เอกสารเพิ่มเติม “การใช้งานไมโครคอนโทรลเลอร์และเซนเซอร์ในการสร้างนวัตกรรม”

Supplementary Information of “Microcontroller and Sensors for Innovative Applications”

โดย สุวิทย์ กิระวิทยา และคณะ

บทเสริมเรื่อง
การใช้งานสวิตช์

SI1 สวิตช์

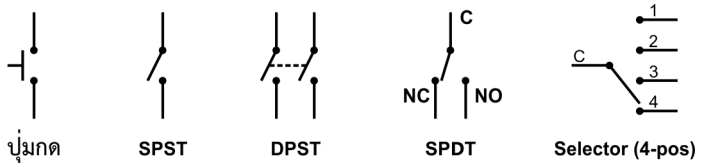
สวิตช์ (switch) เป็นอุปกรณ์พื้นฐานในทางไฟฟ้าและอิเล็กทรอนิกส์ที่มีการใช้งานอย่างแพร่หลาย สวิตช์ไฟฟ้าโดยทั่วไปจะทำหน้าที่เป็นอุปกรณ์ตัดต่อกระแสไฟฟ้า ในขณะที่สวิตช์ในทางอิเล็กทรอนิกส์จะทำหน้าที่รับข้อมูลจากผู้ใช้มาประมวลผล นอกจากนี้ สวิตช์ยังสามารถนำมาใช้เป็นเซนเซอร์ที่บอกตำแหน่งพิกัดของอุปกรณ์ที่มีการเคลื่อนที่ต่าง ๆ ได้ โดยอาศัยการเปลี่ยนสถานะทางวงจรของสวิตช์ด้วยอุปกรณ์ทางกล เช่น ลิมิตสวิตช์

สวิตช์มีหลากหลายชนิด รูปร่าง ขนาด/พิกัด โดยสวิตช์ที่มีการใช้งานกันมากและพบเห็นได้ทั่วไปในทางไฟฟ้า/อิเล็กทรอนิกส์มีสัญลักษณ์ดังแสดงในรูปที่ 1 โดยสวิตช์ปุ่มกด (Pushbutton Switch) และสวิตช์ปิด-เปิดแบบ SPST (Single Pole Single Throw) เป็นสวิตช์พื้นฐานที่ใช้งานได้ง่ายที่สุดเพราะมีเพียงหน้าสัมผัสเดียว สำหรับสวิตช์ที่ใช้ในงานที่ซับซ้อนขึ้นมีด้วยกันมากมายหลายชนิด ตัวอย่างเช่น สวิตช์ปิด-เปิดแบบ DPST (Double Pole Single Throw) และแบบ SPDT (Single Pole Double Throw) และสวิตช์เลือก (Selector Switch) แบบ 4 ตำแหน่ง แสดงรูปที่ 1 โดยทั่วไป ที่ตัวสวิตช์จะมีการระบุขา โดยมีตัวย่อที่สำคัญคือ

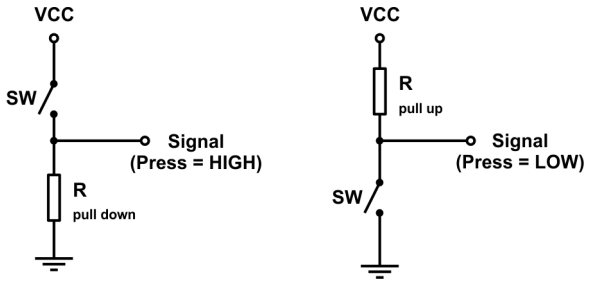
C = Common คือขาร่วม

NO = Normally Open คือขาที่ปกติจะเป็นวงจรเปิด หากไม่มีการกดสวิตช์

NC = Normally Closed คือขาที่ปกติจะเป็นวงจรปิด



รูปที่ 1 สัญลักษณ์ของสวิตช์ต่าง ๆ ที่พบเห็นได้ทั่วไป

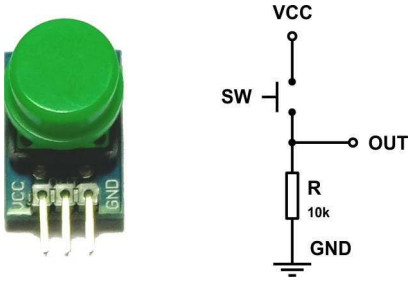


รูปที่ 2 วงจรสวิตช์พื้นฐานสองประเภทที่มีการต่อตัวต้านทานเพื่อดึงแรงดันลง (pull down) และ ดึงแรงดันขึ้นขึ้น (pull up)

สำหรับการใช้งานสวิตช์นั้น เราจะต้องต่อวงจรให้แก่สวิตช์เพื่อให้สามารถส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ได้ รูปที่ 2 แสดงลักษณะการต่อวงจรสวิตช์พื้นฐานสองประเภท ในประเภทแรกจะต่อให้ตัวต้านทานดึงระดับแรงดันลง (pull down) เป็นระดับกราวด์ (0 V) ในขณะที่ไม่มีการกดสวิตช์ เราจึงเรียกตัวต้านทานตัวนี้ว่า $R_{pull\ down}$ โดยเมื่อมีการกดสวิตช์แล้ว สัญญาณจะมีระดับแรงดันเป็น VCC หรือเป็นระดับ HIGH ในการต่อสวิตช์อีกประเภทหนึ่ง หากไม่มีการกดสวิตช์ สัญญาณจะถูกดึงไปอยู่ที่ระดับ VCC ผ่าน $R_{pull\ up}$ โดยระดับสัญญาณจะกลายเป็นกราวด์หรือ LOW เมื่อมีการกดสวิตช์

หากพิจารณาให้ถี่ถ้วนจะพบว่า เราจะต้องมีการนำตัวต้านทานมาต่อเป็นวงจรสวิตช์เสมอ โดย ค่า R ของตัวต้านทานที่นำมาต่อนี้จะบ่งบอกระดับกระแสไฟฟ้าที่ไหลและระดับการสูญเสียกำลังไฟฟ้าในวงจร หากว่าใช้ R ที่มีค่าน้อยเกินไปก็จะทำให้เกิดกระแสไหลมาก ($I = VCC/R$) และเกิดการสูญเสียมาก ($P = VCC^2/R = I \times VCC$) แต่หากใช้ R ที่มีค่ามากเกินไปก็ทำให้สัญญาณกระแสมีระดับต่ำมากและอาจทำให้สัญญาณรบกวนส่งผลกระทบต่อการทำงานของวงจรได้ โดยทั่วไป เรามักจะใช้ R ที่มีค่าระหว่าง 10 kΩ – 100 kΩ สำหรับวงจรสวิตช์นี้

รูปที่ 3 แสดงภาพถ่ายและวงจรโมดูลสวิตช์แบบปุ่มกด ที่นำมาใช้ในการทดลองในบทนี้ โดยเนื่องจากวงจรสวิตช์ทั้งสองประเภทที่แสดงในรูปที่ 2 มีการใช้งานกันทั่วไป ผู้ใช้จึงควรที่จะตรวจสอบการต่อวงจรภายในโมดูลสวิตช์นี้ได้ด้วยตนเอง โดยใช้มัลติมิเตอร์วัดค่าความต้านทานระหว่างขาต่าง ๆ ของโมดูล



รูปที่ 3 ภาพถ่ายและแผนผังวงจรโมดูลสวิตช์แบบปุ่มกด

SI2 การเขียนโปรแกรมรับค่าดิจิทัล

คำสั่งเบื้องต้นในการใช้งานสวิตช์ ใน ภาษา Arduino คือ ฟังก์ชันอินพุตแบบดิจิทัล (Digital Input) ซึ่งมีฟังก์ชันที่ต้องทราบ 2 ฟังก์ชัน คือ

1) **pinMode(pin, mode)** เป็นคำสั่งใช้กำหนดขาหรือพอร์ตใด ๆ ที่ทำหน้าที่เป็นพอร์ตดิจิทัล ปกติจะเรียกใช้ในฟังก์ชัน setup() เพื่อกำหนดการทำงานเริ่มต้นของขาต่าง ๆ ที่เกี่ยวข้อง โดย

- pin หมายถึงชื่อหรือเลขขาของบอร์ด NodeMCU เป็น D0, D1, ... หรือเป็นเลขจำนวนเต็ม โดยชื่อขาเหล่านี้ ถูกกำหนดให้สอดคล้องกับพิน GPIO อยู่ก่อนแล้ว
- mode หมายถึงโหมดการทำงาน กำหนดให้มีค่าเป็น **INPUT** หรือ **OUTPUT** เท่านั้น

ตัวอย่างเช่น คำสั่ง pinMode(D5, INPUT) คือการกำหนดให้ขา D5 ของ NodeMCU ทำงานโดยส่งสัญญาณออกไปเท่านั้น

2) **digitalRead(pin)** เป็นคำสั่งให้อ่านสถานะทางลอจิกของขาที่กำหนด โดยค่าที่ส่งคืนมาคือ ลอจิกสูง (HIGH หรือ ค่าตัวเลข “1”) หรือ ลอจิกต่ำ (LOW หรือ ค่าตัวเลข “0”) เท่านั้น

ตัวอย่างเช่น คำสั่ง digitalRead(D5) คือการอ่านค่าลอจิกที่ขา D5 ของ NodeMCU

ข้อมูลเพิ่มเติม: สำหรับ NodeMCU ช่วงของค่าแรงดันของลอจิก HIGH คือ 1.8 – 3.3 โวลต์ และ ของลอจิก LOW คือ -0.3 – ~0.83 โวลต์ โดยหากทำการอ่านค่าลอจิกจากขาที่ไม่มีการเชื่อมต่อในทางวงจร ก็จะได้ค่าลอจิก HIGH คืนมา

SI3 การทดลองควบคุมรีเลย์ด้วยสวิตช์

วัตถุประสงค์

1. สามารถต่อบอร์ด NodeMCU v.3 กับบอร์ดรีเลย์และสวิตช์ได้
2. สามารถเขียนโปรแกรมให้ NodeMCU รับค่าจากสวิตช์เพื่อควบคุมการทำงานของรีเลย์ได้
3. เขียนโปรแกรมควบคุมรีเลย์ด้วยสวิตช์โดยมีการจดจำสถานะของรีเลย์ด้วย

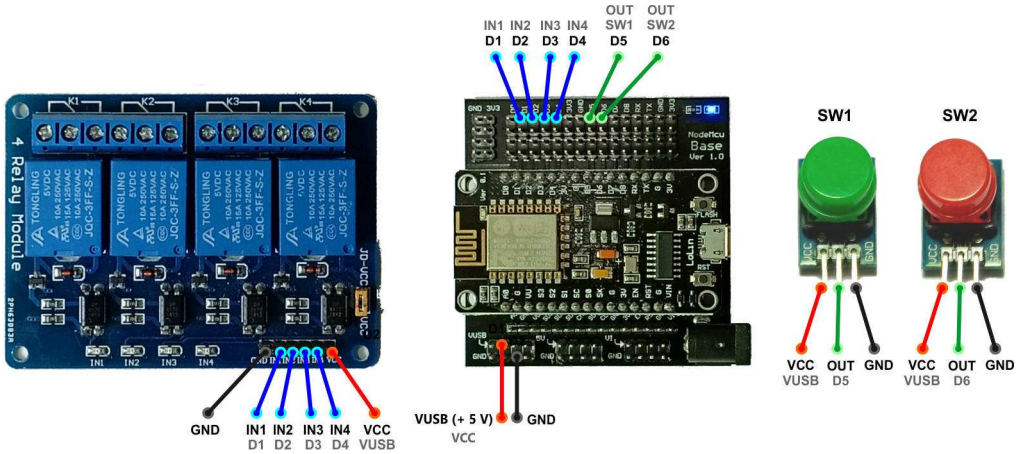
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
5.	สวิตช์แบบปุ่มกด	2 บอร์ด
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	12 เส้น

วิธีการทดลอง

ตอนที่ 1 การควบคุมรีเลย์ด้วยสวิตช์แบบปุ่มกดสองปุ่ม

1. ต่อวงจรตามรูปที่ 4 โดยต่อบอร์ดรีเลย์กับขา D1 – D4 และต่อสวิตช์โมดูลทั้งสองที่ขา D5 และ D6 ของ NodeMCU และต่อไฟ VUSB และ กราวด์ (GND) ด้วย
2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 เพื่อควบคุมการเปิดปิดรีเลย์ทั้งสี่ตัวด้วยสวิตช์ โดยสวิตช์ SW1 จะทำหน้าที่ ON รีเลย์และสวิตช์ SW2 จะทำหน้าที่ OFF รีเลย์ ทั้งสี่ตัว โดยที่จะมีการหน่วงเวลา 0.2 วินาที (ด้วยคำสั่ง delay(200);) เพื่อให้ผู้ทดลองสามารถสังเกตเห็นการทำงานได้อย่างชัดเจน

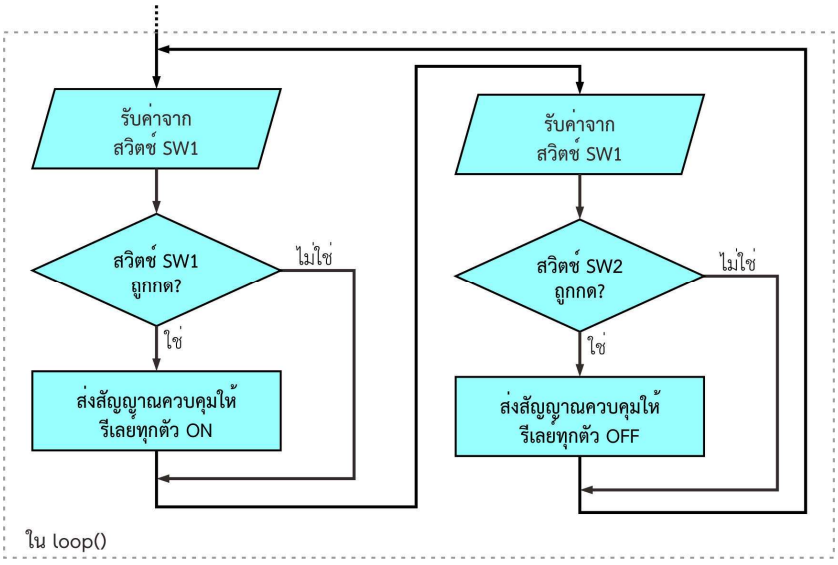


รูปที่ 4 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดรีเลย์และโมดูลสวิตช์

```
// Read Switch & Control Relay by NodeMCU ESP8266

#define ON LOW    // Relay is active low
#define OFF HIGH
int Relay1 = D1;
int Relay2 = D2;
int Relay3 = D3;
int Relay4 = D4;

int SW1 = D5;
int SW2 = D6;
int state;        // Switch State

void Relay_ON() {
    digitalWrite(Relay1, ON); delay(200);
    digitalWrite(Relay2, ON); delay(200);
    digitalWrite(Relay3, ON); delay(200);
    digitalWrite(Relay4, ON); delay(200);
}

void Relay_OFF() {
    digitalWrite(Relay1, OFF); delay(200);
    digitalWrite(Relay2, OFF); delay(200);
    digitalWrite(Relay3, OFF); delay(200);
    digitalWrite(Relay4, OFF); delay(200);
}
```

```
27
28 void setup() {
29     pinMode(Relay1, OUTPUT);
30     pinMode(Relay2, OUTPUT);
31     pinMode(Relay3, OUTPUT);
32     pinMode(Relay4, OUTPUT);
33     pinMode(SW1, INPUT);
34     pinMode(SW2, INPUT);
35     Relay_OFF();
36 }
37
38 void loop() {
39     state = digitalRead(SW1);
40     if(state == HIGH) {
41         Relay_ON();
42     }
43     state = digitalRead(SW2);
44     if(state == HIGH) {
45         Relay_OFF();
46     }
47 }
48
```

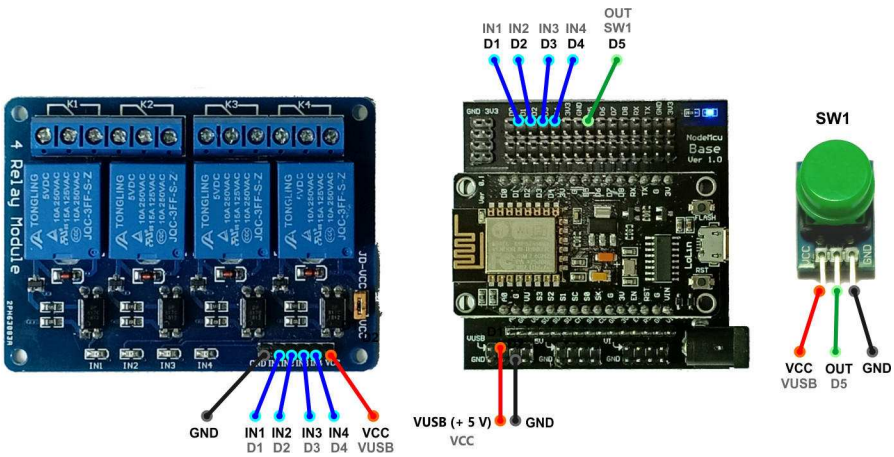
แผนผังการทำงานของโปรแกรมในฟังก์ชัน loop() ของการทดลองนี้แสดงดังรูปที่ 5



รูปที่ 5 แผนผังการทำงานของโปรแกรมที่แสดงในการทดลองตอนที่ 1

ตอนที่ 2 การควบคุมรีเลย์ด้วยสวิตช์แบบปุ่มกดปุ่มเดียว

1. ต่อวงจรตามรูปที่ 6



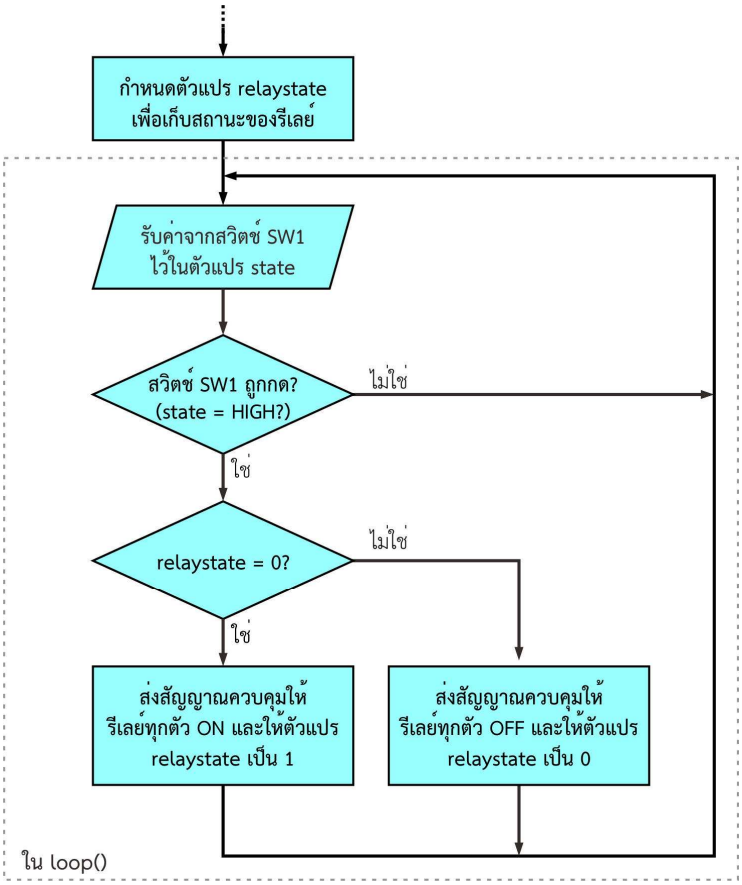
รูปที่ 6 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดรีเลย์และโมดูลสวิตช์

2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 เพื่อควบคุมการเปิดปิดรีเลย์ทั้งสี่ตัวด้วยสวิตช์ โดยสวิตช์ SW1 จะทำหน้าที่ได้ทั้ง ON และ OFF รีเลย์ทั้งสี่ตัวเมื่อมีการกดปุ่ม

```
// Read Switch & Control Relay by NodeMCU ESP8266

#define ON LOW
#define OFF HIGH
int Relay1 = D1;
int Relay2 = D2;
int Relay3 = D3;
int Relay4 = D4;
int SW1 = D5;
int state;    // switch state
int relaystate;  // relay state

void setup() {
    pinMode(Relay1, OUTPUT);
    pinMode(Relay2, OUTPUT);
    pinMode(Relay3, OUTPUT);
    pinMode(Relay4, OUTPUT);
```

```
18      pinMode(SW1, INPUT);
19      Relay_OFF();
20  }
21
22  void Relay_ON() {
23      digitalWrite(Relay1, ON); delay(200);
24      digitalWrite(Relay2, ON); delay(200);
25      digitalWrite(Relay3, ON); delay(200);
26      digitalWrite(Relay4, ON); delay(200);
27      relaystate = 1;
28  }
29
30  void Relay_OFF() {
31      digitalWrite(Relay1, OFF); delay(200);
32      digitalWrite(Relay2, OFF); delay(200);
33      digitalWrite(Relay3, OFF); delay(200);
34      digitalWrite(Relay4, OFF); delay(200);
35      relaystate = 0;
36  }
37
38  void loop() {
39      state = digitalRead(SW1);
40      if(state == HIGH) {
41        if(relaystate == 0) {
42          Relay_ON();
43        }
44        else {
45          Relay_OFF();
46        }
47      }
48  }
49
```

