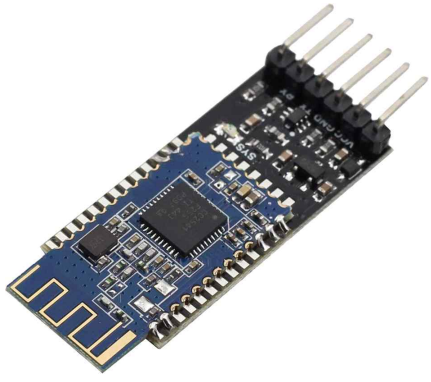
APPENDIX 16

블루투스 연결

필요한 부품

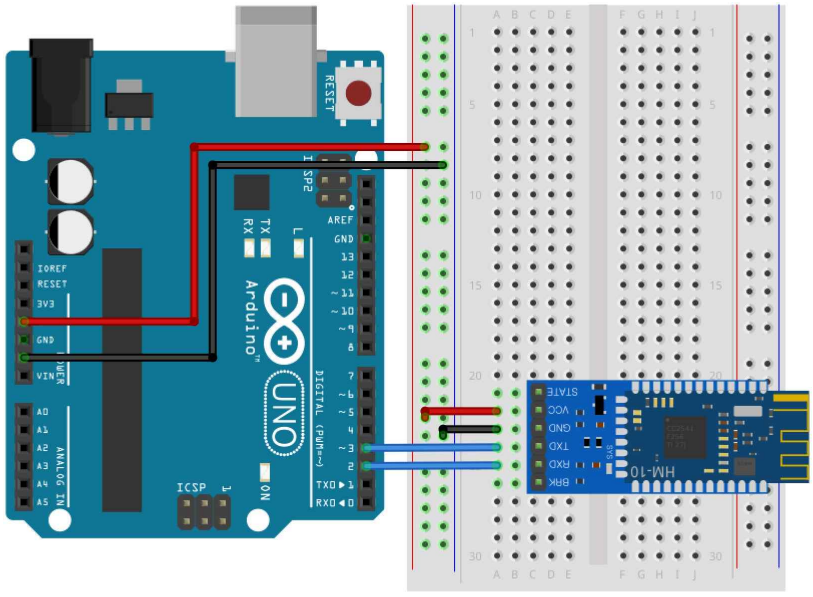
- 아두이노 우노
- 블루투스 4.0 시리얼 모듈(HM-10)
- 10kΩ 가변저항
- 220Ω 저항
- LED

이 장에서는 블루투스 4.0 BLE를 지원하는 시리얼 모듈로 HM-10 기반 모듈을 사용한다. [그림 1]은 HM-10 기반의 블루투스 4.0 BLE 시리얼 모듈로 이후 'HM-10 모듈'이라고 지칭한다.



[그림 1] HM-10 기반 블루투스 4.0 시리얼 모듈

HM-10 모듈은 UART 시리얼 통신을 통해 AT 명령을 전송하여 설정하고 사용할 수 있다. 먼저 HM-10 모듈을 아두이노 우노에 연결하자.



[그림 2] HM-10 모듈 연결

아두이노 우노에는 [코드 1]을 업로드한다. [코드 1]은 시리얼 모니터에 입력된 내용을 HM-10 모듈로 전달하고, HM-10 모듈에서 출력된 내용을 시리얼 모니터로 출력하는 스케치다. 시리얼 모니터는 9,600 보율을 선택하고 추가 문자는 'Both NL & CR'을 선택한다.

HM-10 모듈에 설치된 펌웨어 버전에 따라 HM-10 모듈과의 통신 속도는 9,600 또는 115,200 보율 중 하나로 선택되어 있으므로 설정 과정에서 원하는 결과를 얻지 못하였다면 보율을 변경하면서 테스트해 보아야 한다. 이 장에서는 9,600 보율을 사용하는 것으로 가정하지만 AT 명령으로 속도를 변경할 수 있다. 펌웨어 버전에 따라 AT 명령어 끝에 'CR+LF' 개행문자가 필요하지 않을 수도 있지만, 추가된 개행문자를 자동으로 처리하는 기능이 있으므로 시리얼 모니터에서 'Both NL & CR' 옵션을 선택하고 사용하면 된다.

