Introduction to Software Engineering

Team Project

-Digital Watch Final-

컴퓨터공학과 201611262 박동현
컴퓨터공학과201611272 안찬우
컴퓨터공학과201611300 조승현
컴퓨터공학과201611308 최준오

Contents

- Introduction
- Overall DFD
- Development Detail
- Difficulties
- QnA

Introduction

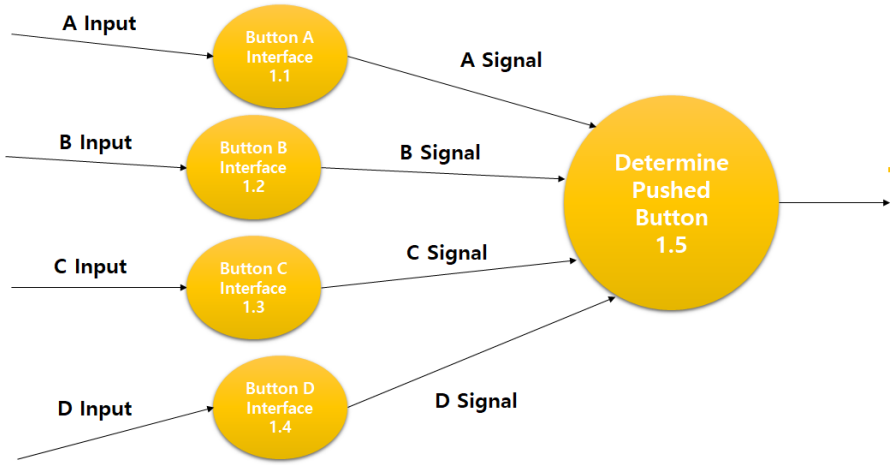
- Development Environment

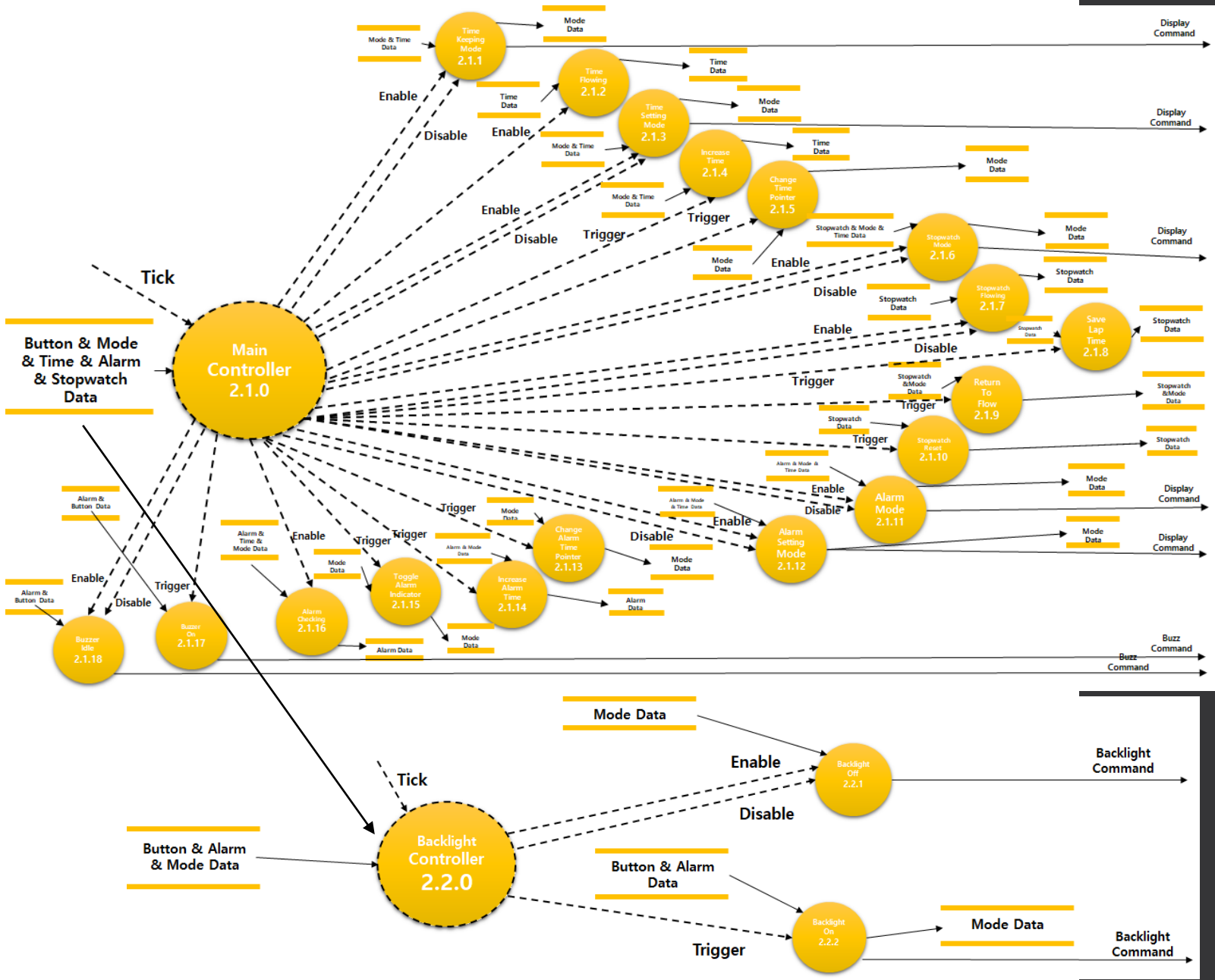
: Ubuntu, VSCode, GitHub

- Main Point

: Signal Handler, Thread, Setitimer

Overall DFD


A Input
B Input
C Input
D Input
Button A Interface 1.1
Button B Interface 1.2
Button C Interface 1.3
Button D Interface 1.4
A Signal
B Signal
C Signal
D Signal
Determine Pushed Button 1.5


Tick
Button & Mode & Time & Alarm & Stopwatch Data
Main Controller 2.1.0
Time Keeping Mode 2.1.1
Time Flowing 2.1.2
Time Setting Mode 2.1.3
Increase Time 2.1.4
Change Time Pointer 2.1.5
Stopwatch Mode 2.1.6
Stopwatch Flowing 2.1.7
Save Lap Time 2.1.8
Return To Flow 2.1.9
Stopwatch Reset 2.1.10
Alarm Mode 2.1.11
Alarm Setting Mode 2.1.12
Change Alarm Time Pointer 2.1.13
Increase Alarm Time 2.1.14
Toggle Alarm Indicator 2.1.15
Alarm Checking 2.1.16
Buzzer On 2.1.17
Buzzer Idle 2.1.18
Enable
Disable
Trigger
Mode Data
Time Data
Display Command
Stopwatch Data
Alarm Data
Buzz Command

Tick
