

การฝึกอบรมหลักสูตร **MicroPython**

ดร. วาณิส ลีลาภัทร

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยขอนแก่น



แนะนำไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์เป็นคอมพิวเตอร์ขนาดเล็กบรรจุอยู่ในแผงวงจรรวม (IC) ประกอบด้วย

- หน่วยประมวลผล (CPU)
- หน่วยความจำ (RAM / ROM / Flash)
- ช่องสัญญาณข้อมูลเข้า-ออก (I/O ports)
- วงจรประกอบอื่นๆ เช่น USB, I2C, SPI, ADC

ไมโครคอนโทรลเลอร์มีหลายผู้ผลิต

- Atmel: AVR
- ARM
- Tensilica (ESP8266, ESP32)

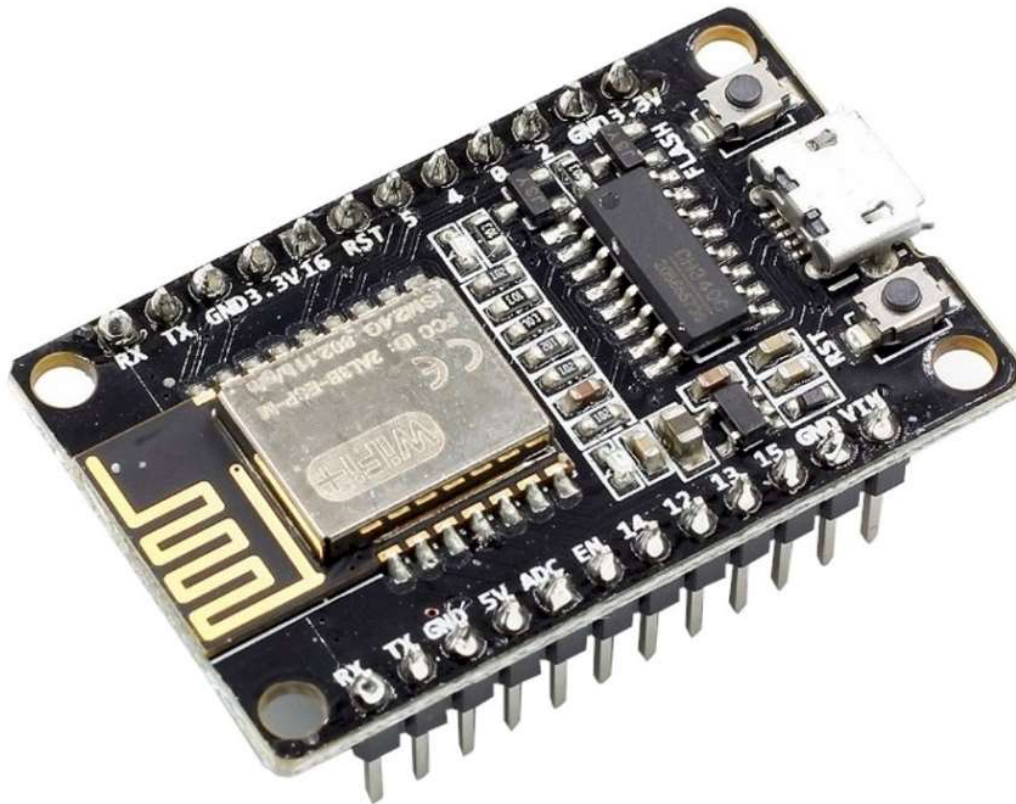
มีการนำมาใช้งานที่หลากหลาย อาทิ:

- เครื่องใช้ไฟฟ้าต่างๆ
- ในรถยนต์
- หุ่นยนต์และระบบควบคุมในโรงงาน

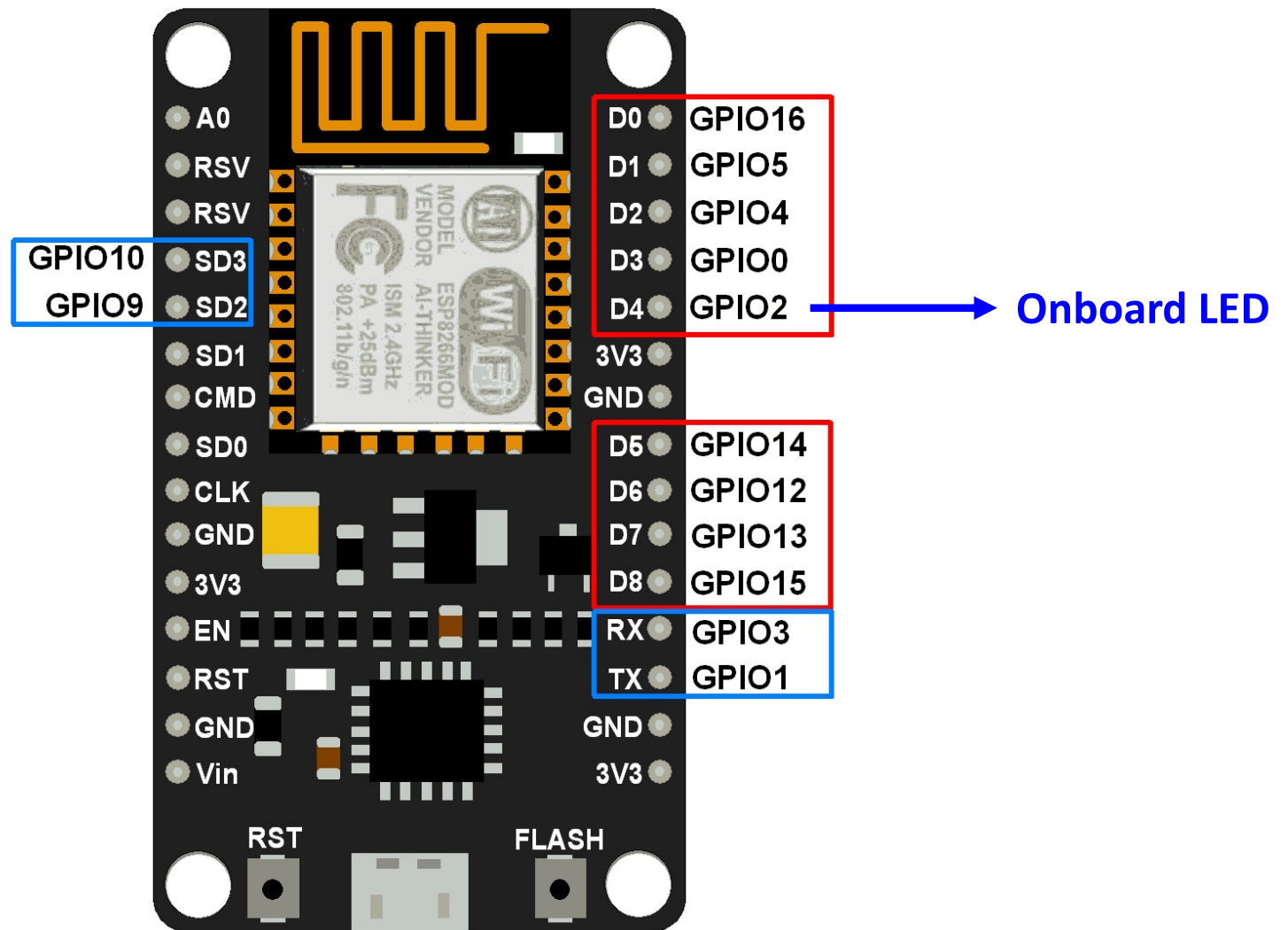


ESP8266 (NodeMCU)

- เป็นไมโครคอนโทรลเลอร์สำหรับการพัฒนาระบบ IoT
- เขียนโปรแกรมโดยใช้ภาษา C/C++ หรือ Python
- MicroPython ถูกพัฒนาสำหรับไมโครคอนโทรลเลอร์



การจัดขาของ NodeMCU



การจัดขาของ NodeMCU

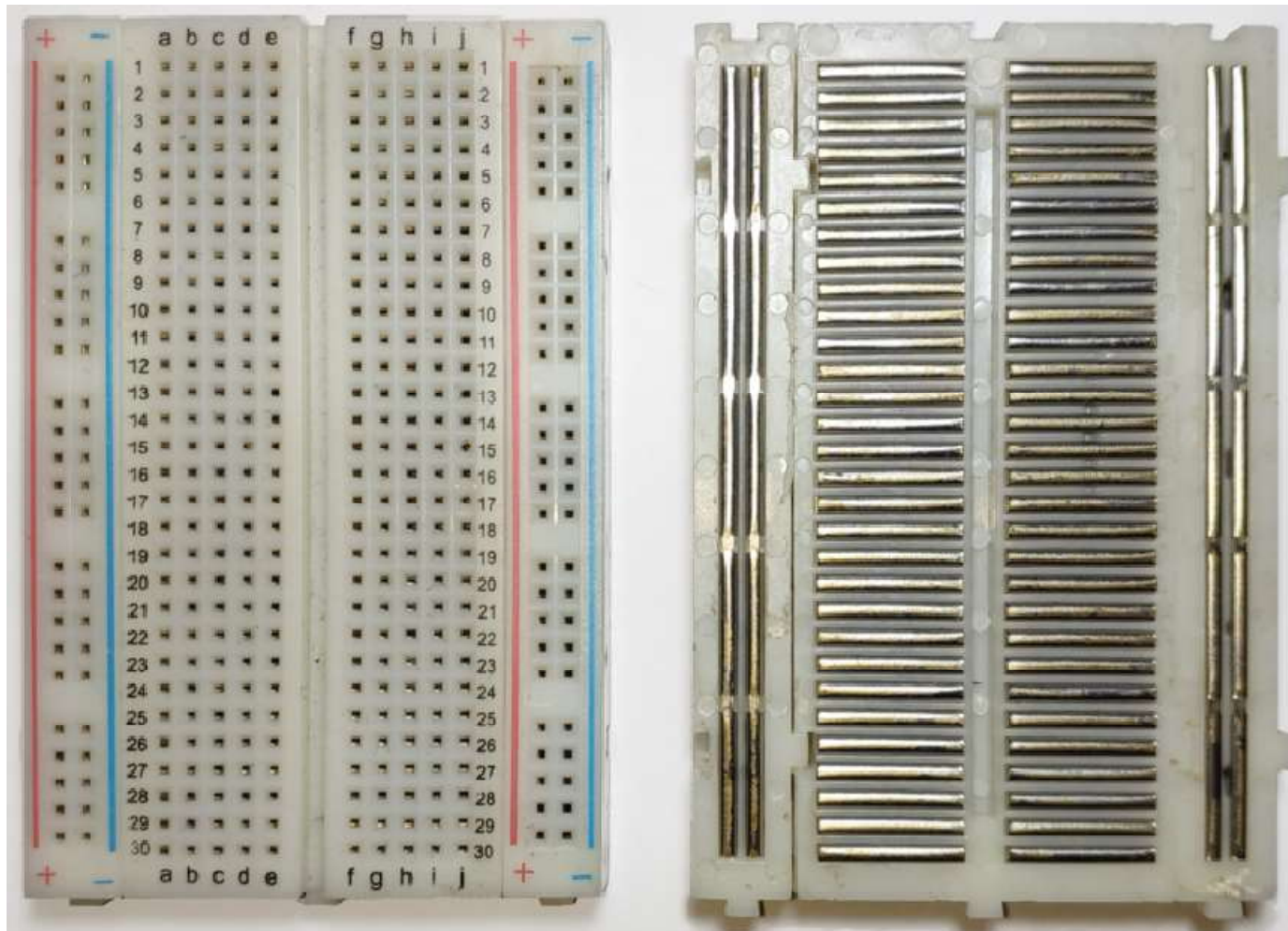
	Label	GPIO	Input	Output	Notes
✗	D0	GPIO16	no interrupt	no PWM or I2C support	HIGH at boot used to wake up from deep sleep
✓	D1	GPIO5	OK	OK	often used as SCL (I2C)
✓	D2	GPIO4	OK	OK	often used as SDA (I2C)
✗	D3	GPIO0	pulled up	OK	connected to FLASH button, boot fails if pulled LOW
★	D4	GPIO2	pulled up	OK	HIGH at boot connected to on-board LED, boot fails if pulled LOW
✓	D5	GPIO14	OK	OK	SPI (SCLK)
✓	D6	GPIO12	OK	OK	SPI (MISO)
✓	D7	GPIO13	OK	OK	SPI (MOSI)
✗	D8	GPIO15	pulled to GND	OK	SPI (CS) Boot fails if pulled HIGH
✗	RX	GPIO3	OK	RX pin	HIGH at boot
✗	TX	GPIO1	TX pin	OK	HIGH at boot debug output at boot, boot fails if pulled LOW
★	A0	ADC0	Analog Input	X	

