オープンソースを活用した簡易遠隔操作システムの実現

磯崎裕臣　宮崎大輔 *　徳山雄斗 †　古瀬涼郁 †　林陸哉 †　西川裕貴 †
中村慶士 ‡　山口顕太郎 §　中山翼 †　黒木孝樹 †　前川孝司

Realization of simple remote control system based on open source technology

Hiroomi ISOZAKI　Daisuke MIYAZAKI*　Yuto TOKUYAMA†　Suzuka KOSE†
Rikuya HAYASHI†　Yuki NISHIKAWA†　Keishi NAKAMURA‡
Kentaro YAMAGUCHI§　Tsubasa NAKAYAMA†　Atsuki KUROKI†　Koji MAEKAWA

Abstract

In recent years, the robot technology began to be utilized for various things. They are composed of high level and complex programs and expensive robots. In particular, the remote controlled robots are active in important matters such as disaster areas and space development. However, these technologies require advanced robots and specialized technician.

Therefore in this research, we realize inexpensive and simple remote control system by using open source technology.

Key words: Open source, Remote Control, Raspbery Pi

1　まえがき

近年、ロボット産業が成長している。多くのロボットは主に人の手で作業できないものや作業効率を高めるために開発されている。特に遠隔操作ロボットは人の手で操作でき、人だけではできない作業ができるため、重要な作業で活躍している。例えば宇宙開発や被災地での復興のロボットなどがある。しかし、これらを構成するには専用のコントローラー、高度な制御装置、高価なロボット、そしてそれを操作する訓練を積んだ操縦者が必要となる。また、開発や導入にもコストがかかるため、遠隔操作ロボットを利用するのは容易ではない。その一方で、介護や医療といった身近な場面でもロボットが注目され、活躍の場を広げている。

そこで、本研究ではオープンソースを活用し安価で簡易的な遠隔操作システムを構築し、専用のロボットではなく、身近にあるものを用いて容易に遠隔操作可能にするシステムの実現を目標とする。まず、遠隔操作システムの実現に必要となる機器や技術について検討を行った。検討結果をもとに特殊な機器や専用の機器を必要としない遠隔操作システムを構築した。実際に構築したシステムを運用することで、簡易的な遠隔操作システムが実現可能であることを確認した。

以下 2 章で遠隔操作システムの構成を説明する。次の 3 章でシステムの構成に必要となる技術や機器について検討する。4 章で提案する遠隔操作システムを述べ、5 章で実際に遠隔操作システムを構築し、動作検証を行う。最後に 6 章で今後の課題などについてまとめる。

2　遠隔操作システム

一般的な遠隔操作システムは、図 1 に示すような構成になっている。コントローラからの操作命令はインターネット等の通信路を経由して、ロボットの制御装置に送られる。また、遠隔操作システムでは、低遅延でリアルタイム性の高いシステムや通信路が必要とされる。そのため 1 章で述べたとおり、災害現場等で用いられる遠隔操作システムは、そのシステム専用に開発されたコントローラや制御装置、ロボットで構成されていることが多い。

そこで本研究では、安価な小型のマイコンとオープンソースソフトウェアを活用し、専用の機器や通信路を必要とせず、普段利用しているような通信環境においても低遅延でリアルタム性を有する簡易遠隔操作システムを実現する。

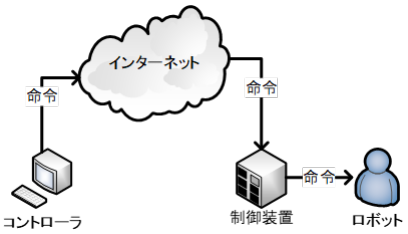

図 1: システムの構成

3　システムの検討

本章では、2 章で述べた遠隔操作システムの構成要素のうち、コントローラと制御装置について検討を行う。また、低遅延でリアルタイム性を有するシステム構築のために、制御プログラムの開発言語と映像伝送技術についての検討も行う。

