หลักสูตรวิศวกรรมระบบไอโอทีและสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Introduction to Internet of Things 1/2566

การทดลองที่ 1 : การติดตั้งโปรแกรม และใช้งาน NodeMCU Board เบื้องต้น

วัตถุประสงค์

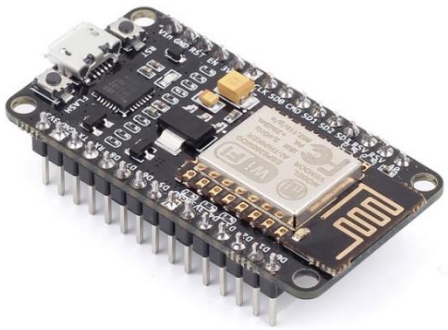
1. เพื่อให้นักศึกษารู้จักการใช้งานบอร์ด NodeMCU เบื้องต้น
2. เพื่อให้นักศึกษารู้ลำดับขั้นตอนในการติดตั้งโปรแกรม และไลบลารีเพื่อใช้งาน NodeMCU
3. เพื่อให้นักศึกษาทบทวนการใช้งาน Arduino IDE ที่ได้เรียนมา และใช้งานคำสั่งเพื่อทดลองกับบอร์ด NodeMCU เบื้องต้น

บทนำ

Internet of Things หรือ IoT “อินเตอร์เน็ตในทุกสรรพสิ่ง” หมายถึง การที่สิ่งต่าง ๆ ถูกเชื่อมโยงสู่โลกอินเตอร์เน็ต บางครั้งอาจเรียกว่า “M2M” ย่อมาจาก Machine to Machine คือเทคโนโลยีอินเตอร์เน็ตที่เชื่อมต่ออุปกรณ์กับเครื่องมือต่างๆ เข้าไว้ด้วยกัน ทำให้มนุษย์สามารถสั่งการ ควบคุม หรือมอนิเตอร์การใช้งานอุปกรณ์ต่าง ๆ ผ่านทางเครือข่ายอินเตอร์เน็ต เช่น การเปิด-ปิด อุปกรณ์เครื่องใช้ไฟฟ้า รถยนต์ โทรศัพท์มือถือ เครื่องมือสื่อสาร เครื่องมือทางการเกษตร อาคาร บ้านเรือน เครื่องใช้ในชีวิตประจำวันต่าง ๆ ผ่านเครือข่ายอินเตอร์เน็ต เป็นต้น

เทคโนโลยี IoT มีความจำเป็นต้องทำงานร่วมกับอุปกรณ์ประเภทเซ็นเซอร์ Sensors ซึ่งเปรียบเสมือนการเติมสมองให้กับอุปกรณ์ต่าง ๆ ที่ขาดไม่คือการเชื่อมต่ออินเตอร์เน็ตเพื่อให้อุปกรณ์สามารถรับส่งข้อมูลถึงกันได้ เทคโนโลยี IoT มีประโยชน์ในหลายด้าน แต่ก็มาพร้อมกับความเสี่ยง เพราะหากระบบรักษาความปลอดภัยของอุปกรณ์และเครือข่ายอินเตอร์เน็ตไม่ดีพอ ก็อาจทำให้มีผู้ไม่ประสงค์ดีเข้ามาขโมยข้อมูลหรือละเมิดความเป็นส่วนตัวของเราได้ ดังนั้นการพัฒนา IoT จึงจำเป็นต้องพัฒนามาตรการ และระบบรักษาความปลอดภัยไอทีควบคู่กันไปด้วย ในปัจจุบันนี้ ปี 2016 ภาพของ Internet of things (IoT) ได้มีเทคโนโลยีเกิดขึ้นมามากมาย เช่น การสื่อสารไร้สาย, real-time analytics, machine learning และ ที่สำคัญที่สุดคือ embedded system ซึ่งปัจจุบันฝังอยู่ในอุปกรณ์ไฟฟ้ามากมาย และกำลังจะต้องถูกพัฒนาให้รองรับ Internet of things (IoT) ในเวลาอันใกล้นี้

1. การใช้งานบอร์ด NodeMCU : ESP8266



NodeMCU คือแพลตฟอร์มหนึ่งที่เป็นการรวมกันระหว่าง **Microcontroller + WiFi Module** ราคาถูก เตัวชิปของ ESP นั้นผลิตโดยบริษัท Espressif System และตัวโมดูลที่เราใช้ ๆ กันในท้องตลาด ผลิตกันจาก Vendor หลายเจ้าตั้งแต่ ESP-01 จนไปถึง ESP-12E และมีการดัดแปลงไปตามแต่ละบริษัทที่ผลิตกันออกไป เช่น NodeMCU Dev Kit v1.0 หรือ NodeMCU โดยในการทดลองนี้จะใช้งานที่ NodeMCU เป็นหลัก อย่างไรก็ตาม หากนักศึกษามีบอร์ดอื่น ๆ ที่ไม่ใช่ NodeMCU ก็สามารถใช้งานได้ ไม่แตกต่างกัน ซึ่งสามารถใช้ไลบลารีที่คล้ายกันและรันบนซอฟต์แวร์ Arduino IDE ได้ NodeMCU เป็นแพลตฟอร์มโดยใช้บอร์ด ESP8266 โดยเป็นตัว ESP-8266EX ที่เพิ่มส่วนของ USB Serial สำหรับติดต่อ USB เพิ่มภาคจ่ายไฟเรกูเลต และขยายขาให้ต่อทดลองได้ง่าย เหมือน Arduino สามารถเขียนโค้ดโดยใช้ Arduino IDE ได้เหมือนบอร์ด Arduino ทั่วไปทุกประการ ESP8266 เป็นไมโครคอนโทรลเลอร์ที่ผลิตโดยบริษัท Espressif (เซี่ยงไฮ้, ประเทศจีน) มีคุณสมบัติเด่นคือการเชื่อมต่อ Wi-Fi ที่มาพร้อมกับ Full TCP/IP มีราคาถูกเป็นอย่างมาก

ESP8266 เป็นสิ่งที่ตอบสนองต่อการมาของยุค Internet of Things จึงทำให้ได้รับความนิยมอย่างแพร่หลาย มีโปรเจคออกมามากมาย

ใน ESP8266 สามารถใช้งานฟังก์ชั่นได้แบบเดียวกับ Arduino ปกติ คือ pinMode, digitalRead และ digitalWrite เช่นถ้าต้องการอ่านค่า GPIO2 สามารถใช้คำสั่ง digitalRead(2)

1.1 คุณสมบัติทั่วไปของ ESP8266

- SDIO 2.0, SPI, UART
- 32-pin QFN package
- Integrated RF switch, balun, 24dBm PA, DCXO, and PMU
- Integrated RISC processor, on-chip memory and external memory interfaces

- Integrated MAC/baseband processors
- Quality of Service management
- I2S interface for high fidelity audio applications
- On-chip low-dropout linear regulators for all internal supplies
- Proprietary spurious-free clock generation architecture
- Integrated WEP, TKIP, AES, and WAPI engines

