เมนู

การเขียนโปรแกรมเบื้องต้น

ระดับชั้นมัธยมศึกษาปีที่ 4

ครูสมยงค์ คำเหมือดแอ่

โรงเรียนเรยีนาเชลีวิทยาลัย

สำนักงานการศึกษาเอกชน จังหวัดเชียงใหม่

สื่อการสอน

คาบที่ 1 20/05/69

ทฤษฎี :

เอกสารประกอบเนื้อหา.PDF

ปฏิบัติ : โปรแกรมรับข้อมูลจากแป้นพิมพ์

```
String inputText = "";

void setup() {
  Serial.begin(9600);
  Serial.println("กรุณาพิมพ์ข้อความ:");
}

void loop() {

  // ตรวจสอบว่ามีข้อมูลส่งเข้ามาหรือไม่
  while (Serial.available() > 0) {

    char c = Serial.read();

    // ถ้ากด Enter
    if (c == '\n') {

      Serial.print("ข้อความที่ได้รับ: ");
      Serial.println(inputText);

      // ล้างข้อความเดิม
      inputText = "";
    }
    else {
      inputText += c;
    }
  }
}
```

print(“ ข้อความ”);
แสดงข้อความออก ทางหน้าจอ

print(ชื่อตัวแปร);
แสดงค่าตัวแปร ทางหน้าจอ

printf("ข้อความ %รหัสรูปแบบข้อความ", ชื่อตัวแปร);

print(); แสดงข้อความ ไม่ขึ้นบรรทัดใหม่
println(); แสดงข้อความ และขึ้นบรรทัดใหม่

เช่น
print(“ somyong ”);
print(“ Khammueat ae ”) ผลที่ลัพธ์ที่ได้ somyong khammueat ae

หรือ
```
println(“  somyong ”);
println(“ Khammueat ae ”)
```
ผลที่ลัพธ์ที่ได้ somyong
khammueat ae

คำถาม
```
print(“ Kim”)
println(“  som ”);
print(“ yong ”)
```
ผลที่ลัพธ์ที่ได้ จะเป็นอย่างไร

คำถาม
```
println(“ Kim”)
print(“  som ”);
println(“yong ”)
```
ผลที่ลัพธ์ที่ได้ จะเป็นอย่างไร kim
som yong

คาบที่ 2 - 27/05/69

ทฤษฎี : โครงสร้างการเขียนโปรแกรม

[เอกสารประกอบ : โครงสร้างการเขียนโปรแกรม.PDF]

ปฏิบัติ : ควบคุม LED ด้วยข้อความ

ตัวอย่างนี้เป็นการควบคุม LED ด้วยข้อความที่พิมพ์จากคอมพิวเตอร์ผ่าน Serial Monitor

- พิมพ์ ON → LED ติด
- พิมพ์ OFF → LED ดับ

เหมาะกับบอร์ด Arduino Uno / Nano / Mega

ต่อวงจร

- LED ขา + → Pin 13
- LED ขา - → GND
  (หรือใช้ LED ในตัวบอร์ดได้เลย)

```
String command = "";

void setup() {

 pinMode(13, OUTPUT);

 Serial.begin(9600);

 Serial.println("พิมพ์ ON หรือ OFF");
}

void loop() {

 while (Serial.available() > 0) {

   char c = Serial.read();

   // เมื่อกด Enter
   if (c == '\n') {

     command.trim(); // ลบช่องว่างและ Enter

     // ตรวจสอบข้อความ
     if (command == "ON") {

       digitalWrite(13, HIGH);
       Serial.println("LED ON");

     }
     else if (command == "OFF") {
```

```
      digitalWrite(13, LOW);
      Serial.println("LED OFF");

    }
    else {

      Serial.println("คำสั่งไม่ถูกต้อง");

    }

    // ล้างข้อความ
    command = "";
  }
  else {

    command += c;

  }
}
}
```

วิธีใช้งาน

1. อัปโหลดโค้ด
2. เปิด Serial Monitor
3. ตั้ง Baud Rate = `9600`
4. ตั้งค่า `Newline`
5. พิมพ์
    - `ON`
    - `OFF`

