4.4 青色LEDに順方向電圧をかけて点滅させる（上級者向け）

ここでは、上級者向けに「青色LEDを明るく点滅させる」という演習を紹介します。書籍の４章の演習で必要な物品は、３章と同じでしたが、この演習では、表4-1に示す追加物品が必要になります。

表 4-1　本節で必要な物品

物品	備考
青色LED １個	必須。秋月電子通商の通販コードI-06411（10個入り）など
100 Ωの抵抗１本	必須。秋月電子通商の通販コードR-25101（100本入り）、千石電商のコード8ASS-6UHG（10本から）など
10 kΩの抵抗１本	必須。秋月電子通商のパーツセット（書籍の65ページ）に含まれている。単品で購入する場合は秋月電子通商の通販コードR-25103（100本入り）、千石電商のコード7A4S-6FJ4（10本から）など
トランジスタ 2SC1815-Y、2SC1815-GR、2SC1815-BL のうちどれか１つ	必須。秋月電子通商の通販コードＩ-04268、千石電商のコードEEHD-4ZNHなど
ブレッドボード用ジャンパーワイヤ（ジャンプワイヤ）（オスーオス）	必須。3章でも紹介。秋月電子通商のパーツセットに含まれている

4.4.1　トランジスタを用いた回路

書籍の３章と４章では、LEDの取り扱いを学んできました。LEDには順方向降下電圧 V_f があり、そのときに流す電流は10mAや20mA程度であることが多いのでした。

書籍の**4.2**で触れたように、Raspberry PiのGPIOに流すことのできる電流のデフォルト値は8mAですので、このLEDに10mAや20mAの電流を流すことはできません。また、青色のLEDの多くはV_f=3.4V程度であることが多いのですが、Raspberry PiのGPIOの3.3Vではこの電圧に足りません。

LEDに流れる電流が8mAでも十分明るいですし、青色LEDに与える電圧が3.3Vであっても（多少暗いですが）点灯はしますので、実用上は大きな問題はありません。しかし、LEDに10mA～20mAの電流を流す方法や、青色LEDに3.3V以上の電圧をかけて点滅させる方法を知っておくことは有用ですのでここで学んでおきましょう。ただし、5Vピンを用いること、以後の章では用いないトランジスタを用いることなどから、本節の内容は上級者向けとします。

目標は、青色LEDに順方向電圧をかけて点滅させることとしましょう。必要な回路を図4-20に示します。

右下に丸で囲まれた図記号で表された部品がありますが、これを「トランジスタ（特にNPNトランジスタと呼ばれるも

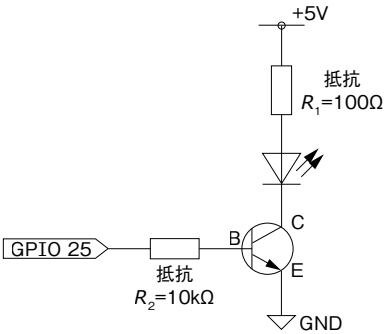

図4-20　青色LEDを明るく点滅させるための回路図

の)」といいます。トランジスタには3つの端子があり、それぞれ「コレクタ(C)」、「ベース(B)」、「エミッタ(E)」と呼ばれます。このトランジスタを、ここではスイッチとして用います。そのためには、GPIO 25からトランジスタのベース(B)へ流れる電流のh_{FE}倍(「直流電流増幅率」といいます)が、5Vピンからコレクタ(C)へ向かって流れる、という性質を用います。

図4-21に即して言えば、GPIO 25がHIGHのときはLEDに電流が流れてLEDが点灯し、LOWのときは電流が流れずLEDは点灯しない、ということになります。このとき、青色LEDは、抵抗を介して順方向降下電圧V_fよりも大きい5Vのピンに接続されていますから、明るく点滅させることができる、というわけです。

この回路をブレッドボード上で実現すると、たとえば次ページの図4-22のようになるでしょう。回路を作成したら、Thonnyでファイル04-02-led.pyを開いて実行します。青色LEDが明るく点滅することを確かめてください。

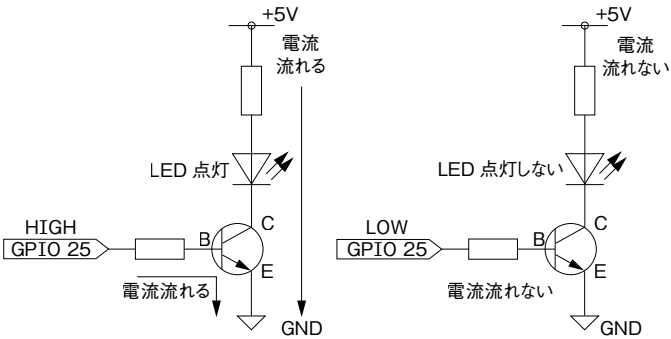

図 4-21　トランジスタはスイッチとして働く

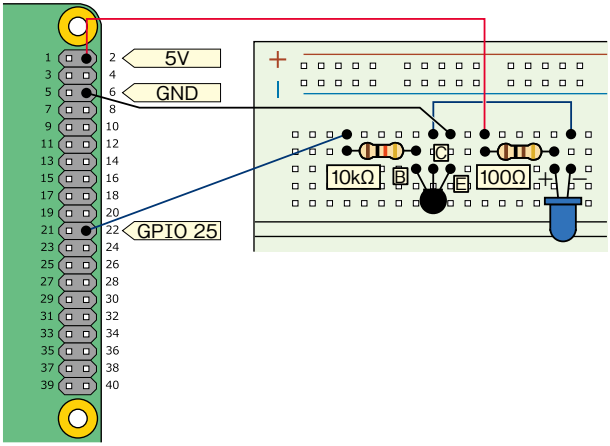

図 4-22　ブレッドボードによる回路の実現

4.4.2　抵抗値の決定方法

さて、図4-20（PDF 2ページ）では5VピンにはR_1=100Ωの、GPIO 25にはR_2=10kΩの抵抗を接続しました。このR_1とR_2の決定方法を学びましょう。

