Appendix J

การทดลองที่ 10 การเชื่อมต่อกับ GPIO

การทดลองนี้คาดว่าผู้อ่านเคยเรียนการเขียนหรือพัฒนาโปรแกรมด้วยภาษา C มาบ้างแล้ว และมีความคุ้นเคยกับ IDE (Integrated Development Environment) จากพัฒนาโปรแกรมและการดีบักโปรแกรมด้วยภาษา C/C++ และแอสเซมบลี ดังนั้น การทดลองมีวัตถุประสงค์เหล่านี้

- เพื่อปฏิบัติการเชื่อมต่อวงจรกับขา GPIO บนบอร์ด Pi3 ตามเนื้อหาในบทที่ 2 หัวข้อที่ 2.11
- เพื่อพัฒนาโปรแกรมภาษา C ควบคุมการทำงานของขา GPIO
- เพื่อพัฒนาโปรแกรมภาษา Assembly ควบคุมการทำงานของขา GPIO

โปรดสังเกตตัวอักษร w ที่คำว่า wiringPi ต้องเป็นตัวอักษรพิมพ์เล็ก

J.1 ไลบรารี wiringPi

ไลบรารี wiringPi เป็นฟังค์ชันที่พัฒนาด้วยภาษา C สำหรับบอร์ด Pi เป็น OpenSource ภายใต้ GNU LGPLv3 license สามารถเรียกใช้งานผ่าน ภาษา C and C++ รวมถึงแอสเซมบลี

เนื่องจากไลบรารีเป็นซอฟต์แวร์แบบ Open Source แจกให้แก่นักพัฒนาทั่วโลกผ่านทาง https://github.com/WiringPi และมีการปรับปรุงแก้ไขตลอดเวลาโดยทีมนักพัฒนา ดังนั้น ผู้อ่านควรต้องติดตั้งและปรับปรุงระบบปฏิบัติการให้ทันสมัยและติดตั้ง ตามขั้นตอนต่อไปนี้

1. ผู้อ่านควรปรับปรุงระบบปฏิบัติการให้เป็นปัจจุบันก่อน โดยพิมพ์คำสั่งนี้บน Terminal โดยใช้สิทธิ์ของ SuperUser:

   ```
   $  sudo apt-get update
   $  sudo apt-get upgrade
   ```

   ขั้นตอนนี้จะใช้เวลานานและความอดทน รวมถึงการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตที่มีเสถียรภาพ

2. ติดตั้ง wiringPi โดยพิมพ์คำสั่งนี้บน Terminal โดยใช้สิทธิ์ของ SuperUser:

```
$  sudo apt-get install wiringPi
```

คำสั่งนี้จะติดตั้งไลบรารีลงบนบอร์ด

3. เรียกคำสั่ง gpio -v เพื่อทดสอบการติดตั้งไลบรารี wiringPi และได้ผลลัพธ์ของการเรียกดังนี้

```
$   gpio  -v
gpio version: 2.50
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 3, Revision: 02, Memory: 1024MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 3 Model B Rev 1.2
  * This Raspberry Pi supports user-level GPIO access.
```

4. เรียกคำสั่ง gpio readall เพื่อตรวจสอบและบันทึกผลลัพธ์ที่แสดงบนหน้าต่าง Terminal ลงในตารางหน้าถัดไป

```
$   gpio readall
```

จงเติมหมายเลขในคอลัมน์ wPi (wiringPi) ให้ตรงกับขาเชื่อมต่อ 40 ขาบนบอร์ด Pi ตามที่แสดงบนหน้าจอลงในตารางต่อไปนี้ เพื่อใช้ประกอบการต่อวงจรที่ถูกต้อง

```
 ----+-----+---------+------+---Pi 3B--+------+---------+-----+----
 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      |     |
   6 |  22 | GPIO.22 |    1 | 31 || 32 | 0    | GPIO.26 |  26 |  12
  13 |  23 | GPIO.23 |    0 | 33 || 34 |      | 0v      |     |
  19 |  24 | GPIO.24 |    0 | 35 || 36 | 0    | GPIO.27 |  27 |  16
  26 |  25 | GPIO.25 |    0 | 37 || 38 | 0    | GPIO.28 |  28 |  20
     |     |      0v |      | 39 || 40 | 0    | GPIO.29 |  29 |  21
 ----+-----+---------+------+---Pi 3B--+------+---------+-----+----
 BCM | wPi |   Name  |    V | Physical | V    |   Name  | wPi | BCM
 ----+-----+---------+------+----++----+------+---------+-----+----
```