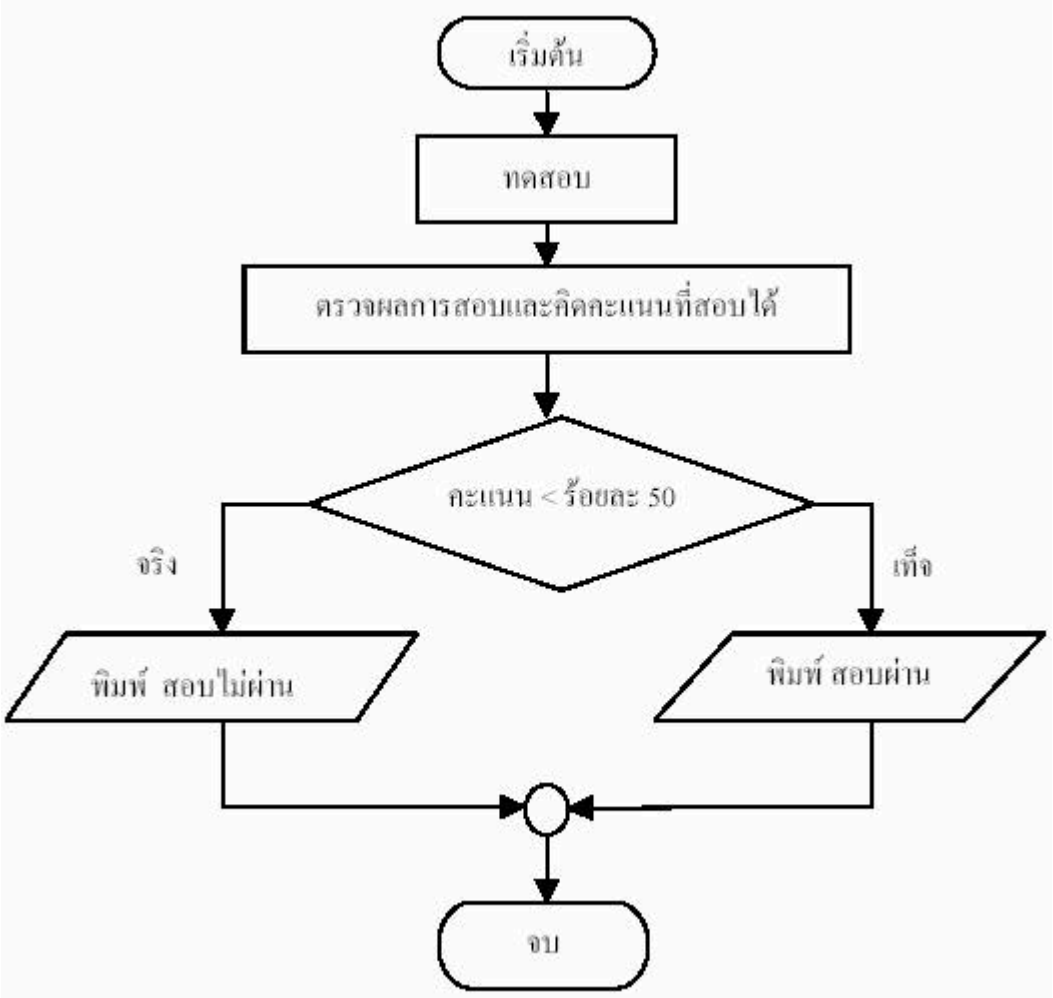
ตัวอย่างผลลัพธ์

ON
LED ON

OFF
LED OFF

คาบที่ 3 3/06/69 วันหยุด

คาบที่ 4 10/06/69

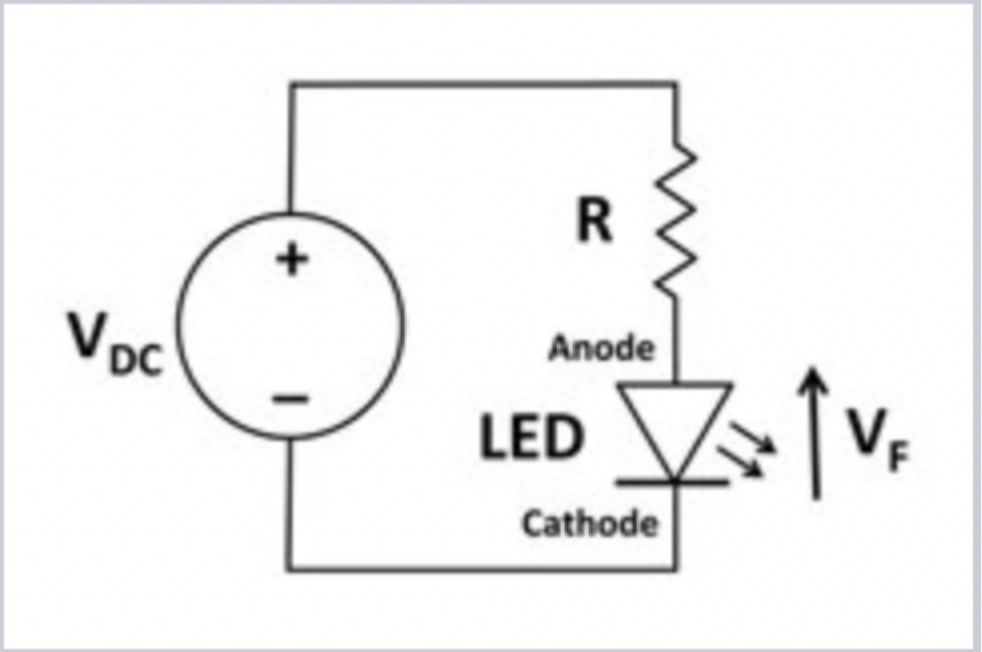


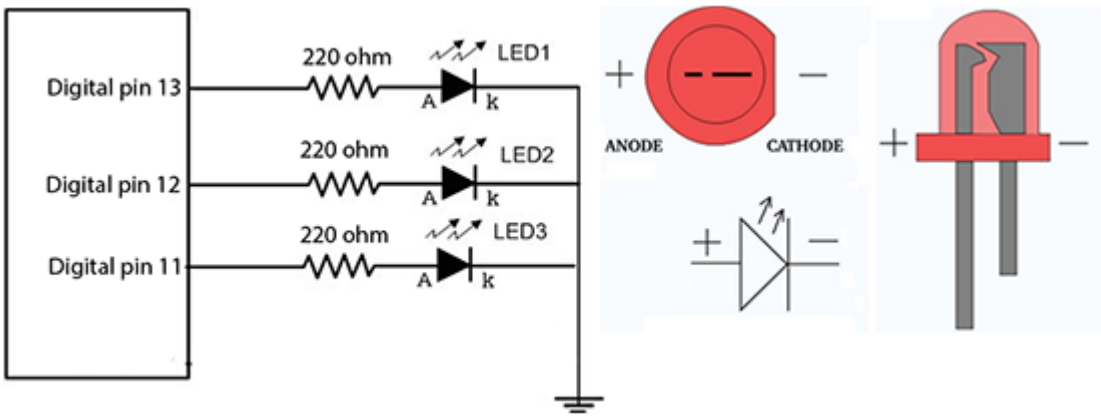
ป.ควบคุม LED หลายดวง

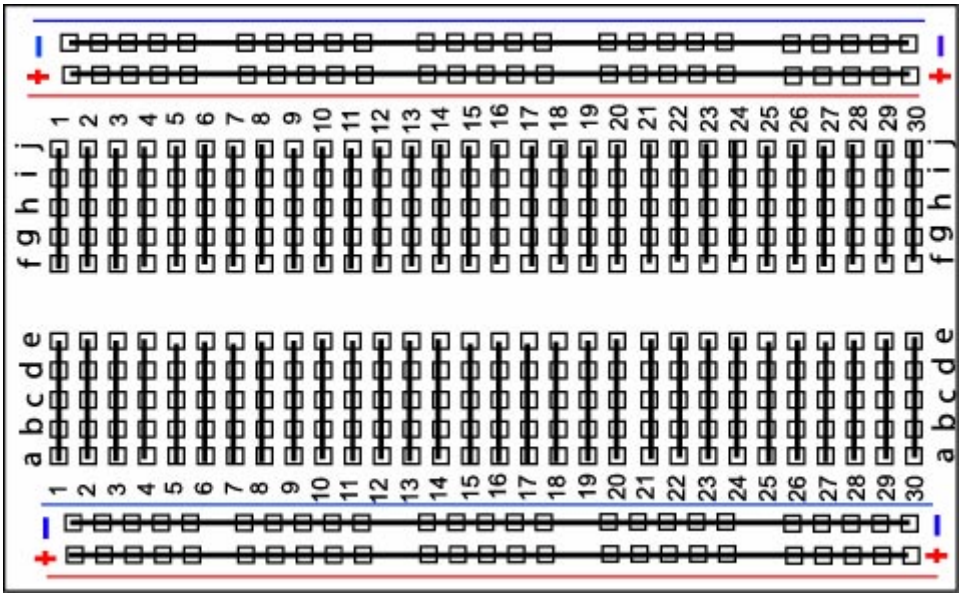
ตัวอย่างนี้เป็นการควบคุม LED หลายดวงผ่าน Serial Monitor
คำสั่งตัวอย่าง

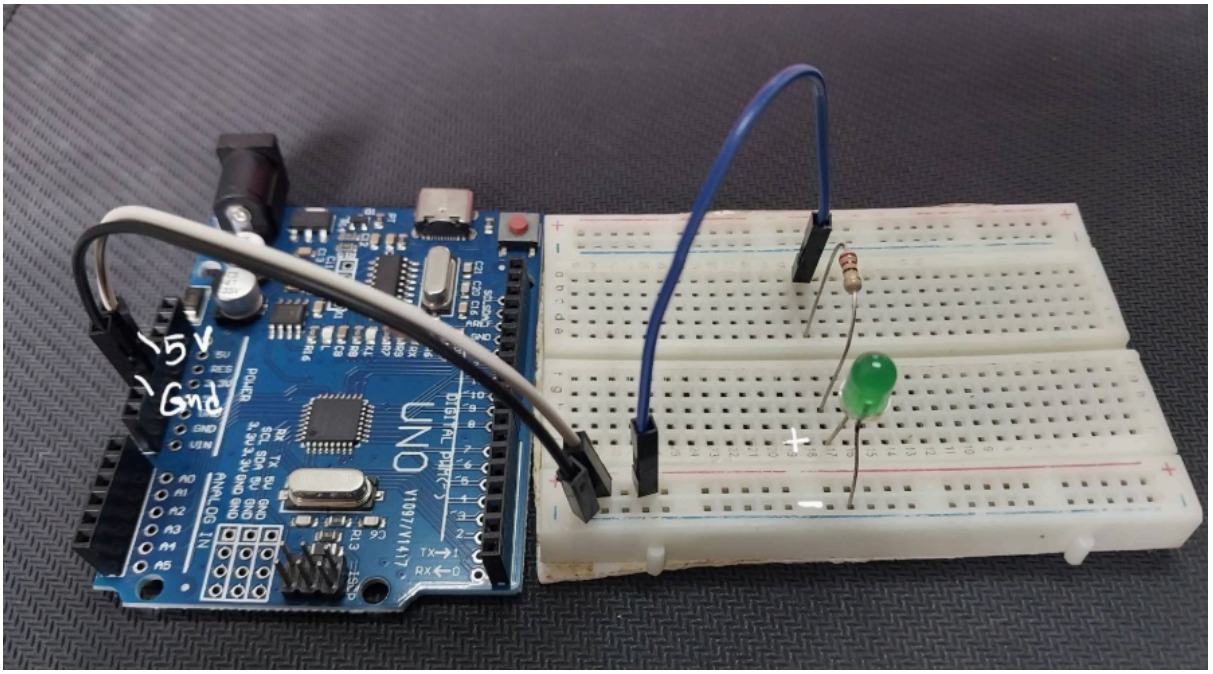
- LED1 ON
- LED1 OFF
- LED2 ON
- LED2 OFF
- ALL ON
- ALL OFF

