임베디드시스템 설계 및 실험

9주차 결과 보고서

2023.10.31

4분반 02조

202155592 이지수

202155553 박은재

201924515 유승훈

201524410 고상현

201924402 강민수

1. 실험 목표

- Bluetooth 동작 및 납땜

- Bluetooth 모듈 (FB755AC) 를 이용한 스마트폰과의 통신

- 기판 납땜을 통해 보드와 모듈 연결

2. 이론적 배경

1) 블루투스

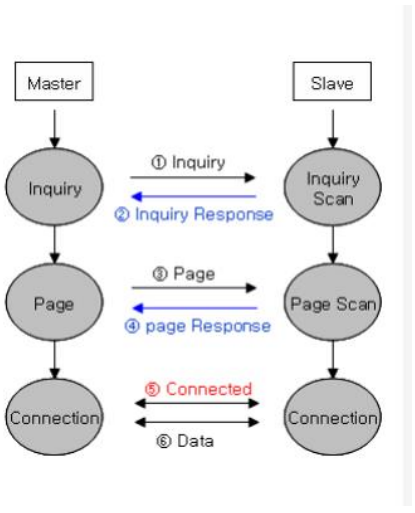
- 근거리 무선 통신 기술

- 스마트폰, 무선 이어폰, 웨어러블 기기 등에서 디지털 데이터를 주고 받는 기술

- 2.4MHz ISM 주파수 대역 사용

- 근거리, 저전력, 높은 신뢰성, 저가의 무선 통신 구현하는 것이 목표

- 기본적으로 Master와 Slave 역할로 동작 -> Master와 Inquiry(검색) 및 Page(연결 요청) / Slave 는 Inquiry Scan(검색 대기) 및 Page Scan(연결 대기)



[그림1] Master, Slave 역할 관계

2) 블루투스 프로파일

- 어플리케이션 관점에서 블루투스 기기의 기능별 성능을 정하는 사양(Specification)

- 블루투스 기기가 다른 블루투스 기기와 통신하는데 사용하는 특성을 규정함

- 다양한 프로파일 존재

- **SPP** (Serial Port Profile)

• RS232 시리얼 케이블 에뮬레이션을 위한 블루투스 기기에 사용되는 프로파일

• 유선 RS232 케이블이 연결된 것처럼 무선 블루투스 통신을 수행할 수 있음

3) Identifier

– SSID (Service Set Identifier)

• 무선랜을 통해 클라이언트가 접속할 때 각 무선랜을 구별하기 위한 고유 식별자

• Wi-Fi의 경우, 각 Wi-Fi 네트워크를 구별하기 위해 사용됨

– UUID (Universally Unique Identifier)

• 네트워크 상에서 서로 다른 개체들을 구별하기 위한 128비트 고유 식별자

• 블루투스에서는 서비스의 종류를 구분하기 위해 사용됨

3. 실험 세부 내용

1. 만능 기판 납땜

2. PC의 putty 프로그램과 Bluetooth 모듈 간 통신이 가능하도록 펌웨어 작성

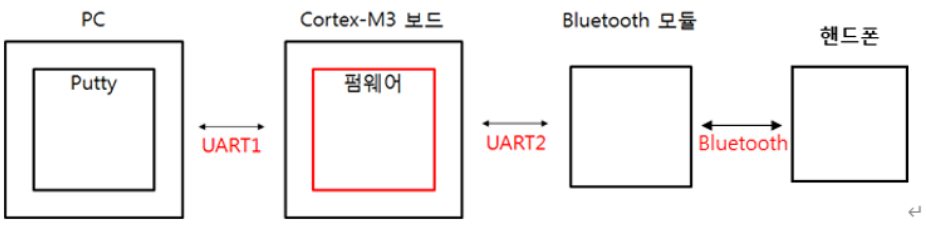
3. Bluetooth의 CONFIG_SELECT 핀에 3v3 준 상태에서 보드를 켠 후 putty에 설정 메뉴가 뜨면 강의 자료 참고하여 설정 변경

