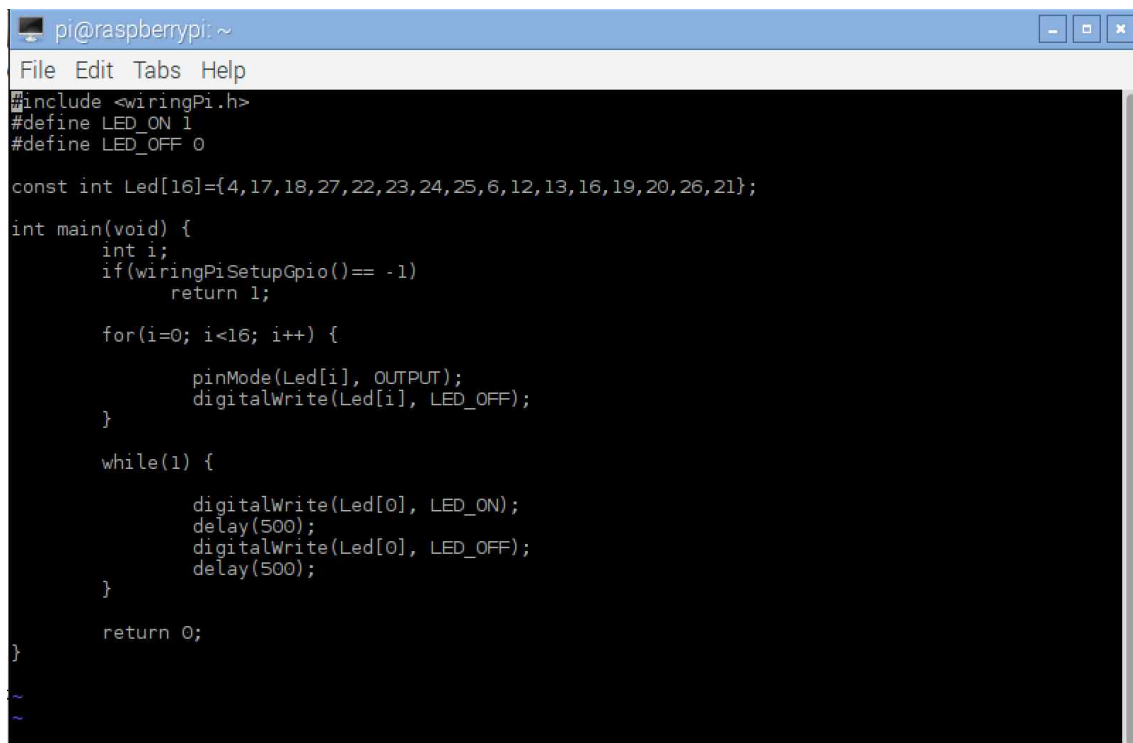


예제 1)

1번 LED를 500ms마다 켜다 끄는 예제입니다. Raspberry Pi Adapter에 LED모듈을 연결해주었고 Raspberry Pi Adapter에 대응 하는 핀 번호를 Led 배열에 넣어 주었습니다. wiringPiSetup()은 GPIO 포트를 wiringPi pin 번호로 매핑해서 사용하도록 초기화 하고, wiringPiSetupGpio()는 매핑하지 않고 GPIO 포트를 그대로 사용하도록 초기화 합니다.

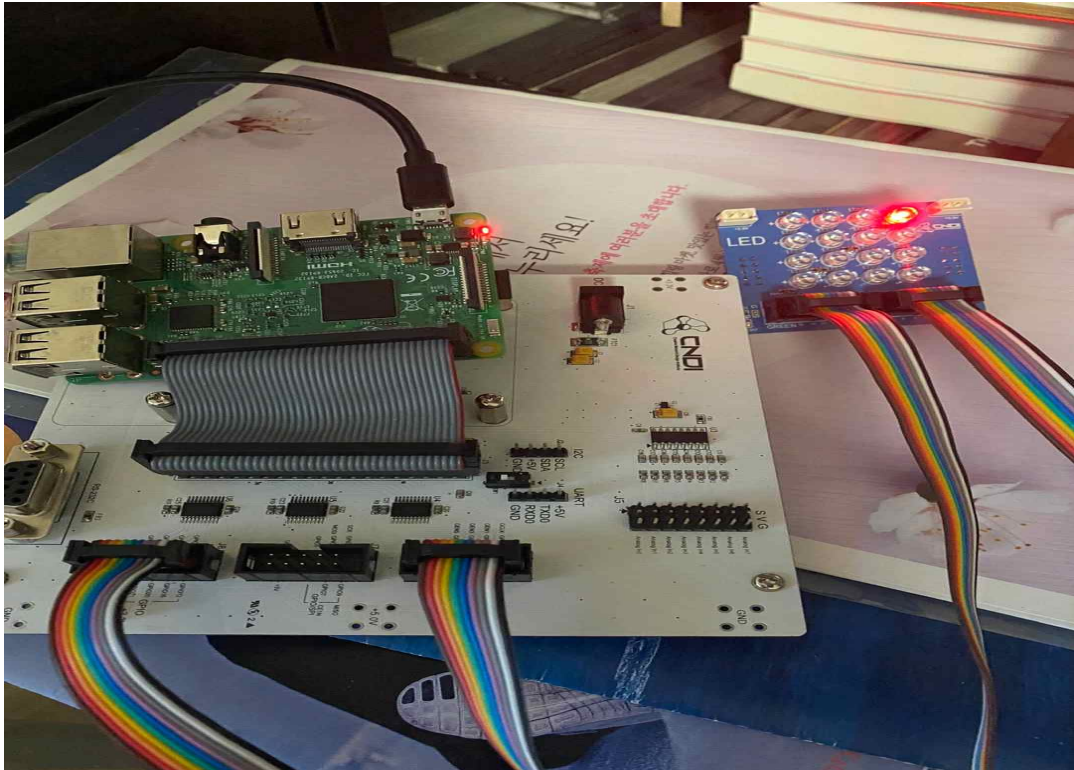
아두이노의 setup에서 while문으로 들어가기 전에 핀을 설정하는 것처럼 라즈베리파이에서도 while문 안으로 들어가기 전에 Led 핀들을 모두 Output을 설정해주었습니다. While문 안에서는 아두이노에서처럼 500ms간격으로 Led를 켜다 끄는 것을 반복해주었습니다.

