Appendix K

การทดลองที่ 11 การเชื่อมต่อกับอินเทอร์รัพท์

การทดลองนี้คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C และแอสเซมบลี ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อพัฒนาการทำงานของอินเทอร์รัพท์ร่วมโปรแกรมภาษา C
- เพื่อศึกษาการทำงานของอินเทอร์รัพท์ร่วมกับขา GPIO ตามเนื้อหาในบทที่ 2 หัวข้อที่ 2.11

K.1 อินเทอร์รัพท์

สัญญาณร้องขออินเทอร์รัพท์ หรือ สัญญาณร้องขอการขัดจังหวะ (Interrupt Request) คือ สัญญาณที่เกิดขึ้นจากอุปกรณ์อินพุท/เอาท์พุท จาก GPU และจากเหตุการณ์พิเศษ สัญญาณเหล่านี้จะทำให้ CPU หยุดพักโปรแกรมที่กำลังรันอยู่เป็นการชั่วคราว แล้วไปดำเนินการบางอย่างเพื่อตอบสนอง (Respond) หรือให้บริการ (Service) ต่อเหตุการณ์ที่เกิดขึ้น การตอบสนองหรือการบริการ นี้เรียกโดยรวมว่า **Interrupt Service Routine (ISR)** เมื่อซีพียูตอบสนองหรือบริการเสร็จสิ้น ซีพียูจะกลับไปทำสิ่งที่หยุดพักนั้นต่อ

รายละเอียดการทำงานของอินเทอร์รัพท์ มีขั้นตอนดังนี้

1. **เหตุการณ์อินเทอร์รัพท์ (Interrupt Event)**: กดปุ่มต่างๆ การกดแป้นพิมพ์ การจับเวลา (Timer) เป็นต้น
2. **การร้องขอการขัดจังหวะ (Interrupt Request)**: ส่งสัญญาณร้องขอไปยังซีพียู
3. **ขั้นตอนการให้บริการ (Interrupt Service Routine เรียกย่อๆ ว่า ISR)**: ฟังค์ชันที่ซีพียูจะต้องปฏิบัติเพื่อให้บริการตามเหตุการณ์ที่ร้องขอ

เหตุการณ์อินเทอร์รัพท์ มีความสำคัญ (Priority) แตกต่างกัน ISR สำหรับแต่ละเหตุการณ์มักจะเขียนในรูปแบบของฟังค์ชัน ที่ไม่มีพารามิเตอร์และไม่มีค่ารีเทิร์น หรือ void เหตุการณ์แบ่งเป็น 2 ชนิด คือ

- **ฮาร์ดแวร์อินเทอร์รัพท์ (Hardware interrutps)**: เหตุการณ์ที่เกิดจากการทำงานร่วมกับอุปกรณ์อินพุทและเอาท์พุท เช่น ปุ่มกดต่างๆ การรับส่งข้อมูลแบบอนุกรม (Seiral communication) เช่น UART (Universal Asynchronous Receive TRansmit), SPI (Serial Peripheral Interface)

เป็นต้น การเปลี่ยนแปลงของขา GPIO, ตัวจับเวลาถึงเวลาที่ตั้งไว้, การแปลงอนาล็อกเป็นดิจิทัลเสร็จสมบูรณ์, ตัวจับเวลาวอทช์ดอก (Watchdog Timer) หมดเวลา (Timeout) เป็นต้น

- **ซอฟท์แวร์อินเทอร์รัพท์ (Software interrupts)**: เหตุการณ์ที่เกิดจากการทำงานหรือสั่งการโดยซอฟท์แวร์ เช่น การเรียกใช้บริการจากโอเอส ความผิดพลาดของโปรแกรม เป็นต้น

K.2 การจัดการอินเตอร์รัพท์ (Interrupt Handling)

K.2.1 การจัดการอินเทอร์รัพท์ของ WiringPi

ไลบรารี wiringPi รองรับการทำอินเทอร์รัพท์ของ GPIO ได้ ทำให้โปรแกรมหลักสามารถทำงาน หลักได้ตามปกติ เมื่อเกิดสัญญาณอินเทอร์รัพท์ขึ้น ไม่ว่าจะเป็นสัญญาณจากการกดปุ่ม ทำให้เกิดขอบขาขึ้นหรือขอบขาลงหรือทั้งสองขอบ โดยการเรียกใช้คำสั่ง

```
wiringPiISR(pin, edgeType, callback)
```

โดย pin หมายถึง เลขขาที่ wiringPi กำหนด edgeType กำหนดจากค่าคงที่ 4 ค่านี้

- INT_EDGE_FALLING,
- INT_EDGE_RISING,
- INT_EDGE_BOTH
- INT_EDGE_SETUP.

