

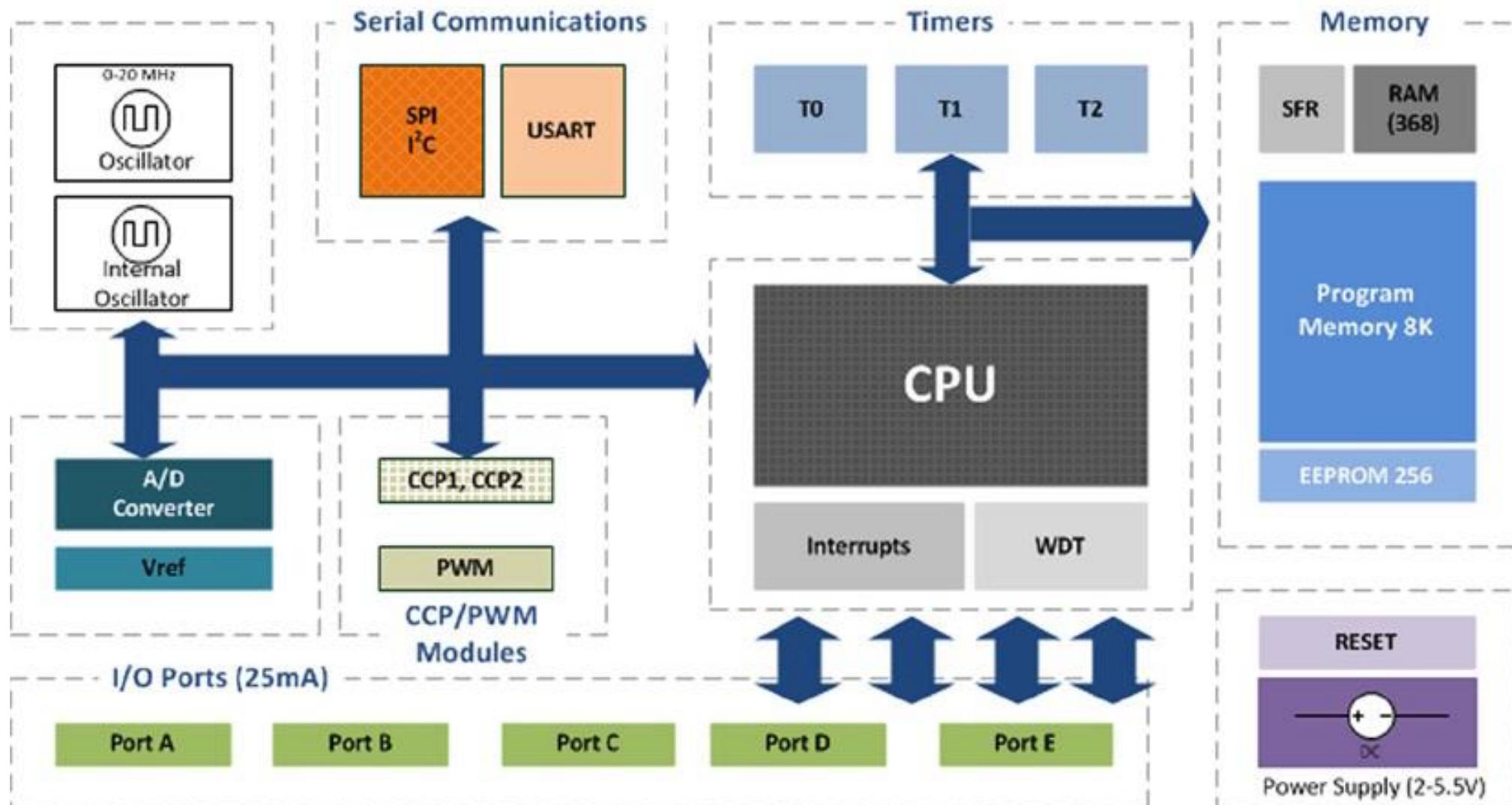
ESP 32



Microcontroller

- ไมโครคอนโทรลเลอร์หรือระบบคอมพิวเตอร์ขนาดเล็ก ประกอบด้วย
 - หน่วยประมวลผลซีพียู (CPU)
 - หน่วยความจำ (Memory)
 - ไฟเมอร์และเคาน์เตอร์ กำหนดเวลาหน่วงต่าง ๆ
 - อินพุตเอาต์พุตพอร์ต (I/O Port) ประเภทต่าง ๆ เช่น SPI หรือ UART เป็นต้น
 - ปัจจุบันยังประกอบด้วยการสื่อสารแบบต่าง ๆ เช่น บลูทูธ หรือ ไวไฟ (WIFI) เป็นต้น

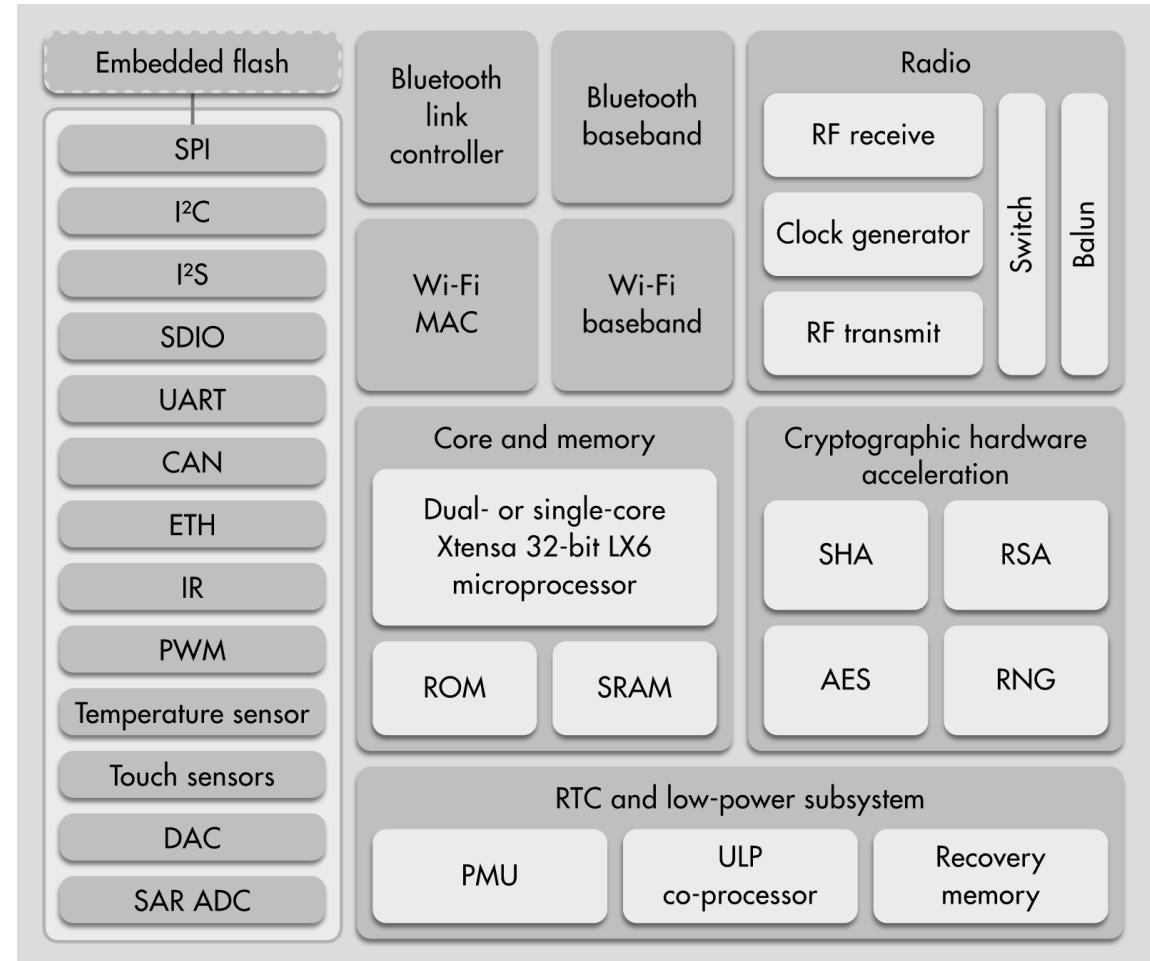
โครงสร้างทั่วไปของไมโครคอนโทรลเลอร์



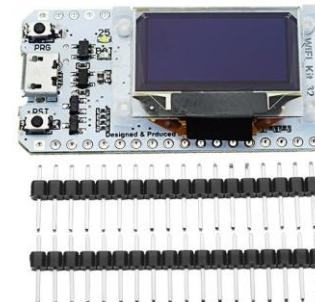
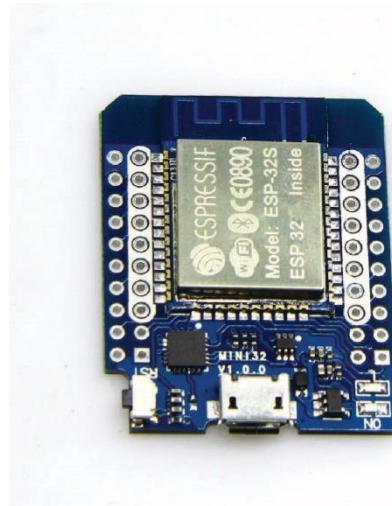
บอร์ดที่ใช้

- ESP 32

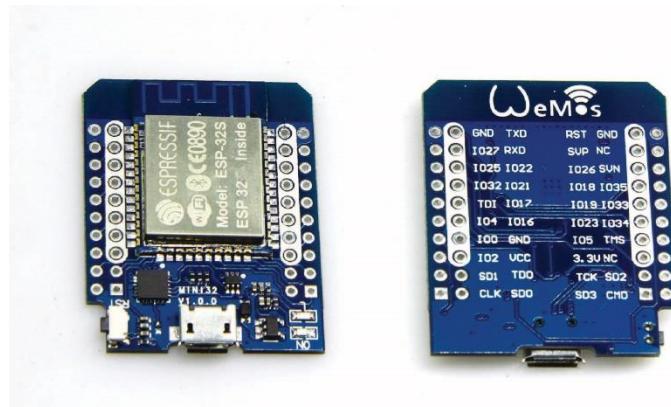
- ชิปปี้ใช้สถาปัตยกรรม Tensilica LX6 แบบ 2 แกน สัญญาณนาฬิกา 240 MHz
- มีแรมในตัว 512KB
- รองรับการเชื่อมต่อรอมภายนอกสูงสุด 16MB
- WiFi มาตรฐาน 802.11 b/g/n 2.4 GHz ส่งได้ สูงสุด 150 Mbps
- Bluetooth เวอร์ชัน 4.2 (BLE)



ໂມຊູລ ESP32



บอร์ดที่ใช้



can be used always
can be used if no other function is used
cannot be used

	RST	RST
ADC2	GPIO36	SVP
	GPIO26	I026
SPI0:SCK	GPIO18	I018
SPI0:MISO	GPIO19	I019
SPI0: MOSI	GPIO23	I023
SPI0:CS0	GPIO5	I05
	3V3	3V3
	GPIO13	TCK
UART1:RXD	GPIO10	SD3

A0
D0
D5
D6
D7
D8
3V3

TX
RX
D1
D2
D3
D4
GND
5V

Pin Lables
Wemos
D1 mini

TXD	GPIO1	UART0:TXD
RXD	GPIO3	UART0:RXD
I022	GPIO22	I2C0:SCL
I021	GPIO21	I2C0:SDA
I017	GPIO17	
I016	GPIO16	
GND	GND	
VCC	5V	
TD0	GPIO15	
SD0	GPIO7	

GND	GND	RST
NC	SVP	
ADC3	GPIO39	I026
ADC1	GPIO35	I035
ADC0	GPIO33	I033
	GPIO34	I034
	GPIO14	I023
UART1:TXD	GPIO9	SD2
	GPIO11	CMD
		SD3

Pin Lables



TXD	GND	GND
RXD	GPIO27	
I022	GPIO25	
I025	GPIO32	DAC0
I021	GPIO12	
I032	GPIO17	
I017	GPIO16	
I016	GPIO4	PWM0:2
I04	GPIO0	PWM0:1
GND	GPIO2	PWM0:0
VCC	GPIO8	
I02	GPIO6	
TD0	SD1	
SD0	CLK	

