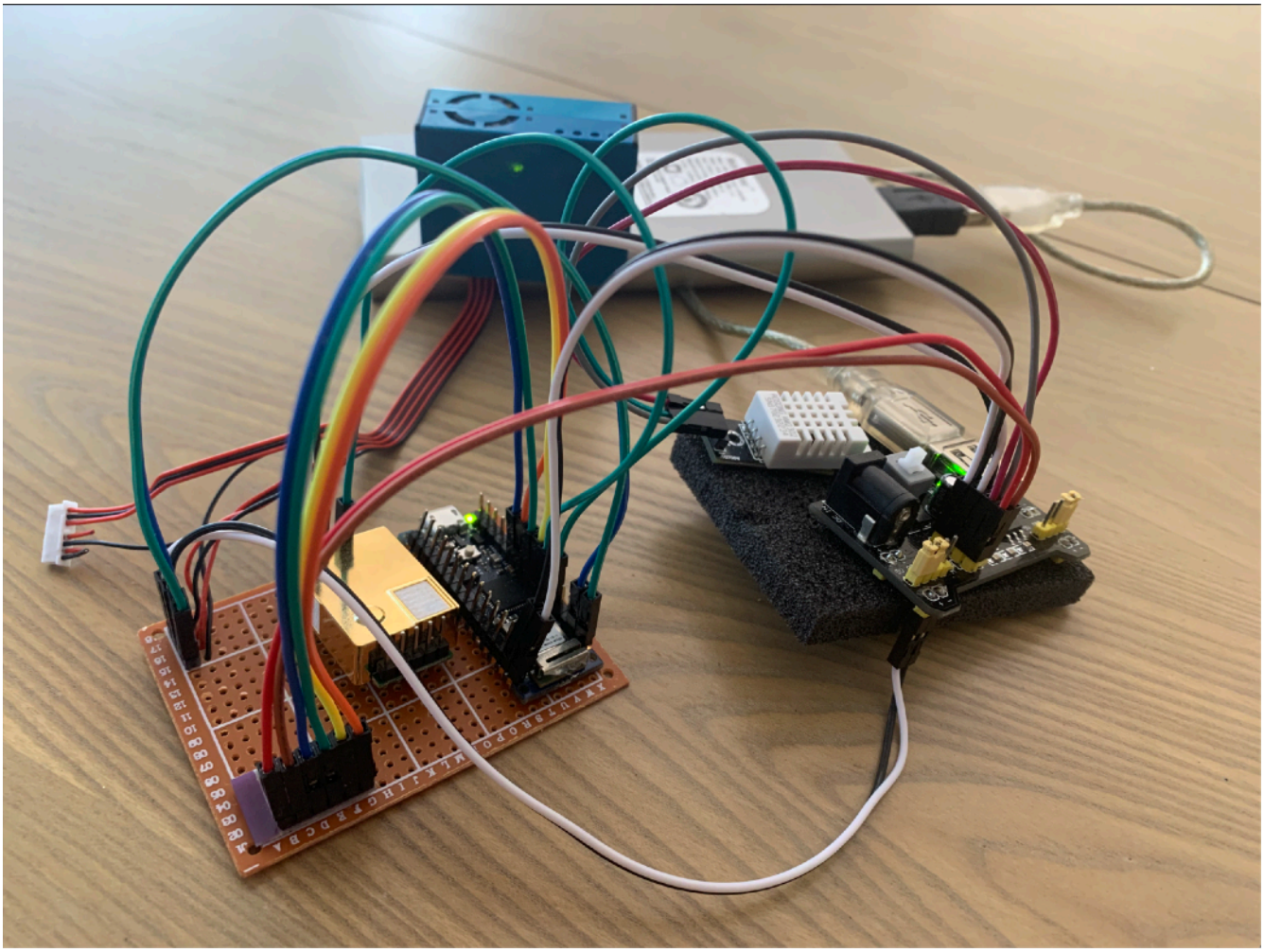
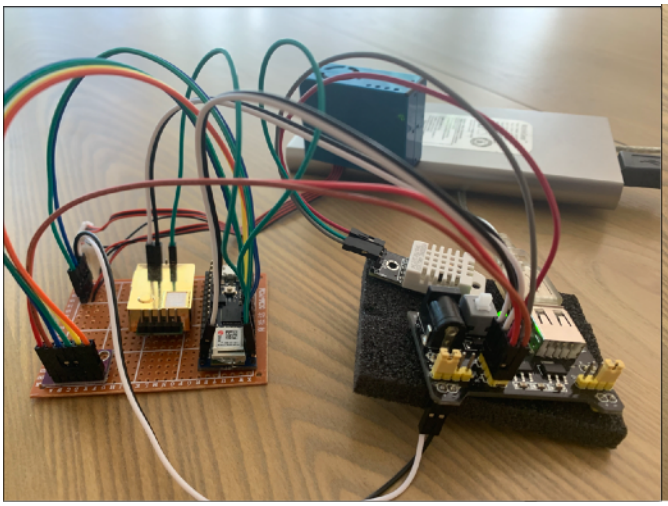
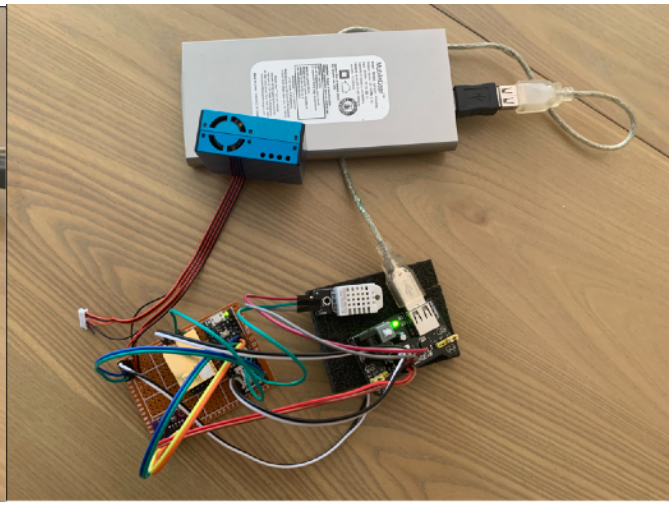
Tricorder - Arduino project







Pavlo Telets

Table of Contents:

1.Brief Introduction

The initial idea of this project was to make a WiFi-connected weather station which would display the information by the network connection via the browser. Some of the examples were found on the Internet and gave the idea of what to do. The initial plan was simple, display just the temperature with humidity and the Carbon Dioxide reading in the air, the information would be transmitted via the WiFi module to the network where it could be opened and checked via the browser. This is the point where the new and a current idea originated..

Since the initial WiFi sensor was too bulky, the plan to utilize more smaller components came in to play. This is where I used Arduino Nano33 IoT as the base of the project. Due to a limited power pinout (3.3V and a Ground pin) there was a need for connecting something what could hold more power connections than base. To maximize the pinout connections, I used Extendable Power Supply which is limited only to the Ground and Power (Power being either 3.3V connection or 5V connection, nothing else). The problem with a limited power pinout was solved in almost no time. Futhermore, now was the time to connect all of the sensors into the project, for this task I've used DHT22, BMP280, CO2 sensor and PM2.5 which I soon will brief about.

2.Bill of Materials

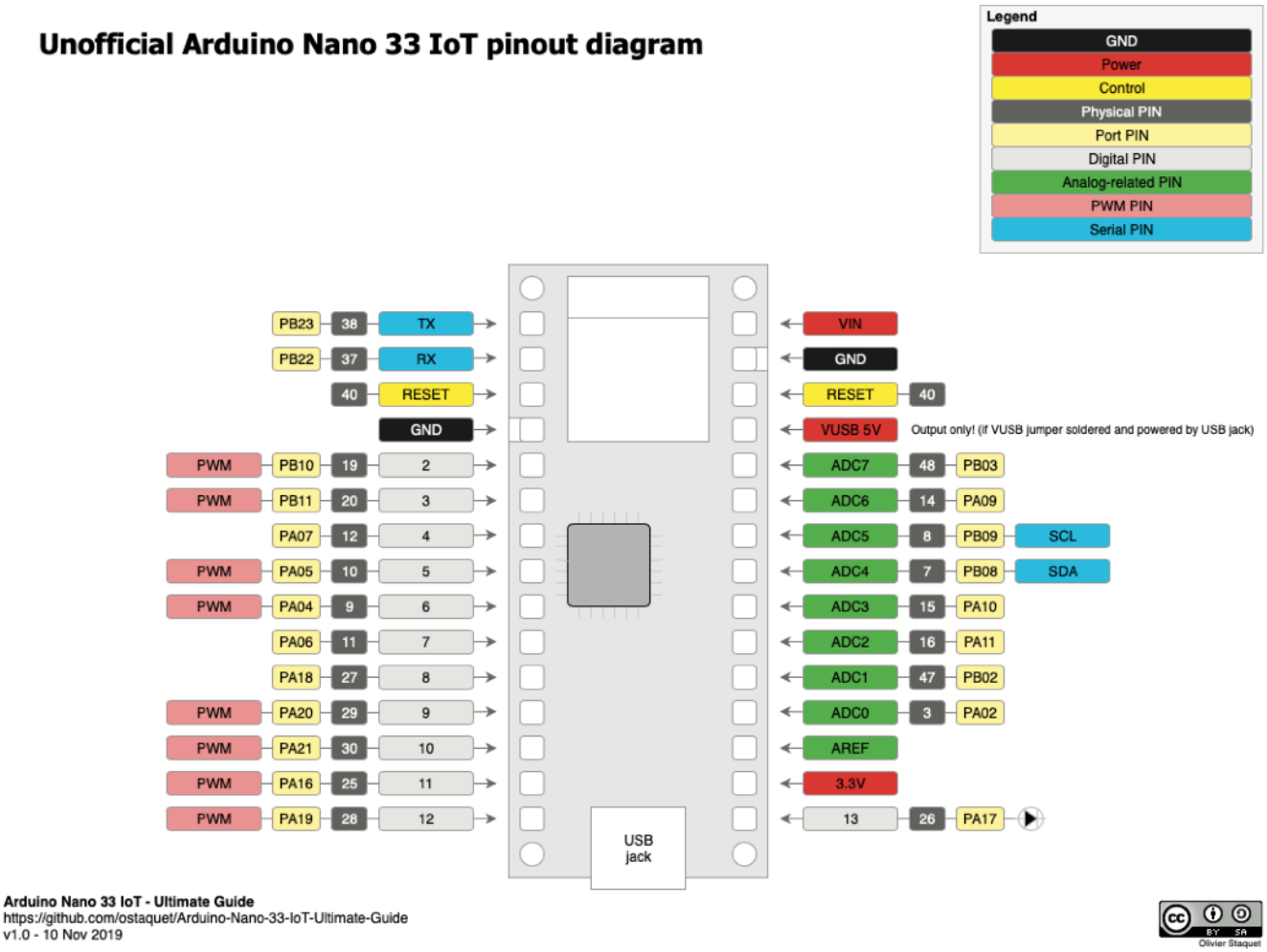
Number	Components	Description	Link
1	Arduino Board	Arduino Nano33 IoT	https://x2robotics.ca/
2	Humidity, Temperature Sensor	DHT22	DHT22 from express
3	Pressure, Altitude, Temperature Sensor	BMP280	BMP280 from aliexpress
4	CO2 sensor	MH-Z19	MH-Z19 from aliexpress
5	Air Quality Sensor	PMS5003	PMS5003 from aliexpress
6	Electronic Circuit Breadboards	BBP-32701	https://www.digikey.ca/
7	Connecting Wires - 15 in total	Jumper Wires	Somewhere from amazon

3.Nano 33 IOT

3.1 Overview

The main pros to consider about Arduino Nano33IoT is the fact that it's one of the cheapest Bluetooth and WiFi configured board available in the market. Due to it's relative small size it's perfect for any casing which may be in need. It also has lots of pinouts which can mean more sensors or other components attached and operating, furthermore, one big downside to it has only one power output and grounding (3.3V). As a solution, I use an electronic circuit breadboard.

