

IoT組込み開発



株式会社NEUGATE

Raspberry Pi Imagerのダウンロード

① Raspberry Pi公式[ダウンロードページ](#)から、Windows版をダウンロードしましょう。

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. [Watch our 40-second video](#) to learn how to install an operating system using Raspberry Pi Imager.

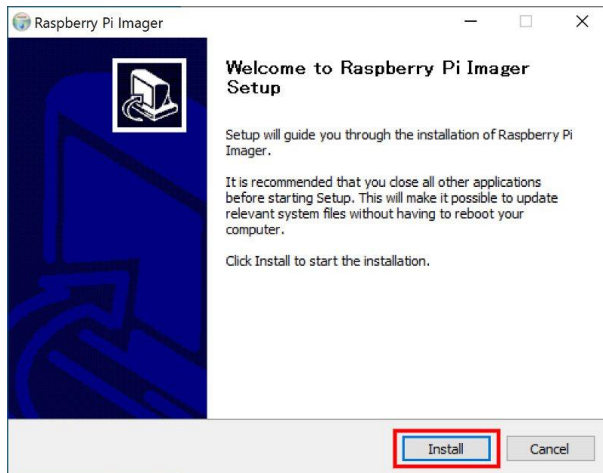
Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Windows](#)

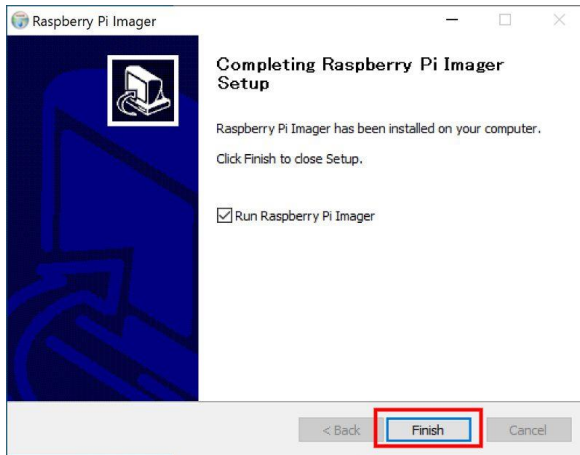
[Download for macOS](#)

[Download for Ubuntu for x86](#)

② ダウンロードしたファイルをダブルクリックすると、インストーラーが起動しますので、「Install」をクリックします。



③少し待つと、終了画面が表示されるので、「Finish」をクリックしてインストール完了です。スタートメニューから起動できるようになります。



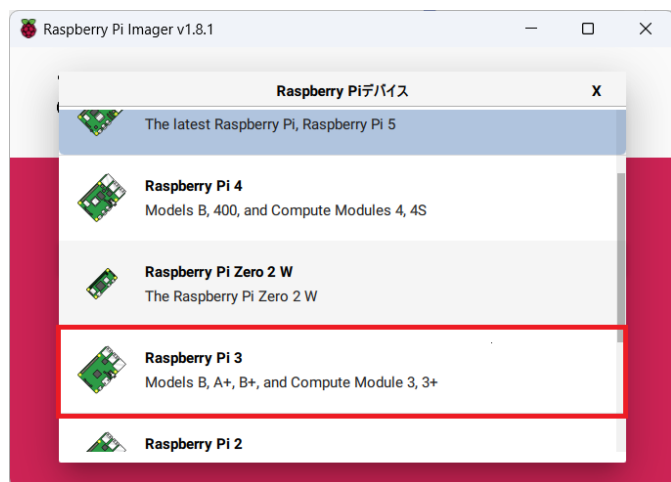
このあと、Raspberry Pi Imagerを起動してSDカードにRaspberry Pi OSをインストールします。

Raspberry Pi OSをSDカードにインストール

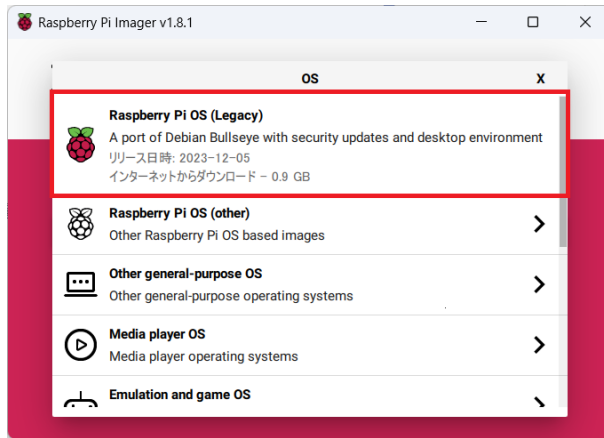
①PCにメモリーカードを挿入した後でImagerを起動すると、以下のような画面が表示されます。



②まず、「デバイスを選択」をクリックします。今回は「Raspberry Pi 3」を選択します。



③次に「OSを選択」をクリックして、1番上にあるRaspberry Pi OS (Legacy)を選択してください。

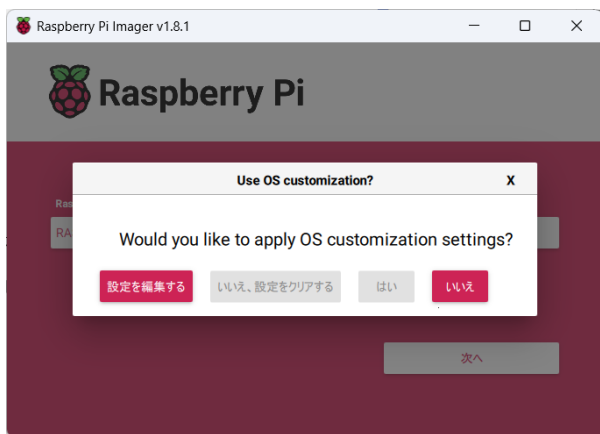


④次に「ストレージを選ぶ」をクリックします。挿入したSDカードが表示されるので選択してください。





⑤「デバイスの選択」「OSの選択」「ストレージの選択」が完了したら、「次へ」をクリックすると、下記の画面が表示されるので、「設定を編集」をクリック。



⑥「サービス」のタブを開きます。



「SSHを有効化する」にチェックをいれ、ユーザー名とパスワードを設定してください。ユーザー名とパスワードは講師の指示に従ってください。入力したら、「保存」をクリックします。



⑦「一般」のタブでユーザ名とパスワードを設定するよう求められるので、講師の指示に従って設定してください。

OS Customization

一般 サービス オプション

☐ ホスト名: raspberrypi .local

☒ ユーザー名とパスワードを設定する

ユーザー名: tanak

パスワード:

☐ Wi-Fiを設定する

SSID: NEUGATE_5F_2G

パスワード:

☐ パスワードを見る ☐ ステルスSSID

Wifiを使う国: GB

☐ ロケール設定をする

タイムゾーン: Asia/Tokyo

キーボードレイアウト: us

保存

⑧「はい」→「はい」をクリックすると書き込みが始まります。

Raspberry Pi Imager v1.8.1

Raspberry Pi

Raspberry Piデバイス OS ストレージ

RASPBERRY PI 3 RASPBERRY PI OS (LEGACY) SDHC CARD

書き込み中... 0%

書き込みをキャンセル



Raspberry Piを他のPCから操作する①

TeraTerm(テラターム)とは、WindowsOSで動作するターミナル・エミュレータの一種です。

TelnetやSSH、TCP / IP、シリアル接続をサポートしていて、ネットワーク上の機器にアクセス・操作することができます。

サーバー、ルーター、スイッチ、ファイアウォールなど、様々な種類の機器へリモート接続して、コマンド実行やファイル転送を行えます。

①「Tera Term 窓の杜」で検索してタイトルをクリック

 <https://forest.watch.impress.co.jp> > software > utf8teraterm 

「Tera Term」定番のターミナルエミュレーター - 窓の杜

Tera Termのダウンロードはこちら Telnetとシリアル接続に対応したターミナルエミュレーター「Tera Term Pro」を、多くの開発者の手で拡張したバージョン。

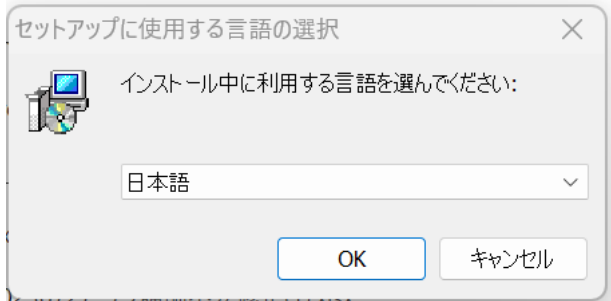
②「窓の杜からダウンロード」(v4系統) をクリック

Tera Term	
 Tera Term v4.106 (21/06/02) インストールアプリ	窓の杜からダウンロード 
定番のターミナルエミュレーター 無料 対応環境 : 64bit版を含むWindows 2000/XP/Vista/7/8/8.1/10および Windows Server 2003/2008 R2/2008 R2/2012/2012 R2	ファイルサイズ 12.2MB



