

피지컬 컴퓨팅 실습

박종화

(경기과학고등학교, 교사)

1. 아두이노 무드 램프 만들어 보기

가. 무드램프


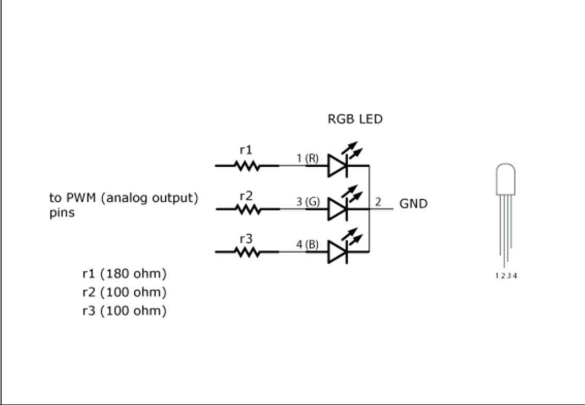
무드 램프는 색상이 변하는 램프로써 다양한 시각적인 효과를 낼 수 있는 장점이 있다. 색상을 조합하기 위해서는 빛의 삼원색을 나타내는 빨강, 파랑, 노랑의 세 가지 LED를 사용하면 된다. 판매되는 LED 중에는 이러한 세 가지 색상을 한꺼번에 표시할 수 있는 RGB LED가 있다. 이번 프로젝트에서는 앞에서 배운 RGB LED를 컨트롤 하는 방법과 더불어 이를 프로세싱이라는 언어를 이용하여 보다 시각적으로 컨트롤 할 수 있는 GUI 프로그램을 제작해 본다.



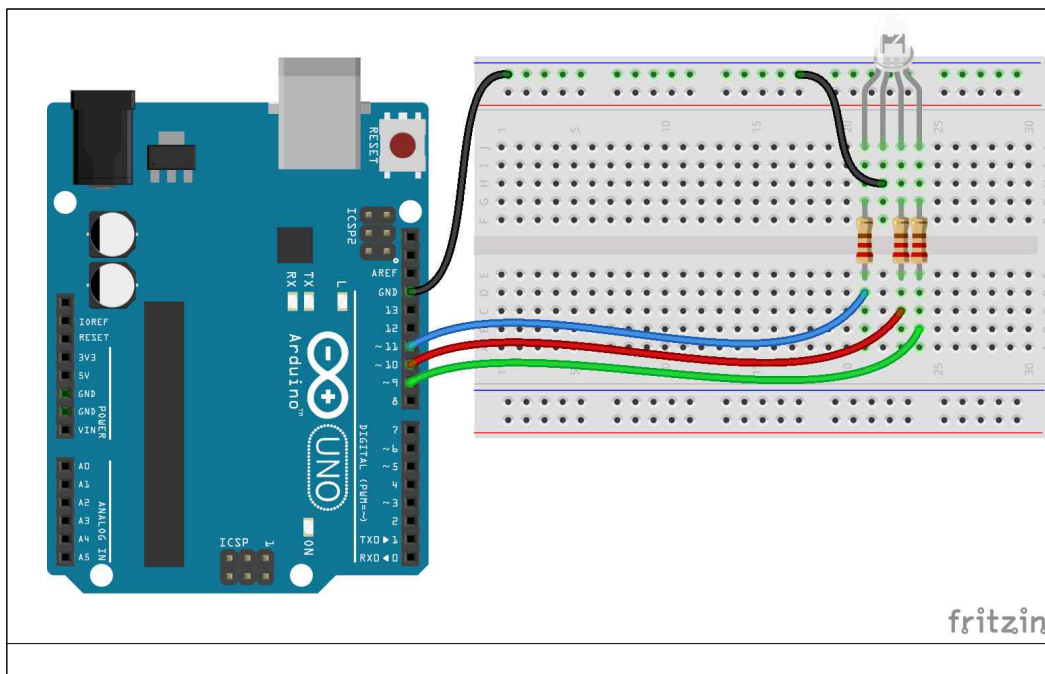
아두이노 무드 램프

<http://www.instructables.com/>

나. 하드웨어 연결하기

	
RGB LED	회로 연결

RGB LED는 세 가지 색상을 동시에 표현할 수 있으며 보통 하나의 GND를 공통으로 사용하는 경우가 많다. 사용하고 있는 LED의 정확한 사양을 확인 한 후 작업을 하도록 한다. 위 그림에서는 다리가 가장 긴 핀이 공통 GND로 사용이 되고 있는 것을 알 수 있다. 다음과 같이 간단한 회로를 구성한다.



아날로그 출력을 통해 밝기를 조절하여 원하는 색상을 표현할 것이므로 RGB의 각각의 단자는 아두이노의 PWM 핀에 연결하도록 한다.

다. 프로그램 작성

앞서 다음과 같이 직접 아두이노에서 RGB LED를 제어하는 것을 살펴보았다.

```
int REDPin = 9;
int GREENPin = 10;
int BLUEPin = 11;
int brightness = 0;
int increment = 5;

void setup()
{
  pinMode(REDPin, OUTPUT);
  pinMode(GREENPin, OUTPUT);
  pinMode(BLUEPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  brightness = brightness + increment;

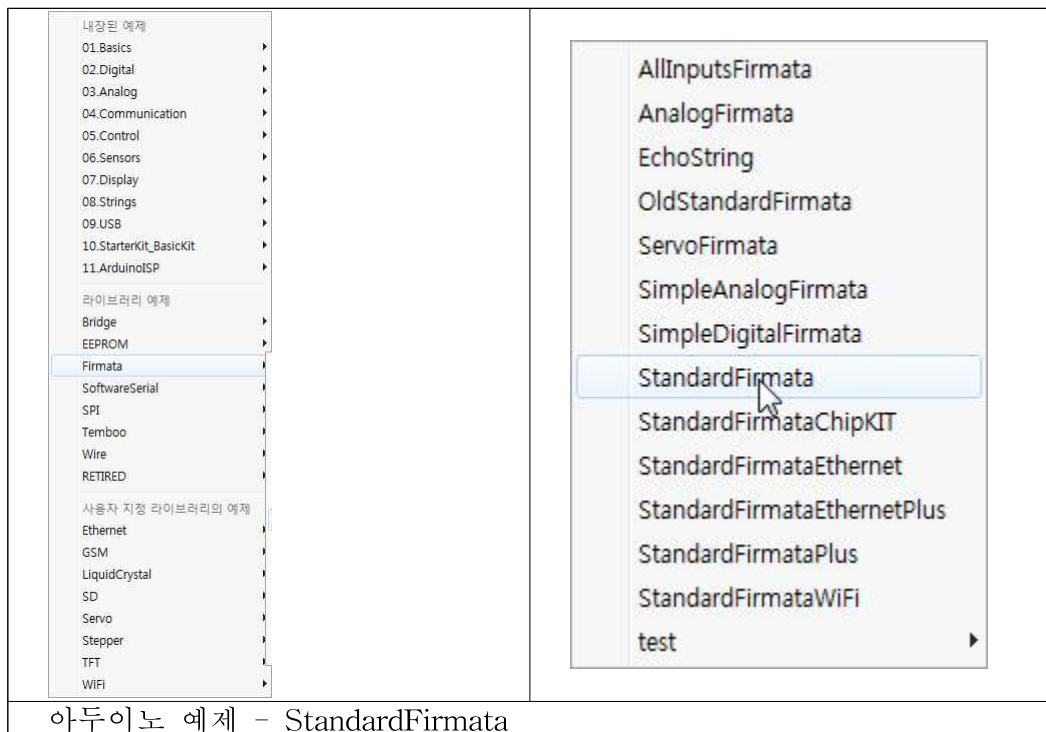
  if (brightness <= 0 || brightness >= 255)
    increment = -increment;

  brightness = constrain(brightness, 0, 255);
  analogWrite(REDPin, brightness);
  analogWrite(GREENPin, brightness);
  analogWrite(BLUEPin, brightness);

  delay(20); // wait for 20 milliseconds to see the dimming effect
}
```

라. Firmata 업로드

Firmata는 호스트 컴퓨터와 마이크로 컨트롤러 보드가 통신을 할 수 있도록 해주는 일반적인 프로토콜을 뜻한다. 이를 통해 호스트 쪽에서 프로토콜을 지원하는 프로그램으로 작성된 어떤 언어든 아두이노와 통신을 수행할 수 있게 된다. 물론 아두이노 측에 Firmata 프로토콜 프로그램을 업로드 하여야 한다. 이번 프로젝트에서는 이 프로토콜을 아두이노에 업로드 하고 프로세싱이라는 프로그래밍 언어를 이용하여 GUI 를 통한 제어를 해본다.

