임베디드 시스템 설계 및 실험

7주차

7조
201424470 서민영
201424421 김시은
201424533 정종진
201424532 정재광

목차

1. 실험목표

1) Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
2) 라이브러리 함수 사용법 숙지

2. 배경지식

1) Polling

```
while(1)
{
     if(Joystick check)
      LED_on;
}
```

<소스코드 1 : Polling 방식>

- CPU에서 특정 Event가 발생할 때까지 계속해서 확인하는 방식
- 구현은 쉬우나 시스템 성능 저하의 원인이 될 수 있다.

2) Interrupt

```
Stick_interrupt();
while(1)
{
     LED_on;
}
```

<소스코드 2 : Interrupt 방식>

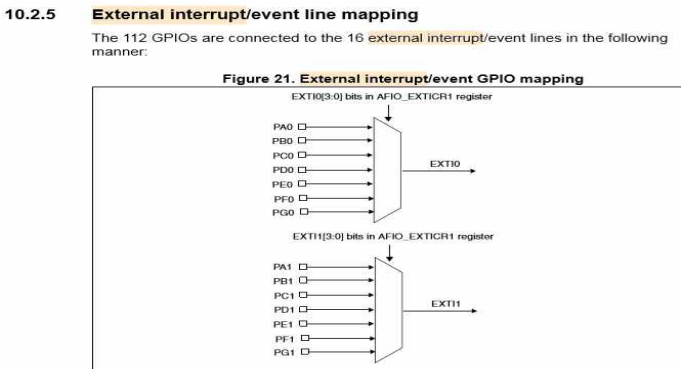
- CPU가 다른 연산을 처리하는 도중에도 특정 Event가 발생하면 Interrupt handler를 이용하여 처리하는 방식
- 구현은 복잡하지만 시스템 성능 저하를 줄일 수 있다.

① Hardware Interrupt
- CPU외부의 컨트롤러나 주변 장치로부터 요구됨
- 이벤트 처리를 요구하는 상황을 알리기 위해 전기적인 신호를 사용
- 입출력 Interrupt(종료, 오류), 기계검사 Interrupt(정전)

② Software Interrupt
- CPU내부에서 자신이 실행한 명령이나 명령을 실행하는 중 관련된 모듈이 변화하는 경우에 발생
- 프로그램 실행 중 프로그램의 오류나 예외 상황을 알림Trap(트랩) or Exception(예외 처리)
- 프로그램 내에서 감시를 목적으로 발생시킨 명령어에 의해 발생되기도 함(supervisor Call)

③ NVIC
- Nested Vectored Interrupt Controller
- 중첩된 인터럽트를 정해진 우선순위에 따라 제어하는 기능
- 인터럽트에 직접 주소 값을 넣어 분기시간을 상당 부분 감소시킴
- Cortex에서는 NVIC를 지원

④ EXTI



- EXTernal Interrupt
- 외부에서 신호가 입력될 경우 장치에 이벤트나 인터럽트가 발생되는 기능.
- 입력 받을 수 있는 신호는 Rising Edge, Falling Edge, Rising & Falling Edge

3) 납땜

- 부품을 기판에 연결하기 위해
- 가열된 인두기에 납을 용해시켜 접착시킴
- 부분과 기판을 가열하고 납을 대서 적당히 녹인 후 인두기와 납을 땐다
- 납땜 실수 시에 납 흡입기 또는 솔더윅으로 납을 제거

