REPORT

임베디드 실습 및 실험(002) 6주차 실험 결과보고서

10조

전기컴퓨터공학부	201824446	김윤재
정보컴퓨터공학부	202055558	송세연
정보컴퓨터공학부	2020555889	임연후
바이오소재과학과	201845626	최이한

2022.10.7

◆ 실험 목적

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

◆ 실험 목표

1. 보드를 켰을 때 LED 물결 기능 유지 (LED 1->2->3->4->1->2->... 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 함
2. 조이스틱 UP시 A 동작, 조이스틱 DOWN시 B 동작
   - 조이스틱을 눌렀을 때 위 동작이 지연시간 없이 바로 이루어 져야함
3. PC의 Putty에서 'a' 문자 입력 시 A 동작, 'b' 문자 입력 시 B 동작(PC->보드 명령)
4. S1 버튼을 누를 경우 Putty로 "TEAM10.₩r₩n" 출력

◆ 실험 과정

1. stm32f10.h, stm32f10x_exti.h, stm32f10x_gpio.h, stm32f10x_usart.h, stm32f10x_rcc.h 라이브러리 사용 및 함수 선언. 주소를 직접 쓰지 않고 정의된 상수를 사용

```
#include "stm32f10x.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_rcc.h"
#include "misc.h"
#define FORWARD 1
#define REVERSE -1
int flag = FORWARD;
void RCC_Configure(void);
void GPIO_Configure(void);
void EXTI_Configure(void);
void USART1_Init(void);
void NVIC_Configure(void);
void EXTI15_10_IRQHandler(void);
```

```
void Delay(void);
void sendDataUART1(uint16_
```

RCC_APB2PeriphClockCmd 함수를 이용하여 Clock enable 설정

- PORT A Enable(UART TX/ RX)
- PORT C Enable(조이스틱 UP/DOWN)
- PORT B Enable(S1)
- PORT D Enable(LED)
- USART1 Enable
- AFIO(Alternate Function IO) Enable

```
void RCC_Configure(void
{
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
        /* JoyStick Up/Down port clock enable */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE); // port C
        /* JoyStick Selection port clock enable */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); // port B
        /* LED port clock enable */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE); // port D
        /* USART1 clock enable */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE); // USART1
        /* Alternate Function IO clock enable */
}
```

3. GPIO_InitTypeDef 구조체와 GPIO 함수를 이용하여 GPIO 핀 초기화, 설정

```
void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIOC_InitStructure;
    /* JoyStick up, down pin setting */
    GPIOC_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_5; // 2,5번 핀 사용
    GPIOC_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // Default Speed
    GPIOC_InitStructure.GPIO_Mode = (GPIO_Mode_IPD) | (GPIO_Mode_IPU); // input.
    pull down, pull up
    GPIO_Init(GPIOC, &GPIOC_InitStructure); // port C에 적용
   /* JoyStick selection pin setting */

    /* button pin setting */
```

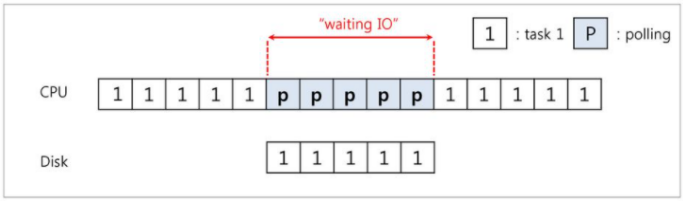
```
    GPIO_InitTypeDef GPIOD_InitStructure;

    GPIOD_InitStructure.GPIO_Pin = GPIO_Pin_11; // 11번핀 사용
    GPIOD_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // Default Speed
    GPIOD_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // input pull up
    GPIO_Init(GPIOD, &GPIOD_InitStructure); // port D에 적용



    /* LED pin setting*/
    GPIO_InitTypeDef GPIOD2_InitStructure;

    GPIOD2_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 |.
    GPIO_Pin_7;   //2,3,4,7번 핀 사용
    GPIOD2_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIOD2_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    //output//
    GPIO_Init(GPIOD, &GPIOD2_InitStructure);

    /* UART pin setting */
    //TX a9
    GPIO_InitTypeDef GPIOA_InitStructure;

    GPIOA_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIOA_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIOA_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIOA_InitStructure);

    //RX a10
    GPIO_InitTypeDef GPIOA2_InitStructure;

    GPIOA2_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIOA2_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIOA2_InitStructure.GPIO_Mode = (GPIO_Mode_IPD) | (GPIO_Mode_IPU);
    GPIO_Init(GPIOA, &GPIOA2_InitStructure);
  }
```

