الوحدة الثالثة: المستشعرات والمشغلات

الربط بالحاسوب

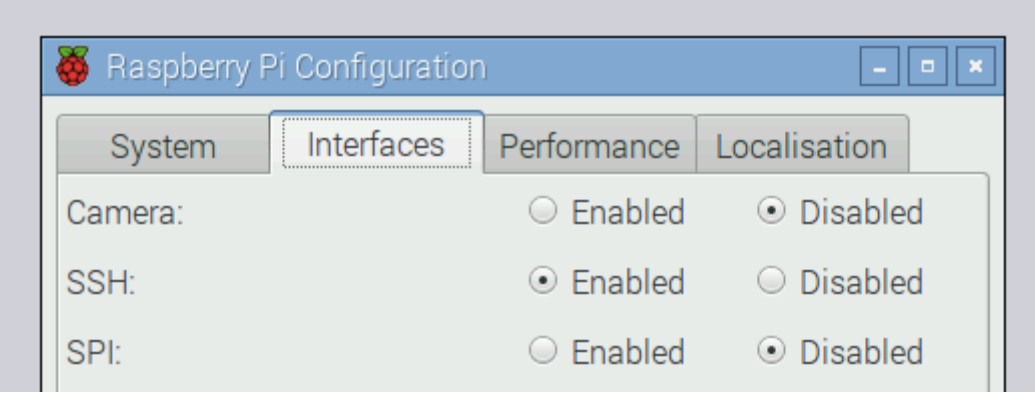
م. غنام الجعبري

وحدات الادخال والاخراج

- المستشعرات (Sensors) هي وحدات ادخال تستخدم في رصد الأحداث والتغييرات في البيئة المحيطة مثل:
  - الضوء (light)
  - الحرارة (temperature)
  - الحركة (motion)
  - المسافة (distance)
- المشغلات (Actuators) هي وحدات اخراج تستخدم في التحكم بالمحركات الكهربائية غالبا

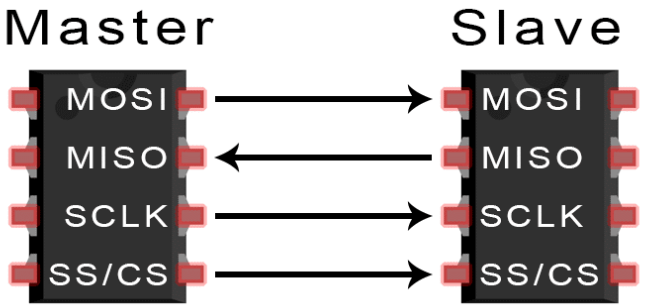
وحدة ADC

- كافة ابر الادخال والاخراج (GPIO) على لوحة راسبيري باي هي قنوات رقمية لان اللوحة لا تحتوي على وحدة تحويل الاشارات التماثلية الى رقمية (ADC)
- يتم وصل وحدة ADC خارجية بلوحة راسبيري باي لقراءة القيم التماثلية مثل MCP3008 التي توفر 8 قنوات تماثلية وبدقة 10 خانات ثنائية (bit)
- يتم الاتصال بين لوحة راسبيري باي ووحدة MCP3008 عن طريق بروتوكول SPI وينبغي تفعيل هذا البروتوكول على اللوحة لقراءة القيم التماثلية على قنوات ADC



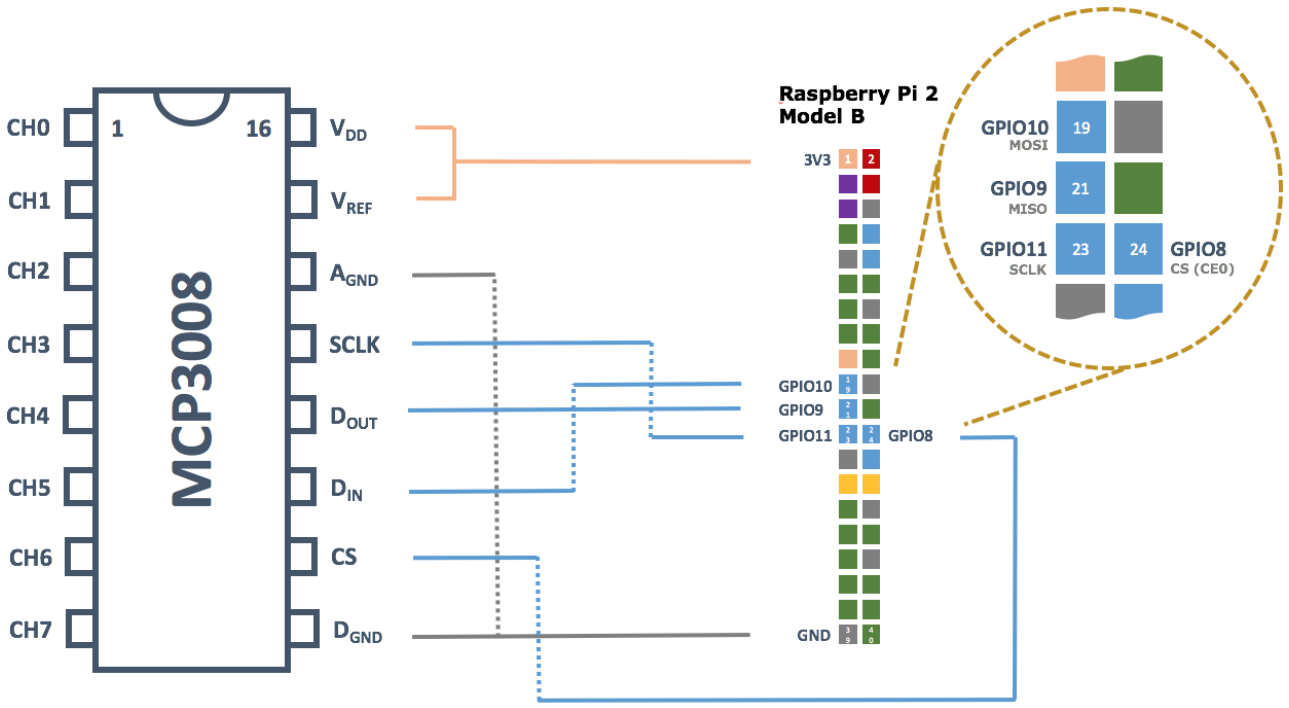
بروتوكول SPI

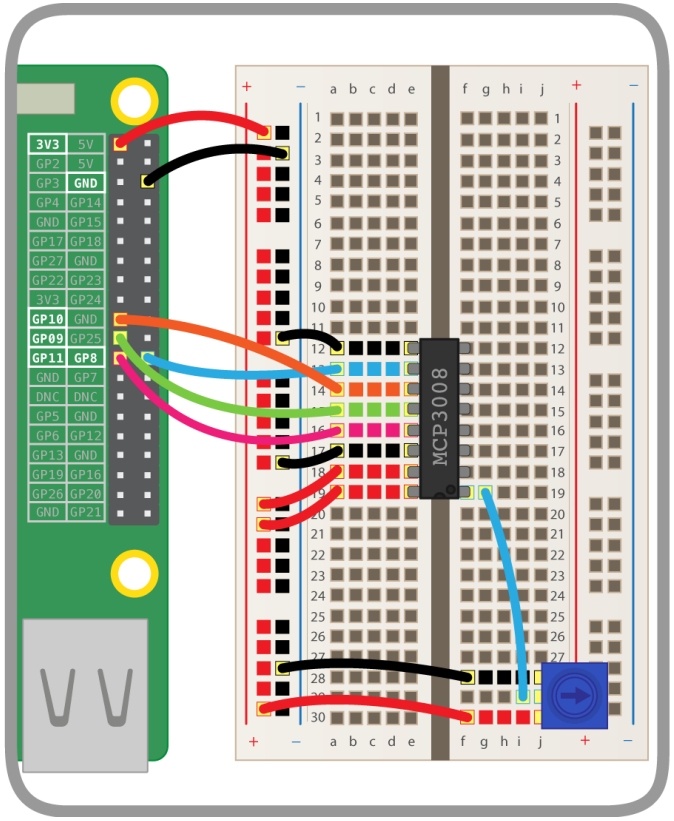
- هو بروتوكول اتصال تسلسلي متزامن يستخدم في ربط عدة وحدات طرفية مع جهاز واحد
- الوحدة الطرفية تمثل الجهاز التابع (Slave) ولوحة راسبيري باي التي تتحكم بالوحدات الطرفية تمثل الجهاز الرئيس (Master)
- يتم الاتصال بين الجهاز التابع والجهاز الرئيس بالاتجاهين في نفس الوقت باستخدام 4 اسلاك (4-wire)
- يوفر اتصالا سريعا بين الاجهزة وملائم للاتصالات قصيرة المسافة في الانظمة المدمجة