まずは、LEDが点灯している状態を考えます。この青色LEDに対し、V_f=3.4Vで流す電流が20mAであるとしましょう。また、このときトランジスタの性質より、トランジスタの「コレクタ(C)」と「エミッタ(E)」の間の電圧は、ほぼ0Vです。そのため、20mA（0.020A）の電流を流すためには、オームの法則よりR_1を次のように定める必要があります。

$$R_1=(5-3.4)/0.020=80\ \Omega$$

これより抵抗が大きく、入手しやすい抵抗として、100Ωを

選びました。

次に、R_2の計算です。LEDに0.020Aが流れるとき、R_2の抵抗には0.020/h_{FE} Aの電流が流れます。h_{FE}は、トランジスタの種類によりさまざまな値を取りますが、ここでは典型的な値として、h_{FE}=100を用いましょう。

トランジスタの「ベース(B)」と「エミッタ(E)」の間の電圧はほぼ0.6Vを取ることがやはりトランジスタの性質として知られており、さらにGPIO 25はHIGH状態のとき3.3Vですから、オームの法則より下記のようになります。

$$R_2=(3.3-0.6)/(0.020/h_{FE})=2.7/(0.020/100)=13500\ \Omega=13.5\ \mathrm{k}\Omega$$

これに近く、入手しやすい抵抗として10 kΩを選びました。

以上、青色LEDに20mAの電流を流すための回路と、その抵抗値の算出方法を学びました。この電流20mAはGPIOではなく、5Vピンを通して流れることに注意してください。青色LEDに限らず、10mA ～ 20mAの電流をLEDに流したい場合は、この方法を用います。ただし、R_1の計算はそのLEDのV_fに応じて計算し直す必要があります。

5.8 タクトスイッチでRaspberry Piをシャットダウン

書籍の**5.6**と**5.7**では、タクトスイッチの状態に応じてアクションを起こす演習を２つ紹介しました。ここでは３つ目の演習として、「タクトスイッチが押されたときにRaspberry Piをシャットダウンする演習」を紹介します。

用いる回路は、**5.6**と**5.7**のときと同じく書籍の118ページの図5-9で、実行するプログラムが記述されたサンプルファイルは、05-06-sw-poweroff.pyです。Thonnyでファイルを開いて実行します。スイッチが押されると、プログラムから`sudo poweroff`コマンドが実行され、Raspberry Piがシャットダウンされます。

このプログラムの７行目と８行目には、実行したいコマンドを次のように「args」という変数に格納し、subprocessモジュールのPopenクラスに渡すように記述されています。

```
7 args = ['sudo', 'poweroff']
8 subprocess.Popen(args)
```

では、このプログラムが必要となる場面について解説しておきましょう。これまで皆さんが行ってきたとおり、Raspberry Piの電源を切るには、Raspberry Pi OSのデスクトップからシャットダウンする必要がありました。この方法を実行するには、Raspberry Piに最低限ディスプレイとマウスが接続されている必要があります。

しかし、Raspberry Piをロボットの頭脳にする、などといった用途を考えた場合、ロボットの電源を切るたびに

Raspberry Piにディスプレイとマウスを接続するのは不便ですね。ここで紹介したプログラムを用いると、ディスプレイとマウスを接続しなくても、タクトスイッチを押すだけでシャットダウンすることができる、というわけです。

なお、このプログラムを実用するためには、このプログラムがRaspberry Piの起動とともに自動的に実行され、タクトスイッチの入力を常に待機した状態になっている必要があります。そうした「プログラムの自動実行の方法」については、**7.5**（PDF 11ページ）や**10.3.3**（書籍の285ページ）で紹介します。

6.5 半固定抵抗で音声のボリュームを変更する(要ネットワーク)

6章で学んだAD変換を用いた応用例として、半固定抵抗のつまみを音声のボリューム変更に用いてみましょう。この例は、書籍の**5.7**でMP3の音を鳴らせた人を対象とするため、ネットワークへの接続が必要になります。

用いる回路は、書籍の143ページの図6-5および144ページの図6-6の半固定抵抗を用いたものです。用いるプログラムが記述されたサンプルファイルは、06-03-volume.pyです。0から4095の値を、0%から100%というボリュームに変換し、amixerというコマンドでボリュームをセットするプログラムです。

プログラムを実行する前に、音声が再生されている状態にしておく必要があります。再生する音声ファイルは、**5.7**でも用いたtest.mp3にします。

ターミナルを起動して、06-03-volume.pyのあるディレクトリで音声を再生するコマンドを実行しますが、巻末の**付録B**でサンプルファイルをホームディレクトリに用意した場合と、「bluebacks」ディレクトリに用意した場合では、実行する内容が異なります。ホームディレクトリに用意した場合、`mplayer test.mp3`を実行します。「bluebacks」ディレクトリに用意した場合、前述のコマンドを実行する前に`cd bluebacks`というコマンドで「bluebacks」ディレクトリに移動する必要があります。そのあと、`mplayer test.mp3`を実行します。

上記コマンドにより音声が再生され、test.mp3の音が鳴るはずです。もし音が鳴らなかったら、**5.7**に戻り、適切なコマ

ンドに変更したり、イヤフォンやスピーカーの接続を見直してください。なお、音声の再生が終了してしまったら、ターミナル上でキーボードの［↑］キーを押してから［Enter］キーを押してください。もう一度同じコマンドが実行され、音が鳴ります。

では、音声が再生されている状態のまま、06-03-volume.pyをThonnyで開いて実行しましょう。半固定抵抗のつまみで、音声のボリュームが調整できるはずです。なお、2020年5月以降のOSで、Raspberry Pi上のイヤフォンジャックから音声を聞きたいが音が鳴らないという場合、まず06-03-volume.pyの57行目の「args = (略)」の左に「#」を記して無効化し、58行目の「#args = (略)」の先頭の「#」を消して有効化します。そしてファイルを保存してから実行する必要があります。

6.6 spidevを用いたSPI通信

6章で用いたプログラムでは、ADコンバータとRaspberry Piとの間の通信に「SPI通信」と呼ばれるものを用いていました。書籍では解説しませんでしたが、このSPI通信を実現しているのは、プログラムのファイル中に記述されているreadadc関数の実装です。

SPI通信の実現には、それ以外にもRaspberry Pi OSの「カーネルモジュール」と呼ばれるものを用いる方法があります。プログラミングに慣れている人にとっては、カーネルモジュールのほうが主流と言えるかもしれません。ここでは、その方法を紹介します。

