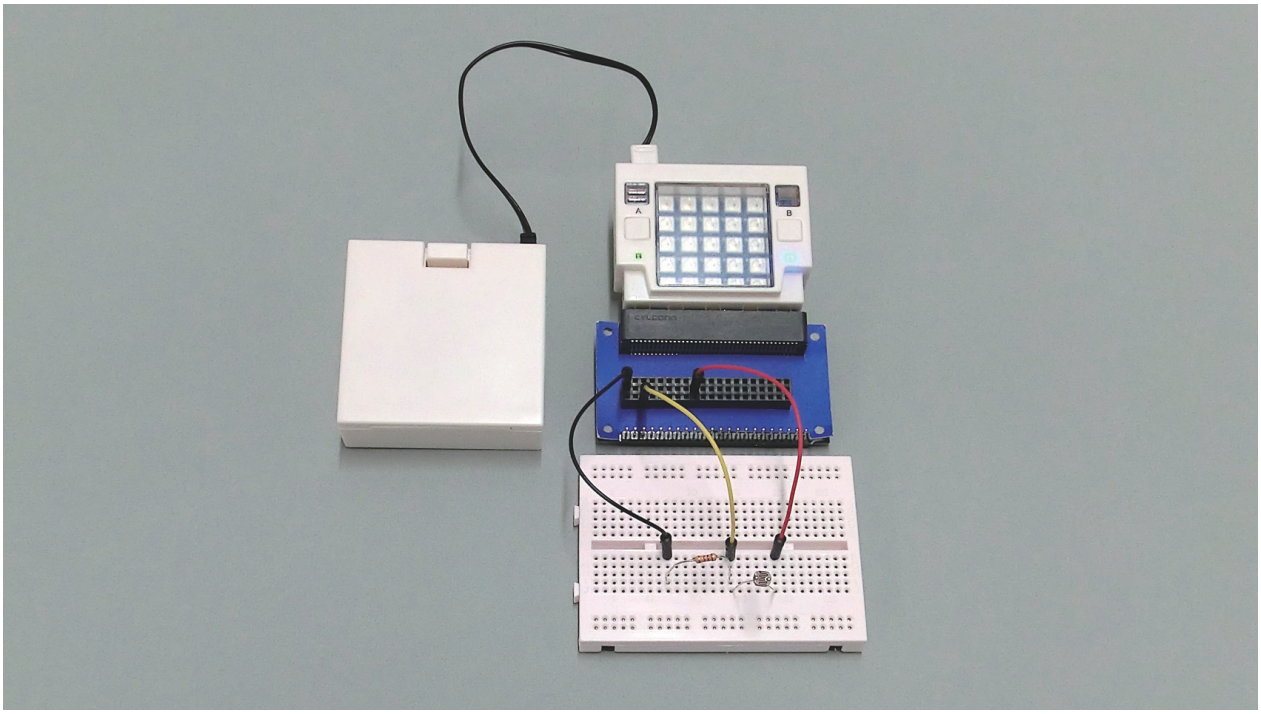
Studuino®:bit
スタディーノ ビット

高校情報Ⅱキット
安否確認システム(IoT)

教材監修:元 東京都立日比谷高等学校教諭／元 東京学芸大学特任教授 天良和男先生

教員用



目 次

準 備

プログラム開発環境のインストール …… 3
複数のプログラムを転送した場合の実行方法 …… 8

演 習

演習① 光センサのアナログ入力プログラム …… 9
演習② 安否確認システムの構築 …… 11
smtp_config 関数について …… 17
sendmail 関数について …… 18
wifi_config 関数について …… 19
wifi_ifconfig 関数について …… 19

付 録

付録 A. ブレッドボードの使い方 …… 24
付録 B. 変換エッジコネクタの使い方 …… 24
付録 C. デジタル入出力・アナログ入力オブジェクトの機能一覧 …… 25
付録 D. モバイルホットスポットの使い方 …… 26
モバイルホットスポットの設定画面に表示されるIPアドレスについて …… 29
付録 E. Wi-Fi機能の一覧 …… 30

本テキストに掲載しているプログラムは下記ページからダウンロードできます。

https://www.artec-kk.co.jp/dl/93275/

準備 プログラム開発環境のインストール

1.ソフトウェアのインストール

PythonでStuduino:bitのプログラムを作成するために必要なプログラム開発環境をインストールします。ここではStuduino:bit用に機能拡張されたMuエディタを使用します。下記ArtecRobo2.0のPythonページ内にあるドキュメントを参考に、Studuino:bit用Muエディタをダウンロード、インストールしてください。

https://www.artec-kk.co.jp/artecrobo2/ja/software/python.php

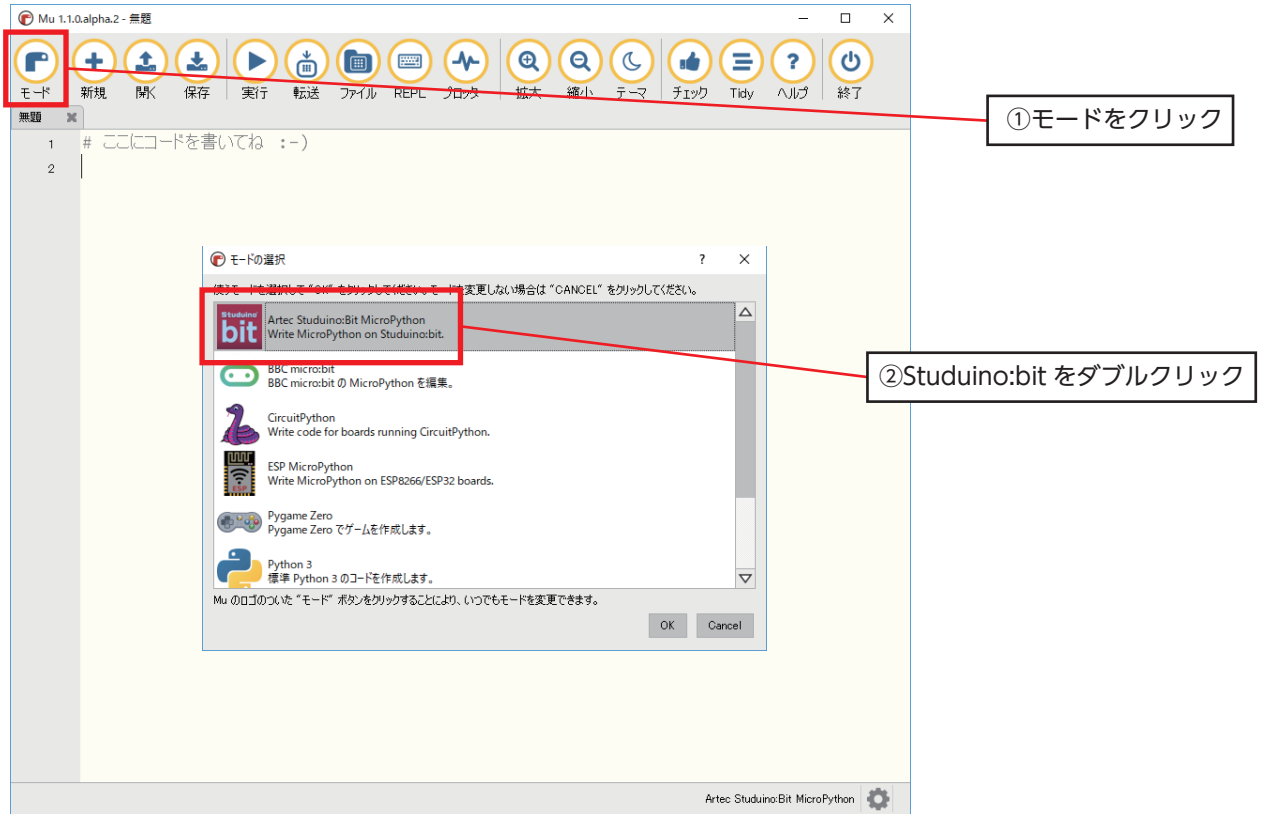
Studuino:bitなどのボード型コンピュータで動作するPythonは、普通のPythonではなく、MicroPythonです。MicroPythonはStuduino:bitなどのボード型コンピュータで実行できるように実装されたPython言語の処理系で、Python3.0との間で高い互換性を有します。MicroPythonを使ったプログラムはMuエディタなどを使って作成します。

2.REPL でプログラムを実行する

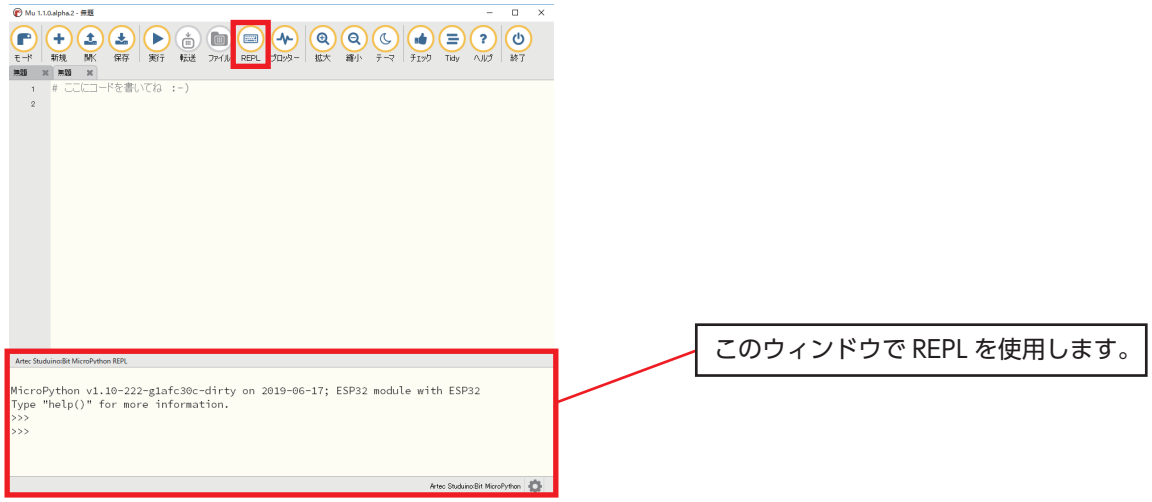
Python の対話型インタープリタREPL(Read-Evaluate-Print Loop)を使用する手順を下記に記します。

① Studuino:bitとPCを接続します。

② Muを起動して、下図のように「モード」ボタンをクリックして、「Artec Studuino:bit MicroPython」をダブルクリックしてください。



③「REPL」ボタンをクリックすると、Muウィンドウの下にウィンドウが表示され、入力待ちを示す「>>>」が表示されます。REPL表示中は「転送」ボタン、「ファイル」ボタンは灰色表示になり使用できません。
再度「REPL」ボタンをクリックすることでウィンドウが非表示になり、「転送」ボタン、「ファイル」ボタンを使用できるようになります。