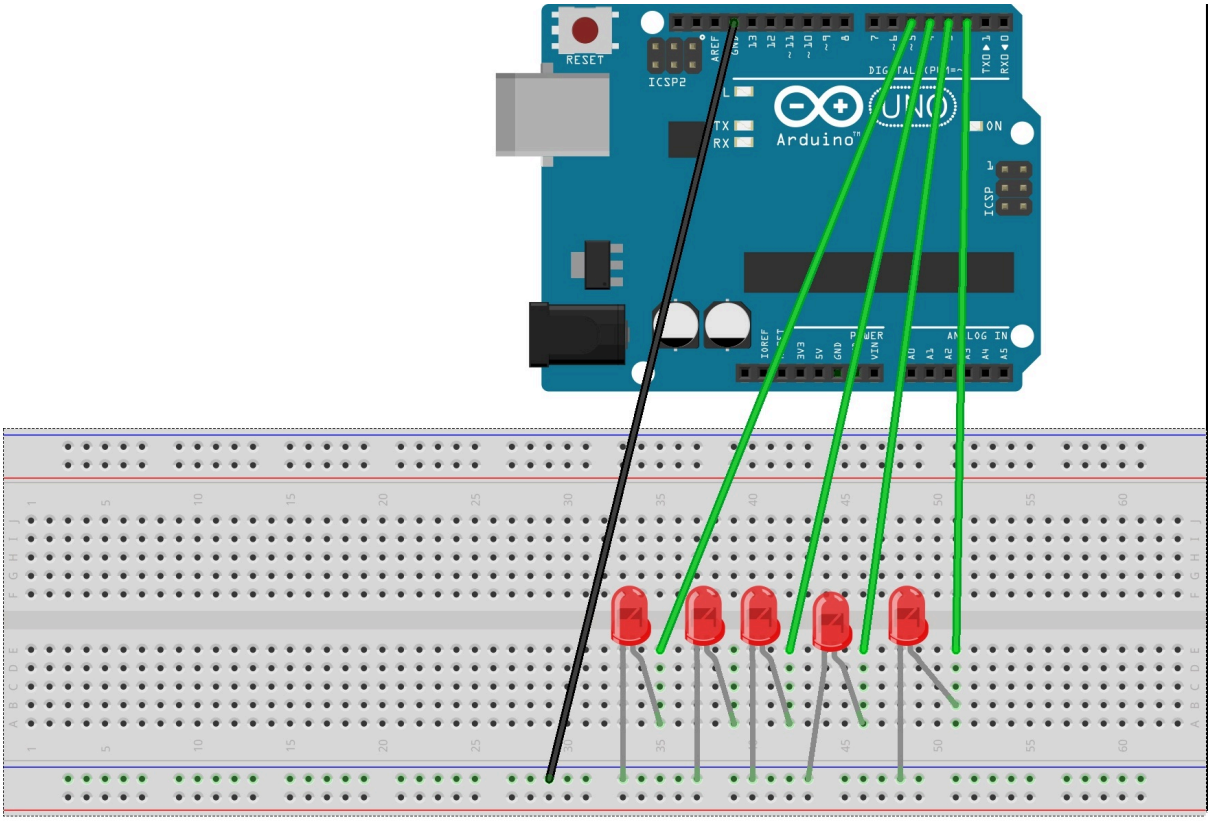
ผังการต่อวงจร






a b c d e
f g h i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30


5V
Gnd
UNO
DIGITAL PWM~
POWER
ANALOG IN


RESET
ICSP2
AREF
L
TX
RX
Arduino
UNO
ON
ICSP
IOREF
5V
GND
VIN
A0
A1
A2
A3
A4
A5

การต่อวงจร

- LED1 → ขา 8
- LED2 → ขา 9
- LED3 → ขา 10

(ต่อผ่านตัวต้านทาน 220Ω ไป GND)

```
String command = "";

// กำหนดขา LED
int led1 = 8;
int led2 = 9;
int led3 = 10;

void setup() {

  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);

  Serial.begin(9600);

  Serial.println("ระบบควบคุม LED");
  Serial.println("ตัวอย่างคำสั่ง:");
  Serial.println("LED1 ON");
  Serial.println("LED1 OFF");
  Serial.println("ALL ON");
  Serial.println("ALL OFF");
}

void loop() {

  while (Serial.available() > 0) {

    char c = Serial.read();

    // เมื่อกด Enter
    if (c == '\n') {

      command.trim();

      // ===== LED1 =====
      if (command == "LED1 ON") {
        digitalWrite(led1, HIGH);
        Serial.println("LED1 ON");
      }

      else if (command == "LED1 OFF") {
        digitalWrite(led1, LOW);
        Serial.println("LED1 OFF");
      }

      // ===== LED2 =====
      else if (command == "LED2 ON") {
        digitalWrite(led2, HIGH);
        Serial.println("LED2 ON");
      }

      else if (command == "LED2 OFF") {
        digitalWrite(led2, LOW);
        Serial.println("LED2 OFF");
      }

      // ===== LED3 =====
      else if (command == "LED3 ON") {
        digitalWrite(led3, HIGH);
        Serial.println("LED3 ON");
      }

      else if (command == "LED3 OFF") {
        digitalWrite(led3, LOW);
        Serial.println("LED3 OFF");
      }
```

```
      // ===== ALL =====
      else if (command == "ALL ON") {

        digitalWrite(led1, HIGH);
        digitalWrite(led2, HIGH);
        digitalWrite(led3, HIGH);

        Serial.println("ALL LED ON");
      }

      else if (command == "ALL OFF") {

        digitalWrite(led1, LOW);
        digitalWrite(led2, LOW);
        digitalWrite(led3, LOW);

        Serial.println("ALL LED OFF");
      }

      else {
        Serial.println("ไม่พบคำสั่ง");
      }

      // ล้างข้อความ
      command = "";
    }

    else {
      command += c;
    }
  }
}
```

ตัวอย่างการพิมพ์ใน Serial Monitor

LED1 ON
LED2 ON
LED3 OFF
ALL OFF

คาบที่ 5 17/06/69

(ติดกิจกรรมงานไหว้ครู)

คาบที่ 6 24/06/69

ตัวแปร และชนิดของตัวแปร

1. ตัวแปรคืออะไร

เช่น

int age = 20;

- `int` คือ ชนิดข้อมูล
- `age` คือ ชื่อตัวแปร
- `20` คือ ค่าที่เก็บอยู่ในตัวแปร

เปรียบเทียบง่าย ๆ

กล่อง	ชื่อตัวแปร	ข้อมูล
📦	age	20
📦	score	95
📦	name	Somyong

---

2. ชนิดของตัวแปรที่ใช้บ่อยใน Arduino

2.1 int (จำนวนเต็ม)

เก็บเลขจำนวนเต็ม

int led1 = 8;

จากโค้ดที่เราคุยกัน

int led1 = 8;
int led2 = 9;
int led3 = 10;

หมายถึง

ตัวแปร	ค่า
led1	8
led2	9
led3	10

ใช้เก็บหมายเลขขา Arduino

2.2 float (เลขทศนิยม)

เช่น

float temperature = 36.5;

ใช้กับ

- อุณหภูมิ
- ความชื้น
- แรงดันไฟฟ้า

ตัวอย่าง

float voltage = 4.85;

2.3 char (ตัวอักษร 1 ตัว)

เก็บได้เพียงตัวเดียว

char grade = 'A';

หรือ

char c = Serial.read();

จากโค้ดเดิม

char c = Serial.read();

เมื่อผู้ใช้พิมพ์

LED1 ON

Arduino จะอ่านทีละตัว

L
E

D
1

O
N

2.4 String (ข้อความ)

เก็บข้อความหลายตัวอักษร

จากโค้ดเดิม

String command = "";

ตัวอย่าง

command = "LED1 ON";

หรือ

String name = "Somyong";

2.5 boolean หรือ bool

เก็บค่า

true
false

ตัวอย่าง

bool ledStatus = true;

แปลว่า

LED เปิดอยู่

---

3. การกำหนดค่าให้ตัวแปร

int score = 80;

เปลี่ยนค่าได้

score = 90;

ตอนนี้

score = 90

---

4. การใช้ตัวแปรในคำสั่ง

ตัวอย่าง

int led1 = 8;

digitalWrite(led1, HIGH);

Arduino จะมองเป็น

digitalWrite(8, HIGH);

ข้อดีคือ

หากย้าย LED ไปขา 12

แก้แค่

int led1 = 12;

ทั้งโปรแกรมจะเปลี่ยนตาม

---

5. ตัวแปรกับการรับข้อมูลจากคอมพิวเตอร์

จากโค้ดเดิม

String command = "";

เมื่อผู้ใช้พิมพ์

LED1 ON

ข้อมูลจะถูกเก็บในตัวแปร

command

ดังนี้

command = "LED1 ON"

