の立ち上がりエッジ付近の拡大図を示す。信号の立ち上がり時間幅に対して、マスターノードとスレーブノード間の信号のタイミングずれは明らかに小さいことが確認される。従って、分散ノード間でクロック時間単位で I/O やその他処理を同期することが可能であると言える。なお、同様の波形をローカルクロックを用いて生成した場合の出力波形を図 4.24 に示す。二つのノード間で出力波形の位相はずれており、グローバルクロック同期の有効性が確認できる。



(a) Synchronized pulse wave.



(b) Enlarged view.

図 4.23: Pulse wave generated by using synchronized global clock on RMTP-02D board.

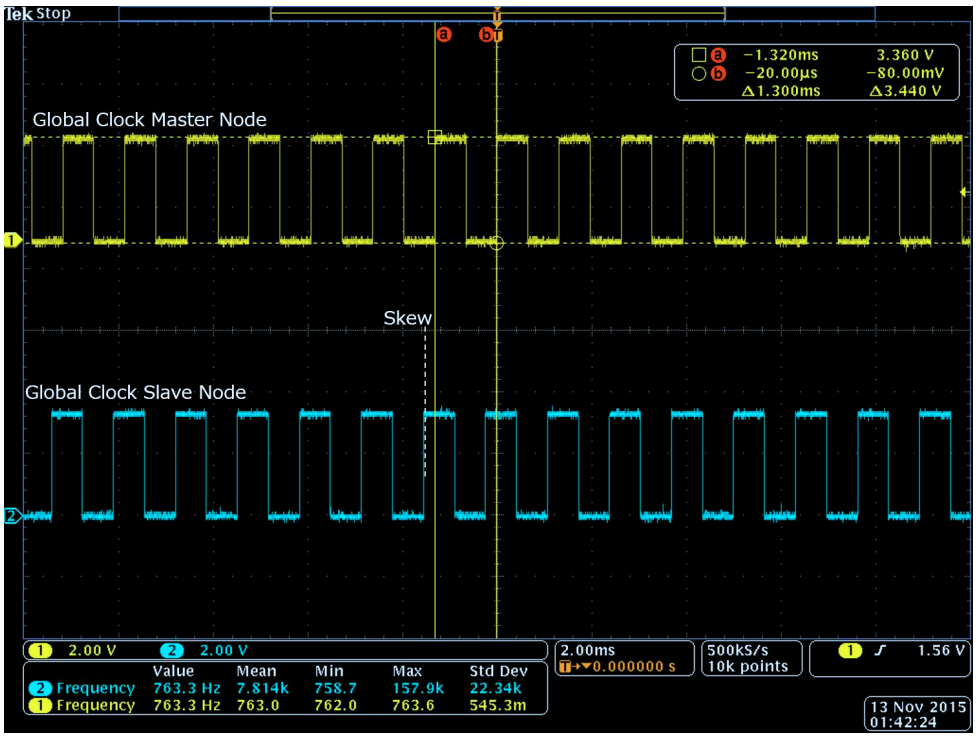


図 4.24: Pulse wave generated by using Local clock on RMTP-02D board.

4.4.5 Light RPC 型プロトコルブロック

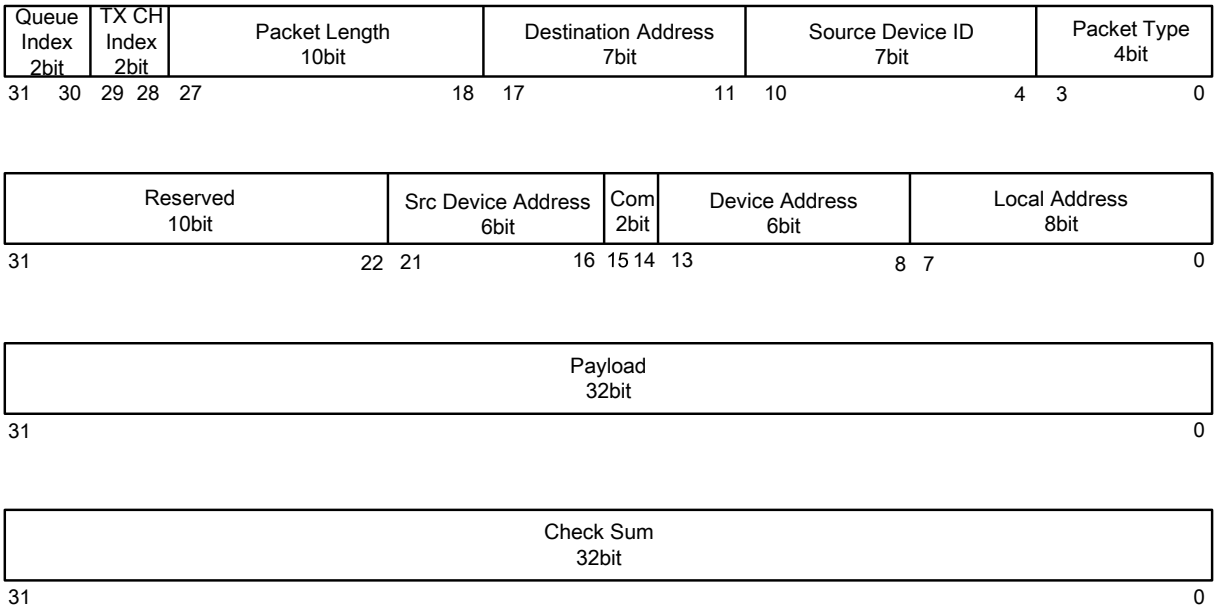


図 4.25: Definition of Remote Procedure Call Block Protocol Packet.

パケットデータフォーマット

4.4.1 節にて述べたように、RPC 型通信モデルを実現するために本研究では Light RPC 型プロトコルをハードウェアロジックとして実装する。Light RPC 型プロトコルにおけるパケットデータフォーマットはコマンド、アドレス、引数データの3 種から構成され、従来 HRP3L-JSK 等で使用されてきた slave link のフォーマット互換となるように設定を行っている。パケットデータフォーマットを図 4.25 に示す。ヘッダ部分を含めて 16 バイト固定長となっており、送受信処理を実時間で行うことが可能である。

Light RPC における2 ノード間通信の流れ

Light RPC では、push、pull、interrupt の3 種のリクエストに基づいたコマンドを即座にコントローラにおける処理に反映させるための機構を有することで、低遅延性を確保することを特徴とする。図 4.26 に Light RPC による制御指令の送信の流れを図示する。Push リクエストによってモータ制御器に指令値を送る例にて説明すると、クライアントからモータ制御器に宛てた Push リクエスト命令並びに制御指令値を書き込んだ Light RPC パケットは、受信側においてプロセッサによる処理を待たずに専用回路にて即座にコマンドとしてモータ制御器の対応レジスタへと書き込まれる。以上の通信の手順によって、非常に低遅延での制御指令値の発行を実現することを目的としているプロトコルである。

プロトコルブロックデータフロー

本プロトコルブロックにおける処理のフロー図を図 4.27 に示す。各ポートからの送受信パケットは全て、フロースイッチにおいてリング型論理トポロジを構成するよう経路制御を行われる。受信チャンネルは何れか1 つに限定されて、レシーバへと入力される。受信パケットはコマンド部、アドレス部、引数部でそれぞれ抽出された後、パケットインタープリタへと渡さる。ここでは自ノード宛のパケットであると解釈された場合にコマンドとアドレスに応じて、push、pull、interrupt の処理命令がディスパッチされる。この命令は FPGA 上に構成されている内部バスを通じて、その他モジュールの制御レジスタに即座に発行される。このような機構によって、ハードウェアロジックによる同期型通信を実現している。pull リクエストであ

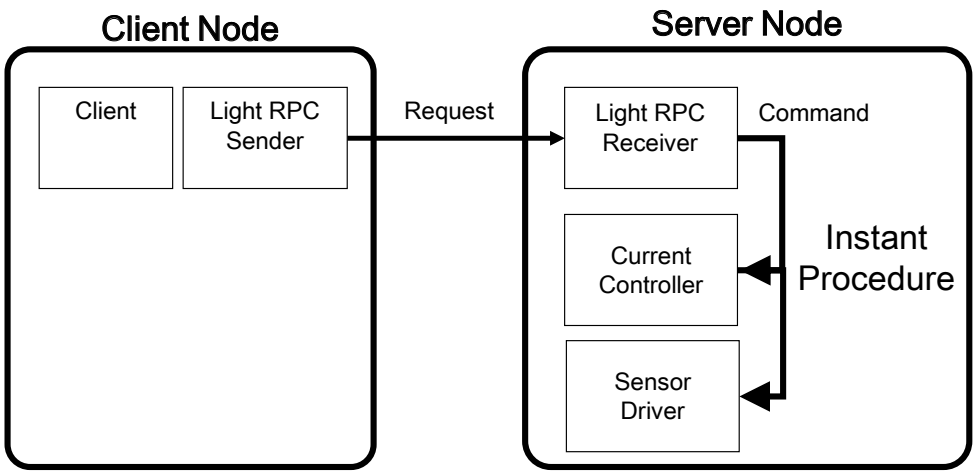


図 4.26: Command transmission using Light RPC protocol. Received command instantly be processed in a related controller.

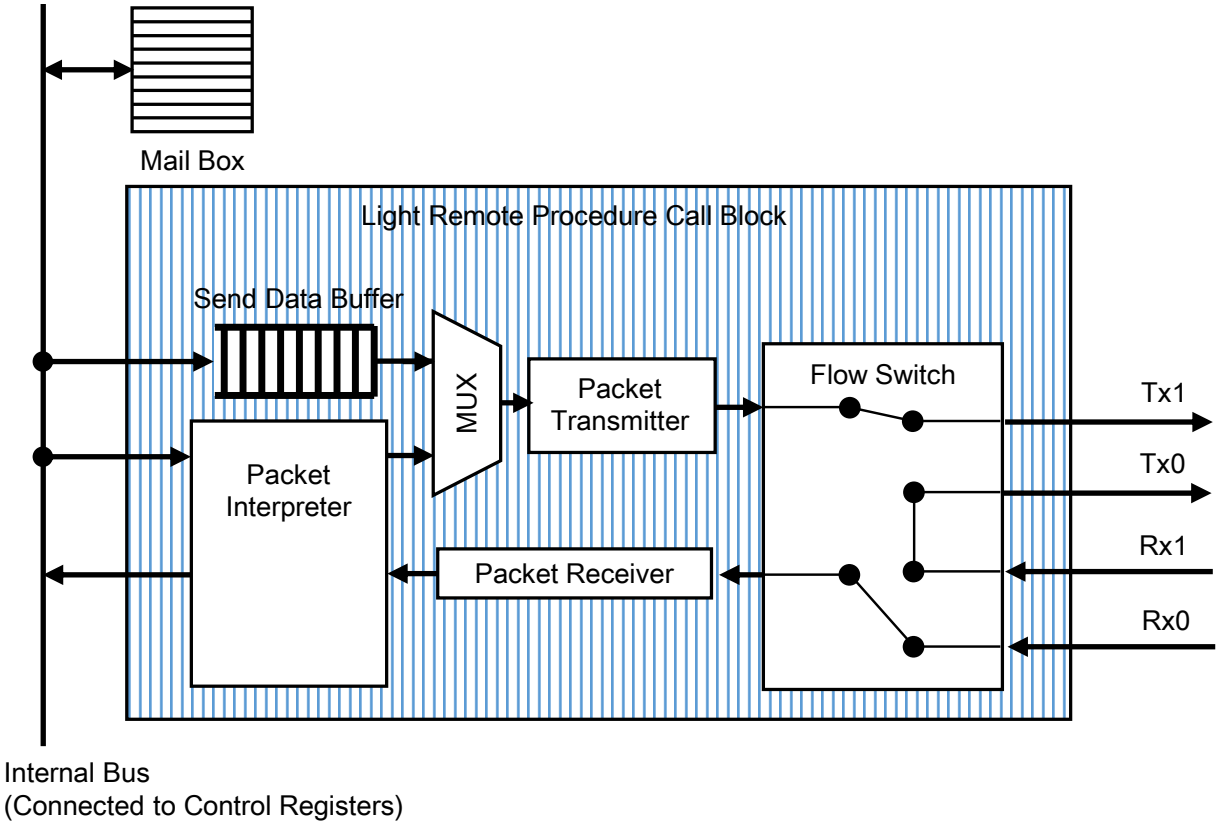


図 4.27: Light RPC block of RMTP-02D.

る場合には、読みだされたレジスタデータを応答パケットに格納し、トランスミッタから送信する。また、自ノード宛でないパケットである場合には命令ディスパッチは発生せずトランスミッタにそのままパケットがパスされ、次のノードへと転送される。

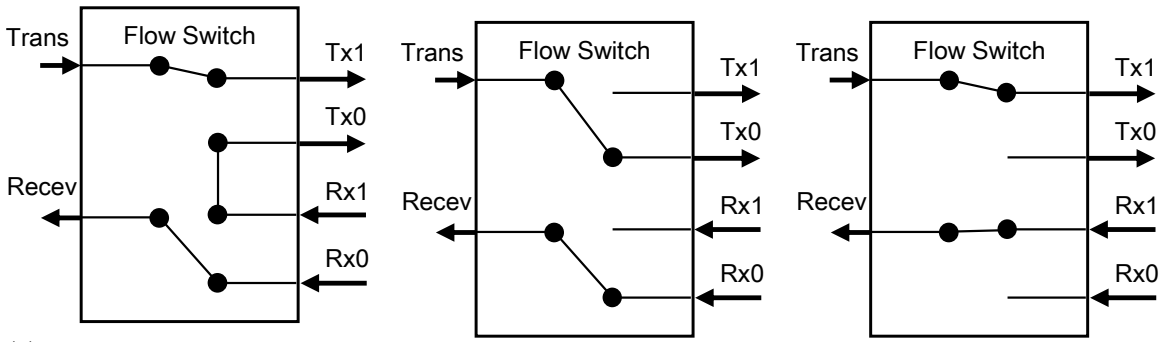
以上の手順によって、非常に低遅延での RPC 型通信処理が実現される。これらの処理は全て FPGA 上のハードウェアロジックとして実装されており、プロセッサからは送信時にパケットデータフォーマットを送信用バッファにプッシュするのみである。送信用バッファにスタックされたパケットは、パケットインタープリタからの送信パケットが無いタイミングでトランスミッタへと引き渡される。この送信処理は、インタープリタのバスリクエスト待ちやレシーバの処理待ち時間（ 4クロック）で完了するため、通信系全体

としての実時間性を損なわない。pull リクエストの返信データの受信方法としては、内部バス上に

1. メールボックス
2. メッセージキュー

を用意しておくことで、プロセッサからの読み出しが可能となる。

フロースイッチ



(a) Both of Ch.0 and Ch.1 is active.

(b) Only Ch.0 is active.

(c) Only Ch.1 is active.

図 4.28: Working model of flow switch in Light RPC block.

リング型論理トポロジを構成する上で重要となるフロースイッチの動作図を、図 4.28 に示す。フロースイッチの動作は2 つ搭載されている通信チャンネルの接続状態によって 3 タイプに場合分けされる。図 4.28a は2 つのポートが共に他ノードに接続されている場合である。この場合は、チャンネル0 から受信したパケットをパケットインタープリタに渡し、送信データをチャンネル1 へと受け渡す。チャンネル1 から受信したパケットは何も処理をしない状態でチャンネル0 から再び送信する。これによって、物理的にライントポロジで接続されたノード間でリング型論理トポロジを構成することになる。また、片方のチャンネルが接続されていない図 4.28b または図 4.28c の場合には、送受信パケットはそのまま、トランスミッタとレシーバを経由する。各チャンネルの接続状態は PHY から信号を得られるため、このスイッチの動作切替は動的に行うことが可能である。そのため、ユーザーは特にネットワーク構成を設定することなく Light RPC 型プロトコルを利用が可能となっている。

この例では2 チャンネルのライントポロジ構成であるが、3 チャンネル以上のデバイスにおいても、接続状態によってパケットインタープリタと接続されるポートの組と、パススルーされる組の場合分けを設定することで同様にリング型論理トポロジを構成可能である。ただし、ノード間の接続チャンネルの組み合わせによっては局所的なループが構成される可能性が生じるため、あらかじめ接続するチャンネルの優先順位を設定しておく等の対策が必要となる。

4.4.6 分散共有メモリプロトコルブロック

異なるノード間でデータ共有するための仕組みとして、分散共有メモリをシステム内に構成することを目指す。分散共有メモリは図 4.29 に示すように、各ノード内に同サイズのメモリを持たせ、あるノードにてメモリに書き込まれた情報が他のメモリの同アドレスからリード可能となるように、ノード間のメモリ情報をミラーリングする仕組みである。単一システム上で利用される共有メモリによるプロセス間通信を、分散型システムでのノード間通信に適用したものと考えられる。コントローラ内のプロセスからは、外部バスに接続されたメモリへのアクセスのみで分散システム全体との間でメッセージの送受信を行うことができる。

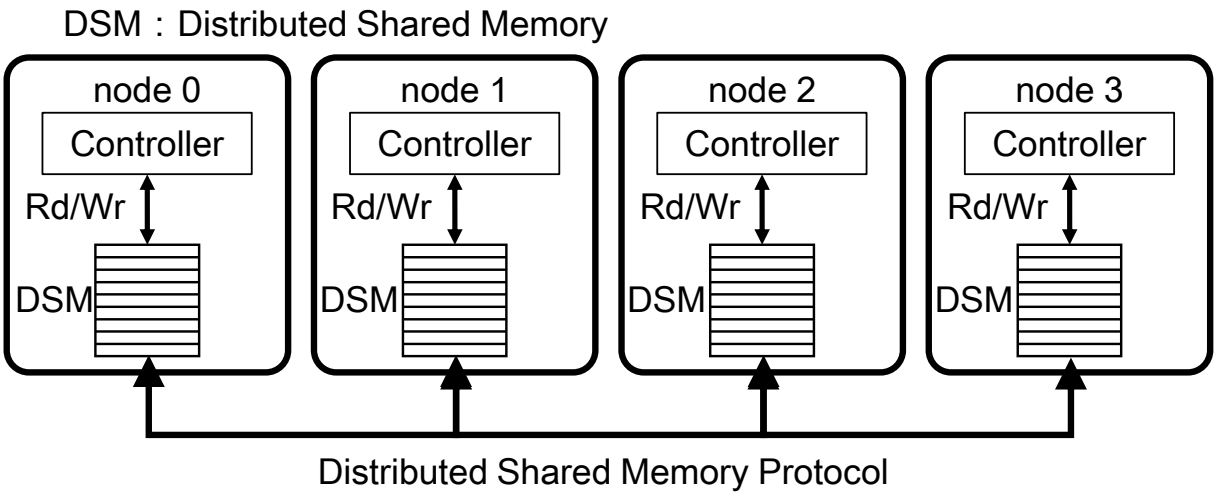


図 4.29: Data sharing with other nodes by distributed shared memory protocol.

分散共有メモリプロトコルの従来研究

Ethernet を介した分散共有メモリの実装については、古くから研究・開発が行われており、特に高性能なクラスタ型コンピュータを構成するためのネットワークとして松本ら [62, 63, 64] による低遅延な分散共有メモリソフトウェアが開発されている。

これらは大規模な計算機での開発例であるが、組み込み向けの省資源環境における分散共有メモリの実装例はあまり見られない。その中で、CUnet[4](Step Technica Co.) は、メールボックス型分散共有メモリプロトコルを実装した IC として提供されている。物理層に RS485 を採用しており、最大データレートは 12Mbps としている。最大接続ノード数は 64 ノードであり、共有メモリサイズが 512 バイト提供される。共有メモリの更新には2 ノード接続で約 102μsec、16 ノード接続で約 501μsec、32 ノード接続で約 1037μsec としている。分散共有メモリによる多ノード間通信を提供するという点で、本研究と共通している。非常に低コストでの実装となるが、ヒューマノイドロボットの体内制御系として使用するにはデータ更新に時間がかかりすぎるという点や、他の実時間通信プロトコルとの併用の可否といった点で本研究との相違が有る。

パケットデータフォーマット

図 4.30 に本研究で実装した分散共有メモリプロトコルのパケット構造を示す。ヘッダ・フッタは前述の通り共通のフレーム構造である。パケットフォーマットの2 ワード目には送信タイミングのグローバルクロックが格納されている。これは、非同期通信では受信データがどのタイミングで処理されるか保証されていないため、受信データが作成された時刻を精細に検証することで制御に異常データが反映されることを防ぐことを可能とするものである。あるいは特定アドレスへの同時書き込みによるデータハザードが起こる問題に対して、最新時刻のデータのみを反映させるというポリシーで保護をかけることが可能となる。このパケットに埋め込まれたグローバルクロックをどのように利用するかは受信側に委ねられる。なお、本研究では後述のトピック通信プロトコルの利用によってメモリ保護や排他制御実装に頼らないノード間の共有メモリベース通信を実現する。

3 ワード目はパケットに含まれるデータの格納アドレスとバイトイネーブル信号及びバースト長を示すデータとなっている。本プロトコルではバースト転送による可変長パケット構造を許可している。バースト転送時のデータ長を示すのがバースト長情報であり、4 ワード目以降の可変長データ部のサイズを示している。また、バイトイネーブル信号をセットで送信することで、32bit 長データ以外のフォーマットのメモリアライメントデータについて分散共有メモリ構成を利用することも可能となっている。

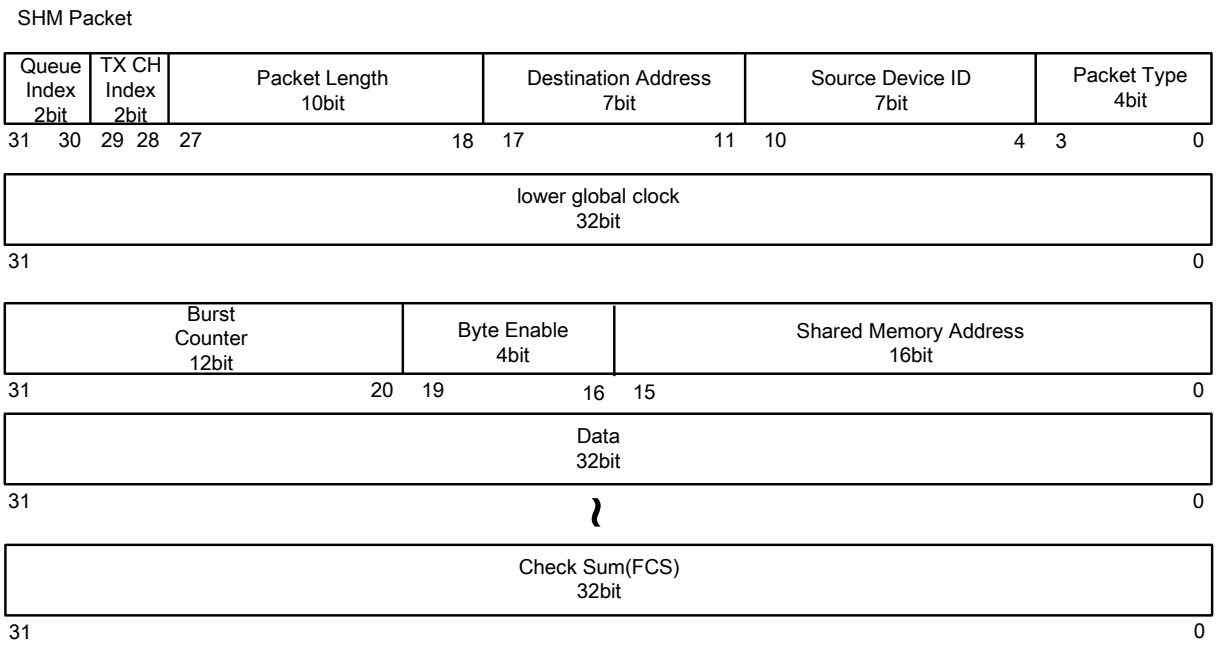


図 4.30: Definition of Distributed Shared Memory Block Protocol Packet.

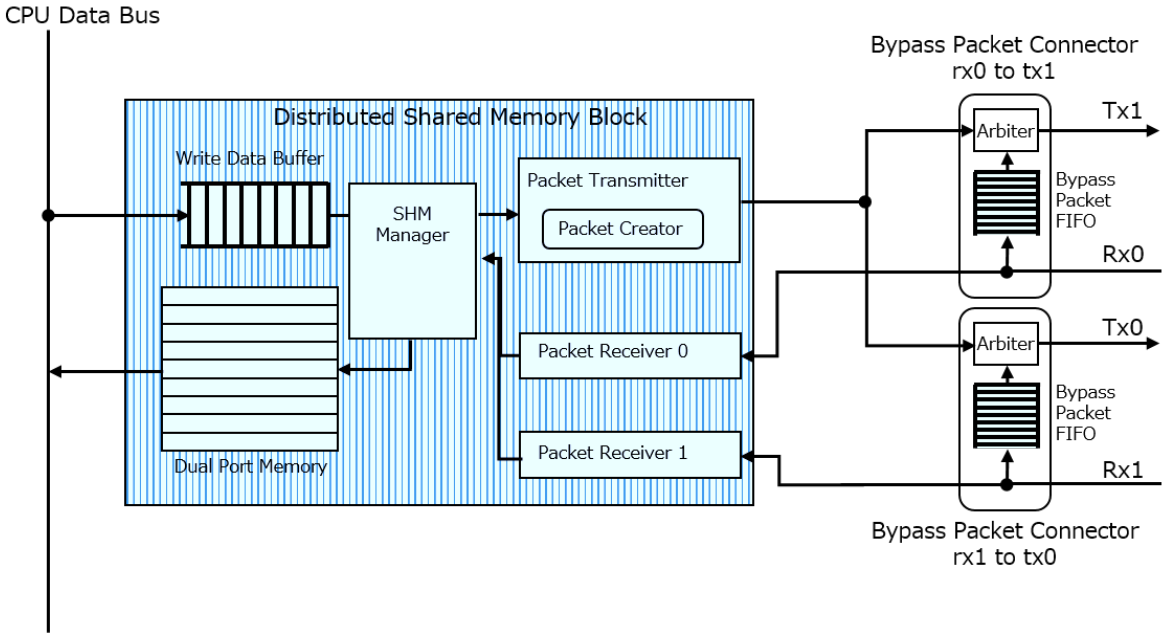


図 4.31: Distributed shared memory block of RMTP-02D.

プロトコルブロックデータフロー

本プロトコルブロックにおける処理のフロー図を図 4.31 に示す。FPGA 上内で提供されているオンチップのデュアルポートメモリを共有メモリとして、CPU バスに接続している。

次に分散共有メモリプロトコル部の実装であるが、メモリ書き込み側の処理は図 4.31 に示すように、一度、書き込みデータ及びアドレスをバッファにストアする。これは、CPU からのメモリ書き込み要求と他ノードから受信したメモリ書き込みデータによる書き込み要求が衝突した場合に、受信データからの書き込み要求を優先して処理するためである。この書き込み要求の処理を管理するのが図 4.31 中の SHM Manager(SHared Memory Manager) である。SHM Manager は受信側からのメモリ書き込み要求が無い場

[4]www.steptechnica.com/jp/cunet/index.html

合には、バッファにストアされていた CPU バスからの書き込みデータの処理を行う。同時に、他ノードへの分散共有メモリプロトコルパケットを生成し、トランスミッタから転送を行う。パケットは、ブロードキャストパケットとしてヘッダを構成することでネットワーク中の全ノードに分配している。図 4.31 中の Bypass Packet Connector ブロックによって、受信したブロードキャストデータストリームを自ノードのパケットレシーバに送ると共に、他チャンネルのトランスミッタにそのままストリームさせることでパケットのルーティングを行っている。転送先のチャンネルがビジー状態の場合にはブロードキャストパケットストリームを一旦停止させるために、Bypass Packet Connector ブロックは内部にバッファを持っている。

また、本研究で実装した分散共有メモリプロトコルでは、バースト転送をサポートすることで、転送速度の向上を図っている。バースト転送は、あらかじめプロトコル制御レジスタ (CSR) に設定しておいたバースト転送長に従って、バッファにストアされている複数の書き込みデータを一度に転送する方式である。これにより、データ転送にかかるオーバヘッドによる遅延を削減し、通信リンクの帯域を有効活用することが可能となる。

共有メモリサイズ

共有メモリは表 4.6 で示す構成としており、全体で最大 64kByte のメモリ空間を共有メモリとして利用することが可能となる。データ幅は 32bit としているが、CPU バスからはバイトイネーブル信号によって char,short,long のどのデータ長からでもメモリライト可能としている。また、メモリサイズの構成は共有メモリプロトコルにより転送したいデータサイズに応じて再構成することは可能である。実際、RMTP-02D 基板で利用可能な最大メモリサイズは 64kByte で 1kWORD の空間が利用可能となっており、実ロボットで使用する第 5 章及び第 6 章の構成では最大の 64kByte サイズの構成を利用した。

表 4.6: Specification of Distributed Shared Memory on RMTP-02D.

Data Width	32bit
WORD Address Width (max) Total WORD Number (max)	9bit (14bit) 512WORD (16kWORD)
RAM port	dual port (read only port and write only port)
RAM read clock frequency (CPU bus side)	62.5MHz
RAM clock frequency (Link controller side)	100MHz

分散共有メモリを大容量化した場合、メモリ更新頻度によってメモリ帯域速度がボトルネックとなってくる。実際、低コストな SRAM で 50MByte/sec、高性能な製品で 200-400MByte/sec となっており、FPGA 内蔵のメモリブロックの読み書き速度もこれにおおよそ従うが、1 サイクル 5000Hz、200μsec での 64kByte の更新に必要なメモリ帯域速度はおよそ 312.5MHz となり、高性能な SRAM でようやく条件を満たせるサイズとなっている (表 4.7)。SRAM の並列配置でさらに高速化することは可能ではあるが、モータードライバーデバイスのパッケージサイズ巨大化や高コスト化を招いてしまう。また、SDRAM 等を利用することも考えられるが、ピン数や配線の複雑化によって基板開発の難度が上がってしまうため、今後の検討事項となっている。このようにメモリ帯域速度は分散共有メモリの構成において、現時点での最大の課題となっている。従って、この段階では主要な共有データは 2kByte のメモリ空間に限定して運用するのが適切であると考える。2kByte のデータサイズであれば表 4.7 に示す通り、1 サイクル 200μsec の高速周期で全更新した場合でも 10MByte/sec の帯域があれば十分であるため、低コスト SRAM で足りる。

2kByte のメモリ空間に 32 個分のアクチュエータ・センサ情報を展開した場合、1 アクチュエータ当たり 16 ワード =64Byte のデータを共有メモリ上に持てる計算となる。アクチュエータの代表的なデータとしては 1. エンコーダ、2. 速度、3. 目標値、4. 目標誤差、5. 温度、6. 電流、7. トルク、8. 制御モード、9. 制御ゲインをすべて long 型の 4Byte 長で表現したとしても、まだメモリ空間には余裕がある構成となっている。最適化を行い、温度やゲイン等の 4Byte 長のデータレンジが必要ないものについては、char や short 型で

表 4.7: Comparison of SRAM band width and required data rate.

Device	Cycle speed	I/O Speed [MB/sec]
Low-cost SRAM	12-20ns	48-80
High performance SRAM	2-4ns	240-477
DSMB 2kByte	200μsec	10
DSMB 64kByte	200μsec	313

管理することによってメモリ空間の余裕はさらに広げられるため、2kByte のメモリサイズは十分であると考えられる。

4.4.7 コネクション型プロトコルブロック

パケットデータフォーマット

コネクション型プロトコルにおいては、パケットデータフォーマットは 4.4.2 節にて示したヘッダ・フッタを除いて、その内部フォーマットは不定としている。これは、ネットワーク層以上のレイヤ処理をハードウェアロジックではなく、ソフトウェア処理として行うためであり、詳細なフォーマットはプロセッサ上で実行されるプロトコルスタックに依存する。

コネクション型プロトコル

ここまで述べてきたプロトコルでは対応の難しい通信処理についてもシステムとして対応可能とするために、コネクション型プロトコルを実装する。ここまで述べたプロトコルでは実時間性を重視するために、パケットデータフォーマットについては、ハードウェアロジックとして実装するのに都合の良い形式で構成を行った。そのため、アプリケーションに応じた柔軟なデータ構造とすることは不可能となっている。

しかしながら、実時間性についての要求は無いが必要となるアプリケーションも少なからず存在する。例としては、分散ノードのリモートアップデート機能がある。ロボット体内の各ノードのファームウェアを更新するためには、通常はデバッグ用ポートから直接更新ソフトをダウンロードする必要がある。しかし、体内通信系経由でファームウェアの更新を行うことが可能であれば、メンテナンス用のケーシングを開ける工数をかける必要無く全ての更新を完了でき、保守性の向上に効果的である。ファームウェアデータファイルは FPGA で 4MByte 以上のサイズがあるが、このファイルが正常に分散ノードへと送信され、さらに EEPROM へと書き込まれる必要がある。コネクション型プロトコルでは、この一連の処理をセッションとして管理し、書き込みエラー等が生じた場合にはデータ再送等の手続きによってファームウェア更新処理の信頼性を向上することを目指すことに利用できる。

このようなセッション管理まで含む場合には、ハードウェアロジックとして処理することは実装面積の誇大化を招き、効率的ではない。従って、一般的な通信スキームである各ノードの MPU 上で実行されているコネクション型プロトコル用プログラムから、各アプリケーション用プログラムを都度呼び出して処理することが、コネクション型プロトコルを実現する構成となる。

プロトコルブロックデータフロー

図 4.32 にコネクション型プロトコルのフロー図を示す。本研究で実装したコネクション型プロトコルブロックでは、Light RPC 型プロトコルブロックと同様なフロースイッチを使用することで、ネットワーク層の処理を省略する構成としている。パケットのレシーバ、トランスミッタはそれぞれリングバッファと接続されており、ヘッダ・フッタを除いたパケットデータは全てリングバッファを経由して、プロセッサと受け渡しが行われる。ネットワーク層以上の通信処理に関してはプロセッサ上で実行されているプログラムに依存する。なお、リングバッファコントローラはヘッダ情報を元に、パケット長情報を抽出しており、プロセッサはリングバッファコントローラからパケット長情報を読み出すことでパケット区切りを検出する

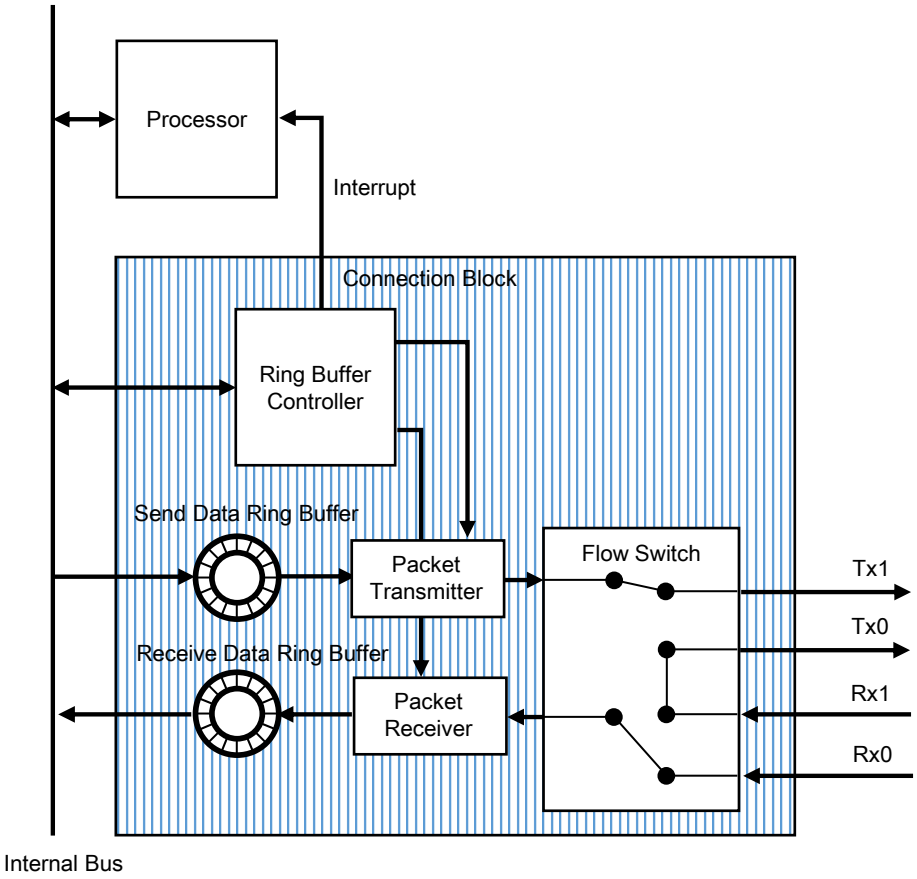


図 4.32: Connection block of RMTP-02D.

ことが可能となっている。また、リングバッファコントローラからは割込み信号がプロセッサへと接続されており、受信割込みによる応答処理も可能となっている。

リングバッファ構成

コネクション型プロトコルの送信･受信に用いられるリングバッファは 32bit 幅で構成されており、デバイスのメモリ資源量に応じて、受信リングバッファを 4kByte、送信リングバッファを 1-4kByte に設定している。組込みプロセッサで用いられる NIC に備えられているバッファサイズも概ねこれらのサイズであり、リングバッファとして構成してもバッファオーバーフローを起こしにくいと考えられる。バッファのオーバーフローをトリガーとした割込みを元に、再送要求を行ったり、残りバッファ長に応じて輻輳制御を行うことで、ファイル転送のような大容量通信においても確実に処理を行うことが可能となる。

4.5 低遅延体内分散通信系を利用したアプリケーション層設計

4.5.1 Light RPC 型プロトコルによるループバック遅延測定

ループバック遅延の計測

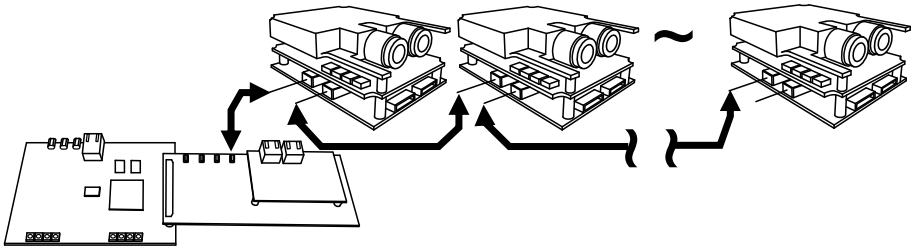


図 4.33: Settings of network loopback latency measurement.

本節では、本章で構成する低遅延通信系の実機を用いて、実時間性能を評価する上で重要な指標となる通信遅延時間について計測を行う。計測方法としては、同期通信を行う Light RPC 型プロトコルを用いて、ループバック通信を行うことで、通信に要する時間を計測する。Light RPC 型プロトコルではパケットはリング型論理トポロジでネットワークを構成されるため、ループバックに要するノード間ホップ数は完全にノード数に比例する。従って、ループバック遅延時間については接続ノード数に比例することが期待される。計測では図 4.33 に示す様にに RMTP-02D 基板を 2-10 ノードライントポロジで接続し、それぞれの場合のループバック遅延時間を計測した。計測のため、5.4 節にて導入する SoCKit 制御デバイスを Light RPC パケットの送信マスター及び計測器として利用した。この SoCKit デバイスでは、Linux OS 上で実行される計測用プログラムが使用されており、また可能な限り正確な時間を計測するためにカーネルドライバ内で FPGA 上のローカルクロックを基準として計測を行った。計測は 2 ノード接続から始めて、各ノードへの送信したループバックパケットが帰還するまでの時間をそれぞれのノードについて 5000 回計測した。なお、FPGA とプロセッサ間のインターコネクト上のデータ転送遅延は FPGA 側で 4Cycle、40[nsec] であった。結果の数値はこの 40nsec 分のオフセットを除去して示す。

計測結果

図 4.34 に実験結果を示す。各接続ノード数の実験について、平均ループバック遅延時間及びベスト・ワースト ループバック遅延時間をグラフに示している。2 ノード接続時のループバック遅延は平均値で 2.784[μsec] であった。ここから1 ノードずつ接続数を増やした場合、1 ノード増加につき約 1.4μsec ずつ遅延値が増加した。図中のエラーバーは各試行におけるワーストケースとベストケースでのループバック遅延時間を示している。2 ノードループバックではベストケース 2.78[μsec]、ワーストケース 2.84[μsec] となりベストケースと平均値が非常に近い値を示し、ワーストケースでも +60[nsec] と非常に高い実時間性を示している。ノード数が増えるにつれて、ベストケースとワーストケースの差が広がり、10 ノードでは 500[nsec] 程度に広がる。また、8 ノード以上でワーストケースが平均値に対して 2000[nsec] と明確な遅れが生じるケースが確認できた。これは、計測を Linux OS 上のデバイスドライバ内で処理したためシステムの割込み処理のタイミングでワーストケースが生じたと考えられる。

ループバック遅延計測結果のまとめ

ループバック遅延の計測結果より、本研究で提案するロボット体内低遅延通信系上での通信遅延は、1 ホップ当たり約 700nsec=70 サイクル@100MHz 程度であると計算できる。これは 2500Mbps のデータレートでは 16 バイトのデータ送信によって生じる遅延は約 5 サイクルであり、高速トランシーバの通過に 4 サイクル、さらにバッファ入出力、トランスミッタ・レシーバ処理、パケットインタープリタ処理がそれぞれ 8-20 サイクル程度要するためである。入出力処理等に最適化の余地があると考えられるが、現時点での実

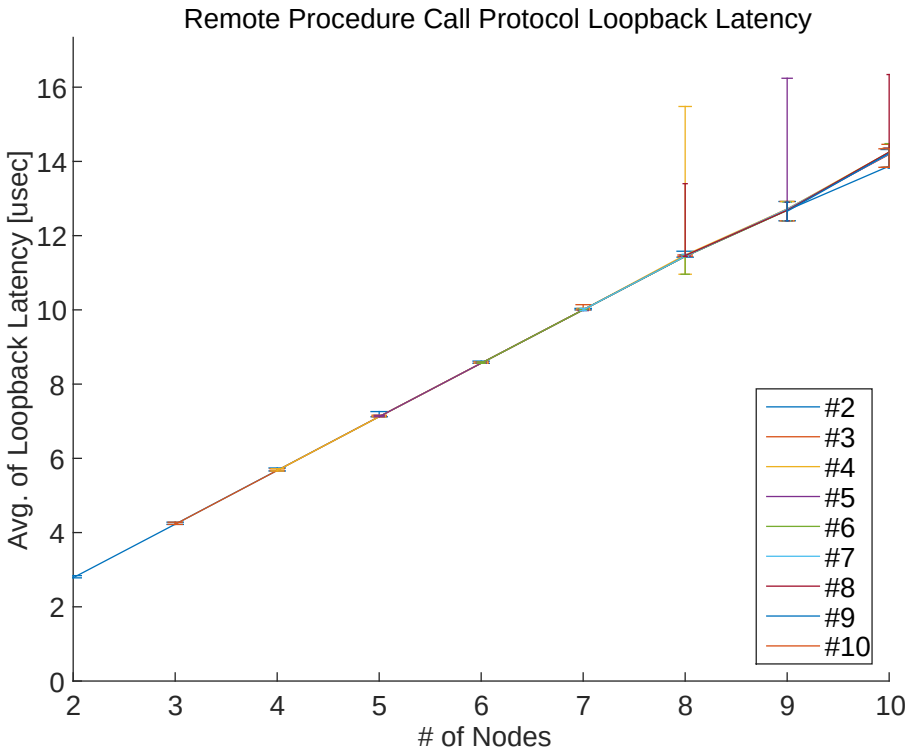


図 4.34: Loop back latency of pull request using Light RPC protocol.

装においてもロボットの目標制御周期 200μsec と比較した場合に十分に短時間であり、目的を達成すると考えられる。

ループバック遅延時間については、前述の通り接続ノード数に比例しており、リング型論理トポロジでパケットが流れていることが確認される。

8 ノード以上の場合においてワーストケース時間が増大する結果が得られているが、この場合においても平均値とベストケースの値は比較的近い値を保っているため、システム側の割込み処理によるデータ取得遅延の生じる頻度は低いと言える。また、FPGA レイヤーまでの到達ならワーストケースはさらに改善されると考えられるため、クリティカルなハードリアルタイム処理については、RMTP やハードロジック上での処理を実装することで仕様を満たすことが可能である。

また、32 ノード接続構成の場合、同期通信による push リクエストは最大 22.4μsec かかると計算される。制御周期 5kHz、200μsec とした場合のおおよそ 10 分の 1 であり許容範囲内と言える。また、通信によって生じるジッタ値はワーストケースを考慮しても 3μsec 程度と考えられるため、プロセッサでの処理に起因するジッタを足し合わせても 2.6.4 節にて述べた目標ジッタ値 20μsec を十分に達成可能な精度となっている。

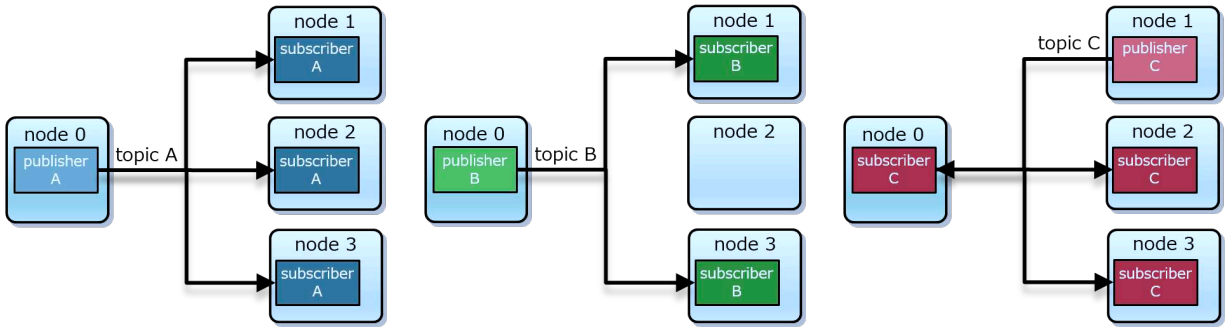
4.5.2 パブリッシュ・サブスクライブモデル型通信

前節で述べた RMTP-02D 基板に実装した分散共有メモリプロトコルを利用して、分散型システム内の制御器間でデータストリーム型通信を実現するためのアプリケーション構成として、パブリッシュ・サブスクライブモデル型メッセージ通信を導入する。

まず、パブリッシュ・サブスクライブモデル型メッセージ通信とは ROS でも使用されている通信モデルの一つであり、分散ノードが持つパブリッシャと呼ばれるデータ送信元と、サブスクライバと呼ばれるデータ受信先の間でトピックと呼ばれるメッセージを転送するモデルとなっている。パブリッシャ、サブスクライバはトピック種別ごとに用意される。図 4.35a に示すように、あるトピックについてパブリッシャ・サブスクライバは複数用意することができ、パブリッシャは任意タイミングでトピックを発行し、他全てのサブスクライバはそのトピックを受け取ることでメッセージの転送を行う。あるトピックについてサブスクライバを持たない場合には、そのトピックは受け取らない図 4.35b 。また、別種類のトピックについてはノー

ド上に持つパブリッシャ・サブスクライバの構成を変更可能なため、図 4.35c のように用途に応じたデータフローを自由に構成可能である。このように、ある特定のトピックに基づいて 1 対多や多対多の一方向のデータ転送を複数のノードで行っていることが特徴となる。分散型システム内のノード間で、他のノードの状態によらず非同期にメッセージのやり取りを実行できるため、多ノードシステム間の通信手法として効果的であると考えられる。

以上のパブリッシュ・サブスクライブモデル型通信を分散共有メモリにて実現した方法を図 4.36 に示す。分散共有メモリの各アドレスがバッファサイズ1 のトピックに対応し、パブリッシャ及びサブスクライバはそのトピックを介して、データのやり取りを行う。データストリーム型メッセージング形態を利用する場合には各トピックに書き込むことが可能なノードを1 つに限定することで、データの混信を回避する。



(a) Publish for all nodes. (b) Subscriber-less node. (c) Another data flow configuration.

図 4.35: Publish and Subscribe communication model with topic.

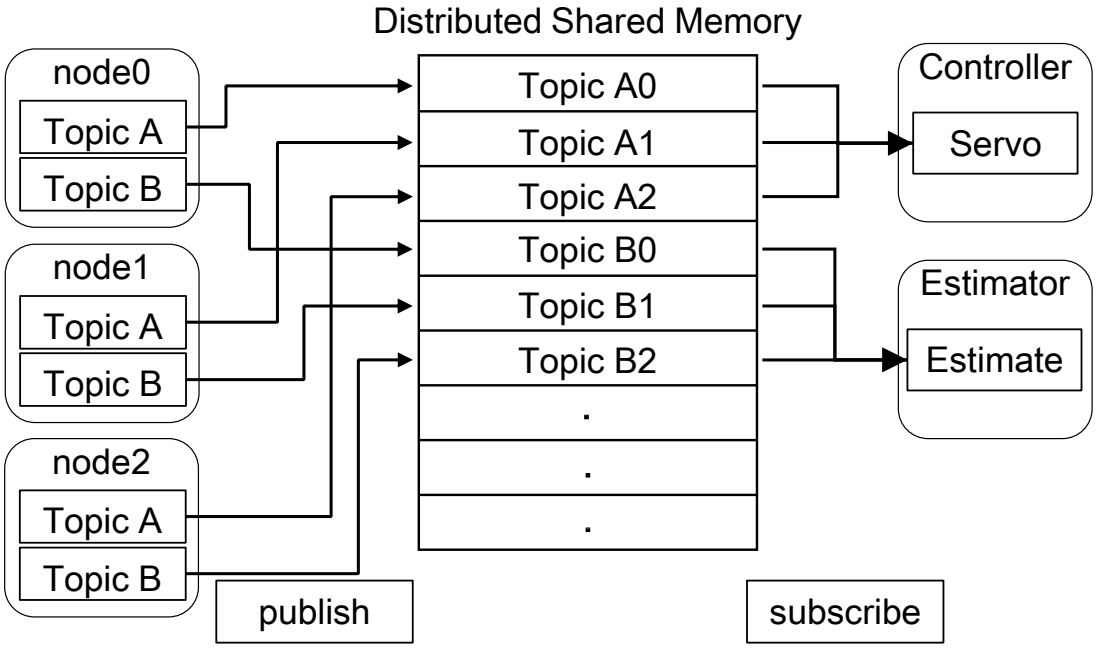


図 4.36: Publish and subscribe communication via distributed shared memory on the system.

4.5.3 分散型共有メモリプロトコルによるトピック通信の検証

前節で述べた分散型共有メモリプロトコルによるトピック通信について、分散型共有メモリプロトコルはブロードキャストパケットとしてネットワークストリームされるため、多ノード接続・多トピック転送を行った場合におけるデータ転送の遅延が懸念される。高速な応答性を目指す本システムにおいて、1 制御周期における全トピックのデータ転送時間が長い場合には、モータ制御タスクの最悪実行時間をオーバして

しまうため、事前に検証を行うことが必要と考えられる。また、その他のプロトコルも転送されるため通信リンクの過剰な利用は問題となる。以下、論理回路シミュレーションによって、RMTP-02D 基板上に実装したトピック通信の検証を行い、多ノード接続時の全トピック転送完了までの最悪実行時間、及び通信リンクの使用率について確認を行う。

シミュレーション条件

ノード数の増減による通信転送時間の検証として、4 ノード、8 ノード、16 ノード、24 ノード、32 ノード接続のケースについて比較を行う。転送トピック数としては、1 ノードあたり 16 ワード =64Byte とする。これは、前節で述べた通り 1 アクチュエータの制御情報をトピックとしてパブリッシュするのに十分なサイズである。通信リンクの駆動クロック周波数と CPU バスの動作クロック周波数は実機と合わせ、それぞれ 100.0MHz、62.5MHz で行っている。

さて、1 制御周期の中で各トピックをどのタイミングで転送するかは、様々なパターンが考えられる。グローバルクロックに基づいて、全ノードで一斉に転送を開始するパターン、パケットの転送に時間がかかるネットワーク上の端に位置するノードから優先して開始するパターンなどである。分散型共有メモリプロトコルでは、優先度付き並列キューによる低遅延 MAC の構成によって多通信が一斉に開始されても、低遅延で転送が開始されるよう設計がなされており、転送パターンによらず、低ジッタな全転送時間が期待される。また、バースト転送モードの実装により、多トピック通信においても通信リンクの帯域使用量は非常に小さく保つことができることも期待される。

ここでは、上記の全ノード一斉送信開始のパターンと端ノード優先パターンの2 つについて検証を行い、転送パターンによらず低ジッタな転送時間となっているか確認を行う。また、バースト転送モードのバースト長についてもバースト無し、4 バースト長、8 バースト長、16 バースト長について比較を行う。また、その際の通信リンク使用率についても調べ、他プロトコルのデータ転送に割ける通信リンクの余力は十分であるか確認を行う。

表 4.8 に、上述した検証条件についてまとめる。1 ノードあたり 16WORD というサイズは、実機において使用されたデータ種別数を元に設定をしている。このデータ種別の参照を表 4.9 に示す。

表 4.8: Experiment condition of Topic communication on Distributed Shared Memory Protocol.

Clock frequency of link logic	100.0MHz
Clock frequency of CPU bus	62.5MHz
Network topology	line topology
# of topics per 1 node	16topic(16WORD = 64Byte)
Total Node numbers	4nodes , 8nodes, 16nodes, 24nodes, 32nodes
Transaction pattern	Start at the same time, Start from the edge nodes
Burst length	no burst, 4burst, 8burst, 16burst

シミュレーション結果

シミュレーション結果を図 4.37 及び図 4.38 に示す。2 つのグラフは同じ結果について横軸のデータを変えたものである。両グラフとも、2 つの転送開始タイミングパターンについてそれぞれ表示している。図 4.37 はバースト転送長に対する全体転送時間のグラフとなっており、バースト転送無しではバースト転送有りと比較して倍ほどの転送時間を要していることが確認される。また、バースト転送有りの場合、バースト転送長の違いによる転送時間への優位な差は見受けられなかった。また、図 4.38 はノード数に対する全体転送時間を示したグラフである。転送時間はノード数に比例して増加する傾向が確認できる。最後に、送信開始タイミングのパターンによる全体転送時間への影響について、やや全ノード一斉送信開始パターンの方が転送時間が短い傾向にあるが、最大で $4\mu sec$ 程度の差であり全体としてパターン間の差は小さい。

表 4.9: Communication data types actually used on a real robot system.

1. Encoder	2. Target joint angle
3. Output	4. Max output
5. Absolute encoder	6. Joint velocity
7. Estimated torque	8. Target torque
9. Frame counter	10. Stiffness param.
11. Motor temp.	12. PD gain
13. Control mode	14. Iq,Id
15. Reserved	16. Reserved

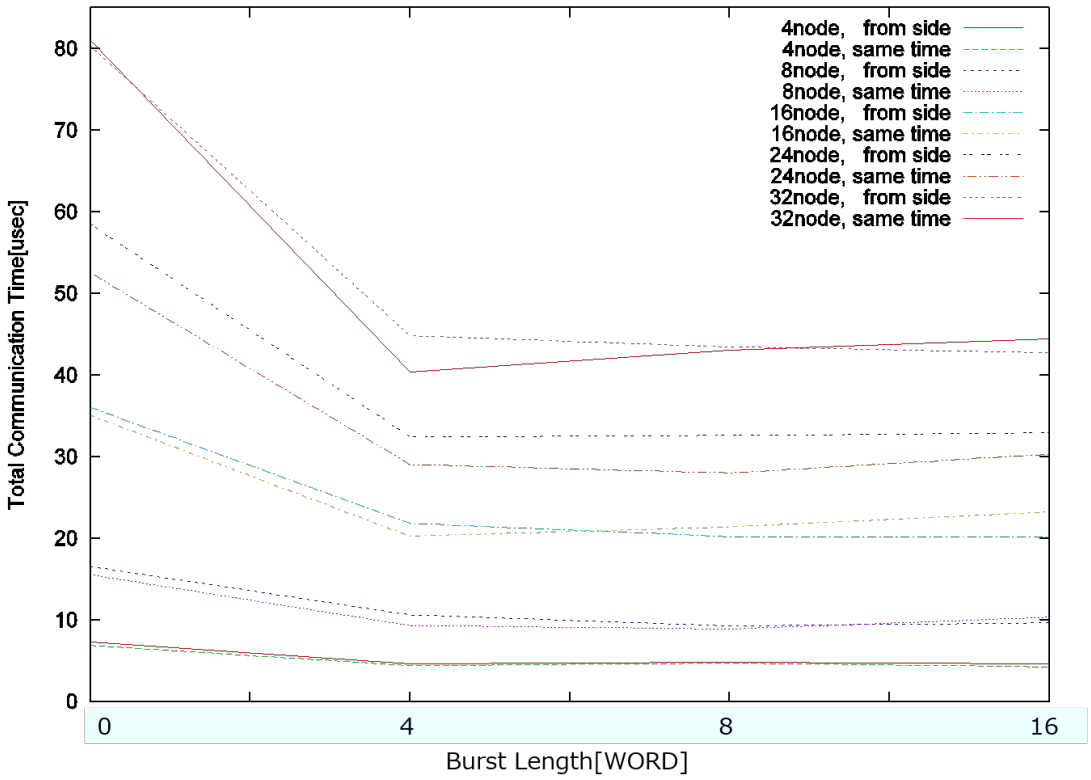


図 4.37: Comparison between communication time and burst length.

次に、通信リンクの使用率の結果を示す。最もデータ転送量の多い 32 ノード接続の条件について、各転送開始パターン、バースト転送長ごとの通信リンク使用率について掲載する。各グラフは横軸に通信開始後の経過時間、縦軸にノード ID、使用率をカラーバーにて表現しており、上りリンク、下りリンクで分割している。

まず、全ノード一斉送信開始パターン、バースト転送無しの通信リンク使用率を図 4.39a と図 4.39b に示す。転送開始直後はリンク全体に渡って使用率が高く、転送時間の内 20 %は使用率 50 %付近を維持している。また、バースト転送有りの場合の通信リンク使用率を図 4.39c から図 4.39h に示す。バースト転送無しと比較して、転送時間の全体に渡って均質な通信リンク使用率になっており、特にバースト長が長くなるほどリンク使用率は安定して 30 %から 40 %を示している。同様に、端ノード優先パターンについても図 4.40a から図 4.40h に示す。使用率の傾向は全ノード一斉送信開始パターンと大きな違いは確認できない。

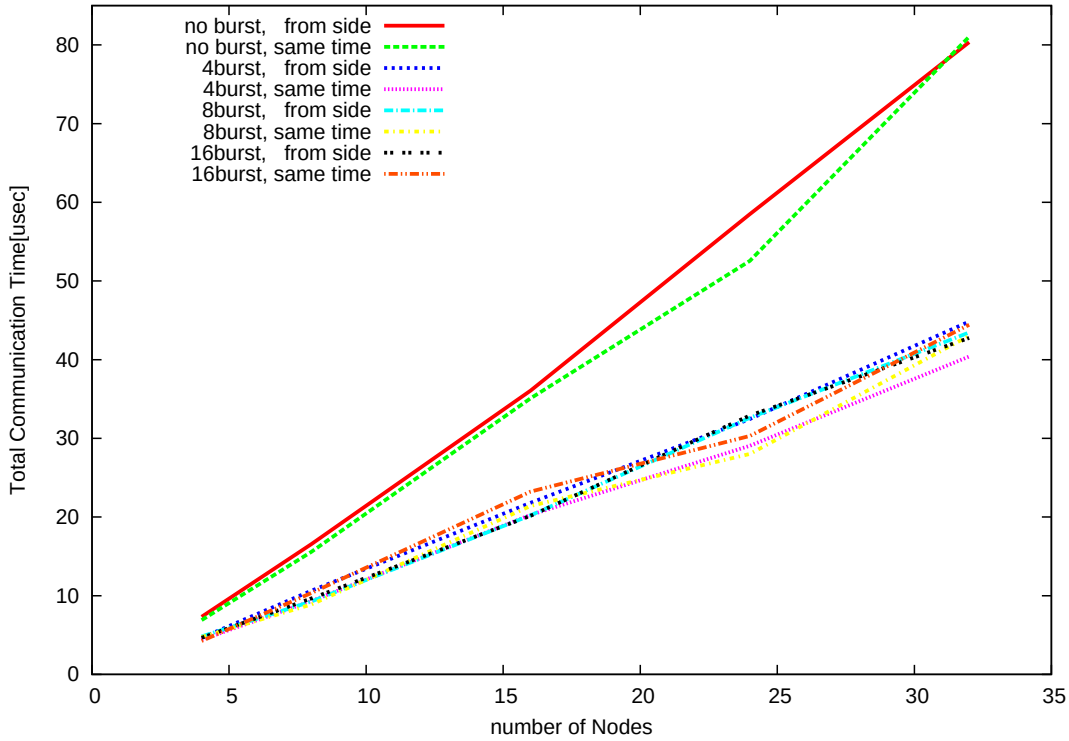


図 4.38: Comparison between communication time and number of nodes.

シミュレーション結果の考察

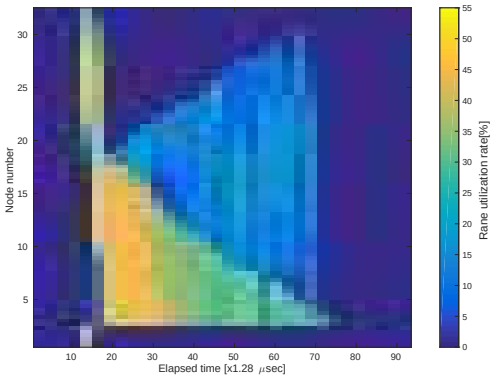
分散型共有メモリプロトコルによるトピック通信は、ノード数に比例した総データ通信量になるが、全体転送時間はこのデータ通信量に合わせて比例する結果が得られた。このことから、データ通信量の増大による通信リンクの遅延は発生しておらず、実時間性の高いデータ通信になっていると言える。また、バースト転送モードの有効化によって転送時間を大幅に削減可能であることが示された。これにより 32 ノード間という実ヒューマノイドの関節数に近いノード数で、目標制御周期 10kHz 時の1 制御周期サイクル 100μsec に対して半分以下の 50μsec での全ノード間トピック通信が実現可能であることが示された。

バースト長の設定については、全体転送時間に対する寄与度は小さい。これは CPU バスからの転送データ書き込み速度や受信データの共有メモリへの書き込み速度にボトルネックがあり、同時に多数のデータを転送してもメモリ書き込み速度を上回ることができないためと考えられる。これに対しては、共有メモリの書き込み側データ幅を拡大することでメモリ書き込み速度を倍増させることを予定している。

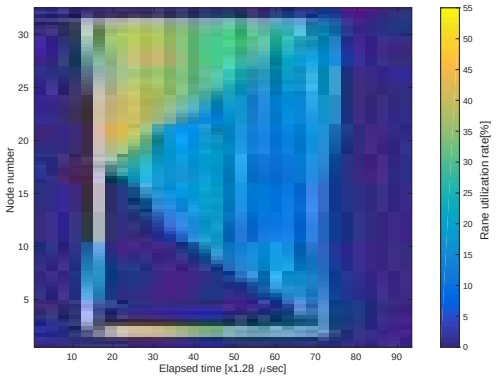
また、バースト長の設定はリンク使用率に対しては使用率の低下に寄与している。そのため、バースト長設定は転送するトピック数に応じて動的に変更することが望ましいと考えられる。リンク使用率はトピック通信全体で 50 %を下回っており、他プロトコルが使用するのに十分な帯域が残っていると考えられる。

実機での分散共有メモリの利用

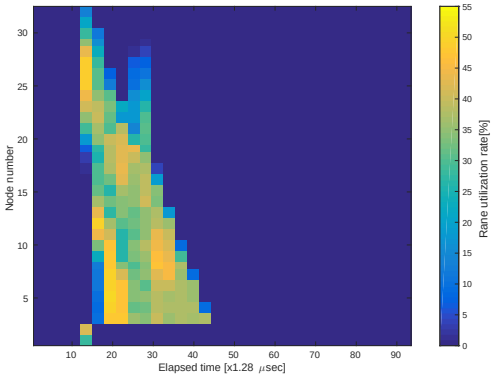
JAXON([65]) では全身 33DOF の関節を持ち、その内6 関節がダブルモータを採用しているため、全身で 39 のモータドライバを搭載している。制御用ノードを含めた 40 ノードを考える場合、シミュレーション結果図 4.38 より共有メモリプロトコルパケットの伝播完了までの時間は約 57μsec と推測される。これは、端ノードから端ノードまでのデータ伝播時間であり、例えば制御ノードが接続トポロジの中央に配置されている場合には実質的な伝播時間はこれ以下となる。



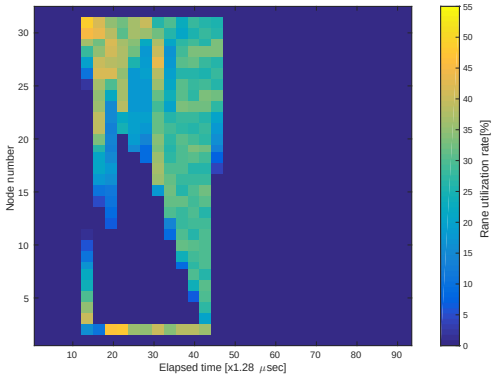
(a) Upstream with burst length 0



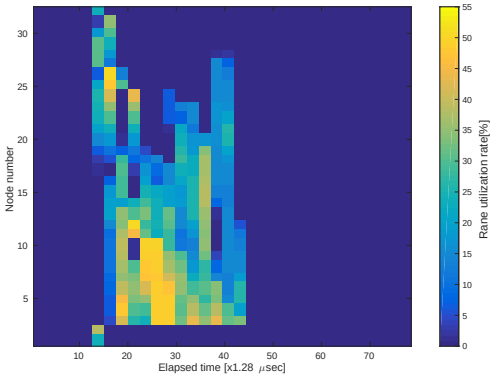
(b) Downstream with burst length 0



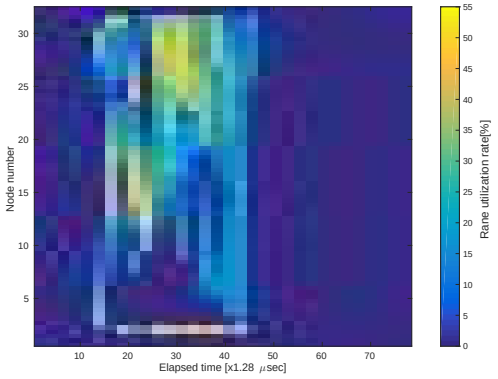
(c) Upstream with burst length 4



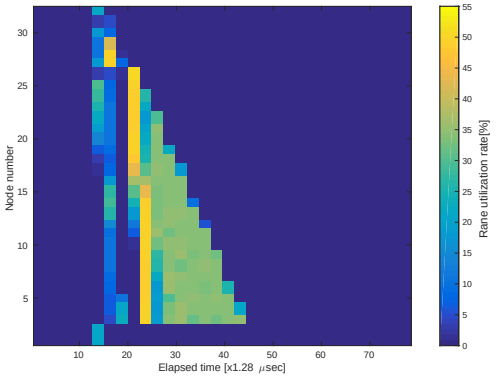
(d) Downstream with burst length 4



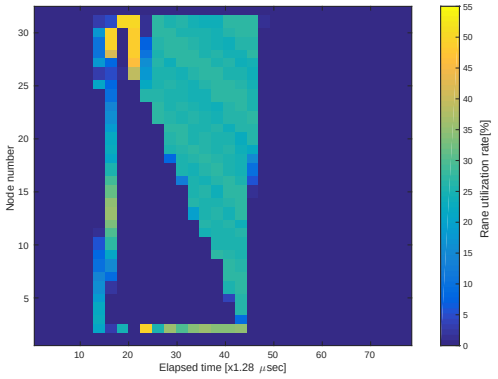
(e) Upstream with burst length 8



(f) Downstream with burst length 8

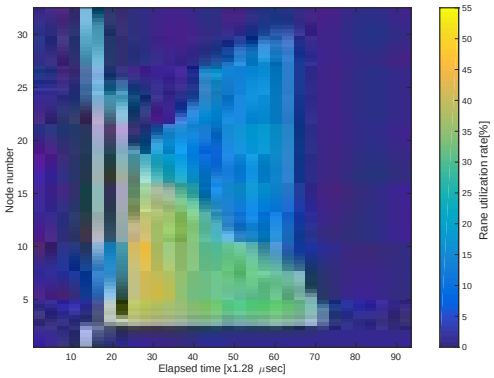


(g) Upstream with burst length 16

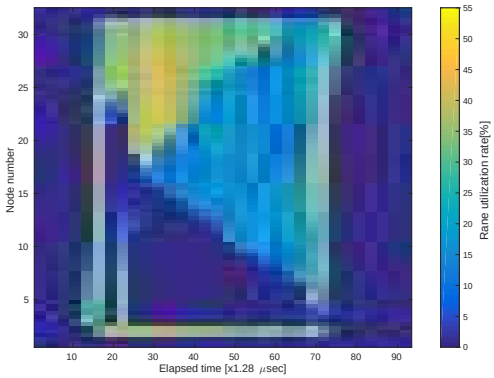


(h) Downstream with burst length 16

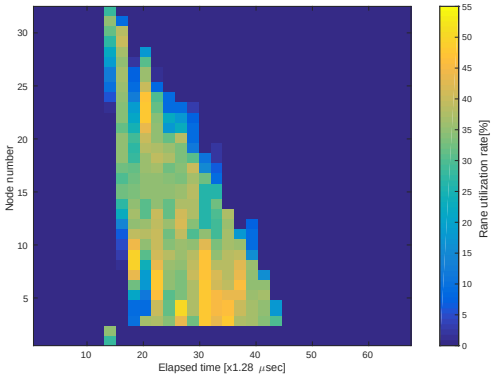
図 4.39: Upstream and downstream link utilization on 32 node system. Configured to transmit packets at the same time.



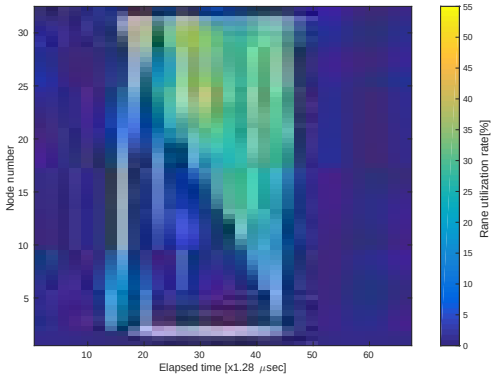
(a) Upstream with burst length 0



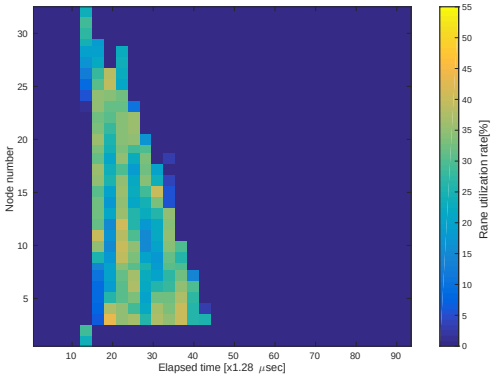
(b) Downstream with burst length 0



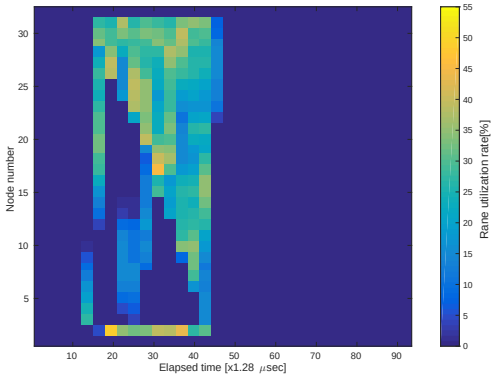
(c) Upstream with burst length 4



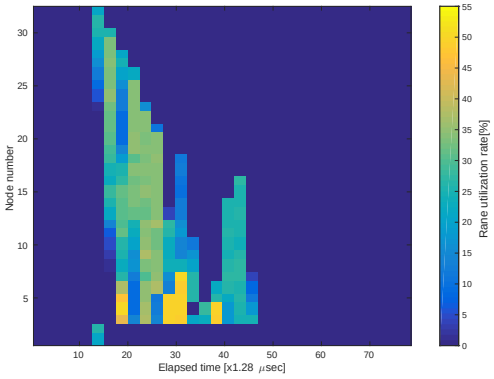
(d) Downstream with burst length 4



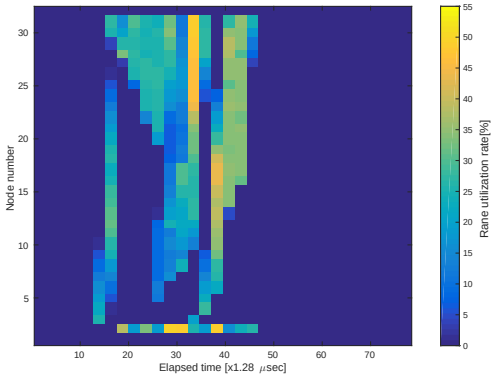
(e) Upstream with burst length 8



(f) Downstream with burst length 8



(g) Upstream with burst length 16



(h) Downstream with burst length 16

図 4.40: Upstream and downstream link utilization on 32 node system. Configured to transmit packets from edge nodes.

表 4.10: 実験条件

ノード数	4
接続トポロジ	相互全接続
センサデータ送信周期	10msec
受信スレッド実行周期	5msec

4.6 *Responsive Link* による分散制御実時間通信

Responsive Link は山崎ら [30] によって開発された分散制御用の実時間通信規格である。特徴として、実時間 OS のスレッドスケジューラ等で用いられるプリエンプティブな実時間スケジューリングアルゴリズムを、通信パケット間における通信リンク使用権の調停方式に採用していることが挙げられる。パケットに付加された優先度に応じた通信リンクの使用権の調停によって、パケット転送要求のコンフリクト時に低遅延でより優先度の高いパケットを転送させることが可能となっている。また、*Responsive Link* は1 ポートにつき Event Link と Data Link の2 系統の通信リンクを規格によって備えている。Event Link はデータペイロード長が短く、低遅延な転送が要求される用途に使用できるようパケット長は 16Byte 固定と少ないが、RMTP の CPU レジスタに直接接続されており、低遅延なデータ転送を可能としている。一方、Data Link は Event Link ほどの低遅延性は要求されないが、転送するデータペイロード長の長い用途に使用される。

また、冗長な接続トポロジを組むことが可能となっており、各ノードが持つルーティングテーブルによって同じ転送先アドレスを持つパケットでも優先度に応じて、最短経路にルーティングしたり、迂回経路にルーティングすることが可能となっている。これにより、通信ケーブルの切断や接触不良、トランシーバの不良といった障害の発生時に動的に経路を切り替えることで、通信の継続を可能としている。

4.6.1 冗長ネットワーク構成による耐障害性の検証

Responsive Link では各ノードが最大4 ポート持つことが可能となっており、各ノード自体がルーターとしての機能を持っているため、ノード to ノードで接続していくことでトポロジフリーなネットワークを構成することが可能となっている。USB 等と異なり、ネットワークを構成するためにハブやスイッチ等が別途必要となるわけではないため、ロボット体内のような空間制約の厳しい用途においては有効である。

フリートポロジネットワークでは、パケットルーティングを決定するための仕組みが必要となるが、*Responsive Link* では各ノードにルーティングテーブルを設定し、それらをパケットのヘッダ情報を元に参照することでパケットの転送先ポートを決定する。そのルーティングテーブルを設定する方法としては、

- あらかじめ実際のトポロジを元に固定のルーティングテーブルを各ノードにプログラムしておく。
- Ethernet 等で使用されている Internet Protocol(IP) を利用することで、接続状況に応じて動的にルーティングテーブルを構成する。

ことが考えられる。

本実験では *Responsive Link* をインターフェースとして備えた I/O コア基板を使用する [41]。実験条件を表 4.10 に示す。実際の基板間接続の様子を図 4.41 に示す。 接続トポロジについては図 4.42 に示すように、全ノードを相互接続している。ノード 33 に張力計測ユニットが接続されており、計測データ取得タスク及び計測データ送信タスクを走らせた。ノード 33 からノード 36 へのデータ送信経路としては、33 → 36 の単経路が最短ルートである。また、33 → 34 → 36 及び 33 → 35 → 36 の2 通りの冗長経路を通るパターンが考えられる。本実験では、パケットのデータペイロードは同一だが単経路を通るパケットと冗長経路を通るパケットの2 種類のパケットを同一サイクルで送信した。

途中、開始 30 秒時点で単経路ルートのケーブルを抜くことで通信不良を意図的に発生させている。図 4.43 に示す様に、開始 30 秒までは両経路のパケットとも目的ノードまで到達し、通信を切断した瞬間から単経路パケットからのデータは途切れているが、冗長経路パケットによって張力センサの計測値は途切れることなく連続して得られていることが確認される。

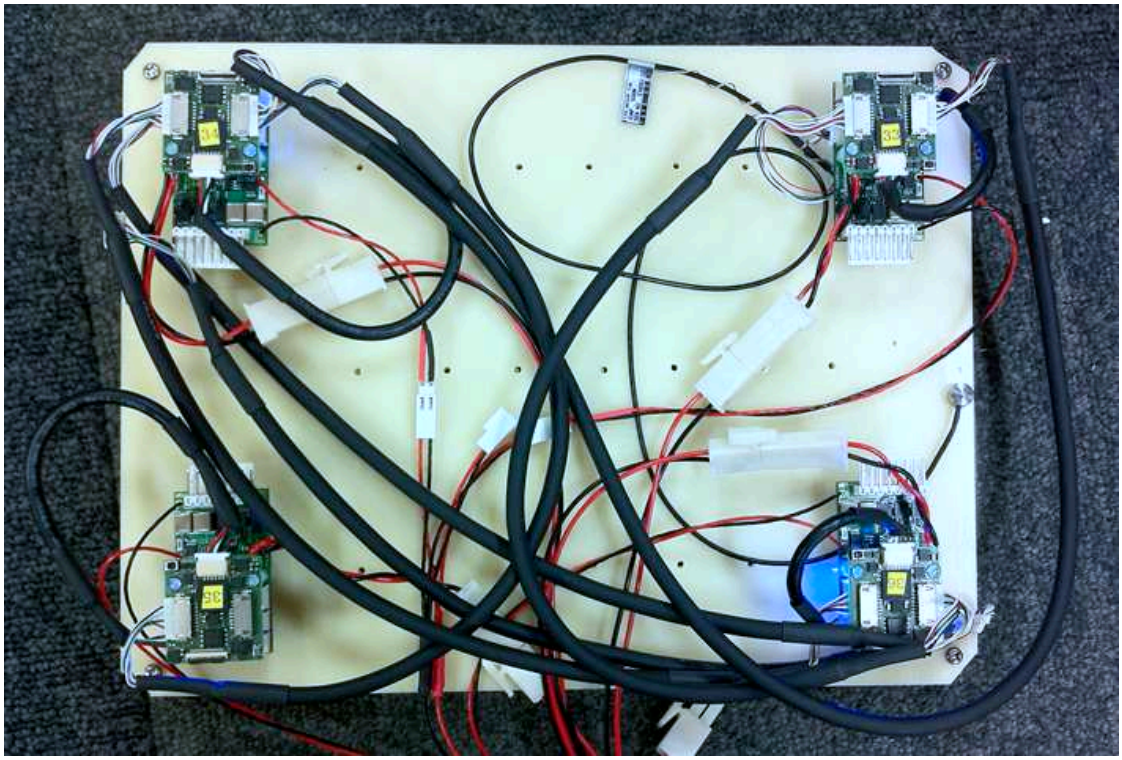


図 4.41: 4 I/O core boards are connected to each other by *Responsive Link*.

