

Chapter 223. 아두이노-아두이노 블루투스 연결

블루투스(Bluetooth)는 RS-232의 유선 통신을 대체하기 위해 만들어진 저가격, 저전력의 무선 통신 기술로 다양한 장치들이 블루투스를 사용하고 있다. 특히 스마트폰의 보급에 힘입어 다양한 스마트폰 액세서리가 블루투스를 이용하여 출시되는 등 블루투스의 이용 범위는 점점 늘어나고 있다. 여기에 최근 사물 인터넷 붐과 더불어 주목받고 있는 블루투스 4.0 저전력 블루투스(Bluetooth Low Energy, BLE)는 향후 다양한 응용이 가능할 것으로 기대되고 있다. 하지만 BLE는 하위 호환성이 없는 기술로 마이크로컨트롤러에서 흔히 사용되는 SPP(Serial Port Profile)과 함께 사용될 수 없으므로 간단한 무선 통신을 위해서는 아직도 2.x 버전의 블루투스가 흔히 사용되고 있으며, 이 책에서도 블루투스 2.x 버전의 SPP를 통한 무선 통신 방식을 다룬다.

1. 블루투스 마스터 모듈

블루투스는 마스터-슬레이브 구조를 가지고 있다. 이전 장에서 컴퓨터와 아두이노를 블루투스로 연결하기 위해 HC-06 모듈을 사용하였으며 이 때 컴퓨터는 마스터로, 아두이노는 슬레이브로 동작하였다. HC-06 모듈의 경우 마스터 모듈과 슬레이브 모듈이 별도로 존재한다. HC-06 마스터 모듈의 모양은 슬레이브 모듈과 거의 유사하지만 슬레이브 모듈에 비해 더 많은 핀을 가지고 있다.

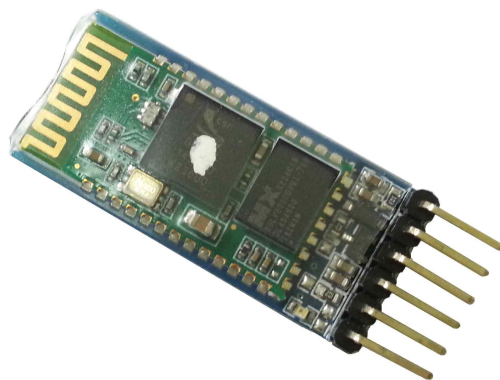


그림 1. HC-06 마스터 모듈

6개의 핀 중 4개는 슬레이브 모듈에서와 동일하게 VCC, GND, RX, TX에 해당하며 나머지

중 하나는 KEY 핀이다. KEY는 마지막에 페어링 된 슬레이브 정보를 삭제하기 위해 사용되며 VCC에 연결하여서도 페어링 정보를 삭제할 수 있다.¹⁾ 그림 2와 같이 마스터 모듈을 아두이노와 연결해 보자. 그림 2에서는 KEY 핀을 별도로 연결하지 않았으므로 슬레이브 모듈을 아두이노와 연결하는 방법과 동일하다.

