임베디드 시스템 설계 및 실험 9주차 실험 결과 보고서

004분반 9조

조장 – 김주송 조원 – 강선호, 박형주, 손봉국

1. 실험 목적

- Bluetooth 동작 및 기판 납땜

2. 실험 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 자료참고하여 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

3. 실험 내용

이번 프로젝트에서는 USART1, USART2를 사용하여 컴퓨터 및 스마트폰과 통신을 수행하는 것이다. 기본적으로 USART1, USART2 및 USART의 TX/RX 포트에 대하여 설정을 해야한다. 이를 위해 RCC_Configure 함수를 다음과 같이 작성하였다.

```
16  void RCC_Configure(void)
17  {
18      // TODO: Enable the APB2 peripheral clock using the function 'RCC_APB2PeriphClockCmd'
19
20      /* USART1, USART2 TX/RX port clock enable */
21      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
22
23      /* USART1, USART2 clock enable */
24      RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
25       RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);
26
27      /* Alternate Function IO clock enable */
28       RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
29  }
```

그림 1. RCC_Configure 함수 소스코드

그리고 USART1, USART2 통신을 위한 GPIO 핀을 설정해주어야 하는데 USART1의 경우 TX는 Port A의 9번을 사용하며 RX는 Port A의 10번을 사용한다. USART2의 경우 TX는 PortA의 2번을 사용하며 RX는 Port A의 3번을 사용한다.

Table 54. USART1 remapping

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

그림 2. USART1 mapping 주소 (Reference Manual p.181)

Table 53. USART2 remapping

Alternate functions	USART2_REMAP = 0	USART2_REMAP = 1(1)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

1. Remap available only for 100-pin and 144-pin packages.

그림 3. USART2 mapping 주소 (Reference Manual p.180)

TX는 데이터를 전송할 때 사용하고 RX는 데이터를 수신할 때 사용하기 때문에 TX는 Output 모드로, RX는 Input 모드로 설정하였다. 해당 내용을 GPIO_Configure 함수에 작성하였다.

```
31  void GPIO_Configure(void)
32  {
33      GPIO_InitTypeDef GPIO_InitStructure;
34
35        // TODO: Initialize the GPIO pins using the structure 'GPIO_InitTypeDef' and the function 'GPIO_Init'
36
37      /* USART1 pin setting */
38      //TX
39      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
40      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
41      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
42      GPIO_Init(GPIOA, &GPIO_InitStructure);
43
44     //RX
45      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
46      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
47      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
48      GPIO_Init(GPIOA, &GPIO_InitStructure);
49
50      /* USART2 pin setting */
51      //TX
52      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
53      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
54      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
55      GPIO_Init(GPIOA, &GPIO_InitStructure);
56
57     //RX
58      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
59      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
60      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
61      GPIO_Init(GPIOA, &GPIO_InitStructure);
62
63  }
```

그림 4. GPIO_Configure 함수 소스코드

USART1 과 USART2 를 사용하기 전 통신 관련 내용을 설정해주어야 하기 때문에 Baud Rate 를 9600 으로 설정하고 USART Mode 는 TX 와 RX 를 모두 사용 가능하게 해주었다. Word Length 는 8 비트(1 바이트)로 설정하였으며 Parity Bit 는 설정하지 않았다. 마지막으로 Stop Bit 는 1 비트로 설정하고 Hardware Flow Control 은 None 으로 설정하였다. 해당 설정은 UART1 과 UART2 이 동일하므로 동일한 내용을 각각 USART1_Init, USART2_Init 함수로 작성하였다.

```
65  void USART1_Init(void)
66  {
67      USART_InitTypeDef USART1_InitStructure;
68
69     // Enable the USART1 peripheral
70     USART_Cmd(USART1, ENABLE);
71
72     // TODO: Initialize the USART using the structure 'USART_InitTypeDef' and the function 'USART_Init'
73     USART1_InitStructure.USART_BaudRate = 9600;
74      USART1_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx ;
75      USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
76      USART1_InitStructure.USART_Parity = USART_Parity_No;
77      USART1_InitStructure.USART_StopBits = USART_StopBits_1;
78      USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
79      USART_Init(USART1,&USART1_InitStructure);
80
81
82     // TODO: Enable the USART1 RX interrupts using the function 'USART_ITConfig' and the argument value 'Receive Data register not empty interrupt'
83     USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
84
85  }
```