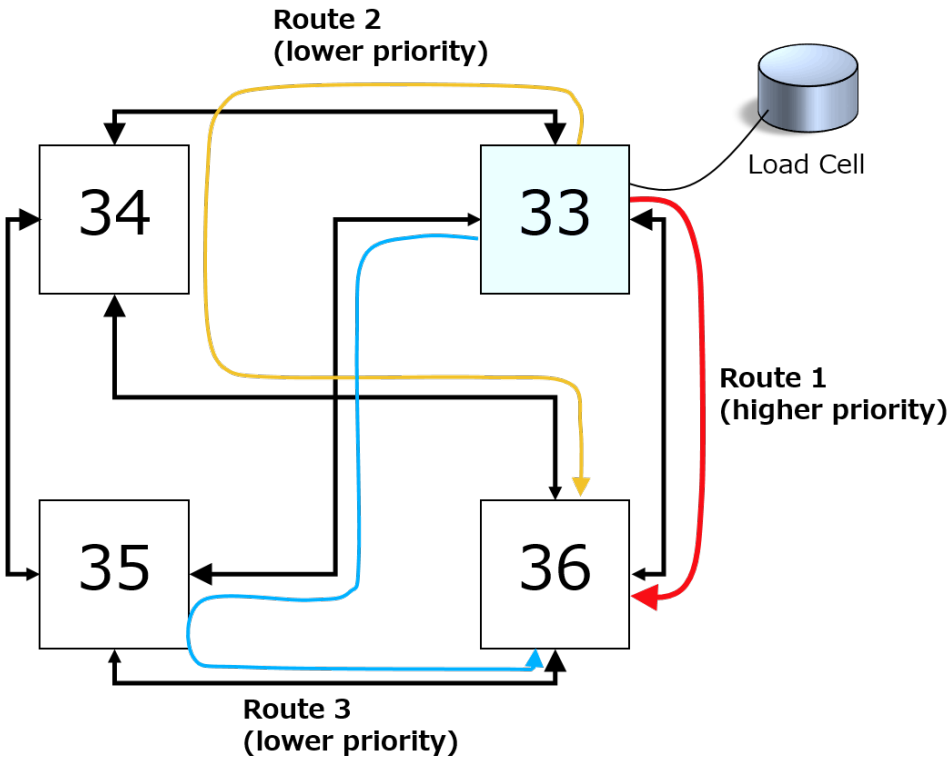


図 4.42: Redundant connection topology by *Responsive Link*.

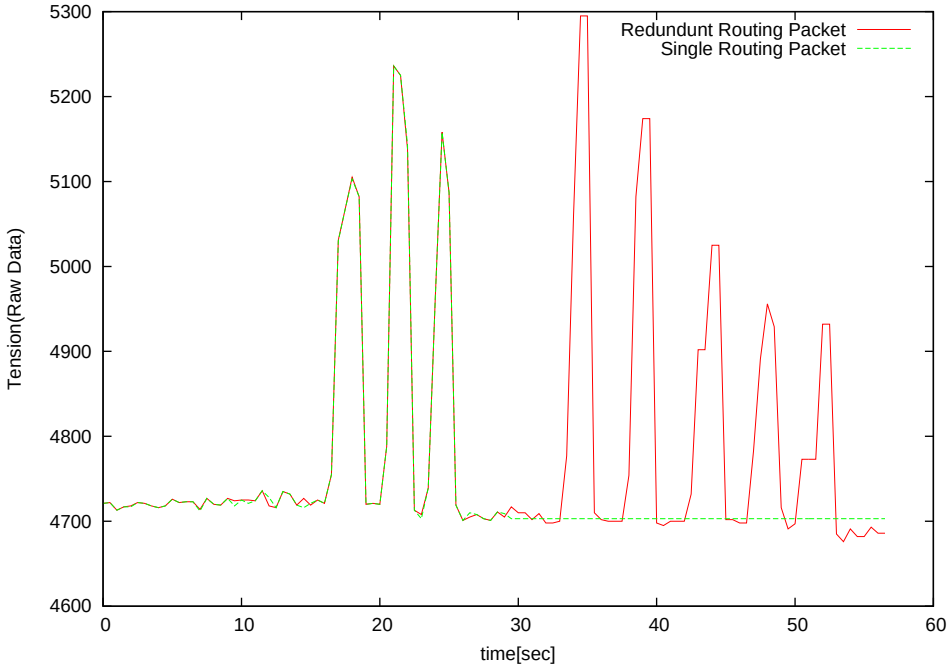


図 4.43: Verification of redunduncy of *Responsive Link* from [9](modified). The main cable was disconnected at 30sec. Then, any single routing packet data didn't reach at the destination device. On the other hand, redundunt routing packet continued to reach at the destination with out any packet losses.

4.7 本章のまとめ

本章ではヒューマノイドロボットの体内低遅延通信系の構成方法について述べた。従来のヒューマノイドロボットにて用いられてきた通信系では、ヒューマノイドロボットの通信系として要求される実時間性能・耐環境性能・データレート・アプリケーション柔軟性をトータルパッケージとして満たす構成が困難であったことを示した。

次に本研究で提案する通信系の具体的なハードウェア構成について、光アクティブコネクタと高速トランシーバを使用することによる耐環境性能の高い通信リンクを利用することについて述べ、さらに 4 本の優先度付き並列キューによる MAC 構成によって物理層の高速トランシーバのデータレートを活かした実時間低遅延データフロー制御を行うことを述べた。

続いて、ネットワーク層以上の構成として第 2 章にて述べたロボット体内通信に使用する RPC 型通信とデータストリーム型通信の2 つの通信モデルを高速なハードウェアロジックとして実現するためのプロトコルブロックの構成について述べた。RPC 型通信モデルについてはデータフロー制御において実時間性能が有利となるリング型論理トポロジと固定長軽量パケットを利用することを述べた。また、データストリーム通信モデルについては多ノード間の非同期通信を簡略化可能な分散共有メモリを利用することを述べた。その他、ロボットの体内通信にて有用となるプロトコルとして、ルーティングテーブル初期化プロトコル、グローバルクロック同期プロトコル、コネクション型プロトコルを実装している。また、これらの優先度付き並列キューへの割り振り方法について、各々のプロトコルの要求実時間性能、通信頻度、パケットサイズを考慮して決定されることについて述べている。ここで構成した通信モジュールの全容を図 4.44 に示す。

さらにアプリケーション層での通信の利用として、分散共有メモリプロトコルを用いたパブリッシュサブスクライブ型通信を導入した。これにより、組込みシステムで構成されるロボットの分散型制御システムにおいて低負荷な多ノード間データストリーム通信を構成できることを述べ、シミュレーションによって実用的なデータ更新周期を実現できることも示した。また、通信系全体のループバック遅延性能についても計測を行っており、ロボットの制御周期に対して低遅延で通信が行われることを示した。

最後に、ロボットの体内での冗長な自由トポロジ構成を実現するために *Responsive Link* による耐障害性能について実験によって検証を行い、その有用性を示した。

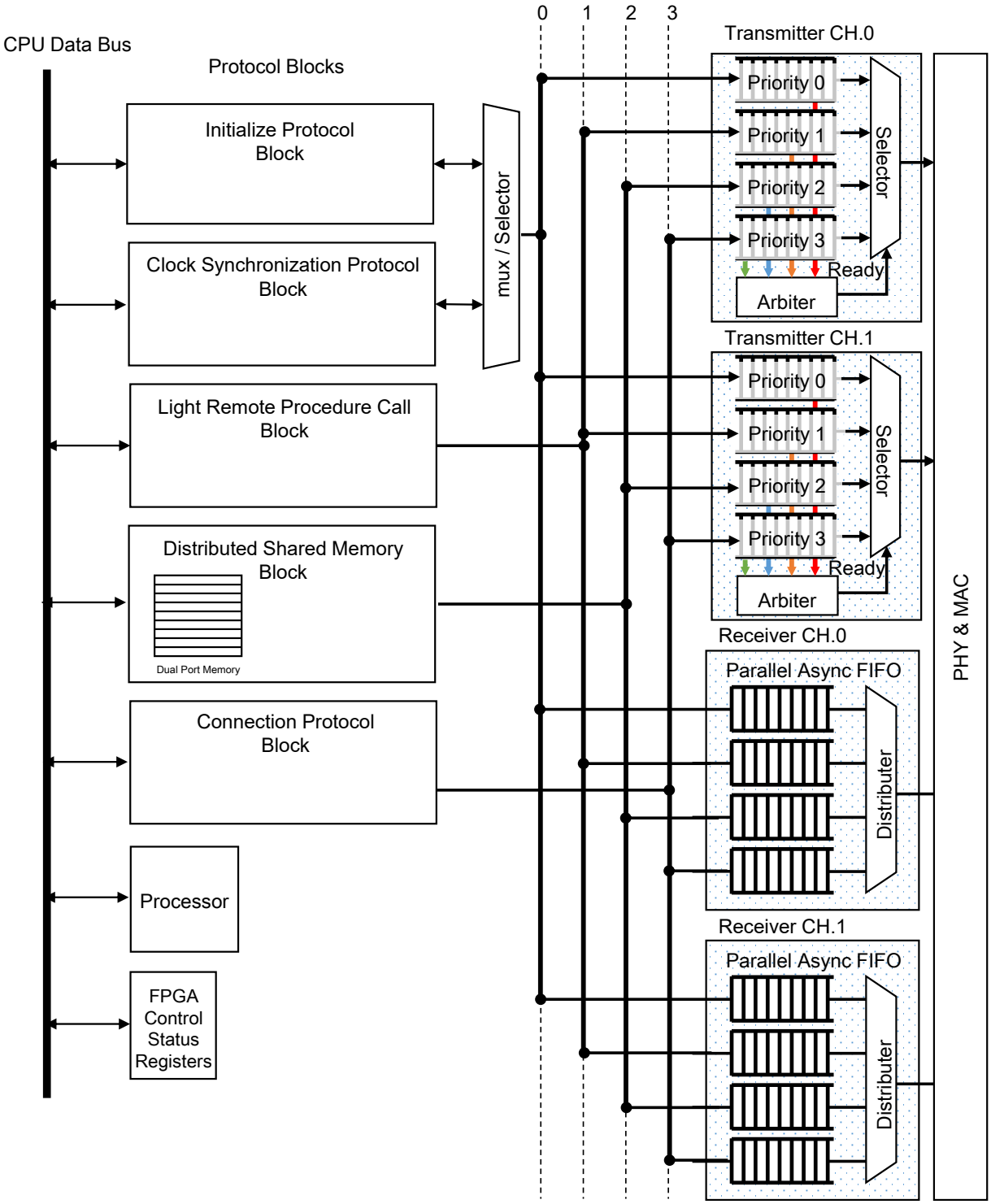
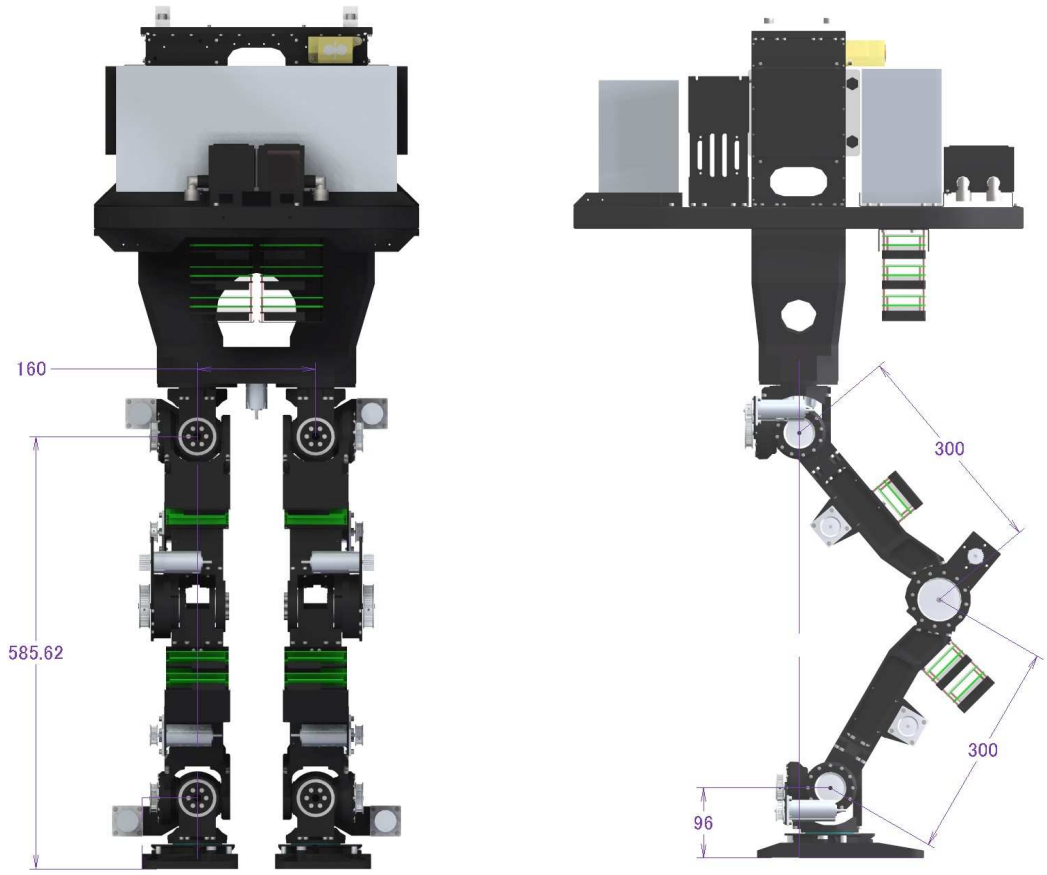


図 4.44: Overall schematics of proposed communication system.

第5章

ヒューマノイドロボットにおける高速応答動作を実現する実行系システムの開発



(a) Front View of HRP3L-JSK CAD Model.

(b) Side View of HRP3L-JSK CAD Model.

図 5.1: HRP3L-JSK CAD Model

5.1 はじめに

本章では前章までに述べた実時間分散制御モータ制御モジュール及びモジュール間低遅延通信系を利用して、実際にロボットを動作させる制御システムの構成方法について論じる。まず 5.2 節にて、従来のロボットで使用されてきた上位制御システム並びに全体のシステム構成について簡単に解説を行う。

次に 5.3.1 節にて本研究で提案する即応的反射行動アーキテクチャについて述べ、その中核となる中間制御層について 5.4 節にて導入を行い、さらにその構成モジュールについて 5.5 節にて述べる。また、本システムで使用されるサブシステム間通信について、5.6 節にて検証を行う。中間制御層システムは多ノード構成を採用することが可能となっており、5.7 節にてその構成方法を述べる。

最後に、高速応答動作の要件として上位制御システム側でのオンライン軌道生成手法について 5.9 節にて述べる。本章で導入した実行系システムの動作検証については、次章にて腕型ロボット、脚型ロボットを用いて述べていく。

5.2 大出力モータドライバを持つ HRP3L-JSK と従来型上位システム構成

5.2.1 HRP3L-JSK

HRP3L-JSK は浦田ら [50] によって開発された大出力モータドライバを搭載した脚ロボットである。HRP3L-JSK の外観と諸元を図 5.1a 、図 5.1b 、図 5.2 , 表 5.1 に示す。特徴としては、瞬間的にモータの定格を超える電流を出力することで、人間の関節トルクと速度に近い特性を単一のモータ・減速機によって実現して

図 5.2: Isometric View of HRP3L-JSK CAD Model.

いることが挙げられる。また、大電流による熱損を防ぐためにモータとそのアンプ基板は水冷システムによって冷却が行われている。

浦田らによるオリジナルの HRP3L-JSK では電源装置に並列接続されたスーパーキャパシタによる電力供給が行われていたが、本研究ではそれは外している。そのため、瞬発的な動作については電圧降下によってモータドライバ本来の最大性能は、本研究の実験においては発揮できていないが、歩行等の最大出力を必要としない動作については電源装置からの給電で、多少電圧降下が生じても問題は生じない。

表 5.1: Specification of HRP3L-JSK

Weight	44[kg]
# of joint	12 (6 for each leg)
Gear Ratio	240:1 (160:1 by harmonic drive and 1.5:1 by pulley)
Water-Cooling	Electrical pump and radiator
Sensor	six axis force sensors (on each foot) Inertial Measurement Unit (IMU)
Rated Voltage	80V (max)

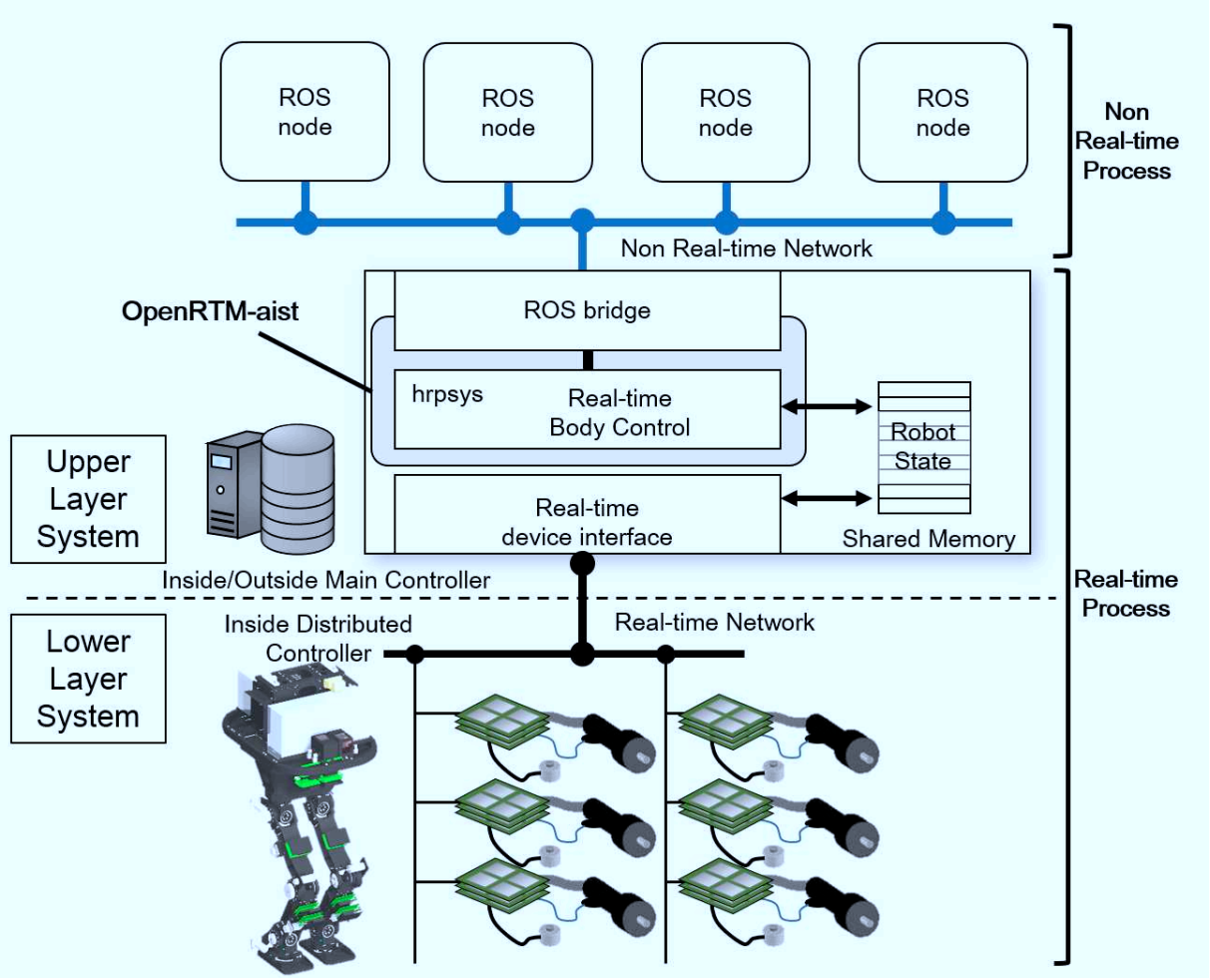


図 5.3: Traditional upper layer control system of HRP3L-JSK.

5.2.2 従来型上位システム構成

体内に搭載された分散型モータ制御システムについては、第 3 章及び第 4 章にて述べたが、それらの運動制御機能の大本となる上位の計算機システムについては、浦田らによるオリジナルの実装から STARO,JAXON といったロボットで共通に使用されているロボット制御用プラットフォーム [67] へと置き換えている。このプラットフォームでは、ソフトウェアモジュール群を束ねるミドルウェアソフトに ROS[68] と openRTM[69] を使用したの2 つのシステムが統合された構成となっており、要求される実時間性能や計算負荷に応じて階層分けされている。従来型上位システムの構成図を図 5.3 に示す。上位システムは x86 アーキテクチャの PC 上で Linux OS をベースに構成される。

主となるロボットの身体制御は金広らによる openRTM をベースとした hrpsys[1]を利用して行っており、目標関節角度軌道等が出力される。hrpsys とデバイスインターフェースは独立したプロセスで実行されており、共有メモリを介してデータ通信を行っている。hrpsys 内には、関節軌道生成用コンポーネントや歩容生成コンポーネント、バランス補償制御用コンポーネント等多数用意されており、実際のロボットに必要な機能に応じて構成を変更して利用する。これらのコンポーネントは hrpsys が提供する実時間コンテキスト上でシリアライズされ、同一プロセス上で順次実行される。新たにコンポーネントを追加することで、身体制御則を追加することも可能であり、後述するオンライン歩行生成器もこの hrpsys 用コンポーネントとして実装を行う。

デバイスインターフェースプロセスでは、hrpsys から受け取った参照関節軌道、パワースイッチコマンド等のイベントを体内分散型制御基板への通信パケットに変換し、デバイスドライバを介して送信を行う。逆に分散型制御基板へデータリードリクエストを送信し、受信パケットの解釈と hrpsys への受け渡しを行

[1]https://github.com/fkanehiro/hrpsys-base

う。これらは全て実時間実行される。基本的にはデバイスインターフェースプロセスでは制御に関わることはなく、hrpsys 等の身体制御用サブシステムと全身のモータ制御サブシステムとの間でデータやイベントをブリッジするのみである。

hrpsys のさらに上位層には ROS ノード群を接続することが可能となっており、実際にロボットへ人が直接指令を与える場合にはこのノード群から行う。本研究ではコマンド実行のヒューマンインターフェースとして、Euslisp[2]を ROS ノードとして実行することで利用する。画像処理等の認識系の処理もこの層で行われるが、本研究の対象範囲外なので割愛する。

5.2.3 従来型システム構成における課題

5.2.2 節にて述べたように、従来の上位システムでは身体制御系の演算処理やインターフェース処理を一括して行われている。また、hrpsys では主となる身体制御処理コンポーネントを 1 コンテキスト内でシリアライズされて実行されることで、制御系全体の処理が俯瞰しやすくなり、開発の容易性を与えている。

しかしながら、完全な実時間 OS 上で実装されているわけではなく、各種インターフェース処理や負荷の高い演算処理と混在していることによって、外乱入力に対して低遅延で高い実時間性を維持した応答処理を新規に開発し実装することが困難となっていた。実際、IMU や足裏力センサ入力をもとにバランス制御を行うスタビライザ—コンポーネントは制御周期をさらに高速化し、応答を低遅延化することで性能の向上が期待されるが、シリアライズされた hrpsys 全体の処理速度がボトルネックとなり、現時点では 2msec 周期に留まっている。

応答性が要求される処理については、hrpsys の実行コンテキストを分割・独立化し、実行周期と実行優先度を高めた設定とすることで改善する方法も考えられる。ただし、有限なハードウェア処理能力やインターフェースを他のプロセスと共有する以上、他処理との完全な独立化は困難であり、システムやアルゴリズムの更新によってシステム負荷が増大した場合に、更新前と同一の制御パラメータで継続して実行可能である保証はない。ロボットの体内制御系の更新が数サイクル遅れとなるのはこのような保守性や継続性の悪さが一因として存在している。また、独自のインターフェースデバイスを用いていることでもいくつか問題が生じており（ 詳細について 5.4 節にて述べる）、要改善点となっていた。

5.3 即応的反射行動アーキテクチャ

5.3.1 即応的反射行動

固い物体との衝突や躓きによる転倒のように、本来の計画された動作軌道を実行すると行動失敗につながるような状況では即座に失敗回避行動をとった上で、動作軌道の再計画を行う必要がある。人間では身体を構成する生体組織の柔軟性によって動作計画との誤差を緩和する身体性や、脊髄反射系により脳での情報処理を介さない即応的な動作指令が運動神経系に発行される仕組みが備わることで怪我を防ぐ頑健性を得ている。また、とっさの動作によって本来の運動計画軌道から外れた場合にでも即座に適切な運動軌道が与えられることで転倒を防いでいる。ロボットの体内制御システムにおいても、このような柔軟な身体性、反射神経系の実装によって急な入力に対して、動作軌道再計画を待たずに即応的な動作により機械構造に過大な負荷が加わることを防ぐとともに、動作軌道の実時間での再計画によって目標タスクの失敗を防ぐことが考えられる。関節の構造に弾性要素や粘性要素を付加することで柔軟身体を実現する方法も採れるが、ハードウェアの更新が必要となるため本研究では対象外とし、制御則による柔軟身体の実現を目指す。

例として、図 5.4 に外乱によって重心位置が目標軌道から外れた際の即応的な動作による外乱吸収と、歩行動作軌道再生成による転倒防止を実行する様子を示す。ここでは①外乱によって重心位置が乱されるが、②では関節の柔軟性によって即座に体幹リンク部を外乱吸収する方向に移動させ、③では歩行動作の再計画によって得られた新規の運動軌道によって転倒を回避し、最終的に安定な状態に復帰するというシナリオで実行されている。このようなシナリオに基づいた動作を本研究では**即応的反射行動**と呼ぶことにする。即応的反射行動では、外乱に対して極めて短い時間で応答動作を実現する**即応的動作制御系**と、本来のタスクを実現するために必要な身体運動軌道を再生成する**身体運動計画系**を持つ必要がある。図 5.5a に示すように即応的動作制御系は反射行動のように状態入力に対し、最短となる制御演算のみで動作へとフィー

[2]https://github.com/euslisp

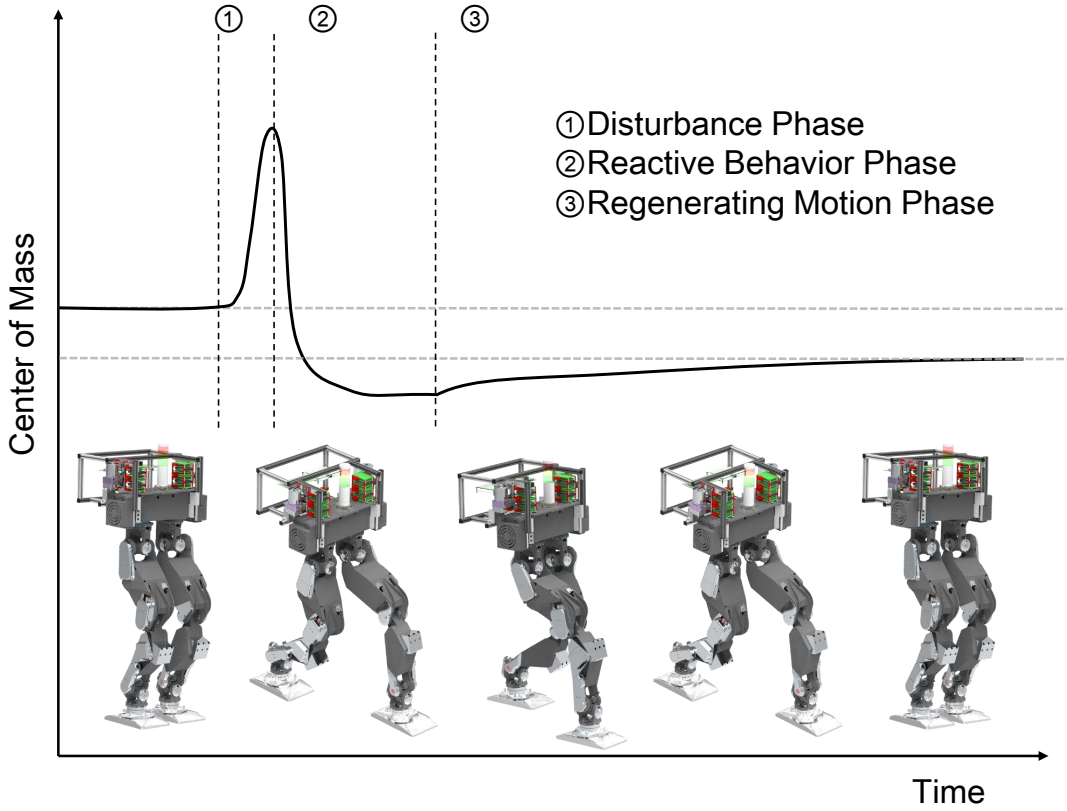


図 5.4: Example of reactive behavior.

ドバックされる。一方、状態入力は身体行動計画系へも伝達され、オンラインでの身体動作生成器を通してロボットへの制御信号にフィードバックされる。身体運動計画系では最適化アルゴリズムを用いることで、演算時間はかかるが現在状態に適した軌道を生成することが可能となる。

これは、制御フローが2 系統あり、即応的動作制御系が最短でのフィードバックを目指すシステム、身体運動計画系が最適なフィードバックを目指すシステムとして実行され、反射行動系を構成している形となっている。図 5.5b に示す従来の基本的な制御ループと比較して、制御ループが多重化され、同時にシステムが多階層化していることを見て取ることができる。

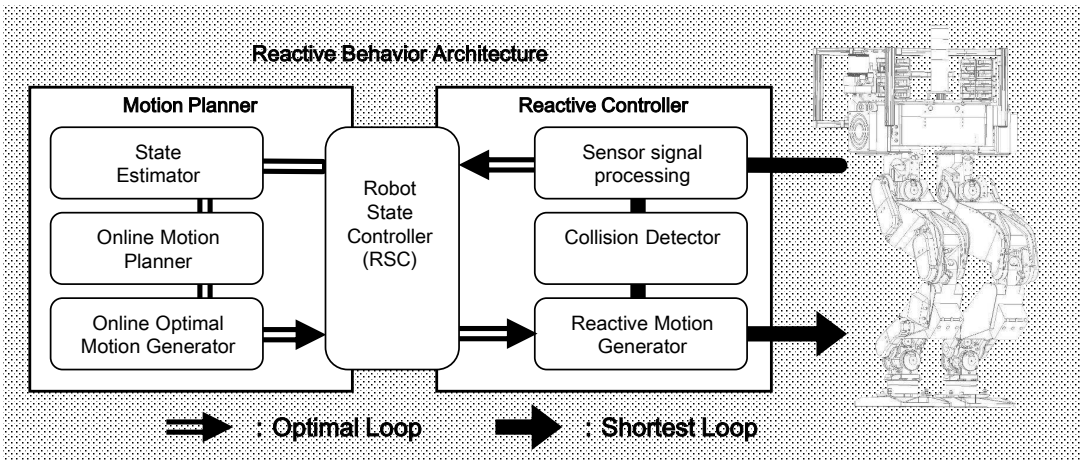
5.3.2 即応的反射行動アーキテクチャに基づいた高速応答行動

外乱入力に対して最短制御演算のみで動作へフィードバックを実現する方法として、関節のサーボ制御則のトルクベース化を考える。従来は関節角度サーボがベースとなっていたため、衝撃のような外乱が入力された場合ににおいても上位システムが目標関節角度軌道を修正しないかぎり外乱に対する応答が行われないため、応答遅れが生じていた。トルクベース化することで、上位システムの制御コマンドの修正を待たずに外乱入力に対してサーボ制御は物理的な作用によって外乱入力を緩和する方向に高速に応答することが可能となる。

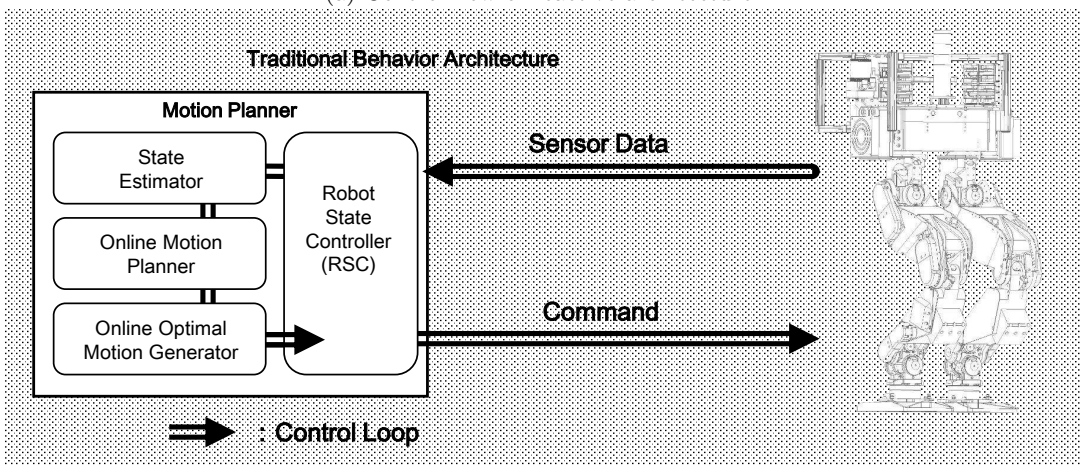
上位システムは関節トルクベース制御が外乱入力に対して緩和動作をしている間に、転倒回避等の新たな目標動作軌道を生成し直すことで、全体としてのタスク達成を実現する。

5.3.3 3 階層制御システム

即応的な身体運動実行系と身体運動計画系では要求される実時間性能と演算性能が異なる。前者は入力に対して 1msec 以内の極短時間での応答を要求する実時間処理である。これに対して、後者は実時間性能としては数 msec 程度の猶予はあるが、高い演算能力を要求する処理である。第 2 章及び 2.9 節にて述べた



(a) Control flow of reactive architecture.



(b) Control flow of traditional architecture.

図 5.5: Comparison of system architecture.

ように、これらの要求を同一の計算機上で実現することには性能上の制約が伴う。そのためこれらの要求を満たすために、それぞれの要求性能に見合った計算機システムに処理ごと分割することが 1 つの方法であると考えられる。本研究で提案する即応的反射行動アーキテクチャでは、ロボットの体内制御システムを以下のように3 つのサブシステムを重ねた階層型制御システムとして構成する。

1. 上位層制御システム
   従来型システムと同様、高度な身体運動計画系の制御を行う。最短で 1-2msec 程度の制御周期、応答でシステムが実行される。

2. 中間層制御システム
   即応的な力応答を実現するためのサーボ制御、センサー情報処理、サブシステム間通信処理を行う。最短で 100-200μsec 程度の制御周期、応答でシステムが実行される。

3. 下位層制御システム
   アクチュエータやセンサデバイスの直接的な制御を行う。

それぞれのサブシステムは上位・下位のサブシステムと低遅延な実時間通信系によって接続され、制御用信号・データを送受信し合う。この3 階層制御システム構成を図 5.6 に図示する。図 5.3 と比較すると分かるように、従来型の制御システムの構成の中間に通信リンクをブリッジする形で中間層制御システムが挿入されており、上位層・下位層の構成には大きな変更点は無い。

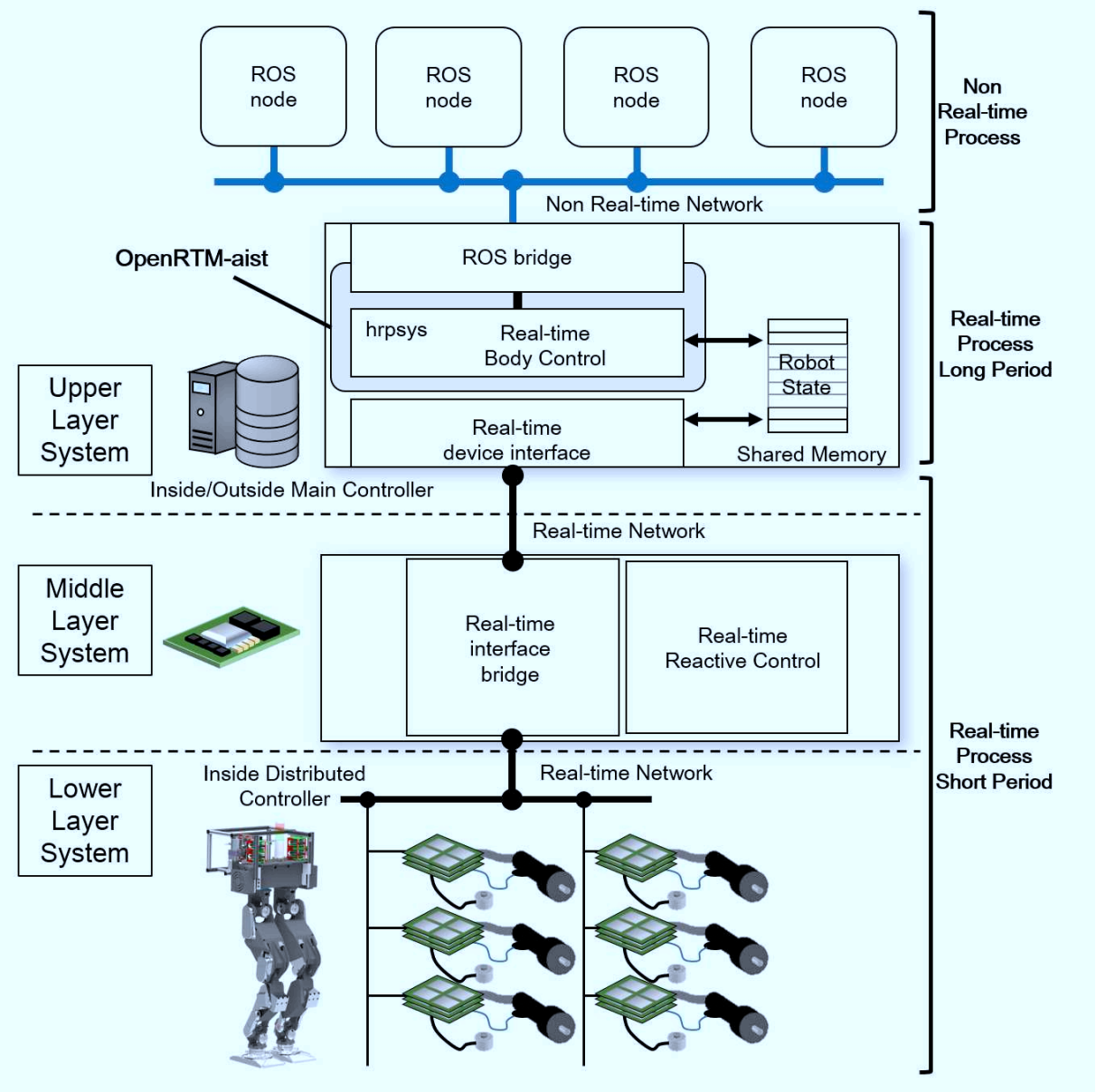


図 5.6: Three layered robot contol system with inserted middle layer controller.

5.4 中間層制御システム

5.2 節にて述べた従来型の通信系では、標準規格ではない独自のデバイスインターフェースを実装していた。この独自デバイスインターフェースを用いる利点と問題点について主要なものを以下に洗い出し、簡潔にまとめる。

- 従来型独自デバイスインターフェースの利点 (利点となる評価項目)
  1. 拡張ボード側で通信処理を分散処理 (頑健性、制御性能)
  2. 必要最小限構成によるパッケージサイズの最適化 (パッケージサイズ、入手性)
  3. 耐ノイズ性能、耐障害性能に優れたプロトコル、物理媒体 (頑健性、耐障害性)
  4. ロボットの構成に最適化したパケットサイズ、データ構造 (拡張性、柔軟性、制御性能)
  5. スレーブ側の基板の通信機能に合わせた最適化 (パッケージサイズ、制御性能)
  6. ロボットに合わせた独自の安全機能 (信頼性、耐障害性)
- 従来型独自デバイスインターフェースの問題点 (問題となる評価項目)
  1. デバイスドライバの開発・メンテナンスが必要 (保守性、容易性、継続性)
  2. 独自デバイスの開発・製造が必要 (保守性、入手性、互換性)
  3. 限定的なデータアクセス性 (検証可能性、容易性、拡張性)
  4. サードパーティ製デバイスは別途インターフェースが必要 (互換性、柔軟性、パッケージサイズ)
  5. 耐振動・耐衝撃性能の低い拡張スロット † (頑健性、耐久性)
  6. 拡張性はスロット数に依存 † (拡張性)
  7. マザーボードの選定に制約 † (互換性、拡張性、入手性、継続性)
  8. 拡張ボードレイアウトに制約 † (パッケージサイズ)

† の項目については、接続用のインターフェースとして PCI カード型を採用していることに起因する問題である。図 5.7 に、さらに従来システムにおける問題点と、それに対応する 2.3 節にて述べたシステム評価項目について表形式でまとめている。

5.4.1 中間層制御システムの透過性

上位層のシステムからは目標値、モードといった制御コマンドが各モータドライバへと送信されている。また、各モータドライバのセンサデータも上位層まで返信される。これらのデータ通信は、上位層からは中間層制御システムへ宛てたパケットとして送信し、中間層にて振り分けてもらうのではなく、各モータドライバと直接送受信されたデータとして扱ったほうが分かりやすい。従来型のシステムとの互換性を保てるという点や、中間層制御システムを意識しない開発が行える点で利点が有る。中間層の存在に依存しないということは、システムの冗長化による信頼性の向上にもつながる。このように、上位層制御システムが中間層制御システムの存在に依存せずに動作可能な性質を**中間層制御システムの透過性**と呼ぶことにする。

図 5.8 に示すように、透過性を持たせるために上位層システムからのデータは通信ブリッジを介して全て下位層システムに送信され、中間層で破棄されない構造となっている必要がある。本研究では通信ブリッジの機能を全て FPGA 上のハードウェアロジックによって実装することで、中間層システムのソフトウェア稼働状態と独立して実行可能とし、ハードウェア的エラーが生じない限り制御コマンドは下位層へと転送されることで透過性を与えることにした。ただし、即応的動作制御において中間層制御システムからの制御コマンドが優先される必要があるため、下位層のモータドライバのロジック部には適切なシステムからの制御コマンドを選択するためのソフトウェアスイッチが実装されている。このスイッチは中間層制御システムを優先するが、中間層制御システムからのデータが途絶えた場合にウォッチドッグタイマによって、自動的に上位層システムからのコマンドを利用するように切り替わることで、中間層制御システムがダウンした際の上位制御システムによるバックアップを可能としている。

カテゴリ	従来システム問題点	制御性能	連続稼働時間	安全性	頑健性	耐久性・堅牢性	パッケージサイズ	入手性	保守性	容易性	継続性	拡張性	柔軟性	互換性	可観測性	検証可能性
致命的	障害時は完全停止	×	×													
	絶縁不良による信号ノイズ				×											
	振動・衝撃でシステムダウン			×	×											
	ケーブル・コネクタの劣化、摺動による接触不良			×		×										
	放熱不良によるCPU、インバータのオーバーヒート		×			×	×									
機能制約	デバイスインターフェースの空間占有体積が大きい						×		×							
	ケーブルによる空間占有体積が大きい						×		×							
	使用デバイスに対応したマザーボードから選定	×					×	×			×	×		×		
	下位システムへの限定的なデータアクセス性									×		×			×	×
開発効率	独自デバイスの開発・製造							×	×					×		
	独自デバイスのドライバの開発・保守								×	×	×					
	制御システムの更新でデバイスドライバ、インターフェースに不具合							×	×		×			×		
	デバイスごとにそれぞれインターフェース・配線が必要						×		×	×		×	×	×		
	インターフェース処理と制御演算処理が同一マシン上	×									×		×	×		
	障害発生時の詳細なログが記録されない								×						×	×

図 5.7: Problems of the traditional robot control system.

5.4.2 中間層制御システム構成

中間層の主な構成機能として、

1. 上位サブシステムと下位サブシステム間のデータ通信中継
2. 即応的動作制御実行系
3. データ通信ロガー

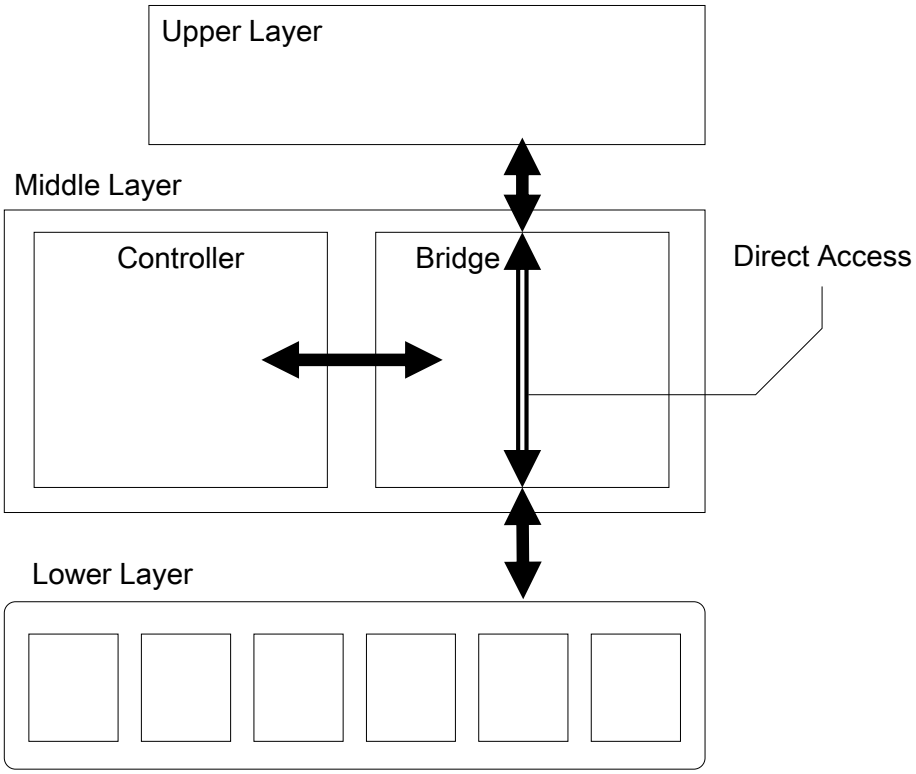


図 5.8: Transparency of middle layer system.

4. 上位系シャットダウン時のバックアップ

を 5.3.1 節にて挙げた。これらは通信インターフェース部とプロセス実行部に分けられる。本研究ではこれらを FPGA-SoC 基板とインターフェース基板を組み合わせて構成することにした。このシステムのハードウェア構成については次節 5.4.3 節にて述べる。項目 2 については、項目が多岐に渡るため、次節以降で述べる。以下では、ハードウェア構成について述べた後、2 項を除く中間層制御システムとして実装を行ったそれぞれの機能項目 1,3,4 について順に述べる。

5.4.3 ハードウェア構成

図 5.9 , 図 5.10 , 図 5.11 に実際に本研究で中間層制御システムとして使用したインターフェース基板及び FPGA-SoC 基板を示す。インターフェース基板はオリジナルの設計となっており、光アクティブコネクタ用ポートを 4 チャンネル搭載している。FPGA-SoC 基板には市販されている評価キットである Terasic 社製 SoCKit[3]を採用した。

図 5.12 に、ハードウェア構成の中で主要な機能を図示する。FPGA-SoC 内部は Hard Processor System (HPS) 部と、FPGA 部に分かれている。HPS は ARM コアで構成されており、2 コア構成となっている。主要な汎用インターフェースとして USB OTG (USB On-The-Go) とギガビット Ethernet を備えている。FPGA 間とは ARM AXI バスによって内部的に接続されているため、高速なアクセスが可能となっている。FPGA 側には本研究で提案する中間層制御システムを構成するための主要機能として、光リンク通信モジュールと EtherCAT ブリッジモジュールを備えている。また、通信系の割込み処理を HPS とは独立して行うために軽量なソフトウェアコアである Nios II プロセッサを導入している。光リンク通信モジュールは基本的な構成は RMTP-02D 基板に実装したものと同一であるが、EtherCAT ブリッジからのデータフローと接続するために追加の RPC 型通信インターフェースマスターを実装している (5.4.6 節)。

これらの基板は CPU、FPGA、インターフェース、その他デバイス全てを含めて消費電力は 10W 未満に留まっており、ファンによる強制空冷やヒートシンクによる放熱が無くても運用が可能となっており、耐振

[3]www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=167&No=816

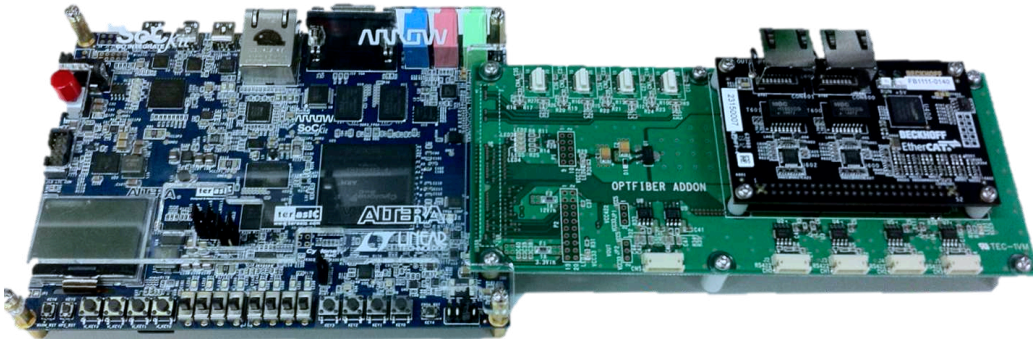


図 5.9: Middle Layer Control System.

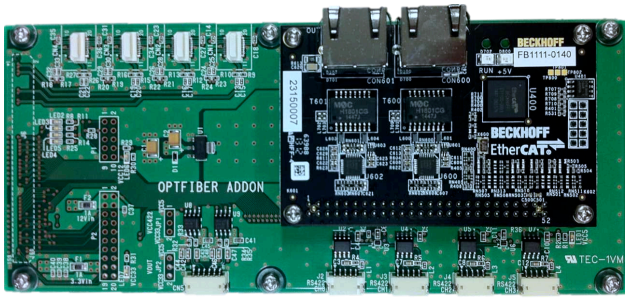


図 5.10: Interface board.

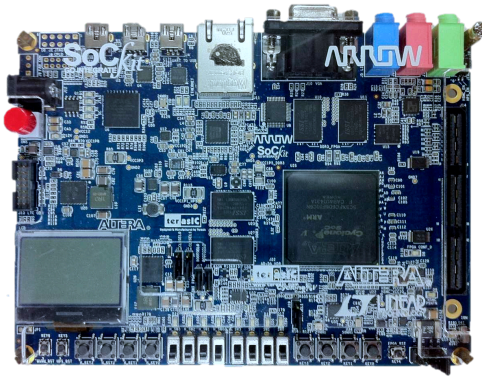


図 5.11: FPGA-SoC board (terasic SoCKit).

動· 耐衝撃· 耐熱性能に大きく貢献していると考えられる。システムの基幹となるプロセッサである ARM Cortex-A9 Dual-Core は 800MHz のクロックで駆動される。ARM Cortex-A9 は 2.5DMIPS/MHz の性能を有しているため、全体では 4,000DMIPS を計上する。上位層システムを構成する Intel 製プロセッサは 100,000DMIPS 以上の性能を有するため、プログラム処理性能については 10 分の 1 以下となるが、FPGA 側の回路と密に接続されているため、FPGA 側のロジックによって処理能力の底上げを行うことが可能となっている。本研究で提案する通信モジュールはその例の一つとなる。また電源は 12VDC の単電源でよいため、バッテリーでの運用が簡単にできる構成となっている。重量は拡張基板を含めて、260g と軽量に収まっている。マウント用のスルーホールも多く設定できるため、ロボット内部に組み込む際に、衝撃が伝わりにくいように防振用ゴムを挟みこむことも可能なため、より高い耐振動· 耐衝撃性能が期待される。

本研究の段階では SoCKit の評価基板を用いているが、将来的には独自の設計で中間層制御システムのハードウェア全体を構成することも可能であり、それによって全体のパッケージサイズの縮小や重量の削減、不要パーツの削除によるコストや消費電力の削減が期待できる。

5.4.4 分散システム用ミドルウェアによるサブシステム間通信

中間層制御システムは汎用の Ethernet ポートを備えており、また Linux OS をベースとしてシステムが実行されているため、既存の分散システム用ミドルウェアを活用したデータ通信を行うことが可能となっている。ミドルウェアとしては特にロボット用として開発されており、上位層制御システムでも広く利用されている ROS 及び OpenRTM-aist を本研究では利用している。

分散システム用ミドルウェアを利用することで、独立したハードウェア上のシステムのデータや機能へも容易にアクセスすることが可能となり、開発の容易性向上や処理の分散化による性能向上が期待される。しかし、これらは Ethernet によるシステム間通信を TCP または UDP によるものを原則としており、通信

[4]SoCKit User Manual (rev.D Hardware)
[5]Cyclone V Device Overview https://www.altera.co.jp/documentation/sam1403480548153.html

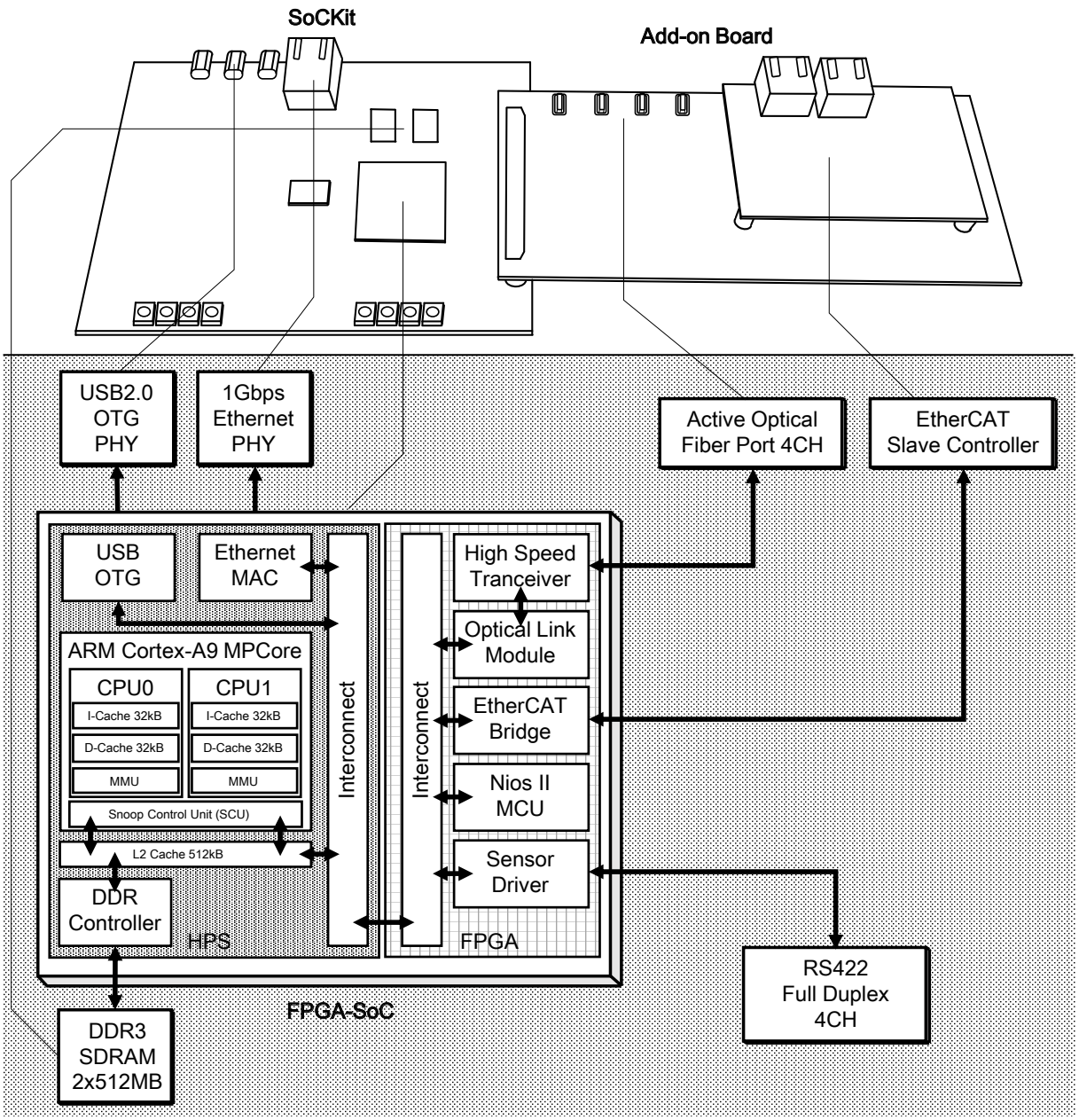


図 5.12: Middle layer system hardware schematics (refered to SocKit Documents[4] and Altera's data sheets[5]).

をソフトウェア処理に依存する部分が比較的大きい。また、これらの通信はロボットの体内制御系で要求される細分化された制御サイクルにおける実時間性能を満足するようなスペックではない。従って、処理能力の限定的な組込みシステムや制御システムでは基幹となる通信系として利用することはできず、パラメータ設定やモード切替等の非実時間用途での補助的な利用が主となる。

基幹となるサブシステム間通信には次節で述べる EtherCAT による同期通信を利用する。なお、分散システム用ミドルウェアによる通信の評価は EtherCAT による同期通信の評価と合わせて、5.6 節にて行う。

5.4.5 EtherCAT による上位層との同期通信

中間層制御システムと上位層の間の実時間データ通信規格として、EtherCAT を使用する。EtherCAT は産業用イーサネットの中では特殊な構成を採用しており、マスターとなる上位層側のハードウェアは標準のイーサネット規格、スレーブとなるデバイス側のハードウェア構成は EtherCAT のための専用のデータ

リンク層を規定している。これにより、上位層システムは標準のイーサネットインターフェースと持った汎用計算機を使うことが可能である一方で、デバイス側では通常のイーサネット構成では難しい低遅延性、実時間性を提供することが可能となっている。こうした特徴から可用性や実時間性能についてはロボットシステムにおいて高く評価することができ、実際に制御モジュール間の実時間通信リンクとして産業利用や複数のロボットで採用されている。本研究でも、この可用性と実時間性能がロボット上位制御システムとのインターフェースに適していると考えた。

しかしながら、通信の基本的な枠組みはマスター・スレーブ型の構成となっており、マスター主導での通信パケットしかネットワーク中には流すことができない。全ての通信はマスターから送信されるパケットに同期して行われる。任意タイミングでのスレーブ間通信が行えない点や、マスター側計算機の障害発生時にスレーブ側システムは完全に孤立する点で、体内制御システムの高速応答性や自律性の獲得を目指す本研究においては要求を完全に満足するものではない。また、物理層が 100BASE-TX を規定しているため通信のデータレートは 100Mbps が基本となる。EtherCAT はパケットのデータフローをデバイス間でシームレスに転送することで、非常に高いリンク使用率を実現している。しかし、本研究で要求するロボット体内システムでの多ノード間多種類トピック転送を実現するにはデータレートは 400Mbps 以上必要であり、データレートの点からも体内システム用ネットワークとしては不十分であると言える。また、イーサネットフレームをベースとしているため、パケット長は原則として 1,500Byte までとなっている。マスター側での処理の都合上、1 パケットに全ノード分の必要送受信データを載せられるのが最適であり、これを超える場合には通信性能に制限が生じる。小数ノードでの運用であれば問題は無いが、全身 40 ノード以上といった多ノードでの運用となり、さらに各ノードとやり取りするデータ量自体が増加した場合には、1,500Byte(実際にはヘッダ等に利用するサイズも含むため、純粋なペイロードはさらに小さくなる) では不足してしまう。

5.4.6 EtherCAT、アクティブ光ファイバーリンクブリッジの構成

EtherCAT Slave Controller (ESC) として、ここでは Beckhoff 社製の ESC 基板 (FB1111-0140)[4]を使用した。この ESC 基板はインターフェース基板にマウントされ、16bit バスを介して FPGA-SoC と接続されている。ESC 基板は 8kB サイズの Dual-Port RAM を内蔵しており、この RAM 空間をバッファまたはメールボックスとして利用することが可能となっている。本システムでは RAM 空間を上下 4kB ずつに分け、上位 4kB を送信用バッファ(master to slave)、下位 4kB を受信用バッファ (slave to master) として設定している。図 5.13 の上部に ESC 基板内部のデータフローを示しており、上流から流れてくるパケットから Write コマンドデータを送信バッファに書き込み、Read コマンド部に受信用バッファ領域をコピーすることでデータ転送が行われる。

実際の EtherCAT とアクティブ光ファイバーリンク間のブリッジ処理は FPGA-SoC 側にて行われる。FPGA-SoC 内のコントローラにて ESC 基板上の送信バッファからデータを読み出し、第 4 章にて述べた Remote Procedure Call (RPC) 用プロトコルに変換を行う。RPC ブロックは ARM プロセッサを含む Hard Processor System (HPS) 上で実行される制御プログラムからもアクセスされるため、HPS からの入出力とブリッジからの入出力は MUX,DEMUX を介して行われる。EtherCAT ブリッジを介した体内通信系の評価実験については、5.6 節にて行う。

このように EtherCAT からのブリッジ接続は RPC プロトコルのみに限定している。これは、RPC プロトコルは従来のシステムで使用されてきた体内通信プロトコルを拡張したものであり、従来の上位サブシステムと互換性を維持したまま接続することが可能であるためである。また、前述の通り EtherCAT ではペイロードが増加した場合、通信性能が悪化するという問題や ESC 基板内の RAM サイズの制約の問題が現時点ではあるため、その他のプロトコルを通す実装は採用していない。分散共有メモリプロトコル等も上位サブシステムとブリッジする実装は、ESC のスペックの見直しや EtherCAT マスター側のデバイスドライバの改善や PC 側インターフェースプログラムの修正で実現は可能であるため、今後引き続き開発を進める予定である。

[4]www.beckhoff.com/english.asp?ethercat/fb1111_fb1122_fb1130.htm

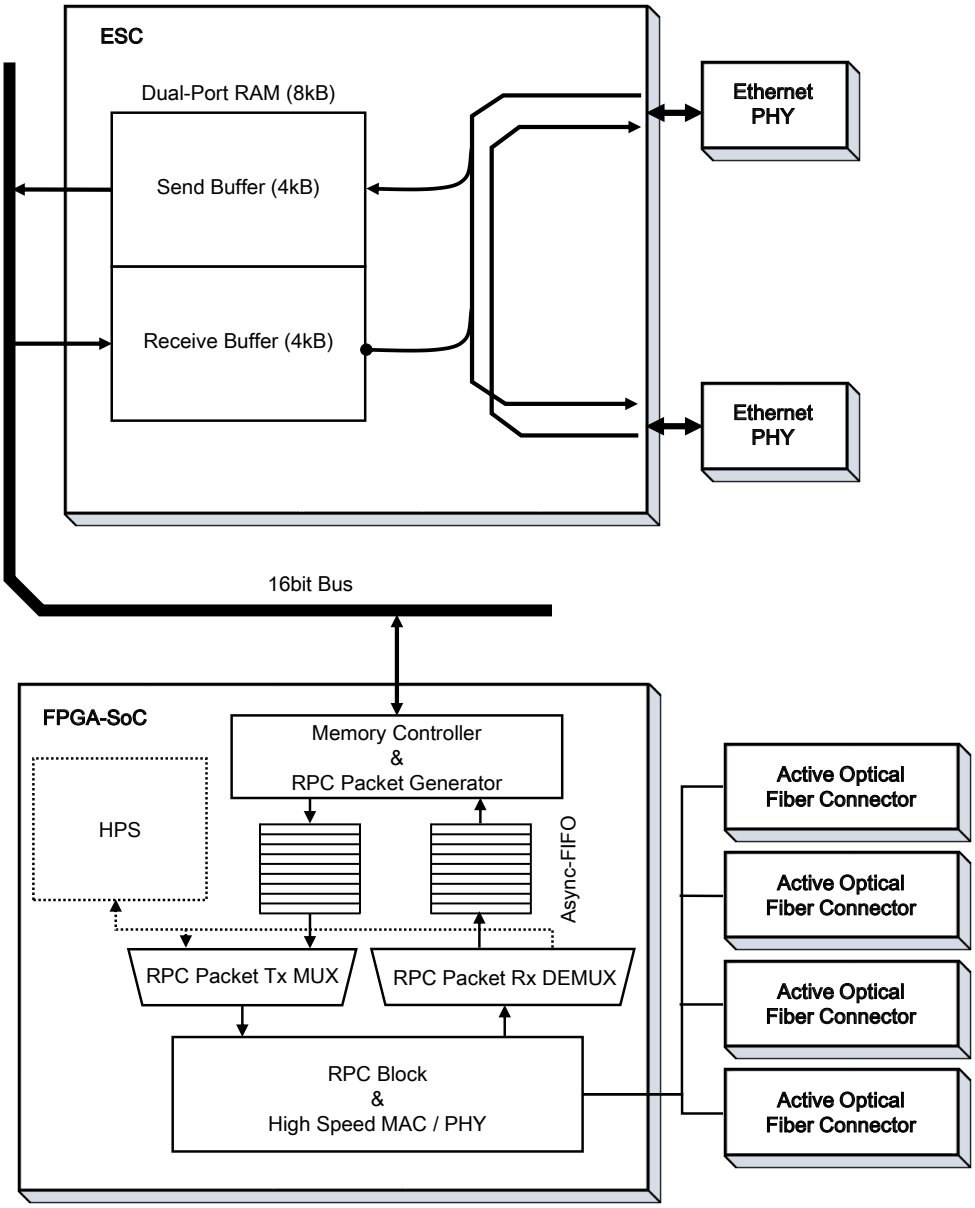


図 5.13: Data flow chart inside EtherCAT from/to active optical fiber link bridge.

5.4.7 データロガー機能

ハードウェアやモータドライバ制御レベルで異常が発生した場合、上位システムで記録されたデータログを参照してデバッグを行うのが一般的である。しかし、電圧や温度といった、フィードバック制御とは直接関係しないデータに関してはサンプリング周期を落としていたり、下位システム内の制御プロセス上のデータはそもそも上位システムには送信を行っていないため、上位システム側のデータロガーでは異常の正確な原因を特定しきれない場合が存在した。

中間層制御システムでは、分散共有メモリや RPC プロトコルによる通信を介してモータドライバ上の制御データを逐次参照することが可能なため、データロガー機能を実装することでこれらのデータを記録しておくことが可能となっている。また、中間層制御システムは 5kHz の制御周期で実行されているため、ロギングされる周期も 5kHz まで上げることが可能である。そのため、電圧の異常低下といった瞬間的な現象についてもログから追跡することが可能となり、複雑なロボット制御システムにおけるデバッグの一助となると考えられる。

ただし、組み込みシステム上での実装なため、メモリサイズやストレージサイズ、ストレージ転送速度が極めて限定的である。従って、取得可能なデータサイズは高々 100MB 程度とするのが現実的であり、これ

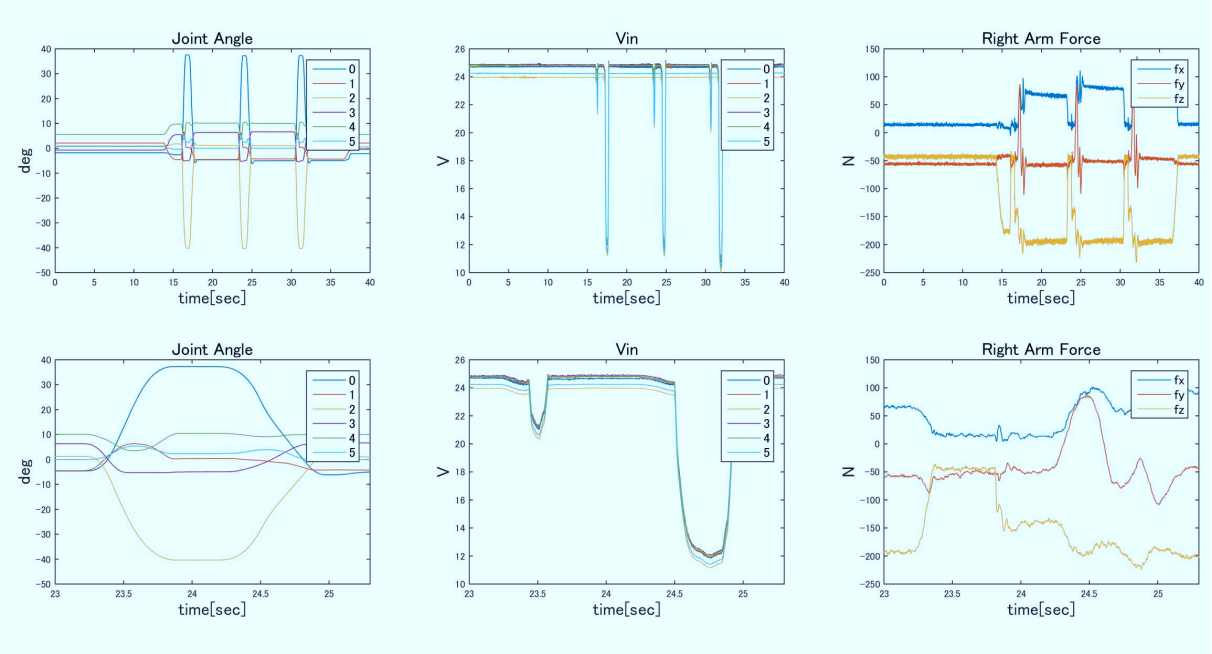


図 5.14: Data logger output sample on a middle layer control system. Figures on an upper row show full-time range of datalogger. Lower rows show the enlarged plots between 23[sec] to 25.5[sec].

は記録するデータの種類にもよるが数十秒程度の時間でしかない。そのため、一連の行動を全て記録することは現実的ではなく、異常が起きた際に記録をストレージに吐き出し、トレースできるようにするブラックボックス的な利用方法が主となる。本研究では、データロガーの記録時間は 40 秒分 (5kHz で 200,000 サンプル点) を記録する構成とした。サーボ制御系で用いる主な変数、計測値について記録したときおよそ 100MByte 程度のファイルサイズとなる。

図 5.14 は、実際にシステムで記録したログデータのうち関節角度、モータドライバ入力電圧、6 軸力センサ値の 3 種についてプロットしたものである。入力電圧やセンサデータについて、40 秒間ではあるが高精細に記録されていることが確認できる。

5.4.8 上位システムバックアップ

中間層制御システムは、上位システム用ハードウェアと比較して演算性能では大幅に劣るプロセッサを採用する対価として、ハードウェアとして物理的な振動や衝撃、電源ノイズに対して頑健な組込み用デバイスを選定することで上位システムがダウンした場合でもロボットを安全に停止させるための最低限の制御を行うことを目指している。汎用計算機では、メモリやインターフェースカード、CPU はスロットに差し込む形式になっている場合が多く、ロボットで使用するには耐振動性能や耐衝撃性能が劣る。現に実機において、強い衝撃が加わったり、あるいは転倒するなどで制御用 PC が完全にダウンしてしまう事例は多く経験されてきた。中間層制御システムはそうした最悪の場合においても、健全な状態を保ち、ロボットの最低限の制御を行うことで安全に停止まで持っていくことを実現するために必要となる。

5.4.9 従来型システムとの比較

図 5.15 に、図 5.7 にて挙げた従来型のシステムで見られた問題点と、それらに対応する中間層制御システム並びに低遅延モジュール間通信系を導入したことによる効果を表形式でまとめている。

表中に示している通り、特に致命的であった信頼性関係の問題について、通信系の改善と中間層制御システムの導入によって改善されていると考えられる。また、機能面についても性能の向上が図られており、従来では困難であった実時間処理性能も実現可能である。一方で、開発面に関してはシステムの複雑化によって難度が上がっている他、スペック面での向上のために開発コストが上がっているという点は否めない。た

カテゴリ	従来システム問題点	中間層制御システム・モジュール間低遅延通信系の導入による効果
致命的	障害時は完全停止	○ 上位層と中間層でシステムが冗長化されることで、障害時に最低限の制御を行うことが可能となる。
	絶縁不良による信号ノイズ	○ 体内通信系は光ファイバ化されることで耐ノイズ性能は大幅に向上。
	振動・衝撃でシステムダウン	○ カードスロット型インターフェースを排除することで耐振動、耐衝撃性能を向上。 ○ 耐衝撃性の高いSMTパッケージ品を選定可能。
	ケーブル・コネクタの劣化、摺動による接触不良	○ 屈曲性能の高く細いファイバ化によってケーブル劣化を低減。 ○ EtherCATは信頼性の高いRJ-45コネクタ。
	放熱不良によるCPU、インバータのオーバーヒート	○ ヒートシンクレスの自然放熱での運用が可能な組込み用プロセッサによって放熱不良によるシステムダウンのリスクを低減。 □ 上位システムについては放熱性能は関係しない。
機能制約	デバイスインターフェースの空間占有体積が大きい	○ 独自デバイスインターフェースが不要になり、拡張スロットレスな小型パッケージ品を上位システムに採用可能。 × 中間制御層用デバイスが別途必要となるが、従来よりレイアウト自由度は高い。
	ケーブルによる空間占有体積が大きい	○ 従来配線の 8 分の 1 の太さのファイバー化、及びセンサデバイス用ケーブルの省配線化で空間占有体積を大幅低減。
	使用デバイスに対応したマザーボードから選定	○ 汎用Ethernetポートが使用できればよく、拡張用インターフェースの制約を排除。パッケージサイズやプロセッサ性能を重視した選定を可能にした。
	下位システムへの限定的なデータアクセス性	○ 実時間通信用に産業用規格であるEtherCATを使用し、さらに非実時間通信用途にEthernet経由も使用可能としたことで、下位システム上のデータへのアクセス性を向上。
開発効率	独自デバイスの開発・製造	× 中間層システム用デバイスを独自に開発する場合は、独自デバイスインターフェースより開発難度・コストが高い。
	独自デバイスのドライバの開発・保守	○ 上位システムでは汎用のEthernetドライバで良いため、システムアップデートの影響を受けにくい。 × EtherCATのパフォーマンスを要求する場合には専用ドライバの導入が望ましい。 × 中間層システムでは光リンク用のドライバが必要。
	制御システムの更新でデバイスドライバ、インターフェースに不具合	○ 実時間処理系、デバイス制御系が上位システムから切り離されるため、上位システムのアップデートが容易になる。 × 中間層システム用デバイスが独自の場合、ハードウェアのアップデートを行うには新規に開発が必要。
	デバイスごとにそれぞれインターフェース・配線が必要	○ 中間層のデバイスや下位層のデバイスで接続可能。
	インターフェース処理と制御演算処理が同一マシン上	○ 演算負荷は比較的小さいが、実時間性能の要求の大きい処理を中間層システムに分離可能。 ○ 低遅延な実時間通信で接続されているため、制御は容易。 × 中間層システム側のプログラムはハードウェア資源に制約
	障害発生時の詳細なログが記録されない	○ 中間層システムでブラックボックス型ロガーを実行可能。

図 5.15: Effects of the middle layer control system and low-latency communication link.