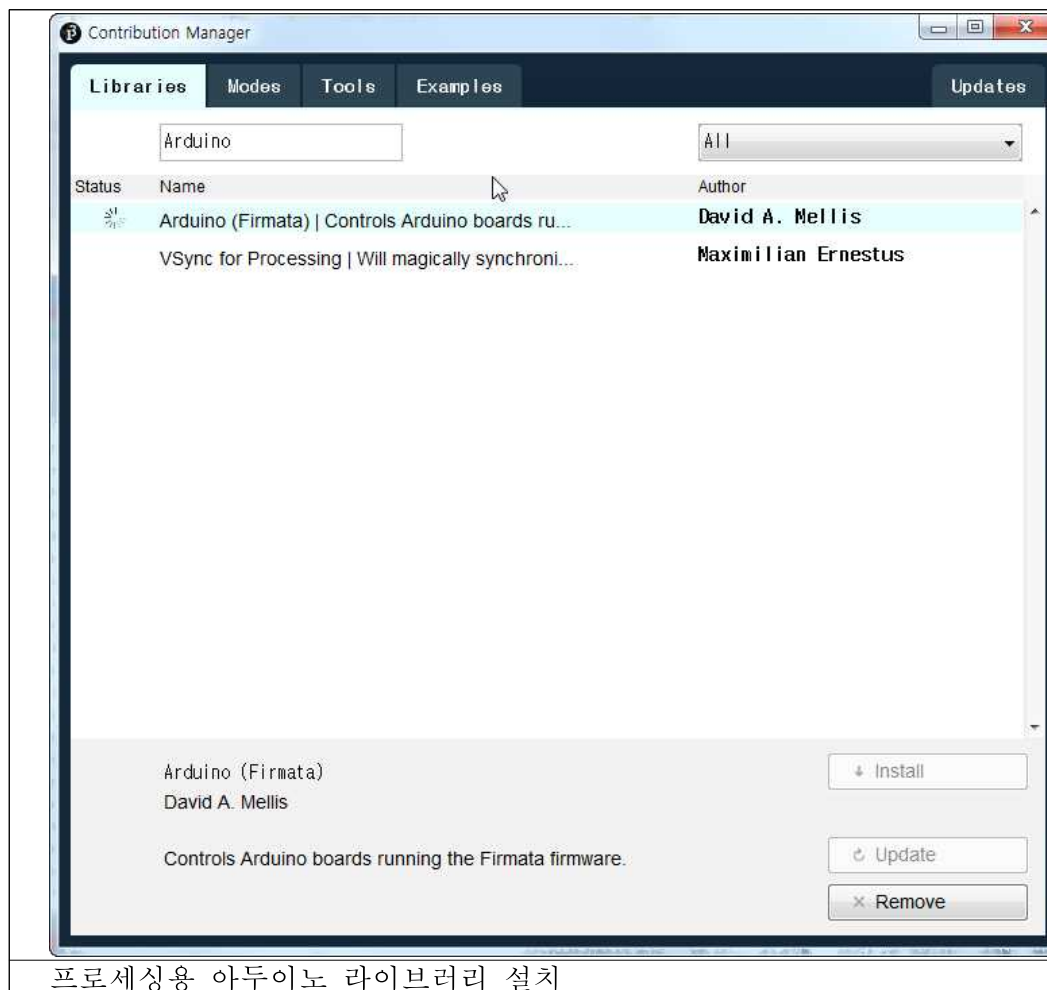


아두이노가 기본적으로 제공하는 예제 중에서 StandardFirmata를 아두이노 보드에 업로드 한다. 물론 이렇게 작업을 하게 되면 아두이노에 올라간 코드에 따라 제한적인 작업을 할 수 밖에 없다는 단점이 있다. 그러나 이 프로토콜을 지원하는 언어, 예를 들어 프로세싱, 파이썬, C# 그 어떤 언어로도 아두이노를 제어할 수 있게 되는 장점도 존재한다. 아두이노에 대한 처리를 호스트에서 실행되는 언어로 국한시킴으로써 아두이노를 블랙박스처럼 생각할 수 있게 된다. 물론 이를 직접 구현할 수 도 있다. 이럴 경우에는 서로 통신을 수행할 수 있도록 모든 프로그램을 작성해주어야 한다. 본 프로젝트는 프로세싱이라는 언어의 GUI 적인 특징을 이용한다는 것에 주목한다.

마. 프로세싱 설치 및 라이브러리 추가

프로세싱 언어는 그래픽이 주요한 프로그램을 보다 쉽게 작성할 수 있도록 해주는 언어이며, 자바를 근간으로 만들어진 언어이다. 프로세싱을 설치하고 추가적인 라이브러리를 설치하자.

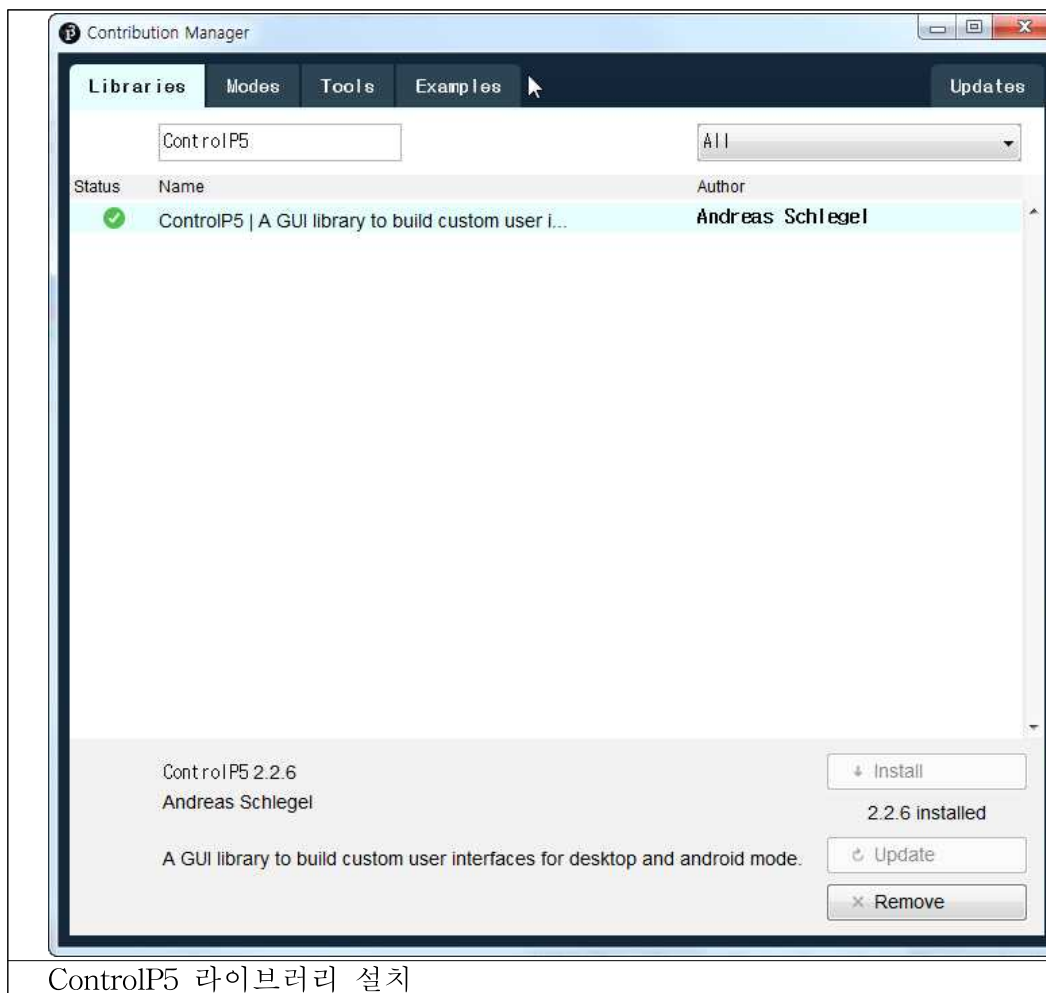
1) 프로세싱용 아두이노 라이브러리 설치



라이브러리의 설치 는 문서 폴더에 이루어지며 이를 통해 해당 라이브러리를 사용할 수 있다.

