스마트 환기 장치

20235128 김민준

목차

배경 및 제안 아이디어

- 배경
  1. 다수의 아파트 iot 시스템엔 공기질 정화에 관한 장치가 없음

- 기존 방법과 한계점
  1. 신식 아파트에선 적용된 사례를 찾아볼수 있으나 보통의 준신식 아파트 에서는 거의 찾아볼수 없음. 따라서 불편함을 감수하고 수동으로 창문을 열어 환기 시켜 줘야함. 외부에서는 환기를 진행할수 없는 단점이 존제

- 제안 아이디어
  1. 창문에 환풍기 장치를 달아 외부에서도 방안의 온도와 습도를 실시간으로 확인할 수 있으며 원격으로 환풍기를 작동시킬수 있는 장치를 만들어 설치하면 이 문제를 극복할수 있음.

프로젝트 내용

- 시스템 구조

CoAP 클라이언트/UI: CoAP 클라이언트와 사용자 인터페이스(UI)가 통합되어, 사용자가 CoAP 프로토콜을 통해 스마트 환기 장치를 제어하고, 실시간으로 상태를 모니터링할 수 있는 환경을 제공. UI는 다양한 버튼을 통해 CoAP 요청을 전송하며, 응답을 화면에 표시.

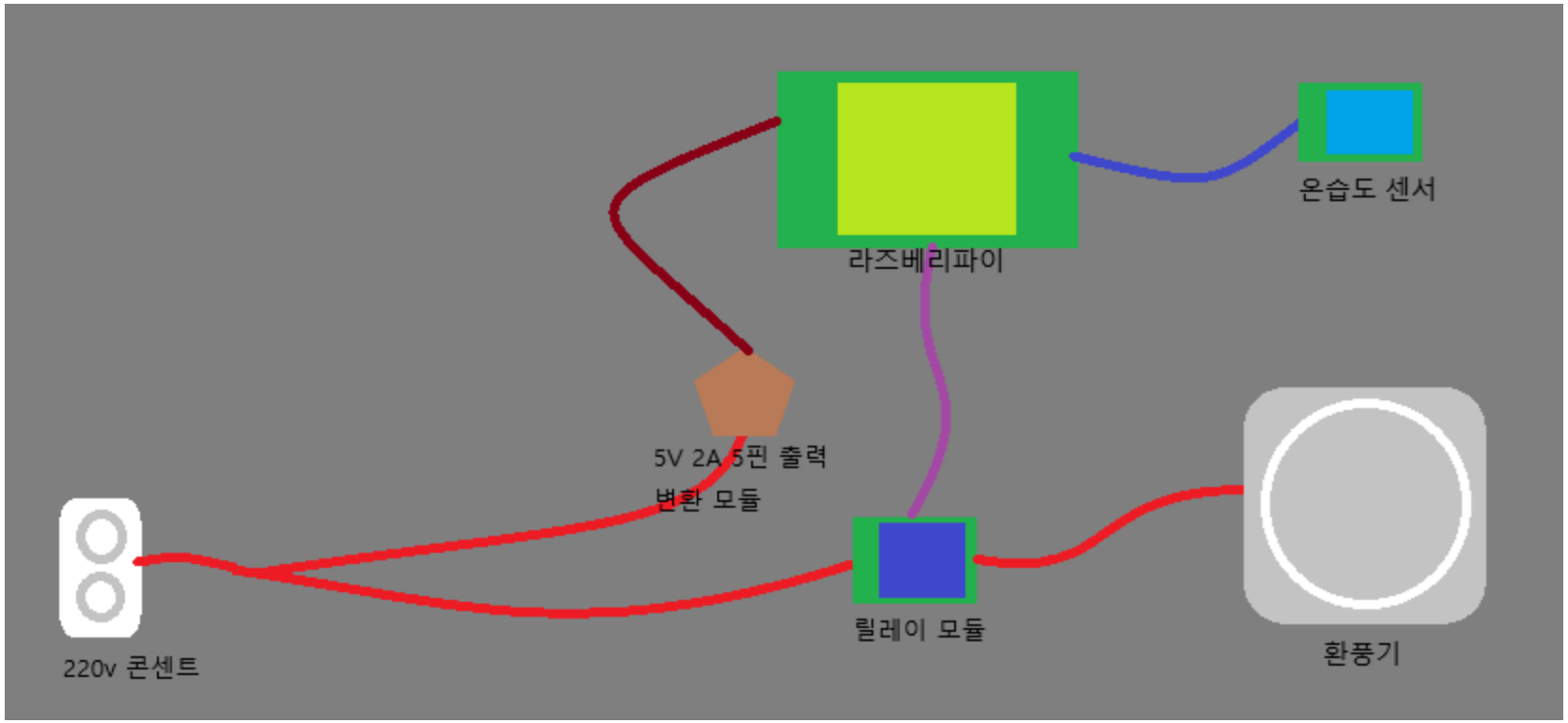
CoAP 서버: CoAP 프로토콜을 이용하여 요청을 처리하고, 스마트 환기 장치의 다양한 리소스를 관리하는 서버. 클라이언트가 요청하는 데이터를 제공하고, 장치의 상태를 제어하기 위해 적절한 응답을 전송.

