임베디드 시스템 설계 및 실험
화요일 분반 7주차 실험 결과 보고서

조 : 6조
조원 : 이주승 김선규 이동현 이지현 최세희

실험목표

1. Bluetooth 동작
2. 기판 납땜

세부 목표

1. 만능 기판 납땜
2. PC의 putty 프로그램과 Bluetooth 모듈 간 통신이 가능하도록 펌웨어 작성
3. Bluetooth의 CONFIG_SELECT 핀에 3v3 준 상태에서 보드를 켠 후 putty에 설정 메뉴가 뜨면 강의 자료 참고하여 설정 변경
   - name은 TUE_XX (XX는 조 번호) 로 설정
4. 안드로이드의 Serial Bluetooth Terminal 어플리케이션을 이용하여 PC의 putty와 통신
   - PC의 putty 입력 -> Bluetooth 모듈을 통해 스마트폰의 터미널에 출력
   - 스마트폰의 터미널 입력 -> PC의 Putty에 출력

이론적 배경

1. 블루투스(Bluetooth)

-> 블루투스는 근거리 무선통신기술로 스마트폰, 무선 이어폰, 웨어러블 기기 등에서 디지털 데이터를 주고 받는다. 주로 높은 신뢰성에 저가의 무선 통신을 구현하는 것이 목표이다.

2. Master/Slave 역할 동작

-> 블루투스를 사용하기 위해서는 Connection하기 위한 Master/Slave 역할에 따른 동작들이 있다.

-> Master는 Inquiry(검색) 및 Page(연결요청)로 되어있고, Slave는 Inquiry Scan(검색 대기) 및 Page Scan(연결대기)로 되어있으며 일련의 동작들로 인해 Connected 되어 Data를 송수신 할 수 있게 된다.

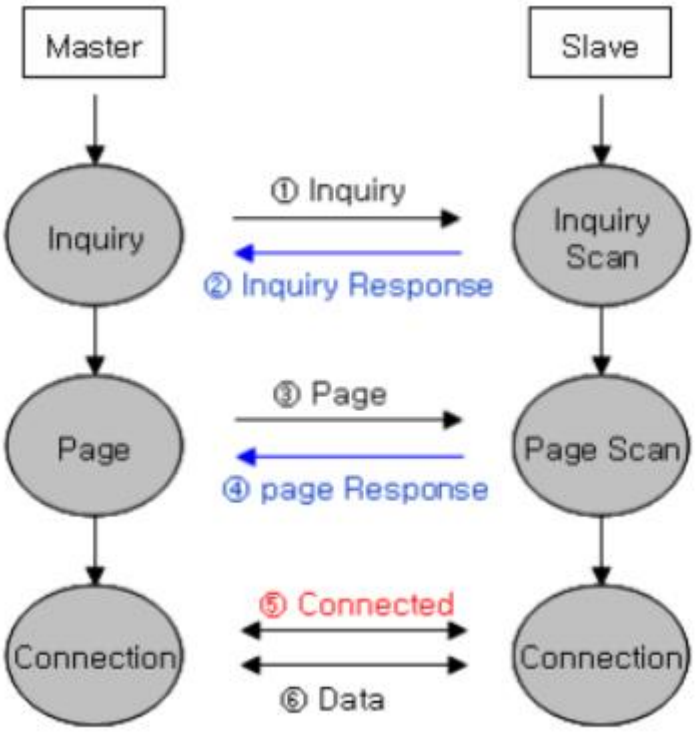


그림 1 Master/Slave의 역할 및 동작

3. 블루투스 프로파일

-> 어플리케이션 관점의 블루투스 기기의 기능별 성능을 정하는 사양으로 다른 블루투스 기기와 통신하는데 사용하는 특성을 규정한다.

-> SPP(Serial Port Profile) : RS232 시리얼 케이블 에뮬레이션을 위한 블루투스 기기에 사용되는 프로파일로 유선 RS232 케이블이 연결된 것처럼 무슨 블루투스 통신이 수행 가능하다.