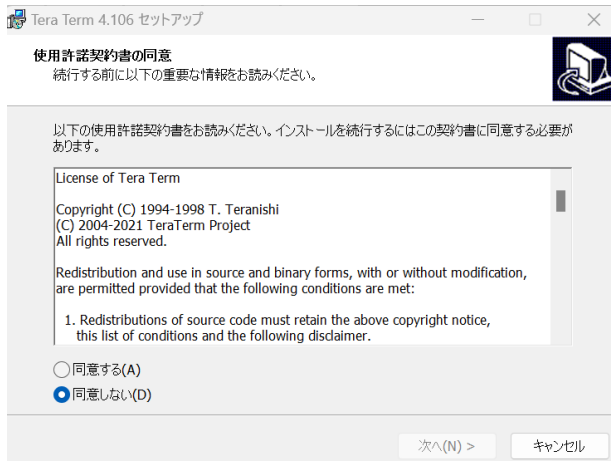
Raspberry Piを他のPCから操作する②

③ダウンロードしたexeファイルを実行→「はい」を選択



日本語を選択

④「同意する」を選択して「次へ」



この後は、「次へ」を選択し続けて「インストール」まで進みます。
インストール終了後はPCを再起動してください。



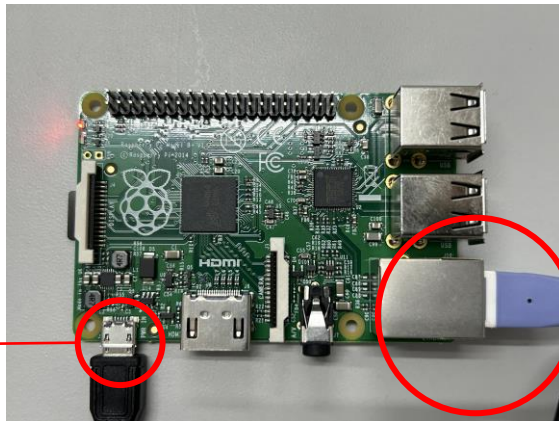
Raspberry Piを他のPCから操作する③

Raspberry PiにRaspberry Pi OSを書き込んだSDカードを差し込みます。



SDカードSLOT

Raspberry Piを電源に接続すると緑のランプが点滅し、OSの読み込みが始まります。※1~2分ほどかかります。



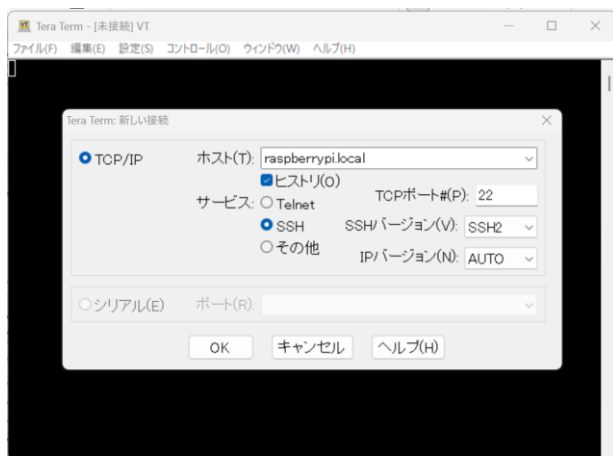
電源

LANポート



Raspberry Piを他のPCから操作する④

PCとRaspberry PiをLANケーブルで接続し、Tera Termを起動します。



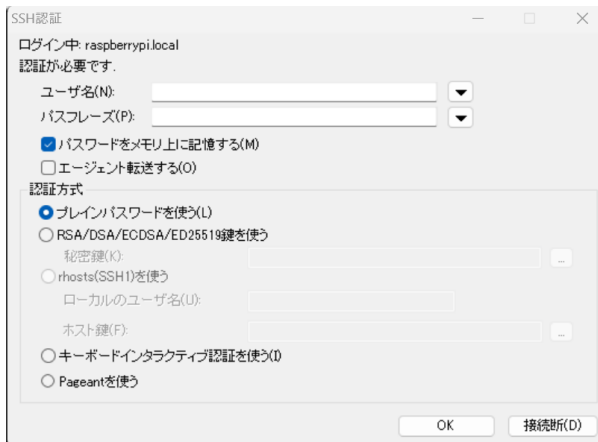
ホストに「raspberrypi.local」と打ち込んで、「OK」を押します。



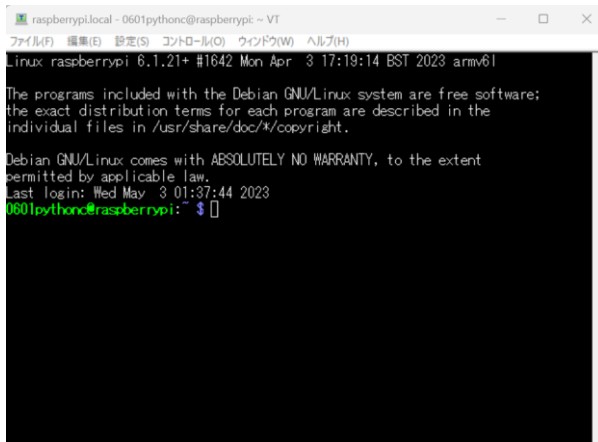
「続行」を押します。



Raspberry Piを他のPCから操作する⑤



Raspberry Pi Imager でOSを書き込む際に設定したユーザー名とパスワードを書き込んで「OK」を押します。



この画面になれば、無事OSが起動しています。
※この画面にならないければOSの読み込みが不十分だと思われます。電源につなぎなおして同様の手順を踏んでください。

Linuxコマンド①

Raspberry Pi OSはLinuxディストリビューションであるDebianがベースです。サーバーなどで広く使用されるLinuxのコマンドに触れてみましょう。

date:日付を表示

man: + コマンド名でマニュアルを呼び出す。

pwd:現在作業中のディレクトリを表示(print working directory)

cd :ディレクトリを移動(change directory)

ls :現在のディレクトリにあるファイルを表示(list segments)

echo:引数を出力

cat :ファイルの内容を表示

rm :ファイルやディレクトリを削除※かならずファイル名を指定すること(remove)

mkdir:ディレクトリを作る(make directory)

touch:空のファイルを作る

rmdir:ディレクトリを削除する※中身があるディレクトリは削除不可
(remove directory)

cp :コピーする

①ファイル→ファイル 新しいファイルor上書き

②ファイル→ディレクトリ ディレクトリの中に新しいファイル

③ディレクトリ→ディレクトリ 相手のディレクトリの中にコピー

※中身ごとコピーするときは"-r"のオプションが必要

Linuxコマンド②

mv: ファイルの移動

mv file1.txt test1 「file1.txtをtest1に移動」

※移動先のディレクトリに同じ名前のファイルがあるときは上書き

リネームとしても使える。

mv file1.txt file2.txt 「file1.txtをfile2.txtにする」

※すでにfile2.txtが存在する場合は上書きになるので注意

chmod: 権限(パーミッション)の変更

ls -l: 詳細表示を実行すると次のような行が表示されます。

 - rwxr-xr-x. 1 root root 62200 11月 17 2020 /bin/date

- rwxr-xr-x : 先頭はファイルタイプ、後ろは3桁ずつ

①rootユーザー ②rootグループに所属するユーザー

③root以外のユーザー

の権限を表しています。

r: 読み取り w: 書き込み x: 実行を表し、3桁ごとに8進数で表されます。この例でいうと、①rootユーザー : $4+2+1=7$

②rootグループに属するユーザー : $4+0+1=5$ ③root以外のユーザー : $4+0+1$ となります。ここで「chmod 567 ファイル名」を実行すると、パーミッションは「-r-xrw-rwx」に変更されます。

Linux上でファイルを編集してみよう！

「vi ファイル名」(存在しないファイル名の場合は新規作成)を実行することによってvimというLinuxのテキストエディタが起動します。起動時はノーマルモードの状態でモードを切り替えながら作業を進めていきます。書き込むときは「i」を入力してインサートモードに移行します。ここでは書き込むことしかできません。文字を削除したり、修正のため任意の場所にカーソルを移動させたいときは必ずノーマルモードに戻る必要があります。「Esc」でノーマルモードに戻ります。