3.2 Pinout

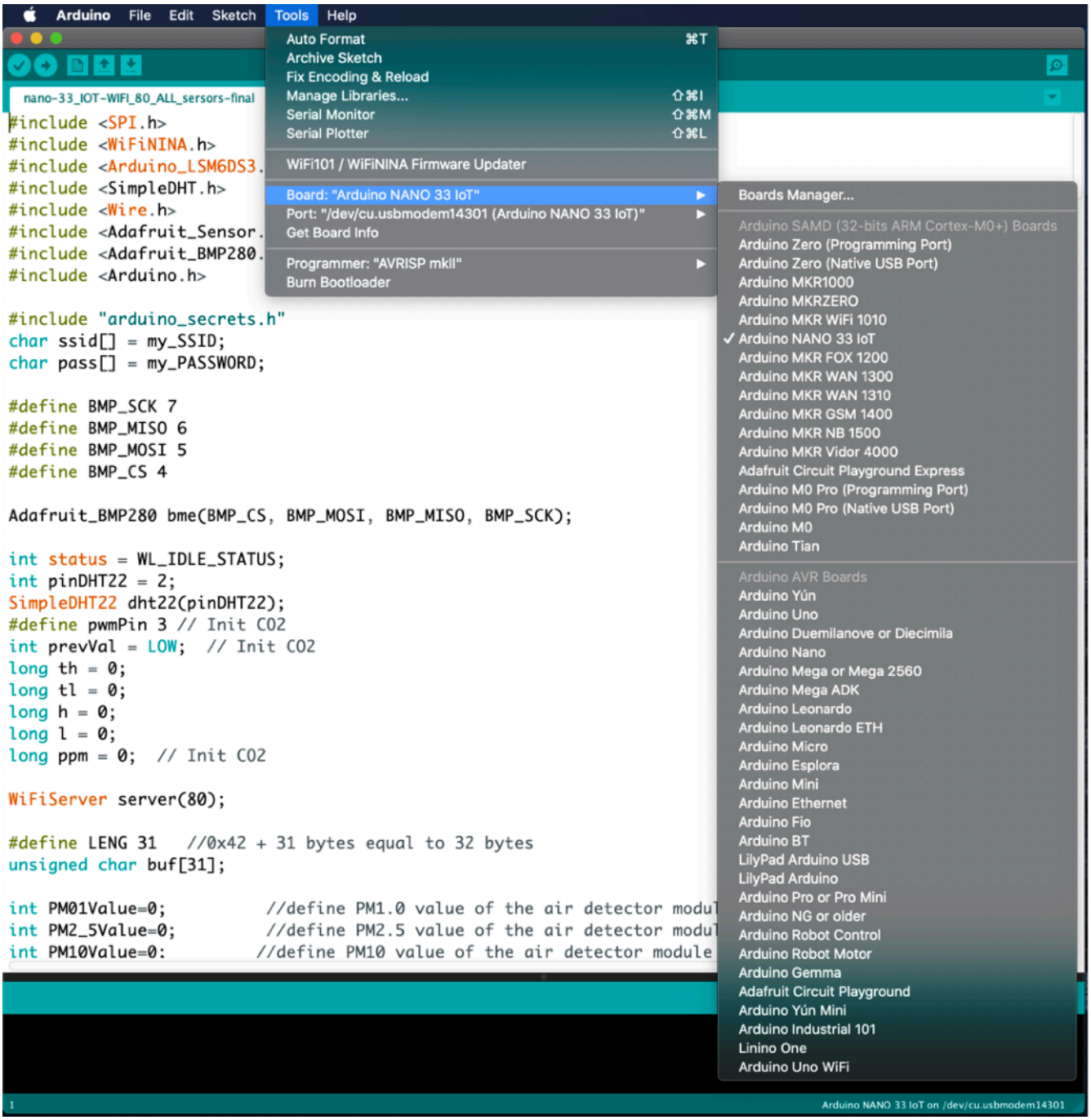


3.3 Prepare for coding

How do I use Arduino NANO 33 IoT on the Arduino Desktop IDE?

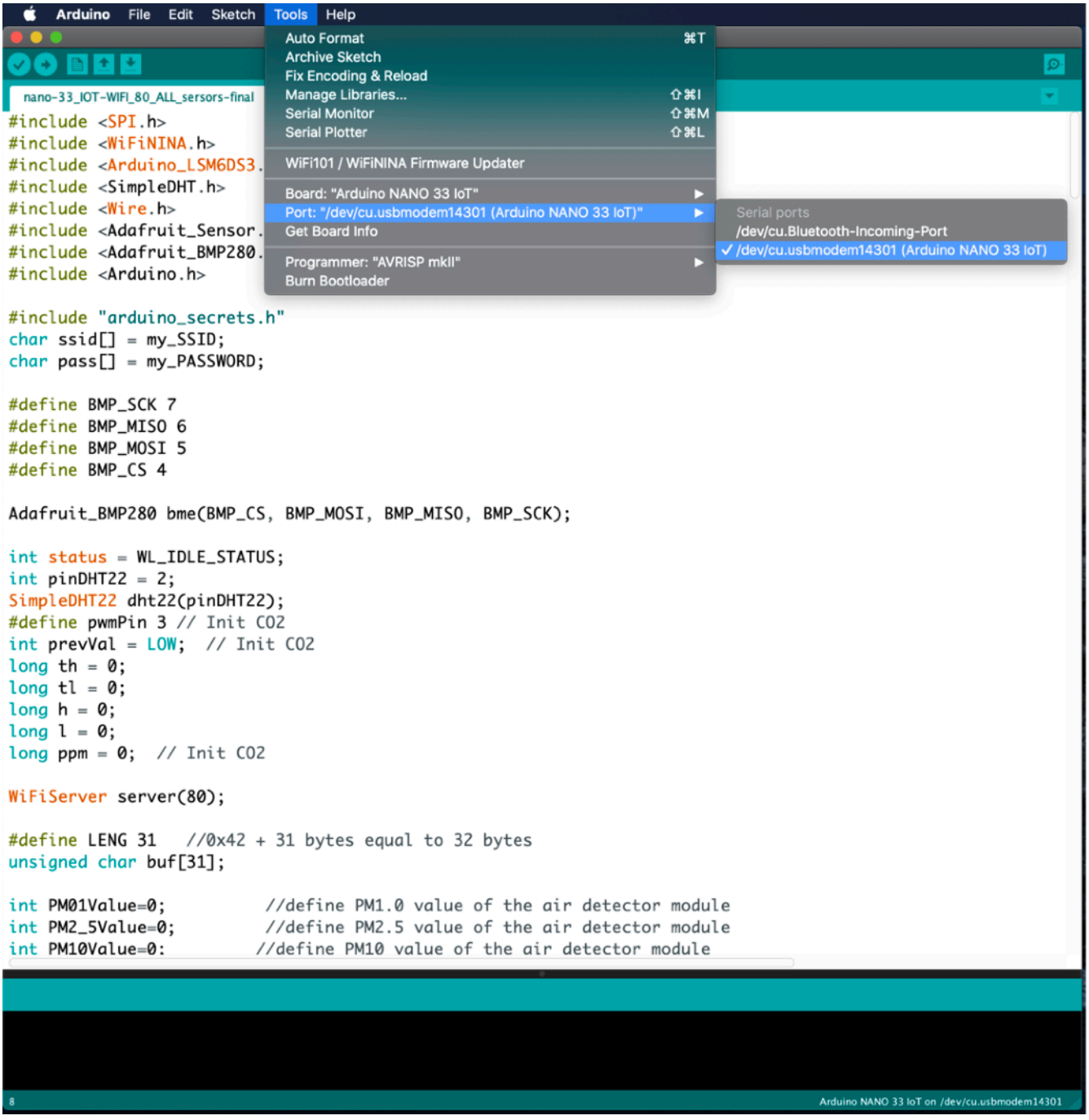
I need install on Arduino Desktop IDE Arduino SAMD Core and from Arduino Boards Manager tab.

I need to select my nano33 board in the Tools > Board menu.