4. Identifier

-> 블루투스를 사용할 때에는 필요한 고유 식별자가 있다.

-> SSID(Service Set Identifier)는 무선랜을 구별하기 위한 고유 식별자로 클라이언트가 무선랜을 통해 접속할 때 사용된다.

-> UUID(Universally Unique Identifier)는 네트워크 상에서 서로 다른 개체들을 구별하기 위한 128비트 고유 식별자로 블루투스에서는 서비스의 종류를 구분하기 위해 사용된다.

5. 블루투스 모듈 (FB755AC)

-> AT명령어를 지원하는 Bluetooth v2.1 가능 모듈로 클라이언트 7개까지 연결 가능하다.

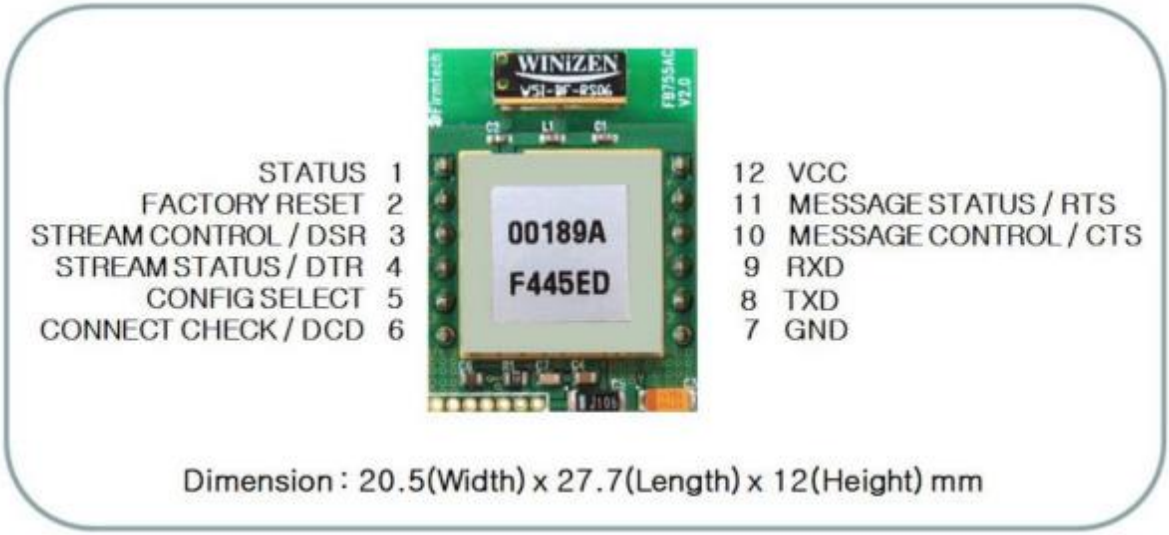


그림 2 FB755AC 모듈

6. AT 명령어

-> 모뎀 모듈을 제어하는데 쓰이는 명령어로 AT명령어 set을 통해 모듈을 제어할 수 있다.

-> CONNECTION MODE 4 가 AT명령어 대기 상태이다.

-> AT+BTSCAN은 블루투스의 검색대기(inquiry scan)과 연결대기(page scan)을 위한 명령어이다.

3.2.5 CONNECTION MODE4

- CONNECTION MODE4 는 AT 명령어 대기 상태로서 전원이 인가되면 명령어 대기만 하고 있기 때문에 일련의 동작을 하기 위해서는 AT 명령어를 입력 하셔서 사용 해야 합니다. AT 명령어 사용에 관해서는 "부록 B AT 명령어 세부 설명"을 참조 하시기 바랍니다.

