การทดลองที่ 4 : Basic Analog Output and PWM

วัตถุประสงค์

1. เพื่อให้นักศึกษารู้จักการใช้งาน NODEMCU กับสัญญาณ PULSE WIDTH MODULATION
2. เพื่อให้นักศึกษารู้การใช้ NodeMCU กับการควบคุม LED Fading ด้วย PWM
3. เพื่อให้นักศึกษาทบทวนการใช้งาน Arduino IDE ที่ได้เรียนมา และใช้งานคำสั่งเพื่อทดลองกับบอร์ด NodeMCU เบื้องต้น

การทดลอง

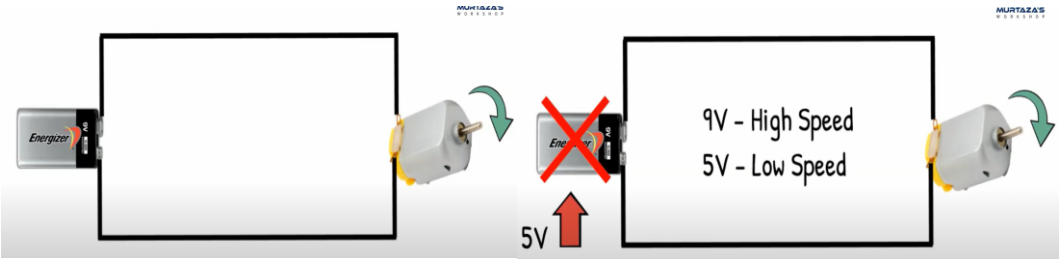
1. ทำการทดลองตามเอกสารการทดลองนี้เพื่อทดสอบใช้งานบอร์ด NodeMCU

อุปกรณ์ที่ใช้

1. คอมพิวเตอร์ พร้อมโปรแกรม Arduino IDE
2. บอร์ด NodeMCU พร้อมสายเชื่อมต่อ , LED, R

Analog Output and PWM

การควบคุมแรงดัน ในอดีตนั้น การควบคุมสิ่งต่าง ๆ ถูกควบคุมด้วยแรงดันดิจิทัลที่มีแค่ HIGH กับ LOW หรือ 5V/3.3V กับ 0V เพื่อควบคุมการทำงาน หรือเปิด-ปิดอะไรต่าง ๆ เมื่อเวลาผ่านไปการควบคุมสั่งการต้องการทำให้ได้ในหลายระดับมากขึ้น และมีความละเอียดมากขึ้นตามไปด้วย หรือต้องการควบคุมสั่งการแบบแอะล็อกนั่นเอง (Analog) ซึ่งวิธีที่นำมาใช้ในระบบควบคุมหรือไมโครคอนโทรลเลอร์นั้น จะยังใช้หลักการของแรงดันดิจิทัลอยู่ แต่จินตนาการเมื่อเราทำการเปิด-ปิดสวิตช์ไฟที่จ่ายแรงดันนี้ด้วยความเร็วสูง พบว่าแรงดันที่วัดได้นั้นเป็นค่าเฉลี่ยของแรงดันสูงสุดที่เราตั้งไว้ และเมื่อปรับความถี่ในการเปิด-ปิดนี้ ก็จะทำให้ค่าเฉลี่ยของแรงดันนี้แตกต่างกันไปด้วย ซึ่งเป็นหลักการของ PWM (Pulse Width Modulation) คือการนำค่าเฉลี่ยของแรงดันมาใช้นั่นเอง