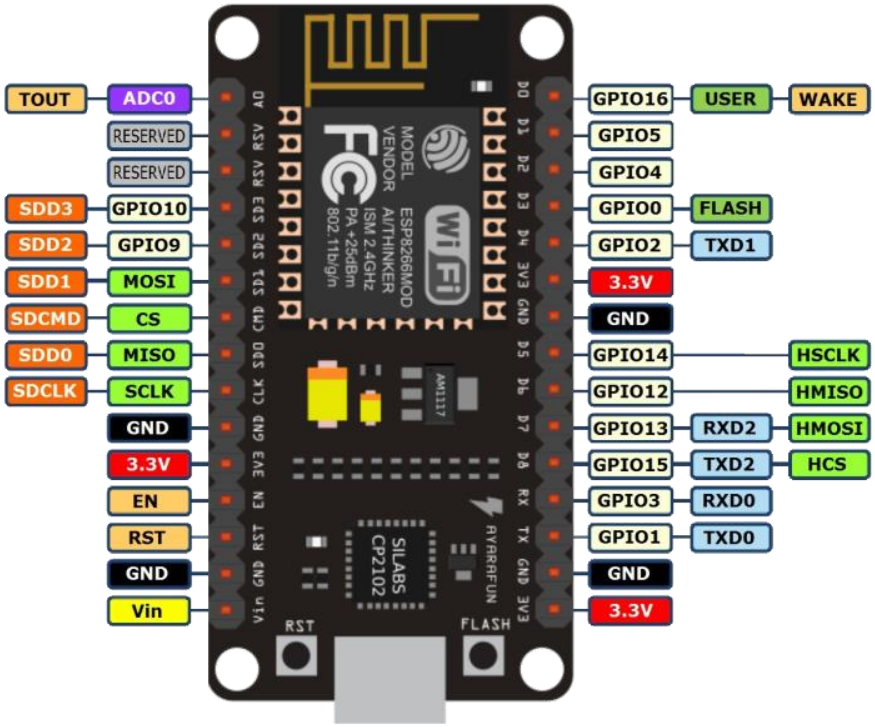
- **SPECIFICATION**
  - 802.11 b/g/n
  - WiFi Direct (P2P), soft-AP
  - Integrated TCP/IP protocol stack
  - Integrated TR switch, balun, LNA, power amplifier and matching network
  - Integrated PLLs, regulators, DCXO and power management units
  - +19.5dBm output power in 802.11b mode
  - Power down leakage current of <10uA
  - Integrated low power 32-bit CPU could be used as application processor
  - SDIO 1.1/2.0, SPI, UART
  - STBC, 1×1 MIMO, 2×1 MIMO
  - A-MPDU & A-MSDU aggregation & 0.4ms guard interval
  - Wake up and transmit packets in < 2ms
  - Standby power consumption of < 1.0mW (DTIM3)

ข้อควรระวัง

1. ชิป ESP8266 ใช้แรงดันไฟสูงสุดที่ 3.6V ทำให้ Logic ที่ใช้ได้คือ 3.3V เท่านั้น
2. หากใช้กับอุปกรณ์ที่ใช้แรงดันไฟ 5V ควรต่อ Logic Level Converter หรือตัวแปลงแรงดัน

1.2 การใช้งานพอร์ตต่าง ๆ ของ NodeMCU

สำหรับบอร์ด NodeMCU มีรายละเอียดของแต่ละขา ดังนี้



รูปที่ 1.1 ขาต่าง ๆ ของ NodeMCU ESP8266

ตารางที่ 1 NodeMCU Port Details and GPIO Map

Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0,ADC
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13

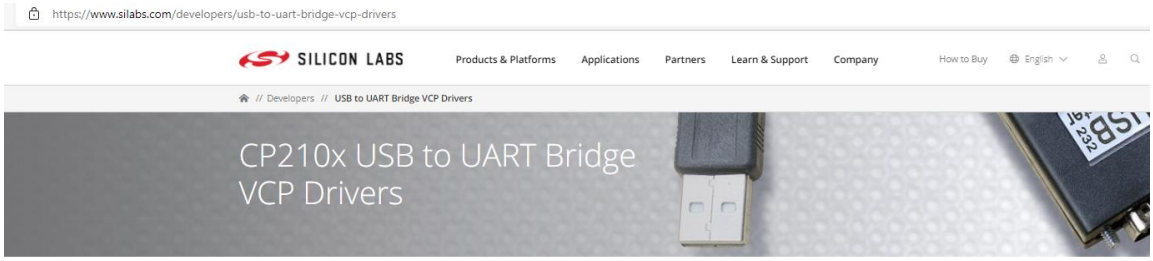
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
3V3	3.3V	3.3V
RST	Reset	RST

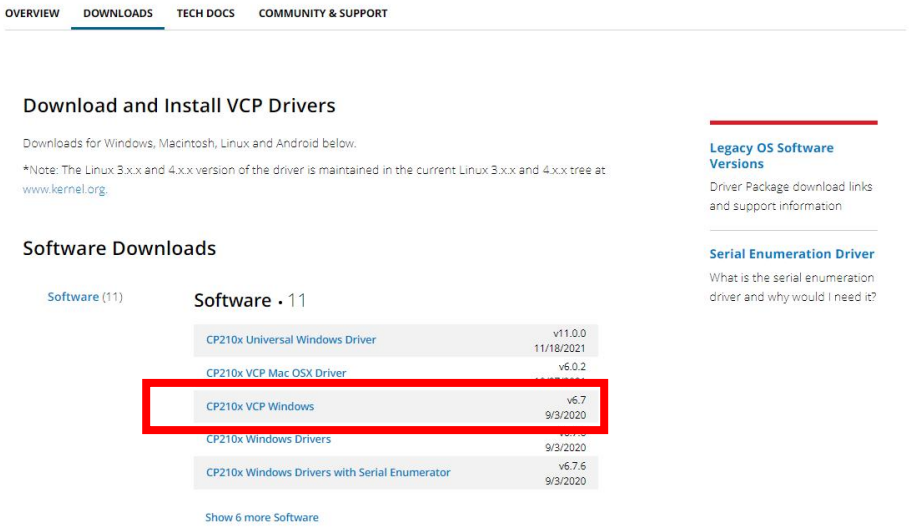
1.3 การติดตั้งไดร์เวอร์สำหรับ NodeMCU

สำหรับบอร์ด NodeMCU อาจมีไดรฟ์เวอร์เป็นสองเวอร์ชัน คือ CH34X และ CP210X ขึ้นอยู่กับชนิดและเวอร์ชันของบอร์ด NodeMCU ที่ใช้งาน เช่นหากเป็น NodeMCU V2 จะใช้ไดร์เวอร์ CP210X หากเป็น NodeMCU V3 จะใช้ไดร์เวอร์ CH34X และล่าสุดจะมาใช้เวอร์ชัน CH91xx แทน

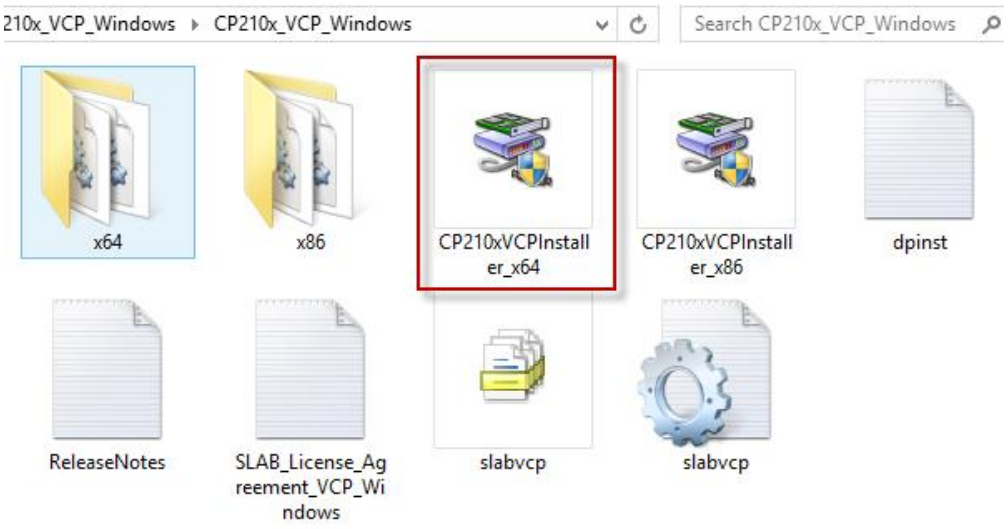
- **ไดร์เวอร์ CP210X / CH91xx**
- https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

