電気電子工学実験３：白井研究室
ボードコンピュータを用いたロボット実験
ボードコンピュータを用いたロボットを題材に、 マイコンのプログラミングやロボットの制御などの技術を学ぶ

Raspberry Pi を用いたマイコン制御のロボットについて学ぶ。1 週目では Raspberry Pi の機能、LinuxOS と Python プログラム、GPIO の制御について学ぶ。2 週目では、PWM の使い方、DC モーターによるロボットの移動制御と、Wifi による PC(他の Raspberry Pi)と Raspberry Pi の通信について学ぶ、3 週目ではロボットのリモコン操作、ウェブカメラからのビデオ配信と、ロボットの操作との連携について学ぶ。 4 週目では今までの実験を踏まえてより高性能なロボット作成に挑戦する。

Raspberry Pi の環境設定

大事なポイント：<u>電源は最後に接続、最初に外す</u>

電源を繋いでいるときは、他の機器の抜き差し禁止 （壊したら弁償すること）

GPIO とは汎用の入出力のこと。Raspberry Pi の GPIO の出力は 3.3V が最大値

注意点:

(1) LED は必ず抵抗を直列に接続する、(2) 電源を入れたまま配線作業する、(3) IC には必ず電源と GND をつなぐ、(4) GPIO の出力ピン同士を接続しな

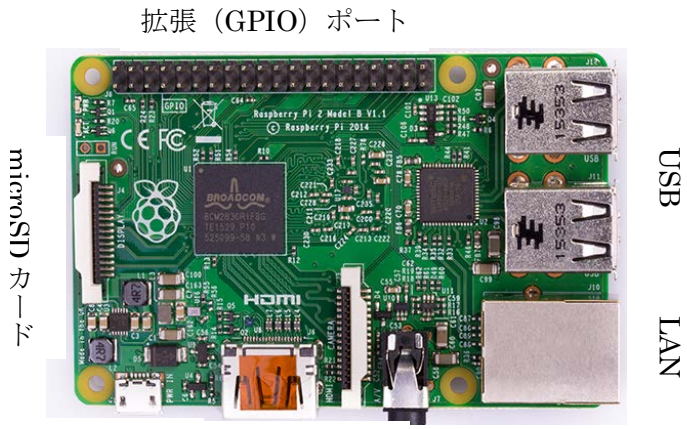


図 2.1. Raspberry Pi 2 B+

2 週目の実験： DC モーターによるロボットの移動制御と、WiFi による P C(他の Raspberry Pi)と Raspberry Pi との通信

復習: Python 環境設定

実験(1): 圧電スピーカーで音を鳴らす

ハード: 1kΩ 抵抗、圧電スピーカー ANYM-SPT08

ソフト: http://lang.sist.chukyo-u.ac.jp/Classes/Experiment3/Programs/playMusic2.py

課題 1. 図 2.2 に示す回路を組み、以下に示すサンプルプログラムを走らせ、音がなることを確認せよ。またプログラムの解説を試みよ。

```
# playing tone with Python 3
import time
import RPi.GPIO as GPIO

pin=12
toneName={'A': 0, 'a':1, 'B':2, 'C':3, 'c':4, 'D':5,
          'd':6,'E':7, 'F':8, 'f':9, 'G':10, 'g':11}
music = 'G-G-<A--->G-G-<A--->G-<A-a-A->G-<A>GD---R'
tempo=0.25
freqcore = 440
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.OUT)

p = GPIO.PWM(pin,freqcore)
p.start(0)

for j in range(len(music)):
    note = music[j]
    if note == 'R':             # pause
        p.ChangeDutyCycle(0)
        time.sleep(tempo)
    elif note == '>':           # 1 octave lower
        freqcore /= 2
    elif note == '<':           # 1 octave higher
        freqcore *= 2
    elif note == '-':           # slur
        time.sleep(tempo)
    else:           # one tone
        i = toneName[note]
        freq = freqcore * (2 ** (i/12))
        p.ChangeDutyCycle(80)
        p.ChangeFrequency(freq)
        print(note, ': %d Hz' % freq)
        time.sleep(tempo)

p.stop()
GPIO.cleanup()
```

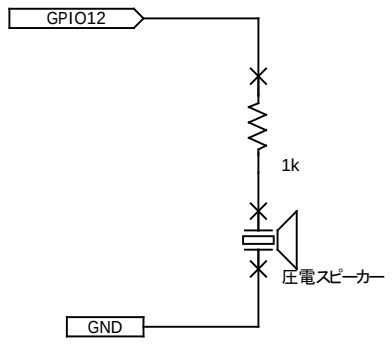


図 2.2 圧電スピーカー回路

圧電スピーカーは、電圧を加えると変形するという声質を持つ圧電素子と、空気を振動させて音を生み出すための振動板から構成されている。つまり、圧電スピーカーの仕組みは以下の通り:

① 圧電素子に加える電圧を高速に変化させる

②圧電素子が高速に変形を繰り返す

③圧電素子の変形に応じて振動板が振動する

④その結果音が発生する

音楽の知識 : 左のプログラムで基準とした「ラ(A)」周波数は 440Hz、1 オクターブあがると周波数は 2 倍、1 オクターブ下がると(その逆で)周波数は 1/2 倍になる。また「ラ(A)」から「ソ(G)」までの周波数は、「ラ」の周波数に 2 の 12 分の 1 乗倍したものに等しい。

実験(2): 超音波距離センサを用いた距離計測

ハード:　超音波距離センサモジュール HC-SR04, 10kΩ と 4.7kΩ抵抗

注: HC-SR04 は入出力とも TTL(5.0V)、Raspberry Pi は 3.3V

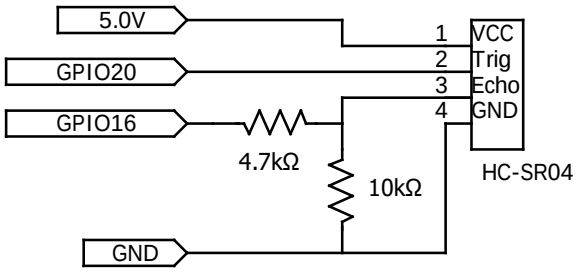


図 2.3 超音波距離センサによる距離計測の回路

距離計測の原理：

超音波距離計測センサは音（超音波）を用いて音源と対象物との距離を計測する。短距離なら精度もよく、人間の可聴領域外の音を使っているため、環境にもやさしい。超音波発振器、受信機、制御回路からなり、計測距離は 2cm から 400cm である。発振器から発せられた超音波が対象物に当たり、その反射音を受信器で受信する、その間の距離を計測する。「音の速さ＝距離／時間」であり、気温を T℃とすると標準大気中において室温付近では「音の速さ＝331.5+0.6T m/s」という近似式が成り立つ。この 2 つの式から距離が求められる(便宜的に音の速さを 340 とすることが多い)。

HC-SR04 を用いた距離計測では次のステップによる：

1) TRIG 入力にトリガー信号を与える。ここで 10μs 以上の間 HIGH 信号を送ることがトリガー信号の条件である。

2) これにより 40kHz の短時間の音波（超音波）8 個が発信される。

3) 前方に障害物があれば、その音波はその障害物に反射する

4) 音波を送信した後、反射波を受信するまで ECHO 出力が HIGH になる。このパルス幅(off-on-off)は対象物との距離により 150μs から 25ms の間である。反射波を受信しなければ(障害物がなければ)38ms である。

ここで、ECHO 出力は 5V であり、Raspberry Pi の GPIO は 3.3V なので、ECHO 出力を直接 Raspberry Pi に入力することはできない。そのため図 2.3 の回路では分圧回路を入れている。

距離測定のためのプログラムは

http://lang.sist.chukyo-u.ac.jp/Classes/Experiment3/Programs/distanceHCSR04.py

を参照のこと

課題 2：障害物を 5cm から 200cm の間の距離に障害物の材質（スポンジ、衣服、紙、木、金属など)や大きさや角度(30 度ずつ角度を変えてみる)をいろいろに変えて距離計測の精度を求めよ。

図 2.4　TA7291P
端子の番号は図の左が 1 、最右端が 10

実験(3): この実験以降は、ロボットに Raspberry Pi を組み込む。　使用する車体は 2 つの DC モーターがついている「三輪型」のものである。残る一つの車輪はどの方向にも回転するが、モーター駆動ではない。

Raspberry Pi の GPIO では DC モーターを駆動させられないため、モーターの回転を制御するための「モーター・ドライバー」TA7291P を利用する(http://akizukidenshi.com/download/ta7291p.pdf)

端子の説明は表 2.1 を、機能については表 2.2 を参照のこと。

表 2.1　TA7291P の端子（3 と 9 は使用しない）

端子記号	端子番号 P	端　子　説　明	
V_{CC}	7	ロジック側電源端子	Vcc は Raspberry Pino の 5V を使用 Vref≦Vs 、20V 以下
V_S	8	出力側電源端子	
V_{ref}	4	制御電源端子	
GND	1	GND	
IN1	5	入力端子	Raspberry Pi と接続
IN2	6	入力端子	
OUT1	2	出力端子	モーターと接続
OUT2	10	出力端子	

表 2.2 TA7291P の機能

入　力		出　力		モード
IN1	IN2	OUT1	OUT2	
0	0	∞	∞	ストップ
1	0	H	L	CW / CCW
0	1	L	H	CCW / CW
1	1	L	L	ブレーキ

