사물인터넷

Chapter 11. I2C 통신

1. SPI와 I2C통신
2. I2C 동작 원리
3. master_reader / slave_sender 통신 실험
4. master_write / slave_receiver 통신 실험
5. 가변저항을 이용하여 원격지 LED 깜박임 조절
6. 버튼을 눌렀을 때 원격지 센서 값 읽기

학습목표

- 이 장에서는 SPI 통신의 기본 원리를 이해한다.
- I2C 통신의 기본 원리를 이해한다.
- 1:N의 아두이노를 이용하여 데이터 송신과 수신을 실험한다.
- I2C 통신 실험을 통해서 이해한다.

1. SPI와 I2C통신에 대하여

1) 비동기식 통신 방법

- UART 통신은 1byte를 송신할 때마다 추가로 start/stop bit를 추가
- 총 10 비트의 전송 시간이 필요하며 데이터 속도에 영향을 미침
- 클럭이 조금이라도 다르게 되면 데이터 전송에 오류가 발생

1. SPI와 I2C통신에 대하여

2) 동기식 통신 방법

- 데이터를 보낼 때마다 데이터가 전달된다고 알려주는 별도의 클럭 라인을 이용
- 클럭 라인을 통신 라인에 연결된 다수의 장치(참여 장치)가 공유
- 대표적인 시스템이 SPI와 I2C 통신
- start/stop 비트가 필요 없으며, 통신 속도를 공유할 필요도 없음

1. SPI와 I2C통신에 대하여

3) SPI 통신

- Serial Peripherial Interface 약자
- 하나의 마스터와 하나 이상의 슬레이브 기기
- SPI는 마스터에서 클럭 신호를 생성
- 속도가 빠르기 때문에 주로 빠른 데이터 전송속도가 필요한 곳에 많이 사용

1. SPI와 I2C통신에 대하여

3) SPI 통신

- 통신을 위해서 4개의 선이 필요
- **MISO(Mater In Slave Out)**: 슬레이브에서 마스터로 데이터 출력하기 위한 선
- **MOSI(Master Out Slave In)**: 마스터에서 슬레이브로 데이터를 출력하기 위한 선
- **SCK**: Clock 선
- **SS(Slave Select)**: 데이터를 송수신 할 슬레이브를 선택하기 위한 선

1. SPI와 I2C통신에 대하여

4) I2C 통신

- "Inter Integrated Circuits"를 의미
- 동기식 통신 프로토콜
- 두 개의 신호선이 필요
- 하나는 Clock 신호에 사용되고 다른 하나는 데이터 송수신에 사용
- 통신은 항상 마스터와 슬레이브 사이에서 발생
- 하나 이상의 슬레이브를 마스터에 연결할 수 있음

2. I2C 동작 원리

- 두 개의 신호선 사용
- SDA: 양방향 데이터 선
- SCL: 클럭 신호

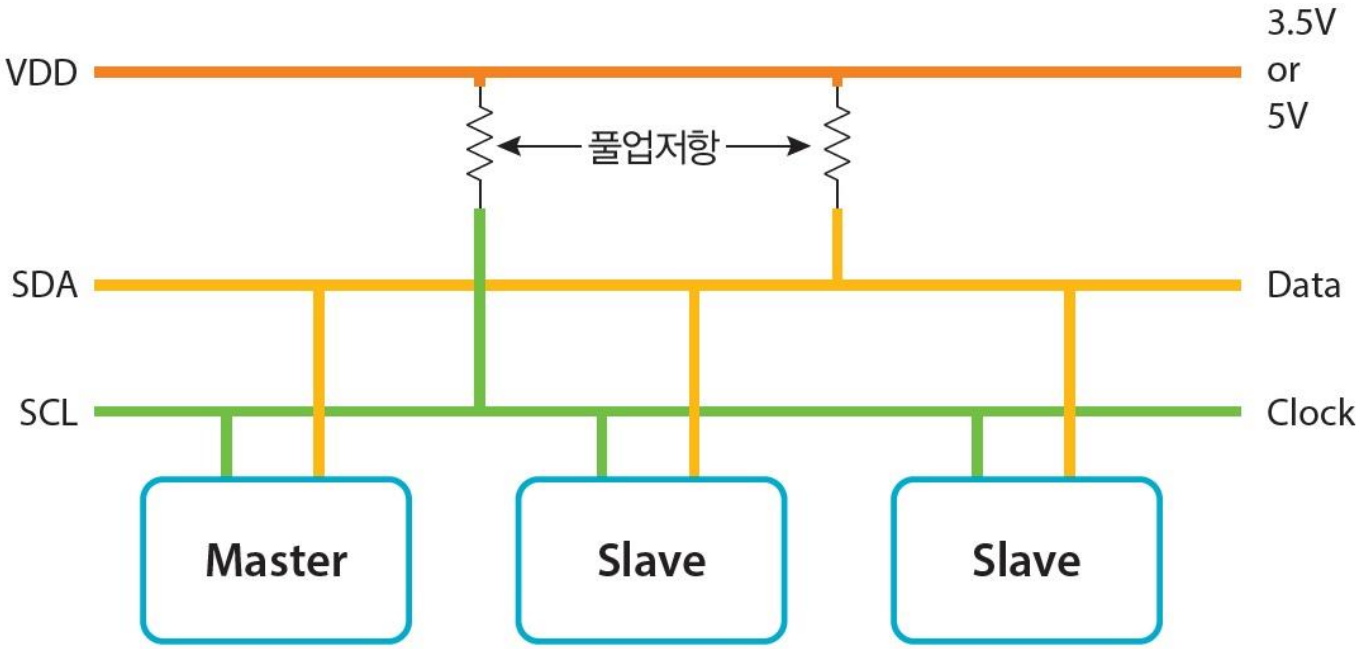


그림 11.1 I2C 신호선 연결도

2. I2C 동작 원리

- 비활성 상태일 때 버스를 공급 전압까지 끌어 올림
- 마스터 장치는 버스를 제어하고 클럭 신호를 공급
- 슬레이브에서는 개별적으로 데이터를 요청
- 마스터 장치에는 할당된 주소가 없음
- 슬레이브 장치에는 주소가 있으며 이 주소는 버스에서 고유해야 함

