D7.3: Pilot training and testing materials

WP7 – Pilot operation and evaluation

Document Information

Grant Agreement Number	688363	**Acronym**	hackAIR
Full Title	Collective awareness platform for outdoor air pollution		
Start Date	1st January 2016	**Duration**	36 months
Project URL	www.hackAIR.eu		
Deliverable	D 7.3 – Pilot training and testing materials		
Work Package	WP 7 – Pilot operation and evaluation		
Date of Delivery	**Contractual** 30 November 2017	**Actual**	31 December 2017
Nature	**Websites, patents filling, etc.**	**Dissemination Level**	Public
Lead Beneficiary	NILU		
Responsible Author	Sonja Grossberndt & Hai-Ying Liu		
Contributions from	Arne Fellermann (BUND), Eleftherios Spyromitros-Xioufis and Stefanos Vrochidis (CERTH), Wiebke Herding, Elroy Bos, Fabiola Benavente (ON:SUBJECT), Panagiota Syropoulou (DRAXIS)		

Document History

Version	Issue Date	Stage	Description	Contributor
1.0	14.12.2017	Draft	First version	NILU, BUND
2.0	20.12.2017	Draft	Review comments	CERTH
3.0	21.12.2017	Draft	Revised based upon the review	NILU
4.0	02.01.2018	Draft	Review comments	DRAXIS
5.0	02.01.2018	Final	Final version	NILU

Disclaimer

Copyright message

Table of Contents

1 Executive summary

This deliverable comprises an overview of the pilot training and testing materials that we will be provided to the volunteers at each pilot location. It refers to the static Arduino platforms with WiFi, Ethernet and SD card to transfer data, as well as the mobile PSOC platform and the static Wemos sensor. In addition, we have developed training materials for the hackAIR card board sensor, and describe also the hackAIR platform and mobile app, since they will be used to engage with people and report air quality (AQ) by taking pictures of the sky or provide individual perception of the surrounding air.

For the sensor platforms, this contains manuals on what pieces to buy, how to assemble them and how to take measurements. Links to the online versions of the manuals that will be updated on a regular base, complete this information.

BUND and NILU have tested the sensors with regard to user friendliness and functionality. The results are also part of this deliverable.

Last but not least, we chose to include a section about the different casings that we tested for the Arduino and Wemos platform, as well as a manual on how to build a sensor case.

2 Introduction

This deliverable provides information about the materials that will be provided to the volunteers in each pilot location. The first section describes the pilot training materials, being different sensors, the hackAIR online platform and the hackAIR smartphone app. The second part contains descriptions of the handling and functionality of the tools. A scientific evaluation will follow in D7.4 Pilot implementation report.

These materials are mainly manuals that are available on the hackAIR web pages (currently http://hackair.onsubject.eu/tutorials/), the hackAIR Github (https://github.com/hackair-project), and that will also be made available in printed version if required by the volunteers. The manuals in this deliverable are in English, but will be translated into the pilot languages if necessary. The present versions are also not to be considered final since there are still outstanding minor adjustments to be carried out in the time between the compilation of this deliverable and the start of the pilot in 2018.

3 Pilot training material

This section holds a collection of manuals for the hackAIR tools as available by December 2017. This contains a description of the different components, a shopping list, and instructions on how to build the sensor platform and how to program the sensor software.

They are available at http://hackair.onsubject.eu/tutorials/ and https://github.com/hackair-project, and currently only in English. Efforts are being made to translate them into the local language of each pilot location; however, this will not be done until the manuals will be available in their final versions. The current versions are expected to undergo minor revisions in the preparation phase to be fit-for-purpose at the beginning of the pilot activities in 2018.

3.1 Arduino manual

The Arduino sensor platform is a Do-It-Yourself (DIY) sensor platform for measuring AQ. It can either be built as a WiFi, Ethernet or SD card solution. Within the hackAIR project, data from the WiFi and Ethernet platforms will be made visible at the hackAIR platform (https://platform.hackair.eu).

3.1.1 Arduino WiFi

3.1.1.1 Major components

The Arduino WiFi hackAIR node consists of four components:

1) The hackAIR WiFi shield for wireless internet connection,
2) The Arduino Uno board for data processing and peripheral control,
3) The air quality sensor for carrying out the air quality measurements, and
4) The DHT22 temperature-humidity sensor shield.

3.1.1.2 Shopping list

Materials

- Air quality sensor: Nova PM SDS011 (one of several sensor models accepted by the hackAIR project)
- Temperature-humidity sensor DHT22
- Microcontroller: Arduino Uno R3
- WiFi shield: a printed circuit board + set of electronic components (see below)
    - 9V Power supply
    - 4 jumper cables
    - 1 WiFi module: ESP-01 ESP8266
    - 1 DIP-20 socket
    - 1 IC Bus Transceiver: 8BIT 20DIP
    - 1 voltage regulator: LD1117
    - 1 transistor: ZVN4206A MOSFET
    - 3 resistors: 1kΩ, 10kΩ, 100kΩ
    - 4 capacitors: 2x 10nF, 2x 100nF
    - 2 electrolytic capacitors: 10uF 16V
    - 3 2-pin screw terminals (3.5mm)
    - Pin header: 2.54mm male
    - Shield headers: 2.54 mm female

Tools

- Small screwdriver
- USB-to-Serial adapter: FTDI

- Soldering iron + soldering tin
- Computer to program the sensor
- Mobile phone to connect sensor platform to the WiFi

3.1.1.3 Steps to set-up the Arduino WiFi

- Assemble the WiFi-shield:
    - Start by soldering the DIP socket
        - Match the notch on the socket to the U1 mark on the circuit board
    - Then solder the ESP WiFI module
    - Then solder the resistors
        - R1: 1kΩ
        - R2: 100kΩ
        - R3: 10kΩ
    - Next step: solder the capacitors
        - C1: 10nF (ceramic)
        - C2: 100nF (ceramic)
        - C3: 10uF (electrolytic)
        - C4: 10nF (ceramic)
        - C5: 100nF (ceramic)
        - C6: 10uF (electrolytic)

For the electrolytic capacitors, ensure that the long lead goes into the hole marked with, and that the colored band matches with the white mark on the circuit board.

- To solder the voltage regulator, ensure that the metal heat ink is on the side of the white markings on the board.
- Solder the ESP module so that the antenna is placed outside the board
- Attach the temperature-humidity sensor to the board.
- Finish by soldering the screw terminals and the pin headers.
- Before using the shield, the user must flash the hackAIR firmware to the WiFi SoC. The wiki page contains information on how to use the flashing utilities (esptool, CuteESP) to upload the firmware.
- Stack the WiFi shield on top of your Arduino and connect your sensors.

For those participants who are not willing to assemble the Wi-Fi shield, they could buy fully assemble Wi-Fi shield directly from the Innovafabs community in Bulgaria

(http://www.innovafabs.eu/index.php?route=product/product&path=18_59&product_id=87).

3.1.1.4 Installing the hackAIR software

- Download the following:
    - Arduino IDE from https://www.arduino.cc/en/Main/Software
    - hackAIR library from https://github.com/hackair-project/hackAir-Arduino/releases
- Acquire an authentication token for your sensor from the hackAIR platform

- Connect your sensor to your computer using a USB cable.
- Choose a port for your Arduino. In the Arduino IDE, go to Tools → Port → choose the COM port with the highest number
- In the Arduino IDE, navigate to Sketch → Include Library → Add .ZIP library and select the hackAIR library just downloaded.
- Open the WiFiSensor sketch using Files → Examples → hackAIR → WiFiSensor
  - Make sure the correct sensor is specified in line 13: hackAIR sensor(SENSOR_SDS011);
- Add your authentication token in line 29: wifi_setToken("REPLACE WITH AUTHENTICATION TOKEN");
- Upload the programme to your sensor node using Sketch → Upload.
- Disconnect the sensor node from your computer and connect it to the power adapter.
- Connect to the sensor node using a mobile device to set up WiFi.
  - Using a mobile device, connect to the SetupGadget-XXXXXX Access Point (XXXXXX being the unique ID of the WiFi shield) and navigate to http://192.168.4.1 to set your SSID and password.