●SPIモジュールの有効化

まず、Raspberry Pi OSでSPI用モジュールを有効にする設定を行います。**2.4.2**の図2-25（書籍の59ページ）で紹介した設定用アプリケーションで「インターフェイス(Interfaces)」タブを選択し、「SPI」の項目を「有効(Enabled)」にしてください。「OK」ボタンで設定用アプリケーションを終了します。

●プログラムの実行

6章で用いてきた**プログラム6-1**[*1]（書籍の145～147ページ）を、ここでインストールしたspidev用に変更したプログラムのサンプルファイルが、06-04-spidev.pyです。回路は、書籍の144ページの図6-6のものをそのまま用います。

ファイルをThonnyで開いて実行してみましょう。**プログラム6-1**を実行したときと同じ結果となるはずです。readadc関数を用いたプログラムより、spidevを用いたプログラムのほうが、若干高速に動作するというメリットがあります。

なお、spidevを用いた06-04-spidev.pyは、readadc関数を用いる06-01-print.pyや06-02-led.pyを実行したあと、そのまま実行しても正しく動作しません。一度Raspberry Piを再起動してから実行してください。そのため、spidevとreadadc関数は併用するのではなく、どちらを使うか決めて、それを使い続けるのがよいでしょう。

もし、この節を終えたあとspidevを用いる予定がない方は、設定用アプリケーションを再び起動して、SPIを「無効(Disabled)」に戻すのがよいでしょう。

[*1]プログラムが記述されたサンプルファイルは、06-01-print.pyです。

7.5 デジタル温度計用プログラムの自動実行（上級者向け）

書籍の**7.4**でデジタル温度計を作成しました。しかし、作成した状態のまま居間などで使うことを考えた場合、下記のような点で問題があります。

・Raspberry Piを起動したあと、「温度センサで読み取った値をLCDに表示するプログラム」を手動で実行しなければならない
・上記を実行するためには、Raspberry Piにキーボード、マウス、ディスプレイを接続することになり、これでは温度計というより、むしろ単なるPCにすぎない

もし、Raspberry Piが起動するときに「温度センサで読み取った値をLCDに表示するプログラム」が自動で実行されれば、Raspberry Piにキーボード、マウス、ディスプレイを接続することなくデジタル温度計が実現できます。その方法を本節で学びましょう。あらかじめ、書籍の171ページの図7-6の回路を作成しておきます。

●コマンドによるプログラムの起動

まず、どのようなコマンドでプログラムが実行されるかを知る必要があります。これは書籍の**7.3**で紹介したように、「温度センサで読み取った値をLCDに表示するプログラム」が記述されたサンプルファイル07-03-LCD-temp.pyが存在するディレクトリで`python3 07-03-LCD-temp.py`を実行すればよいのでした。

プログラムを自動実行する際には、このサンプルファイルが存在するディレクトリも含めて記述する必要があります。そのため、python3 /home/pi/07-03-LCD-temp.pyを実行します。もし、サンプルファイルが「bluebacks」ディレクトリにあればpython3 /home/pi/bluebacks/07-03-LCD-temp.pyを実行することになります。

●プログラムをRaspberry Pi起動時に自動実行

前述の内容をRaspberry PiのOSが起動するときに自動的に実行されるようにします。そのため、起動するときに実行されるファイルである/etc/rc.localに上記のコマンドを追加します。

実際に追加してみましょう。まずターミナルを起動し、sudo mousepad /etc/rc.localのコマンドでファイル/etc/rc.localを管理者権限のテキストエディタで開きます。なお、2019年7月以前のOSを用いている方は、コマンド中の「mousepad」を「leafpad」に置き換えて実行してください。サポートページにはどちらのコマンドも掲載します。

ファイル/etc/rc.localの末尾には「exit 0」という行があります。その行の手前に、前項で述べたプログラムを実行するためのコマンドを1行記述します。

```
python3 /home/pi/07-03-LCD-temp.py &
exit 0
```

1行追加する

サンプルファイルが「bluebacks」ディレクトリにある場合は、上記コマンドを「python3 /home/pi/bluebacks/07-03-

LCD-temp.py &」に変更してください。

末尾の「&」は、書籍の**10.3.3**で解説される、プログラムをバックグラウンドで実行するためのものです。ここでも必ず記述してください。なお、/etc/rc.localにより実行されるファイルは自動的に管理者権限になります。正しく記述できたら、ファイルを保存してからテキストエディタを終了しましょう。

以上が済んだら、Raspberry Piを再起動してみましょう。プログラムが自動的に実行されているはずです。温度センサに指を近づけるなどして、LCD上の温度の読みが変動することを確認しましょう。

温度が変動しない場合、それは以前のLCDの表示がそのまま残っているだけであり、プログラムが自動的に実行されていない可能性があります。もう一度ファイル/etc/rc.localの記述を見直してみましょう。

●タクトスイッチによるRaspberry Piのシャットダウン

プログラムが自動的に実行されていることを確認したら、今度は一旦Raspberry Piをシャットダウンし、キーボード、マウス、ディスプレイを外した上で再びRaspberry Piに電源を入れてみましょう。ここではネットワークケーブルも外して構いません。PCというよりはより「回路」らしい状態でデジタル温度計が機能することがわかるはずです。

ただし、この状態のままではRaspberry Piにシャットダウンの命令を与えることができません。そこで、次はこの部分を改善します。ひとまずキーボード、マウス、ディスプレイを接続し直して、これまでどおりRaspberry Piを操作できるようにしましょう。

キーボード、マウス、ディスプレイを接続せずにRaspberry Piの電源を切る方法は、PDFファイルの**5.8**に説明があります。ブレッドボード上に、書籍の118ページ図5-9の回路のタクトスイッチの部分だけを追加します。

タクトスイッチでRaspberry Piをシャットダウンするプログラムが記述されたサンプルファイルは、05-06-sw-poweroff.pyでした。これを自動実行するためのコマンドは、「python3 /home/pi/05-06-sw-poweroff.py &」となります。このコマンドは、サンプルを「bluebacks」ディレクトリに展開した場合は「python3 /home/pi/bluebacks/05-06-sw-poweroff.py &」となります。これを、前項のようにファイル/etc/rc.localの「exit 0」の1つ上の行に追記すれば希望の動作を実現できます。

