

REPORT



8주차 결과 보고서

수강과목: 임베디드 시스템 설계 및 설계

담당교수: 백윤주 교수님

학 과: 전기컴퓨터공학부 정보컴퓨터공학전공

조 원: 201724651 김장환 201724648 김경호
201624481 박윤형 201524588 조창흠

제출일자: 2021. 11. 02

1. 실험 목표

1. 납땜 방법
 - 납땜을 어떻게 하는지, 납땜을 위해 사용되는 기구 사용법 배우기
2. 납땜 후 연결 확인
 - 납땜 후 도중 끊긴 부분은 없는지 멀티미터로 확인
3. 납땜한 보드를 이용한 UART 통신
 - Putty를 통해 보드와 PC 연결
4. 보드와 휴대폰 간 블루투스 연결
 - 스마트폰의 "Serial Bluetooth Terminal" 어플을 이용하여 보드와의 통신

2. 배경 지식

(1) 블루투스

블루투스 무선 시스템은 ISM (Industrial Scientific and Medical) 주파수 대역인 2400~2488.5 MHz를 사용한다. 그 중 그 외의 범위 주파수를 사용하는 다른 시스템과의 간섭을 막기 위해 2402 ~ 2480 MHz, 총 79개의 채널을 사용한다.

여러 시스템과 같은 주파수 대역을 사용하므로 시스템 간에 전파의 간섭이 생길 수 있는데, 이를 예방하기 위해서 블루투스는 주파수 호핑 (Frequency Hopping) 방식을 사용한다. 이는 다수의 채널을 특정 패턴에 따라 빠르게 이동하면서 패킷을 조금씩 전송하는 방법이다. 이 호핑 패턴이 블루투스 기기 간에 동기화가 되어야 통신이 이루어진다.

블루투스는 기기 간 마스터(Master), 슬레이브(Slave) 관계로 연결되는데, 마스터 기기가 생성하는 주파수 호핑에 슬레이브 기기를 동기화하지 못하면 두 기기 간에 통신이 이루어지지 않는다. 이

것으로 다른 시스템의 전파 간섭을 피해 안정적인 연결을 가질 수 있게 된다. 이때 하나의 마스터 기기에는 최대 7개의 슬레이브 기기를 연결할 수 있고, 마스터 기기와 슬레이브 기기 간 통신만 가능하며, 슬레이브 기기 간의 통신은 불가능하다. 그러나 마스터, 슬레이브 역할은 상황에 따라 바뀌어서 사용할 수 있다.

1) SSID (Service Set Identifier)

SSID는 Wi-Fi 네트워크 이름으로, 주변 장치에서 무선 라우터를 식별할 수 있도록 무선 라우터가 broadcast하는 이름이다.

2) UUID (Universally Unique Identifiers)

범용 고유 번호라고도 하며 128비트의 숫자들을 조합한다.

범용적으로 사용할 수 있는 고유의 ID를 사용하기 위해 생성되며, 그를 위해선 Hex 조합은 Unique해야 한다. 블루투스에서는 device에서 제공하는 서비스를 검색하여 각 서비스마다 UUID를 부여하는 등 많은 부분에서 사용된다.

(2) AT 명령어 (AT Commands)