だし、通信系に分散共有メモリプロトコルのような使い勝手の良いプロトコルの採用や、上位と中位のシステム間通信に EtherCAT と Ethernet で二重化することで従来のロボット体内通信系の開発よりも格段に容易化していると考える。

5.5 中間層制御システム構成モジュール

中間層制御システムではロボット体内システムで必要となる各種の低レイヤな処理を、OpenRTM や ROS を利用したモジュール群として構成する。これは上位システムの構成と類似しており、モジュール化することで複雑なデータフローが生じる制御システムにおいても腕用システムや脚用システム間でパラメータの変更のみで勘弁に切替を可能とし開発効率を向上させている。

また、計算機資源の限られた組込みシステム上での実行となるため、実行コンテキストを処理ごとに細かく分割することで、スレッド優先度、実行周期・実行タイミングといったコンテキストパラメータを処理の負荷に応じて設定できるという利点がある。例えば、実関節角度に依存するダイナミクス系の計算については、力・トルクに依存する物と比較して時定数が大きいと考えられる。そのため、これらのダイナミクス系計算の実行コンテキストを分割することで優先度・実行周期を負荷が小さくなるようシステムに応じた最適化を行うことが可能となる。従来は実行コンテキストの細分化は、プロセス・スレッド間でのデータ通信の複雑化を招いたが、モジュール群間の通信 API を提供するミドルウェアの利用によって複雑度が大幅に低減され、開発が容易となった。

本研究における中間層制御システムでのコアな処理は、ロボットの関節制御則へのトルク制御則の適用となるが、そのためには基本的な運動学計算は不可欠となる。

5.5.1 サーボ制御モジュール

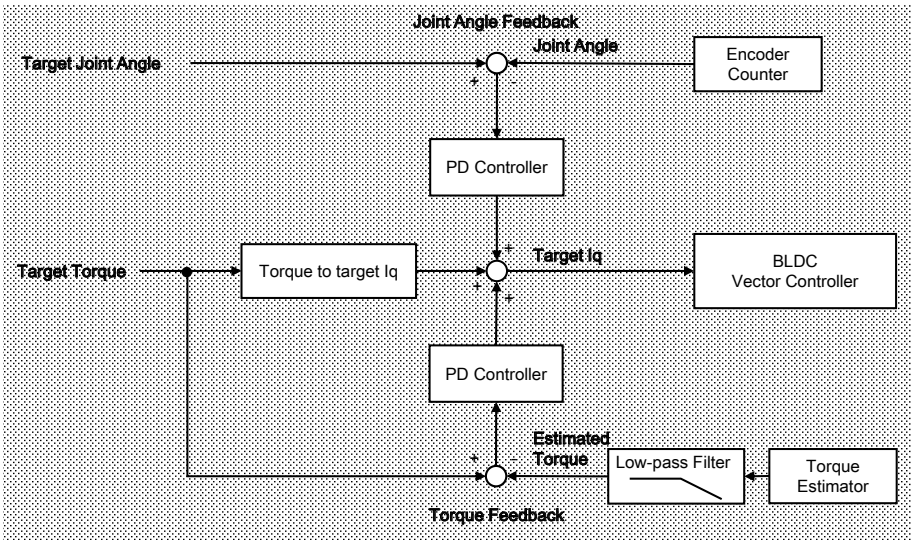
ロボットの各関節を駆動するモータそれぞれのサーボ制御を行う。従来は各分散制御基板内にて電流制御、位置制御、トルク制御を行っていたが、アクティブ光ファイバーリンクによって中間制御層に各ノードのデータを各制御サイクルごとに実時間で収集可能なため、中間制御層で集中的に管理することが可能となる。

二自由度制御

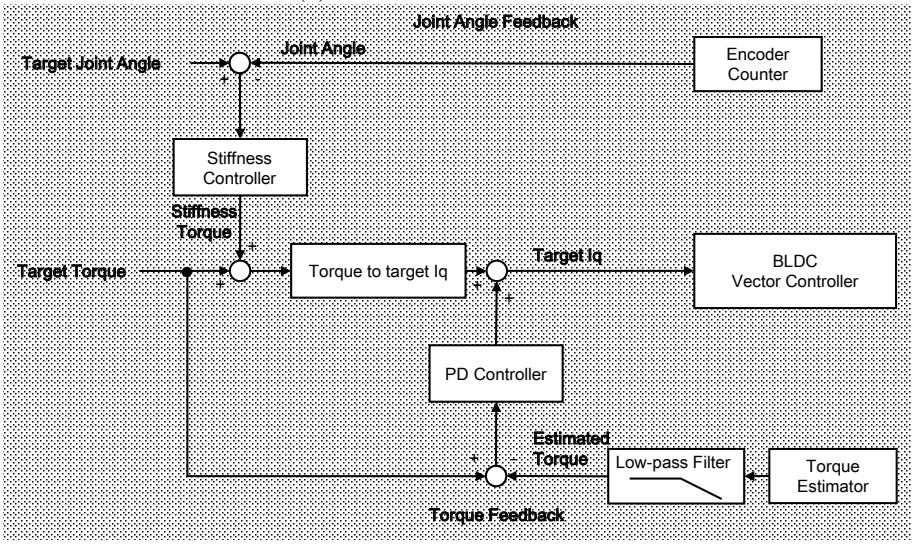
目標電流値の計算に図 5.16a に示される二自由度制御による方法を用いる。目標関節トルクに対して、モータのトルク定数を乗じることで目標電流値のフィードフォワード項は簡単に計算される。サーボ制御モジュールの出力の大部分はこの成分となる。さらに関節角度目標値への追従性の補償のために PD コントローラの出力を目標電流値に加算している。同様に目標関節トルクへの追従のために、トルク推定器からの出力との差分から PD コントローラの出力を計算したものを加算している。

関節剛性制御

前述の関節トルクと関節角度の2 入力がある二自由度制御則では、関節トルクと関節角度のどちらの入力への追従が重視されるかは、フィードバックゲインによって決定される。しかし、物理的な意味合いをもたないため、関節の柔らかさを定量的に設定することが困難である。ここで、関節の柔らかさを表現するパラメータとして関節剛性を考える。目標関節角度を $\boldsymbol{\theta}^{ref}$ とし、エンコーダによって計測される実関節角度を $\boldsymbol{\theta}^{act}$ とすると、関節角度誤差は $\Delta\boldsymbol{\theta} = \boldsymbol{\theta}^{ref} - \boldsymbol{\theta}^{act}$ と表現される。関節剛性パラメータ $\boldsymbol{K}$[Nm/rad] によって、関節角度誤差 $\Delta\boldsymbol{\theta}$[rad] と関節トルク $\boldsymbol{\tau}$[Nm] の間に $\boldsymbol{\tau} = \boldsymbol{K}\Delta\boldsymbol{\theta}$ という関係式を与えることができる。この関係式によって関節トルクと関節角度の追従性を調整する。さらに、発振抑止のために関節粘性パラメータ $\boldsymbol{D}$[Nms/rad] によって粘性項を加え、$\boldsymbol{\tau} = \boldsymbol{K}\Delta\boldsymbol{\theta} + \boldsymbol{D}\Delta\dot{\boldsymbol{\theta}}$ として図 5.16b に示すように目標関節トルクに加えることで、トルクベース制御において関節角度フィードバックループを構成する。この構成によって、関節軸にトルクセンサを新たに搭載する必要が生じないという利点を与えている。なお推定トルク値フィードバックゲインは、フィードフォワードによる出力トルクが目標値に近くなる点や推定トルク値にノイズ・外れ値が含まれる点を踏まえて低めにパラメータ調整されている。そのため、完全なトルクセンサを利用したトルク制御と比較して精度に劣るが、後述のアクチュエータモデル補償によってその欠点を補う工夫を組み込んでいる。



(a) Two DoF servo controller.



(b) Joint stiffness controller.

図 5.16: Joint servo controller schematics.

5.5.2 一般化慣性行列計算モジュール

関節自由度 N のロボットについて、関節角度 $\boldsymbol{\theta} \in \mathbb{R}^N$ と関節トルク $\boldsymbol{\tau} \in \mathbb{R}^N$ の動力学的対応関係は一般に次式で表現される [70]。

$$\boldsymbol{\tau} = \boldsymbol{A}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{B}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{C}(\boldsymbol{\theta}) + \boldsymbol{N} \tag{5.1}$$

右辺初項は慣性力であり、第二項はコリオリ力・遠心力、第三項はポテンシャル力を示す。第四項 $\boldsymbol{N}$ は関節にかかるトルクに換算した外力である。目標関節角度 $\boldsymbol{\theta}^{\boldsymbol{ref}}$ を与えられた時に、目標を実現する関節トルク $\boldsymbol{\tau}^{\boldsymbol{ref}}$ についても式 5.1 で与えられるが、実際にはロボットモデルが正確に実機の動特性を表現できているとは限らないことや完全な外力の測定が困難なため、各パラメータについて推定値を用いる必要がある (式 5.2)。

$$\boldsymbol{\tau}^{\boldsymbol{ref}} = \hat{\boldsymbol{A}}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}^{\boldsymbol{ref}} + \hat{\boldsymbol{B}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}^{\boldsymbol{ref}} + \hat{\boldsymbol{C}}(\boldsymbol{\theta}) + \hat{\boldsymbol{N}} \tag{5.2}$$

各推定値はロボットモデルの公称値から計算されるものを使用する。このように式 5.2 に基づいて参照トルク値を与える手法は計算トルク法として一般に知られている。

本モジュールでは慣性力を補償するための推定パラメータ $\hat{\boldsymbol{A}}(\boldsymbol{\theta})$ の計算を行う。現在フレーム k におけるロボットの実関節角度 $\boldsymbol{\theta}_k$ を元にロボットの各慣性行列を一般化座標に対応させた慣性行列 $\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)$ の更新を行う (本論文では $\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)$ を一般化慣性行列と呼ぶことにする)。

ルートリンクからたどって第 i 番目のリンクについて、その質量を m_i、慣性テンソルを $\boldsymbol{I}_i \in \mathbb{R}^{6\times6}$ としたとき、$\boldsymbol{M}_i$ を以下のように定義する。

$$\boldsymbol{M}_i := \begin{bmatrix} m_i\boldsymbol{E} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_i \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{5.3}$$

$\boldsymbol{E} \in \mathbb{R}^{3\times3}$ は単位行列である。この $\boldsymbol{M}_i$ は各ロボットについて固有の定数となるため、周期実行時には演算負荷とならない。

第 i 番目のリンクの重心運動について $i+1$ 以降の関節運動に依存しないため、各リンク重心についての Basic Jacobian $\boldsymbol{J}^i$ は以下のように $i+1$ 番目以降の列が $\boldsymbol{O}$ で置換された形式として計算される。

$$\boldsymbol{J}^i = \begin{bmatrix} \boldsymbol{J}_0^i & \boldsymbol{J}_1^i & \ldots & \boldsymbol{J}_i^i & \boldsymbol{O} & \ldots & \boldsymbol{O} \end{bmatrix} \in R^{6\times N} \tag{5.4}$$

$$\boldsymbol{J}_j^i = \begin{bmatrix} \boldsymbol{\omega}_j \times \boldsymbol{r}_j \\ \boldsymbol{\omega}_j \end{bmatrix} \tag{5.5}$$

$$\boldsymbol{\omega}_j := \boldsymbol{R}_j\boldsymbol{a}_j \tag{5.6}$$

$$\boldsymbol{r}_j := \boldsymbol{p}_i^{wc} - \boldsymbol{p}_j \tag{5.7}$$

ここで $\boldsymbol{R}$,$\boldsymbol{a}$,$\boldsymbol{p}^{wc}$,$\boldsymbol{p}$ はそれぞれ、リンクの姿勢行列、関節回転軸ベクトル、リンク重心位置、関節中心位置であり下付き添え字 j は第 j 番目のリンクについてであることを示す。$\boldsymbol{p}_j$ と $\boldsymbol{\omega}_j$ はエンドエフェクタについてのフォワードキネマティクス計算でも同様に必要となるため、結果を再利用することでリンク重心に関する Basic Jacobian は高速に計算することが可能である。

$$\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k) = \sum_{i=1}^{N} \boldsymbol{J^i}^{\mathrm{T}}(\boldsymbol{\theta}_k)\boldsymbol{M}_i\boldsymbol{J}^i(\boldsymbol{\theta}_k) \tag{5.8}$$

一般化された慣性行列 $\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)$ は式 5.8 にて計算される。この計算量は $O(n^2)$ であるが、一般的なロボットのマニピュレータや脚の場合、$N = 6-8$ 程度であるため、総計算量としては身体制御において実時間実行を行うのに十分軽量である。ARM Cortex-A9 Dual Core (800MHz) のシステム上で実行した6 自由度腕の一般化慣性行列計算スレッドは実時間実行処理も含めて一サイクル当たり平均 36μsec と十分短い時間で実行された (表 5.2)。

5.5.3 参照トルク生成モジュール

このモジュールでは 5.5.2 節の式 5.1 にて示した、関節運動と関節トルクの動力学関係式から得られる関節トルクをトルクサーボ制御器への参照トルク入力として生成する。また、2.6.3 節にて述べたアクチュエータのダイナミクスにも注目し、

- アクチュエータ慣性力
- ハーモニックドライブ粘性摩擦

についてもモデル化を行い、補償トルクを計算して参照トルク値に重畳させることで、制御誤差の低減を行う。

重力補償トルク

式 5.1 のポテンシャル項について、重力を補償する関節トルクの計算を行うことでゼロトルク状態で腕マニピュレータや脚を支持するための参照トルク値を生成する。第 i 番目のリンクについて、関節位置からみたリンク重心位置 $\boldsymbol{p}_i^{wc}$、リンク重心位置のローカル座標系位置 $\boldsymbol{p}_i^{lc}$、リンク姿勢行列 $\boldsymbol{R}_i$ について、

$$\boldsymbol{p}_i^{wc} = \boldsymbol{R}_i\boldsymbol{p}_i^{lc} \tag{5.9}$$

であり、重心に働く重力 $\boldsymbol{F_g}$、重力による関節位置周りのモーメント $\boldsymbol{N_g}$ は

$$\boldsymbol{F_g} = m_i \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{5.10}$$

$$\boldsymbol{N_g} = \boldsymbol{p}^{wc} \times \boldsymbol{F_g} \tag{5.11}$$

となる。第 i 番目の関節軸まわりのトルク τ_i は、関節軸ベクトル $\boldsymbol{a}_i$ との内積演算より、

$$\tau_i^g = \boldsymbol{N_g} \cdot (\boldsymbol{R}_i \boldsymbol{a}_i) \tag{5.12}$$

となる。実際には、リンク $i+1$ 以降から伝達される力とトルクの入力も関節軸まわりトルクの計算では考慮する必要がある。

慣性力補償トルク

式 5.1 の慣性力項について、5.5.2 節にて計算される一般化慣性行列 $\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)$ を用いて、

$$\boldsymbol{\tau}_k^{iner} = \hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)\ddot{\boldsymbol{\theta}}_k^{ref} \tag{5.13}$$

と計算される慣性力補償トルク $\boldsymbol{\tau}_k^{iner}$ を参照トルクに加算する。

エンドエフェクタ目標力発生トルク

上位システムからコマンドとして与えられるエンドエフェクタで発生させる目標力 $\boldsymbol{Q^{end}} = [\boldsymbol{F^{end}}^{\mathrm{T}}、\boldsymbol{N^{end}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^6$ について、エンドエフェクタについての Basic Jacobian $\boldsymbol{J_b^{end}}$ を用いて、必要な関節トルク $\boldsymbol{\tau^f} \in \mathbb{R}^6$ は以下のように計算される。

$$\boldsymbol{\tau^f} = (\boldsymbol{J_b^{end}})^{\mathrm{T}} \boldsymbol{Q^{end}} \tag{5.14}$$

関節角度上限ポテンシャル障壁トルク

トルク制御時に関節角度上限を超える方向へ指令値が出力されないようにするために、関節角度上限付近に仮想的にポテンシャル障壁を設定する (図 5.17)。各関節 i について、計測された実関節角度 θ_i、関節角度上限・下限 θ_{max}、θ_{min} としてその近傍 d 以内に設定される線形な仮想ポテンシャル障壁は以下の式で計算される。

$$\tau_i^{lim} = \begin{cases} -\frac{\tau_i^{max}}{d}(\theta_i - (\theta_{max} - d)) & \text{if } (\theta_i - (\theta_{max} - d)) > 0 \\ +\frac{\tau_i^{max}}{d}((\theta_{min} + d) - \theta_i) & \text{if } ((\theta_{min} + d) - \theta_i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

この式では、関節トルク最大値 τ_i^{max} が、関節角度限界での出力トルクとなるようにポテンシャル勾配を設定している。

アクチュエータ慣性力補償トルク

減速比 $N:1$ のロボット関節について、関節角度 θ に対応するアクチュエータの等価慣性モーメント I_{equ} は、

$$I_{equ} = I_r N^2 \tag{5.16}$$

と表され、アクチュエータの慣性力を補償するための目標関節トルク τ^{mi} は

$$\tau^{mi} = I_{equ}\ddot{\theta}^{ref} \tag{5.17}$$

となる。慣性モーメントの値は製造元のデータシート値が入手可能な場合にはそれを、3D-CAD モデルが利用可能な場合には CAD 上での計算値を使用する。本研究では関節軸からアクチュエータ回転軸までを繋ぐ以下の駆動系部品の慣性力についてモデルに含めている。

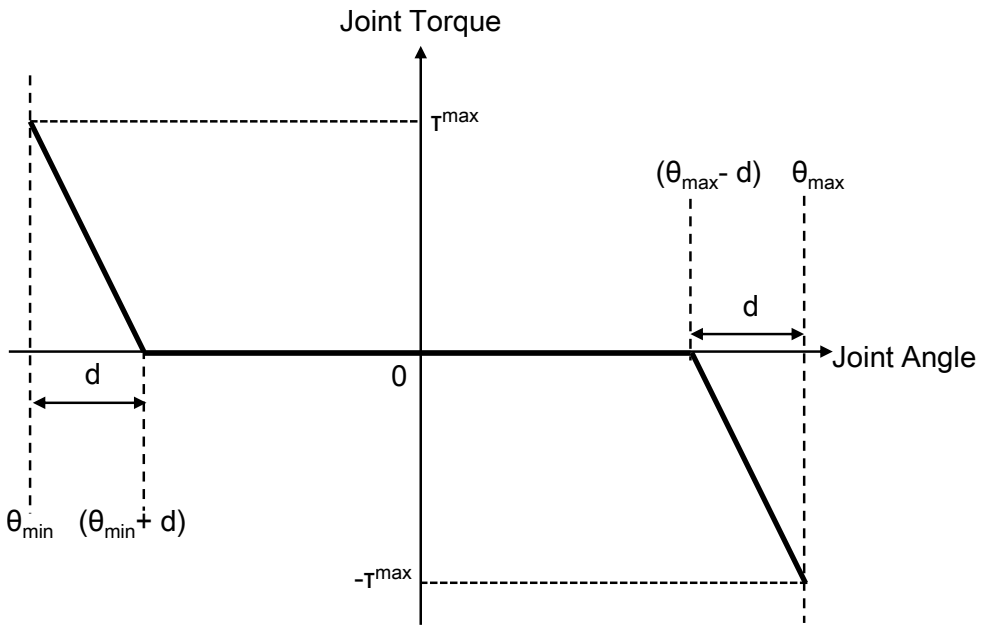


図 5.17: Potential barrier of the joint angle limit.

- モータロータ
- 小プーリ
- 大プーリ
- プーリシャフト
- ハーモニックドライブロータ

本研究で用いる A0-B spec で採用している減速比はハーモニックドライブ 160 : 1、プーリによる減速 72 : 25 で総減速比 460.8 : 1 となっている。モータ回転軸を構成するモータロータと小プーリーは総減速比、ハーモニックドライブ回転入力軸を構成する大プーリ・プーリシャフト・ハーモニックドライブロータはハーモニックドライブ減速比を適用することでアクチュエータの等価慣性モーメント I_{equ} としている。A0-B spec に使用している関節において、この値は 1.29[kgm^2] となる。例えば速めのスイング動作（ 600msec で関節角度変化 20° 程度の動作）時の最大関節加速度はおよそ 8[rad/sec] 程度となるが、この時アクチュエータの慣性力を補償するトルク値はおよそ 10[Nm] となり、無視できない大きさとなっていることがわかる。

ハーモニックドライブ粘性摩擦補償トルク

ハーモニックドライブの粘性摩擦抵抗を参照トルクの生成時に考慮する。粘性摩擦係数は公称値がデータシート等から得られることは少なく、個体間のバラつき・使用状況による変動の大きいパラメータとなっている。これは、

- 組付け誤差、応力歪・熱歪による摩擦
- グリス粘性
- フレクススプライン等のヒステリシスロス

等の管理が著しく困難な複数の要素が要因としてある。データシートではこれらをまとめて伝達効率として係数が提示されているが、負荷トルクによって変動する非線形パラメータとなっており、また個体間のバラつきについては対応ができない。

本研究では、C.3 節のトルク推定モジュールにて使用している粘性摩擦係数同定値 $\hat{\eta}$ を公称値とし、線形な粘性摩擦モデルを仮定して参照トルク値の計算に用いている。

$$\tau^{mv} = \hat{\eta}\dot{\theta}^{ref} \tag{5.18}$$

将来的にはオンラインでのパラメータ同定処理を組み合わせることで、精度を向上させることができると考える。

ハーモニックドライブクーロン摩擦補償トルク

ハーモニックドライブの起動トルクをクーロン摩擦でモデル化し、補償トルクとする。起動トルク τ^{start} はデータシートから得られる。

$$\tau^{mc} = \mathrm{sign}(\dot{\theta}^{ref})\tau^{start} \tag{5.19}$$

なお、$\mathrm{sign}(x)$ は入力の符号を返す関数である。

参照関節トルク

以上を元に、参照関節トルク $\boldsymbol{\tau}^{ref}$ を次式で与える。

$$\boldsymbol{\tau}^{ref} = \boldsymbol{\tau}^{g} + \boldsymbol{\tau}^{iner} + \boldsymbol{\tau}^{f} + \boldsymbol{\tau}^{lim} + \boldsymbol{\tau}^{mi} + \boldsymbol{\tau}^{mv} + \boldsymbol{\tau}^{mc} \tag{5.20}$$

また、動力学系の計算での使用のために各制御サイクルにおけるアクチュエータモデル補償トルク $\boldsymbol{\tau}_k^{motor}$ と、正味出力関節トルク $\boldsymbol{\tau}_k^{net}$ を

$$\boldsymbol{\tau}_k^{motor} := \boldsymbol{\tau}_k^{mi} + \boldsymbol{\tau}_k^{mv} + \boldsymbol{\tau}_k^{mc} \tag{5.21}$$

$$\boldsymbol{\tau}_k^{net} := \boldsymbol{\tau}_k^{ref} - \boldsymbol{\tau}_k^{motor} \tag{5.22}$$

として保存しておく。

5.5.4 制御モードセレクタモジュール

サーボ制御モードの切り替えを行うモジュールになっている。上位サブシステムからの制御モード入力は、オン・オフのイベント入力として受信される。オンイベントに対して、実際にサーボ制御を実行状態にするか判断を中間制御層側で行うためにモジュールが間に挟まる形となっている。サーボ制御がオン状態でどの制御則を適用するかについて、このモジュールでモードセレクターとして働かせる。モードセレクターの入力は、モジュールへの RPC 通信(サービスコール) で切替を行う。制御モード切替は、一連のタスク実行の中で頻繁に起こらないと考えられるので非実時間実行で十分と考えられる。

5.5.5 衝突検知モジュール

Luca らは一般化運動量に着目したロボットアームの衝突検知・判定を提案している [71, 72]。

一般化運動量 $\boldsymbol{P_g} \in \mathbb{R}^N$ は

$$\boldsymbol{P_g} = \boldsymbol{A}(\theta)\dot{\theta} \tag{5.23}$$

で定義される。

また関節動作による運動エネルギー $E_k \in \mathbb{R}$ は

$$E_k = \frac{1}{2}\dot{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{5.24}$$

Luca らは衝突検知のために以下のような量を与えている ([72] の式 11)。

$$\sigma(t)^{col} = k_D\left[E(t) - E(0) - \int_0^t (\dot{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{\tau} + \sigma^{col})ds\right] \tag{5.25}$$

式 5.25 を $t = k * dT$ で離散化して表現して、

$$\sigma_{k+1}^{col} = k_D \left[E_k - E_0 - \sum_{i=0}^{k} (\dot{\boldsymbol{\theta}}_i^{\mathrm{T}} \boldsymbol{\tau}_i + \sigma_i^{col}) dT \right] \tag{5.26}$$

を使用する。式 5.25 はロボットが持つ力学的エネルギーと入力トルクから推定される入力エネルギーの差分をとることで、外力によるエネルギーを推定していると考えられる。ここに減衰項を加えることで誤差の蓄積を防いでいると考えられる。

5.5.6 モジュール群の実行時間計測

主要なモジュール及び関数の実行にかかる時間を計測した。計測は6 自由度腕型ロボットの A0-B spec で、6 つの RMTP-02D 基板を接続した状態で行った。計測時それぞれのスレッドが並列に実行されており、またモニタリング用のプログラムが別途実行されている。それぞれの計測値は、周期実行されている各スレッドが起床してから、スリープに入るまでの実時間を計測した値となる。結果を表 5.2 に示す。実行時間の大きい処理としては、ヤコビアン・ SR インバースの演算が 80μsec と大きく、続いてサーボスレッドにおける受信データ更新が 70μsec と処理に時間がかかっている。ヤコビアンは演算量が多いためであり、受信データ更新については分散共有メモリからローカルメモリへのコピーがボトルネックとなっているためである。これは現在の中間層のハードウェアを構成する評価キットへの実装上、プロセッサと FPGA 上メモリ間のバスアクセスの最適化やデバイスドライバ側での Direct Memory Access Controller (DMAC) の処理の実装が不十分であるためであり、大幅な向上の余地は残っている。その他の処理については、最短の実行周期であるサーボスレッドの 200μsec(5000Hz) よりも大幅に小さく、十分に実時間で処理できる内容であることがわかる。脚型ロボットや、6 自由度以上のロボットになるとこの限りではないが、マルチノード構成に拡張が可能なため、例えば右脚用中間層ノード、左脚用中間層ノードといった具合に分担することで、システム規模が拡大しても実時間性を維持することが可能となっている。

表 5.2: Execution time of principal functinos on 6 DoF System.

Function	Execution Time [μsec]	Thread
SR Inverse	80	Torque Reference Generator
Servo Control (Total)	95-100	Servo Controller
(Receive New Values)	70-74	
(Update Robot State)	12-25	
(Servo Control)	2	
(Send New Servo Command)	9-10	
(Push Data Logger)	5-6	
Mode Select	23	Control Mode Selector
Mass Matrix	36	Mass Matrix Updater
Thread Statistics	2-3	Each thread

5.6 サブシステム間データ通信系の実時間性検証

上位システムと下位システムの間を中間層制御システムが介在する構成となっているが、データ通信に要する実時間性について検証を行った。上位システムで生成した目標関節角度値のデータ通信について EtherCAT を介した経路と、汎用の Ethernet を介して CORBA をベースとした openRTM データポート通信、及び ROS の TCPROS によるトピック通信経由の経路の3 種類について比較を行った。図 5.18 に本実験での各データ通信経路の概略を示す。上位制御システムは、従来システムの hrpsys-base と実時間インターフェー

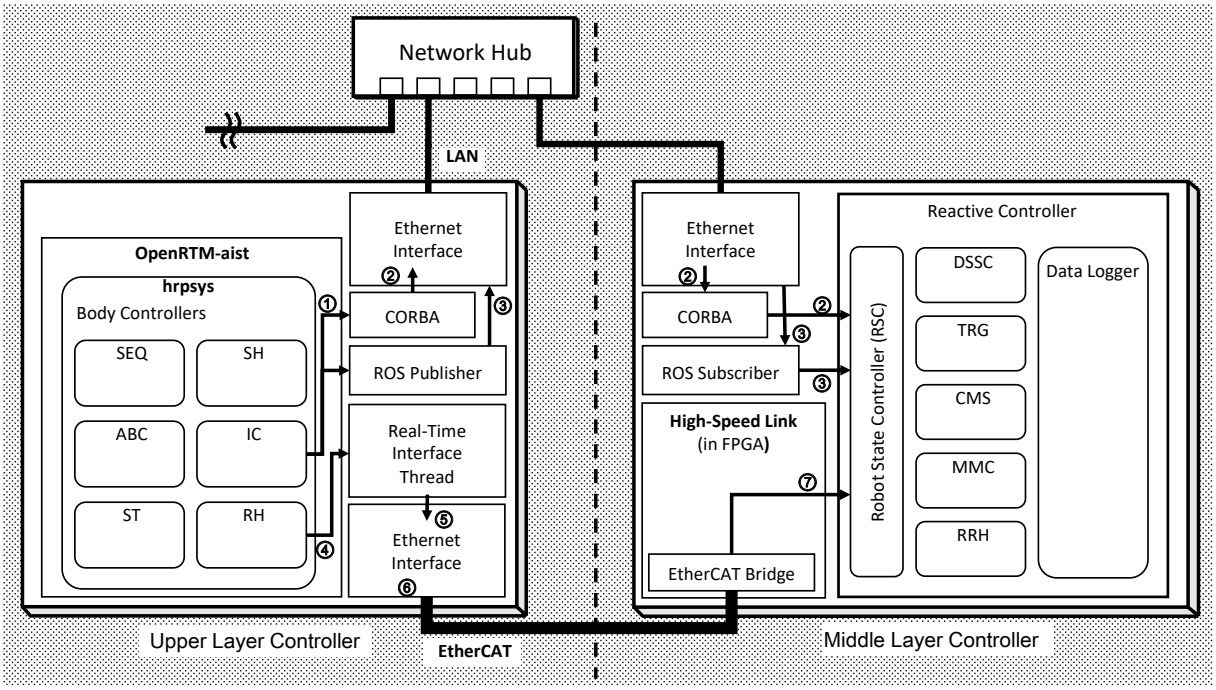


図 5.18: Dataflow in the evaluation of real-timeness of the reference joint angle transmission via three Connections.

スプログラムによる構成となっている。この3 種の経路それぞれで転送されてきた目標関節角度指令値は中間制御層システム上に展開された Robot State Controller (RSC) にて非同期にバッファされ、サーボ制御スレッド (DSSC) に同期して実行されるデータログ処理によって、毎フレーム各値を保存する。上位システム側の hrpsys の実行周期が 500Hz(2msec 周期) なのに対して、中間制御層側実行周期は 5000Hz(0.2msec 周期) と十倍の粒度を実現しているため、受信データの更新エッジをキャッチすることで各通信経路のジッタを計測することが可能となっている。

5.6.1 OpenRTM データポート

Ethernet を介してデータ送信を行う OpenRTM を使用したプロセスについて、それぞれ目標関節角度指令値は hrpsys 内の Impedance Controller (IC) から出力される RTC データアウトポートを利用する (図 5.18 の①)。中間層制御システム側に実装した OpenRTM コンポーネントでのデータ受信は、CORBA を利用した RTC データポート同士の接続 (図 5.18 の経路②) によって最終的に TCP による Ethernet 通信として実行される。

5.6.2 ROSトピック通信 (TCPROS)

OpenRTM データポートと同様に、ROS をミドルウェアとして使用することで Ethernet を介した TCPROS によるトピック通信を実装し検証を行う。目標関節角度指令値は OpenRTM データポートの場合と同様、IC から出力される RTC データアウトポートのデータを一旦 OpenRTM データインポートで取得する (図 5.18 の①)。即座に ROS トピック通信用のメッセージ型に変換し、TCPROS 方式による publish を行う (図 5.18 の③)。この処理は実時間周期実行プロセスで行われ、データインポートのイベントをポーリングしてパブリッシャーの実行が行われる。中間制御層システム側では ROS トピックサブスクライバーがサーボ制御スレッドに同期してトピックの受信を行っており、受信されたデータは RSC にバッファされる。

5.6.3 EtherCAT

EtherCAT を介したデータ通信では、hrpsys 上の Robot Hardware Component(RH) から目標関節角度指令値が共有メモリ (図 5.18 の④) を介して実時間インターフェースプログラムに渡される。そこでアクティブ光ファイバーリンクの RPC プロトコル用パケットに変換され、さらに EtherCAT パケットにパッキングされた後、EtherCAT フレームとして送信される (図 5.18 の⑤及び⑥)。中間制御層側では EtherCAT Bridge を通過後、FPGA 内部バス経由でサーボ制御スレッド上のメモリに直接マップされる (図 5.18 の⑤及び⑦)。

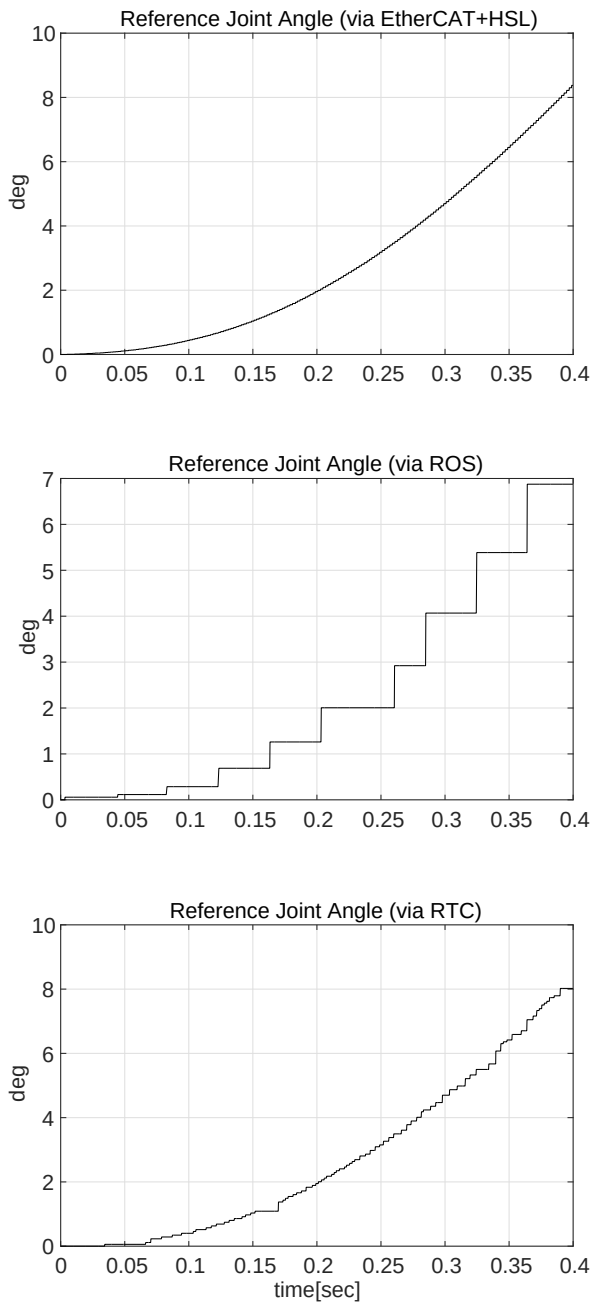
5.6.4 3 経路による目標関節角度指令値の計測結果

単調増加する適当な 6 次元の目標関節角度軌道を送信して、各通信経路の実時間性を比較する。前述の3経路について並列実行することで得られた、それぞれの経路での目標関節角度指令値軌道を図 5.19a に示す。ROSトピック通信によるデータ (図 5.19a の中段) は、他の2 経路と比較して明らかにデータ飛びが発生していることがわかる。また、RTC データポート通信 (図 5.19a の下段) についても、ところどころでデータ飛びが生じており、またグラフの軌道にノイズのように歪が生じていることがわかる。一方で EtherCAT 経由では特に歪は見られず安定した通信が行われていることがわかる。これらは、中間制御層で使用する ARM プロセッサの性能では Ethernet 通信の処理の負荷が大きく、特に CORBA での通信はプロセスの負荷による遅延によって他のプロセスにも影響していることがわかった。現に、RTC プロセスの実行によってプロセッサの使用率が 100%をオーバしてしまい、他プロセスで優先度の低めのスレッドに影響が出てしまっていた。図 5.19b に、RTC データポート通信処理を省略した場合での計測結果を示す。図 5.19a にて生じていた ROSトピック通信のデータ飛びが解消されていることが確認される。しかし、図 5.19a にて RTC データポート通信の結果に生じていたデータ遅延は相変わらず生じており、実時間性の点で Ethernet 経由のデータ通信はロボット制御において困難であることが改めて確認された。一方、EtherCAT+EtherCAT Bridge では、中間制御層側での通信処理のほとんどがハードウェア処理で実行されるため、プロセッサの使用率が著しく高くなった場合においても極めて安定な実時間通信性能を保持し続けることが確認された。

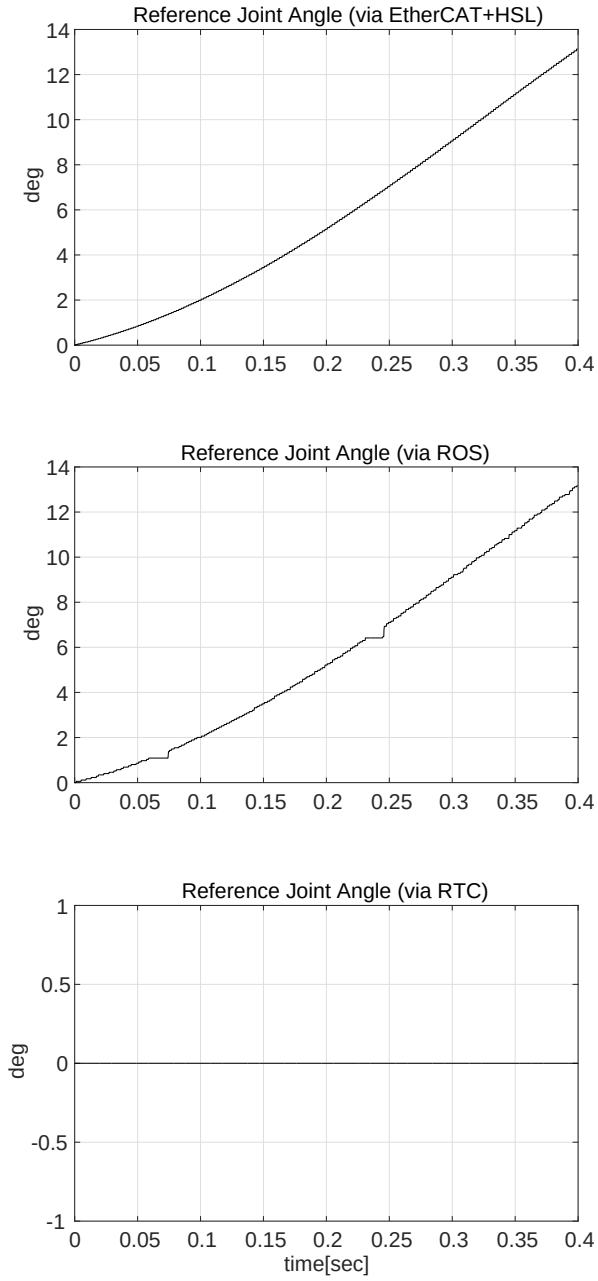
本実験の結果を評価するため、EtherCAT+HSL による通信と RTC データポート、TCPROS による通信それぞれのジッタ性能についても計測を行った。計測結果を図 5.20 と図 5.21 にそれぞれ示す。ジッタ値は目標関節角度の更新周期の期待値 500Hz(2msec) に対して、どの程度増減したかを示す値であり、ジッタ 0 なら完全に 500Hz に同期したタイミングにてデータが転送されていることを示す。ただし、基準となる中間層制御システム側のサーボ制御スレッド自体の実行周期ジッタも含むことに留意する必要がある。

図 5.20 は RTC データポートを含めた3 経路の通信によるジッタ性能を示す。RTC データポートを利用した場合、ジッタ平均値 1.401[msec]、標準偏差 2.130[msec]、最悪値 18.34[msec] となっている。また、TCPROS を使用した経路では、ジッタ平均値 4.032[msec]、標準偏差 36.25[msec]、最悪値 686.3[msec] となっている。EtherCAT+HSL による経路では、ジッタ平均値 0.1048[msec]、標準偏差 0.2297[msec]、最悪値 2.613[msec] となっている。Ethernet ベースの2 経路と比較して、EtherCAT+HSK 経路ではジッタ性能が一桁以上優れていることが確認される。

また、図 5.21 は EtherCAT+EtherCAT Bridge 通信によるジッタ性能を示す。TCPROS を使用した経路では、ジッタ平均値 0.5500[msec]、標準偏差 1.061[msec]、最悪値 13.75[msec] となっている。EtherCAT+HSL による経路では、ジッタ平均値 0.05405[msec]、標準偏差 0.09681[msec]、最悪値 0.6388[msec] となっている。RTC データポートの負荷が軽減されたことによって、2 つの通信方式においてどちらもジッタ性能が向上することが確認された。また、EtherCAT+HSL の平均ジッタ値は目的となる制御周期 0.200[msec] に対して平均的に満足する数字である。図 5.20 における値からさらに半減していることから、ソフトウェア処理の負荷による計測実行スレッドのジッタが大きく影響した数値であると考えられるため、純粋な通信タイミングジッタはさらに低い値であると考えられる。2.6.4 節にて述べた目標ジッタ性能は制御周期に対して 20%以内としていたが、ジッタ平均値 0.05405[msec] は 0.200[msec] の 27%に相当するため、目標値を越える値である。ただ、上述のように計測スレッド自体の実行周期ジッタが含まれていると考えられるため、通信タイミングジッタによる制御性能の低下はこの目標値に近いと推測される。あるいは、上位層側のデバイスインターフェースプログラムの実時間性能の不足や使用する EtherCAT ドライバ・ API が性能不足である可能性も考えられる。本研究において使用されている EtherCAT ドライバは、汎用 Ethernet ドライバ



(a) with EtherCAT+HSL, RTC and TCPROS.



(b) with EtherCAT+HSL and TCPROS excluding RTC.

図 5.19: Comparison of received reference joint angle trajectory.

API を使用するラッパーライブラリ形式であり、デバイスドライバとしてカーネルに組み込まれるタイプではない。そのため、システムコール等のオーバーヘッドによってジッタが生じていることが考えられる。

以上の結果は計算機資源の限られた組み込みプロセッサにおいて、ソフトウェア処理が必要となる通信方式ではモータのサーボ制御に必要とされる周期に間に合うための実時間性を保証できていないことを示している。一方で、通信処理の大部分を専用ハードウェアによってプロセッサのリアルタイムスレッドと並列に実行することで、要求される実時間性能に近い値を確保できていることも示している。また、現状のシステムを成す OS におけるリアルタイムスレッドでは、実時間性能はソフトウェア負荷に大きく依存しており、十分な実時間性能の提供が困難となる可能性が存在することを示唆する結果となった。この意味でも、ハードウェアによるアプリケーションレベルでの通信処理手法の有効性が示されていると言える。

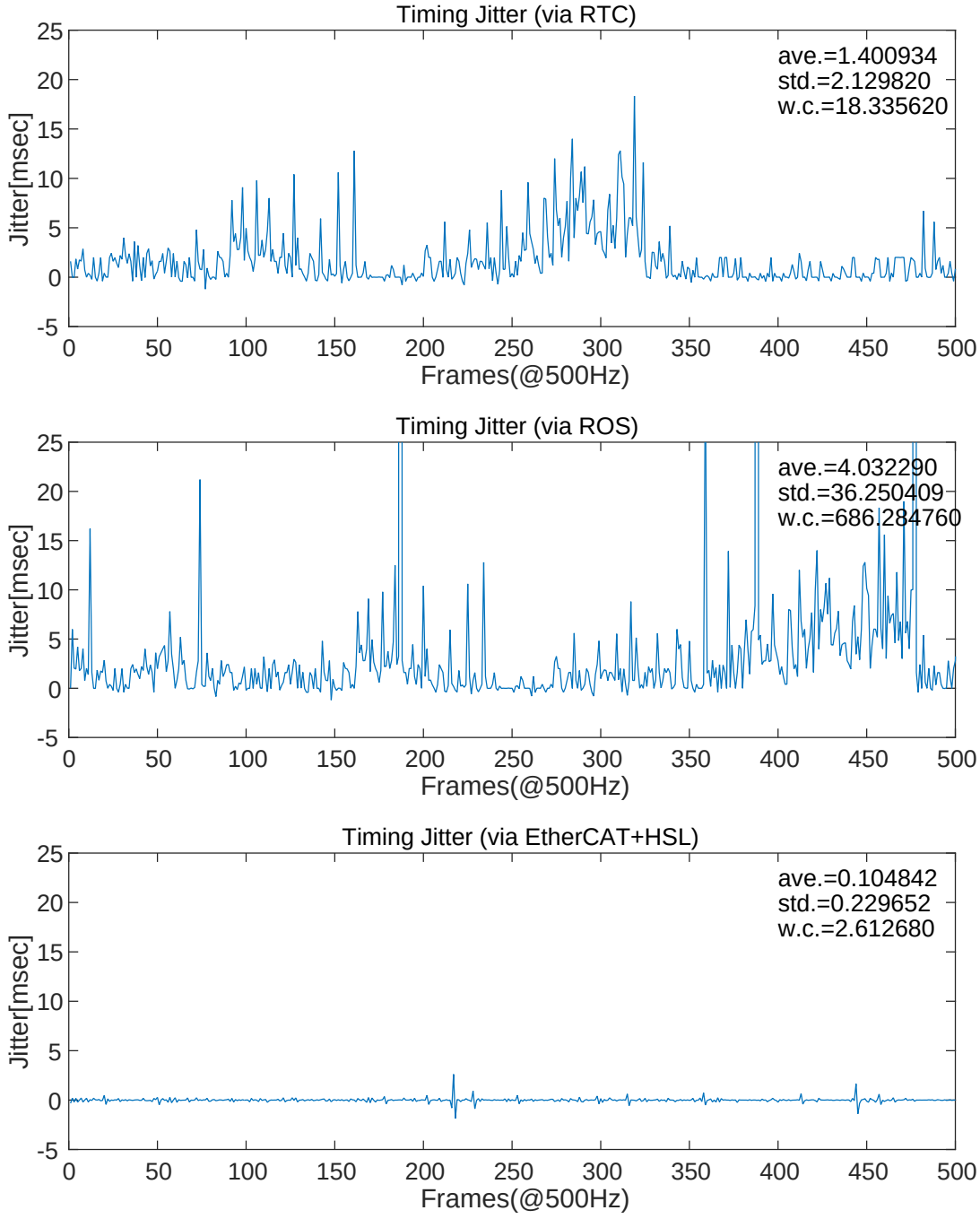


図 5.20: Timing jitter through the sub-system communication (using RTC Data Port).

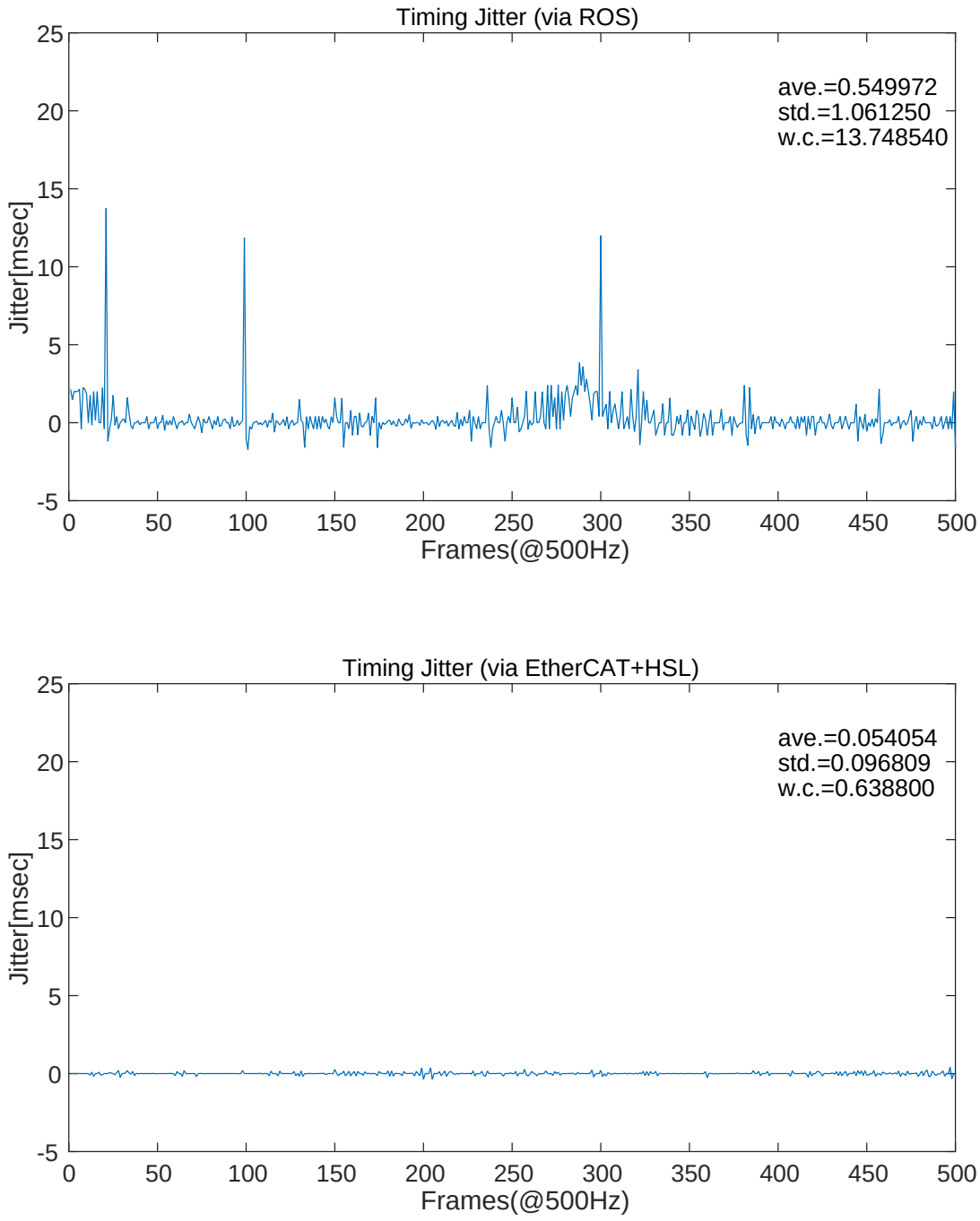


図 5.21: Timing jitter through the sub-system communication (without RTC Data Port).

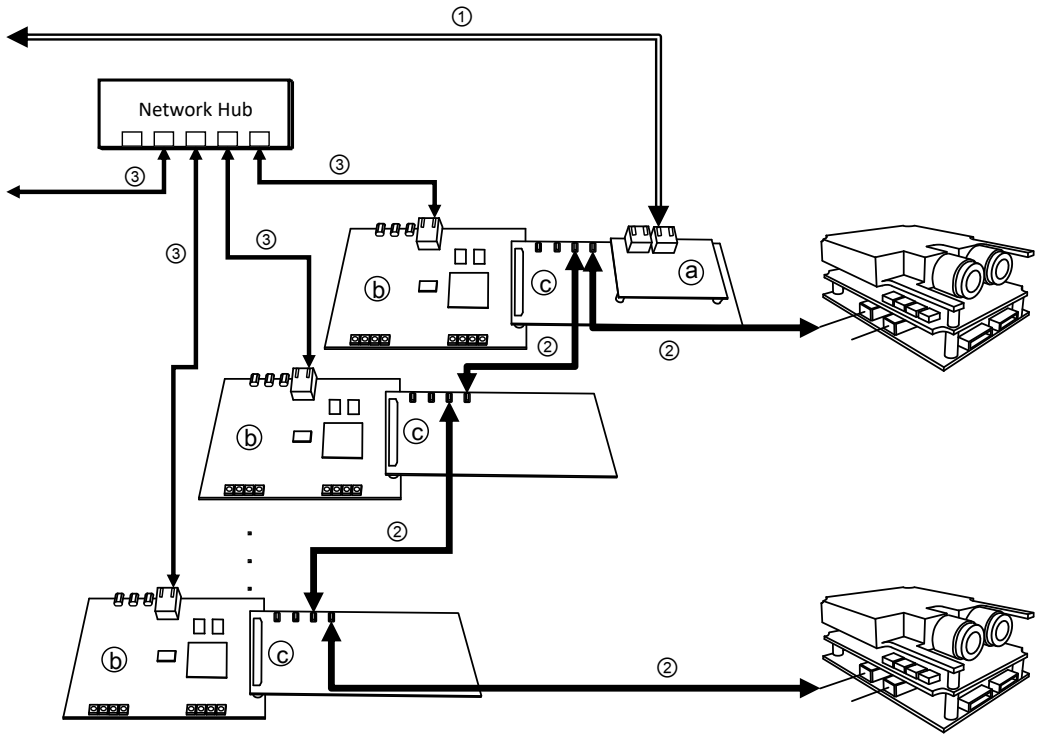


図 5.22: Single-Slave Single-Bridge configuration of multi middle layer controllers.

5.7 中間制御層の分散マルチノード構成

前節までは中間制御層を 1 ノードで運用する前提で述べてきたが、アクティブ光ファイバーリンクやロボット用ミドルウェアを利用することで、分散マルチノード構成を採ることが可能となる。これにより、例えば脚の右脚と左脚でノードを分割することや負荷の高い処理を別ノードで分散して行うということが可能となる。分散処理によって、1 ノード当たりのシステム負荷は低減され、より高度な制御則を適用したり、実時間性の高い処理への影響を低減するなどの利点が得られると考えられる。特に、アクティブ光ファイバーリンクの分散共有メモリブロックでは、利用可能なメモリ空間の制約はあるが非常に低レイテンシな構造体データ転送ができるため、ノード間で制御周期やデータを同期させることも実現可能となる。これらの特徴はシステムにスケーラビリティを持たせる上で重要となる。

5.7.1 分散ノード群でのグローバルクロックの利用

中間制御層においてもアクティブ光ファイバーリンクによって、RMTP-02D 基板と同じグローバルクロックを共有される。このグローバルクロック同期プロトコルブロックは FPGA 上に実装されているが、制御プログラムからはカーネルモジュールを介してアクセスされる。複数ノードでデータログを記録する場合に時刻同期は大きな問題となるが、中間制御層ではグローバルクロックを基準として 10nsec 単位の精度で時刻を記録しておくことが可能となっている。

5.7.2 分散マルチノード構成時のサブシステム間接続トポロジ構成

上位システムとの間のサブシステム間通信には、実時間通信系として EtherCAT を、非実時間通信系として汎用の Ethernet を利用している。Ethernet に関してはスイッチングハブを介して簡単に拡張可能である。EtherCAT に関しては以下の構成方法が考えられる。

1. シングルスレーブ-シングルブリッジ
   複数中間制御層の内、最上流ノードについて EtherCAT スレーブとネットワークブリッジを構成す

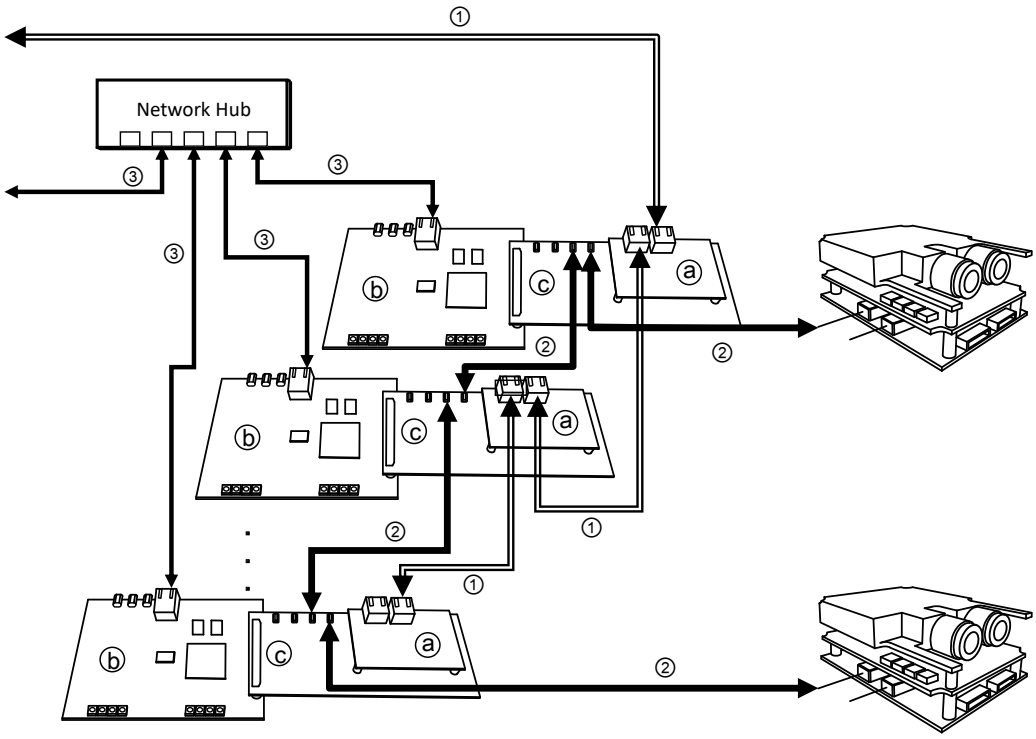


図 5.23: Multi-Slave All-Bridge configuration of multi middle layer controllers.

る。全てのパケットデータは最上流ノードから送受信される。上位システムからはシングル EtherCAT ノードとして扱われる。図 5.22 に配線図を示す。

2. マルチスレーブ-オールブリッジ
   全ての中間制御層に EtherCAT スレーブとブリッジを搭載し、各ノードごとに EtherCAT パケットデータを分担する。パケットデータは各ノードを起点としてアクティブ光ファイバーリンク側で送受信される。上位システムからは、複数の EtherCAT ノードとして扱われる。図 5.23 に配線図を示す。

図中インデックスは表 5.3 を参照。

表 5.3: Explanation of indexes in Fig.5.22 and Fig.5.23

index	Name
①	EtherCAT
②	active optical fiber link
③	Ethernet
ⓐ	EtherCAT Slave Device
ⓑ	Middle Layer Controller
ⓒ	active optical fiber link connector board

上記 2 とした場合、パケット処理がノード間で分散されるため、データフローが増大する大規模システムにおいて効果が高いと考えられる。しかしながら、配線図を見てもわかるように EtherCAT スレーブを各ノードごとに搭載する必要や太い Ethernet ケーブルでの配線部が増加するというハードウェアコスト的な問題が生じる。また、上位システム側は中間制御層の構成に合わせたサブシステム間通信処理を実行する必要が生じる。

一方、1 のシングルスレーブ-シングルブリッジ構成とした場合、上位システムからはシングルノード構成時と変わらない処理で問題は生じない。EtherCAT 周辺のハードウェア構成もスリム化されるため、信頼性の面でも優れていると考えられる。

本研究の時点では、上位システムを従来システムからの変更点を最小限に抑えたいという点や実験で利用可能な EtherCAT スレーブデバイスの都合からシングルスレーブ-シングルブリッジ構成を採用している。ただし、マルチスレーブ-オールブリッジ構成を採用することも実装上可能であり、アクティブ光ファイバーリンクによってロボットのシステム規模に応じて構成を可変できるフレキシビリティを提供していると言える。

5.8 上位層制御システム

上位層の制御システムの構成について、基本的には 5.2 節にて述べた従来型の制御システム構成を用いる。これは中間層システムに透過的な特性を持たせているため、従来型の制御システムとの完全な互換性を維持できているからである。ただし、これは従来の制御則を利用する場合の構成であり、本研究の対象である即応的反射行動アーキテクチャを実現する上では、上位層制御システムで実行される制御則について、ロボットの状態の変化に応じたオンラインでの目標軌道修正を行う身体運動計画系を設計する必要がある。

目標軌道修正は実行しようとしているタスクに応じて多岐に渡るが、ここでは歩行とバランスの制御についてオンラインでの修正に焦点を当てて、必要となる実時間歩行生成器を有する身体運動計画システムの構成方法を述べていく。

5.9 実時間歩行軌道生成

脚を持ったロボットの運用においては、移動指令入力や力センサ・姿勢センサ等からのフィードバック入力に応じて逐次的に歩行用脚関節軌道を生成する必要がある。移動指令に対するレスポンスや転倒を回避するためにはこの逐次実行に要する演算時間を可能な限り短く済ますことが重要となる。

実時間での歩行軌道生成については過去に多くの研究が成されてきており、梶田らの倒立振子モデル (Linear Inverted Pendulum Model, LIPM) に基づいた実時間歩行軌道生成 [73][74]、予見制御に基づいた重心軌道生成手法を示している [75]。加賀美、西脇らはヒューマノイドロボット H5、H7 を使用して、より高速な ZMP 追従重心軌道の生成手法を示している [76][77]。これらの研究で使用されている予見制御手法による重心軌道の計算では重心高が変化する等リカッチ方程式の解が必須となる場合には、この求解のために反復計算が行われるためオンラインでの逐次実行には不利となる。本研究でも用いる HRP3L-JSK では浦田らが予見制御ベースの高速な軌道生成手法 [78] を用いて、数十 msec サイクルでの脚軌道修正計算を実現している。他に重心軌道の生成では、微分動的計画法 (Differential Dynamic Programming, DDP) や iterative Linear Quadratic Regulator (iLQR) 等の軌道最適化手法が用いられており、Feng ら [79] の歩行制御や Tessa ら [80, 81] の運動生成に利用されている。これらの手法の特徴としては、コスト関数、ダイナミクス関数について非線形関数を適用することが容易であり、多くのモデルについて軌道最適化を実施することが可能である。しかしながら、軌道生成に要する演算量は予見制御の場合と比較して非常に多く、特に高階導関数が陽に与えられないモデルについては数値計算によって都度ベクトル・行列微分計算が必要となるため、短周期でのオンライン軌道生成を実現するためには相応の計算機性能をロボットの体内に用意する必要が生じる。これは消費電力やパッケージサイズの観点から好ましくない影響である。ヒューマノイドロボットにおいて特に基本動作となる歩行動作について、演算負荷の小さい予見制御ベースの手法によって実現することで電力負荷の低減につながると考えられる。

また、不整地路面においても歩行を実現するために計画軌道からのずれをセンサから推定し、逐次的に歩行軌道生成にフィードバックする手法として西脇らによって提案されている [82][83]。

本研究は浦田らの手法をベースに hrpsys 上に実時間歩行軌道生成器を実装する。以下に歩行軌道生成の内容について記述していく。

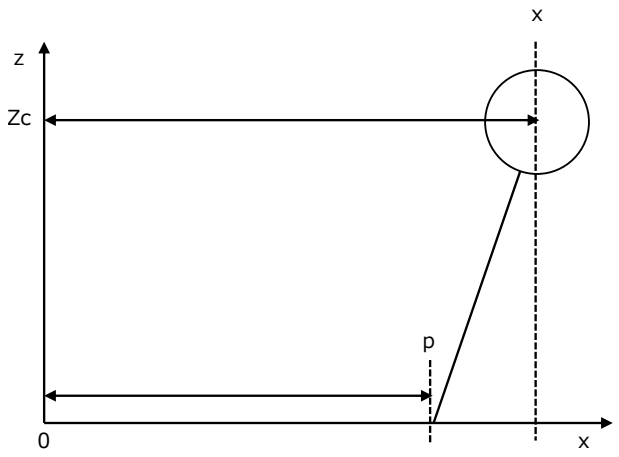


図 5.24: Linear Inverted Pendulum Model (LIPM).

5.9.1 倒立振子モデルベース重心軌道生成

重心高さ Zc を一定とした LIPM(図 5.24) について、サイクル i における状態 x_i を ZMPp、重心並進位置 x、重心並進速度 $\dot{x}$ を用いて以下のように定義する。

$$\boldsymbol{x}_i := \begin{bmatrix} p \\ x \\ \dot{x} \end{bmatrix} \tag{5.27}$$

この状態 x_i について、運動方程式は以下のように表される。

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \Delta t \\ -a^2\Delta t & a^2\Delta t & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \Delta t \\ 0 \\ 0 \end{bmatrix} u_k \tag{5.28}$$

ここで、入力 u_k は ZMP 変分を表す。また、a は重力加速度 g と重心高さ Zc を用いて、

$$a := \sqrt{\frac{g}{Zc}} \tag{5.29}$$

と定義される定数である。

式 5.28 について、以下のように Linear Quadratic(LQ) 制御問題として定式化される。

$$J = \sum_{j=1}^{\infty} Q(p_j^{ref} - p_j)^2 + Ru_j^2 \tag{5.30}$$

ここで、$Q \in \mathbb{R}$、$R \in \mathbb{R}$ の重みづけパラメータである。

梶田らは上記の LQ 制御問題について、以下の予見制御状態フィードバック入力を解として与えることで参照 ZMP 軌道に対する最適な重心軌道を得ることができることを示した [75]。

$$u_k = -\boldsymbol{K}\boldsymbol{x}_k + \sum_{i=1}^{\infty} f_i p_{k+i}^{ref} \tag{5.31}$$

通常、このような LQ 制御問題については Riccati 方程式を解くことによって、フィードバックゲインを導出する。

Peng ら [84] は $\| R \| \to 0$ の極限の場合について上記の LQ 制御問題について、陽に解を与えられることを示しており、浦田らはその解について以下のように与えている [78]。

$$\boldsymbol{K}^* = \begin{bmatrix} \dfrac{1+a\Delta t}{\Delta t} + \dfrac{a}{1+a\Delta t} & -\dfrac{2+a\Delta t}{\Delta t} & -\dfrac{2+a\Delta t}{a\Delta t} \end{bmatrix} \tag{5.32}$$

$$f_i^* = \frac{\delta_{i1}}{\Delta t} - a(2 + a\Delta t)\left(\frac{1}{1 + a\Delta t}\right)^{i+1} \tag{5.33}$$

ここで式 5.31 の右辺畳み込み演算部を以下のように定義する。

$$u_k^{ff} := \sum_{i=1}^{\infty} f_i p_{k+i}^{ref} \tag{5.34}$$

すると、式 5.33 を用いることで u_k^{ff} について以下のように漸化式で書き下すことが可能である。

$$S_n := \sum_{i=n}^{\infty} \frac{p_k^{ref}}{(1 + a\Delta t)^{k-n}} \tag{5.35}$$

$$S_{k-1} = p_k^{ref} + \frac{S_k}{1 + a\Delta t} \tag{5.36}$$

$$u_k^{ff*} = \frac{p_k^{ref}}{\Delta t} - \frac{a(2 + a\Delta t)S_{k-1}}{(1 + a\Delta t)^2} \tag{5.37}$$

漸化式形式は畳み込み形式と比較して、計算量を削減することができるため有効である。

さらに浦田らは上記制御入力について、CoM が発散しないための ZMP 目標入力の条件を求めている。ZMP 追従誤差 e_i について,

$$e_i := p_k - p_{k-1}^{ref} \tag{5.38}$$

と定義するとき、初期 ZMP 追従誤差について

$$e_1 = \left| p_0 - p_0^{ref} + \Delta t\left(- \boldsymbol{K}^* \begin{bmatrix} p_0 \\ x_0 \\ \dot{x}_0 \end{bmatrix} + \sum_{i=1}^{\infty} f_i^* p_i^{ref} \right) \right| = 0| \tag{5.39}$$

を ZMP 完全追従の条件として求めている。

式 5.39 について、参照 ZMP 列を差分形式 $\Delta p_i^{ref} := p_{i+1}^{ref} - p_i^{ref}$ で表現すると以下の式 5.40 ように簡略化される。

$$0 = \begin{bmatrix} 1 & 1 & \dfrac{-1}{a} \end{bmatrix} \begin{bmatrix} p_0 \\ x_0 \\ \dot{x}_0 \end{bmatrix} + \sum_{i=1}^{\infty} \frac{\Delta p_i^{ref}}{(1 + a\Delta t)^{i+1}} \tag{5.40}$$

この条件式は $step$ 歩以降についても同様に成立し、以下の式 5.41 ように記述される。

$$0 = \begin{bmatrix} 1 & 1 & \dfrac{-1}{a} \end{bmatrix} \begin{bmatrix} p_{step} \\ x_{step} \\ \dot{x}_{step} \end{bmatrix} + \sum_{i=1+step}^{\infty} \frac{\Delta p_i^{ref}}{(1 + a\Delta t)^{i+1}} \tag{5.41}$$

一定周期ごとに式 5.41 に従って、現在スタックされている目標 ZMP 列を修正することで転倒を回避する歩容が生成される。

以上の ZMP 修正手法及び予見制御ベースでの重心軌道生成手法を利用して、ロボットの参照体幹軌道、参照脚軌道の生成を行う。図 5.25 に示すように、体幹・脚軌道生成器への入力として、LIPM の初期状態 x_0 並びに、歩行の参照ステップとなる ZMP 変分列 $\{\Delta p_i^{ref}\}_{i=1,\ldots,n}$ を与える。式 5.41 に従って、$\{\Delta p_i^{ref}\}_{i=0,\ldots,n}$ を修正し $\{\Delta p_i^{refmod}\}_{i=1,\ldots,n}$ とする。修正 ZMP$\{\Delta p_i^{refmod}\}_{i=1,\ldots,n}$ は脚軌道生成器、重心軌道生成器それぞれの入力となる。これらの生成器はロボットの体幹軌道 $\{p_i^{body}\}_{i=0,\ldots,n}$、左右脚軌道 $\{\boldsymbol{p}_i^{footL}\}_{i=0,\ldots,n}$,$\{\boldsymbol{p}_i^{footR}\}_{i=0,\ldots,n}$ を計算し、ロボットの関節角度軌道を生成するための参照値となる。

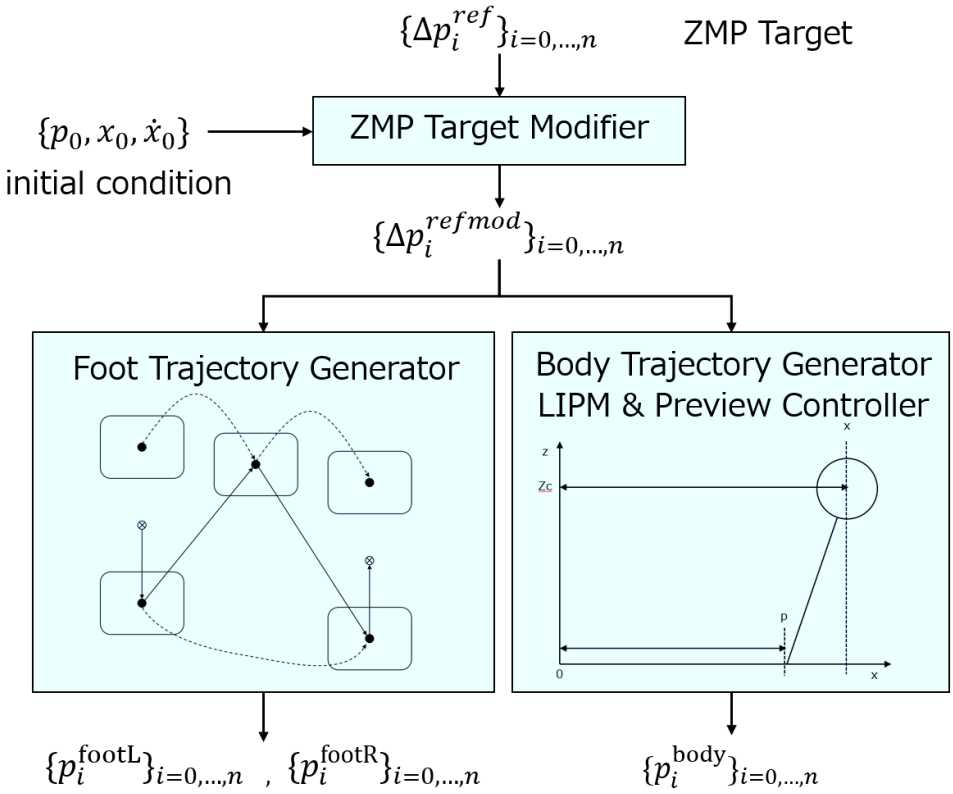


図 5.25: Foot and Body trajectory generation by singular LQ Preview Controller.

5.9.2 足部面内 ZMP 修正許容領域に応じた目標 ZMP 軌道修正

LIPM に基づいた歩行軌道の生成では、目標 ZMP 軌道に従って脚軌道が生成される。標準では、脚軌道は足部の固定位置(足部中心等) が目標 ZMP と重なるように軌道を生成する。しかし、足部の面内であれば任意の点に目標 ZMP を重ねるよう脚軌道を生成しても LIPM に基づいて生成した軌道と差異は生じない。これは、足部の面広さに基づいて脚軌道の目標位置設定に自由度が与えられると言える。これによって、式 5.41 に基づいた目標 ZMP 軌道の修正によって本来の歩容軌道から外れた ZMP 軌道となる場合についても、足部の面内に限り元の歩容軌道を維持することが可能となる。

通常はこの足部面内の領域での ZMP 軌道修正によって、式 5.41 を満足することは可能である。外乱によって、式 5.41 の初期条件 $[p_{step}, x_{step}, \dot{x}_{step}]^{\mathrm{T}}$ に目標値からの大きな誤差が生じ、足部面内での ZMP 軌道修正では式 5.41 を満足することが不可能となった場合には、脚軌道自体が修正され、転倒を回避する歩容を生成する。以上を定式化すると以下のようになる。歩行の k ステップ目において式 5.41 によって p_k^{ref} を修正した目標 ZMP を ${}_{mod}p_k^{ref}$ とし、足部面領域を S_{foot} と表すことにする。脚軌道の目標位置 p_k^{foot} は以下の条件式によって決定される。

$$\begin{cases} p_k^{foot} = p_k^{ref} & \text{if} \quad {}_{mod}p_k^{ref} \in S_{foot} \\ p_k^{foot} = {}_{mod}p_k^{ref} & \text{if} \quad {}_{mod}p_k^{ref} \notin S_{foot} \end{cases} \tag{5.42}$$

ただし、右脚、左脚は適宜切り替えるものとする。

足部面内領域における ZMP 軌道修正について、実装においてはマージンを持たせるために、実際の足部面よりも内側にオフセットした領域を足部 ZMP 修正許容領域として利用する。本研究においては、実足部面に対して 10[mm] 内側にオフセットした領域を設定している。

5.9.3 多リンクモデルによる ZMP 誤差補償

図 5.26 に示すように、LIPM からの出力重心軌道と ZMP 軌道をそのまま多リンクのロボットに適用した場合、各リンク質点の分布や慣性モーメントの存在によって実現される ZMP 軌道は計画時の ZMP 軌道

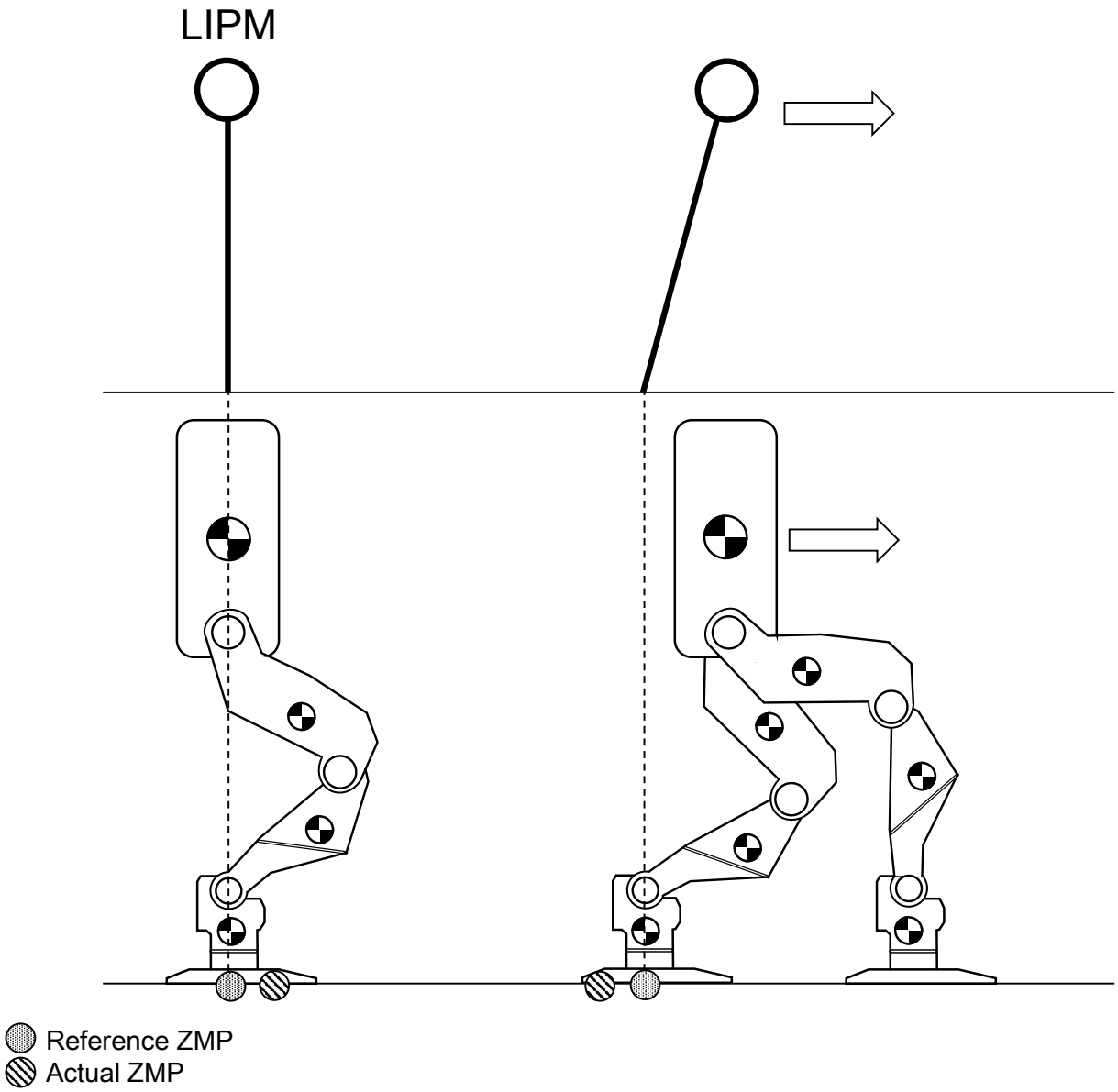


図 5.26: ZMP errors with multi-body dynamics of the robot.

