リアルタイム処理と省電力を支援するロボットミドルウェア

住谷拓馬[†1]　松原豊[†2]　中野美由紀[†1]　菅谷みどり[†1]

近年，計算機を搭載し，汎用で利用できる掃除ロボットなどが普及することで，様々なサービス開発が容易となった．本研究室でもロボットを用いたソフトウェアの開発をすすめているが，特に，物理環境の認識に応じて出力結果を変化させる必要があるロボットにとって，リアルタイム性能と省電力を達成するためにはミドルウェアの支援が必要であることが分かった．本稿では，提案するロボットミドルウェア IXM（Information eXchange Middleware）上に，見守りロボットの開発の中で得られた知見をもとに，拡張機能の設計を示し，これを議論することを目的とした．

A robot middleware for supporting real-time systems and power saving

TAKUMA SUMIYA[†1]　YUTAKA MATSUBARA[†2]
MIYUKI NAKANO[†1]　MIDORI SUGAYA[†1]

In recent years, such a cleaning robot which can be universally used which is provided with a computer has become widespread. So, various service development has become easy. In our laboratory, we are developing software using the robot. The robot is required varying the output in accordance with the recognition of the physical environment. Therefore, it was found that in order to achieve power saving and real-time performance of the robot, the support of middleware is required. In this paper, We discussed that have designed extension of the robot middleware IXM based on the findings obtained in the development of the elderly-care robot.

1. はじめに

近年，お掃除ロボットや警備ロボットなどの自律移動ロボットが，一般家庭やビルなどで利用されている．本研究室でも，移動ロボットの活用方法を研究するために，移動ロボット上にセンサやカメラを取り付けた見守りロボット[1]を開発している.ロボットを自律的に動作させるためには，センサによる入力情報をもとにロボットを制御する必要がある．例えば，人物を追従するロボットでは，カメラ画像やジェスチャを認識するソフトウェアを動作させながら，その値を収集してロボットのタイヤを回転させるためのモータ処理を別のソフトウェアで実行する.このように，ロボット制御ソフトウェアの開発環境においては，デバイスごとに異なるソフトウェアが必要になり，複数のソフトウェアを統合・連携させる必要がある．

ロボット分野では，こうした問題は早くから認識されており,いくつかのミドルウェアが提案されている.例えば，RT ミドルウェア（RT-Middleware：RTM)[2]は，複数のソフトウェアモジュールを組み合わせてロボットシステム（RT システム）を構築する為のソフトウェアプラットフォームの標準規格が提案されており,OpenRTM-aist[3]，OpenRTM.NET[4] などの実装がある．この規格では，コンポーネント間で情報をやり取りするためのインターフェイスが提供されており，この規格に準拠するコンポーネント間であれば容易に通信が可能となる．しかしながら，複数のデバイスを連携させるために重要となる，ソフトウェア間での通信と情報共有の機能に関しては，CORBA を利用することだけが示されているものの，具体的な方法はそれぞれの実装に依存しておりアプリケーションごとに検討しなければならないという問題がある．

ROS（Robot Operating System)[5]では，ソフトウェア間の通信は，メッセージパッシングを用いたプロセス間通信を利用している．具体的には，プロセス間での情報共有のために，publisher/subscriber モデルを利用する事で，複数のデバイス間で情報を共有しやすい仕組みになっている．その一方で，アプリケーション開発者がハードウェア構成を意識する必要がないよう抽象化して隠蔽するために，処理性能を必要とする場合には，独自にチューニングしなくてはならないという問題がある．

我々は，ロボット制御システムの制御アプリケーションを効率良く開発するために，ロボット制御システム向けミドルウェア IXM（Information eXchange Middleware）[6]を提案している．IXM は，制御アプリケーションを構成する複数のプロセスの非同期実行と停止，プロセス間通信という基本的な制御を支援するためのインターフェイスを提

†1 芝浦工業大学
SHIBAURA INSTITUTE TECHNOLOGY
†2 名古屋大学
Nagoya University

供することで，複数の開発環境を統合する必要がないという利点がある．さらに， ROS の問題に示したデータ共有時の性能向上や，従来のミドルウェアで考慮されていない電源管理の仕組みを提供することで，ロボットミドルウェアの有効性の向上を目指している．

本論文では，まず，ロボット制御アプリケーションの要求を明確にするために，見守りロボットを対象として，その動作性能を調査した．その結果，処理時間の大部分を占めているのは，プロセス間でのデータ共有処理であることが判明した．そこで，共有メモリを使用したプロセス間通信機能を実装することで，従来の，ファイルによるデータ共有に比べて，リアルタイム性が向上し，応答時間を約 10 分の 3 に向上できることを確認した．さらに，省電力化に関しては，PC による制御から，性能の異なるヘテロジニアス・マルチコアプロセッサを搭載したマイコンボードによる制御に変更し， IXM でプロセスごとの CPU 利用率を監視して，見守りロボットの制御性能を維持できる範囲で CPU コアの動作周波数を下げる機能や，一時的に不要となった CPU コアの電源を切るといった省電力機能を支援する構想について述べる．