Button & Alarm & Mode Data
Backlight Controller 2.2.0
Backlight Off 2.2.1
Backlight On 2.2.2
Enable
Disable
Trigger
Mode Data
Button & Alarm Data
Backlight Command

Timer

```
static void timer_handler( int sig, siginfo_t *si, void *uc ) {
    timer_t *tidp;
    tidp = si->si_value.sival_ptr;

    if ( *tidp == firstTimerID ) {

        time_flowing(time_data);
        alarm_checking(alarm_data, mode_data, time_data);
    }
    else if ( *tidp == secondTimerID ) {

        stop_watch_flowing(stopwatch_data, mode_data);
    }
}
```

Timer

```
static int install_timer (char *name, timer_t *timerID, int sec, int nsec) {
    struct sigevent         te;
    struct itimerspec       its;
    struct sigaction        sa;
    int                     sigNo = SIGRTMIN;

    /* Set up signal handler. */
    sa.sa_flags = SA_SIGINFO;
    sa.sa_sigaction = timer_handler;
    sigemptyset(&sa.sa_mask);

    if (sigaction(sigNo, &sa, NULL) == -1) {
        printf("sigaction error\n");
        return -1;
    }
```

```
    /* Set and enable alarm */
    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 = nsec;
    its.it_value.tv_sec = sec;
    its.it_value.tv_nsec = nsec;
    timer_settime(*timerID, 0, &its, NULL);

    return 0;
}
```

