嵌入式系統設計概論與實作

曾煜棋、吳昆儒

National Yang Ming Chiao Tung University

Last week

- □ 1. PI的環境設定
- □ 2. 設定遠端桌面連線, 開啟 “direct capture mode”
- □ 3. 使用GPIO + Python + LED
- □ 4. 傳輸檔案到PI

This week

- 嵌入式應用
  - 距離資訊 (ex: 倒車雷達…)
  - 溫溼度資訊 (ex: 空氣清淨機, 寶寶攝影機…)

- Raspberry PI
  - GPIO introduction
  - Python
  - 超音波: HC-SR04
  - 溫溼度: DHT-11

HC-SR04

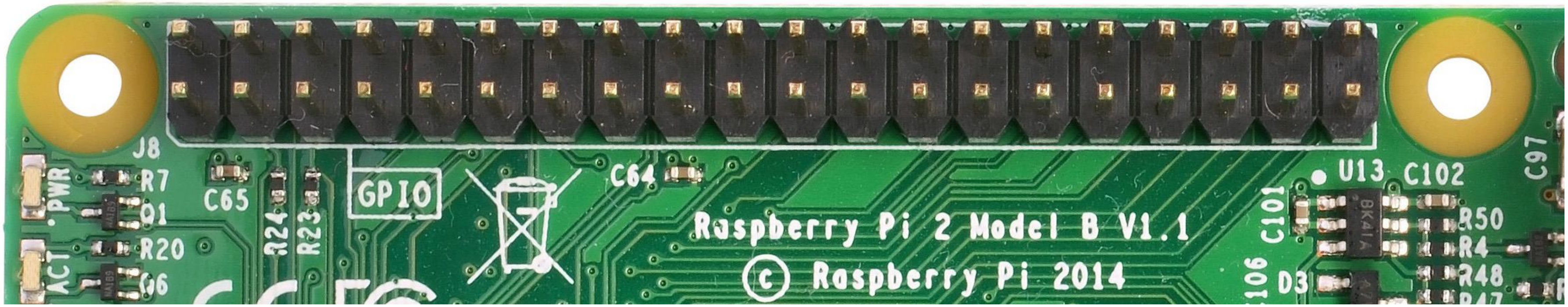
Ultrasonic

DHT-11

Thermometer + Hygrometer

1. GPIO introduction

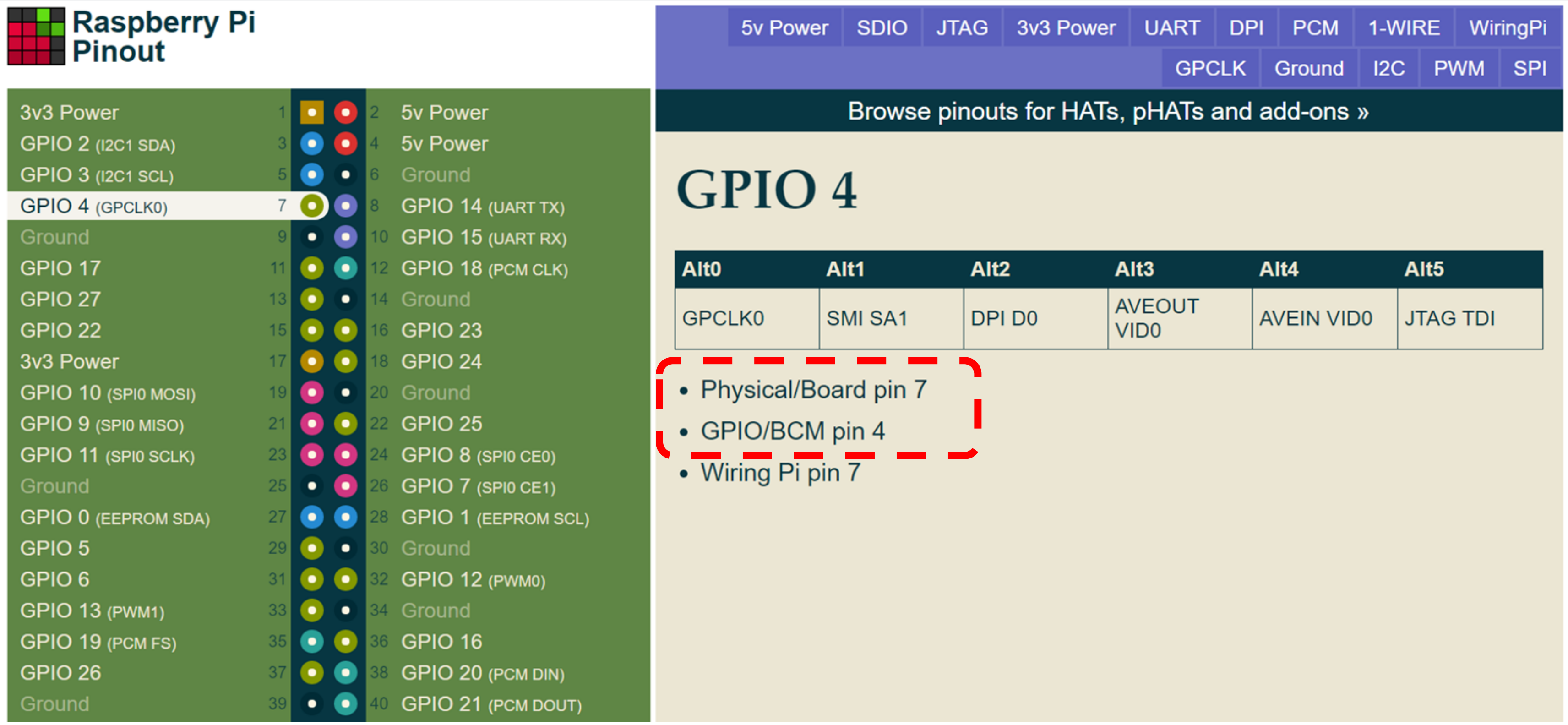
- General-purpose input/output (GPIO)
  - You can set PIN as **Input** or **Output** or **both**(Input and output)
    - Input: write a value on PIN
    - Output: Read the value on PIN



https://www.raspberrypi.org/documentation/usage/gpio/README.md

1. GPIO introduction

- Pin number != GPIO number
  - Physical numbering vs. GPIO numbering



3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

GPIO 4

Alt0	Alt1	Alt2	Alt3	Alt4	Alt5
GPCLK0	SMI SA1	DPI D0	AVEOUT VID0	AVEIN VID0	JTAG TDI

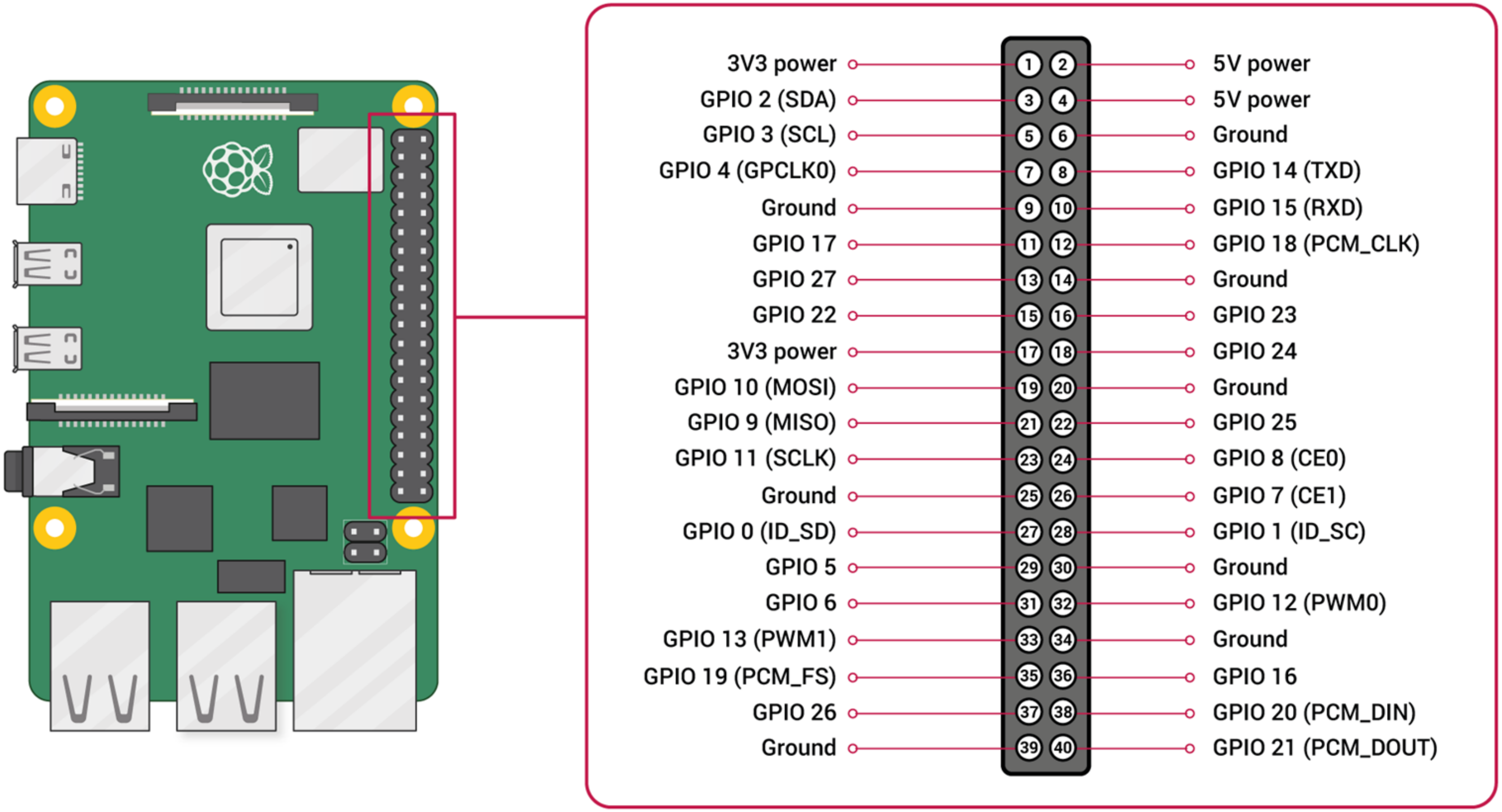
- Physical/Board pin 7
- GPIO/BCM pin 4
- Wiring Pi pin 7

https://pinout.xyz/pinout/pin7_gpio4

http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

1. GPIO introduction

□ The PIN (Physical) numbering is in Z-shape



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

https://www.raspberrypi.org/documentation/usage/gpio/

1. GPIO Limitations

- □ Do not put more than 3.3V on any GPIO pin being used as an input.
- □ Do not draw more than 16mA per output and keep the total for all outputs below 50mA in total for an older 26-pin Raspberry PI, and below 100mA on a 40-pin Raspberry PI.
- □ When using LEDs, 3mA is enough to light a red LED reasonably brightly with a 470 Ohm series resistor.
- □ Do not poke at the GPIO connector with a screwdriver or any metal object when the PI is powered up.
- □ Do not power the PI with more than 5V.
- □ Do not draw more than a total of 250mA from the 5V supply pins.

