
Chapter 222. 아두이노-아두이노 I2C 연결

I2C(Inter-Integrated Circuit)는 저속의 주변장치 연결을 위한 동기식, 반이중 방식의 시리얼 통신 방식으로 1:N 연결을 지원하며 슬레이브 선택을 위해 소프트웨어적인 주소를 사용하므로 SPI에서와 같이 연결하는 장치의 수에 따라 사용하는 핀이 증가하지 않으므로 적은 양의 데이터를 간헐적으로 전송하는 센서 네트워크 등에서 흔히 사용되는 시리얼 통신 방법이다. I2C가 2-Wire라고도 불리는 것은 데이터 송수신을 위한 SDA(Serial Data)와 동기화 클록을 위한 SCL(Serial Clock)의 2개 연결만으로 여러 개의 슬레이브를 제어할 수 있기 때문이다. I2C 통신을 위해 필요로 하는 연결선의 개수가 적은 것은 UART나 SPI와 비교할 때 I2C의 장점이지만 송수신을 위해 하나의 데이터 선을 사용하는 반이중 방식의 통신은 속도에 한계가 있으므로 I2C의 용도는 제한된다고 볼 수 있다. 아두이노에서는 I2C 통신을 위해 Wire 라이브러리를 제공하고 있으므로 I2C 통신을 지원하는 주변장치와 손쉽게 통신을 수행할 수 있다.

1. 슬레이브로 데이터 전송

두 개의 아두이노가 COM35와 COM36에 연결되어 있고, COM35에 연결된 아두이노를 마스터, COM36에 연결된 아두이노를 슬레이브라고 가정하자. 두 개의 아두이노를 그림 1과 같이 연결하자. 아두이노 우노에 사용된 ATmega328은 I2C 연결을 위해 아날로그 4번을 SDA, 아날로그 5번을 SCL로 지정하고 있다.

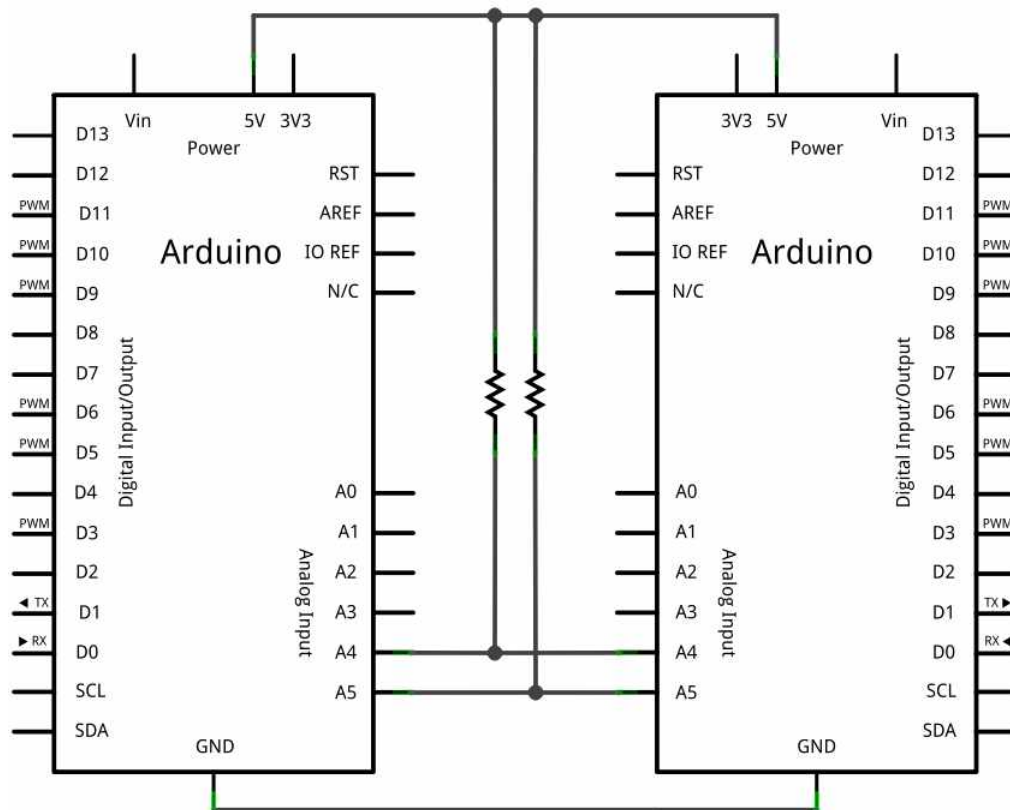


그림 1. 두 개의 아두이노 우노 사이의 I2C 연결 회로도

SPI 라이브러리는 슬레이브 모드를 지원하지 않으므로 SPI 통신을 위해서는 직접 레지스터를 조작해야 하는 불편함이 있지만, Wire 라이브러리는 마스터와 슬레이브 모드를 모두 지원하므로 SPI 통신에 비해 간단하게 I2C 통신을 수행할 수 있다. 스케치 1은 마스터가 1초 간격으로 'Hello World!'라는 문자열을 슬레이브로 전송하는 스케치의 예이다.

