

Chapter 224. 아두이노-아두이노 지그비 연결

지그비는 개인용 근거리 무선 통신 방법 중 하나로 가격이 저렴하고 블루투스에 비해 메시(mesh) 네트워크 구성이 쉬운 등 여러 가지 장점을 가지고 있다. 지그비의 단점 중 하나인 짧은 통신 거리는 저전력 무선 통신에서의 공통적인 특징이지만 메시 네트워크 구성과 노드의 중계(routing) 기능을 활용함으로써 통신 가능 거리는 확장이 가능하다. 또 다른 단점으로 꼽히는 전송 속도의 한계 역시 저전력이라는 목표를 달성하기 위한 방편이므로 지그비는 적은 데이터를 전송하는 응용에 특화되어 있다고 보는 것이 옳다. 최근 블루투스가 4.0으로 업데이트 되면서 새롭게 선보인 BLE(Bluetooth Low Energy)는 지그비와 많은 부분 겹치고 있다. BLE는 지그비보다 더 적은 전력을 소모하는 것으로 알려져 있지만, BLE는 블루투스의 스타 네트워크에 기반을 두고 있으므로 메시 네트워크에 기반을 둔 지그비에 비해 연결할 수 있는 노드 수가 제한되고 네트워크 구성에 제약이 따른다. 이러한 특징으로 인해 BLE는 기존 블루투스를 이용하는 스마트폰 기반의 네트워크에 주로 사용되는 반면, 지그비는 센서 네트워크나 공장 자동화 등에 주로 사용되는 등 각각의 영역을 찾아나가고 있는 것으로 보인다.

1. 지그비 모듈을 이용한 2개 아두이노 연결

지그비는 IEEE 802.15.4를 바탕으로 하고 있다. IEEE 802.15.4의 경우 스타 또는 P2P(Point to Point) 네트워크만을 지원하지만, 지그비는 메시 네트워크 역시 지원한다. 이 장에서는 DiGi International에서 제작하여 판매하는 모듈 중 IEEE 802.15.4 프로토콜을 사용하는 3개의 모듈을 스타 네트워크를 구성하는 방법에 대해 살펴본다. 네트워크 구성에 사용된 각 지그비 모듈 또는 지그비 모듈이 연결된 아두이노를 이 장에서는 ‘노드(node)’라고 지칭한다. 더 많은 수의 모듈들을 사용하는 네트워크를 구성하고자 한다면 IEEE 802.15.4 프로토콜이 아닌, 지그비 프로토콜이나 지그비 프로토콜과 호환되는 DiGi International 사의 자체 프로토콜인 DigiMesh 프로토콜을 사용하는 것을 고려해볼 수 있다. 그림 1은 이 장에서 사용한 지그비 모듈로 Xbee는 DiGi International의 등록상표이므로 지그비와 혼돈하지 않도록 한다.

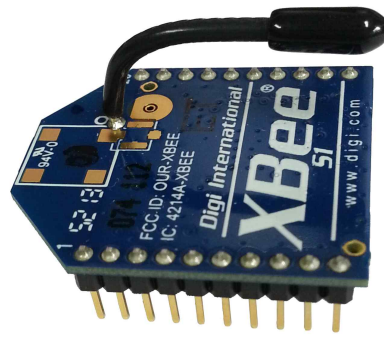


그림 1. IEEE 802.15.4 프로토콜 지원 지그비 모듈

먼저 세 개의 지그비 모듈을 준비하고 각각의 모듈의 주소를 설정한다. 주소 설정은 지그비 모듈을 USB 익스플로러를 통해 컴퓨터와 연결하고 지그비 설정을 위한 전용 프로그램인 XCTU 프로그램을 사용하는 것이 편리하다. 각 노드의 주소는 표 1과 같이 설정한다. 자세한 주소 설정 방법 및 그 의미는 **지그비 장**을 참고하도록 한다.

파라미터	노드 1	노드 2	노드 3	비고
CH	C			통신 채널
ID	0x1234			네트워크 ID
DH	0	0	0	
DL	0xFFFF	1	1	
MY	1	2	3	노드 주소
비고	브로드캐스트	노드 1로 데이터 전송	노드 1로 데이터 전송	

표 1. 지그비 모듈 주소 설정

먼저 두 개의 아두이노에 지그비 모듈을 연결하고 두 개의 노드 사이에 통신을 테스트해 보자. 지그비 모듈은 20개의 핀을 가지며 아두이노 우노와는 전원(핀 1과 핀 10)과 UART 시리얼 통신을 위한 데이터 핀(핀 2과 핀 3)만을 연결하면 된다. 단, 지그비 모듈은 3.3V를 사용한다는 점은 명심하여야 한다.