คำอธิบายขาต่อสำหรับใช้งาน NodeMCU

- 5V & 3.3V เป็นขาที่จ่ายแรงดัน 5V และ 3.3V ใช้สำหรับจ่ายไฟให้กับวงจรภายนอก
- GND เป็นขาแรงดันอ้างอิง (ระดับแรงดัน 0V)
- Reset ขารีเซ็ตการทำงานของบอร์ด
- Vin เป็นขาที่จ่ายแรงดันไฟเลี้ยงเข้ามาให้บอร์ด
- RXD และ TXD เป็นขารับและส่งข้อมูลระหว่างบอร์ดและคอมพิวเตอร์
- A0 เป็นขารับสัญญาณแอนะล็อกจากภายนอก
- D0, D1, ..., D8 เป็นขารับส่งสัญญาณดิจิทัล

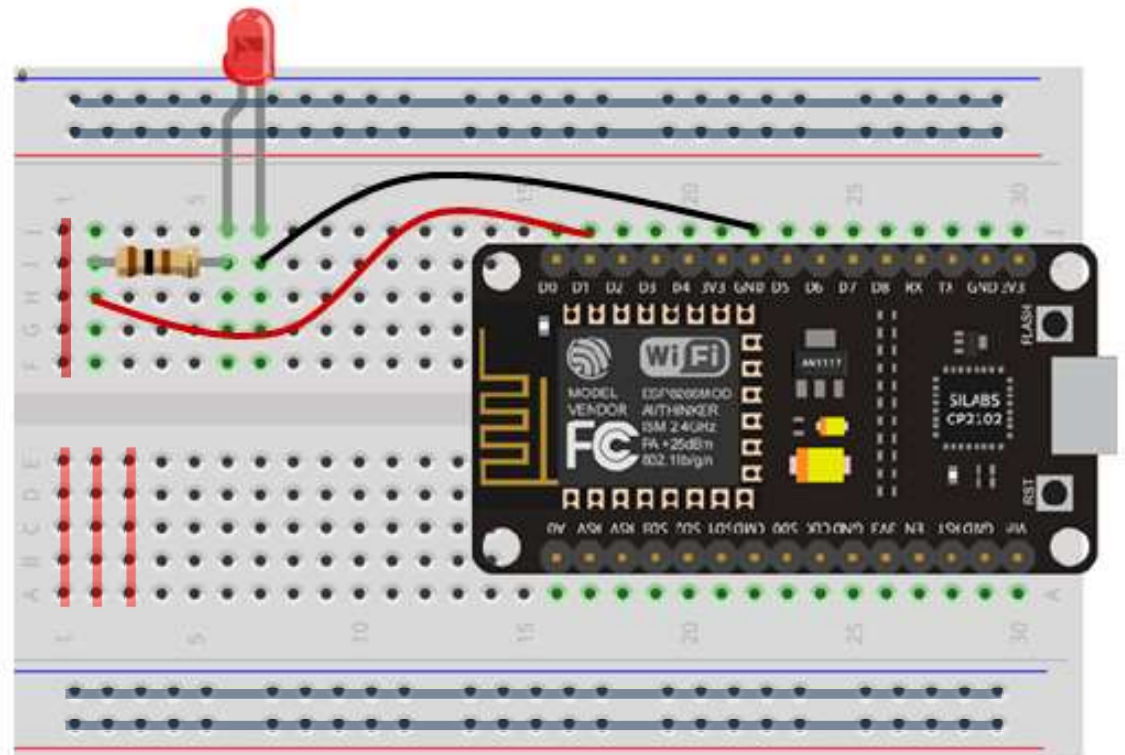
การต่อวงจรเพื่อใช้งาน NodeMCU

- ใช้แผ่น bread board เพื่อการประกอบวงจรที่สะดวก รวดเร็ว
- ด้านในจะมีแถบโลหะตัวนำเชื่อมต่อจุดตามแนวยาวและตามขวางดังภาพ
- ขอบด้านนอก (สีแดง, น้ำเงิน) เชื่อมตามแนวยาว ด้านในจะเชื่อมต่อตามแนวขวาง



การต่อวงจรเพื่อใช้งาน Node MCU

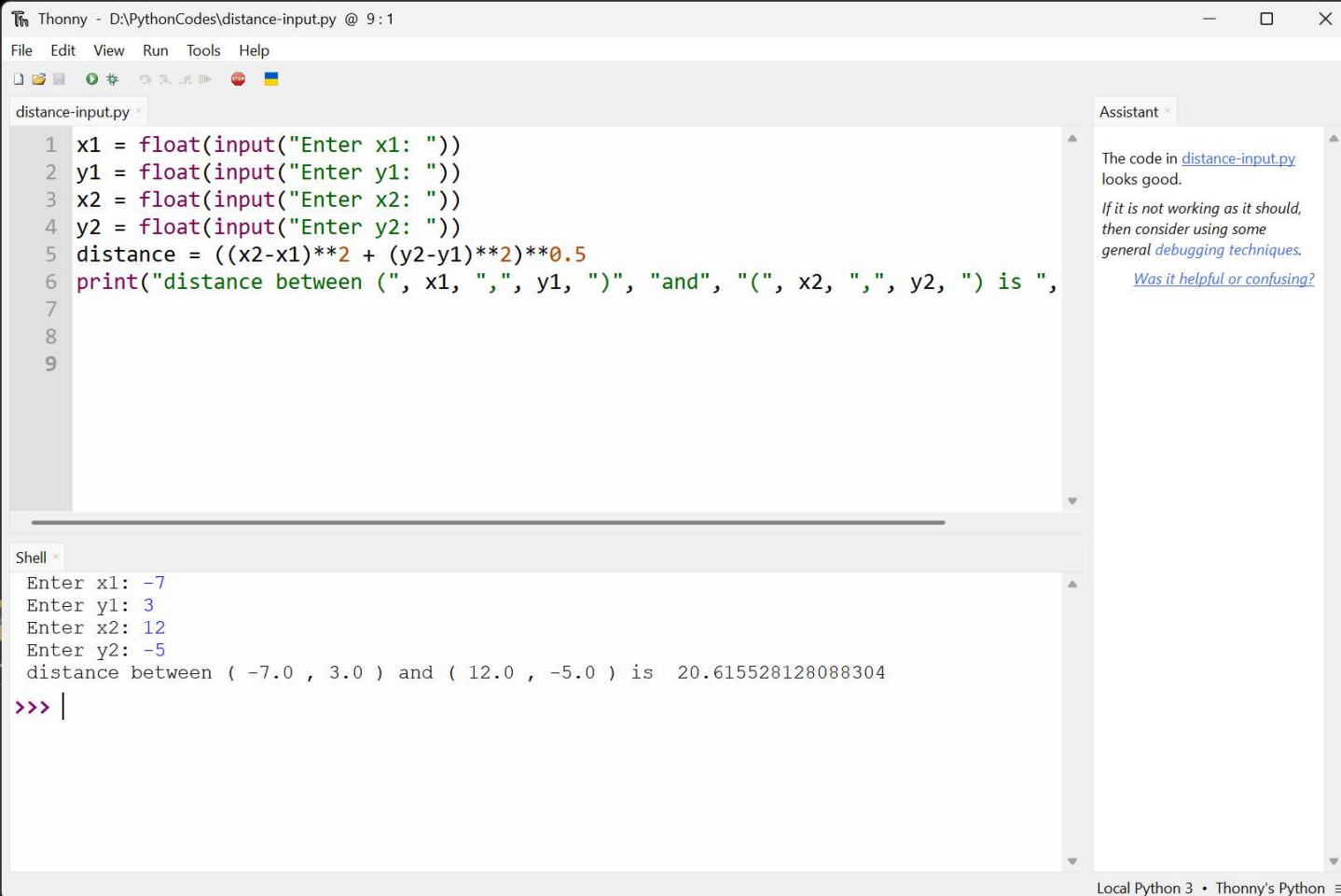
- ต่อวงจรบน Prototyping บอร์ด
- ภายใน prototyping บอร์ดจะมีตัวนำเชื่อมจุดต่างๆ ไว้ให้แล้ว
- ด้านบนและล่างอย่างละ 2 แถว เชื่อมต่อตามแนวนอน
- ตรงกลางเชื่อมต่อตามแนวตั้ง



ซอฟต์แวร์สำหรับพัฒนา

Thonny IDE

- ใช้สำหรับการเขียนและ Download โปรแกรมภาษา Python ลง Node MCU
- ดาวน์โหลดได้จาก www.thonny.org



The screenshot displays the Thonny IDE window. The title bar reads "Thonny - D:\PythonCodes\distance-input.py @ 9:1". The menu bar includes "File", "Edit", "View", "Run", "Tools", and "Help". The toolbar contains icons for file operations and execution. The main editor shows a Python script named "distance-input.py" with the following code:

```
1 x1 = float(input("Enter x1: "))
2 y1 = float(input("Enter y1: "))
3 x2 = float(input("Enter x2: "))
4 y2 = float(input("Enter y2: "))
5 distance = ((x2-x1)**2 + (y2-y1)**2)**0.5
6 print("distance between (", x1, ",", y1, ")", "and", "(", x2, ",", y2, ") is ",
7
8
9
```

Below the editor is the "Shell" window, which shows the execution output:

```
Enter x1: -7
Enter y1: 3
Enter x2: 12
Enter y2: -5
distance between ( -7.0 , 3.0 ) and ( 12.0 , -5.0 ) is 20.615528128088304
>>> |
```

On the right side of the IDE, there is an "Assistant" panel with the following text:

The code in [distance-input.py](#) looks good.

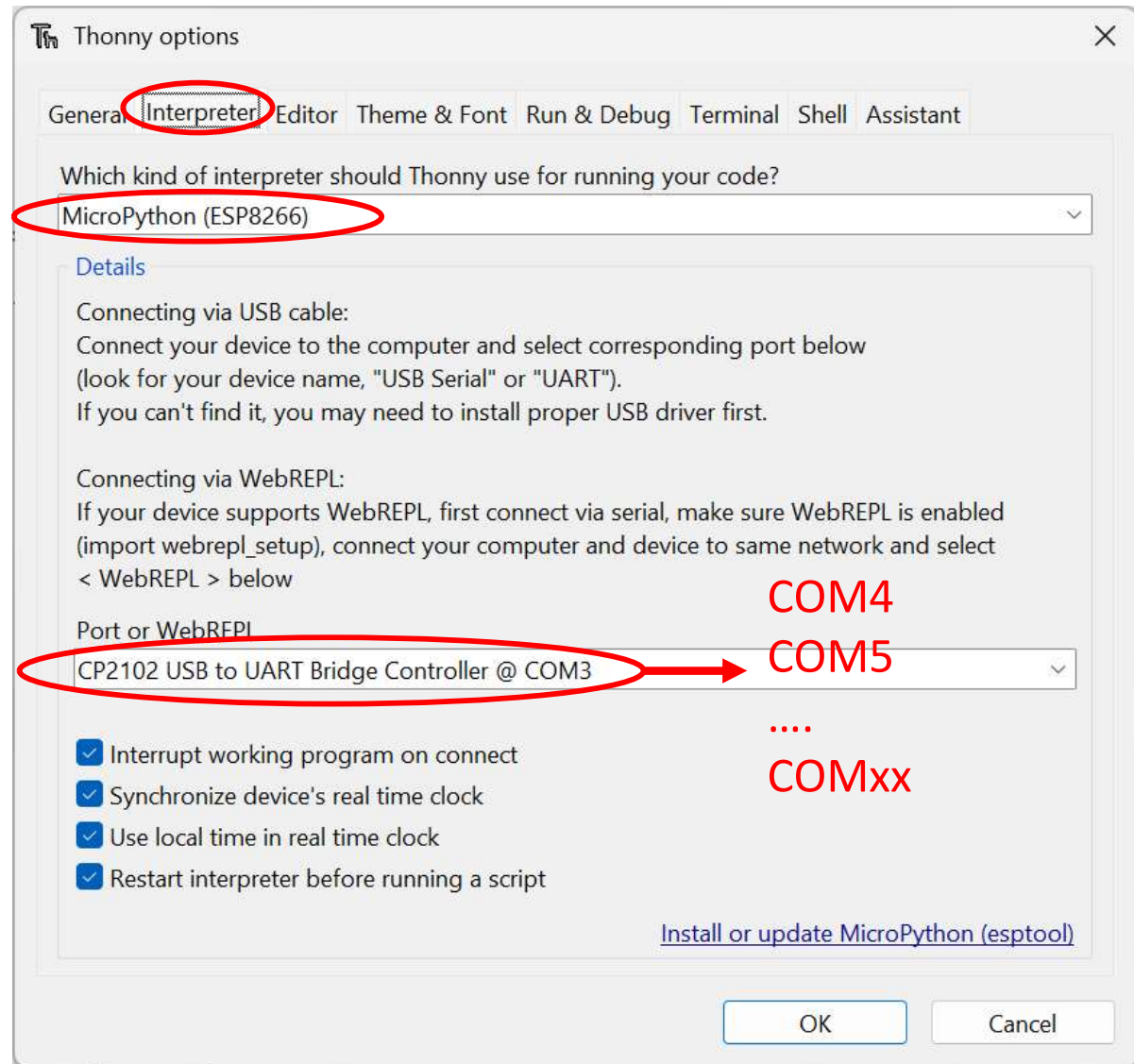
If it is not working as it should, then consider using some [general debugging techniques](#).