코드 1 **HM-10 모듈 설정** APPENDIX_16-1_bluetooth_setting.ino

```
#include <SoftwareSerial.h>

SoftwareSerial HM10(3, 2);              // (RX, TX) -> HM-10의 (TX, RX)

void setup() {
  Serial.begin(9600);                   // 컴퓨터와의 시리얼 통신 초기화
  HM10.begin(9600);                     // 블루투스 모듈과의 시리얼 통신 초기화
```

```
}

void loop() {
  if (Serial.available()) {              // 시리얼 모니터 → 아두이노 → HM-10 모듈
    char ch = Serial.read();
    Serial.write(ch);
    HM10.write(ch);
  }

  if (HM10.available()) {                // HM-10 모듈 → 아두이노 → 시리얼 모니터
    char ch = HM10.read();
    Serial.write(ch);
  }
}
```

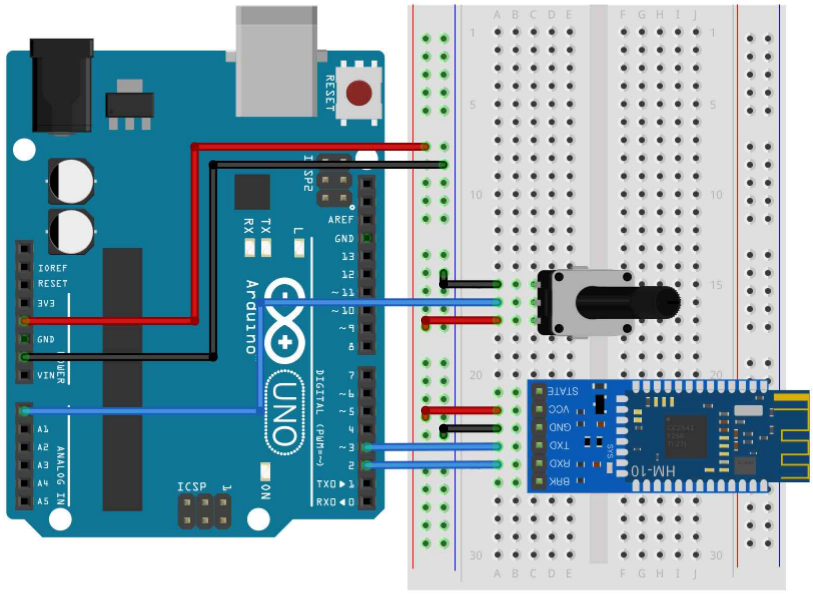
시리얼 모니터에 'AT' 명령을 입력하여 'OK'가 수신되면 HM-10 모듈이 정상적으로 동작하는 것이다. **AT+NAME**은 모듈의 이름을 설정하는 명령이다. 모듈 이름은 스마트폰과 같은 블루투스 기기에서 검색되는 이름이다. **AT+NAMEcapBLE**로 모듈의 이름을 **capBLE**로 변경하자. **AT+BAUD?** 명령은 통신 속도를 검사하는 명령으로 반환되는 상수에 따른 통신 속도는 [표 1]과 같다. [그림 3]은 통신 속도가 9,600으로 설정된 경우로 통신 속도를 115,200 보율로 변경하려면 **AT+BAUD4** 명령을 실행하면 된다. 보율을 변경한 후에는 **AT+RESET** 명령으로 모듈을 리셋하여 변경된 설정을 변경해야 하며, **AT+VERS?**는 모듈에 설치된 펌웨어의 버전을 확인하는 명령이다.

[그림 3] AT 명령 실행 결과

[표 1] 통신 속도에 따른 상수

상수	속도(baud)	상수	속도(baud)
0	9,600	5	4,800
1	19,200	6	2,400
2	38,400	7	1,200
3	57,600	8	230,400
4	115,200		

가변저항의 값을 스마트폰으로 전송해 보자. 아두이노 우노에 [그림 4]와 같이 HM-10 모듈과 가변저항을 연결한다.



