임베디드시스템실습_lab4

2017313107 이승태

memory mapped IO, 실제 메모리 공간에 매핑하여 IO를 사용하는 실습을 해보았다.

1. mmap을 이용한 계산기 구현하기

segtonum은 segment값을 숫자로 바꾸어준다.

operate는 num1과 num2를 op(숫자로 입력받음)연산을 진행한다.

strtonum 주소가 주어지면, 그곳에 있는 문자들을 int로 바꾸어준다.

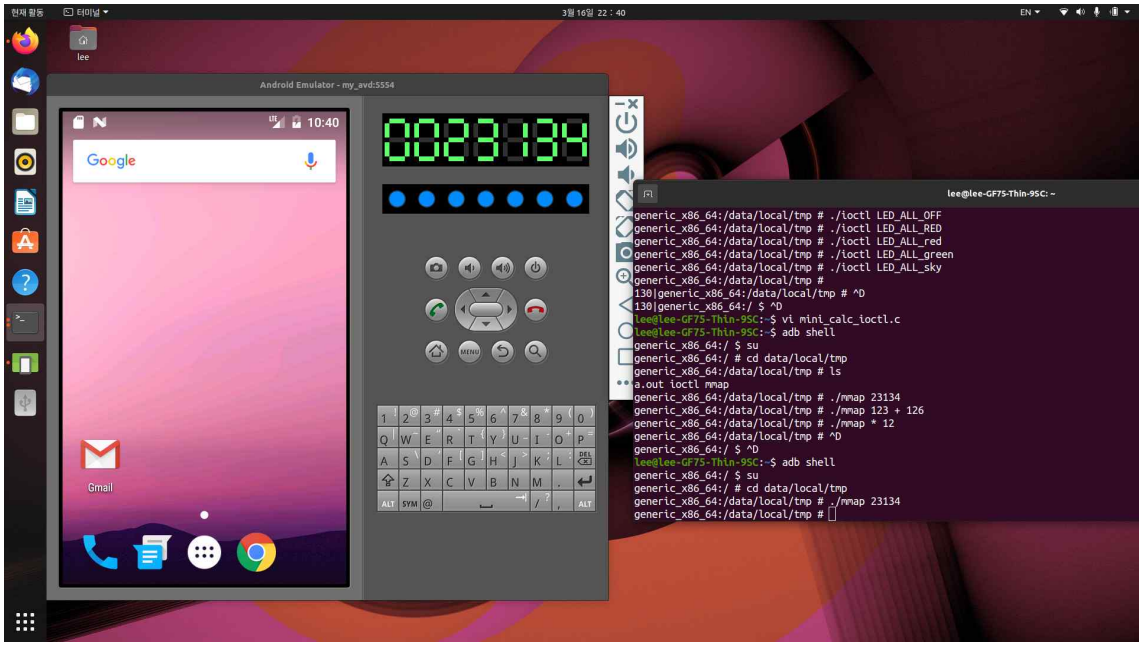
```c
int segtonum(int s)
{
    if(s == NUM1)   return 1;
    if(s == NUM2)   return 2;
    if(s == NUM3)   return 3;
    if(s == NUM4)   return 4;
    if(s == NUM5)   return 5;
    if(s == NUM6)   return 6;
    if(s == NUM7)   return 7;
    if(s == NUM8)   return 8;
    if(s == NUM9)   return 9;
    if(s == NUM0)   return 0;
    if(s == 0)      return 0;
    else            return -1;
}

int operate(int num1, int num2, int op)
{
    if(op == 0) return num1 + num2;
    if(op == 1) return num1 - num2;
    if(op == 2) return num1 * num2;
    if(op == 3) return num1 / num2;
    else        return -1;
}

int strtonum(char *strnum)
{
    int num = 0;
    int i = 0;
    while(strnum[i] != '\0')
    {
        num *= 10;
        num += strnum[i] - 48;
        i++;
    }
    return num;
}
```

```c
int numtmp = 0;
int fnum = 0;
if(argc == 2 || argc == 4)  fnum = strtonum(argv[1]);
if(argc == 4)
{
    int op;
    if(argv[2][0] == '+')       op = 0;
    else if(argv[2][0] == '-')  op = 1;
    else if(argv[2][0] == 'x')  op = 2;
    else if(argv[2][0] == '/')  op = 3;

    numtmp = strtonum(argv[3]);
    fnum = operate(fnum,numtmp,op);
}
if(argc == 3)
{
    fnum = segtonum(*seg6) + segtonum(*seg5) * 10 + seg
    int op;
    if(argv[1][0] == '+')       op = 0;
    else if(argv[1][0] == '-')  op = 1;
    else if(argv[1][0] == 'x')  op = 2;
    else if(argv[1][0] == '/')  op = 3;
    numtmp = strtonum(argv[2]);
    fnum = operate(fnum,numtmp,op);
}

if(fnum > 9999999 || fnum < 0)
{
    *seg6 = number[0];
    *seg5 = number[0];
    *seg4 = number[0];
    *seg3 = number[0];
    *seg2 = number[0];
    *seg1 = number[0];
    *seg0 = number[0];
}
else
{
    *seg6 = number[fnum % 10];
    fnum /= 10;
    *seg5 = number[fnum % 10];
    fnum /= 10;
    *seg4 = number[fnum % 10];
    fnum /= 10;
    *seg3 = number[fnum % 10];
    fnum /= 10;
    *seg2 = number[fnum % 10];
    fnum /= 10;
    *seg1 = number[fnum % 10];
    fnum /= 10;
    *seg0 = number[fnum % 10];
}
```