[Was it helpful or confusing?](#)

The status bar at the bottom right indicates "Local Python 3 • Thonny's Python".

การใช้ Thonny ร่วมกับ Node MCU

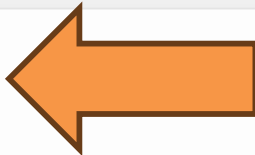
- เปิดโปรแกรม Thonny IDE ไปที่เมนู Run → Config Interpreter...
- เลือก Tab Interpreter



การใช้ Thonny ร่วมกับ Node MCU

เมื่อ Thonny IDE เชื่อมต่อกับ Node MCU ได้ จะมีข้อความปรากฏใน console

```
Shell ×
o0 b b^ ^ ^ ^ ^ ^
MicroPython v1.24.1 on 2024-11-29; ESP module with ESP8266
Type "help()" for more information.
>>> |
```



MicroPython (ESP8266) • CP2102 USB to UART Bridge Controller @ COM3

ตัวอย่างการใช้งานตัวแปลภาษาไพธอน

- เราสามารถใช้ตัวแปลภาษาไพธอน เพื่อการคำนวณได้ โดย
- เปิดโปรแกรม Python จะเห็นหน้าจอแสดงผล (Console)
- ทดลองพิมพ์คำสั่งดังต่อไปนี้

>>> 22/7 $\frac{22}{7}$

3.142857142857143

>>> 2+2-2*2/2

2.0

>>> 8**(1/3) $\sqrt[3]{8}$

2.0

>>> _**(2/3) $\sqrt[3]{2^2}$

1.5874010519681994

เครื่องหมาย _ ใช้แทนค่าผลลัพธ์ล่าสุด

ตัวอย่างการใช้งานตัวแปลภาษาไพธอน

- ทดลองพิมพ์คำสั่งดังต่อไปนี้ใน Console

```
>>> 7/3
```

```
2.333333333333
```

```
>>> 7//3
```

```
2
```

```
>>> 7%3
```

```
1
```

$22 = ?$**

$-22 = ?$**

$(-2)2 = ?$**

$6/2*(2+1) = ?$

ตัวกระทำทางคณิตศาสตร์

- ตัวกระทำทางคณิตศาสตร์ในภาษาไพธอนมีลำดับความสำคัญ
- ความสำคัญมากจะคำนวณก่อน (ถ้าเท่ากัน ทำจากซ้ายไปขวา)

เครื่องหมาย	ตัวกระทำ	ลำดับความสำคัญ
()	วงเล็บ	1
**	ยกกำลัง	2
-	ติดลบ	3
*	คูณ	4
/	หาร	4
//	หาร(ผลลัพธ์จำนวนเต็ม)	4
%	หาร(เศษจากการหาร)	4
+	บวก	5
-	ลบ	5

ตัวแปรในภาษาไพธอน

- ตัวแปร(Variable) คือชื่อที่ใช้เป็นที่เก็บข้อมูลในโปรแกรม
- ข้อมูลที่เก็บอาจเป็น ตัวเลข หรือ ตัวอักษร ก็ได้
- ชื่อตัวแปรจะต้องตั้งโดยใช้ตัวอักษรนำหน้าเท่านั้น เช่น
- **A, b, X, y, Z, r**
- **Day, Month, Year**
- **x1, x2, Person23**
- ต้องไม่มีเครื่องหมายใดๆ ยกเว้น _ เช่น day_of_week, speed_limit เพื่อให้ตัวแปรที่มีชื่อยาว สามารถอ่านได้สะดวกขึ้น
- **ชื่อตัวแปรจะต้องไม่ซ้ำกับคำสั่งในภาษาไพธอน**

ตัวแปรในภาษาไพธอน

- ในชื่อตัวแปรจะถือว่าตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก แตกต่างกัน
- ดังนั้น x และ X จะเป็นตัวแปรคนละตัว
- Area และ area จะเป็นตัวแปรคนละตัว
- คำที่ใช้เป็นชื่อตัวแปรไม่ได้

and, assert, break, class, continue,
def, del, elif, else, except, exec,
finally, for, from, global, if,
import, in, is, lambda, not, or,
pass, print, raise, return, try, while

ตัวแปรในภาษาไพธอน

- การกำหนดค่าข้อมูลให้กับตัวแปรจะใช้เครื่องหมาย =
- มีรูปแบบคือ **ตัวแปร** = **ค่าข้อมูล**
- ตัวแปรจะต้องอยู่ด้านซ้ายของเครื่องหมาย = เท่านั้น ตัวอย่างเช่น
- $x = 2$
- $y = 7 / 3$
- $z = 7 // 3$
- $\text{Area} = (22/7)*r**2$
- $\text{Province} = \text{"KhonKaen"}$

ตัวแปรในภาษาไพธอน

- การกำหนดค่าให้กับตัวแปรพร้อมกันหลายตัวสามารถทำได้
- มีรูปแบบคือ ตัวแปร, ตัวแปร, ... = ค่าข้อมูล, ค่าข้อมูล, ...
- `a, b, c = 1, 2, 3`
- `Name, x = "John", 2/3`
- ทดลองในหน้าจอแสดงผลของไพธอน

```
>>> x = 2
```

```
>>> x
```

```
2
```

```
>>> a, b, c = 5, 6, 7
```

```
>>> c
```

```
7
```


การเขียนข้อความอธิบาย (comment)

- ในการเขียนโปรแกรม ควรมี comment เพื่อให้ทำความเข้าใจได้ง่าย
- Comment จะต้องนำด้วยเครื่องหมาย #

#Program for temperature conversion

#ชื่อ-นามสกุล นักเรียน ห้อง

C = 0 # temperature in Celsius

F = 75 # temperature in Fahrenheit

C = (F-32)*(5/9) # conversion formular

ชนิดของข้อมูลในภาษาไพธอน

- ข้อมูลพื้นฐานในภาษาไพธอน ได้แก่
- ตัวเลข (จำนวนเต็ม), ตัวเลข (จำนวนจริง), ตัวอักษร และ ค่าความจริง
- ตัวอย่างของข้อมูลตัวเลข

$x = 5//2$ x จะมีค่า 2

$\text{Area} = 22/7$ Area จะมีค่า 3.142857142857143

- ตัวอย่างของข้อมูลตัวอักษร
- $\text{Month} = \text{"August"}$ จะมีค่า August
- ตัวอย่างของค่าความจริง
- $z = (2 < 1)$ z จะมีค่าเป็น False
- ค่าความจริงจะมีค่าได้เพียง True (จริง) หรือ False (เท็จ) เท่านั้น

ประโยค (Statement)

ประโยค หมายถึง 1 ชุดคำสั่งของภาษาไพธอน มีได้หลายแบบ อาทิ

- กำหนดค่าให้ตัวแปร (assignment statement)
- ตัดสินใจ (decision statement)
- วนซ้ำ (looping statement)

ประโยคกำหนดค่าให้ตัวแปร

ตัวแปร = ค่าที่กำหนด

เช่น

X = 0



C = (5/9)*(F-32)



Message = "Hello!"



5 = Y

x

K/2 = 20

x

i+j = 14

x

ตัวกระทำทางคณิตศาสตร์ (Math Operators)

แบบฝึกหัด เขียนประโยคภาษาไพธอนเพื่อแปลงอุณหภูมิ $C \leftrightarrow F$

$$\frac{C}{5} = \frac{F - 32}{9}$$

ค่าอุณหภูมิ 36°C จะมีค่าเท่ากับกี่ฟาเรนไฮต์

ตัวกระทำทางคณิตศาสตร์ (Math Operators)

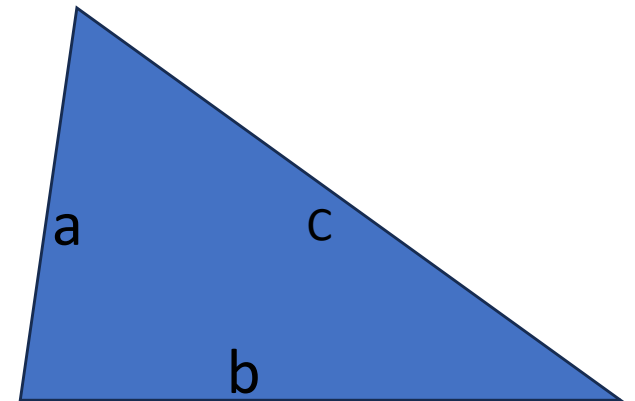
แบบฝึกหัด

เป็นดังนี้

สูตรคำนวณพื้นที่ 3 เหลี่ยม เมื่อทราบความยาวด้านทั้ง 3

$$Area = \sqrt{s(s - a)(s - b)(s - c)}$$

$$s = \frac{a + b + c}{2}$$



จงคำนวณหาพื้นที่ 3 เหลี่ยมที่มีความยาวด้านเป็น 11, 10 และ 12 หน่วยตามลำดับ

การแสดงผล

- ใช้คำสั่ง `print ()` เพื่อแสดงข้อมูลออกทางหน้าจอ
- รูปแบบ

`print(variable)`

`print(variable, variable,...)`

`print("string")`

- ปกติเมื่อแสดงผลแล้วจะขึ้นบรรทัดใหม่ หากไม่ต้องการขึ้นบรรทัดใหม่ ให้ระบุ `end = "` ในวงเล็บ เช่น

`print("Hello", end = "")`

Source Code: Ch3-Print_demo.py

การรับข้อมูลจากแป้นพิมพ์

- ใช้คำสั่ง `input()` เพื่อรับข้อมูลจากแป้นพิมพ์
- ข้อมูลจะได้เป็นตัวอักษรเท่านั้น
- หากต้องการตัวเลข ต้องแปลงจากตัวอักษรให้เป็นตัวเลข
- รูปแบบ

`variable = input(message)`

ตัวอย่าง

```
Name = input("Enter your name")  
Number = int(input("Enter your age"))  
Weight = float(input("Enter your weight"))
```

Source Code: Ch3-Input_demo.py

การตัดสินใจ

- ประโยคตัดสินใจ **if...**
- ใช้สำหรับการเลือกทำคำสั่งเมื่อเงื่อนไขที่กำหนดเป็นจริง
- รูปแบบ

if เงื่อนไข :

statement1 #เงื่อนไขเป็นจริง

statement2 #เงื่อนไขเป็นจริง

statement3 #เงื่อนไขเป็นจริง

next statement #เงื่อนไขเป็นเท็จ

- เงื่อนไขมีค่าเป็น จริง(TRUE) หรือ เท็จ (FALSE)

