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 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。

3 週目　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、ロボット操作との連携

実験(1) この実験では、ロボットに接続してある Raspberry Pi とは別に用意されている Raspberry Pi を用いる。なお、2 週目の実験と同様、一方をロボット、もう一方を Raspbian と読んで区別する。この実験は Raspbian に対するものである。

実験手順：SD カードはロボットから取り外し（電源が接続されていないことを確認すること）、もう一方の「電源が接続されていない」Raspbian に入れておく（Raspbian から取り出した SD カードは後で使用するので保管しておくこと）。

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。 Terminal において luvcview を起動する。これは USB カメラの映像をモニターに映し出すソフトである（同様のものとして cheese があるが、luvcview の方が処理が軽い）。そこで USB カメラが機能していることを確認せよ。カメラの前で手を振ったりして、それがモニターに表示されるまでの遅れを計測せよ(これが後でロボットをリモコン操作するための基礎資料となる)。

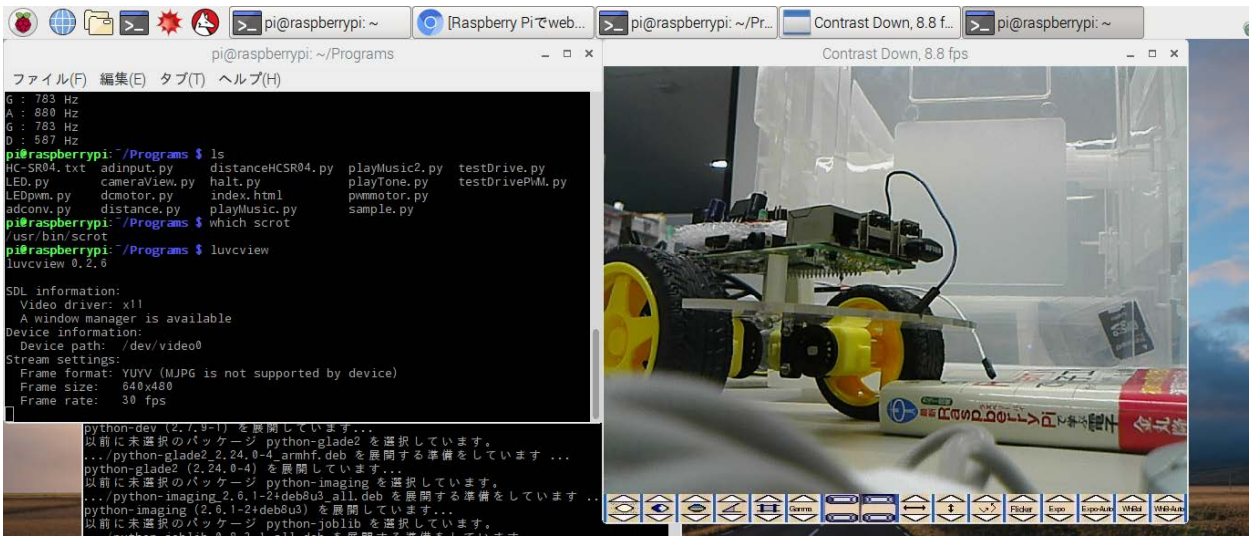

図 3.1　luvcview を実行した画面: 右がカメラからの映像

実験(2)　MJPG-streamer の利用

出典: https://github.com/jacksonliam/mjpg-streamer

説明:　mjpg-streamer は複数の JPEG(画像の一形式)フレームを複数の出力にコピーするためのアプリケーションである。具体的な利用法としては、IP 通信に基づくネットワークを用いウェブカメラからの JPEG ファイルをストリーム配信によって Chrome や Firefox などのビューワーや MJGP ストリーム入力できるソフト(例: mplayer)に配信することがある。もともとは能力の小さな CPU やメモリが限られたシステムで用いるためのものであり、uvc_streamer の後継にあたる。ここで UVC カメラでは直接 JPEG データを生成するため M-JPEG ストリームを高速・高性能で行えるという特徴がある。input_uvc.so という入力モジュールは JPEG フレームを取得し、output_http.so という出ろクモジュールは JPEG データを HTTP (ブラウザとサーバーとの通信プロトコル)で配信するものである。

