

## 임베디드 시스템 실습 lab4

2016310936 우승민

우선 mini\_calc\_mmap.c의 코드를 설명하겠습니다. 기존의 skeleton 코드를 특별히 변경하지 않고 사용하였고, 인자의 개수에 따라 if문을 실행하게 만들어 주었습니다.

```
int main(int argc, char *argv[])
{
    int i, fd;
    unsigned *seg_base_addr;
    unsigned *seg0, *seg1, *seg2, *seg3, *seg4, *seg5, *seg6;

    fd = open("/dev/segment", O_RDWR);
    seg_base_addr = (unsigned *)mmap(NULL, SEGMENT_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, SEGMENT_ADDR);

    seg0 = (unsigned *)(seg_base_addr + 0);
    seg1 = (unsigned *)(seg_base_addr + 1);
    seg2 = (unsigned *)(seg_base_addr + 2);
    seg3 = (unsigned *)(seg_base_addr + 3);
    seg4 = (unsigned *)(seg_base_addr + 4);
    seg5 = (unsigned *)(seg_base_addr + 5);
    seg6 = (unsigned *)(seg_base_addr + 6);

    /* This program turn on all segments (0)*/

    if(argc==2){
        int x = atoi(argv[1]);
        if(x>9999999) x = 9999999;
        if(x<0) x = 0;
        int k=0;
        while(x){
            *(unsigned *)(seg_base_addr + 6 - k) = number[x%10];
            x = x/10;
            k++;
        }
        while(k<7){
            *(unsigned *)(seg_base_addr + 6 - k) = number[0];
            k++;
        }
    }
```

만약 인자가 2개라면 (./data/local/tmp/mmap [int]) 인자의 값이 표현 범위를 초과할 경우 최댓값과 최솟값으로 설정해주었고, while문을 사용하여 10을 나눈 나머지를 segment에 띄우는 것으로 표현하였습니다.

```
    }
    else if(argc==4){
        int x = atoi(argv[1]);
        int y = atoi(argv[3]);

        if(argv[2][0]=='+')
            x = x+y;
        else if(argv[2][0] == '-')
            x = x-y;
        else if(argv[2][0] == 'x')
            x = x*y;
        else x = x/y;

        if(x>9999999) x = 9999999;
        if(x<0) x = 0;

        int k=0;
        while(x){
            *(unsigned *)(seg_base_addr + 6 - k) = number[x%10];
            x = x/10;
            k++;
        }
        while(k<7){
            *(unsigned *)(seg_base_addr + 6 - k) = number[0];
            k++;
        }
    }
}
```

그 후 인자가 4개일 경우(./data/local/tmp/mmap 94 + 123) 2번째 인자와 4번째 인자를 int형으로 변경한 후 3번째 인자가 어떤 연산자에 따라 계산을 해준 후 segment에 표기하게 해주었습니다.

