การทดลองที่ 5 : SERVO MOTOR, Buzzer

วัตถุประสงค์

1. เพื่อให้นักศึกษารู้จักการใช้งาน NODEMCU กับสัญญาณ PULSE WIDTH MODULATION
2. เพื่อให้นักศึกษารู้การใช้ NodeMCU กับการควบคุม SERVO MOTOR
3. เพื่อให้นักศึกษาทบทวนการใช้งาน Arduino IDE ที่ได้เรียนมา และใช้งานคำสั่งเพื่อทดลองกับบอร์ด NodeMCU เบื้องต้น

การทดลอง

1. ทำการทดลองตามเอกสารการทดลองนี้เพื่อทดสอบใช้งานบอร์ด NodeMCU

อุปกรณ์ที่ใช้

1. คอมพิวเตอร์ พร้อมโปรแกรม Arduino IDE
2. บอร์ด NodeMCU พร้อมสายเชื่อมต่อ , LED
3. Servo Motor SG90

การทดลองที่ 1 : 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 1.1 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 1.2 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);
  }
}
```

การทดลองที่ 2 : Servo Motor

Servo Motor คือ Motor ที่เราสามารถสั่งงานหรือตั้งค่า แล้วตัว Motor จะหมุนไปยังตำแหน่งองศาที่ เราสั่งได้เองอย่างถูกต้อง โดยใช้การควบคุมแบบป้อนกลับ (Feedback Control)

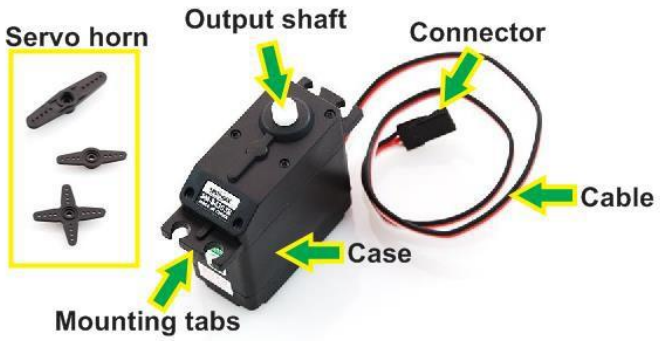
Feedback Control คือ ระบบควบคุมที่มีการวัดค่าเอาต์พุตของระบบนำมาเปรียบเทียบกับค่าอินพุต เพื่อควบคุมและปรับแต่งให้ค่าเอาต์พุตของระบบให้มีค่าเท่ากับ หรือ ใกล้เคียงกับค่าอินพุต

ส่วนประกอบภายนอก Servo Motor

1. Case ตัวถัง หรือ กรอบของตัว Servo Motor
2. Mounting Tab ส่วนจับยึดตัว Servo กับชิ้นงาน
3. Output Shaft เพลาส่งกำลัง
4. Servo Horns ส่วนเชื่อมต่อกับ Output shaft เพื่อสร้างกลไก
5. Cable สายเชื่อมต่อเพื่อ จ่ายไฟฟ้า และ ควบคุม Servo Motor จะประกอบด้วยสายไฟ 3 เส้น และ ใน Servo Motor จะมีสีของสายแตกต่างกันไปดังนี้

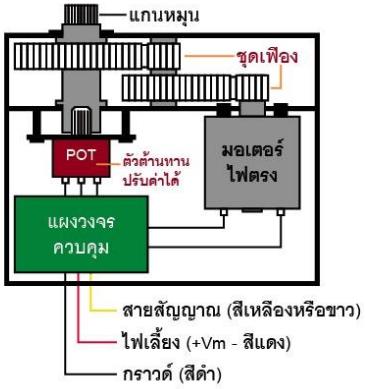
- สายสีแดง คือ ไฟเลี้ยง (4.8-6V)
- สายสีดำ หรือ น้ำตาล คือ กราวด์
- สายสีเหลือง (ส้ม ขาว หรือฟ้า) คือ สายส่งสัญญาณพัลส์ควบคุม (3-5V)

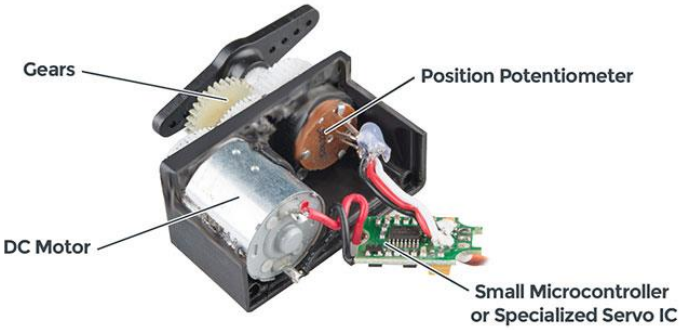
6. Connector จุดเชื่อมต่อสายไฟ



ส่วนประกอบภายนอก Servo Motor

ส่วนประกอบภายใน Servo Motor ซึ่งแสดงดังรูป





1. Motor เป็นส่วนของตัวมอเตอร์
2. Gear Train หรือ Gearbox เป็นชุดเกียร์ทดแรง
3. Position Sensor เป็นเซ็นเซอร์ตรวจจับตำแหน่งเพื่อหาค่าองศาในการหมุน
4. Electronic Control System เป็นส่วนที่ควบคุมและประมวลผล

