บทที่ 1

ภาพรวมของ NodeMCU

NodeMCU (Node Microcontroller Unit) เป็นแพลตฟอร์มการพัฒนาแบบโอเพ่นซอร์สที่นิยมใช้ในงาน IoT (Internet of Things) โดยมีการสร้างขึ้นจากไมโครคอนโทรลเลอร์ ESP8266 หรือ ESP32 และมีความสามารถในการเชื่อมต่อ Wi-Fi ในตัว ทำให้เหมาะสมกับโครงการที่ต้องการเชื่อมต่อเครือข่ายไร้สาย

1. คุณสมบัติของ NodeMCU

1.1. การเชื่อมต่อ

Wi-Fi ชิป ESP8266 หรือ ESP32 ที่อยู่ใน NodeMCU มีความสามารถในการเชื่อมต่อ Wi-Fi ซึ่งทำให้ NodeMCU เหมาะสำหรับโครงการ IoT ที่ต้องการการเชื่อมต่อกับอินเทอร์เน็ตหรือเครือข่ายท้องถิ่น

1.2. ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์: ESP8266 มีไมโครคอนโทรลเลอร์แบบ 32 บิตที่ความเร็ว 80 MHz (สามารถโอเวอร์คล็อกได้ถึง 160 MHz) ในขณะที่ ESP32 มีประสิทธิภาพที่ดีกว่าโดยมีการทำงานแบบ dual-core ที่ความเร็วสูงสุด 240 MHz

1.3. GPIO Pins

NodeMCU มีขา General Purpose Input/Output (GPIO) หลายขา ซึ่งช่วยให้คุณสามารถเชื่อมต่อกับเซ็นเซอร์, อุปกรณ์กระตุ้น และอุปกรณ์อื่นๆ ได้

1.4. การเขียนโปรแกรม

สามารถเขียนโปรแกรมด้วย Arduino IDE, Lua, หรือ MicroPython ทำให้มีความยืดหยุ่นในการใช้งาน Lua เป็นภาษามาตรฐานของ NodeMCU แต่หลายคนมักใช้ Arduino IDE เพราะความง่ายในการใช้งาน

1.5. หน่วยความจำ

- ESP8266: RAM ที่โปรแกรมได้ 80 KB และหน่วยความจำแฟลช 4 MB
- ESP32: มีหน่วยความจำมากกว่าและประสิทธิภาพที่ดีกว่า ESP8266

1.6. พาวเวอร์ซัพพลาย

สามารถใช้พลังงานจาก USB หรือแหล่งจ่ายไฟ 5V และทำงานที่แรงดันไฟ 3.3V

1.7. TCP/IP Stack

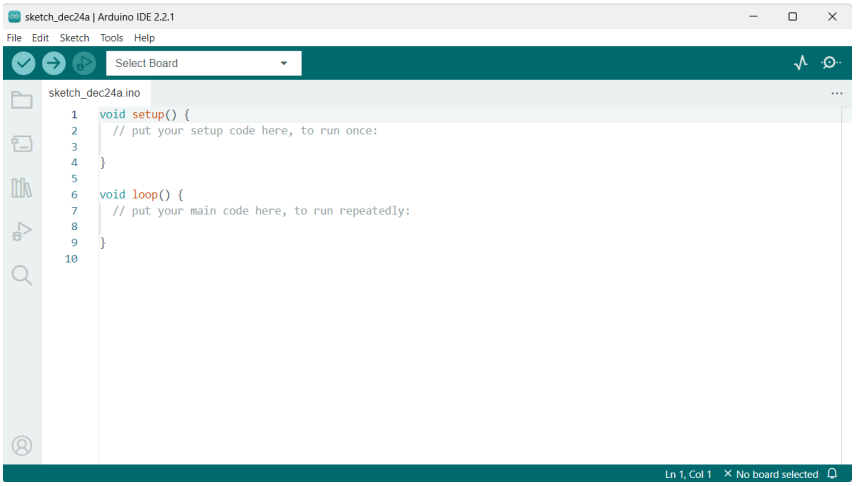
NodeMCU มีโปรโตคอล TCP/IP แบบครบวงจรในตัว ทำให้การสื่อสารเครือข่ายสำหรับงาน IoT ง่ายขึ้น

2. รุ่นการใช้งาน

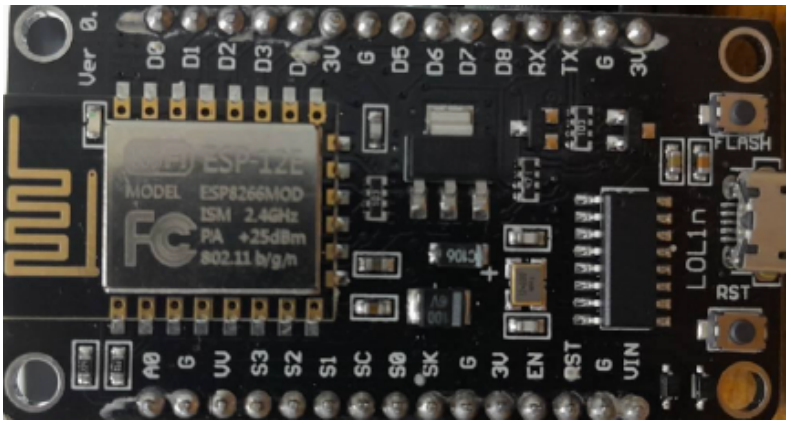
- NodeMCU v1: ใช้ ESP8266 เป็นรุ่นแรกสุด
- NodeMCU v2: ฮาร์ดแวร์ที่ปรับปรุงใหม่พร้อมขนาดที่เล็กลง
- NodeMCU v3: มี GPIO Pins มากขึ้นแต่ขนาดใหญ่กว่า
- NodeMCU-32S: ใช้ชิป ESP32 ที่มีความสามารถและฟังก์ชันเพิ่มเติมเช่น Bluetooth