แผนผังการทำงานของโปรแกรมในฟังก์ชัน loop() ของการทดลองนี้แสดงดังรูปที่ 7



รูปที่ 7 แผนผังการทำงานของโปรแกรมที่แสดงในการทดลองตอนที่ 2

แบบฝึกหัดท้ายการทดลอง

ในการทดลองตอนที่ 2 หากเรากดสวิตช์ปุ่มกดค้างไว้ จะพบว่า รีเลย์จะมีการเปลี่ยนสถานะตลอดเวลา ทั้งนี้เนื่องจากเรามิได้กำหนดให้ปุ่มกดต้องเปลี่ยนสถานะจาก HIGH กลับมาเป็น LOW เสียก่อนแล้วจึงอ่านค่าใหม่ ดังนั้นเพื่อแก้ปัญหาการกดปุ่มค้างนี้ (คือให้รีเลย์ถูกเปลี่ยนสถานะเพียงครั้งเดียวเมื่อมีการกดปุ่มค้าง) เราจะต้องแก้ไขโค้ดให้จดจำสถานะของปุ่มกดก่อนหน้านี้ด้วย

จงเขียนโปรแกรมที่ทำให้รีเลย์ไม่มีการเปลี่ยนสถานะเมื่อมีการกดปุ่มค้าง

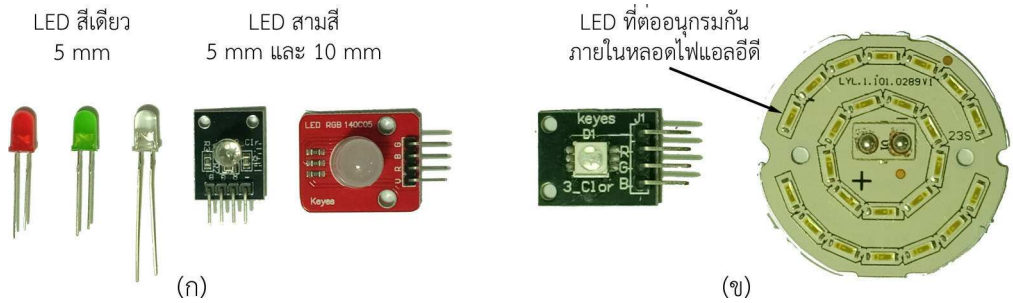
```
int oldstate;    // old switch state

void loop() {
    state = digitalRead(SW1);
    if( (state == HIGH) && (state != oldstate) ) {
      if(relaystate == 0) {
        Relay_ON();
      }
      else {
        Relay_OFF();
      }
    }
    oldstate = state;
}
```

บทเสริมเรื่อง
การใช้งานแอลอีดี

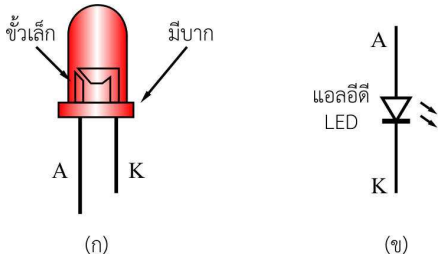
SI1 แอลอีดี

แอลอีดี (LED) หรือไดโอดเปล่งแสง (light emitting diode) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ทำจากสารกึ่งตัวนำและทำหน้าที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานแสง โดยเมื่อมีการฉีดอิเล็กตรอนหรือกระแสไฟฟ้าเข้าไปในตัวแอลอีดีแล้ว อิเล็กตรอนเหล่านั้นก็จะมีการลดพลังงานลง โดยการปลดปล่อยแสงหรืออนุภาคโฟตอนออกมา โดยปรากฏการณ์นี้ในทางวิทยาศาสตร์เรียกว่า อิเล็กโทรลูมิเนสเซนต์ (electroluminescence)

โดยทั่วไป เราแบ่งชนิดของแอลอีดีจากลักษณะสีของแสงที่เปล่งออกมาและจากลักษณะตัวถัง โดยสีของแอลอีดีจะกำหนดจากสารกึ่งตัวนำที่นำมาใช้สร้าง เช่น แอลอีดีสีแดงมักสร้างจาก อลูมิเนียมแกลเลียมอินเดียมฟอสไฟด์ (AlGaInP) และแอลอีดีสีเขียวสร้างจากอินเดียมแกลเลียมไนไตรด์ (InGaN) สำหรับลักษณะของแอลอีดีในปัจจุบันก็มีหลายรูปแบบ รูปที่ 1(ก) แสดงลักษณะแอลอีดีที่พบเห็นได้ทั่วไป (สีเดียวและสามสี ขนาด 5 และ 10 mm) และ รูปที่ 1(ข) แสดงแอลอีดีชนิดเอสเอ็มดี (SMD, surface mount device) ที่มีใช้งานอย่างมากในหลอดไฟในปัจจุบัน สำหรับข้อกำหนดเรื่องขาของแอลอีดีนั้น ผู้ใช้ต้องสังเกตจากลักษณะภายในตัวแอลอีดี คือ ขั้วภายในที่เล็กกว่าจะเป็นขั้วแอโนด (A) และ ขั้วที่ใหญ่กว่าจะเป็นขั้วแคโทด (K) หรือสังเกตจากลักษณะที่มีบากบนตัวถังดังแสดงในรูปที่ 2(ก) สัญลักษณ์ทางวงจรของแอลอีดีแสดงดังรูปที่ 2(ข)



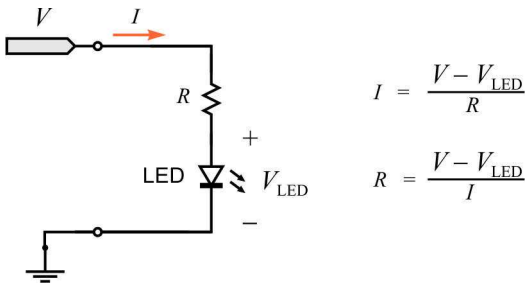
รูปที่ 1 (ก) ภาพแอลอีดีในลักษณะตัวถังต่าง ๆ และ (ข) ภาพแอลอีดีชนิดเอสเอ็มดี



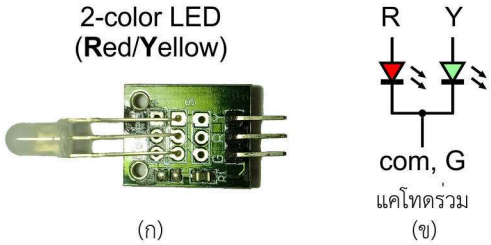
รูปที่ 2 (ก) ภาพลักษณะแอลอีดีและจุดสังเกตขา และ (ข) สัญลักษณ์ทางวงจรของแอลอีดี

โดยทั่วไปในการใช้งานแอลอีดีนั้น เราจะต้องคำนึงถึงขั้วต่อและระดับกระแสที่ฉีด นั่นคือ แอลอีดีจะไม่เปล่งแสงหากเราต่อกลับขั้ว เนื่องจากแอลอีดีเป็นไดโอดประเภทหนึ่งซึ่งจัดอยู่ในอุปกรณ์ประเภทมีขั้ว สำหรับกระแสที่ไหลผ่านแอลอีดีนั้นมักจะถูกจำกัดด้วยตัวต้านทานที่นำมาต่ออนุกรมกับแอลอีดี รูปที่ 3 แสดงลักษณะการต่อแอลอีดีอย่างง่าย โดยเมื่อเราให้แรงดัน V แก่วงจรนี้ ก็จะเกิดกระแส I ไหลและมีแรงดัน V_{LED} ตกคร่อมแอลอีดี (ประมาณ 1.6 – 2.5 V ขึ้นกับสีของแอลอีดี) โดยค่ากระแส I ที่ไหลผ่านแอลอีดีจะเป็นตัวกำหนดความสว่างของหลอดแอลอีดี ซึ่งโดยทั่วไป ค่ากระแสนี้จะมีค่าไม่เกิน 10-20 mA ดังนั้น เราจะต้องเลือกค่าความต้านทานที่เหมาะสมในการใช้งานแอลอีดี

แอลอีดีที่มีการใช้งานกันในปัจจุบัน นอกจากประเภทสีเดียวแล้ว ยังมีแบบประเภทสองสี (2-color LED) และสามสีหรืออาร์จีบีแอลอีดี (RGB LED) ด้วย โดยเราสามารถผสมสีของแสงได้ด้วยแอลอีดีแบบหลายสีนี้ ซึ่งแอลอีดีหลายสีนี้มักจะมีขาร่วมกัน ที่เรียกว่า ขาร่วมหรือขาคอมมอน (Common) โดยมีทั้งแบบแคโทดร่วม (common cathode) และแอโนดร่วม (common anode) ตัวอย่างภาพถ่ายโมดูลแอลอีดีสองสีชนิดแคโทดร่วมและลักษณะวงจรภายในโมดูลแสดงในรูปที่ 4(ก) และ 4(ข) ตามลำดับ



รูปที่ 3 วงจรขับแอลอีดีและสูตรที่เกี่ยวข้องในการคำนวณค่ากระแสและค่าความต้านทาน



รูปที่ 4 (ก) ภาพถ่ายโมดูลแอลอีดีสองสี (สีแดงและเหลือง) ชนิดแคโทดร่วม และ (ข) ลักษณะวงจรภายในโมดูล

สำหรับการใช้งานแอลอีดีร่วมกับบอร์ด NodeMCU ซึ่งใช้ชิป ESP8266 นั้น เนื่องจากชิปนี้สามารถให้กระแสเพื่อสัญญาณได้สูงสุดเพียง 12 mA ดังนั้นในการต่อบอร์ดนี้ร่วมกับแอลอีดีทั่วไป เราจึงไม่จำเป็นต้องต่อตัวต้านทานเพื่อจำกัดค่ากระแสที่ไหลผ่านแอลอีดี และทำให้ใช้งานแอลอีดีร่วมกับบอร์ดนี้ทำได้อย่างง่ายดาย นั่นคือเราสามารถต่อตรงระหว่างแอลอีดีกับขาสัญญาณได้เลย ทั้งนี้ผู้ใช้ควรพึงระลึกไว้ว่าการต่อแอลอีดีตรงกับขาสัญญาณแรงดันนี้อาจทำให้แอลอีดีเสียหายได้หากขาสัญญาณนั้นสามารถจ่ายกระแสได้มากเพียงพอ

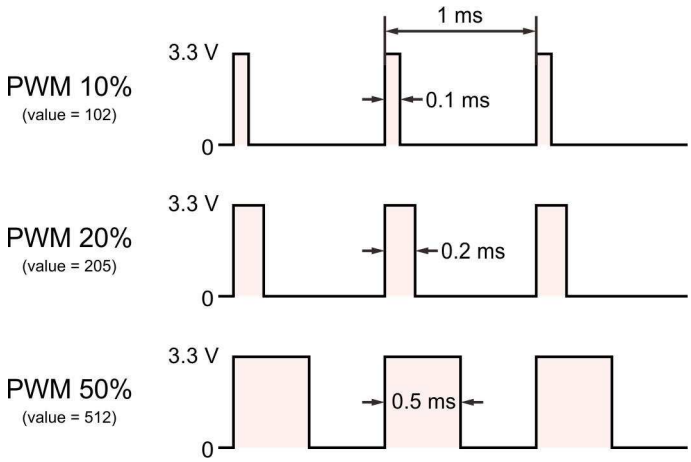
SI2 การเขียนโปรแกรมควบคุมแอลอีดี

การสั่งการแอลอีดีนั้นทำได้ในสองโหมดคือ โหมดดิจิทัลและโหมดแอนะล็อก โดยมีรายละเอียดดังนี้

1) โหมดดิจิทัล คือ การควบคุมให้แอลอีดีติดหรือดับ ซึ่งสั่งการได้ด้วยด้วยคำสั่ง **digitalWrite(pin, value)** โดย value มีค่าเป็นได้เพียงลอจิกต่ำ “LOW” หรือลอจิกสูง “HIGH” เท่านั้น และรายละเอียดของคำสั่ง มีลักษณะเช่นเดียวกันกับคำสั่งในการควบคุมรีเลย์ดังแสดงในหัวข้อที่ 4.2

2) โหมดแอนะล็อก คือ การควบคุมให้แอลอีดีมีความสว่างระดับต่าง ๆ ซึ่ง NodeMCU จะสร้างสัญญาณดิจิทัลชนิด PWM หรือสัญญาณที่มีการมอดูเลตความกว้างพัลส์ (Pulse Width Modulation) ในการควบคุมให้แอลอีดีติดและดับในอัตราเร็วที่เร็วกว่าการตอบสนองของดวงตามนุษย์ คือที่ความถี่ 1 kHz โดยดวงตามนุษย์จะตอบสนองต่อการกระพริบที่ความถี่เพียงประมาณ 10 Hz จึงส่งผลให้เราเห็นความสว่างของแอลอีดีที่ระดับต่างๆ รูปที่ 5 แสดงลักษณะสัญญาณ PWM ที่เปอร์เซ็นต์รอบการทำงาน (duty cycle) 10, 20 และ

50% ตามลำดับ สำหรับสัญญาณ PWM ที่ 0% และ 100% ก็คือสัญญาณดิจิทัลลอจิก LOW (0 V) และลอจิก HIGH (3.3 V) นั่นเอง



รูปที่ 5 ลักษณะสัญญาณ PWM ที่สร้างจาก NodeMCU ด้วยคำสั่ง analogWrite

คำสั่งที่ใช้ในการสร้างสัญญาณ PWM จากบอร์ดไมโครคอนโทรลเลอร์ด้วยภาษา Arduino คือ คำสั่ง **analogWrite(pin, value)** โดยค่า value นี้ จะเป็นค่าเลขฐานสิบระหว่าง 0 ถึง 1023 (= 2^{10}-1) ซึ่งค่าสูงสุด 1023 นี้จะเฉพาะเจาะจงกับบอร์ด NodeMCU เนื่องจาก ESP8266 เป็นชิปประมวลผลแบบ 10 บิต (สำหรับบอร์ด Arduino แบบดั้งเดิม value นี้จะมีค่าอยู่ระหว่าง 0 ถึง 255 (= 2^{8}-1) เท่านั้น)

SI3 การทดลองควบคุมแอลอีดี

วัตถุประสงค์

1. สามารถต่อบอร์ด NodeMCU v.3 กับบอร์ดแอลอีดีได้
2. สามารถเขียนโปรแกรมให้ NodeMCU รับค่าจากคีบอร์ดผ่านพอร์ตอนุกรมเพื่อควบคุมการทำงานของแอลอีดีได้
3. เขียนโปรแกรมควบคุมแอลอีดีให้ปรับความสว่างได้ตามต้องการ

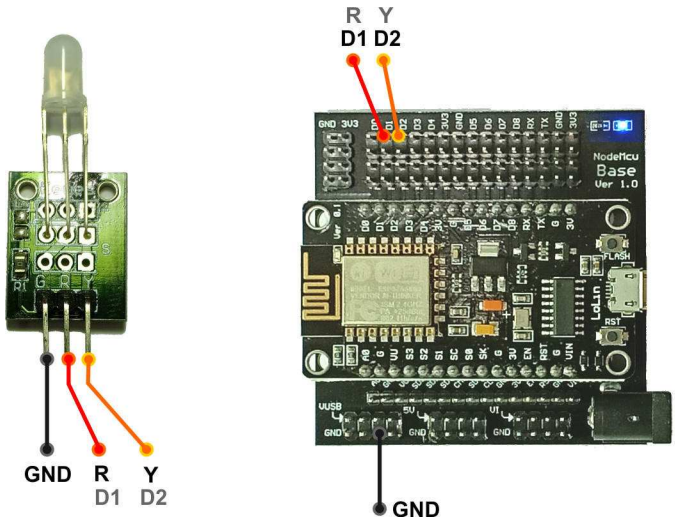
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดแอลอีดีชนิด 2 สีแบบแคโทดร่วม	1 บอร์ด
5.	สาย USB	1 เส้น
6.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	12 เส้น

วิธีการทดลอง

ตอนที่ 1 การควบคุมแอลอีดีด้วยคีบอร์ดผ่านพอร์ตอนุกรม

1. ต่อวงจรตามรูปที่ 6



รูปที่ 6 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดโมดูลแอลอีดีสองสีชนิดแคโทดร่วม

2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 เพื่อควบคุมการทำงานของแอลอีดีทั้งสองสีด้วยการกดปุ่มบนแป้นคีบอร์ด

```
// Control LED via Serial Port

int RLED = D1;
int YLED = D2;

char inchar;   // Keep the reading value in inchar

void setup() {
    // Set serial speed to 9600 bps
    Serial.begin(9600);
    pinMode(RLED, OUTPUT);
    pinMode(YLED, OUTPUT);
}

void loop() {
    // Read Serial and keep it in inchar
    inchar = Serial.read();

    if(inchar == '1')    {
        digitalWrite(RLED, HIGH);
    }

    if(inchar == '2')    {
        digitalWrite(RLED, LOW);
    }

    if(inchar == '3')    {
        digitalWrite(YLED, HIGH);
    }

    if(inchar == '4')    {
        digitalWrite(YLED, LOW);
    }
}

```

3. เปิด Serial Monitor ในเมนู Tools (หรือกด Ctrl+Shift+M บนคีบอร์ด) แล้วทดลองป้อนตัวเลข ‘1’, ‘2’, ‘3’ หรือ ‘4’ แล้วกด Enter (หรือคลิ๊กที่ปุ่ม Send) แล้วสังเกตสีของแอลอีดี โดยเมื่อกดปุ่ม ‘1’ แล้ว ‘3’ ให้แอลอีดีทั้งสองสีสว่างพร้อมกัน แล้วสังเกตสีของแอลอีดี (เนื่องจากแอลอีดีสีแดงมีความสว่างมากกว่าทำให้สีของแสงที่ได้เป็นสีแดง-ส้ม)

4. ทำการแก้ไขโค้ดบรรทัดที่ 20 และเพิ่มบรรทัดที่ 21 คือ

```
20      // digitalWrite(RLED, HIGH);
21      analogWrite(RLED, 100);
22     }
23
```

จากนั้นจึงทำการอัปโหลดโปรแกรมแล้วทดลองตามข้อ 3. ใหม่อีกครั้ง สังเกตสีของแอลอีดีที่เกิดจากการผสมแสงสีแดงและสีเหลือง

ตอนที่ 2 การควบคุมแอลอีดีให้แสดงแสงความเข้มต่าง ๆ อย่างต่อเนื่อง

1. ต่อวงจรตามรูปที่ 6 (วงจรเดิม)

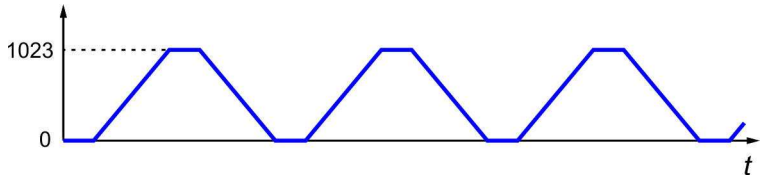
2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดและสังเกตการทำงานของแอลอีดี

```
1  // Flickering of LED
2
3  int RLED = D1;
4
5  void setup() {
6      pinMode(RLED, OUTPUT);
7  }
8
9  void loop() {
10     for (int i=0; i<1024; i++) {
11         analogWrite(RLED, i);
12         delay(1);
13     }
14     delay(100);
15 }
16
```

3. ทำการเปลี่ยนค่าการหน่วงเวลาในคำสั่ง delay จาก 1 เป็น 2 แล้วสังเกตความถี่ในการกระพริบ

แบบฝึกหัดท้ายการทดลอง

จากตัวอย่างโค้ดในการทดลองตอนที่ 2 จงเขียนโค้ดที่ควบคุมให้แอลอีดีค่อย ๆ สว่างขึ้น (จากมืดไปสว่างที่สุด) แล้วค่อย ๆ มืดลง (จากสว่างที่สุดไปมืด) โดยมีลักษณะการเปลี่ยนแปลงของค่าความสว่างของแอลอีดีดังแสดงในรูปที่ 7



