임베디드 시스템 설계 및 실험 6주차 보고서

002분반 3조

201724601 최성렬

201729163 이희근

202055522 김은지

202055574 이다은

202055590 장서윤

1. 실험 목표

이번 실험의 목표는 조이스틱과 Putty(UART)를 이용하여 보드의 led를 순차적으로 제어할 수 있고, Interrupt를 이용하여 동작의 지연시간이 없도록 한다. 그리고 S1버튼을 누를 경우, Putty로 "TEAM 03"을 출력할 수 있도록 한다.

보드를 켜면 LED 물결 기능 유지 (LED 1->2->3->4->1->2->3->4->1->... 반복)
(A) LED 순차 변경 - 1->2->3->4 / (B) LED 순차 변경 - 4->3->2->1
(물결 속도는 delay를 이용하여 천천히 동작, ISR에서는 delay가 없어야 합니다)
(전원 스위치에 가까운 LED = 1이고 차례대로 2,3,4로 생각하시면 됩니다.)

2. 실험 내용

실험을 본격적으로 진행하기 전에, 다음과 같은 기본적인 세팅을 해주어야 한다.

- Clock Enable 수행

```
/* UART TX/RX port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
/* JoyStick Up/Down port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
/* 필요없음 JoyStick Selection port clock enable */
/* LED port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
/* USART1 clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
/* Alternate Function IO clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
```

- GPIO Configuration 수행

  조이스틱, 버튼, LED, UART를 각각 세팅해주기 위해 핀 번호 연결해준다. '|'연산자로 필요한 핀 번호 모두 연결하도록 한다.

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_5;
```

  인풋은 다음 사진과 같이 설정해주도록 한다. (input with pull-up/pull down mode)

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU | GPIO_Mode_IPD;
```

  아웃풋은 다음 사진과 같이 설정해주도록 한다,. (Alternate function output Push-pull)

```
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

  핀 초기화(GPIO{포트}, &GPIO_InitStructure)

```
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

이제는 본격적으로 실험의 세부적인 내용은 다음과 같다.

(1) NVIC와 EXTI를 이용하여 GPIO에 인터럽트 핸들링 세팅

```
/* Joystick Down PC 2 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource2);
EXTI_InitStructure.EXTI_Line = EXTI_Line2;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling; //제공코드
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

/* Joystick Up PC 5*/
GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource5);
EXTI_InitStructure.EXTI_Line = EXTI_Line5;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

/* 필요없음 Joystick Selection */

/* Button  PD11 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
EXTI_InitStructure.EXTI_Line = EXTI_Line11;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
// NOTE: do not select the UART GPIO pin used as EXTI Line here
```

코드는 위와 같으며, EXTI_Line의 뒤에 숫자는 조이스틱, 버튼의 PC,PD 뒤에 있는 숫자를 그대로 붙여주면 되고, 우리는 Interrupt를 사용할 것이므로 EXTI_Mode도 Interrupt로 작성해주도록 한다.

```
// Joystick Down PC2
NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;        // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

// Joystick Up PC5
NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;        // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

// User S1 Button PA11
NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 3; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;        // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

// UART1
// 'NVIC_EnableIRQ' is only required for USART setting
NVIC_EnableIRQ(USART1_IRQn);
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;        // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

NVIC_IRQChannel은 Line 1(EXIT1)/2(EXIT2)/3(EXIT3)/4(EXIT4)/5~9(EXIT9_5)/10~15(EXIT15_10)으로 나눠지므로 각 번호에 맞게 지정해주면 되겠다.

그리고 NVIC_IRQChannelPreemptionPriority 는 주어진 대로 1.조이스틱 up, 2.조이스틱 down, 3.버튼, 4.USART 입력 순서대로 지정해주었다.

(2) 조이스틱 Up/Down 및 PC의 putty 입력에 따른 행동 및 버튼에 따른 putty 출력설정

IRQHandler 함수들은 각각 이용목적에 맞게 설정해주면 된다. USART1_IRQHandler는 putty에 'a','b'를 각각 입력해주면 작동하도록 하고, EXTI15_10_IRQHandler는 버튼을 눌렀을 때 작동하도록 하고, EXTI2_IRQHandler는 조이스틱을 down으로 했을 때, EXTI9_5_IRQHandler는 조이스틱이 up으로 했을 때 작동하도록 한다.

- USART1_IRQHandler

  Flag는 global 변수로 지정을 해 주었고, 상태를 지정해주기 위해 만들었다.

  LED가 정방향으로 움직이는 상태가 되려면 flag에 0을 입력해주고, LED가 역방향으로 움직이는 상태가 되려면 flag에 1을 입력해준다. (putty에 a가 입력되면 flag를 0으로 만들어주고, putty에 b가 입력되면 flag를 1로 만들어준다.)

