Pollution Sensor

Matlab Euro CAQI levels converter for Thingspeak service

I decided to move all calculations & aproximations from arduino code to Thingspeak service. Sensor will be only responsible of uploading raw data to Thingspeak service. This significantly reduces my efforts to upoad new software to sensor module.

Here are functions written in Matlab language that is used by Thingspeak to perform data processing and converting PM2.5 levels to CAQI Euro standard.

```
%// Euro AQI Standard

%// AQI formula: https://en.wikipedia.org/wiki/Air_Quality_Index
function output = toAQI(I_high, I_low, C_high, C_low, C)
  aqiResult = (I_high - I_low) * (C - C_low) / (C_high - C_low) + I_low
end


function output = calculateAQI25(density)
  d10 = density * 10;

  if d10 <= 150
    output =  toAQI(25, 0, 150, 0, d10);
  elseif d10 <= 300
    output =  toAQI(50, 26, 300, 151, d10);
  elseif d10 <= 500
    output =  toAQI(75, 51, 500, 301, d10);
  elseif d10 <= 1000
    output =  toAQI(100, 76, 1000, 501, d10);
  elseif d10 <= 3000
    output =  toAQI(300, 101, 3000, 1001, d10);
  else
    output = 1001;
  end
end


%// Polution levels

function output = aqi25captions(density)
  if density <= 25
    output =  'Perfect =)';
  elseif density <= 50
    output =  'Good :)';
  elseif density <= 75
    output =  'Moderate :|';
  elseif density <= 100
    output =  'Unhealthy :(';
  elseif density <= 200
    output =  'Bad :O';
  else
    output =  'Hazardous X(';
```

```
end
end
```

#PM2.5 #caqi #pollution #thingspeak #dustsensor

Formula to calculate CAQI Index

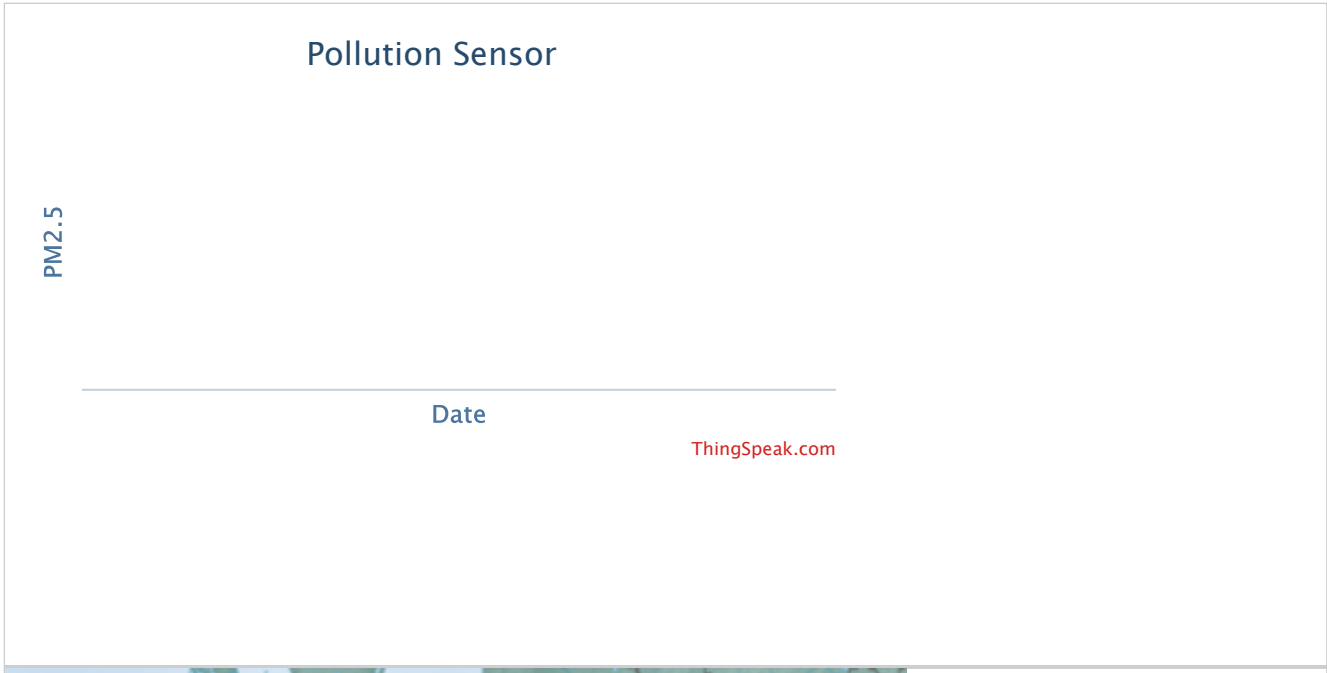
Formula to calculate CAQI is described on Wikipedia.
https://en.wikipedia.org/wiki/Air_quality_index#Europe

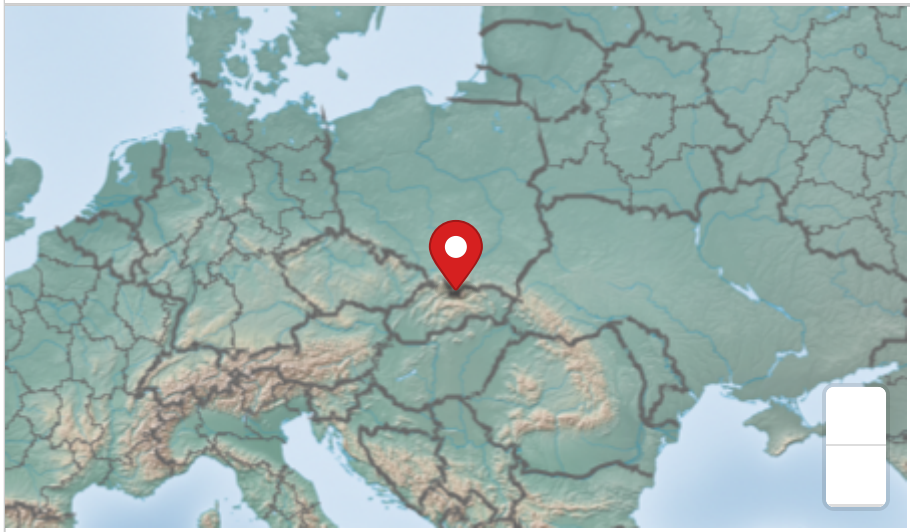
Unfortunately there are only few examples realized in code on the internet.

```
// AQI formula: https://en.wikipedia.org/wiki/Air_Quality_Index#United_States
int toAQI(int I_high, int I_low, int C_high, int C_low, int C) {
  return (I_high - I_low) * (C - C_low) / (C_high - C_low) I_low;
}

//thanks to https://gist.github.com/nfjinjing/8d63012c18feea3ed04e
int calculateAQI25(float density) {

  int d10 = (int)(density * 10);
  if (d10 <= 0) {
    return 0;
  } else if(d10 <= 120) {
    return toAQI(50, 0, 120, 0, d10);
  } else if (d10 <= 354) {
    return toAQI(100, 51, 354, 121, d10);
  } else if (d10 <= 554) {
    return toAQI(150, 101, 554, 355, d10);
  } else if (d10 <= 1504) {
    return toAQI(200, 151, 1504, 555, d10);
  } else if (d10 <= 2504) {
    return toAQI(300, 201, 2504, 1505, d10);
  } else if (d10 <= 3504) {
    return toAQI(400, 301, 3504, 2505, d10);
  } else if (d10 <= 5004) {
    return toAQI(500, 401, 5004, 3505, d10);
  } else if (d10 <= 10000) {
    return toAQI(1000, 501, 10000, 5005, d10);
  } else {
    return 1001;
  }
}
```

Calibrated data

I succed to calibrate my sensor using nearest official sensors. Sensor real time data is published on my open Thingspeak channel.





PMS5003 sensor in action

I suceed to build my second sensor. This time it is more expensive. PMS5003 is not analog sensor like Sharp GP2Y1010AU0F. PMS5003 is laser-based sensor with built in microcontroller that performs all required calculations and sends digital data messages to its TXD pin5. You could use other sensors like PMS6003, PMS7003 and even older ones like PMS1003 or PMS3003. They all could be connected in the same way, just pay attention at message length in the source code (If I am not wrong PMS5003/6003/7003 uses 32byte long messages but PMS3003 uses 24bit).

My first workingh prototype was connected to NodeMCU by only 3 wires: +5V, GND and RX. It was enough to activate sensor and start receiving some data on RX line.

PMS5003 requres +5V but data lines should be connected to 3.3V levels which is used by NodeMCU

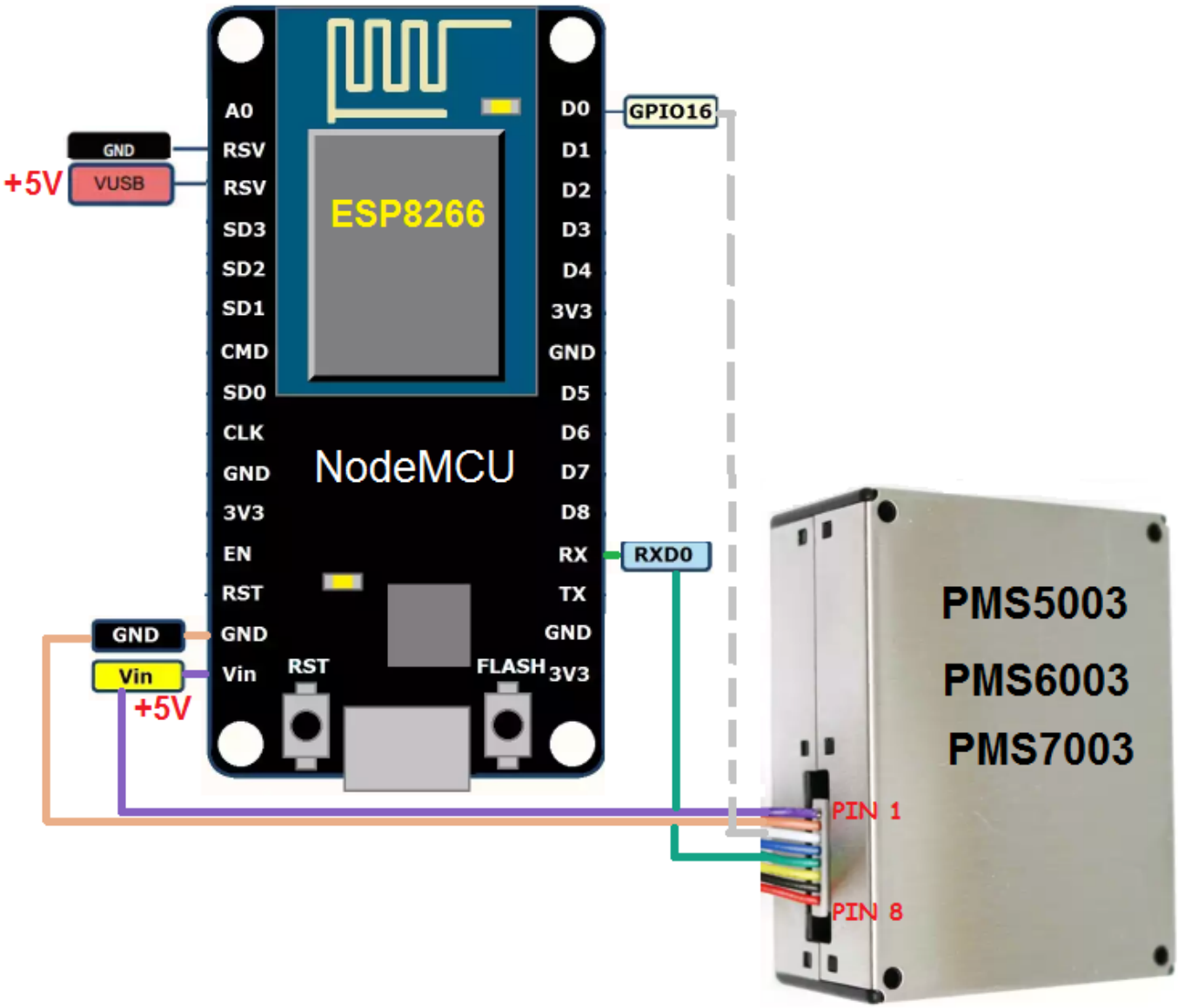
** Be aware if you would like to connect PMS5003 to Arduino you should use 5<=>3.3v logic level converter.*

Also please note that some NodeMCU versions dont support 5V on Vin pin. In my case I have chinese clone LoLin NodeMCU V3 board that has 5V on reserved pins. The simpliest test is to connect power to PMS5003 PIN1 and PIN2 and you should hear a fan noise immediately.

I've built my 1st prototype and succesfully programmed using article from DFRobot: https://www.dfrobot.com/wiki/index.php/PM2.5_laser_dust_sensor_SKU:SEN0177

Although their sample code is for Arduino it works fine on NodeMCU too.

Following diagram has some additional pin connected to digital output but it is not requred for prototype. I used it to force PMS5003 to sleep and save a bit power between measurements. I will describe ti in my next post, together with my source code.