J.2 วงจรไฟ LED กระพริบ

1. รายการอุปกรณ์ที่ต้องใช้:
    - หลอด LED จำนวน 3 หลอด
    - ตัวต้านทาน (Resistor) ที่เตรียมไว้ให้จำนวน 3 ตัว
    - แผ่นต่อวงจรโปรโตบอร์ด
    - สายต่อวงจร
2. ชัทดาวน์และตัดไฟเลี้ยงออกจากบอร์ด Pi3 เพื่อความปลอดภัยในการต่อวงจร
3. ศึกษารูปที่ **??** ให้เข้าใจ แล้วจึงต่อวงจรตามรูปที่ J.1

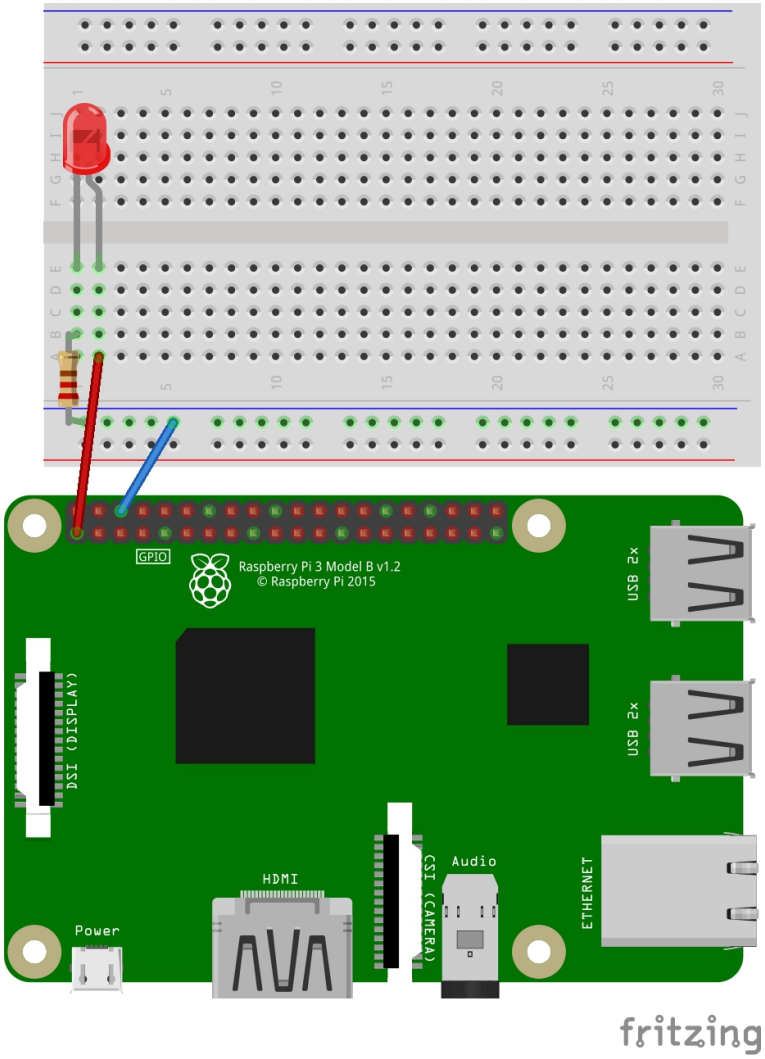

Figure J.1: วงจรเชื่อมต่อหลอด LED กับบอร์ด Pi3 ในการทดลองที่ 10 ที่มา: fritzing.org