[그림 4] HM-10 모듈과 가변저항 연결

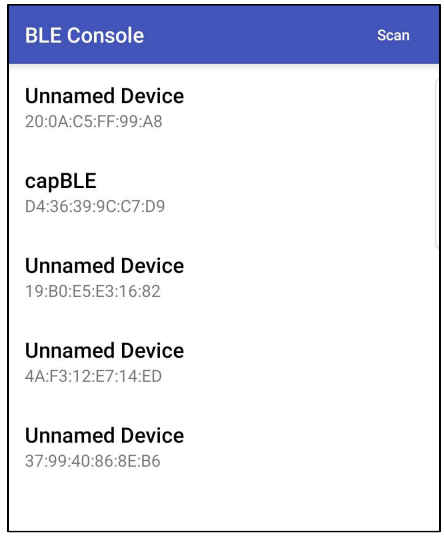
아두이노에는 가변저항의 값을 읽어 1초 간격으로 스마트폰으로 전송하는 [코드 2]를 업로드한다.

코드 2 **HM-10을 통한 센서 데이터 전송** APPENDIX_16-2_BLE_sensor.ino

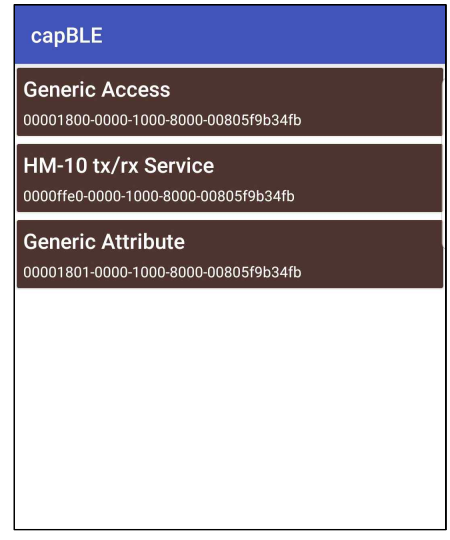
```
#include <SoftwareSerial.h>

SoftwareSerial HM10(3, 2);              // (RX, TX) -> HM-10의 (TX, RX)

const int POT = A0;                     // 가변저항을 A0 핀에 연결

void setup() {
  Serial.begin(9600);                   // 컴퓨터와의 시리얼 통신 초기화
  HM10.begin(9600);                     // 블루투스 모듈과의 시리얼 통신 초기화
}

void loop() {
  int val = analogRead(POT);

  // 시리얼 모니터로 가변저항값 출력
  Serial.print(F("Analog Value: "));
  Serial.println(val);
```

```
  HM10.println(val);                    // 스마트폰으로 가변저항값 전송

  delay(1000);
}
```

스마트폰에서 HM-10 모듈과의 통신을 위해서는 'BLE Serial Console'[1]을 사용한다. 애플리케이션을 실행하면 주변의 BLE 장치를 검색하여 보여준다. 검색된 장치 중 'capBLE'[2]를 찾아 누르면 HM-10 모듈을 통해 사용할 수 있는 서비스 목록을 확인할 수 있다.