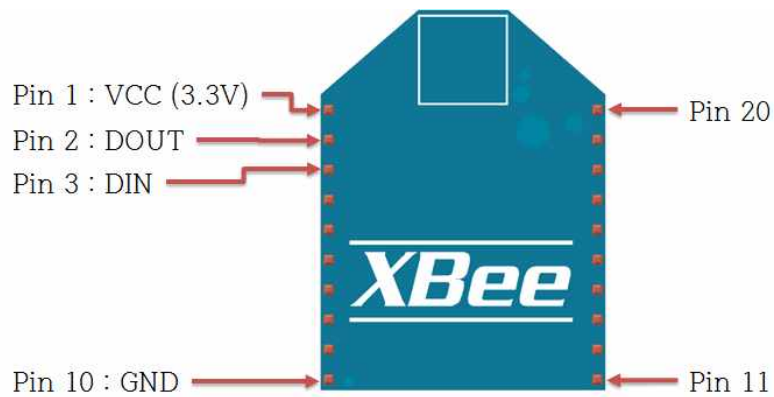


그림 2. 지그비 모듈 핀 번호

지그비 모듈은 그림 3과 같이 아두이노 우노와 연결한다. 노드 1은 다른 노드에서 전송된 데이터를 수신하여 처리하는 역할을 하며 COM35에 연결된 것으로 가정하고, 노드 2는 COM36에 연결된 것으로 가정한다.

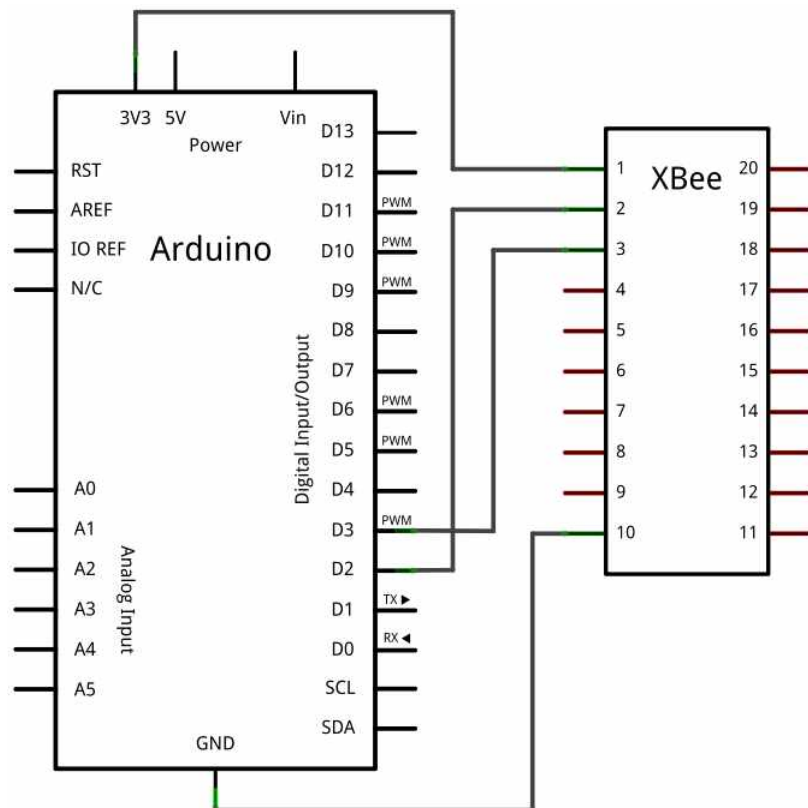


그림 3. 지그비 모듈 연결 회로도

스케치 1은 노드 1에 업로드 할 스케치로 2초 간격으로 노드 2에 현재 카운터 값을 요청하는 스케치의 예이다. 수신된 값은 시리얼 모니터로 출력한다.

스케치 1. 노드 1 - 1:1 통신

```
#include <SoftwareSerial.h>

SoftwareSerial xbee(2, 3);          // (RX, TX)
unsigned long time_previous, time_current;

void setup() {
  Serial.begin(9600);
  xbee.begin(9600);                  // Xbee 모듈 연결 초기화
  time_previous = millis();
}

void loop() {
  time_current = millis();
  // 2초 간격으로 노드 2로 현재 카운터 값 요청
  if(time_current - time_previous > 2000){
    time_previous = time_current;
    xbee.write('C');                 // 카운터 값 요청 문자 'C'
  }

  if(xbee.available()){
    Serial.println(xbee.read());     // 수신 데이터 출력
  }
}
```

스케치 2는 노드 2에 업로드 할 스케치로, 1초 간격을 증가하는 카운터를 가지며, 현재 카운터 값을 요청하는 문자 'C'가 수신된 경우 카운터 값을 노드 1로 반환하는 스케치의 예이다.

