The weather station project

Project description and guidelines for teachers

Contents

1 PART A: Pedagogical Considerations

General approach/considerations
RoboScientists aims at engaging secondary school students in robotic artefact construction through interdisciplinary in nature projects. The set of the projects offer students opportunities to explore different aspects of the field of Science, Technology, Engineering Arts and Maths. Crafting/ handcrafting is a pivotal point in all the projects. Through the crafting process (highly interwoven in the robotic artefact construction) it is likely that the students will explore a number of engineering and design concepts, confront challenges and consider multiple solutions in order to achieve the results that they want.

About the Weather Station project
The project aims to design and develop a weather station to measure temperature, relative humidity, air pressure, wind speed / direction, precipitation amount / intensity / type, UV index, sun position, brightness and twilight and global radiation. For the purposes of this project the weather station will be programmed to measure temperature, humidity, Co2 emissions. For example, the weather station will be programmed to measure the temperature in different rooms of the school, reports the average temperature in the room/class and considers whether this is according to the suggested EU recommendations, measures the humidity levels indoor and outdoor, the CO2 emissions etc. This is a very useful project as it can be applied in various teaching fields such as Mathematics, Science, Chemistry, Geography for experiments, measurements and observations. The weather station project provides the opportunity for learning to be achieved through embodied practice.

When developing the designing and setting up the project, it is important to have in mind the presentations provided and discussions that took place during the training sessions. It is recommended to the educators that for a project to be delivered it is important to: employ the makeology approach, work in teams, encourage experimentation, involves crafting and coding, apply the Engineering design process is employed, encourage sharing, employ the STEAM approach, design and develop robotics models and artefacts, use various tool, equipment and materials, involve students as makers.

The 1^{st} category: 21^{st} century and transversal skills
The 21st century and transversal skills have been outlined and described in the literature by various researchers (e.g. Bybee & Fuchs, 2006; Ananiadou & Claro, 2009; Trilling & Fadel, 2009; Mojika, 2010; Rotherham & Willingham, 2010; Griffin & Care, 2015) and reports from ministries of education, policies and organizations (UNESCO 2014, 2016). These are the following: Communication, collaboration, critical thinking, problem solving, knowledge construction, creativity, innovation, self-directed learning, global citizenship and digital literacy. In the section below, definitions, descriptions and characteristics of the main 21^{st} century and transversal skills are given.

Learn how to learn: It is a very important skill to learn how to acquire knowledge and skills on their own and manage to construct their own knowledge and meanings.

Investigation: Investigation can be defined quite simply as a systematic fact finding and reporting process. It is derived from the Latin word vestigere, to "track or trace," and encompasses a patient, step-by step inquiry. Investigation is finding facts; it is akin to research conducted in the academic arena. In addition, it is a multi-disciplined field of study. It encompasses law, the sciences, communications, and a host of other things. Finally, it requires an inquisitive mind coupled with an attention to detail.

Exploration: Exploration-based learning is an active learning approach. Students' abilities are dynamically balanced with difficulty level in the system to provide exhilarating and fulfilling learning experiences. The visually and intellectually compelling storylines within the environment challenge each student to leverage their own curiosity and passion to solve complex problems using data and evidence, to form arguments and reach conclusions. This approach is positioned to deliver high levels of engagement and concentration while reducing stress and boredom for all students. Through these experiences, students build their levels of confidence and creativity, resulting in improved performance and sustained motivation to learn.

Reflection: Reflecting helps you to develop your skills and review their effectiveness, rather than just carry on doing things as you have always done them. It is about questioning, in a positive way, what you do and why you do it and then deciding whether there is a better, or more efficient, way of doing it in the future.

Problem Solving: Problem-solving skills help students determine the source of a problem and find an effective solution. Although problem-solving is often identified as its own separate skill, there are other related skills that contribute to this ability.

Critical Thinking: Critical thinking is not a matter of accumulating information. A person with a good memory and who knows a lot of facts is not necessarily good at critical thinking. A critical thinker is able to deduce consequences from what he knows, and he knows how to make use of information to solve problems, and to seek relevant sources of information to inform himself.

Digital literacy: Digital literacy refers to a particular set of competencies that allow you to function and participate fully in a digital world. Students, nowadays, are generally considered to be digital natives - able to use technology effectively and easily. They must be able to resolve conflicts, source material ethically and interact with the wider world in a responsible manner.

Creativity: Creativity simply means being able to come up with something new. Therefore, creative thinking is the ability to consider something – a conflict between employees, a data set, a group project – in a new way. The term is referring to the act of turning new and imaginative ideas into reality involves two processes: thinking, then producing. Finally, creativity is characterized by the ability to perceive the world in new ways, to find hidden patterns, to make connections between seemingly unrelated phenomena, and to generate solutions.

Innovation: Innovation skills refer to the talent of exploiting new ideas for the purpose of gaining social or economic value. Innovation skills are usually a combination of one's ability to think creatively, problem-solving ability, as well as functional and/or technical abilities. Fairly speaking, innovation skills are basically one's ability to apply a blend of knowledge, skills and attributes in a specific context.

Cooperation/ Collaboration: Cooperation is a division of labour between-group members. It occurs when a task is divided up into individually manageable subparts, which are subsequently constructed into a final outcome. Although this is conceptually different to collaboration, at a fine-grained level, all collaborative tasks have a degree of cooperation (Lai & Viering, 2012).

Communication: Communication is the art of transmitting information, thoughts and attitudes from one person to a different one's. It is the route of meaningful interaction among human beings. We learn basic communication skills by observing other people and modelling our behaviours based on what we see.

Building knowledge: Knowledge building provides an alternative that more directly addresses the need to educate people for a world in which knowledge creation and innovation are pervasive. Knowledge building may be defined as the production and continual improvement of ideas of value to a community, through means that increase the likelihood that what the community accomplishes will be greater than the sum of individual contributions and part of broader cultural efforts. Knowledge building, thus, goes on throughout a knowledge society and is not limited to education.

The 2nd category: General Pedagogical Skills / Objectives
The second category of skills are the General pedagogical ones. These are the skills to be developed or in other words the general pedagogical objectives of the Curricula of various subject matters. They are mainly outlined within the Curricula of various subject matters and specifically from subject matters such as Mathematics, Science, Technology, Engineering, Social Sciences, Arts and Linguistics.

General skills

Information Management Skills: Students make various calculations and metrics, make estimates and use graphs, tables, charts, and more optical media, to manage the various information and solve the problems which are presented. Also, students communicate with different ideas, criteria, possible solutions and outcomes. This communication takes place through sketches, graphs and representations on paper and computer, making two-dimensional and three-dimensional models and prototypes through symbolic and verbal representations. At the same time, they recognize, organize, analyse, compile and evaluate data information and interpret different views and approaches.