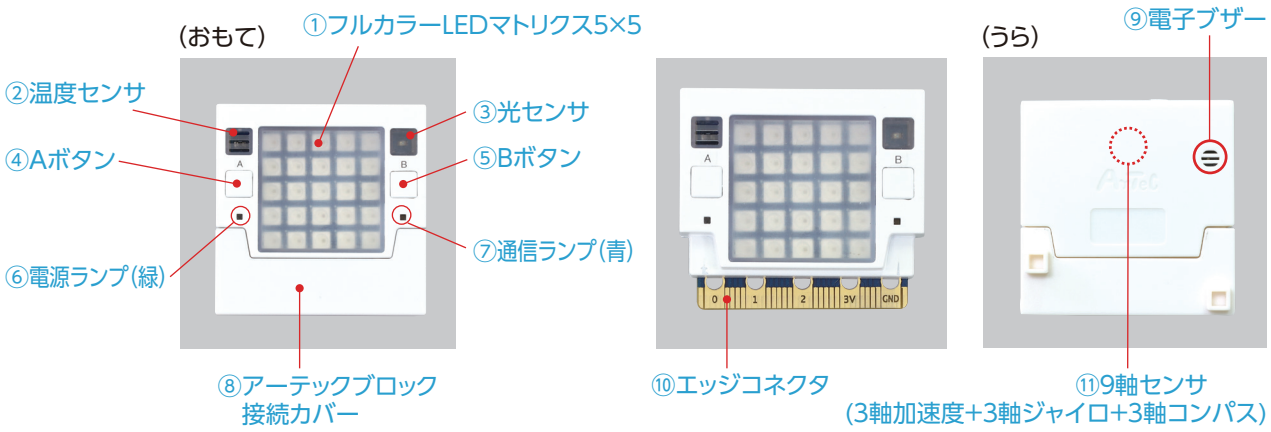


④REPLの自動エコーによって、ウィンドウの「>>>」の後にプログラムを入力すると、すぐに結果を見ることができます。REPLで下記を実行し、"Hello World"が表示されることを確認してください。

```
>>>print('Hello World')
```

3.Studuino:bit ライブラリ

Studuino:bitを制御する、オブジェクトや関数をまとめたモジュールとしてStuduino:bitライブラリを提供しています。Studuino:bitは下図のハードウェアを持っています。



Studuino:bitライブラリを使用することで、Studuino:bitのハードウェアを簡単に制御できます。
ライブラリには、Studuino:bitの各ハードウェアに対応したオブジェクトが定義されています。

番号	ハードウェア	オブジェクト	機能概要
①	フルカラーLEDマトリクス５×５	display	各LEDのON/OFFやLEDを使ったアニメーションなど
②	温度センサ	temperature	温度センサの値の取得
③	光センサ	lightsensor	光センサの値の取得
④	Aボタン	button_a	AボタンのON/OFF判定など
⑤	Bボタン	button_b	BボタンのON/OFF判定など
⑨	電子ブザー	buzzer	電子ブザーで鳴らす音の高さをON/OFFなど
⑩	エッジコネクタ	p0～p16、p19、p20	エッジコネクタの入出力制御
		pin0～pin3	micor:bit 互換エッジコネクタの入出力制御 (※)
⑪	３軸加速度センサ	accelerometer	加速度センサの値の取得
	３軸ジャイロセンサ	gyro	ジャイロセンサの値の取得
	３軸地磁気センサ	compass	地磁気センサの値の取得

※:pin0とp0、pin1とp1、pin2とp2、pin3とp3の併用はできません。

※:pin0～pin3オブジェクトは、p0～p3オブジェクトと同じメソッドが使用できますが、read_analog()メソッドは0～1023の値を返します。

各オブジェクトは、pystubit.boardモジュールに定義されています。

REPLで下記を実行し、Studuino:bitメインユニットの画面に文字列”Hello, MicroPython”がスクロール表示されることを確認してください。

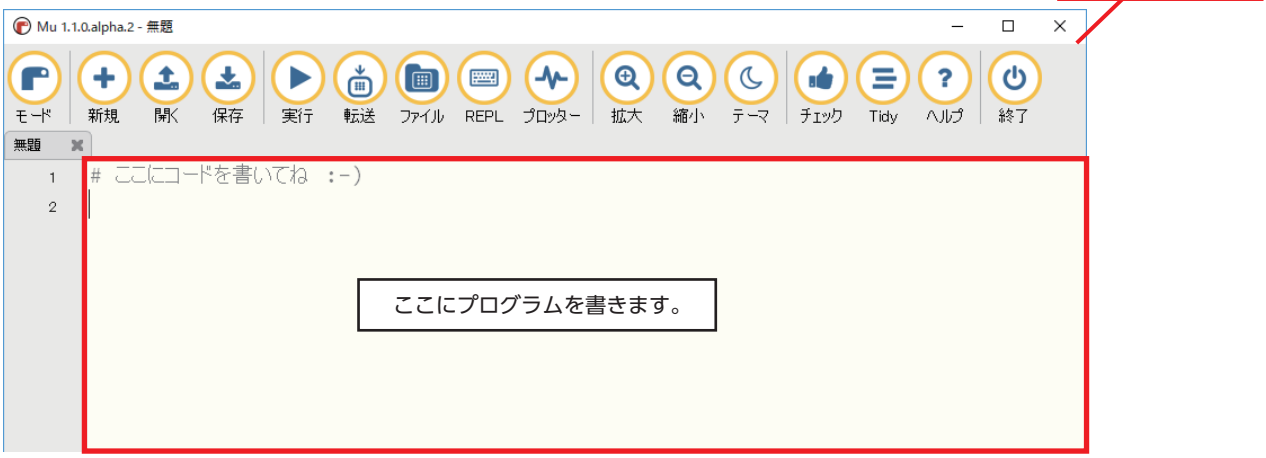
```
>>> from pystubit.board import display
>>> display.scroll('Hello, MicroPython')
```

4.「実行」ボタンでプログラムを実行する

Muエディタで編集したPythonスクリプトをStuduino:bitで実行する手順を下記に記します。

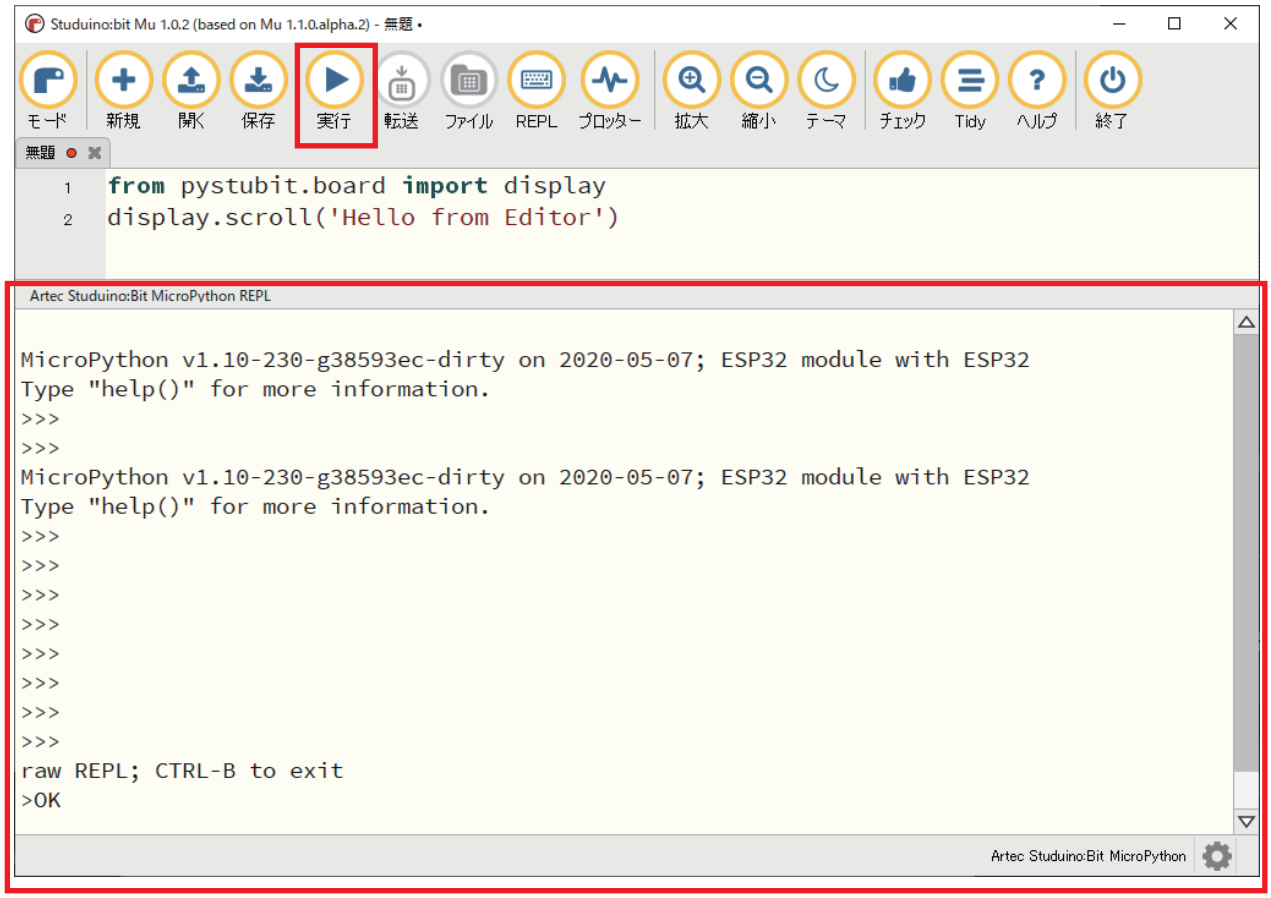
①下記をテキストエディタに入力します。

```
from pystubit.board import display
display.scroll('Hello from Editor')
```



② 「実行」ボタンをクリックするとStuduino:bitでプログラムが実行されます。

Studuino:bitメインユニットの画面に文字列”Hello from Editor”がスクロール表示されることを確認してください。



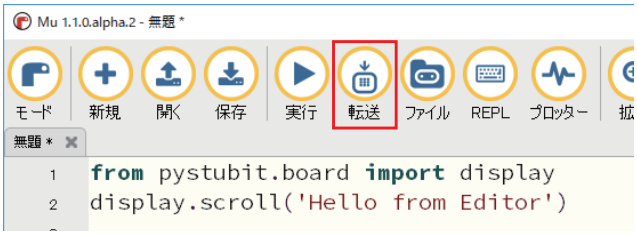
「実行」ボタンをクリックすると下側にウィンドウが表示され、REPLを使用できるようになります。

