9 주차 결과보고서

10 조 (001 분반)
과목명:임베디드시스템설계및실험
담당 교수 : 정상화교수님
담당 조교 : 최호진 조교님
조원 : 201924603 하규승(조장)
201727102 강준혁
201924525 이광훈
202023139 박지원
제출 날짜 : 2023.11.05

목차

1. 실험목표

Bluetooth 동작 및 기판 납땜

그림 1 블루투스

블루투스는 기기 간의 근거리 연결에 사용되는 무선 통신 표준 기술이다. 2.4GHz 대역폭을 사용하여 전파 사용에 대한 허가를 받을 필요가 없어 저전력의 개인 무선기기에 많이 사용된다. 또한 마스터 슬레이브 구조로 동작한다.

그림 2 블루투스 디바이스 주소

블루투스 기기의 주소는 16 진수 12 자리로 표기됨(48-bit). 앞 24-bit 는 기기 제조사를 나타내는 OUI 로 사용됨. 인증, 암호화 등을 통해 보안이 제공됩니다.

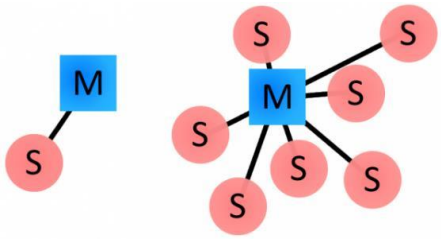

그림 3 마스터 슬레이브 구조

피코넷 사용, Inquiry, Paging, Connecting 의 절차를 거쳐 연결.

블루투스 연결 절차로는
1. Inquiry: Master 가 연결을 원하는 Slave 장치를 탐색, 연결 요청이 오면 FHS 패킷을 통해 Master 의 주소와 clock 을 보냄.

2. Paging: 받은 주소와 clock 정보 등으로 Master 와 동기화하여 실제 커넥션을 수행.

블루투스 프로파일: 블루투스와 연결되는 장비의 종류에 따라 규정되는 개별적인 프로토콜. 통신을 위해서는 같은 프로파일을 사용해야 함. 실험을 위하여 SPP(Serial Port Profile) 사용.

SPP(Serial Port Profile): 가상 시리얼 포트를 설정하고 두 개의 Bluetooth 장치를 연결. 두 장치가 연결되면 케이블로 연결된 것처럼 동작.

SSID(Service Set identifier): Wi-Fi 나 블루투스와 zkxdms 무선 통신에서 네트워크를 식별하기 위해 사용되는 식별자. 블루투스에서는 기기 간 페어링 시 사용.

UUID(Universally Unique Identifier): 고유의 ID 를 나타내는 128-bit 숫자 조합. 블루투스 장치에서 제공되는 서비스를 구분하기 위해 사용.

2. 세부 목표

2.1 만능기판 납땜
2.2 PC 의 Putty 프로그램과 Bluetooth 모듈 간 통신이 가능하도록 펌웨어 작성
2.3 Bluetooth 의 CONFIG_SELECT 핀에 3v3 준 상태에서 보드를 켠 후 putty 에 설정 메뉴가 뜨면 강의 자료 참고하여 설정 변경
- name 은 MON_XX(XX 는 조 번호) 로 설정
2.4 안드로이드의 Serial Bluetooth Terminal 어플리케이션을 이용하여 PC 의 Putty 와 통신

- PC 의 putty 입력 -> Bluetooth 모듈을 통해 스마트폰의 터미널에 출력
- 스마트폰의 터미널 입력 -> PC 의 Putty 에 출력

