

Chapter 127. 블루투스

블루투스(Bluetooth)는 1994년 에릭슨(Ericsson)이 개발한 개인 근거리 무선 통신 (Personal Area Network, PAN)을 위한 표준으로 RS232를 대체하는 저가격, 저전력의 무선 기술로 개발되었다. 블루투스는 기본적으로 10m 이내에서의 통신을 목표로 하며 이 범위는 100m까지 확장 가능하다. 블루투스는 2.4GHz 대역인 ISM (Industrial, Scientific, Medical) 대역을 사용하며 상위 및 하위 영역은 다른 시스템과의 간섭을 막기 위해 제거하고 2402MHz에서 2480MHz 까지 총 79개의 채널을 사용한다. ISM 대역은 산업, 과학 및 의료 목적으로 할당된 대역으로 전파 사용에 대한 허가를 받을 필요가 없어 저전력의 전파를 발산하는 개인 무선기에 많이 사용되고 있으며 무선랜 역시 ISM 대역을 사용한다.

마이크로컨트롤러와 함께 사용되고 있는 블루투스는 2.0이 대부분이며 일부 블루투스 4.0 제품이 소개되고 있다. 블루투스 4.0 중에서도 특히 BLE(Bluetooth Low Energy)는 애플이 아이비콘(iBeacon)을 소개함으로써 주목을 받는 저전력 통신 방법 중 하나이다. 하지만 BLE는 쉽고 빠른 블루투스 기기 연결을 위한 완전히 새로운 방법으로 이전 버전과 호환되지 않으므로 이전 버전과 함께 사용할 수 없다. 이 장에서는 블루투스는 2.0을 사용하여 ATmega328과 스마트폰을 연결하고 통신하는 방법에 대해 알아본다.

1. 블루투스 모듈 설정

블루투스는 마스터와 슬레이브 구성을 가지며 하나의 마스터에는 최대 7개까지 슬레이브가 연결될 수 있다. 블루투스에서는 마스터와 슬레이브 사이의 통신만이 가능하며 슬레이브 사이의 통신은 불가능하다. 블루투스는 스마트폰을 비롯하여 마우스, 키보드, 이어폰, 프린터, 카메라, 자동차, 게임기, TV, 장난감 등 수많은 기기에서 채택되어 주변에서 흔히 볼 수 있는 무선 연결 방식으로 자리 잡고 있다.

이 장에서는 시중에서 쉽게 구할 수 있는 HC-06 슬레이브 모듈을 사용하였다. HC-06 모듈은 VCC 및 GND의 전원과 시리얼 통신을 위한 RX 및 TX의 네 핀을 연결하여 사용한다. 블루투스는 무선 연결로 시리얼 통신과는 그 방식이 다르지만 블루투스 모듈에서 시리얼 통신과 동일한 방법으로 데이터를 송수신할 수 있는 방법을 제공하고 있으며 이를 SPP (Serial Port Profile)라고 한다. 이 장에서는 ATmega328을 스마트폰과 블루투스를 통해 무선으로 연결하여 데이터 송수신을 테스트할 것이며 이 때 ATmega328은 슬레이브로, 스마트폰은 마스터로 동작하게 된다.

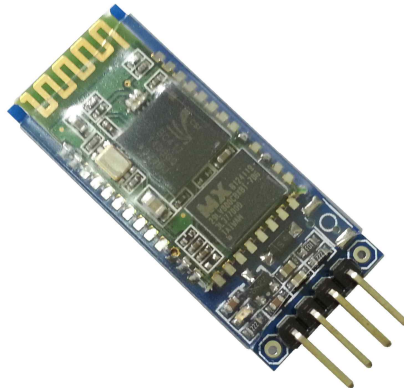


그림 1. HC-06 블루투스 슬레이브 모듈

먼저 블루투스 모듈을 초기화하는 방법을 살펴보자. HC-06 블루투스 모듈은 UART 시리얼 통신을 통해 초기화 가능하며 이를 위해서는 USB-UART 변환 장치가 필요하다. 프로그램 업로드를 위해 사용한 USB-UART 변환 장치를 HC-06 블루투스 모듈과 연결해 보자. 블루투스 모듈을 연결할 때 블루투스의 TX 및 RX는 USB-UART 변환 장치의 TX 및 RX와 교차하여 연결하여야 하는 점은 일반적인 시리얼 연결에서와 동일하다.

