ใบงานที่ 1 การใช้งาน ESP32 LED

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายคุณลักษณะทั่วไปของบอร์ด ESP32 ได้

2.นักศึกษาสามารถติดตั้งและตั้งค่า Arduino IDE สำหรับใช้งาน ESP32 ได้

3.นักศึกษาสามารถเขียนโปรแกรมเบื้องต้นควบคุม LED บนบอร์ด ESP32 ได้

อุปกรณ์ที่ใช้

1.บอร์ด ESP32

2.สาย USB Type-C/Type-Micro (ขึ้นอยู่กับบอร์ด)

3.คอมพิวเตอร์ติดตั้ง Arduino IDE

4.หลอด LED + ตัวต้านทาน 220Ω (กรณีต่อวงจรภายนอก)

ขั้นตอนการทดลอง

1. การติดตั้งบอร์ด ESP32 ใน Arduino IDE

1.1 เปิด Arduino IDE

1.2 ไปที่ File > Preferences

1.3 ในช่อง **Additional Board Manager URLs** ใส่ลิงก์:

```arduino
https://dl.espressif.com/dl/package_esp32_index.json
```

https://dl.espressif.com/dl/package_esp32_index.json

1.4 ไปที่ Tools > Board > Board Manager พิมพ์คำว่า **ESP32** แล้วกดติดตั้ง

2. การเลือกบอร์ดและพอร์ต

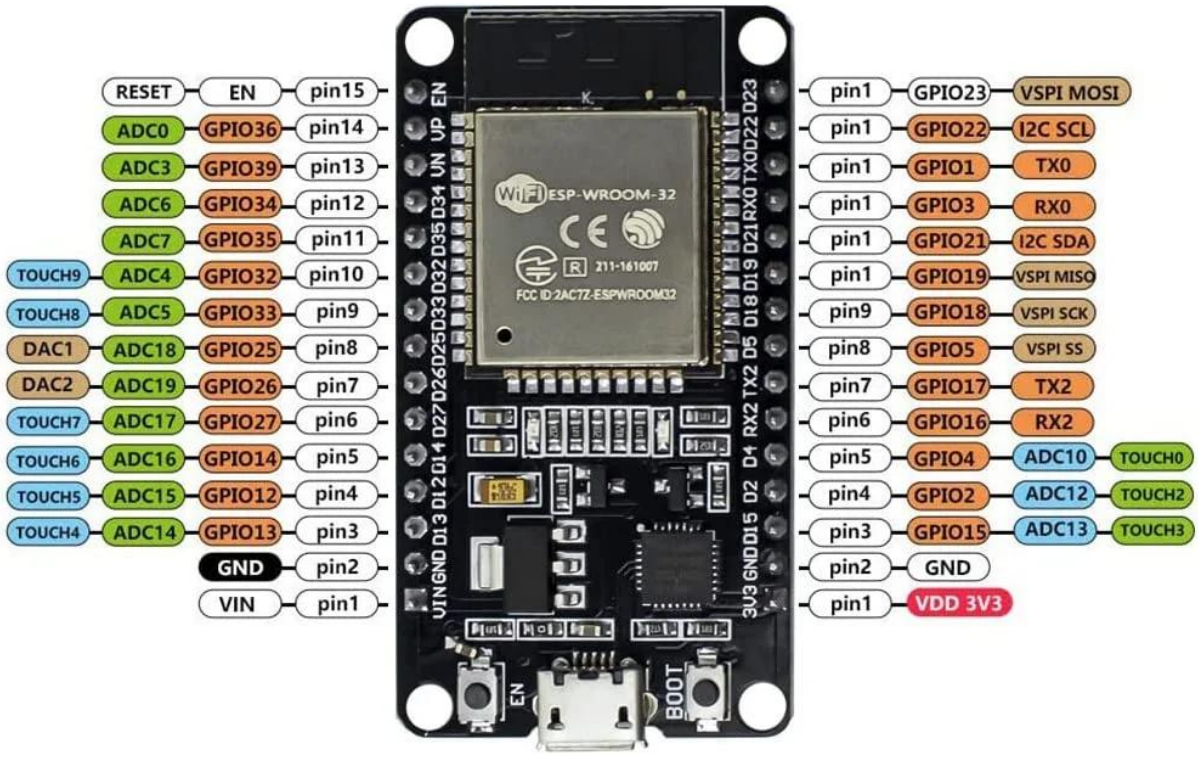
2.1 ไปที่ Tools > Board ⟶ เลือก **ESP32 Dev Module**

2.2 ไปที่ Tools > Port ⟶ เลือกพอร์ตที่เชื่อมต่อกับ ESP32

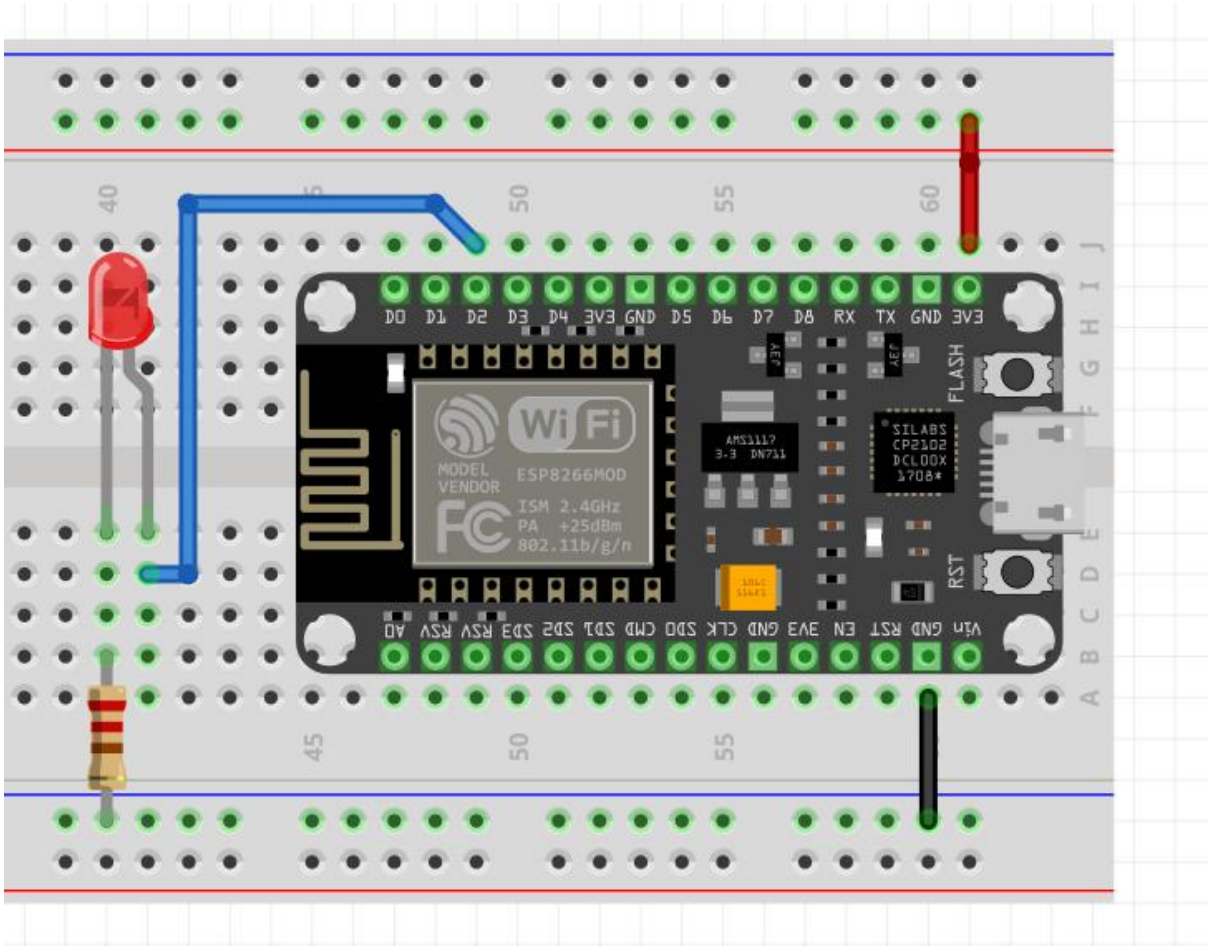
3. การเขียนโปรแกรมทดสอบ LED บนบอร์ด

```
#define LED_PIN 2   // GPIO 2 เป็น LED บนบอร์ด ESP32

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_PIN, HIGH); // LED ติด
  delay(1000);                 // หน่วงเวลา 1 วินาที
  digitalWrite(LED_PIN, LOW);  // LED ดับ
  delay(1000);                 // หน่วงเวลา 1 วินาที
}
```

4.การต่อวงจร



รูปที่1.1แสดงขาของ ESP32



รูปที่ 1.2การต่อวงจร

แบบฝึกหัด / กิจกรรม

1.แก้ไขโปรแกรมให้ LED กระพริบทุก 0.5 วินาที

2.แก้ไขโปรแกรมให้ควบคุม LED 3 ดวงจาก GPIO 2 ไปยัง GPIO 4

3.อธิบายว่าทำไมเราจึงต้องต่อ ตัวต้านทาน เข้ากับ LED

คำถามท้ายใบงาน

1.ESP32 แตกต่างจาก Arduino Unoอย่างไร?

.....

2.ฟังก์ชัน setup() และ loop() ใน ESP32 มีหน้าที่ต่างกันอย่างไร?

.....