เลือก Tab “DOWNLOADS” แอล้วเลือก CP210x VCP windows



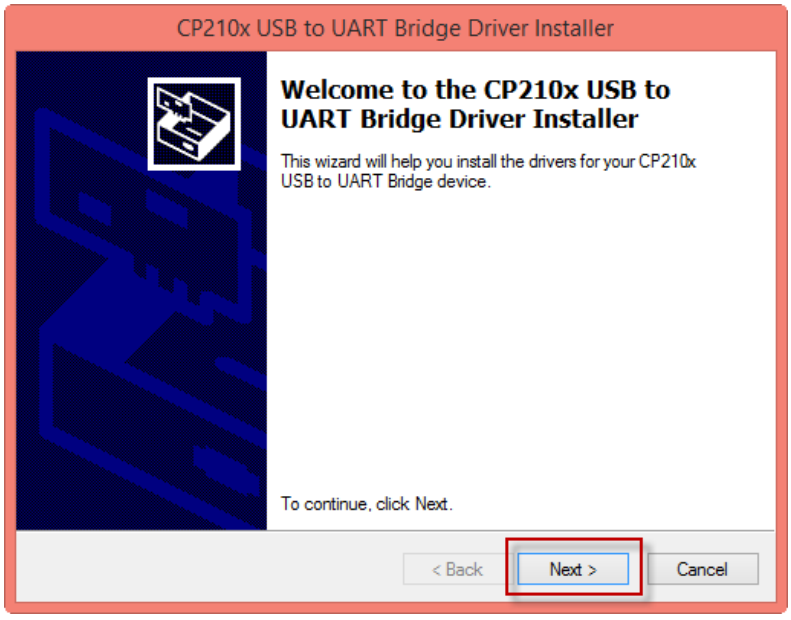


- ดาวโหลดเสร็จ แล้วทำการ unzip ออกมาจะได้โปรแกรมที่สำหรับติดตั้ง



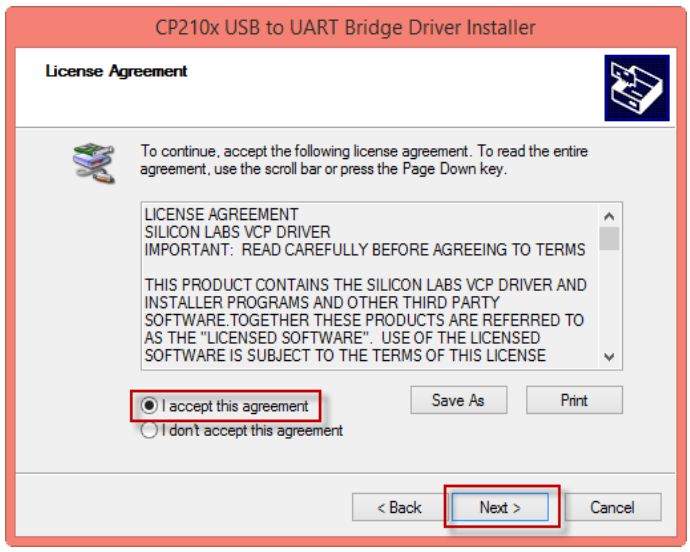
รูปที่ 1.4 หน้าต่างโปรแกรมสำหรับติดตั้งไดร์เวอร์ CP210X

- เปิดโปรแกรม, เสียบสาย USB เข้ากับ NodeMCU และเสียบเข้ากับคอมพิวเตอร์, กด Install จากนั้นกด Next



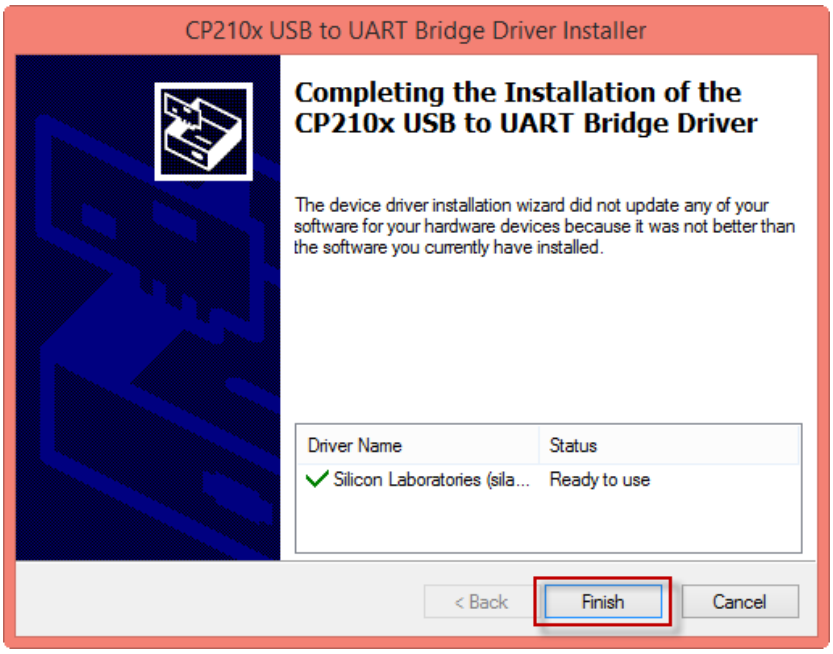
รูปที่ 1.5 การติดตั้งไดร์เวอร์ CP210X

- เลือก I accept this agreement แล้วกด Next



รูปที่ 1.6 การติดตั้งไดร์เวอร์ CP210X

- รอสักครู่ การติดตั้งไดร์เวอร์ก็จะเสร็จเรียบร้อย ดังรูปที่ 11.7 คลิก Finish เพื่อจบการติดตั้ง

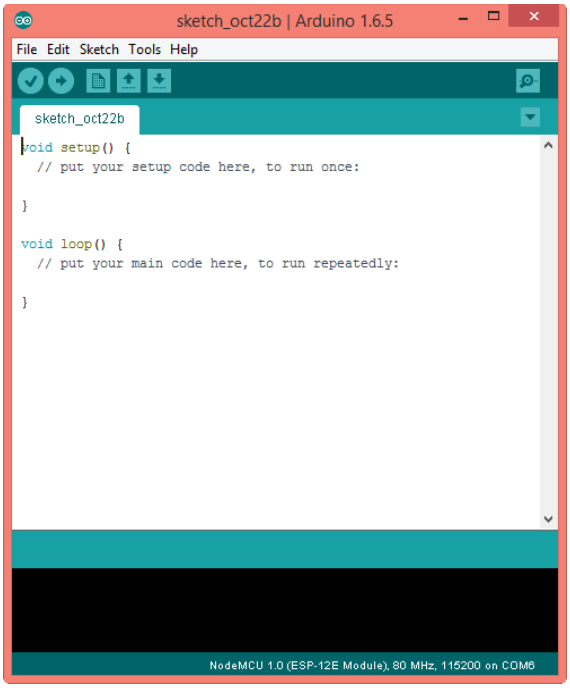


รูปที่ 1.7 การติดตั้งไดร์เวอร์ CP210X เสร็จสมบูรณ์

1.4 การติดตั้งไลบลารีสำหรับใช้งานบอร์ด NodeMCU

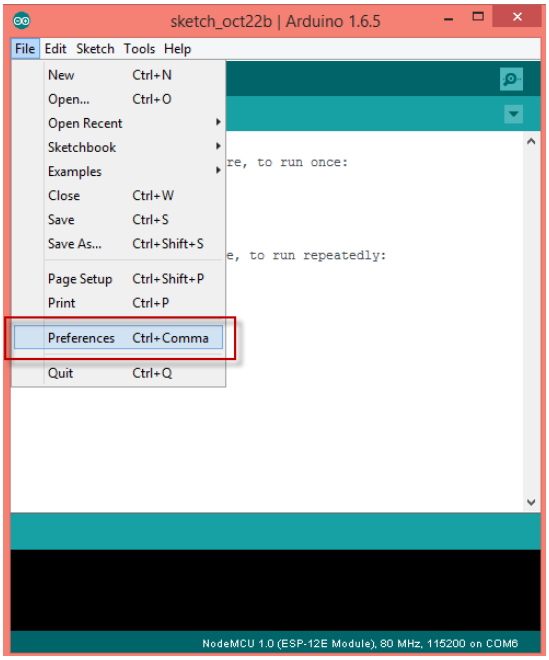
ในขั้นตอนนี้ จะต้องทำการ Install Library ESP8266 ในโปรแกรม Arduino IDE โดยมีขั้นตอนดังนี้