①カーソルを消したい文字に移動して「x」

②途中から書きたい→カーソルを移動して

「i」:カーソルの左からインサートモード

「a」:カーソルの右からインサートモード

編集を終了するときは、「Esc」でノーマルモードに戻った後「:」でコマンドラインモードに移行します。

①wq :保存して終了

②q! :保存せず終了

演習：C言語で「Hello World!」と出力する「hello.c」ファイルをvimで作成し、実行してください。

※実行するときは、ファイルが存在するディレクトリまで移動して

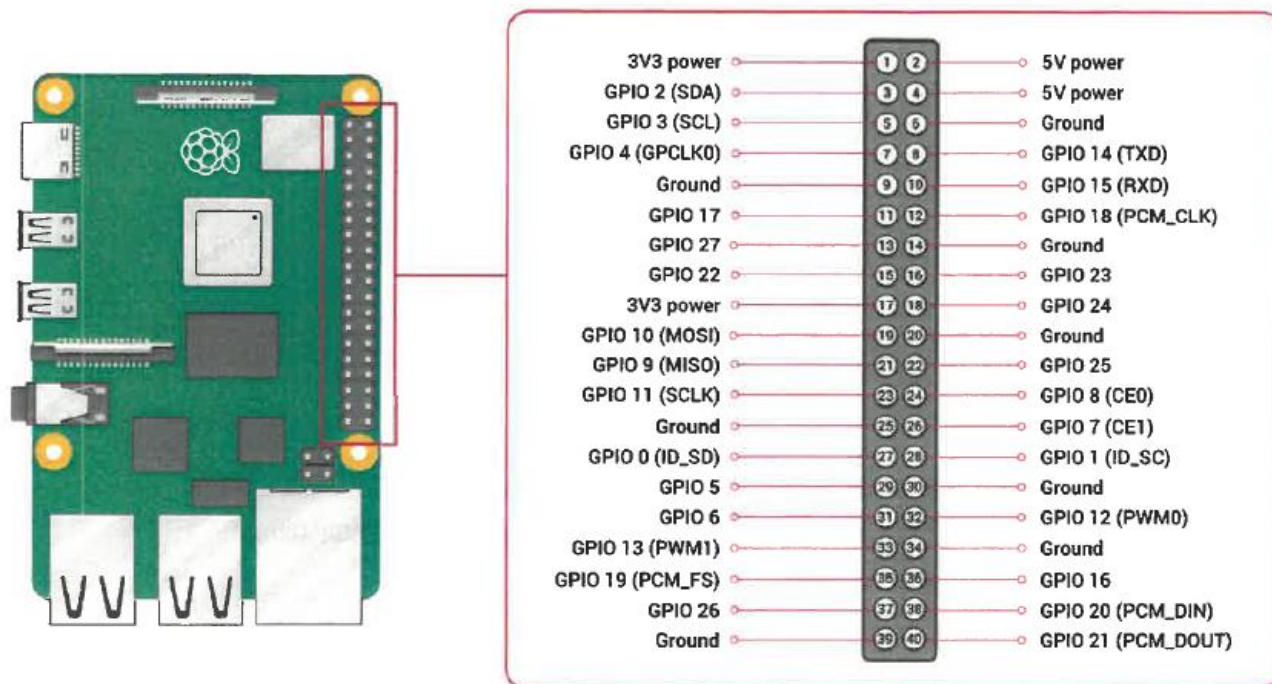
①「gcc -o hello hello.c」でコンパイル

②「./hello」

で実行できます。

電気回路を構築してLEDを点灯させる①

Raspberry Pi ピンアサイン



本講座で使用するのは主に GPIO、3.3V、Ground(GND)です。

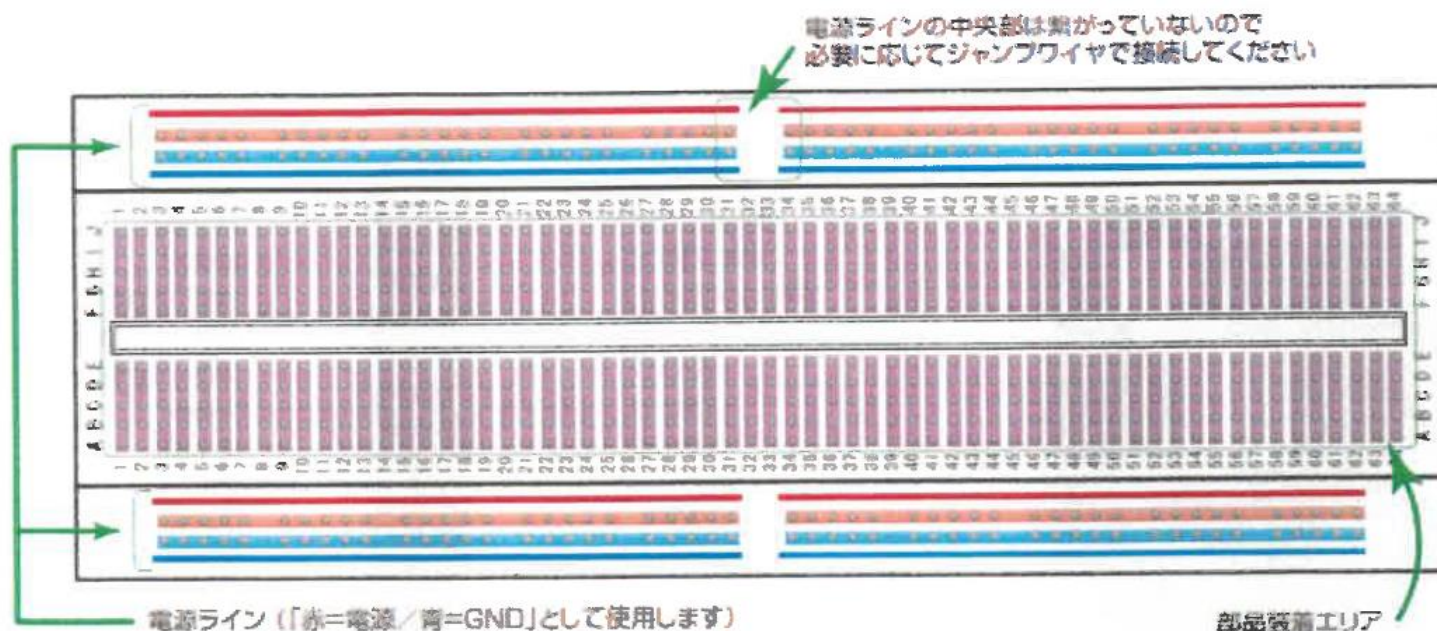
GPIO は General-purpose input/output の略称で「汎用入出力」を意味します。

すでに使用方法が与えられているポートがあります。括弧がついているものは使用しないのが無難です。

5V、3.3V、Ground(GND)を直接つながないように気を付けてください。

電気回路を構築してLEDを点灯させる②

ブレッドボードの構造



無数の穴が開いていて部品やコードを刺して回路を作成する。主に実験や試作に使われる。

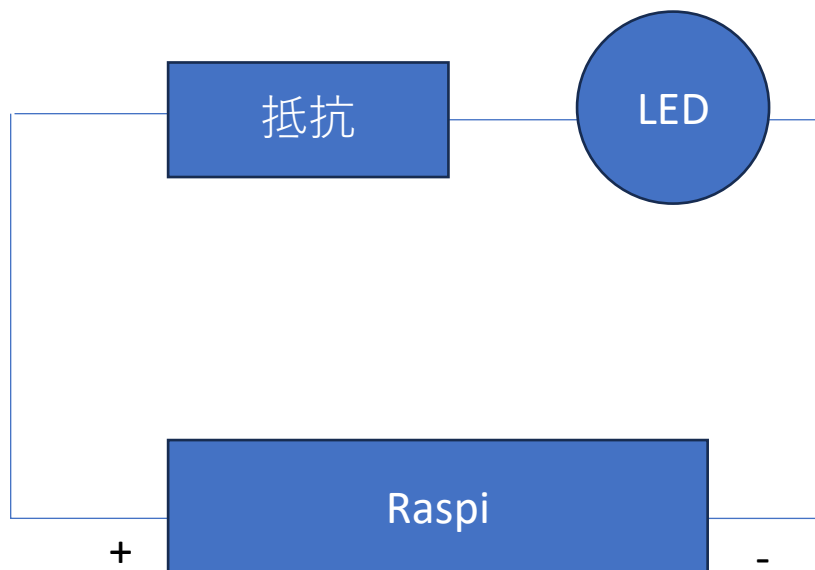
紫のラインが内部的につながっている状態なので FGHIJ どこにさしても動作はかわらない。

両サイドにある赤と青は電源ラインです。

電源と GND を直接接続しないように気を付けてください。

電気回路を構築してLEDを点灯させる③

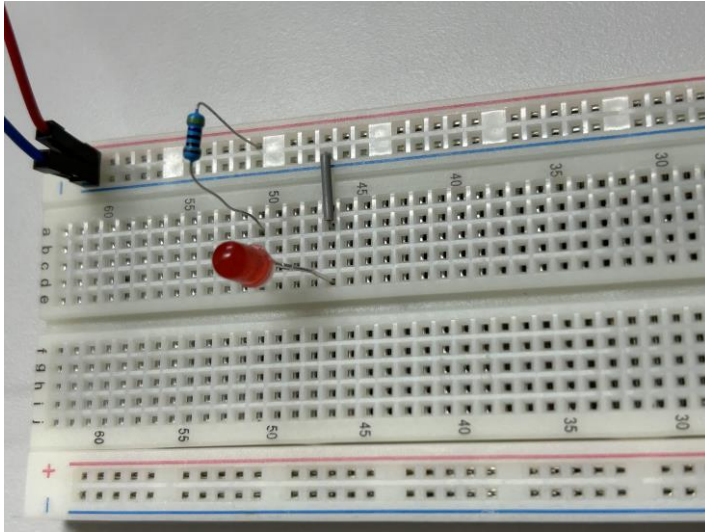
フレッドボード上でこのような回路を作ってみましょう。



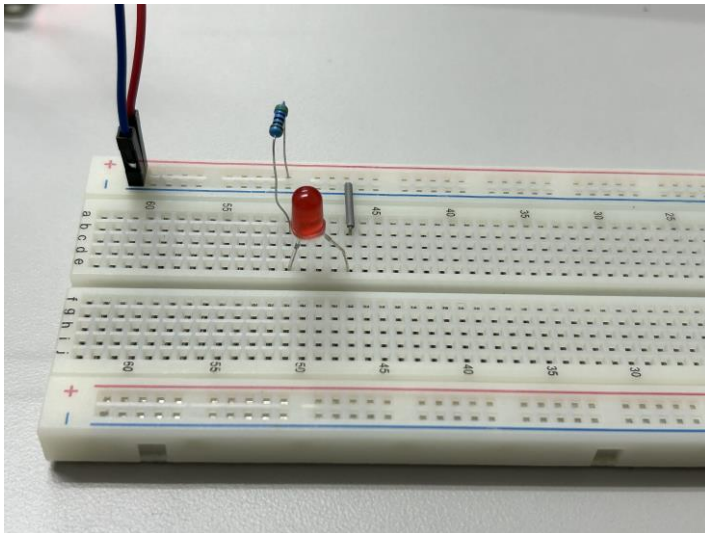
RaspberryPiが電源装置となり、GPIOの「3V3power」が+極、「Ground(GND)」が一極となって電流が流れる仕組みです。抵抗は100Ωを 사용합니다。

もし抵抗を用いなかった場合、回路はショートサーキット、いわゆるショートした状態となり過電流が流れてしまいます。その場合、RaspberryPiも損傷する恐れがあります。組み込み開発で電流回路をつくる時は、決して過電流が流れないように注意してください。