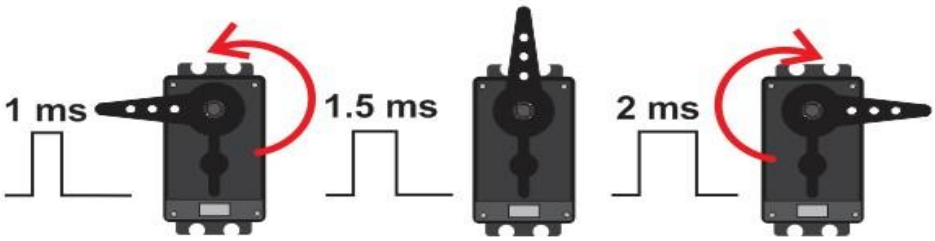
หลักการทำงานของ Servo Motor

เมื่อจ่ายสัญญาณพัลส์เข้ามายัง Servo Motor ส่วนวงจรควบคุม (Electronic Control System) ภายใน Servo จะทำการอ่านและประมวลผลค่าความกว้างของสัญญาณพัลส์ที่ส่งเข้ามาเพื่อแปลค่าเป็นตำแหน่ง องศาที่ต้องการให้ Motor หมุนเคลื่อนที่ไปยังตำแหน่งนั้น แล้วส่งคำสั่งไปทำการควบคุมให้ Motor หมุนไปยัง ตำแหน่งที่ต้องการ โดยมี Position Sensor เป็นตัวเซ็นเซอร์คอยวัดค่ามุมที่ Motor กำลังหมุน เป็น Feedback กลับมาให้วงจรควบคุมเปรียบเทียบกับค่าอินพุตเพื่อควบคุมให้ได้ตำแหน่งที่ต้องการอย่างถูกต้องแม่นยำ

สัญญาณ RC ในรูปแบบ PWM

ตัว Servo Motor ออกแบบมาใช้สำหรับรับคำสั่งจาก Remote Control ที่ใช้ควบคุมของเล่นด้วย สัญญาณวิทยุต่าง ๆ เช่น เครื่องบินบังคับ รถบังคับ เรือบังคับ เป็นต้น ซึ่ง Remote จำพวกนี้ที่ภาครับจะแปลง ความถี่วิทยุออกมาในรูปแบบสัญญาณ PWM (Pulse Width Modulation)

ยกตัวอย่างเช่นหากกำหนดความกว้างของสัญญาณพัลส์ไว้ที่ 1 ms ตัว Servo Motor จะหมุนไปทางด้านซ้ายจนสุด ในทางกลับกันหากกำหนดความกว้างของสัญญาณพัลส์ไว้ที่ 2 ms ตัว Servo Motor จะหมุนไปยังตำแหน่งขวาสุด แต่หากกำหนดความกว้างของสัญญาณพัลส์ไว้ที่ 1.5 ms ตัว Servo Motor ก็จะหมุนมาอยู่ที่ตำแหน่งตรงกลางพอดี



ดังนั้นสามารถกำหนดองศาการหมุนของ Servo Motor ได้โดยการเทียบค่า เช่น Servo Motor สามารถหมุนได้ 180 องศา โดยที่ 0 องศาใช้ความกว้างพัลส์เท่ากับ 1000 us ที่ 180 องศาความกว้างพัลส์เท่ากับ 2000 us เพราะฉะนั้นค่าที่เปลี่ยนไป 1 องศาจะใช้ความกว้างพัลส์ต่างกัน

TowerPro SG90 Servo

Basic Information	
Modulation:	Analog
Torque:	**4.8V:** 25.0 oz-in (1.80 kg-cm)
Speed:	**4.8V:** 0.12 sec/60°
Weight:	0.32 oz (9.0 g)
Dimensions:	Length: 0.91 in (23.0 mm) Width: 0.48 in (12.2 mm) Height: 1.14 in (29.0 mm)
Motor Type:	3-pole
Gear Type:	Plastic
Rotation/Support:	Bushing

Additional Specifications	
Rotational Range:	? (add)
Pulse Cycle:	? (add)
Pulse Width:	500-2400 µs
Connector Type:	JR

ฟังก์ชันภายใน Servo Library

- attach()

คือฟังก์ชันที่ใช้ในการกำหนดขาสัญญาณที่ Servo Motor ต่อกับ Arduino และกำหนดความกว้างของพัลส์ที่ 0 องศาและ 180 องศา