とは異なってしまう。実 ZMP 軌道が計画した参照 ZMP 軌道に追従するようロボットの関節角度軌道は修正される必要がある。浦田らは LIPM 重心軌道を直接実ロボットの関節軌道にマッピングした場合に推定される実 ZMP 軌道をロボットのマスパラメータから計算し、参照 ZMP 軌道との誤差を重心軌道にフィードバックすることで実 ZMP 軌道を補償している。

これらの計算は単質点モデルでの計算と比較すると負荷が大きいため、毎制御周期ごとに演算を行うのは効率的ではない。しかし、実際に単質点モデルと多リンクモデルとの推定 ZMP 誤差が顕著になるのは生成軌道の支持脚切替の生じるタイミングであり、軌道の初期フレームでは単質点モデルから生成された軌道を再生していても実用上は問題が生じない。従って、多リンクモデルへのマッピングのための体幹軌道修正は、主となる軌道生成コンテキストとは異なるスレッドで並行して処理を行うことが可能であり、支持脚切替が生じる数百制御周期（ 数百 msec）以内に修正計算を完了すればよい。実際には HRP3L-JSK の 12 関節,13 リンクのロボットにおいて、数十 msec 以内にこの処理は完了するため、実時間性の観点からも十分なマージンがあると考えられる。

演算コストを低下させるためにさらに実装上は、慣性テンソルの成分で ZMP への寄与が無視できると考えられるものや、3 軸直交関節の中間リンクのような重量が軽量なリンクについては、計算を省略することで演算を高速化している。

各リンクを質点として近似した多質点モデルでのロボットの運動と ZMP の関係式は式 5.43 で表される。

$$p_x = \frac{\sum_{i=1}^{n} m_i z_i \ddot{x}_i - \sum_{i=1}^{n} \{m_i(\ddot{z}_i + g)x_i + [0,1,0]\boldsymbol{I}_i\dot{\boldsymbol{\omega}}_i\}}{-\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{5.43}$$

$$p_y = \frac{\sum_{i=1}^{n} m_i z_i \ddot{y}_i - \sum_{i=1}^{n} \{m_i(\ddot{z}_i + g)y_i - [1,0,0]\boldsymbol{I}_i\dot{\boldsymbol{\omega}}_i\}}{-\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \tag{5.44}$$

上式で計算された ZMP と参照 ZMP との誤差を下記のように係数 $K_x, K_y \in \mathbb{R}$ を用いて参照 ZMP を修正する。

$$_{mod}p_x^{ref} = p_x^{ref} + K_x(p_x^{ref} - p_x) \tag{5.45}$$

$$_{mod}p_y^{ref} = p_y^{ref} + K_x(p_y^{ref} - p_y) \tag{5.46}$$

修正された参照 ZMP 軌道から再度 LIPM によって重心軌道を生成することで、多リンクモデルによる ZMP 誤差を補償している。本研究においては係数 K_x, K_y は 0.5 に設定した。これは、事前検証の結果この値において軌道の発散は起こらず、収束性が高いことから決定した。この ZMP 誤差補償の方法は梶田らの分解運動量制御 [85] による方法と比較して、演算負荷が小さい点が利点となるが、上体も含めた角運動量の補償は行われないため、脚型ロボットにおいて有効な方法であると言える。

倒立振子モデルベース重心軌道生成並びに多リンクモデルによる ZMP 誤差補償によって生成される体幹位置軌道、ZMP 軌道を x 軸 (前後方向)、y 軸 (左右方向) について、前進4 歩の指令によって実際に計算されたものを図 5.27 中の図 5.27a と図 5.27b にそれぞれ示す。ZMP 軌道に対して前後体幹位置軌道はやや後方にオフセットした軌道となっているが、これはロボットの前後重心位置の体幹座標原点からのオフセット距離を反映したものであり、多リンクモデルによる ZMP 誤差補償によって重心位置を考慮した軌道に修正されていることが確認される。また、図 5.27c 及び図 5.27d は左脚、右脚それぞれで発揮される目標力を示しており、図 5.27e は脚軌道の高さ方向についての軌道を示したものである。これらも LIPM から計算される値であり、目標関節トルク計算の入力値として使用される値である。

5.9.4 遊脚接地高修正制御

路面上の凹凸の存在やロボットの体全体の傾きによって、遊脚の着地時の接地タイミングが計画脚軌道と異なる場合、遊脚が必要以上に路面を押し込むことによってロボットの体幹が大きく傾き、転倒を引き起こしてしまう。そのため遊脚の実際の接地タイミングに応じて計画軌道の修正を行う必要がある。従来の hrpsys では足裏力センサによって計測された床反力からダンピング制御を行い、さらに IMU から推定された体幹姿勢に応じて足部高さを修正する制御が行われていた。しかし、IMU や力センサを安定にフィードバック制御に用いるにはローパスフィルタによるノイズの除去が不可欠であり、応答遅れが生じていた。そのため高速な歩行を行うと、接地タイミングの誤差によって姿勢を大きく崩す場合が見られた。従って高速な脚動作において接地安定性の向上が求められる。

接地安定性向上の単純な方策として、力センサで計測される反力について閾値判定によって接地/不接地を判断し、接地のタイミングで高さ方向の足軌道を停止することが考えられる。図 5.28 に、接地判定による足高さ軌道修正の概要を示す。接地タイミング判定時の足高さ p_{footZ}^{orig} を足高さ修正量 p_{footZ}^{mod} として記憶する (式 5.47)。

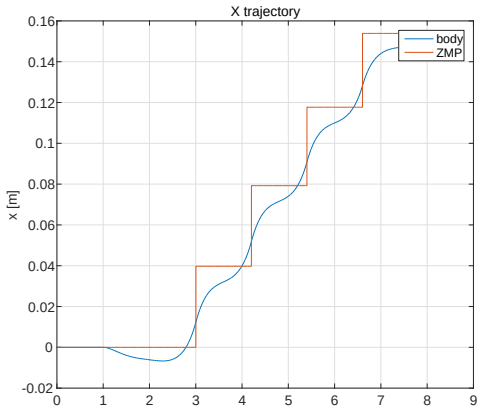
$$p_{footZ}^{mod} = \begin{cases} 0 & (f_z < threshold) \\ p_{footZ}^{orig} & (f_z \geq threshold) \end{cases} \tag{5.47}$$

最終的な足高さ参照値 p_{footZ}^{ref} は次式で決定される。

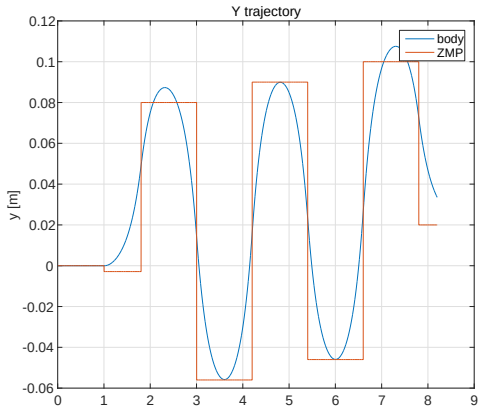
$$p_{footZ}^{ref} = \max(p_{footZ}^{orig}\ ,\ p_{footZ}^{mod}) \tag{5.48}$$

なお実機での接地判定においては、ノイズを考慮してシュミットトリガー方式の閾値判定を用いる (図 5.28 の赤実線)。

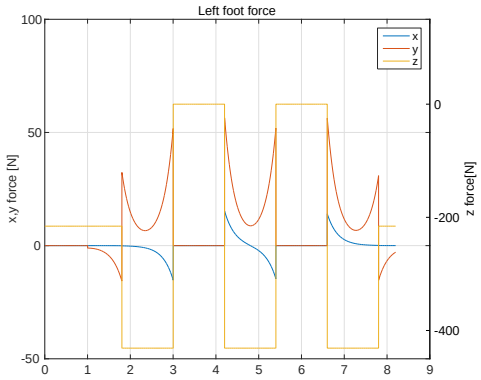
このように単純に接地タイミングでの足高さを維持した場合、足高さ修正量 p_{footZ}^{mod} がステップごとに積算されるため体が沈み込んでしまう場合が見られる。従って、それを防ぐために足高さ修正量に忘却係数を乗じることで足高さ修正量 p_{footZ}^{mod} の積算に制限を加えている。



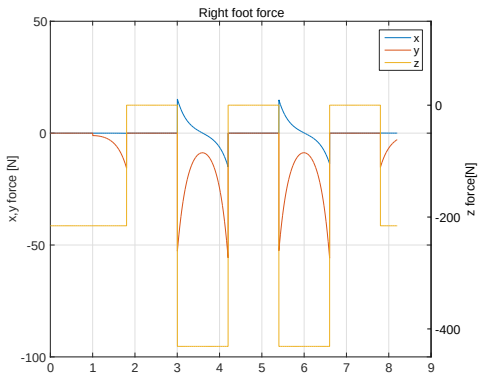
(a) X body pos and target ZMP



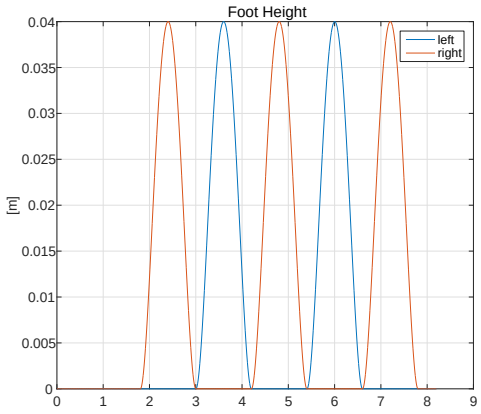
(b) Y body pos and target ZMP



(c) Left foot reference force



(d) Right foot reference force



(e) Foot height

図 5.27: Calculation results of forward four steps.

図 5.29 に実機での足高さ修正による足軌道と床反力の計測値を示す。図中の時刻 0.56 秒付近で足高さはまだ 1cm 近く軌道が残っているが、床反力が立ち上がり始めているのが確認できる。このタイミングで接地を判定し、足高さ修正による軌道変更が行われたことで床反力は一次的に低下し、軌道計画時の本来の接地タイミングである 0.6 秒付近で再び床反力が発生している。本節で実装した足高さ修正は単純な方策であるが、転倒回避に効果があることが確認された。

5.10 絶対座標基準位置姿勢推定器

5.9 節にて述べた歩行軌道生成器では、サイクルの初期状態 $[p_{step}, x_{step}, \dot{x}_{step}]^{\mathrm{T}}$ に基づいて転倒を回避するための ZMP 軌道を生成した。この初期状態について、実ロボットの状態を反映させることによって、オンラインでの転倒回避を実現することが可能になると考えられる。実 ZMPp_{step} については、足先に取り

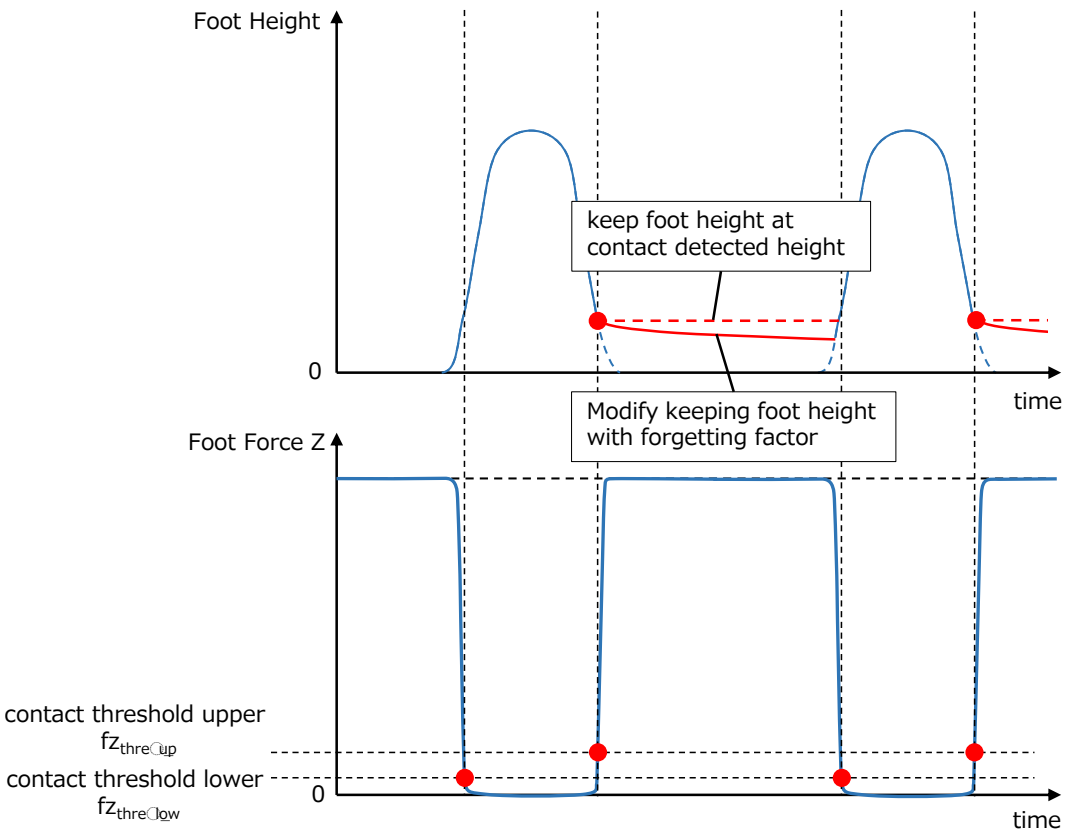


図 5.28: Foot height modification by detecting touch down timing.

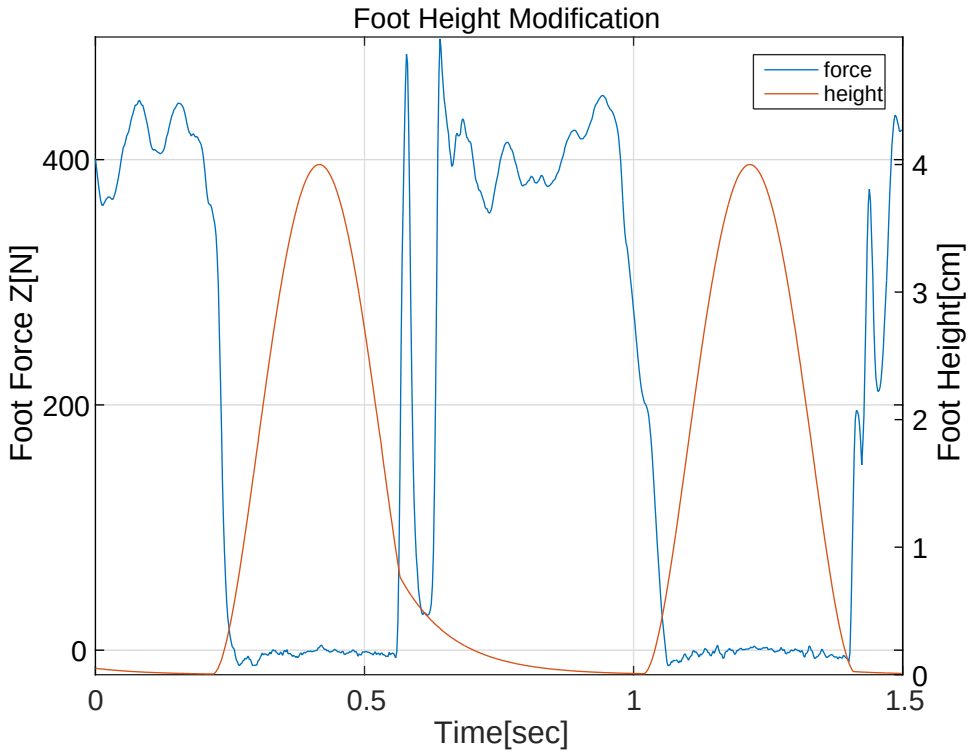


図 5.29: Measured foot height trajectory and reaction force on foot.

付けられた6 軸力センサから計算される値を利用することが可能である。しかしながら、重心位置・速度 $[x_{step}, \dot{x}_{step}]$ については、直接的にセンサから計測することは困難であり、他の状態量から推定する必要がある。ロボットの重心位置・速度の状態推定については多くの先行研究例があり、特に脚移動型については支持脚を基準に積算していく方法が一般的となっている。これは、支持脚は地面に対して滑りが生じないという仮定の下、計測された関節角度から順運動学計算によって重心位置を順次求めていく方法となる。ただし、現実には不正路面上の傾きや滑りの存在によって支持脚は地面に対してわずかながら運動するため、推定される重心軌道に誤差が生じてしまう。Benallegue らは IMU から推定される体幹姿勢を基準に足先の姿勢変化を計算し、カルマンフィルタに基づいた支持脚の姿勢変動の推定によって重心軌道の推定を行っている [86]。Masuya と Sugihara らも支持脚接地点の推定を考慮した方法を提案している [87, 88]。西脇らは支持脚接地点が ZMP に一致すると仮定し、ZMP 周りの回転によって体幹の変位が生じるとしている [89, 82]。また、計測される床反力から運動方程式に基づいて重心運動をカルマンフィルタによって推定する手法について Kwon ら [90] が提案している。Stephens らは IMU 加速度を元に重心軌道の推定を提案している [91]。また、カメラ画像処理に基づいた自己位置推定によって推定値に乗るドリフトを補正することも考えられる。

これらの手法は復号的に利用することでさらにロバストな推定精度を得られることが知られており、本研究段階では比較的実装の容易でパラメータチューニングを要さない支持脚を基準位置とした ZMP 計測値に基づく接地点を考慮する体幹軌道推定による手法を採用するが、加速度情報やカルマンフィルタの利用によってさらに推定精度の向上が期待される。

本研究では第 k フレーム現在における関節角度から順運動学計算によって計算される支持脚足先位置 $\boldsymbol{p}_k^f$ と ZMP 計測位置 $\boldsymbol{p}_k^{zmp}$ について、以下のように支持脚からの体幹相対位置 $\boldsymbol{p}_k^{rb}$ を算出する。

$$\boldsymbol{p}_k^{off} = \boldsymbol{R}_k^{imu}(\boldsymbol{E} - (\boldsymbol{R}_k^f)^{\mathrm{T}})(\boldsymbol{p}_k^f - \boldsymbol{p}_k^{zmp}) \tag{5.49}$$

$$\boldsymbol{p}_k^{rb} = -\boldsymbol{p}_k^f - \boldsymbol{p}_k^{off} \tag{5.50}$$

$$\tag{5.51}$$

ここで、$\boldsymbol{R}_k^{imu} \in \mathbb{R}^{3\times3}$ 及び $\boldsymbol{R}_k^f \in \mathbb{R}^{3\times3}$ は、それぞれ IMU 計測値からカルマンフィルタによって推定された体幹姿勢、並びに支持脚足先姿勢を表す。$\boldsymbol{p}^{off}$ は支持脚の ZMP 周りの回転によって生じる変位を示す。

推定された体幹相対位置 $\boldsymbol{p}_k^{rb}$ を元に、体幹推定速度 $\boldsymbol{v}_k^{est}$ を計算する。

$$\boldsymbol{v}_k^{est} = \begin{cases} \boldsymbol{v}_{k-1}^{est} & \text{if } cs(k) \neq cs(k-1) \\ \frac{\boldsymbol{p}_k^{rb} - \boldsymbol{p}_{k-1}^{rb}}{dT} & \text{if } cs(k) = cs(k-1) \end{cases} \tag{5.52}$$

$cs(k)$ はフレーム k における支持脚接地状態を示し、支持脚切替の判定に用いている。

最終的な体幹推定位置は体幹推定速度を積分して獲得される。

$$\boldsymbol{p}_k^{est} = \boldsymbol{p}_{k-1}^{est} + \boldsymbol{v}_k^{est} dT \tag{5.53}$$

ZMP 計測並びに支持脚の接地判定において、足裏に取り付けられた6 軸力センサから取得される足裏反力情報を利用する必要が生じる。そのため、ノイズによる誤差やフィルタリングに由来する遅延によって、推定値は特に支持脚切替タイミングで大きくドリフトが生じる傾向にある。しかしながら、出力である体幹推定位置は、歩行動作生成器においては支持脚位置との相対差として LIPM に反映するため、絶対空間内での位置ドリフトについては影響を打ち消すことが可能である。

5.11 姿勢安定化制御

前節までの実時間歩行生成では、IMU 情報から推定された姿勢を元に転倒を回避する歩容を計算していた。しかし、多リンクモデルと実機との間に存在するモデル化誤差、不整路面や外乱による ZMP 追従誤差の発生による不安定化に対して完全には対応できない。既存のシステムでは床反力や姿勢変化に対してバランスを維持するために hrpsys の Stabilizer コンポーネントで実行される LIPM トラッキング制御、床反力ダンピング制御によって補償が行われている。一般に LIPM トラッキング制御や床反力ダンピング制御では、足裏に搭載された6 軸力センサによって計測された床反力をフィードバックすることで実現される。しかしながら力センサは特に S/N 比が悪いセンサであるため、関連するフィードバックパラメータは発振

を避けるためにコンサバティブな設定となっている。これにより応答性能が犠牲となっており、体幹が素早く傾くといった外乱に対しては応答が遅れて転倒してしまうという例が見られる。

以上のような問題は、従来の姿勢安定化制御では制御対象である床反力を関節角度の修正によって制御しようとする点に難しさがあると考えられる。理想的な応答を示す力センサは利用できないという前提の下では、床反力を制御するためには関節制御のトルクベース化が有効であると言える。トルクベースでの関節制御によってロボットの運動は外乱に対して物理的な作用によって応答するため、外乱応答性の向上が期待される。反対に関節角度の追従性能は悪化してしまうが、関節角度誤差によって生じるダイナミクス誤差の時定数は例えば等身大倒立振子モデルの場合約 200msec 以上あり、制御周期に対して十分長いため 5.9 節にて導入したオンラインでの動作修正手法によって間に合う。トルクベースの姿勢安定化を試みる研究として、Stephens らは LIPM に基づいた姿勢安定化トルク計算による手法を提案している [91, 92, 93]。床反力センサに依存しないより実機モデルを反映したトルクベースでの姿勢安定化手法として、鈴木らのトルクベーススタビライザ [94] を本研究では使用する。

以下では従来用いられていた床反力フィードバックベースの手法である LIPM トラッキング制御と床反力ダンピング制御について述べた後、本研究の姿勢安定化制御として用いるトルクベーススタビライザの概要と制御システムへの組込み方法について述べる。

5.11.1 床反力フィードバック型姿勢安定化制御

LIPM トラッキング制御

LIPM トラッキング制御は梶田ら [95] が提案する ZMP、体幹姿勢、重心位置をフィードバックし、ZMP を目標に追従させつつ、姿勢変化や重心位置誤差を補償する制御方法であり、以下の式で目標 ZMP 修正が行われ、最終的に LIPM に基づいて目標重心位置、速度の修正が行われる。ZMP の計測値は床反力を元に計算されるため、フィードバックゲインは低めに設定される。

$$p^{ref*} = \boldsymbol{K}(\boldsymbol{x}^{ref} - \boldsymbol{x}) + p^{ref} \tag{5.54}$$

$$\boldsymbol{K} := [k_1\ k_2\ k_3]^{\mathrm{T}} \tag{5.55}$$

$$\boldsymbol{x} := [x\ \dot{x}\ p]^{\mathrm{T}} \tag{5.56}$$

床反力ダンピング制御

梶田らが提案する Foot Torque Control, Foot Force Difference Control[95] が hrpsys に実装されており、床の傾きや、凹凸を踏んだ際に足裏力センサのフィードバック信号を元に足先のダンピング制御を行うことによって、足部が路面に追従しバランスを補償する。ただし、前述のように力センサの S/N 比の悪さのため、応答性の良いゲインを設定すると制御が発振しだすため、かえって不安定化してしまうという問題がある。

5.11.2 トルクベーススタビライザ

床反力センサを使用しないトルクベース姿勢安定化手法として、鈴木らのトルクベーススタビライザ [94] を使用する。トルクベーススタビライザでは、ロボットをリンクモデルとして扱い、逆動力学演算等の多リンクダイナミクスモデルとしての計算を不要としている点を特徴としている。2 次計画法に基づく最適化計算以外は基本的な運動学計算のみで完結しているため、実時間での目標関節トルクの計算が可能となっている。

トルクベーススタビライザは以下の3 手順によって実行される。

1. 体幹・足先位置 PD 制御に基づく目標復元力計算
2. 復元力分配最適化計算

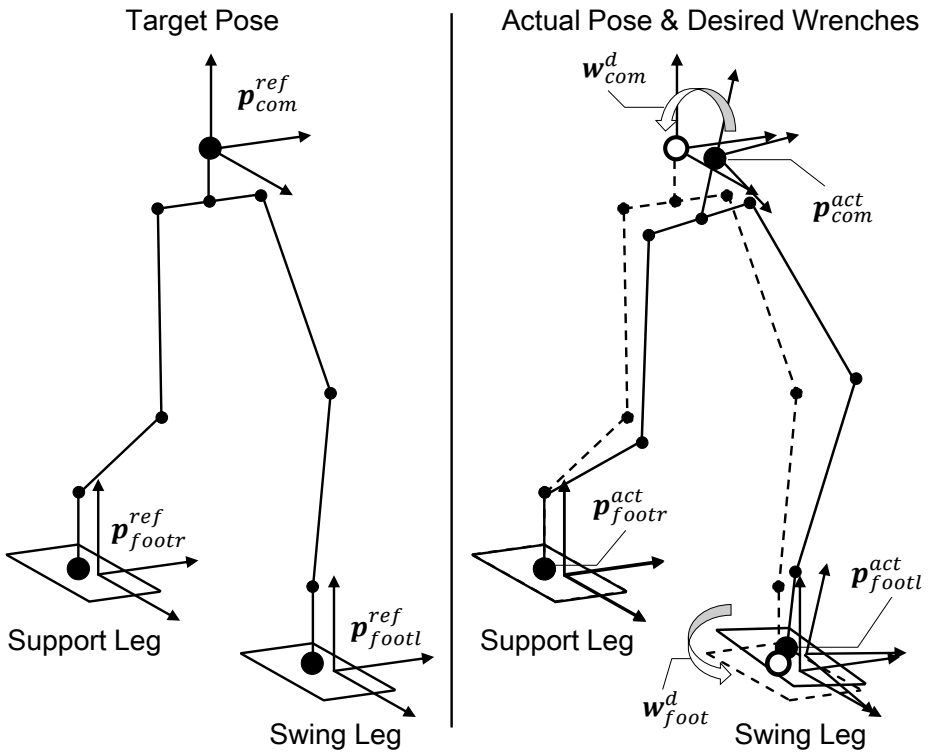


図 5.30: Body and foots coordinates used in torque-based stabilizer.

3. 関節トルク変換

以下に各処理の概要について述べる。

体幹・足先位置 PD 制御に基づく目標復元力計算

トルクベーススタビライザ前段の軌道生成器から得られる目標重心位置 $\boldsymbol{p}_{com}^{ref}$ 及び目標関節角度 $\boldsymbol{\theta}^{ref}$ から順運動学計算によって得られる左右目標足先位置 $\boldsymbol{p}_{footl}^{ref}$、$\boldsymbol{p}_{footr}^{ref}$ について、センサデータから推定される実重心位置、左右実足先位置 $[\boldsymbol{p}_{com}^{act}, \boldsymbol{p}_{footl}^{act}, \boldsymbol{p}_{footr}^{act}]$ との誤差から PD 制御器によって重心位置、遊脚足先位置に発生すべき目標復元力 $[\boldsymbol{w}_{com}^{d} \in \mathbb{R}^6, \boldsymbol{w}_{footl}^{d} \in \mathbb{R}^6]$ がそれぞれ計算される。図 5.30 に重心・左右足先それぞれの目標位置座標と、実位置座標の設定及び目標復元力を示す。

復元力分配最適化計算

前項目標復元力 $[\boldsymbol{w}_{com}^{d}, \boldsymbol{w}_{foot}^{d}]$ について、これらは内力であるため直接制御することが不可能である。支持脚に働く床反力を修正することで目標となる体幹周りの内力成分を実現する必要がある。また、左右足先復元力の発生によって実現 ZMP が目標位置から外れるため、目標 ZMP 追従のための制約が必要となる。

上記項目は全てを同時に満たすことは不可能であり、重みづけをした上で最適化問題を解く必要が生じる。最適化変数を左右足先復元力 $[\boldsymbol{w}_{footl}, \boldsymbol{w}_{footr}] \in \mathbb{R}^{12}$ とした上で、コスト関数と制約を与えた2 次計画法問題として求解を行う。コスト関数は以下の4 項目に重みづけ加算した値が与えられる。

1. 目標重心復元力追従コスト関数
   重心位置から見た足先位置を $[\boldsymbol{r}_l \in \mathbb{R}^3, \boldsymbol{r}_r \in \mathbb{R}^3]$ とし、さらに各足先姿勢行列を $[\boldsymbol{R}_l, \boldsymbol{R}_r]$ と表すと、足先復元力の重心位置についての変換は以下の式で示される。

$$\boldsymbol{w}_{com}^{res} = \begin{bmatrix} \boldsymbol{R}_l & O \\ [\boldsymbol{r}_l \times]\boldsymbol{R}_l & \boldsymbol{R}_l \end{bmatrix} \boldsymbol{w}_{footl}^{d} + \begin{bmatrix} \boldsymbol{R}_r & O \\ [\boldsymbol{r}_r \times]\boldsymbol{R}_r & \boldsymbol{R}_r \end{bmatrix} \boldsymbol{w}_{footr}^{d} \tag{5.57}$$

なお、$[\boldsymbol{r}_l\times]$ は外積演算と等価な歪対称行列を表す。目標重心復元力追従コスト関数は以下となる。

$$cost = ||\boldsymbol{w}^d_{com} - \boldsymbol{w}^{res}_{com}||^2 \tag{5.58}$$

2. 目標足先復元力追従コスト関数
   目標足先復元力について、
$$cost = ||\boldsymbol{w}^d_{*} - \boldsymbol{w}_{*}||^2 \tag{5.59}$$

3. 目標 ZMP 追従コスト関数
   目標 ZMP$\boldsymbol{p}^{ref} \in \mathbb{R}^2$ と、目標復元力によって修正される実現 ZMP$\boldsymbol{p}^{mod} \in \mathbb{R}^2$ について、
$$cost = ||\boldsymbol{p}^{ref} - \boldsymbol{p}^{mod}||^2 \tag{5.60}$$

4. 目標足先力コスト関数
   出力される足先力が過大とならないためのコスト制約であり、重み係数 $\boldsymbol{w}_l \in \mathbb{R}^3$、$\boldsymbol{w}_r \in \mathbb{R}^3$ を用いて、
$$cost = ||\boldsymbol{w}_l \boldsymbol{w}^{\mathrm{T}}_{footl}||^2 + ||\boldsymbol{w}_r \boldsymbol{w}^{\mathrm{T}}_{footr}||^2 \tag{5.61}$$

また、制約条件は以下の2 条件が与えられる。

1. 足裏摩擦力制約
   目標足先復元力によって、足裏滑りが生じないための制約条件となる。鈴木らのトルクベーススタビライザでは摩擦円錐モデルに基づいた摩擦制約式は用いておらず、x 軸と y 軸で独立した摩擦制約条件を課している。

2. 足裏圧力中心領域制約
   $[\boldsymbol{w}_{footl}, \boldsymbol{w}_{footr}]$ から計算されるそれぞれの脚の足裏圧力中心が、足裏許容領域内に含まれることを保証するための制約条件となる。

関節トルク変換

最適化計算によって得られた目標足先力 $[\boldsymbol{w}_{footl}, \boldsymbol{w}_{footr}]$ から、最終的な目標関節トルク $\boldsymbol{\tau}^{ref}_i$ を計算する。この目標関節トルクはあくまでも姿勢誤差修正のための復元力を働かせる関節トルクであり、目標関節軌道に追従するための関節トルク値ではないことに留意する必要がある。目標関節軌道に追従するための関節トルク制御については、5.4 節にて述べた中間層制御システムにて逆動力学演算によって行うこととなる。

5.12 即応的反射行動制御系全体構成

前節までにて述べた上位層制御システム内の実時間歩行動作軌道生成器群と中間層制御システム・下層モータ制御システムが統合された歩行動作制御系について、その制御フローを図 5.31 に示す。上位層制御システム内の制御処理は基本的に hrpsys 内のコンポーネントとして実装されており、これらの実行コンテキストは 500Hz で周期実行される 1 つのコンテキストにシリアライズされる。まず、図 5.31 内の Foot Step Generator にて歩行指令を元に支持脚歩容列が生成され、次に目標 ZMP 差分列 Δp^{ref}_i が生成される。また、実機から取得されるセンサ情報を元に絶対姿勢推定器 (Absolute Pose Estimator) 内にて、現在の推定支持脚 (ZMP) 位置、体幹位置、速度 $[p_{step}, x_{step}, \dot{x}_{step}]$ が計算され、LIPM 重心軌道生成器 (LIPM Trajectory Generator) に渡される。ここでは、Δp^{ref}_i 及び $[p_{step}, x_{step}, \dot{x}_{step}]$ を入力として、Singular-LQR ベースの予見制御器によって目標となる重心軌道 ${p^{com}_i}_{i=1,\ldots,n}$ 及び脚軌道 ${p^{footl}_i}_{i=1,\ldots,n}, {p^{footr}_i}_{i=1,\ldots,n}$ を生成する。続いて多リンクモデル ZMP 誤差補償器 (Multi-Body Dynamics Model) にて、実 ZMP が目標 ZMP に追従するよう体幹位置 $\boldsymbol{p}^{body}_i$ を修正する。また、逆運動学計算によって目標関節角度 $\boldsymbol{\theta}^{ref}_i$ を生成する。さらに、外乱による姿勢誤差を修正するための姿勢安定化制御器 (Torque-Based Stabilizer) にて、姿勢を復元するための足裏反力を発生させる目標関節トルク $\boldsymbol{\tau}^{ref}_i$ を生成する。また、オンラインでの歩容生成、バランス

制御を行わない腕に関しては、図中に示す関節角度補間器から制御周期に同期して直接関節角度が出力される。

ここまでが上位層における制御フローであり、中間層制御システムには制御指令値として目標関節角度と関節トルク $[\boldsymbol{\theta}_i^{ref}, \boldsymbol{\tau}_i^{ref}]$ がサブシステム間通信によって送信される。中間層制御システム内では、送信されてきた目標関節角度と、センサから計算される現在のロボットの状態を元に逆動力学演算とアクチュエータモデル補償計算によって、関節角度軌道追従のための目標関節トルクを生成する。この演算は 2500Hz の周期で実行される。この目標関節トルクと、上位層から送信される姿勢安定化のための関節トルク $\boldsymbol{\tau}_i^{ref}$ を加算した値が、トルクサーボ制御器 (Torque-Based Servo Controller) の入力として渡される。トルクサーボ制御器は入力された目標トルク指令値と下層にて推定されたトルク値及び目標関節角度と実関節角度を元に、2 自由度制御ベースの剛性制御によって目標トルクを発揮するための目標 Q 軸電流値 $\boldsymbol{q}_i^{ref}$ を生成し、下層へと通信する。この処理は 5000Hz で周期実行される。また、下層から送信される各種センサデータもこの周期に同期して行われる。

下層のモータ制御基板では、$\boldsymbol{q}_i^{ref}$ に追従するよう FPGA 内にハードウェアロジックとして実装された電流制御器によって出力電圧を決定し、最終的にベクトル制御器で各相出力が決定され、3 相インバータ出力となる。また、D-RMTP やソフトウェアプロセッサコアが搭載されている場合には、下層においてサーボ制御を行うことも可能である。

以上が歩行動作生成系の制御フローであり、目標歩行指令から最終的な 3 相インバータの出力として決定されるまでの一連の処理内容となる。階層型分散型制御システムとして、処理が分散化されており、下層に降りるに従って実行周期、並列実行処理数が増加する構成となっている。また、上位層と中間層の間で送受信されるデータは関節空間データとなっており、既存システムと互換のフォーマットを維持している。この上位層システムは歩行動作生成系のみの構成であるが、実運用上はタスク実行のためのその他処理がさらに積み上げられていくことになる。その場合でも、外力に対する応答を行うための主処理となるトルク制御系については、中間層制御システムとして完全に独立しているため、上位層への処理の追加による実時間性能への影響を受けない。本研究では上位層内での処理として実装されるトルクベーススタビライザによる姿勢安定化制御についても、中間層制御システム側で実行することによって、実時間性能の安定化、上位層異常時のバランス制御バックアップ等が期待できるため、2 次計画法の求解を組み込みシステムにて利用可能なレベルまで軽量化することは重要な課題となる。

5.13 脚型ロボットにおける体内制御システム構成例

本節では前節までに述べた上中下の各制御層について、実際のデバイス接続構成の例を示していく。光アクティブコネクタを利用した体内通信系のトポロジ構成方法はいくつか存在しており、その中で実際に利用し得るライントポロジ、リングトポロジ、ツリー(スター) トポロジの3 構成について以下に述べる。なお、システム構成は次章にて実験に使用する脚型ロボット L1(A.2 節) を使用することを前提に述べていく。

L1 では全 16 個のモータが搭載されており、下位層制御モータドライバも 16 ノードが接続される。また、L1 では中間層制御システムは右脚と左脚をそれぞれ制御分担するために2 ノード構成となっている。これらのサブシステムについて、脚の身体的構成に合わせたライン・リング・ツリー(スター) それぞれのトポロジ接続構成による配線図を図 5.32 、図 5.33 、図 5.34 に示す。なお、表 5.4 は上記図中で使用されている配線の説明を示している。

5.13.1 ライントポロジ接続構成

図 5.32 に示す構成では、左右各脚に搭載される RMTP-02D 基板を順にディジーチェーン接続による物理的なライントポロジで接続している。この構成では、各ノード間を接続する光アクティブコネクタのファイバ長を短くすることが可能であり、配線作業も容易である。ファイバ長の種類も骨格内接続と骨格間接続の2 種で基本的に十分である。また、パケットのルーティングについても、論理トポロジをリングとすることで一意的に決定することが可能であるため通信のオーバーヘッドは小さい。実際には、ただし、接続されているデバイスの一つでも通信機構に障害が発生した場合に、ネットワークが分断されてしまうという欠点が存在する。光アクティブコネクタと FPGA による通信処理の構成はハードウェア的に頑健な構成であるため、障害の発生確率は低いが、実運用を目指す上では適切でない構成となっている。

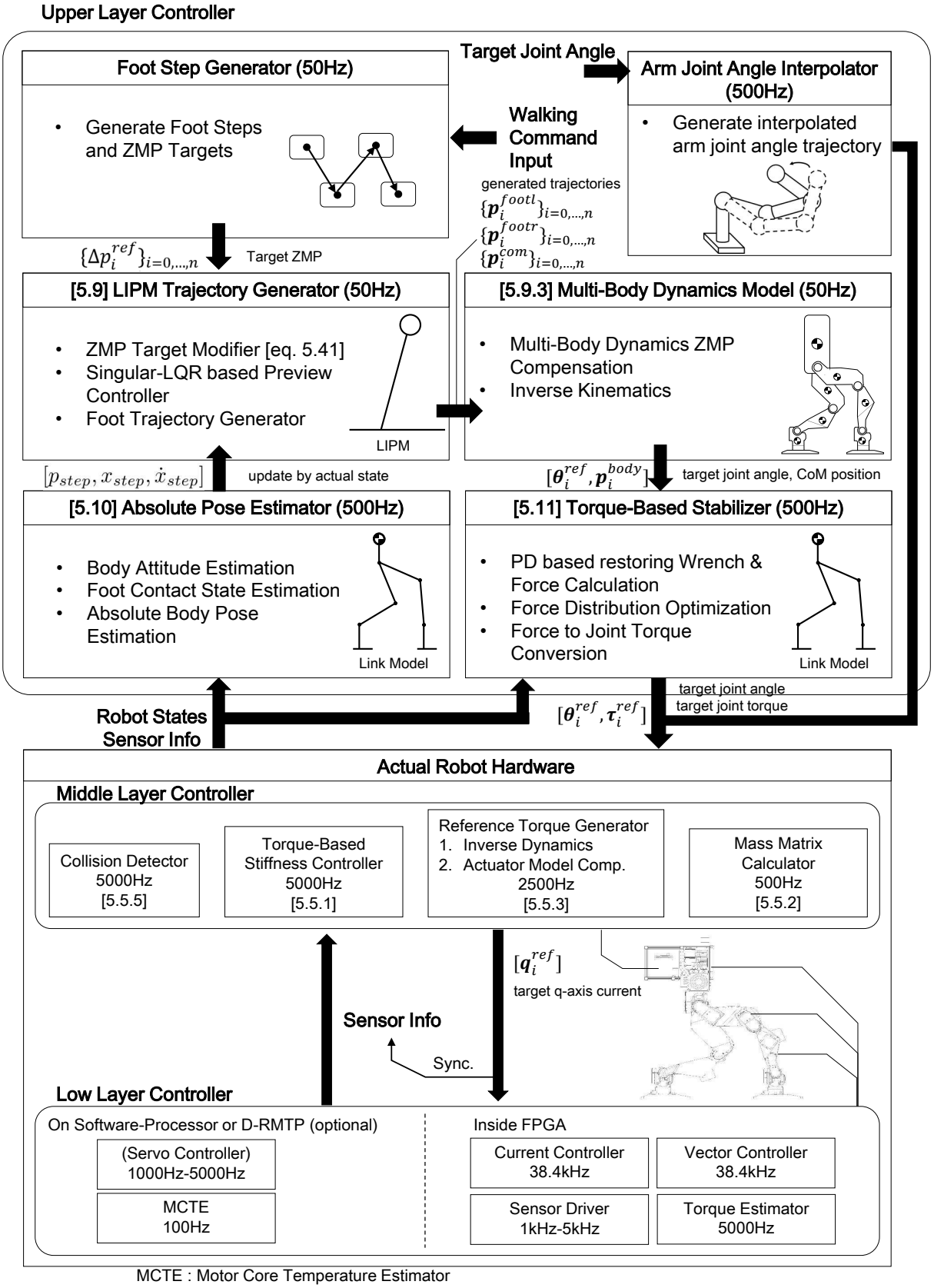


図 5.31: Overall control flow of walking pattern generation.

表 5.4: Explanation of indexes in Fig.5.32

index	Physical Layer	Bit Rate	Duplex
①	EtherCAT	100Mbps	Full duplex
②	active optical fiber link	2,500Mbps	Full duplex
③	RS422	4Mbps	Simplex
④	Three phase power line	-	-
⑤	Isolated RS422	1Mbps	Full duplex
⑥	Isolated I^2C	1Mbps	half duplex
⑦	USB	MAX 480Mbps (High-Speed Mode)	half duplex
⑧	Ethernet	MAX 1000Mbps	full duplex

5.13.2 リングトポロジ接続構成

図 5.33 に示す構成では、ライントポロジ接続構成で末端ノードとなる足先の RMTP-02D 基板について、図中に点線で示す中間層制御システムと直接接続する経路を追加することで、リングトポロジを実現している。この構成では、足先から腰部の中間層制御システムまで光アクティブコネクタを配線する必要があり、製品の標準規格から外れた特注品を発注する必要が生じるためやや入手性において難が有る。実際には、リング接続構成を工夫することで、ライントポロジ構成と同じケーブル種で対応は可能であるため、問題とはならないが、変則的な接続順序になるためアセンブリ時等に注意が必要となる。ルーティングについては一方向へと流すのみで良く、通信のオーバーヘッドは小さい。通信先までの物理的なホップ数はライントポロジ構成と比較して期待値で小さくなるため、通信遅延の改善も可能である。また、1 つのデバイスの通信障害までであれば、その他のノードの通信に関しては正常状態を維持可能である冗長性を有するため、信頼性の面においても実運用に適していると考えられる。

5.13.3 ツリー(スター)トポロジ接続構成

図 5.34 に示す構成では、中間層制御システムに搭載されている光アクティブコネクタのポート数に応じて、直接脚中の各 RMTP-02D 基板に配線を行う方法である。この通信系ではポート数は FPGA のトランシーバ数に応じて拡張することが可能であり、開発段階で必要となるポート数をあらかじめ搭載しておくことで、全ノード直接接続のスタートポロジ構成も実現可能である。全ノード分のポート数が無い場合でも、ツリー構成としてライントポロジやリングトポロジと組み合わせて利用することが可能である。この構成方法は、各ノードが小ホップ数で中間層制御システムと通信を行えるという利点があり、通信の低遅延化に利用可能である。また、1 つ以上の RMTP-02D 基板の障害に対してその他の通信系の正常状態は維持されるため冗長性は高いと言える。中間層制御システム側で通信障害が発生した場合にはこの限りではないが、従来の故障モードの頻度として高かったのはモータドライバ基板のオーバーヒートや過電圧、衝撃振動による電子素子の物理的な破損、体内摺動部をまたぐケーブルの不良が主であるため、中間層制御システムの通信障害による全身障害の発生確率は低いと考えられる。なお、脚のように中間層制御システムがマルチノード構成である場合には、リングトポロジ構成と組み合わせることで、他方の中間層制御ノードがダウンした場合においても、残りのノードでバックアップを働かせることは可能である。構成上、3 リンク以上を跨るファイバが存在してしまうため、保守性の点で不利が生じる。また、ファイバ長については、ノード数分だけ種別を用意する必要が生じる等、本構成はコストの面で他の構成と比較すると不利となるが、通信遅延性能や冗長性について重要視されるタスクである場合には、費用対効果が見合う構成になると言える。イーサネットや *Responsive Link* のようにフリートポロジ構成を可能とすることで、より冗長性の高いネットワークを構成することも考えられるが、現状ではルーティングによるオーバヘッドを削減することを優先し、一意的に経路が決定可能な物理トポロジのみの採用となっている。

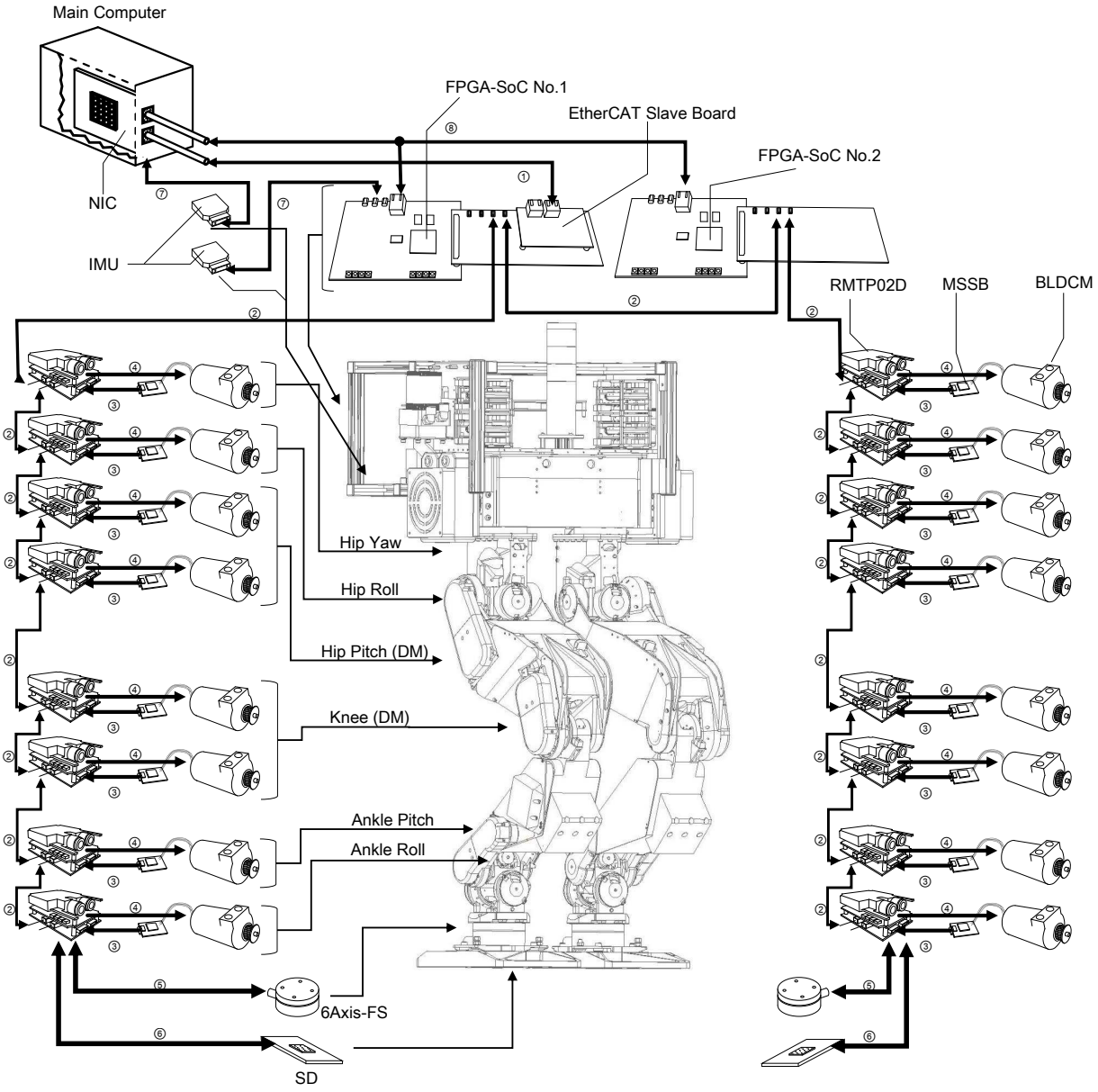


図 5.32: Connection schematics of L1 control system.

このツリー(スター) トポロジ接続構成が可能なのは、通信規格としてポート数の拡張に対応可能であり、また使用する通信ケーブルが従来のケーブルよりも十倍以上細く柔らかい光アクティブコネクタとなっているため、配線数が増えることによる空間圧迫が実用上無視可能なレベルであることが理由として挙げられる。

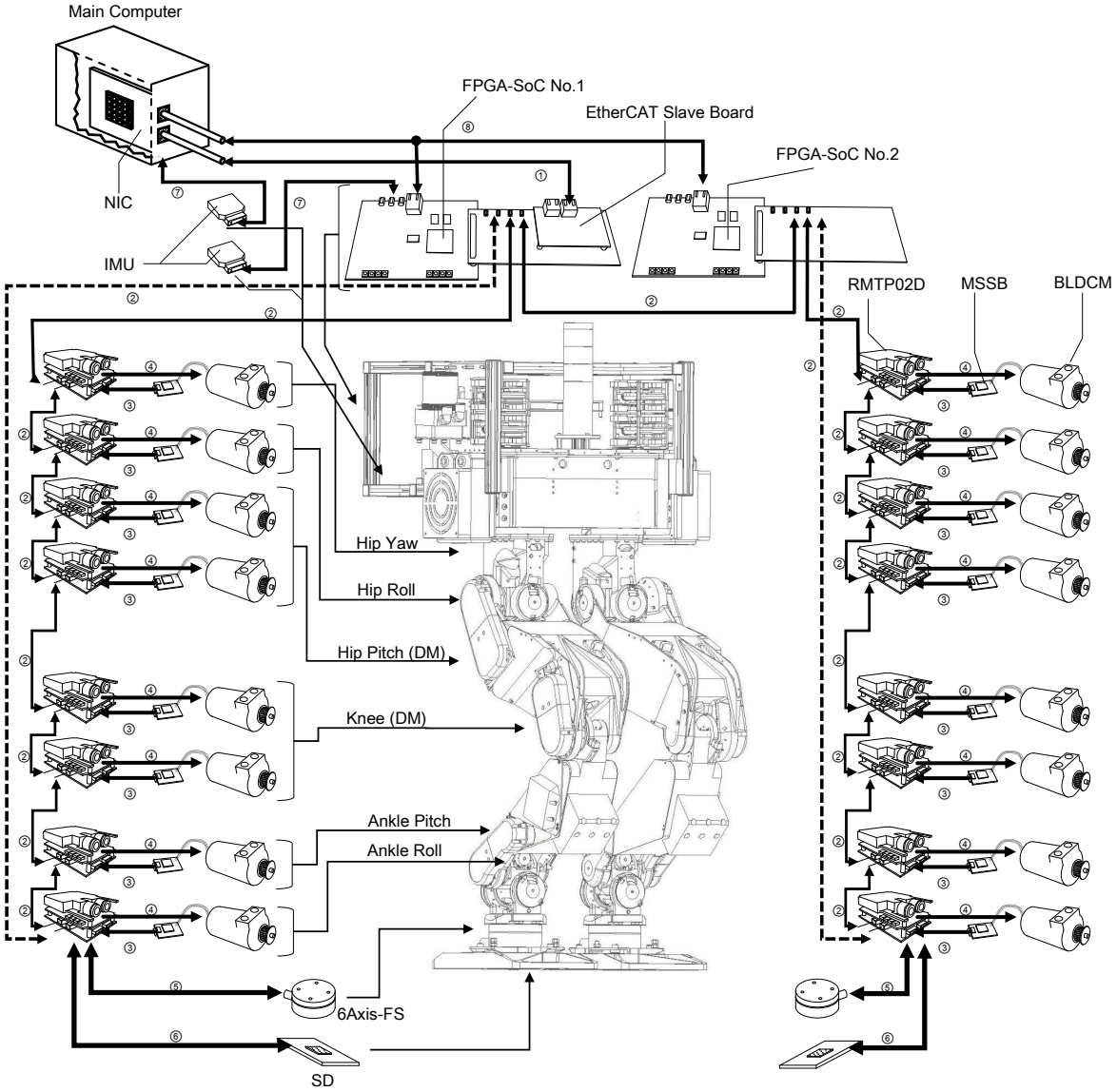


図 5.33: Physically ring topological connected system configuration.

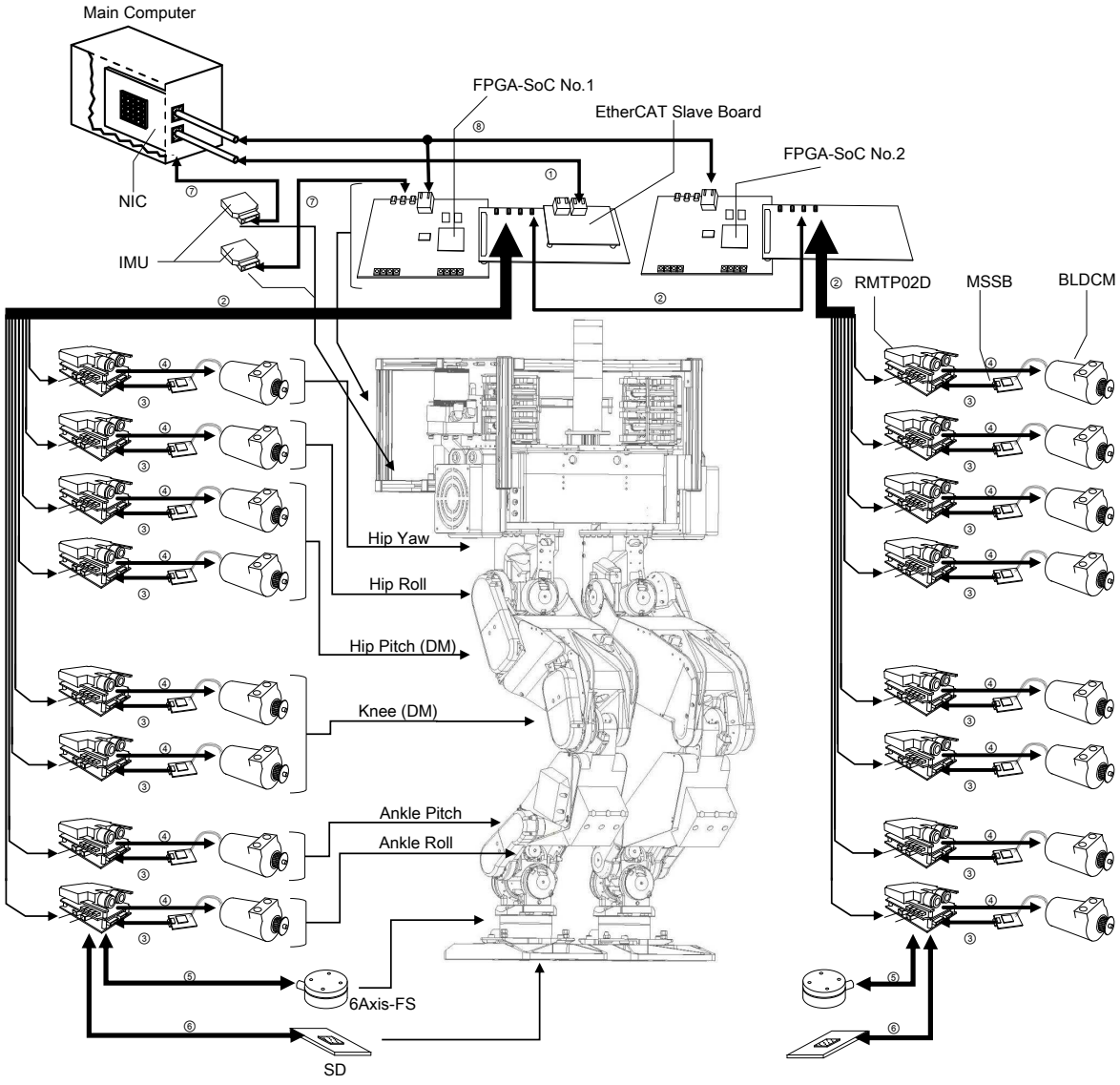


図 5.34: Physically tree (star) topological connected system configuration.

5.14 本章のまとめ

本章では、ロボットが衝撃入力のような外乱に対して生物のような柔らかさで対応するための即応的反射行動アーキテクチャについて導入を行った。これは即応的動作制御系と身体運動計画系の2 つで、センサデータ入力からアクチュエータへの指令へと変換されるまでの制御フローを最短系と、最適系の2 つで反射行動系として構成される。ハードウェア構成を三階層の制御システム構成とすることで、実時間性能要求に見合ったプロセッサに制御系の割り振りを行った。

このようなシステム構成はアクティブ光ファイバーリンクの提供する低遅延通信性能、ハードウェアレベル通信プロトコル処理、分散ノード間データ共有メモリ(ソフトウェア処理の簡略化)、高精度グローバルクロックによって成り立っており、アクティブ光ファイバーリンクによる体内通信系の有用性を示している。

即応的動作制御として、関節のサーボ制御則をトルクベースに変更しており、計算トルク法によるフィードフォワードトルク制御を二自由度制御系に与えることで外乱に対する即応的な緩和動作を実現している。また、上位層制御システムでは身体運動計画系の実装として、実時間歩行生成器を適用し、オンラインでの転倒回避動作を実現している。

第 6 章では、実際の腕型ロボット、脚型ロボットに本システムを適用し、衝撃吸収動作実験を行っていき、本章で提言する即応的反射行動アーキテクチャの効果を検証する。

第6章

ヒューマノイドロボットにおける衝撃外乱に対する高速応答行動実験

6.1 中間層制御システムを利用した腕ロボットの動作実験

第 5 章にて導入した中間層制御システムを実際の腕型ロボットの制御システムとして組込み、物体との衝突を伴う動作実験によって検証を行う。腕型ロボットでは身体運動計画系のプログラムは実行されていないため、即応的動作制御系による外力への応答のみが検証の対象となる。

6.1.1 実験構成

本節では A0-B spec を用いて、腕型ロボットとしての動作実験を行う。足平形状のエンドエフェクタの足裏面には、生物の皮膚の柔らかさを模すために厚さ 10mm の低反発ウレタンスポンジを貼り付けている。

A0-B spec で使用した上・中・下位システムそれぞれの構成を表 6.1 に示す。上位システムでは通常の Ethernet と EtherCAT 用で2 つのポートが必要となるが、オンボード搭載の標準 NIC は1 ポートのみであったため、USB 接続タイプの Ethernet アダプタを汎用の Ethernet 用 NIC として追加している。それ以外の点については、標準的な構成となっている。上・中・下位3 つのサブシステムは図 6.1 に示す配線図に従って接続されている。表 6.2 は図 6.1 中で使用されている配線の説明を示している。

本実験では上位システムは従来システム (5.2 節) と、EtherCAT インターフェースを除いて全く同一の構成、処理内容となっている。ただし、User 環境から Euslisp を介して中間層制御システム上の ROS ノード、RTM コンポーネントが提供するサービスポートを利用した制御パラメータ変更、制御モード変更、機能有効化コマンド等の非実時間処理用のユーティリティは新規に追加している。実験では Euslisp プログラム上で作成した腕のスイング動作(またはステップ動作) の関節角度シーケンスを hrpsys に入力し、そのまま制御サイクル毎に補間された目標関節角度が EtherCAT インターフェースを介して下位システムへと送信される。

表 6.1: Configuration of A0-B spec control system.

Upper Layer	# of Nodes	1
	CPU	Intel Core i7-2700K @ 3.50GHz 8threads (Hyper-Threading enabled)
	memory	8GB
	OS	Desktop Linux (Ubuntu)
	NIC 1	On board NIC, Realtek RTL8111 (for EtherCAT)
	NIC 2	USB2.0 Ethernet Adapter 100Mbps, ASIX AX88772 (for Ethernet)
Middle Layer	# of Nodes	1
	CPU	ARM Cortex-A9 Multi-Core @ 800MHz on Hardware Processing System (HPS) in FPGA
	Memory	512MB×2 (DDR3)
	OS	Linux (Xillinux)
	EtherCAT slave	1 (FB1111-0140)[5]
	EtherCAT bridge	1
	active optical fiber link port	2
Lower Layer	# of Nodes	6
	Hardware	RMTP-02D
	Sensor	Shock detector 1 on end-effector
		Six-Axis force sensor 1 on end-effector (WEF-6A1000-80-40-RCXTi2)[6]

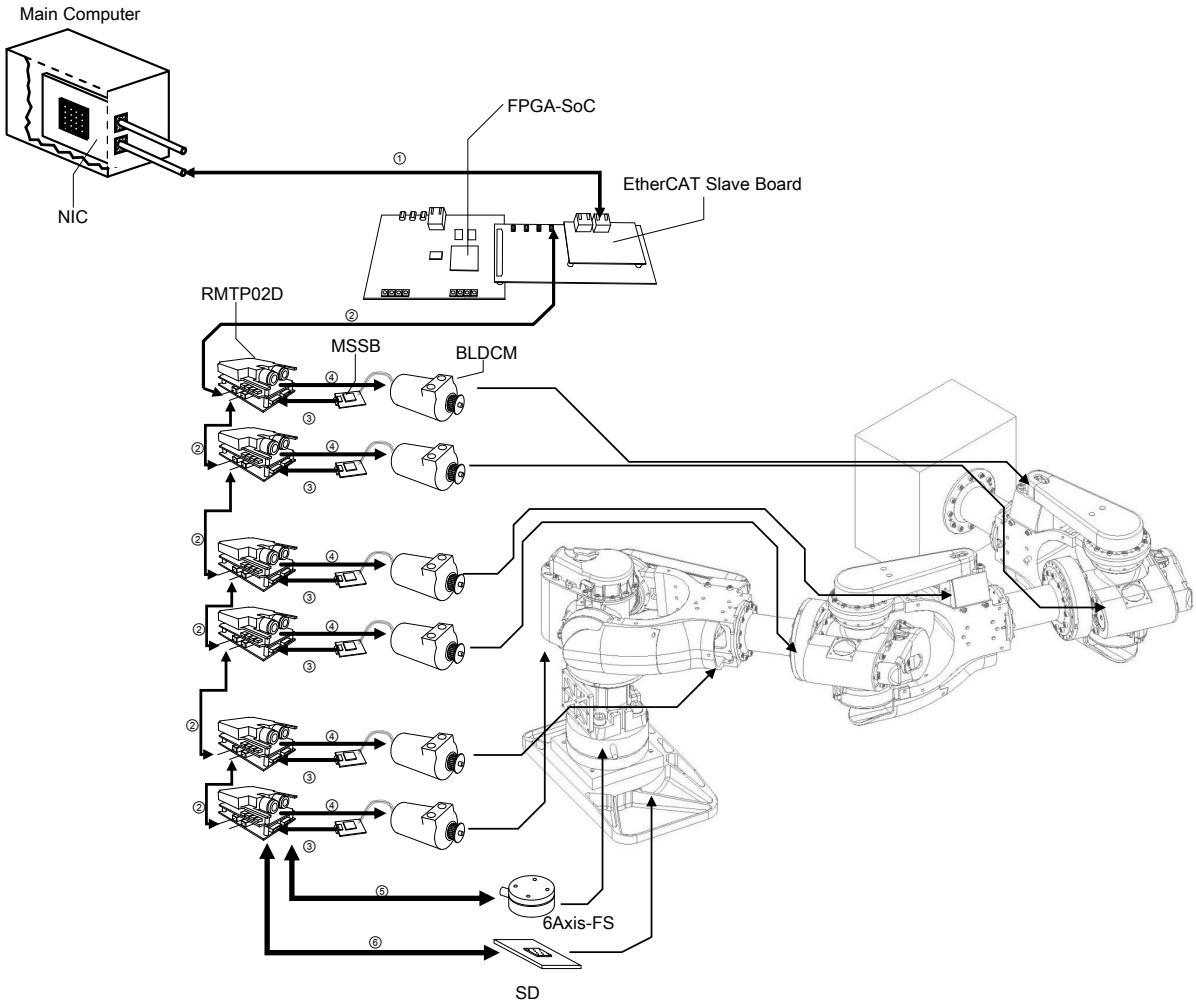


図 6.1: Connection schematics of A0-B spec control system.

表 6.2: Explanation of indexes in Fig.6.1

index	Physical Layer	Bit Rate	Duplex
①	EtherCAT	100Mbps	Full duplex
②	active optical fiber link	2,500Mbps	Full duplex
③	RS422	4Mbps	Simplex
④	Three phase power line	-	-
⑤	Isolated RS422	1Mbps	Full duplex
⑥	Isolated I^2C	1Mbps	half duplex

[5]www.beckhoff.com/english.asp?ethercat/fb1111_fb1122_fb1130.htm
[6]www.wacoh-tech.com/

6.1.2 スイング動作時の衝突実験

腕型ロボットに単純なスイング動作を9 回繰り返し行わせ、その軌道を確認する。特に、中間層制御システムを導入したことによってサーボ制御はトルクベースとなっているため、衝突による外乱に対してなじむ動作となることが期待される。本実験ではスイング動作をしているロボットに手を差し出して衝突させ、関節制御のなじみ性能について検証する。

関節角度換算で 20 ° 程度の移動となる運動軌道を Euslisp プログラムで生成し、hrpsys 経由でロボットへと送信する。本実験ではスイング時間 1000msec、600msec、400msec と三段階の速度設定をそれぞれ試した。図 6.2a 、図 6.3a 、図 6.4a にそれぞれ中間層制御システム上のデータロガーで記録されたスイング時間 1000msec、600msec、400msec 時の目標関節角度軌道を示す。

図 6.2 に実験で得られたデータの内、主要なものを示している。また、図 6.5 及び図 6.6 に動作実験時に撮影した動画像を示す。動作時の最大関節速度は 30-40[deg/sec] 程度となっている。この実験では、13 秒、15 秒、18 秒、22 秒、25 秒付近の計5 回、実験者が差し出した手とロボットアームを衝突させた。13 秒、15 秒、25 秒付近の衝突はスイングしている腕のエンドエフェクタに衝突するように手を差し出した (図 6.5 参照)。また、18 秒付近では第4 リンク(第3 関節と第4 関節の間) を掴んで動作を阻害 (図 6.6 参照) し、22 秒付近では第2 リンク(第1 関節と第2 関節の間) を抑えて動作を阻害した。図 6.2c に示す関節角度の目標値と実値の誤差を見てわかるように、衝突時に誤差が大きくなっていることからロボットアームは衝突した実験者の腕による外力になじんでいることが確認される。しかしながら、非衝突時においても誤差角度が5 ° 程度生じており、完全な目標角度追従が達成されているわけではなかった。特にエンドエフェクタに近い第4 関節では誤差が他と比較して大きい傾向が見られた。また、図 6.2f はエンドエフェクタに取り付けた6 軸力センサの出力であるが、エンドエフェクタと手の衝突時に発生している力は概ね 30N 未満程度であり、非常に軽い力でロボットアームのスイング動作を止められていることも確認できる。

スイング動作の速度を高めた実験についても、同様に図 6.3 、図 6.4 に示す。600msec でのスイング動作時は、12 秒、16 秒、20 秒、23 秒付近で衝突が発生している。400msec でのスイング動作時は、18 秒、20 秒、24 秒、26 秒付近で衝突が発生している。これらの実験においても、衝突後に関節角度誤差値が大きくなっており、実験者の手に対してなじみが生じていることが確認された。600msec 時の力センサデータ (図 6.3f) では、80-100N 程度の力が瞬間的に発生していることが確認される。また、400msec 時では 170N 前後であることも確認される。

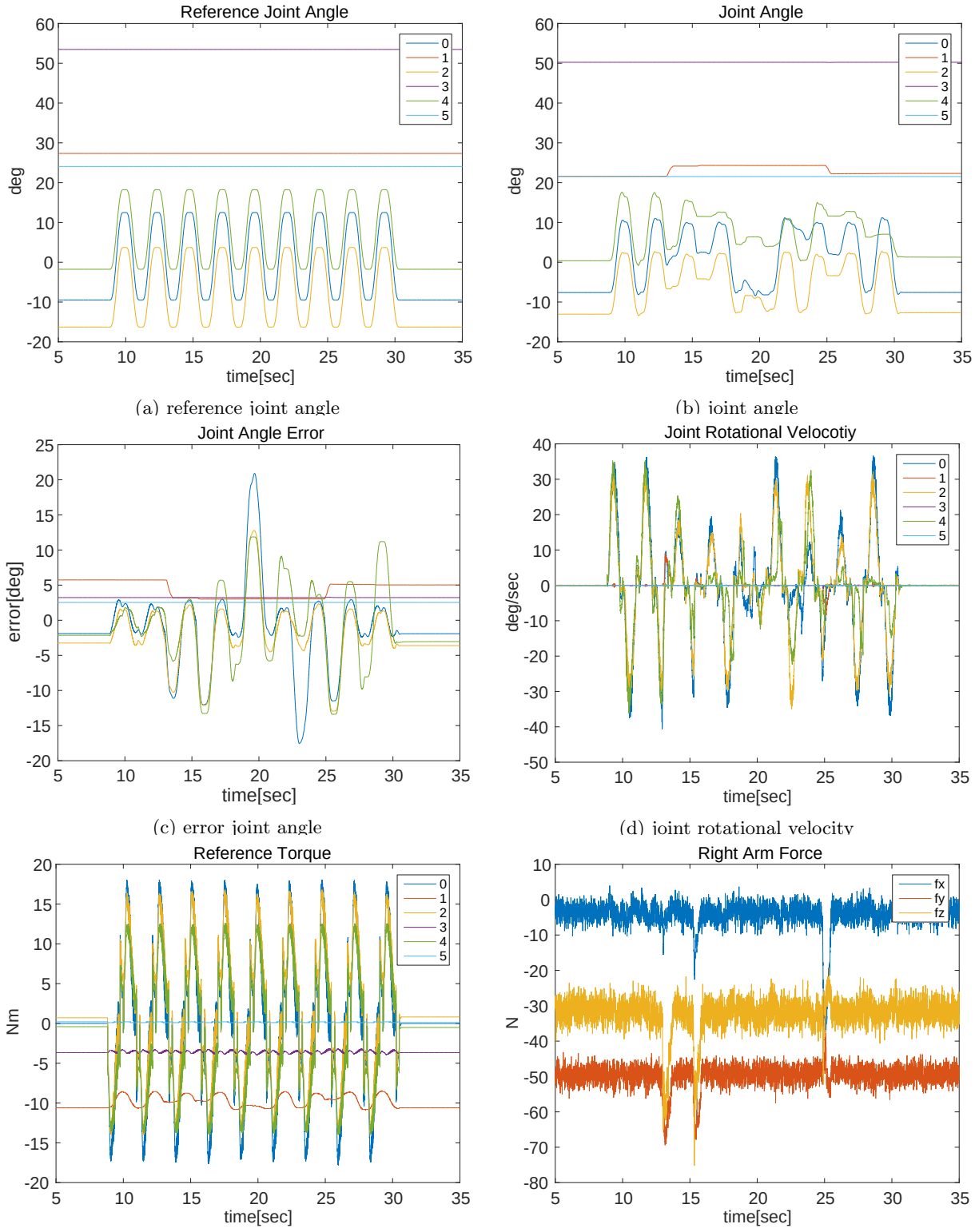
実感として、1000msec、600msec でスイングしているロボットアームとの衝突は反力はほとんど感じない、あるいは非常に軽量であるという感覚を得られた。400msec でのスイングでは高速での衝突となり、大きく拍手をする際に感じる程度の痛みは感じられるが、怪我をするような危険性を全く感じない程度であり、軽い力で簡単に押し返すことが可能な反力である。20 秒の付近で発生している衝突では実際に衝突後に手で押し返すことをしているが、関節角度誤差が他の衝突時と比べて特に大きく生じていることからもわかるように、容易に押し返しを行えていることが分かる。

6.1.3 スイング動作実験における衝突検出の検証

図 6.7 、図 6.8 、図 6.9 にスイング動作実験で得られた衝突検出に関連するデータログを示す。データはそれぞれ関節角度誤差、衝撃センサデータ値(ハイパスフィルタ処理済)、6 軸力センサデータ値、式 5.26 の σ_{col} 値の4 種となる。各々について、衝突検出性を検証する。

1. 関節角度誤差
   図 6.7a 、図 6.8a 、図 6.9a の3 種のスイング速度において、全ての記録で衝突· 非衝突を判別可能なレベルで誤差が生じていることがわかる。しかし、衝突後にも誤差は残っており、単純な閾値判定で衝突非衝突を判定することは困難である。同一の衝突過程においても関節の剛性パラメータの差異で誤差は大きく変わるため、衝突検出には他データの利用が必要になると考えられる。

2. 衝撃センサ
   図 6.7b 、図 6.8b では衝撃加速度は数 G 程度に収まっており、衝突を検知しているとは言い難い。図 6.9b に示す、高速動作時の衝撃加速度センサ値の場合、一回目及び4 回目の衝突時に 10G を超える大きな衝撃加速度を検出している。しかしながら、2 回目、3 回目については低速時の衝突過程と同様に衝撃加速度から衝突判別は困難であり、全検出が可能ではないことが確認された。



(a) reference joint angle

(b) joint angle

(c) error joint angle

(d) joint rotational velocity

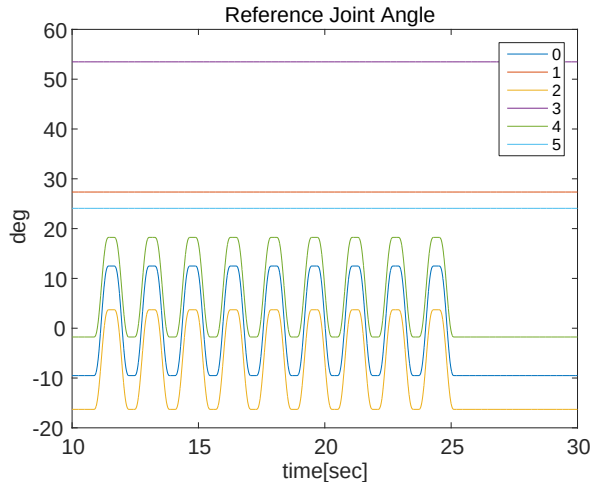
(e) reference torque

(f) six-axis force

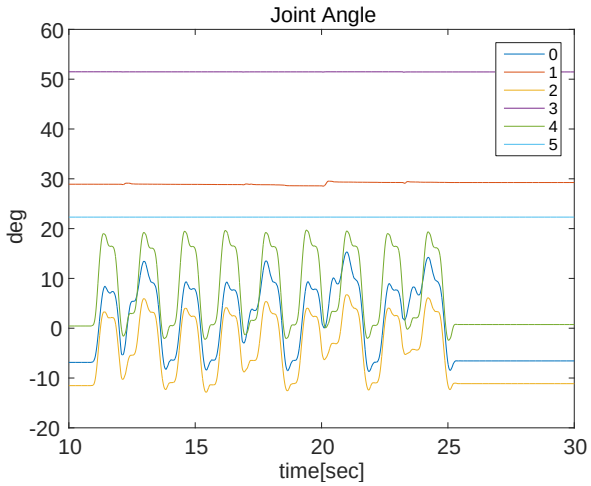
図 6.2: Logged data of swing motion test at 1000msec speed.

3. 6 軸力センサ
   衝突時の衝撃力を 3 種の速度それぞれで検出しており、閾値判定で容易に衝突を検出可能である。しかしながら、検出可能な衝突はエンドエフェクタとのものに限定され、リンク等の他の部位への衝突は検出不可能である。

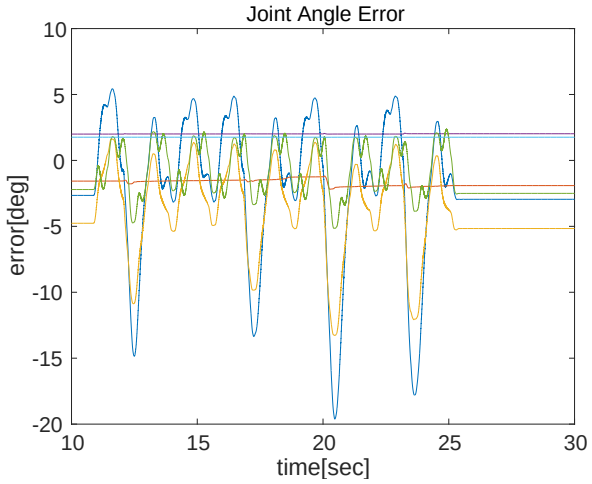
4. σ_{col} 値



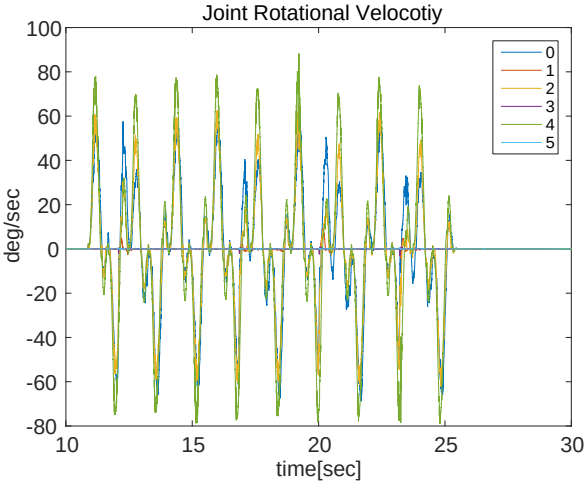
(a) reference joint angle



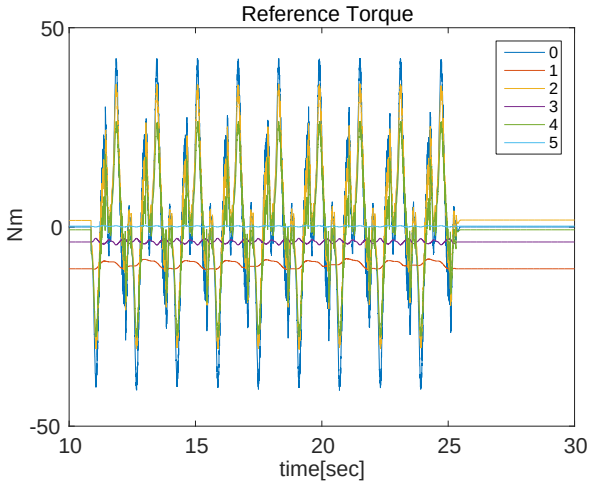
(b) joint angle



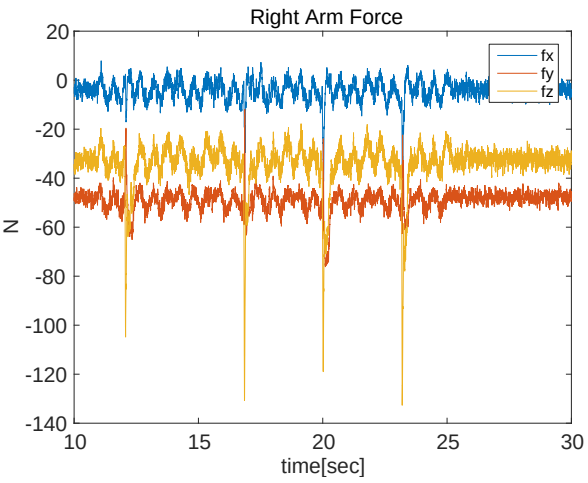
(c) error joint angle



(d) joint rotational velocity



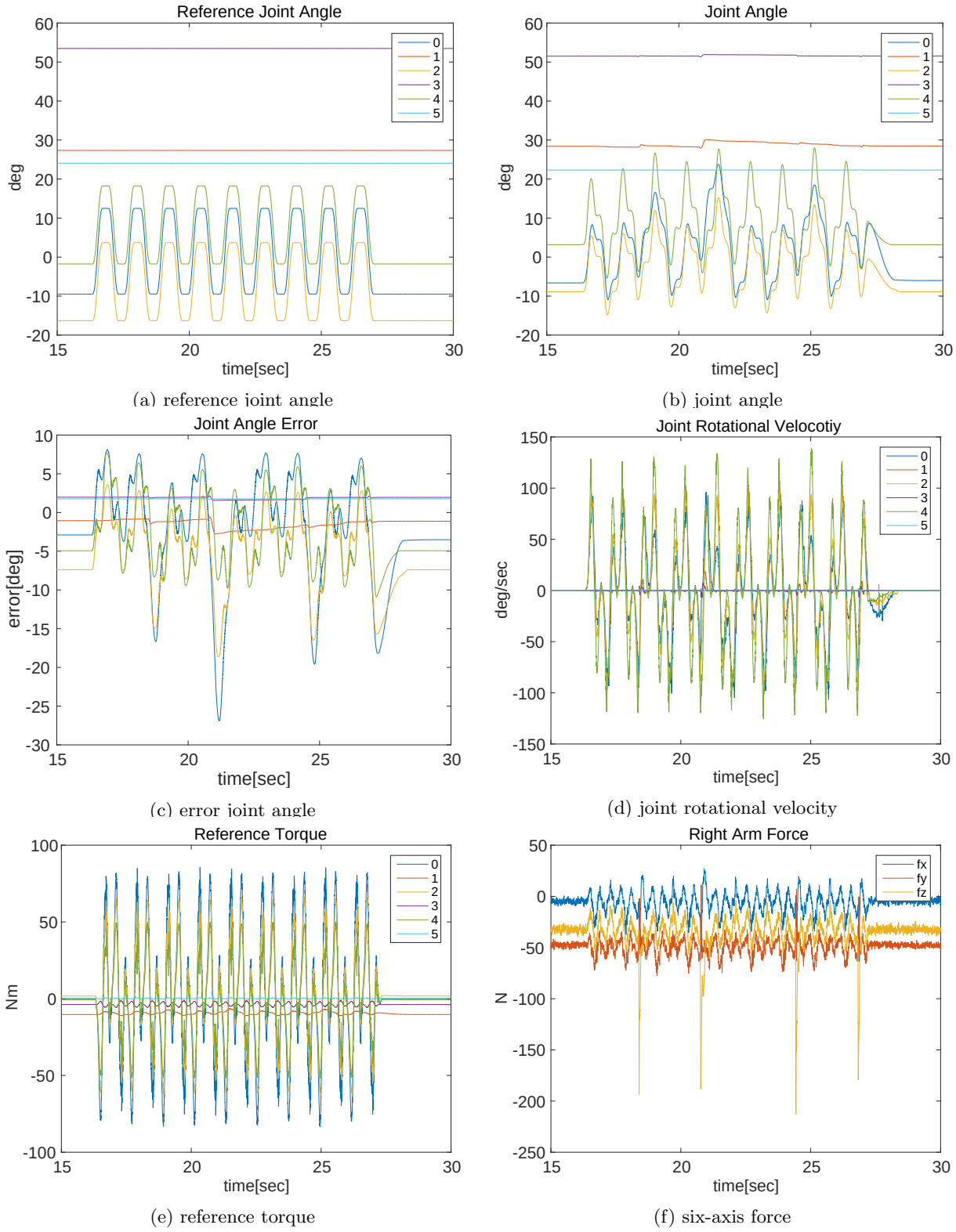
(e) reference torque



(f) six-axis force

図 6.3: Logged data of swing motion test at 600msec speed.

