임베디드 소프트웨어 최종 과제 보고서

컴퓨터공학과

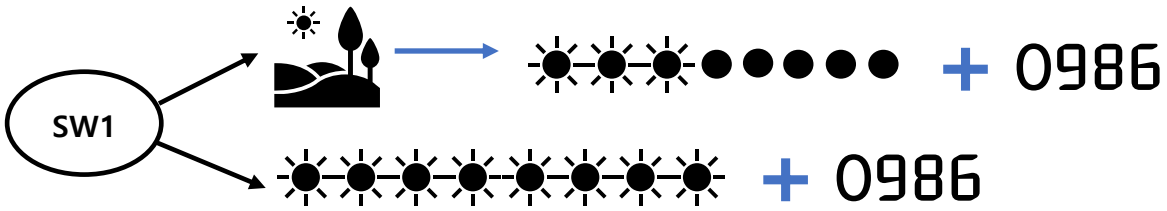
12181597 김현지

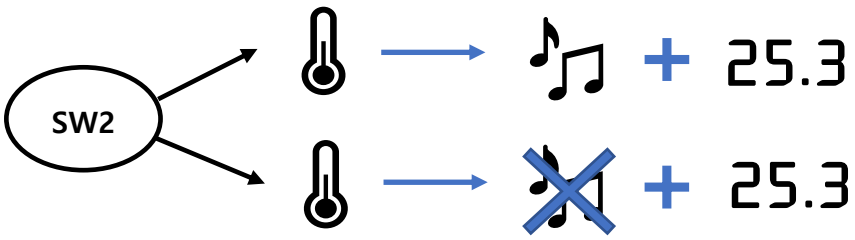
1. 동작 구상

이전에 설계했던 기능을 좀 더 보완하여, 각 모드에서 FND를 항상 출력할 수 있게 변경하였다. 사용자는 두 개의 스위치를 반복적으로 누르면서 모드나 모드 내의 동작을 변경할 수 있다. 처음 실행 시 기본 동작은 광센서 모드(Mode 0)의 '밝기에 따른 LED 수 증가' 동작이다.

① SW1 : 광센서 모드(Mode 0). 광센서를 이용해 얻은 밝기값이 작을수록 켜지는 LED 수가 증가한다. 스위치를 한 번 더 누르면 모든 LED가 켜진다(다시 한 번 더 누르면 LED 수 변경 동작). 광센서 모드일 때는 항상 FND에 밝기 값을 출력한다.

② SW2 : 온도센서 모드(Mode 1). 온도센서를 이용해 얻은 온도값이 기준 미만일 경우 낮은 음, 기준 이상일 경우 높은 음을 BUZZER에 출력한다. 스위치를 반복해서 눌러 BUZZER를 켜고 끌 수 있다. 온도센서 모드일 때는 항상 FND에 온도 값을 출력한다.





2. 구현 방법

1) 함수

① ISR(INT4_vect)

: SW1을 누르면 호출된다. 이전 모드가 Mode 1일 경우 Mode 0으로 변경하고 모드가 바뀌기 전의 동작을 지속한다. 스위치를 누르기 전에도 Mode 0이었을 경우 동작을 변경한다(처음으로 모드에 진입했을 때의 동작은 '밝기에 따른 LED 출력 증가'이다).

② ISR(INT5_vect)

: SW2를 누르면 호출된다. 이전 모드가 Mode 0일 경우 Mode 1로 변경하고 모드가 바뀌기 전의 동작을 지속한다. 스위치를 누르기 전에도 Mode 1이었을 경우 동작을 변경한다(처음으로 모드에 진입했을 때의 동작은 'BUZZER에 음계 출력'이다).

③ read_adc

: 광센서로부터 밝기값을 읽는다.

④ ReadTemperature

: 온도 센서로부터 온도값을 읽는다.

