ラズベリーパイマウスの走行プログラムの改良とチャンピオンシップにおける活用

Improvement of program to drive Raspberry Pi Mouse and application in “Championship”

坂下 達哉，森 文彦，相馬 正宜，岡田 浩之

Tatsuya Sakashita, Fumihiko Mori, Masaki Sohma, and Hiroyuki Okada

玉川大学工学部情報通信工学科，194–8610 東京都町田市玉川学園 6–1–1
Department of Information & Communication Technology, College of Engineering, Tamagawa University,
6–1–1 Tamagawa-gakuen, Machida-shi, Tokyo 194–8610

Abstract

The Department of Information & Communication Technology has several lectures of IoT focused on programming. To motivate to learn programming, first-year students enjoy manipulation of simple robots named “RaspberryPi Mouse” in the lecture “championship” last year. Concretely, students played race game to try to shorten driving time by adjusting parameters of given program to run this robot. In this year, the present authors rewrote the program to improve driving speed and operability dramatically, and employed it in this lecture.

Keywords: Raspberry Pi Mouse, IoT education, first-year experience, Championship

1 はじめに

情報通信工学科では，２年次以降，プログラミングに重点を置いた IoT の講義と実験が開講される．その動機づけのため，１年次の「チャンピオンシップ」という科目の後半では，ラズベリーパイマウス（以下，ラズパイマウス）というロボットの操作を体験させている．昨年度は，学生が走行プログラムのパラメータを試行錯誤的に調節して，走行時間を競わせる大会を実施した [1,2]．

今年度も昨年度に引き続き，同様の大会を実施した [3]．その際，ラズパイマウスの走行プログラムを作り直し，走行速度と操作性を向上させた．本稿では，プログラムの改良方法と PC 環境の準備といった技術面について報告する．

2 走行プログラムの改良点

昨年度のチャンピオンシップでは，デバイスドライバ [4] を使用して bash スクリプトで書かれたプログラム [5] を用いていた．このデバイスドライバは，ラズパイマウスの開発元により公開されている [6]．これを用いると，車輪のモータ，距離センサ，ブザーをキャラクタ型の入出力デバイスとして扱える．

しかしながら，以下の副節で詳しく述べるように，このデバイスドライバをそのまま使ったのでは不便な点が出てきた．そこで，我々は GPIO (General-Purpose Input/Output) を直接制御する方法 [7] で，C++ 言語により一からプログラムを書いた．この実装には，ライブラリ bcm2835 [8] を用いた．bcm2835 は静的ライブラリとしてリンクする事で，実行プログラムを自己完結させた．これにより「必要なライブラリやデバイスドライバが実行時に見つからない」といったエラーを防止できる．

2.1 モータの制御方法

ラズパイマウスの車輪には，パルス数で回転角を制御するステッピングモータが採用されている．このモータの制御にはハードウエアによる PWM (Pulse Width Modulation) が使われている．

これらのモータをデバイスドライバを用いて制御するには，２つのパラメータ（符号付きの整数と符号なし整数）を指定する [5,9]．前者は，モータを駆動したい周波数に回転方向を表す符号を付けて得られる整数である．後者は，PWM をオンにする時間である．後者の実装には，指定した時間が経過したら，PWM 信号をオフにするように割り込みが用いられている．

PWM をオン・オフする制御と回転方向の指示は，別々

の GPIO に割り当てられている．つまり，ハードウエア的には別物である概念を，ユーザが意識しなくても良いように，デバイスドライバの中で統合している訳である．

一方，本授業で扱う簡単な走行プログラムには，以下のような要求がある．

- モータの回転方向（つまり走行の向き）を変える事は少ない．
- 現状の PWM をオンにする時間を指定する方式では，PWM 信号が間欠的になる．この間欠性は，平均的な走行速度を遅くしてしまうだけでなく，ラズパイマウスに断続的な振動を加えてしまうため底面のネジが緩むというトラブルを起こしていた．この対策として，明示的に停止を指示しない限り，車輪を回転させる PWM 信号は出しっぱなしにすることが考えられる．