From: Raspberry Pi Cookbook: Software and Hardware Problems and Solutions
https://books.google.com.tw/books?id=0skvDAAAQBAJ&pg=PT270&lpg=PT270#v=onepage&q&f=false

1. GPIO Limitations

- 在GPIO 輸入電壓不可以超過3.3V
  - 超音波的Echo腳位是5V, 直接用會燒毀(!?)
- 不要拿金屬物體接觸GPIO PIN (會短路)
  - 使用杜邦線, 針腳不要碰到板子
- 使用GPIO PIN啟動PI時, 電壓不可以超過5V
- GPIO PIN的輸出電流有上限
  - 早期資料
    - 3.3V的供電腳位不可以超過50mA
    - 5V的供電腳位不可以超過250mA
  - 實驗資訊
    - 3.3V的供電腳位大約可以支援500mA

https://pinout.xyz/pinout/3v3_power

2. Ultrasonic (HC-SR04)

- Speed of sound
  - At 20°C (68°F), the speed is 343 m/s.
  - The approximate speed of sound (c) can be calculated from:

$$\mathbf{c_{air} = (331.3 + 0.606 * \boldsymbol{\theta})} \ \textbf{(m/s)}$$

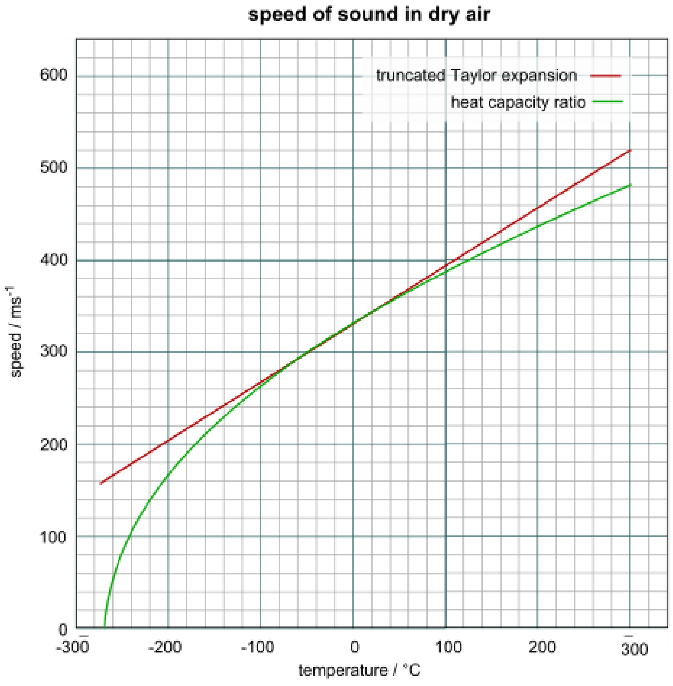
where $\boldsymbol{\theta}$ is the temperature in degrees Celsius (°C).

$$Speed = \frac{Distance}{Time}$$

$$34300 = \frac{Distance}{Time/2}$$

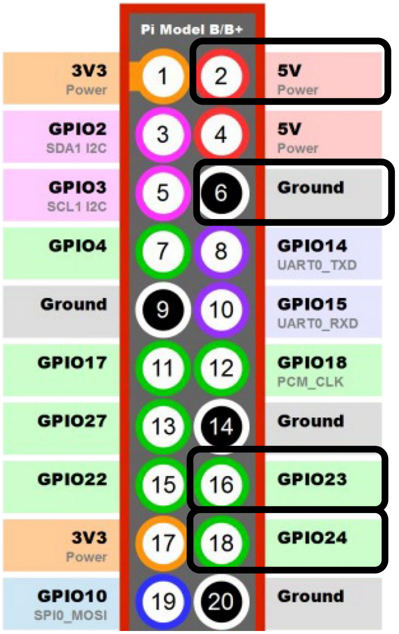
$$17150 = \frac{Distance}{Time}$$

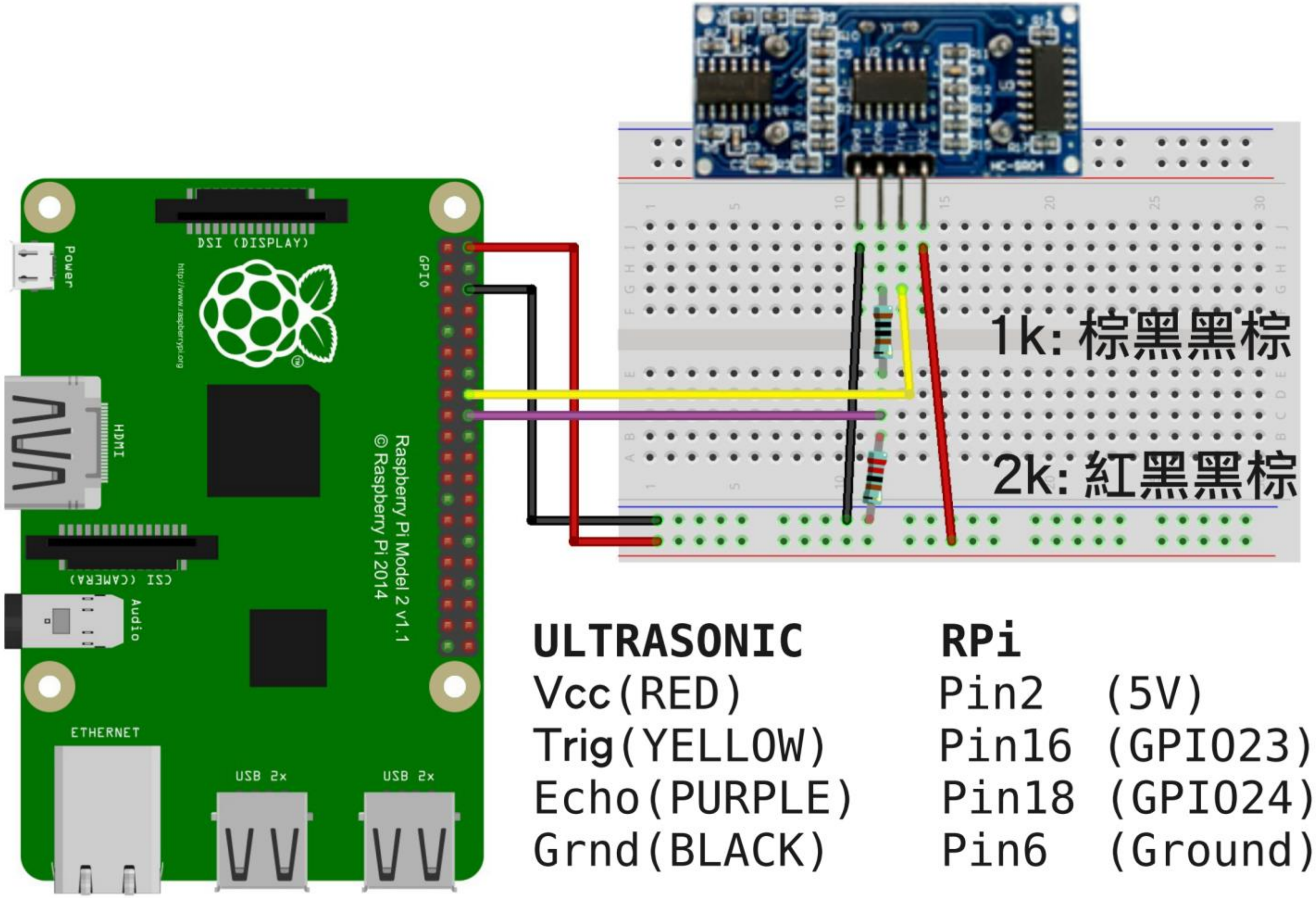
$$17150 \times Time = Distance$$



http://en.wikipedia.org/wiki/Speed_of_sound
https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

2. Ultrasonic (HC-SR04)

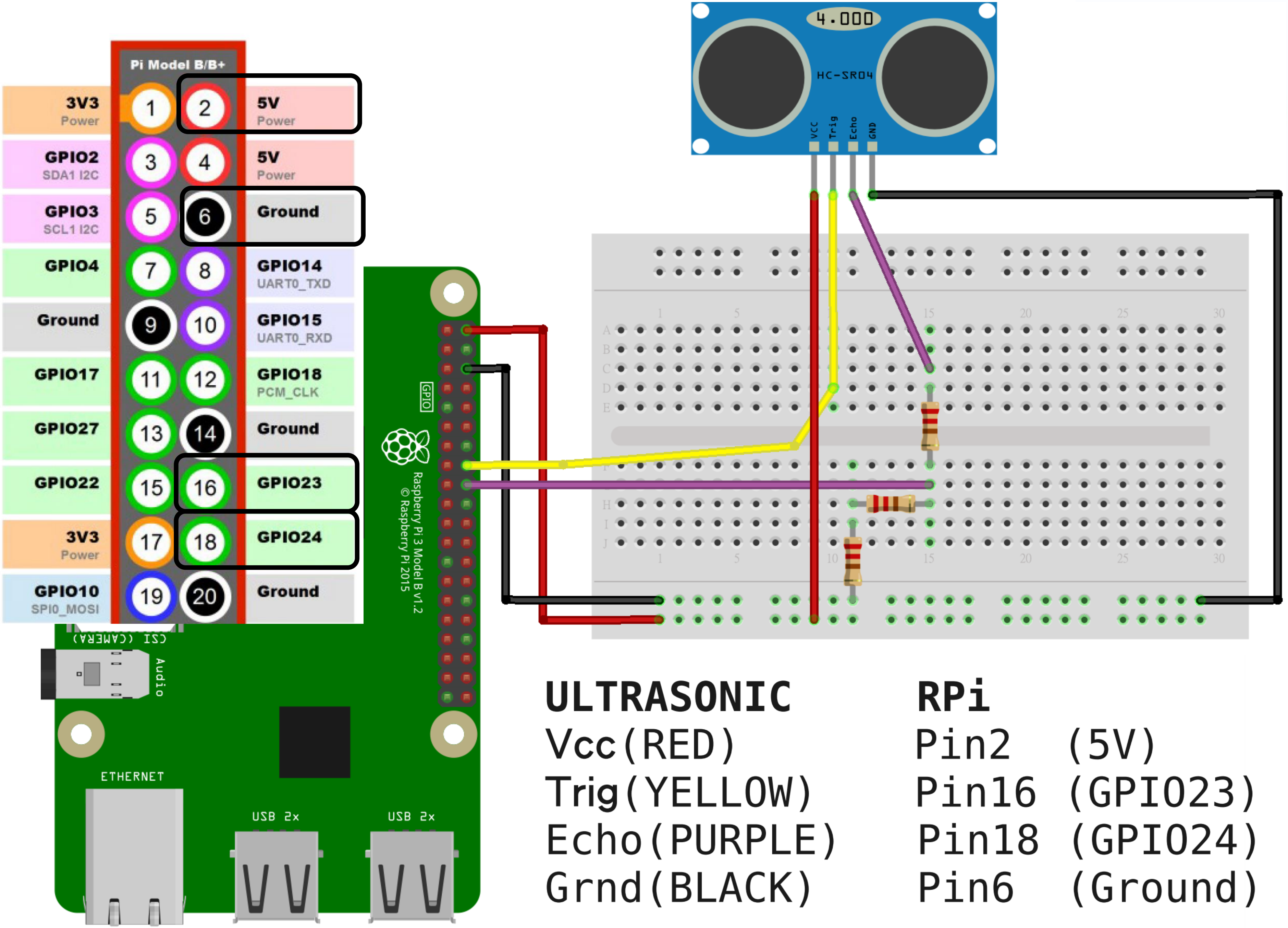




1k: 棕黑黑棕

2k: 紅黑黑棕

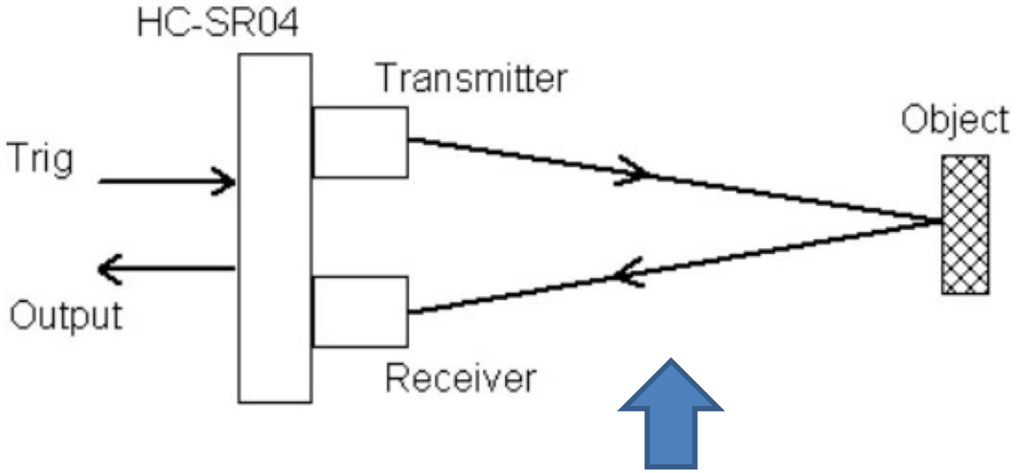
ULTRASONIC	RPi	
Vcc(RED)	Pin2	(5V)
Trig(YELLOW)	Pin16	(GPIO23)
Echo(PURPLE)	Pin18	(GPIO24)
Grnd(BLACK)	Pin6	(Ground)



