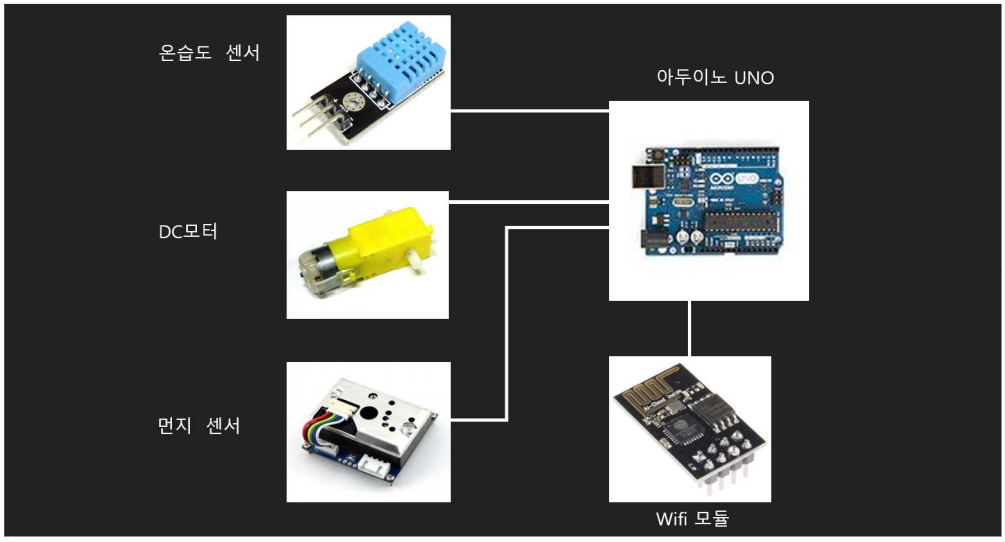
과제 보고서

작품(과제)명	창문 자동환기 시스템
개발자	엄정기(201301968), 김지석(201301944)
1. 개발동기 (필요성)	- 요즘의 공기상태가 좋지 안항, 장시간 창문을 개방할 경우, 피해가 있음 - 방 안의 미세먼지 배출의 필요성을 느낌
2. 개발목표	1. 미세먼지의 측정 값에 따라 방 안의 미세먼지의 유입을 줄임 2. 온습도를 측정하여 방 안의 상태를 쾌적하게 유지
3. 관련연구 (선행기술)	LG 하우시스, 스마트윈도우
4. 과제해결 방안	창문 틀에 미세먼지 센서를 부착하여, 실내의 미세먼지를 측정하고, 모터를 이용해 방 안의 미세먼지의 농도를 줄일 수 있도록 한다.
5. 개발환경	- 아두이노
6. 시스템 설계	< 아두이노 전체적인 설계도 >  - 온습도 센서(DHT11)을 이용하여서, 온도와 습도를 측정한다. - 먼지 센서(GP2Y1010AU0F)을 이용하여 먼지의 농도를 측정한다. - DC기어드 모터를 이용하여 창문을 열고, 닫고 제어한다. - 창문을 열고 닫을 때에는 타이밍벨트와 풀리를 이용하여 창문을 고정시키고, 움직인다. - WIFI 모듈(ESP8266)을 이용하여, WIFI에 연결 시킨다. - thingspeak에 온습도 센서와 먼지 센서를 등록하여, 측정값을 확인한다. - 먼지 센서의 값에 따라 창문을 열고 닫는다.