上記の要求を満たすには，モータの回転方向の制御と PWM 信号のオン・オフの制御を別々に行う必要がある．その制御には，デバイスドライバを用いることができないため，独自にライブラリを実装する必要がある．その際，PWM 信号を一定時間後に停止するための割り込みは不要であるから，その分だけオーバーヘッドも少なくなる．

2.2 距離センサからの読み取り

ラズパイマウスには，距離センサが 4 つ搭載されている．その位置と名前を図 1 に示す．センサーの値は，0〜3000 程度であり，壁に近いほど値が大きくなる点に注意を要する．

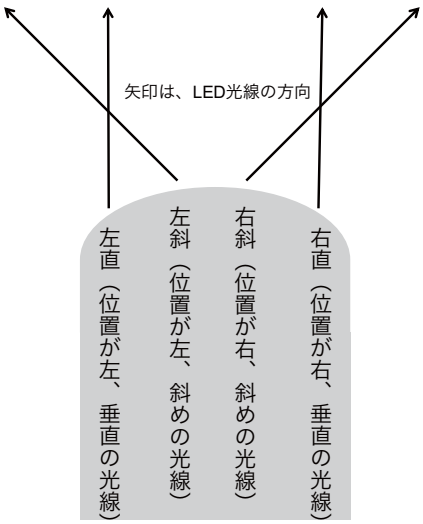

図 1　距離センサの位置と名前

この距離センサは赤外線 LED を用いているため，精度が悪い．そのため，3 回測って算術平均を取った値を使用することにした．

距離センサの読み取りには，SPI (Serial Peripheral Interface) [7] というシリアル通信が用いられる．この通信についても，デバイスドライバを介さず，直接 GPIO を制御して行うようにした．その理由は，計測は一度に 3 回行うため，計測に要するオーバーヘッドをできるだけ少なくするためである．

2.3 操作性の改良と安全面への配慮

1 年生は PC に不慣れであるため，操作法はできるだけ簡単化することが望ましい．

昨年度用いたプログラムでは，ラズパイマウス本体に取り付けられたタクトスイッチを押すと，走行が開始するようになっていた．しかし，触れた物が即座に動き出すのでは操作がしづらいと思われる．また，PC から遠隔操作している感覚を味わえるようにもしたい．

そこで，これを PC のキーボード操作に置き換えた．

- キー「y」が押下されると，走行開始
- キー「q」が押下されると，走行停止

このような操作は，Linux のシステムコールを用いて，キーボードを非カノニカルモードに変更すれば可能となる．これらのキーが押された直後に，以下のような初期化および終了処理を行う．

キー「y」が押下されると，以下の初期化処理が行われる．

1. モータの電源オン
2. ブザーを用いて，スタートシグナルを演奏
3. 走行の開始時間を記録（clock_gettime 関数を呼ぶ．）

キー「q」が押下されると，以下の終了処理が行われる．

1. 走行の終了時間を記録（clock_gettime 関数を呼ぶ．）
2. モータの PWM 信号をオフにする．
3. モータの電源オフ（火傷の防止のため，ここで行う．）
4. キーボードのモードを元に戻す．（これを忘れるとターミナルが乱れる．）
5. 走行時間の計算と画面表示

一般的に，プログラムの強制終了には CTRL+C の押下が用いられる．PC に親しんだ学生が咄嗟にこの方法を使うこともあり得る．そのため，CTRL+C が押下さ

れた場合もキー「q」が押された場合と同様の終了処理を行う必要がある．そのため，この終了処理を行う関数を，割り込みシグナル SIGINT を捕捉した場合のシグナルハンドラ（コールバック関数）として登録した．

3　PC からのログイン方法の変更

学生 PC からラズパイマウスへのログインには，リモートデスクトップを用いていた [1,2]．しかし，リモートデスクトップは GUI であるため通信量が大きいので，反応が遅いという欠点がある．また，サーバ側とクライアント側のキーボードの種類を同一にしなければならないといった制約がある．