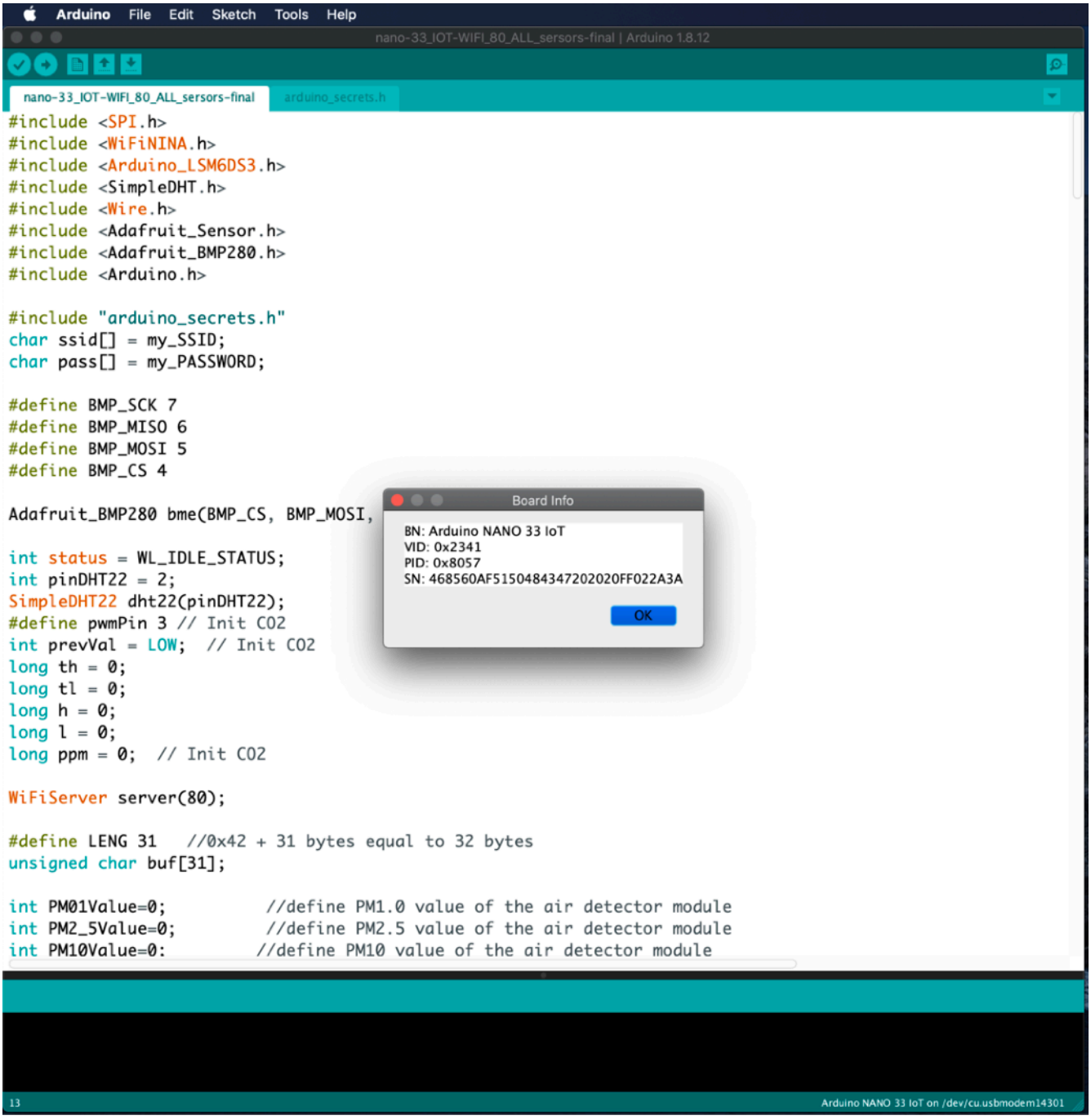


I Select the serial device port of nano33 from the Tools | Serial Port menu.

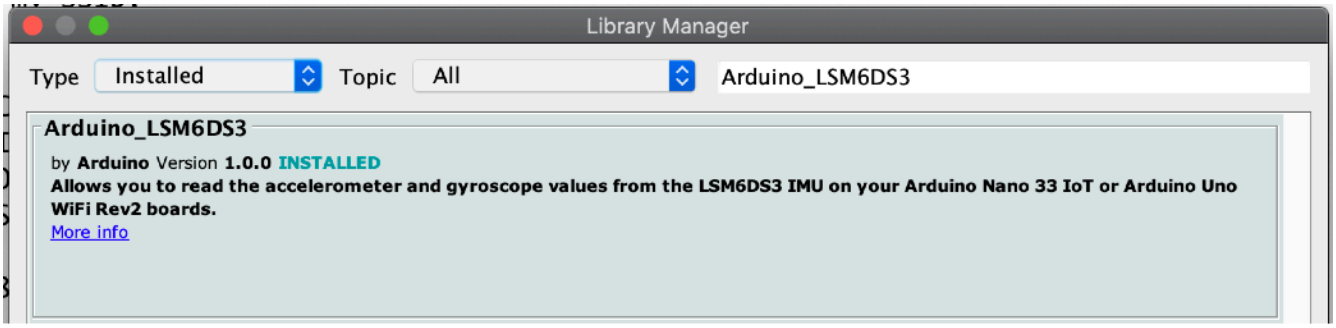
MacOS does me a favour and named the necessary device :)



Get my board info, this test means the communication is good.



Installing library for internal sensors. in the Tools > Manage Libraries…



4. Humidity and Temperature with DHT22

4.1 Overview

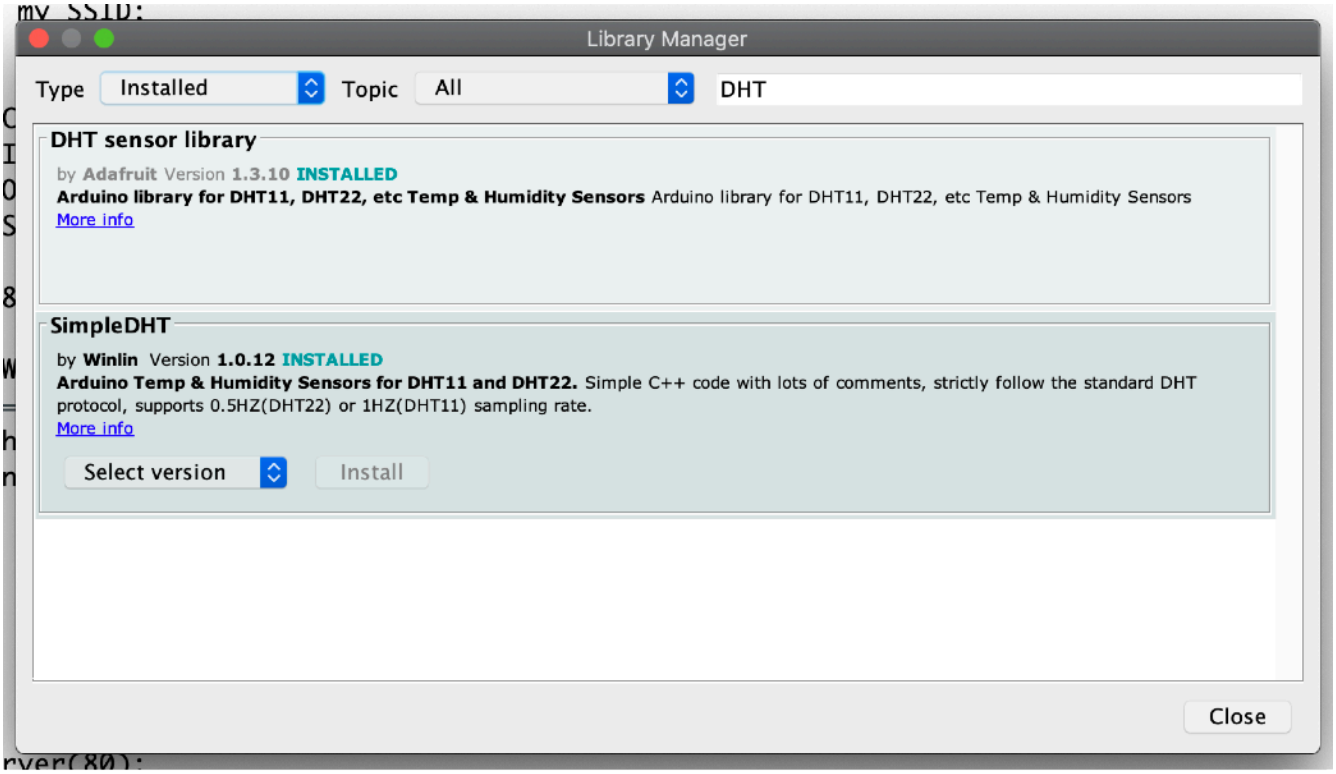
DHT22 is Humidity and Temperature configured sensor for the use. 3 pins of the connection make it quite easy to connect and in the meanwhile easy to remember.