마지막으로 인자가 3개일 경우(./data/local/tmp/mmap x 156)는 기존의 segment값을 우선 읽어야 하기 때문에 for문을 통해 segment를 read하고 새로운 char형 array에 넣어준 후 int형으로 바꾸는 과정을 거치도록 만들었습니다.

```
else if(argc==3){
    char aa[8];
    for(int i=0; i<7; i++){
        if(*(unsigned*)(seg_base_addr + i) == number[0]) aa[i]='0';
        if(*(unsigned*)(seg_base_addr + i) == number[1]) aa[i]='1';
        if(*(unsigned*)(seg_base_addr + i) == number[2]) aa[i]='2';
        if(*(unsigned*)(seg_base_addr + i) == number[3]) aa[i]='3';
        if(*(unsigned*)(seg_base_addr + i) == number[4]) aa[i]='4';
        if(*(unsigned*)(seg_base_addr + i) == number[5]) aa[i]='5';
        if(*(unsigned*)(seg_base_addr + i) == number[6]) aa[i]='6';
        if(*(unsigned*)(seg_base_addr + i) == number[7]) aa[i]='7';
        if(*(unsigned*)(seg_base_addr + i) == number[8]) aa[i]='8';
        if(*(unsigned*)(seg_base_addr + i) == number[9]) aa[i]='9';
    }
    aa[7]='\0';
    int x = atoi(aa);
    int y = atoi(argv[2]);

    if(argv[1][0]=='+')
        x = x+y;
    else if(argv[1][0] == '-')
        x = x-y;
    else if(argv[1][0] == 'x')
        x = x*y;
    else x = x/y;

    if(x>9999999) x = 9999999;
    if(x<0) x = 0;

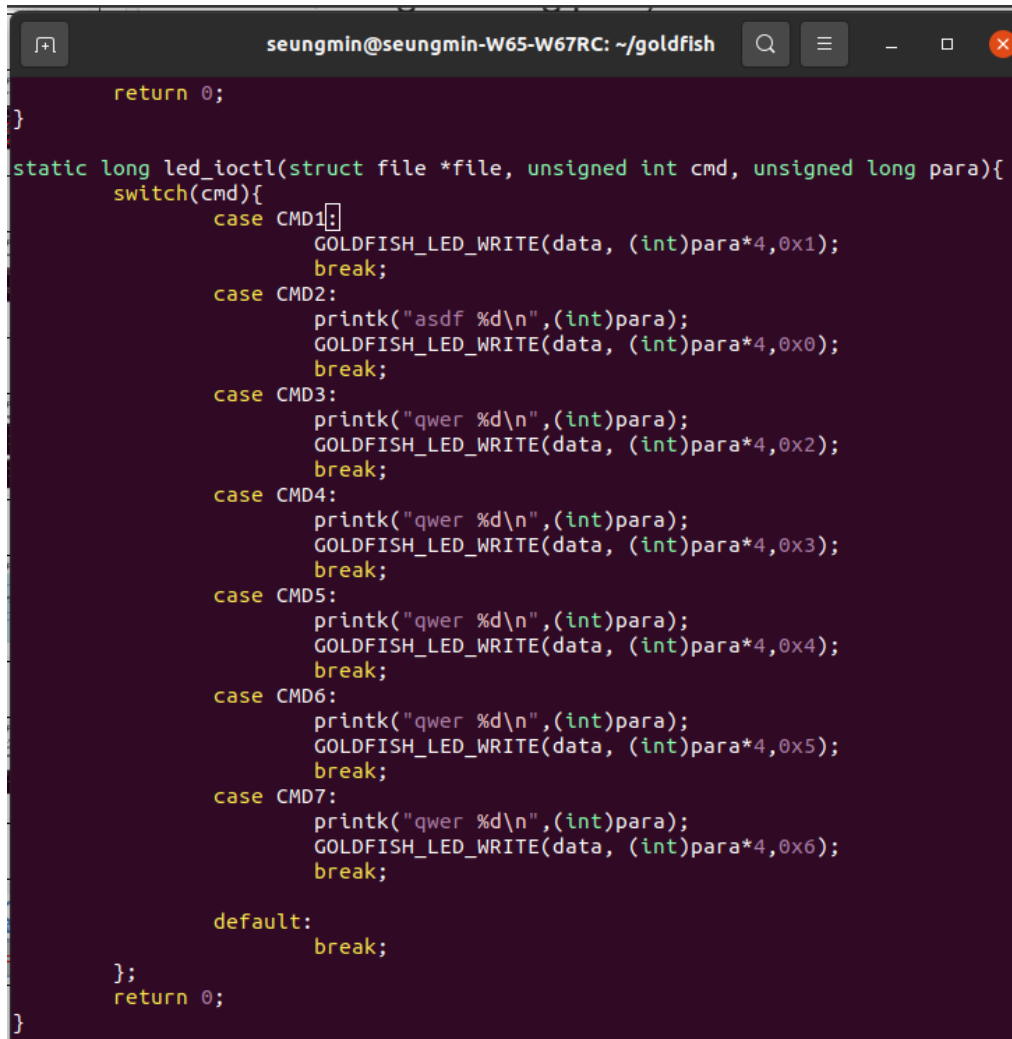
    int k=0;
    while(x){
        *(unsigned*)(seg_base_addr + 6 - k) = number[x%10];
        x = x/10;
        k++;
    }
    while(k<7){
        *(unsigned*)(seg_base_addr + 6 - k) = number[0];
        k++;
    }
}

*seg0 = number[0];
*seg1 = number[1];
*seg2 = number[2];
*seg3 = number[3];
*seg4 = number[4];
*seg5 = number[5];
*seg6 = number[6];

munmap(seg_base_addr, 4096);
close(fd);
return 0;
```

Segment 숫자를 읽는 과정 외에는 나머지와 동일하게 진행하였습니다.