Problem Solving Skills: The Design and Technology Study Program is particularly useful for developing problem solving skills. Ploblem solving is closely related to the development of critical, reflective and logical thinking mindset, the development of imagination and creativity, problem determination and analysis, the exploration, construction and control of products and constructions, the evaluation of processes and products.

Project Management Skills: Through cross-thematic activities proposed and implemented through teamwork, pupils can develop skills in targeting, managing time and the available resources, computation, risk-taking and dispute resolution.

Social and Interpersonal Skills: The proposed activities as well as the respective framework offer a rich and authentic communication environment between pupils and teachers, working in groups, respect and cooperation, etc.

Skill Category: Design

Middle School

1. Ask appropriate questions and through ideas of stature propose ideas for various constructions and procedures.
2. They discuss ready-made technology products, referring to their form, function and safety.
3. Analyse the factors that affect a problem, through the collection and utilization of various information.
4. Report and develop problem-solving ideas, taking into account security, ergonomics, aesthetics, economy, applying the design process.
5. Carry out research and evaluate sources and information about a particular product or process.
6. Evaluate products and processes based on criteria that have been set.
7. Apply the stages of the design process.
8. Recognize and use symbols in diagrams, circuits and drawings, in applications on paper and on PC.

High School

1. Investigate and evaluate industrial products and processes based on specifications.
2. Implement a manufacturing process according to the product they are going to manufacture.
3. Draw up an action plan and implement the planning process stages.

4. Evaluate products based on specifications and needs that have been put forward and propose modifications.
5. Report and document modifications and variations made during the design and construction phases and explain the necessity of these differentiations.

Skill Category: Communication

Middle School

1. Describe verbatim and / or design the design process for ideas to be implemented.
2. Use lines, shapes, and simple design methods to present their ideas.
3. Recognize and use symbols that recognize within diagrams, circuits and patterns.
4. Communicate using sketches and 3D drawings and spelling projections.
5. Communicate using recognized symbols.

High School

1. Enhance their designs by adding information through detailed three-dimensional drawings and magnifications.
2. They present ideas and ways of construction through three-dimensional drawings and spelling projections.

Skill Category: Construction

Middle School

1. Collect and categorize materials from simple constructions.
2. Prepare simple constructions with various materials, using different skills and manufacturing methods.
3. Cut, bind and shape materials to use in simple constructions.
4. Mark, cut and assemble with precision various materials.
5. Safely use a range of tools and machines to manufacture products made up of more than one kind of materials.

High School

1. Use manufacturing techniques, materials, tools and machinery in a way that appears to be familiar with manufacturing processes, taking into account safety during manufacture and quality assurance of the final product.
2. Propose and apply alternatives to implement their ideas.

Additionally, exploratory skills are promoted through the Curricula (Programs of Study) of Secondary Education. The exploratory skills are summarized below:

- Writing hypotheses that can be checked.
- Designing and conducting research, determining which variables will change, what will remain stable and what will be measured.
- Selecting appropriate tools, technological equipment and suitable materials for a construction.
- Presenting and interpreting the results using a range of representations and dynamic images, simulations and models.
- Communicating results and explaining structures to classmates and other audiences / users, using appropriate vocabulary.
- Evaluating ready-made technology products and suggesting improvements.
- Presenting a design and explaining the use of the finished product.

The 3^{rd} category: The Learning Objectives
The third category includes the Learning Objectives to be achieved within various subject matters, or in other words across various disciplines. The Theremin project aims to achieve learning objectives from the disciplines/subject matters of Music, History, Language and Literature, Maths, Physics and ICT :

- **ICT skills:** programming, connecting physical and digital world through the use and synchronization of multiple sensors.
- **Chemistry & Geography:** CO2, temperature, air, earth, humidity, meteorology, weather conditions, different climates, changes in the atmosphere.
- **Environmental Education:** EU and government policies, which factors affect the climates and the weather conditions.
- **Physics:** electrical circuit making, understanding what a motor is and how it works, controlling motion.
- **Maths:** measurements, variables, observation, complete a worksheet, database.
- **Language and Literature:** brainstorm, discuss and answer various questions develop reports, present, i.e. How does climate/ weather is changing? How does weather affect our daily life activities?

The process

The process to be followed from the students is the Engineering Design Process as presented in the following two diagrams.

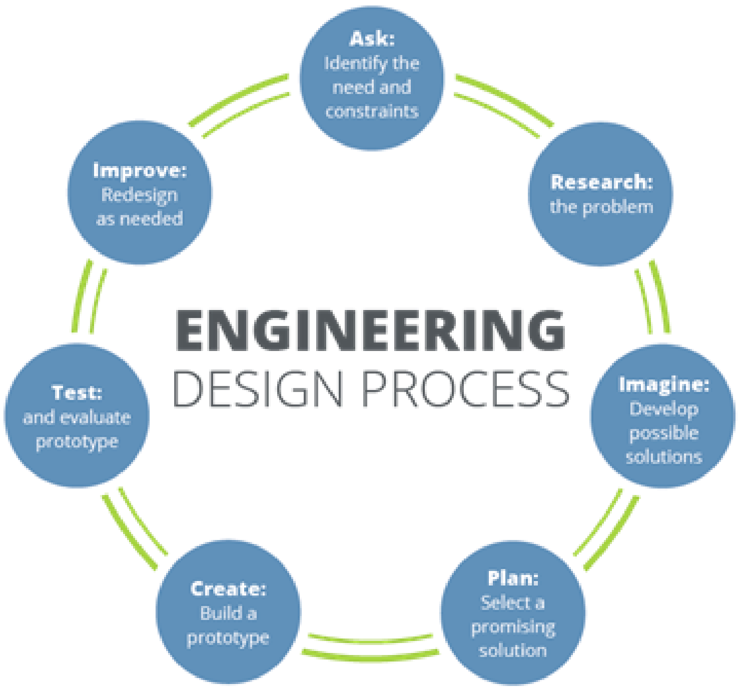
Ask: Identify the need and constraints
Research: the problem
Imagine: Develop possible solutions
Plan: Select a promising solution
Create: Build a prototype
Test: and evaluate prototype
Improve: Redesign as needed
ENGINEERING DESIGN PROCESS

Figure 1: The Engineering Design Process.