릴레이 제어: CoAP 클라이언트를 통해 릴레이 상태를 제어할 수 있는 기능. 이를 통해 환기 장치의 작동을 제어하거나, 외부 장치와의 상호작용을 가능하게 함.

DHT11 센서: 온도 및 습도 데이터를 측정하여 스마트 환기 장치가 작동할 수 있도록 환경 정보를 제공. DHT11 센서 데이터를 CoAP 요청을 통해 서버에서 클라이언트로 전달하고, 이를 UI에서 확인할 수 있도록 함.

하드웨어 및 소프트웨어 구현

- 하드웨어

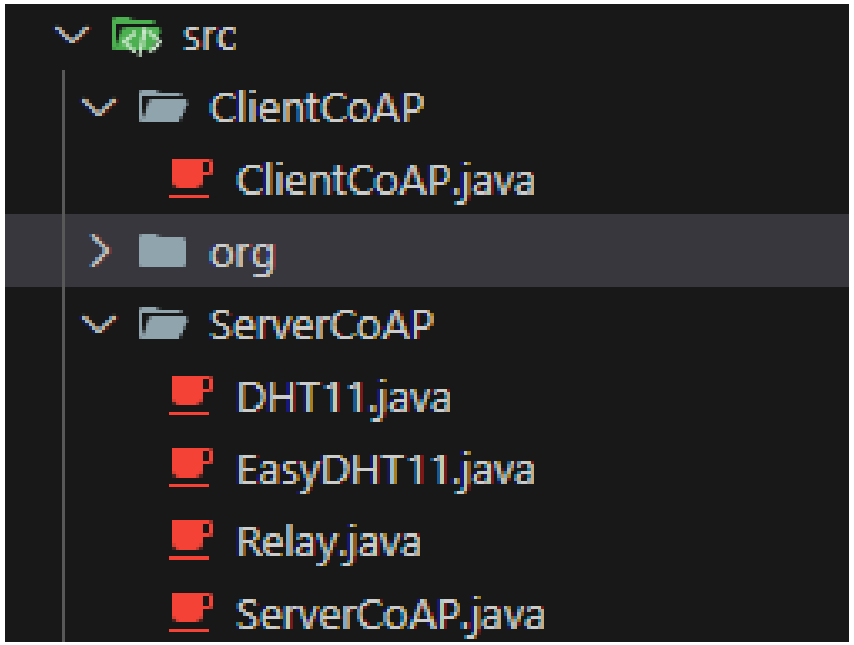


하드웨어 및 소프트웨어 구현

- 소프트웨어

  Java 기반으로 개발

  사진 속의 소스코드를 포함



하드웨어 및 소프트웨어 구현

- 소프트웨어 [EasyDHT11 클래스]