アクチュエータモデルや自機の動力学モデルの不完全性によって、動作中は非衝突過程においても運動量誤差が蓄積されてしまっている。比較的低速な図 6.7d と図 6.8d のデータに関しては、衝突時に大きく負方向に検出されていることが確認される。しかしながら、内側リンク部位との衝突が生じている図 6.7d の 18 秒、22 秒付近については、判別可能な値が出力されているとは言えず、6 軸力センサでの判別と同様の問題を抱えてしまっている。また、図 6.9d が示す高速動作時の値については、



(a) reference joint angle

(b) joint angle

(c) error joint angle

(d) joint rotational velocity

(e) reference torque

(f) six-axis force

図 6.4: Logged data of swing motion test at 400msec speed.

完全にモデル化誤差に埋もれてしまっており、判別は不可能となっている。

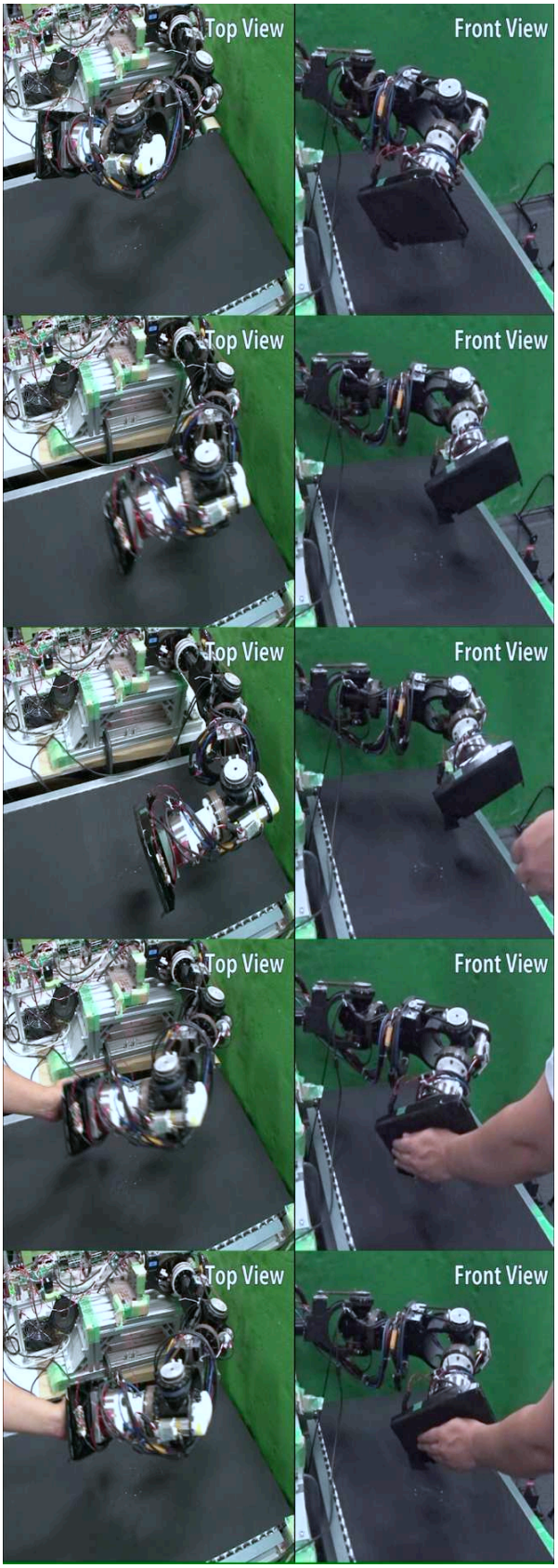


図 6.5: Collide with an end-effector of swinging A0-B spec arm by hand.

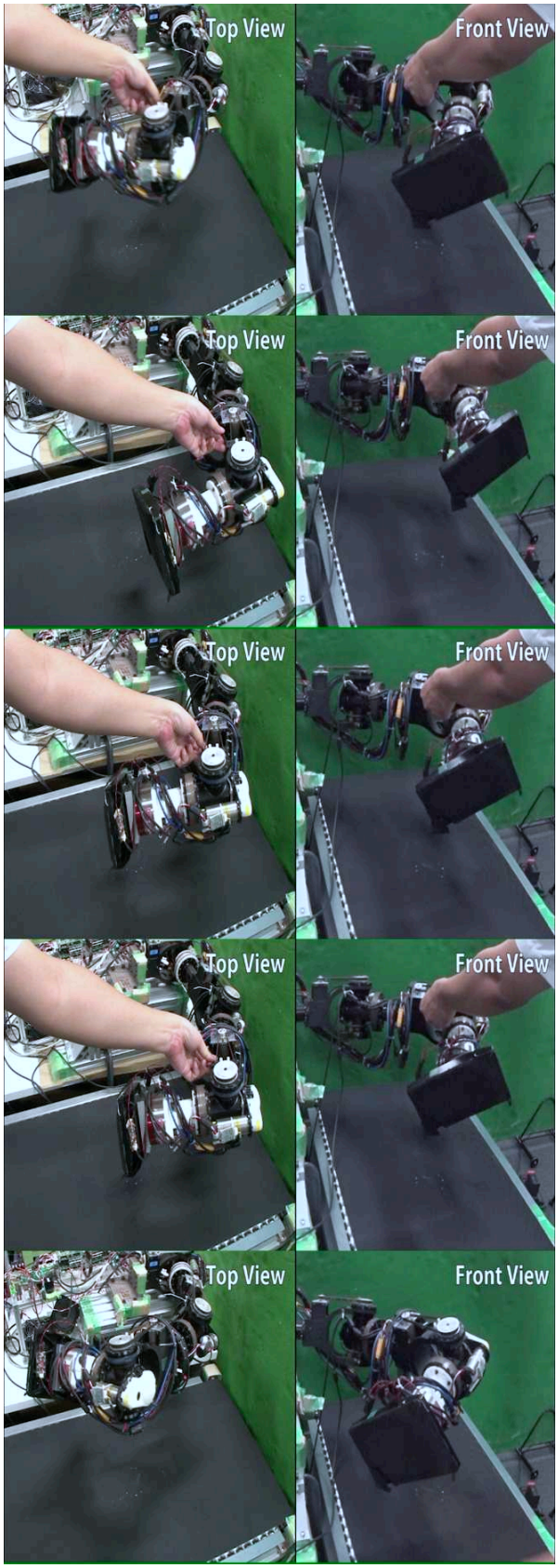
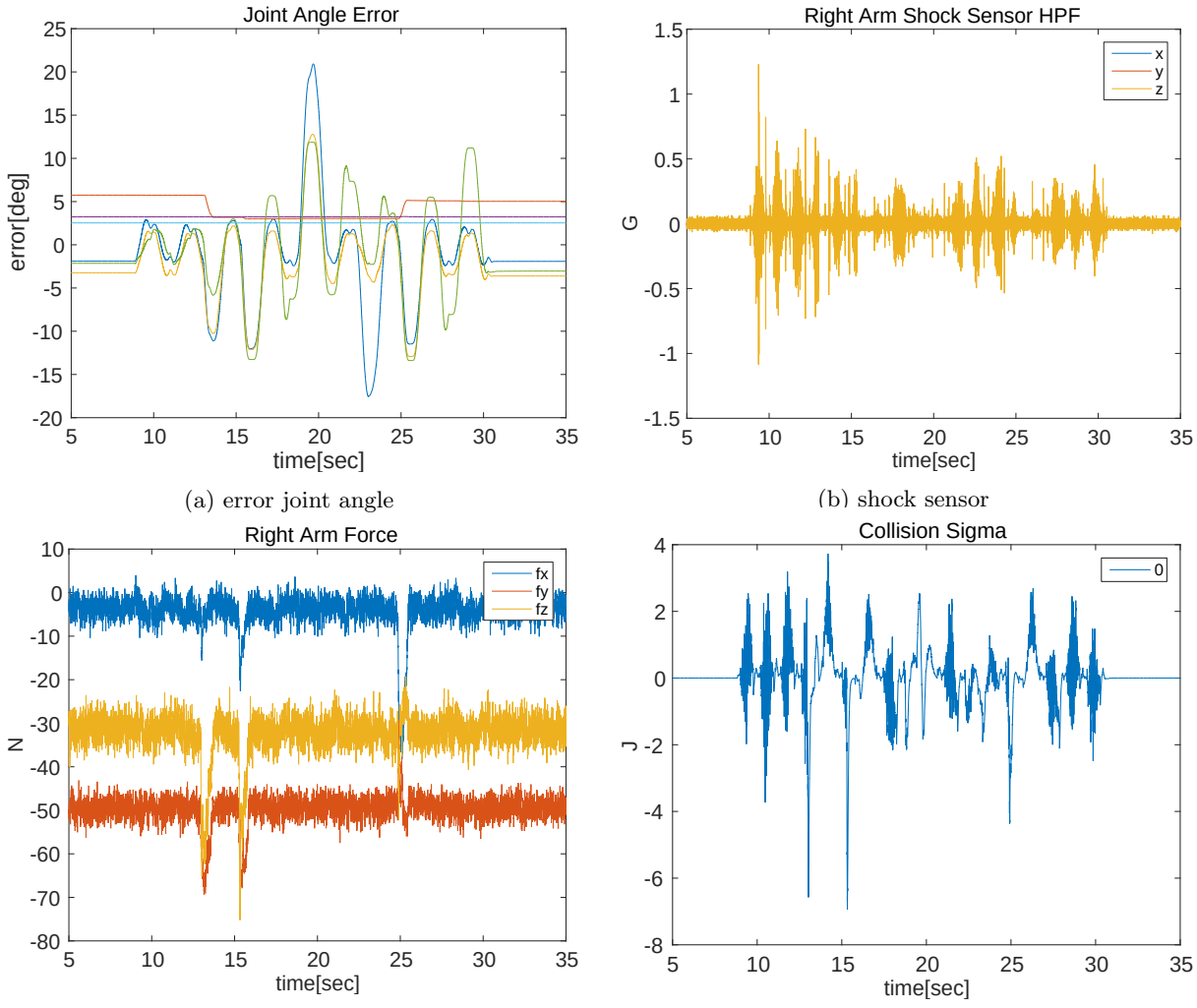


図 6.6: Collide with a link of swinging A0-B spec arm by hand.



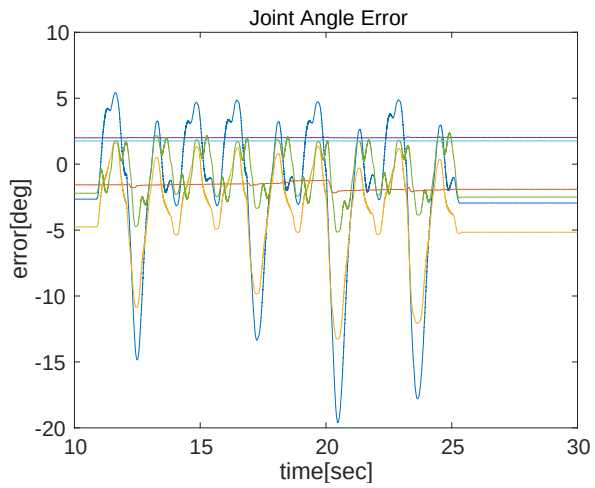
(a) error joint angle

(b) shock sensor

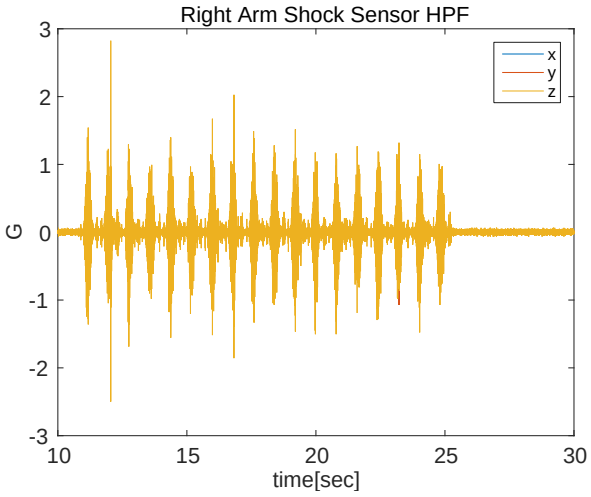
(c) six-axis force

(d) colligion sigma

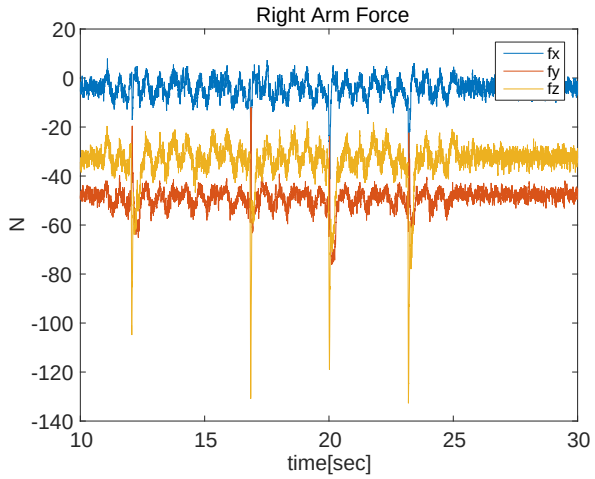
図 6.7: Colligion related data of swing motion test at 1000msec speed.



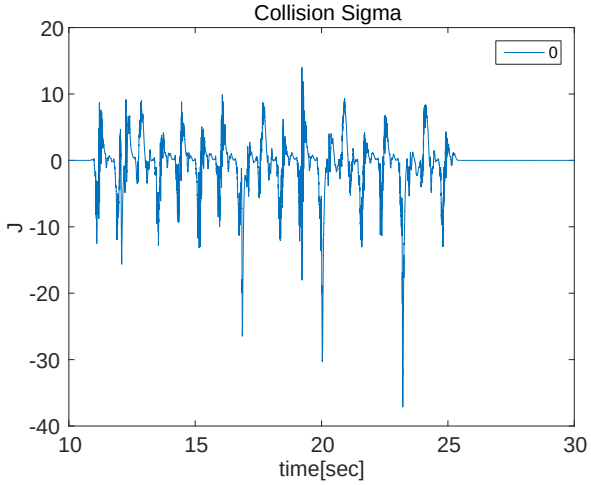
(a) error joint angle



(b) shock sensor

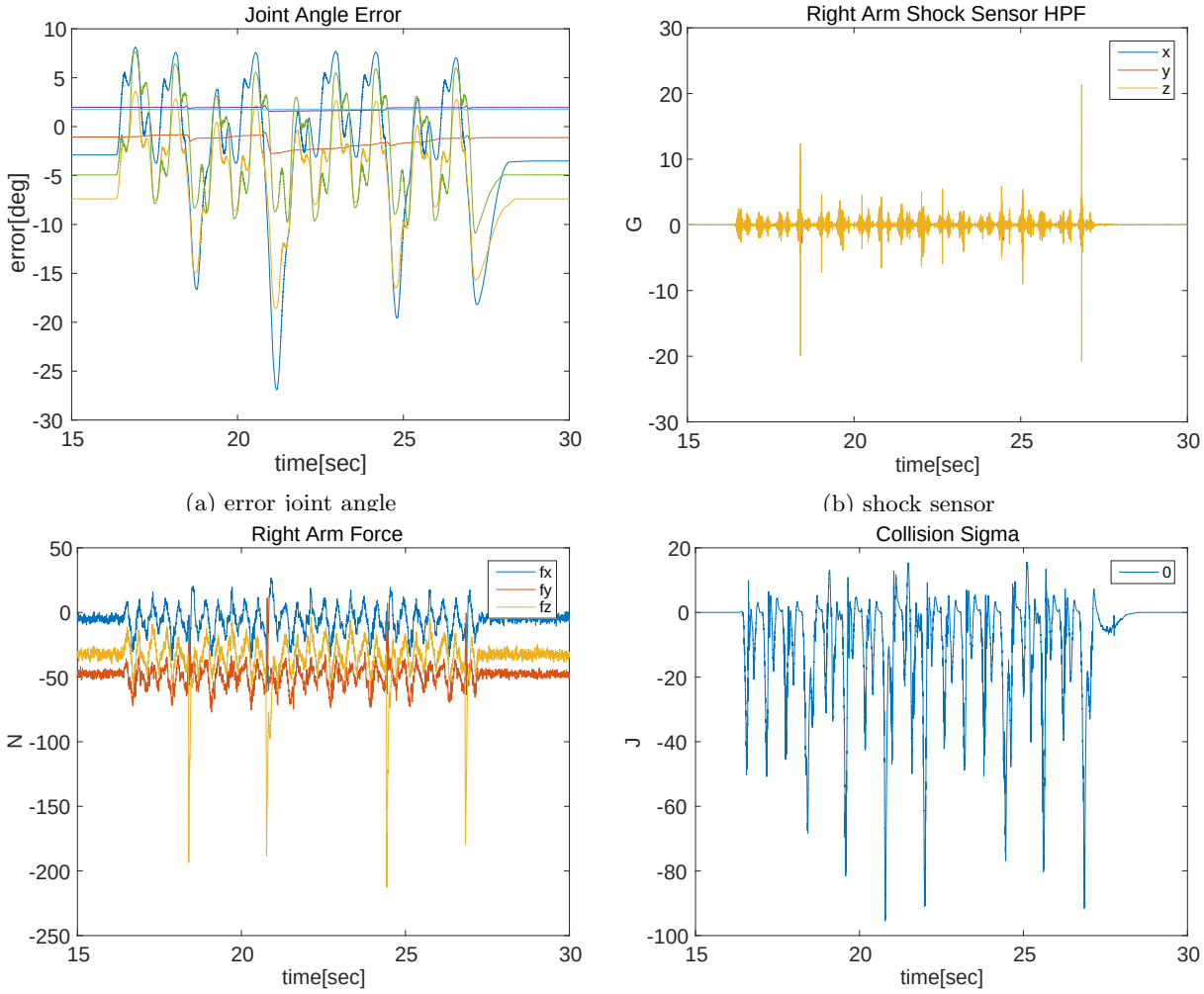


(c) six-axis force



(d) colligion sigma

図 6.8: Colligion related data of swing motion test at 600msec speed.



(a) error joint angle

(b) shock sensor

(c) six-axis force

(d) colligion sigma

図 6.9: Colligion related data of swing motion test at 400msec speed.

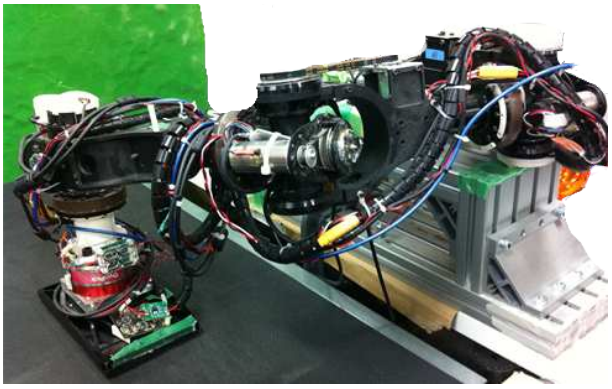


図 6.10: Leg style configuration of A0-B spec.

6.1.4 ステップ動作実験

脚型ロボットでの適用を見据えて、生成された歩容動作についても本制御システムで実行した際の柔軟性と動作速度・精度の達成度合いについて検証を行った。本実験では A0-B spec を図 6.10 に示すようにエンドエフェクタを脚構成に変更して実験を行った。

生成した歩容軌道は前方向 200mm、頂点高さ 63mm、ステップ時間 1000msec のサイクロイド曲線補間軌道となっている。A0-B spec のルートリンクは固定されているため、歩行時のバランス制御については考慮されていない。そのため、エンドエフェクタで発生する目標力・トルク値についてはそれぞれ0 としている。トルク制御により歩行軌道と実路面形状に差異があってもなじみながら歩行軌道を実行することが期待される。歩行動作はトレッドミル上で行ったため、歩行動作方向については接地後にエンドエフェクタに従動することが可能となっている。

本実験ではトレッドミル上で歩行動作を行わせ、

1. 平面上の歩行動作
2. 厚さ 20mm 程度のアルミ板の踏破動作
3. コンクリートブロックと足部の衝突が生じる躓き動作

の3 パターンについて行った。1 と 2 については、歩容は5 歩分生成し、3 については3 歩分の歩容について生成を行い、一歩目でブロックに躓くような配置とした。いずれのパターンについても元となる歩行動作関節軌道は同一のものとなっているが、トルク制御によってトレッドミル表面への足部のなじみ、トレッドミル従動方向へのなじみ、障害物との衝突力の緩和が実現されるか検証を行う。

図 6.11 に平面上の歩行動作を、図 6.12 に障害物を踏ませた場合について、及び図 6.13 にコンクリートブロックと衝突させた場合についての撮影動画からの切り出し画像を示す。また、図 6.14 、図 6.15 、図 6.16 にそれぞれ主要なデータについてのデータログを示す。

結果、1 と 2 それぞれについて、前方への歩行動作は実行され、トレッドミルと相対的に前方へ前進することが確認された。また、3 についてはコンクリートブロックを無理に押し込むような動作とはならなかった。

次に、衝突検出について図 6.17 、図 6.18 、図 6.19 に関連するデータ（ 関節角度誤差、衝撃センサ値、6 軸力センサ値、σ_{coll} 値）の4 種についてグラフを示す。1 と 2 については、トレッドミル平面との接地による関節角度誤差や力が検出されており、平面でのステップとアルミブロック上のステップとの区別はこれらの指数では判定が困難となっている。一方、図 6.13 のコンクリートブロックでの躓き動作については、衝撃センサ（ 図 6.19b ）が 10G を超える数値で反応しており、特に躓きについての検出に効果的であることが示されている。この躓き動作の際、6 軸力センサ（ 図 6.19c ）の歩行動作方向（ y 軸方向）について 100N 程度のパルス状の撃力が加わっていることが確認される。トルク制御により撃力はある程度緩和されていると考えられるが、撃力をさらに緩和するためには衝撃センサによる衝突の検出に合わせて即応的に衝突回避行動を生成し直すことが重要であることがわかる。

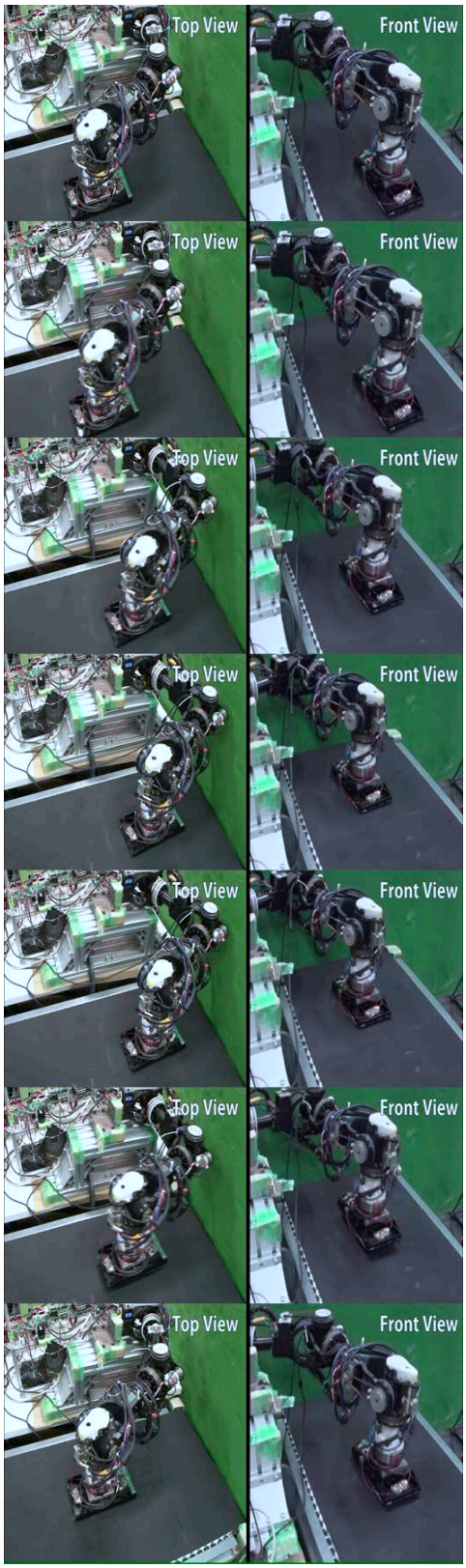


図 6.11: Stepping motion of A0-B spec on a treadmill.

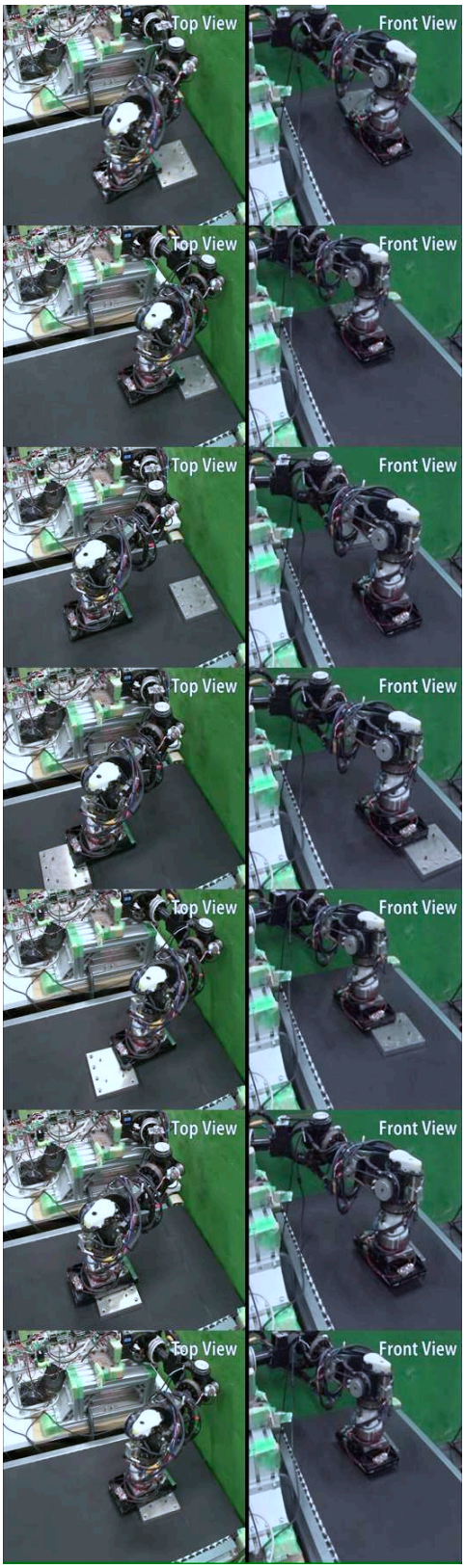


図 6.12: Stepping motion of A0-B spec stamping an obstacle.

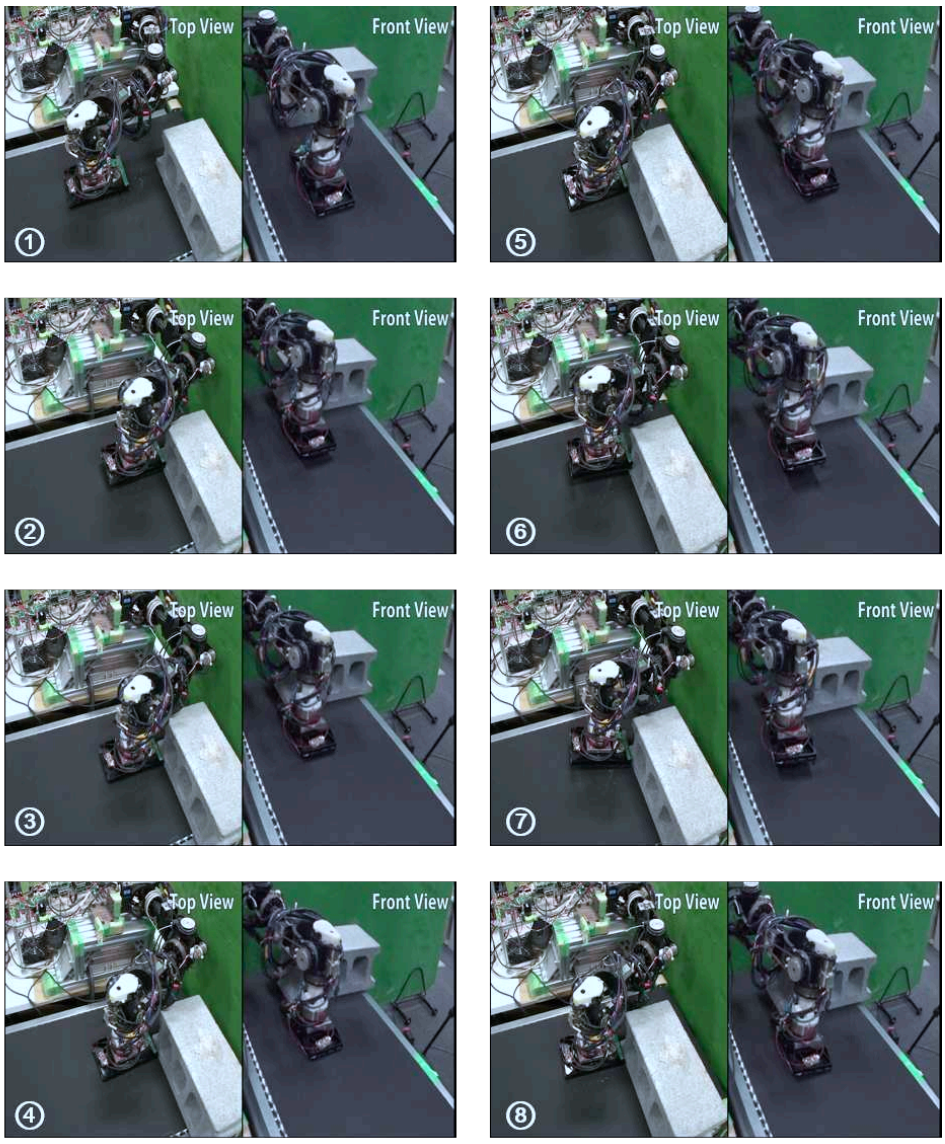
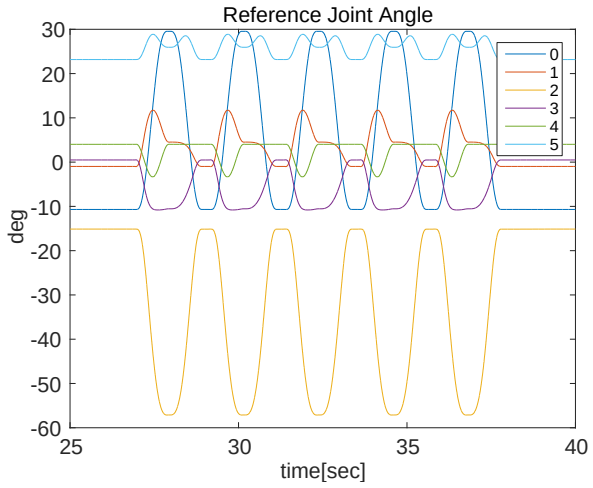
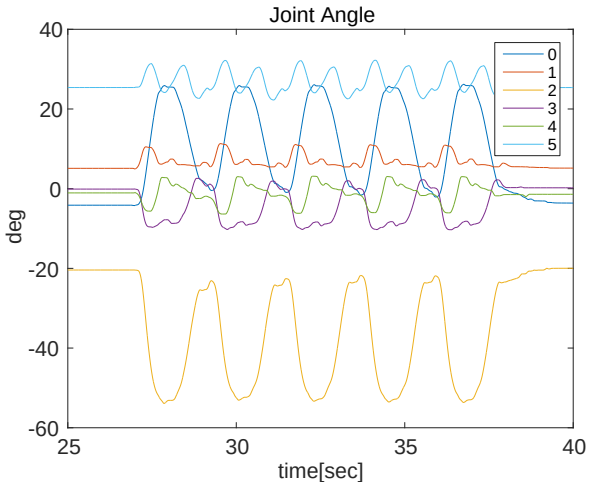


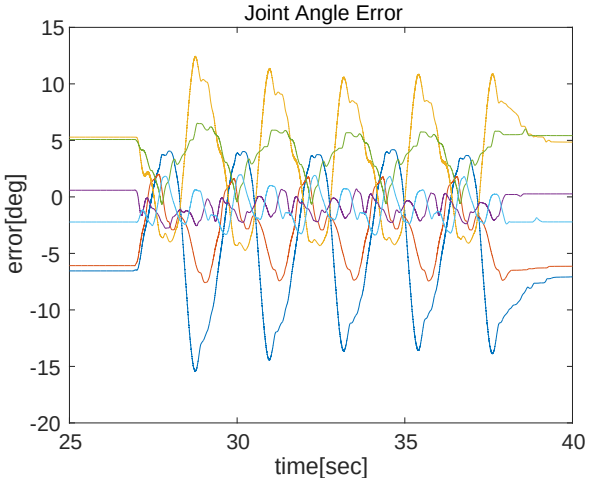
図 6.13: Collision with a block while stepping motion.



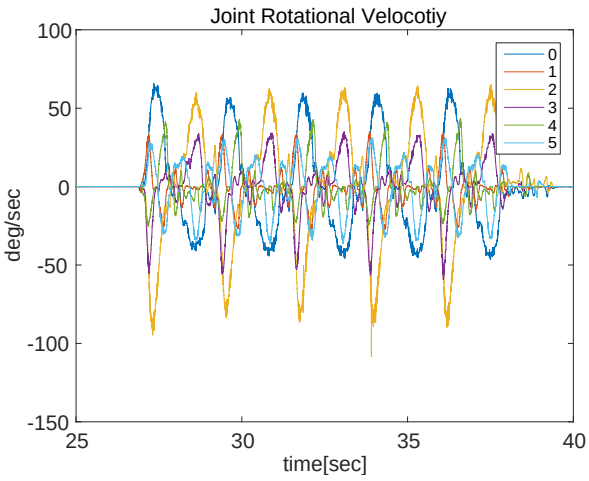
(a) reference joint angle



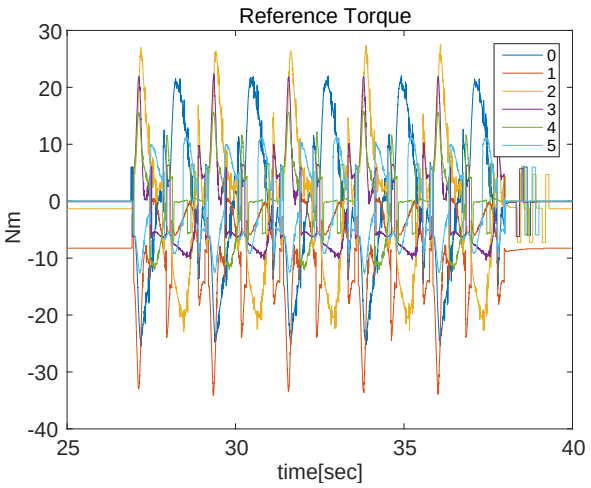
(b) joint angle



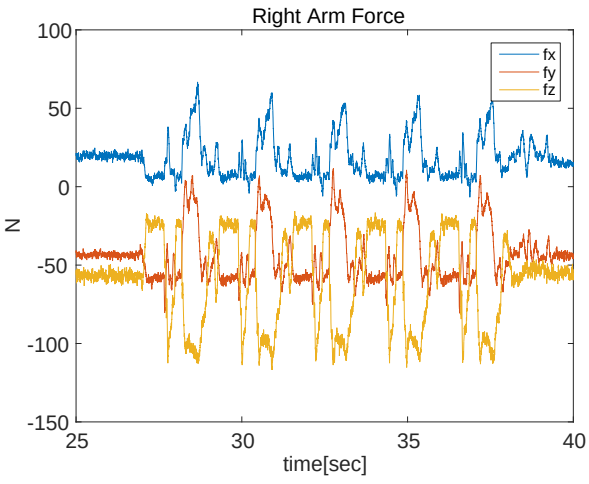
(c) error joint angle



(d) joint rotational velocity

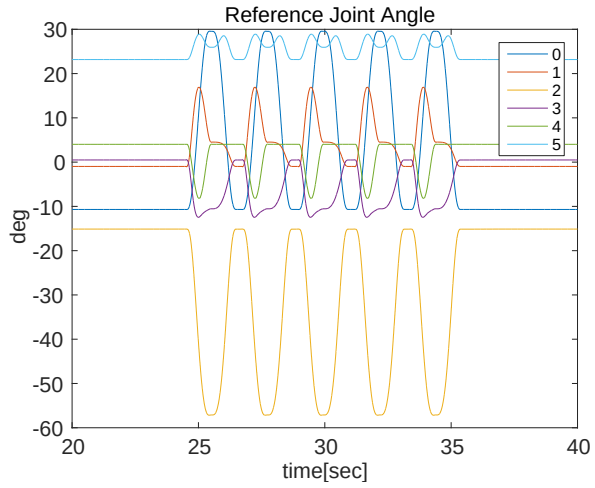


(e) reference torque

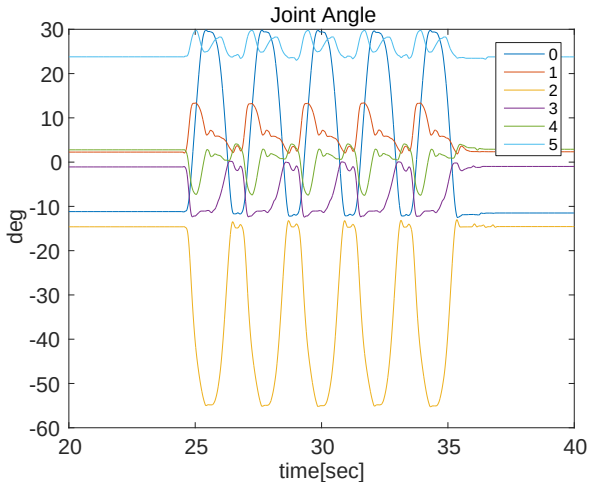


(f) six-axis force

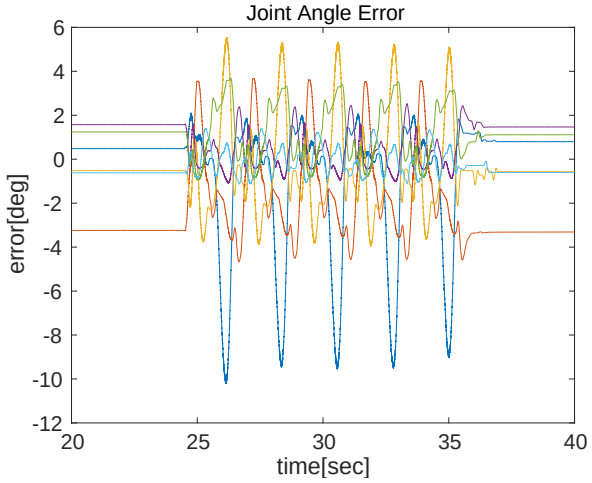
図 6.14: Logged data of step motion test.



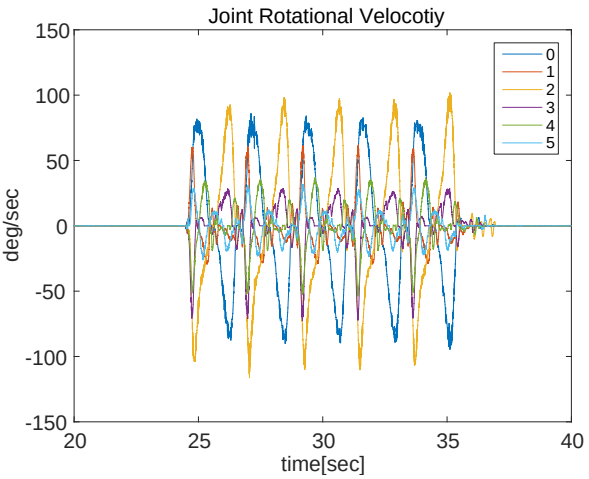
(a) reference joint angle



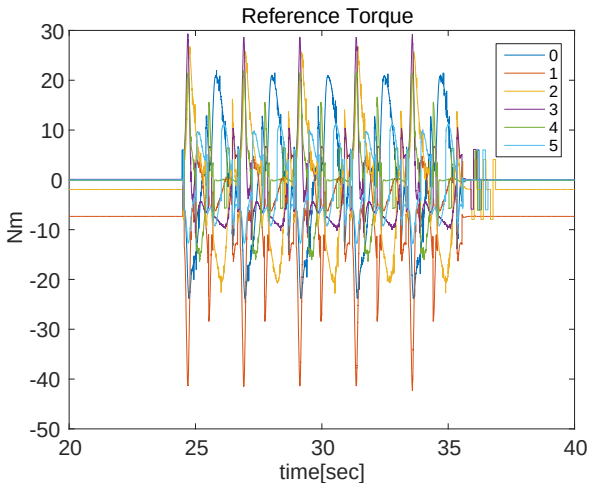
(b) joint angle



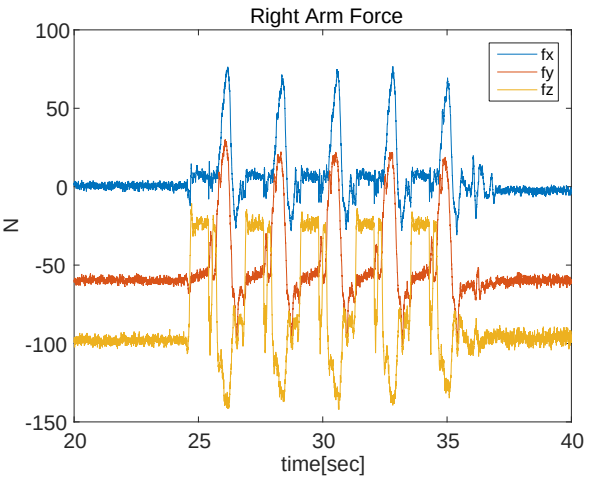
(c) error joint angle



(d) joint rotational velocity

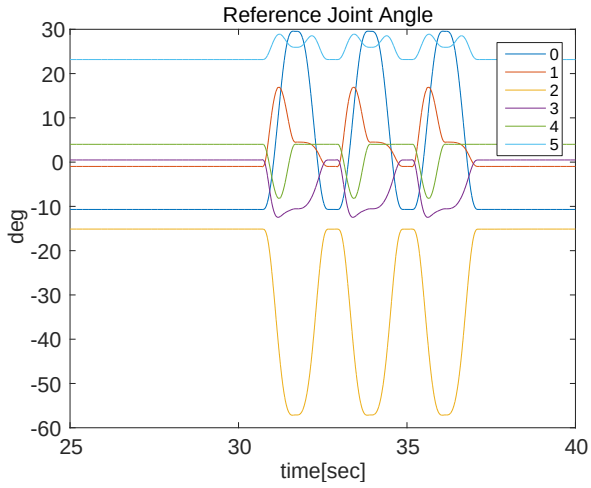


(e) reference torque

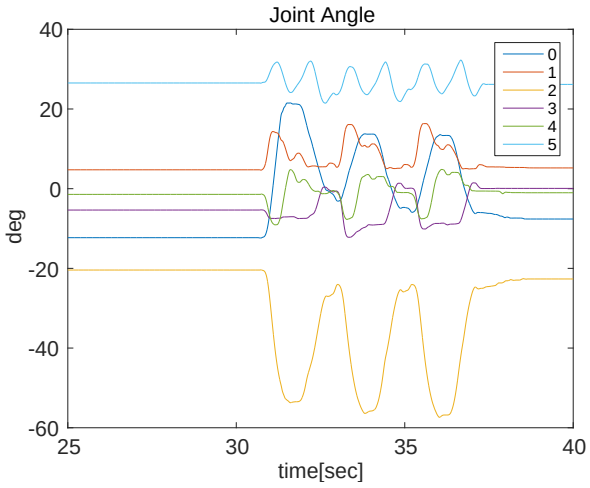


(f) six-axis force

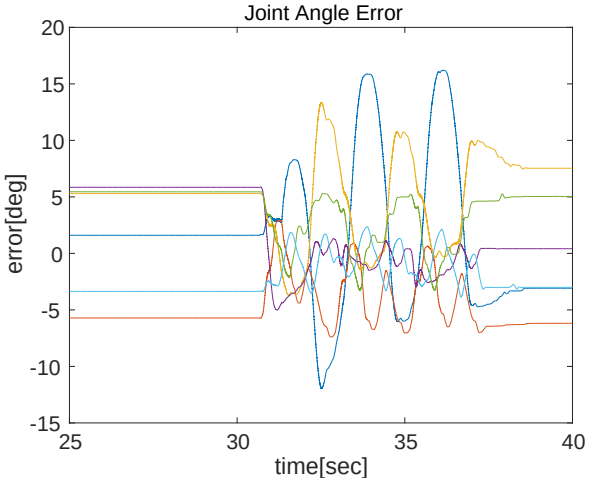
図 6.15: Logged data of step motion test with stamping an obstacle.



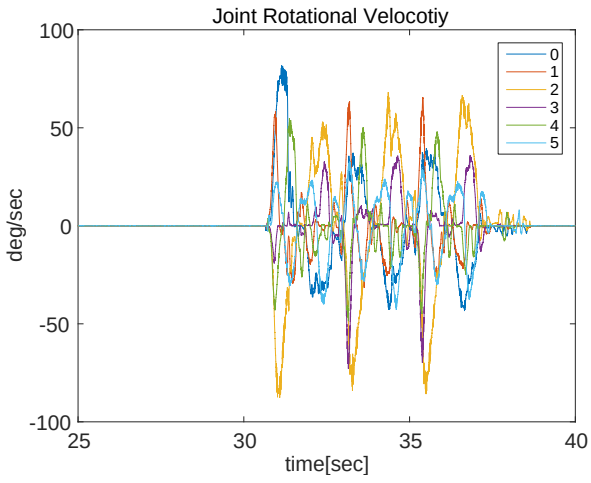
(a) reference joint angle



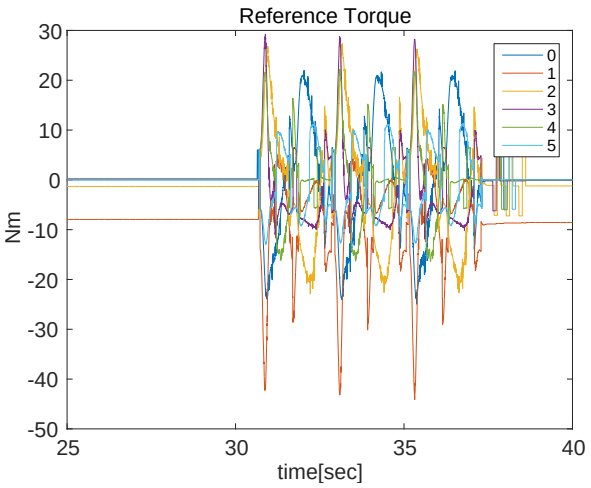
(b) joint angle



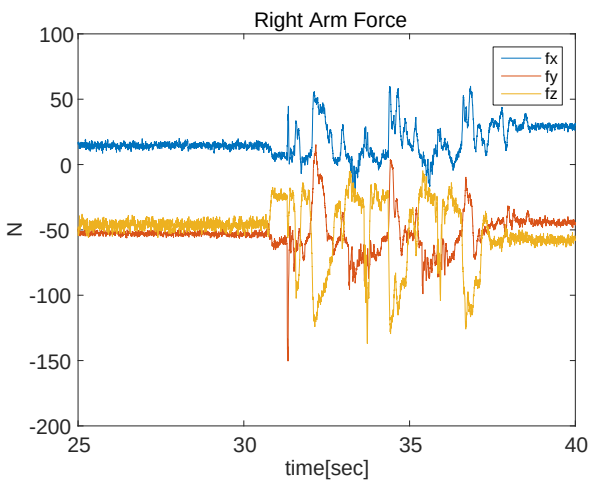
(c) error joint angle



(d) joint rotational velocity

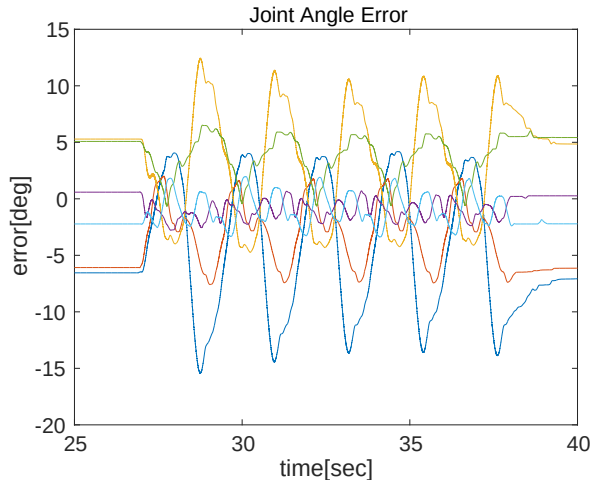


(e) reference torque

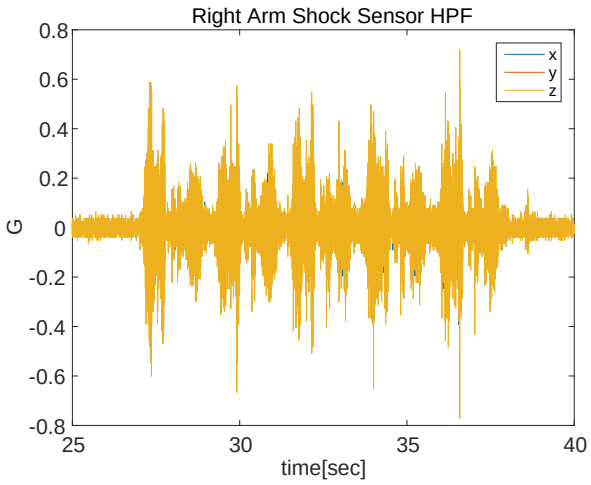


(f) six-axis force

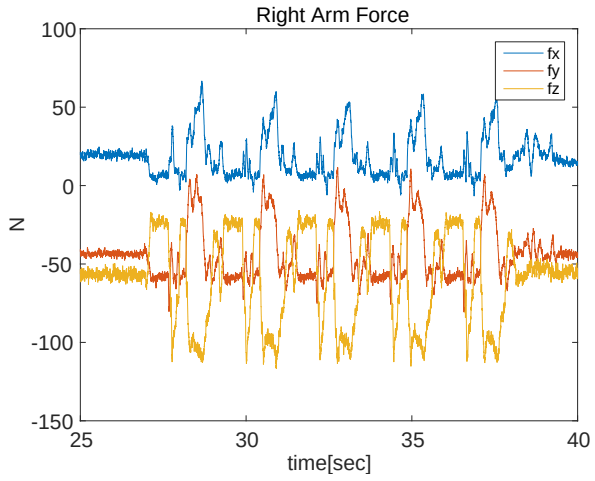
図 6.16: Logged data of step motion test with collision.



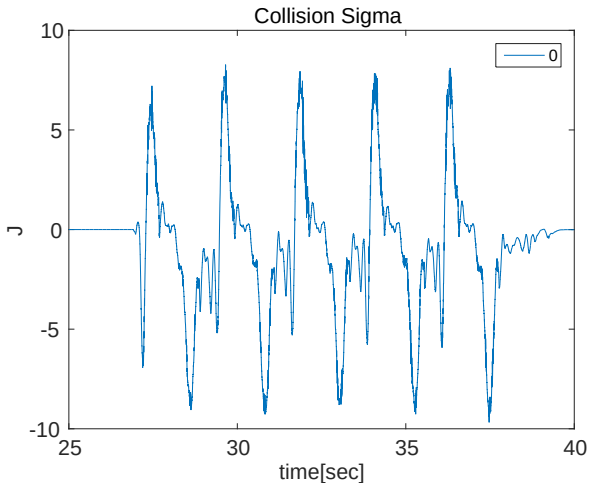
(a) error joint angle



(b) shock sensor

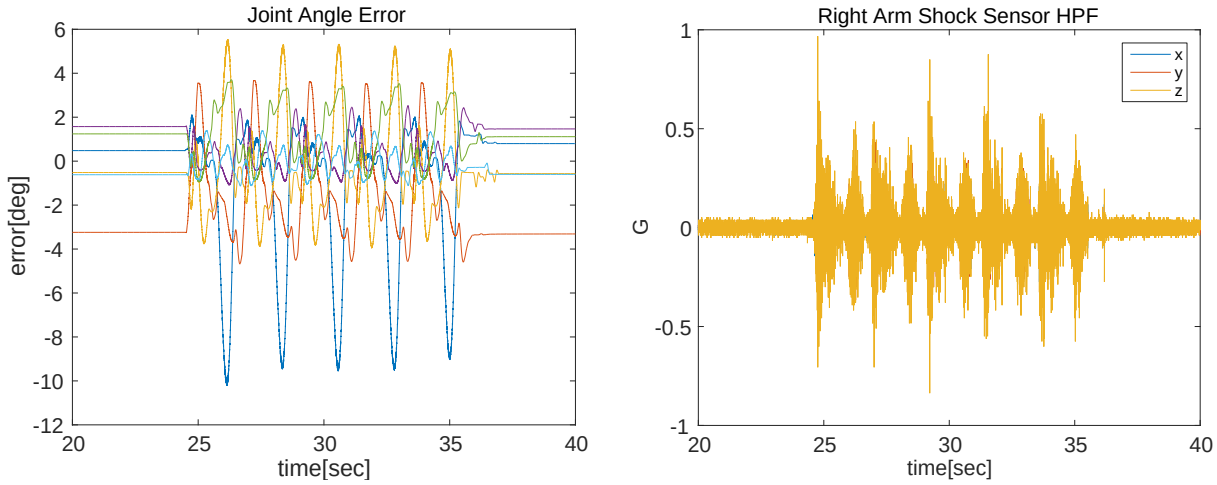


(c) six-axis force



(d) colligion sigma

図 6.17: Colligion related data of step motion test.



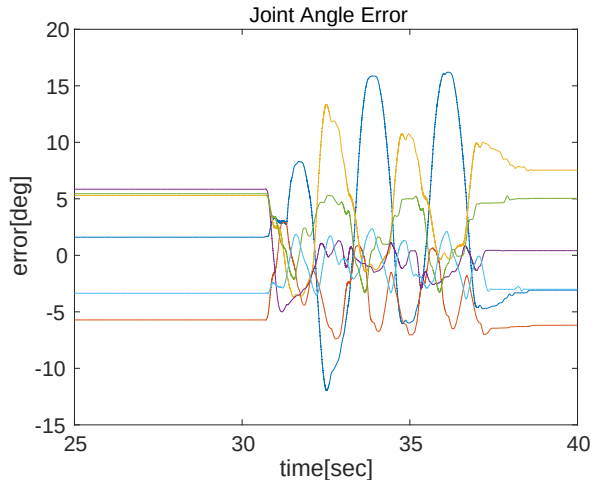
(a) error joint angle

(b) shock sensor

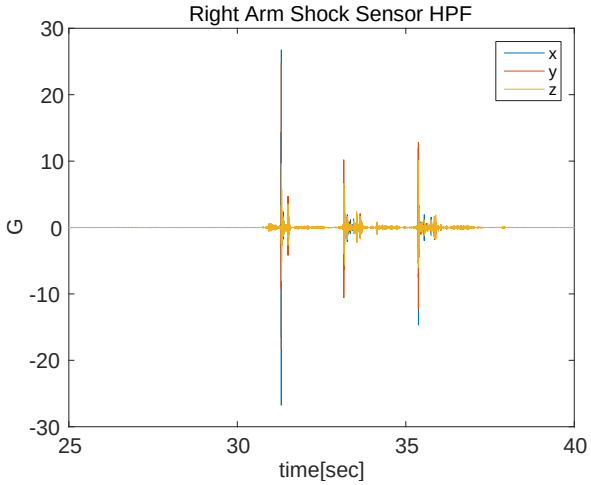
(c) six-axis force

(d) colligion sigma

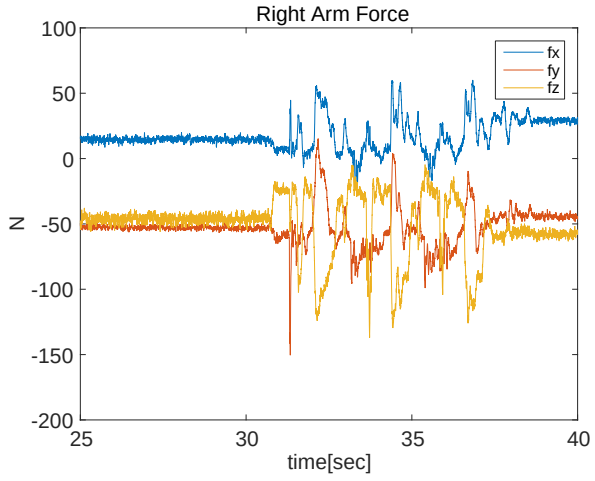
図 6.18: Colligion related data of step motion test with stamping an obstacle.



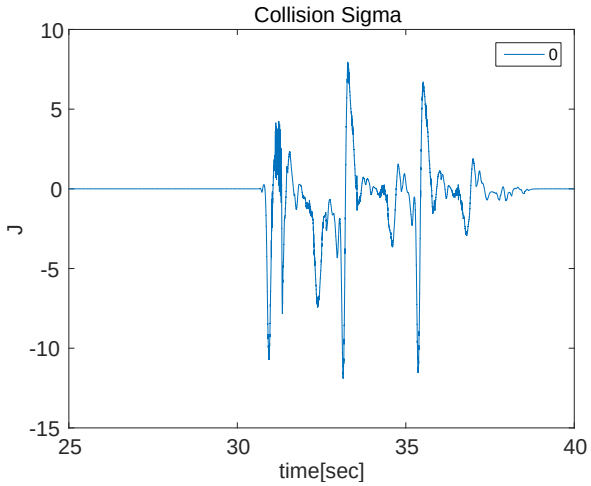
(a) error joint angle



(b) shock sensor



(c) six-axis force



(d) colligion sigma

図 6.19: Colligion related data of step motion test with stamping an obstacle.

6.1.5 制御周期高速化に伴う衝撃応答性能への影響の検証

実験概要

次にコントローラの制御周期による応答性への影響を検証するために、前項の腕型ロボットを用いた実験について、再度制御周期を変更して比較実験を行った。操作指令や設定したパラメータは同一であるが、サーボ演算制御周期のみ 1000Hz、5000Hz と変更して比較を行った。なお、フィルター等のパラメータは制御周期の数値に依存してカットオフ周波数等が同一となるようにプログラムされている。前項での実験と同様に腕のスイング動作時間を 1000msec、600msec、400msec の3 種類についてそれぞれ制御周期 1000Hz、5000Hz の比較を行った。

実験時に用いたパラメータの内、公称値の使用の困難な粘性項パラメータについては慣性力が無視できる低速での動作実験で追従性が十分に確保できる値を 1000Hz と 5000Hz の制御周期それぞれで取得したものであり、その値は同一の粘性係数 6.0Nm/(rad/sec) 及びクーロン摩擦係数 5.8Nm を使用した。

結果

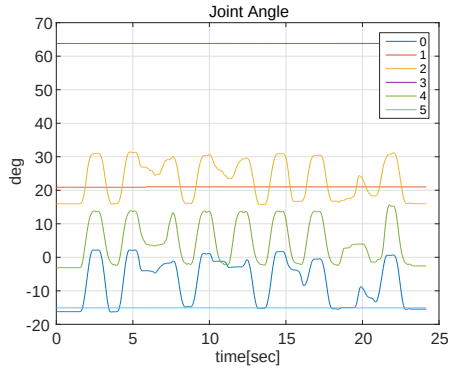
実験によって得られたログデータを 1000Hz と 5000Hz それぞれ比較して示す。図 6.20 に各スイング速度設定の実験について、1000Hz、5000Hz それぞれの制御周期設定にて行った際の関節軌道をグラフに示している。図 6.20a と図 6.20b の比較から、低速の 1000msec でのスイング動作では 1000Hz と 5000Hz の制御周期の差異による動作の違いは確認できるレベルでは存在しなかった。一方、スイング速度を速めた 600msec、400msec の設定においては、1000Hz の制御周期に設定した場合の関節角度軌道にオーバーシュートが発生していることが確認された。特に停止動作領域においてオーバシュートが生じ気味であることが分かった。反対に 5000Hz の制御周期では、動作はダンピングが良く効いており、停止領域におけるオーバーシュートはほとんど見られなかった。

より詳細に差異を確認するために、スイング速度 400msec の際の実験について図 6.21 、図 6.22 及び図 6.24 にロボット第4 関節についての関節角度、関節角速度、関節角度誤差、関節目標トルク、関節目標角加速度を示している。

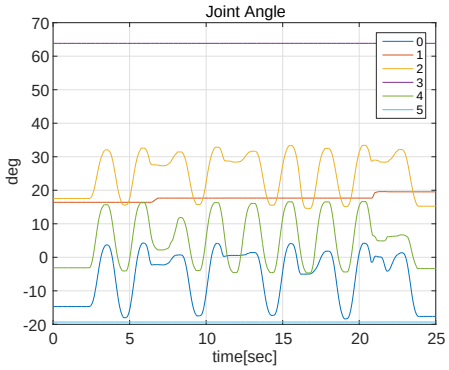
図 6.21a 及び図 6.21b の比較をすると、1000Hz の場合には目標関節角度に対して実関節角度に最大5 度程度のオーバーシュートが生じていることが確認されるが、5000Hz ではほぼ外乱が無い区間で目標軌道に追従していることが分かる。1000Hz の制御周期における動作では、停止時にオーバーシュートが発生し、次いでキックバックによるオーバーシュートが発生することで、大きく軌道から逸脱していることが示される。図 6.22a 及び図 6.22b の比較でも同様の傾向があることが確認される。図 6.24b に示す、関節速度誤差の初動区間にあたる1 秒から 1.4 秒の区間を拡大したグラフにおいて、1000Hz の制御周期では初動に遅れが生じていることが確認される。図 6.24d は、同区間における関節目標トルクを示すが、同一の目標関節軌道を与えられているにも関わらず、出力トルクの立ち上がりに遅れが生じていることが示されており、この立ち上がりの遅れがそのまま初動速度の遅れにつながっていると考えられる。出力トルクの決定において、特に初動トルクの発生には目標関節角加速度を使用して計算される慣性力が支配的である。図 6.24f は、上記時間区間の各制御器で計算された目標関節角加速度軌道を示しているが、確かに 1000Hz の制御周期の場合に角加速度軌道の立ち上がりの遅れが生じていることが記録されている。目標関節角加速度は目標関節角度軌道から微分演算をロバストにするためのロバスト速度推定器 (C.1 節) を用いてオンラインで算出されている。この推定器はフィルタとしての性質を持つ。フィルタパラメータは制御周期を考慮して決定されるが、フィルタ性能として制御周期の影響が強く出る形となり、結果参照トルクの値の違いとして影響していると考えられる。図 6.23 にそれぞれの制御周期における推定器の周波数応答を示す。このような微分演算のための速度推定器を2 段用いることによる加速度計算の影響によって、図 6.24e にも見られる加速度推定値の外れ値が生じているため、トルク制御のための関節角加速度のよりロバストな演算手法を適用することは次の課題となっている。

実験考察

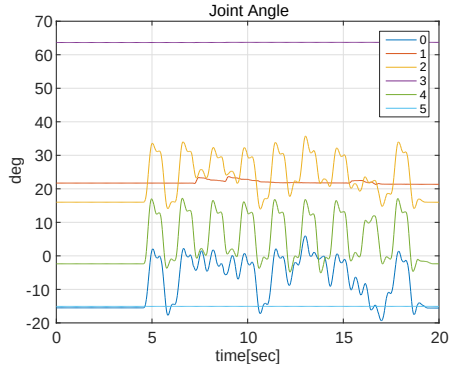
以上のように制御周期の高速化によって、同一パラメータにおいてトルク制御による関節軌道の追従性は向上し、オーバシュートの低減による衝突時の撃力のある程度の緩和が期待されることが分かった。しか



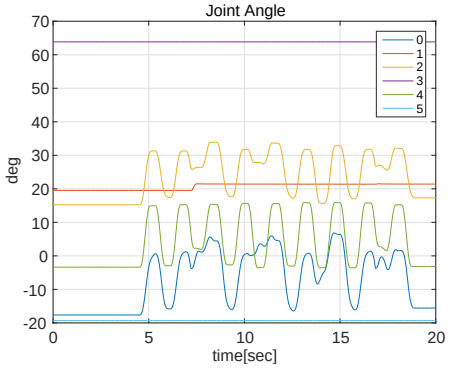
(a) joint angle 1000Hz with 1000msec move



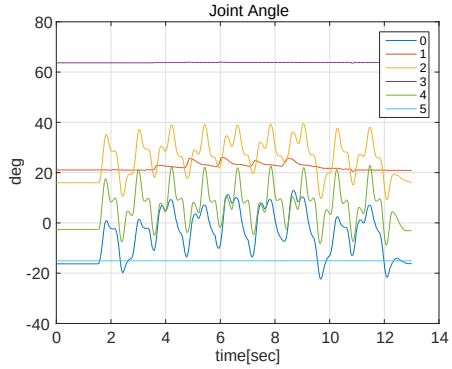
(b) joint angle 5000Hz with 1000msec move



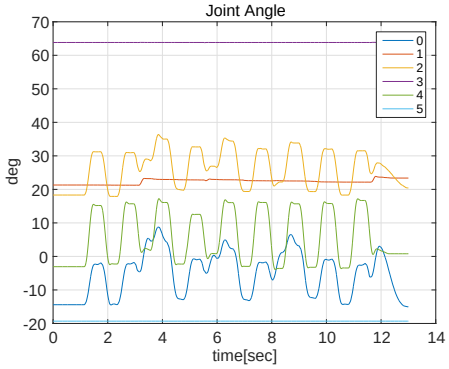
(c) joint angle 1000Hz with 600msec move



(d) joint angle 5000Hz with 600msec move



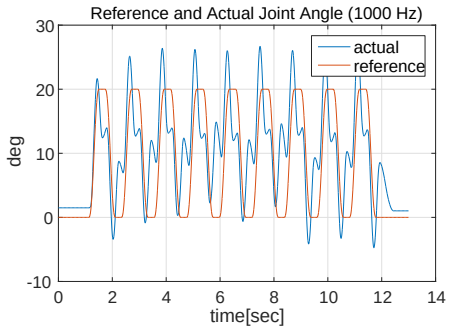
(e) joint angle 1000Hz with 400msec move



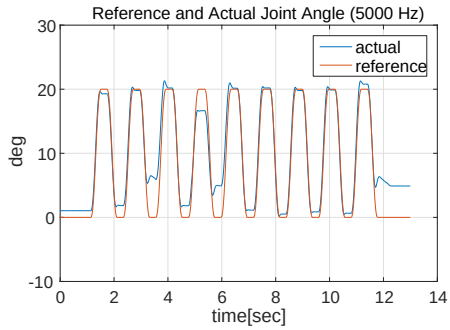
(f) joint angle 5000Hz with 400msec move

図 6.20: Comparison of actual joint angle trajectory by 1000Hz and 5000Hz controller.

しながら、その差は小さく、パラメータの最適化· 調整による向上分の方が大きい程度である。特に粘性系のパラメータの設定は実際の挙動に大きく影響し、実際の関節軸のコンディションに見合ったパラメータの同定が重要であると考えられる。

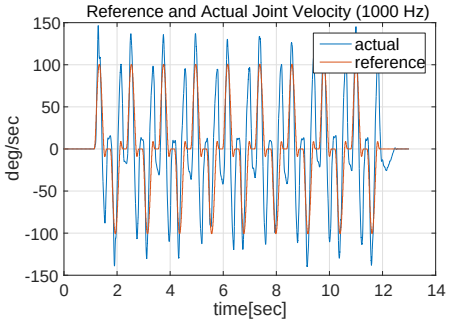


(a) 1000Hz

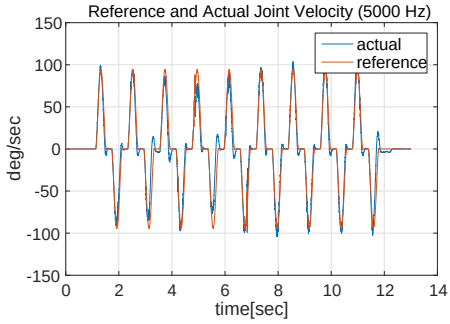


(b) 5000Hz

図 6.21: Comparison of reference and actual joint angle on 400msec swing motion with 1000Hz and 5000Hz controller.



(a) 1000Hz



(b) 5000Hz

図 6.22: Comparison of reference and actual joint velocity on 400msec swing motion with 1000Hz and 5000Hz controller.

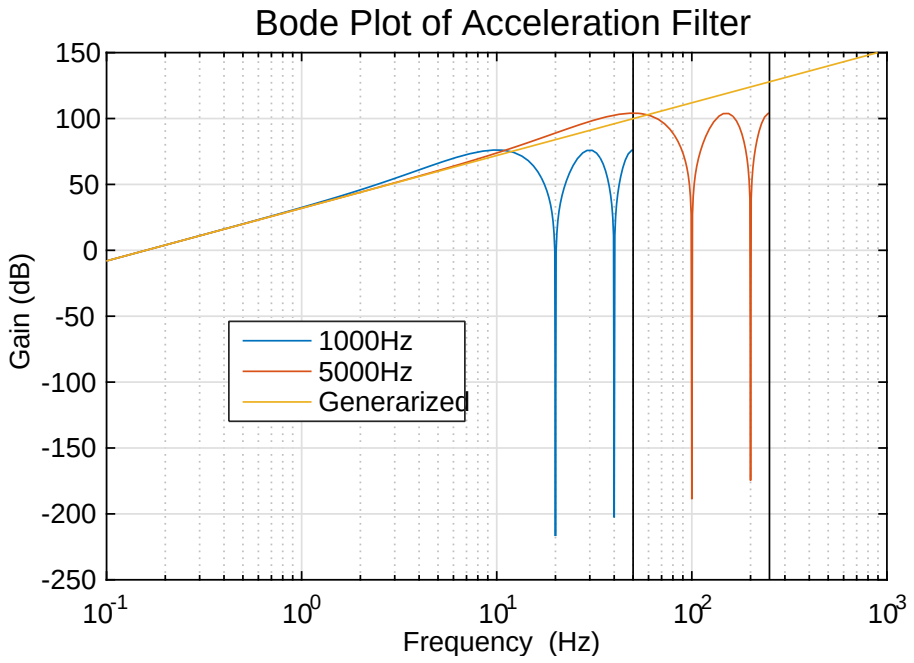
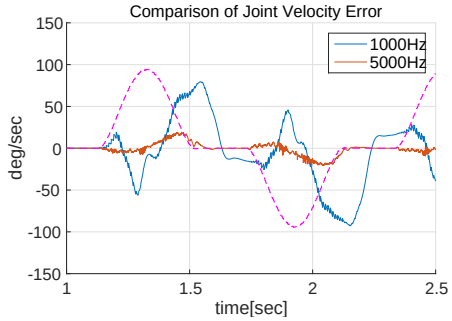
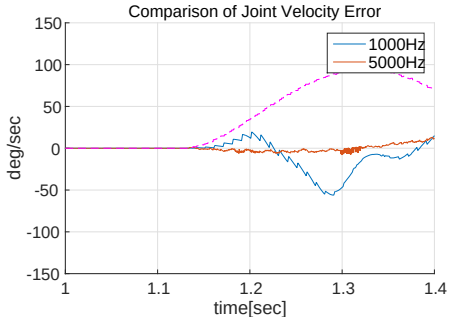


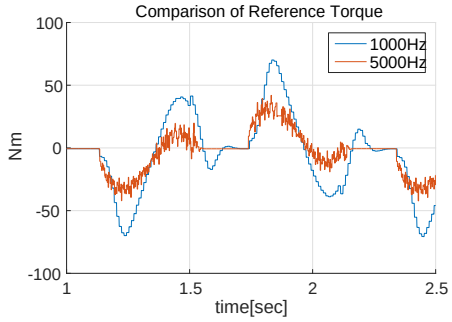
図 6.23: Bode plot of acceleration estimating filter with upsampler comparing 1000Hz, 5000Hz and general acceleration.



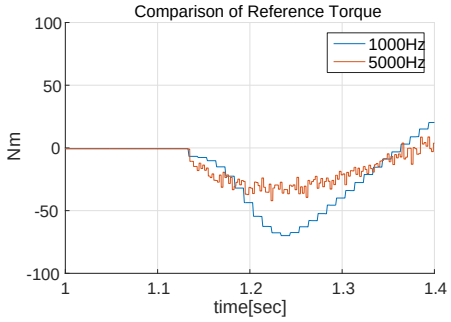
(a) Joint velocity error



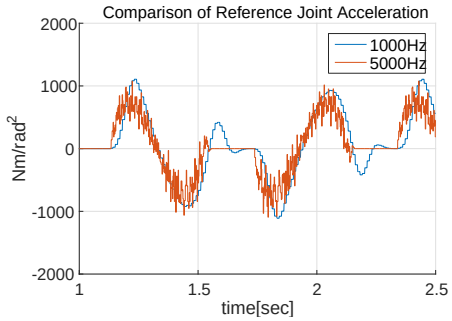
(b) Joint velocity error (enlarged)



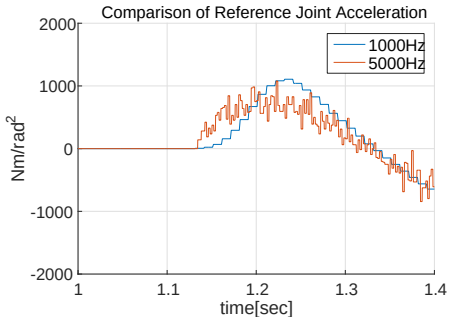
(c) Reference torque



(d) Reference torque (enlarged)



(e) Reference joint acceleration



(f) Reference joint acceleration (enlarged)

図 6.24: Comparison of joint velocity error, reference torque and reference joint acceleration on 400msec swing motion with 1000Hz and 5000Hz controller. Figures on left side are full ranged graph. Figures on right side are enlarged one from 1[sec] to 1.4[sec].

6.2 脚型ロボットによる身体運動計画系の実験

5.9 節にて述べた実時間歩行軌道生成器による身体運動計画系を用いて脚型ロボットでの歩行動作の検証を行う。

6.2.1 実験構成

本実験では脚型ロボット L1(A.2 節) を実験に使用した。実験で得られたデータをグラフに示す際には、表 6.3 に定義する各関節の名称の略称を用いる。

表 6.3: Abbreviates of each joints on L1.

CR	Crotch joint on right leg
CL	Crotch joint on left leg
KR	Knee joint on right leg
KL	Knee joint on left leg
AR	Ankle joint on right leg
AL	Ankle joint on left leg

L1 で使用した上・中・下位システムそれぞれの構成は、5.13 節にて述べたライントポロジ構成を使用している。基本的には 6.1 節にて使用したものと同様の構成となっているが、中間層制御システムは右脚と左脚をそれぞれ制御分担するために2 ノード構成となっている。

本研究においては、上位層制御システムはロボットの体外に配置し、EtherCAT ケーブルを引き延ばすことで体内の中間層制御システムに接続している。また、体内にブリッジ構成とした WiFi ルーターを搭載し、中間層制御システムの Ethernet 接続は無線 LAN による接続となっている。また、搭載されているデバイスの詳細は表 6.4 の通りである。

[5]www.intersense.com/
[6]www.beckhoff.com/english.asp?ethercat/fb1111_fb1122_fb1130.htm
[7]www.vectornav.com/products/vn-100
[8]www.wacoh-tech.com/

表 6.4: Configuration of L1 control system.