Pin Lables

LED0

ขาต่อ ๆ ของ ESP32

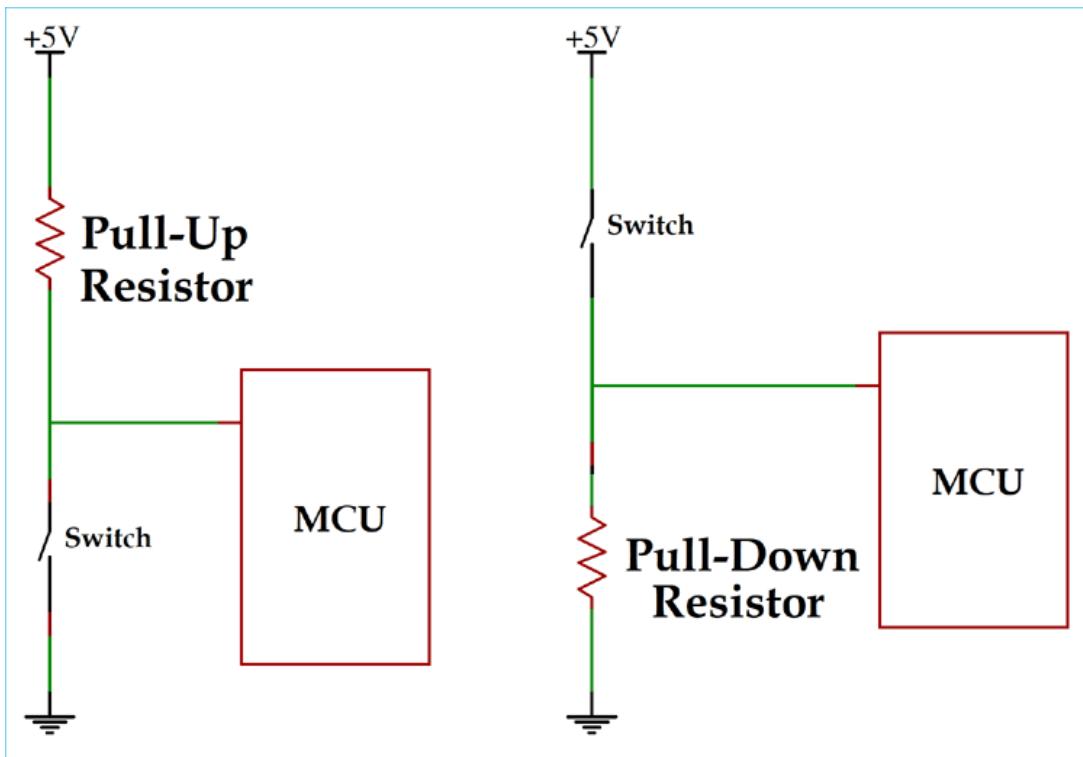
- GPIO จำนวน 32 ช่อง
- รองรับ PWM ทุกช่อง
- รองรับ ADC จำนวน 12 ช่อง
- รองรับ DAC จำนวน 2 ช่อง
- รองรับ SPI จำนวน 3 ช่อง
- รองรับ I2C จำนวน 2 ช่อง
- รองรับ UART จำนวน 3 ช่อง

GPIO

- General Purpose Input/Output (GPIO)

- เป็นขาที่ต่อตรงไปยังส่วนของโปรเซลเซอร์แต่ละขาของอุปกรณ์
- สามารถกำหนดได้อย่างอิสระเป็นอินพุตและเอาท์พุต
- แรงดันของ GPIO ทำงานที่ 3.3 โวลท์ การเชื่อมต่ออุปกรณ์ที่แรงดันสูงกว่า อาจเกิดความเสียหายได้

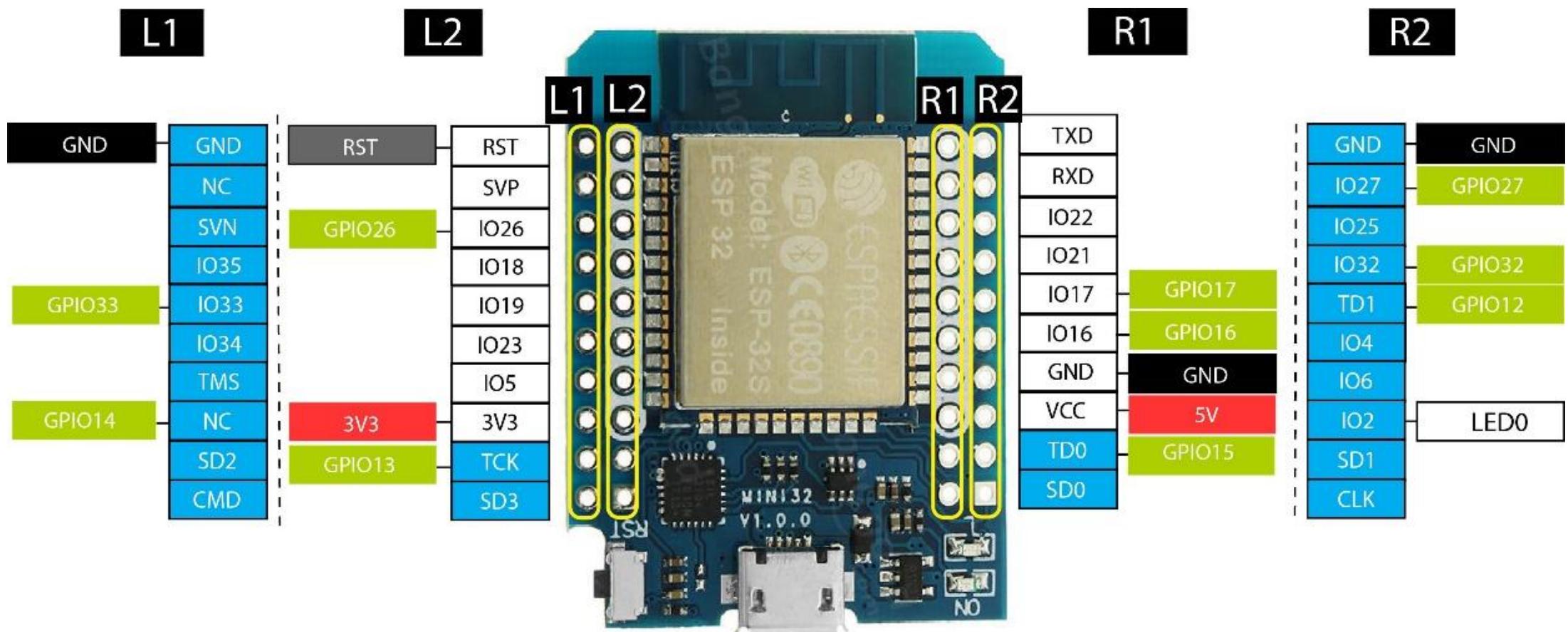
Pull-Up/Pull-Down



- GPIO ไม่มีการเชื่อมต่อจะอยู่ในสภาพ floating state
 - Pull up ทำให้อยู่สถานะ high
 - Pull down ทำให้อยู่สถานะ low
- การ pull up

```
1 from machine import Pin  
2 p4 = Pin(4, Pin.IN, Pin.PULL_UP)
```

การใช้งาน GPIO

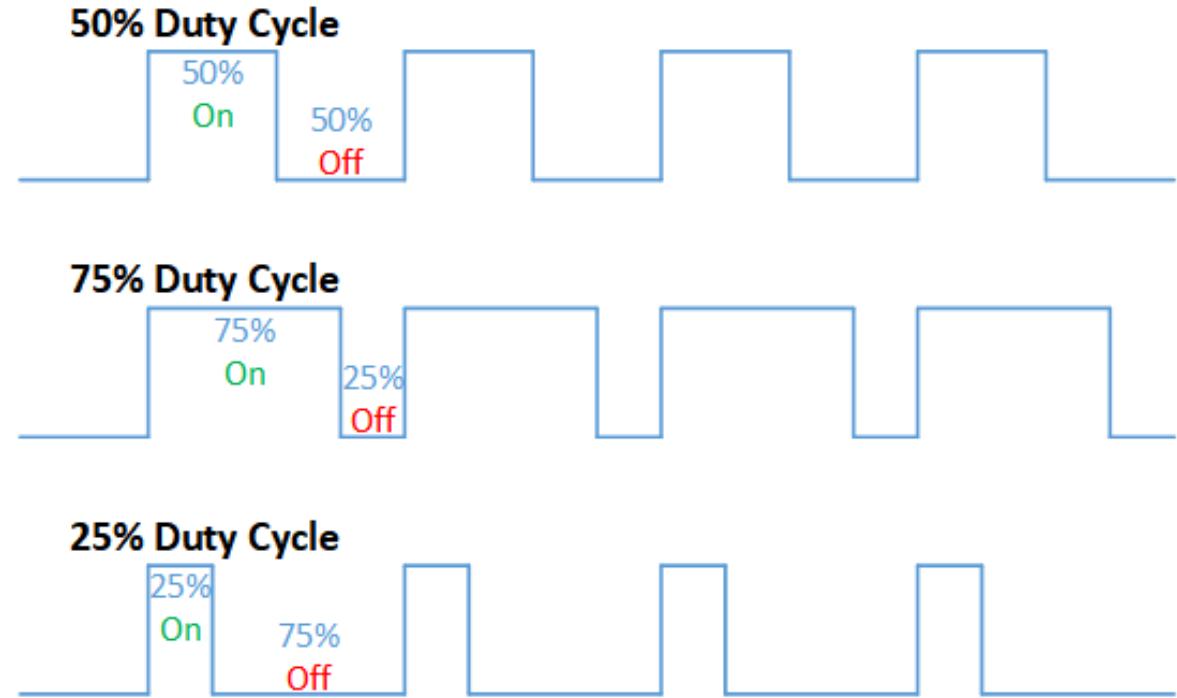


PWM

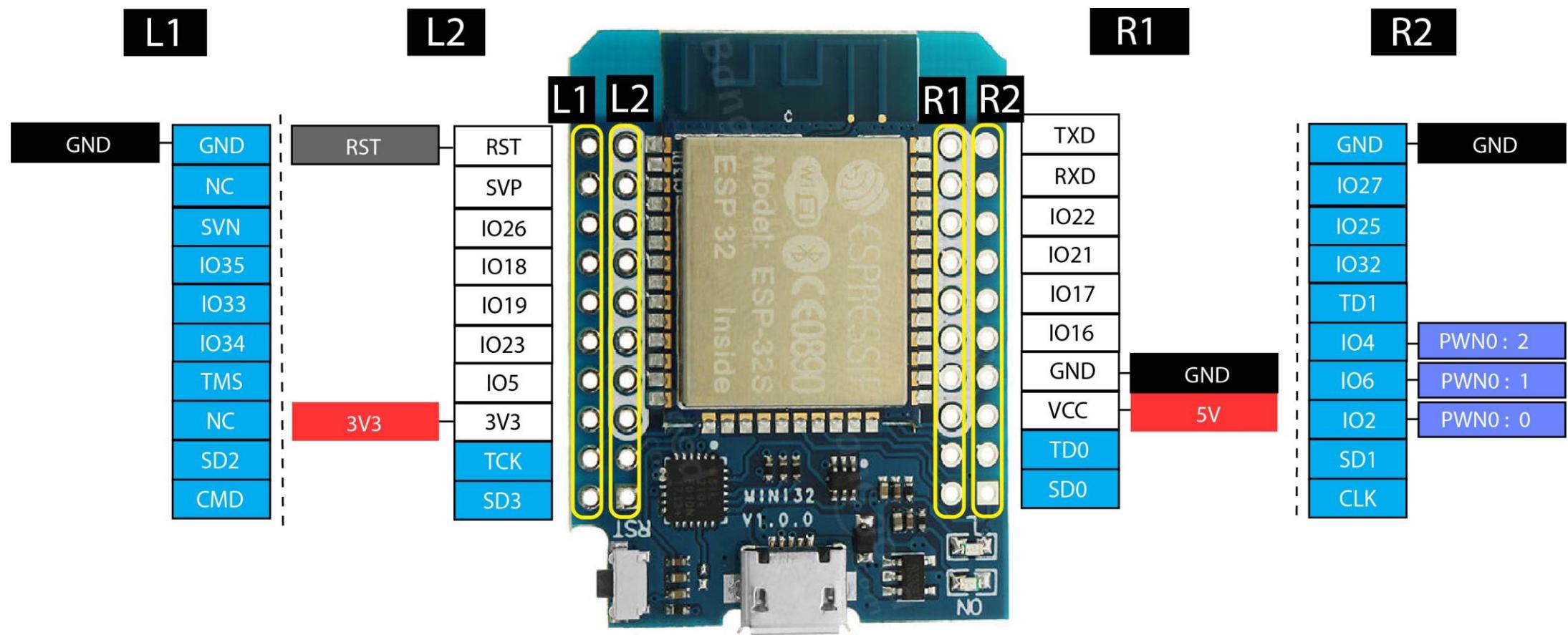
- Pulse Width Modulation (PWM) ใช้กำเนิด