4. ตรวจสอบความถูกต้อง โดยให้ผู้ควบคุมการทดลองตรวจสอบ
5. จ่ายไฟเลี้ยงให้กับบอร์ดแล้วสังเกตการเปลี่ยนแปลงที่หลอด LED

J.3 โปรแกรมไฟ LED กระพริบภาษา C

1. เรียกโปรแกรม Code::Blocks ผ่านทาง Terminal โดยใช้สิทธิ์ของ SuperUser ดังนี้

```
$ sudo codeblocks
```

2. สร้าง project ใหม่ชื่อ Lab10 จนเสร็จสิ้น
3. คลิกเมนู ”Setting/Compiler...” เลือก แท็บ ”Linker settings” แล้วกดปุ่ม ”Add”
4. ป้อนประโยค ”/usr/lib/libwiringPi.so;” ในหน้าต่าง Add Library แล้วกดปุ่ม ”OK” เพื่อปิดหน้าต่าง
5. กดปุ่ม ”OK” เพื่อยืนยัน
6. ป้อนโปรแกรมลงในไฟล์ใหม่ที่สร้างขึ้นโดยให้ชื่อว่า main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
int main ( void ) {
int pin = 7;
  printf("wiringPi LED blinking\n");
  if (wiringPiSetup() == -1) {
     printf( "Setup problem ... Abort!" );
     exit (1);
  }
  pinMode(pin, OUTPUT);
  int i;
  for ( i=0; i<10; i++ ) {
    digitalWrite(pin, 1);    /* LED On  */
    delay(250);
    digitalWrite(pin, 0);    /* LED Off */
    delay(250);
  }
  return 0;
}
```

7. ทำการ Build และแก้ไขหากมีข้อผิดพลาดจนสำเร็จ

8. Run และสังเกตการเปลี่ยนแปลงที่หลอดไฟ LED
9. จับเวลาช่วงเวลาที่หลอดสว่างและดับตั้งแต่เริ่มรันโปรแกรมจนเสร็จสิ้น เพื่อหาค่าเฉลี่ยของการสว่างดับ 1 รอบ

J.4 โปรแกรมไฟ LED กระพริบภาษา Assembly

1. เปิดโฟลเดอร์ **/home/pi/asm** ในโปรแกรมไฟล์เมเนเจอร์
2. สร้างโฟลเดอร์ใหม่ชื่อ **Lab10**
3. สร้างไฟล์ใหม่ชื่อ **Lab10.s** โดยใช้คำสั่ง **touch**
4. กรอกโปรแกรมภาษาแอสเซมบลีเหล่านี้ลงไป

```
    #--------------------
    # data segment
    #--------------------
         .data
         .balign 4
intro:   .asciz  "wiringPi LED blinking\n"
errMsg:  .asciz "Setup problem ... Abort!\n"
pin:     .int   7
i:       .int   0
duration:.int   250
OUTPUT   = 1    @constant
    #--------------------
    # text segment
    #--------------------
        .text
        .global main
        .extern printf
        .extern wiringPiSetup
        .extern delay
        .extern digitalWrite
        .extern pinMode

main:   PUSH    {ip, lr} @push link return register on stack segment
        LDR     R0, =intro
```

```
        BL      printf
        BL      wiringPiSetup
        MOV     R1,#-1
        CMP     R0, R1
        BNE     init
        LDR     R0, =errMsg
        BL      printf
        B       done
init:
        LDR     R0, =pin
        LDR     R0, [R0]
        MOV     R1,#OUTPUT
        BL      pinMode
        LDR     R4, =i
        LDR     R4, [R4]
        MOV     R5,#10
forLoop:
        CMP     R4, R5
        BGT     done
        LDR     R0, =pin
        LDR     R0, [R0]
        MOV     R1,#1
        BL      digitalWrite
        LDR     R0, =duration
        LDR     R0, [R0]
        BL      delay
        LDR     R0, =pin
        LDR     R0, [R0]
        MOV     R1,#0
        BL      digitalWrite
        LDR     R0, =duration
        LDR     R0, [R0]
        BL      delay
        ADD     R4,#1
        B       forLoop
done:
```

