|  | [PMS5003](https://shop.pimoroni.com/products/pms5003-particulate-matter-sensor-with-cable?variant=29075640352851) | [PAS-OUT-01](https://www.pranaair.com/us/air-quality-sensor/outdoor-pm-sensor/) | [PAS-IN-01](https://www.pranaair.com/air-quality-sensor/indoor-pm-sensor/) | [SPS30](https://sensirion.com/products/catalog/SPS30/) |
| --- | --- | --- | --- | --- |
| Range (concentration) | 0-500 ug/m3 | 0-31000 ug/m3 | 0-1500 ug/m3 | 0-1000 ug/m3 |
| Accuracy | ±10% or ±10 ug/m3, whichever is greater | ±10% or ±10 ug/m3, whichever is greater | ±10% or ±10 ug/m3, whichever is greater | ±10% or ±10 ug/m3, whichever is greater |
| Power Consumption (On) | ~500 mW | ~350 mW | ~350mW? | ~275 mW |
| Price ($) | 39.95 | 19 | 19 | 49.53 |
| Notes | 30 seconds before accurate data, needs calibration every 30 days | 30 seconds before accurate data | 15 seconds before accurate data, | 10 seconds before accurate data |

PMS5003 ruled out due to calibration needs,

PAS-IN-01 sensor - 19$ each for less than 25, 25+ is 17$, 100+ is 15$

CO2 sensor same price - very reliable for year plus

Call - (612)299-2211 for more information or ordering process

Sensor selected SPS30

Resources for connecting to energia

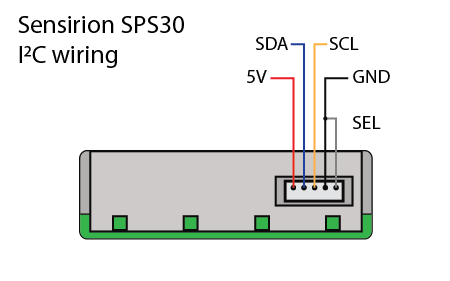


Figure 1: pinout of SPS30 sensor

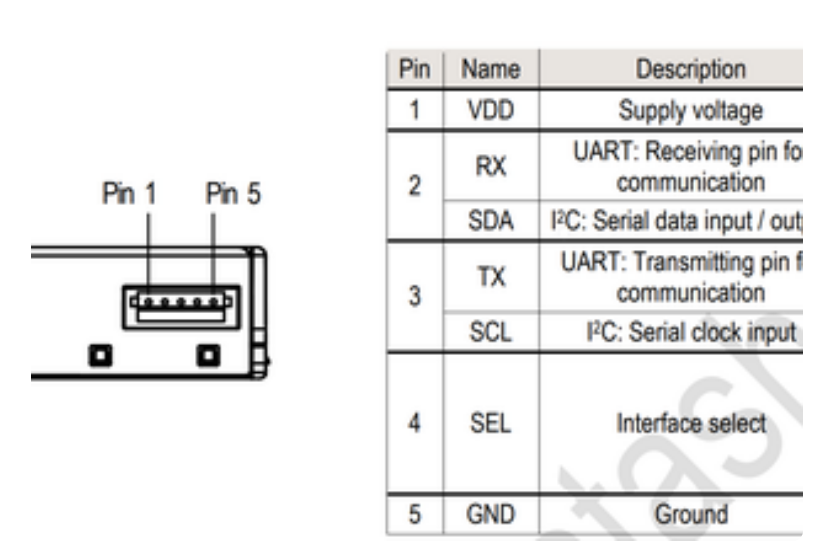


Figure 2: pin break down

#include <sps30.h>

// Example arduino sketch, based on

// https://github.com/Sensirion/embedded-sps/blob/master/sps30-i2c/sps30\_example\_usage.c

// uncomment the next line to use the serial plotter

#define PLOTTER\_FORMAT

void setup() {

int16\_t ret;

uint8\_t auto\_clean\_days = 4;

uint32\_t auto\_clean;