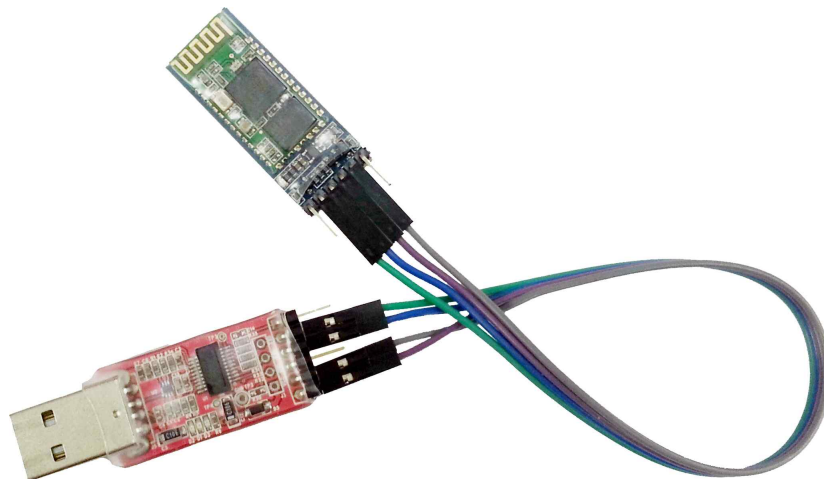


그림 2. HC-06 슬레이브 모듈과 USB-UART 변환 장치 연결

USB-UART 변환 장치를 컴퓨터에 연결하고 터미널 프로그램을 통해 블루투스 모듈을 설정할 수 있다. 터미널 프로그램은 여러 종류가 있지만 여기서는 무료로 사용할 수 있는 RealTerm을 사용한다. RealTerm 프로그램을 다운받아¹⁾ 설치해 보자. 설치가 완료되면 RealTerm 프

1) <http://realterm.sourceforge.net/>

로그래를 실행시킨다.

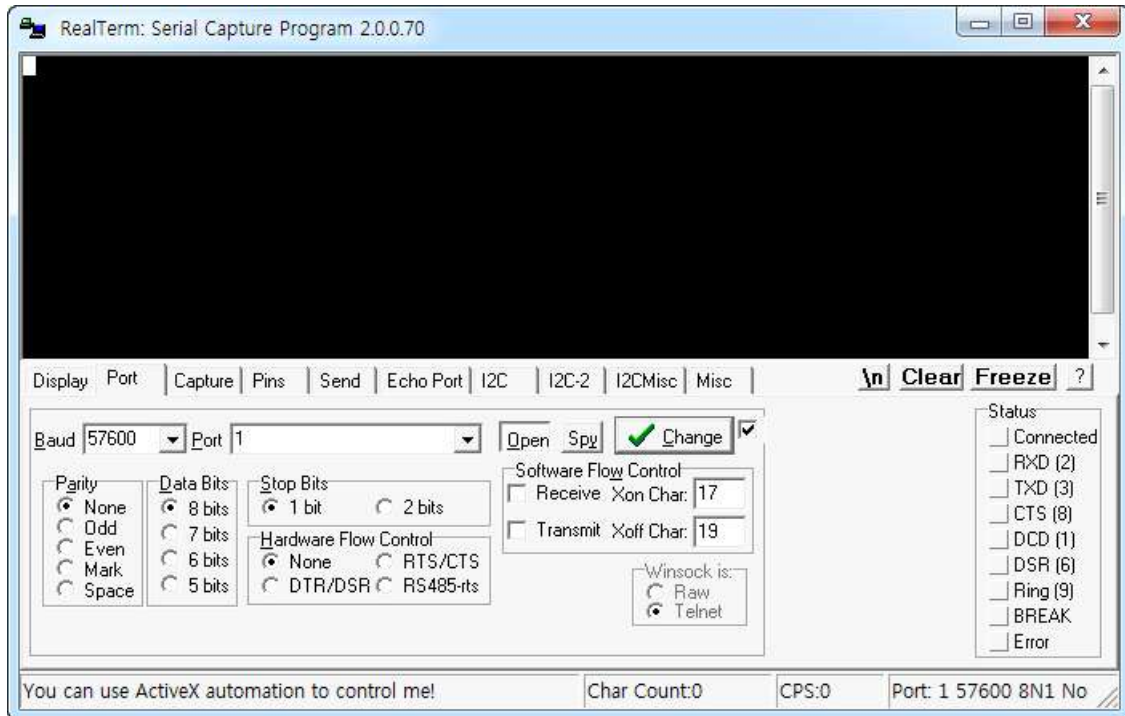


그림 3. RealTerm 실행 화면

먼저 'Port' 탭에서 Baud를 9600으로 Port를 USB-UART 변환 장치에 할당된 포트로 지정하고 'Change' 버튼을 누르면 지정한 포트와 속도로 연결된다. 상태 표시 부분이 바뀐 것으로 연결이 완료된 것을 확인할 수 있다.