実験手順:既に　mjpg-streamer の環境設定は済んでいる。ホーム・ディレクトリにある testMjpg.sh を Terminal 上で実行せよ(コマンドは `./testMjpg.sh` もしくは `bash testMjpg.sh`)。図 3.2 に示すように(右の画面)、いろいろメッセージが出るが無視せよ。次に、ブラウザを立ち上げ、 `localhost:8080/stream.html` にアクセスせよ。ここで localhost とは「今使っている」コンピュータを指し、`8080` とはポート番号である(注:ブラウザは普通は HTTP (ポート番号 80) でサーバーと通信する。このように他のコンピュータと通信するためにはポート番号が使われるが、1000 以下の番号は固有の使いみちが決まっている。`8080` は MJPG が独自に設定したポート番号である(変更することも可能)。ここでは localhost (つまり自分)と `8080` というポート番号で通信することを宣言している。次に ifconfig を用いてこの IP 番号を調べ (仮に `192.168.0.99` とする)、スマホ(もしくは PC)を白井ゼミの WiFi サーバー(AirPort)に接続してから、 `http://192.168.0.99:8080/stream.html` にアクセスし、図 3.2 (左のブラウザ画面)と同様の画面が見られることを確認せよ。

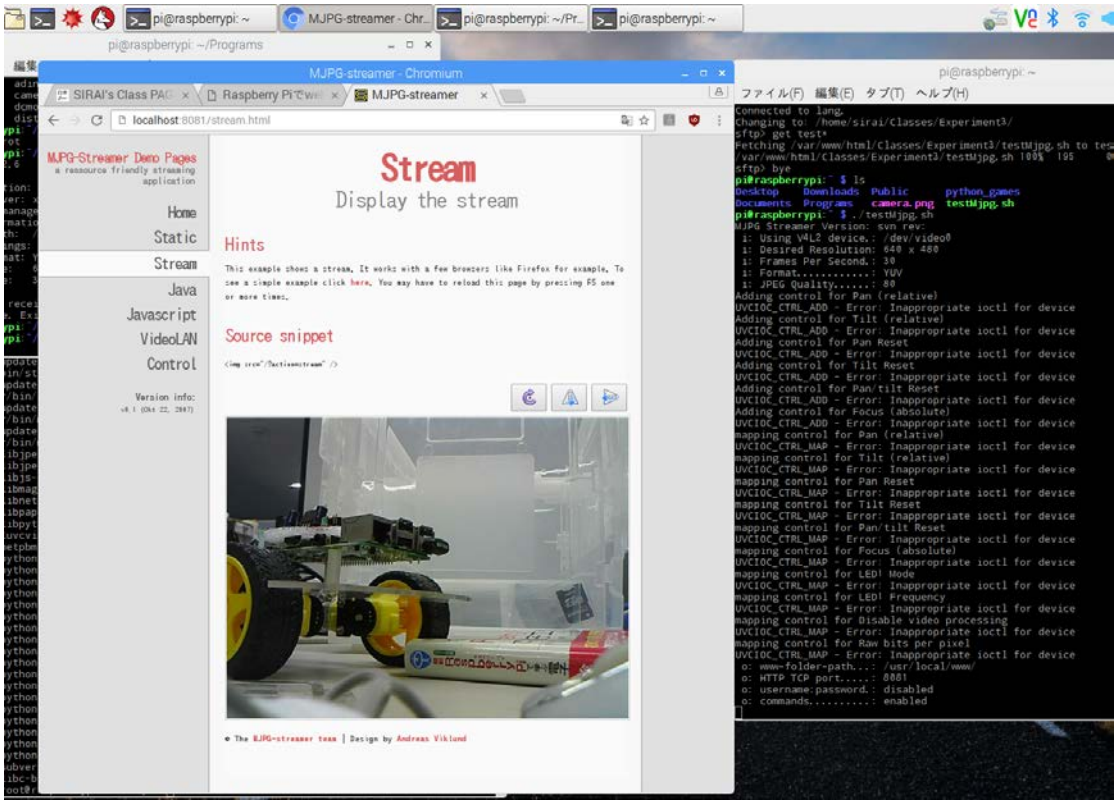

図 3.2 mjpg-streamer を実行した画面:左はウェブ配信されたカメラからの画像

実験(3): ロボットのリモコン操作

ここからは 2 週目の実験の延長に相当する実験を行う。ロボットと Raspbian の電源を切り、SD カードを Raspbian からロボット本体に戻す。これによりロボットの SD カードには、ロボット操作のためのライブラリと、MJPG-streamer のためのプログラムの両方が揃ったことになる。また Raspbian には保管しておいた SD カードを挿す。