A more updated description with pictures will soon be available online: http://hackair.onsubject.eu/hackair-home/ and https://hackair-project.github.io/hackAir-Arduino/general/

A video tutorial about how to assemble the Arduino WiFi sensor platform is available here: https://www.youtube.com/watch?v=j-JxIYk2myc&feature=youtu.be

3.1.2 Arduino Ethernet

3.1.2.1 Major components

The Arduino Ethernet hackAIR node consists of three components:

1) The hackAIR Ethernet shield for wired internet connection,
2) The Arduino Uno board for data processing and peripheral control, and
3) The air quality sensor for carrying out the air quality measurements.

3.1.2.2 Steps to set-up the Arduino Ethernet

Connect the AQ sensor with Ethernet shield. Each sensor has a different pinout and a wiki page with information on how to set it up on Ethernet shield. The pin table for SDS011 and Ethernet shield and their connection are showed in Table 3.1. The connection between sensor pin and Ethernet pin is showed in Figure 3.1.

Table 3.1. Information of the sensor pin and Ethernet shield pin and their connection.

Sensor Pin	Description	Arduino/Ethernet shield Pin	Description
5V	Power supply	5V	Max 220mA
GND	Ground	GND	
Rx	Receive data	7	Transmit data
Tx	Transmit data	8	Receive data

3.1.2.3 Installing the hackAIR software

To send data, each node has an authentication token that is used to identify it and store its data. To get an authentication token, each user must create an account on the hackAIR platform website and register their nodes.

Information on how to install the hackAIR software and see the data is available here: https://hackair-project.github.io/hackAir-Arduino/library/installation/

3.1.3 Arduino SD card

At NILU, we have tested the Arduino SD card. Although this solution does not allow for transfer of real time data to the hackAIR platform website, the user can still see his/her own measurement data on the SD card.

3.1.3.1 Major components

The Arduino SD card approach consists of four components:

1) SD-card logger shield with integrated RTC (real time clock)
2) Arduino Uno board for data processing and peripheral control
3) Air quality sensor for carrying out the air quality measurements, and
4) DHT 22 temperature-humidity sensor shield

3.1.3.2 Shopping list

Materials

- Data logger shield compatible for Arduino, MicroSD-card + RTC (Assembled) with battery
- Arduino Uno R3
- Air quality sensor: Nova PM dust sensor SDS011
- DHT22, Digital Temperature and Humidity Sensor
- Jumper wires: M/M and M/F
- Power supply: USB cable or 9V adapter
- Memory SD card (e.g., 32G)

3.1.3.3 Steps to set-up the Arduino SD card

Step 1: Download the Arduino IDE

Download the latest version of the Arduino IDE from this link https://www.arduino.cc/en/Main/Software).

Step 2: Stack the data logger shield on top of Arduino UNO board and connect SDS011 sensor and DHT 22 temperature-humidity sensor

The data logger shield should be stacked on top of the Arduino UNO board while making sure all pins are inserted without bending out of the way. The shield should not touch any of the Arduino's components to avoid shorting.

It is ok to connect the sensors to the shields by using jumper wires while testing, but it is highly recommended to solder the wires afterwards (Figures 3.1 and 3.2). The wires will easily fall off and there is a risk for short circuits.

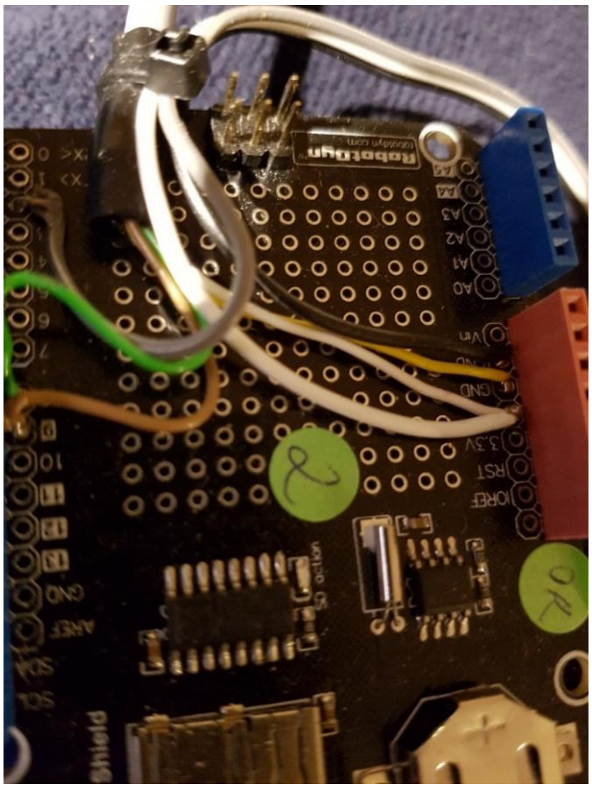


Figure 3.1. Picture of the soldered wires.

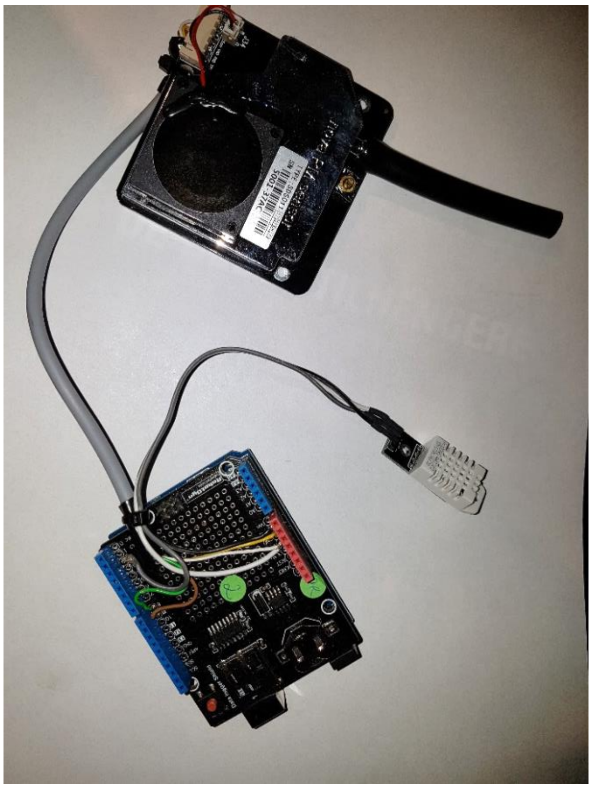


Figure 3.2. Major components.

Step 3: Format SD card

The SD-card must be formatted with the Fat32 option. It is not recommended to let the files on the SD-card grow too big. The Arduino platform is not capable of handling big files, so the storage can sometimes stop and data get lost. It is best to delete the file after copying.

Step4: Open Arduino IDE and write and upload the Arduino SD sketch

It is now time to put power to the Arduino. Then open Arduino IDE and open the Arduino SD card code in the File menu (the code can be found in Appendix A), and upload the Arduino SD card sketch from the IDE uploading button.