ใบงานที่ 2

เรื่อง: งานควบคุมหลอด LED ด้วยสวิตช์และแสดงผลด้วยจอ LCD I2C

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถต่อวงจรควบคุม LED ด้วยสวิตช์บนบอร์ด ESP32 ได้

2.นักศึกษาสามารถเชื่อมต่อและเขียนโปรแกรมใช้งานจอ LCD I2C กับ ESP32 ได้

3.นักศึกษาสามารถประยุกต์ใช้โปรแกรมควบคุม LED พร้อมแสดงข้อความบน LCD ได้

อุปกรณ์ที่ใช้

1.บอร์ด ESP32

2.สวิตช์กดติด–ปล่อยดับ (Tact Switch)

3.หลอด LED 1 ดวง + ตัวต้านทาน 220Ω

4.จอ LCD 16x2 แบบ I2C

5.สาย Jumper + Breadboard

ความรู้เบื้องต้น

- **สวิตช์** ใช้สำหรับสั่งงานแบบ Digital Input (กด = LOW / ปล่อย = HIGH)
- **LED** ใช้เป็นเอาต์พุตสำหรับแสดงผลทางกายภาพ (ติด–ดับ)
- **LCD I2C** ใช้แสดงข้อความหรือตัวเลข โดยเชื่อมต่อกับ ESP32 ผ่านบัส I2C (SDA, SCL) ซึ่งช่วยลดจำนวนสายสัญญาณ

ขั้นตอนการทดลอง

1. การต่อวงจร

GPIO 4 ⟶ LED (ผ่านตัวต้านทาน) ⟶ GND

GPIO 5 ⟶ สวิตช์ ⟶ GND (ใช้ INPUT_PULLUP)

LCD I2C ⟶ ต่อสายดังนี้

SDA ⟶ GPIO 21

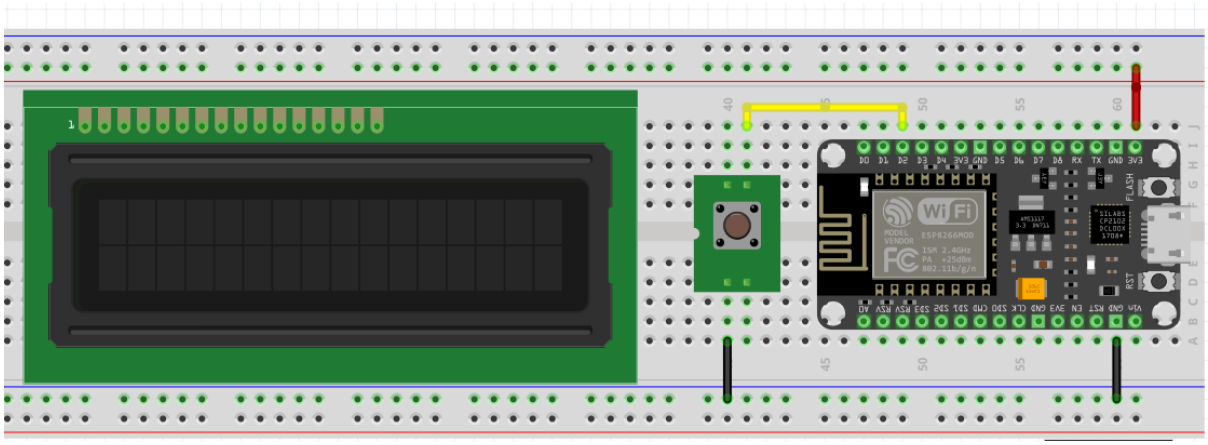
SCL ⟶ GPIO 22

VCC ⟶ 5V

GND ⟶ GND

2. การเขียนโปรแกรม

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
// กำหนดที่อยู่ของ LCD (ปกติ 0x27 หรือ 0x3F)
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define LED_PIN 4
#define SW_PIN  5
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW_PIN, INPUT_PULLUP); // ใช้ internal pull-up
  lcd.init();       // เริ่มต้นใช้งาน LCD
  lcd.backlight();  // เปิดไฟ backlight
  lcd.setCursor(0,0);
  lcd.print("ESP32 LED+SW");
}
void loop() {
  int swState = digitalRead(SW_PIN);
  if (swState == LOW) {        // เมื่อกดสวิตช์
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,1);
    lcd.print("LED: ON        ");
  } else {                     // เมื่อปล่อยสวิตช์
    digitalWrite(LED_PIN, LOW);
    lcd.setCursor(0,1);
    lcd.print("LED: OFF       ");
  }
}
```



รูปที่ 2.1 แสดงการต่อวงจร

แบบฝึกหัด / กิจกรรม

1.แก้ไขโปรแกรมให้กดสวิตช์แล้ว LED ติดค้าง 3 วินาที ก่อนดับ

2.ทดลองเพิ่ม LED อีกดวง แล้วแสดงผลบน LCD ว่า LED1 / LED2 ติดหรือดับ

3.อธิบายความแตกต่างระหว่าง INPUT และ INPUT_PULLUP

คำถามท้ายใบงาน

1. บัส I2C ที่ใช้ต่อ LCD มีขาอะไรบ้าง และต่อกับ GPIO ใดของ ESP32?

   .....

2. คำสั่ง lcd.setCursor(0,1); มีความหมายว่าอย่างไร?

   .....

3. ถ้าอยากให้ LCD แสดงจำนวนครั้งที่กดสวิตช์ ต้องเขียนโปรแกรมเพิ่มอย่างไร?

   .....

ใบงานที่ 3

เรื่อง: ชนิดข้อมูลและตัวแปรควบคุมหลอด LED โดยใช้ ESP32 และสวิตช์

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายความหมายของชนิดข้อมูลและตัวแปรในภาษา C/Arduino ได้

2.นักศึกษาสามารถเขียนโปรแกรม ESP32 ที่ใช้ตัวแปรควบคุมการทำงานของ LED และสวิตช์ได้

3.นักศึกษาสามารถทดลองและสังเกตผลการใช้ตัวแปรในการควบคุมเอาต์พุตและการนับจำนวนครั้งได้

อุปกรณ์ที่ใช้

1.บอร์ด ESP32

2.หลอด LED + ตัวต้านทาน 220Ω จำนวน 1–2 ดวง

3.สวิตช์กดติด–ปล่อยดับ (Tact Switch) 1–2 ตัว

4.สาย Jumper + Breadboard

5.คอมพิวเตอร์ติดตั้ง Arduino IDE

ความรู้เบื้องต้น

- **ตัวแปร (Variable)** คือพื้นที่เก็บข้อมูลในหน่วยความจำที่สามารถเปลี่ยนค่าได้
- **ชนิดข้อมูล (Data type)** ที่ใช้บ่อยใน Arduino/ESP32 ได้แก่
    - int เก็บค่าจำนวนเต็ม
    - bool เก็บค่าจริง/เท็จ (true/false)
    - float เก็บค่าทศนิยม
    - String เก็บข้อความ
- การใช้ตัวแปรช่วยให้โปรแกรมอ่านง่ายและปรับเปลี่ยนสะดวก

ตัวอย่างโปรแกรม

ตัวอย่างที่ 1: ใช้ตัวแปรออก Serial Monitor

```
int value = 0;  // ตัวแปรจำนวนเต็ม
void setup() {
  Serial.begin(9600);
}
void loop() {
  value = value + 1;            // เพิ่มค่าทีละ 1
  Serial.println(value);        // แสดงผลใน Serial Monitor
  delay(1000);
}
```

ตัวอย่างที่ 2: ใช้ตัวแปรออก Serial และกดสวิตช์ควบคุม LED

```
#define LED_PIN 4
#define SW_PIN  5
int swState = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  swState = digitalRead(SW_PIN);
  if (swState == LOW) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("LED ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LED OFF");
  }
  delay (200);}
```

ตัวอย่างที่ 3: ใช้ตัวแปรออก Serial กดสวิตช์ 2 ตัวควบคุม LED

```
#define LED_PIN 4
#define SW1_PIN 5
#define SW2_PIN 18
int sw1State = 0;
int sw2State = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW1_PIN, INPUT_PULLUP);
  pinMode(SW2_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  sw1State = digitalRead(SW1_PIN);
  sw2State = digitalRead(SW2_PIN);
  if (sw1State == LOW || sw2State == LOW) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("LED ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LED OFF");
  }
}
```

ตัวอย่างที่ 4: ใช้ตัวแปรออก Serial กดสวิตช์ 2 ตัวควบคุม LED และนับจำนวนครั้ง

```
#define LED_PIN 4
#define SW1_PIN 5
#define SW2_PIN 18
int sw1State = 0;
int sw2State = 0;
int count = 0;        // ตัวแปรเก็บจำนวนครั้งที่กด
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW1_PIN, INPUT_PULLUP);
  pinMode(SW2_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  sw1State = digitalRead(SW1_PIN);
  sw2State = digitalRead(SW2_PIN);
  if (sw1State == LOW || sw2State == LOW) {
    count++;
    Serial.print("Count: ");
    Serial.println(count);
    if (count >= 5) {           // เมื่อกดครบ 5 ครั้ง
      digitalWrite(LED_PIN, HIGH);
      Serial.println("LED ON");
    }
    delay(500); // ป้องกันการนับหลายครั้งจากการกดค้าง
  }
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 2 ให้ใช้ตัวแปร bool แทน int

2.เพิ่มตัวนับจำนวนครั้งที่กดสวิตช์ในตัวอย่างที่ 2 และแสดงผลบน Serial Monitor

3.ทดลองให้ LED ติดค้าง 3 วินาที หลังจากกดครบ 3 ครั้งในตัวอย่างที่ 4

คำถามท้ายใบงาน

1. ชนิดข้อมูล int และ bool แตกต่างกันอย่างไร?

   .....

2. ทำไมในโปรแกรมต้องใช้ INPUT_PULLUP แทน INPUT?

   .....

3. หากต้องการให้ LED ติดเมื่อกดครบ 10 ครั้ง ต้องแก้ไขโปรแกรมตัวอย่างที่ 4 อย่างไร?

   .....