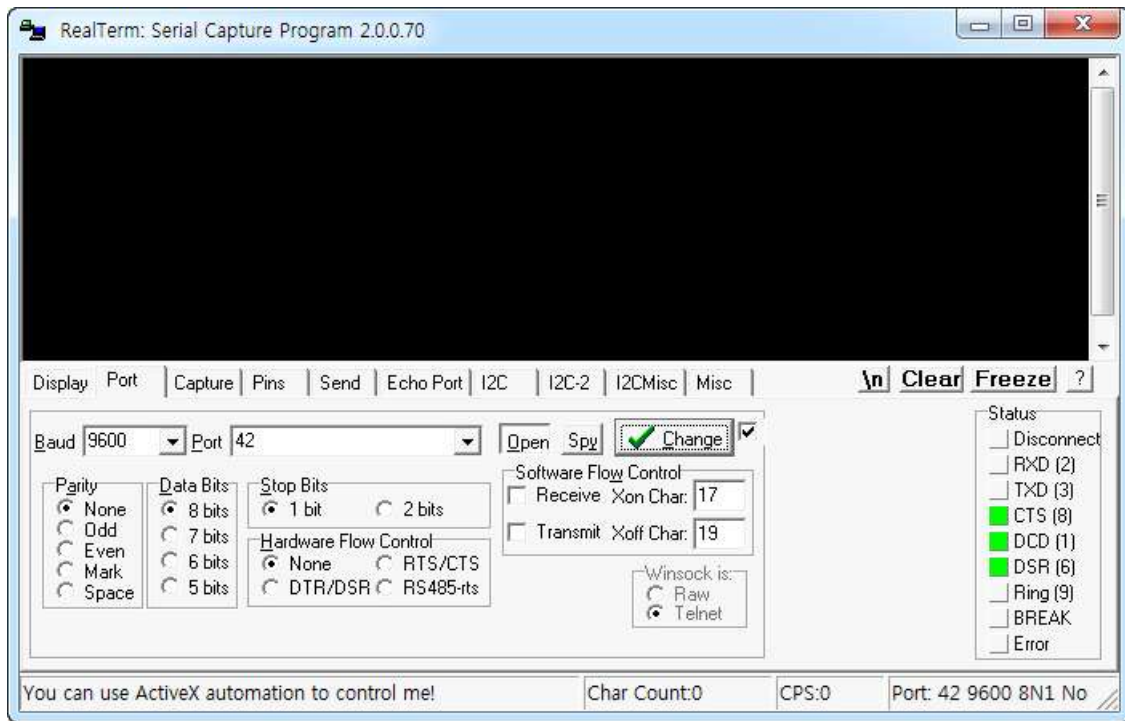


그림 4. 연결 완료

‘Send’ 탭에서 설정 명령어를 ASCII 형식으로 전송하면 블루투스 모듈은 설정 결과를 알려준다. ‘AT’ 명령을 블루투스 모듈로 전달하였을 때 ‘OK’를 반환하며 블루투스 모듈은 정상적으로 동작하는 것으로 생각할 수 있다. 스마트폰에 나타나는 블루투스 모듈의 이름은 ‘AT+NAMEAtmelBlue’ 명령을 전송하면 ‘AtmelBlue’로 이름이 변경되며 ‘AT+PIN1234’ 명령은 비밀번호를 ‘1234’로 변경하는 명령이다. 전송 속도는 ‘AT+BAUD4’ 명령으로 변경 가능하며 숫자에 따른 전송 속도는 표 1과 같다.

숫자	전송 속도(Baud)	숫자	전송 속도(Baud)
1	1200	5	19200
2	2400	6	38400
3	4800	7	57600
4	9600	8	115200

표 1. 전송 속도 설정

그림 5는 블루투스 모듈의 이름, 비밀번호, 통신 속도를 설정한 예를 보여주고 있다. 이 때 통신 속도를 기본 값인 9600에서 다른 값으로 변경하면 RealTerm에서도 해당 속도에 맞게

다시 연결하여야 추가 설정이 가능하다.