그 후 GPIO 4번 핀에 대응하는 첫 번째 Led인 1번 Led에서 불이 깜빡이는 것을 확인할 수 있었습니다.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
#include <wiringPi.h>  
#define LED_ON 1  
#define LED_OFF 0  
  
const int Led[16]={4,17,18,27,22,23,24,25,6,12,13,16,19,20,26,21};  
  
int main(void) {  
    int i;  
    if(wiringPiSetupGpio()== -1)  
        return 1;  
  
    for(i=0; i<16; i++) {  
        pinMode(Led[i], OUTPUT);  
        digitalWrite(Led[i], LED_OFF);  
    }  
  
    while(1) {  
        digitalWrite(Led[0], LED_ON);  
        delay(500);  
        digitalWrite(Led[0], LED_OFF);  
        delay(500);  
    }  
  
    return 0;  
}
```

<vi에디터를 이용해서 작성한 lab2-2_1.c>



<1번 Led가 켜다 켜지는 모습>

예제2)

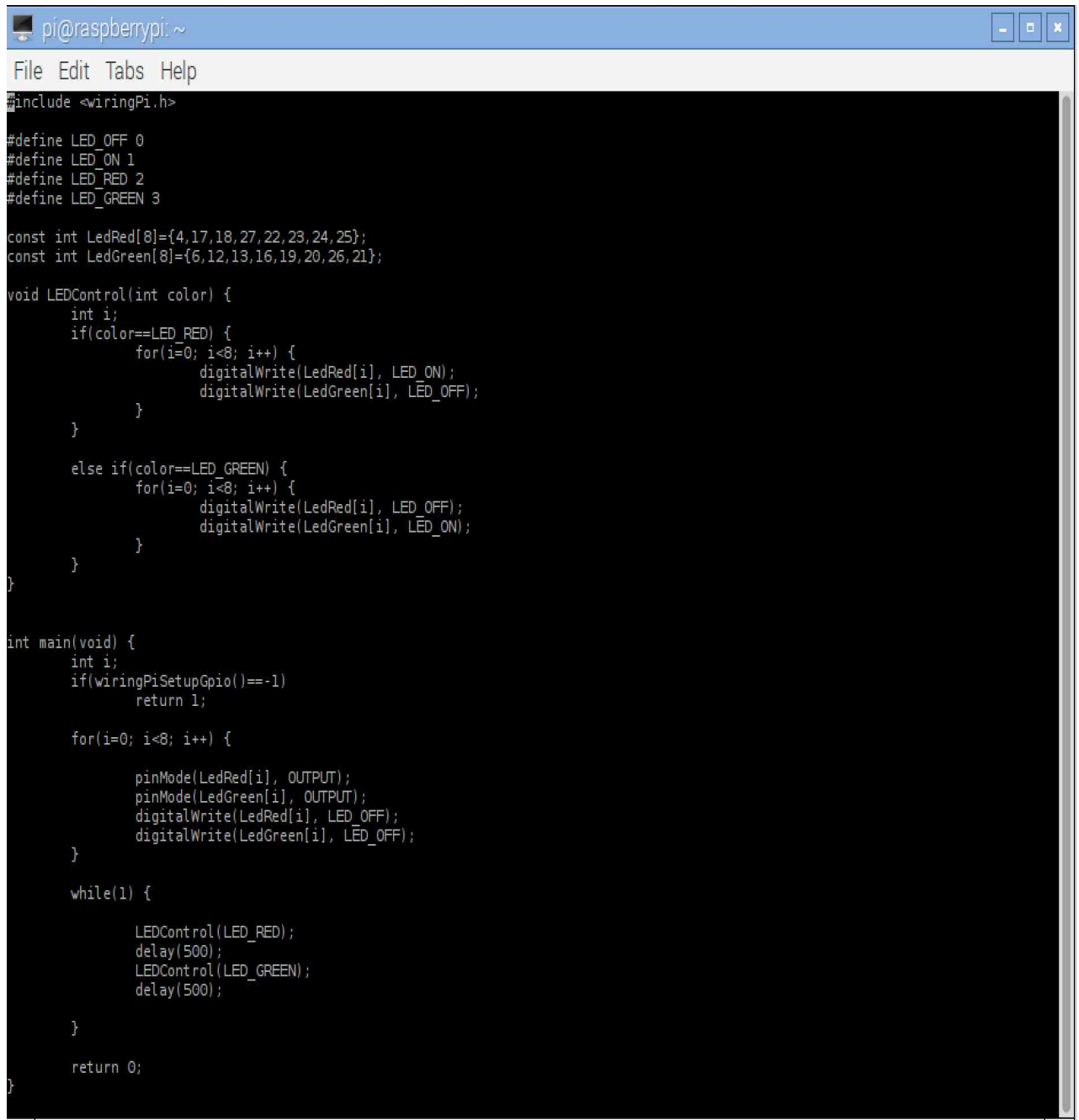
빨간색 LED와 초록색 LED를 500ms마다 번갈아 켜다가 끄는 예제입니다.

그리고 1번~8번 Led에 해당하는 LedRed는 Raspberry Pi Adapter에 적힌 GPIO에 해당하는 번호로 인덱스에 값을 넣어주었고 9~16번 Led에 해당하는 LedGreen도 마찬가지로 설정 해주었습니다.

LedControl 함수에서는 parameter 변수로 color를 입력받습니다. color RED, GREEN 둘 중의 하나로 LED_RED값인 2가 들어가면 0~7번 인덱스의 Red값이 모두 켜지고 Green 값은 모두 꺼집니다. 반대로 LED_GREEN값인 3이 들어가면 0~7번 인덱스의 Green 값이 모두 켜지고 Red값이 모두 꺼집니다.