สัญญาณ PWM ประกอบด้วยสองส่วนหลัก

- Duty Cycle
- ความถี่
- สำหรับเปิดปิดไฟ หรือเปิดปิดปั๊มน้ำ เป็นต้น

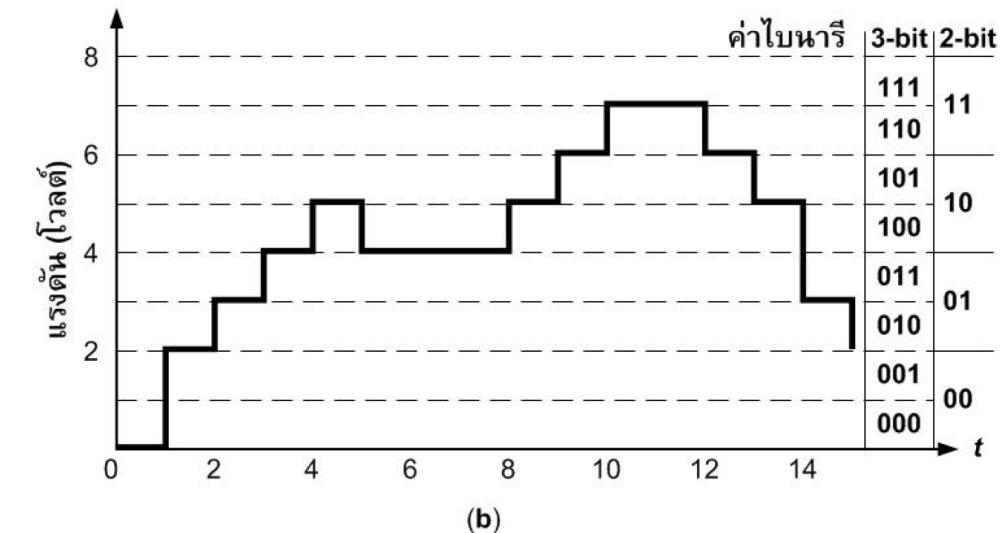
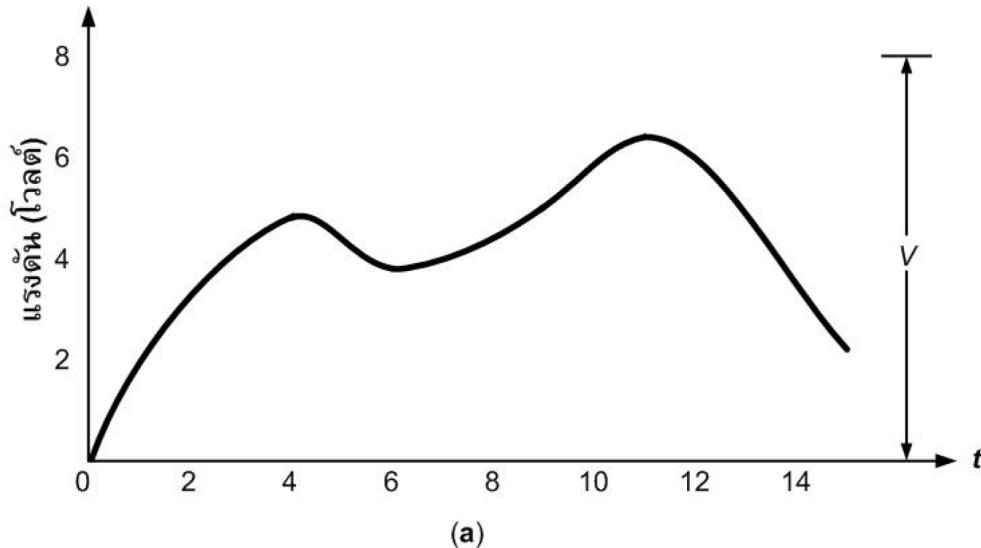


กลุ่มเพื่อการทำงาน PWM



ADC

- Analog to Digital Conversion (ADC) เป็นการใช้เพื่อแปลงสัญญาณอนาล็อกไปเป็นดิจิทัล



การปรับตั้งค่า

- การลดทอนสัญญาณ

- ADC.ATTN_0DB: 0dB รองรับแรงดันสูงสุด 1.0 V (ดีฟอลท์)
- ADC.ATTN_2_5DB: ลดทอน 2.5 dB รองรับแรงดันสูงสุดประมาณ 1.34 V
- ADC.ATTN_6DB: ลดทอน 6dB รองรับแรงดันสูงสุดประมาณ 2.00 V
- ADC.ATTN_11_DB : ลดทอน 11dB รองรับแรงดันสูงสุดประมาณ 3.6 V

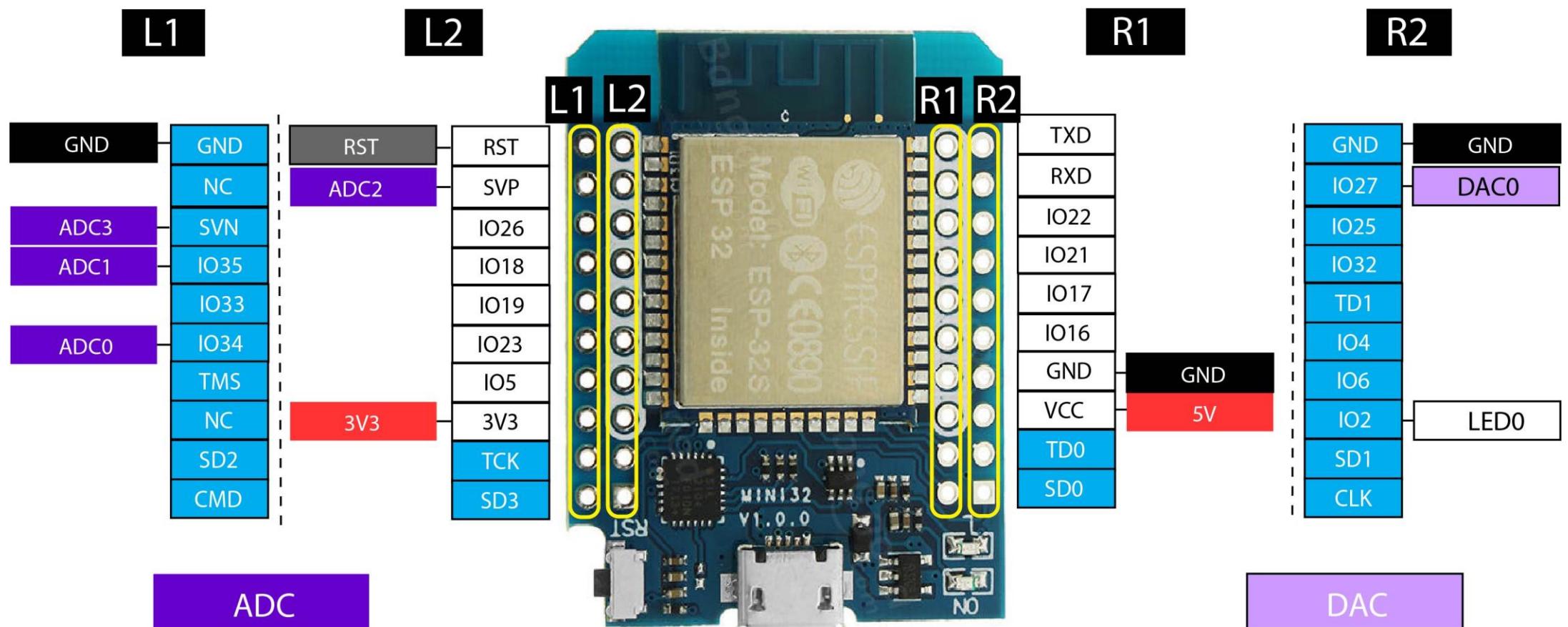
- ความละเอียดของสัญญาณ

- ADC.WIDTH_9BIT: ข้อมูล 9 บิต
- ADC.WIDTH_10BIT: ข้อมูล 10 บิต
- ADC.WIDTH_11BIT: ข้อมูล 11 บิต
- ADC.WIDTH_12BIT: ข้อมูล 12 บิต (ดีฟอลท์)

DAC

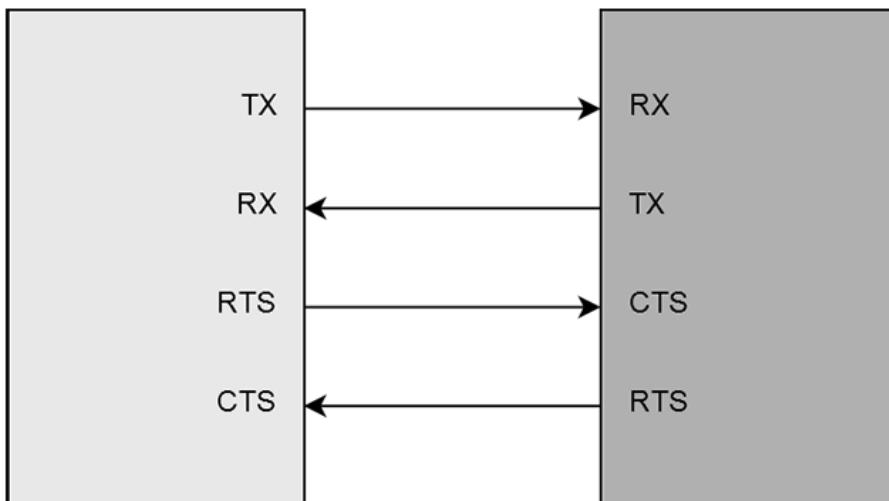
- Digital to Analog Conversion (DAC) แปลงสัญญาณจากดิจิทัลไปเป็นอนาล็อก ไม่นิยมเท่าที่ควร เนื่องจากจะเกิดการลดTHONสัญญาณ

กลุ่มการแปลงสัญญาณ



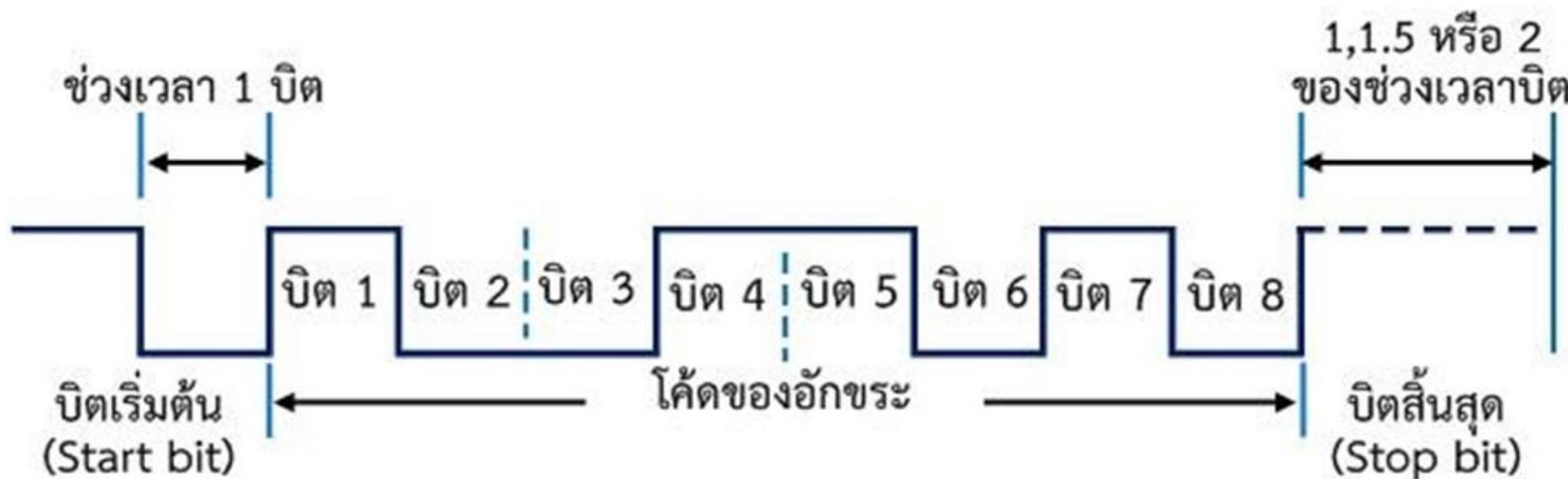
การสื่อสาร UART

- Universal Asynchronous Receiver Transmitter (UART) เป็นช่องทางการสื่อสารแบบอะซิงโครนัสแบบอนุกรม รองรับการสื่อสารที่ความเร็วสูงสุดที่ 5 Mbps.
- ESP32 ประกอบด้วย UART 3 ชุด ได้แก่ UART0, UART1 และ UART2



