임베디드 시스템 설계 및 실험
9주차 실험 결과 보고서

조 : 8조

조원 : 서진욱, 이승민, 하연지, 하태훈

● 실험목표

1. **블루투스 통신에 필요한 장치들을 납땜을 통해 연결한다.**
2. **블루투스 모듈의 각 핀 별로 담당하는 기능을 학습하고, 스마트폰과의 통신에 이용한다.**

● 개념설명

1. Bluetooth

- 근거리 무선 통신 기술(약 10m의 거리에서 통신)
- 스마트폰, 무선 이어폰, 웨어러블 기기 등에서 디지털 데이터를 주고 받는 기술
- 2402~2480MHz 주파수 대역, ISM 방식의 무선 시스템 사용
- FFHS(Frequency Hop Spread Spectrum, 주파수 호핑 기법)
  - 좁은 대역대에 많은 기기가 몰려 충돌이나 간섭이 생길 가능성을 방지하기 위한 기법
  - 짧은 시간에 주파수를 이동하면서 패킷을 잘게 보내는 방식

2. 블루투스의 통신 구조

- **Master-Slave 구조**
  - 1개의 블루투스 장치에 최대 7개의 장치가 동시 접속 가능
  - Master : 검색(Inquiry) 및 연결 요청(Page)을 하는 장치
  - Slave : 검색 대기(Inquiry Scan) 및 연결 대기(Page Scan)를 하는 장치
  - 상황에 따라 master와 slave는 역할을 바꿀 수 있음

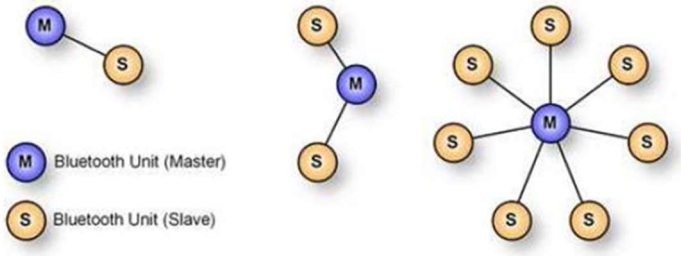

그림 1. Master-Slave 구조

- **통신 구조**

1. Inquiry(검색)
   ▷ Master가 Piconet 에 참여하기 원하는 Slave 디바이스를 탐색 후에 ID packet 과 IAC(inquiry access code)를 전송
2. Inquiry Response(검색 대기)
   ▷ Inquiry Scan 후 요청이 왔으면 FHS Packet을 리턴하여 자신의 주소를 Master에게 알림
3. Paging(연결)
   ▷ Master가 Slave 각각의 주소에 대해 paging하여 디바이스 별로 호출. Master는 DAC 를 access code로 사용
4. Paging Response(연결 대기)
   ▷ Slave 가 응답, Master가 FHS 를 Slave에게 알림

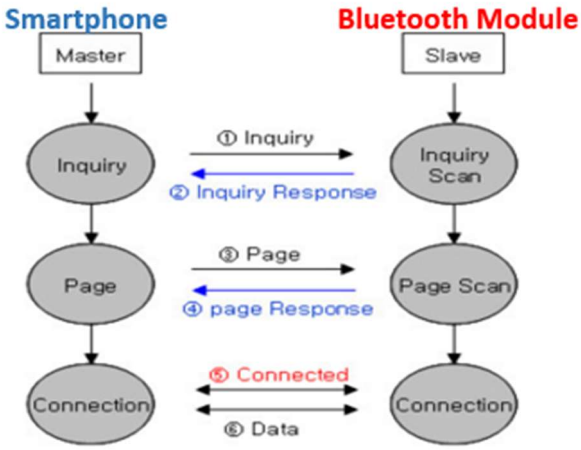

그림 2. 블루투스의 통신 구조

3. SPP(Serial Port Profile)

- 가상 시리얼 포트를 설정하고 두 개의 Bluetooth 지원 장치를 연결하는 방법 정의
  - A : 다른 디바이스에 대한 연결을 형성하도록 주도권을 취하는 장치
  - B : 연결 주도권을 쥐고 있는 장치를 기다리고 있는 장치