2) 태스크

각 Task의 우선 순위는 아래와 같이 부여된다.

```
OSTaskCreate(SelectTask, (void *)0, (void *)&TaskStk[0][TASK_STK_SIZE - 1], 0);

OSTaskCreate(BrightTask, (void *)0, (void *)&TaskStk[1][TASK_STK_SIZE - 1], 1);
OSTaskCreate(TempTask, (void *)0, (void *)&TaskStk[2][TASK_STK_SIZE - 1], 2);

OSTaskCreate(FndLedTask, (void *)0, (void *)&TaskStk[3][TASK_STK_SIZE - 1], 3);
OSTaskCreate(FndBuzzerTask, (void *)0, (void *)&TaskStk[4][TASK_STK_SIZE - 1], 4);

OSTaskCreate(BrightDisplayTask, (void *)0, (void *)&TaskStk[5][TASK_STK_SIZE - 1], 5);
OSTaskCreate(TempDisplayTask, (void *)0, (void *)&TaskStk[6][TASK_STK_SIZE - 1], 6);
```

① SelectTask

: 현재 Mode 값이 0일 경우 BrightTask, 1일 경우 TempTask를 선택해 신호를 보낸다. 이후 1초 쉬는 동안 Bright/TempTask가 실행된다.

② BrightTask

: read_adc() 함수로 현재 밝기 값을 읽어 FndLedTask에 전달하고 1초 동안 쉰다. Mode가 0으로 유지되는 한 이 과정을 반복한다. BrightTask가 쉬는 동안 FndLedTask가 실행된다.

③ FndLedTask

: BrightTask로부터 받은 밝기 값을 전역 변수에 저장한다. 그 뒤 대기 상태에 들어가면 BrightDisplayTask가 실행된다.

④ BrightDisplayTask

: 전역 변수에 저장된 밝기 값을 FND에 출력한다. 현재 해야 할 동작(전역변수에 저장됨)에 따라 밝기 값 구간에 따라 수를 달리하여 LED를 켜거나, 모든 LED를 켠다.

⑤ TempTask

: ReadTemperature() 함수로 현재 온도 값을 읽어 FndBuzzerTask에 전달하고 1초 동안 쉰다. Mode가 1로 유지되는 한 이 과정을 반복한다. TempTask가 쉬는 동안 FndBuzzerTask가 실행된다.

⑥ FndBuzzerTask

: TempTask로부터 받은 온도 값을 전역 변수에 저장한다. 그 뒤 대기 상태에 들어가면 TempDisplayTask가 실행된다.

⑦ TempDisplayTask

: 전역 변수에 저장된 온도 값을 FND에 출력한다. 현재 해야 할 동작(전역변수에 저장됨)에 따라 음의 높낮이를 달리하여 BUZZER에 출력하거나, 아예 출력하지 않는다.

3) 통신

① Event flag

: SW1, SW2를 누를 때마다 ISR(INT4_vect), ISR(INT5_vect) 함수 내에서 전역 변수 Mode값을 0, 1로 변경한다. SelectTask는 반복적으로 Mode값을 확인해 0이면 BrightTask, 1이면 TempTask를 선택해 FlagPost로 신호를 준다. Bright에는 0x01, Temp에는 0x02를 Post하고 각 Task들은 그와 일치하는 비트가 될 때까지 FlagPend한다. Mode가 변하지 않는 한 두 Task들은 밝기/온도값을 읽어 넘겨주는 작업을 반복해야 하므로 consume 옵션은 주지 않는다. Mode가 변경될 때 작업 중이던 Bright/TempTask를 멈춰야 하는데, 그 일은 ISR(INT4_vect), ISR(INT5_vect)가 한다. 스위치를 눌러 ISR 함수로 진입했을 때 FlagPost에 OS_FLAG_CLR 옵션으로 상대 태스크가 Pend하던 플래그를 지워버리면 된다.

② Mailbox

: OSFlagPend를 통해 BrightTask 혹은 TempTask에 진입한 뒤 밝기/온도 값을 읽어 Mailbox에 담아 FndLedTask / FndBuzzerTask에 보낸다. FndLedTask와는 Mbox[0], FndBuzzerTask와는 Mbox[1]을 사용한다. 보낸 뒤 1초 쉬는 동안 FndLed/FndBuzzerTask

는 MboxPend를 통해 값을 받아 전역 변수에 저장한다.