그림 3 CONNECTION MODE 4의 설명

```
BTWIN Slave mode start

OK
AT+BTSCAN
OK
```

그림 4 AT명령어

3.14 AT+BTSCAN

FEATURE	BT 의 검색대기(inquiry scan)와 연결대기(page scan)를 하도록 합니다.
APPLY PRODUCTS	FB100, FB200, FB755, FB155, FB155 HID, FB570
RESPONSE	∠OK∠ ∠CONNECT 123456789012∠ (다른 장치와 연결이 된 경우의 응답)
DESCRIPTION	블루투스 장치는 장치검색 작업을 할 때 검색대기를 하고 있는 블루투스 장치만 검색 할 수 있으며, 또한 연결대기를 하고 있는 장치로만 연결이 가능합니다. 이 명령을 사용하면 다른 블루투스 장치들이 BT 를 검색할 수 있고, 항상 연결도 가능합니다. 블루투스 장치로부터 연결이 이루어지면, "CONNECT 123456789012" 메시지가 표시되며, 연결이 종료되면 다시 검색대기와 연결대기를 수행합니다. 따라서 명령어 대기상태로 전환하려면 AT+BTCANCEL 명령을 사용하여야 합니다.

그림 5 AT+BTSCAN의 설명

구현 내용

1. Clock Enable

```
void RCC_Configure(void) {
        /* 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);
}
```

실험에 필요한 GPIO A(TX/RX), USART1, USART2, AFIO Clock을 Enable한다.

2. USART Initialization

```c
void USART1_Init(void){
	USART_InitTypeDef USART1_InitStructure;
	NVIC_InitTypeDef NVIC_InitStructure;

	/* UART pin setting */
	//TX
	GPIO_InitTypeDef GPIO_InitStructure_TX;
	GPIO_InitStructure_TX.GPIO_Pin = GPIO_Pin_9;
	GPIO_InitStructure_TX.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure_TX.GPIO_Mode = GPIO_Mode_AF_PP;
	GPIO_Init(GPIOA, &GPIO_InitStructure_TX);

	//RX
	GPIO_InitTypeDef GPIO_InitStructure_RX;
	GPIO_InitStructure_RX.GPIO_Pin = GPIO_Pin_10;
	GPIO_InitStructure_RX.GPIO_Mode = GPIO_Mode_IPU|GPIO_Mode_IPD;
	GPIO_Init(GPIOA, &GPIO_InitStructure_RX);

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

	// 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);

	// 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);

	// 'NVIC_EnableIRQ' is only required for USART setting
	NVIC_EnableIRQ(USART1_IRQn);
	NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x01 ;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_Init(&NVIC_InitStructure);
}
```

각 GPIO의 Pin을 InitTypeDef 구조체를 사용하여 설정한다.

1) GPIO A에서 TX(Pin 9)를 Alternative Function Push-Pull mode로 설정, RX(Pin 10)를 Input Pull-Up, Input Pull-Down mode로 설정한다. 마지막으로 GPIO_Init 함수를 통해 initialize한다.

2) 5주차의 USART 설정과 라이브러리를 참고하여 USART 설정을 정의한다.

Baud rate는 9600, Hardware flow control은 None, USART mode는 RX/TX, Parity bit는 Disable, Stop bits는 1bit, Wordlength는 8bit로 설정한다.

USART interrupt source를 Receive Data register not empty interrupt로 설정한다. stm32f10x_usart.c 파일을 참고하여 parameter에 USART_IT_RXNE를 적용한다.

3) NVIC에서는 USART 세팅을 위해 USART1 Interrupt를 enable한다.

```c
void USART2_Init(void){
}
```

4) USART2도 같은 방식으로 initialize 한다.