5.「転送」ボタンでプログラムを実行する

Muエディタで編集したPythonスクリプトをStuduino:bitに転送し、実行する手順を下記に記します。

① 4.①のプログラムをテキストエディタに入力し、「転送」ボタンをクリックします。

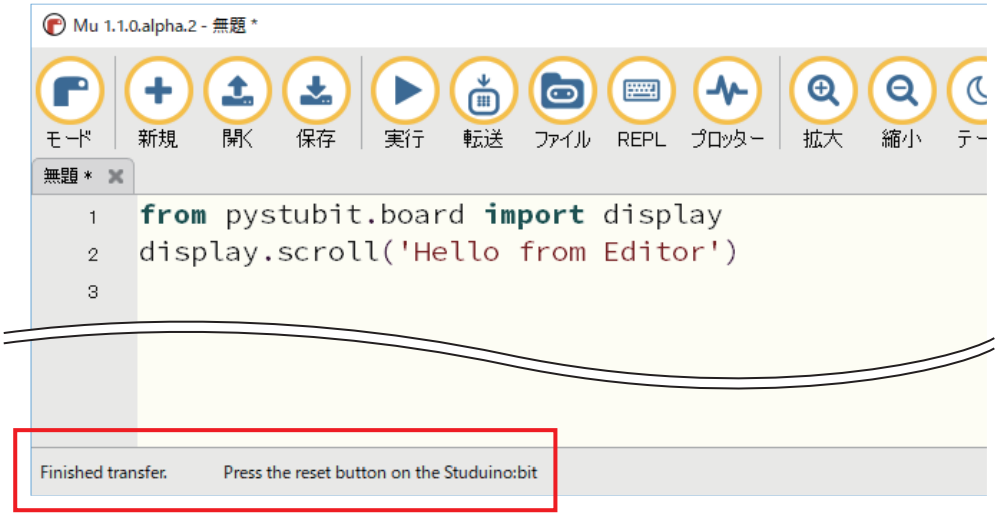
※「転送」ボタンが無効（灰色表示）になっている場合は「REPL」ボタンをクリックし、REPLウィンドウを閉じることで「転送」ボタンが有効になります。



②表示されるダイアログで、プログラムの転送先0～9のいずれかの「Transfer」ボタンをクリックします。



ステータスバーに「Finished transfer.」と表示されたら、転送完了です。



Studuino:bitメインユニットの画面に文字列”Hello from Editor”がスクロール表示されることを確認してください。表示終了後は、Studuino:bitメインユニットのリセットボタンを押すことで再度プログラムが実行されます。プログラムは転送されているため、USB ケーブルを取り外し、電池ボックスを接続してスタンドアローンで動作を確認できます。

●複数のプログラムを転送した場合の実行方法

複数のプログラムを 0 ～ 9の番号を指定して転送することができます。

通常は最後に転送したプログラムが実行されますが、以下の方法で番号を指定して転送済みのプログラムを実行することができます。



①電源を接続する（通常起動）

(1)電池ボックスもしくはUSBケーブルをメインユニットに接続します。
(2)電源ランプ(緑)が点灯し、電源が自動的にONになり起動します。
通常起動時は前に実行したプログラムが自動的に実行されます。



②プログラム選択モードへの移行

(1)Aボタンを押したまま、リセットボタンを押してください。リセットボタンを押すと再起動のため一時的に電源ランプが消灯します。

(2)リセットボタンを離して3秒後に電源ランプが再点灯したら、Aボタンを離します。

(3)緑のLEDで0が表示されるとプログラム選択モードへ移行しています。





③プログラムの選択・起動

(1)プログラム選択モード時にAボタンを押すと、表示が0・1・2・・・と切り替わります。9まで切り替わると0に戻ります。
(2)選択した番号でBボタンを押すと、各番号に割り振られたプログラムが実行されます。次回より通常起動時に(2)で選択したプログラムが自動で実行されます。

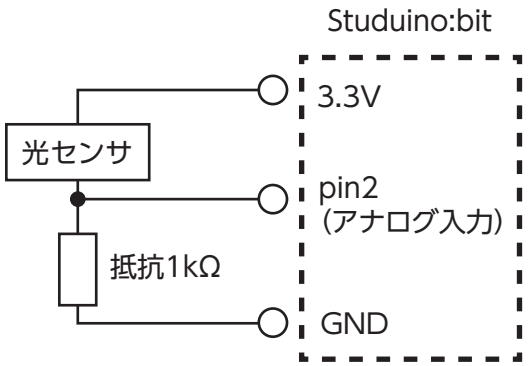


※工場出荷段階で動作確認用に各番号にはサンプルプログラムが書き込まれています。
※サンプルプログラムはソフトウェアを使用して新たなプログラムを書き込むと上書きされます。

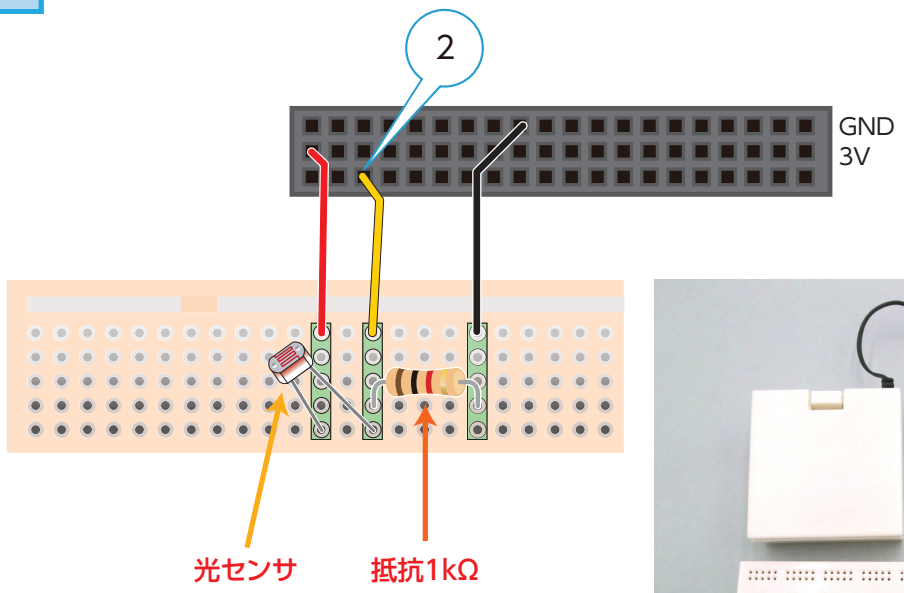
演習① 光センサのアナログ入力プログラム

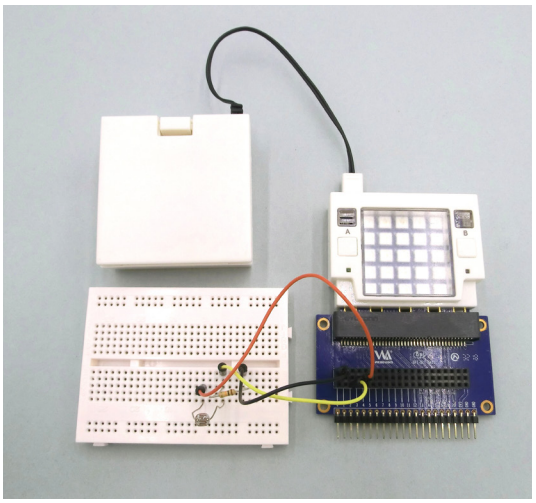
光センサで周囲の明るさを計測し、Studuino:bit に明るさを表示するプログラムを作成します。
Studuino:bit 本体内部には光センサが内蔵されていますが、ここでは外部に接続した光センサを使って計測する方法について説明します。

① 光センサとStuduino:bitを接続します。
ブレッドボードの使い方詳細は付録Aを参照してください。
変換エッジコネクタの使い方詳細は付録Bを参照してください。

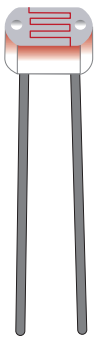


接続例





光センサ



使用するセンサはCDS(硫化カドミウム)セルと呼ばれる、明るさを計測するセンサです。
抵抗器の一種で、当たる光の量によって抵抗値が変化します。
センサによって形状や出力形式は異なります。

明抵抗	10～20kΩ(10Lux時)
暗抵抗	1MΩ

②Studuino:bitとPCを接続します。

③光センサのアナログ入力から取得したデジタル値を、1秒ごとにREPLに表示するプログラムを記します。
下記をテキストエディタに入力し、「実行」ボタンを押すと、REPLに0～1023の間の整数が表示されます。
手を使って光センサを覆い、光が当たらないようにすると値が変化することを確認してください。
値が変化しない場合はブレッドボードの接続が間違っていないか確認してください。
デジタル入出力・アナログ入力オブジェクトの機能の詳細は付録Cを参照してください。

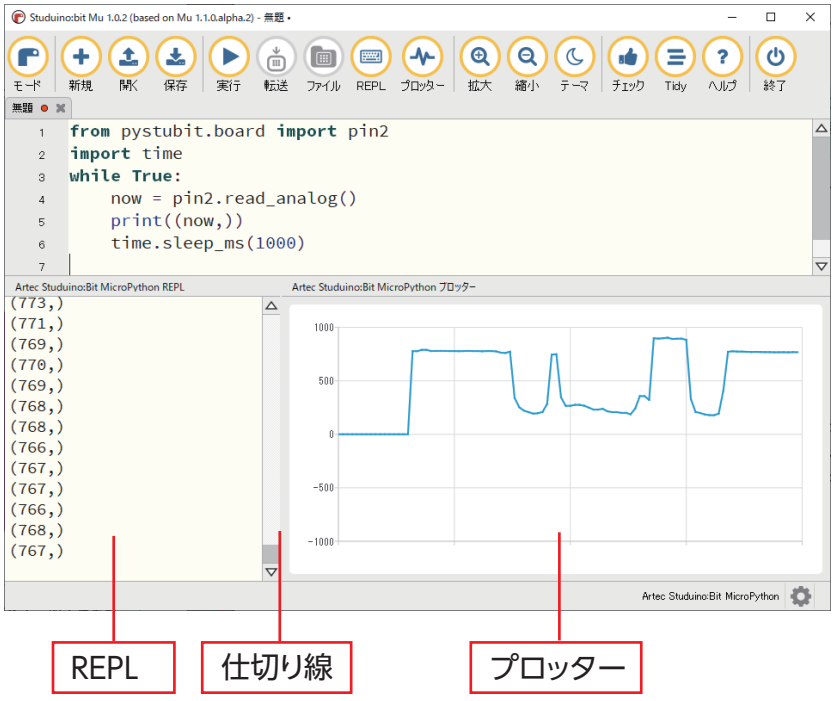
hikari.py

```
from pystubit.board import pin2
import time
while True:
    now = pin2.read_analog()
    print((now,))
    time.sleep_ms(1000)
```