스케치 2. 노드 2 - 1:1 통신

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

```
SoftwareSerial xbee(2, 3);           // (RX, TX)
unsigned long time_previous, time_current;
byte count = 0;                      // 카운터

void setup() {
    xbee.begin(9600);                 // Xbee 모듈 연결 초기화
    time_previous = millis();
}

void loop() {
    time_current = millis();
    if(time_current - time_previous > 1000){    // 1초 간격으로 카운터 값 증가
        time_previous = time_current;
        count++;
    }

    if(xbee.available()){
        byte data = xbee.read();
        if(data == 'C')                // 카운터 요청 문자를 수신한 경우
            xbee.write(count);         // 현재 카운터 값 전송
    }
}
```

그림 4는 스케치 1과 스케치 2의 실행 결과로 2초 간격으로 2씩 증가하는 카운터 값이 시리얼 모니터로 출력됨을 확인할 수 있다.

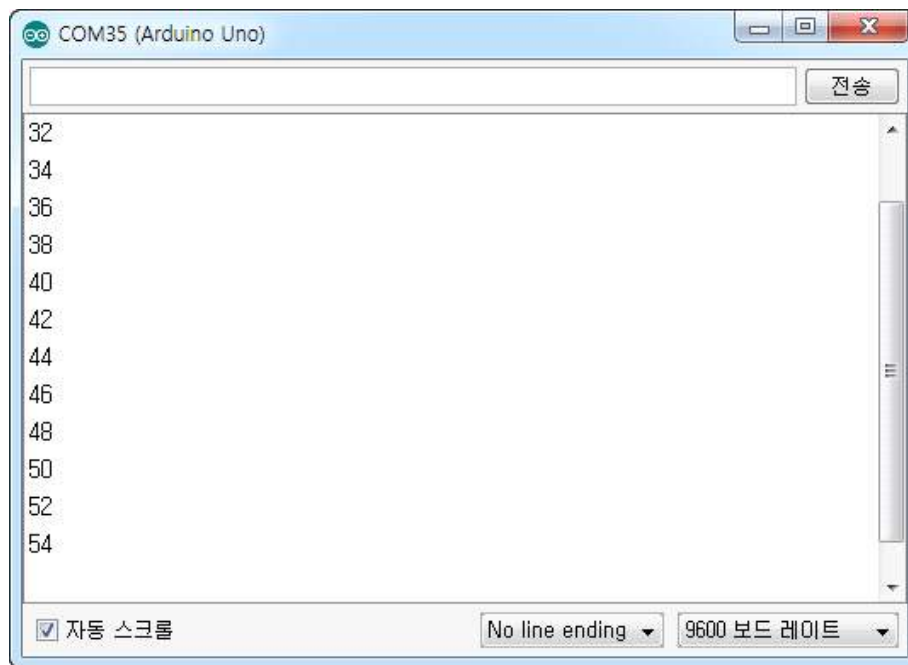


그림 4. 스케치 1과 스케치 2 실행 결과

2. 지그비 모듈을 이용한 3개 아두이노 연결

앞의 예는 2개의 모듈만을 사용하였으므로 데이터를 전송한 노드를 구별할 필요가 없었다. 하지만 2개 이상의 노드에서 데이터를 전송한다면 데이터를 전송한 노드를 구별하여야 한다. 이를 위해 데이터 전송은 항상 3바이트로 이루어진다고 가정해 보자. 첫 번째 바이트는 데이터를 전송한 노드의 ID를 나타내고, 두 번째 바이트는 데이터 값을, 그리고 마지막 바이트는 종료 문자로 '\$'를 전송한다고 가정하면 각 노드가 전송하는 데이터는 표 2와 같은 형식을 가지게 될 것이다. 각 노드가 전송하는 데이터는 알파벳 대문자와 소문자로 가정한다.

바이트 1 노드 번호	바이트 2 데이터	바이트 3 종료 문자
1	A~Z	\$
2	a~z	\$

표 2. 노드의 데이터 전송 형식

노드 1에는 스케치 3을 업로드 한다. 스케치 3은 데이터를 수신하여 버퍼에 저장하고 종료 문자 '\$'가 발견된 경우 지금까지 수신된 데이터를 출력하는 스케치의 예이다.

