

OLED 디스플레이와 온도·기압센서

OLED 디스플레이란

- OLED는 유기발광다이오드 또는 유기EL이라 한다. 낮은 전압에서 구동이 가능하고 넓은 시야각과 빠른 응답속도를 갖고 있어 일반 LCD와 달리 바로 옆에서 보아도 화질이 변하지 않으며 화면에 잔상이 남지 않는다

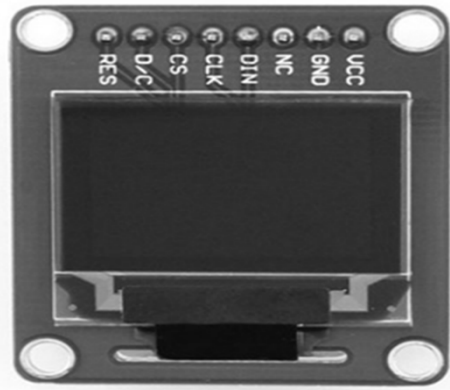
	LCD	OLED
응답속도	30/1,000sec	1/100,000sec
시야각	제한 있음	제한 없음
작동온도	0°C ~ 65°C	-30°C ~ 80°C
발광방식	Back Light Unit 필요	자체 발광형
수명	약 60,000시간 이상	약 25,000시간
명암비	LCD < OLED	
크기	다양	작음
유연성	낮음	높음

LCD와 OLED 디스플레이 비교

OLED 디스플레이 모듈 | SH1106

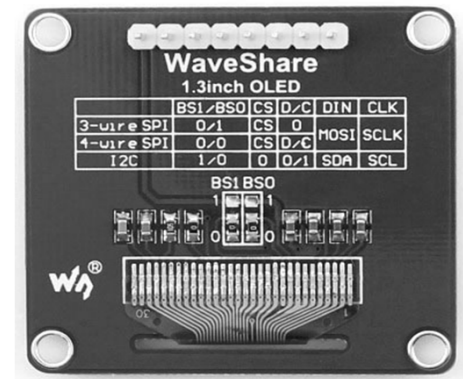
Color	Mono(Blue)
Pixels	128X64
Size	29.42X14.7(mm), 1.3inch
Interface	SPI, I2C

