임베디드 설계 및 실험

실험 결과 보고서

8 주차 3 조

실험 내용

1. Configure

- RCC, GPIO, USART, NVIC

2. 함수 정의

- Delay, LED2, LED1, turnoff, SendData, printTeam, printUP, printDN

3. Handler

- USART1, EXTI3, EXTI4, EXTI9, EXTI15

4. Main()

설정 (Configuration)

- RCC 설정

```
void RCC_Configure() {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */
     RCC_APB2PeriphClockCmd( // 사용하고자하는 port들의 clock을 enable 한다.
     RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOD
           | RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC, ENABLE);
}
```

Port a, b, c, d, usart1 을 enable 한다.

- GPIO 설정

Input, output pin/port 설정, speed 설정

```
void GPIO_Configure() {

        GPIO_InitTypeDef GPIO_InitStructure;

        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7)
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        /*TODO: USART1, JoyStick Config */
        // USART TX
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        // USART RX
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA, &GPIO_InitStructure);
        GPIO_InitTypeDef gpB, gpC, gpD, gpD_btn;

        // 버튼 PD11
        gpD_btn.GPIO_Mode = GPIO_Mode_IPD;
        gpD_btn.GPIO_Pin = GPIO_Pin_11;
        gpD_btn.GPIO_Speed = GPIO_Speed_50MHz;

        // 조이스틱 셀렉트
        gpB.GPIO_Mode = GPIO_Mode_IPD;
        gpB.GPIO_Pin = GPIO_Pin_8;
        gpB.GPIO_Speed = GPIO_Speed_50MHz;

        // 조이스틱 right, left
        gpC.GPIO_Mode = GPIO_Mode_IPD;
        gpC.GPIO_Pin = (GPIO_Pin_3 | GPIO_Pin_4);
        gpC.GPIO_Speed = GPIO_Speed_50MHz;
```

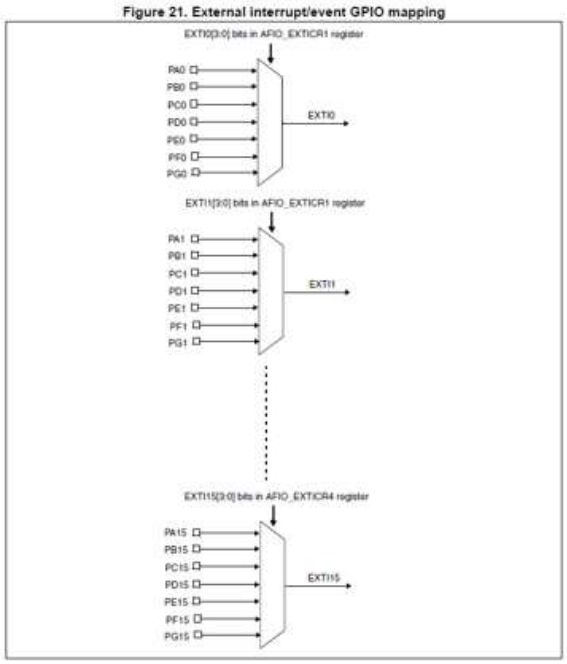
```
        GPIO_Init(GPIOD, &gpD_btn); // PD11 configuration
        GPIO_Init(GPIOB, &gpB); // 조이스틱 SELECT configuration
        GPIO_Init(GPIOC, &gpC); // 조이스틱 Right, Left configuration

        // init
        /*TODO: GPIO EXTILineConfig*/
        EXTI_InitTypeDef exti, exti_btn;
        //usarthandler랑 confighandler랑 동시에 들어가서 충돌일어남
        //GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource10);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource3);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource4);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource8);
        // 인터럽트와 핀을 연결. 해당 핀을 통해 인터럽트
}
```

- USART 설정

```
void USART_Configure() {
        USART_InitTypeDef usart;
        /*TODO: USART1 configuration*/
        usart.USART_BaudRate = 115200;
        usart.USART_WordLength = USART_WordLength_8b;
        usart.USART_Mode = USART_Mode_Tx | USART_Mode_Rx ;
        usart.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        usart.USART_Parity = USART_Parity_No;
        usart.USART_StopBits = USART_StopBits_1;
        USART_Init(USART1, &usart);
        /*TODO: USART1 cmd ENABLE*/
        USART_Cmd(USART1, ENABLE);
        /*TODO: USART1 IT Config*/
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
        //USART 구조체에 설정한 값을
        //실제 MCU 레지스터에 설정하는 부분
}
```

