Raspberry Pi を用いた
カラーライン追従型
サウンドロボットの作成

5 年　情報工学科　16 番　今野夏子

指導教員　東海林智也

目次

1 序論

1.1　目的

子ども達にモータやセンサーに興味を持ってもらえるようなライントレースロボットを作成する。

1.2　背景

私達の身の回りにはモータやセンサーなどが沢山あり、それらを普段の生活で何気なく使っている。例えば車や自動ドア、　掃除機や携帯電話などである。

しかし、それらを何気なく使っていても、モータがどのように動作するか、センサーがどんな時に反応するかなど実際どのような仕組みで働いているか考える人は少ない。そこで、それらについて考えてもらうにはまず子ども達にモータやセンサーに興味を持ってもらうことが必要だと考えた。

近年、人型ロボットやお掃除ロボットなど、ロボットは年々私達の生活に溶け込んでいる[1]。その中でもライントレースロボットは学校の授業でも、ものづくりの一環として取り扱われることが増えてきた[2]。元々、周回コースを出来るだけ早くトレースすることを競うライントレース競技などでも使用されており、子ども達にモータやセンサーに興味を持ってもらうことに適していると考えた。

従来のライントレースロボットはLEGOによるものが多く[3]、モータやセンサーが内部に搭載されていて、簡単なプログラムを記述するだけでライントレースが出来るようになるというものが多い。しかし、すべての機能が内部に搭載されていると、どのような動作をしているかが分かりづらいため、本研究ではすべての回路が外側から見えるようにした。

1.3　英文アブストラクト

Creating Color Line Trace Sound Robot

Abstract: We usually use a motor and a sensor casually. However, few people are thinking about those mechanisms. We that children might think about the structure of the motor and the sensor by watching working a robot. Therefore, in this study, we aim to make a robot by which children can be interested in the motor and the sensor. The robot detects the line on the floor with the light sensor called the photo-reflector, and performs line trace. Also, the robot distinguishes the color of the line by using the color sensor, and runs while outputting the different sound by color. We use Raspberry Pi to control the robot.

Key words: motor, sensor, photo-reflector, raspberry pi

2　ロボット仕様

2.1　概要

本研究で作成するロボットは、フォトリフレクタを使用して床の上の黒線を検知してライントレースを行う。さらに黒線と並列に色線も描き、ロボットが色線の上を走る際にはカラーセンサーを用いて線の色を識別し、色によって違う音を出しながら走る。用意する色は<赤・青・緑・黄・橙・紫・黄緑・桃>の 8 種類で、それぞれ<ドレミファソラシド>の 8 つの音に対応している。

これらの制御は Raspberry Pi を用いて行う[4]。ここで、図 2.1 にロボットの完成図、図 2.2 に動作図、図 2.3 に全体のブロック図を示す。