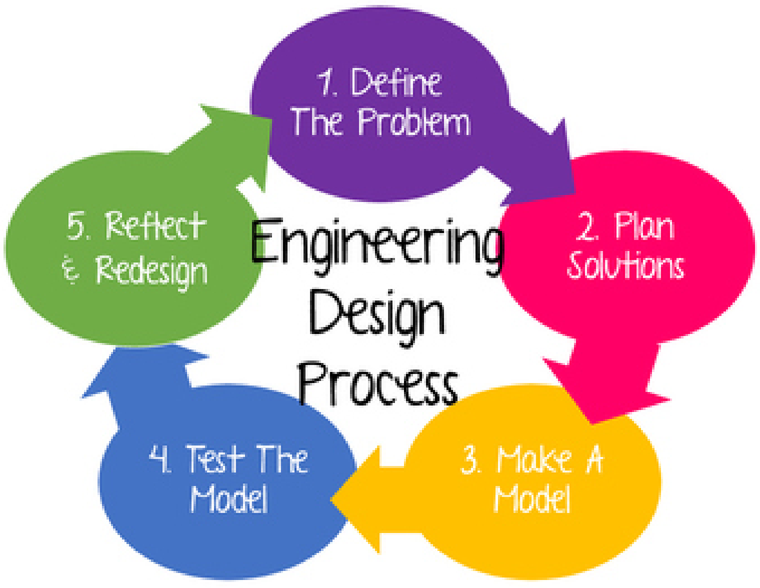

Figure 2: The Engineering Design Process.

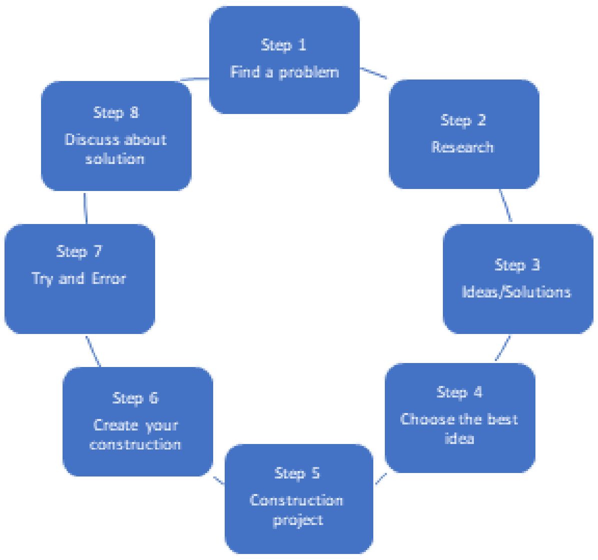

Figure 3: The Design process according the Design and Technology Curriculum (Cyprus Ministry of Education and Culture).

Additional technical tips

For more technical details and information about the selected technologies and tools please see O1 (technical tutorial: `http://www.roboscientists.eu/outputs/output-1/`).

References

Ananiadou, K. & Claro, M. (2009), ”21st Century Skills and Competences for New Millennium Learners in OECD Countries”, OECD Education Working Papers, No. 41, OECD Publishing, Paris. DOI: http://dx.doi.org/10.1787/218525261154

Bybee, R. W., & Fuchs, B. (2006). Preparing the 21st century workforce: A new reform in science and technology education. J. Res. Sci. Teach., 43: 349–352. doi: 10.1002/tea.20147

Griffin, P., & Care, E. (Eds) (2105). Assessment and Teaching of 21st Century Skills, Methods and Approach. Dordrecht: Springer. DOI: 10.1007/978-94-017-9395-7

Mojica, K.D. (2010). Ordered effects of technology education units on higher-order critical thinking skills of middle school students (Doctoral dissertation). Retrieved from: ProQuest Dissertation and Theses database.

Rotherham, J. A., & Willingham, D. T. (2010). ”21st-Century” Skills: Not New, but a Worthy Challenge. American Educator, 34 (1), p. 17-20.

Trilling, B., & Fadel, C. (2009). 21st Century Skills: Learning for Life in Our Times. San Francisco, CA: Jossey-Bass.

UNESCO. (2014). ‘Teaching and Learning: Achieving quality for all’. Education for All Global Monitoring Report, UNESCO, Paris (2014) UNESCO (2016). A Global measure of digital and ICT literacy skills. Background paper prepared for the 2016 Global education monitoring report, Education for people and planet: creating sustainable futures for all, UNESCO, Paris (2016).

2 PART B: Practice

2.1 Level 1: Pressure, temperature and humidity measurements

2.1.1 Assembling the circuit

During this project, the Grove series of sensors will be used. This solution makes connection of elements easier and fault tolerant. Otherwise making connections wire by wire are error prone and in some circumstances may damage the sensors. Therefore, special shield is needed. If you use Arduino Uno board, you will need *Grove Base Shield*. If you use Arduino Mega 2560 board, you will need *Grove Mega Shield*. The second necessary element is *Grove Barometer Sensor (BME280)*, which consists of temperature, pressure and humidity sensors.

First, the Grove Shield should be connected to Arduino board as is shown in Fig. 4. If you use Arduino Mega board, the connections is similar.