4. GPIO_EXITLineConfig 함수를 이용하여 조이스틱과 버튼의 EXIT 라인 설정, EXTI_InitTypeDef 구조체와 EXTI_Init을 이용하여 EXTI 설정

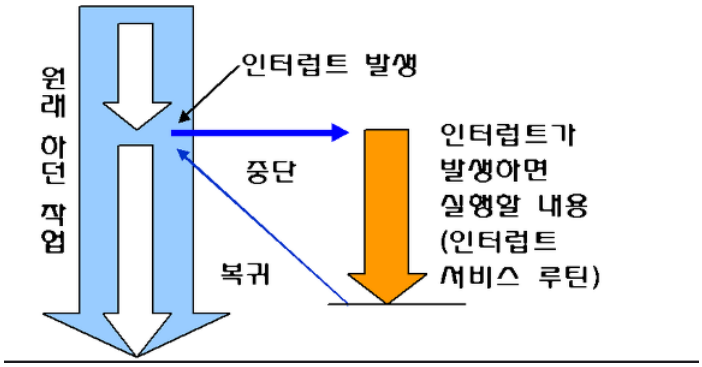
- Mode: Interrupt 모드

-Trigger: Falling 방식

Polling 방식: CPU가 특정 이벤트를 처리하기 위해 이벤트가 발생할 때까지 모든 연산을 이벤트가 발생하는지 감시하는 방식



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



Harware Interrupt	Software Interrupt
- 비동기식 이벤트 처리로 주변장치의 요청에 의해 발생하는 인터럽트 - 높은 우선 순위 - 하드디스크 읽기 요청/디스크 읽기 끝남 / 키보드 입력 등에 발생	- 동기식 이벤트 처리로 사용자가 프로그램 내에서 인터럽트가 발생하도록 설정하는 인터럽트 - 낮은 우선순위 - Trap, Exception 등이 여기에 포함

- EXTI (External Interrupt)
  외부에서 신호가 입력될 경우 Device 에 Event나 Interrupt 가 발생되는 기능
  입력 받을 수 있는 신호는 Rising-Edge, Falling-Edge, Rising & Falling-Edge
  각 Port 의 n번 Pin의 EXTI n 에 연결

- EXTI 는 Event Mode 와 Interrupt Mode 를 선택하여 설정 가능
  Interrupt Mode 로 설정할 경우 Interrupt 가 발생해 해당 Interrupt Handler 가 동작
  20개의 Edge Detector Line 으로 구성되어 각 Line 이 설정에 따라 Rising/Falling Trigger 를 감지

```
void EXTI_Configure(void)
{
      EXTI_InitTypeDef EXTI_InitStructure;
      /* Joystick Down */
     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;
      EXTI_InitStructure.EXTI_LineCmd = ENABLE;
      EXTI_Init(&EXTI_InitStructure);
      /* Joystick Up */
      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);
    /* Joystick Selection */
      /* Button */
      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);
```

5. USART1을 Enable, USART_InitTypeDef 구조체와 USART_Init 함수를 이용하여 USART 설정, USART_ITConfig 함수를 이용하여 USART1 RX를 Enable

```
void USART1_Init(void)
{
    USART_InitTypeDef USART_InitStructure;

    // Enable the USART1 peripheral
    USART_Cmd(USART1, ENABLE);
      USART_InitStructure.USART_BaudRate = 28800; // Baud Rate
      USART_InitStructure.USART_WordLength = USART_WordLength_8b; // word length
```