3.1　コントローラの検討

コントローラには、ノートパソコン等の持ち運びが容易な小型のコンピュータが用いられることが一般的である。しかしこれらのコンピュータでは、キーボードやマウスでの操作が中心となり、直感的な操作が難しくなる。またコントローラは、基本的に制御装置に操作命令を送信するだけ良いので、高い処理能力は必要とされない。

このため本研究では、直感的に操作でき、身近に存在するスマートフォンやタブレット端末に着目し、中でも比較的画面サイズが大きく持ち運びも容易であるタブレット端末を用いる。

*株式会社 NTT フィールドテクノ
†電気電子工学科 5 年
‡雪印メグミルク株式会社
§福岡市市役所

3.2 制御装置の検討

コントローラと同様に処理装置も、ノートパソコンや小型のコンピュータを用いることが考えられる。しかし、一般的なコンピュータでは、開発するロボットのサイズを考慮したコンピュータの選定が必要となり、そのようなコンピュータは、比較的高価となる傾向がある。

このため本研究では、比較的に安価に入手可能なマイコンボードを用いる。小型のマイコンボードとして代表的なものArduino [1] や Raspberry Pi [2] と呼ばれるものがある。ここで Arduino と Raspberry Pi の特徴を表 1 に示す。表より、Arduino は、GPIO 端子を用いた制御やセンサー等からの信号の入力の機能が充実しており、容易に実現できる。一方、ネットワークやカメラ等の外部機器との接続は、容易でない。また、プログラムを内蔵のメモリに記録するため、内蔵メモリの容量以下のプログラムしか作成できず、複雑なプログラムを作成することができない。一方、Raspberry Pi は OS を搭載しているため、ネットワークやカメラ等の外部機器との接続が比較的容易である。また、システムやプログラムは SD カードに記録されるため、容量制限を受けることなく、複雑なプログラムを作成することが可能である。しかし、GPIO 端子を用いた制御やセンサー等からの信号の入力の機能は、Arduino ほど充実していない。

表 1: Arduino と Raspberry Pi の比較

	GPIO 機能	プログラム可能サイズ	ネットワーク接続	外部機器接続
Arduino	○	△	△	△
Raspberry Pi	△	○	○	○

すでに述べたように Arduino と Raspberry Pi は、一長一短であることがわかる。本研究では、ネットワークや外部機器が容易に接続でき、プログラム可能なサイズに制限を受けないという理由で Raspberry Pi を用いる。

3.3 システムの開発言語

現在 Raspberry Pi では、プログラミング言語として Ruby や Python が一般的に利用されている。プログラムの実行速度は、プログラミング言語によって異なり、実行速度が速いプログラミング言語を用いることでコントローラからの操作命令を短時間で処理でき、リアルタイム性を向上させることができる。そのため、プログラミング言語の実行速度を検証する。

一般的に C 言語やアセンブラ言語は、実行速度が高速であるとされているが、文字列処理が不得意であるため、WEB ベースのアプリケーション開発ではあまり利用されていない。本研究でも操作命令として文字情報を送受信することになるため、C 言語やアセンブラ言語は、利用しない。また、WEB ベースのアプリケーション開発で用いられ、文字列処理の機能が非常に充実している言語に Perl と呼ばれる言語が存在する。

ここでは、実行速度を比較するために Perl, Ruby, Python で 3000 個の乱数をバブルソートで並べ替えを行うのに必要な時間を計測した。計測は、Raspberry Pi 3 Model B で行った。表 2 にそれぞれの言語で 10 回ずつ計測を行い、その平均時間を示している。表より、Ruby が最も早く、Python が最も遅いことが確認できた。また [3] では、ベンチマークを行い、Perl が最も高速で、Python が最も遅いことが述べられている。ベンチマークの結果と今回の計測結果で実行速度の順位が異なるが、行わせた処理の内容が影響していると考えられる。このため、本研究では、文字列処理が得意で軽量である Perl を開発言語として用いる。

表 2: バブルソートに要する時間

	Perl	Ruby	Python
平均時間 [msec]	8.486	6.474	15.336

3.4 動画配信システムの検討

遠隔操作システムでは、ロボットに取り付けられたカメラ映像を見ながら様々な操作を行うことになるため、以下の 3 つの動画配信方法について検証を行う。

- VLC [4] を用いたストリーミング配信
- Gstreamer [5] を用いたストリーミング配信
- MJPG-Streamer [6] を用いたストリーミング配信