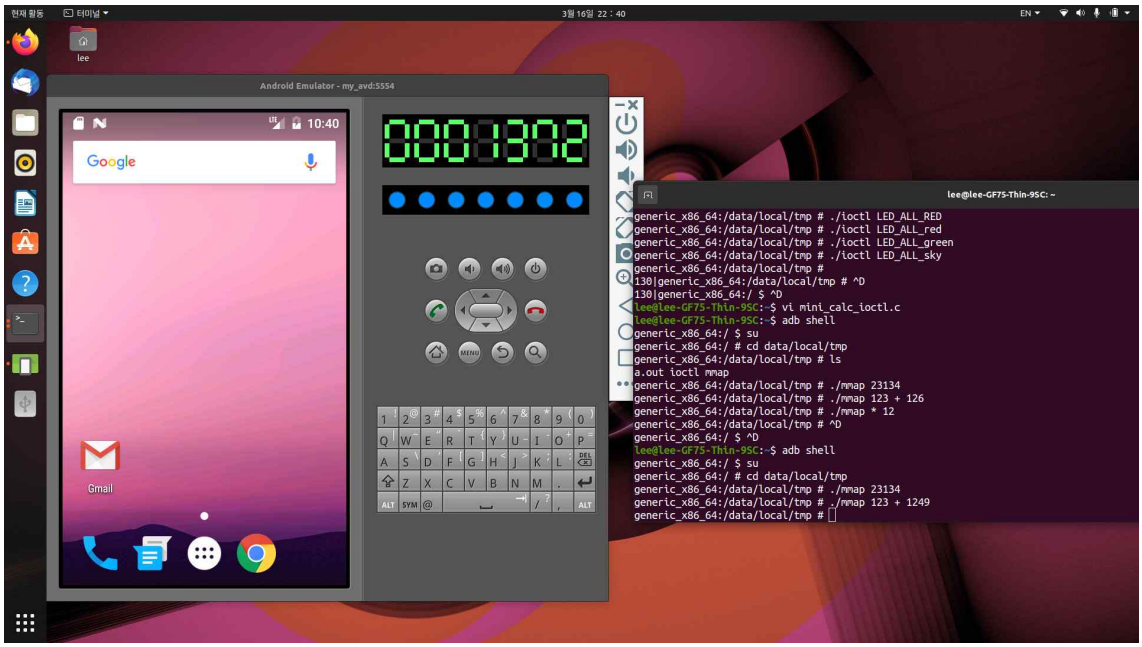
그 후, argv값을 봐서, 2인 경우 그냥 숫자를 써주고, 4인 경우, 두 숫자를 계산한 값을 띄우고, 3인 경우, 숫자를 메모리에서 읽어서 들어온 숫자와 연산해준다.