ポートP2の値を読み取り、0(0V)から1023(3.3V)までの間の整数を返す。

```
Artec Studuino:Bit MicroPython REPL
(164,)
(705,)
(278,)
(225,)
(249,)
(340,)
(784,)
```

メニューの「プロッター」ボタンをクリックすると、プロッターに光センサの値がグラフ表示されます。
REPLとプロッターの間の仕切り線を左右にドラッグすると、それぞれの幅を変えることができます。
printの引数の「(now,)」は、タプル(一次元配列)で指定するため丸括弧()で囲み、要素が1個のタプルの末尾にはカンマ「,」が必要です。「print((now,))」を「print(now)」に変更すると、REPLには表示されますが、プロッターには表示されません。プロッターに表示するデータはタプルで指定する必要があります。
また、「print((now1,…now2,…now3))」のように指定すると、複数データのグラフを表示することもできます。



演習② 安否確認システムの構築

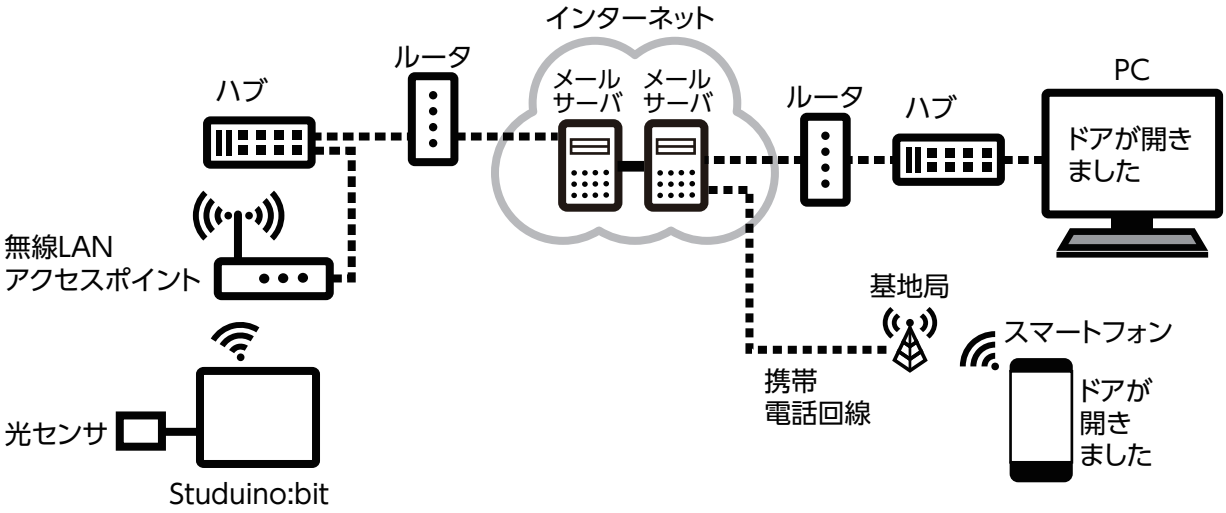
冷蔵庫内に設置した光センサが明るさの変化を検知することで、冷蔵庫のドアが開かれたことをメールで通知し、一人暮らしの高齢者の安否確認をするシステム作成します。
サーバとして、PythonのWebサーバではなく、プログラムからメールを送信する機能を持つGmailのメールサーバを用います。
また、冷蔵庫の代わりに白箱を用います。
※Gmailを利用するにはGoogleのアカウントが必要です。

作例動画はコチラ

https://youtu.be/8feQUq5Y7vg

※ブロックなどはセットに付属していません

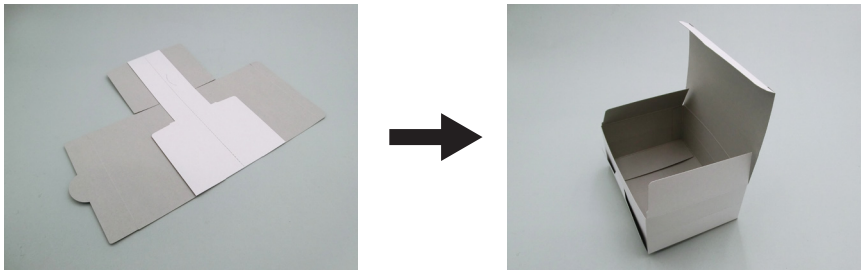
ネットワーク環境例



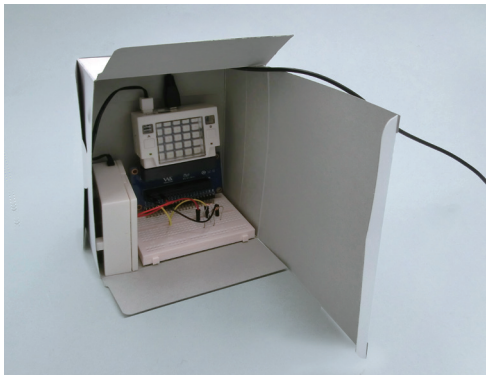
無線LAN アクセスポイントとしてWi-Fiルータを使用する場合は、Wi-Fiルータの取扱説明書を参照のうえ、ルータモード(RTモード)をOFFにして、アクセスポイントモード(APモード)に設定してください。
無線LAN アクセスポイントとしてモバイルホットスポットを利用する場合は、付録D を参照してください。
モバイルホットスポットとは、有線LAN で接続されたWindows10やWindows11のコンピュータをWi-Fiルータ化する機能です。なお、図中のルータやハブ、PCは、コンピュータ教室などに設置されている既存の装置を使用します。ハブの空きポートに無線LANアクセスポイントをLANケーブルで接続し、これにStuduino:bitを無線で接続します。また、メールサーバはプロバイダが所有する装置です。

1.しきい値の決定

①冷蔵庫の代わりとなる白箱を組み立てます。



②演習①のStuduino:bit、ブレッドボードを使用します。
Studuino:bitとPCを接続し、白箱の中に入れます。

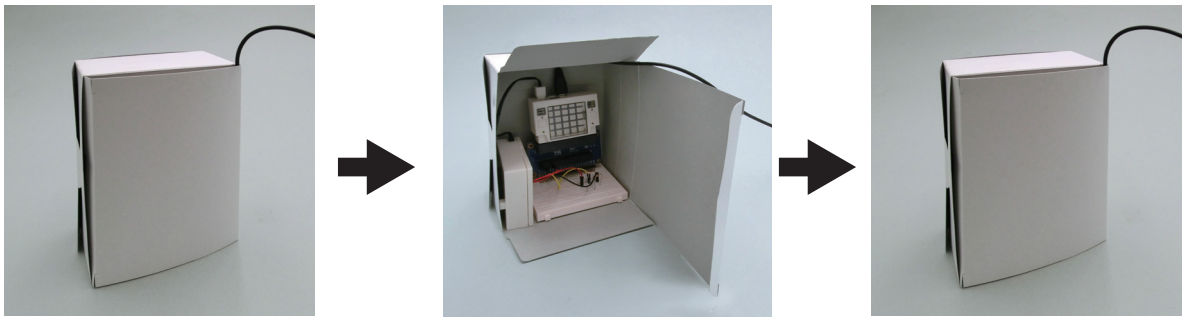


③演習①で作成した hikari.py を Mu エディタの実行ボタンで実行し、REPL の値を確認します。
白箱の蓋（扉）を開け閉めして、閉じているときの光センサの値と、開いているときの光センサの値を確認してください。閉じているときの光センサの値と、開いているときの光センサの値の差を次項の anpi.py にてしきい値として使用します。

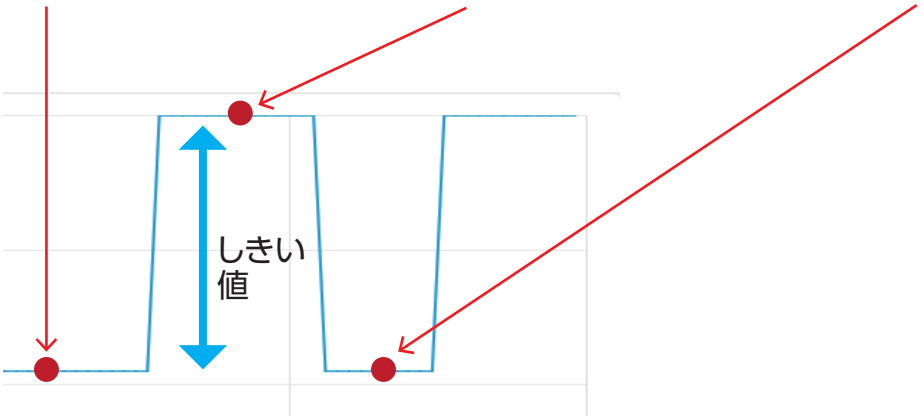
※光センサの数値、しきい値（threshold）は一例

Artec Studuino:Bit MicroPython REPL

```
(50,)
(55,)
(54,)
(54,)
(53,)
(52,)
(560,)
(563,)
(559,)
(564,)
(563,)
```



光センサの値：約50　　光センサの値：約550　　光センサの値：約50



しきい値（threshold）
＝開いているときの光センサの値 ー 閉じているときの光センサの値

→ threshold=500

2. Gmailのアプリパスワードの取得

GmailのSMTPサーバを使ってメールを送信するため、Gmailアカウントのページで「アプリパスワード」を生成します。

①PCのWebブラウザからGoogleアカウントにログインしたあと、Googleトップページの右上の「Googleアカウントの管理」をクリックしてください。

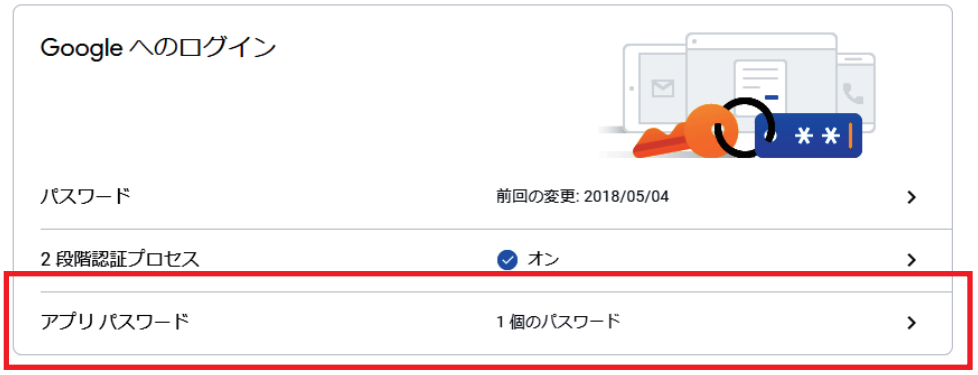


②左サイドバーの「セキュリティ」をクリックして、「Google へのログイン」→「2段階認証プロセス」が「オン」になっている場合は④に進み、「オフ」になっている場合は③に進んでください。
なお、2段階認証プロセスをオン（有効）にするにあたってスマートフォンなどの電話番号が必要になります。