Pin number	Function	Explain
PIN1	VCC	Power supply DC5V
PIN2	GND	Negative power supply
PIN3	SET(Internal 50K pull-up)	Set pin 3.3V level
PIN4	RXD/I2C_SCL	Digital pin 3.3V level
PIN5	TXD/I2C_SDL	Digital pin 3.3V level
PIN6	RESET	Module reset signal 3.3V level
PIN7	PWM output	3.3V level only for single sensor output
PIN8	Mode selection (Internal 50K pull-up)	High or NC Serial port mode 3.3V level Low level I2C model

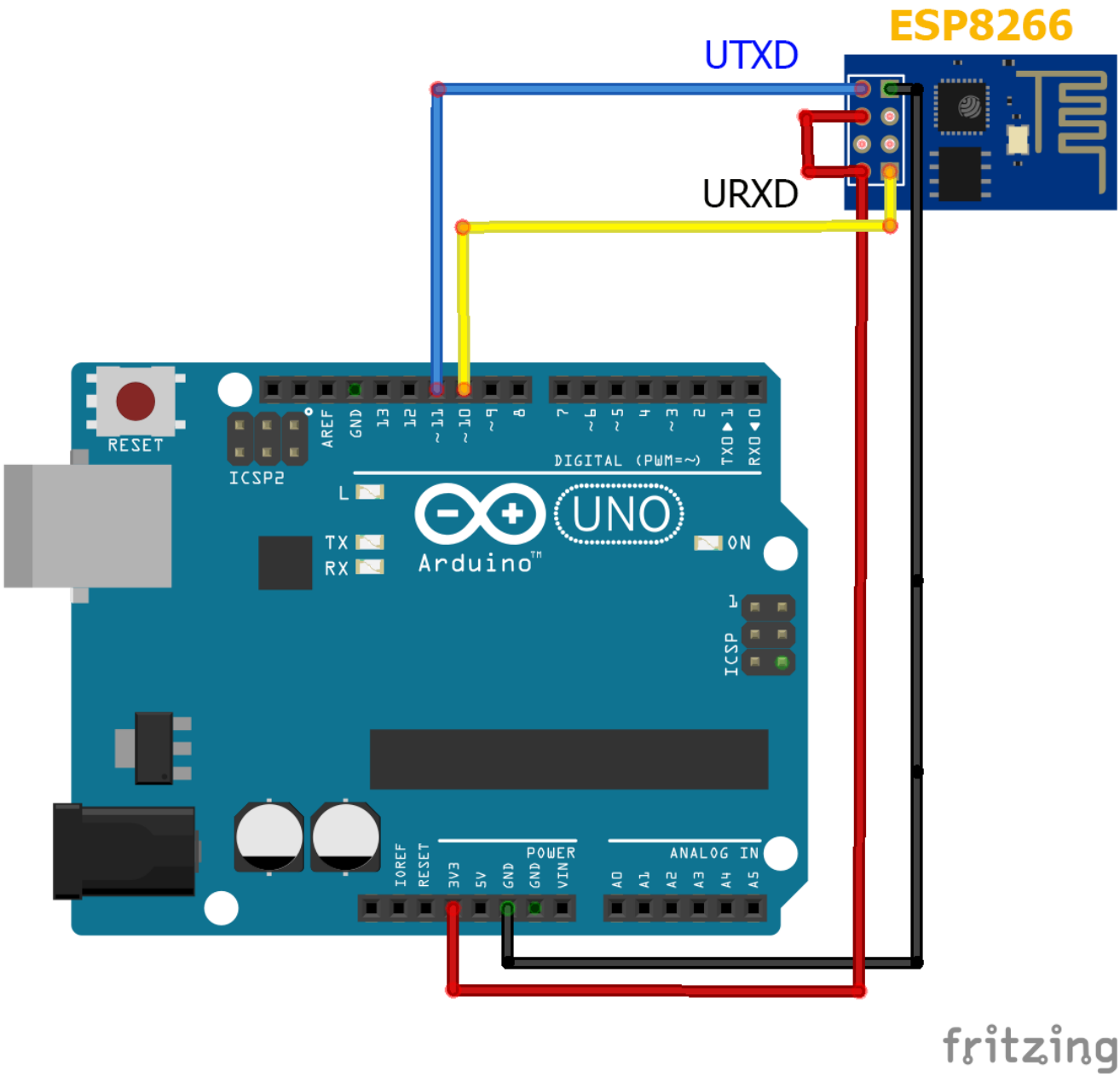
Usefull links:

- https://www.dfrobot.com/wiki/index.php/PM2.5_laser_dust_sensor_SKU:SEN0177
- http://aqicn.org/sensor/pms5003-7003/
- https://github.com/vlytsus/demcusensor/blob/master/dust_wifi.ino

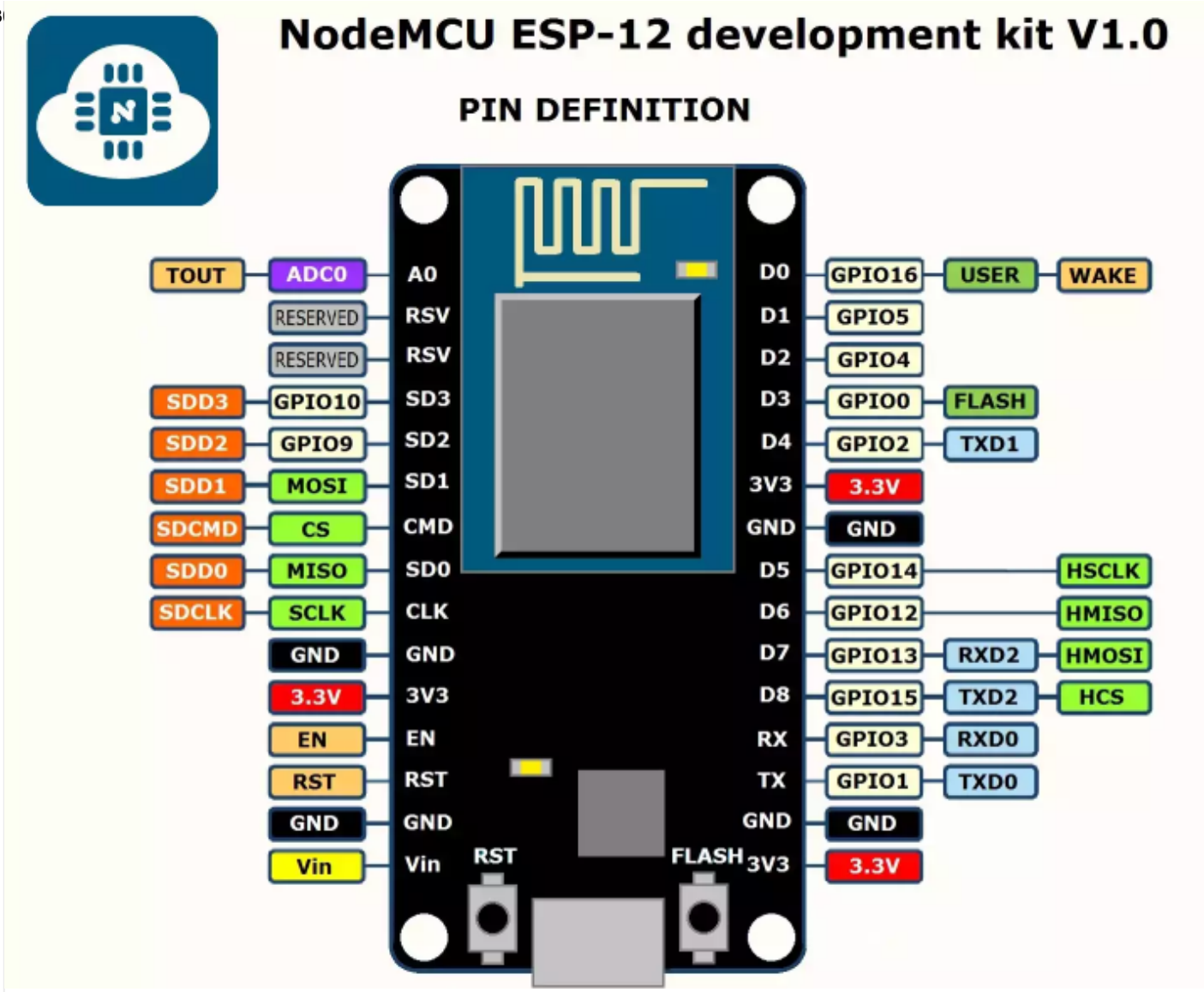
#PMS5003 #PMS6003 #PMS7003 #PMS1003 #PMS3003 #NodeMCU #PM2.5 #Pollution

ESP8266 NodeMCU

I have been looking for a simple solution to connect pollution sensor to the network and push data to colud for aggregation & visualisation. Arduino is good if you have USB connection to some PC that is connected to internet. But unfortunately there are no simple (and inexpensive) solutions for Arduino if you like to have autonomous device and sporadicaly connect to internet by WIFI to post sensor data. My first solution was ESP8266 WIFI chipset connected to Arduino:



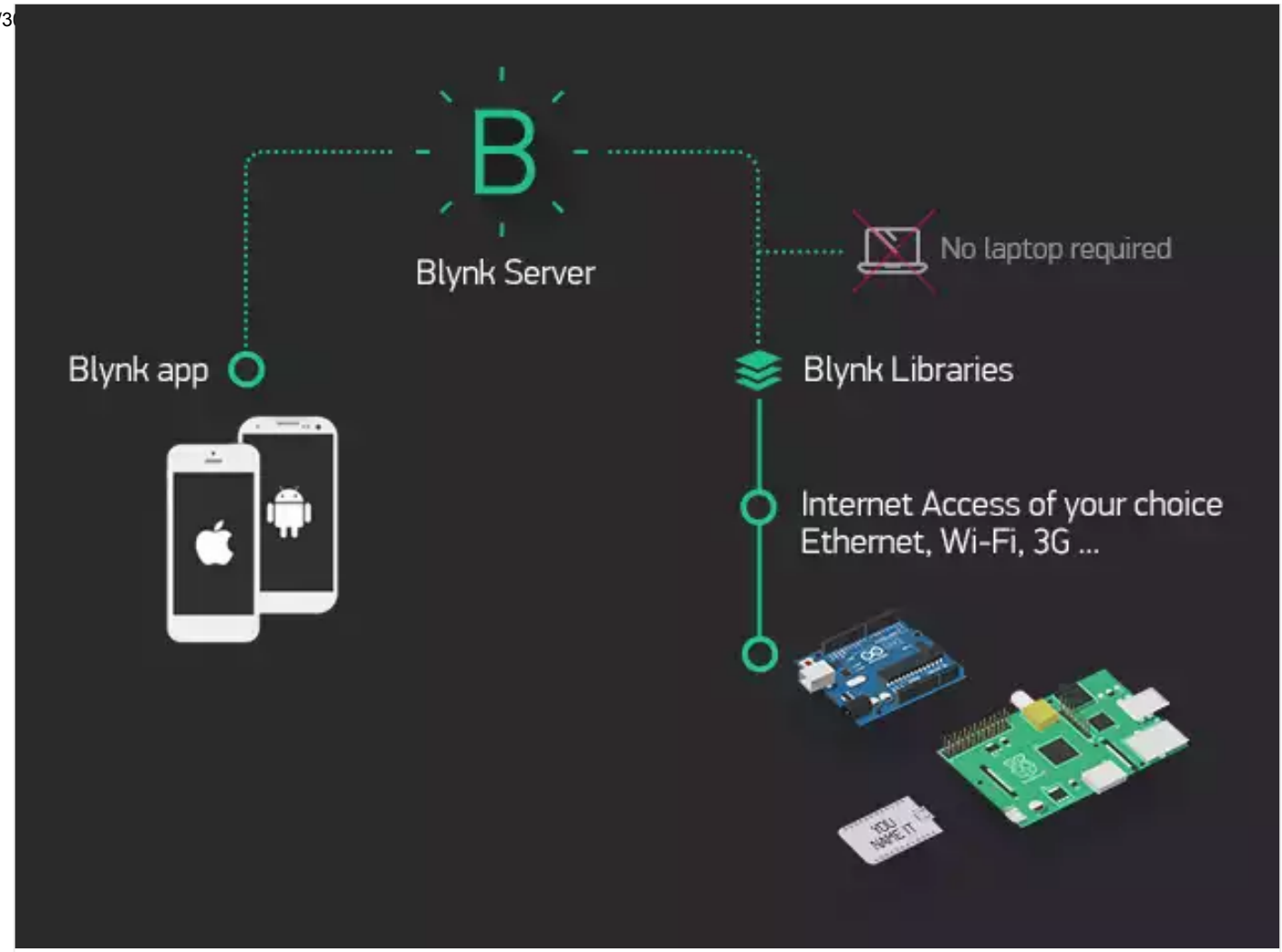
Trying to connect Arduino to WIFI using ESP8266 I accidently found better solution. **NodeMCU** board - Arduino like microcontroller based on **ESP8266** chipset.



Great advantage of **NodeMCU** is that you could program it with onboard USB bridge using Arduino IDE and there are several frameworks that allow you to control chipset from your Android or iOS device and push data directly from microcontroller to the cloud services using HTTP protocol.

For example you can easily create Android application using Blynk IoT service: http://www.blynk.cc/


T-Mobile
6:24 PM
27%
TEMPERATURE C°
V27
24.786
0
50
HUMIDITY. %
V22
28.6
0
100
30
28
26
24
22
20
18:23 20:47
23:11
01:35
03:59
06:23
08:47
1h
6h
1d
1w
BEDROOM
V5
KITCHEN
D7
ZERGBA
V15 255 255 255
Widget Box
YOUR ENERGY BALANCE
2000
+Add
DISPLAYS
25°C
Value Display
200
LED
100
Gauge
300
LCD
400
Graph
400
History Graph
900



Here are several resources about how to program NodeMCU using Arduino IDE:

- http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/arduino-esp8266-lolin-nodemcu-getting-started/
- https://www.youtube.com/watch?v=tMRpYmDgkL0
- http://www.instructables.com/id/Quick-Start-to-Nodemcu-ESP8266-on-Arduino-IDE/

I have bought cheap chinise **NodeMCU** clone for 3$ and succesfully programmed it using those tutorials. It is able to enumerate all nearby WIFI routers and connect one with WPA2 authentication & provided password.