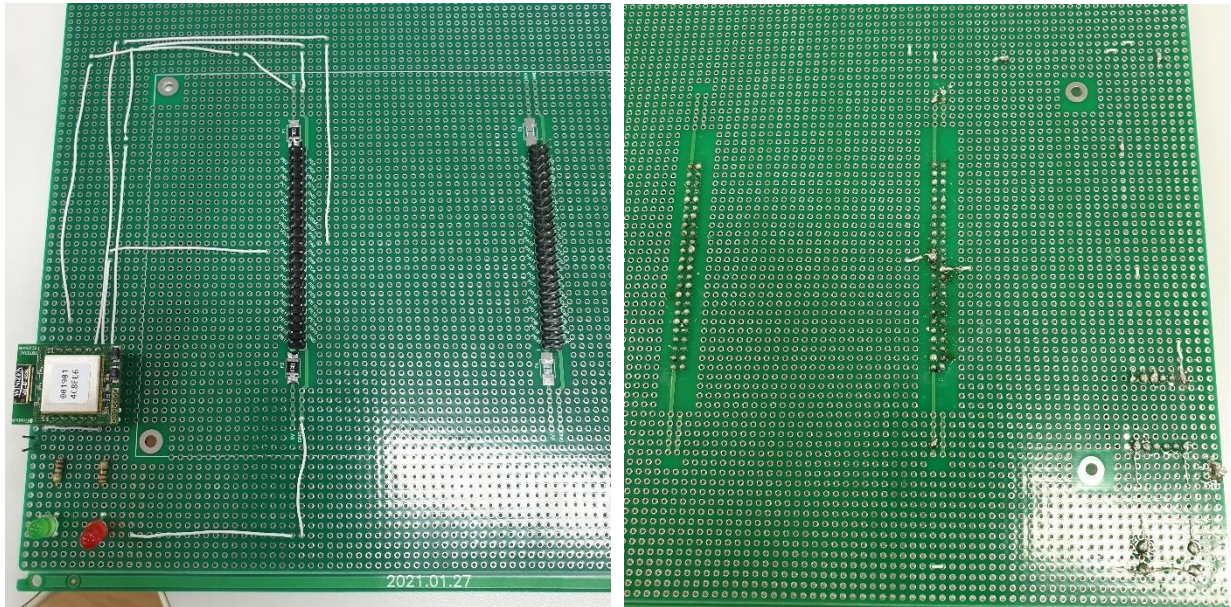
모뎀이 연결된 컴퓨터가 있을 때 컴퓨터는 모뎀에 데이터를 보내면 모뎀은 그 데이터의 신호를 변조하여 외부로 전송한다. 이 때 컴퓨터가 모뎀 자체에 명령을 주어 제어하고 싶을 때는 컴퓨터와 모뎀 사이에 데이터 선 이외의 제어선을 연결해야 한다. 하지만 AT 명령어를 사용하면 데이터 선 하나만으로 데이터 전송과 모뎀 제어 두가지를 할 수 있다.

AT 명령어 기법에서 모뎀은 Data Mode와 Command Mode의 2가지 모드로 변환될 수 있다. Data Mode에서는 컴퓨터에서 모뎀으로 입력된 데이터가 변조되어 그대로 전송되고, Command Mode에서는 컴퓨터에서 모뎀으로 입력된 데이터가 모뎀 자체 제어 명령으로 인식되어 입력된 데이터가 모뎀 제어에만 사용되고 외부로는 전송되지 않는다.