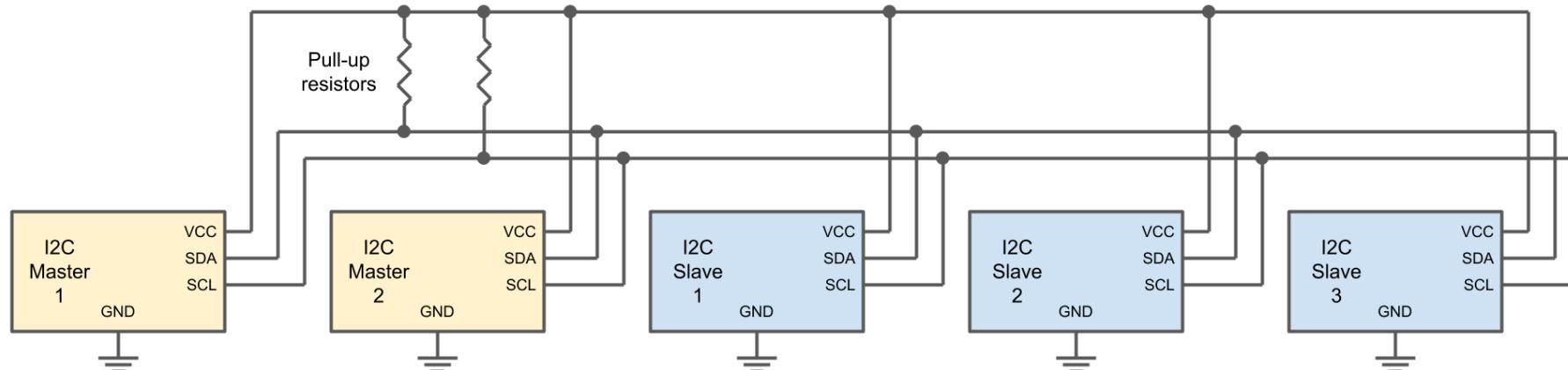
- TX (Transmitted Data) เป็นขาเพื่อส่งข้อมูลจากไมโครคอนโทรลเลอร์ไปยังอุปกรณ์อื่น
- RX (Received Data) เป็นขาเพื่อรับข้อมูลจากอุปกรณ์อื่น
- CTS (Clear to Send) สำหรับอุปกรณ์อื่นเพื่อยืนยันว่าพร้อมที่จะรับข้อมูลจากไมโครคอนโทรลเลอร์
- RTS (Request to send) สำหรับไมโครคอนโทรลเลอร์แจ้งไปยังอุปกรณ์อื่นเพื่อร้องขอการส่งข้อมูล

รูปแบบการสื่อสาร

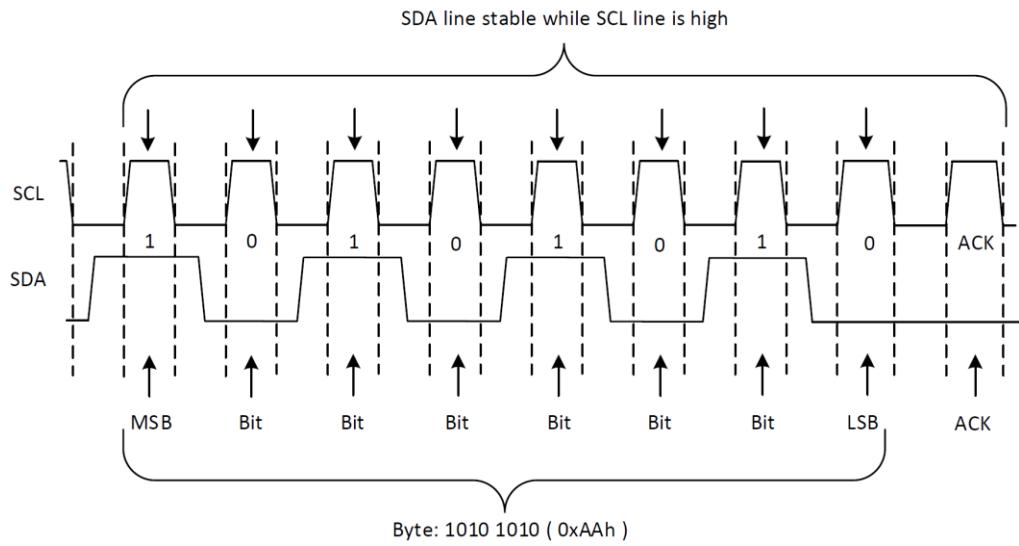


การสื่อสาร I2C

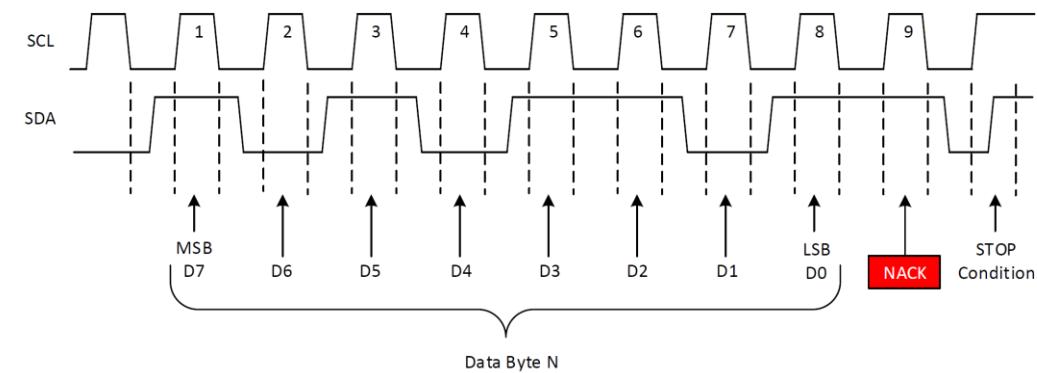
- Inter-Integrated Circuit (I2C) เป็นการสื่อสารรูปแบบบัสสัญญาณ โดยอุปกรณ์ทุกตัวจะเชื่อมบนบัส
- ข้อดี คือใช้สายเพียง 2 เส้นในรูปแบบบัส สามารถสื่อสารอุปกรณ์ได้ถึง 127 อุปกรณ์
- การสื่อสารแบบ I2C แต่ละอุปกรณ์จะมีแอดเดรสของตนเอง มาสเตอร์จะทำหน้าที่สื่อสารเพื่อรับข้อมูลจาก slave ที่ต้องการ
- อุปกรณ์แต่ละตัวสามารถทำหน้าที่ เป็นมาสเตอร์เพื่ออ่านค่าจากอุปกรณ์อื่น และเป็นสเลฟให้กับมาสเตอร์อื่นได้ด้วย



การใช้งาน Acknowledge (ACK)/Not Acknowledge (NACK)



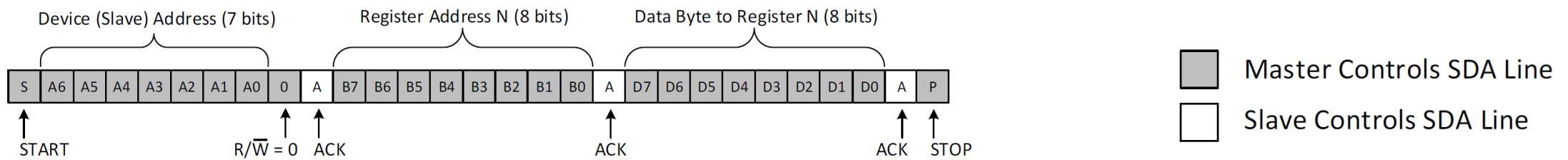
ทุกไบต์ของข้อมูลจะต้องมี ACK เพื่อแสดงว่าการสื่อสารสำเร็จ



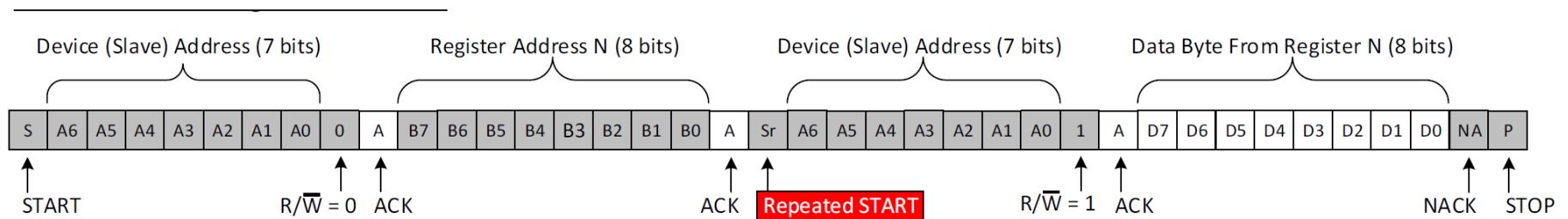
- ภาครับไม่สามารถรับข้อมูลได้
- ภาครับไม่เข้าใจคำสั่งที่ได้รับ
- ภาครับไม่สามารถรับข้อมูลเพิ่มได้อีก
- สิ้นสุดการอ่านระหว่างมาสเตอร์กับภาครับ

การอ่านและเขียนไปยังอุปกรณ์

- *Writing to a Slave On The I2C Bus*

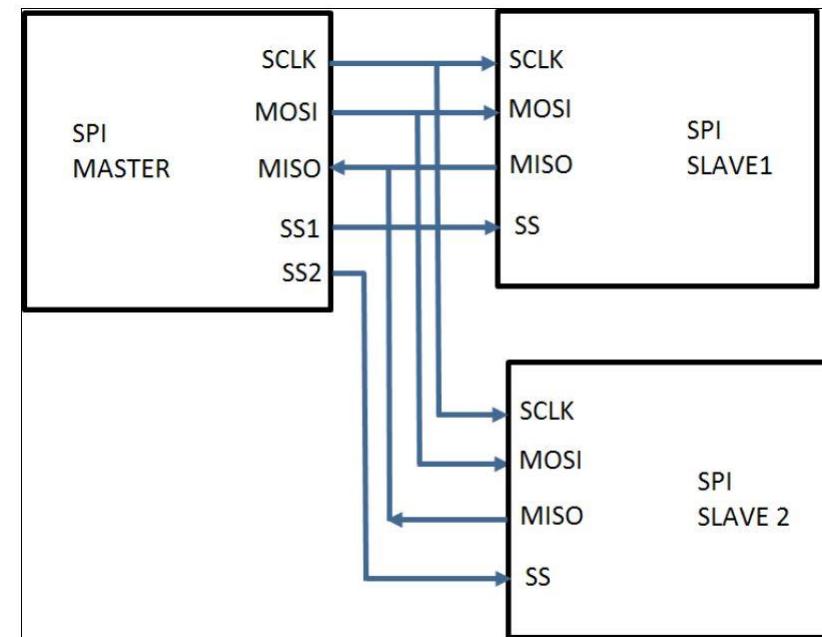
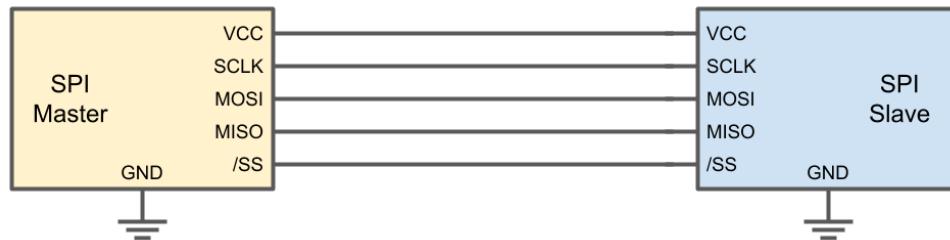