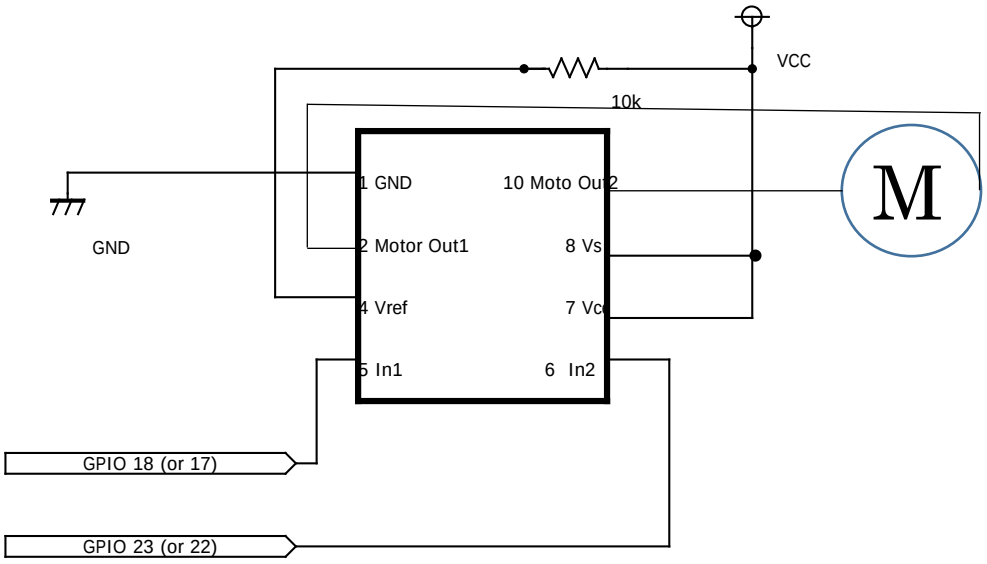


図 2.5 モーター・ドライバー回路(モーター１個の制御)

課題３：図 2.5 に基づき回路を組み、左のモーターと右のモーターに対し時計回りの回転、反時計回りの回転をそれぞれ行わせてみよ。なおここでは「ディジタル信号によるモーター制御」を行っている。つまり、表 2.2 において入力(0 は LOW, 1 は HIGH を出力する)ことによって DC モーターに通電し、それによってモーターを回転させているのである。そのためのプログラムとして dcmotor.py が用意されている。

注意：最初はどちらの方向に動くかわからないので、ロボットを手に持って動作させたほうがよい。またモーター停止プログラム(halt.py: ウェブページからダウンロード)をすぐ実行できるように構えておくこと。

これらがすべて動くことを確認した後に、右と左のモーターの回転を組み合わせて、「前進」「後退」「左回転」「右回転」それぞれを行う Python の関数を作り（参考のプログラムとして testDrive.py が用意されているが、ロボットによって回路が異なることがあり、そのままでは多分望んだようには動かないので、改変が必要である)、それぞれ期待通り動くことを確認せよ。

参考：これらの関数は、time モジュールの sleep を用いて、ある一定時間だけ動作した後、stop もしくは brake を呼ぶようにすると良い。

注意：回路の作成、および Raspberry Pi の GPIO の接続には十分注意すること。例えば電源端子と GND 端子とを間違えると、簡単に電子素子や Raspberry Pi は壊れてしまう。２人で責任をもって注意深く行うこと。例えば一人が回路をくみ、もうひとりがそれをチェックし、その上で通電するようにするのがよい。回路が熱くなったり、異臭がしたりした場合には、すぐに電源を切ること。

電源をつないだまま、回路を変更したり、USB を抜き差ししたりしないこと。壊れた場合は弁償するように。

課題 4: 回路はそのままにして、今度は PWM によるモーター制御を行え。そのために pwmmotor.py というプログラムと、それを用いたプログラムとして testDrivePWM.py が用意されている（課題３と同じ理由で、そのままでは望んだようには動かないので、改変が必要である)。

ディジタル信号によるモーター制御と PWM によるモーター制御を「定められた距離を進む、もしくは定められた角度だけ回転する」 という観点から、どちらの方式がよいか、比較・考察せよ。

以上の実験によって感じたことは、キーボード、マウス、モニター（そして電源）のケーブルが邪魔、ということだろう。電源としてはモバイルバッテリーを用い、また Wifi によって PC もしくはワークステーション(OS は Linux)から Raspberry Pi にアクセスすることでこれらのケーブルをなくすことに取り組もう。