SH1106의 사양



Interface	BS1	BS0
3-wire SPI	0	1
4-wire SPI	0	0
I2C	1	0

SH1106의 통신방법 설정

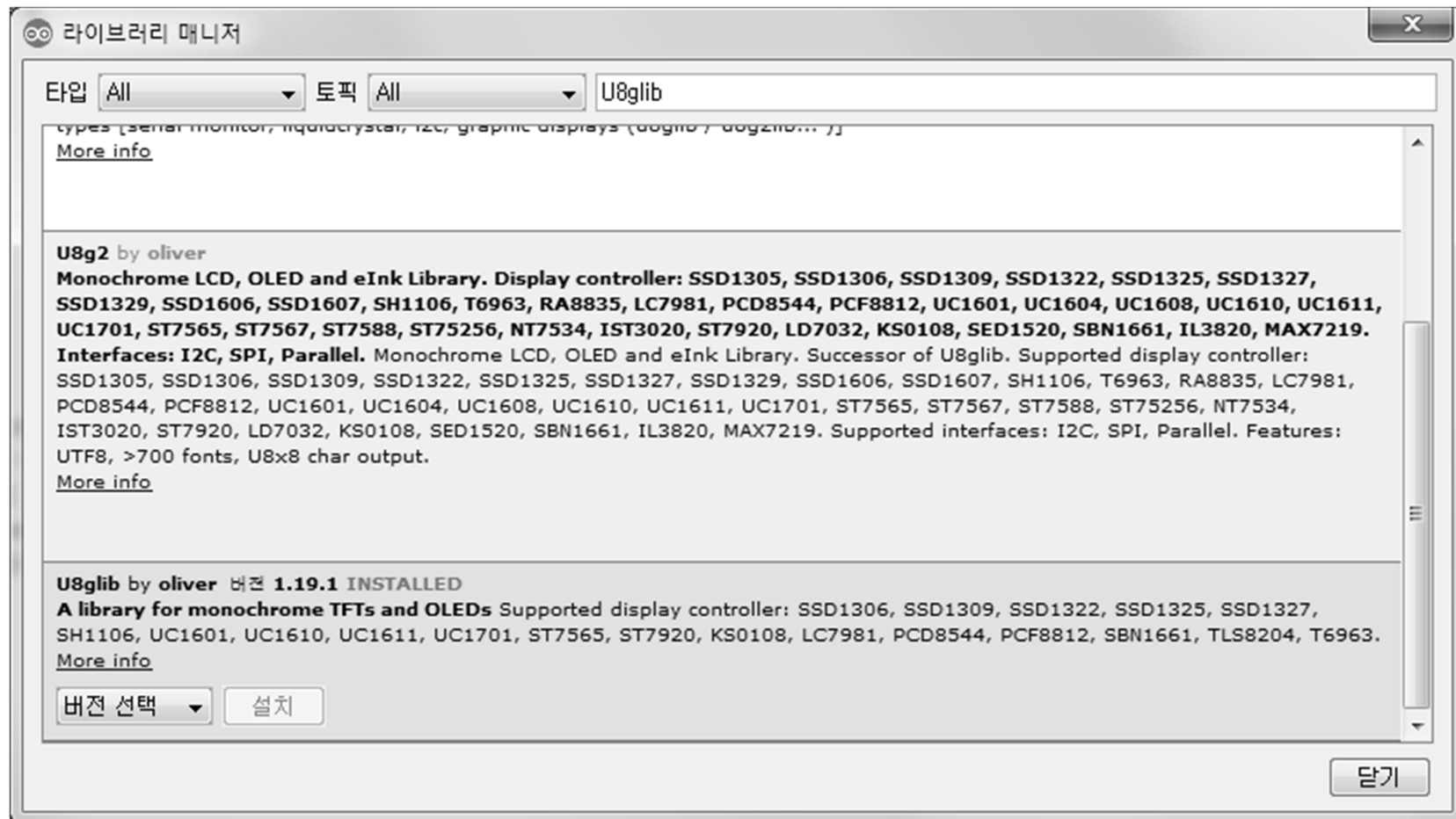


핀맵 2-1 SH1106의 연결 PIN MAP

SH1106	VCC	GND	NC	DIN	CLK	CS	D/C	RES
아두이노 우노	VCC	GND		D9	D10	D12	D11	D13
라두이노								

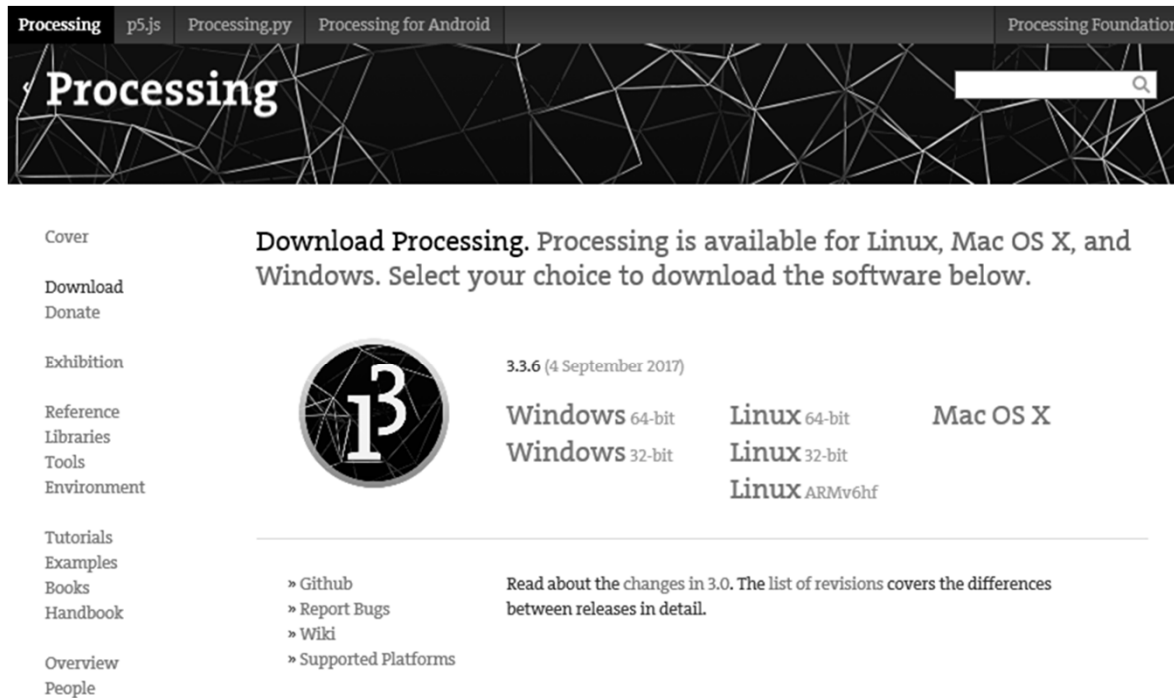
U8G 라이브러리 설치

<라이브러리 관리자에서 U8glib 1.19.1 설치>



프로세싱 설치 및 사용방법

우선 Processing 공식 홈페이지 <https://processing.org> 에 접속하여 설치



The screenshot shows the Processing.org website. The top navigation bar includes links for Processing, p5.js, Processing.py, Processing for Android, and Processing Foundation. The main header features the Processing logo and a search bar. On the left, a sidebar lists various resources: Cover, Download, Donate, Exhibition, Reference, Libraries, Tools, Environment, Tutorials, Examples, Books, Handbook, Overview, and People. The main content area is titled 'Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.' It features a large circular logo with the number '13' and the text '3.3.6 (4 September 2017)'. Below the logo, there are links for Windows 64-bit, Windows 32-bit, Linux 64-bit, Linux 32-bit, Linux ARMv6hf, and Mac OS X. At the bottom, there are links to Github, Report Bugs, Wiki, and Supported Platforms, along with a link to read about the changes in 3.0.

Processing p5.js Processing.py Processing for Android Processing Foundation

Processing

Cover Download Donate Exhibition Reference Libraries Tools Environment Tutorials Examples Books Handbook Overview People

Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.

3.3.6 (4 September 2017)

Windows 64-bit Windows 32-bit Linux 64-bit Linux 32-bit Linux ARMv6hf Mac OS X

» Github » Report Bugs » Wiki » Supported Platforms

Read about the changes in 3.0. The list of revisions covers the differences between releases in detail.

프로세싱 설치 및 사용방법

U8glib Graphical Editor

38 commits1 branch0 releases1 contributor

Branch: masterNew pull requestFind fileClone or download

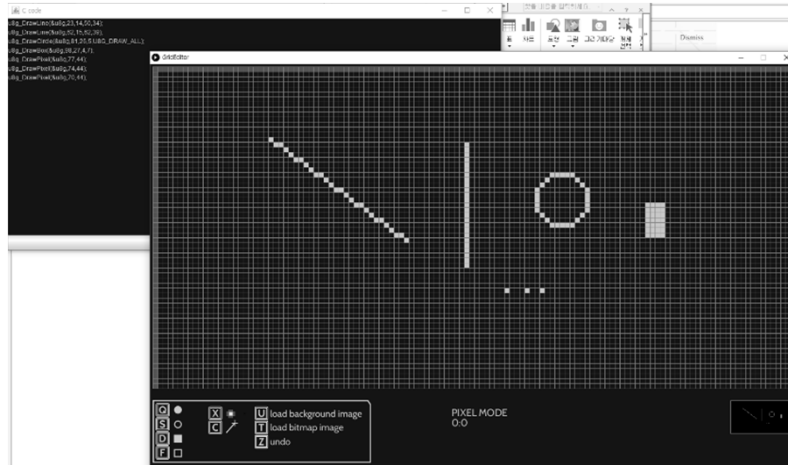
toane added screenshot to README

Latest commit cdb5cbd on Nov 29 2017

GridEditor	embedded free font, new toolbar	7 months ago
README.md	added screenshot to README	4 months ago
screen1.png	added screenshot to README	4 months ago

Github에서 U8glib Graphical Editor를 검색하여 'toane/curiousgrid' 이름의 processing 파일을 다운받으면 GridEditor 폴더에 GridEditor.pde 파일이 존재하는데 이 파일을 프로세싱을 이용하여 실행

프로세싱 설치 및 사용방법



<이미지 그리기 및 이미지 -> 코드변환>

위의 도형들을 그리면 아래와 같은 코드가 변환되어 나오게 되는데 이번 예제에 사용한 코드에 적용하기 위해서는 약간의 수정이 필요하다.

```
u8g_DrawLine(&u8g,62,15,62,39);
```

```
u8g_DrawCircle(&u8g,81,26,5,U8G_DRAW_ALL);
```

```
u8g_DrawBox(&u8g,98,27,4,7);
```

```
u8g_DrawPixel(&u8g,77,44);
```



```
u8g.drawLine(62,15,62,39);
```

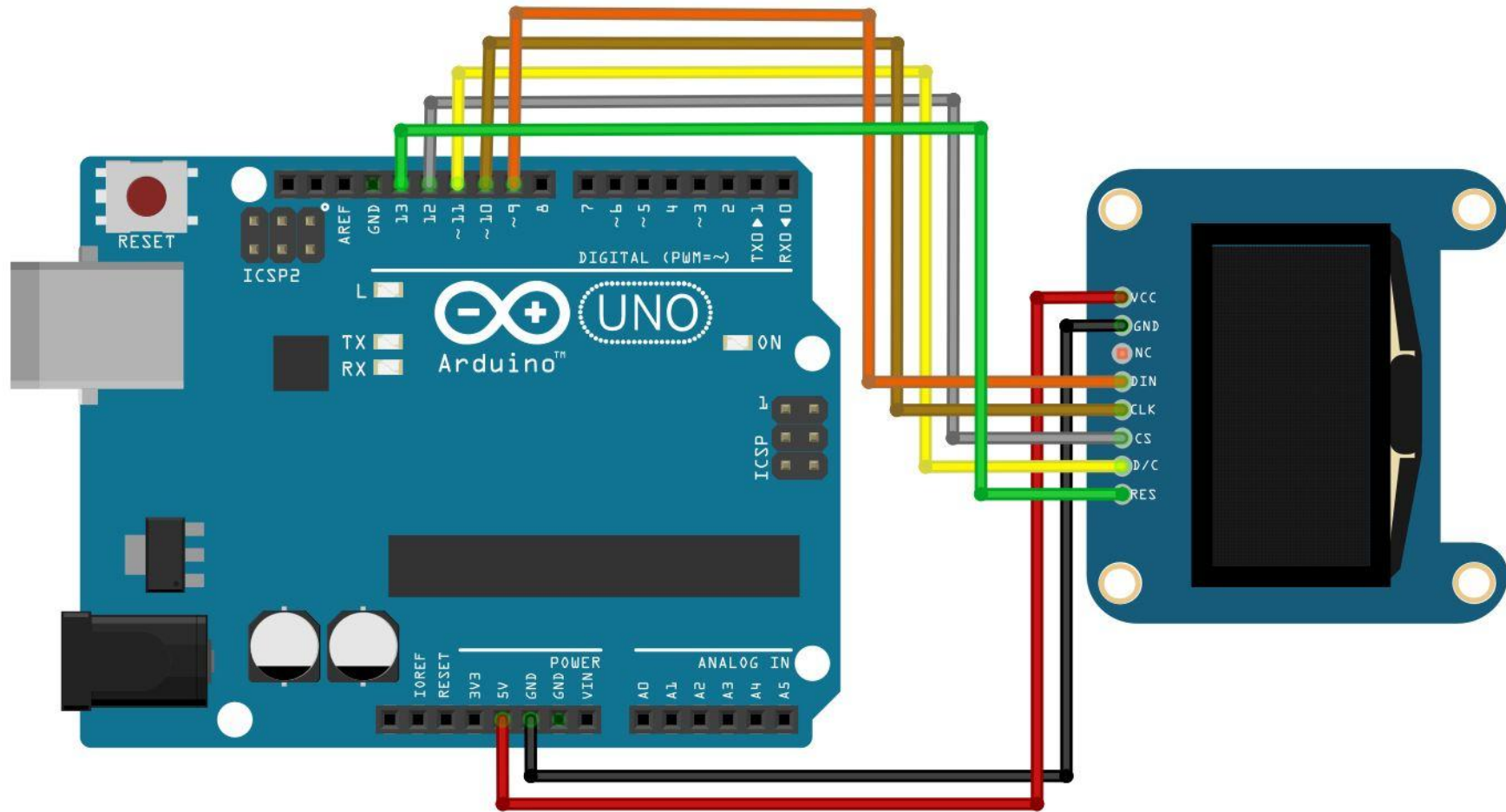
```
u8g.drawCircle(81,26,5);
```

```
u8g.drawBox(98,27,4,7);
```