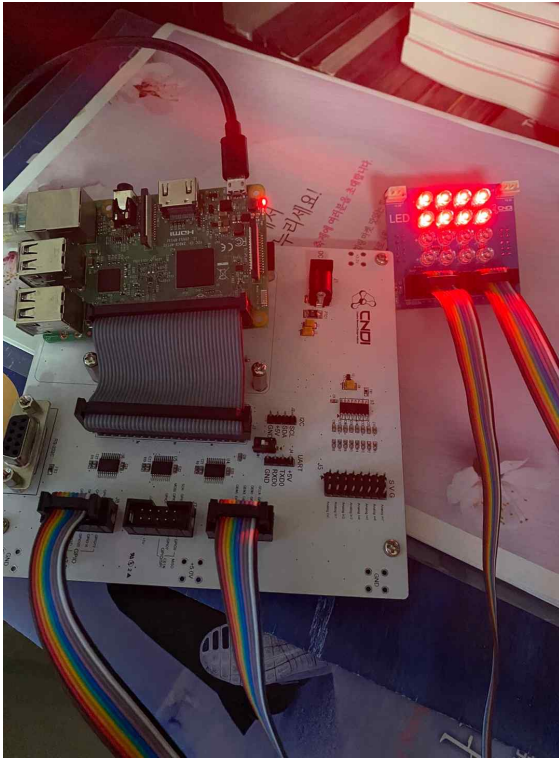
예제 1과 마찬가지로 while문 안으로 들어가기 전에 Led 핀들을 모두 Output을 설정해주었습니다. While문 안에서는 LedControl 함수를 500ms 간격으로 한 번은 빨간색, 한 번은 초록색이 설정되도록 입력해 주었습니다.

컴파일 후 실행 결과 1~8번 Red LED와 9~16번 Green LED가 번갈아서 켜다 켜지는 것을 확인할 수 있었습니다.

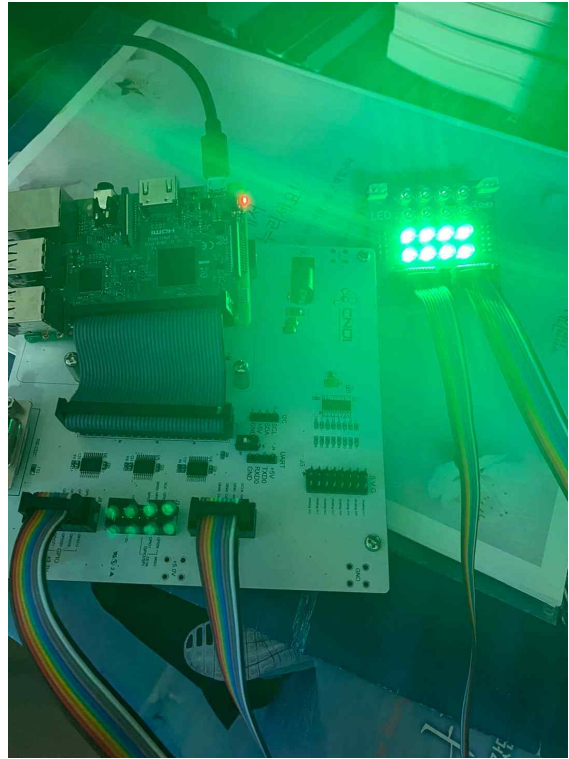


```
pi@raspberrypi: ~  
File Edit Tabs Help  
#include <wiringPi.h>  
  
#define LED_OFF 0  
#define LED_ON 1  
#define LED_RED 2  
#define LED_GREEN 3  
  
const int LedRed[8]={4,17,18,27,22,23,24,25};  
const int LedGreen[8]={6,12,13,16,19,20,26,21};  
  
void LEDControl(int color) {  
    int i;  
    if(color==LED_RED) {  
        for(i=0; i<8; i++) {  
            digitalWrite(LedRed[i], LED_ON);  
            digitalWrite(LedGreen[i], LED_OFF);  
        }  
    }  
  
    else if(color==LED_GREEN) {  
        for(i=0; i<8; i++) {  
            digitalWrite(LedRed[i], LED_OFF);  
            digitalWrite(LedGreen[i], LED_ON);  
        }  
    }  
}  
  
int main(void) {  
    int i;  
    if(wiringPiSetupGpio()==-1)  
        return 1;  
  