2. I2C 동작 원리

- I2C 라이브러리

표 11.1 I2C 라이브러리

함수명	내용
begin()	• I2C통신 초기화 및 활성화하는 함수 • Slave 모드일 경우 자신의 번호를 지정 • 번호가 지정되지 않으면 Master 모드로 설정
requestFrom()	• Slave로부터 데이터를 요청하기 위해 Master에 의해 설정
beginTransmission()	• Master에서 전송하기 위해 Slave의 주소 값을 지정 • 데이터를 버퍼에 저장 후 한꺼번에 전송 • Master 모드에서만 사용
endTransmission()	• 데이터 버퍼에 저장된 데이터를 전송 • Master 모드에서만 사용
write()	• I2C버스로 데이터를 보내기 위해 사용 • Master/Slave 모드에서 사용
available()	• 수신된 데이터의 Byte 수를 반환
read()	• I2C버스(버퍼)로부터 데이터를 읽어옴 • available() 함수로 수신 여부가 확인된 후 사용
SecClock()	• 특정 주파수를 설정하기 위해 Master에 의해 사용
onReceive()	• Master로부터 데이터가 수신되었을 때 호출될 이벤트 처리 • 수신된 데이터의 길이가 전달됨
onRequest()	• Master로부터 데이터를 요청받았을 때 호출되는 함수 • 이벤트 처리의 매개변수는 없음

