นาย ภูมิภัทร เชื้อลือ 62010722
นาย ธีรวิทย์ ธุรวัฒนวิทย์ 62010948

# ภาคผนวก K

# การทดลองที่ 11 การเชื่อมต่อกับสัญญาณอินเทอร์รัพท์

การทดลองนี้คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C และแอสเซมบลีจากการทดลองก่อนหน้า ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อพัฒนาการทำงานของอินเทอร์รัพท์ร่วมโปรแกรมภาษา C และแอสเซมบลี ตามเนื้อหาในหัวข้อที่ 6.12
- เพื่อศึกษาการทำงานของอินเทอร์รัพท์ร่วมกับขา GPIO ตามเนื้อหาใน หัวข้อที่ 6.11

## K.1 การจัดการอินเทอร์รัพท์ของ WiringPi

ไลบรารี wiringPi รองรับการทำอินเทอร์รัพท์ของ GPIO ได้ ทำให้โปรแกรมหลักสามารถทำงาน หลักได้ตามปกติ เมื่อเกิดสัญญาณอินเทอร์รัพท์ขึ้น ไม่ว่าจะเป็นสัญญาณจากการกดปุ่ม ทำให้เกิดขอบขาขึ้นหรือขอบขาลง หรือทั้งสองขอบ 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.

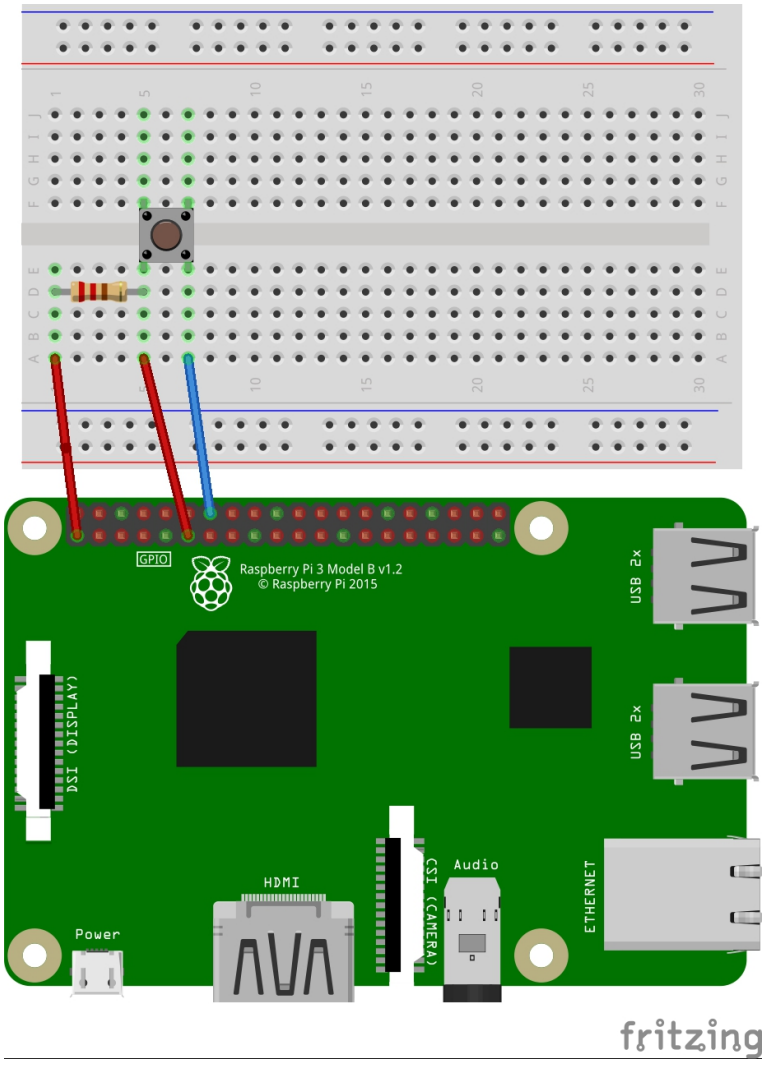
การกำหนดชนิดขอบขาเป็น 3 ชนิดแรก ไลบรารีจะตั้งค่าเริ่มต้น (Innitialization) ให้โดยอัตโนมัติ หากกำหนดชนิดขอบเป็น INT_EDGE_SETUP ไลบรารีจะไม่ตั้งค่าเริ่มต้น (Initialization) ให้ เนื่องจากโปรแกรมเมอร์จะต้องกำหนดเอง

