임베디드 소프트웨어 [최종 프로젝트]

MP3 player

12171578 고정민, 12171706 조수민

1. Project Concept

|  |
| --- |
| <Track List>  1. ----------------------------- 비행기  2. ----------------------------- 산토끼  3. ----------------------------- 무엇이 무엇이 똑같을까  4. ----------------------------- 나비야 |

연결이 시작되면 버저에서는 첫번째 Track인 노래 ‘비행기’가 흘러나오기 시작한다. 노래가 흘러나오는 동안에 노래의 음계에 따른 피아노 건반이 LED에 출력된다. 또 현재 진행 중인 노래의 Track number가 FND에 출력된다. 첫 번째 노래가 끝나면 두 번째 트랙으로 넘어가 노래 ‘산토끼’가 흘러나온다. 역시 Track number ‘2’가 FND에 출력된다. 현재 진행 중인 노래가 끝나지 않았는데 다음 노래를 듣고 싶다면 두 번째 스위치(SW2)를 누르면 다음 트랙으로 넘어갈 수 있다.

노래를 멈추게 하는 방법에는 두 가지가 있다. 첫 번째는 조도 센서를 어두워지게 하는 방법이다. 조도 센서가 약 10 lux 정도의 값을 가지면 노래를 멈추도록 한다. 조도 센서가 밝아지면 멈추었던 노래가 다시 재생된다. 두 번째 방법은 첫 번째 스위치(SW1)를 이용하는 것이다. 첫 번째 스위치는 low level을 이용하여 눌려있는 동안만 동작하도록 구현하였다. 스위치가 눌려 있는 동안에는 노래를 멈추고 FND에 이니셜 - S. J – (수민, 정민의 약자)을 출력하도록 하였다.

사용한 장치 : FND, LED, 버저, CDS 센서, 스위치

2. Task

구현한 태스크 : LedTask , MelodyTask, FndTask, CdsTask 총 네 개

Task 진행 순서

1. Cds태스크에서 빛을 읽어온다.

2. Fnd태스크에서 CDS 값에 따른 동작을 한다.

3. Melody태스크에서 state에 따른 동작을 취한다.

4. LED태스크에서 현재 음계에 맞는 LED 값을 출력한다.

🡺 태스크 우선순위는 반대가 된다.

Task 우선 순위

0 LedTask

1 MelodyTask

2 FndTask

3 CdsTask

3. 각각의 Task

사용한 이벤트 : Semaphore, Mailbox, Message Queue, Event Flag

<Cds Task>

조도 값을 읽어오는 태스크이다.

OSSemPend(Sem, 0, &err);

* 공유자원에 안전하게 접근하기 위해 Semaphore 사용한다.

ADCSRA |= 0x40;

* A/D 컨버터 변환이 시작된다.

while ((ADCSRA & 0x10) != 0x10);

* ADC 변환 완료 검사를 한다.

adc\_low = ADCL;

adc\_high = ADCH;

adc\_value = (adc\_high << 8) | adc\_low;

* 변환된 값을 읽어온다.

OSQPost(Queue, &adc\_value);

* value 값을 Message Queue를 사용하여 FndTask로 전달한다.

OSSemPost(Sem);

* 공유자원에 대한 접근을 마쳤으니 Semaphore 해제한다.

<Fnd Task>

CDS 값에 따라서 동작을 취하는 태스크이다.

cds\_result = \*(unsigned short\*)(OSQPend(Queue, 0, &err));

* CdsTask로 부터 value값을 Message Queue를 통해 전송받는다.

OSSemPend(Sem, 0, &err);

* 공유자원에 안전하게 접근하기 위해 Semaphore 사용한다.

if (cds\_result < CDS\_VALUE)

{

state = OFF;

}

* cds\_result값이 CDS\_VALUE 값보다 작으면(더 어두우면) 노래를 정지

else

{

state = ON;

fnd[3] = digit[track\_num + 1] + dot;

PORTC = fnd[3]; PORTG = fnd\_sel[3];

}

* cds\_result값이 CDS\_VALUE 값보다 크면 FND[3]에 트랙 넘버 출력

OSFlagPost(mel\_grp, 0x01, OS\_FLAG\_SET, &err);

OSFlagPost(led\_grp, 0x01, OS\_FLAG\_SET, &err);

* 태스크의 마침을 MelodyTask와 LedTask에게 Event Flag를 사용하여 전달한다.