3. master_reader / slave_sender 통신 실험

- Master에서 Slave(ID # 8)에 1Byte 데이터를 요구
- Slave는 요청마다 1씩 카운트가 증가된 값을 반환하는 것

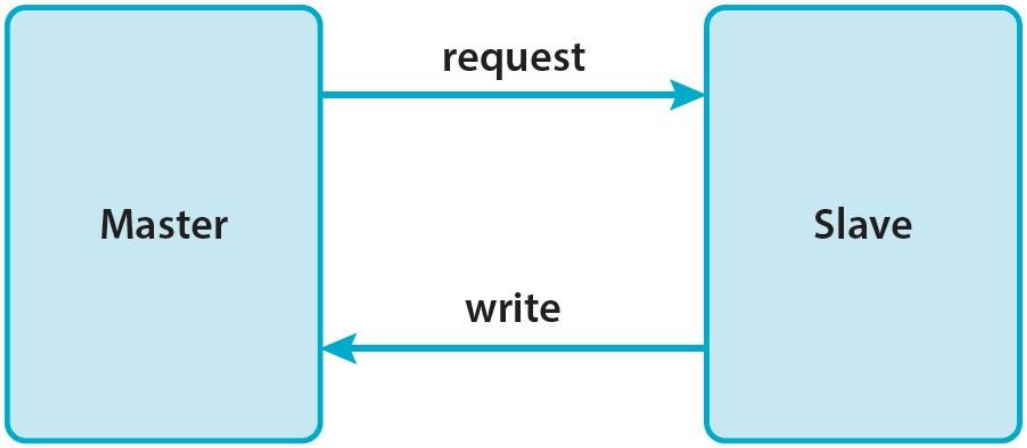


그림 11.2 master_reader / slave_sender 통신 개념도

3. master_reader / slave_sender 통신 실험

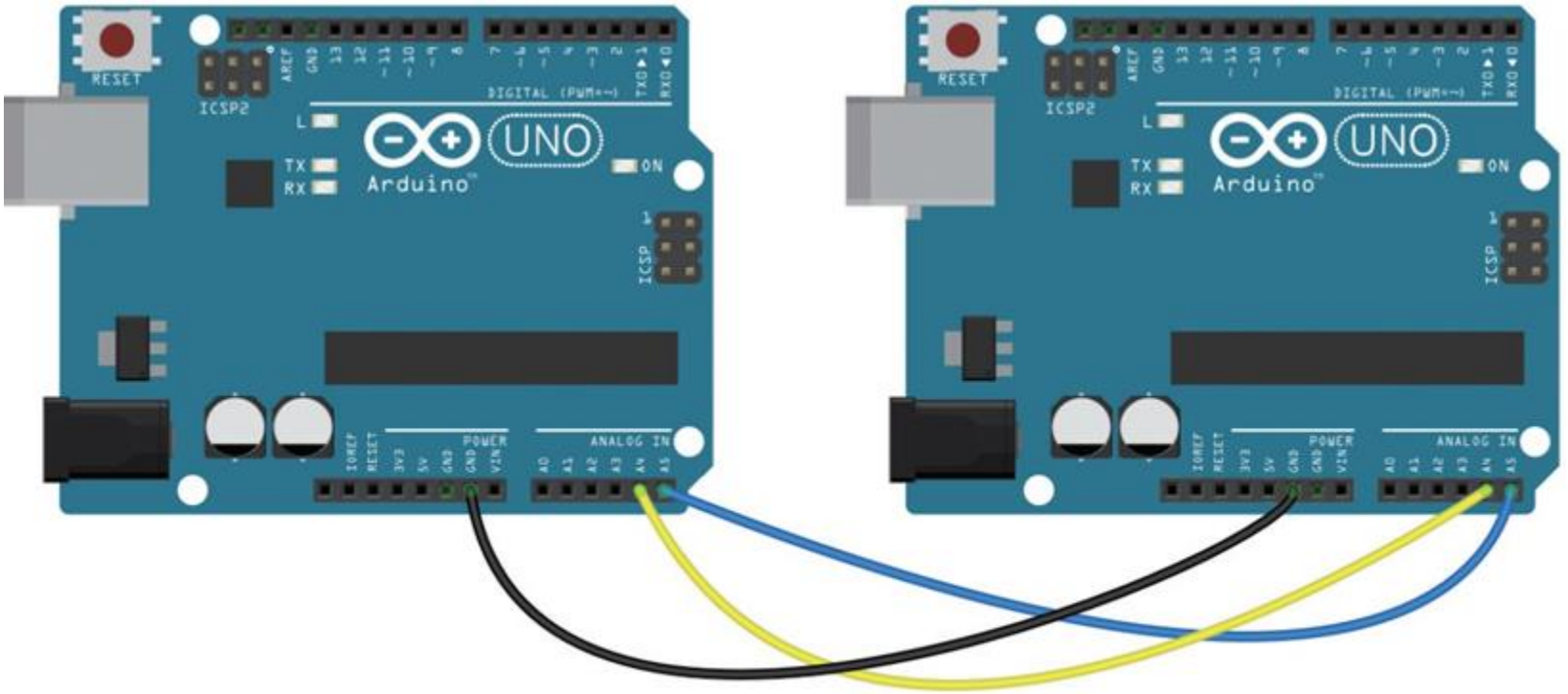


그림 11.3 master_reader / slave_sender 통신 실험 연결도

3. master_reader / slave_sender 통신 실험

- 마스터 코드

```
#include <Wire.h>
void setup() {
  Wire.begin();
  Serial.begin(9600);
}
void loop() {
  Wire.requestFrom(8, 1); // Slave ID #8로부터 1 bytes 요청
  while (Wire.available()) {
    byte b = Wire.read();
    Serial.println(b);
  }
  delay(500);
}
```

3. master_reader / slave_sender 통신 실험

- 슬레이브 코드

```
#include <Wire.h>
byte b=0;
void setup() {
   Wire.begin(8);  // Slave ID #8 지정
   Wire.onRequest(requestEvent);
   Serial.begin(9600);
}
void loop() {
}
void requestEvent() {
   Wire.write(b++);
   Serial.println(b);
}
```

