FABLAB北加賀屋　電子工作チーム「えんや」ワークショップ

クリスマスボードを自由に光らせよう！

2020年12月05日

2021年1月1日 rev.3

担当：ぺんきち

はじめに

　MicroPythonでプログラミングして、クリスマスボード上のLEDを自由に点灯させましょう。MicroPythonを使うので複雑なプログラミングが簡単にできます。ぜひ、オリジナルなクリスマスボードに仕上げてください。また、このワークショップを通じて、Pythonの便利さや、Aiに向いている理由を感じてもらえると幸いです。

使用するマイコン（ＥＳＰ－ＷＲＯＯＭ－０２）

　WROOM02は、Espressif 社のESP8266という32bitマイコンを搭載したWifiモジュールです。電子工作界隈では、Arduino IDEで使うのが一般的のようですが、今回は、本家MicroPython を組み込んだ環境を準備しました。

WebREPLの準備

|  |
| --- |
| PCから、<https://github.com/micropython/webrepl> にアクセス  「Code」の耳　→　Download ZIP をクリックして自分のPCに保存  → 保存したものを解凍してください。 |

WebREPLの設定

|  |
| --- |
| クリスマスボードは、AP\_IF（アクセスポイント・インターフェース）とSTA\_IF（ステーション・インターフェース）という2つのWiFiインターフェースを持っています。  AP\_IFは、固有のessid（例えば MicroPython-ef5b35）をもっています。自分のボードに電源をつないで（USB給電）、essid（袋に表示してます）が実際に出ているかPCで確認ください。  PCを、WiFi設定で自分のボードのessidにつないでください。パスワードは、micropythoN です（最後のNだけ大文字）。  無事つながったら、Webブラウザー（FireFoxがオススメです。それ以外だと、コピペがうまくいきません）で先にダウンロードしたwebrepl-masterフォルダの中にあるwebrepl.htmlを開きます。  Connect ボタンを押します。  パスワードを聞いてきます。xmas と入力してください。 --> WebREPL connected と表示されれば成功です（既にmain.py が実行中の場合は、Ctrl+Cを押して中断してください）。  このままWebREPLを使っても良いですが、インターネットを使うことができずチョット不便。インターネットも使いたい場合は、次のようにします。  >>> sta\_if.active(True) >>> sta\_if.connect("essid", "password")　（PCをインターネットに繋げるessidとパスワードを” ”でくくります。対応WiFiプロトコルは 802.11b/g/n(2.4GHz)です。  >>> sta\_if.isconnected() --> True が表示されること確認。なお、次回からは、sta\_if.connect("essid","password")の部分は記憶されるので、単に、sta\_if.connect() でOK。  >>> sta\_if.ifconfig() --> ('192.168.1.5', '255.255.255.0', '192.168.1.1', '192.168.1.1') のように表示される。 先頭のIPアドレス（この例では、192.168.1.5）が、自分のクリスマスボードのアドレスなので、これをメモしておきます。  Disconnectボタンを押して接続を切ります。  PCをインターネットWiFiにつないだ後、WebREPLの左端のws://192.168.4.1:8266/ のip部分をメモしておいたipアドレス（この例では、192.168.1.5）に変えて、Connectボタンを押します。  パスワードを聞いてくるので、xmas と入力してください。  以上で、インターネットを使いながらWebREPLを使うことができます。 |

ボードのMicroPythonプログラムが実行される順番

|  |  |
| --- | --- |
| １）Boot.py | 必ず実行されます。本ワークショップ用の設定や関数を入れてます。 |
| ２）main.py | 存在すれば、必ず実行される。最初は存在してません。 |
| ３）その他のプログラム | 実行するように設定してあれば実行される |
| ４）WebREPL | 上記プログラム終了後、あるいは、Ctrl+Cを押して、上記実行を中断した場合に使えるようになります。 |

