ロボットにおけるタスクのリアルタイム処理を支援する RMA スケジューラの設計と実装

指導教員　菅谷みどり　　菊池　敬裕

１．課題と目的

近年，内閣府は新たな社会の形，Society5.0 を提案している[1]. Society5.0 の構成要素の中でも, ロボットはネットワークにつながり, 大量のデータをクラウドに集め, 人工知能を介し処理をし，フィードバックするものとして期待されている．これらのロボットは災害現場における人命救助などで緊急性が求められることから，データ収集，およびフィードバックの処理のリアルタイム性が求められる.

しかし，ロボットを動作させる際に一般的に利用されるミドルウェア ROS(Robot Operating System)はリアルタイム性を保証していない．本来であれば，リアルタイム性を保証するための仕組みとして，リアルタイム理論に基づくアプリケーションの設計，また，それを支援するための OS の機能が必要となるが，その検討が十分でない．そこで予備実験では，リアルタイム理論として有効とされる RMA(Rate Monotonic Analysis)に基づき, Linux 上にロボットのタスクを想定したタスクセットを設計，実装，評価を行った．その結果から，Linux のスケジューラの API を用いることで, リアルタイム処理の実現が可能であることが分かった．これは LinuxOS に RMA 支援機構があるために可能であった.

しかし，ロボットに用いられる計算機は多種多様であり, 常にリアルタイム処理を実現するための API が提供され，OS 上に支援機構があるとは限らない．そこで，本研究では，OS にリアルタイム支援機構がない Haribote OS を用いて RMA スケジューラの設計，実装を行い，性能および実現方法を議論することを目的とした.

２．Linux におけるリアルタイム処理支援

２．１　概要

本研究では，まず, ROS などで広く利用される OS である Linux のリアルタイム処理の実現が可能であるか調査を行った．リアルタイム処理を支援する理論として RMA を用いる.

以下，RMA 理論および予備実験を行った結果について述べる.

２．２　Rate Monotonic Analysis

Rate Monotonic Analysis とはリアルタイム性を保証する理論である[2]. 以下の式を満たすことでリアルタイム性が保証される.

$$\sum_{i=1}^{m} \mathrm{Ci/Ti} \leqq \mathrm{m}(2^{1/m} - 1) \qquad (式1)$$

ここで，m はタスク数，Ci は最悪実行時間，i はタスクの周期である.

２．３　予備実験

RMA を 3 つのプログラムに適用し，リアルタイム性の保証を理論と実地調査の両者で行った.

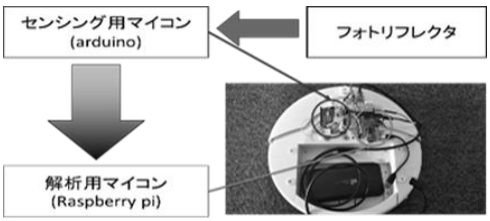


図１　動作環境

開発した自作ロータリーエンコーダーを搭載した iRobot Create を Raspberry Pi に繋ぎ，Raspberry Pi 上で実行する(図 1)．タスクは (1)フォトリフレクタを利用し，移動距離を算出(Task1)，(2)バッテリー残量確認 (Task2)，(3)計算 (Task3)の 3 つである．RMA を利用する上で，タスクの最悪実行時間を特定する必要があるので測定ベース解析により特定した．周期はセンサのサンプリングレートから決定した．これらタスクセットは表 1 に示す．また，3 つのタスクを RMA に則り処理するために制御プログラムを作成した．制御プログラムでは周期が短いタスクを高優先度に処理するために、システムコールの set_scheduler()を用いた.

表１　タスクセット

	Task1	Task2	Task3
最悪実行時間　C(ms)	1.7	0.9	1.4
周期　T(ms)	4.0	5.0	10.0

２．４　結果

3 つのプログラムの最悪実行時間, 周期を RMA (式 1) を用いて評価した結果，左辺値＜右辺値となり，リアルタイム性の理論上の保証が確認できた(式 2).

$$\sum_{i=1}^{3} \mathrm{Ci/Ti} = \frac{0.9}{5.0} + \frac{1.7}{4.0} + \frac{1.4}{10} = 0.754 < 3(2^{1/3} - 1) = 0.78 \quad (式2)$$

CPU 使用率と(式 2)の計算結果の左辺より，実行環境と理論値が酷似していた．これは実際の処理が理論通りであり，リアルタイム処理の実現が可能であることを示す.

