Final Report
for Digital Watch System

Project Team
Team 4

Date
2019-10-28

Team Information
201212519 김선우
201612368 이지우
201614157 김도연
201714521 신효혜

Table of Contents

1 Introduction

1.1 Objectives

SRS & SAS & SDS (for Digital Watch System)에 맞게 Digital Watch System 을 SW 로 구현한 것을 보고하는 문서이다.

1.2 References

SRS, SA_v2, SDS

2 Code

2.1 DWS.h

```
//status
#define TIME_MODE 1
#define SEC_SET 2
#define HOUR_SET 3
#define MINUTE_SET 4
#define YEAR_SET 5
#define MONTH_SET 6
#define DAY_SET 7
#define ALARM_MODE 8
#define AL_HOUR_SET 9
#define AL_MINUTE_SET 10
#define STOPWATCH_MODE 11
#define START 12
#define STOP 13
#define LAP_TIME 14
#define TURN_ON 15
#define TURN_OFF 16
#define TURN_YELLOW 17
```

```
//display command
#define PRINT_TIME_MODE 20
#define PRINT_SEC_SET 21
#define PRINT_HOUR_SET 22
#define PRINT_MINUTE_SET 23
#define PRINT_YEAR_SET 24
#define PRINT_MONTH_SET 25
#define PRINT_DAY_SET 26
#define PRINT_ALARM_MODE 27
#define PRINT_AL_HOUR_SET 28
#define PRINT_AL_MINUTE_SET 29
#define PRINT_STOPWATCH_MODE 30
#define PRINT_START 31
#define PRINT_STOP 32
#define PRINT_LAP_TIME 33
#define PRINT_BACKLIGHT 34
```

DWS 의 각 상태와 각 상태에서 출력할 display command 를 헤더 파일에 정의했다.

```
typedef struct _alarm_information{
   bool alarm_power; // 알람 켜짐 여부
   char display_alarm_indicator; // Alarm indicator on/off 여부
   int beep_time;
}alarm_information;
```

Alarm_information 구조체에는 알람 켜짐 여부(alarm_power), 알람 indicator 켜짐 여부(display_alarm_indicator), 알람이 울릴 시간(beep_time)을 저장했다.

```
// Data storage
int st_info; //Status information
int al_info; //Alarm information
int mo_info; //mode information
int bl_info; //backlight information
```

DWS 에서 필요한 data storage 는 st_info, al_info, mo_info, bl_info 총 4 가지다. St_info 는 DWS 의 현재 상태를, al_info 는 알람 전원 여부를, mo_info 는 DWS 의 세 가지 상태 중 하나를, bl_info 는 backlight 의 전원 여부를 저장한다.

```
int backlight_time; // 2sec
int display_command;
int btn; // button input storage

time_t current_time;
time_t alarm_time;

struct tm current_tm;
struct tm alarm_tm;
struct tm st_tm;
struct timeval st_tv;
struct timeval st_start;
struct timeval st_stop;

// 스톱워치 변수
unsigned int stop_min;
unsigned int stop_sec;
unsigned int stop_milisec;
```

St_tv 는 스톱워치의 기준이 되는 시간을 저장하는 구조체로 st_tv._tv_sec 와 st_tv.tv_usec 로 나뉜다.

버튼은 a,b,c,d(소문자)의 입력을 받고 나머지의 입력은 무시한다.

2.2 DWS.c

```
// 월마다 날짜를 구하는 함수
int date(int year, int month)
{
        int mdays[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };
        if ((year % 4 == 0) && ((year % 100 != 0) || (year % 400 == 0)))//윤년 판단
        {
                mdays[1] = 29;//2월달의 날 수를 29로 설정
        }

        //month는 mdays의 인덱스로 사용해서 실제 출력할 때는 1을더해서 출력합니다.
        int day = mdays[month];
        return day;
}
```

달마다 일 수가 다르기 때문에 각 월의 day 수를 구하기 위해 date 함수를 만들었다.

```
// Current 타이머를 만드는 함수
int createCurrent(timer_t *timerID, int sec, int msec){
        struct sigevent te;
        struct itimerspec its;
        struct sigaction sa;
        int sigNo=SIGRTMIN;

        sa.sa_flags=SA_SIGINFO;
        sa.sa_sigaction = current;
        sigemptyset(&sa.sa_mask);

        if(sigaction(sigNo, &sa, NULL)== -1 ){
                printf("sigaction error\n");
                return -1;
        }
        te.sigev_notify = SIGEV_SIGNAL;
        te.sigev_signo = sigNo;
        te.sigev_value.sival_ptr = timerID;
        timer_create(CLOCK_REALTIME, &te, timerID);

        its.it_interval.tv_sec = sec;
        its.it_interval.tv_nsec = msec * 1000000;
        its.it_value.tv_sec = sec;

        its.it_value.tv_nsec = msec * 1000000;
        timer_settime(*timerID, 0, &its, NULL);

        return 0;


}
```