3. 실험과정

1) 보드 연결

<그림 1 : Coretex M3/JTAG/DSTREAM을 연결한 모습>

1-1) Coretex M3/JTAG/DSTREAM 연결

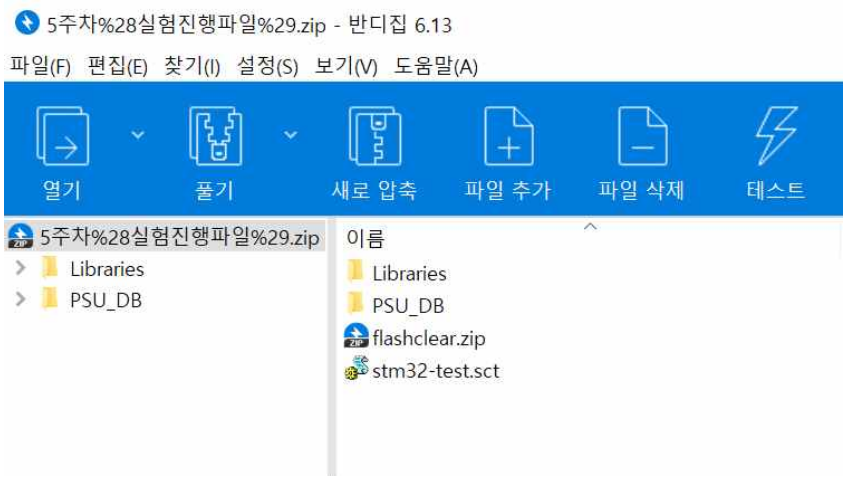
다음과 같은 순서로 보드를 연결한다. 이 때 연결 및 분리 순서를 제대로 지키지 않으면 장비가 망가질 수 있으니 주의해야한다.

① 보드와 DSTREAM JTAG 연결
② 보드 전원선만 연결
(보드의 전원은 OFF 상태)
③ DSTREAM 전원 연결 및 ON
④ DSTREAM Status LED 점등 확인
⑤ 보드 전원 ON
⑥ DSTREAM Target LED 점등 확인
⑦ DS-5에서 'connect target'

<표 1 : 보드 연결 순서>

2) DS-5 디바이스 데이터베이스 추가

2-1) 수업게시판에서 실습파일로 제공되는 PSU_DB, LIBRARIES, SCATTER FILE을 다운



<그림 7 : 실습파일로 제공된 파일들>

2-3) Eclipse에 데이터베이스 추가

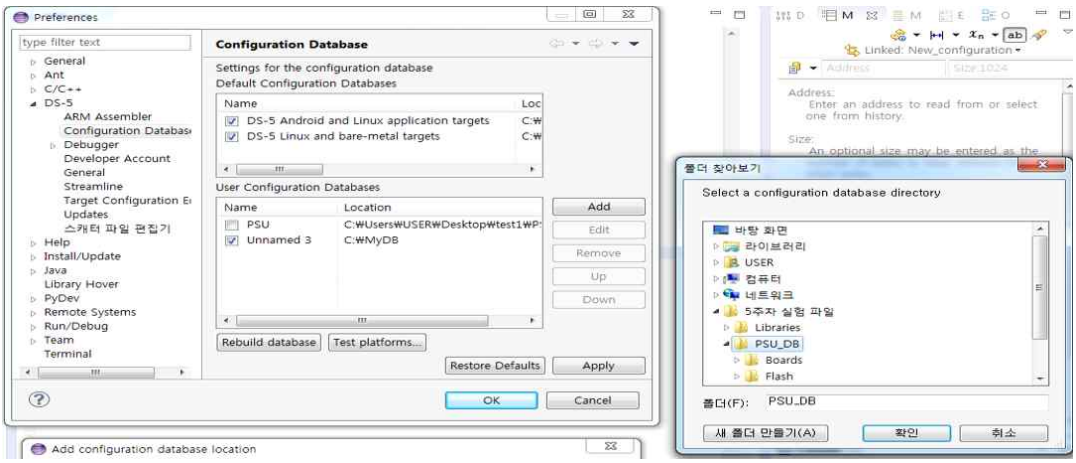
① 시작 → 모든 프로그램 → ARM DS-5 → Eclipse for DS-5 메뉴를 선택

② Windows → Preferences

③ DS-5 → Configuration Database 항목을 선택

④ Add 버튼을 클릭하여 사용자 데이터베이스의 디렉터리를 지정

⑤ Rebuild database 버튼을 클릭하여 데이터베이스 추가를 완료



<그림 8 : Eclipse에 데이터베이스 추가>

3) C Project 생성 및 환경설정

3-1) C Project 생성

① New Project → C project → Excutable → Empty Project를 선택해주고 Toolchains는 ARM Cmpiler 5(DS-5 built in)로 선택 후 프로젝트 생성

3-2) C Project Properties 설정

① C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → ARM Linker 5 → Image Layout → Scatter file 설정