3. การติดตั้งซอฟท์แวร์และเชื่อมต่อฮาร์ดแวร์

ซอฟแวร์สำหรับพัฒนา Arduino (IDE) เป็นสิ่งจำเป็นสำหรับการใช้งาน Arduino ใช้สำหรับเขียนโค้ดโปรแกรมและอัพโหลดลงไปที่บอร์ด Arduino เราสามารถดาวน์โหนดซอฟท์แวร์ได้จากเว็บไซต์ผู้พัฒนา Arduino url: https://www.arduino.cc/en/software ซอฟท์แวร์ IDE จะมีให้เลือกดาวน์โหลด 3 ประเภทคือ Windows, Mac และ Linux เมื่อติดตั้งโปรแกรมเรียบร้อยแล้ว เราจะเห็นหน้าต่างหลักของโปรแกรมดังรูป

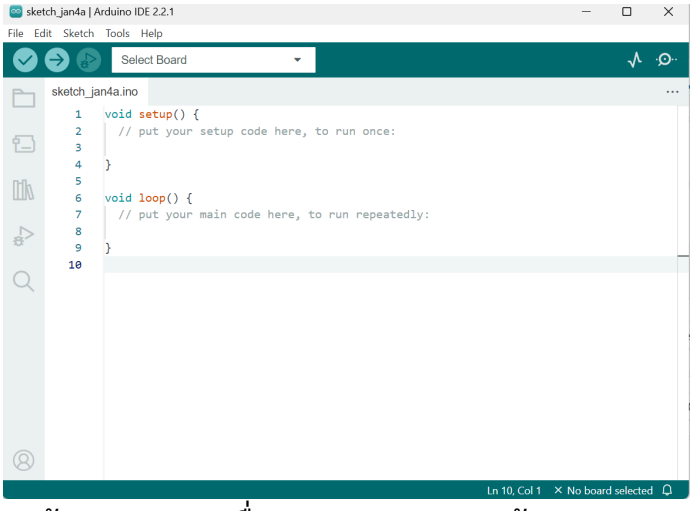


Code ที่เขียนโดย Adruino IDE จะทำงานบนบอร์ด NodeMCU บอร์ด NodeMCU ถูกสร้างเพื่อเชื่อมต่อ sensor และ actuator และ เพื่อสั่ง sensor และ actuator



3.1. การเขียน Sketch บน Arduino IDE

Sketch คือตัวโปรแกรมย่อยบน Arduino IDE ที่เป็นหน้าต่างสำหรับเขียน code คำสั่งเป็นภาษา C โดยจะเป็นไฟล์นามสกุล .ino



ก่อนที่จะ upload จะต้องกด verify เพื่อตรวจสอบความถูกต้องของ code คอมไพล์ code หากกด verify แล้ว กรณีที่ code ไม่ถูกต้อง (คอมไพล์ไม่ผ่าน) ด้านล่างขึ้นเตือนเป็นตัวหนังสือสีแดงและ error massage กรณีนี้จะไม่สามารถ upload ได้ หาก code ถูกต้อง (คอมไพล์ผ่าน) ด้านล่างขึ้น Done compiling กรณีนี้เราจะสามารถ upload โปรแกรมไปยัง Arduino ได้ โดยกด upload

4. โครงสร้างพื้นฐานของโปรแกรม Arduino

โปรแกรม Arduino ประกอบด้วยองค์ประกอบ 3 ส่วน คือโครงสร้างโปรแกรม (structure) ตัวแปร (variable) ค่าคงที่ (constant) ฟังก์ชันคำสั่ง (function)

4.1. โครงสร้างโปรแกรม ประกอบด้วย

- **void setup()** เป็นคำสั่งเริ่มต้นของบอร์ด Arduino โดย Arduino จะถูกดำเนินการ (executed) 1 ครั้งเท่านั้น ภายใน { } ของ void setup นอกจากนี้ยังสามารถตั้งค่าตัวแปร กำหนด PIN เริ่มใช้ Library ฯลฯ
- **void loop()** จะอยู่หลังจาก void setup () เป็นส่วนที่เราต้องการดำเนินการซ้ำคำสั่งใน { } ของ void setup ของ void loop จะถูกดำเนินการซ้ำไปเรื่อยๆ จนกว่าจะปิดอุปกรณ์

  เครื่องหมาย “{ }” เรียกว่าวงเล็บ (bracket) เป็นสัญลักษณ์ที่ใช้ขยายขอบเขตของคำสั่ง ความหมายคือถ้าไม่มี { } คำสั่งจะถูกดำเนินการเฉพาะบรรทัดแรกเท่านั้น ขณะที่ถ้ามี { } คำสั่งภายใน { } ทั้งหมดจะถูกดำเนินการ

4.2. ตัวแปร (variable) ของ Arduino

Arduino ใช้ตัวแปรหลากหลายประเภท โดยปกติ Arduino จะใช้ int (integer, 16 bit) ในการระบุจำนวน แต่ยังสามารถใช้ตัวแปรประเภทอื่นในตารางด้านล่าง

Type	Byte	Range	Meanimg
int	2	-32768 ถึง 32767	จำนวนบวกและจำนวนลบ
unsigned int	2	0 ถึง 65535	เหมือน int แต่เฉพาะจำนวนบวก
long	4	-2147483648 ถึง 2147483647	จำนวนบวกและจำนวนลบขนาดใหญ่
unsigned long	4	0 ถึง 4294967295	จำนวนบวกขนาดใหญ่
float	4	-3.4028235E+38 ถึง 3.4028235E+38	เลขทศนิยม ใช้รับค่าจากการวัดจริง
double	4	เหมือนกับ float	เหมือนกับ float
boolean	1	false (0) หรือ true (1)	ค่าจริงหรือเท็จ
char	1	-128 ถึง 127	ตัวอักษร หรือ ค่าระหว่าง -128 และ 127
byte	1	0 ถึง 255	หมือนกับ char แต่เป็นบวกเท่านั้น