プログラミング演習

# ex1  
# 単色（赤）  
# WebREPL を使って、LEDを点灯させてみます。以下の2行をWebREPLから入力します。

|  |
| --- |
| np[0] = red np.write() |

# ex2  
# 単色（黄）

|  |
| --- |
| np[0] = yellow np.write() |

# ex3  
# 色がどのように定義されているか表示させてみる  
# pythonの「リスト」という型で定義。左からRGBの明るさを0〜255で指定。

|  |
| --- |
| red |

# ex4  
# 他の色の定義を表示させてみる（コンマで一気に）

|  |
| --- |
| green, yellow, blue |

# ex5  
# 自分で好きなように定義して光らせる

|  |
| --- |
| pink = [100, 50, 50] np[0] = pink np.write() |

# ex6  
# 3箇所光らせる

|  |
| --- |
| np[0] = green np[1] = yellow np[2] = blue np.write() |

# ex7  
# 3箇所光らせる（Python独自の表記法です。簡単でしょ！）

|  |
| --- |
| np[0], np[1], np[2] = red, blue, sky  np.write() |

# ex8  
# 7箇所全部光らせる

|  |
| --- |
| np[0], np[1], np[2], np[3], np[4], np[5], np[6] = red, orange, yellow, green, sky, blue, purple np.write() |

# ex8  
# リストを使う

# colors7 = [red, orange, yellow, green, sky, blue, purple] はboot.pyで定義済です。

# リストのインデックスは、ゼロから始まります。

# １つづつ指定するのは面倒ですね。

|  |
| --- |
| np[0], np[1], np[2], np[3], np[4], np[5], np[6] = colors7[0], colors7[1], colors7[2], colors7[3], colors7[4], colors7[5], colors7[6] np.write() |

# ex9  
# 関数で簡単に。

# 関数setcolors()は、boot.pyで定義済です。

# colors7 = [orange, yellow, green, sky, blue, purple, red]はboot.pyで定義済です。

# リストを関数に渡すだけなので、かなり簡単になりました。

|  |
| --- |
| setcolors(np, colors7) np.write() |

# ex10  
# 全部消す

# 関数clear()は、boot.pyで定義済です。

|  |
| --- |
| clear() |

# ex11  
# スライス

# colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]は、boot.pyで定義済です。

# スライスは、Python独自の要素指定方法です。非常に便利です。

# colors[0:4]は、リストの要素０番目から４ー１番目まで連続指定する意味です。

# なお、端っこは省略できます。

# colors[0:4] -> colors[:4] と省略可。colors[4:11] -> colors[4:] と省略可。

|  |
| --- |
| c = colors[0:4] setcolors(np, c) np.write() |

# ex12  
# スライス

|  |
| --- |
| clear() c = colors[4:9] setcolors(np, c) np.write() |

# ex12  
# マイナスで指定することもできます。（-1 ・・・最終要素）

# なお、要素の最後まで指定する場合は、colors[-5:] のように端を省略するとよい。

|  |
| --- |
| clear() c = colors[-5:-1] setcolors(np, c) np.write() |

# ex13  
# スライスと連結

|  |
| --- |
| clear() c = colors[1:3] + colors[7:10] setcolors(np, c) np.write() |

# ex14  
# Lチカ  
# Ctrl+E した後で、コピペし、その後Ctrl+Dしてください。

# while 条件式: は、条件式が真（True）の間、段落内を繰り返します。  
# 注）WebREPLでは、残念ながら8行くらいまでしかコピペできません。  
# 中断は、Ctrl+C

|  |
| --- |
| while True:  np[0] = red  np.write()  time.sleep(0.5)  clear()  time.sleep(0.5) |

# ex15  
# ぐるぐる回してみる

# colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]は、boot.pyで定義済です。

# time.sleep(n) は、n秒間何もしない（眠ってる）という意味です。

# c = c[1:] + c[0:1] で先頭要素が削除され、末尾に追加されます。

|  |
| --- |
| c = colors[:7] while True:  c = c[1:] + c[0:1]  setcolors(np, c)  np.write()  time.sleep(0.5) |

# ex16  
# 明るくなったり暗くなったり

|  |
| --- |
| clear() while True:  for i in range(256):  power = i / 256  np[0] = [int(red[0] \* power), int(red[1] \* power), int(red[2] \* power)]  np.write()  time.sleep(0.002) |