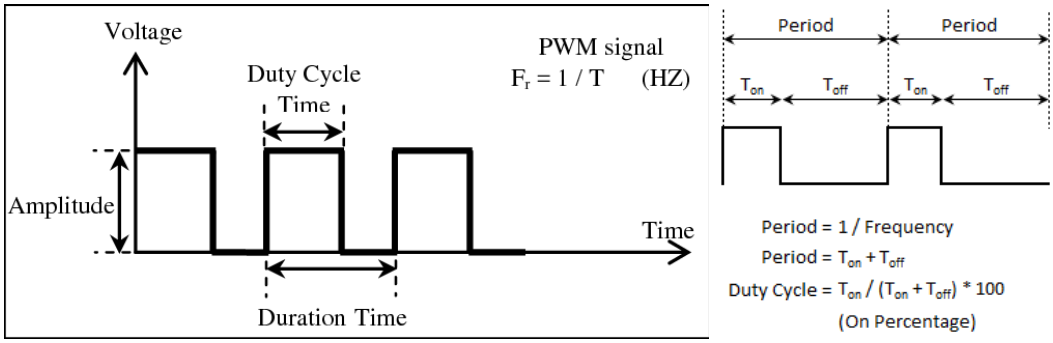


การจะควบคุมมอเตอร์โดยการปรับแรงดัน

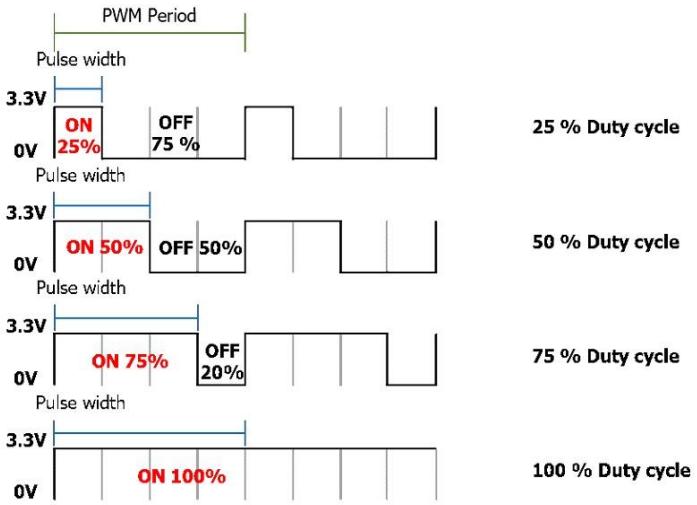
PWM Applications

- Drive buzzer with different loudness
- Control speed of the motor
- Control the direction of a servo
- Provide an analog output
- Generate audio signal
- Telecommunication: Encode message

สัญญาณ PWM จึงเป็นเสมือนการสร้างสัญญาณแอนะล็อกผ่านขาดิจิทัล (GPIO) นั่นเองโดย



duty of cycle = 100 x (ความกว้างของ pulse) / (คาบเวลา)