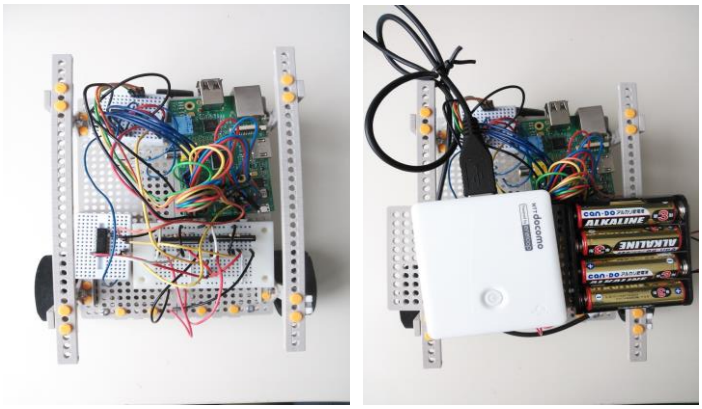


図 2.1　完成図

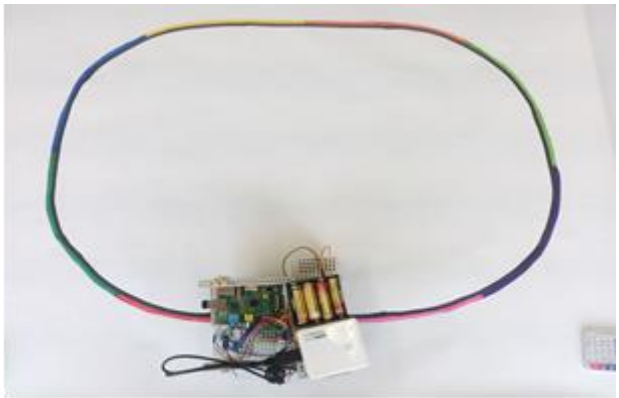


図 2.2　動作図

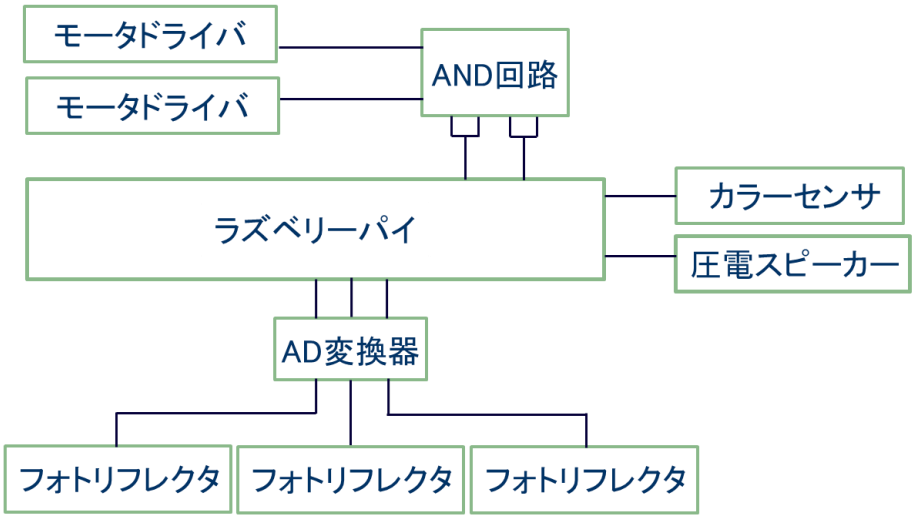


図 2.3　ブロック図

2.2　開発環境

開発 PC の OS：Linux

開発言語：C

制御用小型 PC：Raspberry Pi　Type B

ライブラリ：wiringPi

Raspberry Pi に PC からプログラムを送る方法：sftp

Raspberry Pi の OS：Raspbian

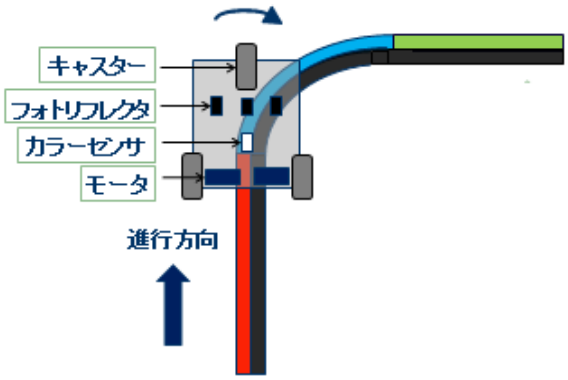
2.3　動作

作成するロボットには以下の 3 つの動作モードがある。

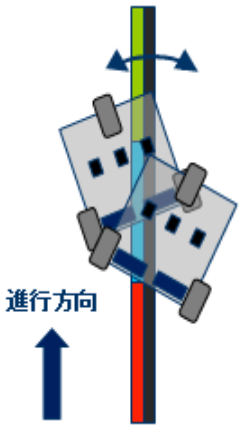
右折：3 つあるフォトリフレクタのうち一番右のフォトリフレクタが黒線の上に乗ったら右折する。・・・図 2.4（1）

左折：3 つあるフォトリフレクタのうち一番左のフォトリフレクタが黒線の上に乗ったら左折する。

直進：中央のフォトリフレクタが常に黒線の上にくるように左右にずれた軌道を戻しながら進む・・・図 2.4（2）



（1）右折



（2）直進

図 2.4　動作モード図

2.4　車体仕様

本研究では、TAMIYA の楽しい工作シリーズ[5]を使用して筐体を作成した。図 2.5 の部品を用いて、図 2.6 の筐体を作成した。筐体は 3 輪で、前輪のみキャスターを用いている。また、後輪タイヤには、スポンジ製のテープを貼っている。筐体の上側には Raspberry Pi、モータドライバ、AND 回路、A/D 変換器、圧電スピーカーの回路、裏側にはフォトリフレクタ、カラーセンサーの回路を設置している。

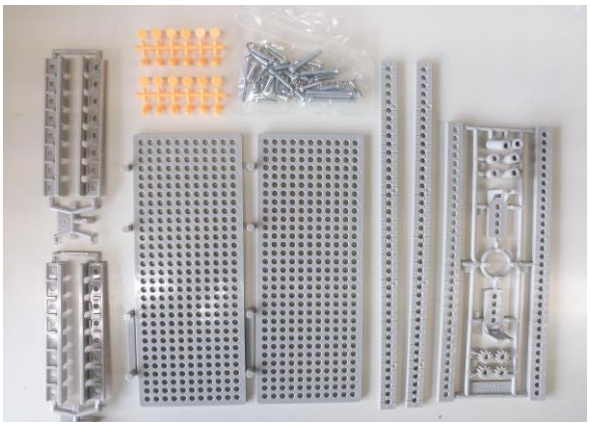


図 2.5　筐体の部品

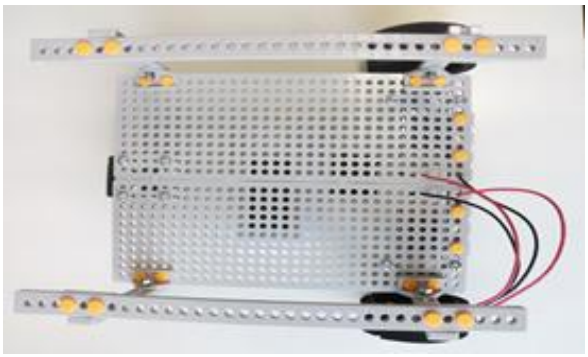


図 2.6　筐体

2.5　製作コスト

本研究で用いる機器や部品のコストを表 2.1 と 2.2 に示す。表 2.1 が筐体部分の製作コスト、表 2.2 が回路部分の製作コストである。この 2 つを合わせた総コストは、￥12112 となり、LEGO で作成するロボットよりもコストを抑えることが出来た。