- *Reading From a Slave On The I2C Bus*



การสื่อสาร SPI

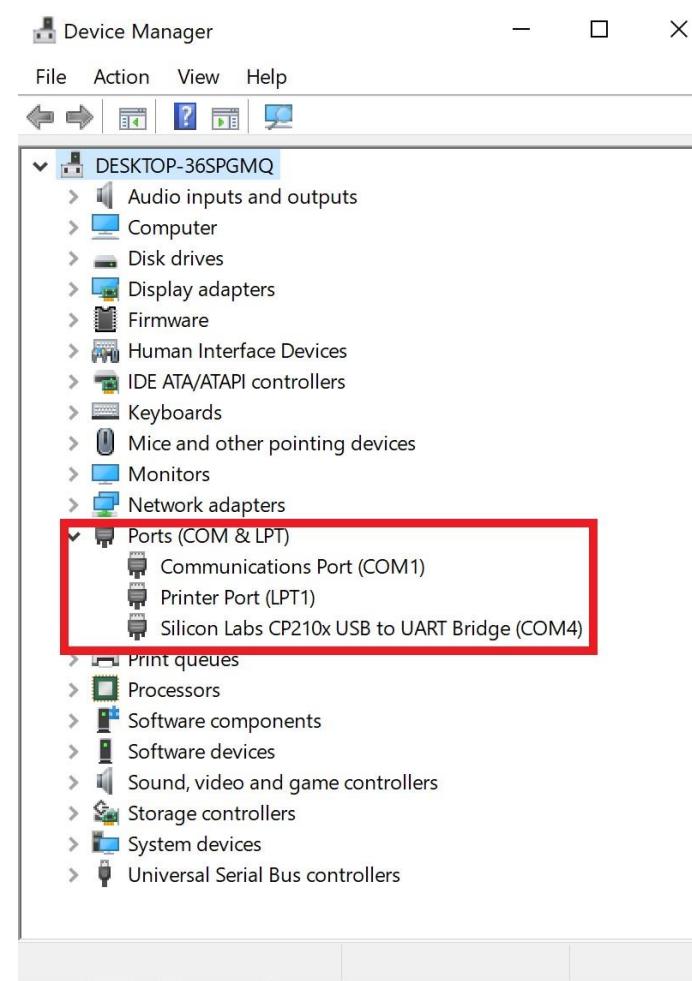
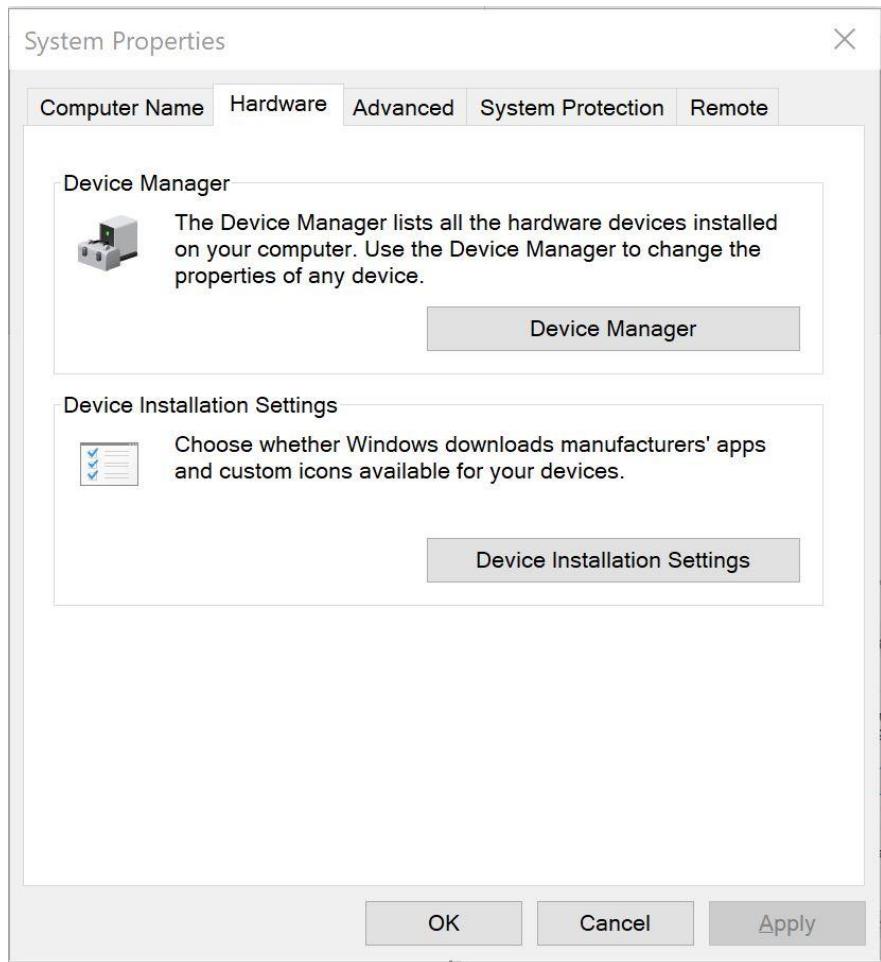
- Serial Peripheral Interface (SPI) เป็นการสื่อสารแบบชิงโครนัสระยะใกล้ เช่น การสื่อสารระหว่างโปรเซสเซอร์กับอุปกรณ์รอบข้าง
- ประกอบด้วยส่วนของมาสเตอร์และสเลฟ
- ความเร็วของการสื่อสารสูงสุด ที่ 80 MHz



- Master Out, Slave In (MOSI) เป็นสายสัญญาณข้อมูล เพื่อการสื่อสารจากมาสเตอร์ไปยังสลิฟ โดยการเขียนของข้อมูลจะซิงโครain ซึ่งกับสัญญาณนาฬิกา
- Master In, Slave Out (MISO) เป็นสายสัญญาณข้อมูล เพื่อการสื่อสารจากสลิฟไปยังมาสเตอร์
- Serial Clock (SCLK) เป็นสัญญาณนาฬิกา เพื่อซิงโครain ซึ่งกับอุปกรณ์ภายในระบบ โดยความเร็วของสัญญาณนาฬิกาเป็นตัวกำหนดความเร็วของการสื่อสารข้อมูล
- Slave Select (SS) ใช้กรณฑ์ที่มีการเชื่อมต่อสลิฟมากกว่า 1 ตัว โดยกำหนดให้มีค่าเป็นลอจิก 1

เริ่มต้นอุปกรณ์ + MicroPython

ตรวจสอบหมายเลขคอมพอร์ต (com port)

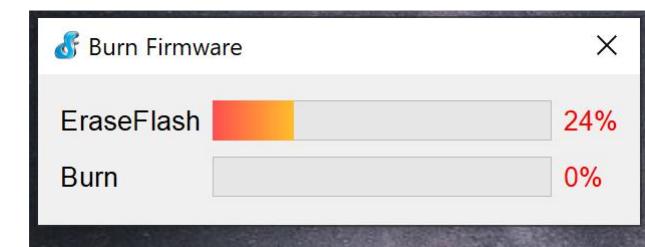
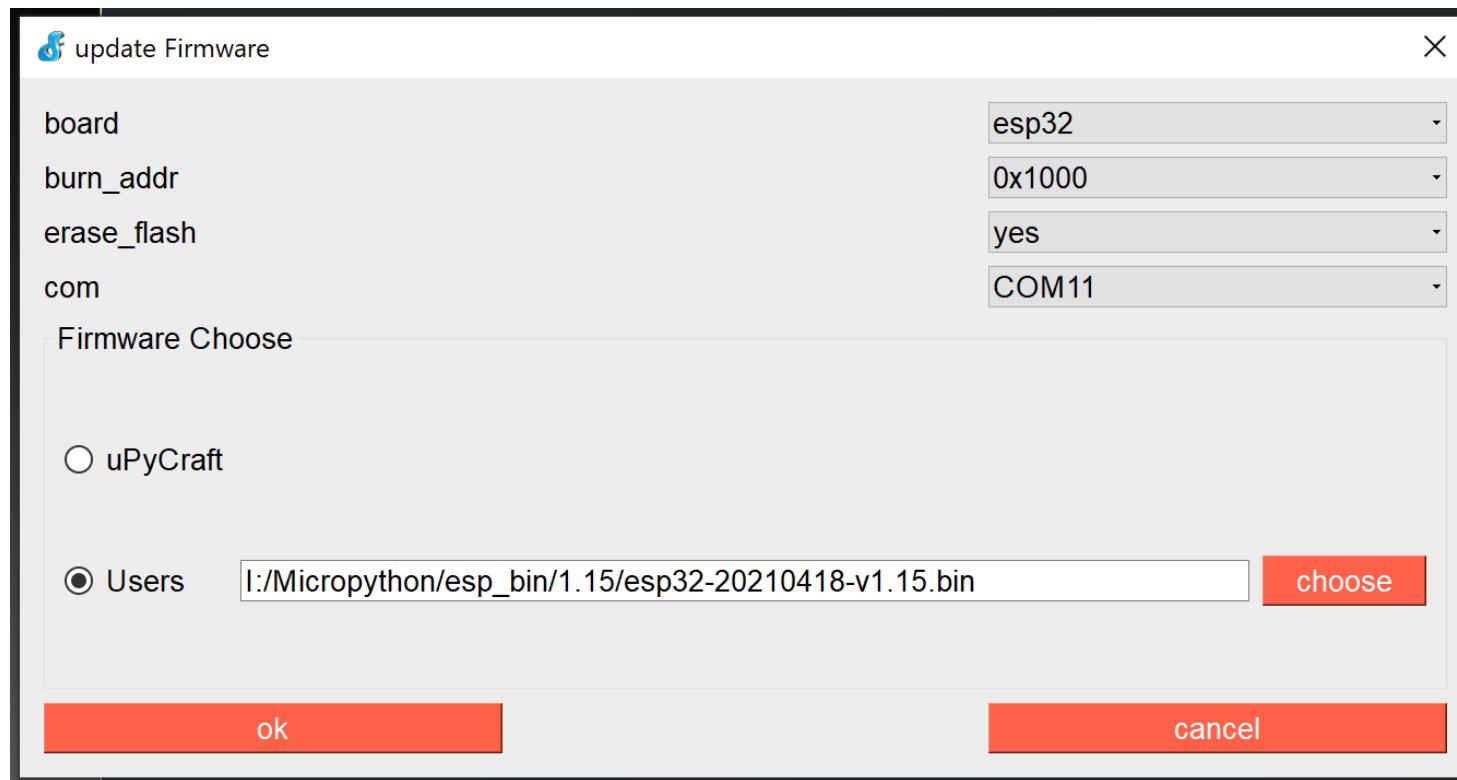


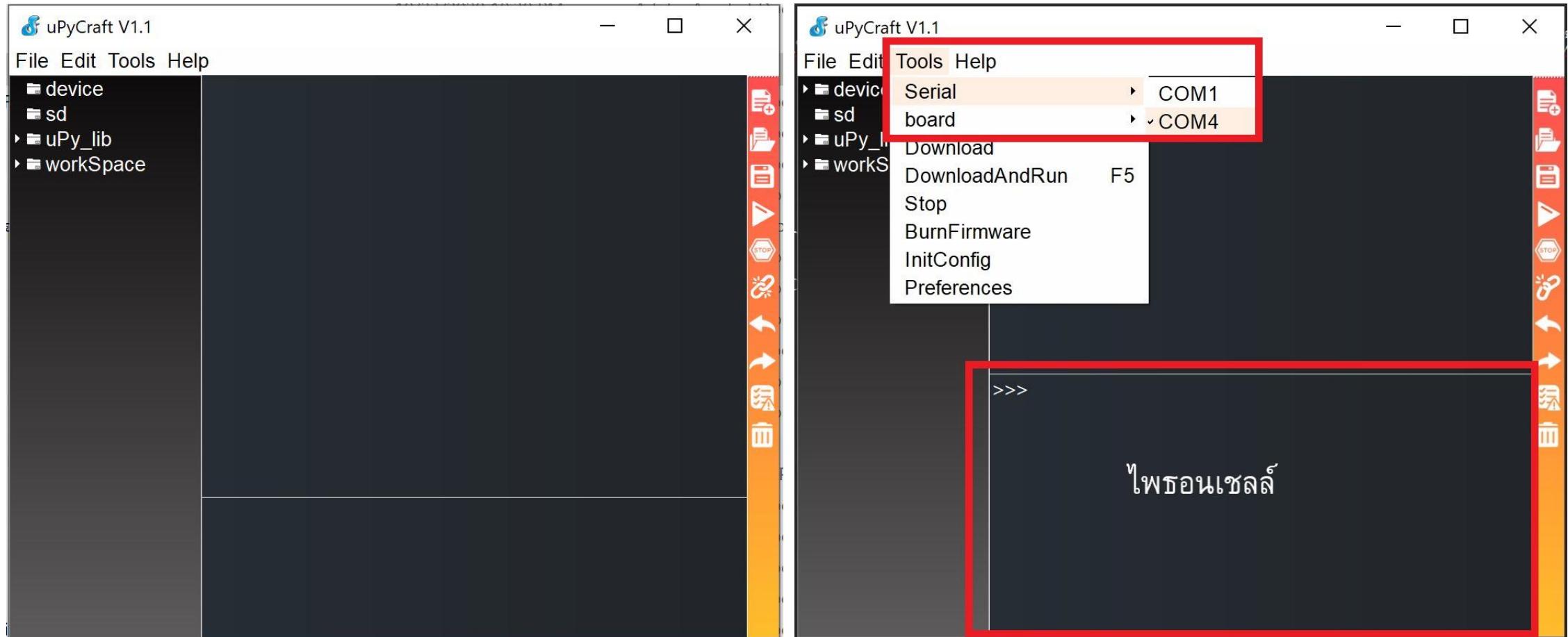
หากไม่พบ หมายเลข com port
ให้ติดตั้งไดร์เวอร์ก่อน