ใบงานที่ 4

เรื่อง: ตัวดำเนินการ (Operator) ที่ใช้ควบคุมหลอด LED และแสดงผลบนจอ LCD I2C

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายชนิดของตัวดำเนินการ (Operator) ในภาษา C/Arduino ได้

2.นักศึกษาสามารถเขียนโปรแกรม ESP32 ควบคุม LED โดยใช้ตัวดำเนินการต่าง ๆ ได้

3.นักศึกษาสามารถแสดงผลสถานะของ LED ผ่านจอ LCD I2C ได้

อุปกรณ์ที่ใช้

1.บอร์ด ESP32

2.หลอด LED + ตัวต้านทาน 220Ω จำนวน 1–2 ดวง

3.สวิตช์กดติด–ปล่อยดับ 1–2 ตัว

4.จอ LCD 16x2 พร้อมโมดูล I2C

5.สาย Jumper + Breadboard

ความรู้เบื้องต้น

- **ตัวดำเนินการ (Operator)** คือเครื่องหมายที่ใช้ในการคำนวณ หรือตรวจสอบเงื่อนไขในโปรแกรม
    - ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic): +, -, *, /, %
    - ตัวดำเนินการเปรียบเทียบ (Comparison): ==, !=, >, <, >=, <=
    - ตัวดำเนินการตรรกะ (Logical): &&, ||, !
    - ตัวดำเนินการกำหนดค่า (Assignment): =, +=, -=

การใช้ตัวดำเนินการร่วมกับตัวแปร ทำให้สามารถควบคุมการทำงานของ LED ได้หลากหลายรูปแบบ

ตัวอย่างโปรแกรม

ตัวอย่างที่ 1: Operator กำหนดค่า (=)

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define LED_PIN 4
void setup() {
  pinMode(LED_PIN, OUTPUT);
  lcd.init();
  lcd.backlight();
}
void loop() {
  digitalWrite(LED_PIN, HIGH); // LED ติด
  lcd.setCursor(0,0);
  lcd.print("LED = ON ");
  delay(1000);
  digitalWrite(LED_PIN, LOW);  // LED ดับ
  lcd.setCursor(0,0);
  lcd.print("LED = OFF");
  delay(1000);
}
```

ตัวอย่างที่ 2: Operator เปรียบเทียบ (==) + สวิตช์ควบคุม LED

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define LED_PIN 4
#define SW_PIN  5
int swState = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW_PIN, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
}
void loop() {
  swState = digitalRead(SW_PIN);
  if (swState == LOW) {
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,0);
    lcd.print("LED == ON ");
  } else {
    digitalWrite(LED_PIN, LOW);
    lcd.setCursor(0,0);
    lcd.print("LED == OFF");
  }
}
```

ตัวอย่างที่ 3: Operator ตรรกะ (&&) ควบคุม LED ด้วยสวิตช์ 2 ตัว

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define LED_PIN 4
#define SW1_PIN 5
#define SW2_PIN 18
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW1_PIN, INPUT_PULLUP);
  pinMode(SW2_PIN, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
}
void loop() {
  int sw1 = digitalRead(SW1_PIN);
  int sw2 = digitalRead(SW2_PIN);
  if (sw1 == LOW && sw2 == LOW) {  // ต้องกดพร้อมกัน
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,0);
    lcd.print("LED && = ON ");
  } else {
    digitalWrite(LED_PIN, LOW);
    lcd.setCursor(0,0);
    lcd.print("LED && = OFF");
  }
}
```

ตัวอย่างที่ 4: Operator เพิ่มค่า (++) + เงื่อนไขเปรียบเทียบ (>=)

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define LED_PIN 4
#define SW_PIN  5
int count = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(SW_PIN, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Press Button");
}
void loop() {
  if (digitalRead(SW_PIN) == LOW) {
    count++;              // เพิ่มค่าทีละ 1
    lcd.setCursor(0,1);
    lcd.print("Count: ");
    lcd.print(count);
    delay(400);           // ป้องกันการนับซ้ำ

    if (count >= 5) {     // เมื่อกดครบ 5 ครั้ง
      digitalWrite(LED_PIN, HIGH);
      lcd.setCursor(0,0);
      lcd.print("LED >=5: ON ");
    }
  }
}
```

แบบฝึกหัด / กิจกรรม

1.แก้ไขโปรแกรมตัวอย่างที่ 3 ให้ใช้ตัวดำเนินการ || (OR) แทน &&

2.เพิ่มตัวดำเนินการ % เพื่อให้ LED กระพริบทุกครั้งที่กดปุ่มครบ 2 ครั้ง

3.ให้โปรแกรมนับจำนวนครั้งและแสดงบน LCD จนถึง 10 แล้วรีเซ็ตค่าเป็น 0

คำถามท้ายใบงาน

ตัวดำเนินการ = และ == แตกต่างกันอย่างไร?

.....

ถ้าใช้ || (OR) แทน && (AND) ผลลัพธ์ของตัวอย่างที่ 3 จะเปลี่ยนไปอย่างไร?

.....

อธิบายการทำงานของคำสั่ง count++ และ if(count >= 5) ในตัวอย่างที่ 4

.....

ใบงานที่ 5

การใช้งานคำสั่ง if–else และ switch–case บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของคำสั่ง **if–else** และ **switch–case** ได้

2.นักศึกษาสามารถเขียนโปรแกรม ESP32 ที่อ่านค่าจากสวิตช์ และควบคุม LED ได้

3.นักศึกษาสามารถใช้ Serial Monitor และ LCD I2C แสดงผลการทำงานได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.สาย USB สำหรับเชื่อมต่อคอมพิวเตอร์

3.LED 3 ดวง + ตัวต้านทาน 220Ω

4.สวิตช์กด 1–2 ตัว

5.จอ LCD 16x2 I2C

6.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **if–else** ใช้ตรวจสอบเงื่อนไขและเลือกทำงานตามจริง/เท็จ
- **switch–case** ใช้เลือกทำงานตามค่าของตัวแปร (หลายทางเลือก)
- **Serial Monitor** ใช้พิมพ์ข้อความจาก ESP32 เพื่อดูผลลัพธ์การทำงาน
- **LCD I2C** ใช้แสดงข้อความได้สะดวกโดยใช้เพียง 2 ขา (SDA, SCL)

ตัวอย่างโปรแกรม

ตัวอย่างที่ 5.1 if–else ควบคุม LED ด้วยสวิตช์ + Serial Monitor

```
#define BUTTON_PIN 4
#define LED_PIN 2
void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == LOW) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("Button Pressed: LED ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("Button Released: LED OFF");
  }
  delay(300);
}
```

ตัวอย่างที่ 5.2 if–else ควบคุม LED ด้วยสวิตช์ + LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define BUTTON_PIN 4
#define LED_PIN 2
LiquidCrystal_I2C lcd(0x27, 16, 2); // กำหนดที่อยู่จอ LCD
void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("ESP32 IF-ELSE");
}
void loop() {
  int buttonState = digitalRead(BUTTON_PIN);
  if (buttonState == LOW) {
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,1);
    lcd.print("LED ON ");
  } else {
    digitalWrite(LED_PIN, LOW);
    lcd.setCursor(0,1);
    lcd.print("LED OFF");
  }
  delay(300);
}
```

ตัวอย่างที่ 5.3 switch–case ควบคุม LED 3 ดวง + Serial Monitor

```
#define LED1 16
#define LED2 17
#define LED3 18
int mode = 1;
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  switch (mode) {
    case 1:
      digitalWrite(LED1, HIGH);
      digitalWrite(LED2, LOW);
      digitalWrite(LED3, LOW);
      Serial.println("Mode 1: LED1 ON");
      break;
    case 2:
      digitalWrite(LED1, LOW);
      digitalWrite(LED2, HIGH);
      digitalWrite(LED3, LOW);
      Serial.println("Mode 2: LED2 ON");
      break;
    case 3:
      digitalWrite(LED1, LOW);
      digitalWrite(LED2, LOW);
      digitalWrite(LED3, HIGH);
      Serial.println("Mode 3: LED3 ON");
      break;
    default:
```

```
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    Serial.println("No LED ON");
  }
  delay(1000);
  mode++;
  if (mode > 3) mode = 1;
}
```

ตัวอย่างที่ 5.4 switch–case ควบคุม LED + LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LED1 16
#define LED2 17
#define LED3 18
int mode = 1;
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("ESP32 SwitchCase");
}
void loop() {
  switch (mode) {
    case 1:
      digitalWrite(LED1, HIGH);
      digitalWrite(LED2, LOW);
      digitalWrite(LED3, LOW);
      lcd.setCursor(0,1);
      lcd.print("LED1 ON ");
      break;
    case 2:
      digitalWrite(LED1, LOW);
      digitalWrite(LED2, HIGH);
      digitalWrite(LED3, LOW);
      lcd.setCursor(0,1);
      lcd.print("LED2 ON ");
      break;
```

```
    case 3:
      digitalWrite(LED1, LOW);
      digitalWrite(LED2, LOW);
      digitalWrite(LED3, HIGH);
      lcd.setCursor(0,1);
      lcd.print("LED3 ON ");
      break;
    default:
      digitalWrite(LED1, LOW);
      digitalWrite(LED2, LOW);
      digitalWrite(LED3, LOW);
      lcd.setCursor(0,1);
      lcd.print("All OFF ");
  }
  delay(1000);
  mode++;
  if (mode > 3) mode = 1;
}
```

แบบฝึกหัด / กิจกรรม

1.แก้ไขโปรแกรมตัวอย่างที่ 5.1 ให้กดปุ่มครั้งแรก LED ติด กดอีกครั้ง LED ดับ (toggle)

2.แก้ไขโปรแกรมตัวอย่างที่ 5.2 ให้แสดงข้อความ "ON" และ "OFF" สลับกันพร้อมแสดงเวลาที่เปลี่ยนสถานะ

3.ปรับโปรแกรมตัวอย่างที่ 5.3 ให้ LED วิ่งจากซ้ายไปขวา แล้วกลับขวาไปซ้าย

4.ปรับโปรแกรมตัวอย่างที่ 5.4 ให้กดปุ่มสวิตช์เปลี่ยนโหมดการทำงานของ LED

คำถามท้ายใบงาน

1.คำสั่ง if–else กับ switch–case แตกต่างกันอย่างไร?

.....

2.ถ้าเราต้องการควบคุม LED 10 ดวงตามลำดับ ควรใช้ if–else หรือ switch–case เพราะเหตุใด?