電気回路を構築してLEDを点灯させる④

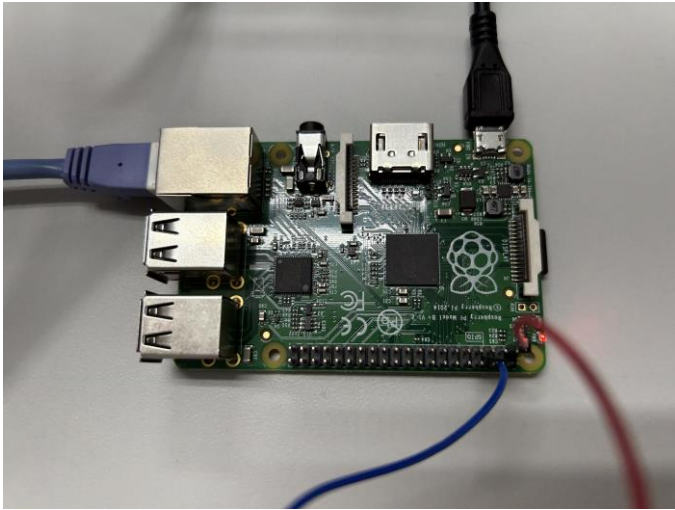


←上から見た状態

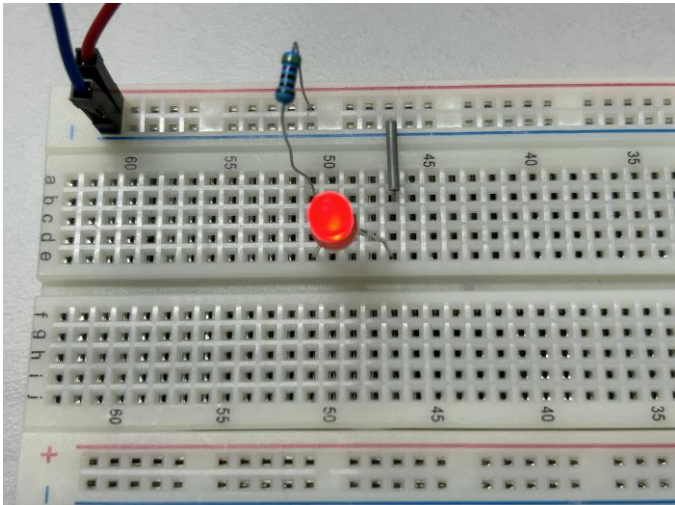


←横から見た状態

電気回路を構築してLEDを点灯させる⑤



RaspberryPiのGPIOの中からピンアサイン表の1番ピン（3V3power）に+側のコードを差し込み、6番ピン（Ground）に一側のコードを差し込みます。



点灯します。

GPIOを制御してLEDを点灯させる①

RaspberryPiはGPIOの出力を制御することができます。先ほどの回路で+極のコードを1番ピン（3V3power）から12番ピン(GPIO18)にに変更します。

Tera Termに以下のように順に入力していきます。echoコマンドを用いて、必要な値を絶対パスで指定したファイルに書き込んでいることに注目してください。

①GPIO18を使用可能にする

```
echo 18 > /sys/class/gpio/export
```

②GPIO18を出力モードにする

```
echo out > /sys/class/gpio/gpio18/direction
```

③LED点灯

```
echo 1 > /sys/class/gpio/gpio18/value
```

④LED消灯

```
echo 0 > /sys/class/gpio/gpio18/value
```

⑤GPIO18の使用を終了する

```
echo 18 > /sys/class/gpio/unexport
```


GPIOを制御してLEDを点灯させる②

各コマンドを実行するごとに、lsコマンドを実行してみました。

raspberrypi.local - 0601pythonc@raspberrypi: ~ VT

ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

```
0601pythonc@raspberrypi:~ $ ls /sys/class/gpio
export gpiochip0 unexport
0601pythonc@raspberrypi:~ $ echo 18 > /sys/class/gpio/export
0601pythonc@raspberrypi:~ $ ls /sys/class/gpio/
export gpio18 gpiochip0 unexport
0601pythonc@raspberrypi:~ $ ls /sys/class/gpio/gpio18/
active_low device direction edge power subsystem uevent value
0601pythonc@raspberrypi:~ $ echo out > /sys/class/gpio/gpio18/direction
0601pythonc@raspberrypi:~ $ echo 1 > /sys/class/gpio/gpio18/value
0601pythonc@raspberrypi:~ $ echo 0 > /sys/class/gpio/gpio18/value
0601pythonc@raspberrypi:~ $ echo 18 > /sys/class/gpio/unexport
0601pythonc@raspberrypi:~ $ ls /sys/class/gpio
export gpiochip0 unexport
0601pythonc@raspberrypi:~ $
```

各コマンドを実行したとき、RaspberryPiの内部でどのような動きが起きたのかを考えてみましょう。

GPIOを制御してLEDを点灯させる③

各コマンドを実行することでどのような動きが生じたのかを理解しましょう。

①GPIO18を使用可能にする

```
echo 18 > /sys/class/gpio/export
```

→gpioディレクトリ内にgpio18ディレクトリとその内部に各種ファイルが作られる。

②GPIO18を出力モードにする

```
echo out > /sys/class/gpio/gpio18/direction
```

→directionファイルに「out」を書き込むことでgpio18が出力モードになる。

③LED点灯

```
echo 1 > /sys/class/gpio/gpio18/value
```

→valueファイルに「1」を書き込むことでgpio18から電流が流れる。

④LED消灯

```
echo 0 > /sys/class/gpio/gpio18/value
```

→valueファイルに「0」を書き込むことでgpio18からの電流が流れなくなる。

⑤GPIO18の使用を終了する

```
echo 18 > /sys/class/gpio/unexport
```

→gpio18ディレクトリを削除する

プログラムを実行してLEDを点灯させる①

基本的な仕組みは理解できましたか？ここからいよいよ、C言語のプログラムでGPIOを制御していきます。まず、①GPIO18ディレクトリを作り、②GPIO18を出力モードにするところまでをLinuxコマンドで実行しておきます。

①GPIO18を使用可能にする

```
echo 18 > /sys/class/gpio/export
```

②GPIO18を出力モードにする

```
echo out > /sys/class/gpio/gpio18/direction
```

homeディレクトリに作業用のディレクトリを作り、そこにファイルを作成していきます。

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~$ mkdir 0601pythonc
0601pythonc@raspberrypi:~$ ls
0601pythonc Bookshelf Desktop Documents Downloads Music off2 off2.c off.c on on.c Pictures Public Templates Videos
0601pythonc@raspberrypi:~$ cd 0601pythonc
0601pythonc@raspberrypi:~/0601pythonc$
```

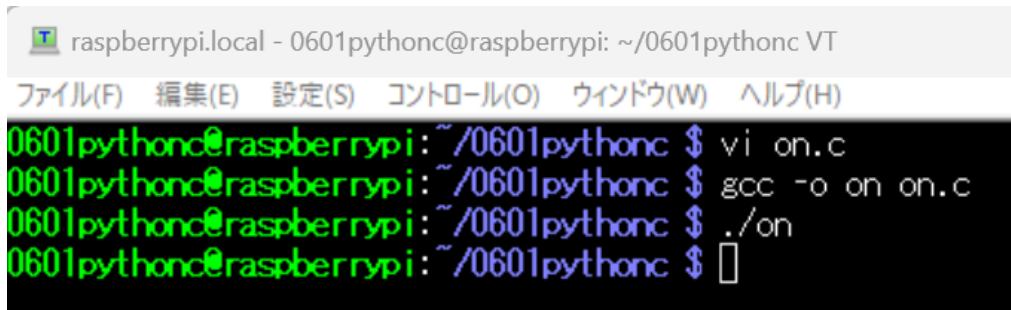

プログラムを実行してLEDを点灯させる②

viコマンドを実行し、「on.c」ファイルを作成します。

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
```

```
int main()
{
    int fd = 0;
    fd = open("/sys/class/gpio/gpio18/value", O_WRONLY);
    write(fd, "1", 2);

    close(fd);
}
```



The screenshot shows a terminal window titled 'raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT'. The menu bar includes 'ファイル(F)', '編集(E)', '設定(S)', 'コントロール(O)', 'ウィンドウ(W)', and 'ヘルプ(H)'. The command history shows the following steps:

```
0601pythonc@raspberrypi:~/0601pythonc $ vi on.c
0601pythonc@raspberrypi:~/0601pythonc $ gcc -o on on.c
0601pythonc@raspberrypi:~/0601pythonc $ ./on
0601pythonc@raspberrypi:~/0601pythonc $
```

LEDが点灯すれば成功です。

プログラムを実行してLEDを点灯させる③

プログラムの各行を見ていきましょう。

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
```

} fcntl.hはopen関数、unistd.hはwrite関数
とclose関数を呼び出すために必要です。

```
int main()
```

```
{
```

```
int fd = 0; fd:ファイルディスクリプタを格納するための変数
```

```
fd = open("/sys/class/gpio/gpio18/value", O_WRONLY);
```

```
valueファイル(絶対パスで記述)を書き込みモードで開く。
```

```
open関数の戻り値は開いたファイルのファイルディスクリプタになっている。
```

```
write(fd, "1", 2);
```

```
開いたファイルに"1"を書き込む(第3引数は書き込むデータのメモリサイズ)。
```

```
close(fd);
```

```
ファイルを閉じる
```

```
}
```

※ファイルディスクリプタ：プログラムからファイルを操作する際、操作対象のファイルを識別・同定するために割り当てられる番号。整数値。

プログラムを実行してLEDを点灯させる④

【演習1】 LEDを消灯するファイルを作成し、実行して消灯した後、「GPIO18」のディレクトリを削除してください。

LEDを消灯のうえ、下図のように表示されれば成功です。

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~/0601pythonc $ echo 18 > /sys/class/gpio/unexport
0601pythonc@raspberrypi:~/0601pythonc $ ls /sys/class/gpio
export  gpiochip0  unexport
0601pythonc@raspberrypi:~/0601pythonc $
```