# ex17  
# 明るくなったり暗くなったり（リスト全体で）

# colors7 = [red, orange, yellow, green, sky, blue, purple] はboot.pyで定義済です。

# 関数 colors\_powered()は、boot.py で定義済です。カラーリストの要素のそれぞれにpowerの値を乗じます。

|  |
| --- |
| c = colors7[:] while True:  for i in range(256):  power = i / 256  setcolors(np, colors\_powered(c, power))  np.write()  time.sleep(0.002) |

# ex18  
# 明るさ調整（明るさリストで個別に）

# c = colors7[:] は、ex17で定義済みです。

# powers は、明るさの値をリストにしたものです。

# 関数colors\_powersed() は、boot.pyで定義済です。カラーリストの要素のそれぞれにpowersを乗じます。

|  |
| --- |
| powers = [0.05, 0.1, 0.2, 0.5, 0.7, 1.0, 1.2] setcolors(np, colors\_powersed(c, powers)) np.write() |

# ex19  
# 明るさリストを回す

# c = colors7[:] は、ex17で定義済みです。

|  |
| --- |
| powers = [0.05, 0.1, 0.2, 0.5, 0.7, 1.0, 1.2] while True:  powers = powers[1:] + powers[0:1]  setcolors(np, colors\_powersed(c, powers))  np.write()  time.sleep(0.5) |

# ex20  
# 色と明るさリストを回す

# c = colors7[:] は、ex17で定義済みです。

|  |
| --- |
| powers = [0, 1, 0, 1, 0, 1, 0] while True:  c = c[1:] + c[0:1]  for i in range(7):  powers = powers[1:] + powers[0:1]  setcolors(np, colors\_powersed(c, powers))  np.write()  time.sleep(0.1) |

# ex21  
# ランダムな値を生成(micropython 標準関数)  
# urandom.getrandbits(n) --> 0から2のn乗-1までのランダムな値を生成

|  |
| --- |
| urandom.getrandbits(12) |

# ex22  
# ランダムな値（自作関数）

# 関数randint() は、boot.pyで定義済みです。  
# randint(n) --> 0からn-1までのランダムな値を生成します。

|  |
| --- |
| randint(100) |

# ex23  
# リスト内包表記（Python独自の便利機能です）

|  |
| --- |
| [i for i in range(7)] |

# ex24  
# リスト内包表記（めちゃ簡単にリストが生成できます）

|  |
| --- |
| [random(10) for i in range(7)] |

# ex25  
# ランダムなカラーリスト生成

# colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]は、boot.pyで定義済です。

|  |
| --- |
| [colors[random(11)] for i in range(7)] |

# ex26  
# じゃ光らせてみる

# colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]は、boot.pyで定義済です。

|  |
| --- |
| c = [colors[random(11)] for i in range(7)] setcolors(np, c) np.write() |

# ex27  
# じゃあLチカ

|  |
| --- |
| while True:  setcolors(np, [colors[random(11)] for i in range(7)])  np.write()  time.sleep(0.5) |

# ex28  
# サイン関数  
# math.sin(n) nはラジアン（MicroPython 標準）  
# 関数 sine() は、boot.py で定義済です。sine(n) nは角度で指定します。

|  |
| --- |
| sine(0) sine(30) sine(90) |

# ex29  
# サイン関数を使って明るさ変化

# colors7 = [red, orange, yellow, green, sky, blue, purple] はboot.pyで定義済です。

|  |
| --- |
| phase = 0 while True:  powers = [(sine(phase)+1)/2 for i in range(7)]   setcolors(np, colors\_powersed(colors7, powers))  np.write()  time.sleep(0.001)  phase = phase + 5  phase = phase % 360 |

# ex30  
# サイン関数を使って明るさ変化（色ランダム）

# colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]は、boot.pyで定義済です。

# phase += 5 は、 phase = phase + 5 と同じです。

# phase %= 360は、phase = phase % 360 と同じ意味です。

# 一度では、コピペできないので・・・main.py に書いて、アップロードしましょ。

|  |
| --- |
| phase = 0 while True:  c = [colors[random(11)] for i in range(7)]  for i in range(72):  powers = [(sine(phase)+1)/2 for i in range(7)]   setcolors(np, colors\_powersed(c, powers))  np.write()  time.sleep(0.001)  phase += 5  phase %= 360 |