2) GUI 라이브러리 추가

프로세싱은 GUI를 직접 구성하기 위해서는 좌표를 계산하고 복잡한 도형들을 직접 그려주어야 한다. 이러한 작업을 보다 편리하게 수행할 수 있도록 ControlP5라는 라이브러리를 이용해 보자. 이를 통해 보다 다양한 GUI를 구성할 수 있게 된다.



3) 프로세싱 프로그래밍

```
//color picker
import javax.swing.JColorChooser;
import java.awt.Color;
import processing.serial.*;
import cc.arduino.*;
import controlP5.*;
Arduino arduino;

ControlP5 cp5;
int myColorBackground = color(0,0,0);
int knobValue = 100;
Knob knobR, knobG, knobB;
Knob knobG;
Knob knobB;
int r, g, b;
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

//color picker
Color selectColor;
boolean isFirst = true;
void setup() {
    size(700, 600);
    println(Arduino.list());
    arduino = new Arduino(this, Arduino.list()[1], 57600);
    smooth();
    noStroke();
    cp5 = new ControlP5(this);
```

```

knobR = cp5.addKnob("knobR")
    .setRange(0,255)
    .setValue(0)
    .setPosition(100,50)
    .setRadius(50)
    .setNumberOfTickMarks(10)
    .setTickMarkLength(4)
    .snapToTickMarks(true)
    .setColorForeground(color(255))
    .setColorBackground(color(255, 0, 0))
    .setColorActive(color(255,255,0))
    .setDragDirection(Knob.HORIZONTAL)
;

knobG = cp5.addKnob("knobG")
    .setRange(0,255)
    .setValue(0)
    .setPosition(100,200)
    .setRadius(50)
    .setNumberOfTickMarks(10)
    .setTickMarkLength(4)
    .snapToTickMarks(true)
    .setColorForeground(color(255))
    .setColorBackground(color(0, 255, 0))
    .setColorActive(color(255,255,0))
    .setDragDirection(Knob.HORIZONTAL)
;

knobB = cp5.addKnob("knobB")
    .setRange(0,255)
    .setValue(0)
    .setPosition(100,350)
    .setRadius(50)
    .setNumberOfTickMarks(10)
    .setTickMarkLength(4)
    .snapToTickMarks(true)
    .setColorForeground(color(255))
    .setColorBackground(color(0, 0, 255))
    .setColorActive(color(255,255,0))
    .setDragDirection(Knob.HORIZONTAL)
;

```



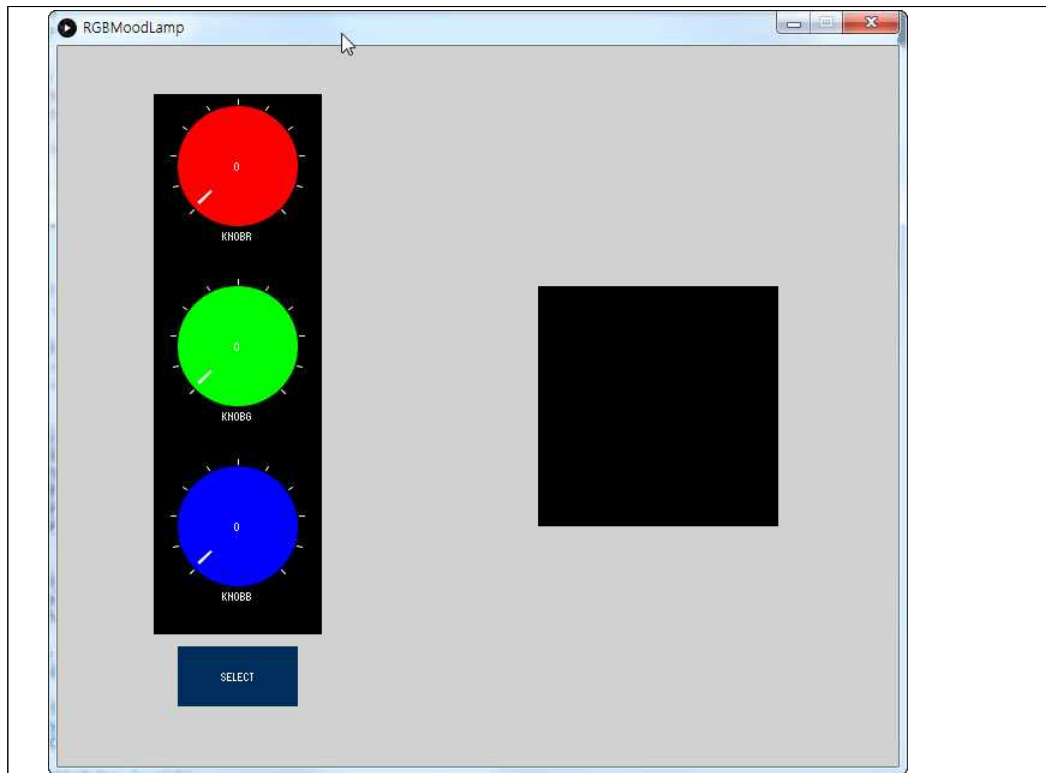
```

cp5.addButton("select")
    .setValue(0)
    .setPosition(100, 500)
    .setSize(100, 50);
}
void draw() {
    fill(0);
    rect(80, 40, 140, 450);
    fill(0);
    rect( 400, 200 , 200, 200);
    fill(r, g, b);
    rect( 450, 250 , 100, 100);
    setColor();
}
void knobR(int theValue) {
    r = theValue;
}void knobG(int theValue) {
    g = theValue;
}void knobB(int theValue) {
    b = theValue;
}public void select(int theValue) {
    if (isFirst == false) {
        selectColor = JColorChooser.showDialog(null,"Java Color
Chooser",Color.white);
        if(selectColor!=null) {
            r = selectColor.getRed();
            g = selectColor.getGreen();
            b = selectColor.getBlue();
            knobR.setValue(r);      knobG.setValue(g);      knobB.setValue(b);

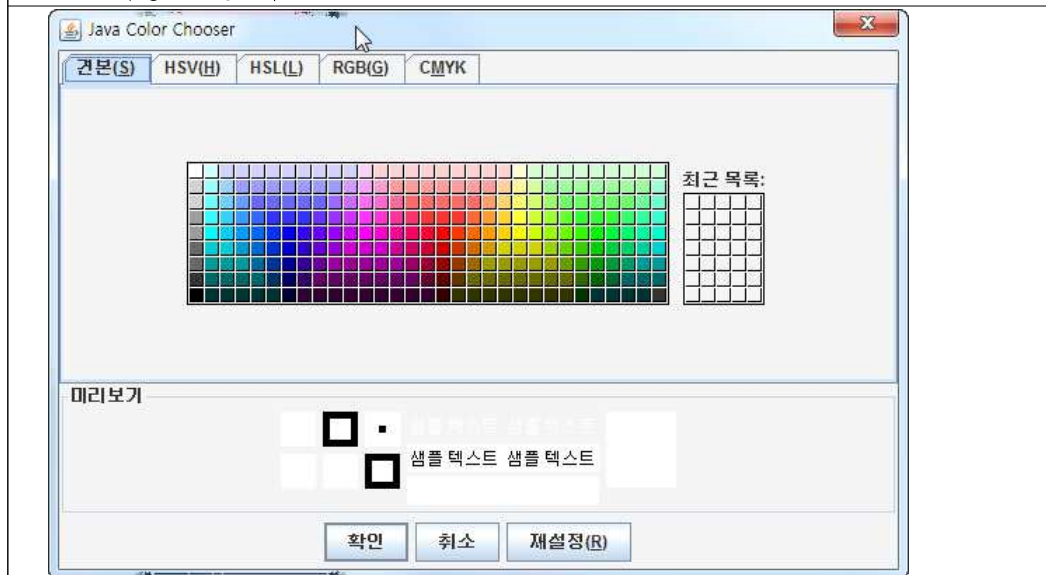
        }
    }
    isFirst = false;
}void setColor() {
    arduino.analogWrite(redPin, r);
    arduino.analogWrite(greenPin, g);
    arduino.analogWrite(bluePin, b);
}

