

---

# 임베디드 시스템

DHT11 온습도 센서 제어

---

In-Hyeok Kang

M23522@hallym.ac.kr

연구실: 공학관 1321호

---

# Contents

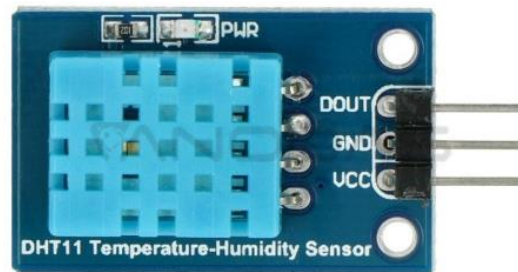
1. DHT11 온습도 센서
2. DHT11 온습도 센서 연결
3. DHT11 온습도 센서 제어

# DHT11 온습도 센서

- DHT11 온습도 센서

- 습도 및 온도 측정 부품이 결합된 센서로 디지털 신호를 출력함(Digital Output)
- 고성능 8-bit MCU (Micro Controller Unit)를 탑재함(한 번에 8-bit까지만 처리 가능)
- 저비용, 장기 안정성, 상대 습도 및 온도 측정, 우수한 품질, 빠른 응답, 간섭 방지 기능 등이 특징임
- 온습도 센서 활용 예
  - ✓ 제습기, 테스트 및 검사 장비, 기상 관측소, 가전 제품, 습도 조절기, 의료 및 기타 관련 습도 측정 및 제어 등
- 홈페이지

- ✓ <https://www.waveshare.com/temperature-humidity-sensor.htm>



DHT11 온습도 센서

## Specifications

- Temperature
  - Resolution : 1°C
  - Accuracy :  $\pm 2^{\circ}\text{C}$
  - Measuring range : 0°C ~ 50°C
- Humidity
  - Resolution : 1%RH
  - Accuracy :  $\pm 5\% \text{RH}$  (0~50°C)
  - Measuring range : 20%RH ~ 90%RH (25°C)
- Operating voltage : 3.3V ~ 5.5 V
- Recommended storage condition
  - Temperature : 10°C ~ 40°C
  - Humidity : 60%RH or below

## Applications

- Weather station
- Humidity controller
- Test & detection device

## How to Use

In the case of working with a MCU:

- VCC  $\leftrightarrow$  3.3V ~ 5.5V
- GND  $\leftrightarrow$  GND
- DOUT  $\leftrightarrow$  MCU.I/O

---

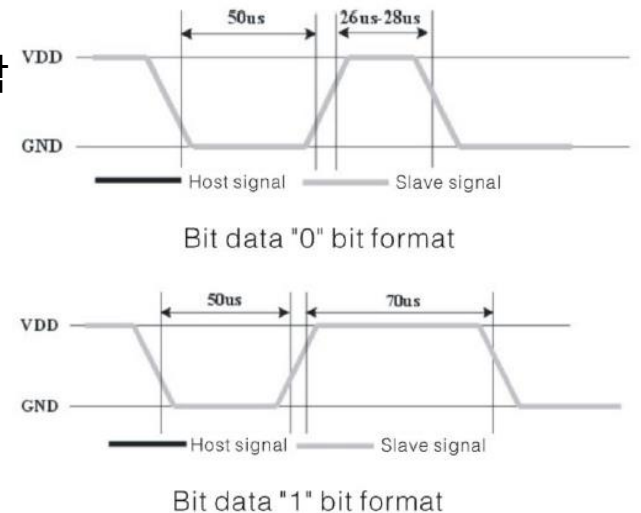
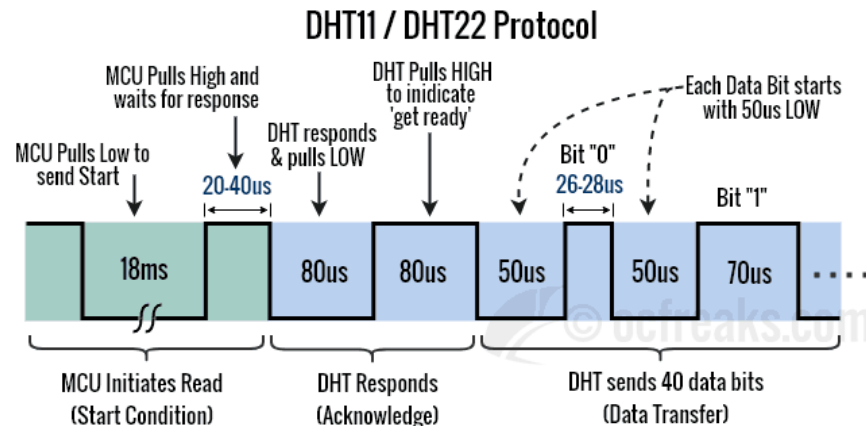
# DHT11 온습도 센서