- **ตัวแปร int** ถ้าไม่ได้คำนึงถึงสมรรถนะหรือประสิทธิภาพการเก็บข้อมูล และถ้าค่าสูงสุด และค่าต่ำสุดอยู่ในขอบเขต เราไม่จำเป็นต้องใช้ตัวแปรที่เป็นเลขทศนิยม และควรจะใช้ตัวแปรประเภท int ตัวอย่าง Arduino ส่วนใหญ่ใช้ตัวแปรประเภท int
- **ตัวแปรแบบ signed และ unsigned** บางครั้งจำเป็นต้องมีจำนวนลบ จำนวนจึงถูกแบ่งประเภทเป็นแบบมีเครื่องหมาย +- (signed) และไม่มีเครื่องหมาย (ไม่ติดลบ) (unsigned) ปกติไม่ว่าค่าจะเป็น +- หรือเป็น + อย่างเดียว เราจะใช้ signed ยกเว้นค่าเป็นบวกและค่าสูงสุดเกินขอบเขตของ signed จึงจะใช้ unsigned
- **ตัวแปรประเภท boolean** Boolean จะเป็น เท็จ (false) หรือ จริง (true) อย่างใดอย่างหนึ่ง ตัวแปรประเภทนี้ใช้คล้ายกับเช็คว่าสวิตช์ถูกกดหรือไม่กด จะคล้ายกับการแสดงสถานะของ pin ขาออกของ Arduino ซึ่งจะแสดงค่าแรงดัน High หรือ Low แทน จริง หรือ เท็จ
- **ตัวแปรประเภท float และ double** float และ double ใช้สำหรับข้อมูลที่เป็นจำนวนทศนิยมและใช้เมื่อรับค่าประมาณของค่าจากการวัด ตัวแปรทั้งสองประเภทนี้มักจะใช้เพื่อคำนวณค่าที่ได้จากเซนเซอร์

- **ตัวแปรประเภท char และ string** char เก็บข้อมูลอักขระในช่องเก็บข้อมูล อาจเป็นตัวอักษร หรือ ตัวเลข string กำหนดตัวแปร string (อักษรหรือตัวเลขที่เรียงต่อกัน) จะกำหนดจากตัวแปรประเภท char มาเรียงต่อกัน

4.3. ค่าคงที่

คือ ค่าที่กำหนดไว้ล่วงให้กับตัวแปรแต่ละประเภทซึ่งจะมีค่าแตกต่างกันตามที่ระบุในตารางในหัวข้อ 3.2

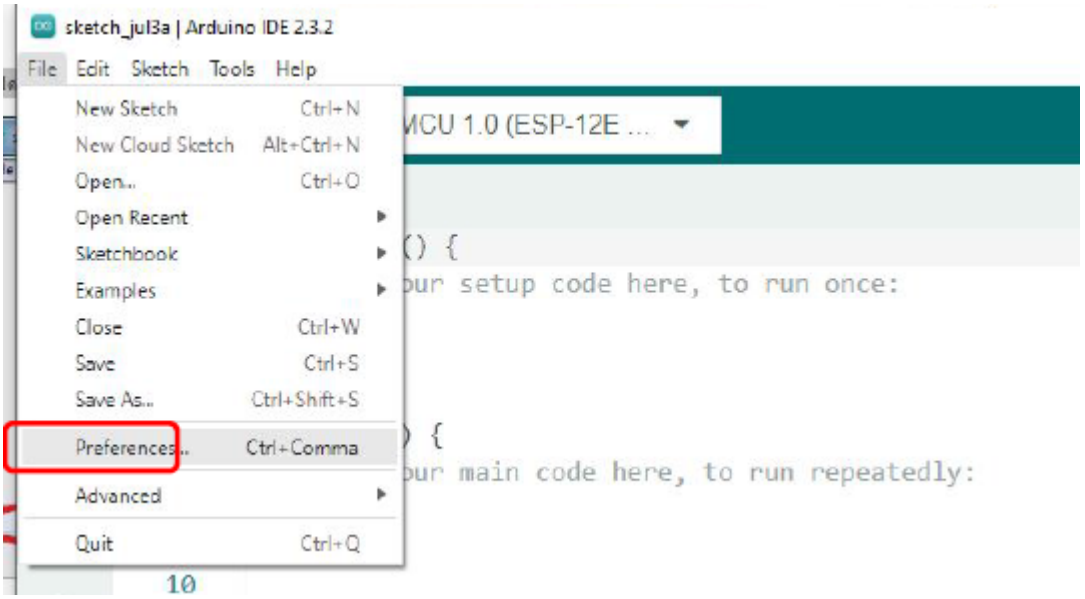
4.4. ฟังก์ชันคำสั่งของ Arduino

ฟังก์ชั่นช่วยให้จัดโครงสร้างโปรแกรมที่เป็นชุดคำสั่ง (code) เพื่อทำงานแต่ละงานได้ กรณีทั่วไป เราจะสร้างฟังก์ชันเมื่อจำเป็นต้องดำเนินการเดียวกันหลายครั้งในโปรแกรม ฟังก์ชั่นจะประกอบด้วย ประเภท ชื่อฟังก์ชัน ตัวแปรขาเข้าของฟังก์ชัน และคำสั่งภายในฟังก์ชัน (statement) ภายในคำสั่ง loop () (นั่นคือภายใน { } ที่ต่อจากคำสั่ง loop ()) ข้อความที่ไม่ใช่ตัวแปรจะเรียกว่าฟังก์ชัน ช่องว่างที่อยู่ใน () ที่ติดกับชื่อฟังก์ชันเกิดจากการที่ฟังก์ชันเป็นประเภท void function (ฟังก์ชันที่ไม่ต้องมี input) เราไม่สามารถใช้ชื่อของฟังก์ชันต่อไปนี้ : setup, loop, if, for, switch เพราะคำเหล่านี้เป็นฟังก์ชันพื้นฐานที่ถูกกำหนดไว้แล้ว