Step 5: Put power to the Arduino and finished

Now the shield is ready to use. At this point the SDS011 sensors with SD card approach should be ready to collect and transmit data. To see if it is actually working, open the serial monitor window in the tools menu in the Arduino IDE. The Arduino will after a few seconds print out records in the format as following:

Date	time	PM2.5	PM10	hum	temp
2017-12-13	22:54:18	6.20	17.30	27.40	22.40

3.2 Wemos manual

The Wemos sensor platform is a DIY solution like the Arduino that can be used to measure AQ.

3.2.1 Major components

The Wemos D1 mini sensor configuration consists of three main components:

1) The Wemos D1 mini board for data processing and peripheral control,
2) The air quality sensor for carrying out the air quality measurements, and
3) Humidity sensor DHT22

3.2.2 Shopping list

Materials

- Wemos D1 mini, soldered with 2 1x8 long sleeve headers 2.54mm
- Air quality sensor: Nova PM SDS011 (one of several sensor models accepted by the hackAIR project)
- Humidity sensor DHT22
- USB cable
- Power to USB plug
- 7 jumper cables (4 wires m/m (short length ok) 2 wires m/m (longer length) 1 wire m/f (longer length)
- Short tube for air inlet

Tools

- Scissors
- Office tape
- Computer to program sensor

3.2.3 Steps to set-up the Wemos

First the Wemos D1 mini has to be setup correctly. This means that it needs to have two 1x8 long sleeve headers soldered to both pin sides.

After this is done, first the SDS011 sensor needs to be connected to the female side of the jumper wires (Table 3.2).

Table 3.2. The connection between sensor pin and Wemos pin.

Sensor pin (counting from left side)	Connect to Wemos pin
5v	5V
GND	G
RXD	D6
TXD	D7

Now it is time to connect the humidity sensor (Figure 3.3). First, connect the appropriate jumper cables to the pins. Connect the two male/female cables to VCC and DATA. Then connect the female/female cable to the GND pin. Then secure the cables to the sensor by taping the sensor to the pins with office tape and tape around the pins to make the connection more durable.

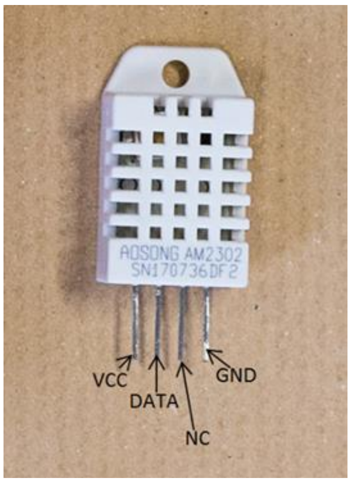


Figure 3.3. Humidity sensor DHT22.

Then you can connect the cables of the humidity sensor to the Wemos D1 mini (Table 3.3).

Table 3.3. The connection between humidity sensor pin and Wemos pin.

Humidity sensor pin (counting from left side)	Connect to Wemos Pin
VCC	3V3
DATA	D4
NC (don't connect)	-
GND	G (male side)

Be advised that the Wemos D1 only has one GND or G connection. To give a G connection to both sensors, for the second sensor DHT022, the GND of the humidity sensor is connected to the back side of the Wemos on the male connector.

3.3 PSOC manual

This sensor is a portable one, based on a PSoC (Programmable System-on-Chip).

3.3.1 Major components

This sensor is powered with a mobile power bank and uploads its measurements through Bluetooth on your phone. You can move it around and measure in multiple locations.

3.3.2 Shopping list

Materials

- Air quality sensor: Nova PM SDS011
- Microcontroller: PSOC BLE CY5671
- Micro-USB breakout board
- Jumper wires: M/M and M/F
- Pin header: 5-pin 2.54 male
- USB power bank

Tools

- Programming tool: CY8CKIT-059

- Soldering iron and soldering tin
- Windows computer

3.3.3 Steps to set-up the PSOC

- After receiving your PSoC 5LP Prototyping Kit proceed to detach the programmer module by breaking at the indicated area
- Solder a 5-pin header facing upward on the programmer board and connect to the Bluetooth module
- Solder the micro USB module
- Connect the sensor to the programmed BLE module

3.3.4 Installing the hackAIR software

- Download the following:
  - PSoC Programmer Software from Cypress (https://www.cypress.com/user/login?destination=file/365491)
  - .hex file for your sensor from https://github.com/hackair-project/hackAIR-PSoC.
- Connect the programming tool to the PSOC and a computer.
- Launch PSOC Programmer software: Open the .hex file. Select the device and click 'Program'.
- Disconnect and power the device using the power bank.

A more updated description with pictures will be available online: http://hackair.onsubject.eu/hackair-mobile/

A video tutorial is available here: https://www.youtube.com/watch?v=kKMY8_PN4P4&feature=youtu.be

3.4 Cardboard sensor manual

The hackAIR cardboard sensor is an easy way to measure particles in the air. It consists of an empty milk box covered with petroleum jelly exposed for 24 hours on PMs. After collecting the milk box and using a smartphone macro-lens a picture is taken as close as possible. The smartphone runs the computer vision algorithms on the captured photo and send the results to the central hackAIR system.

3.4.1 Major components

The sensor consists of an empty milk box covered with petroleum jelly exposed for 24 hours on PMs. After collecting the milk box and using a smartphone macro-lens a picture is taken as close as possible. When uploaded to hackAIR, the hackAIR mobile app runs computer vision algorithms on the captured photo and sends the results to the central hackAIR system.

3.4.2 Shopping list

- An empty carton of milk
- One small jar of petroleum jelly
- Commonly used thread
- Mobile macro lens of x6 or greater magnification. Make sure they are super macro and not just macro.
- Counter weight (batteries, paper clippers, coins etc.)
- Butter knife

3.4.3 Steps to set-up the cardboard sensor

1) Get an empty carton of milk, clean it and let it dry.
2) Cut square pieces with dimensions 5cm x 5cm.
3) Along the diagonal axis of your choice draw two small dots with the tip of a pencil, near the centre of the test surface.
4) Bend the surface slightly outward, along the same diagonal axis you chose earlier. This will result to a constant capture of air particles, on the surface.
5) Pierce the upper right corner and put a thread through the whole.
6) At the lower left corner, adjust a counter weight (batteries, paper clippers, coins, etc).
7) Apply a small amount of petroleum jelly on the test surface using the butter knife. Before applying the petroleum jelly, make sure the test surface is clean and completely dry.

How to capture your image

After 24 hours of constant exposure, retrieve the sensor. When you exceed the 24 hour exposure the cardboard sensor is useless and you need to replace it simply because it is not predicted by the algorithm or the pilot measurements which are currently contacted by the writer. Also, exposing the test surface for less than 24h will result in a lower amount of accumulated air particles in the vaseline layer.

Place the aforementioned macro lens on top of the two small dots, you previously drew. Let your mobile adjust the focus of its camera and capture your image. Use flash if necessary (if the picture is dark, due to poor lighting inside the room using flash is recommended).

Depending on your lens model the necessary distance between the lens and the test surface may differ. Make sure you read the manual of your lens model first.

When you buy lens, make sure they are super macro and not just macro. Macro lenses do not offer the necessary zoom for capturing the desired picture. You should aim for the two dots you made with your pencil during the preparation of the sensor. As mentioned above, the necessary distance between the lens and the test surface may differ depending on the lens you use.

