프로젝트 주제: 밝아지면 알림이 울리고, 퀴즈를 3번 맞춰야 꺼지는 알람

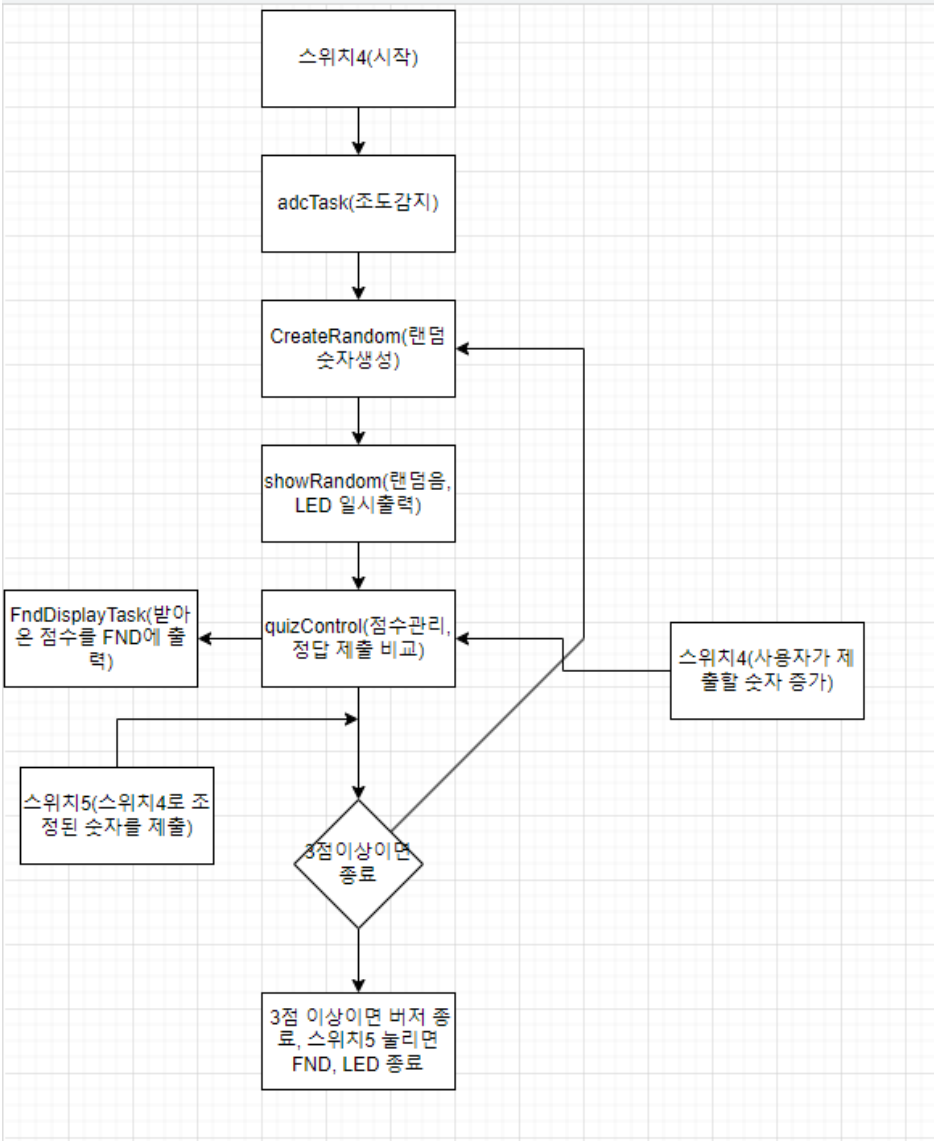
일정 조도 이상이면 울리는 알람을 구현하였다.

알람이 시작되면 0~7사이의 랜덤 숫자를 생성하고, 일정시간 동안 랜덤 숫자에 해당하는 버저음(도~도)를 울리고 LED를 점등한다. 이 후 사용자는 랜덤 숫자를 맞춰야한다.

사용자는 울렸던 버저음과 LED위치를 기억했다가 스위치(4)로 버저음, LED위치를 조절해서 이전에 울렸던 버저음, LED위치가 동일하다고 생각한다면 스위치(5)로 정답제출을 한다. FND에는 Score가 표시된다.

만약 정답이면 FND에 표시되던 Score가 1점 오른다. 오답이면 점수가 오르지 않는다. Score를 3점을 채우지 못했다면 다시 랜덤숫자를 뽑아 스위치를 조절해 랜덤숫자를 맞추어 Score를 올려야 한다.