مثال: مقاومة متغيرة

- دائرة الكترونية لقراءة قيمة تماثلية باستخدام ADC ومقاومة متغيرة (Potentiometer)

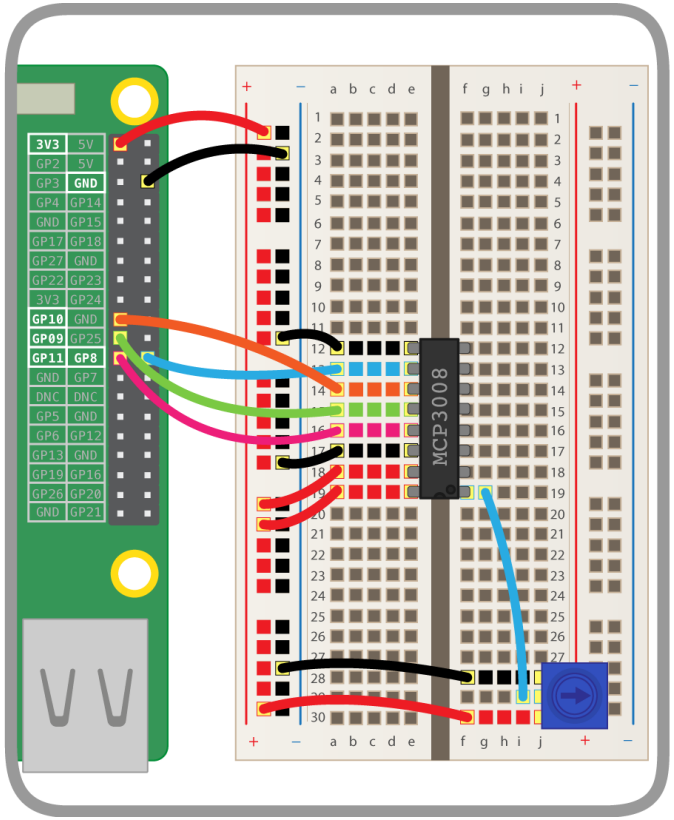




مثال: مقاومة متغيرة

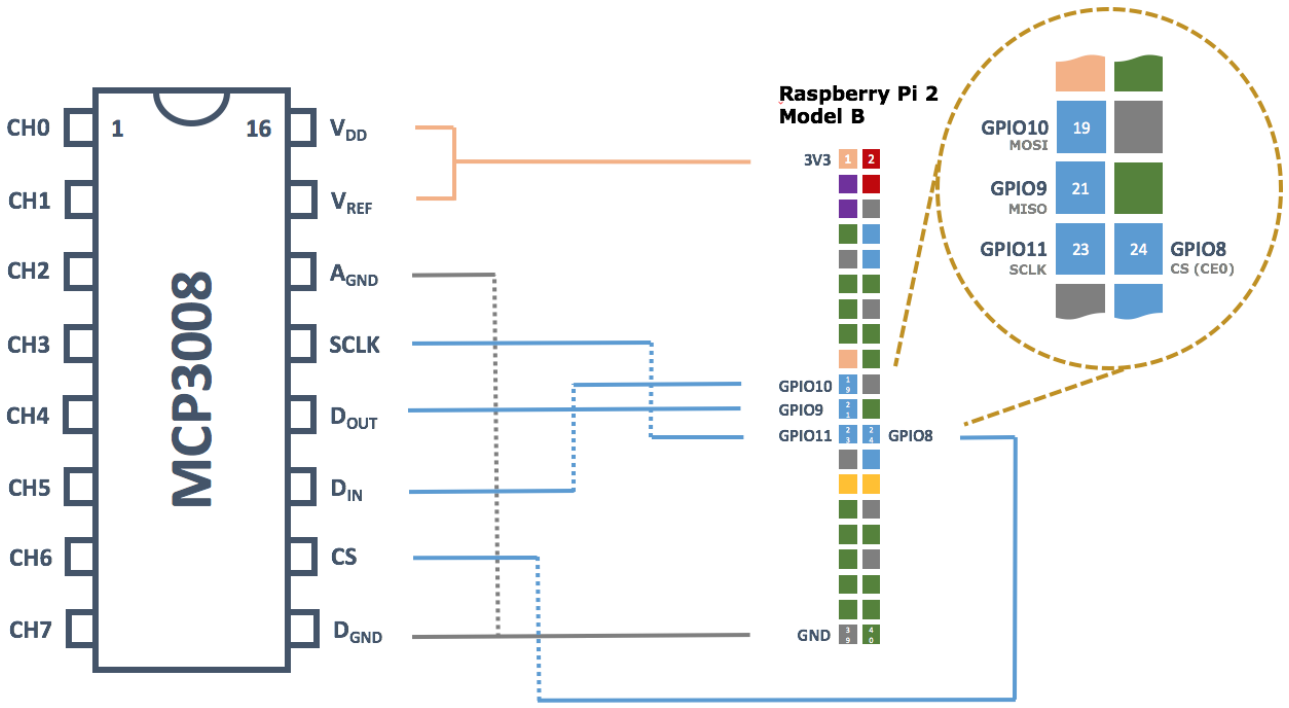
- برنامج بلغة بايثون لقراءة قيمة تماثلية باستخدام ADC ومقاومة متغيرة

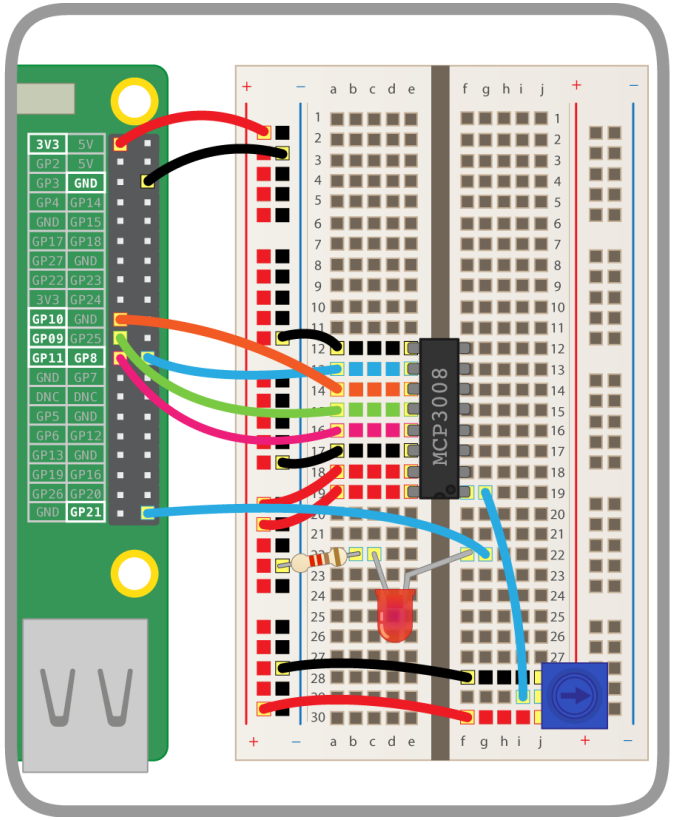
```
import spidev
import time

def read_adc(channel = 0):
    spi = spidev.SpiDev()
    spi.open(0,0)
    adc_data = spi.xfer2([1, (8 + channel) << 4, 0])
    spi.close()
    result = ((adc_data[1] & 3) << 8) + adc_data[2]
    return result

while True:
        value = read_adc(0)
        print(value)
        time.sleep(0.5)
GPIO.cleanup()
```



تمرين: مفتاح تحكم

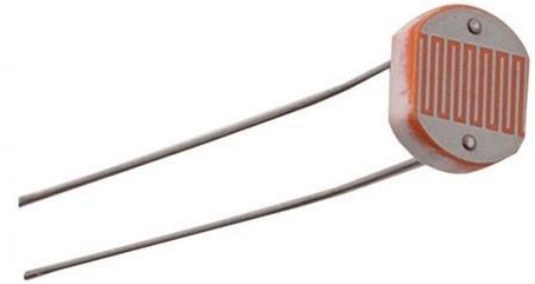
- اكتب برنامج بلغة بايثون للتحكم بشدة الضوء باستخدام مقاومة متغيرة