③「2段階認証プロセス」の「オフ」をクリックし、「使ってみる」をクリックします。
2段階認証ではログインする時に確認コードを取得して入力する手順が追加されます。確認コードを取得する方法はいくつかありますが、ここではスマートフォンへのテキストメッセージ（SMS）を使った方法を選択します。使用する端末の電話番号を入力し、「コードの取得方法」として「テキストメッセージ」を選択します。電話番号を入力し、「次へ」をクリックします。
入力した電話番号宛てに SMS で確認コードが送信されるので、それを入力欄に入力して「次へ」をクリックします。
「2段階認証を有効にしますか?」の画面で「有効にする」をクリックします。
「←2段階認証プロセス」の「←」の部分をクリックすると、「セキュリティ」の画面に戻り、「Google へのログイン」の中にある「2段階認証プロセス」が「オン」になることを確認してください。

④左サイドバーの「セキュリティ」をクリックして、「Googleへのログイン」→「アプリパスワード」をクリックしてください。



⑤「アプリパスワード」の画面で、「アプリを選択」をクリックし、「メール」を選択します。
「デバイスを選択」をクリックし 、「その他（名前を入力）」を選択し、「Studuino:bit」を入力して、「生成」をクリックします。



⑥「お使いのデバイスのアプリパスワード」の下に16桁のアプリパスワードが表示されます。
次項のanpi.pyにてこのパスワードを使用しますので、コピーしてメモ帳などのテキストエディタに貼りつける等メモしてください。(※パスワードは1度しか表示されません。)
最後に「完了」をクリックしてください。



⑦「アプリパスワード」の画面で、登録した「名前」と「作成日」が表示されます。
アプリパスワードを使用すると、「最終使用日」の下に日付が表示されます。
作例動作確認後、アプリパスワードを削除したい場合ははゴミ箱のアイコンをクリックしてください。



3. Studuino:bit側のプログラム

① Studuino:bitとPCを接続します。

②下記のanpi.pyをMuエディタで作成し、Studuino:bitに転送します。
thresholdには「1.しきい値の決定」において確認したしきい値を設定します。
smtp_config関数の引数には「2.Gmailのアプリパスワードの取得」において、アプリパスワードの作成の際に用いたGoogleアカウントのGmailアドレス、作成したアプリパスワードを設定します。
Wi-Fiオブジェクトの機能詳細は付録Eを参照してください。

anpi.py

```
from pystubit.board import pin2
from pystubit_iot import wifi_config, smtp_config , wifi_connect, wifi_ifconfig, sendmail
import time

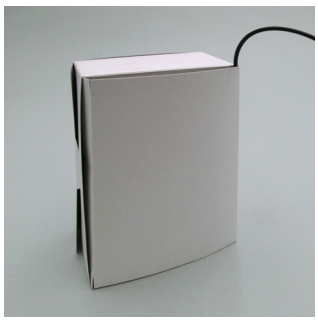
# WiFiに接続する
wifi_config(ssid='アクセスポイントのSSID', pwd='パスワード')
smtp_config(username='ユーザ名', password='パスワード')
wifi_connect(trytime=5)
wifi_ifconfig(('Ⓐ', 'Ⓑ', 'Ⓒ', 'Ⓓ'))  # Studuino:bitに固定のIPアドレスを割り振る

threshold = 500

past = pin2.read_analog()
while True:
    now = pin2.read_analog()
    diff = now - past    # 過去の値(past)と現在の値(now)の差(diff)
    past = now
    if diff > threshold:
        print('メール送信!')
        sendmail(
            '宛先メールアドレス',
            '冷蔵庫安否確認システム',
            '扉が開きました。明るさは{0} です。'.format(now))
    time.sleep_ms(1000)
    print((now,))
```

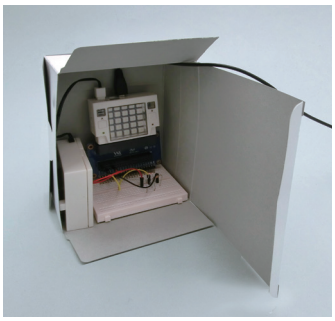
※wifi_ifconfig関数内の各記号には右記の情報を入力してください。(IPアドレスを自動で割り当てることができるDHCPの場合は、この関数は不要です。)

記号	アドレス
Ⓐ	Studuino:bit に割り振る IP アドレス
Ⓑ	サブネットマスク
Ⓒ	デフォルトゲートウェイ
Ⓓ	DNS サーバ



過去の値(past)

差(diff)がしきい値(threshold)より大きい場合、閉まっている扉を開けた瞬間と見なす



現在の値(now)

※しきい値(threshold)は一例として500に設定していますが、この値が小さ過ぎると少しの明るさの変化でメールが送られ、大き過ぎるとメールが送られないことになるため、適切な値に設定してください。閾値を差(開いているときの光センサの値―閉じているときの光センサの値)よりも少し小さな値(差が500の場合は300～400程度)にすると、安定して動作します。

③Muエディタの「実行」ボタンをクリックしてanpi.pyを実行します。
扉を開けたときにMuエディタのREPL に「メール送信!」が表示され、「宛先メールアドレス」にメールが届くことを確認してください。

④以下の方法によりスタンドアローンで実行します。
Muエディタの「転送」ボタンをクリックしてanpi.pyをStuduino:bitに転送してからStuduino:bitをPCから取り外し、電池ボックスまたはUSBケーブルで給電します。Studuino:bitのリセットボタンを押してanpi.pyを実行します。扉を開けたときに「宛先メールアドレス」にメールが届くことを確認します。

●smtp_config関数について

アプリパスワードの作成の際に用いたGoogleアカウントのGmailアドレスをsmtp_config関数の引数Ⓐに設定します。
また、作成した「アプリパスワード」をsmtp_config 関数の引数Ⓑに設定します。
※引数Ⓑには Googleアカウントのパスワードではなく前述の「アプリパスワード」を設定してください。

```
smtp_config(username='Ⓐ', password='Ⓑ')
```

Ⓐ:ユーザ名(Gmailのメールアドレス)
Ⓑ:パスワード

●sendmail関数について

sendmail関数には宛先のメールアドレス、メールのタイトル、メールの本文を文字列で設定します。
なお、今回はメールの本文で文字列のformat( )メソッドを使用しています。
波括弧{ }の置換フィールドがformat( )メソッドの引数で置換されます。
sendmail 関数の引数Ⓐに指定する宛先メールアドレスには、smtp_config 関数の引数Ⓐに指定するメールアドレスとは異なるものを用います。
なお、宛先メールアドレスに Gmail 以外のメールアドレスを指定すると、送られてきたメールの本文やタイトルに文字化けが生じることがあります。このような場合は、宛先メールアドレスにsmtp_config関数の引数Ⓐに指定するメールアドレスとは異なるGmailのメールアドレスを指定すると文字化けが解消します。

```
sendmail('Ⓐ' , 'Ⓑ' , 'Ⓒ')
```

Ⓐ:宛先メールアドレス
Ⓑ:メールのタイトル
Ⓒ:メールの本文

●wifi_config関数について

無線LANアクセスポイントとしてWi-Fiルータを使用する場合は、wifi_config関数の引数にWi-Fiルータの本体もしくは取扱説明書に記載されているSSIDとパスワードをそれぞれ設定してください。無線LANアクセスポイントとしてモバイルホットスポットを使用する場合は、付録Dを参照してください。

```
wifi_config(ssid=' アクセスポイントの SSID', pwd=' パスワード ')
```

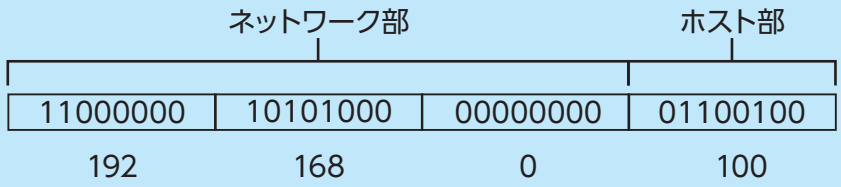
●wifi_ifconfig関数について

(1) IPアドレスのネットワーク部とホスト部

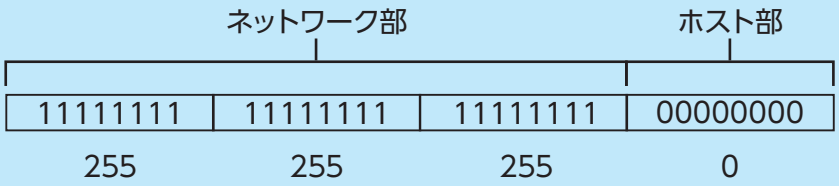
IPアドレスは、上位のネットワーク部と下位のホスト部から成り立っています。ネットワーク部は、そのIPアドレスが属しているセグメント(ネットワークの最小のまとまり)を識別するための部分で、ホスト部は、ネットワーク内のコンピュータを識別するための部分です。同一セグメント内ではネットワーク部はすべて等しくなります。例えば、256個のIPアドレスが利用できるクラスCでは、上位の24ビットがネットワークアドレス、下位の8ビットがホストアドレスになります。

<例>

ネットワーク部が24ビットのIPアドレスが「192.168.0.100」の場合、ネットワーク部は「192.168.0」、ホスト部は「100」になります。



ネットワーク部とホスト部の区別は、サブネットマスクを調べることによってわかります。サブネットマスクとは、ネットワーク部とホスト部を区別するための数値のことで、サブネットマスクの1の部分はIPアドレスのネットワーク部を、サブネットマスクの0の部分はIPアドレスのホスト部をそれぞれ表しています。サブネットマスクが255.255.255.0の場合は、先頭24ビットがネットワーク部で、残り8ビットがホスト部になります。



(2) wifi_ifconfig関数とその引数の確認・設定方法

wifi_ifconfigは、Studuino:bitに固定IPアドレスを設定する関数ですが、引数はタプルで指定します。

```
wifi_ifconfig(('Ⓐ','Ⓑ','Ⓒ','Ⓓ'))
```