ULTRASONIC	RPi
Vcc(RED)	Pin2 (5V)
Trig(YELLOW)	Pin16 (GPIO23)
Echo(PURPLE)	Pin18 (GPIO24)
Grnd(BLACK)	Pin6 (Ground)

2. Ultrasonic (HC-SR04)

- 內建發射 (40kHz) 與接收電路
- 根據發射與接收的時間差計算距離
- 特殊功能 :US-020(長距離) 、 US-100(溫度補償)

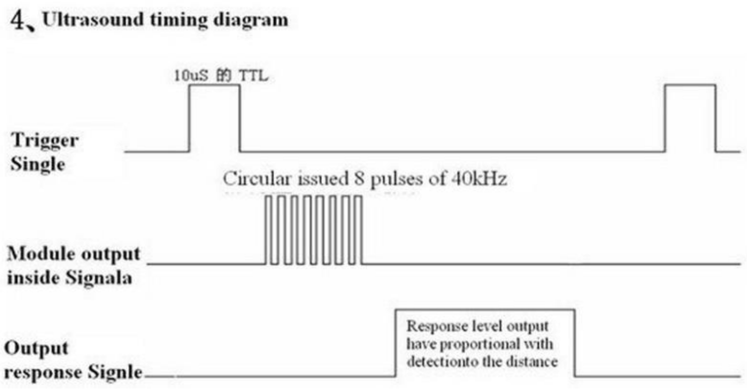


可得到時間差

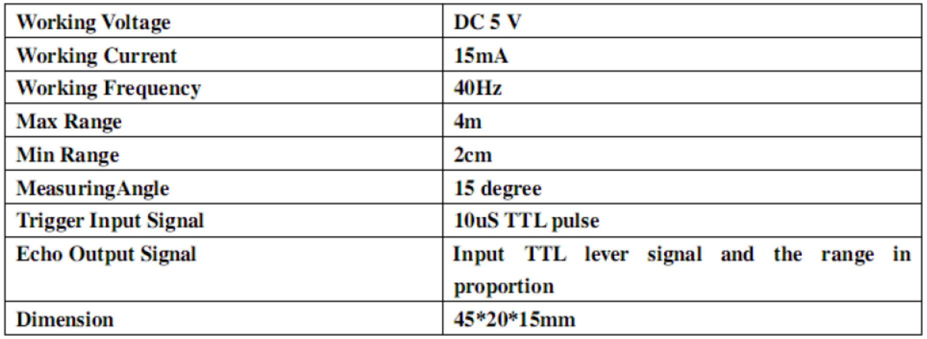
VCC, Trigger, Echo, GND

http://coopermaa2nd.blogspot.tw/2012/09/hc-sr04.html

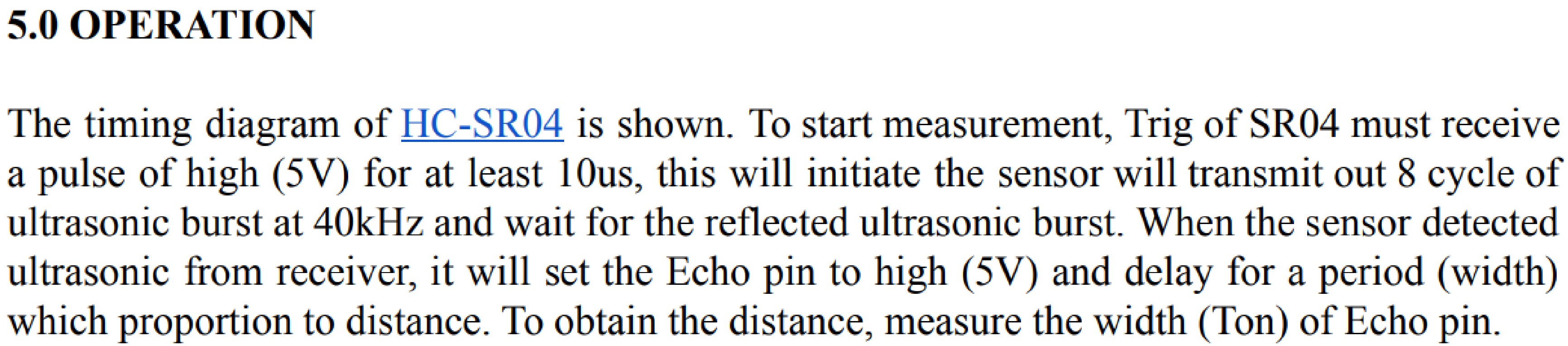
2. Ultrasonic (HC-SR04)


4、Ultrasound timing diagram

10uS 的 TTL
Trigger Single
Circular issued 8 pulses of 40kHz
Module output inside Signala
Output response Signle
Response level output have proportional with detectionto the distance

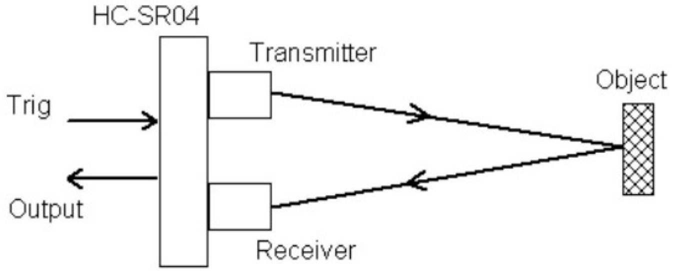


Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm



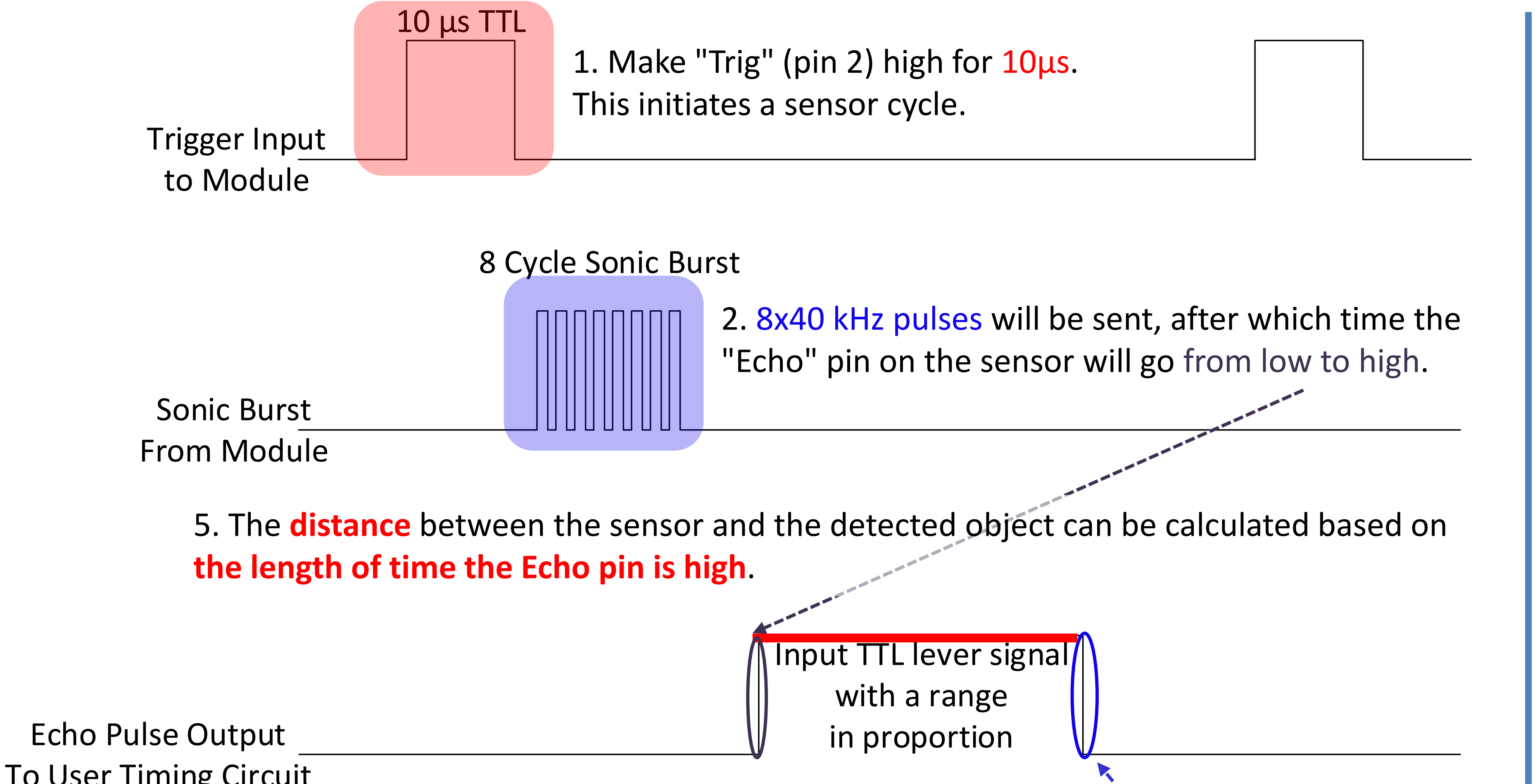
5.0 OPERATION

The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.



3. The 40kHz sound wave will bounce off the nearest object and return to the sensor.

VCC, Trigger, Echo, GND



10 µs TTL

1. Make "Trig" (pin 2) high for 10µs. This initiates a sensor cycle.

Trigger Input to Module

8 Cycle Sonic Burst

2. 8x40 kHz pulses will be sent, after which time the "Echo" pin on the sensor will go from low to high.

Sonic Burst From Module

5. The **distance** between the sensor and the detected object can be calculated based on **the length of time the Echo pin is high**.

Input TTL lever signal with a range in proportion

Echo Pulse Output To User Timing Circuit

4. When the sensor detects the reflected sound wave, the Echo pin will go low again.

http://www.micropik.com/PDF/HCSR04.pdf

2. Ultrasonic (HC-SR04)

1. Make "Trig" (pin 2) high for 10µs. This initiates a sensor cycle.
2. 8x40 kHz pulses will be sent, after which time the "Echo" pin on the sensor will go from low to high.
3. The 40kHz sound wave will bounce off the nearest object and return to the sensor.
4. When the sensor detects the reflected sound wave, the Echo pin will go low again.
5. The distance between the sensor and the detected object can be calculated based on the length of time the Echo pin is high.
6. If no object is detected, the Echo pin will stay high for 38ms and then go low.

Datasheet: http://www.micropik.com/PDF/HCSR04.pdf

2. Ultrasonic (HC-SR04)

- TRIG 腳位收到高電位 (3.3V) 後發送超聲波
- ECHO 腳位維持低電位 (0V), 收到回應後拉到高電位 (5V)
- Raspberry Pi 腳位的容忍電位為 3.3V
  - => 將 ECHO 腳位的 5V 降壓為 3.3V 左右