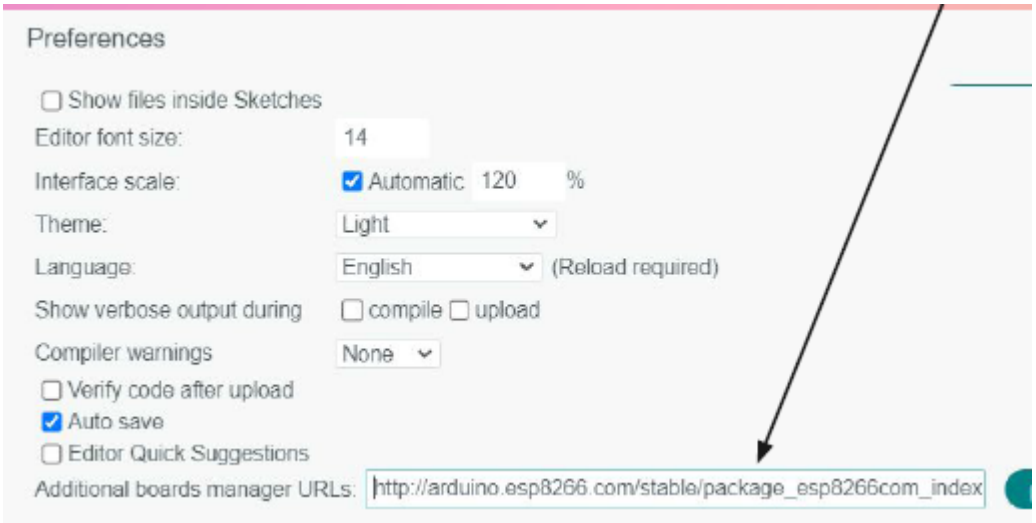
5. การติดตั้ง NodeMCU

การติดตั้งเพื่อให้ NodeMCU สามารถสื่อสารกับคอมพิวเตอร์ผ่านสาย micro USB ได้ มีวิธีดังนี้

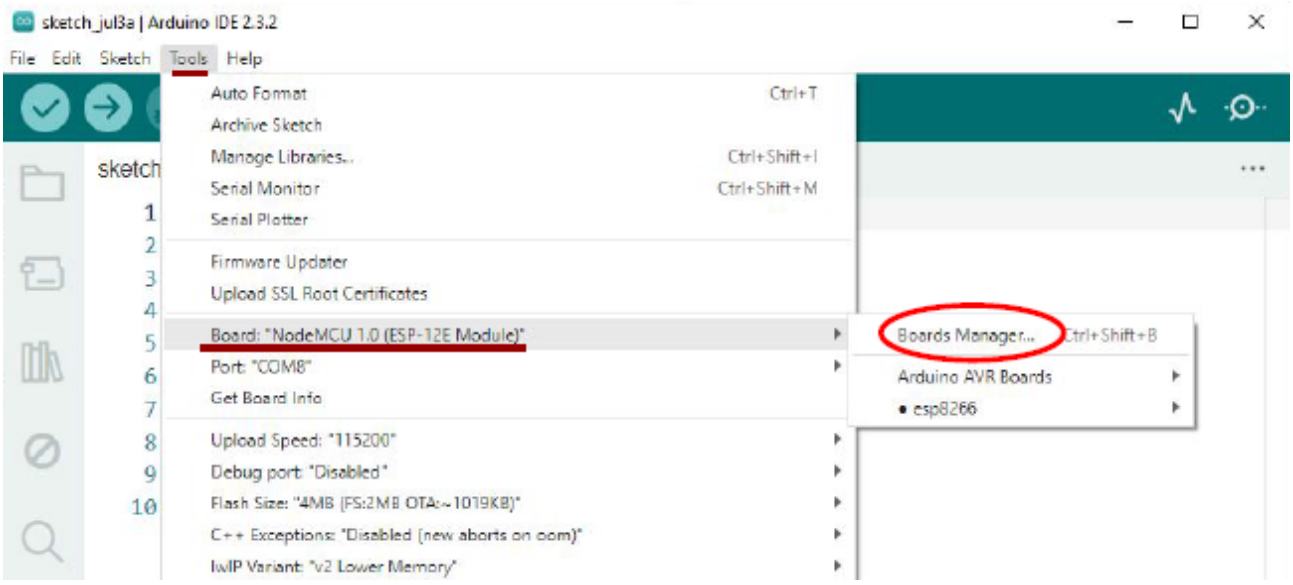
5.1. เปิดโปรแกรม Arduino IDE คลิกไปที่เมนู File -> Preferences