7. 상세 설계	- 온습도 센서를 부착하여, 실시간으로 측정한 값을 thingspeak로 보내고, 그 값을 thingspeak앱을 이용하여서, 스마트폰으로 데이터 확인 - 먼지 센서의 값을 측정하여서 먼지 농도가 0.1을 초과하면 문을 개방한다. - 먼지농도가 0.1이하가 되면 공기가 청정됐을거라고 판단하여 창문을 닫고, 쿨랭팬의 동작을 멈춘다. - 창문을 개방하게 되면 쿨링 팬을 돌려서 효과적인 공기순환을 돕는다. - 신뢰성 있는 wifi에 연결시킨다. - 창문을 열고 닫는 것을 확인할 수 있도록, A4종이상자를 이용하여 집의 모형을 구현한다 - 창문은 아크릴판을 이용해서 2개의 창문을 구상하고, 1개의 창문은 고정하고, 1개의 창문은 모터를 이용해서 움직인다 - 자동문이 열려있으면 중복으로 안 열리게 하고, 닫혀있으면 중복으로 문이 닫히지 않도록 한다. - 방 안의 먼지의 농도가 높아서 창문을 열었는데, 실외의 먼지농도가 더 높으면 창문을 닫는다.
8. 결론	- 먼지의 농도라 따라서 창문을 개방할 수 있고, 먼지를 환기할 수 있음 - 먼지 센서를 통해서 미세먼지의 값을 측정하고, 측정한 값에 따라서, 먼지의 농도가 높으면 창문을 열고, 낮으면 창문을 닫는 것이 가능 - WIFI 모듈을 연결에 성공했고, Thingspeak를 통해서 측정값을 전달하는데 성공 - 창문이 열리면 쿨링팬 동작 가능 (전원이 부족한지 처음 시작이 살짝 건드려줘야함) - 창문으로 움직을 아크릴판의 창문에 타이밍벨트를 부착하여 DC기어드모터로 타이밍벨트를 움직이면 아크릴판까지 움직임
9. 참고문헌	● ESP8266 WIFI 펌웨어 업그레이드 - http://deneb21.tistory.com/269 ● ESP8266 WIFI 연결 - http://webnautes.tistory.com/755 ● ESP8266 - http://deneb21.tistory.com/274 ● DC기어드모터 - http://deneb21.tistory.com/281 ● Thingspeak에 데이터 전송 - http://deneb21.tistory.com/355 ● L298N 모터드라이버 설명서 ● 먼지센서 회로 http://arduinosensor.tumblr.com/post/134224223120/gp2y1010au0fpart1

첨부: **1. 소스코드**

2. 동영상 (별도 첨부)