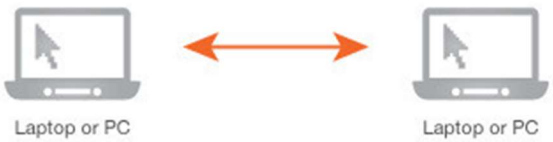

그림 3. SPP(Serial Port Profile)

4. SSID & UUID

- **SSID(Service Set Identifier)**
  - 인터넷 사용자의 무선 네트워크 이름
  - 자유롭게 이름을 설정할 수 있음
  - 주변 장치에서 무선 라우터를 식별할 수 있도록 무선 라우터가 broadcast 하는 이름
  - Wi-Fi 네트워크를 구분하는 기능 수행
- **UUID(Universally Unique Identifier)**
  - 범용 고유 번호
  - 소프트웨어 구축에 사용되는 식별자(ID) 표준
  - 128비트의 숫자들의 Hex조합 => Unique 해야함
  - 블루투스에서는 서비스의 종류를 구분하기 위해 사용

5. 블루투스 모듈(FB755AC)

- Firmtech사에서 개발한 모듈
- master와 slave 형태의 주종관계로 구성
- 하나의 모듈에 최대 7개의 slave(장치)가 동시에 접속이 가능
  - 통신 거리 : 100m, 인터페이스 : USART
  - AT 명령어를 지원, AT 명령어를 이용하여 FB755AC의 제어가 가능

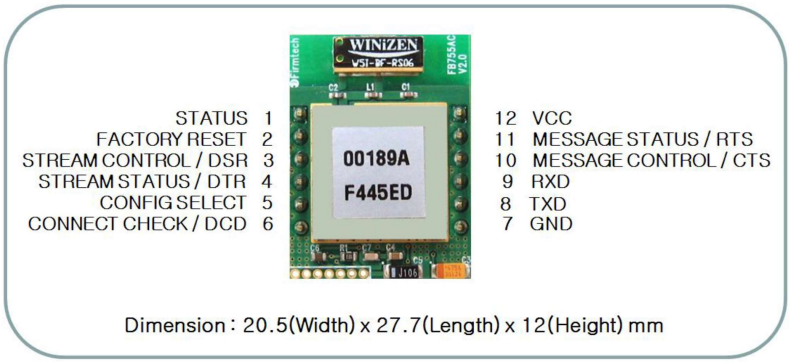

그림 4. 블루투스 모듈(FB755AC)

- STATUS PIN
  - ▷ 연결 대기 및 시도, 검색할 때 Low, High 값 반복
- STREAM CONTROL, STREAM_STATUS, MESSAGE_CONTROL, MESSAGE_STATUS
  - ▷ 1:N 통신을 위한 연결
  - ▷ 1:1 통신시 사용 X
- CONFIG SELECT
  - ▷ 연결 블루투스 모듈 설정 시 사용, HIGH 를 입력한 채로 전원을 켜면 설정 모드
- CONNECT CHECK/DCD
  - ▷ 설정된 연결 수 만큼 Master 연결 시 Low, 하나라도 해지되면 High

- **AT 명령어**
  - 모뎀 모듈을 제어하는데 쓰이는 명령어
  - AT 명령어 set을 통해 FB755AC 모듈을 제어 가능
  - AT+BTSCAN : 블루투스 모듈이 모바일 디바이스(스마트폰)과 연결 및 검색할 수 있도록 하기 위한 명령어

● 실험 진행

● 코드 작성

- 코드1에서는 USART1_TX, USART1_RX에 설정된 pin 이 PA9, PA10이기 때문에 아래와 같이 세팅

```
/* 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_IPU | GPIO_Mode_IPD;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

코드 1 USART pin 세팅

- 코드2에서는 USART2_TX, USART2_RX에 설정된 pin 이 PD5, PD6이기 때문에 아래와 같이 세팅

```
/* USART2 pin setting */

GPIO_PinRemapConfig(GPIO_Remap_USART2,ENABLE);
  //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_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU | GPIO_Mode_IPD;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

