DWS PRESENTATION #3
FINAL PRESENTATION

TEAM #1

201711376 김경진

201810568 박용준

201811263 백종원

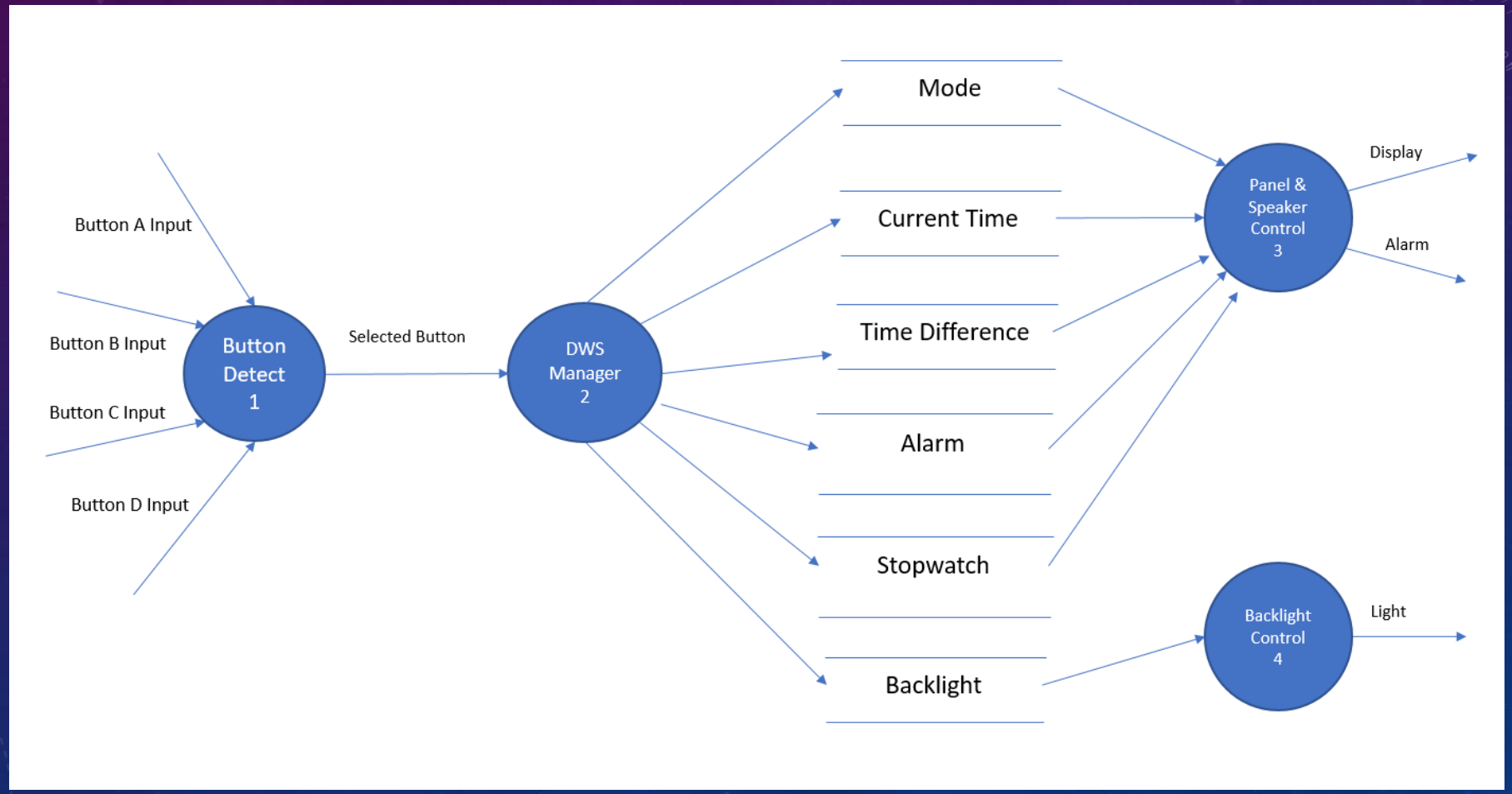
목차

- DFD 변경 사항
- 구현 사항
  - 개발 환경
  - Data Store
  - 구현 기능들
  - 헤더 파일들
  - 결과
- 소감
  - 개인 소감
  - 프로젝트 리뷰

