

ภาคผนวก J

การทดลองที่ 10 การเชื่อมต่อกับขา GPIO

การทดลองนี้คาดว่าผู้อ่านเคยศึกษาและทดลองเขียนหรือพัฒนาโปรแกรมด้วยภาษา C มาบ้างแล้ว และมีความคุ้นเคยกับ IDE (Integrated Development Environment) จากพัฒนาโปรแกรมและการดีบั๊กโปรแกรมด้วยภาษา C/C++ และแอสเซมบลี ดังนั้น การทดลองนี้มีวัตถุประสงค์เหล่านี้

- เพื่อปฏิบัติการเชื่อมต่อวงจรกับขา GPIO บนบอร์ด Pi ตามเนื้อหาในบทที่ 6 หัวข้อที่ 6.11
- เพื่อพัฒนาโปรแกรมภาษา C ควบคุมการทำงานของขา GPIO ด้วยไลบรารี wiringPi
- เพื่อพัฒนาโปรแกรมภาษาแอสเซมบลีควบคุมการทำงานของขา GPIO ด้วยไลบรารี wiringPi

โปรดสังเกตตัวอักษร w ที่คำว่า wiringPi ต้องเป็นตัวอักษรพิมพ์เล็ก

J.1 ไลบรารี wiringPi

ไลบรารี wiringPi รวบรวมฟังก์ชันที่พัฒนาด้วยภาษา C สำหรับบอร์ด Pi เป็น OpenSource ภายใต้ GNU LGPLv3 license สามารถเรียกใช้งานผ่าน ภาษา C and C++ รวมถึงภาษาแอสเซมบลี

เนื่องจาก ไลบรารี wiringPi เป็นซอฟต์แวร์แบบโอเพนซอร์ส แจกให้แก่นักพัฒนาทั่วโลกผ่านทาง <https://github.com/WiringPi> และมีการปรับปรุงแก้ไขตลอดเวลาโดยทีมนักพัฒนา ดังนั้น ผู้อ่านควรต้องติดตั้งและปรับปรุงระบบปฏิบัติการให้ทันสมัยและติดตั้งตามขั้นตอนต่อไปนี้

1. ผู้อ่านควรตรวจสอบว่าบอร์ดที่มีติดตั้งไลบรารี WiringPi แล้วหรือยัง โดยใช้คำสั่ง

```
$ gpio -v
```

2. หากบอร์ดยังไม่ได้ติดตั้งผู้อ่านควรปรับปรุงระบบปฏิบัติการให้เป็นปัจจุบันก่อน โดยพิมพ์คำสั่งนี้บนโปรแกรม Terminal โดยใช้สิทธิ์ของ SuperUser:

```
$ cd /tmp
```

```
$ wget https://unicorn.drogon.net/wiringpi-2.46-1.deb
```

ขั้นตอนนี้จะใช้เวลานานและความอดทน รวมถึงการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตที่มีเสถียรภาพ

- ติดตั้งด้วยไลบรารี wiringPi โดยพิมพ์คำสั่งนี้บน Terminal โดยใช้สิทธิ์ของ SuperUser:

```
$ sudo dpkg -i wiringpi-2.46-1.deb
```

คำสั่งนี้จะติดตั้งไลบรารีลงบนการ์ดหน่วยความจำ SD ในบอร์ด

- หากบอร์ดติดตั้งแล้ว คำสั่ง `gpio -v` ได้ผลลัพธ์ของการเรียกดังนี้

```
$ gpio -v
gpio version: 2.46
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: 1024 MiB, Maker: Sony
* Device tree is enabled.
*--> Raspberry Pi 3 Model B Rev 1.2
* This Raspberry Pi supports user-level GPIO access.
```

- เรียกคำสั่ง `gpio readall` เพื่อตรวจสอบและบันทึกผลลัพธ์ที่แสดงบนหน้าต่าง Terminal ลงในตารางหน้าถัดไป

```
$ gpio readall
```