●プログラムの自動実行を無効に戻す

最後に、プログラムを自動実行する設定を無効にして、もとの設定に戻す方法を解説しましょう。

キーボードとマウスを接続して、ファイル/etc/rc.localを管理者権限のテキストエディタで開き、自動実行用のコマンドの先頭に「#」を記述することでコメントアウトします。「exit 0」の行には手を付けずにそのまま残してください。

```
#python3 /home/pi/07-03-LCD-temp.py &
exit 0
```

自動実行用のコマンドの先頭に「#」を記述してコメントアウトする

ファイル/etc/rc.localを保存し、Raspberry Piを再起動

すると、プログラムが自動実行されない状態で起動します。

7.6 小型LCDにカタカナを表示する

7.3で用いたプログラム7-2（07-02-LCD.py）は実はカタカナも表示できるようなプログラムとなっています。その方法をここで紹介しましょう。

07-02-LCD.pyの82～83行目に、「#」によるコメントアウトで無効にされた下記の２行があります。

```
#s = chr(0xd7)+chr(0xbd)+chr(0xde)+chr(0xcd)+chr(0xde)
+chr(0xd8)+chr(0xb0)+' '+chr(0xca)+chr(0xdf)+chr(0xb2)
#write_string(s)
```

この２行はLCDに「ラズベリー　パイ」というカタカナを表示するためのものなのですが、これを有効にするために下記の変更を行ってみましょう。

・78行目のwrite_string('Hello World')の先頭に「#」を記述して無効にする
・その後、上記２行の先頭の「#」を削除して有効にする

その後、07-02-LCD.pyを上書き保存してから実行すると、LCDにはカタカナで「ラズベリー　パイ」と表示されます。「chr(0xd7)」の「0xd7」がカタカナ「ラ」の「文字コード」を表します。「0x」は16進法の数字の先頭につけるのでしたから、「ラ」の文字コードは16進法でd7ということになりま

す。このLCDモジュールで表示できる文字の文字コードの一覧が図7-8です。

図7-8を見ると、先ほど示した２行はカタカナ一文字一文字の文字コードをchr関数に渡し、それを「+」演算子で結合していることがわかります。ただし、空白文字だけは英数字と同じ扱いで一重引用符「'」で囲って表現しています。一旦変数sに格納してから、関数write_stringに渡していることにも注意しましょう。

以上と同じ方法を用いると、図7-8に示されたさまざまな文字を表示できますので試してみてください。

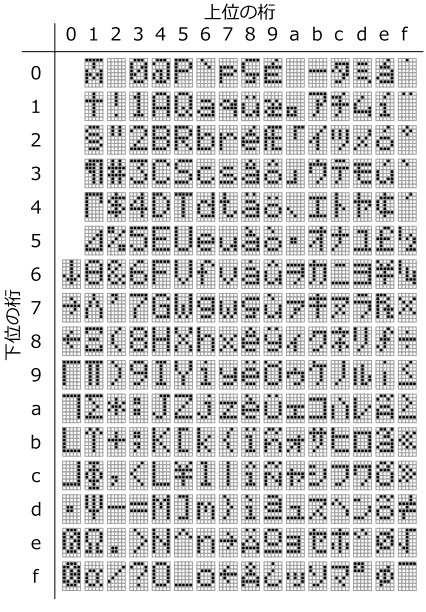

図 7-8　LCD に表示できる文字のコード表

8.6.3 サーボモーター２個を同時に用いる

8.6.2にてサーボモーター１個を精度の高いハードウェアPWM信号で制御できるようになりました。ハードウェアPWM信号は同時に２つ出力できますので、その方法をここで紹介します。１つの場合とほとんど変わりませんので、手順のみを簡単に解説します。

必要な回路は次ページの図8-14です。書籍の215ページの図8-13と比べると、操作用の半固定抵抗とサーボモーターがそれぞれ１つずつ増えていることがわかります。また、２つのサーボモーターはGPIO 18と19にそれぞれ接続されているのも見て取れるでしょう。

この回路を動かすためのプログラムが08-06-2servos.pyです。**8.6.2**で学んだように、まだ起動していなければ、コマンド`sudo service pigpiod start`でpigpiodを起動しておきます。そして、Thonnyで08-06-2servos.pyを開いて実行しましょう。うまく実行できると、２つの半固定抵抗のつまみに応じて、２つのサーボモーターの角度が変化することがわかるでしょう。

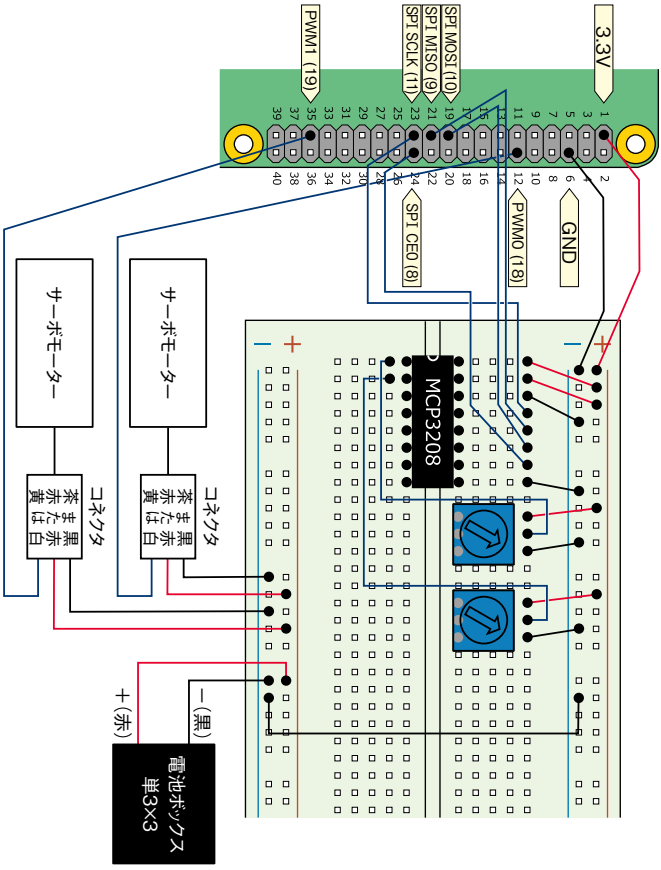

図 8-14　２つのサーボモーターの角度制御用回路

9.7 ブラウザによるサーボモーターの制御

この節では、８章で用いたサーボモーターをブラウザから制御します。８章では半固定抵抗によりサーボモーターを動かしましたが、本章では**9.5**と同様にブラウザのスライダを利用します。