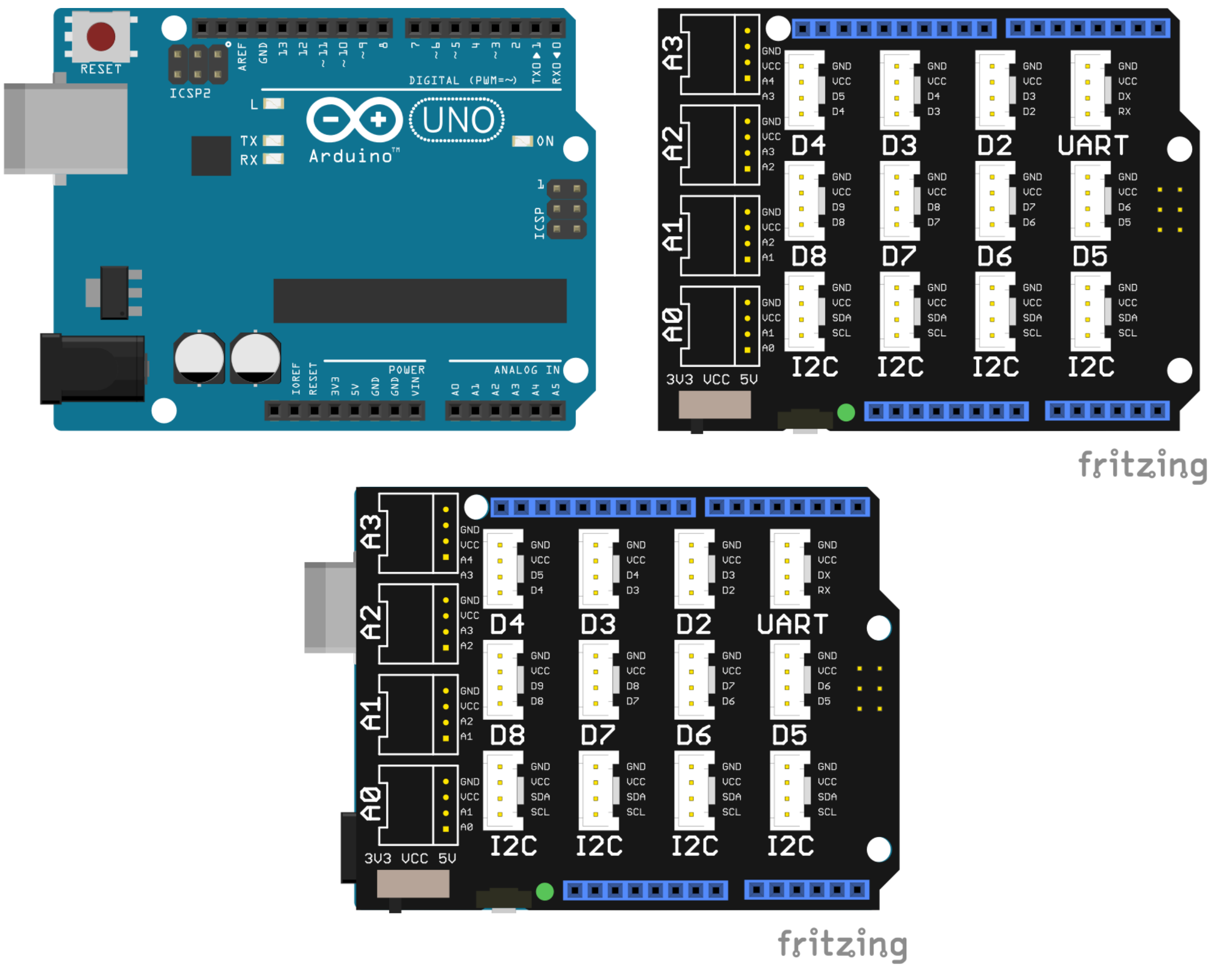

Figure 4: Top figure: Arduino Uno board and Grove Base Shield separately. Bottom figure: Grove Base Shield connected with Arduino Uno board (Grove Base Shield put on top of Arduino Uno board).

Secondly, the BME280 sensor should be connected to the shield as is shown in Fig. 5.

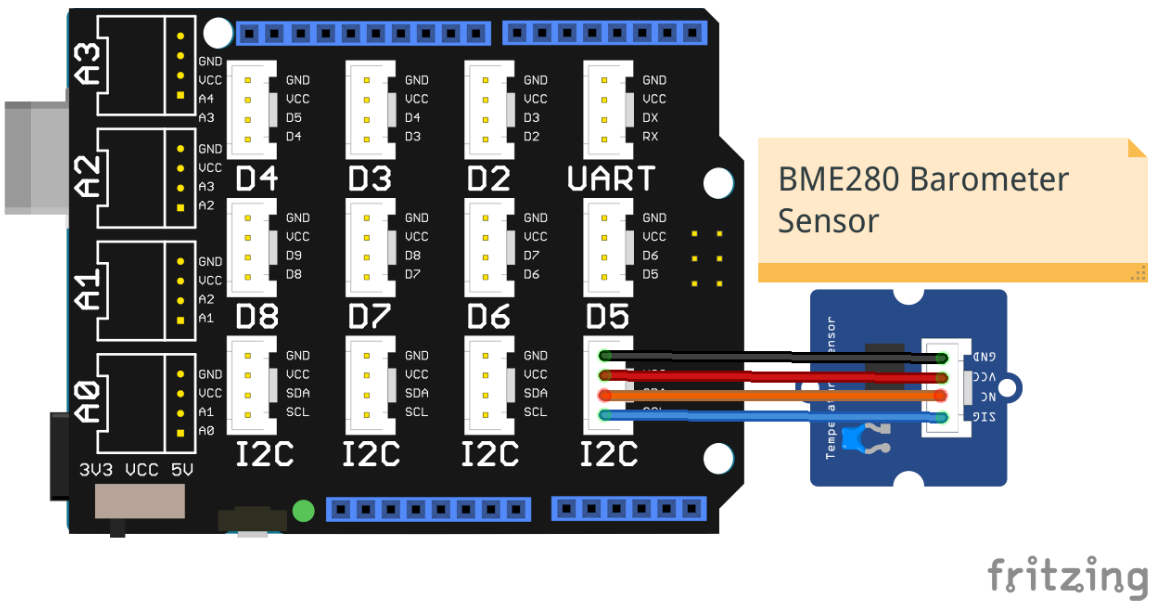

Figure 5: The BME280 sensor connected to the Grove shield.

The connectors (sockets and plugs) have a special shape. They fit in only one way, so it is impossible to connect them wrong.

Questions that can be raised/discussed:

- How temperature can be measured?
- How pressure can be measured?
- What values of temperature, pressure and humidity are optimal for people?

2.1.2 Towards Arduino IDE solution

The goal of this level is acquiring the temperature, pressure and humidity values from BME280 sensor. The measured values should be listed in serial monitor.

You will need the special libraries for this sensor, which can be download from the page: https://github.com/Seeed-Studio/Grove_BME280 . After downloading library, save it to the *libraries* folder inside the Arduino IDE folder. Than, unzip the library and run again the Arduino IDE.

The program was divided into following steps:

1. First the necessary libraries are added:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

void setup() {
  // put your setup code here, to run once:
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
}
```

2. The object of BME280 class is created:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

3. Configure the serial monitor, which will be used to print values and comments:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

4. The BME280 sensor is initialized inside *setup* function. In this function, all commands are run only once. If initialization fails, *init* function returns *false*. The ! sign means negation therefore, *false* value will be changed to *true*. If $!bme280.init()$ is equal to true, then Serial.println will execute and print information on serial monitor about problem with connection:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port
  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");
}
```

```
void loop() {
// put your main code here, to run repeatedly:
}
```

5. Create variables, where values will be stored. The temperature and humidity are floating variables and humidity is integer variable:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port
  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

6. Read variables from sensor and store in created variables:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port
  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();

  //Read pressure value and print to serial port
  p=bme280.getPressure();
```

```
    //Read humidity value and print to serial port
    H=bme280.getHumidity();

    delay(1000);
  }
```

7. Display values in serial monitor. First line will inform about the order of appearance of measured quantities. This line is put inside the *setup* function, because it should be executed only once. Then, the read values from sensors are printed in serial monitor inside the *loop* function, which repeats these actions continuously. The *print* function just prints value in serial monitor. The *println* function prints value in serial monitor and add new line character after it. The \t is a tabular character:

```
  #include "Seeed_BME280.h" // Including libraries
  #include <Wire.h>

  float temp; //Create variables to store values
  float p;
  int H;