- โปรแกรม uPyCraft IDE สำหรับ Windows



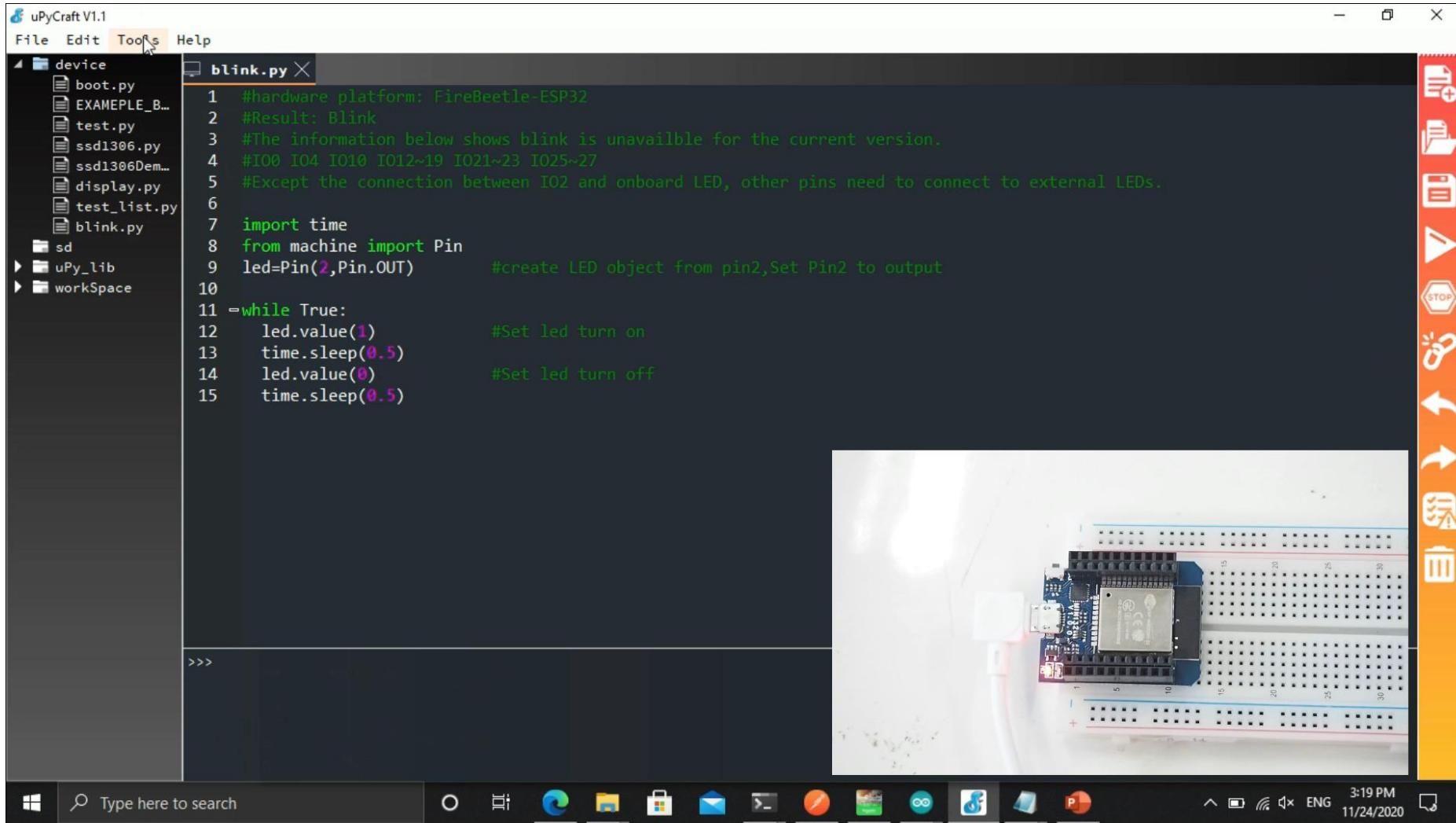
การติดตั้ง Micropython ผ่าน editor





เริ่มต้นใช้งาน

ทดสอบบอร์ดไฟกระพริบ (File → Examples → Basic → **Blink.py**)



The screenshot shows the uPyCraft V1.1 IDE interface. The menu bar includes File, Edit, Tools (which is highlighted in orange), and Help. The left sidebar displays a file tree with categories like device, sd, uPy_lib, and workSpace, containing files such as boot.py, EXAMPEL_B..., test.py, ssd1306.py, ssd1306Dem..., display.py, test_list.py, and blink.py. The main editor window shows the following Python code for the 'blink' example:

```
1 #hardware platform: FireBeetle-ESP32
2 #Result: Blink
3 #The information below shows blink is unavailable for the current version.
4 #IO0 IO4 IO10 IO12~19 IO21~23 IO25~27
5 #Except the connection between IO2 and onboard LED, other pins need to connect to external LEDs.
6
7 import time
8 from machine import Pin
9 led=Pin(2,Pin.OUT)      #create LED object from pin2,Set Pin2 to output
10
11 =while True:
12     led.value(1)        #Set led turn on
13     time.sleep(0.5)
14     led.value(0)        #Set led turn off
15     time.sleep(0.5)
```

The status bar at the bottom shows the Windows taskbar with various pinned icons, the search bar "Type here to search", and system information including the date and time (3:19 PM, 11/24/2020). To the right of the editor is a vertical toolbar with several icons: a red plus sign, a red folder, a red document, a red play button, a red stop button, a red key icon, a red left arrow, a red right arrow, a red gear icon, and a red clipboard icon.

ปรับแรงดัน HW-131 ให้ถูกต้อง



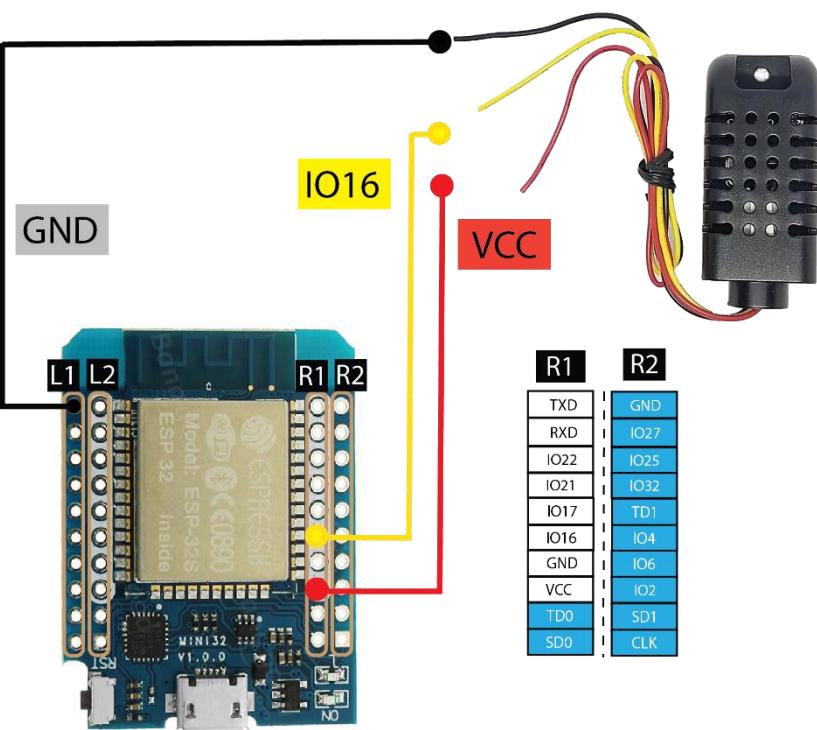
5V สำหรับ ด้านบน



3.3 V สำหรับ ด้านล่าง

ทดสอบอุณหภูมิ ความชื้น

L1	L2
GND	
NC	SVP
SVN	IO26
IO35	IO18
IO33	IO19
IO34	IO23
TMS	IO5
NC	3V3
SD2	TCK
CMD	SD3



กรณีการต่อสายไม่ดี

- ป้อนโค้ดต่อไปนี้

```
1 from machine import Pin
2 import dht
3 sensor = dht.DHT22(Pin(16))
4 sensor.measure()
5 temp = sensor.temperature()
6 humi = sensor.humidity()
7 print("temperature: %3.1f" %temp)
8 print("Huminity: %3.1f" %humi)
```

```
>>> sensor.measure()
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
File "dht.py", line 17, in measure
```

```
OSError: [Errno 110] ETIMEDOUT
```

รีเลย์

ประเภทของรีเลย์

1. รีเลย์หัวไป

2. โซลิดสเตตอรีเลย์ (Solid State Relay)



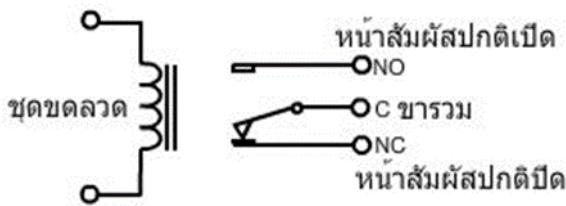
รีเลย์ (Relay)	โซลิดสเตตอรีเลย์ (Solid State Relay)
อายุการใช้งานสั้น เนื่องจากหน้าสัมผัสแบบแมคคานิค	มีอายุการใช้งานนาน
มีเสียงระหว่างการตัดต่อ	ไม่มีเสียงเวลาตัดต่อ
อาจเกิดสัญญาณรบกวน เนื่องจากการตัดต่อ	ไม่เกิดการรบกวน
สามารถตรวจสอบได้ง่าย	เกิดความร้อนการใช้งานเป็นเวลานาน ควรมีการใช้ชีสซิงค์เพื่อบาധความร้อน

ส่วนประกอบของรีเลย์ทั่วไป

1. ขดลวด (Coil)

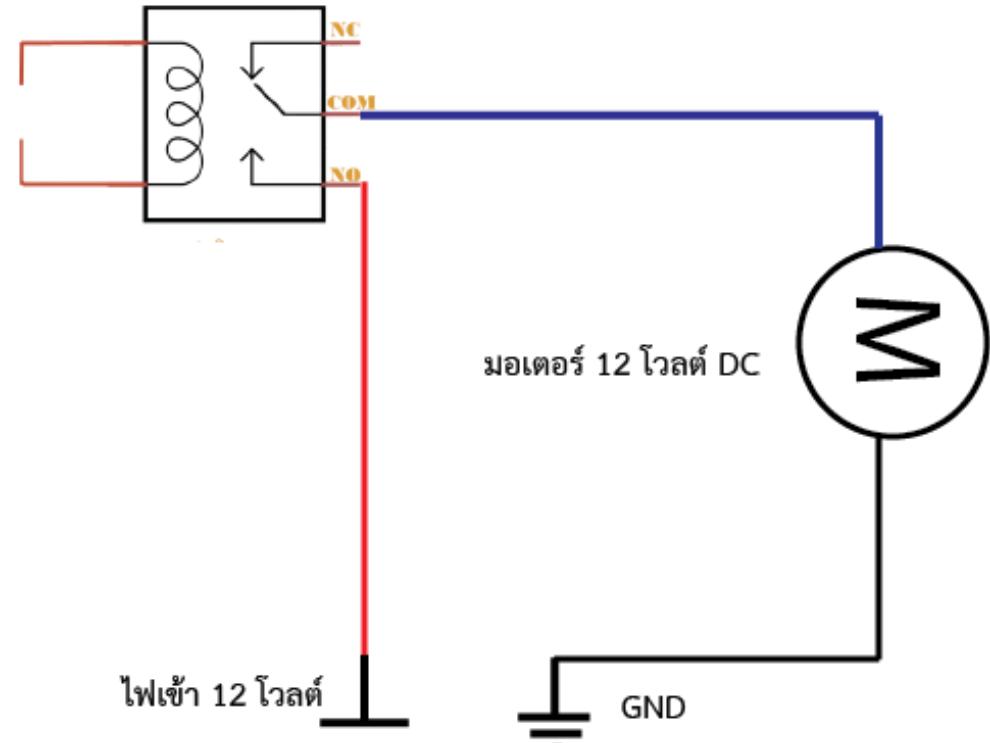


2. หน้าสัมผัส (Contact)

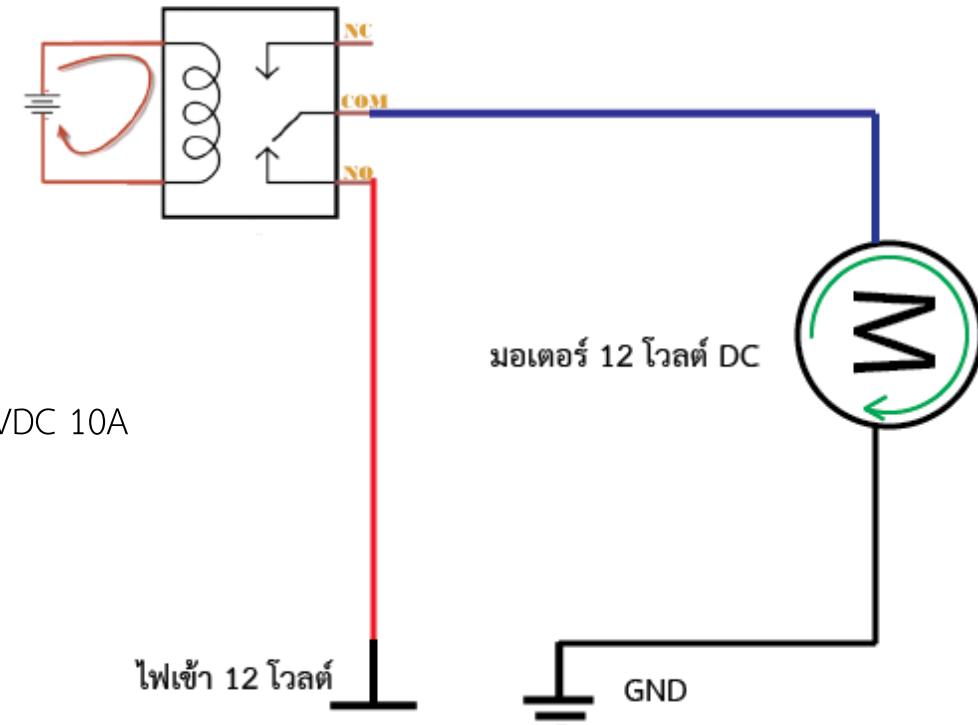


- จุดต่อ NC : จุดต่อ NC ย่อมาจาก normal close หมายความว่า ปกติปิดหรือหากยังไม่จ่ายไฟให้ขดลวดเห็นี่ยวนำหน้าสัมผัสจะติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการให้ทำงานตลอดเวลา
- จุดต่อ NO : จุดต่อ NO ย่อมาจาก normal open หมายความว่า ปกติเปิดหรือหากยังไม่จ่ายไฟให้ขดลวดเห็นี่ยวนำหน้าสัมผัสจะไม่ติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการควบคุมการเปิดปิด
- จุดต่อ C : Common (C) หมายถึง จุดร่วมที่ต่อมาจากการแหล่งจ่ายไฟ