```

4) 프로세싱 실행화면



프로세싱 실행 화면

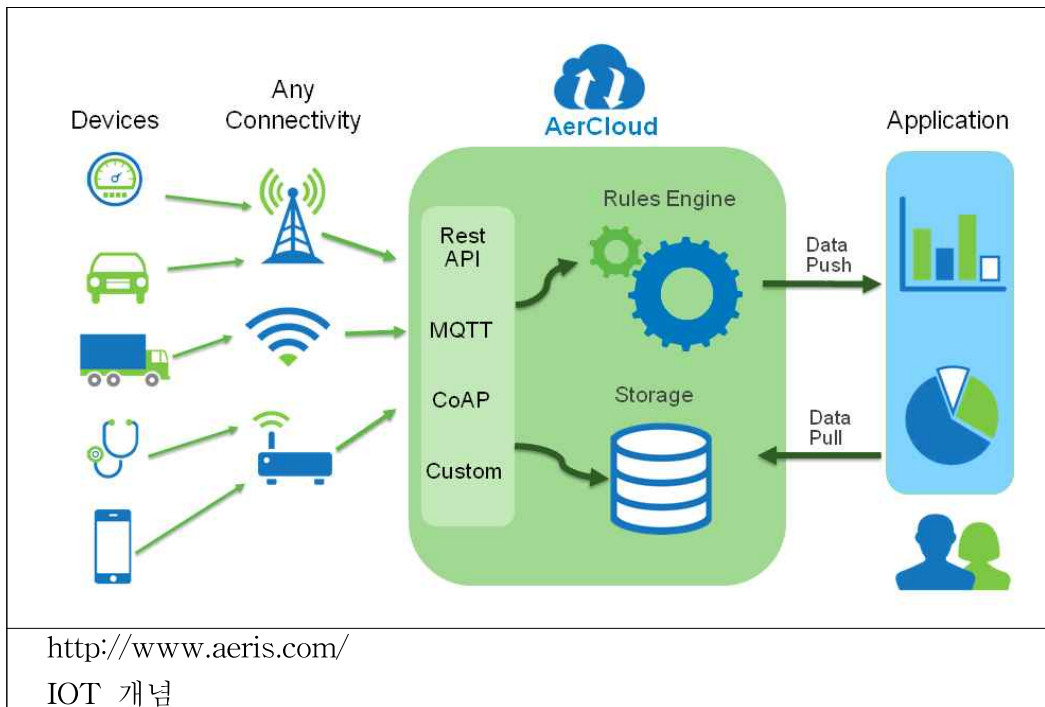


컬러 선택 창

2. 데이터 기록 장치 만들기

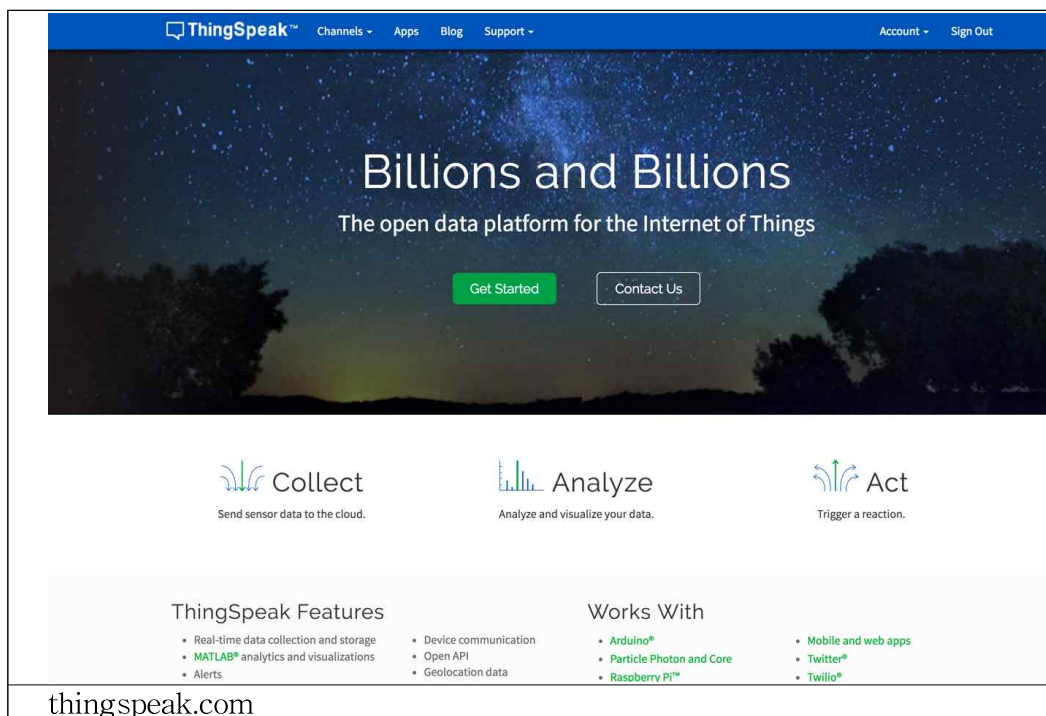
최근 들어 각광을 받고 있는 사물 인터넷(IOT: Internet of Things)은 각종 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술을 의미한다. 단순히 데이터를 인터넷에 연결하여 보내는 것에서 끝나는 것이 아니라 해당 데이터를 통해 판단을 내리고 인간의 생활에 보다 편리한 도움을 줄 수 있는 자료가 피드백이 되는 것이 바로 IOT라고 할 수 있다. 아두이노에 연결된 다양한 센서의 자료는 SD 카드 혹은 이더넷 모듈, 와이파이 모듈등을 통해 인터넷에 업로드가 될 수 있으며 다수의 IOT 플랫폼들은 아두이노와 통신 모듈에서 사용할 수 있는 라이브러리와 무료로 사용할 수 있는 데이터 저장 공간을 제공한다. 이러한 서비스를 IOT Cloud 플랫폼이라고 부를 수 있다.

본 프로젝트에서는 인터넷에 연결하는 도구로서 이더넷 실드나 와이파이 실드를 사용하지 않는다. 대신 프로세싱을 이용하여 아두이노가 보내온 데이터를 프로세싱이 직접 인터넷에 올리는 도구로서 사용을 한다. 이렇게 할 때의 장단점은 존재할 수 있으나, 통신 모듈이 없을 때에도 호스트쪽의 PC를 사용하여 작업을 할 수 있게 된다.



가. IOT Cloud 플랫폼 선택

다양한 IOT Cloud 플랫폼이 존재하지만 그들의 성격은 상당히 유사하다. 사용자가 원하는 데이터를 올릴 수 있는 공간을 제공하며, 해당 데이터에 접근하기 위한 API를 제공하는 형태가 그것이다. 본 프로젝트에서는 thingspeak.com을 이용하여 간단하게 아두이노에서 취득한 데이터를 업로딩하고 이를 모니터링 하는 작업을 수행한다.



2) 회원 가입 및 채널 개설

Sign up to start using ThingSpeak

User ID

Email

Time Zone

Password

Password Confirmation

☐ By signing up, you agree to the [Terms of Use](#) and [Privacy Policy](#).

[Create Account](#)

New Channel

Name

Description

Field 1 ☒

Field 2 ☐

Field 3 ☐

Field 4 ☐

Field 5 ☐

Field 6 ☐

Field 7 ☐

Field 8 ☐

Metadata

Tags

(Tags are comma separated)

Make Public ☐

URL

Help

ThingSpeak Channel

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- Channel Name: Enter a unique name for the ThingSpeak channel.
- Description: Enter a description of the ThingSpeak channel.
- Fields: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata: Enter information about channel data, including JSON, XML, or CSV data.
- Tags: Enter keywords that identify the channel. Separate tags with commas.
- Latitude: Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
- Longitude: Specify the position of the sensor or thing that collects data in decimal degrees. For example, the longitude of the city of London is -0.1275.
- Elevation: Specify the position of the sensor or thing that collects data in meters. For example, the elevation of the city of London is 35.092.
- Make Public: If you want to make the channel publicly available, check this box.
- URL: If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Video ID: If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingSpeak channel. You can then visualize data and transform it using [ThingSpeak Apps](#).