プログラムを実行してLEDを点灯させる⑤

【演習1 解答例】

vi off.cを実行し、

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    int fd = 0;
    fd = open("/sys/class/gpio/gpio18/value", O_WRONLY);
    write(fd, "0", 2);

    close(fd);
}
```

を保存、コンパイル、実行で消灯。その後、Linuxコマンドで
echo 18 > /sys/class/gpio/unexport
を実行してlsコマンドで確認。

FTPソフトを活用する①

テキストエディタとしてのvimはVisualStudioやVScodeと比べて、プログラムコードの記述が面倒であることは否めません。そこでFTPソフトを活用してプログラミング作業の効率化を実践してみましょう。

- ①PC端末のプログラミング用テキストエディタでコードを作成
- ②FTPソフトでファイルをRaspberryPiのディレクトリに移動して
コンパイル→実行

今後は以上の流れで作業していきます。

Winscpのダウンロードとインストール

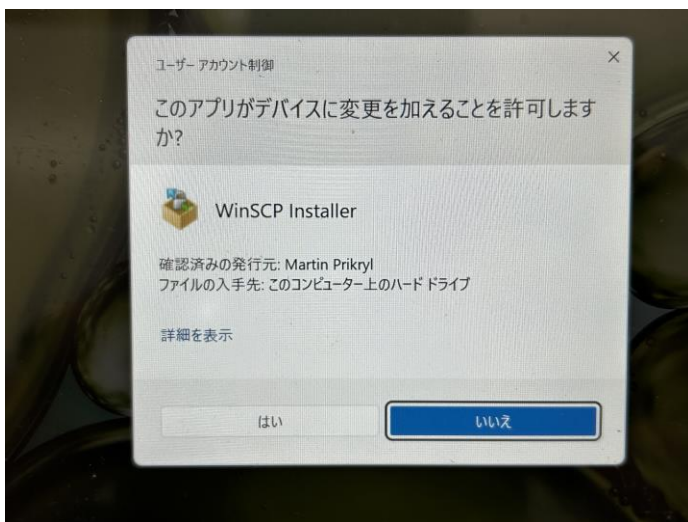


①下記リンクを検索
<https://forest.watch.impress.co.jp/library/software/winscp/>

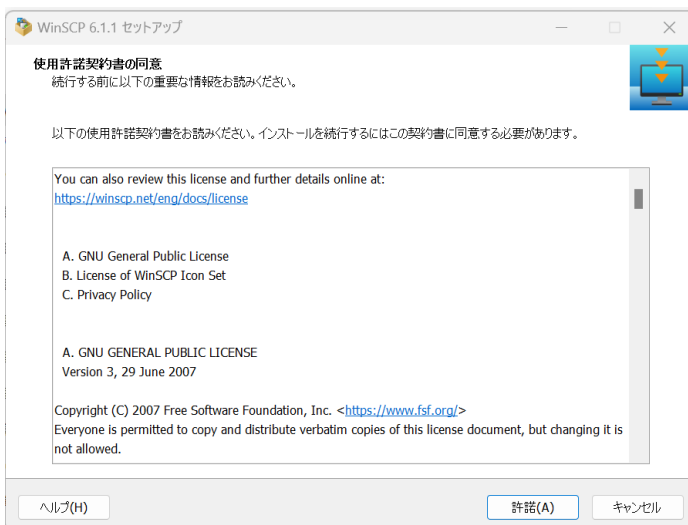
②ここをクリックするとexe
ファイルがダウンロードされ
ます。

FTPソフトを活用する②

③exeファイルを実行します。

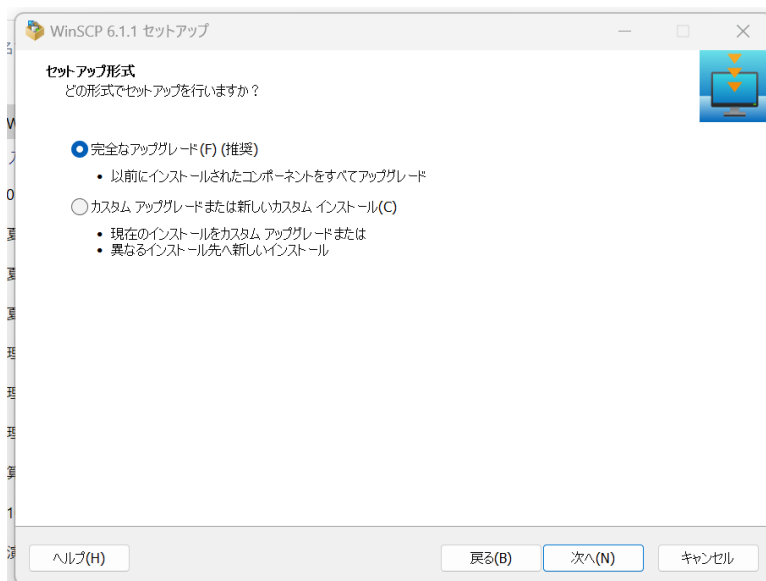


④ 「はい」 をクリック



⑤ 「許諾」 をクリック

FTPソフトを活用する③



⑥ 「完全なアップグレード」を選択して「次へ」をクリック



⑦ 「インストール」をクリック

インストールが完了したらPCを再起動しましょう。

FTPソフトを活用する④

「プログラムを実行してLEDを点灯させる①」ではあらかじめGPIO18を使用可能にして出力モードにするまでをLinuxコマンドで実行しておきましたが、今回はここまでを関数で実装し、main関数で呼び出して実行するところまでをWinscpを活用して実施してみましょう。

VisualStudioで2つのファイルを作成します。

standby.c

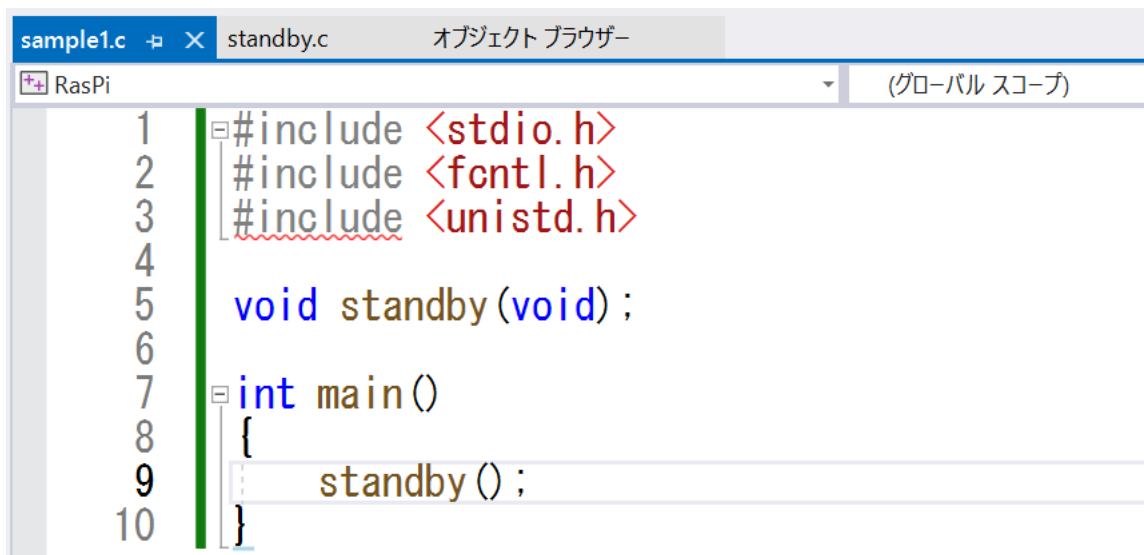


```
sample1.c  standby.c  オブジェクト ブラウザー
RasPi  (グローバル スコープ)
1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <unistd.h>
4
5  void standby(void) {
6      int fd = 0;
7      fd = open("/sys/class/gpio/export", O_WRONLY);
8      write(fd, "18", 2);
9
10     close(fd);
11
12     fd = 0;
13     fd = open("/sys/class/gpio/gpio18/direction", O_WRONLY);
14     write(fd, "out", 3);
15
16     close(fd);
17 }
```


FTPソフトを活用する⑤

ここではmain関数の前でプロタイプ宣言をすることで、standby関数を呼び出せるようにしています。ヘッダファイルを作り、sample1.cでincludeすることでも呼び出し可能になります。

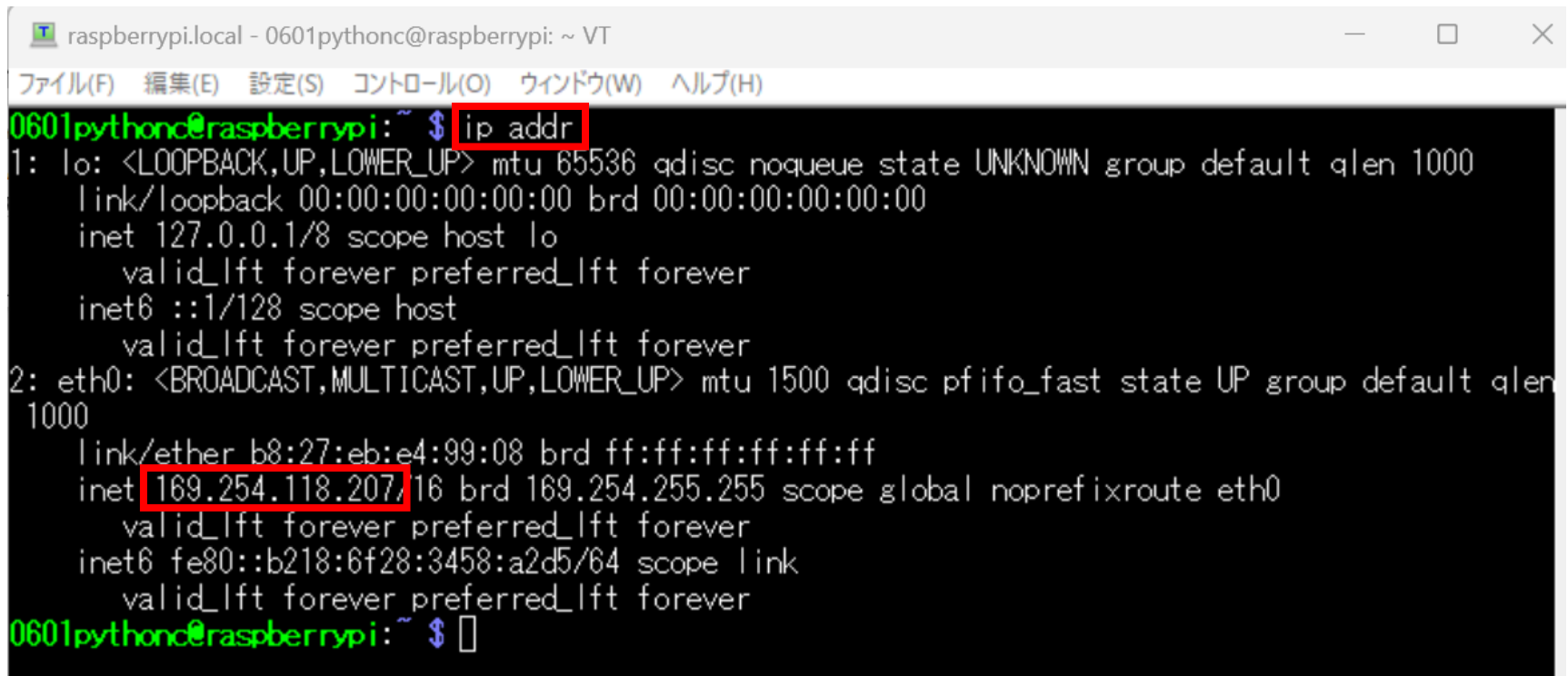
sample1.c