3. master_reader / slave_sender 통신 실험

- 실험결과

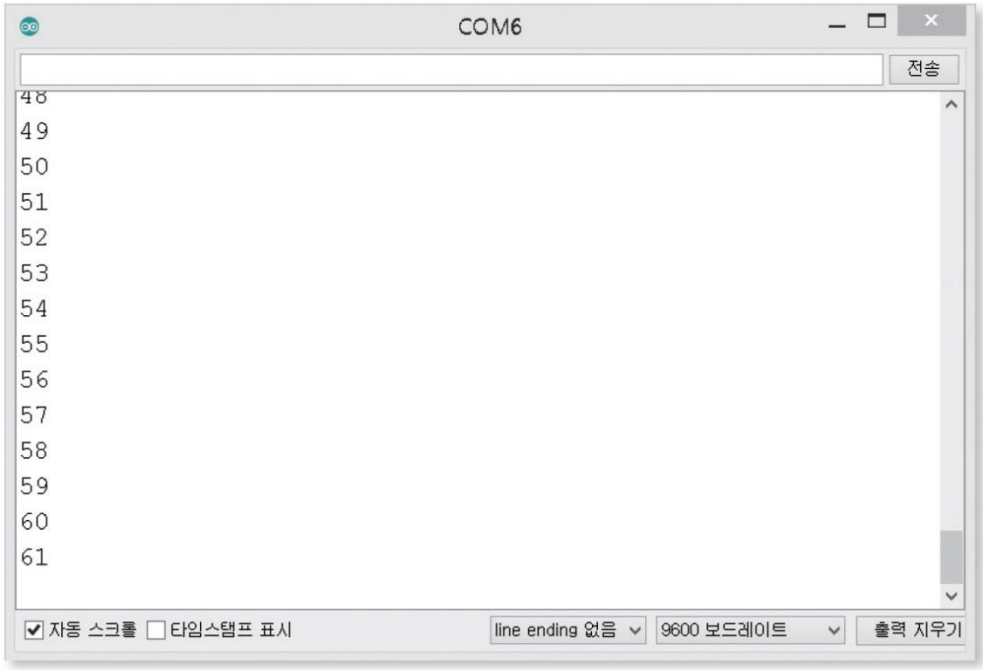


그림 11.4 master_reader / slave_sender 실험 결과

4. master_write / slave_receiver 통신 실험

- Master에서 Slave(ID # 8)에 1Byte 데이터를 보냄
- Slave는 수신된 데이터를 시리얼 모니터에 출력

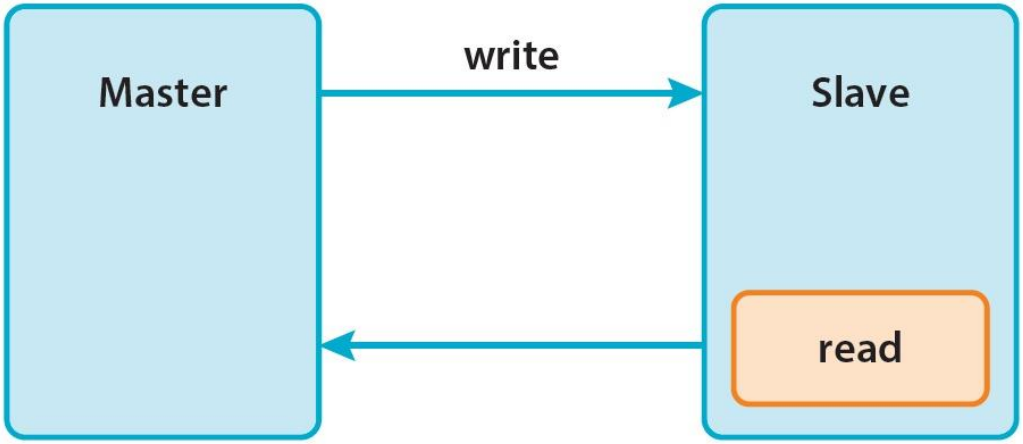


그림 11.5 master_write / slave_receiver 통신 개념도

4. master_write / slave_receiver 통신 실험

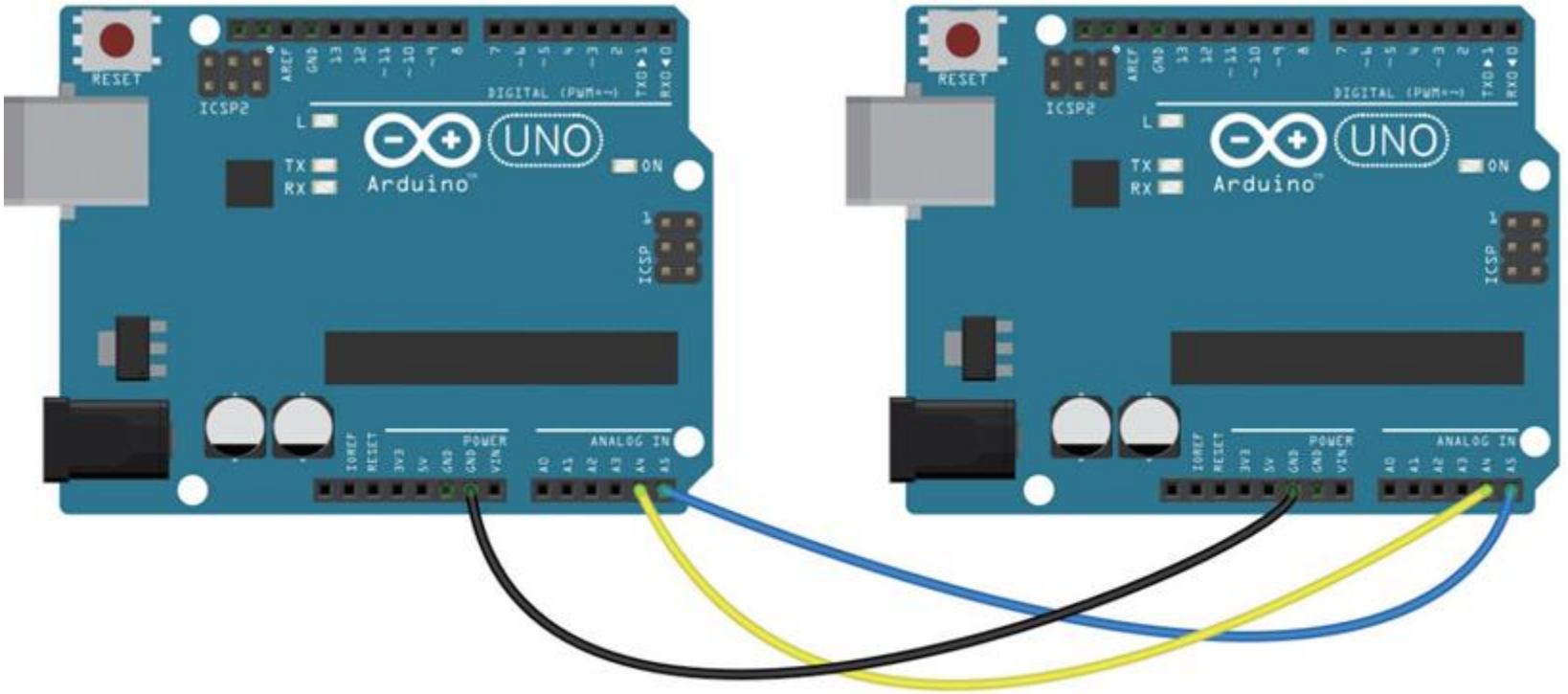


그림 11.3 master_reader / slave_sender 통신 실험 연결도

4. master_write / slave_receiver 통신 실험

- 마스터 코드

```
#include <Wire.h>
void setup() {
   Wire.begin();
   Serial.begin(9600);
}
byte x=0;
void loop() {
   Wire.beginTransmission(8); // Slave의 주소 값을 지정
   Wire.write("x is ");              // sends five bytes
   Wire.write(x);                    // sends one byte
   Wire.endTransmission();    // 데이터 버퍼에 저장된 데이터를 전송
   x++;
   delay(500);
   Serial.println(x);
}
```

