실험 결과 보고서
9주차 Interrupt GPIO 제어
및 UART 통신파일

실험 일자 : 2024. 10. 29.

실험 참여 : 3조 강태진, 김기윤, 김도완, 임성표

차례

I. 실험 목적

i) 라이브러리를 활용하여 IAR 코드 작성 및 보드 포팅

ii) Interrupt 방식을 활용한 GPIO 제어 및 UART 통신

iii) 라이브러리 함수 사용법 숙지

II. 세부 목표

i). 레지스터 및 주소에 대한 설정 이해

- Datasheet 및 Reference Manual 참고.

ii) NVIC와 EXTI를 이용해여 GPIO에 인터럽트 핸들링 세팅

- 보드에 코드를 업로드 하면 LED 물결 형태로 작동
- Putty에서 A를 입력하면 1 -> 2 -> 3 -> 4 순서로 LED 점등
- Putty에서 B를 입력하면 4 -> 3 -> 2 -> 1 순서로 LED 점등

iii) KEY 1 버튼 입력 A 입력과 같은 동작

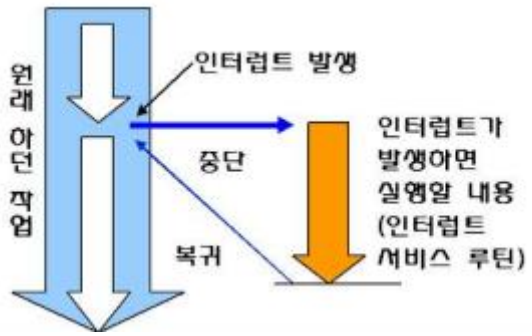
iv) KEY 2 버튼 입력 B 입력과 같은 동작

v) KEY 3 버튼을 누른 경우 Putty에서 "Team03.₩r₩r" PC화면에 출력

III. 실험 과정

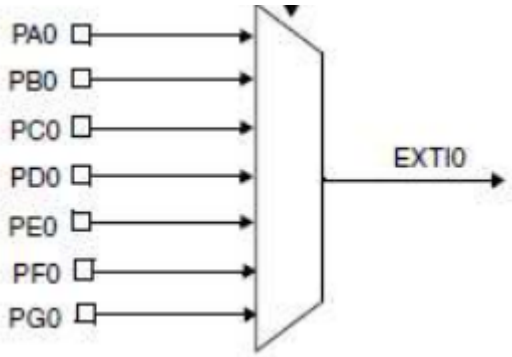
i) 배경 지식

- **Interrupt :** CPU가 특정 이벤트 발생시 현재 작업을 멈추고 해당 인터럽트 서비스 루틴을 수행 후 다시 이전 작업으로 돌아가는 방식



<Interrupt 예시 그림>

- **EXTI(External Interrupt)** : 외부에서 신호가 입력될 경우 해당 장치에 이벤트나 인터럽트가 발생하는 기능, 외부 Interrupt는 EXTI0 ~ EXTI15까지 각 Port의 Pin 번호가 Interrupt Pin과 매치
  - ■ EXTI는 외부 신호가 입력되는 여러 GPIO 포트에서 발생하는 인터럽트를 관리, 같은 번호의 핀들은 같은 EXTI 라인을 공유



<EXTI0라인이 각 포트의 0번 핀과 연결되는 모습>

ii)NVIC(Nested Vectored Interrupt Controller)

- 인터럽트가 발생할 시, 우선순위를 설정하여 인터럽트를 관리
- 우선순위가 높은 인터럽트가 발생하면, 현재 처리중인 우선순위가 낮은 인터럽트를 중단하고 높은 우선순위의 인터럽트 처리
- 값이 작을수록 우선순위가 높음

@code
The table below gives the allowed values of the pre-emption priority and subpriority according to the Priority Grouping configuration performed by NVIC_PriorityGroupConfig function

NVIC_PriorityGroup	NVIC_IRQChannelPreemptionPriority	NVIC_IRQChannelSubPriority	Description
NVIC_PriorityGroup_0	0	0-15	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PriorityGroup_1	0-1	0-7	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PriorityGroup_2	0-3	0-3	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PriorityGroup_3	0-7	0-1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PriorityGroup_4	0-15	0	4 bits for pre-emption priority 0 bits for subpriority