- EXTI 설정

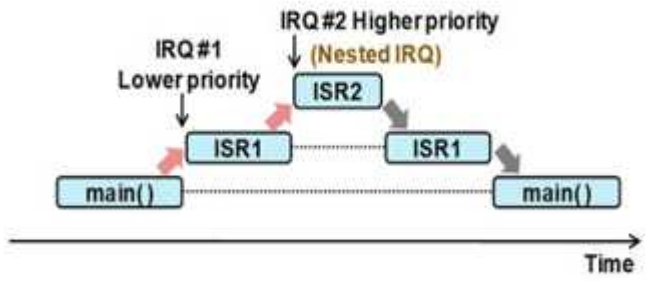


외부에서 신호가 입력될 경우 Device 에 Interrupt 또는 Event 를 발생한다. Interrupt 는 CPU 가 ISR 핸들러를 처리하게 하고 Event 는 pulse 를 발생시켜 특정 기능을 하게 한다. 모든 GPIO 핀들은 EXTI line 을 통해 연결되고 같은 번호의 핀들은 같은 라인을 공유한다.

```
void EXTI_Configure() {
        /*TODO: EXTI configuration [ mode interrupt ] [Trigger_falling] */
        EXTI_InitTypeDef exti, exti_btn;
        exti_btn.EXTI_LineCmd = ENABLE;
        exti_btn.EXTI_Mode = EXTI_Mode_Interrupt;
        exti_btn.EXTI_Trigger = EXTI_Trigger_Falling;
        exti_btn.EXTI_Line = (EXTI_Line3 | EXTI_Line4 | EXTI_Line11 |EXTI_Line8);
        EXTI_Init(&exti_btn);
        //3,4,11,8 버튼에 해당하는 인터럽트를 사용
}
```

- NVIC 설정

인터럽트 처리 중 다른 인터럽트 발생 시 우선순위를 관리한다. 우선순위가 높은 인터럽트부터 처리 후 다른 인터럽트를 처리한다. 4 bits 로 preemption/sub priority 설정한다. 작은 값일수록 높은 우선 순위를 가진다. 이번에는 우선순위가 0 으로 다 같았다.



```
void NVIC_Configure() {
        /*TODO: NVIC_configuration */
        NVIC_InitTypeDef nvic, nvic_btn, nvic_sel, nvic_left,nvic_right;

        //각 port의 n번 핀은 EXTI n에 연결된다
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
        nvic.NVIC_IRQChannel = USART1_IRQn;
        nvic.NVIC_IRQChannelCmd = ENABLE;
        nvic.NVIC_IRQChannelPreemptionPriority = 0;
        nvic.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic);
        //우선순위 설정 및 USART 인터럽트를 사용하기 위해 설정

        // 각 버튼 및 조이스틱 인터럽트를 사용하기 위해 설정
        nvic_btn.NVIC_IRQChannel = EXTI15_10_IRQn;
        nvic_btn.NVIC_IRQChannelCmd = ENABLE;
        nvic_btn.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_btn.NVIC_IRQChannelSubPriority =0;
        NVIC_Init(&nvic_btn);

        nvic_sel.NVIC_IRQChannel = EXTI9_5_IRQn;
        nvic_sel.NVIC_IRQChannelCmd = ENABLE;
        nvic_sel.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_sel.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_sel);

        nvic_left.NVIC_IRQChannel = EXTI3_IRQn;
        nvic_left.NVIC_IRQChannelCmd = ENABLE;
        nvic_left.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_left.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_left);

        nvic_right.NVIC_IRQChannel = EXTI4_IRQn;
        nvic_right.NVIC_IRQChannelCmd = ENABLE;
        nvic_right.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_right.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_right);
```