A more updated description with pictures will be available online: http://hackair.onsubject.eu/hackair-cardboard/

3.5 hackAIR platform

The main objective of the hackAIR platform is to combine official AQ data with community-driven data sources for collecting, analysing and sharing air quality measurements to community members through low-cost open hardware sensors easily assembled by citizens, web and/or mobile phones. Its URL is https://platform.hackair.eu/.

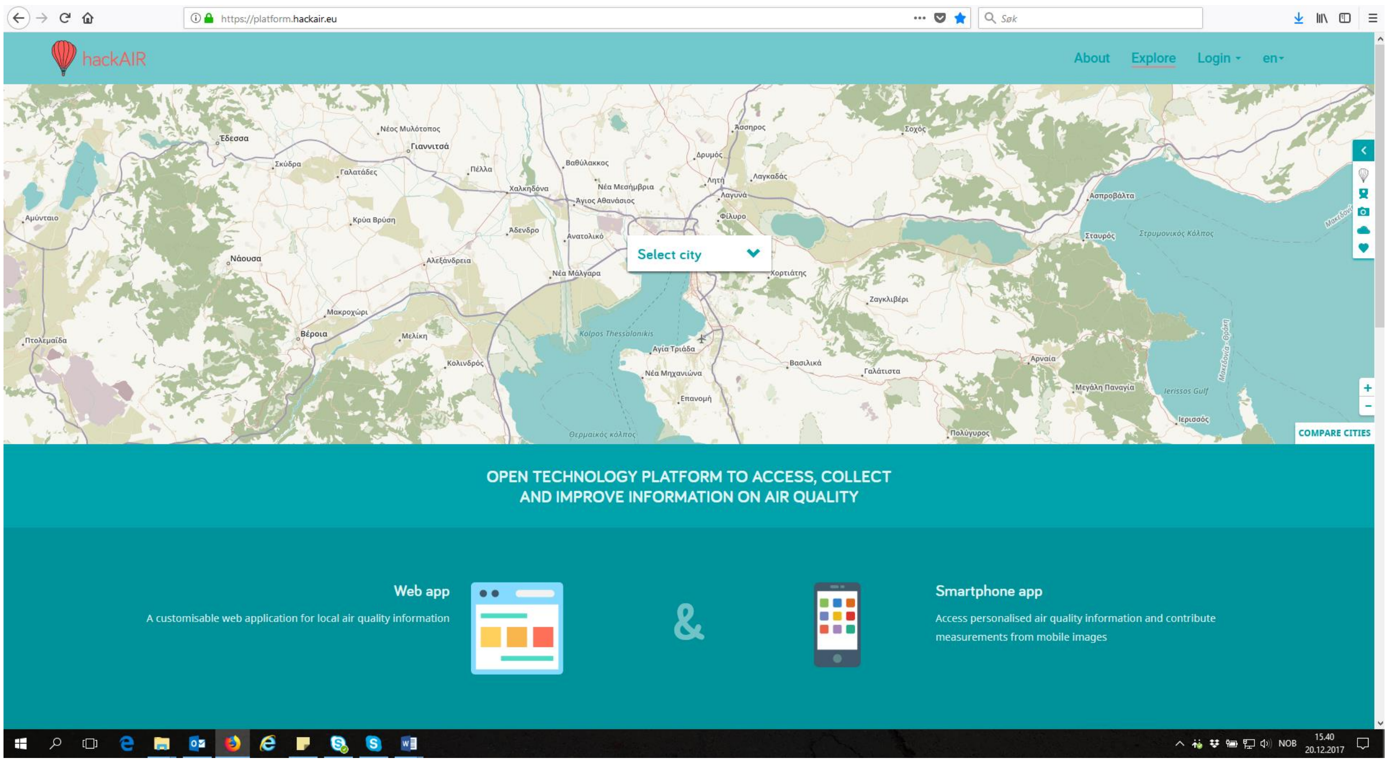
The current platform version is still in a testing phase, which includes fixing bugs, ensuring privacy and testing for user friendliness within the project consortium. Thus, minor changes in function and design will be made during the next weeks before the official launch in February 2018.

This section provides only a short update of what is available now in December 2017. For information about architecture and further functionalities, please read D5.2.

Start page/ Explore

The start page of the hackAIR platform offers access to the different pilot locations through a map: Athens, Berlin, Brussels, Oslo, and Thessaloniki. A little window on the right-hand side allows for choosing a map filter. Options are: Fusion map, sensors, photos, open data, perceptions. Once the user has chosen a pilot location, the respective city appears on the map and on the left-hand side another window shows the current AQI of the location. At the moment, this feature is only available for Oslo and Berlin. The map shows also the different sensors and their measuring results.

Further below on the start page, an overview of the different ways to participate is shown. Also, information about and links to the download pages of the mobile app is available here. Further links are: Download the app, Terms and conditions, Support, and Tutorials.



hackAIR
About
Explore
Login
en
Select city
COMPARE CITIES
OPEN TECHNOLOGY PLATFORM TO ACCESS, COLLECT AND IMPROVE INFORMATION ON AIR QUALITY
Web app
A customisable web application for local air quality information
&
Smartphone app
Access personalised air quality information and contribute measurements from mobile images

Figure 3.4. hackAIR platform Start page/ Explore.

About

A click on this link will lead to the official hackAIR project web page. Here, all relevant information about the project, partners, news and events can be found. It is currently located at http://hackair.onsubject.eu/ due to some changes in design and functionality. After the launch of the pilots in February 2018, the URL will be http://www.hackair.eu.