    for(i=0; i<8; i++) {  
        pinMode(LedRed[i], OUTPUT);  
        pinMode(LedGreen[i], OUTPUT);  
        digitalWrite(LedRed[i], LED_OFF);  
        digitalWrite(LedGreen[i], LED_OFF);  
    }  
  
    while(1) {  
        LEDControl(LED_RED);  
        delay(500);  
        LEDControl(LED_GREEN);  
        delay(500);  
    }  
  
    return 0;  
}
```

<vi에디터를 이용해서 작성한 lab2-2_2.c>



< RED LED ON >



< GREEN LED ON >

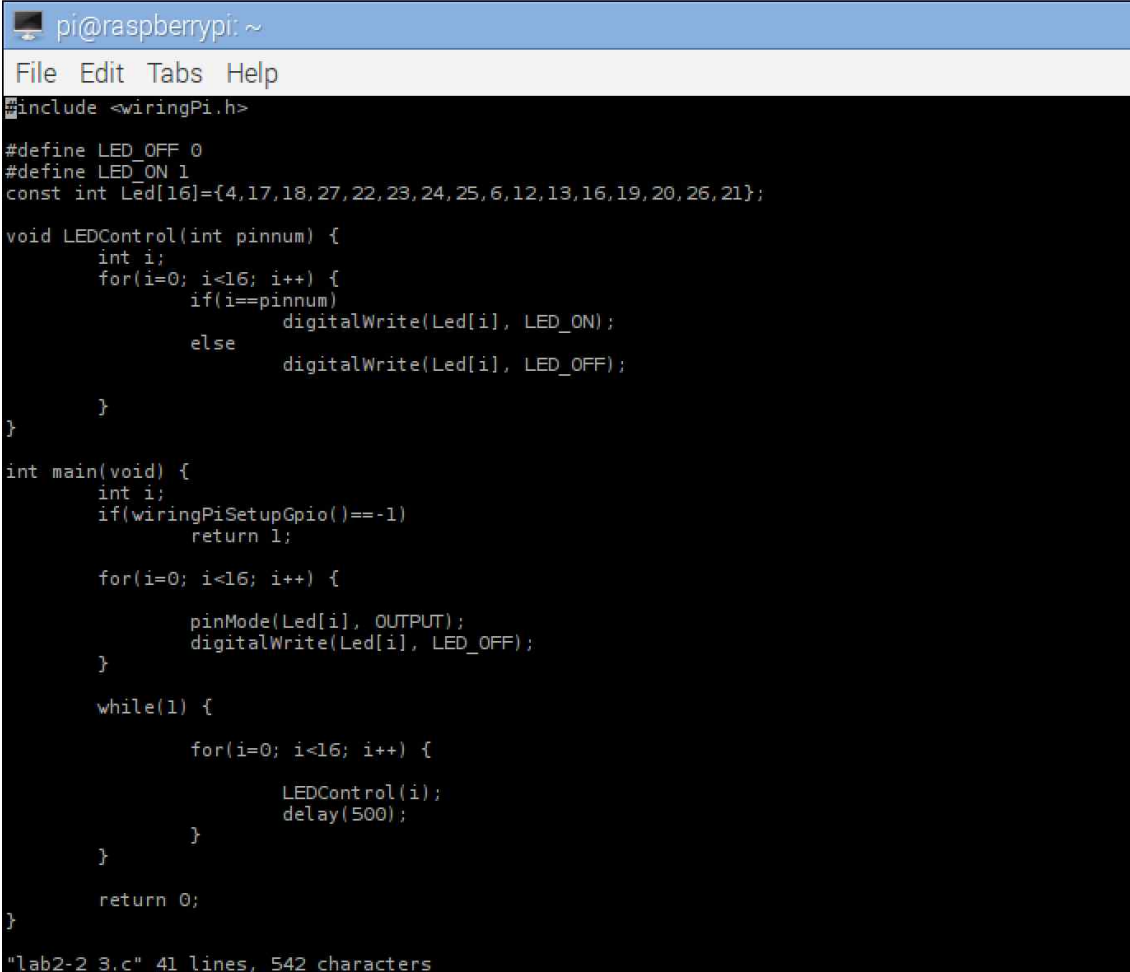
예제3)

LED 1부터 16까지 순서대로 500ms간격으로 켜다 끄는 예제입니다. 예제 1처럼 Raspberry Pi Adapter에 LED모듈을 연결해주었고 Raspberry Pi Adapter에 대응 하는 핀 번호를 Led 배열에 넣어 주었습니다.

LED Control 함수는 pinnum을 변수로 동작하는 함수입니다. 만일 pinnum에 해당하는 인덱스의 GPIO가 있을 시 그 GPIO output에는 불이 켜지고 해당 GPIO가 아닌 output출력은 모두 끄는 간단한 함수입니다. 예제 1, 2번처럼 해당 GPIO를 output으로 셋팅하고 while문 안에서는 인덱스 순서대로 LED Control 함수를 동작시켰습니다.

컴파일 후 실행 결과 1번부터 16번까지의 LED가 자기 혼자만 단독으로 순서대로 켜지는 것을 반복하는 것을 확인할 수 있었습니다.

기록 영상은 폴더에 첨부하겠습니다.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
#include <wiringPi.h>  
  
#define LED_OFF 0  
#define LED_ON 1  
const int Led[16]={4,17,18,27,22,23,24,25,6,12,13,16,19,20,26,21};  
  
void LEDControl(int pinnum) {  
    int i;  
    for(i=0; i<16; i++) {  
        if(i==pinnum)  
            digitalWrite(Led[i], LED_ON);  
        else  
            digitalWrite(Led[i], LED_OFF);  
    }  
}  
  
int main(void) {  
    int i;  
    if(wiringPiSetupGpio() == -1)  
        return 1;  
  
    for(i=0; i<16; i++) {  
        pinMode(Led[i], OUTPUT);  
        digitalWrite(Led[i], LED_OFF);  
    }  
  
    while(1) {  
        for(i=0; i<16; i++) {  
            LEDControl(i);  
            delay(500);  
        }  
    }  
  
    return 0;  
}  
"lab2-2_3.c" 41 lines, 542 characters
```