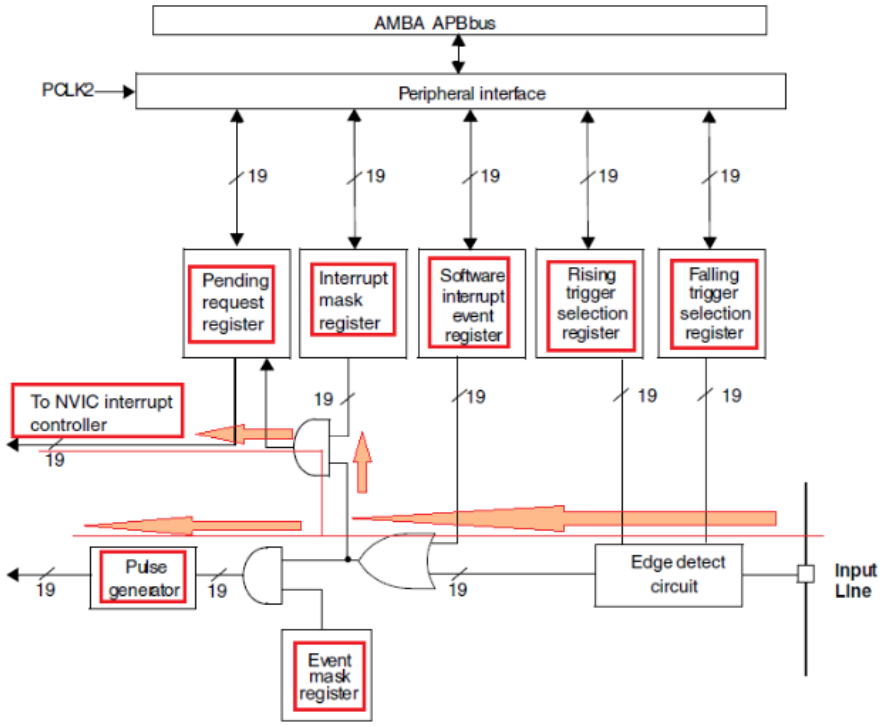
```
8bit
    USART_InitStructure.USART_StopBits = USART_StopBits_1; // stop bit 1bit
    USART_InitStructure.USART_Parity = USART_Parity_No ; // no parity bits
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; // rx&tx
mode
    USART_InitStructure.USART_HardwareFlowControl =.
    USART_HardwareFlowControl_None;
    USART_Init(USART1,&USART_InitStructure);   //USART1에적용
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
}
```

