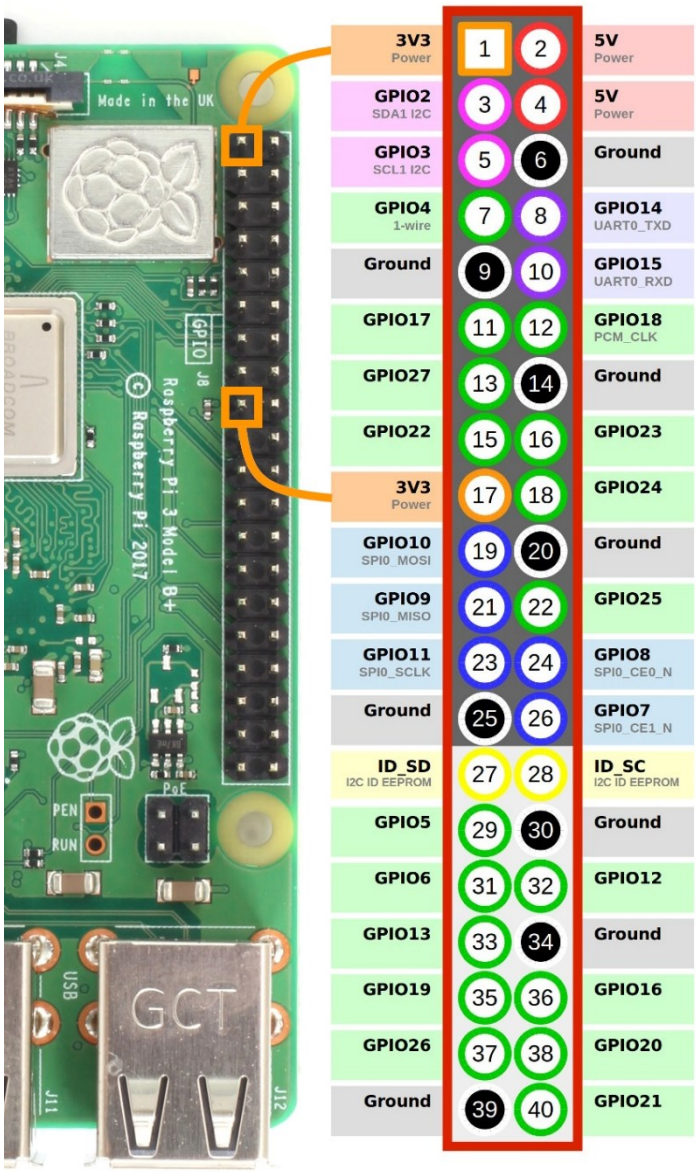
Bài thí nghiệm

GIAO TIẾP GPIO VÀ XÂY DỰNG WEB SERVER TRÊN RASPBERRY PI

1. Phần 1: Giao tiếp GPIO trên Raspberry Pi



Mô tả chân (pinout) trên Raspberry Pi 3 Model B

- Tìm hiểu các phương pháp giao tiếp gpio trên Raspberry Pi:
https://elinux.org/RPi_GPIO_Code_Samples

1.1. Giao tiếp GPIO sử dụng thư viện python RPi.GPIO

Thư viện này được cài đặt mặc định cùng với Raspbian OS

Bước 1: Khai báo sử dụng thư viện Python RPi.GPIO

import RPi.GPIO as GPIO
print(GPIO.RPI_INFOR)

Bước 2: Xác định kiểu số hiệu chân (pin) muốn sử dụng:

- GPIO.BOARD – Sử dụng số hiệu chân như header của bo mạch
  GPIO.setmode(GPIO.BOARD)
- GPIO.BCM – Sử dụng số hiệu chân theo đặc tả của Broadcom.
  GPIO.setmode(GPIO.BCM)

Bước 3: Cấu hình chế độ IN/OUT cho pin muốn sử dụng

- setup([pin], [GPIO.IN, GPIO.OUT])
- Example: GPIO.setup(18, GPIO.OUT)

Bước 4: Đọc/Ghi dữ liệu (Read Inputs, Give Outputs)

Output:

- **Digital Output:**
  - Sử dụng hàm: GPIO.output([pin], [GPIO.LOW, GPIO.HIGH])
  - Ví dụ xuất 1 (HIGH) ra chân 18:
    GPIO.output(18, GPIO.HIGH)
  - GPIO.HIGH = 1 = True
  - GPIO.LOW = 0 = False
- **PWM** ("Analog") Output**:** (Pulse Width Modulation)
  - Khởi tạo PWM sử dụng hàm
    GPIO.PWM([pin], [frequency])
  - Bắt đầu xuất ra xung bằng hàm:
    pwm.start([duty cycle])
  - Ví dụ:
    pwm = GPIO.PWM(18, 1000)
    pwm.start(50)
  - Thay đổi độ rộng xung (PWM duty cycle) bằng hàm:
    pwm.ChangeDutyCycle([duty cycle]). (0-100%)
    Dừng xuất xung bằng hàm pwm.stop()

Input:

- Đọc giá trị trên chân input dùng hàm:
  GPIO.input([pin])
- Giá trị trả về: TRUE/ FALSE (HIGH/LOW)

Bước 5: Giải phóng chân GPIO sau khi sử dụng

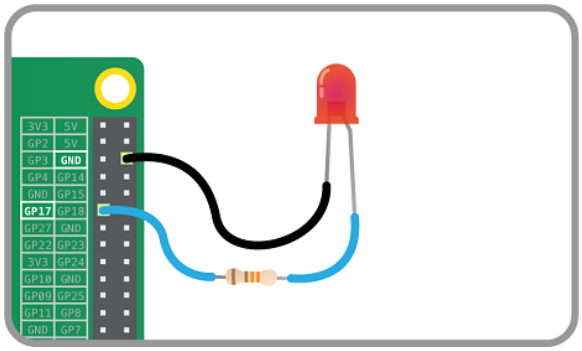
- Giải phóng chân để tránh xung đột khi sử dụng lại chân đó trong chương trình khác
  GPIO.cleanup()
- Hoặc tắt cảnh báo bằng hàm:
  GPIO.setwarnings(False)

Sử dụng Delay trong thư viện time của python

```
import time
time.sleep([seconds])
time.sleep(0.25)
```