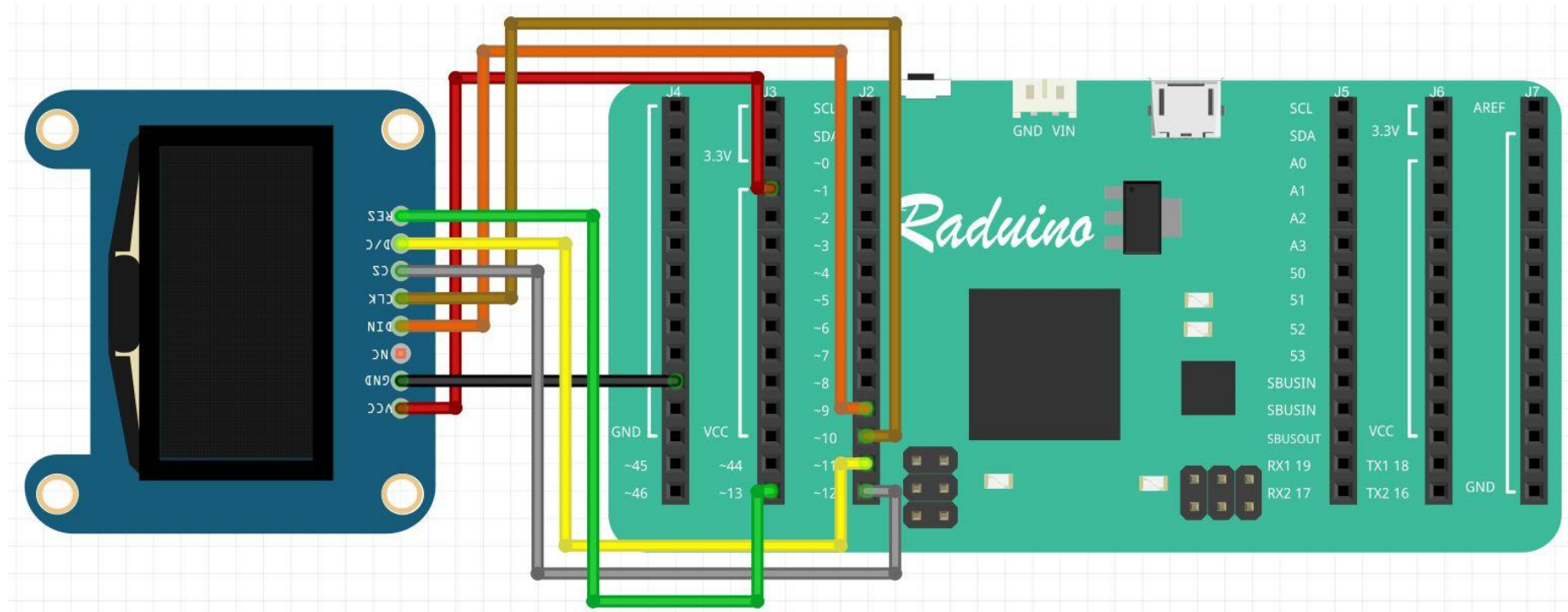
```
u8g.drawPixel(77,44);
```

이렇게 수정된 코드를 아두이노 스케치 코드 **void draw() { }** 함수 내부에 넣어주면 위의 프로그램을 이용해 그렸던 도형들이 OLED 디스플레이에 출력된다.

연결 회로도



연결 회로도



스케치코드 2-4 예제

#include <U8glib.h> SH1106 OLED 디스플레이 제어
위해 라이브러리 추가

//U8GLIB_SH1106_128X64 u8g(SCK(CLK), MOSI(DIN), CS, A0(D/C),RES);
U8GLIB_SH1106_128X64 u8g(10, 9, 12, 11, 13);

int Sec1, Sec2, Min1, Min2 = 0;

unsigned long previousTime, currentTime;

void draw(void) {

사용자 설정에 의해
실질적으로 OLED에
그림을 그리는 함수

u8g.setFont(u8g_font_profont17r);

u8g.drawStr(25, 18, "TIMER ");

u8g.drawLine(5, 3, 120, 3);

u8g.drawLine(5, 22, 120, 22);

u8g.drawLine(5, 3, 5, 60);

u8g.drawLine(120, 3, 120, 60);

u8g.drawLine(5, 60, 120, 60);

u8g.setFont(u8g_font_courB18r);

u8g.setPrintPos(30,50);

u8g.print(Min1);

u8g.setPrintPos(45,50);

초와 분에 대한 변수선언
이전시간과
현재시간에 대한
변수선언

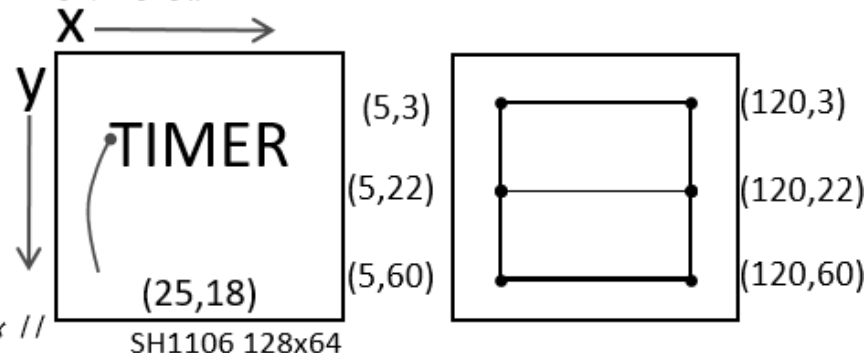
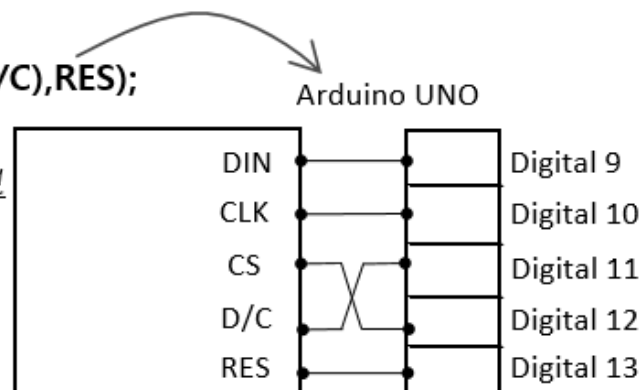
사용할 (Font)폰트설정.

같은 이름의 폰트에서 저용량 폰트를 골랐다

(25, 18)을 시작 좌표로
문자열(string) 출력

좌표1 (x1, y1)과
좌표2 (x2,y2)를
잇는 직선을 그린다.

사용할 폰트(font) 설정.
폰트그룹 fontgroupadobe x //
에 포함된 폰트로,
u8g_font_courB18
(6420byte)보다 작은 용량
(3001byte)을 갖는다.