6. NVIC_InitTypeDef 구조체와 NVIC_Init 함수를 이용하여 NVIC 설정, Priority설정

- NVIC (Nested Vectored Interrupt Controller)
  인터럽트 처리 중 또다른 인터럽트 발생시 우선순위를 사용
  우선순위가 높은 인터럽트부터 처리 후 다른 인터럽트 처리
  ARM 보드에서 인터럽트 사용시 NVIC 통하여 우선순위를 결정
  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
```

-
- Interrupt Request 는 Mask Register 를 통해 알 수 있다
  Processor 는 Interrupt 를 인지하여 처리하기 전에 Pending Register
  (어떤 Interrupt 가 발생되었는지 저장) 를 검사하여 발생된 Interrupt 중 Priority가 가장 높은 Interrupt 를 처리
- 외부 Interrupt 는 EXTI0 ~ EXTI15까지 각 Port 의 Pin 번호가 Interrupt Pin과 매치

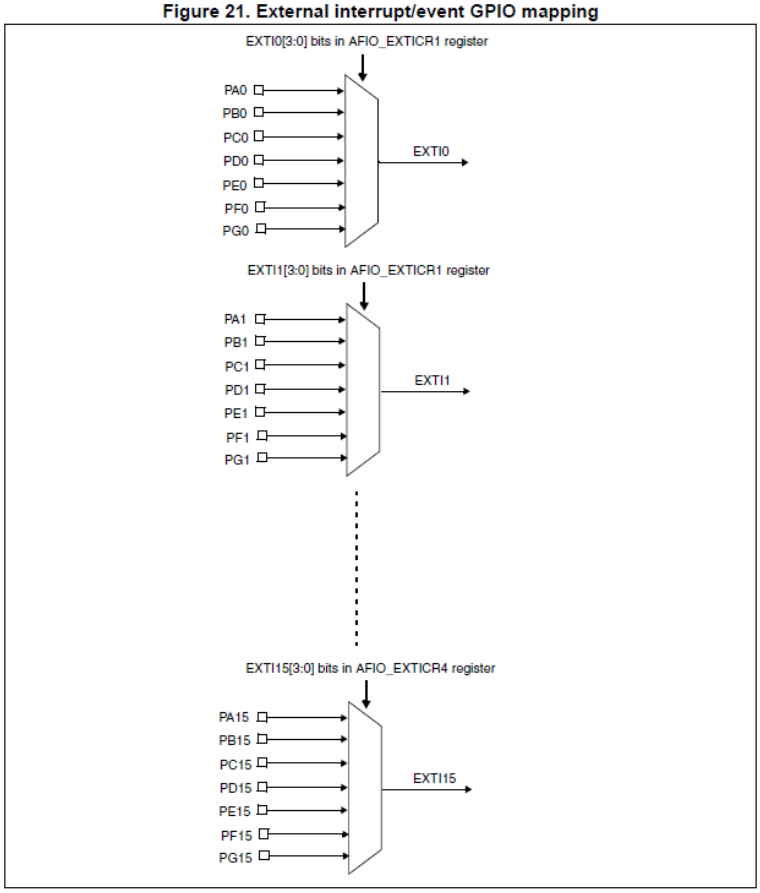


EXTI (External Interrupt)

모든 GPIO 핀들은 EXTI line 을 통해 연결되어 있다

EXTICR1 레지스터를 통해 입력 받을 포트를 선택 하며 같은 번호의 핀들은 같은 라인을 공유

Figure 21. External interrupt/event GPIO mapping



- Pre-emption: 우선순위가 높음, interrupt가 들어오면, 현재 작업을 멈추고 해당 interrupt를 진행(선점)
- Pre-emption priority로 선점 우선순위 결정
- Sub priority로 아직 대기 중인 ISR들의 순서가 결정

```
void NVIC_Configure(void) {


    NVIC_InitTypeDef NVIC_InitStructure;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);


    // Joystick Down

    NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;

    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
```

```
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    // Joystick Up
    NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    // User S1 Button
    NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;
    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 = 1; // TODO
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3; // TODO
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

7. 인터럽트 핸들러 설정

- EXTI를 선언 했을 시에는 반드시 Handler 또한 구현이 필요하다.

(1) "putty에서의 입력" 인터럽트 핸들러

```
void USART1_IRQHandler() {
    uint16_t word;
     if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET){
          word = USART_ReceiveData(USART1);
if (word == 'a') {
               flag = FORWARD;
          } else if (word == 'b') {
               flag = REVERSE;
          }
     USART_ClearITPendingBit(USART1,USART_IT_RXNE);
     }
}
```

(2) "S1버튼" 인터럽트 핸들러

```
void EXTI15_10_IRQHandler(void) {


    if (EXTI_GetITStatus(EXTI_Line11) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_11) == Bit_RESET) {
                sendDataUART1('T');
                sendDataUART1('E');
                sendDataUART1('A');
                sendDataUART1('M');
                sendDataUART1('1');
```

```
                sendDataUART1('0');
                sendDataUART1('₩r');
                sendDataUART1('₩n');
        }
            EXTI_ClearITPendingBit(EXTI_Line11);
    }
}
```

(3)"조이스틱 UP,DOWN" 인터럽트 핸들러

```
void EXTI2_IRQHandler(void) {
    if (EXTI_GetITStatus(EXTI_Line2) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_2) == Bit_RESET) {
                flag = REVERSE;
        }
            EXTI_ClearITPendingBit(EXTI_Line2);
    }
}
void EXTI9_5_IRQHandler(void) {
    if (EXTI_GetITStatus(EXTI_Line5) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5) == Bit_RESET) {
            flag = FORWARD;
        }
            EXTI_ClearITPendingBit(EXTI_Line5);
    }
}
```