.....

3.Serial Monitor และ LCD I2C แตกต่างกันในการใช้งานอย่างไร?

.....

ใบงานที่ 6 การวนรอบทำซ้ำ (Loops) บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของคำสั่งวนรอบ for และ while ได้

2.นักศึกษาสามารถเขียนโปรแกรม ESP32 ที่ใช้คำสั่งวนรอบควบคุม LED ได้

3.นักศึกษาสามารถแสดงผลลัพธ์การทำงานทาง Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.สาย USB สำหรับเชื่อมต่อคอมพิวเตอร์

3.LED 3–5 ดวง + ตัวต้านทาน 220Ω

4.สวิตช์กด 1–2 ตัว

5.จอ LCD 16x2 I2C

6.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **Loop (การวนซ้ำ)** คือการสั่งให้โปรแกรมทำงานซ้ำตามจำนวนครั้ง หรือจนกว่าเงื่อนไขจะเปลี่ยน
- **for loop** ⟶ ใช้เมื่อรู้จำนวนรอบที่แน่นอน
- **while loop** ⟶ ใช้เมื่อจำนวนรอบไม่แน่นอน แต่มีเงื่อนไขควบคุม
- ESP32 สามารถใช้ Loops ในการควบคุม LED เช่น ไฟวิ่ง ไฟกระพริบหลายดวง

ตัวอย่างโปรแกรม

ตัวอย่างที่ 6.1 for loop + Serial Monitor

```
#define LED_PIN 2
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  for (int i = 0; i < 5; i++) {
    digitalWrite(LED_PIN, HIGH);
    Serial.print("Loop round: ");
    Serial.println(i+1);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500);
  }
  delay(2000);
}
```

ตัวอย่างที่ 6.2 for loop + LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LED_PIN 2
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  pinMode(LED_PIN, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("For Loop Example");
}
void loop() {
  for (int i = 1; i <= 5; i++) {
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,1);
    lcd.print("Round: ");
    lcd.print(i);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    delay(500);
  }
  delay(2000);
}
```

ตัวอย่างที่ 6.3 while loop + ปุ่มกด + Serial Monitor

```
#define LED_PIN 2
#define BUTTON_PIN 4
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  int count = 0;
  while (digitalRead(BUTTON_PIN) == LOW) { // กดปุ่มค้าง
    digitalWrite(LED_PIN, HIGH);
    Serial.print("Button pressed, Count: ");
    Serial.println(++count);
    delay(300);
    digitalWrite(LED_PIN, LOW);
    delay(300);
  }
}
```

ตัวอย่างที่6.4 while loop + ปุ่มกด + LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LED_PIN 2
#define BUTTON_PIN 4
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("While Loop");
}
void loop() {
  int count = 0;
  while (digitalRead(BUTTON_PIN) == LOW) {
    digitalWrite(LED_PIN, HIGH);
    lcd.setCursor(0,1);
    lcd.print("Count: ");
    lcd.print(++count);
    delay(300);
    digitalWrite(LED_PIN, LOW);
    delay(300);
  }
}
```

แบบฝึกหัด / กิจกรรม

1.แก้ไขโปรแกรมตัวอย่างที่ 6.1 ให้ LED กระพริบตามจำนวนครั้งที่ผู้ใช้กำหนด (เช่น 10 ครั้ง)

2.ปรับโปรแกรมตัวอย่างที่ 6.2 ให้ LED วิ่งจากซ้ายไปขวา (LED 3 ดวง)

3.แก้โปรแกรมตัวอย่างที่ 6.3 ให้เมื่อปล่อยปุ่ม LED หยุดทันที และ Serial พิมพ์ว่า “Stopped”

4.แก้โปรแกรมตัวอย่างที่ 6.4 ให้ LCD แสดง “Press Button” เมื่อไม่กด และแสดงจำนวนรอบเมื่อกดปุ่ม

คำถามท้ายใบงาน

1.คำสั่ง for loop และ while loop ต่างกันอย่างไร?

.....

2.ถ้าต้องการให้ LED กระพริบไม่รู้จบ ควรใช้ for loop หรือ while loop เพราะเหตุใด?

.....

3.หากต้องการนับจำนวนครั้งที่กดปุ่ม แล้วเก็บค่าไว้ ควรใช้ loop แบบใด และต้องมีตัวแปรชนิดใดช่วย?

.....

ใบงานที่ 7 การแสดงผลด้วย 7-Segment Display บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของ **7-Segment Display** ได้

2.นักศึกษาสามารถต่อวงจร 7-Segment Display กับ ESP32 ได้

3.นักศึกษาสามารถเขียนโปรแกรมให้ ESP32 แสดงตัวเลขบน 7-Segment ได้

อุปกรณ์ที่ใช้

1.บอร์ด ESP32

2.สาย USB สำหรับเชื่อมต่อคอมพิวเตอร์

3.7-Segment Display (ชนิด **Common Cathode** หรือ **Common Anode**)

4.ตัวต้านทาน 220Ω (จำนวน 7 ตัวขึ้นไป)

5.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **7-Segment Display** มี LED 7 ดวง (a–g) + จุดทศนิยม (dp)
- ใช้แสดงตัวเลข 0–9 และตัวอักษรบางตัว
- การต่อแบบ Common Cathode ⟶ ขา GND ร่วมกัน, Common Anode ⟶ ขา VCC ร่วมกัน
- ESP32 สามารถควบคุมได้โดยการกำหนดค่า HIGH/LOW ที่ขา GPIO

ตัวอย่างโปรแกรม

ตัวอย่างที่ 7.1 การใช้ 7-segment แสดงเลข 0–9 อัตโนมัติ

```
int seg[] = {23, 22, 21, 19, 18, 5, 4}; // a,b,c,d,e,f,g
byte digits[10][7] = {
  {1,1,1,1,1,1,0}, //0
  {0,1,1,0,0,0,0}, //1
  {1,1,0,1,1,0,1}, //2
  {1,1,1,1,0,0,1}, //3
  {0,1,1,0,0,1,1}, //4
  {1,0,1,1,0,1,1}, //5
  {1,0,1,1,1,1,1}, //6
  {1,1,1,0,0,0,0}, //7
  {1,1,1,1,1,1,1}, //8
  {1,1,1,1,0,1,1}  //9
};
void displayDigit(int num) {
  for(int i=0;i<7;i++){
    digitalWrite(seg[i], digits[num][i]);
  }
}
void setup() {
  for(int i=0;i<7;i++) pinMode(seg[i], OUTPUT);
}
void loop() {
  for(int n=0;n<=9;n++){
    displayDigit(n);
    delay(1000);
  }
}
```

ตัวอย่างที่ 7.2 การใช้ 7-segment กดสวิตช์เปลี่ยนเลข 0–9

```
#define BUTTON_PIN 15
int seg[] = {23, 22, 21, 19, 18, 5, 4};
byte digits[10][7] = { /* เหมือนเดิม */ };
int num = 0;
void displayDigit(int num) {
  for(int i=0;i<7;i++) digitalWrite(seg[i], digits[num][i]);
}
void setup() {
  for(int i=0;i<7;i++) pinMode(seg[i], OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}
void loop() {
  if(digitalRead(BUTTON_PIN) == LOW){
    num++;
    if(num > 9) num = 0;
    displayDigit(num);
    delay(300);
  }
}
```

ตัวอย่างที่ 7.2 การใช้ 7-segment กดสวิตช์เปลี่ยนเลข 0–9

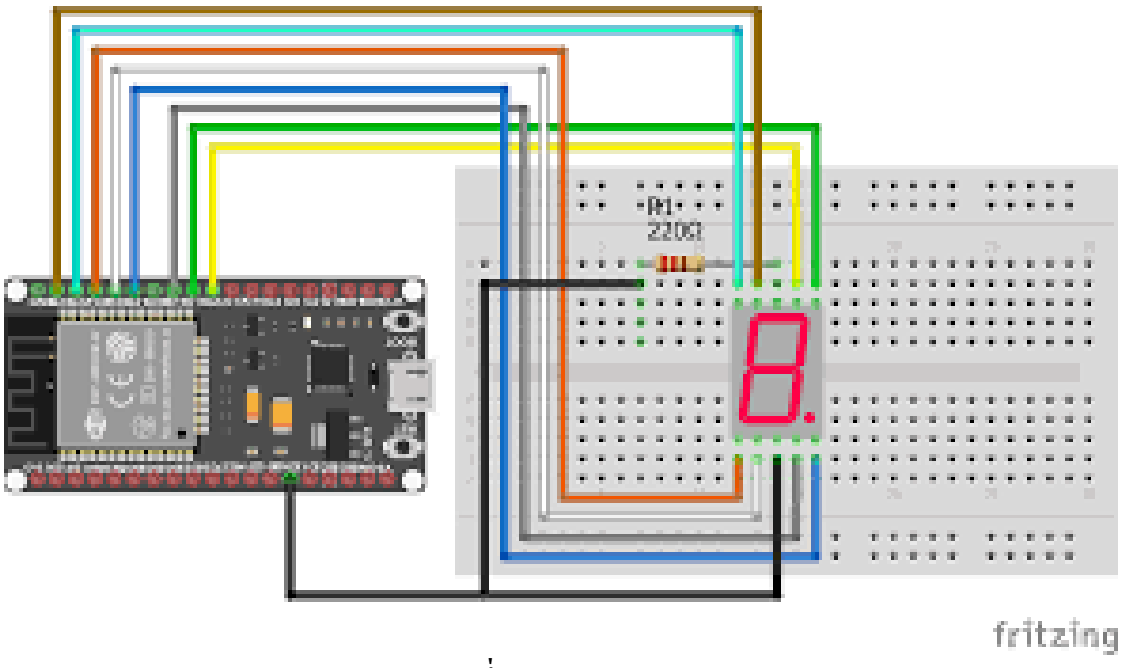
```
#define BUTTON_PIN 15
int seg[] = {23, 22, 21, 19, 18, 5, 4};
byte digits[10][7] = { /* เหมือนเดิม */ };
int num = 0;
void displayDigit(int num) {
  for(int i=0;i<7;i++) digitalWrite(seg[i], digits[num][i]);
}
void setup() {
  for(int i=0;i<7;i++) pinMode(seg[i], OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}
void loop() {
  if(digitalRead(BUTTON_PIN) == LOW){
    num++;
    if(num > 9) num = 0;
    displayDigit(num);
    delay(300);
  }
}
```