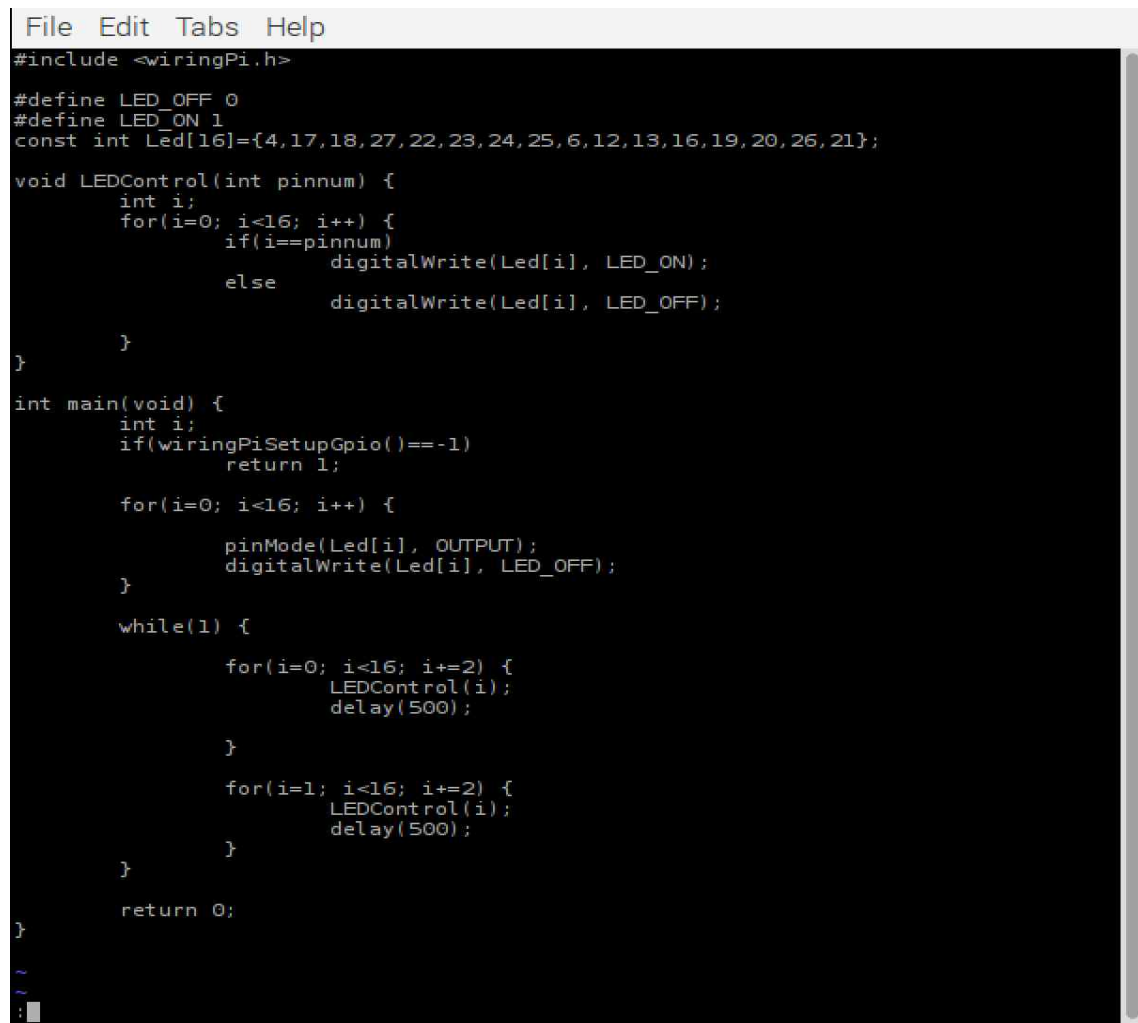
<vi에디터를 이용해서 작성한 lab2-2_3.c>

과제1)

홀수 번 LED 순서대로 켜다가 짝수 번 LED를 순서대로 키는 것을 반복하는 예제입니다. LED Control 함수는 pinnum을 변수로 동작하는 함수입니다. 예제 3번과 동작이 똑같은 함수입니다. 저는 while문에서의 동작을 제어하였습니다. 먼저 i=0일 때는 1번 LED가 켜지는 것이므로 1번 led가 켜질 때 인덱스를 2씩 더해서 홀수 번 LED가 켜지도록 하였습니다. 또한 i=1일 때는 2번 LED가 켜지는 것이므로 이때도 마찬가지로 인덱스를 2씩 더하였습니다. 이렇게 While문안에서 두 개의 for문을 돌려서 홀수 및 짝수 번 Led를 순서대로 켤 수 있었습니다.

컴파일 후 실행 결과 홀수 번 LED 이후 짝수 번 LED가 순서대로 켜지는 것을 확인할 수 있었습니다.