mmap시연 사진

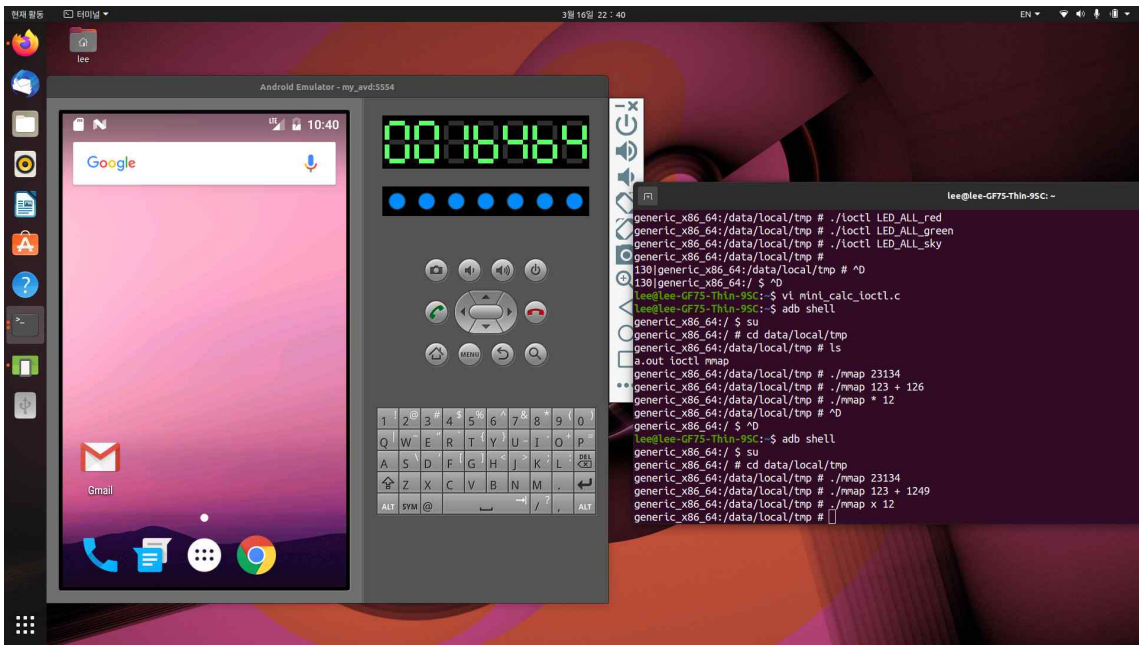
(1) 23134를 그냥출력



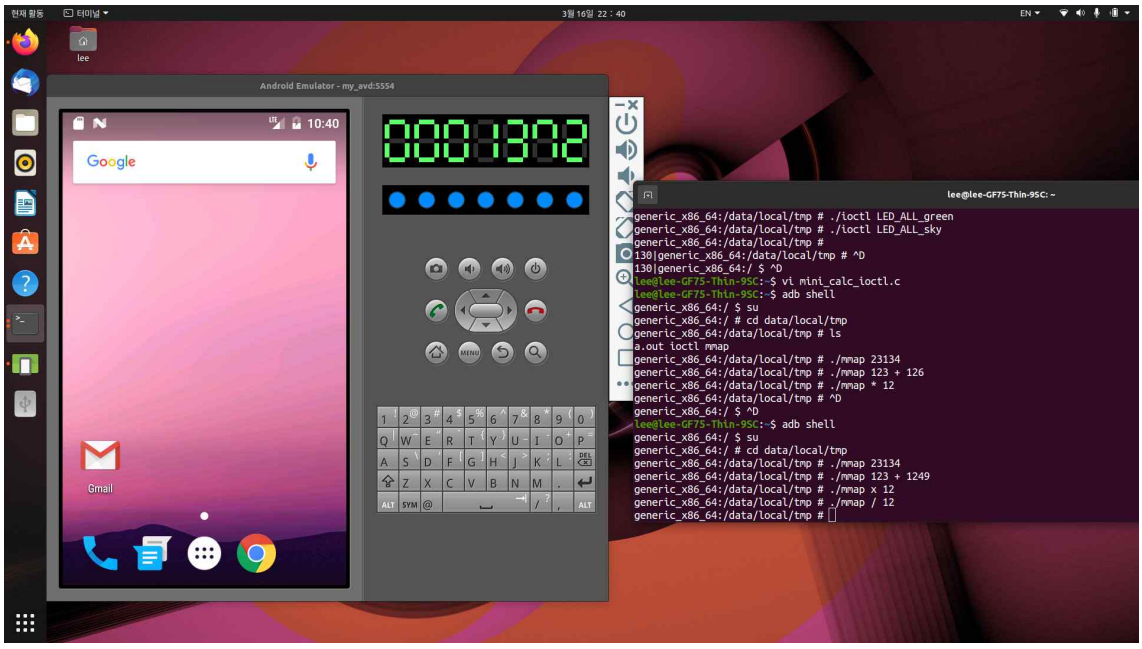
(2) 123 + 1249를 출력