แล้วนำไปตรวจสอบ

```
if(command == "LED1 ON")
{
  digitalWrite(led1,HIGH);
}
```

---

6. ตัวอย่างการใช้หลายชนิดร่วมกัน

```
int ledPin = 8;
bool ledStatus = false;
float voltage = 4.98;
String command = "LED ON";
char c = 'A';
```

ตัวแปร	ชนิด	ค่า
ledPin	int	8
ledStatus	bool	false
voltage	float	4.98
command	String	LED ON
c	char	A

---

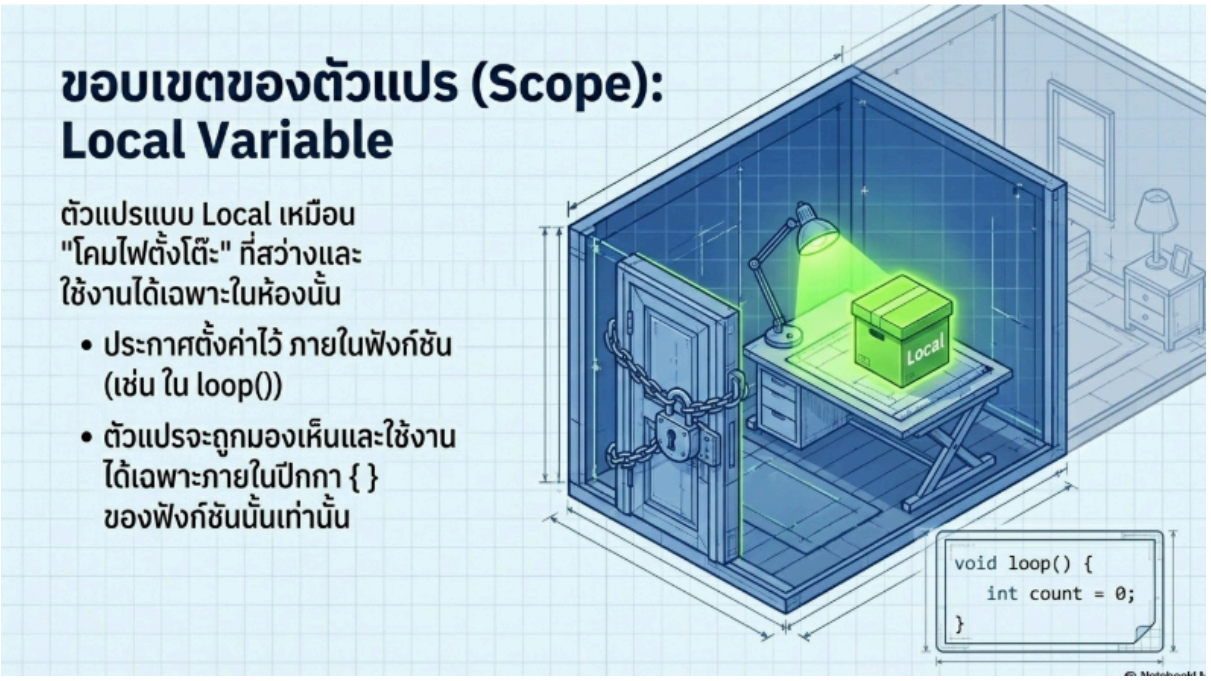
7. ตัวแปร Local และ Global

Global Variable

ประกาศนอกฟังก์ชัน

```
int led1 = 8;

void setup() {
}

void loop() {
}
```

ใช้งานได้ทั้งโปรแกรม



Local Variable

ประกาศภายในฟังก์ชัน

```
void loop() {

 int count = 0;

}
```

ตัวแปร `count` ใช้ได้เฉพาะใน `loop()`

---

8. ตัวอย่างจากโปรเจกต์ตรวจจับบุหรี่ไฟฟ้า

หากคุณทำโปรเจกต์เซนเซอร์ตรวจจับควัน

```
int smokeValue = 0;
int threshold = 500;
bool alarmStatus = false;
```

ความหมาย

ตัวแปร	ใช้เก็บ
smokeValue	ค่าควันจากเซนเซอร์
threshold	ค่าเกณฑ์แจ้งเตือน
alarmStatus	สถานะการเตือน

ตัวอย่าง

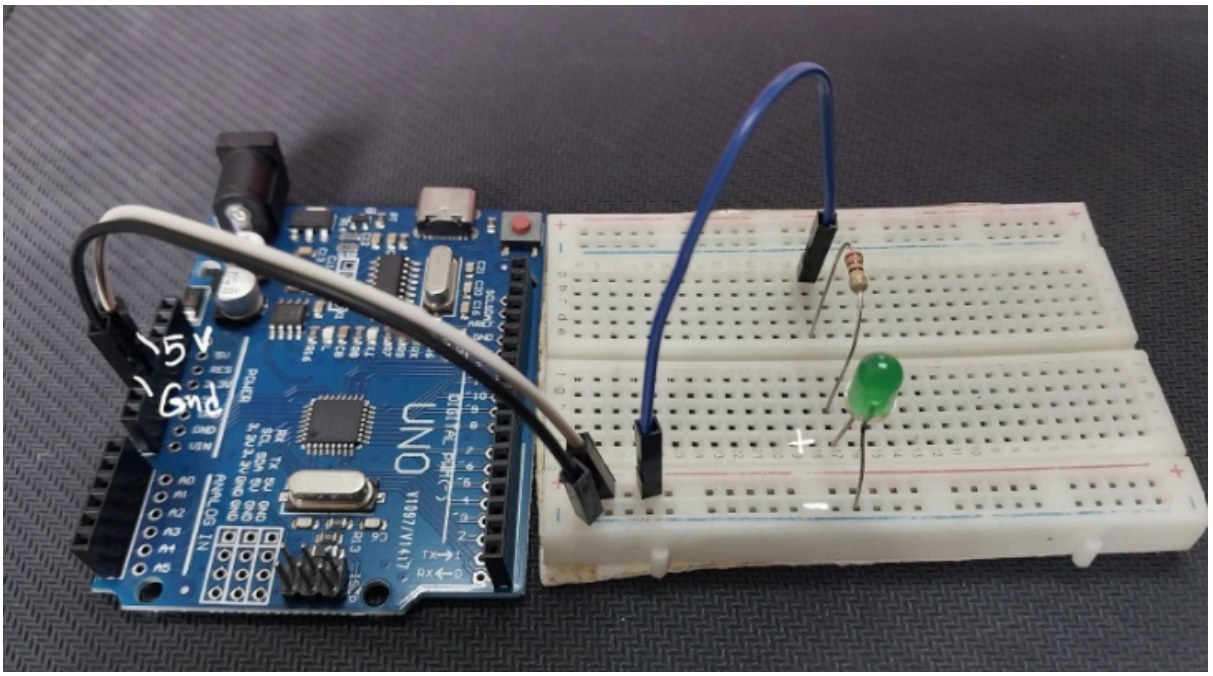
```
if(smokeValue > threshold)
{
  alarmStatus = true;
}
```