مستشعر الضوء

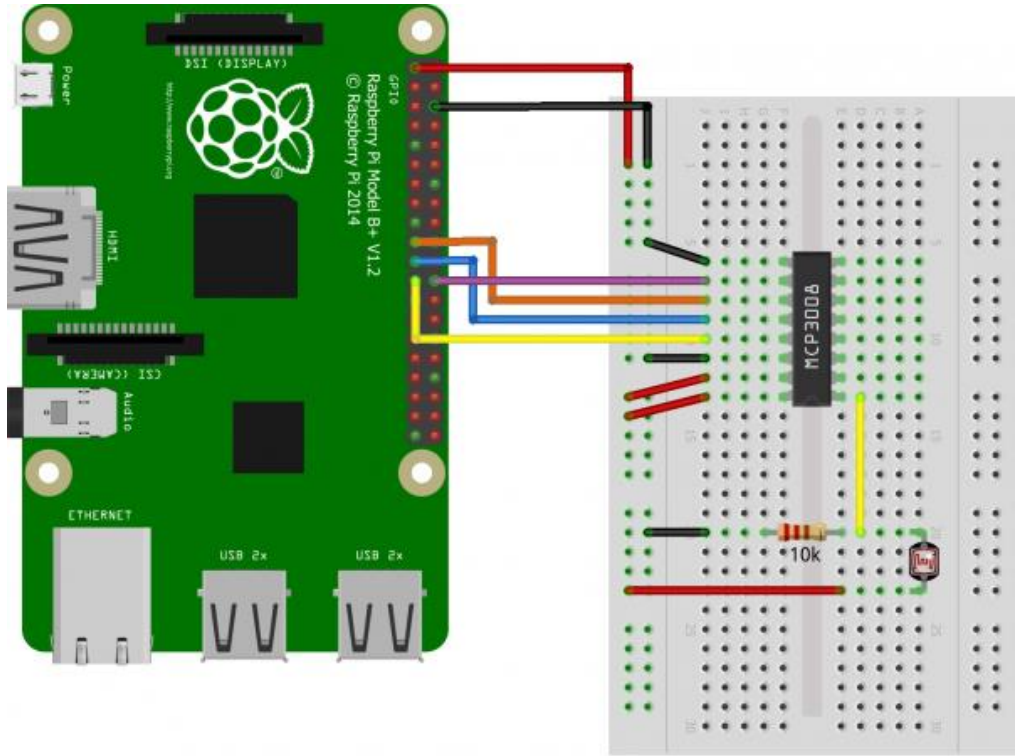
- مستشعر الضوء (Photoresister) او (LDR) هو مقاومة كهربائية حساسة للضوء تستخدم في قياس مستوى الاضاءة حيث تقل قيمة المقاومة عندما تزداد شدة الضوء



مثال: مستشعر الضوء

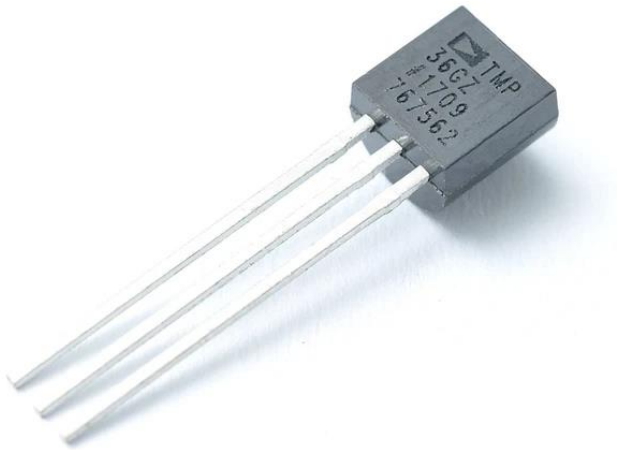
- برنامج بلغة بايثون لقراءة شدة الاضاءة باستخدام ADC ومستشعر الضوء LDR

```
import spidev
import time

def read_adc(channel = 0):
    spi = spidev.SpiDev()
    spi.open(0,0)
    adc_data = spi.xfer2([1, (8 + channel) << 4, 0])
    spi.close()
    result = ((adc_data[1] & 3) << 8) + adc_data[2]
    return result

while True:
    value = read_adc(0)
    print(value)
    time.sleep(0.5)
GPIO.cleanup()
```



مستشعر الحرارة

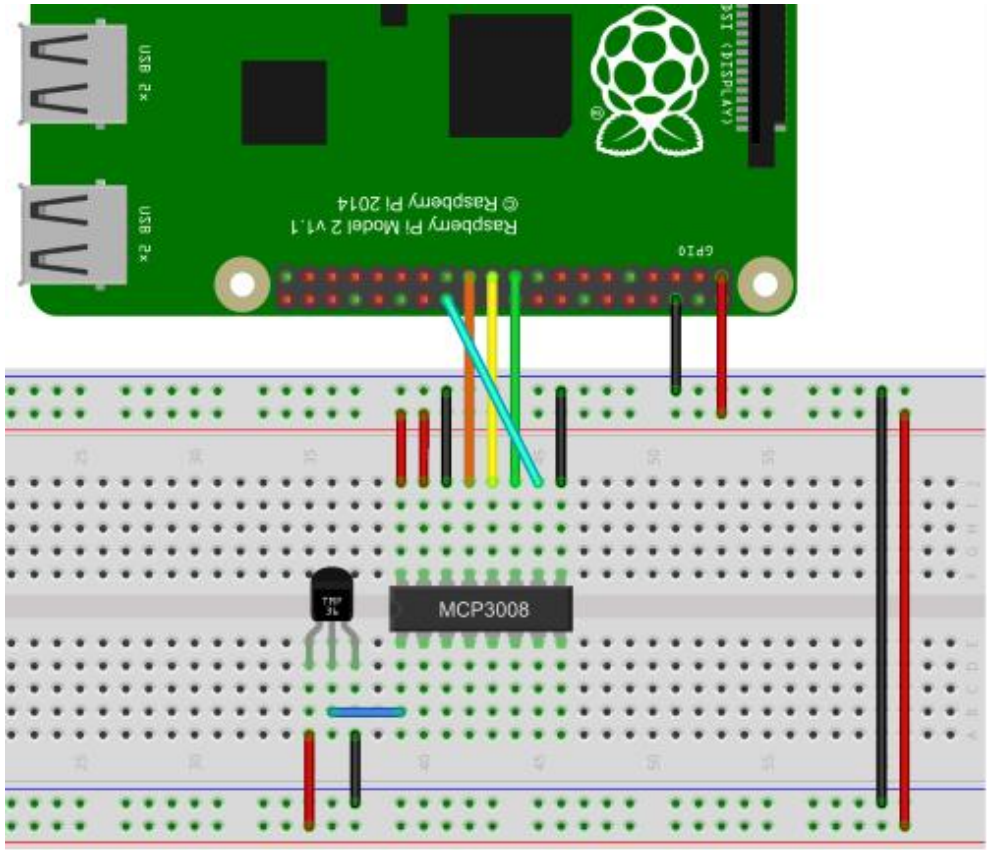
- مستشعر الحرارة (TMP36) يعتمد على الثنائيات (diodes) في قياس درجة الحرارة بمعدل ثابت أي وفق معادلة خطية، على عكس المقاومة الحساسة للحرارة (thermistor)
- يستخدم مستشعر TMP36 في قياس درجة الحرارة من 50°C- الى 150°C



مثال: مستشعر الحرارة

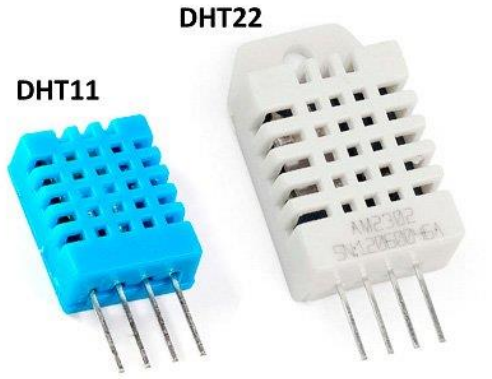
- برنامج بلغة بايثون لقراءة درجة الحرارة باستخدام ADC ومستشعر الحرارة TMP36

```
import spidev
import time

def read_adc(channel = 0):
   spi = spidev.SpiDev()
   spi.open(0,0)
   adc_data = spi.xfer2([1, (8 + channel) << 4, 0])
   spi.close()
   result = ((adc_data[1] & 3) << 8) + adc_data[2]
   return result
def convert_temp(value):
   volt = (value / 1023) * 5
   temp = (volt - 0.5) * 100
   return round(temp, 1)
while True:
        value = read_adc(0)
        temp = convert_temp(value)
        print(temp)
        time.sleep(0.5)
GPIO.cleanup()
```