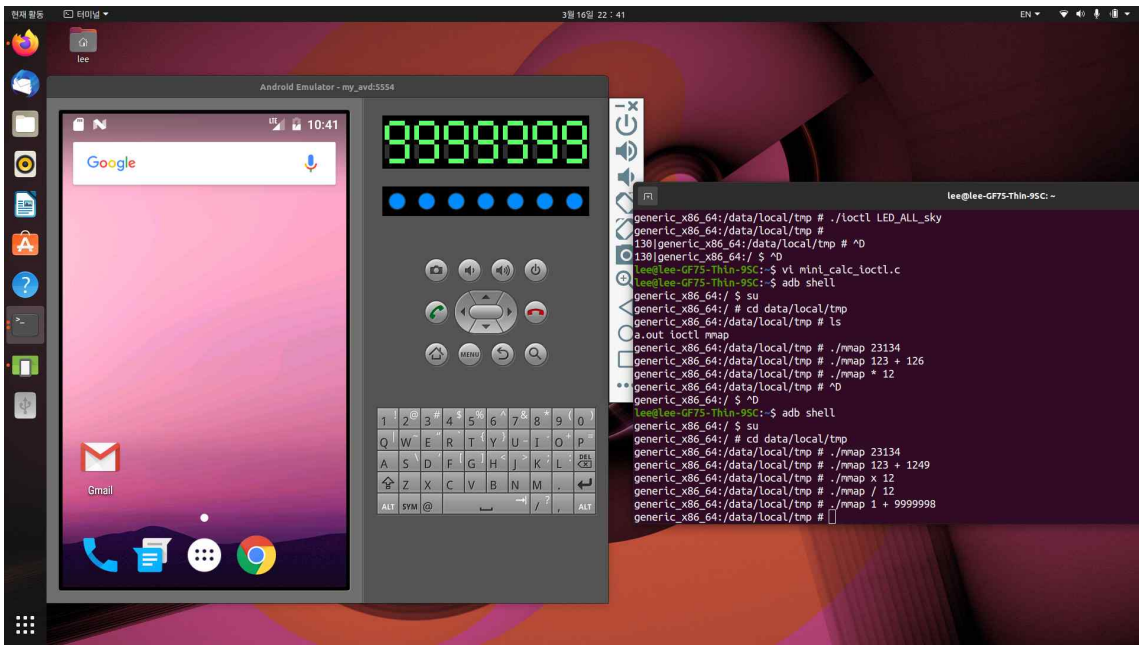
(3) 원래 있던 값에 x 12



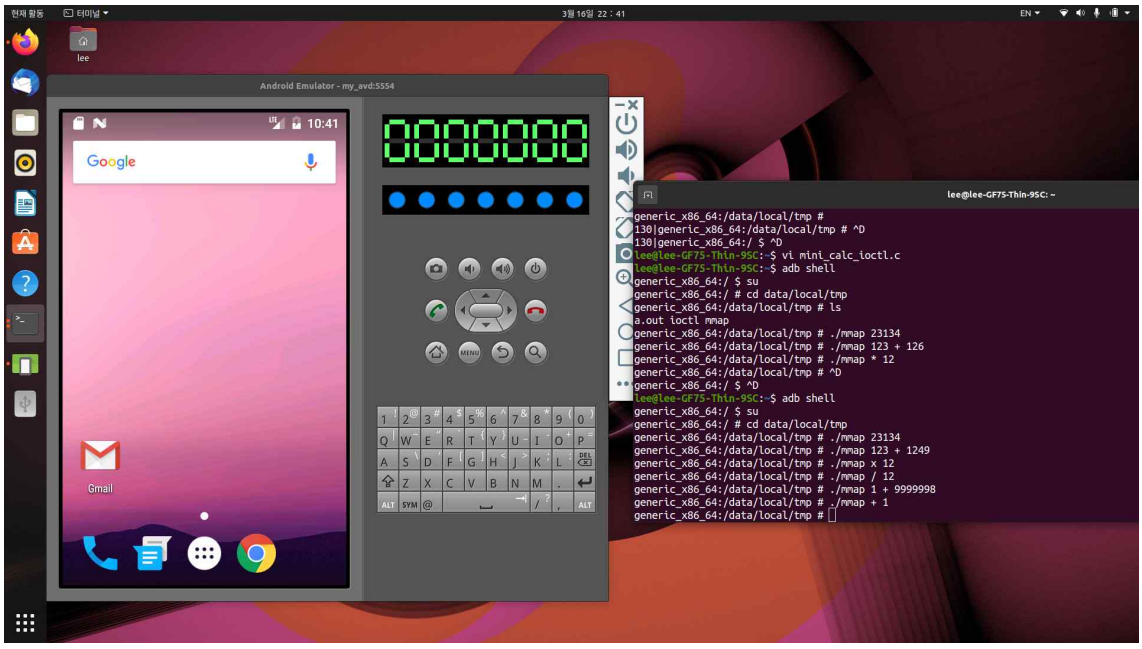
(4) 원래 있던 값에 / 12



(5) 1 + 9999998을 출력



(6) 원래 있던 값에 + 1



2. ioctl을 이용하는 방법

먼저, goldfish_ioctl에서 다음과 같이 cmd와 para가 들어왔을 때, LED가 값에 따라서 켜지게 설정해줘야한다.