If(Min1==6) Min1, Min2, Sec1, Sec2=0;

```

u8g.setPrintPos(30,50);
u8g.print(Min1);
u8g.setPrintPos(45,50);
u8g.print(Min2);
u8g.setPrintPos(58,47);
u8g.print(":");
u8g.setPrintPos(70,50);
u8g.setPrintPos(85,50);
u8g.print(Sec2);
}

```

u8g_font_courB18
(6420byte)보다 작은 용량
(3001byte)을 갖는다.

출력할 좌표를
SetPrintPos()로
설정 후 변수 또는
문자열을 출력한다.

```

void setup(void)
  previousTime = millis();
}

```

시계알고리즘에 사용할 이전 시간을 기록한다
Millisecond (1/1000s)단위의 측정시간

```

void loop(void) {

```

시계알고리즘에 사용할 현재시간을 기록한다

```

  currentTime = millis();
  if(currentTime - previousTime > 1000){
    previousTime = currentTime;
    Sec2++;

    if(Sec2 == 10){
      Sec2 = 0;
      Sec1++;
    }
  }

```

```

  if(Sec1 == 6 && Sec2 == 0){
    Sec1 = 0;
    Sec2 = 0;
  }

```

```

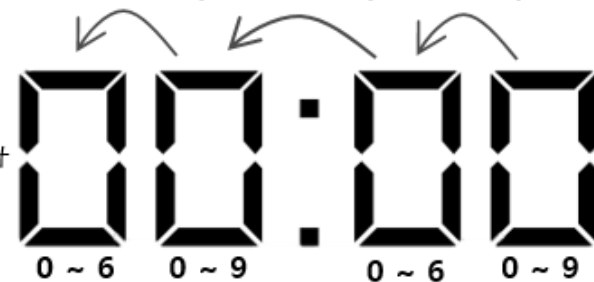
If(Min1==6) Min1, Min2, Sec1, Sec2=0;
If(Min2==10)Min1++; Min2=0;

```

```

If(sec1==6) Min2++; sec1=0; sec2=0;
If(sec2==10) sec1++; sec2=0;

```



시계알고리즘

이전 시간과 현재시간 변수를 비교해 1000ms(1초)를 초과하면
이전 시간을 현재시간으로 갱신하고 초단위의 일의자리 변수(sec2)을
1 증가시킨다.

이때, 1 증가시킨 sec2가 10이라면 0으로 초기화 시켜주고
초 단위 십의자리 변수(sec1)을 1 증가시킨다.

만약, 1증가시킨 sec1이 6이고 sec2가 0이라면 초단위의 변수들을 0으로
초기화하고 분단위의 일의자리 변수(min2)을 1 증가시킨다 (1분=60초)

```
Min2++;  
}
```

```
If(Min2 == 10){  
    Min2 = 0;  
    Min1++;  
}
```

```
If(Min1 == 6 && Min2 == 0){  
    Sec1 = 0;  
    Sec2 = 0;  
    Min1 = 0;  
    Min2 = 0;  
}
```

초 단위와 같은 방식으로 Min2가 10이
되면 Min2을 0으로 초기화 하고
분 단위의 십의 자리 변수(min1)을 1 증가시킨다.
이 때, 증가한 Min1이 6이고 Min2가 0이라면
모든 분과 초 단위의 변수를 초기화 한다.

```
// picture loop OLED
```

```
u8g.firstPage();
```

```
do {
```

```
    draw();
```

```
    } while( u8g.nextPage() );
```

```
}
```

```
}
```

이 프로시저(procedure)를 호출하면 그리기 구간(Picture loop)이 시작된다.
장치를 초기화하고 기본 상태로 설정한 후 디스플레이의 너비, 높이, Mode,
페이지(로컬 프레임 버퍼)등의 정보를 가져온다.

사용자 설정에 의한 그리기를 수행한다.

이 프로시저를 호출하면 그리기 구간 본문의 끝을 나타낸다.

그리기 구간이 끝난 경우, 0을 반환하며, 다시 그리기가 필요한 경우
1을 반환한다.

스케치코드 2-5 예제

```
#include <U8glib.h>
U8GLIB_SH1106_128X64 u8g(10, 9, 12, 11, 13);
//U8GLIB_SH1106_128X64 u8g(SCK(CLK), MOSI(DIN), CS, A0(D/C), RES);

int xpos, xpos1, dir = 0;

void tree(){
  u8g.drawFilledEllipse(xpos + 10, 15, 5, 10);
  u8g.drawLine(xpos + 10, 26, xpos + 10, 36);
  u8g.drawEllipse(xpos+42, 15, 5, 10);
  u8g.drawLine(xpos+42, 26, xpos+42, 36);
  u8g.drawFilledEllipse(xpos+74, 15, 5, 10);
  u8g.drawLine(xpos+ 74, 26, xpos+74, 36);
  u8g.drawEllipse(xpos+106, 15, 5, 10);
  u8g.drawLine(xpos+106, 26, xpos+106, 36);
  u8g.drawFilledEllipse(xpos+138, 15, 5, 10);
  u8g.drawLine(xpos+138, 26, xpos+170, 36);
  u8g.drawEllipse(xpos+170, 15, 5, 10);
  u8g.drawLine(xpos+170, 26, xpos+170, 36);
  u8g.drawEllipse(xpos+202, 15, 5, 10);
  u8g.drawLine(xpos+202, 26, xpos+202, 36);
  u8g.drawEllipse(xpos+234, 15, 5, 10);
  u8g.drawLine(xpos+234, 26, xpos+234, 36);
}
```

SH1106 OLED 디스플레이 제어를 위해 라이브러리 추가

OLED 핀을 알맞게 아두이노 UNO 디지털핀에 연결

나무와 자동차의 X좌표 변수, 방향, 변수 선언 및 0으로 초기화

색이 채워진 타원을 그린다. 좌표 (10,15)를 시작점으로 장축의 길이 5pixel, 단축의 길이 10pixel

//색 있는 나무 나무1

//색 없는 나무 나무2

좌표(x1, y1)과 좌표(x2,y2)를 잇는 직선을 그린다.

색상 없이 테두리만 타원을 그린다.

SH1106 128x64

(10,15)

5pixel

10pixel

(10,26)

(10,36)

xpos 변수를 통해 디스플레이 상에 표현된 나무를 지나가는 배경으로 보이게 하는 것이 가능하다.

```
void car(){
```

```
//몸통
```

```
u8g.drawFrame(xpos1 + 54, 35, 27, 10);
u8g.drawFrame(xpos1 + 45, 44, 45, 13);
u8g.drawLine(xpos1 + 54, 35, xpos1 + 50, 44);
u8g.drawLine(xpos1 + 57, 35, xpos1 + 57, 44);
u8g.drawLine(xpos1 + 67, 35, xpos1 + 67, 44);
u8g.drawLine(xpos1 + 77, 35, xpos1 + 77, 44);
u8g.drawLine(xpos1 + 80, 35, xpos1 + 85, 44);
u8g.drawFrame(xpos1 + 86, 48, 4, 3);
u8g.drawLine(xpos1 + 55, 36, xpos1 + 79, 36);
```

```
//바퀴
```

```
u8g.drawDisc(xpos1 + 55, 56, 5);
```

```
u8g.drawDisc(xpos1 + 79, 56, 5);
```

```
//전조등
```

```
u8g.drawLine(xpos1 + 91, 51, xpos1 + 107, 61);
```

```
u8g.drawLine(xpos1 + 91, 48, xpos1 + 126, 61);
```

```
u8g.drawFilledEllipse(xpos1 + 116, 61, 8, 2);
```

```
}
```

```
void draw(void){
```

```
tree();
```

```
car();
```

```
}
```

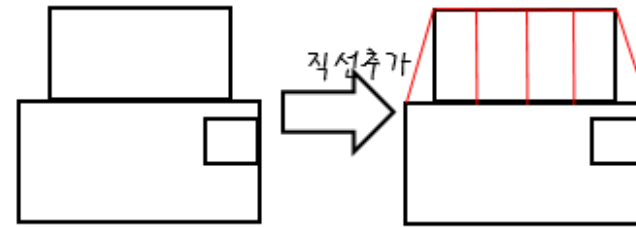
```
void setup(void) {
```

사용자 설정에 의해
실질적으로 OLED에
그림을 그리는 함수

자동차와 나무를 그린다

설정 없음

자동차의 차체를 구성한다.
좌표(53, 35)를 기준으로
너비 27, 높이 10pixel의
사각형을 그린다.