本論文の構成は，次の通りである．まず，2 章で，対象とする見守りロボットの概要について述べる．次に，第 3 章で，見守りロボットの要件について述べ，第 4 章でそれらの要件に対する考察と，要件を満たすための追加機能を提案する．そして，第 5 章で関連研究について述べ，最後に，第 6 章で本論文の結論と今後の展望について述べる．

2. ロボットの処理の特性

ロボットの一般的な処理の特性は，センサにより現実世界の情報を取得し，それを分析した結果に基づき何らかの動作を行うというものである(図 1)．こうした動作は，大きくわけて入力部と出力部に分けられる．例えば，外界の認識のために多数のセンサを搭載した場合は，多数のセンサプログラムによる入力があり得ることになり，その入力に合わせた出力の割り当てや組み合わせなどの処理が発生する．

こうした場合に備えて，我々は入力部と出力部をそれぞれ別の機能を提供するプロセスをとして設計するものとし，名前をモニタリングとコントロールと定義した．これらのプロセスはある意味サーバとして機能し，これらのサーバ同士が互いに，プロセス間の通信によりデータをやり取りする形にした．これらを統合的に管理するために，ミドルウェアである IXM 経由により，提供するものとした．．我々が開発した見守りロボットも，基本的には，モニタリングと，コントロールのモデルに従って動作している（図 1）．

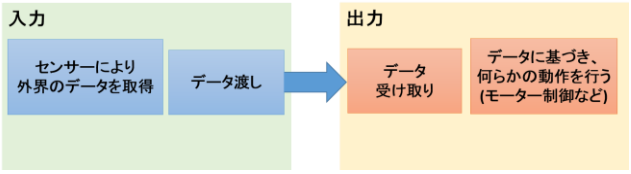

図 1　ロボットの処理の特性

Figure1　Characteristics of the process of the robot

,

3. ロボットサービスとミドルウェア

1 節に述べたように，我々は，ロボットの要求を明確にするために，汎用ハードウエア（掃除機）を用いたロボットサービスを開発している[1]．本論文で，ロボットミドルウェアについて議論するにあたり，ロボットサービスを実現するための手法について議論し，それを支援するミドルウェアとその問題について述べるものとする．本節では，ロボットサービスを主に，見守りロボットとし，その機能概要とシステム構成について述べる．

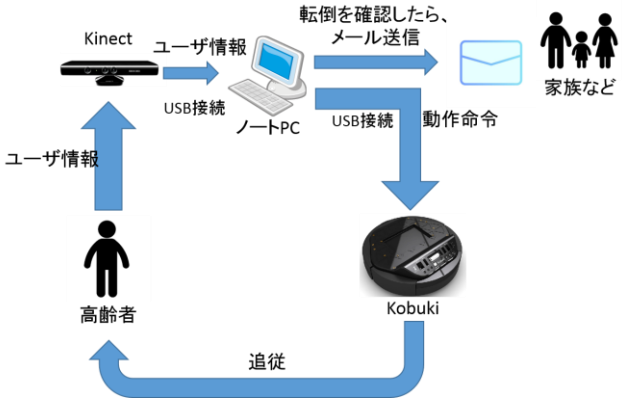

図 2　見守りロボットの動作概要

Figure2　Outline of the operation of the elderly-care robot

3.1 動作概要

ここで議論の対象とする見守りロボットは，図 2 に示すように，一般家庭で生活する高齢者や障害者（以降，ユーザと呼ぶ）を主な監視対象とし，ユーザを追従して，何らかの理由で転倒した際には，それを検出して家族に電子メールによって通知するシステムである．ユーザが常時センサを身につけるシステムと比較すると，ユーザ自身がセンサ類を身に着ける必要がないこと，ユーザのプライバシーを配慮して，個人情報は扱わないという特徴がある．

3.2 ハードウェア構成

図 3 に，見守りロボットのハードウェア構成を示す．

図 3　見守りロボットのハードウェア構成
Figure3　Hardware configuration of the elderly-care robot

ロボットは，家庭用として最も普及している，円形の掃除機ロボットと同機種の移動ロボット Kobuki[7]を用いた．Kobuki は，軽量で，車輪，バンパーなどを装備しており，シリアル通信として，USB を利用することができるため，PC との接続が容易で，プロトタイプシステムの開発に適している．ユーザの状態を把握するために，人認識センサである Kinect[8]を用い，距離情報を取得する．Kinect は，投光した赤外線レーザの反射から三角測量により画面上の各点の深度を算出することで，3D での距離の把握が可能である．本研究では，Kinect センサをロボットに搭載し，これらが連携できるハードウェア構成とした．