servo.attach(pin,min,max)

pin : คือ ขาสัญญาณของ Arduino ที่ใช้เชื่อมต่อกับ Servo Motor

min : คือ ความกว้างของพัลส์ที่ 0 องศาของ Servo ตัวที่ใช้ในหน่วยไมโครวินาที (us) โดยปกติแล้วหากไม่มีการตั้งค่าโปรแกรมจะกำหนดค่าไว้ที่ 544 us

max : คือ ความกว้างของพัลส์ที่ 180 องศาของ Servo ตัวที่ใช้ในหน่วยไมโครวินาที (us) โดยปกติแล้วหากไม่มีการตั้งค่าโปรแกรมจะกำหนดค่าไว้ที่ 2400 us

- write()

servo.write(angle)

angle : คือมุมที่ต้องการให้ RC Servo Motor แบบ 0-180 องศาหมุนไป แต่หากเป็น Servo Motor แบบ Full Rotation ค่า Angle คือ การกำหนดความเร็ว และทิศทางในการหมุนคือฟังก์ชันที่ใช้ควบคุมตำแหน่งที่ต้องการให้ Servo Motor หมุนไปยังองศาที่กำหนดสามารถกำหนดเป็นค่าองศาได้เลย คือ 0-180 องศา แต่ใน Servo Motor ที่เป็น Full Rotation คำสั่ง write จะเป็นการกำหนดความเร็วในการหมุน โดย

- ค่าเท่ากับ 90 คือคำสั่งให้ Servo Motor หยุดหมุน
- ค่าเท่ากับ 0 คือการหมุนด้วยความเร็วสูงสุดในทิศทางหนึ่ง
- ค่าเท่ากับ 180 คือการหมุนด้วยความเร็วสูงสุดในทิศทางตรงกันข้าม

- writeMicroseconds()

servo.writeMicroseconds(uS)

uS : คือค่าความกว้างของพัลส์ที่ต้องการกำหนดในหน่วยไมโครวินาที (โดยตัวแปร int)

คือ ฟังก์ชันที่ใช้ควบคุมตำแหน่งที่ให้ Servo Motor หมุนไปยังตำแหน่งองศาที่กำหนดโดยกำหนดเป็นค่าความกว้างของพัลส์ในหน่วย us

- read()

servo.read()

Return ค่า 0-180

คือฟังก์ชันอ่านค่าองศาที่สั่งเข้าไปด้วยฟังก์ชัน write() เพื่อให้รู้ว่าตำแหน่งองศาสุดท้ายที่เราสั่งเข้าไปนั้นมีค่าเท่าไหร่ซึ่งค่าที่อ่านออกมานั้นจะมีค่าอยู่ในช่วง 0 – 180

- attached()

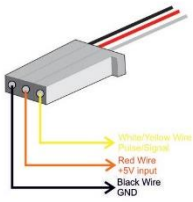
คือฟังก์ชันตรวจสอบว่า Servo ที่เราต้องการใช้กำลังต่ออยู่กับสัญญาณของ Arduino หรือไม่

servo.attached()

จะ Return ค่า True ออกมา หาก Servo Motor เชื่อมต่ออยู่กับ Arduino แต่ถ้าหาก Return ออกมาเป็นค่าอื่นถือว่าไม่เชื่อมต่อ

การทดลอง การควบคุม Servo Motor เบื้องต้น

1. ทำการเชื่อมต่อ SERVO MOTOR SG90 กับ NodeMCU ดังรูป



หมายเหตุ *

สายสีแดง (RED) : VCC 5VDC

สายสีน้ำตาล (Brown) : GND

สายสีส้ม (Orange) : Signal (ในที่นี้ใช้ D2)

2. ทำการดาวน์โหลดไลบลารี Servo.h

Example 2.1 NodeMCU - Servo