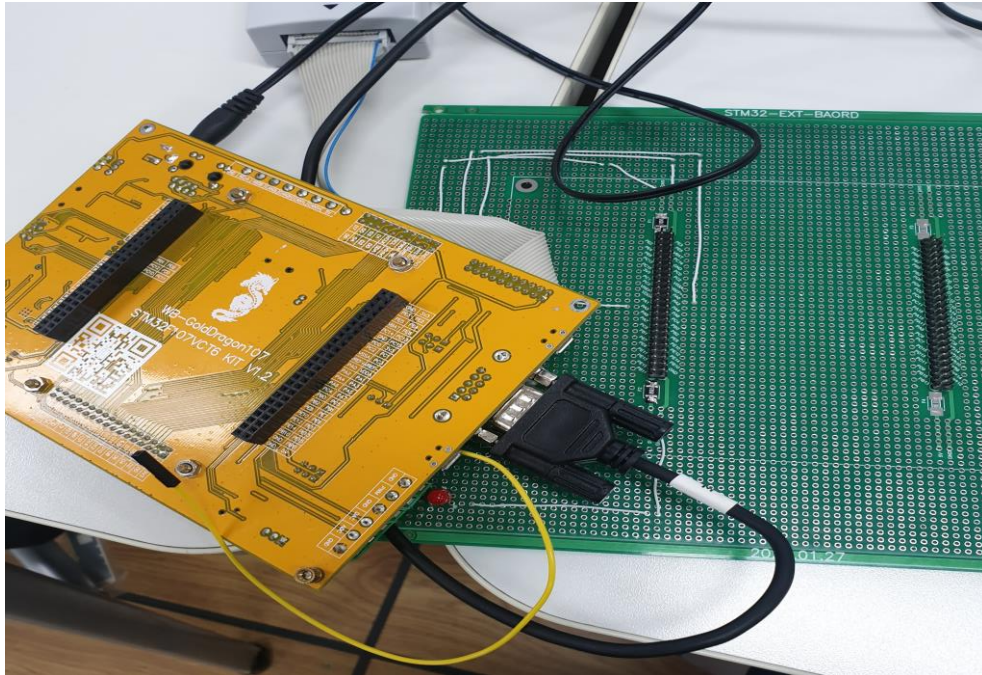
3. 실험 내용

주어진 실험 방법에 따라 다음과 같은 순서로 진행하면 된다.

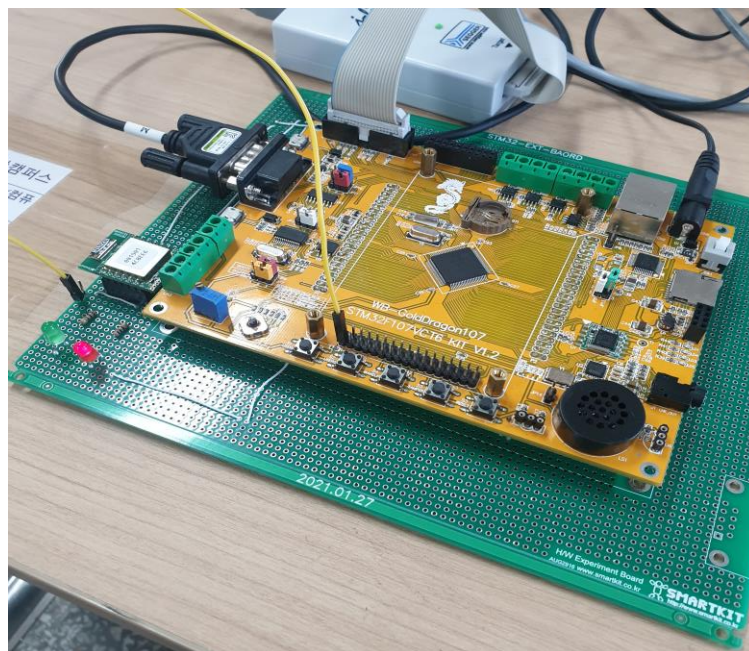
- 각각 레지스터에 RCC를 이용하여 Clock을 인가해준다.
- 각각의 포트에 맞는 모드를 Configuration 한다.
- USART통신을 위한 USART Configuration 한다.
- NVIC를 설정하고, 각 인터럽트에 맞게 handler 함수를 만든다.



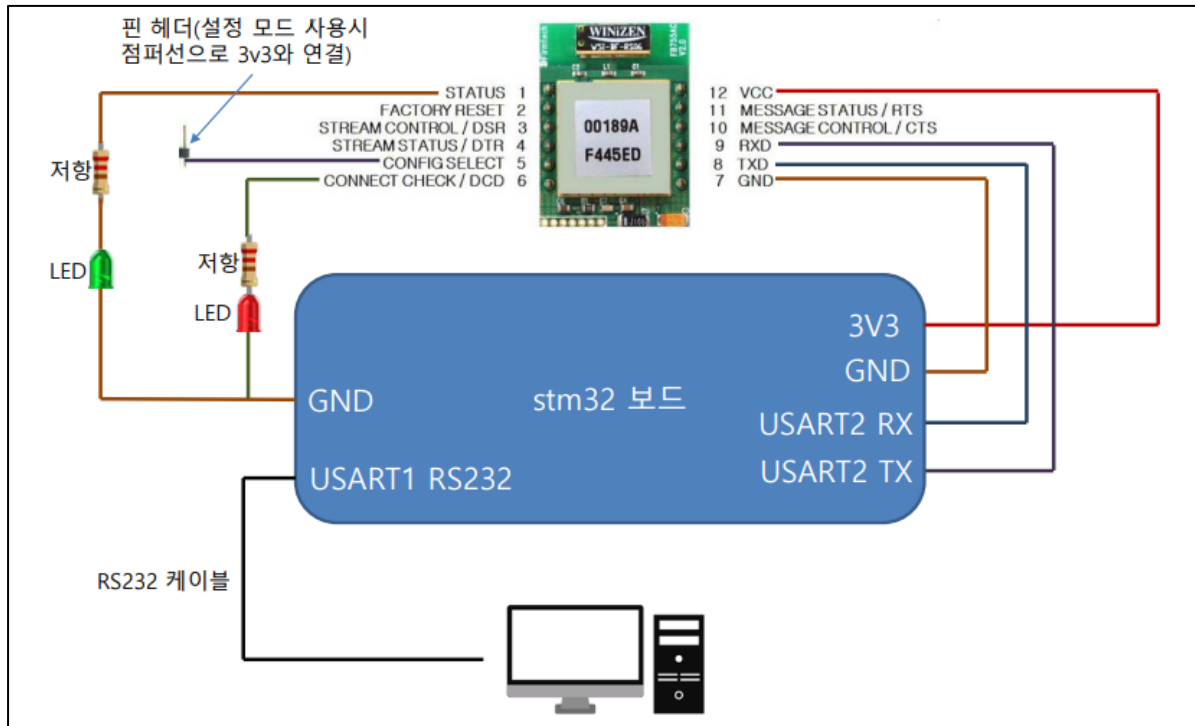
(납땀한 보드 앞뒤 사진)