3.3 ソフトウェア構成

図 4 にソフトウェア構成図を示す．

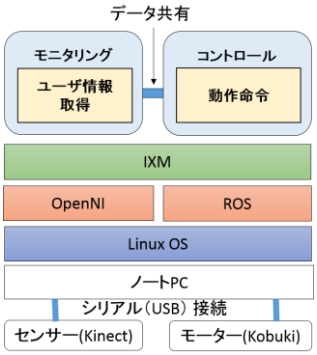

図 4　見守りロボットのソフトウェア構成
Figure4　Software configuration of the elderly-care robot

ソフトウェア構成は，Kinect のセンサを利用して，ユーザの情報を取得するプロセスと，ユーザ情報をもとに，ロボットの動作を制御する動作命令プロセスで構成する．これらのプロセスでは，ユーザ情報を，IXM を介して共有することで連携した動作を可能とした．そして，情報を取得するプロセスは，モニタリングであり，ロボットの動作を制御するプロセスは，コントロールである．なお，これまでの見守りロボットのソフトウェアでは，モニタリングアプリケーションで取得した対象者のユーザ情報をファイルによってコントロールアプリケーションと共有している．

見守りロボットのソフトウェアプラットフォームとして，我々は，Linux（Linux 3．5．0-17-generic)と ROS(ver. Groovy Galapagos）をベースとした開発環境を構築した．本開発環境では，Kobuki を動作させるために，ROS を用いた．ROS とは，ソフトウェア開発者のロボット・アプリケーション作成を支援するライブラリやツールとして，ハードウェア抽象化，デバイスドライバ，ライブラリ，視覚化ツール，メッセージ通信，パッケージ管理などを提供しているミドルウェアである．また，Kinect を動作させるために，Kinect のライブラリである OpenNI(ver． 1．5．4．0)を用いた．OpenNI とは，Kinect のデバイスをコントロールする Device 部とそのデータから画像処理を行い，ジェスチャ認識などを行うミドルウェア部を持ち，それらを統合して扱うインターフェイスとなっている．

3.4 IXM (Information eXchange Middleware)

ここまでで述べてきたように，見守りサービスの実現においては，センサ入力と，その入力をもとにした解析，解析結果による制動というように，それぞれ異なるアプリケーションから得られる処理を連携させる必要がある．センサ入力処理のための開発環境，出力処理のための開発環境はそれぞれ異なることが多く，扱いづらい問題がある．そこで，異なる言語やライブラリの入出力結果を連携するために，これらを隠蔽し，処理を透過的に行うためのミドルウェアとして， IXM(Information eXchange Middleware)を開発した[1]．IXM には，複数のアプリケーションの非同期実行と停止を行う機能と，アプリケーション間の通信，データ共有を行う機能がある．

見守りロボットでは，図 4 のように，モニタリングとコントロールのアプリケーションを非同期実行し，データ共有部分を提供している．モニタリングとコントロールを別々のプロセスとして実装することで，それぞれをサーバとして，多様な入出力を扱うことができる．また，これらを異なる周期で動作させることなども可能となる．例えば，見守りロボットのモニタリングのアプリケーションであるユーザ情報取得プロセスは，0.3 秒に一回共有データ書き込みを行い，コントロールのアプリケーションである動作命令プロセスは，1 秒に一回共有データ読み込みを行っている．

また，IXM はこれらのプロセスを統一的に動作させるためのコマンドを実装しており，コマンドから複数のアプリケーションを実行することができる．

3.5 IXM の基本設計

IXM は，移動ロボット上でのソフトウェア開発者に向けたミドルウェアである．異なるソフトウェア環境を隠蔽し，ソフトウェア開発者が用意に拡張機能を利用できることを目的としている．また，遠隔地から移動ロボットを操作するための通信機能を提供する．具体的には，ソフトウェア開発者が使用できる機能は，コマンドによる複数プロセス

の実行と停止，ファイルとソケットによるプロセス間の通信である．

IXM の基本的な機能である統一的なインターフェイスの提供を実現するために，API を設計した． API の設計にあたっては,対象とする見守りロボットに絞ったものとした．図 5 に整理した見守りロボットの機能関係を示す．