مستشعر الحرارة والرطوبة

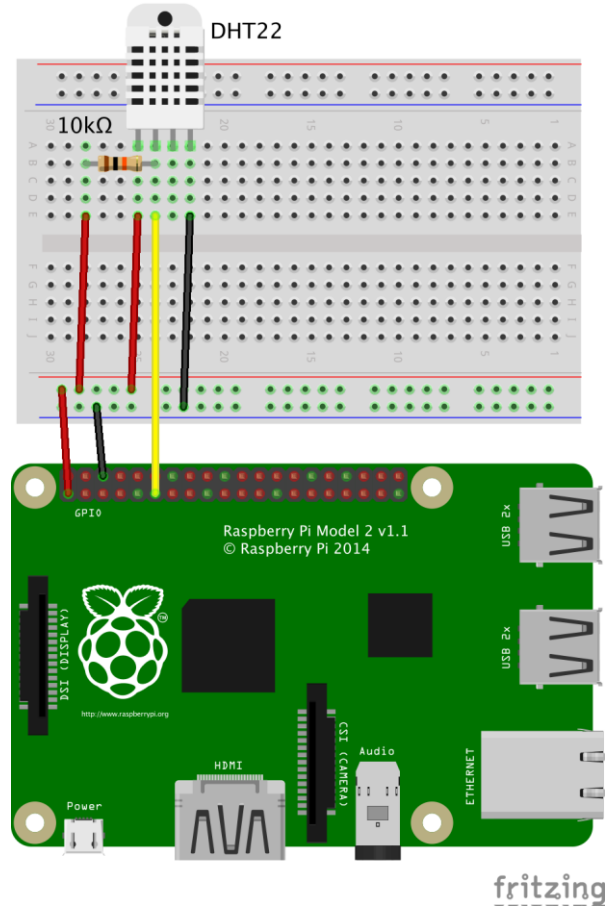
- يستخدم مستشعر DHT في قياس درجة الحرارة والرطوبة النسبية (humidity)
- هنالك نوعان من مستشعر الحرارة والرطوبة النسبية:
  - مستشعر DHT11 لقياس درجة الحرارة من 0-50°C والرطوبة النسبية من 20-80%
  - مستشعر DHT22 لقياس درجة الحرارة من 40-80°C- والرطوبة النسبية من 0-100%



مثال: مستشعر الحرارة والرطوبة

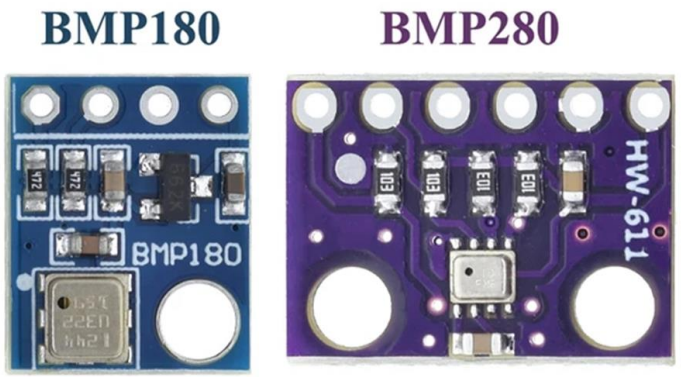
- برنامج بلغة بايثون لقراءة درجة الحرارة والرطوبة النسبية باستخدام مستشعر DHT22

```
import Adafruit_DHT
import time

sensor = Adafruit_DHT.DHT22
pin = 17

while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

    print(f'Temperature: {temperature:.2f}°C')
    print(f'Humidity: {humidity:.2f}%')

    time.sleep(2)
```



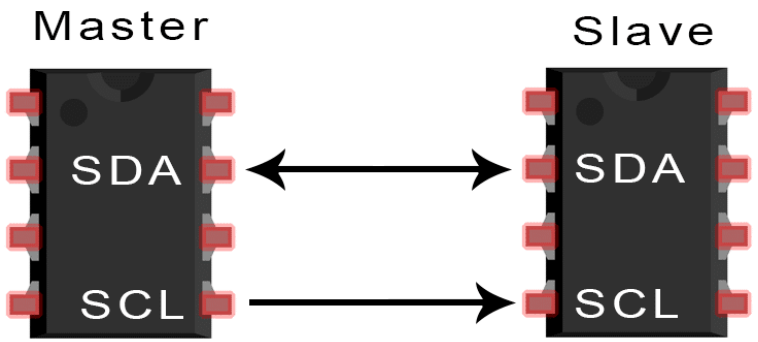
مستشعر الحرارة والضغط الجوي

- يستخدم مستشعر BMP في قياس درجة الحرارة والضغط الجوي (pressure)
- هنالك نوعان من مستشعر الحرارة والضغط الجوي:
  - مستشعر BMP180 يدعم بروتوكول I2C
  - مستشعر BMP280 يدعم بروتوكول I2C وبروتوكول SPI
- ينبغي تفعيل بروتوكول I2C على لوحة راسبيري باي للاتصال مع مستشعر BMP180



بروتوكول I2C

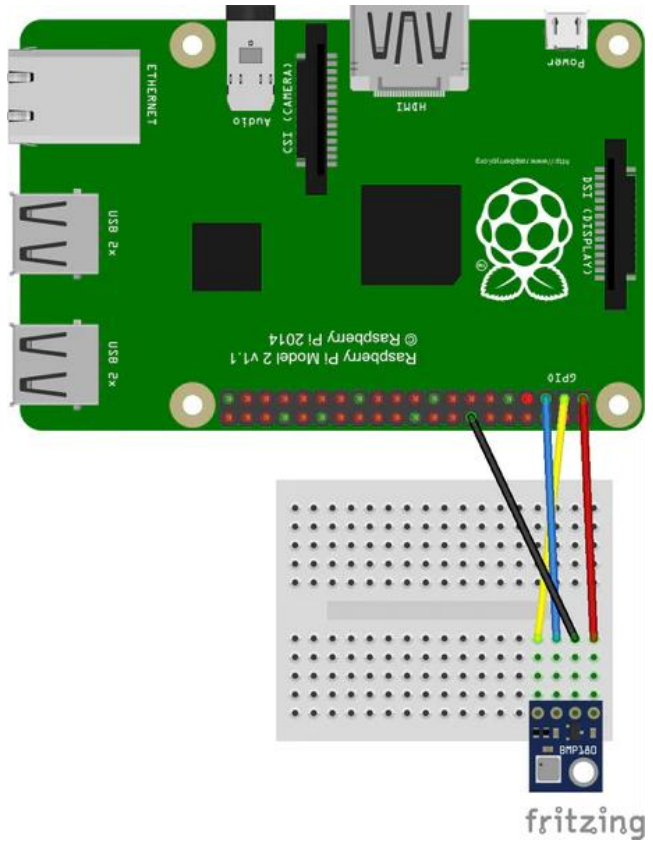
- هو بروتوكول اتصال تسلسلي متزامن يستخدم في ربط عدة وحدات طرفية بنفس الناقل
- الوحدة الطرفية تمثل الجهاز التابع (Slave) ولوحة راسبيري باي التي تتحكم بالوحدات الطرفية تمثل الجهاز الرئيس (Master)
- يتم الاتصال بين الجهاز التابع والجهاز الرئيس باستخدام سلكين (2-wire)، السلك الاول لنقل البيانات بالاتجاهين (SDA) والسلك الثاني لمزامنة النقل (SCL)
- يوفر اتصالا بعدد اقل من الابر لوصل عدة اجهزة في الانظمة المدمجة



مثال: مستشعر الحرارة والضغط الجوي

- برنامج بلغة بايثون لقراءة درجة الحرارة والضغط الجوي باستخدام مستشعر BMP180

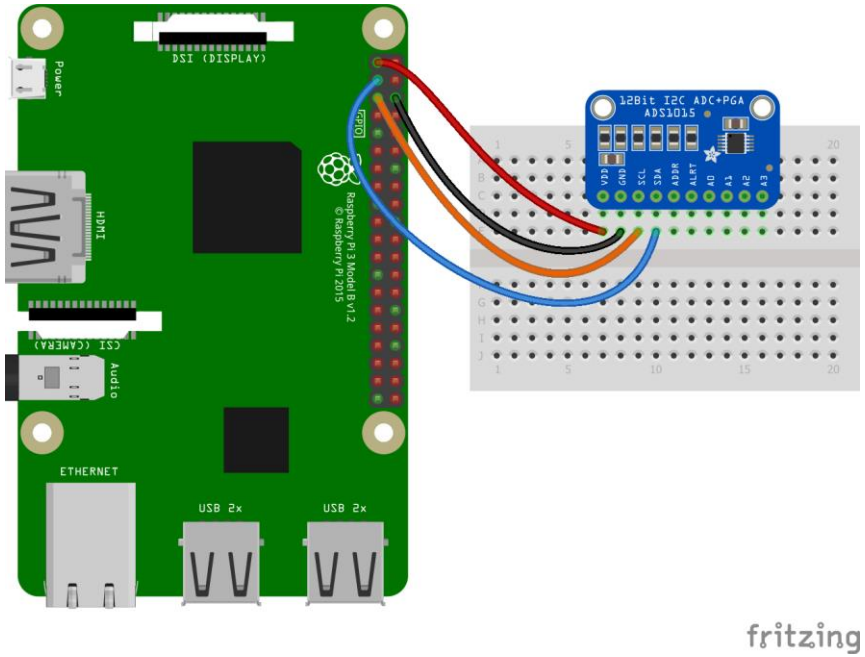
```
from Adafruit_BMP085 import BMP085
import time

address = 0x77
bmp = BMP085(address)

while True:
    temperature = bmp.readTemperature()
    pressure = bmp.readPressure()

    print(f'Temperature: {temperature:.2f} °C')
    print(f'Pressure: {pressure:.2f} Pa')

    time.sleep(2)
```