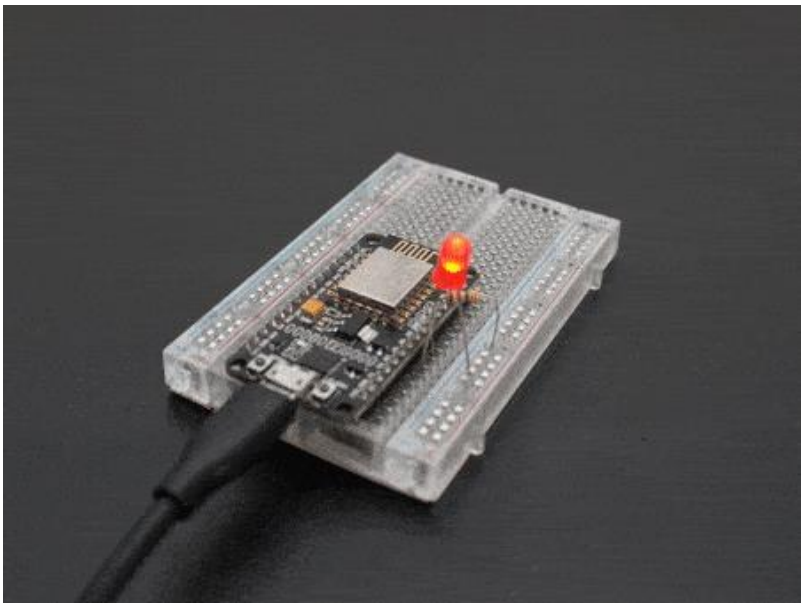


เปรียบเทียบ %duty cycle ค่าต่าง ๆ

ใน ESP8266 สามารถใช้งาน PWM ได้ทุกขา GPIO

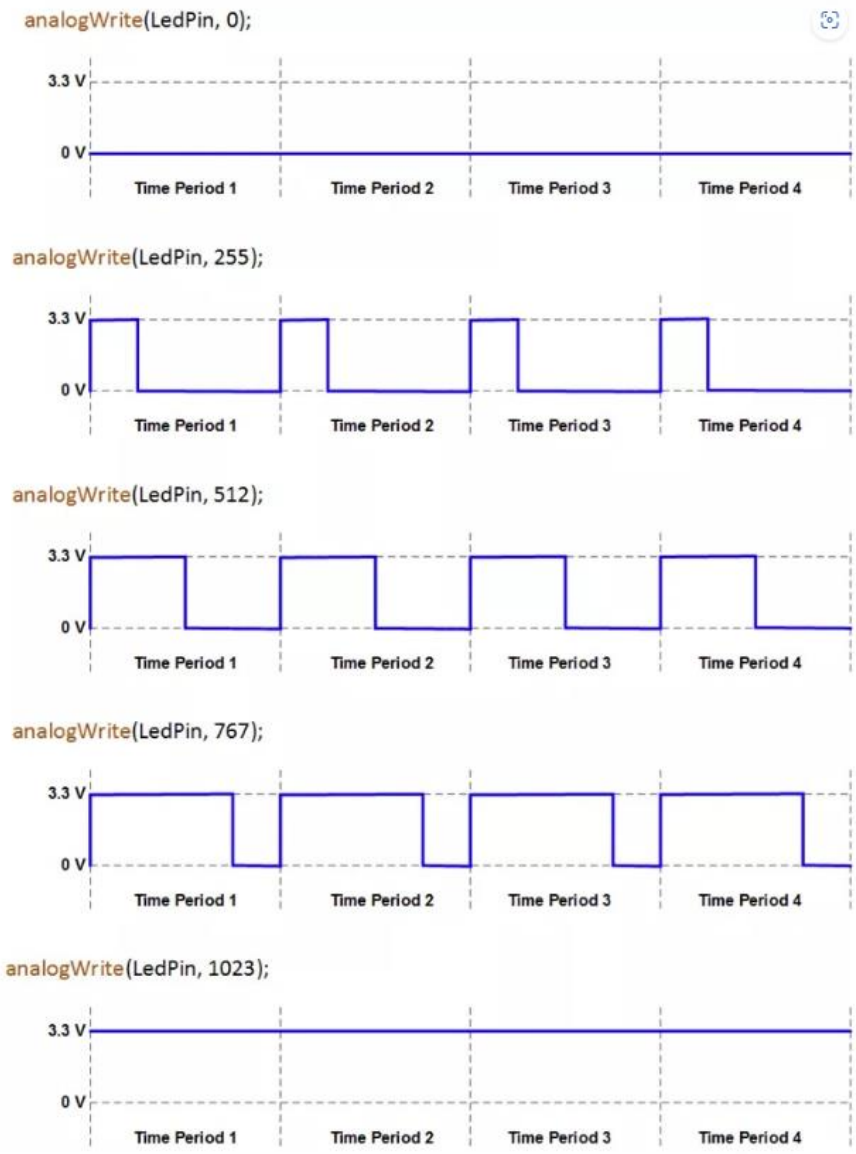
ความถี่ของ GPIO กำหนดได้ที่ 1 ถึง 1000 Hz.