รูปที่ 7 ลักษณะการเปลี่ยนแปลงของค่าความสว่างของแอลอีดี ที่กำหนดในแบบฝึกหัด

บทเสริมเรื่อง
การควบคุมมอเตอร์ไฟตรง

SI1 มอเตอร์ไฟตรง

มอเตอร์เป็นอุปกรณ์ไฟฟ้าที่มีการใช้งานอย่างแพร่หลาย โดยทำหน้าที่เป็นส่วนประกอบหลักในการทำให้เกิดการเคลื่อนที่ของชิ้นส่วนต่าง ๆ ดังนั้นการศึกษาโครงสร้าง การทำงานและการควบคุมมอเตอร์ จึงเป็นส่วนหนึ่งที่มีความสำคัญ มอเตอร์จะทำหน้าที่แปลงพลังงานไฟฟ้าเป็นพลังงานกล โดยเราอาจแบ่งมอเตอร์ตามระบบไฟที่ใช้ได้ 2 ประเภท คือ

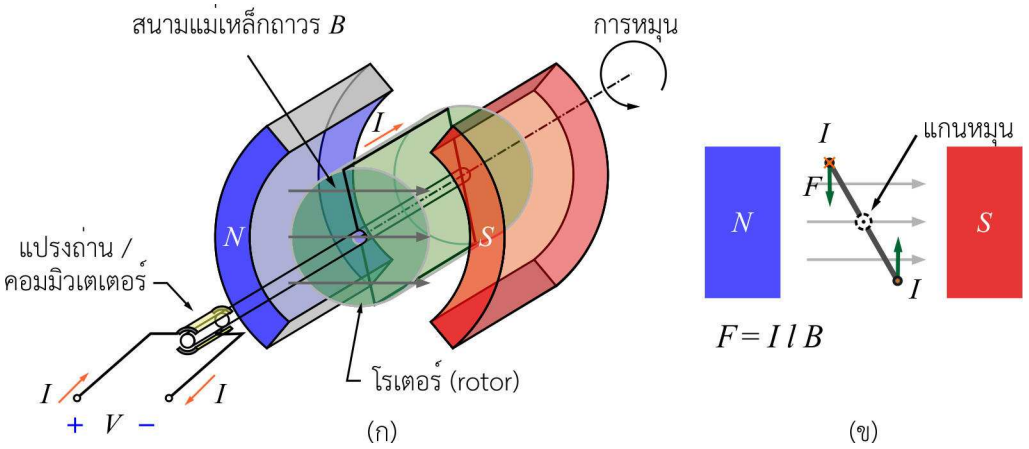
1. มอเตอร์ไฟฟ้ากระแสตรง (DC Motor)
2. มอเตอร์ไฟฟ้ากระแสสลับ (AC Motor)

โดยมอเตอร์ทั้งสองประเภทจะมีส่วนประกอบที่แตกต่างกันออกไปบ้างแต่ส่วนประกอบหลักคือจะมีส่วนที่อยู่กับที่เราเรียกว่า สเตเตอร์ (Stator) และส่วนที่เคลื่อนที่ซึ่งเราเรียกว่า โรเตอร์ (Rotor) มอเตอร์ไฟฟ้ากระแสตรงมีด้วยกันหลายแบบ สำหรับมอเตอร์ไฟฟ้ากระแสตรงอย่างง่ายคือมอเตอร์ที่ใช้แม่เหล็กถาวร

หลักการทำงาน

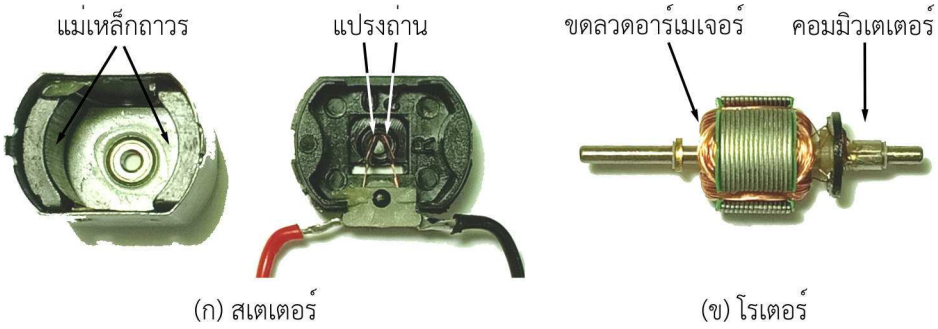
มอเตอร์จะอาศัยแรงดูดและแรงผลักที่เกิดจากสนามแม่เหล็กเมื่อมีกระแสไหลผ่านลวดตัวนำจะทำให้เกิดสนามแม่เหล็กรอบตัวนำนั้น ถ้าเรานำตัวนำดังกล่าวไปวางไว้ในสนามแม่เหล็กจากแม่เหล็กถาวรก็จะเกิดการต้านและเสริมกับเส้นแรงแม่เหล็กจากแม่เหล็กถาวร ทำให้เกิดแรงดูดและแรงผลักขึ้นที่ขดลวด รูปที่ 1(ก) แสดงลักษณะโครงสร้างของมอเตอร์ไฟตรงที่ใช้แม่เหล็กถาวรในส่วนของสเตเตอร์และมีการป้อนกระแสผ่านคอมมิวเตเตอร์ไปยังโรเตอร์ เพื่อให้แกนเหล็กที่โรเตอร์เป็นแม่เหล็กไฟฟ้า ดังแสดงในรูปที่ 1(ข) โดยกระแสไฟฟ้า I จะทำให้เกิดแรง F ขึ้น

จากรูปที่ 1 จะเห็นได้ว่าเมื่อจ่ายกระแสผ่านขั้วต่อที่เรียกว่าแปรงถ่าน (brushes) ไปยังวงแหวนพิเศษที่เรียกว่า คอมมิวเตเตอร์ (commutator) ซึ่งต่อเข้ากับวงรอบตัวนำกระแสที่ไหลผ่านตัวนำจะทำให้เกิดเส้นแรงแม่เหล็กรอบตัวนำโดยด้านหนึ่งจะเกิดเป็นแรงผลักขึ้น ทำให้วงรอบตัวนำมีการหมุน โดยแรงที่เกิดจะแปรตาม กระแสที่ไหลผ่าน I ความยาวของตัวนำ l และความเข้มของสนามแม่เหล็กถาวร B



รูปที่ 1 (ก) โครงสร้างและ (ข) การทำงานของมอเตอร์ไฟตรง

รูปที่ 2 แสดงภาพถ่ายลักษณะภายในของมอเตอร์ไฟตรงขนาดเล็กทั่วไป โดยรูปที่ 2(ก) แสดงส่วนของสเตเตอร์ซึ่งประกอบด้วยแม่เหล็กถาวรและส่วนของแปรงถ่านซึ่งเป็นส่วนที่อยู่นิ่งของขั้วที่ส่งกระแสไฟฟ้าเข้าไปยังขดลวดอาร์เมเจอร์ และ รูปที่ 2(ข) แสดงส่วนของโรเตอร์ โดยมอเตอร์ไฟตรงที่ใช้แม่เหล็กถาวรในทางปฏิบัติจะมีการเพิ่มความยาวของลวดตัวนำซึ่งเป็นส่วนของขดลวดบนโรเตอร์โดยการพันขดลวดรอบแกนโลหะ ซึ่งเรามักเรียกขดลวดนี้ว่า ขดลวดอาร์เมเจอร์ (armature) เพื่อเป็นการเพิ่มแรงบิดให้กับมอเตอร์ คอมมิว-เตเตอร์เป็นส่วนที่หมุนของขั้วที่รับกระแสไฟฟ้าเข้าไปยังขดลวดอาร์เมเจอร์ที่อยู่บนโรเตอร์ ดังแสดงในรูปที่ 2(ข)



รูปที่ 2 ภาพถ่ายภายในมอเตอร์ไฟตรง (ก) โครงสร้างและ (ข) การทำงานของมอเตอร์ไฟตรง

วิธีการควบคุมมอเตอร์ทั่วไป อาจแบ่งตามชนิดของปริมาณที่ต้องการควบคุม คือ

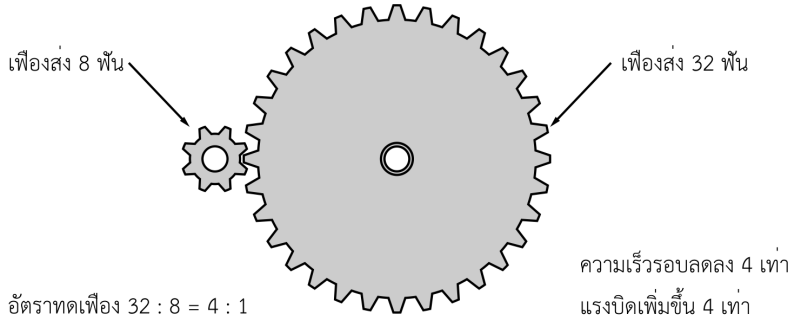
1. การควบคุมทิศทางการหมุนของมอเตอร์ (direction control)
2. การควบคุมความเร็วของมอเตอร์ (speed control)
3. การควบคุมแรงบิดของมอเตอร์ (torque control)

สำหรับการใช้งานเบื้องต้นที่จะกล่าวถึงในเอกสารนี้ จะแนะนำเพียงการควบคุมทิศทางการหมุนและการควบคุมความเร็วของมอเตอร์ในลักษณะอย่างง่ายเท่านั้น ในการใช้งานมอเตอร์ขั้นสูง จะต้องมีการใช้ระบบควบคุมและมีการป้อนกลับร่วมด้วย ซึ่งศาสตร์ด้านการควบคุมมอเตอร์เป็นสิ่งที่ศึกษากันในสาขาวิศวกรรมไฟฟ้าและวิศวกรรมเครื่องกล

ชุดเฟืองขับมอเตอร์

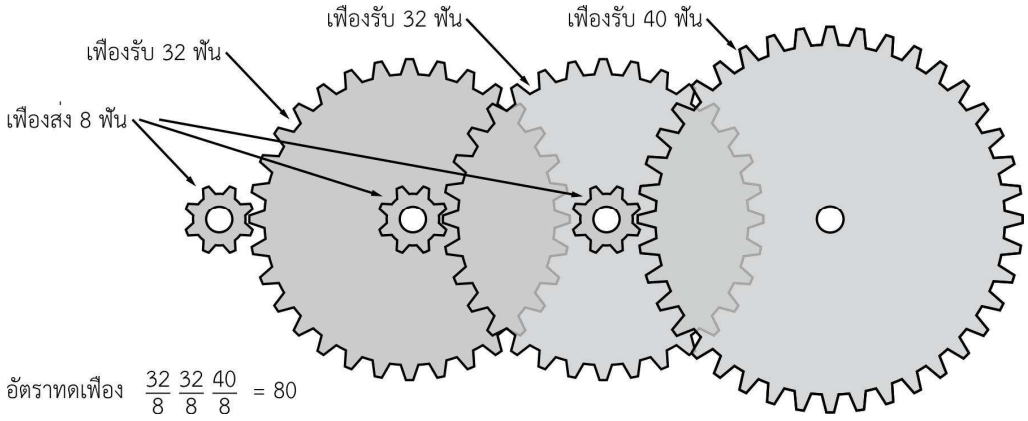
การขับเคลื่อนให้ระบบทางกลเคลื่อนไหวได้นั้นมักมีต้นกำลังทางกลจากมอเตอร์ แต่โดยลำพังมอเตอร์เพียงอย่างเดียวอาจไม่สามารถทำงานระบบทำงานได้อย่างเหมาะสม เนื่องจากความเร็วในการหมุนของมอเตอร์ทั่วไปมีค่าค่อนข้างสูง เราจึงจำเป็นต้องมีการปรับความเร็วรอบด้วยชุดเฟืองขับ (gearbox)

อัตราทดเฟือง คือสัดส่วนของจำนวนฟันเฟือง 2 ตัว โดยเราจะเรียกเฟืองตัวแรกของระบบว่าเฟืองส่ง เฟืองที่ต่อเพิ่มเข้ามาและนำส่งแรงบิดต่อไปเรียกว่าเฟืองรับ ถ้าเฟืองรับมีขนาดใหญ่กว่าเฟืองส่งจะทำให้แรงบิดเพิ่มขึ้นและความเร็วลดลง จากรูปตัวอย่างในรูปที่ 3 จะได้ว่า ถ้าเฟืองส่งหมุนไป 4 รอบ ฟันเฟืองจะทำให้เฟืองรับหมุนเพียงรอบเดียวเท่านั้น จากลักษณะนี้บอกได้ว่าเฟืองเกียร์มีอัตราทด 4:1 ซึ่งมาจากการนำค่าจำนวนฟันเฟืองของเฟืองรับตั้งแล้วหารด้วยค่าจำนวนฟันเฟืองของเฟืองส่ง



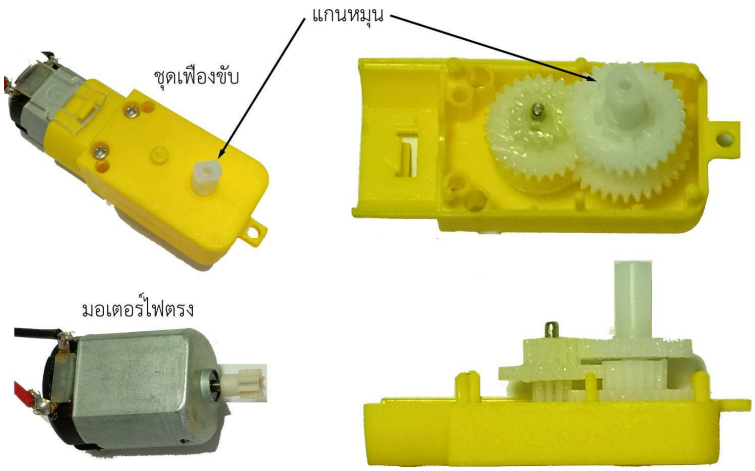
รูปที่ 3 ตัวอย่างการทดรอบด้วยเฟือง โดยมีอัตราทด 4:1

ในกรณีที่ต้องการให้อัตราทดเฟืองมีค่าสูงมาก ๆ เช่น 80:1 หากเป็นระบบเฟืองเพียง 1 ชั้น ถ้าเฟืองส่งมีขนาด 8 ฟันจะต้องสร้างเฟืองรับมีขนาดถึง 640 ฟัน ซึ่งในทางปฏิบัติเราไม่สามารถทำเช่นนั้นได้ เพราะต้องใช้เฟืองที่มีขนาดใหญ่ เราจึงต้องมีการสร้างระบบทดเฟืองเป็นชั้น ๆ ต่อกันไปจนได้อัตราทดที่ต้องการ การคำนวณหาอัตราทดทำได้ดังแสดงในตัวอย่างในรูปที่ 4



รูปที่ 4 ตัวอย่างการทดเฟืองหลายชั้น

รูปที่ 5 แสดงภาพภายในกล่องเฟืองขับที่มักจะติดตั้งอยู่กับมอเตอร์ไฟตรงขนาดเล็ก โดยกล่องเฟืองขับนี้มีอัตราทดเฟืองรวมมีค่าเท่ากับ 48 (จากระบบเฟืองที่ซ้อนกันอยู่ภายใน)

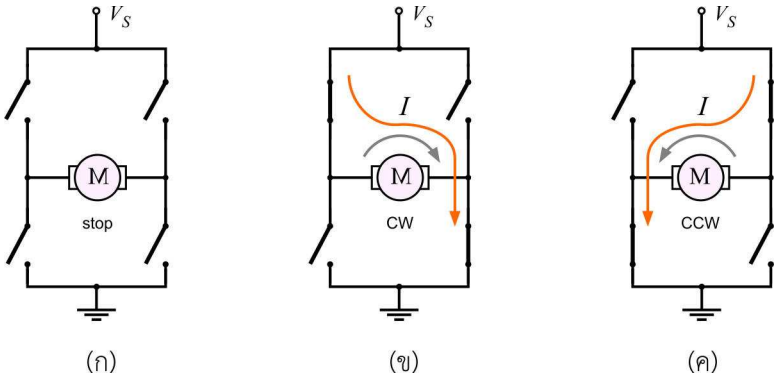


รูปที่ 5 ภาพส่วนประกอบภายในชุดเฟืองขับ

SI2 การควบคุมมอเตอร์ไฟตรง

การควบคุมทิศทางการหมุน

การควบคุมทิศทางการหมุนของมอเตอร์ไฟตรง ทำได้โดยการควบคุมทิศการไหลของกระแสไฟตรงที่ผ่านมอเตอร์ นั่นคือ มอเตอร์จะหมุนกับทิศทางหากมีการกลับทิศทางการฉีดกระแส โดยรูปแบบการต่อสวิตช์สำหรับควบคุมการหมุนของมอเตอร์นี้ แสดงดังรูปที่ 6(ก) และมีชื่อเรียกว่าวงจรเฮดบริดจ์ (H-bridge circuit) เนื่องจากวงจรมีลักษณะเป็นรูปตัว H โดยในการเปลี่ยนทิศทางการหมุนของมอเตอร์ (รูปที่ 6(ข) และ 6(ค)) สวิตช์ทั้งสี่ตัวจะต้องถูกสลับพร้อม ๆ กัน เพื่อมิให้มอเตอร์หยุดหมุนและไม่ให้เกิดการลัดวงจร ในทางปฏิบัติเรามักจะใช้ไอซีสำเร็จรูปในการควบคุมการหมุนของมอเตอร์เพื่อหลีกเลี่ยงปัญหาการลัดวงจร

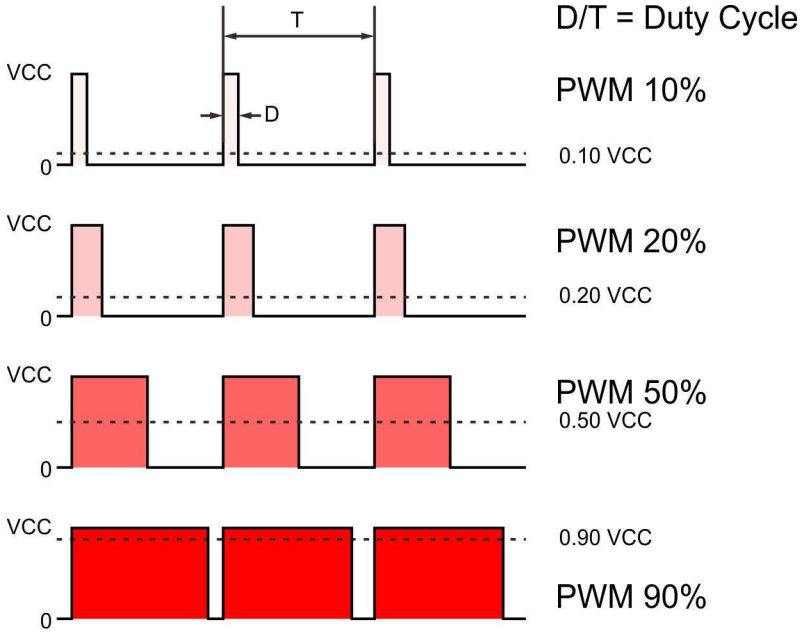


(ก) (ข) (ค)

รูปที่ 6 ภาพลักษณะวงจรเฮดบริดจ์ขณะ (ก) มอเตอร์ไม่หมุน (ข) มอเตอร์หมุนตามเข็มนาฬิกา (clockwise, CW) และ (ค) มอเตอร์หมุนทวนเข็มนาฬิกา (counter clockwise, CCW)

การควบคุมความเร็วมอเตอร์

ในการควบคุมความเร็วของมอเตอร์ไฟตรงสามารถทำได้การใช้สัญญาณ PWM (Pulse Width Modulation, การมอดูเลตความกว้างพัลส์) โดยกำหนดความกว้างของสัญญาณจะทำให้มอเตอร์ซึ่งเป็นอุปกรณ์ทางกลที่มีการตอบสนองช้าเมื่อเทียบกับสัญญาณไฟฟ้า ‘รู้สึก’ ถึงค่าเฉลี่ยของสัญญาณ และทำให้ความเร็วในการหมุนของมอเตอร์ซึ่งแปรผันโดยตรงกับระดับแรงดันมีค่าเปลี่ยนแปลงไป รูปที่ 7 แสดงตัวอย่างสัญญาณ PWM และค่าเฉลี่ย ซึ่งมีค่าเท่ากับผลคูณของค่าแรงดันและค่ารอบการทำงาน (duty cycle) ของสัญญาณ



รูปที่ 7 ลักษณะสัญญาณ PWM ที่รอบการทำงานต่าง ๆ
เส้นประแสดงระดับค่าเฉลี่ยของสัญญาณนี้