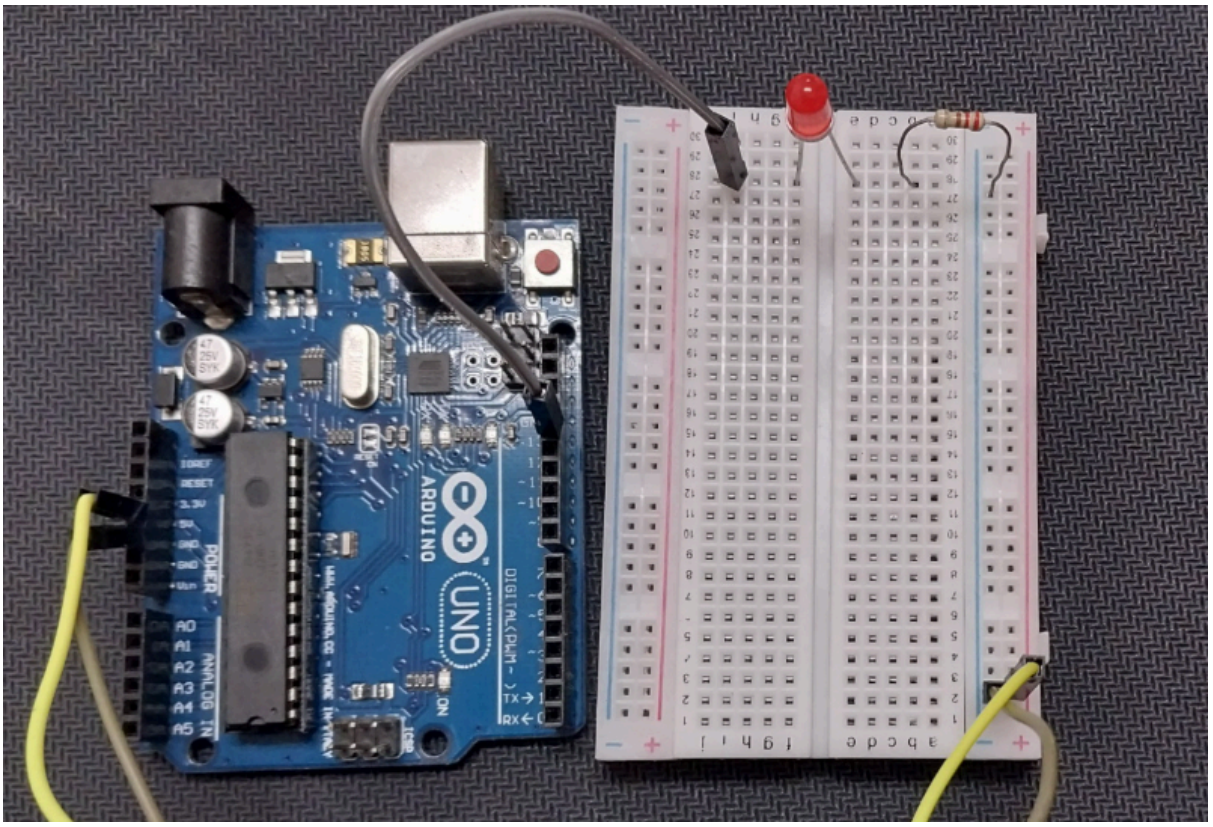
สรุปชนิดตัวแปรที่ควรรู้สำหรับ Arduino

ชนิด	ตัวอย่าง	ใช้เก็บ
int	10	จำนวนเต็ม
float	3.14	ทศนิยม
char	'A'	ตัวอักษร 1 ตัว
String	"HELLO"	ข้อความ
bool	true	จริง/เท็จ

แบบทดสอบความรู้เรื่องการเขียนโปรแกรม (ครั้งที่ 1 20 คะแนน)

ปฏิบัติ





ตัวอย่างโค้ด Arduino ให้ LED 4 ดวง ติดพร้อมกัน 2 วินาที แล้วดับพร้อมกัน 2 วินาที วนซ้ำไปเรื่อย ๆ
การต่อวงจร

LED1 → ขา 8
LED2 → ขา 9
LED3 → ขา 10
LED4 → ขา 11

(ต่อผ่านตัวต้านทาน 220Ω)

```
// กำหนดขา LED
int led1 = 8;
int led2 = 9;
int led3 = 10;
int led4 = 11;

void setup() {

  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);

}

void loop() {

  // เปิด LED ทุกดวง
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);

  delay(2000);  // 2 วินาที

  // ปิด LED ทุกดวง
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);

  delay(2000);  // 2 วินาที
}
```

<u>ตัวอย่างนี้เป็นการควบคุม LED ด้วยข้อความที่พิมพ์จากคอมพิวเตอร์ผ่าน Serial Monitor</u>

โจทย์ : ให้นักเรียนศึกษา ตัวอย่างโปรแกรม 2 โปรแกรมนี้

แล้วสร้างโปรแกรมใหม่ขึ้นมา โดย

ให้ กดแป้น พิมพ์ 0 1 2 3 และ 4

ถ้ากด 0 ให้ LED ทุกดวง ดับหมด

ถ้ากด 1 ให้ LED ดวงที่ 1 ติด ดวงอื่นๆ ดับหมด

ถ้ากด 2 ให้ LED ดวงที่ 2 ติด ดวงอื่นๆ ดับหมด

ถ้ากด 3 ให้ LED ดวงที่ 3 ติด ดวงอื่นๆ ดับหมด

ถ้ากด 4 ให้ LED ดวงที่ 4 ติด ดวงอื่นๆ ดับหมด

คาบที่ 7 1/07/69

คำสั่งใน arduion

1. โครงสร้างหลักของโปรแกรม (Program Structure)
    - void setup(): ฟังก์ชันสำหรับประกาศค่าเริ่มต้น เช่น โหมดของขาใช้งาน หรือการเริ่มสื่อสาร โดยจะทำงานเพียงครั้งเดียวเมื่อเปิดบอร์ด
    - void loop(): ฟังก์ชันหลักที่บรรจุคำสั่งควบคุม โดยจะทำงานวนซ้ำต่อเนื่องไปเรื่อย ๆ

2. คำสั่งควบคุมขาใช้งาน (I/O Control)
    - pinMode(pin, mode): กำหนดหน้าที่ของขา เช่น INPUT เพื่อรับสัญญาณ หรือ OUTPUT เพื่อส่งสัญญาณ
    - digitalWrite(pin, value): ส่งสัญญาณดิจิทัลออกไปเป็น HIGH (เปิด/5V) หรือ LOW (ปิด/0V)
    - digitalRead(pin): อ่านค่าลอจิกดิจิทัลจากขาที่กำหนด (ได้ผลเป็น HIGH หรือ LOW)
    - analogRead(pin): อ่านค่าแรงดันแอนะล็อกจากขา A0-A5 แล้วแปลงเป็นตัวเลขช่วง 0 ถึง 1023
    - analogWrite(pin, value): ส่งสัญญาณ PWM ออกไปเพื่อจำลองแรงดันแอนะล็อก (ใช้ค่า 0 ถึง 255)

3. การสื่อสารผ่านพอร์ตซีเรียล (Serial Communication)
    - Serial.begin(rate): เริ่มต้นการสื่อสารที่ความเร็วที่กำหนด เช่น 9600
    - Serial.print(data) / Serial.println(data): ส่งข้อมูลไปแสดงที่ Serial Monitor (println จะขึ้นบรรทัดใหม่)
    - Serial.available(): ตรวจสอบจำนวนข้อมูลที่ถูกส่งมาจากคอมพิวเตอร์ว่ามีหรือไม่
    - Serial.read(): อ่านค่าตัวอักษรหรือข้อมูลที่รับมาจากคอมพิวเตอร์

4. คำสั่งควบคุมเวลา (Time)
    - delay(ms): หยุดรอการทำงานชั่วคราวตามเวลาที่ระบุในหน่วยมิลลิวินาที
    - delayMicroseconds(us): หยุดรอในระดับไมโครวินาที
    - millis(): ส่งคืนค่าเวลาเป็นมิลลิวินาทีนับตั้งแต่เริ่มเปิดบอร์ด

5. โครงสร้างการควบคุมและตรรกะ (Control & Logic)
    - if...else: ตรวจสอบเงื่อนไขเพื่อเลือกทิศทางการทำงาน
    - for: สั่งให้วนรอบทำซ้ำตามจำนวนครั้งที่ระบุ
    - while: วนรอบทำงานซ้ำตราบเท่าที่เงื่อนไขยังเป็นจริง
    - do...while: ทำงานก่อนหนึ่งรอบแล้วค่อยตรวจสอบเงื่อนไขเพื่อวนซ้ำ

แบบทดสอบ คำสั่งโปรแกรม Arduino

การเขียนโปรแกรม

โจทย์ : ให้เขียนโปรแกรม รับข้อมูลจากแป้นพิมพ์เพื่อกำหนดการติด ของ LED จำนวน 4 ดวง

เงื่อนไข กด 1 LED ดวงที่ 1 ติด ดวงอื่นๆ ดับหมด
กด 2 LED ดวงที่ 2 ติด ดวงอื่นๆ ดับหมด
กด 3 LED ดวงที่ 3 ติด ดวงอื่นๆ ดับหมด
กด 4 LED ดวงที่ 4 ติด ดวงอื่นๆ ดับหมด

(ขั้นที่ 1)