Source Code: Ch3-If_demo1.py

เงื่อนไขและการเปรียบเทียบ

- ตัวอย่างเงื่อนไข

$$X = 5$$

$$Y = 3$$

$$X < Y \quad \rightarrow \text{เท็จ}$$

$$X > Y \quad \rightarrow \text{จริง}$$

$$X >= Y \quad \rightarrow \text{จริง}$$

$$X == Y \quad \rightarrow \text{เท็จ}$$

$$-1 < Y \quad \rightarrow \text{จริง}$$

$$X < 7 \quad \rightarrow \text{จริง}$$

เงื่อนไขและการเปรียบเทียบ

- เครื่องหมายเปรียบเทียบ

== เท่ากับ

< น้อยกว่า

> มากกว่า

<= น้อยกว่าหรือเท่ากับ

>= มากกว่าหรือเท่ากับ

!= ไม่เท่ากับ

Source Code: Ch3-If_demo2.py

เงื่อนไขและการเปรียบเทียบ

- เงื่อนไขอาจเชื่อมกันได้โดยใช้

AND	และ
OR	หรือ
NOT	นิเสธ

a, b, c = 1, 2, 3

x, y, z = 3, 4, 5

(a+b == c) and (b**2 <= z) #TRUE

(a*b+c != z) or (a*b*c >= a+b+c) #TRUE

not (z != a+b-c) #FALSE

Source Code: Ch3-If_demo3.py

การตัดสินใจ

ตัวอย่าง

```
#compare two numbers
x = int(input("Enter x : "))
y = int(input("Enter y : "))
if x == y :
    print("x is equal to y")
if x < y :
    print("x is less than y")
if x > y :
    print("x is greater than y")
```

Source Code: Ch3-If_demo4.py

การตัดสินใจ

- ประโยคตัดสินใจ **if...else....**
- ใช้สำหรับการเลือกทำคำสั่งชุดที่ 1 เมื่อเงื่อนไขเป็นจริง และทำชุดที่ 2 เมื่อเงื่อนไขเป็นเท็จ
- รูปแบบ

if เงื่อนไข :

statement1 #เงื่อนไขเป็นจริง

statement2 #เงื่อนไขเป็นจริง

else :

statement3 #เงื่อนไขเป็นเท็จ

statement4 #เงื่อนไขเป็นเท็จ

statement outside if

Source Code: Ch3-If_demo5.py

การตัดสินใจ

- ประโยคตัดสินใจ **if...elif....else....**
- ใช้สำหรับกรณีมีหลายเงื่อนไข และต้องเลือกทำคำสั่งแต่ละชุดตามเงื่อนไข รูปแบบ

if เงื่อนไข1 :

statement1 #เงื่อนไข 1 เป็นจริง

elif เงื่อนไข2

statement2 #เงื่อนไข 2 เป็นจริง

elif เงื่อนไข3

statement3 #เงื่อนไข 3 เป็นจริง

....

else

statement_n #เงื่อนไขอื่นเป็นเท็จทั้งหมด

statement outside if

Source Code: Ch3-If_demo6.py

การตัดสินใจ


```
#grading program
score = float(input("Enter your score : "))
if score > 100 or score < 0 :
    print("score must be 0-100")
elif score < 100 and score >= 80 :
    print("you get A ")
elif score < 80 and score >= 70 :
    print("you get B ")
elif score < 70 and score >= 60 :
    print("you get C ")
elif score < 60 and score >= 50 :
    print("you get D ")
else :
    print("you get F ")
```

Source Code: Ch3-Grade.py

การวนซ้ำ (Looping)

- ประโยค วนซ้ำ **for....**
- ใช้สำหรับควบคุมให้กลุ่มประโยคทำงานซ้ำ ตามจำนวนครั้งที่กำหนด

รูปแบบ


for variable in range(number) :
statement1 #ประโยคที่ต้องการวนซ้ำ
statement2 # ประโยคที่ต้องการวนซ้ำ
....
statement outside loop

ตัวอย่าง

```
for k in range(5) :      #k = 0, 1, 2, 3, 4  
    print("Statement inside loop", k)  
print("Statement outside loop")
```

การวนซ้ำ (Looping)

ตัวอย่าง 1

```
for k in range(10) :  
    m = k*2  
    print(k, m)  
print("End of loop")
```

=====

```
0 0  
1 2  
2 4  
3 6  
4 8  
5 10  
6 12  
7 14  
8 16  
9 18  
End of loop
```

ตัวอย่าง 2

```
for x in range(1,5) :  
    print(x, x*3)  
print("End of loop")
```

=====

```
1 3  
2 6  
3 9  
4 12  
End of loop
```

Source Code: Ch3-Loop_demo2.py

การวนซ้ำ (Looping)

- ประโยค วนซ้ำ **while....**
- ใช้สำหรับควบคุมให้กลุ่มประโยคทำงานซ้ำ ตามเงื่อนไขที่กำหนดรูปแบบ

while เงื่อนไข :

statement1 #ประโยคที่ต้องการวนซ้ำ จนกว่าเงื่อนไขเป็นเท็จ

statement2 # ประโยคที่ต้องการวนซ้ำจนกว่าเงื่อนไขเป็นเท็จ

....

statement outside loop

ตัวอย่าง

k = 0

while k < 7 :

print("Statement inside loop")

 k = k+1

print("Statement outside loop")

Source Code: Ch3-Loop_demo3.py

การประยุกต์ใช้การวนซ้ำ (Looping)

ตัวอย่าง 1

โปรแกรมคำนวณผลรวม $1+2+3+....+99+100$

Source Code: Ch3-Loop_Sum1.py

ตัวอย่าง 2

โปรแกรมคำนวณผลรวมของตัวเลขที่รับทางแป้นพิมพ์

Source Code: Ch3-Loop_Sum2.py

ดิจิทัลพอร์ท (Digital Port)


- ดิจิทัลพอร์ท (GPIO) ทำหน้าที่รับหรือส่งสัญญาณดิจิทัล (“0” หรือ “1”) หรือเทียบเป็นแรงดัน 0 โวลต์ และ 3.3 โวลต์

คำอธิบาย

กำหนดหน้าที่ของดิจิทัลพอร์ท โดย input จะเป็นการรับ และ output จะเป็นการส่ง

รูปแบบ

```
from machine import Pin  
pin_object = Pin(GPIO, mode)
```

mode =  Pin.OUT
Pin.IN

Parameters

ชา: ขาสัญญาณของ Node MCU

mode: INPUT, OUTPUT

ตัวอย่าง

```
LED = Pin(2, Pin.OUT)
```


การเขียน (ส่ง) ข้อมูลออกทางดิจิทัลพอร์ท

.value()

คำอธิบาย

เป็นการส่งข้อมูลออกทางดิจิทัลพอร์ทที่กำหนด

รูปแบบ

Pin_object.value(value) value = 0, 1

Parameters

value: 1 (จ่ายแรงดัน 3.3V) หรือ 0 (ปล่อยแรงดัน 0V)

ตัวอย่าง

LED.value(1) #LED ติดสว่าง

LED.value(0) #LED ดับ

ตัวอย่างการใช้งานติจิทัลพอร์ท

โปรแกรมไฟกระพริบโดยใช้ LED บน Node MCU

```
from machine import Pin
from time import sleep
```

```
LED = Pin(2, Pin.OUT) #GPIO = 2 is output pin
```

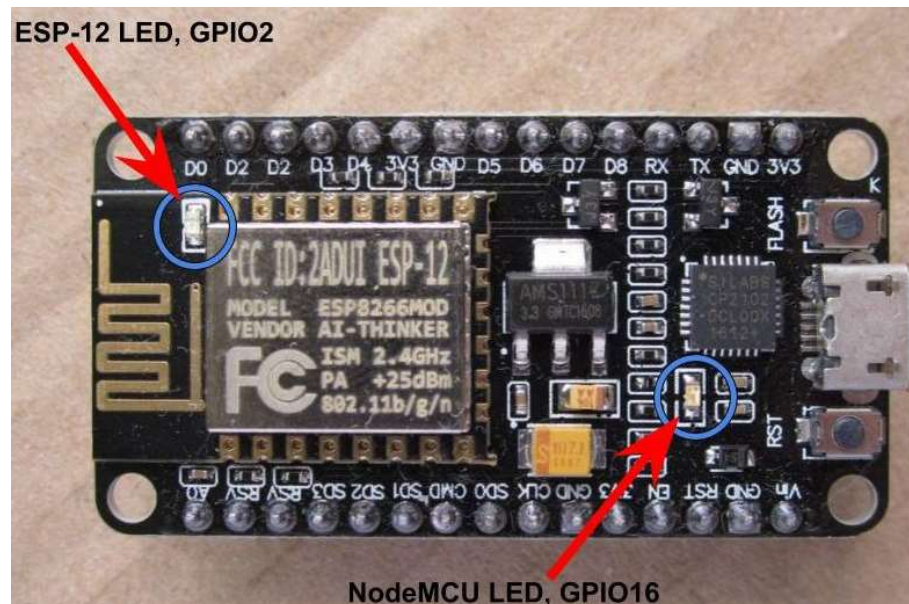
```
while True:
```

```
    LED.value(1)
```

```
    sleep(0.5)
```

```
    LED.value(0)
```

```
    sleep(0.5)
```



แบบฝึกหัด

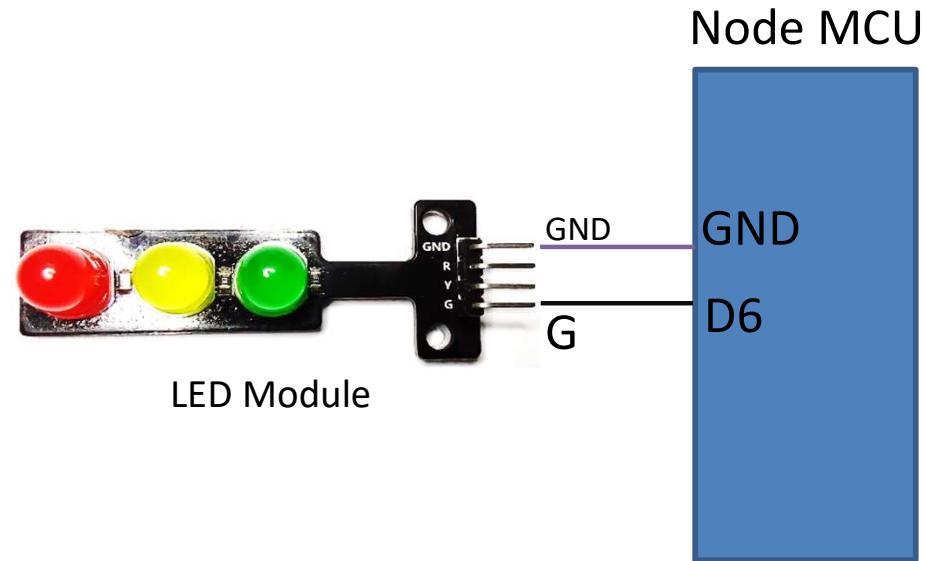
1. เขียนโปรแกรมเพื่อให้หลอดไฟกระพริบ 10 ครั้งแล้วดับ
2. เขียนโปรแกรมเพื่อให้หลอดไฟกระพริบ 20 ครั้งแล้วดับ พร้อมทั้งแสดงจำนวนครั้งที่กระพริบผ่านทาง Console

การต่อ LED ภายนอก

- ป้อนโปรแกรมและต่อวงจรตามภาพด้านล่าง

```
from machine import Pin
from time import sleep
```

```
Blue = Pin(2, Pin.OUT) #blue LED
Green = Pin(12, Pin.OUT) #green LED
while True:
    Blue.value(1)
    Green.value(0)
    sleep(0.5)
    Blue.value(0)
    Green.value(1)
    sleep(0.5)
```



การอ่าน (รับ) ข้อมูลเข้าทางดิจิตอลพอร์ท

.value()

คำอธิบาย

เป็นการรับข้อมูลเข้าทางดิจิตอลพอร์ทที่กำหนด

รูปแบบ

`variable = Pin_object.value()` #ไม่ต้องระบุ parameter ใน ()

ค่าส่งกลับ

`variable = 1` ถ้ามีแรงดัน 3.3V ที่ขาGPIO

`variable = 0` ถ้ามีแรงดัน 0V ที่ขา GPIO

ตัวอย่าง

`switch = sw.value()` #อ่านค่าสถานะสวิตช์

การต่อสวิตช์

- ต่อวงจรและป้อนโปรแกรมตามรูปด้านล่างนี้

```
from machine import Pin
from time import sleep
```

```
K1 = Pin(14, Pin.IN) #Switch is at D5
```