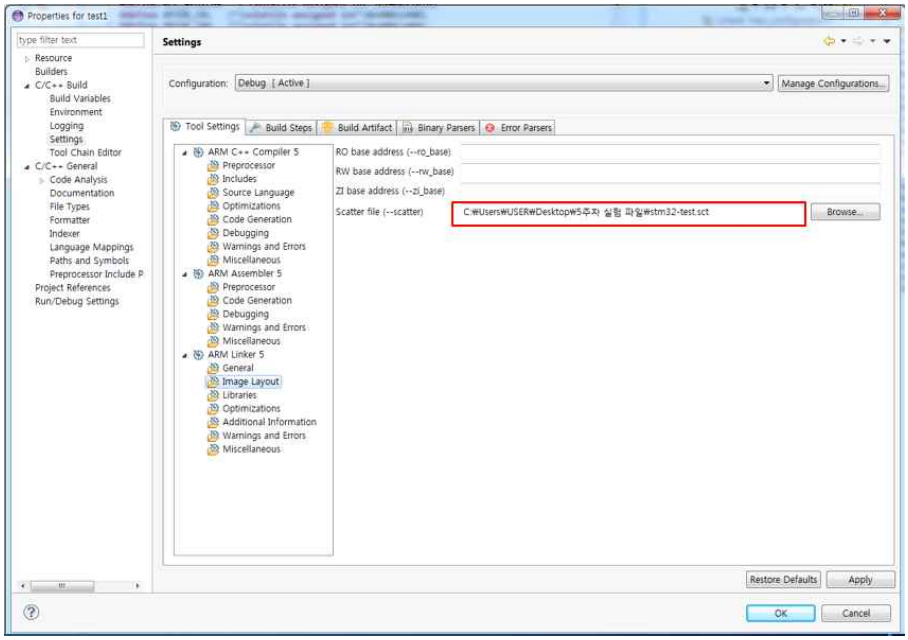
(Scatter file에서 RO/RW base address를 지정해주므로 설정해줄 필요가 없음)

② C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → Code Generation과 General의 Target CPU를 Cortex M3로 설정

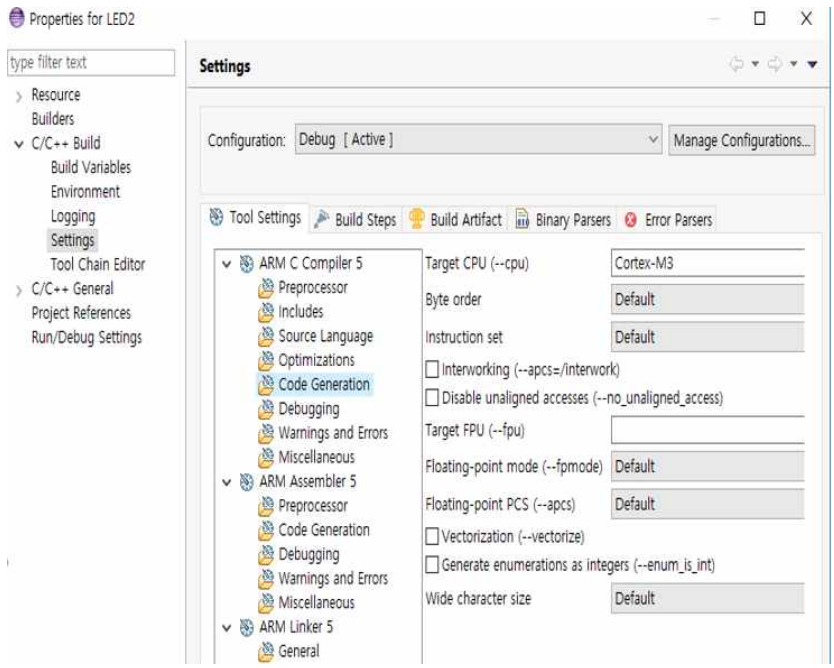
(entry point를 main으로 지정해주지 않아도 됨)