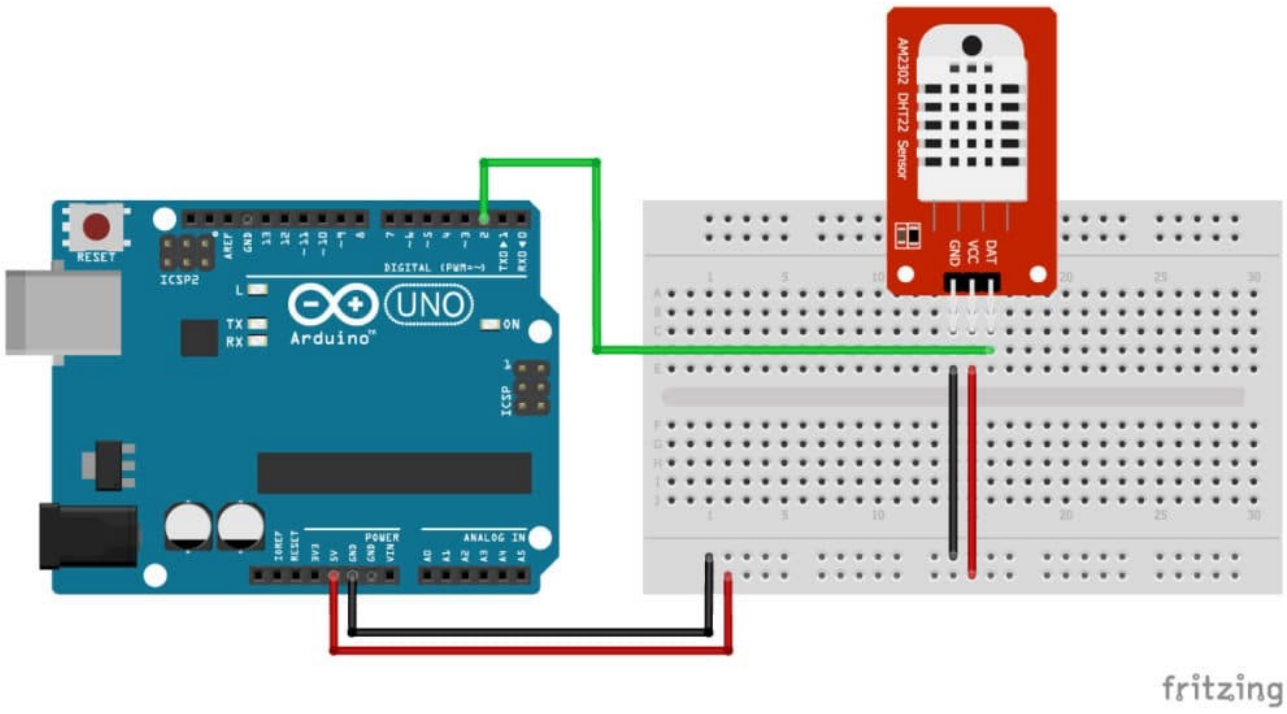
4.2 Pinout

4.3 Test Sketch

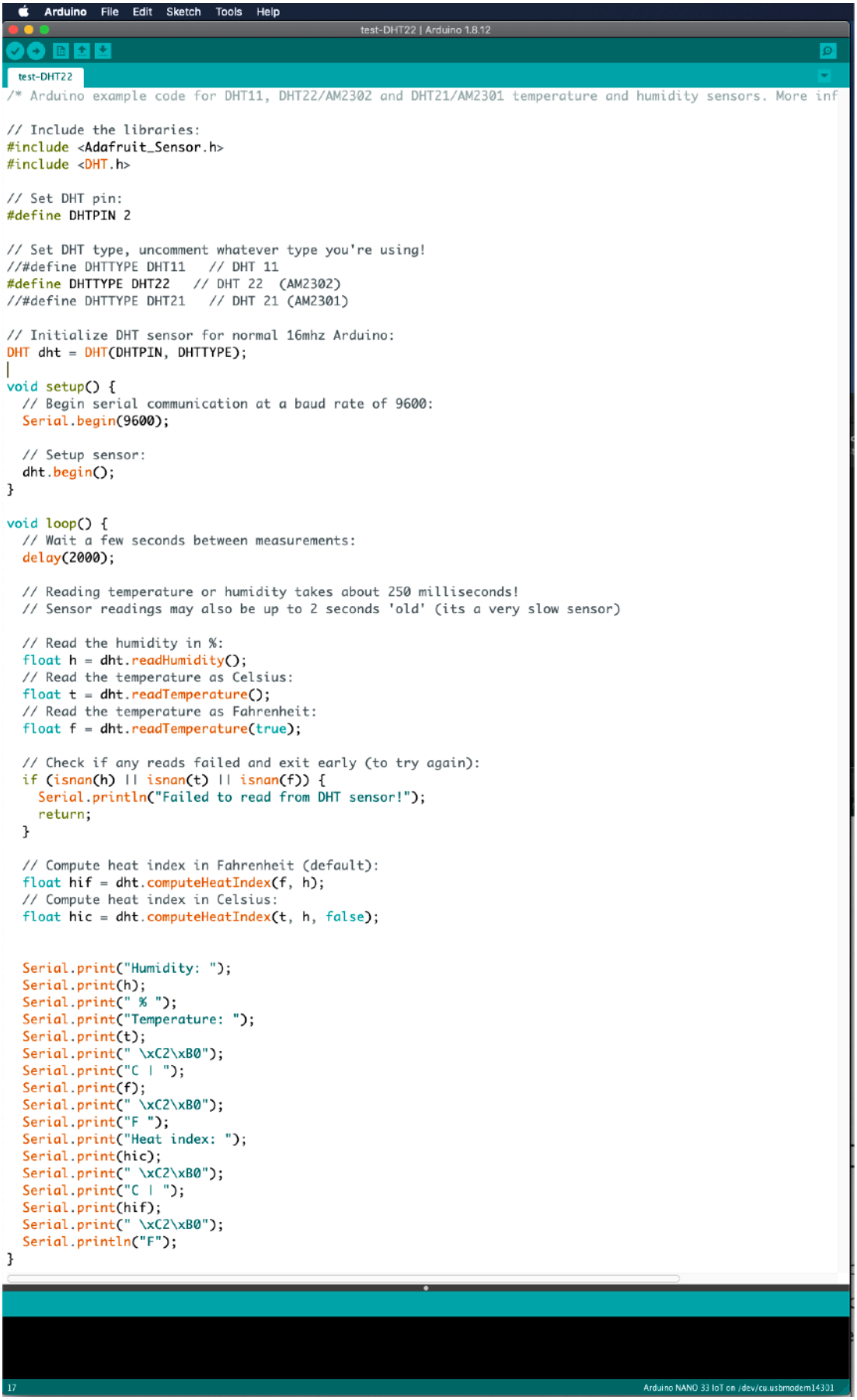
The main task of the code will be to work with Nano33IoT, therefore, has to be supported by the next libraries: Adafruit, SimpleDHT. There are 3 wires needed to be connected, VCC, OUT, GND

Installing libraries in the Tools > Manage Libraries…





Test code:


Arduino File Edit Sketch Tools Help
test-DHT22 | Arduino 1.8.12
test-DHT22
/* Arduino example code for DHT11, DHT22/AM2302 and DHT21/AM2301 temperature and humidity sensors. More inf
// Include the libraries:
#include <Adafruit_Sensor.h>
#include <DHT.h>
// Set DHT pin:
#define DHTPIN 2
// Set DHT type, uncomment whatever type you're using!
//#define DHTTYPE DHT11 // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
//#define DHTTYPE DHT21 // DHT 21 (AM2301)
// Initialize DHT sensor for normal 16mhz Arduino:
DHT dht = DHT(DHTPIN, DHTTYPE);
void setup() {
// Begin serial communication at a baud rate of 9600:
Serial.begin(9600);
// Setup sensor:
dht.begin();
}
void loop() {
// Wait a few seconds between measurements:
delay(2000);
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
// Read the humidity in %:
float h = dht.readHumidity();
// Read the temperature as Celsius:
float t = dht.readTemperature();
// Read the temperature as Fahrenheit:
float f = dht.readTemperature(true);
// Check if any reads failed and exit early (to try again):
if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
// Compute heat index in Fahrenheit (default):
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius:
float hic = dht.computeHeatIndex(t, h, false);
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" % ");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" \xC2\xB0");
Serial.print("C | ");
Serial.print(f);
Serial.print(" \xC2\xB0");
Serial.print("F ");
Serial.print("Heat index: ");
Serial.print(hic);
Serial.print(" \xC2\xB0");
Serial.print("C | ");
Serial.print(hif);
Serial.print(" \xC2\xB0");
Serial.println("F");
}
17
Arduino NANO 33 IoT on /dev/cu.usbmodem14301

5. Pressure, Altitude and Temperature with BMP280

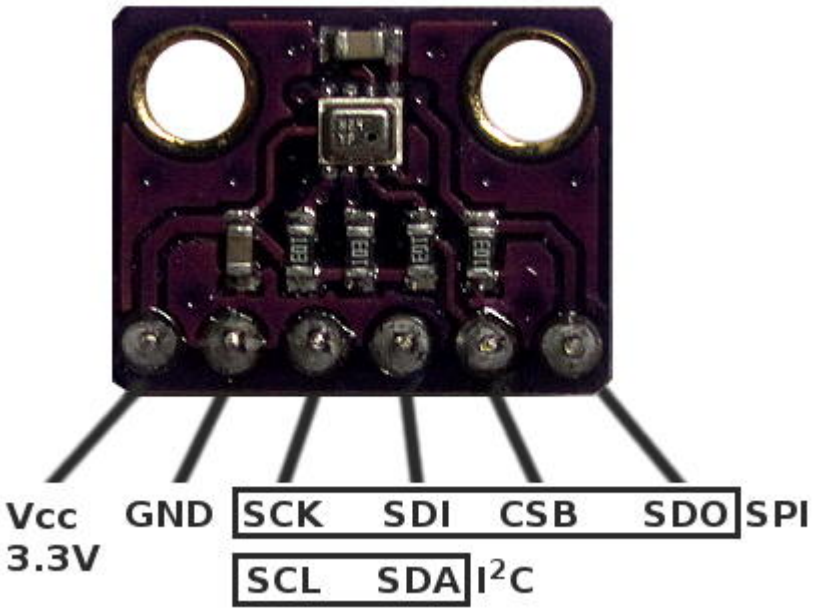
5.1 Overview

BMP280 is sensor for pressure, humidity and temperature. Due to it's small size it makes a quite good opposing candidate to DHT senor. Main con with BMP280 due to the location of power supply near the temperature sensor, the readings are off by .5 Celsius degrees, probably has to do with the heating of the component. Furthermore, it also can be configured to show approximate attitude at which sensor thinks it's located.

I use the second sensor also for verify how big be different in temperature between two sensors.

BMP always warmed up to .6 degree of Celsius than DHT. I think BMP is more correct ☺

