2021년 2학기
임베디드 시스템설계 및 실험
9주차 실험보고서

201712159 조현우
201724566 전승윤
201724611 최호진
201924650 박지호

목차

1. 실험 목적

2. 실험 과정

3. 실험 세부 내용

4. 실험 결과

1. 실험 목적

UART 통신을 활용한 Bluetooth동작
기판 납땜

2. 실험 과정

1) 만능 기판 납땜
2) PC의 putty 프로그램과 UART 통신을 하기 위해 펌웨어 코드 작성

3. 실험 세부 내용

1) 만능 기판 납땜

블루투스 모듈과 M3보드를 연결하기 위해서 만능 기판에 전선을 납땜해준다. 블루투스 모듈의 RX, TX와 M3보드의 USART2 RX, TX의 교차 연결을 위한 전선을 납땜하고 연결 여부를 확인하기 위한 LED, 저항, GND등도 납땜을 해준다.

2) Putty 프로그램과 UART통신을 하기 위해 펌웨어 코드 작성

먼저 사용할 Port의 RCC enable과 GPIO configuration이 필요하다.

```
/* UART TX/RX port clock enable */
/* PA9, 10 (UART1) PA2, 3(UART2) */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

먼저 putty와 M3보드 간의 통신을 위한 PORT A (9, 10) 과 M3보드와 블루투스 모듈간의 통신을 위한 PORT A (2, 3)을 RCC ENABLE 을 시켜준다.

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

Alternate Function IO clock enable을 위한 AFIO enable을 해야한다.
지난 주차와 동일하게 헤더 파일에 정의된 구조체와 함수를 사용한다.

```
/* UART1 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);

/* UART2 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_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

UART1과 UART2핀의 GPIO setting코드이다.
UART1의 TX는 UART2의 RX와 교차 연결을 해야하고 UART1의 RX도 마찬가지로 UART2의 TX과 교차 연결을 해주어야 한다.

미리 정의된 GPIO_InitStructure 변수를 통해서 사용할 PORT의 핀 번호와 GPIO_Speed, GPIO_Mode를 부여해 GPIO_Init()의 인자로 넣어 원하는 설정을 한다.

```
USART_InitTypeDef USART1_InitStructure;

// Enable the USART1 peripheral
USART_Cmd(USART1, ENABLE);
    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_Tx | USART_Mode_Rx;
    USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_Init(USART1, &USART1_InitStructure);

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

실질적인 통신을 위해 UART1과 UART2의 BaudRate와 Mode 등을 설정해 주는 작업이다.
UART1 설정과 UART2 설정은 USART1을 입력하는 부분만 USART2로 바뀐 것 이외에 모두 동일하다.

지난 주차와 설정 값도 모두 동일하다.

그리고 지난 주차와 동일하게 NVIC 를 이용해 GPIO 인터럽트 핸들링 세팅이 필요하다.

```
// 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' is only required for USART setting
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);
```

NVIC는 지난 주차와 동일하다. 이번 실험에서도 사용될 동작들은 동시에 일어날 시나리오를 따로 가정하지 않았기 때문에 모두 같은 PreemptionPriority와 Subpriority를 설정해 주었다.

```
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);

        sendDataUART2(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 USART1 peripheral
        word = USART_ReceiveData(USART2);

        sendDataUART1(word);

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

UART1에서 입력이 들어오면 USART1_IRQHandler()가 동작한다.
즉, putty를 통해서 PC에 문자를 입력하는 등의 인터럽트가 발생하면 해당 메시지를 받아 word 변