Ví dụ: LED Blinky

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
led = 17
GPIO.setup(led, GPIO.OUT)
while True:
      GPIO.output(led, True)
      sleep(1)
      GPIO.output(led, False)
      sleep(1)
```



1.2.Giao tiếp GPIO sử dụng thư viện C wiringPi

Bước 1. Thêm thư viện Wiringpi : #include<wiringPi.h>
Bước 2. Thiết lập chọn kiểu đánh số chân GPIO

wiringPiSetupGpio();

Wiringpi có 4 kiểu chọn đánh số chân.
- wiringPiSetup() : thiết lập đánh số theo các riêng của Pi
- wiringPiSetupGpio() : đánh số theo Broadcom GPIO
- wiringPiSetupPhys() : đánh số theo chân header trên board.
- wiringPiSetupSys() : đánh số theo system class GPIO.

Bước 3. Chọn và thiết lập chế độ Output hoặc Input

pinMode(pin, OUTPUT);
//pinMode(pin, INPUT);

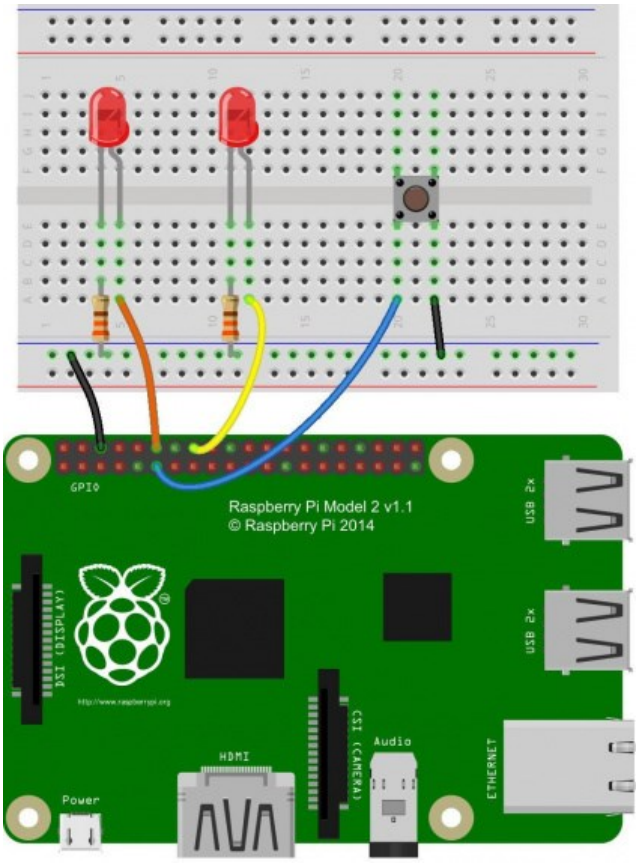
Bước 4. Đọc/ghi dữ liệu
Bật-tắt LED (2 kiểu)
digitalWrite(17, 1); //digitalWrite(17, HIGH);
digitalWrite(17, 0); //digitalWrite(17, LOW);

Bước 5. Thực hiện build chương trình trên terminal
gcc -Wall -o blink blink.c -lwiringPi
sudo ./blink

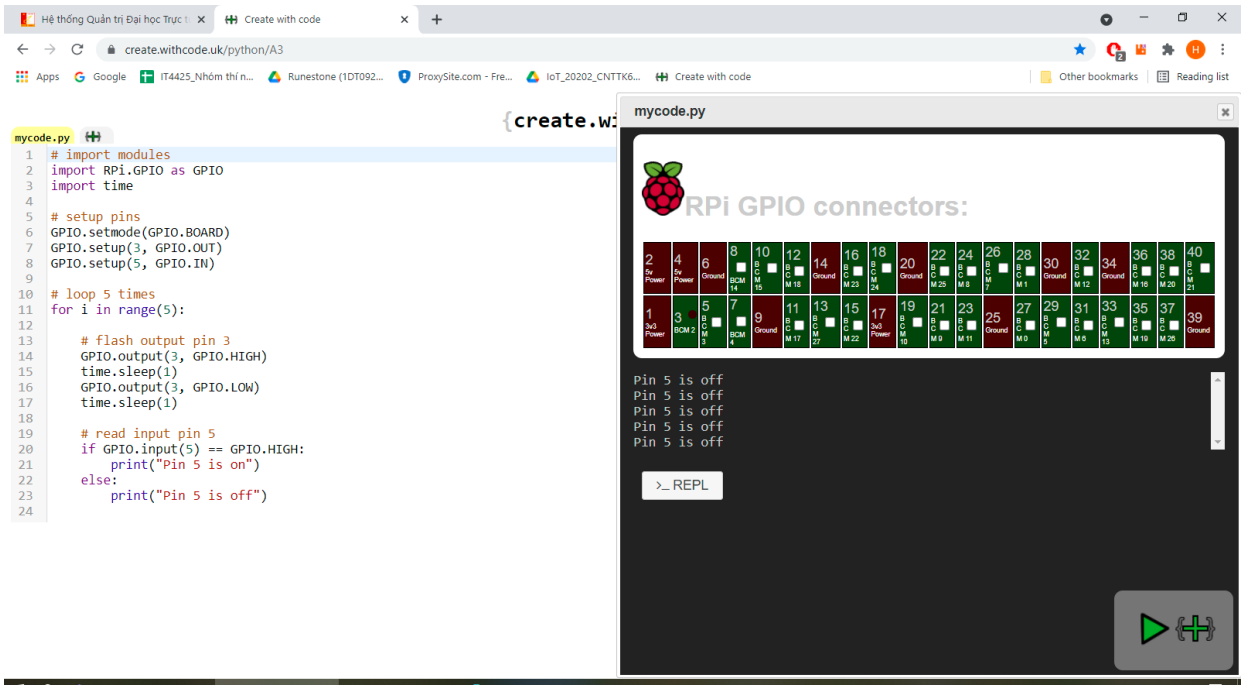
Bài tập 1. Điều khiển LEDs, đọc trạng thái button