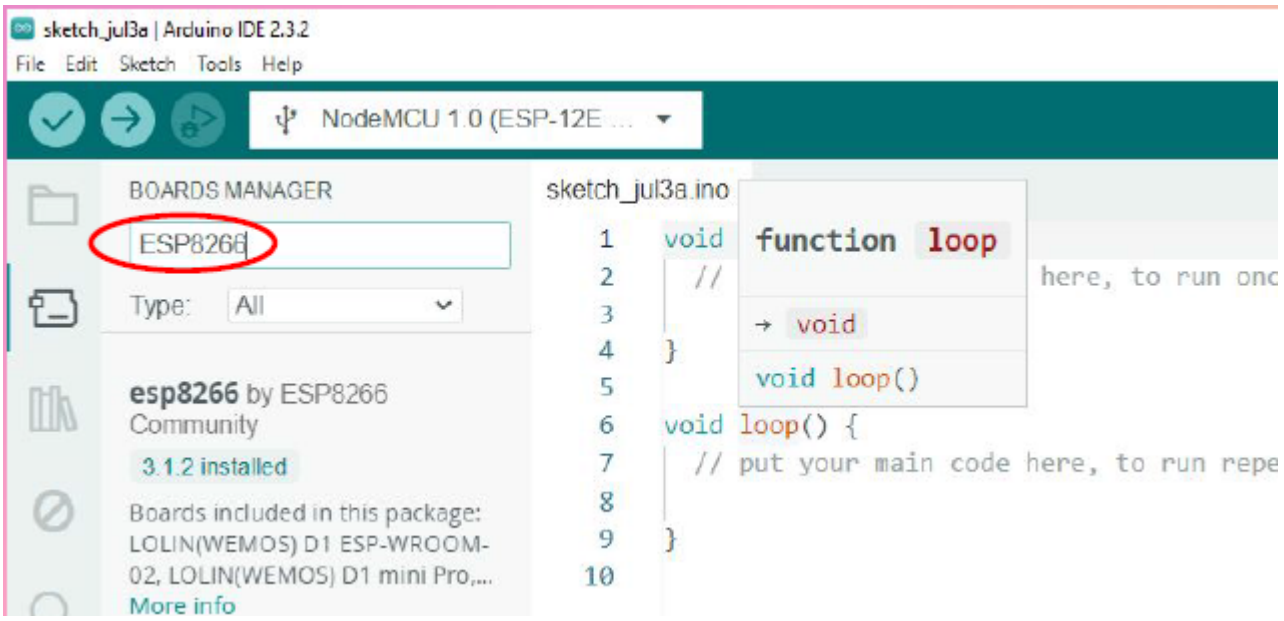
5.2. เพิ่ม http://arduino.esp8266.com/stable/package_esp8266com_index.json ลงในช่อง Additional Boards Manager URLs ดังภาพ แล้วกด OK



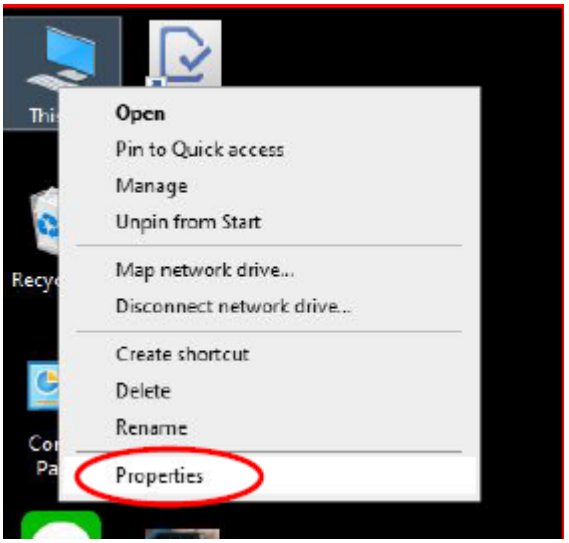
5.3. คลิกไปที่เมนู Tools -> Board -> Board Manager

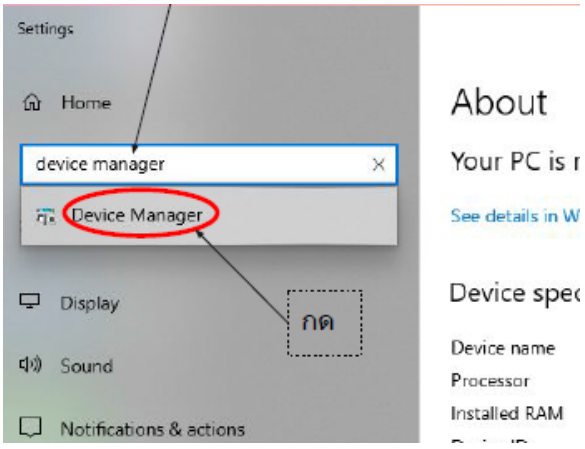


5.4. พิมพ์คำว่า ESP8266 ลงในช่อง และกด Install

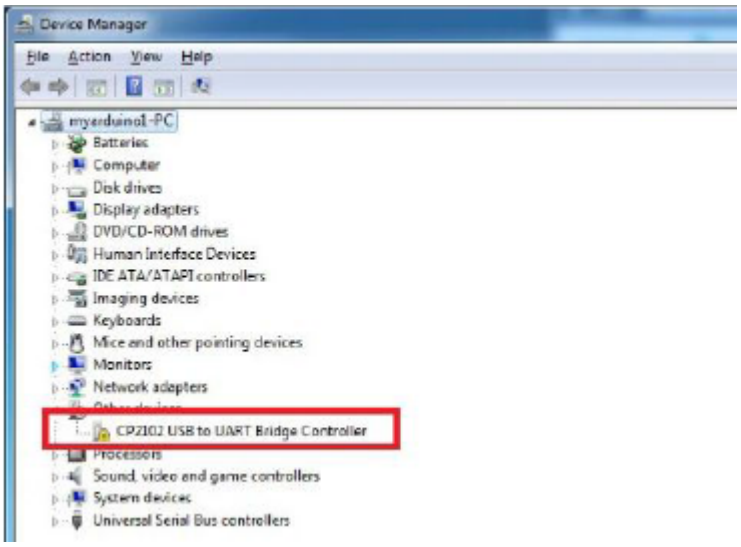


5.5. เสียบบอร์ด NodeMCU/ESP8266 เข้ากับคอมพิวเตอร์ จากนั้นไปที่ Properties -> Device Manager