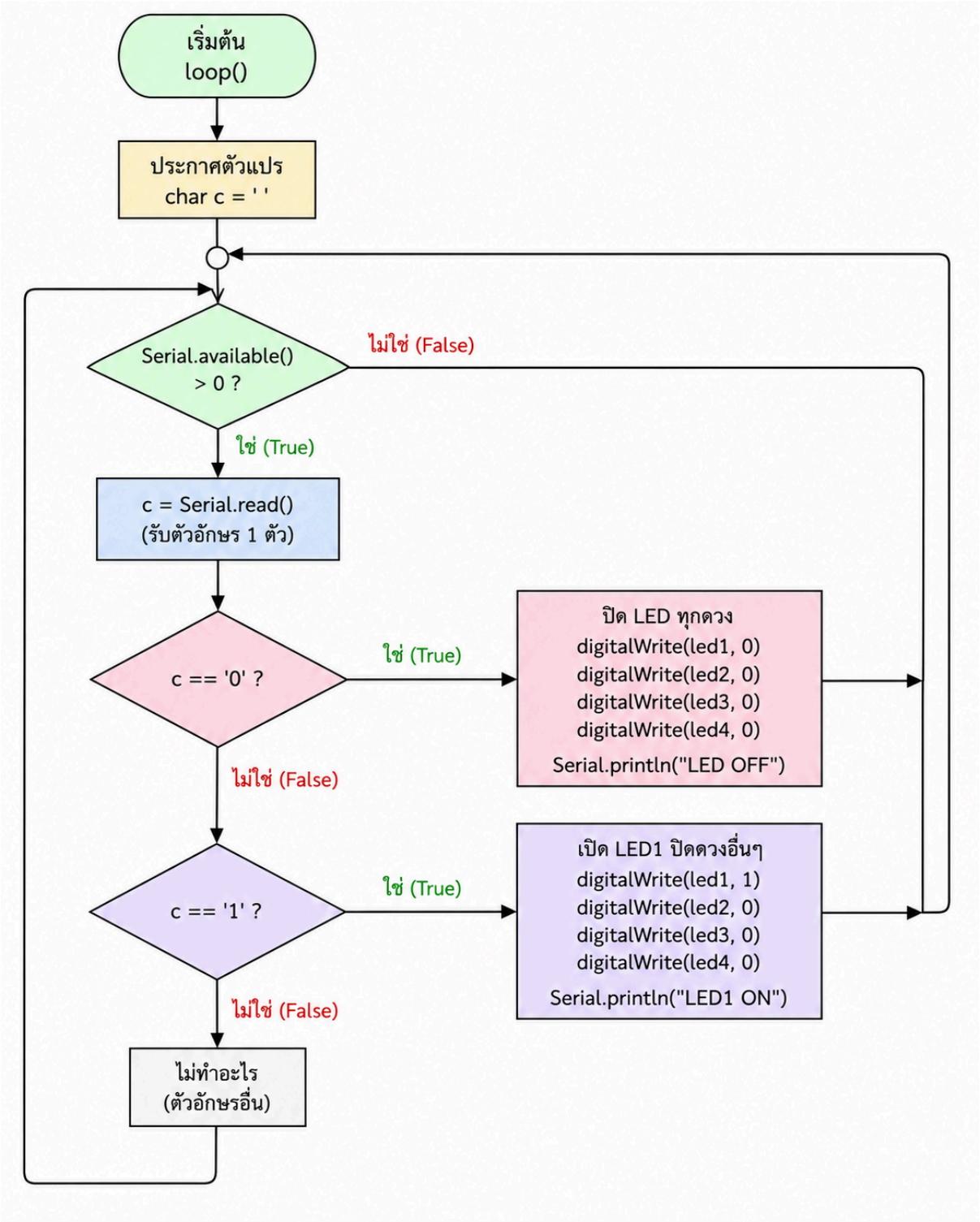
```
void loop() {

    //รับตัวอักษรจากแป้นพิมพ์
    while (Serial.available() > 0) {

        char c = Serial.read();
    } // end while


} // end void loop
```

(ขั้นที่ 2)

```
void loop() {

        //รับตัวอักษรจากแป้นพิมพ์
        while (Serial.available() > 0) {

                char c = Serial.read();

        // ตรวจสอบข้อความ 0
        if (c == "0") {

                // ปิด LED ทุกดวง
                digitalWrite(led1, 0);
                digitalWrite(led2, 0);
                digitalWrite(led3, 0);
                digitalWrite(led4, 0);

                Serial.println("LED OFF");

        } // end if 0

        // ตรวจสอบข้อความ 1
        if (c == "1") {

                // ปิด LED ทุกดวง
                digitalWrite(led1, 1);
                digitalWrite(led2, 0);
                digitalWrite(led3, 0);
                digitalWrite(led4, 0);

                Serial.println("LED1 ON");

        } // end if 1

    } // end while
} // end void loop
```

📌 โครงสร้าง Flowchart การทำงาน

- **(จุดเริ่มต้น)** เริ่มต้นรอบการทำงานของ `loop()`
    - **[เงื่อนไขที่ 1]** มีข้อมูลเข้ามาทาง Serial Port หรือไม่? (`Serial.available() > 0`)
        - ➡️ **มี (Yes):** อ่านค่าตัวอักษรมาเก็บไว้ในตัวแปร `c`
        - ➡️ **ไม่มี (No):** ข้ามการอ่านค่าไป
    - **[เงื่อนไขที่ 2]** ค่าในตัวแปร `c` เท่ากับ '0' ใช่หรือไม่?
        - ➡️ **ใช่ (Yes):**
            - สั่ง **ปิด** ไฟ LED ทั้ง 4 ดวง (`led1` ถึง `led4` = 0)
            - แสดงข้อความ "LED OFF" ออกทาง Serial Monitor
        - ➡️ **ไม่ใช่ (No):** ข้ามไปตรวจสอบเงื่อนไขถัดไป
    - **[เงื่อนไขที่ 3]** ค่าในตัวแปร `c` เท่ากับ '1' ใช่หรือไม่?
        - ➡️ **ใช่ (Yes):**
            - สั่ง **เปิด** ไฟ `led1` (= 1)
            - สั่ง **ปิด** ไฟ `led2`, `led3`, `led4` (= 0)
            - แสดงข้อความ "LED1 ON" ออกทาง Serial Monitor
        - ➡️ **ไม่ใช่ (No):** ข้ามการทำงานในส่วนนี้ไป
- **(จุดสิ้นสุด)** จบการทำงานรอบปัจจุบัน และวนกลับไปเริ่มต้นใหม่ที่ **(จุดเริ่มต้น)**


เริ่มต้น
loop()
ประกาศตัวแปร
char c = ' '
Serial.available()
> 0 ?
ไม่ใช่ (False)
ใช่ (True)
c = Serial.read()
(รับตัวอักษร 1 ตัว)
c == '0' ?
ใช่ (True)
ปิด LED ทุกดวง
digitalWrite(led1, 0)
digitalWrite(led2, 0)
digitalWrite(led3, 0)
digitalWrite(led4, 0)
Serial.println("LED OFF")
ไม่ใช่ (False)
c == '1' ?
ใช่ (True)
เปิด LED1 ปิดดวงอื่นๆ
digitalWrite(led1, 1)
digitalWrite(led2, 0)
digitalWrite(led3, 0)
digitalWrite(led4, 0)
Serial.println("LED1 ON")
ไม่ใช่ (False)
ไม่ทำอะไร
(ตัวอักษรอื่น)

เขียนโปรแกรมแบบ switch case

```
// กำหนดขา LED
int led1 = 8;
int led2 = 9;
int led3 = 10;

void setup() {

  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);

  Serial.begin(9600);

  Serial.println("ระบบควบคุม LED");
   }


void loop() {

// รับตัวอักษรจากแป้นพิมพ์
while (Serial.available() > 0) {

char c = Serial.read();

switch (c) {

// ====== 0 ======
case '0':

digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);

Serial.println("LED OFF");
break;

// ====== 1 ======
case '1':

digitalWrite(led1, HIGH);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);

Serial.println("LED1 ON");
break;

// ====== 2 ======
case '2':

digitalWrite(led1, LOW);
digitalWrite(led2, HIGH);
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
```

```
Serial.println("LED2 ON");
break;

// ====== 3 ======
case '3':

digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, HIGH);
digitalWrite(led4, LOW);

Serial.println("LED3 ON");
break;

// ====== 4 ======
case '4':

digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);
digitalWrite(led4, HIGH);

Serial.println("LED4 ON");
break;

// ====== Default ======
default:
Serial.println("Unknown Command");
break;
}

}

}
```