@endcode

<Priority 및 SubPriority에 따른 비트구성 방식>

iii) 추가 라이브러리의 구조체와 함수

- stm32f10x.h에서 GPIO 구조체를 사용해서 핀, 속도, 모드등의 설정을 지정 가능
- GPIO_SetBits와 GPIO_ResetBits 함수를 통해 핀을 켜고 끄는 동작 구현 가능

IV. 실험 시행

i) RCC_Configure

```
/* 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);
```

<RCC_Configure 내부 코드>

RCC_APB2PeriphClockCmd 함수를 사용해서, 사용할 포트의 클럭을 활성화

ii) GPIO_Configure

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_13;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

<GPIO_Configure 내부의 Key1, Key3핀 세팅>

- GPIO_InitTypeDef GPIO_InitStructure; 코드를 통해서, GPIO핀을 설정하는 구조체를 호출

```
typedef struct
{
  uint16_t GPIO_Pin;             /*!< Specifies the GPIO pins to be configured.
                                  This parameter can be any value of @ref GPIO_pins_define */

  GPIOSpeed_TypeDef GPIO_Speed;  /*!< Specifies the speed for the selected pins.
                                  This parameter can be a value of @ref GPIOSpeed_TypeDef */

  GPIOMode_TypeDef GPIO_Mode;    /*!< Specifies the operating mode for the selected pins.
                                  This parameter can be a value of @ref GPIOMode_TypeDef */
}GPIO_InitTypeDef;
```

<Stm32f10x_gpio.h에서 GPIO_InitTypeDef 구조체를 정의>

```
/**
  * @brief  Initializes the GPIOx peripheral according to the specified
  *         parameters in the GPIO_InitStruct.
  * @param  GPIOx: where x can be (A..G) to select the GPIO peripheral.
  * @param  GPIO_InitStruct: pointer to a GPIO_InitTypeDef structure that
  *         contains the configuration information for the specified GPIO peripheral.
  * @retval None
  */
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
```

<Stm32f10x_gpio.c에서 GPIO_Init 함수를 정의>

- 위와 같은 방식으로 Key2, LED, Tx와 Rx핀을 설정
- 송신 Tx핀은 Alternate function Push-pull모드로, 수신 Rx핀은 Input Pull-up모드로 설정

iii) EXTI_Configure

```
/* Button S1 PC4 */
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);
```

<Key1에 대한 EXTI라인의 핀 설정>

- EXTI_InitTypeDef EXTI_InitStructure; 코드를 통해서, EXTI 라인을 설정하는 구조체를 호출

```
typedef struct
{
  uint32_t EXTI_Line;
  /*!< Specifies the EXTI lines to be enabled or disabled.
  This parameter can be any combination of @ref EXTI_Lines */
  EXTIMode_TypeDef EXTI_Mode;
  /*!< Specifies the mode for the EXTI lines.
  This parameter can be a value of @ref EXTIMode_TypeDef */
  EXTITrigger_TypeDef EXTI_Trigger;
  /*!< Specifies the trigger signal active edge for the EXTI lines.
  This parameter can be a value of @ref EXTIMode_TypeDef */
  FunctionalState EXTI_LineCmd;
  /*!< Specifies the new state of the selected EXTI lines.
  This parameter can be set either to ENABLE or DISABLE */
}EXTI_InitTypeDef;
```

<Stm32f10x_exti.h에서 EXTI_InitTypeDef 구조체를 정의>

```
/**
  * @brief  Initializes the EXTI peripheral according to the specified
  *         parameters in the EXTI_InitStruct.
  * @param  EXTI_InitStruct: pointer to a EXTI_InitTypeDef structure
  *         that contains the configuration information for the EXTI peripheral.
  * @retval None
  */
void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct)
```

<Stm32f10x_exti.c에서 EXTI_Init 함수를 정의>

- 위와 같은 방식으로 Key2, Key3에 대한 EXTI 라인을 설정

iv) USART1_Init

```
typedef struct
{
  uint32_t USART_BaudRate;
  uint16_t USART_WordLength;
  uint16_t USART_StopBits;
  uint16_t USART_Parity;
  uint16_t USART_Mode;
  uint16_t USART_HardwareFlowControl;
} USART_InitTypeDef;
```