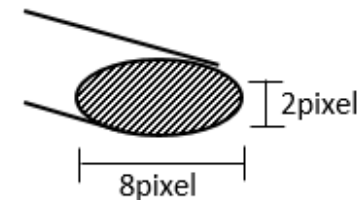


u8g.drawFrame();

자동차의 바퀴를 구성한다.
좌표(55, 56)을 기준으로
지름 5pixel의 원을 그린다.



자동차 전조등에서
나오는 빛을 묘사한다



```
void loop(void) {
  //나무 움직임
  xpos--;
```

나무와 자동차가 움직이는 간단한 알고리즘이다
배경인 나무는 x좌표 값을 1씩 빼주면 뒤로 지나가는
나무 배경을 표현할 수 있다

```
  //차 앞뒤 움직임
```

```
  if(dir==0){
    xpos1-=2;
    if(xpos1 < -30){ dir = 1; }
  }
```

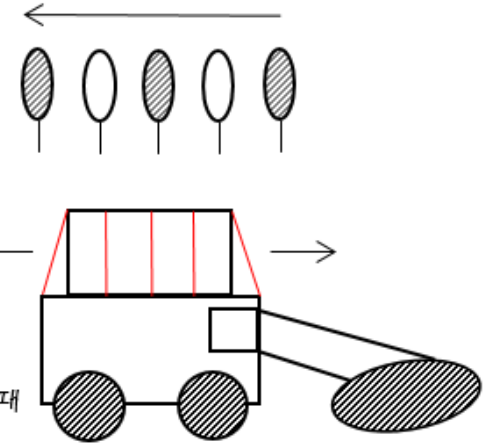
자동차는 움직이는 방향이 0일 때, 뒤로 움직이며
자동차에 속하는 그림의 x좌표가 2씩 감소하여 추진하는 장면을
연출할 수 있다.

```
  if(dir==1){
    xpos1+=2;
    if(xpos1 > 10){ dir = 0 ; }
  }
```

자동차의 그림 x좌표가 -30이 되었을 때,
방향 변수(dir)을 1로 바꾸고 1씩 x좌표값을 증가시키면
전진하는 장면을 연출할 수 있다.
이때, 자동차 x좌표를 변화시키는 변수(xpos1)이 10이상일때
자동차의 움직임 방향은 뒤(dir=0)로 바뀐다

```
  // picture loop
  8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );
}
```

그리기 구간(picture loop)의 프로시저(procedure)로 들어가
draw() 함수에 표현한 그리기를 수행하고 끝나면 0을 반환하여
그리기 구간을 끝낸다.



연습문제

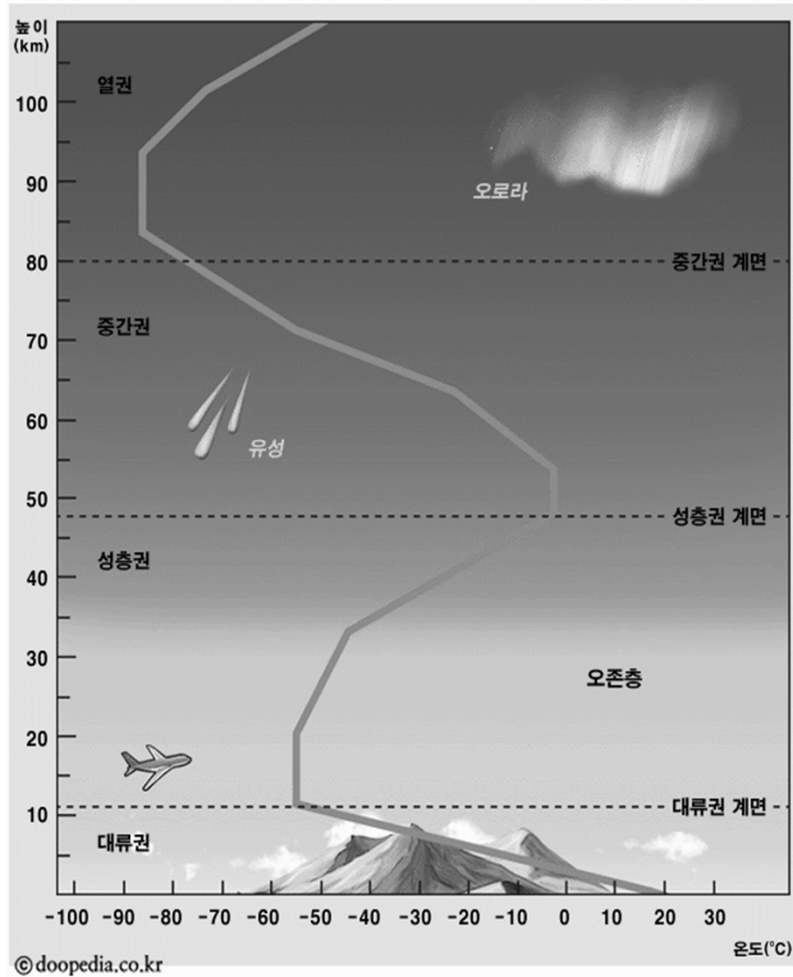
- 1/10 초를 표시하는 시계(00:00:00)를 만들어 보자
- 자기 이름을 쓰고 좌우로 움직여 보자.
- 아날로그 시계를 만들어 보자(초와 분침만 있는).

```
// Calculate clock pin position
double RAD=3.141592/180;
double LR = 89.99;
void showTimePin(int center_x, int center_y, double pl1, double pl2, double pl3)
{
    double x1, x2, y1, y2;
    x1 = center_x + (iRadius * pl1) * cos((6 * pl3 + LR) * RAD);
    y1 = center_y + (iRadius * pl1) * sin((6 * pl3 + LR) * RAD);
    x2 = center_x + (iRadius * pl2) * cos((6 * pl3 - LR) * RAD);
    y2 = center_y + (iRadius * pl2) * sin((6 * pl3 - LR) * RAD);

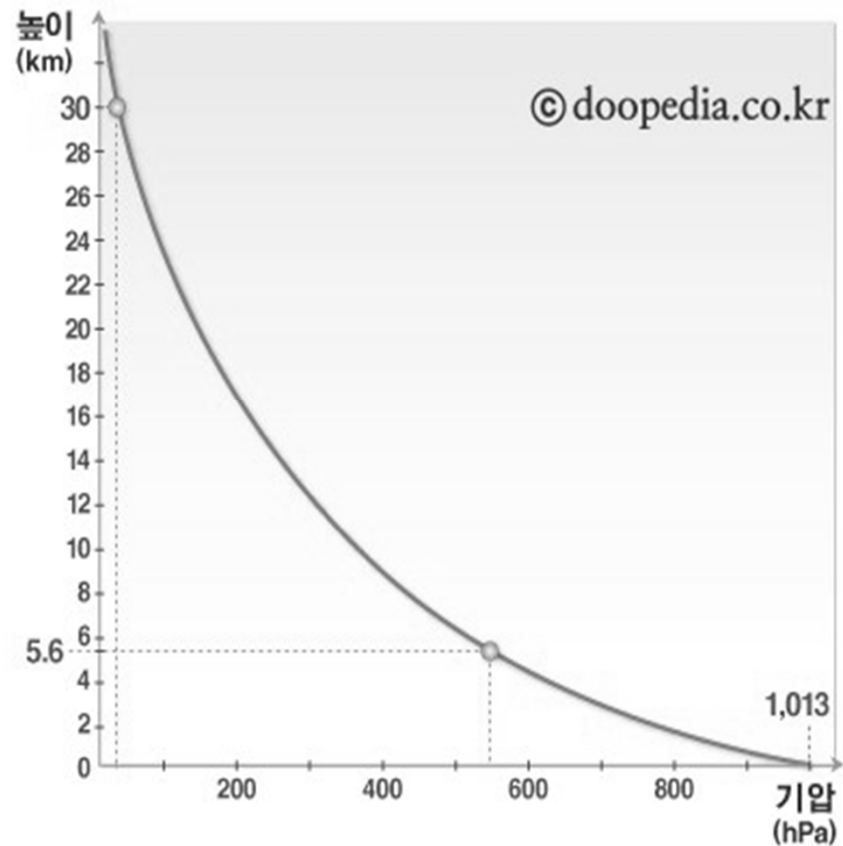
    u8g.drawLine((int)x1, (int)y1, (int)x2, (int)y2);
}
```