Viết chương trình nhấp nháy LED trên GPIO18, đọc trạng thái nút bấm và điều khiển tắt/bật LED trên GPIO23

- **2 LEDs** kết nối đến chân **GPIO 18 và GPIO 23** (Broadcom chip-specific numbers = BCM pin number)
- Số hiệu chân trên header P1 (BOARD pin number):
  - GPIO 18 = Pin 12
  - GPIO 23 = Pin 16
- 1 **button** kết nối đến chân Broadcom **GPIO 17**, tương ứng pin 11 trên heard P1 của Board.

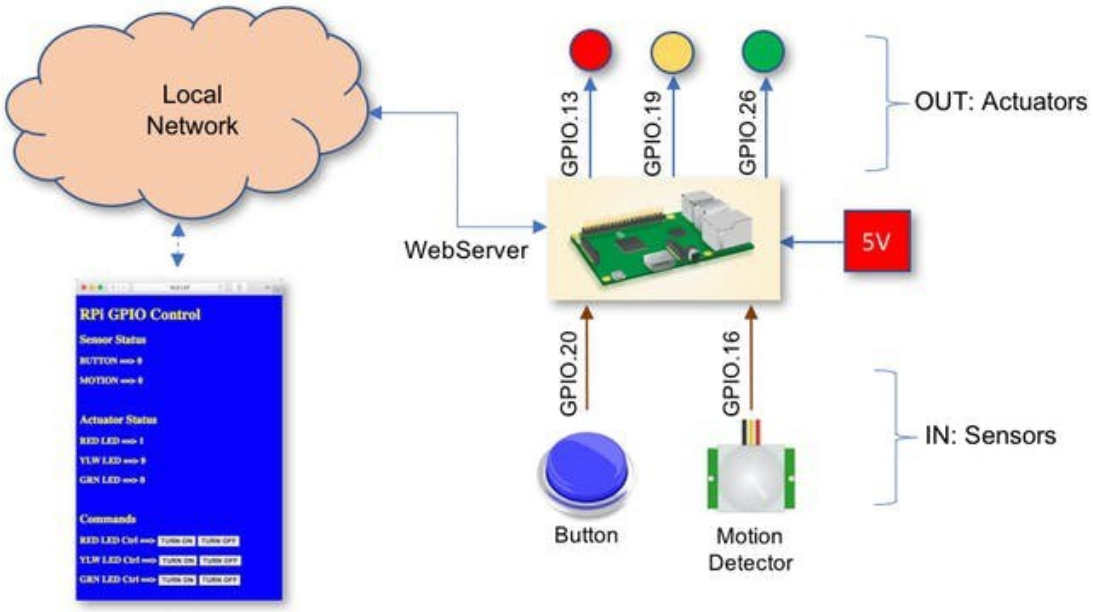


Viết chương trình Python và chạy chương trình trên công cụ mô phỏng: https://create.withcode.uk/python/A3



Phần 2. Xây dựng web server trên Raspberry Pi sử dụng Flask

Có thể loại bỏ phần GPIO để thực hiện web server chạy trên máy tính dùng Python Flask



- Nội dung chính:
  - Xây dựng web server sử dụng Flask chạy trên Pi
  - Định tuyến (routes) các trang web
  - Sử dụng html templates và CSS
  - Điều khiển GPIO qua webserver

Bước 1. Cài đặt, tạo thư mục lưu trữ web server.

- Truy cập vào Raspberry Pi qua ssh
- Cài đặt Flask trên Pi:
  sudo apt-get install python3-flask
- Tạo thư mục chứa các files của local web server (nằm trong thư mục /home/pi/Documents)
  mkdir rpiWebServer
- Tạo 2 thư mục con: *static* chứa các files css, Javascript và *templates* chứa các files HTML