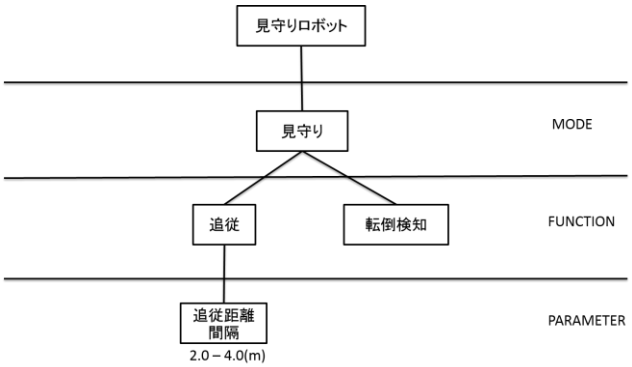

図 5　見守りロボットの機能関係図

Figure 5　Function relationship diagram of the elderly-care robot

これから見守りロボットの機能が拡張されていくことを考慮し，見守りロボットの主な振る舞いの一つとして，見守りとした．次に，見守りの動作に必要な別々のプロセスを整理した．見守りにおいて，追従，転倒検知の二つとし，そして，追従の追従距離間隔のパラメータを設定できるものとした．また，図 5 のように階層を設けそれぞれ，MODE，FUNCTION，PARAMETER とした．IXM のコマンドを使用して移動ロボットを動作させる際に，初期化の段階で MODE の選択，実行するプロセスの選択，PARAMETER の設定を行えるものとした．実装した API を下記に示す．

・Ixm Set/Get Mode(struct ixm_service *ixm)
・Ixm Set/Get Func(struct ixm_service *ixm)
・Ixm Set/Get Param(struct ixm_service *ixm)
・Ixm Start/Stop(struct ixm_service *ixm)

Set/Get は，MODE，FUNCTION，PARAMETER を移動ロボットに設定，または，設定の取得をする．これにより，現在の移動ロボットの状態を取得することと，初期化を行うことができる．Start/Stop は，必要なソフトウェアを実行，停止する．複数のソフトウェアを実行する際には，それぞれのソフトウェアごとに子プロセスを生成し，非同期に実行できる．

4. 要件定義

本章では，第 3 章で述べたロボットでの見守りロボット制御システムを用い，ミドルウェアへの課題を明確にし，それに基づき，IXM の基本設計を拡張することを目標とした調査について述べる．

結論を述べると，正常に動作できるような要求を満たすためのリアルタイム性が必要であることと，ロボットの実用を考えた場合に駆動時間を増やすための省電力化が必要であることである．また，システムに異常があった場合の誤動作防止のための仕組みが必要である．この三つの課題の詳細について述べていく．

4.1 リアルタイム性

多くのロボットにおける処理の特性として，センサにより現実世界のデータを取得し，それを分析した結果に基づき何らかの制御を行うというものである．このデータ共有の際の処理時間がロボット全体の応答時間に関わり，どの速度まで反応できるかのリアルタイム性が決まる．

4.1.1 見守りロボットの制御システムの処理時間調査

見守りロボットの処理において，ユーザ情報取得，データ共有，追従動作命令の 3 つの流れの中で，どの処理に最も時間がかかっているのかを調査した．実際にユーザの見守り処理を実行し，3 つの処理の流れを 1 ループとし，100 回計測を行った．ノート PC は，DELL Vostro 3550 で，CPU の動作周波数は Intel Core i3 2．1GHz×4，メモリ 3．9GiB のものを使用した．表 1 に処理時間の 100 回の平均を示す．共有したデータは，4kbyte のテキストファイル一つであり，排他制御は行っていない．単位はマイクロ秒である．

	骨格情報	データ共有	動作命令
処理時間	7.9	455	3

表 1　見守りロボットの処理時間の平均値

Table1　The average value of the processing time of the elderly-care robot

結果として，一連の処理の中でデータ共有に最も時間がかかっていることが確認できた．よって，このデータ共有方法を考察する必要がある．既存の見守りロボットでは，ライブラリ関数を用いてファイルによるデータ共有を行っていた．実装によって処理時間に違いが出るかを調査するために，システムコールを用いてファイルによるデータ共有を行う手法と，共有メモリによるデータ共有を行う手法も実装して実験を行った．ユーザ情報取得と動作命令に関しては，実装の変更は行っていない．実験の環境と方法は前回と同様である．

平均	骨格情報	データ共有	動作命令
ライブラリ	7.9	455	3
システムコール	8.7	146.7	3.5
共有メモリ	6.7	157.5	2

表 2　処理時間の平均値

Table2　The average value of the processing time

標準偏差	骨格情報	データ共有	動作命令
ライブラリ	19.4	65.2	1
システムコール	20.9	28	6.4
共有メモリ	2.6	36.6	1.2

表 3　処理時間の標準偏差値

Table3　Standard deviation value of the processing time

最悪	骨格情報	データ共有	動作命令
ライブラリ	199.9	812.3	12.8
システムコール	215.2	346.1	66.6
共有メモリ	18.8	298.7	11.5