③ Message queue

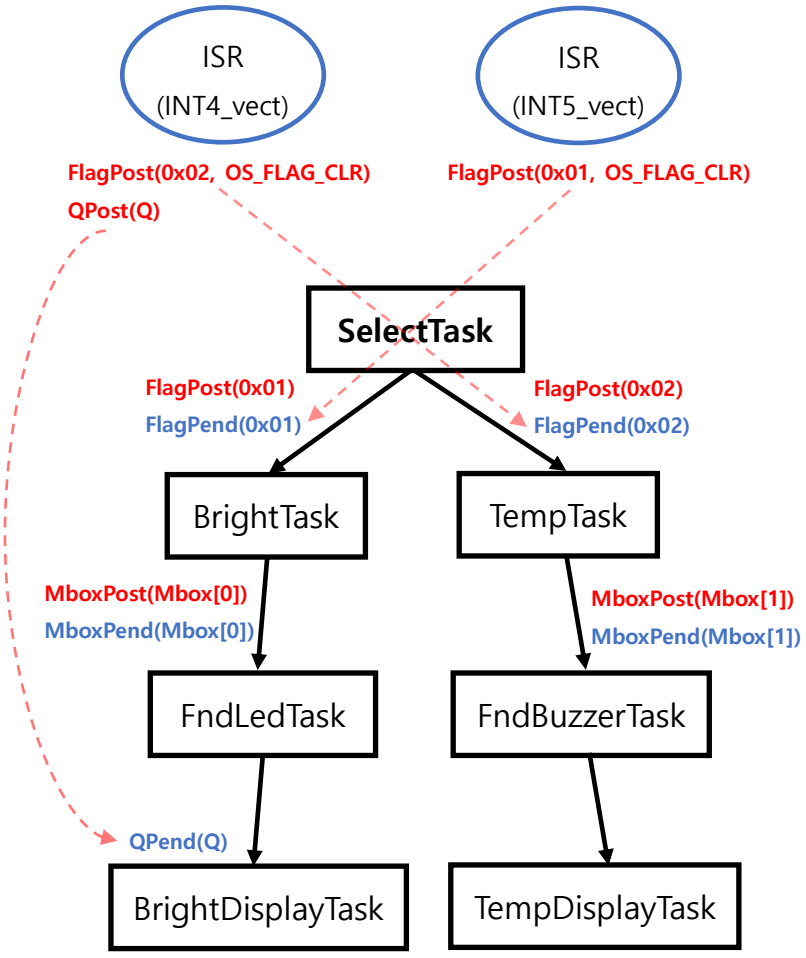
: BrightDisplayTask의 우선 순위가 TempDisplayTask의 우선 순위보다 높은데, BrightDisplayTask 내에 대기하는 코드가 없다면 Mode가 1이어서 FndBuzzerTask가 실행돼도 그 다음 순위가 BrightDisplayTask이므로 TempDiplayTask는 작업할 수 없게 된다.

```
OSTaskCreate(BrightDisplayTask, (void *)0, (void *)&TaskStk[5][TASK_STK_SIZE - 1], 5);
OSTaskCreate(TempDisplayTask, (void *)0, (void *)&TaskStk[6][TASK_STK_SIZE - 1], 6);
```

따라서 BrightDisplayTask 내에서 현재 모드가 1이라면 QPend를 통해 대기하도록 만든다. 대기 중이다가 SW1이 눌리면 ISR(INT4_vect) 함수 내에서 QPost 하여 BrightDisplayTask가 대기상태에서 풀릴 수 있게 한다.

TempDiplayTask는 우선 순위가 가장 낮기 때문에 Pend를 통해 대기하지 않아도 된다.

④ 모식도



3. 구현

① ISR(INT4_vect)

```
ISR(INT4_vect) { //SW1
    INT8U err;
    if (Mode == 0) { // 같은 스위치를 두 번 눌렀을 때
        if (led_all == 0)
            led_all = 1;
        else
            led_all = 0;
    }
    else            //모드 전환
        Mode = 0;
    OSFlagPost(grp_mode, 0x02, OS_FLAG_CLR, &err);
    OSQPost(queue_bright, (void *)Mode);
    _delay_ms(10);
}
```

- 전역 변수 Mode 값이 0(광센서 모드)인데 ISR(INT4_vect)에 진입한 경우는 같은 스위치를 두 번 눌렀을 때이므로 led_all 전역 변수 값을 바꾼다. led_all이 1이면 모든 LED를 켜고, 0이면 빛에 따라 켜지는 LED 수가 달라진다.

- Mode가 1이었을 경우 0으로 바꾼다.

- OSFlagPost(grp_mode, 0x02, OS_FLAG_CLR, &err)로 0x02 비트를 0으로 클리어해 TempTask가 작동되지 못하도록 한다.