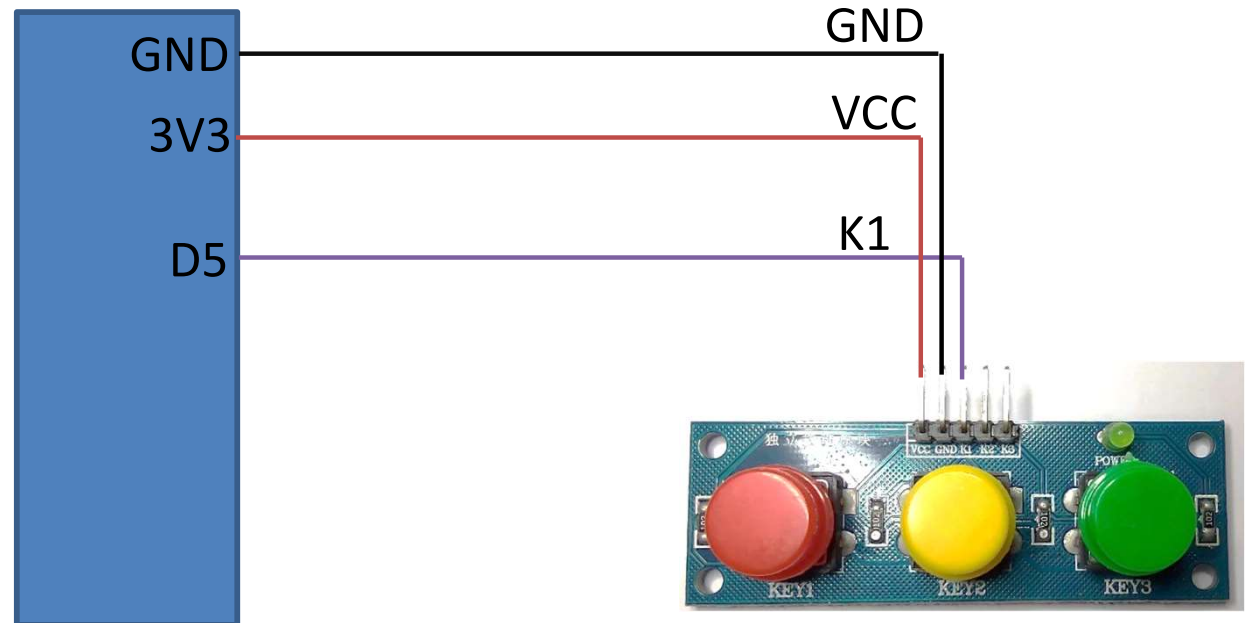
```
while True:
```

```
    Sw = K1.value()
```

```
    print(Sw)
```

```
    sleep(0.1)
```

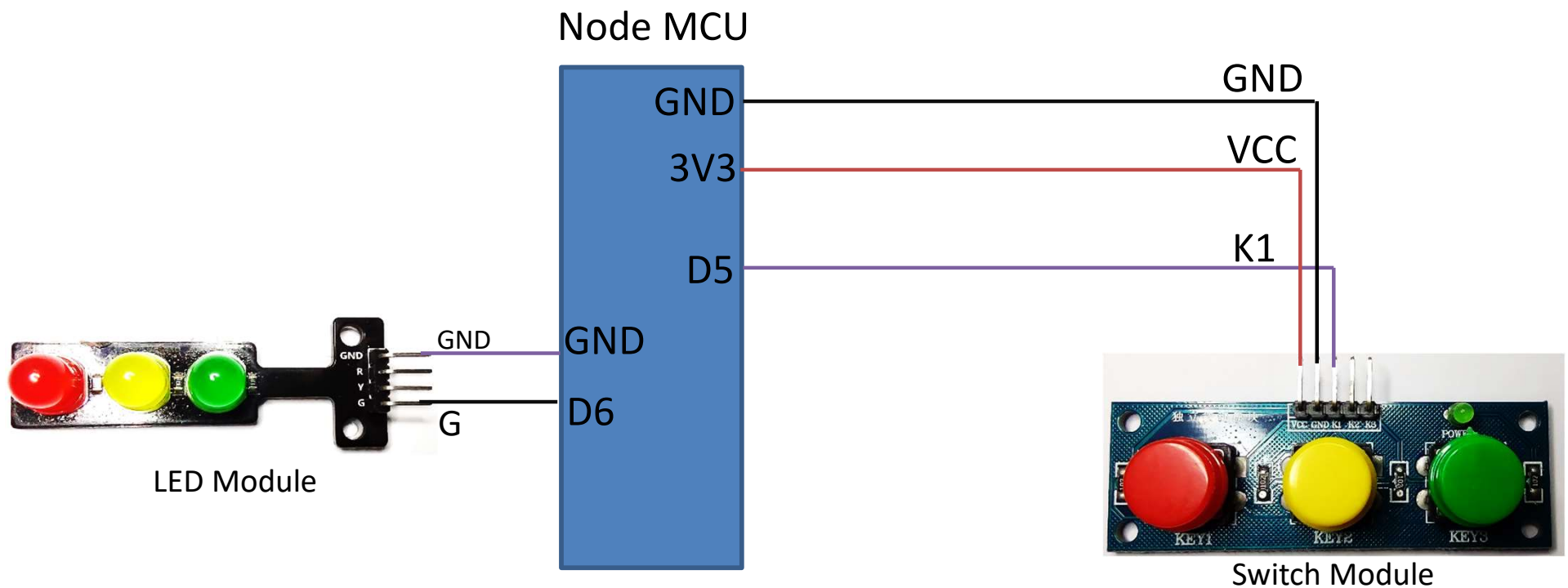
Node MCU



Switch Module

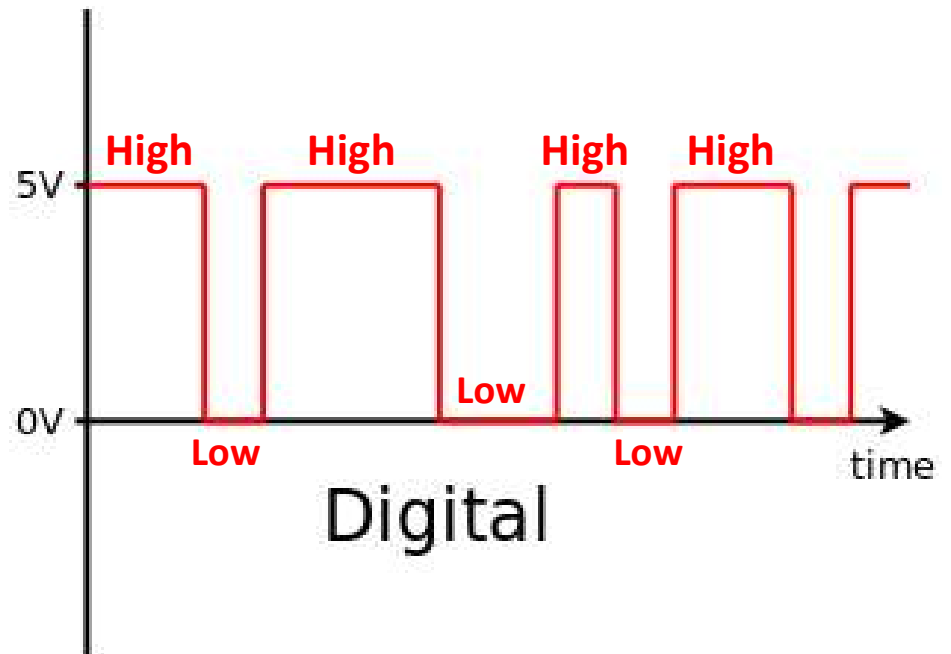
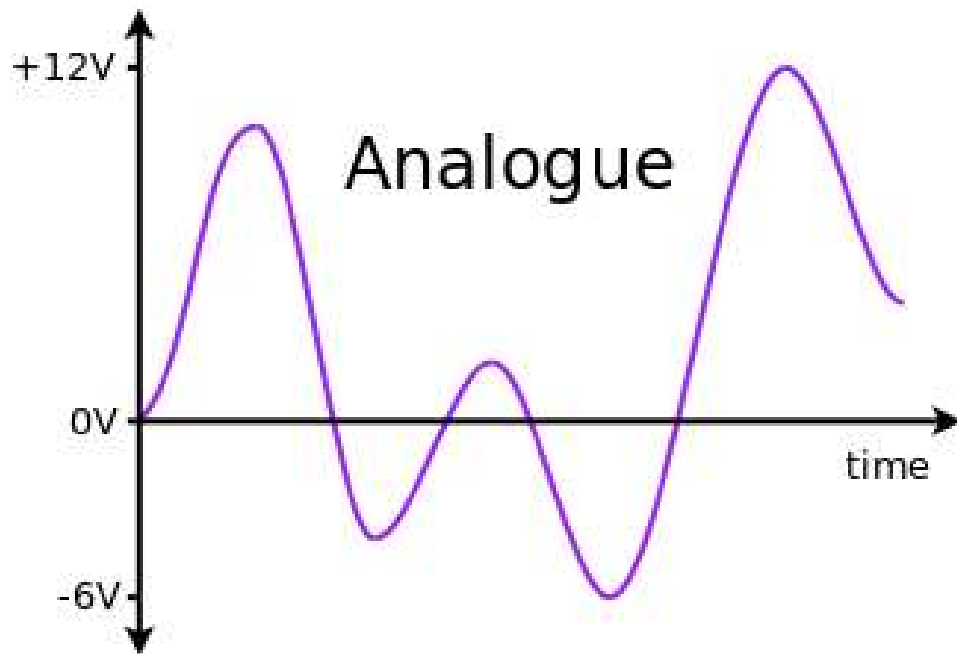
แบบฝึกหัดการใช้งานสวิตช์และ LED

1. เขียนโปรแกรมให้กดปุ่มแล้ว LED สว่าง เมื่อปล่อยปุ่ม LED ดับ
2. เขียนโปรแกรมให้กดปุ่มแล้ว LED ดับ เมื่อปล่อยปุ่ม LED สว่าง
3. เขียนโปรแกรมให้กดปุ่มแล้ว LED กระพริบ 5 ครั้ง



