

임베디드 시스템 설계 및 실험 보고서

7 주차 실험_Bluetooth 및 납땜

분반 : 001 분반

교수님 : 정 상화 교수님

조교님 : 유 동화 조교님

실험일 : 2019-10-14

제출일 : 2019-10-21

00. 목차

- 01. 실험 목적 ... p.2
- 02. 실험 과제 ... p.2
- 03. 실험 준비 ... p.2
- 04. 실험 및 과제 해결 ... p.8
- 05. 실험 결과 ... p.13
- 06. 결론 ... p.17

7 조

장 수현

박 창조

임 다영

이 힘찬

01. 실험 목적

- Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
- 라이브러리 함수 사용법 숙지
- 블루투스 통신 모듈 사용법 숙지
- 모듈 사용을 위한 기판 납땜 방법 숙지

02. 실험 과제

2.1 주 과제

PC와 스마트폰 간에 블루투스를 활용한 채팅

- PC의 Putty에서 입력 시 PC의 Putty와 스마트폰의 Terminal APP에서 출력
- 스마트폰의 Terminal APP에서 입력 시 PC의 Putty로 출력

2.2 세부 과제

- 개발 환경 구축
- DS-5에서 프로젝트 생성 및 설정
- DB 파일, 라이브러리, scatter 파일, flashclear 파일을 프로젝트 폴더 안으로 복사
- 블루투스 통신을 위한 납땜

03. 실험 준비

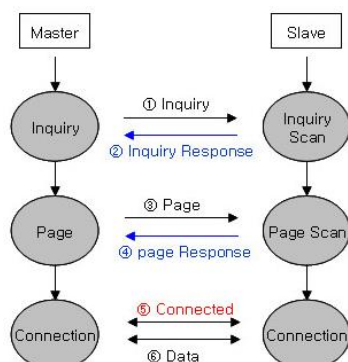
3.1 실험에 필요한 기초지식

3.1.1 Bluetooth

- 휴대기기 등을 서로 연결해 통신 근거리 무선 기술로 2.400~2.4835GHz 주파수를 사용한다.
- 10~100m의 데이터 전송거리를 가지는 근거리 통신이다.
- 좁은 대역대로 인한 충돌, 간섭을 피하기 위해 FHSS를 사용한다.

3.1.2 Bluetooth 통신 구조

- 블루투스는 기본적으로 Master와 Slave인 주종의 역할로 동작한다.
- Inquiry와 Page를 하는 쪽을 Master라고 한다.
- Inquiry Scan 및 Page Scan을 하는 쪽을 Slave라고 한다.
- Master가 주변의 Slave를 찾으면 Slave는 자신의 정보를 Master에게 송신한다.
- Slave의 정보가 Master와 일치하면 상호 연결이 이루어지면, 데이터 전송이 가능하다.



3.1.3 Bluetooth 프로파일

- 어떤 종류의 데이터를 보내는 지 명확하게 정의하기 위한 블루투스의 프로토콜이다.
- HSP, HID, SPP, A2DP 등의 종류가 있으며 실험에서 사용한 것은 SPP 이다.

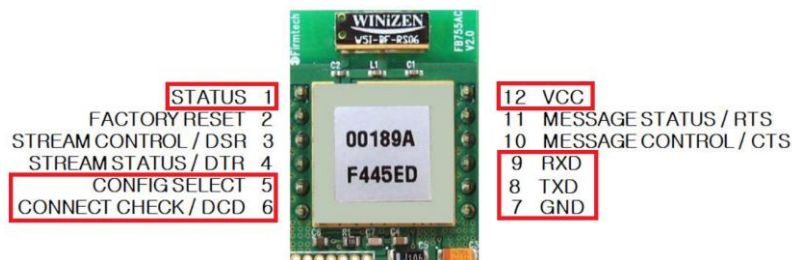
3.1.4 Bluetooth SSID

- Wi-Fi 네트워크 이름이다.
- 주변 장치에서 무선 라우터를 식별할 수 있도록 무선 라우터가 broadcast 하는 이름이다.

3.1.5 Bluetooth UUID

- 범용 고유 번호라고 불리며, 128 비트의 숫자들을 조합한다.
- 범용적으로 사용할 수 있는 고유 ID 를 사용하기 위해 생성된다.

3.1.6 FB755AC 모듈



실험에서 사용하는 핀은 1, 5 ~ 9, 12 번이다.

- 1 번 : 모듈의 상태를 모니터링. 검색 대기/연결 대기/주변 블루투스 장치 검색 시 LOW, HIGH 를 반복.
- 5 번 : 5 번 핀에 High 신호(3.3V) 인가한 상태에서 모듈에 전원 인가 시 설정 모드 진입.
- 6 번 : Slave 에 설정된 장치의 수만큼 Master 와 연결되면 LOW 상태로 변경(기본 High).
- 7 번 : 접지, 보드의 GND 와 연결.
- 8 번 : UART 데이터 출력. 보드의 수신부와 연결.
- 9 번 : UART 데이터 입력. 보드의 송신부와 연결.
- 12 번 : 3.3V 전원 인가..

3.1.7 AT 명령어

- 헤이즈 마이크로컴퓨터 사가 개발한 헤이즈 스마트 모뎀과 그 호환 모뎀을 제어하기 위해 사용되는 명령어로 현재에는 사실상 거의 모든 모뎀의 표준 명령어이다.
- AT 란 준비라는 뜻을 가진 Attention 의 앞 글자인 AT 에서 따온 말로 헤이즈 명령어는 AT + Command 와 같이 매우 간단한 명령어 구조를 가진다.

3.2 납땜 시 주의사항

- 선을 이을 시에 최대한 당겨서 납땜한다.
- 블루투스 모듈의 Tx, Rx 는 보드 UART 의 Tx, Rx 와 반대로 연결한다.
- 블루투스 모듈을 핀 소켓에 끼운 채로 납땜하지 않도록 한다.
- 멀티미터를 사용하여 연결한 선이 정상적으로 신호가 통하는지 확인한다.
- LED 를 납땜 전에 보드에 꽂아서 불이 정상적으로 점등되는 지 확인한다.
- 인두기의 끝이 더러워지지 않게 인두 팁 크리너를 잘 사용한다.