3. IRQHandler 정의

USART IRQHandler에 Interrupt 발생 시 실행할 동작을 정의한다.

```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);

        USART_SendData(USART2, word);

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

```c
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);

        USART_SendData(USART1, word);

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

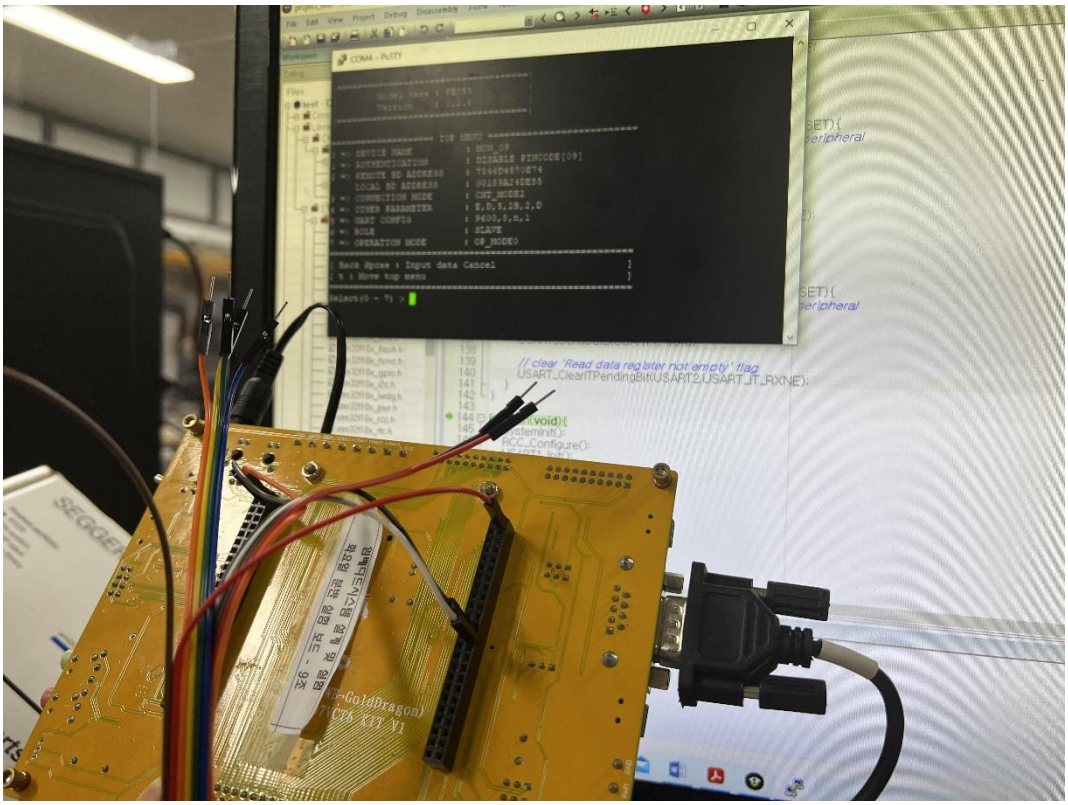
USART Interrupt 발생 시 입력 받은 Data를 USART를 통해 전달한다.