5.2 Pinout



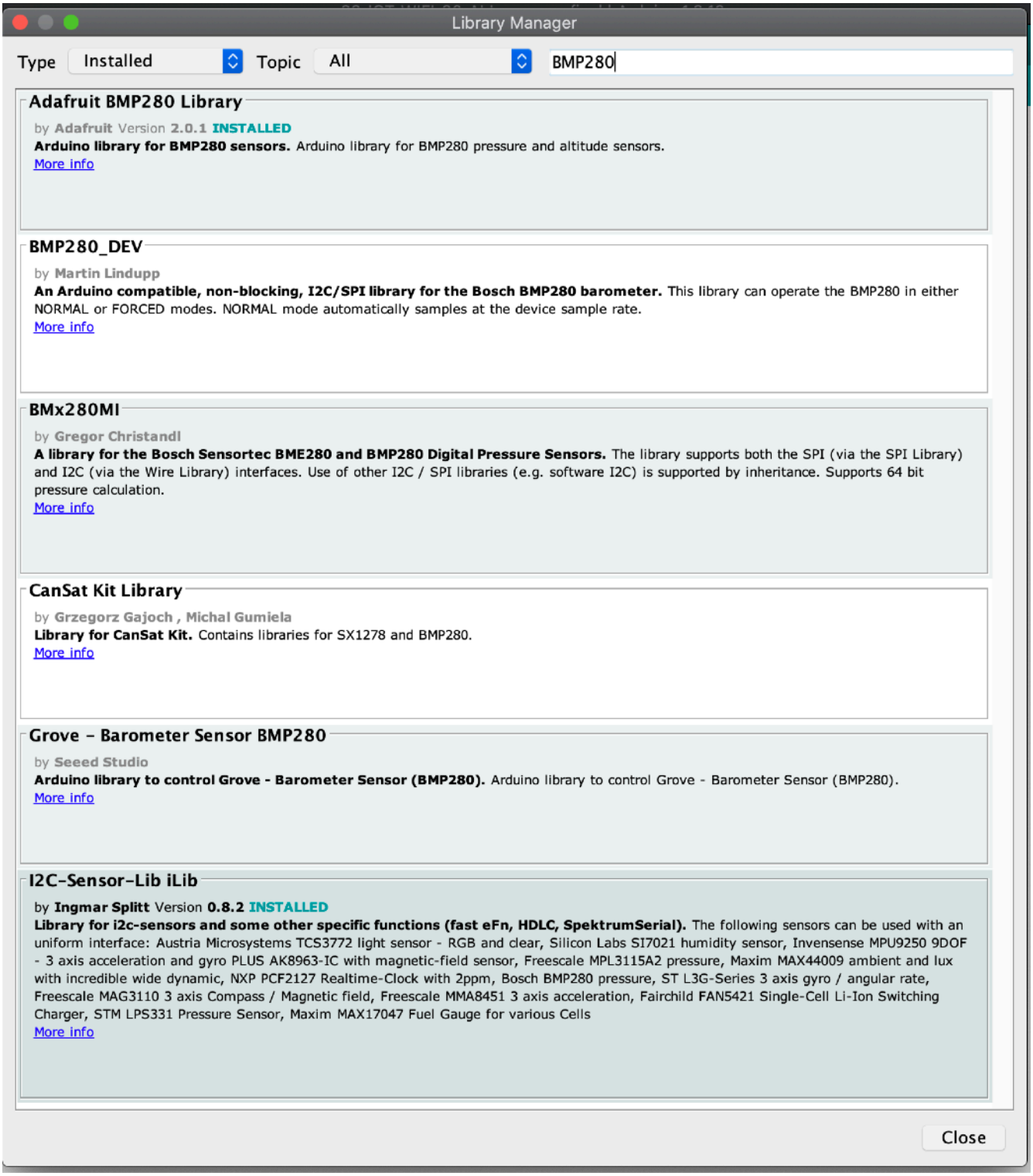
5.3 Sample Code

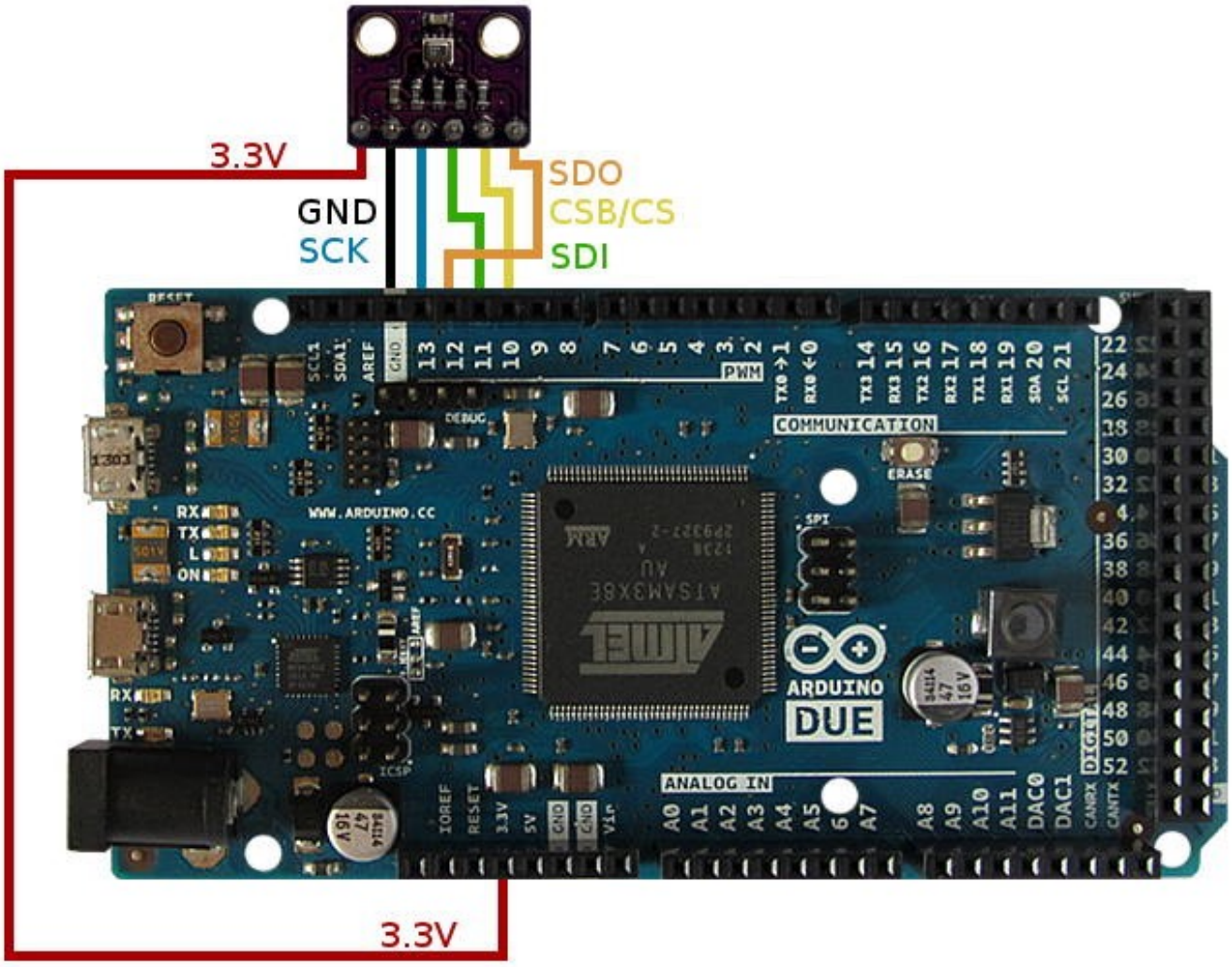
The main task for this component is to transmit the data collected which includes atmospheric pressure, humidity and temperature, the sample code is below. Some libraries for practical use include Wire, SPI, Adafruit BMP280.

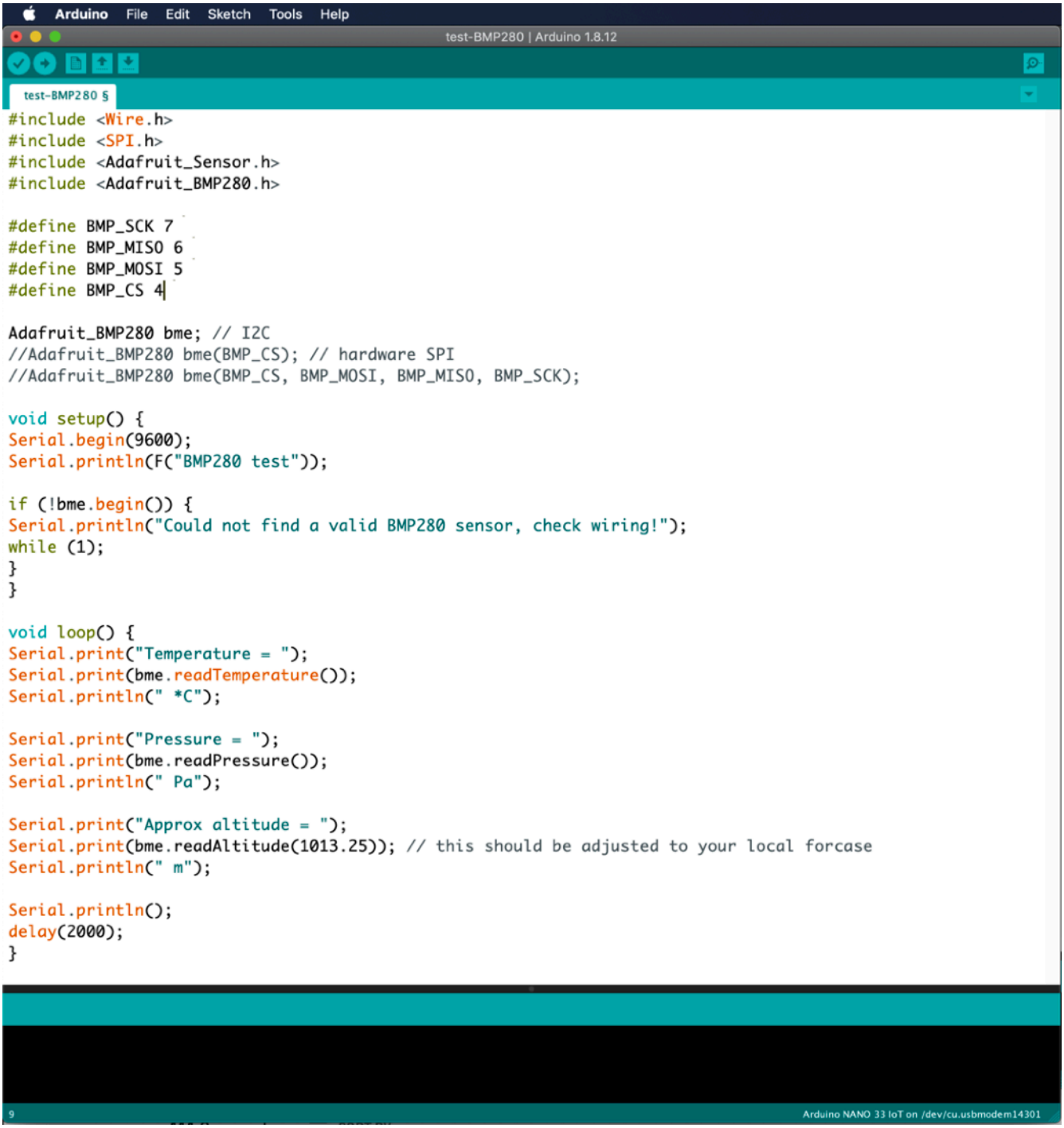
Some change was using this sensor in software SPI mode, Arduino Nano 33 IOT has only 11 digital pins, but on default pins for hardware, SPI sensor wants to start.
Software pins looks like:

- Digital Pin 7 to SCK (Serial Clock)
- Digital Pin 6 to SDO (Serial Data Out)
- Digital Pin 5 to SDI (Serial Data In)
- Digital Pin 4 to CS (Chip Select)

Installing libraries in the Tools > Manage Libraries…




3.3V
GND
SCK
SDO
CSB/CS
SDI
3.3V
ARDUINO
DUE
COMMUNICATION
ANALOG IN
PWM
ERASE
SPI
WWW.ARDUINO.CC


Arduino File Edit Sketch Tools Help
test-BMP280 | Arduino 1.8.12
test-BMP280 §
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#define BMP_SCK 7
#define BMP_MISO 6
#define BMP_MOSI 5
#define BMP_CS 4
Adafruit_BMP280 bme; // I2C
//Adafruit_BMP280 bme(BMP_CS); // hardware SPI
//Adafruit_BMP280 bme(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);
void setup() {
Serial.begin(9600);
Serial.println(F("BMP280 test"));
if (!bme.begin()) {
Serial.println("Could not find a valid BMP280 sensor, check wiring!");
while (1);
}
}
void loop() {
Serial.print("Temperature = ");
Serial.print(bme.readTemperature());
Serial.println(" *C");
Serial.print("Pressure = ");
Serial.print(bme.readPressure());
Serial.println(" Pa");
Serial.print("Approx altitude = ");
Serial.print(bme.readAltitude(1013.25)); // this should be adjusted to your local forcase
Serial.println(" m");
Serial.println();
delay(2000);
}
9
Arduino NANO 33 IoT on /dev/cu.usbmodem14301

6. CO2 Air Sensor with MH-Z19

6.1 Overview

MH-Z19 is the CO2 sensor. This sensor is configured to the readings of the CO2 presence in the air. It's configured to show the reading in the "ppm" format with number. The following link shows the safe air quality of rooms:

https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms

I am using one digital pin (3 pin in the final sketch) to get data from sensor. There is some change on nano 33 IOT has only one hardware serial, what I keep for air pollution sensor. It is not possible user the regular software serial library, in my solution sensor need a sometime to be wormed for work.