- จดหมายเหตุหมายเลขในคอลัมน์ wPi (wiringPi) ให้ตรงกับขาเชื่อมต่อ 40 ขาบนบอร์ด Pi ตามที่แสดงบนหน้าจอลงในตารางต่อไปนี้ เพื่อใช้ประกอบการต่อวงจรที่ต้องการ

-----+-----+-----+-----+---Pi 3B---+-----+-----+-----+-----									
BCM	wPi	Name	V	Physical	V	Name	wPi	BCM	
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----									
		3.3v		1 2		5v			
2	<u>8</u>	SDA.1	1	3 4		5v			
3	<u>9</u>	SCL.1	1	5 6		0v			
4	<u>7</u>	GPIO. 7	1	7 8	0	TxD	<u>15</u>		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
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----									
BCM	wPi	Name	V	Physical	V		Name	wPi	BCM
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----									

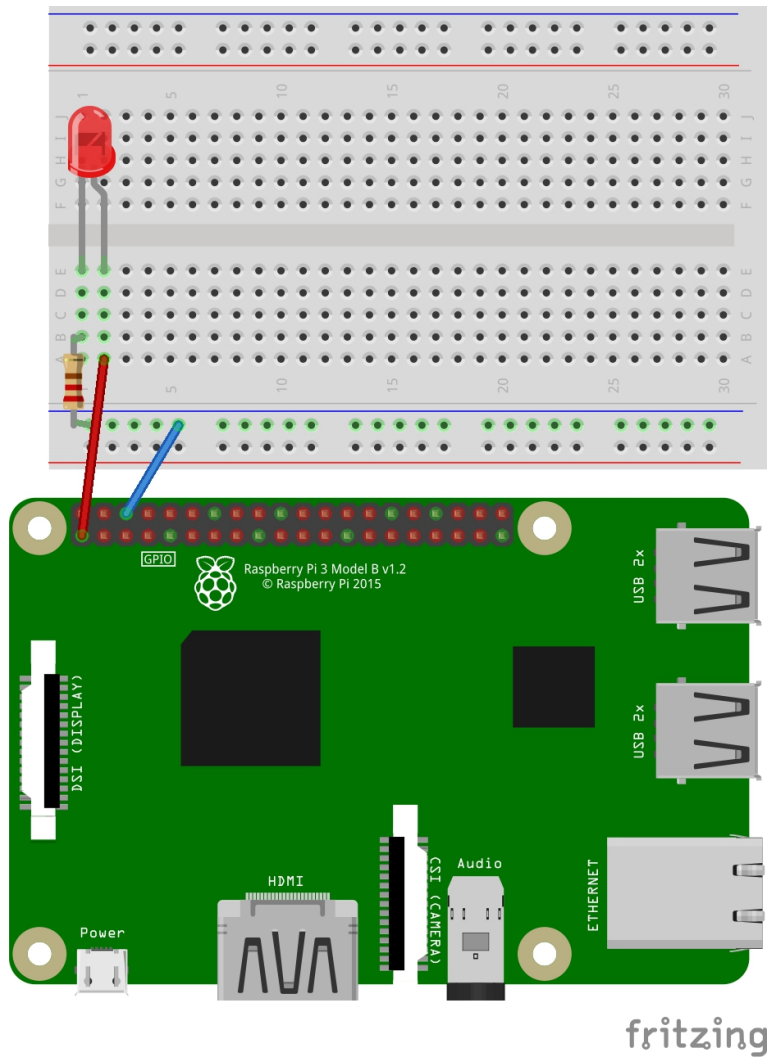
J.2 วงจรไฟ LED กระพริบ

1. รายการอุปกรณ์ที่ต้องใช้:

- หลอด LED จำนวน 1 หลอด
- ตัวต้านทาน (Resistor) ขนาด 2-10 กิโลโอห์ม จำนวน 1 ตัว
- แผ่นต่อวงจรโปรโตบอร์ด
- สายต่อวงจรชนิดต่าง: ผู้-เมีย (Male-Female) และ ผู้-ผู้ (Male-Male) จำนวนหนึ่ง