Upper Layer	# of Nodes	1
	CPU	Intel Core i7-2700K CPU @ 3.50GHz 8threads (Hyper-Threading enabled)
	memory	8GB
	OS	Desktop Linux (Ubuntu14.04)
	NIC 1	On board NIC, Realtek RTL8111 (for EtherCAT)
	NIC 2	USB2.0 Ethernet Adapter 100Mbps, ASIX AX88772 (for Ethernet)
	IMU (USB Connected)	1 (NavChip)[5]
Middle Layer	# of Nodes	2
	CPU	ARM Cortex-A9 Multi-Core @ 800MHz on Hardware Processing System (HPS) in FPGA
	Memory	512MB×2 (DDR3)
	OS	Linux 3.8.0 (Xillinux, Ubuntu14.04)
	EtherCAT slave	1 (FB1111-0140)[6]
	EtherCAT bridge	1
	active optical fiber link port	2
	IMU (USB Connected)	1 (VN-100)[7]
Lower Layer	# of Nodes	12
	Hardware	RMTP-02D
	Sensor	Shock detector 2 on foots
		Six-Axis force sensor 2 on foots (WEF-6A1000-80-40-RCXTi2)[8]

6.2.2 歩行実験

実験概要

この実験では本研究で提案するシステムを搭載した脚型ロボット L1 が、5.9 節にて導入したオンライン歩行生成器を用いて歩行動作が可能であることを実証することを目的とする。ここでは、まずシンプルな歩行動作として前方への歩行及び後方への歩行を行う。この実験では歩行動作指令は、操作者がジョイスティックを用いて行っている。

歩行生成器のパラメータとして、一歩当たりの時間を 400msec、歩行軌道足部高さを 40mm、最大前進歩幅 200mm を設定している。またこの実験では、オンライン歩行生成器の動作検証であるため関節の制御モードは関節角度サーボ制御としており、関節トルク指令は利用されていない。さらに通常 hrpsys にて実行される ZMP トラッキング制御や床反力制御を含むスタビライザーコンポーネントの処理については無効とした。

結果

図 6.25 に実験時に撮影された動画から切り出した連続写真を示している。図 6.26 には上位システムのオンライン歩行生成器にて記録したログデータを示す。また、図 6.27 には中間層制御システムにて記録されたログデータを示す。

図 6.26a と図 6.26b に、X 軸（前後方向）及び Y 軸（左右方向）についてのオンライン歩行生成器にて生成された目標 ZMP 軌道とその目標重心軌道を示している。初期4 歩分についてはその場での足踏み動作が生成され、その後 11 歩分の前進歩行軌道が生成された。続いて、3 歩のその場での足踏み動作の生成の後、6 歩分の後進歩行軌道が生成された。図中で 7.5 秒付近及び 12.5 秒付近で軌道が一度0 にリセットされているのは、歩行指令が停止され、生成された歩行軌道が全て再生されたことを示す。またその際、実体幹位置が0 点へと復元し、初期姿勢と一致するように、足位置との相対関係から推定された体幹位置をオフセットとして加算することで補正をかけている。

図 6.26c と図 6.26d には、X、Y 軸それぞれについて体幹部を基準とした相対 ZMP 目標位置、及び両足部に備え付けられた6 軸力センササンプル値から計算される実 ZMP 計測位置を示している。実 ZMP は、目標 ZMP に対して完全な追従には至っていないものの、概ねの波形としては目標値と相関した波形が得られている。実験時の歩行速度やロボットの重量や慣性モーメントのモデルと実機の誤差、6 軸力センサのノイズ等があることや、本実験ではスタビライザによる ZMP 補償のためのトラッキング制御を切っていることを考慮すると、5.9.3 節にて述べた多リンクモデルベースでの ZMP 誤差補償によって出力された重心軌道が正しい歩行動作となっていることが分かる。

また、実 ZMP は 7.5 秒付近及び 12.5 秒付近の歩行動作の停止領域で大きく追従性が損なわれているが、これは歩行指令の停止に対して即座に終端用の歩容を挿入するため、停止のための重心軌道の猶予時間が短くなり、結果予見制御によって生成される重心軌道が必ずしも安定であることが保証されないためである。これを回避するためには、生成された重心軌道の終端時の安定度について評価を行い、場合によっては追加の歩容を生成するといった処理が必要になると考えられる。

図 6.27d のヨー軸や連続写真を見てもわかる通り、本実験では指令値として与えていないヨー軸まわりの回転が生じている。これは、遊脚のスイングによって生じているヨー軸まわりの角運動量に対して、支持脚の接地面のヨー軸まわり摩擦が不十分で滑りが生じているためである。前述の通り、L1 の足裏はアルミの加工面がむき出しであるため床面との摩擦係数が低い状態である。また、脚型ロボットの身体構成の都合上、遊脚のスイングによって生じる角運動量を打ち消すために、他の自由関節によって角運動量の補償を行う余地が無い。理想的には足裏の摩擦係数を高め、且つ遊脚スイングによる角運動量を打ち消せるだけの補償用角運動量を生成可能な余剰自由関節を身体に備えることが望まれ、将来的な課題となっている。

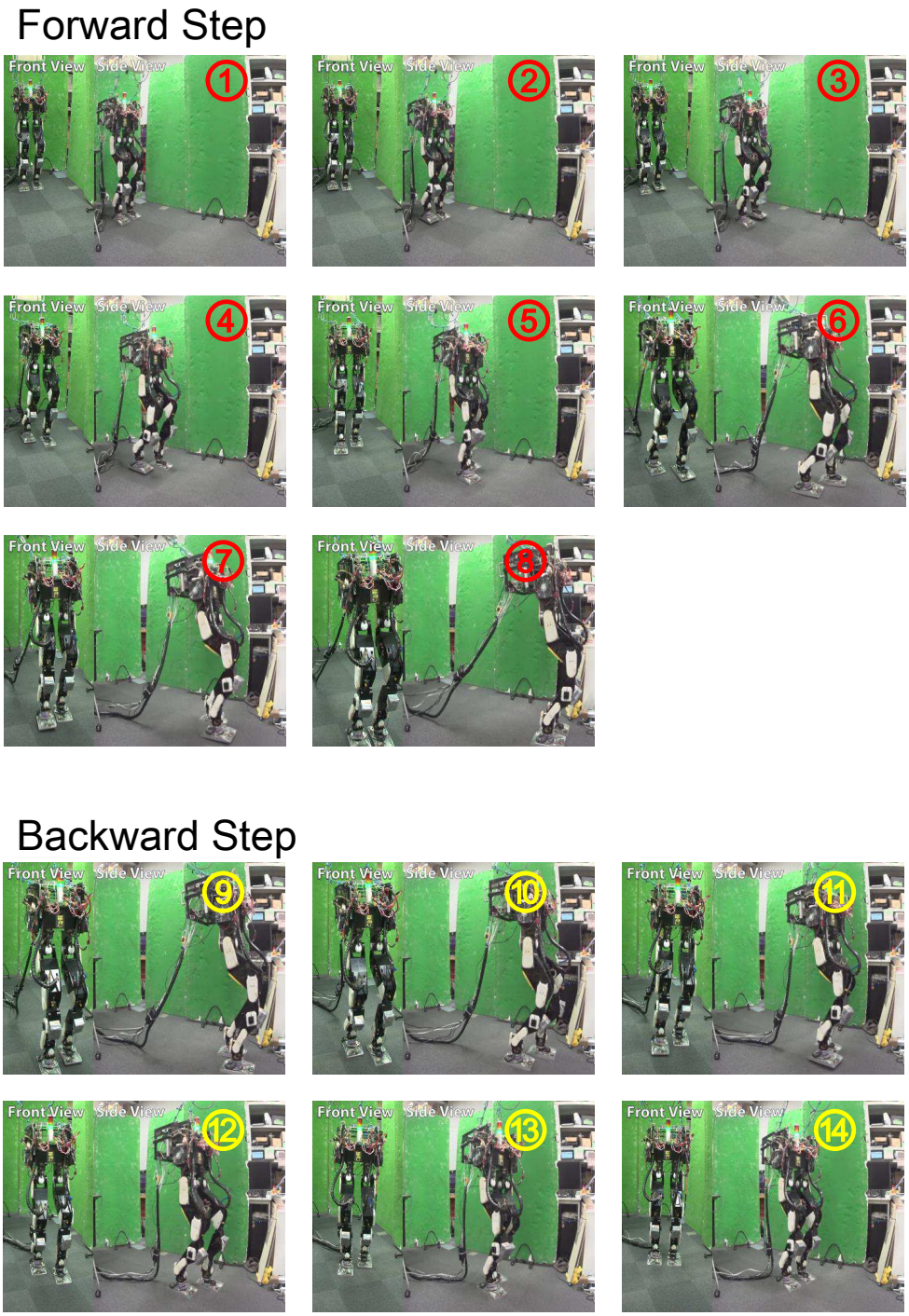
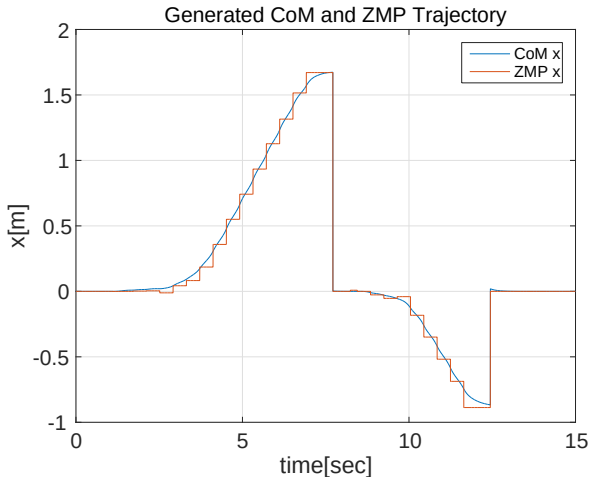
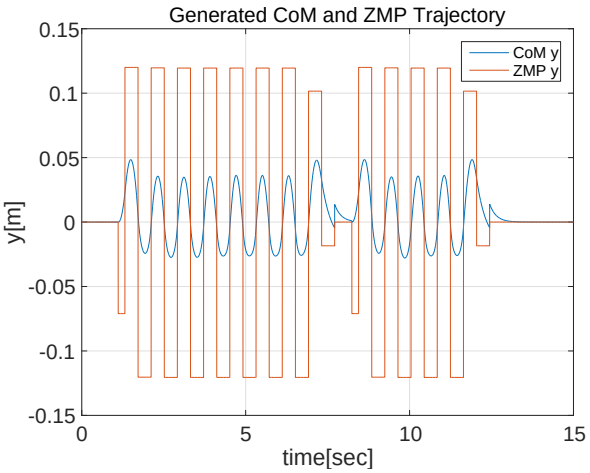


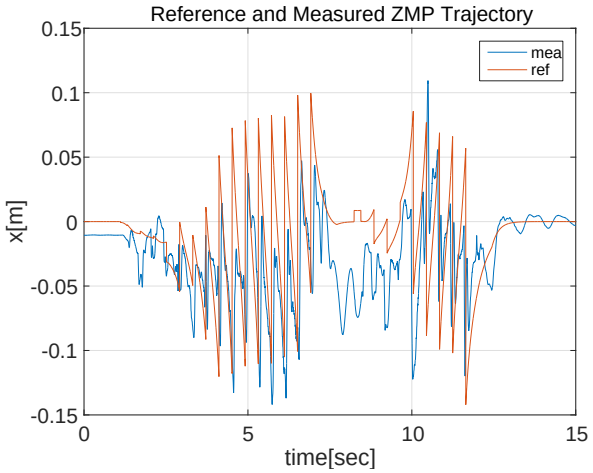
図 6.25: Photographs of forward and backward walk experiment.



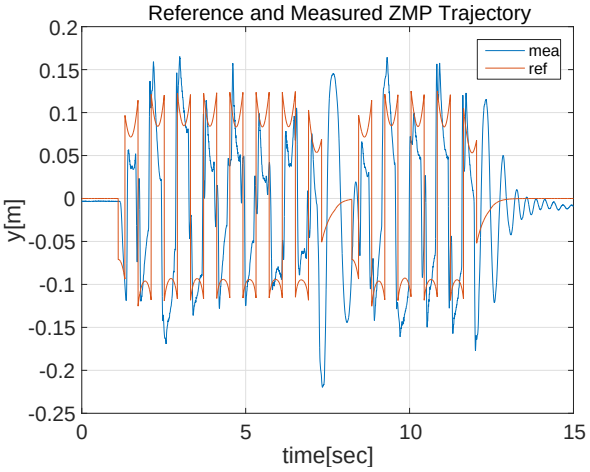
(a) X axis generated CoM and ZMP trajectory.



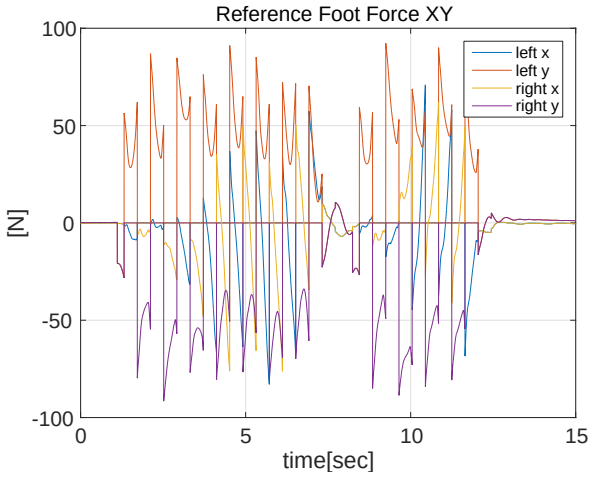
(b) Y axis generated CoM and ZMP trajectory.



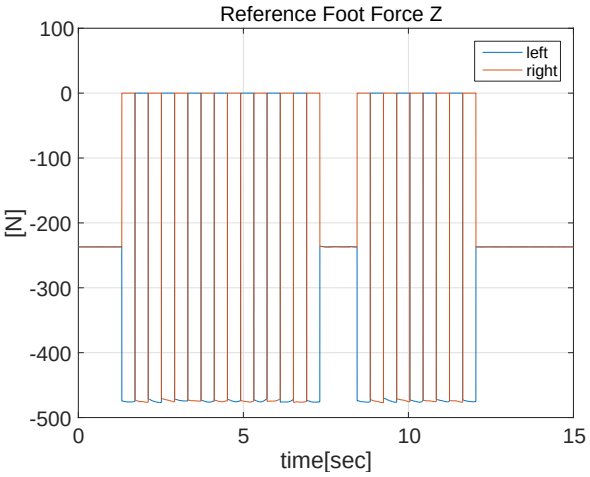
(c) X axis reference and measured ZMP relative to CoM.



(d) Y axis reference and measured ZMP relative to CoM.

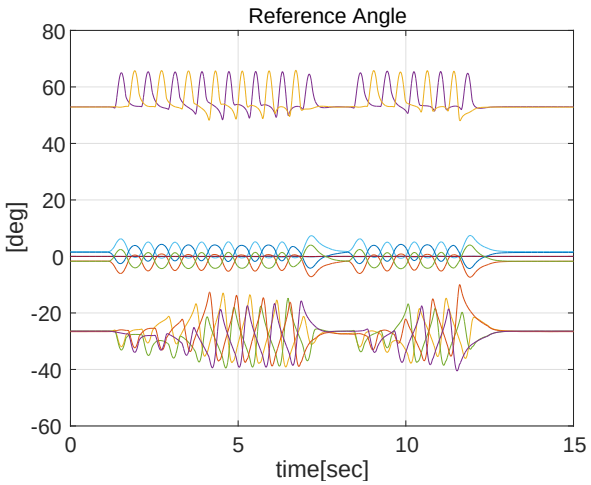


(e) X and Y axis reference force on foots.

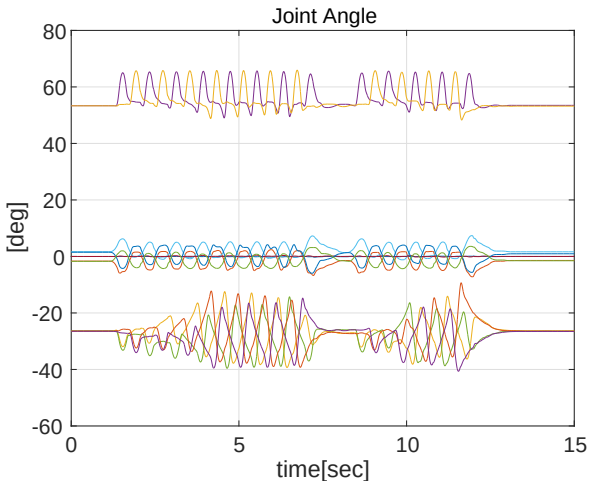


(f) Z axis reference force on foots.

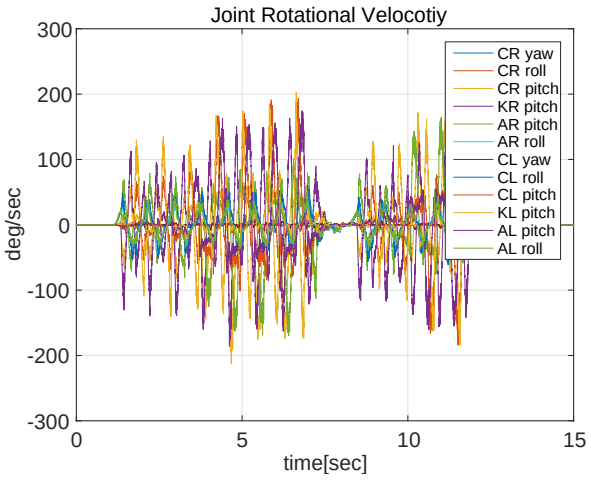
図 6.26: Logged data of online walking motion experiment on upper layer system.



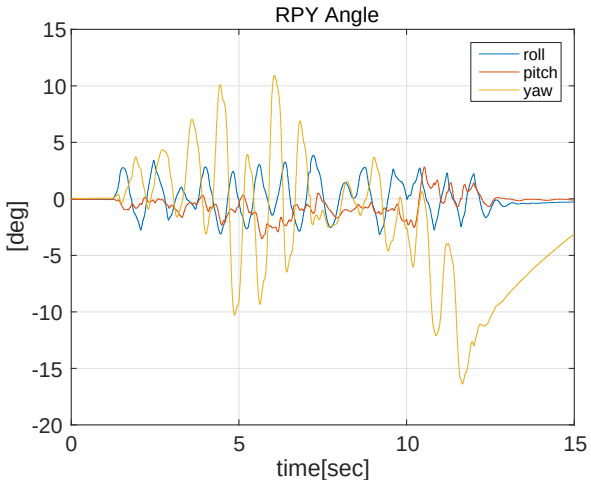
(a) Reference joint angle.



(b) Joint angle.



(c) Joint rotational velocity.



(d) RPY angle.

図 6.27: Logged data of online walking motion experiment on middle layer system.

図 6.28: Objects to be stamped during this experiment.

6.2.3 物体踏破実験

実験概要

次に、オンラインでの歩行動作生成を活用した、物体の踏破実験を行う。図 6.28 にこの実験で脚型ロボットに踏破させた物体の写真を示す。2 つの物体は全てアルミ製で、1 つは7 mm の DIN レール用部材、もう一つは 15mm 角の C チャンネル材である。

これらの物体を床に置き、その上をロボットに踏破させることでオンラインでの歩行動作生成の様子を検証する。L1 では足部には特にゴム系素材は用いず、アルミの加工平面がそのまま地面に接する形となっている。そのため、厚さ 7mm の板材でも片足で踏んだ際には線接触に近い状態となるため、非常に不安定となる。その結果、本来の軌道との誤差によって大きな反力が発生し、体幹部の大きな傾きを生じさせる場合がある。5.9.4 節にて導入した遊脚接地高修正制御は、この反力を低減する制御であるが、体幹の傾きは残ったままとなり、体幹部に本来の軌道と異なる運動量が生じる原因となるため、2 歩3 歩と歩行を継続した先で転倒する恐れが生じる。オンラインでの歩行生成器は体幹の傾きを検出し、LIPM に基づいた最適な重心軌道によって転倒を回避する制御である。

歩行途中で物体を踏むような位置に配置した上で、L1 に前方への歩行指令を出し、物体を踏んだ後の歩行挙動について上位システムと中間層制御システムそれぞれで記録を取った。歩行パラメータは前節と同様、一歩当たり 400msec の時間で最大 200mm の歩幅と 40mm の遊脚高さに設定されている。

なお、留意事項として本実験では前節と同様に hrpsys に組み込まれているスタビライザーコンポーネントによる ZMP トラッキング制御や足裏反力制御による、バランスフィードバック制御は行っていない。上位システムにおけるフィードバック要素としては、IMU から受け取った姿勢情報による体幹基準姿勢の修正、傾き検出による歩行軌道修正、足裏反力による遊脚接地高修正制御のみとなっている。また、動作に必要な前方への歩行指令は操作者がジョイスティックによって与えている。

結果

図 6.29 に実験時に撮影された動画から切り出した連続写真を、図 6.30 に上位システムのオンライン歩行生成器にて記録したログデータを示す。また、図 6.31 には中間層制御システムにて記録されたログデータを示す。

図 6.31d の IMU センサロール角を見ると、図中 5.6 秒付近から -3.7 度程度と傾きが大きくなっていることがわかる。また、ピッチ角についても同タイミングで後方に転倒する向きに傾きが大きくなっている。このステップは図 6.29 の⑦のタイミングに相当する。図 6.30a と図 6.30b を見ると、その 5.6 秒付近のタイミングで目標 ZMP 軌道の歩幅が大きくなり、横方向である y 軸については-0.5m 付近まで移動する歩行が生成されている。実 ZMP についても、図 6.30c と図 6.30d を見ると、5.6 秒以降に一度大きく目標値から実 ZMP が外れている。その直前のステップで、板材を踏んだことで姿勢が大きく傾き、結果次に出していた遊脚の着地時刻が目標時刻より大幅に早まったことがこの実 ZMP の乖離の原因であると考えられる。その次のステップ (6.0 秒付近) では、修正された歩行軌道によって目標値に近い位置まで修正されている。

なお 8.9 秒以降に図 6.30a と図 6.30b の値が0 に飛んでいるのは、歩行指令が停止され生成された歩行軌道が全て再生し終わったタイミングで軌道のリセットがかかるためである。

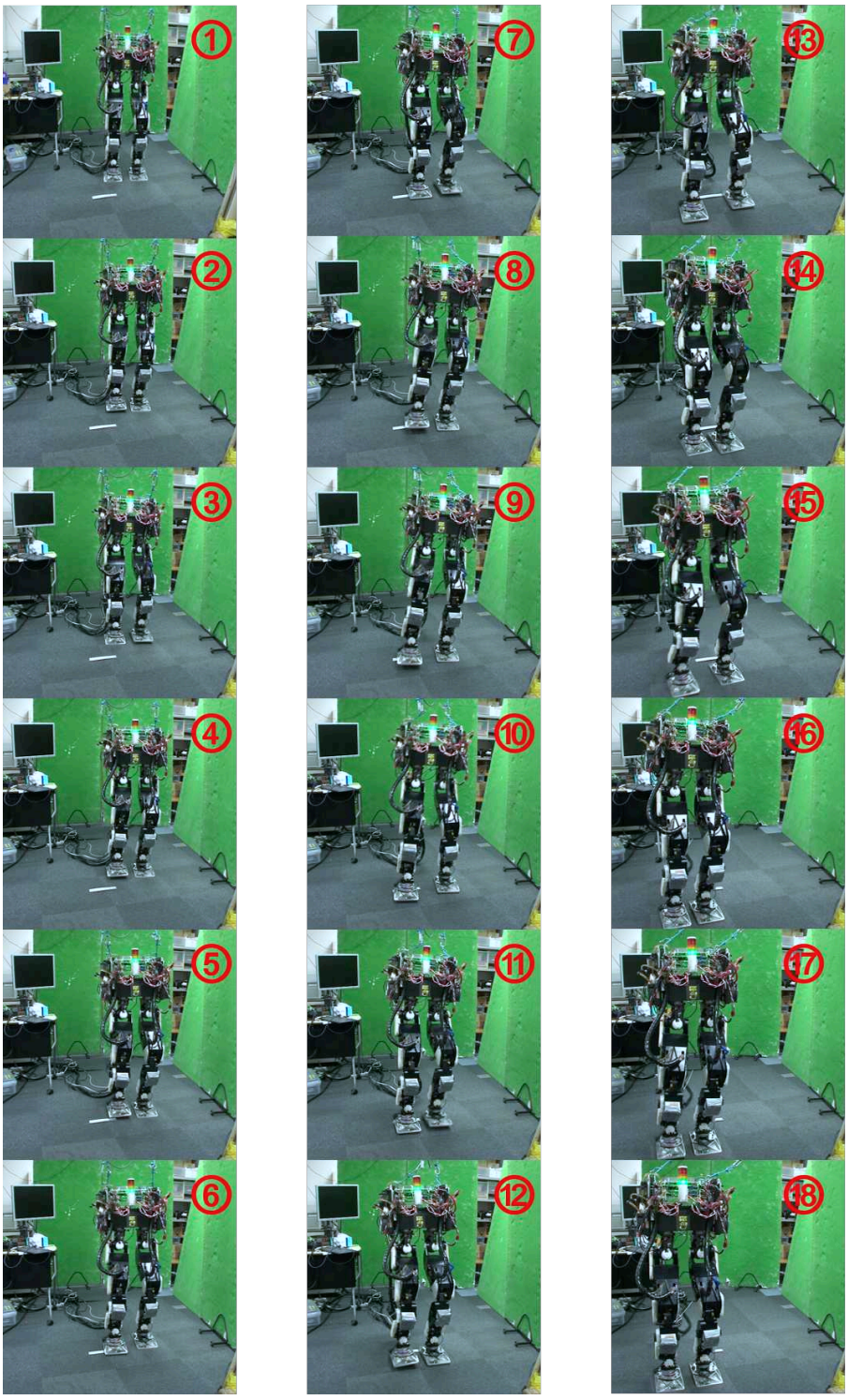
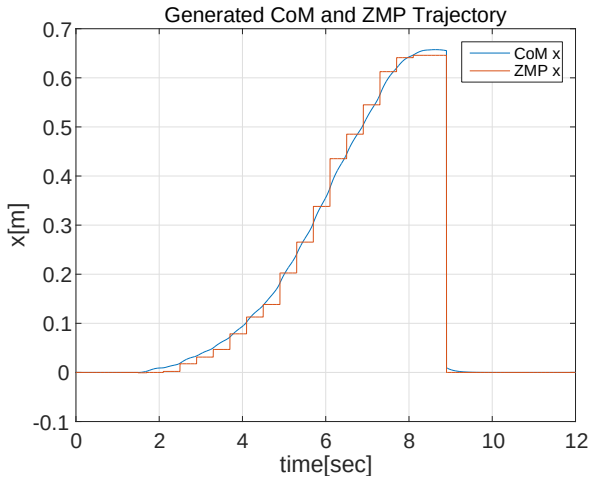
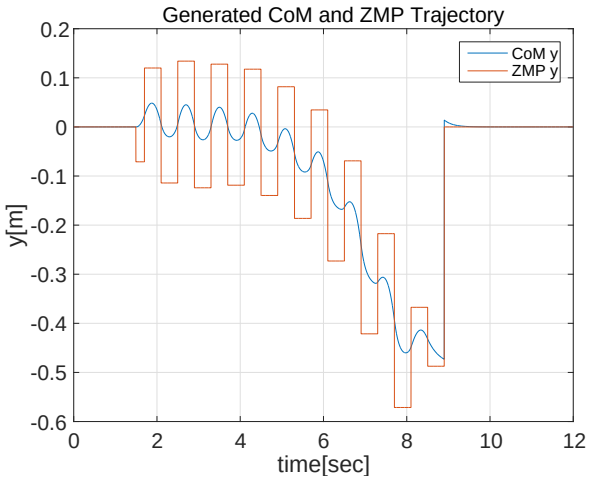


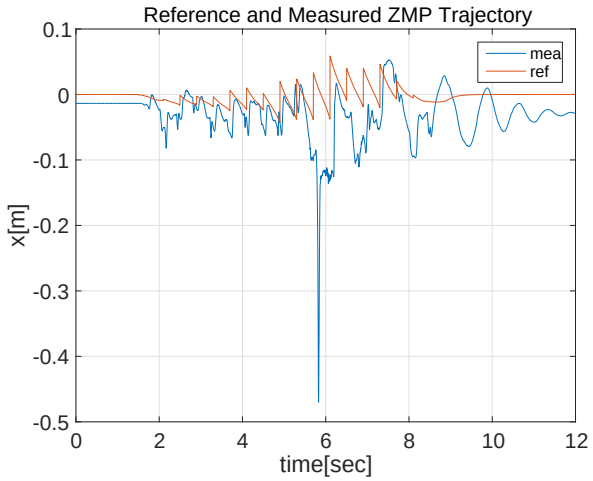
図 6.29: Photographs of online walking experiment.



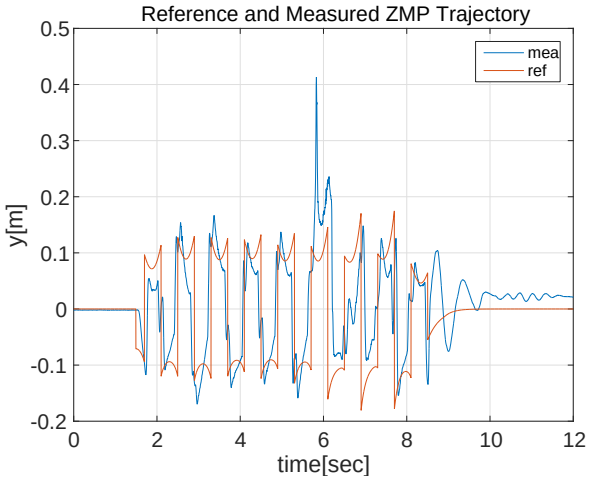
(a) X axis generated CoM and ZMP trajectory.



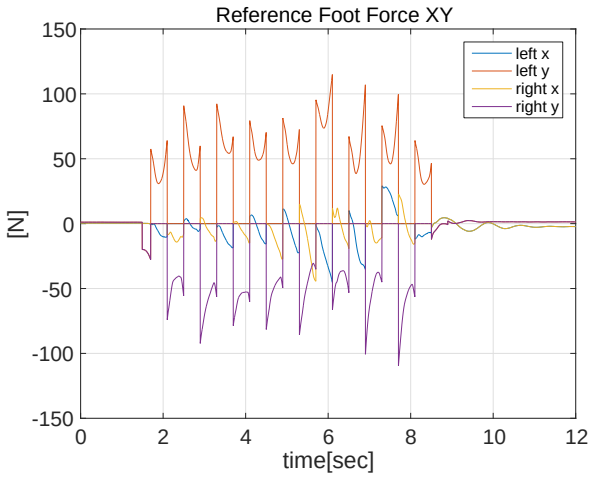
(b) Y axis generated CoM and ZMP trajectory.



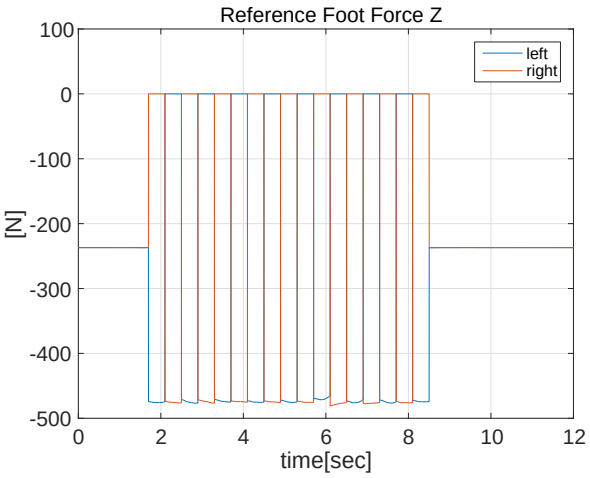
(c) X axis reference and measured ZMP relative to CoM.



(d) Y axis reference and measured ZMP relative to CoM.

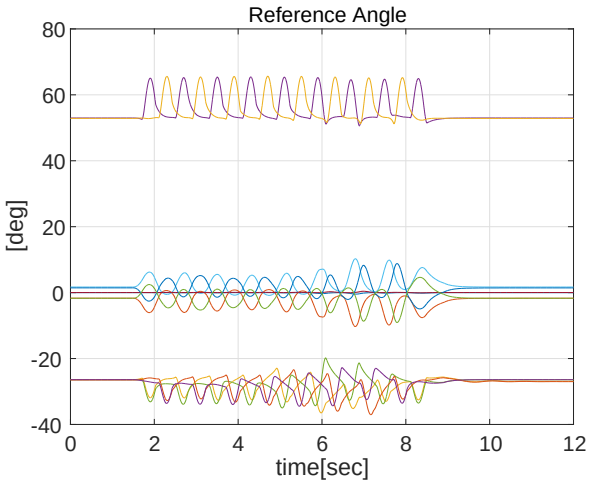


(e) X and Y axis reference force on foots.

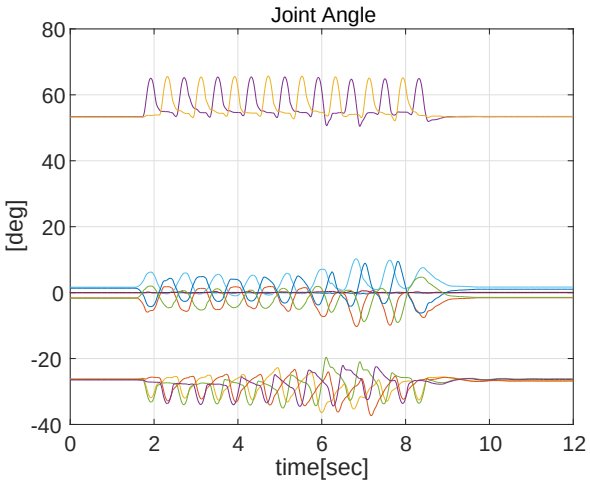


(f) Z axis reference force on foots.

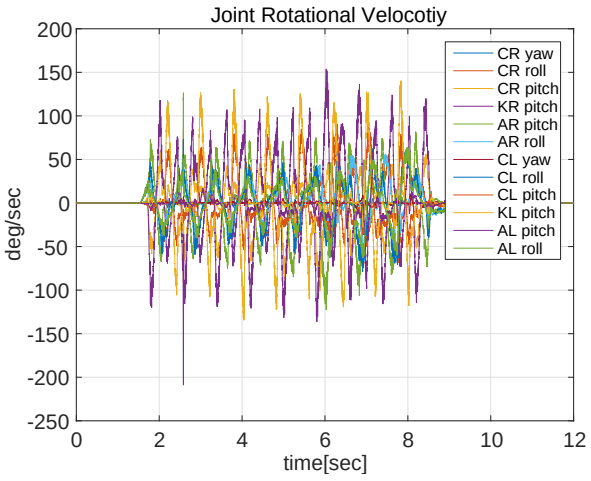
図 6.30: Logged data of online walking motion experiment on upper layer system.



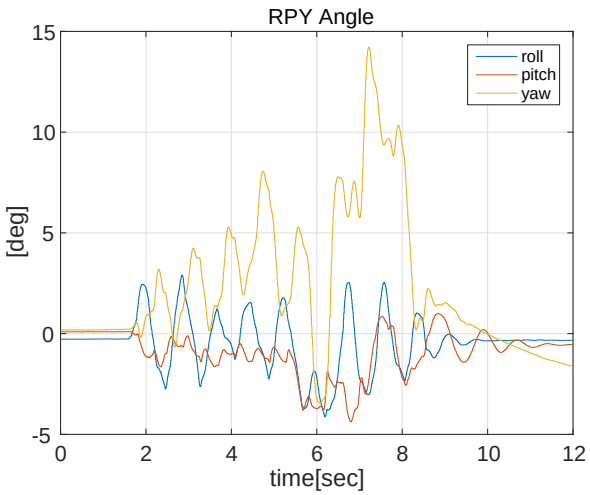
(a) Reference joint angle.



(b) Joint angle.



(c) Joint rotational velocity.



(d) RPY angle.

図 6.31: Logged data of online walking motion experiment on middle layer system.

6.3 脚型ロボットによる即応的反射行動の統合実験

本節では、身体運動計画系と即応的動作制御系の全てを統合した状態で実験を行うことで、本研究が提案する即応的反射行動アーキテクチャが目的とする、外乱に対して物理的な作用によって衝撃緩和を行う柔軟な身体運動とバランス保持のための身体運動計画を実現することを実証していく。

6.3.1 トルク制御状態での立位バランス実験

実験概要

この実験では、脚型ロボット L1 をトルク制御モードで自立させ、バランス状態について実験を行う。各関節のトルク制御について、剛性制御を適用することで、関節角度の指令値からの誤差に対して復元力が働くように設定している。この剛性パラメータについては、表 6.5 に示す値を与えている。

表 6.5: Configuration of spring and dumping parameter.

Joint Name	Spring Coefficient [Nm/rad]	Dumping Coefficient [Nm/(rad/sec)]
CL/CR Yaw	1200	18
CL/CR Roll	1200	18
CL/CR Pitch	1200	18
KL/KR Pitch	1200	18
AL/AR Pitch	500	8
AL/AR Roll	500	8

上位層からは、立位状態時にロボットの自重を支持するために必要となる足裏反力が LIPM から出力されており、その足裏反力を発生させるために必要となる各関節トルク値が、上位層から中間層制御システムへと目標トルク値として通信されている。この目標トルク値は現在の姿勢や関節角度に合わせて常時変更される。また、バランス保持のために IMU から得られる体幹姿勢の傾き情報を用いて、体幹を水平に復元するための姿勢変化を与えている。スタビライザーコンポーネントによる ZMP 等の足裏力から重心軌道へのフィードバック制御は行っていない。実験中に操作者が体幹部分を前後左右に押すことで姿勢を崩そうとすることでバランス状態を確認する。

実験結果

図 6.32 に実験中に撮影された動画から切り出した連続写真を示す。脚は操作者の外乱入力に対して、なじみつつバランスを維持することができた。図 6.33 には上位層制御システムにて、そして図 6.34 には中間層制御システムにて取得されたログデータをそれぞれ示す。

図 6.33a には目標関節角度軌道が、図 6.33b には計測された体幹姿勢の RPY 角がそれぞれ示されている。動作指示としては直立するのみで他の動作指示は行っていないため、能動的な関節軌道は生成されていない。しかし、前述のように体幹姿勢の傾きに応じて、姿勢を水平に復元する向きに関節角度指令値が修正されるため、図 6.33a に示すように、RPY 角の変化に合わせて、関節角度の指令値も微小変化していることが分かる。

また、図 6.33c と図 6.33d には、目標 ZMP と計測された実 ZMP の腰リンク相対位置をそれぞれ X 軸、Y 軸方向に分けて示している。前述のように動作指令は無いため、目標 ZMP 軌道は0 を維持しているが、操作者が加える外乱によって実 ZMP は前後左右に移動していることが分かる。前後左右とも実 ZMP 値が 0.1m 付近まで移動しており、ロボットの立幅が 0.2m であることを考えると、ほぼ片足にロボットの自重が負荷としてかかるレベルまで外乱入力を与えていることが分かる。図 6.33e と図 6.33f に示す6 軸力センサの出力より、足裏反力は外乱によって多少の変動はあるが、接地状態は保たれていることが分かる。

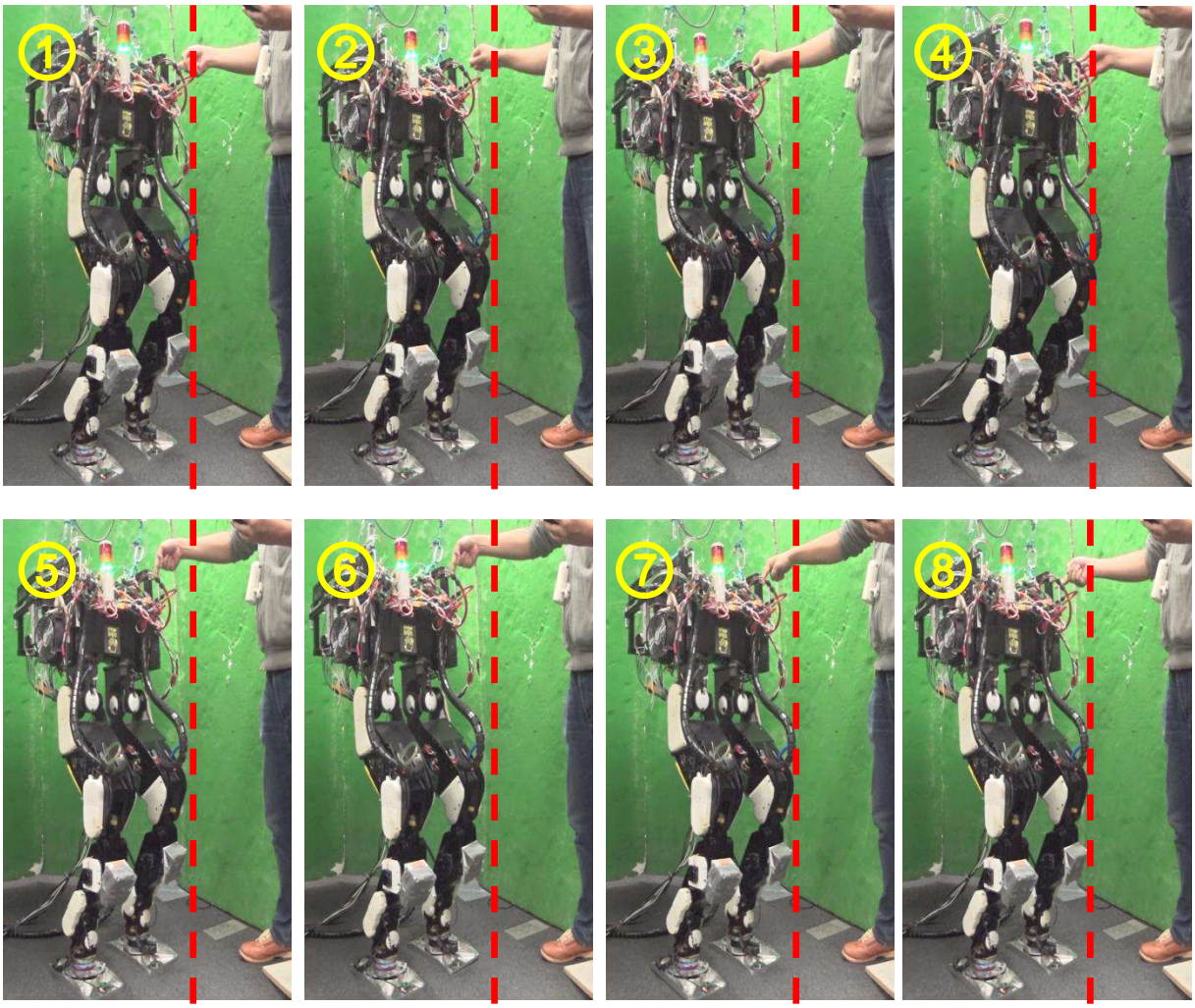
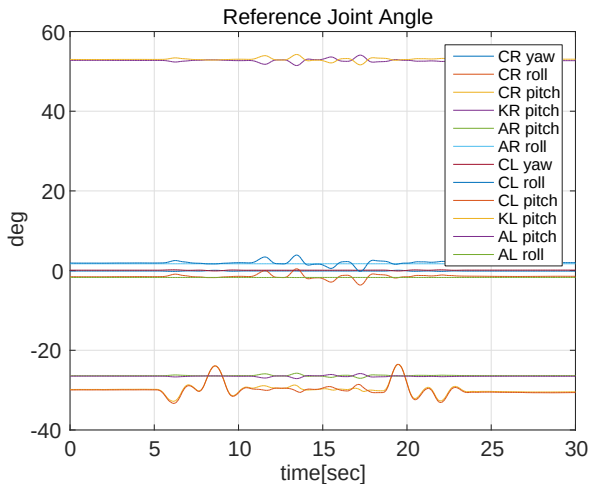


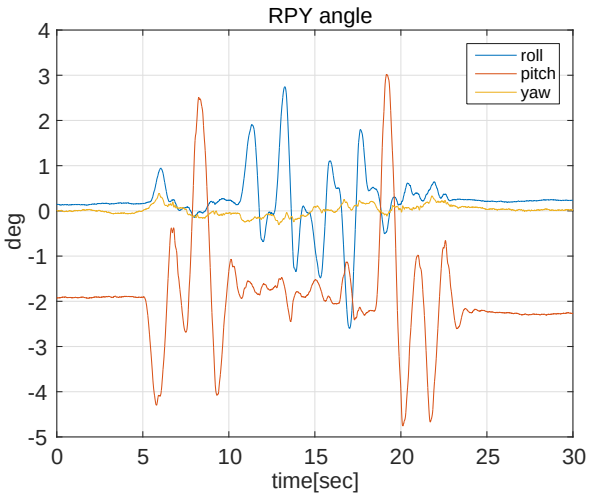
図 6.32: Photographs of standing by torque mode experiment.

次に図 6.34a 及び図 6.34b は、上位層のオンライン歩行生成器内にて LIPM から計算される目標足裏力から計算される各関節の目標トルクを示す。このトルク値は、自重を支えるための足裏反力を発生させるためのトルクであるが、状態の姿勢変化と関節角度の変化を検知して適切な値をフィードバックしている。また、図 6.34c 及び図 6.34d は関節角度の指令値との誤差によって生じさせる弾性項トルク計算値である。外乱による姿勢変化を復元する弾性項として機能し、脚の立位バランスを保持するためのトルクを出力していると考えられる。

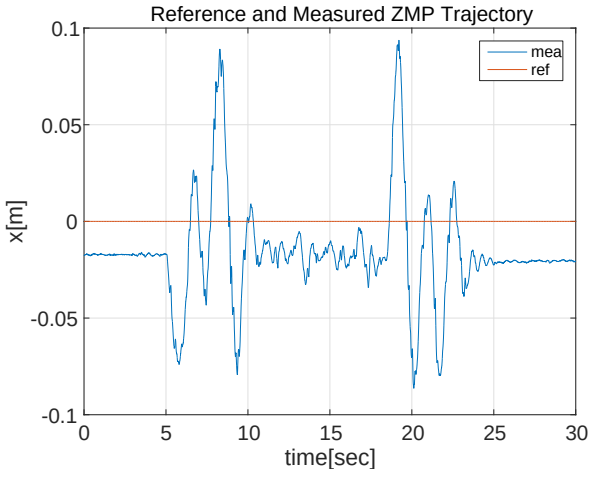
図 6.34e 及び図 6.34f は、上記のトルク値を含めて参照トルク生成モジュールにて生成された正味の参照関節トルク値となる。この値にさらにアクチュエータモデルによる補償トルクを加えた値が、トルクサーボ制御の目標値として送られる。正味トルク値は、最も負荷の高い膝関節 (KL/KR) を見ると自重を支える足裏反力発生トルクが支配的となっており、姿勢変化に合わせた弾性トルク項によって変位するという形になっている。



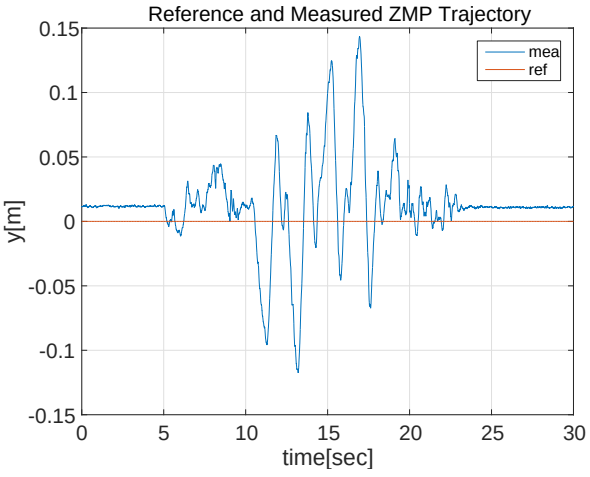
(a) Reference joint angle.



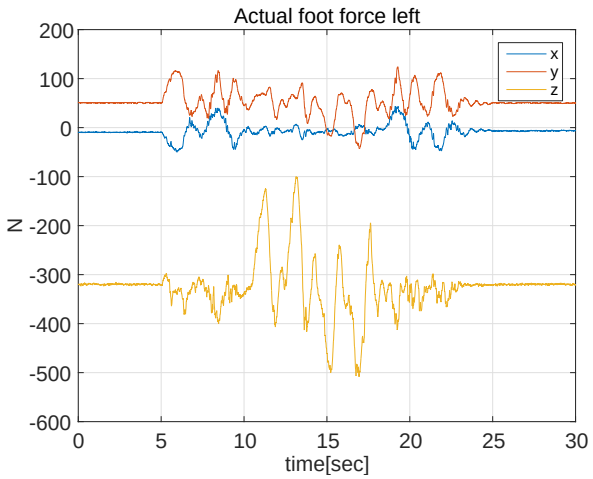
(b) RPY angle.



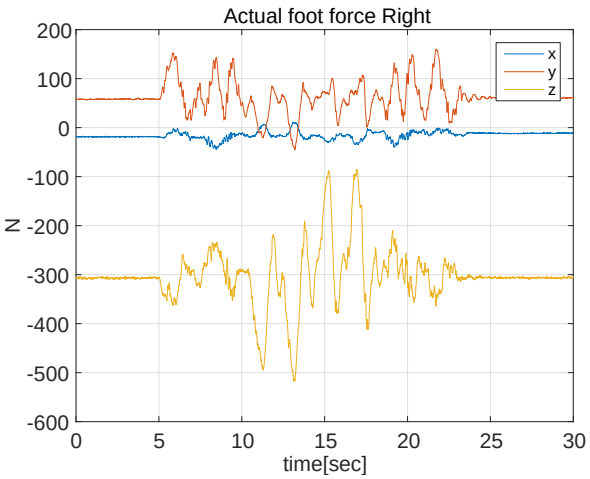
(c) Measured relative ZMP X.



(d) Measured relative ZMP Y.

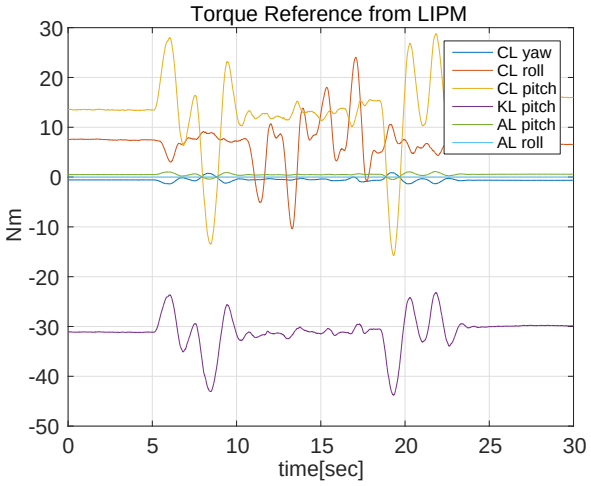


(e) Measured foot force left.

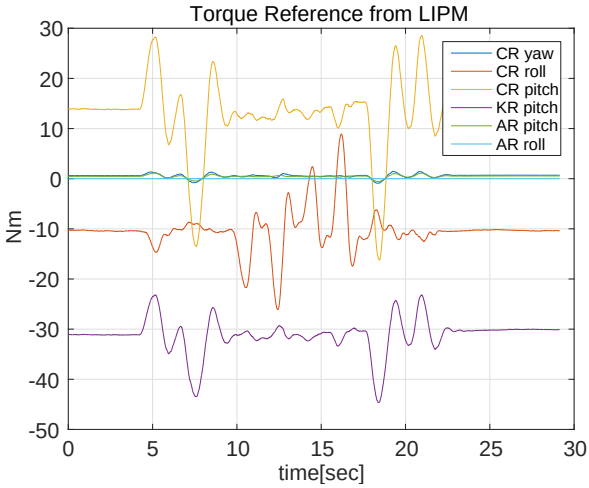


(f) Measured foot force right.

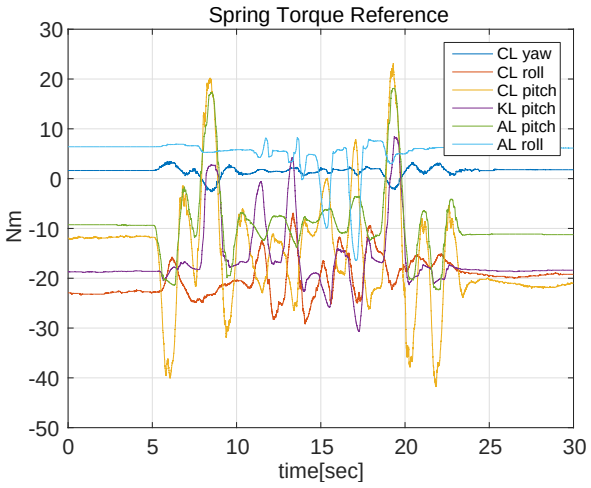
図 6.33: Logged data of standing by torque mode experiment on upper layer system.



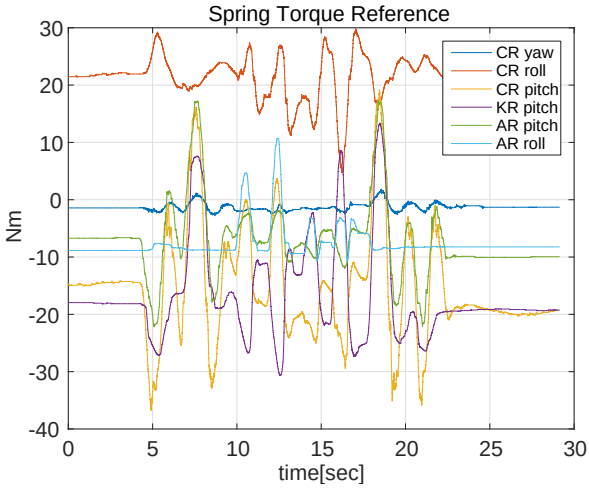
(a) FF Torque reference left.



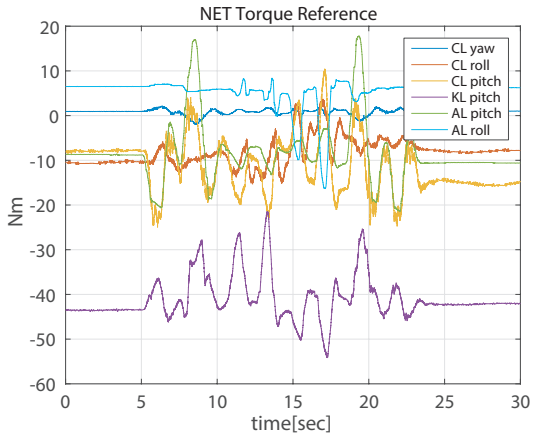
(b) FF Torque reference right.



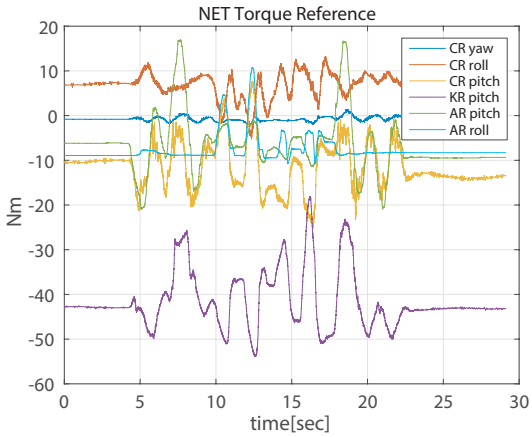
(c) Spring torque reference left.



(d) Spring torque reference right.



(e) NET torque reference left.



(f) NET torque reference right.

図 6.34: Logged data of standing by torque mode experiment on middle layer system.

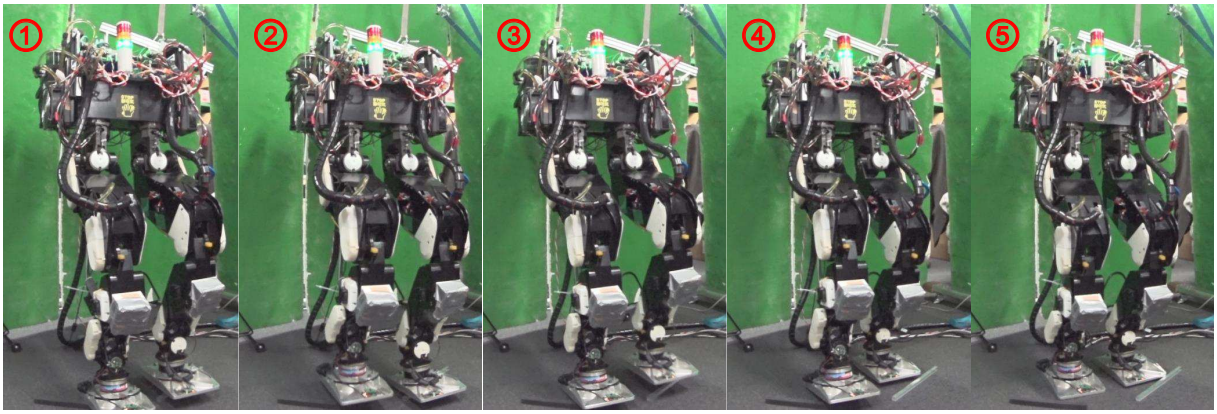


図 6.35: Sequence photographs of balancing experiment on forward step.

6.3.2 フットステップ修正による不整路面歩行における転倒防止実験

実験概要

ジョイスティックから小幅前進・並進歩行指令を入力し、20mm 角のゴム棒と 15mm 角のアルミ C チャンネル材によって構成される不整路面を横切るよう動作生成する。この状態で足部が不整物体に乗るとバランスが崩れて体幹姿勢に傾きが生じるが、ZMP 修正に基づく転倒防止フットステップがオンラインで生成されるため、転倒を回避することが期待される。

また、本実験ではトルクスタビライザによる姿勢復元を利用するため、不整路面に対する足部の接地性を向上させるために、足首の剛性パラメータについて表 6.5 に示す値に対して、次の表 6.6 のように 200[Nm/rad] だけ低下させた。

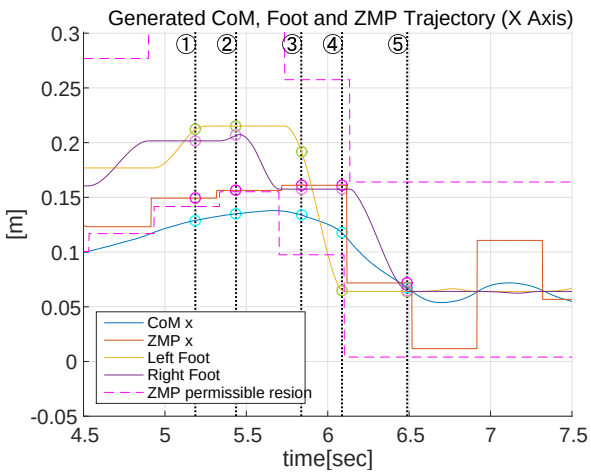
表 6.6: Configuration of spring and dumping parameter.

Joint Name	Spring Coefficient [Nm/rad]	Dumping Coefficient [Nm/(rad/sec)]
CL/CR Yaw	1200	18
CL/CR Roll	1200	18
CL/CR Pitch	1200	18
KL/KR Pitch	1200	18
AL/AR Pitch	300	8
AL/AR Roll	300	8

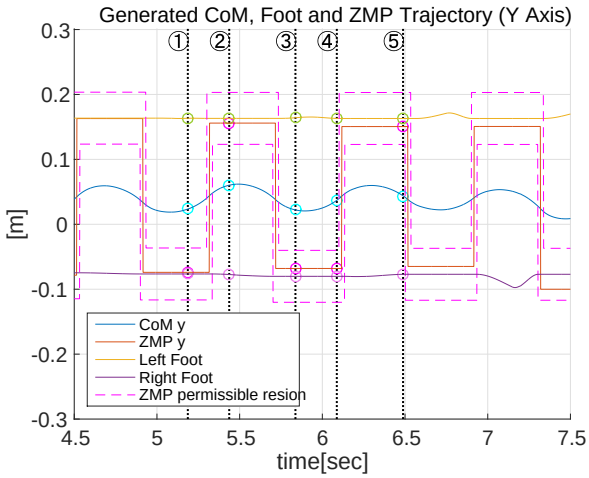
前後方向転倒回避実験結果

図 6.35 に実験中フットステップ修正による転倒防止動作が起きた際の連続写真を示す。また、図 6.36 、図 6.37 、はそれぞれシステムにて計測されたデータを表示したものである。各図中の①から⑤は、それぞれ図 6.35 中の番号の瞬間と対応させている。

図中①及び②においては左右脚でそれぞれ角材を踏んでいるが、姿勢安定化制御による出力トルクによってバランスを保っている状態である。この出力トルクは図 6.38b の y 軸まわりに発生している 50Nm 以上のモーメント指令値によって与えられている。この出力トルクは、図 6.37a に示す前後方向の計測 ZMP に表れている、支持脚期間の初期から中間あたりまでにかけての計測 ZMP の大きな後方への移動に効果を発揮し、これにより後方への重心移動を抑制していると考えられる。



(a) X axis Target CoM and ZMP Trajectory.



(b) Y axis Target CoM and ZMP Trajectory.

図 6.36: Recorded data of Target CoM and ZMP trajectory. ZMP permissible region is the area a target ZMP potentially be placed in.

図中③は姿勢安定化制御によるバランス補償が不十分で後方への転倒が始まった点となる。図 6.38a に示す IMU による体幹部の姿勢推定の値から、③のあたりで8 度程度後方へ傾いたことが確認される。

図中③での姿勢の変化を受けて、実時間歩行動作生成器は①から③にかけて前方へと移動させていた ZMP 目標軌道を④にて 0.1m 程度後方へと修正し、合わせて左脚を 0.15m ほど後方へ軌道修正していることが図 6.36a 確認される。さらに続けて⑤において ZMP 軌道を 0.05m 程度後方へ修正し、右脚軌道を左右脚の前後位置がそろう程度に修正を与えている。この段階で後方へと大きく傾いた体幹姿勢は、図 6.38a の⑤に示されるように、バランスを保っていた①及び②の点における値まで復帰していることが確認される。

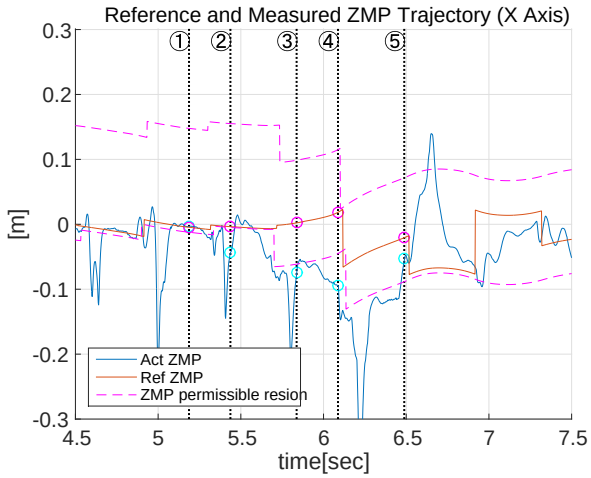
また、左右方向については図 6.37b のフットステップ修正が発生している④から⑤にかけて、接地タイミングのずれ等の影響で計測 ZMP と目標 ZMP の乖離が大きくなっているが、図 6.36b の目標重心軌道、目標脚軌道、図 6.38a の体幹ロール角の値に表れるように、ほとんど姿勢の崩れは発生しておらず、適切に姿勢安定化が働いていることが確認された。⑤の直後からややロール角度の増大が見られるが、図 6.38b の x 軸まわりに発生している復元モーメントによって姿勢変化が抑え込まれていることが示されている。

以上から、姿勢安定化制御による不整路面に対する最適足裏反力の生成によるバランス修正と実時間歩行動作生成系による転倒回避行動のための軌道修正について、互いに機能することで効果的に歩行動作の継続を実現できた。なお、歩行指令は常時前方への移動であるため、理想的には後方への転倒がそもそも発生しないように姿勢安定化制御や重心軌道が生成されるべきである。この点については、制御則の改善や足部構造を靴底のような柔軟素材化する等の接地性の改善を図る必要があると考える。

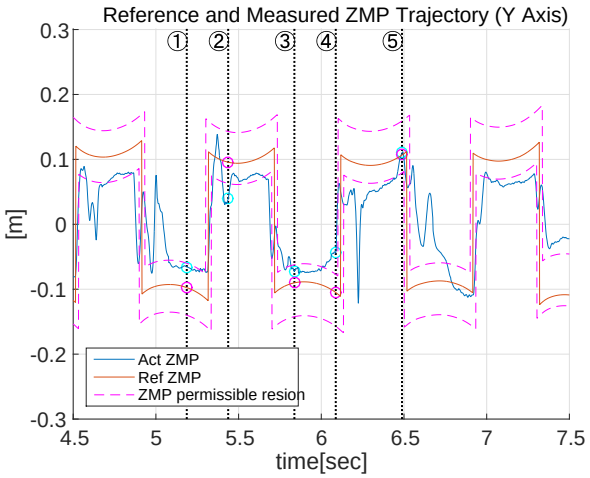
6.3.3 遊脚外乱に対する緩和動作の定量的評価

実験概要

本実験では、歩行中の脚に対する外乱に対して即応的反射行動アーキテクチャに基づいたなじみ動作によって外乱の緩和動作が機能することを実証するための定量的な評価として、重量球を振り子の原理を利用して一定速度で衝突させた際の応答を計測する実験を行う。球は表 6.7 に示す、硬質ゴム製のメディシンボールを使用した。この球をロープで吊るし、落差がおよそ 300mm 以上となる位置から最下点にて遊脚の膝下に備え付けられているクラッシャブルパーツに衝突させる。この場合、力学的エネルギーの保存則を用いると衝突時の球の速度は 8.7[km/h] 以上になると計算される。実際にはロープの摩擦や質量等が無視できないため、実衝突速度はこの8 割程度の 5-7[km/h] 程度になると推測される。この速度は早歩き程度の速度であり、歩行時に据え置かれた障害物と衝突した際の相対速度に近い実験条件になっていると言える。衝突時の実速度は、側方から撮影した動画を元に推定する。撮影に使用したビデオカメラのフレームレートは 30fps である。

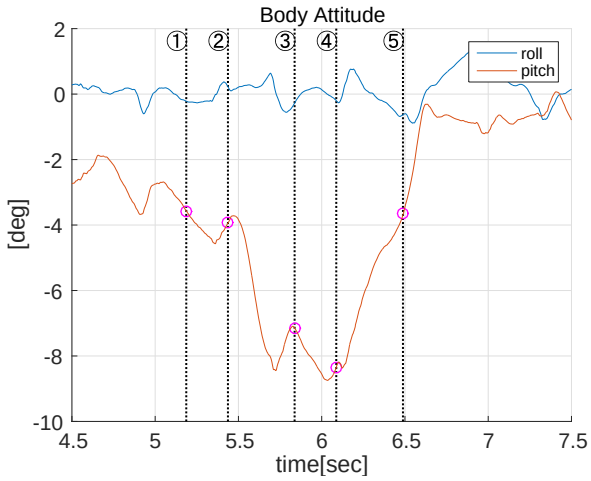


(a) Measured relative ZMP X.

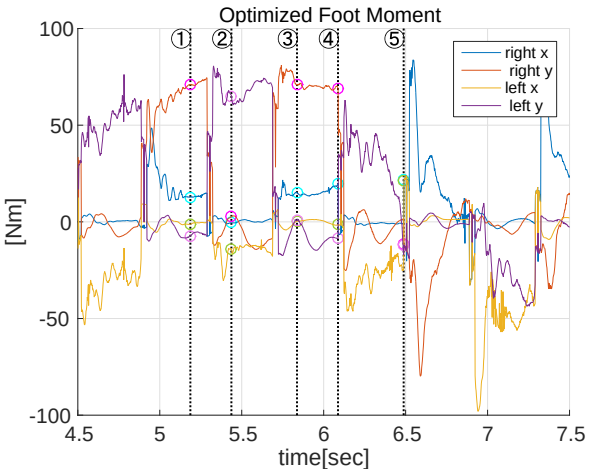


(b) Measured relative ZMP Y.

図 6.37: Reference and Measured relative ZMP from body.



(a) Recorded data of body attitude roll and pitch from IMU.



(b) Moment x and y axis.

図 6.38: Recorded data of optimized body balancing foot moment and body attitude.

ロボットは、球の最下点にて歩行周期 400[msec] の足踏みを行わせる。球はロボットの前方から後方へ向かう方向に、右脚の膝下部に衝突させるよう設定した。ジョイスティックからは、足踏み指令のみを送信し、移動指令は送信しない。ただし、実時間重心軌道生成器におけるオンラインでの着地位置修正は有効であり、これによる並進移動は指令と無関係に自律的に行われる。

表 6.7: Specifications of the ball used in this experiment.

Diameter	0.270[m]
Weight	5.0[kg]
Material	Hard rubber

実験結果

本実験を撮影した動画から切り出した連続写真を図 6.39 に示す。図中 1 から 13 までの各図の対応する時刻は、以下に示すグラフ中で①から⑬の時刻に対応させた。①から④は、球がロボットの膝に衝突する直

図 6.39: Sequence photographs of the hitting ball to the swing leg experiment.

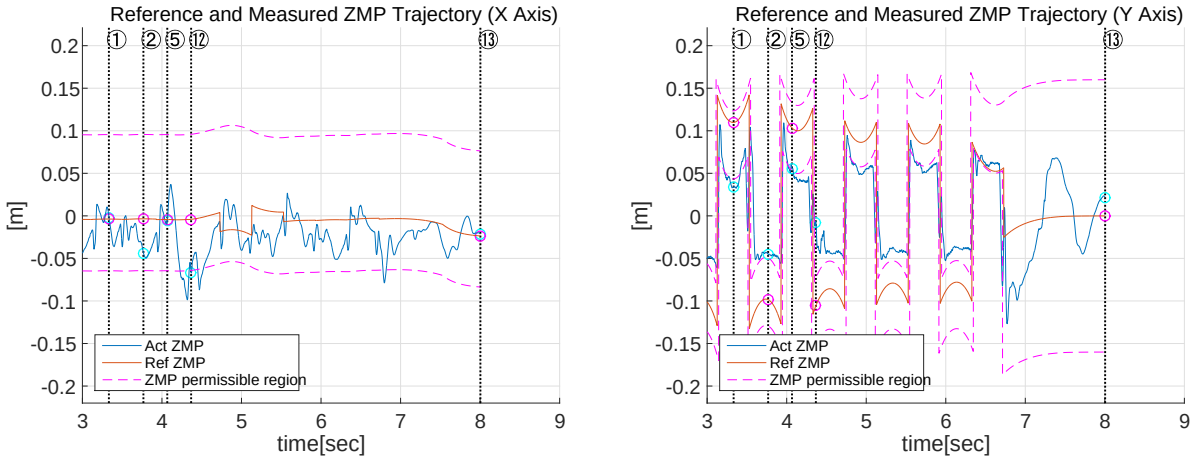
前までを示しており、図中⑤のタイミングにてボールとロボットの右遊脚膝部との間で衝突している。⑤から⑪までは衝突過程をより詳細に確認するために撮影された 30fps の動画の毎フレームを抜き出している。⑫は、球が衝突した遊脚が支持脚に切り替わったタイミングを示す。また、⑬は衝突後、転倒等の問題が発生せず、正常に足踏み動作を終了したタイミングを示している。

⑤にて衝突した球は、次の⑥に示されるフレーム以降でほぼ停止状態になったため、球はロボットの膝下との衝突によって、1 フレーム以内の時間で衝撃を吸収して速度が0 になったと考えられる。球の直径が 270mm であることを基準に、図 6.39 の④と⑤の比較から、衝突直前の球の速度を推定するとおよそ 5.8-7.3[km/h] の速度が出ていた計算となった(表 6.8 参照)。この速度は落差から計算される想定衝突速度 9.1-9.5[km/h] の 6-8 割程度であり、実験概要にて述べた想定衝突速度と概ね一致する速度となっている。上記速度で 5.0[kg] の球が衝突し、1 フレーム内に停止したと仮定した場合、膝下部が受ける平均外力は運

表 6.8: Measured values used to calculate the ball speed at the collision.

Travel length at one frame	54.0-67.5[mm]
Gap of fall	330-358[mm]
Time per frame	1/30[sec]

動量と力積の関係からおよそ 240-300[N] 程度であったと推定される。



(a) Measured relative ZMP X.

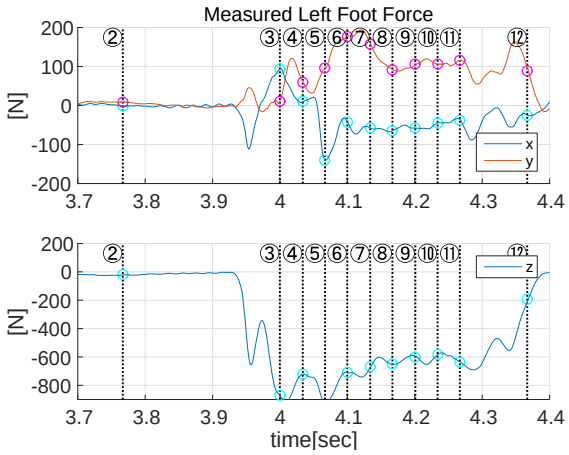
(b) Measured relative ZMP Y.

図 6.40: Recorded data of Target CoM and ZMP trajectory on ball hitting experiment. ZMP permissible region is the area a target ZMP potentially be placed in.

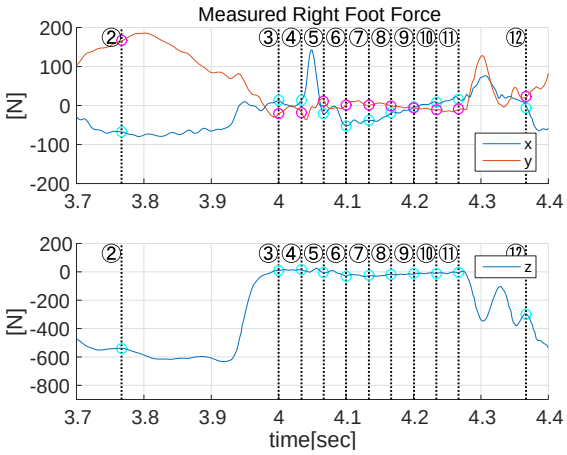
図 6.40a に前後方向の腰部相対目標 ZMP 軌道及び計測された ZMP 軌道を示す。球が衝突した⑤から⑫にかけて計測 ZMP が足部の ZMP 許容領域後端を超える程度まで目標軌道から外れてしまっている。この影響によって図中5 秒前後に ZMP 軌道の修正が加わっており、これによって次のステップまでには目標 ZMP 軌道に復帰し直したことが分かる。また、図 6.40b に左右方向の腰部相対目標 ZMP 軌道及び計測された ZMP 軌道を示す。球が衝突した⑤から⑫にかけて、及びその後の時間においても大きな軌道の誤差は見られない。以上より、球はロボットの前方から衝突させたことを考慮すると、ロボットの歩行生成系及び姿勢安定化制御系は、球の衝突による前後方向の誤差に対して歩行軌道修正を素早く行うことで対応し、左右方向等の制御に影響を及ぼさなかったことが分かった。

図 6.41a 及び図 6.41b は足部 6 軸力センサの衝突前後付近の出力を表示している。④から⑤にかけて右脚に衝突していることが図 6.41b の x 軸出力から判別可能となっている。右脚の足部には球は直接接触はしていないが、衝突時の衝撃加速度によって生じた足部慣性力が計測されたものと考えられる。支持脚である左脚側のセンサでは衝突から1 フレーム (33msec) 後の⑤から⑥にかけて、後方へ 150[N] 程度の撃力を計測しており、これはロボットの体全体が後方へと傾いた運動を姿勢安定化制御によって復元させるための左脚足部トルクによって生じた力であると考えられる。実際に図 6.42b に示す姿勢安定化制御器における左脚の目標出力トルク値は④から⑤にかけて前方への− 66[Nm] 程度の復元トルクが出力されており、この出力が1 フレーム遅れでセンサに計測されていると考えられる。

また、図 6.43 に示す右脚ピッチ軸周り関節の角度誤差を見ると、④から⑤にかけて腰部ピッチ軸関節は衝突の瞬間約 0.75 度変位していることがグラフより確認される。膝部接触点と腰部ピッチ軸関節の距離は約 485mm のため、この 0.75 度の緩和動作は接触点における約 6.3mm 程度の変位に相当する。衝突点と膝関節は約 156mm と近いため、腰部関節と比較すると外力によるモーメントが十分ではなく、膝関節での衝撃緩和動作は起こらなかったと考えられる。以上の関節動作による緩和に加えて、直接計測はできていないがロボット体全体の変位による緩和効果も重畳して、球の衝突による撃力を緩和していると言える。これは、図 6.44 に示す IMU から推定された体幹部姿勢の軌道について、衝突直後に前方に1 度程度の急激な傾きが生じていることから推測することが可能である。重心周りにこの回転が生じたと仮定すると、ロボット全体の傾きによる膝下部の変位はおよそ 8.5[mm] に相当する。これは先ほどの腰部関節の衝突緩和動作による変位量に相当する。このことから、単純に衝突部位に関する関節のしなやかさによる衝突緩和のみでなく、ロボットの体全体で撃力に対してある程度受け流すことを想定することが重要であると言える。本システムはこのロボットの体全体での変位による衝突緩和によって生じるバランスの崩れに対して、上位の身体運動最適化制御ループによって補償することで、動作全体としての継続性を維持する。この実験では、図 6.44a の⑤において生じた傾きを、次のステップである⑫以降で復元しており、これは上述の図 6.42b に示された姿勢安定化制御の支持脚復元トルクの効果と図 6.40a に示した重心軌道の修正の効果の複合的な作用によるものと考えられ、本システムの制御的な頑健性を実証していると言える。

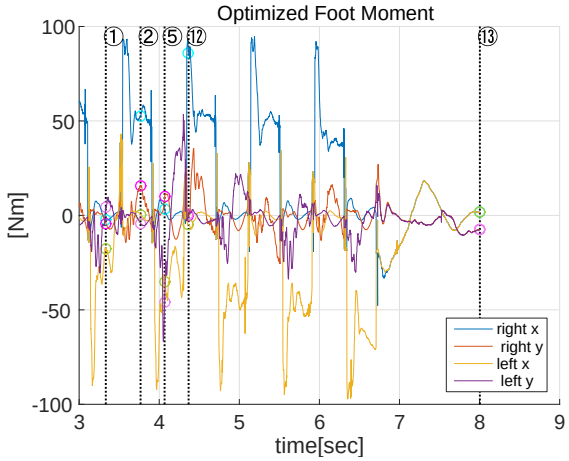


(a) Measured left foot force.

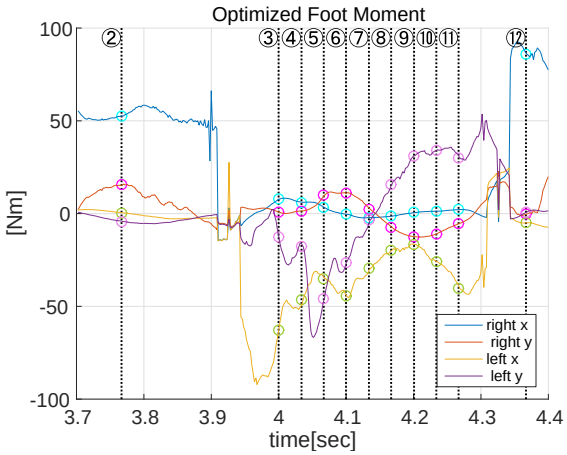


(b) Measured right foot force.

図 6.41: Recorded data of foot forces by six-axis force sensor in ball hitting experiment. This graph shows around the conflicting timing.