```
POP     {ip, pc} @pop return address into pc
```

5. ทำการแปลและลิงค์ Lab10.s จนกว่าจะสำเร็จ:

```
$   as -o Lab10.o Lab10.s
$   gcc -o Lab10 Lab10.o -lwiringPi
```

6. รันโปรแกรม Lab10 และสังเกตการเปลี่ยนแปลงที่หลอดไฟ LED

```
$  sudo ./Lab10
```

7. จับเวลาช่วงเวลาที่หลอดสว่างและดับตั้งแต่เริ่มรันโปรแกรมจนเสร็จสิ้น เพื่อหาค่าเฉลี่ยของการสว่างดับ 1 รอบ

J.5 กิจกรรมท้ายการทดลอง

1. สำรวจไฟล์ชื่อ wiringPi.c ในไดเรคทอรีชื่อ /home/pi/wiringPi/wiringPi/ เพื่อค้นหาตัวแปรชื่อ piGpioBase ว่า
   - ใช้งานในฟังค์ชันชื่ออะไร
   - ได้รับการตั้งค่าที่ฟังค์ชันชื่ออะไร และค่าเท่ากับเท่าไหร่
   - นำตัวแปร piGpioBase นี้ไปใช้ทำอะไรต่อได้อีก จงยกตัวอย่าง
   - หมายเลขแอดเดรส 0x2000_0000 นี้เกี่ยวข้องกับหมายเลข 0x7E00_0000 ในตารางที่ 2.4 และรูปที่ 2.16 อย่างไร
2. จงตอบคำถามจากประโยคต่อไปนี้

```
gpio = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE,
                        MAP_SHARED, fd, GPIO_BASE) ;
```

   - อยู่ในฟังค์ชันชื่ออะไร
   - ตัวแปร fd มาจากไหน เกี่ยวข้องกับ ไฟล์ /mem และไฟล์ /dev/gpiomem อย่างไร
   - ฟังค์ชัน mmap() มีหน้าที่อะไร รีเทิร์นค่าอะไรกลับมา และเป็นตัวแปรชนิดใด เหตุใดจึงต้องมีประโยค (uint32_t *) นำหน้า
   - นำตัวแปร gpio นี้ไปใช้ทำอะไรต่อได้อีก จงยกตัวอย่าง
   - จงอธิบายว่าตัวแปร gpio นี้เกี่ยวข้องกับหลักการ Memory Map IO อย่างไร
3. จงตอบคำถามจากประโยคต่อไปนี้