مثال: وحدة I2C ADC

- برنامج بلغة بايثون لقراءة قيمة تماثلية باستخدام وحدة ADS1015

```
import smbus
import time

ADC_ADDRESS = 0x48
bus = smbus.SMBus(1)

def read_adc(channel):
    config = [(0xC0 | ((channel & 0x07) << 4)), 0x00]
    bus.write_i2c_block_data(ADC_ADDRESS, 0x01, config)
    time.sleep(0.1)
    data = bus.read_i2c_block_data(ADC_ADDRESS, 0x00, 2)
    value = ((data[0] & 0xFF) << 8) | (data[1] & 0xFF)
    return value >> 4
while True:
    value = read_adc(0)
    voltage = (value / 2047.0) * 6.144
    print(f'Voltage: {voltage:.2f}')
    time.sleep(1)
```



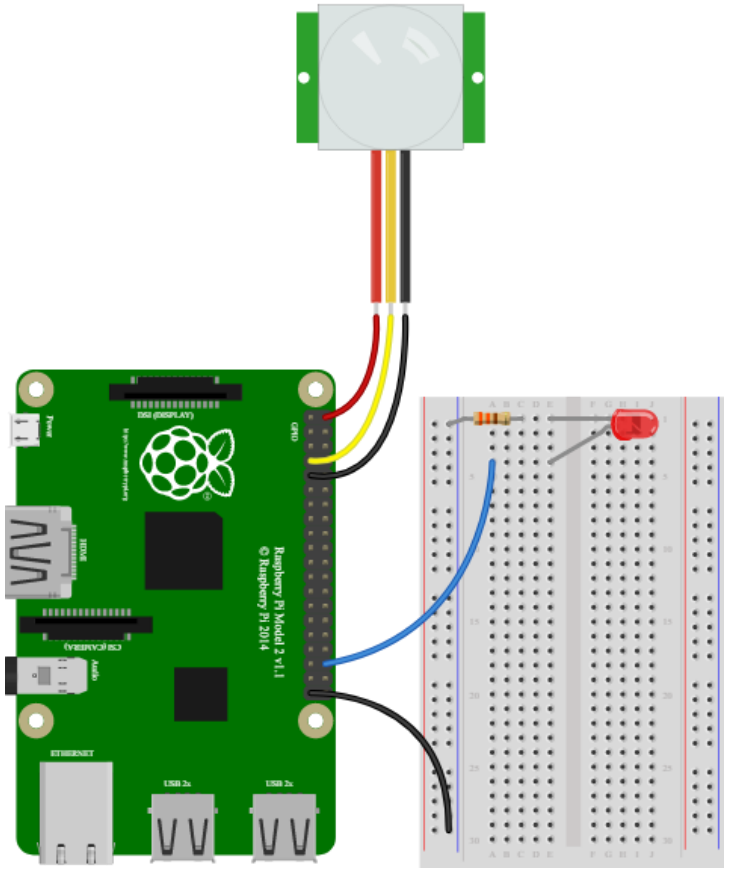
مستشعر الحركة

- مستشعر الحركة (PIR) يعتمد على الاشعة تحت الحمراء (IR) وارتدادها في رصد الحركة ويستخدم في رصد حركة الأشخاص والحيوانات أي الاجسام التي تولد حرارة
- يمكن تعديل حساسية الرصد (sensitivity) باستخدام مقياس الجهد على اليمين، وتعديل مدة بقاء الاشارة على المخرج (timeout) باستخدام مقياس الجهد على اليسار

مثال: مستشعر الحركة

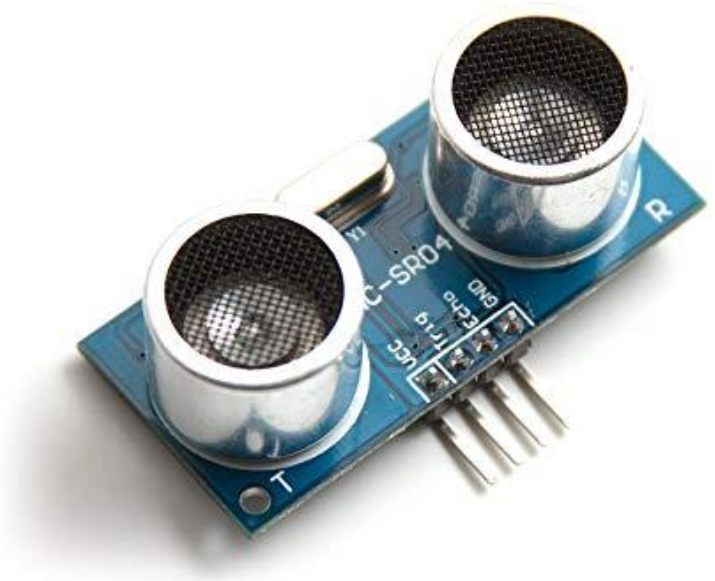
- برنامج بلغة بايثون لتشغيل دايود ضوئي عند رصد الحركة

```
import RPi.GPIO as GPIO
import time

led = 16
pir = 4
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)
GPIO.setup(pir, GPIO.IN)

while True:
    if GPIO.input(pir) == GPIO.HIGH:
        GPIO.output(led, GPIO.HIGH)
    else:
        GPIO.output(led, GPIO.LOW)
    time.sleep(1)

GPIO.cleanup()
```



مستشعر المسافة

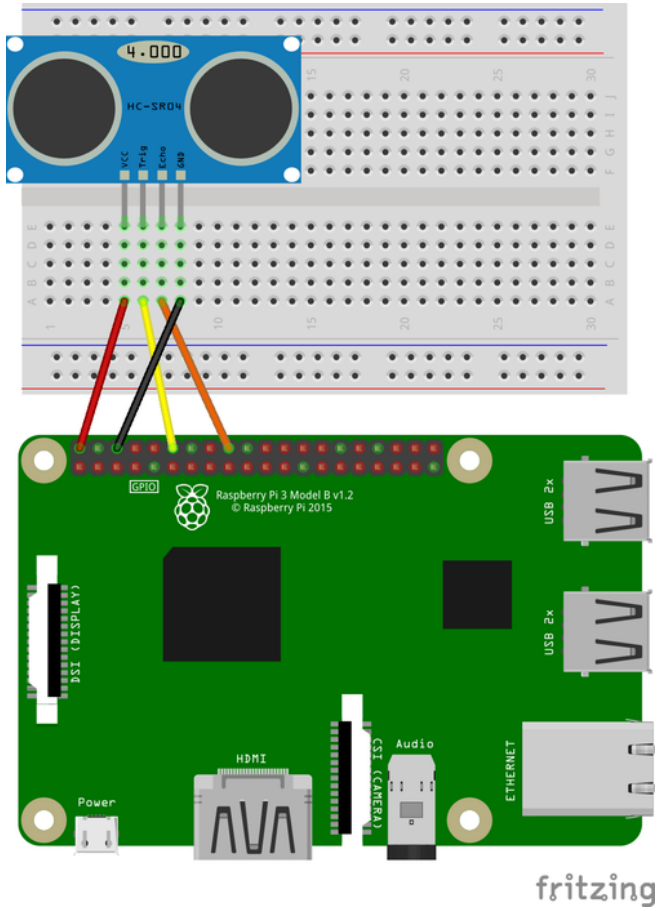
- مستشعر المسافة (Ultrasonic) يعتمد على الموجات فوق الصوتية وصداها في حساب المسافة ويستخدم في قياس المسافة بين المستشعر والاجسام التي لها سطح صلب
- مستشعر HC-SR04 لا يكتشف الاجسام التي تبعد اكثر من 4 متر او اقل من 2 سم