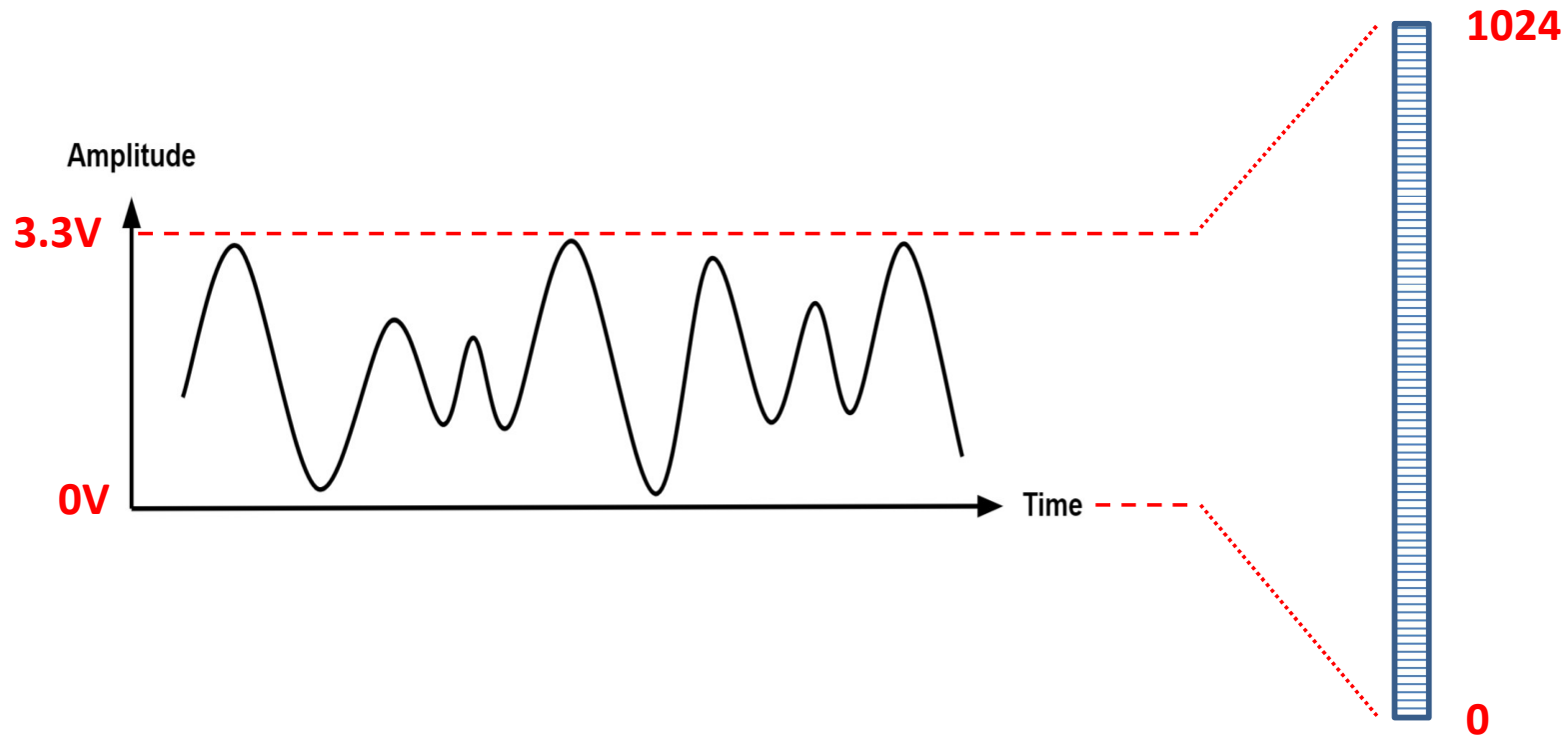
สัญญาณแอนะล็อก

- สัญญาณแอนะล็อกจะมีความต่อเนื่องและมีค่าแรงดันไฟฟ้า (voltage) ได้ไม่จำกัด
- สัญญาณดิจิทัลจะมีค่าได้เฉพาะ 0 หรือ 1 เท่านั้น (มีแรงดันหรือไม่มีแรงดันไฟฟ้า)
- การรับสัญญาณแอนะล็อกจะต้องอาศัยวงจร **Analog-to-digital converter (ADC)**



การรับสัญญาณแอนะล็อก

- วงจร ADC จะแปลงสัญญาณแอนะล็อกให้เป็นตัวเลข มีค่า 0-1024
- ตัวเลขที่ได้จะเป็นอัตราส่วนกับค่าแรงดันแอนะล็อก โดย 0V จะถูกแปลงเป็น 0 และ 3.3V จะถูกแปลงเป็น 1024



- สูตรคำนวณ: ค่าตัวเลข = $1024 * (X / 3.3)$ เมื่อ X คือค่าแรงดันแอนะล็อก
- Ex แรงดันแอนะล็อก = 2.5V, วงจร ADC จะให้ค่าเป็น $1024 * (2.5 / 3.3) = 775$

การรับสัญญาณแอนะล็อก

- ต้องใช้ ADC class จาก machine library
- สร้าง object จาก ADC class
- `adc_object = ADC(0)` สำหรับ ESP8266 จะใช้ได้เฉพาะ ADC(0) เท่านั้น
- ใช้ method `.read()` ในการอ่านค่า โดยค่าที่ได้จะอยู่ระหว่าง 0-1024

```
from machine import Pin, ADC  
from time import sleep
```

```
Pot = ADC(0)
```

```
while True:
```

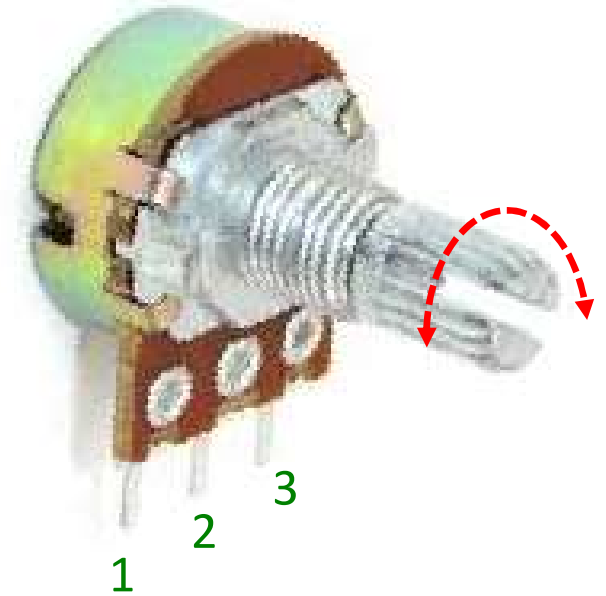
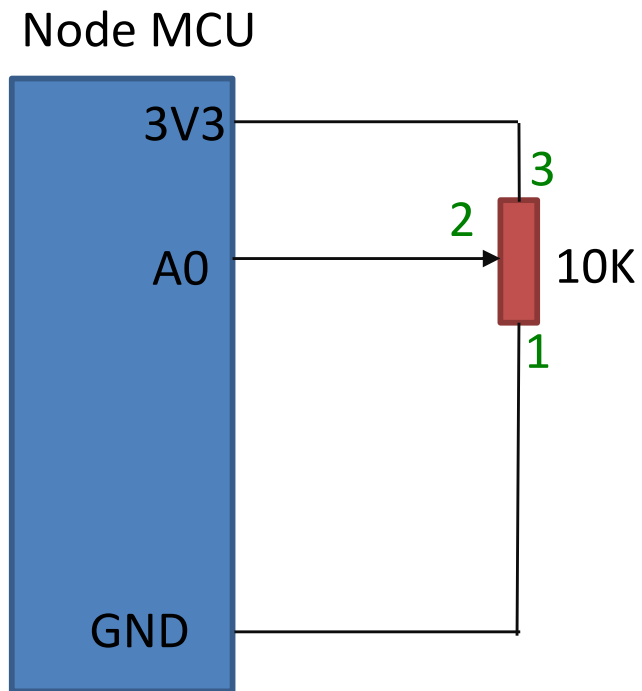
```
    pot_value = Pot.read()
```

```
    print(pot_value)
```

```
    sleep(0.5)
```

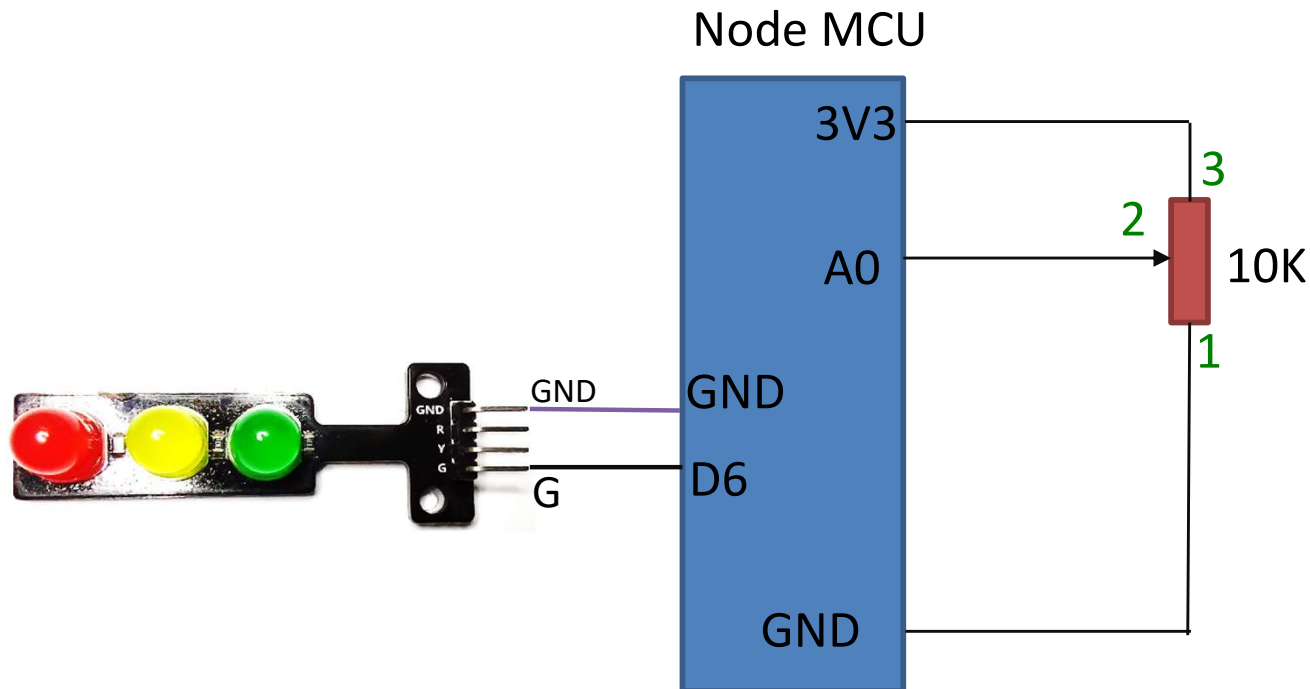
การรับสัญญาณแอนะล็อก

- ต่อดังต่อไปนี้เพื่อใช้ทดสอบการทำงานของโปรแกรม



แบบฝึกหัด

เขียนโปรแกรมควบคุมไฟกระพริบที่สามารถปรับอัตราเร็วของการ
กระพริบได้โดยใช้ตัวต้านทานปรับค่าได้

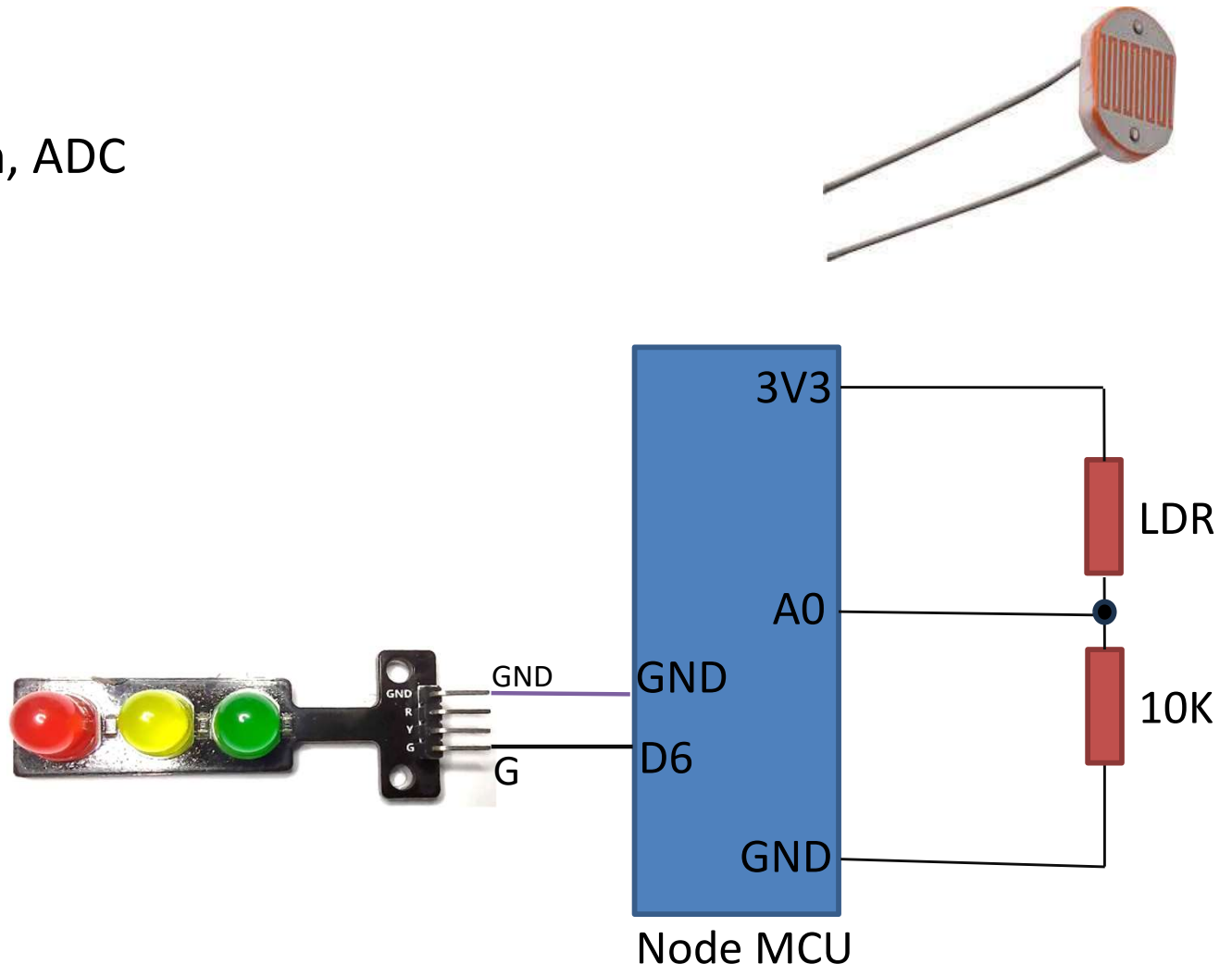


ตัวต้านทานเปลี่ยนค่าตามแสง

- ตัวต้านทานเปลี่ยนค่าตามแสง Light-dependent Resistor (LDR) จะมีความต้านทานเปลี่ยนไปตามความเข้มแสงที่ตกกระทบ
- ความเข้มแสงสูงความต้านทานจะต่ำ ความเข้มแสงต่ำความต้านทานจะสูง

```
from machine import Pin, ADC  
from time import sleep
```

```
LDR = ADC(0)  
while True:  
    Light = LDR.read()  
    print(Light)  
    sleep(1)
```