9.7.1 動作させるための手順

サーボモーターを制御するための回路は次ページの図9-14です。２つのサーボモーターを同時に操作できる回路としましたが、サーボモーターを１個しか接続していない状態でも正常に動作します。

サーボモーターの角度をブラウザで変更するためには、**9.2.5**でコピーしたサンプルファイル一式に含まれている「/usr/share/webiopi/htdocs/bb/07/」ディレクトリ内にあるファイルを用います。設定ファイルを、この演習用のPythonプログラムを読み込むよう設定してからWebIOPiを起動します。

まず、設定ファイル/etc/webiopi/configを管理者権限のテキストエディタで開き*2、その中の［SCRIPTS］セクションに、以下のように１行追加します。**9.3**以降の演習で追加したmyscriptの設定が残っている場合、既存の行に記された番号（たとえば「06」）の部分を「07」に書き換えるだけで構いません。

```
#myscript = /home/pi/webiopi/examples/scripts/macros/script.py
myscript = /usr/share/webiopi/htdocs/bb/07/script.py
```

［SCRIPTS］セクションに１行追加する

*2「●設定ファイルの変更について」（241ページ）参照

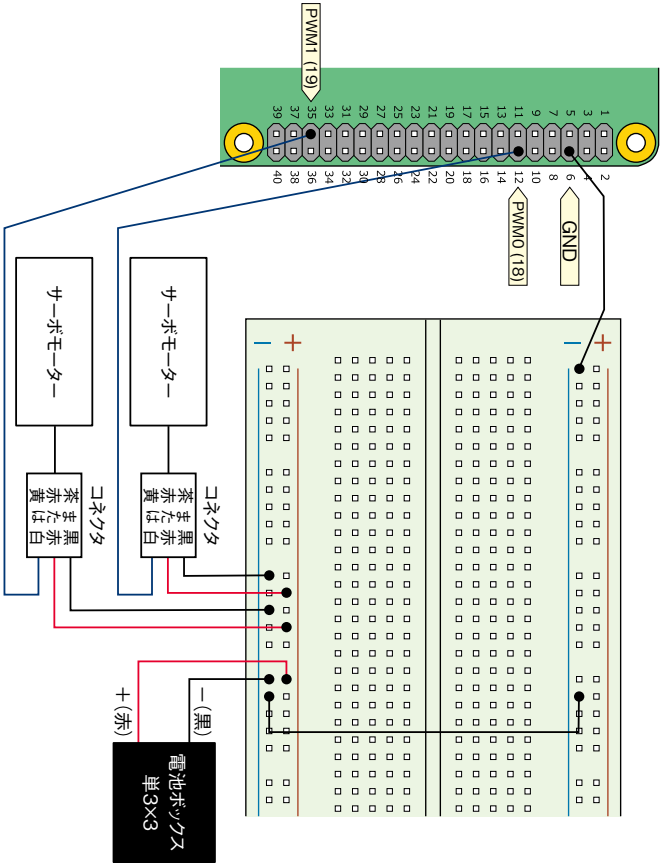

図 9-14　サーボモーター２個を制御するための回路

追加したら、設定ファイルを保存してからテキストエディタを閉じます。

そして、**8.6.2**で学んだように、コマンド`sudo service pigpiod start`でpigpiodを起動します。それから、**9.2.3**で学んだように、`ps ax | grep webiopi`コマンドでWebIOPiがすでに起動していないか確認してから、`sudo service webiopi start`で起動してください。すでに起動されている場合、`sudo service webiopi stop`コマンドで停止してから起動しなおしましょう。場合によってはpigpiodとwebiopiの両方を停止し、それぞれ順に起動しなおさなければならないことがあります。

9.7.2 動作確認

動作確認するために、PCやスマートフォンのブラウザで「http://192.168.1.3:8000/bb/07/」にアクセスしてください（IPアドレスの部分は、皆さんのRaspberry Piに割り当てられたもの[*3]を記述します）。「http://」を省略せず、末尾の「/」（スラッシュ）も忘れずに記述しましょう。

正しくページが開かれると、図9-15のようなページが現れます。設定によってはパスワードを聞かれますので、**9.2.4**で

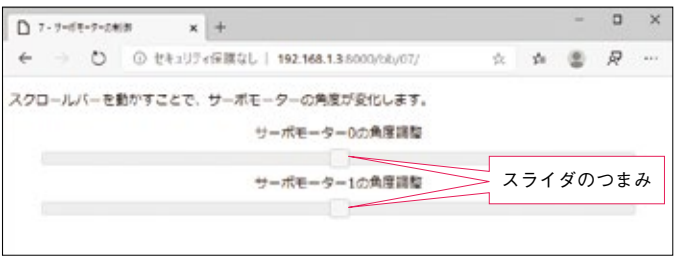

図9-15　ブラウザでサーボモーター２個を制御するためのサンプル画面

[*3]「●Raspberry PiのIPアドレスを調べよう」（233ページ）参照

学んだように入力してください。なお、前ページの図9-15のアドレス末尾の「/」がブラウザによっては省略して表示されることがありますが、その場合もアドレス末尾の「/」の入力は必要ですのでご注意ください。2つのスライダがあり、それぞれGPIO 18とGPIO 19に接続したサーボモーターの角度に対応します。

9.5と同様、スライダのつまみをつかんで移動する際、スマートフォンやタブレットのブラウザではつまみをタッチでつかみにくい場合があります。そのような場合、スライダのバーをタッチしてつまみを移動すると操作しやすいでしょう。

以下ではこの実現方法の概略を解説します。

9.7.3 サンプルの解説

9.3.3と同様、このサンプルを実現しているのは、「/usr/share/webiopi/htdocs/bb/07/」ディレクトリにある4つのファイルです。

「index.html」では下記のようにスライダ用の場所を確保しています。`slider0_servo`と`slider1_servo`が2つのスライダに対応します。**9.5**と同様にjQuery UIに含まれるものを用います。

```
<div id="text0_servo">サーボモーター0の角度調整</div>
<div id="slider0_servo"></div>
<div id="text1_servo">サーボモーター1の角度調整</div>
<div id="slider1_servo"></div>
```