고민 끝에 시스템 콜의 설정된 sec 와 msec 에 따라 current() 함수를 부르는 타이머를 만들어주는 함수를 만들었다.

```
// 현재 시간을 매초 업데이트하는 프로세스
static void current()
{

        current_tm.tm_sec++;

        if(current_tm.tm_sec==60){
                current_tm.tm_sec=0;
                current_tm.tm_min++;
        }
        else if(current_tm.tm_min==60){
                current_tm.tm_min=0;
                current_tm.tm_hour++;
        }
        else if(current_tm.tm_hour==24){
                current_tm.tm_hour=0;
                if(current_tm.tm_wday==6){
                        current_tm.tm_wday=0;
                }else{
                        current_tm.tm_mday++;
                        current_tm.tm_wday++;
                }
        }
        else if(current_tm.tm_mday==date(current_tm.tm_year, current_tm.tm_mon)){
                current_tm.tm_mday=0;
                current_tm.tm_mon++;
        }
        else if(current_tm.tm_mon==12){
                current_tm.tm_mon=1;
                current_tm.tm_year++;
        }
        else if(current_tm.tm_year==2099){
                current_tm.tm_year=2019;
        }
}
```

current 함수는 1 초마다 호출 되서 현재 시간을 저장하는 구조체의 current_tm 의 sec 변수를 1 씩 증가시킨다.

2.2.1 Main Controller

STD 로 표현시에 controller 에 의해 Enable/Disable 로 control 되는 상태의 경우는 모두 비슷한 방식으로 구현했다.

1) 버튼의 입력을 받아 상태 변수를 다른 변수로 변경시키는 부분 + 2) 현재 상태에서 처리해야 할 프로세스 + display 에 display_command 를 보내는 부분으로 구성된다.

```
// 현재 시간을 보여주는 프로세스
void time_mode()
{

                if (btn == 'a') {
                st_info = SEC_SET;
                }
                if (btn == 'c') {
                        mo_info = ALARM_MODE;
                }
        display_command = PRINT_TIME_MODE;
        btn = 0;
}
```

1) a: sec_set 으로 이동, c: alarm_mode 로 이동

2) button input 을 저장하는 btn 변수를 초기화

```
void sec_set()
{
                if (btn == 'a') {

                        st_info = 0;
                }
                if (btn == 'b') {
                        if (current_tm.tm_sec == 59) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c') {
                        st_info = HOUR_SET;
                }

        //display
        display_command = PRINT_SEC_SET;
        btn = 0;

}
```

1) a: time_mode 으로 이동, b: 선택된 변수를 1 씩 증가 and 최대값에 도달시 최소값으로 돌아감 c: hour_set 로 이동

2) button input 을 저장하는 btn 변수를 초기화

```
void hour_set()
{
                //TIME_MODE
                if (btn == 'a' && mo_info == TIME_MODE) {
                        st_info = 0;
                }
                if (btn == 'b' && mo_info == TIME_MODE) {
                        if (current_tm.tm_hour == 23) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c' && mo_info == TIME_MODE) {
                        st_info = MINUTE_SET;
                }

                //ALARM_MODE
                if (btn == 'a' && mo_info == ALARM_MODE) {
                                st_info = 0;
                        alarm_info.alarm_power = true;
                        alarm_info.display_alarm_indicator = 'I';
                }
                if (btn == 'b' && mo_info == ALARM_MODE){
                                if (alarm_tm.tm_hour == 23) set_min(st_info);
                                        else plus_one(st_info);
                }
                if (btn == 'c' && mo_info == ALARM_MODE) {
                                st_info = AL_MINUTE_SET;
                }


        //display
        if(mo_info==TIME_MODE) {display_command = PRINT_HOUR_SET;}
        if(mo_info==ALARM_MODE) {display_command = PRINT_AL_HOUR_SET;}
        btn = 0;

}
```

Hour_set 의 경우 현재 시간 설정과 알람 시간 설정 모두에서 사용되는 state 이므로 mo_info(모드를 저장하는 변수)에 저장된 변수에 따라 st_info 가 다르게 변화하고, display_command 또한 다르다.