ค่าในเมท็อด .duty คือมีค่าระหว่าง 0 – 1023



Analog Output และสัญญาณ PWM

Pulse width modulation หรือเรียกโดยย่อว่า PWM นั้นเป็นเทคนิคในการสร้างสัญญาณอนาล็อกด้วยค่าเฉลี่ยของสัญญาณดิจิตอล เป็นการควบคุมสัญญาณดิจิตอลให้สร้างสัญญาณคลื่นรูปสี่เหลี่ยม (square wave) ขึ้นมา โดยสลับกันระหว่างสัญญาณสูง และสัญญาณต่ำ โดยสัญญาณที่สร้างขึ้นมานี้สามารถจำลองเป็นค่าอนาล็อกได้ ตามสัดส่วนของสัญญาณสูงและต่ำ สัญญาณ PWM แสดงในรูปที่ 1



รูปที่ 1 สัญญาณ PWM

สำหรับคำสั่งที่ใช้ในการสร้างสัญญาณ PWM หรือสร้างสัญญาณจำลองของอนาล็อกตามขาที่เรากำหนดคือ analogWrite คำสั่งนี้สามารถควบคุมให้ LED ที่อยู่บนบอร์ดไมโครคอนโทรลเลอร์ หรือ LED ที่เราต่อวงจรเพิ่มเข้า

ไป สามารถปรับความสว่างได้ สามารถควบคุมความเร็วในการหมุนของมอเตอร์กระแสตรงได้ หลังจากที่มีการเรียกใช้คำสั่ง analogWrite ขาที่กำหนดจะสร้างสัญญาณคลื่นสี่เหลี่ยมด้วย duty cycle ค่าหนึ่ง จนกว่าจะมีการเรียกคำสั่ง analogWrite ครั้งต่อไป ความถี่ของสัญญาณ PWM นั้นมีค่าประมาณ 490 Hz โดยที่ขา 5 และ ขา 6 ของบอร์ดมีความถี่ประมาณ 980 Hz ในการใช้คำสั่ง analogWrite ไม่จำเป็นต้องมีการใช้คำสั่ง pinMode

analogWrite(pin, value)

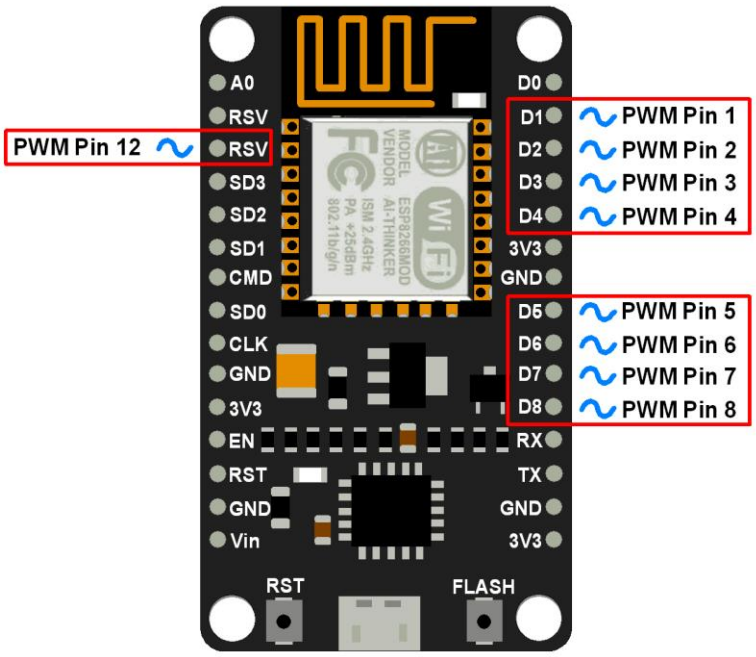
syntax ของการใช้คำสั่ง analogWrite

Parameter ของคำสั่ง analogWrite เป็นดังนี้

pin หมายถึง หมายเลขขาที่ต้องการให้กำเนิดสัญญาณ PWM สำหรับขาที่สามารถสร้างสัญญาณ PWM ได้กล่าวไว้แล้วในบทที่ 1

value หมายถึง เปอร์เซ็นต์ของ duty cycle ที่เราต้องการ โดย 0 หมายถึง 0เปอร์เซ็นต์ และ 1023 หมายถึง 100 เปอร์เซ็นต์