Score가 3점이 되었다면 버저는 더 이상 울리지 않고 스위치(5)를 눌러 FND와 LED를 끌 수 있다.



총 5개의 Task와 3개의 input(스위치4,5, 조도), 3개의 Output(버저, FND, LED), 2개의 EventFlag, 4개의 MBox를 사용하였다.

스위치4

```
ISR(INT4_vect) {
    INT8U err;
    if (isStart == 0)// 1.조도센서를 작동시키며 시작을 알림.
    {
        isStart = 1;
        OSFlagPost(E_F1, 0x01, OS_FLAG_SET, &err);
    }
    else if (quizStart) // 2. 퀴즈 중에 스위치를 누르면 표기하는 음과 LED가 한단계 증가함.
    {
        num = (num + 1) % 8;
        index = num;
        PORTA = number[num];
    }
}
```

두 개의 역할을 가졌다. 1. 조도센서를 작동시킨다.(이벤트 플래그를 통해 깨운다) 2. 퀴즈가 시작됐을 때 음과 LED를 한 단계씩 증가시킨다. num은 전역변수이다.

adcTask

```
void init_adc(void) //조도 센서 시작
{
    ADMUX = 0x00;
    ADCSRA = 0x87;
}

unsigned short read_adc(void) //조도센서값을 읽는 함수
{
    unsigned char adc_low, adc_high;
    unsigned short value;
    ADCSRA |= 0x40;
    while ((ADCSRA & 0x10) != 0x10);
    adc_low = ADCL;
    adc_high = ADCH;
    value = (adc_high << 8) | adc_low;

    return value;
}

void adcTask(void* data)
{
    INT8U err;
    data = data;
    USHORT value = 0;
    init_adc();
    OSFlagPend(E_F1, 0x01, OS_FLAG_WAIT_SET_ALL + OS_FLAG_CONSUME, 0, &err);//스위치4가 깨워준다.
    PORTA = 0xff;
    while (1)
    {
        value = read_adc();
        if (value > CDS_VALUE) { //10LUX보다 밝아지면 알람시작!
            OSFlagPost(E_F2, 0x01, OS_FLAG_SET, &err);
            break;
        }
    }
    OSTimeDlyHMSM(0, 0, 1, 0);
}
```

조도를 측정하는 Task와 함수이다. 스위치 4번이 눌리면 그 때부터 조도 측정이 시작된다. 10LUX 이상의 조도가 측정되면 **createRadom**를 깨운다. (이벤트플래그를 이용해 깨운다)

createRandom

```
void createRandom(void* data) { //랜덤숫자를 생성해준다.
    data = data;
    INT8U err;
    USHORT r;
    while (1) {
        OSFlagPend(E_F2, 0x01, OS_FLAG_WAIT_SET_ALL + OS_FLAG_CONSUME, 0, &err);
        r = (13 + rand()) % 8;
        OSMboxPost(Mbox4, (void*)&r);
    }
}
```

createRandom은 랜덤숫자를 만들어서 **showRandom**을 깨우고, 다시 Pend상태에 진입한다.(**quizControl**가 다시 깨워줄 것이다)

showRandom

```
void showRandom(void* data) { //랜덤으로 생성한 숫자로 버저와 LED를 켠다.
    data = data;
    INT8U err;
    USHORT Rvalue;
    while (1) {
        Rvalue = *(USHORT*)OSMboxPend(Mbox4, 0, &err);
        TIMSK = 0x40; //overFlow
        DDRB = 0x10; //버저활성화
        index = Rvalue;
        PORTA = number[Rvalue];

        _delay_ms(3000); //일정시간 랜덤음과, LED를 보여주고 끈다.

        PORTA = 0x00;
        TIMSK = 0x00; //overFlow비활성화
        PORTB = 0x00;
        _delay_ms(3000);

        OSMboxPost(Mbox2, (void*)&Rvalue);
    }
}
```

createRandom이 생성한 Random숫자를 받으면서 깨어난다. 받은 숫자에 해당하는 버저음과 LED를 일정시간동안 켜고, 다시 끈다. 이 후 사용자는 이 숫자를 맞추어야한다. Random숫자를 **quizControl**에 MBox를 통해 전달한다.