```
void USART1_IRQHandler()
{
    uint16_t word;
    if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        // the most recent received data by the USART1 peripheral
        word = USART_ReceiveData(USART1);
        if (word == 'a')
        {
            flag = 0;
        }
        else if (word == 'b')
        {
            flag = 1;
        }

        // clear 'Read data register not empty' flag
        USART_ClearITPendingBit(USART1, USART_IT_RXNE);
    }
}
```

- EXTI15_10_IRQHandler

```
void EXTI15_10_IRQHandler(void)
{ // when the button is pressed => putty에 "team~ 출력"

    if (EXTI_GetITStatus(EXTI_Line11) != RESET)
    {
        if (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_11) == Bit_RESET)
        {
            // TODO implement
            int i=0;
            char str[] = "TEAM03.\r\n";
            while (str[i]!= NULL){
                sendDataUART1(str[i]);
                i++;
            }
        }
        EXTI_ClearITPendingBit(EXTI_Line11);
    }
}
```

- EXTI2_IRQHandler, EXTI9_5_IRQHandler

```
void EXTI2_IRQHandler(void)
{ // 조이스틱 down (PC2) -> b 동작 , 역방향, flag = 1

    if (EXTI_GetITStatus(EXTI_Line2) != RESET)
    {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_2) == Bit_RESET)
        {
            flag = 1;
        }
        EXTI_ClearITPendingBit(EXTI_Line2);
    }
}

void EXTI9_5_IRQHandler(void)
{ // 조이스틱 up (PC5) -> a 동작 , 순방향, flag = 0

    if (EXTI_GetITStatus(EXTI_Line5) != RESET)
    {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5) == Bit_RESET)
        {
            flag = 0;
        }
        EXTI_ClearITPendingBit(EXTI_Line5);
    }
}
```

조이스틱이 down이 된다면 flag를 1, 조이스틱이 up이 된다면 flag를 0으로 만들어준다.

이제 마지막으로 led 제어를 하는 함수를 작성해주면 되겠다. Seq 배열은 global 변수로 설정해주었으며, led 1개씩 지정해주었다. (GPIO_Pin_2,GPIO_Pin_3,GPIO_Pin_4,GPIO_Pin_7)