- name은 THU_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



[그림2] 안드로이드의 Serial Bluetooth Terminal 어플리케이션을 이용하여 PC의 putty와 통신

4. 실험 과정

```
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);

    /* USART1, USART2 clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART2, ENABLE);

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

[그림3] RCC_Configure 함수

RCC_Configure 함수에서는 [그림3]과 같이 실험에 사용할 USART1, USART2에 clock을 enable 한다. 이 때 'RCC_APB1PeriphClockCmd', 'RCC_APB2PeriphClockCmd' 함수 를 사용한다.

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

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

    /* 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_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART2 pin setting */
    //TX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    //RX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_PinRemapConfig(GPIO_Remap_USART1, ENABLE);
    GPIO_PinRemapConfig(GPIO_Remap_USART2, ENABLE);
}
```

[그림4] GPIO_configure 함수

GPIO_Configure 함수에서 'GPIO_InitTypeDef' 구조체와 'GPIO_Init' 함수를 사용해 GPIO 핀을 초기화한다. UART1을 통해 putty 데이터를 수신하면 UART2를 통해 Bluetooth 모듈로 전송하므로 UART1과 UART2의 TX/RX를 사용 한다. 그리고 RS232 시리얼 케이블과 블루투스를 통해 USART1, USART2 동시 사용 시에 보드에서 소음(부저음)이 발생했다. 이 문제를 Default Pin 대신 Remap Pin을 사용하여 해결하였다. A2, A3 대신 D5, D6 Pin을 사용해주었고 다음과 같은 코드를 추가하였다.

```
GPIO_PinRemapConfig(GPIO_Remap_USART1, ENABLE);
GPIO_PinRemapConfig(GPIO_Remap_USART2, ENABLE);
```

```
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.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_5;
    USART1_InitStructure.USART_BaudRate=9600;
    USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART1_InitStructure.USART_StopBits = USART_StopBits_1;
    USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART1_InitStructure.USART_Parity = USART_Parity_No;
    USART1_InitStructure.USART_Mode= USART_Mode_Rx|USART_Mode_Tx;
    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);
}
```

[그림5] USART1_Init 함수

USART1_Init 함수에서 'USART_Cmd'를 사용해 USART1을 enable 한 후, USART1_InitStructure 구조체와 USART_Init 함수를 사용해 USART1를 초기화한다. 그리고 USART1의 BaudRate, WordLength, StopBit, HardwareFlowControl, Parity, Mode를 설정한다.

```c
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.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_5;
    USART2_InitStructure.USART_BaudRate=9600;
    USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART2_InitStructure.USART_StopBits = USART_StopBits_1;
    USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART2_InitStructure.USART_Parity = USART_Parity_No;
    USART2_InitStructure.USART_Mode= USART_Mode_Rx|USART_Mode_Tx;
    USART_Init(USART2, &USART2_InitStructure);

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

[그림6] USART2_Init 함수

USART1_Init 함수와 비슷한 방법으로 초기화 한다.

```c
void NVIC_Configure(void) {

    NVIC_InitTypeDef NVIC_InitStructure;

    // TODO: fill the arg you want
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0); // need modify

    // USART1
    // 'NVIC_EnableIRQ' is only required for USART setting
    NVIC_EnableIRQ(USART1_IRQn);
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    //NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = ; // TODO
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0000; // 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 = ; // TODO
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0000; // TODO
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

[그림7] NVIC_Configure 함수

Interrupt 방식을 활용해 UART 통신을 수행 할 것이므로 USART1과 USART2에 대해 NVIC를 설정

해 준다. 이번 실험에서 인터럽트 간 우선순위는 중요하지 않으므로 SubPriority는모두 0x0000으로 설정 하였다.

```
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
        USART_SendData(USART2, word);

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

}
```

