임베디드 시스템 설계 및 실험
7주차 실험 결과 보고서

조 : 8조

조원 : 서진욱, 이승민, 하연지, 하태훈

● 실험목표

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

● 개념설명

1. Interrupt

A. 모든 연산을 주기적으로 감시하면서 이벤트가 발생하는 지 파악하고 이벤트가 발생 시 이를 처리하는 Polling 방식과 달리, 모든 연산을 감시하는 것이 아니라 특정 이벤트가 발생했을 때 원래 하던 작업을 일시 중단하고 이벤트 발생 시의 서비스 루틴을 수행 하고 다시 이전 작업으로 돌아가는 것

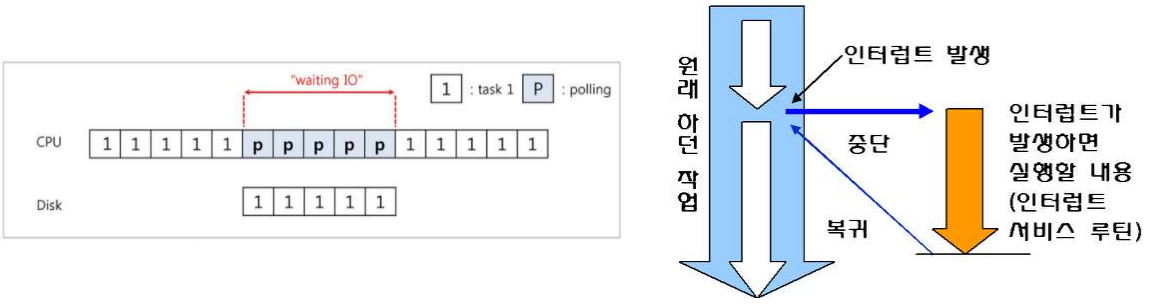


그림 1 Polling(왼쪽)과 Interrupt(오른쪽)

B. Hardware Interrupt : 비동기식으로 이벤트를 처리하는 방식으로, 주변 장치의 요청에 의해 발생하는 인터럽트로써 높은 우선순위를 가짐. 하드 디스크 읽기 요청, 디스크 읽기 끝남, 키보드 입력 등에 의해 발생

C. Software Interrupt : 동기식으로 이벤트를 처리하며 프로그램 내에서 인터럽트가 발생하도록 사용자가 설정하는 것으로 낮은 우선순위를 가짐. Trap, Exception 등에 의해 발생

2. EXTI(External Interrupt)

A. 외부에서 신호가 입력되는 경우 이벤트나 인터럽트가 발생하도록 하는 기능으로, Rising Edge, Falling Edge, Rising & Falling Edge를 신호로써 입력 받을 수 있음

B. Event Mode와 Interrupt Mode 중 하나를 선택하여 동작이 가능하고, Interrupt Mode로 설정할 경우 설정한 Interrupt 발생 시 Interrupt Handler 가 동작하며 이를 처리함

C. STM32에서는 Port의 종류(A, B, C, D, E, F, G)에 관계 없이 같은 Port 번호(k) 일 경우 하나의 EXTIk에 대응됨. 예를 들어 PA0, PG0 모두 EXTI0에 대응됨

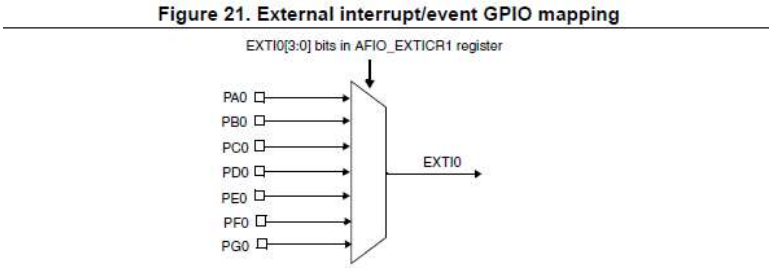


그림 2 Port 번호에 따른 EXTI 대응

D. Interrupt 요청은 Mask Register를 통해 알 수 있으며, Processor는 Interrupt를 처리하기 전에 발생한 Interrupt 정보를 저장하는 Pending Register에서 Interrupt의 우선순위를 비교하여 높은 것부터 차례로 처리함