#esp8266 #nodemcu

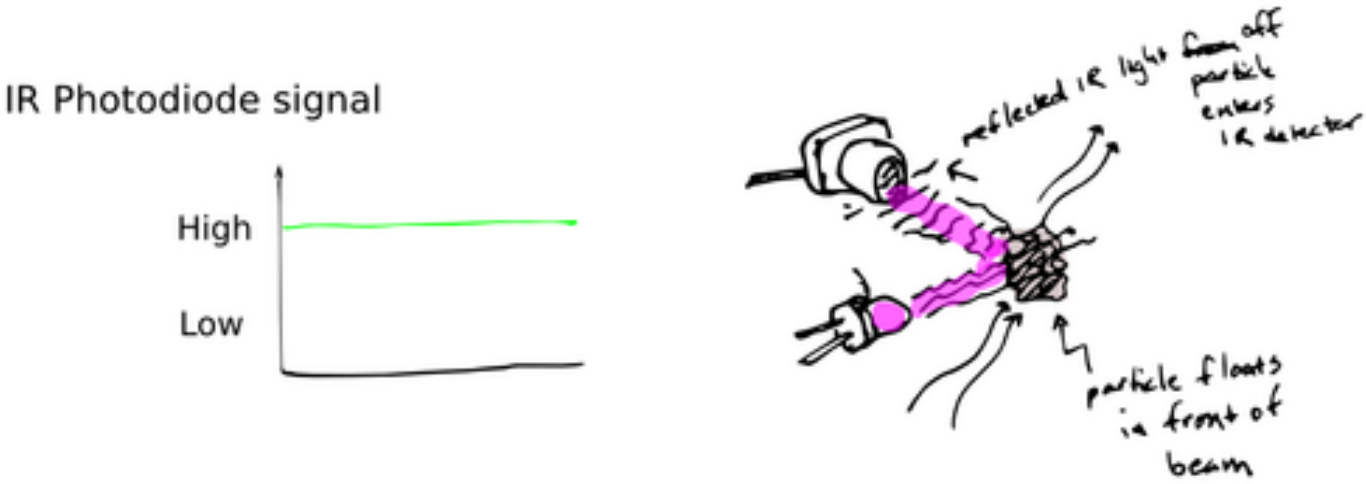
Plantower PMS 5003

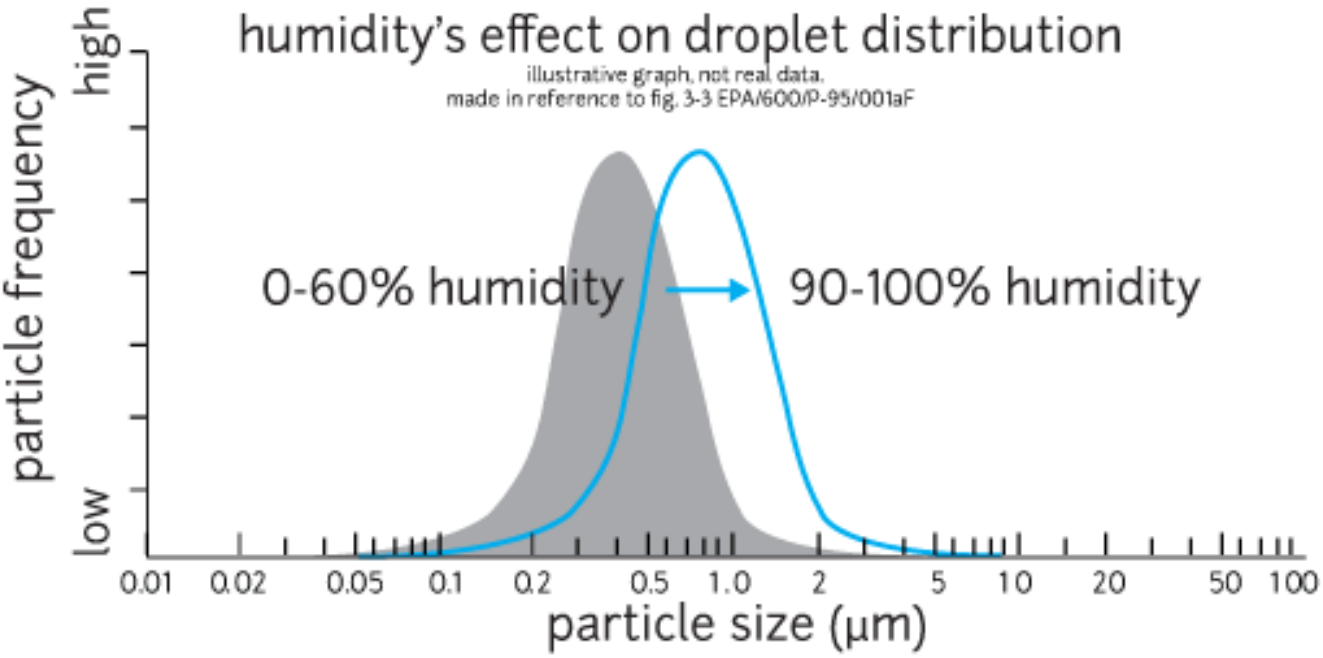
I’ve bought more precise (and 5x-times expensive) sensor - Plantower PMS 5003. Soon I will make a post about how to connect it to arduino and some comparisons with sharp sensor.

How optical particle sensors work

Interresting article about subject: https://publiclab.org/wiki/optical-pm

A sensor and a focused light (sometimes a laser) sit at an angle to each other. As a particle passes in front of the light, some light is reflected towards the sensor. The sensor registers a pulse for as long as the particle reflects light to the sensor. If the air is moving at a consistent speed, the length of this pulse can be used to estimate the particle's diameter…





Anonymous asked:

Hi, Thanks for your great articles. I'm trying to do the same PM2.5 measurement by using Arduino Nano + GP2Y1010AU0F, too(without additional Dust Sensor board). But what I'm going to do is using 3.3v as power supply. Because I want to use 3.7v lithium battery for mobile purpose. My question is: By using 3.3v, I found the output

> voltage isn't reflect to the factor as 3.3/5. Do I need to note anything else? Thank you very much in advance. --Jasper Chen

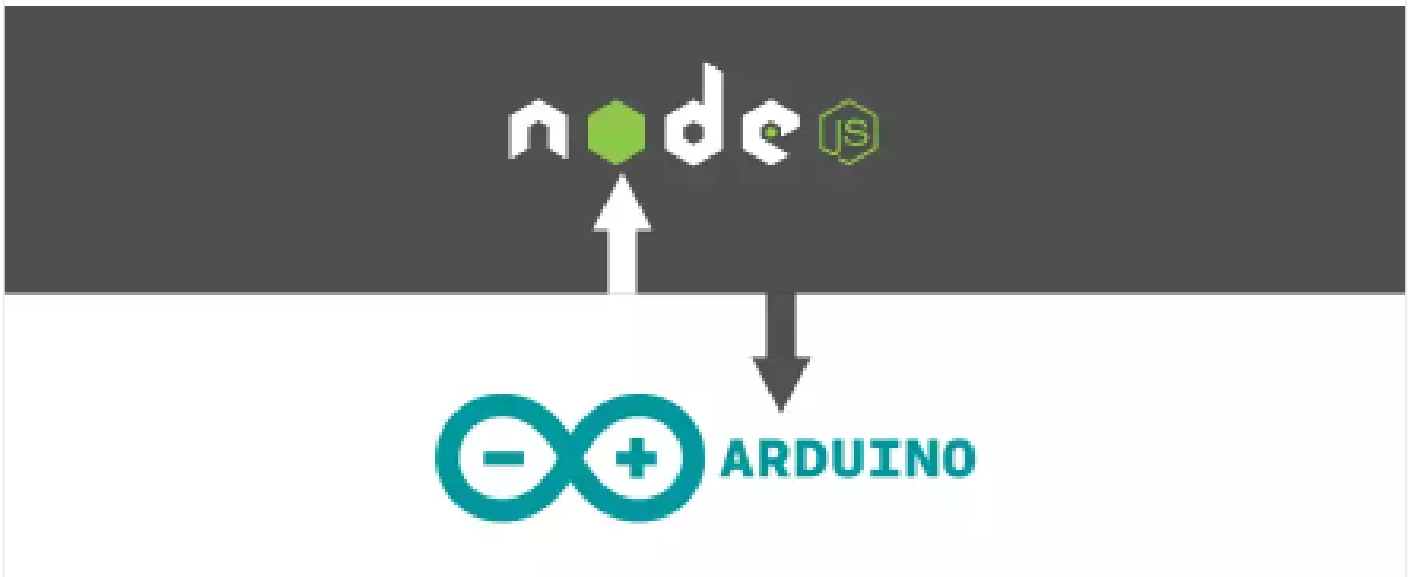
Hard to say, because GP2Y1010AU0F specification said that sensor power should be 5V. You could try to calibrate it using 5V and then switch to 3.7V to check if it still works well.

Read serial data from USB

I would like to read data about PM2.5 levels from Arduino and proceed on my home PC. Then it will be possible to generate charts, send notifications and calculate pollution statistic.

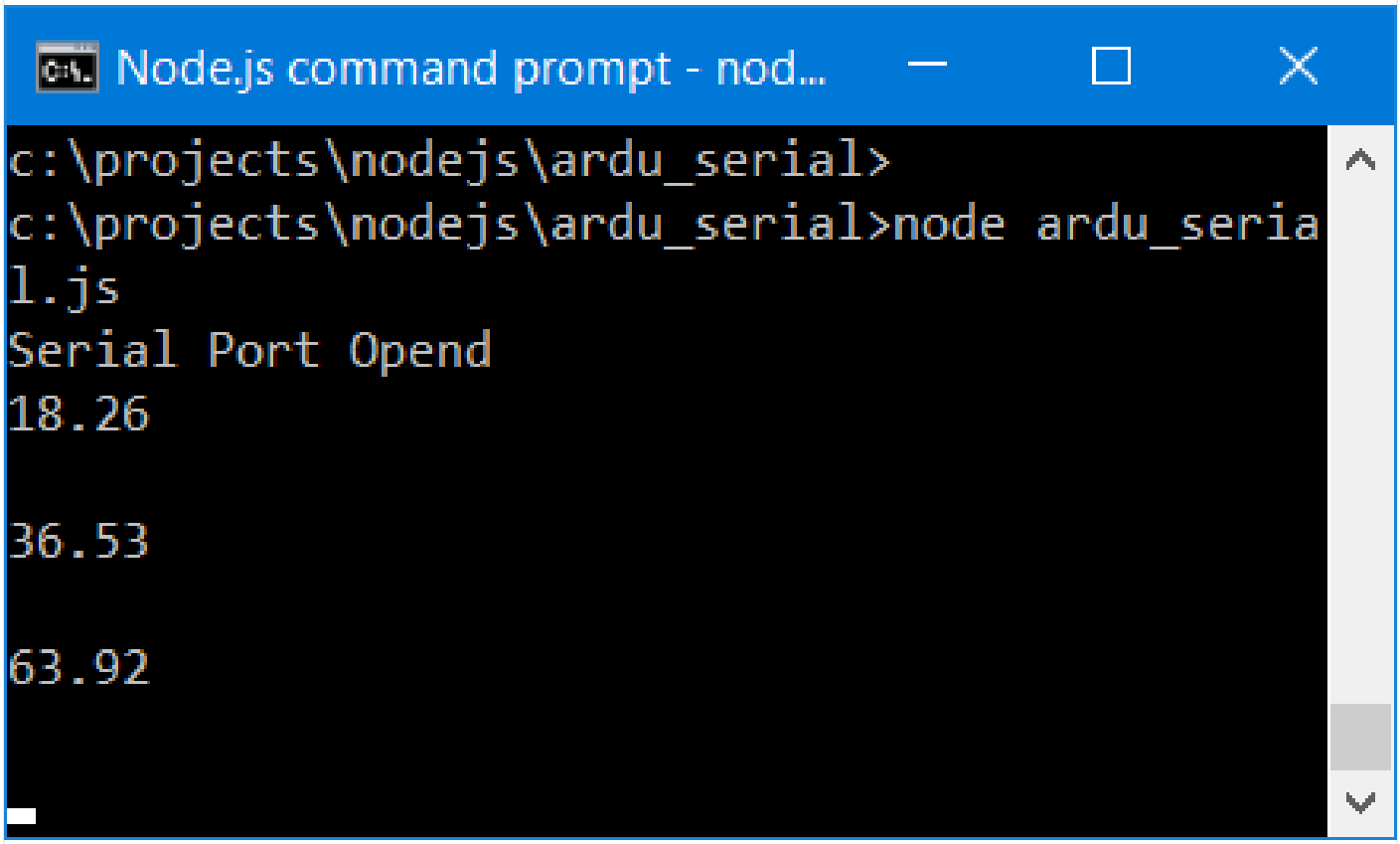
As you know it is not a problem to read serial data from Arduino IDE. But how to proceed it?

I decided to try one of modern popular and simple to use technologies - Node JS. It is JavaScript based framework that dones't require Web Browser to execute Javascript on computer and it has a lot of libraries to work with web serices as well as communication with local peripheral devices connected to USB or serial ports.



It took me 20 minutes to download & install Node JS software from https://nodejs.org/ and run my first program. It is able to read and print serial data from Arduino.

```
1: var SerialPort = require('serialport');
2: var serialport = new SerialPort('COM3');
3:
4: serialport.on('open', function(){
5:   console.log('Serial Port Opend');
6:   serialport.on('data', function(data){
7:   console.log(data.toString('utf8'));
8:   });
9: });
```



First of all you need to install “serialport” library that is not included to Node JS package.