$$\frac{3.3}{5} = \frac{R2}{1000 + R2}$$

$$0.66 = \frac{R2}{1000 + R2}$$

$$0.66(1000 + R2) = R2$$

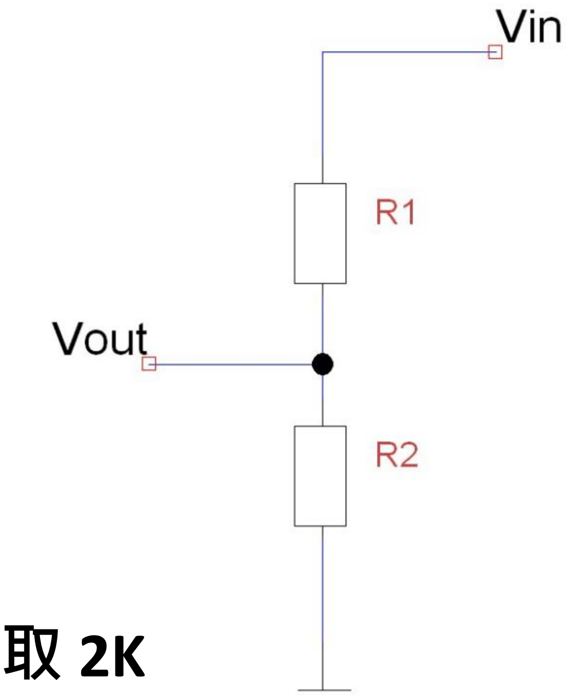
$$660 + 0.66R2 = R2$$

$$660 = 0.34R2$$

$$1941 = R2$$

$$Vout = Vin \times \frac{R2}{R1 + R2}$$

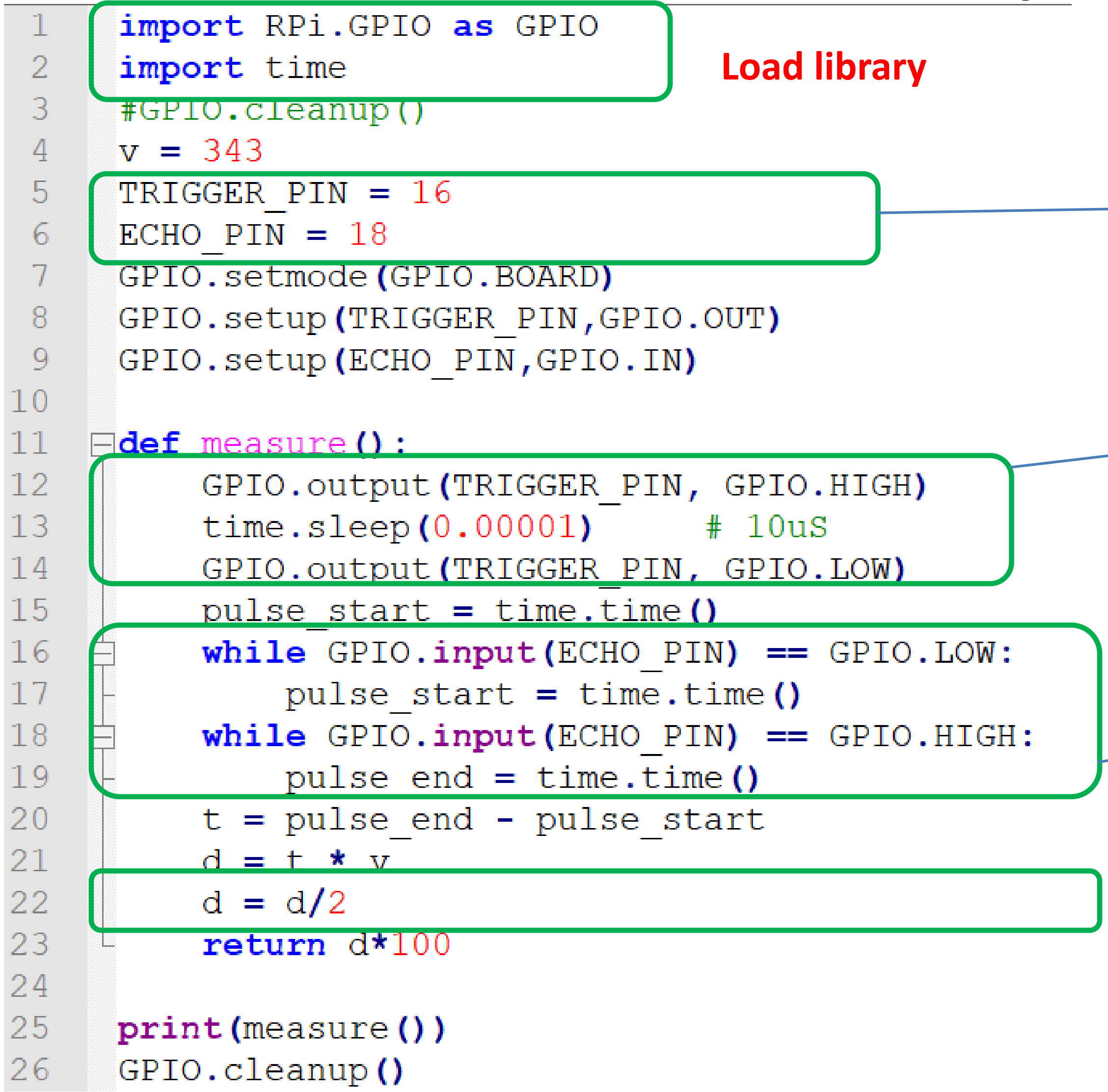
$$\frac{Vout}{Vin} = \frac{R2}{R1 + R2}$$



計算結果: R1=1K, R2 取 2K

https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

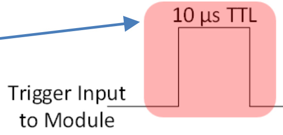
2. Ultrasonic (HC-SR04)



```
import RPi.GPIO as GPIO
import time
#GPIO.cleanup()
v = 343
TRIGGER_PIN = 16
ECHO_PIN = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(TRIGGER_PIN,GPIO.OUT)
GPIO.setup(ECHO_PIN,GPIO.IN)

def measure():
    GPIO.output(TRIGGER_PIN, GPIO.HIGH)
    time.sleep(0.00001)      # 10uS
    GPIO.output(TRIGGER_PIN, GPIO.LOW)
    pulse_start = time.time()
    while GPIO.input(ECHO_PIN) == GPIO.LOW:
        pulse_start = time.time()
    while GPIO.input(ECHO_PIN) == GPIO.HIGH:
        pulse_end = time.time()
    t = pulse_end - pulse_start
    d = t * v
    d = d/2
    return d*100

print(measure())
GPIO.cleanup()
```

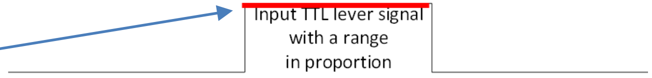
Load library

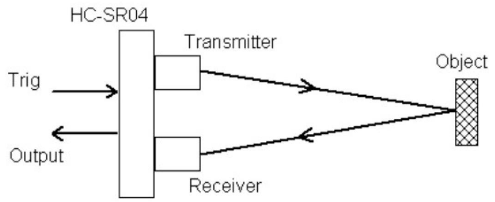
ULTRASONIC	RPi
Vcc(RED)	Pin2 (5V)
Trig(YELLOW)	Pin16 (GPIO23)
Echo(PURPLE)	Pin18 (GPIO24)
Grnd(BLACK)	Pin6 (Ground)



1. Make "Trig" (pin 2) high for 10µs. This initiates a sensor cycle.

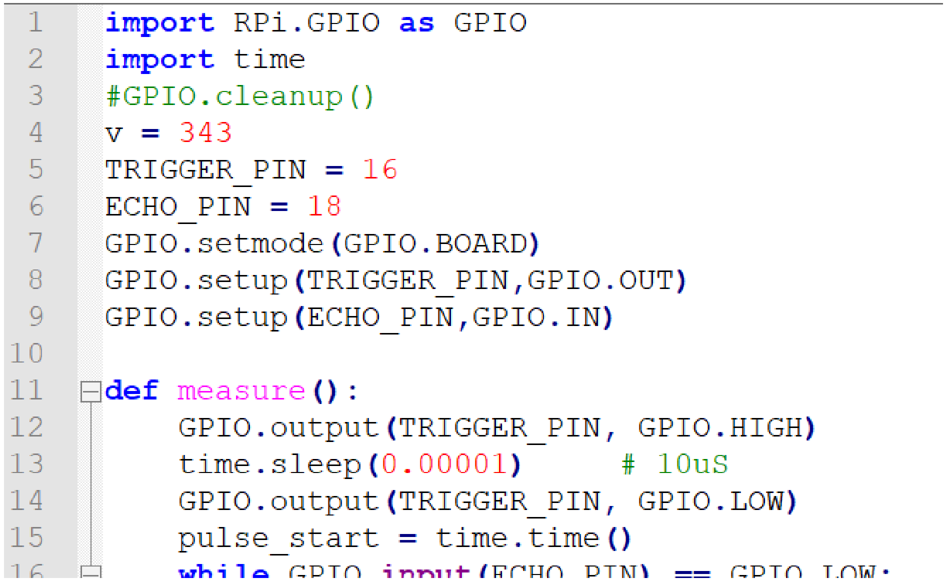
5. The **distance** between the sensor and the detected object can be calculated based on **the length of time the Echo pin is high**.





2. Ultrasonic (HC-SR04)

- Create a new Python code
  - nano ultrasonic_distance.py
- Run the code
  - sudo python ultrasonic_distance.py

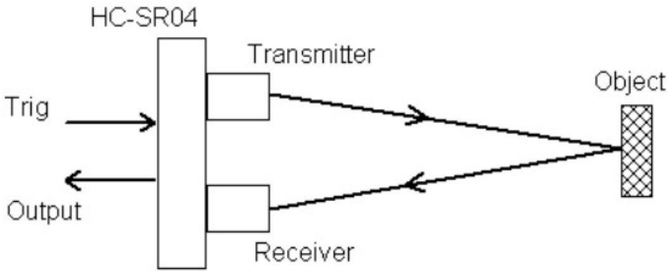


```python
import RPi.GPIO as GPIO
import time
#GPIO.cleanup()
v = 343
TRIGGER_PIN = 16
ECHO_PIN = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(TRIGGER_PIN,GPIO.OUT)
GPIO.setup(ECHO_PIN,GPIO.IN)

def measure():
    GPIO.output(TRIGGER_PIN, GPIO.HIGH)
    time.sleep(0.00001)      # 10uS
    GPIO.output(TRIGGER_PIN, GPIO.LOW)
    pulse_start = time.time()
    while GPIO.input(ECHO_PIN) == GPIO.LOW:
```

This is picture!
Try to write code by yourself.

Discussion 1

- □ Why do we need to put resistors in the circuit?
- □ Read datasheet. What is the **max and min distance** that it can detect?
- □ Based on distance measurement, is there any other application?



max distance??

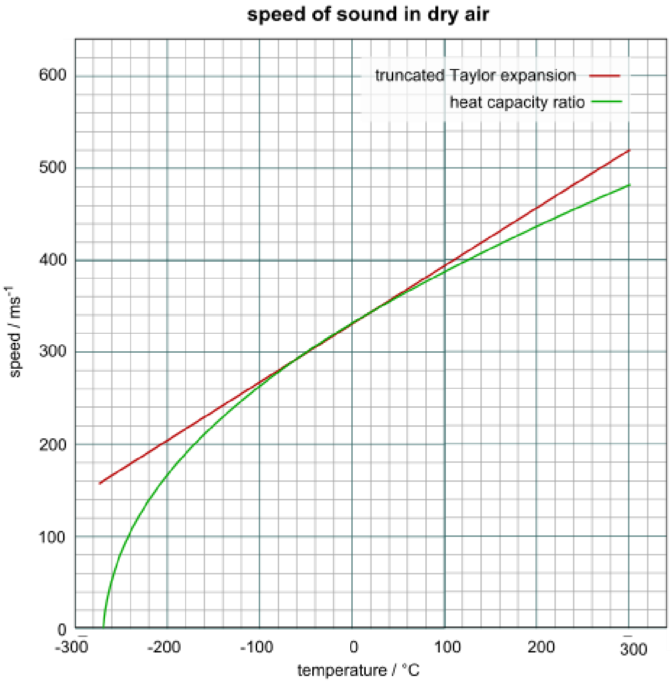
VCC, **Trigger, Echo**, GND

3. Temperature (DHT-11)

- Speed of sound
  - At 20°C (68°F), the speed is 343 m/s.
  - The approximate speed of sound (c) can be calculated from:

$$\mathbf{c_{air} = (331.3 + 0.606 * \theta)}\ \textbf{(m/s)}$$

where θ is the temperature in degrees Celsius (°C).