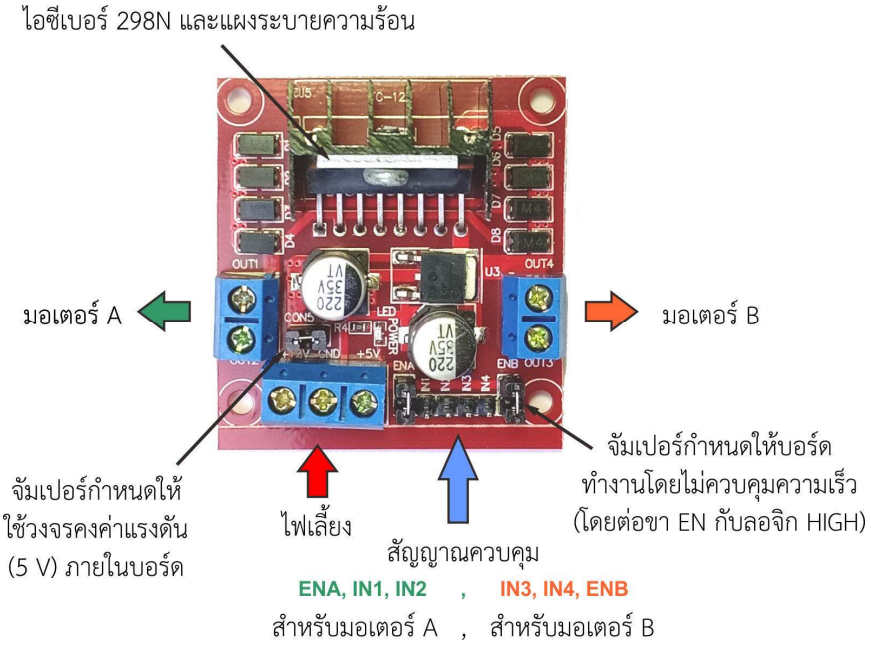
ในการเขียนโปรแกรมด้วย Arduino IDE คำสั่ง analogWrite() เป็นคำสั่งที่ใช้ในการสร้างสัญญาณ PWM จากบอร์ดไมโครคอนโทรลเลอร์ โดยคำสั่งนี้มีรูปแบบคือ

analogWrite(pin, value)

โดยค่า pin คือชื่อขาดิจิทัล (D0-D8) ที่ต้องการส่งสัญญาณ PWM ออกไป และ ค่า value นี้จะเป็นค่าเลขฐานสิบ ระหว่าง 0 ถึง 1023 (= 2^{10}-1) ซึ่งค่าสูงสุด 1023 นี้จะเฉพาะเจาะจงกับบอร์ด NodeMCU เนื่องจาก ESP8266 เป็นชิปประมวลผลแบบ 10 บิต โดยในโค้ดต่าง ๆ ในไลบรารีของบอร์ดนี้ จะกำหนดให้ค่า 1023 คือตัวแปรชื่อ PWMRANGE และค่ารอบการทำงานของสัญญาณที่ส่งออกมาจะขึ้นกับค่า value นี้ โดยสัญญาณจะมีค่าคาบคงที่เท่ากับ 1 มิลลิวินาที (ความถี่ 1 kHz) สำหรับชิป ESP8266 นี้

บอร์ดขับมอเตอร์

ในการขับมอเตอร์ด้วยคำสั่งจากไมโครคอนโทรลเลอร์นั้นเรามักจะต้องใช้บอร์ดขับมอเตอร์แยกจากบอร์ดไมโครคอนโทรลเลอร์ และใช้ไฟเลี้ยงมอเตอร์แยกจากไฟเลี้ยงบอร์ดไมโครคอนโทรลเลอร์ เนื่องจากไมโครคอนโทรลเลอร์ทั่วไป ไม่ได้ออกแบบมาให้สามารถจ่ายกระแสสูง ๆ สำหรับการทำงานของมอเตอร์ โดยในที่นี้เราจะกล่าวถึงบอร์ดขับมอเตอร์ที่ใช้ไอซีเบอร์ L298N (จ่ายกระแสได้สูงสุด 2 A/มอเตอร์) ซึ่งบอร์ดนี้สามารถขับมอเตอร์ได้ 2 ตัวพร้อมกันและมีลักษณะดังรูปที่ 8



รูปที่ 8 บอร์ดขับมอเตอร์ L298N

ในการควบคุมการหมุนและความเร็วของมอเตอร์นั้น ทำโดยการป้อนสัญญาณควบคุมจาก บอร์ดไมโครคอนโทรลเลอร์ไปยังบอร์ดขับมอเตอร์ โดยมีลักษณะของสัญญาณควบคุมดังแสดงในตารางที่ 1 โดยหากเราจะควบคุมทิศทางการหมุนของมอเตอร์ A ก็ทำได้ผ่านสัญญาณที่ขา IN1 และ IN2 และการควบคุมความเร็วของมอเตอร์ด้วยสัญญาณ PWM ที่ป้อนให้กับขา ENA ซึ่งหากไม่ต้องการควบคุมความเร็ว ก็ทำได้โดยการใช้จัมเปอร์บนบอร์ดแทนการป้อนสัญญาณ และสำหรับมอเตอร์ B เราก็สามารถทำได้ในทำนองเดียวกัน

ตารางที่ 1 ตารางแสดงวิธีการส่งสัญญาณควบคุมทิศทางและความเร็วมอเตอร์ A และ B

	ทิศการหมุน	IN1	IN2	IN3	IN4	ควบคุมความเร็วด้วยสัญญาณ PWM	
						ENA	ENB
มอเตอร์ A	ตามเข็ม	HIGH	LOW	-	-	HIGH	-
	ทวนเข็ม	LOW	HIGH	-	-	HIGH	-
	ไม่หมุน	LOW	LOW	-	-	HIGH	-
	ไม่หมุน	HIGH	HIGH	-	-	HIGH	-
มอเตอร์ B	ตามเข็ม	-	-	HIGH	LOW	-	HIGH
	ทวนเข็ม	-	-	LOW	HIGH	-	HIGH
	ไม่หมุน	-	-	LOW	LOW	-	HIGH
	ไม่หมุน	-	-	HIGH	HIGH	-	HIGH

SI3 การทดลองควบคุมมอเตอร์ไฟตรง

วัตถุประสงค์

1. สามารถต่อบอร์ด Node MCU v.3 กับบอร์ดขับมอเตอร์และมอเตอร์ได้
2. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมทิศทางการหมุนของมอเตอร์ไฟตรงได้
3. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมความเร็วในการหมุนของมอเตอร์ไฟตรงได้

อุปกรณ์ที่ใช้ในการทดลอง

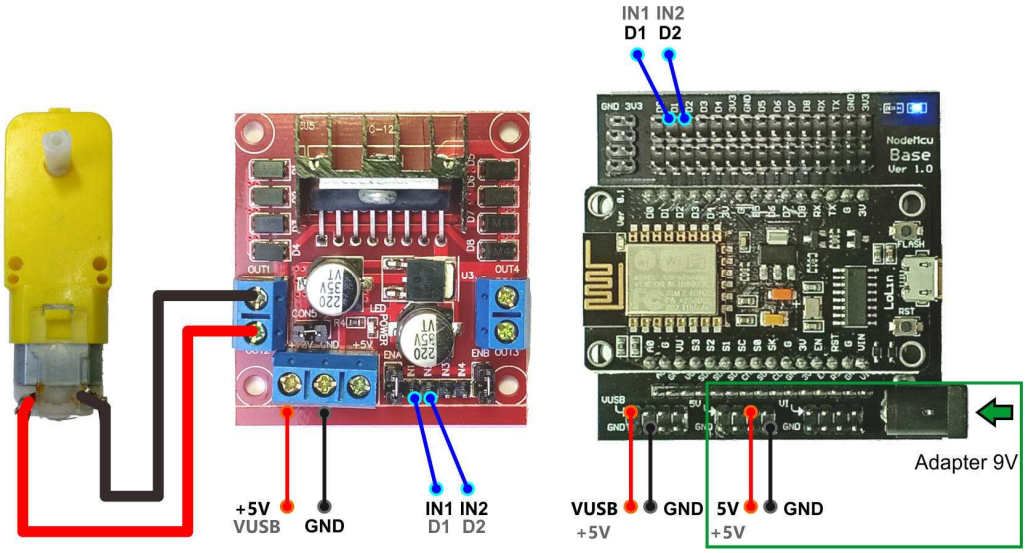
1. เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT 1 เครื่อง
2. NodeMCU v.3 1 บอร์ด
3. NodeMCU Base Ver 1.0 1 บอร์ด
4. บอร์ดขับมอเตอร์ที่ใช้ไอซีเบอร์ L298N 1 บอร์ด

5.	มอเตอร์ไฟตรง	1 ตัว
6.	อะแด็ปเตอร์ 9 V	1 ตัว
7.	สาย USB	1 เส้น
8.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	3 เส้น
9.	สายต่อวงจร (สายจัมพ์ ผู้-ผู้)	2 เส้น
10.	สายต่อวงจร (สายจัมพ์ ผู้-เมีย)	2 เส้น

วิธีการทดลอง

ตอนที่ 1 การควบคุมทิศทางการหมุนของมอเตอร์

1. ต่อวงจรดังรูปที่ 9
2. เขียนโปรแกรมดังที่แสดงในโค้ดหน้าถัดไป จากนั้นจึงอัปโหลดลง NodeMCU v.3 แล้วสังเกตทิศทางการหมุนของมอเตอร์ โดยหากมอเตอร์ไม่หมุนอย่างต่อเนื่อง เนื่องจากไฟเลี้ยงจาก USB ไม่เพียงพอ ก็ขอให้ใช้อะแด็ปเตอร์ร่วมในการจ่ายไฟไปยังบอร์ดขับมอเตอร์



รูปที่ 9 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดขับมอเตอร์และมอเตอร์ไฟตรง โดยในกรอบสี่เหลี่ยมสีเขียวคือการต่อสายในกรณีที่ไฟเลี้ยงจากพอร์ต USB นั้นไม่เพียงพอ

```
1  // Control DC Motor via L298N by NodeMCU ESP8266
2
3  int IN1 = D1;
4  int IN2 = D2;
5
6  void setup() {
7      pinMode(IN1, OUTPUT);
8      pinMode(IN2, OUTPUT);
9  }
10
11 void loop() {
12     digitalWrite(IN1, HIGH);
13     digitalWrite(IN2, LOW);
14 }
15
```

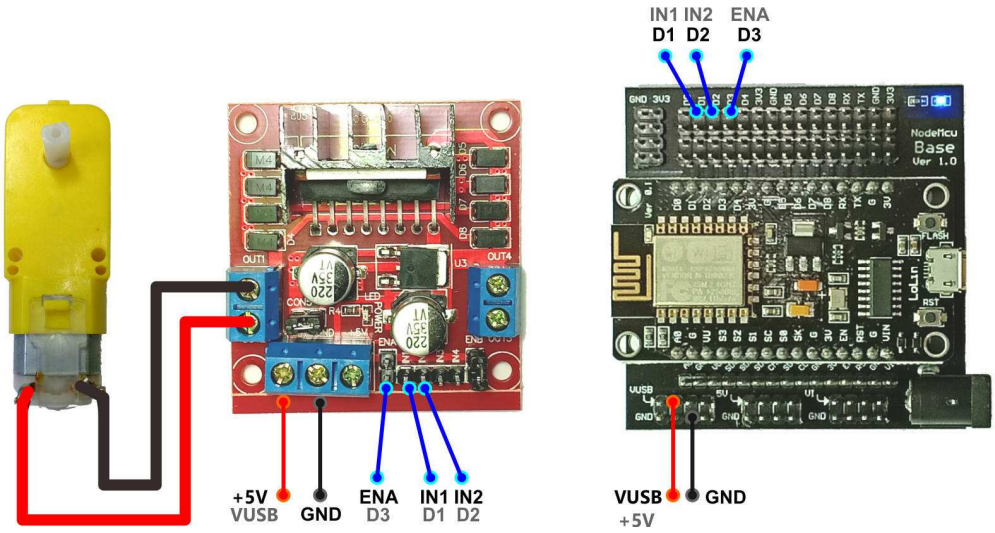
3. แก้ไขโปรแกรมบรรทัดที่ 12 และ 13 โดยสั่งให้ขา IN1 เป็นลอจิก LOW และ IN2 เป็นลอจิก HIGH จากนั้นจึงอัปโหลดโปรแกรมใหม่ แล้วสังเกตการหมุนของมอเตอร์

```
12     digitalWrite(IN1, LOW);
13     digitalWrite(IN2, HIGH);
```

ตอนที่ 2 การควบคุมความเร็วมอเตอร์

1. ต่อวงจรดังรูปที่ 10 โดยเอาจัมเปอร์ที่ขา ENA ออกก่อน
2. เขียนโปรแกรมดังที่แสดงในโค้ดหน้าถัดไป จากนั้นจึงอัปโหลดลง NodeMCU v.3 แล้วสังเกตความเร็วในการหมุนของมอเตอร์

จากการทดลองในข้อ 2 นี้ จะสังเกตได้ว่า มอเตอร์จะส่งเสียงแหลมออกมาในขณะที่ยังไม่หมุน เนื่องจากแรงดันและกระแสที่จ่ายให้กับมอเตอร์ยังไม่เพียงพอที่จะเอาชนะแรงเสียดทานสถิตย์ในโครงสร้างทางกล และเมื่อมอเตอร์เริ่มหมุนแล้ว ก็จะหมุนเร็วขึ้นเรื่อย ๆ แล้วหยุดและเริ่มจึงหมุนใหม่ ตามวงรอบการทำงานในฟังก์ชัน loop()

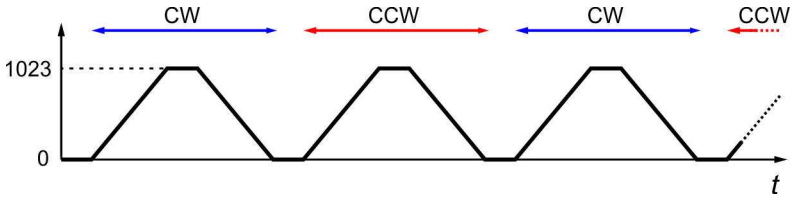


รูปที่ 10 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดขับมอเตอร์
และมอเตอร์ไฟตรงเพื่อควบคุมความเร็ว

```
// Control DC Motor via L298N by NodeMCU ESP8266

int IN1 = D1;
int IN2 = D2;
int ENA = D3;

void setup() {
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENA, OUTPUT);
}

void loop() {
    for (int speed=0; speed<1024; speed++) {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(ENA,speed);
        delay(10);
    }
}

```

แบบฝึกหัดท้ายการทดลอง

จงเขียนโค้ดที่ควบคุมให้มอเตอร์ค่อย ๆ เริ่มหมุนตามเข็มนาฬิกา จากหยุดนิ่งไปเร็วที่สุด แล้วค่อย ๆ หยุดลง (จากเร็วที่สุดไปหยุดนิ่ง) แล้วจึงหมุนทวนเข็มนาฬิกาจากหยุดนิ่งไปเร็วที่สุด แล้วค่อย ๆ หยุดลง (จากเร็วที่สุดไปหยุดนิ่ง) โดยมีลักษณะการเปลี่ยนแปลงของค่าความเร็วแบบเชิงเส้นดังแสดงในรูปที่ 11



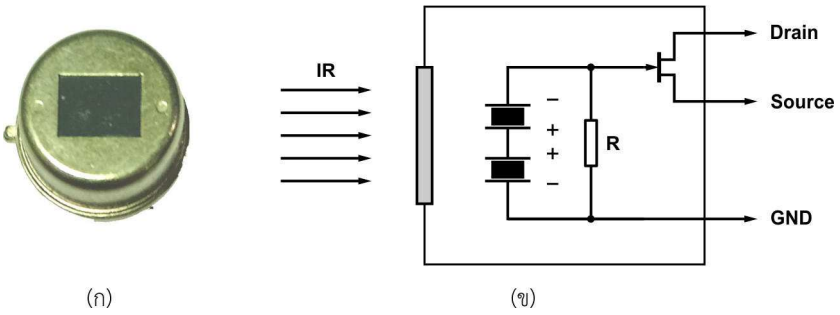
รูปที่ 11 ลักษณะการเปลี่ยนแปลงของค่าความเร็วมอเตอร์ที่กำหนดในแบบฝึกหัด

บทเสริมเรื่อง
การใช้งานโมดูลเซนเซอร์ตรวจจับการเคลื่อนไหว

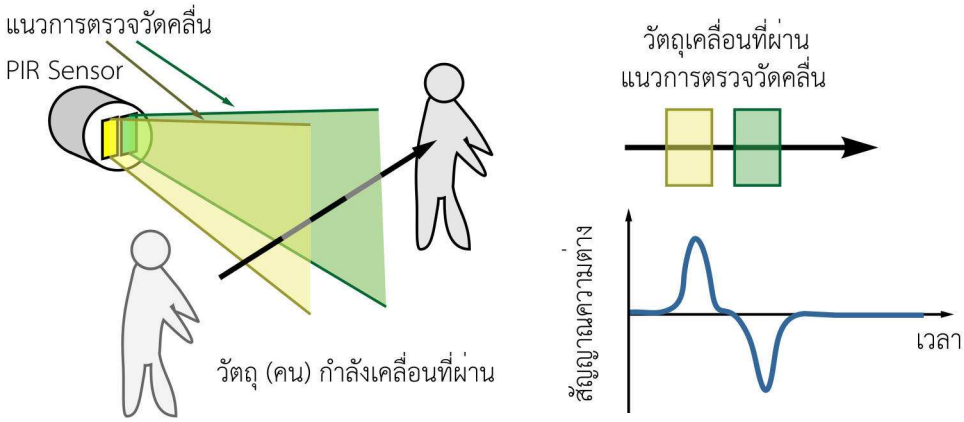
SI1 เซนเซอร์ตรวจจับการเคลื่อนไหว

การตรวจจับการเคลื่อนไหวที่จะกล่าวถึงในบทนี้ คือการตรวจจับการเคลื่อนไหวของมนุษย์ด้วยเซนเซอร์ชนิดไพโรอิเล็กทริกอินฟราเรด (pyroelectric infrared sensor) หรือมีอีกชื่อหนึ่งว่าเซนเซอร์อินฟราเรดชนิดพาสซิฟ (passive infrared sensor) และเรียกย่อ ๆ ว่า พีไออาร์เซนเซอร์ (PIR sensor) โดยเซนเซอร์นี้สามารถแปลงคลื่นแสงอินฟราเรด (infrared, IR) ที่เกิดจากการแผ่รังสีจากวัตถุที่มีอุณหภูมิสูงกว่าสิ่งแวดล้อมให้เป็นสัญญาณทางไฟฟ้าได้ รูปที่ 1(ก) แสดงภาพถ่ายของเซนเซอร์ชนิดนี้

พีไออาร์เซนเซอร์มีส่วนประกอบภายในที่สำคัญคือ ภายในตัวเซนเซอร์จะมีช่องรับคลื่นแสงอินฟราเรดสองช่อง (ดูรูปที่ 1(ข)) ซึ่งเปรียบได้ว่าภายในมีเซนเซอร์สองตัวที่ต่ออนุกรมกันอยู่ภายใน และมีวงจรทรานซิสเตอร์ (เจเฟต) อยู่ด้วย เซนเซอร์ที่ใช้นี้จะตรวจจับคลื่นแสงอินฟราเรดที่เปล่งออกมาจากสิ่งมีชีวิต เนื่องจากสิ่งที่มีความร้อนทุกชนิดจะเปล่งรังสีอินฟราเรดออกมา นั่นคือ ในขณะที่ไม่มีการเคลื่อนไหว เซนเซอร์ทั้งสองช่องจะได้รับสัญญาณพื้นหลังจากสิ่งของต่าง ๆ เท่ากันทำให้ไม่มีความต่างของสัญญาณจากเซนเซอร์ทั้งสอง แต่เมื่อมีวัตถุที่เปล่งคลื่นแสงอินฟราเรดเคลื่อนที่ผ่านเซนเซอร์ก็จะก่อให้เกิดสัญญาณความต่างขึ้น เนื่องจากเซนเซอร์ตรวจจับสัญญาณจากวัตถุได้ไม่เท่ากันในขณะที่วัตถุเคลื่อนที่ โดยสัญญาณนี้จะเป็นค่าบวกและลบสลับกันตามลำดับเวลา รูปที่ 2 แสดงลักษณะการเคลื่อนที่ผ่านของวัตถุและลักษณะสัญญาณที่ได้รับ โดยเมื่อได้รับสัญญาณแล้ว วงจรประมวลผลก็จะเอาค่าสูงสุดของสัญญาณมาเปรียบเทียบกับค่าเทรสโฮลด์ (threshold value) ในลำดับถัดไป เพื่อบ่งบอกว่ามีการเคลื่อนที่ของวัตถุหรือไม่