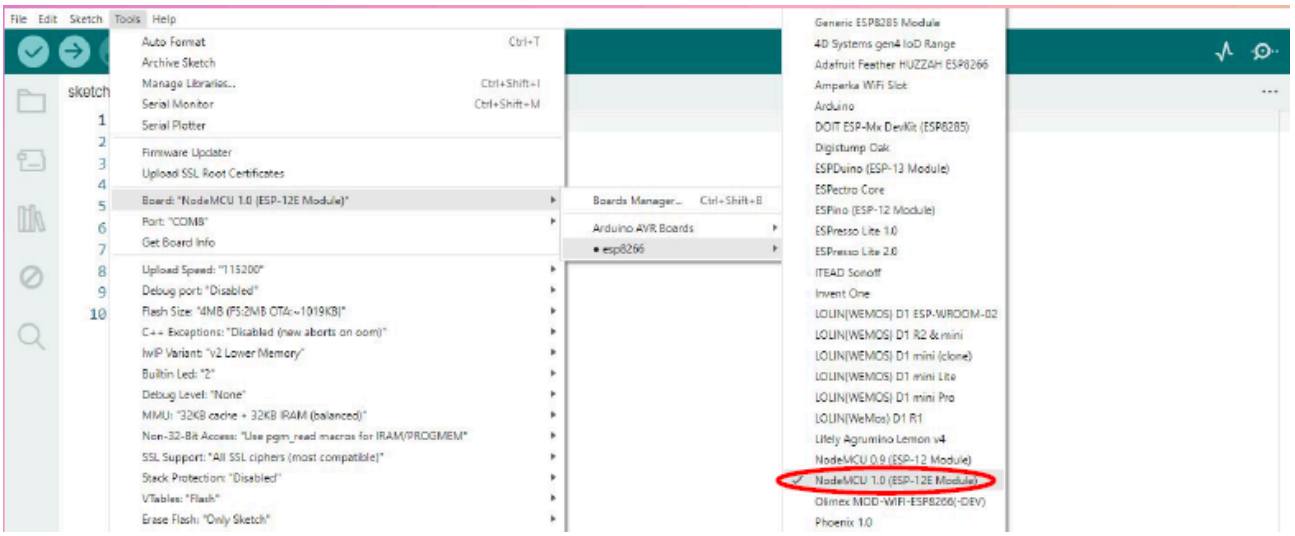


5.6. หากขึ้นเครื่องหมายตกใจ ให้เราติดตั้ง Driver ก่อน โหลดได้จากลิ้งค์ http://www.mediafire.com/file/l47semi8ek3o978/CP210x_Windows_Drivers.zip/file จากนั้นแตกไฟล์และ Install

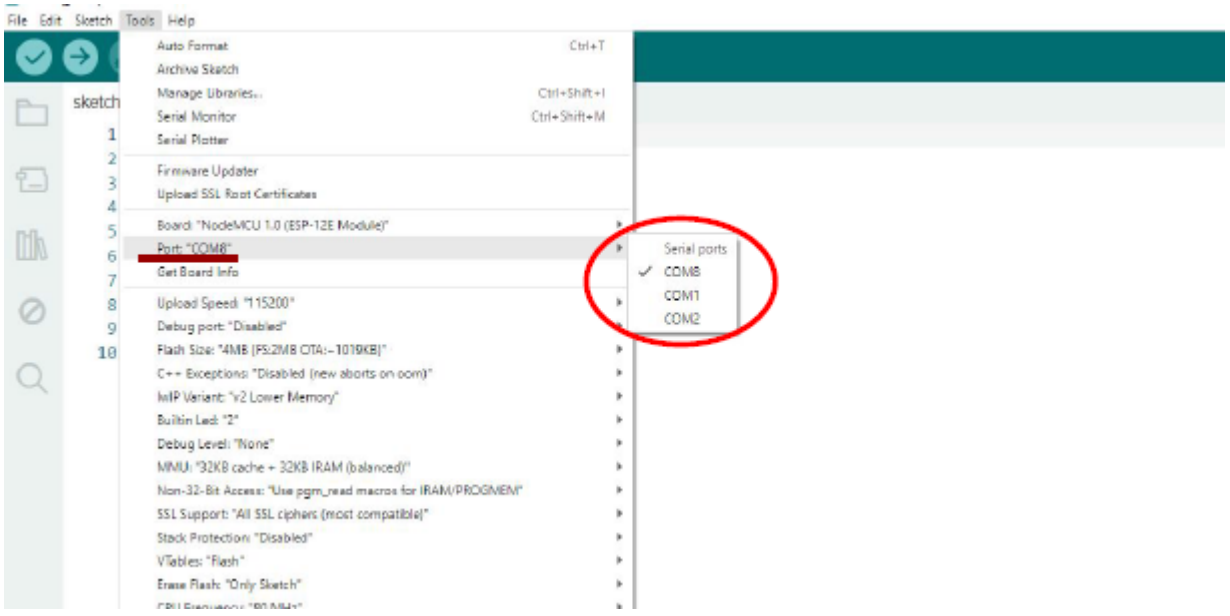


5.7. กลับไปที่ Device Manager ตรวจสอบที่ Ports จะเห็นว่าตอนนี้ คอมพิวเตอร์รู้จัก NodeMCU/ESP8266 แล้ว ให้จำหมายเลข Port ไว้