3-3) LIBRARIES 추가

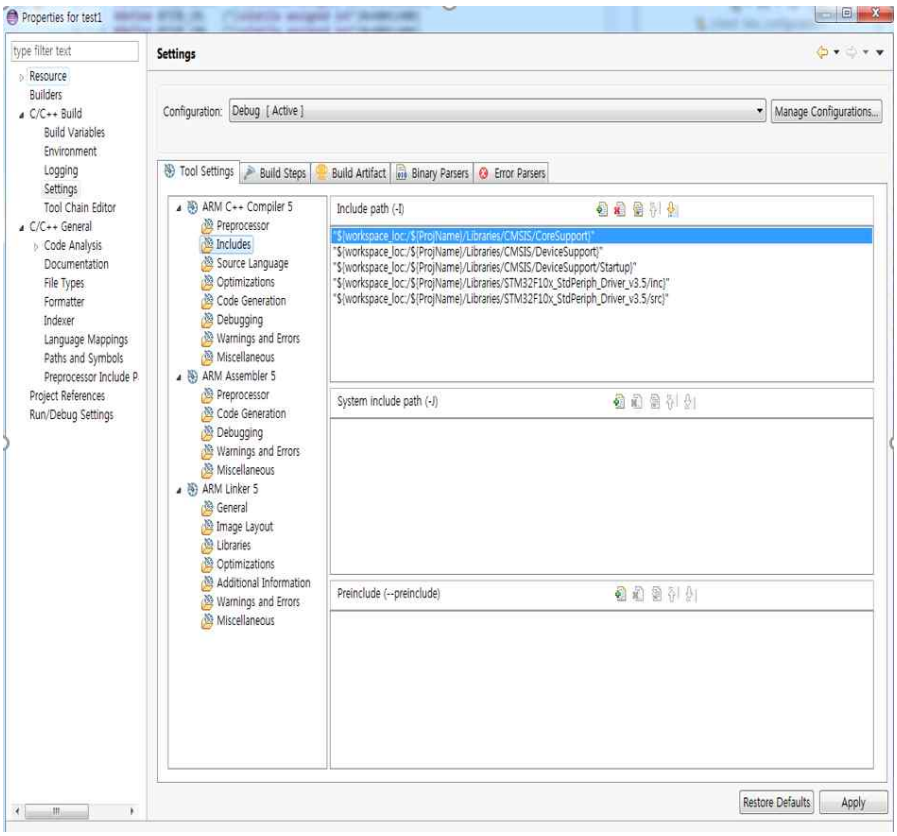
① C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → Includes에서 제공되는 LIBRARIES 추가