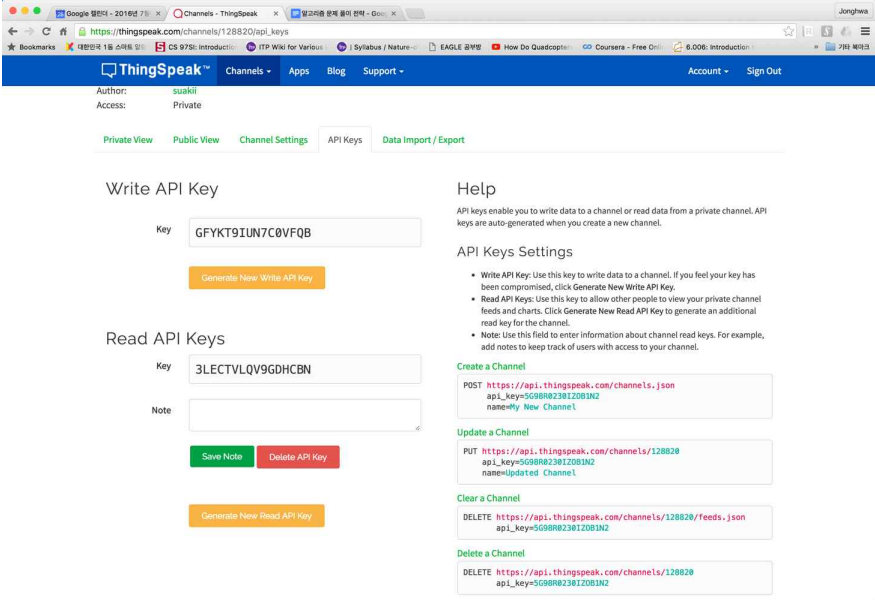
See Tutorial: [ThingSpeak and MATLAB](#) for an example of measuring flow rate from a

채널 정보 설정

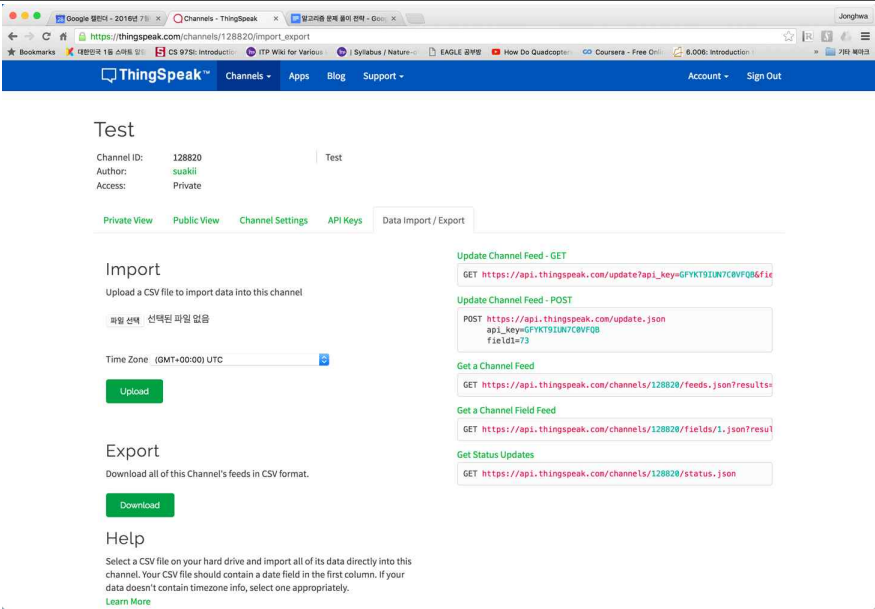
본 프로젝트에서는 일단 1개의 정보만 업로드 하는 작업을 진행한다. 보다 다양한 데이터를 추가하여 업로드 하는 것도 물론 가능하다.

3) 데이터 업로딩을 위한 정보

데이터의 업로드를 위해서는 해당 사이트가 제공하는 API의 규칙을 준수하여야 한다.



API 키



업로딩 방식

나. 하드웨어 연결하기

추가적인 하드웨어는 필요 없으며 Firmata를 아두이노에 올려둔다.

다. 프로세싱 프로그래밍

```
import processing.net.*;
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
String SERVER = "api.thingspeak.com"; // MyRobots or ThingSpeak
String APIKEY = "6U5Y63LAWIF956EE"; // your channel's API key
Client c;

int savedTime;
int totalTime = 5000;
String[] fields = { "field1" };
float[] datum = new float[1];

void setup() {
    size(200, 200);
    frameRate(60);
    println(Serial.list());
    arduino = new Arduino(this, Arduino.list()[1], 57600);
}

int i = 100; int time5 = 0;

void draw() {
    if (c != null) {
        if (c.available() > 0)
            //println(c.readString());
    }

    int passedTime = millis() - savedTime;
    if (passedTime > totalTime) {
        time5 += 5;
        sendDatum(fields, datum);
        savedTime = millis();
        datum[0] = arduino.analogRead(0);
        background(255); fill(0);
        text(time5 + " seconds have passed!", 10, 50);
        text("send data " + datum[0], 10, 80);
    }
}
```

```

void sendDatum(String fields[], float num[]) {

    String url = ("GET /update?api_key="+APIKEY);
    StringBuffer sb = new StringBuffer(url);

    for(int i=0; i<fields.length; i++) {
        String s = ("&"+fields[i]+"="+ num[i]);
        sb.append(s);
    }

    sb.append(" HTTP/1.1\n");

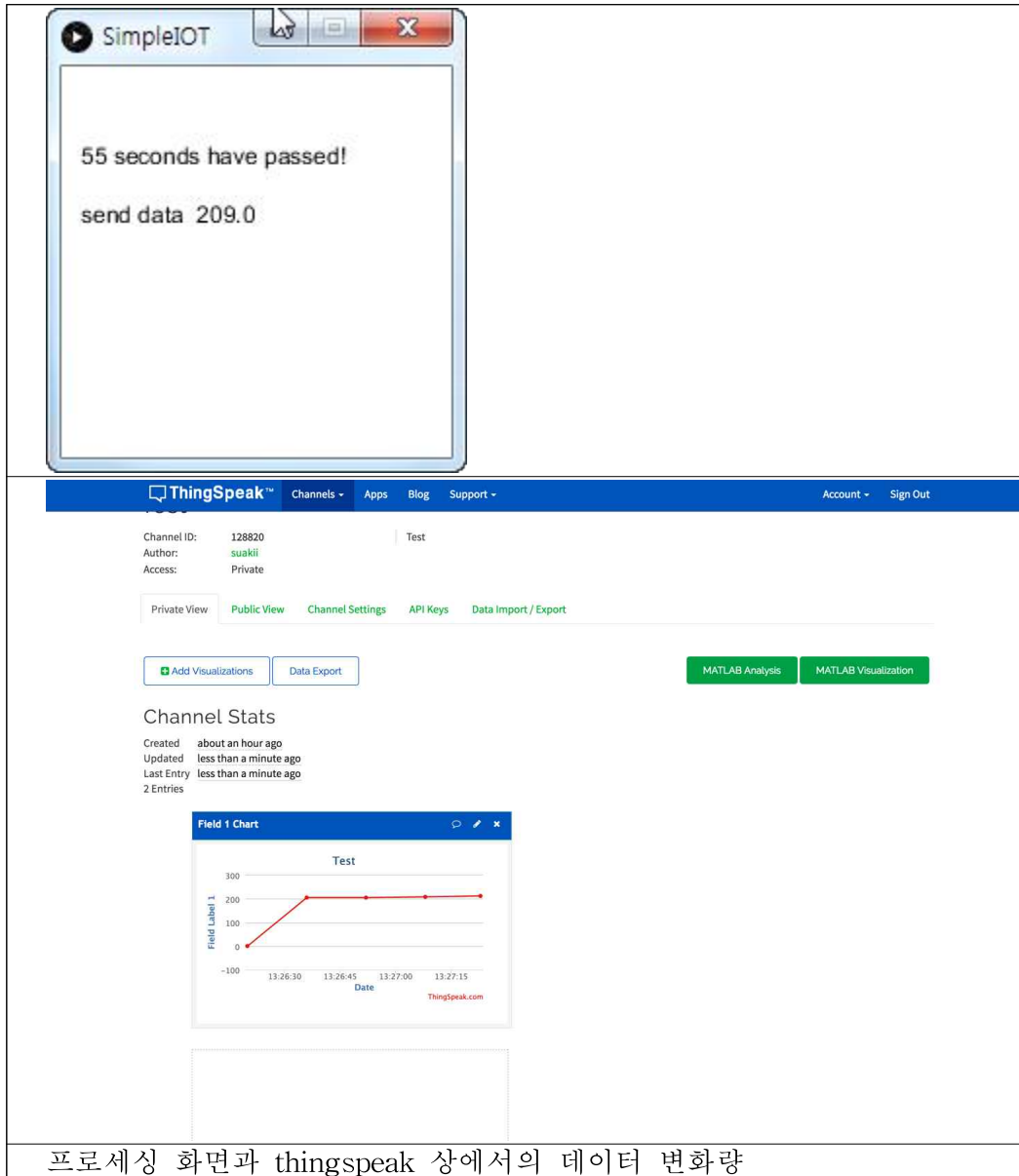
    String finalurl = sb.toString();
    print("sending data: " + finalurl);

    c = new Client(this, SERVER, 80);

    if (c != null) {
        c.write(finalurl);
        c.write("Host: suakii.com\n\n");
    }
}