รีเลย์บันบอร์ด



- กระแสเต็มข่านาด 12 โวลต์
- แบบ SPDT
- หน้าสัมผัสข่านาด 120VAC/24VDC 10A
หรือ 250VDC 10A/6A



การเชื่อมต่อรีเลย์



GND
GPIO27
GPIO26
5V

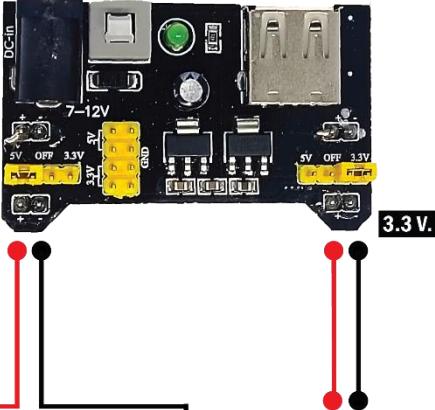
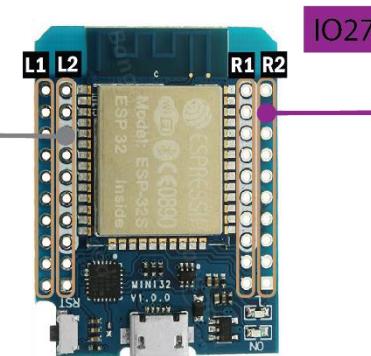


L1	R1
GND	TXD
NC	RXD
SVN	IO22
IO35	IO25
IO33	IO21
IO34	IO17
TMS	IO16
NC	IO4
3V3	IO6
SD2	VCC
CMD	IO2
SD3	TD0

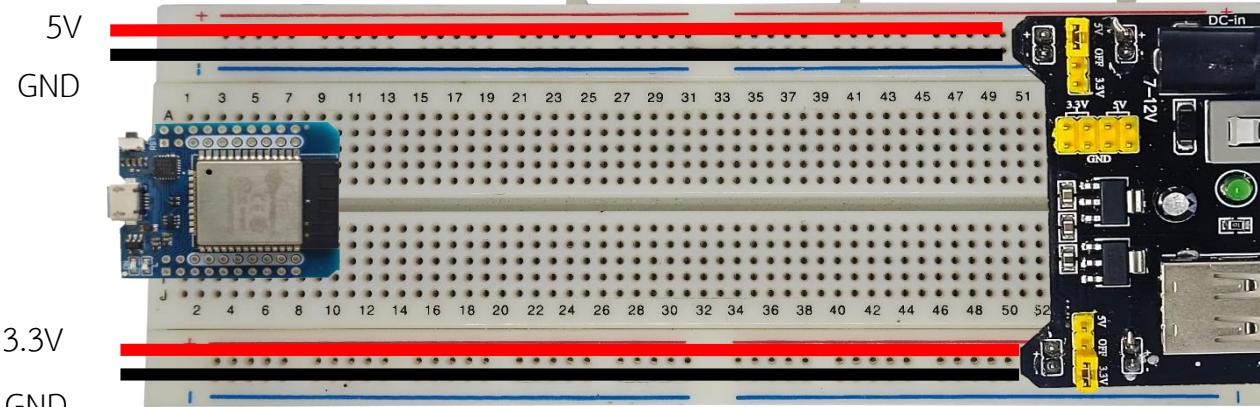


GND IN1 IN2 VCC

IO26



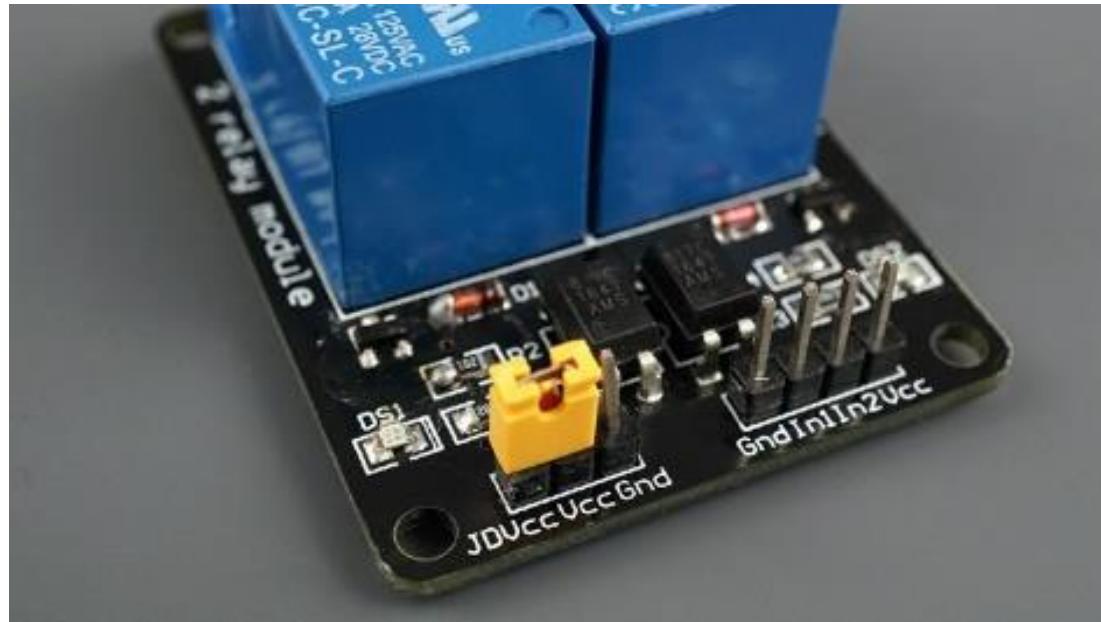
R1	GND
	IO27
	IO22
	IO25
	IO21
	IO17
	IO16
	GND
	IO4
	IO6
	VCC
	IO2
	SD1
	CLK
R2	GND
	IO27
	IO25
	IO32
	TD1
	IO4
	IO6
	VCC
	IO2
	SD1
	CLK



การต่อรีเลย์

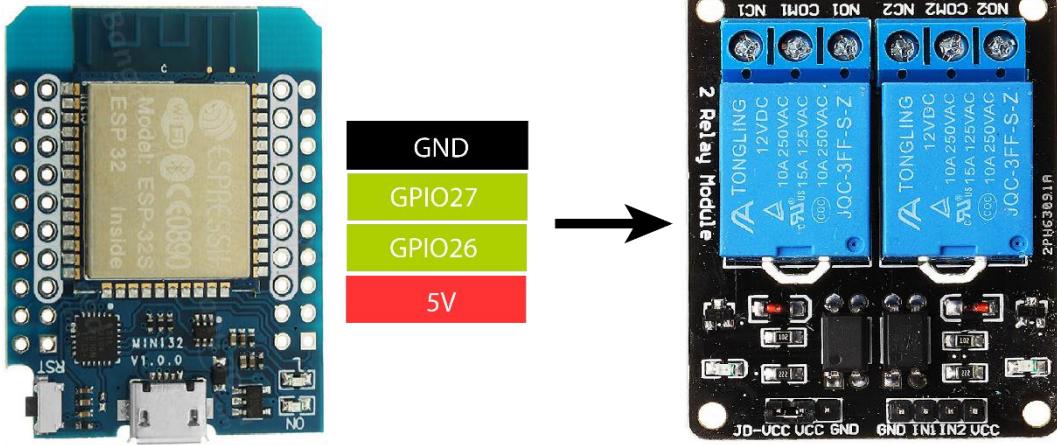
วิธีการต่อสายใช้งานรีเลย์

การจ่ายไฟให้กับรีเลย์



ข	หน้าที่	หมายเหตุ
1	GND	เชื่อมต่อกราวน์ด์ร่วมกับวงจรภายใน
2	VCC	ขาไฟเลี้ยงบนบอร์ดวงจร ร่วมไฟขาเข้า
3	JD-VCC	ขาไฟเลี้ยงเพื่อขับรีเลย์

ทดสอบรีเลย์ 2 ช่องสัญญาณ



from machine import Pin

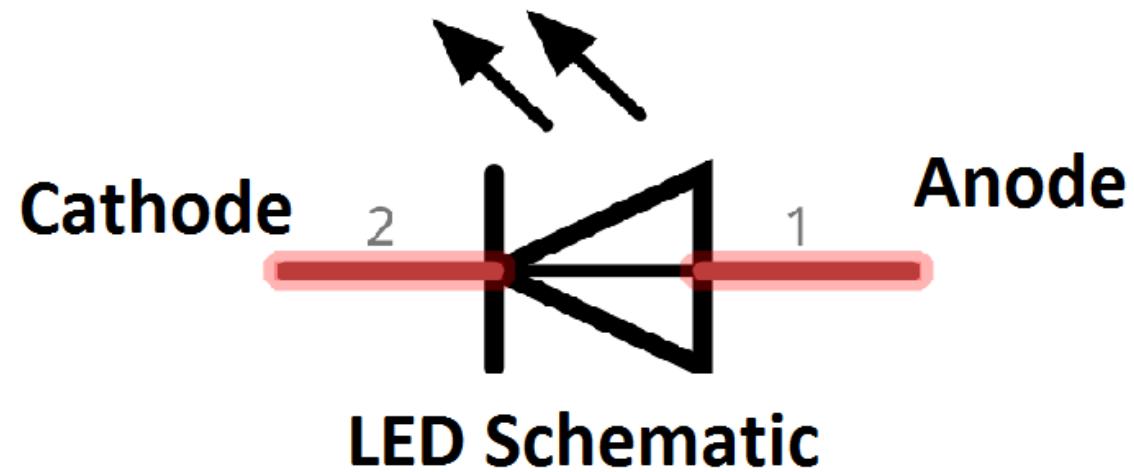
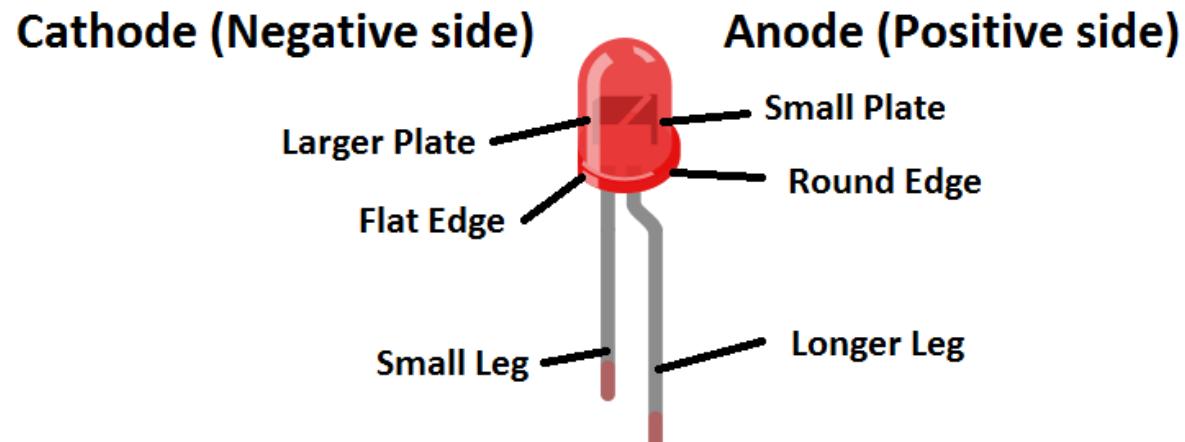
relay1 = Pin(26, Pin.OUT)

relay2 = Pin(27, Pin.OUT)