2.2.3 기압센서를 이용한 기압 및 온도 측정

대기권의 구성

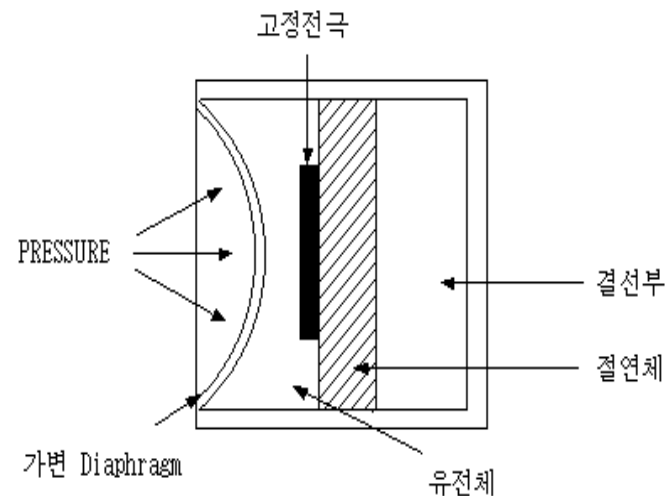


기압과 높이의 관계도

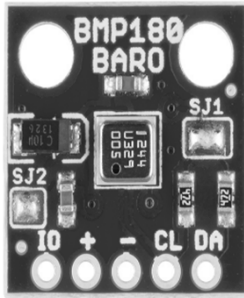


기압계란

기압센서는 스트레인 게이지(Strain gauge)의 저항치가 압력에 비례해 변화하는 것을 이용해 압력을 아날로그 전압으로 변화시키는 반도체 피에조(Piezo) 저항형 센서로, 기압력을 아날로그 전압으로 변환한 신호를 이용하여 기압고도(Pressure altitude)를 계산하므로 물체의 고도를 알 수 있다.



기압측정센서 | BMP180



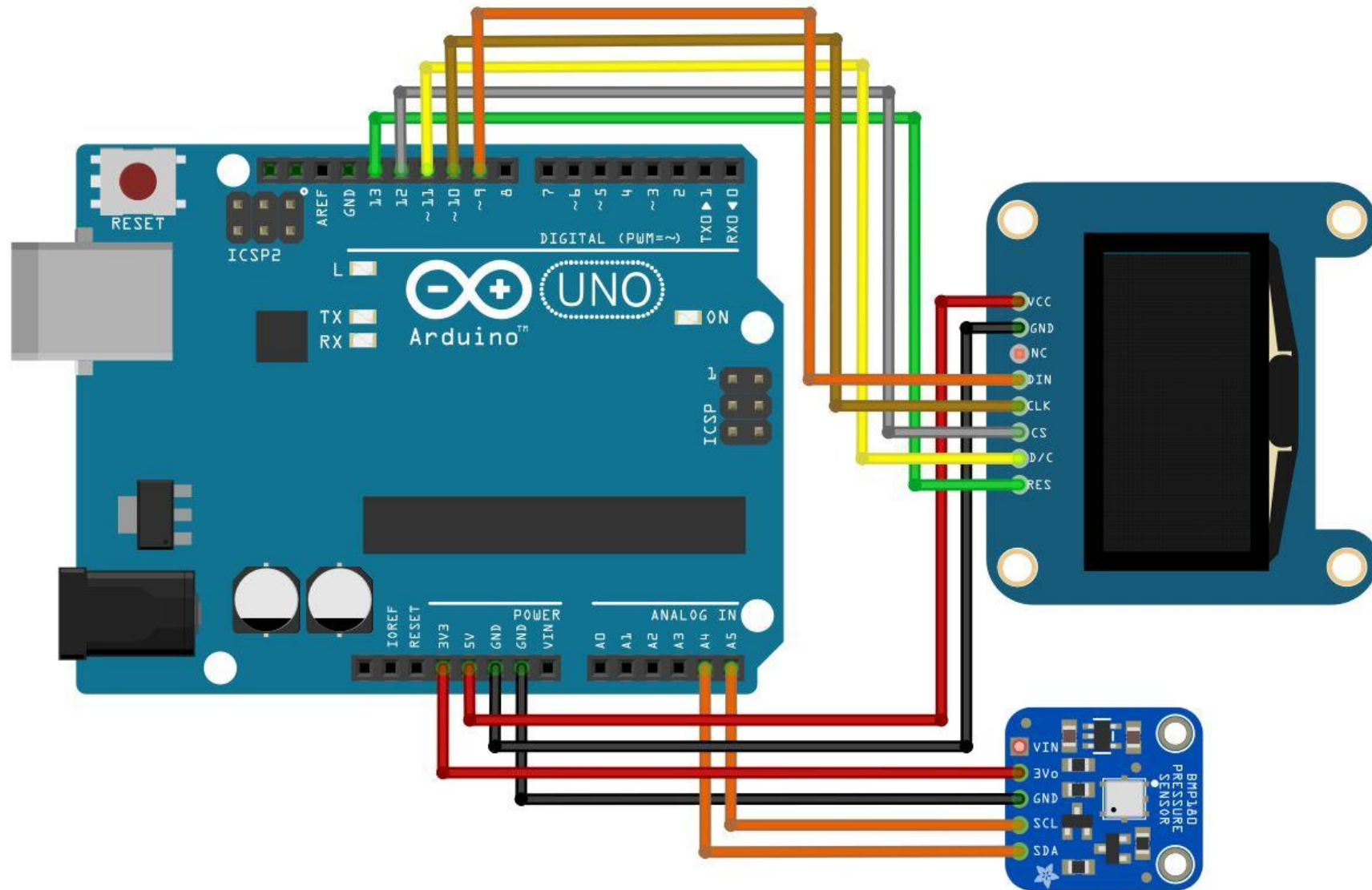
항목	범위	분해능	오차
압력	300~1100 hpa	0.01 hPa	±0.25m (실제 ±1m)
온도	-40~85 °C	0.1 °C	

기압센서는 I2C 통신을 통해서 센서 측정값을 수신 받을 수 있다.