speed of sound in dry air

$$Speed = \frac{Distance}{Time}$$

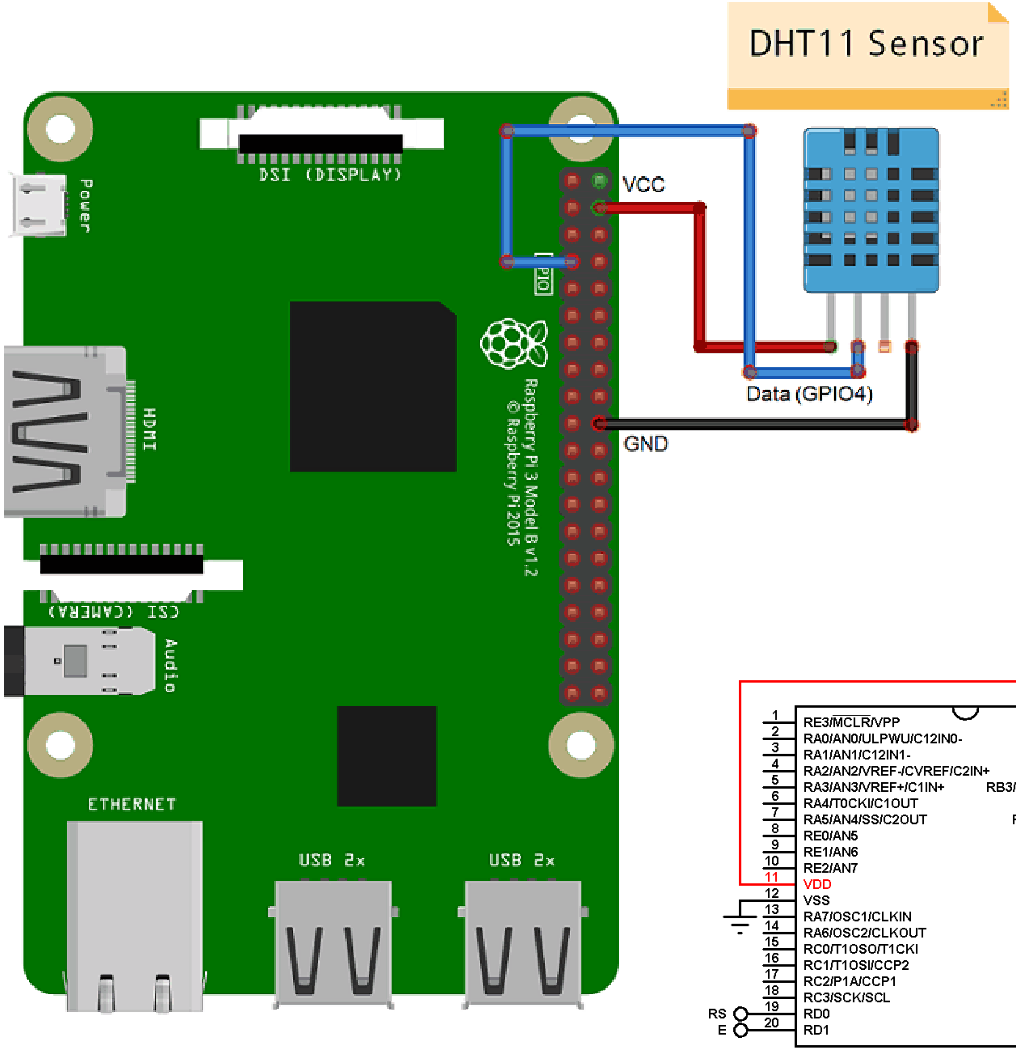
$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

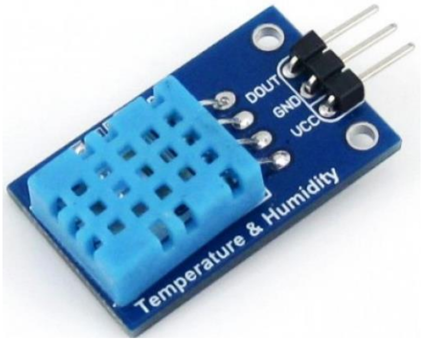
$$17150\ \times Time = \ Distance$$

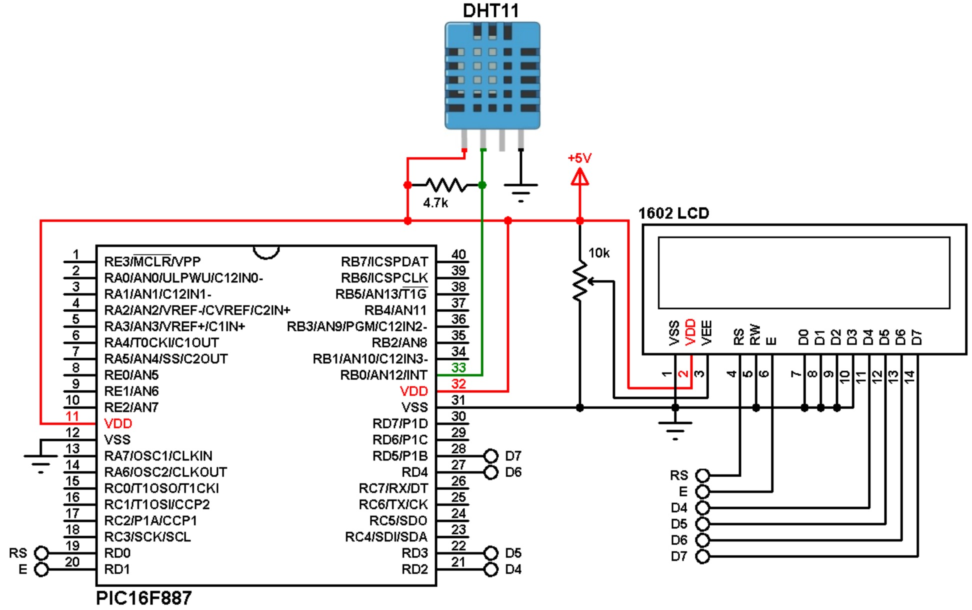
http://en.wikipedia.org/wiki/Speed_of_sound
https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

3. DHT-11



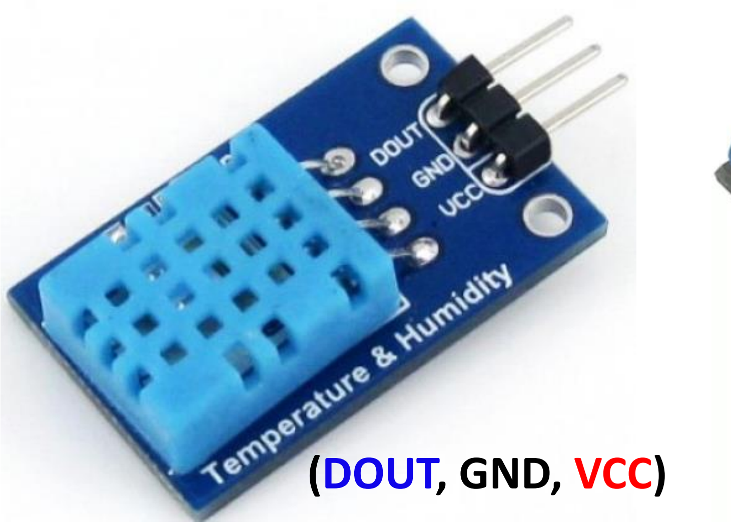
(DOUT, GND, VCC)



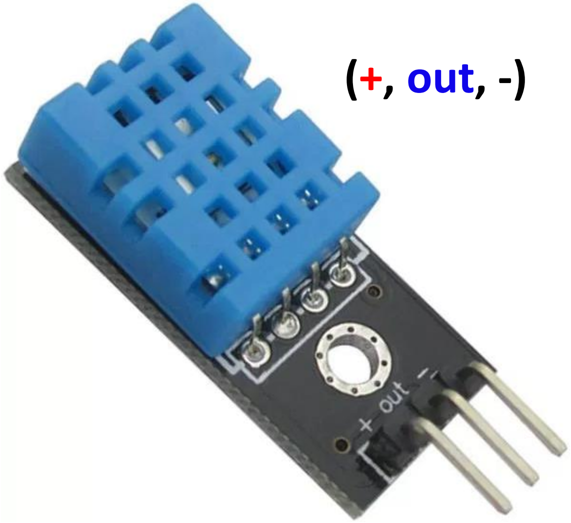


https://simple-circuit.com/mikroc-dht11-sensor-pic-microcontroller/

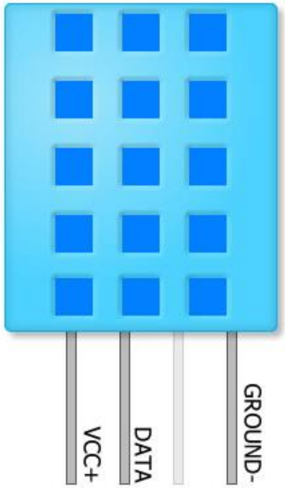
3. All kinds of DHT-11



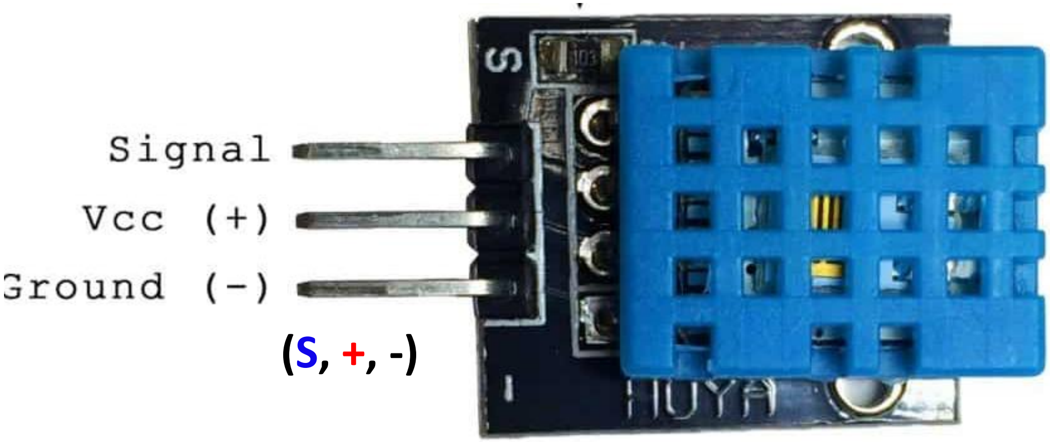
(DOUT, GND, VCC)



(+, out, -)

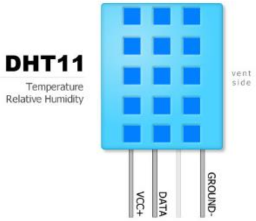


(VCC, Data, GND)

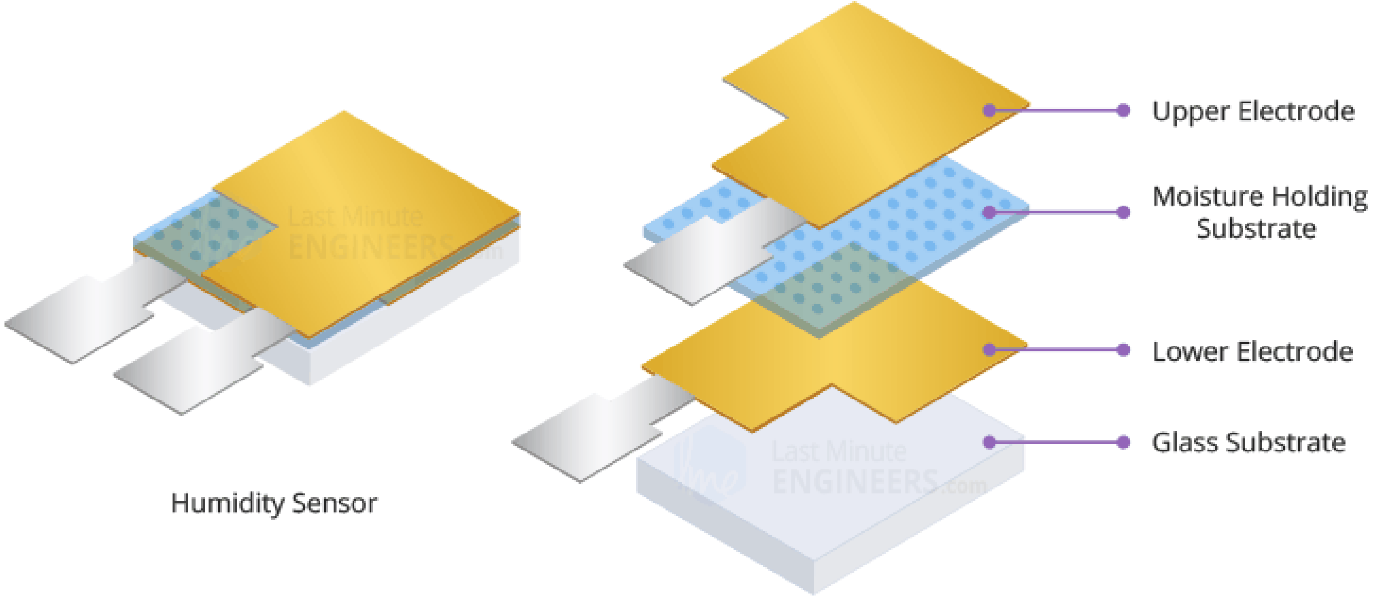


(S, +, -)