- 핀 소켓에 인두를 오래 접촉하지 않는다.
- 핀 헤더에 인두를 오래 접촉하지 않는다.
- 만능 기판에 인두를 오래 접촉하지 않는다.
- 선을 만능기판에 뜨개질할 때 선을 수직으로 납땜한다.
- 납땜 중, 완료 후에는 청소를 철저히 하고, 인두기를 다 쓰면 전원을 끈다.

3.3 환경 설정

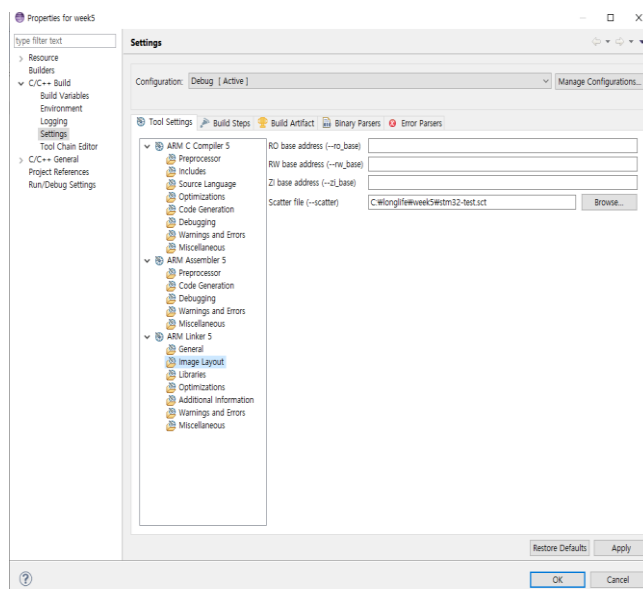
실험을 진행하기 전 아래의 프로젝트 생성 및 기본 환경 설정을 해주어야 한다.

3.3.1 프로젝트 생성

C 언어로 소스코드를 작성하기 때문에 C++프로젝트가 아닌 C 프로젝트를 생성해준다. Project type 은 Bare-metal Executable -> Empty Project 로, Toolchains 은 Arm Compiler 5 로 설정한다.

3.3.2 프로젝트 Properties-Settings

4 주차 실험과 동일하게 스캐터 파일을 이용한다.



3.3.3 보드 연결

보드 연결 시에는 반드시 연결 순서를 지키고, 규격에 맞는 전원선을 사용해야한다.

연결 순서 : 보드와 Dstream JTAG 연결 -> 보드전원선 연결(보드 전원은 OFF) -> Dstream 전원 연결 및 ON -> Dstream Status LED 점등 확인 후 보드 전원 ON -> Dstream Target LED 점등 확인 후 DS-5에서 'connect target'

3.3.4 데이터 베이스 설정

Dstream 를 USB 로 컴퓨터에 연결하고 Debug Hardware config 에서 보드를 connect 한다. 보드 연결 후에는 Auto Configure 를 클릭하여 설정을 진행하고, rvc 파일로 저장한다. rvc 파일 저장 경로에 한글이 들어가지 않도록 주의한다.