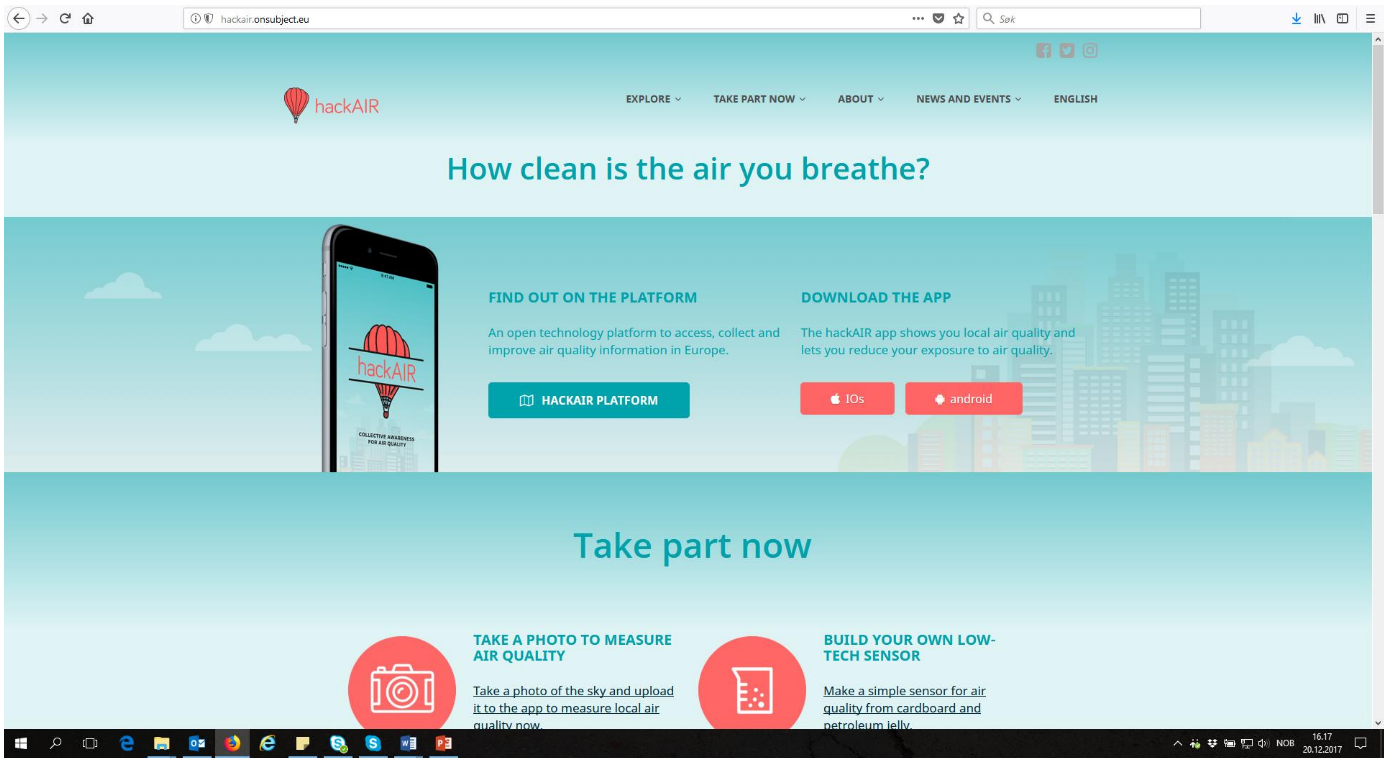

hackair.onsubject.eu
hackAIR
EXPLORE
TAKE PART NOW
ABOUT
NEWS AND EVENTS
ENGLISH
How clean is the air you breathe?
FIND OUT ON THE PLATFORM
An open technology platform to access, collect and improve air quality information in Europe.
HACKAIR PLATFORM
DOWNLOAD THE APP
The hackAIR app shows you local air quality and lets you reduce your exposure to air quality.
IOs
android
Take part now
TAKE A PHOTO TO MEASURE AIR QUALITY
Take a photo of the sky and upload it to the app to measure local air quality now.
BUILD YOUR OWN LOW-TECH SENSOR
Make a simple sensor for air quality from cardboard and petroleum jelly.

Figure 3.5. hackAIR platform About.

Login

Clicking at the Login tab, the user will get the option to register to be able to participate with own AQ measurements to the platform. Registration is also possible through social media. Once the user has registered, this tab will be used to login to the platform.

After successful registration, the user will see new tabs in the upper right corner of the screen.

Dashboard: This is the "start page" after being logged in. The user can switch to any other pilot location. A little window on the right hand side allows for choosing a map filter. Options are: Fusion map, sensors, photos, open data, perceptions. Once the user has chosen a pilot location, the respective city appears on the map and on the left hand side another window shows the current AQI of the location, in addition to "tips of the day" with recommendations in relation to AQ. The map shows also the different sensors and their measuring results. A time series of AQ of the selected pilot is also available below the map.

Social: This section will be the gateway to other platform users. Here, the user can see notifications, news feeds etc that are customized according to his/her settings. These functions are, however, not available yet.

Username: The profile pages are individual and display the user's latest activities, photos, perceptions, sensors and followers. Here it will be possible to logout.

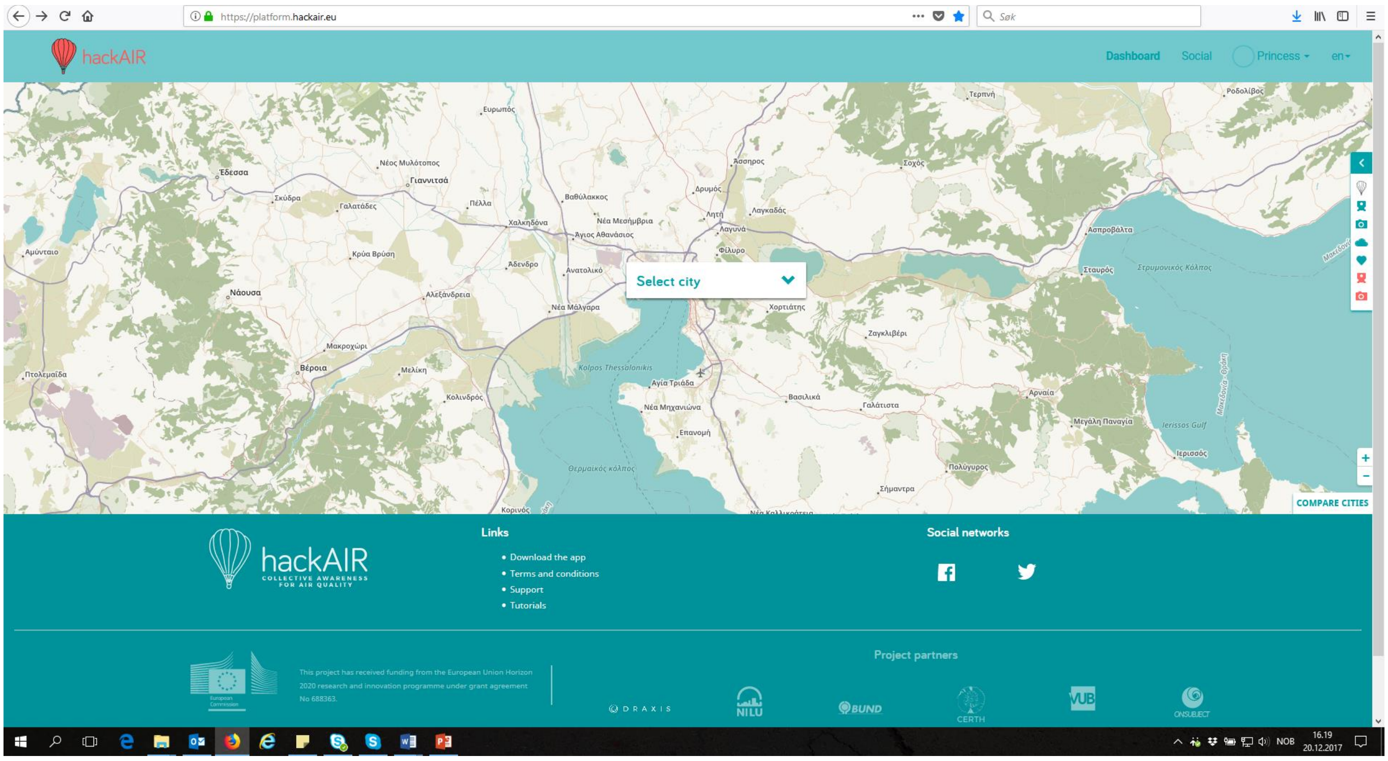

https://platform.hackair.eu
hackAIR
Dashboard
Social
Princess
en
Select city
COMPARE CITIES
Links
Download the app
Terms and conditions
Support
Tutorials
Social networks
Project partners
This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 688363.

Figure 3.6. hackAIR platform Login.

Language

The user can currently choose between three languages: English, German and Norwegian. Information may also be available in Greek, Flemish and French at some later point.

3.6 hackAIR mobile app

The hackAIR smartphone app has two main functionalities. It provides access to the hackAIR platform, but it serves also as sensor – for taking pictures of the sky, taking pictures of the hackAIR cardboard sensor and providing information about the users' perception of the surrounding AQ. The app is available both for Android and iOS.

When using the app for the first time, the user has to log in with his/her login credentials from the hackAIR platform. The hackAIR smartphone app has different sections:

Home

The first page shows the start page. It also provides the tip of the day, a recommendation of how to improve AQ.

Take a photo

This section allows the user to take (1) a picture of the sky or (2) a picture of the cardboard sensor and upload it to the platform. GPS has to be enabled.