3. DHT-11



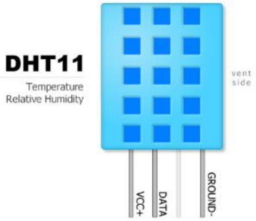
Humidity sensing component



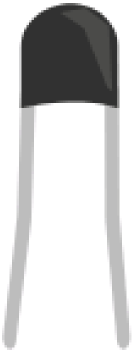
The change in resistance between the two electrodes is proportional to the relative humidity.

https://lastminuteengineers.com/dht11-module-arduino-tutorial/

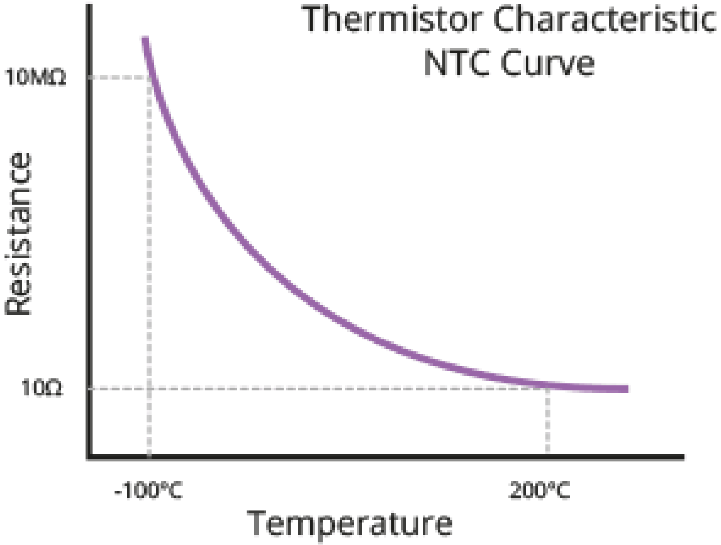
3. DHT-11



Temperature sensing component



NTC Thermistor

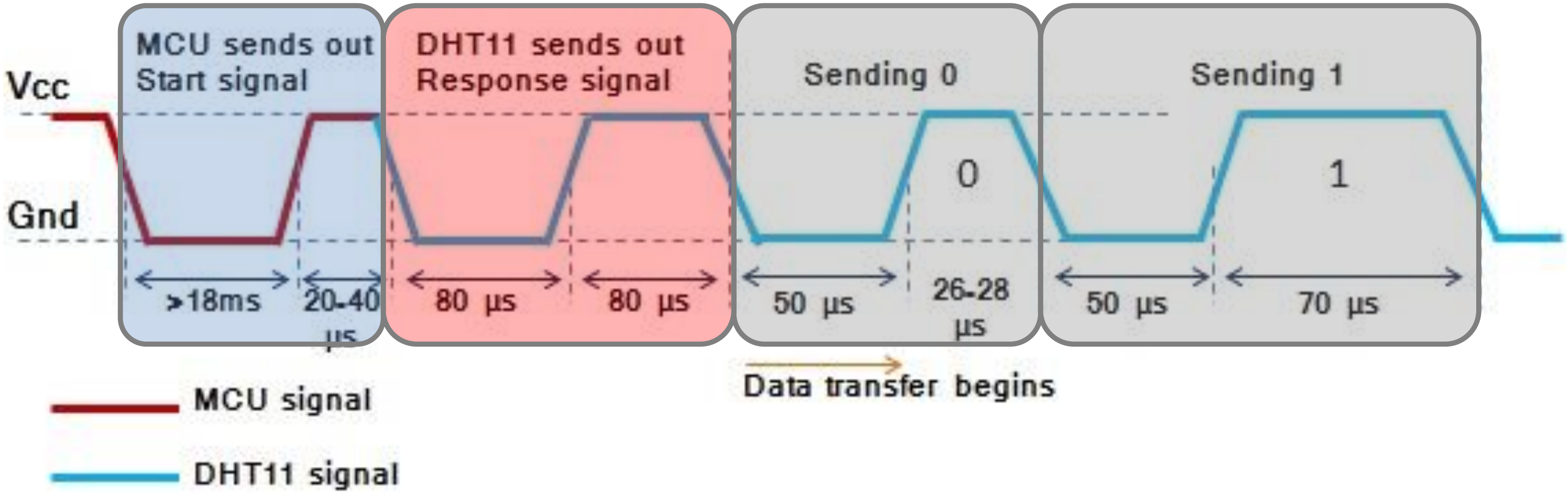


*A thermistor is a **thermal resistor** whose resistance changes drastically with temperature.*

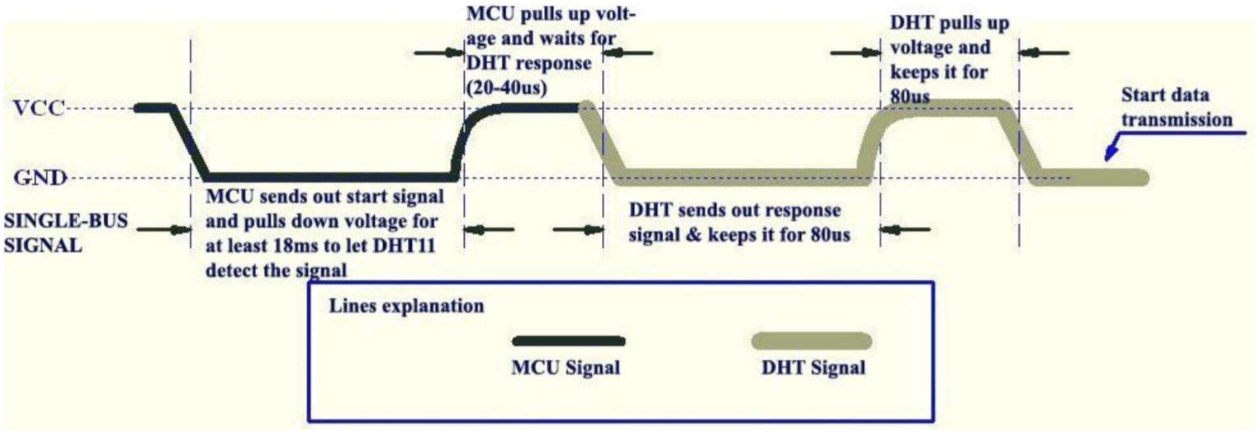
https://lastminuteengineers.com/dht11-module-arduino-tutorial/

3. DHT-11: Communication Process

- MCU: send start signal, then collect data from DHT11
  - **Data** (40-bit) =
    Integer Byte of RH + Decimal Byte of RH +
    Integer Byte of Temp. + Decimal Byte of Temp. +
    Checksum Byte.
    (4 byte data + 1 byte checksum)

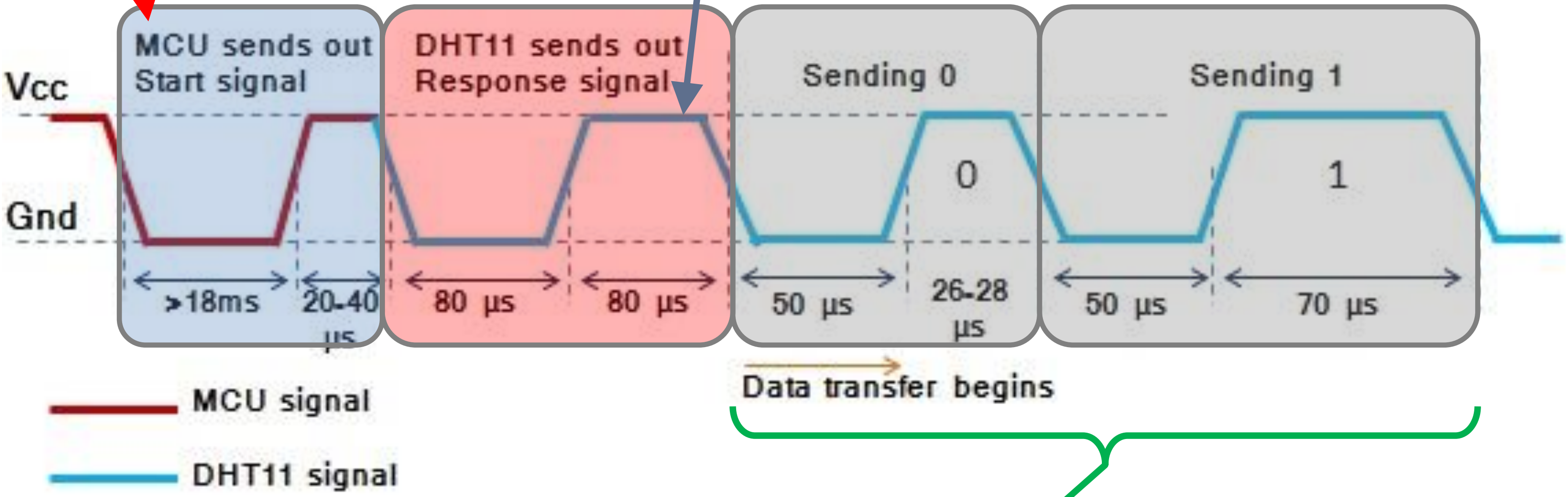


https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf


MCU pulls up voltage and waits for DHT response (20-40us)
DHT pulls up voltage and keeps it for 80us
VCC
Start data transmission
GND
SINGLE-BUS SIGNAL
MCU sends out start signal and pulls down voltage for at least 18ms to let DHT11 detect the signal
DHT sends out response signal & keeps it for 80us
Lines explanation
MCU Signal
DHT Signal

1. Send “Start signal”

2. pull up voltage and keeps it for 80us and prepares for data transmission

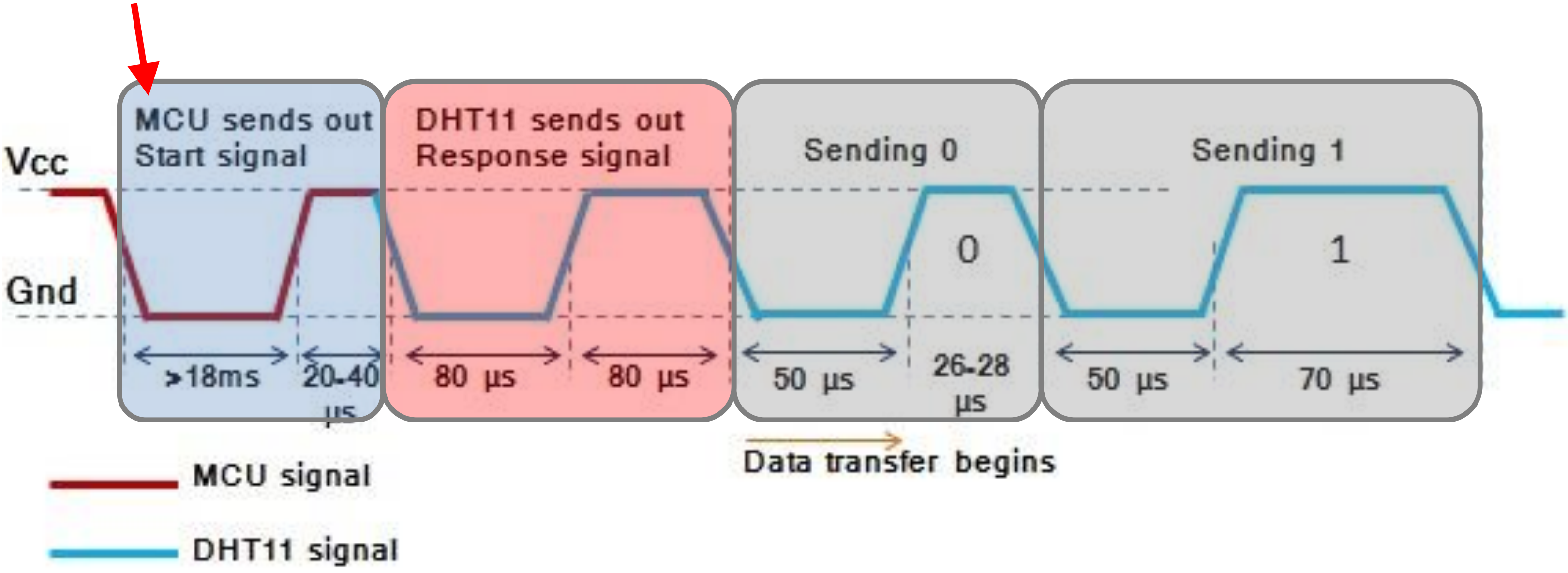


3. When DHT is sending data to MCU

- every bit of data begins with the 50us low-voltage-level
- **the length of the following high-voltage-level signal determines whether data bit is "0" or "1"**