実際にスライダを配置しているのはJavaScriptファイル

「javascript.js」内部の下記の部分です。変数にセットされた初期設定を渡してスライダを作成しています。具体的にはスライダの最小値（min）は0、最大値（max）は20、刻み幅（step）は1、初期値（value）は10です。さらに、スライダの値が確定されたとき（change）、スライダを移動している最中（slide）に呼ばれるイベントハンドラsliderHandler0も渡しています。これと同様の命令がslider1_servoに対しても実行されています。

```
$( "#slider0_servo" ).slider({
     min: sliderMin,
     max: sliderMax,
     step: sliderStep,
     value: sliderValue0,
     change: sliderHandler0,
     slide: sliderHandler0
 });
```

イベントハンドラは下記のように定義されています。スライダの現在の値（ui.value）をスライダの最大値（sliderMax、実際は20）で割った0.0から1.0の値をPythonスクリプトscript.pyに送ることになります。なお、サーボの向きを逆にしたい場合「ratio = 1.0 - ratio;」という行を無効に、という指示がありますが、先頭に「//」をつけて「// ratio = 1.0 - ratio;」とすることで無効化できます。２つのサーボ用に２つ同じ命令がありますので、２つとも無効化して保存してください。

```
var sliderHandler0 = function(e, ui){
    var ratio = ui.value/sliderMax;
    // サーボの回転の向きを逆にしたい場合次の行を無効に
    ratio = 1.0 - ratio;
    webiopi().callMacro("setHwPWM", [0, ratio, commandID++]);
};
```

なお、得られたratioを「webiopi().callMacro("setHwPWM", [0, ratio, commandID++]);」という命令でPythonスクリプトscript.pyに送っています。このイベントハンドラはslider0_servo用ですのでスライダのIDとして0も一緒に送っています。slider1_servo用には1を一緒に送ります。また、commandIDは**9.6.3**と同様の理由で追加しています。

送られたratioは、script.py内で下記のようにPWMのデューティ比としてセットされます。IDが0の場合はGPIO 18のPWMに、IDが1の場合はGPIO 19のPWMにデューティ比がセットされていることがわかります。

```
38 @webiopi.macro
39 def setHwPWM(servoID, duty, commandID):
40     id = int(servoID)
41     duty = getServoDutyForWebIOPi(float(duty))
42     if id==0:
43         pi.hardware_PWM(18, 50, duty)
44     elif id==1:
45         pi.hardware_PWM(19, 50, duty)
```

getServoDutyForWebIOPiは同じくscript.py内で下記のように定義されています。これは**8.6.2**で解説した内容とほとんど同じですので解説は省略します。

```
5  def getServoDutyForWebIOPi(val):
6      val_min = 0.0
7      val_max = 1.0
8      servo_min = 35000  # 最小デューティ比3.5%を10000倍して格納
9      servo_max = 100000 # 最大デューティ比10%を10000倍して格納
10
11     duty = int((servo_max-servo_min)*(val-val_min)/(val_
   max-val_min) + servo_min)
12     return duty
```

紙面スペースの都合、プログラムの11行目は複数行になっています。
本来１行のもので、途中に改行は入りません。

10.5 キャタピラ式模型に搭載したカメラを上下に動かす（オプション）

10.5.1 準備

10.4にてキャタピラ式模型にカメラを搭載しましたが、本節ではさらにそのカメラをサーボモーター１個により上下に動かしてみましょう。カメラの左右方向の移動はすでにキャタピラ式模型の移動によって実現されていますので、より広い範囲を映像で確認できるようになります。

そのためには、まず**10.4**の動作確認が済んでいる必要があります。

10.5.2 組み立て

図10-11がカメラにサーボモーターを取り付けた様子です。カメラにアームが取りつけられており、そのアームがサーボモーターにより上下に動きます。

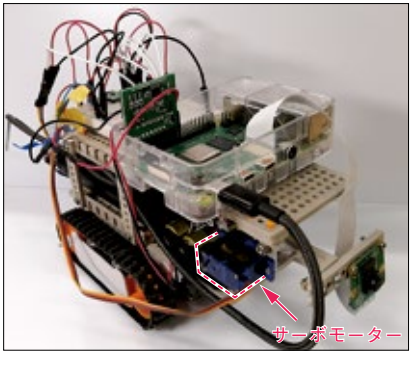

図 10-11　カメラをサーボモーターで操作できるようにしたキャタピラ式模型

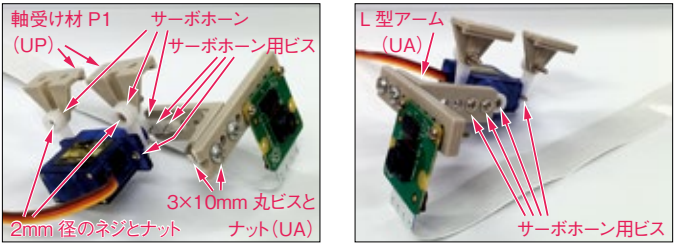

図 10-12　カメラとサーボモーターを固定する方法

サーボモーター、アーム、カメラのみを取り外して表示したのが図10-12です。これまでどおり、可能な限りTAMIYAのキットで実現しています。図中で「UP」と記されているのはユニバーサルプレートの付属品であり、「UA」と記されているのはユニバーサルアームセットの付属品であることを示しています。サーボホーンとサーボホーン用ビスはサーボモーターの付属品です。ここでは書籍の65ページの秋月電子通商のパーツセットに含まれるサーボモーター SG90を用いました。2つのサーボホーンをニッパでカットして整形し、軸受け材P1から吊り下げる方式でサーボモーターを固定しています。なお、サーボホーン用ビスは4つ必要ですが、1つのサーボモーターの付属品では足りないことがあります。2つめを購入してそちらの付属品を用いるか、ネットショップなどで取り扱われている「エスコ　M2×8mmナベ頭タッピングビス」(40本入り）などを用いると、代用できます。

このように、カメラとサーボモーターの取り付けは、耐久性やメンテナンス性を考えるとすべてねじ止めするのが理想です。しかし、試してみるとわかりますが、これは工夫が必要な作業です。どうしても難しい場合は両面テープなどによる仮止