```
int index=0;
while (1)
{
    // TODO: implement
  GPIO_WriteBit(GPIOD,seq[index],Bit_SET);
  Delay();
  GPIO_WriteBit(GPIOD,seq[index],Bit_RESET);
  if (flag==0){
    index+=1;
    }
    else{
        if (index<=0) index=4;
        index-=1;
    }
index%=4;
    // Delay
    Delay();
}
```

flag가 0이 되면 순방향으로 가야 하므로 index가 0에서부터 하나씩 증가하도록 하고, 4보다 커질 경우 mod4를 적용시켜주도록 하였다. flag가 1이 되면 역방향이 되어야 하므로 index가 4에서부터 1씩 줄어들도록 하였다. (index가 0보다 작거나 같아질 경우 , 4로 다시 만들어주도록 하였다.)

3. 관련이론

(1) Interrupt

CPU가 다른 연산을 수행하고있던 중이라도 특정 이벤트가 발생하면 인터럽트 서비스 루틴을 수행한다. 즉, 이벤트 수행 신호 발생 -> Interrupt Service Routine -> 이벤트를 수행 -> 다시 이전 작업으로 복귀

(2) EXTI (External Interrupt)

외부에서 신호가 입력될 경우 device에 event나 Interrupt가 발생되는 기능

(3) NVIC (Nest Vectored Interrupt Controller)

중첩된 Interrupt를 제어하는 기능으로 모든 exception에 대해 Priority가 설정되어 있고, 이 우선 순위에 따라 Interrupt를 처리함

4. 실험 결과

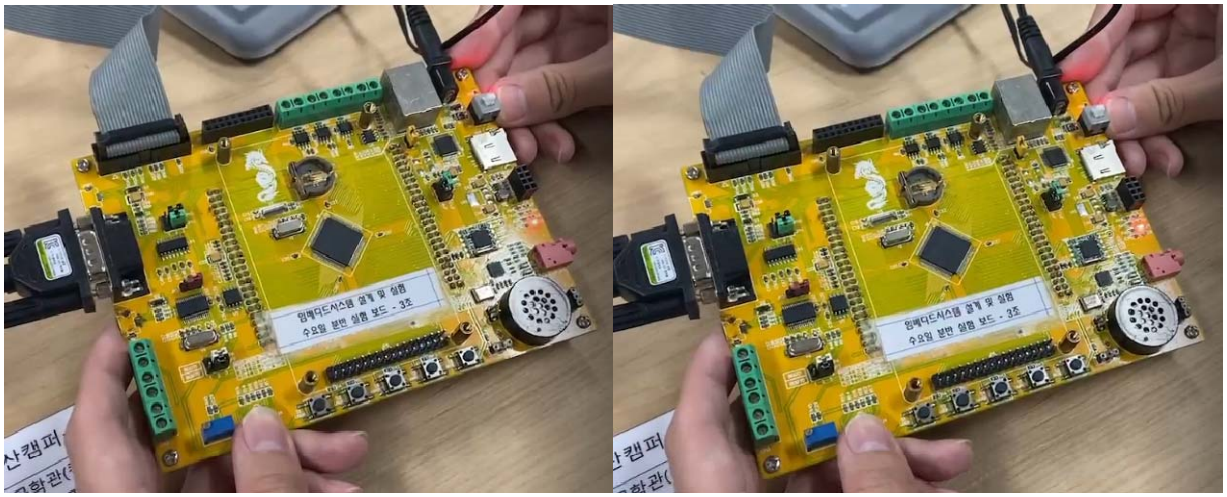

Figure. 1,2. 조이스틱을 up 해주어 순차적으로 올라간다.

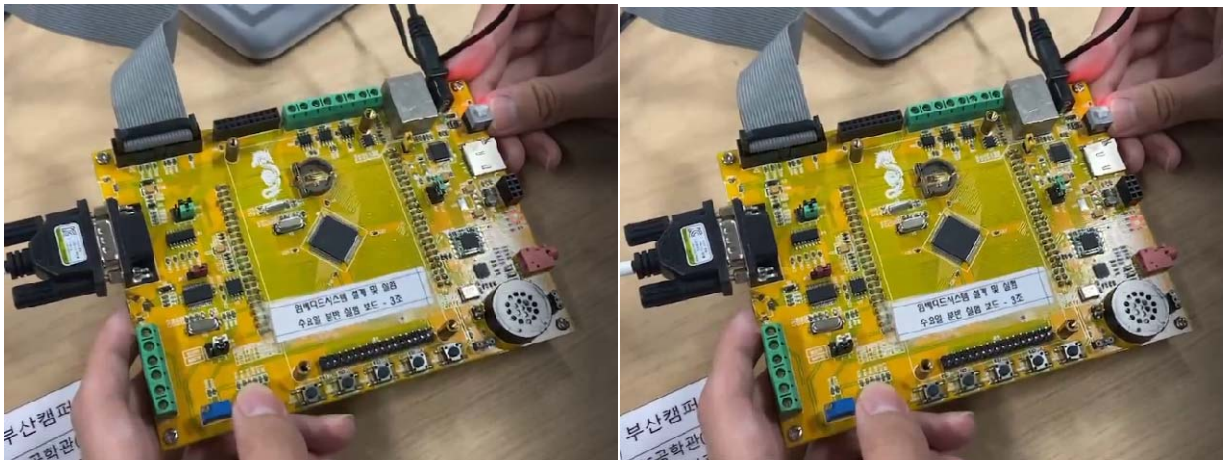

Figure. 3,4. 조이스틱을 down 해주어 순차적으로 내려간다.

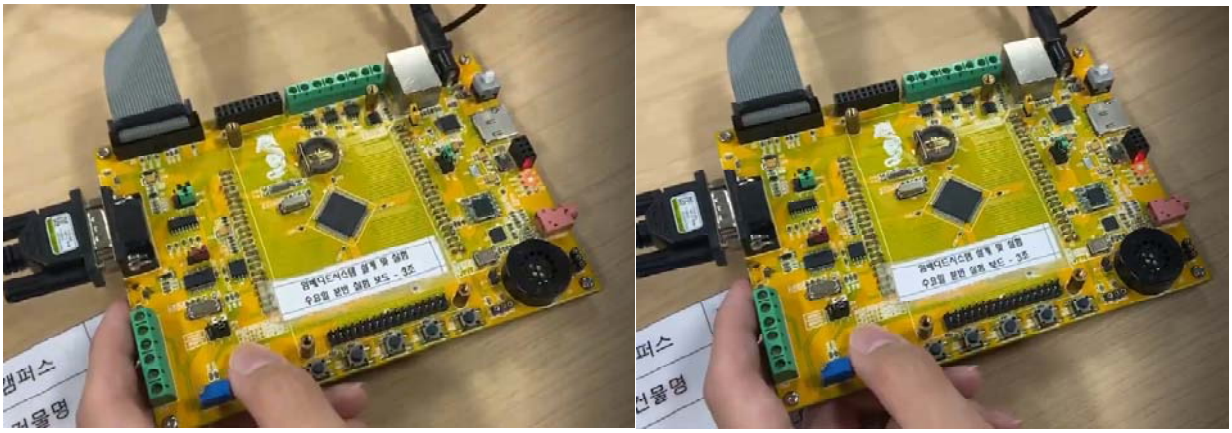

Figure.5,6. PC의 Putty에 a를 입력해주자 다시 순차적으로 올라간다.