3.3.5 cdbimporter

DS-5 Command Prompt 에서 RVC 파일이 있는 폴더로 이동 후 아래와 같이 명령문을 작성한다.

```

DS-5 Command Prompt - cmdsuite.exe
Environment configured for ARM DS-5 (build 5180018)
Please consult the documentation for available commands and more details

C:\Program Files\DS-5\bin>cd #
C:\Program Files\DS-5\bin>cd C:\longlife\week5\PSU_DB\Boards\PSU\STM32F107VCT6
C:\longlife\week5\PSU_DB\Boards\PSU\STM32F107VCT6>pwd
'pwd'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
C:\longlife\week5\PSU_DB\Boards\PSU\STM32F107VCT6>cdbinimporter -t C:\longlife\week5\PSU_DB\Boards\PSU\STM32F107VCT6_ST
M32F107VCT6.rvc
DS-5 Config Database Import Utility v1.2
Copyright 2011-2014 ARM Ltd

Reading C:\longlife\week5\PSU_DB\Boards\PSU\STM32F107VCT6\STM32F107VCT6.rvc
Enter DS-5 source configuration path
(the location of the database that contains the necessary data to identify the target)
[default: 'C:\Program Files\DS-5\sw\debugger\configdb'] >

Found 1 ARM core
Import Summary -
ID Name Definition Associated TCF files
-----
2 Cortex-M3 Cortex-M3 <none>

Select a core to modify (enter its ID and hit return) or press enter to continue. []

Enter Platform Manufacturer
[default: 'Imported'] >hi

Enter Platform Name
[default: 'STM32F107VCT6'] >yo

Building configuration XML...
Creating database entry...

DTS script assumptions:
The Cortex-M3 cores are using trace sources of type ETMv3.4.
All ETM devices occur after the core definitions in the RVG file.
The ETMv3.4 devices for the Cortex-M3 cores are already unlocked.

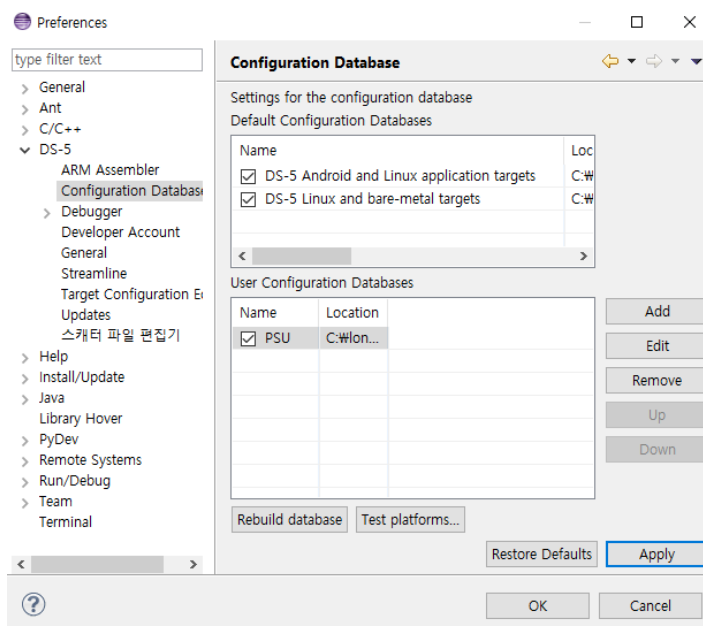
Import successfully completed

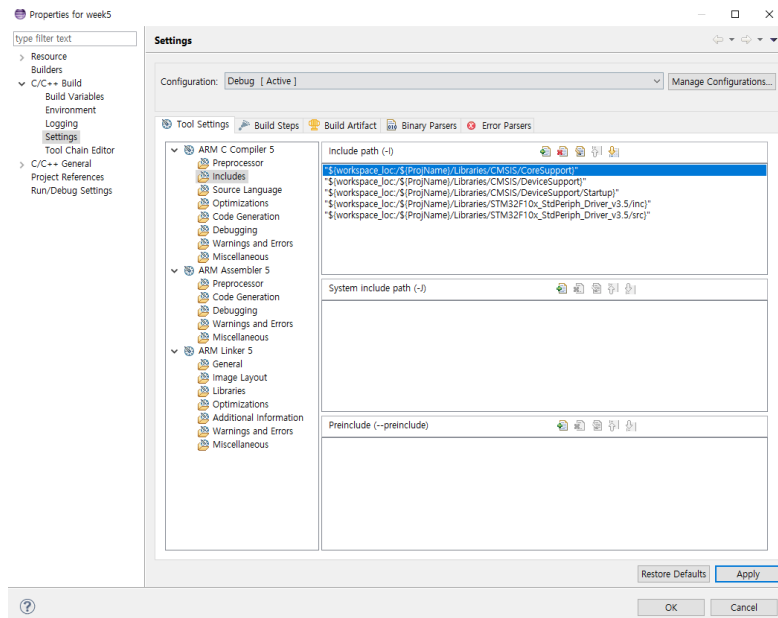
The new platform will not be visible in the DS-5 Debugger until the destination database
has been added to the "User Configuration Databases" list and the database has been rebuilt.
A rebuild is done either when DS-5 is (re)started, a user configuration database is added or
by forcing a database rebuild.
To force a rebuild or add a database, select the "Window -> Preferences" menu item,
then expand the DS-5 group. To rebuild, select "Configuration Database", then press
the "Rebuild database..." button.
To add a database to the "User Configuration Databases" list, click the "Add" button
and supply a suitable "Name" (E.g. Imported) and "Location" for the database.

```

3.3.6 데이터 베이스 등록 및 라이브러리 추가

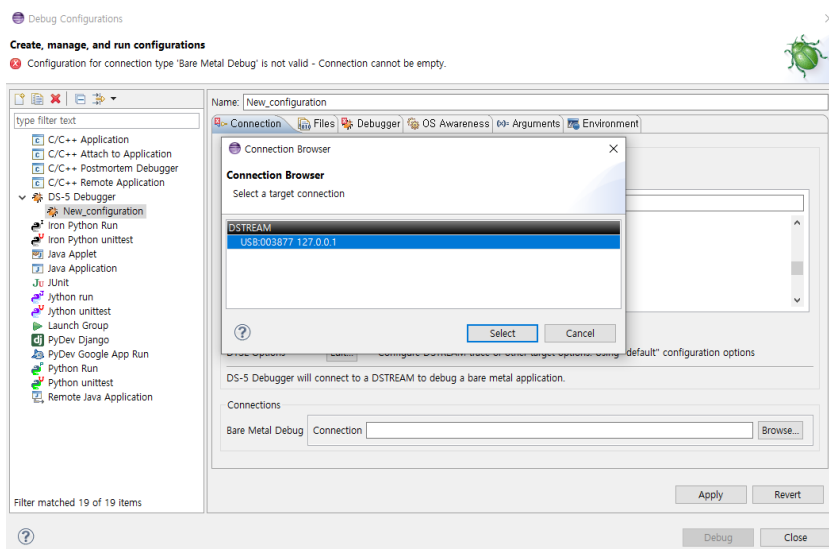
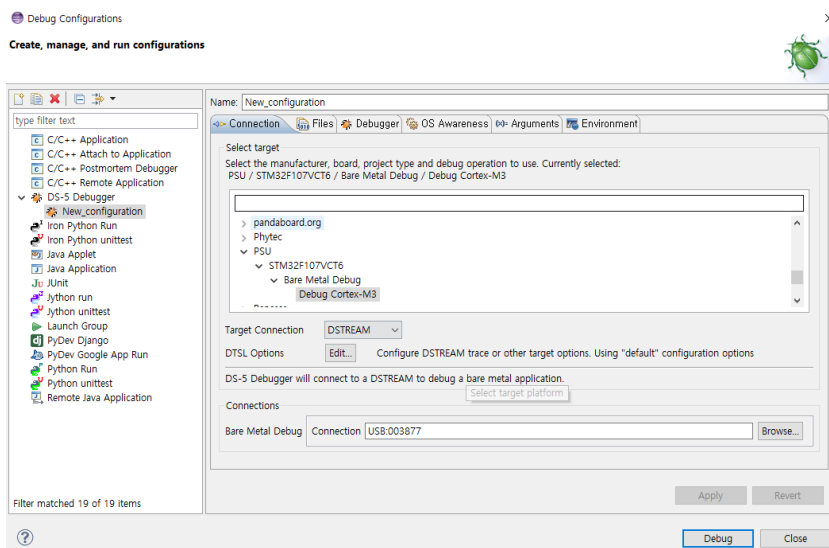
아래와 같이 데이터 베이스 등록 후 프로젝트 폴더에 라이브러리를 추가한다.