# ex31  
# サイン関数を使って明るさ変化（色ランダム、位相ずらし）

|  |
| --- |
| phase = 0 while True:  c = [colors[random(11)] for i in range(7)]  p = [1,0,1,0,0,1,0]  for i in range(72):  powers = [(sine(phase + p[i]\*90)+1)/2 for i in range(7)]   setcolors(np, colors\_powersed(c, powers))  np.write()  time.sleep(0.001)  phase += 5  phase %= 360 |

# ex32  
# サイン関数を使って明るさ変化（色ランダム、位相ランダムずらし）

|  |
| --- |
| phase = 0 while True:  c = [colors[random(11)] for i in range(7)]  p = [random(2) for i in range(7)]  for i in range(720):  powers = [(sine(phase + p[i]\*90)+1)/2 for i in range(7)]   setcolors(np, colors\_powersed(c, powers))  np.write()  time.sleep(0.001)  phase += 5  phase %= 360 |

# ex33  
# npオブジェクトを、もう１つ作ってみる

|  |
| --- |
| clear() np2 = neopixel.NeoPixel(machine.Pin(5),7) np2[0] = red np2.write() |

# ex34  
# np と np2 を交互に点灯

|  |
| --- |
| np[0],np[1] = black,sky np2[0],np2[1] = red,black while True:  np.write()  time.sleep(0.5)  np2.write()  time.sleep(0.5) |

参考情報

プログラミング言語（MicroPython）

<https://github.com/micropython/micropython/tree/master/docs/esp8266/tutorial>

<http://docs.micropython.org/en/latest/esp8266/tutorial/>

ESP-WROOM-02

|  |
| --- |
| <https://akizukidenshi.com/catalog/g/gM-09607/> |

フルカラーLED（NeoPixel）(マイコン内蔵ＲＧＢ　ＬＥＤ　ＷＳ２８１２Ｂ)

|  |
| --- |
| <https://akizukidenshi.com/catalog/g/gI-07915/> |

シリアルーUSBコンバーター (ＦＴ２３４Ｘ　超小型ＵＳＢシリアル変換モジュール)

|  |
| --- |
| <https://akizukidenshi.com/catalog/g/gM-08461/> |

ファームウエアのダウンロード

|  |
| --- |
| <https://micropython.org/download/esp8266/>[esp8266-20200911-v1.13.bin](https://micropython.org/resources/firmware/esp8266-20200911-v1.13.bin)  （最新版をダウンロードします） |

ファームウエアを書き換えるためのツールの説明

|  |
| --- |
| [http://docs.micropython.org/en/latest/esp8266/tutorial/intro.html#deploying-the-firmware](http://docs.micropython.org/en/latest/esp8266/tutorial/intro.html" \l "deploying-the-firmware) |

ファームウエアを書き換えるツールのインストール（ubuntu）

|  |
| --- |
| sudo apt install python3-pip  pip3 install esptool  (sudo apt install esptool) |

ファームウエアの書き換え（ubuntuの場合）

|  |
| --- |
| まず、ジャンパピンをG側とショートして電源を供給して次のコマンド  sudo .local/bin/esptool.py --port /dev/ttyUSB0 erase\_flash  電源を切って、再度電源を供給して次のコマンド（１行です）  sudo .local/bin/esptool.py --port /dev/ttyUSB0 --baud 460800 write\_flash --flash\_size=detect 0 esp8266-20170108-v1.8.7.bin  成功したら、ジャンパピンは元に戻しておきます。 |

シリアルポートのつなぎかた

写真のように、シリアル-USBコンバーターのRx を基板のTxに、Txを基板のRxに接続（GNDはGに）します。PCの端末アプリ（Windows はTeraterm, Macは、端末からscreenコマンド、Ubuntu は minicom など）を使います。8bitノンパリティー 11520bps

ubuntuの例

|  |
| --- |
| minicom -D /dev/ttyUSB0 |

WebREPLを使うための設定

シリアルポートをつないだあと、

|  |
| --- |
| >>> import webrepl\_setup  WebREPL daemon auto-start status: enabled  Would you like to (E)nable or (D)isable it running on boot?  (Empty line to quit)　→　E を入力します。  この後、パスワード設定を聞いてきます（本機では xmas としました）。 |