```
/rpiWebServer
        /static
        /templates
```

Để cài đặt thư viện flask cho python3 trên Windows, mở cửa sổ terminal, dùng lệnh:

python3 -m pip install flask

Nếu sử dụng python2 => Sử dụng lệnh pip install flask

Bước 2. Viết ứng dụng web server

- Viết ứng dụng web app bằng python (ví dụ app.py) đặt trong thư mục rpiWebServer đã tạo

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index(): return 'Hello world'
if __name__ == '__main__':
        app.run(debug=True, host='0.0.0.0')
```

- Ứng dụng web server đơn giản (app.py):
  - import thư viện Flask, tạo đối tượng

```
from flask import Flask
app = Flask(__name__)
```

  - Hàm route() để định tuyến, khi truy cập một địa chỉ URL sẽ kích hoạt hàm tương ứng

```
@app.route('/')
def index(): return 'Hello world'
```

  - NOTE: Note here the host='0.0.0.0' means the web app will be accessible to any device on the network.

Bước 3. Chạy web server (thay thế bằng chạy trên máy tính)

- Chạy web server: **python3 app.py**
- Output:

```
* Running on
* Restarting with reloader
```

- Mở trình duyệt và truy nhập địa chỉ http://127.0.0.1:5000/. Kết quả:

Bước 4. Route (định tuyến) cho các trang

- @app.route('/'): xác định trang bắt đầu (entry point); (the / means the root of the website, so just http://127.0.0.1:5000/).

```
@app.route('/')
def index(): return 'Hello world'
```

- def index(): hàm được gọi khi truy cập URL tương ứng.
- return 'Hello world': nội dung trang web trả về cho trình duyệt.
- Định tuyến cho trang /cakes sẽ gọi hàm xử lý def cakes() tương ứng

```
@app.route('/cakes')
def cakes(): return 'Yummy cakes!'
```

- Ví dụ tạo file index.html

```
<html>
<body>
<h1>Hello from a template!</h1>
</body>
</html>
```

- Sử dụng template trong ứng dụng web server app.py:

```
from flask import Flask, render_template
@app.route('/')
def index(): return render_template('index.html')
```

Bước 6. Sử dụng CSS cho trang web

- Tạo file style.css, đặt trong thư mục **static** của web app
- Ví dụ style.css:

```
body {
        background: red;
        color: yellow;
}
```

- Sử dụng trong file html

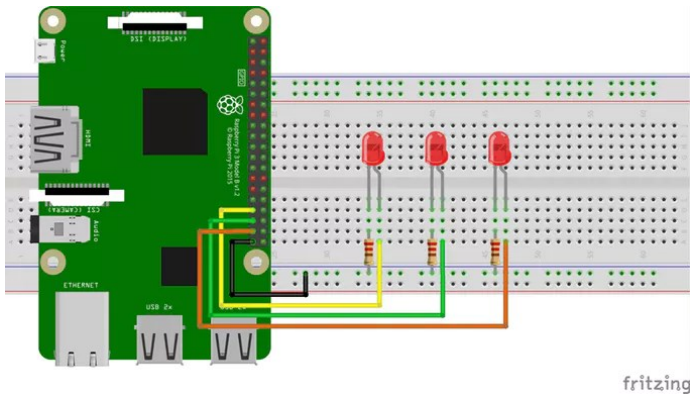
```
<html>
<head>
<link rel="stylesheet" href='../static/style.css' />
</head>
<body>
<h1>Hello from a template!</h1>
</body>
</html>
```

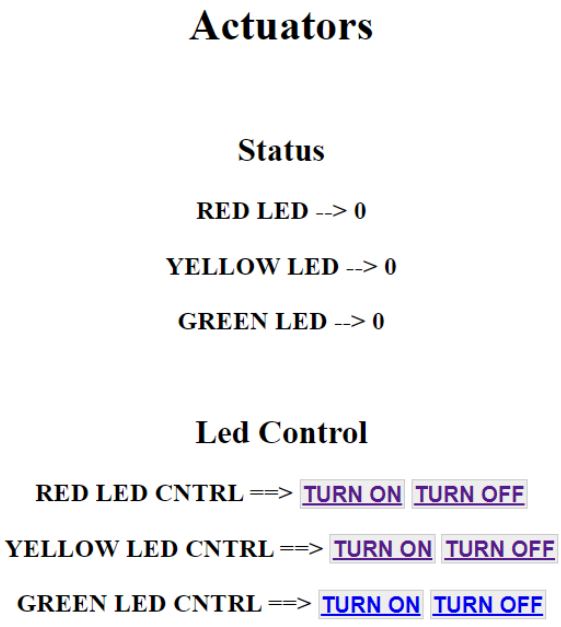
Bước 7. Truy cập web server từ thiết bị khác (trong cùng mạng LAN)