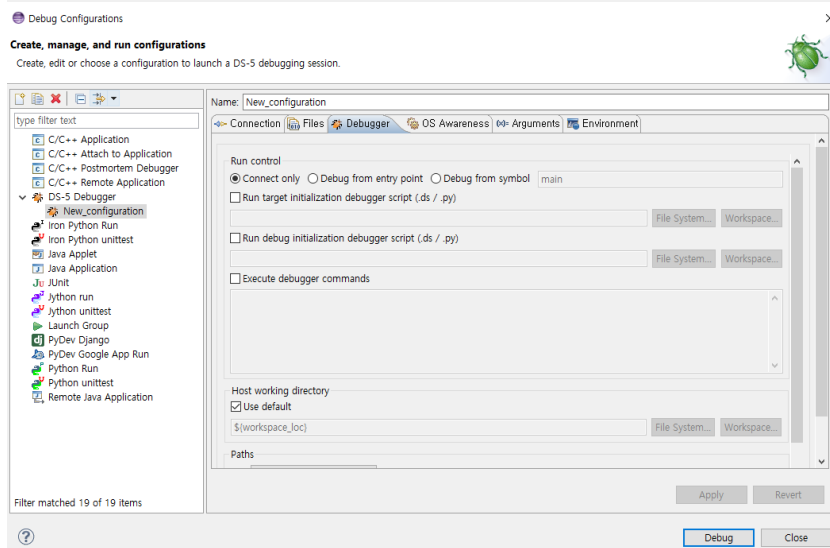
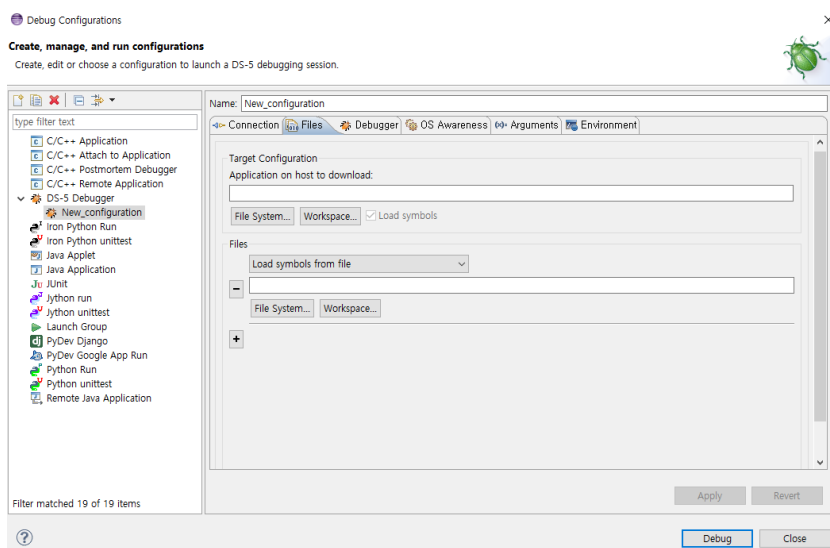
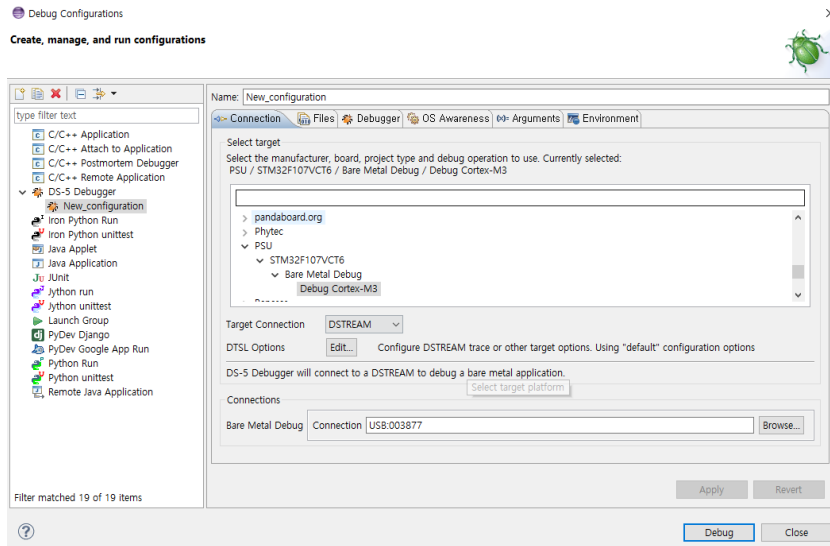




3.3.7 디버그 설정

C 언어 소스파일을 만들어 빌드하고 디버그 설정을 한다. 4 주차와 동일한 환경으로 진행한다.





04. 실험 및 과제 해결

4.1 RCC Configuration

```
void RCC_Configure() {
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_USART1, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
}
```

- USART1, USART2, USART(포트 A) 를 모두 ENABLE 해준다.

4.2 GPIO Configuration

```
void GPIO_Configure() {

    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;    //PA9
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;   //PA10
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

```
typedef struct
{
    uint16_t GPIO_Pin;           /*!< Specifies the GPIO pins to be configured.
                                   This parameter can be any value of @ref GPIO_pins_define */

    GPIOSpeed_TypeDef GPIO_Speed; /*!< Specifies the speed for the selected pins.
                                   This parameter can be a value of @ref GPIOSpeed_TypeDef */

    GPIOMode_TypeDef GPIO_Mode;  /*!< Specifies the operating mode for the selected pins.
                                   This parameter can be a value of @ref GPIOMode_TypeDef */
}GPIO_InitTypeDef;
```

- USART1, 2 에 대해서 Input, Output 을 각각 설정한다(Rx, Tx).

4.3 USART Configuration