4. master_write / slave_receiver 통신 실험

- 슬레이브 코드

```
#include <Wire.h>
void setup() {
  Wire.begin(8);                 //  I2C 버스가 ID #8로 연결
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);            // 시리얼 시작
}
void loop() {
  delay(100);
}
void receiveEvent(int howMany){
  while(1 < Wire.available())  {
    char c = Wire.read();      // receive byte as a character
    Serial.print(c);           // print the character
  }
  int x = Wire.read();         // receive byte as an integer
  Serial.println(x);           // print the integer
}
```

4. master_write / slave_receiver 통신 실험

- 실험 결과

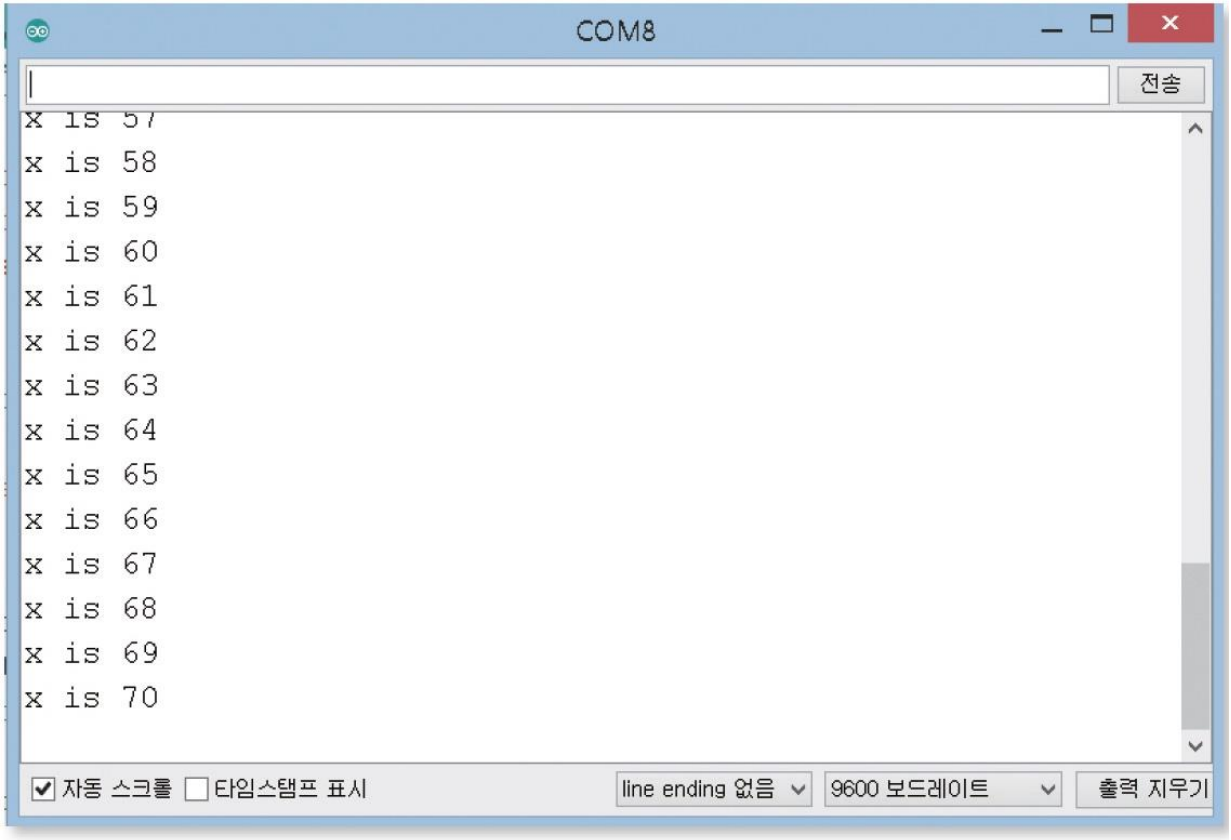


그림 11.7 master_write / slave_receiver 실험 결과

5. 가변저항을 이용하여 원격지 LED 깜박임 조절

- 저항기를 이용하여 원격지 LED의 ON/OFF 주기를 I2C 통신을 이용하여 실시

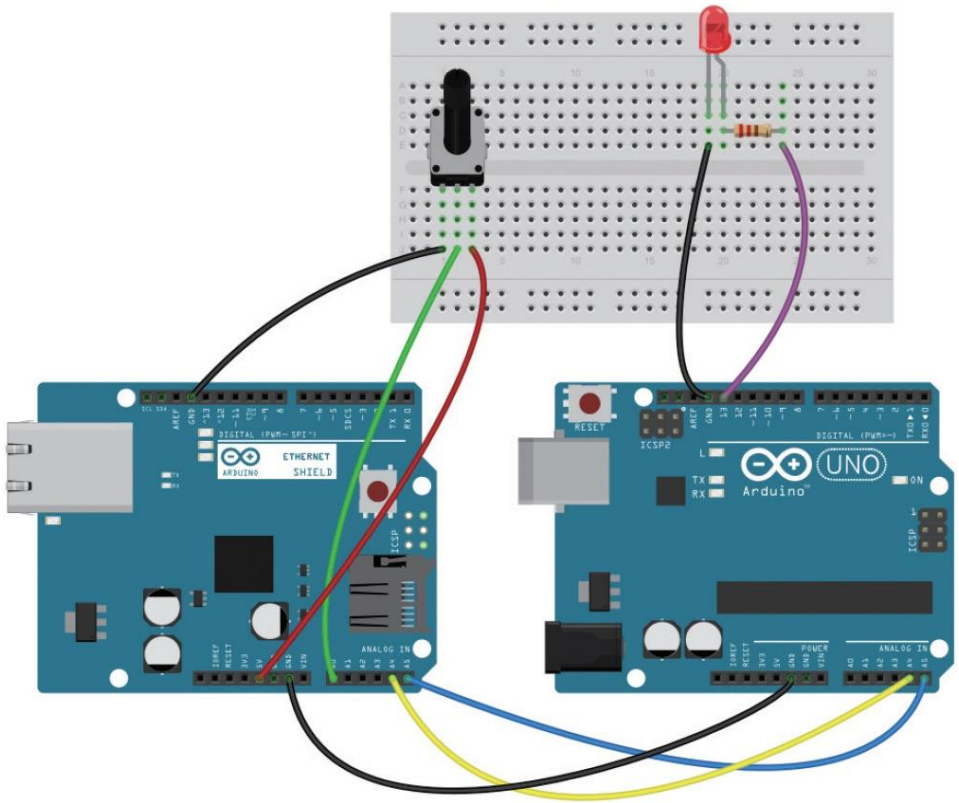


그림 11.8 가변저항기 이용 원격지 LED 깜박임 조절 실험 연결도

5. 가변저항을 이용하여 원격지 LED 깜박임 조절

- 마스터 코드

```
#include <Wire.h>
#define SLAVE_ADDR 9 // 슬레이브 ID 지정
int analogPin = 0;
int val = 0;
void setup() {
   Wire.begin(); // 마스터로써 I2C 통신 초기화
}
 void loop() {
    delay(50);
    // 읽은 값을 1-255로 정규화 시킴
     val = map(analogRead(analogPin), 0, 1023, 255, 1);
    // 슬레이브로 값 보내기