検証は、図 2 に示すように 無線で LAN に接続した Raspberry Pi で動画のストリーミング配信を行い、有線で LAN に接続した計測用 PC で視聴した動画の遅延を確認するという方法で行った。また、計測用 PC と無線 LAN アクセスポイントは、1 Gbps で同じスイッチに接続されているため、 計測用 PC と無線 LAN アクセスポイント間のネットワーク遅延はほとんど存在しない。

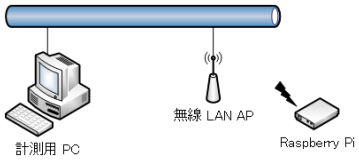

図 2: 計測環境

3.4.1 VLC によるストリーミング配信

まず、ストリーミングを行う動画形式として高画質で圧縮率が高い H.264 を用いることを考えた。この動画形式で VLC を用いたストリーミング配信を行ったところ、4 秒程度の遅延が確認できた。そのため、ストリーミング配信時のフレームレートや解像度といったオプションを調整することで、4 秒程度あった遅延を 2、3 秒に短縮可能であることが確認できた。

オプションの調整で短縮できなかった 2, 3 秒の遅延は、VLC が H.264 の動画形式の生成にソフトウェアエンコーダを利用していることが要因であると考えらえる。

3.4.2 Gstreamer によるストリーミング

Raspberry Pi には、H.264 対応のハードウェアエンコーダが搭載されており、このハードウェアエンコーダを利用可能なものに Gstreamer が存在する。VLC と同様に Gstreamer を用いてストリーミング配信を行ったところ、ほとんど遅延無く配信可能であることが確認された。しかし、Gstreamer によるストリーミング配信は、配信側と視聴側で Gstreamer を用い

る必要がある。Linux や Windows 等の OS では、Gstreamer をアプリケーションに組み込みことは、難しくないが、iOS や Android 等のモバイル OS では、セキュリティ確保のための制限の影響で組み込みには、高度な知識を必要とする。

このため、Gstreamer を用いて Gstreamer 以外のアプリケーションで視聴可能にするために、Gstreamer で生成したストリーミングを Nginx [7] を用いて再配信するという方法を用いた。この結果、15 秒程度の遅延が生じた。これは、Nginx による再配信処理にとても長くの時間がかかっていると考えられる。

したがって、Gstreamer 以外のアプリケーションで視聴可能な H.264 形式で配信するには、Raspberry Pi 単体では負荷が多すぎで不可能であることがわかった。

3.4.3 MJPG-Streamer によるストリーミング

VLC や Gstreamer のように H.264 といった動画形式を用いず、単純に JPEG 画像をパラパラ漫画の要領で送信することで低負荷で動画配信可能なものに MJPG-Streamer がある。この MJPG-Streamer を用いて配信を行ったところ、Gstreamer 同士で行った場合と同じようにほとんど遅延することなく配信することができた。この MJPG-Streamer によるストリーミングの視聴には、特別なアプリケーションが不要で画像が閲覧可能な Web ブラウザであれば良い。このため、動画視聴機能をアプリケーションに組み込むことが容易になる。

したがって、本研究では、動画配信システムとして MJPG-Streamer を用いる。

4 提案する遠隔操作システム

3 章での検討結果にもとづき、遠隔操作システムを提案する。システムを構成する要素として

- コントローラ ··· Android タブレット端末
- 制御装置 ··· Raspberry Pi 3 Model B
- システムの開発言語 ··· Perl
- 映像伝送技術 ··· MJPG-Streamer

を用いる。ここでは、遠隔操作の対象を身近なものとしてプラレールの車両を用いる。複雑な機構によって多彩な操作ができるラジコンカーやロボットと異なり、プラレールの車両は、簡単な機構のみで走るか止まるかといった簡単な動作しか行えない。このため、遠隔操作の対象の機構が遠隔操作システム全体に与える影響を最小限に抑えることができる。またコントローラについては、開発環境を比較的に容易かつ安価に構築可能な Android 端末を用いる。次に提案システムの構成を図 3 に示す。図に示すようにコントローラである Android タブレット端末から制御装置である Raspberry Pi に操作命令が送られる。操作命令は Raspberry Pi で処理され、制御信号が制御回路に出力される。また、Raspberry Pi からは、カメラの映像がタブレット端末に送られるようになっている。