그림 6 전체적인 설계 구조

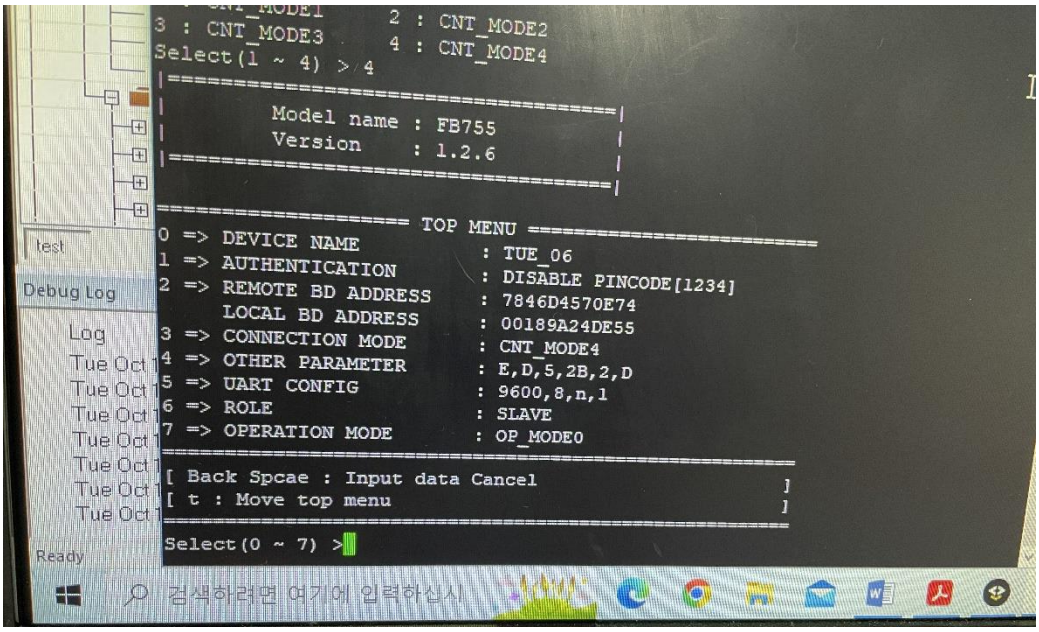
UART1을 통해 Putty의 데이터 1바이트를 수신하면 바로 UART2를 통해 Bluetooth 모듈로 전송하고, UART2를 통해 Bluetooth 모듈의 데이터 1바이트를 수신하면 바로 UART1을 통해 Putty로 전송한다.

