

Chapter 237. 블루투스

1. 블루투스

블루투스(Bluetooth)는 1994년 에릭슨(Ericsson)이 개발한 개인 근거리 무선 통신 (Personal Area Network, PAN)을 위한 표준으로 RS-232를 대체하기 위한 저가격, 저전력의 무선 기술로 개발되었다. 블루투스는 기본적으로 10m 이내에서의 통신을 목표로 하지만 통신 범위는 100m까지 확장이 가능하다. 블루투스는 2.4GHz 대역인 ISM (Industrial, Scientific, Medical) 대역을 사용한다. ISM 대역은 산업, 과학 및 의료 목적으로 할당된 대역으로 전파 사용에 대한 허가를 받을 필요가 없어 저전력의 전파를 발산하는 개인 무선기기에 많이 사용되고 있으며 무선랜과 지그비(ZigBee) 역시 ISM 대역을 사용한다. 블루투스는 기기 간의 통신 방식을 정의하기 위해 프로파일(profile)을 사용한다. 프로파일이란 블루투스를 이용한 어플리케이션을 구현할 때 특정 어플리케이션에서 사용해야 하는 프로토콜의 종류와 구조 및 사용 방법을 정의한 것을 가리킨다. 따라서 프로파일은 특정 어플리케이션을 개발할 때의 기준을 제시하는 역할을 하게 되며, 프로파일에 따라 제작된 어플리케이션은 모두 호환될 수 있는 장점이 있다. 블루투스 표준을 관리하는 Bluetooth SIG(Special Interest Group)에서는 어플리케이션에서 사용할 수 있는 다양한 프로파일을 정의하고 있으며 여기에는 파일 전송 프로파일, 전화 접속 네트워크 프로파일, 팩스 프로파일, 시리얼 포트 프로파일 등이 포함되어 있다. 이 중 가장 기본이 되면서 다른 프로파일의 기초가 되는 프로파일은 일반 액세스 프로파일(generic access profile)로 블루투스 장치를 연결하기 위해 연결 대상을 발견하고 연결을 설정하는 방법 및 이와 관련된 보안 관련 내용을 규정하고 있다. 이들 프로파일 중 이 책에서 사용하는 프로파일은 시리얼 포트 프로파일(Serial Port Profile, SPP)로 시리얼 통신을 에뮬레이션하기 위해 사용되는 프로파일로, 마이크로컨트롤러에서 사용하는 경우 UART 통신 코드를 약간만 수정하고 블루투스 모듈을 연결하면 바로 블루투스 통신이 가능하다.

현재 흔히 사용되고 있는 블루투스는 2.x 버전이다. 블루투스 3.0은 2009년 블루투스 2.1의 후속으로 고속통신 기능이 추가된 것이지만 고속 전송을 위해서는 WiFi를 사용하므로 WiFi Direct와의 차별성 부족으로 널리 사용되지는 못하고 있다. 2010년 제정된 블루투스 4.0은 저전력 블루투스(Bluetooth Low Energy, BLE)로 인해 주목을 받고 있다. 하지만 블루투스 4.0에는 서로 호환되지 않는 블루투스 2.x까지의 기술인 클래식 블루투스(Classic Bluetooth)와 BLE가 합해진 형태를 띠고 있다. 블루투스 4.0을 지원하는 기기 중 BLE만을 지원하는 기기는 'Bluetooth Smart'라고 표시하며, 클래식 블루투스와 BLE를 모두 지원하여 하위 호환성을

가지는 기기는 'Bluetooth Smart Ready'라고 구별하여 표시하므로 혼돈하지 말아야 한다. BLE는 SPP를 지원하지 않으며 이 책에서도 블루투스 2.x 버전을 지원하는 모듈을 사용하여 SPP 프로파일을 통한 통신을 테스트 할 것이다.



(a) Bluetooth Smart



(b) Bluetooth Smart Ready

그림 1. 블루투스 로고¹⁾

블루투스는 마스터와 슬레이브 구성을 가지며 하나의 마스터 기기에는 최대 7개 까지 슬레이브 기기를 연결할 수 있다. 블루투스에서는 마스터와 슬레이브 사이의 통신만이 가능하고 슬레이브 사이의 통신은 불가능하다.