พารามิเตอร์ **callback** คือ ชื่อฟังค์ชันที่จะทำหน้าที่เป็น ISR สัญลักษณ์ & หมายถึง แอดเดรสของฟังค์ชัน callback ฟังค์ชัน callback นี้จะเริ่มต้นทำงานโดยแจ้งต่อวงจร Dispatcher ในหัวข้อที่ 6.12 ก่อนจะเริ่มต้น

ทำงาน โดยฟังค์ชัน callback จะสามารถอ่าน หรือเขียนค่าของตัวแปรโกลบอลในโปรแกรมได้ ซึ่งตัวอย่างการทำงานจะได้กล่าวในหัวข้อถัดไป

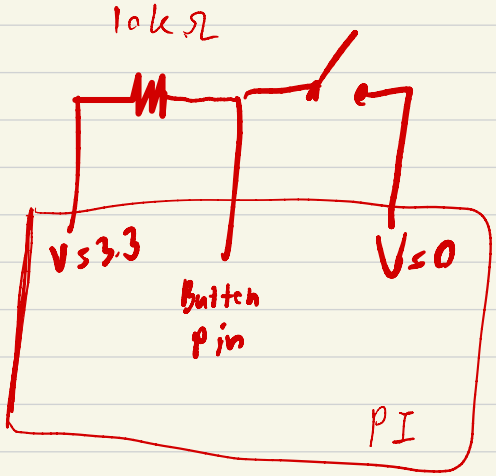
## K.2 วงจรปุ่มกด Push Button เชื่อมผ่านขา GPIO

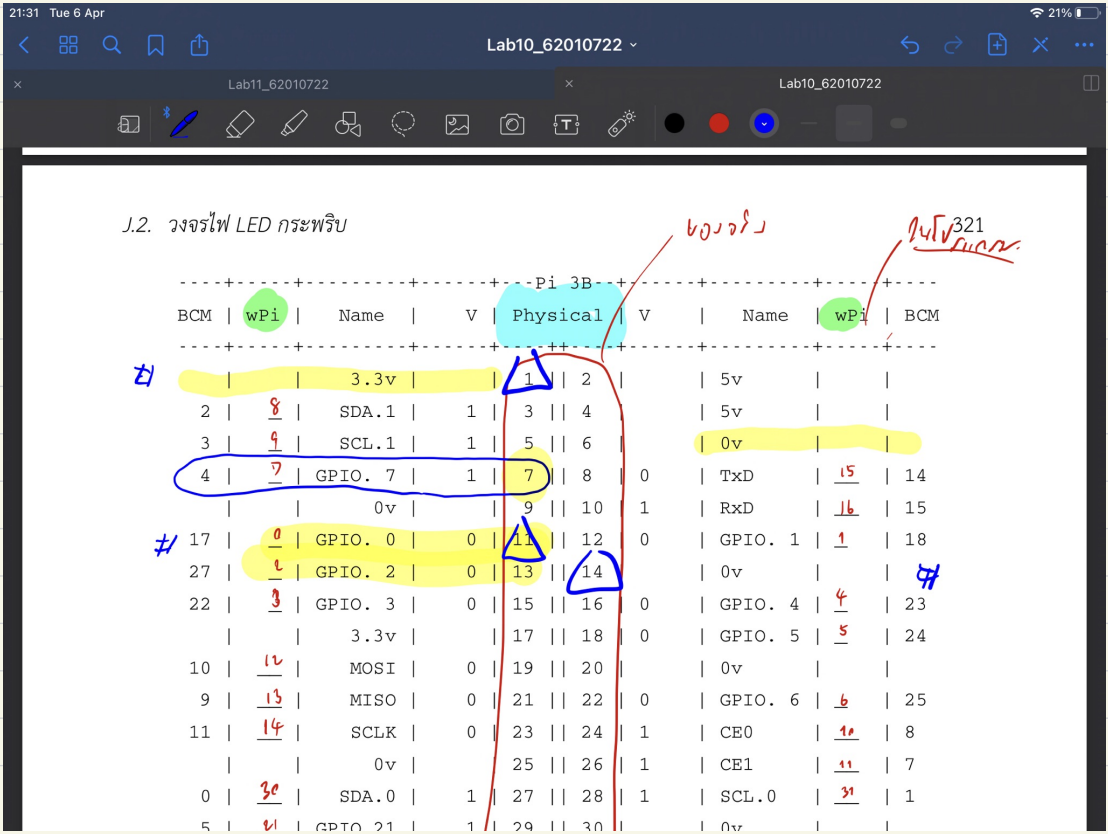
1. ชัทดาวน์และตัดไฟเลี้ยงออกจากบอร์ด Pi3 เพื่อความปลอดภัยในการต่อวงจร
2. ต่อวงจรตามรูปที่ K.1