```
void USART_Configure() {
    USART_InitTypeDef USART_InitStructure;
    USART_InitTypeDef USART2_InitStructure;

    /*TODO: USART1 configuration*/
    USART_InitStructure.USART_BaudRate = 9600;
    USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;

    USART2_InitStructure.USART_BaudRate = 9600;
    USART2_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
    USART2_InitStructure.USART_Parity = USART_Parity_No;
    USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART2_InitStructure.USART_StopBits = USART_StopBits_1;

    USART_Init(USART1, &USART_InitStructure);
    USART_Init(USART2, &USART2_InitStructure);

    /*TODO: USART1 cmd ENABLE*/
    USART_Cmd(USART1, ENABLE);
    USART_Cmd(USART2, ENABLE);

    /*TODO: USART1 IT Config*/
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
}
```

```
typedef struct
{
    uint32_t USART_BaudRate;          /*!< This member configures the USART communication baud rate.
                                         The baud rate is computed using the following formula:
                                         - IntegerDivider = ((PCLKx) / (16 * (USART_InitStruct->USART_BaudRate)))
                                         - FractionalDivider = ((IntegerDivider - ((u32) IntegerDivider)) * 16) + 0.5 */

    uint16_t USART_WordLength;        /*!< Specifies the number of data bits transmitted or received in a frame.
                                         This parameter can be a value of @ref USART_Word_Length */

    uint16_t USART_StopBits;          /*!< Specifies the number of stop bits transmitted.
                                         This parameter can be a value of @ref USART_Stop_Bits */

    uint16_t USART_Parity;             /*!< Specifies the parity mode.
                                         This parameter can be a value of @ref USART_Parity
                                         @note When parity is enabled, the computed parity is inserted
                                              at the MSB position of the transmitted data (9th bit when
                                              the word length is set to 9 data bits; 8th bit when the
                                              word length is set to 8 data bits). */

    uint16_t USART_Mode;               /*!< Specifies whether the Receive or Transmit mode is enabled or disabled.
                                         This parameter can be a value of @ref USART_Mode */

    uint16_t USART_HardwareFlowControl; /*!< Specifies whether the hardware flow control mode is enabled
                                         or disabled.
                                         This parameter can be a value of @ref USART_Hardware_Flow_Control */
} USART_InitTypeDef;
```

- 과제에 제시한 설정대로 Parity_No, Wordlength 는 8bit, HardwareFlowControl 은 None 으로 설정했다.
- Baudrate, Stopbits 는 default 로 9600, 1 비트를 주었다.
- 이번 실험은 Rx, Tx 모두 사용하므로 모드에 두 값을 모두 주었다.
- 그 후 USART1, 2 모두에 대해서 Usart Init, Cmd Enable, IT configuration 을 순서대로 진행했다.

4.4 NVIC Configuration

```
void NVIC_Configure() {
    /*TODO: NVIC_configuration */

    NVIC_InitTypeDef NVIC_USART_InitTD;
    NVIC_InitTypeDef NVIC_USART2_InitTD;

    NVIC_USART_InitTD.NVIC_IRQChannel = USART1_IRQn;
    NVIC_USART2_InitTD.NVIC_IRQChannel = USART2_IRQn;

    NVIC_USART_InitTD.NVIC_IRQChannelCmd = ENABLE;
    NVIC_USART2_InitTD.NVIC_IRQChannelCmd = ENABLE;

    NVIC_USART_InitTD.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_USART2_InitTD.NVIC_IRQChannelPreemptionPriority = 0;

    NVIC_USART_InitTD.NVIC_IRQChannelSubPriority = 0;
    NVIC_USART2_InitTD.NVIC_IRQChannelSubPriority = 0;

    NVIC_Init(&NVIC_USART_InitTD);
    NVIC_Init(&NVIC_USART2_InitTD);}
```

```

#ifdef STM32F10X_LD
    ADC1_2_IRQn      = 18,      /*!< ADC1 and ADC2 global Interrupt
    USB_HP_CAN1_TX_IRQn = 19,      /*!< USB Device High Priority or CAN1 TX Interrupts
    USB_LP_CAN1_RX0_IRQn = 20,      /*!< USB Device Low Priority or CAN1 RX0 Interrupts
    CAN1_RX1_IRQn     = 21,      /*!< CAN1 RX1 Interrupt
    CAN1_SCE_IRQn      = 22,      /*!< CAN1 SCE Interrupt
    EXTI9_5_IRQn       = 23,      /*!< External Line[9:5] Interrupts
    TIM1_BRK_IRQn      = 24,      /*!< TIM1 Break Interrupt
    TIM1_UP_IRQn       = 25,      /*!< TIM1 Update Interrupt
    TIM1_TRG_COM_IRQn  = 26,      /*!< TIM1 Trigger and Commutation Interrupt
    TIM1_CC_IRQn       = 27,      /*!< TIM1 Capture Compare Interrupt
    TIM2_IRQn          = 28,      /*!< TIM2 global Interrupt
    TIM3_IRQn          = 29,      /*!< TIM3 global Interrupt
    I2C1_EV_IRQn       = 31,      /*!< I2C1 Event Interrupt
    I2C1_ER_IRQn       = 32,      /*!< I2C1 Error Interrupt
    SPI1_IRQn          = 35,      /*!< SPI1 global Interrupt
    USART1_IRQn        = 37,      /*!< USART1 global Interrupt
    USART2_IRQn        = 38,      /*!< USART2 global Interrupt
    EXTI15_10_IRQn     = 40,      /*!< External Line[15:10] Interrupts
    RTCAlarm_IRQn      = 41,      /*!< RTC Alarm through EXTI Line Interrupt
    USBWakeUp_IRQn     = 42,      /*!< USB Device WakeUp from suspend through EXTI Line Ir
#endif /* STM32F10X_LD */