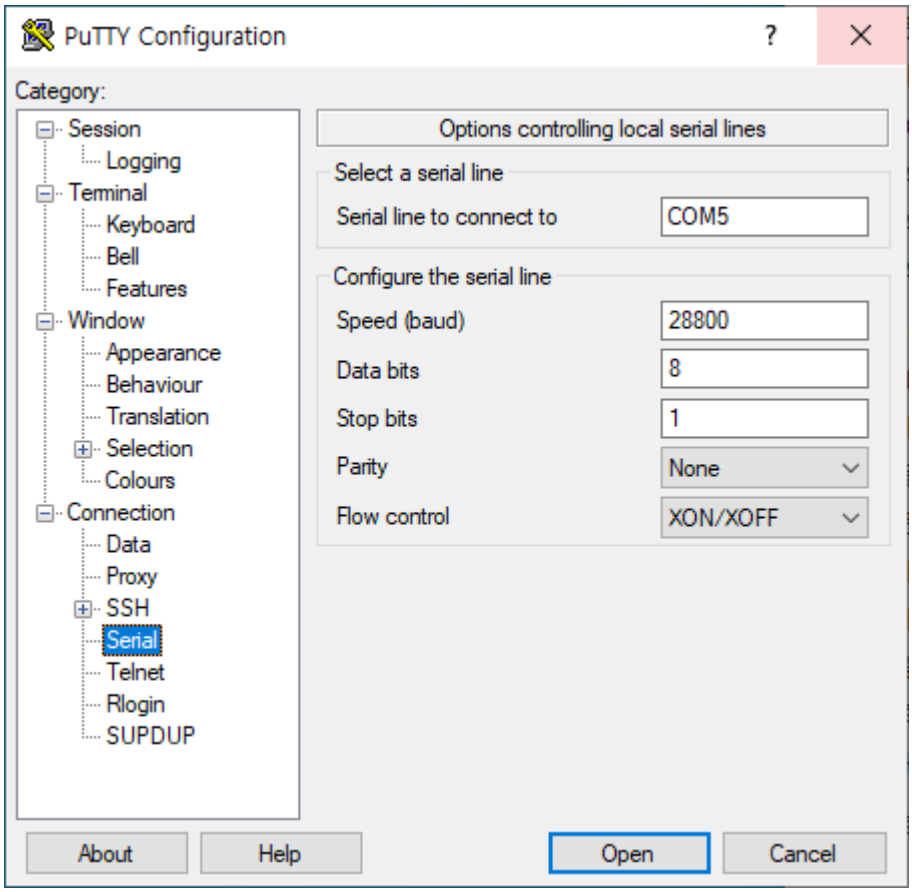
8. main 함수 작성

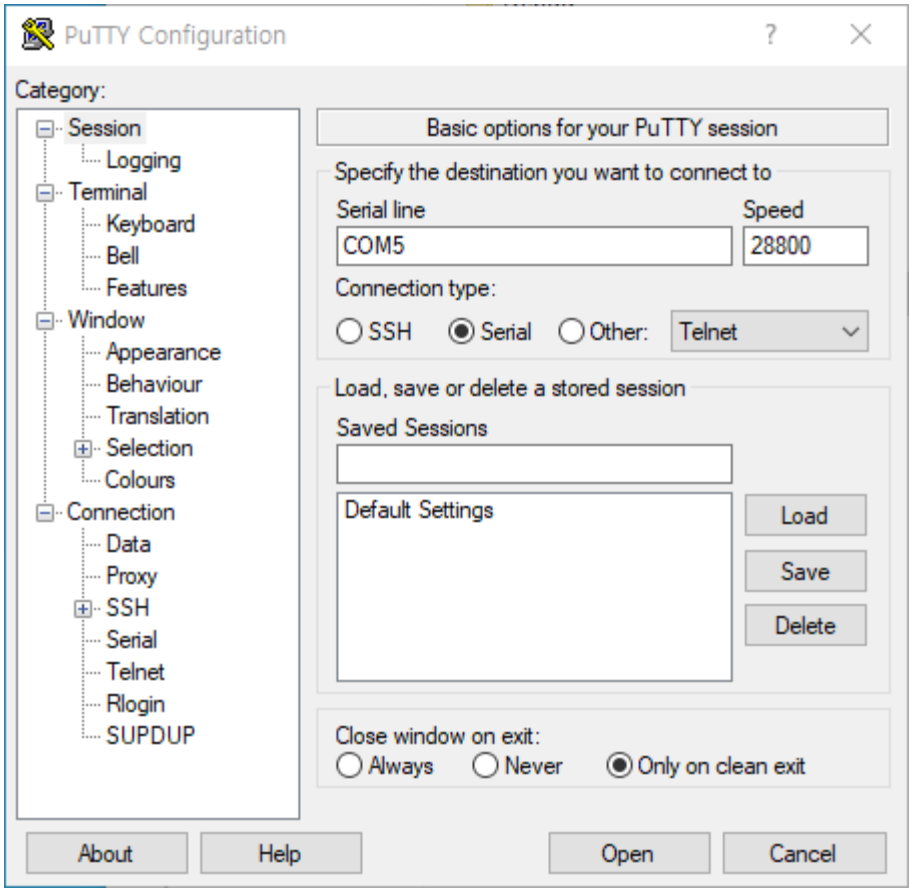
```
int main(void)
{
	SystemInit();
	RCC_Configure();
	GPIO_Configure();
	EXTI_Configure();
	USART1_Init();
	NVIC_Configure();
	int i=0;
	while (1) {
		uint16_t arr[4][4] ={
			{UINT16_MAX, 0 ,0 ,0},
			{0, UINT16_MAX ,0 ,0},
			{0, 0 ,UINT16_MAX ,0},
			{0, 0 ,0 ,UINT16_MAX}};
			GPIOD->ODR = ((GPIO_ODR_ODR2 & arr[i%4][0]) | // port d2
					(GPIO_ODR_ODR3 & arr[i%4][1]) | // port d3
					(GPIO_ODR_ODR4 & arr[i%4][2]) | // port d4
					(GPIO_ODR_ODR7 & arr[i%4][3])); // port d7
		 Delay();
  if( flag ==FORWARD)   i++;
	else i--;

	}
		return 0;
```

