Pythonを用いたIO制御

USBデバイスの活用

USBデバイスを活用するには一般的にドライバと呼ばれるハードウェアとアプリケーションの間に入ってデータの授受を手助けする機能を持ったモジュールが必要になる.

が，非同期送受信プロトコルの一つであるUART（Universal Asynchronous Receiver/Transmitter）については一般的なデバイスのドライバはOSに含まれていたり，ドライバの入手性が良い場合が多い.

> よく知られているRS323Cなども非同期送受信プロトコルのうちの一つ”

今回はRaspberry Pi OSにドライバが含まれているFTDI製「FT232R」が搭載されているUSB-UARTデバイスであるRFIDリーダーを用いて，シリアル通信デバイス制御のイメージを掴む.

※Silicon Labs製「CP2102」も多くのUSB-UARTデバイスに採用されているのに加えて，Raspberry Pi OSでもドライバのインストールなしで活用できる.

Raspberry piでのUSB-UARTデバイスの扱い方

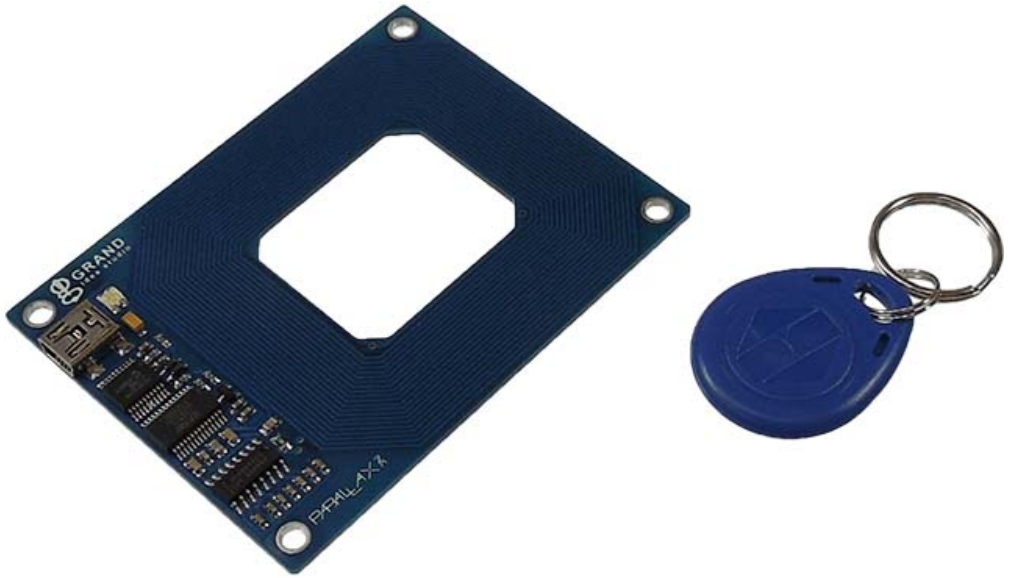
本体にデバイスを接続した際，認識したかどうかは`lsusb`コマンドで確認することができる.

USB-UARTデバイスは本体に接続後，デバイスの認識が終わると`/dev/ttyUSBX`（`X`は0から始まる連番の数字）にデバイスファイルが現れる.
プログラムからデバイスを利用する場合はデバイスファイル名を指定して利用する.

USB-UARTを利用できるRFIDリーダーを利用する

下記のデバイスを利用して実習する.

> 秋月電子通商｜ＵＳＢ接続ＲＦＩＤリード（読み込み）モジュール（タグサンプル付き）
> https://akizukidenshi.com/catalog/g/gM-06826/



- RFIDのリード(読み込み)に対応したモジュール基板です。付属のRFIDタグ(円盤型、カード型、キーホルダー型)の他に、125kHz EM4100のタグに対応します。通信はUSBで行いますので、PCでの使用に便利です。（Parallax Inc.製）
- 仕様
  - 電源：+5V(アイドル時：10mA以下、動作時：100~200mA)
  - 通信方式：USB(仮想COMポート接続：2,400bps,N,8,1 設定)※1
  - 基板寸法：6.2x8.26cm
- セット内容
  - RFIDのリードモジュール基板本体 x1個
  - 円盤型RFIDタグ(直径約50mm) x1枚
  - 円盤型RFIDタグ(直径約25mm) x1枚
  - カード型RFIDタグ x1枚
  - キーホルダー型RFIDタグ x1個
  - USBミニBケーブル 1本

利用手順

- 本体にRFIDリーダーを接続する
- USBデバイスが認識されていることを確認するとともに，デバイスファイル名を確認する
  - `lsusb` コマンドを実行してUSBデバイスが認識されていることを確認する
  - `/dev/ttyUSB0` が存在していることを確認する
- pyserialモジュールをインストールする
  - `$ python3 -m pip install pyserial`
- シリアル通信プログラムをpythonで実装する
- タグを読んでタグ固有のIDを取得する