OSSemPost(Sem);

* 공유자원에 대한 접근을 마쳤으니 Semaphore 해제한다.

OSTimeDlyHMSM(0, 0, 0, 2);

* 2ms의 delay를 준다.

<MelodyTask>

melody에 맞춰 버저를 울려 소리내는 태스크이다.

OSFlagPend(mel\_grp, 0x01, OS\_FLAG\_WAIT\_SET\_ALL + OS\_FLAG\_CONSUME, 0, &err);

* FndTask로부터 신호가 올 때까지 대기한다.

OSSemPend(Sem, 0, &err);

* 공유자원에 안전하게 접근하기 위해 Semaphore 사용한다.

if (state == ON)

{

PORTB = 0x10; TIMSK = 0x41; \_delay\_ms(300);

mel\_idx = (mel\_idx + 1) % 152;

if (mel\_idx == 0 || mel\_idx == 32 || mel\_idx == 64 || mel\_idx == 88)

{ track\_num = (track\_num + 1) % songs; }

}

else

{ PORTB = 0x00; TIMSK = 0x00; }

* state가 ON이면 버저는 1ms동안 ON인 상태를 유지한다.
* 타이머/카운터2의 오버플로우 인터럽트를 enable하고 300ms 지연
* mel\_idx를 1씩 증가시키고 노래가 시작되는 구간마다 track number를 설정해준다.
* state가 OFF이면 버저를 울리지 않는다.

OSMboxPost(Mbox, (void\*)& song[mel\_idx]);

* song[mel\_idx]를 LedTask로 Mailbox를 이용하여 전송한다.

OSSemPost(Sem);

* 공유자원에 대한 접근을 마쳤으니 Semaphore 해제한다.

OSTimeDlyHMSM(0, 0, 0, 2);

* 2ms의 delay를 준다.

<Led Task>

현재 흘러나오는 멜로디의 음계에 맞춰 LED를 출력하는 태스크이다.

const uc melody[8] = { 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01 };

* 좌측에서부터 LED를 하나씩 켜는 값의 배열

OSFlagPend(led\_grp, 0x01, OS\_FLAG\_WAIT\_SET\_ALL + OS\_FLAG\_CONSUME, 0, &err);

* FndTask로부터 신호가 올 때까지 대기한다.

mel\_result = \*(uc\*)OSMboxPend(Mbox, 0, &err);

* MelodyTask로부터 값을 Mailbox를 통해 전송 받는다.

OSSemPend(Sem, 0, &err);

* 공유자원에 안전하게 접근하기 위해 Semaphore 사용한다.

if (mel\_result == DO) { PORTA = melody[0]; }

if (mel\_result == RE) { PORTA = melody[1]; }

if (mel\_result == MI) { PORTA = melody[2; }

if (mel\_result == FA) { PORTA = melody[3]; }

if (mel\_result == SOL) { PORTA = melody[4]; }

if (mel\_result == LA) { PORTA = melody[5]; }

if (mel\_result == TI || mel\_result == TIF) { PORTA = melody[6]; }

if (mel\_result == UDO) { PORTA = melody[7]; }

* 음계에 따라 LED를 출력한다 (예) 낮은 도 : 맨 왼쪽 LED ON

OSSemPost(Sem);

* 공유자원에 대한 접근을 마쳤으니 Semaphore 해제한다.

OSTimeDlyHMSM(0, 0, 0, 2);

* 2ms의 delay를 준다.

4.

void Int\_init(void)

{

SREG |= 0x80;

EICRB = 0x08;

EIMSK = 0x30;

}

* 제어 레지스터의 I비트를 enable
* INT4는 low level, INT5는 falling edge
* INT4, INT5 활성화

ISR(INT4\_vect)

{

fnd[3] = bar; fnd[2] = S; fnd[1] = J; fnd[0] = bar;

PORTC = fnd[3]; PORTG = fnd\_sel[3]; \_delay\_us(100);

PORTC = fnd[2]; PORTG = fnd\_sel[2]; \_delay\_us(100);

PORTC = fnd[1]; PORTG = fnd\_sel[1]; \_delay\_us(100);

PORTC = fnd[0]; PORTG = fnd\_sel[0]; \_delay\_us(100);