表 4　処理時間の最悪値

Table4　Worst value of the processing time

表 2 に処理時間の 100 回の平均，表 3 に標準偏差値，表 4 に処理時間の最も遅かった場合の値(最悪時間)を示す．単位はマイクロ秒である．データ共有処理の平均時間は，システムコールによる実装が最も短かった．次に平均時間が短いのは，共有メモリによる実装であった．表 3 より，出力タイミングのばらつきも，システムコールによる実装が最も小さかった．しかし，表 4 より，処理時間の最悪時間は，共有メモリによる実装が最も短かった．ライブラリとシステムコールに差があるのは，ライブラリでは，小さなデータにも関わらずバッファを使用している点であると考えられる．これについては，まだ憶測であり，原因を特定するための調査が必要である．平均時間と，出力タイミングの安定性の観点では，システムコールによる実装が優れている．しかし，今回はリアルタイム性が重要であるため，最悪時間が最も短い，共有メモリによる実装が適すると思われる．また，ファイルによるデータ共有であると，通信ごとにファイルを用意しなければならないという問題もある．

以上より，ロボットのアプリケーションを開発する際の課題として，

1）　プロセス間の通信方法
2）　通信ごとのチャネル
3）　通信そのものの時間

という 3 つの課題がある．

4.2 省電力化

ロボットの実用化を考えた時にロボットが連続で動作できる時間を考慮しなければならないが，バッテリ消費を考慮した機能をミドルウェアが持っていないという問題がある．省電力化を行うにあたって，動作周波数の異なる CPU を持った高性能なマルチコアボードにより，ソフトウェア側での細かい制御が必要である．しかし，OS 側には制御があるが，簡易に利用できるソフトウェア側での制御を行うための仕組みがまだ十分に提供されていない．図 4 の例では，モニタリングとコントロールのそれぞれのアプリケーションに動作周期管理が必要である．

4.3 安全性

ロボットを要件通りに動作させるためには，何らかの異常が発生した場合に備える必要がある．例えば，見守りロボットにおいては，センシングの値が異常値であった場合に，ロボットがその値をもとに制御したために，見守り対象者にぶつかる問題があった．対策としてセンシングの入力値の範囲を詳細に定め，これらの問題を解決した．さらに，ロボット利用者の安全性を確保するためには，非安全系と安全系のアプリケーションが単一のシステムに混在する場合には，非安全系(信頼性の低いものを含む)のアプリケーションの異常動作が，安全系のアプリケーションの動作に影響を及ぼさない仕組みを導入する必要がある[10]．

5. 提案

三つの問題に対し，ロボットミドルウェアとして IXM を提案する．本章では，性能向上のための共有メモリサーバおよび，電源管理の基本機能についての設計を示す．

5.1 設計

ファイル共有のアクセス速度軽減のための共有メモリを用いたメモリサーバの導入と，プロセスの実行時間管理によるモータ実行制御と CPU のコア制御を行う機能を追加することで，性能の向上および，電源管理を行うことができるようにする．また，計算機動作の安全性を確保するための仕組みを提供する．これらの目的を実現するために，下記の 5 つの機能を設計する．

1）データ共有サーバ機能
2）CPU 使用率監視機能
3）動作命令周期変更機能
4）プロセスのコア移動機能
5）安全機能

見守りロボットを対象として IXM を設計し実装するにあたっての基本構成および基本構成を構成するプロセス関係を図 6 に示す．機能とプロセスは一対一で対応している．

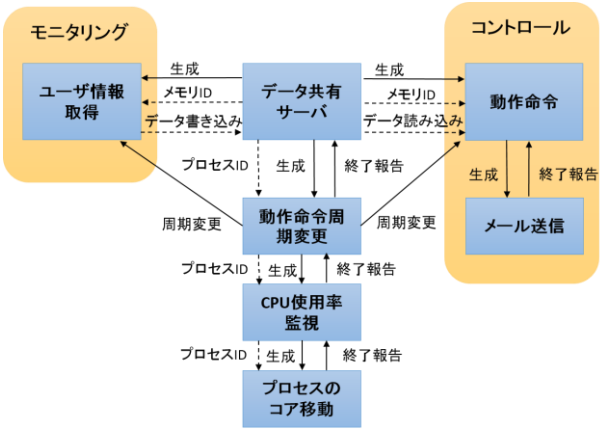

図 6　プロセス関係図

Figure6　Process relationship diagram

IXM は，データ共有サーバ，動作命令周期変更，CPU 使用率監視，プロセスのコア移動を提供する．基本的に動作しているものは，データ共有サーバとユーザ情報取得と動作命令である．その他は，必要に応じて生成される．次に，それぞれの機能の設計の詳細について述べる．