```
1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <unistd.h>
4
5 void standby(void);
6
7 int main()
8 {
9     standby();
10 }
```

FTPソフトを活用する⑥

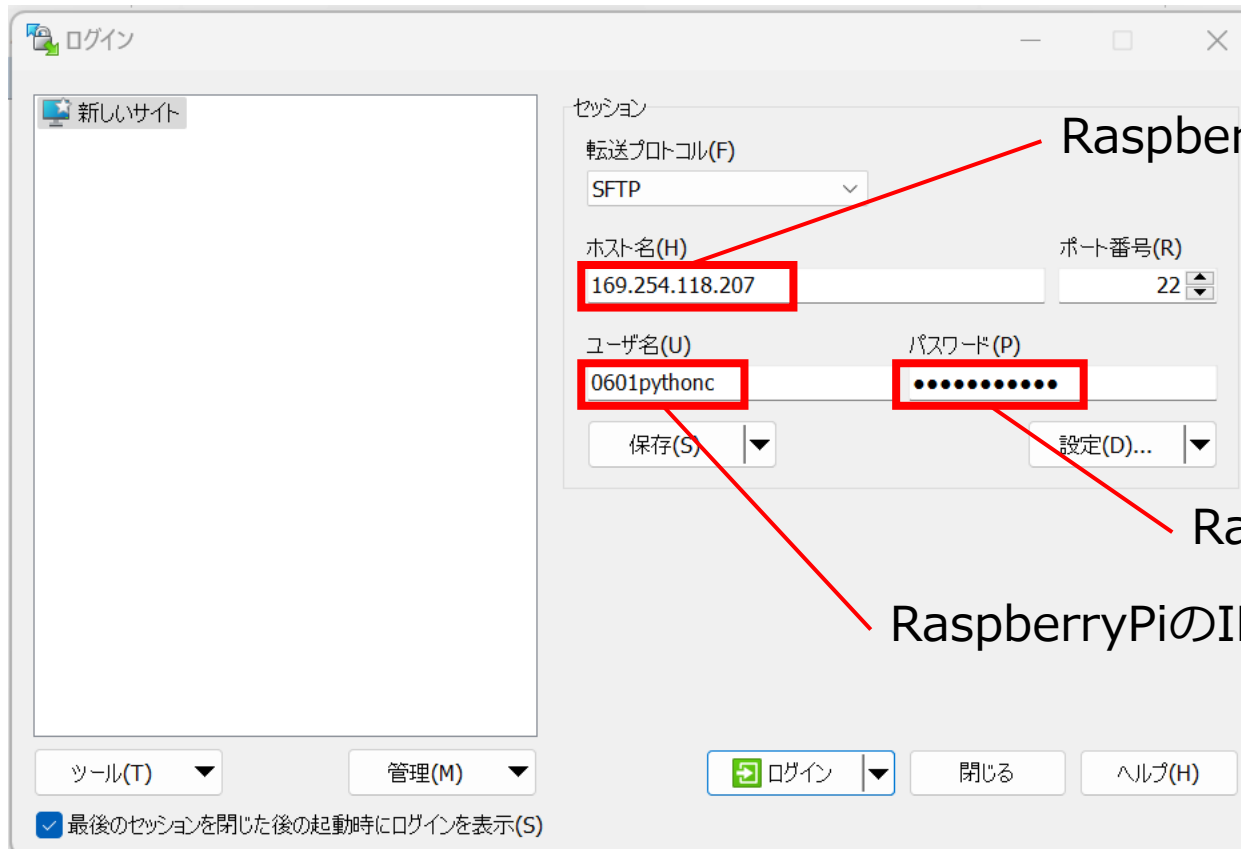
VisualStudioで作成した2つのファイルをRaspberryPiの1101pythoncのディレクトリにコピーします。接続するためにはRaspberryPiのIPアドレスが必要になります。TeraTermでip addrコマンドを使用します。



```
raspberrypi.local - 0601pythonc@raspberrypi: ~ VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~ $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen
1000
    link/ether b8:27:eb:e4:99:08 brd ff:ff:ff:ff:ff:ff
    inet 169.254.118.207/16 brd 169.254.255.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::b218:6f28:3458:a2d5/64 scope link
        valid_lft forever preferred_lft forever
0601pythonc@raspberrypi:~ $
```

FTPソフトを活用する⑦

Winscpを起動します。



FTPソフトを活用する⑦

PC側の移動させたいファイルが入っているフォルダ、RaspberryPi側のファイルを受け入れたいディレクトリまでそれぞれ移動させます。移動させたいファイルをクリックアンドドラッグで移動させましょう。

PC側のフォルダ

RaspberryPi側のディレクトリ

The screenshot displays the WinSCP interface with two panes. The left pane shows the PC's local file system (C:\Users\hzm77\cocode\RasPi\), and the right pane shows the Raspberry Pi's file system (/home/0601pythonc/). Both panes have a table view showing files and folders with columns for name, size, type, update time, permissions, and owner.

PC側のフォルダ (Left Pane):

名前	サイズ	種類	更新日時
..		ひとつ上のディレクトリ	2023/09/15 18:26:08
RasPi.sln	2 KB	Visual Studio Soluti...	2023/09/15 17:36:45
RasPi.vcxproj	7 KB	VCXPROJ ファイル	2023/09/15 17:36:44
RasPi.vcxproj.filters	1 KB	VC++ Project Filter...	2023/09/15 17:36:45
RasPi.vcxproj.user	1 KB	Per-User Project O...	2023/09/15 17:36:45
sample1.c	1 KB	C Source	2023/09/15 18:00:56
standby.c	1 KB	C Source	2023/09/15 18:10:49

RaspberryPi側のディレクトリ (Right Pane):

名前	サイズ	更新日時	パーミッション	所有者
..		2023/05/03 9:38:16	rw-r--r--	root
0601pythonc		2023/05/03 19:22:49	rw-r--r--	0601pyt...
Bookshelf		2023/05/03 9:16:32	rw-r--r--	0601pyt...
Desktop		2023/05/03 9:38:08	rw-r--r--	0601pyt...
Documents		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Downloads		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Music		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Pictures		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Public		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Templates		2023/05/03 9:38:13	rw-r--r--	0601pyt...
Videos		2023/05/03 9:38:13	rw-r--r--	0601pyt...
off.c	1 KB	2023/05/03 12:48:26	rw-r--r--	0601pyt...
off2	8 KB	2023/05/03 12:52:32	rw-r--r--	0601pyt...
off2.c	1 KB	2023/05/03 12:52:17	rw-r--r--	0601pyt...
on	8 KB	2023/05/03 12:43:18	rw-r--r--	0601pyt...
on.c	1 KB	2023/05/03 12:42:29	rw-r--r--	0601pyt...

FTPソフトを活用する⑧

作業中のディレクトリに移動してlsコマンドで確認しましょう。

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~ $ cd ./0601pythonc
0601pythonc@raspberrypi:~/0601pythonc $ ls
off off.c on on.c sample1.c standby.c
0601pythonc@raspberrypi:~/0601pythonc $
```

2つのファイルを同時にコンパイルする「分割コンパイル」に注目してください。

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~/0601pythonc $ gcc -o sample1 sample1.c standby.c
0601pythonc@raspberrypi:~/0601pythonc $ ./sample1
0601pythonc@raspberrypi:~/0601pythonc $ ls /sys/class/gpio
export gpio18 gpiochip0 unexport
0601pythonc@raspberrypi:~/0601pythonc $
```

分割コンパイル

gpio18が作られているので成功です。

FTPソフトを活用する⑨

【演習2】 GPIO18の使用を終了するdeleteFile関数を実装し、main関数で実行してください。

close関数を定義するdeleteFile.cとdeleteFile関数を実行するsample2.cを作成して、分割コンパイルの後、実行しましょう。

```
0601python@raspberrypi:~/0601pythonc $ ls /sys/class/gpio
export gpio18 gpiochip0 unexport
0601python@raspberrypi:~/0601pythonc $ gcc -o sample2 sample2.c deleteFile.c
0601python@raspberrypi:~/0601pythonc $ ./sample2
0601python@raspberrypi:~/0601pythonc $ ls /sys/class/gpio
export gpiochip0 unexport
0601python@raspberrypi:~/0601pythonc $
```

gpio18が削除されているので成功です。

FTPソフトを活用する⑩

【演習2 解答例】

<deleteFile.c>

```
1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <unistd.h>
4
5  void deleteFile(void) {
6      int fd = 0;
7      fd = open("/sys/class/gpio/unexport", O_WRONLY);
8      write(fd, "18", 2);
9
10     close(fd);
11 }
```

<sample2.c>

```
1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <unistd.h>
4
5  void deleteFile(void);
6
7  int main()
8  {
9      deleteFile();
10 }
```

FTPソフトを活用する⑪

これまで、C言語のプログラミングでLEDの点灯、消灯、GPIO18を使用可能にし、出力モードにする手順の関数化、GPIO18の使用を終了する手順の関数化を実践しました。これらを組み合わせて、①GPIO18を使用可能にしてから出力モードにし②LEDを5秒点灯させた後、③消灯し④GPIO18の使用を終了するところまでをプログラムで自動化してみましょう。

LEDを5秒間点灯させるにはusleep関数を使います。GPIO18を使用可能にし、出力モードにしたあと以下のコードを作成し、実行してみましょう。

<sample3.c>