ロボットの USB 端子には Wifi ドングルと USB カメラだけを接続する(キーボードやマウスは外す)。Raspbian には実験(2)の状態、つまりキーボード、マウス、WiFi ドングル、HDMI モニター端子を接続しておく。また Raspbian には別途用意されている SD カードを挿入してあることを確認する。そして、両方の電源をいれる:ロボットにはモバイルバッテリーの電源からつなぎ、Raspbian には別途用意された電源をつなぐ。ここではロボットから起動するのがコツである。こうすることで Raspbian が立ち上がったときにはロボットの準備もできているはずである。

実験内容: 「前進」の関数を利用して、10, 30, 50 cm それぞれの距離だけ前進する関数を作れ。ただし「直進」するのはかなり難しい(左か右に曲がることが予想されるので、左右のモーターの電流値を工夫し、ずれを 2~3cm 程度に収めるように工夫せよ。(ヒント:どのくらい前進状態を保っていればこの条件を達成できるかを考える)。もっとも厳密に 10cm や 30cm などの距離を移動するのは難しいので平均値がその距離だけ進む関数を作り、10 回(くらい)実行したときの平均と誤差の分布を求めよ。

次に、「右回転」(および「左回転」)の関数を利用して、 30°, 90°, 180° それぞれ回転する関数を作れ(「前進」の場合と同様に正確に回ることは難しいので、大体その角度くらいに回転する関数を作り、それを 10 回(くらい)実行したときの平均と誤差の分布を求めよ。

注意:机の上と床では摩擦係数が異なるため、どのような条件下で実験下かを明記すること。床の場合滑りやすいので、stop させてもなかなか止まらないことが予想される。

実験(4) 実験室の机の上を落ちずに一周するプログラムを書き、周回タイムを測れ。トライアルを最大 3 回までとし、そのうちのベストの記録を答えよ。またそのプログラムについて考察すること。

実験(5): ウェブカメラからの画像を見ながらリモコン操作（この実験は第 4 週目に行うのでもよい）

図 3.3 ウェブブラウザに映し出された画面。矢印をクリックするとロボットが動く

この項目の内容は、金丸 (2016)『最新 Raspberry Pi で学ぶ電子工作』講談社,に基づくものである。
ウェブブラウザでロボットのカメラが写した情景を見ながらロボットを操作するシステムを作成する。そのためにここで新しく WebIOPI[1]というシステムと実験 2 で使用した MJPG-streamer を用いる。
WebIOPI は Eric Ptak 氏が開発したアプリケーションで、ウェブブラウザを通して Raspberry Pi の GPIO にアクセスすることができることである。GPIO へのアクセスには管理者の権限が必要であるため、ブラウザからのアクセスは困難なのであるが、WebIOPI によって実現されているのである。

本実験用の Raspberry Pi (実態は SD カードに実装された Raspbian)では既に WebIOPI のインストールは済んでいる。`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はずである（2101 のような数は状況によって異なる）。ここで `/usr/bin/python3 -m webiopi` というようなメッセージが表示されていれば、WebIOPI はバックグラウンドで動いている。そうでない場合は、Terminal で `sudo service webiopi start` を実行してからふたたび `ps aux | grep webiopi` を実行して、WebIOPI が動いていることを確認すること。

実験 5(1): WebIOPI の動作確認

ロボットは、モニター、マウス、キーボードを外し WiFi とカメラだけを USB 端子に接続してあることを確認する。また Raspbian は、モニター、マウス、キーボード、WiFi を USB 端子に接続し、さらに 1 週目で行った LED チカ実

[1] WebIOPI:: http://webiopi.trouch.com/

験の回路を接続する。ここで両方の IP アドレスを確認しておくこと。そして、ロボットと Raspbian を起動する。次に Raspbian で上記の手順により WebIOPI がバックグラウンドで動いていることを確認したあと、のブラウザを立ち上げ、http://192.168.0.99:8080/ にアクセスする（注意: ここで Raspbian の IP 番号を 192.168.0.99 と仮定した）。なお 8080 は WebIOPI のポート番号である（MJPG-streamer のポート番号とは別である）
そして、図 3.4 に示すような画面がブラウザに表示されることを確認せよ。