[그림 5] 검색된 BLE 장치 목록

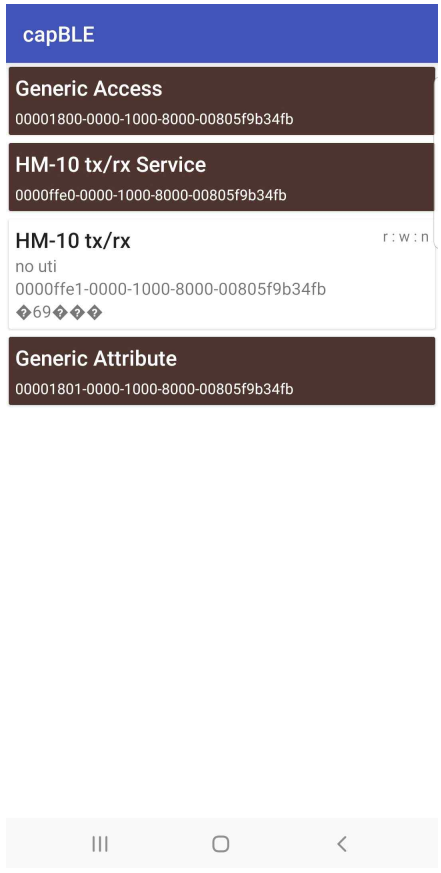


[그림 6] capBLE 선택

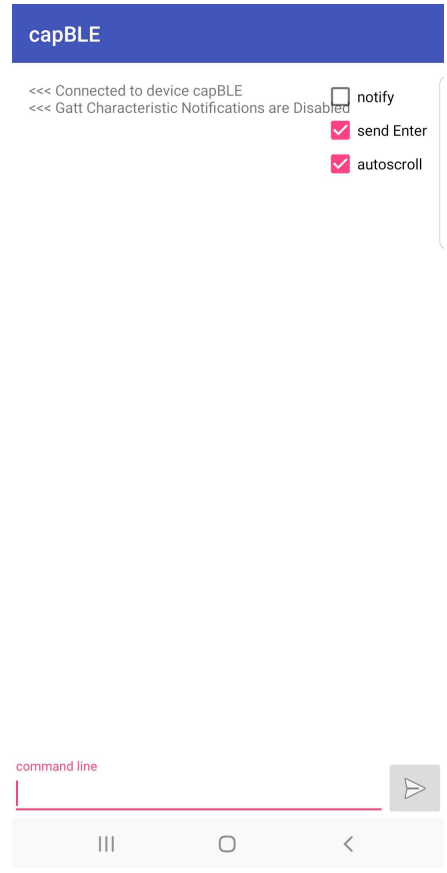
이 장에서는 HM-10 모듈의 'HM-10 tx/rx Service'를 사용한다. tx/rx Service는 블루투스 클래식과 비슷한 방법으로 데이터를 주고받을 수 있도록 해주는 서비스다. 'HM-10 tx/rx Service'를 누르면 'HM-10 tx/rx' 속성이 나타나고 'HM-10 tx/rx' 누르면 데이터를 주고받을 수 있는 터미널이 나타난다.

1 https://play.google.com/store/apps/details?id=org.pampanet.mobile.serialconsole&hl=ko&gl=US

2 'AT+NAME' 명령으로 설정한 이름이 장치 목록에 나타날 것이며 여기서는 'capBLE'로 설정한 것으로 가정한다.