```
1 #include <stdio.h>
2 #include <fcntl.h>
3 #include <unistd.h>
4
5 int main() {
6     int fd = 0;
7     fd = open("/sys/class/gpio/gpio18/value", O_WRONLY);
8     write(fd, "1", 2);
9     close(fd);
10    usleep(5000000);
11
12    fd = open("/sys/class/gpio/gpio18/value", O_WRONLY);
13    write(fd, "0", 2);
14    close(fd);
15 }
```

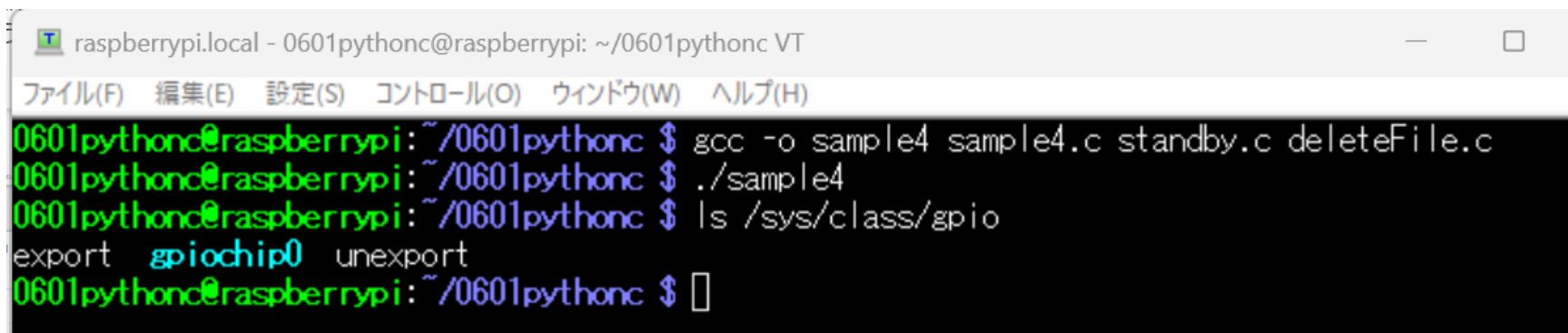
usleep関数はプログラムの実行を指定時間待機させます。マイクロ秒単位で指定するので5秒は5000000です。

FTPソフトを活用する⑫

【演習3】 ①GPIO18を使用可能にしてから出力モードにし②LEDを5秒点灯させた後、③消灯し④GPIO18の使用を終了するところまでを実行するsample4.cを作成してください。

※①はstandby関数、④はdeleteFile関数を使用しましょう。

終了後、GPIO18のディレクトリが削除されていることを確認してください。

A terminal window titled 'raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT'. The menu bar shows 'ファイル(F)', '編集(E)', '設定(S)', 'コントロール(O)', 'ウィンドウ(W)', and 'ヘルプ(H)'. The terminal shows the following commands and output:

```
0601pythonc@raspberrypi: ~/0601pythonc $ gcc -o sample4 sample4.c standby.c deleteFile.c
0601pythonc@raspberrypi: ~/0601pythonc $ ./sample4
0601pythonc@raspberrypi: ~/0601pythonc $ ls /sys/class/gpio
export  gpiochip0  unexport
0601pythonc@raspberrypi: ~/0601pythonc $
```

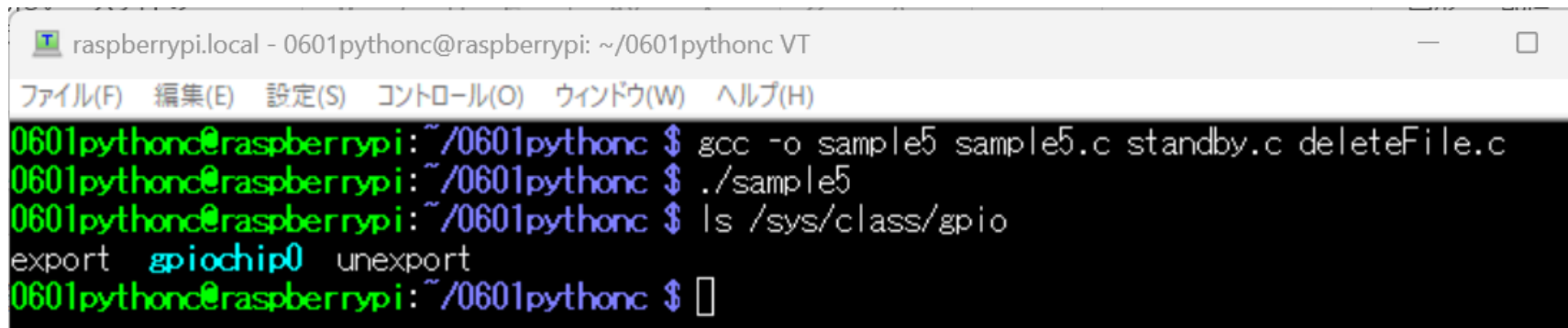
点灯後、こうなっていれば成功です。

FTPソフトを活用する⑬

【演習4】①GPIO18を使用可能にしてから出力モードにし②LEDを2秒間点灯を5回くりかえした後、③消灯し④GPIO18の使用を終了するところまでを実行するsample5.cを作成してください。

※①はstandby関数、④はdeleteFile関数を使用しましょう。

終了後、GPIO18のディレクトリが削除されていることを確認してください。