    Wire.beginTransmission(SLAVE_ADDR);
    Wire.write(val);
    Wire.endTransmission();
}
```

5. 가변저항을 이용하여 원격지 LED 깜박임 조절

- 슬레이브 코드

```
#include <Wire.h>
#define SLAVE_ADDR 9          // 슬레이브 ID 지정
int LED = 13;                 // Define LED Pin
int rd;                       // Variable for received data
int br;                       // Variable for blink rate
void setup() {
   pinMode(LED, OUTPUT);
   // Initialize I2C communications as Slave

   Wire.begin(SLAVE_ADDR);
   // Function to run when data received from master
   Wire.onReceive(receiveEvent);

   // Setup Serial Monitor
   Serial.begin(9600);
   Serial.println("I2C Slave Demonstration");
}
```

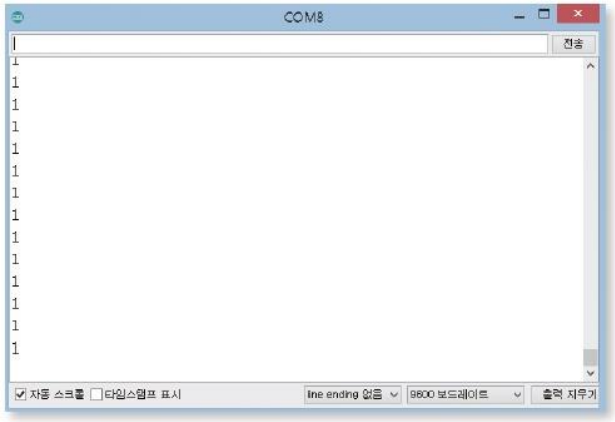
5. 가변저항을 이용하여 원격지 LED 깜박임 조절

- 슬레이브 코드

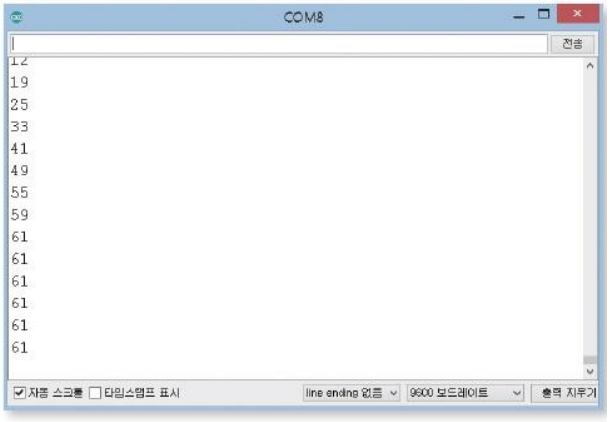
```
void receiveEvent() {
   rd = Wire.read();    // read one character from the I2C
   Serial.println(rd);    // Print value of incoming data
}
void loop() {
   delay(50);
   br = map(rd, 1, 255, 100, 2000); // 수신된 값을 이용하여 delay
                                    시간 계산
   digitalWrite(LED, HIGH);
   delay(br);
   digitalWrite(LED, LOW);
   delay(br);
}
```