1) TIME_MODE

a: time_mode 으로 이동, b: 선택된 변수를 1 씩 증가 and 최대값에 도달시 최소값으로 돌아감 c: minute_set 로 이동

ALARM_MODE

a: alarm_indicator 를 켜고 화면에 표시 b: 선택된 변수(알람 시간)를 1 씩 증가 and 최대값 도달 시 최소값으로 돌아감, c: min_set 으로 이동

2) button input 을 저장하는 btn 변수를 초기화

```
void minute_set()
{
                //TIME_MODE
                if (btn == 'a' && mo_info == TIME_MODE) {
                        st_info = 0;
                }

                if (btn == 'b' && mo_info == TIME_MODE) {
                        if (current_tm.tm_min == 59) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c' && mo_info == TIME_MODE) {
                        st_info = YEAR_SET;
                }

                //ALARM_MODE
                if (btn == 'a' && mo_info == ALARM_MODE) {
                                        st_info = 0;
                        alarm_info.alarm_power = true;
                        alarm_info.display_alarm_indicator = 'I';
                 }
                if (btn == 'b' && mo_info == ALARM_MODE){
                        if (alarm_tm.tm_hour == 59) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c' && mo_info == ALARM_MODE) {
                        st_info = AL_HOUR_SET;
                }


        //display
        if(mo_info==TIME_MODE) {display_command = PRINT_MINUTE_SET;}
                if(mo_info==ALARM_MODE) {display_command = PRINT_AL_MINUTE_SET;}
        btn = 0;

}
```

Hour_set 의 경우와 동일

```
void year_set()
{
                if (btn == 'a') {
                        st_info = 0;
                }
                if (btn == 'b') {
                        if (current_tm.tm_year == 199) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c') {
                        st_info = MONTH_SET;
                }

        //display
        display_command = PRINT_YEAR_SET;
        btn = 0;

}

void month_set()
{
                if (btn == 'a') {
                        st_info = 0;
                }
                if (btn == 'b') {
                        if (current_tm.tm_mon == 12) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c') {
                        st_info = DAY_SET;
                }

        //display
        display_command = PRINT_MONTH_SET;
        btn = 0;
}
```

year_set

1) a: time_mode 으로 이동, b: 선택된 변수를 1 씩 증가 and 최대값에 도달시 최소값으로 돌아감 c: month_set 로 이동

2) button input 을 저장하는 btn 변수를 초기화

month_set

1) a: time_mode 으로 이동, b: 선택된 변수를 1 씩 증가 and 최대값에 도달시 최소값으로 돌아감 c: day_set 로 이동

2) button input 을 저장하는 btn 변수를 초기화

```
void day_set()
{
                if (btn == 'a') {
                        st_info = 0;
                }
                if (btn == 'b') {
                        //달 별로 일 수가 다르기 때문에 date함수를 구현해 맞는 max_day를 넣어주었습니다.
                        if (current_tm.tm_mday == date(current_tm.tm_year, current_tm.tm_mon)) set_min(st_info);
                        else plus_one(st_info);
                }
                if (btn == 'c') {
                        st_info = SEC_SET;
                }

        //display
        display_command = PRINT_DAY_SET;
        btn = 0;
        printf("일\n");
}
```

day_set

1) a: time_mode 으로 이동, b: 선택된 변수를 1 씩 증가 and 최대값에 도달시 최소값으로 돌아감 c: sec_set 로 이동

2) button input 을 저장하는 btn 변수를 초기화

```
void plus_one(int st_info)
{
        switch (st_info)
        {
                case SEC_SET:
                        current_tm.tm_sec++; break;
                case HOUR_SET:
                        current_tm.tm_hour++; break;
                case MINUTE_SET:
                        current_tm.tm_min++; break;
                case YEAR_SET:
                        current_tm.tm_year++; break;
                case MONTH_SET:
                        current_tm.tm_mon++; break;
                case DAY_SET:
                        current_tm.tm_mday++;
                        if(current_tm.tm_wday==6) current_tm.tm_wday=0;
                        else current_tm.tm_wday++;
                        break;
                case AL_HOUR_SET:
                        alarm_tm.tm_hour++; break;
                case AL_MINUTE_SET:
                        alarm_tm.tm_min++; break;
                default: break;
        }
}
```