Run Node.js command prompt and run

```
npm install serialport
```

After that you could call your application. I stored scipt to ardu_serial.js file. So, to call it as application just run it from Node.js console:

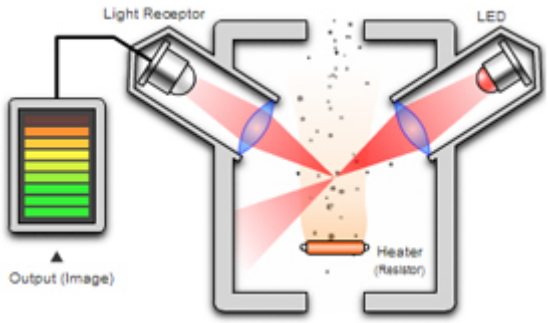
```
1: node ardu_serial.js
```

In next posts I am going to show how to proceed data in node.js and upload real time data samples to some online pollution analytics service or post messages to popular channels.

#arduino #pm2.5 #node.js #pollution

Calibration

If take to consideration, that sensor uses reflected IR light from dust particles to measure air quality. We can make automatic calibration for cleanest air if sample data without flashing IR LED. No light/reflection means cleanest air.



But unfortunately when I tried to measure dust level without IR, I've got maximum dust value, like I have by putting some stick inside the hole :(

Now I don't understand why is that…

New fixed application available

Finally I found and fixed my mistake in formulas. I've used incorrect axes in linear equation y = ax + b

or: voltage = a*dust + b

Fixed: dust = (voltage - b) / a

Please download new version
from: https://github.com/vlytsus/arduinosensor/blob/master/arduino_sensor_main.ino

jaspercmc asked:

> Hi, I'm the one who just asked about 3.3v power input for Arduino Nano problem. I just registered an account here. My name is Jasper. Thank you.

Hi, sorry for late response. It might work fine, but anyway you need to calibrate it for your voltage. Try to use this article: http://www.pocketmagic.net/sharp-gp2y1010-dust-sensor/

especially: adcVoltage = adcValue * (3.3 / 1024);

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #6

I've performed some measurements of output signan to ILED and found that all Arduino programs that I've found on internet have incorrect sample timing.

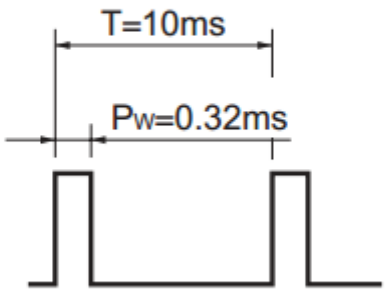
Typical code is:

```
digitalWrite(PIN_LED, HIGH); // power on the LED

delayMicroseconds(280);

int analogData = analogRead(SENSOR_PIN);

delayMicroseconds(40);

digitalWrite(PIN_LED, LOW); // power off the LED

delayMicroseconds(9680);
```

280 + 40 + 9680 = 10000

Excelent according to sensor specification

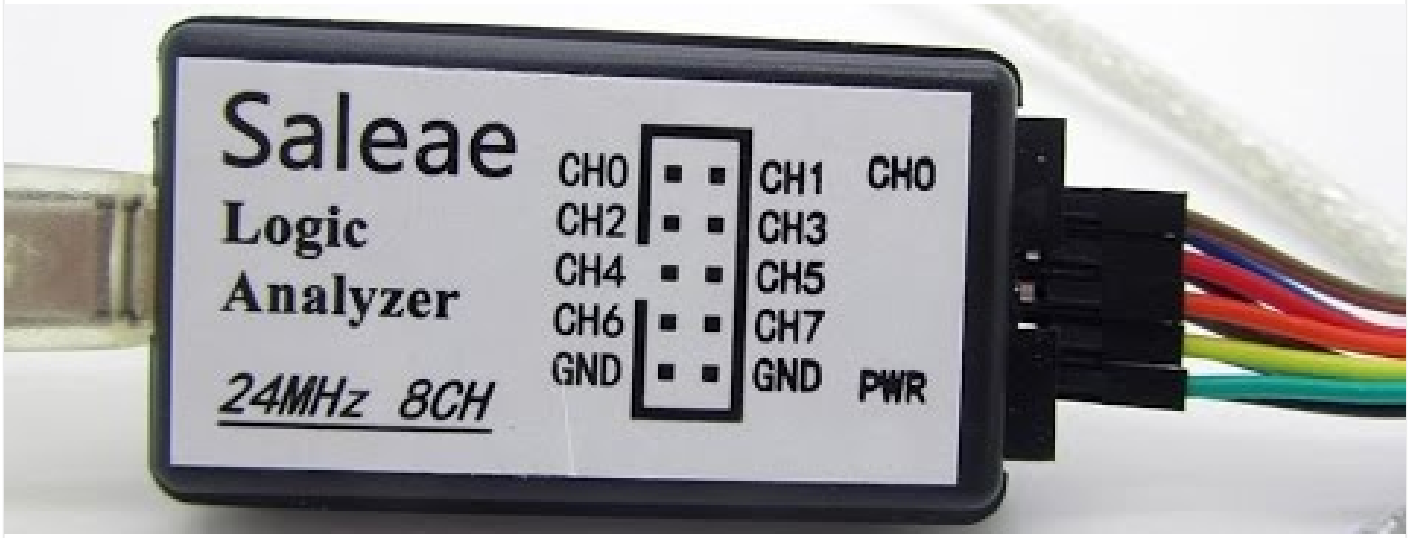
Pulse-driven wave form



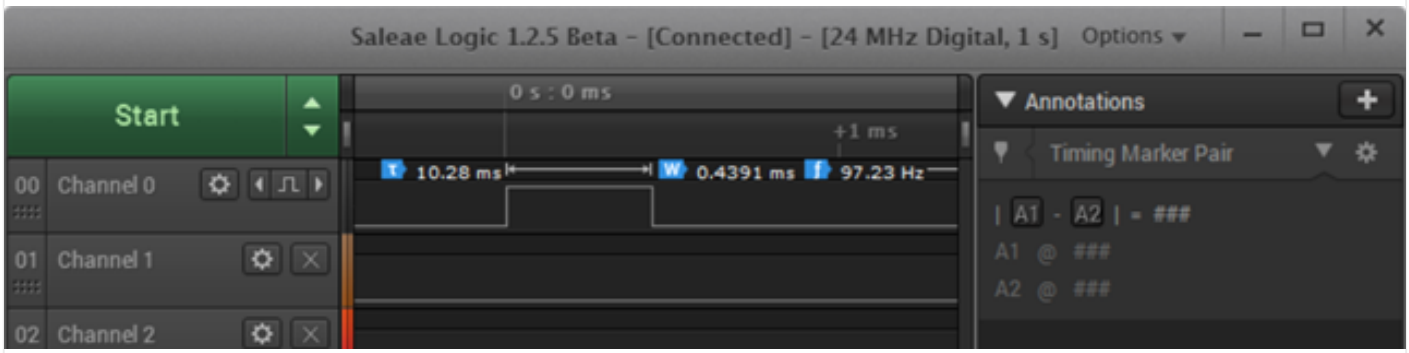
But no one takes to consideration that analogRead is slow operation, that lasts for about 10 microseconds. Other operations also takes some time.

So, I've decided to check real pulse timings.

I've had Salae Logic usb digital osciloscope, that can measure impulse lenght, and I've performed several tests.



That code above gives following picture in analyzer program window:



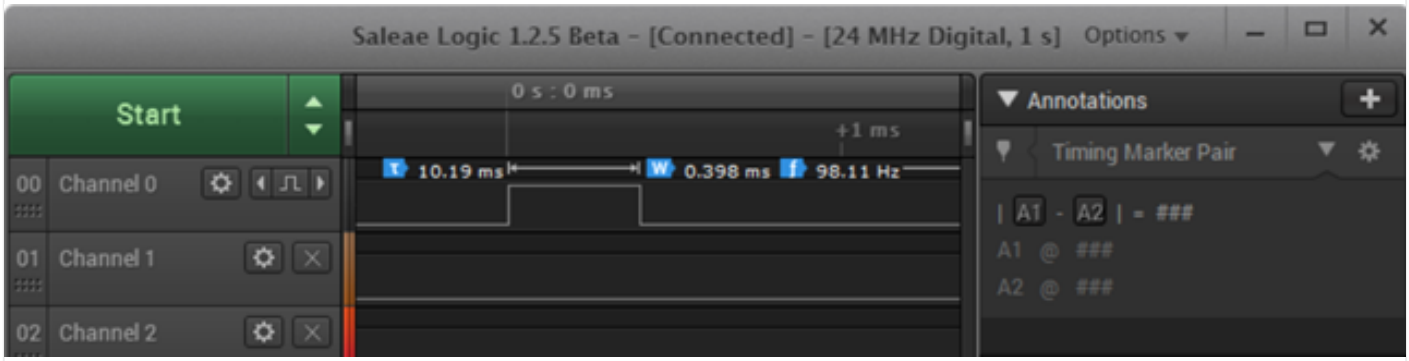
Total period T = 10.28ms

ILED light time = 0.4391ms instead 0.32

As you see it is more than expected. If we calculate how much takes analogRead = 10280 - 10000 = 280 microseconds.

So I've decided to remove 40 microseconds delay befor switch-off

```
digitalWrite(PIN_LED, HIGH); // power on the LED

delayMicroseconds(280);

int analogData = analogRead(SENSOR_PIN);

delayMicroseconds(40);

digitalWrite(PIN_LED, LOW); // power off the LED

delayMicroseconds(9680);
```



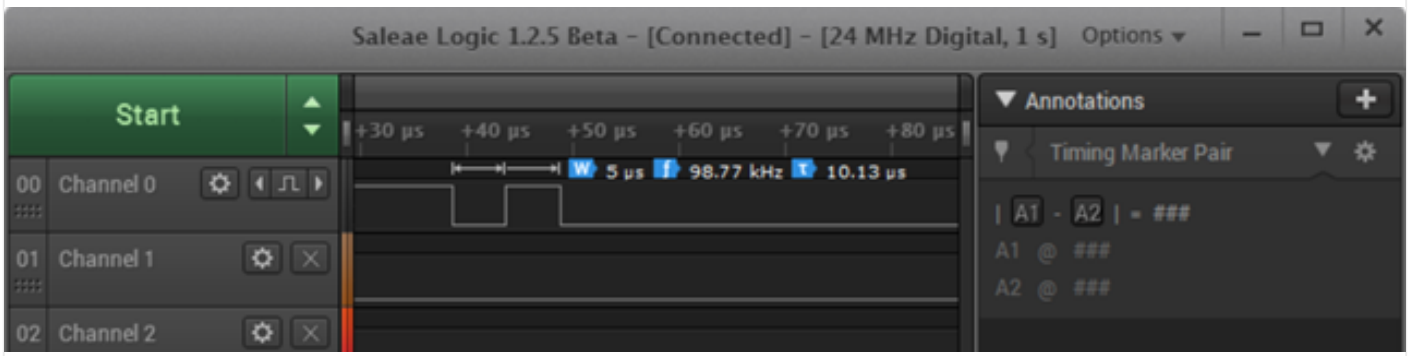
New walues are slightly better

Total period T = 10.19ms

ILED light time = 0.398 ms instead 0.32

Finaly, to calculate how much time is needed to switch-on/off ILED I've calculated following code:

```
digitalWrite(PIN_LED, LOW); // power off the LED

digitalWrite(PIN_LED, HIGH); // power on the LED

digitalWrite(PIN_LED, LW); // power off the LED
```



So, each operation takes 0.05ms

After all I’ve decreased samples sleep delay time to 9500 to have period time close to 10ms.

https://github.com/vlytsus/arduinosensor

#GP2Y1010AU0F #Arduino #dustsensor

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #5

I’ve decided to add LCD dispaly to the project to show calculated value. In this case dust sensor will be autonomous. I’ve used 16x2 LCD display HD44780. To handle display printing I’ve used LiquidCrystal library. By default enabled only serial printing: #define SERIAL_PRINT true. If you would like to enable LCD printing please set #define LCD_PRINT true.

Since I’ve got a lot of noise during measurements, I decided to filter it out from final results. It is performed by calculating median over 5 samples, and from that value is removed maximum & minimum results (avgDust - maxDust - minDust) / (SAMPLES_PER_COMP - 2). But this is not the end. To display data more smoothly I’ve added one more filter - stack[100] to store already filtered data and calculate median once more, over 100 samples.
I still was not happy with result’s that I’ve achieved. My sensor was able to show hazardous concentrations of dust, but calculated dust veight values ug/m3 was far from data that I’ve got from official sensors, published in internet. I’ve found very usefull article about similar project, performed by Matthias Budde, Mathias Busse, and Michael Beigl from Karlsruhe Institute of Technology
http://www.teco.edu/~budde/publications/MUM2013_budde.pdf
I was on the right direction. Now I need to perform additional calibration and include ambient temperature to calculations.