- OSQPost(queue_bright, (void *)Mode)를 통해 BrightDisplayTask가 동작할 수 있게 한다.

② ISR(INT5_vect)

```
ISR(INT5_vect) { //SW2
    INT8U err;
    if (Mode == 1) { // 같은 스위치를 두 번 눌렀을 때
        if (buzzer_off == 0)
            buzzer_off = 1;
        else
            buzzer_off = 0;
    }
    else            // 모드 전환
        Mode = 1;
    OSFlagPost(grp_mode, 0x01, OS_FLAG_CLR, &err);
    _delay_ms(10);
}
```

- Mode 1(온도센서 모드)인데 ISR(INT5_vect)에 진입한 경우는 같은 스위치를 두 번 눌렀을 때

이므로 buzzer_off 전역 변수 값을 바꾼다. buzzer_off가 1 버저를 울리지 않고, 0이면 온도에 따라 버저에 출력되는 음계가 달라진다.

- OSSFlagPost(grp_mode, 0x01, OS_FLAG_CLR, &err)로 0x01 비트를 0으로 클리어해 BrightTask가 작동되지 못하도록 한다.

③ SelectTask

```
void SelectTask(void *data) {
    INT8U err;
    data = data;
    while (1) {
        if (Mode == 0) //광센서 모드
            OSFlagPost(grp_mode, 0x01, OS_FLAG_SET, &err);
        else //온도센서 모드
            OSFlagPost(grp_mode, 0x02, OS_FLAG_SET, &err);

        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}
```

- OSFlagPost 함수를 이용해 Mode가 0일 경우 0x01에 해당하는 비트를 1로, 1일 경우 0x02를 1로 세팅한다.

- 이후 1초 쉬면 OSFlagPend로 대기하던 Bright/TempTask가 동작할 수 있게 된다.

④ BrightTask

```
void BrightTask(void *data) {
    USHORT value;
    INT8U err;
    data = data;

    init_adc();
    while (1) {
        OSFlagPend(grp_mode, 0x01, OS_FLAG_WAIT_SET_ALL, 0, &err);

        OS_ENTER_CRITICAL();
        value = read_adc();
        OS_EXIT_CRITICAL();
        OSMboxPost(Mbox[0], (void*)&value);

        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}
```

- OSFlagPend(grp_mode, 0x01, ..)를 통해 첫 번째 비트가 1로 세팅될 때까지 기다린다.

- read_adc()함수로 현재 밝기 값을 읽은 뒤 OSMboxPost(Mbox[0], (void*)&value)로 FndLedTask에 전달한다.

- 이후 1초 쉬는 동안 FndLedTask가 동작할 수 있게 된다.

- 1초가 지나면 OSFlagPend 라인으로 돌아간다. Mode가 0인 한 대기하지 않으므로 값 읽기-보내기가 반복된다.

⑤ FndLedTask

```
void FndLedTask(void *data) {
    INT8U   err;
    data = data;

    while (1) {
        led_value = *(USHORT *)OSMboxPend(Mbox[0], 0, &err);
    }
}
```

- OSMboxPend(Mbox[0], ..)으로 받은 밝기 값을 전역 변수에 저장한다.

- while(1)이므로 MboxPend를 반복하게 되는데, BrightTask가 다시 MboxPost를 하기 전까진 대기 상태에 들어간다. 그 동안에 BrightDisplayTask가 실행된다.

⑥ BrightDisplayTask

```
void BrightDisplayTask(void *data) {
    INT8U i, err;
    USHORT value;

    while (1) {
        if (Mode == 1) { // 현재 온도센서 모드임
            PORTA = 0x00;
            OSQPend(queue_bright, 0, &err); // 값을 받지는 않고, 대기 용도로 사용
        }
```

- 현재 모드가 1인데 BrightDisplayTask가 동작 중이라는 것은 TempDisplayTask가 이 Task의 우선 순위에 밀렸기 때문이다. 따라서 OSQPend(queue_bright, ..)을 통해 이 Task를 대기 상태로 만들어 준다. 이후 SW1을 눌러 모드를 변경했다면 ISR(INT4_vect)에서 QPost를 해 주기 때문에 대기 상태에서 풀리고, 이후의 코드를 실행할 수 있게 된다.

```
//FND에 밝기값 출력
USHORT fnd[4];
fnd[0] = digit[value / 1000];
fnd[1] = digit[(value / 100) % 10];
fnd[2] = digit[(value / 10) % 10];
fnd[3] = digit[value % 10];

for (i = 0; i < 4; i++) {
    PORTC = fnd[i];
    PORTG = fnd_sel[i];
    _delay_ms(2);
}
```