```
#include <Servo.h>
Servo myservo;
void setup()
{
      myservo.attach(4); //GPIO 4 (D2)
}

void loop()
{
    myservo.write(0);
    delay(1000);
    myservo.write(90);
    delay(1000);
    myservo.write(180);
    delay(1000);
}
```

สังเกตผลลัพธ์ที่เกิดขึ้นกับ Servo Motor

3. ทดลองตามคำสั่งด้านล่าง

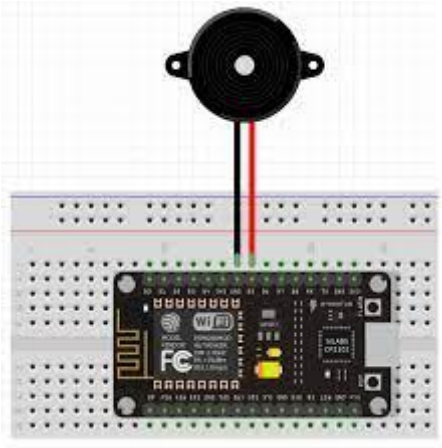
Example 2.2 NodeMCU - Servo

```
#include <Servo.h>
Servo myservo;

int pos = 0;
void setup(){
myservo.attach(D2,544,2400); // GPIO 4 (D2)
}

void loop()
{
 for(pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
      {
   myservo.write(pos); // tell servo to go to position in variable 'pos'
    delay(15); // waits 15ms for the servo to reach the position
       }
       for(pos = 180; pos>=1; pos-=1)// goes from 180 degrees to 0 degrees
      {
  myservo.write(pos); // tell servo to go to position in variable 'pos'
           delay(15); // waits 15ms for the servo to reach the position
      }
     }
```

สังเกตผลลัพธ์ที่เกิดขึ้นกับ Servo Motor

การทดลองที่ 3 : Buzzer

Buzzer หรือบัซเซอร์ คือ ลำโพงแบบแม่เหล็กหรือ แบบเปียโซที่มีวงจรกำเนิดความถี่ (oscillator) อยู่ภายในตัว ใช้ไฟเลี้ยง 3.3 - 5V สามารถสร้างเสียงเตือนหรือส่งสัญญาณที่เป็นรูปแบบต่างๆ เราอาจจะเคยได้ยินเสียง Buzzer อยู่บ่อยๆ เช่น เสียง ปี๊บที่อยู่ในคอมพิวเตอร์ก็ใช้ Buzzer ในการส่งสัญญาณให้ทราบสถานะของคอมพิวเตอร์ให้ทราบว่ามีปัญหาอะไร



โดยทั่วไปแล้ว Buzzer สามารถใช้ digitalWrite เพื่อสร้างสัญญาณเสียงแบบ digital (ดัง/ไม่ดัง) หรือใช้ analogWrite (PWM) เพื่อสร้างสัญญาณเสียงแบบแอนะล็อก (ดังที่ระดับต่าง ๆ ได้) โดยใน Arduino ได้มีไลบลารีสำหรับกำหนดความถี่ที่ต้องการและระยะเวลาในการจ่ายพัลส์ออกไป คือ คำสั่ง tone

tone(pin, frequency, duration)

โดยที่ frequency คือ ความถี่ที่กำหนด และ duration คือระยะเวลาที่ต้องการ (ms)

อันดับแรกทดลอง เมื่อเปลี่ยน duration

Example 3.1 buzzer example