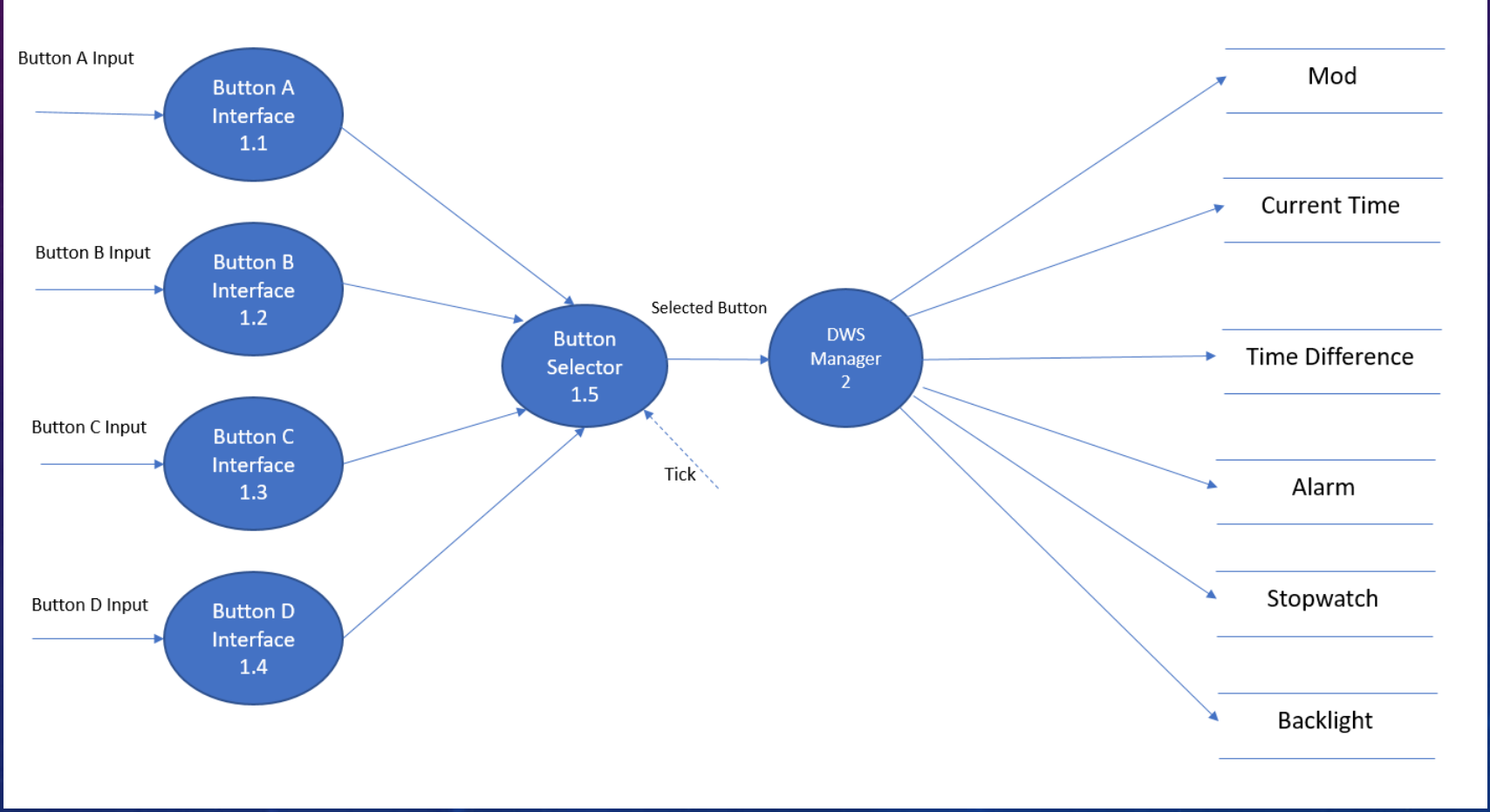
DFD의 주요 변경 사항

- Data Store 변경
  - Time Difference 가 추가됨 : 절대 시간과 DW에서 설정된 시간과의 시간 차이를 저장
- initialTime 추가
  - 구현에서 설명
- Ring Changer 추가
  - 구현에서 설명