記号	アドレス
Ⓐ	Studuino:bit に割り振る IP アドレス
Ⓑ	サブネットマスク
Ⓒ	デフォルトゲートウェイ
Ⓓ	DNS サーバ

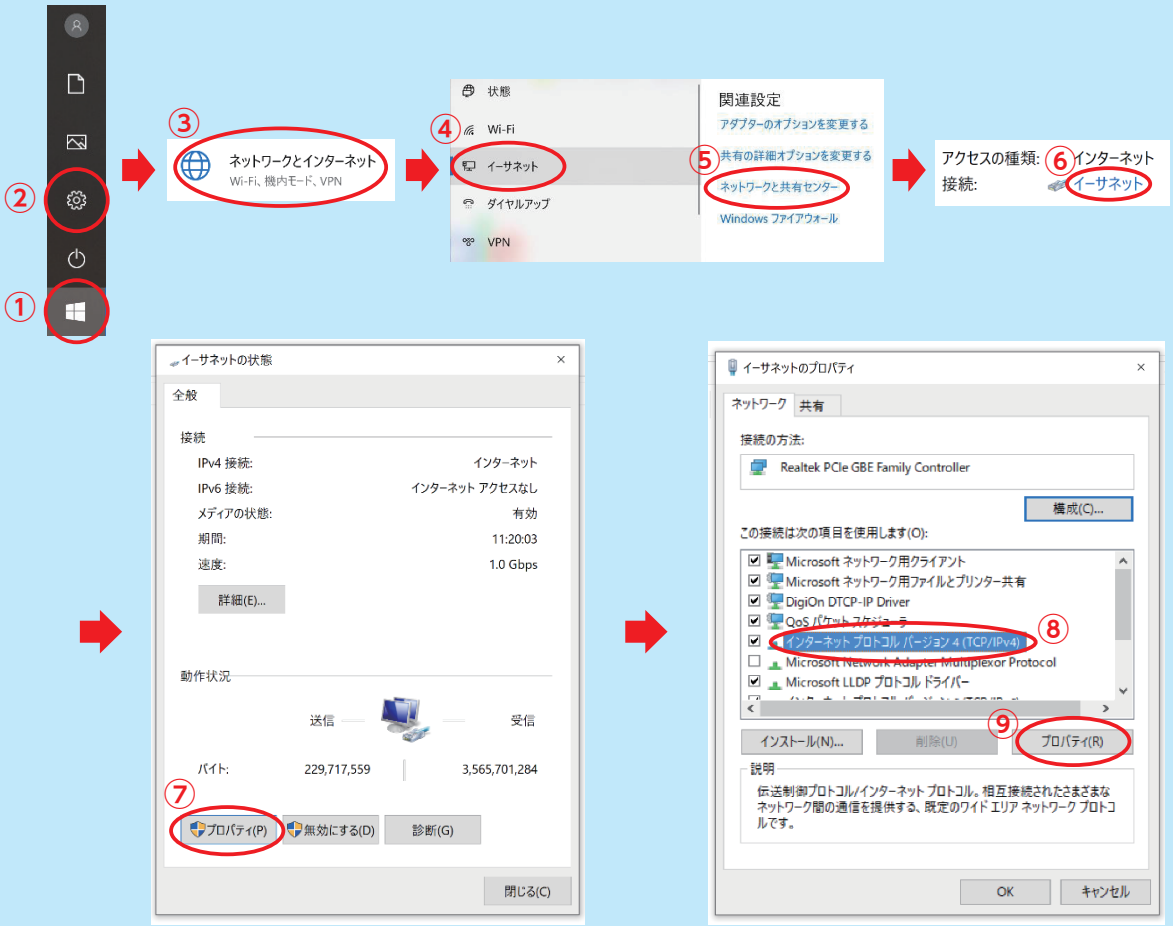
Ⓐ、Ⓑ、Ⓒ、Ⓓの各アドレスは以下のように設定します。

[コンピュータ教室の各アドレスの確認方法]

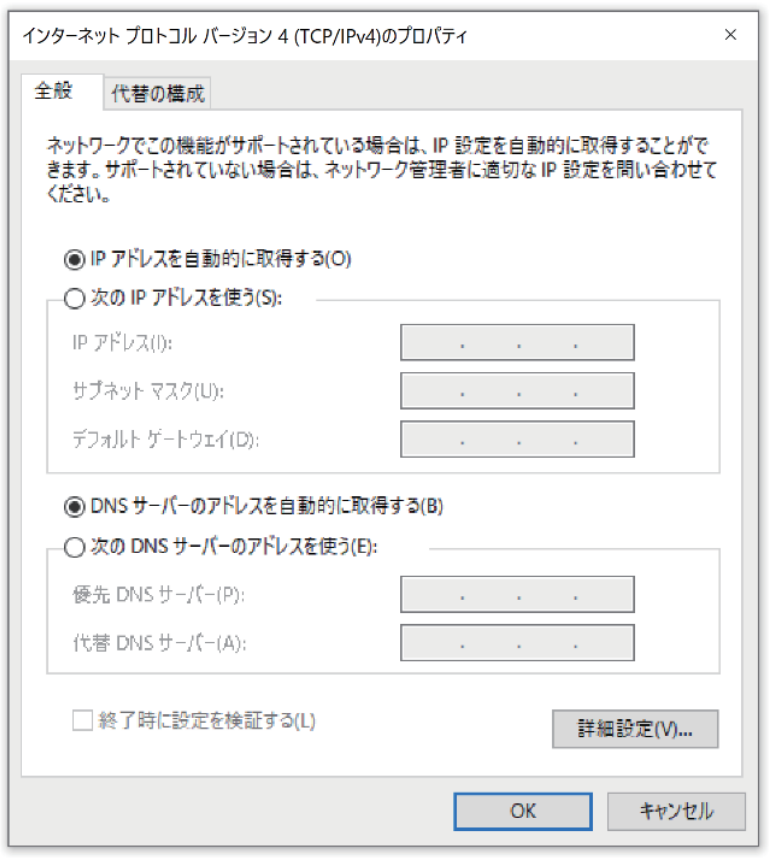
Windows 環境

コンピュータ教室のLAN内にある任意のPCを起動します。

①スタートボタン → ②設定 → ③ネットワークとインターネット → ④イーサネット →⑤ネットワークと共有センター → ⑥イーサネット → ⑦プロパティ → ⑧インターネットプロトコルバージョン4(TCP/IPv4) → ⑨プロパティの順にクリックします。



※インターネットプロトコルバージョン4(TCP/IPv4)のプロパティの画面が以下のように表示された場合は、LAN内の装置にはIPアドレスが自動で割り当てられています（自動で割り当てる仕組みをDHCPと呼びます）。このような場合は、プログラムの中のwifi_ifconfig関数がある部分を削除してください。

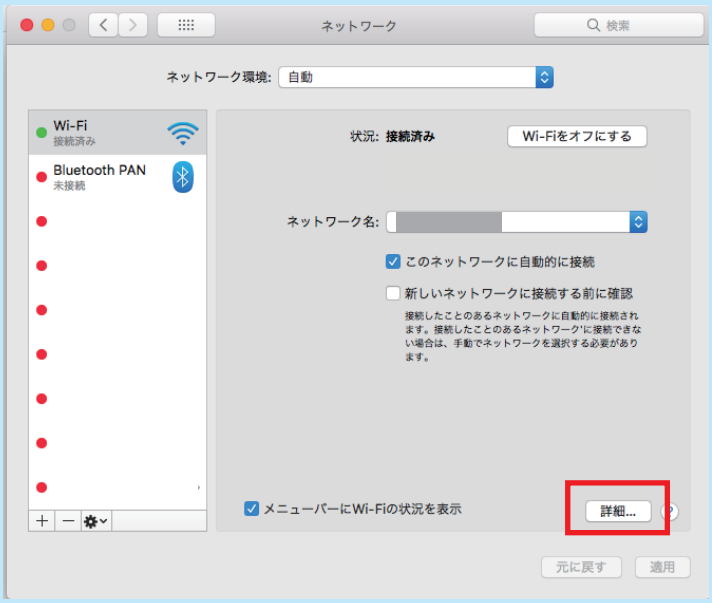


Mac 環境

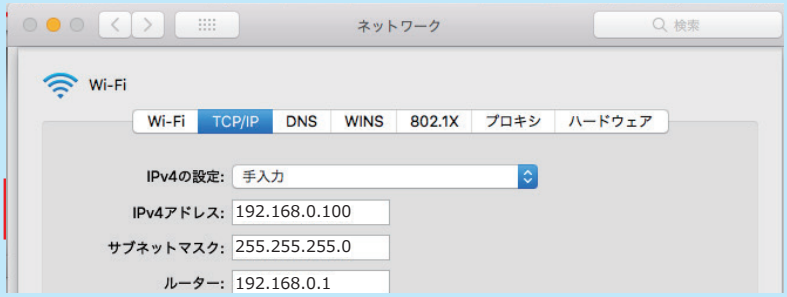
「Appleメニュー」（画面左上リンゴのマーク）-「システム環境設定」-「ネットワーク」をクリックします。



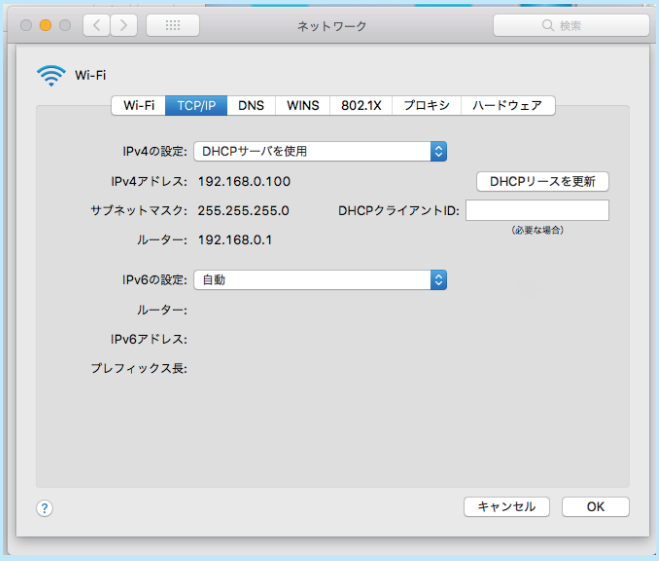
左側の項目から、コンピュータ教室のLANに接続している「Wi-Fi」(無線LAN)または、「Ethernet」(有線LAN)を選択し、「詳細」ボタンをクリックします



