# PM2.5 laser dust sensor SKU: SEN0177

## Contents

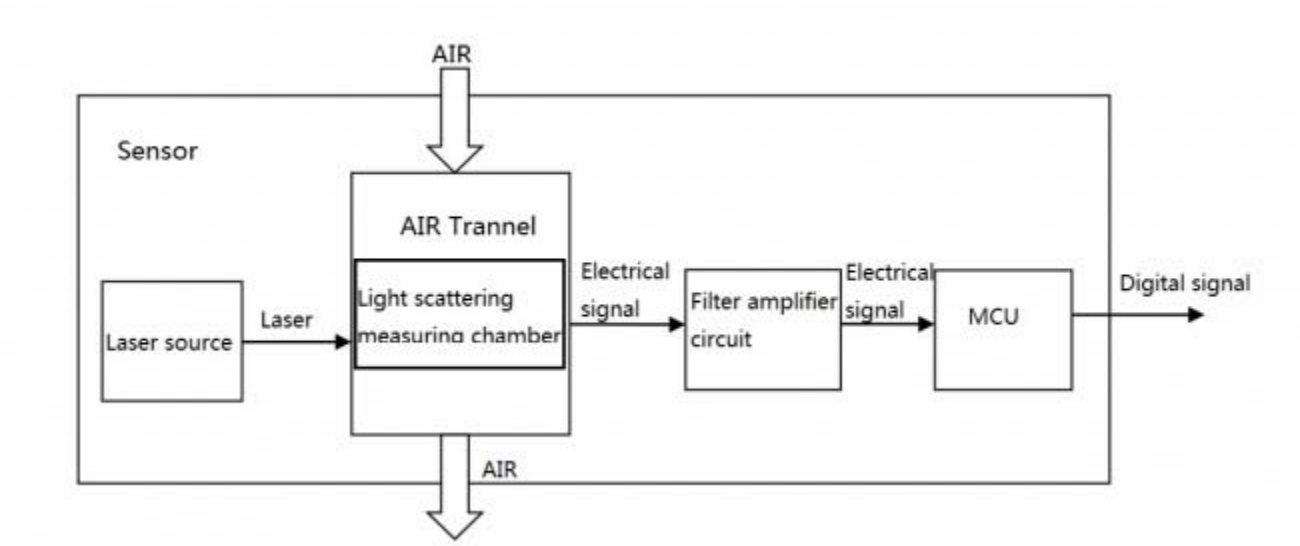


## Introduction

PM2.5 laser dust sensor is a digital universal particle concentration sensor, it can be used to obtain the number of suspended particulate matter in a unit volume of air within 0.3 to 10 microns, namely the concentration of particulate matter, and output with digital interface, also can output quality data of per particle. The sensors can be embedded in a variety of concentrations of environment-related instruments suspended particulate matter in the air ,to provide timely and accurate concentration data.

## How it works?

The sensor uses a laser scattering theory. And namely the scattering of laser irradiation in the air suspended particles, while collecting the scattered light at a specific angle, to obtain the scattering intensity versus with time curve. After the microprocessor data collection, get the relationship between the time domain and frequency domain by Fourier transform, and then through a series of complex algorithms to obtain the number of particles in the equivalent particle size and volume units of different size. Each functional block diagram of the sensor portion as shown:




sensor structure diagram

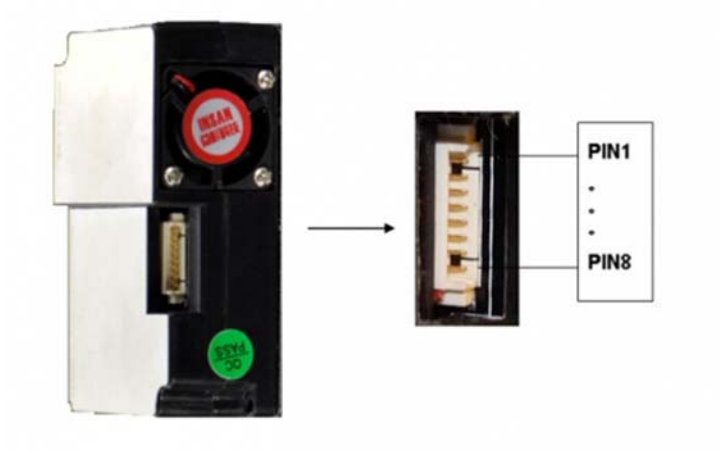
## Specification

| Basic | Feature |
|---|---|
| Operating voltage :4.95 ~ 5.05V<br>Maximum electric current: 120mA<br>Measuring pm diameter: 0.3~1.0、1.0~2.5、2.5~10（um）<br>Measuring pm range：0~500 ug/m3<br>Standby current: ≤200 uA<br>Response time: ≤10 s<br>Operating temperature range:: -20 ~ 50C<br>Operating humidity range: 0 ~ 99% RH<br>Maximum size: 65 × 42 × 23 (mm)<br>MTBF: >= 5 years | Quick response<br>Standard serial input word output<br>Second-order multi-point calibration curve<br>The minimum size of 0.3 micron resolution |

Power supply quality requirements:

1. Voltage ripple: less than 100mV.
2. The power supply voltage stability: 4.95 ~ 5.05V.
3. Power supply: more than 1W (5V@200mA).
4. The upper and lower electric voltage surge is less than 50% of the system power supply voltage.

## Connection




| Sensor Pin | Arduino Pin | Function Description |
|---|---|---|
| Pin 1 | VCC | Positive Power |
| Pin 2 | GND | Negative Power |
| Pin 3 | SET | Mode setting (More hereof later) |
| Pin 4 | RXD | receive serial port pin (3.3V level) |
| Pin 5 | TXD | Transferring serial port pin (3.3V level) |
| Pin 6 | RESET | Reset |
| Pin 7/ 8 | NC | NUll |

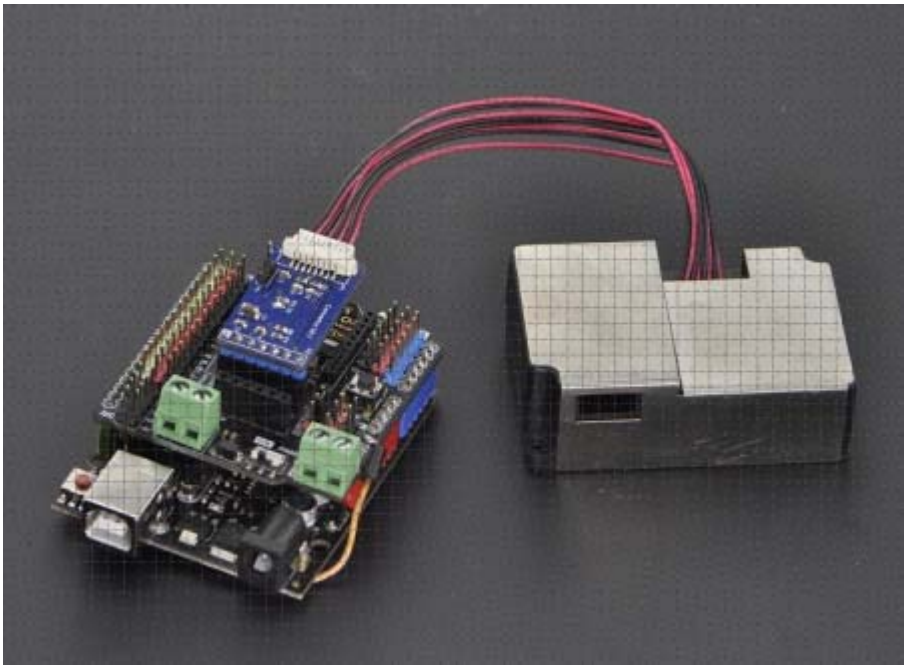
**NOTE**:

- SET:
  SET = 1, the module works in continuous sampling mode, it will upload the sample data after the end of each sampling. (The sampling response time is 1S)
  SET = 0, the module enters a low-power standby mode.
- RESET: leave it empty is OK.