I’ve fixed several bugs and improved dust calculation formulas. All code sources and last chnages please check at: https://github.com/vlytsus/arduinosensor

```
/*  ********************************************************** *
************ GP2Y1010AU0F Scharp Dust Sensor ***************** *
************************************************************ *  Dust
sensor calculates dust density based on reflected  *  infrared light from IR
diode. Light brightness coresponds  *  to amount of dust in the air.
Following program is responsible *  to light-up IR diode, perform dust
sampling and swith-off diode, *  according to GP2Y1010AU0F specification.  *
Arduino program performs sequence of several measurements,  *  filters input
data by mid point calulation based on several  *  measurements, to avoid
voltage spikes. Then it performs dust  *  values transformation to ug/m2.
Finaly calculated data could  *  be printed to Arduino serial output or to
LCD display. *
************************************************************ *  AQI Index
could be calculated as *  ------------------------------- *  PM2.5 ug |
AQI   |  Pollution *  =========|=========|============ *     0-35  |  0-50
|  Excelent *  ---------|---------|------------ *    35-75  | 51-100  |
Average *  ---------|---------|------------ *    75-115 | 101-150 |  Light *
---------|---------|------------ *   115-150 | 151-200 |  Modrate *  -------
--|---------|------------ *   150-250 | 201-300 |  Heavy *  ---------|------
---|------------ *   250-500 |  > 300  |  Serious *  ---------|---------|---
--------- */#include <LiquidCrystal.h>#include <stdlib.h>// initialize the
library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define ADC_BASE_V  1100.0    //5000 //mv  4500 - real USB voltage#define
MICROGR_2_MLVOLT_RATIO    0.2     //ug/m3 / mv#define ADC_RESOLUTION
1023.0#define PIN_LED          7#define PIN_ANALOG_OUT   0#define
POWER_ON_LED_DELAY   280//#define POWER_ON_LED_SLEEP 40 // not used, digital
read takes about 100 microseconds#define POWER_OFF_LED_DELAY  9500#define
SENSOR_PIN           0#define DISPLAY_REFRESH_INTERVAL 30#define
SAMPLES_PER_COMP     5#define STACK_SIZE           100#define
MAX_UNSIGNED_INT     65535#define LCD_PRINT       true#define SERIAL_PRINT
true#define RAW_OUTPUT_MODE false // if true then raw analog data 0-1023
will be printed// Additional correction after calibration // to calculate
dust as ug/m3// by minimum & maximum values.// According to specification:
min=600mv; max=3600mv// using linear function: y = a*x + b;// here x is
voltage = ADC_val * V_adc_base / ADC_resolution#define A_CORRECTION
2.36#define B_CORRECTION -76
float a_correction = A_CORRECTION * (ADC_BASE_V / ADC_RESOLUTION);

unsigned int stack[STACK_SIZE+1];// stack is used to calculate middle value
for display output
unsigned int stackIter; // current stack iteration
unsigned int refresh; // current display counter, used to not print data too
frequently

char str_temp[6];

void setup() {

  if(LCD_PRINT){
    // set up the LCD's number of columns and rows:
    lcd.begin(16, 2);
    // Print a message to the LCD.
    lcd.print("-=Dust+Sensor=-");
  }
```

```
  if(ADC_BASE_V < 4000)
    analogReference(INTERNAL);

  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_ANALOG_OUT, INPUT);
  digitalWrite(PIN_LED, LOW);

  stackIter = 0;
  refresh = 0;
  for(int i = 0; i < STACK_SIZE ; i++){
     stack[i] = 0;
  }

  if(SERIAL_PRINT)
    Serial.begin(9600);
}

void loop(void){

   if(stackIter >= STACK_SIZE)
     stackIter = 0;

   stack[stackIter] = computeSensorSequence();

   if(refresh < DISPLAY_REFRESH_INTERVAL){
     refresh++;
   }else{
     refresh = 0;
     //calculate midpoint value and print
     int yResult = a_correction * calculateStackMidVal() + B_CORRECTION;
     if(yResult > 0){
      print(yResult);
     }
   }

   stackIter++;
}

unsigned int computeSensorSequence(){
  //perform several measurements and store to stack//for later midpoint
calculations

   unsigned int maxDust = 0;
   unsigned int minDust = MAX_UNSIGNED_INT;
   unsigned long avgDust = 0;
   unsigned int dustVal = 0;

   //perform several sensor reads and calculate midpoint
   for(int i = 0; i< SAMPLES_PER_COMP; i++){
     dustVal = readRawSensorData();
     if (dustVal > 0){

      //find max dust per sample
      if(dustVal > maxDust)
        maxDust = dustVal;
```

```
      //find min dust per sample
      if(dustVal < minDust)
        minDust = dustVal;
        avgDust += dustVal;
     }
   }

   //filter input data//don't take to consideration max & min values per
sample//and save average to stack
   return (avgDust - maxDust - minDust) / (SAMPLES_PER_COMP - 2);
}

unsigned int calculateStackMidVal(){
   int midVal = 0;
   for(int i = 0; i < STACK_SIZE ; i++){
     midVal += stack[i];
   }
   return midVal / STACK_SIZE;
}

unsigned int readRawSensorData(){
  unsigned int analogData; //ADC value 0-1023
  digitalWrite(PIN_LED, HIGH); // power on the LED
  delayMicroseconds(POWER_ON_LED_DELAY);
  analogData = analogRead(SENSOR_PIN);
  //delayMicroseconds(POWER_ON_LED_SLEEP);//not used, digital read takes
about 100 microseconds
  digitalWrite(PIN_LED, LOW); // power off the LED
  delayMicroseconds(POWER_OFF_LED_DELAY);//9500
  return analogData;
}

void print(int val){
  print(String(val));
}

void print(String msg){
  if(SERIAL_PRINT)
    Serial.println(msg);
  if(LCD_PRINT){
   lcd.setCursor(0, 1);
   lcd.print(msg);
  }
}
```

#GP2Y1010AU0F #dustsensor #Arduino #pollution #PM2.5

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #4

It’s time to get payload from data that we’ve received from dust sensor. There are a lot of controversial ideas about how to calculate dust particles weight per volume, based on air transparency and measured light from infrared diode. We should take to consideration several factors that could lead to incorrect results during measurements and try to eliminate them or add corrections to formula, that will be used to calculate dust density.

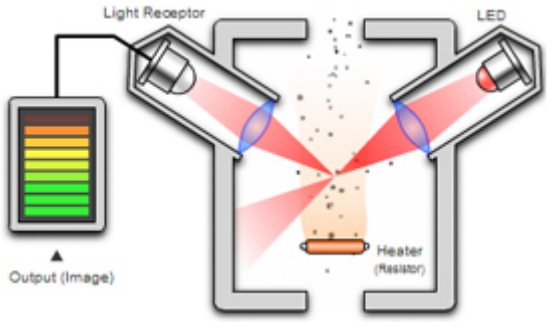
Voltage

Arduino will perform calculations based on measured voltage that comes to analog input. According to specification Arduino board & dust sensor are powered by 5V, but we are living in not ideal world. The USB 1.x and 2.0 specifications provide a 5 V supply on a single wire to power connected USB devices. The specification provides for no more than 5.25 V and no less than 4.75 V (5 V ± 5%) between the positive and negative bus power lines. That could lead to significant errors in calculations.

The sensor lower voltage output if there is no dust is specified from 0V to 1.5V. The 0V is more of a problem, as the ADC’s often don’t operate close to ground values.

Humidity

Since sensor is based on photoelectric effect from dust particles, that appears in focus of photo-resistor & reflected infrared diode light, dust sensor could calculate water fog particles as pollution. Also fog could condensate on lenses, that also could lead to decrease quality of results.



Sun light

Some developers mentioned that direct sun light has impact on measurement results. Possible that caused by light that hit to lenses or dust particles thru the sensor hole.

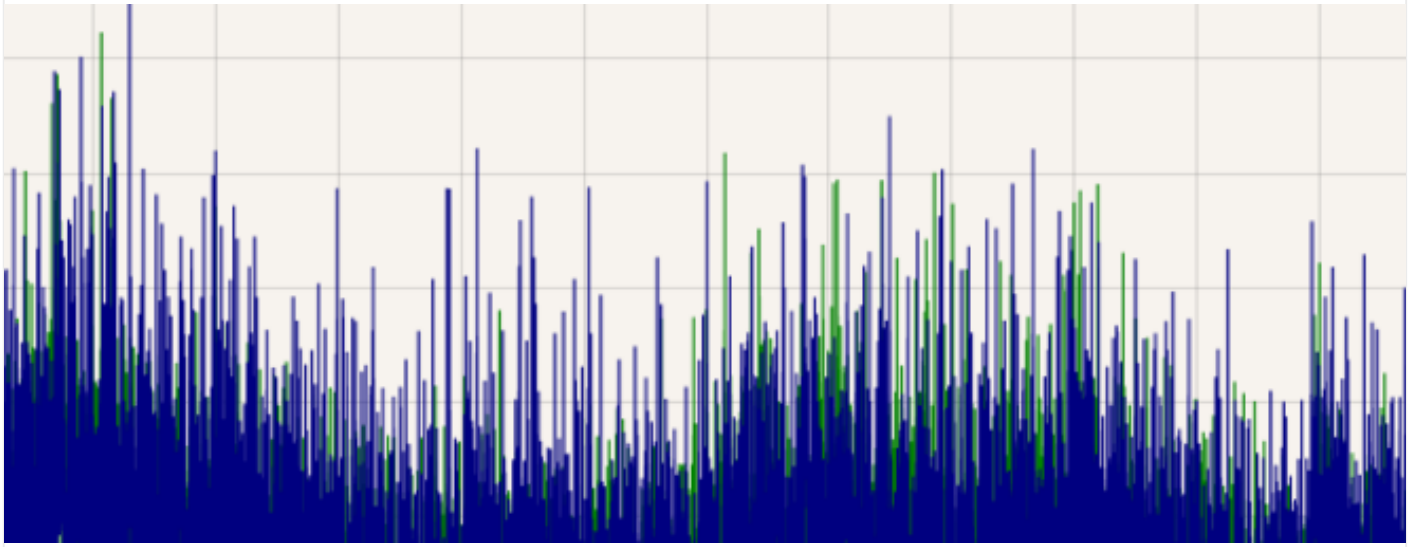
Temperature

Despite dust sensor specification promises that it would work in different temperature conditions from −10 to +65 °C, but according to experiment results with open/closed sensor hole, it calculates different values depending on ambient temperature. Also I've observed significant pollution data right after dust sensor is moved from cold to warm area. It could be caused by water fog condensation on cold lenses. For example see what will happen with your lenses if camera will be long time on cold.

Particles size

During my experiments I saw significant spikes in results. I've tried to perform some basic math results filtering to eliminate spikes data if there is significant difference from calculated medium value. But I am not sure that I was right, when decided to eliminate such spikes from final

calculations. Possible that huge dust particles that appears in infrared beam focus could cause such spikes. If that is true - we shouldn't remove them from results and consider as pollution. But from other side, huge particles are less hazardous, so, I have no final decision here. I am going to perform more measurements producing big particles from rug dust and fine particles from cigarette smoke to compare amount of spices in per each pollution type. Maybe in final solution I will add some kind of lightweight HEPA filter, to measure only PM2.5 particles and ignore huge dust particles.



Sampling rate

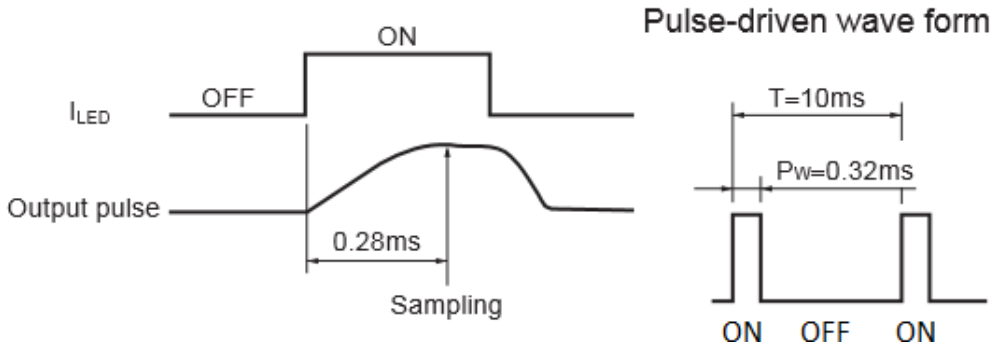
According to GP2Y1010AU0F datasheet we should follow 10ms sample rate to get correct measurement results. Possible it dictated by sensor self RC parameters, that could cause significant impact on infrared diode light brightness or photo-resistor currents. I've saw the same Arduino code samples in many internet articles, that performs reading from analog input and does 40 microseconds delay after that, to meet sensor specification.

```
delayMicroseconds(280);
DUST = analogRead(SENSOR_PIN); // read data
delayMicroseconds(40);
```

But probably performing all those developers haven't taken to consideration one fact, that Arduino takes about 100 microseconds to read an analog input.

https://www.arduino.cc/en/Reference/AnalogRead

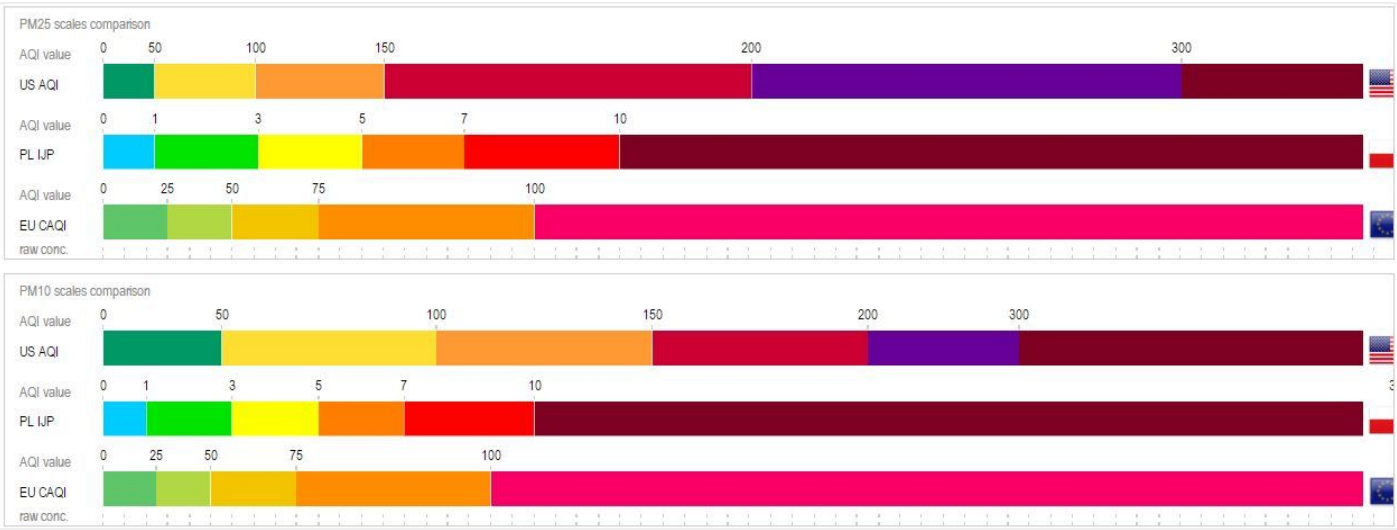
So probably this peace of code is incorrect: delayMicroseconds(40);



So we've reviewed several aspects that we need take to consideration during analog input interpretation.