Perception of air quality

Here the user can indicate how he/she perceives the AQ right where they are (very good, good, medium, bad). This information will also be made visible on the platform. GPS has to be enabled.

Map

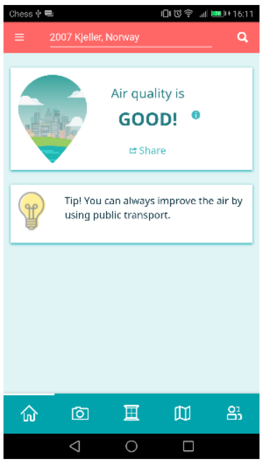
The map has different layers that can be enabled as required. It can display the different hackAIR sensor types, their measurements values and additional AQ information (open sources, pictures of the sky, etc). The map shows also data fusion maps of the AQ of the participating countries.

My profile

In this section the user can change the personal settings, see their achievements, followers, sensors, or communities.

The current mobile app version is still in a testing phase, which includes fixing bugs, ensuring privacy and testing for user friendliness. It will be available for both iOS and Android through i-/google play store.

This section provides only a short update of what is available now in December 2017. For information about architecture and further functionalities, please read D5.2.



Tip! You can always improve the air by using public transport.

Figure 3.7. hackAIR smartphone app Home.

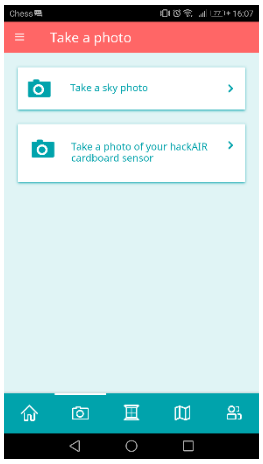
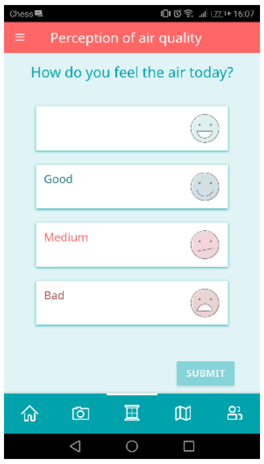


Figure. 3.8. hackAIR smartphone app Take a photo.



How do you feel the air today?

Figure. 3.9. hackAIR smartphone app Perception of air quality.

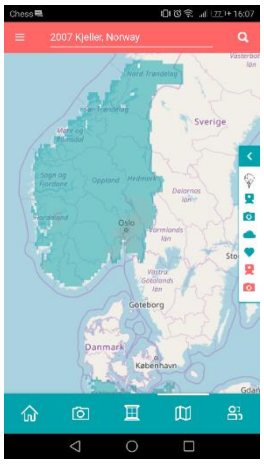
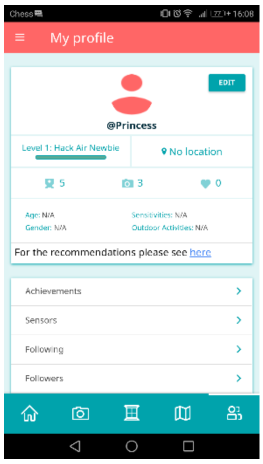


Figure 3.10. hackAIR smartphone app Map.

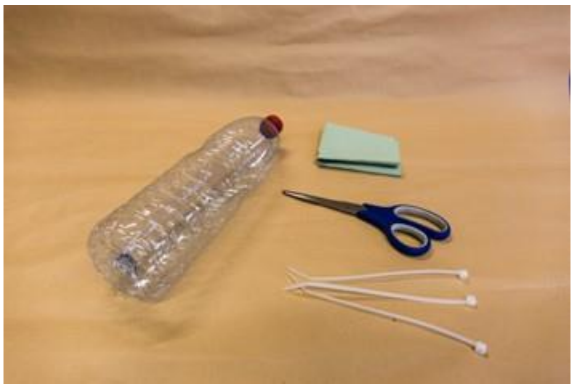


For the recommendations please see here

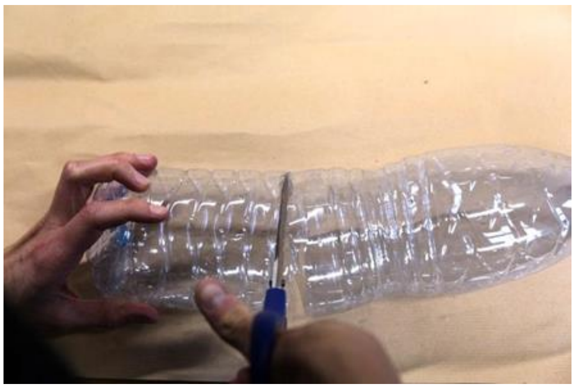
Figure 3.11. hackAIR smartphone app My profile.

3.7 Sensor casing

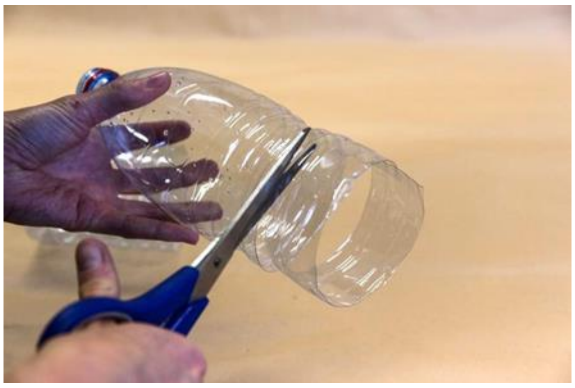
This overview shows the steps for the suggested casing instruction of PET upcycling solution for the sensor as suggested by the submission date of this report. In this example, we have been using a Wemos sensor platform, but the suggested casing is also suitable for an Arduino sensor platform.

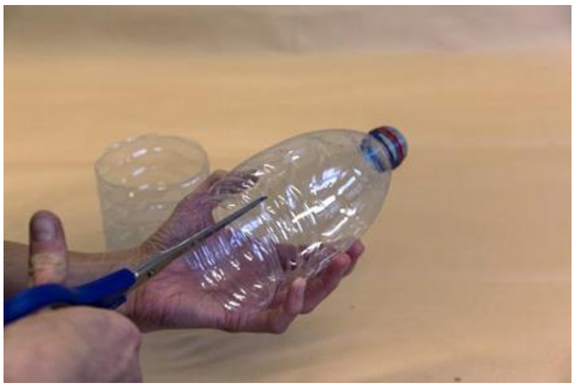


Material needed: Empty PET-bottle, scissors, cable ties, cloth or paper towel to dry bottle



Cut PET-bottle into two pieces.

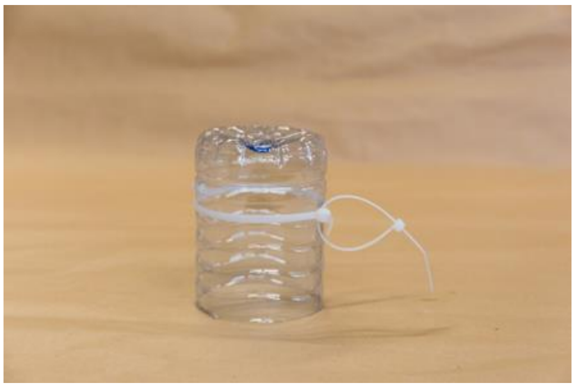




If longer the middle part can be removed, in particular if it is not as thick as the rest of the bottle.