- 시리얼 통신(Serial communication): 직렬 통신
    - Single-wire bi-directional
    - 단일 버스 (Single bus) 통신 사용
    - 하나의 데이터 라인(Data line)이 존재하며, 단일 버스 제어를 통해 데이터 교환을 수행함
    - **마스터(Master) – 슬레이브(Slave) 구조(Raspberry Pi – DHT11)**
      - ✓ 호스트가 Slave를 호출할 때만 Slave가 응답 가능 → 호스트 액세스 장치는 Single bus sequence를 엄격히 따라야 함
    - MCU와 DHT11 간의 통신에 사용되는 데이터 라인은 4.7K 또는 10K 풀업(Pull-up) 저항을 사용해 HIGH로 당겨짐 (Pull)
      - ✓ 통신이 이루어지지 않을 때 버스를 IDLE 상태로 만들기 위함
    - 라인에서 연속적인 HIGH는 IDLE 상태를 나타냄
    - MCU는 버스의 Master 역할을 하므로 통신 시작(Read)을 담당함
    - DHT11은 항상 Slave로 유지되며 MCU가 요청할 때 데이터로 응답함
-

# DHT11 온습도 센서

- 통신 과정(Communication Process)

1. Line이 IDLE 상태일 때 MCU는 Line을 18ms 동안 LOW로 당김
2. 이후, MCU는 Line을 약 20~40 $\mu$ s 동안 HIGH로 당김
3. DHT11은 MCU를 통해 이 경우를 START로 감지하고, Line을 80 $\mu$ s 동안 LOW로 당겨 응답함
4. 다음으로, DHT11은 데이터를 보낼 준비가 되었거나 준비되었음을 나타내는 80 $\mu$ s 동안 Line을 HIGH로 당김
5. 이후, DHT11은 40-bit의 데이터를 전송함. 각 bit는 50 $\mu$ s LOW로 시작하고 "0"의 경우 26-28 $\mu$ s, "1"의 경우 70  $\mu$ s 동안 HIGH로 유지됨
6. 통신이 완료되면, 풀업 저항에 의해 Line이 HIGH로 당겨지고 IDLE 상태로 진입함