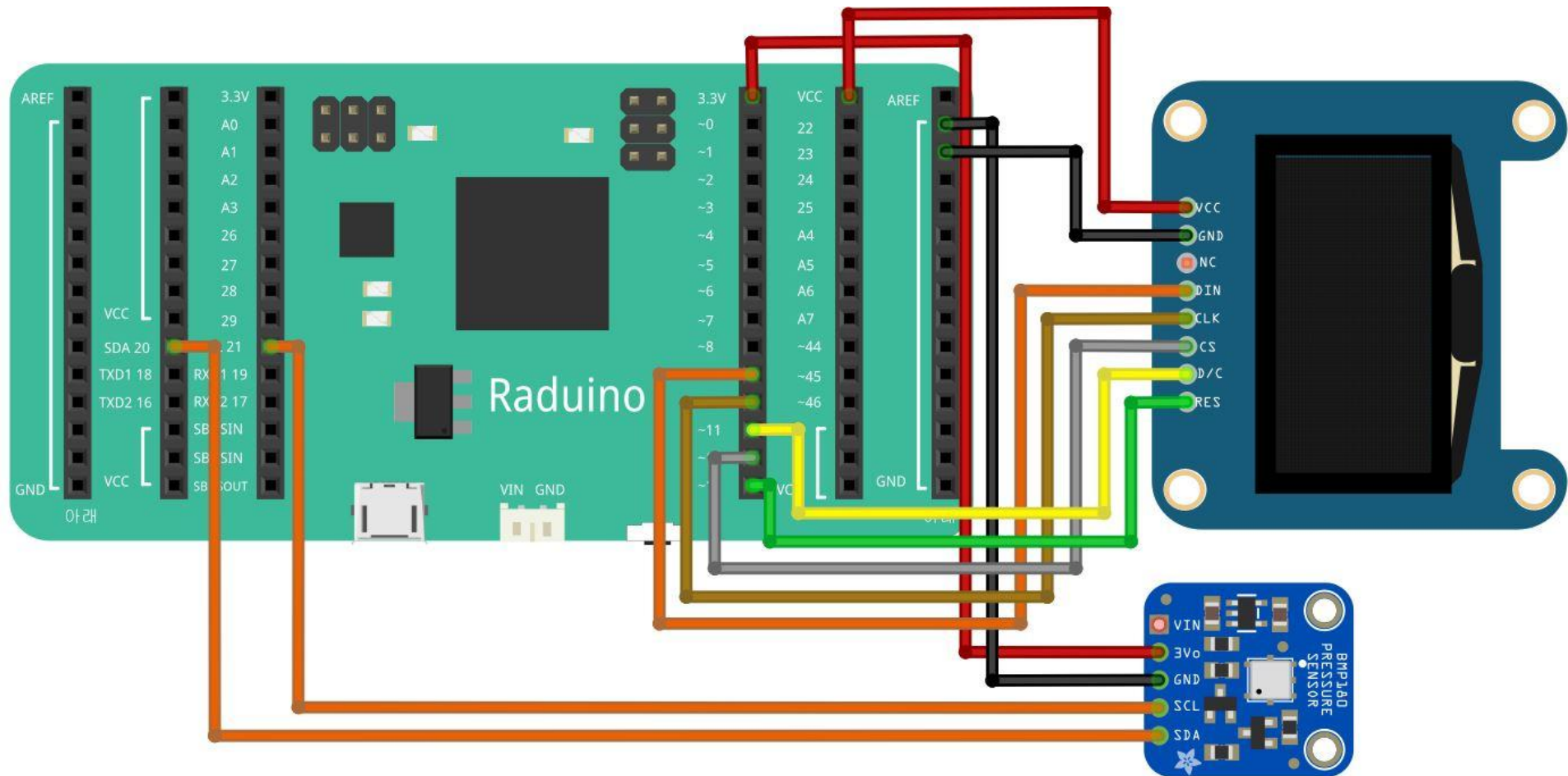
핀맵 2-2 기압측정센서 BMP180의 연결 PIN MAP

BMP180	IO	VIN	GND	SCL	SDA
아두이노 우노		3.3V	GND	A5	A4
라두이노		3.3V	GND	SCL	SDA

연결 회로도



연결 회로도



스케치코드 2-6 예제

```
#include <U8glib.h>
#include <SFE_BMP180.h>

//U8GLIB_SH1106_128X64 u8g(SCK(CLK), MOSI(DIN), CS, A0(D/C), RESET)
U8GLIB_SH1106_128X64 u8g(10, 9, 12, 11, 13);

Double T,P,p0,a;
int xpos = (int)T + 40;
Float cur, prev = 0;

// SFE_BMP180 객체를 "pressure"로 선언:
SFE_BMP180 pressure;

#define ALTITUDE 34.91 // Altitude of DNU's HQ in Jeonju, CO. in meters

void draw(void){
  u8g.setFont(u8g_font_unifont);
  u8g.setPrintPos(20,10);
  u8g.print(a, 0);
  u8g.drawStr( 65, 10, "meter");
  u8g.setPrintPos(20,30);
  u8g.print(T, 2);
  u8g.drawStr( 70, 30, "C");
  u8g.drawCircle( 65, 20, 2);
  u8g.drawRBox(1,52,xpos,9,3);
  u8g.drawFrame(0,48,127,15);
}
```

SH1106 OLED 디스플레이 제어를 위해 라이브러리 추가

온도, 기압센서 BMP180 제어를 위해 라이브러리 추가

아두이노와 SH1106 OLED를 연결한다

온도 · 기압 · 해수면의 압력,절대고도에 대한 변수 선언

측정 기압과 온도에 따른 온도계 그림의 변화를 나타내도록, 변수선언
x좌표값(x_pos)과 현재온도(cur),이전온도(prev)

SFE_BMP180라이브러리에 사용된 변수나 함수를 사용하기 위해 클래스의 (참고)변수 선언!

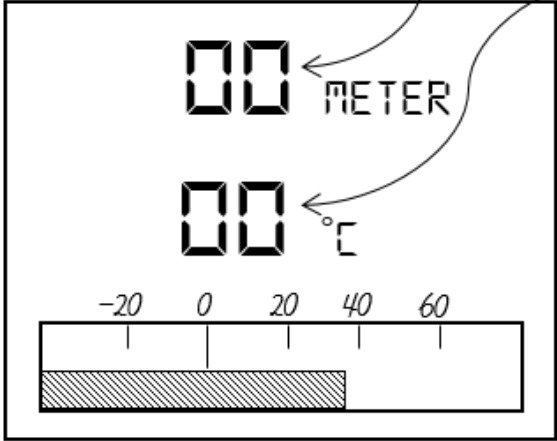
현재 기압을 측정하는 곳의 해발고도를 정의한다 변수a

변수T

사용자 설정에 의해 실질적으로 OLED에 그림을 그리는 함수

OLED에 다음과 같이 표현한다

모서리가 둥근 상자를 (1, 52) 좌표 기준으로 길이 xpos, 높이 9pixel, 모서리 30그린다



```

u8g.drawLine(41,46,41,50);
u8g.drawLine(21,47,21,49);
u8g.drawLine(61,47,61,49);
u8g.drawLine(81,47,81,49);
u8g.drawLine(101,47,101,49);
u8g.setFont(u8g_font_u8glib_4);
u8g.drawStr(22,46, "-20");
u8g.drawStr(42,46, "0");
u8g.drawStr(62,46, "20");
u8g.drawStr(82,46, "40");
u8g.drawStr(102,46, "60");
}

```

직사각형의 프레임은 (0,48)좌표 기준으로
길이 127pixel, 높이 15pixel로 그린다

```

void setup(){
  Serial.begin(9600);
  Serial.println("REBOOT");

```

PC와 VART 시리얼통신을 baudrate : 9600 으로 설정

BMP180 begin() 함수의
리턴값이 참(1)이라면

Pressure.begin() 함수에서 I2C통신을 시작하며
BMP180의 EEPROM에서 공장보정값
(FactoryCalibration data)을 읽어 온다.

```

if (pressure.begin()) Serial.println("BMP180 init success");
else{
  Serial.println("BMP180 init fail\n\n");
  while(1); // Pause forever.
}
}

```

BMP180의 초기화 성공메세지를
시리얼 모니터에 출력

BMP180의 초기화가 수행되지 않으면
초기화 실패 메시지와 함께 더 이상 작업은
수행하지 않는다.

```
void loop(){
  char status;
```

상태변수를 선언

```
  status = pressure.startTemperature();
  if (status != 0){
    delay(status);
    status = pressure.getTemperature(T);
    if (status != 0){
```

보상되지 않은
온도 측정값을 읽는다.