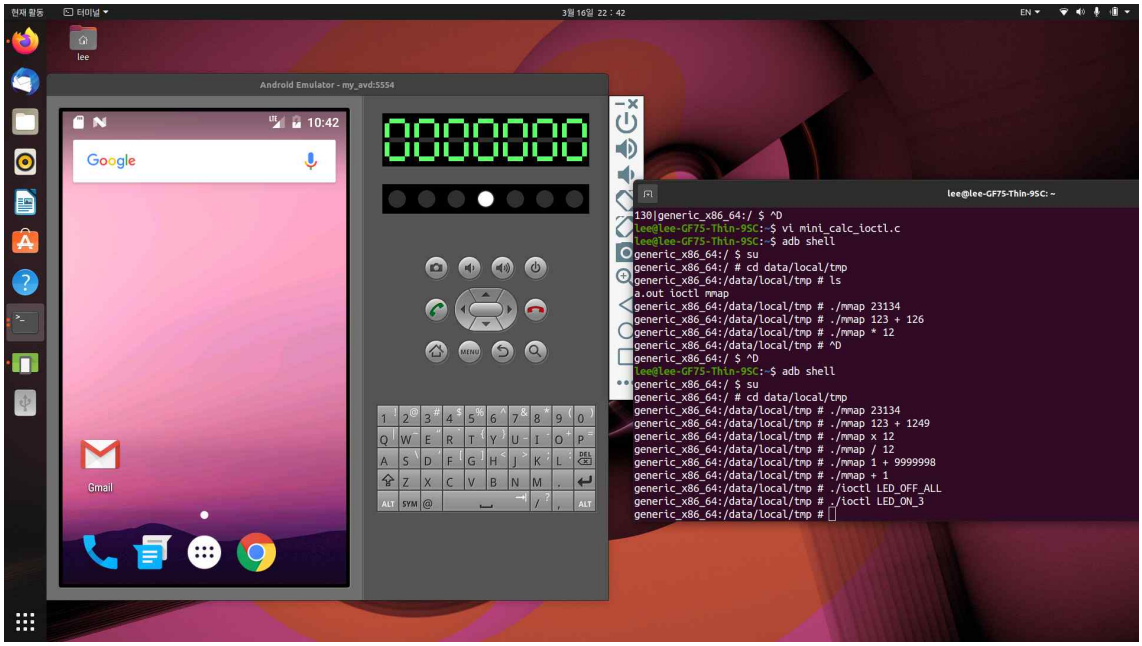
```
static long led_ioctl(struct file *file, unsigned int cmd, unsigned long para)
{
    switch(cmd)
    {
        case LED_CMD0:
            GOLDFISH_LED_WRITE(led_data, LED0, para);
            break;
        case LED_CMD1:
            GOLDFISH_LED_WRITE(led_data, LED1, para);
            break;
        case LED_CMD2:
            GOLDFISH_LED_WRITE(led_data, LED2, para);
            break;
        case LED_CMD3:
            GOLDFISH_LED_WRITE(led_data, LED3, para);
            break;
        case LED_CMD4:
            GOLDFISH_LED_WRITE(led_data, LED4, para);
            break;
        case LED_CMD5:
            GOLDFISH_LED_WRITE(led_data, LED5, para);
            break;
        case LED_CMD6:
            GOLDFISH_LED_WRITE(led_data, LED6, para);
            break;
        default:
            break;
    }
    return 0;
}

//fix
```

```
if(strncmp(argv[1], "LED_", 4)) return 0;

//led_number_color
if(argv[1][4] >= 48 && argv[1][4] <= 57)
{
    cmd += (argv[1][4] - 48) * 10;
    if(argv[1][8] == 'a')   cmd += 0;
    if(argv[1][8] == 'i')   cmd += 1;
    if(argv[1][8] == 'd')   cmd += 2;
    if(argv[1][8] == 'l')   cmd += 3;
    if(argv[1][8] == 'e')   cmd += 4;
    if(argv[1][8] == 'y')   cmd += 5;
    if(argv[1][8] == 'r')   cmd += 6;
}
//led_all_color
else if(argv[1][4] == 'A')
{
    cmd += 100;
    if(argv[1][10] == 'a')  cmd += 0;
    if(argv[1][10] == 'i')  cmd += 1;
    if(argv[1][10] == 'd')  cmd += 2;
    if(argv[1][10] == 'l')  cmd += 3;
    if(argv[1][10] == 'e')  cmd += 4;
    if(argv[1][10] == 'y')  cmd += 5;
    if(argv[1][10] == 'r')  cmd += 6;
}
//led_on_number
else if(argv[1][5] == 'N')
{
    if(argv[1][7] == 'A')   cmd = 101;
    else                    cmd = (argv[1][7] - 48) * 10 + 1;
}
//led_off_number
else if(argv[1][5] == 'F')
{
    if(argv[1][7] == 'A')   cmd = 100;
    else                    cmd = (argv[1][8] - 48) * 10;
}
else    return 0;

if(cmd < 100)           ioctl(fd, cmd/10+3, cmd%10);
else
{
    for(i=3;i<=9;i++)   ioctl(fd, i, cmd%10);
}

close(fd);
return 0;
```