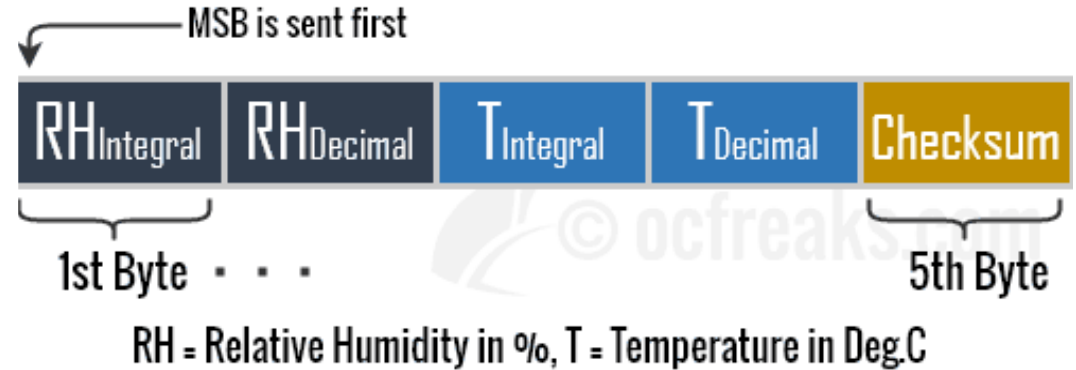


# DHT11 온습도 센서

- DHT11 Data Format

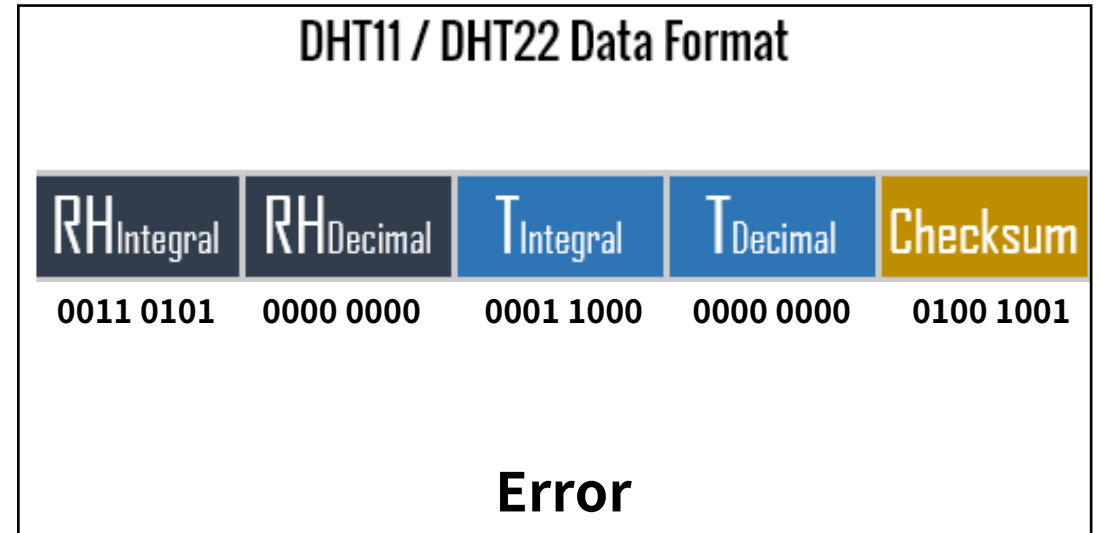
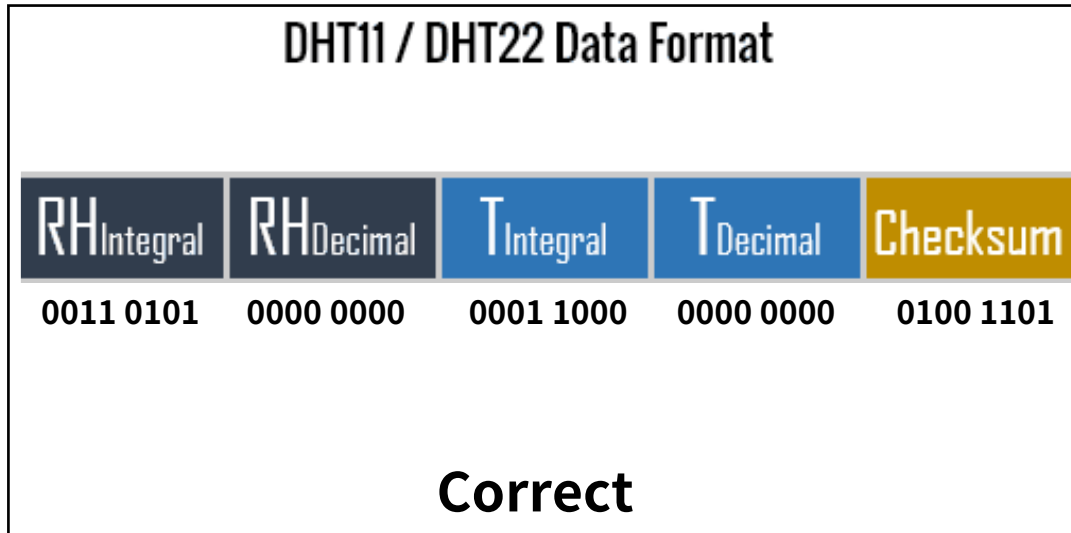
- 40-bit(5-byte)
- **MSB(Most Significant Bit) first**
- DHT11의 경우, 2번째 및 4번째 Byte가 항상 0임
- 각 Byte 의미
  - ✓ **1st Byte:** Relative Humidity Integral Data in % (Integer Part)
  - ✓ **2nd Byte:** Relative Humidity Decimal Data in % (Fractional Part) – **Zero for DHT11**
  - ✓ **3rd Byte:** Temperature Integral in Degree Celsius (Integer Part)
  - ✓ **4th Byte:** Temperature in Decimal Data in % (Fractional Part) – **Zero for DHT11**
  - ✓ **5th Byte:** Checksum (Last 8 bits of {1st Byte + 2nd Byte + 3rd Byte+ 4th Byte})
- DHT11로 유효한 측정 값을 얻기 위해서는 Probing Interval (데이터 요청 주기)을 1초 이상으로 유지해야 함
  - ✓ 1초 이상의 간격을 유지해야 센서가 새로운 데이터를 정확히 측정하고 응답할 수 있음