مثال: مستشعر المسافة

- برنامج لقياس المسافة الى اقرب جسم صلب

```
import RPi.GPIO as GPIO
import time
trig = 18
echo = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(trig, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)
def measure_distance():
    GPIO.output(trig, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(trig, GPIO.LOW)
    while GPIO.input(echo) == GPIO.LOW:
        pulse_start = time.time()
    while GPIO.input(echo) == GPIO.HIGH:
        pulse_end = time.time()
    duration = pulse_end - pulse_start
    distance = (duration / 2) * 34300
    return distance
while True:
    distance = measure_distance()
    print(f'Distance: {distance:.2f} cm')
    time.sleep(1)
GPIO.cleanup()
```



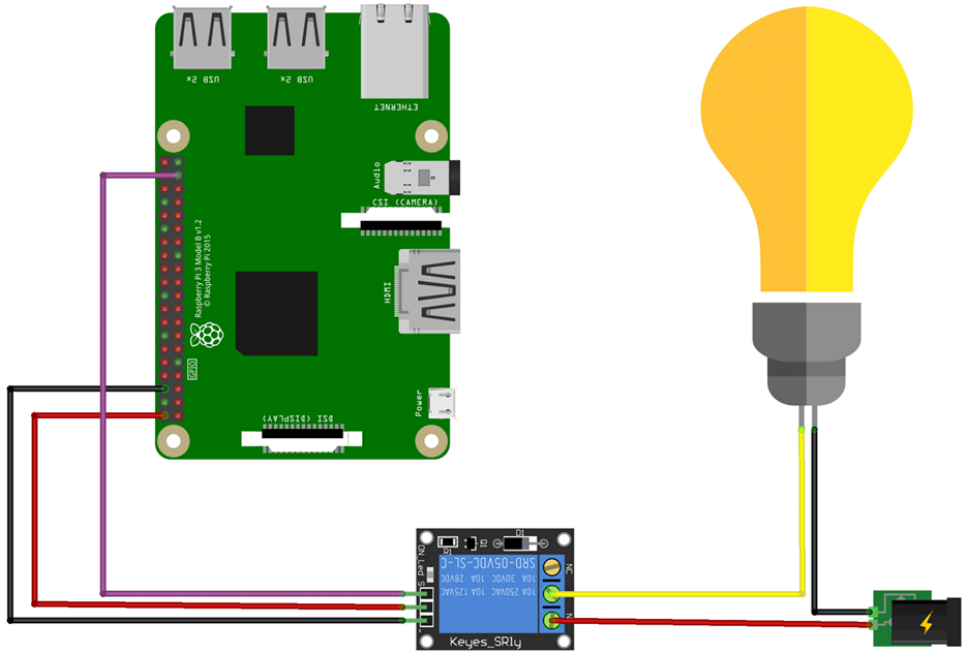
المرحل

- المرحل (Relay) هو مفتاح كهربائي يعمل على فتح واغلاق الدائرة الكهربائية
- يستخدم المرحل في التحكم بالاجهزة الكهربائية ذات التيار المرتفع مثل المصباح والمروحة
- يتم وصل المرحل بطريقتين: عندما تكون الدائرة عادة مفتوحة (normally open) او عندما تكون الدائرة عادة مغلقة (normally closed)

مثال: المرحل

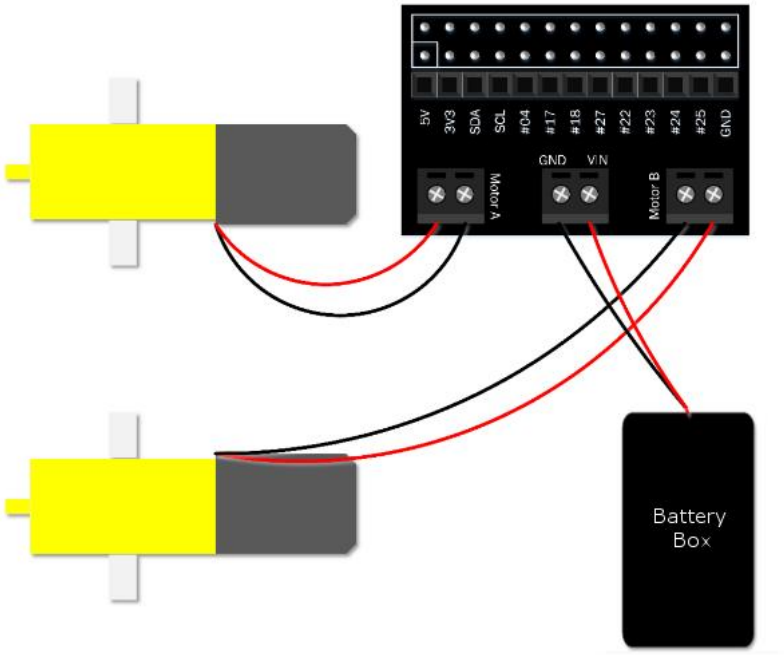
- برنامج للتحكم بالمصباح الكهربائي

```
import RPi.GPIO as GPIO
import time

bulb = 26
GPIO.setmode(GPIO.BCM)
GPIO.setup(bulb, GPIO.OUT)

while True:
    GPIO.output(bulb, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(bulb, GPIO.LOW)
    time.sleep(1)

GPIO.cleanup()
```



محرك التيار المستمر

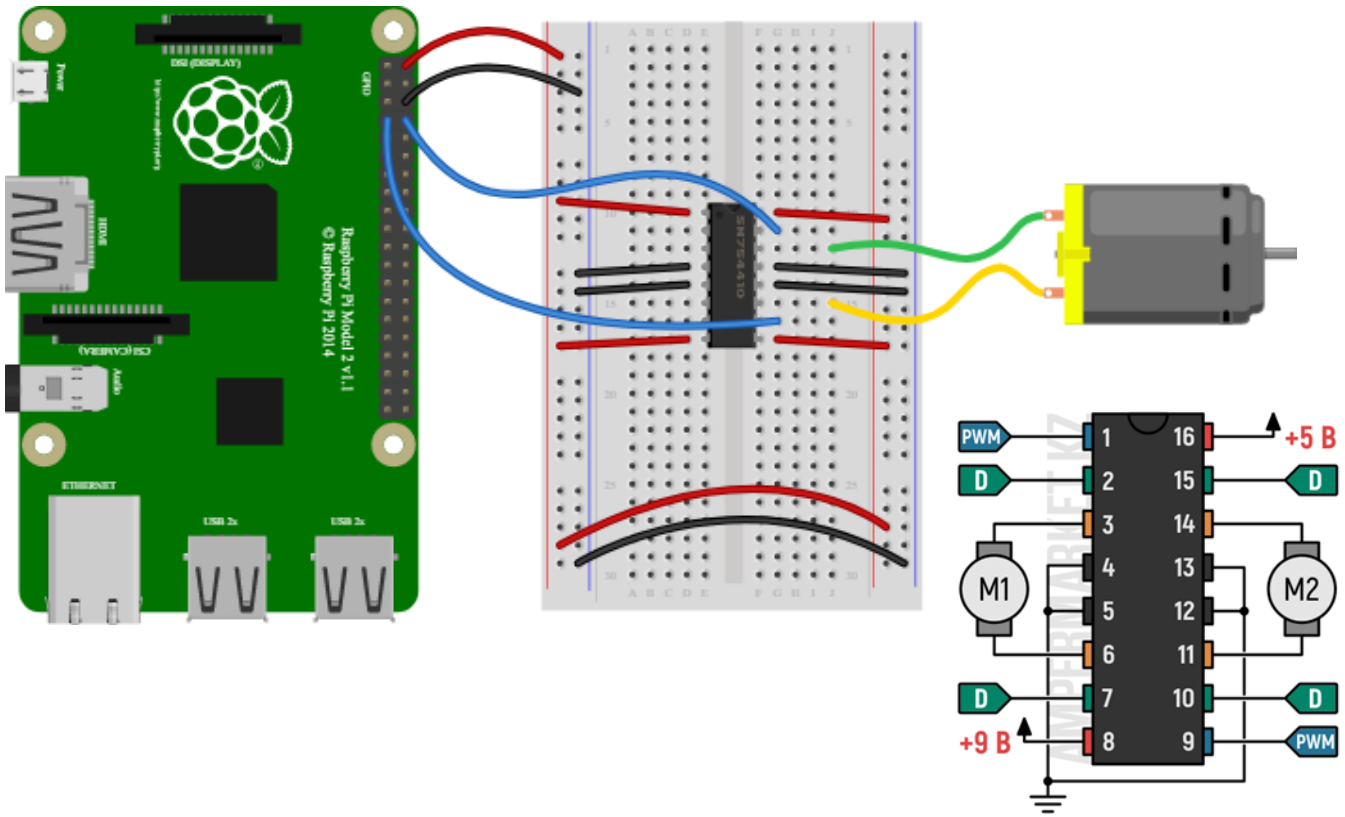
- تستخدم محركات التيار المستمر (DC) في التحكم بسرعة الدوران مثل عجلات سيارة
- يمكن تشغيل محركات التيار المستمر والتحكم بسرعتها وتغيير اتجاه دورانها باستخدام دائرة تحكم تدعى H-bridge مثل دائرة L293D ودائرة L298N