そこで，今年度からは，SSH (Secure Shell) コマンドを用いることにした．SSH は，現在，Linux マシンへのリモートログインに使われる標準的な方法であり，Windows 10 でも Power Shell やコマンドプロンプトに標準で搭載されている．

SSH によるログインにはパスワード認証を用いた．公開鍵認証は便利で安全であるが，1 年生には敷居が高いためである．

学生の PC とラズパイマウスの間は，昨年度は LAN ケーブルによる有線接続を行っていたが，今年度は無線 LAN ルータを介した無線接続とした．その際，無線 LAN ルータの DHCP リースの機能を使用し，ラズパイマウスの IP アドレスを固定化した．

4　走行プログラムの実行方法

プログラムの実行例に即して，画面表示と操作方法の改良点について述べる．

4.1　走行プログラムの実行例

走行プログラムの実行時の画面出力の例を図 2 に示す．画面出力はすべて日本語にしてある．（Raspberry Pi のターミナルのロケールも日本語に設定した．）

図 2 では，以下のコマンドを実行している：

```
main 200 5
```

ここで，./main ではなく main でプログラムが実行できるのは，エイリアスを作成しておいたからである．カレントディレクトリを意味する./という特殊記号の入力方法を知らなくても使えるための工夫である．

コマンド名 main に続くコマンドラインパラメータの意味は，5 節で解説する．昨年度は入力ファイルによりパラメータを指定していたが [1,2]，ほとんどの学生はエディタの操作を体験したことがないため，エディタ操作が不要なコマンドラインパラメータを使用することにした訳である．

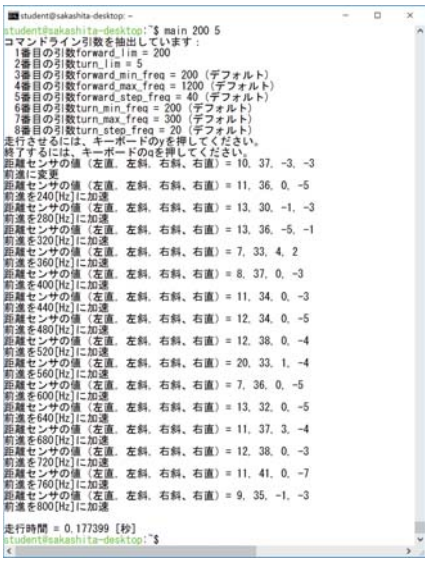

図 2　走行プログラムの画面表示

コマンドラインパラメータは全部で 8 個あるが，後半のものは入力を省略しデフォルト値を使用することができる．上記の例では，省略された 3 番目以降のパラメータはデフォルト値が使用される．

コマンドラインパラメータは，簡単のため数値だけを列挙する方式としたので，ユーザは順番を誤る恐れがある．意図した通りにパラメータが設定されたかを確認できるように，実行後にはまずパラメータの一覧が画面表示されている．その際，入力を省略したパラメータについては，デフォルト値が使われる旨，表示される．

図 2 の実行結果の最後では，キー「q」を押下したことにより走行が停止された後，走行時間が表示されている．

4.2　コマンド履歴による入力の省略

走行プログラムは，パラメータの一部のみを試行錯誤的に変更して繰り返し実行される．その度にコマンドを打つのは面倒である．そこで，ターミナルのコマンド履歴を活用すれば良い．（これは Linux のターミナルの操作法を修得させる良い機会となる．）

コマンド履歴の表示法について，以下のように説明を行った．

- カーソルキー「↑」を押すと，直前に入力したコマンドが表示される．
- もう一回，カーソルキー「↑」を押すと，2 個前のコマンドが表示される．

上記を用いて，プログラム実行を以下の手順で行える．

1. コマンド履歴を表示する．
2. カーソルキー「←」「→」を用いて，パラメータ（数値）を変えたい位置にカーソルを移動する．
3. 数値を修正する．
4. Enter キーを押すと，コマンドが実行される．