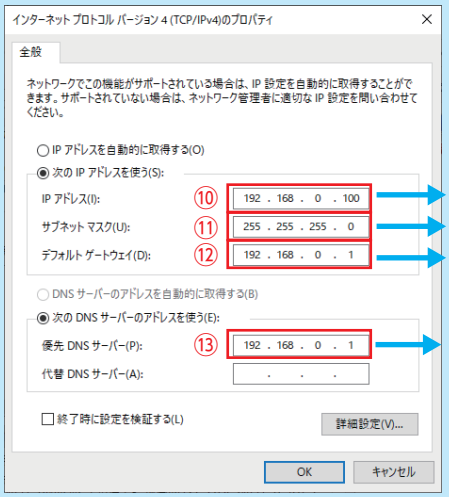
「TCP/IP」タブをクリックします。



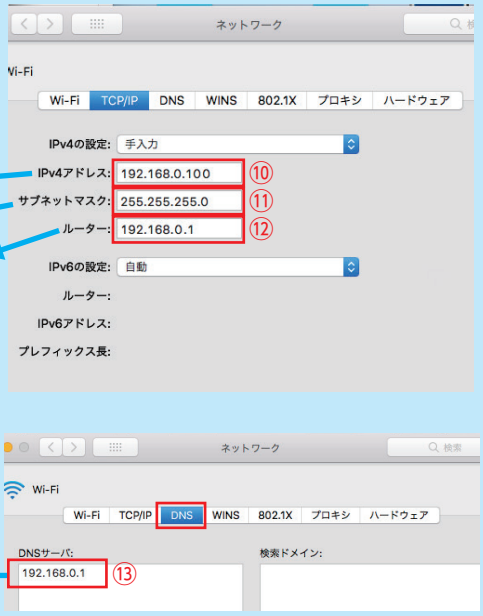
※以下のような画面が表示された場合は、LAN内の装置にはIPアドレスが自動で割り当てられています(自動で割り当てる仕組みをDHCPと呼びます)。このような場合は、プログラムの中のwifi_ifconfig関数がある部分を削除してください。



Windows 環境



Mac 環境



Ⓐ：下記参照

Ⓑ：サブネットマスク

Ⓒ：デフォルトゲートウェイ

Ⓓ：DNS サーバ

※⑫と⑬が同じ値になっていることがあります。
これはルータの DNS サーバの機能を使用しているためです。

Ⓐのアドレスの指定例：192.168.0.＊＊＊

同一セグメント内で割り当てられている IP アドレスのネットワーク部を指定

コンピュータ教室内の他の装置に割り当てられているホスト部と重ならない値を指定

「IPアドレス」の⑩の部分に表示されるアドレス（この例では192.168.0.100）は、あくまで現在使用しているパソコンのIPアドレスです。（※この例ではサブネットマスクが255.255.255.0であることから、IPアドレスのネットワーク部は192.168.0で、ホスト部は100です。）

p.20の(2)のⒶのStuduino:bitに割り振るIPアドレスは、同一セグメント内で重ならない値を割り当てる必要があります。

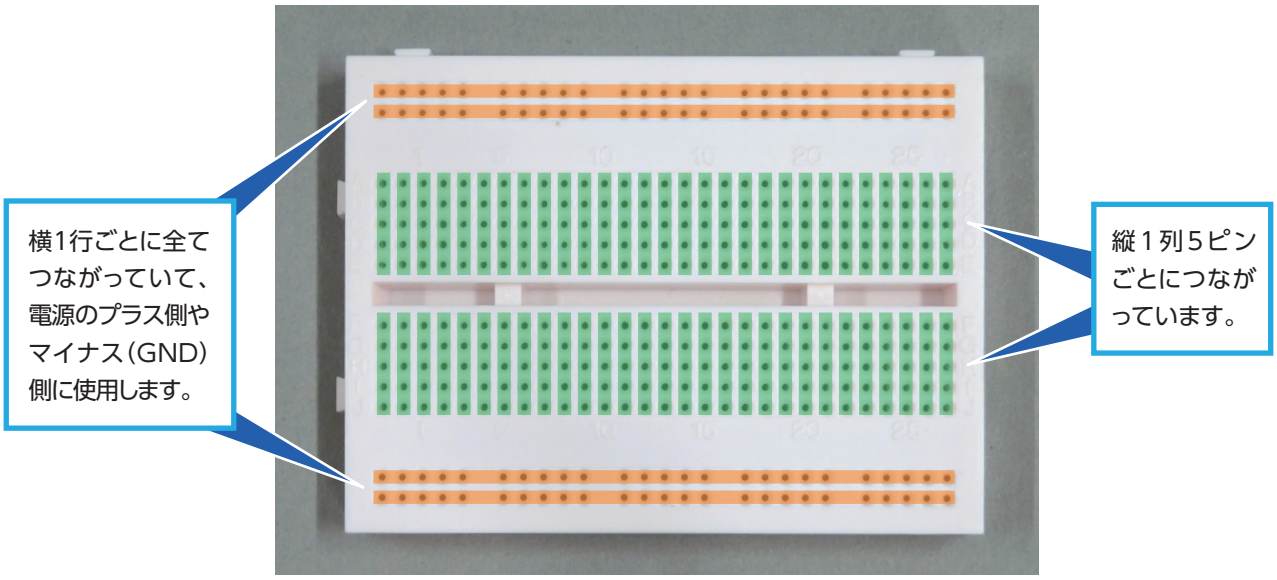
ⒶのIPアドレスのネットワーク部には、コンピュータ教室内の各装置に割り当てられているIPアドレスのネットワーク部と同じ値を使用します。

ⒶのIPアドレスのホスト部には、コンピュータ教室内の他の装置に割り当てられているホスト部と重ならない値を使用します。

事前に同じLAN（コンピュータ教室）内にある他のパソコンやプリンタ、イメージスキャナーなどの装置に割り当てられているIPアドレスを調査してください。（LANを構築した時の設定書などを調査します）。

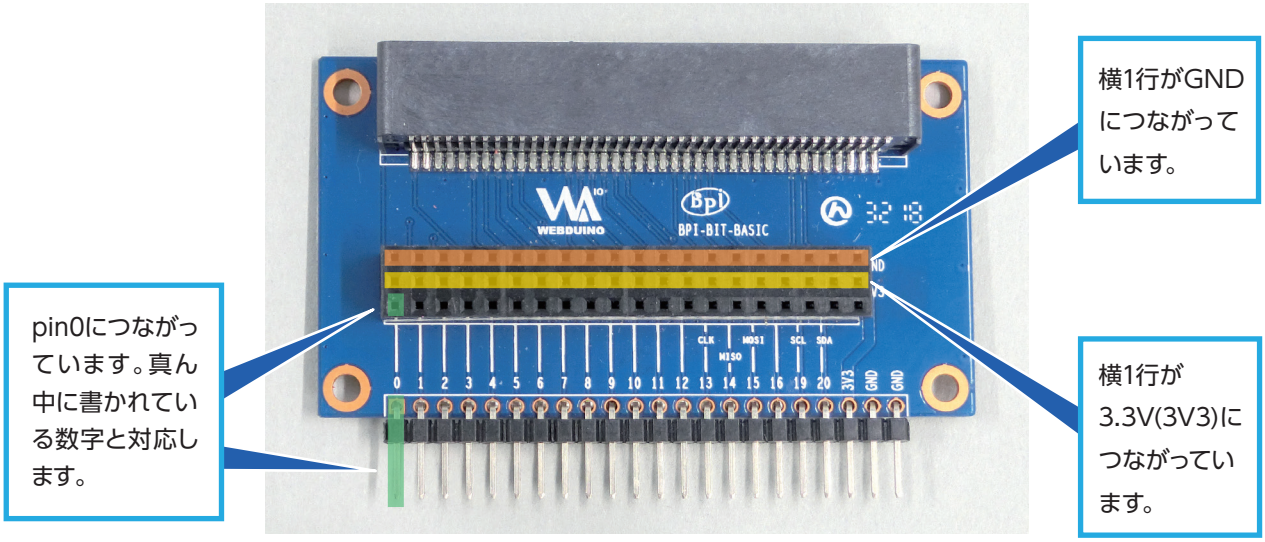
付録 A. ブレッドボードの使い方

ブレッドボードは内部がつながっており、部品を差し込むだけで電子回路を組み立てることができます。



付録 B. 変換エッジコネクタの使い方

Studuino:bit下部のエッジコネクタ主要端子全てにアクセスすることを可能にする、変換基板です。
ジャンパワイヤを介して、回路の実装などが簡単に行えます。



注意事項

エッジコネクタは金属がむき出しになっている部分があります。
そこに金属製のものが接触してショートしないように注意してください。

付録 C. デジタル入出力・アナログ入力オブジェクトの機能一覧

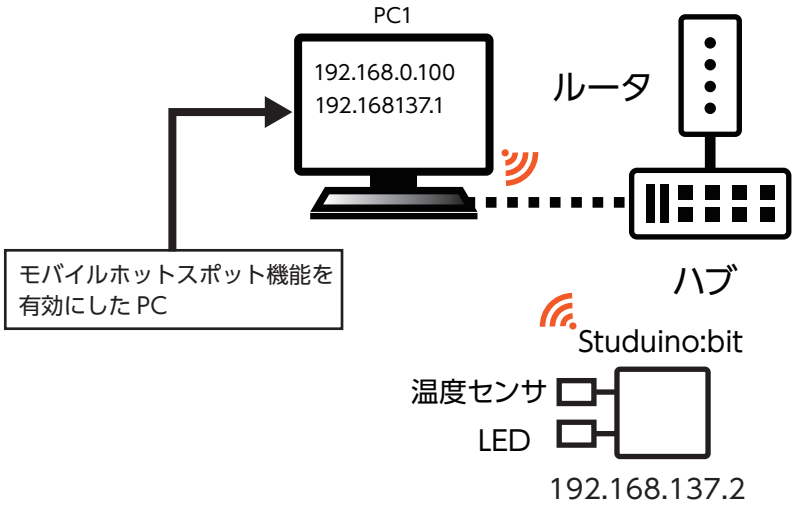
端子モジュールはpin0/pin1/pin2/pin3です。pin2/pin3はデジタル入出力に対応していないため、pin2.write_digital(value)やpin3.write_digital(value)を実行するとエラーになります。
デジタル入力用のread_digitalメソッドは使用できます。

メソッド	説明
write_digital(value)	value引数が1の場合はHighに、0の場合はLowにデジタル信号を設定します。
read_digital()	デジタル信号を取得します。High の場合は 1 を、Low の場合は 0 を返します。
write_analog(value)	PWM信号を端子に出力します、value引数は0(0%)～1023(100%)です。
status()	PWM の使用状況を表示します。
read_analog()	端子の電圧を読み取り、0(0V)から1023(3.3V)までの間の整数値を返します。

付録 D.　モバイルホットスポットの使い方

モバイルホットスポットとはインターネットに接続されたWindows10やWindows11のPCを無線LANアクセスポイントに変換する機能です。
図中のルータやハブ、PC1は、コンピュータ教室などに設置されている既存の装置を使用します。PC1にモバイルホットスポット機能がない場合は、モバイルホットスポット機能があるPCをハブに接続し、そのPCにStuduino:bitを無線LAN接続させることもできます。
なお、モバイルホットスポット機能を有効にしたPC（図ではPC1）には、有線LAN側のIPアドレス（図では192.168.0.100）とは別に、モバイルホットスポット側に192.168.137.1がゲートウェイとして割り当てられるため、Studuino:bitに割り当てるIPアドレスのネットワーク部は、192.168.137にする必要があります。図では、Studuino:bitのIPアドレスとして192.168.137.2を割り当てています。