Time Flowing

```
void time_flowing(timedata_t *curr_time) {
    if (++(curr_time->sec) == 60) {
        curr_time->min++;
        curr_time->sec = 0;
    }
    if (curr_time->min == 60) {
        curr_time->hour++;
        curr_time->min = 0;
    }
    if (curr_time->hour == 24) {
        curr_time->date++;
        curr_time->hour = 0;
    }
    if (curr_time->month == 2) {
        if (leap_year(curr_time->year)){
            if (curr_time->date == 30) {
                curr_time->date == 1;
                curr_time->month++;
            }
        }
        else {
            if (curr_time->date == 29) {
                curr_time->date == 1;
                curr_time->month++;

 else if (curr_time->month==1||curr_time->month==3|
|curr_time->month==5||curr_time->month==7||curr_t
ime->month==8||curr_time->month==10||curr_time->
month==12) {
            if (curr_time->date == 32) {
                curr_time->date == 1;
                curr_time->date++;
            }
        }
        else {
            if (curr_time->date == 31) {
                curr_time->date == 1;
                curr_time->date++;
            }
        }
}
```

Determine Pushed Button

```
//get keyboard input in nonblocking manner
int determine_pushed_button() {
  char ch;
  int nread;
  int ich;
    new_term.c_cc[VMIN] = 0;
    new_term.c_cc[VTIME] = 0;
    tcsetattr(0, TCSANOW, &new_term);
    nread = read(0, &ch, 1);
    new_term.c_cc[VMIN] = 1;
    new_term.c_cc[VTIME] = 0;
    tcsetattr(0, TCSANOW, &new_term);
    if (nread >= 1) {
      ch_read = (int) ch;
      return ch_read;
    }
    else {
      return -1;
    }
}
```

Time Keeping Mode

```
int time_keeping_mode(modedata_t *modedata, timedata_t *curr_time) {
    char button;
    while (1) {
        backlight_curr_sec = curr_time->sec;
        if (backlight_curr_sec == backlight_finish_sec) {
            printf("₩033[0m"); //reset
        }

        if (!(modedata->buzzing)&&((button = determine_pushed_button()) > 0)) {
            if (button == 'A') { //change to time setting mode
                modedata->s_m_data = 1;
                time_setting_mode(modedata, curr_time);
            }
            else if (button == 'C') { //change to alarm mode
                modedata->mode = 1;
                return 0;
            }
```

```
            else if (button == 'D') {
                        if
(backlight_curr_sec == 58 ||
backlight_curr_sec == 59)

backlight_finish_sec =
(backlight_curr_sec + 2 )- 60;
                        else {

backlight_finish_sec =
backlight_curr_sec + 2;
                        }
                        printf("₩033[1;33m");
//yellow

                    }
                    else {
                        continue;
                    }
                }

                system("clear");
                printf("%s-%04d-%02d-
%02d₩n%02d:%02d:%02d %02d₩n", curr_time-
>day[get_day(curr_time->year,
curr_time->month, curr_time->date)],
curr_time->year, curr_time->month,
curr_time->date,
                        curr_time->hour,
curr_time->min, curr_time->sec);
                usleep(100000);
            }
}
```