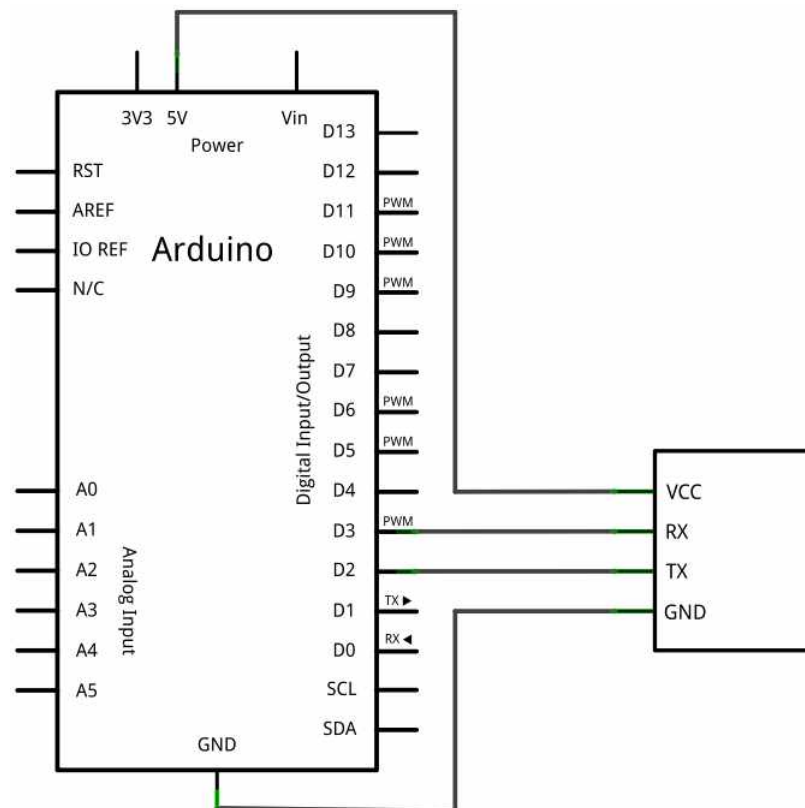


그림 2. 블루투스 마스터 모듈 연결 회로도

아두이노에는 스케치 1을 업로드 한다. 스케치 1은 슬레이브 모듈 설정을 위해 사용한 코드와 동일하다.

스케치 1. 블루투스 모듈 설정

```
#include <SoftwareSerial.h>

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

1) 최근 KEY 핀이 존재하지 않고 페어링 정보 삭제를 위한 버튼이 추가된 모듈도 판매되고 있다.

```

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());
}

```

HC-06 블루투스 마스터 모듈 설정을 위해 사용할 수 있는 AT 명령어는 표 1과 같다. 슬레이브 모듈과 달리 마스터 모듈은 이름을 바꿀 수 없는 점을 제외하면 슬레이브 모듈 설정에 사용하는 명령어와 동일하다.

명령어	사용 방법	반환값	비고
AT	AT	OK	
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 마스터 설정 AT 명령어

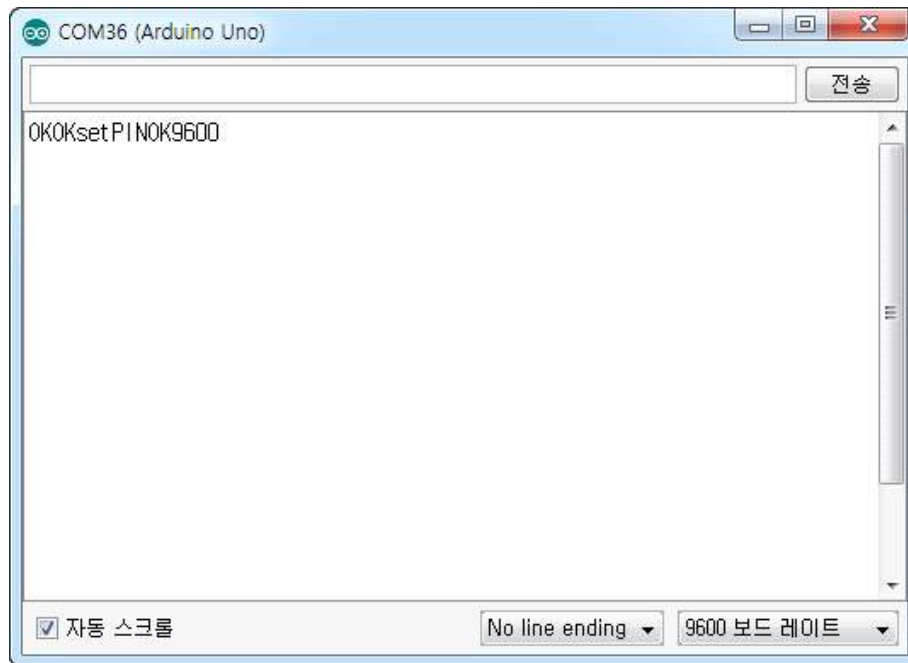


그림 3. 블루투스 마스터 모듈 설정 결과

마스터 모듈 설정에서 주의할 점은 마스터 모듈의 핀 코드를 슬레이브 모듈의 핀 코드와 동일하게 설정해야 한다는 점이다. 블루투스 마스터에는 여러 개의 슬레이브 연결이 가능하지만 HC-06 마스터 모듈은 하나의 슬레이브만 연결될 수 있다. 마스터와 슬레이브에 동일한 핀 코드가 설정되어 있으면 마스터와 슬레이브는 자동으로 연결되며 마스터에는 슬레이브와의 페어링 정보가 저장된다. 따라서 다른 장치와 연결하고자 한다면 KEY 핀을 통해 슬레이브 정보를 제거하여야 한다.

두 개의 아두이노에 각각 마스터와 슬레이브 모듈을 연결하고 전원을 연결해 보자. 마스터 모듈이 연결된 아두이노는 COM36에 연결되어 있고, 슬레이브 모듈이 연결된 아두이노는 COM35에 연결되어 있는 것으로 가정한다. 슬레이브 모듈의 붉은색 LED가 빠른 속도로 깜빡거리다 마스터와 연결이 완료되면 계속 켜져 있는 것을 확인할 수 있다. 마스터 모듈의 경우 이전 페어링 정보가 없는 상황에서 연결을 대기하는 상황이라면 빠른 속도로 LED가 깜빡이고, 이전 페어링 정보가 있는 상황에서 연결을 대기하는 상황이라면 느린 속도로 LED가 깜빡인다. 연결이 이루어지면 슬레이브 모듈과 마찬가지로 깜빡임을 멈추고 LED가 계속 켜져 있다. 스케치 2는 (COM36에 연결된) 슬레이브에 업로드 할 스케치로, 문자열을 전송 받아 다시 마스터로 돌려보내는 역할을 한다.

스케치 2. 블루투스 통신 - 슬레이브