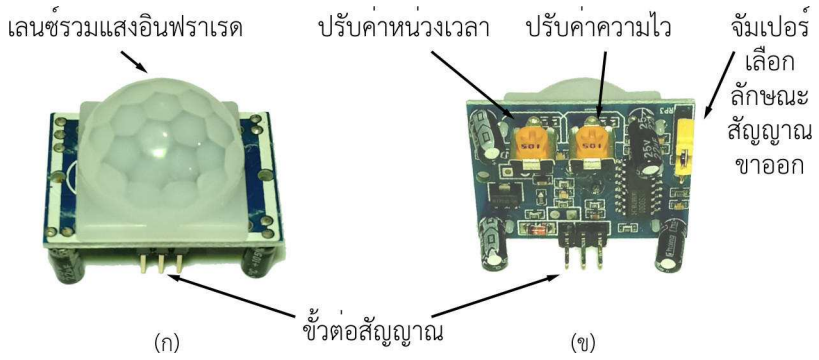


รูปที่ 1 (ก) ภาพถ่ายและ (ข) ลักษณะวงจรภายในของพีไออาร์เซนเซอร์



รูปที่ 2 หลักการและสัญญาณจากการตรวจจับการเคลื่อนไหวด้วยพีไออาร์เซนเซอร์

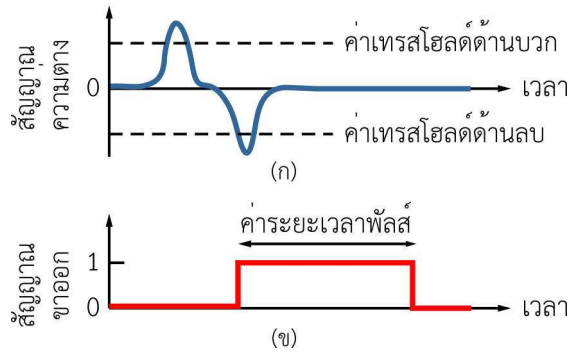
ในการตรวจจับการเคลื่อนไหวนั้น เรามักจะต้องการให้สัญญาณคลื่นแสงอินฟราเรดมีลักษณะเป็นลำเข้ามายังเซนเซอร์ (เหมือนตัวตรวจจับแสงชนิดอื่น ๆ) ดังนั้น เซนเซอร์นี้จึงมักจะมาพร้อมกับเลนซ์สำหรับรวมคลื่นแสงอินฟราเรด และในทางข้อมูลซึ่งเป็นสัญญาณไฟฟ้านั้น เรามักจะต้องการเพียงข้อมูลดิจิทัลที่บ่งบอกว่า มีการเคลื่อนไหวเกิดขึ้นหรือไม่ ดังนั้น พีไออาร์เซนเซอร์จึงมีการใช้งานในลักษณะเป็นโมดูลที่มาพร้อมกับเลนซ์ที่ติดตั้งด้านหน้าและวงจรดิจิทัลที่อยู่ด้านหลังและที่ทำหน้าที่ประมวลสัญญาณในเบื้องต้นเพื่อส่งต่อให้แก่ไมโครคอนโทรลเลอร์ รูปที่ 3 แสดงลักษณะของพีไออาร์เซนเซอร์โมดูลที่มีขายทั่วไปในท้องตลาด



รูปที่ 3 พีไออาร์เซนเซอร์โมดูล (ก) ด้านหน้า และ (ข) ด้านหลัง

พีไออาร์เซนเซอร์โมดูลนี้จะรับไฟเลี้ยง (VCC และ GND) และส่งสัญญาณขาออก (Output Signal) แบบดิจิทัล ที่บ่งบอกว่าตรวจพบการเคลื่อนไหวหรือไม่ออกไป ในโมดูลบางรุ่น (เช่นรุ่น HC-SR501 ดังแสดงในรูปที่ 3) จะมีตัวต้านทานปรับค่าได้สำหรับปรับค่าความไวของการตรวจจับ (sensitivity) ซึ่งก็คือการปรับค่าเทรสโฮลด์ และปรับค่าระยะเวลาของพัลส์ (time duration) สัญญาณดิจิทัลที่บ่งบอกว่าตรวจพบการเคลื่อนไหวแล้ว รูปที่ 4 แสดงลักษณะสัญญาณจากเซนเซอร์เทียบกับสัญญาณอ้างอิงและสัญญาณดิจิทัลขาออกซึ่งมีค่าระยะเวลาพัลส์ที่ปรับเปลี่ยนได้

สำหรับจัมเปอร์ที่ใช้ในการเลือกลักษณะของสัญญาณขาออกนั้น (ดูรูปที่ 3(ข)) จะเป็นการเลือกให้การตรวจจับการเคลื่อนที่มีความต่อเนื่องหรือไม่ นั่นคือจัมเปอร์นี้สามารถกำหนดให้ค่าระยะเวลาพัลส์สามารถเพิ่มขึ้น หากมีการเคลื่อนไหวอย่างต่อเนื่อง โดยการเซตในรูปที่ 3(ข) นั้นเป็นลักษณะที่สัญญาณเป็นพัลส์เดี่ยว มีระยะเวลาคงที่ ตามค่าที่ปรับด้วยค่าตัวต้านทานปรับค่าหน่วงเวลา ซึ่งหากเลื่อนจัมเปอร์นี้มาที่คู่ล่าง จะเป็นการกำหนดให้พัลส์นี้สามารถยาวนานขึ้นได้ หากเซนเซอร์ยังคงตรวจพบการเคลื่อนไหวต่อไป (หลังจากพบในการเคลื่อนไหวครั้งแรกแล้ว)



รูปที่ 4 ลักษณะ (ก) สัญญาณความต่างเทียบกับค่าเทรสโฮลด์ และ (ข) สัญญาณขาออกของโมดูลเซนเซอร์ตรวจจับการเคลื่อนไหว

SI2 การเขียนโปรแกรมรับค่าดิจิทัล

คำสั่งในการใช้งานพีไออาร์เซนเซอร์โมดูลใน ภาษา Arduino คือ ฟังก์ชันอินพุตแบบดิจิทัล (Digital Input) ซึ่งมีฟังก์ชันที่ต้องทราบ 2 ฟังก์ชัน คือ **pinMode(pin, mode)** และคำสั่ง **digitalRead(pin)** เป็นคำสั่งให้อ่านสถานะทางลอจิกของขาที่ต่อกับเซนเซอร์ โดยค่าที่ส่งคืนมาคือ ลอจิกสูง (HIGH หรือ ค่าตัวเลข “1”) หรือ ลอจิกต่ำ (LOW หรือ ค่าตัวเลข “0”) เท่านั้น ขึ้นกับสถานะการตรวจจับการเคลื่อนไหวของเซนเซอร์ โดยการใช้งานคำสั่งนี้ สามารถศึกษาเพิ่มเติมได้จากบทเสริมที่ 1 การใช้งานสวิตช์

SI3 การทดลองอ่านค่าจากพีไออาร์เซนเซอร์โมดูล

วัตถุประสงค์

1. สามารถต่อบอร์ด NodeMCU v.3 กับพีไออาร์เซนเซอร์โมดูลได้
2. สามารถเขียนโปรแกรมให้ NodeMCU รับค่าจากเซนเซอร์โมดูลและแสดงผลผ่านพอร์ตอนุกรมได้
3. สามารถปรับเปลี่ยนค่าความไวในการตรวจจับการเคลื่อนไหว ค่าระยะเวลาพัลส์ และโหมดการส่งสัญญาณออกของพีไออาร์เซนเซอร์โมดูลได้

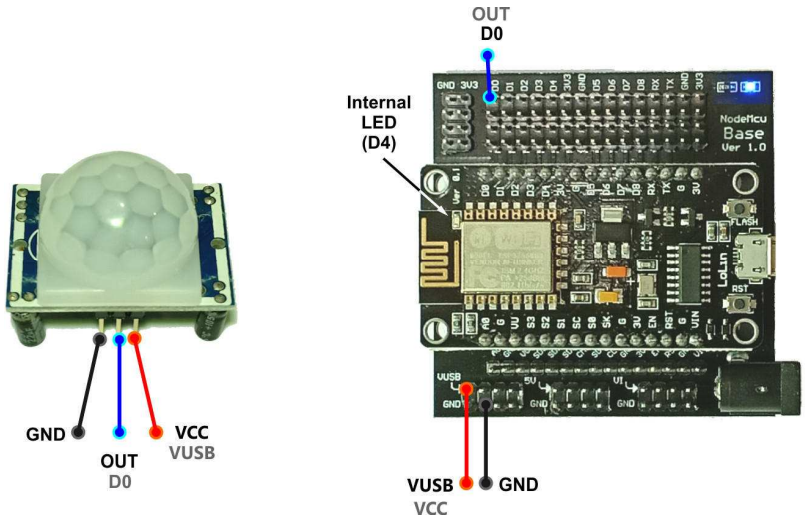
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	พีไออาร์เซนเซอร์โมดูล HC-SR501	1 บอร์ด
5.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

วิธีการทดลอง

1. ต่อวงจรตามรูปที่ 5

2. เขียนโค้ดข้างล่างนี้แล้วอัปโหลดลง NodeMCU v.3 เพื่อแสดงผลการตรวจจับการเคลื่อนไหวผ่านพอร์ตอนุกรม

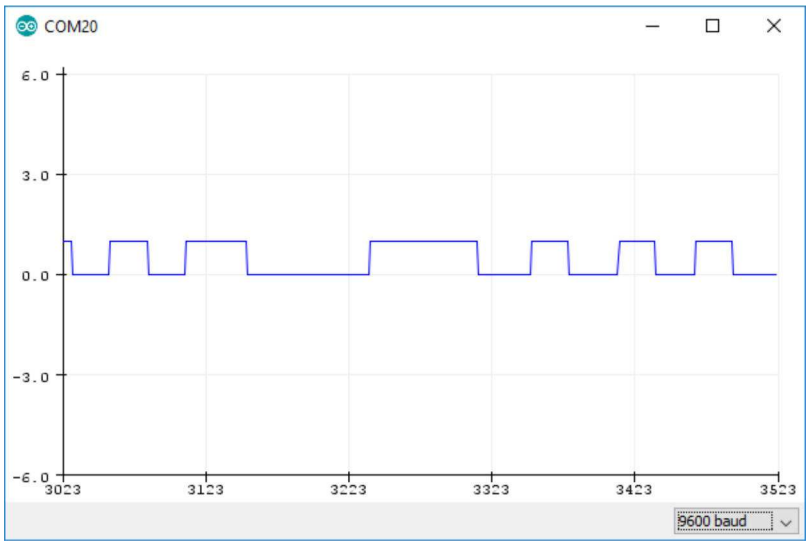


รูปที่ 5 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดพีไออาร์เซนเซอร์โมดูล

3. เปิด Serial Monitor (Ctrl+Shift+M) หรือ Serial Plotter (Ctrl+Shift+L) และสังเกตแอลอีดีบนบอร์ด โดยสังเกตผลที่เกิดขึ้นเมื่อมีการเคลื่อนไหวผ่านโมดูลเซนเซอร์

4. ทำการทดลองปรับตัวต้านทานปรับค่าได้ที่อยู่บนโมดูล และสังเกตการเปลี่ยนแปลงของผลที่ได้รับ รูปที่ 5 แสดงตัวอย่างผลการทดลองที่สังเกตได้

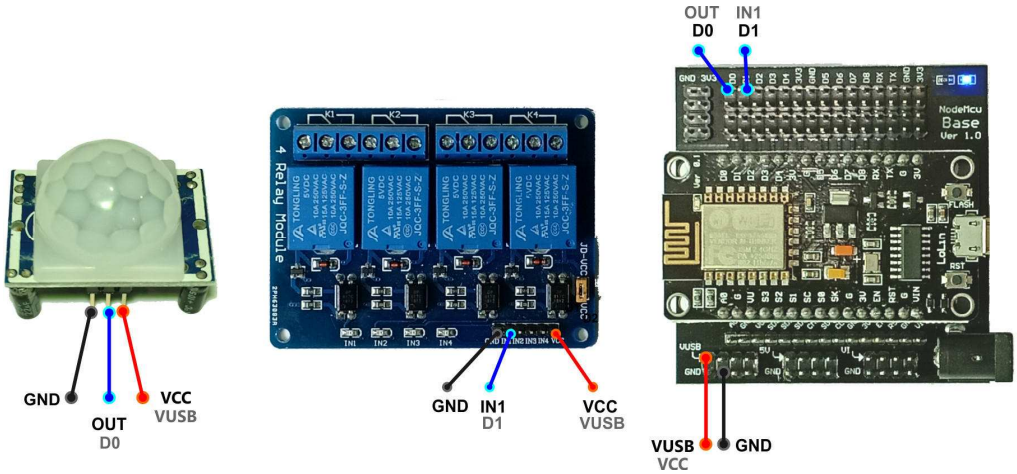
```
// Detect Motion with PIR Sensor Module
// by NodeMCU ESP8266

#define ON LOW      // LED is active low.
#define OFF HIGH

int PIR_PIN = D0;
int LED_PIN = D4;   // Internal LED

void setup() {
    Serial.begin(9600);
    pinMode(PIR_PIN, INPUT);
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    int st;    // st = state of PIR_Signal
    st = digitalRead(PIR_PIN);
    if(st == HIGH) {
        Serial.println(1);
        digitalWrite(LED_PIN, ON);
    } else {
        Serial.println(0);
        digitalWrite(LED_PIN, OFF);
    }
    delay(200);
}

```



รูปที่ 6 ตัวอย่างผลการทดลองที่สังเกตได้ผ่าน Serial Plotter

แบบฝึกหัดท้ายการทดลอง

ในการทดลอง หากเราต้องการให้สัญญาณจากพีไออาร์เซนเซอร์ ไปสั่งการทำงานของรีเลย์เปิดปิดไฟที่มีการต่อวงจรดังรูปที่ 7 จงเขียนโปรแกรมคำสั่งเพิ่มเติมในส่วนนี้



รูปที่ 7 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดพีไออาร์เซนเซอร์โมดูล และ บอร์ดรีเลย์

บทเสริมเรื่อง
การอ่านสัญญาณจากแก๊สเซนเซอร์

SI1 แก๊สเซนเซอร์และโมดูลแก๊สเซนเซอร์

แก๊สเซนเซอร์ (gas sensor) ที่ใช้งานจริงและมีขายในท้องตลาดปัจจุบัน มีด้วยกันหลายประเภท ขึ้นกับทั้งชนิดของแก๊สที่ต้องการตรวจสอบและรูปแบบการใช้งาน โดยในบทนี้เราจะกล่าวถึงเฉพาะโมดูลเซนเซอร์ ที่นิยมนำมาใช้ในการทำโครงงานสิ่งประดิษฐ์ โดยโมดูลเซนเซอร์นี้มีรูปร่างทั่วไป ดังแสดงในรูปที่ 1 โดยตัวเซนเซอร์มีด้วยกันหลากหลายประเภท (ขึ้นกับแก๊สที่ตรวจสอบ) ตัวอย่างเช่น

MQ-2 Flammable Gas & Smoke Sensor – เซนเซอร์ตรวจจับแก๊สติดไฟและควัน

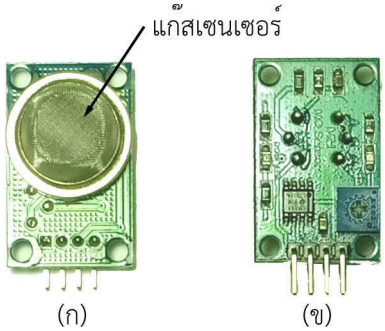
MQ-6 LPG Sensor – เซนเซอร์ตรวจจับแก๊สแอลพีจี (แก๊สหุงต้ม)

MQ-9 Carbon Monoxide Sensor – เซนเซอร์ตรวจจับก๊าซคาร์บอนมอนอกไซด์

MQ-135 Air Quality Sensor – เซนเซอร์ตรวจสอบคุณภาพอากาศโดยรวม

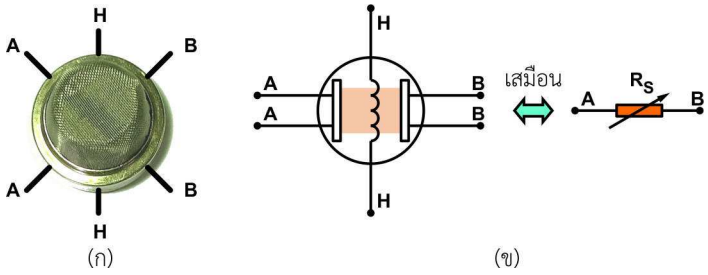
MG-811 Carbon Dioxide Sensor – เซนเซอร์ตรวจจับก๊าซคาร์บอนไดออกไซด์

รายละเอียดข้อมูลจำเพาะของการตรวจจับแก๊สของเซนเซอร์แต่ละประเภท สามารถศึกษาได้จาก datasheet ของเซนเซอร์นั้น ๆ



รูปที่ 1 ภาพถ่ายโมดูลแก๊สเซนเซอร์ทั่วไป (ก) ด้านหน้า และ (ข) ด้านหลัง

โมดูลแก๊สเซนเซอร์นี้ จะมีตัวเซนเซอร์เป็นอุปกรณ์หลัก มีขนาดค่อนข้างใหญ่และปกปิดด้วยตาข่ายกรองฝุ่นที่อาจมีลักษณะภายนอกแตกต่างกันบ้าง รูปที่ 2 แสดงตัวเซนเซอร์และลักษณะแผนผังภายในเซนเซอร์ โดยเซนเซอร์นี้จะมี 6 ขา โดยมีขา 2 ขา ที่ชื่อ H (มาจาก Heater) ที่ใช้ให้พลังงานความร้อนแก่ตัวเซนเซอร์ และอีก 4 ขา ที่ชื่อ A และ B และใช้ส่งสัญญาณออกไป โดยเมื่อเราจ่ายพลังงานให้แก่เซนเซอร์แล้ว ขดลวดความร้อนจะทำงาน ทำให้แก๊สที่ผ่านเข้ามาภายในเซนเซอร์ เกิดปฏิกิริยากับสารประกอบที่เคลือบบนผิวเซนเซอร์ (เช่น SnO_2) ซึ่งการตรวจจับแก๊สจะอาศัยหลักการการเปลี่ยนแปลงค่าความต้านทานของตัวเซนเซอร์ R_S โดยเราสามารถวัดค่าความต้านทานนี้โดยตรงได้จากขาทั้ง 4 (ขา A และ B) โดยค่าคามต้านทานของเซนเซอร์จะลดลงเมื่อค่าความเข้มข้นของแก๊สเพิ่มขึ้น ซึ่งค่าความต้านทานนี้จะขึ้นกับทั้งชนิดของแก๊ส อุณหภูมิและความชื้นที่มีในสภาพแวดล้อมด้วย ดังนั้นในการใช้งานจริง หากเราต้องการผลการวัดที่แน่นอน เราก็จะต้องทำการเทียบวัด (calibration) เสียก่อน นอกจากนี้ เราอาจต้องให้ความร้อนแก่เซนเซอร์วัดแก๊สบางประเภทเป็นเวลานานเสียก่อน ที่เรียกว่า preheat เพื่อให้เซนเซอร์เสถียรและอ่านค่าได้อย่างถูกต้อง



รูปที่ 2 (ก) แก๊สเซนเซอร์และ (ข) แผนผังภายในตัวเซนเซอร์ ที่เมื่อทำงานแล้วจะเป็นเสมือนตัวต้านทานปรับค่าได้

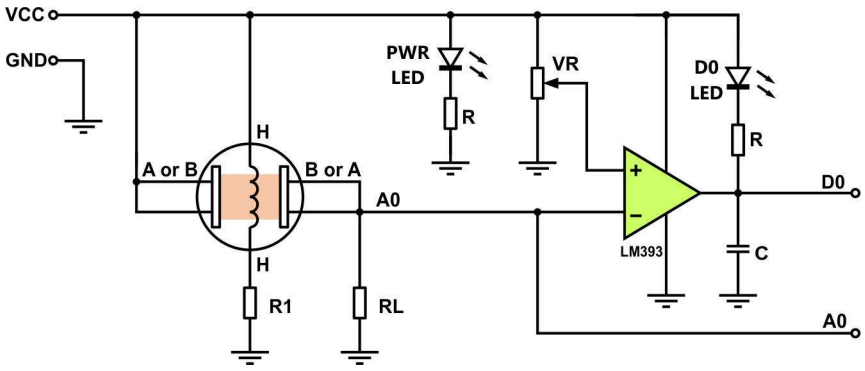
การตรวจวัดแก๊สจะทำได้โดยใช้วงจรแบ่งแรงดัน เพื่อแปลงค่าความต้านทาน (ที่ขึ้นกับค่าความเข้มข้นของแก๊ส) มาเป็นค่าแรงดัน โดยวงจรในโมดูลนี้ จะมีลักษณะดังรูปที่ 3 ซึ่งจะมีส่วนประกอบหลัก ๆ 2 ส่วน คือ