สำหรับ ESP8266 แล้วสามารถใช้งาน PWM ได้ทุกขา I/O



ทำการเชื่อมต่อ LED กับ Pin ขาเบอร์ D1

Code Example 4.1 Basic Analog Write

```
const int ledPin = D1;


void setup() {
   pinMode(ledPin,OUTPUT);
}
void loop() {
    analogWrite(ledPin, 0);
    delay(1000);
    analogWrite(ledPin, 512);
     delay(1000);
     analogWrite(ledPin, 1023);
     delay(1000);
  }
```

สังเกตผลลัพธ์ที่ได้ เมื่อคำสั่งทำงานที่บรรทัดต่าง ๆ
ต่อไปจะเป็นการทดลองเพิ่มความสว่างของแอลอีดี ให้เป็นแบบ Fading ค่อย ๆ สว่าง และค่อย ๆ ดับ โดยจะต้องใช้การวนลูป for ดัง ตัวอย่าง จากนั้น ทดลองเขียนโค้ด 4.2

for (ค่าตัวแปรเริ่มต้น; เงื่อนไขการวนซ้ำ; ตัวแปรเงื่อนไขที่เปลี่ยนแปลง)
{ statement;
}

Code Example 4.2 – LED On-Off Fading

```
const int ledPin = D1;
void setup() {
   pinMode(ledPin,OUTPUT);
}
void loop() {
```

```
  // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle < 1023; dutyCycle++){
    // changing the LED brightness with PWM
    analogWrite(ledPin, dutyCycle);
    delay(1);
  }

  for(int dutyCycle = 1023; dutyCycle > 0; dutyCycle--){
    analogWrite(ledPin, dutyCycle);
    delay(1);
  }
}
```

ผลลัพธ์ที่ได้เป็นอย่างไร ?

หากต้องการให้ LED มีการ Fading เร็วขึ้น ให้ทำการแก้ไข ตัวแปรเงื่อนไขที่เปลี่ยนแปลง เช่น dutyCycle += 30

คำถามท้ายการทดลอง

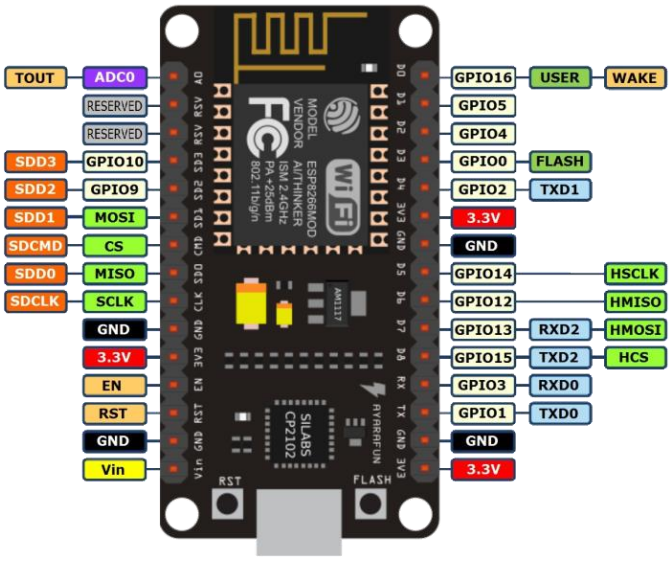
ให้เชื่อมต่อ LDR เพิ่มเติมกับ NodeMCU จากนั้น ทำการเขียนโปรแกรมให้ LED มีค่าความสว่างแบบแอนะล็อกที่แปรผันตรงกับสถานะของ LDR แบบแอนะล็อก

<u>การส่งงาน</u>

** เรียก TA ตรวจ ก่อน 17.00 น.

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