**รูปที่** K.1: วงจรกดปุ่มสำหรับทดลองการเขียนโปรแกรมอินเทอร์รัพต์ในการทดลองที่ 11 ที่มา: fritzing.org

3. จงวาดวงจรที่ต่อในรูปที่ K.1 ประกอบด้วย สวิทช์กดปุ่ม ตัวต้านทาน ไฟเลี้ยง 3.3 โวลท์ ขา BUTTON_PIN และกราวด์ (0 โวลท์)
4. ตรวจสอบความถูกต้อง โดยให้ผู้ควบคุมการทดลองตรวจสอบ
5. สร้าง project ใหม่ชื่อ Lab11 ภายใต้ไดเรคทอรี /home/pi/asm/Lab11




*J.2. วงจรไฟ LED กระพริบ*

ของจริง

ในโปรแกรม 321

| BCM | wPi | Name | V | Physical | V | Name | wPi | BCM |
|---|---|---|---|---|---|---|---|---|
| | | 3.3v | | 1 \|\| 2 | | 5v | | |
| 2 | 8 | SDA.1 | 1 | 3 \|\| 4 | | 5v | | |
| 3 | 9 | SCL.1 | 1 | 5 \|\| 6 | | 0v | | |
| 4 | 7 | GPIO. 7 | 1 | 7 \|\| 8 | 0 | TxD | 15 | 14 |
| | | 0v | | 9 \|\| 10 | 1 | RxD | 16 | 15 |
| 17 | 0 | GPIO. 0 | 0 | 11 \|\| 12 | 0 | GPIO. 1 | 1 | 18 |
| 27 | 2 | GPIO. 2 | 0 | 13 \|\| 14 | | 0v | | |
| 22 | 3 | GPIO. 3 | 0 | 15 \|\| 16 | 0 | GPIO. 4 | 4 | 23 |
| | | 3.3v | | 17 \|\| 18 | 0 | GPIO. 5 | 5 | 24 |
| 10 | 12 | MOSI | 0 | 19 \|\| 20 | | 0v | | |
| 9 | 13 | MISO | 0 | 21 \|\| 22 | 0 | GPIO. 6 | 6 | 25 |
| 11 | 14 | SCLK | 0 | 23 \|\| 24 | 1 | CE0 | 10 | 8 |
| | | 0v | | 25 \|\| 26 | 1 | CE1 | 11 | 7 |
| 0 | 30 | SDA.0 | 1 | 27 \|\| 28 | 1 | SCL.0 | 31 | 1 |
| 5 | 21 | GPIO.21 | 1 | 29 \|\| 30 | | 0v | | |

## K.3 โปรแกรมภาษา C สำหรับทดสอบวงจรอินเทอร์รัพท์

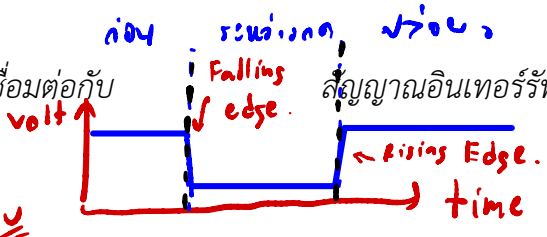
ผู้อ่านต้องทำความเข้าใจกับตัวโปรแกรมก่อนคอมไพล์หรือรันโปรแกรม เพื่อความเข้าใจสูงสุด โดยเฉพาะชื่อตัวแปร ชนิดของตัวแปร evenCounter การติดตั้งฟังค์ชัน wiringPiISR เพื่อเชื่อมโยงกับขา GPIO ชนิดของการตรวจจับ และชื่อฟังค์ชัน myInterrupt ซึ่งทำหน้าที่เป็น ISR หรือ ฟังค์ชัน callback

```
#include <stdio.h>
#include <errno.h>
#include <wiringPi.h>
#define BUTTON_PIN 0
// Use GPIO Pin 17 = Pin 0 of wiringPi library
volatile int eventCount = 0;
void myInterrupt(void) {  // called every time an event occurs
  eventCount++; // the event counter
}
int main(void) {
  if (wiringPiSetup()<0)   // check the existence of wiringPi library
  {
    printf ("Cannot setup wiringPi: %s\n", strerror (errno));
    return 1; // error code = 1
  }
  // set wiringPi Pin 0 to generate an interrupt from 1-0 transition
  // myInterrupt() = my Interrupt Service Routine
  if (wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0) {
    printf ("Cannot setup ISR: %s\n", strerror (errno));
    return 2; // error code = 2
  }
  // display counter value every second
  while(1) {
    printf("%d\n", eventCount);
    eventCount = 0;
    delay(1000); // wait 1000 milliseconds = 1 second
  }
  return 0;  // error code = 0 (No Error)
}
```