注意: `ps aux | grep webiopi` を Terminal から実行すると、

```
2101  ?   Ssl  0:03  /usr/bin/python3  -m webiopi -l /var/log/webiopi  -c /etc/webiopi/config
2112  ps/0   S+  0:00  grep –color=auto webiopi
```

というようなメッセージが表示されるときは（2101 のような数は状況によって異なる）WebIOPI はバックグラウンドで動いているため、`python` プログラムからモーター制御ができない(`CPIO` の制御を

WebIOPi が取ってしまっているため)ことがある。その場合、

sudo kill -9 2101 (この番号は ps aux|grep webiopi した結果から得よ、いつも 2101 とは限らない)

として、WeibIOPi の実行を停止させること。

実験(4): Wifi によるリモートアクセス

最初に、ロボットの Raspberry Pi の IP アドレスを確認する。Terminal ウィンドゥで、 ifconfig コマンドを実行する。すると、おおよそ次のようなメッセージが表示される：

(前半略)

wlan0 Link encap:Ethernet HWaddr 00:00:00:00:05:89

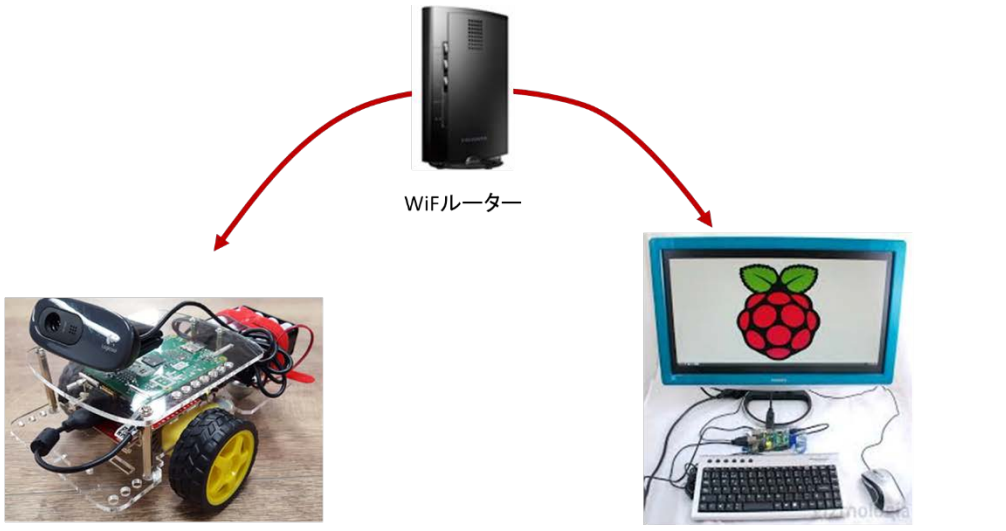
inet addr:192.168.0.57 Bcast:192.168.0.0 Mask:255.255.255.0

inet6 addr: fe80::200:ff:fe00:589/64 Scope:Link

赤枠で囲んだところが、ロボットの Raspberry Pi の IP アドレスである（注：上は一例であり、実際には違う値が表示されるはず)。それを記録しておく。shutdown したあと、Raspberry Pi に必要な電子回路を接続しておく。ただし、モニター (HDMI ケーブル)、キーボード、マウスは外しておく。そして、モバイルバッテリーから電源を接続し、2−3 分待つ。（マウスやモニターなどが接続されていなくても、Raspberry Pi はちゃんと起動する。ただ、モニターを接続していないので、それを見る手段がないだけである)。また別に用意されている Raspberry Pi を用いる。こちらの Raspberry Pi を Raspbian と呼んでロボットと区別することにする。

Raspbian の USB 端子に WiFi ドングル、キーボード、マウス、USB カメラを接続、HDMI ケーブルを通してモニターを接続したのちに、電源を接続する。そして、ロボットの Raspberry Pi と Raspbian とを Wifi で接続する（図 2.6)。Raspbian の Terminal ウィンドゥから `slogin IP` アドレス (エンター)のあと、パスワードとして raspberry を入力する（ただし、このパスワードは表示されない)。なお、Windows PC でも Tera Term のような端末アプリケーションによって同じことが可能である（時間があれば試みよ)。これにより、Raspbian の Terminal がロボットの Raspberry Pi の Terminal として機能するようになる。（参考：VNC を使う方法もあるが、やや処理が重い)。

この上で、実験(3)および実験(4)の課題が実現できることを確かめよ。終了はロボットと Raspbian の両方を shutdown させる(Terminal で `sudo shutdown -h now` とし 1,2 分待つ)こと。



https://codezine.jp/article/detail/9829 から借用

http://techcrash.net/minecraft-pi-raspberry-pi-edition/ から借用

図 2.6 WiFi による Raspberry Pi 間の通信