2. ชั้ตดาว์นและตัดไฟเลี้ยงออกจากบอร์ด Pi เพื่อความปลอดภัยในการต่อวงจร

3. ศึกษาตารางที่กรอกก่อนหน้าให้เข้าใจ แล้วจึงต่อวงจรตามรูปที่ [J.1](#)



รูปที่ J.1: วงจรเชื่อมต่อหลอด LED กับบอร์ด Pi ในการทดลองที่ 10 เพื่อทดสอบว่าหลอด LED ทำงาน
ที่มา: fritzing.org

4. จงวาดวงจรที่ต่อในรูปที่ J.1 ประกอบด้วย ตัวต้านทาน ไฟเลี้ยง 3.3 โวลต์ ขา LED และกราวด์ (0 โวลต์)
5. ตรวจสอบความถูกต้อง โดยให้ผู้ควบคุมการทดลองตรวจสอบ
6. จ่ายไฟเลี้ยงให้กับบอร์ดแล้วสังเกตการเปลี่ยนแปลงที่หลอด LED หากหลอด LED ไม่สว่าง ขอความช่วยเหลือจากผู้ควบคุมการทดลอง

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("LED blinking by wiringPi\n");
if (wiringPiSetup() == -1) {
printf( "Setting up problem ... Abort!" );
exit (1);
}
pinMode(pin, OUTPUT); /* set pin=7 to Output mode */
int i;
for ( i=0; i<10; i++ ) {
digitalWrite(pin, 1);    /* LED On */
delay(500);
digitalWrite(pin, 0);    /* LED Off */
delay(500);
}
return 0;
}
```

7. ทำการ Build และแก้ไขหากมีข้อผิดพลาดจนสำเร็จ
8. ย้ายสายจากขา 1 ของหัวเชื่อมต่อ 40 ขาไปยังขาหมายเลข 7 ซึ่งจะตรงกับ pin = 7 หรือ GPIO 7 ในตารางที่กรอกก่อนหน้านี้
9. Run และสังเกตการเปลี่ยนแปลงที่หลอดไฟ LED หากหลอด LED ไม่สว่าง ขอความช่วยเหลือจากผู้ควบคุมการทดลอง
10. จับเวลาช่วงเวลาที่หลอดสว่าง และนับตั้งแต่เริ่มรันโปรแกรมจนเสร็จสิ้น เพื่อหาค่าเฉลี่ยของการสว่างดับ 1 รอบ **ค่าเฉลี่ย 1 วิ หาก $\frac{20}{20} = 1$**

J.4 โปรแกรมไฟ LED กระพริบภาษาแอสเซมบลี

1. เปิดไดเรกทอรี /home/pi/asm ในโปรแกรมไฟล์เมเนเจอร์
2. สร้างไดเรกทอรีใหม่ชื่อ Lab10
3. สร้างไฟล์ใหม่ชื่อ Lab10.s โดยใช้คำสั่ง touch
4. กรอกโปรแกรมภาษาแอสเซมบลีเหล่านี้โดยใช้ editor ที่ถนัด

```
#-----
# data segment
#-----

.data
.balign 4
intro: .asciz "LED blinking by wiringPi\n"
errMsg: .asciz "Setting up problem ... Abort!\n"
pin: .int 7
i: .int 0
duration: .int 500
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. ทำการแปลและลิงก์ Lab10.s กับด้วยไลบรารี wiringPi จนกว่าจะสำเร็จ:

```

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

```

6. รันโปรแกรม Lab10 ด้วยสิทธิ์ของ SuperUser และสังเกตการเปลี่ยนแปลงที่หลอดไฟ LED

```
$ sudo ./Lab10
```