シリアルポート経由（REPL）でファイルを書き込む方法

|  |
| --- |
| f = open(“ファイル名”, mode=”w”)  Ctrl+E でコピペモードにします。  ===f.write(”””　　　←手で入力  ===ここに、ファイルにしたい内容をコピペします  　（WebREPLと違い、行数に制限ない）  ===”””)　　　　　　←手で入力  Ctrl+D でモードを終了します。  f.close()  以上で、ファイルとしてボードの中のフラッシュメモリに保存されます。 |

保存されているファイル名の表示方法

|  |
| --- |
| >>> import uos  >>> uos.litdir() |

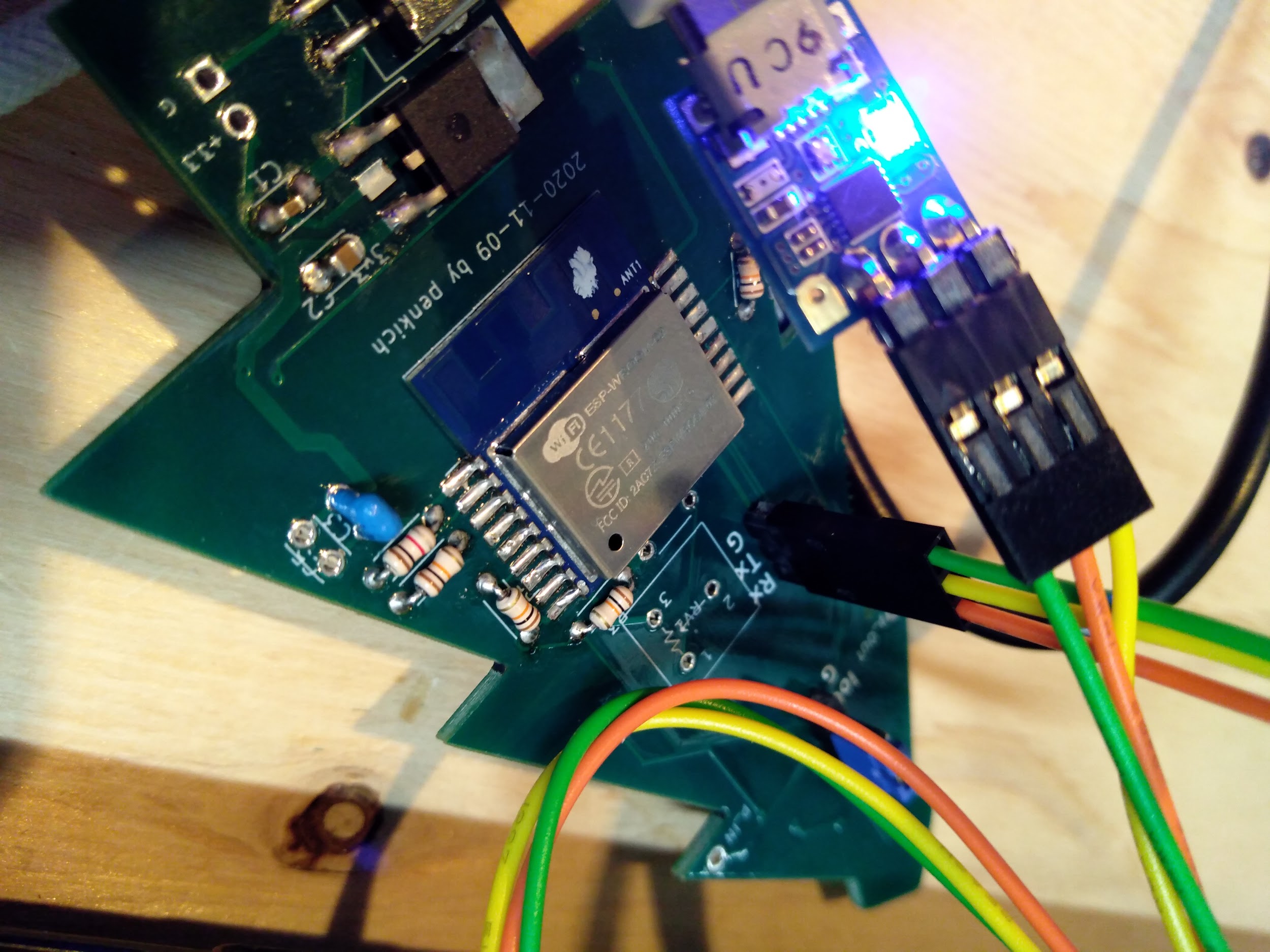
example1.py（サンプルプログラムです。ご参考）

|  |
| --- |
| def pow(i):  return abs(math.sin(math.pi\*i/200))  while True:  ps = [randint(200) for i in range(3)]  c = [colors[randint(11)] for i in range(7)]  for i in range(200):  setcolors(np, colors\_powered(c, pow(i)))  np[0]=[int(white[0]\*pow(i+ps[0])),int(white[1]\*pow(i+ps[1])),int(white[2]\*pow(i+ps[2]))]  np.write()  time.sleep(0.01) |