```
raspberrypi.local - 0601pythonc@raspberrypi: ~/0601pythonc VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~/0601pythonc $ gcc -o sample5 sample5.c standby.c deleteFile.c
0601pythonc@raspberrypi:~/0601pythonc $ ./sample5
0601pythonc@raspberrypi:~/0601pythonc $ ls /sys/class/gpio
export  gpiochip0  unexport
0601pythonc@raspberrypi:~/0601pythonc $
```

5回点灯した後、こうなっていれば成功です。

FTPソフトを活用する⑭

関数の定義の詳細です。

open関数

書式 `#include <fcntl.h>`
 `int open(const char *filename, int amode);`

戻値 -1 オープン失敗
 正常にオープンできた場合は、ファイルディスクリプタが返される。

動作 **filename** で指定されたファイルを、**amode** で指定されたモードでオープンし、ファイルディスクリプタを返す。
 amode は以下のオプションを使用し、「| (or)」で複数指定が可能。

 O_RDONLY 読み込み専用
 O_WRONLY 書き込み専用
 O_RDWR 読み書き両用
 O_APPEND 追記モード
 O_CREAT ファイルが存在しない場合作成
 O_TRUNC ファイルを 0 に切り詰める

close関数

書式 `#include <unistd.h>`
 `int close(int fd);`

戻値 -1 クローズ失敗
 0 正常終了

動作 **fd** で指定されたファイルディスクリプタに対応するファイルをクローズする。

FTPソフトを活用する⑮

write関数

書式	<pre>#include <unistd.h> int write(int <i>fd</i>, void *<i>buf</i>, unsigned <i>size</i>);</pre>
戻値	書き込んだデータバイト数
動作	<i>fd</i> で指定されたファイルディスクリプタのファイルに <i>buf</i> で指定されるデータを <i>size</i> で指定されるバイト数分、書き込みを行う。

usleep関数に関しては同じようにプログラムの進行を停止させるsleep関数があります。こちらは停止時間を秒単位で指定しますが引数の型がint型であるため、1秒未満の時間を指定できません。そのため今回はusleep関数を使用しました。

7セグメントLEDの利用①

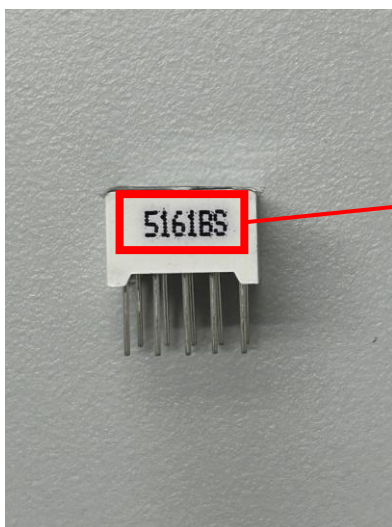
7セグメントLEDは、数字情報の表示に特化したデジタル表示モジュールです。表示する数字の形状部に発光ダイオード（LED）を配しているため、大変視認性に優れています。

「LED数字表示器」や「7セグLED」と呼ばれることもあります。

7セグメントLEDには、アノードコモンとカソードコモンの2種類の回路があります。

- ・アノードコモン：共通ピン(common)が+の場合
- ・カソードコモン：共通ピン(common)が-の場合

今回使用する7セグメントLEDがどちらになるかは、本体に記されている番号を検索するとわかります。



「5161BS」で検索します。

7セグメントLEDの利用②

XLITX®

www.xlitx.com



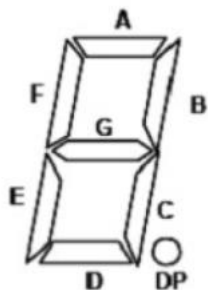
Model : 5161BS
Size : 0.56-inch
Emitting color : Red (Ultra-Bright)
Mode : Common-Anode (CA)
Digit : Single Digit
Category : LED 7-Segment Display
Maker : XLITX Technology

Common-Anode(アノードコモン)とあります。

7セグメントLEDの利用③



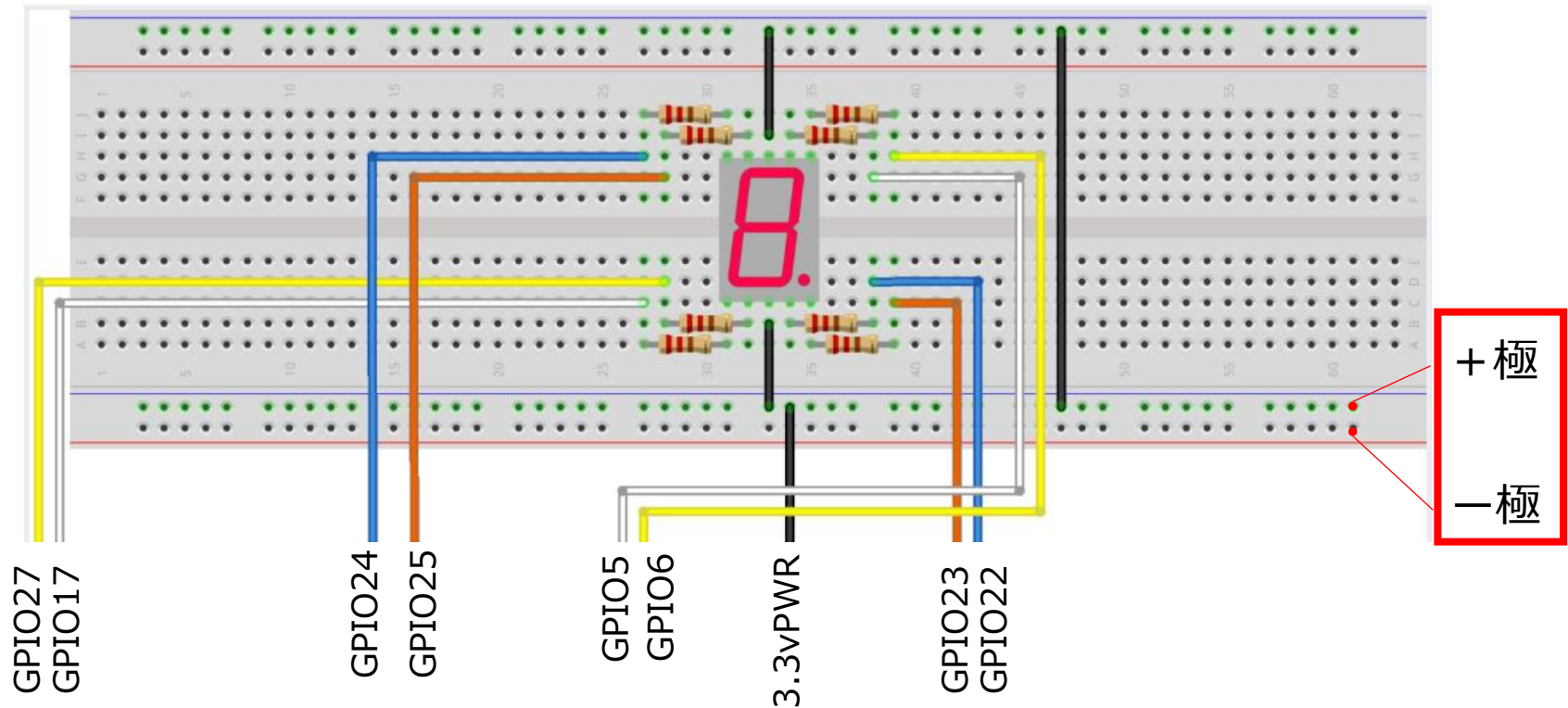
7セグメントLEDには10本のピンがあり、その内、中央の3番と8番は共通ピンになっています。この共通ピンが+極になっているのがアノードコモン、一極になっているのがカソードコモンです。



7セグメントLEDには計8個のLEDがあり、共通ピンを除いた1,2,4,5,6,7,9,10番のピンに接続したGPIOから出力することによって各LEDが点灯します。このときのLEDとGPIOの対応関係を調べる必要があります。

7セグメントLEDの利用④

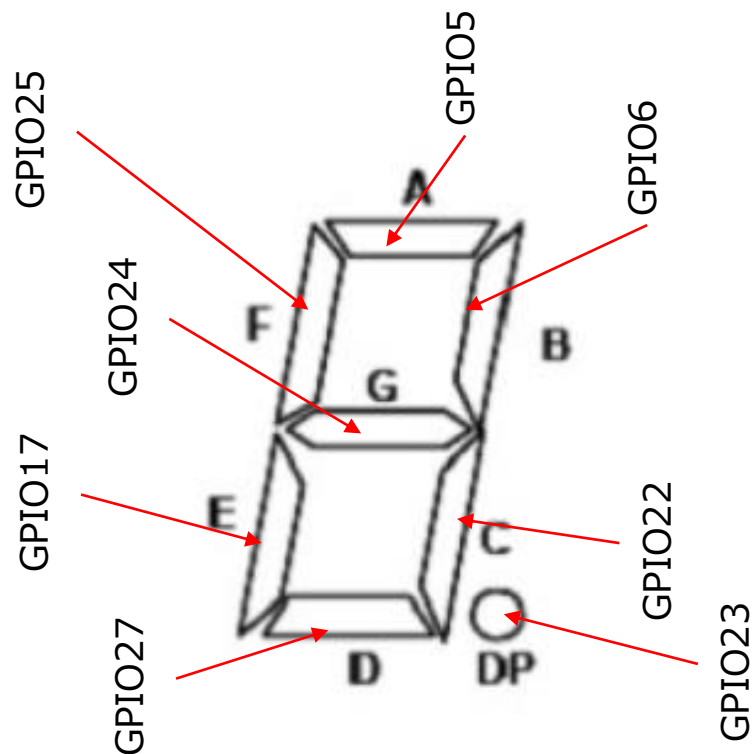
使用するGPIOは3.3vPWR,GPIO17,27,22,23,24,25,5,6です(GPIOの番号順に記載しています)。下記のようにブレッドボード上で配線してください。



※アノードコモンなのでこの図と実際のブレッドボードとで+-の接続が逆になります！

7セグメントLEDの利用⑤

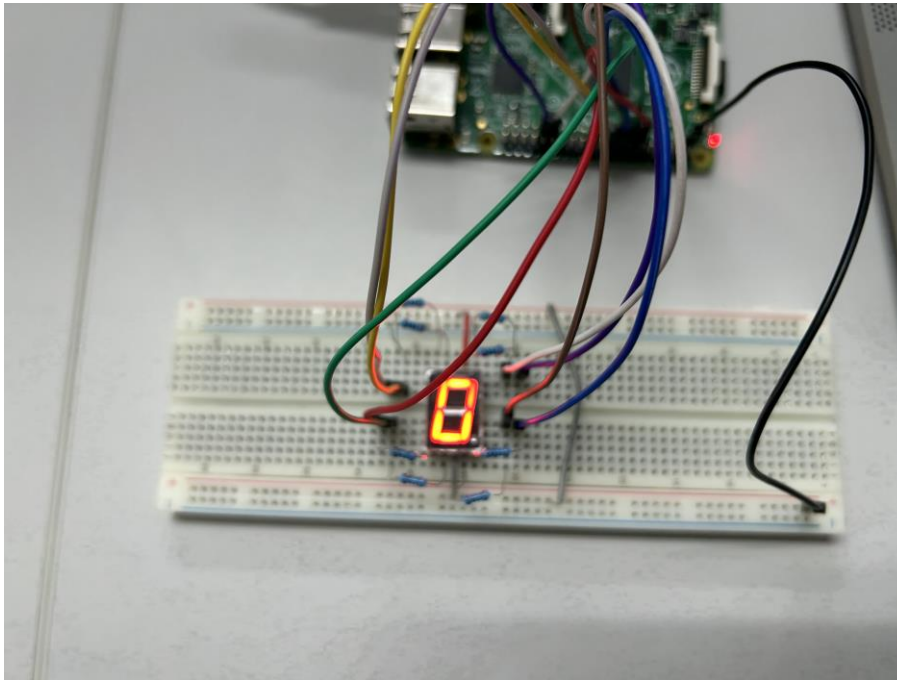
GPIO17を使用可能にし、ここでは出力状態にします(アノードコモンであるため)。この時点で接続されているLEDが点灯するので、位置を確認・記録した後、value ファイルに「1」を書き込んで消灯します。このようにして、すべてのGPIOと点灯するLEDの対応関係を調べていきましょう。



ここではこのようになりましたが、配線の状況では異なる結果になることがあります。その場合でもGPIOとLEDに1対1の対応関係があればよしとします。

7セグメントLEDの利用⑥

さきほど調べたGPIOとLEDの対応関係を使って数字の「0」を点灯させてみましょう。A,B,C,D,E,FのLEDを点灯させればよいことがわかります。



消灯して、1~9までをすべて点灯させてみましょう。
すべてのLEDを消灯したら、この後の作業用ディレクトリとして、ホームディレクトリ直下にsevensgmentsディレクトリを作っておきましょう。

7セグメントLEDの利用⑦

実機を用いた動作確認はできたので、ここからはプログラムによって制御していきます。組込み開発は基本的にこのような流れになります。

【演習5】 C言語を用いて7セグメントLEDで0~9を表示するプログラムを以下の要件に基づいて作成してください。ファイルを作成する場所はsevensgmentsディレクトリとします。

- ①現状、各GPIOが使用可能で、出力モードになっています。ここから始めてかまいません。
- ②0~9の各数字を点灯させるon1関数~on9関数を定義するon.cを作成してください。
- ③0~9の各数字を消灯させるoff1関数~off9関数を定義するoff.cを作成してください。
- ④9→0の順でカウントダウン表示をするcountdown.cを作成し、実行して確認してください。

7セグメントLEDの利用⑦

【演習6】 GPIOを使用可能にしてから、演習5のカウントダウンを実行し、GPIOを使用終了状態にするまでを自動化したautocountdown.cを作成し、実行してください。ファイルを作成する場所はsevensgmentsディレクトリとします。

この状態から実行して

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/sevensgments VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~/sevensgments $ ls /sys/class/gpio
export gpiochip0 unexport
0601pythonc@raspberrypi:~/sevensgments $ █
```

カウントダウン実行後に

```
raspberrypi.local - 0601pythonc@raspberrypi: ~/sevensgments VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
0601pythonc@raspberrypi:~/sevensgments $ gcc -o autocountdown autocountdown.c autostandby.c autodown.c on.c off.c
0601pythonc@raspberrypi:~/sevensgments $ ./autocountdown
0601pythonc@raspberrypi:~/sevensgments $ ls /sys/class/gpio
export gpio23 gpiochip0 unexport
0601pythonc@raspberrypi:~/sevensgments $ █
```

こうなることを確認してください。