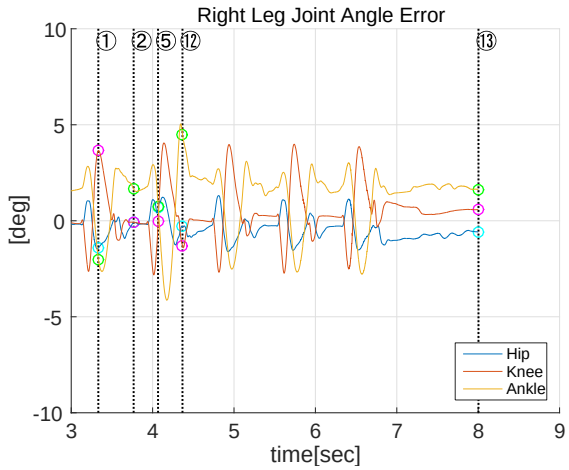


(a) Optimized reference foot moment.

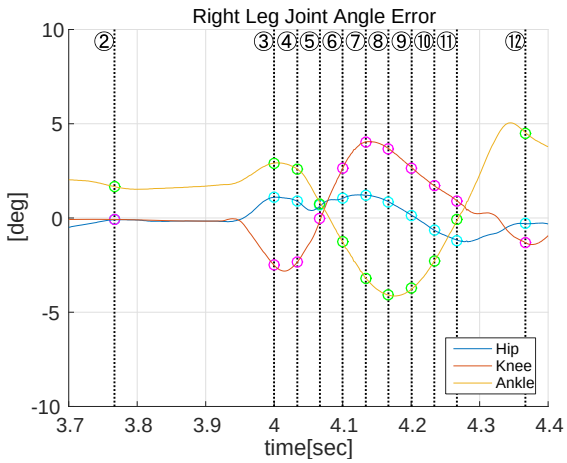


(b) Enlarged around the conflicting timing.

図 6.42: Recorded data of optimized reference foot moment in ball hitting experiment.

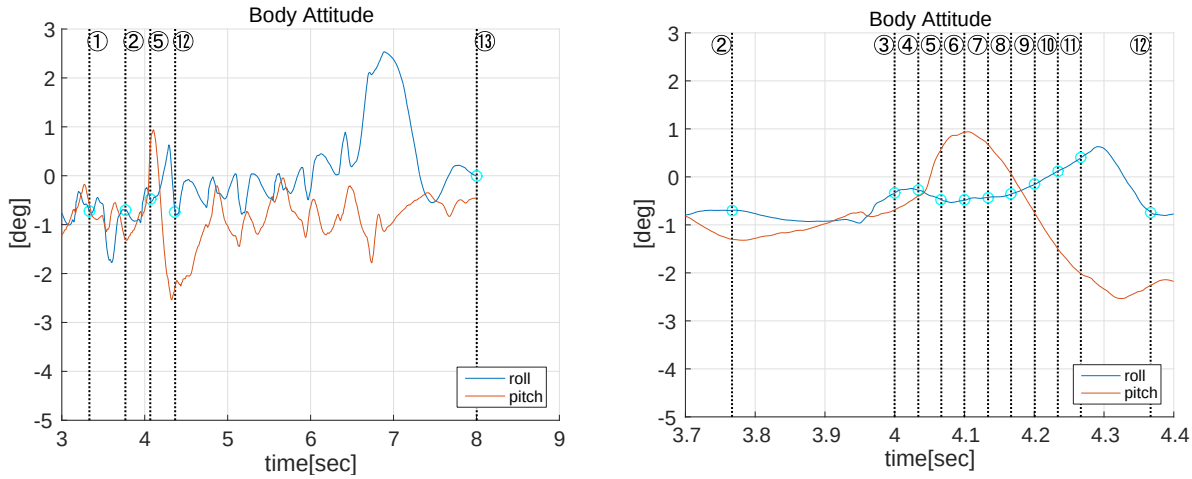


(a) Measured joint angle errors on right leg.



(b) Enlarged around the conflicting timing.

図 6.43: Recorded data of right leg joint angle errors in ball hitting experiment.



(a) Measured body attitudes.

(b) Enlarged around the conflicting timing.

図 6.44: Recorded data of body attitudes by IMU in ball hitting experiment.

6.3.4 足払い外乱に対する緩和動作と転倒防止動作実験

実験概要

前節では歩行中のロボットに対して球による定量的な外乱を与えたが、本設ではよりランダムな外乱入力への対応性を測るために人間がロボットに対して足払いを試みる実験を行う。これは歩行動作を行っているロボットに対して、外部から足払いとなる強制外乱を加えた際に、トルクベース制御による関節の柔らかい即応的反射行動による外乱の抑制、並びに外乱抑制によって生じるバランスの崩れに対して身体運動計画系における転倒回避動作の生成によって回避動作が成されるか確認を行うものである。歩行動作はジョイスティックから、その場での足踏み動作指令のみを送り、並進移動指令は送らない。この足踏み状態のロボットに対して、実験者が外乱としてロボットの足部に足払い外乱を与え、バランスを崩すことを試みることで挙動を確認する。剛性パラメータについては 6.3.2 節にて示した値と同値を用いた。

実験結果

図 6.45 に示す連続写真は、実験中にロボットの脚に足払い外乱を加えた際に実行された外乱抑制動作並びに転倒回避動作の様子を示したものであり、全体の様子と足部を写した図の 2 種を合わせて並べている。また、図 6.46 、図 6.47 、はその時間区間にそれぞれシステムにて計測されたデータを表示したものである。各図中の①から ⑨は、それぞれ図 6.45 中の番号の動画が撮影されたタイミングと対応させている。

図中①から ③においてロボットは左脚支持から右脚支持へと移行する動作を行っており、正常にバランスが維持されている状態である。③から ④にかけて、左遊脚足部に対して足払い外乱を加えており、その結果、図 6.48a に示す体幹姿勢のロール角度が −4 度程度まで傾いていることが分かる。ただ、図 6.45 の ④に写されているように、遊脚は外乱に対して緩和動作によって後ろへと移動しており、これによって図 6.47 に示す計測 ZMP 軌道は目標軌道から大きく外れるという状態を回避していることが確認される。実際、図 6.49 に示す足部に取り付けた6 軸力センサ出力値から計算された左右各脚の足部にかかる外力を見ると、足払い外乱を加えられている ③から ④にかけて前後方向 (x 軸方向) 及び左右方向 (y 軸方向) では最大 54[N] の計測値であり、これらは支持脚時の歩行のために出力する値 100[N] 程度と比較しても小さい値であるため、遊脚の衝撃緩和効果が十分に働いていることが示されている。また上下方向 (z 軸方向) では下方へ最大 141[N] の外力が生じているという計測値を得ているが、この外乱が上述の図 6.48a に示す体幹姿勢のロール角度の傾きに表れていると考えれらる。

④から ⑤にかけて、外乱を受けた左脚の目標軌道は修正され、0.0196[m] ほど外側へと着地位置を修正していることが、図 6.46b の左脚軌道の記録から分かる。遊脚の着地位置は、足部軌道の速度が過大とならないよう、着地までの残り時間によって制限されるため、この時点での着地位置ではバランスの崩れの修正は不完全であったと考えられる。そのため、⑤から ⑥にかけて、右脚を内側に寄せ、さらに次の⑥から ⑦

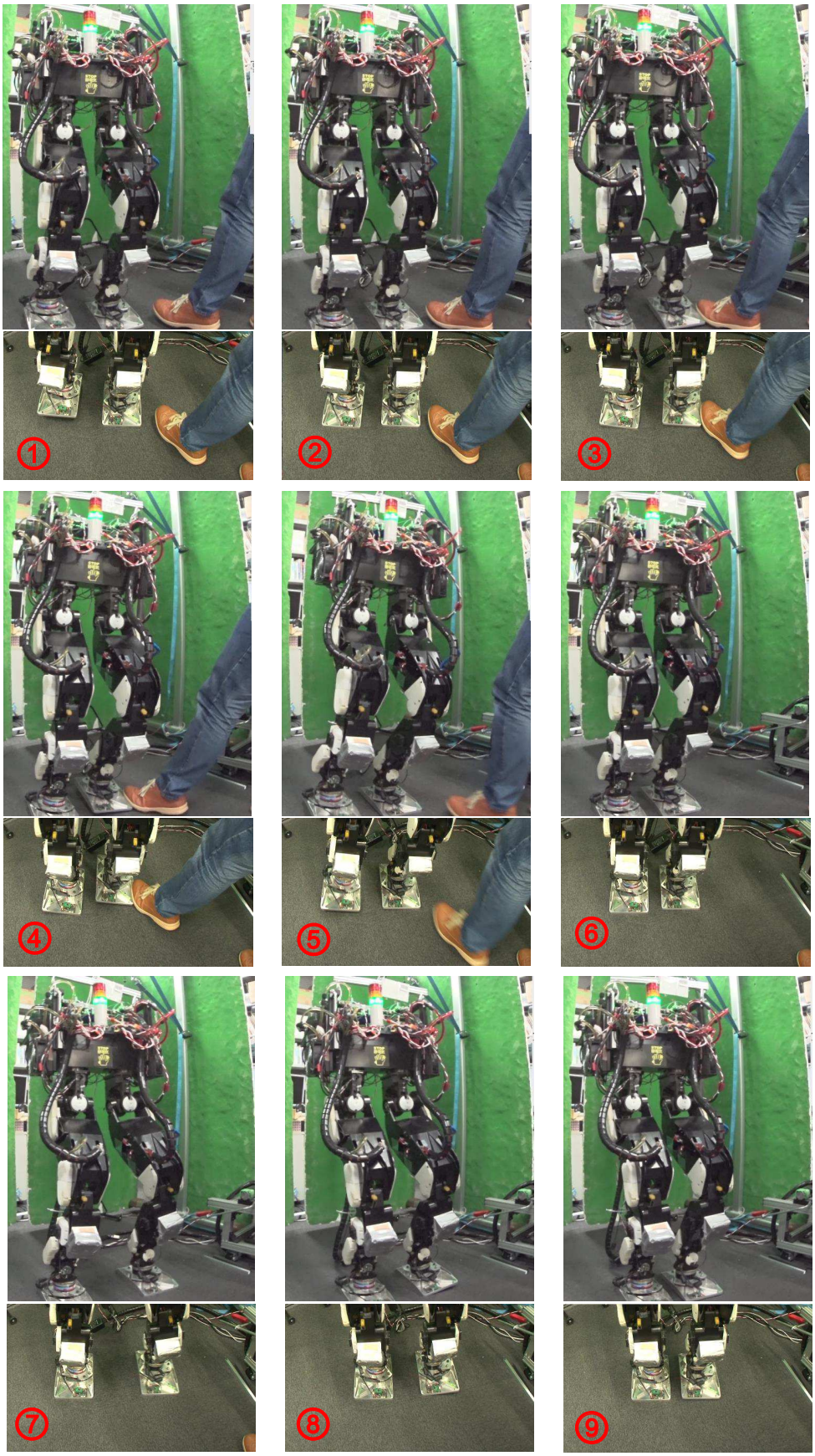
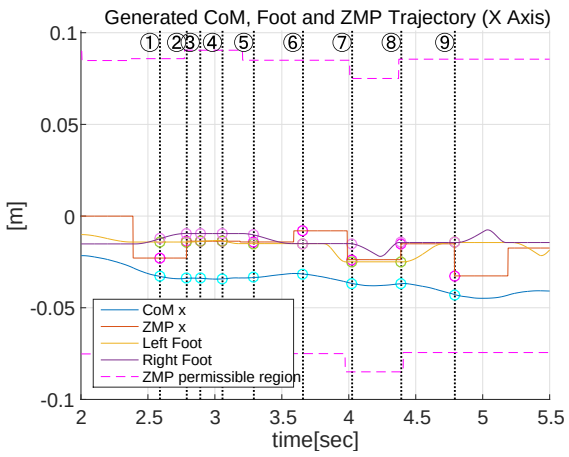
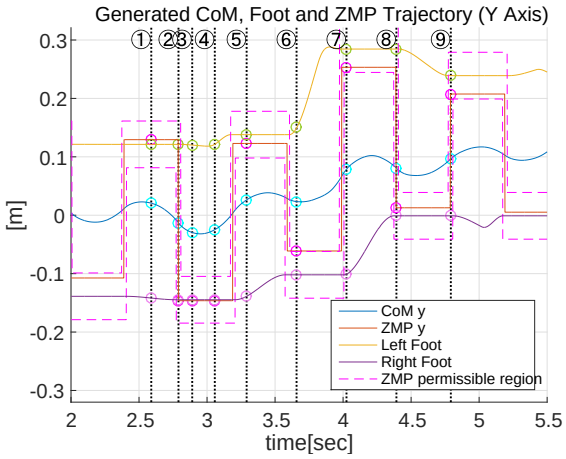


図 6.45: Sequence photographs of foot disturbance experiment.



(a) X axis Target CoM and ZMP Trajectory.



(b) Y axis Target CoM and ZMP Trajectory.

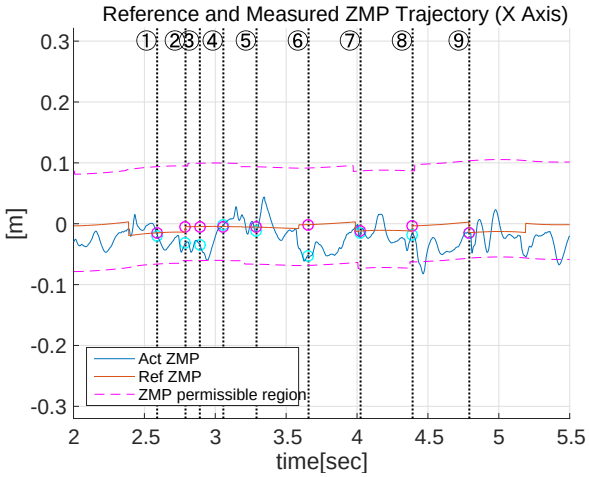
図 6.46: Recorded data of Target CoM and ZMP trajectory on foot disturbance experiment. ZMP permissible region is the area a target ZMP potentially be placed in.

にかけて左脚を大きく外側に 0.147[m] ほどフットステップを生成することでバランスの修正が行われたことが分かる。⑦から⑨にかけて、左へ大きく移動した重心の減速フェーズとして働いていると考えられる。

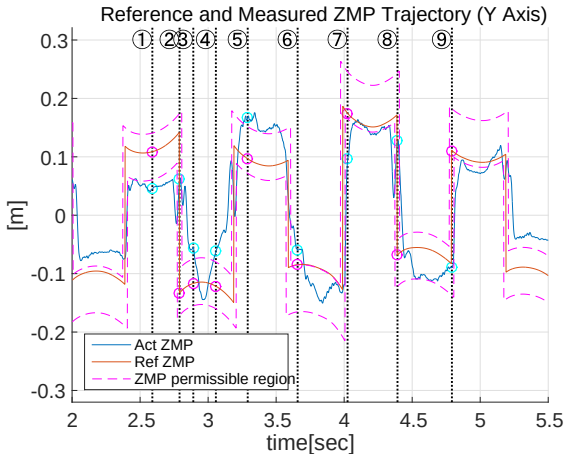
前後方向については、図 6.46a の ZMP 軌道に示される様に大きな着地位置修正は発生しなかった。また、図 6.47b に示す姿勢安定化のための足裏反力について、特に前後の運動に関係するピッチ軸まわりの y 軸モーメント値は 50Nm 未満と低い値で実行されている。同様に図 6.47a に示される前後方向についての計測 ZMP も大きな乱れは確認されない。このことから、足部に加えた外乱に対して即応的反射行動によって緩和動作が実現されたことによって、前後方向の外乱入力は打ち消すことができたと考えられる。特に脚の関節自由度の配置の都合上、足部前後方向は腰、膝、踵の 3 軸の緩和動作によってそれぞれで吸収できるため効果が高かったと考えられる。左右方向については、膝関節の自由度が利用できない点で不利であると考えられるため、腰関節のロール軸まわり剛性パラメータを柔らかくする方向で調整代があると言える。

以上の動作から、外乱に対してなじむことで入力を緩和する即応的応答行動が機能していること、及びなじみ動作によって本来の歩行動作軌道から外れた位置に着地したことや、十分に緩和しきれなかった外乱によって生じるバランスの崩れに対して、実時間での転倒回避歩行動作生成によるバランス修正機能がお互いに正しく機能することが確認された。

また、本実験では遊脚期間と支持脚期間で関節剛性パラメータは変更していないが、遊脚期間中はつまづきのような外乱に対しての緩和動作の効果を大きくするために、剛性パラメータを動的に低減するといったことが考えられる。

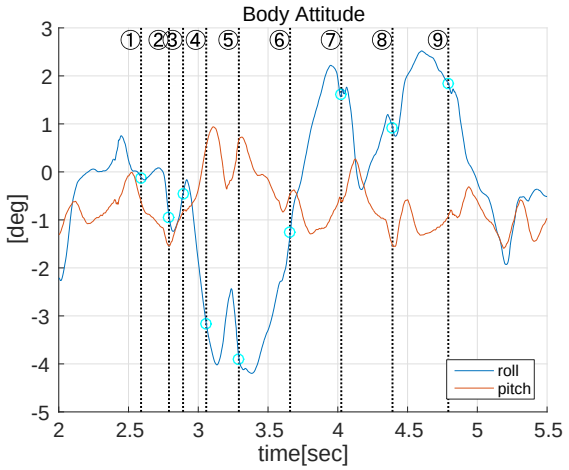


(a) Measured relative ZMP X.

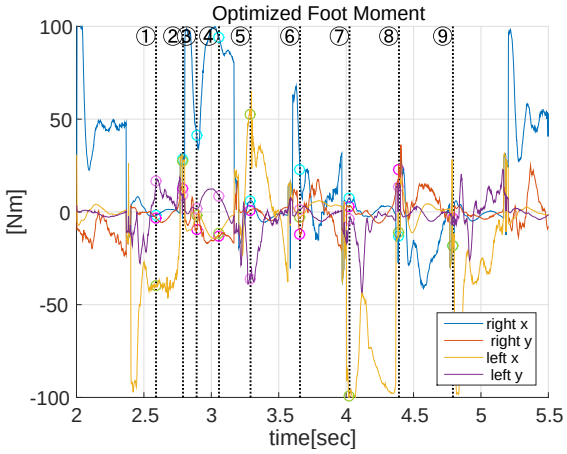


(b) Measured relative ZMP Y.

図 6.47: Reference and Measured relative ZMP from body on foot disturbance experiment.

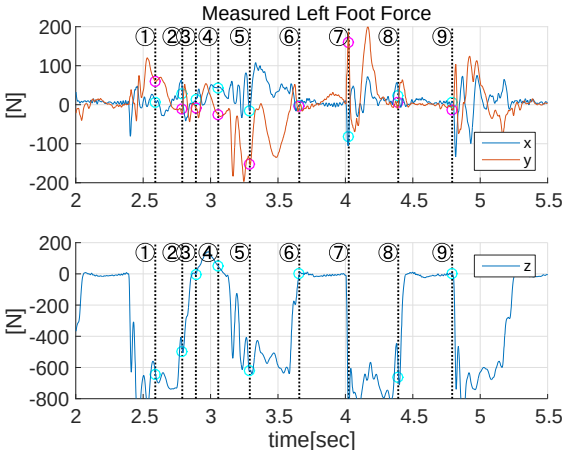


(a) Recorded data of body attitude roll and pitch from IMU.

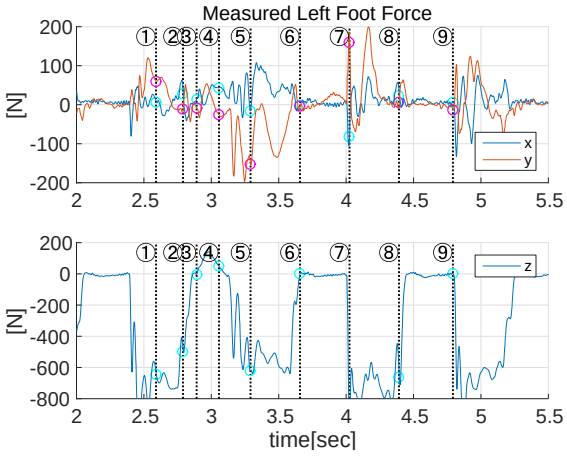


(b) Moment x and y axis.

図 6.48: Optimized body balancing foot moment and body attitude on foot disturbance experiment.



(a) Measured data of Six-axis force sensor on the left foot.



(b) Measured data of Six-axis force sensor on the right foot.

図 6.49: Measured foot force on foot disturbance experiment.

6.4 本章のまとめ

本章では、本研究にて提案する即応的反射行動アーキテクチャ分散型制御システムの構成方法に則った実ロボットにおけるシステムを構成し、その動作実験によって実証を行った。

腕型ロボットの A0-B spec にて、スイング動作を行うことで中間層制御システムにて実行されるトルク制御をベースとした即応的動作制御系の応答性検証を行い、物体との衝突時に生じる衝撃外乱に対して、なじみ動作として応答が可能であるか確認を行った。この実験では、3 パターンのスイング速度にて実験を行い、低速から高速まで広い速度レンジにおいて、外乱となる実験者の手との衝突に対して、柔軟になじむ挙動を示した。逆動力学計算とアクチュエータモデルによる関節トルク補償演算によって、動作に必要最低限の出力トルクを実現できていることを実証した。高速な動作領域での衝突においても、衝突物体や自身の構成部品の破損を生じない接触ができる可能性を示した。

また、外乱検出のために、衝撃センサ、6 軸力センサの2 種のセンサを活用する方法と、動力学モデルから計算される運動量積算地の誤差値から評価される外乱検出手法をそれぞれ比較し、外乱検出性について考察を行った。高速域での衝突には衝撃センサ、低速域での衝突には動力学モデル、エンドエフェクタとの衝突時には6 軸力センサとそれぞれで使い分けることで、広い速度レンジや接触場所について対応が可能となることを示した。これらはシステムの分散化によって特に高速な周期でのサンプリングと演算を行うノードを別途システム内に組み込むことが可能になったことが貢献として大きく、今後のロボット展開において重要な機能の一つとなる。また、検出精度を向上させるためには、パラメータの同定手法を適用する必要があることについても触れ、今後のロボットの構造設計の指針を与える結果でもある。

さらにサーボ制御の演算周期について 1000Hz と 5000Hz で比較を行い、制御周期の高速化によって逆動力学演算によって生成するトルク軌道による、目標関節角度軌道への追従性能が向上することが確認された。

次に脚型ロボットを用いて、身体運動計画系と即応的動作制御系の2 つによる即応的反射行動アーキテクチャによって従来のシステムで行われていた歩行動作を含めて、基本制御システムとして実現可能であることを検証した。オンライン歩行生成器から生成された関節角度軌道によって、平面床上において歩行動作が可能であることを実験により実証した。この実験では、実機とモデルとの誤差によって生じるバランスの不安定化を補償する安定化制御を無効化して実現しており、オンライン歩行生成器のフィードフォワードな ZMP 補償手法の有効性を確認した。また、7mm から 15mm 厚程度の物体の上を踏破する実験を行い、IMU から取得される姿勢角の傾きから、転倒を回避する歩行軌道をオンラインで生成し、実行することが可能であることを実験によって示した。このオンライン歩行生成器と即応的動作制御系による関節トルク・関節剛性制御及びトルクベーススタビライザを有効化することで、外乱吸収効果の検証、及びオンライン歩行生成による計画軌道との差異補償の効果が両立した状態で機能することの実証を行った。足払いといった強い外乱に対しても、衝撃の緩和と転倒回避行動を両立可能であることが実験によって確認され、本研究にて構成した即応的反射行動アーキテクチャの有効性を実証した。

第7章

結論

7.1 本研究の結論

本研究は、ヒューマノイドロボットが実環境下において継続的に作業を行うための制御システムプラットフォームの開発を目的として、高速応答可能なロボットのための体内分散通信制御システムの構成方法について考案し、実ロボットの制御システムとして設計・実装することでその実証を行ったものである。ヒューマノイドロボットが実環境下において作業を行う際の問題として、転倒や衝突による機構破損や、振動・衝撃・熱・電磁ノイズ等の影響によるシステム障害といったハードウェアが持つ脆弱性によって持続的な行動が困難となる事例があることを述べた。また、ヒューマノイドロボットの継続的な開発において、システムの動作状態やセンサからの知覚情報がすべて参照可能な状態で記録されることが重要である。

第 2 章「 高速応答行動を可能とする体内分散通信制御系の設計要件と評価項目」ではヒューマノイドロボットの分散型制御システムについて一般的な枠組みについて述べた。ヒューマノイドロボットではその身体構造に収まるシステムパッケージとして独立した状態で完結する必要があるため、既存のシステムパッケージの流用では持続的な動作を行う上で支障が生じるという課題を明らかにした。上述の目的を達成するために性能・信頼性・設計/運用・展開・テストの5 カテゴリについて 15 項目にわたるシステム設計要件・評価項目を定義した。また、5.4 節では従来システムにおける問題点について致命的問題・機能制約問題・開発効率問題の3 カテゴリについて整理を行い、上記評価項目について評価を行い、制御システムの開発のための指針とヒューマノイドロボット特有の制約を得た。

機構の破損を防止する関節の柔らかいトルク制御を導入する上で、現在の一般的な制御周期 1kHz よりも高速化することが効果的であることを示し、目標制御周期として 5kHz を掲げた。また、制御周期の高速化においてジッタの低減が問題となることについて触れ、制御プログラム実行機構と通信系が 5kHz(200μsec) 周期の 10%である 20μsec 程度に抑えることが望ましいことを示した。システムの分散化においてノード間の通信系構成は、通信品質の決定において重要となることを論じた上で、実時間性能を最重要特性として、これを実現する通信モデルの設計について述べた。

次に示すのは本研究の提案する制御システムを構成する 3 つの主要要素となる。

1. **実時間プロセッサによる分散モータ制御モジュール （ 第 3 章）**
2. **マルチプロトコル対応分散ノード間低遅延通信系（ 第 4 章）**
3. **即応的反射行動アーキテクチャに基づいた実行系システム （ 第 5 章）**

これらの要点について、以下に順に述べていく。

7.1.1 実時間プロセッサによる分散モータ制御モジュール

実時間プロセッサによるモータサーボ制御

D-RMTP の低遅延実時間実行機構による高周期低ジッタサーボ演算システムを構成した。さらに D-RMTP 20mm SiP 基板を主 MPU として搭載可能な大出力モータドライバ RMTP-02D 基板の開発を行った。スレッド切替時のコンテキストスイッチによるオーバヘッドを削減する D-RMTP のコンテキストキャッシュ機構と論理コア内に埋め込まれたハードウェアタイマ割込みによる起床機構を活用した Responsive Task によって 5kHz の制御周期で 1μsec 以下のジッタ性能を持ったサーボ制御系を構成することが可能となった。また、RMTP-02D 基板と D-RMTP を用いた単軸試験機でのステップ応答実験によって、制御周期の高速化及び実時間実行機構による低ジッタ化による制御性能の向上を実証した。

専用ハードウェアロジックとセンサデバイス

各種センサデバイスをインターフェース種別に依存しない形でネットワーク内に統合するために FPGA 内のハードウェアロジックレベルで実装されたデバイスドライバを組み込んだ。カスタムセンサデバイスとして小型衝撃検出基板も開発され、力センサと組み合わせて使用することでより高感度な衝突検知を可能としている。これらのセンサデータは低遅延体内通信系によって分散共有メモリを介してシステム内のどのノードからもアクセスすることが可能となっており、専用ハードウェアロジックで処理されるためデータ取得遅延は 100μsec 以内で完了する。モータドライバも同様にハードウェアロジックレベルで電流制御器・

ベクト ル制御器が構成されており、小規模なモータ制御モジュールレ ベルで高精度なブラシレ ス DC モータのサーボ制御を 行っている。

耐ノ イズ信号系

大出力モータド ライバでは、スイッチングに伴う 電磁ノ イズによって電源系統に大きなノ イズが重畳される。それらはグラウンド パターンや電磁輻射によって信号系統に同相ノ イズとして影響を 与えており、さらに分散配置されたノード 間を 接続するケーブルを 介して 全身にノ イズが伝播していた。これらは通信エラーやセンサノ イズといった問題を 引き起こしていたが、通信系の光メディア 化や電源系へのコモンモードフィルタの配置、センサデバイスのアイソレーショ ンを 行うことで対策を 行った。

7.1.2 マルチプロト コル対応分散ノ ード 間低遅延通信系

従来のヒューマノ イドロボット にて用いられてきた既存の通信系では実時間性能や耐環境性能、システム構成の柔軟性、パッケージサイズといったヒューマノ イドロボット 特有の要求項目を 満足する 構成が採れなかったことを 論じた。**第 4 章「 マルチプロト コル対応多ノ ード 間低遅延体内分散通信系の開発」**にてこれらの課題を 解決するための独自の分散システム用通信系を 提案した。また、第 5 章では階層化した 分散型制御システムのサブシステム間通信について、その実時間通信の実現方法についても 論じている。これらの構成方法について以下に要点をまとめる。

同期形態と 接続形態に着目した 通信モデルとプロト コル設計

ロボット の制御システムで用いられる 通信方式について、アクチュエータ等へのコマンド に使用する 通信とセンサからのデータを 収集するための通信では要求される 性能が異なる 点について 言及し、低遅延で1 対1 での同期通信に対応する 通信モデルとして RPC 型通信モデルを 採用することを 述べた。また、非同期通信で多ノード 間での多種データの通信に対応するための通信モデルとしてデータストリーム型通信モデルを 使用することで演算性能に制約のある 組込み制御システムにおいても 効率的に分散型制御システムを構成可能となることを 論じた。

これらの通信モデルを 小規模なデバイスで実現するためにプロトコル処理を 省力化し、代わりに FPGA を用いたハードウェアロジックでの実装が可能な規模とすることで、大幅な低遅延通信処理を 実現することを 提案した。

RPC 型通信モデルは Light RPC プロトコルとして、固定長軽量のパケット でリング型論理ト ポロジ構成を 採用した。これは低遅延性能について Deterministic となるような構成方法であり、他の実時間通信規格でも 用いられる 手法であるが、通信によって呼び出されるプロシージャ 部をプロトコル処理部と 同様に専用の FPGA ハードウェアロジックとして実装したことで、同期処理ながら 優れた低遅延性能を 達成することを 示している。

また、データストリーム通信モデルとして、分散共有メモリプロトコルを 採用した。これは、分散共有メモリをバッファ 長1 の非同期バッファとして利用する 形態であり、アドレスサイズによって扱うことが可能なデータ種別を 管理できる 点を 評価したものである。さらに分散共有メモリはアプリケーションレ ベルでは、パブリッシュサブスクライブ型通信を 行うためのベースプロトコルとして使用することが可能であり、これによって多対多での多種データストリームを 5kHZ 以上の制御周期に間に合う 実時間性能で達成可能であることを 示した。

体内通信系の通信品質向上

通信系の耐ノ イズ性能とノード 間絶縁化のための根本的な解決方法として、物理メディア 層の光通信化が最も 効果的な手法であることについて 述べ、それらを 組み込み系の省資源デバイスでも 活用するための実装として 光アクティブ型コネクタを 採用した。また、光通信化によって最大データレート の大幅な向上も同時に可能となることから、高速ト ランシーバの採用を 合わせて行うことで、体内通信系の通信品質が大

幅に改善されることを示した。通信品質の改善は従来エラー訂正符号処理や冗長符号化によるデータ長の拡大によって生じていた通信遅延をノード間あたり 10 分の1 以下まで低減することを可能とした。

また、MAC 層でにおいては優先度付き並列キューによるデータフロー制御を行うことで、通信プロトコルが要求する実時間性能に応じた遅延性能を提供することが可能となり、小規模な組込みデバイスにて複数の実時間プロトコル通信に対応させることを可能とした。

サブシステム間通信モデル

身体運動の実行系と計画系では、要求される実時間性能と演算性能が異なる点について触れ、これらの差異を解決するための一手法として分散型制御システムが有ることを論じた。分散化における課題として、各サブシステム間で授受する入出力データの通信について、制御周期に応じた実時間性能が要求されることを述べた。また、汎用システムにおいて実時間通信系を組み込む手法はヒューマノイドロボット特有の制約を考慮すると選択肢が限定的であり、目的である継続的な活動を達成するには不十分であったことを示した。

本研究では、汎用システム上のサブシステムとの通信には産業用 Ethernet 規格である EtherCAT を利用することで上位層ハードウェア構成の汎用化を行った。また、EtherCAT と光通信系はブリッジによって接続されることで相互通信が可能であり、インターフェースの汎用化とヒューマノイドロボット特有の制約への適合を両立している。また、制御系データの EtherCAT と光通信系による実時間通信と並列して、ロボット用ミドルウェアを活用した非実時間のオブジェクト指向通信を利用することで、体内制御系へのデータアクセス性や実時間通信系の帯域余裕の確保に効果のある構成とした。

既存のヒューマノイドロボットに組み込む上で、上位層制御システム側のインターフェース処理と新規に採用する通信システムが互換性を有すると開発が省力化できるため都合がよい。中間層制御システムを導入する上で、通信パケットのデータフローが下位層の通信系へとそのままブリッジを介してパスすることで透過性を持たせることが可能となり、上位層側からは統一的に下位層の制御を行うことが可能である。この性質は、制御システムを上位層と中間層でお互いにバックアップする際にも効果を発揮するため、システムの冗長性を高める要素となっている。

7.1.3 即応的反射行動アーキテクチャに基づいた実行系システム

物体との衝突による衝撃の緩和策として、関節のサーボ制御則を生体における関節のように外乱に対して柔軟な動作となるトルク制御則を適用し、即応的な応答となる反射系を構成することが方針の一つとなることを述べた。また、衝撃緩和によって生じる本来の計画軌道からのずれに対して、目的タスクの達成を実現するための適切な修正計画軌道を即座に与えられることで、頑健な身体実行系となることも合わせて述べた。これらを実現するための制御システム構成として即応的反射行動アーキテクチャについて論じ、その基本構成として制御の最短ループ経路と最適ループ経路を組み合わせることを述べた。以下にその構成要素についてまとめる。

実時間性能に応じた実行系の階層化、制御フローの多重化

フィードバックループの遅延時間の最短化を目指す即応制御システムと、最適な動作軌道生成を行う身体動作計画系について、要求される実時間性能と演算性能が異なるため、分散型制御システムとして階層化したサブシステムとしてそれぞれ構成することを述べた。上位制御システムでは演算性能を活かして、現在の状態から最適身体動作軌道をオンラインで計算することで、ロボットのハードウェア本体が本来の計画軌道と異なる動作が実現されてしまった場合にも対応する。一方、中間層制御システムとして構成される即応制御システムでは、低遅延通信系を活用して関節トルク制御則を高精度で実現するのに必要となる 5kHz 以上の制御周期での制御指令とセンサデータの通信を実現することで最短でのフィードバックを達成している。上位層で計算された最適身体動作軌道は、中間層制御システムにおいて計算トルク法による動力学演算によって、目標関節トルク指令値へと変換することによって最適ループ経路と最短ループ経路の制御フローを統合することを可能としている。

動力学を考慮した関節トルク制御則の適用

ベースとなるサーボ制御則をトルク制御化することで、外乱に対して物理的な作用によって高い応答性で緩和的動作が実現される。関節が有する機械的な弾性は、本研究で使用するロボットにおいてはほぼ無視できる程度であるため、制御則による弾性特性を模擬する必要がある。また、減速機の粘性摩擦の影響やロータ慣性の存在によって実関節トルクとアクチュエータの出力トルクの間に非線形な関係が生じてしまう。これらのアクチュエータの動特性を数理モデルで表現し、ロボットの身体本体の慣性力等の動力学モデルと統合して必要なモータ出力トルクを計算することで、目標軌道への追従を実現する。また、各モータ制御モジュールにハードウェアロジックとして実装された関節トルク推定器が出力する推定値を利用したフィードバックループを構成することで、2 自由度制御系としてサーボ演算を構成した。

オンライン動作生成器による動作軌道再生成

上位層制御システムにて実行される身体運動の計画系において、ロボットが外乱やトルク制御則の適用によって生じる本来の計画軌道からのずれにたいして修正軌道を即座に計算し直すことが要求される。このオンライン動作生成器では、下位制御層から上がってくる身体知覚情報であるセンサデータ、ロボット状態情報について、最適化演算によって次の制御フレームでの目標動作軌道を与える。本研究では転倒を防ぐためのオンライン歩行動作生成器をシステムに組込み、各サンプリングフレームについて、最適な歩行軌道を逐次的に生成することで転倒を防止する。この歩行動作生成器は LIPM の最適レギュレータについての陽解を用いることで、数歩分の制御サイクル長について実時間で演算することが可能とするアルゴリズムを採用した。以上の構成によって脚が外乱について緩和動作を行いながらも、高速に転倒を回避する歩行動作を継続することで、実環境下のような未知物体の上を踏破する状況においても転倒を回避することが可能となっている。

7.1.4 本研究の意義

本研究は既存のインターフェース規格によってその構成方法に制約が生じていた従前の制御システムについて、ヒューマノイドロボットにおけるトータルパッケージを考慮した設計要件を定義し、システムのハードウェア構成・ソフトウェア構成の両側面から論じたものである。光通信化による高い通信品質とマルチプロトコル対応な低遅延通信機構を有する独自のモータ制御モジュールを開発することで、この設計要件に沿った階層化された分散型制御システムのデザインを実現することが可能となり、従来のシステム構成が構造的に抱える信頼性に関連した致命的な問題について改善できることを示した。また、即応的反射行動アーキテクチャに基づいたトルク制御ベースによる高速応答行動を可能とすることで衝撃外乱に対して物理的な作用による柔らかい緩和動作を実現しつつ、オンライン動作計画器によってタスクの継続も合わせて継続させることを提案した。これらは、既存のヒューマノイドロボットの制御システムと置換する形で設計・実装を行うことで、本研究で提案したシステム構成の有用性について実証を行っている。

新規に開発を行った制御システムであるが、過去に蓄積されてきたシステムの資産については互換性を維持する形で接続や内包することが可能であるため有用性が非常に高いと言える。実環境下での運用を意識した構成であるため、従来は限定的であった屋外でのヒューマノイドロボットの実験・活用をさらに促進させることができると考えており、ヒューマノイドロボット応用において波及効果は高いと言える。

7.2 今後の展開

本研究で述べた分散型制御システムの構成方法によって、予測されていない衝突や転倒の危険のある実環境下においても頑健で継続的に作業を続けられるヒューマノイドロボットの実現に向かえると考える。特に、これまでは必要な制御システムのハードウェアを体内に全て搭載するために大型化していたヒューマノイドロボットの身体について、大幅に小型化を実現することが期待される。

しかし、現実に実環境下のあらゆる事象について作業を継続できる頑健性を得るには、まだ多くの課題と改善点を残していると考える。まず、本研究で対象とした身体運動計画処理は歩行動作のみとなっているが、実際には必要となるオンライン動作生成器はこの一つには収まらない。腕を持つ場合にはマニピュ

レーションの計画系を持つことになり、これも同様のオンライン動作計画処理について実現する必要がある。特に、本研究の歩行動作計画器は転倒の回避を最優先とした最適化を行うため、場合によってはマニピュレーションタスクの失敗につながる歩行動作となる可能性がある。マニピュレーションタスクを継続しつつ、転倒を回避するために、例えば余剰マニピュレータによって手すり等につかまれる場合にはつかまるといった、多点接触問題のオンラインでの最適化を実現するといった方法が考えられている。また、歩行動作計画を LIPM ではなく、角運動量やフットトルクの存在をモデルに落とし込んで最適化することで、踏み出しを行うことなく転倒を堪えるという、より人間らしい転倒回避方法をとることも同様に考えられている。このように転倒回避だけでも、採り得る行動パターンは複数存在し、それらについて最適行動指針の評価、最適軌道の計算を実時間で実行するためには、多くの獲得すべき技術的蓄積が残っていることが明らかである。また、身体運動計画系は下位層から得られる知覚情報のみによってその動作が決定されていたが、実際にはさらに上位に位置する認識系の結果を同様に知覚情報として最適計算の入力に組み込む必要がある。些細な課題であるが、生成された関節角度軌道から関節トルク参照値の計算に必要となる関節角加速度のロバストな演算手法についても必要とされる所である。単純な差分計算手法では非連続で外れ値も生じ得ることは一般に良く知られており、本研究でも付録にてまとめているロバストな速度推定手法等も適用して加速度演算を行ったがまだ完全な値を得るに至っていない。ローパスフィルタのような遅延の生じる手法も適用は不可であり、本システムに適した速度推定手法の研究も必要であると考える。

また、本研究で実証した最大の制御システム構成は脚型に留まるが、実際には上半身の制御系を含めたシステムに拡張する必要がある。本論で述べたように、本システムは拡張性に優れた構成であり、単純にノード数を拡大するのみであれば問題が無いと考える。もしノード数の増大によって通信系の遅延性能の悪化が見られる場合には、通信系自体を複数のサブネットで構成できるように拡張する必要が生じると考えている。例えば、上半身と下半身で同一のネットワークとして扱う要求は低いと考えることができるため、その間にネットワークブリッジを設けることでサブネットに分割し、ブリッジパケット以外はサブネット内で完結させることで、データフローを抑制し、通信系の遅延の増大を阻止することは可能であると考える。

通信系に関しては冗長化による信頼性の拡大も課題であると言える。光通信化によって、電磁ノイズに対して大幅に耐性が向上したが、ケーブルの物理的な寸断等で通信が不可能となるリスクは依然として存在する。根本的な解決方法としては通信経路の冗長化であり、経路が多重化されることで上記リスクは低減される。冗長化によるネットワーク構成方法の複雑化については、1）航空機等で見られる独立した並行通信経路による多重化、2 ）*Responsive Link* で見られる冗長テーブルによるパケットルーティング機構のネットワーク層への適用、等が考えられている。これらは、開発コストや通信系の構成に要求されるハードウェア資源の拡大につながるため、研究レベルでの適用にはまだ日数がかかると思われるが、研究レベルでは現状の信頼性で十分実環境下での運用は可能であるため、現状においては解決優先度は低いと考える。同様に物理層における FEC の適用も研究レベルにおいては必要性が低いが、将来的に FPGA 内に構成可能なゲート数規模のコストパフォーマンスが高まった場合には適用対象となる。

また、本研究では動力学を考慮したトルク制御系を構成したが、必要となるパラメータは公称値と一部計測実値を用いたため、実際の制御性能は本来期待される数値以下となっている。このパラメータが真値に近づくことでより柔らかい制御が可能になると考えるのは自然であり、このための推定手法については古くから数多くの研究が成されている。特にオンラインでのパラメータ同定手法を適用することで、時々刻々変化する関節の粘性摩擦係数や足裏の摩擦係数等をより正確に制御演算に組み込むことが可能となり、目標とする継続的な作業の実現により近づくと考える。また、本論で採り上げた衝突検知アルゴリズムについてもより精度が高まると考えられるため、力センサや衝撃センサと統合することで正確な衝突検知機構が達成される。これらのオンラインでの同定結果は通信系を介してログデータとして収集することが可能であり、近年活発に応用が進んでいるディープラーニング等の学習手法との統合にも適用が期待される。また、本研究にて使用したアクチュエータモデルは軸弾性を考慮していないことによる誤差が存在する。実装においては軸を完全な剛体と仮定しているが、現実には特に減速機に弾性が存在することで、モータ軸出力と関節軸出力角度との間に軸トルクに応じた誤差が生じているとモデル化するのが正確である。このモデルを制御システムに組み込むには、関節出力軸角度を測定するためのエンコーダ等を搭載することが必要となると考える。次期ロボットの開発においては、関節出力軸角度の測定用の精度を持ったエンコーダが搭載され始めているため、それを利用した制御系へと発展させる予定である。

最後に、本研究では対象外であったがロボットの身体機構本体についても、実環境適用において改善点が多い領域であると考える。従来のヒューマノイドロボットの構成では、転倒しない・物体と衝突しないことを前提とした作りであり、制御システムのマウント方法や保護方法は簡易な構成となっている場合も見

られた。しかし、本質的にヒューマノイドロボットでは転倒や衝突は避けられないものであり、これらを前提とした制御システムの保護方法を機構として採用するのが望ましい。また、本研究で提案した即応的反射行動アーキテクチャでは、脚や腕は外乱に対してなじむ動作が基本となった。これは人間を含む生物についても同様と考えられる。しかし、現状のロボットのように外装が固い素材であったり、構成部品がむき出しの状態ではエンドエフェクタ以外の部分でのなじみ動作は、自機部品や接触対象物に傷や破損を与えてしまう結果となることが予想される。そのため、全身を柔らかい外装素材で覆うことが重要である。本研究で提案した制御システムでは、システムを構成するデバイス類のパッケージサイズを小型化する効果があるため、それに合わせてロボットの主要骨格も細形化することで、人間に近いプロポーションで柔らかいロボットを構成することを期待している。

付 録 A

ロボット諸元

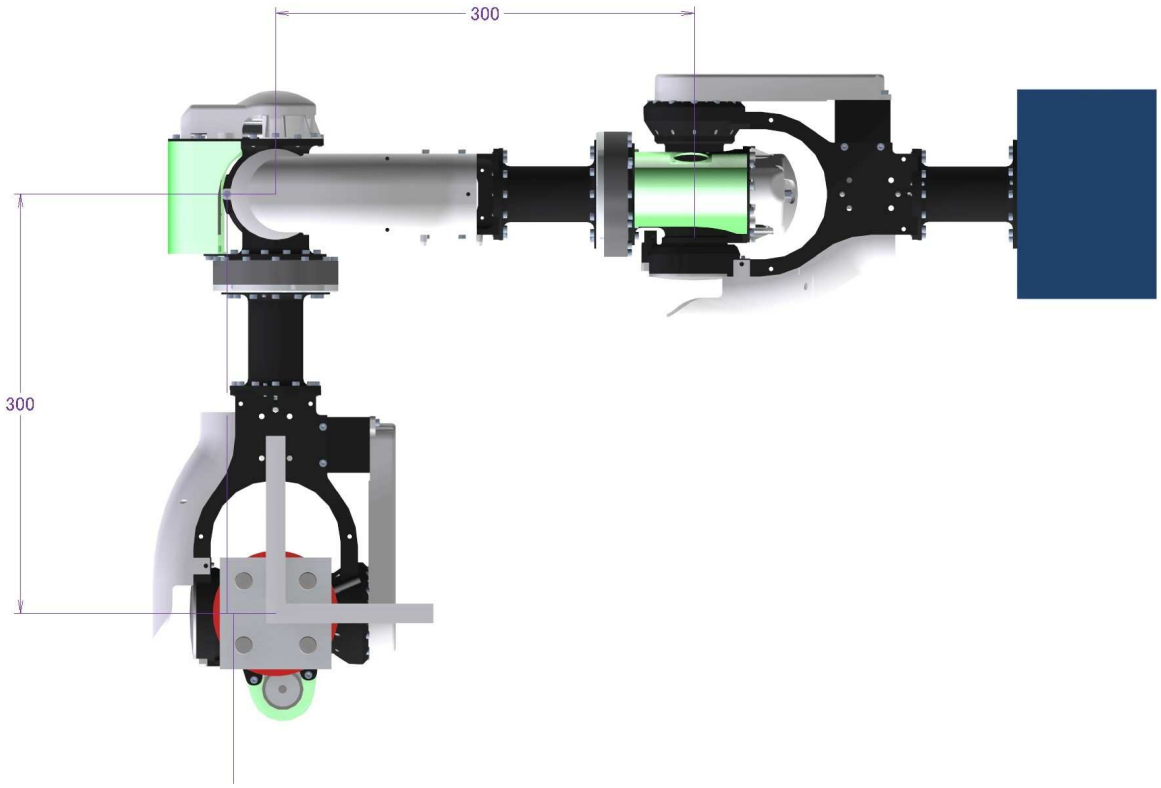


図 A.1: Front View of A0-B spec CAD Model.

A.1 A0-B spec

大出力モータドライバを用いた6 自由度ロボットアームの構成。

A0-B spec は浦田らが開発した大出力脚型ロボット HRP3L-JSK[50] の技術をベースに、等身大型ヒューマノイドロボットとして開発された STARO[96] の腕部分のみを使用したロボットである。主なスペックは表 A.1 に示す。オリジナルでは8 自由度構成であったが、本研究においては演算の簡略化のために6 自由度構成として使用している。また、図 A.1 、図 A.2 と図 A.3 に CAD モデルによる外観を示す。分散制御基板は全て、腕外に搭載されており、モータとの間の必要なケーブルは全て、骨格部品の間を通して配線されている。本研究においては A0-B spec のベース部分は固定されている。

表 A.1: Specification of A0-B spec

Link length	300mm
# of joint	6
Gear Ratio	460.8:1 (160:1 by harmonic drive and 72:25 by pulley)
Cooling	Natural air cooling
Sensor	Six axis force sensors (on end-effector) Shock detect sensor 16G and 200G
Rated Voltage	80V (max)

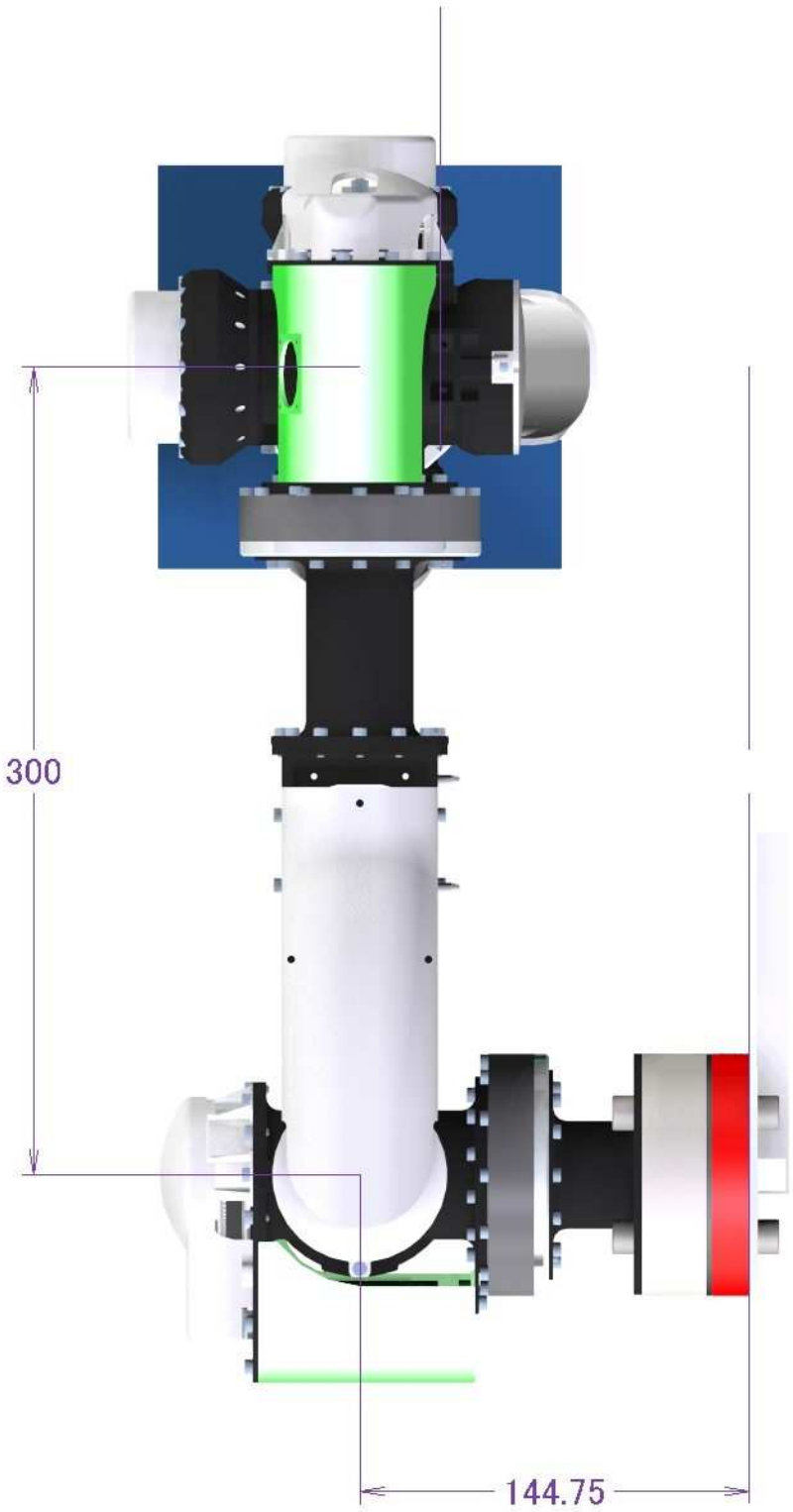


図 A.2: Side View of A0-B spec CAD Model.

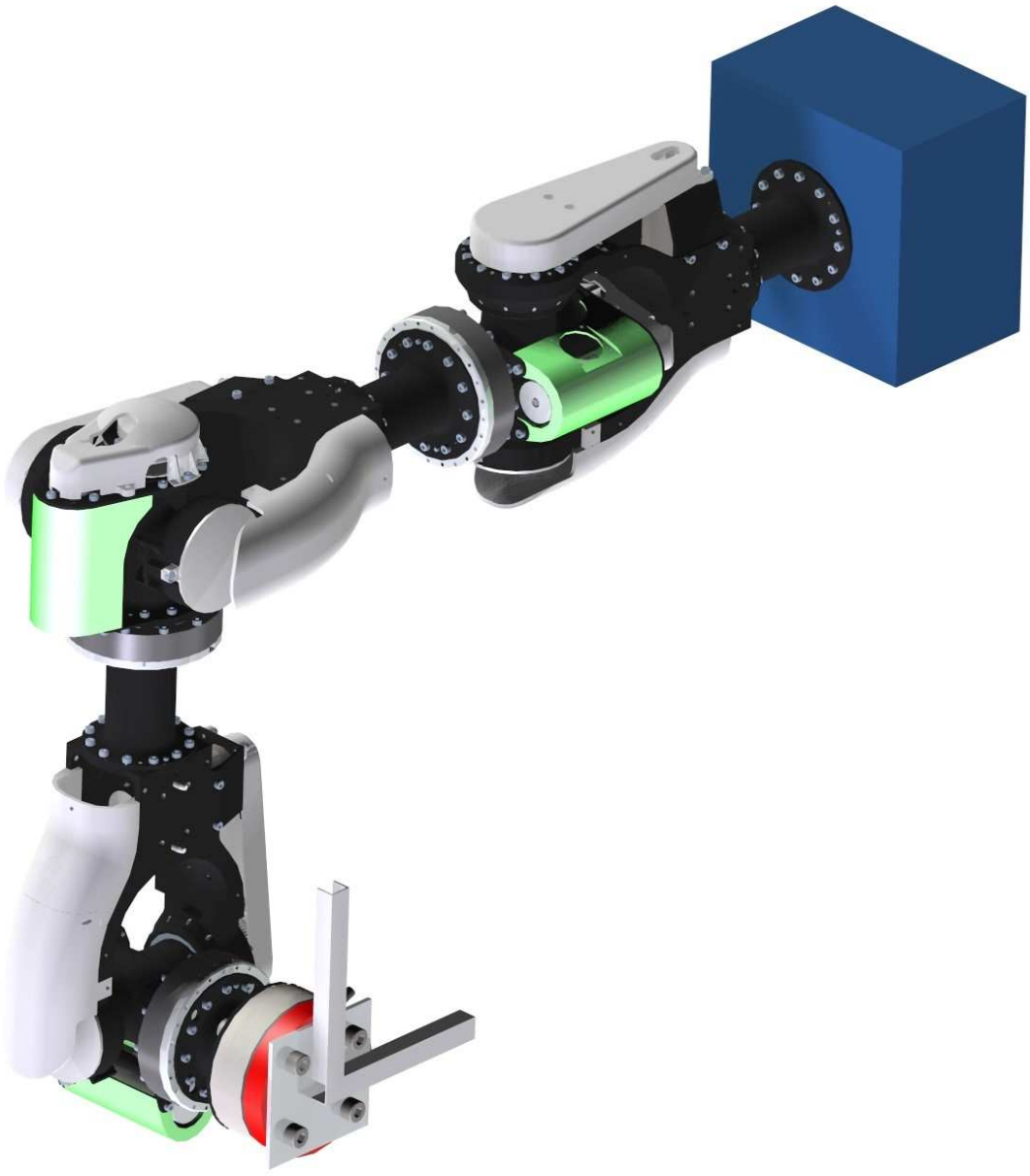


図 A.3: Isometric View of A0-B spec CAD Model.

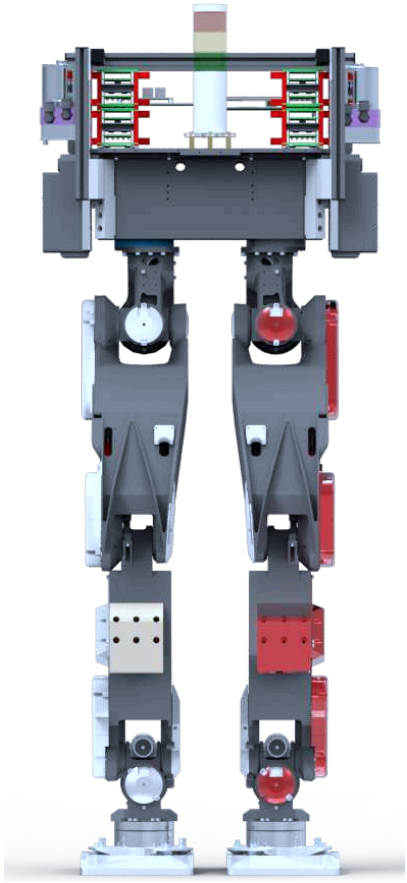


図 A.4: Front View of L1 CAD Model.

A.2 L1

L1 は浦田らが開発した大出力脚型ロボット HRP3L-JSK[50] の技術をベースに、等身大型ヒューマノイドロボットとして開発された STARO[96] の脚部分のみを使用したロボットである。主なスペックは表 A.2 に示すように、基本的な関節構成は HRP3L-JSK と変わらないが、足首2 自由度関節の内、ロール軸関節がピッチ軸関節と直交しておらず、下方へオフセットしている点が異なる。また、膝関節と股関節のピッチ軸についてはダブルモータ構成となっており、出力限界が向上している。分散基板の内、膝から足首の関節を駆動する基板やケーブルの大部分については、リンク構造内に収められており、転倒や物体との接触によって破損する危険性を低下している。

図 A.4 と図 A.5 に CAD モデルによる外観を示す。

表 A.2: Specification of L1

Weight	44[kg]
# of joint	12 (6 for each leg)
Gear Ratio	240:1 (160:1 by harmonic drive and 60:40 by pulley)
Water-Cooling	Electrical pump and radiator
Sensor	six axis force sensors (on each foot) Shock detect sensor 16G and 200G Inertial Measurement Unit (IMU)
Rated Voltage	80V (max)

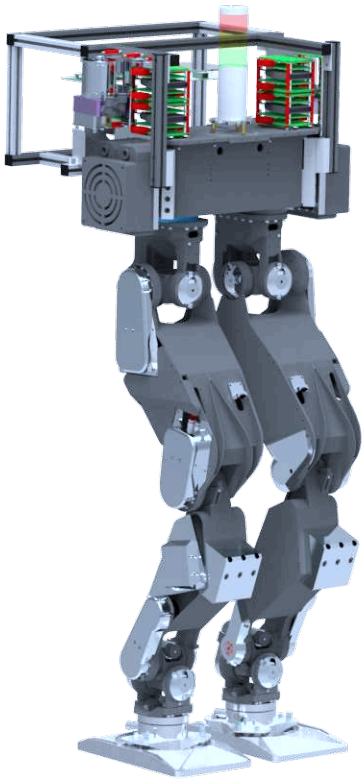


図 A.5: Isometric View of L1 CAD Model.

付 録B

ロボット 体内ノイズと通信系への影響

B.1 ロボット 体内のノイズと 通信システムへの影響

図 B.2 は大出力脚 (図 B.1) の膝駆動モータドライバ付近にてモータのサーボコントロール開始前と開始後のロボット 周辺の空間中磁力線強度をスペクトルアナライザ (図 B.3) によって計測したものである。図に示されている通り、サーボコントロール開始によって空間中の磁界強度は 10MHz 80MHz 帯で約 30dBw 上昇している。

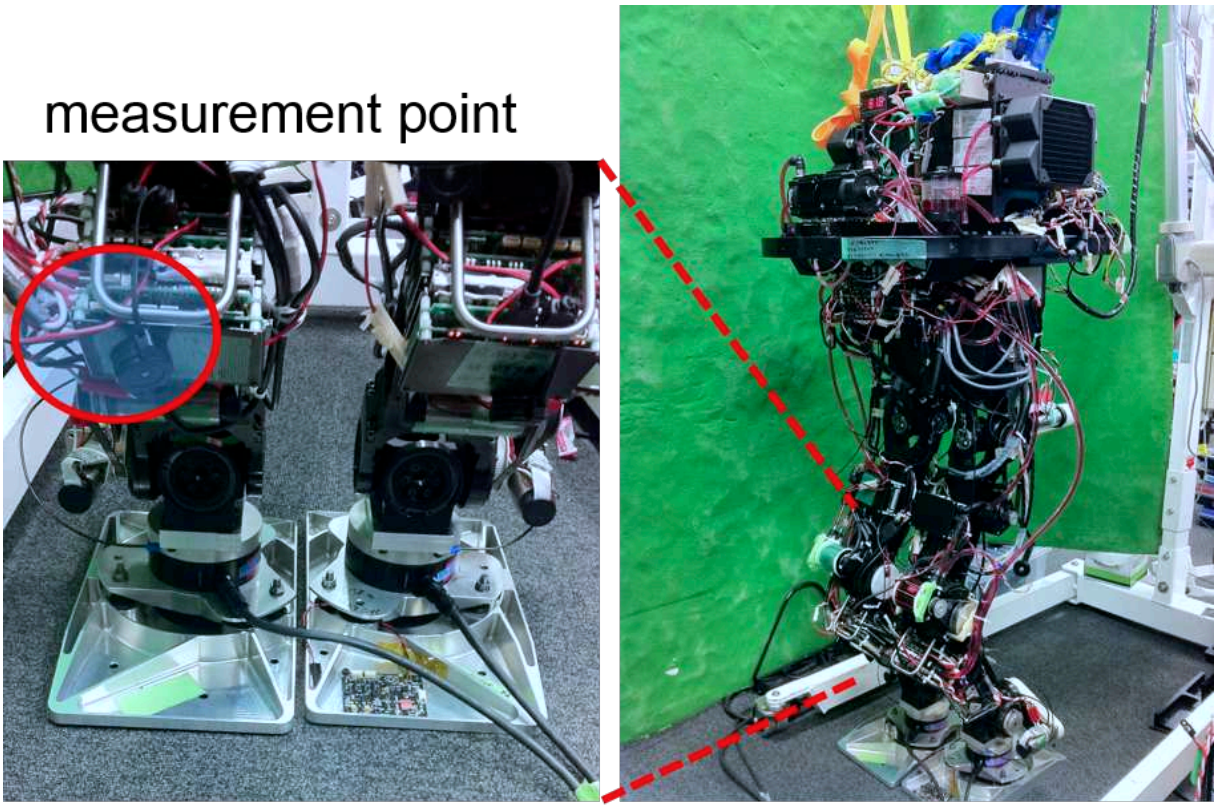


図 B.1: Magnetic field measurement point on HRP3L-JSK.

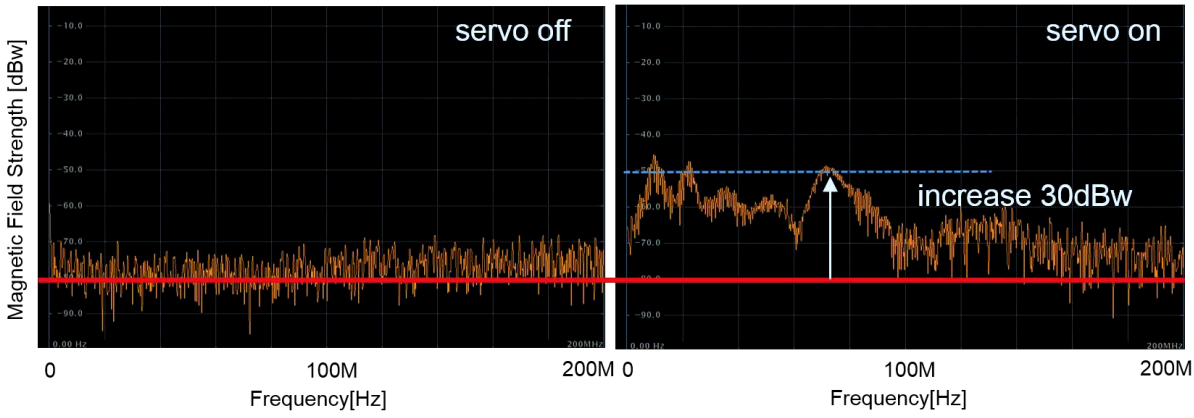


図 B.2: Magnetic field strength before and after servo on.

図 B.3: Magnetic field probe and spectrum analyzer.

B.1.1 同相ノイズの差動信号への影響

電線1 本の電圧の high,low によって信号伝送するシングルエンド 信号は低速なシリアル通信系で使用されているが、基準となるグラウンド のゆらぎや信号性にのるノイズによって high-low の判定にエラーが生じてしまう。そのため、信頼性を 求める通信や高速なシリアル通信等では電背2 本の電圧差によって信号を伝送する 差動信号による 伝送が用いられる。差動信号化によって同相ノイズは打ち消されるため、品質の高い信号を 受信側は得ることが可能となる。しかし、実際には同相ノイズは受信側のグラウンド への影響や受信器入力端子の同相電圧レンジが存在するため、同相ノイズ対策が不完全では結局問題を 生じることとなる。図 B.4 は、LVDS による2 基板間通信を 行っている 際に 大出力ロボット のモータノイズが通信線に重畳している 様子を オシロスコープで 計測したものである。図 B.4 の下側紫色の波形は差動プローブによって 計測した 差動信号であり、ノイズの有無に関係なくきれいな 波形を 保ち、ノイズに 対する 耐性の高さを 示している。一方、同じ 信号をシングルエンド プローブで 測定した 様子が同図の 黄色と 水色の波形であるが、こちらはモータノイズによって 激しく 揺さぶられており、ピーク 値でグラウンド 電圧基準で 0V を 下回る 点も 確認される。このような 環境下では 差動信号としての 波形は 健全性を 保っていても 通信はエラーが頻出し、通信を 行うことができなかった。従って、ノイズに 強い差動信号による 信号伝送規格を 採用していたとしても 同相ノイズ対策は 不可欠であり、十分に 同相ノイズを 減衰できない 場合には 通信不良を 引き起こすことが確認された。同相ノイズ対策としては 次節に 挙げるフェライト コアによるものが一般的である。あるいは、光伝送のような 原理的に 電磁ノイズの 影響を 受けない 伝送媒体を 用いることや、電気的耐性の高い規格を 採用することが考えられる。LVDS と 比較して、RS422 は電気信号としてはより 大きな 電圧範囲を 扱うため、同じ 同相ノイズに 対して 高い信頼性を 得られるが、その分スイッチングに 伴う スキューによって 通信速度の高速化は 難しくなる。

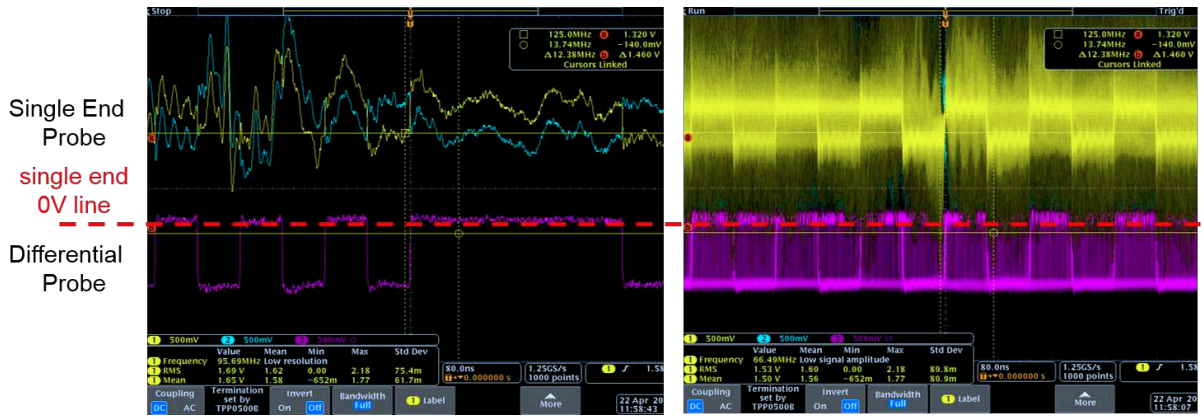


図 B.4: Common mode noise wave measured on CAT7 cable.

B.1.2 フェライトコア・フェライトシースケーブルによる同相ノイズ除去

一般的に電気的ノイズへの対策で採られる対策としてシールドケーブルの使用、適切なアーシング等が考えられる。しかし、接地線への接続や体内で安定したグラウンドを取ることが困難な移動ロボットでは、ノイズ対策としてシールド線を使用しても十分な効果を出せない場合がある。従って、ロボットのモータドライバから発生するノイズの対策としてしばしばフェライトコアによるノイズ自体の吸収が用いられる (図 B.5)。フェライトコアはケーブルを包み込む形で装着され、電気的にはインダクタンスに近い特性を持ち、広い周波数帯で高いインピーダンス周波数特性を示す (図 B.6)。 特に差動信号を扱うケーブルにフェライ

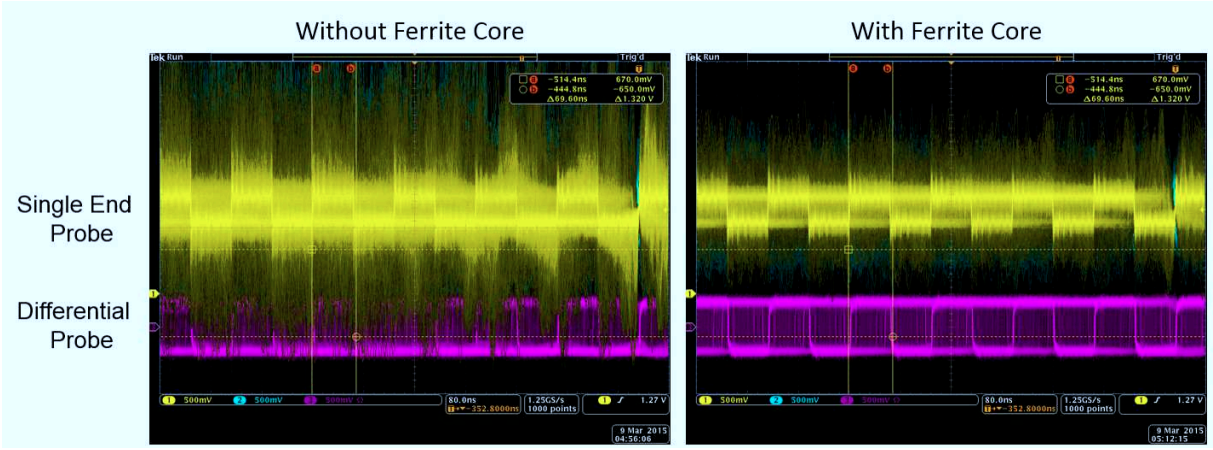


図 B.5: Common mode noise wave measured on CAT7 cable.

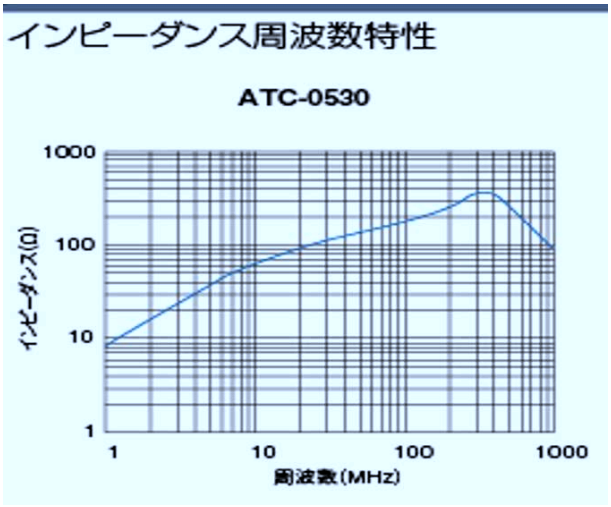


図 B.6: Impedance frequency characteristics of ferrite core (ATC-5030) used for HRP3L-JSK. This graph is from a datasheet of manufacturer[97].

トコアを使用した場合、差動信号成分の減衰はほとんど無く、同相信号成分に対してのみ減衰効果が表れるため、同相ノイズのみを効果的に減衰することが可能である。また、フェライトコアは後付けで組み込むことが容易なため、ノイズの状況に応じてフェライトコアを増減させることで効果を調整することが可能である。ただし、

- 部分的にだがケーブル径の数倍の部品がつくことによる空間的制約
- 密度が高く重い部品のため、全身では数百グラムから数キログラムの重量増になってしまう。
- ケーブルの自重よりも重いため、適切に取り付けられていない場合ロボットの振動や加速度が発生した際にコネクタやケーブルに負荷をかけてしまう。

といった問題点もあるため、濫用はロボットの運動性能に影響を与える点に留意しなければならない。

また、基板側に実装可能なフェライト素子や同相ノイズ除去用チョークコイルもあるため、基板開発可能な場合や実装面積に余裕がある場合には基板側で対策を行うことが可能である。こちらの場合は、後付けによるノイズ減衰効果の増減を行うことは難しくなるが、重量の問題を小さくでき、高いノイズ減衰効果も得ることが可能である。ただし、直接電流が素子に流れるため最大定格電流を考慮する必要は出てくるため、パワー系統のケーブルでは考慮しなければならない。

また、フェライトコアに近いノイズ対策手法としてフェライトシースを使用したケーブル製品を使用するという方法がある。フェライトシースは通常ケーブルのシースとして使用される PVC 等の材料にフェライト粉末を練りこんだもので、フェライトコアと同様の効果を示す。フェライトコアを使用する場合と比較して、

- ケーブル全長に渡ってフェライトシースに包まれる。
- ケーブル径は通常のケーブルと同等に収まる。
- ケーブル重量も通常のケーブルと同等に収まる。

といった利点が挙げられるが、

- フェライトコアと比較して詳細なインピーダンス特性等の技術情報が乏しい。
- ロボットのノイズ対策として重要な 100MHz 以下の帯域についてはフェライトコアと比較してインピーダンス特性が弱い。

といった課題も見られる。実際にフェライトコアとフェライトシースケーブルで同相ノイズによる通信エラーの有無及び、低周波帯域におけるインピーダンスを計測した結果を図 B.7 、表 B.1 、図 B.8 に示す。図 B.4 と比較して、多少ノイズレベルは下がっているが、通信エラーは発生したため、十分なノイズ吸収効果には至っていない。実際、フェライトコアを使った図 B.5 と比較して、明らかにノイズの低減能力は不足していると言える。

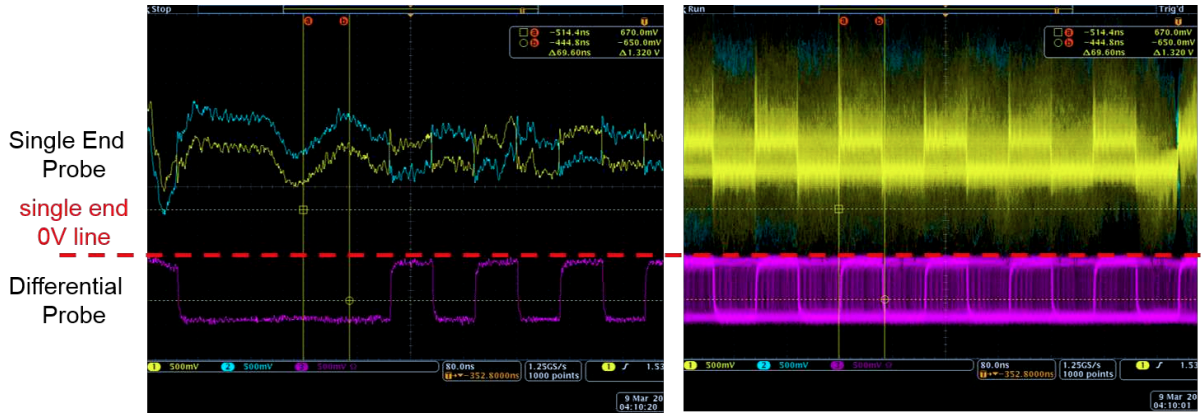


図 B.7: Common mode noise wave measured on ferrite sheath cable.

表 B.1: ケーブルインピーダンス測定

ケーブル	Frequency[Hz]	Z[Ω]	C[F]	L[H]	θ[deg]
フェライト シースケーブル (フェライト コア1 個有)	5.0M	40.9	762p	**1.33u**	79.5
	500k	4.09	77.4n	**1.31u**	83.7
	5.0k	67.3m	407u	2.48u	59.2
フェライト シースケーブル (フェライト コア無し)	5.0M	10.1	3.17n	**321n**	89.7
	500k	906m	351n	**289n**	87.6
	5.0k	30.6m	306u	3.32u	17.0
撚り 線ケーブル (フェライト コア1 個有)	5.0M	38.9	799p	**1.27u**	77.2
	500k	3.60	87.7n	**1.15u**	82.9
	5.0k	97.3m	188u	5.2u	36.0
撚り 線ケーブル (フェライト コア無し)	5.0M	3.82	8.33n	**122n**	88.7
	500k	343m	890n	**114n**	73.6
	5.0k	98.4	10.1u	98.6u	1.82

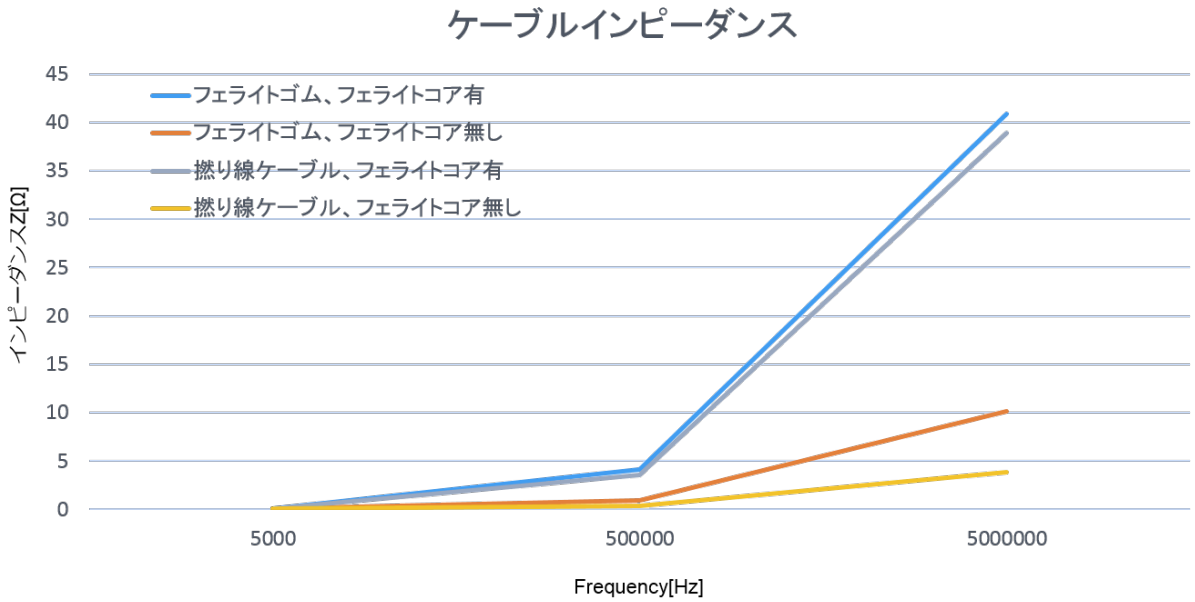


図 B.8: Impedance of ferrite-sheath cable and twisted wire cable.