ตัวอย่างที่ 7.3 ใช้ 7-segment + 2 สวิตช์ (เพิ่ม/ลดตัวเลข)

```
#define BUTTON_UP   15
#define BUTTON_DOWN 2
int seg[] = {23, 22, 21, 19, 18, 5, 4};
byte digits[10][7] = { /* เหมือนเดิม */ };

int num = 0;

void displayDigit(int num) {
  for(int i=0;i<7;i++) digitalWrite(seg[i], digits[num][i]);
}

void setup() {
  for(int i=0;i<7;i++) pinMode(seg[i], OUTPUT);
  pinMode(BUTTON_UP, INPUT_PULLUP);
  pinMode(BUTTON_DOWN, INPUT_PULLUP);
}

void loop() {
  if(digitalRead(BUTTON_UP) == LOW){
    num++;
    if(num > 9) num = 0;
    displayDigit(num);
    delay(300);
  }
  if(digitalRead(BUTTON_DOWN) == LOW){
    num--;
    if(num < 0) num = 9;
    displayDigit(num);
    delay(300);
  }
}
```

ตัวอย่างที่ 7.4 การใช้ 7-segment + LCD I2C แสดงผลซ้ำกัน

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define BUTTON_PIN 15
int seg[] = {23, 22, 21, 19, 18, 5, 4};
byte digits[10][7] = { /* เหมือนเดิม */ };
int num = 0;
LiquidCrystal_I2C lcd(0x27,16,2);
void displayDigit(int num) {
  for(int i=0;i<7;i++) digitalWrite(seg[i], digits[num][i]);
}
void setup() {
  for(int i=0;i<7;i++) pinMode(seg[i], OUTPUT);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("7-Seg + LCD");
}
void loop() {
  if(digitalRead(BUTTON_PIN) == LOW){
    num++;
    if(num > 9) num = 0;
    displayDigit(num);
    lcd.setCursor(0,1);
    lcd.print("Number: ");
    lcd.print(num);
    lcd.print(" ");
    delay(300);
  }
}
```



รูปที่ 7.1แสดงการต่อ

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 7.1 ให้เลขแสดงย้อนหลังจาก 9 ⟶ 0

2.ปรับโปรแกรมตัวอย่างที่ 7.2 ให้กดปุ่มค้างแล้วตัวเลขเพิ่มขึ้นเรื่อย ๆ

3.แก้ไขโปรแกรมตัวอย่างที่ 7.3 ให้กดปุ่ม + หรือ – แล้วแสดงค่าปัจจุบันทาง Serial Monitor ด้วย

4.ปรับโปรแกรมตัวอย่างที่ 7.4 ให้ LCD แสดงเวลาที่ตัวเลขเปลี่ยนด้วย millis()

คำถามท้ายใบงาน

1.7-segment แบบ Common Cathode และ Common Anode ต่างกันอย่างไร?

.....

2.หากต้องการแสดงตัวเลขมากกว่า 1 หลัก (เช่น 2 หลัก) ควรใช้วิธีใด?

.....

3.LCD I2C ช่วยอำนวยความสะดวกมากกว่า 7-segment อย่างไร?

.....

ใบงานที่ 8 การใช้งานเซนเซอร์ LDR ควบคุมหลอด LED บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของเซนเซอร์ LDR ได้

2.นักศึกษาสามารถอ่านค่าความสว่างจาก LDR ผ่าน ESP32 ได้

3.นักศึกษาสามารถเขียนโปรแกรมควบคุม LED ตามค่าความสว่างที่อ่านได้

4.นักศึกษาสามารถแสดงผลค่าความสว่างบน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.LDR (Light Dependent Resistor)

3.ตัวต้านทาน 10kΩ (สำหรับวงจรแบ่งแรงดัน)

4.LED 1–3 ดวง + ตัวต้านทาน 220Ω

5.จอ LCD 16x2 I2C

6.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **LDR (Light Dependent Resistor)** เป็นตัวต้านทานไวแสง ความต้านทานจะเปลี่ยนไปตามความสว่างที่ตกกระทบ
- ใช้หลักการ **วงจรแบ่งแรงดัน (Voltage Divider)** เพื่อนำไปเข้าขา ADC ของ ESP32
- ESP32 มี ADC (Analog to Digital Converter) อ่านค่าแรงดันได้ช่วง 0–4095

ตัวอย่างโปรแกรม

ตัวอย่างที่ 8.1 อ่านค่า LDR และแสดงผลทาง Serial Monitor

```
#define LDR_PIN 34  // ขา ADC บน ESP32
int ldrValue = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  ldrValue = analogRead(LDR_PIN);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);
  delay(500);
}
```

ตัวอย่างที่ 8.2 ใช้ LDR ควบคุม LED (ไฟติดเมื่อมืด)

```
#define LDR_PIN 34
#define LED_PIN 2
int ldrValue = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  ldrValue = analogRead(LDR_PIN);
  Serial.print("LDR: ");
  Serial.println(ldrValue);
  if (ldrValue < 2000) { // กำหนด threshold
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  delay(500);
}
```

ตัวอย่างที่ 8.3 แสดงค่า LDR บน LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LDR_PIN 34
LiquidCrystal_I2C lcd(0x27,16,2);
int ldrValue = 0;
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("ESP32 LDR Test");
}
void loop() {
  ldrValue = analogRead(LDR_PIN);
  lcd.setCursor(0,1);
  lcd.print("LDR: ");
  lcd.print(ldrValue);
  lcd.print("    "); // ลบตัวเลขเก่า
  delay(500);
}
```

ตัวอย่างที่ 8.4 LDR ควบคุม LED หลายดวง + แสดงผลบน LCD

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LDR_PIN 34
#define LED1 16
#define LED2 17
#define LED3 18
LiquidCrystal_I2C lcd(0x27,16,2);
int ldrValue = 0;
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("LDR Control LED");
}
void loop() {
  ldrValue = analogRead(LDR_PIN);
  if (ldrValue < 1000) {       // มืดมาก
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("Dark: LED1 ON ");
  } else if (ldrValue < 2500) { // แสงปานกลาง
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("Medium: LED2ON");
  } else {                     // สว่างมาก
```

```
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    lcd.setCursor(0,1);
    lcd.print("Bright: LED3ON");
  }
  delay(500);
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 8.1 ให้แสดงค่า LDR ทุก 0.1 วินาทีแทน 0.5 วินาที

2.ปรับโปรแกรมตัวอย่างที่ 8.2 ให้ LED ติดเมื่อสว่าง และดับเมื่อมืด (กลับเงื่อนไข)

3.ปรับโปรแกรมตัวอย่างที่ 8.3 ให้แสดงค่า LDR พร้อมข้อความ “Bright” หรือ “Dark”

4.ปรับโปรแกรมตัวอย่างที่ 8.4 ให้ LED ทั้ง 3 ดวงทำงานพร้อมกันตามระดับความสว่าง (เช่น มืดมาก ⟶ LED1,2,3 ติดทั้งหมด)

คำถามท้ายใบงาน

1.หลักการทำงานของ LDR คืออะไร?

.....

2.ทำไมต้องใช้ตัวต้านทานร่วมกับ LDR?

.....

3.หากต้องการควบคุมไฟในบ้านให้เปิดอัตโนมัติเมื่อมืด ควรใช้ LDR ร่วมกับคำสั่งใดในโปรแกรม?

.....

ใบงานที่ 9 การใช้งานตัวต้านทานปรับค่าได้ (Potentiometer) ควบคุมหลอด LED บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของตัวต้านทานปรับค่าได้ (Potentiometer) ได้

2.นักศึกษาสามารถอ่านค่าจาก Potentiometer ผ่านขา ADC ของ ESP32 ได้

3.นักศึกษาสามารถนำค่าที่อ่านได้มาควบคุมความสว่างของ LED ได้

4.นักศึกษาสามารถแสดงผลค่าที่อ่านได้บน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.Potentiometer 10kΩ (ตัวต้านทานปรับค่าได้)

3.LED 1–3 ดวง + ตัวต้านทาน 220Ω

4.จอ LCD 16x2 I2C

5.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **Potentiometer** เป็นตัวต้านทานที่สามารถหมุนเพื่อปรับค่าได้
- ใช้หลักการ **แบ่งแรงดันไฟฟ้า** โดยเอาขากลาง (wiper) ต่อเข้าขา ADC ของ ESP32
- ค่า ADC ของ ESP32 อยู่ในช่วง **0 – 4095** (0V–3.3V)
- สามารถใช้ analogWrite() (PWM) ควบคุมความสว่าง LED ตามค่าที่อ่านได้

ตัวอย่างโปรแกรม

ตัวอย่างที่ 9.1 อ่านค่า Potentiometer และแสดงผลทาง Serial Monitor

```
#define POT_PIN 34  // ขา ADC
int potValue = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  potValue = analogRead(POT_PIN);
  Serial.print("Pot Value: ");
  Serial.println(potValue);
  delay(300);
}
```

ตัวอย่างที่ 9.2 ควบคุมความสว่าง LED ด้วย Potentiometer

```
#define POT_PIN 34
#define LED_PIN 2
int potValue = 0;
int brightness = 0;
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  potValue = analogRead(POT_PIN);
  brightness = map(potValue, 0, 4095, 0, 255); // แปลงค่าให้เป็น PWM
  analogWrite(LED_PIN, brightness);
  Serial.print("Brightness: ");
  Serial.println(brightness);
  delay(100);
}
```