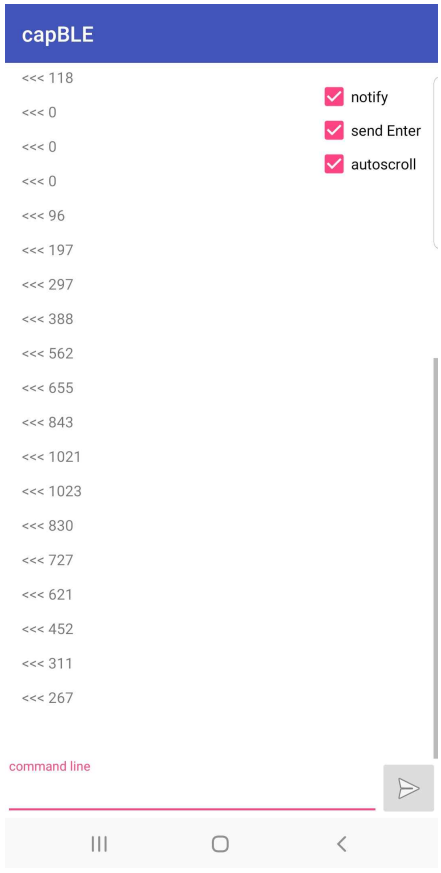


[그림 7] HM-10 tx/rx

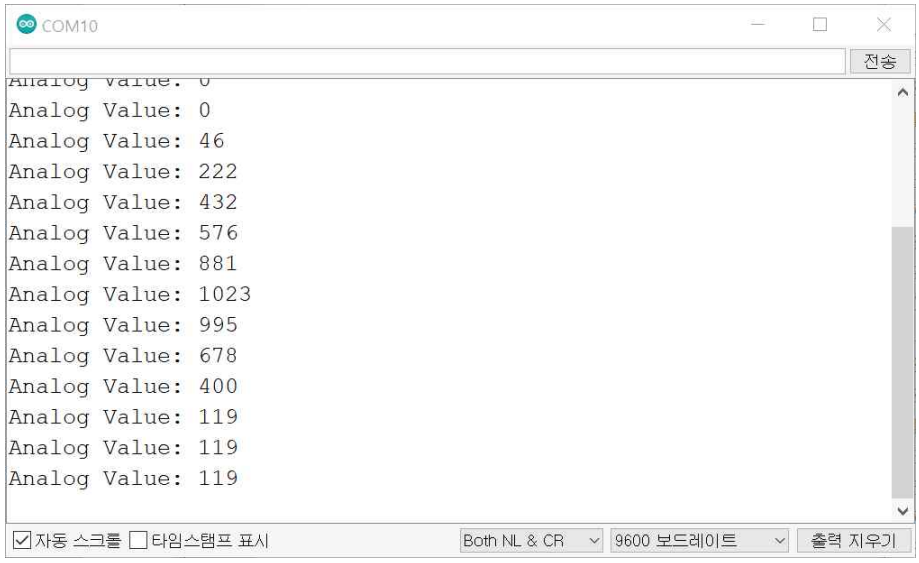


[그림 8] 시리얼 콘솔

시리얼 콘솔에서 'notify'를 선택하면 아두이노에서 전송하는 가변저항의 값을 확인할 수 있다. notify는 데이터를 수신했을 때 이를 자동으로 알려주는 기능이다. BLE Serial Console에는 수신한 데이터를 그래프로 보여주는 기능은 없다.