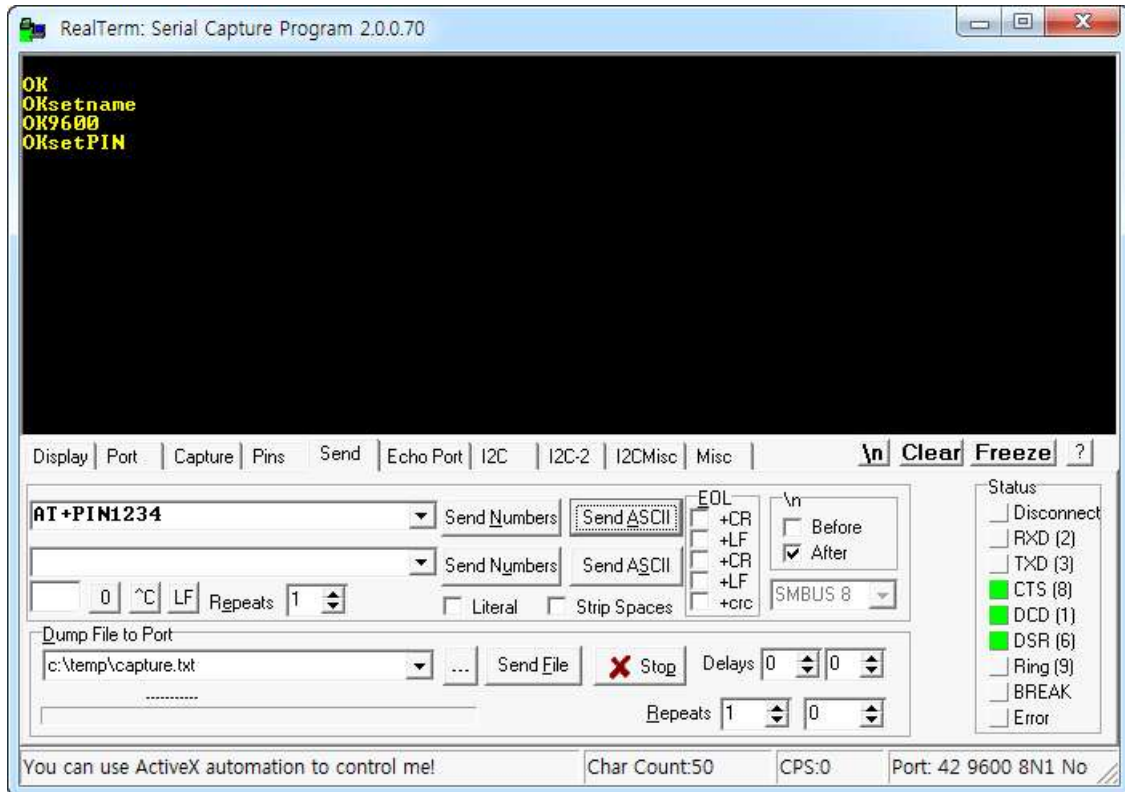


그림 5. 블루투스 모듈 설정

2. 스마트폰 설정

블루투스 모듈의 설정이 끝났으므로 이제 블루투스 통신을 수행할 수 있도록 스마트폰을 설정해 보자. 블루투스 모듈에는 전원이 공급되어 있어야 하므로 블루투스 모듈을 ATmega328의 하드웨어 시리얼 포트인 PD0과 PD1 핀(아두이노의 디지털 0번과 1번 핀)에 그림 6과 같이 연결한다. 모듈에 전원이 공급되고 연결이 되어있지 않은 상태에서는 LED가 깜빡거리고 있으며 연결이 된 후에는 켜져 있는 상태로 바뀐다.

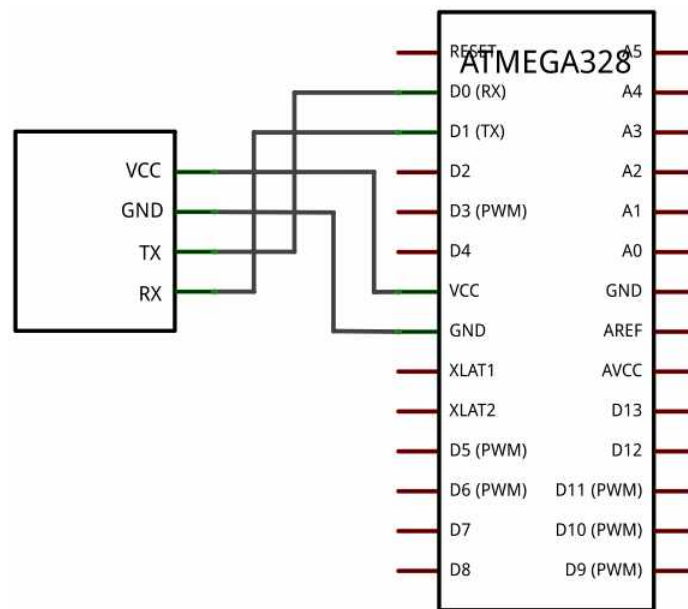


그림 6. ATmega328과 블루투스 슬레이브 모듈 연결

스마트폰의 블루투스를 활성화시키면 주변의 블루투스 기기들을 검색하여 보여준다.