기록 영상은 폴더에 첨부하겠습니다.



```
File Edit Tabs Help
#include <wiringPi.h>

#define LED_OFF 0
#define LED_ON 1
const int Led[16]={4,17,18,27,22,23,24,25,6,12,13,16,19,20,26,21};

void LEDControl(int pinnum) {
    int i;
    for(i=0; i<16; i++) {
        if(i==pinnum)
            digitalWrite(Led[i], LED_ON);
        else
            digitalWrite(Led[i], LED_OFF);
    }
}

int main(void) {
    int i;
    if(wiringPiSetupGpio()==-1)
        return 1;

    for(i=0; i<16; i++) {
        pinMode(Led[i], OUTPUT);
        digitalWrite(Led[i], LED_OFF);
    }

    while(1) {
        for(i=0; i<16; i+=2) {
            LEDControl(i);
            delay(500);
        }

        for(i=1; i<16; i+=2) {
            LEDControl(i);
            delay(500);
        }
    }

    return 0;
}
```

<vi에디터를 이용해서 작성한 ass2-2_1.c>

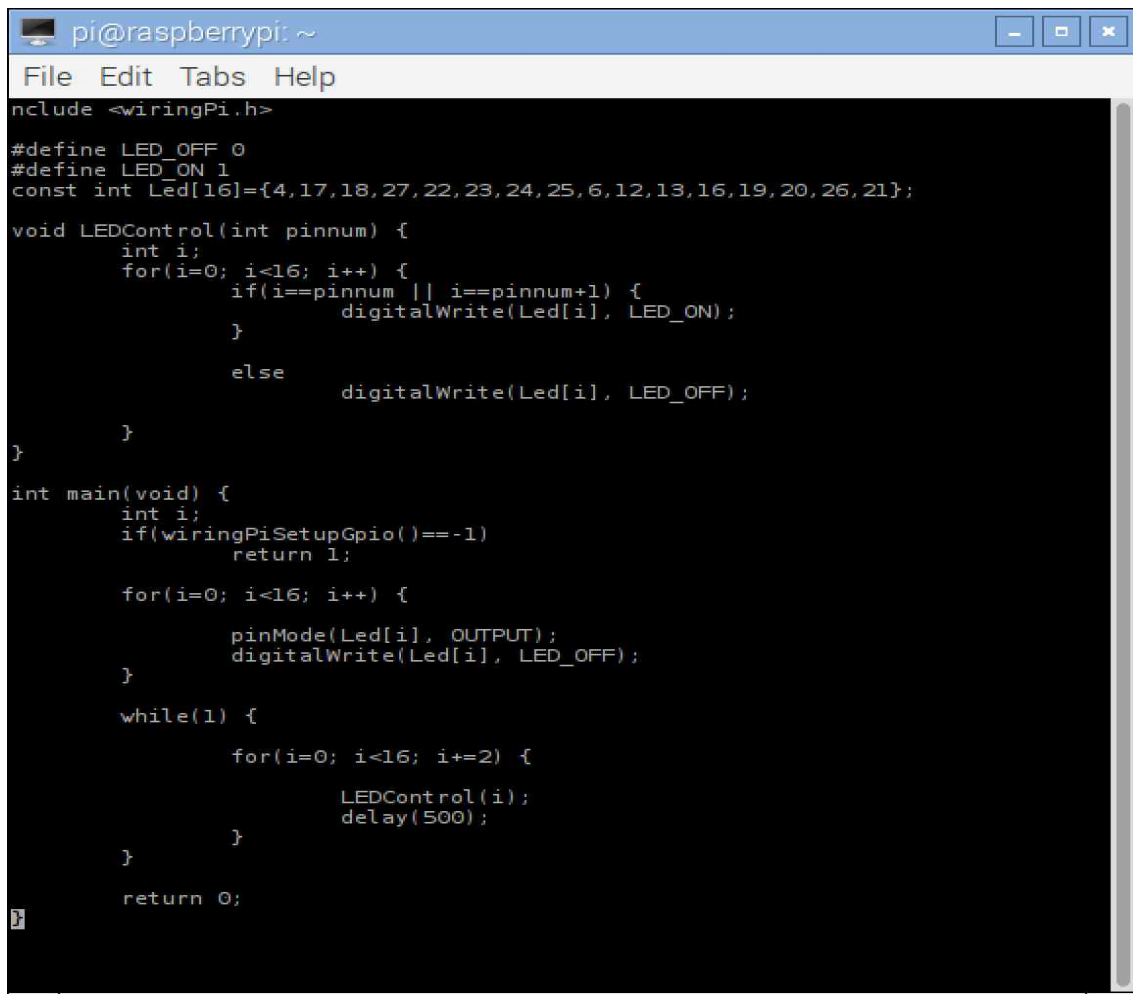
과제2)

LED를 두 개 씩 순서대로 키는 것을 반복하는 예제입니다. 즉 1+2번 LED, 3+4번 LED ... 15+16번 LED가 켜져야 합니다.

LED Control 함수는 pinnum을 변수로 동작하는 함수입니다. 과제 1번과 다르게 해당 pinnum이 설정되면 그 pin number의 1을 더한 값의 led도 불이 켜지게 하였습니다. 그리고 2+3 LED 등의 LED가 켜지지 못하도록 즉, 오로지 1+2, 3+4 ,5+6번 LED만이 켜지도록 While문 안에서 인덱스를 2씩 더해 주었습니다.

컴파일 후 실행 결과 1+2번 LED, 3+4번 LED ... 15+16번 LED가 켜지는 것을 확인할 수 있었습니다.

기록 영상은 폴더에 첨부하겠습니다.