<그림 9 : Scatter file 설정>



<그림 10 : Target CPU 설정>



<그림 11 : 제공되는 LIBRARIES 추가>

4) DS-5 Debugger 연결

4-1) Debug Configuration 설정

① Run → Debug Configuration 메뉴를 선택

② DS-5 Debugger 더블 클릭하여 새로운 하위 오브젝트 생성

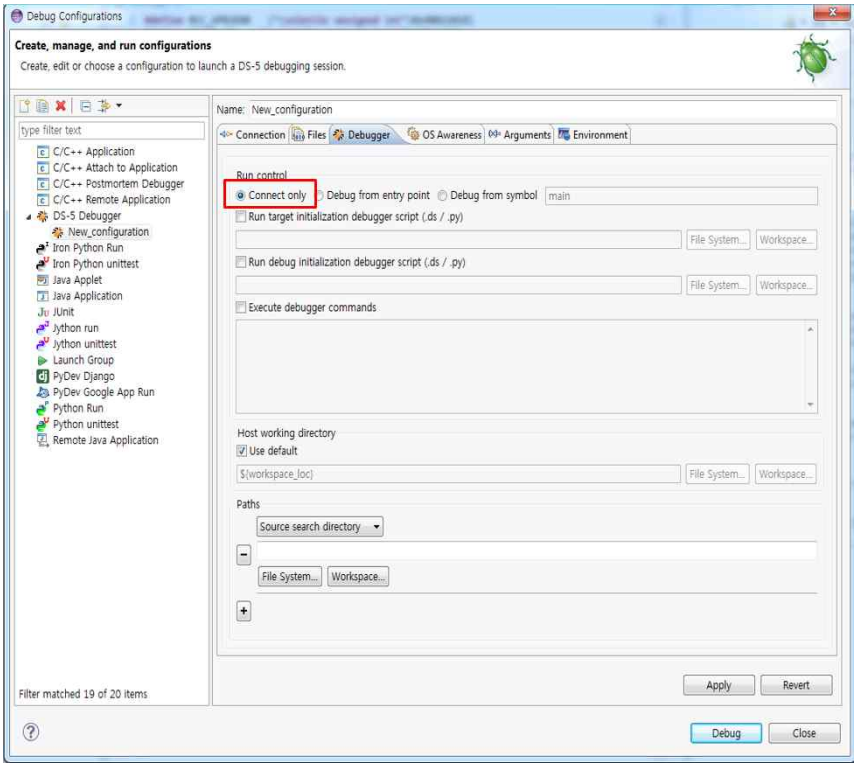
③ Name, Platform 등 Debug 환경설정을 변경

④ Browse 버튼을 클릭하여 DSTREAM 장비를 detection

4-2) Debug

① Debugger 탭에서 Connect only 체크

② Apply 클릭

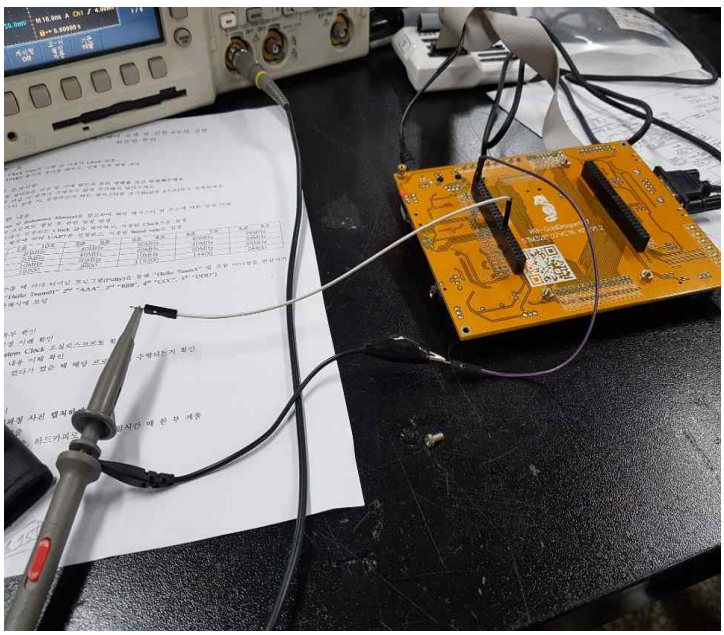


<그림 12 : Debug Configurations에서 Debugger탭의 설정화면>

5) C source code 작성

① 라이브러리 함수를 사용해 코드 작성

6) 오실로스코프 연결



<그림 14 : Cortex-M3와 오실로스코프 연결, UART Cable 과 PC 연결>

7) C project 빌드 및 .axf 업로드

7-1) C project 빌드 및 Debug As

① C Project 빌드

(.axf 파일이 생성된 것을 확인할 수 있음)

② C Project 우클릭 → Debug → Debug As

7-2) flashclear.axf 및 생성된 .axf 업로드

① command 라인에 다음과 같은 명령어를 입력해서 flashclear.axf를 업로드
flash load “flashclear.axf 파일경로”

② disconnect한 다음 보드를 껐다가 켬

③ command 라인에 다음과 같은 명령어를 입력해서 생성된 .axf를 업로드
flash load “생성된 .axf 파일경로”

④ disconnect한 다음 보드를 껐다가 켬
(flash load 후에는 반드시 diconnect를 하고 보드를 껐다가 켜야 함)

7-3) 오실로스코프에 나타나는 파형 확인

① 소스코드에서 작성한 주파수와 일치하는 지 확인한다.

② 제대로 동작하지 않으면 5) C source code 작성으로 돌아감

8) 보드 연결 해체

앞서 보드 연결과 마찬가지로, 보드 연결 해체 시에도 순서를 제대로 지키지 않으면 보드가 망가질 수 있으므로 유의해야한다. 보드 연결 해체 순서는 다음과 같다.

① DS-5에서 ‘disconnect target’ ② 보드 전원 OFF ③ DSTREAM 전원 해제 및 OFF ④ 보드 전원선 분리 ⑤ DSTREAM과 보드 JTAG 분리