Time Setting Mode

```c
void time_setting_mode(modedata_t *modedata, timedata_t *curr_time) {
    char button;
    while (1) {
        backlight_curr_sec = curr_time->sec;
        if (backlight_curr_sec == backlight_finish_sec) {
            printf("₩033[0m"); //reset
        }
        if(!(modedata->buzzing)&&((button = determine_pushed_button()) > 0)) {
            if (backlight_curr_sec == backlight_finish_sec) {
                printf("₩033[0m"); //reset
            }
            if (button == 'A') { //return to to time keeping mode
                modedata->s_m_data = 0;
                modedata->time_pointer == 0;
                return;
            }
            else if (button == 'B') { //change to alarm mode
                increase_time(modedata, curr_time);
            }
            else if (button == 'C') { //change cursor
                change_time_pointer(modedata);
            }
            else if (button == 'D') {
                if (backlight_curr_sec == 58 || backlight_curr_sec == 59)
                    backlight_finish_sec = (backlight_curr_sec + 2 )- 60;
                else {
                    backlight_finish_sec = backlight_curr_sec + 2;
                }
                printf("₩033[1;33m"); //yellow
            }
        }
```

```c
        switch (modedata->time_pointer) {
            case 0: //flicker second
                system("clear");
                printf("%04d-%02d-%02d₩n%02d:%02d %02d₩n", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->min, curr_time->sec);
                usleep(50000);
                system("clear");
                printf("%04d-%02d-%02d₩n%02d:%02d   ₩n", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->min);
                usleep(50000);
                break;
            case 1: //flicker hour
                system("clear");
                printf("%04d-%02d-%02d₩n%02d:%02d %02d₩n", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->min, curr_time->sec);
                usleep(50000);
                system("clear");
                printf("%04d-%02d-%02d₩n  :%02d %02d₩n", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->min, curr_time->sec);
                usleep(50000);
                break;
```

Time Setting Mode

```
case 2: //flicker minute
                system("clear");
                printf("%04d-%02d-%02dWn%02d:%02d %02dWn", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->min, curr_time->sec);
                usleep(50000);
                system("clear");
                printf("%04d-%02d-%02dWn%02d:   %02dWn", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->sec);
                usleep(50000);
                break;
case 3: //flicker year
                system("clear");
                printf("%04d-%02d-%02dWn%02d:%02d %02dWn", curr_time->year,
curr_time->month, curr_time->date,
                        curr_time->hour, curr_time->min, curr_time->sec);
                usleep(50000);
                system("clear");
                printf("    -%02d-%02dWn%02d:%02d %02dWn", curr_time-
>month, curr_time->date,
                        curr_time->hour, curr_time->min, curr_time->sec);
                usleep(50000);
                break;

                break;
        }

    }
}
```

```
                case 4: //flicker month
                    system("clear");
                    printf("%04d-%02d-%02dWn%02d:%02d %02dWn",
curr_time->year, curr_time->month, curr_time->date,
                            curr_time->hour, curr_time->min, curr_time-
>sec);
                    usleep(50000);
                    system("clear");
                    printf("%04d-  -%02dWn%02d:%02d %02dWn", curr_time-
>year, curr_time->date,
                            curr_time->hour, curr_time->min, curr_time-
>sec);
                    usleep(50000);
                    break;
                case 5: //flicker date
                    system("clear");
                    printf("%04d-%02d-%02dWn%02d:%02d %02dWn",
curr_time->year, curr_time->month, curr_time->date,
                            curr_time->hour, curr_time->min, curr_time-
>sec);
                    usleep(50000);
                    system("clear");
                    printf("%04d-%02d-  Wn%02d:%02d %02dWn", curr_time-
>year, curr_time->month,
                            curr_time->hour, curr_time->min, curr_time-
>sec);
                    usleep(50000);
                    break;
            }

        }
}
```

Increase Time

```
void increase_time(modedata_t *modedata, timedata_t *set_time) {
    int cursor = modedata->time_pointer;
    int month = set_time->month;
    int year = set_time->year;

    switch(cursor) {
        case 0:
            set_time->sec = ((set_time->sec + 1)%60);
            break;
        case 2:
            set_time->min = ((set_time->min + 1)%60);
            break;
        case 1:
            set_time->hour = ((set_time->hour +1)%24);
            break;
```

```
        case 5:
                if (month==2) {
                    if (leap_year(year)){
                        set_time->date = ((set_time->date)%29 + 1);
                    }else{
                        set_time->date = ((set_time->date)%28 + 1);
                    }
                }
                else if
(month==1||month==3||month==5||month==7||month==8||month==10||month==12) {
                    set_time->date = ((set_time->date)%31 +1);
                }
                else {
                    set_time->date = ((set_time->date)%30 + 1);
                }
                break;
            case 4:
                set_time->month = ((set_time->month)%12 + 1);
                if(set_time->month == 2 && set_time->date > 28
&& !(leap_year(year))) { //윤년아닐 때
                    set_time->date = 28;
                }
                else if(set_time->month == 2 && set_time->date > 29 &&
leap_year(year)){ //윤년일 때
                    set_time->date = 29;
                }else if(set_time->date > 30 && (set_time->month != 8 ||
set_time->month != 1)){
                    set_time->date = 30;
                }
                break;
            case 3:
                set_time->year = ((((set_time->year+1)-2019)%81)+2019);
        }
}
```