5 走行アルゴリズムとコマンドラインパラメータ

走行アルゴリズムと 4.1 節で触れた各種のコマンドラインパラメータについて述べる．

5.1 走行方向を決めるアルゴリズムとパラメータ

走行方向は，距離センサ（図 1）の値から図 3 に示すアルゴリズムに従って決められる．

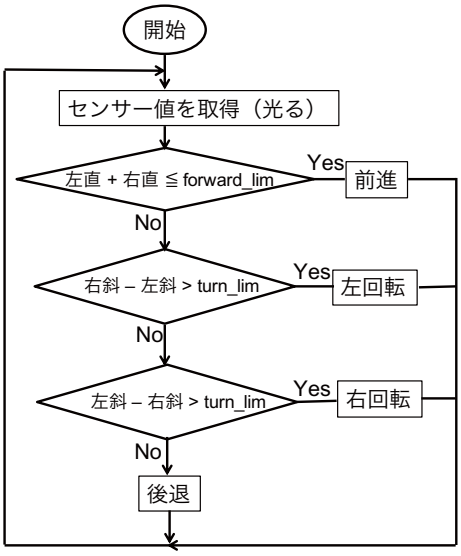

図 3　走行方向を決めるアルゴリズム

図 3 の条件式に現れる閾値パラメータの意味を表 1 に掲げる．

表 1　走行方向を決定するコマンドラインパラメータ

並び順	パラメータ名	デフォルト値
1	forward_lim	400
2	turn_lim	10

図 3 の前進，後退，左回転，右回転の条件式が適切である理由を，授業では図を用いて説明した．実際に手で覆うなどして壁を作り，センサの値がこれらの条件式を満たすことを確認するように学生に指導した．

5.2 モータの加速に関するパラメータ

車輪にはステッピングモータが使われているため，走行速度はモータに与えるパルスの周波数に比例する．つまり，速く移動させるには，この周波数を高くすれば良い．しかし，急に周波数を高くするとスリップの恐れがあるため，徐々に加速する．加速は，同じ方向の移動（後退を除く）が続いた時に行う．

前進の加速に関するコマンドラインパラメータを表 2 に示す．

表 2　前進の加速に関するコマンドラインパラメータ

並び順	パラメータ名	デフォルト値	意味
3	forward_min_freq	300	前進における周波数の初期値
4	forward_max_freq	1200	前進における周波数の上限値
5	forward_step_freq	40	前進における周波数の増分

例えば，以下のパラメータ

- forward_min_freq $= 200$
- forward_max_freq $= 1200$
- forward_step_freq $= 40$

を指定した場合，前進が続くと，モータの周波数は，$200, 240, \ldots, 1160, 1200, 1200, 1200, \ldots$ と変化する．

左・右回転の加速に関するコマンドラインパラメータを表 3 に示す．

表 3　左・右回転の加速に関するコマンドラインパラメータ

並び順	パラメータ名	デフォルト値	意味
6	turn_min_freq	300	回転における周波数の初期値
7	turn_max_freq	1200	回転における周波数の上限値
8	turn_step_freq	40	回転における周波数の増分

5.3 走行速度の求め方

車輪のステッピングモータは，1 つのパルスで 0.9 [度] だけ回転する [9]．パルス周波数が 400 [Hz] のとき，1 [秒] あたりの回転角は，上記より 0.9 [度]×400 [Hz] = 360 [度] である．つまり，このとき車輪は 1 秒間に 1 回転する．車輪の直径は 45 [mm] であるから，1 回転で進む距離は $45\pi \fallingdotseq 141$ [mm] である．

上記より，パルス周波数を f とすると，速さ $v = 45\pi \times f/400$ [mm/秒] という式が得られる．例えば，$f = 1200$ [Hz] の場合，$v \fallingdotseq 45\pi \times 1200/400 = 424$ [mm/秒] である．

以上は，パルスの意味を補足すれば，高校物理の知識でも理解可能である．

6 競技大会の結果

昨年度と同様に図 4 に示すコースを使用した．ゴールの位置は，壁から 8[cm] ほど手前とし，目印としてテープを貼った．その理由は，ゴールが壁に近すぎると，アルゴリズムが後退モードに切り替わったままとなり，ゴー