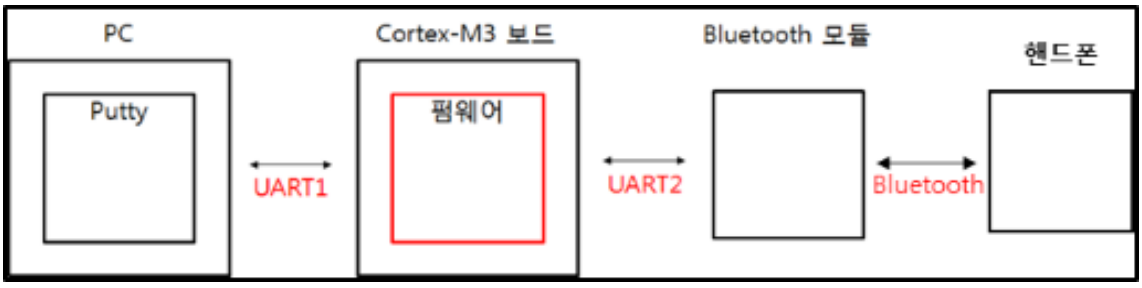

그림 4 통신 순서

3. 실험 기구

- STM32F107 보드
- Serial to USB
- 만능기판
- 실납, 인두기, 전선
- 전선탈피기, 인두기클리너
- 핀홀더 , 납흡입기
- IAR 프로그램

4. 실험 과정

먼저 프로젝트를 실행하여 세팅을 진행해줍니다. 그리고 블루투스 모듈과 STM32 보드의 각 핀들을 수월하게 연결하기 위해서 기판에 전선을 인두기를 이용해서 납땜해줍니다. 이번 실험에서는 UART1 를 통해 Putty 의 데이터 1 바이트를 수신하면 바로 UART2 를 통해 Bluetooth 모듈로 전송하고 UART2 를 통해 Bluetooth 모듈의 데이터 1 바이트를 수신하면 바로 UART1 을 통해 Putty 로 전송이 되는 기능을 구현해야 하므로 UART1(USART1), UART2(USART2)를 설정해주어야 합니다.

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

    /* USART1, USART2 TX/RX port clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

    /* USART1, USART2 clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
    /* Alternate Function IO clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
}
```

그림 5 RCC_Configure 코드

USART1, USART2 의 TX/RX port 를 사용해주기 위해 clock 을 활성화시켜줍니다. USART1 의 TX/RX pin 은 PA9, PA10 이고 USART2 의 TX/RX port 는 PA2, PA3 이므로 GPIOA, USART1, USART2 에 RCC_APB2PeriphClockCmd 를 이용해 clock 을 enable 시켜줍니다.

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

    /* USART1 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;
    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_IPD;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART2 pin setting */
    //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_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

그림 6 GPIO_Configure 코드

그 다음 USART1, USART2 의 TX, RX 의 설정 값들을 설정해주기 위해 구조체를 이용합니다. 공통적으로 Max Speed 는 50MHz 으로 설정해주고, TX는 output으로 Alternate Function output mode로, RX는 input으로 input with pull-down mode 로 설정해주었습니다.

```
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_WordLength = USART_WordLength_8b;
        USART1_InitStructure.USART_StopBits = USART_StopBits_1;
        USART1_InitStructure.USART_Parity = USART_Parity_No;
        USART1_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART1, &USART1_InitStructure);
    // TODO: Enable the USART1 RX interrupts using the function 'USART_ITConfig' and the argument value '
    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_WordLength = USART_WordLength_8b;
        USART2_InitStructure.USART_StopBits = USART_StopBits_1;
        USART2_InitStructure.USART_Parity = USART_Parity_No;
        USART2_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART2, &USART2_InitStructure);


    // TODO: Enable the USART2 RX interrupts using the function 'USART_ITConfig' and the argument value
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);