付 録C

高速動作のためのセンサ信号処理

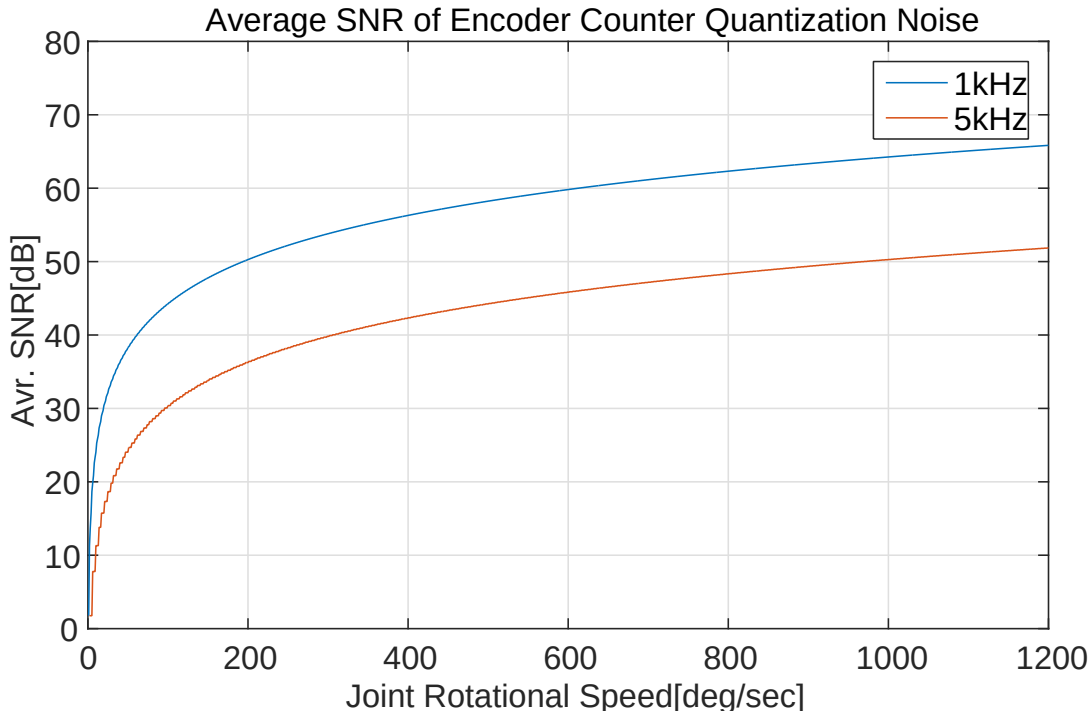


図 C.1: Effect of quantization noise on a calculated joint rotational speed.

C.1 高速関節動作のための関節速度推定器

従来のシステムでは各関節角速度 ω_k は計測されたモータエンコーダパルス数の周期 ΔT ごとの後退差分から推定値の計算を行っている (式 C.1)。

$$\omega_k = \frac{\theta_k - \theta_{k-1}}{\Delta T} \tag{C.1}$$

特に FPGA 内のパルスカウンタを利用することで非常に低ジッタな ΔT を得られるため、高精度での計測が可能となっている。しかし、極低速域での運動を行う場合や制御周期の高速化を行う場合、エンコーダの量子化ノイズによる速度計算の誤差が大きくなってしまう問題が生じていた。関節角度制御を行う際には、この速度誤差は問題にはならないが、関節トルク推定等を行う場合には推定角速度の計測品質も重要となってくるため、量子化ノイズ対策が必要となる。

N[bit] フルスケールの ADC について、平均量子化ノイズと信号の比率 (Signal to Noise Ratio, SNR) は一般に以下のように表される ([98][99])。

$$\mathrm{SNR} = 6.02N + 1.76\mathrm{dB} \tag{C.2}$$

1kHz 制御周期と 5kHz 制御周期で 5 倍の周期差がある場合、同一の関節角速度を計測する場合 5:1 のエンコーダパルスカウント 差が生じる。上記式より 制御周期差による SNR の差は $6.02(\log_2 5 - 1) = 7.96\mathrm{dB}$ となる。また、極低速域で計測されるエンコーダパルスカウント 差が 1bit レンジの場合、$\mathrm{SNR} = 6.02 + 1.76 = 7.78\mathrm{dB}$ となり 非常にノイズの大きい値となる。図 C.1 に関節角速度とエンコーダカウンタパルスの量子化ノイズによる平均 SNR の理論値を 制御周期 1kHz と 5kHz の場合についてそれぞれ示す。制御周期の高速化によって、推定関節角速度に与える 量子化誤差の影響が 10dB 以上あることが確認される。また、100[deg/sec] 以下の低速度領域では特に量子化誤差の影響が顕著に表れることが見て取れる。

エンコーダパルスのカウント 値から角速度を 推定する手法は古くから 提案がなされており Benkhoris ら [100] や Bartolini ら [101] 等数多く 存在する。本研究では FPGA への実装容易性も 考慮してエンコーダからの角速度推定手法を 選定していく。

最も古典的な方法としては、ウィンドウサイズを N として計測値をバッファに格納しておき、差分計算にウィンドウ幅を持たせる方法である (式 C.3)。

$$\omega_k = \frac{\theta_k - \theta_{k-N}}{N\Delta T} \tag{C.3}$$

また、二次項までのテーラー展開近似による方法は式 C.4 で与えられる。

$$\omega_k = \frac{2}{3N\Delta T}(\theta_k - \theta_{k-N}) - \frac{1}{3}\omega_{k-N} \tag{C.4}$$

また、近傍3 点の最小二乗近似による計算方法が式 C.5 で与えられる。

$$\omega_k = \frac{3\theta_k - 4\theta_{k-N} + \theta_{k-2N}}{2N\Delta T} \tag{C.5}$$

また、テーラー展開近似による手法を近傍4 点にまで拡張した Brown らによる方法は式 C.6 で与えられる。

$$\omega_k = \frac{13\theta_k - 19\theta_{k-N} + 7\theta_{k-2N} - \theta_{k-3N}}{8N\Delta T} \tag{C.6}$$

また、Bartolini らが提案した Sliding Mode Control に基づいた角速度推定手法 [101] によって、関節トルク推定に利用可能であると提案されている。

以上の5 手法について実際に実装を行い、実機の関節角度軌道を入力することで評価を行う。なお、Sliding Mode Control による手法についてはパラメータゲインを 0.01 と 0.04 の2 パターンについて検証を行っている。図 C.2 はテスト用に計測した、実機での歩行時の膝関節角度のプロットである。5kHz の制御周期で実行されるプロセスでサンプリングされている。図 C.3 に各関節速度推定器で計算された関節速度軌道と、その関節速度軌道を再度積分して得られた推定関節角度軌道を示す。この時、ウィンドウサイズは1 (つまり、ウィンドウ無し) と設定している。図 C.5 は実関節軌道と推定関節角度軌道の誤差を示す。図 C.4 及び図 C.6 はウィンドウサイズを 5 とした場合のプロットである。

ウィンドウサイズ1 とウィンドウサイズ5 を比較した場合、明らかにウィンドウサイズ5 とした場合の方が推定関節角速度に重畳しているノイズレベルが低いことが分かる。また、5 つの手法の中で最小二乗近似による手法と Brown の手法は実関節軌道と推定関節角度軌道の誤差が小さく、良く真値に近い推定値を計算していると考えられる。Sliding Mode Control に基づいた手法はゲインによって適切な角速度レンジが異なると考えられ、広範な角速度レンジを実現可能な大出力ロボットシステムではゲインチューニングが必要となることから他の手法と比して不利と考えられる。FPGA で実装するにはコストが高い点も、本システムで採用するには不利な要素となる。最小二乗近似の方法と Brown の方法を比較した場合、乗算器の数および必要バッファサイズは最小二乗近似による方法が低コストで実現可能であるため、本システムでは最小二乗近似による関節角速度推定を実装として採用した。

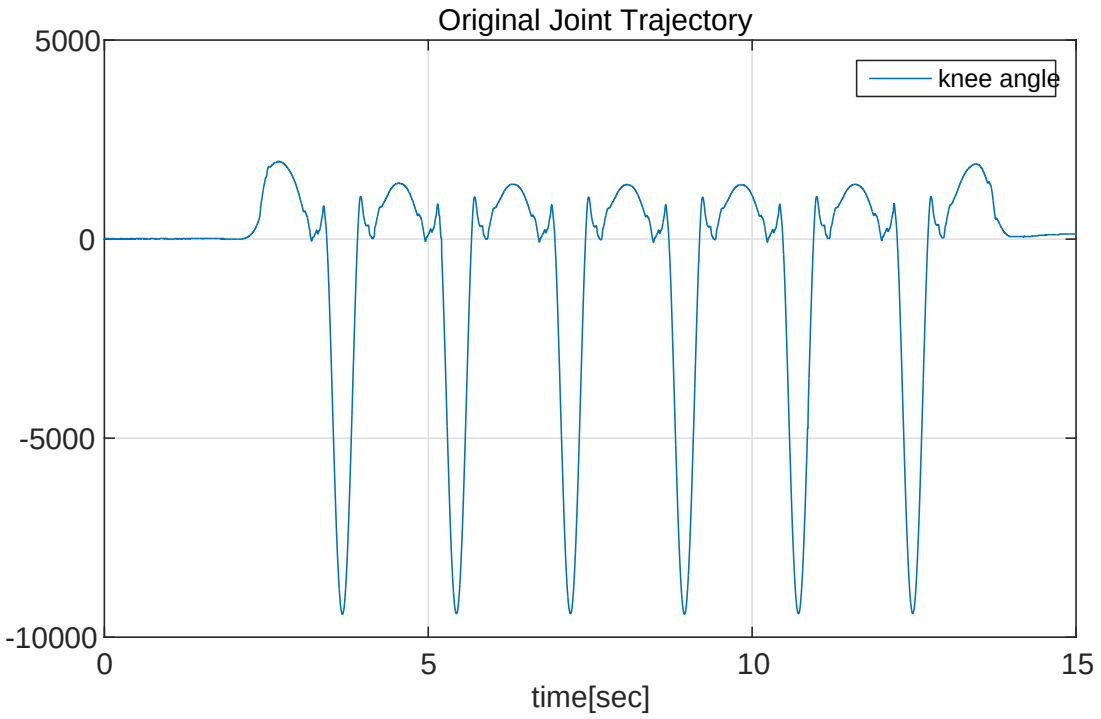


図 C.2: The original knee joint angle trajectory, sampled at 5kHz, while walking to be estimated.

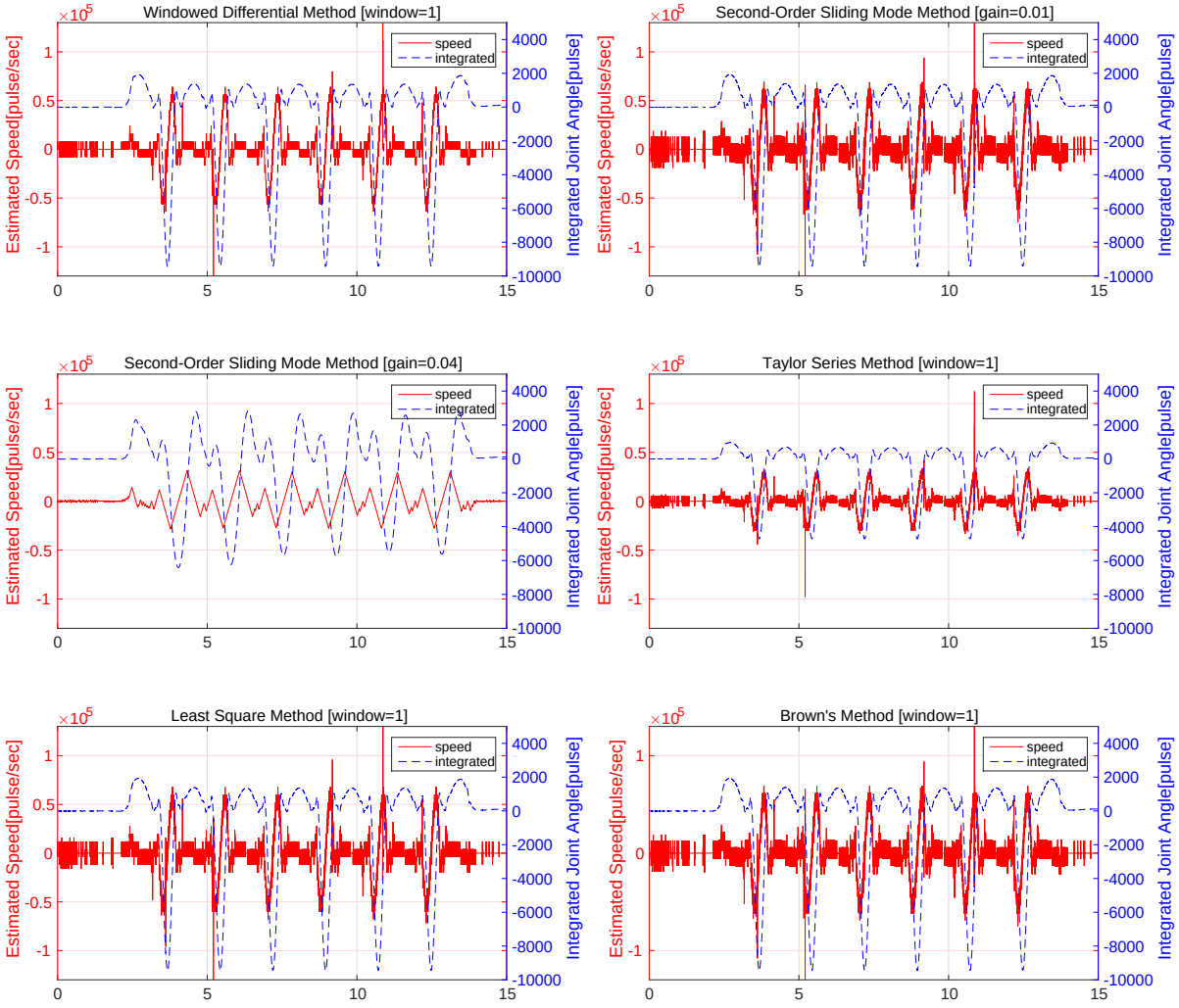


図 C.3: Comparison of five speed estimating methods.

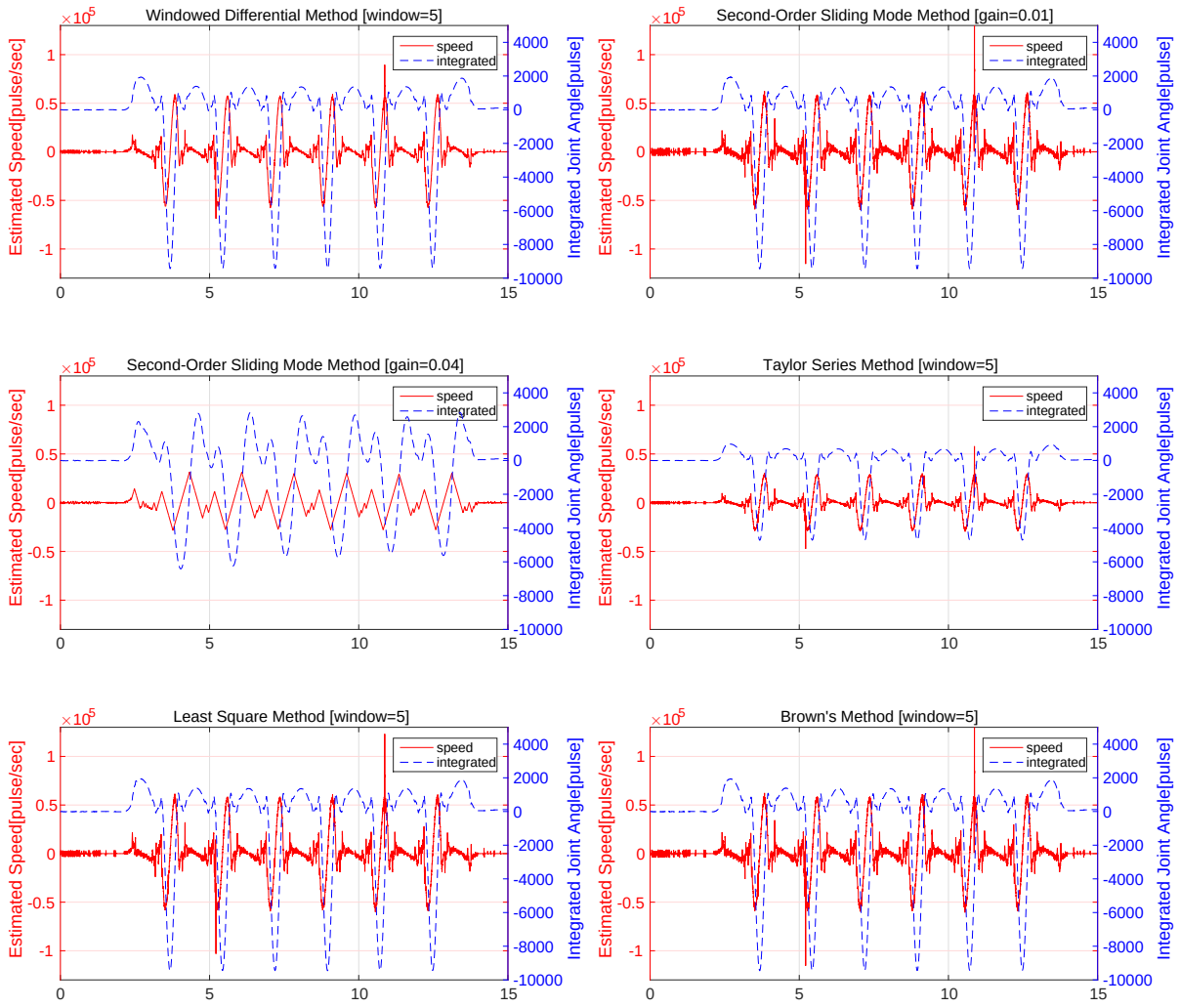


図 C.4: Comparison of five speed estimating methods with window=5.

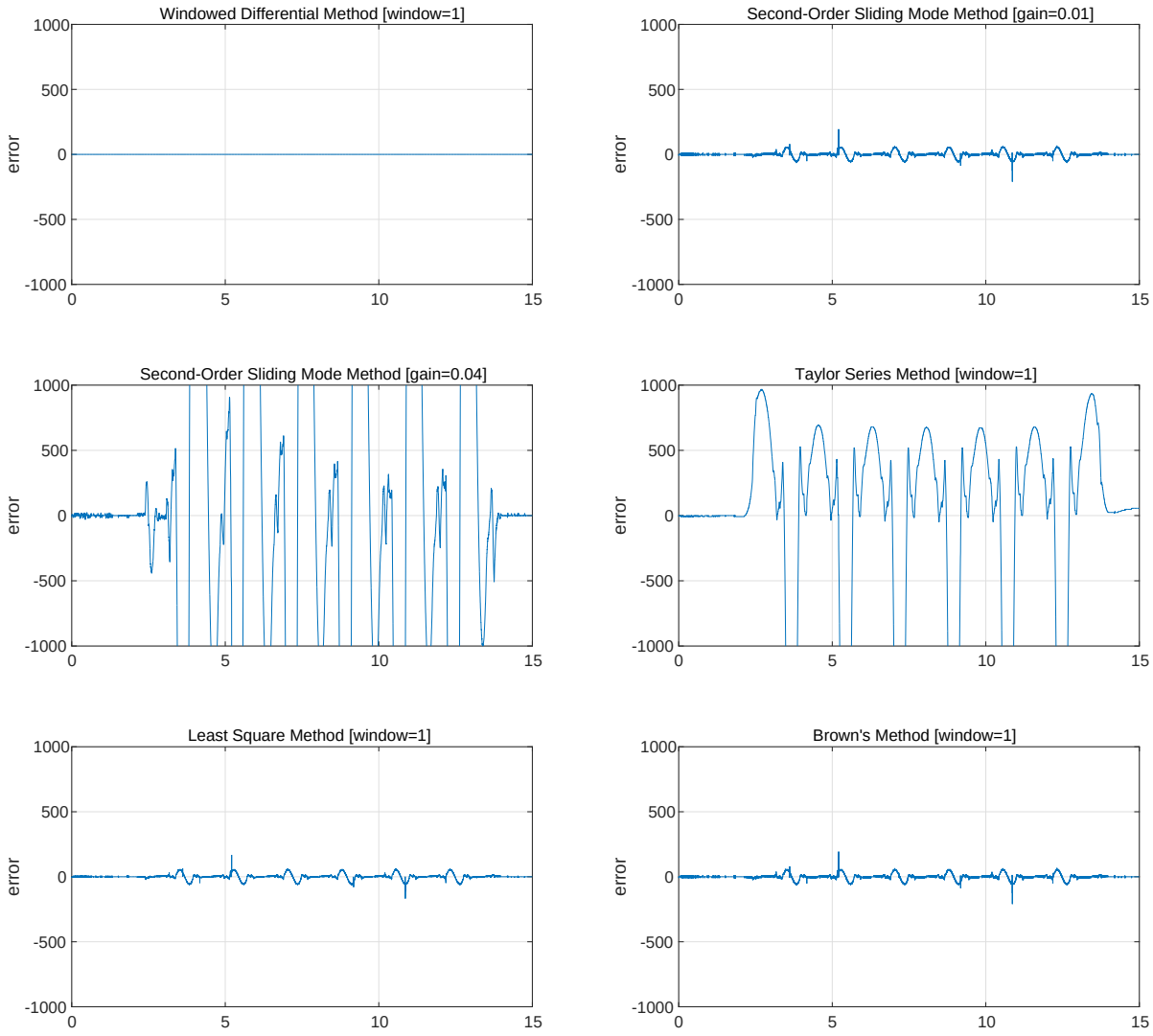


図 C.5: Comparison of five speed estimating methods error.

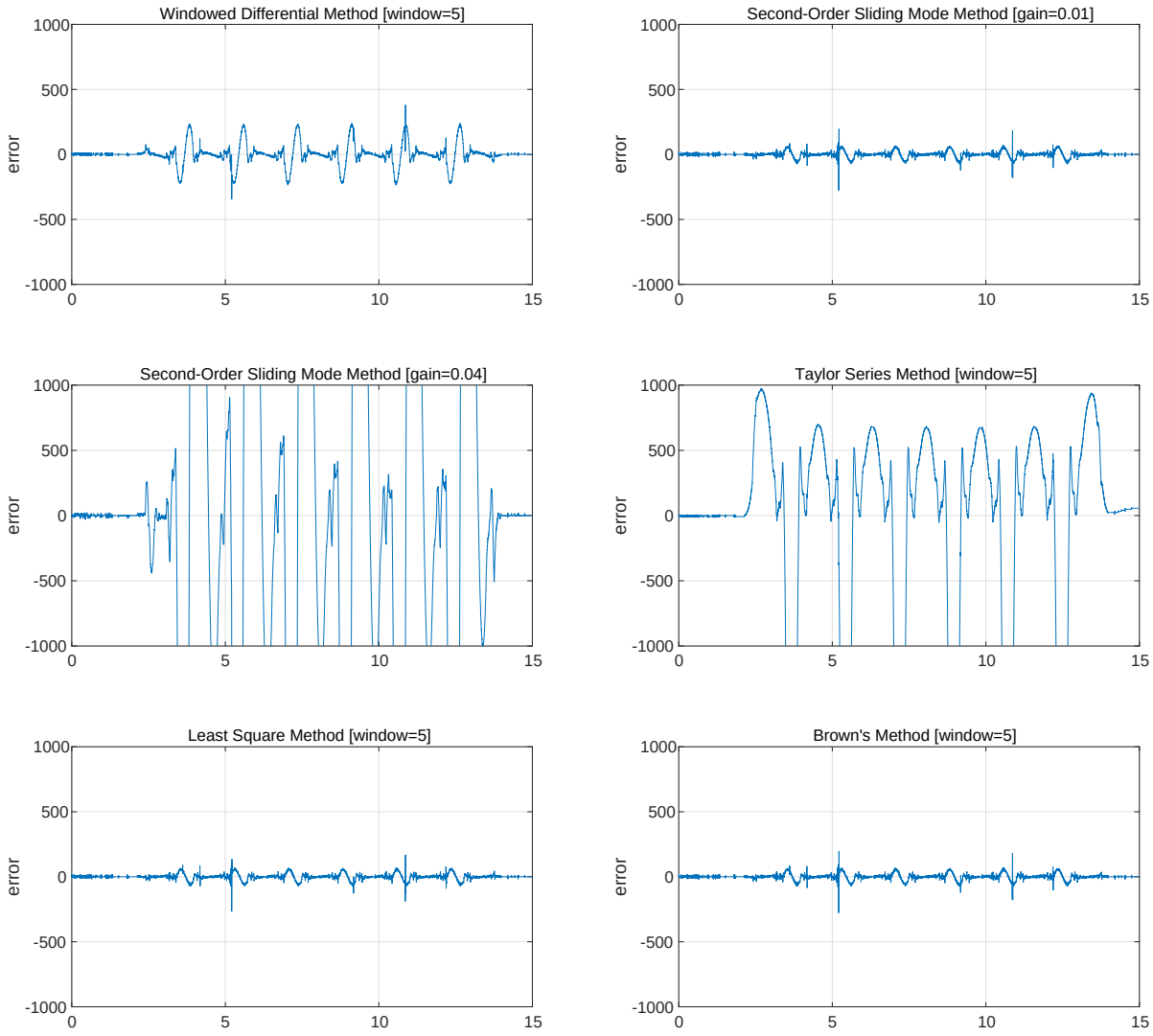


図 C.6: Comparison of five speed estimating methods error with window=5.

C.2 衝撃検出センサの実時間データ処理

C.2.1 DC 成分ウォッシュアウトフィルタ

加速度ベースの衝撃検出センサでは出力値に重力加速度が DC 成分として重畳する。衝撃検出において DC 成分による出力値のオフセットは不要なため、DC 成分をウォッシュアウトするハイパスフィルタによって重力加速度成分を排除する。

本システムではウォッシュアウトフィルタは ADC 出力値を入力とするデジタルフィルタによって構成する。X 軸、Y 軸、Z 軸の 3 チャンネルについて 5-8kHz という細粒度のサンプリング周期に対応するため FPGA の論理回路としてデジタルフィルタを構成する。I^2C クロック周波数を 1.0MHz と設定したため、サンプリング周期は 1.0MHz/174=5747.126Hz となる。

$$H_d(z) = \frac{(1-2^{-M})z^2 + (-2+2^{-M+1})z + (1-2^{-M})}{z^2 + (-2+2^{-M+3})z + (1-2^{-M+2})} \tag{C.7}$$

$M \in \mathbb{N}$ は右シフト数となり、フィルタ強さを調整するパラメータとなる。M=1-10 までの $H_d(z)$ の周波数応答特性を図 C.7 に示す。実機では歩行モーション時の着地衝撃の応答等を見ながらチューニングし M=5 を使用した。上記ハイパスフィルタにより、重力加速度や通常の動作に伴う加速度 DC 成分は除去され、純粋な衝撃加速度のみを検出することが可能となる。

C.2.2 衝撃加速度による制御パラメータの調整

ウォッシュアウトフィルタを通過した衝撃加速度値は、衝撃入力に対してパルス状の応答を示す (図 C.8)。衝撃入力の正確なモデル化が困難であるため、単純に加速度として物理パラメータに応じた制御システム中で利用することは困難である。

簡単な方法として、閾値関数を通すことで制御モードのスイッチを行うことが考えられる。ただ、単純に閾値関数を通した場合にはチャタリングによる誤判定が生じてしまうため、制御が不安定となってしまう。この問題を回避するために、以下の式によって衝撃加速度入力の平滑化を行う。

$$V_a[k+1] = \alpha V_a[k] + (1-\alpha) * |a[k]| \tag{C.8}$$

以上によって得られた平滑化された衝撃加速度入力について、ダンピングパラメータに乗じるゲインを

$$g[k] = 1 - \frac{V_a[k]}{V_{MAX}} \tag{C.9}$$

と与えることで、衝撃入力時に反射的にダンピングを弱め、外部入力に対しての抵抗力を緩和することが可能となる。図 C.8 は、腕型ロボットにおいてインピーダンス制御時に、本手法によって反射的なダンピング調整を行った際に取得されたログデータである。26 秒付近の大きな衝撃加速度に対してダンピングゲインを一時的に 0.5 付近まで大きく低減し、抵抗力を弱めていることが確認される。また、36 秒付近の衝撃は先のものほど強くはないため、V_a 値が小さくなり、結果ダンピングゲインの低減も 0.8 程度に留まっている。

C.3 関節トルク推定モジュールとトルク制御の定性的評価

本節では永松らによって FPGA 上の論理回路として実装された関節トルク推定モジュール [102] について、その概要を述べる。ブラシレスモータと減速機によって駆動される関節に対して，モータの出力トルクからモータ慣性力と関節回転摩擦を差引いたものが関節のトルクとして発揮されるものとみなし，モータに付属する電流センサおよび回転エンコーダの値から関節トルクの推定を行う．

モータに出力されるトルク発揮電流の値を i_m，モータの回転位置を θ_m，として，関節トルク τ を，

$$\tau \simeq RK_i i_m - RI_m\ddot{\theta}_m - K_{fc}\,\mathrm{sgn}\,\dot{\theta}_m - K_{fv}\dot{\theta}_m \tag{C.10}$$

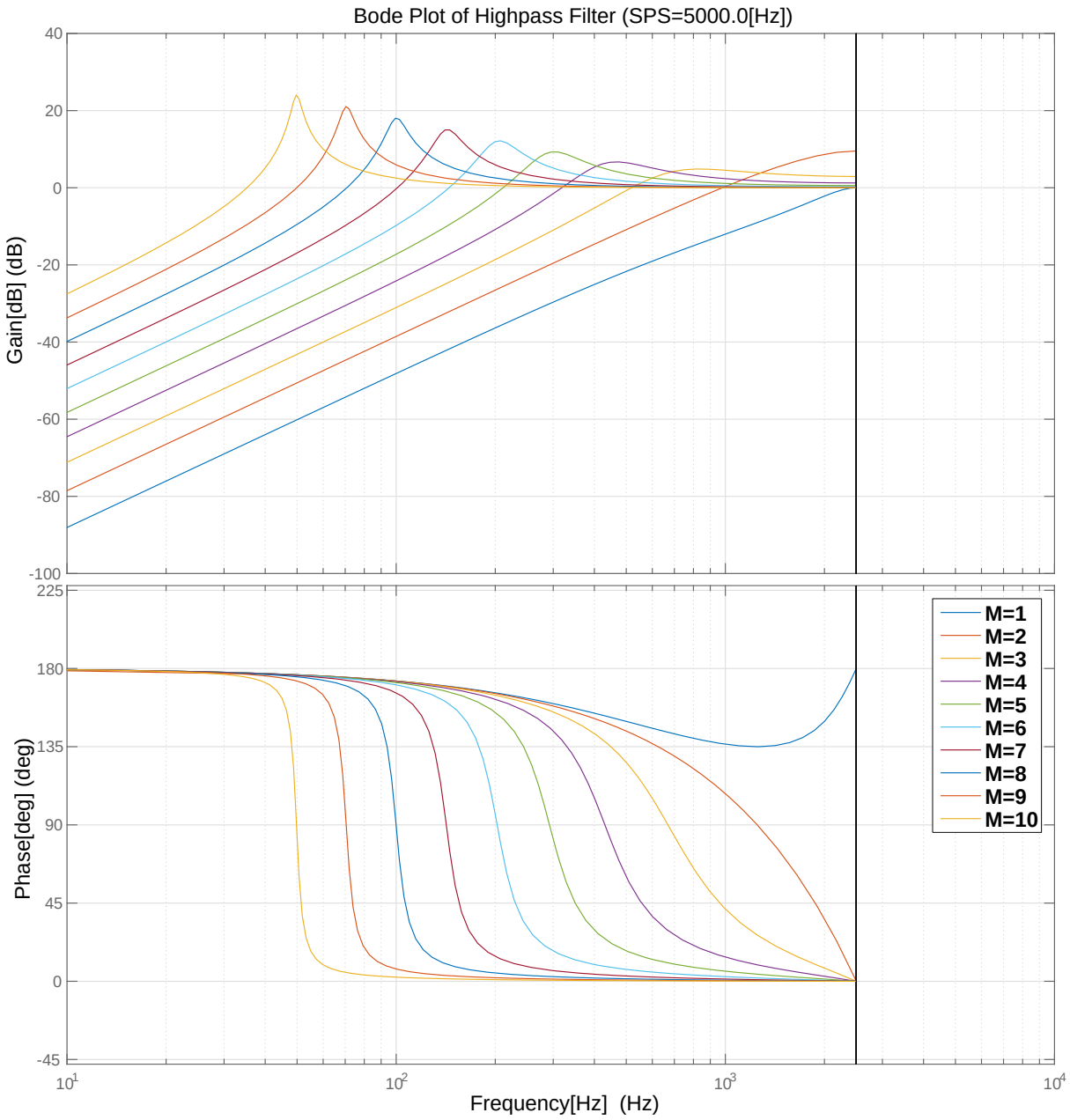


図 C.7: Bode plot of highpass filter corresponding to Eq. C.7 .

と近似する．ここで R は関節の減速比，K_i はモータの電流トルク定数，I_m はモータ回転子を含む減速前段部品の慣性モーメント，K_{fc} は関節のクーロン摩擦の大きさ，K_{fv} はモータ回転速度に対する関節の粘性摩擦係数を表す．

モータドライバに搭載された FPGA 上に関節トルク推定器を実装する．関節トルク近似式に基づいて，入力 i_m，$\dot{\theta}_m$，$\ddot{\theta}_m$ およびパラメータ R，K_i，I_m，K_{fc}，K_{fv} を与えると，関節トルク近似値 τ が求められる．

入力のうち，モータ電流 i_m は，ブラシレスモータの 3 相電流センサ値および回転エンコーダ値からベクトル変換によってトルクに寄与する q 軸電流を求めた後，ノイズを軽減する目的で LPF に通すことで得られる．モータ回転速度 $\dot{\theta}_m$ および加速度 $\ddot{\theta}_m$ は回転エンコーダ値の離散微分によって求める．モータは 1 回転あたり 2000 カウントのインクリメンタルエンコーダを搭載しており，FPGA は 24[MHz] の駆動クロックでインクリメンタルエンコーダのカウント値を取得している．実用的な関節の回転速度を考慮して，$\dot{\theta}_m$ は 1[ms] ごとのエンコーダ差分値をサンプリングして下位ビットを拡張し LPF に通して平滑化することで求め，$\ddot{\theta}_m$ は $\dot{\theta}_m$ のクロックごとの差分値を LPF に通して平滑化することで求める．

パラメータのうち，減速比 R と慣性モーメント I_m は設計値から得られる．電流トルク定数 K_i はモータ

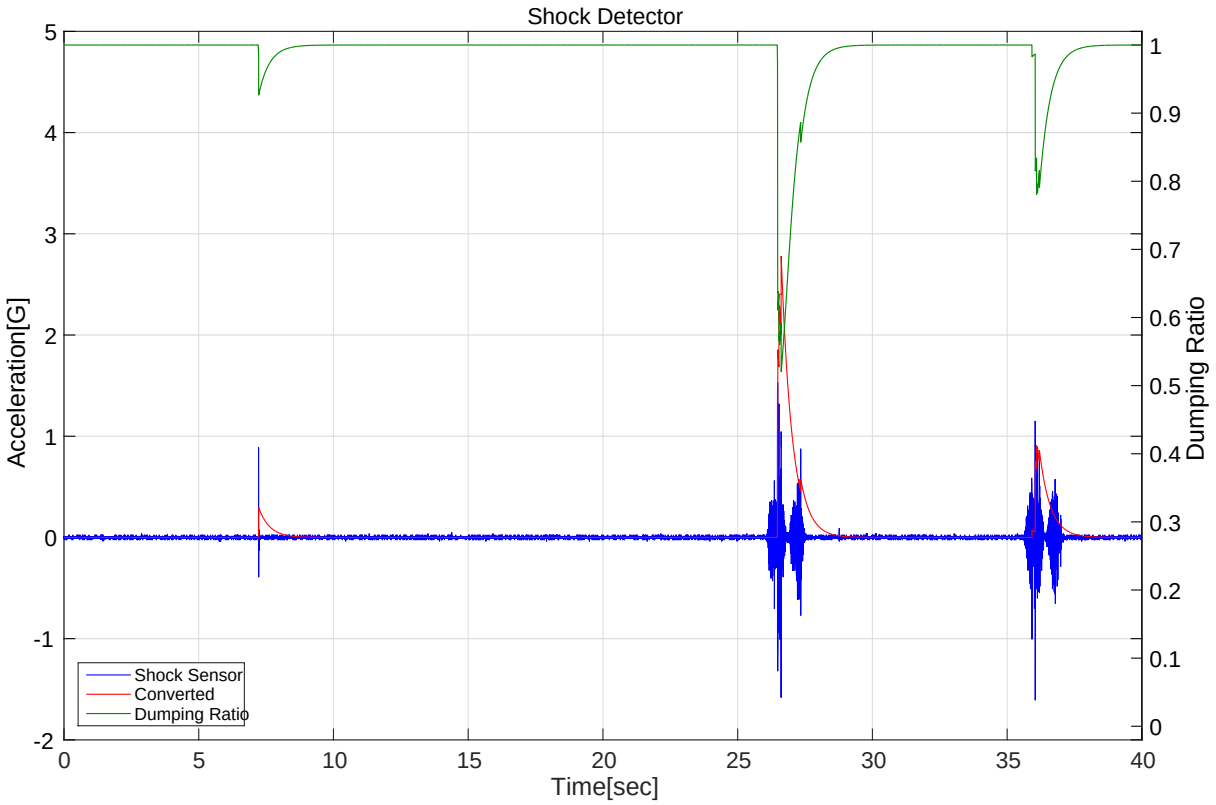


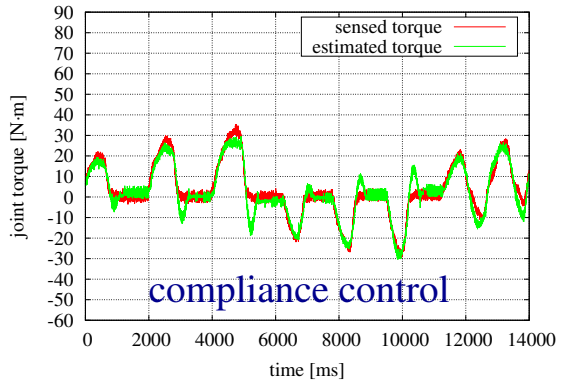
図 C.8: Dumping gain refrexion by smoothed shock acceleration input.

ごとの個体差が大きくないことから，試験用のモータ単品での実測値から同定する．クーロン摩擦 K_{fc} と粘性摩擦係数 K_{fv} はロボット実機を用いた同定実験により求めるか，困難な場合は便宜上，類似する構成の関節のパラメータを用いる．同定実験には参照用の関節トルク値が必要となるが，これは試験機に搭載されたトルクセンサ，外力の印加を測定する力センサ，あるいはロボットのモデルに基づく動力学計算のいずれかが利用可能である．

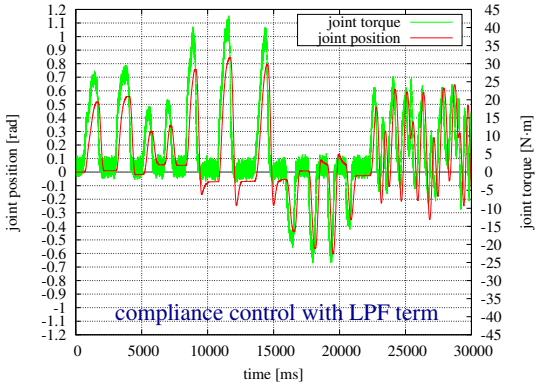
このトルク推定器について、永松らは単軸試験機に取り付けたトルクセンサ出力と比較して評価しており、図 C.9a に示す。トルク推定値は概ねトルクセンサ出力に追従しており、トルクセンサの代用として利用が可能であることが分かる。ただし、トルクを抜いた際などに見られるオーバーシュートは関節速度から算出される粘性摩擦項の影響が強く、粘性摩擦係数の精度が推定値の動的な追従精度に重要であることが判断できる。また、図 C.9b は上記単軸試験機において永松らのコンプライアンス制御時のトルク計測値と関節角度応答を示している。推定トルクのフィードバック制御であるが、発振等は発生せず、入力トルクに対して関節角度応答が速やかに立ち上がっていることが分かる。

永松らは単軸の試験機において5 kg のおもりを高さ 220mm から振り子で落として衝突させた際のトルク応答を計測しており、既存の関節角度制御を使用した場合とトルク制御を使用した場合を比較して、撃力応答トルクはトルク制御は関節角度制御の 75.4%に抑えられることを示している。また、衝突を検知して回避方向にトルクを出力することでこの撃力応答トルクは 59.5%まで低減させることが可能であることも示している。

本研究における関節剛性制御では、トルクの制御に上記のトルク推定ロジックを採用し、さらに動力学演算による目標トルク生成ロジックを統合することで、しなやかな関節動作を実現している。



(a) Sensed joint torque and estimated joint torque of the single-axis testbed. (from [102] Fig.6)



(b) Joint torque and joint position of the single-axis testbed. (from [102] Fig.8(b))

図 C.9: Evaluation of torque estimator and torque based feedback controller by nagamatsu.

謝辞

本博士論文は稲葉雅幸教授のご指導の下で執筆を行った論文であり、稲葉先生には数多くのご指導・ご教示をいただきながら研究・開発を行ったものです。ここまでまとめるまでに非常に長い時間を要してしまいましたが、私の納得がいくまで機会を与えていただき大変感謝しております。

また、本論文を整理しまとめる上で非常に有益なご意見・ご助言を頂いた國吉康夫教授、中村仁彦教授、新山龍馬講師、岡田慧准教授には感謝を申し上げます。

組み込み関連技術についてゼロに近い経験から始めた私ですが、本論文で提唱した多岐項目に渡るシステム構成要素を組めるまでに至ることができたのは、研究室教員・先輩方や慶応義塾大学の山崎信行教授、産業技術大学院大学の千代浩之助教のご助言があったからこそで、心からお礼申し上げます。トルク制御系の構築、L1 のアセンブリにおいて強力なサポートをしてくれた永松祐弥君、鈴木裕登君にも大変お世話になりました。

最後にここまで長い間支えてくれた家族に感謝の意を表して、この論文の結びといたします。

2017 年 10 月 13 日　白井　拓磨

参考文献

[1] ロボット 革命実現会議 / ロボット 新戦略. www.kantei.go.jp/jp/singi/robot/, 2015.01.23.

[2] 稲葉雅幸. 脳を 持ち 歩かないロボット による 知能ロボット の研究. 第 10 回日本ロボット 学会学術講演会予稿集, pp. 1145–1148, 1992.

[3] M. Inaba. Remote-Brained Robotics: Interfacing AI with Real World Behaviors. In *Robotics Research: The Sixth International Symposium*, pp. 335–344. International Foundation for Robotics Research, 1993.

[4] 松井俊浩, 比留川博久, 石川裕, 山崎信行, 加賀美聡, 堀俊夫, 金広文男, 斎藤元, 稲邑哲也. ヒューマノイド・ロボット のための実時間分散情報処理. 電子情報通信学会技術研究報告: 実時間処理に関するワークショップ (RTP2004), Vol. 1, No. 8, 2004.

[5] Iec 62278, railway applications - specification and demonstration of reliability, availability, maintainability and safety (rams). https://webstore.iec.ch/publication/6747, 2016.10.08.

[6] Iso/ts 15066, robots and robotic devices – collaborative robots. https://www.iso.org/standard/62996.html, 2016.10.08.

[7] 浦田順一, 岡田慧, 水内郁夫, 稲葉雅幸. 人間と 同等の関節出力を 持つ人型ロボット 実現のためのモータ温度制御に 基づく 関節駆動システムの開発. 第 26 回日本ロボット 学会学術講演会講演論文集, pp. 3E2–08, 9 2008.

[8] Dennis E. Anderson, Michael L. Madigan, and Maury A. Nussbaum. Maximum voluntary joint torque as a function of joint angle and angular velocity: Model development and application to the lower limb. *Journal of Biomechanics*, Vol. 40, No. 14, pp. 3105 – 3113, 2007.

[9] 牧川方昭, 吉田正樹. 運動のバイオメカニクス—運動メカニズムのハード ウェアとソフト ウェア—, ロボティクスシリーズ, 第 17 巻. コロナ社, 第 1 版, 2008.

[10] AV Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, Vol. 126, No. 843, pp. 136–195, 1938.

[11] 伊藤宏司. 身体知システム論. 共立出版, 2005.

[12] 真島英信, 猪飼道夫. 生体の運動機構とその制御. 杏林書院, 1974.

[13] CS Sherrington. Note on the knee-jerk and the correlation of action of antagonistic muscles. *Proceedings of the Royal Society of London*, Vol. 52, No. 315-320, pp. 556–564, 1892.

[14] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, Vol. 39, No. 10, pp. 1482–1493, October 1991.

[15] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, Vol. 36, No. SI, pp. 147–163, December 2002.

[16] K Lee, John C Eidson, Hans Weibel, and Dirk Mohl. Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems. In *Conference on IEEE*, Vol. 1588, p. 2, 2005.

[17] Ethercat technology group. www.ethercat.org.

[18] G. Cena, I.C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino. On the accuracy of the distributed clock mechanism in EtherCAT. In *2010 8th IEEE International Workshop on Factory Communication Systems (WFCS)*, pp. 43–52, May 2010.

[19] M. Rehnman and T. Gentzell. Synchronization in a force measurement system using EtherCAT. In *IEEE International Conference on Emerging Technologies and Factory Automation, 2008. ETFA 2008*, pp. 1023–1030, September 2008.

[20] 永井 正武澤田 勉. 実用組込みＯＳ 構築技法 -情報通信を 支える 基礎技術 RTOS 入門-. 共立出版, 初版, 2001.

[21] K. Suito, R. Ueda, K. Fujii, T. Kogo, H. Matsutani, and N. Yamasaki. The Dependable Responsive Multithreaded Processor for Distributed Real-Time Systems. *IEEE Micro*, Vol. 32, No. 6, pp. 52–61, December 2012.

[22] A. Ray. Introduction to networking for integrated control systems. *IEEE Control Systems Magazine*, Vol. 9, No. 1, pp. 76–79, January 1989.

[23] K. Hirai and Y. Satoh. Stability of a system with variable time delay. *IEEE Transactions on Automatic Control*, Vol. 25, No. 3, pp. 552–554, June 1980.

[24] T. Mori. Criteria for asymptotic stability of linear time-delay systems. *IEEE Transactions on Automatic Control*, Vol. 30, No. 2, pp. 158–161, February 1985.

[25] Mo-Yuen Chow and Y. Tipsuwan. Network-based control systems: a tutorial. In *The 27th Annual Conference of the IEEE Industrial Electronics Society, 2001. IECON '01*, Vol. 3, pp. 1593–1602 vol.3, 2001.

[26] N. D. Dalt, M. Harteneck, C. Sandner, and A. Wiesbauer. On the Jitter Requirements of the Sampling Clock for Analog-to-Digital Converters. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, Vol. 49, No. 9, pp. 1354–1360, September 2002.

[27] J. Skaf and S. Boyd. Analysis and Synthesis of State-Feedback Controllers with Timing Jitter. *IEEE Transactions on Automatic Control*, Vol. 54, No. 3, pp. 652–657, March 2009.

[28] A. Y. Bhave and B. H. Krogh. Performance Bounds on State-Feedback Controllers with Network Delay. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pp. 4608–4613, December 2008.

[29] Ieee osi reference model. `www.iso.org/standard/20269.html`, 2016.10.08.

[30] Nobuyuki Yamasaki. Responsive link for distributed real-time processing. In *Innovative architecture for future generation high-performance processors and systems, 2007. iwia 2007. international workshop on*, pp. 20–29. IEEE, 2007.

[31] Hermann Kopetz and G. Grunsteidl. TTP-a protocol for fault-tolerant real-time systems. *Computer*, Vol. 27, No. 1, pp. 14–23, January 1994.

[32] P. Tabuada. Event-Triggered Real-Time Scheduling of Stabilizing Control Tasks. *IEEE Transactions on Automatic Control*, Vol. 52, No. 9, pp. 1680–1685, September 2007.

[33] Ieee 802.3 ber requirements. `www.ieee802.org/3/efm/public/nov01/khermosh_3_1101.pdf`, 2016.10.08.

[34] Eunmin Choi, Sungmin Han, Jaeseok Lee, Suwon Kang, and Ji-Woong Choi. Ber analysis of can under bpsk passband can-hd interference. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 178–180, July 2016.

[35] Arm infocenter / cortex-r series programmer's guide. `infocenter.arm.com/help/index.jsp`, 2015.08.17.

[36] Arm infocenter / cortex-r4 and cortex-r4f technical reference manual. `infocenter.arm.com/help/index.jsp`, 2015.08.17.

[37] Nobuyuki Yamasaki. Responsive multithreaded processor for distributed real-time systems. *Journal of Robotics and Mechatronics*, Vol. 17, No. 2, pp. 130–141, 2005.

[38] Dependable rmt processor specification (in japanese). www.ny.ics.keio.ac.jp/research/rmt/, 2015.08.17.

[39] H. Watanabe, K. Mizotani, Y. Hatori, H. Chishiro, and N. Yamasaki. A Low Latency Real-Time Execution with Interrupt Wake-up Structure. In *Proceedings of the Embedded System Symposium 2015*, pp. 74–83, October 2015. (in Japanese).

[40] 溝谷圭悟, 羽鳥雄介, 久村雄輔, 高須雅義, 千代浩之, 山崎信行. Dependable responsive multithreaded processor における低遅延リアルタイム実行. 情報処理学会研究報告. SLDM,[システム LSI 設計技術], Vol. 2015, No. 40, pp. 1–6, 2015.

[41] 溝谷圭悟, 上田陸平, 高須雅義, 千代浩之, 松谷宏紀, 山崎信行. インプリサイス計算モデルにおける温度を考慮した動的電圧周波数制御の実機評価. 情報処理学会論文誌, Vol. 55, No. 8, pp. 1841–1855, 2014.

[42] Shinpei Kato and Nobuyuki Yamasaki. Global edf-based scheduling with laxity-driven priority promotion. *Journal of Systems Architecture*, Vol. 57, No. 5, pp. 498–517, 2011.

[43] 千代浩之, 武田瑛, 船岡健司, 山崎信行. Rate monotonic に基づく拡張インプリサイスタスク用リアルタイムスケジューリング. 情報処理学会論文誌, Vol. 52, No. 8, pp. 2365–2377, 2011.

[44] T. Shirai, K. Osawa, H. Chishiro, N. Yamasaki, and M. Inaba. Design and implementation of a high power robot distributed control system on dependable responsive multithreaded processor (d-rmtp). In *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, pp. 19–24, October 2016.

[45] Junichi Urata, Toshinori Hirose, Namiki Yuta, Yuto Nakanishi, Ikuo Mizuuchi, and Masayuki Inaba. Thermal control of electrical motors for high-power humanoid robots. pp. 2047–2052, 9 2008.

[46] Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Design of high torque and high speed leg module for high power humanoid. In *Proceedings of The 2010 IEEE/RSJ International Conference on Intelligent Robots and Syste ms*, pp. 4497–4502, 10 2010.

[47] 浦田順一, et al. ヒューマノイドロボットのための実時間プロセッサ:D-RMTP 搭載大出力モータドライバモジュールの設計. 第 28 回日本ロボット学会学術講演会講演論文集, pp. 1A3–3, 9 2010.

[48] Nobuyuki Ito, Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Development of very small high output motor driver for realizing forceful musculoskeletal humanoids. pp. 385–390, 12 2010.

[49] 永松祐弥, 白井拓磨, 大久保壮一, 熊谷伊織, 菅井文仁, 垣内洋平, 岡田慧, 稲葉 雅幸 (東大), 大沢幸平, 山崎 信行 (慶大). 実時間通信 responsive link 型体内通信系を備えた大出力ヒューマノイドにおける衝撃反射応答行動の実現. 日本機械学会ロボティクス・メカトロニクス講演会'15 講演論文集, pp. 1P2–B07, may 2015.

[50] 浦田順一, 中西雄飛, 岡田慧, 稲葉雅幸. 高速・高トルク動作のための大出力2 脚ロボットの開発. 日本ロボット学会誌, Vol. 28, No. 7, pp. 91–97, 2010.

[51] Yuto Nakanishi, Yuki Asano, Toyotaka Kozuki, Hironori Mizoguchi, Yotaro Motegi, Masahiko Osada, Takuma Shirai, Junichi Urata, Kei Okada, and Masayuki Inaba. Design concept of detail musculoskeletal humanoid "kenshiro" -toward a real human body musculoskeletal simulator-. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 1–6, 11 2012.

[52] A. Nagakubo, Y. Kuniyoshi, and G. Cheng. Development of a high-performance upper-body humanoid system. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, Vol. 3, pp. 1577–1583 vol.3, 2000.

[53] Akihiko Nagakubo, Yasuo Kuniyoshi, and Gordon Cheng. The etl-humanoid system—a high-performance full-body humanoid system for versatile real-world interaction. *Advanced robotics*, Vol. 17, No. 2, pp. 149–164, 2003.

[54] T. Asfour J. Schill, J. Bucker H. Peters, C. Klas, S. Schulz C. Sander, T. Werner A. Kargov, and V. Bartenbach. Armar-4: A 63 dof torque controlled humanoid robot. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2013.

[55] Matthias Fuchs, Christoph Borst, Paolo Robuffo Giordano, Andreas Baumann, Erich Kraemer, Jörg Langwald, Robin Gruber, Nikolaus Seitz, Georg Plank, Klaus Kunze, et al. Rollin'justin-design considerations and realization of a mobile platform for a humanoid upper body. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 4131–4137. IEEE, 2009.

[56] S Parkes and J Rosello. Spacewire- links, nodes, routers and networks. In *DASIA 2001- DAta Systems in Aerospace; Proceedings of the Conference*, 2001.

[57] SM Parkes and Philippe Armbruster. Spacewire: a spacecraft onboard network for real-time communications. In *Real Time Conference, 2005. 14th IEEE-NPSS*, pp. 6–10. IEEE, 2005.

[58] Markus Grebenstein, Alin Albu-Schaffer, Thomas Bahls, Maxime Chalon, Oliver Eiberger, Werner Friedl, Robin Gruber, Sami Haddadin, Ulrich Hagn, Robert Haslinger, et al. The dlr hand arm system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3175–3182. IEEE, 2011.

[59] Altera / cyclone v transceiver architecture handbook. `www.altera.com`, 2015.10.11.

[60] IF.HOTARU Consortium. If.hotaru 広域でシンプルな画像インターフェース. `http://if-hotaru.org/`, 2015-08-21.

[61] Hermann Kopetz. Event-triggered versus time-triggered real-time systems. In *Operating Systems of the 90s and Beyond*, pp. 86–101. Springer, 1991.

[62] 松本尚, 野村真義, 國澤亮太, 平木敬. 中粒度メモリベース通信を支援する memory-based processor ii. 情報処理学会研究報告 98-ARC-130 (SWoPP'98), Vol. 1998, No. 70, pp. 103–108, August 1998.

[63] Takashi Matsumoto and Kei Hiraki. Memory-based communication facilities and asymmetric distributed shared memory. In *Innovative Architecture for Future Generation High-Performance Processors and Systems, 1997*, pp. 30–39. IEEE Computer Society Press, October 1997.

[64] Takashi Matsumoto and Kei Hiraki. Mbcf: A protected and virtualized high-speed user-level memory-based communication facility. In *Proc. of 1998 Int. Conf. on Supercomputing (ICS'98)*, pp. 259–266. ACM, July 1998.

[65] Kunio Kojima, Tatsuhi Karasawa, Toyotaka Kozuki, Eisoku Kuroiwa, Sou Yukizaki, Satoshi Iwaishi añd Tatsuya Ishikawa, Ryo Koyama, Shintaro Noda, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Development of life-sized high-power humanoid robot jaxon for real-world use. In *Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015)*, pp. 838–843, November 2015.

[66] 垣内洋平, 白井拓磨, 菅井文仁, 大久保壮一, 熊谷伊織, 永松祐弥, 岡田慧, 稲葉雅幸, 和田喜久男, 山崎信行. ディペンダブルなロボット用組込プロセッサ系の開発 - 組込みリアルタイムシステム用ディペンダブル soc 及び sip に関する基盤技術の研究成果報告-. 第 32 回日本ロボット学会学術講演会講演論文集, pp. 1N1–05, sep 2014.

[67] Yohei Kakiuchi, Kunio Kojima, Eisoku Kuroiwa, Shintaro Noda, Masaki Murooka, Iori Kumagai, Ryohei Ueda, Fumihito Sugai, Shunichi Nozawa, Kei Okada, et al. Development of humanoid robot system for disaster response through team nedo-jsk's approach to darpa robotics challenge finals. In

Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on, pp. 805–810. IEEE, 2015.

[68] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, Vol. 3, p. 5. Kobe, 2009.

[69] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku, and Woo-Keun Yoon. Rt-middleware: distributed component middleware for rt (robot technology). In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3933–3938. IEEE, 2005.

[70] 内山勝, 中村仁彦. 岩波講座ロボット学 2 ロボットモーション. 岩波書店, 2004.

[71] Alessandro De Luca and Raffaella Mattone. Actuator failure detection and isolation using generalized momenta. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, Vol. 1, pp. 634–639. IEEE, 2003.

[72] A. D. Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1623–1630, Oct 2006.

[73] Shuuji Kajita, Osamu Matsumoto, and Muneharu Saigo. Real-time 3d walking pattern generation for a biped robot with telescopic legs. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, Vol. 3, pp. 2299–2306. IEEE, 2001.

[74] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi, and Hirohisa Hirukawa. A realtime pattern generator for biped walking. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, Vol. 1, pp. 31–37. IEEE, 2002.

[75] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, Vol. 2, pp. 1620–1626. IEEE, 2003.

[76] Satoshi Kagami, Koichi Nishiwaki, Tomonobu Kitagawa, Tomomichi Sugihara, Masayuki Inaba, and Hirochika Inoue. A fast generation method of a dynamically stable humanoid robot trajectory with enhanced zmp constraint. In *Proc. of IEEE International Conference on Humanoid Robotics (Humanoid2000)*, 2000.

[77] Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Vol. 3, pp. 2684–2689. IEEE, 2002.

[78] Junichi Urata, Koichi Nshiwaki, Yuto Nakanishi, Kei Okada, Satoshi Kagami, and Masayuki Inaba. Online decision of foot placement using singular lq preview regulation. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pp. 13–18. IEEE, 2011.

[79] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson. 3d walking based on online optimization. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 21–27, Oct 2013.

[80] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, Oct 2012.

[81] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175, May 2014.

[82] 西脇光一, 加賀美聡. 推定絶対運動状態を始点とする短周期軌道生成を用いたヒューマノイドの不整地適応歩行制御. 日本ロボット学会誌, Vol. 29, No. 1, pp. 111–121, 2011.

[83] 西脇光一, 加賀美聡. 将来にわたる zmp 許容領域を考慮したオンライン 2 足歩行軌道生成法. 日本ロボット学会誌, Vol. 30, No. 7, pp. 702–710, 2012.

[84] Youbin Peng and Michel Kinnaert. Explicit solution to the singular lq regulation problem. *IEEE Transactions on Automatic Control*, Vol. 37, No. 5, pp. 633–636, 1992.

[85] 梶田秀司, 金広文男, 金子健二, 藤原清司, 原田研介, 横井一仁, 比留川博久. 分解運動量制御: 運動量と角運動量に基づくヒューマノイドロボットの全身運動生成. 日本ロボット学会誌, Vol. 22, No. 6, pp. 772–779, 2004.

[86] M. Benallegue and F. Lamiraux. Humanoid flexibility deformation can be efficiently estimated using only inertial measurement units and contact information. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 246–251, Nov 2014.

[87] K. Masuya and T. Sugihara. Com motion estimation of a humanoid robot based on a fusion of dynamics and kinematics information. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3975–3980, Sept 2015.

[88] Ken Masuya and Tomomichi Sugihara. Dead reckoning for biped robots that suffers less from foot contact condition based on anchoring pivot estimation. *Advanced Robotics*, Vol. 29, No. 12, pp. 785–799, 2015.

[89] K. Nishiwaki and S. Kagami. Frequent walking pattern generation that uses estimated actual posture for robust walking control. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 535–541, Dec 2009.

[90] S. Kwon and Y. Oh. Real-time estimation algorithm for the center of mass of a bipedal robot with flexible inverted pendulum model. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5463–5468, Oct 2009.

[91] B. Stephens and C. Atkeson. Modeling and control of periodic humanoid balance using the linear biped model. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 379–384, Dec 2009.

[92] B. J. Stephens and C. G. Atkeson. Dynamic balance force control for compliant humanoid robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1248–1255, Oct 2010.

[93] B. J. Stephens and C. G. Atkeson. Push recovery by stepping for humanoid robots with force controlled joints. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pp. 52–59, Dec 2010.

[94] Hiroto Suzuki, Yuya Nagamatsu, Takuma Shirai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Mãsayuki Inaba. Torque based stabilization control for torque sensorless humanoid robots. In *Proceedings of the 2017 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2017)*, pp. 425–431, November 2017.

[95] Shuuji Kajita, Mitsuharu Morisawa, Kanako Miura, Shin'ichiro Nakaoka, Kensuke Harada, Kenji Kaneko, Fumio Kanehiro, and Kazuhito Yokoi. Biped walking stabilization based on linear inverted pendulum tracking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 4489–4496. IEEE, 2010.

[96] Yoshito Ito, Takuya Nakaoka, Junichi Urata, Kazuya Kobayashi, Shunich Nozawa, Yuto Nakanishi and Kei Okada, and Masayuki Inaba. Development and verification of life-size humanoid with high-output actuation system. In *Proceedings of The 2014 IEEE International Conference on Robotics and Automation*, pp. 3433–3438, may 2014.

[97] Misumi-vona ミスミの総合 web カタログ ; 制御部品・ pc 部品/電源/フェライトコア/小型フェライトコア（ ワンタッチタイプ）/ atc-5030 データシート . jp.misumi-ec.com/, 2015.09.29.

[98] Kester Walt. Analog devices, mt-001 tutorial, taking the mystery out of the infamous formula, snr = 6.02n + 1.76db, and why you should care. http://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf, 2017.06.27.

[99] William Ralph Bennett. Spectra of quantized signals. *The Bell System Technical Journal*, Vol. 27, No. 3, pp. 446–472, July 1948.

[100] M. F. Benkhoris and M. Ait-Ahmed. Discrete speed estimation from a position encoder for motor drives. In *1996 Sixth International Conference on Power Electronics and Variable Speed Drives (Conf. Publ. No. 429)*, pp. 283–287, Sept 1996.

[101] G. Bartolini, A. Damiano, G. Gatto, I. Marongiu, A. Pisano, and E. Usai. Robust speed and torque estimation in electrical drives by second-order sliding modes. *IEEE Transactions on Control Systems Technology*, Vol. 11, No. 1, pp. 84–90, Jan 2003.

[102] Yuya Nagamatsu, Takuma Shirai, Hiroto Suzuki, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Distributed torque estimation toward low-latency variable stiffness control for gear-driven torque sensorless humanoid. In *Proceedings of The 2017 IEE/RSJ International Conference on Robotics and Systems*, pp. 5239–5244, September 2017.

発表文献

[1] 白井拓磨, 小林巧実, 本郷一生, 太田茂樹, 中西雄飛, 稲葉雅幸. 筋骨格ヒューマノイドの筋長・張力ハイブリッド制御に基づくなじみ動作の実現. 第1 0 回 SICE システムインテグレーション部門講演会講演概要集, pp. 1H1–1, 12 2009.

[2] 白井拓磨, 小林巧実, 本郷一生, 中西雄飛, 水内郁夫, 岡田慧, 稲葉雅幸. 筋骨格ヒューマノイドの張力制御を利用した姿勢教示による物体拘束の獲得. 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, pp. 2P1–C19, 6 2010.

[3] 白井拓磨, 中西雄飛, 岡田慧, 稲葉雅幸. 球関節を有する腱駆動多自由度ロボットの筋張力弾性制御による環境接触行動. 第 29 回日本ロボット学会学術講演会講演論文集, pp. 3P1–3, 9 2011.

[4] Takuma Shirai, Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Whole body adapting behavior with muscle level stiffness control of tendon-driven multijoint robot. In *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2229–2234, 12 2011.

[5] 白井拓磨, 上月豊隆, 溝口弘悟, 浅野 悠紀 長田将彦, 中西雄飛, 岡田慧, 稲葉雅幸. 人体模倣筋骨格ヒューマノイド「 腱志郎」の運動制御-人体の関節構造を模した面状筋機構を有する腱駆動ロボットでの動作教示のためのシステム構成-. 日本機械学会ロボティクス・メカトロニクス講演会'12 講演論文集, pp. 1A1–K10, 5 2012.

[6] 白井拓磨, 浦田順一, 中西雄飛, 垣内洋平, 上田陸平, 久村雄輔, 山崎信行, 稲葉雅幸. 実時間プロセッサ rmtp と実時間通信 responsive link によるモータの 2 軸同期制御. pp. 2E1–01, sep 2013.

[7] Takuma Shirai, Kouhei Osawa, Hiroyuki Chishiro, Nobuyuki Yamasaki, Masayuki Inaba. Design and implementation of a high power robot distributed control system on dependable responsive multithreaded processor (d-rmtp). In *2016 IEEE 4th International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, pp. 19–24, Oct 2016.

[8] 永松祐弥, 白井拓磨, 大久保壮一, 熊谷伊織, 菅井文仁, 垣内洋平, 岡田慧, 稲葉 雅幸 (東大), 大沢幸平, 山崎 信行 (慶大). 実時間通信 responsive link 型体内通信系を備えた大出力ヒューマノイドにおける衝撃反射応答行動の実現. 日本機械学会ロボティクス・メカトロニクス講演会'15 講演論文集, pp. 1P2–B07, may 2015.

[9] 大久保壮一, 白井拓磨, 永松祐弥, 熊谷伊織, 菅井文仁, 垣内洋平, 岡田慧, 稲葉雅幸 (東京大学), 溝谷圭悟, 久村雄輔, 山崎信行 (慶應義塾大学). 分散制御システム向け実時間通信 responsive link を備えた i/o core 基板による多自由度ヒューマ ノイド制御システムの構成と評価. 第 15 回 SICE システムインテグレーション部門講演会講演概要集, Dec 2014.

[10] 長田将彦, 白井拓磨, 小林巧実, 本郷一生, 中西雄飛, 稲葉雅幸. 腱駆動機構に容易に追加可能なアドオン型非線形ばねユニットの開発. 日本機械学会ロボティクス・メカトロニクス講演会'10 講演論文集, pp. 2A2–B12, 6 2010.

[11] 溝口弘悟, 小林巧実, 白井拓磨, 長田将彦, 中西雄飛, 岡田慧, 稲葉雅幸. 筋骨格ヒューマノイドの反射系を用いた環境接触によるバランス動作の実現. 第 11 回 SICE システムインテグレーション部門講演会講演概要集, pp. 1G4–2, 12 2010.

[12] 中西雄飛, 溝口弘悟, 黒飛朋子, 梯百合子, 伊藤佳人, 伊沢多聞, 白井拓磨, 長田将彦, 小林巧実, 伊東信之, 太田茂樹, 浦田順一, 稲葉雅幸. 筋骨格ヒューマノイドの運動再評価のための目次自動生成に基づく身体運動情報記憶データベースの構築. 第 11 回 SICE システムインテグレーション部門講演会講演概要集, pp. 1G2–3, 12 2010.

[13] Yuriko Kakehashi, Tamon Izawa, Takuma Shirai, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. The trials of hula hooping by a musculo-skeletal humanoid kojiro nearing dancing motions using the soft spine. pp. 423–428, 10 2011.

[14] 伊東信之, 中西雄飛, 白井拓磨, 長田将彦, 伊沢多聞, 岡田慧, 稲葉雅幸. 電磁クラッチを用いた開放機構を持つ腱駆動アームの開発. 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, pp. 2A1–H15, 5 2011.

[15] 梯百合子, 伊沢多聞, 白井琢磨, 中西雄飛, 岡田慧, 稲葉雅幸. ロボットの体幹を用いるしなやかな動作としてのフラフープ運動の実現. 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, pp. 2P2–Q10, 5 2011.

[16] 溝口弘悟, 白井拓磨, 長田将彦, 中西雄飛, 岡田慧, 稲葉雅幸. 背骨を有する腱駆動ヒューマノイドにおける身体環境協応による歩行行動の実現. 日本機械学会ロボティクス・メカトロニクス講演会'11 講演論文集, pp. 2P2–I05, 5 2011.

[17] Yuriko Kakehashi, Tamon Izawa, Takuma Shirai, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Achievement of hula hooping by robots through deriving principle structure towards flexible spinal motion. *Journal of Robotics and Mechatronics*, Vol. 24, No. 3, pp. 540–546, 2012.

[18] Tomoko Kurotobi, Takuma Shirai, Yotaro Motegi, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Controlling tendon driven humanoids with a wearable device with direct-mapping method. In *21th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 437 –442, sept. 2012.

[19] Toyotaka Kozuki, Hironori Mizoguchi, Yuki Asano, Masahiko Osada, Takuma Shirai, Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Design methodology for thorax and shoulder of human mimetic musculoskeletal humanoid kenshiro -a thorax with rib like surface-. pp. 4367–4372, 10 2012.

[20] Yotaro Motegi, Takuma Shirai, Tamon Izawa, Tomoko Kurotobi, Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Motion control based on modification of the jacobian map between the muscle space and work space with musculoskeletal humanoid. pp. 835–840, 11 2012.

[21] Yuto Nakanishi, Yuki Asano, Toyotaka Kozuki, Hironori Mizoguchi, Yotaro Motegi, Masahiko Osada, Takuma Shirai, Junichi Urata, Kei Okada, and Masayuki Inaba. Design concept of detail musculoskeletal humanoid "kenshiro" -toward a real human body musculoskeletal simulator-. pp. 1–6, 11 2012.

[22] 上月豊隆, 溝口弘悟, 浅野 悠紀 長田将彦, 白井拓磨, 中西雄飛, 岡田慧, 稲葉雅幸. 人体模倣筋骨格ヒューマノイド 腱志郎の胸郭・ 肩の設計-肋骨状曲面を有する肩甲骨・ 胸郭構造の設計-. 日本機械学会ロボティクス・メカトロニクス講演会'12 講演論文集, 5 2012.

[23] 溝口弘悟, 浅野悠紀, 上月豊隆, 長田将彦, 浦田順一, 中西雄飛, 岡田慧, 稲葉雅幸. 体模倣筋骨格ヒューマノイド「 腱志郎」の骨盤設計 - 人体の冗長筋骨格系を模した高密度高出力筋群を備えた金属性強化骨盤・ 大腿骨の開発-. 日本機械学会ロボティクス・メカトロニクス講演会'12 講演論文集, pp. 1A1–K09, 5 2012.

[24] 中西雄飛, 長田将彦, 上月豊隆, 溝口弘悟, 浅野悠紀, 白井拓磨, 浦田順一, 稲葉雅幸. 人体模倣筋骨格ヒューマノイド 腱志郎の全身設計. 日本機械学会ロボティクス・メカトロニクス講演会'12 講演論文集, pp. 1A1–K07, 5 2012.

[25] 茂木陽太郎, 白井拓磨, 伊沢多聞, 黒飛朋子, 浦田順一, 中西雄飛, 岡田慧, 稲葉雅幸. 筋骨格ヒューマノイドの筋長-手先位置マップの逐次修正に基づく身体制御. 日本機械学会ロボティクス・メカトロニクス講演会'12 講演論文集, pp. 1P1–Q11, 6 2012.

[26] Yuto Nakanishi, Shigeki Ohta, Takuma Shirai, Yuki Asano, Toyotaka Kozuki, Yuriko Kakehashi, Hironori Mizoguchi, Tomoko Kurotobi, Yotaro Motegi, Kazuhiro Sasabuchi, Junichi Urata, Kei Okadaand Ikuo Mizuuchi, and Masayuki Inaba. Design approach of biologically-inspired musculoskeletal humanoids. *International Journal of Advanced Robotic Systems*, Vol. 10, pp. 1–18, 2013.

[27] Toyotaka Kozuki, Yotaro Motegi, Takuma Shirai, Yuki Asano, Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Design of upper limb by adhesion of muscles and bones -detail human mimetic musculoskeletal humanoid kenshiro-. In *Proceedings of The 201 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 935–940, November 2013.

[28] Yuki Asano, Takuma Shirai, Toyotaka Kozuki, Yotaro Motegi, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Motion generation of redundant musculoskeletal humanoid based on robot-model error compensation by muscle load sharing and interactive control device. pp. 336–341, October 2013.

[29] 上月豊隆, 茂木陽太郎, 浅野悠紀, 白井拓磨, 浦田順一, 中西雄飛, 岡田慧, 稲葉雅幸. 詳細人体模倣筋骨格ヒューマノイド「 腱志郎」の上肢設計. pp. 2P1–A12, may 2013.

[30] 浅野悠紀, 白井拓磨, 上月豊隆, 茂木陽太郎, 中西雄飛, 岡田慧, 稲葉雅幸. 実機モデル間誤差を 考慮したインタラクティブ操縦デバイスと筋負荷分散による冗長筋骨格ヒューマノイド の動作生成法. pp. 2G2–01, sep 2013.

[31] 中島慎介, 浅野悠紀, 白井拓磨, 中西雄飛, 岡田慧, 稲葉雅幸. 等身大筋骨格ヒューマノイド 腱志郎における筋長重心ヤコビアンに基づく バランス動作. 第 14 回 SICE システムインテグレーション部門講演会講演概要集, pp. 1501–1504, Dec 2013.

[32] 上月豊隆, 白井拓磨, 茂木陽太郎, 浅野悠紀, 中西雄飛, 岡田慧, 稲葉雅幸. ヒト の筋腱複合体を 模した張力制御筋と非線形バネ腱の統合機構による筋骨格ヒューマノイド のコンプライアンス動作の実現. 第 19 回ロボティクスシンポジア予稿集, pp. 37–42, 3 2014.

[33] 茂木陽太郎, 川崎宏治, 上月豊隆, 白井拓磨, 浅野悠紀, 中西雄飛, 岡田慧, 稲葉雅幸. 筋骨格ヒューマノイド における頸部筋群負荷を 支持可能な剛性可変脊椎構造の開発. 日本ロボット 学会誌, Vol. 32, No. 7, pp. 615–623, 2014.

[34] 浅野悠紀, 溝口弘悟, 上月豊隆, 茂木陽太郎, 白井拓磨, 浦田順一, 中西雄飛, 岡田慧, 稲葉雅幸. 終末強制回旋機構を 備えた人体模倣膝機構の実装と筋骨格ヒューマノイド による環境接触下における動作実現. 日本ロボット 学会誌, Vol. 32, No. 10, pp. 887–894, 2014.

[35] Toyotaka Kozuki, Takuma Shirai, Yuki Asano, Yotaro Motegi, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Muscle-tendon complex control by“ tension controlled muscle ” and“ non-linear spring ligament ”for real world musculoskeletal body simulator kenshiro. In *Proceedings of The 2014 IEEE International Conference on Biomedical Robotics and Biomechatronics*, pp. 875–880, August 2014.

[36] 浅野悠紀, 川崎宏治, 趙漠居, 白井拓磨, 上月豊隆, 茂木陽太郎, 大久保壮一, 矢口裕明, 岡田慧, 稲葉雅幸. 人体姿勢計測スーツを 用いた筋骨格ヒューマノイド の膝回旋自由度操作による自動車運転動作の実現. 日本機械学会ロボティクス・メカトロニクス講演会’14 講演論文集, pp. 3P1–F06, may 2014.

[37] 上月豊隆, 茂木陽太郎, 白井拓磨, 浅野悠紀, 垣内洋平, 岡田慧, 稲葉雅幸. 詳細人体模倣筋骨格ヒューマノイド「 腱志郎」の前腕設計– センサ内蔵直動アクチュエータの開発による橈骨・ 尺骨構造前腕. 日本機械学会ロボティクス・メカトロニクス講演会’14 講演論文集, pp. 3P1–F06, may 2014.

[38] 木村航平, 浅野悠紀, 白井拓磨, 上月豊隆, 茂木陽太郎, 中島慎介, 岡田慧, 稲葉雅幸. 腱駆動足首の柔軟弾性特性を 利用したヒューマノイド のペダリング適応動作の実現. 日本機械学会ロボティクス・メカトロニクス講演会’14 講演論文集, pp. 3P1–F05, may 2014.

[39] 大久保壮一, 白井拓磨, 上月豊隆, 浅野悠紀, 茂木陽太郎, 岡田慧, 稲葉雅幸. 面状筋への筋剛性制御適用による脊椎系の馴染み動作と肩甲上腕帯の動作生成. 日本機械学会ロボティクス・メカトロニクス講演会’14 講演論文集, pp. 3P1–G07, may 2014.

[40] 浅野悠紀, 上月豊隆, 川崎宏治, 茂木陽太郎, 趙漠居, 白井拓磨, 大久保壮一, 木村航平, 矢口裕明, 垣内洋平, 岡田慧, 稲葉雅幸. 反射的振る舞いに対する自動車衝突試験のための筋骨格ヒューマノイド による運転と衝突反射行動の実現. 第 32 回日本ロボット 学会学術講演会講演論文集, pp. 2D1–01, sep 2014.

[41] 垣内洋平, 白井拓磨, 菅井文仁, 大久保壮一, 熊谷伊織, 永松祐弥, 岡田慧, 稲葉雅幸, 和田喜久男, 山崎信行. ディペンダブルなロボット 用組込プロセッサ系の開発 - 組込みリアルタイムシステム用ディペンダブル soc 及び sip に関する基盤技術の研究成果報告-. 第 32 回日本ロボット 学会学術講演会講演論文集, pp. 1N1–05, sep 2014.

以上

1p～ 284p 完

博士論文

平成 29 年 10 月 13 日提出

知能機械情報学専攻
48127504 白井 拓磨