```


라. 결과화면

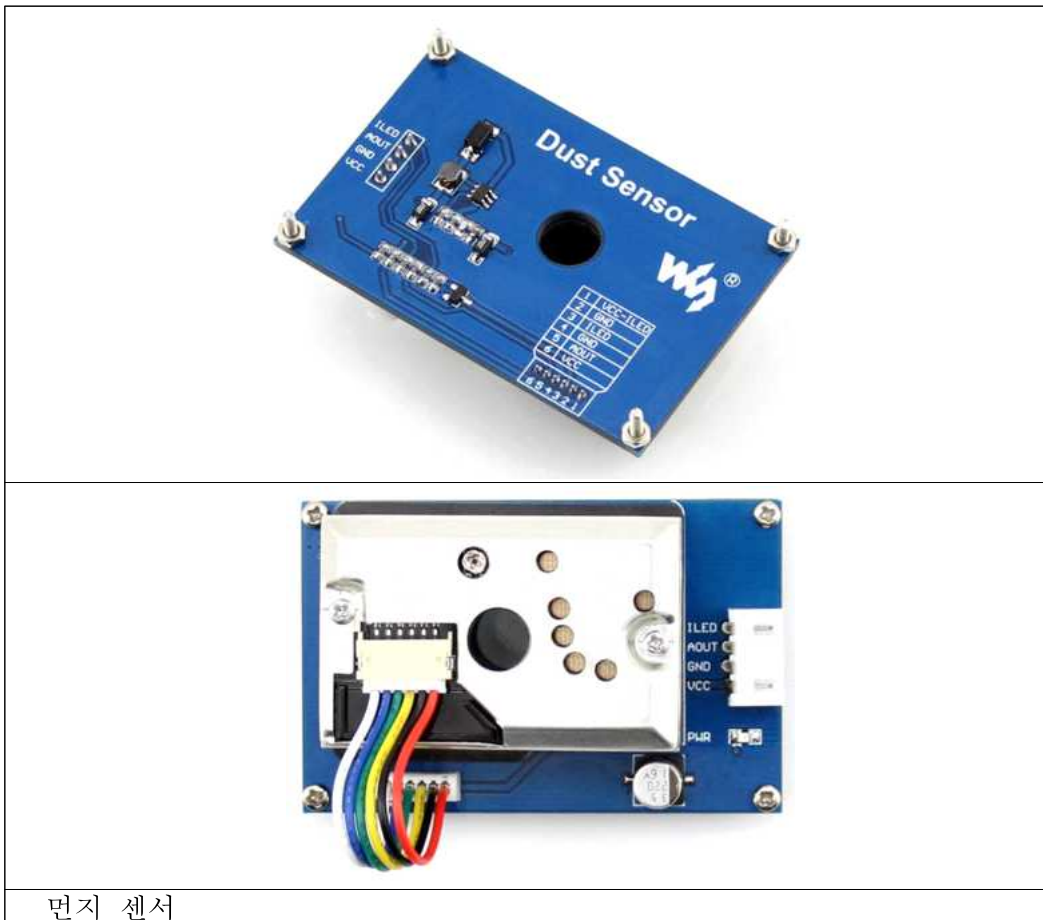


실제 프로세싱에서 업로드 하는 시간과 Thingspek 상에서 표시되는 데이터의 시간은 약간의 지연이 발생하게 된다. 이로써 자신이 원하는 데이터를 웹에 업로드 하여 지속적인 모니터링이 가능하게 된다. 어떤 데이터를 업로드 하고 무엇을 볼지는 선택의 몫이된다.

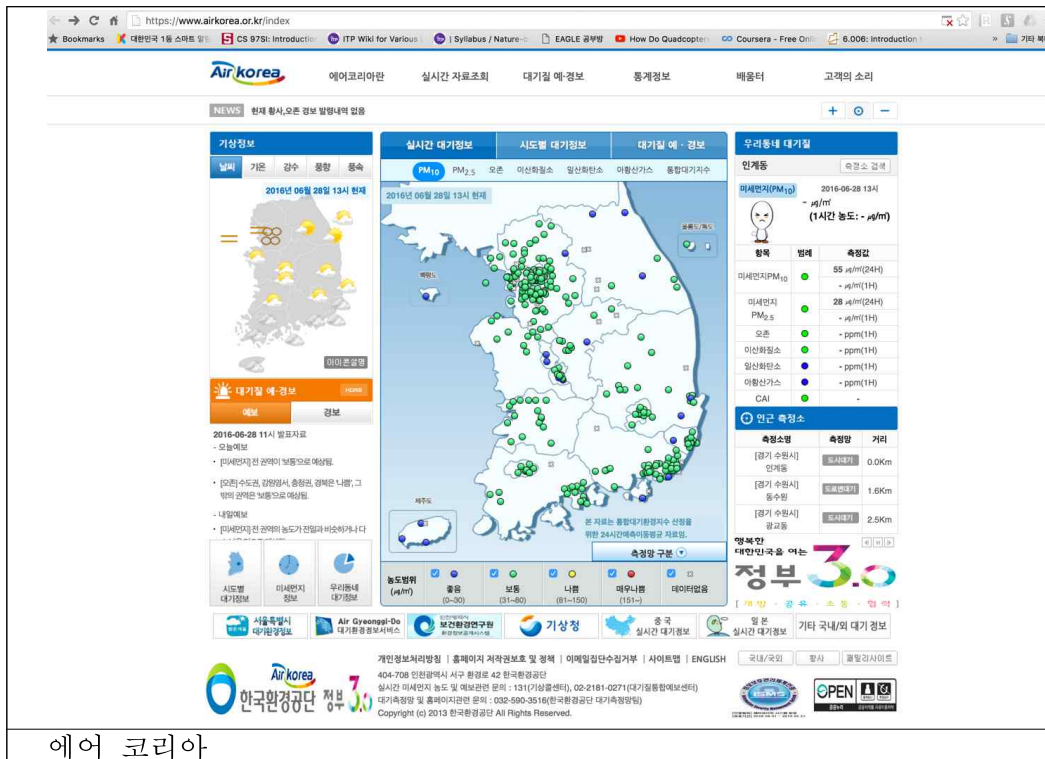
3. 미세먼지를 측정해 보자~

미세먼지는 지름이 $10\mu\text{m}$ (마이크로미터, $1\mu\text{m}=1000$ 분의 1mm) 이하의 먼지로 PM(Particulate Matter)10이라고 한다. 자동차 배출가스나 공장 굴뚝 등을 통해 주로 배출되며 중국의 황사나 심한 스모그때 날아오는 크기가 작은 먼지를 말한다. 미세먼지중 입자의 크기가 더 작은 미세먼지를 초미세먼지라 부르며 지름 $2.5\mu\text{m}$ 이하의 먼지로서 PM2.5라고 한다. 주로 자동차 배출가스 등을 통해 직접 배출된다. (네이버 캐스트)

아두이노와 샤프사의 먼지 센서인 “GP2Y1010AU0F” 를 이용하면 간단하게 먼지의 양을 측정할 수 있다.