온도샘플링 값에
따른, delay 값을
받으면

온도값은 I2C통신을 통해
수신 받았다면 return(5),
못 받았다면 return(0)

```
      status = pressure.startPressure(3);
      if (status != 0){
        delay(status);
```

보상되지 않은 기압값을 읽는다
이때, oversampling resolution을 (최대값)
3으로 설정했기 때문에 return(26)

압력샘플링에
따른 delay 값을
받으면

```
      status = pressure.getPressure(P,T);    측정압력값을 읽는다
```

측정	최대 변환값(ms)
온도	4.5
압력 0	4.5
압력 1	7.5
압력 2	13.5
압력 3	25.5

출처 BST-BMP185-DS000-09.pdf
↑ 데이터시트 참조!

```
  if (status != 0){
```

```
    p0 = pressure.sealevel(P,ALTITUDE);
```

해수면의 압력을 계산한다. 이때, 초기 절대고도값(ALTITUDE)으로

해발고도 정보를 사용(주)DNU는 34.91m)

```
    a = pressure.altitude(P,p0)
```

절대고도값을 계산한다
(기압고도)

```
  }
```

```
  }
```

```
  }
```

```
}
```



```
// picture loop
u8g.firstPage();
do{
    draw();
} while( u8g.nextPage() );
```

} 그리기(*Active loop*)를 시작하여
Draw() 함수를 실행하고 그리기 구간을
끝낸다.

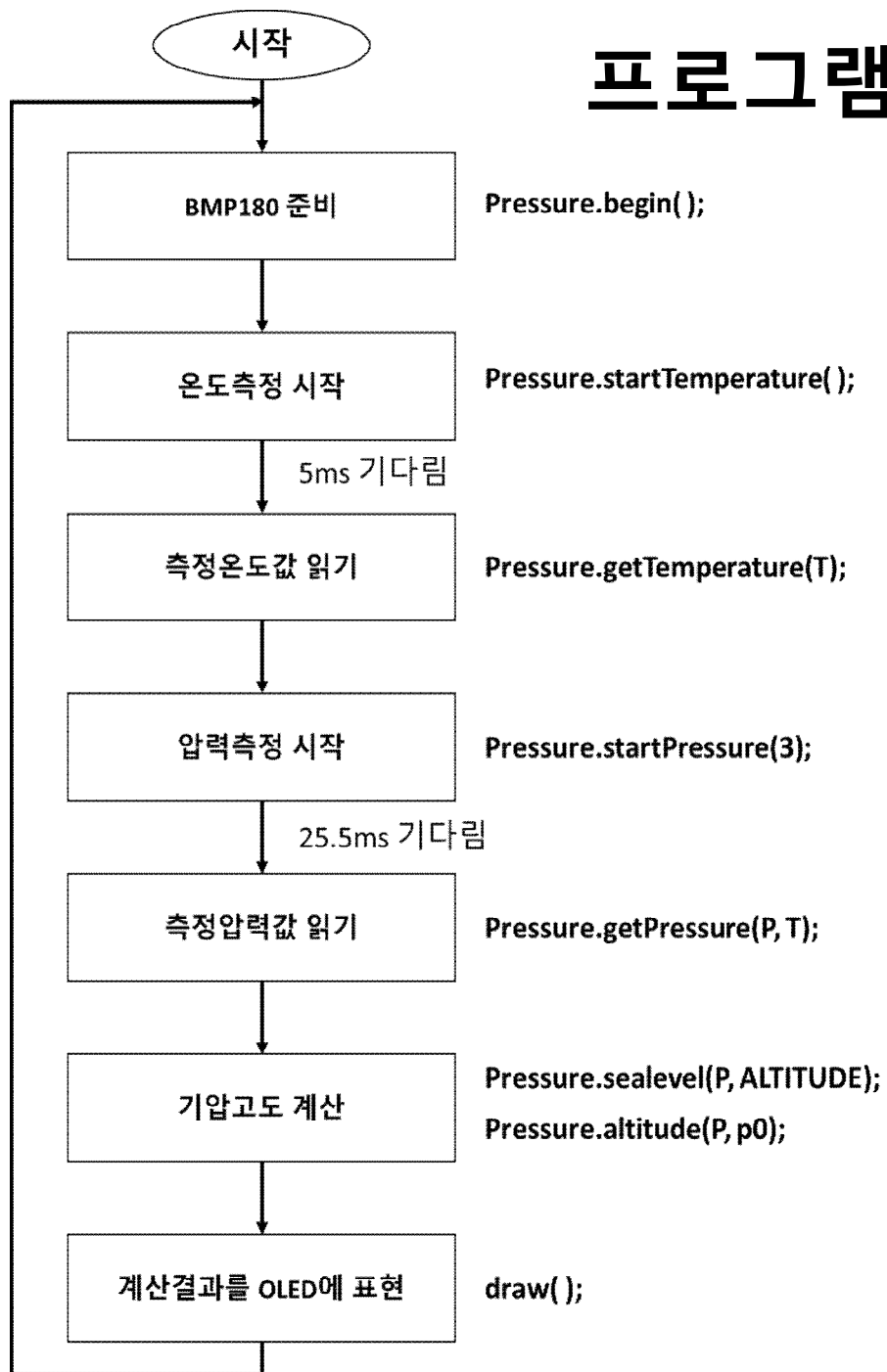
```
prev=cur;    이전 온도와 현재 온도 변수 값을 갱신한다
cur = (int)T; 현재 온도 값에는 측정 온도 변수 T를 int형으로 형변환한 후 저장
```

```
if(cur == prev)      xpos = xpos;
else if(cur > prev) xpos = xpos+(cur-prev);
else if(cur < prev) xpos = xpos-(prev-cur);
```

} 현재 온도와 이전 온도 차에 따라
OLED에 표현한 온도계 그림의 길이를
조정하는 알고리즘이다.

```
}
```

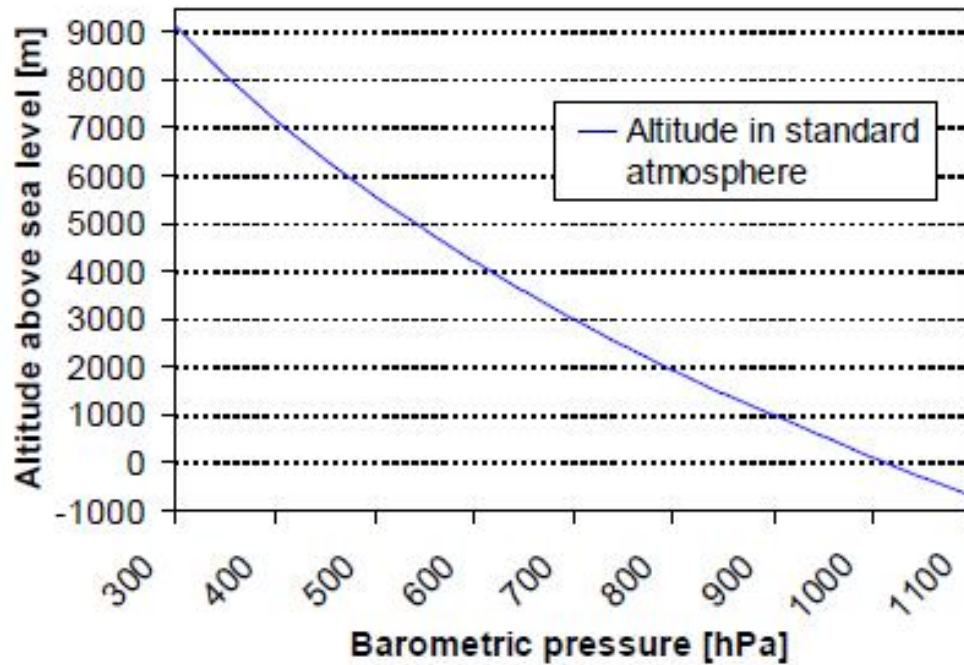
프로그램 순서도



기압과 고도의 관계식

- 해수면의 압력 계산식 : `pressure.sealevel()` 함수

$$p_0 = \frac{p}{\left(1 - \frac{altitude}{44330}\right)^{5.255}}$$



- 절대 고도 계산식 : `pressure.altitude()` 함수

$$altitude = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

[기압과 해발고도의 관계 그래프]