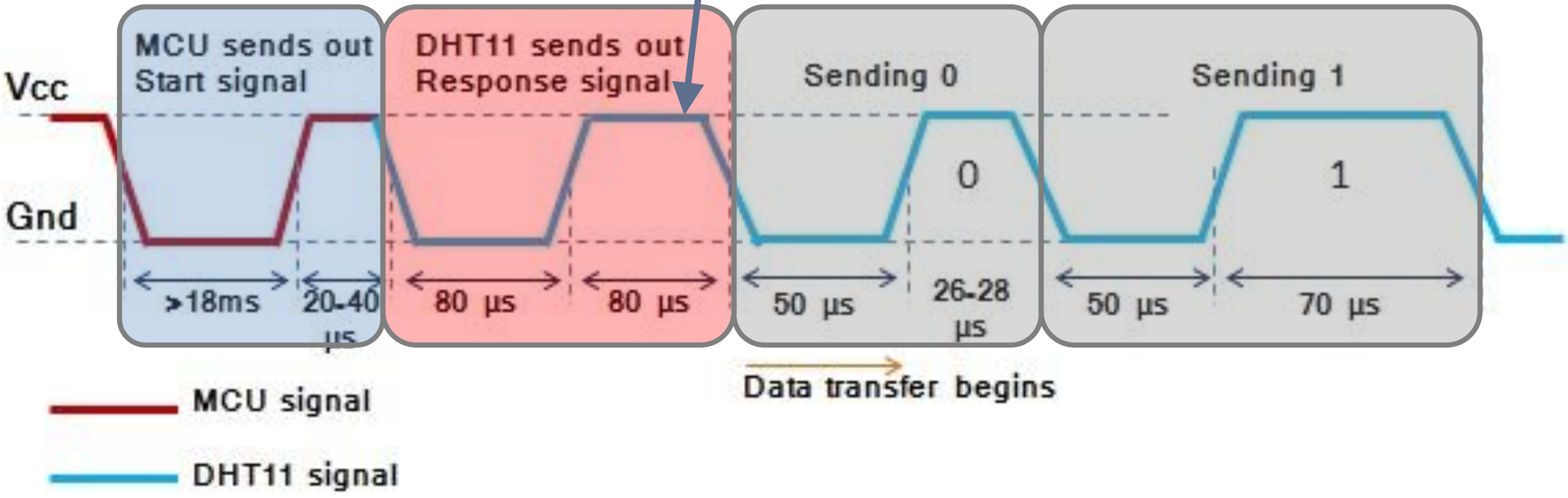
```
RPi.GPIO.setup(self.__pin, RPi.GPIO.OUT)

# send initial high
self.__send_and_sleep(RPi.GPIO.HIGH, 0.05)

# pull down to low
self.__send_and_sleep(RPi.GPIO.LOW, 0.02)
```

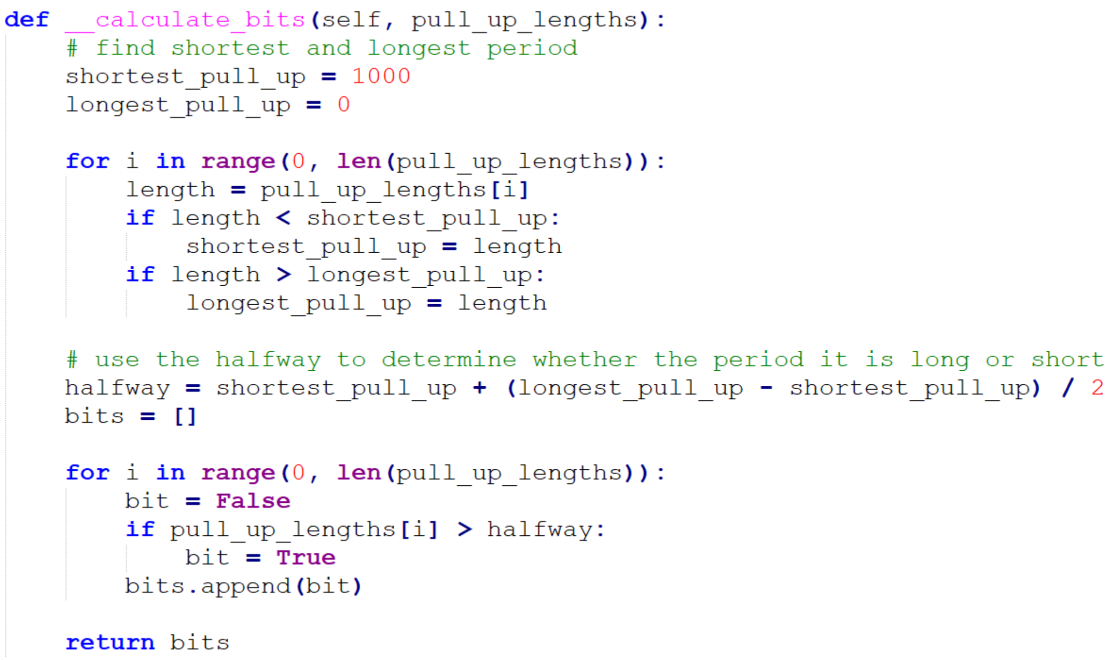
1. Send “Start signal”



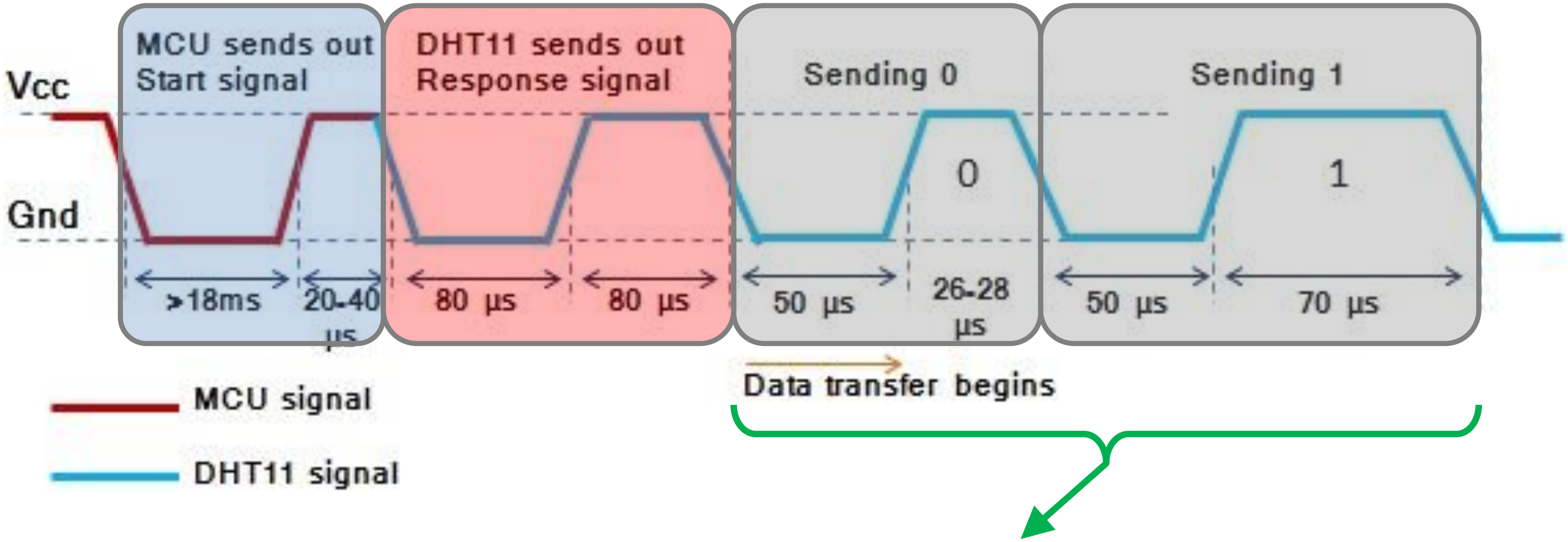
```
# change to input using pull up
RPi.GPIO.setup(self.__pin, RPi.GPIO.IN, RPi.GPIO.PUD_UP)

# collect data into an array
data = self.__collect_input()
```

2. pull up voltage and keeps it for 80us and prepares for data transmission





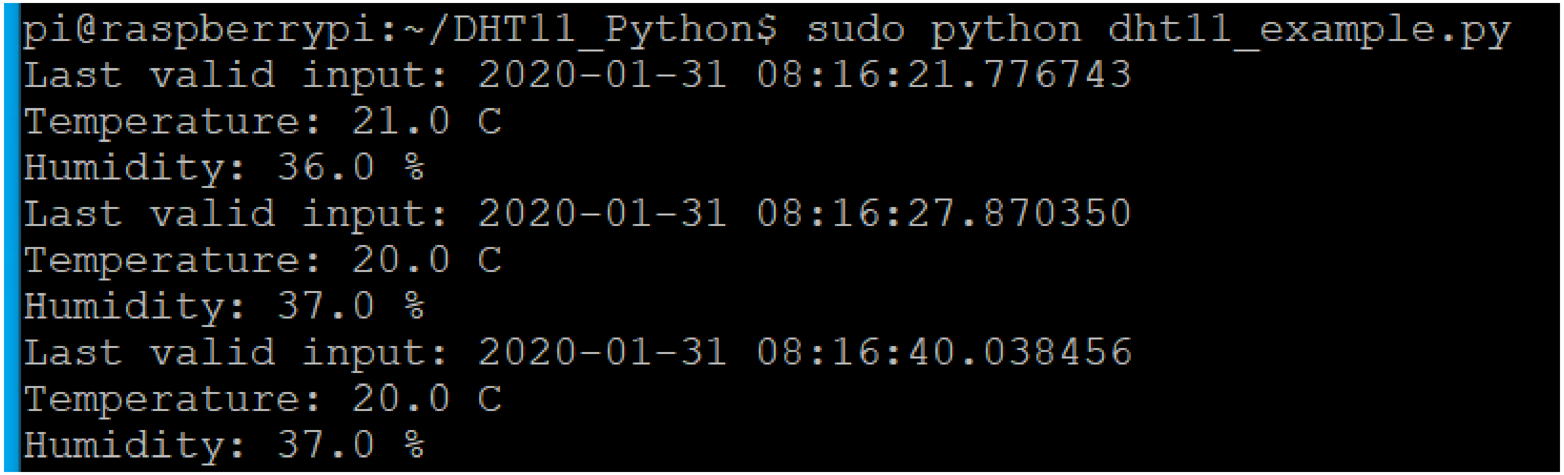
```python
def __calculate_bits(self, pull_up_lengths):
    # find shortest and longest period
    shortest_pull_up = 1000
    longest_pull_up = 0

    for i in range(0, len(pull_up_lengths)):
        length = pull_up_lengths[i]
        if length < shortest_pull_up:
            shortest_pull_up = length
        if length > longest_pull_up:
            longest_pull_up = length

    # use the halfway to determine whether the period it is long or short
    halfway = shortest_pull_up + (longest_pull_up - shortest_pull_up) / 2
    bits = []

    for i in range(0, len(pull_up_lengths)):
        bit = False
        if pull_up_lengths[i] > halfway:
            bit = True
        bits.append(bit)

    return bits
```