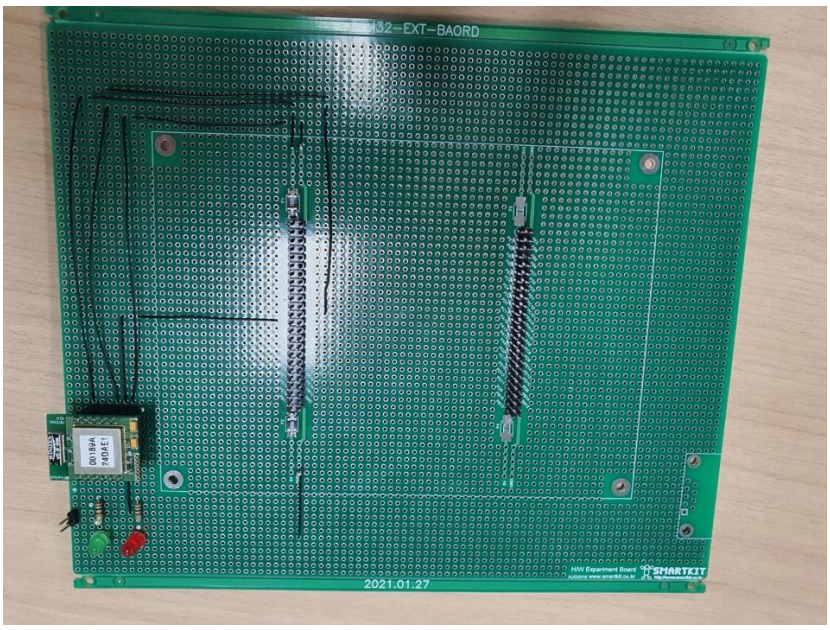
수에 저장한 후 UART2로 연결된 블루투스 기기로 받은 메시지를 그대로 전달한다.

휴대폰에 설치된 블루투스 터미널에 메시지를 입력하는 경우도 같다. UART2의 메시지를 받아 word에 저장한 후 UART1으로 연결된 PC의 putty로 받은 메시지를 그대로 전달한다.

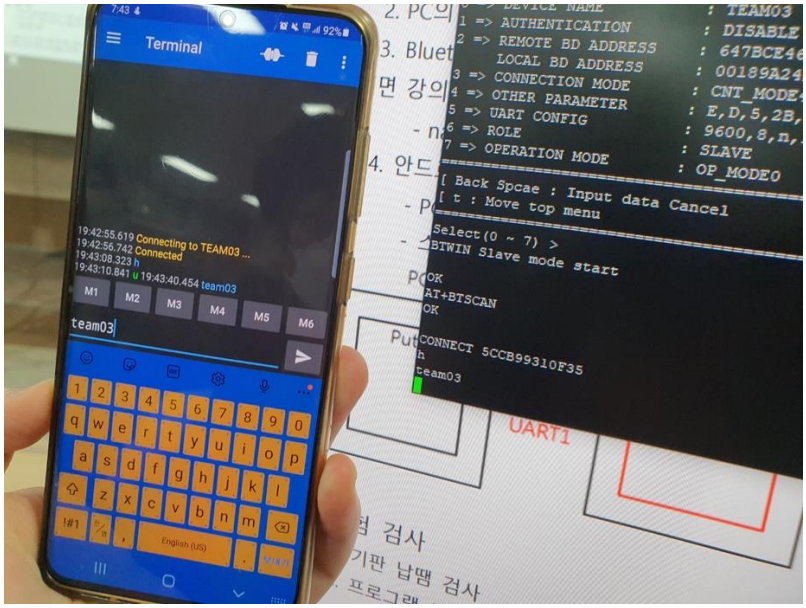
이번 실험은 따로 버튼을 통해서 보드를 제어하는 상황이 없기 때문에 main 함수의 while문 내부에는 아무것도 필요하지 않다. Main 함수가 종료되면 UART통신도 종료되기 때문에 While문은 단순히 main 함수가 종료되지 않도록 해주는 역할을 하게 된다.

마지막으로 블루투스 모듈의 CONFIG SELECT에 점프선으로 3v3을 연결한 후 putty에 접속을 한다. 보드의 전원을 껏다 킨 후 Device Name과 PinCode, Connection Mode, Uart config(9600, 8, n, 1)을 설정한다. 3v3 점프선을 제거한 후 AT+BTSCAN을 입력하여 연결을 마무리한다.

4. 실험 결과



만능 기판에 납땜을 완료하고 블루투스 모듈 및 LED, 저항을 연결한 모습니다.



Play store에서 블루투스 시리얼 터미널을 설치한 후 블루투스를 검색해보니 3v3 점프선을 연결해서 설정했던 Device Name으로 기기를 발견할 수 있었다. 기기를 연결하고 휴대폰과 putty 에서 각각 문자를 전송하면 서로 전송한 문자를 즉시 확인할 수 있었다.