[그림8] USART1_IRQHandler

IRQHandler 함수는 USART1과 USART2에 대해 인터럽트가 발생했을 때 어떤 처리를 할지 정의한다. USART1은 입력 데이터(char)를 받아 USART2에 전송할 것이므로 다음과 같이 작성한다.
`USART_SendData(USART2, word);`

```
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
        USART_SendData(USART1, word);

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

[그림9] USART2_IRQHandler

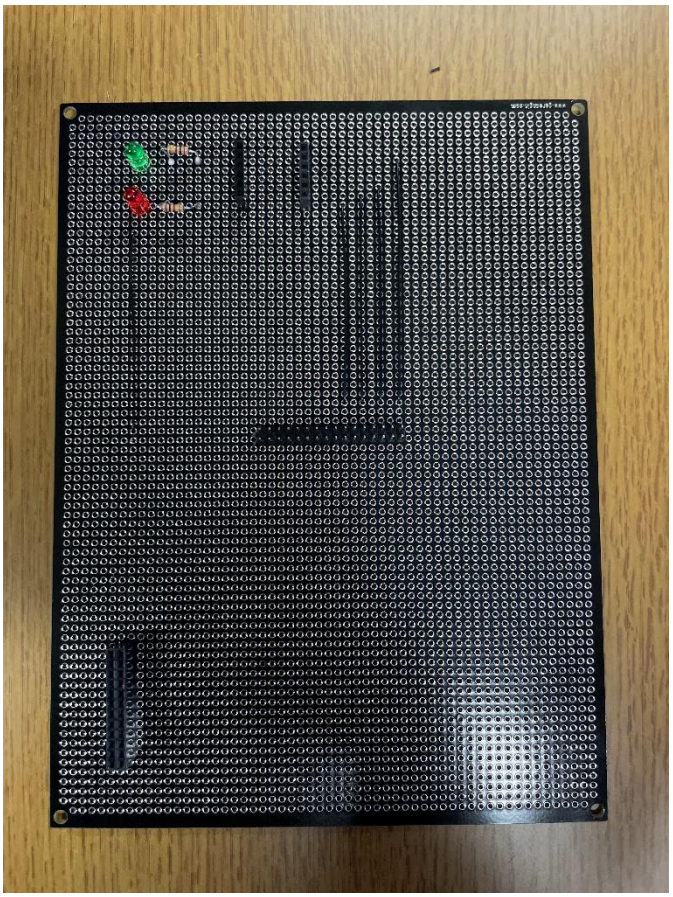
USART2는 USART1과 반대로 입력데이터를 받으면 USART1로 전송하게 한다. 위 두 함수를 통해 이제 양방향 통신이 가능하게 되었다. 납땜한 기판과 블루투스 모듈을 사용하는 포트에 맞게 연결해주면 이제 블루투스 모듈을 이용해 보드와 데이터를 주고받을 수 있게 된다.

```
int main(void)
{
  char msg[] = "abcde₩r₩n";
  unsigned int i;

    SystemInit();

    RCC_Configure();

    GPIO_Configure();

    USART1_Init();      // pc

    USART2_Init();      // bluetooth

    NVIC_Configure();