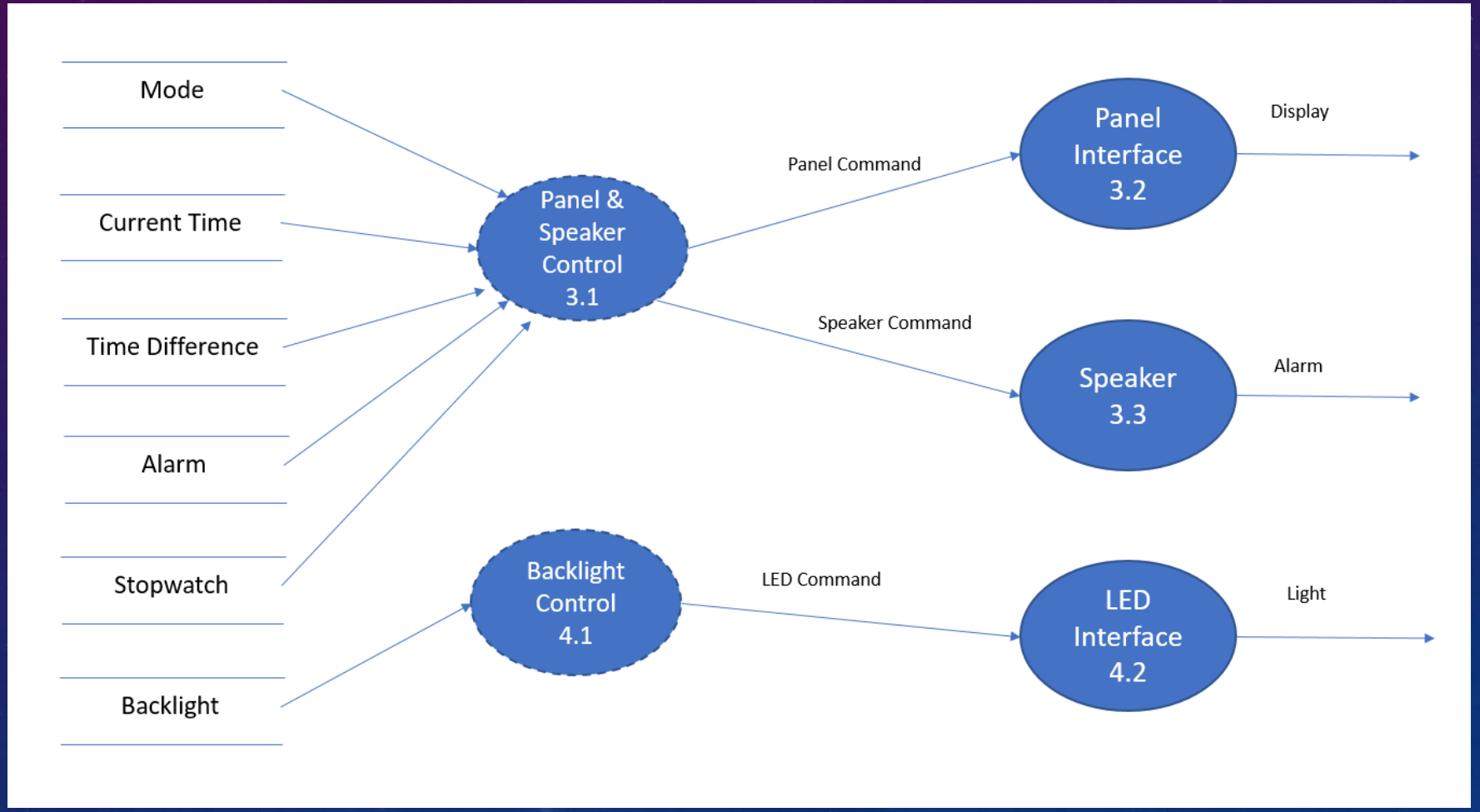
DFD LEVEL 1


Button A Input
Button B Input
Button C Input
Button D Input
Button Detect 1
Selected Button
DWS Manager 2
Mode
Current Time
Time Difference
Alarm
Stopwatch
Backlight
Panel & Speaker Control 3
Display
Alarm
Backlight Control 4
Light