USBデバイスの認識状況の確認

接続前：

```
$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

接続後：

```
$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0403:6001 Future Technology Devices International, Ltd FT232 Serial (UART) IC
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

シリアル通信を行うコード

```
import serial

PORTNAME = "/dev/ttyUSB0"
BAUDRATE = 2400

if __name__ == '__main__':
    port = serial.Serial(PORTNAME, BAUDRATE, timeout=None)
    try:
        while(True):
            line = port.readline()
            code = str(line.decode().replace('\r\n',''))
            print(code)
    except KeyboardInterrupt:
        print('interrupted!') # Ctrl-Cが入力された場合の処理
        pass
    port.close()
    print("Done")
```

Raspberry PiでのIO制御

Raspberry PiにはIO（GPIO：General-purpose input/output）ピンがヘッダによって引き出されており，それぞれのピンについてON/OFFの制御を行うことができる．

ピンを制御することにより，外部デバイスのON/OFFを取得・制御したり，外部のセンサやデバイスとI2CやSPI，UARTなどのプロトコルを用いて通信を行うことができる．

今回はRaspberry PiからGPIOを利用して，外部に接続したLEDの制御を行うためのプログラムの実装を行う．

> GPIOのON/OFF制御はハードウェア（CPU）に近いレイヤーで実現する機能であるため，CPUのほかMMU，チップセットなどを備えた基盤上で稼働するOSからアクセスすることは非常に難しく，一般的なPCでは実現しにくいアプリケーションである．
> （最近では見かけなくなったが）パラレルポートなどがIO制御を行うためのインターフェイスとしてかつて存在していた