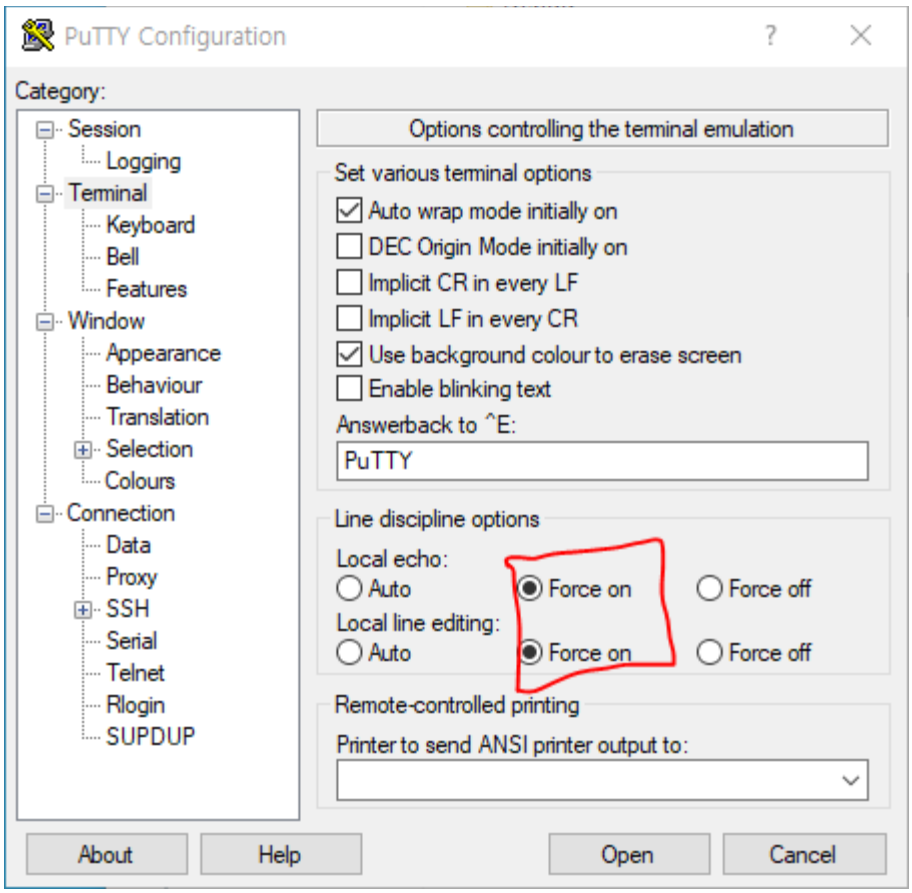
```
}
```

9. putty 설정

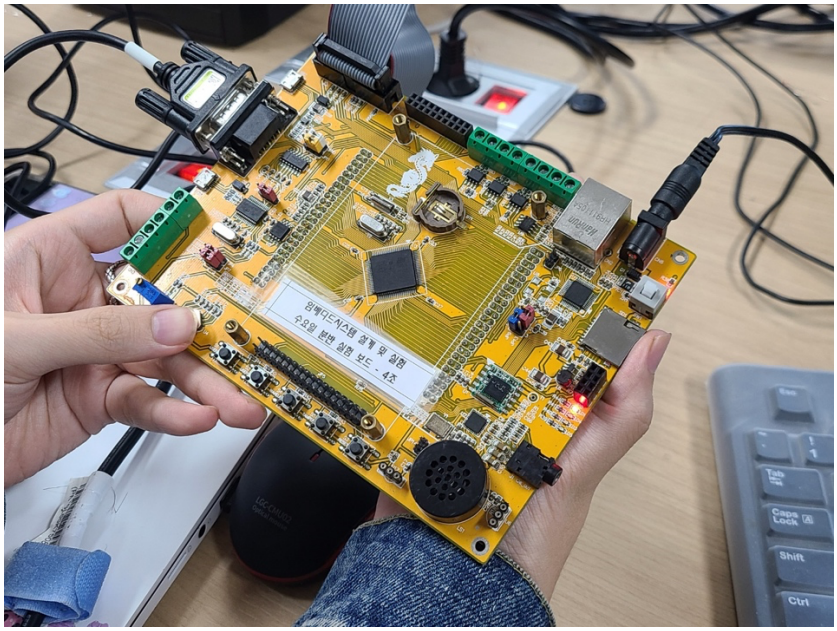
기본적으로 putty에서는 키보드 입력이 되지 않기 때문에 강제적으로 입력가능하게 설정이 필요하다.




PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Serial
Telnet
Rlogin
SUPDUP
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM5
28800
Connection type:
SSH
Serial
Other:
Telnet
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Serial
Telnet
Rlogin
SUPDUP
Options controlling the terminal emulation
Set various terminal options
Auto wrap mode initially on
DEC Origin Mode initially on
Implicit CR in every LF
Implicit LF in every CR
Use background colour to erase screen
Enable blinking text
Answerback to ^E:
PuTTY
Line discipline options
Local echo:
Auto
Force on
Force off
Local line editing:
Auto
Force on
Force off
Remote-controlled printing
Printer to send ANSI printer output to:
About
Help
Open
Cancel

◆ 실험 결과 및 고찰



(1) 보드를 켰을 때 LED 물결 기능이 유지되었다.

(2) 조이스틱 UP, DOWN 조작 시 LED 물결 방향이 딜레이 없이 변경되었다.

(3) S1 버튼을 눌렀을 때 PUTTY 창에 TEAM10 출력이 되었다.

(4) PUTTY창에 'a' 또는 'b' 입력 시 각 입력에 맞게 LED물결 방향이 변경되었다.

- 시리얼 포트를 끼우지 않아서 putty 입력이 되지않았다. 코드에 문제가 있는지 찾느라 시간을 소비했었다.

-putty 설정을 제대로 하지 않아서 키보드입력이 되지않았었다.

- 일정칸에서 조이스틱 조작시 led가 한칸생략되고 방향이 바뀌는 현상이 일어났었다. Priority를 다시 설정해주니 해결되었다.