다음으로 mini\_led\_ioctl.c 코드 설명에 앞서 goldfish\_led.c 코드에 추가한 부분을 설명하겠습니다.



```
return 0;
}

static long led_ioctl(struct file *file, unsigned int cmd, unsigned long para){
    switch(cmd){
        case CMD1:
            GOLDFISH_LED_WRITE(data, (int)para*4,0x1);
            break;
        case CMD2:
            printk("asdf %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x0);
            break;
        case CMD3:
            printk("qwer %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x2);
            break;
        case CMD4:
            printk("qwer %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x3);
            break;
        case CMD5:
            printk("qwer %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x4);
            break;
        case CMD6:
            printk("qwer %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x5);
            break;
        case CMD7:
            printk("qwer %d\n",(int)para);
            GOLDFISH_LED_WRITE(data, (int)para*4,0x6);
            break;

        default:
            break;
    };
    return 0;
}
```

led\_ioctl 함수 안에 command number에 따라 LED에 나타낼 색상을 설정해주었습니다. (CMD1 : white, CMD2 : gray, CMD3 : red, CMD4 : yellow, CMD5 : green, CMD6 : sky, CMD7 : purple)

표시할 led의 위치는 para인자를 통해 받도록 설정하였습니다.

```

#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>

#define LED_ADDR 0xff012000
#define LED_SIZE 4096

#define LED0 0x000
#define LED1 0x004
#define LED2 0x008
#define LED3 0x00C
#define LED4 0x010
#define LED5 0x014
#define LED6 0x018

#define COL0 0x0
#define COL1 0x1
#define COL2 0x2
#define COL3 0x3
#define COL4 0x4
#define COL5 0x5
#define COL6 0x6

#define CMD1 0x04
#define CMD2 0x05

#define CMD3 0x06
#define CMD4 0x07
#define CMD5 0x08
#define CMD6 0x09
#define CMD7 0x0A

```

다음으로 mini\_led\_ioctl.c 코드에 대해 설명하겠습니다. 우선 기존에 주어진 mini\_calc\_ioctl.c 코드가 segment의 기준에 맞추어 값이 설정되어 있기 때문에 led형으로 변경을 해주었습니다.

main함수에서 처음으로 해준 것은 두번째 인자를 새로운 char형 array에 저장하게 하였습니다. 두번째 인자의 형태가 "LED\_"로 시작하기 때문에 앞 4글자는 생략하게 하였습니다.

```

unsigned char color[] = {COL0, COL1, COL2, COL3, COL4, COL5, COL6};

int main(int argc, char *argv[])
{
    int i, fd;
    unsigned *led_base_addr;
    unsigned *led0, *led1, *led2, *led3, *led4, *led5, *led6;

    fd = open("/dev/led", O_RDWR);

    /*
     * Access device by ioctl
     * ex. ioctl(fd, cmd_num, parameter)
     */
    if(argc==2){
        int i=0;
        char x[30];
        while(argv[1][i+4]){
            x[i] = argv[1][i+4];
            i++;
        }
    }
}

```

현재 x array 에 저장된 문자는 "LED\_" 이후의 글자이기 때문에 "ON"이나 "OFF"에 따라 if 문을 구분해주었습니다. (x[0]은 'O'로 공통이기 때문에 x[1]로 구분)

```
    }
    x[i]='\0';
    if(x[1]=='N'){
        if(x[3]=='A'){
            for(int i=0; i<=6; i++) ioctl(fd, CMD1, i);
        }
        else{
            char y[2];
            y[0] = x[6];
            y[1] = '\0';
            char z[2];
            z[0] = x[8];
            z[1] = '\0';
            int yy = atoi(y);
            int zz = atoi(z);
            for(int i=yy; i<=zz; i++) ioctl(fd, CMD1, i);
        }
    }
    else if(x[1]=='F'){
        if(x[4]=='A'){
            for(int i=0; i<=6; i++) ioctl(fd, CMD2, i);
        }
        else{
            char y[2];
            y[0] = x[7];
            y[1] = '\0';
            char z[2];
            z[0] = x[9];
            z[1] = '\0';
            int yy = atoi(y);
            int zz = atoi(z);
            for(int i=yy; i<=zz; i++) ioctl(fd, CMD2, i);
        }
    }
}
```

"ON"일 경우에는 CMD1을 전달하고 "OFF"일 경우에는 CMD2를 전달합니다.