More about real working code will be in next **Part #5.**

#dustsensor #Arduino #PM2.5 #pollution #GP2Y1010AU0F



Open large image

US EPA's AQI and Europe (Poland) IJP comparison

Because graphics are better than long sentences, the above graphics is a compariosn of the AQI and IJP scales for PM2.5 and PM10.

Values are presented in mg/m3.

Please refer to the source page at http://aqicn.org/faq/2015-09-03/air-quality-scale-in-poland/

Legend

	Air quality	Recommendation
0-1	Very good (Excellent) Bardzo dobry	The air quality is good. The air pollution pose no threat. The conditions ideal for outdoor activities.
1-3	Good Dobry	Air quality is still good. The air pollution pose minimal risk to exposed persons. Conditions very good for outdoor activities.
3-5	Moderate Umiarkowany	Air quality is acceptable. Air pollution can endanger people at risk. Conditions good for for outdoor activities.
5-7	Satisfactory Dostateczny	Air quality is average. The air pollution pose a threat for people at risk * which may experience health effects. Other people should limit spending time outdoors, especially when they experience symptoms such as cough or sore throat.
7-10	Bad Zły	Air quality is bad. People at risk * should avoid to go outside. The rest should be ograniczyć.Nie are recommended for outdoor activities.
10+	**Hazardous** **Bardzo zły**	The quality of air is dangerously wrong. Those at risk should be avoided to go outside. The others should limit the output to minimum.Wszelkie outdoor activities are discouraged.

#AQI #Pollution #PM2.5

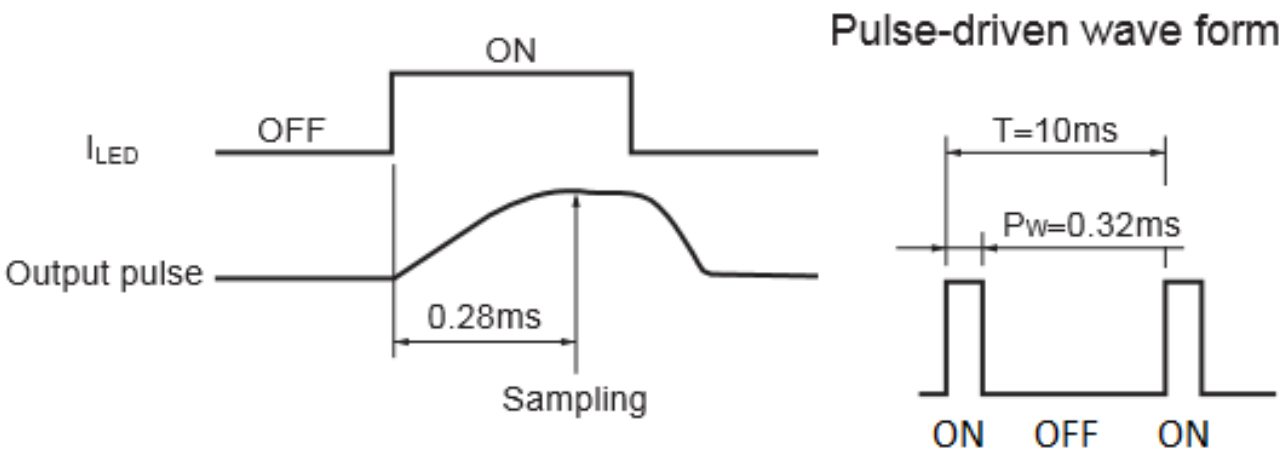
1 note

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #3

In this part we will review simple Arduino program that is designed to read analog signals from Arduino A0 port and converts them to dust pollution level in milligrams per square meter. That program is far from final solution, but is good to have basic understanding of data sampling.

Control principle

According to specification for GP2Y1010AU0F



we need to perform following steps to do sampling measurement:

1. Enable the internal infrared emitting diode by setting the pin ILED to HIGH.
2. Wait 0.28ms, then the external controller starts to sample the voltage from the pin AOUT of the module. Notes that the output wave will take 0.28ms to reach steady state after the internal infrared emitting diode is enabled.
3. There is a period of 0.04ms for sampling. When finished, set the pin ILED to LOW to disable the internal infrared emitting diode.
4. Calculate the dust concentration according to the relationship between output voltage and dust concentration.

Arduino program

Lets create Arduino program to perform that steps:

We assume that ILED diode is connected to Digital output D7 and Analog output from sensor is connected to A0. So, we could configure it and switch off at program start:

```
#define PIN_LED   7
#define SENSOR_PIN 0
pinMode(PIN_LED, OUTPUT);
digitalWrite(PIN_LED, LOW);
```

Now, during processor cycle we should turn on diode, wait 280 microseconds to warm-up, read voltage value from A0, wait 40 microseconds and switch-off the diode. After that we should sleep till next operation.

```
digitalWrite(PIN_LED, HIGH); // power on the LED
delayMicroseconds(280); // wait to warm-up
DUST = analogRead(SENSOR_PIN); // read data
delayMicroseconds(40);
digitalWrite(PIN_LED, LOW); // power off the LED

Serial.print( DUST );
delayMicroseconds(9680); // sleep till next iteration
```

That looks simple! But our printout will show some voltage data, that should be additionally calculated to mg/m3, otherwise it is hard to understand current pollution level.

After that calculation we could use following table as a reference to get pollution level: **US and Europe air quality standards**

http://arduinosensor.tumblr.com/post/134273104325/open-large-imageus-epas-aqi-and-europe-poland

So in next post I am going to more aspects related to pollution level calculations.

Go to Part # 4

#GP2Y1010AU0F #Arduino #Dust Sensor #PM2.5 #Pollution #dustsensor

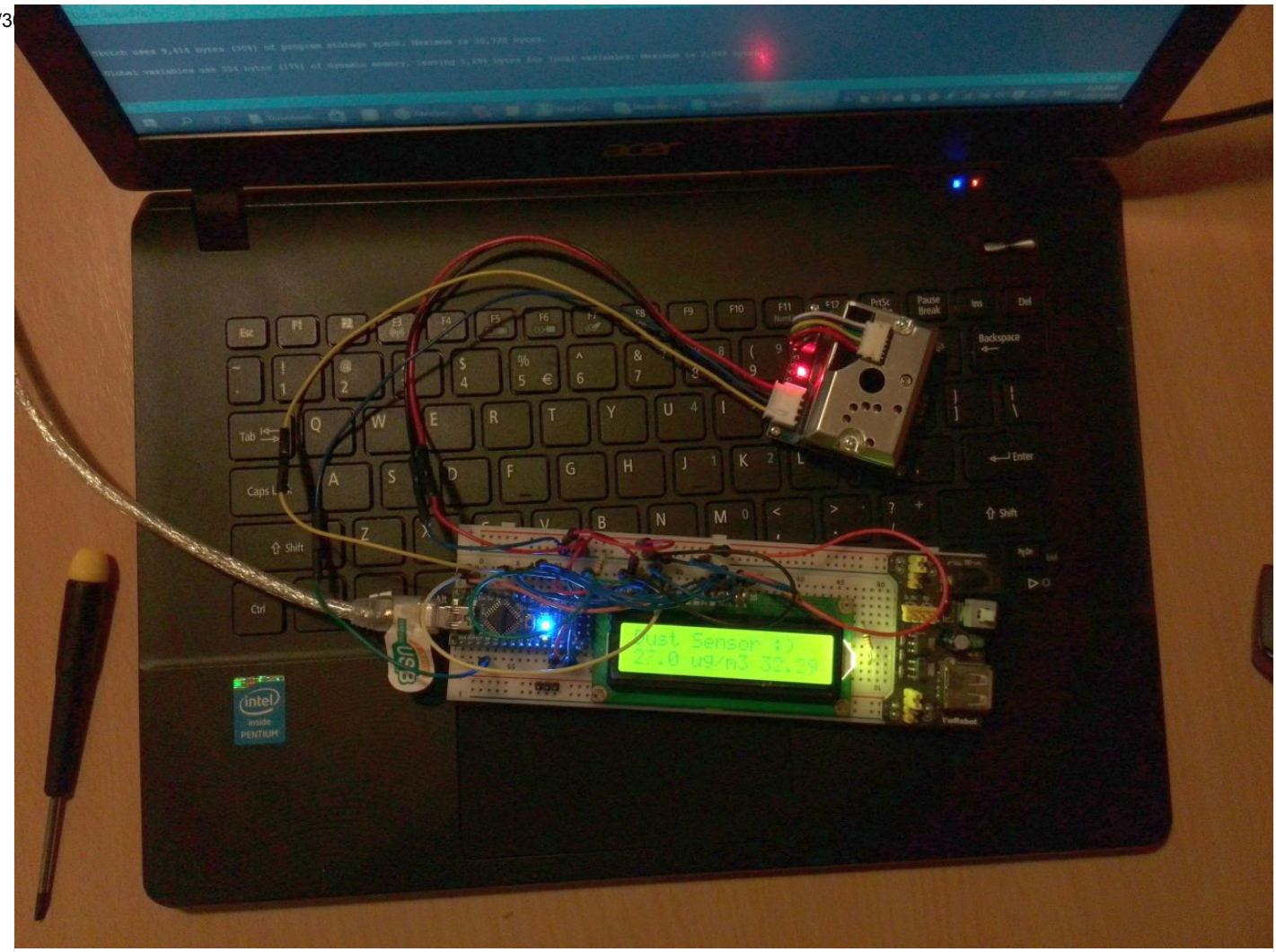
1 note

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #2

In previous **Part #1** I've described basics of Sharp Optical Dust Sensor. Now it is time to connect it to Arduino and run the code, to read sensor data and convert it to useful format.

I've bought cheap and small Arduino Nano board, breadboard and bunch of wires because I don't like to spend time with soldering. I am going to build prototype, and later create smaller solution with LCD panel, some buttons to change modes and batteries pack.

I've got following solution, that could show some dust measurement results. I've spend a lot of time to proceed data, since that sensor produces analog signal of optical air measurement, and it should be converted to some useful units like milligrams per square meter.



As I mentioned in previous Part #1 my goal is to measure PM2.5 & PM10 pollution. So 25 µg/m^3 of PM2.5 is considered as 100% pollution value. And I want to know when level is more than 300% which is considered as unhealthy conditions and it is better to stay at home and not open the windows.

Since PM2.5 is more dangerous, I would like to be able to measure PM2.5 & PM10 separately. But unfortunately GP2Y1010AU0F is unable to distinguish between different particles and shows general pollution level.

So, lets review how to connect Sharp Optical Dust Sensor to Arduino Nano board and show data on computer, connected to Arduino Nano via USB cable. If you have some LCD display then we also could use that to show data and have mobile solution.