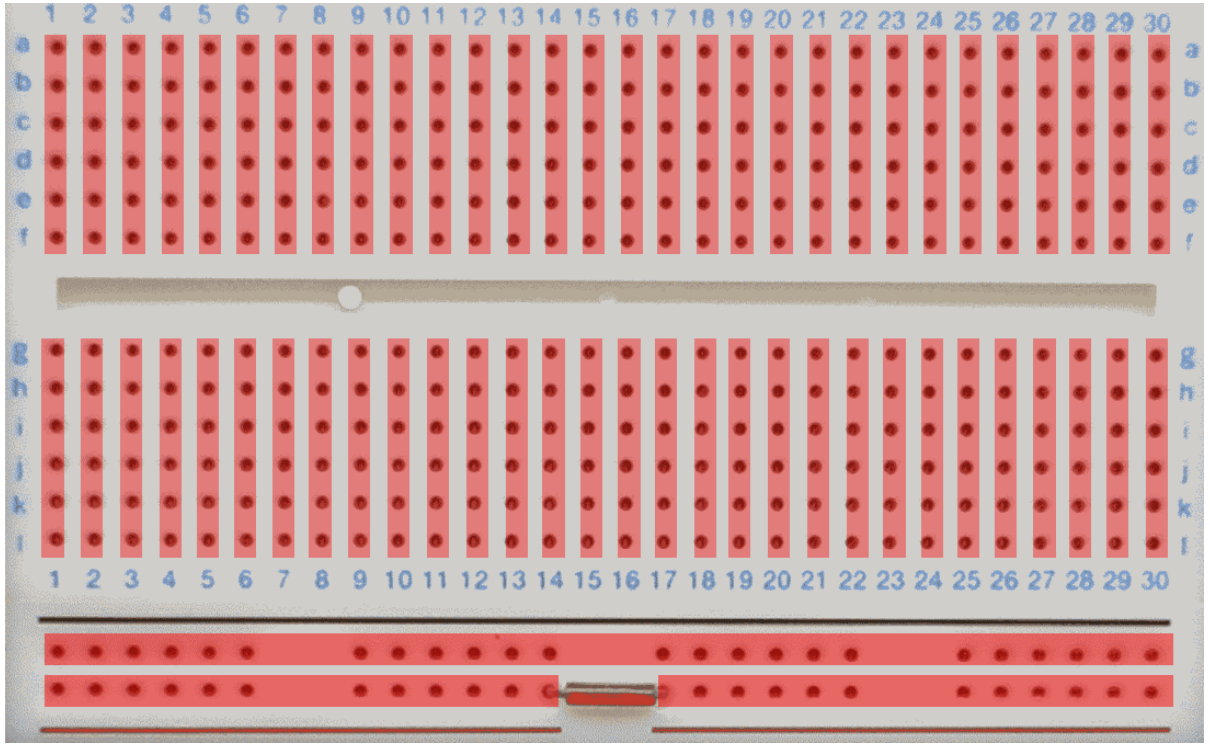
GPIO制御によるLチカ

> Lチカ：
> LEDをつないで点滅させる作業のこと．新型のマイコンなどを購入した場合などにmIO制御のコードを実装する練習としてよく行う．

Raspberry Pi本体とLEDを配線したブレッドボードを接続し，プログラムを用いて点灯させる．

ブレッドボードの仕組み配線

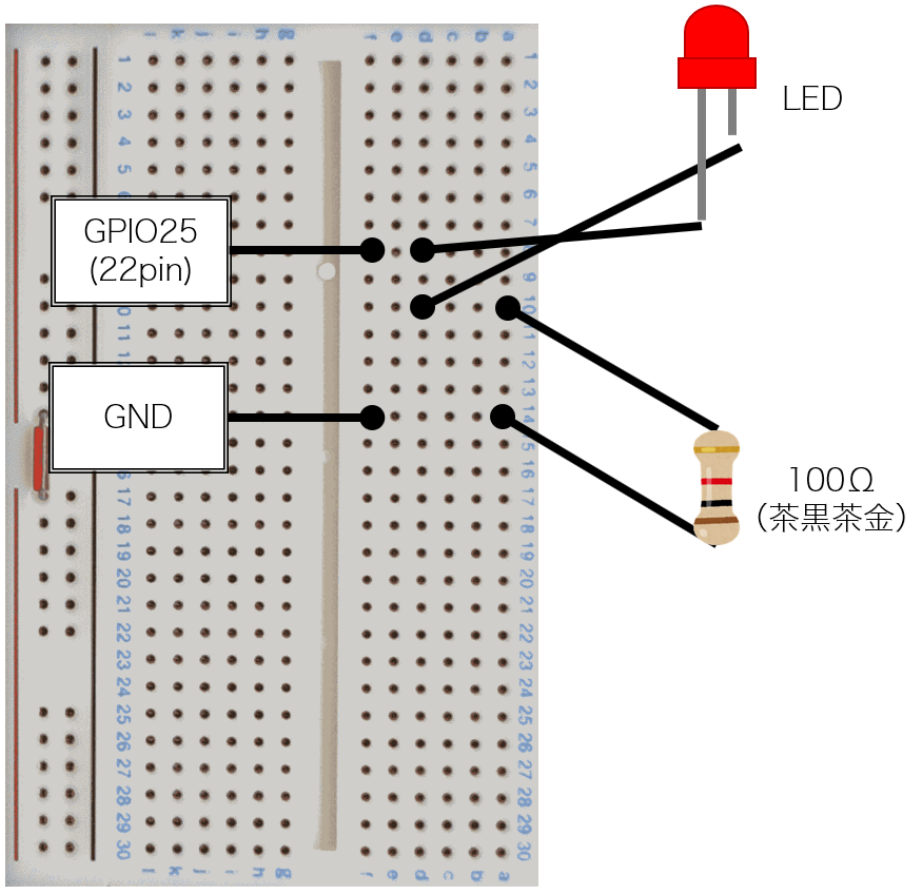
ブレッドボードは左図の方向に結線済みの回路が組まれている.