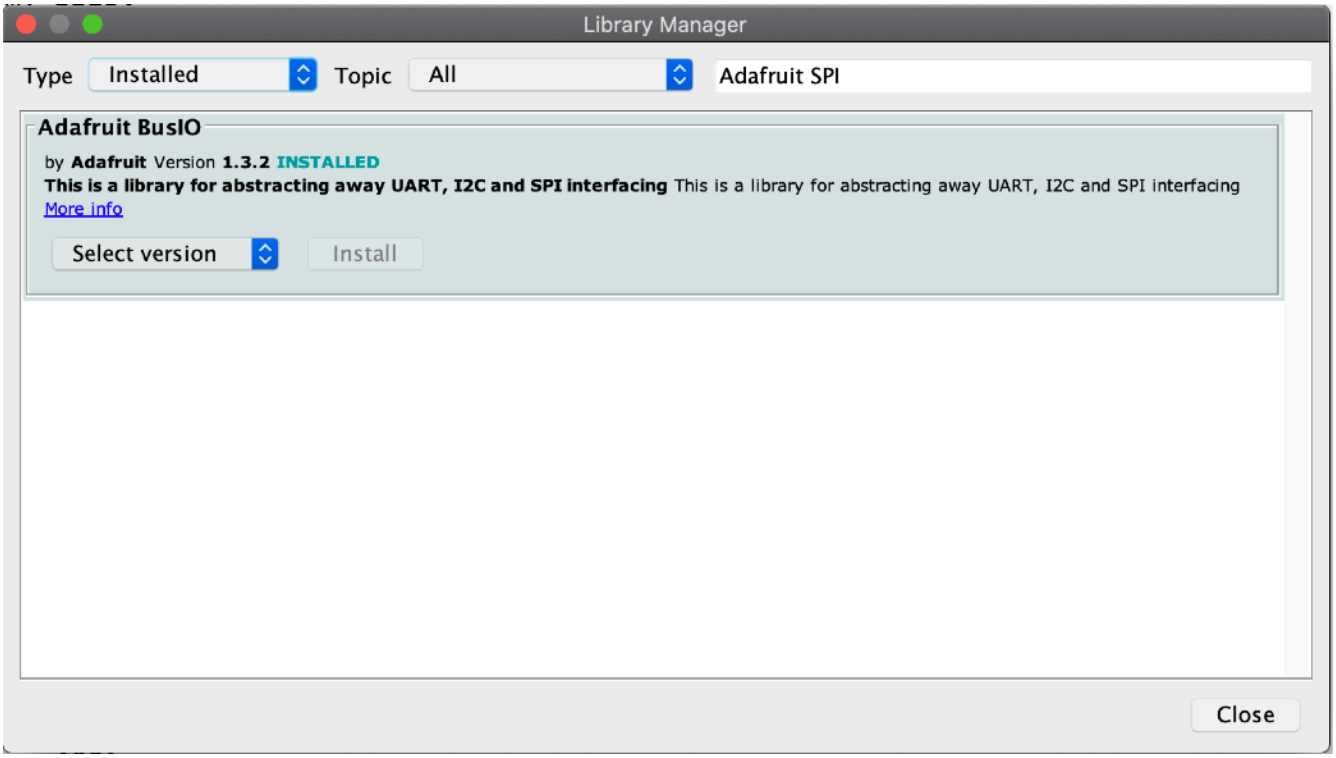
6.2 Pinout

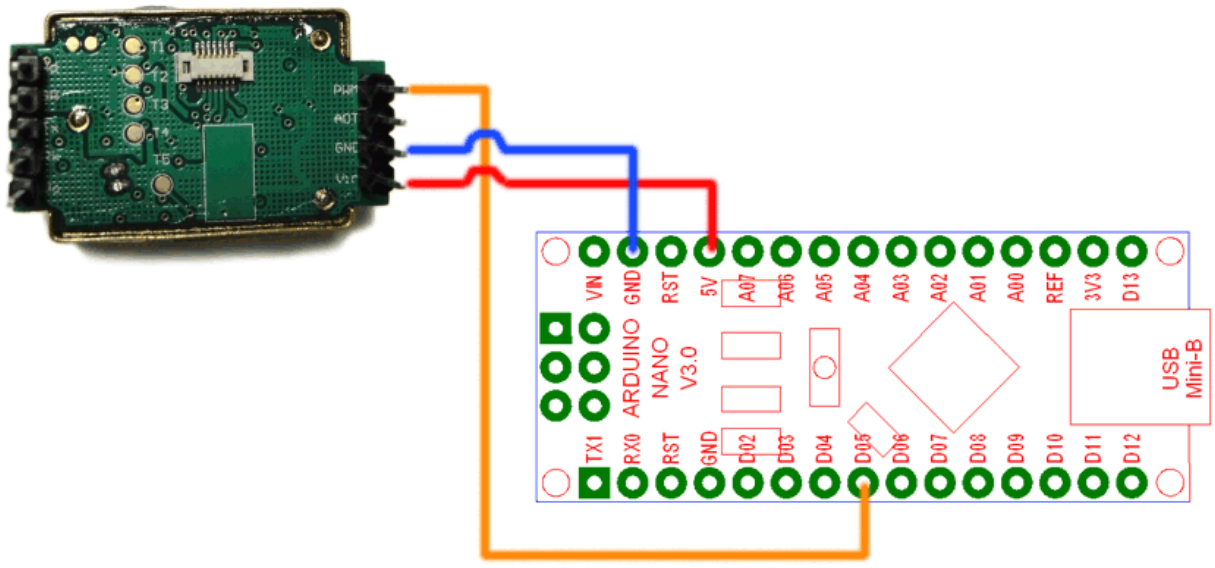
PIN	Description
Pin 6	Vin (voltage input)
Pin 7	GND
Pin 1	Vout (output voltage 3.3V, output current lower than 10mA)
Pin 9	PWM
Pin 5	HD (zero calibration, low level above 7 seconds) (Factory Reserved)
Pin 2	UART (RXD) 0~3.3V digital input
Pin 3	UART (TXD) 0~3.3V digital output
Pin 4	SR (Factory Reserved)
Pin 8	AOT (Factory Reserved)

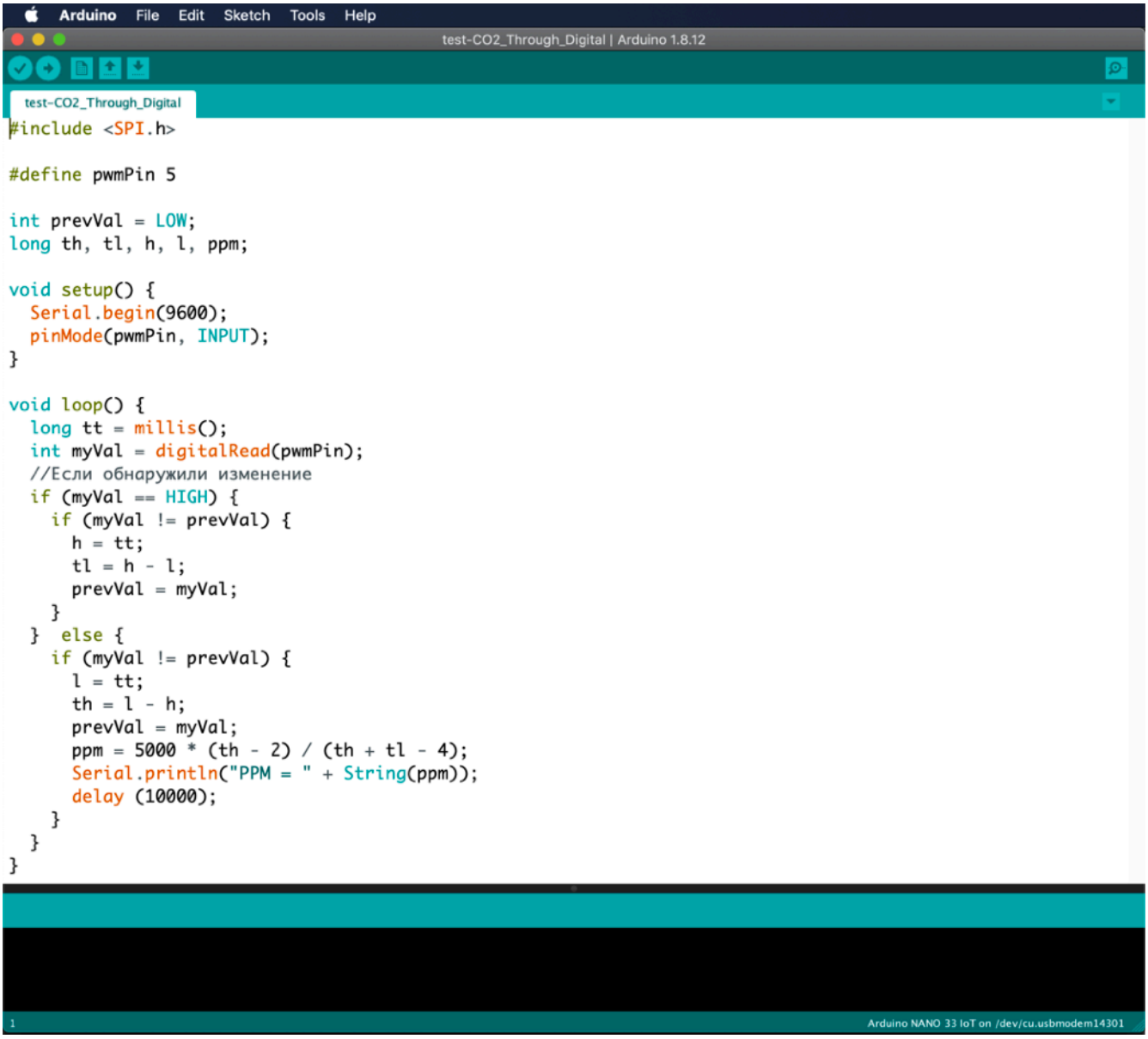
6.3 Test Sketch

The main task of this sensor is to collect the air quality readings. Some of the libraries needed for the job - SPI.