表 2.1　筐体部分の製作コスト

機器名称	価格
TAMIYA ユニバーサルプレート	¥1,281
タイヤ・キャスター	¥216
TAMIYA ユニバーサルギヤーボックス	¥1424
計	¥2921

表 2.2　回路部分の製作コスト

機器名称	価格
Raspberry Pi （Type B）	¥5292
ブレッドボード（4 枚）	¥670
フォトリフレクタ（TPR-105F）	¥389
モータドライバ（TA3291P）	¥324
A/D 変換器（MCP3208）	¥324
カラーセンサー（S9706）	¥972
圧電スピーカー（バルク）	¥54
Bluetooth ドングル（BT-MicroEDR2X）	¥1166
計	¥9191

3.使用機器について

3.1　Raspberry Pi

Raspberry Pi とは、CPU・メモリ・USB 端子・LAN 端子・HDMI などが搭載されている超小型コンピュータのことである[4]。外観を図 3.1、スペックを表 3.1 に示す。

Raspberry Pi には GPIO と呼ばれる入出力端子があり、それぞれ 5V 端子、GND 端子、3.3V を出力する端子がある。これらのうち、プログラムで制御出来るのは 3.3V 出力端子の 8 つのみである。図 3.1 の右下にある GPIO ピンと回路とをつなぎ合わせ、プログラムを実行する。本研究では Raspberry Pi に、フォトリフレクタ（TPR-105F）と、モータの正回転・逆回転を制御するモータドライバ(TA3291P)、A/D 変換器(MCP3208)、AND 回路(74LS08N)、カラーセンサー（S9706)、圧電スピーカーを接続している[6][7][8][9][10]。なお本研究では、Raspberry Pi の電源にモバイルバッテリーを用いた。

本研究で Arduino[11]ではなく Raspberry Pi を用いた理由は、Arduino には OS がないため制御しにくく、Raspberry Pi と比べてメモリが少ないため複雑な処理がしにくいためである。よって今後の機能の拡張を考えて Arduino ではなく Raspberry Pi を使用した。

図 3.1　Raspberry Pi

表 3.1　Raspberry Pi のスペック

	モデルB
価格	35ドル
サイズ	8.6×5.4×1.7（cm）
メモリ	512（MB）
SoC	Broadcom BCM2835 （CPU, GPU, DSP, SDRAM内蔵）
CPU	700MHz
GPU	Broadcom VideoCore IV,OpenGL ES 2.0, 1080p 30fps H.264/ MPEG-4 AVC high-profile デコーダー
ネットワーク	10/100Mbps イーサネットRJ45（有線）
マルチタスク	可
電源	700mA（3.5W）
入力電圧（電源ソース）	5V
フラッシュメモリ	SDカード（2～16G）
USB	2口
OS	Linux、Debian、Fedora、Arch

3.2　モータドライバ

モータドライバとはモータを駆動・制御する回路のことで、今回使用した TA3291P の 10 個あるピンのうち IN1(5 番ピン)と IN2(6 番ピン)のピンを表 3.2 のような 0/1 の組み合わせで制御することにより、OUT1(2 番ピン)または OUT2(10 番ピン)に繋いだモータの回転方向を制御することが出来る。今回は左右のモータに 1 つずつ、計 2 つのモータドライバを使用した。

しかし、そのまま Raspberry Pi のピンを繋ぐと、モータドライバは 3.5V から 5.5V の入力電圧を必要とするため、Raspberry Pi の 3.3V 出力では入力電圧が足りずモータが動かない。そのため、本研究では AND 回路(74LS08N)を使用して 5V の出力を得た。AND 回路は、入力電圧が 2.0 以上だと HIGH と見なされ、電源電圧に近い値を出力する[12]ため、AND 回路の 2 つの入力に Raspberry Pi の 3.3V 出力ピン、出力にモータドライバの入力ピンを繋げると、モータドライバに 5V が入力され、正常に動作することが出来る。

表 3.2　真理値表

IN1	IN2	OUT1	OUT2	機能
0	0	∞	∞	ストップ
1	0	H	L	時計回り
0	1	L	H	反時計回り
1	1	L	L	ブレーキ

3.3　フォトリフレクタ

フォトリフレクタとは、赤外線 LED から放出された光を床の上の線に反射させ、反射光を検知することで出力電流が変化する光センサーのことである。

本研究ではフォトリフレクタ（TPR-105F）を筐体の背面中央に 1 つ、左右に 1 つずつで合計 3 つ設置している。色線では光の反射が色によって違うと考えたため、床面にカラーセンサー用の色線とフォトリフレクタ用の黒線の両方を描いた。また、周囲の光の影響を減らすため、LED をそれぞれのフォトリフレクタの横に設置している。