- Interrupt Handler

각 인터럽트 핸들러에서 호출되는 함수의 프로토타입이 정의되어 있으므로 정의된 함수명을 그대로 사용해야 한다. 아래처럼 라이브러리에 있는 함수명으로 그대로 써야한다.

```
void EXTI3_IRQHandler(void) { //left 누르면 led2()
  if (EXTI_GetFlagStatus(EXTI_Line3) != RESET) {
      flag = 2;
      EXTI_ClearITPendingBit(EXTI_Line3);
  }
}
void EXTI4_IRQHandler(void) { //right 누르면  led1()
  if (EXTI_GetFlagStatus(EXTI_Line4) != RESET) {
      flag = 1;
      EXTI_ClearITPendingBit(EXTI_Line4);
  }
}
```

핸들러 함수 마지막에 해당 인터럽트 pending bit 를 반드시 clear 해야한다.

인터럽트 핸들러는 최대한 빠른 시간에 끝내고 메인 task 로 돌아가야 하므로 시간이 걸리는 동작 및 delay 등은 전역변수를 이용하여 메인 task 에서 처리하도록 하는 것이 좋다. Flag 를 쓰는 이유다.

Putty 를 통해 데이터 출력

9.2.3 Port input data register (GPIOx_IDR) (x=A..G)

Address offset: 0x08h

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data (y= 0 .. 15)
These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