그림 5. USART1_Init 함수 소스코드

```
87  void USART2_Init(void)
88  {
89      USART_InitTypeDef USART2_InitStructure;
90
91     // Enable the USART2 peripheral
92     USART_Cmd(USART2, ENABLE);
93
94     // TODO: Initialize the USART using the structure 'USART_InitTypeDef' and the function 'USART_Init'
95     USART2_InitStructure.USART_BaudRate = 9600;
96      USART2_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx ;
97      USART2_InitStructure.USART_WordLength = USART_WordLength_8b;
98      USART2_InitStructure.USART_Parity = USART_Parity_No;
99      USART2_InitStructure.USART_StopBits = USART_StopBits_1;
100     USART2_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
101     USART_Init(USART2, &USART2_InitStructure);
102
103
104    // TODO: Enable the USART2 RX interrupts using the function 'USART_ITConfig' and the argument value 'Receive Data register not empty interrupt'
105    USART_ITConfig(USART2, USART_IT_RXNE,ENABLE);
106
107 }
```

그림 6. USART2_Init 함수 소스코드

그리고 각 USART 통신에 대해 우선순위를 정해주어야 하는데 이번 프로젝트에서는 PC와 스마트폰이 동시에 보드로 데이터를 전송하는 상황은 가정하지 않았기 때문에 우선순위는 모두 같은 값으로 설정하였다. 그리고 해당 내용을 NVIC_Configure 함수로 작성하였다.

```
109 void NVIC_Configure(void) {
110
111     NVIC_InitTypeDef NVIC_InitStructure;
112
113     // TODO: fill the arg you want
114     NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
115
116     // USART1
117     // 'NVIC_EnableIRQ' is only required for USART setting
118     NVIC_EnableIRQ(USART1_IRQn);
119     NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
120     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
121     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
122     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
123     NVIC_Init(&NVIC_InitStructure);
124
125     // USART2
126     // 'NVIC_EnableIRQ' is only required for USART setting
127     NVIC_EnableIRQ(USART2_IRQn);
128     NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
129     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
130     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
131     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
132     NVIC_Init(&NVIC_InitStructure);
133 }
```

그림 7. NVIC_Configure 함수 소스코드

통신 과정은 스마트폰의 앱을 통해 블루투스 모듈로 데이터를 전송할 때에는 USART2를 사용하며, 이후 데이터를 USART1을 통해 PC로 전송한다. 그리고 PC의 터미널을 이용해 데이터를 블루투스 모듈로 전송할 때에는 USART1을 사용하며, 이후 데이터를 USART2를 통해 스마트폰으로 전송한다. 해당 내용을 각각 USART1_IRQHandler, USART2_IRQHandler 함수로 작성하였다

.

```
135 void USART1_IRQHandler() {
136     uint16_t word;
137     if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET){
138         // the most recent received data by the USART1 peripheral
139         word = USART_ReceiveData(USART1);
140
141         // TODO implement
142         USART_SendData(USART2, word);
143
144         // clear 'Read data register not empty' flag
145         USART_ClearITPendingBit(USART1,USART_IT_RXNE);
146     }
147 }
```

그림 8. USART1_IRQHandler 함수 소스코드

```
149  void USART2_IRQHandler() {
150      uint16_t word;
151      if(USART_GetITStatus(USART2,USART_IT_RXNE)!=RESET){
152          // the most recent received data by the USART2 peripheral
153          word = USART_ReceiveData(USART2);
154
155          // TODO implement
156          USART_SendData(USART1, word);
157
158          // clear 'Read data register not empty' flag
159         USART_ClearITPendingBit(USART2,USART_IT_RXNE);
160      }
161  }
```

그림 9. USART2_IRQHandler 함수 소스코드

마지막으로 Main 함수의 경우 시스템을 설정하는 함수들을 실행한 후 아무 내용이 없는 무한반복문을 실행하였다.

```
163  int main(void)
164  {
165    char msg[] = "abcde\r\n";
166    unsigned int i;
167
168      SystemInit();
169
170      RCC_Configure();
171
172      GPIO_Configure();
173
174      USART1_Init();      // pc
175
176      USART2_Init();      // bluetooth
177
178      NVIC_Configure();
179
180      while (1) {
181      }
182      return 0;
183  }
```

그림 10. Main 함수 소스코드

블루투스 모듈 사용을 위해 기판 위에 핀헤더, 저항, LED 등을 납땜하여 점프선으로 보드와 모듈을 연결하여 사용가능한 형태로 제작하였다. 한 번 납땜하면 복구하기 번거롭기 때문에 미

리 전선이 교차하지 않도록 회로도를 그려보고 해당 회로도를 보며 납땜을 진행하였다. 아래는 수업 자료에 있는 회로도를 보고 실제 기판을 찍어 모의로 그려본 회로도 이다.