<표 2 : 보드 연결 해체 순서>

4. 작성한 소스코드 및 실험결과

1) 작성한 소스코드

전체 소스코드

//flash load "C:\Users\Team07\week07\team07\flashclear\flashclear.axf"
//flash load "C:\Users\Team07\week07\team07\Debug\team07.axf"
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "core_cm3.h"
#include "misc.h"

```
int flag = 0;
char c = '1';
void delay(void) {
    int i;
    for(i=0;i<600000;++i);
}

void GPIOD_Init() {
    GPIO_InitTypeDef GPIOD_LED, GPIOD_11, GPIOD_12, GPIOC_Joystick;

    // LED Pin Set
    GPIOD_LED.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7;
    GPIOD_LED.GPIO_Mode = GPIO_Mode_Out_PP;  // GPIO Output mode = Push
Pull
    GPIOD_LED.GPIO_Speed =  GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIOD_LED);

    // Pin D11(Button) Set
    GPIOD_11.GPIO_Pin = GPIO_Pin_11;
    GPIOD_11.GPIO_Mode = GPIO_Mode_IPD; // Input pull down
    GPIOD_11.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIOD_11);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);

    // Pin D12(Button) Set
    GPIOD_12.GPIO_Pin = GPIO_Pin_12;
    GPIOD_12.GPIO_Mode = GPIO_Mode_IPD; // Input pull down
    GPIOD_12.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIOD_12);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource12);

    // Joystick Up(PC5), Down(PC2) Set
    GPIOC_Joystick.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_5;
    GPIOC_Joystick.GPIO_Mode = GPIO_Mode_IPD; // Input pull down
    GPIOC_Joystick.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIOC_Joystick);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource2);
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource5);
}

void USART1_Init() {
    USART_InitTypeDef uart;
    GPIO_InitTypeDef GPIO_USART_TX, GPIO_USART_RX;

    //GPIO_USART_RX
    GPIO_USART_RX.GPIO_Pin = GPIO_Pin_10;
    GPIO_USART_RX.GPIO_Mode = GPIO_Mode_IN_FLOATING; // Input pull down
    GPIO_USART_RX.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_USART_RX);
    //GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource10);

    //GPIO_USART_TX
    GPIO_USART_TX.GPIO_Pin = GPIO_Pin_9;
```

```
    GPIO_USART_TX.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_USART_TX.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_USART_TX);
    //GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource9);

    uart.USART_BaudRate = 9600;
    uart.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    uart.USART_Mode = USART_Mode_Rx|USART_Mode_Tx;
    uart.USART_Parity = USART_Parity_Even;
    uart.USART_WordLength = USART_WordLength_9b;
    uart.USART_StopBits = USART_StopBits_1;

    USART_Init(USART1, &uart);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
    USART_Cmd(USART1, ENABLE);
}

void EXTI_SET() {
    EXTI_InitTypeDef EXTI_Pin_2, EXTI_Pin_5, EXTI_Pin_11, EXTI_Pin_12;

    EXTI_Pin_2.EXTI_Line = EXTI_Line2;
    EXTI_Pin_2.EXTI_LineCmd = ENABLE;
    EXTI_Pin_2.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_Pin_2.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_Init(&EXTI_Pin_2);

    EXTI_Pin_5.EXTI_Line = EXTI_Line5;
    EXTI_Pin_5.EXTI_LineCmd = ENABLE;
    EXTI_Pin_5.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_Pin_5.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_Init(&EXTI_Pin_5);

    EXTI_Pin_11.EXTI_Line = EXTI_Line11;
    EXTI_Pin_11.EXTI_LineCmd = ENABLE;
    EXTI_Pin_11.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_Pin_11.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_Init(&EXTI_Pin_11);

    EXTI_Pin_12.EXTI_Line = EXTI_Line12;
    EXTI_Pin_12.EXTI_LineCmd = ENABLE;
    EXTI_Pin_12.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_Pin_12.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_Init(&EXTI_Pin_12);
}

void NVIC_SET() {
    NVIC_InitTypeDef NVIC_Pin_2, NVIC_Pin_9_5, NVIC_Pin_15_10, USART1_NVIC;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);


    // EXTI_Line 2
    NVIC_Pin_2.NVIC_IRQChannel = EXTI2_IRQn;
    NVIC_Pin_2.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_Pin_2.NVIC_IRQChannelSubPriority = 0;
```