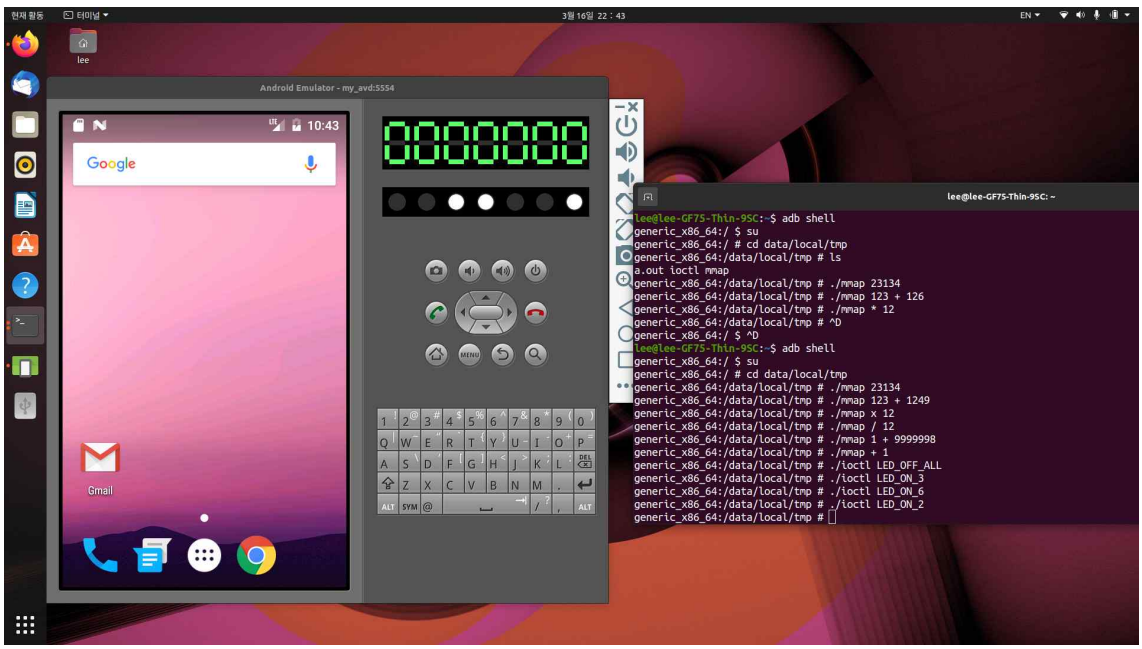
그 후, argv[1]에 값이 뭐가 들어가 있는지 확인하여 동작을 실행한다. 위의 코드의 동작은 주석코드의 내용대로 작동한다.