フォトリフレクタは白い線の上だと光が反射して電流が流れるが、黒い線の上だと光は反射せず電流はほとんど流れない(図 3.2)。そのため、プログラムでは黒い線の上にフォトリフレクタがあるとき進むというように設定してある。

しかし、フォトリフレクタの信号はアナログ信号なのでアナログ入力の無い Raspberry Pi にはそのままでは接続出来ない。そこで、A/D 変換器(MCP3208）を使用してアナログ信号をデジタル信号に変換した。

フォトリフレクタから読み取ったアナログ値を AD 変換器でデジタル値に変換し、プログラムで設定したしきい値によって今床の上にいるのか、線の上にいるのかを判断する。フォトリフレクタの値がしきい値よりも小さい場合、センサーは黒線の上にあるという判断となる。これを踏まえた状態遷移図を図 3.3 に示す。ここで、F_left は左のフォトリフレクタ、F_right は右のフォトリフレクタ、F_center は中央のフォトリフレクタ、stop_count は軌道から外れた時間をカウントする変数とする。F_left=0、F_right=0、F_center=0 は、それぞれのフォトリフレクタが黒線の上にあるということを示し、stop_count=5000 はカウントが 5000 回行われたということを示している。またライントレースを行う関数のソースコードを付録 1 に示す。

図 3.2　フォトリフレクタの動作説明図

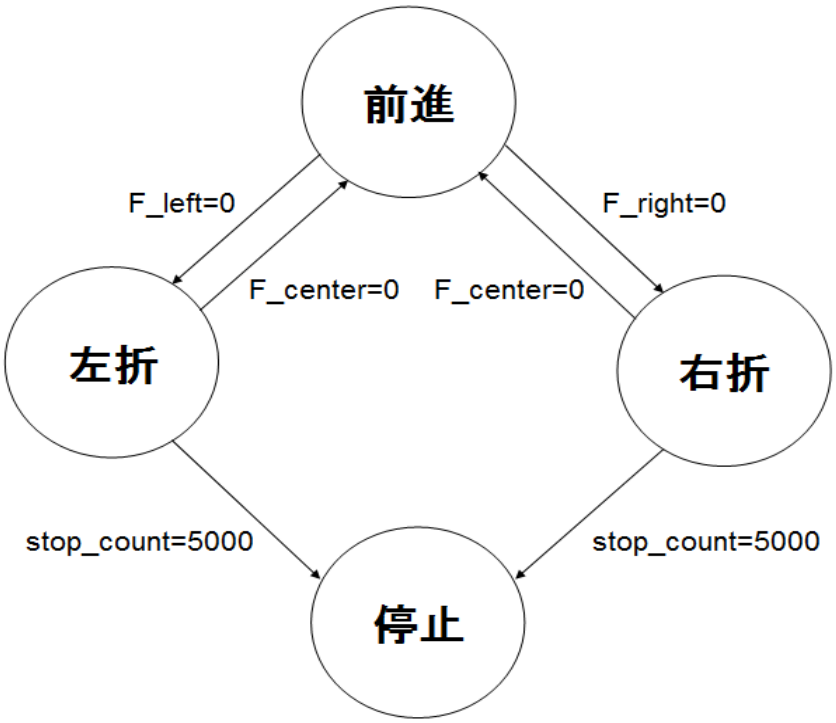


図 3.3　状態遷移図

3.4　カラーセンサー

本研究で使用したセンサー（S9706）は、RGB3 色の同時測光が可能なデジタルカラーセンサーである[10]。S9706 の特長、動作手順を以下に示す。

【特徴】

・12 ビットデジタル出力
・RGB3 色を同時測光
・2 段階の感度切り替え機能
・低電圧(3.3V)動作
・外付け部品が不要

【動作手順】

① Gat 端子と CK 端子を Low にする。
② Range 端子で、所望の感度を選択する。
③ Gate 端子を Low→High にして光量の積算を開始する。
④ 指定した積算時間の後に Gate 端子を High→Low に戻して光量の積算を終了する。
⑤ 測定データは、CK 端子に 36 個のパルスを入れることで、Dout 端子より出力される。

図 3.4 は、カラーセンサーのプログラムをコマンドプロンプトで実行し、RGB 値(10 進:2 進)を表示した結果例である。一番上を例にとると、23 が R の値、44 が G の値、60 が B の値となっている。この場合、B の値が最も大きいため色認識は「青」となる。この RGB の組み合わせで色の判別をプログラムによって行った。