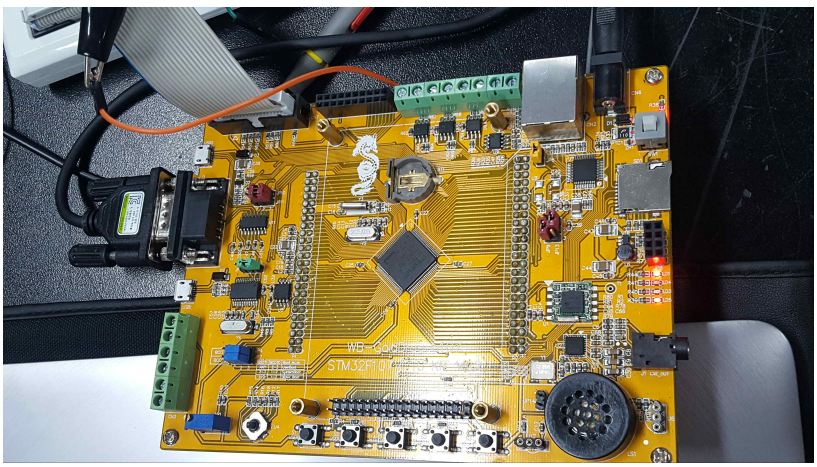
```
    NVIC_Pin_2.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_Pin_2);

    // EXTI_Line 5 ~ 9
    NVIC_Pin_9_5.NVIC_IRQChannel = EXTI9_5_IRQn;
    NVIC_Pin_9_5.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_Pin_9_5.NVIC_IRQChannelSubPriority = 1;
    NVIC_Pin_9_5.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_Pin_9_5);

    // EXTI_Line 10 ~ 15
    NVIC_Pin_15_10.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_Pin_15_10.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_Pin_15_10.NVIC_IRQChannelSubPriority = 2;
    NVIC_Pin_15_10.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_Pin_15_10);

    // USART1
    USART1_NVIC.NVIC_IRQChannel = USART1_IRQn;
    USART1_NVIC.NVIC_IRQChannelPreemptionPriority = 0;
    USART1_NVIC.NVIC_IRQChannelSubPriority = 3;
    USART1_NVIC.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&USART1_NVIC);
}

void EXTI2_IRQHandler(void) {
    if(EXTI_GetITStatus(EXTI_Line2) != RESET) {
        flag = 1;
    }

    EXTI_ClearITPendingBit(EXTI_Line2);
    EXTI_ClearITPendingBit(EXTI_Line5);
    EXTI_ClearITPendingBit(EXTI_Line11);
    EXTI_ClearITPendingBit(EXTI_Line12);

}

void EXTI9_5_IRQHandler(void) {
    if(EXTI_GetITStatus(EXTI_Line5) != RESET) {
        flag = 2;
    }
    EXTI_ClearITPendingBit(EXTI_Line2);
    EXTI_ClearITPendingBit(EXTI_Line5);
    EXTI_ClearITPendingBit(EXTI_Line11);
    EXTI_ClearITPendingBit(EXTI_Line12);
}

void EXTI15_10_IRQHandler(void) {
    if(EXTI_GetITStatus(EXTI_Line11) != RESET) {
        USART_SendData(USART1, 'G');
    }
    if(EXTI_GetITStatus(EXTI_Line12) != RESET) {
        USART_SendData(USART1, 'H');
    }
```

```
    EXTI_ClearITPendingBit(EXTI_Line2);
    EXTI_ClearITPendingBit(EXTI_Line5);
    EXTI_ClearITPendingBit(EXTI_Line11);
    EXTI_ClearITPendingBit(EXTI_Line12);
}

void USART1_IRQHandler(void)
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET) {
        c = (char)USART_ReceiveData(USART1);
    }
    USART_ClearITPendingBit(USART1, USART_IT_RXNE);
    EXTI_ClearITPendingBit(EXTI_Line2);
    EXTI_ClearITPendingBit(EXTI_Line5);
    EXTI_ClearITPendingBit(EXTI_Line11);
    EXTI_ClearITPendingBit(EXTI_Line12);
}


int main() {
    SystemInit();

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO | RCC_APB2Periph_GPIOA |
RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD | RCC_APB2Periph_USART1,
ENABLE);
    GPIOD_Init();
    USART1_Init();
    EXTI_SET();
    NVIC_SET();

    while(1){
        if(flag == 2){
            if(c != '1'){
                GPIO_SetBits(GPIOD, GPIO_Pin_2);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            }
            if(c != '2'){
                GPIO_SetBits(GPIOD, GPIO_Pin_3);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_3);
            }
            if(c != '3'){
                GPIO_SetBits(GPIOD, GPIO_Pin_4);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_4);
            }
            if(c != '4'){
                GPIO_SetBits(GPIOD, GPIO_Pin_7);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_7);
            }

        }
```

```
        else if(flag == 1){
            if(c != '4'){
                GPIO_SetBits(GPIOD, GPIO_Pin_7);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_7);
            }
            if(c != '3'){
                GPIO_SetBits(GPIOD, GPIO_Pin_4);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_4);
            }
            if(c != '2'){
                GPIO_SetBits(GPIOD, GPIO_Pin_3);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_3);
            }
            if(c != '1'){
                GPIO_SetBits(GPIOD, GPIO_Pin_2);
                delay();
                GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            }

        }
    }
}
```

