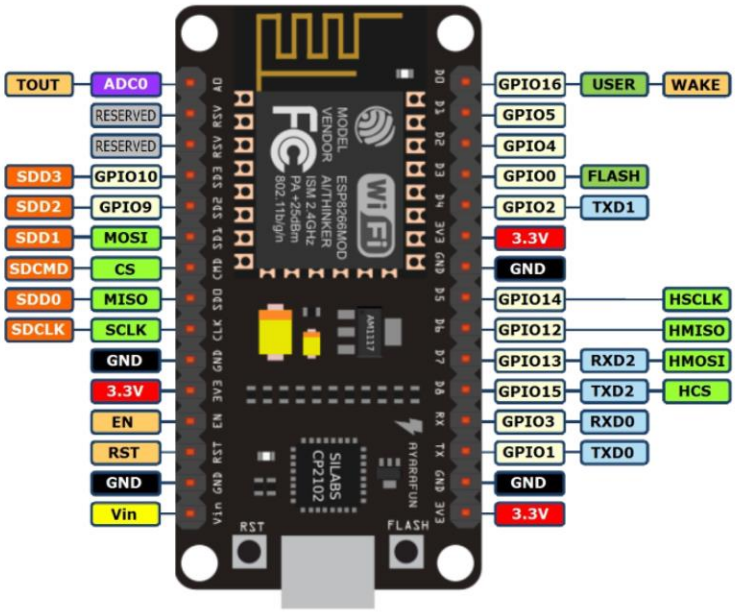
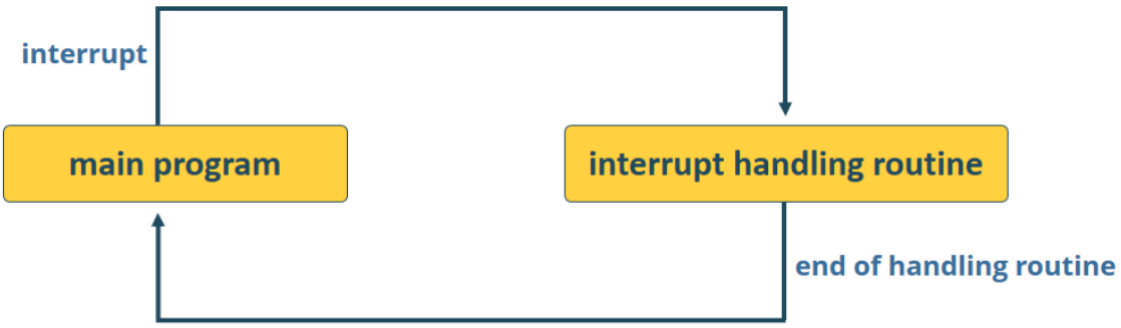
การใช้งานการอินเตอร์รัพท์ (Interrupt) NodeMCU ESP8266



อินเตอร์รัพท์ (Interrupt) คือ การขัดจังหวะการทำงานของโปรแกรมปกติ เมื่อเกิดเหตุการณ์บางอย่างขึ้น ทำให้ซีพียูไปทำงานที่กำหนดไว้เมื่อเกิดอินเตอร์รัพท์ หากเปรียบเทียบการเกิดอินเตอร์รัพท์กับชีวิตประจำวันละก็ มันก็จะเหมือนกับตอนที่เรากำลังดูทีวีอยู่ แต่มีคนโทรเข้ามาในโทรศัพท์มือถือ ทำให้เราต้องหยุดดูทีวีแล้วไปรับโทรศัพท์ เมื่อคุยโทรศัพท์เสร็จแล้วจึงกลับมาดูทีวีตามเดิม



รูปที่ 1 กระบวนการอินเตอร์รัพท์

การขัดจังหวะมีประโยชน์ในการทำให้สิ่งต่าง ๆ เกิดขึ้นโดยอัตโนมัติในโปรแกรมไมโครคอนโทรลเลอร์ และสามารถช่วยแก้ปัญหาเรื่องเวลาได้

ด้วยการขัดจังหวะ คุณไม่จำเป็นต้องตรวจสอบค่าพินปัจจุบันอย่างต่อเนื่อง เมื่อตรวจพบการเปลี่ยนแปลงเหตุการณ์จะถูกทริกเกอร์ – ฟังก์ชันจะถูกเรียก ฟังก์ชันนี้เรียกว่ารูทีนบริการขัดจังหวะ (ISR)

เมื่ออินเตอร์รัปต์เกิดขึ้น โปรเซสเซอร์จะหยุดการทำงานของโปรแกรมหลักเพื่อรันงาน จากนั้นกลับไปที่โปรแกรมหลักดังแสดงในรูป 1

สิ่งนี้มีประโยชน์อย่างยิ่งในการทริกเกอร์การดำเนินการเมื่อตรวจพบการเคลื่อนไหวหรือเมื่อใดก็ตามที่มีการกดปุ่มโดยไม่จำเป็นต้องตรวจสอบสถานะอย่างต่อเนื่อง