E. 이들 EXTI line을 통해 입력 받은 신호와 레지스터 설정을 비교하여 NVIC controller로 보냄

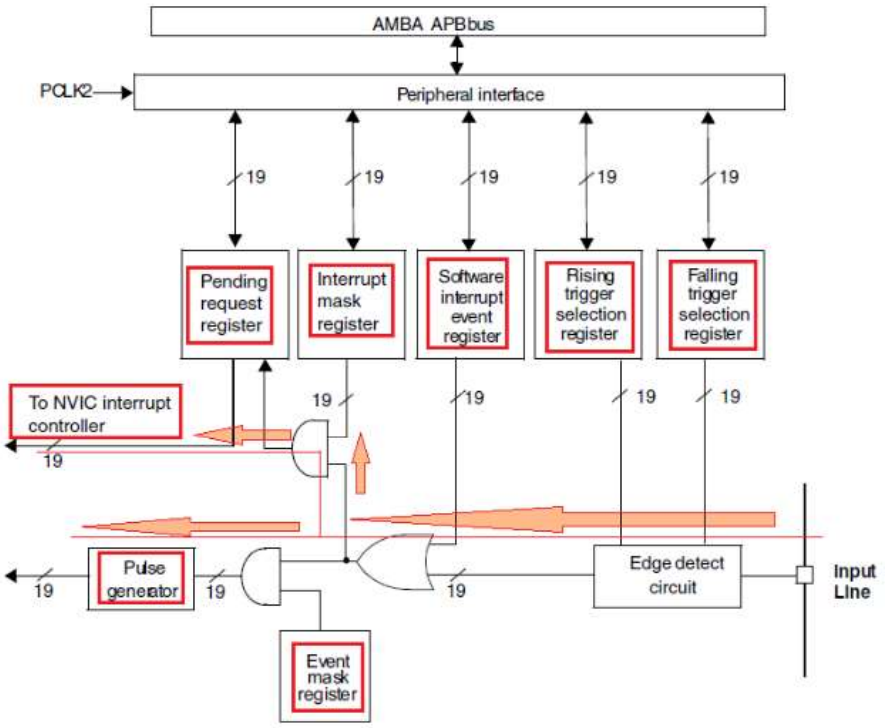


그림 3 Interrupt 처리 구조

● 실험 진행

- **코드 작성**

1. Enable the APB2 peripheral clock using the function 'RCC_APB2PeriphClockCmd'

   - RCC_APB2PeriphClockCmd함수를 사용해 GPIOA, GPIOB, GPIOC, GPIOD 포트의 클락과 USART1의 클락을 활성화시킨다. 또한 AFIO클락도 활성화해 다른 모듈과의 클락연결을 관리한다.

```
/* UART TX/RX port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
/* Button S1, S2, S3 port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, 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);
```

2. Button pin setting

```
/* Button KEY1, KEY2, KEY3 pin setting */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_13;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU | GPIO_Mode_IPD;
GPIO_Init(GPIOC, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU | GPIO_Mode_IPD;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

3. LED pin setting

```
/* LED pin setting*/
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

4. UART pin setting

```
/* UART pin setting */
//TX
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; // PUSH PULL 이었을 때, 통신이 안됨
GPIO_Init(GPIOA, &GPIO_InitStructure);
//RX
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU | GPIO_Mode_IPD;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

5. Button 눌렀을 때

- GPIO_EXTILineConfig 함수를 이용해 Button 1,2,3의 EXTI line을 설정하고 EXTI_Mode를 interrupt로 설정해 각 line이 설정에 따라 Rising/Falling Trigger를 감지할 수 있도록 한다.

```
/* Button 1(PC4) is pressed */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource4);
EXTI_InitStructure.EXTI_Line = EXTI_Line4;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

/* Button 2 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource10);
EXTI_InitStructure.EXTI_Line = EXTI_Line10;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);


/* Button 3 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource13);
EXTI_InitStructure.EXTI_Line = EXTI_Line13;
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
```

6. USART 초기화

- USART_InitStructure을 사용해 USART를 초기화

```
USART1_InitStructure.USART_BaudRate = 9600;
 USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
 USART1_InitStructure.USART_Parity = USART_Parity_No;
 USART1_InitStructure.USART_StopBits = USART_StopBits_1;
 USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
 USART1_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
 USART_Init(USART1, &USART1_InitStructure);
```

7. NVIC 초기화

- Interrupt priority 설정 및 interrupt 관리제어를 위해 NVIC 초기화를 수행

```
//Button S1
NVIC_InitStructure.NVIC_IRQChannel = EXTI4_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1; // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
// Button S2 , ButtonS3
NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

8. EXTI interrupt 핸들러함수

- IRQHandler함수를 사용해 EXTI line에서 발생하는 interrupt를 처리하도록 함

```
void EXTI4_IRQHandler(void) {
    if (EXTI_GetITStatus(EXTI_Line4) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_4)== Bit_RESET) {

        //mode = 0;
        setDirection('a');

        }
        EXTI_ClearITPendingBit(EXTI_Line4);
    }
}

void EXTI9_5_IRQHandler(void) {
    if (EXTI_GetITStatus(EXTI_Line10) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_10) == Bit_RESET) {

        //mode=1;
        setDirection('b');

        }
        EXTI_ClearITPendingBit(EXTI_Line10);
    }
}
```