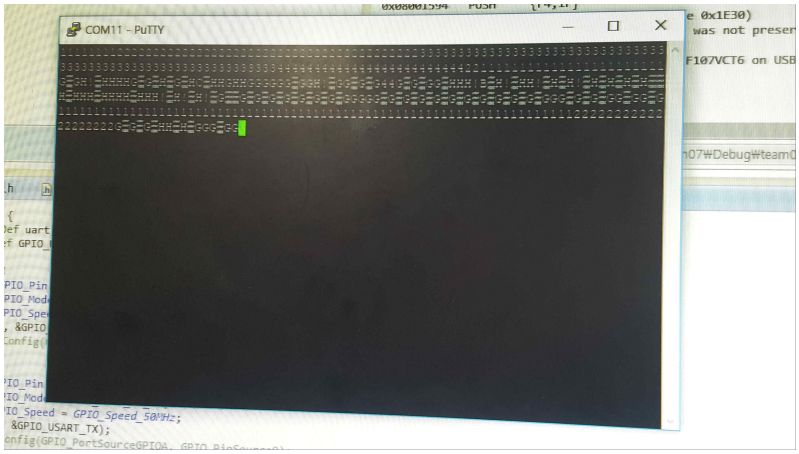
2) 실험결과

2-1) Putty를 통해 숫자를 입력할 경우 입력받은 숫자 제외하고 LED물결 출력

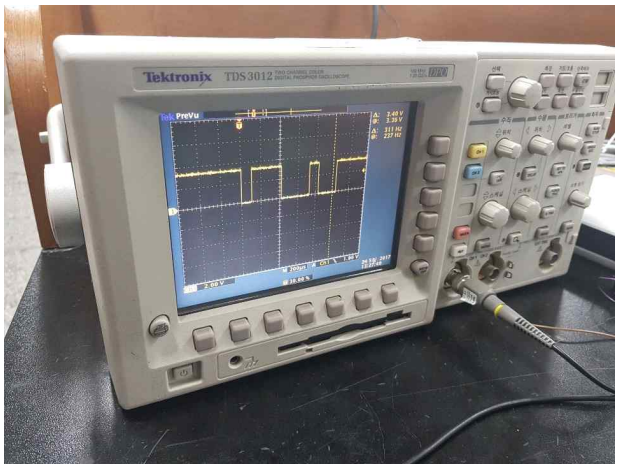


<그림 15 LED물결 출력>

2-2) 버튼1, 버튼2 누를 경우 Putty로 문자(G, H) 출력 및 오실로스코프로 확인



<그림 16 Putty를 통한 입출력 결과>



<그림 17 오실로스코프를 통한 파형 확인>

5. 결론 및 느낀점

기존 실험에서는 polling 방식을 통해 CPU가 event를 처리했다면, 이번 실험에서는 interrupt 방식을 통하여 GPIO를 제어하고 UART 통신을 해보았다. 여러 주어진 event(조이스틱 조작, 버튼입력 등)에 대해 interrupt handling을 세팅하고 각 event들에 대해서 priority를 설정해보았는데, interrupt handling을 세팅하는 과정에서 NVIC와 EXTI의 개념에 대해 이해하고 학습하는 시간을 가질 수 있었다.

이번 실험에서도 저번 실험과 마찬가지로 header file에 #define되어 있는 값들을 활용하여 소스코드를 작성하였다. 초반 실험에서 소스코드를 작성하던 방식과 달라 익숙하지 않았지만, header file을 활용하니 복잡한 소스코드가 훨씬 더 직관적으로 이해할 수 있어서 소스코드 작성 및 수정 시간이 빨라지고 가독성도 높일 수 있어서 좋았다.

6. 참고 문헌 및 사이트

[1] 부산대학교 ENS 연구실 (https://enslab.pusan.ac.kr/)
[2] 2018_7주차_임베디드_시스템_설계_및_실험 예비조 발표자료