Alarm Mode

```
int alarm_mode(modedata_t *modedata, alarmdata_t *alarmdata, timedata_t *timedata) {
    char button;
    while (1) {
        backlight_curr_sec = timedata->sec;
        if (backlight_curr_sec == backlight_finish_sec) {
            printf("₩033[0m"); //reset
        }
        if(!(modedata->buzzing)&&((button = determine_pushed_button()) > 0)) {
            if (button == 'A') {
                modedata->s_m_data = 2;
                alarm_setting_mode(modedata, alarmdata, timedata);
            } //change to alarm setting mode
            else if (button == 'B') {
                if (modedata->alarm_indicator_data)
                    modedata->alarm_indicator_data = false;
                else
                    modedata->alarm_indicator_data = true;
            } //toggle alarm
            else if (button == 'C') {
                modedata->mode = 2;
                return 0;
            } //change to stopwatch mode
            else if (button == 'D') {
                if (backlight_curr_sec == 58 || backlight_curr_sec ==
59)
                    backlight_finish_sec = (backlight_curr_sec + 2 )-
60;
                else {
                    backlight_finish_sec = backlight_curr_sec + 2;
                }
                printf("₩033[1;33m"); //yellow
            }
        }
        if (modedata->alarm_indicator_data == 0) {
            system("clear");
            printf("   %02d-%02d₩n%02d:%02d₩n", timedata->month,
timedata->date,
                    alarmdata->al_hour, alarmdata->al_minute);
            usleep(100000);
        }
        else {
            system("clear");
            printf("AL %02d-%02d₩n%02d:%02d₩n", timedata->month,
timedata->date,
                    alarmdata->al_hour, alarmdata->al_minute);
            usleep(100000);
        }
    }

}
```

Alarm Checking /Buzz / Buzz handle

```
void alarm_checking(alarmdata_t *alarmdata, modedata_t *modedata, timedata_t *curr_time) {
    if (modedata->alarm_indicator_data&&alarmdata->al_hour == curr_time->hour && alarmdata->al_minute == curr_time->min && curr_time->sec == 0){
        modedata->buzzing = true;
        raise(SIGUSR1);
    }
}
```

```
void *buzz(void *arg) {
    int curr_sec;
    int finish_sec=5;

    char button;
    if(mode_data->s_m_data!=1 && mode_data->s_m_data!=2){
    while (1) {
        if ((button = determine_pushed_button()) > 0) {
          mode_data->buzzing=false;
            return;
        } //if button is pressed, stop buzzing
       curr_sec=time_data->sec;
        if (curr_sec >= finish_sec) {
            mode_data->buzzing=false;
            return;
        }
        printf("Wa");

    }
    }
    mode_data->buzzing=false;
    return;
}
```

```
void buzzer(int signum) {
    if (signum == SIGUSR1) {
        pthread_t buzz_thread;
       pthread_create(&buzz_thread, NULL, buzz, NULL);


    }

}
```