quizControl

```
void quizControl(void* data) { //퀴즈가 시작된다.
    data = data;
    INT8U err;
    USHORT correctValue;
    USHORT score = 0; //점수가 3점이 되면 알람이 꺼진다!
    USHORT tmp;
    while (1) {
        correctValue = *(USHORT*)OSMboxPend(Mbox2, 0, &err);
        num = 0; // 0번 음, LED에서 시작한다.
        quizStart = 1;
        TIMSK = 0x40; //overFlow
        PORTA = number[num];
        OSMboxPost(Mbox3, (void*)&score); //FND에 현재 점수를 표기한다.
        *(USHORT*)OSMboxPend(Mbox1, 0, &err);//스위치 5로 정답이 제출된다.
        quizStart = 0;
        if (correctValue == num) { //정답(랜덤생성값)과 현재 num을 비교한다.
            score++; //같다면 점수를 올려준다.
            PORTA = 0xff;//맞췄을 때는 LED전체를 점등한다.
        }
        else {//틀렸을 때는 LED를 띄엄띄엄 점등한다.
            PORTA = 0xaa;
        }
        TIMSK = 0x00; //overFlow비활성화
        PORTB = 0x00;
        PORTC = 0x00;

        _delay_ms(2000);
        if (score < 3) //3점이 되지 않았다면 다시 랜덤 숫자를 뽑아 퀴즈를 진행한다.
            OSFlagPost(E_F2, 0x01, OS_FLAG_SET, &err);
        else {
        }
    }

}
```

quizControl은 Random 숫자를 **showRandom**으로부터 넘겨받으며 깨어난다. num이라는 전역변수를 이용해 LED와 버저를 조절한다. num변수는 스위치 4가 눌릴 때마다 1씩 증가한다. **FndDisplayTask**를 활성화해서 현재 Score를 FND에 표시하도록한다.

스위치5가 눌릴 때까지 Pend한다. 스위치5가 눌리면 사용자가 현재 num값을 정답으로 제출한 것이다. num값을 전달받은 Random숫자와 비교하여 사용자가 맞췄다면 점수를 올리고 LED 전체를 점등(0xff)하여 피드백한다.

틀렸다면 점수는 변동이 없고, LED를 띄엄띄엄 점등(0xaa)하여 피드백한다.

만약 점수가 3점이 넘지 않았다면 **createRandom**을 다시 깨워 퀴즈를 계속 진행한다. 다음 퀴즈 진행 전 까지는 FND, LED, 버저를 전부 끄고 잠시 대기한다.

스위치 5

```
ISR(INT5_vect) {
    INT8U err;
    USHORT i = 0;
    if (quizStart == 1) {//1. 퀴즈 중에 스위치를 누르면 정답이 제출됨.
        OSMboxPost(Mbox1, (void*)&i);
    }
    else {//2. 퀴즈로 score를 3점 달성한 후 스위치를 눌러 FND와 LED를 끈다.
        isFinish = 1;
        PORTA = 0x00;
    }
}
```

두 개의 역할을 가졌다. 1. 퀴즈 중에 눌리면 정답을 제출하는 것으로 인지하여 **quizControl**에 알린다. 2. Score를 3점을 달성하여 버저가 종료되었을 때 스위치5를 눌러 LED와 FND 또한 꺼서 기기를 완전히 종료시킬 수 있다.

FndDisplayTask

```
void FndDisplayTask(void* data)
{
    data = data;
    INT8U err;
    UCHAR FND_DATA[] = { 0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7c, 0x07, 0x7f, 0x67, 0x40, 0x00 };
    UCHAR fnd_sel[4] = { 0x01, 0x02, 0x04, 0x08 };
    USHORT* display;
    int i;

    display = (USHORT*)OSMboxPend(Mbox3, 0, &err);
    DDRC = 0xff;
    DDRG = 0x0f;
    while (1) {
        if (isFinish) //끝났다면 FND를 끈다.
        {
            *display = 11;
        }
        for (i = 0; i < 4; i++)
        {
            PORTC = FND_DATA[*display];
            PORTG = fnd_sel[i];
            OSTimeDlyHMSM(0, 0, 0, 1);
        }
    }
}
```

FND에 **quizControl**로부터 받은 Score를 4칸 전부에 표시한다. 예를들어 1점이면 1 1 1 1을 표시하는 형식이다. 종료되었음을 스위치5가 알려주면 FND에 아무것도 표시하지 않도록한다.

TimerInterrupt

```
ISR(TIMER2_OVF_vect) { //타이머 인터럽트
    if (state == ON) {
        PORTB = 0x00;
        state = OFF;
    }
    else {
        PORTB = 0x10;
        state = ON;
    }
    TCNT2 = scale[index]; //음 조절
}
```

음 조절을 위한 TimerInterrupt이다.