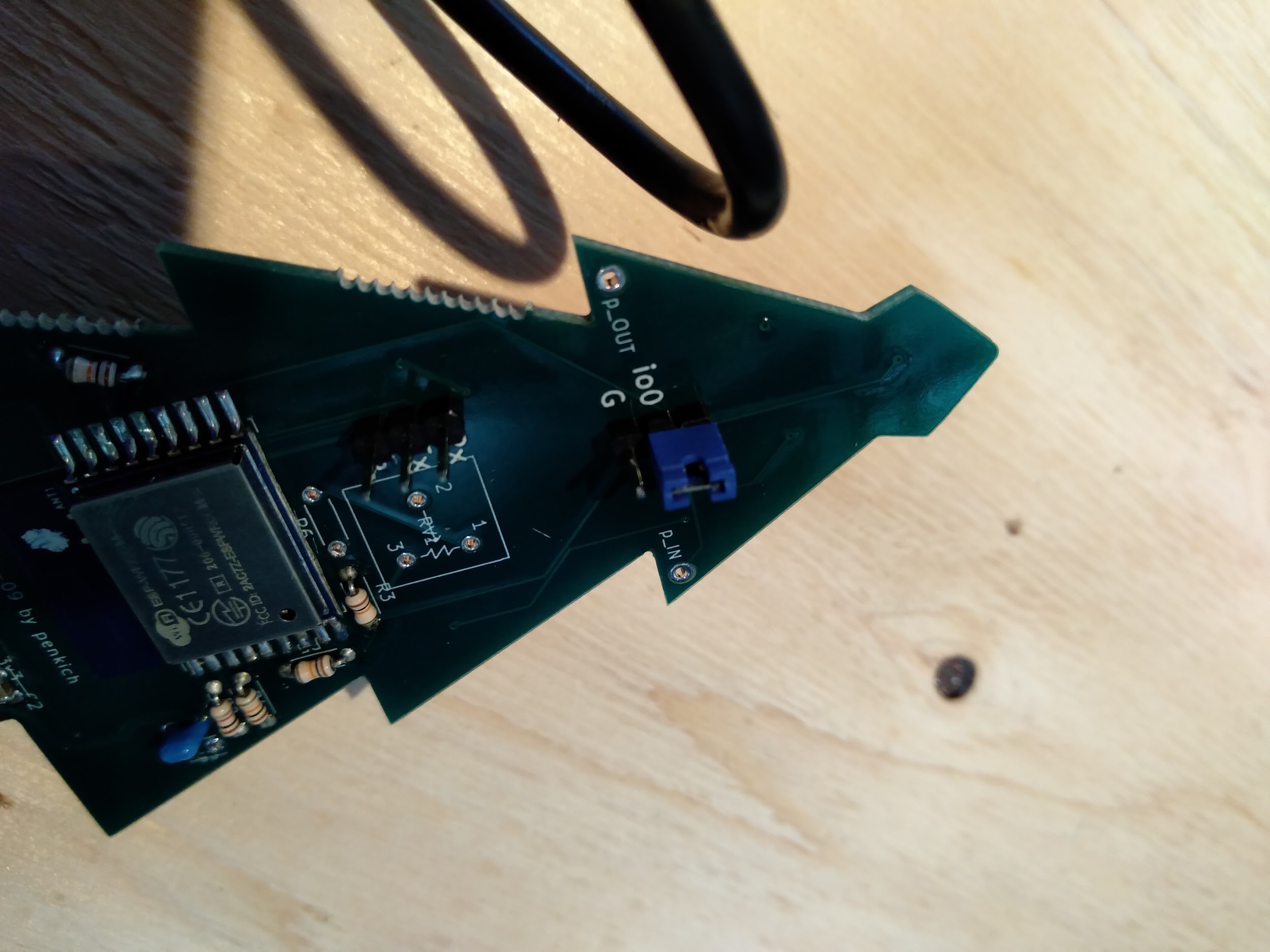
boot.py

|  |
| --- |
| # This file is executed on every boot (including wake-boot from deepsleep)  #import esp  #esp.osdebug(None)  import uos, machine  #uos.dupterm(None, 1) # disable REPL on UART(0)  import gc  import webrepl  webrepl.start()  gc.collect()  import network  sta\_if = network.WLAN(network.STA\_IF); sta\_if.active(False)  # 2020-12-03 Nishijima  import sys  def reload(mod):  mod\_name = mod.\_\_name\_\_  del sys.modules[mod\_name]  gc.collect()  return \_\_import\_\_(mod\_name)  import neopixel, machine, time, urandom, math  np = neopixel.NeoPixel(machine.Pin(5),7)  red = [120, 0, 8]  orange = [90, 20, 0]  yellow = [80, 60, 0]  ygreen = [50, 80, 0]  green = [0, 70, 5]  tur = [0, 70, 20]  blue = [0, 0, 160]  bluep = [20, 0, 100]  purple = [50, 0, 120]  white = [60, 60, 60]  sky = [0, 70, 70]  black = [0,0,0]  colors7 = [red,orange,yellow,green,sky,blue,purple]  