- PortD 11 누를 때 team03 전송
- Putty 로 up, dn 입력받아 보드로 전송 후 함수 동작

실행 결과

Figure 1 - Right 눌렀을 때 LED1

Figure 2 - Left 눌렀을 때 LED2

Figure 3 - user 버튼 1 눌렀을 때 TEAM03 출력

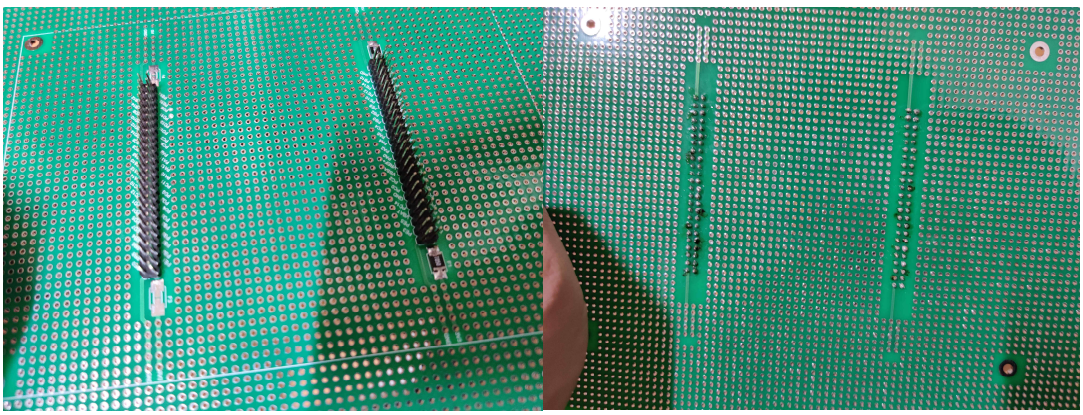


Figure 4 – 납땜 결과물

전체 코드 11

```c
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "misc.h"
#include "core_cm3.h"

int flag = 0;
int flag2 = 0;
char input[2];
void RCC_Configure() {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */
     RCC_APB2PeriphClockCmd( // 사용하고자하는 port들의 clock을 enable 한다.
     RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOD
           | RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC, ENABLE);
}
void GPIO_Configure() {

        GPIO_InitTypeDef GPIO_InitStructure;

        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
        GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7);
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        /*TODO: USART1, JoyStick Config */
        // USART TX
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        // USART RX
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA, &GPIO_InitStructure);
        GPIO_InitTypeDef gpB, gpC, gpD, gpD_btn;

        // 버튼 PD11
        gpD_btn.GPIO_Mode = GPIO_Mode_IPD;
        gpD_btn.GPIO_Pin = GPIO_Pin_11;
        gpD_btn.GPIO_Speed = GPIO_Speed_50MHz;

        // 조이스틱 셀렉트
        gpB.GPIO_Mode = GPIO_Mode_IPD;
        gpB.GPIO_Pin = GPIO_Pin_8;
        gpB.GPIO_Speed = GPIO_Speed_50MHz;

        // 조이스틱 right, left
        gpC.GPIO_Mode = GPIO_Mode_IPD;
        gpC.GPIO_Pin = (GPIO_Pin_3 | GPIO_Pin_4);
        gpC.GPIO_Speed = GPIO_Speed_50MHz;

        GPIO_Init(GPIOD, &gpD_btn); // PD11 configuration
        GPIO_Init(GPIOB, &gpB); // 조이스틱 SELECT configuration
        GPIO_Init(GPIOC, &gpC); // 조이스틱 Right, Left configuration
```

```
        // init
        /*TODO: GPIO EXTILineConfig*/
        EXTI_InitTypeDef exti, exti_btn;
        //usarthandler랑 confighandler랑 동시에 들어가서 충돌일어남
        //GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource10);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource3);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource4);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource8);
        // 인터럽트와 핀을 연결. 해당 핀을 통해 인터럽트
}

void USART_Configure() {
        USART_InitTypeDef usart;
        /*TODO: USART1 configuration*/
        usart.USART_BaudRate = 115200;
        usart.USART_WordLength = USART_WordLength_8b;
        usart.USART_Mode = USART_Mode_Tx | USART_Mode_Rx ;
        usart.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
        usart.USART_Parity = USART_Parity_No;
        usart.USART_StopBits = USART_StopBits_1;
        USART_Init(USART1, &usart);
        /*TODO: USART1 cmd ENABLE*/
        USART_Cmd(USART1, ENABLE);
        /*TODO: USART1 IT Config*/
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
        //USART 구조체에 설정한 값을
        //실제 MCU 레지스터에 설정하는 부분
}
```

```
void EXTI_Configure() {
        /*TODO: EXTI configuration [ mode interrupt ] [Trigger_falling] */
        EXTI_InitTypeDef exti, exti_btn;
        exti_btn.EXTI_LineCmd = ENABLE;
        exti_btn.EXTI_Mode = EXTI_Mode_Interrupt;
        exti_btn.EXTI_Trigger = EXTI_Trigger_Falling;
        exti_btn.EXTI_Line = (EXTI_Line3 | EXTI_Line4 | EXTI_Line11 |EXTI_Line8);
        EXTI_Init(&exti_btn);
        //3,4,11,8 버튼에 해당하는 인터럽트를 사용
}

void NVIC_Configure() {
        /*TODO: NVIC_configuration */
        NVIC_InitTypeDef nvic, nvic_btn, nvic_sel, nvic_left,nvic_right;

       //각 port의 n번 핀은 EXTI n에 연결된다
       NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
       nvic.NVIC_IRQChannel = USART1_IRQn;
       nvic.NVIC_IRQChannelCmd = ENABLE;
       nvic.NVIC_IRQChannelPreemptionPriority = 0;
       nvic.NVIC_IRQChannelSubPriority = 0;
       NVIC_Init(&nvic);
       //우선순위 설정 및 USART 인터럽트를 사용하기 위해 설정

       // 각 버튼 및 조이스틱 인터럽트를 사용하기 위해 설정
       nvic_btn.NVIC_IRQChannel = EXTI15_10_IRQn;
       nvic_btn.NVIC_IRQChannelCmd = ENABLE;