- 전역 변수에 저장된 밝기 값을 FND에 출력한다.

```
if (led_all) // 모든 LED를 켬
    PORTA = 0xff;
else {      // 밝기값에 따라 켜지는 LED 수를 달리함
    if (value < CDS_VALUE[0])      // **** ****
        PORTA = 0xff;
    else if (value < CDS_VALUE[1]) // **** ***-
        PORTA = 0xfe;
    else if (value < CDS_VALUE[2]) // **** **--
        PORTA = 0xfc;
    else if (value < CDS_VALUE[3]) // **** *---
        PORTA = 0xf8;
    else if (value < CDS_VALUE[4]) // **** ----
        PORTA = 0xf0;
    else if (value < CDS_VALUE[5]) // ***- ----
        PORTA = 0xe0;
    else if (value < CDS_VALUE[6]) // **-- ----
        PORTA = 0xc0;
    else if (value < CDS_VALUE[7]) // *--- ----
        PORTA = 0x80;
    else                           // ---- ----
        PORTA = 0x00;
}
```

- 전역 변수 led_all 값이 1이면 모든 LED를, 0이면 밝기 값 구간에 따라 켜지는 LED 수를 다르게 출력한다. (USHORT CDS_VALUE[8] = { 790, 820, 850, 880, 920, 960, 1000, 1050 };)

⑦ TempTask

```
void TempTask(void *data) {
    int value;
    INT8U   err;
    data = data;
    InitI2C();

    write_twi_1byte_nopreset(ATS75_CONFIG_REG, 0x00); // 9비트, Normal
    write_twi_0byte_nopreset(ATS75_TEMP_REG);
    while (1) {
        OSFlagPend(grp_mode, 0x02, OS_FLAG_WAIT_SET_ALL, 0, &err);
        OS_ENTER_CRITICAL();
        value = ReadTemperature();
        OS_EXIT_CRITICAL();
        OSMboxPost(Mbox[1], (void*)&value);
        OSTimeDlyHMSM(0, 0, 1, 0);
    }
}
```

- OSFlagPend(grp_mode, 0x02, ..)를 통해 두 번째 비트가 1로 세팅될 때까지 기다린다.

- ReadTemperature() 함수로 현재 온도 값을 읽어 OSMboxPost(Mbox[1], (void*)&value)로 FndBuzzerTask에 전달한다.

- 이후 1초 쉬는 동안 FndBuzzerTask가 동작할 수 있게 된다.

- 1초가 지나면 OSFlagPend 라인으로 돌아간다. Mode가 1인 한 대기하지 않으므로 값 읽기-보내기가 반복된다.

⑧ FndBuzzerTask

```
void FndBuzzerTask(void *data) {
    INT8U   err;
    data = data;

    while (1) {
        temp_value = *(USHORT *)OSMboxPend(Mbox[1], 0, &err);
    }
}
```

- OSMboxPend(Mbox[1], ..)으로 받은 온도 값을 전역 변수에 저장한다.

- while(1)이므로 MboxPend를 반복하게 되는데, TempTask가 다시 MboxPost를 하기 전까진 대기 상태에 들어간다. 그 동안에 TempDisplayTask가 실행된다.

⑨ TempDisplayTask

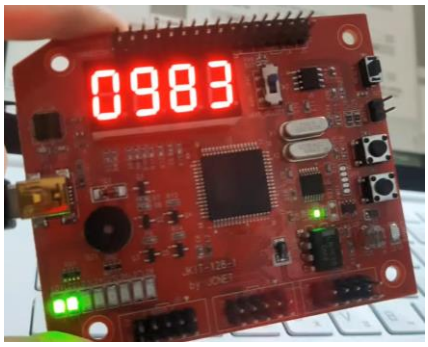
```
while (1) {
    value = temp_value;
    if ((value & 0x8000) != 0x8000)  // Sign 비트 체크
        num[3] = 11;
    else {
        num[3] = 10;
        value = (~value) - 1;   // 2's Compliment
    }
    value_int = (unsigned char)((value & 0x7f00) >> 8);
    value_deci = (unsigned char)(value & 0x00ff);

    num[2] = (value_int / 10) % 10;
    num[1] = value_int % 10;
    num[0] = ((value_deci & 0x80) == 0x80) * 5;

    if (buzzer_off == 0) { //0:버저 출력, 1:버저 끔
        // 24도 미만: 낮은 음, 24도 이상: 높은 음
        time_us = (value_int < 24) ? 200 : 700;
        for (i = 0; i < 5; i++) {
            PORTB = 0x10;
            _delay_us(time_us);
            PORTB = 0x00;
            _delay_us(time_us);
        }
    }
```