2. HC-06 블루투스 모듈

이 장에서 사용할 블루투스 모듈은 HC-06 블루투스 모듈로 SPP를 지원하므로 RX(receive)와 TX(transmit)의 2개 연결선을 통해 무선 통신을 수행할 수 있다. 하지만 HC-06 모델은 1:1 통신만을 지원하므로 하나의 슬레이브 기기만을 연결할 수 있는 한계는 있다.

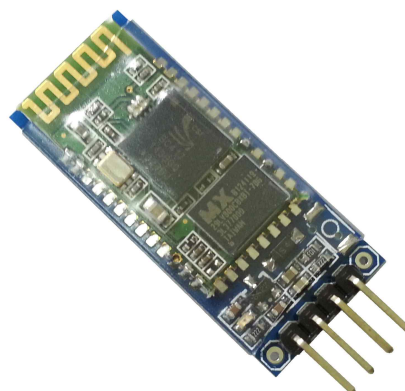


그림 2. HC-06 블루투스 슬레이브 모듈

1) 이미지 출처: <https://www.bluetooth.org>

먼저 블루투스 모듈을 설정하는 방법을 살펴보자. 블루투스 모듈을 그림 3과 같이 연결하자. HC-06 블루투스 모듈은 SPP 프로파일을 사용하므로 아두이노의 Serial 또는 SoftwareSerial을 통해 통신을 수행할 수 있다. 그림 3에서는 프로그램 업로드와의 충돌을 방지하기 위해 디지털 2번과 3번 핀에 연결하여 SoftwareSerial을 사용한다.

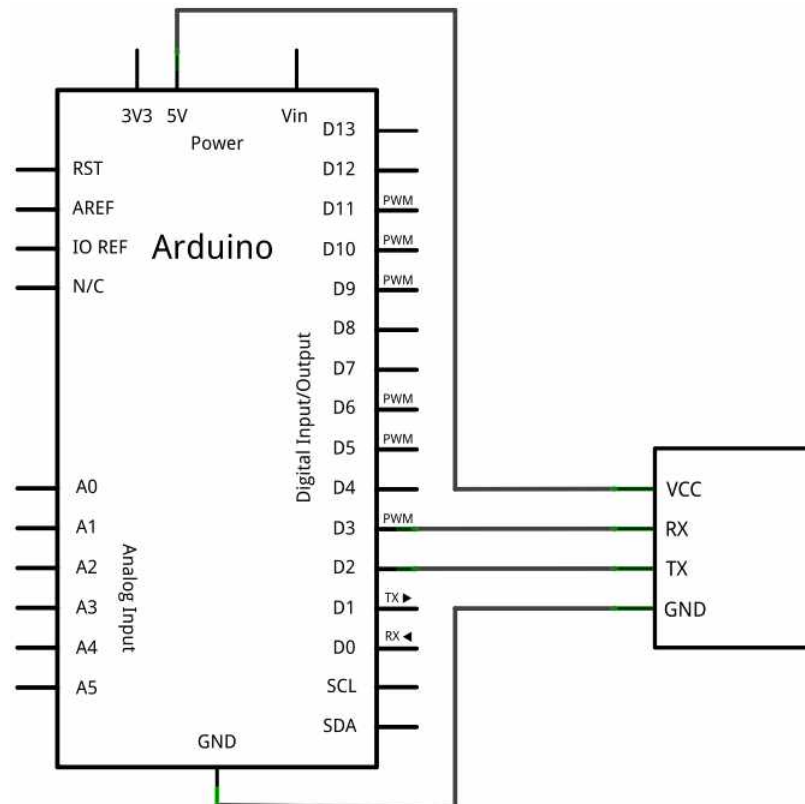


그림 3. 블루투스 모듈 연결 회로도

스케치 1을 아두이노에 업로드 하자. 스케치 1은 시리얼 모니터로 입력되는 데이터를 블루투스 모듈로 전달하고, 블루투스 모듈에서 출력되는 데이터를 시리얼 모니터로 출력하는 스케치이다.