ตัวอย่างที่ 9.3 แสดงค่าจาก Potentiometer บน LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define POT_PIN 34
LiquidCrystal_I2C lcd(0x27,16,2);
int potValue = 0;
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Potentiometer");
}
void loop() {
  potValue = analogRead(POT_PIN);
  lcd.setCursor(0,1);
  lcd.print("Value: ");
  lcd.print(potValue);
  lcd.print("    "); // ลบค่าเก่า
  delay(300);
}
```

ตัวอย่างที่ 9.4 ใช้ Potentiometer ควบคุม LED หลายดวง + LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define POT_PIN 34
#define LED1 16
#define LED2 17
#define LED3 18
LiquidCrystal_I2C lcd(0x27,16,2);
int potValue = 0;
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Pot Control LED");
}
void loop() {
  potValue = analogRead(POT_PIN);
  if (potValue < 1365) { // 0–1/3
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("LED1 ON       ");
  } else if (potValue < 2730) { // 1/3–2/3
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("LED2 ON       ");
  } else { // 2/3–max
```

```
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    lcd.setCursor(0,1);
    lcd.print("LED3 ON       ");
  }
  delay(200);
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 9.1 ให้พิมพ์ข้อความ “Low / Medium / High” ตามค่าที่อ่านได้

2.แก้ไขโปรแกรมตัวอย่างที่ 9.2 ให้ LED ดับสนิทเมื่อค่า Pot < 500

3.ปรับโปรแกรมตัวอย่างที่ 9.3 ให้แสดงค่า Pot เป็นเปอร์เซ็นต์ (%) แทนค่า ADC

4.แก้ไขโปรแกรมตัวอย่างที่ 9.4 ให้ LED ติดเพิ่มขึ้นตามลำดับ (เช่น ค่ากลาง ⟶ LED1 และ LED2 ติดพร้อมกัน)

คำถามท้ายใบงาน

1.หลักการทำงานของ Potentiometer คืออะไร?

.....

2.เพราะเหตุใดจึงต้องใช้ map() เมื่อแปลงค่าจาก ADC ไปเป็น PWM?

.....

3.ถ้าต้องการปรับความสว่างของหลอดไฟ 220V ด้วย Potentiometer และ ESP32 จำเป็นต้องใช้อุปกรณ์อะไรเพิ่มเติม?

.....

ใบงานที่ 10 การใช้งานเซนเซอร์ DHT ควบคุมหลอด LED บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของเซนเซอร์ DHT11/DHT22 ได้

2.นักศึกษาสามารถเขียนโปรแกรมอ่านค่าอุณหภูมิและความชื้นจากเซนเซอร์ DHT ได้

3.นักศึกษาสามารถนำค่าที่อ่านได้ไปควบคุมการทำงานของ LED ได้

4.นักศึกษาสามารถแสดงผลค่าที่อ่านได้ทั้งบน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.เซนเซอร์ DHT11 หรือ DHT22

3.หลอด LED 1–3 ดวง + ตัวต้านทาน 220Ω

4.จอ LCD 16x2 I2C

5.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

-DHT11/DHT22 เป็นเซนเซอร์ที่ใช้วัดอุณหภูมิ (°C) และความชื้นสัมพัทธ์ (%)

-ใช้การสื่อสารแบบ single-wire protocol กับ ESP32

-การใช้งานต้องใช้ไลบรารี DHT.h จาก Arduino IDE

-ค่าที่อ่านได้สามารถนำไปใช้ควบคุมอุปกรณ์ เช่น พัดลม หรือไฟเตือน

ตัวอย่างโปรแกรม

ตัวอย่างที่ 10.1 อ่านค่า DHT และแสดงผลทาง Serial Monitor

```
#include "DHT.h"
#define DHTPIN 4       // ขาที่ต่อ DHT
#define DHTTYPE DHT11  // หรือ DHT22
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%  Temp: ");
  Serial.print(t);
  Serial.println(" *C");
  delay(2000);
}
```

ตัวอย่างที่ 10.2 ควบคุม LED ตามค่าอุณหภูมิ

```
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT11
#define LED_PIN 2
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float t = dht.readTemperature();
  if (t > 30) {
    digitalWrite(LED_PIN, HIGH); // LED ติดเมื่อร้อนเกิน 30°C
    Serial.println("LED ON - Hot");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LED OFF - Normal");
  }
  delay(2000);
}
```

ตัวอย่างที่ 10.3 แสดงค่าอุณหภูมิและความชื้นบน LCD I2C

```
#include "DHT.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  dht.begin();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("DHT Sensor Test");
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(" C   ");
  lcd.setCursor(0,1);
  lcd.print("Hum : ");
  lcd.print(h);
  lcd.print(" %   ");
  delay(2000);
}
```

ตัวอย่างที่ 10.4 ควบคุม LED หลายดวงตามอุณหภูมิ + แสดงผลบน LCD

```
#include "DHT.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define DHTPIN 4
#define DHTTYPE DHT11
#define LED1 16
#define LED2 17
#define LED3 18
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  dht.begin();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("DHT Control LED");
}
void loop() {
  float t = dht.readTemperature();
  if (t < 25) {             // เย็น
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("Cool: LED1 ON  ");
  } else if (t < 30) {      // อุ่น
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, HIGH);
```

```
    digitalWrite(LED3, LOW);
    lcd.setCursor(0,1);
    lcd.print("Warm: LED2 ON  ");
  } else {                  // ร้อน
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, HIGH);
    lcd.setCursor(0,1);
    lcd.print("Hot: LED3 ON   ");
  }
  delay(2000);
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 10.2 ให้ LED ติดตามค่าความชื้นมากกว่า 70%

2.ปรับโปรแกรมตัวอย่างที่ 10.3 ให้ LCD แสดงข้อความ “HOT” เมื่ออุณหภูมิ > 30°C

3.แก้ไขโปรแกรมตัวอย่างที่ 10.4 ให้ LED 2 ดวงติดพร้อมกันเมื่ออุณหภูมิอยู่ระหว่าง 28–32°C

4.เพิ่ม Serial Monitor ในตัวอย่างที่ 10.4 ให้พิมพ์ข้อความเหมือนกับที่แสดงบน LCD

คำถามท้ายใบงาน

1. DHT11 และ DHT22 แตกต่างกันอย่างไร?

   .....

2. หากอ่านค่า DHT ได้ NaN (Not a Number) แสดงว่าเกิดปัญหาอะไร?

   .....

3. นอกจากการควบคุม LED แล้ว ค่าที่อ่านจาก DHT สามารถนำไปใช้ควบคุมอุปกรณ์ใดได้บ้าง?

   .....

ใบงานที่ 11 การใช้งานเซนเซอร์ Ultrasonic วัดระยะทางควบคุมหลอด LED บน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของเซนเซอร์ Ultrasonic HC-SR04 ได้

2.นักศึกษาสามารถเขียนโปรแกรม ESP32 อ่านค่าระยะทางจาก Ultrasonic ได้

3.นักศึกษาสามารถนำค่าที่วัดได้มาควบคุมการทำงานของ LED ได้

4.นักศึกษาสามารถแสดงผลค่าระยะทางทั้งบน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.Ultrasonic Sensor HC-SR04

3.หลอด LED 1–3 ดวง + ตัวต้านทาน 220Ω

4.จอ LCD 16x2 I2C

5.บอร์ดทดลอง (Breadboard) และสาย Jumper

ความรู้เบื้องต้น

- **HC-SR04** ใช้คลื่นอัลตราโซนิกในการวัดระยะทาง โดยมีขา **Trig** และ **Echo**
- หลักการคำนวณระยะทาง:

$$\text{ระยะทาง(cm)} = \text{เวลา} \times 0.034/2$$

ESP32 จะส่งสัญญาณ Trigger ⟶ รับค่ากลับมาที่ Echo ⟶ คำนวณเวลาเพื่อหาค่าระยะทาง

ตัวอย่างโปรแกรม

ตัวอย่างที่ 11.1 อ่านค่า Ultrasonic และแสดงผลทาง Serial Monitor

```
#define TRIG_PIN 5
#define ECHO_PIN 18
long duration;
int distance;
void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(500);
}
```

ตัวอย่างที่ 11.2 ควบคุม LED ตามค่าระยะทาง

```
#define TRIG_PIN 5
#define ECHO_PIN 18
#define LED_PIN 2
long duration;
int distance;
void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  if (distance < 20) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("Object Close - LED ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("Object Far - LED OFF");
  }
  delay(500);
}
```