1. เปิดโปรแกรม Arduino IDE สำหรับการเขียนโปรแกรม



รูปที่ 1.8 โปรแกรม Arduino IDE

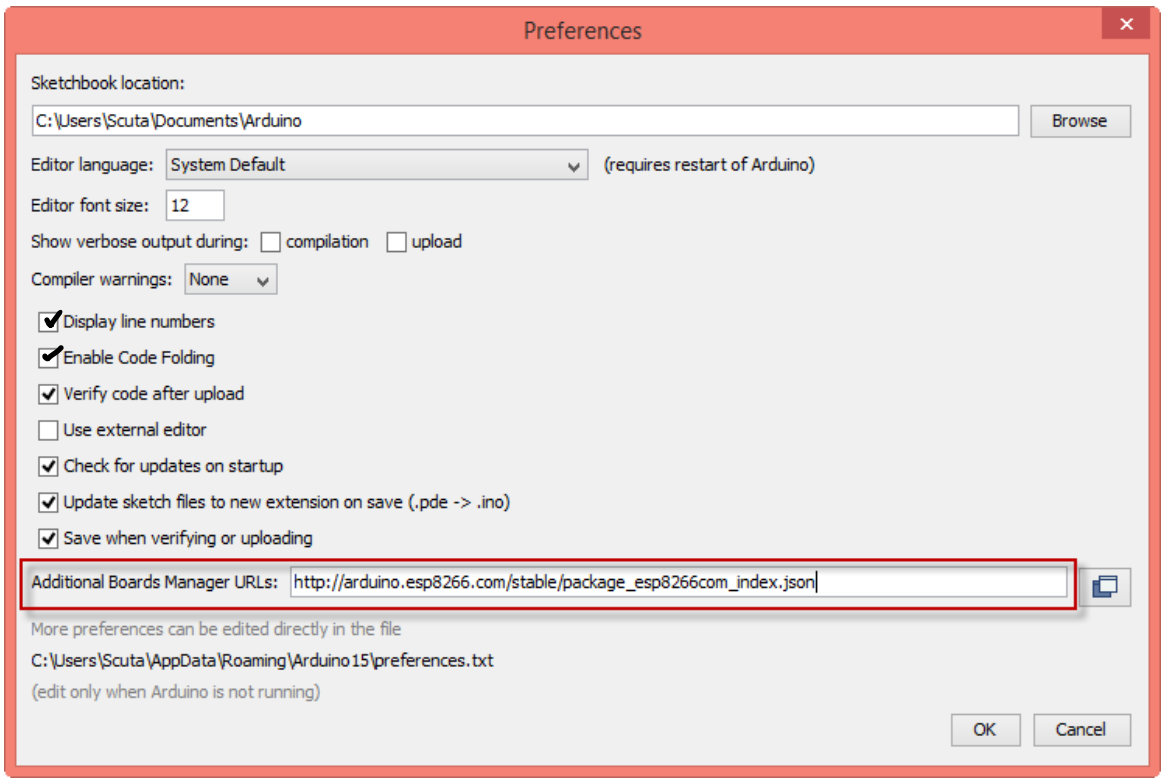
2. ไปที่เมนู File > Preferences



รูปที่ 1.9 การเลือก File -> Preferences

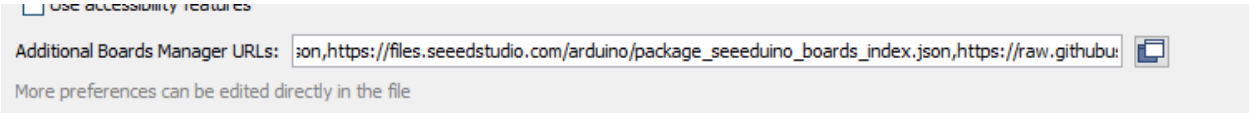
3. ที่ช่อง Additional Board Manager URLs ให้ใส่

http://arduino.esp8266.com/stable/package_esp8266com_index.json จากนั้นกด OK

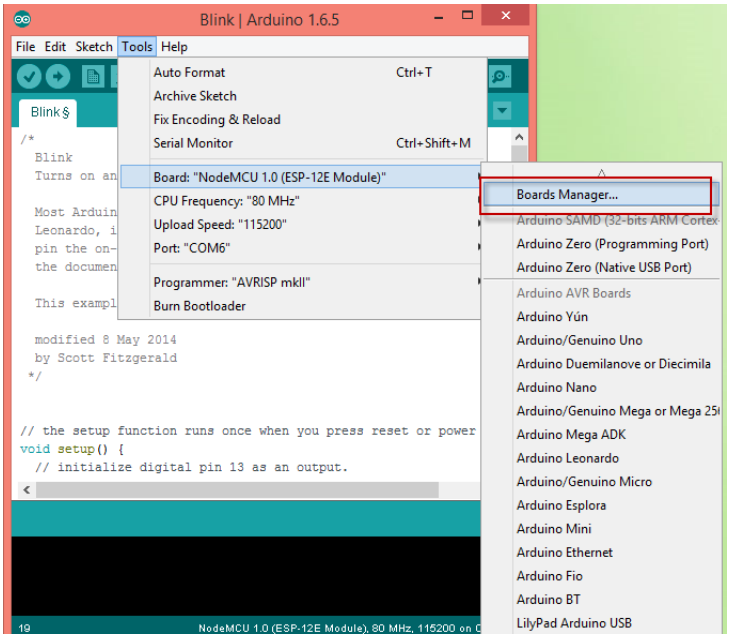


รูปที่ 1.10 การกำหนด Additional Boards Manager

**** หมายเหตุ หากขั้นตอนนี้ ภายใน Additional Board Manager URLs: มีอันเดิมหรือมี Package อื่น ๆ อยู่เราสามารถคั่นด้วย , ได้ เช่น