ชนิดของอินเตอร์รัพท์

แบ่งตามชนิดของการเกิดได้ดังนี้

1. อินเตอร์รัพท์จากภายนอก เช่น การเปลี่ยนสถานะลอจิกของพอร์ตใดพอร์ตหนึ่ง
2. อินเตอร์รัพท์จากภายใน เช่น อินเตอร์รัพท์ที่เกิดจากทามเมอร์

การควบคุมอินเตอร์รัพท์

การควบคุมอินเตอร์รัพท์ คือการควบคุมว่าจะให้ซีพียูตอบสนองต่ออินเตอร์รัพท์หรือไม่ แบ่งได้ดังนี้

• Disable Interrupt คือ การควบคุมให้ซีพียูไม่ตอบสนองกับอินเตอร์รัพท์ เมื่อเกิดการอินเตอร์รัพท์ขึ้นซีพียูจะปล่อยผ่านอินเตอร์รัพท์นั้น

• Enable Interrupt คือ การควบคุมให้ซีพียูตอบสนองต่ออินเตอร์รัพท์ไปตามปกติ

การควบคุมอินเตอร์รัพท์จะใช้ในกรณีที่ต้องการให้ซีพียูกระทำคำสั่งที่ไม่สามารถหยุดการทำงานได้ เช่น การนับเวลา หากมีการอินเตอร์รัพท์เกิดขึ้นจะทำให้การนับเวลาคลาดเคลื่อนได้

การใช้งานการอินเตอร์รัพท์ใน NodeMCU ESP8266

NodeMCU ESP8266 การใช้งานอินเตอร์รัพท์จากภายนอกใช้เพียงแค่การสร้างฟังก์ชั่นรอรับ แล้วจึงใช้คำสั่งที่กำหนดว่าจะให้เกิดอินเตอร์รัพท์เมื่อไร แต่หากเป็นการอินเตอร์รัพท์จากภายในจะค่อนข้างยุ่งยากมากๆ ดังนั้นในใบงานนี้จึงจะกล่าวถึงการใช้อินเตอร์รัพท์จากภายนอกเท่านั้น

NodeMCU ESP8266 มีคุณสมบัติในการใช้งานอินเตอร์รัพท์จากภายนอกได้เกือบทุกขา ได้แก่ขา GPIO0(D3)-GPIO15(D8) *ยกเว้นขา GPIO16 (D0)* จะไม่สามารถใช้งานอินเตอร์รัพท์ได้

ฟังก์ชัน Interrupt สำหรับ NodeMCU

ในการใช้งานการอินเตอร์รัพท์ใน Arduino IDE นั้น เราสามารถใช้ฟังชั่น attachInterrupt() ในการเรียกใช้งานการอินเตอร์รัพท์ โดยมีรูปแบบการใช้งานของคำสั่ง ดังนี้

รูปแบบคำสั่ง attachInterrupt(digitalPinToInterrupt(GPIO), ISR, mode);

ขา GPIO สำหรับการอินเตอร์รัพท์

สิ่งแรกที่ต้องกำหนดในฟังชั่น attachInterrupt() นั่นก็ คือ ขา GPIO สำหรับการกระตุ้น (Triger) ให้เกิดการอินเตอร์รัพท์ โดยมีรูปแบบกำหนดขาการใช้งานผ่านคำสั่ง digitalPinToInterrupt(GPIO) ตัวอย่าง เช่น หากต้องการใช้ขา GPIO14 เป็นขาสำกรับกระตุ้นการเกิดอินเตอร์รัพท์ จะกำหนดดังนี้

digitalPinToInterrupt(14)

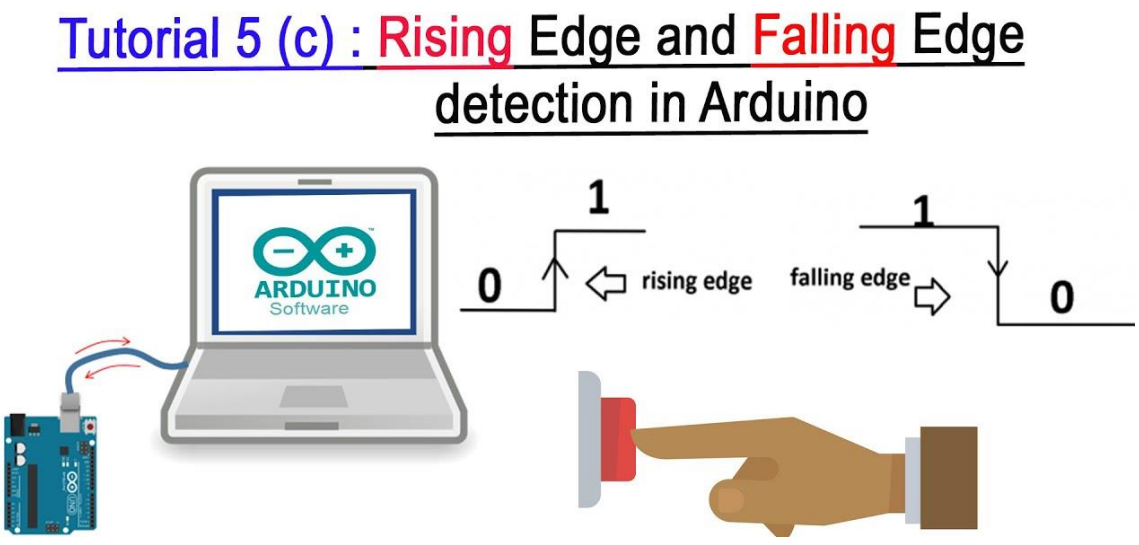
โปรแกรมบริการอินเตอร์รัพท์ (interrupt service routine : ISR)