แบบฝึกหัด

ตัวอย่าง

โปรแกรมปรับอัตราการกระพริบตามค่าระดับแสงที่ตกกระทบ LDR

```
from machine import Pin, ADC  
from time import sleep
```

```
LED = Pin(2, Pin.OUT) #Builtin LED GPIO2
```

```
LDR = ADC(0)
```

```
while True:
```

```
    Light = LDR.read()
```

```
    LED.value(0)
```

```
    sleep(Light/100)
```

```
    LED.value(1)
```

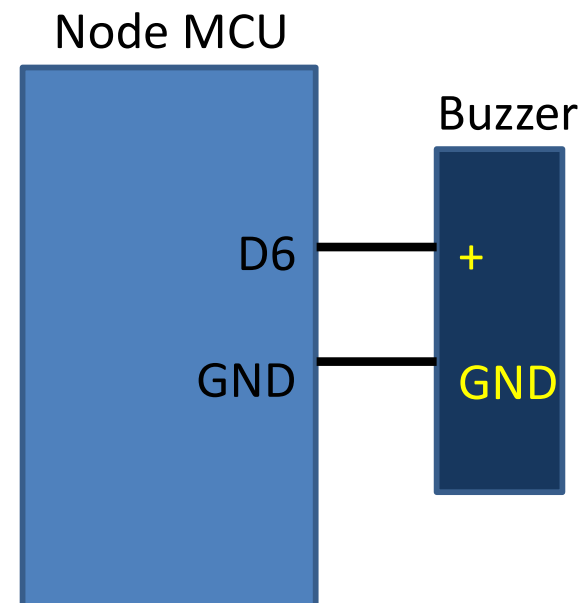
```
    sleep(Light/100)
```

การใช้งาน buzzer

- Buzzer ทำหน้าที่สร้างความถี่เสียง
- มี 2 ชนิด
 1. สร้างความถี่เสียงตัวเอง
 2. สร้างความถี่เสียงจากภายนอก
- ชนิดที่ 1 ต่อกับ gpio และควบคุมการปิด-เปิดได้เช่นเดียวกับ LED

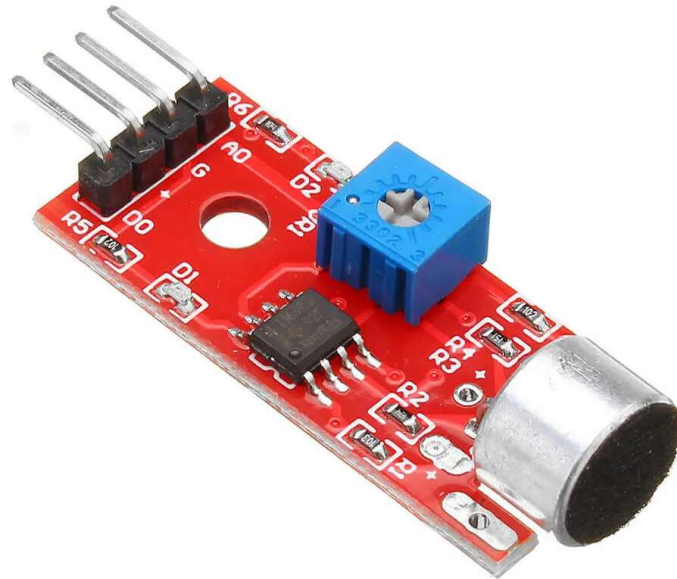


```
from machine import Pin
from time import sleep
buzzerPin = 12 #D6
buzzer = Pin(buzzerPin, Pin.OUT)
while True:
    buzzer.value(1)
    sleep(0.5)
    buzzer.value(0)
    sleep(0.5)
```



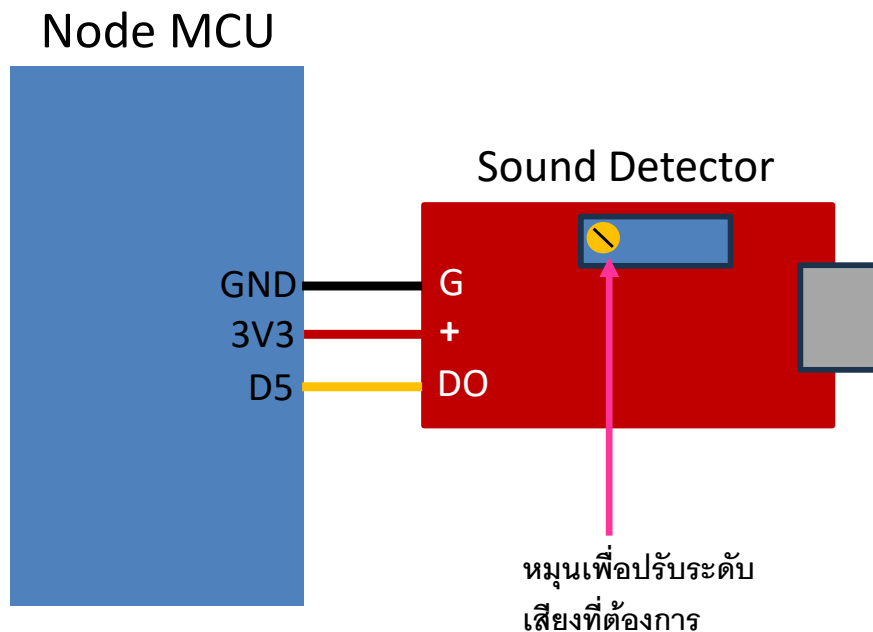
เซนเซอร์ตรวจจับเสียง (Sound detector)

- ให้สัญญาณออกได้ทั้งดิจิทัลและแอนะล็อก
- ปรับระดับความดังที่ต้องการให้ตรวจจับได้
- สัญญาณดิจิทัลจะเป็น 0 ถ้าเสียงดังน้อยกว่าระดับที่กำหนดและ
เป็น 1 เมื่อเสียงดังถึงระดับที่กำหนด



เซนเซอร์ตรวจจับเสียง (Sound detector)

- การใช้งานสัญญาณดิจิทัล
- ต้องปรับระดับความดังที่ต้องการให้ตรวจจับได้
- สัญญาณดิจิทัลจะเป็น 0 ถ้าเสียงดังน้อยกว่าระดับที่กำหนดและ
เป็น 1 เมื่อเสียงดังถึงระดับที่กำหนด

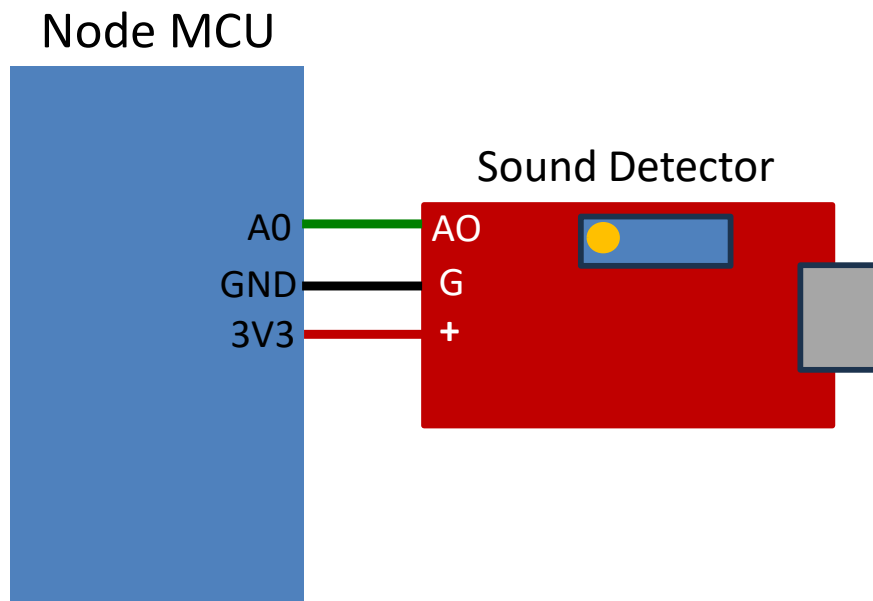


```
from machine import Pin  
from time import sleep
```

```
Mic = Pin(14, Pin.IN) #D5  
while True:  
    sound = Mic.value()  
    print(sound)  
    sleep(0.1)
```

เซนเซอร์ตรวจจับเสียง (Sound detector)

- การใช้งานสัญญาณแอนะล็อก
- สัญญาณแอนะล็อกจะมีค่าระหว่าง 0-1024 ขึ้นกับความดังของเสียง
- ใช้ Plotter ใน Thonny เพื่อดูกราฟระดับเสียง

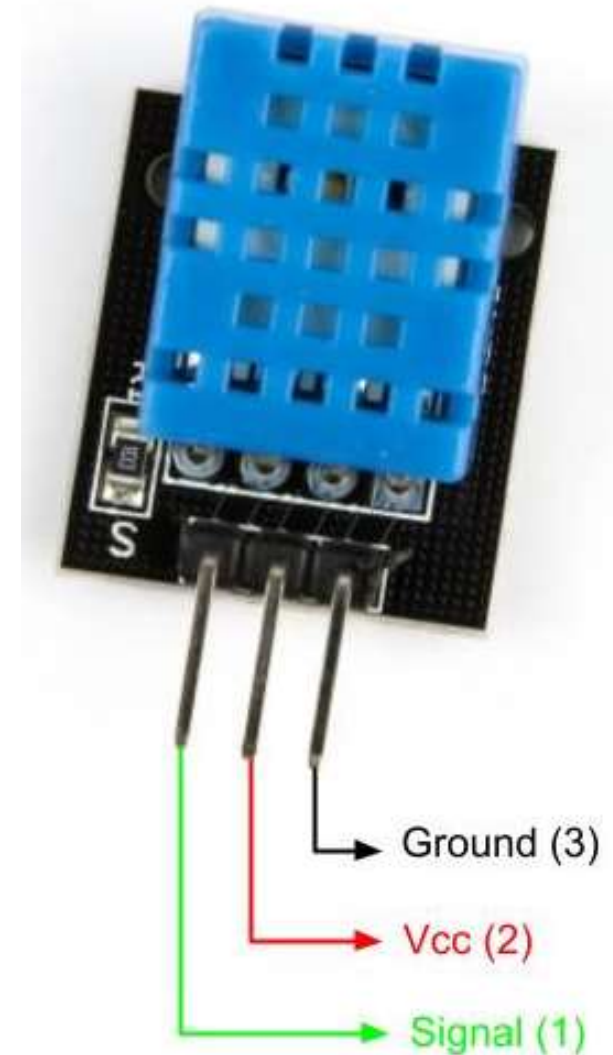


```
from machine import Pin, ADC  
from time import sleep
```

```
Mic = ADC(0)  
while True:  
    sound = Mic.read()  
    print(sound)  
    sleep(0.05)
```