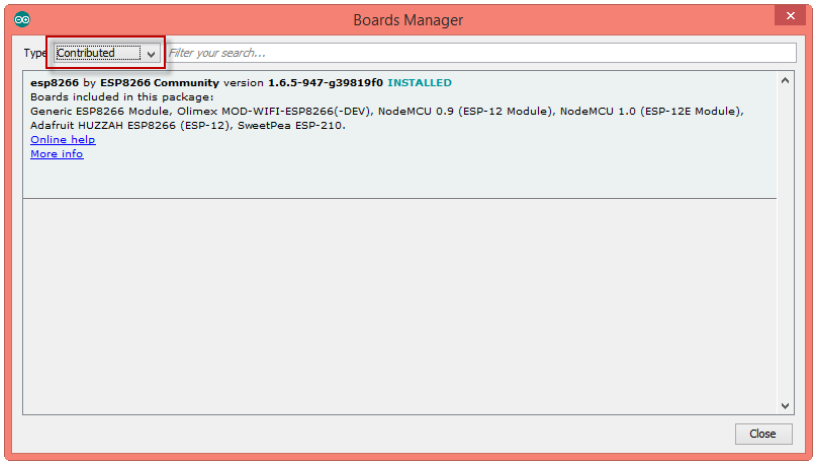


4. ไปที่เมนู Tools > Board > Board Manager



รูปที่ 1.11 การเลือกการจัดการ Boards Manager

5. ที่ช่อง Type เลือก Contributed จากนั้นจะมี esp8266 ให้เรากด Install

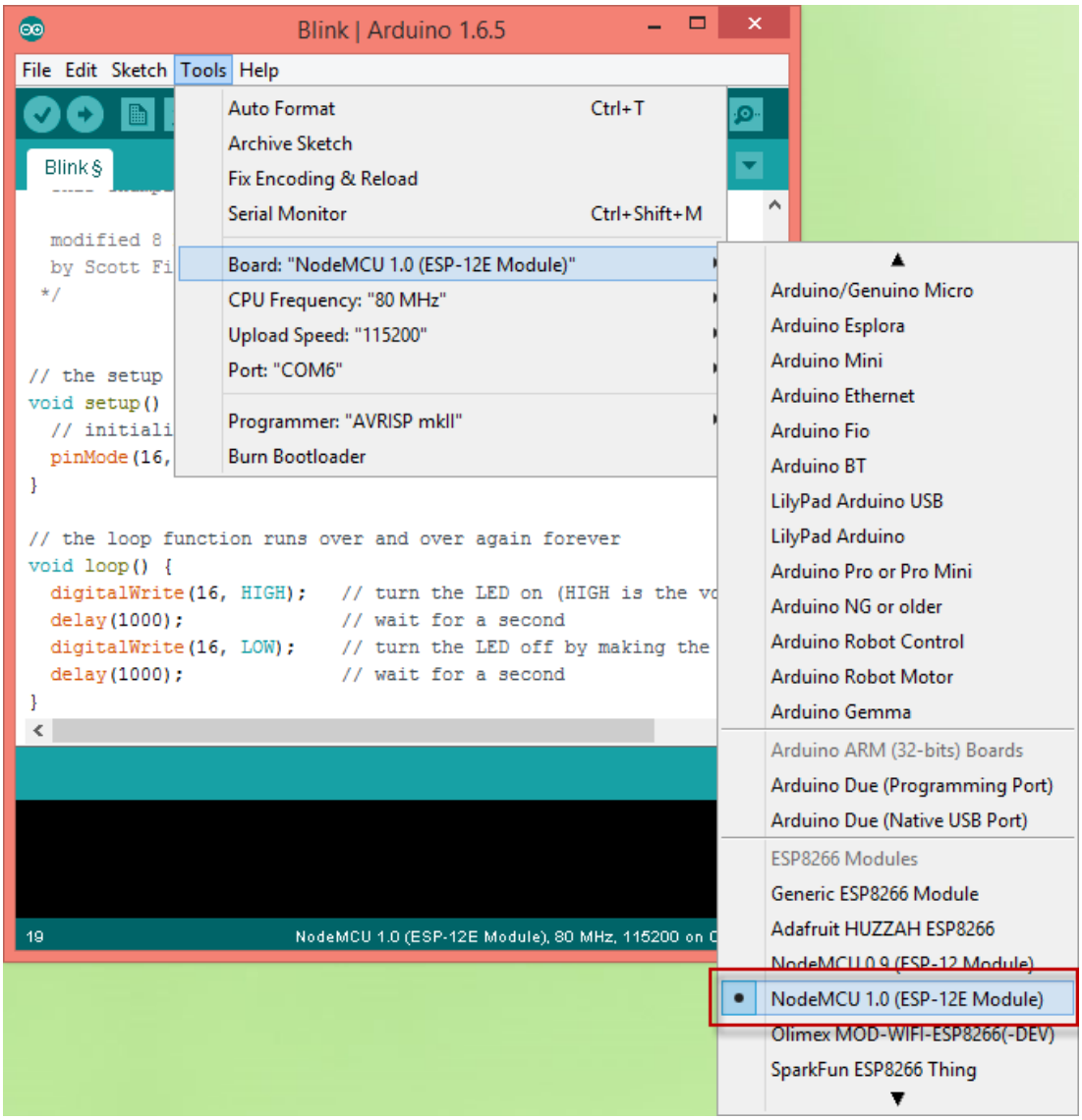


รูปที่ 1.12 การติดตั้ง ESP8266 Library

6. รอจน Install เสร็จ หรือขึ้นว่า Installed

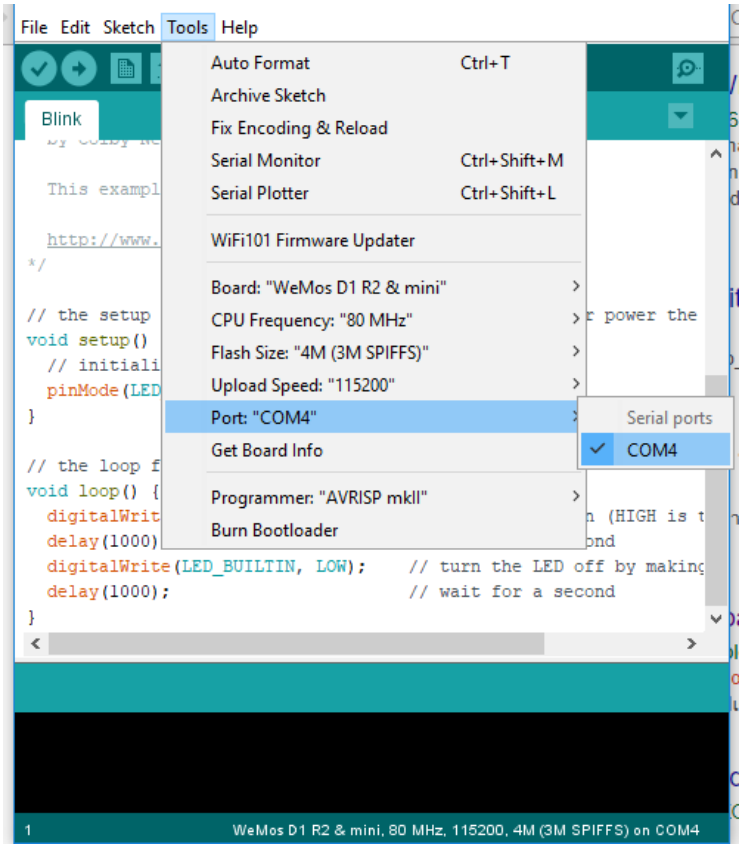
1.5 การทดสอบใช้งาน NodeMCU

- เปิดโปรแกรม Arduino IDE
- เลือกบอร์ด ไปที่ Tools > Board > NodeMCU เลือก NodeMCU 1.0 (ESP-12E Module)



รูปที่ 1.13 การกำหนดบอร์ด NodeMCU ที่จะใช้งาน

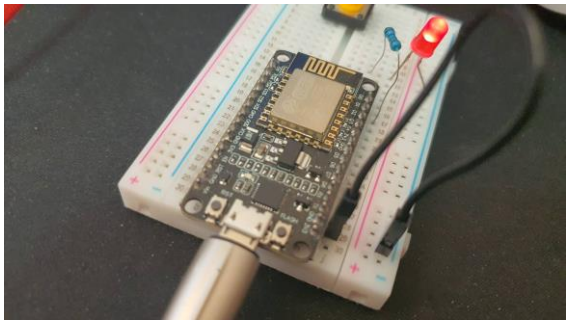
- เลือกพอร์ต Tools -> Port (ในที่นี้คือ COM4)



รูปที่ 1.14 การเลือก PORT ที่ทำการเชื่อมต่อ

- ทำการทดสอบดดลองกับโค๊ด Blink โดยบอร์ด NodeMCU จะมี LED มาให้ 1 ดวงที่ GPIO 16 หรือในที่นี่จะใช้เป็น LED_BUILTIN ซึ่งให้ทำการเลือกดังนี้