การกำหนดชนิดขอบขาเป็น 3 ชนิดแรก ไลบรารีจะตั้งค่าเริ่มต้น (Innitialization) ให้โดยอัตโนมัติ หากกำหนดชนิดขอบเป็น INT_EDGE_SETUP ไลบรารีจะไม่ตั้งค่าเริ่มต้น (Innitialization) ให้ เนื่องจากโปรแกรมเมอร์จะต้องทำเอง

พารามิเตอร์ callback คือ ชื่อฟังค์ชันที่จะทำหน้าที่เป็น ISR ฟังค์ชัน callback นี้จะเริ่มต้นทำงานโดยแจ้งต่อวงจร Dispatcher ในหัวข้อที่ 2.12 ก่อนจะเริ่มต้นทำงาน โดยฟังค์ชัน callback จะสามารถอ่านหรือเขียนค่าของตัวแปรโกลบอลในโปรแกรมได้ ซึ่งตัวอย่างการทำงานจะได้กล่าวในหัวข้อถัดไป

K.2.2 วงจรปุ่มกด Push Button เชื่อมผ่านขา GPIO

1. ชัทดาวน์และตัดไฟเลี้ยงออกจากบอร์ด Pi3 เพื่อความปลอดภัยในการต่อวงจร
2. ต่อวงจรตามรูปที่ K.1

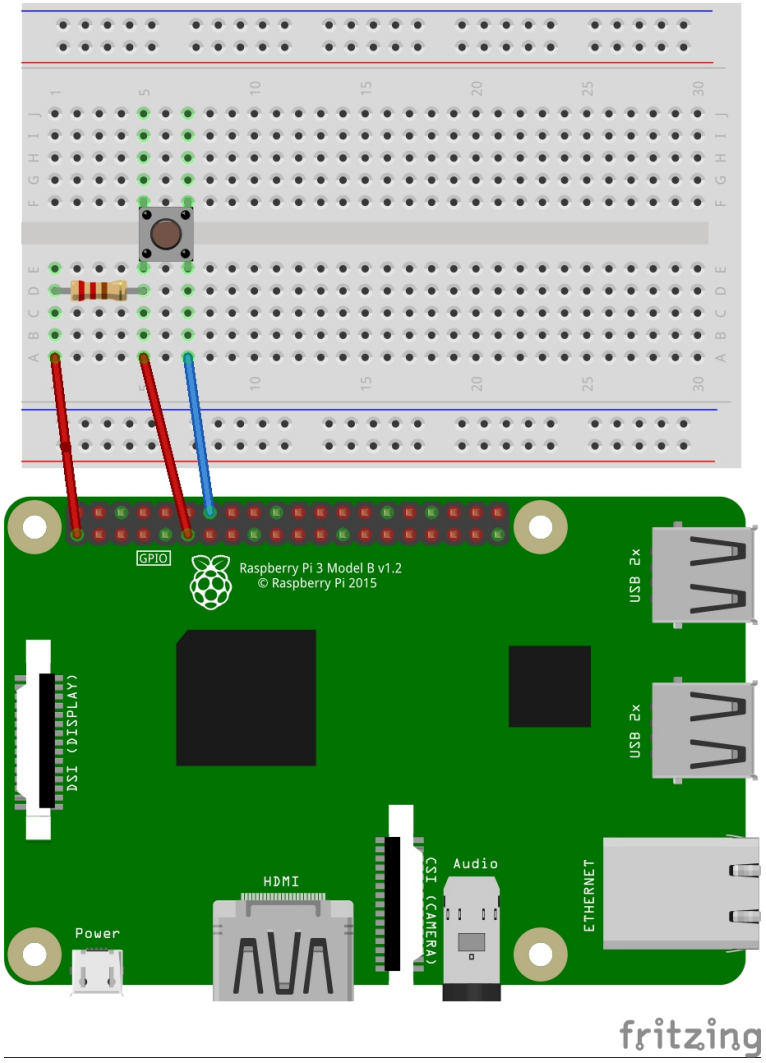

Figure K.1: วงจรกดปุ่มสำหรับทดลองการเขียนโปรแกรมอินเทอร์รัพต์ในการทดลองที่ 11 ที่มา: fritzing.org