9. Button 3 눌렀을 때, 메시지 전송

- Button3 을 눌렀을 때 Putty로 메시지를 전송하도록 구현한 코드

```
if (EXTI_GetITStatus(EXTI_Line13) != RESET) {
    if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == Bit_RESET) {
     // TODO implement

        char *tmp = &msg[0];
        while (*tmp !='\0') {
            sendDataUART1(*tmp);
            tmp++;
        }

     EXTI_ClearITPendingBit(EXTI_Line13);
    }
}
```

10. sendDataUART1, setDirection

- sendDataUART1는 UART1을 사용해 데이터를 전송하는 역할을 한다. 또한 setDirection으로 문자 a,b를 입력했을 때 A동작, B동작을 수행해 보드를 제어할 수 있도록 한다

```
void sendDataUART1(uint16_t data) {
   /* Wait till TC is set */
   while ((USART1->SR & USART_SR_TC) == 0);
   USART_SendData(USART1, data);
}

void setDirection(char direction) {
  if (direction == 'a'){
    mode = 'a';

  }
  else if (direction == 'b'){
    mode = 'b';
  }
}
```

11. Ledindex

- Led를 제어하는 코드로 mode가 a인 경우와 b인 경우로 나뉜다. a인 경우, led index를 증가시켜 다음 led로 이동하도록 함(LED 1->2->3->4)

- B인경우 반대방향으로 LED 4->3->2->1순서로 가도록 함

```
void ledIndex(void) {
    if(mode == 'a'){
        led_idx++;
    }

    else if(mode == 'b') {
        if(led_idx == 0){
            led_idx = 3;
        }
        else{
            led_idx--;
        }
    }
}
```

12. LED를 번갈아가며 켜고 끄고 mode에 따라 led표시 변경

- Led_idx 값을 4로 나눈 나머지 값을 index 변수에 저장해 led_idx가 0~3까지 가며 index 선택함.

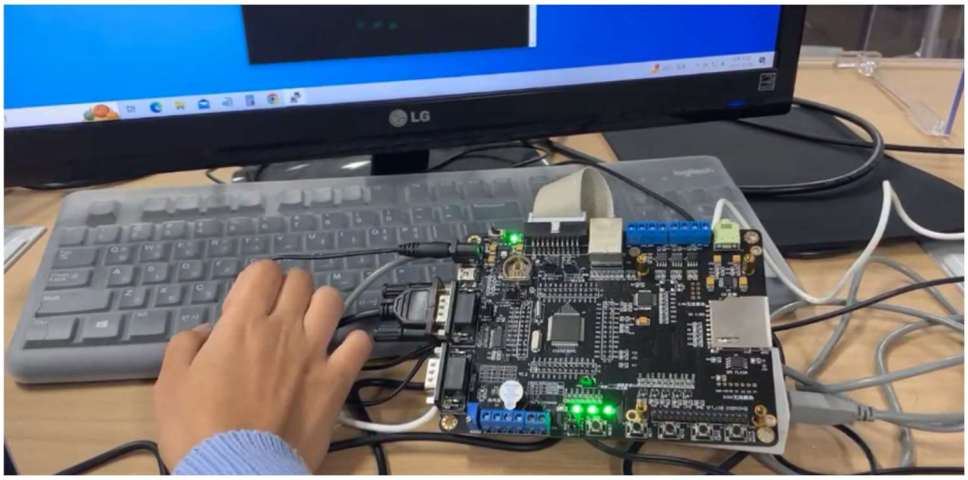
- if (GPIO_ReadOutputDataBit(GPIOD, led_array[index]) == 0) 으로 선택된 led가 꺼져있는지 확인하고 꺼져있으며 키고 켜져있으며 선택된 led를 끄도록 함.

- 현재 mode와 이전 mode를 비교하고 mode가 변경되었다면 이전모드를 업데이트함.