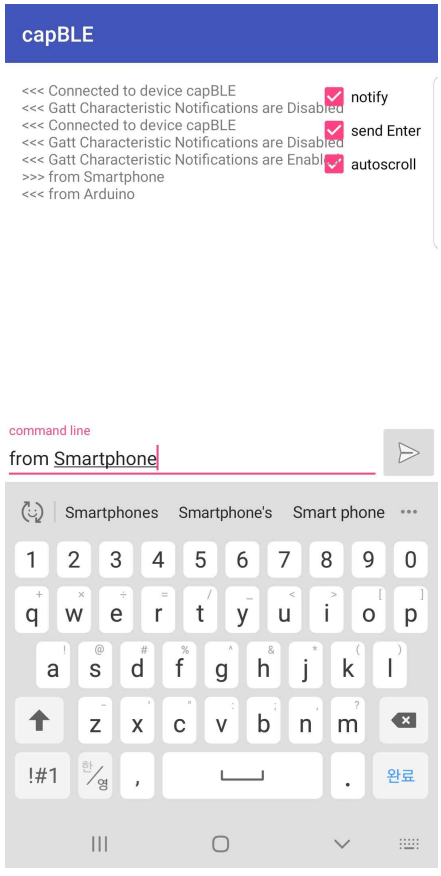


[그림 9] 가변저항 값 수신



[그림 10] 가변저항 값 송신

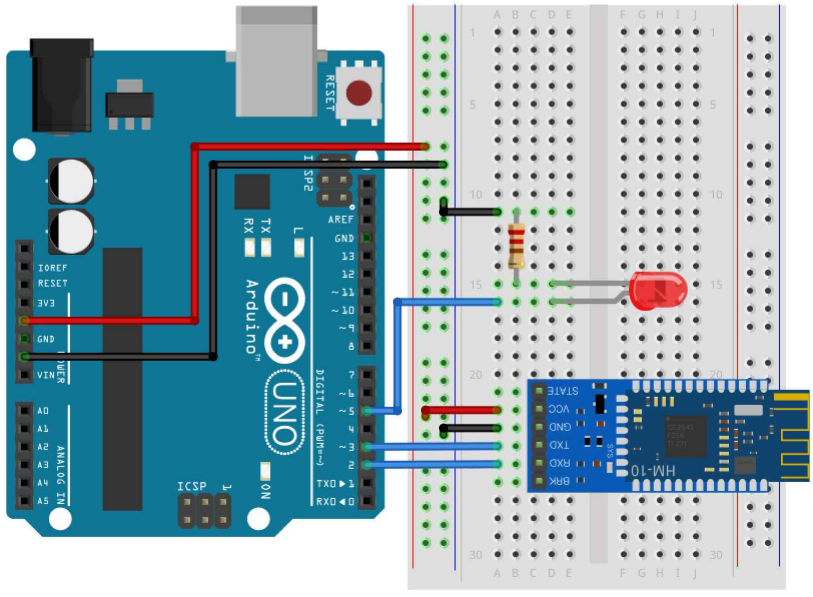
[코드 1]을 아두이노에 업로드하고 스마트폰에 BLE Serial Console을 실행하여 연결하였다면 데이터를 주고받는 것이 가능하다. [그림 9]의 'command line'에 전송할 데이터를 입력하고 전송 버튼을 누르면 시리얼 모니터에 표시되고, 반대로 시리얼 모니터의 입력 창에 입력한 데이터는 BLE Serial Console에 표시된다. [그림 12]에서 'OK+CONN'은 스마트폰과 연결되었다는 것을, 'OK+LOST'는 스마트폰과 연결이 끊어진 것을 나타낸다.



[그림 11] 아두이노와 스마트폰의 데이터 교환 - 스마트폰

[그림 12] 아두이노와 스마트폰의 데이터 교환 - 아두이노

BLE Serial Console을 사용하여 LED를 제어해 보자. 아두이노의 5번 핀에 [그림 13]과 같이 LED를 연결한다.



[그림 13] HM-10 모듈과 LED 연결

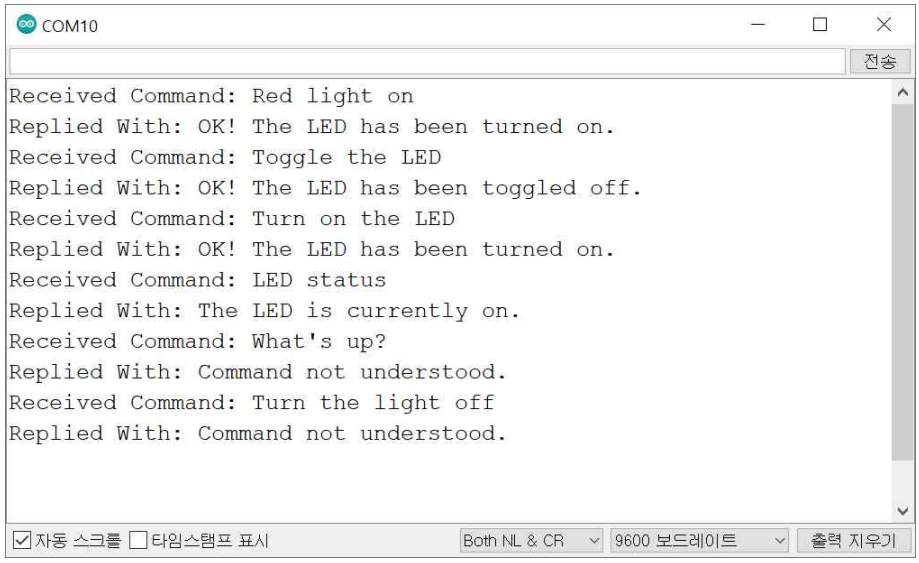
[코드 3]은 스마트폰의 BLE Serial Console에 영어 문장을 입력하고 이를 해석하여 LED를 제어하는 스케치의 예다.