ข้อดีของ switch...case

เมื่อมีคำสั่งจำนวนมาก เช่น

0 = ปิดทุกดวง
1 = LED1
2 = LED2
3 = LED3
4 = LED4
5 = เปิดทุกดวง
6 = วิ่งซ้าย
7 = วิ่งขวา
8 = กระพริบ
9 = ไฟกระพริบสลับ
A = โหมด Auto
B = โหมด Manual

การใช้ `switch...case` จะทำให้อ่านง่ายและจัดการคำสั่งได้เป็นระเบียบกว่าการใช้ `if...else if` จำนวนมาก จึงเป็นรูปแบบที่นิยมใช้ในการเขียนโปรแกรมควบคุมอุปกรณ์ด้วย Arduino

โครงสร้างแบบวนซ้ำ

โครงสร้างของคำสั่ง `for`

คำสั่ง `for` ใน Arduino (ซึ่งใช้ภาษา C/C++) เป็นคำสั่งสำหรับ **การวนซ้ำ (Loop)** เมื่อเราต้องการให้โปรแกรมทำงานเดิมซ้ำหลายครั้ง โดยไม่ต้องเขียนโค้ดซ้ำหลายบรรทัด

โครงสร้างของคำสั่ง `for`

```
for (ค่าเริ่มต้น; เงื่อนไข; การเปลี่ยนแปลงค่า) {

  // คำสั่งที่ต้องการให้ทำซ้ำ

} // end for
```

มีทั้งหมด 3 ส่วน

for (1; 2; 3)

1. **ค่าเริ่มต้น (Initialization)**
   - กำหนดค่าตัวแปรครั้งแรก
   - ทำงานเพียงครั้งเดียว
2. **เงื่อนไข (Condition)**
   - ตรวจสอบว่าจะวนต่อหรือไม่
   - ถ้าเป็น `true` จะทำงานต่อ
   - ถ้าเป็น `false` จะออกจากลูป
3. **การเปลี่ยนแปลงค่า (Increment/Decrement)**
   - เพิ่มหรือลดค่าตัวแปรทุกครั้งหลังทำงานในลูป

ตัวอย่างที่ 1 นับเลข 0 ถึง 4

```
void setup() {
 Serial.begin(9600);

 for (int i = 0; i <=4; i++) {
   Serial.print(i);
  Serial.print(“ ”);
 }
}

void loop() {
}
```

ผลลัพธ์

```
*****************************************
0
1
2
3
4
```

อธิบายการทำงานทีละรอบ

รอบ	i ก่อนตรวจ	เงื่อนไข `i<5`	แสดงผล	i หลังจบรอบ
1	0	จริง	0	1
2	1	จริง	1	2
3	2	จริง	2	3
4	3	จริง	3	4
5	4	จริง	4	5
6	5	เท็จ	หยุด	-

ตัวอย่างที่ 2 เปิด LED ทีละดวง

```
int led[] = {2,3,4,5};

void setup() {

 for(int i=0;i<4;i++){
    pinMode(led[i],OUTPUT);
 }

}

void loop() {

 for(int i=0;i<4;i++){

    digitalWrite(led[i],HIGH);
    delay(300);
    digitalWrite(led[i],LOW);

 }

}
```

การทำงาน

i=0 → LED ขา 2

i=1 → LED ขา 3

i=2 → LED ขา 4

i=3 → LED ขา 5

ตัวอย่างที่ 3 นับถอยหลัง

```
for(int i=10; i>0; i--){
  Serial.println(i);
}
```

ผลลัพธ์

```
10
9
8
7
6
5
4
3
2
1
```

สังเกตว่า

```
i--
```

คือ

```
i = i - 1;
```

ตัวอย่างที่ 4 เพิ่มครั้งละ 2

```
for(int i=0; i<=10; i+=2){
  Serial.println(i);
}
```

ผลลัพธ์

```
0
2
4
6
8
10
```

เพราะ

```
i += 2
```

หมายถึง

```
i = i + 2;
```

ตัวอย่างที่ 5 วนซ้อนกัน (Nested for)

```
for(int i=1;i<=3;i++){

  for(int j=1;j<=3;j++){

      Serial.print("(");
      Serial.print(i);
      Serial.print(",");
      Serial.print(j);
      Serial.println(")");

  }

}
```

ผลลัพธ์

```
(1,1)
(1,2)
(1,3)

(2,1)
(2,2)
(2,3)

(3,1)
(3,2)
(3,3)
```

ลูปด้านใน (`j`) จะทำงานครบทุกครั้ง ก่อนที่ลูปด้านนอก (`i`) จะเปลี่ยนค่า

เปรียบเทียบ `for` กับ `while`

for

เหมาะเมื่อ **รู้จำนวนรอบที่ต้องการวนซ้ำ**

```
for(int i=0;i<10;i++){

}
```

ตัวอย่าง

- เปิด LED 4 ดวง
- อ่านเซ็นเซอร์ 8 ตัว
- แสดงข้อมูล 20 บรรทัด

`while`

เหมาะเมื่อ **ไม่รู้ว่าจะวนกี่รอบ** และต้องรอจนกว่าเงื่อนไขจะเปลี่ยน

```
while(digitalRead(2)==LOW){

}
```

ตัวอย่าง

- รอให้กดปุ่ม
- รอรับข้อมูลจาก Serial
- รอเซ็นเซอร์ตรวจพบวัตถุ

สรุปการจำแบบง่าย

ให้จำโครงสร้างนี้ไว้

```
for (เริ่มต้น; เงื่อนไข; เปลี่ยนค่า) {

  คำสั่งที่ต้องการทำซ้ำ

}
```

เช่น

```
for (int i = 0; i < 4; i++) {
  digitalWrite(led[i], HIGH);
  delay(500);
  digitalWrite(led[i], LOW);
}
```

แปลเป็นภาษาง่าย ๆ คือ

> **เริ่มที่ `i = 0` → ถ้า `i < 4` ให้ทำคำสั่งในลูป → เมื่อจบรอบให้เพิ่ม `i` ทีละ 1 → ทำซ้ำจน `i` มีค่าเป็น 4 แล้วจึงออกจากลูป**

หลักการนี้เป็นพื้นฐานที่ใช้บ่อยมากใน Arduino ไม่ว่าจะเป็นการควบคุม LED, รีเลย์, เซอร์โว, LCD, การอ่านเซ็นเซอร์ หรือการจัดการอาร์เรย์ (Array) ครับ

คาบที่ 8 8/07/69

ตัวแปรนี้เป็น Array (อาร์เรย์)

https://drive.google.com/file/d/1HArwPGeyAXW3TMbt46RIP_rcuMtJCknE/view?usp=drive_link

`sizeof()` เป็น ตัวดำเนินการ (Operator) ในภาษา C/C++ ที่ใช้สำหรับ หาขนาดของข้อมูลเป็นหน่วยไบต์ (Byte) ซึ่ง Arduino ก็รองรับคำสั่งนี้

รูปแบบการใช้งาน
sizeof(ตัวแปร)

หรือ

sizeof(ชนิดข้อมูล)

ตัวอย่าง

```
int a;
Serial.println(sizeof(a));
```

ผลลัพธ์ (Arduino Uno)

2

เพราะ `int` บน Arduino Uno มีขนาด 2 ไบต์

ตัวอย่างที่ 1 หาขนาดของชนิดข้อมูล

```
void setup() {
  Serial.begin(9600);

  Serial.print("char = ");
  Serial.println(sizeof(char));

  Serial.print("int = ");
  Serial.println(sizeof(int));

  Serial.print("float = ");
  Serial.println(sizeof(float));

  Serial.print("long = ");
  Serial.println(sizeof(long));
}

void loop() {
}
```

ผลลัพธ์บน Arduino Uno

ชนิดข้อมูล	ขนาด
char	1 Byte
int	2 Bytes
float	4 Bytes
long	4 Bytes

ตัวอย่างที่ 2 ใช้กับ Array

```
int led[] = {2,3,4,5};

void setup() {

  Serial.begin(9600);

  Serial.println(sizeof(led));
```

```
}

void loop() {
}
```