}
```

그림 7 USART1_Init, USART2_Init 코드

이제 각 USART1, USART2 를 설정 값들을 설정해줍니다. Blue tooth 모듈의 기본설정이 다음과 같으므로 이에 맞춰 BaudRate 는 9600, WordLength 는

8bit, Parity bit 는 no, Mode 는 Rx/Tx, HardwareFlowControl 은 none 으로 설정해줍니다.

4 UART PARAMETERS

- 기본적으로 **DATA BIT** 는 **8 BIT** 로 고정 되어 있습니다.

4.1 BAUDRATE

항목	제품	기본 설정
BAUDRATE	FB100AS, FB200AS	9600 bps
	FB755AC, FB755AS	
	FB155BC, FB155BS	
	FB570AC, FB570AS	

- 통신속도는 1200 ~ 230400 bps 까지 지원합니다.
- 기본설정은 9600 bps 로 설정 되어 있습니다.

4.2 PARITY BIT

항목	제품	기본 설정
PARITY BIT	FB100AS, FB200AS	NONE
	FB755AC, FB755AS	
	FB155BC, FB155BS	
	FB570AC, FB570AS	

- PARITY BIT 는 NONE, ODD, EVEN 을 지원합니다.
- 기본설정은 NONE 로 설정 되어 있습니다.

4.3 STOP BIT

항목	제품	기본 설정
STOP BIT	FB100AS, FB200AS	1 BIT
	FB755AC, FB755AS	
	FB155BC, FB155BS	
	FB570AC, FB570AS	

- STOP BIT 는 1BIT, 2BIT 를 지원합니다.
- 기본설정은 1BIT 로 설정 되어 있습니다.

4.4 HARDWARE FLOWCONTROL

- 추후 지원예정

그림 8 FB755AC manual

```c
void NVIC_Configure(void) {

    NVIC_InitTypeDef NVIC_InitStructure;

    // TODO: fill the arg you want
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

    // USART1
    // '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);

    // USART2
    // 'NVIC_EnableIRQ' is only required for USART setting
    NVIC_EnableIRQ(USART2_IRQn);
    NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; // TODO
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