}

* SW1을 누르면 FND에 이니셜 - S. J – (수민, 정민의 약자)을 출력
* low level에서 인터럽트 발생하기 때문에 누르고 있을 때만 동작

ISR(INT5\_vect)

{

track\_num = (track\_num + 1) % songs;

if (mel\_idx >= 0 && mel\_idx <= 31) { mel\_idx = 32; }

else if (mel\_idx >= 32 && mel\_idx <= 63) { mel\_idx = 64; }

else if (mel\_idx >= 64 && mel\_idx <= 87) { mel\_idx = 88; }

else if (mel\_idx >= 88 && mel\_idx <= 151) { mel\_idx = 0; }

\_delay\_ms(2);

}

* SW2을 누르면 트랙 넘버가 하나 올라가면서 다음 트랙이 재생된다.

ISR(TIMER2\_OVF\_vect)

{

if (state == ON)

{

PORTB = 0x00; TIMSK = 0x40; state = OFF;

}

else

{

PORTB = 0x10; TIMSK = 0x41; state = ON;

}

TCNT2 = song[mel\_idx];

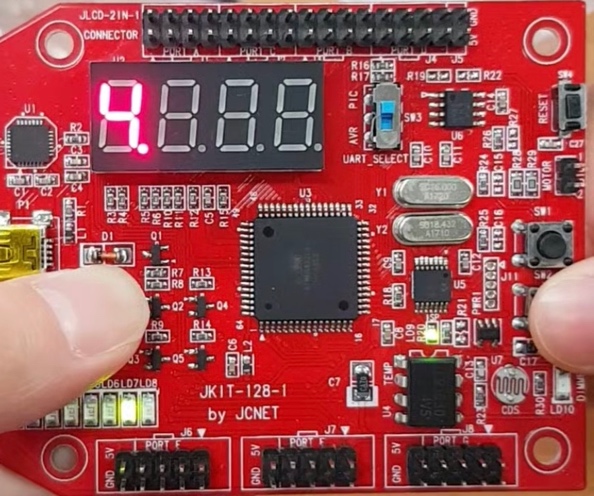
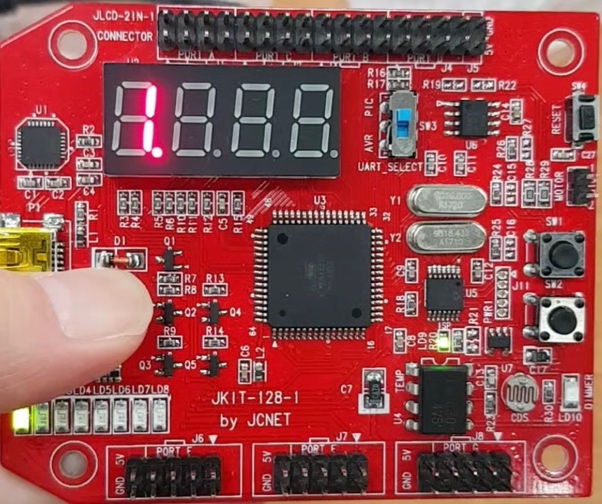
}

* 인터럽트가 들어오면 반대로 동작하도록 설정한다.

­­

5. 결과 사진 첨부

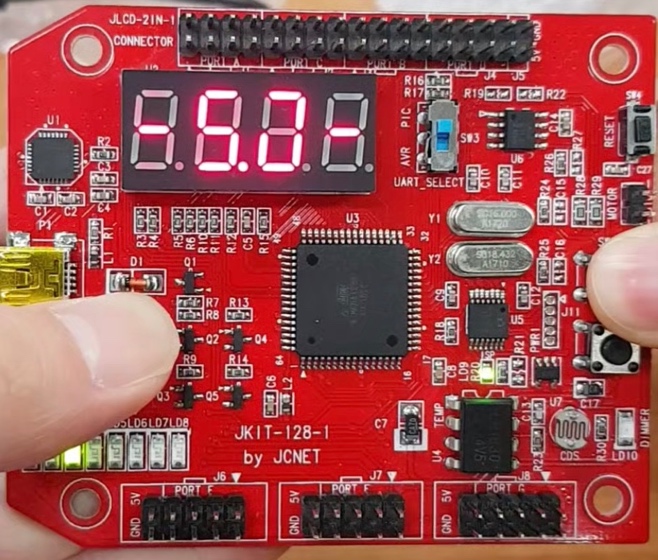
[평소 실행 화면]