스케치 1. 블루투스 모듈 테스트

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

```
// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결
SoftwareSerial BTSerial(2, 3);

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

void loop()
{
  // 블루투스 모듈 → 아두이노 → 시리얼 모니터
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  // 시리얼 모니터 → 아두이노 → 블루투스 모듈
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

블루투스 모듈의 설정을 위해서는 특별한 명령어를 블루투스 모듈로 전달하여야 하며, 이들 명령어를 AT 명령어라고 한다. 정상적으로 연결이 되었다면 시리얼 모니터에서 'AT'를 입력해 보자. 'AT'는 블루투스 모듈로 전달되고, 'AT' 명령을 전달받은 블루투스 모듈은 현재 상태를 반환한다. 블루투스 모듈에 아무런 이상이 없다면 'OK' 문자열을 반환하며 반환된 문자열은 시리얼 모니터에 표시된다. 단, HC-06 블루투스 모듈은 디폴트값으로 9600 보율의 속도로 설정되어 있으므로 스케치 1에서도 9600 보율로 설정하고 있음에 유의하여야 한다. 또한 시리얼 모니터에서 'No line ending'이 선택되어 있어야 하며 개행문자가 추가되면 명령이 정상적으로 전달되지 않는다는 점도 기억해야 한다.

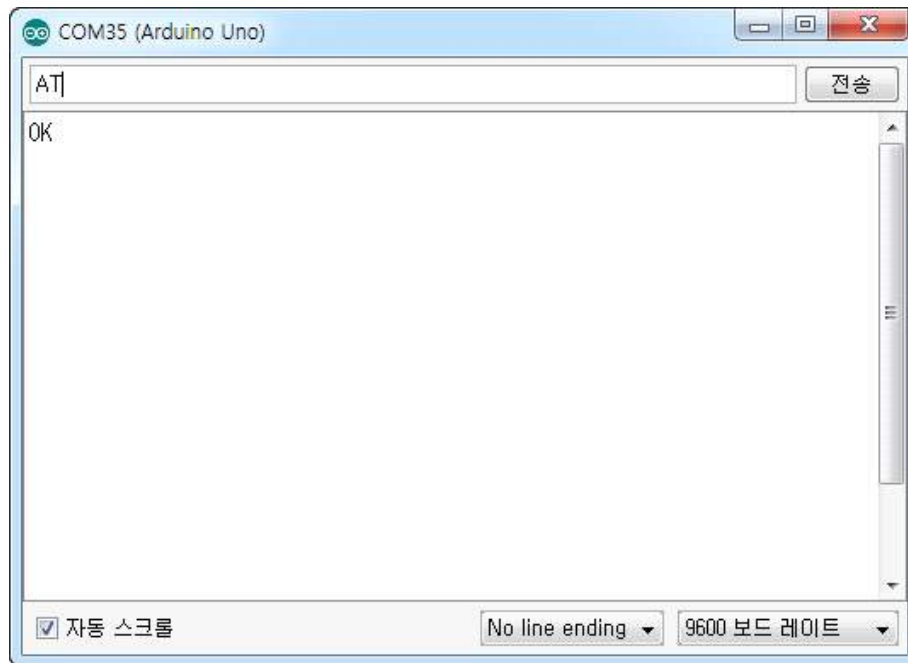


그림 4. 블루투스 동작 확인

HC-06 블루투스 모듈 설정을 위해서는 표 1의 명령어들을 사용할 수 있다.

명령어	사용 방법	반환값	비고
AT	AT	OK	
AT+NAME	AT+NAME <u>AtmelBlue</u>	OKsetname	이름 설정
AT+PIN	AT+PIN <u>1111</u>	OKsetPIN	PIN 설정
AT+BAUD	AT+BAUD <u>4</u>	OK9600	통신 속도 설정

표 1. HC-06 설정 AT 명령어

NAME은 모듈의 이름을 설정하기 위해 사용된다. PIN은 연결 코드, 패스 코드, 핀 코드 등으로도 불리며 페어링 과정에서 사용되는 일종의 비밀번호 역할을 한다. 페어링이 이루어진 이후에는 다시 핀을 사용할 필요는 없다. BAUD는 통신 속도를 설정하기 위해 사용되며 숫자에 따른 통신 속도는 표 2와 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 2. 전송 속도