# Tutorial

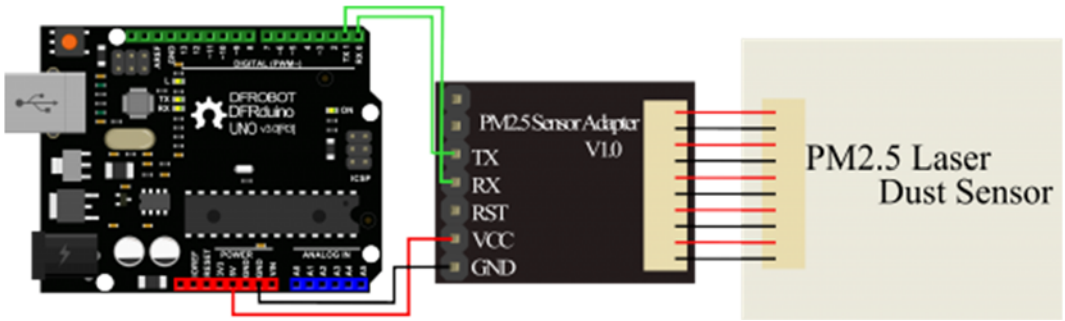
## Connection Diagram

If you have an IO expansion shield, you can simply insert the PM2.5 sensor adapter onto it, and you can use the serial to monitor the data.



pm2.5 laser dust

If you have no IO expansion shield, you can follow the wiring diagram to do wiring.




pm2.5 laser dust connecting uno

## Sample Code

**NOTE**: This code can only be verified in ArduinoIDE 1.6.x or above.

```
 //******************************
 //*Abstract: Read value of PM1,PM2.5 and PM10 of air quality
 //
 //*Version：V3.1
 //*Author：Zuyang @ HUST
 //*Modified by Cain for Arduino Hardware Serial port compatib
ility
 //*Date：March.25.2016
 //******************************
#include <Arduino.h>
#define LENG 31   //0x42 + 31 bytes equal to 32 bytes
unsigned char buf[LENG];


int PM01Value=0;          //define PM1.0 value of the air dete
ctor module
int PM2_5Value=0;         //define PM2.5 value of the air dete
ctor module
int PM10Value=0;         //define PM10 value of the air detect
or module



void setup()
{
  Serial.begin(9600);   //use serial0
  Serial.setTimeout(1500);    //set the Timeout to 1500ms, lon
ger than the data transmission periodic time of the sensor


}
```