Components:

- Arduino Nano board
- Sharp Optical Dust Sensor GP2Y1010AU0F
- Waveshare Dust Sensor connection board
- Breadboard
- Jumper cables
- 16x2 LCD Display (optional)

Fig 1 - Arduino Nano board

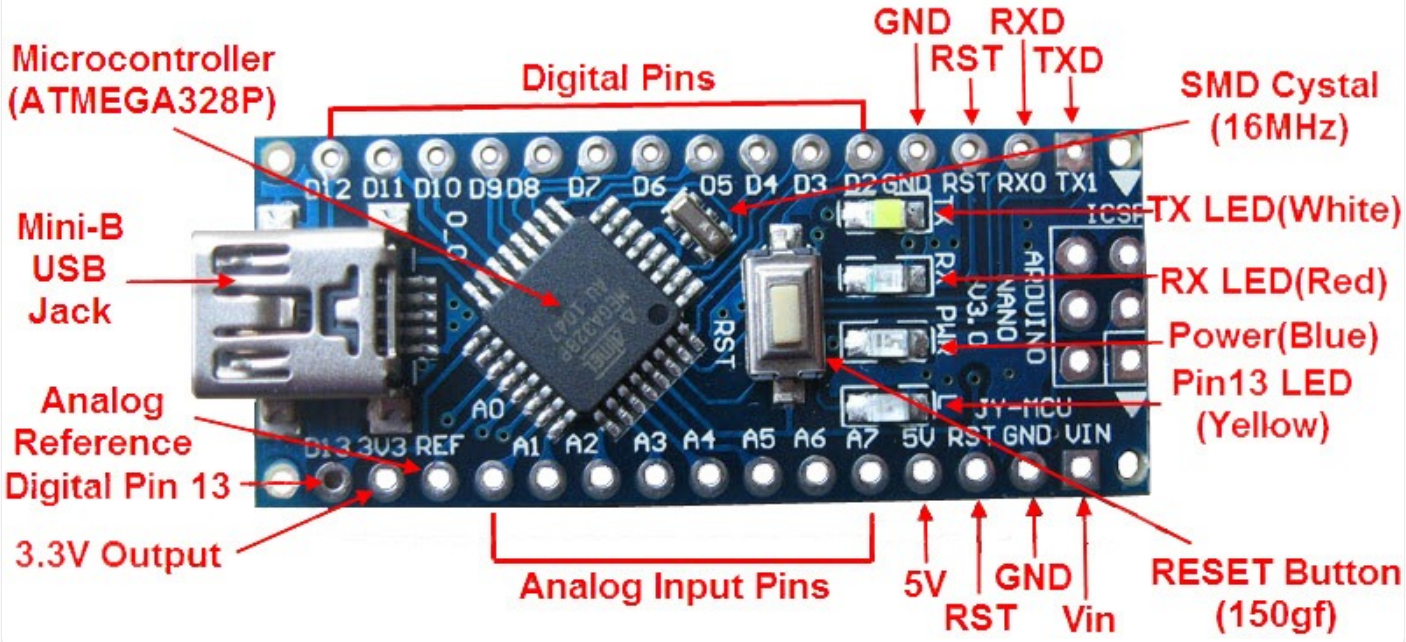
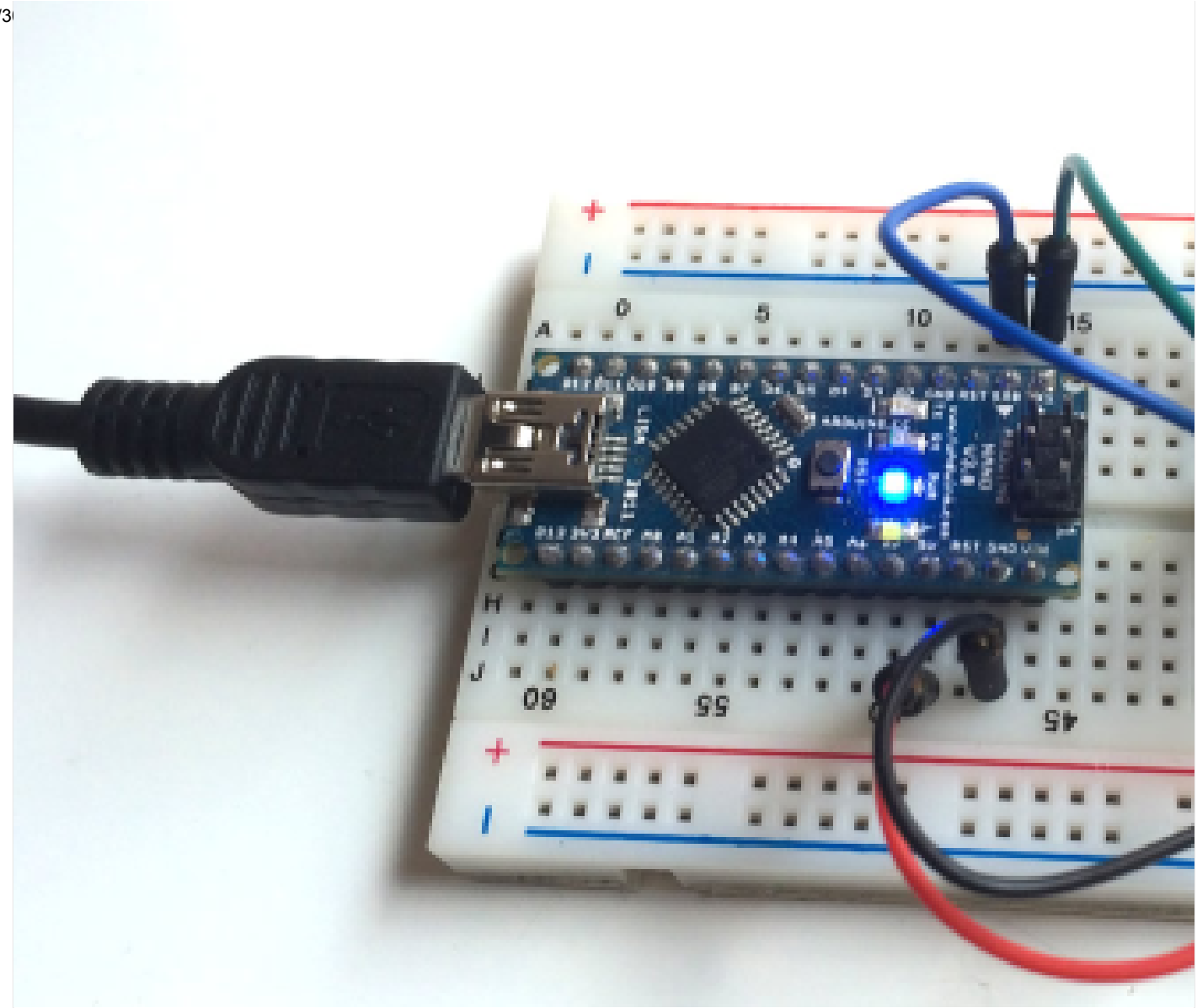


Fig 2 - Waveshare Dust Sensor board with connected GP2Y1010AU0F

Fig 3 - Breadboard



Sensor connection

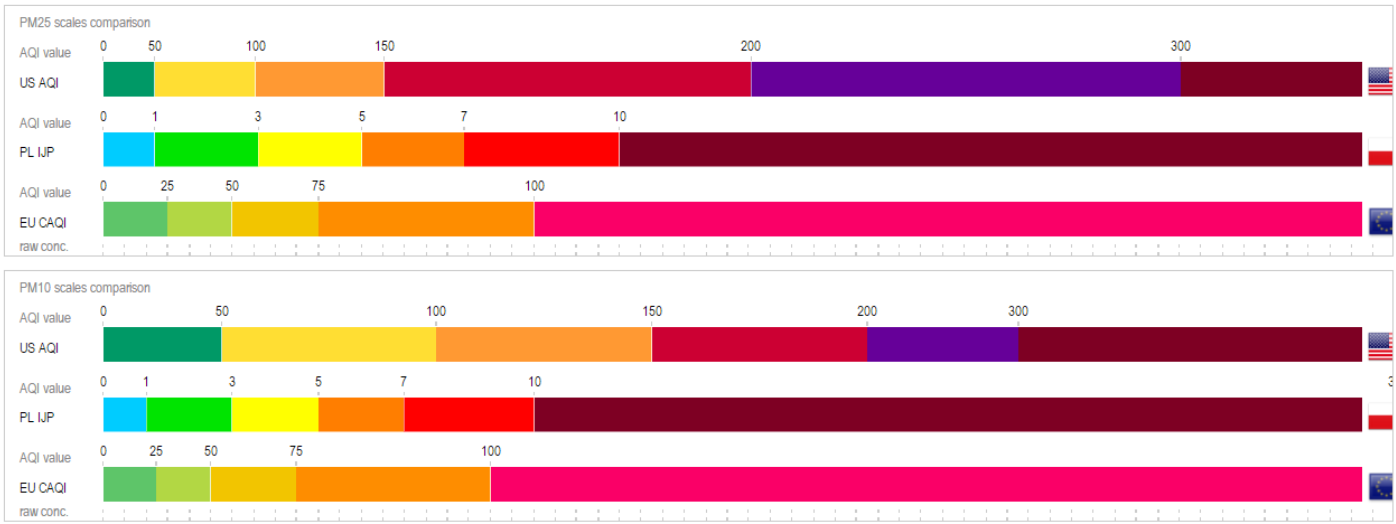
Waveshare Sensor Board has 4 pin connector and is very easy to connect it to Arduino.

- Pin 1 VCC is connected to Arduino 5V pin.
- Pin 2 GND is connected to any Arduino ground pin.
- Pin 3 AOUT (Analog Output) is connected to Arduino A0 pin.
- Pin4 ILED (Infrared LED) is connected to Arduino D7 pin

Whole system is powered by mini USB connector that is located on Arduino Nano board. Waveshare Dust Sensor board has red LED, that lights if power is connected to VCC.

In next Part #3 we will review Arduino program that is designed to read analog signals from Arduino A0 port.

Later we will convert analog data to pollution levels based on US and Europe standards.



Open this chart as large image in another post…

Go to next Part #3: **Arduino PM2.5 Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F**

Part #3

#GP2Y1010AU0F #Arduino #Dust Sensor #PM2.5 #dustsensor

2 notes

Arduino PM2.5 / PM10 Pollution Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F Part #1

Atmospheric particulate matter – also known as **particulate matter** (**PM**) or **particulates** – is microscopic solid or liquid matter suspended in the Earth's atmosphere. They have impacts on climate and precipitation that adversely affect human health. The size of the particle is a main determinant of where in the respiratory tract the particle will come to rest when inhaled. Larger particles are generally filtered in the nose and throat via cilia and mucus, but particulate matter smaller than about 10 micrometers, referred to as PM_{10}, can settle in the bronchi and lungs and cause health problems. $PM_{2.5}$ leads to high plaque deposits in arteries, causing vascular inflammation and atherosclerosis – a hardening of the arteries that reduces elasticity, which can lead to heart attacks and other cardiovascular problems.

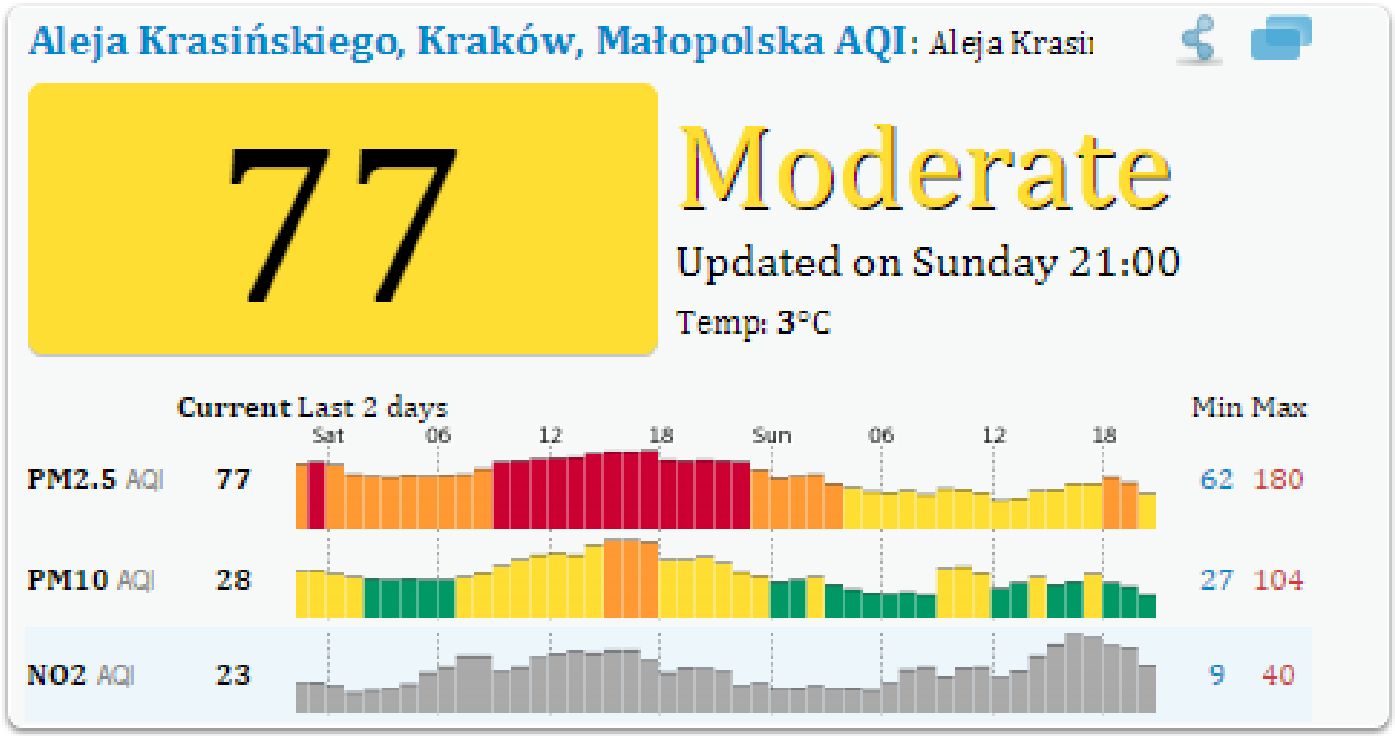
The European Union has established the European emission standards which include limits for particulates in the air:

- PM10 - Yearly average 40 μg/m^3

- PM2.5 - Yearly average 25 µg/m^3

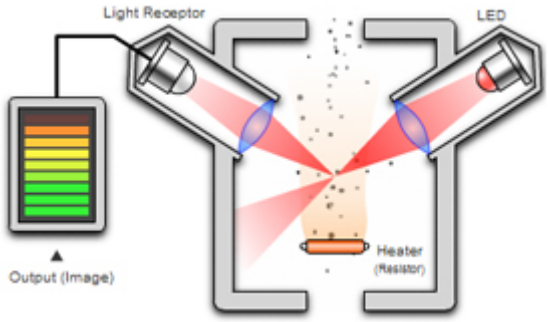
I am living in Krakow - Poland, that is quite polluted city, especially by old-style charcoal heaters around the city and suburbs. In warm period there is no problem with pollution, it is around 60% for PM2.5 during the year. But in winter time pollution level could reach 800% and above.