## DHT11 / DHT22 Data Format







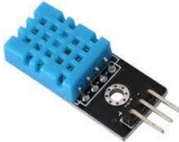
# DHT11 온습도 센서

- DHT11 Data Format
  - Ex)



# DHT11 온습도 센서 연결

- 1. 구성품 준비

번호	구성요소	사진
1	Raspberry Pi 본체	<div> &lt;Raspberry Pi 3 Model B+&gt;</div> <div> &lt;Raspberry Pi 4 Model B&gt;</div>
2	점프 와이어(F/F 7개)	
3	LED 센서 모듈	
4	온습도 센서 모듈	



# DHT11 온습도 센서 연결

- 2. 구성품 연결

- [LED 센서 모듈]

✓ [점프 와이어(F/F)]로 연결

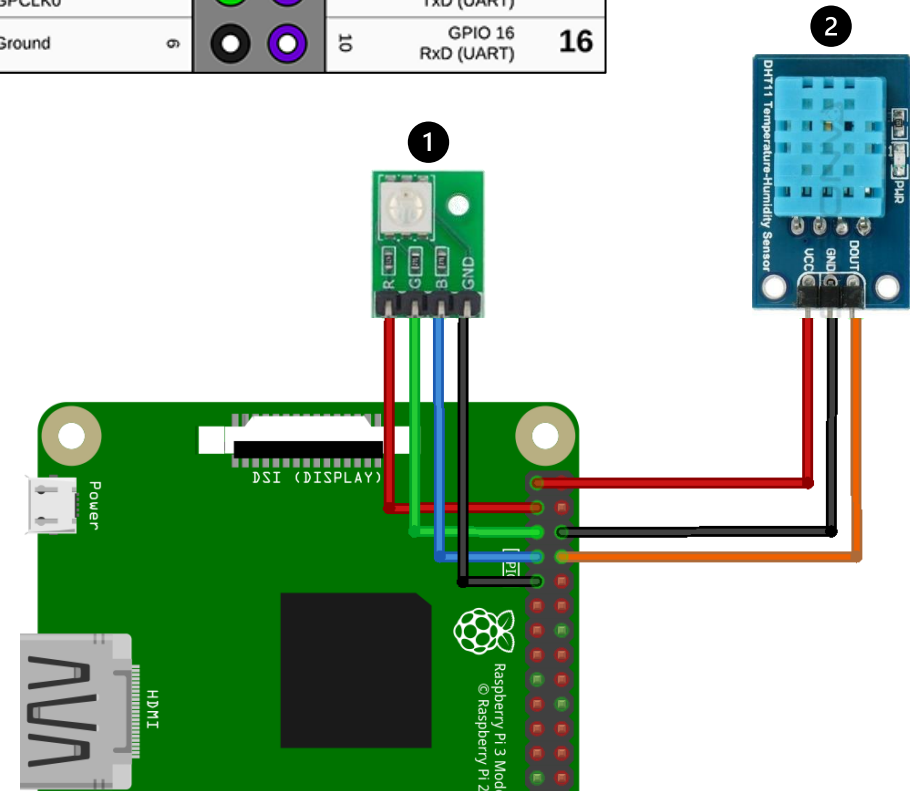
LED 센서 모듈	GPIO Pins
R	8
G	9
B	7
GND	Ground

- [DHT11 온습도 센서]

✓ [점프 와이어(F/F)]로 연결

DHT11 온습도 센서	GPIO Pins
DOUT	15
GND	Ground
VCC	3.3 VDC

GPIO#	NAME		NAME	GPIO#
	3.3 VDC Power	1	5.0 VDC Power	2
8	GPIO 8 SDA1 (I2C)	3	5.0 VDC Power	4
9	GPIO 9 SCL1 (I2C)	5	Ground	6
7	GPIO 7 GPCLK0	7	GPIO 15 TxD (UART)	15
	Ground	9	GPIO 16 RxD (UART)	16