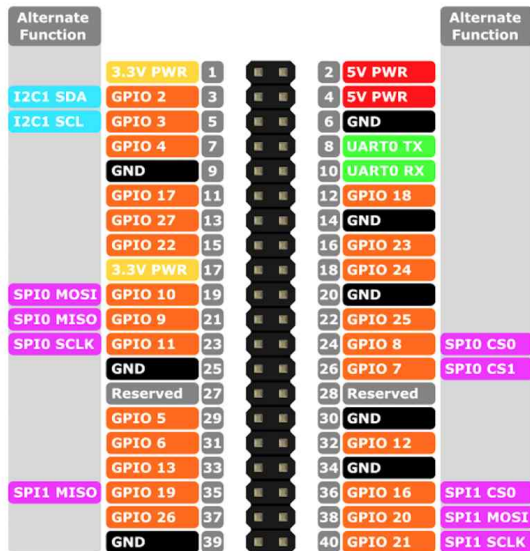


```
pi@raspberrypi: ~  
File Edit Tabs Help  
#include <wiringPi.h>  
#define LED_OFF 0  
#define LED_ON 1  
const int Led[16]={4,17,18,27,22,23,24,25,6,12,13,16,19,20,26,21};  
  
void LEDControl(int pinnum) {  
    int i;  
    for(i=0; i<16; i++) {  
        if(i==pinnum || i==pinnum+1) {  
            digitalWrite(Led[i], LED_ON);  
        }  
        else  
            digitalWrite(Led[i], LED_OFF);  
    }  
}  
  
int main(void) {  
    int i;  
    if(wiringPiSetupGpio()==-1)  
        return 1;  
  
    for(i=0; i<16; i++) {  
        pinMode(Led[i], OUTPUT);  
        digitalWrite(Led[i], LED_OFF);  
    }  
  
    while(1) {  
        for(i=0; i<16; i+=2) {  
            LEDControl(i);  
            delay(500);  
        }  
    }  
  
    return 0;  
}
```

<vi에디터를 이용해서 작성한 ass2-2_2.c>

과제3)

WiringPi 라이브러리를 사용하지 않고 mmap을 사용하여 레지스터에 접근하여 Led를 제어하는 실습입니다. GPIO하나의 출력만 제어하면 되므로 Raspberry Pi 18번 핀을 Led모듈의 1번에 연결해 주었습니다.



BCM2837 데이터 시트에 따르면 LED를 켜고 끄는 데에는 다음 레지스터들을 사용하면 됩니다.

GPFSSELx - GPIO핀을 Input 또는 Output으로 사용할지 결정.

GPSETx - GPIO핀을 하이레벨로 만들어줌. Led가 켜진 상태.

GPCLR x - GPIO핀을 로우 레벨로 만들어줌. Led가 꺼지는 상태.

Address	Field Name	Description	Size	Read/Write
0x 7E20 0000	GPFSSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0018	-	Reserved	-	-
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 0024	-	Reserved	-	-
0x 7E20 0028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 002C	GPCLR1	GPIO Pin Output Clear 1	32	W

위 사진은 LED를 제어하는데 사용할 3가지 레지스터의 메모리 주소들을 보여주고 있습니다.

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL19	FSEL19 - Function Select 19 000 = GPIO Pin 19 is an input 001 = GPIO Pin 19 is an output 100 = GPIO Pin 19 takes alternate function 0 101 = GPIO Pin 19 takes alternate function 1 110 = GPIO Pin 19 takes alternate function 2 111 = GPIO Pin 19 takes alternate function 3 011 = GPIO Pin 19 takes alternate function 4 010 = GPIO Pin 19 takes alternate function 5	R/W	0
26-24	FSEL18	FSEL18 - Function Select 18	R/W	0
23-21	FSEL17	FSEL17 - Function Select 17	R/W	0
20-18	FSEL16	FSEL16 - Function Select 16	R/W	0
17-15	FSEL15	FSEL15 - Function Select 15	R/W	0
14-12	FSEL14	FSEL14 - Function Select 14	R/W	0
11-9	FSEL13	FSEL13 - Function Select 13	R/W	0
8-6	FSEL12	FSEL12 - Function Select 12	R/W	0
5-3	FSEL11	FSEL11 - Function Select 11	R/W	0
2-0	FSEL10	FSEL10 - Function Select 10	R/W	0

Table 6-3 – GPIO Alternate function select register 1

GPIO18번은 GPF_SEL1레지스터에 해당 항목이 있습니다.

24번째 비트부터 26번째 비트까지의 값을 001로 바꾸어주면 GPIO18번은 출력모드가 되게 됩니다.

그리고 GPIO18번의 위치에 해당되는 비트를 세팅 해주면 LED가 켜지고 꺼지게 할 수 있습니다.