ตัวอย่างที่ 11.3 แสดงค่าระยะทางบน LCD I2C

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define TRIG_PIN 5
#define ECHO_PIN 18
long duration;
int distance;
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  lcd.init();
  lcd.backlight();
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  lcd.setCursor(0,0);
  lcd.print("Ultrasonic Test");
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  lcd.setCursor(0,1);
  lcd.print("Dist: ");
  lcd.print(distance);
  lcd.print(" cm   ");
  delay(500);
}
```

ตัวอย่างที่ 11.4 ควบคุม LED หลายดวงตามระยะทาง + แสดงผลบน LCD

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define TRIG_PIN 5
#define ECHO_PIN 18
#define LED1 16
#define LED2 17
#define LED3 19
long duration;
int distance;
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Ultrasonic Ctrl");
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  if (distance < 10) {        // ใกล้มาก
    digitalWrite(LED1, HIGH);
```

```
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  lcd.setCursor(0,1);
  lcd.print("Very Close <10cm");
 } else if (distance < 30) {   // ระยะกลาง
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
  lcd.setCursor(0,1);
  lcd.print("Medium 10-30cm  ");
 } else {                    // ไกล
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  lcd.setCursor(0,1);
  lcd.print("Far >30cm       ");
 }
 delay(500);
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 11.1 ให้พิมพ์ระยะทางเป็น “mm” แทน “cm”

2.แก้ไขโปรแกรมตัวอย่างที่ 11.2 ให้ LED ติดเมื่อระยะทาง < 50 cm

3.ปรับโปรแกรมตัวอย่างที่ 11.3 ให้ LCD แสดงทั้งระยะทางและข้อความ “Close” หรือ “Far”

4.แก้ไขโปรแกรมตัวอย่างที่ 11.4 ให้ LED ทุกดวงกระพริบเมื่อระยะทาง < 5 cm

คำถามท้ายใบงาน

1. หลักการทำงานของเซนเซอร์ Ultrasonic HC-SR04 คืออะไร?

   .....

2. เพราะเหตุใดจึงต้องหาร 2 ในสูตรการคำนวณระยะทาง?

   .....

3. เซนเซอร์ Ultrasonic มีข้อจำกัดด้านการใช้งานอย่างไรบ้าง?

   .....

ใบงานที่ 12 การใช้งาน Servo Motor ควบคุมทิศทางและแสดงผลบน ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของ Servo Motor ได้

2.นักศึกษาสามารถเขียนโปรแกรมควบคุม Servo Motor ด้วย ESP32 ได้

3.นักศึกษาสามารถควบคุมมุมของ Servo Motor ตามค่าที่ต้องการหรือจากอินพุตต่าง ๆ ได้

4.นักศึกษาสามารถแสดงผลค่ามุมการหมุนบน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.Servo Motor (SG90 หรือ MG90S)

3.Potentiometer 10kΩ (สำหรับควบคุมมุม Servo)

4.จอ LCD 16x2 I2C

5.สาย Jumper และแหล่งจ่ายไฟ (ถ้าใช้ Servo หลายตัว)

ความรู้เบื้องต้น

- **Servo Motor** เป็นมอเตอร์ที่ควบคุมมุมหมุนได้ (ปกติ 0–180°) โดยใช้สัญญาณ **PWM**
- ESP32 สามารถสร้างสัญญาณ PWM ได้หลายช่อง ผ่านคำสั่ง ledcAttachPin() และ ledcWrite() หรือใช้ **Servo.h** ที่พอร์ตมาให้ ESP32
- การใช้งาน Servo ต้องใช้ไฟเลี้ยงที่เพียงพอ (3.3V/5V)

ตัวอย่างโปรแกรม

ตัวอย่างที่ 12.1 ควบคุม Servo หมุนไป–กลับและแสดงผลทาง Serial Monitor

```
#include <ESP32Servo.h>
Servo myServo;
void setup() {
  Serial.begin(9600);
  myServo.attach(15);  // ต่อ Servo เข้าขา GPIO15
}
void loop() {
  for (int pos = 0; pos <= 180; pos += 10) {
    myServo.write(pos);
    Serial.print("Servo Angle: ");
    Serial.println(pos);
    delay(500);
  }
  for (int pos = 180; pos >= 0; pos -= 10) {
    myServo.write(pos);
    Serial.print("Servo Angle: ");
    Serial.println(pos);
    delay(500);
  }
}
```

ตัวอย่างที่ 12.2 ใช้ Potentiometer ควบคุม Servo + แสดงผล Serial Monitor

```
#include <ESP32Servo.h>
#define POT_PIN 34
Servo myServo;
void setup() {
  Serial.begin(9600);
  myServo.attach(15);
}
void loop() {
  int potValue = analogRead(POT_PIN);
  int angle = map(potValue, 0, 4095, 0, 180);
  myServo.write(angle);
  Serial.print("Pot: ");
  Serial.print(potValue);
  Serial.print("  Servo Angle: ");
  Serial.println(angle);
  delay(100);
}
```

ตัวอย่างที่ 12.3 แสดงค่ามุม Servo บน LCD I2Cใบงานที่ 12 ESP32 Servo Motor

```
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define POT_PIN 34
Servo myServo;
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  myServo.attach(15);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Servo Control");
}
void loop() {
  int potValue = analogRead(POT_PIN);
  int angle = map(potValue, 0, 4095, 0, 180);
  myServo.write(angle);
  lcd.setCursor(0,1);
  lcd.print("Angle: ");
  lcd.print(angle);
  lcd.print("   ");
  delay(100);
}
```

ตัวอย่างที่ 12.4 ควบคุม Servo ตามปุ่มกด (ซ้าย–ขวา) + แสดงผลบน LCD

```
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define BUTTON_LEFT  14
#define BUTTON_RIGHT 27
Servo myServo;
LiquidCrystal_I2C lcd(0x27, 16, 2);
int angle = 90;
void setup() {
  myServo.attach(15);
  pinMode(BUTTON_LEFT, INPUT_PULLUP);
  pinMode(BUTTON_RIGHT, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Button Servo");
  myServo.write(angle);
}
void loop() {
  if (digitalRead(BUTTON_LEFT) == LOW) {
    angle -= 10;
    if (angle < 0) angle = 0;
    myServo.write(angle);
    delay(200);
  }
  if (digitalRead(BUTTON_RIGHT) == LOW) {
    angle += 10;
    if (angle > 180) angle = 180;
    myServo.write(angle);
    delay(200);
  }
```

```
  lcd.setCursor(0,1);
  lcd.print("Angle: ");
  lcd.print(angle);
  lcd.print("   ");
}
```

แบบฝึกหัด / กิจกรรม

1.ปรับโปรแกรมตัวอย่างที่ 12.1 ให้ Servo หมุนทีละ 1 องศาแทน 10 องศา

2.แก้ไขโปรแกรมตัวอย่างที่ 12.2 ให้ Servo หมุนในช่วง 45° – 135° เท่านั้น

3.ปรับโปรแกรมตัวอย่างที่ 12.3 ให้ LCD แสดงทั้งค่ามุม Servo และค่า ADC จาก Potentiometer

4.แก้ไขโปรแกรมตัวอย่างที่ 12.4 ให้กดปุ่มค้างแล้ว Servo หมุนต่อเนื่องจนสุดมุม

คำถามท้ายใบงาน

1. Servo Motor ต่างจาก DC Motor อย่างไร?

   .....

2. เพราะเหตุใด Servo จึงต้องใช้สัญญาณ PWM?

   .....

3. ถ้า Servo ไม่หมุนหรือสั่นผิดปกติ ควรตรวจสอบสิ่งใดบ้าง?

   .....

ใบงานที่ 13 การใช้งาน Bluetooth กับ ESP32

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของ Bluetooth Module บน ESP32 ได้

2.นักศึกษาสามารถเขียนโปรแกรมส่งและรับข้อมูลผ่าน Bluetooth ได้

3.นักศึกษาสามารถควบคุมอุปกรณ์ (เช่น LED) ผ่าน Bluetooth ได้

4.นักศึกษาสามารถแสดงผลข้อมูลที่รับ–ส่งได้ทั้งบน Serial Monitor และ LCD I2C ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.โทรศัพท์มือถือ (Android/iOS) พร้อมแอป Bluetooth Terminal

3.หลอด LED 1–3 ดวง + ตัวต้านทาน 220Ω

4.จอ LCD 16x2 I2C

5.สาย Jumper และบอร์ดทดลอง

ความรู้เบื้องต้น

- ESP32 มี **Bluetooth Classic และ BLE** ในตัว
- ไลบรารีที่นิยมใช้คือ BluetoothSerial.h
- การสื่อสารทำได้แบบ **Serial Port (UART)** แต่เป็นไร้สายผ่าน Bluetooth
- สามารถใช้แอป **Serial Bluetooth Terminal** บนมือถือเพื่อทดสอบการรับ–ส่งข้อมูล

ตัวอย่างโปรแกรม

ตัวอย่างที่ 13.1 เปิดใช้งาน Bluetooth ESP32 และส่งข้อความไปยังมือถือ

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
void setup() {
  Serial.begin(9600);
  SerialBT.begin("ESP32_BT"); // ตั้งชื่อ Bluetooth
  Serial.println("Bluetooth Started! Pair and connect.");
}
void loop() {
  SerialBT.println("Hello from ESP32!");
  delay(2000);
}
```

ตัวอย่างที่ 13.2 รับข้อความจากมือถือและแสดงบน Serial Monitor

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
void setup() {
  Serial.begin(9600);
  SerialBT.begin("ESP32_BT");
  Serial.println("Waiting for messages...");
}
void loop() {
  if (SerialBT.available()) {
    String msg = SerialBT.readString();
    Serial.print("Received: ");
    Serial.println(msg);
  }
}
```

ตัวอย่างที่ 13.3 ควบคุม LED ผ่าน Bluetooth (เปิด/ปิดด้วยข้อความ)

```
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
#define LED_PIN 2
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  SerialBT.begin("ESP32_LED");
  Serial.println("Send ON or OFF via Bluetooth");
}
void loop() {
  if (SerialBT.available()) {
    String msg = SerialBT.readString();
    msg.trim();
    if (msg == "ON") {
      digitalWrite(LED_PIN, HIGH);
      Serial.println("LED ON");
      SerialBT.println("LED is ON");
    } else if (msg == "OFF") {
      digitalWrite(LED_PIN, LOW);
      Serial.println("LED OFF");
      SerialBT.println("LED is OFF");
    }
  }
}
```

ตัวอย่างที่ 13.4 ควบคุม LED + แสดงสถานะบน LCD I2C