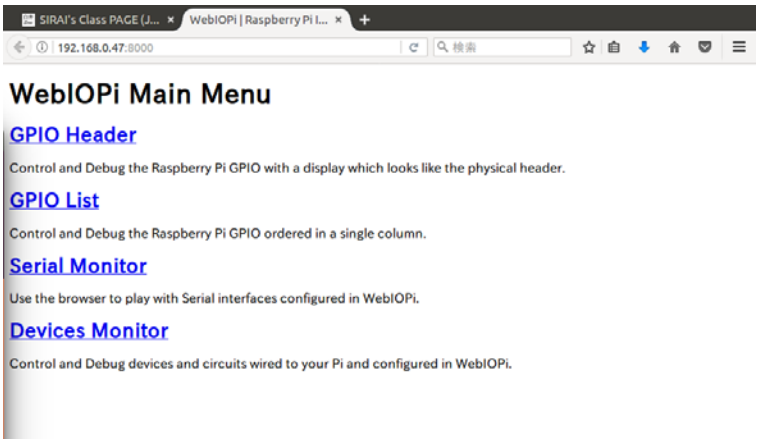

図 3.4 WebIOPI のメニュー画面

実験 5(2): WebIOPI の動作確認(LED チカ)
GPIO Header をクリックすると図 3.5 の画面になる。

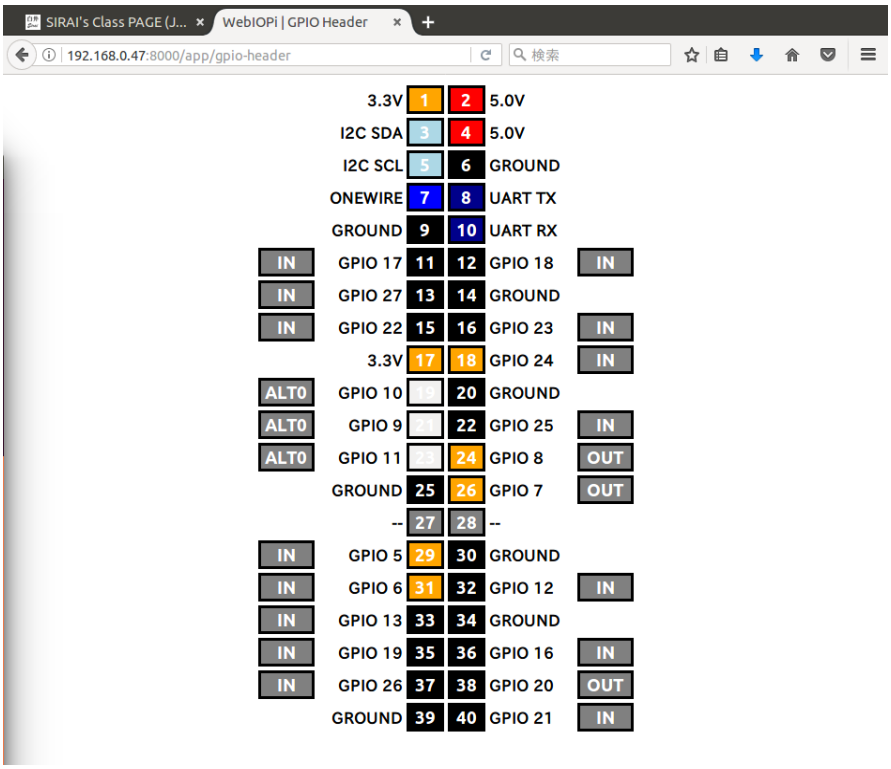

図 3.5　WebIOPI の GPIO Header 画面　（GPIO20 のところは OUT ではなく IN となるのがデフォルト）

見て分かるようにこれは Raspberry Pi の GPIO ピンの状態を示している。LED チカの回路が GPIO 17 に接続されているとする(そうでない場合は適宜読み替えよ)。1 週目の実験では Python プログラムによって GPIO ピンの状態を ON にすることにより、LED に 3.3V の電圧が供給され、LED が光ったことを思い出そう。ここでは WebIOPI によってそれと同じことを行う。それにはまず、GPIO 17 の状態を OUT (出力用を意味する。IN は入力用であり、これがデフォルトの状態)にするため、マウスで GPIO 17 の横にある IN をクリックして OUT に変わることを確認する(注意:変わらない場合は WebIOPI が動いていないか、OS がおかしいか、Raspberry Pi がおかしいか、それに供給されている電流が小さすぎるか、のいずれかである)