  BME280 bme280; //Create object of BME280 class

  void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600); //Configure baud rate of serial port

    //Initialize connection with sensor
    if(!bme280.init()) Serial.println("Error with connection!");

    //Print in serial port, which values will be read
    Serial.println("T (C) \t p (Pa) \t H (%)");
  }

  void loop() {
    // put your main code here, to run repeatedly:
    //Read temperature value and print to serial port
    temp=bme280.getTemperature();
    Serial.print(temp);
    Serial.print("\t");

    //Read pressure value and print to serial port
    p=bme280.getPressure();
    Serial.print(p);
    Serial.print("\t");

    //Read humidity value and print to serial port
    H=bme280.getHumidity();
    Serial.println(H);

    delay(1000);
```

```
}
```

8. Delay the loop. In each measurement phase, we need to stop for a while. The reasonable delay in our project is about 1 second, so our measurement values will be acquired once a second. We use *delay(1000)* function which gets argument in milliseconds, so we have 1 second delay. This ensures that sensor has enough time for measurement and providing valid value to the user. Another thing is that we don't need much data in such measurements, especially if they don't change quickly, so there is no need to store the same repeating values.

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port

  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");

  //Print in serial port, which values will be read
  Serial.println("T (C) \t p (Pa) \t H (%)");
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure();
  Serial.print(p);
        Serial.print}("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.println(H);

  delay(1000);
}
```

2.2 Level 2: Measurement of dust concentration

2.2.1 Assembling the circuit

The second level is focused on reading the dust concentration based on optical sensor. This sensor has IR LED inside, which emits IR light. If dust is present, the light bounces off it and the light is registered by photodiodes. If intensity of light is higher, the analog signal returned from sensor has higher voltage.

The relation between dust concentration and voltage is mostly linear (see Fig. 6).

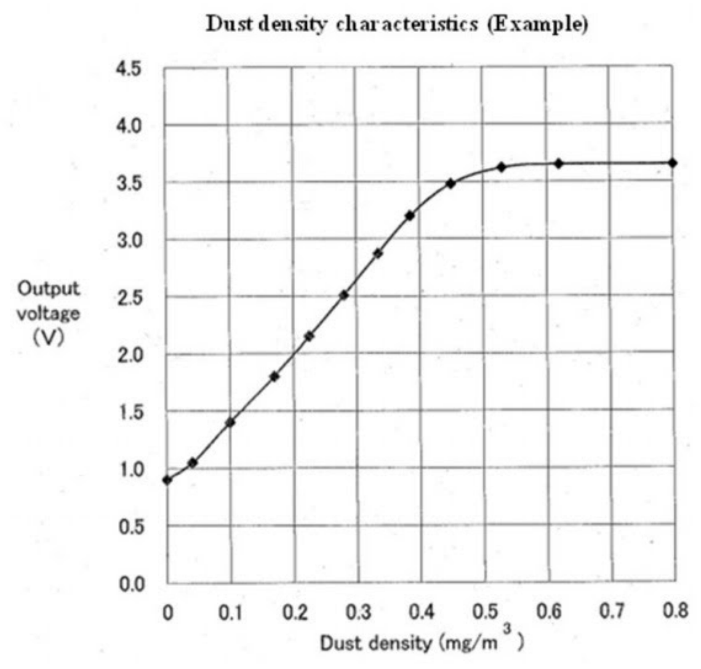

Figure 6: The characteristic of dust sensor. Source: http://www.theorycircuit.com/dust-sensor-arduino-interface/ .

In this level, the optical, dust sensor GP2Y1010AU0F will be used. The sensor should be connected to PWM output and analog input as is shown in Fig. 7.

Questions that can be raised/discussed:

- How is dust concentration measured by the sensor?
- What is a difference between PM2.5 and PM10?
- What do you thing is the main source of dust in the atmosphere?

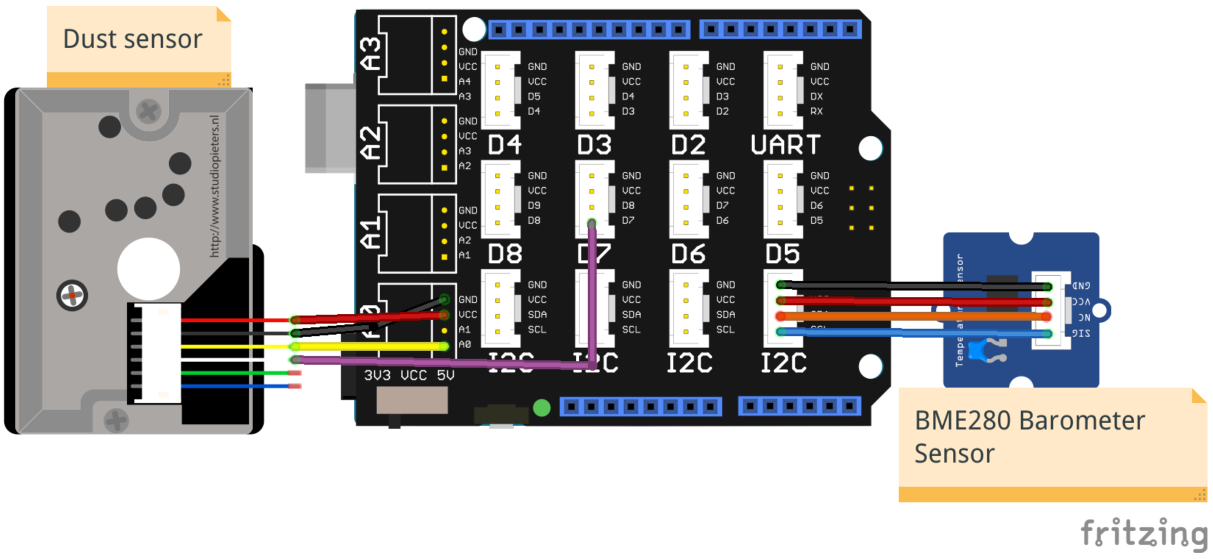

Figure 7: The dust sensor connected to the Grove shield. Red line should be connected to 5 V (VCC). Black line should be connected to ground (GND). The analog output (yellow line) (AOUT) should be connected to the analog pin. The LED input (white line) should be connected to the PWM pin.

2.2.2 Towards Arduino IDE solution

During this level, the code from level 1 will be modified by adding commands, which will be used to read and calculate the dust concentration.

The code was divided into following steps:

1. First, the necessary variables are created:

   - dust_output - which will store read value from analog pin (ADC - Analog to Digital Conversion),
   - dust_voltage - which will store calculated voltage value returned by sensor,
   - dust - which will store dust concentration,
   - LED - which will store PWM pin number to which LED is connected,
   - DUSTpin - which will store analog pin number to which analog output from sensor is connected.