Installing libraries in the Tools > Manage Libraries…






Arduino File Edit Sketch Tools Help
test-CO2_Through_Digital | Arduino 1.8.12
test-CO2_Through_Digital
#include <SPI.h>
#define pwmPin 5
int prevVal = LOW;
long th, tl, h, l, ppm;
void setup() {
Serial.begin(9600);
pinMode(pwmPin, INPUT);
}
void loop() {
long tt = millis();
int myVal = digitalRead(pwmPin);
//Если обнаружили изменение
if (myVal == HIGH) {
if (myVal != prevVal) {
h = tt;
tl = h - l;
prevVal = myVal;
}
} else {
if (myVal != prevVal) {
l = tt;
th = l - h;
prevVal = myVal;
ppm = 5000 * (th - 2) / (th + tl - 4);
Serial.println("PPM = " + String(ppm));
delay (10000);
}
}
}
1
Arduino NANO 33 IoT on /dev/cu.usbmodem14301

7. PM2.5 Air Quality Sensor with PMS5003

7.1 Overview

PMS5003 is the Air Quality Sensor PM2.5 sensor. This sensor collects the data of how many dust particles of the size 1.0,2.5,10.0 microns or smaller in diameter, present in the air, reason being these particles are the ones which can cause health related problems. Toronto Air Pollution site is providing some corporation, I guess at home, I have more clear air ☺

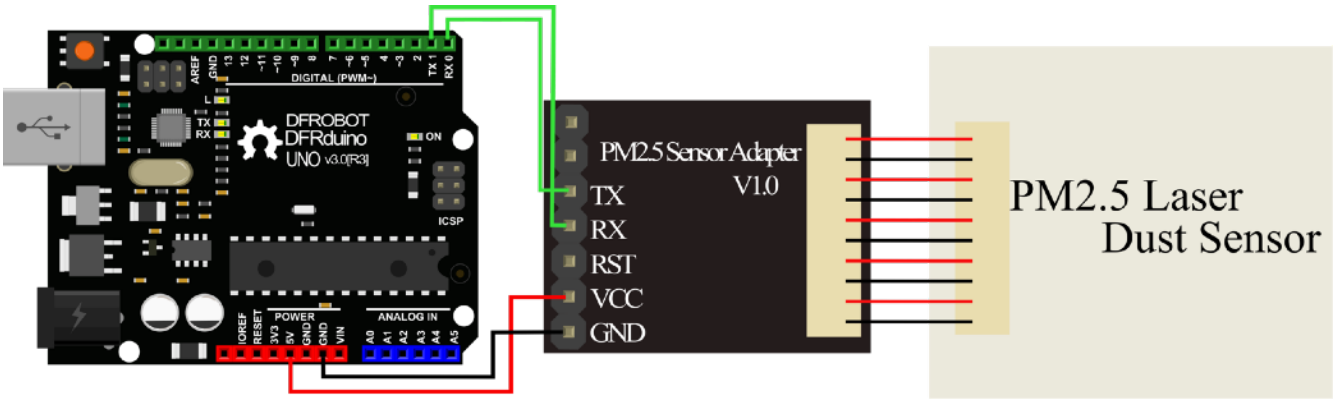
Small problem which causes significant problems during test is that sensor powers up for 5-10 minutes after which it shut down, to fix it, some of the wires were soldered into the Electronic Components Breadboard and as described early, I found that using hardware serial the sensor online. Now it works without a problem.

7.2 Pinout

- PIN1: VCC -- Positive power 5V
- PIN2: GND -- Negative power
- PIN3: -- SET Set pin /TTL level@3.3V， high level or suspending is normal working status, while low level is sleeping mode.
- PIN4: RX -- Serial port receiving pin/TTL level@3.3V
- PIN5: TX -- Serial port sending pin/TTL level@3.3V
- PIN6: RESET -- Module reset signal /TTL level@3.3V， low reset.
- PIN7/8: NC

7.3 Test Sketch

There is a need for 4 wires to be utilized. Ground, Power, RX, TX.



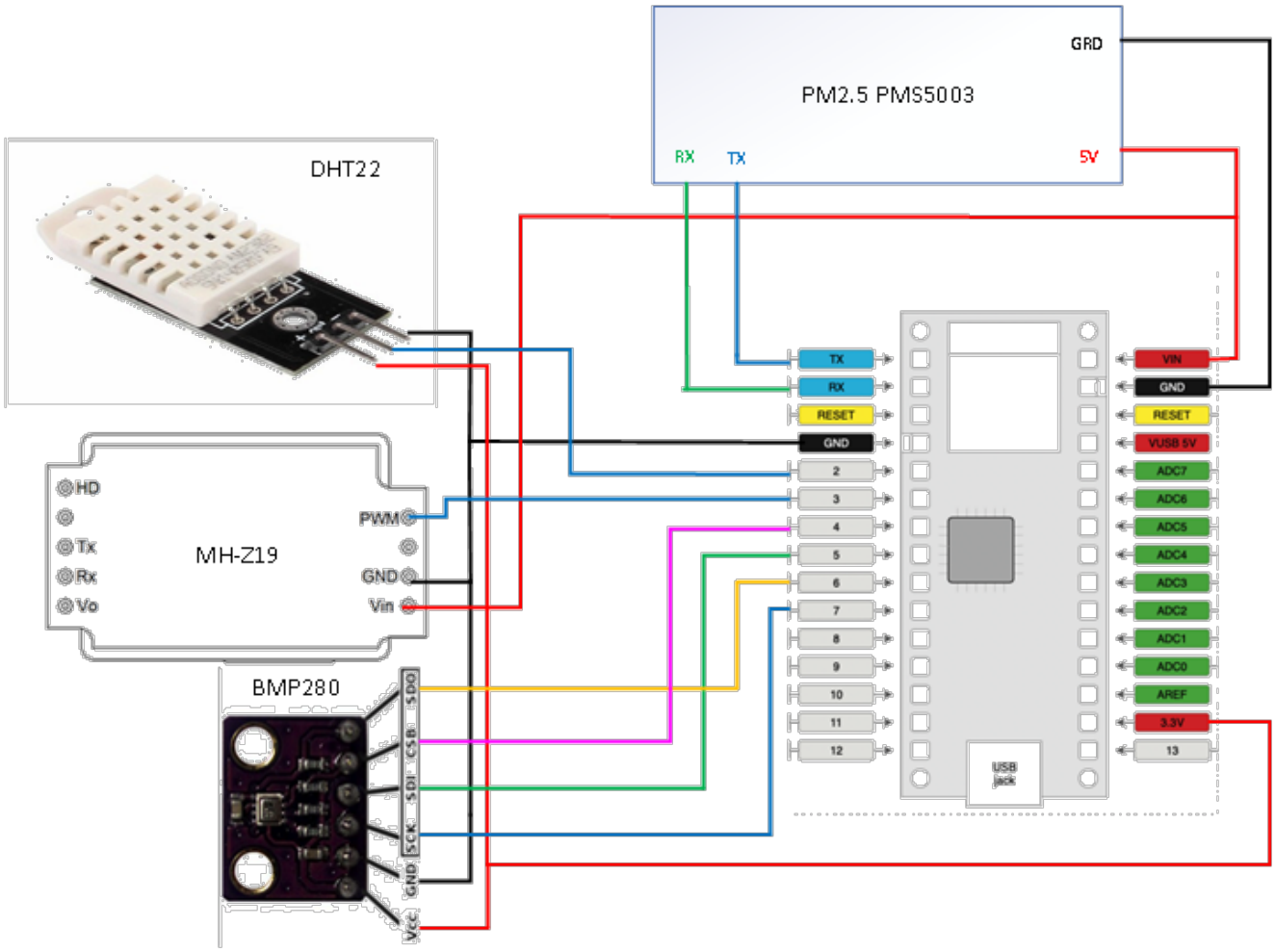

```
void setup() {
  // our debugging output
  Serial.begin(115200);
  Serial.println("Warming...");
  // sensor baud rate is 9600
  Serial1.begin(9600);

}

struct pms5003data {
  uint16_t framelen;
  uint16_t pm10_standard, pm25_standard, pm100_standard;
  uint16_t pm10_env, pm25_env, pm100_env;
  uint16_t particles_03um, particles_05um, particles_10um, particles_25um, particles_50um, particles_100um;
  uint16_t unused;
  uint16_t checksum;
};
struct pms5003data data;
void loop() {
  if (readPMSdata(&Serial1)) {
    // reading data was successful!
    Serial.println();
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (standard)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_standard);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_standard);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_standard);
    Serial.println("---------------------------------------");
    Serial.println("Concentration Units (environmental)");
    Serial.print("PM 1.0: "); Serial.print(data.pm10_env);
    Serial.print("\t\tPM 2.5: "); Serial.print(data.pm25_env);
    Serial.print("\t\tPM 10: "); Serial.println(data.pm100_env);
    Serial.println("---------------------------------------");
    Serial.print("Particles > 0.3um / 0.1L air:"); Serial.println(data.particles_03um);
    Serial.print("Particles > 0.5um / 0.1L air:"); Serial.println(data.particles_05um);
    Serial.print("Particles > 1.0um / 0.1L air:"); Serial.println(data.particles_10um);
    Serial.print("Particles > 2.5um / 0.1L air:"); Serial.println(data.particles_25um);
    Serial.print("Particles > 5.0um / 0.1L air:"); Serial.println(data.particles_50um);
    Serial.print("Particles > 10.0 um / 0.1L air:"); Serial.println(data.particles_100um);
    Serial.println("---------------------------------------");
    delay(30000);
  }
}
boolean readPMSdata(Stream *s) {
  if (! s->available()) {
    return false;
  }
  // Read a byte at a time until we get to the special '0x42' start-byte
  if (s->peek() != 0x42) {
    s->read();
    return false;
  }
  // Now read all 32 bytes
  if (s->available() < 32) {
    return false;
  }
  uint8_t buffer[32];
  uint16_t sum = 0;
  s->readBytes(buffer, 32);
  // get checksum ready
  for (uint8_t i=0; i<30; i++) {
    sum += buffer[i];
  }
  // The data comes in endian'd, this solves it so it works on all platforms
  uint16_t buffer_u16[15];
  for (uint8_t i=0; i<15; i++) {
    buffer_u16[i] = buffer[2 + i*2 + 1];
    buffer_u16[i] += (buffer[2 + i*2] << 8);
  }

  // put it into a nice struct :)
  memcpy((void *)&data, (void *)buffer_u16, 30);

  if (sum != data.checksum) {
    Serial.println("Checksum failure");
    return false;
  }
  // success!
  return true;
}
```

8.Full Sketch used

After all of the components been utilized for this project, this is the code for all components to be used.

The final sketch as well as all test sketches I uploaded into [github](), tank you my papa for let me know about this powerful website to store my sketches.