Serial.begin(9600);

delay(2000);

sensirion\_i2c\_init();

while (sps30\_probe() != 0) {

Serial.print("SPS sensor probing failed\n");

delay(500);

}

#ifndef PLOTTER\_FORMAT

Serial.print("SPS sensor probing successful\n");

#endif /\* PLOTTER\_FORMAT \*/

ret = sps30\_set\_fan\_auto\_cleaning\_interval\_days(auto\_clean\_days);

if (ret) {

Serial.print("error setting the auto-clean interval: ");

Serial.println(ret);

}

ret = sps30\_start\_measurement();

if (ret < 0) {

Serial.print("error starting measurement\n");

}

#ifndef PLOTTER\_FORMAT

Serial.print("measurements started\n");

#endif /\* PLOTTER\_FORMAT \*/

#ifdef SPS30\_LIMITED\_I2C\_BUFFER\_SIZE

Serial.print("Your Arduino hardware has a limitation that only\n");

Serial.print(" allows reading the mass concentrations. For more\n");

Serial.print(" information, please check\n");

Serial.print(" https://github.com/Sensirion/arduino-sps#esp8266-partial-legacy-support\n");

Serial.print("\n");

delay(2000);

#endif

delay(1000);

}

void loop() {

struct sps30\_measurement m;

char serial[SPS30\_MAX\_SERIAL\_LEN];

uint16\_t data\_ready;

int16\_t ret;

do {

ret = sps30\_read\_data\_ready(&data\_ready);

if (ret < 0) {

Serial.print("error reading data-ready flag: ");

Serial.println(ret);

} else if (!data\_ready)

Serial.print("data not ready, no new measurement available\n");

else

break;

delay(100); /\* retry in 100ms \*/

} while (1);

ret = sps30\_read\_measurement(&m);

if (ret < 0) {

Serial.print("error reading measurement\n");

} else {

#ifndef PLOTTER\_FORMAT

Serial.print("PM 1.0: ");

Serial.println(m.mc\_1p0);

Serial.print("PM 2.5: ");

Serial.println(m.mc\_2p5);

Serial.print("PM 4.0: ");

Serial.println(m.mc\_4p0);

Serial.print("PM 10.0: ");

Serial.println(m.mc\_10p0);

#ifndef SPS30\_LIMITED\_I2C\_BUFFER\_SIZE

Serial.print("NC 0.5: ");

Serial.println(m.nc\_0p5);

Serial.print("NC 1.0: ");

Serial.println(m.nc\_1p0);

Serial.print("NC 2.5: ");

Serial.println(m.nc\_2p5);

Serial.print("NC 4.0: ");

Serial.println(m.nc\_4p0);

Serial.print("NC 10.0: ");

Serial.println(m.nc\_10p0);

Serial.print("Typical partical size: ");

Serial.println(m.typical\_particle\_size);

#endif

Serial.println();

#else

// since all values include particles smaller than X, if we want to create buckets we

// need to subtract the smaller particle count.

// This will create buckets (all values in micro meters):

// - particles <= 0,5

// - particles > 0.5, <= 1

// - particles > 1, <= 2.5

// - particles > 2.5, <= 4

// - particles > 4, <= 10

Serial.print(m.nc\_0p5);

Serial.print(" ");

Serial.print(m.nc\_1p0 - m.nc\_0p5);

Serial.print(" ");

Serial.print(m.nc\_2p5 - m.nc\_1p0);

Serial.print(" ");

Serial.print(m.nc\_4p0 - m.nc\_2p5);

Serial.print(" ");

Serial.print(m.nc\_10p0 - m.nc\_4p0);

Serial.println();

#endif /\* PLOTTER\_FORMAT \*/

}

delay(1000);

}

Example code gleaned from https://www.instructables.com/Interfacing-Sensirion-SPS-30-Particulate-Matter-Se/

Github arduino based SPS30 code <https://github.com/Sensirion/arduino-sps/tree/master/examples/sps30>

Arduino to energia

<https://www.embeddedrelated.com/showarticle/525.php>

Red - 5V

orange/bluecap -p1.2 SDA

yellow/blue - p1.3 SCL

black/white - GND