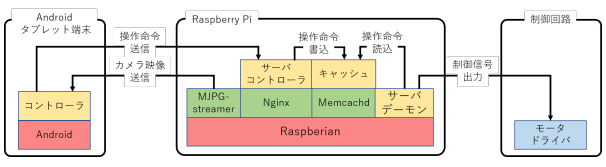

図 3: プログラムの構成

4.1 制御回路

まず、プラレール車両のモータを制御する回路を図 4 に示す。基本的な回路は、Raspberry Pi からモータドライバに PWM 信号を入力することでモータの回転方向や速度制御している。モータドライバは、大きな電流を必要とする高トルクのモータに対応できるように TOSHIBA TA-8428K を用いている。また、このモータドライバは、参照電源端子が存在しないため、PWM 信号が出力される Raspberry Pi の GPIO 18 と正転反転信号が出力される GPIO 23 と GPIO 24 の論理積をモータドライバに入力することで、モータドライバの IN1 と IN 2 に PWM 信号が入力されるようにしている。GPIO の出力とモータドライバの入出力を表 3 に示す。

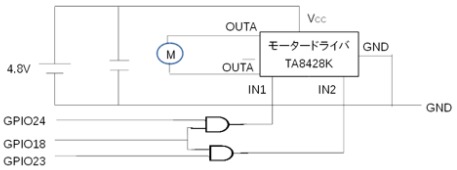

図 4: 回路図

表 3: Arduino と Raspberry Pi の比較

GPIO 18	GPIO 23	GPIO 24	IN 1	IN2	OUT A	OUT $\overline{A}$	
PWM	HIGH	LOW	PWM	LOW	PWM	LOW	前進
PWM	LOW	HIGH	LOW	PWM	LOW	PWM	後進
PWM	HIGH	HIGH	PWM	PWM	PWM	PWM	ブレーキ
PWM	LOW	LOW	LOW	LOW	∞	∞	ストップ

4.2 制御プログラム

次に制御プログラムについて述べる。制御プログラムは、コントローラからの操作命令を受信するサーバコントローラと GPIO を用いて制御回路に制御しているサーバデーモンの 2 つで構成される。2 つのプログラムは、図 3 に示すようにキャッシュを介して制御情報のやりとりを行っている。

サーバコントローラは、Nginx 等の軽量な WEB サーバ上で動作するプログラムであり、コントローラから送信された制御命令から制御回路の制御に必要となる情報を生成し、キャッシュに記録する。サーバデーモンは、コンソールで実行されるプログラムであり、基本的にバックグランドのサービスとして動作する。このプログラムは、100 ミリ秒程度の間隔でキャッシュの情報を確認し、必要があれば、制御回路に出力する信号を変化させる。

一般的に WEB サーバ上のアプリケーションは、リクエスト毎にスレッドやサブプロセスでプログラムが実行される。このため、今回のシステムでは操作命令が送られる毎にプログラムが実行されることになる。このとき、コントローラからの操作命令の処理や制御回路の制御を 1 つのプログラムで実現した場合、プログラムが並列的に動作する期間が長くなり、システムに負荷が生じ、リアルタイム性が低下することになる。

このため、すでに述べたようにプログラムを 2 つに分け、WEB サーバ上のプログラムの処理を少なくし、短時間で終了するようにすることで、システムへの負荷を軽減し、リアルタイム性を向上させている。また、キャッシュを介することで、各プログラムで保持する情報を最小限にし、システムの軽量化を行っている。キャッシュには、軽量で高速な Memcached [8] を用いている。