```

#include <SoftwareSerial.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결
SoftwareSerial BTSerial(2, 3);
char buffer[100];           // 수신 문자열 버퍼

void setup()
{
    BTSerial.begin(9600);    // 블루투스 모듈 통신 속도의 디폴트값
}

void loop()
{
    if(BTSerial.available()){ // 마스터로부터의 데이터 수신
        int no = BTSerial.readBytes(buffer, 99);
        BTSerial.write(buffer, no); // 마스터로 재전송
    }
}

```

스케치 3은 마스터에 업로드 할 스케치로, 시리얼 모니터로 입력받은 문자열을 시리얼 모니터로 출력함과 동시에 블루투스 연결을 통해 슬레이브로 전송한다. 또한 슬레이브에서 되돌려준 메시지를 시리얼 모니터로 출력하는 역할도 함께 하고 있다.

스케치 3. 블루투스 통신 - 마스터

```

#include <SoftwareSerial.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결
SoftwareSerial BTSerial(2, 3);
char buffer[100];           // 수신 문자열 버퍼

void setup()
{
    Serial.begin(9600);      // 시리얼 통신 초기화
}

```

```
BTSerial.begin(9600);          // 블루투스 모듈 초기화
}

void loop()
{
    if(BTSerial.available()){    // 슬레이브로부터 데이터 수신
        int bytes = BTSerial.readBytes(buffer, 99);
        buffer[bytes] = 0;
        Serial.print("Slave Echo : ");
        Serial.println(buffer);
    }