       nvic_btn.NVIC_IRQChannelPreemptionPriority = 0;
       nvic_btn.NVIC_IRQChannelSubPriority =0;
       NVIC_Init(&nvic_btn);

       nvic_sel.NVIC_IRQChannel = EXTI9_5_IRQn;
       nvic_sel.NVIC_IRQChannelCmd = ENABLE;
       nvic_sel.NVIC_IRQChannelPreemptionPriority = 0;
       nvic_sel.NVIC_IRQChannelSubPriority = 0;
```

```
        nvic_sel.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_sel);

        nvic_left.NVIC_IRQChannel = EXTI3_IRQn;
        nvic_left.NVIC_IRQChannelCmd = ENABLE;
        nvic_left.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_left.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_left);

        nvic_right.NVIC_IRQChannel = EXTI4_IRQn;
        nvic_right.NVIC_IRQChannelCmd = ENABLE;
        nvic_right.NVIC_IRQChannelPreemptionPriority = 0;
        nvic_right.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&nvic_right);
}

void delay(int a){
  for(int i = 0; i < a; i++);
}

void LED2(){
  GPIO_SetBits(GPIOD, GPIO_Pin_3);
  delay(500000);
  GPIO_ResetBits(GPIOD, GPIO_Pin_3);
  delay(500000);
}
void LED1(){
  GPIO_SetBits(GPIOD, GPIO_Pin_2);
  delay(500000);
  GPIO_ResetBits(GPIOD, GPIO_Pin_2);
  delay(500000);
}
void turnOff(){
  GPIO_ResetBits(GPIOD, GPIO_Pin_3);
  GPIO_ResetBits(GPIOD, GPIO_Pin_2);
}
void SendData(uint16_t data) {
```

```c
void SendData(uint16_t data) {
    /* Transmit Data */
   USART1->DR = data;

   /* Wait till TC is set */
   while ((USART1->SR & USART_SR_TC) == 0);
}

void printTeam(){
  SendData('T');
  SendData('E');
  SendData('A');
  SendData('M');
  SendData('0');
  SendData('3');
  SendData('\r');
  SendData('\n');
}

void printUP(){
       GPIO_SetBits(GPIOD, GPIO_Pin_7);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      GPIO_SetBits(GPIOD, GPIO_Pin_4);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_4);
      GPIO_SetBits(GPIOD, GPIO_Pin_3);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_3);
      GPIO_SetBits(GPIOD, GPIO_Pin_2);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_2);
}

void printDN(){
      GPIO_SetBits(GPIOD, GPIO_Pin_2);
      delay(500000);
```

```c
void printDN(){
      GPIO_SetBits(GPIOD, GPIO_Pin_2);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_2);
      GPIO_SetBits(GPIOD, GPIO_Pin_3);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_3);
      GPIO_SetBits(GPIOD, GPIO_Pin_4);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_4);
      GPIO_SetBits(GPIOD, GPIO_Pin_7);
      delay(500000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_7);
}
/*TODO: IRQHandler */
void USART1_IRQHandler(void) {
  if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET){
    char get;
    get = USART_ReceiveData(USART1);
    if(get == 'u')
      input[0]=get;
    if(get == 'p' && input[0] == 'u')
      input[1]=get;
    if(get == 'd')
      input[0]=get;
    if(get == 'n' && input[0] == 'd')
      input[1]=get;
    if(input[0] == 'u' && input[1] == 'p')
      flag = 3;
    if(input[0] == 'd' && input[1] == 'n')
      flag = 4;
  }
}
void EXTI3_IRQHandler(void) { //left 누르면 led2()
  if (EXTI_GetFlagStatus(EXTI_Line3) != RESET) {
      flag = 2;
```

```c
void EXTI4_IRQHandler(void) { //right 누르면  led1()
  if (EXTI_GetFlagStatus(EXTI_Line4) != RESET) {
      flag = 1;
      EXTI_ClearITPendingBit(EXTI_Line4);
  }
}
void EXTI9_5_IRQHandler(void) { //select 누르면 turnoff()
   if (EXTI_GetFlagStatus(EXTI_Line8) != RESET) {
      flag = 0;
      EXTI_ClearITPendingBit(EXTI_Line8);
  }
}
void EXTI15_10_IRQHandler(void)  { //PD11 누르면 team03 출력
  if(EXTI_GetFlagStatus(EXTI_Line11)  != RESET){
    flag2 = 11;
    EXTI_ClearITPendingBit(EXTI_Line11);
  }
}

int main() {
  SystemInit();
  RCC_Configure();
  GPIO_Configure();
  USART_Configure();
  EXTI_Configure();
  NVIC_Configure();

  SendData('\0');

  while(1){
    if(flag == 1)
      LED1();
    if(flag == 2)
      LED2();
    if(flag == 0)
      turnOff();
    if(flag == 3)
      printUP();
    if(flag == 4)
      printDN();
    if(flag2 == 11){
      printTeam();
      flag2=0;
    }
  };
```