5.8. กลับไปที่ Arduino ide ไปที่เมนู Tools เพื่อตั้งรุ่นบอร์ด NodeMCU 1.0



5.9. เลือกมายเลขพอร์ตให้ตรงกับใน device manager

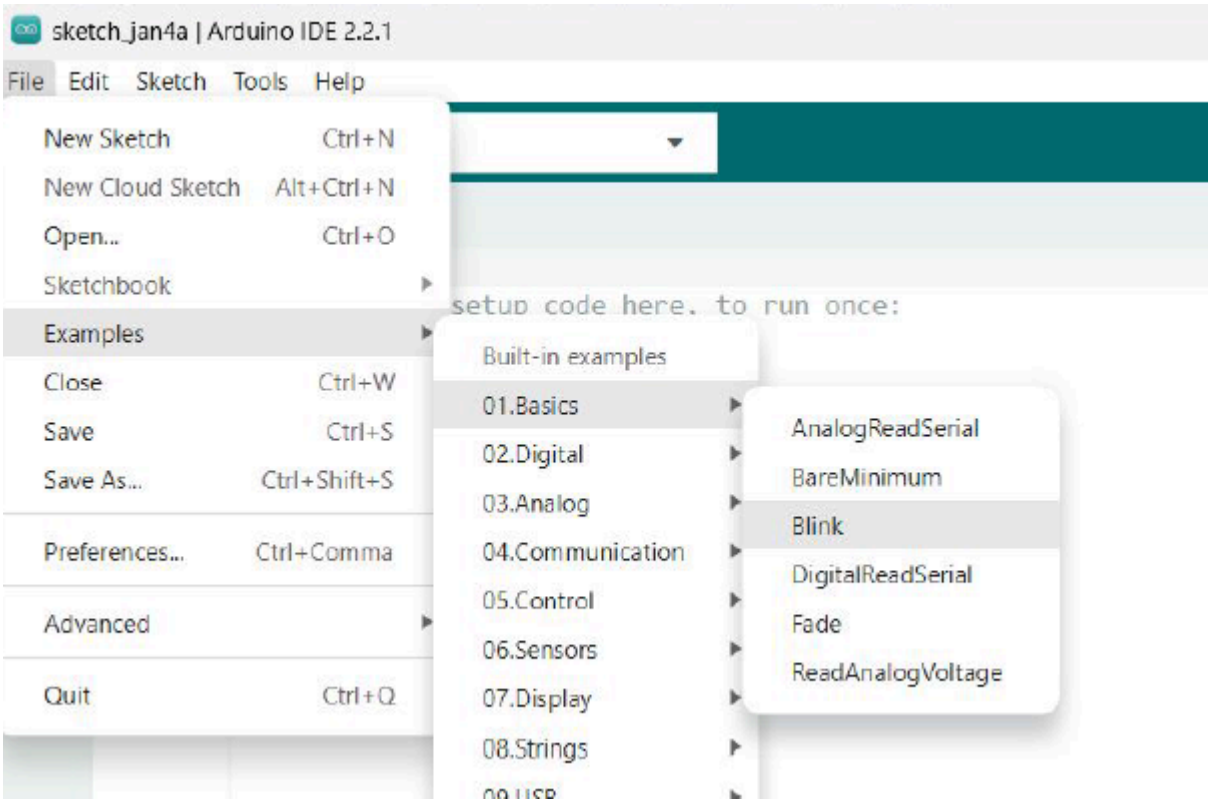


6. การทดสอบ NodeMCU

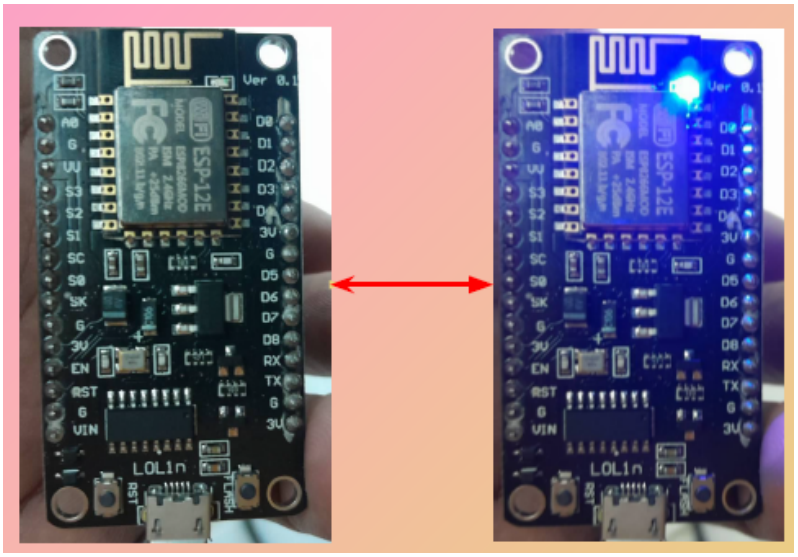
เรามักทดสอบการเชื่อมต่อระหว่างคอมพิวเตอร์และ NodeMCU โดยใช้ Blink Sketch ซึ่งเป็น Sketch ที่สำคัญในการตรวจสอบว่าบอร์ดที่ใช้งานอยู่ทำงานปกติ หรือเข้ากับ IDE ที่กำบะงใช้อยู่หรือไม่

การใช้งาน Blink Sketch มีขั้นตอนดังนี้

1) เปิด blink โดยไปที่ File>> Example::



2) จะปรากฏหน้าต่างตามรูป ให้กด uooad



3) เมื่อ upload เสร็จแล้ว LED (D0)

7. โครงสร้างการควบคุมของ IDE

โครงสร้าง if คือเครื่องหมายเปรียบเทียบจำนวน เครื่องหมายที่สามารถใช้ใน if() ได้มีดังนี้

code	สัญลักษณ์คณิตศาตร์	ความหมาย
x==y	x=y	x มีค่าเท่ากับ y
x!=y	x≠y	x มีค่าไม่เท่ากับ y
x<y	x<y	x มีค่าน้อยกว่า y
x>y	x>y	x มีค่ามากกว่า y
x<=y	x≤y	x มีค่าน้อยกว่าหรือเท่ากับ y
x>=y	x≥y	x มีค่ามากกว่าหรือเท่ากับ y