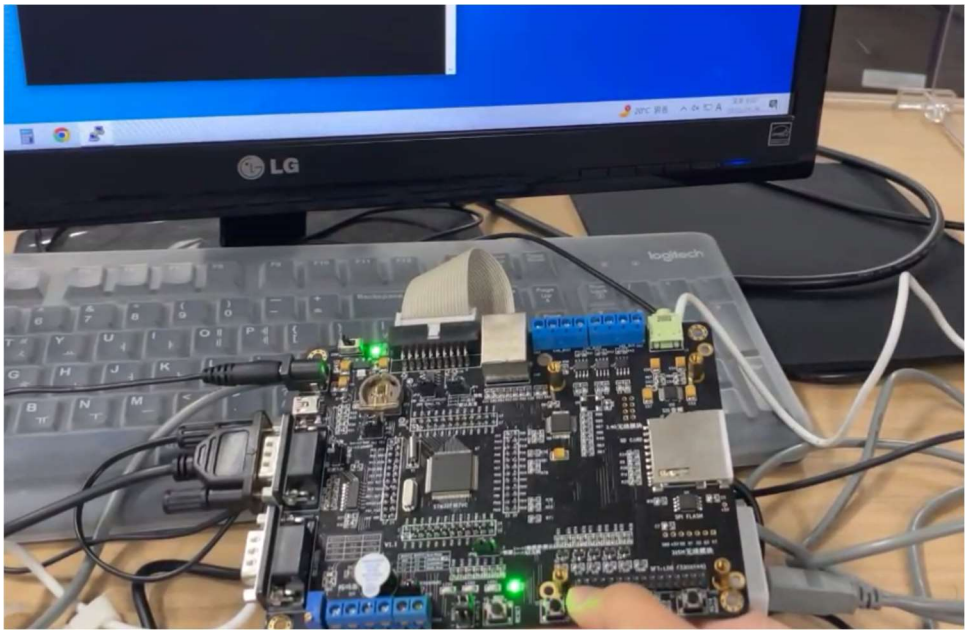
- 그렇지 않다면 ledIndex함수를 호출해 led index를 mode에 따라 변경하도록 함

```
while (1) {
    // TODO: implement

     index = led_idx % 4;
     if (GPIO_ReadOutputDataBit(GPIOD,led_array[index])==0){
       GPIO_SetBits(GPIOD, led_array[index]);
     }
     else {
       GPIO_ResetBits(GPIOD, led_array[index]);
     }


     if (prev_mode != mode) {
           prev_mode = mode;
     }

     else{
         ledIndex();
     }

    // Delay
    Delay();

}
```

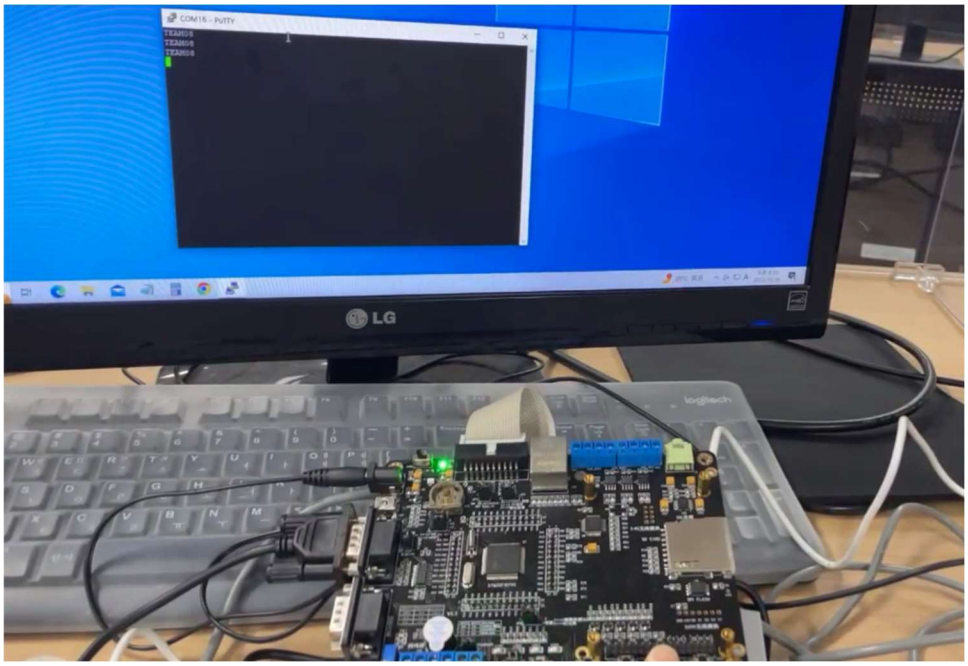
- **코드실행 및 동작 확인**



시리얼케이블로 연결되어 있는 PC의 키보드 입력을 통한 LED 제어



직접 보드의 버튼 입력을 통한 LED 제어



보드의 버튼 입력시 시리얼 케이블 PC의 putty에 메시지 출력

● 결론

- EXTI를 통해 Interrupt 발생시키고 Handler를 통해 하드웨어를 제어하는 방법을 알게 되었다.
- 지난 시간에 이어 라이브러리의 활용법을 더욱 숙지할 수 있었다.
- 보드 자체의 버튼 뿐만 아니라 시리얼 케이블로 연결된 컴퓨터의 입력장치로도 보드를 제어할 수 있고, 어떻게 제어해야 하는지 알 수 있게 되었다.