표 1의 명령어를 사용하여 블루투스 모듈을 설정한 결과는 그림 5와 같다.

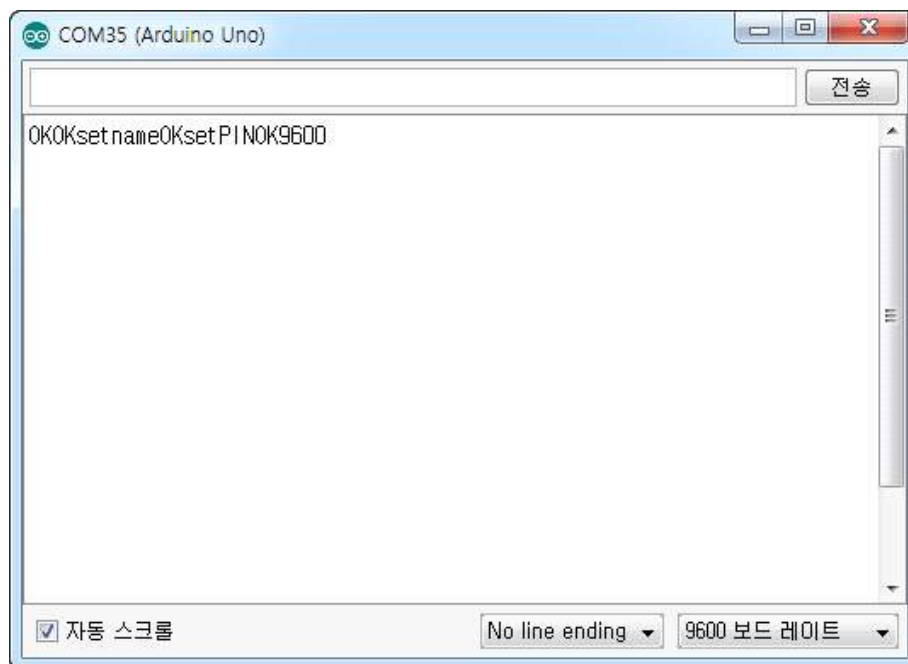


그림 5. 블루투스 모듈 설정 결과

3. 컴퓨터의 블루투스 설정

최근 판매되는 노트북에는 블루투스가 지원되는 경우가 대부분이다. 블루투스를 지원하지 않는 컴퓨터라면 블루투스 동글을 설치함으로써 블루투스를 사용할 수 있다. 블루투스 동글을 연결하고 드라이버를 설치하자. 설치가 완료되었으면 '제어판 → 하드웨어 및 소리 → Bluetooth 장치 추가'를 선택해 보자.²⁾

2) 이 책에서는 Windows 7 64bit를 기준으로 설명한다.



그림 6. 블루투스 장치 추가

블루투스 모듈에 전원이 공급되고 있다면 모듈의 이름으로 설정한 'AtmelBlue'가 목록에 표시된다.

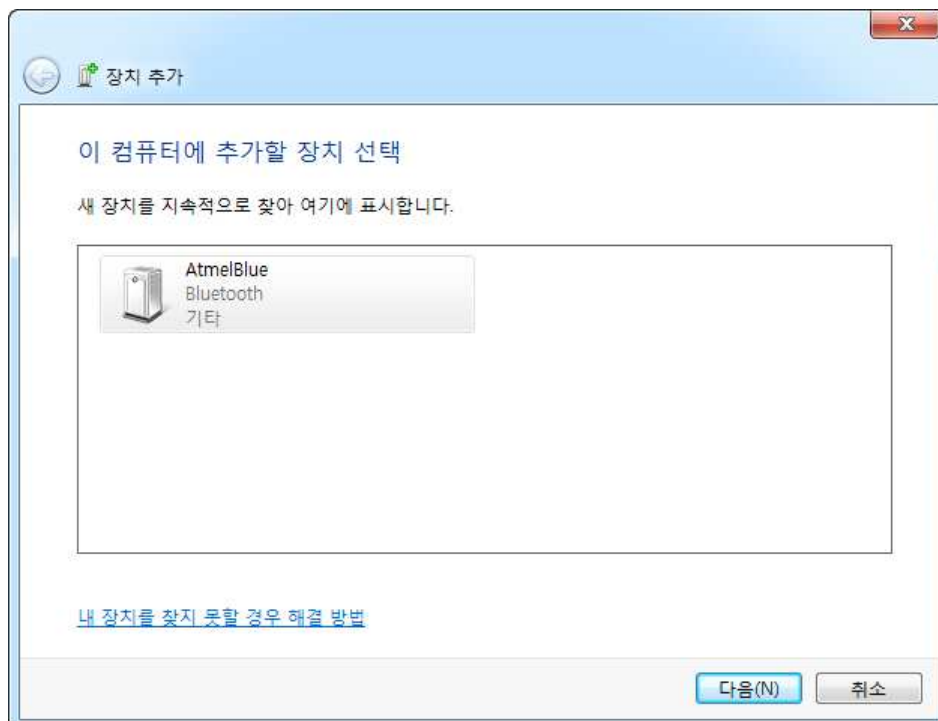


그림 7. 블루투스 장치 추가

‘AtmelBlue’를 선택하고 ‘다음’ 버튼을 누르면 연결 옵션을 선택하는 창이 출력된다.



그림 8. 블루투스 장치의 연결 옵션 선택

모듈 설정 과정에서 핀 코드를 설정하였으므로 ‘장치의 연결 코드 입력’을 선택하고 핀 코드를 입력한다.

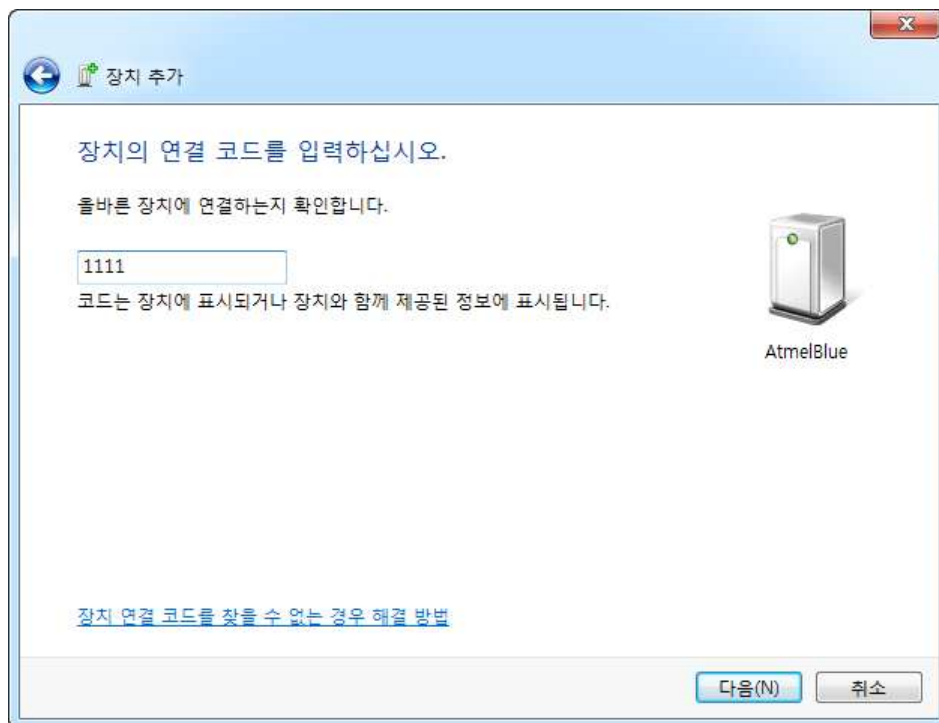


그림 9. 핀 코드 입력