실험 결과

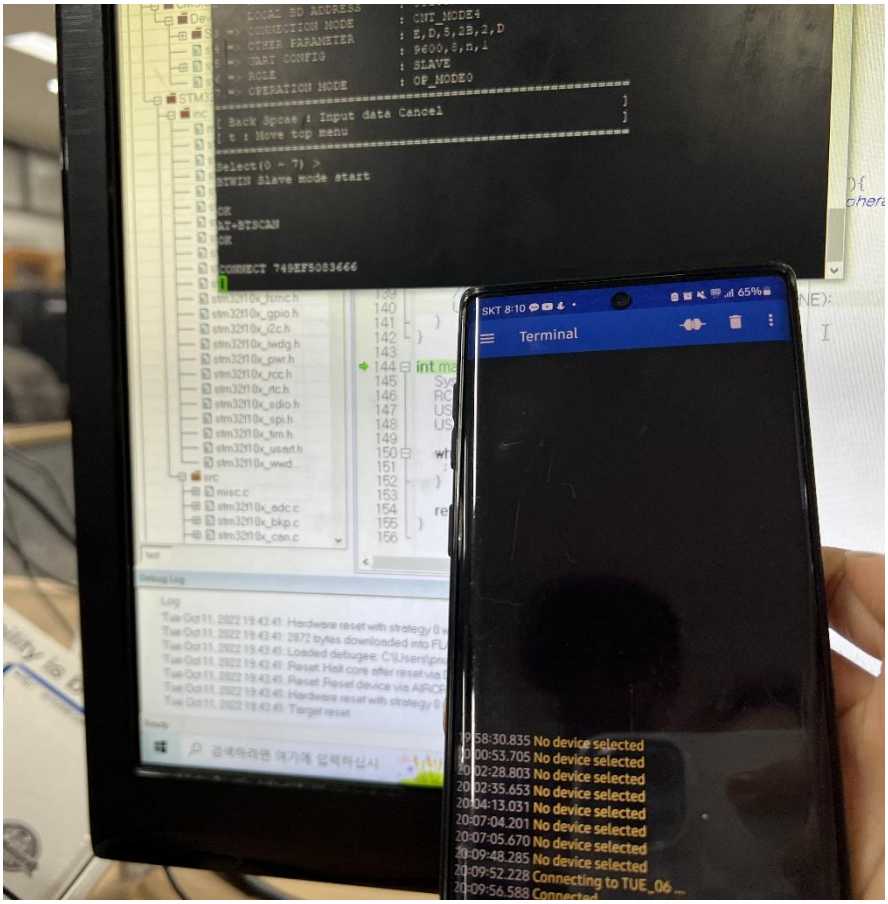
1. 블루투스 모듈 설정



CONFIG SELECT에 점프선으로 3v3와 연결한 상태로 보드 전원을 껐다 켜면 Putty를 통해 설정 모드를 시작한다. Device name과 Pincode(블루투스 연결 비밀번호), Connection mode 4 slave, UART config(9600, 8, n, 1) 을 설정한다.