Plus_one 함수는 st_info(DWS 의 현재 상태)를 인자로 받아 상태에 따라 변수를 1 씩 증가시킨다. Day_set 의 경우 요일도 바꿔준다.

```
void set_min(int st_info)
{
        switch (st_info)
        {
                case SEC_SET:
                        current_tm.tm_sec = 0;
                        break;
                case HOUR_SET:
                        current_tm.tm_hour = 0;
                        break;
                case MINUTE_SET:
                        current_tm.tm_min = 0;
                        break;
                case YEAR_SET:
                        current_tm.tm_year = 2019;
                        break;
                case MONTH_SET:
                        current_tm.tm_mon = 1;
                        break;
                case DAY_SET:
                        current_tm.tm_mday = 1;
                        break;
                case AL_HOUR_SET:
                        alarm_tm.tm_hour = 0;
                        break;
                case AL_MINUTE_SET:
                        alarm_tm.tm_min = 0;
                        break;
                default: break;
        }
}
```

요구사항에서 지정한 각 변수의 최대값에 도달했을 때, 최소값으로 변경되는 함수.

```
void alarm_mode()
{

                if (btn == 'a') {
                        st_info =AL_HOUR_SET;
                }
                if (btn == 'b') {
                        alarm_indicator();
                }
                if (btn == 'c') {
                        mo_info = STOPWATCH_MODE;
                }

                //display
                display_command = PRINT_ALARM_MODE;
                btn = 0;
}

void alarm_indicator()
{
        if(alarm_info.alarm_power == true){
                alarm_info.alarm_power=false;
                alarm_info.display_alarm_indicator = ' ';
        }else{
                alarm_info.alarm_power=true;
                alarm_info.display_alarm_indicator = 'I';
        }

}
```

Alarm_mode

1) a: al_hour_set 으로 이동, b: alarm_indicator 함수를 trigger, c: stopwatch_mode 로 이동

2) button input 을 저장하는 btn 변수를 초기화

Alarm_indicator

Alarm_indicator 함수는 trigger 로 호출되는 함수이므로 호출되면 한번 실행되고 상태를 유지하지 않는다. Alarm_power 와 화면 표시를 조정한다.

```
void stopwatch_mode()
{
                if (btn == 'b') {
                        st_info=START;
                }
                if (btn == 'c') {
                                mo_info = TIME_MODE;
                        }


        // Stopwatch에 필요한 기준값 초기화
        gettimeofday(&st_start, NULL);
        stop_milisec = 0;
        stop_sec = 0;
        stop_min = 0;
        display_command = PRINT_STOPWATCH_MODE;
        btn = 0;
}

void start()
{
                if (btn == 'a') {
                        st_info=LAP_TIME;
                }
                if (btn == 'b') {
                        st_info=STOP;
                }


        //display
        gettimeofday(&st_tv, NULL);
        st_tv.tv_sec -= st_start.tv_sec;
        st_stop.tv_sec = st_tv.tv_sec;
        st_tm = *localtime(&st_tv.tv_sec);
        stop_milisec = st_tv.tv_usec/10000;
        stop_sec = st_tm.tm_sec;
        stop_min = st_tm.tm_min;
        display_command = PRINT_START;


        btn = 0;
}
```

Stopwatch 모드에 진입하면 stopwatch 의 기준이 될 시간을 현재 시간을 gettimeofday()로 호출한다.

Start() 모드에 진입하면, 진입 시의 시간을 측정해서 stopwatch 모드에서 구한 시간(기준값)을 뺀다. 차이값을 st_tm 구조제에 저장하고 min, sec, millisec 단위로 저장해서 출력한다.

```
void lap_time()
{
                if (btn == 'a') {
                        lap_time();
                }
                if (btn == 'b') {
                        st_info=START;
                }


                //display

                display_command = PRINT_LAP_TIME;
                btn = 0;
}

void stop()
{
                if (btn == 'a') {
                        st_info=0;
                }
                if (btn == 'b') {
                        st_info=START;
                        gettimeofday(&st_tv, NULL);
                        st_tv.tv_sec -= st_start.tv_sec;
                        st_start.tv_sec += (st_tv.tv_sec-st_stop.tv_sec);
                }

        //display
        display_command = PRINT_STOP;
        btn = 0;
}
```

Lap_time 은 호출될 때 당시의 시간을 출력하고 stopwatch 에서 지정한 기준 값의 변화는 없다.

stop 에서는 stop 함수가 호출될 때의 시간을 구해서 start 함수에서 구한 시간까지의 차이를 구한다. 구한 시간은 start 에서 현재까지 지난 시간이다. 이 시간을 저장하고 새로운 start 시간으로 지정해준다.