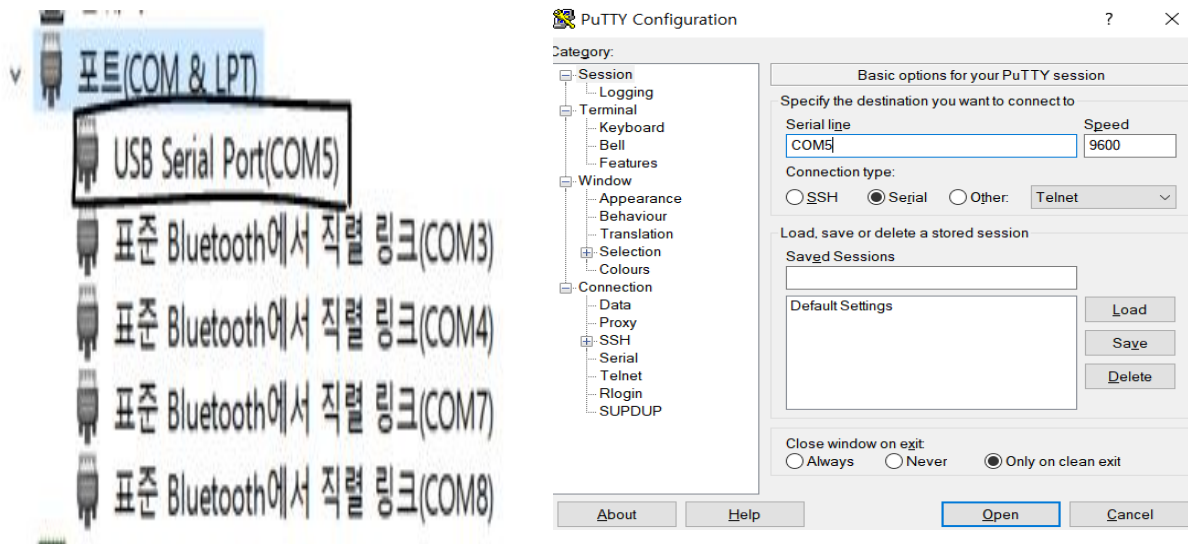
위의 사진을 잘 보면 STM32와 납땀한 보드 간의 유사한 점을 볼 수 있는데, 검은 단자가 딱 들어맞는다. 서로 일치하게 아래의 사진과 같이 블루투스 모듈과 함께 결합시켜준다.



위의 사진을 보면 노란색 점프선이 서로 연결되어 있는 것을 볼 수 있는데, 이는 아래의 사진에서 핀 헤더 부분을 보면 되는데, 블루투스 모듈에 3v3의 전압을 주는 부분이다.



그러면 이제, 작성한 main.c 코드를 모듈에 download 해주자. 그러면, 적색, 녹색 LED 2개가 둘다 불이 켜지고, 블루투스 모듈의 STATUS에 연결된 녹색 LED는 점등하는 것을 확인할 수 있다. 이것은 Putty와의 블루투스 통신이 가능한 상태임을 나타낸다.