مثال: محرك التيار المستمر

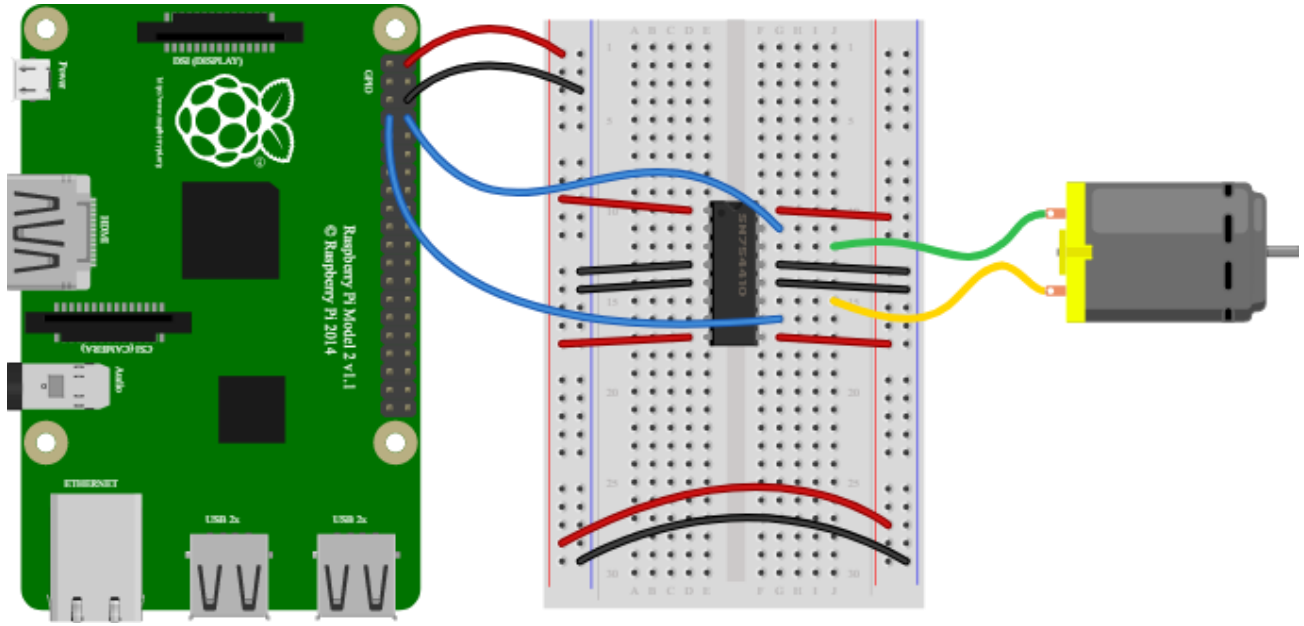
- برنامج للتحكم باتجاه دوران المحرك الكهربائي (DC)

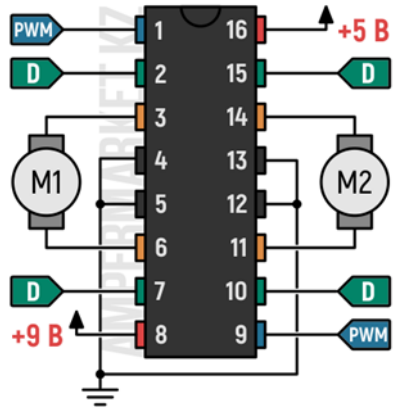
```
import RPi.GPIO as GPIO
import time

in1 = 4
in2 = 14
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1, GPIO.OUT)
GPIO.setup(in2, GPIO.OUT)

while True:
    # forward
    GPIO.output(in1, GPIO.HIGH)
    GPIO.output(in2, GPIO.LOW)
    time.sleep(5)
    # backward
    GPIO.output(in1, GPIO.LOW)
    GPIO.output(in2, GPIO.HIGH)
    time.sleep(5)
GPIO.cleanup()
```



تمرين: التحكم بسرعة المحرك

- اكتب برنامج للتحكم بسرعة المحرك الكهربائي واتجاه الدوران؟





محرك الخطوة

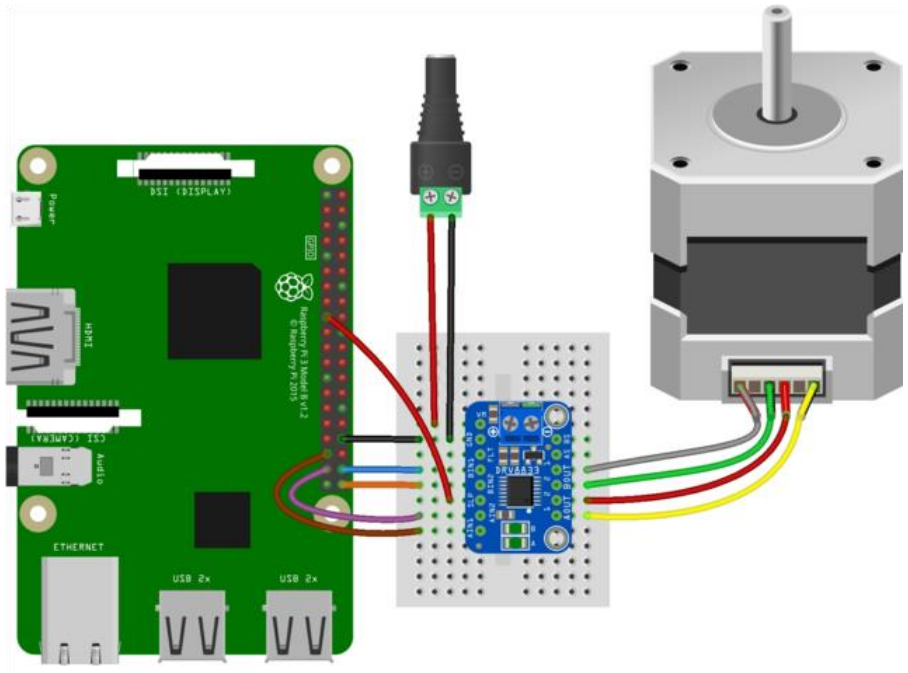
- تستخدم محركات الخطوة (Stepper) في التحكم بدرجات الدوران او عدد الدورات
- هنالك نوعان من محرك الخطوة: احادي القطبية (unipolar) وثنائي القطبية (bipolar) وكلاهما يتيح تحريك المحرك بخطوات ثابتة، مثلا 200 خطوة في الدورة اي ما يعادل 1.8 درجة لكل خطوة
- يمكن تشغيل محركات الخطوة والتحكم باتجاه دورانها باستخدام دائرة Darlington Array مثل ULN2003 اذا كانت احادية القطبية او باستخدام دائرة H-Bridge مثل L293D اذا كانت ثنائية القطبية

مثال: محرك الخطوة

- برنامج للتحكم بدرجات الدوران في المحرك الكهربائي (Stepper)

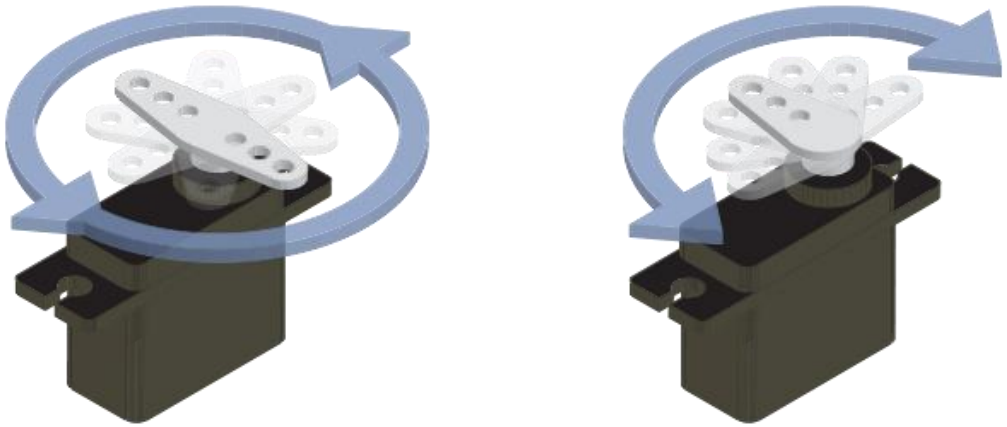
```
import RPi.GPIO as GPIO
import time

in1 = 19
in2 = 26
in3 = 21
in4 = 20

# full step
sequence = [
    [1, 0, 0, 1],
    [1, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 1, 1],
]

GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(in1, GPIO.OUT)
GPIO.setup(in2, GPIO.OUT)
GPIO.setup(in3, GPIO.OUT)
GPIO.setup(in4, GPIO.OUT)

def rotate(steps, delay):
    for _ in range(steps):
        for s in sequence:
            GPIO.output(in1, s[0])
            GPIO.output(in2, s[1])
            GPIO.output(in3, s[2])
            GPIO.output(in4, s[3])
            time.sleep(delay)

while True:
    rotate(200, 0.001)
GPIO.cleanup()
```