Schematic of my project:



Arduino nano 33 IOT server now available through Internet by this url: [https://telets.myddns.me:8443/]()

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <Arduino_LSM6DS3.h>
#include <SimpleDHT.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <Arduino.h>
```

```
#include "arduino_secrets.h"
char ssid[] = my_SSID;
char pass[] = my_PASSWORD;

#define BMP_SCK 7
#define BMP_MISO 6
#define BMP_MOSI 5
#define BMP_CS 4

Adafruit_BMP280 bme(BMP_CS, BMP_MOSI, BMP_MISO, BMP_SCK);

int status = WL_IDLE_STATUS;
int pinDHT22 = 2;
SimpleDHT22 dht22(pinDHT22);
#define pwmPin 3 // Init CO2
int prevVal = LOW;  // Init CO2
long th = 0;
long tl = 0;
long h = 0;
long l = 0;
long ppm = 0;  // Init CO2

WiFiServer server(80);

#define LENG 31   //0x42 + 31 bytes equal to 32 bytes
unsigned char buf[31];

int PM01Value=0;          //define PM1.0 value of the air detector module
int PM2_5Value=0;         //define PM2.5 value of the air detector module
int PM10Value=0;         //define PM10 value of the air detector module

void setup() {
  //Initialize serial and wait for ports to open:
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial1.setTimeout(1500);    //set the Timeout to 1500ms, longer than the data transmission periodic
time of the sensor
  Serial.println(F("BMP280 test"));
   if (!bme.begin()) {
   Serial.println("Could not find a valid BMP280 sensor, check wiring!");
   //while (1);
   }
   pinMode(pwmPin, INPUT);
   ppm = 0;

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv < "1.0.0") {
    Serial.println("Please upgrade the firmware");
```

```
  }

  // check gyroscope
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");

    while (1);
  }

  // attempt to connect to Wifi network:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  server.begin();
  // you're connected now, so print out the status:
  printWifiStatus();
}

void loop() {
  delay(1000);
  WiFiClient client = server.available();
  if (client) {
    Serial.println("new client");
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the re-
sponse
          client.println("Refresh: 30");  // refresh the page automatically every 30 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.println("<br/>");
          client.println("<h2>Tricorder - Arduino project. Writen by  <a href=https://github.com/ptelets/ar-
duino><b>Pavlo Telets</b></a></h2>");
          client.println("<i>Engage(c) Jean-Luc Picard.</i>");
          client.println("<br/>");
          client.println("<br/>");
          client.println("<br/>");
          // show DTH22
           for (int Channel = 0; Channel < 1; Channel++) {
```

```
          float temperature = 0;
          float humidity = 0;
          int err = SimpleDHTErrSuccess;
          if ((err = dht22.read2(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
          client.print("Read DHT22 failed, err="); client.println(err);delay(2);
          client.println("<br/>");
          return;
           }

          client.print("<b>Reading DHT22 sensor</b>");
          client.println("<br/>");
          if ((float)temperature > 25) {
            client.println("<i>To boldly go where no man has gone before. And switch on air conditioner.</
i>");
            client.println("<br/>");
          }
          client.print("temperature: "); client.print((float)temperature); client.print(" *C, ");
          client.println("<br/>");
          if ((float)humidity > 80) {
            client.println("<i>I canna change the laws of physics. (c) Scotty - we need an umbrella</i>");
            client.println("<br/>");
          }
          client.print("humidity: "); client.print((float)humidity); client.println(" RH%");
          client.println("<br/>");
          client.println("<br/>");

           }
        // Show BMP280
        client.println("<b>Reading BMP280 sensor: </b>");
        client.println("<br/>");
        client.print(F("Temperature = "));
        client.print(bme.readTemperature());
        client.println(" *C");
        client.println("<br/>");
        if (bme.readPressure() < 101825 && bme.readPressure() > 100825) { // 101325 - normal pres-
sure.
          client.println("<i><b>The normal atmosphere, looks like we are on our Earth.</b></i>");
          client.println("<br/>");
        } else if (bme.readPressure() < 100825) { // just -500 out from normal
                                  client.println("<i><b>Warning!</b> Low pressure, our ship is losing some
air </i>");
       client.println("<br/>");
                               } else if (bme.readPressure() > 101825) { // just +500 out from normal
                                  client.println("<i> <b>Warning!</b> High pressure, we are in a  star ship
not in a zeppelin </i>");
       client.println("<br/>");
                               }
         client.print(F("Pressure = "));
         client.print(bme.readPressure());
         client.println(" Pa");
         client.println("<br/>");
         client.print(F("Approx altitude = "));
         client.print(bme.readAltitude(1013.25)); /* Adjusted to local forecast! */
         client.println(" m");
         client.println("<br/>");
```

```
        client.println("<br/>");
        //Show CO2
        client.println("<b>Reading CO2 MH-Z19 sensor: </b>");
        client.println("<br/>");
        sensorCO2();
        if (ppm > 1200) {
          client.println("Someone set phasers to stun...");
        }
        client.println("CO2 PPM = " + String(ppm));
        client.println("<br/>");
        client.println("<br/>");
        // Show PM2.5 sensoer, air polution will be discovery, cross fingers.
        if(Serial1.find(0x42)) {    //start to read when detect 0x42
          Serial1.readBytes(buf,LENG);
          if(buf[0] == 0x4d) {
            if(checkValue(buf,LENG)) {
              PM01Value=transmitPM01(buf); //count PM1.0 value of the air detector module
              PM2_5Value=transmitPM2_5(buf);//count PM2.5 value of the 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();
        client.println("<b>Reading PM2.5 sensor: </b>");
        client.println("<br/>");
        client.print("PM1.0 um: ");
        client.print(PM01Value);
        client.println("  ug/m3");
        client.println("<br/>");
        client.print("PM2.5 um: ");
        client.print(PM2_5Value);
        client.println("  ug/m3");
        client.println("<br/>");
        client.print("PM10. um: ");
        client.print(PM10Value);
        client.println("  ug/m3");
        client.println("<br/>");
        client.println("<br/>");
        }

        // Show gyroscope
        client.println("<b>Reading internal Gyroscope sensor: </b>");
        client.println("<br/>");
        client.print("Gyroscope sample rate = ");
        client.print(IMU.gyroscopeSampleRate());
        client.println(" Hz");
        client.println("<br/>");
        client.println("Gyroscope in degrees/second");
        client.println("X\tY\tZ");
        client.println("<br/>");
        float x, y, z;
       if (IMU.gyroscopeAvailable()) {
```

```
            IMU.readGyroscope(x, y, z);
          client.print(x);
          client.print('\t');
          client.print(y);
          client.print('\t');
          client.println(z);
          client.println("<br/>");
          client.println("<br/>");
        }
        // Snow Accelerometer
        client.println("<b>Reading internal Accelerometer sensor: </b>");
        client.println("<br/>");
        client.println("<i> We can't mager our warp speed with this Tricorder...</i>");
        client.println("<br/>");
         float a, b, c;
        client.print("Accelerometer sample rate = ");
        client.print(IMU.accelerationSampleRate());
        client.println(" Hz");
        client.println();
        client.println("<br/>");
        client.println("Acceleration in G's");
        client.println("Z\tY\tZ");
        client.println("<br/>");
        if (IMU.accelerationAvailable()) {
            IMU.readAcceleration(a, b, c);
          client.print(a);
          client.print('\t');
          client.print(b);
          client.print('\t');
          client.println(c);
          client.println("<br/>");
          client.println("<br/>");
          client.println("<br/>");
        }
         // print my board's IP/SSID/etc. info:
          client.println("<br/>");
          client.println("<b>Server Info:</b>");
          client.println("<br/>");
        IPAddress ip = WiFi.localIP();
          client.print("IP Address: ");
          client.println(ip);
          client.println("<br/>");
        long rssi = WiFi.RSSI();
          client.print("signal strength (RSSI):");
          client.print(rssi);
          client.println(" dBm");
          client.println("<br/>");
        String fv = WiFi.firmwareVersion();
          client.print("Wifi firmware is ");
          client.println(fv);
          client.println("<br/>");
          client.println("Sever power <a href=https://www.arduino.cc/en/Guide/NANO33IoT><b>by Ar-
duino Nano 33 IOT</b></a>");
          client.println("</html>");
        break;
```

```
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data in ms
    delay(100);

    // close the connection:
    client.stop();
    Serial.println("client disonnected");
  }
}

void sensorCO2() {
           long tt = millis();
           int myVal = digitalRead(pwmPin);
           if (myVal == HIGH) {
             if (myVal != prevVal) {
               h = tt;
               tl = h - l;
               prevVal = myVal;
             }
           }  else {
             if (myVal != prevVal) {
               l = tt;
               th = l - h;
               prevVal = myVal;
               ppm = 5000 * (th - 2) / (th + tl - 4);
             }
           }
           delay(1000);

}

void printWifiStatus() {
//  some debug info about board
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```

```
//High math just for precaution that we are still in Canada ;)
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]))  //check 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;
}
```

9. References:

The official Arduino Nano 33 IOT quick start guide: https://www.arduino.cc/en/Guide/NANO33IoT
Arduino Programming Course Contents: https://startingelectronics.org/software/arduino/learn-to-program-course
Arduino Uno pinout https://www.circuito.io/blog/arduino-uno-pinout/
Arduino Nano 33 IOT does not support Sofware.Serial - https://stackoverflow.com/questions/57175348/softwareserial-for-arduino-nano-33-iot
https://github.com/ostaquet/Arduino-Nano-33-IoT-Ultimate-Guide

CO2 requrement and regulation:
https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms
https://www.osstf.on.ca/en-CA/services/health-safety/information-bulletins/inadequate-ventilation-and-high-co2-levels.aspx

CO2 References:
https://habr.com/ru/post/391157/
https://www.2150692.ru/faq/87-co2-mhz19-arduino

PM2.5 References:
Real-time Air Quality Index: https://aqicn.org/city/toronto/
https://learn.adafruit.com/pm25-air-quality-sensor/arduino-code
https://www.instructables.com/id/Make-one-PM25-monitor-with-Arduino-UNO/
https://how2electronics.com/interfacing-pms5003-air-quality-sensor-arduino/
https://wiki.dfrobot.com/PM2.5_laser_dust_sensor_SKU_SEN0177

DHT22 References:
https://create.arduino.cc/projecthub/mafzal/temperature-monitoring-with-dht22-arduino-15b013
https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/
https://circuitdigest.com/microcontroller-projects/arduino-nano-33-ble-sense-board-review-and-getting-started-guide

BMP280 References:
https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout/arduino-test
https://startingelectronics.org/tutorials/arduino/modules/pressure-sensor/
Pinout - https://startingelectronics.org/pinout/GY-BMP280-pressure-sensor-module/

Got some ideas from:
https://create.arduino.cc/projecthub/projects/tags/fun?page=3
https://www.instructables.com/circuits/arduino/projects/?page=31
https://www.instructables.com/id/Make-Your-Own-Retro-Nixie-Clock-With-an-RTC/

https://www.instructables.com/id/SteamPunk-Radio/
https://www.instructables.com/id/DIY-Mini-CNC-Laser-Engraver/
https://www.instructables.com/id/Tide-and-Weather-Clock/
https://www.instructables.com/id/DIY-Radioactivity-Counter-IoT-and-Eco-system/
https://www.instructables.com/id/Art-Deco-FM-Radio-Project-Using-Arduino/
https://www.instructables.com/id/ESP32-E-Paper-Thermometer/
https://www.instructables.com/id/Get-Started-Building-a-PM-Monitoring-Station/
https://www.instructables.com/id/Smart-Switch-1/
https://www.instructables.com/id/ITTT-Arduino-Vulploeg-Timer/
https://www.instructables.com/id/Simple-IOT-Alarm/
https://www.hackster.io/eben-kouao/smart-mirror-touchscreen-with-face-recognition-6c84cc
https://rootsaid.com/arduino-home-safety-monitor/