ผลลัพธ์

8

หลายคนสงสัยว่า ทำไมได้ 8 ไม่ใช่ 4

เพราะ

led มีข้อมูล 4 ตัว

แต่ int แต่ละตัว = 2 Bytes

ดังนั้น

4 × 2 = 8 Bytes

ภาพประกอบ

led[]

```
+-----+-----+-----+-----+
|  2  |  3  |  4  |  5  |
+-----+-----+-----+-----+
```

แต่ละช่อง = 2 Bytes

รวมทั้งหมด = 8 Bytes

ตัวอย่างที่ 3 หาจำนวนสมาชิกใน Array

นี่คือการใช้งานที่พบบ่อยที่สุด

```
int led[] = {2,3,4,5};

int totalLED = sizeof(led) / sizeof(led[0]);
```

มาดูทีละส่วน

①
sizeof(led)

ได้

8

เพราะมีทั้งหมด 8 Bytes

---

②
sizeof(led[0])

คือ

sizeof(int)

ได้

2

เพราะสมาชิกตัวแรกเป็น `int`

---

③

นำมาหารกัน

8 ÷ 2 = 4

จึงได้

มีสมาชิกทั้งหมด 4 ตัว

ทำไมต้องใช้วิธีนี้

สมมติวันนี้มี LED 4 ดวง

int led[] = {2,3,4,5};

พรุ่งนี้เพิ่มอีก 2 ดวง

int led[] = {2,3,4,5,6,7};

หากเขียนแบบนี้

for(int i=0;i<4;i++)

ต้องกลับมาแก้เป็น

for(int i=0;i<6;i++)

แต่ถ้าใช้ `sizeof()`

```
int totalLED = sizeof(led) / sizeof(led[0]);

for(int i=0;i<totalLED;i++)
```

จะได้

totalLED = 6

โดยอัตโนมัติ ไม่ต้องแก้โค้ดในลูป

ตัวอย่างการใช้งานจริง

```
int led[] = {2,3,4,5};

void setup() {

  int totalLED = sizeof(led) / sizeof(led[0]);

  for(int i=0;i<totalLED;i++){

      pinMode(led[i], OUTPUT);

  }

}

void loop() {

}
```

ข้อดีคือ หากเพิ่มหรือลดจำนวน LED เพียงแก้ไขข้อมูลในอาร์เรย์ ลูปจะทำงานได้ถูกต้องโดยอัตโนมัติ

สรุป

คำสั่ง	ความหมาย
sizeof(int)	ขนาดของชนิดข้อมูล `int` (ไบต์)
sizeof(led)	ขนาดรวมของอาร์เรย์ `led` (ไบต์)
sizeof(led[0])	ขนาดของสมาชิกตัวแรกในอาร์เรย์ (ไบต์)
sizeof(led) / sizeof(led[0])	จำนวนสมาชิกทั้งหมดในอาร์เรย์

ข้อควรระวัง: วิธี `sizeof(array) / sizeof(array[0])` ใช้ได้เฉพาะในขอบเขตที่ตัวแปรยังเป็น อาร์เรย์จริง เท่านั้น หากส่งอาร์เรย์เข้าไปในฟังก์ชัน เช่น `void test(int led[])` ภายในฟังก์ชัน `led` จะกลายเป็นตัวชี้ (pointer) ทำให้ `sizeof(led)` ไม่ได้คืนค่าขนาดของอาร์เรย์ทั้งหมดอีกต่อไปครับ

ตัวอย่างที่ 1 : LED วิ่งทีละดวง

การต่อวงจร

- LED1 → ขา D2
- LED2 → ขา D3
- LED3 → ขา D4
- LED4 → ขา D5

```
// กำหนดขา LED
int led[] = {2, 3, 4, 5};
int totalLED = 4;

void setup() {
 // ตั้งค่าขา LED เป็น OUTPUT
 for (int i = 0; i < totalLED; i++) {
   pinMode(led[i], OUTPUT);
 }
}

void loop() {

 // เปิด LED ทีละดวง
 for (int i = 0; i < totalLED; i++) {

   // ปิดทุกดวงก่อน
   for (int j = 0; j < totalLED; j++) {
     digitalWrite(led[j], LOW);
   }

   // เปิดเฉพาะดวงที่ต้องการ
   digitalWrite(led[i], HIGH);

   delay(300);
 }

}
```

ตัวอย่างที่ 2 : LED วิ่งไป-กลับ

```
int led[] = {2, 3, 4, 5};
int totalLED = 4;

void setup() {
 for (int i = 0; i < totalLED; i++) {
   pinMode(led[i], OUTPUT);
 }
}

void loop() {

 // วิ่งไปข้างหน้า
 for (int i = 0; i < totalLED; i++) {
   digitalWrite(led[i], HIGH);
   delay(200);
   digitalWrite(led[i], LOW);
 }

 // วิ่งย้อนกลับ
 for (int i = totalLED - 2; i > 0; i--) {
   digitalWrite(led[i], HIGH);
   delay(200);
   digitalWrite(led[i], LOW);
 }

}
```

ลำดับจะเป็น

LED1
↓
LED2
↓
LED3
↓
LED4
↑
LED3
↑
LED2
แล้ววนใหม่

ตัวอย่างที่ 3 : เปิด LED สะสม แล้วดับพร้อมกัน

```
int led[] = {2, 3, 4, 5};
int totalLED = 4;

void setup() {
 for (int i = 0; i < totalLED; i++) {
   pinMode(led[i], OUTPUT);
 }
}

void loop() {

 // เปิดสะสม
 for (int i = 0; i < totalLED; i++) {
   digitalWrite(led[i], HIGH);
   delay(300);
 }

 delay(500);

 // ดับพร้อมกัน
 for (int i = 0; i < totalLED; i++) {
   digitalWrite(led[i], LOW);
 }

 delay(500);

}
```

ผลลัพธ์

●○○○
●●○○
●●●○
●●●●
ดับทั้งหมด
วนใหม่

ตัวอย่างที่ 4 : ใช้ `switch-case` ควบคุม LED แบบวนซ้ำ

```
int step = 0;

void setup() {
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
}

void loop() {

 // ปิดทุกดวงก่อน
 digitalWrite(2, LOW);
 digitalWrite(3, LOW);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);

 switch (step) {
   case 0:
     digitalWrite(2, HIGH);
     break;

   case 1:
     digitalWrite(3, HIGH);
     break;

   case 2:
     digitalWrite(4, HIGH);
     break;

   case 3:
     digitalWrite(5, HIGH);
     break;
 }

 delay(300);

 step++;

 if (step > 3) {
   step = 0;
 }
}
```

วิธีนี้เหมาะเมื่อแต่ละขั้นตอนต้องทำงานแตกต่างกัน เช่น เปิดรีเลย์, สั่งเซอร์โว, แสดงข้อความบนจอ LCD หรือทำงานหลายอย่างในแต่ละสถานะ

หากต้องการ **ตัวอย่างระดับมืออาชีพ** ผมสามารถยกตัวอย่างการเขียนเป็น **State Machine (FSM)** ซึ่งเหมาะกับโปรเจกต์ Arduino ขนาดใหญ่ เช่น ระบบควบคุมไฟจราจร เครื่องแยกเหรียญ หรือระบบตรวจจับต่าง ๆ ที่คุณกำลังพัฒนาอยู่ โดยโค้ดจะอ่านง่ายและขยายระบบได้สะดวกกว่า `for` หรือ `switch-case` ทั่วไปครับ

ขอคำอธิบายการใช้ คำสั่ง for

คาบที่ 9 15/07/69

แนวคิดการสร้างเมนู เลือกรูปแบบไฟวิ่ง

1. สร้างคำสั่งเพื่อรับแป้นพิมพ์
   ทดสอบว่ากด ตัวเลขแล้วแสดงตัวเลขหรือไม่
   (โดยไม่ต้องรอกด Enter)

2. แทรกโค๊ด switch-case เพียง 1 การทำงาน
   ทดสอบว่า กดหมายเลข 1 แล้ว ทำงานตามที่ตั้งใจหรือไม่

3. แทรกโค๊ด switch-case การทำงาน
   ทดสอบว่า กดหมายเลข 1 , 2 แล้วให้ผลการทำงานที่แตกต่างกัน

4. แทรกโค๊ด switch-case ให้ครบ 5 รูปแบบ โดยทุกขั้นตอน ต้องทดสอบ เพื่อจะได้รู้ว่า ขั้นตอนใดมีข้อผิดพลาด