```
const int buzzer_pin = D5;
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
    tone(buzzer_pin, 494, 100);
    delay(1000);
    tone(buzzer_pin, 494, 300);
    delay(1000);
    tone(buzzer_pin, 494, 900);
    delay(1000);
    tone(buzzer_pin, 494, 1500);
    delay(1000);
```

ต่อมาทดลองเปลี่ยน frequency

Example 3.2 buzzer example 2

```
const int buzzer_pin = D5;
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
     tone(buzzer_pin, 94, 1000);
    delay(1000);
    tone(buzzer_pin, 394, 1000);
    delay(1000);
    tone(buzzer_pin, 894, 1000);
    delay(1000);
    tone(buzzer_pin, 1494, 1000);
    delay(1000);
```

คำถามท้ายการทดลอง

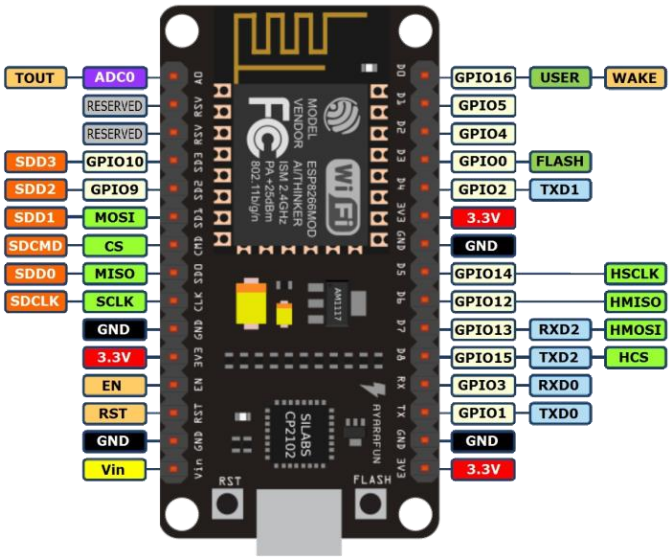
1. ใช้ตัวต้านทานปรับค่าได้ (Potentiometer) ทำการควบคุม servo motor ให้หมุนโดยมีมุมไปตามทิศทางของตัวต้านทานปรับค่าได้ (ตั้งแต่ 0 – 180) องศา โดยกำหนดให้คำสั่งที่ใช้ขับ servo motor คือ servo.writeMicroseconds(uS)
2. ใช้ Buzzer สร้างเพลง ขึ้นมาโดยให้ สร้างตัวแปรอาร์เรย์สองตัวสำหรับเก็บความถี่ และเก็บ duration จากนั้นให้วนลูปเล่นเพลงไปตามอาร์เรย์ทั้งสองตัวนี้ ไปโดยให้มีเวลาประมาณไม่ต่ำกว่า 10 วินาที

<u>การส่งงาน</u>

** เรียกพี่ TA ตรวจ

การใช้งานพอร์ตต่าง ๆ ของ NodeMCU

สำหรับบอร์ด NodeMCU มีรายละเอียดของแต่ละขา ดังนี้



รูปที่ 1 ขาต่าง ๆ ของ NodeMCU ESP8266

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