<stm32f10x_usart.h에서 USART_InitTypeDef 구조체를 정의>

```
/**
  * @brief  Initializes the USARTx peripheral according to the specified
  *         parameters in the USART_InitStruct .
  * @param  USARTx: Select the USART or the UART peripheral.
  *   This parameter can be one of the following values:
  *   USART1, USART2, USART3, UART4 or UART5.
  * @param  USART_InitStruct: pointer to a USART_InitTypeDef structure
  *         that contains the configuration information for the specified USART
  *         peripheral.
  * @retval None
  */
void USART_Init(USART_TypeDef* USARTx, USART_InitTypeDef* USART_InitStruct)
```

<stm32f10x_usart.c에서 USART_Init 함수를 정의>

- USART_InitTypeDef USART1_InitStructure; 코드를 통해 구조체를 호출하고, USART_Cmd(USART1, ENABLE); 코드를 사용해서 USART1 모듈을 활성화 시킨 후, 정의에 따라서 USART를 설정

```
/**
  * @brief  Enables or disables the specified USART interrupts.
  * @param  USARTx: Select the USART or the UART peripheral.
  *   This parameter can be one of the following values:
  *   USART1, USART2, USART3, UART4 or UART5.
  * @param  USART_IT: specifies the USART interrupt sources to be enabled or disabled.
  *   This parameter can be one of the following values:
  *     @arg USART_IT_CTS:  CTS change interrupt (not available for UART4 and UART5)
  *     @arg USART_IT_LBD:  LIN Break detection interrupt
  *     @arg USART_IT_TXE:  Transmit Data Register empty interrupt
  *     @arg USART_IT_TC:   Transmission complete interrupt
  *     @arg USART_IT_RXNE: Receive Data register not empty interrupt
  *     @arg USART_IT_IDLE: Idle line detection interrupt
  *     @arg USART_IT_PE:   Parity Error interrupt
  *     @arg USART_IT_ERR:  Error interrupt(Frame error, noise error, overrun error)
  * @param  NewState: new state of the specified USARTx interrupts.
  *   This parameter can be: ENABLE or DISABLE.
  * @retval None
  */
void USART_ITConfig(USART_TypeDef* USARTx, uint16_t USART_IT, FunctionalState NewState)
```

<stm32f10x_usart.c에서 USART_ITConfig 함수를 정의>

- USART_ITConfig함수를 사용해서 'Receive Data register not empty interrupt'인 USART1 RX인터럽트를 활성화
- USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); 활성화 하는 코드

v) NVIC_Configure

```
typedef struct
{
  uint8_t NVIC_IRQChannel;
  uint8_t NVIC_IRQChannelPreemptionPriority;
  uint8_t NVIC_IRQChannelSubPriority;
  FunctionalState NVIC_IRQChannelCmd;
} NVIC_InitTypeDef;
```

<misc.h에서 NVIC_InitTypeDef 구조체를 정의>

```
/**
  * @brief  Initializes the NVIC peripheral according to the specified
  *         parameters in the NVIC_InitStruct.
  * @param  NVIC_InitStruct: pointer to a NVIC_InitTypeDef structure that contains
  *         the configuration information for the specified NVIC peripheral.
  * @retval None
  */
void NVIC_Init(NVIC_InitTypeDef* NVIC_InitStruct)
```

<misc.c에서 NVIC_Init 함수를 정의>

```
/**
  * @brief  Configures the priority grouping: pre-emption priority and subpriority.
  * @param  NVIC_PriorityGroup: specifies the priority grouping bits length.
  *   This parameter can be one of the following values:
  *     @arg NVIC_PriorityGroup_0: 0 bits for pre-emption priority
  *                                4 bits for subpriority
  *     @arg NVIC_PriorityGroup_1: 1 bits for pre-emption priority
  *                                3 bits for subpriority
  *     @arg NVIC_PriorityGroup_2: 2 bits for pre-emption priority
  *                                2 bits for subpriority
  *     @arg NVIC_PriorityGroup_3: 3 bits for pre-emption priority
  *                                1 bits for subpriority
  *     @arg NVIC_PriorityGroup_4: 4 bits for pre-emption priority
  *                                0 bits for subpriority
  * @retval None
  */
void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)
{
  /* Check the parameters */
  //assert_param(IS_NVIC_PRIORITY_GROUP(NVIC_PriorityGroup));

  /* Set the PRIGROUP[10:8] bits according to NVIC_PriorityGroup value */
  SCB->AIRCR = AIRCR_VECTKEY_MASK | NVIC_PriorityGroup;
}
```