  File > Examples > ESP8266 > Blink



เชื่อมต่อบอร์ดกับคอมพิวเตอร์ด้วย Micro USB

- จากนั้นกดปุ่ม Upload เพื่อทดสอบว่าสามารถเบิร์นโค้ดลงบอร์ดได้หรือไม่

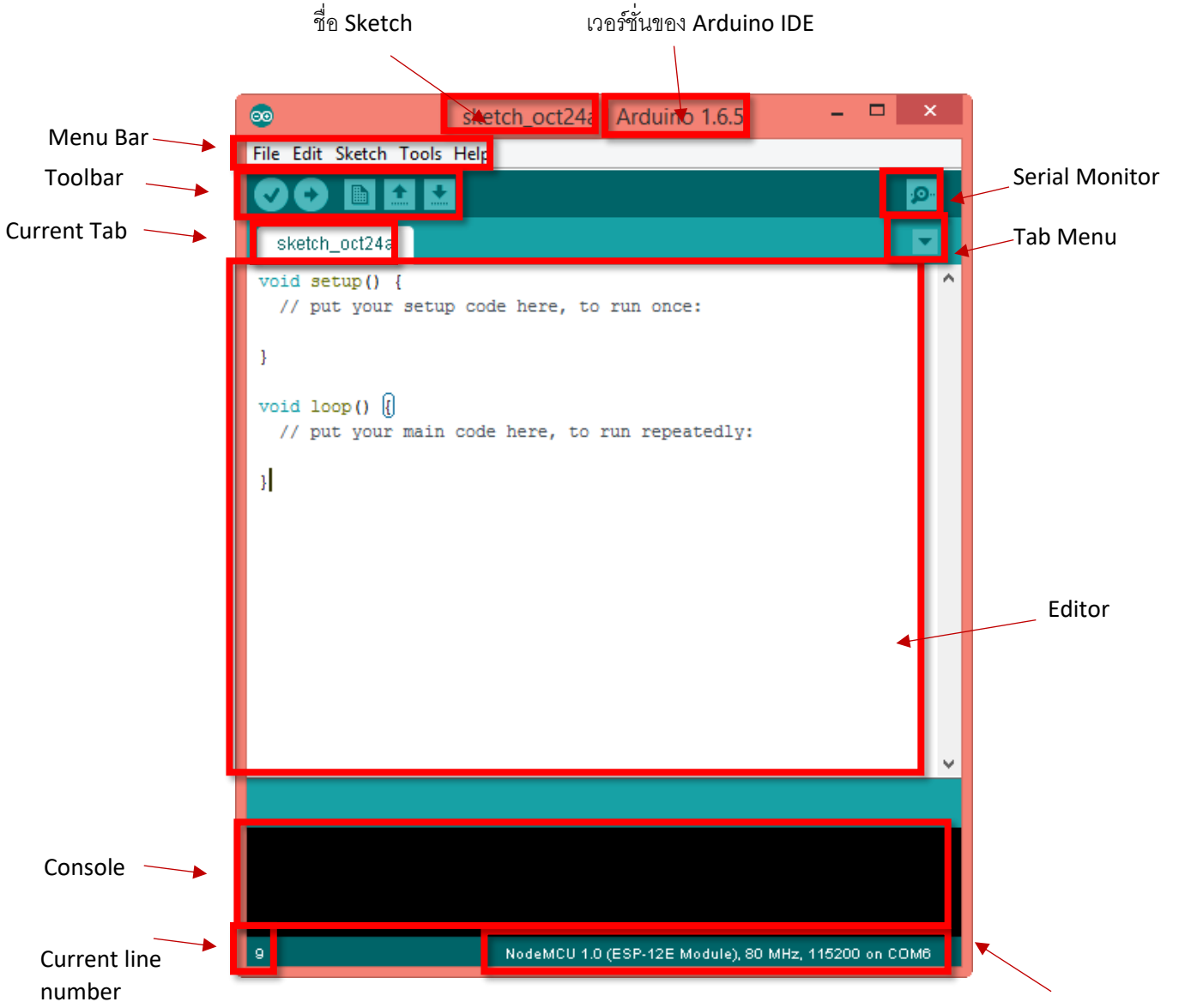
Example 1.1 – BUILTIN LED – Example Test NodeMCU

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
```

การใช้งาน NodeMCU กับ Arduino IDE

ทำการติดตั้งโปรแกรม Arduino IDE, Library ESP8266 และ Driver โดยนักศึกษาสามารถศึกษาได้จากเอกสารเตรียมการก่อนการทดลอง (Laboratory Preparation)

- **ทำความรู้จักกับ Arduino IDE**



- **Toolbar**

Verify

- เช็ค Error ของ Code ว่า Compile ผ่านหรือไม่

Upload

- Compiles และ upload ลงบอร์ด Arduino

 New

- สร้าง Sketch ใหม่

 Open

- เปิด Sketch

 Save

- บันทึก Sketch

โครงสร้างของการเขียนโปรแกรม

ในการเขียนโปรแกรมเพื่อสั่งงาน Arduino Nano นั้นใช้ภาษาซีในการเขียน โดยโครงสร้างของการเขียนโปรแกรมจะคล้ายๆกับการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ทั่วๆไป แต่จะมีความง่ายกว่าเพราะ Arduino ได้มีการรวมรวมคำสั่งและ Library ต่างๆไว้ให้ผู้ใช้สามารถเรียกใช้งานได้เลย ถ้าจะกล่าวถึงการเขียนโปรแกรมด้วยภาษาซีนั้น ฟังชั้นหลักที่จะขาดไม่ได้เลยนั้นคือฟังก์ชัน Main แต่การเขียนโปรแกรมกับ Arduino จะเป็นฟังก์ชัน setup และฟังก์ชัน loop ตามรูป

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

โครงสร้างการเขียนโปรแกรมขั้นต่ำของ Arduino

โดยที่ฟังก์ชันทั้งสองนี้เป็นโครงสร้างที่ขาดไม่ได้ในการเขียนโปรแกรม การทำงานของฟังก์ชันทั้งสองนี้คือ

การทำงานของ void setup()

เป็นฟังก์ชันที่ใช้สำหรับกำหนดค่าเริ่มต้นต่างๆให้กับบอร์ด เมื่อเริ่มต้นทำงาน Arduino จะทำตามคำสั่งต่างๆ อยู่ใน void setup() 1 รอบ ก่อนที่จะเข้าสู่ void loop() ต่อไป

การทำงานของ void loop()

เป็นฟังก์ชันสำหรับสั่งให้ Arduino ทำตามคำสั่งซ้ำๆกัน จะเริ่มต้นทำงานเมื่อผ่านจาก void setup() มาแล้วนั่นเอง

2. การทำงานของสัญญาณ Digital สำหรับ NodeMCU

2.1 คำสั่ง pinMode

สัญญาณแบ่งออกเป็นสองรูปแบบนั่นคือ สัญญาณขาเข้า หรือเรียกว่าสัญญาณอินพุท (input) และสัญญาณขาออก หรือเรียกว่าสัญญาณเอาท์พุท (output) ขาของไมโครคอนโทรลเลอร์แต่ละขานั้น สามารถรับสัญญาณอินพุท และสามารถส่งสัญญาณเอาท์พุทออกมาได้ โดยเราต้องกำหนดโหมดการทำงานของมันเสียก่อน โดยใช้คำสั่ง pinMode() โดย syntax ของการใช้งานคำสั่งนี้เป็นไปดังนี้

```
pinMode(pin,mode);
```

โดย pin เป็นหมายเลขของขาบนบอร์ดไมโครคอนโทรลเลอร์ที่ใช้ในการทดลอง และในส่วนของ mode เป็นการเลือกโหมดการทำงานของขานั้นโดยมีสามรูปแบบคือ INPUT, OUTPUT และ INPUT_PULLUP โดยมีความหมายดังนี้

INPUT	หมายถึงการกำหนดให้ขานั้นทำหน้าที่เป็น input
OUTPUT	หมายถึงการกำหนดให้ขานั้นทำหน้าที่เป็น output
INPUT_PULLUP	หมายถึงการกำหนดให้ขานั้นทำหน้าที่เป็น input แบบที่มีการต่อ internal resistor แบบ pull-up

2.2 คำสั่ง digitalWrite

เมื่อมีการกำหนดให้ขาใดขาหนึ่ง ของไมโครคอนโทรลเลอร์ทำหน้าที่เป็น output ด้วยคำสั่ง pinMode เราจะสามารถใช้คำสั่ง digitalWrite เพื่อเป็นการสร้างสัญญาณเอาท์พุทแบบดิจิตอล โดย syntax ของการใช้งานคำสั่งนี้เป็นไปดังนี้