코드 3 **HM-10을 통한 LED 제어** APPENDIX_16-3_BLE_led.ino

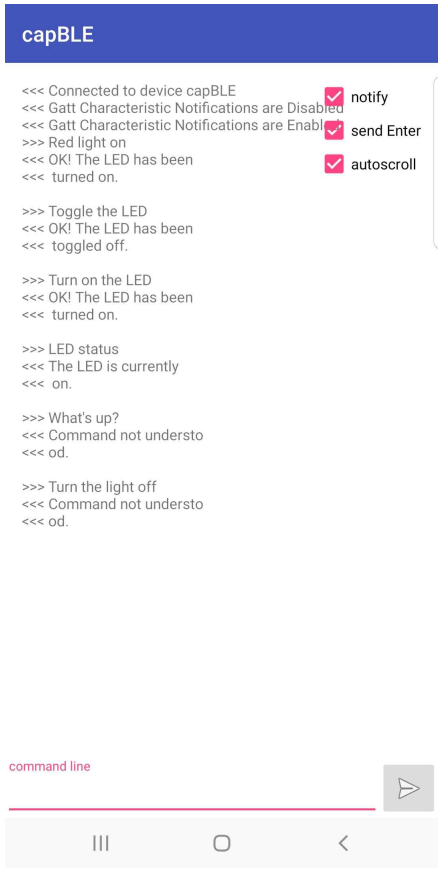
```
#include <SoftwareSerial.h>

SoftwareSerial HM10(3, 2);              // (RX, TX) -> HM-10의 (TX, RX)

// 제어할 LED는 5번 핀에 연결되어 있음
const int CTRL_LED = 5;

// LED 상태와 제어 및 반환 문자열을 위한 변수
bool led_state = LOW;
String cmd = "";
String reply = "";

void setup() {
  Serial.begin(9600);                   // 컴퓨터와의 시리얼 통신 초기화
  HM10.begin(9600);                     // 블루투스 모듈과의 시리얼 통신 초기화

  pinMode(CTRL_LED, OUTPUT);
}

void loop() {
  // 수신 데이터가 있으면 읽어 LED 제어 명령을 찾아낸다.
  while (HM10.available() > 0) {
    // 새 줄 문자를 만날 때까지 수신 데이터를 읽어 버퍼에 저장한다.
    cmd = HM10.readStringUntil('\n');
    Serial.print(F("Received Command: "));
    Serial.println(cmd);

    // 소문자로 변환하여 대소문자 관계없이 명령을 인식할 수 있도록 한다.
    cmd.toLowerCase();

    // "red" 또는 "led" 문자열이 발견되면 해석을 시작한다.
    if (cmd.indexOf(F("red")) != -1 || cmd.indexOf(F("led")) != -1) {
```

```
    // "on" 포함 여부 검사
    if (cmd.indexOf(F("on")) != -1) {         // "on"이 포함된 경우
      led_state = HIGH;
      reply = F("OK! The LED has been turned on.");
    }

    // "off" 포함 여부 검사
    else if (cmd.indexOf(F("off")) != -1) {
      led_state = LOW;
      reply = F("OK! The LED has been turned off.");
    }

    // "toggle" 포함 여부 검사
    else if (cmd.indexOf(F("toggle")) != -1) {
      led_state = !led_state;
      if (led_state) reply = F("OK! The LED has been toggled on.");
      else reply = F("OK! The LED has been toggled off.");
    }

    // "red" 또는 "led"가 포함되어 있지만 LED 제어 키워드가 포함되어 있지 않음
    else {
      if (led_state) reply = F("The LED is currently on.");
      else reply = F("The LED is currently off.");
    }

    // LED 상태 설정
    digitalWrite(CTRL_LED, led_state);
  }
  else {
    reply = F("Command not understood.");
  }

  // 수신한 명령 문자열에 대한 응답 전송
  HM10.println(reply);
  Serial.print(F("Replied With: "));
  Serial.println(reply);
```

```
    }
  }
```



[그림 14] [코드 3] 실행 결과 - 시리얼 모니터



[그림 15] [코드 3] 실행 결과 – 스마트폰

UART 전송 서비스는 20바이트 크기의 버퍼를 사용하므로 스마트폰에서 수신된 데이터가 20바이트 크기로 나뉘어 표시되고 있다. 또한 BLE Serial Console 애플리케이션의 버그로 20바이트보다 긴 문자열을 전송하는 경우 오류가 발생할 수 있으므로 'What is the LED status ?'와 같이 20바이트보다 긴 문자열을 전송하지 않는 것이 좋다.