7. จับเวลาช่วงเวลาที่หลอดสว่างและดับ ตั้งแต่เริ่มรันโปรแกรมจนเสร็จสิ้น เพื่อหาค่าเฉลี่ยของการสว่างดับ 1 รอบ **ค่าเฉลี่ย 1 วิ จาก $\frac{20}{20} = 1$**

J.5 กิจกรรมท้ายการทดลอง

- ไลบรารี libwiringPi.so ทำหน้าที่อะไร และเกี่ยวข้องกับ #include <wiringPi.h> อย่างไร
- ประโยค \$ gcc -o Lab10 Lab10.o -lwiringPi มีความหมายอย่างไร และเชื่อมโยงกับคำถามข้อที่แล้วอย่างไร
- ฟังก์ชัน digitalWrite ใช้กับขา GPIO ในโหมดไหน **โหมด GPIO ในโหมด output**
- ประโยค PUSH {ip, lr} ทำหน้าที่อะไร เหตุใดจึงต้องเรียกใช้ก่อนประโยคอื่น ๆ **ทำหน้าที่เก็บ link register เอาไว้ก่อนจะถูกเขียนทับ ในกรณีที่จะกลับมาที่ตำแหน่งเดิม**
- ประโยค POP {ip, pc} ทำหน้าที่อะไร เหตุใดจึงต้องเรียกใช้เป็นประโยคสุดท้าย
- คลิก บน ลิงก์ ชื่อ <https://github.com/WiringPi/WiringPi/blob/master/wiringPi/wiringPi.c> เพื่อใช้เบราว์เซอร์เปิดและตอบคำถามต่อไปนี้
- สำรวจไฟล์ชื่อ wiringPi.c ที่เปิดเพื่อค้นหาตัวแปรชื่อ piGpioBase ว่า **ทำหน้าที่ load ค่า link register ที่เก็บ push ไว้ในบรรทัดแรกเพื่อหามาเก็บไว้ใน pc**

- ใช้งานในฟังก์ชันชื่ออะไร
- ได้รับการตั้งค่าที่ฟังก์ชันชื่ออะไร และค่าเท่ากับเท่าไร
- นำตัวแปร piGpioBase นี้ไปใช้ทำอะไรต่อได้อีก จงยกตัวอย่าง
- หมายเลขแอดเดรส 0x2000 0000 นี้เกี่ยวข้องกับ หมายเลข 0x7E00 0000 ใน ตารางที่ 6.4 และรูปที่ 6.16 อย่างไร

8. จงค้นหาประโยคและตอบคำถามต่อไปนี้

```
gpio = (uint32\_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE,
                        MAP_SHARED, fd, GPIO_BASE) ;
```

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

9. จงตอบคำถามจากประโยคต่อไปนี้

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

- อยู่ในฟังก์ชันชื่ออะไร
- ตัวแปร GPIO_BASE มีหน้าที่อะไร
- เมื่อ บวก แล้ว ได้ ผลลัพธ์ เป็น หมายเลข แอดเดรส อะไร และ เกี่ยวข้อง กับ หมายเลข 0x7E20 0000 ในตารางที่ 6.6 อย่างไร
- นำตัวแปร GPIO_BASE นี้ไปใช้ทำอะไรต่อได้อีก จงยกตัวอย่าง
- จงอธิบายว่าตัวแปร GPIO_BASE นี้เกี่ยวข้องกับขา gpio แต่ละขาอย่างไร

10. ต่อหลอด LED เพิ่มอีก 2 ดวงรวมเป็น 3 ดวงแล้วพัฒนาโปรแกรมภาษา C เดิมให้นับเลขจำนวนเต็มฐานสิบ 0 - 7 และแสดงผลทางหลอด LED เป็นเลขฐานสองวนไปเรื่อย ๆ
11. ใช้วงจรหลอด LED 3 ดวงที่มีอยู่และพัฒนาโปรแกรมภาษาแอสเซมบลีเดิมนับเลขจำนวนเต็มฐานสิบ 0 - 7 และแสดงผลทางหลอด LED เป็นเลขฐานสองวนไปเรื่อย ๆ