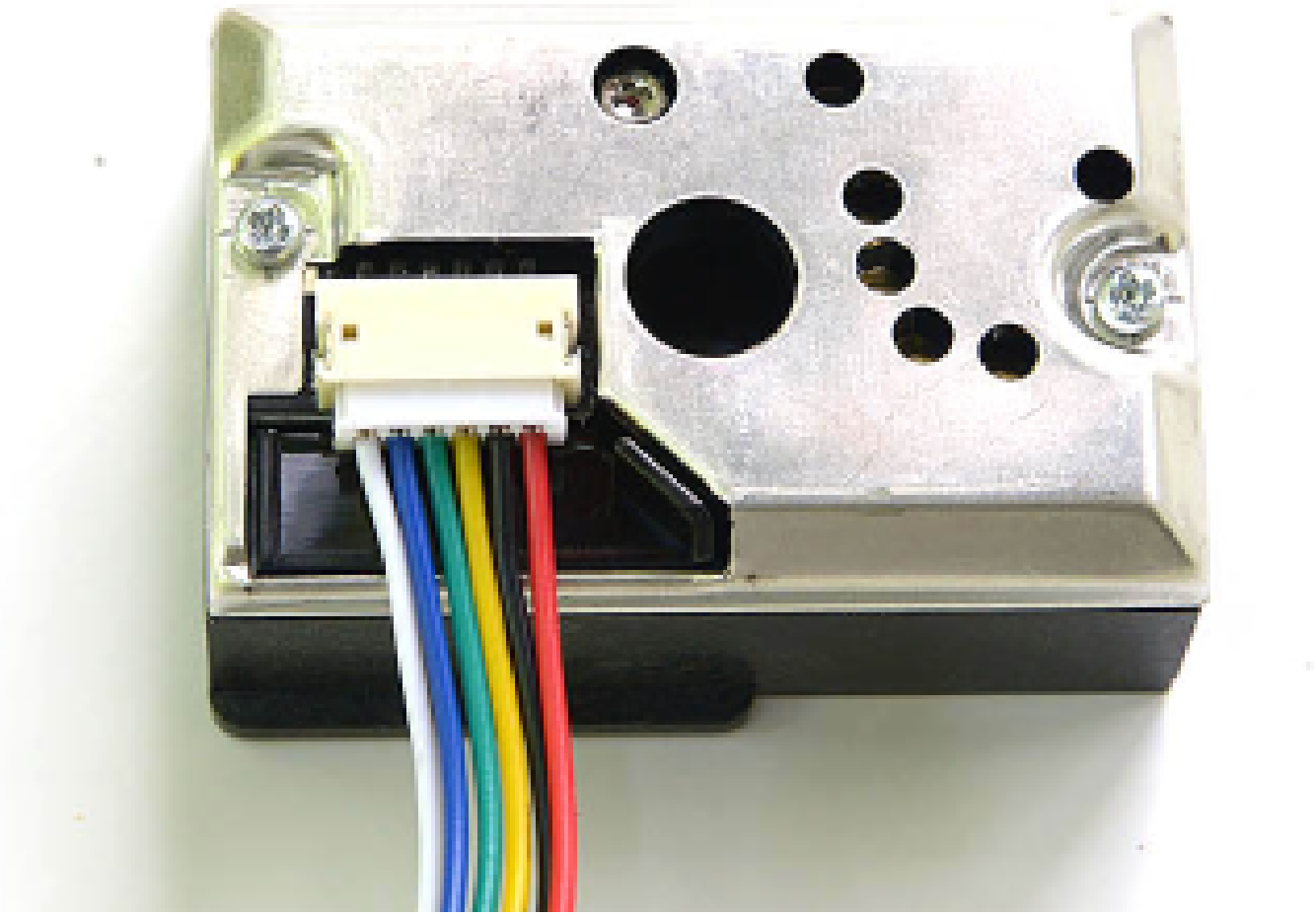
In Krakow there are only 3 PM2.5/PM10 sensors that could be observed thru internet at http://aqicn.org/city/krakow.



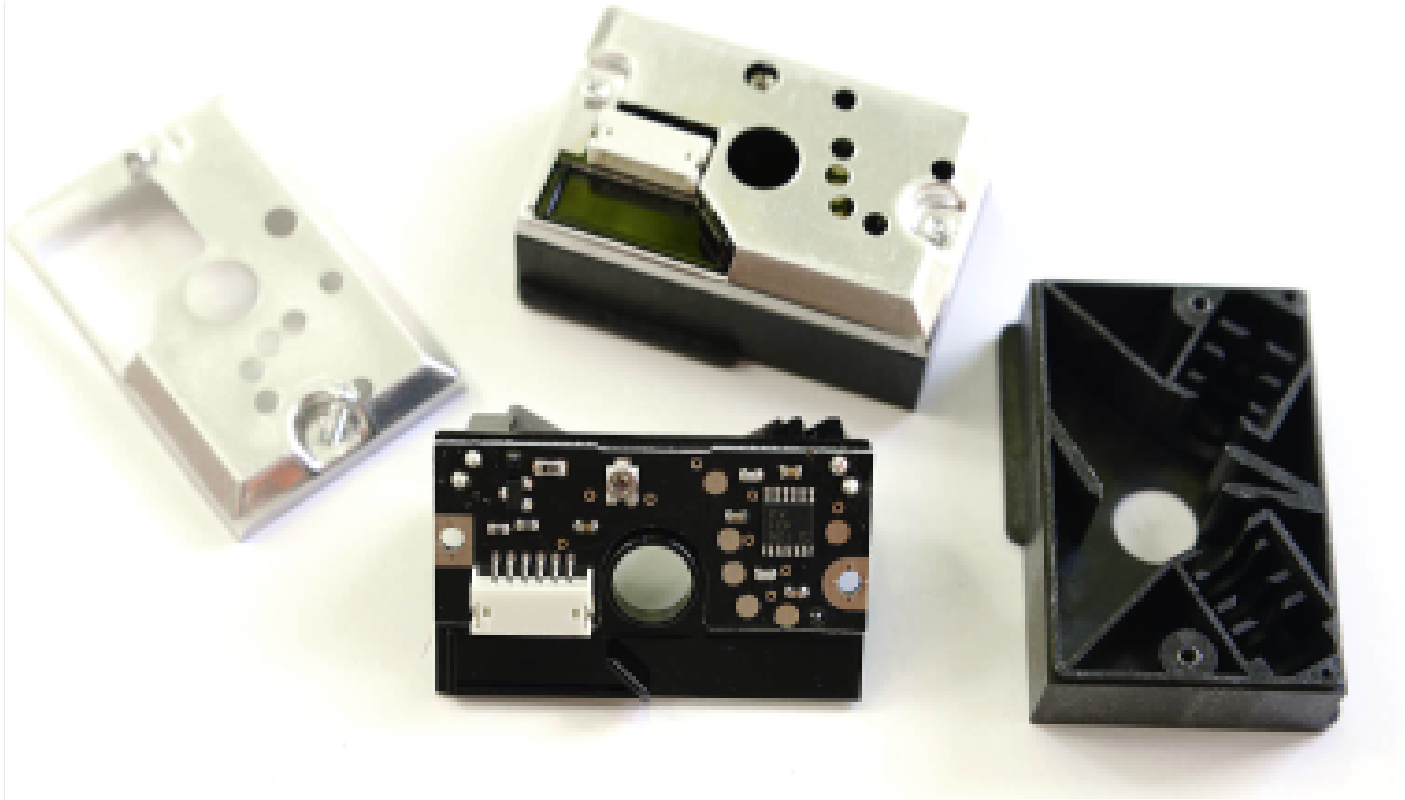
And they are far from my apartment, so I don't know real level in my area. I've decided to build my own sensor, since I don't like to keep window open or go ride bicycle at time when it is highly polluted outdoor.

I've researched several articles about Arduino based DIY sensors and decided to start building my own solution using cheap and well known sensor from Sharp - GP2Y1010AU0F, that is an optical air quality sensor, designed to sense dust particles. An infrared emitting diode and a photo-transistor are diagonally arranged into this device, to allow it to detect the reflected light of dust in air. It is especially effective in detecting very fine particles like cigarette smoke, and is commonly used in air purifier systems. It generates an output voltage proportional to dust density.

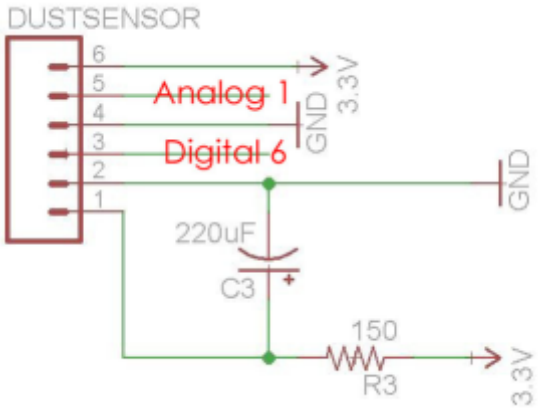


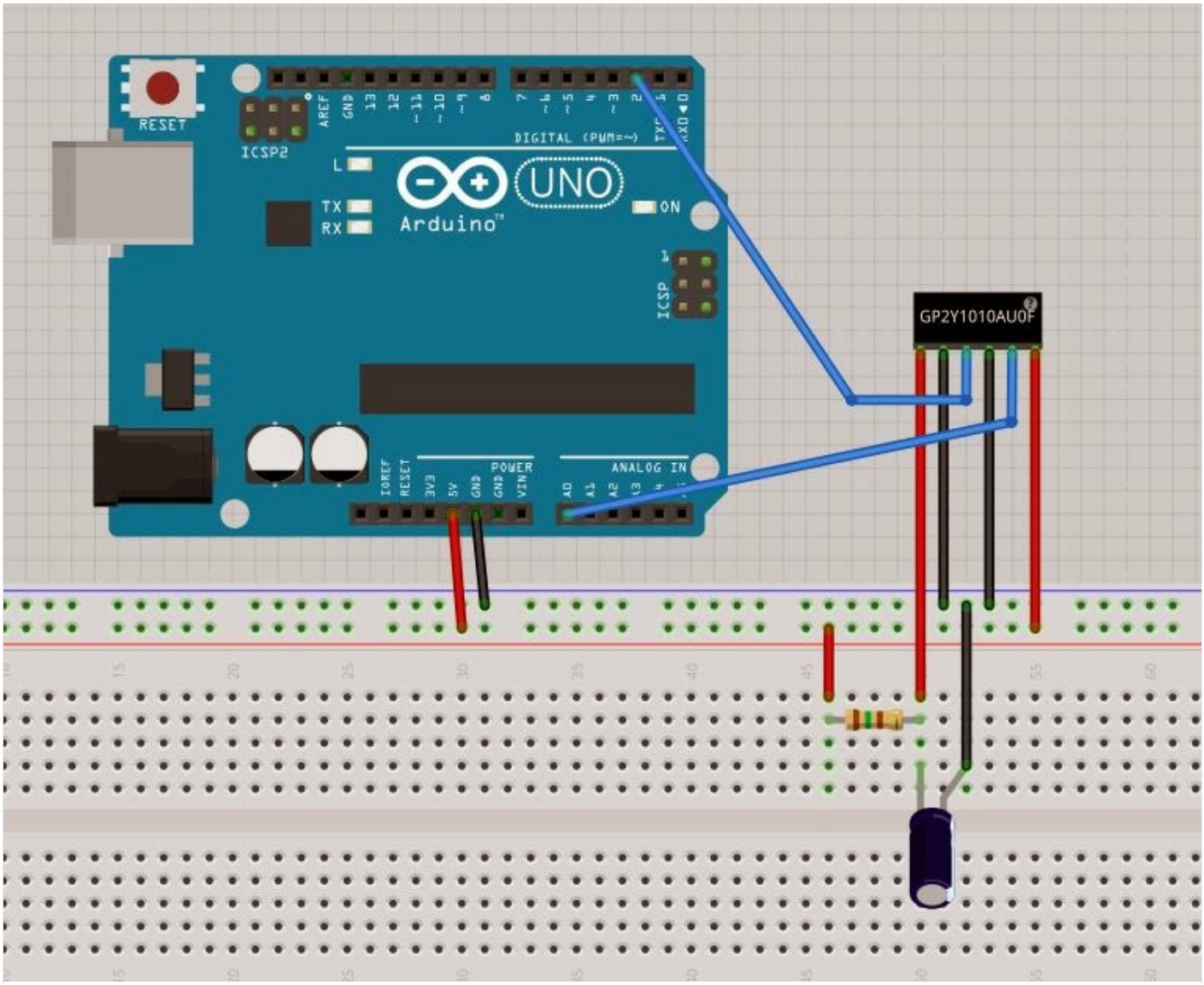


There are some lenses inside, that could be vacuumed thru the hole, after some time, in case if dust appeared inside.



To connect it to Arduino you also need 220 uF capacitor and 150 Ohm resistor.





An important thing about the code in case if you will connect sensor to Arduino using schema above is that in order to turn the Infrared LED on, you have to digitalWrite LOW on LED input terminal. Because the transistor that opens the drain is PNP type of transistor which needs negative base voltage.

But also there are Arduino kits on market, that already have small board with some elements and connectors, to support Arduino. So you don't need any additional details or soldering. You also could use HIGH signal on digitalWrite to turn the led on and LOW - to switch off. I found it useful, since that board support different voltage and has some circuits to improve connection & signal.

It costs about 15$

Next **PART #2** will consider aspects of connecting and programming that dust sensor to Arduino Nano board.

Go to next part: **Arduino PM2.5 Sensor based on Sharp Optical Dust Sensor GP2Y1010AU0F**

Part #2

#GP2Y1010AU0F #Dust Sensor #Arduino

2 notes