colors=[red,orange,yellow,ygreen,green,tur,sky,blue,bluep,purple,white]  def clear():  np.fill(black)  np.write()  def setcolors(np, colors):  for i,color in enumerate(colors):  np[i] = color  def colors\_powered(colors, power):  return [[int(colors[i][0]\*power), int(colors[i][1]\*power), int(colors[i][2]\*power)] for i in range(7)]  def colors\_powersed(colors, powers):  return [[int(colors[i][0]\*powers[i]), int(colors[i][1]\*powers[i]), int(colors[i][2]\*powers[i])] for i in range(7)]  randint = lambda n : urandom.getrandbits(12) % n  sine = lambda n : math.sin(n/180 \* math.pi) |

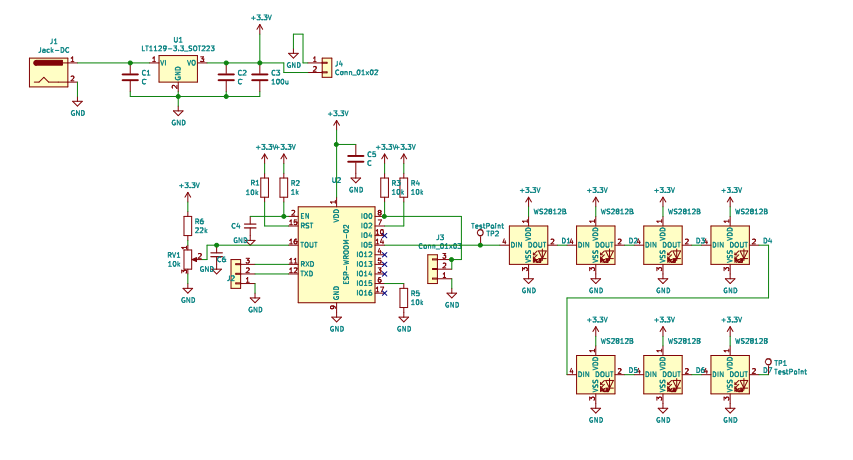
シリアルポートのつなぎ方



通常のジャンパーピンの位置（io0 は、Gにつながない）



回路図（未実装の部分や違う実装部分あります）



main.py 以外のプログラム(ex. test.py）の実行方法

|  |
| --- |
| >>> import test |

但し、最初のimport時のみ実行されるので、再度実行したい場合は、reload()します。

なお、reload() は、boot.py で定義しています。

|  |
| --- |
| >>> reload(test) |