5. 가변저항을 이용하여 원격지 LED 깜박임 조절

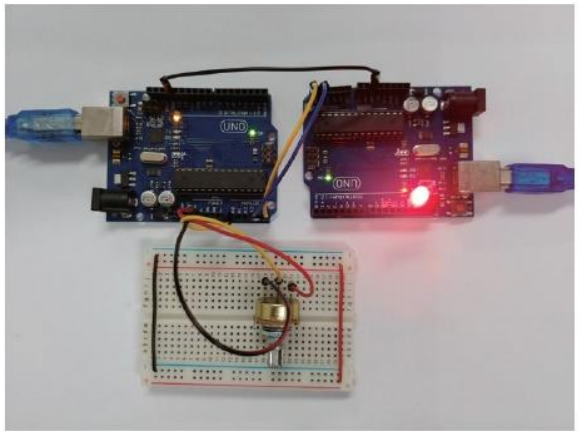
- 실험 결과



(a)



(b)



(c)

그림 11.9 슬레이브 시리얼 모니터 결과

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 마스터에서 슬레이브의 센서 값을 읽어오는 실험
- 마스터에 있는 버튼을 누르면 마스터는 지정된 슬레이브에 자료를 요청
- 슬레이브는 마스터로부터 요청이 있는지를 검사하여 요청이 있을 경우, 마스터로 해당된 데이터를 마스터로 보냄

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

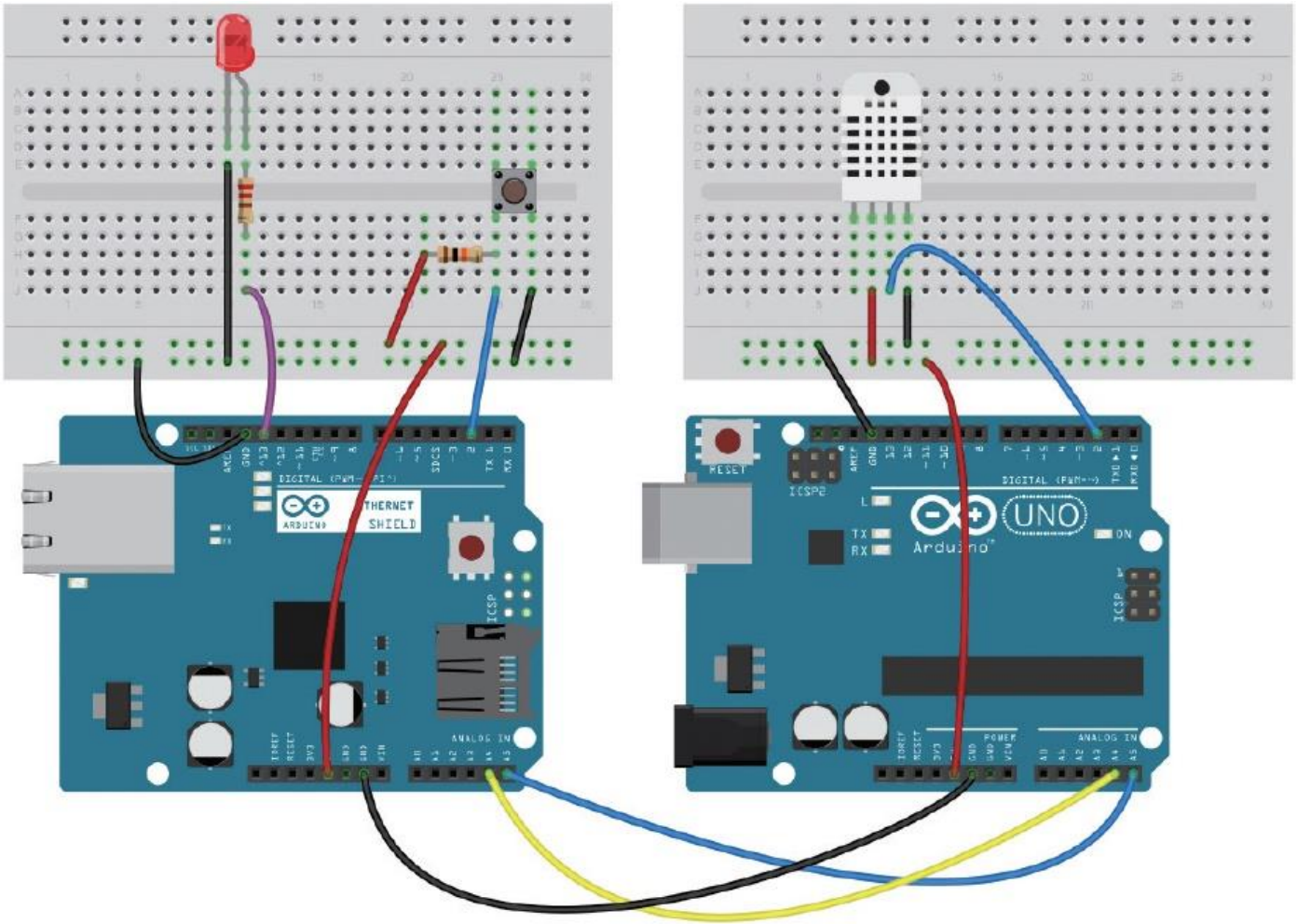


그림 11.10 원격지 센서 값 읽기 회로도

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 마스터 코드

```
#include <Wire.h>
const int button1Pin = 2; // push button을 디지털 2번 핀에 연결
const int ledPin =  13;      // led를 디지털 13번 핀에 연결
int button1State = 0;        // 버튼에서 읽어 올 디지털 값을 저장

void setup() {
   Wire.begin();
   Serial.begin(9600);
   pinMode(button1Pin, INPUT);  // 2번 핀을 입력 핀으로 설정
   pinMode(ledPin, OUTPUT);     // 13번 핀을 출력 핀으로 설정
}
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 마스터 코드