- Ví dụ URL = http://192.168.1.3:5000/

Bài 2. Xây dựng web server như sau

- Minh họa điều khiển 3 chân GPIO qua web server
- Xây dựng giao diện html đơn giản với 6 buttons (turn on/off) trên trang web





Ghi chú: Thực hiện phần giao diện html, css, python trên máy tính

Hướng dẫn:

Bước 1. Xây dựng html template

```
<html>
<head> <title> GPIO Control Web App</title>
<link rel="styleSheet" href="/static/style.css"/>
</head>
<body>
<h1>Actuators</h1><br>
<h2>Status</h2>
<h3> RED LED: {{ledRed}}</h3>
<h3> YELLOW LED: {{ledYellow}}</h3>
<h3> GREEN LED: {{ledGreen}}</h3><br>
<h2>Led Control</h2>
<h3> RED LED CNTRL ==> <a href ="/ledRed/on" class="button">TURN
ON</a>
<a href="/ledRed/off" class = "button">TURN OFF</a></h3>
<h3> YELLOW LED CNTRL ==> <a href="/ledYellow/on"
class="button">TURN ON</a>
<a href="/ledYellow/off" class="button">TURN OFF</a></h3>
<h3> GREEN LED CNTRL ==> <a href="/ledGreen/on"
class="button">TURN ON</a>
<a href="/ledGreen/off" class="button">TURN OFF</a></h3>
</body>
</html>
```

Bước 2. Xây dựng css

```
body {
        text-align: center;
        background: #FFFFFF;
        color: #000000;
}
.button{
        font: bold 16px Arial;
        background-color:#EEEEEE;
        padding: 1px;
        border: 1px solid #CCCCCC;
}
```

Bước 3. Xây dựng phần server app bằng python

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
ledRed = 13
```

```
ledYellow= 19
ledGreen= 26
ledRedSts = 0
ledYellowSts = 0
ledGreenSts = 0
GPIO.setup(ledRed, GPIO.OUT)
GPIO.setup(ledYellow,GPIO.OUT)
GPIO.setup(ledGreen, GPIO.OUT)
GPIO.output(ledRed, GPIO.LOW)
GPIO.output(ledYellow, GPIO.LOW)
GPIO.output(ledGreen, GPIO.LOW)
@app.route('/')
def index():
   ledRedSts = GPIO.input(ledRed)
   ledYellowSts = GPIO.input(ledYellow)
   ledGreenSts = GPIO.input(ledGreen)
   templateData = { 'ledRed' : ledRedSts, 'ledYellow' :
ledYellowSts, 'ledGreen' : ledGreenSts }
   return render_template('index.html', **templateData)
@app.route('/<deviceName>/<action>')
def do(deviceName, action):
   if deviceName == "ledRed":
      actuator = ledRed
   if deviceName == "ledYellow":
      actuator = ledYellow
   if deviceName == "ledGreen":
      actuator = ledGreen
   if action == "on":
      GPIO.output(actuator, GPIO.HIGH)
   if action == "off":
      GPIO.output(actuator, GPIO.LOW)

   ledRedSts = GPIO.input(ledRed)
   ledYellowSts = GPIO.input(ledYellow)
   ledGreenSts = GPIO.input(ledGreen)
   templateData = { 'ledRed' : ledRedSts, 'ledYellow' : ledYellowSts,
'ledGreen' : ledGreenSts }
   return render_template('index.html', **templateData )
```

Ghi chú: *Phần web server chạy trên Pi sử dụng thư viện RPi.GPIO, có thể loại bỏ phần giao tiếp GPIO để cho web server chạy được trên máy tính.*