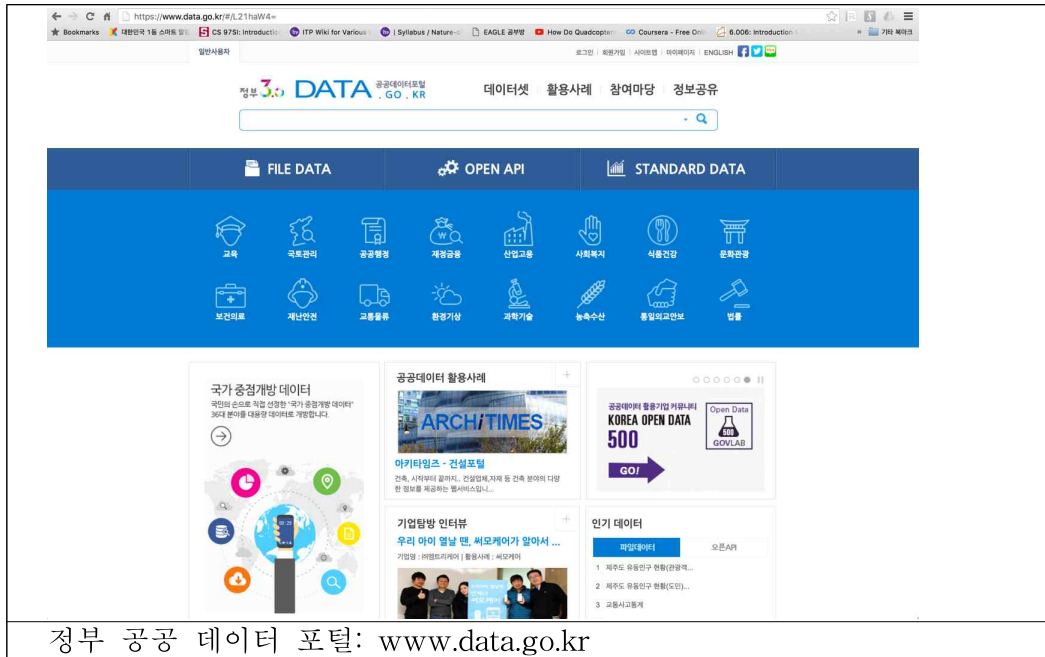


그러나 이는 정확한 측정의 어려움과 지속적인 측정의 어려움이 따른다. 그리고 이미 우리 주위에는 미세먼지와 각종 대기질의 측정 정보를 알려주고 있는 곳이 많이 있다.



또한 정부에서 제공하고 있는 공공데이터 포털을 이용한다면 무료로 제공되는 다양한 자료들을 얻을 수 있다. 이번 프로젝트는 프로세싱을 이용하여 정보 공공 데이터 포털에서 제공하는 대기질 측정 자료를 xml로 가지고 와서 이를 파싱한다. 이 작업은 프로세싱이 수행하며, 파싱된 자료를 통해 아두이노의 서브모터에 신호를 보내어 현재 대기질의 좋고 나쁨을 표시해주는 작업이다.

가. 공공데이터 포털 회원 가입 및 인증 키 받기



정부 공공 데이터 포털: www.data.go.kr



정부 공공 데이터 포털: 미세먼지 조회



정부 공공 데이터 포털: www.data.go.kr

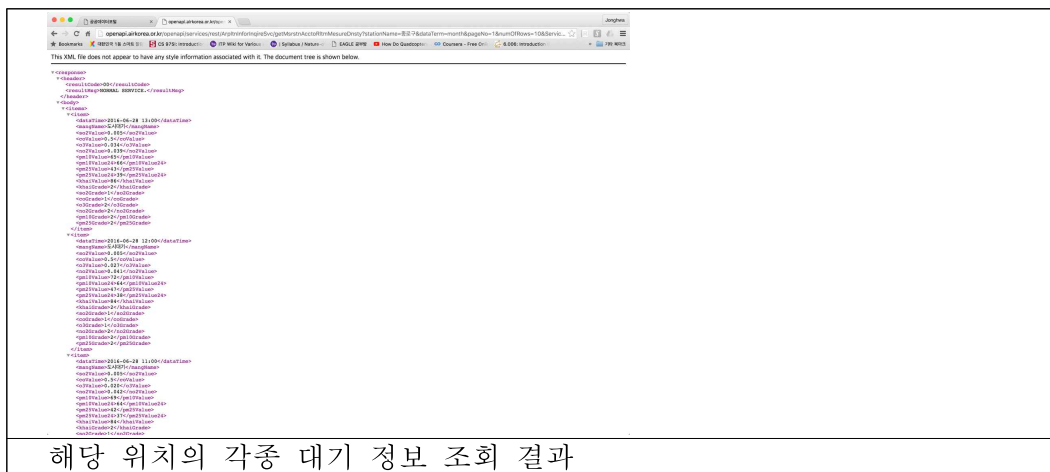


정부 공공 데이터 포털: 대기오염정보 조회 서비스 신청



나. 인증키 테스트

<http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInquireSvc/getMrstnAcctoRltmMeasureDnсты?stationName=종로구&dataTerm=month&pageNo=1&numOfRows=10&ServiceKey=8n%2BI3gJn6o2pN%2BwLFHj6i9anqDFAjpD97TWoC%2FeV4zynQPsgQ3TCxhgua6DYylpMGg%2Fn1%2BH7rX%2Fya6uDkcltA%3D%3D&ver=1.2>



다. 관심 지역

본 프로젝트에서는 관심지역을 “영통동”으로 조회를 한다. 조회 결과는 다음과 같은 10개의 항목이 포함된 xml로 나타난다.

```
<response>
<header>
<resultCode>00</resultCode>
<resultMsg>NORMAL SERVICE.</resultMsg>
</header>
<body>
<items>
<item>
<dateTime>2016-06-28 13:00</dateTime>
<so2Value>-</so2Value>
<coValue>-</coValue>
<o3Value>-</o3Value>
<no2Value>-</no2Value>
<pm10Value>-</pm10Value>
<khaiValue>-</khaiValue>
<khaiGrade/>
<so2Grade/>
<coGrade/>
<o3Grade/>
<no2Grade/>
<pm10Grade/>
</item>
.....
<numOfRows>10</numOfRows>
<pageNo>1</pageNo>
<totalCount>742</totalCount>
</body>
</response>
```

영통동의 조회 결과

미세먼지는 pm10Value 항목이다.

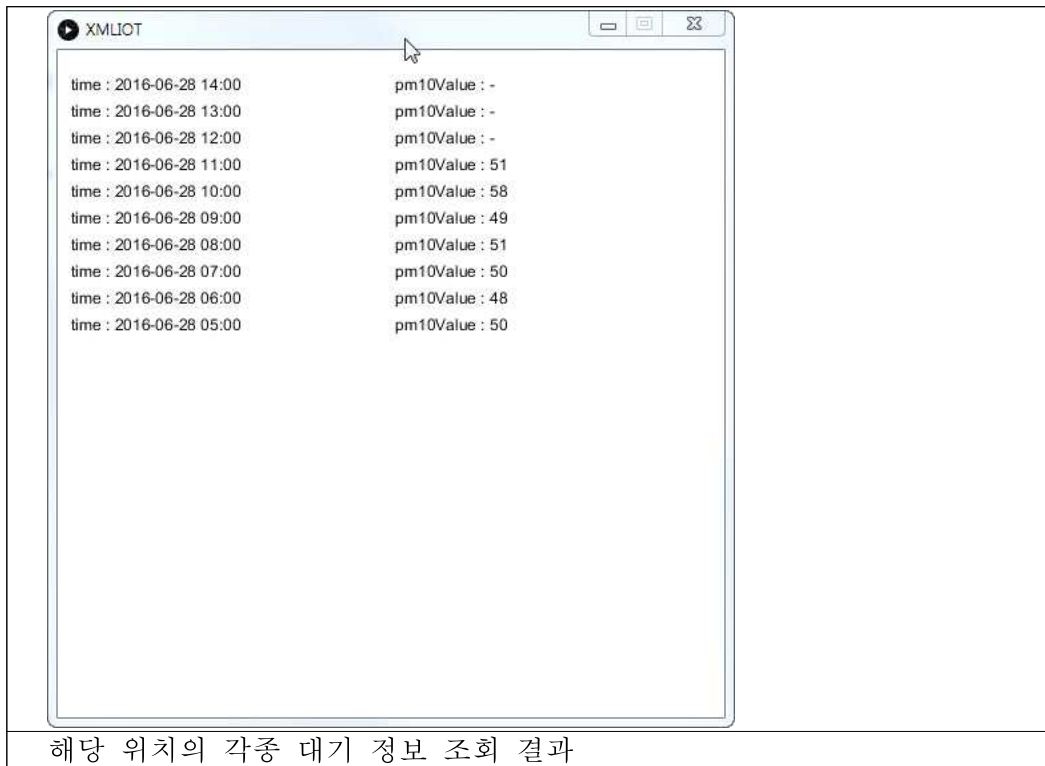
측정된 시간에서 최근 10개의 항목을 보여주고 있으며 미세먼지의 표현은 pm10Value 이며 단위는 $\mu\text{g}/\text{m}^3$ 이다.