스케치 1. I2C 통신 테스트 - 마스터

```
#include <Wire.h>
#define SLAVE 4           // 슬레이브 주소

void setup() {
    Wire.begin();         // 마스터 모드로 Wire 라이브러리 초기화
}

void loop() {
```

```
const char *p = "Hello, World\n";

// I2C 통신을 통한 전송 시작
Wire.beginTransmission(SLAVE);

for(int i = 0; i < strlen(p); i++){           // 버퍼에 데이터 기록
    Wire.write(p[i]);
}

// I2C 통신을 통한 전송 끝
Wire.endTransmission(SLAVE);

delay(1000);
}
```

스케치 1에서 볼 수 있듯이 I2C를 통한 데이터 전송은 beginTransmission 함수에서 시작되고 write 함수는 전송하고자 하는 데이터를 버퍼에 기록된다. 실제 데이터 전송이 일어나는 시점은 endTransmission 함수가 호출될 때이다.

스케치 2. I2C 통신 테스트 - 슬레이브

```
#include <Wire.h>
#define SLAVE 4

void setup() {
    Wire.begin(SLAVE);    // 슬레이브 모드로 Wire 라이브러리 초기화
    // 마스터로부터 데이터가 전송된 경우 처리할 함수 등록
    Wire.onReceive(receiveFromMaster);

    Serial.begin(9600);
}

void loop () {
}
```

```
void receiveFromMaster(int bytes) {  
  for(int i = 0; i < bytes; i++){  
    char ch = Wire.read();           // 수신 버퍼 읽기  
    Serial.print(ch);  
  }  
}
```

스케치 2는 슬레이브를 위한 코드로 마스터에서 전송된 데이터를 시리얼 모니터로 출력하는 예이다. 마스터 코드에서와 마찬가지로 begin 함수로 Wire 라이브러리를 초기화하지만 슬레이브 모드인 경우에는 주소를 지정해야 한다. 스케치 1에서는 마스터 모드로 초기화하기 때문에 주소를 지정하지 않았다. 마스터로부터 데이터가 수신되면 슬레이브의 처리 함수가 자동으로 호출되며, 호출되는 함수는 onReceive 함수로 지정한다. 지정된 함수는 수신된 바이트 수를 매개변수로 가지므로 수신된 데이터의 바이트 수를 알 수 있다. 이처럼 슬레이브는 인터럽트 방식으로 수신 데이터를 처리하므로 loop 함수는 비어있다. 그림 2는 슬레이브와 연결된 시리얼 모니터에서의 출력 결과를 보여준다.