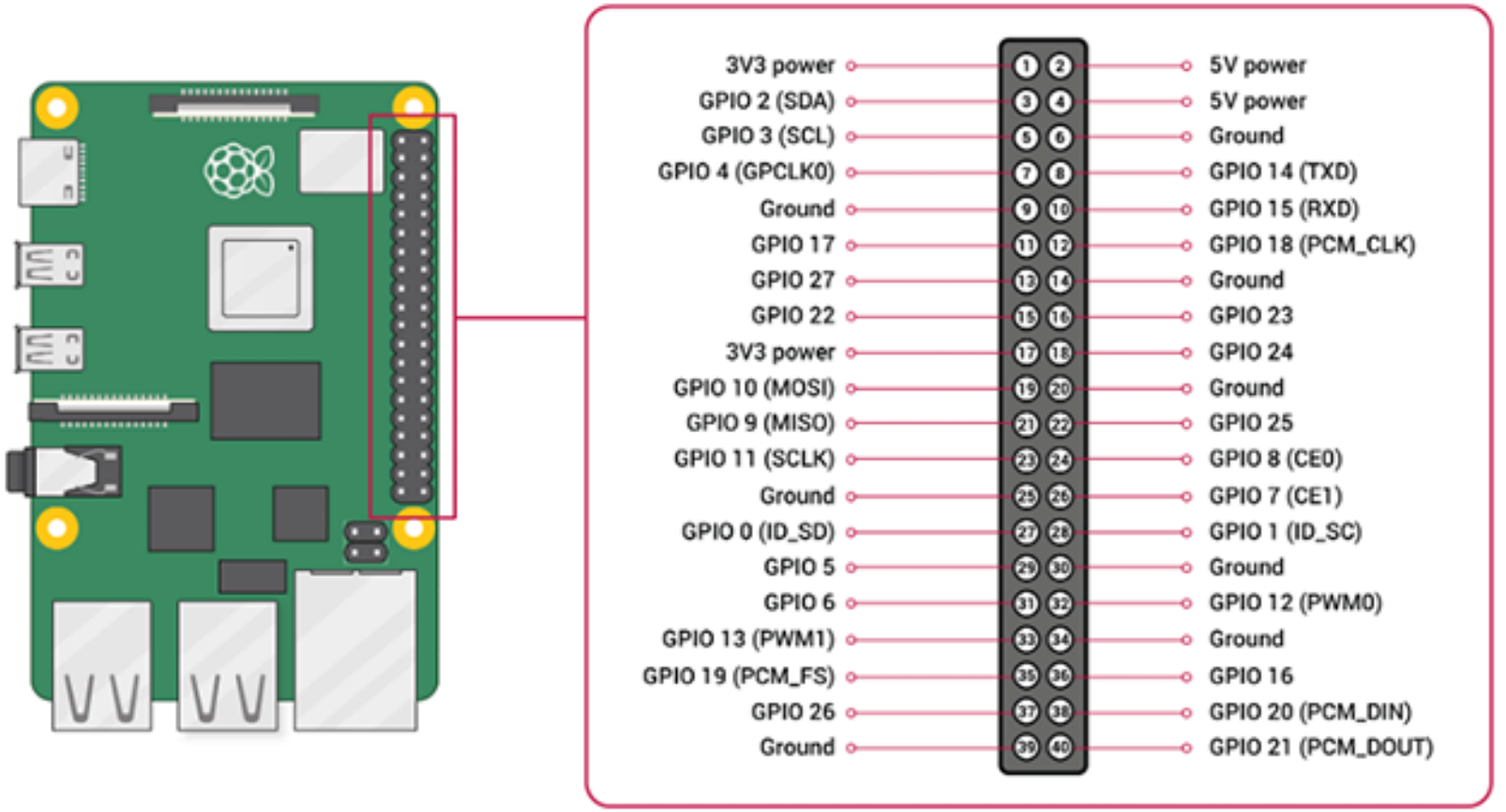
プロトタイピングなどの際に回路を簡単に構成するために用いられる.

Raspberry Piとブレッドボードの接続

図の通り結線する



【参考】Raspberry PiのGPIOピンアサイン



GPIO制御プログラム

```
import RPi.GPIO as GPIO
from time import sleep

if __name__ == '__main__':
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(25, GPIO.OUT)     #GPIO25を出力に設定
    try:
        while True:
            GPIO.output(25, GPIO.HIGH)   # Highを0.5s出力
            sleep(0.5)
            GPIO.output(25, GPIO.LOW)    # Lowを0.5s出力
            sleep(0.5)
    except KeyboardInterrupt:
        # `KeyboardInterrupt`をハンドリングして,
        # 終了時に必ず`GPIO.cleanup()` が実行されるようにする.
        pass
    GPIO.cleanup()
```

GPIO制御プログラムの解説

GPIO25を出力に設定し，Highを0.5S，Lowを0.5s交互に出力することで，22番ピンに接続されたLEDが点滅する.

プログラム実行後，`Ctrl+C`でアプリケーションを停止することを考慮して`try/except`文を用いて`KeyboardInterrupt`をハンドリングして，終了時に必ず`GPIO.cleanup()`が実行されるように設計している.

GPIOのその他の応用

今回はGPIOピンを出力に設定して外部の機器を駆動させたが，入力に設定して外部機器のステータスを取得することも可能．入力ピンへの入力電圧がVcc（3.3V）を超えると本体が故障するため注意が必要となる.

また，ポートから出力可能な電流はすべてのポートをあわせて50mAであり，モータなどの電流を要する負荷は直接接続することができないため，モータドライバなどを用いる必要がある.
（駆動電圧の異なる負荷はリレーユニットやフォトカプラ，ソリッドステートリレーを噛ませる）

おわりに

今回はRaspberry Piのセットアップから，実践的な pythonプログラミング方法やアプリケーション開発手法をお話しました.

が，今後実際にプログラムを行う場合に躓きやすい分野であり，プログラムの設計そのものと密接に関わる「データ構造」や「アルゴリズム」，「オブジェクト指向」に関する話題については実践的であるが故に時間の関係で割愛しています.

今後業務などでプログラミングを行う場合にはこのあたりの知識を備えておかないと，壁を感じることになると思います.

プログラミングを行いたい・興味が出てきたという方は，pythonを足がかりにしてこれらの書籍やWebの資料を通して学習してみることをおすすめします.

自身が実装したいことを適切に検索するとサンプルコードが容易に入手できる時代です.
サンプルコードを自分流に書き換えることができる応用力を備えて，省力化やDXの推進をプログラムを用いて実践してみてください.