그림 9 NVIC_Configure

지연시간 없이 바로 동작해야 하기 때문에 NVIC 또한 설정해줍니다. USART1 의 PreemptionPriority 를 0, SubPriority 를 0, USART2 의 PreemptionPriority 를 1, SubPriority 를 0 으로 해서 우선순위를 임의로 설정해줍니다.

```c
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);

        // TODO implement
    while ((USART1->SR) == 0);
    USART_SendData(USART2, word);

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

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

        // TODO implement
    while ((USART2->SR) == 0);
    USART_SendData(USART1, word);

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

그림 10 USART1_IRQHandler, USART2_IRQHandler 코드

마지막으로 IRQHandler() 함수를 통해 블루투스 연결로 data 를 전달하거나 전달받을 수 있도록 구현해줍니다. (USART1->SR)==0 일때 USART2 로 word 를 보내고, (USART2->SR)==0 일때 USART1 로 word 를 보내 통신이 가능하게 합니다.

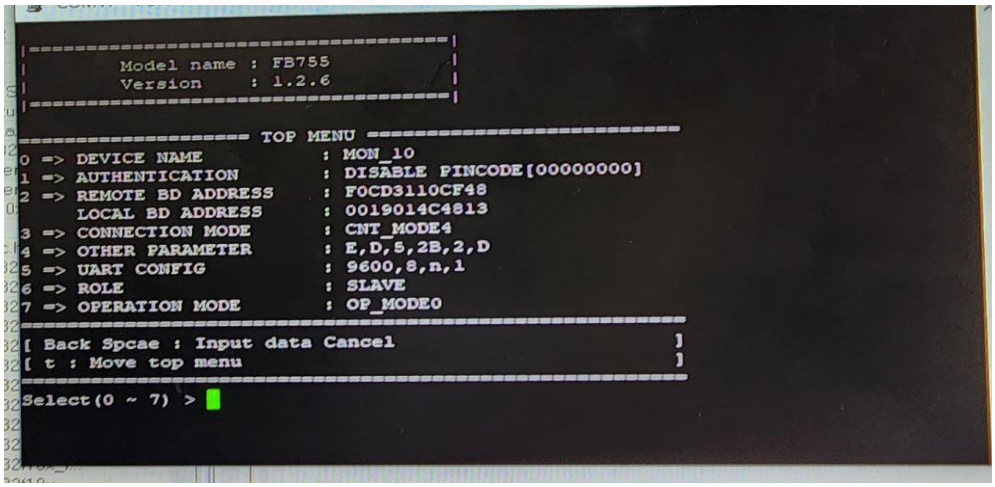

그림 11 Putty 설정 모드

이제 코드를 컴파일 하고 Putty 를 실행시켜줍니다. CONFIG SELECT 에 점프선으로 3v3 을 입력한 상태로 보드 전원을 껐다 킵니다. Putty 설정 모드가 뜨면 Device name 을 MON_10 으로 설정해주고 Pincode 비밀번호를

00000000 으로 설정해줍니다. Connection mode 4 slave 및 UART config (9600, 8, n, 1)로 설정을 해줍니다.

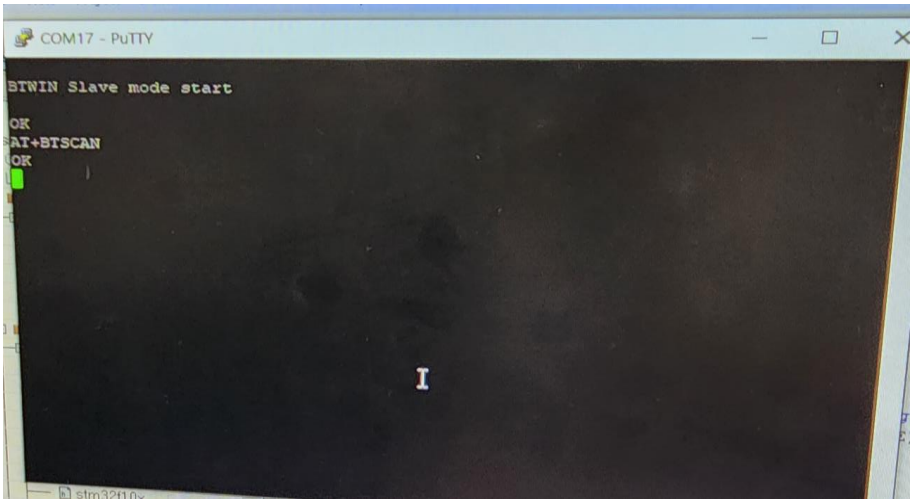

그림 12 Putty AT 명령어 대기 모드

CONFIG SELECT 의 3v3 입력을 해제하고 보드 전원을 껐다 켜서 AT 명령어 대기 모드를 킵니다. "AT+BTSCAN" 커맨드를 입력하여 연결 대기 모드를 돌입하고 스마트폰을 이용해서 연결을 시도 합니다.

5. 실험 결과 및 결과에 대한 논의

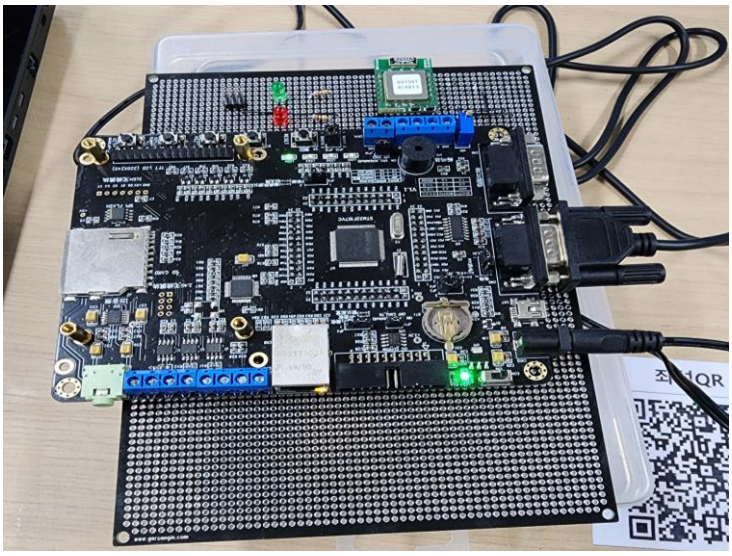

그림 13 기판과 STM32 연결

기판을 납땜을 시킨 후 STM32 보드와 연결한 모습입니다. LED 를 통해 잘 연결되어 있는지 확인할 수 있습니다.

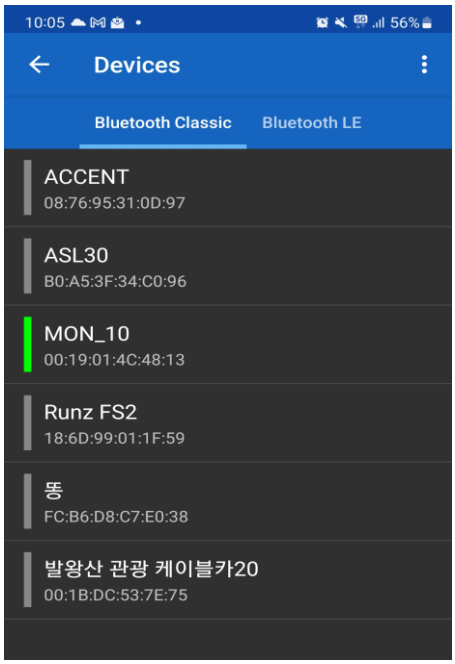

그림 14 스마트폰 연결 화면

스마트폰과 블루투스를 연결한 후 스마트폰에 "Serial Bluetooth Terminal" 어플리케이션을 설치해서 device name 을 MON_10 으로 설정해주었기 때문에 MON_10 과 연결을 합니다.