핀 코드를 입력하면 장치가 완료된다.



그림 10. 블루투스 장치 추가 완료

추가된 블루투스 장치는 '제어판 → 하드웨어 및 소리 → 장치 및 프린터'에서 확인할 수 있다.

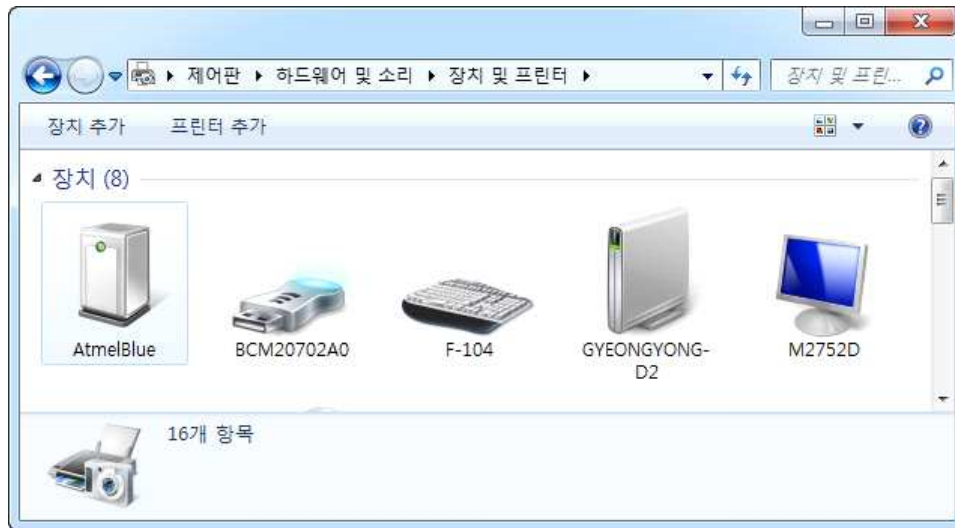


그림 11. 장치 및 프린터 목록

'AtmelBlue'를 선택하고 마우스 오른쪽 버튼을 눌러 '속성'을 선택해 보자. '서비스' 탭을 선택하면 AtmelBlue 모듈과 SPP 통신을 수행할 수 있는 포트 번호를 확인할 수 있다.

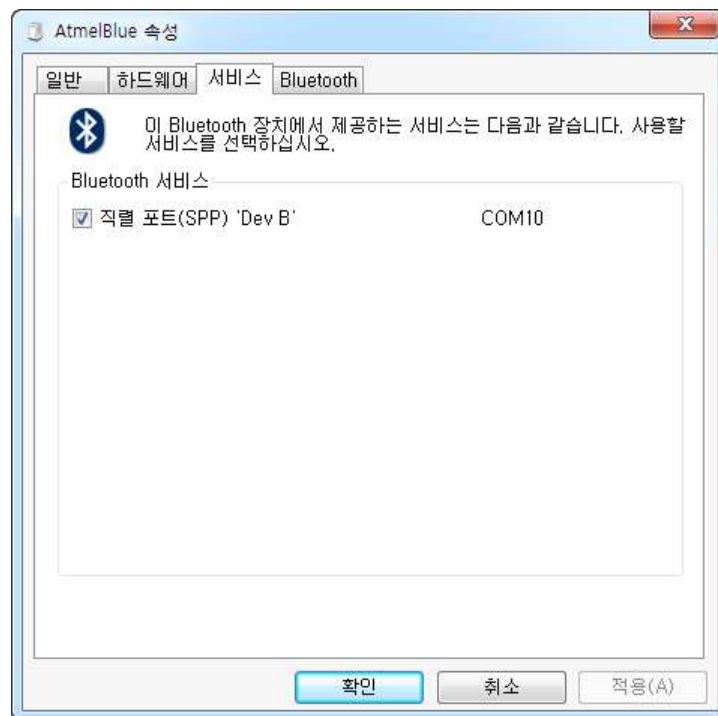


그림 12. AtmelBlue 모듈과 통신을 위한 직렬 포트

4. 컴퓨터와 아두이노의 블루투스 통신

HC-06 블루투스 모듈 설정과 컴퓨터의 블루투스 동글 설정이 완료되었으니 컴퓨터와 아두이노 사이에 블루투스를 이용하여 데이터를 주고받도록 해보자. 아두이노 우노가 컴퓨터는 두 가지 방법으로 연결되어 있다. 하나는 프로그램 업로드를 위해 사용되는 USB 유선 연결로 시리얼 모니터와 연결되어 있다. 다른 하나는 블루투스를 통한 무선 연결이다.

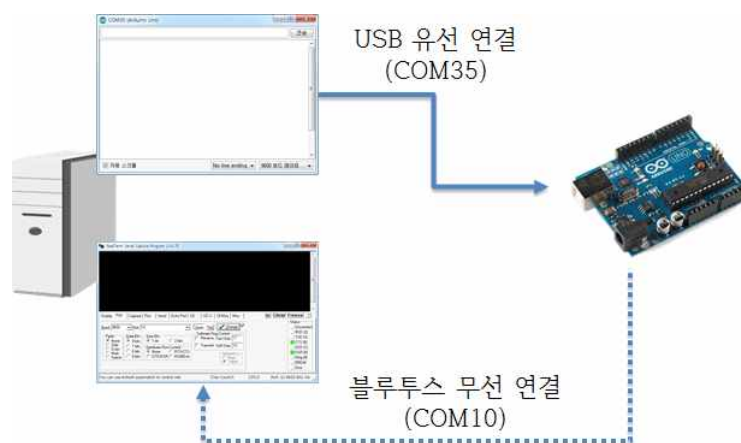


그림 13. 컴퓨터와 아두이노 우노의 유무선 연결