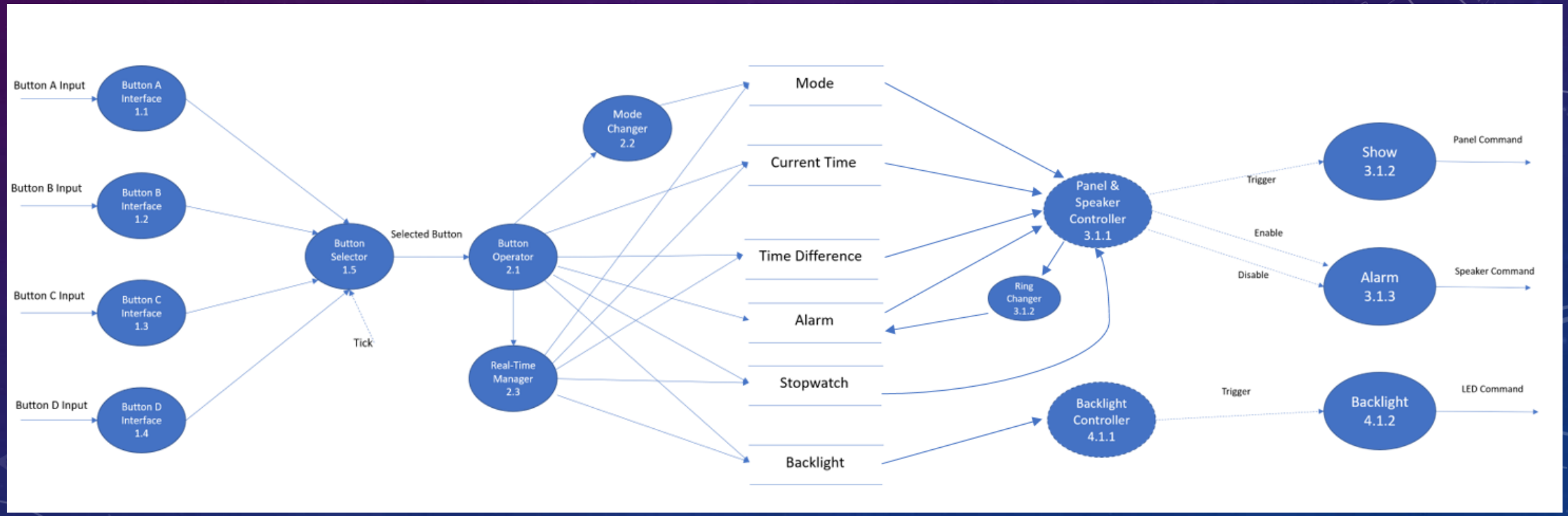
DFD LEVEL 2


Button A Input
Button A Interface 1.1
Button B Input
Button B Interface 1.2
Button C Input
Button C Interface 1.3
Button D Input
Button D Interface 1.4
Button Selector 1.5
Selected Button
Tick
DWS Manager 2
Mod
Current Time
Time Difference
Alarm
Stopwatch
Backlight

DFD LEVE2(CONT'D)



DFD LEVEL 3

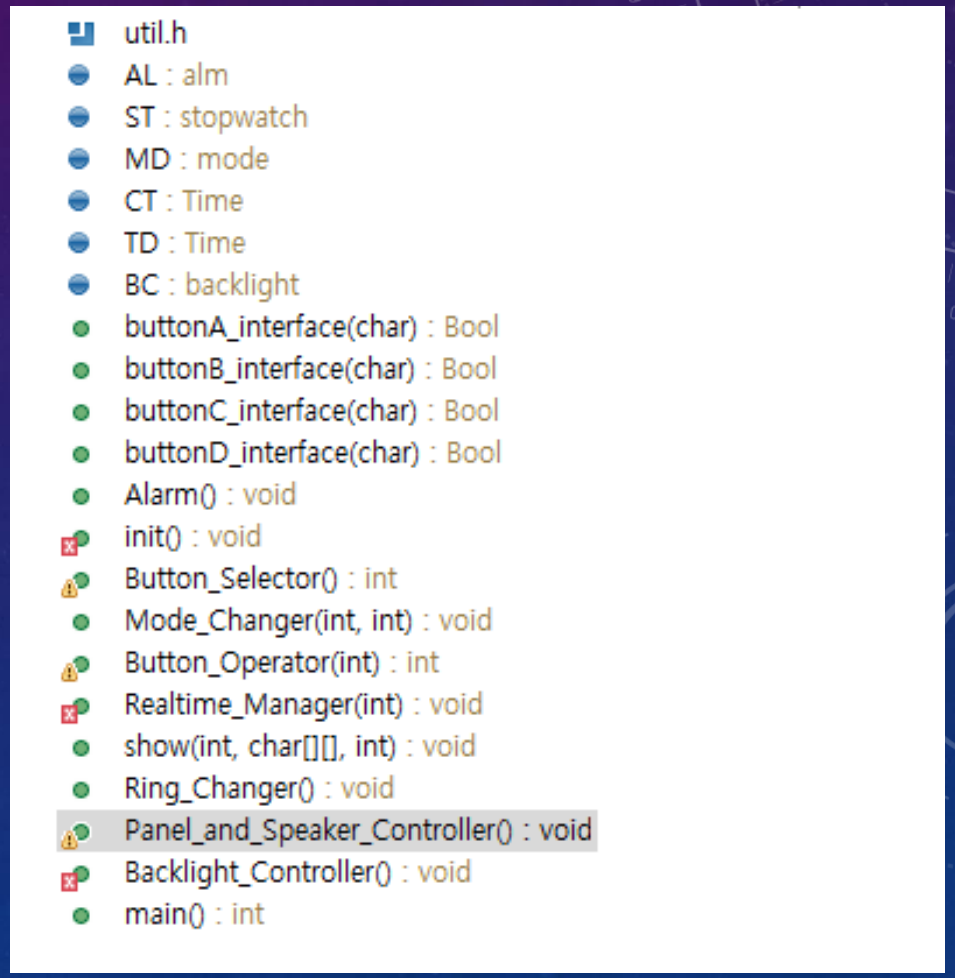

Button A Input
Button A Interface 1.1
Button B Input
Button B Interface 1.2
Button C Input
Button C Interface 1.3
Button D Input
Button D Interface 1.4
Button Selector 1.5
Tick
Selected Button
Button Operator 2.1
Mode Changer 2.2
Real-Time Manager 2.3
Mode
Current Time
Time Difference
Alarm
Stopwatch
Backlight
Panel & Speaker Controller 3.1.1
Ring Changer 3.1.2
Backlight Controller 4.1.1
Trigger
Enable
Disable
Show 3.1.2
Alarm 3.1.3
Panel Command
Speaker Command
Trigger
Backlight 4.1.2
LED Command