그러면, 장치관리자를 통해 PC와 연결된 포트 번호 (COM5)를 확인하고, 오른쪽과 같이 Putty Configuration을 설정해주고 Putty 창을 연다. 그러면 처음에는 아무것도 안 뜬 흑백창이 뜨는데, 거기서 STM32 모듈을 켜다 켜주면 아래의 사진과 같은 설정 모드 창이 뜬다.

```

|=====|
|           Model name : FB755           |
|           Version    : 1.2.6           |
|=====|

===== TOP MENU =====
0 => DEVICE NAME           : TEAM_08
1 => AUTHENTICATION        : DISABLE PINCODE[0000]
2 => REMOTE BD ADDRESS     : ACF6F70D9827
   LOCAL BD ADDRESS       : 0019014C8EE6
3 => CONNECTION MODE       : CNT_MODE4
4 => OTHER PARAMETER       : E,D,5,2B,2,D
5 => UART CONFIG           : 9600,8,n,1
6 => ROLE                   : SLAVE
7 => OPERATION MODE        : OP_MODE0
=====
[ Back Spcae : Input data Cancel      ]
[ t : Move top menu                    ]
=====
Select(0 ~ 7) > 

```

이 창은 블루투스 Device name과 Pincode(비밀번호), Connection mode, Uart config 등 설정을 할 수 있다. 우선, 0번을 눌러 Device Name을 TEAM_08로 바꿔주자.

```

Select(0 ~ 7) > 0
Change Device name
Within 12 character > TEAM_08
|=====|
|           Model name : FB755           |
|           Version    : 1.2.6           |
|=====|

```

그 다음 Pincode를 주어진대로 0000으로 변경해주자.

```

Change Pincode
Within 12 character > 0000
===== AUTHENTICATION SUB MENU =====
1 => AUTHENTICATION        : DISABLED
2 => PIN CODE(PASS KEY)    : 0000
3 => ENCRYPTION            : DISABLED
=====

```

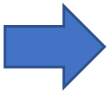
그 다음 Connection mode를 CNT_MODE 4 SLAVE로 바꿔주자 (ROLE 은 이미 SLAVE)


```

Select(0 ~ 7) > 3
Change Connection mode
1 : CNT_MODE1      2 : CNT_MODE2
3 : CNT_MODE3      4 : CNT_MODE4
Select(1 ~ 4) > 4
|=====|
|          Model name : FB755          |
|          Version   : 1.2.6          |
|=====|

```

이 상태에서 CONFIG SELECT의 3v3 입력을 해제(점프선 해제)하고 보드 전원을 껐다 켜면 아래의 사진처럼 AT 명령어 대기 모드가 된다. 그러면 AT+BTSCAN 커맨드를 입력한다.

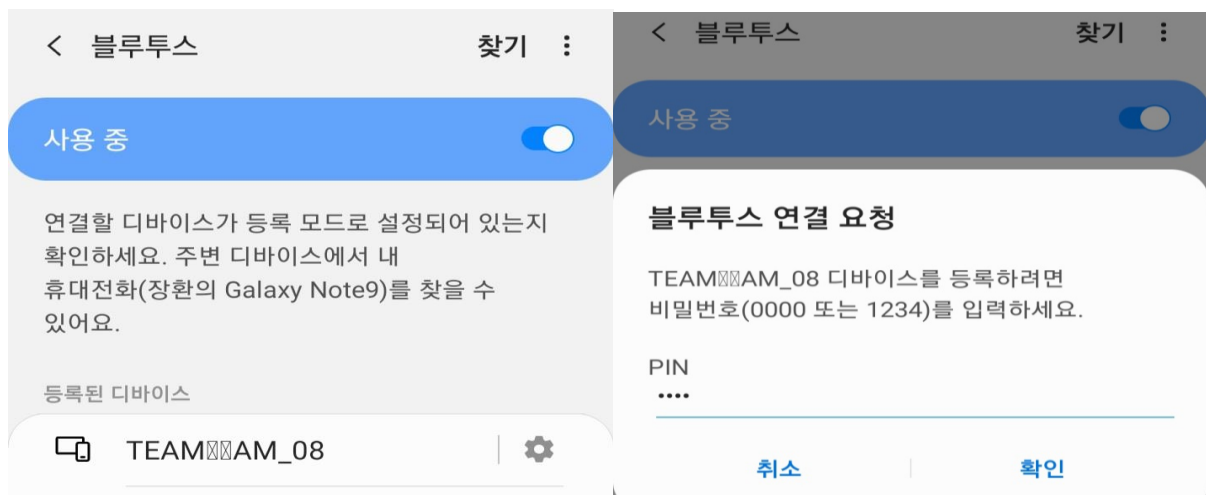


```

BTWIN Slave mode start
OK
AT+BTSCAN
ERROR
AT+BTSCAN
OK