<misc.c에서 NVIC_PriorityGroup 함수를 정의>

- NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); 를 사용해서 우선순위를 2비트, 서브 우선순위를 2비트로 설정하도록 그룹2를 사용한다.

```
NVIC_InitStructure.NVIC_IRQChannel = EXTI4_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

<Key1의 NVIC를 설정하는 코드>

- 이와 같은 방식으로 Key2, Key3, USART1의 NVIC를 설정

	우선순위	서브 우선순위
Key1	0	0
Key2	1	1
Key3	2	2
USART1	3	3

vi) USART1_IRQHandler, EXIT15_10_IRQHandler, EXIT4_IRQHandler

- USART1, Key1, 2, 3에 대한 EXIT의 Handler 함수를 구현
- direction값이 1이면 LED가 1->2->3->4순서대로 점등
- direction값이 0이면 LED가 4->3->2->1순서대로 점등

```
word = USART_ReceiveData(USART1);

// TODO implement
if(word == 'a') {
    direction = 1;
}
else if (word == 'b') {
    direction = 0;
}
// clear 'Read data register not empty' flag
USART_ClearITPendingBit(USART1,USART_IT_RXNE);
```

<USART1의 Handler에서 Putty에서 입력하는 값에 따라 방향 조절>

- Putty에서 A를 입력하면 direction을 1로, B를 입력하면 direction을 0으로 변경
- Key1은 A를 입력한 결과와 동일, Key2는 B를 입력한 결과와 동일
- Key3를 입력하면 Putty화면에 “Team03₩r₩n”를 수행한 결과를 출력

```
//S3, PC13
if (EXTI_GetITStatus(EXTI_Line13) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == Bit_RESET) {
        // TODO implement
          char msg[] = "Team03.\r\n";
          for (int j = 0; ; j++) {
            sendDataUART1(msg[j]);
            if(msg[j] == '\n') break;
          }
        }
    EXTI_ClearITPendingBit(EXTI_Line13);
    }
```

<EXIT15_10_IRQHandler에서 Key3를 입력받은 경우의 함수>

vii) sendDataUART1

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

<sendDataUART1의 함수 내부 코드>

- USART1->SR, USART_SR_TC를 확인해서 상태 레지스터와 전송 완료 플래그를 확인 데이터를 보낼 준비가 되고, 전송이 완료된 경우 USART1을 통해 새로운 데이터를 전송

viii) Main

```
int i = 0;
int pin[] = {GPIO_Pin_2,GPIO_Pin_3,GPIO_Pin_4,GPIO_Pin_7};

while (1) {
    // TODO: implement
    // LED 1->2->3->4
    if(direction){
      i=(i+1)%4;
    }
    // LED 4->3->2->1
    if(!direction){
      i=(i+3)%4;
    }
    GPIO_SetBits(GPIOD,GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_7);
    GPIO_ResetBits(GPIOD, pin[i]);
    // Delay
    Delay();
}
```

<main 함수에서 LED점등>