4.3 コントローラプログラム

最後にコントローラプログラムについて述べる。プログラムの操作画面を図 5 に示す。図にあるように操作画面の背景全体にプラレールの車両に搭載されたカメラの映像が表示される。右下と左下のバーを上下することで、アクセルとブレーキの操作が可能で、車両の速度が変更できる。左上のボタンをクリックすることで、車両の進行方向を切り替えることができる。また、中央下のボタンをクリックすることで、車両の状態に関係なく即座に停止させることができる。右上のボタンは、オプションで、今回は車両前方に取り付けられている照明が点灯に利用している。

図 5: コントローラプログラムの操作画面

5 システムの動作検証

本章では、4 章で提案した簡易遠隔操作システムを構築し、動作確認を行う。

遠隔操作システムを組み込んだプラレール車両とコントローラの Android タブレット端末を無線で学内ネットワークに接続し、動作確認を行った。

図 6 に動作の様子を示す。図のようにコントローラであるタブレット端末にカメラの映像が表示され、タブレット端末の操作に応じて車両の速度や進行方向が操作できることが確認できた。しかし、カメラの映像や車両の動きに若干の遅延が確認された。これは、学内ネットワークに無線で接続したことによるネットワーク遅延の影響であると考えらえる。このため、無線環境の改善や低遅延の通信プロトコルを用いることで、改善可能であると考えられる。

6 おわりに

本研究では、安価なマイコンボードとオープンソースアプリケーションを用いた簡易的な遠隔操作システムを実現した。実現に必要となる技術や仕組みについて様々な実現方法を検討し、低遅延でリアルタイム性を確保可能な実現方法の組み合わせを明らかにした。

検討結果に基づいて、コントローラからの操作命令に対して応答速度が速い制御プログラムを実装した。最後に提案したシステムの動作検証を行うため、機構が簡単であるプラレール車両に提案システムを組みんだ。動作検証の結果、ネットワーク上のトラヒックの影響による遅延がわずかに確認されたものの

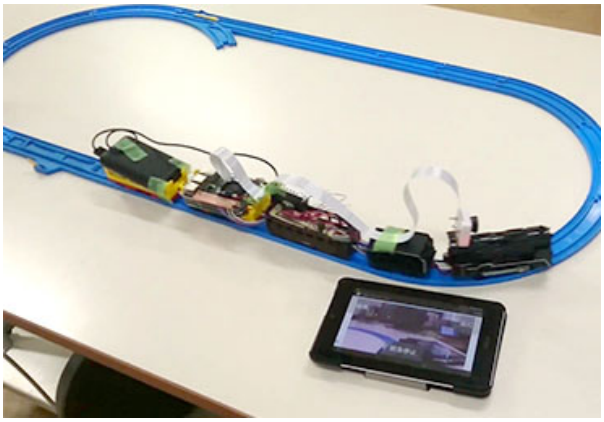

図 6: 動作中のシステム

のコントローラからの操作に従って、車両の速度や進行方向が変化することが確認できた。また、車両に搭載されたカメラの映像もネットワークの遅延の影響はあるものの、大きな遅延はなく、伝送できていることが確認できた。

今後は、現状のリアルタイム性を維持したまま、より複雑な機構を制御するシステムの検討、ネットワーク上のトラヒックによる遅延を低減し、これまで以上のリアルタイム性を実現する方法の検討を進める必要がある。

参考文献

[1] Arduino AG, “Arduino.” https://www.arduino.cc/, 2005.

[2] Raspberry Pi Foundation, “Raspberry pi.” https://www.raspberrypi.org/, February 2012.

[3] G. Isaac, “The computer language benchmarks game. web.” http://benchmarksgame.alioth.debian.org/.

[4] VideoLan Project, “VLC.” https://www.videolan.org/vlc/, February 2001.

[5] E. Walthinsen, “GSreamer.” https://gstreamer.freedesktop.org/, October 1999.

[6] A. Redmer and T. Stoeveken, “MJPG-Streamer.” https://sourceforge.net/projects/mjpg-streamer/, 2007.

[7] I. Sysoev, “Nginx.” https://nginx.org/, October 2004.

[8] Danga Interactive, “Memcached.” https://memcached.org/, May 2003.

(2017 年 11 月 6 日　受理)