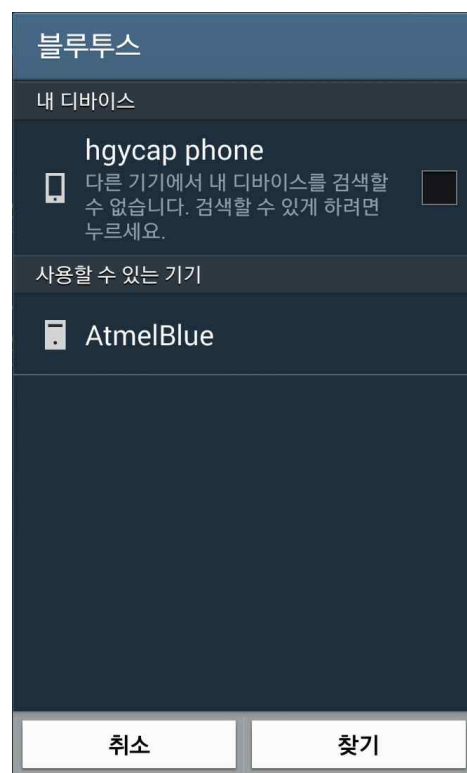


그림 7. 블루투스 기기 검색

검색 목록에서 블루투스 모듈의 이름으로 지정한 'AtmelBlue'를 확인할 수 있다. 블루투스 연결은 두 단계로 이루어진다. 먼저 연결 대상이 되는 기기들 사이의 인증 과정이 필요하며 이를 페어링(pairing)이라고 한다. 페어링 과정에서는 보안을 위해 비밀번호를 입력하도록 하고 있으며, 한 번 페어링이 된 기기는 스마트폰에서 관리되고 있으므로 페어링을 해제하기 전까지 비밀번호를 다시 입력할 필요는 없다. 페어링이 이루어진 이후에는 실제로 데이터를 주고받기 위한 연결이 진행될 수 있다. 그림 7에서 'AtmelBlue'를 선택하면 그림 8과 같이 비밀번호를 입력하는 창이 나타난다. 비밀번호는 PIN, Passcode, Pairing Code, Passkey 등으로 불린다.

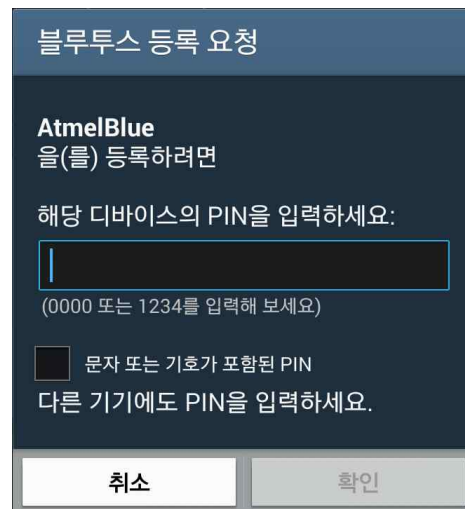
A screenshot of a mobile application interface for Bluetooth pairing. The title bar at the top is dark blue with the text '블루투스 등록 요청' (Bluetooth Registration Request) in white. Below the title bar, the text 'AtmelBlue' is displayed in bold, followed by '을(를) 등록하려면' (to register). The main instruction is '해당 디바이스의 PIN을 입력하세요:' (Enter the PIN of the device:). Below this is a text input field with a blue border. Underneath the input field, a note in parentheses says '(0000 또는 1234를 입력해 보세요)' (Try entering 0000 or 1234). There is a checkbox with a dark square icon, followed by the text '문자 또는 기호가 포함된 PIN' (PIN containing letters or symbols). Below this, it says '다른 기기에도 PIN을 입력하세요.' (Enter PIN for other devices as well). At the bottom, there are two buttons: '취소' (Cancel) on the left and '확인' (Confirm) on the right.

그림 8. 비밀번호 입력 창

그림 8의 창에 위에서 설정한 '1234'를 입력하면 페어링이 완료된다. 그림 6에서 '사용할 수 있는 기기'가 그림 9에서와 같이 '등록된 기기'로 바뀐 것으로 페어링이 이루어진 것을 확인할 수 있다.

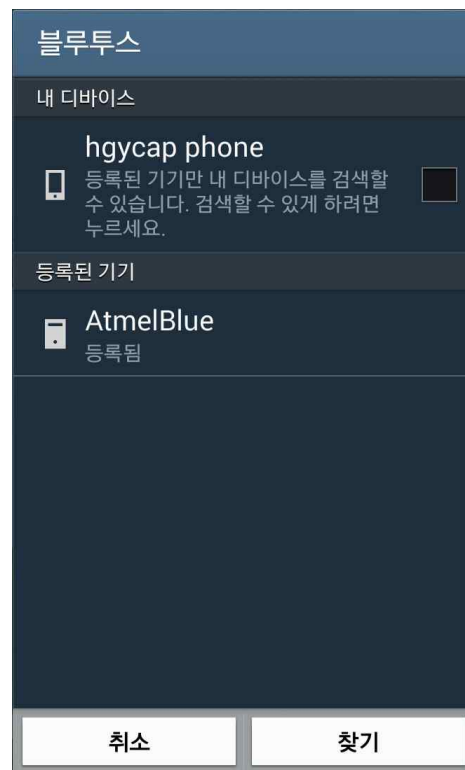


그림 9. 페어링 완료

SPP 방식의 데이터 송수신을 위해서는 ‘Bluetooth spp tools pro’ 어플리케이션을 사용한다. 어플리케이션은 플레이스토어에서 무료로 다운받을 수 있다²⁾. 어플리케이션을 설치하고 실행하면 현재 등록된 기기 목록을 보여준다.