1. วงจรแบ่งแรงดัน (voltage divider) ที่ใช้ตัวต้านทานภายนอกอีกตัวหนึ่งมาต่อกับตัวเซนเซอร์ (คือ R_L ในรูปที่ 3) เพื่อแปลงค่าความต้านทาน R_S ของเซนเซอร์เป็นค่าแรงดันแอนะล็อก โดยค่าความต้านทานที่เหมาะสมในการนำมาต่อนี้จะขึ้นกับความต้านทานของตัวเซนเซอร์ด้วย สำหรับสัญญาณแอนะล็อก A0 ที่อ่านได้ จะมีค่าแรงดันคำนวณได้จาก

$$V_{A0} = \frac{R_S}{R_S + R_L} \times V_{CC}$$

โดยค่าแรงดันนี้ จะถูกแปลงเป็นข้อมูลดิจิทัลด้วยไมโครคอนโทรลเลอร์ที่นำมาอ่านค่านี้ในลำดับต่อไป ซึ่งสำหรับ NodeMCU นี้ ค่าแอนะล็อกจะมีค่าอยู่ระหว่าง 0 ถึง 1023

2. วงจรเปรียบเทียบแรงดัน (voltage comparator) เป็นวงจรที่ใช้ในการเปลี่ยนค่าแอนะล็อกที่วัดได้จากวงจรแบ่งแรงดันเป็นค่าดิจิทัล ขนาด 1 บิต (คือ “0” หรือ “1” เท่านั้น) โดยการเปรียบเทียบกับค่าแรงดันที่ได้จากตัวต้านทานปรับได้ (VR) ที่ติดตั้งบนโมดูล โดยสัญญาณนี้มักจะนำมาใช้เมื่อผู้ใช้สนใจเพียงว่าระดับค่าความเข้มข้นแก๊สมากเกินกว่าค่าหนึ่งเท่านั้น



รูปที่ 3 แผนผังวงจรภายในโมดูลแก๊สเซนเซอร์

SI2 การทดลองอ่านค่าจากโมดูลแก๊สเซนเซอร์

วัตถุประสงค์

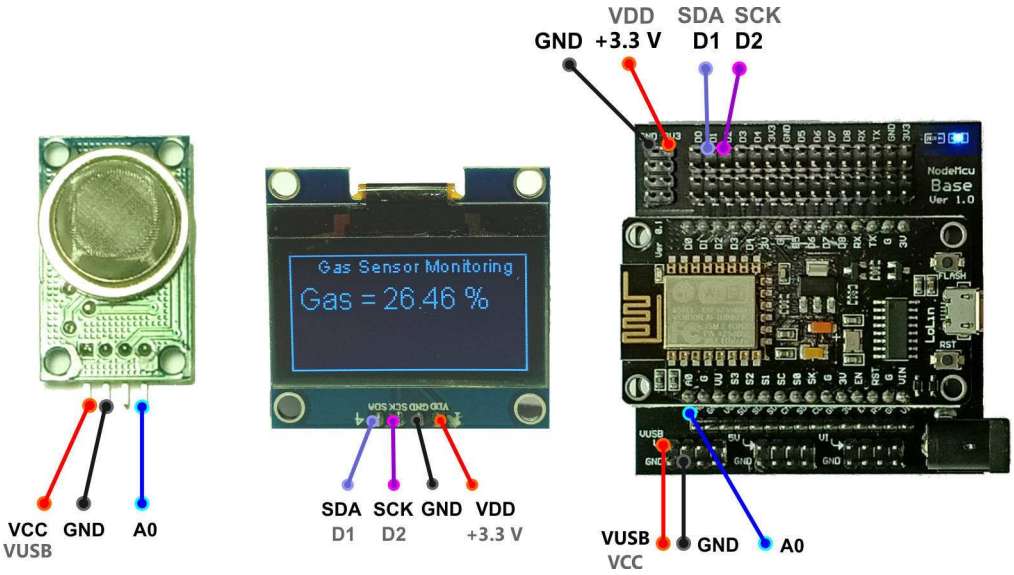
1. สามารถต่อบอร์ด NodeMCU v.3 กับบอร์ดโมดูลแก๊สเซนเซอร์ได้
2. สามารถเขียนโปรแกรมอ่านค่าปริมาณของแก๊สที่ตรวจวัดได้

อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดโมดูลแก๊สเซนเซอร์ MQ-135	1 บอร์ด
5.	บอร์ดโมดูล OLED Display ขนาด 128×64 พิกเซล	1 บอร์ด
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

วิธีการทดลอง

1. ต่อวงจรตามรูปที่ 4 ให้สัญญาณเอาต์พุตของโมดูลแก๊สเซนเซอร์ แบบแอนะล็อกต่อเข้าที่ A0 ของ NodeMCU และ ต่อ OLED Display แสดงผลเป็นค่าเปอร์เซ็นต์ของก๊าซที่ตรวจวัด
2. เขียนโค้ดโปรแกรมดังแสดงในหน้าถัดไป แล้วอัปโหลดและสังเกตผลที่แสดงบน OLED ภาพบนจอ OLED ในรูปที่ 4 แสดงตัวอย่างผลการวัดที่อ่านได้
3. ขอให้ผู้ใช้ ทดลองปรับค่า VR ที่อยู่บนบอร์ดโมดูลดู เพื่อแสดงให้เห็นการติดและดับของแอลอีดีบนบอร์ด



รูปที่ 4 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดโมดูลแก๊สเซนเซอร์และจอแสดงผล OLED

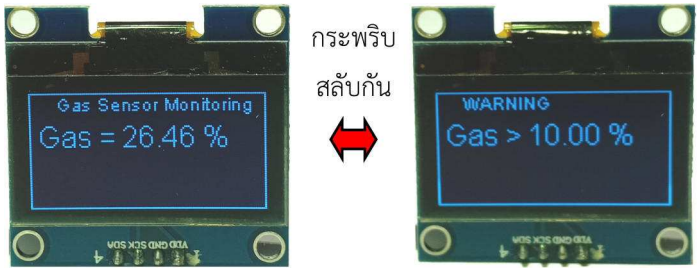
```
// Read Gas Sensor by NodeMCU ESP8266

#include <Wire.h>
#include <SH1106.h>

SH1106 display(0x3c, D1, D2);  // (Addr, SDA, SCL)

float gas;
int Pin = A0;

void setup() {
    display.init();
}

float read_gas() {
    float gas_sensor;
    gas_sensor = analogRead(Pin);
    gas_sensor = (100.*(gas_sensor/1024.));
    return (gas_sensor);
}

```

```
22 void show_gas() {
23     display.clear();
24     display.drawRect(0,0,128,64);
25     display.setFont(ArialMT_Plain_10);
26     display.drawString(15,0, "Gas Sensor
   Monitoring");
27     display.setFont(ArialMT_Plain_16);
28     display.drawString(5, 15, "Gas =     ");
29     display.drawString(50,15, String(gas) + " %");
30     display.display();
31 }
32
33 void loop() {
34     gas = read_gas();
35     show_gas();
36     delay(500);
37 }
38
```

แบบฝึกหัดท้ายการทดลอง

หากเราต้องการสร้างระบบแจ้งเตือนเมื่อเกิดแก๊สรั่ว เราอาจทำโดยการเปรียบเทียบค่าความเข้มข้นแก๊สที่วัดได้ (เก็บอยู่ในตัวแปร gas) กับค่าที่กำหนดในโปรแกรมที่อาจให้ชื่อเป็น gas_th (มาจาก threshold gas value) และเมื่อค่าที่วัดได้มีค่ามากเกินค่านั้น ก็ให้ส่งข้อความเตือนบนจอ OLED สลับกับการแสดงค่าแก๊สที่อ่านได้

จงพัฒนาโปรแกรมสำหรับสร้างระบบแจ้งเตือนข้างต้นต่อจากโปรแกรมในการทดลอง



รูปที่ 5 ตัวอย่างการแสดงผลการแจ้งเตือนในแบบฝึกหัดท้ายการทดลอง

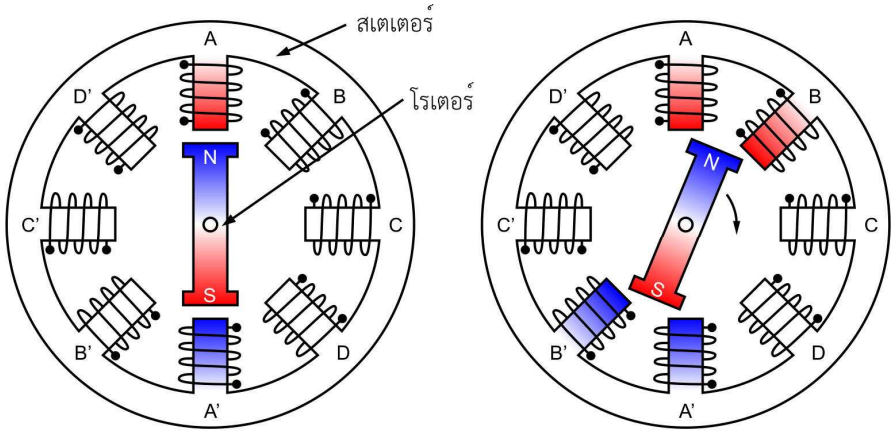
บทเสริมเรื่อง
การควบคุมสเต็ปปิ้งมอเตอร์

SI1 สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ (Stepping Motor) หรือ สเต็ปมอเตอร์ (Step Motor) หรือ สเต็ปเปอร์มอเตอร์ (Stepper Motor) คือมอเตอร์ไฟตรงประเภทหนึ่งที่การหมุนแบ่งออกเป็นสเต็ปหรือขั้นย่อย ๆ ที่เท่ากัน โดยเราสามารถควบคุมให้มอเตอร์นี้หมุนได้ตามจำนวนสเต็ปที่ต้องการและยังสามารถทำให้มอเตอร์คงตำแหน่งนั้นไว้ด้วยการจ่ายกระแสไฟฟ้า ในการควบคุมตำแหน่งของมอเตอร์ประเภทนี้ มีข้อดีคือ เราไม่จำเป็นต้องใช้เซนเซอร์ตรวจจับตำแหน่งสำหรับการป้อนกลับ แต่มอเตอร์ประเภทนี้มีข้อเสียเมื่อเทียบกับมอเตอร์ไฟตรงคือ มีแรงบิดค่อนข้างต่ำ และกินกระแสสูงแม้ในขณะที่มอเตอร์ไม่หมุน

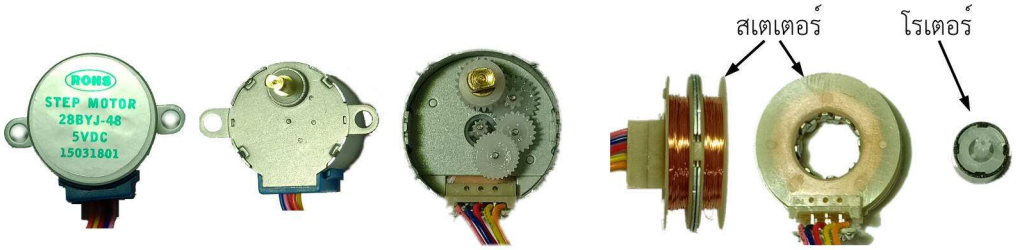
สำหรับเอกสารนี้เราจะกล่าวถึงสเต็ปปิ้งมอเตอร์ทั่วไปชนิดสองหรือสี่เฟสเท่านั้น สำหรับสเต็ปปิ้งมอเตอร์ชนิดหลายเฟสจะมีข้อดีคือมีระดับการสั่นต่ำ ประสิทธิภาพดีกว่า แต่ใช้งานได้ยากและมีราคาแพง

หลักการทำงานของสเต็ปปิ้งมอเตอร์คือ เราจะจ่ายไฟให้แก่ขดลวดที่ติดตั้งอยู่บนสเตเตอร์ (stator) เพื่อให้ส่งสนามแม่เหล็กออกมา และส่วนที่เป็นตัวหมุนหรือโรเตอร์ (rotor) จะมีแกนเป็นแม่เหล็กถาวร การจ่ายกระแสเข้าไปที่ขดลวดทำให้เกิดแรงดึงดูดจากสนามแม่เหล็ก ทำให้โรเตอร์หมุน โดยเมื่อเราทำการจ่ายกระแสให้กับขดลวดแต่ละขดอย่างต่อเนื่องจะทำให้มอเตอร์หมุนอย่างต่อเนื่อง และหากเราต้องการให้มอเตอร์หมุนไปตามจำนวนสเต็ปที่กำหนดก็สามารถทำได้โดยการจ่ายพัลส์กระแสผ่านขดลวดตามจำนวนสเต็ปที่ต้องการ รูปที่ 1 (ก) แสดงการทำงานของสเต็ปปิ้งมอเตอร์ โดยเมื่อจ่ายไฟให้แก่ขดลวดชุด A-A’ ก็จะทำให้แกนหมุนถูกล็อคอยู่ที่ตำแหน่งนั้น ไม่เคลื่อนที่ไปไหนแม้มีแรงกระทำจากภายนอก ในรูปที่ 1 (ข) คือเมื่อมีการป้อนกระแสไปยังขดลวดชุด B-B’ ด้วยก็จะทำให้แกนหมุนหมุนไปตามทิศทางของสนามแม่เหล็กที่เหนี่ยวนำ โดยการหมุนในทิศตามเข็มนาฬิกาจะเกิดขึ้นเมื่อมีการป้อนกระแสให้เกิดสนามแม่เหล็กหมุนวนตามลำดับ A-B-C-D และหากต้องการให้หมุนในทิศทวนเข็มนาฬิกา ก็สามารถทำได้เช่นกัน โดยการกลับลำดับการป้อนกระแสให้แก่ขดลวดบนมอเตอร์



รูปที่ 1 การทำงานของสเต็ปปิ้งมอเตอร์ (ก) ขณะล็อคให้หยุดนิ่ง และ (ข) ขณะกำลังหมุนไปครึ่งสเต็ป

รูปที่ 2 แสดงตัวอย่างภาพถ่ายลักษณะภายนอกและภายในของสเต็ปปิ้งมอเตอร์เบอร์ 28BYJ-48 ซึ่งเป็นสเต็ปปิ้งมอเตอร์ไฟตรงขนาดเล็ก (5 V) ที่มีใช้กันอย่างแพร่หลาย โดยภายในตัวมอเตอร์จะมีเฟืองทดอยู่ด้วย (รูปที่ 2 (ข)) ทำให้สามารถส่งแรงบิดได้มากขึ้น โดยมอเตอร์ชนิดนี้เป็นแบบสี่เฟสแบบขั้วเดียว (unipolar) ที่มีการเชื่อมต่อหรือแท็ป (tap) ตรงกลางของขดลวดแต่ละชุด ทำให้มอเตอร์นี้มีสี่เฟส และทำให้เราสามารถควบคุมมอเตอร์ได้อย่างง่ายดายยิ่งขึ้นและมีโอกาสที่จะต่อสายไฟผิดพลาดได้น้อยลง

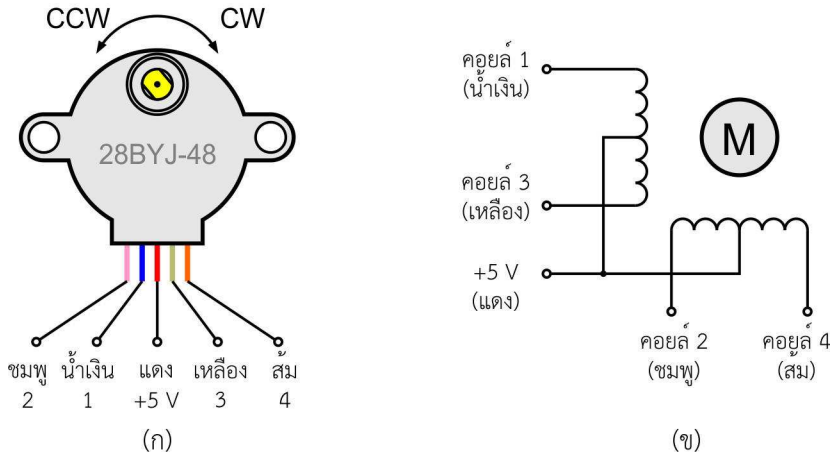


รูปที่ 2 ภาพถ่าย (ก) ลักษณะภายนอก (ข) ภายในส่วนของเฟืองทด และ (ค) ภายในส่วนของสเตเตอร์และโรเตอร์ของสเต็ปปิ้งมอเตอร์ 28BYJ-48

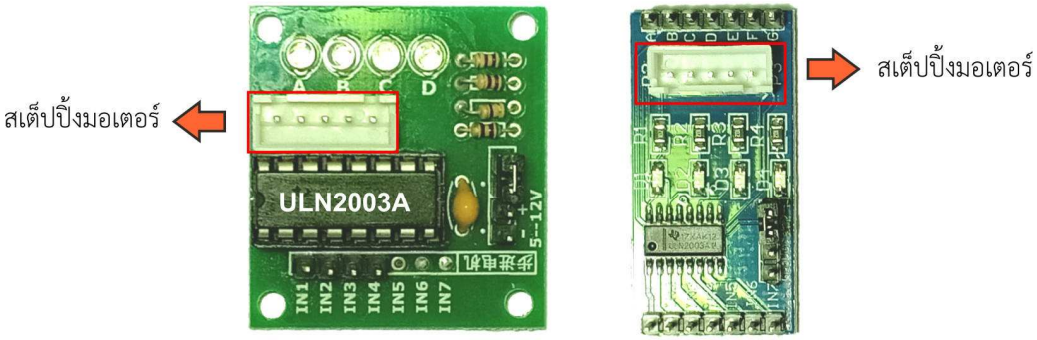
SI2 การควบคุมสเต็ปปิ้งมอเตอร์

ในการใช้งานสเต็ปปิ้งมอเตอร์ เบื้องต้นเราจะต้องทราบลักษณะการพันขดลวดของมอเตอร์ที่เลือกใช้ ซึ่งเราสามารถทราบได้จากแผ่นข้อมูลของมอเตอร์นั้น ๆ โดยส่วนมากสีของสายไฟของสเต็ปปิ้งมอเตอร์จะสัมพันธ์กับการพันขดลวดตามที่ผู้ผลิตกำหนด รูปที่ 3 แสดงตัวอย่าง แสดงลักษณะการวางสายและสีของขดลวดภายในสเต็ปปิ้งมอเตอร์ 28BYJ-48 และสำหรับการหมุนของมอเตอร์นี้ เราจะกำหนดให้หมุนได้ทั้งในทิศตามเข็มนาฬิกา (clockwise, CW) และทวนเข็มนาฬิกา (counter clockwise, CCW) ขึ้นกับลำดับการส่งกระแสเข้าสู่ขดลวด

สำหรับการควบคุมสเต็ปปิ้งมอเตอร์นั้น เราจะทำได้ผ่านการสั่งการด้วยไมโครคอนโทรลเลอร์ โดยเนื่องจากไมโครคอนโทรลเลอร์ไม่สามารถให้ขับกระแสขดลวดภายในมอเตอร์ได้โดยตรง เราจึงจำเป็นต้องมีบอร์ดขับ (driver) ที่ทำหน้าที่จ่ายกระแส โดยไอซีเบอร์ที่นิยมนำมาใช้ควบคุมสเต็ปปิ้งมอเตอร์ขนาดเล็ก คือ ไอซีเบอร์ ULN2003A รูปที่ 4 แสดงตัวอย่างภาพถ่ายบอร์ดขับที่ใช้ไอซีเบอร์นี้โดยมีตัวถังแบบ DIP (dual inline package) (รูปที่ 4(ก)) และแบบ SMD (surface mount device) (รูปที่ 4(ข)) โดยบอร์ดขับนี้ จ่ายกระแสสูงสุดได้ 0.5 A และสามารถรองรับแรงดันได้ในช่วง 5 - 12 V และวงจรบอร์ดขับที่แสดงนี้ ออกแบบสำหรับสเต็ปปิ้งมอเตอร์ 4 เฟส 5 สาย