3. ตรวจสอบความถูกต้อง โดยให้ผู้ควบคุมการทดลองตรวจสอบ
4. จ่ายไฟเลี้ยงให้กับบอร์ดแล้วสังเกตการเปลี่ยนแปลงที่หลอด LED
5. เรียกโปรแกรม Code::Blocks ผ่านทาง Terminal โดยใช้สิทธิ์ของ SuperUser ดังนี้

```
sudo codeblocks
```

6. สร้าง project ใหม่ชื่อ Lab11.2 ภายใต้โฟลเดอร์ /home/pi/asm/Lab11.2 รายละเอียดบางอย่างต้องเปิดในการทดลองที่ 8 ภาคผนวก J

K.2.3 โปรแกรมภาษา C สำหรับทดสอบวงจรอินเทอร์รัพท์

ผู้อ่านต้องทำความเข้าใจกับตัวโปรแกรมก่อนคอมไพล์หรือรันโปรแกรม เพื่อความเข้าใจสูงสุด โดยเฉพาะชื่อตัวแปร ชนิดของตัวแปร evenCounter การติดตั้งฟังค์ชัน wiringPiISR เพื่อเชื่อมโยงกับขา GPIO ชนิดของการตรวจจับ และชื่อฟังค์ชัน myInterrupt ซึ่งทำหน้าที่เป็น ISR หรือ ฟังค์ชัน callback

```
#include <stdio.h>
#include <errno.h>
#include <wiringPi.h>
#define BUTTON_PIN 0
// Use GPIO Pin 17, which is Pin 0 for wiringPi library

volatile int eventCounter = 0;

// myInterrupt:  called every time an event occurs
void myInterrupt(void) {
  eventCounter++; // the event counter
}

int main(void) {
  if (wiringPiSetup () < 0)   // check the existence of wiringPi library
  {
    printf ( "Unable to setup wiringPi: %s\n", strerror (errno));
    return 1;
  }
  // set wiringPi Pin 0 to generate an interrupt from 1-0 transition
  // myInterrupt() = ISR
  if ( wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0)
  {
    printf ( "Unable to setup ISR: %s\n", strerror (errno));
    return 1;
  }
  // display counter value every second
  while ( 1 ) {
    printf( "%d\n", eventCounter );
    eventCounter = 0;
    delay( 1000 ); // wait 1 second
```

```
  }
  return 0;
}
```

1. ป้อนโปรแกรมด้านบนใน main.c แล้วคอมไพล์จนไม่เกิดข้อผิดพลาด
2. รันโปรแกรม ทดสอบการทำงานด้วยการกดปุ่มที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน
3. จงบอกความหมายและการประยุกต์ใช้งานตัวแปรชนิด volatile
4. ปรับแก้ volatile ออกเหลือแค่ int eventCounter = 0;
5. รันโปรแกรม ทดสอบการทำงานด้วยการกดปุ่มที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน
6. เปรียบเทียบการทำงานของโปรแกรมก่อนและหลังการปรับแก้ และหาเหตุผล

K.3 กิจกรรมท้ายการทดลอง

1. จงตอบคำถามจากประโยคต่อไปนี้

   ```
   if ( wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0) {

   }
   ```

   - ฟังค์ชัน wiringPiISR ทำหน้าที่อะไร เหตุใดอยู่ในประโยคเงื่อนไข if
   - ตัวแปร &myInterrupt คืออะไร เหตุใดจึงมีสัญลักษณ์ & นำหน้า
   - ฟังค์ชันนี้เชื่อมโยงกับตารางที่ 2.6 อย่างไร

2. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 8 นับขึ้นจาก 0-7-0 โดยเพิ่มปุ่มกดในการทดลองนี้ และเพิ่มฟังค์ชันการอินเทอร์รัพท์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับเพิ่มขึ้น หรือ delay สั้นลงครึ่งหนึ่ง เมื่อกดครั้งที่ 2 จะสั้นลงอีกครึ่งหนึ่ง เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น
3. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 8 แต่นับล งจาก 7-0-7 โดยเพิ่มปุ่มกดในการทดลองนี้ และเพิ่มฟังค์ชันการอินเทอร์รัพท์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับลดลง หรือ delay เพิ่มขึ้นเท่าตัว เมื่อกดครั้งที่ 2 Delay เพิ่มขึ้นอีกเท่าตัว เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น

ข้อ 1)

- ฟังก์ชัน wiringPiISR ทำหน้าที่สั่งให้ฟังก์ชัน myInterrupt ทำงานเมื่อมีสัญญาณ Interrupt เข้ามาจาก pin ที่กำหนดไว้ และที่กำหนดไว้ใน if เพื่อไว้ตรวจสอบ error ที่อาจจะเกิดขึ้นได้
- &myInterrupt คือ ฟังก์ชัน callback เมื่อเกิด Interrupt ส่วนสัญลักษณ์ & หมายถึง pointer
- เรียกใช้ GPIO ต่างๆ

ข้อ 2)

```
#include<stdio.h>
#include<stdlib.h>
#include<wiringPi.h>
int led0 = 0;
int led1 = 2;
int led2 = 3;
int btn = 29;
int i = 0;
int fwd = 1;
unsigned long c, debounce = 200;
int base = 2000;
volatile int ms = 2000;
void halfms(void) {
    if(millis() - c > debounce){
        ms >>= 1;
        if(ms < base>>2) ms = base;
        printf("\t\tms: %d\n",ms);
        c = millis();
    }
}
int main (void) {
       printf("wiringPi LED blinking\n");
       if(wiringPiSetup() < 0){
              printf("Setup problem ... Abort!");
              return 1;
       }
       if ( wiringPiISR (btn, INT_EDGE_RISING, &halfms) < 0)
    {
        printf ( "Unable to setup ISR\n");
        return 1;
    }
       pinMode(led0, OUTPUT);
       pinMode(led1, OUTPUT);
       pinMode(led2, OUTPUT);
       printf("ms: %d\n",ms);
       while(1){
        printf("%d %d%d%d\n",i,(i & 0b100) >> 2,(i & 0b010)>>1,i & 0b001);
              digitalWrite(led0, (i & 4) >> 2);
              digitalWrite(led1, (i & 2)>>1);
              digitalWrite(led2,i & 1);
              delay(ms);
        if(fwd) {
            i++;
            if(i==7) {fwd = !fwd;printf("\n");}
        }
        else {
            i--;
            if(i==0) {fwd = !fwd;printf("\n");}
        }
       }
       return 0;
}
```

ข้อ 3)

```
#include<stdio.h>
#include<stdlib.h>
#include<wiringPi.h>
int led0 = 0;
int led1 = 2;
int led2 = 3;
int btn = 29;
int i = 7;
int fwd = 0;
unsigned long c, debounce = 200;
int base = 500;
volatile int ms = 500;
void halfms(void) {
    if(millis() - c > debounce){
        ms <<= 1;
        if(ms > base<<2) ms = base;
        printf("\t\tms: %d\n",ms);
        c = millis();
    }
}
int main (void) {
       printf("wiringPi LED blinking\n");
       if(wiringPiSetup() < 0){
              printf("Setup problem ... Abort!");
              return 1;
       }
       if ( wiringPiISR (btn, INT_EDGE_RISING, &halfms) < 0)
    {
        printf ( "Unable to setup ISR\n");
        return 1;
    }
       pinMode(led0, OUTPUT);
       pinMode(led1, OUTPUT);
       pinMode(led2, OUTPUT);
       printf("ms: %d\n",ms);
       while(1){
        printf("%d %d%d%d\n",i,(i & 0b100) >> 2,(i & 0b010)>>1,i & 0b001);
              digitalWrite(led0, (i & 4) >> 2);
              digitalWrite(led1, (i & 2)>>1);
              digitalWrite(led2,i & 1);
              delay(ms);
        if(fwd) {
            i++;
            if(i==7) {fwd = !fwd;printf("\n");}
        }
        else {
            i--;
            if(i==0) {fwd = !fwd;printf("\n");}
        }
       }
       return 0;
}
```