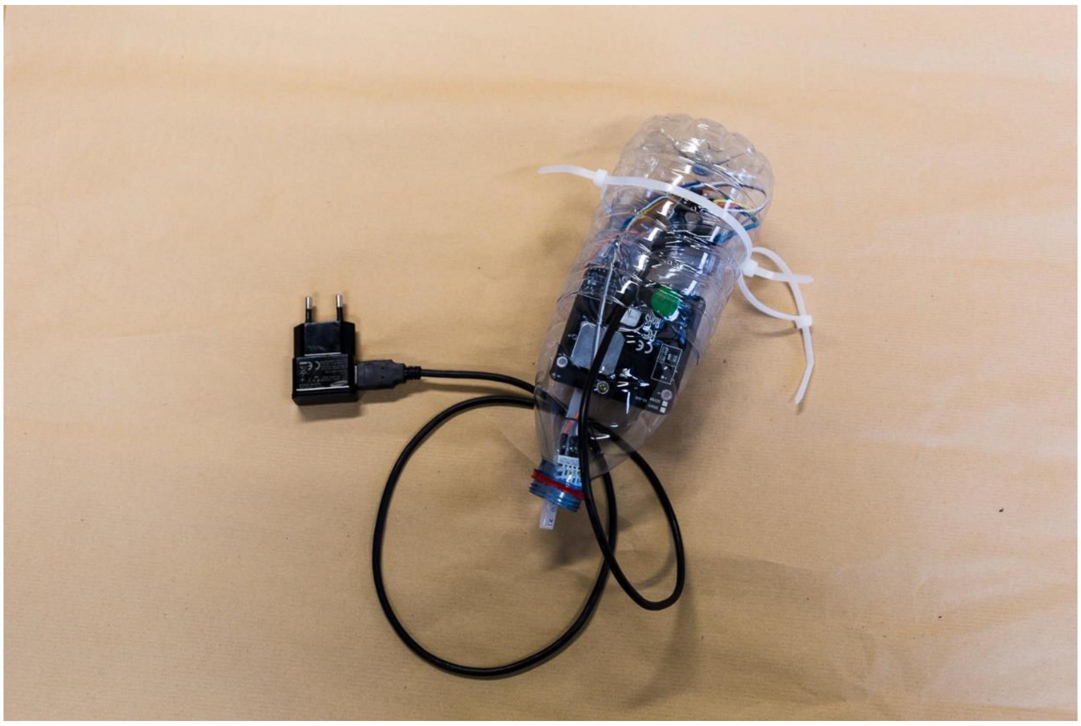
Dry bottle.

Cut little cuts into the part of the bottle with the opening. This will later help pushing the two parts into each other



Attach cable ties to the part without the hole to create a kind of hook to use to attach the sensor later.

Once finalized, the casing is able to house the sensor, the Wemos D1 and the humidity sensor. It has one opening for the sensors to be exposed to outside air and it has different openings for the USB cable and possibilities to let in further air to ventilate the electronics better. This picture shows the first functioning prototype:



4 Testing of the hackAIR sensor technology to be used in the pilots

To guarantee a smooth functioning of the sensor technology in the pilots, NILU and BUND have been testing the tools during the last months internally. This chapter describes both the testing and the results.

4.1 Arduino

4.1.1 Arduino WiFi

The Arduino WiFi has been tested by BUND and NILU. Both partners have ordered the components according to the shopping list and assembled the sensor platforms. It turned out that the shopping list had been modified several times by sensor platform developers, and thus, the testing has been postponed by several weeks. Ordering the components according to the shopping list required also soldering work. Since none of the testing partners at BUND or NILU had any previous experience with this activity, it took a while to get hold of the required equipment and to gain competence in this activity. After assembling the sensor platform, the next step was to flash it with Arduino's software. For those of us who never did this before, it took a while to figure out the procedure. After successful application, it turned out that the WiFi connection has been very challenging to establish and maintain.

4.1.2 Arduino Ethernet

NILU has tested the Arduino Ethernet sensor platform. However, test results are not available at this point.

4.1.3 Arduino SD card

We tested Arduino platform with SD card together with reference equipment in the lab and the field, respectively.

Laboratory test results

The lab testing took place in the laboratory of building P35 of HiOA (Oslo and Akershus University College of Applied Sciences), for one hour from 9:30 AM to 10:30 AM, on the 21th of March, 2017. The laboratory is a clean room, with a constant room temperature of 20°C. The LAS-AIR II was used as a reference device.

The graphs below show the comparison of the LAS-AIR II and the SDS011 sensor values and the correlation between them. The PM2.5 values have a better correlation compared to the PM10 values, although it is still quite low with a value of R^2: 0.0189. However, this is to be expected because the AQ conditions in the room are almost stable within this very small time frame. The LAS-AIR II sensor appears to be more sensitive to small PM2.5 changes than SDS011. Since the hackAIR sensor platforms do not aim at providing precise measurements, but rather trends, the results show that they are working as expected.

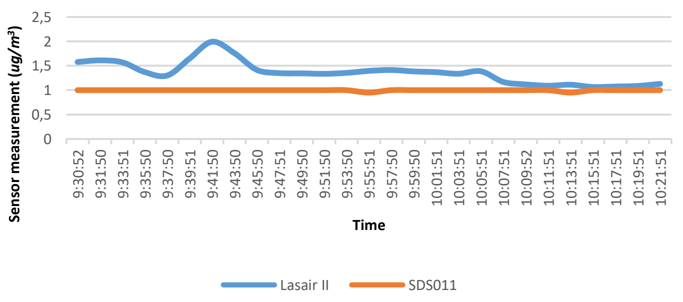


Figure 4.1. Comparison of PM2.5 measurement between SDS011 and Lasair II in the lab.

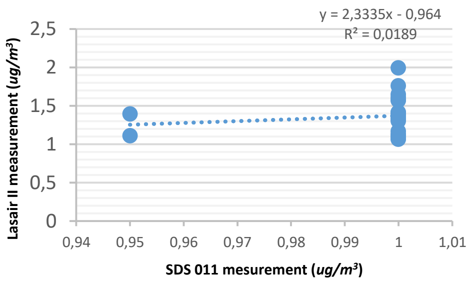
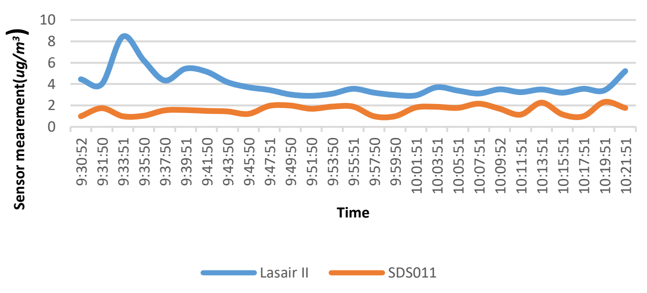


Figure 4.2. Correlation of PM2.5 measurement between the SDS011 and Lasair II in the lab.



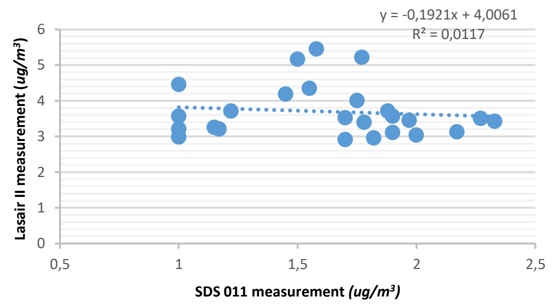


Figure 4.3. Comparison of PM10 measurement between SDS011 and Lasair II in the lab.

Figure 4.4. Correlation of PM10 measurement between the SDS011 and Lasair II in the lab.