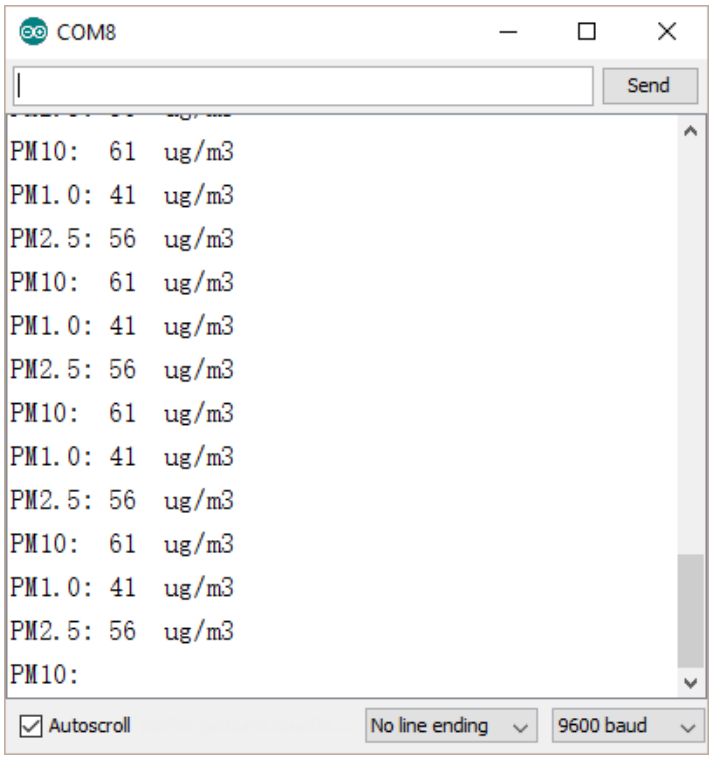
```
void loop()
{
  if(Serial.find(0x42)){    //start to read when detect 0x42
    Serial.readBytes(buf,LENG);


    if(buf[0] == 0x4d){
      if(checkValue(buf,LENG)){
        PM01Value=transmitPM01(buf); //count PM1.0 value of th
e air detector module
        PM2_5Value=transmitPM2_5(buf);//count PM2.5 value of t
he air detector module
        PM10Value=transmitPM10(buf); //count PM10 value of the
air detector module
      }
    }
  }


  static unsigned long OledTimer=millis();
    if (millis() - OledTimer >=1000)
    {
      OledTimer=millis();


      Serial.print("PM1.0: ");
      Serial.print(PM01Value);
      Serial.println("  ug/m3");


      Serial.print("PM2.5: ");
      Serial.print(PM2_5Value);
      Serial.println("  ug/m3");


      Serial.print("PM1 0: ");
      Serial.print(PM10Value);
      Serial.println("  ug/m3");
      Serial.println();
    }
```

```
}
char checkValue(unsigned char *thebuf, char leng)
{
  char receiveflag=0;
  int receiveSum=0;


  for(int i=0; i<(leng-2); i++){
  receiveSum=receiveSum+thebuf[i];
  }
  receiveSum=receiveSum + 0x42;


  if(receiveSum == ((thebuf[leng-2]<<8)+thebuf[leng-1]))  //ch
eck the serial data
  {
    receiveSum = 0;
    receiveflag = 1;
  }
  return receiveflag;
}


int transmitPM01(unsigned char *thebuf)
{
  int PM01Val;
  PM01Val=((thebuf[3]<<8) + thebuf[4]); //count PM1.0 value of
the air detector module
  return PM01Val;
}


//transmit PM Value to PC
int transmitPM2_5(unsigned char *thebuf)
{
  int PM2_5Val;
  PM2_5Val=((thebuf[5]<<8) + thebuf[6]);//count PM2.5 value of
the air detector module
```

```
  return PM2_5Val;
  }


//transmit PM Value to PC
int transmitPM10(unsigned char *thebuf)
{
  int PM10Val;
  PM10Val=((thebuf[7]<<8) + thebuf[8]); //count PM10 value of
the air detector module
  return PM10Val;
}
```

## Result

Please wait 30s for the data.




PM2.5_Result

## Communication protocol

Serial port baudrate: 9600; Parity: None; Stop Bits: 1; packet length is fixed at 32 bytes.

| | |
|---|---|
| Start Character 1 | 0x42(fixed bit) |
| Start Character 2 | 0x4d(fixed bit) |
| Frame Length 16-byte | Frame Length = 2*9+2 (data+check bit) |
| Data 1, 16-byte | concentration of PM1.0, ug/m3 |
| Data 2, 16-byte | concentration of PM2.5, ug/m3 |
| Data 3, 16-byte | concentration of PM10.0, ug/m3 |
| Data 4, 16-byte | Internal test data |
| Data 5, 16-byte | Internal test data |
| Data 6, 16-byte | Internal test data |
| Data 7, 16-byte | the number of particulate of diameter above 0.3um in 0.1 liters of air |
| Data 8, 16-byte | the number of particulate of diameter above 0.5um in 0.1 liters of air |
| Data 9, 16-byte | the number of particulate of diameter above 1.0um in 0.1 liters of air |
| Data 10, 16-byte | the number of particulate of diameter above 2.5um in 0.1 liters of air |
| Data 11, 16-byte | the number of particulate of diameter above 5.0um in 0.1 liters of air |
| Data 12, 16-byte | the number of particulate of diameter above 10.0um in 0.1 liters of air |
| Data 13, 16-byte | Internal test data |
| Check Bit for Data Sum, 16-byte | Check Bit = Start Character 1 + Start Character 2 + ...all data |

Dimensions