    if(Serial.available()){      // 시리얼 모니터의 데이터 입력
        int bytes = Serial.readBytes(buffer, 99);
        buffer[bytes] = 0;
        Serial.println(buffer);
        BTSerial.write(buffer);
    }
}
```

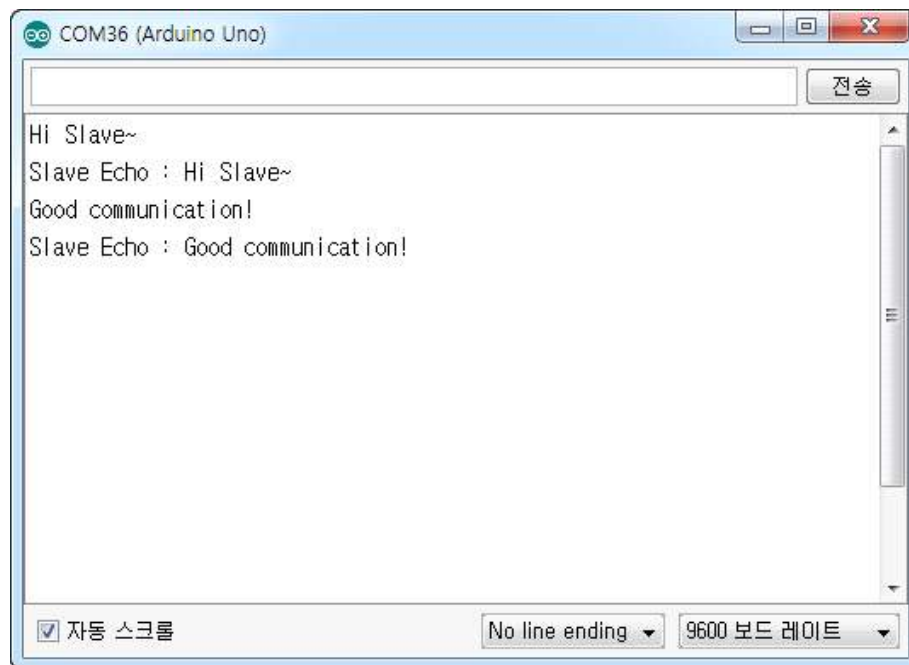


그림 4. 스케치 2와 스케치 3 실행 결과

2. 블루투스를 통한 원격 제어

슬레이브에 LED 2개를 디지털 3번과 4번에 연결하자. 마스터에는 디지털 3번과 4번에 버튼을 연결하고 3번 버튼을 누른 경우 버튼이 'A' 또는 'B'를 슬레이브로 전송하여 각각의 LED 상태를 반전시키도록 해보자. 마스터와 슬레이브에는 그림 5 및 그림 6과 같이 버튼과 LED를 연결한다.

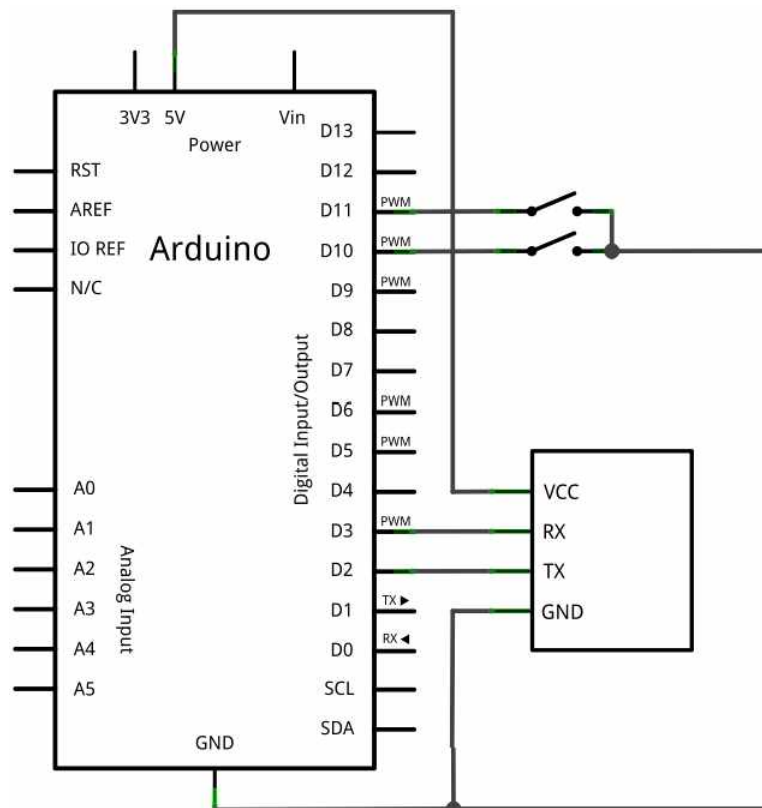


그림 5. 버튼과 블루투스 마스터 모듈 연결 회로도

그림 5에서 버튼은 내부 풀업 저항을 사용할 것이므로 별도의 저항은 연결하지 않았다.

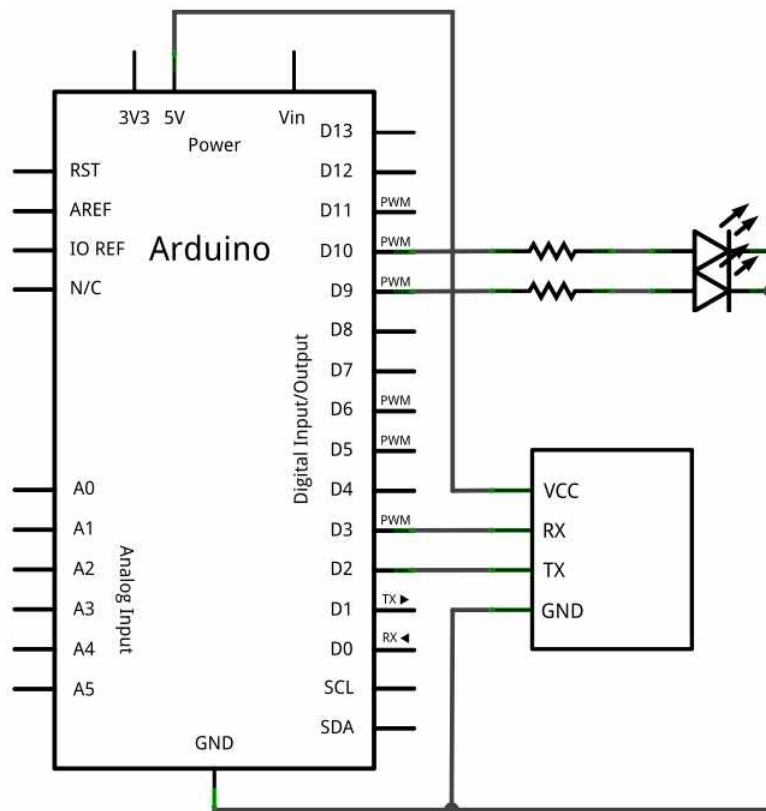


그림 6. LED와 블루투스 슬레이브 모듈 연결 회로도

스케치 4는 마스터에 업로드 할 스케치로, 버튼의 상태를 검사하여 해당 문자를 슬레이브로 전달하는 스케치의 예이다.

스케치 4. 블루투스를 통한 원격 제어 - 마스터