```
/**
  * @brief  Sets the selected data port bits.
  * @param  GPIOx: where x can be (A..G) to select the GPIO peripheral.
  * @param  GPIO_Pin: specifies the port bits to be written.
  *   This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
  * @retval None
  */
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
```

<stm32f10x_gpio.c에서 정의된 GPIO_SetBits 함수>

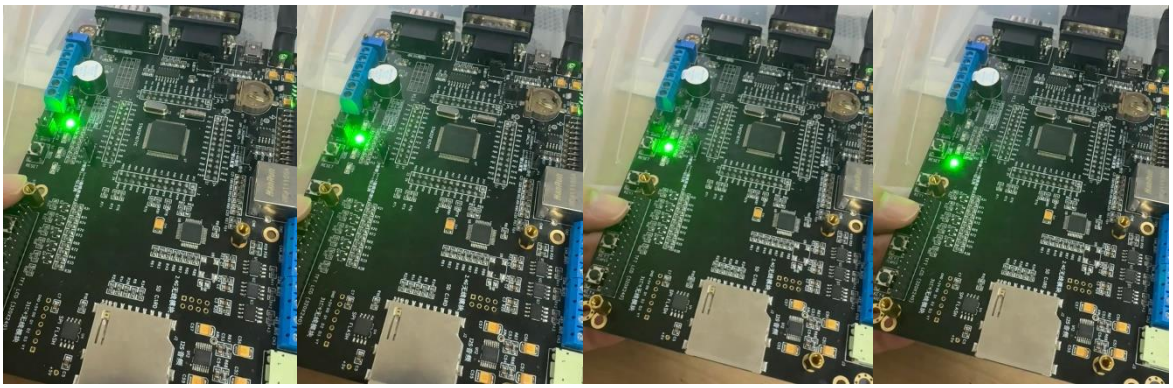
```
/**
  * @brief  Clears the selected data port bits.
  * @param  GPIOx: where x can be (A..G) to select the GPIO peripheral.
  * @param  GPIO_Pin: specifies the port bits to be written.
  *   This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