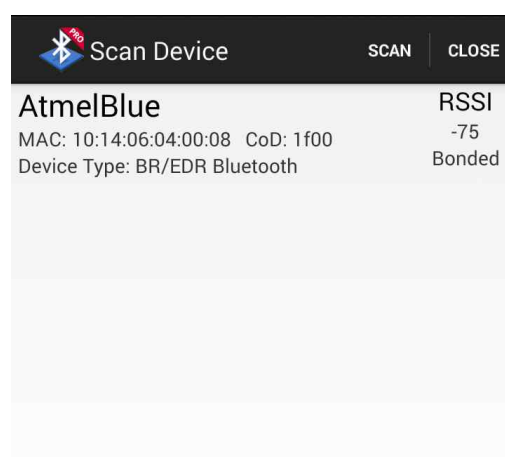


그림 10. 등록된 기기 목록

2) https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO&hl=ko

‘AtmelBlue’를 선택하고 ‘Connect’ 버튼을 누르면 데이터 송수신을 위한 실제 연결이 이루어진다.

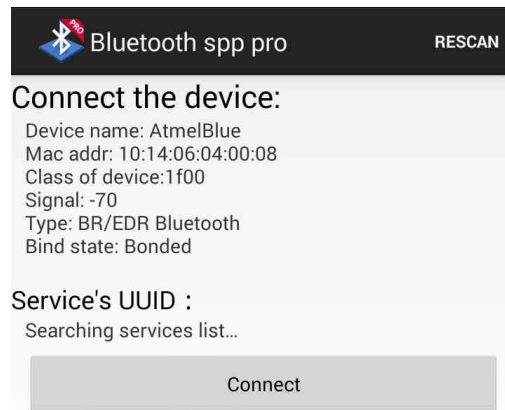


그림 11. 선택한 블루투스 장치와의 연결

연결이 되면 블루투스 모듈의 LED는 더 이상 깜빡이지 않고 켜져 있는 상태로 바뀐다. 통신 모드는 'Byte stream mode'를 선택하자.

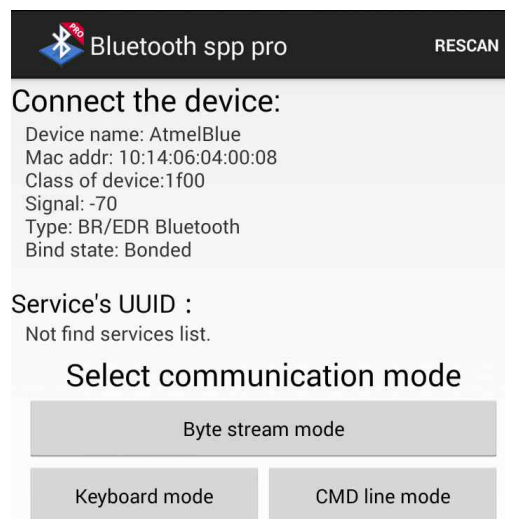


그림 12. 블루투스 장치와의 연결 모드 설정

모든 설정이 끝나면 데이터를 송수신할 수 있는 창이 나타난다. 윗부분은 수신된 데이터를 표시하는 부분이며 아랫부분은 송신할 메시지를 입력하는 부분으로 나뉘어져 있다.

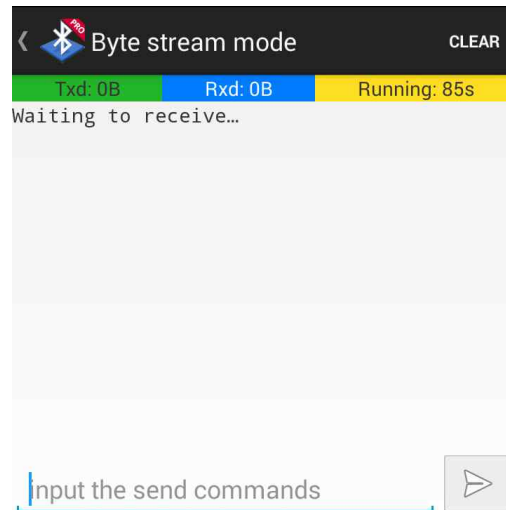


그림 13. Bluetooth spp tools pro 실행 화면

3. 블루투스 통신

그림 6에서 알 수 있듯이 블루투스 모듈은 UART 통신 포트에 연결되어 있고, HC-06 블루투스 모듈은 직렬 통신 프로파일인 SPP(Serial Port Profile)를 사용한다. 즉, 설정과 연결이 완료된 이후에는 UART 통신과 동일한 방법으로 통신이 가능하다. 코드 1은 스마트폰에서 전송된 문자열을 ATmega328에서 수신하여 다시 스마트폰으로 전송하는 코드이다.