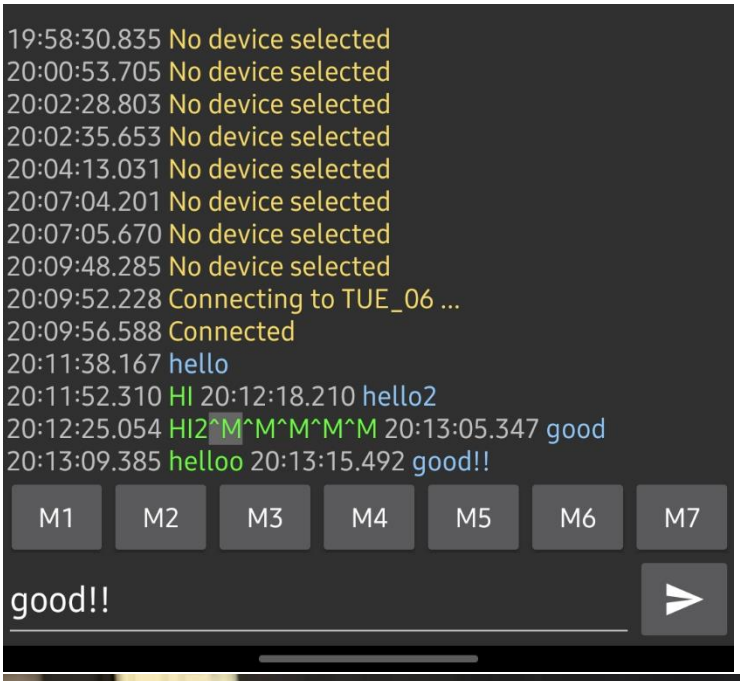
설정 후 모습

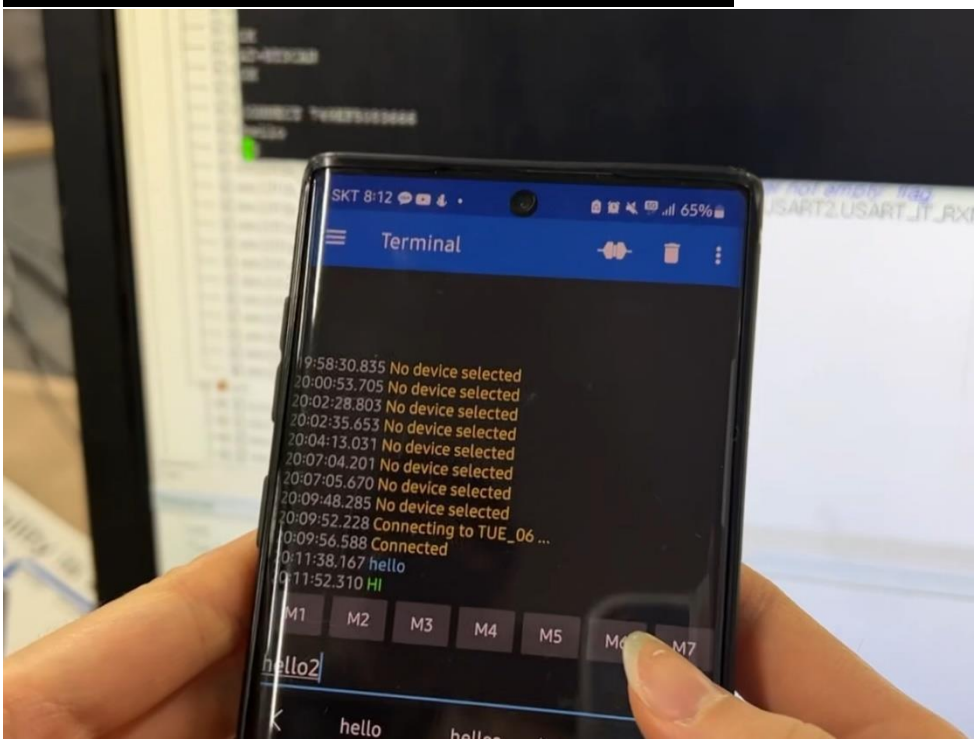


CONFIG SELECT의 3v3 입력을 해제하고 보드 전원을 껐다 켜면 AT 명령어 대기 모드로 진입한다. “AT+BTSCAN” 명령어 입력하여 연결을 기다린다.

2. 스마트폰과 Putty 블루투스 통신

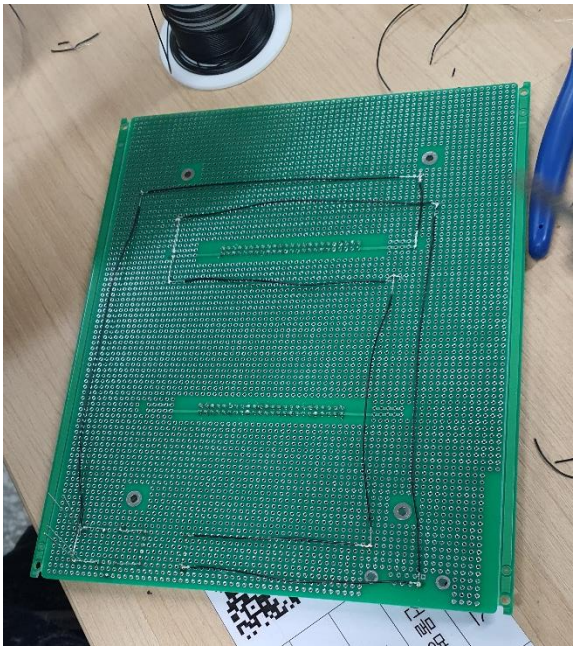
스마트폰과 블루투스 장치를 페어링하고 'Serial Bluetooth Terminal' 앱을 통해 연결한다.