1. 이 클래스는 DHT11 센서에서 실제 데이터를 읽어오는 클래스입니다. getData() 메서드는 GPIO 핀에서 데이터를 읽고, 체크섬을 확인한 후 온도와 습도를 반환합니다.

```java
package ServerCoAP;

import com.pi4j.wiringpi.Gpio;

public class EasyDHT11 {
    private static final int MAXTIMINGS = 85;
    private final int[] dht11_f = { 0, 0, 0, 0, 0 }; // DHT11 data format (5 bytes)

    public EasyDHT11() {
        // Setup wiringPi
        if (Gpio.wiringPiSetup() == -1) {
            System.out.println(x:" ==>> 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++) {
```

하드웨어 및 소프트웨어 구현

- 소프트웨어 [DHT11 클래스]

1. 이 클래스는 DHT11 온습도 센서를 읽고 CoAP 클라이언트가 요청했을 때 온도와 습도 정보를 반환하는 역할을 합니다.

DHT11 클래스는 EasyDHT11 클래스와 서로 의존 하며 작동합니다

```java
public class DHT11 extends BasicCoapResource {
    /* Field */
    private float temperature; // 온도 값
    private float humidity; // 습도 값

    /* Constructor */
    DHT11(String path, String value, CoapMediaType mediaType) {
        super(path, value, mediaType);
    }

    DHT11() {
        this(path:"/dht11", value:"0", CoapMediaType.text_plain);
        this.temperature = 0.0f;
        this.humidity = 0.0f;
    }

    /* Method */
    private void readDHT11() {
        final boolean FOR_TEST = false;
        if(FOR_TEST){
            this.temperature = 25.5f; // 가상의 온도 값
            this.humidity = 60.0f; // 가상의 습도 값
        }else{
            final EasyDHT11 dht = new EasyDHT11();
            float[] read = dht.getData(pin:28); // Read data from GPIO 28
            this.temperature = read[0];
            this.humidity = read[1];
        }
    }

    @Override
    public synchronized CoapData get(List<String> query, List<CoapMediaType> mediaTypesAccepted) {
        return this.get(mediaTypesAccepted);
    }
```

하드웨어 및 소프트웨어 구현

- 소프트웨어 [Relay 클래스]

1. 이 클래스는 릴레이 모듈을 제어하고, CoAP 요청을 받아 상태를 변경할 수 있게 합니다

```java
public class Relay extends BasicCoapResource {
    /* Field */
    private GpioController gpio;
    private GpioPinDigitalOutput pin;
    private boolean state; // Boolean으로 상태 관리

    /* Constructor */
    Relay(String path, String value, CoapMediaType mediaType) {
        super(path, value, mediaType);
    }

    Relay() {
        this(path:"/relay", value:"off", CoapMediaType.text_plain);
        this.gpio = GpioFactory.getInstance();
        this.pin = this.gpio.provisionDigitalOutputPin(RaspiPin.GPIO_27, PinState.LOW); // Relay 모듈의 GPIO 핀은 27번
        this.state = false; // 초기 상태는 off(false)
    }

    /* Method */
    @Override
    public synchronized CoapData get(List<String> query, List<CoapMediaType> mediaTypesAccepted) {
        return this.get(mediaTypesAccepted);
    }

    @Override
    public synchronized CoapData get(List<CoapMediaType> mediaTypesAccepted) {
        // 텍스트 형식으로 반환 (보통 CoAP는 텍스트일 때 string으로 응답하지만, byte로 변환한 후 적절한 미디어 타입 지정)
        String data = state ? "ON" : "OFF";  // boolean 상태를 "ON" 또는 "OFF"로 문자열 변환

        return new CoapData(data.getBytes(), CoapMediaType.text_plain);
    }


    @Override
    public synchronized boolean setValue(byte[] value) {
        // byte[] 값을 boolean으로 변환
        if (value != null && value.length > 0) {
```

Demo 영상

!!! 영상이 ppt에 안들어가서 별도로 첨부 했습니다 !!!

감사합니다.

20235128 김민준