개발 환경

- Compiler : GCC Cygwin
- IDE : Eclipse for C
- 버전 관리
  - Github Repository : IntroductionToSE
- Files
  - alpha.c
  - Linux_kbhit.h
  - getch.h
  - util.h

구현 기능들

- Init
- Button Interface(s)
- Button Selector
- Button Operator
- Realtime Manager
- Panel and Speaker Controller
  - Show
  - Alarm
- Backlight Controller
- Mode Changer
- Ring Changer
- Main



DATA STORE : TYPE DEFINITION

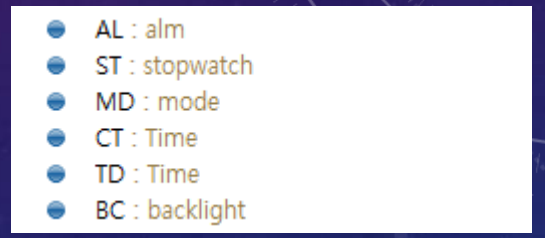
- DFD 상의 Data Store들을 구현하기 위해 struct들을 정의하였다.
- 정의된 구조체들은 “util.h”에 구현되어 있다.
  - Mode
  - Time
  - Alm
  - Stopwatch
  - Backlight

```
typedef struct Mode{
    Bool alarm_buzzing;
    int category_alpha; //대분류
    int category_beta; //소분류
    Bool stopwatch_indicator; //indicator 켜졌는지,
    Bool alarm_indicator; //indicator 꺼졌는지,
}mode;
typedef struct Time{
    int YY; //년도
    int MT; //월
    int DD; //일
    int HH; //시
    int MM; //분
    int SS; //초
    int MS; //ms
    int WD; //요일
}Time;
typedef struct Alm{ //시작 시간
    Time alarmTime;
    Bool snooze; // snooze는 0 < SS < 5인 경우에서,
    //Selected_Button이 0이 아닌 경우
    //잠시 false가 됩니다.(잠시 : SS < 5까지)
    //이외의 경우에서는 계속 true가 대입됩니다.
    int ring; // CT.SS와 비교되어 알람 소리를 일정하게 내는 역할을 합니다
}alm;
typedef struct StopWatch {
    //LapTime은 StartTime을 기반으로 업데이트 된다.
    Time stopwatchTime;
    Time startTime;
    Time lapTime;
    Time initialTime; // commit 안 됨

}stopwatch;
typedef struct Backlight{
    Time BacklightTime; //backlight 켜기 시작한 시간
    int value; //backlight 색깔 값
}backlight;
```

DATA STORE : DESCRIPTION

- DFD 상의 Data Store들은 사진과 같이 alpha.c에서 전역에 구조체로 선언한다.
- 각각의 구조체가 가지는 의미는 아래와 같다.
  - alm AL : Alarm
  - stopwatch ST : Stopwatch
  - mode MD : Mode
  - time CT : Current Time
  - time TD : Time Difference
  - backlight BC : Backlight



INIT

- 프로그램의 시작과 동시에 실행되는 함수, 여러 초기화를 진행한다.
- 절대 시간 불러오기
  - timb를 활용하여 ms 단위 까지 시간을 구해준다. 이는 TD의 초기화에 쓰인다.
- Data Store 초기화
  - Data Store : AL, ST, MD, CT, TD, BC 들에 초기값을 대입하여 초기화
- Console 초기화
  - 콘솔 창 이름 설정
  - 콘솔 창 마우스 커서 숨기기
  - 콘솔 창에 시계 기본 화면 그리기

BUTTON INTERFACE(S)

- 4개의 버튼 A,B,C,D에 대한 interface
  - buttonA_interface, buttonB_interface, buttonC_interface, buttonD_inteface
- 키보드로 입력을 받았다면, 해당하는 버튼의 interface가 true를 return
- Button_Operator에서 호출되어 사용된다.

BUTTON SELECTOR

- linux_kbhit()를 통해 키보드 입력을 대기한다.
  - linux_kbhit() : 입력 버퍼가 비어 있는지, 아닌지를 판별하여 0또는 1을 리턴한다
- 입력이 있었다면 while문을 수행한다.
  - getch()를 통해 버퍼의 첫번째 글자를 가져온다.
  - 가져온 글자를 buttonD_interface, buttonC_interface, buttonB_interface, buttonA_interface 순으로 호출한다.
  - 각 interface를 거치면서 true가 리턴이 된다면 Selected Button을 해당 버튼으로 설정하고 break;
- 입력이 없었다면 Selected Button으로 No Button을 return;
- Selected Button은 No Button/A/B/C/D에 대해 0,1,2,3,4로 대응되어 처리합니다.