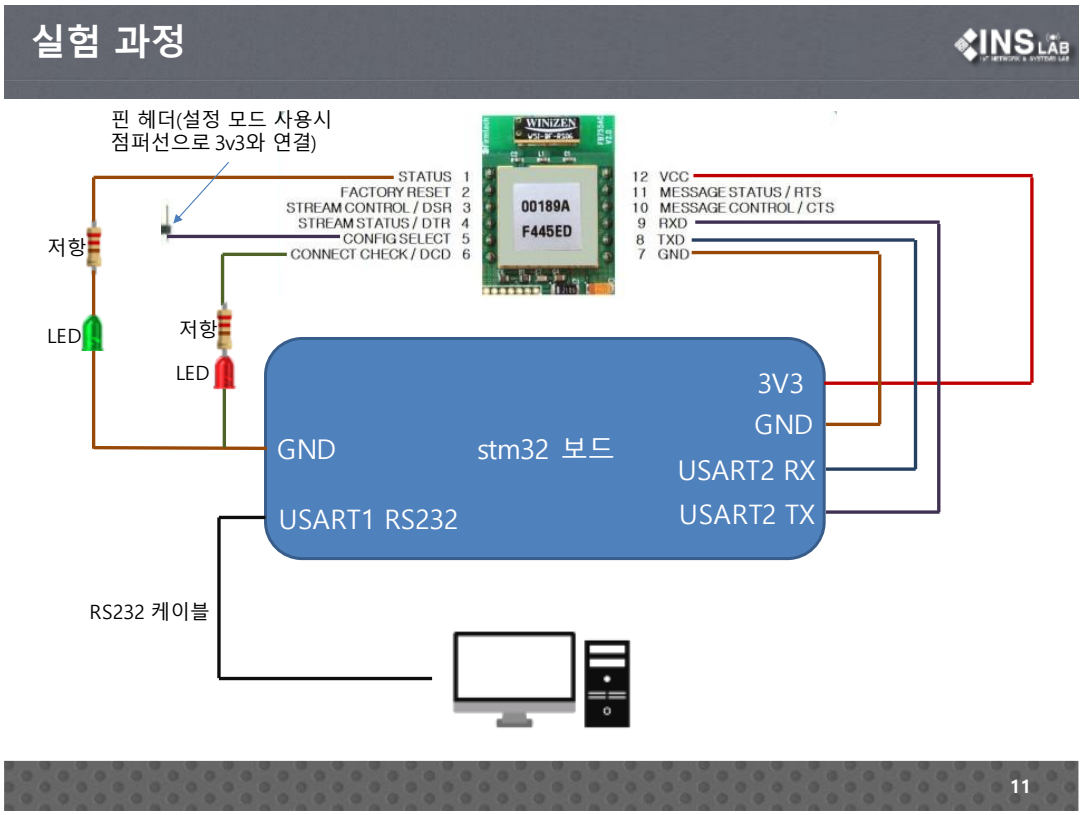

그림 11. 9주차 강의 자료 p.11

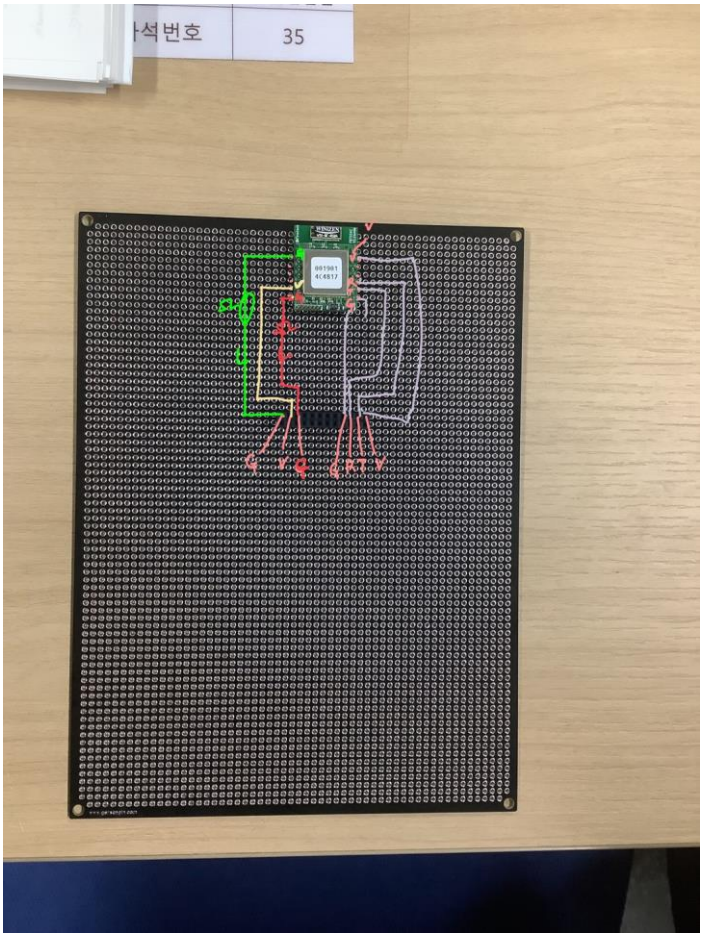

그림 12. 납땜 전 기판에 회로도를 그린 그림

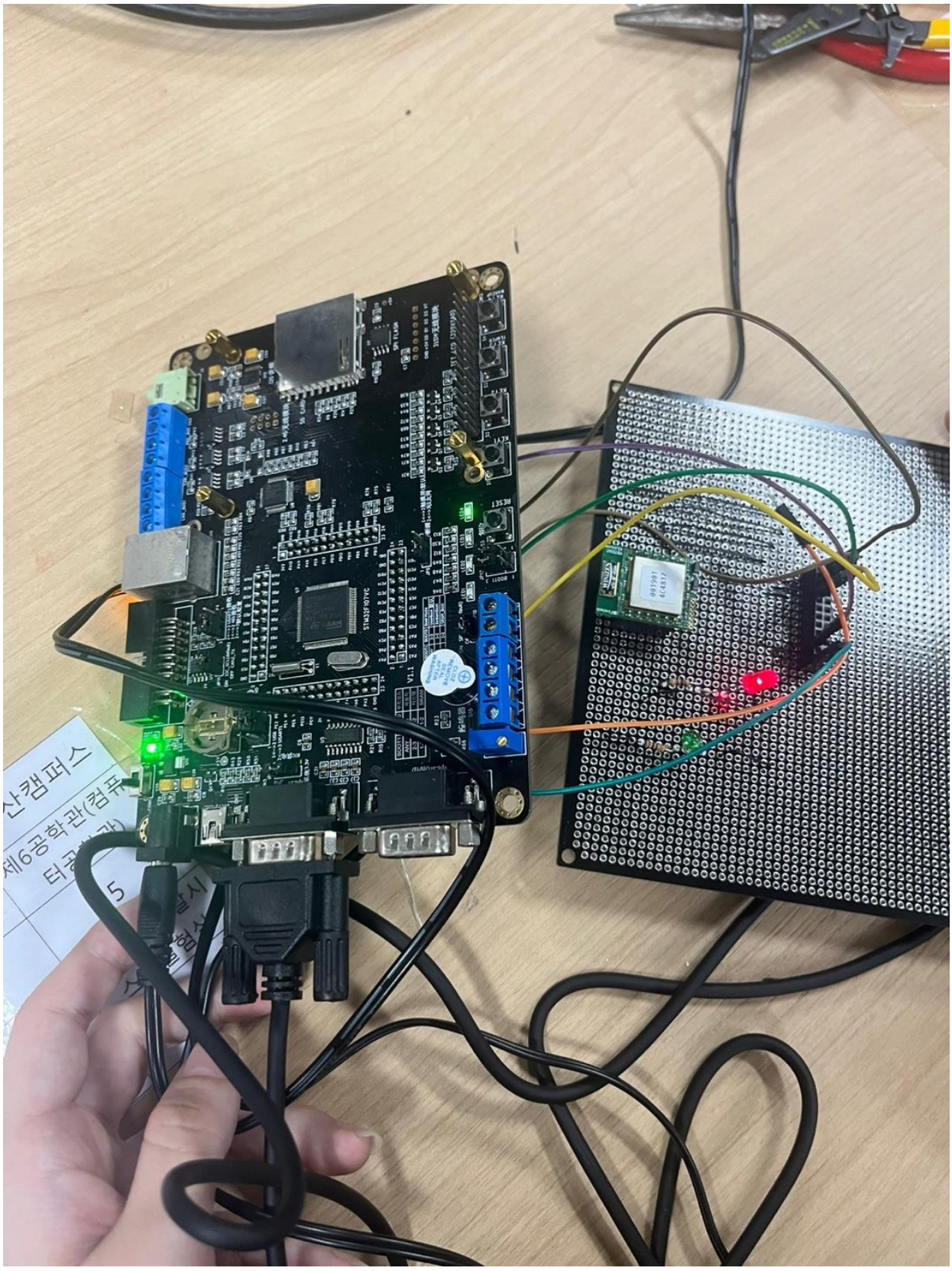

그림 13. 납땜 된 기판에 보드를 연결한 모습

4. 실험 결과

먼저 3.3V 전원을 블루투스 모듈의 CONFIG SELECT 포트에 인가하여 SLAVE 모드를 활성화 시켰다. 이후 3.3V 전원을 해제한 후 보드를 재시작한 다음, AT+BTSCAN 명령어를 입력하여 스마트폰과 보드를 연결하였다.

```
|==================================|
|        Model name : FB755        |
|        Version    : 1.2.6        |
|==================================|