왼쪽 : 1번 트랙 ‘비행기’ 연주 중, 도 음계를 소리 내는 중

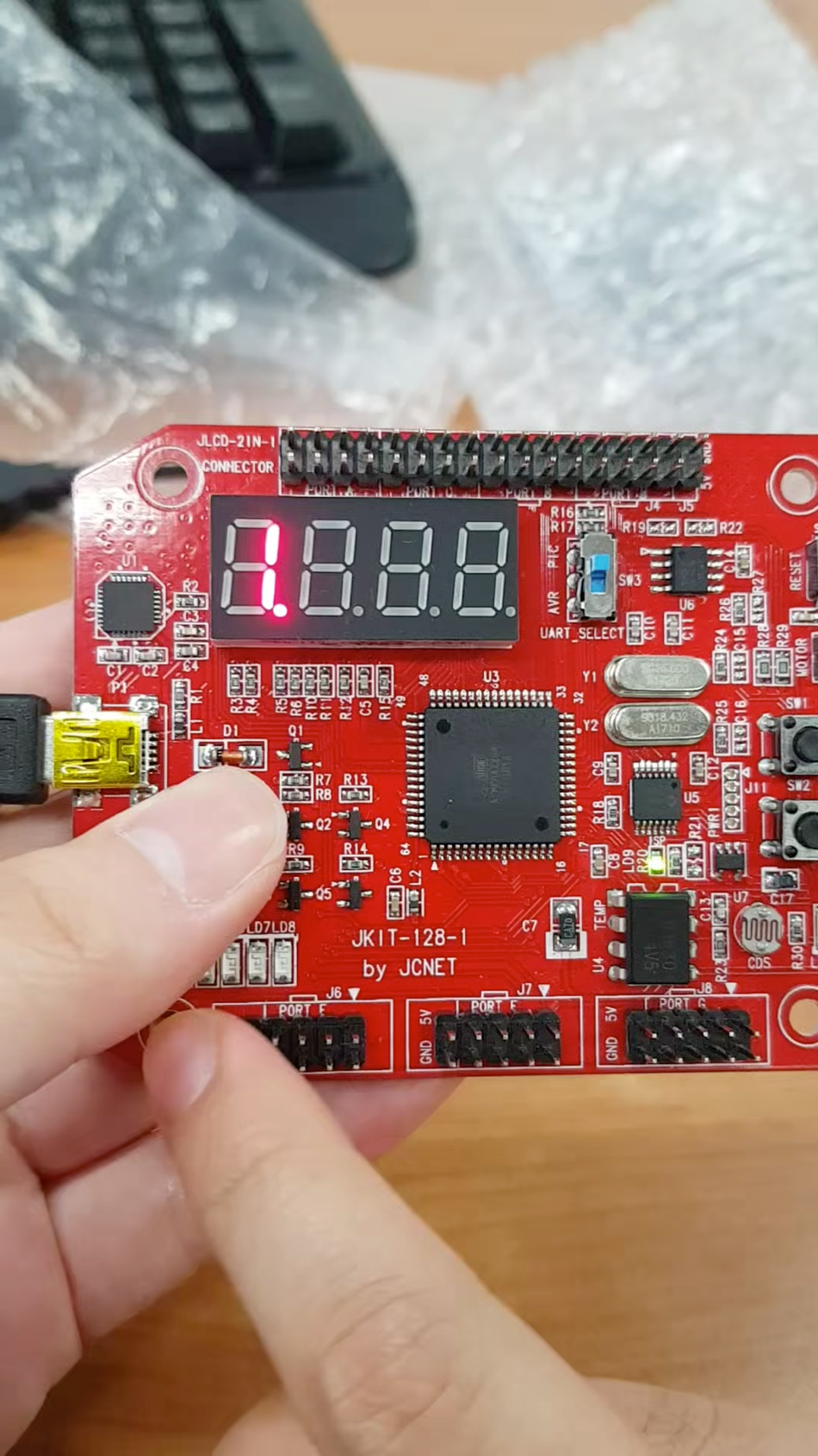
오른쪽 : 4번 트랙 ‘나비야’ 연주 중, 시 음계를 소리 내는 중

[SW1을 눌렀을 때]



이니셜 출력 – S. J –

동영상



5. 전체 코드 캡쳐