２．５　課題

Linux 上で RMA を API を用いて実現できた理由として，Linux がリアルタイム処理を支援するための OS 機構である静的優先度スケジューラが存在することが挙げられる．しかし，ロボットを実現する計算機や OS は多種多様で，OS がリアルタイム処理を支援する機構を持つとは限らない．

３．リアルタイム処理を支援する RMA スケジューラの設計と実装

３．１　概要

課題の解決のために RMA の支援機構がない Haribote OS に RMA スケジューラの設計，実装，評価を行い，それをもとにロボット OS における RMA 支援について議論するものとした．

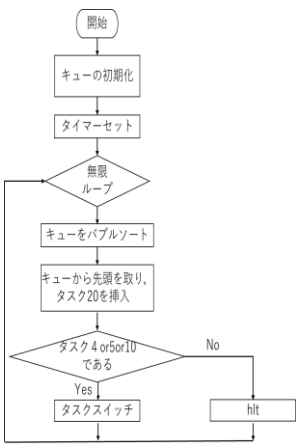
３．２　Haribote OS とそのスケジューラ

Haribote OS は CPU386 と 486 以降の命令列に対応した OS であり，32bitOS となっている[3]．メモリをセグメンテーション分ける機能，fifo バッファ，タイマ，割り込み，描画の機能を持っている．

Haribote OS のスケジューラはタスクをタイムスライスに実行するものであり，リアルタイム性の保証はない．

３．３　RMA スケジューラの実装

RMA の実装には，(1)タスクの属性(周期，実行時間，実行フラグ)の管理，(2)周期実行するタスクをためるキュー，(3)キューにためられたタスクを周期でソートする機能，が必要である．そのため，(1)はタスクの属性を管理する構造体を用意した．(2)はタスクをためる配列を用意し，タイマを用い一定周期で配列にタスクをためることとした．(3)はキューにタスクを入れた直後に行い，入れられたタスクの周期を参照してバブルソートし，仮に同一周期のタスクがあれば，それはデッドラインをミスしたとして削除することにした(図 2)．また，割り込みの処理は実行フラグを確認し，実行中のタスクより割り込みされるタスクの周期が短ければ実行し，長ければキューに入れるものとした(図 3)．



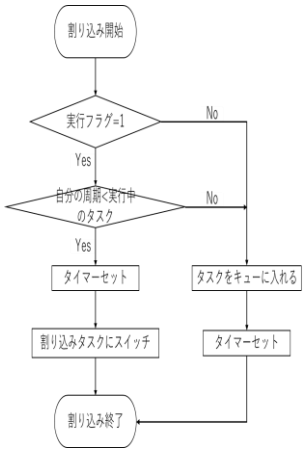


図 2 RMA スケジューラ処理　　図 3 割り込み処理

４．評価

４．１　内容

評価にあたり，3 つのタスクセットを用意した．このタスクセットは予備実験と同じものである(表 1)．

このタスクセットをそれぞれ 100 回実行した際のデッドラインミス回数，ジッタを評価の対象とする．

４．２　結果と考察

用意されたタスクセットを RMA の理論式上にあてはめると式は満たされ，リアルタイム性は保証される．従って，デッドラインミスはなく，ジッタも(表 2)の上部のようになると分かった．しかし，RMA の理論を満たすため Haribote OS 上に実装したスケジューラではジッタ，デッドラインミス回数は(表 2)の下部のようになり，ともに理論値とは異なる値となった．これには 2 つの理由が考えられる．1 つがメインのフローチャートにおいて，タスクが起動してから実際にタスクにスイッチするまでの処理量である．2 つ目が割り込みの処理量である．優先度の高いタスクの処理中においても割り込みが起きるため，優先度の高いタスクの処理の阻害の原因になる．

表 2　デッドラインミス回数とジッタ

		T1	T2	T3
理論値	ジッタ(ms)	0	1.7	2.6
	デッドラインミス回数	0	0	0
HariboteOS	ジッタ(ms)	0.82	2.27	4.84
	デッドラインミス回数	0	11	32

５．まとめと今後の課題

本研究では，RMA を支援する機構が備わっていない Haribote OS に RMA 理論を満たす機構を実装し，その評価を行った．結果，周期が短く優先度の高いタスクほどジッタ，デッドラインミス回数が少なく，優先度の低いタスクほどリアルタイム性の保証がなされなかった．

RMA を支援する機構としてタイマ，タスクの属性管理，タスクを入れるキュー，キューのソート機能，割り込み，が必要であった．しかし，タイマをセットする，つまりタスクを起動してから，実際にタスクスイッチするまでのキューのソート，予測不可能な割り込みがボトルネックになった．今後はソートのタイミング，割り込み処理の軽量化をすることで RMA 支援機構の向上を図っていく．

参考文献

[1] “Society5.0- 科学技術政策”．内閣府．http://www8.cao.go.jp/cstp/society5_0/index.html,　(参照 2018-12-22).

[2] C．Liu and J．W．Layland: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, J．of ACM, Vol 20, No．1、pp．46-61、1973.

[3]川合秀実．30 日でできる!OS 自作入門．マイナビ出版．2018-4-13. P14-311.