めでもよいと思います。

10.5.3 動作させるための手順

作成すべき回路は次ページの図10-13です。これは、書籍の288ページの図10-6の回路にサーボモーターを追加しただけのものです。

ブラウザからこの回路にアクセスするためには、**9.2.5**でコピーしたサンプルファイル一式に含まれている「/usr/share/webiopi/htdocs/bb/08/」ディレクトリ内にあるサンプルファイルを用います。以下のように、設定ファイルを、演習用のPythonプログラムを読み込むように設定してからWebIOPiを起動しましょう。

まず、設定ファイル/etc/webiopi/configを管理者権限のテキストエディタで開き[*4]、この中の［SCRIPTS］セクションに、次のように1行追加します。過去の演習の設定が残っていますので、既存の行に記された番号（たとえば「07」）の部分を「08」に書き換えるだけで構いません。

```
#myscript = /home/pi/webiopi/examples/scripts/macros/script.py
myscript = /usr/share/webiopi/htdocs/bb/08/script.py
```

［SCRIPTS］セクションに1行追加する

追加したら設定ファイルを保存してからテキストエディタを閉じます。

そして、**8.6.2**で学んだように、コマンド`sudo service pigpiod start`でpigpiodを起動します。それから、**9.2.3**で学んだように、`ps ax | grep webiopi`コマンドで

[*4]「●設定ファイルの変更について」（241ページ）参照

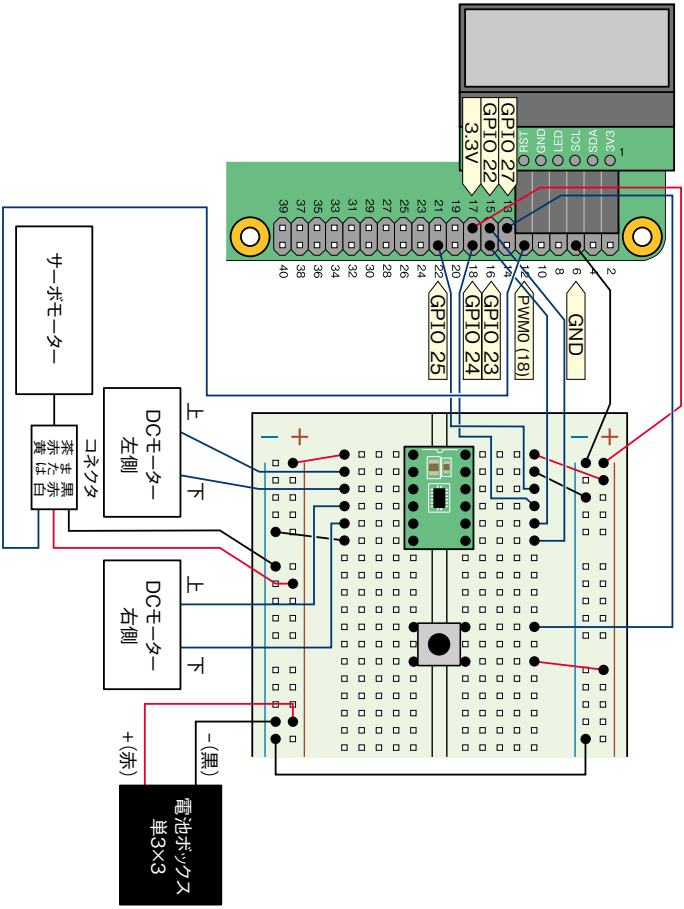

図 10-13　カメラをサーボモーターで操作するための回路図

WebIOPiがすでに起動していないか確認してから、

`sudo service webiopi start` で起動してください。すでに起動

されている場合、`sudo service webiopi stop`コマンドで停止してから起動しなおしましょう。場合によってはpigpiodとwebiopiの両方を停止し、それぞれ順に起動しなおさなければならないことがあります。

10.5.4　動作確認

動作確認するために、PCやスマートフォンのブラウザで「http://192.168.1.3:8000/bb/08/」にアクセスしてください（IPアドレスは皆さんのRaspberry Piに割り当てられたものを記述します）。「http://」を省略せず、末尾の「/」（スラッシュ）も忘れずに記述しましょう。

操作ページが正しく動作すれば、図10-14のような画面が現れます。書籍の297ページの図10-9と比べると、カメラから

図 10-14　ブラウザによる操作用の画面

の映像の右側にスライダが縦に表示されており、これを動かすことでサーボモーターを上下に操作することができます。

なお、このプログラムを自動起動するためには、pigpiodを自動起動する必要があります。そのためのコマンドは`sudo systemctl enable pigpiod`です。**9.2.3**で学んだWeb IOPiの自動起動のコマンドと似ていますね。pigpiodの自動起動をやめたい場合のコマンドは`sudo systemctl disable pigpiod`です。

付録 E IPアドレスを用いずにRaspberry Piにアクセスする

書籍の9章や10章では、ブラウザからRaspberry Piにアクセスする際に、ルーターなどからRaspberry Piに割り振られたIPアドレスを用いました。Raspberry PiのIPアドレスがたとえば192.168.1.3だった場合、下記のようなアドレスを用いました。

http://192.168.1.3:8000/bb/01/

しかし、この方法はifconfigコマンドで事前にIPアドレスを調べておく必要があり、やや面倒でした。

そこで、IPアドレスを用いずにRaspberry Piにアクセスする方法を紹介します。なお、この方法を主に利用できるのはmacOS、iPhone、iPadです。Androidスマートフォンでは現時点ではこの方法を利用できません。また、iTunesがインストールされているWindowsでもこの方法は利用可能です。iTunesに含まれるBonjourというソフトウェアの機能を利用するためです。ただし、iTunesとBonjourを一度削除して再インストールしなければ動作しなくなることが時々あります。

利用する方法は簡単で、ブラウザのアドレス欄にたとえば下記のように入力すればOKです。

http://raspberrypi.local:8000/bb/01/

すなわち、IPアドレス「192.168.1.3」などの代わりに

「raspberrypi.local」という記法が使える、というわけです。これは、Raspberry Pi OS上で動作しているAvahiというソフトウェアの働きによります。