ioctl 시연 사진

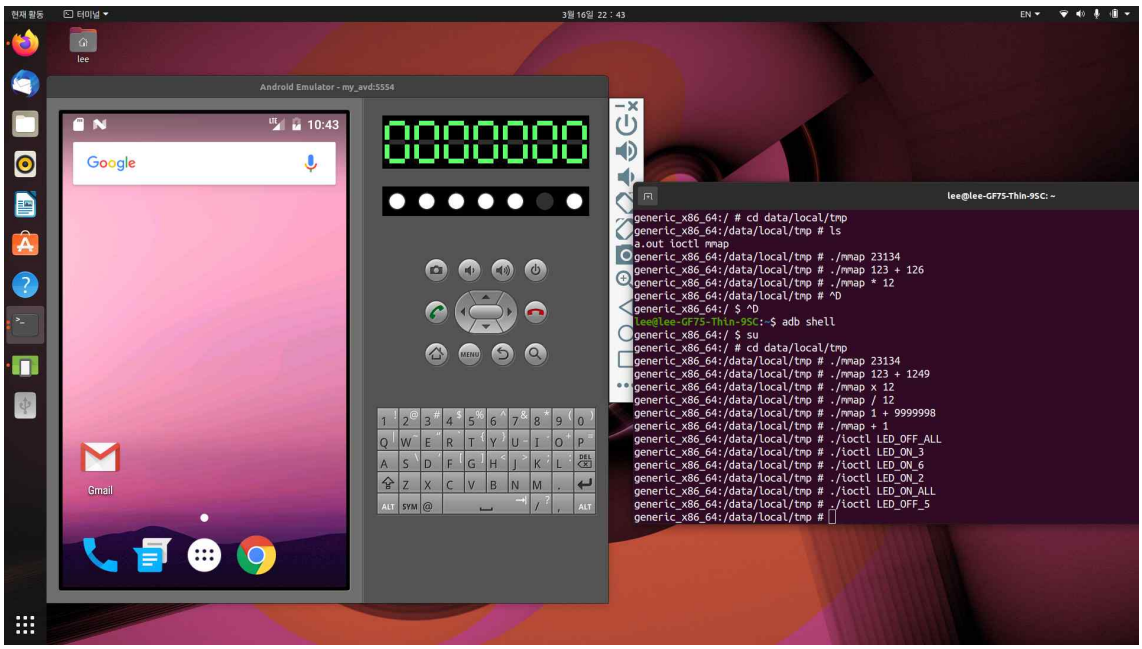
1. LED3번만 킨 모습



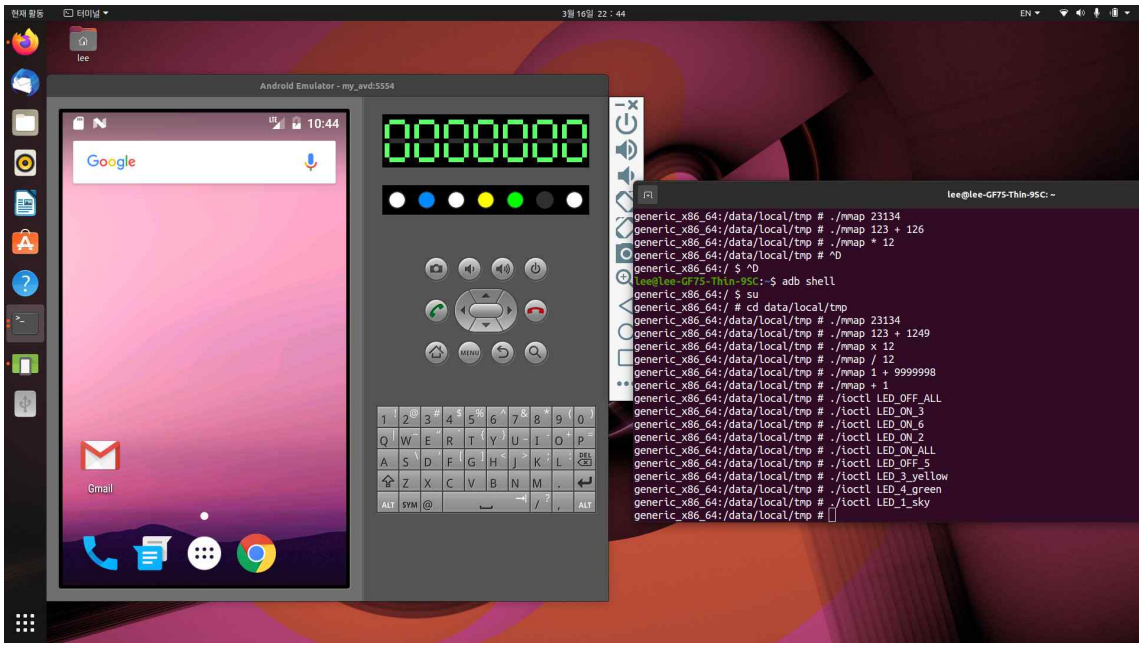
2. LED6, 2번을 켠 모습



3. LED를 다 켜고 5번만 끈 모습



4. 3번은 노란색, 4번은 초록색, 1번은 하늘색으로 켠 모습



5. LED를 빨간색으로 다 켠 모습