코드 1. 블루투스 통신 테스트

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>
#include "UART.h"

int main(int argc, char *argv[])
{
    unsigned char data;

    UART_INIT();                // UART 통신 초기화
```

```

while (1) {
    data = UART_receive();           // 데이터 수신
    UART_transmit(data);             // 수신된 문자 재전송
}

return 0;
}

```

그림 14는 스마트폰의 어플리케이션에서 문자열을 전송한 경우 전송한 문자열이 다시 수신되어 화면에 나타나는 예를 보여준다.

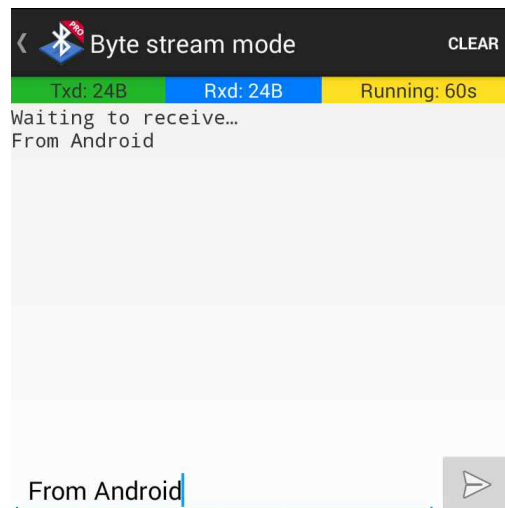


그림 14. 스마트폰 어플리케이션과의 블루투스 통신

4. 아두이노의 블루투스 통신

아두이노에서 블루투스 통신 방법 역시 간단하다. 코드 2는 코드 1을 아두이노 함수를 이용하여 작성한 예이다. 코드 1에서 UART_receive 함수에서는 데이터가 수신될 때까지 대기하는 기능이 있지만 아두이노 Serial 클래스의 read 함수에는 수신을 확인하는 기능이 없으므로 available 함수를 통해 데이터의 수신 여부를 확인하는 점에서 차이가 있다.

코드 2. 블루투스 통신 테스트 - 아두이노

```
void setup() {  
    Serial.begin(9600);           // UART 시리얼 통신 초기화  
}  
  
void loop() {  
    if(Serial.available()){       // 수신 데이터 확인  
        byte data = Serial.read(); // 데이터 수신  
        Serial.write(data);       // 수신된 문자 재전송  
    }  
}
```

그림 6의 회로도를 이용하는 경우 한 가지 주의할 점이 있다. 아두이노에서 USB 연결을 통해 프로그램을 업로드 하는 경우 블루투스 모듈이 연결된 하드웨어 시리얼 포트와 동일한 포트를 사용한다. 따라서 블루투스 모듈이 연결되어 있는 경우에는 USB를 통한 프로그램 업로드에 실패할 수 있으므로 블루투스 모듈을 제거하고 프로그램을 업로드한 후 다시 블루투스 모듈을 연결하고 테스트를 진행하여야 한다. 이는 ATmega328에서 하나의 UART 연결만을 사용할 수 있기 때문에 발생하는 문제이다. 이를 해결하기 위해 아두이노에서는 소프트웨어에 의한 UART 통신을 지원하는 SoftwareSerial 클래스를 지원하고 있다.

SoftwareSerial 클래스는 아두이노의 기본 라이브러리 중 하나로 하드웨어 시리얼 포트가 아닌 일반 입출력 핀을 통해 UART 통신을 사용할 수 있도록 해주는 클래스이다. 아두이노 우노의 경우 모든 핀을 소프트웨어 시리얼 클래스에서 사용할 수 있다. SoftwareSerial 클래스를 사용하면 여러 개의 UART 통신을 사용할 수 있지만 한 번에 하나의 통신만이 가능하다는 한계는 있다. 또한 하드웨어로 구현된 UART 통신에 비해 안정성이 떨어지는 것으로 알려져 있다. SoftwareSerial 클래스는 Serial 클래스와 사용 방법이 거의 동일하다. 다만 SoftwareSerial 클래스의 객체를 생성할 때 RX와 TX로 사용할 핀의 번호를 지정해주어야 한다는 점은 다르다. 하드웨어 시리얼의 경우 디지털 0번과 1번이 RX와 TX로 지정되어 있어 별도로 사용 핀의 번호를 지정할 필요가 없다. 생성자에서 `inverse_logic`은 반전 논리의 사용 여부를 나타내는 값으로 논리 1을 0V로, 논리 0을 5V로 나타내도록 하기 위해 사용한다.