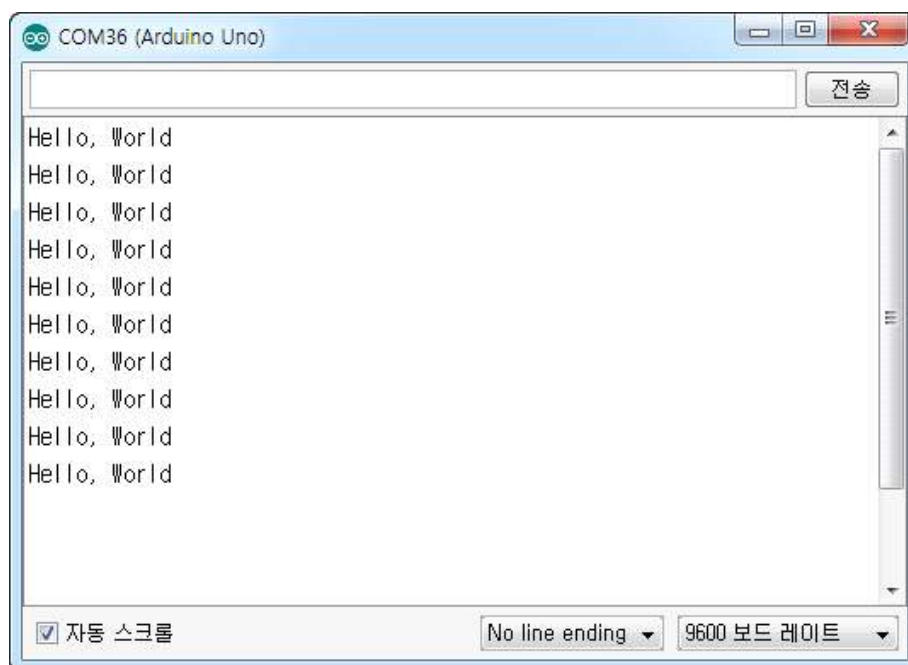


그림 2. 스케치 1과 스케치 2 실행 결과

2. 슬레이브로부터 데이터 수신

I2C는 반이중 방식을 사용하므로 슬레이브로부터 데이터를 수신하기 위해서는 먼저 슬레이브로 요청을 보내야 한다. 스케치 3은 마스터에 업로드 할 스케치로 1초에 한 번 슬레이브에 1바이트의 값을 요청하고 이를 수신하여 시리얼 모니터에 출력하는 예이다. 슬레이브로 데이터를 요청하기 위해서는 requestFrom 함수를 사용하는 점이 데이터를 전송할 때와는 다르다. 데이터 요청 후 실제 슬레이브에서 전송된 데이터를 읽기 위해서는 read 함수를 사용한다.

스케치 3. 카운터 값 요청 - 마스터

```
#include <Wire.h>

#define SLAVE 4           // 슬레이브 주소

void setup() {
    Wire.begin();          // Wire 라이브러리 초기화
    Serial.begin(9600);    // 직렬 통신 초기화
}

void loop() {
    i2c_communication();   // 슬레이브로 데이터 요구 및 수신 데이터 처리
    delay(1000);
}

void i2c_communication() {
    Wire.requestFrom(SLAVE, 1); // 1 바이트 크기의 데이터 요청

    char c = Wire.read();      // 수신 데이터 읽기
    Serial.println(String(c, DEC)); // 수신 데이터 출력
}
```

스케치 4는 슬레이브를 위한 스케치로 마스터의 요청이 있는 경우 1바이트 크기의 카운터를 증가시키고 마스터로 전송하는 예이다. 슬레이브가 마스터로부터의 데이터를 수신한 경우 처리 함수를 onReceive 함수를 통해 등록한 것과 유사하게, 마스터로부터의 데이터 요청을 처리할 함수는 onRequest 함수를 통해 등록한다. 데이터 요청을 처리할 함수는 데이터 수신을 처리하는 함수와 달리 매개변수를 가지지 않는다.

스케치 4. 초 단위 카운터 - 슬레이브

```
#include <Wire.h>
#define SLAVE 4

byte count = 0;          // 카운터

void setup() {
  // Wire 라이브러리 초기화
  // 슬레이브로 참여하기 위해서는 주소를 지정해야 한다.
  Wire.begin(SLAVE);

  // 마스터의 데이터 전송 요구가 있을 때 처리할 함수 등록
  Wire.onRequest(sendToMaster);
}

void loop () {
}

void sendToMaster() {
  Wire.write(++count);    // 카운터 값을 증가시키고 마스터로 전송
}
```