#ifdef STM32F10X_LD_VL
    ADC1_IRQn          = 18,      /*!< ADC1 global Interrupt
    EXTI9_5_IRQn       = 23,      /*!< External Line[9:5] Interrupts
    TIM1_BRK_TIM15_IRQn = 24,      /*!< TIM1 Break and TIM15 Interrupts
    TIM1_UP_TIM16_IRQn = 25,      /*!< TIM1 Update and TIM16 Interrupts
    TIM1_TRG_COM_TIM17_IRQn = 26,      /*!< TIM1 Trigger and Commutation and TIM17 Interrupt
    TIM1_CC_IRQn       = 27,      /*!< TIM1 Capture Compare Interrupt
    TIM2_IRQn          = 28,      /*!< TIM2 global Interrupt
    TIM3_IRQn          = 29,      /*!< TIM3 global Interrupt
    I2C1_EV_IRQn       = 31,      /*!< I2C1 Event Interrupt
    I2C1_ER_IRQn       = 32,      /*!< I2C1 Error Interrupt
    SPI1_IRQn          = 35,      /*!< SPI1 global Interrupt
    USART1_IRQn        = 37,      /*!< USART1 global Interrupt
    USART2_IRQn        = 38,      /*!< USART2 global Interrupt
    EXTI15_10_IRQn     = 40,      /*!< External Line[15:10] Interrupts
    RTCAlarm_IRQn      = 41,      /*!< RTC Alarm through EXTI Line Interrupt
    CEC_IRQn           = 42,      /*!< HDMI-CEC Interrupt
    TIM6_DAC_IRQn      = 54,      /*!< TIM6 and DAC underrun Interrupt
    TIM7_IRQn          = 55,      /*!< TIM7 Interrupt
#endif /* STM32F10X_LD_VL */

typedef struct
{
    uint8_t NVIC_IRQChannel;          /*!< Specifies the IRQ channel to be enabled or disabled.
                                     This parameter can be a value of @ref IRQn_Type
                                     (For the complete STM32 Devices IRQ Channels List, please
                                     refer to stm32f10x.h file) */

    uint8_t NVIC_IRQChannelPreemptionPriority; /*!< Specifies the pre-emption priority for the IRQ channel
                                     specified in NVIC_IRQChannel. This parameter can be a value
                                     between 0 and 15 as described in the table @ref NVIC_Priority_Table */

    uint8_t NVIC_IRQChannelSubPriority; /*!< Specifies the subpriority level for the IRQ channel specified
                                     in NVIC_IRQChannel. This parameter can be a value
                                     between 0 and 15 as described in the table @ref NVIC_Priority_Table */

    FunctionalState NVIC_IRQChannelCmd; /*!< Specifies whether the IRQ channel defined in NVIC_IRQChannel
                                     will be enabled or disabled.
                                     This parameter can be set either to ENABLE or DISABLE */
} NVIC_InitTypeDef;

```

- 인터럽트에 대해서 우선순위를 정해야 한다. NVIC 가 이런 역할을 한다.
- 먼저 설정했던 핀소스 번호에 맞는 채널들을 설정해주고 채널 커맨드는 Enable 한다.
- PreemptionPriority 가 주 우선순위고, Subpriority 가 보조 우선순위라고 생각하면 된다.
- 위와 같이 모두 0 을 주면 세 인터럽트 사이에 우선순위는 없는 것이나 마찬가지이다.

4.5 Interrupt Handler

```
/*TODO: IRQHandler */

void USART1_IRQHandler(void) {
    char c;

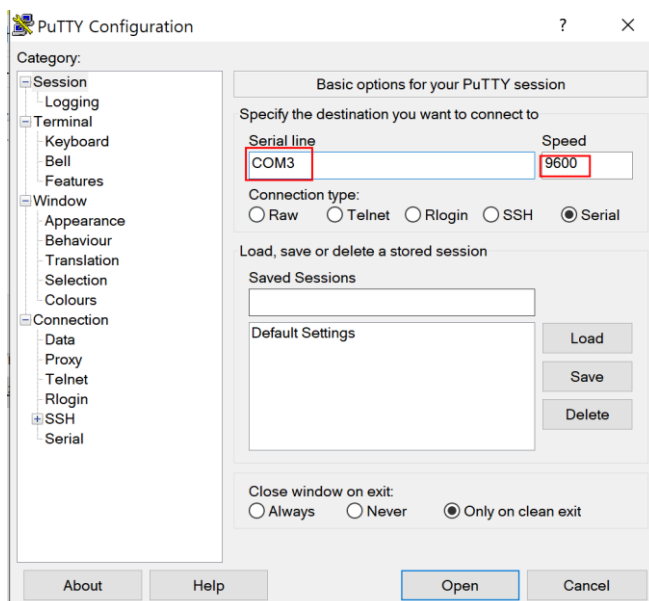
    if(USART_GetITStatus(USART1, USART_IT_RXNE ) != RESET) {
        c = USART_ReceiveData(USART1);
        USART_SendData(USART1, c);
        USART_SendData(USART2, c);
        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
    }
}

void USART2_IRQHandler(void) {
    char c;