```
SoftwareSerial(uint8_t rx, uint8_t tx, bool inverse_logic = false)
```

rx : 수신 핀 번호

tx : 송신 핀 번호

inverse_logic : 송수신 비트 반전

반환값 : 없음

그림 15와 같이 블루투스 모듈을 디지털 2번과 3번 핀에 연결해 보자.

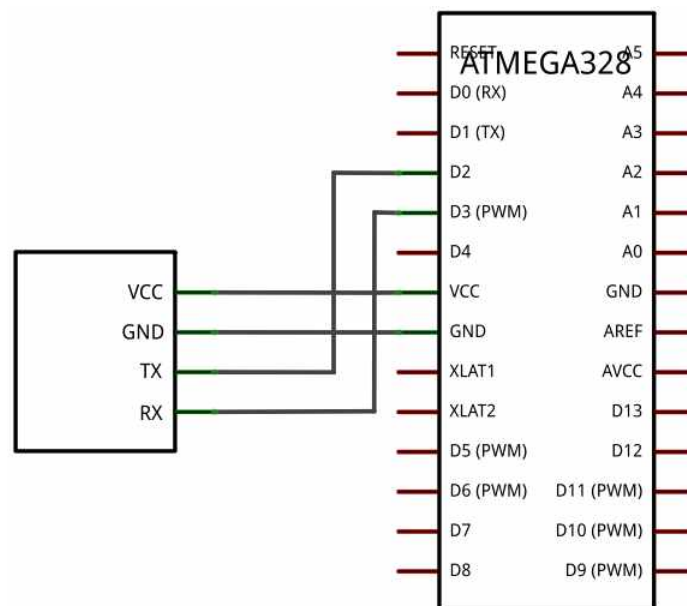


그림 15. 소프트웨어 시리얼 사용을 위한 ATmega328과 블루투스 슬레이브 모듈 연결

코드 3은 두 개의 UART 통신을 사용하는 예이다. 하나는 블루투스 모듈이 소프트웨어 시리얼을 통해 연결되고, 다른 하나는 컴퓨터와 하드웨어 시리얼을 통해 연결된다. 이 때 아두이노 우노는 컴퓨터와 스마트폰의 데이터를 중개하는 역할을 하게 된다.

코드 3. 소프트웨어 시리얼 테스트

```
#include <SoftwareSerial.h>
```

```
// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결된다.
```

```
// 2 -> TX of Bluetooth Module
// 3 -> RX of Bluetooth Module
SoftwareSerial BTSerial(2, 3);

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

void loop()
{
  // 블루투스 모듈 --> 아두이노 --> 컴퓨터
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // 컴퓨터 --> 아두이노 --> 블루투스 모듈
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

코드 3을 실행시키고 스마트폰에서 문자를 입력하면 시리얼 모니터에 문자가 출력되며, 시리얼 모니터에 문자를 입력하면 스마트폰에 나타남을 확인할 수 있다.

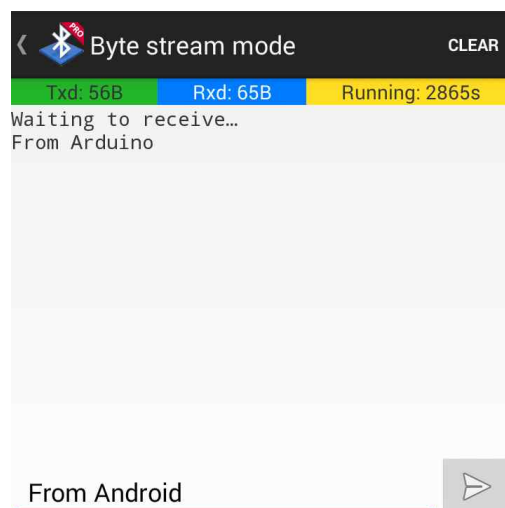


그림 16. 코드 3 실행 결과 - 스마트폰

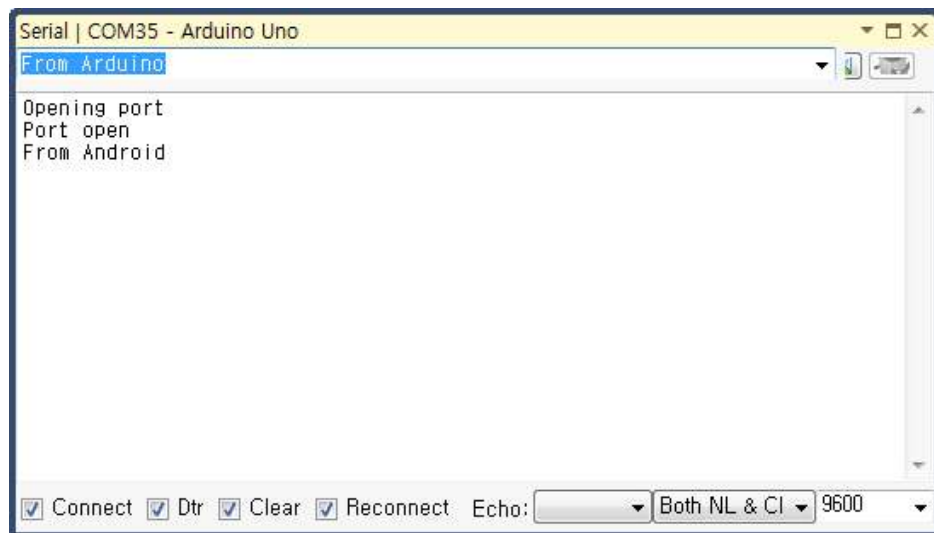


그림 17. 코드 3 실행 결과 - 시리얼모니터