```

그러면 아래의 사진처럼 스마트폰으로 TEAM_08이라고 Device name을 설정한 블루투스 기기를 찾을 수 있고, 아까 설정한 Pincode 0000을 입력해주면 연결이 가능하다.



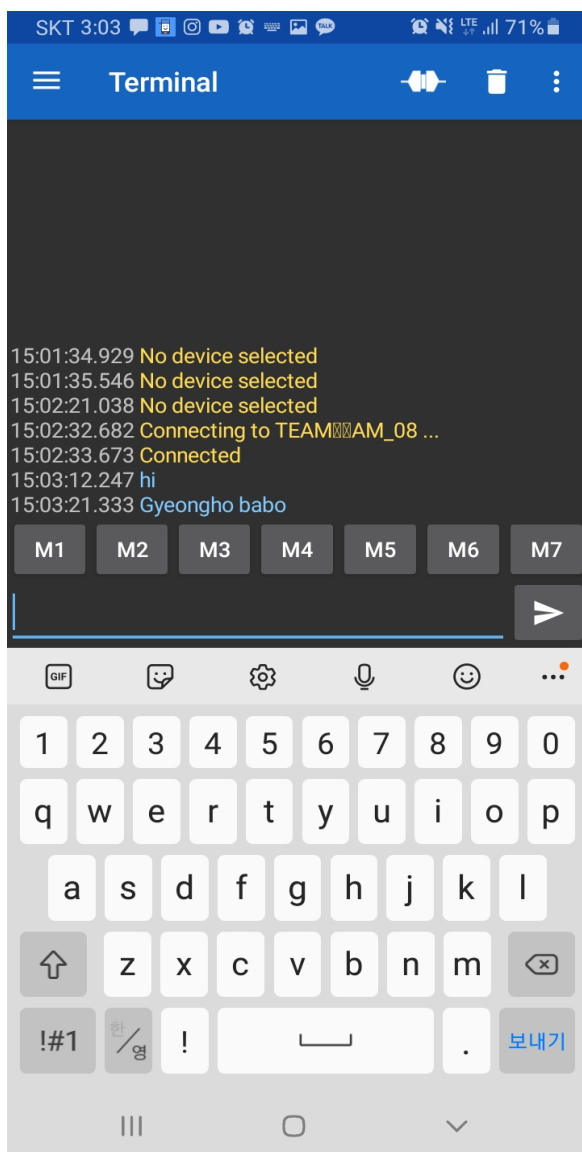
자 그러면 이제 스마트폰으로 "Serial Bluetooth Terminal" 어플을 다운 받자. 그러면, 해당 TEAM_08 블루투스 기기와 연결되는 것을 볼 수 있고, 연결되면 아래와 같이 Putty창에 CONNECT ~~가 메시지 나타나고, 스마트폰 어플에도 Connecting to TEAM_08 / Connected 라는 메시지가 나타나며 서로 연결되는 것을 볼 수 있다.


```
Select(0 ~ 7) >
BTWIN Slave mode start

OK
AT+BTSCAN
ERROR
AT+BTSCAN
OK

CONNECT 08AED6F0E2EA
█
```

스마트폰 어플에 아무 글자나 쳐보자. 그럼 메시지가 전송이 되는 것을 볼 수 있다.



```
=====
Select(0 ~ 7) >
BTWIN Slave mode start

OK
AT+BTSCAN
ERROR
AT+BTSCAN
OK

CONNECT 08AED6F0E2EA
hi
Gyeongho babo
█
```

4 코드 설명

(1) RCC_Configure

```
void RCC_Configure(void)
{
    // TODO: Enable the APB2 peripheral clock using the function 'RCC_APB2PeriphClockCmd'

    /* UART TX/RX port clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    /* USART1 clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);

    /* USART2 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

    /* Alternate Function IO clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
}
```

- TX/RX를 사용하기 위해 PORT A, USART 통신을 위해 USART1, USART2를 활성화 시킨다.