    if(USART_GetITStatus(USART2, USART_IT_RXNE ) != RESET) {
        c = USART_ReceiveData(USART2);
        USART_SendData(USART1, c);
        USART_ClearITPendingBit(USART2,USART_IT_RXNE);
    }
}
```

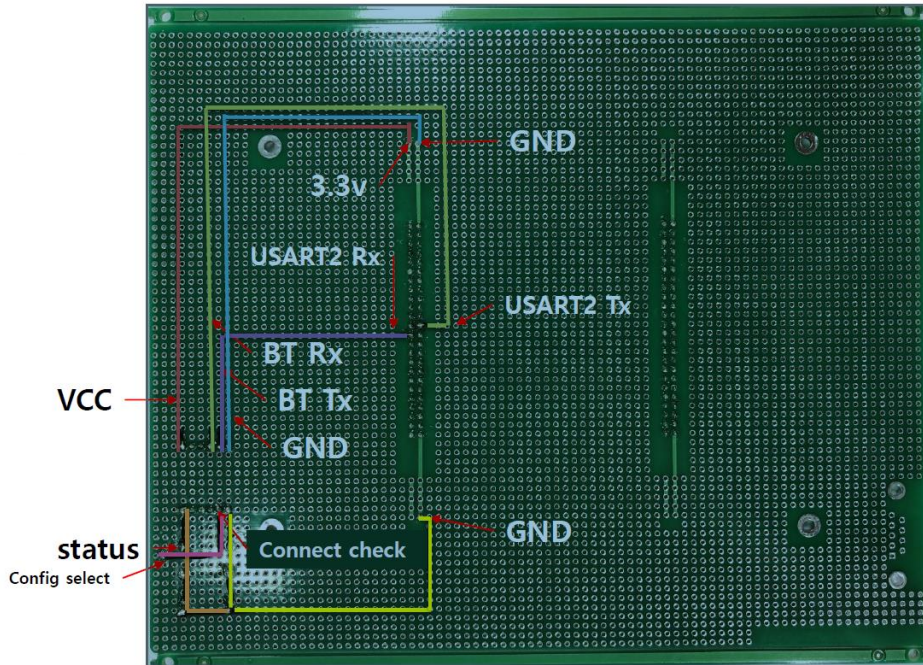
- 인터럽트는 모두 인터럽트 핸들러를 통해 처리가 가능하다.
- USART1 의 경우 Putty 에서 USART1 으로 데이터를 입력하면 USART1 과 USART2 를 통해 각각 Putty 와 블루투스 통신화면에 출력한다.
- USART2 의 경우 블루투스 통신화면에 데이터를 입력하면 USART1 으로 데이터를 보내 Putty 화면에 데이터를 출력한다.

4.6 Putty setting



UART 통신을 사용하여 Putty 에 문자열을 출력하기 위해 Putty 를 다운로드하고 위와 같이 설정해준다. Serial Line 에는 보드와 연결된 COM3 port 를 설정해주고, Speed 에는 baud rate 인 9600 을 설정한다. Serial 통신을 할 것이기 때문에 Connection type 은 Serial 로 설정한다.

4.7 납땜



블루투스 통신 모듈을 보드와 연결하기 위해 위의 매뉴얼을 참고하여 납땜을 진행한다. 이 때 납땜 시 주의 사항을 잘 지키면서 진행해야 한다. 중간중간에는 멀티미터로 점검해가며 진행하도록 한다.

05. 실험 결과

5.1 소스코드

위의 내용을 바탕으로 작성한 전체 소스코드는 아래와 같다.

```
/*
flash load "C:\longfile\week07\flashclear.axf"
flash load "C:\longfile\week07\Debug\week07.axf"
*/

#include "misc.h"
#include "core_cm3.h"
#include "stm32f10x.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"

void RCC_Configure() {

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA| RCC_APB2Periph_USART1, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
}
```

```

void GPIO_Configure() {

    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;    //PA9
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;    //PA10
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

}

void USART_Configure() {
    USART_InitTypeDef USART_InitStructure;
    USART_InitTypeDef USART2_InitStructure;

    /*TODO: USART1 configuration*/
    USART_InitStructure.USART_BaudRate = 9600;
    USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;

    USART2_InitStructure.USART_BaudRate = 9600;
    USART2_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
    USART2_InitStructure.USART_Parity = USART_Parity_No;
    USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART2_InitStructure.USART_StopBits = USART_StopBits_1;

    USART_Init(USART1, &USART_InitStructure);
    USART_Init(USART2, &USART2_InitStructure);

    /*TODO: USART1 cmd ENABLE*/
    USART_Cmd(USART1, ENABLE);
    USART_Cmd(USART2, ENABLE);

    /*TODO: USART1 IT Config*/
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
}

void NVIC_Configure() {
    /*TODO: NVIC_configuration */

```



```

NVIC_InitTypeDef NVIC_USART_InitTD;
NVIC_InitTypeDef NVIC_USART2_InitTD;

NVIC_USART_InitTD.NVIC_IRQChannel = USART1_IRQn;
NVIC_USART2_InitTD.NVIC_IRQChannel = USART2_IRQn;

NVIC_USART_InitTD.NVIC_IRQChannelCmd = ENABLE;
NVIC_USART2_InitTD.NVIC_IRQChannelCmd = ENABLE;

NVIC_USART_InitTD.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_USART2_InitTD.NVIC_IRQChannelPreemptionPriority = 0;

NVIC_USART_InitTD.NVIC_IRQChannelSubPriority = 0;
NVIC_USART2_InitTD.NVIC_IRQChannelSubPriority = 0;

NVIC_Init(&NVIC_USART_InitTD);
NVIC_Init(&NVIC_USART2_InitTD);
}

/*TODO: IRQHandler */

void USART1_IRQHandler(void) {
    char c;

    if(USART_GetITStatus(USART1, USART_IT_RXNE ) != RESET) {
        c = USART_ReceiveData(USART1);
        USART_SendData(USART1, c);
        USART_SendData(USART2, c);
        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
    }
}

void USART2_IRQHandler(void) {
    char c;

    if(USART_GetITStatus(USART2, USART_IT_RXNE ) != RESET) {
        c = USART_ReceiveData(USART2);
        USART_SendData(USART1, c);
        USART_ClearITPendingBit(USART2,USART_IT_RXNE);
    }
}

int main()
{
    int i=0;