محرك الزاوية

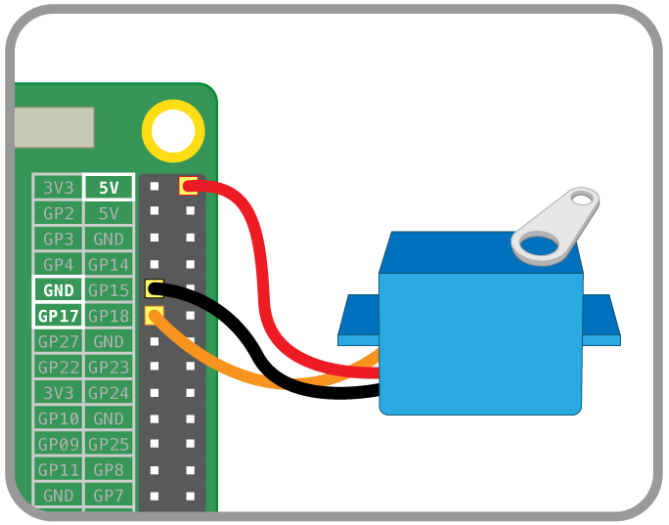
- تستخدم محركات الزاوية (Servo) في التحكم بزاوية الدوران مثل ذراع الروبوت
- محركات الزاوية تتحرك بزاوية دوران 0-180 (نصف دورة) او 0-360 (دورة كاملة)

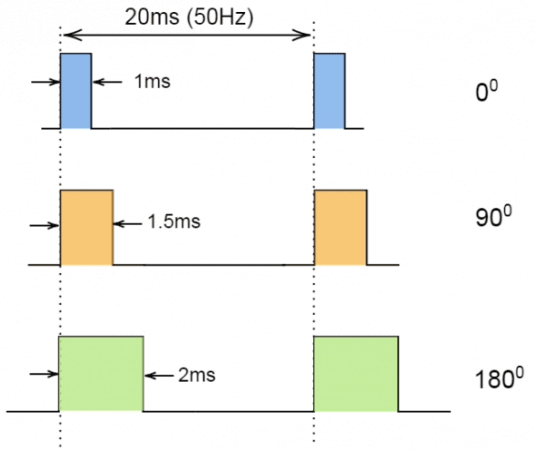


مثال: محرك الزاوية

- برنامج للتحكم بزاوية دوران المحرك الكهربائي (Servo)

```
import RPi.GPIO as GPIO
import time

servo = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(servo, GPIO.OUT)
pwm = GPIO.PWM(servo, 50)
pwm.start(0)

while True:
    pwm.ChangeDutyCycle(5)
    time.sleep(1)
    pwm.ChangeDutyCycle(7.5)
    time.sleep(1)
    pwm.ChangeDutyCycle(10)
    time.sleep(1)

pwm.stop()
GPIO.cleanup()
```





التحكم والمراقبة عبر الويب

- يمكن برمجة تطبيقات الويب في لغة بايثون باستخدام اطار العمل Flask
- يتيح Flask انشاء خادم ويب بسيط على الحاسوب المصغر للتحكم بالمشغلات ومراقبة المستشعرات من خلال متصفح الويب
- يستخدم Flask في نشر محتوى الويب الثابت (static) والديناميكي (dynamic)
- يتم انشاء وتحديث محتوى الويب الثابت يدويا باستخدام لغة HTML بينما يتم معالجة محتوى الويب الديناميكي باستخدام لغة بايثون قبل عرضه في متصفح الويب

مثال: التحكم باحد ابر GPIO

- قبل تثبيت Flask على الحاسوب المصغر، نقوم اولا بانشاء مجلد جديد لتطبيق الويب باستخدام سطر الاوامر في لينكس

```
mkdir webapp
```

- ثم نقوم بانشاء البيئة الافتراضية (virtual environment) وتفعيلها

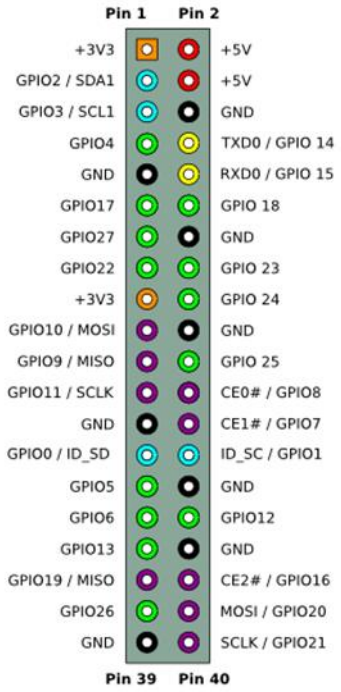
```
cd webapp
python3 -m venv .venv
source .venv/bin/activate
```

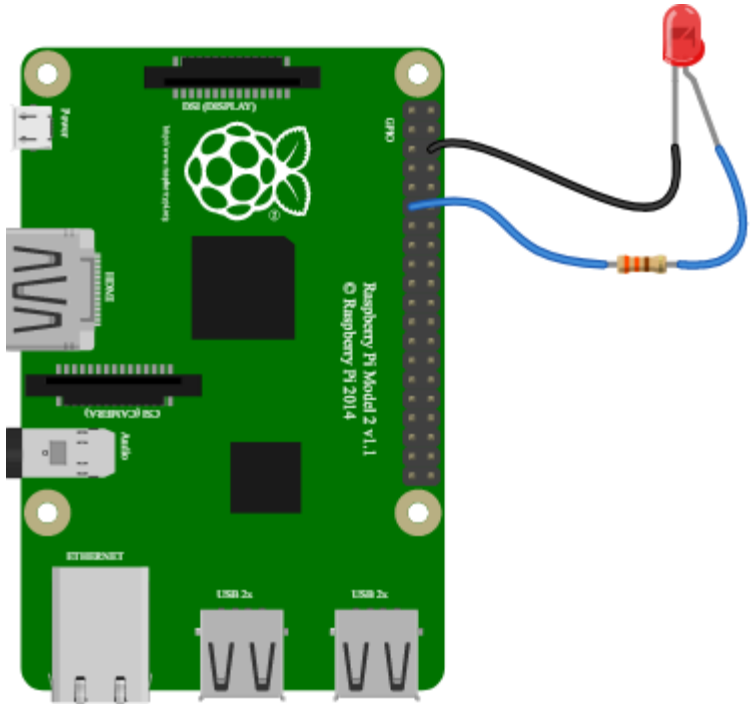
- وتثبيت Flask باستخدام اداة تثبيت الحزم البرمجية في بايثون

```
pip install Flask
```

مثال: التحكم باحد ابر GPIO

- بعد تثبيت Flask، نقوم بانشاء ملف بايثون (مثلا app.py) للتحكم باحد ابر الادخال والاخراج (GPIO) عبر الويب





مثال: التحكم باحد ابر GPIO

- ثم اضافة الكود التالي الى الملف app.py

```
from flask import Flask, render_template, request
import RPi.GPIO as GPIO

app = Flask(__name__)

led = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/toggle", methods=["POST"])
def toggle():
    GPIO.output(led, not GPIO.input(led))  # Toggle the GPIO state
    return "Success"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=True)
```

مثال: التحكم باحد ابر GPIO

- ثم انشاء المجلد templates في نفس المجلد الذي يحتوي على ملف app.py

```
mkdir templates
```

- واضافة الملف index.html داخل المجلد templates

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>GPIO Control</title>
</head>
<body>
    <h1>LED Control</h1>
    <form action="/toggle" method="post">
        <button type="submit">Toggle LED</button>
    </form>
</body>
</html>
```

مثال: التحكم باحد ابر GPIO

- اخيرا، نقوم بتشغيل ملف app.py

```
sudo python app.py
```

- يمكن الوصول الى تطبيق الويب عن طريق ادخال عنوان الحاسوب المصغر (IP) على متصفح الويب كما في المثال التالي

```
http://192.168.1.100
```