1. 소스코드

```
#include <DHT11.h>
#include <SoftwareSerial.h>    //라이브러리 불러옴
#include <stdlib.h>
#define DEBUG true

SoftwareSerial esp8266(10, 11); // RX/TX 설정, serial 객체생성

int doortime = 0;
// 문이 열리고 닫히는 우선순위
int priority = 0, tmppriority = -1;
// 문을 닫을지 열지 결정
int isopen = 0;

int dhtpin = 3, err;
int timeman = 12;

DHT11 dht11(dhtpin);

// 0 : 문이 닫혔을때, 1 : 문이 열렸을 때
int sts = 0;

int measurePin = 0; //Connect dust sensor to Arduino A0 pin
int ledPower = 2;   //Connect 3 led driver pins of dust sensor to Arduino D2

int samplingTime = 280;
int deltaTime = 40;
int sleepTime = 9680;
int redPin = 7, greenPin = 6, bluePin = 8;

// 온도, 습도, 먼지농도
float temp = 0, humi = 0, dust = 0;
float tmptemp = 0, tmphumi = 0, tmpdust = 0;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

// 자신의 thingspeak 채널의 API key 입력
String apiKey = "0Y5054PM0EY6PM07";


void setup() {

        //시리얼통신속도 9600보드레이트 설정
        Serial.begin(9600); // 시리얼 시작
        esp8266.begin(9600); // 와이파이

         // 먼지센서 세팅
        pinMode(ledPower, OUTPUT);
        pinMode(4, OUTPUT);

        // 기어드모터 PinMode 세팅
        pinMode(12, OUTPUT);
        pinMode(13, OUTPUT);

        pinMode(7, OUTPUT);
        pinMode(8, OUTPUT);

        sendData("AT+RST₩r₩n", 2000, DEBUG); // reset module
        sendData("AT+CIOBAUD?₩r₩n", 2000, DEBUG); // check baudrate (redundant)
        sendData("AT+CWMODE=3₩r₩n", 1000, DEBUG); // configure as access point (working mode: AP+STA)
```

```
        sendData("AT+CWLAP₩r₩n", 3000, DEBUG); // list available access points
        sendData("AT+CWJAP=₩"jungki₩",₩"1234567890₩"₩r₩n", 5000, DEBUG); // join the access point
        sendData("AT+CIFSR₩r₩n", 1000, DEBUG); // get ip address
        sendData("AT+CIPMUX=1₩r₩n", 1000, DEBUG); // configure for multiple connections
        sendData("AT+CIPSERVER=1,80₩r₩n", 1000, DEBUG); // turn on server on port 80
}

void loop() {

        // DHT22 온습도 불러옴
        getDust();
        TempHumi();

        // 3분마다 점검을 한다.
        if (timeman >= 20) {
                PrioritySet(); // 계산을 해서 우선순위를 따진다.
                   // 문이 닫혀있을 때
                Serial.print("sts : ");
                Serial.print(sts);
                Serial.print("  ");
                if (sts == 0 && isopen == 1) {
                        Serial.println("  문 열림");
                        MotorOpen();
                        delay(1180);
                        MotorStop();

                        digitalWrite(12, HIGH);
                        digitalWrite(13, LOW);
                        analogWrite(5, 250);

                        tmppriority = priority;
                        sts = 1;
                        }
                        // 문이 열려있을 때
                else if (sts == 1 && isopen == 0) {
                        Serial.println("문 닫힘");
                        MotorClose();
                        delay(1180);
                        MotorStop();

                        digitalWrite(12, HIGH);
                        digitalWrite(13, LOW);
                        analogWrite(5, 0);

                        tmppriority = priority;
                        sts = 0;
                }
                timeman = 0;
        }

        // 1분마다 thingspeak에 데이터를 전송
        if ((timeman % 10) == 0) { // 1분마다 thingspeak에 데이터전송
                SendToThingspeak();
        }
        delay(1000);
        timeman += 1;
}

void PrioritySet() {

        if (dust > 0.1) {
                if (sts == 0) {
                        isopen = 1; // 문 열어야 한다
                        tmpdust = dust;
                }
```

```
                else if (sts == 1 && dust > tmpdust + 0.2) {
                isopen = 0; // 문을 열었는데 먼지가 더 심해지면 문 닫아야 한다.
                // delay(300000); // 5분동안 못열음
                }
        }
        else if (sts == 1 && dust <= 0.1) {
                isopen = 0;
        }
}

void TempHumi() {
        if ((err = dht11.read(humi, temp)) == 0) //온도, 습도 읽어와서 표시
        {
                Serial.print("temperature:");
                Serial.print(temp);
                Serial.print("  humidity:");
                Serial.print(humi);
                Serial.print("  Dust:");
                Serial.print(dust);
                Serial.println();
        }
}

void PanGo() {
        digitalWrite(12, HIGH);
        digitalWrite(13, LOW);
        analogWrite(5, 0);
}

void PanStop() {

}

// 모터를 작동을 제어
void MotorStop() {
        //최대속도의 50%로 정회전
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        analogWrite(6, 0);
}

void MotorOpen() {
        digitalWrite(7, LOW);
        digitalWrite(8, HIGH);
        analogWrite(6, 120);
}

void MotorClose() {
        digitalWrite(7, HIGH);
        digitalWrite(8, LOW);
        analogWrite(6, 120);
}

// 먼지의 값을 계산해서 반환
void getDust() {
        digitalWrite(ledPower, LOW); // power on the LED
        delayMicroseconds(samplingTime);

        voMeasured = analogRead(measurePin); // read the dust value

        delayMicroseconds(deltaTime);
        digitalWrite(ledPower, HIGH); // turn the LED off
        delayMicroseconds(sleepTime);

        calcVoltage = voMeasured * (5.0 / 1024.0);
```

```
        dustDensity = 0.17 * calcVoltage - 0.1;

        dust = dustDensity;
}

void SendToThingspeak() {
        // String 변환
        char buf[16];
        String strTemp = dtostrf(temp, 4, 1, buf);
        String strHumi = dtostrf(humi, 4, 1, buf);
        String strDust = dtostrf(dust, 4, 1, buf);

        // TCP 연결
        String cmd = "AT+CIPSTART=₩"TCP₩",₩"";
        cmd += "184.106.153.149"; // api.thingspeak.com 접속 IP
        cmd += "₩",80";           // api.thingspeak.com 접속 포트, 80
        esp8266.println(cmd);

        if (esp8266.find("Error")) {
                Serial.println("AT+CIPSTART error");
        return;
        }
        // GET 방식으로 보내기 위한 String, Data 설정
        String getStr = "GET /update?api_key=";
        getStr += apiKey;
        getStr += "&field1=";
        getStr += String(strTemp);
        getStr += "&field2=";
        getStr += String(strHumi);
        getStr += "&field3=";
        getStr += String(strDust);
        getStr += "₩r₩n₩r₩n";

        // Send Data
        cmd = "AT+CIPSEND=";
        cmd += String(getStr.length());
        esp8266.println(cmd);

        if (esp8266.find(">")) {
                esp8266.print(getStr);
        }
        else {
                esp8266.println("AT+CIPCLOSE");
                // alert user
        }
}

String sendData(String command, const int timeout, boolean debug) {
        String response = "";
        esp8266.print(command); // send the read character to the esp8266
        long int time = millis();

        while ((time + timeout) > millis()) {
                while (esp8266.available()) {
                        // The esp has data so display its output to the serial window
                        char c = esp8266.read(); // read the next character.
                        response += c;
                }
        }
        if (debug) {
                Serial.print(response);
        }
        return response;
}
```