(2) GPIO_Configure

```
void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    // TODO: Initialize the GPIO pins using the structure 'GPIO_InitTypeDef' and the function 'GPIO_Init'

    /* UART1 pin setting */
    //Tx : PA9 - send signal with putty
    //Rx : PA10 - receive signal with putty
    //TX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    //RX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*USART2 pin setting */
    //Tx : PA2 - send signal with bluetooth module ( FB755AC )
    //Rx : PA3 // receive signal with bluetooth modul ( FB755AC )
    //TX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    //Rx
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

- Datasheet , Reference Manual 을 통해 입출력에 필요한 모드들을 설정하였고 세부사항은 다음과 같다.

< TX >

TX의 출력으로 사용하기 위해 PORT A의 9번 PIN을 GPIO_PIN으로 사용하고 최대 출력속도는 50MHz, 모드는 AFPP(Alternative-Function Push-Pull)모드로 설정한다.

< RX >

RX의 입력으로 사용하기 위해 PORT A의 10번 PIN을 GPIO_PIN으로 사용하고 모드는 Input with Pull-Down 모드로 설정한다.

< TX – USART2 >

TX의 출력으로 사용하기 위해 PORT A의 2번 PIN을 GPIO_PIN으로 사용하고 최대 출력 속도는 50MHz, 모드는 AFPP(Alternative-Function Push-Pull)모드로 설정한다

< RX – USART2 >

RX의 입력으로 사용하기 위해 PORT A의 3번 PIN을 GPIO_PIN으로 사용하고 모드는 Input with Pull-Down 모드로 설정한다.

(3) GPIO_Configure

```
void USART1_Init(void)
{
    USART_InitTypeDef USART1_InitStructure;

    // Enable the USART1 peripheral
    USART_Cmd(USART1, ENABLE);

    // TODO: Initialize the USART using the structure 'USART_InitTypeDef' and the function 'USART_Init'
    USART1_InitStructure.USART_BaudRate = 9600;
    USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART1_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART1_InitStructure.USART_Parity = USART_Parity_No;
    USART1_InitStructure.USART_StopBits = USART_StopBits_1;
    USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_Init(USART1, &USART1_InitStructure);

    // TODO: Enable the USART1 RX interrupts using the function 'USART_ITConfig' and the argument value 'Receive Data register not empty interrupt'
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
}

void USART2_Init(void)
{
    USART_InitTypeDef USART2_InitStructure;

    // Enable the USART2 peripheral
    USART_Cmd(USART2, ENABLE);

    // TODO: Initialize the USART using the structure 'USART_InitTypeDef' and the function 'USART_Init'
    USART2_InitStructure.USART_BaudRate = 9600;
    USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART2_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART2_InitStructure.USART_Parity = USART_Parity_No;
    USART2_InitStructure.USART_StopBits = USART_StopBits_1;
    USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_Init(USART2, &USART2_InitStructure);

    // TODO: Enable the USART1 RX interrupts using the function 'USART_ITConfig' and the argument value 'Receive Data register not empty interrupt'
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
}
```

- 주어진 블루투스 FB755AC 모듈의 UART 설정은 다음과 같다.

Baud rate – 9600

Hardware Flow Control None – 수신 측에서 수신이 불가능한 경우 송신 측에서 데이터를 전송하지 않고, 수신 측이 다시 수신 가능한 상태일 경우에 데이터를 전송하도록 해주는 설정이다.

RX , TX – 데이터 송수신 모두 사용 가능하도록 설정하였다.

Parity No – Error 검사를 위한 Parity bit는 None으로 설정하였다.

StopBits_1 – 통신 종료를 알리기 위한 stop bit로 1 bit로 설정하였다.

WordLength – 데이터 길이를 위한 설정으로 8 bit로 설정하였다.

- 인터럽트를 감지가능 하도록 하기 위해서 USART1,2 RX를 활성화시킨다.

(4) NVIC_Configure