5.1.1 データ共有サーバ機能

データ共有サーバプロセスは，共有メモリを保有し，プロセス間のデータの受け渡しを補助し，データ共有のアクセス速度を改善する．メモリに保持するデータは，モニタリングとコントロールのプロセス ID と，モニタリングとコントロールがやり取りするデータである．プロセス ID は，動いているプロセスを把握するために取得する．また，このプロセスが最初に実行されモニタリングとコントロールのプロセスを生成するものとなる．その際に，共有メモリの ID を送信し，またモニタリングとコントロールのプロセス ID を取得し，保持する．データのやり取りは同期，非同期を選べるものとする．共有するデータに変化が見られない場合には，動作命令周期変更プロセスを生成する．

5.1.2 動作命令周期変更機能

動作命令周期変更プロセスは，モニタリングとコントロールがある一定周期で動作しているものを，必要に応じて周期を変更させるものである．周期の変更は，各プロセスに値を返すことで変更を行う．データ共有サーバプロセスからモニタリングとコントロールの動作周期を把握し，共有しているデータに大きな変動がない場合には，動作周期を落とす．もしくは，動作自体を停止する．これにより，カメラやモータのようなハードを動かす頻度を減らし，消費電力を低減する．

5.1.3 CPU 使用率監視機能

CPU 使用率監視プロセスは，モニタリングとコントロールのプロセスがどの程度 CPU を使用しているかを監視するプロセスである．現在のプロセスの状態を把握する．現在，ノート PC でロボットの制御をしているが，ロボットの軽量化と電力量計測を行うために，マルチコアプロセッサを載せたボードである HardKernel 社の ODROID[9]に置き換え，CPU の使用率監視を行う．ODROID は，1．6GHz と 1．2GHz の CPU を載せており，それぞれ 4 つのコアを持っている．1．6GHz の CPU は A15，1．2GHz の CPU は A7 と定義されている．図 7 に ODROID の概観を示す．

図 7　ODROID の概観

Figure7　Overview of ODROID

このODROID上で動作しているプロセスのCPU使用率を監視することを想定している．

プロセスのコア移動プロセスは，CPU 使用率監視プロセスから生成される．CPU affinity マスクを設定し，動作しているプロセスを適するコアへ割り当てる．設定するパターンを図 8 に示す．

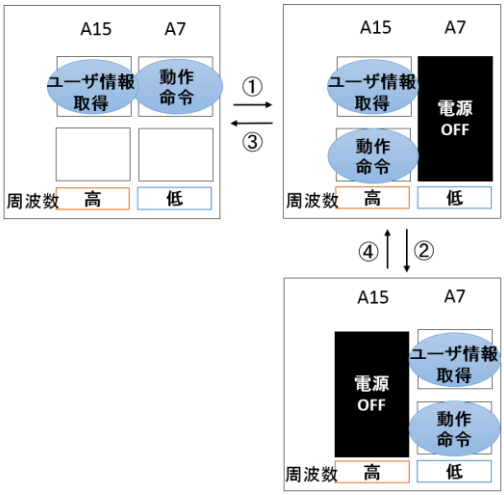

図 8　CPU 使用率監視の動作

Figure8　Behavior of the CPU management

初期状態は，A15 と A7 両方を使用する状態である．A15 のみで性能を維持できると判断した場合は，A15 にプロセスを移動し，A7 の電源を落とす状態へと遷移する依頼を行う．A7 のみで性能を維持できると判断した場合は，A7 にプロセスを移動し，A15 の電源を落とす状態へと遷移する依頼を行う．逆も同様である．条件式は以下のとおりである．X:ユーザ情報取得プロセス，Y:動作命令プロセスとする．α は，安全な動作を保障するためのマージンである．この条件式は，プロトタイプのものであり，実際にアプリケーションを動かす実験を行い，改善していく予定である．

① $x + \frac{1.2}{1.6} y + \alpha < 100$

② $\frac{1.6}{1.2}(x + y) + \alpha < 100$

③ $(x + y + \alpha \geqq 100)\ \&\&\ (\frac{1.6}{1.2}(x + y) + \alpha \geqq 100)$