If문 안에서도 "LED\_ON\_", "LED\_OFF\_"이후에 "ALL"이 올 경우와 아닐 경우로 나누어 진행하였습니다. 만약 "ALL"이면 모든 led에 command를 보내주게 하였고, 아닐 경우에는 숫자만을 읽어서 그 범위 내의 led에 command를 보내게 해주었습니다.

"LED\_" 이후에 "ON"과 "OFF"가 오지 않을 경우에는 다음과 같이 진행됩니다.

```
else if(x[1]=='x'){
    char y[2];
    y[0] = x[3];
    y[1] = '\0';
    char z[2];
    z[0] = x[5];
    z[1] = '\0';
    int yy = atoi(y);
    int zz = atoi(z);

    if(x[11]=='g' && x[13]=='a')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD2, i);

    if(x[11]=='w')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD1, i);

    if(x[11]=='r')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD3, i);

    if(x[11]=='y')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD4, i);

    if(x[11]=='g' && x[13]=='e')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD5, i);

    if(x[11]=='s')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD6, i);

    if(x[11]=='p')
        for(int i=yy; i<=zz; i++) ioctl(fd, CMD7, i);
}
```

숫자 범위를 파악하고, "LED\_[x:0~6]\_[y:" 이후에 오는 알파벳에 맞추어 각 색상에 맞는 command를 보내도록 해주었습니다.

```
else{
    if(x[7]=='g' && x[9]=='a')
        for(int i=0; i<=6; i++) ioctl(fd, CMD2, i);

    if(x[7]=='w')
        for(int i=0; i<=6; i++) ioctl(fd, CMD1, i);

    if(x[7]=='r')
        for(int i=0; i<=6; i++) ioctl(fd, CMD3, i);

    if(x[7]=='y')
        for(int i=0; i<=6; i++) ioctl(fd, CMD4, i);

    if(x[7]=='g' && x[9]=='e')
        for(int i=0; i<=6; i++) ioctl(fd, CMD5, i);

    if(x[7]=='s')
        for(int i=0; i<=6; i++) ioctl(fd, CMD6, i);

    if(x[7]=='p')
        for(int i=0; i<=6; i++) ioctl(fd, CMD7, i);

    }

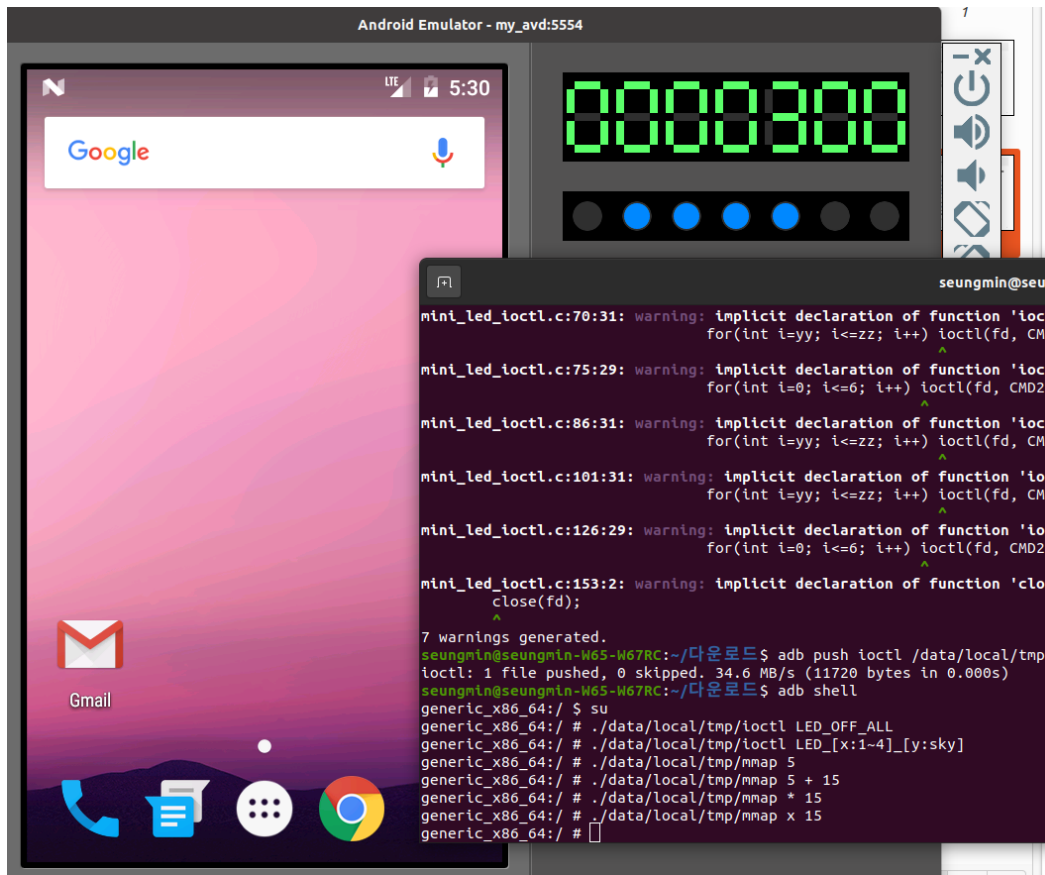
}

//ioctl(fd, CMD3, 1);

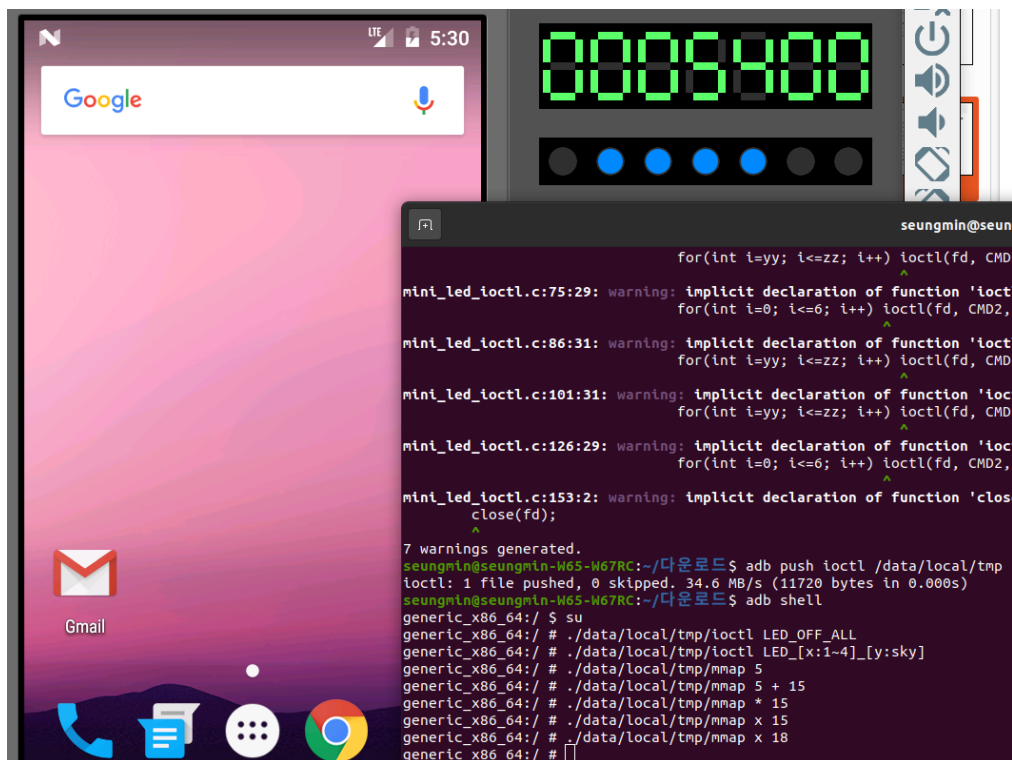
close(fd);
return 0;
}
```

"LED\_ALL\_"의 경우도 동일하게 해주었습니다.