เซนเซอร์อุณหภูมิและความชื้น (DHT11)

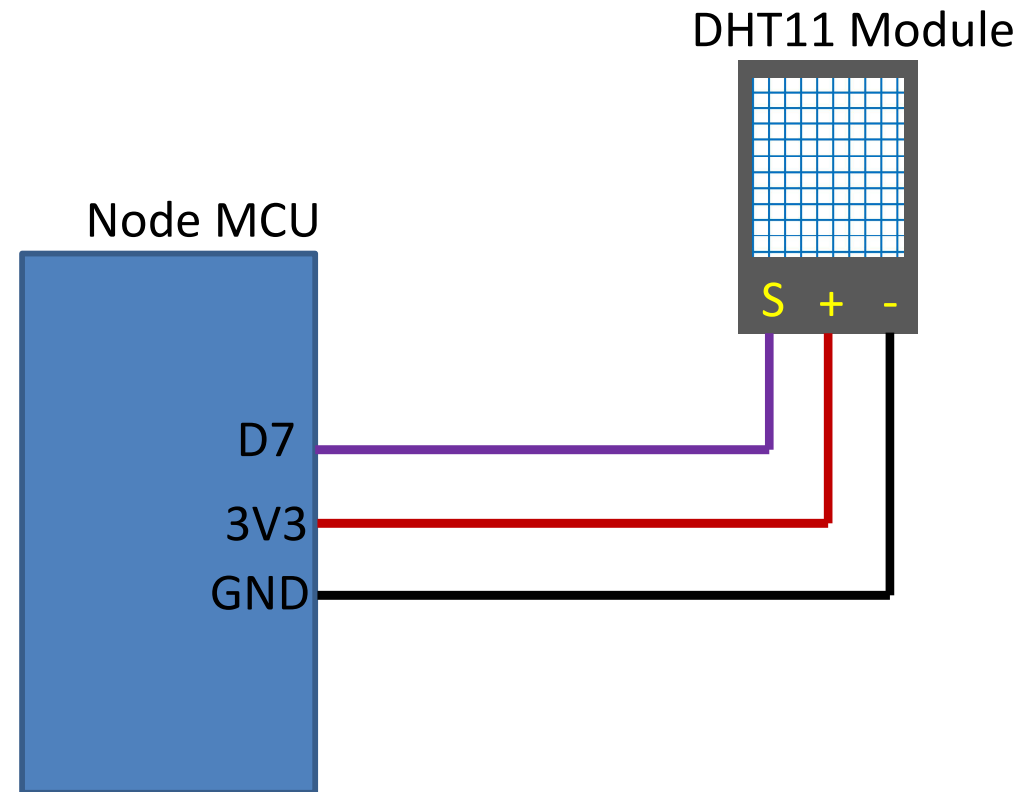
- วัดอุณหภูมิในช่วง 0-50 °C
- วัดความชื้นในช่วง 20-95%
- ทำงานที่แรงดัน 3.3V-5V



เซนเซอร์อุณหภูมิและความชื้น (DHT11)

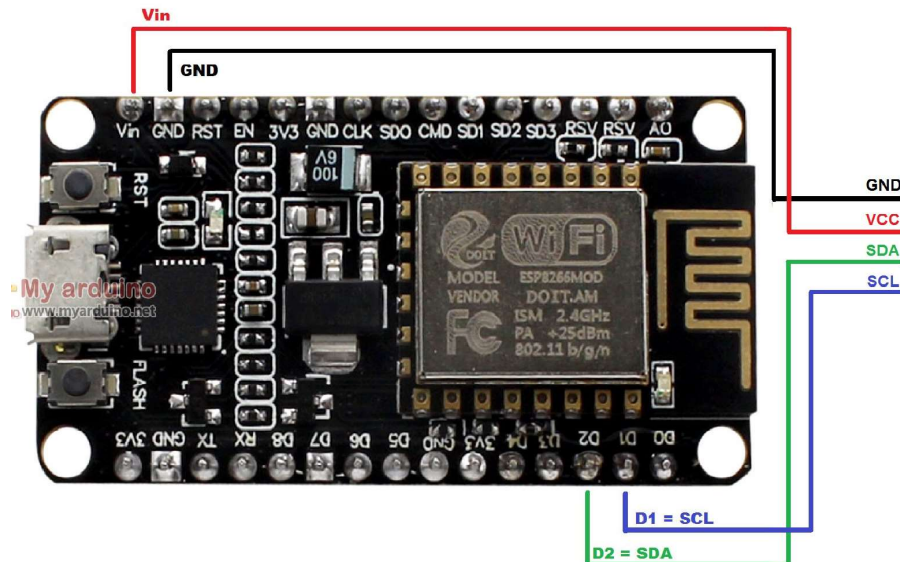
```
from machine import Pin
from time import sleep
import dht
```

```
sensor = dht.DHT11(Pin(13))
while True:
    try:
        sleep(2)
        sensor.measure()
        temp = sensor.temperature()
        humid = sensor.humidity()
        print("Temperature:",temp, end=' ')
        print("°C Humidity:",humid,"%")
    except OSError as e:
        print('Failed to read sensor.')
```



Liquid Crystal Display (LCD)

- แสดงตัวอักษรและตัวเลข (Alphanumeric) 2 บรรทัด x 16 ตัวอักษร
- เชื่อมต่อผ่าน I²C (SCL, SDA)
- ทำงานที่แรงดัน 5V (V_{IN})
- ต้องติดตั้ง Library (LiquidCrystal_I2C) ก่อนจะใช้งาน

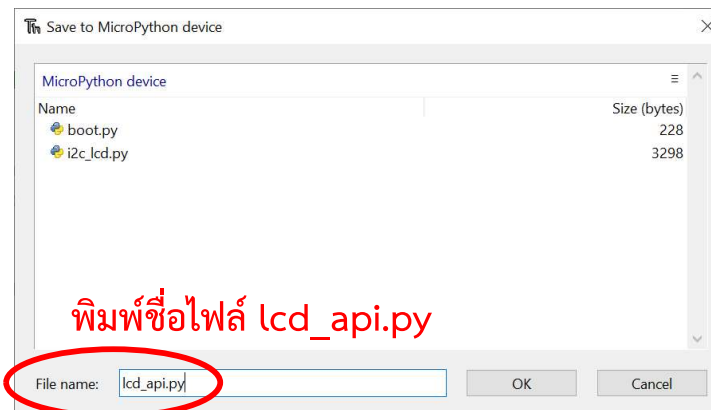
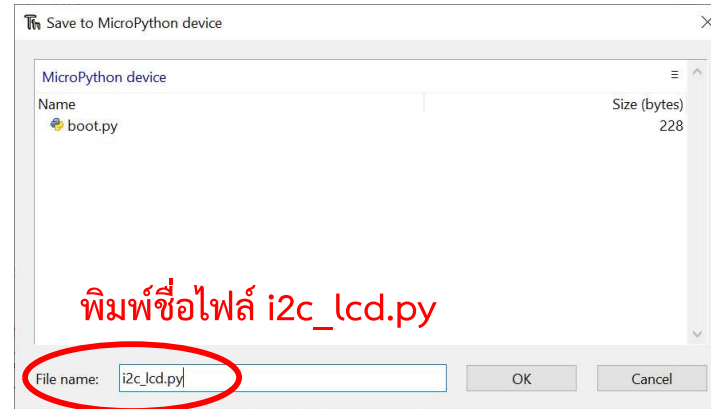
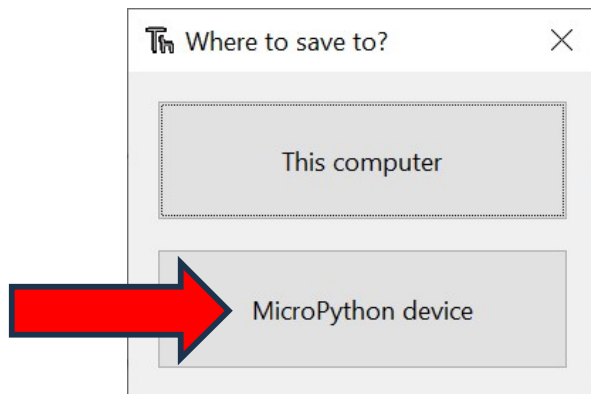


LCD	NodeMCU
GND	GND
Vcc	V _{USB}
SDA	D2
SCL	D1



Liquid Crystal Display (LCD)

- ต้องติดตั้ง Library ก่อนจะใช้งาน โดยเปิดไฟล์ต่อไปนี้ใน Thonny
 - i2c_lcd.py
 - lcd_api.py
- จากนั้นเลือก save as... เพื่อจัดเก็บทั้ง 2 ไฟล์ลงใน ESP8266

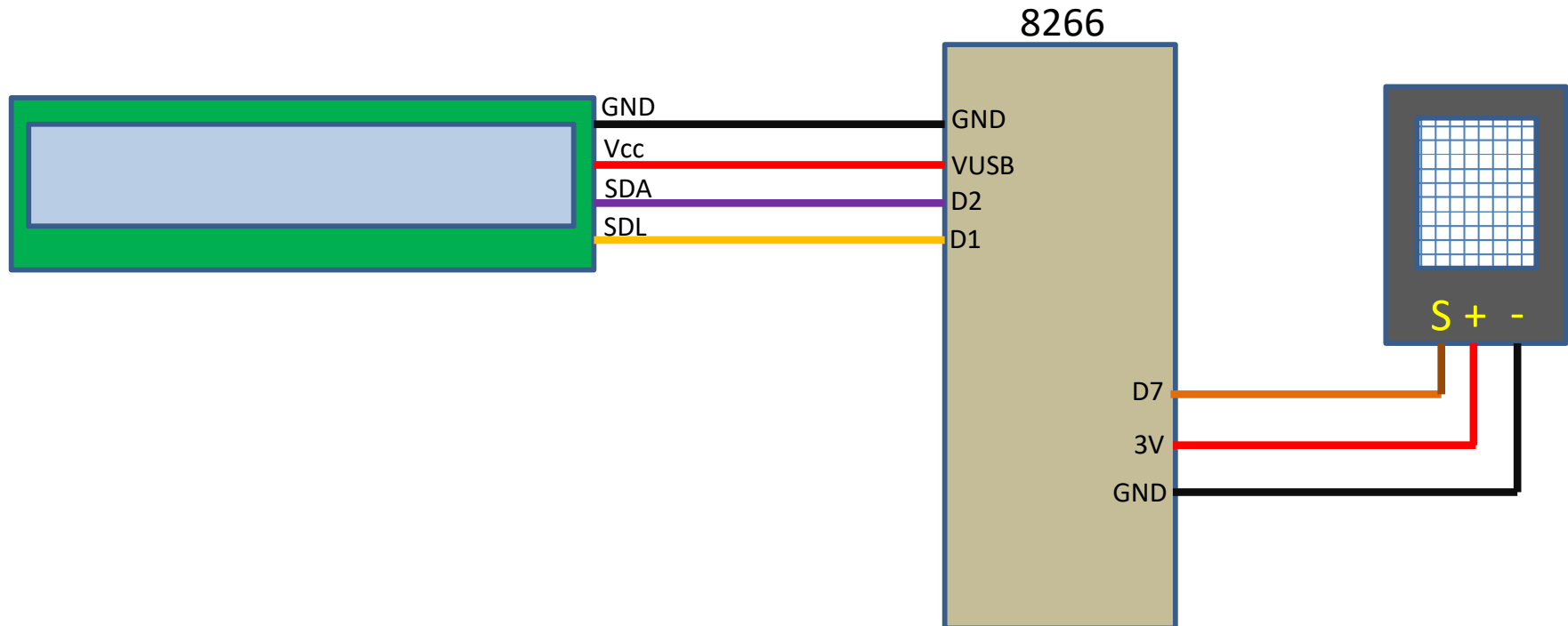


Liquid Crystal Display (LCD)

- ตัวอย่าง **LCD_Temp_Display.py**

<u>DHT11</u>	<u>NodeMCU</u>
GND	GND
Vcc	3V3
OUT	D4

<u>LCD</u>	<u>NodeMCU</u>
GND	GND
Vcc	V _{IN}
SDA	D2
SCL	D1



Q&A