시리얼 모니터를 통해 문자열을 입력하면 이를 아두이노 우노에서 블루투스로 컴퓨터로 다시 전송하도록 해보자. 블루투스를 통해 수신된 데이터는 RealTerm을 사용하여 출력하도록 한다. 반대로 RealTerm에서 입력하는 데이터 역시 시리얼 모니터로 출력된다. 아두이노 우노에는 스케치 1을 업로드 한다.

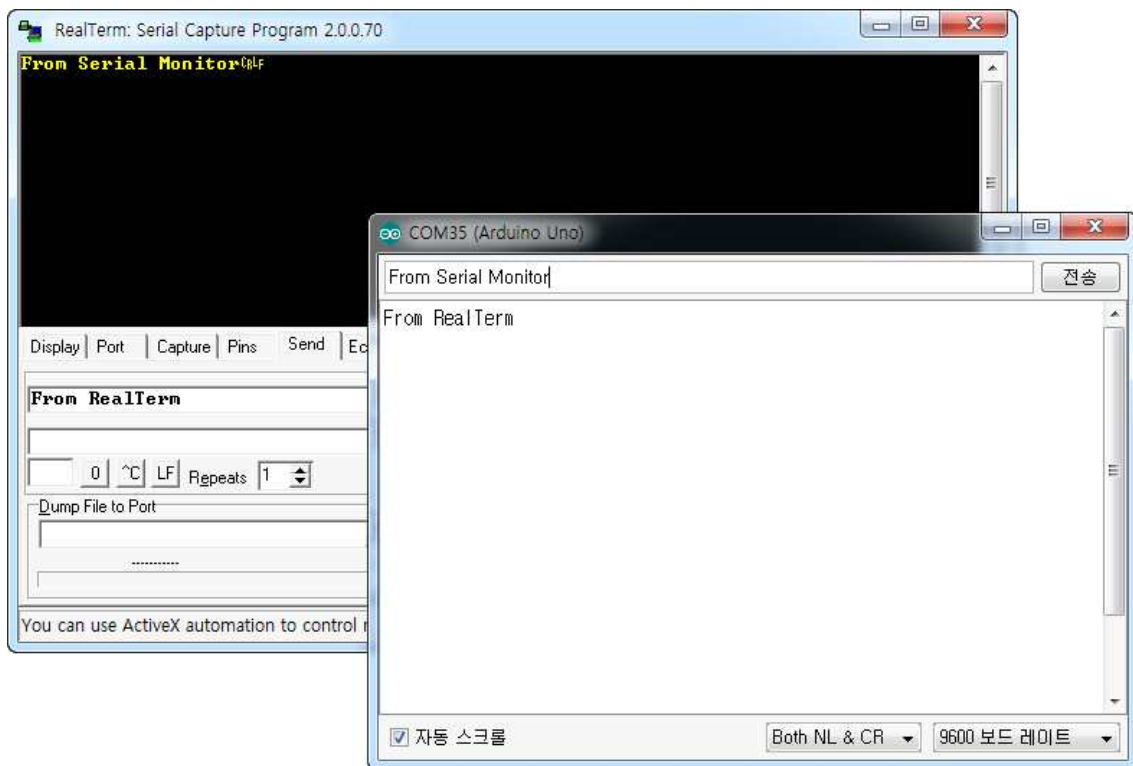


그림 14. 컴퓨터와 아두이노 우노 사이의 유무선 통신

RealTerm으로 0에서 255 사이의 정수를 전송하고, 아두이노 우노에서는 수신된 정수를 이진수로 변환하여 비트 단위로 아두이노 우노에 연결된 8개의 LED를 제어하도록 해보자. 아두이노 우노에는 그림 15와 같이 8개의 LED를 연결한다. 블루투스 모듈을 디지털 2번과 3번에 연결하였으므로 LED는 디지털 4번부터 11번까지 연결하였다.

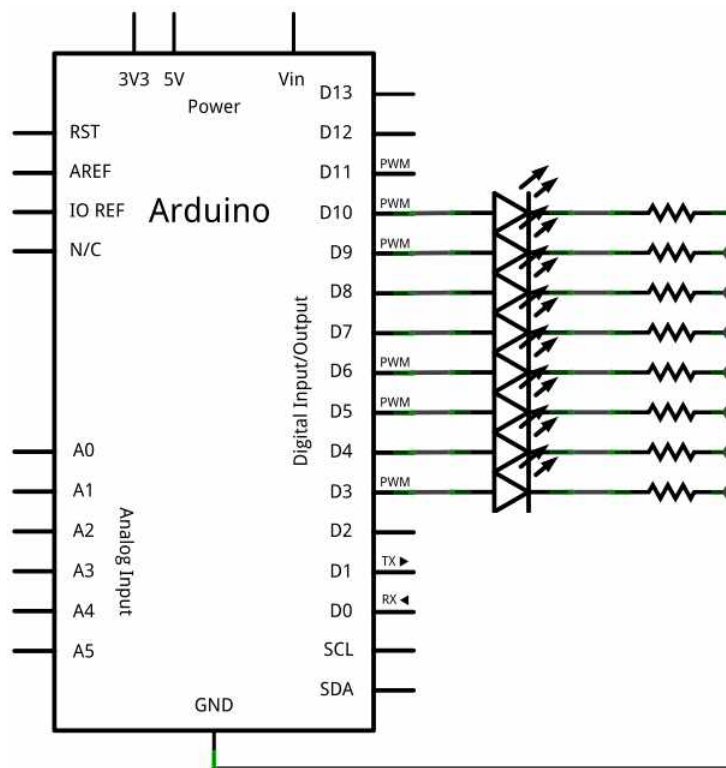


그림 15. 8개 LED 연결 회로도

스케치 2는 블루투스로 전송된 데이터를 바탕으로 LED를 제어하는 스케치의 예이다.

스케치 2. 블루투스를 통한 LED 제어

```
#include <SoftwareSerial.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결
SoftwareSerial BTSerial(2, 3);
int pins[] = {4, 5, 6, 7, 8, 9, 10, 11};    // LED 연결 핀

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

    for(int i = 0; i < 8; i++){              // LED 제어 핀 설정
```

```
    pinMode(pins[i], OUTPUT);
    digitalWrite(pins[i], LOW);
  }
}

void loop()
{
  if(BTSerial.available()){                // 블루투스로 데이터 수신
    byte no = BTSerial.read();
    if(no < 0 || no > 255)                  // 범위 검사
      Serial.println("** Wrong Number...");
    else{
      for(int i = 0; i < 8; i++){          // LED 제어
        if( no & (1 << i) )
          digitalWrite(pins[i], HIGH);
        else
          digitalWrite(pins[i], LOW);
      }
    }
  }
}
```