```
digitalWrite(pin,value);
```

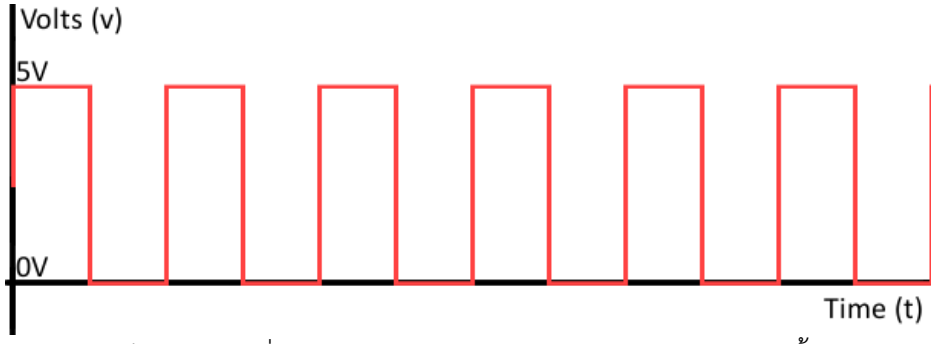
โดย pin เป็นหมายเลขของขาบนบอร์ดไมโครคอนโทรลเลอร์ และ value เป็นค่าของสัญญาณ output ที่เราต้องการให้ไมโครคอนโทรลเลอร์ส่งออกมา มีสองแบบคือ HIGH และ LOW โดยมีความหมายตามตาราง

value	logic	voltage	Pull-up mode
HIGH	“1”	3.3 โวลต์, 5 โวลต์	enable
LOW	“0”	0 โวลต์	disable

แรงดันที่ไมโครคอนโทรลเลอร์ส่งออกมาเมื่อเราสั่งด้วย HIGH นั้นขึ้นอยู่กับรุ่นของบอร์ดไมโครคอนโทรลเลอร์ที่เราใช้งานอยู่ หากเป็นบอร์ดไมโครคอนโทรลเลอร์แบบ 3.3 โวลต์ การสั่ง HIGH จะเป็นการสร้างสัญญาณ 3.3 โวลต์ออกมาที่ขานั้น และหากเป็นไมโครคอนโทรลเลอร์แบบ 5 โวลต์ สัญญาณ HIGH ที่ออกมาจะเป็น 5 โวลต์ การใช้คำสั่ง digitalWrite กับขาที่เลือกโหมดเป็น input จะเป็นการเปิด-ปิด การต่อ pull-up ภายในวงจรของไมโครคอนโทรลเลอร์ โดย HIGH เป็นการเปิดโหมดการต่อ Pull-up และ LOW เป็นการปิดโหมด pull-up

<u>หมายเหตุ</u> การใช้คำสั่ง digitalWrite เพื่อสั่งให้ LED สว่าง โดยที่ไม่ได้กำหนดโหมดการทำงานของขาด้วยคำสั่ง pinMode จะส่งผลให้ LED ที่ต่ออยู่กับขานั้นไม่สว่างเท่าที่ควร เพราะการไม่ใช้คำสั่ง pinMode คือการเปิดการทำงานของโหมด pull-up หรือเปรียบเสมือนมีตัวต้านทานมาขวางการไหลของกระแสไว้

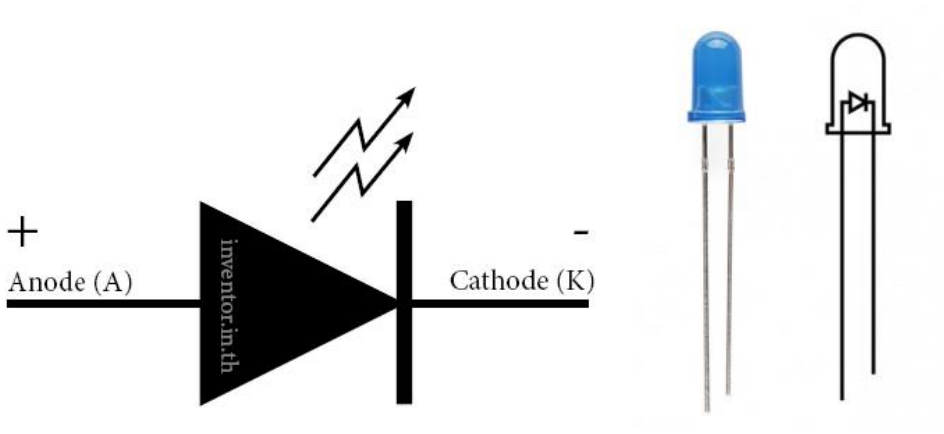
สัญญาณ Digital



เป็นสัญญาณที่มีค่าเพียง 0 และ 1 หรือ LOW และ HIGH เท่านั้น

2.3 แอลอีดี (LED)

LED (อ่านว่า แอล-อี-ดี) ย่อมาจาก Light Emitting Diode หรือ ไดโอดเปล่งแสงได้ ทำหน้าที่ให้แสงสว่างเหมือนหลอดไฟนั้นเอง เพียงแต่ใช้พลังงานที่ต่ำกว่ามาก เมื่อเทียบกับแสงสว่างที่ได้ อีกทั้งยังไม่ร้อนอีกด้วย (ถ้าไม่จ่ายไฟมากเกินไป) โดย**แอลอีดีขายาว จะเป็นขั้วบวก (+)** หรือขั้วแอโนด และ**ขาสั้นเป็นขั้วลบ (-)** หรือแคโทด

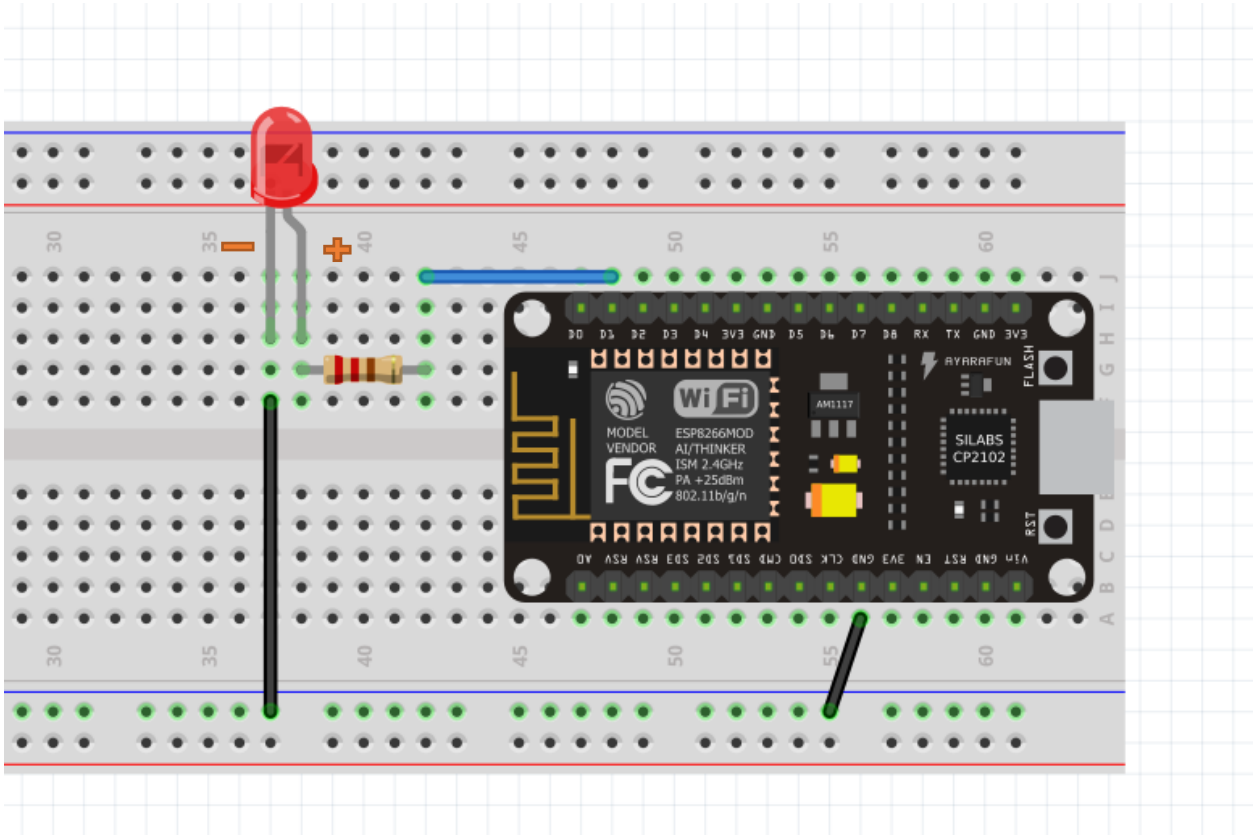


สัญลักษณ์ของแอลอีดี ในวงจรไฟฟ้า

2.4 การทดลอง : พื้นฐานคำสั่ง PinMode และ DigitalWrite

2.4.1 พื้นฐานการใช้งานคำสั่ง digitalWrite

1. ทำการเชื่อมต่อบอร์ด NodeMCU กับ LED โดยต่อขาบวกของ LED กับ D1 และขาลบกับ GND



2. เขียนโค้ด ลงใน Arduino IDE

Example 2.1 Blink – Digital Output (digitalWrite)