각 라이브러리를 추가시켜준 후 gpio18번에 해당하는 레지스터를 셋팅 하였습니다. 그 후 컴파일 후 실행하였지만 입력이 잘 되지 않아 관리자 명령으로 실행하였습니다. LED가 5번 켜다 꺼지는 것을 확인할 수 있었습니다.

```

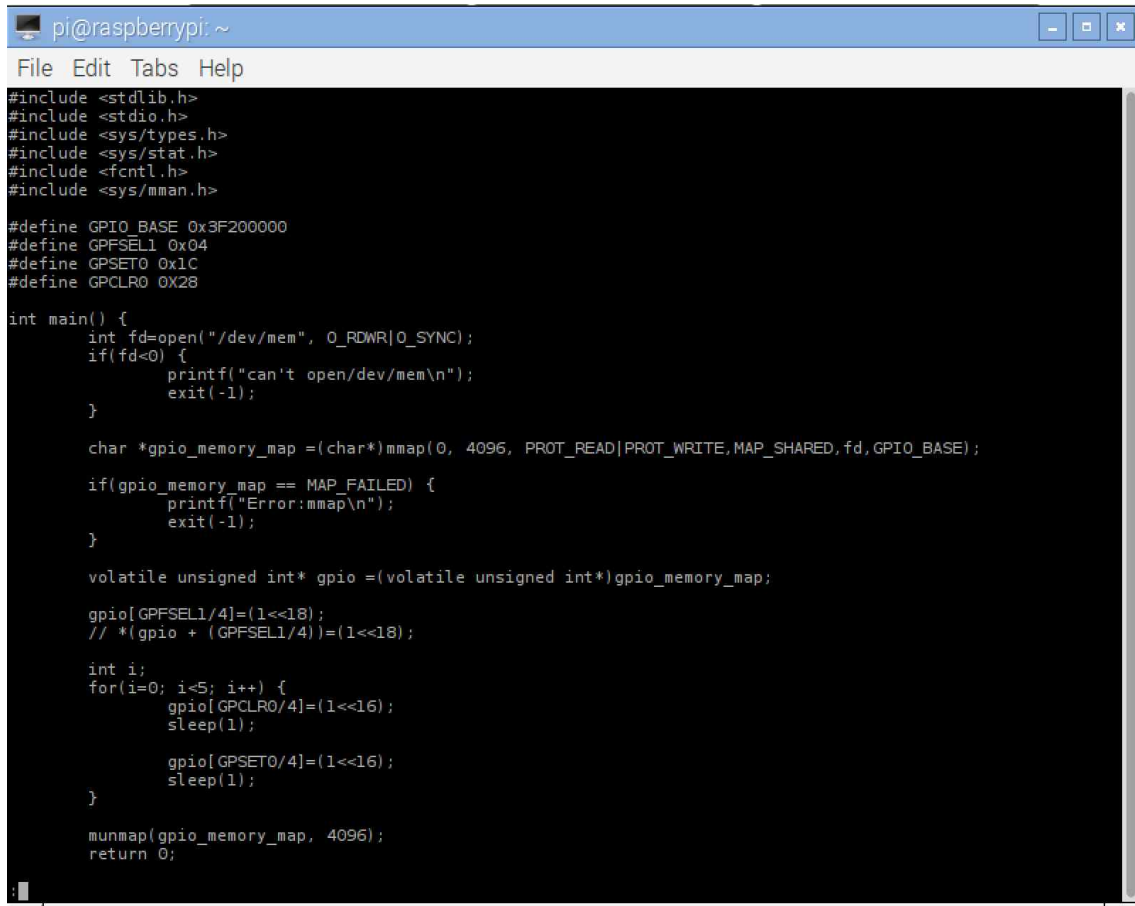
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ gcc ass2-2_3.c -o ass2-2_3
pi@raspberrypi:~ $ sudo ./ass2-2_3
pi@raspberrypi:~ $ vi ass2-2_4.c
pi@raspberrypi:~ $ vi ass2-2_4.c
pi@raspberrypi:~ $ gcc ass2-2_4.c -o ass2-2_4
pi@raspberrypi:~ $ sudo ./ass2-2_4
pi@raspberrypi:~ $ sudo ./ass2-2_4
pi@raspberrypi:~ $ sudo ./ass2-2_4
pi@raspberrypi:~ $

```

<3번 과제와 4번과제를 sudo 명령어로 실행한 터미널 창>

과제4)

과제 4는 gpio를 18번에서 16번으로 바꿔주기만 하면 되었습니다. GPIO 18번일 때는 24~26을 001로 하였지만 GPIO 16일 때는 18~20을 셋팅해 주면 되었습니다. 그리고 18번 일 때 와 16번 일때의 GPFSEL은 동일하기 GPFSEL1번이므로 동일하게 설정하였고 18번 레지스터를 제어하는 것을 16번으로 바꿔주면 되었습니다.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
#include <stdlib.h>  
#include <stdio.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <sys/mman.h>  
  
#define GPIO_BASE 0x3F200000  
#define GPFSEL1 0x04  
#define GPSET0 0x1C  
#define GPCLR0 0X28  
  
int main() {  
    int fd=open("/dev/mem", O_RDWR|O_SYNC);  
    if(fd<0) {  
        printf("can't open/dev/mem\n");  
        exit(-1);  
    }  
  
    char *gpio_memory_map =(char*)mmap(0, 4096, PROT_READ|PROT_WRITE,MAP_SHARED,fd,GPIO_BASE);  
  
    if(gpio_memory_map == MAP_FAILED) {  
        printf("Error:mmap\n");  
        exit(-1);  
    }  
  
    volatile unsigned int* gpio =(volatile unsigned int*)gpio_memory_map;  
  
    gpio[GPFSEL1/4]=(1<<18);  
    // *(gpio + (GPFSEL1/4))=(1<<18);  
  
    int i;  
    for(i=0; i<5; i++) {  
        gpio[GPCLR0/4]=(1<<16);  
        sleep(1);  
  
        gpio[GPSET0/4]=(1<<16);  
        sleep(1);  
    }  
  
    munmap(gpio_memory_map, 4096);  
    return 0;  
}
```

<vi에디터를 이용해서 작성한 ass2-2_4.c>

3번 및 4번과제의 영상은 폴더에 첨부하겠습니다.