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port

  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");

  //Print in serial port, which values will be read
  Serial.println("T (C) \t p (Pa) \t H (%)");
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure();
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.println(H);

  delay(1000);
}
```

2. At the beginning, the LED diode should be configured as *OUTPUT* and turn off. The returned analog signal by dust sensor has small voltage. Therefore, standard range of voltage (0-5V) is too high and cause large inaccuracy. This reference voltage can be changed from 5V to 1.1V by *analogReference* function. This will greatly help in reducing the noise. However by this limit we will also loose the dynamic range of sensor. Fortunately this is not a problem if we want to monitor the relatively clean air.

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
```

```
 float dust_voltage;
 float dust;

 int LED=7; //Dust and LED pin numbers
 int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

 void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600); //Configure baud rate of serial port

   //Initialize connection with sensor
   if(!bme280.init()) Serial.println("Error with connection!");

   //Change reference voltage from 5V to 1.1V
   //option: INTERNAL for Arduino UNO
   //option: INTERNAL1V1 for Arduino Mega
   analogReference(INTERNAL);
   pinMode(LED,OUTPUT);
   digitalWrite(LED,LOW); //Turn off LED diode

   //Print in serial port, which values will be read
   Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
 }

 void loop() {
   // put your main code here, to run repeatedly:
   //Read temperature value and print to serial port
   temp=bme280.getTemperature();
   Serial.print(temp);
   Serial.print("\t");

   //Read pressure value and print to serial port
   p=bme280.getPressure();
   Serial.print(p);
   Serial.print("\t");

   //Read humidity value and print to serial port
   H=bme280.getHumidity();
   Serial.println(H);

   delay(1000);
 }
```

3. Measurement of the dust concentration. First, we should turn on LED diode. Than, wait 280 μs to give a time to the sensor for light registering. After the delay, the analog value can be read from sensor. The last step is to turn off the LED diode.

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

 float temp; //Create variables to store values
 float p;
 int H;

 int dust_output;
 float dust_voltage;
 float dust;

 int LED=7; //Dust and LED pin numbers
 int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

 void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600); //Configure baud rate of serial port

   //Initialize connection with sensor
   if(!bme280.init()) Serial.println("Error with connection!");

   //Change reference voltage from 5V to 1.1V
   //option: INTERNAL for Arduino UNO
   //option: INTERNAL1V1 for Arduino Mega
   analogReference(INTERNAL);
   pinMode(LED,OUTPUT);
   digitalWrite(LED,LOW); //Turn off LED diode

   //Print in serial port, which values will be read
   Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
 }

 void loop() {
   // put your main code here, to run repeatedly:
   //Read temperature value and print to serial port
   temp=bme280.getTemperature();
   Serial.print(temp);
   Serial.print("\t");

   //Read pressure value and print to serial port
   p=bme280.getPressure();
   Serial.print(p);
   Serial.print("\t");

   //Read humidity value and print to serial port
   H=bme280.getHumidity();
   Serial.println(H);

   //Measure dust
   digitalWrite(LED,HIGH);
   delayMicroseconds(280);
   dust_output=analogRead(DUSTpin);
   digitalWrite(LED,LOW);

   delay(1000);
 }
```

4. In this step, the read from sensor analog value will be converted to voltage value. The Arduino board returns analog value in range (0;1023) - 1024 possible values. The maximum read voltage value (as set by analogReference function) is 1.1 V (1100 mV) therefore voltage (in mV) can be calculated as:

$$U_{dust} = \frac{1100}{1024} A \tag{1}$$

where: A is analog value returned by Arduino board.

According to the dust sensor documentation, the divider was used. Therefore, the final value should be multiplied by 11:

$$U_{dust} = 11 \cdot \frac{1100}{1024} A \tag{2}$$

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); //Configure baud rate of serial port

  //Initialize connection with sensor
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Turn off LED diode

  //Print in serial port, which values will be read
  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure();
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.println(H);

  //Measure dust
  digitalWrite(LED,HIGH);
  delayMicroseconds(280);
  dust_output=analogRead(DUSTpin);
  digitalWrite(LED,LOW);

  //Calculate voltage
  //Multiply voltage by 11 because voltage divider was used
  dust_voltage = (1100.0/1024.0)*dust_output*11;

  delay(1000);
}
```

5. According to the dust sensor documentation, the minimal voltage returned by sensor is 600 mV. Therefore the dust concentration can be calculated as:

$$dust = 0.2 \cdot (U_{dust} - 600mV) \tag{3}$$

Calculation of the dust concentration is added and value is printed in serial monitor:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");

}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  //Measure dust
  digitalWrite(LED,HIGH);
  delayMicroseconds(280);
  dust_output=analogRead(DUSTpin);
  digitalWrite(LED,LOW);

  //Calculate voltage
  //Multiply voltage by 11 because voltage divider was used
  dust_voltage = (1100.0/1024.0)*dust_output*11;

  //Calculate dust when voltage is higher than minimal
  if(dust_voltage>600) dust=(dust_voltage-600)*0.2;
  else dust=0.0;
  Serial.println(dust);

  delay(1000);
}
```

This solution is simple but we can observe significant fluctuations of dust concentration value. This problem can be solved by calculating average value of dust concentration based on e.g. 10 measurements. Additionally, because the dust measurement is long, therefore it should be enclosed in separate function. In our case it is *getDust()* function.

Modified code is shown here:

```
#include "Seeed_BME280.h" // Including libraries
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}
float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
    //Multiply voltage by 11 because voltage divider was used
    dust_voltage = (1100.0/1024.0)*dust_output*11;

    //Calculate dust when voltage is higher than minimal
    if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
    delay(500);
  }

  return (averageDust/10.0);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  dust = getDust();
  Serial.println(dust);