```
int pinLED = 5;
```

```
void setup() {
  pinMode(pinLED, OUTPUT);
}
void loop() {
  digitalWrite(pinLED, HIGH);
  delay(1000);
  digitalWrite(pinLED, LOW);
  delay(1000);
}
```

4. จากนั้นทำการเขียนโค้ดต่อมา ลงใน Arduino IDE

Example 2.2 Blink – Digital Output (digitalWrite)

```
int pinLED = D1;
void setup() {
  pinMode(pinLED, OUTPUT);
}
void loop() {
  digitalWrite(pinLED, HIGH);
  delay(1000);
  digitalWrite(pinLED, LOW);
  delay(1000);
}
```

2.4.2 Blynk without Delay (millis())

จากการทดลองในปฏิบัติการที่ผ่านมา ฟังก์ชัน delay เป็นเหมือนการหยุดรอหรือให้อุปกรณ์ถูกแช่แข็งให้หยุดทำงานโดยเท่าจำนวนเลขที่ใส่ไป ซึ่งในการทำงานนั้น โปรแกรมจะไม่ทำงานบรรทัดถัดไป จนกว่าจะถึงเวลา 1 วินาทีต่อไป ทำให้เกิดปัญหา เช่น หากมี LED 3 ดวง ให้กะพริบพร้อม ๆ แต่ติดดับนานไม่เท่ากัน หรือ ใน loop การทำงานจำเป็นต้องอ่านค่าเซ็นเซอร์บางอย่างอย่างต่อเนื่อง เช่น เซ็นเซอร์วัดการเอียง ต้องการความถี่ 50-100 รอบต่อวินาที หากเราเขียนโปรแกรมโดยมี delay บน loop (เช่น ใช้ delay หยุดรอมอเตอร์เลี้ยวจนถึง 90 องศา

เป็นต้น) ก็จะไปขวางการทำงานของส่วนอ่านค่าเซ็นเซอร์ ทำให้จังหวะนั้นเซ็นเซอร์ก็จะหยุดอัพเดทค่า และจะส่งผลต่อการทำงานโดยภาพรวมของระบบโดยรวมแน่นอน

ฟังก์ชัน **millis()** จึงมักถูกใช้แทน delay จากปัญหาข้างต้น เพื่อให้ loop ยังทำงานต่อไปได้โดยไม่ติดขัด (non-blocking) และได้ระยะเวลาที่แม่นยำกว่า

millis() ไม่ใช่ฟังก์ชันหยุดเวลาแต่อย่างใด หากแต่เป็นการ**บันทึกเวลา**ของระบบนับตั้งแต่เปิดเครื่อง millis() เป็นฟังก์ชันที่ return ค่าเวลาเป็นมิลลิวินาที ค่าเวลา ที่นับจากเริ่มต้น เช่นเมื่อระบบเริ่มต้นผ่านไป 1 วินาที ค่าที่จะได้มีค่า 1000 และเมื่อผ่านไปอีก 1 วินาที ค่าที่ได้จะเป็น 2000 และจะเพิ่มขึ้นไปเรื่อยๆ ค่าสูงสุดที่นับได้คือ 4,294,967,295 หรือประมาณ 710 วัน เมื่อค่าเกินจำนวนนี้ เวลาจะกลับไปเริ่มนับจาก 0 อีกครั้ง ฟังก์ชันนี้จึงเหมาะสมอย่างมากที่จะเอามาใช้กำกับฐานเวลาในการพัฒนาโปรแกรม

Example 2.3 millis - example

```
unsigned long period = 1000; //ระยะเวลาที่ต้องการรอ
unsigned long last_time = 0; //ประกาศตัวแปรเป็น global เพื่อเก็บค่าไว้ไม่ให้ reset จากการวนloop
void setup() {
    Serial.begin(9600);
}
void loop() {
 if(millis() - last_time > period) {
     last_time = millis(); //เซฟเวลาปัจจุบันไว้เพื่อรอจนกว่า millis() จะมากกว่าตัวมันเท่า period
     Serial.println("Hello World!");
 }
}
```

ตัวอย่างถัดไปจะเป็นการใช้เพื่อแทนคำสั่ง delay โดยให้ต่อ LED กับบอร์ดดังตัวอย่างที่ผ่านมา แล้วเขียนโค้ดดังนี้

Example 2.4 millis - example

```
const int ledPin =  LED_BUILTIN; // the number of the LED pin
int ledState = LOW;             // ledState used to set the LED
// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;        // will store last time LED was updated
const long interval = 1000;           // interval at which to blink (milliseconds)
void setup() {
```

```
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }
    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

โจทย์การทดลอง

1. เชื่อมต่อกับแอลอีดีจำนวนสามดวง จากนั้นเขียนโปรแกรมเพื่อให้แอลอีดีทุกดวง สว่าง-ดับ พร้อมกัน
2. ให้นักศึกษาเชื่อมต่อแอลอีดีสามดวง จากนั้นเขียนโปรแกรม ให้แอลอีดีแต่ละดวง สว่างแล้วดับ ทีละดวง สลับกันไปทีละดวง จนสนกลับมาดวงแรกเป็นเช่นนี้ไปเรื่อย ๆ

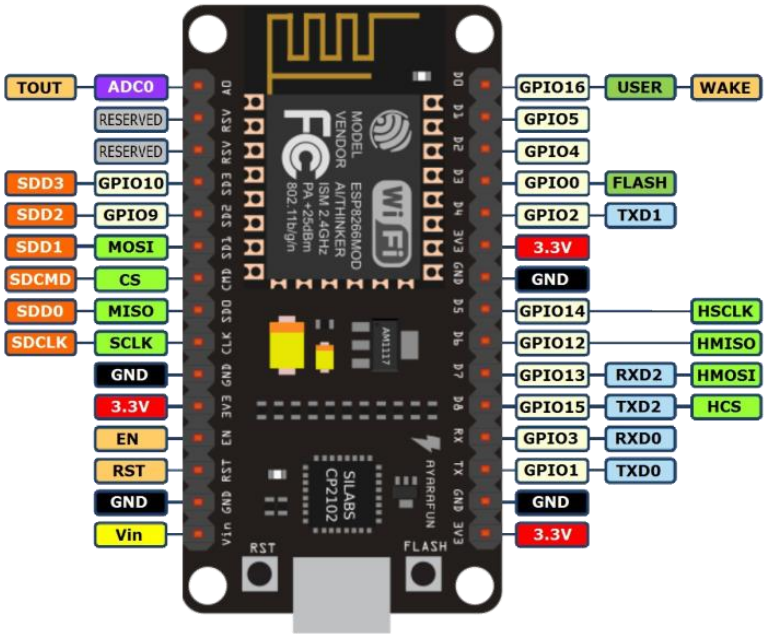
<u>การส่งงาน</u>

** ทำโจทย์การทดลองข้อ 1 และข้อ 2 ให้ TA ตรวจลงคะแนน

Reference

การใช้งานพอร์ตต่าง ๆ ของ NodeMCU

สำหรับบอร์ด NodeMCU มีรายละเอียดของแต่ละขา ดังนี้



รูปที่ 1 ขาต่าง ๆ ของ NodeMCU ESP8266

ตารางที่ 4 NodeMCU Port Details and GPIO Map

Pin	Function	ESP-8266 Pin
TX	TXD	TXD

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