つぎに GPIO 17 の横のピン番号「11」をマウスでクリックする。すると、黒色からオレンジ色に変わるはずである。これは、このピンが ON(3.3V が供給)になったことを意味する。そして LED が光っていることを確認せよ。

次に、ピン番号「11」をマウスでクリックし(今度は色がオレンジから黒に変わる)、LED が消灯することを確認せよ。以上が成功すれば、WebIOPI が正常に動作している。

実験 5(3): ロボットの WebIOPI の動作の確認

今度は Raspbian のブラウザからロボットの WebIOPI の動作を確認する。(1) 第 2 週目に行ったように、Raspbian の Terminal ウィンドゥから `slogin` ロボットの `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) -これを「Raspbian からロボットに slogin する」という。そして、`ps aux | grep webiopi` を実行し、ロボット上でも WebIOPI が動作していることを確認する。もしも起動していない場合は、`sudo service webiopi start` とする。

実験 5(4): ウェブ上でカメラ画像を見ながらロボットを操作する

金丸(2016)のサンプルプログラムもアクセス可能となっているので、その条件の上で次を実行してみよう:

1) ロボットの MJPG-streamer を起動:
   testMJPG.sh を起動する(ポート番号 8000 で画像がストリーミングされる)
2) Raspbian のブラウザから http://`192.168.0.98`:8000/bb/06/ にアクセス
   (注: ロボットの IP 番号を `192.168.0.98` と仮定した)

これで図 3.3 のような画面が見えたらひとまず成功である。ここで、bb/06 とは、/usr/share/webiopi/htdocs/の下にある bb ディレクトリ(金丸(2016)のサンプルプログラム)にある 06 ディレクトリを意味する。ブラウザで見えているのはそのディレクトリにある index.html ファイルである。ここでは javascript プログラムの実行結果により画像が挿入され、その上にロボットの向きを示した図がオーバーラップされ、その下に解説の文言が表示されている。このウェブページの表示についてちゃんと理解するには javascript と python の両方のプログラミング言語を知らないといけないため、この実験では script.py にあるピン番号をいじることにより、ロボットのリモコン操作を実現することだけを考えよう。

ブラウザの画面の表示に従い、前進や後退をさせてみよう。それにはマウスでクリックする。なお画面中央の横線をクリックすることで停止でき、これをクリックしない限りは動き続けるので注意する。また、マウスでクリックする位置によってデューティー比が変化する(したがってモーターに供給される電流の大きさが変わる)ので、いろいろな場所をクリックしてその関係を探っておく。もしも矢印の方向とロボットの動作とが合わない場合は多少プログラムを直す必要がある。それについて簡単に説明する。

/usr/share/webiopi/htdocs/bb/06/script.py の最初の方に、

PWM1 = 23

PWM2 = 18

PWM3 = 22

PWM4 = 17

という記述がある。PWM1 と PWM2、および PWM3 と PWM4 がそれぞれ組になって、モーターを制御する。番号はピン番号(BMC)である。これが自分のロボットにあっていないと、前進のつもりが回転したりするため、実験を通して、このピン番号を自分のロボットに適切な番号になるよう設定させよう。

その上で、カメラからの画像だけを通して、ロボットを操作させてみよ（これが、月や原発内部にロボットを送り込んで操作する状況である）。

発展課題: WebIOPi によってロボットを操作する方法に代えて、MJPGstreamer によりカメラからの画像を見ながら、Wifi によってロボットを操作する（ロボットのシステムに slogin し、適切なコマンドを打ち込む、もしくは F で前進、スペースで停止、L で左回転させるというような簡易インターフェースを作って操作する）という方法がある。この 2 つを較べて操作性を比較せよ。