라. 프로세싱 프로그램

아두이노를 사용하지 않고 현재 관심 지역의 미세먼지 농도를 표시하는 프로그램을 작성해 본다.

```
XML res;
String sunrise, sunset;
//영통동
String
source="http://openapi.airkorea.or.kr/openapi/services/rest/ArpltnInforInquireSvc/getMsrstnAcctoRltmMesureDnsty?stationName=%EC%98%81%ED%86%B5%EB%8F%99&dataTerm=month&pageNo=1&numOfRows=10&ServiceKey=";
String
apiKey="8n%2B13gJJn6o2pN%2Bw1FHj6i9anqdFAjpd97TWoC%2FeV4zynQPsgQ3TCxhgua6DYylpMGg%2Fn1%2BH7rX%2Fya6uDkcItA%3D%3D";
void setup() {
    size(500,500);
    background(255);
    fill(0);
    res = loadXML(source+apiKey);
    XML[] children = res.getChildren("body");
    XML res2 = children[0];
    XML[] answer =
    res.getChildren("body")[0].getChildren("items")[0].getChildren("item");
    int value = -10;
    boolean done = false;
    for (int i = 0; i < answer.length; i++) {
        XML time = answer[i].getChild("dateTime");
        XML pm10Value = answer[i].getChild("pm10Value");
        text("time : " + time.getContent(), 10, i*20+30);
        text(" pm10Value : " + pm10Value.getContent(), 250, i*20+30);
        int temp = int(pm10Value.getContent());
        if (temp > 0 && done == false) {
            value = temp;
            done = true;
        }
    }
    println("Value = " + value);
}
```


마. 프로세싱 실행 결과



환경부에서 제시하고 있는 미세먼지의 농도별 지표는 다음과 같다.

예보	농도
좋은	0~30 $\mu\text{g}/\text{m}^3$
보통	31~80 $\mu\text{g}/\text{m}^3$
약간 나쁨	81~120 $\mu\text{g}/\text{m}^3$
나쁨	121~200 $\mu\text{g}/\text{m}^3$
매우 나쁨	201~300 $\mu\text{g}/\text{m}^3$ 이상

프로세싱에서 취득한 결과를 토대로 아두이노의 서보 모터에 값을 출력하여 농도별 위험 수치를 알 수 있도록 하자.

마. 프로세싱 서보 모터 컨트롤

지금까지의 모든 작업은 아두이노에 추가적인 프로그램을 하지 않은 것이다. 모든 프로그램은 프로세싱에서 이루어지고 있다. 위에서 받은 미세먼지 데이터를 가지고 아두이노에 서보모터를 연결하여 움직여 본다. 서보 모터는 앞에서 언급하였기에 추가적인 설명은 하지 않는다.

```
//추가 되는 아두이노 제어 코드
import processing.serial.*;

import cc.arduino.*;

Arduino arduino;
...
...
void setup() {
  println(Arduino.list());

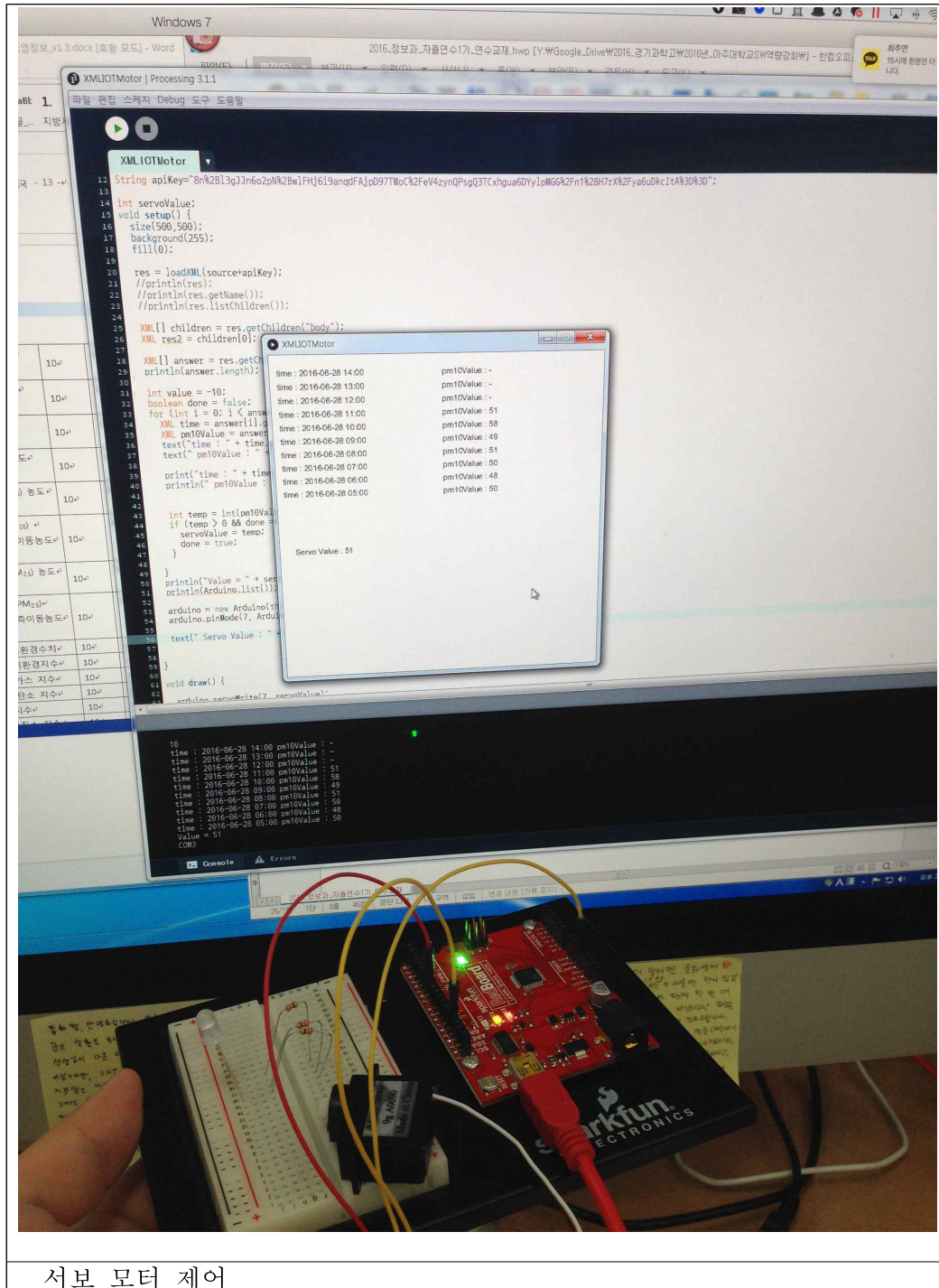
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(7, Arduino.SERVO);

  text(" Servo Value : " + servoValue, 20, 300);
}
void draw() {

  arduino.servoWrite(7, servoValue);

}
```

바. 실행 결과



서보 모터 제어