```
GPIO_BASE    = piGpioBase + 0x00200000 ;
```

- อยู่ในฟังค์ชันชื่ออะไร
- ตัวแปร GPIO_BASE มีหน้าที่อะไร
- เมื่อบวกแล้วได้ผลลัพธ์เป็นหมายเลขแอดเดรสอะไร และเกี่ยวข้องกับหมายเลข 0x7E20_0000 ในตารางที่ 2.6 อย่างไร
- นำตัวแปร GPIO_BASE นี้ไปใช้ทำอะไรต่อได้อีก จงยกตัวอย่าง
- จงอธิบายว่าตัวแปร GPIO_BASE นี้เกี่ยวข้องกับขา gpio แต่ละขาอย่างไร

4. ต่อหลอด LED เพิ่มอีก 2 ดวงรวมเป็น 3 ดวงแล้วพัฒนาโปรแกรมภาษา C เดิมให้นับเลข 0-7 และแสดงผลทางหลอด LED เป็นเลขฐานสองวนไปเรื่อยๆ
5. ใช้วงจรหลอด LED 3 ดวงที่มีอยู่และพัฒนาโปรแกรมภาษาแอสเซมบลีเดิมให้นับเลข 0-7 และแสดงผลทางหลอด LED เป็นเลขฐานสองวนไปเรื่อยๆ

1. - ใช้ในฟังก์ชัน **int wiringPiSetup (void)**
   - ตั้งค่าที่ฟังก์ชัน **int wiringPiSetup (void)**
     มีค่าเท่ากับ **GPIO_PERI_BASE_OLD, GPIO_PERI_BASE_NEW, 0**
   - ใช้ตั้งค่า offset เข้าไปใน memory interface
   - 0x7E00_0000 เป็น Bus address ที่ถูก map มาจาก 0x2000_0000 ที่เป็น Physical address

2. - อยู่ในฟังก์ชัน **int wiringPiSetup (void)**
   - อยู่ในฟังก์ชัน **int wiringPiSetup (void)** เก็บค่าที่ return จากฟังก์ชัน open() ที่เปิดไฟล์ใน /mem หรือ /dev/gpiomem จะมีค่าเป็นบวกแทนไฟล์นั้นๆ มีค่าเป็น -1 หากเปิดไม่ได้
   - mmap() จะ return pointer เข้าไปใน memory ที่เพิ่งถูก map เป็นตัวแปร pointer ส่วน (uint32_t *) มีไว้เพื่อแปลงชนิดตัวแปรให้เป็น uint32_t ตามรูปแบบ address ที่ต้องการ
   - นำไปใช้ map กับขาของ hardware
   - ใช้เก็บค่าที่ map เรียบร้อยแล้ว

3. - ฟังก์ชัน **int wiringPiSetup (void)**
   - เก็บค่า address ที่ใช้ในการ map Physical address กับ Bus address
   - บวกแล้วได้ 0x2020_0000 เมื่อ map จะได้ 0x7E20_0000 เป็น General Purpose I/O
   - ใช้ใน mmap() เช่น **gpio = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, GPIO_BASE);**
   - เป็น General Purpose I/O

4.

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
int main (void) {
    int led0 = 0;
    int led1 = 2;
    int led2 = 3;
    printf("wiringPi LED blinking\n");
    if(wiringPiSetup() == -1){
        printf("Setup problem ... Abort!");
        exit(1);
    }
    pinMode(led0, OUTPUT);
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    while(1){
    for(int i =0; i<8; i++){
        printf("%d %d %d %d\n",i,(i/4)%2,(i/2)%2,i%2);
        digitalWrite(led0, (i/4)%2);
        digitalWrite(led1, (i/2)%2);
        digitalWrite(led2,i%2);
        delay(1000);
    }
    printf("\n");}
    return 0;
}
```

5.

```
.data
    .balign 4
intro: .asciz "Start\n"
ending: .asciz "End\n"
waiting: .asciz "...\n"
errMsg: .asciz "Error\n"
addr: .word 2116026424
led0: .int 0
led1: .int 2
led2: .int 3
duration: .int 1000
OUTPUT = 1 @constant
    .text
    .global main
    .extern printf
    .extern wiringPiSetup
    .extern delay
    .extern digitalWrite
    .extern pinMode

main:
    PUSH {ip, lr}
    LDR R0, =intro
    BL printf
    BL wiringPiSetup
    CMP R0, #-1
    BNE init
    LDR R0, =errMsg
    BL printf
    B done
init:
    LDR R0, =led0
    LDR R0, [R0]
    MOV R1, #1
    BL pinMode
    LDR R0, =led1
    LDR R0, [R0]
    BL pinMode
    LDR R0, =led2
    LDR R0, [R0]
    BL pinMode
    MOV R0, #7
    MOV R1, #0
    BL pinMode
while:
    MOV R4, #0
for:
    CMP R4, #8
    BGE while

    LDR R0, =led0
    LDR R0, [R0]
    AND R1, R4, #4
    LSR R1, R1, #2
    BL digitalWrite
    LDR R0, =led1
    LDR R0, [R0]
    AND R1, R4, #2
    LSR R1, R1, #1
    BL digitalWrite
    LDR R0, =led2
    LDR R0, [R0]
    AND R1, R4, #1
    BL digitalWrite

    LDR R0, =waiting
    BL printf

    LDR R0, =duration
    LDR R0, [R0]
    BL delay
    ADD R4, R4, #1
    B for
done:
    LDR R0, =ending
    BL printf
    POP {ip,pc}
```