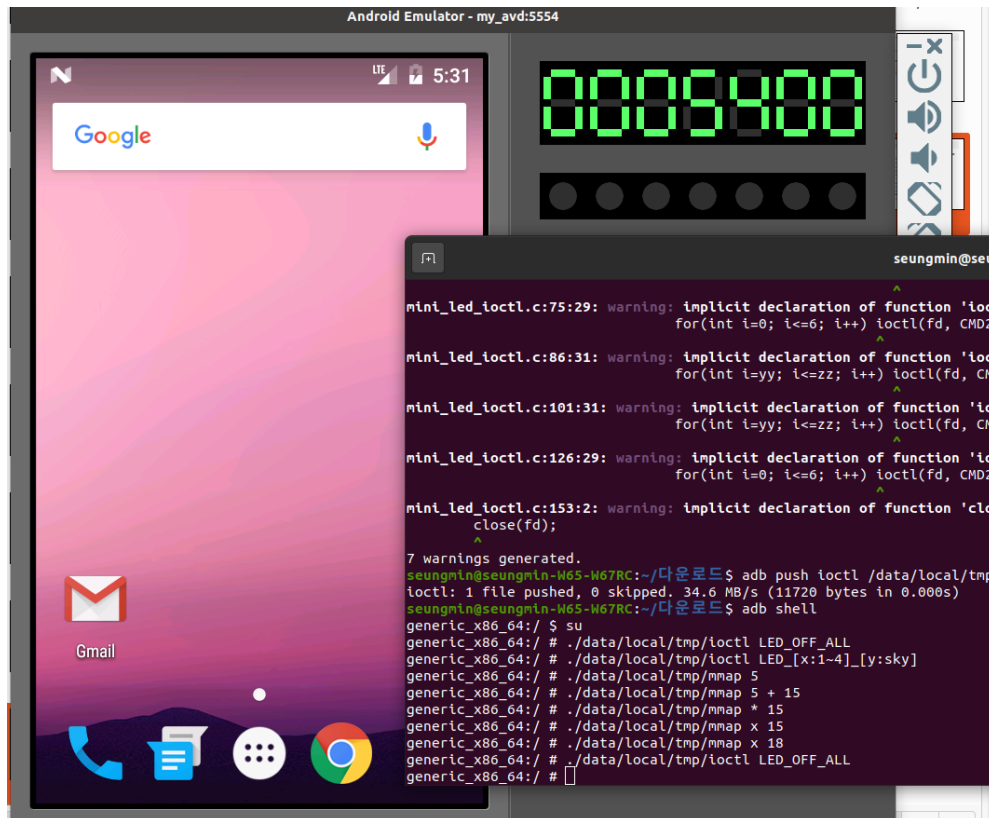
실행 화면입니다.



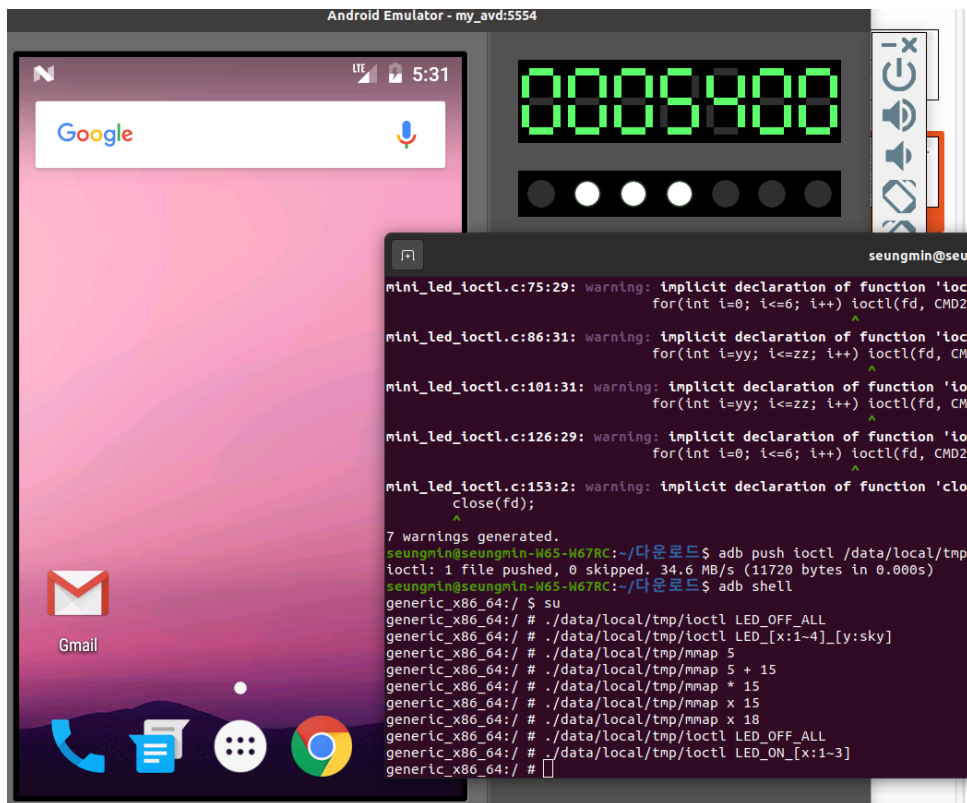
위 화면에서 (./data/local/tmp/mmap x 18)을 실행하면 아래와 같이 나옵니다.