MODE

- Button Operator는 Mode와 Selected Button에 따라 작동합니다.
- Mode는 다음과 같은 요소로 구성됩니다.
  - category_alpha
    - 대분류로서 timekeeping, stopwatch, alarm 3개 가능
  - category_beta
    - 대분류 별 세부적인 기능에 대한 소분류
  - stopwatch_indicator
    - stopwatch가 켜졌는지
  - alarm_indicator
    - alarm이 켜졌는지
  - alarm_buzzing
    - 알람이 울리고 있는지

BUTTON OPERATOR

- 어떠한 경우에든 Alarm_Buzzing을 최우선적으로 처리
  - Alarm_Buzzing이 true일 경우 버튼에 대한 처리를 생략하고 Realtime Manaer로 이동.
- 모든 모드에서 D를 누르면 Backlight가 켜집니다.
  - 이때 BC.value를 노란색으로 변경, BC.BacklightTime을 현재 시간에서 2초 뒤로 설정합니다.
- 기본 모드의 경우
  - C를 누르면 모드 변경(Timekeeping - > Stopwatch -> Alarm)
- Data Store의 정보들을 적절히 조합하여 명령 처리
  - Mode, Selected Button 에 따라서 생기는 케이스들에 대해 적합하게 처리하도록 구현되었다.
- 버튼의 역할에 따라 Mode를 미리 세분화하였습니다.
- 해야 할 명령들은 모두 switch/case 문으로 구현하였습니다.
- 모드를 변경해야 할 경우, Mode Changer를 통해 변경할 수 있습니다.

BUTTON OPERATOR-TIMEKEEPING

- Timekeeping 에서는 모드가 다음과 같이 세분화 됩니다.
  - 기본 모드 : 현재 시간을 출력한다.
  - 시간 변경 : 명세된 시간 변경 수행(초, 시간, 분, 년 ,월, 일 (요일)-> 초)
- Button Operator에서는 세분화된 모드에 따라 값, 모드 변경 등의 명령을 수행합니다.

BUTTON OPERATOR-STOPWATCH

- Stopwatch에서는 모드가 다음과 같이 세분화 됩니다.
  - 스톱워치 모드 : 스톱 워치 기능을 수행합니다.
    - 스톱워치 시작, 정지, 재개, 초기화 기능
  - Lap 모드 : Lap Time을 기록한다.
    - Lap time 갱신 기능

BUTTON OPERATOR-ALARM

- AIarm에서는 모드가 다음과 같이 세분화 됩니다.
  - 기본 모드 : 알람 시각을 표시
  - 알람 시각 변경 모드 : 알람 시각을 변경(시, 분, 시)

REAL TIME MANAGER

- 자연적인 시간 흐름을 구현하기 위한 프로세스
- CT 생성
  - timeb를 활용하여 절대 시간(표준 시) 을 불러온다.
  - DW에서 표시될 시간과의 차이인 TD를 기반으로 CT 생성
- CT를 기반으로 동작
  - Alarm
  - Stopwatch
  - Backlight
- TD를 제외한 모든 Time들은 시간 범위가 조정된다.
  - 유효한 시간 범위인지 확인하기 위함. ex) 15일 26시 -3분 -> 16일 01시 57분
  - timeCheck(Time *) 함수를 활용한다. “util.h”에 구현되어 있다.

```
// 1. CT를 동기화 (int 연산들을 시간 범위 내로 맞춰줌)
struct timeb itb;
struct tm *now;
time_t ltime;
int milisec;
ftime(&itb);
ltime = itb.time;
milisec = itb.millitm;
now = localtime(&ltime);

//시간 계산
CT.YY = now->tm_year - TD.YY;
CT.MT = now->tm_mon - TD.MT;
CT.DD = now->tm_mday - TD.DD;
CT.HH = now->tm_hour - TD.HH;
CT.MM = now->tm_min - TD.MM;
CT.SS = now->tm_sec - TD.SS;
CT.MS = milisec - TD.MS;
//여기서 CT가 2099년을 초과하였는지 체크하여야 합니다. 만약 자연스럽게 시간이 흘러 2100년이 된다면, 아래에 보정하는 과정이
//일어나기 전에 CT.YY에는 잠시 100이상의 값이 저장됩니다. 보정이 끝나면 자연스럽게 2019년 이상으로 바뀌겠지만,
//TD의 경우 자연스럽게 바뀌는 과정이 없습니다. 다시말해, CT.YY >= 100인 경우 자연스럽게 CT.YY -= 81이 되지만,
//TD의 값은 변화가 없습니다. 그러므로 자연스럽게 2099년을 초과하는 경우는 TD.YY도 81만큼 빼 주어야 합니다.
//초과하지 않았따면, CT만 보정하여 주면 됩니다.

//CT 보정
CT = timeCheck(&CT);
```