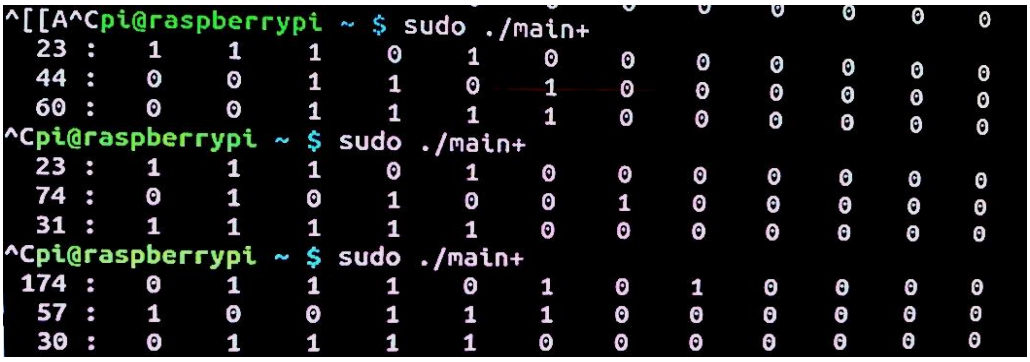


図 3.4　RGB 値の出力例

この結果を踏まえて、プログラムで色を識別するための基準をカラーコード[13]を基にして以下の様に設定した。この識別基準に従った色の識別結果例を図 3.5 に示す。

赤・・・RGB 値の中で R の値が最も大きく、黄、紫、橙、桃以外である場合
緑・・・RGB 値の中で G の値が最も大きく、黄、黄緑以外である場合
青・・・RGB 値の中で B の値が最も大きく、紫以外である場合
黄・・・RGB 値の中で R の値が最も大きく、R の値と G の値の差が 5 以内である場合
橙・・・RGB 値の中で R の値が最も大きく、G の値が R の値の半分よりも大きい場合
紫・・・RGB 値の中で R の値が最も大きく、R の値と B の値の差が 5 以内である場合
黄緑・・RGB 値の中で G の値が最も大きく、R の値が一定の値よりも大きい場合
桃・・・RGB 値の中で R の値が最も大きく、G の値、B の値も一定の値（30）よりも大きい場合

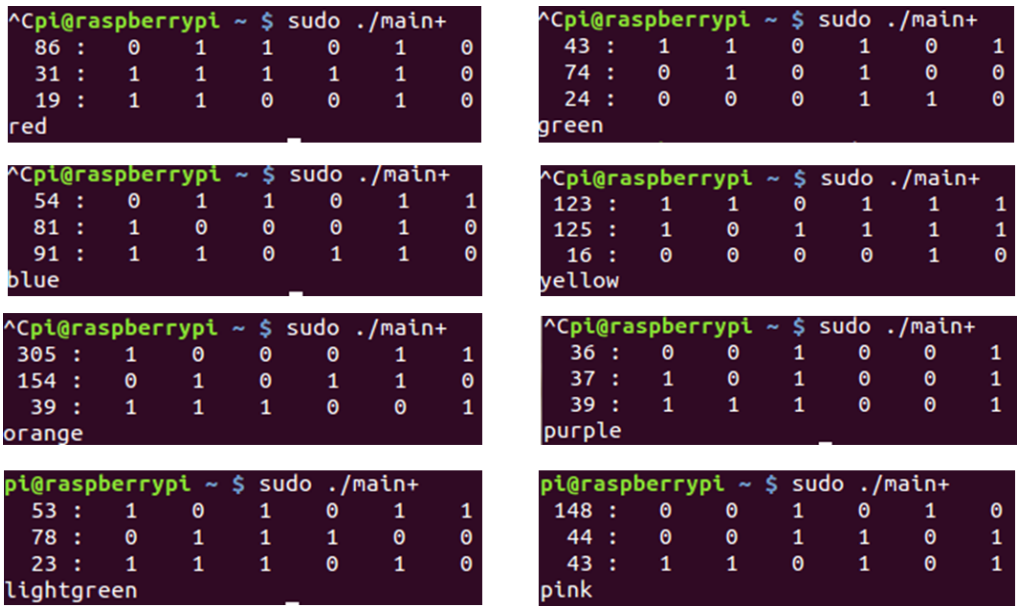


図 3.5　カラーセンサーにより色の識別結果例

それぞれの色はカラーセンサーで大まかには判断出来るが、周囲の光の量によって値が変化するため、朝と夜での値の差が大きいことが問題点である。解決策としては、フォトリフレクタのようにセンサーの横に LED を設置し、周囲の光が与える影響を減らすことなどが挙げられる。ここで、カラーセンサーにより色を取得・識別する関数のソースコードを付録 2 に示す。

3.5　圧電スピーカー

カラーセンサーによって判断した色に対応した「ドレミファソラシド」の音を出す。音の周波数と対応している色を表 3.3 に示す。また圧電スピーカーを鳴らす関数のソースコードを付録 3 に示す。

表 3.3　周波数と音と色の対応表

周波数	音	色
262	ド	赤
294	レ	緑
330	ミ	青
349	ファ	黄
392	ソ	橙
440	ラ	紫
494	シ	黄緑
525	ド	桃

3.6　Bluetooth アダプター

Bluetooth ドングル(BT-MicroEDR2X)(図 3.6)を Raspberry Pi に搭載されている USB ポートに差し込み接続することで、Bluetooth の機能を追加出来る。