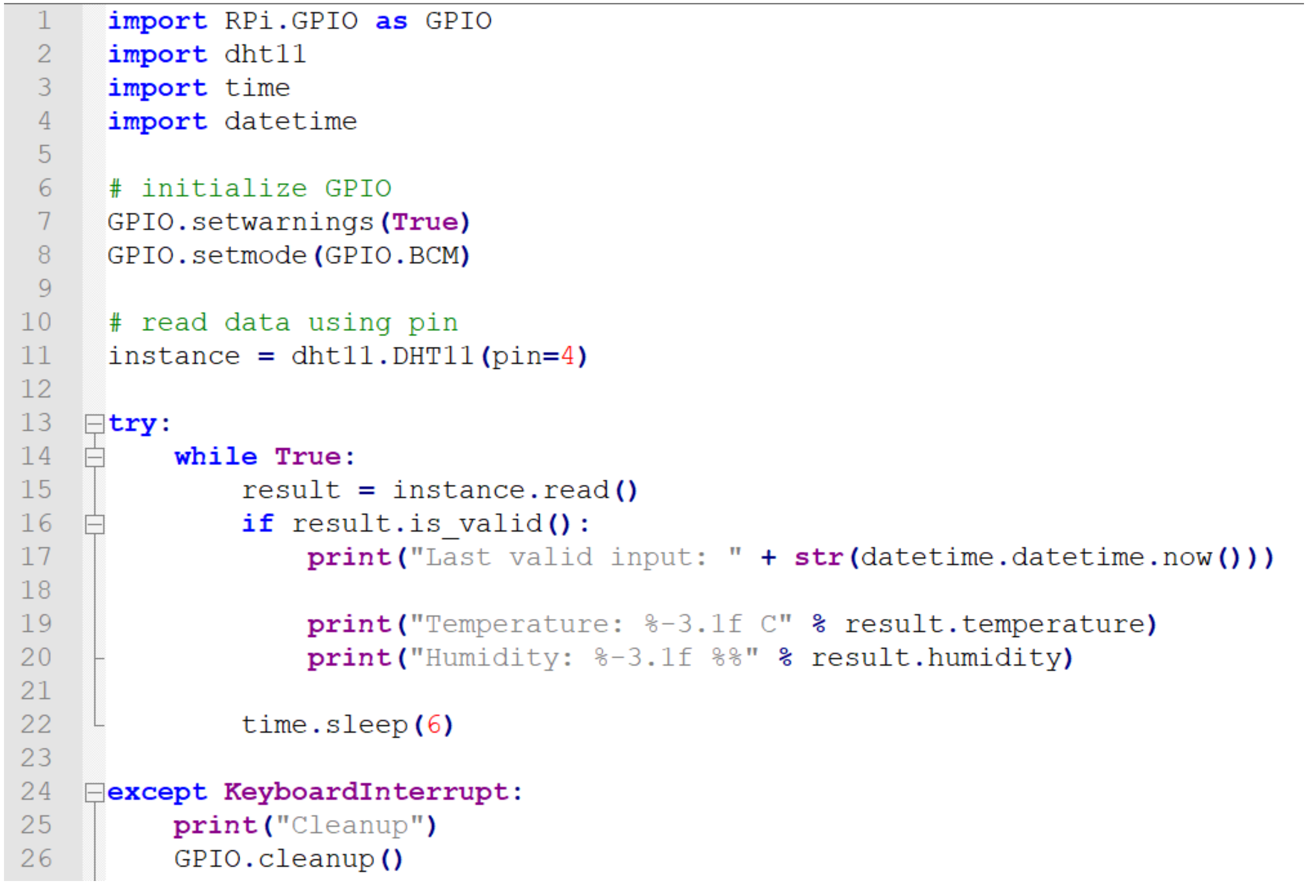
- **the length of the following high-voltage-level signal determines whether data bit is "0" or "1"**

3. DHT-11 sample code

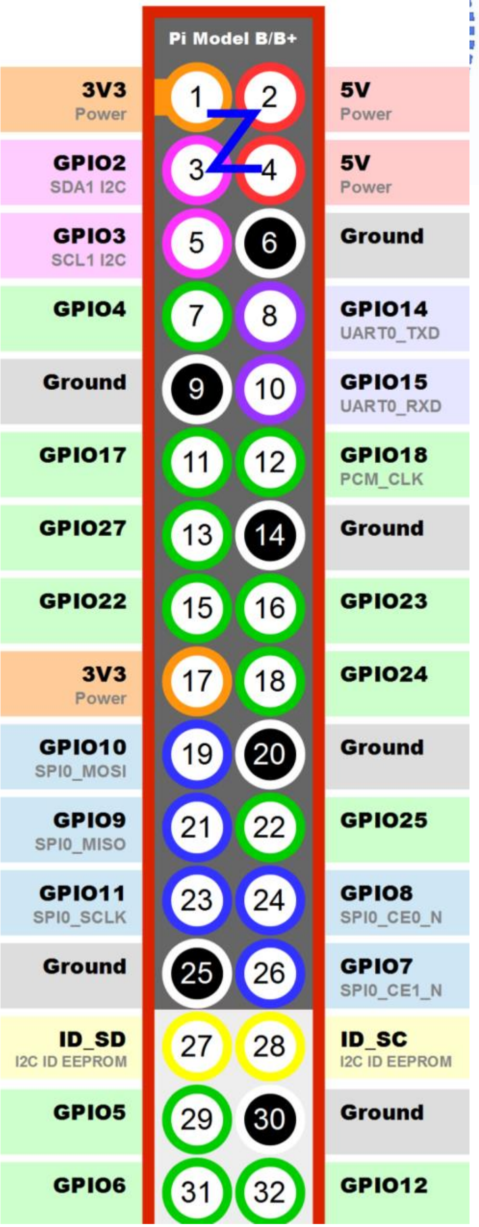
- Download sample code, then unzip it
- sudo python dht11_example.py



```
pi@raspberrypi:~/DHT11_Python$ sudo python dht11_example.py
Last valid input: 2020-01-31 08:16:21.776743
Temperature: 21.0 C
Humidity: 36.0 %
Last valid input: 2020-01-31 08:16:27.870350
Temperature: 20.0 C
Humidity: 37.0 %
Last valid input: 2020-01-31 08:16:40.038456
Temperature: 20.0 C
Humidity: 37.0 %
```



```python
import RPi.GPIO as GPIO
import dht11
import time
import datetime

# initialize GPIO
GPIO.setwarnings(True)
GPIO.setmode(GPIO.BCM)

# read data using pin
instance = dht11.DHT11(pin=4)

try:
    while True:
        result = instance.read()
        if result.is_valid():
            print("Last valid input: " + str(datetime.datetime.now()))

            print("Temperature: %-3.1f C" % result.temperature)
            print("Humidity: %-3.1f %%" % result.humidity)

        time.sleep(6)

except KeyboardInterrupt:
    print("Cleanup")
    GPIO.cleanup()
```

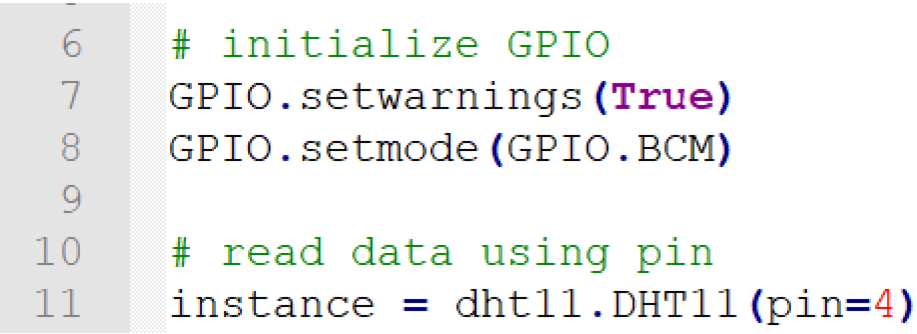


Pi Model B/B+

3V3 Power	1	2	5V Power
GPIO2 SDA1 I2C	3	4	5V Power
GPIO3 SCL1 I2C	5	6	Ground
GPIO4	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SPI0_MOSI	19	20	Ground
GPIO9 SPI0_MISO	21	22	GPIO25
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CE0_N
Ground	25	26	GPIO7 SPI0_CE1_N
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12

Discussion 2

□ In sample code:

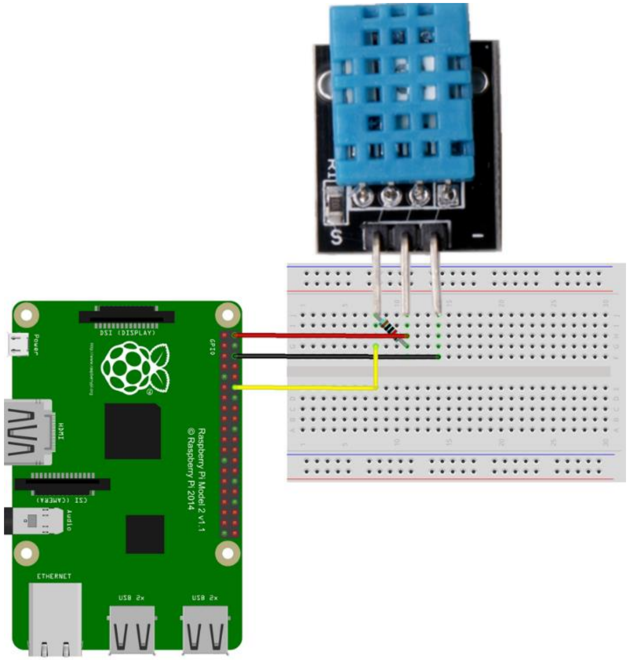


```
# initialize GPIO
GPIO.setwarnings(True)
GPIO.setmode(GPIO.BCM)

# read data using pin
instance = dht11.DHT11(pin=4)
```

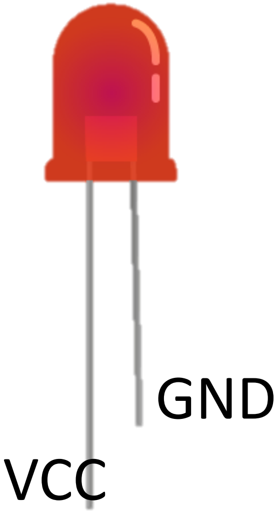
□ **If we want to use Physical PIN number, how to modify the code?**

Quiz 1

- Temperature alarm (溫度警示燈)
  - When the temperature exceeds the threshold (ex: 26.0*C), turn on the LED.

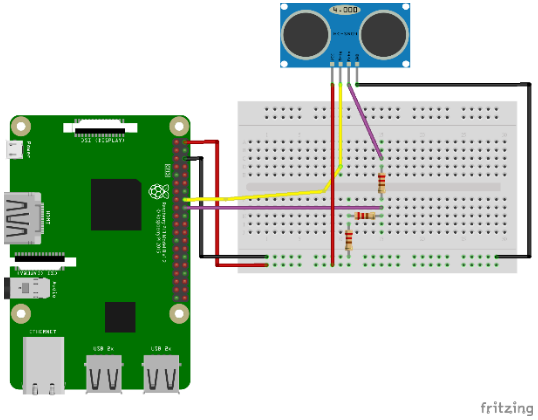


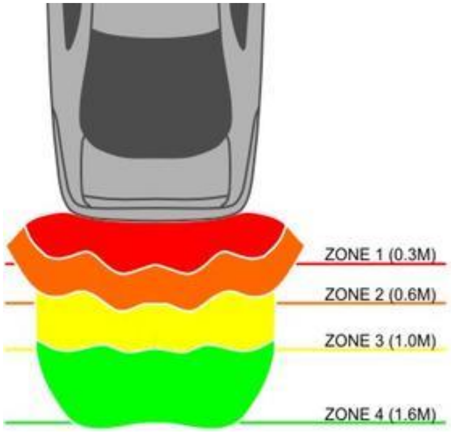
(此電路的電阻為上拉電阻, 可以忽略.
通常已經放在sensor電路板上)

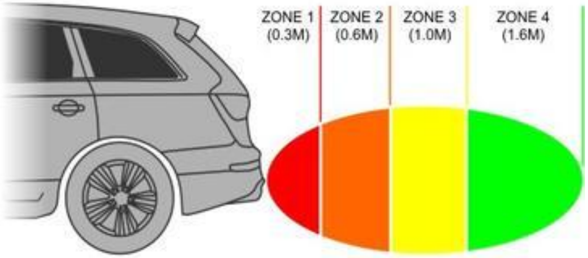


Quiz 2

- Design a Parking Assist System (倒車雷達)
  - use ultrasonic and temperature sensor to measure
  - Divide the detecting distance into three parts:
    a) Safe; b) Be careful; and c) Dangerous.
  - Use the blinking LED to reminder the driver.
    - Safe: no response (> 1m)
    - Be careful: blinking (0.3 to 1m)
    - Dangerous: fast blinking (<0.3 m)







Summary

- Practice Lab (ultrasonic, DHT11)
- Write down the answer for discussion
  - Discussion 1: Ultrasonic application
  - Discussion 2: How to assign Physical PIN number
    - Deadline: Before 3/19, 11:59

- Write code for Quiz 1 - 2, then demonstrate it to TAs
  - Quiz1: Temperature alarm
  - Quiz2: Design a Parking Assist System
    - Deadline: Before 3/12, 15:10
    - Late Demo: Before 3/19, 15:10