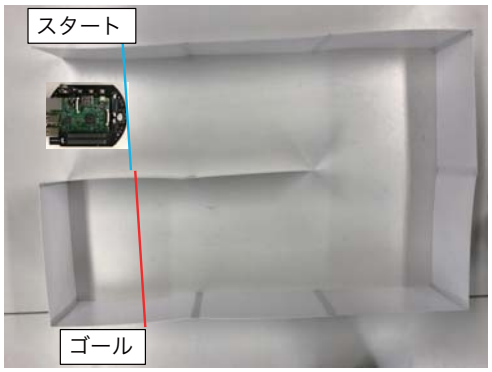

図 4　使用したコース

ルに到達できなくなるためである．

各班の最短時間を表 4 に示す．表 4 より，走行時間は

表 4　各班の最短の走行時間

班	走行時間 [秒]	順位
1 班	4.9167	
2 班	4.7468	
3 班	4.3	
4 班	4.7179	
5 班	3.1824	2
6 班	5.488	
7 班	4.76	
8 班	4.6282	
9 班	3.6293	5
10 班	3.4051	4
11 班	4.2903	
12 班	7.2643	
13 班	3.387	3
14 班	5.0072	
15 班	5.4844	
16 班	4.9339	
17 班	3.8735	
18 班	2.9955	1

約 3 ～ 5[秒] である．昨年度の走行時間は約 20 ～ 50[秒] であった [1]．したがって，およそ 1/10 程度の走行時間の短縮が実現できたといえる．これは，プログラムにモータを加速する機能を追加した事と，PWM 信号の連続化により途切れずに走行できるようになった事の効果である．

7　まとめ

ラズパイマウスの走行プログラムをハードウエアを直接制御する形で書き直し，走行の速度と安定性，操作性を向上させた．競技大会に用いた結果，走行時間は 1/10 程度にまで短縮された．

このように走行時間が短くなったため，次の課題として，コースの延長（迷路に置き換える等）が考えられる．だが，コースを複雑化すると，距離センサの性能が悪いため，以下の懸念がある：

- 壁にぶつからずに走行することが難しくなる．
- 走行時間は偶発的なノイズに左右され，パラメータの調節が意味をなさない．

この問題は，高性能な超音波センサや測域センサを用いた安価な車輪付きロボットが登場すれば解決に向かうだろう．

他の課題を以下に挙げる．

- わかりやすさの観点から加速は線型に行ったが，通例のように S 字加減速カーブを用いる．
- ゴールテープを踏んだかを自動検出できるようにしたい．そうすれば，ゴールテープの位置を任意に設定できる．
- プログラムを ROS (Robot Operating System) を使って整理する方法もある．ROS のオーバーヘッドが問題となるかを，現在のプログラムと比較して検討する．

参考文献

1) 森文彦，チャンピオンシップにおけるアクティブラーニングの実践，玉川大学工学部紀要，Vol. 53 (2018).
2) 森文彦，チャンピオンシップ（ラズベリーパイマウス）マニュアル，工学部 情報通信工学科 (2017).
3) 坂下達哉，チャンピオンシップ　－ラズパイマウスの解説，工学部 情報通信工学科 (2018).
4) 米田聡，Raspberry Pi で学ぶ ARM デバイスドライバープログラミング，ソシム (2014).
5) 上田隆一，「ラズパイマウス」を自在に動かそう，ラズパイマガジン 2016 年春号，pp. 112–145 (2016).
6) 株式会社アールティ，ラズベリーパイマウスのデバイスドライバ，`https://github.com/rt-net/RaspberryPiMouse` (2017).
7) 西永俊文，BareMetal で遊ぶ Raspberry Pi (2014).
8) Mike McCauley，C library for Broadcom BCM 2835，`http://www.airspayce.com/mikem/bcm2835/`.

9) 上田隆一, Raspberry Pi で学ぶ ROS ロボット入門, 日経 BP (2017).

2019 年 2 月 28 日原稿受付，2019 年 3 月 5 日採録決定
Received, February 28, 2019; accepted, March 5, 2019