図 3.6　Bluetooth ドングル

4.Bluetooth による無線通信

Bluetooth を用いて RFCOMM プロトコル[14]による仮想シリアル接続を行った。Raspberry Pi を操作したり、PC で作成したプログラムを Raspberry Pi に送ったりするために Raspberry Pi と PC との接続を有線で行っていると、ロボットを実際に動かす場合に線がロボットの進行の妨げになり、ロボットの動作が途中で停止してしまう場合がある。

そこで本研究では Bluetooth を用いて無線接続を行った。なお、無線接続を行った後のロボットへのログインは minicom というシリアル通信プログラムを用いて行っている。接続手順の詳細を以下に示す。

【接続手順】

(1) PC と Raspberry Pi を LAN ケーブルに繋ぎ、PC 側の端末から Raspberry Pi にログインする。

(2) Raspberry Pi に Bluetooth のライブラリがインストールされていない時は以下の様にしてインストールする。

```
raspberry$ sudo apt-get install bluez
```

ここで、Raspberry Pi をシャットダウンしないままドングルを差してしまうと、Raspberry Pi が強制的にシャットダウンしてしまうので Raspberry Pi を一旦シャットダウンする。

(3) Raspberry Pi に Bluetooth のドングルを差し、認識しているか確認する。以下の例ではデバイス番号 hci0 として認識されている。

```
raspberry $ hciconfig
hci0:   Type: BR/EDR   Bus: USB
        BD Address: 00:1B:DC:0F:49:5C   ACL MTU: 310:10   SCO MTU: 64:8
        UP RUNNING PSCAN
        RX bytes:1365 acl:0 sco:0 events:58 errors:0
        TX bytes:1206 acl:0 sco:0 commands:55 errors:0
```

(4) PC 側の Bluetooth とペアリングを行う。PC 側に Bluetooth のドングルを差し、Raspberry Pi 側で以下のコマンドを実行する。

```
raspberry $ sudo hciconfig hci0 piscan
raspberry $ sudo bluetooth-agent 0000
```

(5) 同様のコマンドを PC 側でも実行する。

PC $ sudo hciconfig hci0 piscan
PC $ sudo bluetooth-agent 0000

(6) RFCOMM プロトコルを用いて仮想シリアル回線を繋ぐ。Raspberry Pi 側で以下のコマンドを実行する。以下の例ではチャンネル番号を 3 にした。

raspberry $ sdptool add --channel=3 SP
S serial Port service registered

(7) シリアルポートが出来ているか確認する。以下の例では /dev/rfcomm0 というシリアルポートが出来た。

raspberry $ sdptool browse local
raspberry $ sudo rfcomm watch /dev/rfcomm0 3 /sbin/agetty rfcomm0 linux 115200
Waiting for connection on channel 3
Connection from 00:1B:DC:0F:75:82 to /dev/rfcomm0
Press CTRL-C for hangup

(8) PC 側でペアリングができているか確認する。
PC$ hcitool scan
Scanning ...
	00:1B:DC:0F:49:5C	raspberrypi-0

(9) PC 側でも RFCOMM プロトコルを用いて仮想シリアル回線を繋ぐ

PC$ sdptool add --channel=3 SP
S serial Port service registered

(10)シリアルポートが出来ているか確認する。以下の例では /dev/rfcomm0 というシリアルポートが出来た。

PC$ sdptool browse local
PC$ sudo rfcomm watch /dev/rfcomm0 3 /sbin/agetty rfcomm0 linux 115200
Waiting for connection on channel 3

Connection from 00:1B:DC:0F:75:82 to /dev/rfcomm0
Press CTRL-C for hangup

図 4.1 に仮想シリアル接続完了時の画面を示す。上が PC 側の画面で、下が Raspberry Pi 側の画面である。赤く囲んでいるところのように、PC 側と RaspberryPi 側の両方でコネクトされたというメッセージが出ると接続完了となる。

図 4.1　接続完了画面

(11) PC 側で別の端末を開き、minicom を使って Raspberry Pi にログインする。

(12) LAN ケーブルを抜く。

【トラブルシューティング】

(Q) ~$ sudo rfcomm connect hci0 00:1B:DC:0F:49:5C
を実行したときに
Can't connect RFCOMM socket: Connection refused
というエラーが出た。

(A) ポート番号を後ろに付けることによって以下のような正常な結果が得られる
~$ sudo rfcomm connect hci0 00:1B:DC:0F:49:5C 3
Connected /dev/rfcomm0 to 00:1B:DC:0F:49:5C on channel 3
Press CTRL-C for hangup

(Q) Can't connect RFCOMM socket: Operation now in progress
というエラーが出た。

(A) コマンドのどこかに入っている全角を取り除く

(Q) Can't create RFCOMM TTY: Address already in use
というエラーが出た。

(A) sudo rfcomm release hci0
を実行して hci0 のアドレスを解放することにより、すでにアドレスが使われているというエラーが出なくなる