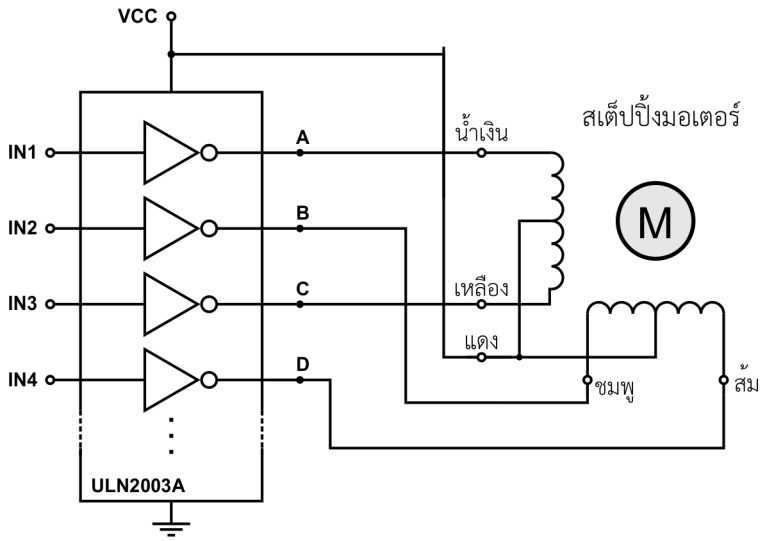


รูปที่ 3 (ก) ภาพลักษณะการวางสาย สี และ (ข) ภาพลักษณะการพันขดลวดของสเต็ปปิ้งมอเตอร์ 28BYJ-48 ชนิดสี่เฟสแบบขั้วเดียวที่มีแท็ปตรงกลางของขดลวดและเชื่อมต่อกัน



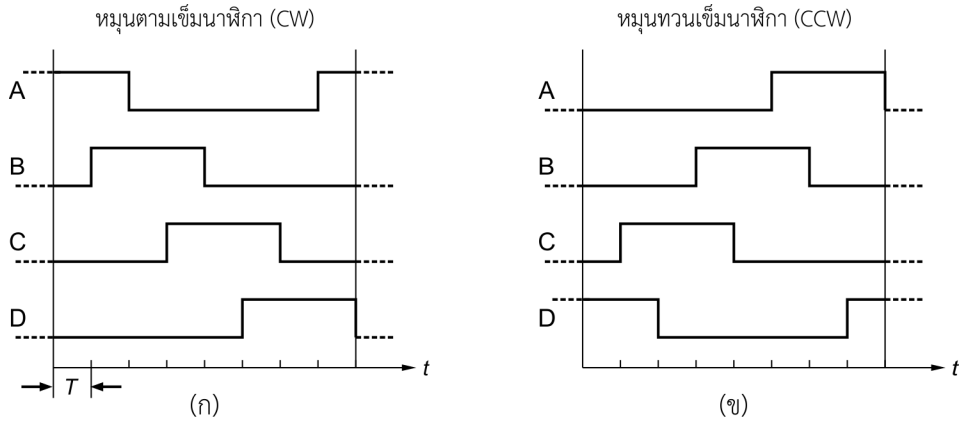
รูปที่ 4 บอร์ดขับสเต็ปปิ้งมอเตอร์ที่ใช้ไอซีเบอร์ ULN2003A แบบ (ก) DIP และ (ข) SMD

การเชื่อมต่อของสเต็ปปิ้งมอเตอร์ที่ใช้ไอซีเบอร์ ULN2003A และสเต็ปปิ้งมอเตอร์เบอร์ 28BYJ-48 ทำได้โดยง่ายและมีลักษณะวงจรดังรูปที่ 5 โดยเนื่องจากลักษณะของสัญญาณขาออกจากไอซีเป็นแบบกลับขั้ว ดังนั้นจึงมีการจ่ายไฟ VCC ให้แก่ขดลวดของมอเตอร์นี้เพื่อที่จะทำให้ เมื่อวงจรได้รับสัญญาณขาเข้าที่เป็นลอจิก HIGH แล้ว จะเกิดกระแสไหลผ่านขดลวดภายในมอเตอร์ (จาก VCC ไปยังกราวด์ ผ่านไอซี) และมอเตอร์จะทำงาน



รูปที่ 5 ภาพลักษณะวงจรขับสเต็ปปิ้งมอเตอร์โดยไอซีเบอร์ ULN2003A และสเต็ปปิ้งมอเตอร์เบอร์ 28BYJ-48

สำหรับสัญญาณที่ใช้ในการควบคุมการหมุนของมอเตอร์ 4 เฟส (ให้ชื่อเป็น A, B, C, และ D) นั้น จะมีลักษณะดังรูปที่ 6 นั่นคือ เราจะสั่งการให้จ่ายกระแสไปยังขดลวดทีละขดตามลำดับ เพื่อให้มอเตอร์หมุนตามที่ต้องการ โดยสำหรับรูปที่ 6(ก) คือการสั่งการให้มอเตอร์หมุนตามเข็มนาฬิกา โดยจะมีการกลับโลจิกจาก HIGH เป็น LOW ที่ไอซี ULN2003A เพื่อให้กระแสไหลในขดลวดที่ต่ออยู่ และรูปที่ 6(ข) คือการสั่งการให้มอเตอร์หมุนทวนเข็มนาฬิกา ซึ่งเป็นเพียงการกลับลำดับการสั่งการ สำหรับการสั่งการให้มอเตอร์หมุนเร็วหรือช้านั้น เราสามารถทำได้โดยการควบคุมความกว้างของพัลส์สัญญาณที่ป้อนให้แก่บอร์ดขับ (ค่า *T* ในรูปที่ 6(ก)) โดยค่านี้จะต้องมีค่าไม่น้อยเกินไป เพราะมอเตอร์ซึ่งเป็นอุปกรณ์ทางกลอาจตอบสนองไม่ทันกับสัญญาณไฟฟ้าที่ป้อนให้แก่ขดลวด สำหรับความเร็วในการตอบสนองสูงสุดของสเต็ปปิ้งมอเตอร์ทั่วไปนั้นจะอยู่ในระดับ 1-10 มิลลิวินาที ต่อการป้อนกระแสแต่ละเฟส ซึ่งทำให้เราจะต้องใช้คำสั่งหน่วงเวลาในโปรแกรมเพื่อให้สามารถควบคุมมอเตอร์ได้ตามที่ต้องการ



รูปที่ 6 ลักษณะสัญญาณที่ต้องป้อนให้แก่สเต็ปปิ้งมอเตอร์
เพื่อให้มอเตอร์หมุน (ก) ตามเข็มนาฬิกา และ (ข) ทวนเข็มนาฬิกา

SI3 การทดลองควบคุมสเต็ปปิ้งมอเตอร์

วัตถุประสงค์

1. สามารถต่อบอร์ด Node MCU v.3 กับบอร์ดขับสเต็ปปิ้งมอเตอร์และสเต็ปปิ้งมอเตอร์ได้
2. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมทิศทางการหมุนของสเต็ปปิ้งมอเตอร์ได้
3. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ผ่านพอร์ตอนุกรมได้

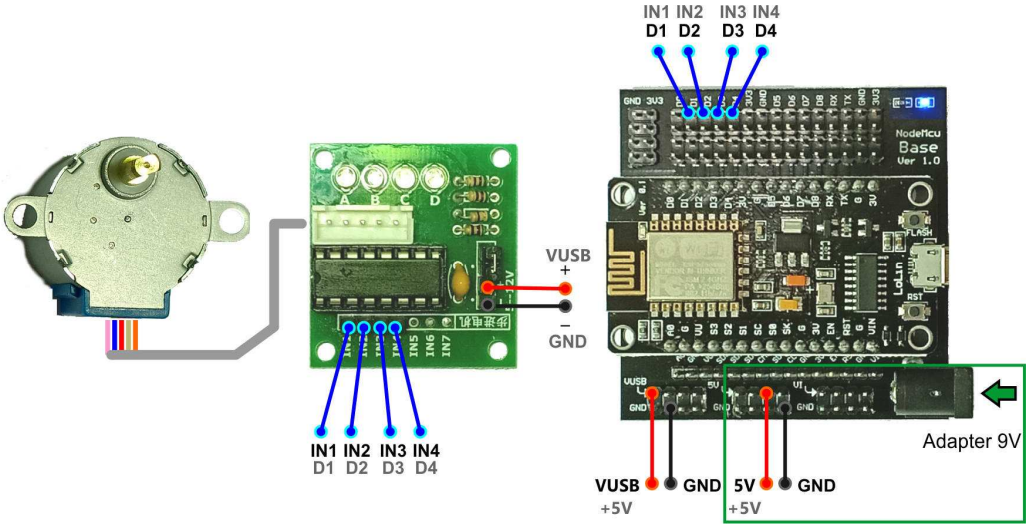
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	บอร์ดขับมอเตอร์ที่ใช้ไอซีเบอร์ ULN2003A	1 บอร์ด
5.	สเต็ปปิ้งมอเตอร์ 28BYJ-48 พร้อมสาย	1 ตัว
6.	อะแด็ปเตอร์ 9 V	1 ตัว
7.	สาย USB	1 เส้น
8.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	6 เส้น

วิธีการทดลอง

ตอนที่ 1 การควบคุมทิศทางการหมุนของมอเตอร์

1. ต่อวงจรดังรูปที่ 7
2. เขียนโปรแกรมดังที่แสดงในโค้ดหน้าถัดไป จากนั้นจึงอัปโหลดลง NodeMCU v.3 แล้วสังเกตทิศทางการหมุนของสเต็ปปิ้งมอเตอร์ โดยหากมอเตอร์ไม่หมุน เนื่องจากไฟเลี้ยงจาก USB ไม่เพียงพอ ก็ขอให้ใช้อะแด็ปเตอร์ร่วมในการจ่ายไฟไปยังบอร์ดขับมอเตอร์



รูปที่ 7 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดขับสเต็ปปิ้งมอเตอร์และสเต็ปปิ้งมอเตอร์ โดยในกรอบสี่เหลี่ยมสีเขียวคือการต่อสายในกรณีที่ไฟเลี้ยงจากพอร์ต USB นั้นไม่เพียงพอ

```
// Control 4-Phase Step Motor with ULN2003A
//
// 5-V Step Motor is 28BYJ48
// Pin 1-4 = (Blue, Pink, Yellow, and Orange)
// Pin 5 = VCC (Red)

int motorPin1 = D1;
int motorPin2 = D2;
int motorPin3 = D3;
int motorPin4 = D4;

int motorDelay = 5;  //variable to set motor speed
//Long delay = slow speed (8*motorDelay per 1 rev)

void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
}

```

```
22 void loop() {
23
24     // Rotate CW
25     for(int i=0; i < 100; i++)    {
26         motorCW();     // Rotate 1 rev. CW
27     }
28     delay(1000);
29
30     // Rotate CCW
31     for(int i=0; i < 100; i++)    {
32         motorCCW();     // Rotate 1 rev. CCW
33     }
34     delay(1000);
35 }
36
37 void motorCW() {
37     digitalWrite(motorPin1, HIGH);
38     digitalWrite(motorPin2, LOW);
39     digitalWrite(motorPin3, LOW);
40     digitalWrite(motorPin4, LOW);
41     delay(motorDelay);
42     digitalWrite(motorPin1, HIGH);
43     digitalWrite(motorPin2, HIGH);
44     digitalWrite(motorPin3, LOW);
45     digitalWrite(motorPin4, LOW);
46     delay(motorDelay);
47     digitalWrite(motorPin1, LOW);
48     digitalWrite(motorPin2, HIGH);
49     digitalWrite(motorPin3, LOW);
50     digitalWrite(motorPin4, LOW);
51     delay(motorDelay);
52     digitalWrite(motorPin1, LOW);
53     digitalWrite(motorPin2, HIGH);
54     digitalWrite(motorPin3, HIGH);
55     digitalWrite(motorPin4, LOW);
56     delay(motorDelay);
57     digitalWrite(motorPin1, LOW);
58     digitalWrite(motorPin2, LOW);
59     digitalWrite(motorPin3, HIGH);
60     digitalWrite(motorPin4, LOW);
61     delay(motorDelay);
62     digitalWrite(motorPin1, LOW);
63     digitalWrite(motorPin2, LOW);
64     digitalWrite(motorPin3, HIGH);
```

```
    digitalWrite(motorPin4, HIGH);
    delay(motorDelay);
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, HIGH);
    delay(motorDelay);
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin4, HIGH);
    delay(motorDelay);
}

void motorCCW() {
    digitalWrite(motorPin4, HIGH);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin1, LOW);
    delay(motorDelay);
    digitalWrite(motorPin4, HIGH);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin1, LOW);
    delay(motorDelay);
    digitalWrite(motorPin4, LOW);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin2, LOW);
    digitalWrite(motorPin1, LOW);
    delay(motorDelay);
    digitalWrite(motorPin4, LOW);
    digitalWrite(motorPin3, HIGH);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin1, LOW);
    delay(motorDelay);
    digitalWrite(motorPin4, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin1, LOW);
    delay(motorDelay);
    digitalWrite(motorPin4, LOW);
    digitalWrite(motorPin3, LOW);
    digitalWrite(motorPin2, HIGH);
    digitalWrite(motorPin1, HIGH);
```

```
109     delay(motorDelay);
110     digitalWrite(motorPin4, LOW);
111     digitalWrite(motorPin3, LOW);
112     digitalWrite(motorPin2, LOW);
113     digitalWrite(motorPin1, HIGH);
114     delay(motorDelay);
115     digitalWrite(motorPin4, HIGH);
116     digitalWrite(motorPin3, LOW);
117     digitalWrite(motorPin2, LOW);
118     digitalWrite(motorPin1, HIGH);
119     delay(motorDelay);
120 }
121
```

3. ตรวจสอบการหมุนของมอเตอร์ จากนั้นทดลองแก้ไขค่า `motorDelay` จาก 5 เป็น 10 จากนั้นจึงอัปโหลดโปรแกรมใหม่ แล้วสังเกตการหมุนของมอเตอร์

ตอนที่ 2 การควบคุมการหมุนของมอเตอร์ผ่านพอร์ตอนุกรม

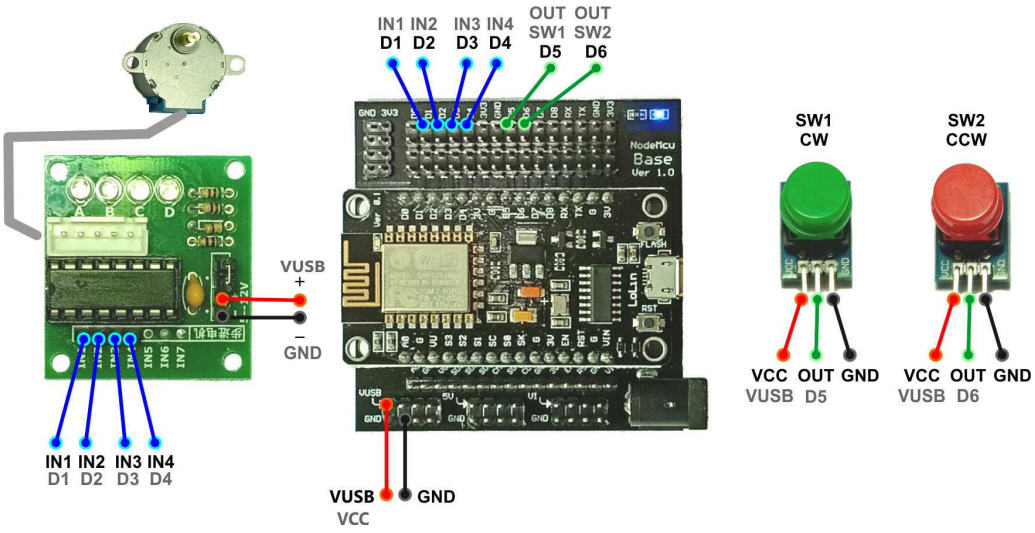
1. ต่อวงจรดังรูปที่ 7 (เหมือนการทดลองตอนที่ 1)
2. เขียนโปรแกรมโดยมีโค้ดดังที่แสดงในกรอบด้านล่าง โดยนำฟังก์ชัน `motorCW()` และ `motorCCW()` จากการทดลองตอนที่ 1 มาใส่ลงในโปรแกรมนี้ด้วย จากนั้นจึงอัปโหลดลง NodeMCU v.3
3. เปิด Serial Monitor (Ctrl+Shift+M) และป้อนคำสั่ง ‘1’ หรือ ‘2’ แล้วสังเกตการหมุนของมอเตอร์

```
 1 // Control 4-Phase Step Motor with ULN2003A
 2 // by Serial Data
 3 //
 4 // 5-V Step Motor is 28BYJ48
 5 // Pin 1-4 = (Blue, Pink, Yellow, and Orange)
 6 // Pin 5 = VCC (Red)
 7
 8 int motorPin1 = D1;
 9 int motorPin2 = D2;
10 int motorPin3 = D3;
11 int motorPin4 = D4;
12
```

```
13 int motorDelay = 5;  //variable to set motor speed
14 // Long delay = slow speed(8*motorDelay per 1 rev)
15
16 char inchar; // Input control character
17
18 void setup() {
19   Serial.begin(9600);
20   pinMode(motorPin1, OUTPUT);
21   pinMode(motorPin2, OUTPUT);
22   pinMode(motorPin3, OUTPUT);
23   pinMode(motorPin4, OUTPUT);
24 }
25
26 void loop() {
27
28     inchar = Serial.read();
29
30     if(inchar == '1') {
31     for(int i=0; i < 20; i++)    {
32         motorCW();
33     }
34     delay(100);
35     }
36
37     if(inchar == '2') {
38     for(int i=0; i < 20; i++)    {
39         motorCCW();
40     }
41     delay(100);
42     }
43 }
44
```

แบบฝึกหัดท้ายการทดลอง

จงต่อวงจรดังรูปที่ 8 และเขียนโค้ดโปรแกรมที่ทำให้ ผู้ใช้สามารถควบคุมสเต็ปปิ้งมอเตอร์ให้หมุนตามทิศที่กำหนดได้เมื่อมีการกดปุ่มกด

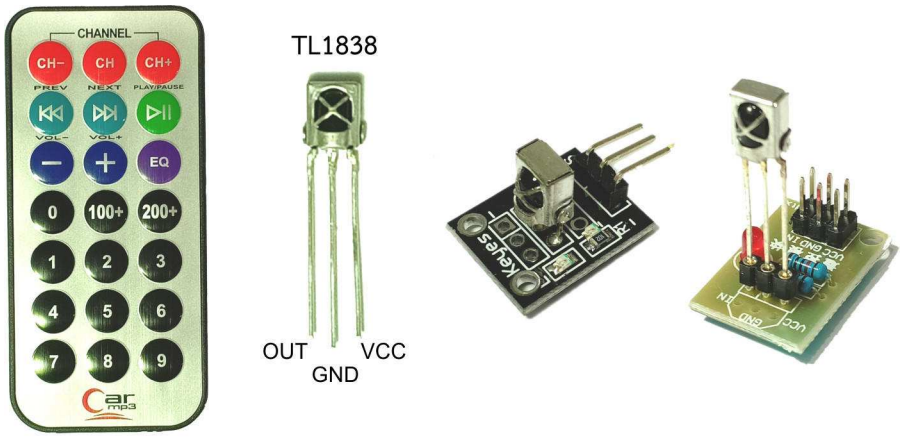


รูปที่ 8 การเชื่อมต่อ NodeMCU v.3 กับบอร์ดขับสเต็ปปิ้งมอเตอร์ สเต็ปปิ้งมอเตอร์ และปุ่มกดสองปุ่มเพื่อควบคุมการทำงานของมอเตอร์ด้วยปุ่มกด

บทเสริมเรื่อง
การใช้งานรีโมตอินฟราเรด

SI1 ชุดสื่อสารรีโมตอินฟราเรด

ในการสื่อสารแบบไร้สาย ตัวควบคุมแบบรีโมตอินฟราเรด (Infrared Remote Controller) ถูกนำมาใช้กันอย่างแพร่หลายในงานหลายประเภท โดยตัวอย่างที่เห็นได้ทั่วไปคือการในการควบคุมโทรทัศน์ และเรียกอุปกรณ์นี้สั้น ๆ ว่า “รีโมต” โดยรีโมตลักษณะนี้จะใช้คลื่นแสงอินฟราเรด (แสงที่ตามองไม่เห็น) ในการสื่อสาร ระบบการสื่อสารไร้สายด้วยรีโมตประกอบด้วย ตัวส่ง (transmitter) และตัวรับ (receiver) รูปที่ 1 แสดงภาพถ่ายของตัวส่งหรือตัวรีโมตและตัวรับแบบต่าง ๆ ที่มักนำมาใช้ในการสื่อสารร่วมกับการควบคุมอัตโนมัติด้วยไมโครคอนโทรลเลอร์ โดยสำหรับตัวส่งมักจะมีลักษณะเป็นอาเรย์ของปุ่ม (รูปที่ 1(ก)) และตัวรับที่นิยมใช้นั้นคือ ไอซีเบอร์ TL1838 ดังรูปที่ 1(ข) ในท้องตลาดจะมีจำหน่ายทั้งแบบที่เป็นตัวอุปกรณ์เดี่ยว ๆ (รูปที่ 1 (ข)) และแบบที่มีการนำมาติดตั้งบนแผ่นวงจรเพื่อให้การใช้งานทำได้อย่างง่ายดาย (รูปที่ 1(ค) และ (ง)) ซึ่งสำหรับตัวรับนี้จะประกอบด้วยโฟโต้ทรานซิสเตอร์และส่วนของวงจรถอดรหัสสัญญาณที่ได้รับให้เป็นข้อมูลดิจิทัล