위 화면에서 (./data/local/tmp/ioctl LED\_OFF\_ALL)을 실행하면 아래와 같이 나옵니다.

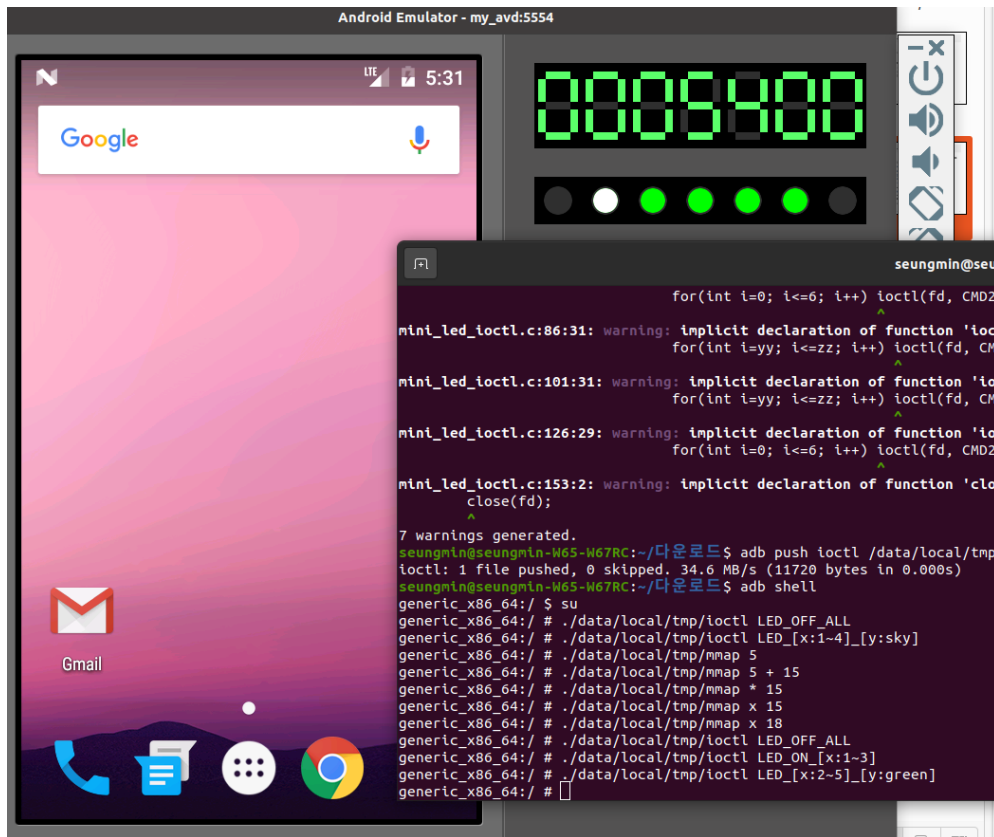


위 화면에서 (./data/local/tmp/ioctl LED\_ON\_[x:1~3])을 실행하면 아래와 같이 나옵니다.





위 화면에서 (./data/local/tmp/iocctl LED\_[x:2~5]\_[y:green])을 실행하면 아래와 같이 나옵니다.



위 화면에서 (./data/local/tmp/mmap / 8)을 실행하면 아래와 같이 나옵니다.