# DHT11 온습도 센서 제어

- 1. DHT11.java 소스 코드(이어서 작성)

```
package week4;

import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
import com.pi4j.wiringpi.Gpio;

public class DHT11 {
    private static final int MAXTIMINGS = 85;
    private final int[] dht11_f = { 0, 0, 0, 0, 0 }; // DHT11 data format (5 bytes)

    public DHT11() {
        // Setup wiringPi
        if (Gpio.wiringPiSetup() == -1) {
            System.out.println(" ==>> GPIO SETUP FAILED");
            return;
        }
    }
}
```



```
public float[] getData(final int pin) {
    int laststate = Gpio.HIGH; // signal 상태 변화를 알기 위해 기존 상태를 기억
    int j = 0; // 수신한 Bit의 index counter
    float h = -99; // 습도
    float c = -99; // 섭씨 온도
    float f = -99; // 화씨 온도

    // Integral RH, Decimal RH, Integral T, Decimal T
    dht11_f[0] = dht11_f[1] = dht11_f[2] = dht11_f[3] = dht11_f[4] = 0;

    // 1. DHT11 센서에게 start signal 전달
    Gpio.pinMode(pin, Gpio.OUTPUT);
    Gpio.digitalWrite(pin, Gpio.LOW);
    Gpio.delay(18);

    // 2. Pull-up -> 수신 모드로 전환 -> 센서의 응답 대기
    Gpio.digitalWrite(pin, Gpio.HIGH);
    Gpio.pinMode(pin, Gpio.INPUT);

    // 3. 센서의 응답에 따른 동작
    for (int i = 0; i < MAXTIMINGS; i++) {
        int counter = 0;
        while (Gpio.digitalRead(pin) == laststate) { // Gpio pin 상태가 바뀌지 않으면 대기
            counter++;
            Gpio.delayMicroseconds(1);
            if (counter == 255) {
                break;
            }
        }
        laststate = Gpio.digitalRead(pin);
        if (counter == 255) {
            break;
        }

        // 각각의 bit 데이터 저장
        if (i >= 4 && i % 2 == 0) { // 첫 3개의 상태 변화는 무시, laststate가 low에서 high로 바뀔 때만 값을 저장
            // data 저장
            dht11_f[j / 8] <<= 1; // 0 bit
            if (counter > 16) {
                dht11_f[j / 8] |= 1; // 1 bit
            }
            j++;
        }
    }
}
```

# DHT11 온습도 센서 제어

- 1. DHT11.java 소스 코드(이어서 작성)
  - getChecksum(): 체크섬 계산을 통해 데이터가 정상적으로 수신되었는지 확인하는 메서드
    - ✓ dht11\_f[4]는 앞의 4개의 바이트의 합과 일치해야 함
  - main(): 메인 메서드
    - ✓ DHT11 - GPIO 15번 핀 사용 (DOUT)
    - ✓ RGB LED - GPIO 8, 9, 7번 핀 사용 (RGB)
    - ✓ 온습도 변화에 따른 LED On/Off 조건
      - 온도가 29도 이상이면 Red LED On
      - 습도가 50% 이상이면 Green LED On

```
// Checksum 확인
// Check we read 40 bits (8 bit x 5) + verify checksum in the last
if (j >= 40 && getChecksum()) {
    h = (float) ((dht11_f[0] << 8) + dht11_f[1]) / 10;
    if (h > 100) {
        h = dht11_f[0]; // for DHT11
    }
    c = (float) (((dht11_f[2] & 0x7F) << 8) + dht11_f[3]) / 10;
    if (c > 125) {
        c = dht11_f[2]; // for DHT11
    }
    if ((dht11_f[2] & 0x80) != 0) {
        c = -c;
    }
    f = c * 1.8f + 32;
    System.out.println("Humidity = " + h + "% Temperature = " + c + "°C | " + f + "°F");
}
else {
    System.out.println("Checksum Error");
}

float[] result = { h, c, f };
return result;
}
```

**getChecksum()**

```
public static void main(final String[] args) throws InterruptedException {
```

**main()**

```
Thread.sleep(1500); // Probing interval
}
```

```
}
```

# DHT11 온습도 센서 제어

- 2. JAR 파일 생성 후 XFTP를 통해 Raspberry Pi로 전송
- 3. Raspberry Pi에서 JAR 파일 실행
  - `sudo java -jar DHT11.jar`
- 4. 결과
  - 습도가 50%보다 높은 경우 → Green LED On
  - 온도가 29°C보다 높은 경우 → Red LED On

```
Humidity = 70.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 73.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 75.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 76.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 78.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 79.0% Temperature = 28.0℃ | 82.399994°F  
Humidity = 80.0% Temperature = 28.0℃ | 82.399994°F
```





감사합니다

Thank You