5.動作実験

5.1　実験概要

実際に黒線のみでライントレースを行う実験と、色線を含めたライントレースを行う 2 つの実験を行った。なお黒線のみでのライントレース実験時はランダムに音を出した。

5.2　結果

黒線のみでのライントレース実験では正常にトレース動作した。ただし色線も含めたライントレース実験においては正しく音が鳴らなかった。

5.3　考察

【良かった点】
・モータやセンサーがむき出しになっているため、外側から見てどのように動作しているかが分かりやすかった
・通常のライントレースロボットとは違い音が出ている分、周りの興味を引きやすかった

【悪かった点】
・左右の揺れが大きかった
・全体的に重いため速度に多少の変化が生じた
・周囲の光の量によってカラーセンサーの出力値が変わってしまった
・電池の残量が一定の値よりも減ってしまうとモータの動作が弱くなり、ロボットの進む速度が遅くなった。

悪かった点の改善策としては、まずライントレース中の左右の揺れについてはフォトリフレクタの数を増やし、線の感知度を上げることが考えられる。また重さによる速度の変化については、重いブレッドボードを使う代わりに回路を基盤にはんだ付けすること、モータの強度を上げることなどが考えられる。カラーセンサー値の一定化については、無色 LED をセンサーの近くに取り付けて光の量を一定にすることが考えられる。また電池残量が LED などで分かるようにすることが考えられる。

また今回は色によって音を出しながらライントレースするまでの実装には至らなかった。原因としてはフォトリフレクタ使用時の周囲の光を考慮していなかったことが考えられる。

6.まとめ

本研究では、子ども達にモータやセンサーに興味を持ってもらうことを目的としたライントレースロボットを作成した。

今後の展望としては、色によって音を出しながらライントレースすることに成功した後ロボットを小型化し、幼稚園や保育園、公開講座、地域のイベントなどで出前授業をして実際に子供たちの前でロボットを動かすことが挙げられる。その際には、主催者側が用意したライントレースの基本となる黒線のコースの周りに好きな色線を組み合わせて描いてもらう予定である。それにより思った通りの音が出るため、より子供たちの興味を得られると考えている。また、簡単な説明が書いてあるパンフレットなどを配ることでよりセンサーやモータについての理解を得られると考えている。

参考文献

[1] 身近なロボット
http://www.robo2007.jp/technology/japan.html
[2] ロボット作成授業実践事例
http://www.nier.go.jp/ecase/document_p/D0003153_-1_GoOutBlock.html
[3] マインドストーム EV3
http://education.lego.com/ja-jp/preschool-and-school/secondary/mindstorms-education-ev3
[4] Raspberry Pi
http://www.raspberrypi.org/
[5] TAMIYA 工作・ロボクラフト情報
http://www.tamiya.com/japan/robocon/index.htm
[6] 日経 Linux, 2014 年 4 月号, P105～P107.
[7] 日経 Linux, 2014 年 11 月号, P46.
[8] フォトリフレクタの使い方
http://usicolog.nomaki.jp/engineering/avr_lineTracer/photoReflector.html
[9] モータドライバの使い方
http://usicolog.nomaki.jp/engineering/avr_lineTracer/motorDriver.html
[10] カラーセンサーを使ってみよう
http://robot.tamagawa.ac.jp:8080/cyber/mbed/colorsensor.html
[11] Arduino
http://www.arduino.cc/
[12] 負論理
http://ja.wikipedia.org/wiki/%E8%B2%A0%E8%AB%96%E7%90%86
[13] HTML カラーコード
http://www.colordic.org
[14] RFCOMM
http://www.wdic.org/w/WDIC/RFCOMM

【付録１　ライントレースを行う関数】

```
PI_THREAD (car)
{
	//モータ制御の初期化
	softPwmCreate(M_rightPin,0,100);
	softPwmCreate(M_leftPin,0,100);

	int state = S_zensin;
	int stop_count = 0;


	//無限ループ
	for(;;){

	// センサー値取得
		//if(digitalRead(SwitchPin) == 1){
			//digitalRead(F_rightPin) = AD(CS,CH2);

			int F_left = AD(CS,CH0);
			int F_center = AD(CS,CH1);
			int F_right = AD(CS,CH2);
			//printf("%d¥n",F_left);
			//printf("%d¥n",F_center);
			//printf("%d¥n",F_right);

			line = F_center;


	// 状態遷移
		if( state == S_zensin ){
			if( F_left == 0 ){
				state = S_sasetu;
			}
			else if( F_right == 0 ){
				state = S_usetu;
			}
```

```
        }
        else if( state == S_sasetu ){
                stop_count++;
                if( stop_count >= 5000 ){
                        state = S_stop;
                }
                else{
                        if( F_center == 0 ){
                                state = S_zensin;
                                stop_count = 0;
                        }
                }
        }
        else if( state == S_usetu ){
                stop_count++;
                if( stop_count >= 5000 ){
                        state = S_stop;
                }
                else{
                        if( F_center == 0 ){
                                state = S_zensin;
                                stop_count = 0;
                        }
                }
        }
        else{

        }


// モータを動かす
        switch(state){
          //前進
          case S_zensin:
                softPwmWrite(M_rightPin,50);
                softPwmWrite(M_leftPin,70);
```

```
                    //printf("zensin¥n");
                    break;

                //左折
                case S_sasetu:
                    softPwmWrite(M_rightPin,50);
                    softPwmWrite(M_leftPin,0);
                    //printf("sasetu¥n");
                    break;

                //右折
                case S_usetu:
                    softPwmWrite(M_rightPin,0);
                    softPwmWrite(M_leftPin,70);
                    printf("usetu¥n");
                    break;

                //停止
                case S_stop:
                    softPwmWrite(M_rightPin,0);
                    softPwmWrite(M_leftPin,0);
                    printf("teisi %d¥n",stop_count);
                    break;
                }

        delay(1);
        }
}
```