2.2.2 Beep Controller

```
void turn_off()
{
        if((alarm_info.alarm_power == true) && (alarm_tm.tm_hour == current_tm.tm_hour) && (alarm_tm.tm_min == current_tm.tm_min
)) al_info = TURN_ON;


}

void turn_on()
{
        if (btn == 'a' || btn == 'b' || btn == 'c' || btn == 'd' ) {
                alarm_tm = *localtime(&current_time);
                alarm_info.beep_time=0;
                alarm_info.alarm_power=false;
                alarm_info.display_alarm_indicator = ' ';
                al_info = TURN_OFF;
        }

        //display
        if(alarm_info.beep_time<5){
                alarm_info.beep_time++;
                printf("\a");
                sleep(1);
        }

        if(alarm_info.beep_time==5) {
                alarm_tm = *localtime(&current_time);
                alarm_info.beep_time=0;
                alarm_info.alarm_power=false;
                alarm_info.display_alarm_indicator = ' ';
                al_info = TURN_OFF;
        }
        btn = 0;
}
```

Turn off 는 알람이 켜져 있고, 알람 시간과 현재 시간의 minute, sec 가 동일하면 al_info 의 값을 TURN_ON 으로 바꾼다.

Turn on 은 모든 버튼 입력을 받거나, alarm_info 가 저장한 beep_time 이 5 가 되면 알람이 울린 시간, 알람 전원 여부, alarm_indicator 를 모두 off 로 만들고 al_info 를 TURN_OFF 로 만든다. 다른 상황에서는 알람을 울린다.

2.2.3 Backlight Controller

```
void idle() {
    if (btn == 'd') {
        bl_info=TURN_YELLOW;
    }
    btn = 0;
}

void turn_yellow() {

    display(PRINT_BACKLIGHT);

}
```

2.2.4 기타 함수

```
int getch()
{
        int c;
        struct termios oldattr, newattr;

        tcgetattr(STDIN_FILENO, &oldattr);
        newattr = oldattr;
        newattr.c_lflag &= ~(ICANON | ECHO);
        newattr.c_cc[VMIN] = 1;
        newattr.c_cc[VTIME] = 0;
        tcsetattr(STDIN_FILENO, TCSANOW, &newattr);
        c = getchar();
        tcsetattr(STDIN_FILENO, TCSANOW, &oldattr);
        return c;
}



int kbhit()
{
        struct termios oldt, newt;
        int oldf;

        tcgetattr(STDIN_FILENO, &oldt);
        newt = oldt;
        newt.c_lflag &= ~(ICANON | ECHO);
        tcsetattr(STDIN_FILENO, TCSANOW, &newt);
        oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
        fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);

        btn = getch();

        tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
        fcntl(STDIN_FILENO, F_SETFL, oldf);

        if (btn != EOF){return 1;}

        return 0;
}


void gotoxy(int x, int y) {

     printf("\033[%d;%df",y,x);

     fflush(stdout);

}
```