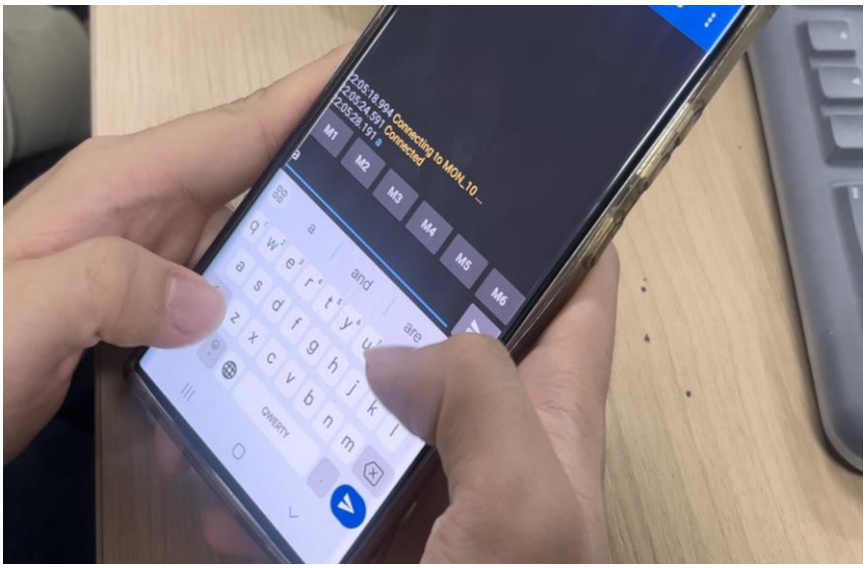

그림 15 스마트폰에서 PC로 송신 화면

스마트폰에서 연결이 된 것을 확인한 후 a 라는 문자를 보냈습니다. 이는 Putty 에서 a 라는 문자가 뜬 것을 확인하여 스마트폰에서 PC(Putty)로의 송신은 잘 되는 것을 확인할 수 있었습니다.

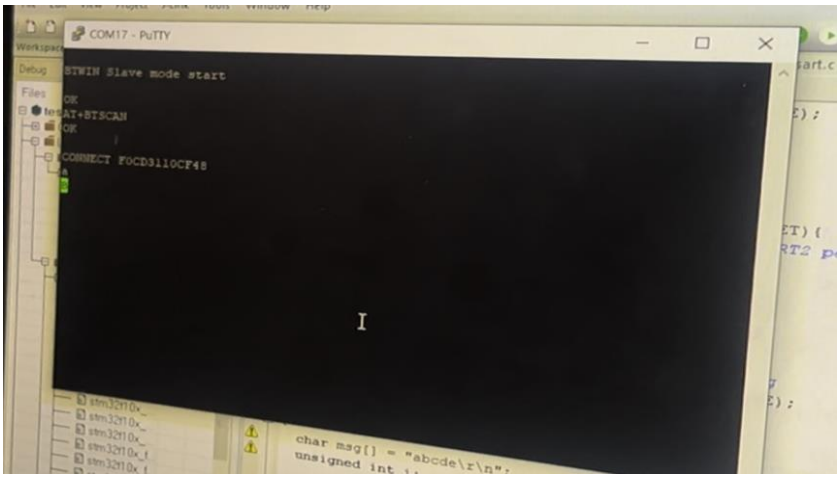

그림 16 PC에서 수신 및 송신 화면

a 라는 문자를 수신 받은 후 마찬가지로 B 라는 문자를 입력하여 스마트폰으로 전송이 잘 되는지 확인해보았습니다.

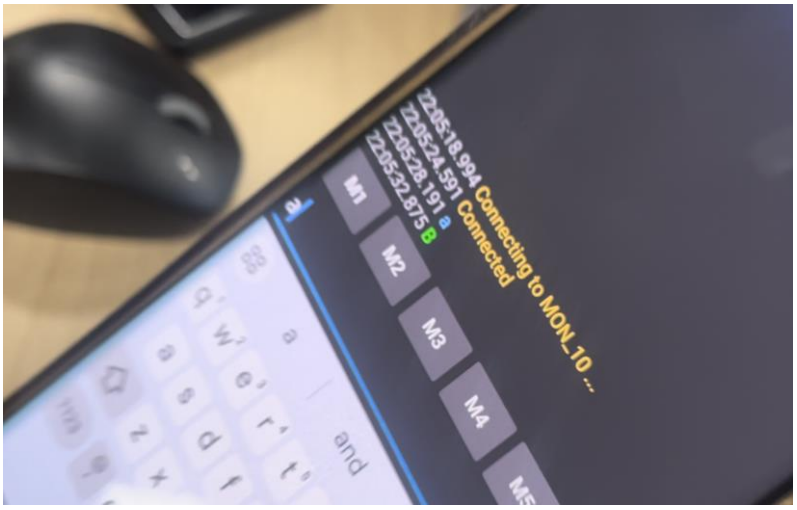

그림 17 스마트폰에서 수신 화면

스마트폰에서 B 라는 문자가 잘 출력된 것을 확인하였고 PC(Putty)에서 스마트폰으로도 송신이 잘 되는 것을 확인할 수 있었습니다. 이에 따라

목표인 PC 의 putty 프로그램과 Bluetooth 모듈 간 통신이 잘 수행되었으며 정상 작동함을 알 수 있습니다.

6. 결론

이번 실험을 통해 기판을 STM32 보드 각 pin위치에 맞춰 납땜을 한 후 STM32 보드 및 블루투스 모듈과 연결하여 통신을 위한 USART1, USART2 활성화와 관련된 코드를 작성해보고 컴파일을 해보았습니다. 그 후 Putty를 통해 설정 값을 설정한 후 연결을 시도해보았으며, PC의 Putty와 스마트폰 간의 문자 전달이 잘 되는 것을 알 수 있었습니다. 이에 따라 원래 목표였던 Bluetooth 동작 및 기판 납땜을 해보았고 이에 대해 배울 수 있었습니다.