==================== TOP MENU ============================
0 => DEVICE NAME           : THU_09
1 => AUTHENTICATION        : DISABLE PINCODE[BTWIN]
2 => REMOTE BD ADDRESS     : 5CCB99ED9A56
     LOCAL BD ADDRESS      : 0019014C4817
3 => CONNECTION MODE       : CNT_MODE4
4 => OTHER PARAMETER       : E,D,5,2B,2,D
5 => UART CONFIG           : 9600,8,n,1
6 => ROLE                  : SLAVE
7 => OPERATION MODE        : OP_MODE0
==========================================================
[ Back Spcae : Input data Cancel                         ]
[ t : Move top menu                                      ]
==========================================================
Select(0 ~ 7) >
BTWIN Slave mode start

OK

OK

CONNECT 5CCB992052CB
gufhthy
gufhthy
gufhthy55

DISCONNECT
```

그림 14. SLAVE 모드 활성화가 시작된 터미널 화면.

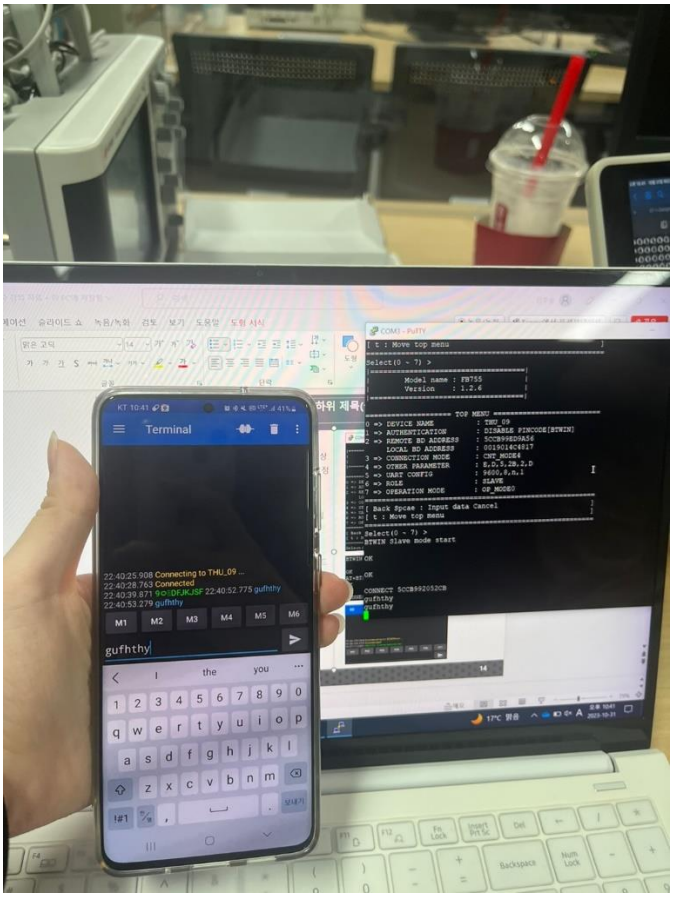

그림 15. 스마트폰과 PC가 연결된 모습

이후 스마트폰의 Serial Bluetooth Terminal 애플리케이션을 통해 PC와 통신이 정상적으로 작동하는 것을 확인할 수 있다.