  delay(1000);
}
```

2.3 Level 3: Visualization of the measurements

2.3.1 Assembling the circuit

This level is focused on visualization of the measurements. The weather station should be portable. Therefore, the LCD RGB screen will be added to print the values. After adding LCD screen, the weather station can be disconnected from computer. The simple power bank can be used to power on the weather station.

The LCD RGB screen connected to Arduino board is shown in Fig. 8.

2.3.2 Towards Arduino IDE solution

During this level, the improved code from level 2 will be modified by adding commands to display temperature, pressure, humidity and dust concentration on the LCD RGB screen.

The LCD RGB screen needs dedicated libraries, which can be downloaded from the page: `https://github.com/Seeed-Studio/Grove_LCD_RGB_Backlight` . As previously, the library should be download and stored in *libraries* folder inside Arduino IDE folder. Then, the library should be unzipped and Arduino IDE should be restarted.

The code was divided onto following steps:

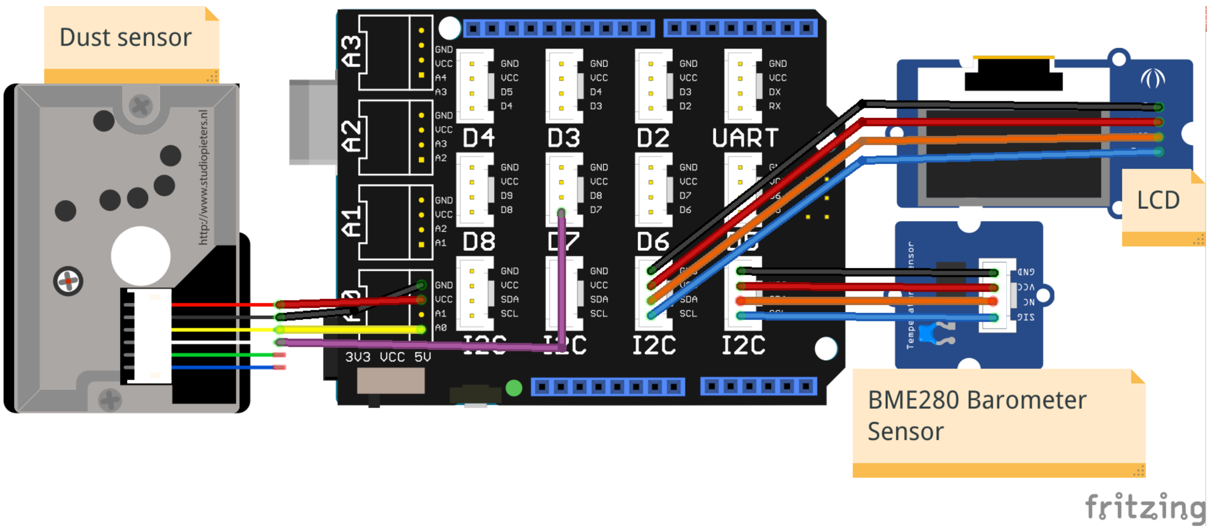

Figure 8: The LCD RGB screen connected to shield.

1. First, the dedicated library for LCD RGB screen is added:

```
#include "Seeed_BME280.h" // Including libraries
#include "rgb_lcd.h" // Including the library for LCD
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

BME280 bme280; //Create object of BME280 class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}
```

```
float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
    //Multiply voltage by 11 because voltage divider was used
    dust_voltage = (1100.0/1024.0)*dust_output*11;

    //Calculate dust when voltage is higher than minimal
    if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
    delay(500);
  }

  return (averageDust/10.0);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  dust = getDust();
  Serial.println(dust);

  delay(1000);
}
```

2. Next, the object of LCD RGB class is created. Additionally, the variables which store color numbers in RGB were defined:

   - colorR - contribution of red color in range 0-255,
   - colorG - contribution of green color in range 0-255,

- colorB - contribution of blue color in range 0-255.

```
#include "Seeed_BME280.h" // Including libraries
#include "rgb_lcd.h" // Including the library for LCD
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

int colorR = 255;
int colorG = 0;
int colorB = 0;

BME280 bme280; //Create object of BME280 class
rgb_lcd lcd; //Create object of rgb_lcd class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}

float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
```

```
      //Multiply voltage by 11 because voltage divider was used
      dust_voltage = (1100.0/1024.0)*dust_output*11;

      //Calculate dust when voltage is higher than minimal
      if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
      delay(500);
    }

    return (averageDust/10.0);
  }

  void loop() {
    // put your main code here, to run repeatedly:
    //Read temperature value and print to serial port
    temp=bme280.getTemperature();
    Serial.print(temp);
    Serial.print("\t");

    //Read pressure value and print to serial port
    p=bme280.getPressure()/100.0;
    Serial.print(p);
    Serial.print("\t");

    //Read humidity value and print to serial port
    H=bme280.getHumidity();
    Serial.print(H);
    Serial.print("\t");

    dust = getDust();
    Serial.println(dust);

    delay(1000);
  }
```

3. The LCD size and background color should be defined inside *setup* function:

```
  #include "Seeed_BME280.h" // Including libraries
  #include "rgb_lcd.h" // Including the library for LCD
  #include <Wire.h>

  float temp; //Create variables to store values
  float p;
  int H;

  int dust_output;
  float dust_voltage;
  float dust;

  int LED=7; //Dust and LED pin numbers
  int DUSTpin=0;

  int colorR = 255;
  int colorG = 0;
  int colorB = 0;
```

```
BME280 bme280; //Create object of BME280 class
rgb_lcd lcd; //Create object of rgb_lcd class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  //option: INTERNAL for Arduino UNO
  //option: INTERNAL1V1 for Arduino Mega
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin

  //Define size of LCD and colors
  lcd.begin(16,2);
  lcd.setRGB(colorR, colorG, colorB);

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}

float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
    //Multiply voltage by 11 because voltage divider was used
    dust_voltage = (1100.0/1024.0)*dust_output*11;

    //Calculate dust when voltage is higher than minimal
    if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
    delay(500);
  }

  return (averageDust/10.0);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");
  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  dust = getDust();
  Serial.println(dust);

  delay(1000);
}
```

4. The last step is displaying all values on the screen. First cursor should be put at the beginning of screen by function *setCursor*. The first value inside function is a row number. The second value is a column number. Next, values are displayed by function *print*:

```
#include "Seeed_BME280.h" // Including libraries
#include "rgb_lcd.h" // Including the library for LCD
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

int colorR = 255;
int colorG = 0;
int colorB = 0;

BME280 bme280; //Create object of BME280 class
rgb_lcd lcd; //Create object of rgb_lcd class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin
  //Define size of LCD and colors
  lcd.begin(16,2);
  lcd.setRGB(colorR, colorG, colorB);

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");

}

float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
    //Multiply voltage by 11 because voltage divider was used
    dust_voltage = (1100.0/1024.0)*dust_output*11;

    //Calculate dust when voltage is higher than minimal
    if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
    delay(500);
  }

  return (averageDust/10.0);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  dust = getDust();
  Serial.println(dust);

  lcd.setCursor(0,0);
```

```
    lcd.print(temp);
    lcd.print("C ");
    lcd.print(dust);
    lcd.print("ug/m3");
    lcd.setCursor(0,1);
    lcd.print("");
    lcd.print(p);
    lcd.print("hPa ");
    lcd.print(H);
    lcd.print("%");

    delay(1000);
  }
```

The above code can be modified by adding new visually appealing functionality. The LCD RGB screen can change color depending on the temperature. The different colors can be stored in table. The modification is shown here:

```
#include "Seeed_BME280.h" // Including libraries
#include "rgb_lcd.h" // Including the library for LCD
#include <Wire.h>

float temp; //Create variables to store values
float p;
int H;

int dust_output;
float dust_voltage;
float dust;

int LED=7; //Dust and LED pin numbers
int DUSTpin=0;

int colorR[] = {0,0,0,255,255,255};
int colorG[] = {128,255,255,255,170,0};
int colorB[] = {255,255,0,0,0,0};

BME280 bme280; //Create object of BME280 class
rgb_lcd lcd; //Create object of rgb_lcd class

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  if(!bme280.init()) Serial.println("Error with connection!");

  //Change reference voltage from 5V to 1.1V
  analogReference(INTERNAL);
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW); //Set 0V to LED pin
  //Define size of LCD and colors
  lcd.begin(16,2);

  Serial.println("T (C) \t p (Pa) \t H (%) \t Dust (ug/m3)");
}

float getDust()
{

  float averageDust=0.0;

  for(int i=0; i<10; i++)
  {
    //Measure dust
    digitalWrite(LED,HIGH);
    delayMicroseconds(280);
    dust_output=analogRead(DUSTpin);
    digitalWrite(LED,LOW);

    //Calculate voltage
    //Multiply voltage by 11 because voltage divider was used
    dust_voltage = (1100.0/1024.0)*dust_output*11;

    //Calculate dust when voltage is higher than minimal
    if(dust_voltage>600) averageDust+=(dust_voltage-600)*0.2;
    delay(500);
  }

  return (averageDust/10.0);
}

void loop() {
  // put your main code here, to run repeatedly:
  //Read temperature value and print to serial port
  temp=bme280.getTemperature();
  Serial.print(temp);
  Serial.print("\t");

  //Read pressure value and print to serial port
  p=bme280.getPressure()/100.0;
  Serial.print(p);
  Serial.print("\t");

  //Read humidity value and print to serial port
  H=bme280.getHumidity();
  Serial.print(H);
  Serial.print("\t");

  dust = getDust();
  Serial.println(dust);

  // Changing background color
  if(temp<0) lcd.setRGB(colorR[0], colorG[0], colorB[0]);
  if(temp>0 && temp<5) lcd.setRGB(colorR[1], colorG[1], colorB[1]);
  if(temp>5 && temp<15) lcd.setRGB(colorR[2], colorG[2], colorB[2]);
  if(temp>15 && temp<25) lcd.setRGB(colorR[3], colorG[3], colorB[3]);
  if(temp>25 && temp<30) lcd.setRGB(colorR[4], colorG[4], colorB[4]);
  if(temp>30) lcd.setRGB(colorR[5], colorG[5], colorB[5]);

  lcd.setCursor(0,0);
  lcd.print(temp);
  lcd.print("C ");
  lcd.print(dust);
  lcd.print("ug/m3");
  lcd.setCursor(0,1);
  lcd.print("");
  lcd.print(p);
  lcd.print("hPa ");
  lcd.print(H);
  lcd.print("%");

  delay(1000);
}
```

2.4 Level 4: Interpretation of collected data

If you run above program and wait several minutes, you will collect much data. This level will focus on interpretation of the data by calculation of the average values and uncertainties of measured quantities (temperature, pressure, humidity and dust concentration).

Analysis of the data consists of the following steps::

1. Calculation of the average value of chosen quantities by using equation:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{4}$$

where: x_i is a single measured value, N is a number of measurements.

Example:

- Measured temperature points: 24.3, 24.9, 23.5, 24.0, 24.6,
- Sum of points is: $24.3 + 24.9 + 23.5 + 24.0 + 24.6 = 121.3$,
- The average value is: $\bar{x} = 121.3/5 = 24.26$

2. Calculation of the uncertainty (type A) related to statistical analysis:

$$u_x(type\ A) = \frac{s_x}{\sqrt{N}} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N(N-1)}} \tag{5}$$

where: s_x is a sample standard deviation.

	Accuracy
Temperature	± 1.25 C
Pressure	± 1.0 hPa
Humidity	± 3.0 %

Table 1: The accuracies of quantities measured by BME280 sensor.

Example:

- Measured temperature points: 24.3, 24.9, 23.5, 24.0, 24.6,
- The average value is: $\bar{x} = 24.26$,
- $\sum_{i=1}^{N}(x_i - \bar{x})^2 = (24.3 - 24.26)^2 + (24.9 - 24.26)^2 + (23.5 - 24.26)^2 + (24.0 - 24.26)^2 + (24.6 - 24.26)^2 = 1.17$
- Statistical uncertainty (type A) is: $u_x(type\ A) = \sqrt{\frac{1.17}{5\cdot(5-1)}} = 0.24$

3. Calculation of the uncertainty (type B) connected with other sources like resolution of the device, error of reading the value etc. This uncertainty is also called systematic uncertainty:

$$u_x(type\ B) = \frac{\Delta x}{\sqrt{3}}, \tag{6}$$

where: Δx is a boundary uncertainty e.g. accuracy of the device usually read from the sensor documentation. The accuracies of BME280 sensor are given in Table 1.

Example:

- Measured temperature points: 24.3, 24.9, 23.5, 24.0, 24.6,
- The average value is: $\bar{x} = 24.26$,
- Statistical uncertainty (type A) is: $u_x(type\ A) = 0.24$
- Systematic uncertainty (type B) is: $u_x(type\ B) = 1.25/\sqrt{3} = 0.72$,

4. Calculation of the total uncertainty:

$$u_x = \sqrt{(u_x(type\ A))^2 + (u_x(type\ B))^2}. \tag{7}$$

Example:

- Measured temperature points: 24.3, 24.9, 23.5, 24.0, 24.6,
- Statistical uncertainty (type A) is: $u_x(type\ A) = 0.24$
- Systematic uncertainty (type B) is: $u_x(type\ B) = 0.72$,
- Total uncertainty: $u_x = \sqrt{0.24^2 + 0.72^2} = 0.76$
- The measured temperature is: $T = 24.6 \pm 0.24 \pm 0.72^oC$ or $T = 24.6 \pm 0.76^oC$.

The task is to calculate average values of temperature, pressure, humidity and dust concentration and their total uncertainty values according with above scheme.

ROBOSCIENTISTS PROJECT

Motivating secondary school students towards STEM careers through robotic artefact making

Erasmus+ KA2 2018-1PL01-KA201-051129

Creators:
Part A
Nikleia Eteokleous, Raphaela Neophytou (Frederick University)
Part B
Angelika Tefelska (Warsaw University of Technology)

Declaration
This report has been prepared in the context of the ROBOSCIENTISTS project. Where other published and unpublished source materials have been used, these have been acknowledged.

Funding Disclaimer
This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.