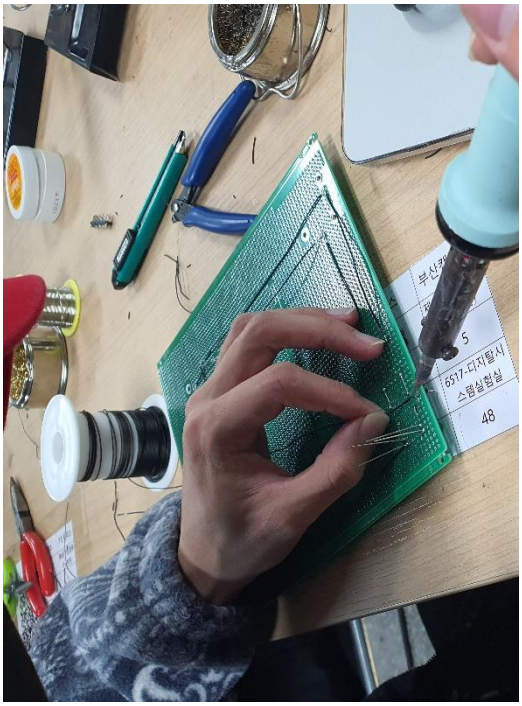


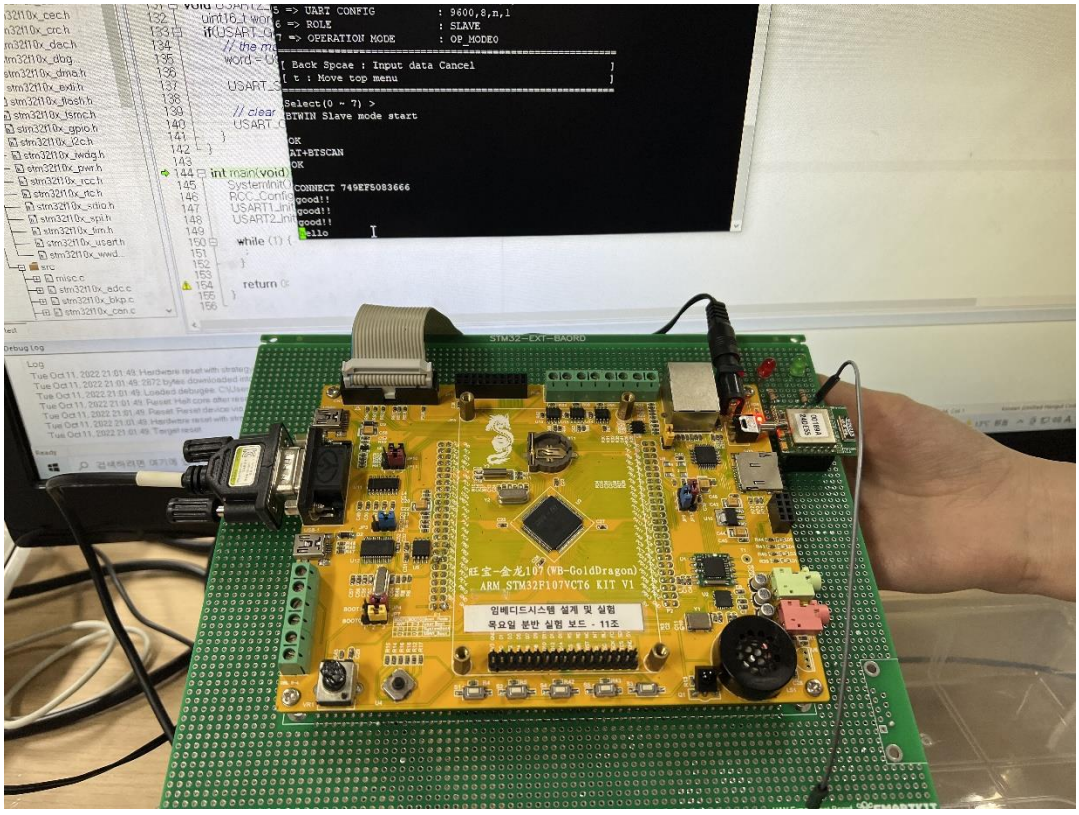


3. 만능 기판 납땜

보드와 블루투스 장치를 연결할 기판을 납땜했다.







어려웠던 점

1. Putty 입력 과정에서 터미널에 입력하는 값이 나오지 않았다. 'Putty -> Change Settings -> Terminal -> Local echo -> Force on'를 통해 해결했다.

2. Bluetooth connection 하는 과정에서 Bluetooth기기가 오랫동안 검색되지 않았다. 처음에는

Rx, Tx의 연결 위치가 바뀌어서 검색되지 않았지만, 이후에도 기판의 전원을 껐다 켜는 등, 코드와 여러 부분을 검사했지만 찾지 못했다. 이후 기기를 검색할 때, 혹은 통신을 준비할 때에는 생각보다 시간이 오래 걸린다는 점을 알게 되어서 기다렸다가 connection 하였다.

3. 기판 납땜할 때 Rx, Tx 위치를 바꾸어서 블루투스가 동작하지 않았다. 다시 제대로 납땜하여 해결했다.