※IP アドレスの数値は一例



①「スタートボタン」→歯車の形の「設定」をクリックするか、キーボードの「Windows」キー+「Ｉ」キーを押して、設定画面を表示します。



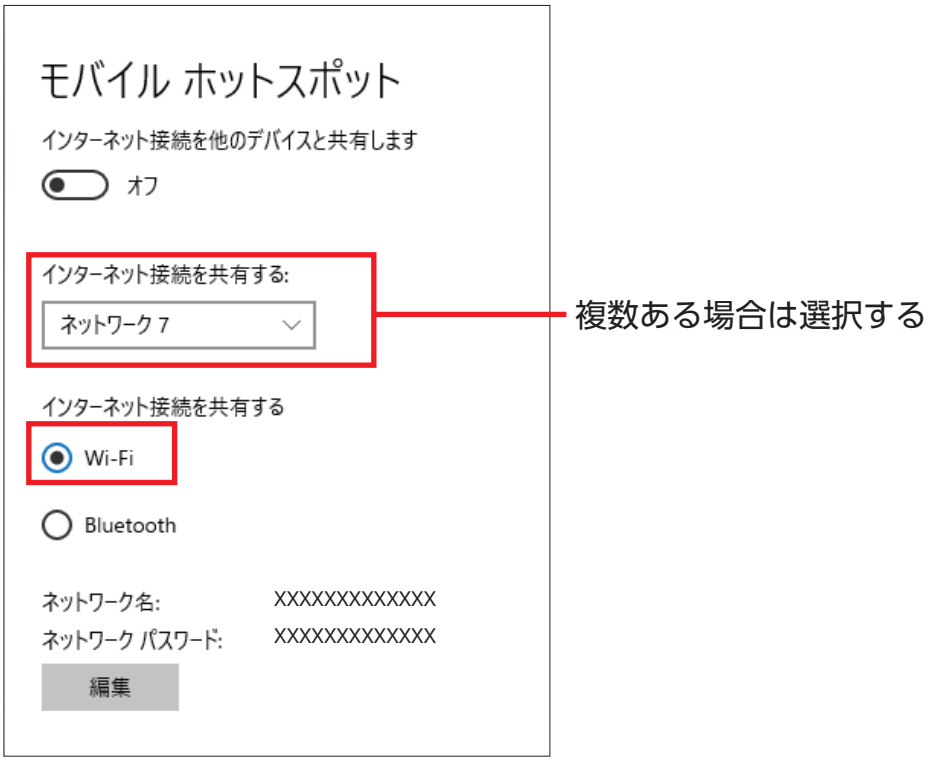
②「ネットワークとインターネット」をクリックします。



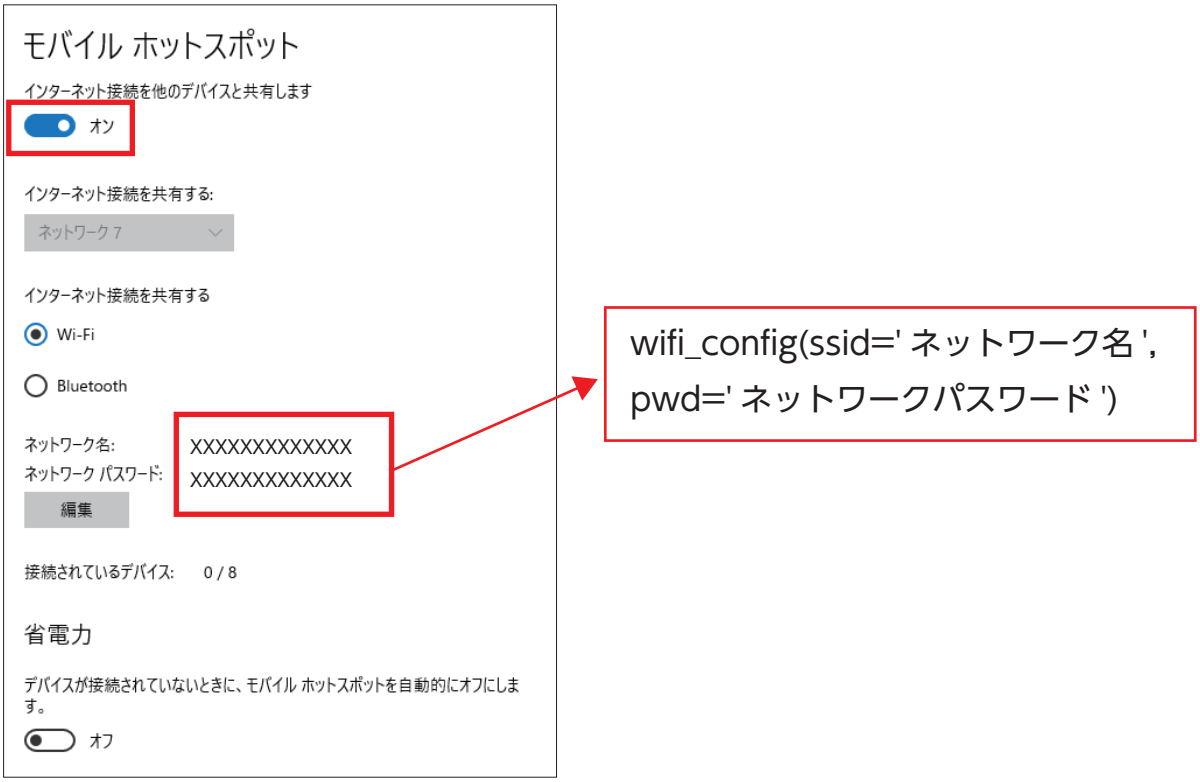
③「モバイルホットスポット」をクリックします。



④複数のインターネット接続がある場合は、共有するインターネット接続を選択して、「Wi-Fi」を選択します。



⑤モバイルホットスポットを「オン」にします。ネットワーク名とネットワークパスワードの部分に表示される値を、Studuino:bit の wifi_config に設定して、モバイルホットスポットに接続します。



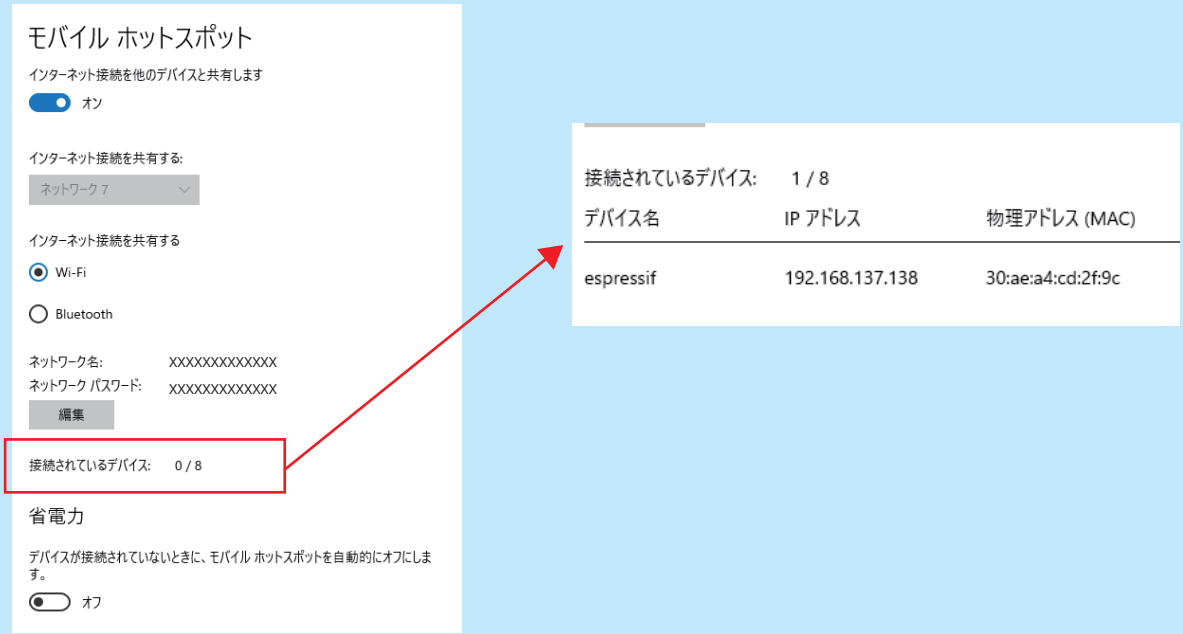
●モバイルホットスポットの設定画面に表示される IP アドレスについて

モバイルホットスポットの設定画面には接続されているデバイスの「デバイス名」「IPアドレス」「物理アドレス（MAC）」が表示されます。
Studuino:bit側でwifi_ifconfig関数を使用して「192.168.137.2」を割り当てている場合も、モバイルホットスポットの設定画面には別のIPアドレス（下記の画像の例では「192.168.137.138」）が表示されてしまいます。
これは初めにStuduino:bitがモバイルホットスポットに接続したときに「192.168.137.138」がDHCP機能によって動的に割り当てられ、その後指定した固定IPアドレスの「192.168.137.2」が割り当て直されることによって発生します。

```
from pystubit_iot import wifi_ifconfig, wifi_config, wifi_connect
wifi_config(ssid=' ネットワーク名 ',pwd=' ネットワークパスワード ')
wifi_ifconfig(('192.168.137.2', '255.255.255.0', '192.168.137.1', '192.168.137.1'))
wifi_connect()
```

最終的にはこの設定が割り当てられている

Studuino:bit 側のプログラム



wifi_config 関数を使わない DHCP 機能による動的 IP アドレスを割り当てた場合は、モバイルホットスポットの設定画面に表示された IP アドレスが、Studuino:bit の IP アドレスです。

付録 E.　Wi-Fi 機能の一覧

メソッド	説明
wifi_config(ssid='Wi-Fi ルータの SSID', pwd=' Wi-Fi ルータのパスワード ')	Studuino:bit を接続する Wi-Fi の設定を引数に設定します。
wifi_ifconfig(('Ⓐ','Ⓑ','Ⓒ','Ⓓ'))	Studuino:bit に固定の IP アドレスを設定します。 Ⓐ Studuino:bit に割り振る IP アドレス Ⓑ サブネットマスク Ⓒ デフォルトゲートウェイ Ⓓ DNS サーバ
wifi_connect(Ⓐ)	Studuino:bit を Wi-Fi 接続します。 Ⓐ: trytime= 再接続回数
wifi_disconnect()	Studuino:bit の Wi-Fi 接続を解除します。