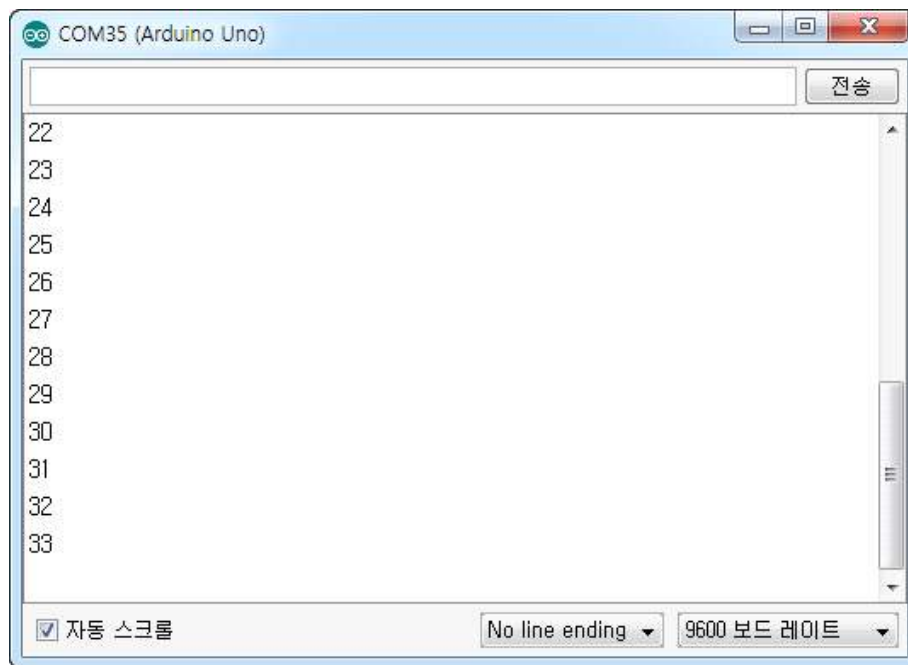


그림 3. 스케치 3과 스케치 4 실행 결과

2바이트 크기의 값, 예를 들어 정수값을 받고자 한다면 `requestFrom` 함수에서 요청하는 바이트 수가 2바이트임을 알려주어야 한다. 이후 `read` 함수를 통해 두 번 데이터를 읽으면 2바이트의 데이터를 수신할 수 있다. 하지만 2바이트의 데이터가 `int` 형의 정수값인 경우에는 주의가 필요하다. ATmega328은 리틀 엔디안(little endian) 방식으로 숫자를 저장하므로 먼저 읽어 들인 바이트는 하위 바이트에 해당한다. 스케치 5는 시리얼 모니터로 'C'가 입력된 경우 슬레이브로부터 카운터 값을 요청하여 출력하는 스케치의 예이다.

스케치 5. 2바이트 크기 데이터 전송 - 마스터

```
#include <Wire.h>

#define SLAVE 4           // 슬레이브 주소

void setup() {
    Wire.begin();          // Wire 라이브러리 초기화
    Serial.begin(9600);    // 직렬 통신 초기화
}

void loop() {
```

```

    if(Serial.available()){
        char ch = Serial.read();
        if(ch == 'C'){
            i2c_communication();    // 슬레이브로 데이터 요구 및 수신 데이터 처리
        }
    }
}

void i2c_communication() {
    Wire.requestFrom(SLAVE, 2);    // 2바이트 데이터 요청

    byte c1 = Wire.read();         // 수신 데이터 읽기
    byte c2 = Wire.read();

    Serial.println((int)( c2 << 8) + c1 );    // 수신 데이터 출력
}

```

슬레이브에서도 마스터의 요청이 있을 때 2바이트의 데이터를 버퍼에 기록해 주어야 한다. 스케치 6은 초 단위로 증가하는 정수형 카운터를 가지고 마스터의 요청이 있을 때 현재 카운터 값을 전송하는 스케치의 예이다.

스케치 6. 2바이트 크기 데이터 전송 - 슬레이브

```

#include <Wire.h>
#define SLAVE 4

int count = 0;    // 카운터

void setup() {
    // Wire 라이브러리 초기화
    // 슬레이브로 참여하기 위해서는 주소를 지정해야 한다.
    Wire.begin(SLAVE);

    // 마스터의 데이터 전송 요구가 있을 때 처리할 함수 등록

```

```
Wire.onRequest(sendToMaster);  
}  
  
void loop () {  
    count++;                // 초 단위 카운터  
    delay(1000);  
}  
  
void sendToMaster() {  
    Wire.write( (uint8_t *)&count, 2 );    // 2바이트 크기의 카운터 값 전송  
}
```

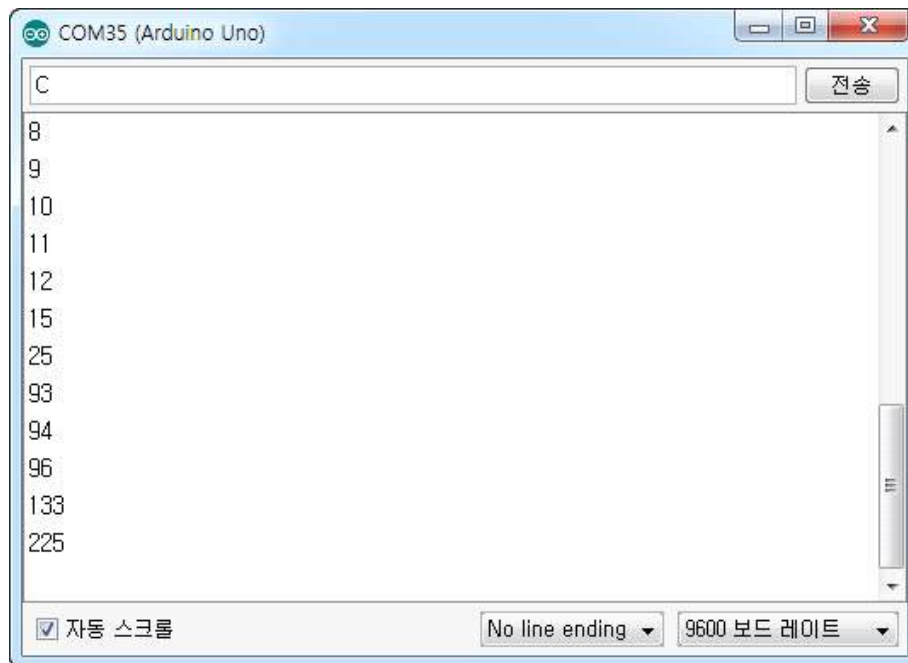


그림 4. 스케치 5와 스케치 6 실행 결과