- 전역 변수 temp_value 값을 저장한 뒤 변환한다. 전역 변수 buzzer_off 값이 0이면 버저를 끄지 않는다. value_int 값이 24보다 작을 경우 딜레이 200us, 클 경우 딜레이 700us를 주어 온도에 따라 출력되는 음계가 달라지게 한다.

```
    //fnd에 온도값 출력
    for (i = 0; i < 4; i++) {
        PORTC = digit[num[i]];
        PORTG = fnd_sel[i];
        if (i == 1) PORTC |= 0x80;
        _delay_ms(2);
    }
```

- 온도 값을 FND에 출력한다.

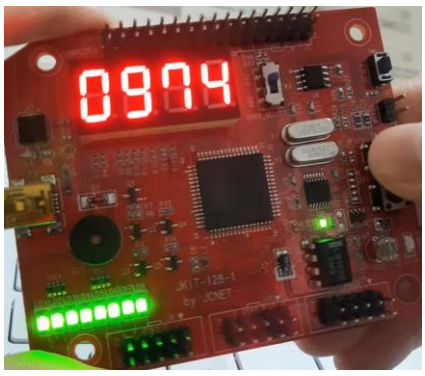
4. 실행 결과



1) download 후 모습.

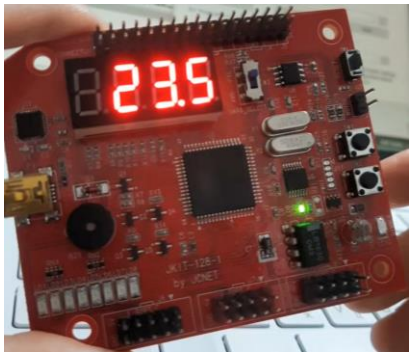
광감지센서를 통해 얻은 밝기 값 983을 FND에 출력하고 LED 두 개를 켬.

2) 광감지센서를 손가락으로 가리자 출력된 밝기 값이 828으로 감소. LED 여섯 개를 켬.



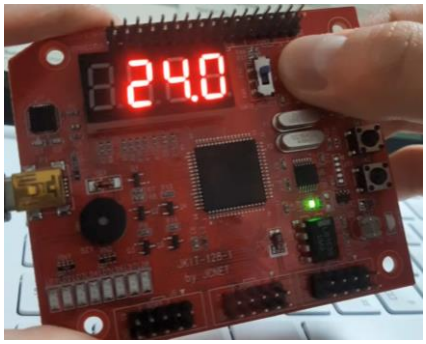
3) SW1을 한 번 누르면 광센서 모드를 유지한 채 모든 LED를 켬.

SW1을 다시 누르면 밝기값에 따라 출력 LED 수를 달리함.



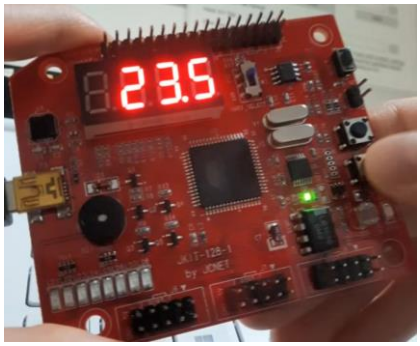
4) SW2를 누르면 온도센서 모드로 변경하고 FND에 온도 값을 출력.

24.0도 미만이므로 낮은 음이 버저에 출력.



5) 온도 센서를 손가락으로 감싸자 24.0의 온도값이 FND에 출력됨.

24.0도 이상이므로 보다 높은 음이 버저에 출력됨.



6) SW2를 한 번 더 누르면 버저에 음을 출력하지 않음.

SW2를 다시 누르면 온도값에 따라 해당 음을 버저에 출력.