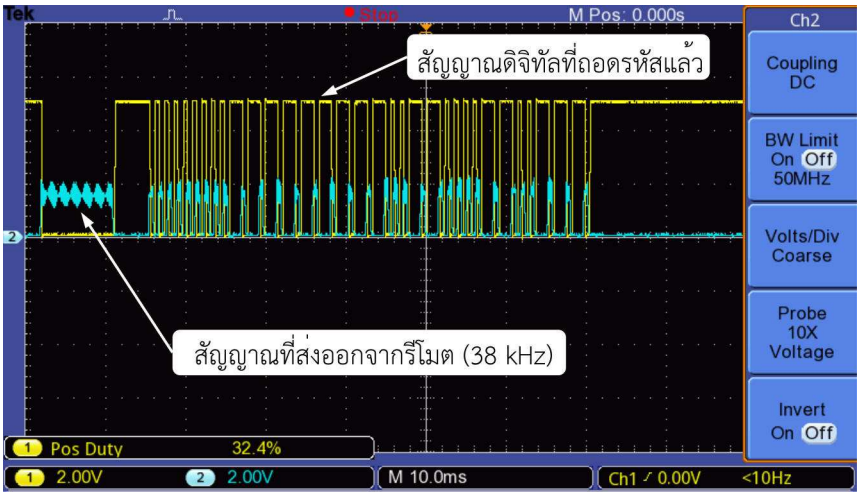
(ก) (ข) (ค) (ง)

รูปที่ 1 ภาพถ่ายโมดูล (ก) ตัวส่งและ (ข) – (ง) ตัวรับในชุดสื่อสารรีโมตอินฟราเรด

FFA25D	FF629D	FFE21D
FF22DD	FF02FD	FFC23D
FFE01F	FFA857	FF906F
FF6897	FF9867	FFB04F
FF30CF	FF18E7	FF7A85
FF10EF	FF38C7	FF5AA5
FF42BD	FF4AB5	FF52AD

รูปที่ 2 รหัสข้อมูลดิจิทัลเลขฐานสิบหกที่ส่งเมื่อมีการกดปุ่มต่าง ๆ บนโมดูลตัวส่งของรีโมต ชนิด 7 แถว 3 หลัก ที่แสดงในรูปที่ 1(ก)

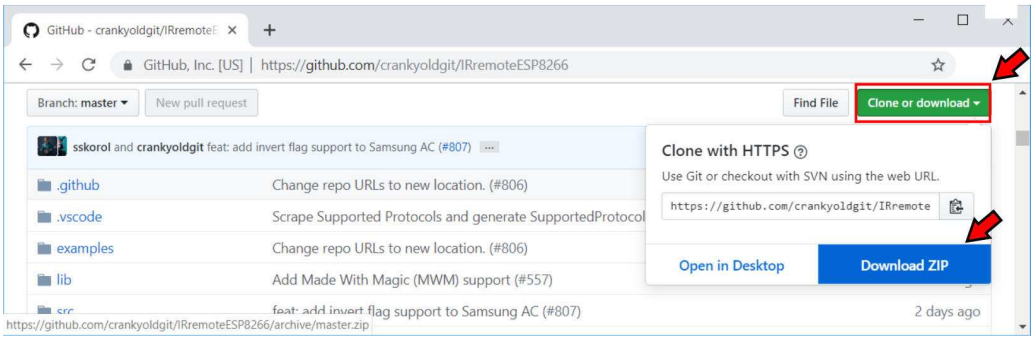
ข้อมูลดิจิทัลที่ส่งจากโมดูลตัวส่งนั้น จะมีลักษณะที่ขึ้นกับปุ่มที่กด โดยสามารถเขียนเป็นเลขฐานสิบหกได้ดังแสดงในรูปที่ 2 โดยผู้ใช้งานจะต้องทราบรหัสเหล่านี้เพื่อนำไปใช้ในการเขียนโปรแกรมรับและอ่านข้อมูล ในการทำงานจริงนั้น โมดูลตัวส่งจะทำการมอดูเลตหรือแปลงรหัสข้อมูลดิจิทัลให้สามารถส่งได้ตามมาตรฐานการสื่อสาร โดยใช้คลื่นความถี่ 38 kHz และตัวรับข้อมูลนั้นจะมีการถอดรหัสดิจิทัลออกมา รูปที่ 3 แสดงลักษณะสัญญาณที่ส่งออกจากรีโมต (วัดโดยตัวตรวจวัดแสงทั่วไป) และสัญญาณดิจิทัลที่ถอดรหัสแล้ว ด้วยไอซี TL1838



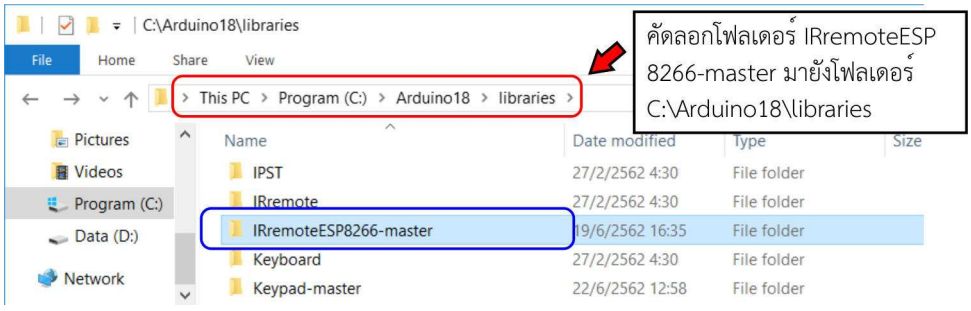
รูปที่ 3 ภาพลักษณะสัญญาณที่ส่งออกจากโมดูลตัวส่งและสัญญาณดิจิทัลที่ถอดรหัสแล้ว

SI2 ไลบรารี IRremoteESP8266

แม้ว่าสัญญาณจากตัวรับสัญญาณรีโมตจะมีการแปลงเป็นสัญญาณดิจิทัลแล้ว เราก็ยังจำเป็นต้องมีการอ่านสัญญาณดิจิทัลนี้อย่างถูกต้องเพื่อให้ได้รับข้อมูลการกดปุ่มจากตัวส่งที่ถูกต้อง การเขียนโปรแกรมเพื่ออ่านข้อมูลแต่ละบิตเองนั้นเป็นเรื่องยุ่งยาก ดังนั้นในการเขียนโปรแกรมด้วย Arduino IDE เพื่อควบคุมไมโครคอนโทรลเลอร์ที่ใช้ชิป ESP8266 ให้อ่านค่าจากสัญญาณจากรีโมตนั้น เรามักจะใช้ไลบรารีสำเร็จรูป ซึ่งเราจะต้องมีการเพิ่มไลบรารีนี้ที่มีชื่อว่า IRremoteESP8266 เข้าไป เสียก่อน โดยเราทำได้โดยการไปดาวน์โหลดมาติดตั้งในเครื่อง (เช่นเดียวกับไลบรารีอื่น ๆ ที่กล่าวถึงในบทก่อนหน้านี้ เช่น OLED) โดยผู้ใช้สามารถดาวน์โหลดไลบรารีสำหรับการสื่อสารด้วยรีโมตนี้ได้ที่ https://github.com/crankyoldgit/IRremoteESP8266 รูปที่ 4 การดาวน์โหลดและรูปที่ 5 แสดงการติดตั้งไลบรารี IRremoteESP8266 ลงในโปรแกรม Arduino IDE



รูปที่ 4 การดาวน์โหลดไลบรารีของ IRremoteESP8266 จาก github



รูปที่ 5 การติดตั้งไลบรารี IRremoteESP8266 ลงใน Arduino IDE

คำสั่งหรือฟังก์ชันที่ใช้ในการอ่านค่าที่ได้รับจากโมดูลตัวรับสัญญาณรีโมตจะเก็บอยู่ในไฟล์ไลบรารีชื่อ IRrecv.h โดย เราจะต้องเพิ่มไฟล์ไลบรารีฐาน 2 ไฟล์ ที่บรรจุคำสั่งที่จะถูกเรียกใช้ด้วย นั่นคือไฟล์ไลบรารี Arduino.h และ IRremoteESP8266.h นอกจากนี้ ในการแสดงผลเลขฐานสิบหก (ในรูปที่ 2) ผ่านพอร์ตอนุกรม เราจะต้องใช้คำสั่งหรือฟังก์ชันแสดงผลที่มีอยู่ไฟล์ไลบรารี IRutils.h ด้วย ดังนั้นในการอ่านค่าจากตัวรับสัญญาณรีโมตนี้ เราจะต้องเพิ่มไฟล์ไลบรารีไว้ที่ส่วนหัวของโปรแกรมคือ

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>    // for serialPrintUint64
```

สำหรับผู้ที่สนใจรายละเอียดคำสั่งต่าง ๆ ที่อยู่ในไลบรารีเหล่านี้ ก็สามารถเปิดไฟล์เหล่านั้นมาอ่านเพิ่มเติมได้

ในการใช้งานไลบรารีนี้ เราจะต้องกำหนด “วัตถุ” หรือออปเจค 2 ตัว จาก คือ ตัวรับรีโมตที่บ่งบอกพินที่เราเชื่อมต่อ ระหว่าง ตัวรับและ NodeMCU และ ตัวข้อมูลที่เราได้รับ โดยกำหนดจากคลาส IRrecv และ decode_results ตามลำดับ เช่น

```
int irPin = D1;
IRrecv irrecv(irPin);
decode_results results;
```

คำสั่งที่เกี่ยวข้องกับการใช้งานตัวรับสัญญาณรีโมต ที่อยู่ในไลบรารีที่กล่าวมาข้างต้นนี้คือ

- enableIRIn() เป็นเมดธอดที่เป็นการบอกให้โปรแกรมเริ่มต้นการทำงานของตัวรับสัญญาณ
- decode() เป็นเมดธอดที่ใช้ถอดรหัสข้อมูลที่ตัวรับสัญญาณได้รับ
- resume() เป็นเมดธอดที่สั่งให้ถอดรหัสข้อมูลที่ตัวรับสัญญาณได้รับต่อไป

และ

- serialPrintUint64() เป็นฟังก์ชันหรือคำสั่งให้พิมพ์ตัวเลขขนาดใหญ่ โดยมีตัวโค้ดเก็บอยู่ใน IRutils.cpp และเรียกใช้โดยเพิ่มไฟล์เฮดเดอร์ IRutils.h ลงในโปรแกรม

SI3 การทดลองใช้งานรีโมตอินฟราเรด

วัตถุประสงค์

1. สามารถต่อบอร์ด Node MCU v.3 กับโมดูลรับสัญญาณอินฟราเรดได้
2. สามารถเขียนโปรแกรมให้ NodeMCU อ่านค่าที่ได้รับจากโมดูลรับสัญญาณอินฟราเรดได้
3. สามารถเขียนโปรแกรมให้ NodeMCU ควบคุมรีเลย์จากคำสั่งที่ได้รับจากโมดูลรับสัญญาณอินฟราเรดได้

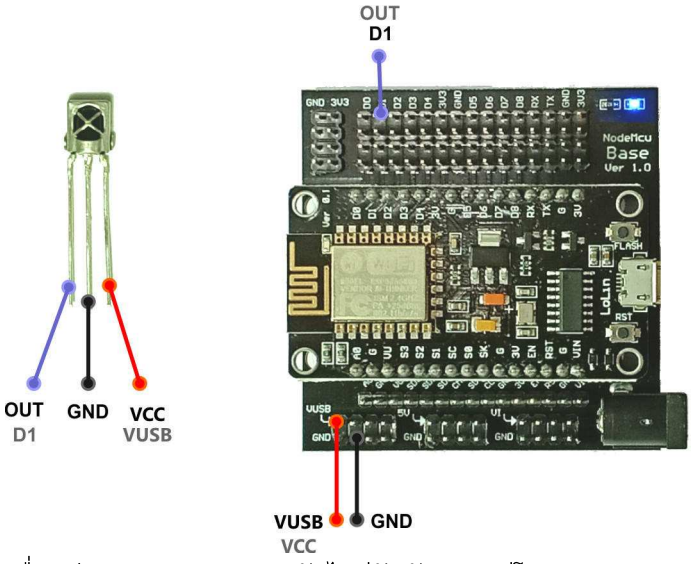
อุปกรณ์ที่ใช้ในการทดลอง

1.	เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows (ตั้งแต่ Windows 7 ขึ้นไป) พร้อมติดตั้งโปรแกรม Arduino IDE 1.8.8 IoT	1 เครื่อง
2.	NodeMCU v.3	1 บอร์ด
3.	NodeMCU Base Ver 1.0	1 บอร์ด
4.	ชุดรับส่งรีโมตอินฟราเรด ประกอบด้วยตัวส่งและตัวรับ	1 ชุด
5.	บอร์ดรีเลย์ชนิด 4 ช่อง	1 บอร์ด
6.	สาย USB	1 เส้น
7.	สายต่อวงจร (สายจัมพ์ เมีย-เมีย)	9 เส้น

วิธีการทดลอง

ตอนที่ 1 การอ่านค่าเลขฐานสิบหกจากตัวรับสัญญาณรีโมต

1. ต่อวงจรดังรูปที่ 6
2. เขียนโปรแกรมดังที่แสดงในโค้ดหน้าถัดไป จากนั้นจึงอัปโหลดลง NodeMCU v.3
3. เปิด Serial Monitor (ในเมนู Tools) จากนั้นทดสอบกดปุ่มบนโมดูลตัวส่งสัญญาณรีโมตและสังเกตผลลัพธ์ที่ได้ (เทียบกับ ข้อมูลในรูปที่ 2)



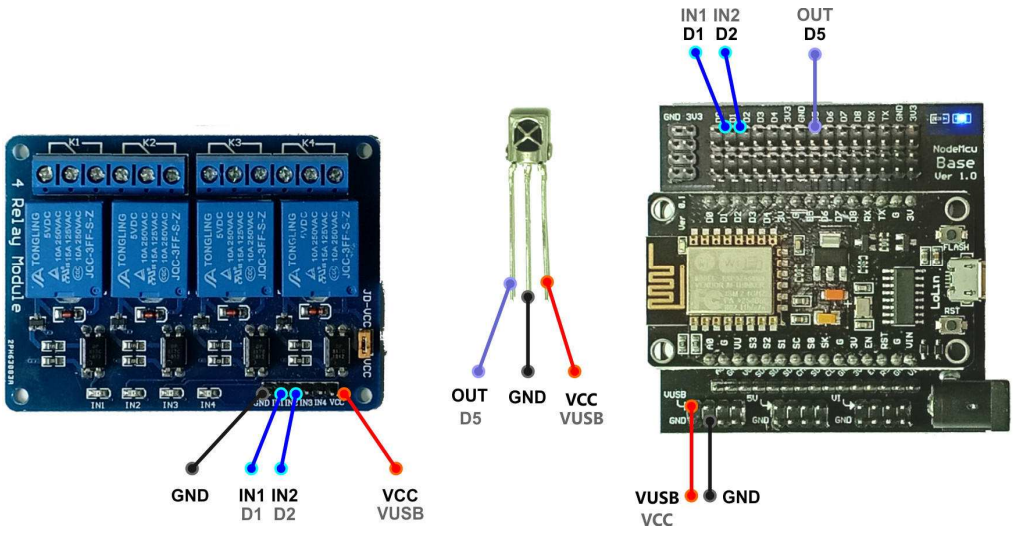
รูปที่ 6 การเชื่อมต่อ NodeMCU v.3 กับไอซีรับสัญญาณรีโมตอินฟราเรดเบอร์ TL1838

```
// Read IR Remote Code by NodeMCU ESP8266
// using VS1838B IR Receiver

#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>    // for serialPrintUint64

int irPin = D1;   // Pin D1 = GPIO 5, D5 = GPIO 14

IRrecv irrecv(irPin);

decode_results results;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();  // Start the receiver
  while (!Serial)    // Wait for the serial comm.
    delay(50);
  Serial.println();
  Serial.print("IRrecv Test is now running on Pin
");
  Serial.println(irPin);
}
```

```
24
25 void loop() {
26   if (irrecv.decode(&results)) {
27     // print() & println() cannot handle printing
   long longs.
28     serialPrintUint64(results.value, HEX);
29     Serial.println("");
30     irrecv.resume();  // Receive the next value
31   }
32   delay(100);
33 }
34
```

ตอนที่ 2 การควบคุมรีเลย์ด้วยรีโมตอินฟราเรด

1. ต่อวงจรดังรูปที่ 7
2. เขียนโปรแกรมดังที่แสดงในโค้ดหน้าถัดไป จากนั้นจึงอัปโหลดลง NodeMCU v.3
3. ทดสอบกดปุ่มบนโมดูลตัวส่งสัญญาณรีโมต ที่ปุ่มเลข ‘0’, ‘1’, ‘2’ หรือ ‘3’ ดู สังเกตการทำงานของรีเลย์



รูปที่ 7 การเชื่อมต่อ NodeMCU v.3 กับไอซีรับสัญญาณรีโมตอินฟราเรดและบอร์ดรีเลย์

```
// Remote Control Relay by NodeMCU ESP8266
// using VS1838B IR Receiver

#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
// #include <IRutils.h>  // for serialPrintUint64

#define ON LOW
#define OFF HIGH
int Relay1 = D1;
int Relay2 = D2;

int irPin = D5; // Pin D1 = GPIO 5, D5 = GPIO 14

IRrecv irrecv(irPin);

decode_results results;

/* Define function of the buttons
'0' = Switch Relay1 OFF
'1' = Switch Relay1 ON
'2' = Switch Relay2 OFF
'3' = Switch Relay2 ON
*/

const uint64 b0 = 0xFF6897;   // 0
const uint64 b1 = 0xFF30CF;   // 1
const uint64 b2 = 0xFF18E7;   // 2
const uint64 b3 = 0xFF7A85;   // 3

void setup() {
  // Serial.begin(9600);       // For debugging
  /  /     Serial.println("Remote   Control   Relay
Module");
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  digitalWrite(Relay1, OFF);
  digitalWrite(Relay2, OFF);
  irrecv.enableIRIn();  // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
```

```
44
45      if(results.value == b0) {
46        // Serial.println("'0' is pressed.");
47        digitalWrite(Relay1, OFF);
48      }
49
50      if(results.value == b1) {
51        // Serial.println("'1' is pressed.");
52        digitalWrite(Relay1, ON);
53      }
54
55      if(results.value == b2) {
56        // Serial.println("'2' is pressed.");
57        digitalWrite(Relay2, OFF);
58      }
59
60      if(results.value == b3) {
61        // Serial.println("'3' is pressed.");
62        digitalWrite(Relay2, ON);
63      }
64
65      irrecv.resume();  // Receive the next value
66    }
67    delay(100);
68  }
69
```

แบบฝึกหัดท้ายการทดลอง

จงเขียนต่อวงจรและโค้ดเพิ่มเติมจากการทดลองตอนที่ 2 เพื่อให้รีโมตสามารถควบคุมการทำงานของรีเลย์ทั้ง 4 ตัว โดยในการควบคุมนี้ เราทำได้สองกรณี คือ

1. เพิ่มการเปรียบเทียบค่าที่ตัวรับสัญญาณรับได้ กับค่าของปุ่ม ‘4’, ‘5’, ‘6’, และ ‘7’ หรือ

2. ใช้ 4 ปุ่มเดิม แต่เปลี่ยนแปลงโค้ดให้แต่ละปุ่มใช้ควบคุมรีเลย์ ปุ่มละ 1 ตัว ซึ่งทำได้โดยการเก็บสถานะของรีเลย์ไว้ในหน่วยความจำด้วย (ศึกษาเพิ่มเติมได้จากการทดลองตอนที่ 2 ในบทเสริมเรื่องการใช้งานสวิตช์)