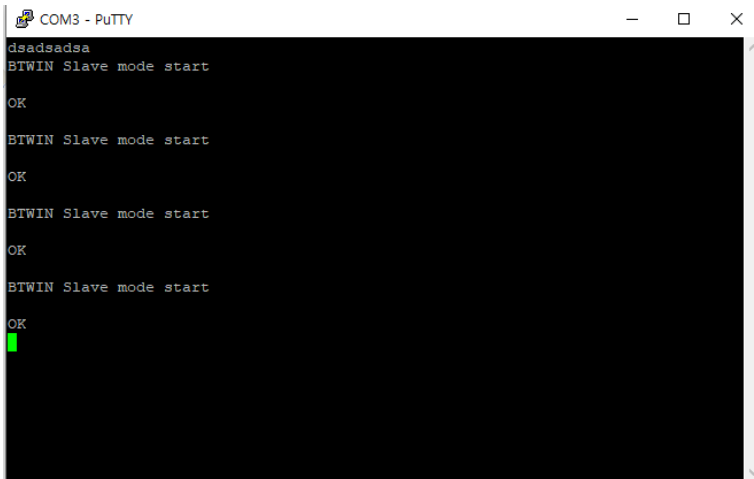
    SystemInit();
    RCC_Configure();
    GPIO_Configure();
    USART_Configure();
    NVIC_Configure();

    while(1) {
    }
}

```

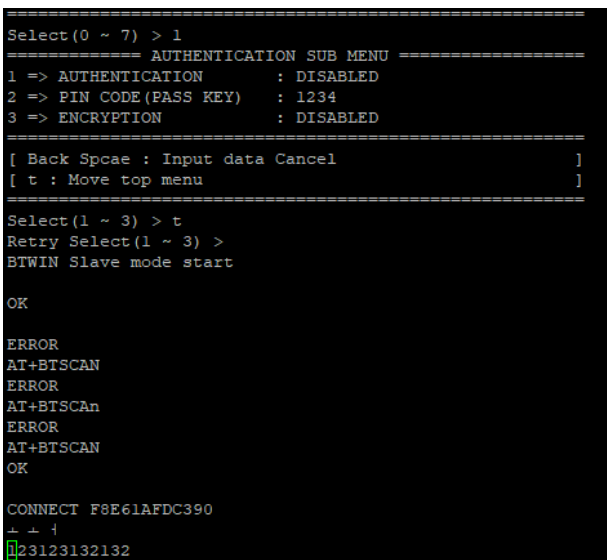
5.2 결과 확인 및 사진

해당 소스 코드대로 동작하는 프로그램은 다음과 같은 화면으로 확인되었다.



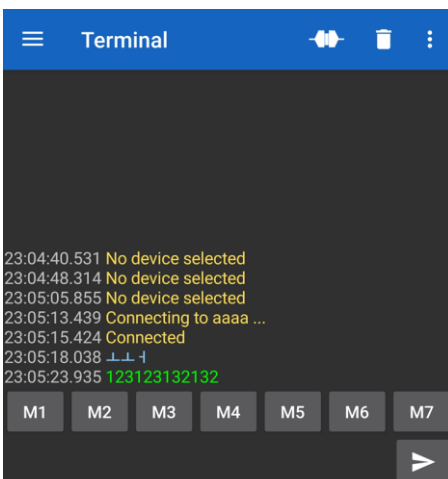
```
COM3 - PUTTY
dsadsadsa
BTWIN Slave mode start
OK
BTWIN Slave mode start
OK
BTWIN Slave mode start
OK
BTWIN Slave mode start
OK
BTWIN Slave mode start
OK
█
```

블루투스가 연결된 상태이다. 이때 스마트 폰이 Master 이고, 블루투스 모듈이 Slave 상태임을 볼 수 있다.



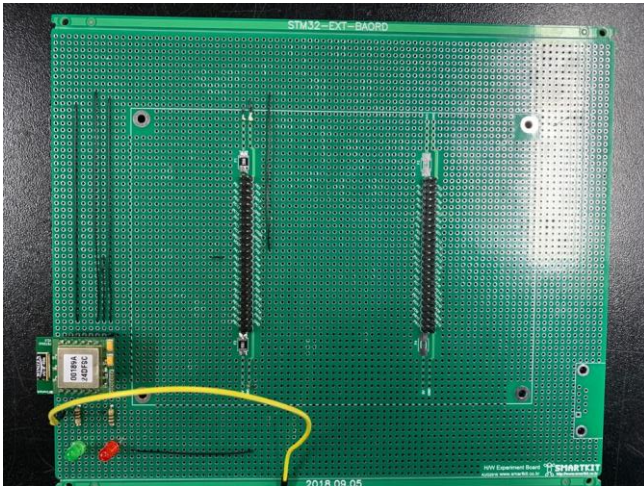
```
=====
Select(0 ~ 7) > 1
===== AUTHENTICATION SUB MENU =====
1 => AUTHENTICATION      : DISABLED
2 => PIN CODE(PASS KEY)  : 1234
3 => ENCRYPTION          : DISABLED
=====
[ Back Spcae : Input data Cancel          ]
[ t : Move top menu                        ]
=====
Select(1 ~ 3) > t
Retry Select(1 ~ 3) >
BTWIN Slave mode start
OK
ERROR
AT+BTSCAN
ERROR
AT+BTSCAN
ERROR
AT+BTSCAN
OK
CONNECT F8E61AFDC390
+ + +
123123132132
```

Putty 통신화면에서 확인할 수 있는 출력화면이다.



```
Terminal
23:04:40.531 No device selected
23:04:48.314 No device selected
23:05:05.855 No device selected
23:05:13.439 Connecting to aaaa ...
23:05:15.424 Connected
23:05:18.038 + + +
23:05:23.935 123123132132
M1 M2 M3 M4 M5 M6 M7
➤
```

스마트폰 블루투스 통신화면에서 확인할 수 있는 출력 화면이다.



블루투스 모듈 연결을 위한 납땜이 완료된 기판이다.



납땜을 완성하고 보드를 끼워 놓은 상태이다. LED 에 불이 들어오는 것을 확인할 수 있다.

06. 결론

이번 실험은 6 주차와 매우 유사한 실험이었다. 그래서 이해나 코딩부분은 어렵지 않았다. 문제는 납땜이었는데 생각보다 오래 걸렸고 어려웠다. 혼자 하니 고정도 잘 안되고 납땜도 잘 안되어서 시간이 많이 지체되었다. 후반에는 모든 조원이 함께 납땜을 하게 되어 훈훈한 분위기 속에 실험을 마쳤다.