REALTIME MANAGER-ALARM

- 알람은 정각에 울리는 정보의 특성 상 현재 시각의 초가 0~5일 때 울린다
- if(현재시 == 알람시 && 현재분 == 알람분 && 현재초<5)
  - alarm_buzzing을 true로 바꾼다
- if (Selected_Button != 0), 즉 버튼이 입력된다면
  - alarm_buzzing = false
- Ring
  - 알람이 울리는 중이 아니라면 0이 계속 대입된다

REALTIME MANAGER-STOPWATCH

- 스톱워치 시간 = 현재 시간 – 스톱워치 시작 시간 + initlalTime 으로 처리합니다.

```
CT = timeCheck(&CT);
if (MD.stopwatch_indicator) {
        ST.stopwatchTime.YY = CT.YY - ST.startTime.YY + ST.initialTime.YY;
        ST.stopwatchTime.MT = CT.MT - ST.startTime.MT + ST.initialTime.MT;
        ST.stopwatchTime.DD = CT.DD - ST.startTime.DD + ST.initialTime.DD;
        ST.stopwatchTime.HH = CT.HH - ST.startTime.HH + ST.initialTime.HH;
        ST.stopwatchTime.MM = CT.MM - ST.startTime.MM + ST.initialTime.MM;
        ST.stopwatchTime.SS = CT.SS - ST.startTime.SS + ST.initialTime.SS;
        ST.stopwatchTime.MS = CT.MS- ST.startTime.MS + ST.initialTime.MS;
```

- 만약 60분을 넘겼다면, 관련된 시간(StopwatchTime,StartTime,Laptime,InitialTime)을 전부 0으로

```
if(ST.stopwatchTime.HH >= 60) { // 60분 이상이면 전체 스탑워치를 초기화한다.
    MD.stopwatch_indicator = 0;

    ST.stopwatchTime.YY = 0;
    ST.stopwatchTime.MT = 0;
    ST.stopwatchTime.DD = 0;
    ST.stopwatchTime.HH = 0;
    ST.stopwatchTime.MM = 0;
    ST.stopwatchTime.SS = 0;
    ST.stopwatchTime.MS = 0;
```

REALTIME MANAGER-BACKLIGHT

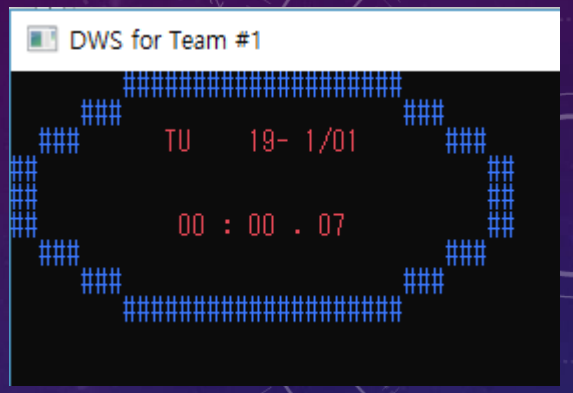
- Realtime Manager에서는 현재 시간과 BacklightTime을 비교합니다.
  - 만약 시/분/초 가 같다면 BC.value = 기본값으로 변경시켜 줍니다.

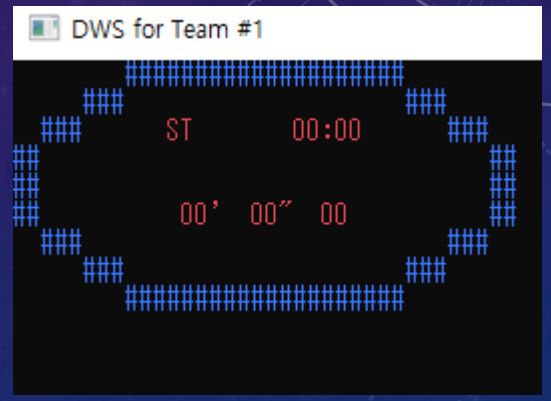
```
//3. Backlight 조작하기
if(CT.HH==BC.BacklightTime.HH && CT.MM ==BC.BacklightTime.MM && CT.SS==BC.BacklightTime.SS){
    BC.value = COLOR_DEF;
}
```