스케치 3. 노드 1 - 1:2 통신

```
#include <SoftwareSerial.h>

SoftwareSerial xbee(2, 3);           // (RX, TX)
String receiveBuffer = "";
boolean process_it = false;

void setup() {
  Serial.begin(9600);
  xbee.begin(9600);                  // Xbee 모듈 연결 초기화
}

void loop() {
  if(xbee.available()){
    byte data = xbee.read();

    if(data == '$')                  // 종료 문자 수신
      process_it = true;
    else                             // 버퍼에 저장
      receiveBuffer = receiveBuffer + (char)data;
  }

  if(process_it){                   // 수신 데이터 처리
    process_it = false;
    if(receiveBuffer.length() == 2){ // 메시지 길이는 종료 문자 제외 2바이트
      Serial.print((int)(receiveBuffer.charAt(0))); // 송신 ID
      Serial.print(" : ");
      Serial.println(receiveBuffer.charAt(1));      // 송신 데이터
    }
    receiveBuffer = "";              // 버퍼 비움
  }
}
```

노드 2에는 스케치 4를 업로드 한다. 스케치 4는 1초 간격으로 알파벳 소문자를 전송하는 스케치의 예이다.

스케치 4. 노드 2 & 3 - 1:2 통신

```
#include <SoftwareSerial.h>

#define DATA_START 'a'           // 전송 데이터 범위 시작
#define DATA_END   'z'           // 전송 데이터 범위 끝
#define NODE_ID     1             // 노드 ID

SoftwareSerial xbee(2, 3);         // (RX, TX)
unsigned long time_previous, time_current;
char sendData = DATA_START;

void setup() {
  xbee.begin(9600);                // Xbee 모듈 연결 초기화
  time_previous = millis();
}

void loop() {
  time_current = millis();
  if(time_current - time_previous > 1000){
    time_previous = time_current;
    xbee.write(NODE_ID);           // 노드 ID
    xbee.write(sendData);          // 전송 데이터
    xbee.write('$');               // 종료 문자

    sendData++;
    if(sendData > DATA_END)
      sendData = DATA_START;
  }
}
```

노드 3에는 스케치 4의 #define 문장을 수정하여 노드 ID와 전송되는 문자를 대문자로 수정한 후 업로드 한다.

```
#define DATA_START 'A'  
#define DATA_END   'Z'  
#define NODE_ID     2
```

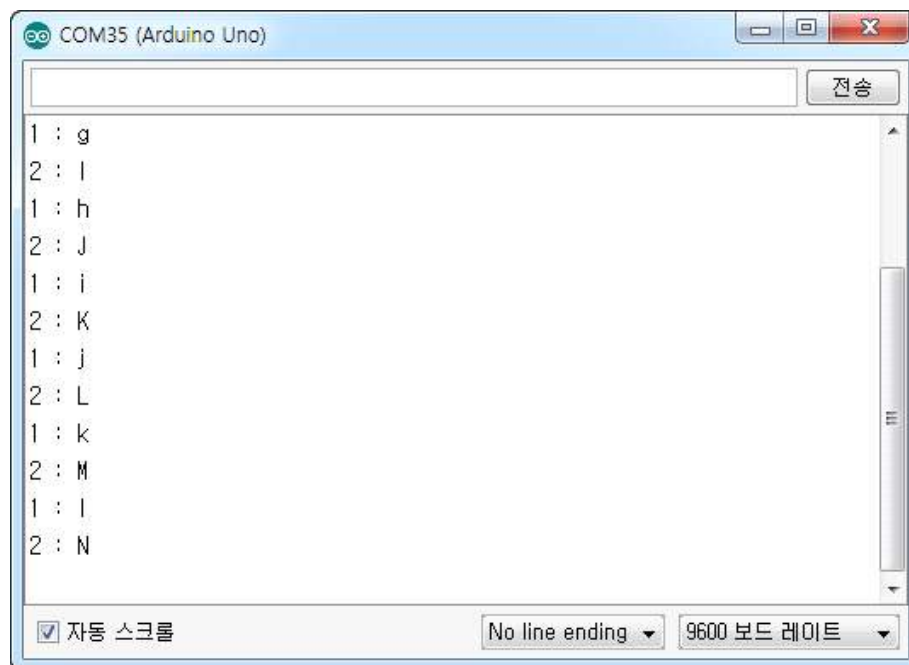


그림 5. 스케치 3과 스케치 4 실행 결과