④ $x + y + \alpha \geqq 100$

設定後，設定後の状態を CPU 使用率監視プロセスへと報告する．

5.2　シーケンス

見守りロボット制御システムに，提案した IXM を実装し，すべての機能が動作した際のシーケンスを図 9 に示す．

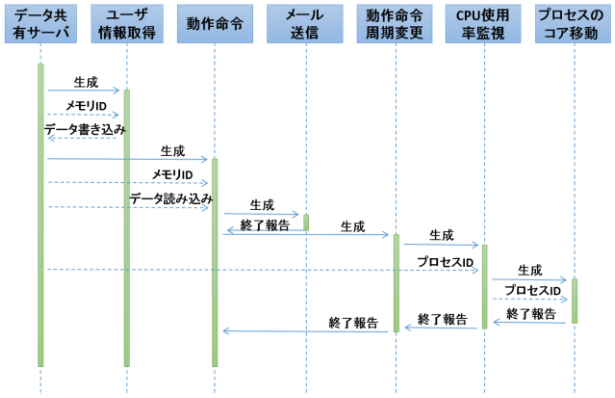

図 9　シーケンス図

Figure9　Sequence diagram

まず，初期の段階でデータ共有サーバプロセスが，見守り対象者のセンシングのための情報取得プロセス（以降，ユーザ情報取得プロセス）と動作命令プロセスを生成する．その際にメモリの ID を両者に渡す．ユーザ情報取得プロセスは，受け取ったメモリ ID にデータを書き込む．動作命令プロセスは，受け取ったメモリ ID からデータを読み込みロボットの制御を行う．その時に，対象人物が転倒し，しばらく動かない場合にはメール送信をする．一方，対象人物に転倒以外で動きがなくなった場合には，動作命令周期変更プロセスを生成する．動作命令周期変更プロセスは，ロボットの動作周期を減らすことで消費電力を低減する．その後，CPU 使用率監視プロセスを生成する．CPU 使用率監視プロセスは，生成された直後にデータ共有サーバプロセスから，実行中のプロセスの ID を読み出す．そして，対象のプロセスの CPU 使用率の監視を行う．性能を落とさずに低周波数の CPU でプロセスを実行できると判断した場合，CPU 使用率監視プロセスを生成し，使わない CPU の電源を落とす．

5.3　安全性の確保

システムが要求されたとおりに正しく動作する事を保証するためには，アプリケーションの信頼性が低い場合に，異常に備えた仕組みを導入する必要がある．本研究では CPU 使用率監視プロセスにおいて，他のプロセスが異常な動作をした場合に，それをコンテナへと隔離し，システムの動作を守る機能を追加するものとした．隔離は，システム設計に置ける安全性の確保として重要な手段である．

コンテナへと隔離する方法としては，移動のための安全機能の設計を行うものとする．本来，コンテナへの隔離は，プロセス自身が隔離のための仕組み持つ必要があるが，通常，異常を発生するプロセスには，そうした機能は実装されていないことが多い．そこで隔離処理を持った安全装置をプロセスに付加して実行し，異常プロセスを発見した際にはこれを作動させることで，実現するものとした．

実現のために，安全装置として動作するプロセスを実装し，IXM からシグナル通知が届いた際に，安全装置経由で隔離処理を実行するものとした．実現のために fork を利用し，親プロセスが異常プロセスと安全装置の二つのプロセスを生成する．これによりユーザが明示的に安全化のための処理を書かずとも隔離処理が行われる．図 10 にフローチャートを示す．

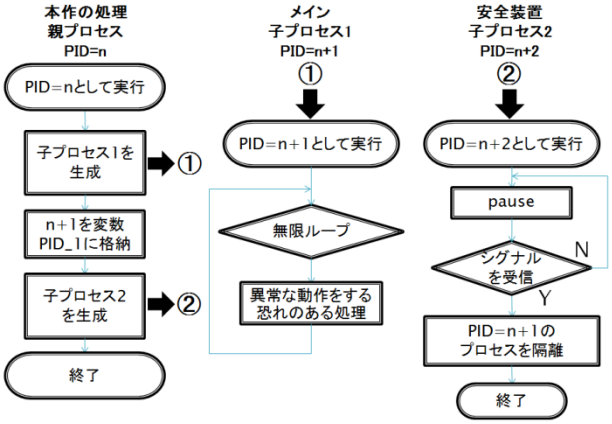

図 10　安全装置の処理フロー

Figure10　Process flow of the safety device

隔離後の動作として，コンテナを利用して CPU 使用率に制限をかけた空間を予め設けておく．安全装置に実装されているシグナルハンドラによって異常プロセスは制限のある空間に隔離されるものとした．その後のプロセスを回復させるまでの設計は検討中である．

6.　関連研究

ロボット制御システムにおいて，別々に動作するアプリケーション情報を統合するミドルウェアとして，ROS(Robot Operating System) [5]がある．