코드 2 USART2 pin 세팅

- <코드3> , <코드4> 에서는 'USART_InitTypeDef' 구조체와 함수 'USART_Init'를 사용해 USART1과 USART2를 각각 초기화

```
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);
```

코드 4 USART1 초기화

```
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);
```

코드 3 USART2 초기화

- 데이터가 수신될 때마다 interrupt가 발생하므로 아래 <코드5>, <코드6>과 같이 USART1과 USART2의 RX interrupt를 활성화

```
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
```

코드 5 Enable USART1 RX interrupt

```
USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
```

코드 6 Enable USART2 RX interrupt

- <코드7>은 USART1에서 수신된 데이터를 읽고 이를 USART2로 전송하고 interrupt를 관리

```
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);
    }
}
```

코드 7 USART1 IRQ Handler

- <코드8>은 USART2에서 수신된 데이터를 읽고 이를 USART1로 전송하고 interrupt를 관리

```
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);
    }
}
```

코드 8 USART2 IRQ Handler

● 실습 결과

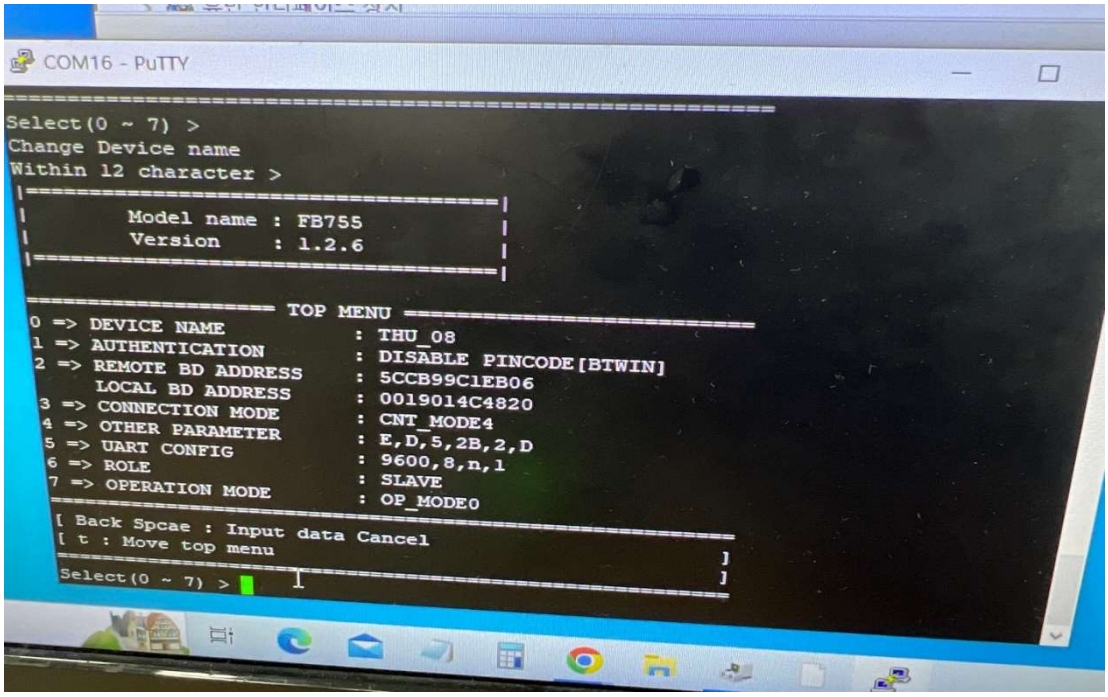

그림 5. 장치 이름 설정 후 화면

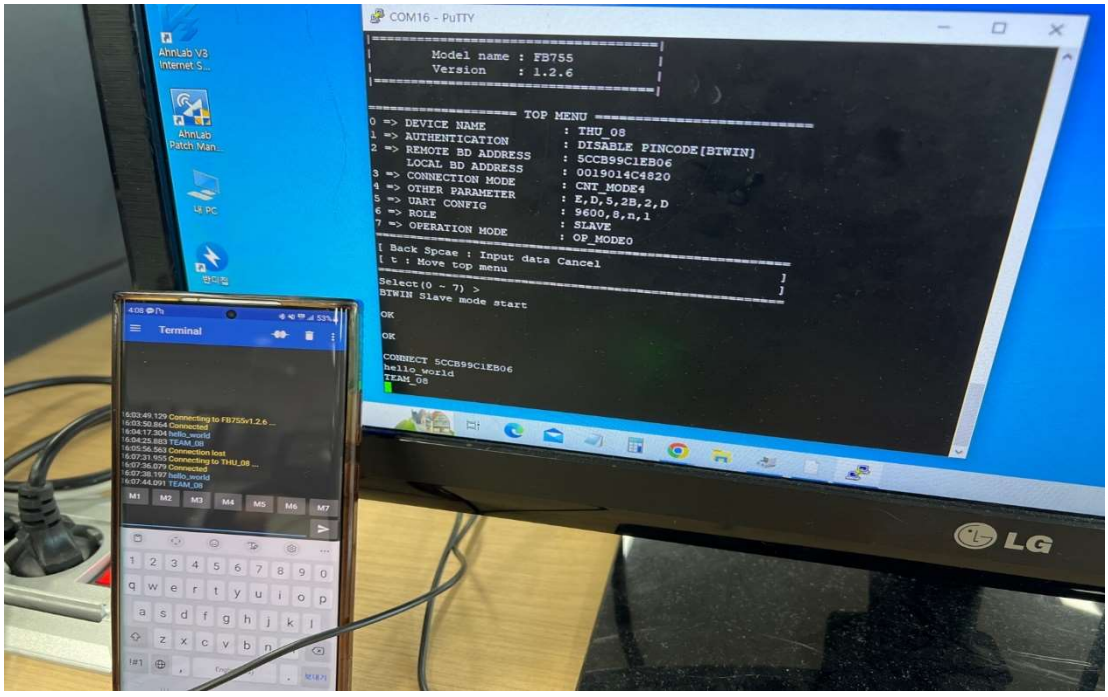

그림 7. Putty의 통신 화면

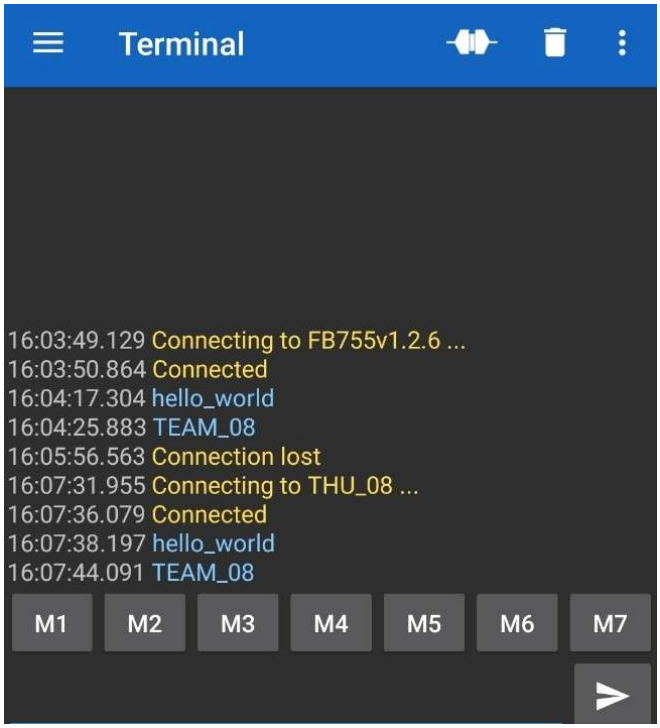

그림 6. 휴대폰에서 Putty로 보낸 메세지

● 결론

- 라이브러리를 활용해 직접 메모리의 주소로 접근하는 것 보다 더 직관적이고 간단하게 코드를 짤 수 있다.

- 시스템 및 사용자 Clock 을 별도로 설정할 수 있으며, 설정하는 방법을 알게 되었고, 시리얼케이블 연결 및 Putty를 통한 USART 통신에 대해 알게 되었다.