なお、同じネットワーク内でAvahiが起動したRaspberry Piが２台以上あると、同じ名前「raspberrypi」のマシンが２つ以上ある状態になり、名前の衝突が起こります。その場合、２台目以降のRaspberry Piには「raspberrypi-2.local」などの名前が自動的につけられますのでご注意ください。

付録F　抵抗のカラーコード

抵抗には色付きの帯が描かれています。これは抵抗の大きさを表しています。ここではその読み方を紹介します。

抵抗の帯の本数は、４本のものや５本のものなどがありますが、多く見かけるのは４本のものです。判別する際は、図F-1のように、帯の間隔が広いほうや、端の帯が太いほうを右に置きます。帯の太さや間隔がほぼ左右対称に見えることもありま

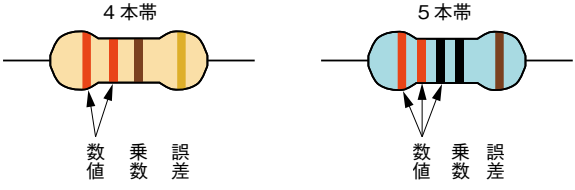

図 F-1　抵抗上の帯の意味

すが、入手しやすい抵抗では、片方の端が金色のことが多いので、その場合は金色を右に置くようにします。そうすると、数値、乗数、誤差の意味を持つ帯が図のように配置されます。

それぞれの色の意味は表F-1のとおりです。

表 F-1　抵抗の色の意味

色	数値	乗数	誤差
黒	0	1	―
茶	1	10^1	±1%
赤	2	10^2	±2%
オレンジ	3	10^3	±0.05%
黄	4	10^4	―
緑	5	10^5	±0.5%
青	6	10^6	±0.25%
紫	7	10^7	±0.1%
グレー	8	10^8	―
白	9	10^9	―
銀	―	10^{-2}	±10%
金	―	10^{-1}	±5%
無色	―	―	±20%

ここでは、本書でよく用いる330 Ωと10 kΩの抵抗の読み方を記しておきましょう。

・330 Ω

オレンジ、オレンジ、茶、金

→33×10^1 Ω ± 5% = 330 Ω（誤差 ± 5%）

・10 kΩ
　茶、黒、オレンジ、金
→$10 \times 10^3\ \Omega \pm 5\% = 10 \times 1000\ \Omega \pm 5\% = 10\ \mathrm{k}\Omega$（誤差±5%）

最後の計算では、1000 Ω =1 kΩ という関係を用いています。

付録 G　入手しやすいI2C接続のセンサ用サンプルファイル

7章で述べたように、I2C接続のセンサの値を読むプログラムを書くことは、プログラムの学び始めのレベルではなかなか大変です。そこで、入手しやすいセンサのためのプログラムを記述済みのサンプルファイルに含めました。表G-1はその一覧

表 G-1　記述済みのサンプルファイルで用いることのできるI2C 接続センサ

種類	搭載チップ	型番など	サンプルファイル名	備考
温度センサ	TMP102	千石電商（SEN-13314）	07-04-temp.py	温度を計測する
加速度センサ	LIS3DH	秋月電子通商（K-06791）	07-05-acc.py	センサを動かしたときの加速度（速度の変化）を計測する
	ADXL345	秋月電子通商（M-06724）	07-06-acc.py	
加速度、ジャイロ、磁気センサ	BMX055	秋月電子通商（K-13010）	07-07-acc-gyro-mag.py	加速度と角速度と磁気を計測する
大気圧、温度センサ	LPS25HB	秋月電子通商（K-13460）	07-08-pressure-temp.py	大気圧（ヘクトパスカル）と温度を計測する

です。

本書では、各センサの使用法の解説は割愛しています。**7.2**で扱った温度センサの演習がヒントになりますし、プログラム中のコメントに回路の接続方法などが記されていますので、参考にしてください。

3章から10章の参考文献

以下で紹介する3章以降の参考文献は、すべて英語で書かれたウェブサイトとなっています。英語で書かれたウェブサイトを参考にした理由は、Raspberry Piは英国で開発されたものであり、網羅的な情報に関しては海外に一日の長があるからです。

なお、以下で紹介するウェブサイトは、PDFファイル上ではリンクとして掲載しています。実際にウェブサイトをご覧になるときは、URL上をクリックしてください。

3章

Raspberry PiのGPIOなどに関する情報は、次のサイトにまとめられています。

・「GPIO」
https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md

4章から5章

GPIOをPythonで制御するためのRPi.GPIOモジュールの使い方は、次のページの「Examples」という箇所にまとめられています。

・「RPi.GPIO Python Module」
https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

6章

spidevを用いずにADコンバータを用いる方法は、Adafruit

の次のページを参考にしています。

・「Analog Inputs for Raspberry Pi Using the MCP3008」
https://learn.adafruit.com/reading-a-analog-in-and-controlling-audio-volume-with-the-raspberry-pi/overview

spidevを用いるための情報は、次のページが参考になります。

・「py-spidev」
https://github.com/Gadgetoid/py-spidev

7章

Pythonから I2Cを用いる方法は、次のページにまとめられています。

・「Using the I2C Interface」
https://www.raspberry-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2

8章

PWM信号を生成するための「精度の低い方法」は、4章から5章の参考文献で紹介した「RPi.GPIO Python Module」ページのPWMページで解説されています。

・「Using PWM in RPi.GPIO」
https://sourceforge.net/p/raspberry-gpio-python/wiki/PWM/

pigpioを用いた精度の高いハードウェアPWM信号を生成する方法は次のページにまとめられています。

・「Python Interface」
http://abyz.me.uk/rpi/pigpio/python.html

9章から10章

WebIOPiについての情報は、公式ページがまず役立ちます。

・「WebIOPi -The Raspberry Pi Internet of Things Toolkit」
http://webiopi.trouch.com

あとはソースコードに含まれるサンプルコードやwebiopi.jsというファイルが参考になるでしょう。

また、Raspberry Pi用カメラモジュールの映像を配信する方法は、次のページを参考にしました。

・「Raspberry Pi camera board video streaming」
https://miguelmota.com/blog/raspberry-pi-camera-board-video-streaming/

なお、ここに掲載されているウェブサイトのURLは、すべて2020年5月時点のものです。皆さんがご覧になる際は、変更されている可能性があります。あらかじめご了承ください。