    while (1) {
    }
    return 0;
}
```

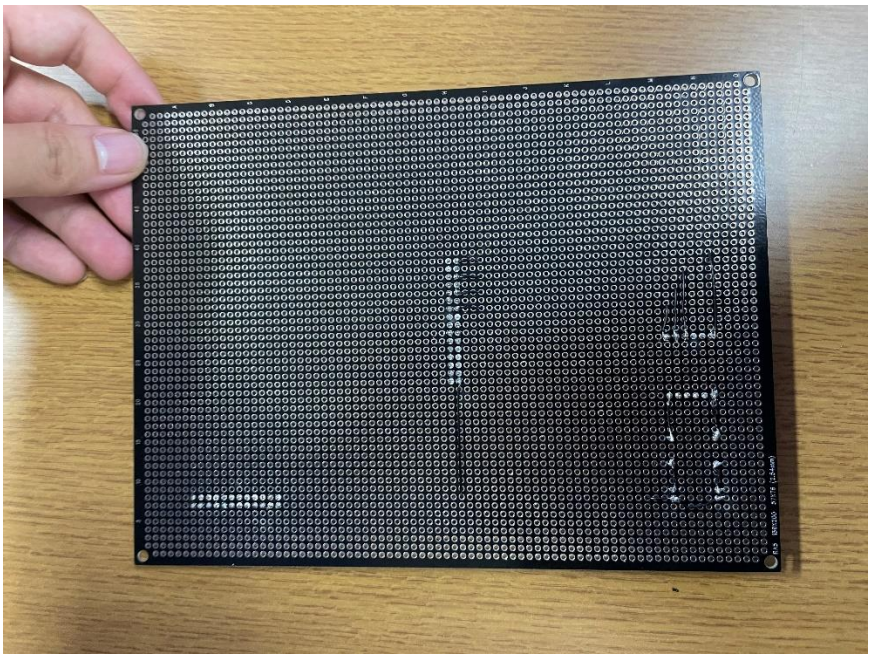
[그림10] main

main 함수에는 SystemInit(), RCC_Configure(), GPIO_Configure(), USART1_Init(), USART2_Init(), NVIC_Configure() 함수를 순차적으로 실행시킨다.

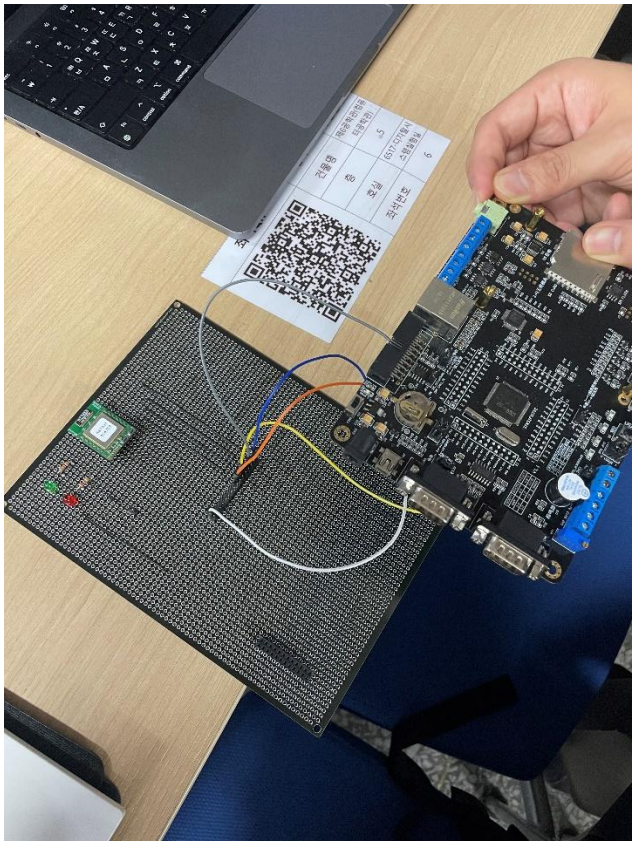
5. 실험 결과



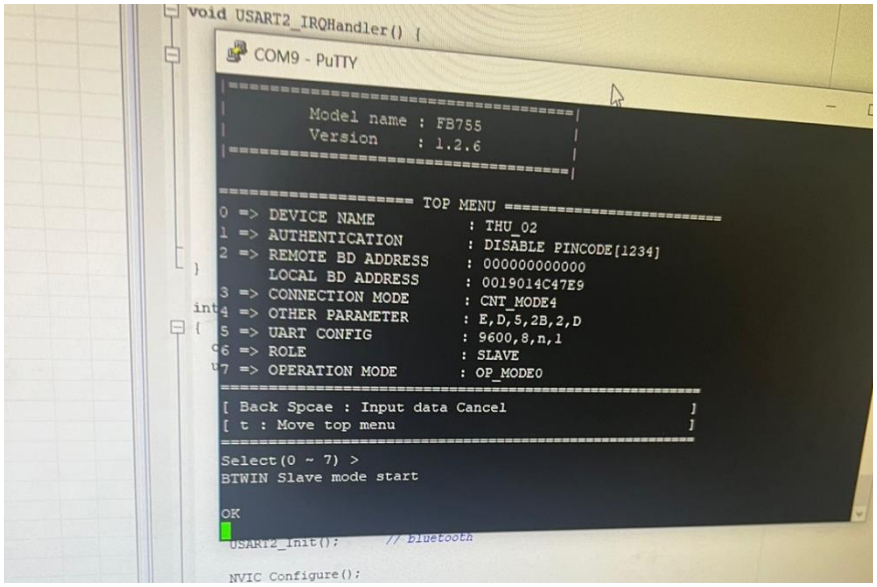
[그림11] 가판 앞면



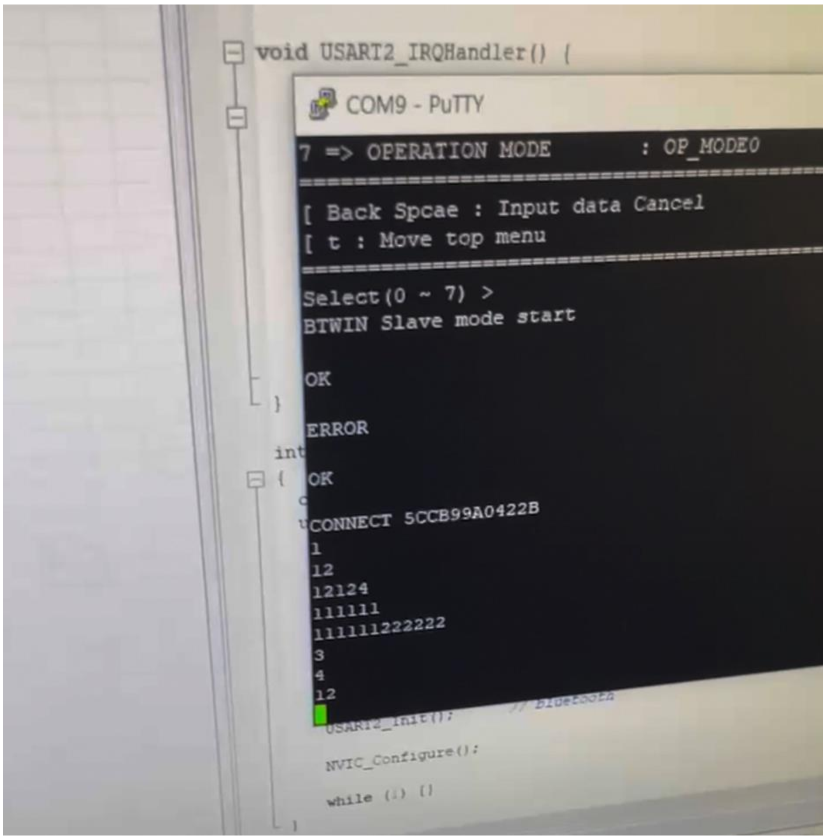
[그림12] 가판 뒷면



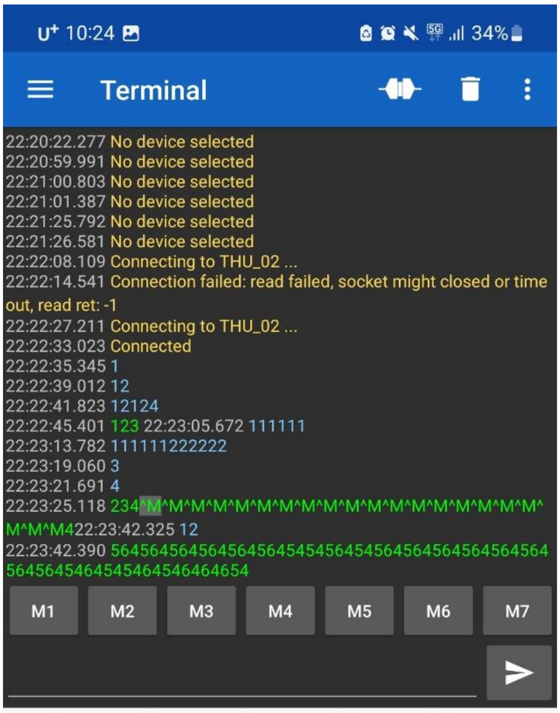
[그림13] 가판과 보드 연결



[그림14] putty 화면1



[그림15] putty 화면2



[그림16] 스마트폰에 "Serial Bluetooth Terminal" 어플리케이션을 실행한 화면

6. 결론 및 느낀점

이번 실험은 USART1을 통해 펌웨어와 PC가 통신할 수 있도록 하고, USART2를 통해 펌웨어와 Bluetooth 모듈이 통신할 수 있도록 하여 최종적으로 PC와 핸드폰에서 통신 결과를 확인하는 것 였다. 납땜하는 데 시간이 많이 소요되었지만 추후에 진행할 텀프로젝트에 도움이 될 것 같아서 보람찼다. 그리고 RS232 시리얼 케이블과 블루투스를 통해 USART1, USART2 동시 사용 시 보드에서 소음(부저음)이 발생했다. 이 문제를 Default Pin 대신 Remap Pin을 사용하여 해결하였다.