relay1.value(0) # on

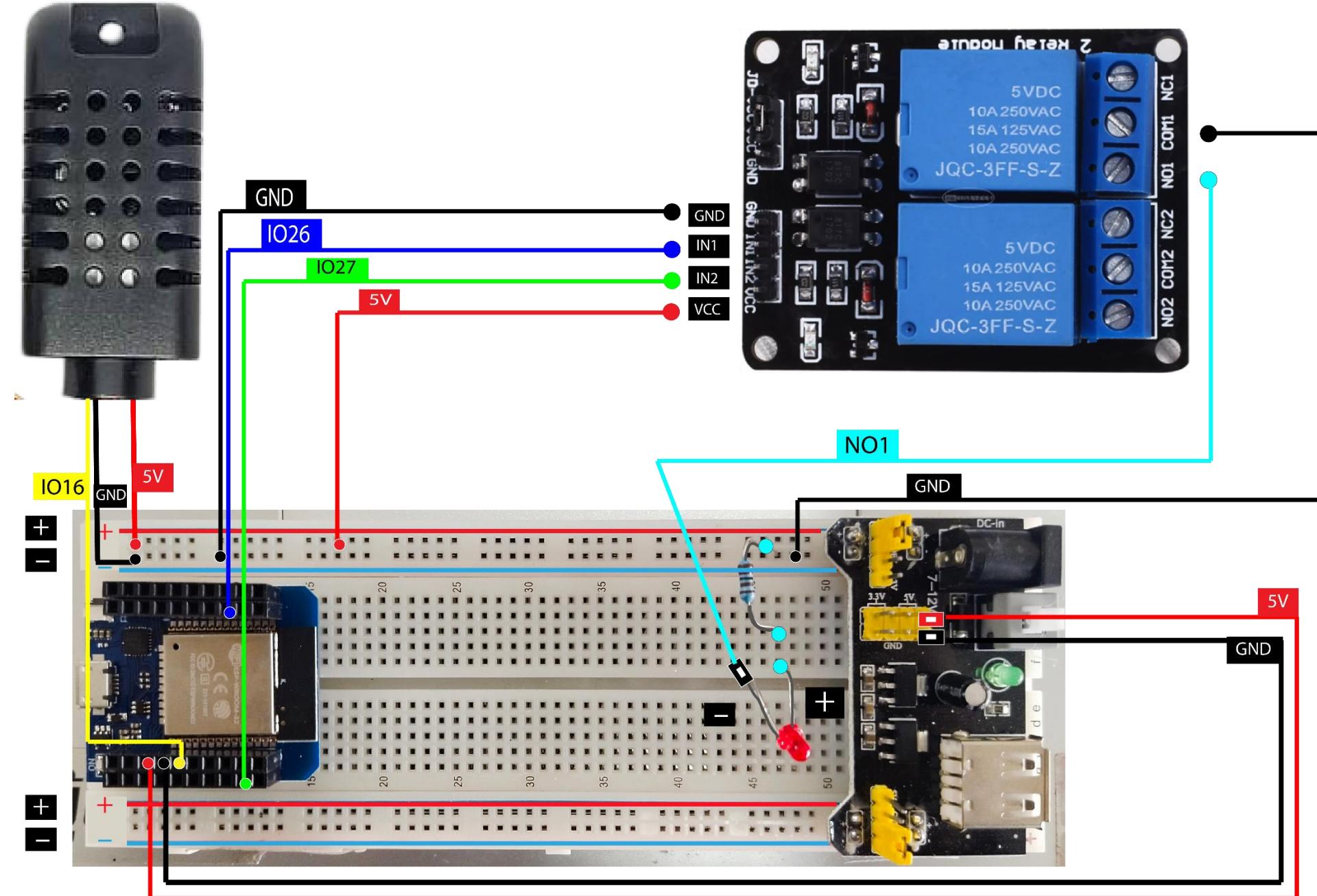
relay1.value(1) # off

LED



ทดสอบห้องมอดรวมกัน

https://github.com/ckboa/smartfarm_training-/blob/main/allsensor_with_relay.py



Wi-Fi

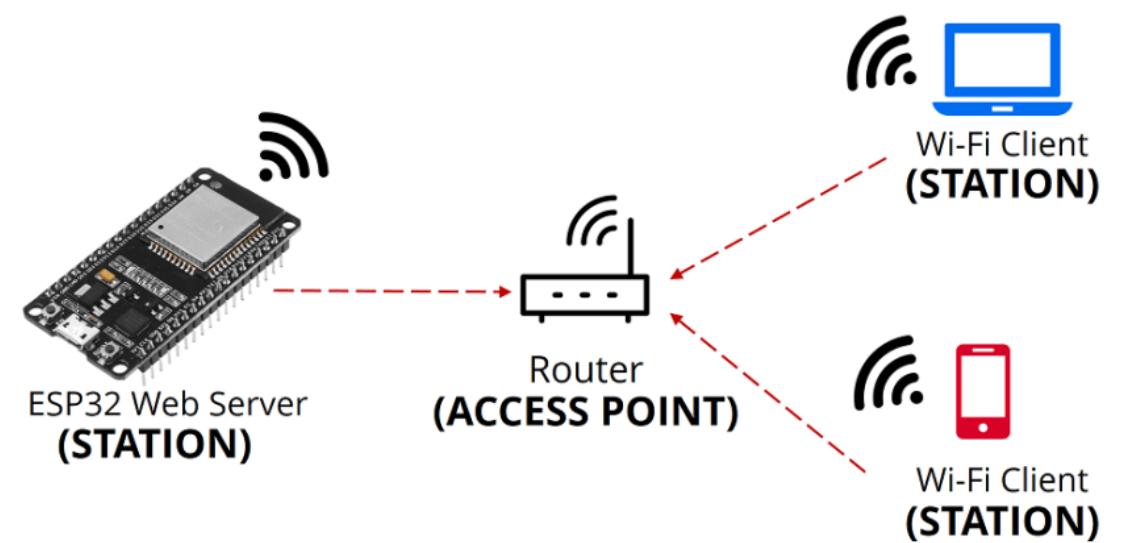


Wi-Fi Mode

Access point mode



Station mode



มาตรฐานที่รองรับ

มาตรฐาน	ฟังก์ชัน
b	รองรับการสื่อสารที่ความถี่ 2.4 GHz ที่อัตราเร็ว 11 Mbps
g	รองรับการสื่อสารที่ความถี่ 2.4 GHz ที่อัตราเร็ว 54 Mbps
n	รองรับการสื่อสารที่ความถี่ 2.4 GHz และ 5 GHz ที่อัตราเร็วมากกว่า 100 Mbps
e	เพื่อให้การสื่อสารใน 802.11 MAC รองรับคุณภาพการให้บริการ (QoS)
i	เพิ่มความปลอดภัยให้ 802.11 MAC และการทำการพิสูจน์ทราบตัวตน

Access Point Mode

```
1 import network
2 ssid = "mc_ap"
3 password = "123456"
4
5 -define create_ap(ssid, password):
6     ap = network.WLAN(network.AP_IF)
7     ap.active(True)
8     ap.config(essid=ssid, password = password)
9
10 -    while ap.active() == False:
11         password
12
13     print(Acess point ready!)
14     ipinfo = ap.ifconfig()
15     print(ipinfo)
16     return ipinfo[0]
17
18 ip = create_accesspoint(ssid, password)
```

- ทดสอบการสร้าง Access Point Mode
- เชื่อมต่อไปยัง Access Point ที่สร้างขึ้น

Station Mode

```
1 import network
2 station = network.WLAN(network.STA_IF)
3 station.active(True)
4 station.connect("<AP_Name>", "<password>")
```

แบบฝึกหัด

ทดสอบการสร้าง Station mode พร้อมตรวจสอบค่าต่าง ๆ ในหน้าถัดไป

- ทดสอบว่ามีการเชื่อมต่อเรียบร้อย

```
>>> station.isconnected()  
True
```

- ตรวจสอบหมายเลข IP address ที่ได้ผลที่จะเป็น IP address, subnet mask, gateway และ DNS

```
>>> station.ifconfig()  
(192.168.0.64, 255.255.255.0, 192.168.0.1, 192.168.0.1)
```

- ทดสอบความแรงของสัญญาณ

```
>>> station.status(rssi)  
-55
```

- การตรวจสอบหมายเลข mac

```
>>> import ubinascii  
>>> ubinascii.hexlify(network.WLAN().config('mac'),':').decode()  
7c:9e:bd:f5:4e:2c
```

บลูทูธ (Bluetooth)

1. บลูทูธคลาสสิก ได้แก่ กลุ่มของบลูทูธตั้งแต่เวอร์ชัน 1 ถึง เวอร์ชัน 3 ซึ่งการพัฒนาจะเน้นเพื่อ การ เข้ามต่อระหว่างอุปกรณ์รอบข้าง และการส่งข้อมูลที่อัตราเร็วที่สูงขึ้น

- เวอร์ชัน 2 ปีค.ศ. 2004 สามารถส่งข้อมูลที่ความเร็ว 3 Mbps จากนั้นใน
- เวอร์ชัน 3 ปีค.ศ. 2009 ได้ปรับปรุงให้สามารถส่งข้อมูลสูงถึง 24 Mbps

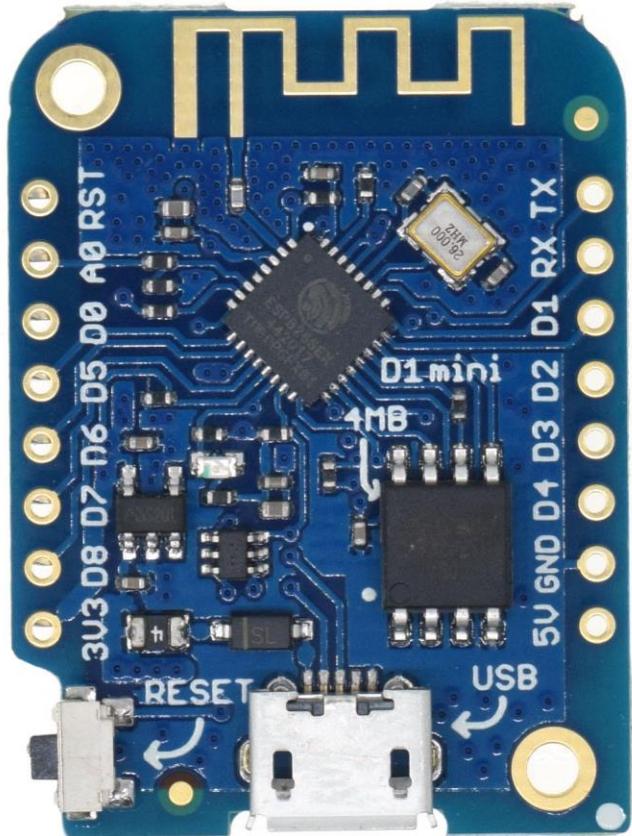
2. บลูทูธพลังงานต่ำ (Bluetooth Low Energy, BLE) ได้แก่ กลุ่มของบลูทูธตั้งแต่เวอร์ชัน 4.0 ปี ค.ศ. 2010 เพื่อให้สามารถทำงานด้วยแบตเตอรี่ขนาดเล็ก

- นิยมมาใช้ในอุปกรณ์ IoT ในขณะเดียวกันด้วยกระแสความนิยม
- ปัจจุบันอยู่ระหว่างการพัฒนาการสื่อสารในรูปแบบเมช (mesh) ในเวอร์ชัน 5

บลูทูธ (Bluetooth)

Feature	บลูทูธคลาสสิค	บลูทูธ 4.x	บลูทูธ 5
ระยะการสื่อสาร	ประมาณ 100	ประมาณ 100	ประมาณ 200
การใช้งานช่องสัญญาณ	ฟรีเควนซีชีอปปิ้ง	ฟรีเควนซีชีอปปิ้ง	ฟรีเควนซีชีอปปิ้ง
อัตราการส่งข้อมูล (Mbps)	1-3	1	2
ความหน่วง (ms)	<100	<6	<3
เน็ตเวิร์ค拓扑ology	Piconet, Scatternet	Star-bus, Mesh	Star-bus, Mesh
การสื่อสารมากกว่า 1 ชอป	Scatternet	Yes	Yes
โปรไฟล์	Yes	Yes	Yes
จำนวนโนดหรือสภาพ	7	Unlimited	Unlimited
ขนาดของเมสเสจ (ไบต์)	Up to 358	31	255

ตัวอย่างโมดูลที่ใกล้เคียงกัน ESP 8266



Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(3.2V Max)
Clock Speed	80/160MHz
Flash	4M Bytes
Size	34.2*25.6mm
Weight	3g

ข้าการเชื่อมต่อของ ESP 8266

Pin	Function	ESP-8266 Pin	Pin	Function	ESP-8266 Pin
TX	TXD	TXD	D5	IO, SCK	GPIO14
RX	RXD	RXD	D6	IO, MISO	GPIO12
A0	Analog input, max 3.2V	A0	D7	IO, MOSI	GPIO13
D0	IO	GPIO16	D8	IO, 10k Pull-down, SS	GPIO15
D1	IO, SCL	GPIO5	G	Ground	GND
D2	IO, SDA	GPIO4	5V	5V	-
D3	IO, 10k Pull-up	GPIO0	3V3	3.3V	3.3V
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2	RST	Reset	RST