Interrupt

1. ป้อนโปรแกรมด้านบนใน main.c โดยใช้โปรแกรม Text Editor ทั่วไป
2. สร้าง makefile สำหรับคอมไพล์และลิงค์โปรแกรมจากการทดลองก่อนหน้านี้ จนไม่เกิดข้อผิดพลาด
3. รันโปรแกรม ทดสอบการทำงานด้วยการกดปุ่มที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน

```
$ sudo ./Lab11
```




## K.4 กิจกรรมท้ายการทดลอง

1. จงวาดสัญญาณที่ขา BUTTON_PIN ก่อนกดปุ่ม ระหว่างกดปุ่ม และปล่อยมือจากปุ่มกด โดยให้แกนนอนเป็นแกนเวลา แกนตั้งเป็นค่าโวลเตจ หรือ ค่าลอจิคของขาสัญญาณ BUTTON_PIN

2. จงบอกความหมายและการประยุกต์ใช้งานตัวแปรชนิด volatile

3. ปรับแก้ volatile ออกเหลือแค่ int eventCount = 0; make แล้วจึงรันโปรแกรมทดสอบการทำงานด้วยการกดปุ่มที่ต่อไว้ สังเกตผลลัพธ์ทางหน้าจอ Terminal ที่รัน เปรียบเทียบการทำงานของโปรแกรมก่อนและหลังการปรับแก้ และหาเหตุผล

4. จงปรับแก้โปรแกรมที่ทดลองตามประโยคต่อไปนี้

```
if (wiringPiISR (BUTTON_PIN, INT_EDGE_RISING, &myInterrupt) < 0) {
 ...
}
```

ทำ make ใหม่และทดลองกดปุ่ม สังเกตการเปลี่ยนแปลงและอธิบาย

5. จงตอบคำถามจากประโยคต่อไปนี้

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

- ฟังค์ชัน wiringPiISR ทำหน้าที่อะไร เหตุใดอยู่ในประโยคเงื่อนไข if
- ตัวแปร &myInterrupt คืออะไร เหตุใดจึงมีสัญลักษณ์ & นำหน้า
- ฟังค์ชันนี้เชื่อมโยงกับตารางที่ 6.6 อย่างไร

6. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 10 นับขึ้นจาก 0-7-0 โดยเพิ่มปุ่มกดในการทดลองนี้ และเพิ่มฟังค์ชันการอินเทอร์รัพท์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับเพิ่มขึ้น หรือ Delay สั้นลงครึ่งหนึ่ง เมื่อกดครั้งที่ 2 จะสั้นลงอีกครึ่งหนึ่ง เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น

7. จงใช้วงจรหลอด LED 3 ดวงและโปรแกรมจากการทดลองที่ 10 แต่นับลงจาก 7-0-7 โดยเพิ่มปุ่มกดในการทดลองนี้ และเพิ่มฟังค์ชันการอินเทอร์รัพท์จากโปรแกรม Lab11.2 นี้ เมื่อกดปุ่มแต่ละครั้งจะทำให้ความเร็วในการนับลดลง หรือ Delay เพิ่มขึ้นเท่าตัว เมื่อกดครั้งที่ 2 Delay เพิ่มขึ้นอีกเท่าตัว เมื่อกดครั้งที่ 3 จะทำให้ Delay กลับไปเป็นค่าเริ่มต้น