ROS は，ソケットによる TCP/IP 通信を行なうことで，複数のロボット間でのネットワーク通信を可能とする．しかし，ローカルでのモジュール間通信でもソケットによる TCP/IP 通信を行っている．よって，単体のロボット内のモジュールを ROS によって通信することは，通信速度に問題がある．そこで，IXM はローカルでの通信において，共有メモリを用い，通信速度の向上を図る．そして，その共有メモリを扱うための API を提供し，デバイス間の通信部分の実装を容易にする．

また，ROS は，ロボットの省電力化を考慮したミドルウェア設計ではない．ロボットは，通常のデバイスと異なり，多くは自律動作をするものであるため，IXM では，従来の CPU 管理による省電力化だけでなく，ロボットに向けた省電力化手法を提供する．IXM における省電力化手法として，現段階では，OS 制御とアプリケーション制御の二つを考慮している．OS 制御による省電力は，コアの管理を行うものであり，三つのモデルを用意し，CPU 使用率を用いた条件式によって状態遷移を行うものとなっている．スケジュー

ラの既存研究[11]では，統計情報に基づき性能予測を行うものがあり，そのような仕組みを導入することも必要であると考えられる．一方，アプリケーション制御による省電力は，ロボットにおいてモータなどのハードを動かすことが大きな電力消費となると考えられるので，アプリケーション制御を行うことでハードの動作頻度を制御するものである．センサからのデータの変化を見ることでハードの動作頻度を変更するものとなっている．

7. まとめ

見守りロボットにおけるデータ共有の問題と省電力化と安全性の問題を示し，それぞれに対する解決手法を提案した．データ共有においては，リアルタイム性考慮のための応答時間の実験を行った．最悪時間が最も短い共有メモリを用いてモニタリングとコンロトールのデータ共有を行うことで処理時間を削減する方法を提案した．しかし，平均時間と標準偏差では，システムコールが優れていたので，さらに評価回数やデータ量を増やした場合の実験を行う必要がある．省電力化においては，アプリケーションの CPU 利用率を監視し，それによりアプリケーションを実行する CPU を決定し，使わない CPU の電源を落とす方法と，アプリケーションの動作周期を変更する方法を提案した．しかし，アプリケーションの動作頻度を落とした場合，データに大きな変化があった時に反応が遅れてしまう問題も考えられる．安全性においては，あるプロセスに異常が起きた場合にシステム全体を保護するために，動作しているプロセスを監視する安全装置プロセスを用意し，異常を発見した場合には，その安全装置プロセスが異常プロセスの隔離を行う仕組みを提案した．隔離後から回復までの動作においては，これから検討する必要がある．

今後，これらの問題について調査を行い，提案した機能を既存の IXM に実装する．また，ROS で実装をした場合との比較も行うべきだと考えられる．

参考文献

1) 住谷 拓馬,菅谷 みどり:家庭用移動ロボットを用いた見守りシステムの実現，第 184 回 SE・第 33 回 EMB 合同研究発表会，茨城，2014． 5
2) 安藤 慶昭， 末廣 尚士， 北垣 高成， 神徳 徹雄， 尹 祐根:RT コンポーネントによるロボットシステム開発 -RT ミドルウェアの基本機能に関する研究開発(その 10)-， 計測自動制御学会 システムインテグレーション部門 講演会 2004 (SI2004)， pp. 264-265，つくば国際会議場， 茨城県，2004．12
3) OpenRTM-aist
http://www.penrtm.org/openrtm/ja/content/openrtm-aist-official-website
4) OpenRTM.NET
http://www.openrtm.org/openrtm/ja/project/NEDO_Intelligent_PRJ_ID150
5) Morgan Quigley, Brian Gerkey, Ken Conley, Josh Fausty, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, Andrew Ng:ROS:an open-source Robot Operating System, ICRA Workshop on Open Source Software, 2009
6) 住谷 拓馬,菅谷 みどり:マルチパーパスロボットを実現する機能統合システムの提案,第 76 回情報処理学会全国大会,東京,2014．3．
7) YUJIN ROBOT:Kobuki,
http://kobuki.yujinrobot.com/home-en
8) Microsoft:Kinect,
http://www.xbox.com/ja-JP/Kinect
9) HardKernel:ODROID
http://www.hardkernel.com/main/main.Php
10) 松原豊， 高田広章，〝組込みシステムにおけるソフトウェア障害の安全対策〟， 日 本ソフトウェア科学会学会誌 コンピュータソフトウェア， Vol．30， No．1， Feb 2013．
11) 金井 遵，佐々木 広，近藤 正章，中村 宏，並木 美太郎:統計情報に基づく省電力 Linux スケジューラ，情報処理学会研究報告．[システムソフトウェアとオペレーティング・システム]2007-OS-106，2007．8