```
void NVIC_Configure(void) {  
  
    NVIC_InitTypeDef NVIC_InitStructure;  
  
    // TODO: fill the arg you want  
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);  
  
    // TODO: Initialize the NVIC using the structure 'NVIC_InitTypeDef' and the function 'NVIC_Init'  
  
    // 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);  
  
    //UART2  
    NVIC_EnableIRQ(USART2_IRQn);  
    NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;  
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO  
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO  
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;  
    NVIC_Init(&NVIC_InitStructure);  
}
```

- Pre-emption priority가 모두 동일하다는 가정하에 진행하였고, subpriority를 0으로 설정하여 동일한 우선순위를 주었다. 그리고 다른 Pre-emption priority group이 필요하지 않기 때문에 NVIC_PriorityGroup_0으로 설정한다.

(5) USART1_IRQHandler

```
//PC <-> board
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);

        //send data
        sendDataUART1(word);
        sendDataUART2(word);

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

- PC의 Putty에서 문자가 입력될 경우 블루투스 모듈을 통해 스마트폰의 터미널에 출력되고, 스마트폰의 터미널에서 문자가 입력될 경우 Putty에 출력이 되도록 인터럽트 핸들러를 설정하였다.

(6) USART2_IRQHandler

```
//board <-> bluetooth
void USART2_IRQHandler() {
    uint16_t word;
    if(USART_GetITStatus(USART2, USART_IT_RXNE) != RESET){
        // the most recent received data by the USART1 peripheral
        word = USART_ReceiveData(USART2);

        //send data
        sendDataUART1(word);

        // clear 'Read data register not empty' flag
        USART_ClearITPendingBit(USART2, USART_IT_RXNE);
    }
}
```

- 스마트폰의 터미널에서 문자가 입력될 경우 Putty에 출력이 되도록 인터럽트 핸들러를 설정하였다.

(7) sendDataUART1 , sendDataUART2

```
void sendDataUART1(uint16_t data) {  
    /* Wait till TC is set */  
    USART_SendData(USART1, data);  
}
```

```
void sendDataUART2(uint16_t data) {  
    /* Wait till TC is set */  
    USART_SendData(USART2, data);  
}
```

- 데이터를 전송하는 함수이다.

(8) main

```
int main(void)  
{  
  
    // init system, RCC, CPI0  
    SystemInit();  
    RCC_Configure();  
    GPIO_Configure();  
  
    // init UART1_config  
    USART1_Init();  
    // init UART2_config  
    USART2_Init();  
  
    // init NVIC_config  
    NVIC_Configure();  
  
    while (1) {  
        // TODO: implement  
        // Delay  
        Delay();  
    }  
    return 0;  
}
```

- 위에서 정의한 함수들과 while문을 순차적으로 실행시켜 main문을 완성한다.

5 실험 결과

3. 실험 내용 부분에서 모든 실험결과를 확인할 수 있습니다.

6 결론 및 느낀점

이번 주 실험에서 납땜을 수행하였는데 납땜연결이 정상적으로 이루어졌는지 확인하는 작업이 필수적으로 필요하였다. 만약 납땜 연결 여부를 확인하지 않으면 납땜이 정상적으로 이루어진 것인지 확인하기 힘들었다. 멀티미터를 이용하여 연결 여부를 확인하며 순차적으로 작업이 이루어져 해당 주차 실험내용을 정상적으로 수행할 수 있었고 블루투스 모듈을 처음으로 사용해보면서 블루투스 기능을 숙지할 수 있는 기회가 되었다. 이외에도 USART, 블루투스 모듈과 스마트폰 간에 동작 원리를 상기하였으며 마스터와 슬레이브 관계에 대해서 직접적인 경험을 통해 이해할 수 있었다.

프로젝트 진행 시 블루투스 기능을 잘 사용하여 구현하면 좋은 결과물을 만들 수 있을 것 같다는 생각이 들었다..