```
#include <SoftwareSerial.h>

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

int switch1 = 9;           // 스위치 연결 핀
int switch2 = 10;

void setup()
{
```

```

BTSerial.begin(9600);                // 블루투스 연결 초기화

pinMode(switch1, INPUT_PULLUP);      // 내부 풀업 저항을 사용하는 버튼 설정
pinMode(switch2, INPUT_PULLUP);
}

void loop()
{
  if(digitalRead(switch1) == 0){      // 1번 버튼이 눌리지면 'A' 전송
    BTSerial.write('A');
    delay(200);
  }

  if(digitalRead(switch2) == 0){      // 2번 버튼이 눌리지면 'B' 전송
    BTSerial.write('B');
    delay(200);
  }
}

```

스케치 5는 슬레이브에 업로드 할 스케치로, 마스터로부터 'A'나 'B'의 문자가 수신된 경우 해당 LED를 반전시키는 스케치의 예이다.

스케치 5. 블루투스를 통한 원격 제어 - 슬레이브

```

#include <SoftwareSerial.h>

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

int LED1 = 9;                        // LED 연결 핀
int LED2 = 10;

boolean state1 = false;              // 현재 LED의 상태
boolean state2 = false;

```

```
void setup()
{
    BTSerial.begin(9600);                // 블루투스 연결 초기화

    pinMode(LED1, OUTPUT);               // LED 연결 핀을 출력으로 설정
    pinMode(LED2, OUTPUT);
    digitalWrite(LED1, LOW);             // LED는 꺼진 상태로 시작
    digitalWrite(LED1, LOW);
}

void loop()
{
    if (BTSerial.available()){           // 마스터로부터 데이터 수신
        char ch = BTSerial.read();

        if(ch == 'A'){                  // 'A'를 수신한 경우 첫 번째 LED 반전
            state1 = !state1;
            digitalWrite(LED1, state1);
        }
        else if(ch == 'B'){             // 'B'를 수신한 경우 두 번째 LED 반전
            state2 = !state2;
            digitalWrite(LED2, state2);
        }
    }
}
```

마스터에 연결된 버튼을 눌러 슬레이브에 연결된 LED가 반전되는 지 확인해 보자.