Stopwatch Mode /Laptime

```
int stopwatch_mode(stopwatchdata_t* stopwatch_data, modedata_t*
modedata, timedata_t* timedata) {
    char button;
    pthread_t sttime_pid;


    while (1) {
        backlight_curr_sec = timedata->sec;
        if (backlight_curr_sec == backlight_finish_sec) {
            printf("\033[0m"); //reset
        }
        if(!(modedata->buzzing)&&((button = determine_pushed_button()
) > 0)) {

            if (button == 'C') {
                modedata->mode = 0;
                return 0;
            }  //change to timekeeping mode
            else if (button == 'D') {
                if (backlight_curr_sec == 58 || backlight_curr_sec == 59)
                    backlight_finish_sec = (backlight_curr_sec + 2 )- 60;
                else {
                    backlight_finish_sec = backlight_curr_sec + 2;
                }
                printf("\033[1;33m"); //yellow
            }
```

```
//stopwatch mode
            if (modedata->lap_time_state == 0) {
                if (button == 'B') {
                            modedata->stopwatch_state=(++mode
data->stopwatch_state)%2;
                } // start stopwatch
                else if (button == 'A') {
                    //stopwatch running
                    if (modedata->stopwatch_state == 1) {
                        modedata->lap_time_state = 1;
                        stopwatch_data->lap_centisecond = stopwa
tch_data->centisecond;
                        stopwatch_data->lap_second = stopwatch_
data->second;
                        stopwatch_data->lap_minute = stopwatch_
data->minute;
                    } //save lap time
                    //stopwatch not running
                    else {
                        stopwatch_data->centisecond = 0;
                        stopwatch_data->second = 0;
                        stopwatch_data->minute = 0;
                    } // reset stopwatch
                }
            }
```

Stopwatch Mode /Laptime

```
    //laptime mode
        else {
            if (button == 'B') {
                modedata->lap_time_state = 0;
            } //return to stopwatch mode
            else if (button == 'A') {
                stopwatch_data->lap_centisecond = stopwatch_data->centis
econd;
                stopwatch_data->lap_second = stopwatch_data->second;
                stopwatch_data->lap_minute = stopwatch_data->minute;
            } //save lap time
        }
    }
    if (modedata->lap_time_state == 0) { //stopwatch time display
        system("clear");
        printf("ST %02d:%02d\n%02d\'%02d\"%02d\n", timedata->hou
r, timedata->min,
                stopwatch_data->minute, stopwatch_data->second, stopwat
ch_data->centisecond);
        usleep(10000);
    }
    else { //lap time display
        system("clear");
        printf("ST %02d:%02d\n%02d\'%02d\"%02d\n", timedata->hou
r, timedata->min,
                stopwatch_data->lap_minute, stopwatch_data->lap_second,
stopwatch_data->lap_centisecond);
        usleep(10000);
```

BackLight

```
int backlight_curr_sec = 0;
int backlight_finish_sec = 0;

int time_keeping_mode(modedata_t *modedata, timedata_t *curr_ti
me) {
    char button;
    while (1) {
        backlight_curr_sec = curr_time->sec;
        if (backlight_curr_sec == backlight_finish_sec) {
            printf("\033[0m"); //reset
        }

        if (!(modedata->buzzing)&&((button = determine_pushed_button
()) > 0)) {
            if (button == 'A') { //change to time setting mode
                modedata->s_m_data = 1;
                time_setting_mode(modedata, curr_time);
            }
            else if (button == 'C') { //change to alarm mode
                modedata->mode = 1;
                return 0;
            }
            else if (button == 'D') {
                if (backlight_curr_sec == 58 || backlight_curr_sec == 59)
                    backlight_finish_sec = (backlight_curr_sec + 2 )- 60;
                else {
                    backlight_finish_sec = backlight_curr_sec + 2;
                }
                printf("\033[1;33m"); //yellow

            }
            else {
                continue;
            }
        }

        system("clear");
        printf("%s-%04d-%02d-%02d\n%02d:%02d %02d\n", curr_time-
>day[get_day(curr_time->year, curr_time->month, curr_time->date)],
curr_time->year, curr_time->month, curr_time->date,
                curr_time->hour, curr_time->min, curr_time->sec);
        usleep(100000);
    }
}
```

Main controller

```
void *main_controller(void *arg) {
    //circulate between modes: "timekeeping mode", "alarm mode", "stop
watch mode"
    while (1) {
        time_keeping_mode(mode_data, time_data);
        alarm_mode(mode_data, alarm_data, time_data);
        stopwatch_mode(stopwatch_data, mode_data, time_data);
    }

    return 0;
}
```

개발 도중 어려웠던 점

1. 시간 오차 발생

2. 쓰레드 간 동기화 문제

QnA