โครงสร้าง if และ else เป็นโครงสร้างที่ให้ดำเนินการอย่างหนึ่งเมื่อเงื่อนไขสอดคล้องและดำเนินการอีกอย่างเมื่อเงื่อนไขไม่สอดคล้อง

```
int x = 3;
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  if (x>5)
  {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(500);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(500);                      // wait for a second
  }
  else
  {
   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on
  }
}
```

โครงสร้าง if และ else if เป็นโครงสร้างแบบ 2 เงื่อนไขขึ้นไป โดยพิจารณาตามลำดับ เช่น ถ้า if() สอดคล้อง จะไม่สนใจเงื่อนไข else if()

```
int x = 6;
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
  if (x>5)
  {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(500);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(500);                      // wait for a second
 }
  else if (x>=3)
  {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on
  }
  else
  {
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off
  }
}

```

โครงสร้าง for() จะเป็นคำสั่งให้ทำซ้ำคำสั่งใน { } หลัง for() ตราบเท่าที่เงื่อนไขใน () เป็นจริง มีรูปแบบการใช้คำสั่งดังนี้

for(ค่าเริ่มต้น, เงื่อนไข, การเพิ่มค่าต่อรอบ) เช่น

```
for(int i=0;i<3;i++)
{...}
```

- int i=0 หมายถึงกำหนดตัวแปร i เริ่มต้นเท่ากับ 0
- i<3 เงื่อนไขที่จะให้ดำเนินการคำสั่งใน {...}
- i++ คือเมื่อดำเนินการคำสั่งใน {...} 1 ครั้งแล้วให้ i เพิ่ม 1 ค่า

โครงสร้าง for() อยู่ใน loop() จะเป็นลักษณะของการทำซ้ำ ซ้อนกับคำสั่งทำซ้ำอีกที (ลอง upload code ด้านล่างแล้วสังเกตผล)

```
1   void setup() {
2       pinMode(LED_BUILTIN, OUTPUT);
3   }
4   void loop() {
5     for (int i=0;i<3;i++)
6     {
7     digitalWrite(LED_BUILTIN, HIGH);
8     delay(200);
9     digitalWrite(LED_BUILTIN, LOW);
10    delay(200);
11   }
12    delay(2000); // wait for 2 sec
13  }
```

โครงสร้าง while() จะมีลักษณะการใช้งานคล้าย for() แต่ต่างกันที่กำหนดค่าเริ่มต้นก่อน while() และกำหนดการเพิ่มค่าที่บรรทัดสุดท้ายใน { }

```
1   void setup() {
2       pinMode(LED_BUILTIN, OUTPUT);
3   }
4   void loop() {
5     int i=0;
6     while (i<3)
7     {
8     digitalWrite(LED_BUILTIN, HIGH);
9     delay(200);
10    digitalWrite(LED_BUILTIN, LOW);
11    delay(200);    // wait for 0.2 sec
12    i++;
13   }
14    delay(2000); // wait for 2 sec
15  }
```

โครงสร้าง break ใช้สำหรับออกจาก for หรือ while ก่อนเงื่อนไขที่กำหนดไว้ มักจะใช้คู่กับ if() เพื่อกำหนดเงื่อนไขว่าตอนไหนจะ break

```
1   void setup() {
2       pinMode(LED_BUILTIN, OUTPUT);
3   }
4   void loop() {
5     for(int i=0;i<10;i++)
6     {
7     digitalWrite(LED_BUILTIN, HIGH);
8     delay(200);
9     digitalWrite(LED_BUILTIN, LOW);
10    delay(200);
11    }
12    delay(2000);
13  }
14
```

```
1   void setup() {
2       pinMode(LED_BUILTIN, OUTPUT);
3   }
4   void loop() {
5     for(int i=0;i<10;i++)
6     {
7       if(i==5) break;
8     digitalWrite(LED_BUILTIN, HIGH);
9     delay(200);
10    digitalWrite(LED_BUILTIN, LOW);
11    delay(200);
12    }
13    delay(2000);
14  }
15
```

โครงสร้าง switch() จะมีลักษณะคล้าย if และ else if แต่เงื่อนไขแต่ละกรณีจะเป็นตัวเลข 1,2,3… เท่านั้น

```
switch(i)
{
  case 1: (เงื่อนไขนี้ทำงานเมื่อค่า i=1)
    คำสั่งต่าง ๆ สำหรับ case 1
  break;
 case 2: (เงื่อนไขนี้ทำงานเมื่อค่า i=2)
    คำสั่งต่าง ๆ สำหรับ case 2
```

break;
case 3: (เงื่อนไขนี้ทำงานเมื่อค่า i=3)
คำสั่งต่าง ๆ สำหรับ case 3
break;
}

```
1   void setup() {
2       pinMode(LED_BUILTIN, OUTPUT);
3   }
4   void loop() {
5     int i=3;
6     switch(i)
7     {
8     case 1:
9     digitalWrite(LED_BUILTIN, HIGH);
10    delay(200);
11    digitalWrite(LED_BUILTIN, LOW);
12    delay(200);   // wait for 0.2 se
13    break;
14    case 2:
15    digitalWrite(LED_BUILTIN, HIGH);
16    delay(1000);
17    digitalWrite(LED_BUILTIN, LOW);
18    delay(1000);   // wait for 1 sec
19    break;
20    case 3:
21    digitalWrite(LED_BUILTIN, HIGH);
22    break;
23    }
24  }
```