  * @retval None
  */
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
```

<stm32f10x_gpio.c에서 정의된 GPIO_ResetBits 함수>

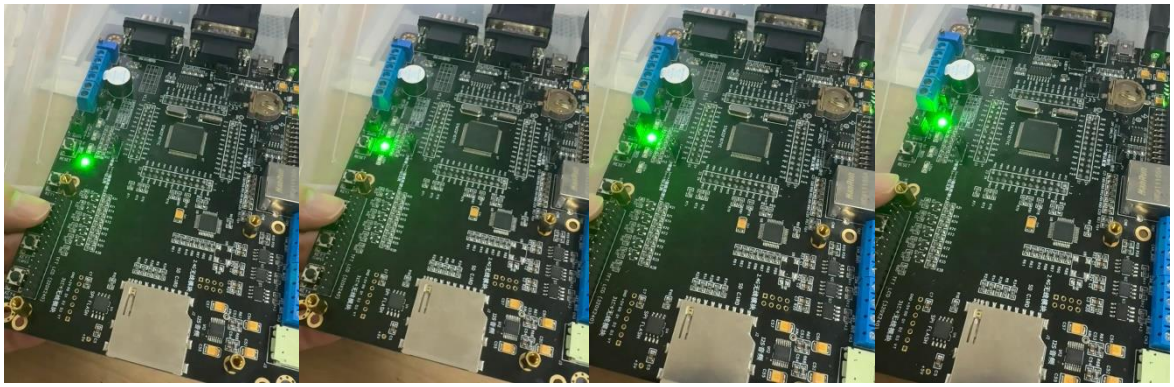
- GPIO의 SetBits와 ResetBits함수를 사용해서, 원하는 LED 핀을 점등

ix) PC와 보드 UART 통신 및 LED 점등 확인

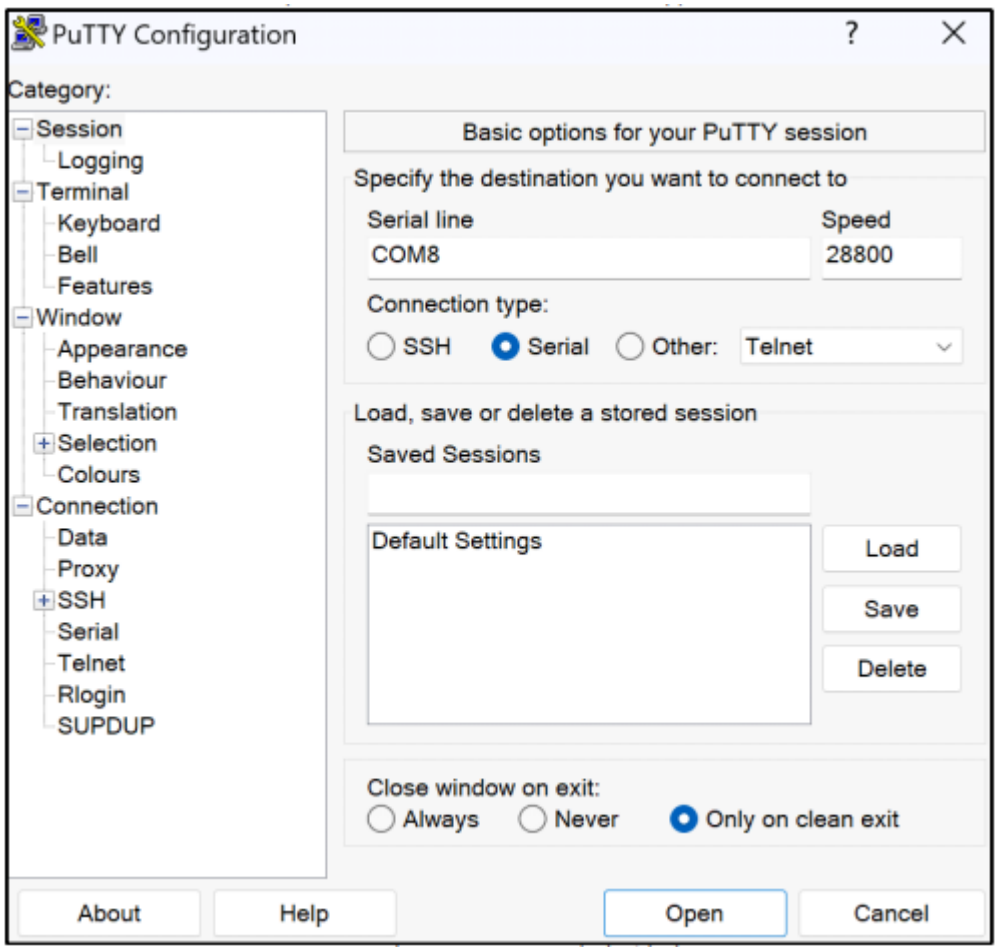
- A 입력 시 점등 순서(1->2->3->4)



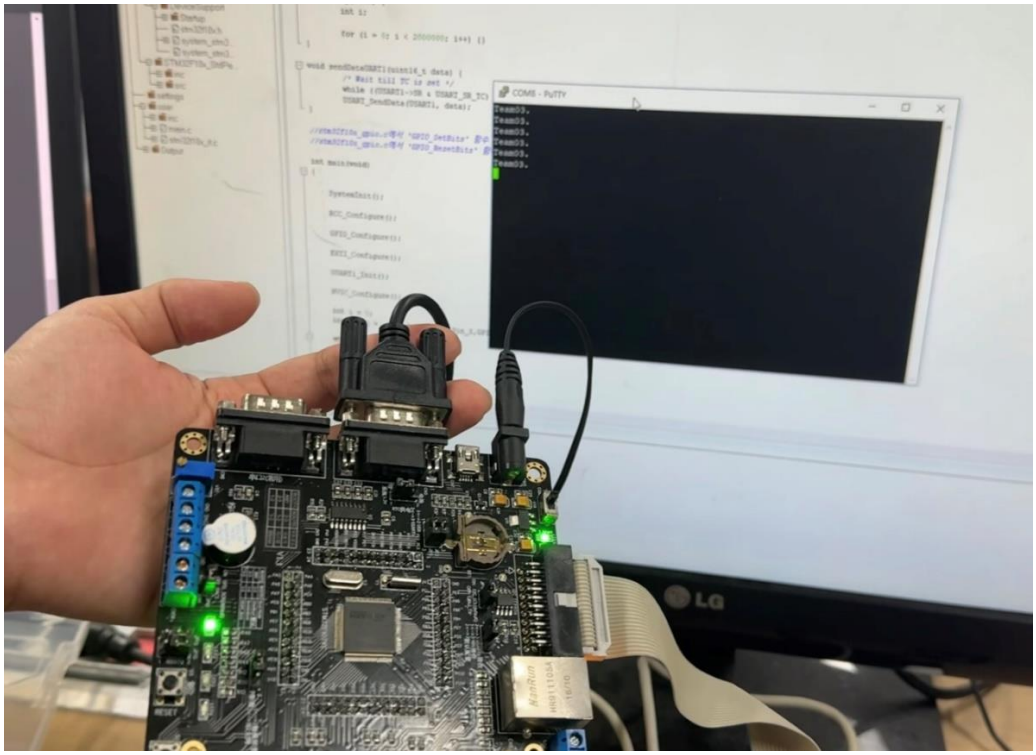
- B 입력 시 점등 순서(4->3->2->1)



- Putty와 UART통신
- Serial line: COM8, Speed: 28800, Connection type: Serial으로 설정 Putty 통신



- Key3버튼 입력 시 Putty창에 Team03.₩r₩n출력



VI. 참고문헌

- stm32_Datasheet.pdf
- stm32_ReferenceManual.pdf
- STM32107VCT6_schematic.pdf
- 9주차 Interrupt GPIO 제어 및 UART 통신.pdf