```
#include "BluetoothSerial.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
BluetoothSerial SerialBT;
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define LED_PIN 2
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  SerialBT.begin("ESP32_LCD");
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("BT LED Control");
}
void loop() {
  if (SerialBT.available()) {
    String msg = SerialBT.readString();
    msg.trim();
    if (msg == "ON") {
      digitalWrite(LED_PIN, HIGH);
      lcd.setCursor(0,1);
      lcd.print("LED: ON        ");
      SerialBT.println("LED is ON");
    } else if (msg == "OFF") {
      digitalWrite(LED_PIN, LOW);
      lcd.setCursor(0,1);
      lcd.print("LED: OFF       ");
      SerialBT.println("LED is OFF");
    }
  }
}
```

แบบฝึกหัด / กิจกรรม

1.แก้โปรแกรมตัวอย่างที่ 13.1 ให้ส่งข้อความ “ESP32 Ready” ทุก ๆ 1 วินาที

2.ปรับโปรแกรมตัวอย่างที่ 13.2 ให้พิมพ์ข้อความ “No Data” ถ้าไม่มีข้อมูลภายใน 5 วินาที

3.แก้ไขโปรแกรมตัวอย่างที่ 13.3 ให้รองรับคำสั่ง “BLINK” เพื่อให้ LED กระพริบ

4.ปรับโปรแกรมตัวอย่างที่ 13.4 ให้ LCD แสดงชื่อผู้ส่งข้อความด้วย (เพิ่ม Serial Monitor แสดงข้อความนั้นด้วย)

คำถามท้ายใบงาน

1. ESP32 รองรับ Bluetooth กี่แบบ และต่างกันอย่างไร?

   .....

2. หากต้องการส่งข้อมูลต่อเนื่องระหว่าง ESP32 สองตัว ควรใช้ Bluetooth Classic หรือ BLE? เพราะเหตุใด?

   .....

3. Bluetooth มีข้อจำกัดด้านระยะทางและการรบกวนสัญญาณอย่างไร?

   .....

ใบงานที่ 14 การใช้งาน ESP32 กับ Blynk

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถอธิบายหลักการทำงานของแพลตฟอร์ม Blynk ได้

2.นักศึกษาสามารถเชื่อมต่อ ESP32 กับแอป Blynk ผ่าน WiFi ได้

3.นักศึกษาสามารถเขียนโปรแกรม ESP32 เพื่อควบคุม LED ผ่านแอป Blynk ได้

4.นักศึกษาสามารถอ่านค่าจากเซนเซอร์และส่งไปแสดงผลบนแอป Blynk ได้

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.สมาร์ทโฟนที่ติดตั้งแอป **Blynk IoT** (Android/iOS)

3.หลอด LED + ตัวต้านทาน 220Ω

4.เซนเซอร์ เช่น LDR หรือ DHT11 (ใช้ประกอบการทดสอบ)

5.อินเทอร์เน็ต (WiFi Router)

ความรู้เบื้องต้น

- **Blynk** เป็นแพลตฟอร์ม IoT ที่ใช้ควบคุมและแสดงผลอุปกรณ์ IoT ผ่านมือถือหรือเว็บแดชบอร์ด
- การเชื่อมต่อกับ Blynk ต้องมี
    - **Auth Token** (จากแอป Blynk)
    - **WiFi SSID และ Password**
- ใช้ไลบรารี BlynkSimpleEsp32.h ในการเขียนโปรแกรม

ตัวอย่างโปรแกรม

ตัวอย่างที่ 14.1 การเชื่อมต่อ ESP32 กับ Blynk และส่งข้อความ

```
#define BLYNK_TEMPLATE_ID   "YourTemplateID"
#define BLYNK_DEVICE_NAME   "ESP32Test"
#define BLYNK_AUTH_TOKEN    "YourAuthToken"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";
void setup() {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  Serial.println("Connecting to Blynk...");
}
void loop() {
  Blynk.run();
}
```

ตัวอย่างที่ 14.2 ควบคุม LED ผ่านปุ่มใน Blynk App

```
#define BLYNK_TEMPLATE_ID   "YourTemplateID"
#define BLYNK_DEVICE_NAME   "ESP32_LED"
#define BLYNK_AUTH_TOKEN    "YourAuthToken"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";
#define LED_PIN 2
BLYNK_WRITE(V0) { // ปุ่มในแอปเชื่อมกับ Virtual Pin V0
  int value = param.asInt();
  digitalWrite(LED_PIN, value);
}
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}
void loop() {
  Blynk.run();
}
```

ตัวอย่างที่ 14.3 อ่านค่า LDR แล้วส่งไปแสดงใน Blynk App

```
#define BLYNK_TEMPLATE_ID   "YourTemplateID"
#define BLYNK_DEVICE_NAME   "ESP32_LDR"
#define BLYNK_AUTH_TOKEN    "YourAuthToken"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";
#define LDR_PIN 34
void setup() {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}
void loop() {
  int ldrValue = analogRead(LDR_PIN);
  Blynk.virtualWrite(V1, ldrValue); // ส่งค่าไป Virtual Pin V1
  Serial.print("LDR: ");
  Serial.println(ldrValue);
  Blynk.run();
  delay(1000);
}
```

ตัวอย่างที่ 14.4 ควบคุม LED ตามค่าจาก Blynk + แสดงค่าจาก DHT11 บนแอป

```
#define BLYNK_TEMPLATE_ID   "YourTemplateID"
#define BLYNK_DEVICE_NAME   "ESP32_DHT_LED"
#define BLYNK_AUTH_TOKEN    "YourAuthToken"
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHT.h"
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "YourWiFiSSID";
char pass[] = "YourWiFiPassword";
#define LED_PIN 2
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
BLYNK_WRITE(V0) { // ปุ่มควบคุม LED
  int value = param.asInt();
  digitalWrite(LED_PIN, value);
}
void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  dht.begin();
}
void loop() {
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  Blynk.virtualWrite(V1, t);
  Blynk.virtualWrite(V2, h);
  Blynk.run();
  delay(2000);
}
```

แบบฝึกหัด / กิจกรรม

1.แก้โปรแกรมตัวอย่างที่ 14.1 ให้พิมพ์ข้อความ "Blynk Connected" บน Serial Monitor เมื่อเชื่อมต่อสำเร็จ

2.ปรับโปรแกรมตัวอย่างที่ 14.2 ให้เพิ่มปุ่มอีกปุ่มหนึ่งควบคุม LED คนละขา (เช่น V1 ⟶ LED_PIN 16)

3.แก้โปรแกรมตัวอย่างที่ 14.3 ให้ส่งค่า LDR เป็นเปอร์เซ็นต์ (0–100%) แทนค่า ADC

4.ปรับโปรแกรมตัวอย่างที่ 14.4 ให้เพิ่มการแจ้งเตือน (Notification) เมื่ออุณหภูมิ > 30°C

คำถามท้ายใบงาน

1. Blynk ใช้การเชื่อมต่ออินเทอร์เน็ตแบบใด?

   .....

2. Virtual Pin ใน Blynk มีประโยชน์อย่างไร?

   .....

3. หาก ESP32 ไม่สามารถเชื่อมต่อกับ Blynk ได้ ควรตรวจสอบอะไรบ้าง?

   .....

ใบงานที่ 15 การประยุกต์ใช้ ESP32 (Mini Project)

วันที่.....คะแนน.....

ชื่อ.....รหัส.....ระดับชั้น.....

จุดประสงค์การเรียนรู้

1.นักศึกษาสามารถบูรณาการความรู้จากใบงานที่ผ่านมาเพื่อทำโครงงานย่อยได้

2.นักศึกษาสามารถเชื่อมต่อ Input, Output และการสื่อสาร IoT ได้อย่างเหมาะสม

3.นักศึกษาสามารถเขียนโปรแกรม ESP32 ประยุกต์ใช้ในชีวิตจริงได้

4.นักศึกษาสามารถนำเสนอผลลัพธ์การทำงานได้ทั้งในรูปแบบ Hardware และ Software

อุปกรณ์ที่ใช้

1.ESP32 DevKit Board

2.Sensor (เลือกอย่างน้อย 1 ตัว เช่น LDR, DHT11, Ultrasonic)

3.Actuator (เลือกอย่างน้อย 1 ตัว เช่น LED, Servo, Motor)

4.จอ LCD I2C หรือการแสดงผลผ่าน Blynk / Web Server

5.อุปกรณ์เสริม (ปุ่มกด, ตัวต้านทาน, สาย Jumper, Breadboard)

ภารกิจ / ตัวอย่างกิจกรรม

1.เลือกโครงงานที่สนใจ (เช่น Smart Home, Smart Door, Distance Alarm)

2.ออกแบบวงจรและเชื่อมต่ออุปกรณ์เข้ากับ ESP32

3.เขียนโปรแกรม ESP32 ให้ทำงานตามที่ออกแบบ

4.แสดงผลการทำงานทั้งทาง Serial Monitor, LCD หรือ Blynk

5.ทดลองใช้งานและบันทึกผล

คำถามท้ายใบงาน

1.อธิบายการทำงานของระบบที่เลือกทำ (Input, Processing, Output)

2.จุดเด่นของการใช้ ESP32 ในโครงงานนี้คืออะไร?

3.หากนำโครงงานนี้ไปใช้จริง ควรปรับปรุงหรือเพิ่มฟังก์ชันอะไรบ้าง?

ที่	รายการประเมิน	รายละเอียด	คะแนนเต็ม	คะแนนที่ได้
1	การออกแบบวงจร	ความถูกต้อง ครบถ้วน เหมาะสมของการต่อวงจร ESP32 กับอุปกรณ์	15	
2	การเขียนโปรแกรม	ความถูกต้องของโค้ด ความสามารถในการทำงานตามที่ออกแบบ	15	
3	การทำงานของระบบ	ระบบสามารถทำงานได้จริงตามวัตถุประสงค์	20	
4	การประยุกต์ใช้งาน	สามารถนำไปใช้ได้จริง มีความคิดสร้างสรรค์	15	
5	การนำเสนอผลงาน	การอธิบาย การสาธิต ความชัดเจนและการตอบคำถาม	10	
6	การทำงานเป็นทีม	การแบ่งหน้าที่ ความร่วมมือ และการส่งงานตรงเวลา	15	
7	เอกสาร/รายงานโครงงาน	ความครบถ้วน รูปแบบเป็นระบบ ถูกต้องตามหลักวิชาการ	10	
รวม			100	

ระดับคุณภาพ (Rubric)

- **ดีมาก (4)** : ถูกต้องครบถ้วนเกิน 80% ของเกณฑ์
- **ดี (3)** : ถูกต้องครบถ้วน 60–79% ของเกณฑ์
- **พอใช้ (2)** : ถูกต้องครบถ้วน 40–59% ของเกณฑ์
- **ควรปรับปรุง (1)** : ถูกต้องต่ำกว่า 40% ของเกณฑ์

สรุปผลการประเมิน

คะแนนรวมที่ได้..... / 100

ระดับคุณภาพ.....

ผู้ประเมิน.....

ตำแหน่ง.....

วันที่.....