【付録２　カラーセンサーで色を取得・識別する関数】

```
PI_THREAD (sound)
{
        int color[36];
        int col2[36];
        int i,j;
        int R,G,B;

        //無限ループ
        for(;;){
                //if( digitalRead(SwitchPin) == 1 && line == 1 ){
                        //色を読み取るためにゲート,クロックを 0 にする
                        digitalWrite(C_gatePin,0);
                        digitalWrite(C_ckPin,0);
                        // Range を 1 にする。
                        digitalWrite(C_rangePin,1);

                        delay(500);

                        digitalWrite(C_gatePin,1);      // 測定開始
                        delay(500*1);                   // 待つ
                        digitalWrite(C_gatePin,0);      // 測定終了
                        delay(500*1);

                        delayMicroseconds(4);

        //カラーセンサーから色を読み取る
                        for(i=0;i<3;i++){
                                int value = 0;
                                for(j=0;j<12;j++){
                                        digitalWrite(C_ckPin,1);
                                        delay(5);
                                        //シフト計算
                                        value += digitalRead(C_doutPin) << j;
                                        col2[i*12+j] = digitalRead(C_doutPin);
                                        digitalWrite(C_ckPin,0);
```

```
                delay(5);
        }
        color[i] = value;
        if(i != 2){
                delayMicroseconds(3);
        }
}

for(i=0;i<3;i++){
        printf("%4d :",color[i]);
        for(j=0;j<12;j++){
                printf("%4d ",col2[i*12+j]);
        }
        switch(i){
                case 0:
                        R = color[i];
                case 1:
                        G = color[i]*0.8;
                case 2:
                        B = color[i]*1.5;
        }
        printf("¥n");
}


//色の判定の計算

        //白か色があるか
        if((R > 100)&&(G > 100)&&(B > 100)){
                printf("white¥n");
                delay(500);
        }
        //赤系
        else{
                if((R > 30)&&(R > G)&&(R > B)){
                        if(R-G < 5){
```

```
                printf("yellow¥n");
                softToneWrite(SPK,scale[3]);
                delay(500);
        }
        else if(R-B < 5){
                printf("purple¥n");
                softToneWrite(SPK,scale[4]);
                delay(500);
        }
        else if(G > (R/2)){
                printf("orange¥n");
                softToneWrite(SPK,scale[5]);
                delay(500);
        }
        else if((G > 30)&&(B > 30)){
                printf("pink¥n");
                softToneWrite(SPK,scale[6]);
                delay(500);
        }
        else{
                printf("red¥n");
                softToneWrite(SPK,scale[0]);
                delay(500);
        }
}
//緑系
if((G > 30)&&(G > R)&&(G > B)){
        if(G-R < 5){
                printf("yellow¥n");
                softToneWrite(SPK,scale[3]);
                delay(500);
        }
        else if((R > 30)){
                printf("lightgreen¥n");
                softToneWrite(SPK,scale[7]);
                delay(500);
```

```
                                        }
                                        else{
                                                printf("green¥n");
                                                softToneWrite(SPK,scale[1]);
                                                delay(500);
                                        }
                                }
                                //青系
                                if((B > 30)&&(B > R)&&(B > G)){
                                        if(B-R < 5){
                                                printf("purple¥n");
                                                softToneWrite(SPK,scale[4]);
                                                delay(500);
                                        }
                                        else{
                                                printf("blue¥n");



        softToneWrite(SPK,scale[2]);
                                                delay(500);
                                        }
                                }
                        }

                //}
        }
}
```

【付録 3　圧電スピーカーを鳴らす関数】

```
PI_THREAD (speaker)
{
	int i;

		if(wiringPiSetupGpio()<0){
			//エラー処理
		}
		if(softToneCreate(SPK)!=0){
			//エラー処理
		}
		/*for(;;){
			for(i=0;i<8;i++){
				printf("%03d¥n",scale[i]);
				softToneWrite(SPK,scale[i]);
				delay(500);
			}
		}*/

		/*switch(oto){
			case 0:
				softToneWrite(SPK,scale[0]);
				delay(500);
				break;
			case 1:
				softToneWrite(SPK,scale[1]);
				break;
		}*/

	return 0;

}
```