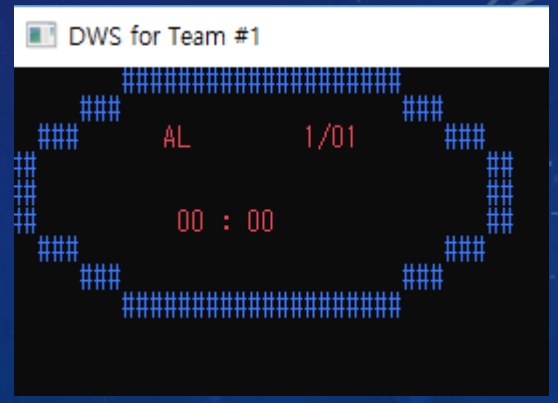
PANEL AND SPEAKER CONTROLLER

- 모드를 읽고 Panel 과 Speaker를 관리합니다.
- Panel
  - 화면에 표시될 정보들을 configure 합니다.
  - 이때 표시될 정보는 현재 모드 대분류, 위치 별 내용, 깜빡일 위치 입니다.
  - show()를 호출하고 표시할 정보들을 전달합니다.
- Speaker
  - 알람이 울릴 때, 스피커는 작동합니다.

```
if (MD.alarm_buzzing) {// 알람이 울리는 중
    if(AL.ring == CT.SS){// 현재 시각 초와 ring이 같다
        Alarm();//알람 울리기
        Ring_Changer();//ring++
    }
}
```







SHOW

- Panel & Speaker 에서 Configure 된 정보를 화면에 출력합니다.
- 깜빡여야 할 위치를 전달받은 대로 깜빡입니다.
- gotoxy()를 활용하여 시계의 적합한 위치로 커서 이동 후 정보를 출력합니다.

ALARM

- Panel & Speaker Controller 에서 호출되었을때, printf(“\a”); 를 실행합니다.

BACKLIGHT CONTROLLER

- 현재 Backlight를 보고 콘솔에 표현될 글자 색깔을 변경시켜줍니다.

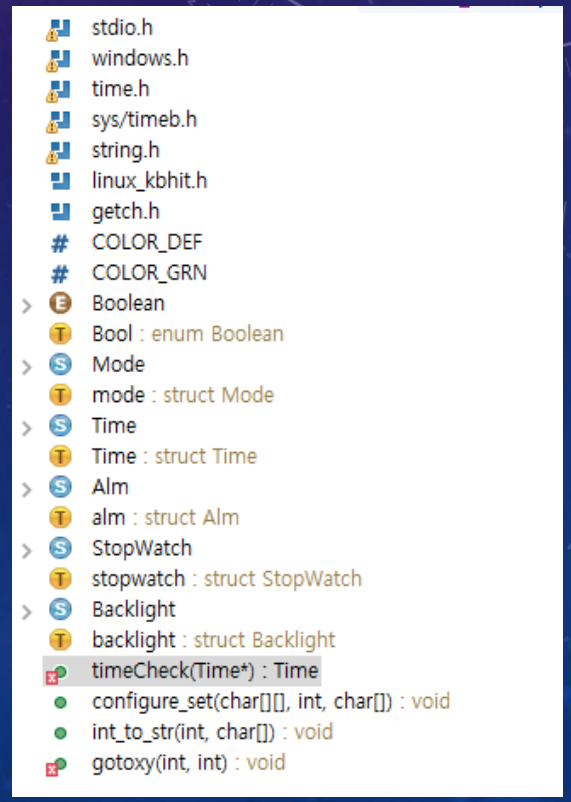
```
void Backlight_Controller(){ //색 변경
    //요청된 backlight 색깔대로 변경하는 형태
    if(BC.value==COLOR_DEF || BC.value==COLOR_GRN){
        SetConsoleTextAttribute( GetStdHandle( STD_OUTPUT_HANDLE ), BC.value);
    }
}
```

- 사용하는 글자 색들은 다음과 같습니다.

```
#define COLOR_DEF 12 //연한 빨강
#define COLOR_GRN 14 //연한 노랑
```

HEADER FILE DESCRIPTION

- “ linux_kbhit.h“
  - VS C++ 컴파일러의 kbhit() 기능을 모사.
- “getch.h “
  - VS C++ 컴파일러의 getch() 기능을 구현
- “util.h”
  - Data Store Struct들의 정의 및 구현 상의 편의를 위해 구현한 함수들
  - Time timeCheck(time* dest);
    - 시간의 연산 후 음수나 초과값들을 보정해주는 역할을 하는 함수
  - void int_to_str(int to, char temp[3]);
    - 두 자리 int를 string으로 바꾸는 기능
  - void configure_set(char list[7][3], int location, char goal[3]);
    - 8개의 패널의 각각의 위치에 string을 넣는 과정
  - void gotoxy(int x, int y);
    - 커서를 옮기는 과정. Show할 때 사용된다



Q & A

팀프로젝트 끝!! 다들 수고 많으셨습니다!

팀프로젝트 끝!!!!

@@@수고하셨습니다@@@@