2.2.5 기타 함수

Display 함수는 display_command 에 맞는 출력화면을 출력한다.

```
void display(int display_command)
{
        //if(bl_info==TURN_YELLOW) printf("%c[1;33m", 27);
        switch(display_command) {

                case PRINT_BACKLIGHT:
                        if(backlight_time<5) {
                                printf("%c[1;33m", 27);
                                backlight_time++;
                        }
                        else {
                                backlight_time = 0;
                                bl_info=0;
                                printf("%c[0m", 27);
                        }
                        usleep(100000);
                        break;
                case PRINT_TIME_MODE:
                        system("clear");
                        gotoxy(70,20);
                        printf("%s %c %d-%d-%d %d:%d:%d\n", week_day[current_tm.tm_wday], alarm_info.display_alarm_indicator, cu
rrent_tm.tm_year + 1900, current_tm.tm_mon + 1, current_tm.tm_mday, current_tm.tm_hour, current_tm.tm_min, current_tm.tm_sec);
                        usleep(100000);
                        break;
                case PRINT_SEC_SET:
                        system("clear");
                        gotoxy(70,20);
                        printf("%s %c %d-%d-%d %d:%d:%c[4m%d%c[0m\n", week_day[current_tm.tm_wday], alarm_info.display_alarm_ind
icator, current_tm.tm_year + 1900, current_tm.tm_mon + 1, current_tm.tm_mday, current_tm.tm_hour, current_tm.tm_min, 27, current
_tm.tm_sec, 27);
                        usleep(100000);
                        break;
                case PRINT_HOUR_SET:
                        system("clear");
                        gotoxy(70,20);
                        printf("%s %c %d-%d-%d %c[4m%d%c[0m:%d:%d\n", week_day[current_tm.tm_wday], alarm_info.display_alarm_ind
icator, current_tm.tm_year + 1900, current_tm.tm_mon + 1, current_tm.tm_mday, 27, current_tm.tm_hour, 27, current_tm.tm_min, cur
rent_tm.tm_sec);
                        sleep(1);
                        break;
```

2.3 Main.c

```
#include "DWS.h"

void main() {

        // Data Storage 및 변수 초기화
        st_info = 0;
        al_info = TURN_OFF;
        mo_info = TIME_MODE;
        bl_info = 0;

        display_command = 0;

        // Alarm information 관련 변수
        alarm_information alarm_info;
        alarm_info.alarm_power = false;
        alarm_info.display_alarm_indicator = ' ';
        alarm_info.beep_time = 0;
        alarm_time = 1546268400;
        alarm_tm = *localtime(&alarm_time);

        // Stopwatch 관련 변수
        stop_min=0;
        stop_sec=0;
        stop_milisec=0;

        // Backlight 관련 변수
        backlight_time = 0;

        // 초기 시간 설정 바꾸기
        current_time = 1546268400;
        current_tm = *localtime(&current_time);

        timer_t timerID;
        createCurrent(&timerID,1,0);

        while (1) {


                if(kbhit()==1){

                }

                // Beep Controller
                if (al_info == TURN_OFF) turn_off();
                if (al_info == TURN_ON) turn_on();

                // Backlight Controller
               if (btn =='d' && bl_info == 0) idle();
                if (bl_info == TURN_YELLOW) turn_yellow();


                // Main Controller
                if(mo_info==TIME_MODE)
                {
                        if (st_info == 0) time_mode();
                        if (st_info == SEC_SET) sec_set();
                        if (st_info == HOUR_SET) hour_set();
                        if (st_info == MINUTE_SET) minute_set();
                        if (st_info == YEAR_SET) year_set();
                        if (st_info == MONTH_SET) month_set();
                        if (st_info == DAY_SET) day_set();

                }
                else if (mo_info == ALARM_MODE) {

                        if (st_info == 0) alarm_mode();
                        if (st_info == AL_HOUR_SET) hour_set();
                        if (st_info == AL_MINUTE_SET ) minute_set();

                }
                else if (mo_info == STOPWATCH_MODE){

                        if(st_info == 0) stopwatch_mode();
                        if(st_info == START) start();
                        if(st_info == LAP_TIME) lap_time();
                        if(st_info == STOP) stop();

                }

                display(display_command);

        }


}
```

3 Difficulty in Implementation

3.1 current 함수

처음 구현 시 DWS 메인 루프 자체가 1초에 한번씩 출력을 했기 때문에 current 함수 자체도 1초에 한 번씩 sec 변수를 증가시켰다. 하지만 stopwatch 모드를 구현하면서 1초에 한 번씩 출력하는 구현 방식이 잘못되었다는 것을 알게 되었다. Current 함수를 모드에 독립적으로 1초에 한번씩 실행하기 위한 방법을 생각하게 되었고, 타이머를 이용해 1초에 한번씩 호출하는 방법으로 구현하게 되었다.

3.2 stopwatch 구현

start 모드에서 시간이 ms 단위로 출력되는 것이 어려웠다. Stopwatch가 시작될 때 시스템 시간을 초단위의 long int(time_t)로 받아 변수에 저장시킨다. Start 함수 내부에서 함수 불릴 때 마다 현재시간을 timeval을 이용해 ms 단위의 시간을 받아 stopwatch가 시작될 때 저장시킨 변수를 빼줘서 출력시켰다.

Stop 모드에서 정지된 시간만큼을 Stopwatch가 시작될 때 저장한 변수에 더해줬다.