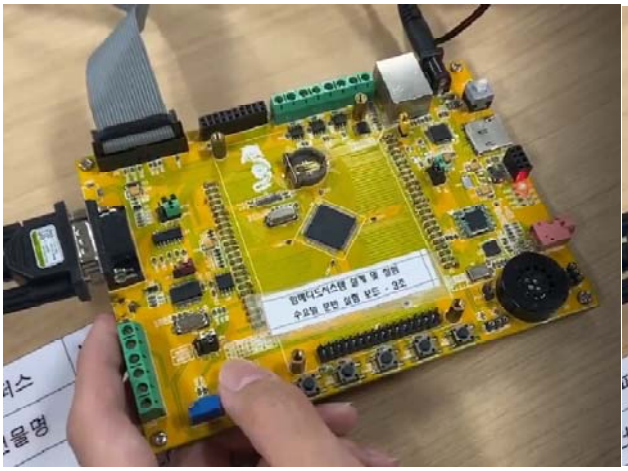
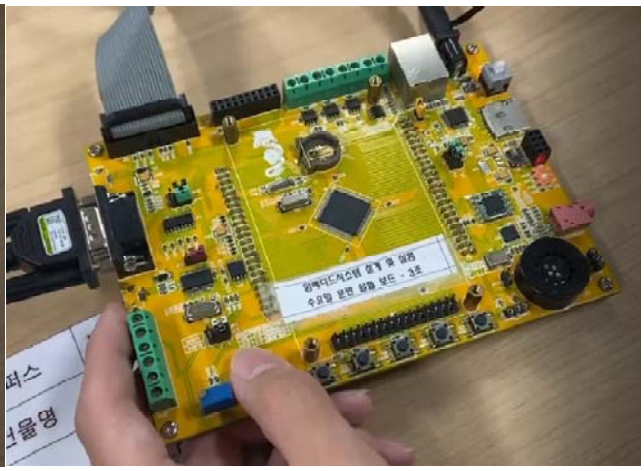

Figure.7,8. PC의 Putty에 b를 입력해주자 다시 순차적으로 내려간다.

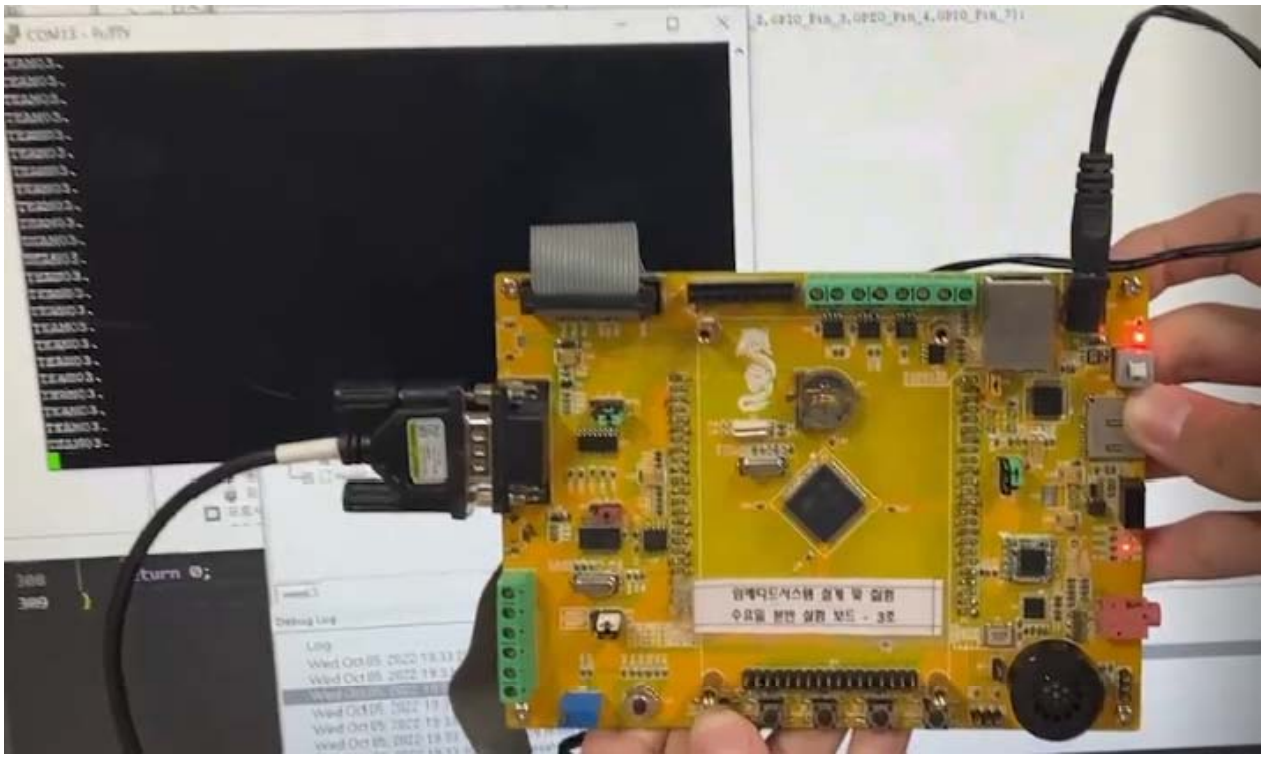

Figure.9. Button 1을 누르자, Putty에 "TEAM03." 이 출력되었다.

5. 고찰

포트를 잘 보고 헷갈리지 않아야 한다. 예를 들면 PC3, PD3 과 같이 비슷하게 생겨서 코드를 작성할 때, 헷갈릴 수 있으므로 주의하도록 해야 한다. 그리고 변수의 타입이나 변수에 대한 정보를 더 찾아 봐야하는 경우에는 Reference file이나 제공자료를 찾아보는 것이 좋다. 우리 조 같은 경우는 NVIC_IRQChannel이라는 타입을 찾아내어 사용하였다. 마지막으로 debug를 할 때, 중단점을 사용하는 것도 좋지만, 직접 중간중간에 숫자를 출력(printf("n") //n은 자연수)하도록 하여 어디에서 정상적으로 코드가 작동하지 않는지(컴파일 오류가 아닌 논리적 오류) 알아내는 것도 좋다.