```
void loop() {
  // 버튼의 상태를 읽고 변수에 저장
  button1State = digitalRead(button1Pin);
  // 버튼이 눌렸다면
  if(button1State == LOW){
  // i2c통신으로 센서값(습도) 요청
     Wire.beginTransmission(18);
     Wire.write('H');
     Wire.endTransmission();

     Wire.requestFrom(18, 2);
     byte response[2];
     int index = 0;
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 마스터 코드

```
        // 센서값 받기를 기다림
        while (Wire.available()) {
            byte b = Wire.read();
            Serial.println(int(b));
        }
        digitalWrite (ledPin, HIGH);
        //Serial.println("Botton on");
        delay(1000);
    }
    else{           // 버튼이 눌리지 않았다면
        digitalWrite(ledPin, LOW);
    }
    delay(100);
}
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 슬레이브 코드

```
#include "DHT.h"                    // DHT.h 라이브러리를 포함
#include <Wire.h>
#define DHTPIN 2                    // DHT핀을 2번으로 정의(DATA핀)
#define DHTTYPE DHT11               // DHT타입을 DHT11로 정의
DHT dht(DHTPIN, DHTTYPE);  // DHT설정 - dht (디지털2, dht11)
char c;
int h;
int t;
void setup() {
   Wire.begin(18);
   Wire.onRequest(requestEvent); // data request to slave
   Wire.onReceive(receiveEvent);   // data slave received

   Serial.begin(9600);                 // 9600 속도로 시리얼 통신을 시작
}
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 슬레이브 코드

```
void loop() {
    h = dht.readHumidity();          // 변수 h에 습도 값을 저장
    t = dht.readTemperature();       // 변수 t에 온도 값을 저장
    Serial.print("Humidity: ");      // 문자열 “Humidity: “를 출력
    Serial.print(h);                 // 변수 h(습도)를 출력
    Serial.print("%₩t");             // %를 출력
    Serial.print("Temperature: ");   // 이하 생략
    Serial.print(t);
    Serial.println(" C");
    delay(1000);
}
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 슬레이브 코드

```
void receiveEvent(int howMany) {
    // remember the question: H=humidity, T=temperature
    while (0 < Wire.available()) {
        byte x = Wire.read();
        c = x;
        Serial.println(c);
    }
}
```

6. 버튼을 눌렀을 때 원격지 센서 값 읽기

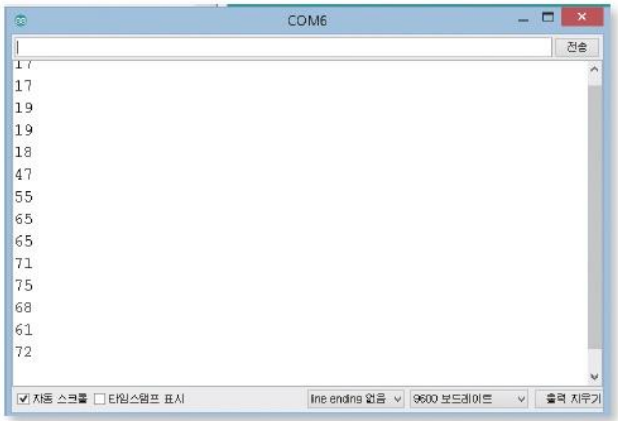
- 슬레이브 코드

```
void requestEvent() {

      // respond to the question
      if (c == 'H') {
            Wire.write(h);
            Serial.print("Current humidity =");
            Serial.println(h);
        }
        else {
            Wire.write(t);
            Serial.print("Current temperature =");
            Serial.println(t);
       }
}
```

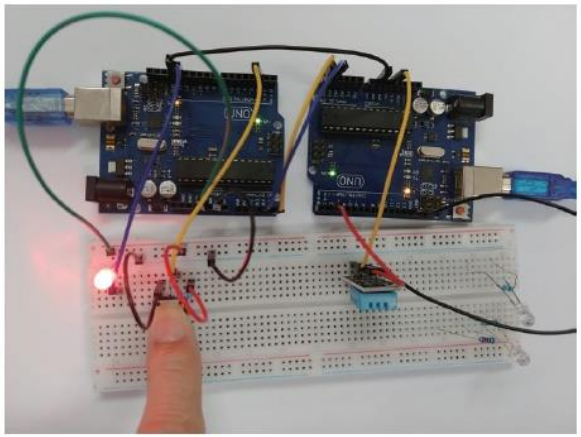
6. 버튼을 눌렀을 때 원격지 센서 값 읽기

- 실험 결과



(a) 마스터 결과

(b) 슬레이브 결과



(c)

그림 11.11 마스터에서 버튼을 눌렀을 때 결과

요약

- I2C통신을 이용한 실험을 할 수 있다.
- I2C통신은 마스터와 슬레이브의 조합이다.
- 마스터에서 슬레이브로 데이터를 보내거나 요청을 할 수 있다.
- 슬레이브는 수신된 데이터를 표시하거나 마스터의 요청에 따라 데이터를 마스터로 보낸다.
- 슬레이브는 데이터를 요청할 수 없다.