สิ่งต่อมาต้องกำหนดในฟังชั่น attachInterrupt() นั่นก็ คือ ชื่อของฟังชั่นที่ถูกเรียกขึ้นมาใช้งานในทุกๆ ครั้งเมื่อเกิดการกระตุ้น (Triger) ให้เกิดการอินเตอร์รัพท์ ฟังก์ชัน ISR ควรจะเรียบง่ายที่สุดเท่าที่จะเป็นไปได้ ควรหลีกเลี่ยงการหน่วงเวลา เพื่อให้โปรเซสเซอร์กลับไปดำเนินการตามโปรแกรมหลักได้อย่างรวดเร็ว

Interrupt Modes

รูปแบบ หรือ mode การเกิดอินเตอร์รัพท์นั้น มีทั้งหมด 3 รูปแบบดังนี้

- **CHANGE** จะเกิดอินเตอร์รัพท์เมื่อพอร์ตที่กำหนดไว้มีการเปลี่ยนสถานะ เช่น จากสถานะ HIGH เป็น LOW หรือจาก LOW เป็น HIGH
- **RISING** จะเกิดอินเตอร์รัพท์เมื่อพอร์ตที่กำหนดไว้มีการเปลี่ยนสถานะจาก LOW เป็น HIGH
- **FALLING** จะเกิดอินเตอร์รัพท์เมื่อพอร์ตที่กำหนดไว้มีการเปลี่ยนสถานะจาก HIGH เป็น LOW



รูปที่ 2 ลักษณะรูปแบบการเกิดอินเตอร์รัพท์

ตัวอย่างการใช้งานโปรแกรมการอินเตอร์รัพท์ ในรูปแบบ หรือ โหมด (Mode) ต่างๆ

การอินเตอร์รัพท์ใน Mode : CHANGE

```
#define LED_PIN D0
#define GPIO_PIN D7

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

  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(GPIO_PIN), IntCallback, FALLING);
}

void loop() {
  delay(10);
}
/* ---------- Interrurpt Service Rutine ---------- */
ICACHE_RAM_ATTR void IntCallback() {
  Serial.println("Interrurpt OK!");
  digitalWrite(LED_PIN, !digitalRead(LED_PIN));
}
/* ------------------------------------------------ */
```

การอินเตอร์รัพท์ใน Mode : RISING

```
#define LED_PIN D0
#define GPIO_PIN D7

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

  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(GPIO_PIN), IntCallback, RISING);
}

void loop() {
  delay(10);
}
/* ---------- Interrurpt Service Rutine ---------- */
ICACHE_RAM_ATTR void IntCallback() {
  Serial.println("Interrurpt OK!");
  digitalWrite(LED_PIN, !digitalRead(LED_PIN));
}
/* ------------------------------------------------ */
```

การอินเตอร์รัพท์ใน Mode : FALLING

```
#define LED_PIN D0
#define GPIO_PIN D7

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

  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(GPIO_PIN), IntCallback, FALLING);
}

void loop() {
  delay(10);
}
/* ---------- Interrurpt Service Rutine ---------- */
ICACHE_RAM_ATTR void IntCallback() {
  Serial.println("Interrurpt OK!");
  digitalWrite(LED_PIN, !digitalRead(LED_PIN));
}
/* ------------------------------------------------ */
```

ตัวอย่างการประยุกร์ใช้งานการอินเตอร์รัพท์

```
#define LED1_PIN D0
#define LED2_PIN D4
#define GPIO_PIN D3

void setup() {
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);

  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(GPIO_PIN), IntCallback, FALLING);
}

void loop() {
  digitalWrite(LED2_PIN,1);
  delay(2000);
  digitalWrite(LED2_PIN,0);
  delay(2000);
}
/* ---------- Interrurpt Service Rutine ---------- */
ICACHE_RAM_ATTR void IntCallback() {
  Serial.println("Interrurpt OK!");
  digitalWrite(LED1_PIN, !digitalRead(LED1_PIN));
}
/* ------------------------------------------------ */
```