Co-location at reference station

We co-located three SDS011 sensors with Arduino SD card approach at one air quality monitoring station, i.e., Kirkevien station, from 24 to 30, November 2017. Kirkevien station is located at urban area with dense traffic, along Ring 2 in Oslo (with coordinate 59.932305° N, 10.724519° E). The Figures and Table 4.1 below show the comparison of the AQ monitoring station and the SDS011 sensor values and the correlation between them. For both PM2.5 and PM10, the correlation coefficient between three SDS011 sensors and reference station are quite good (Table 1). Two of the SDS011 (i.e., SDS011-1 and SDS011-3) sensors showed very high correlation coefficient with the reference station data ($R^2 \geq 0.8$).

Table 4.1. Correlation coefficient between three SDS011 sensors and reference station.

R^2	PM2.5	PM10
SDS011-1	0.80	0.80
SDS011-2	0.78	0.74
SDS011-3	0.81	0.82

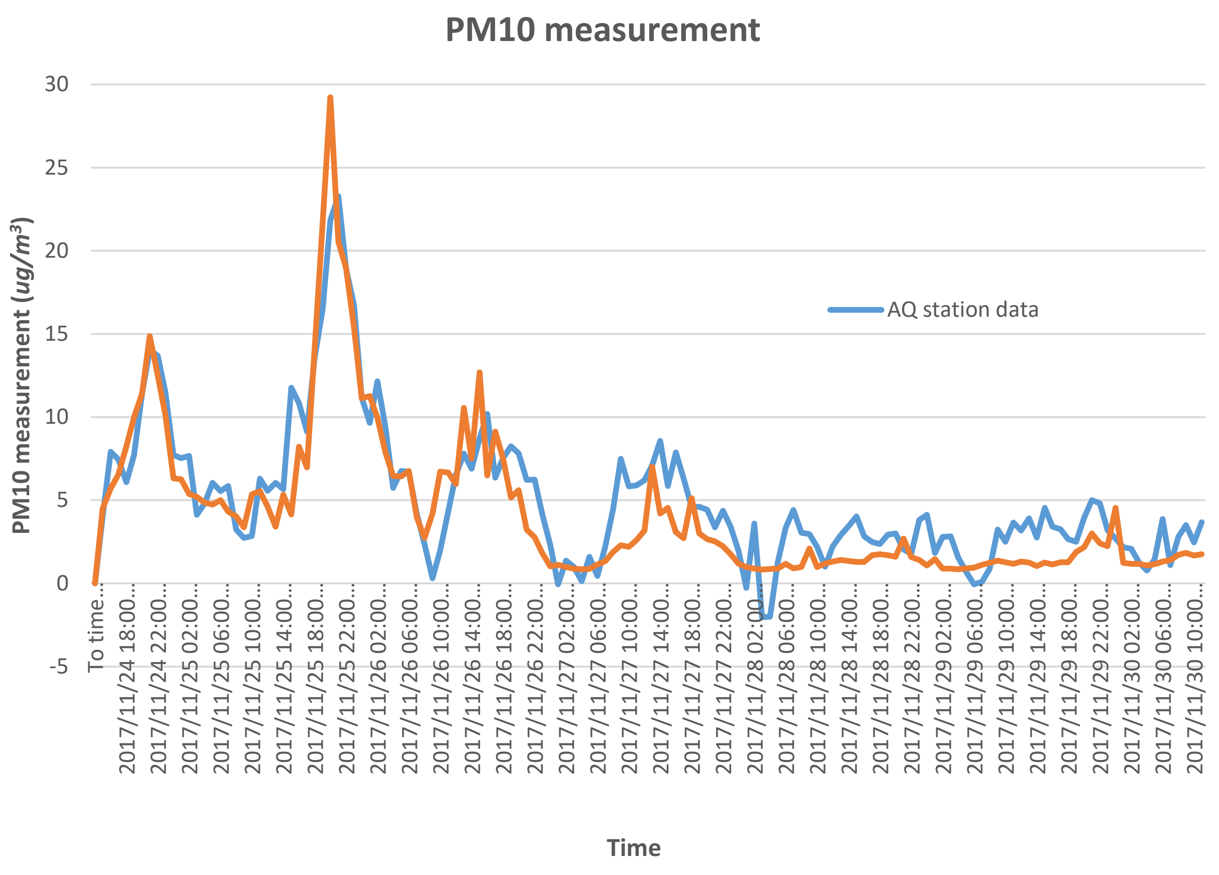


Figure 4.5. Comparison of PM10 measurement between SDS011-1 and air quality monitoring station in the field.

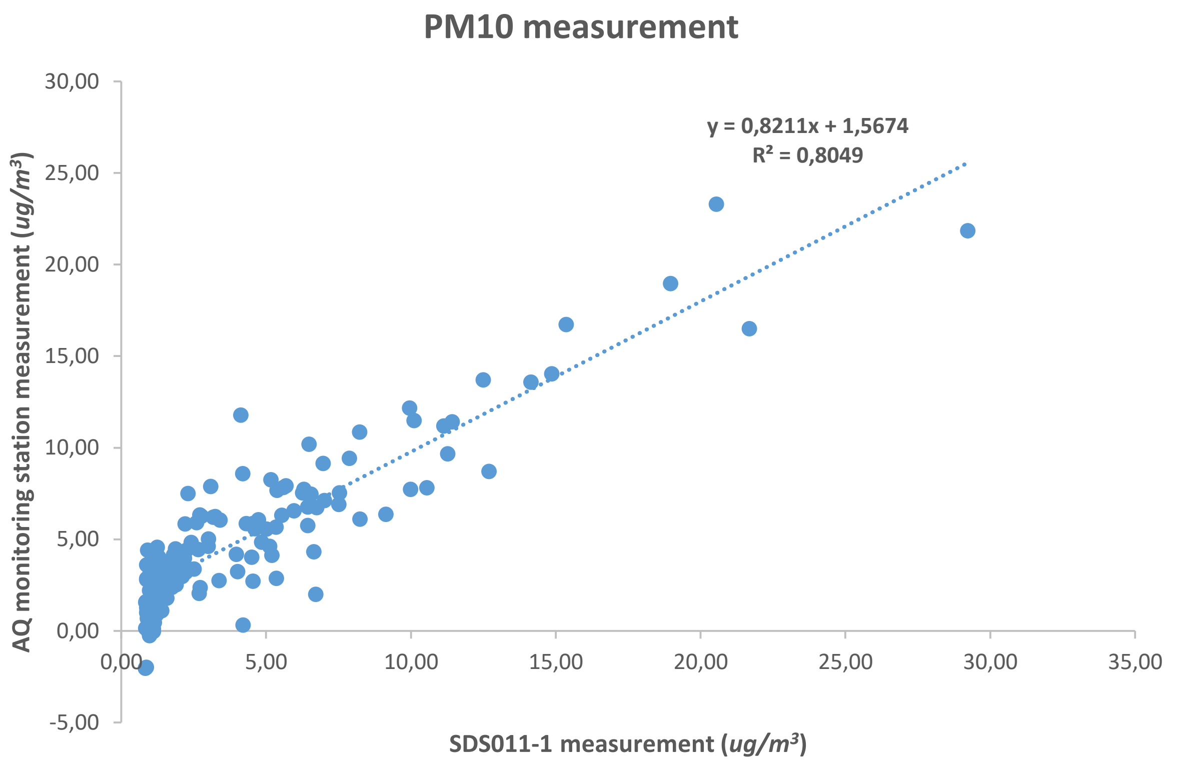


Figure 4.6. Correlation of PM10 measurement between the SDS011-1 and air quality monitoring station in the field.

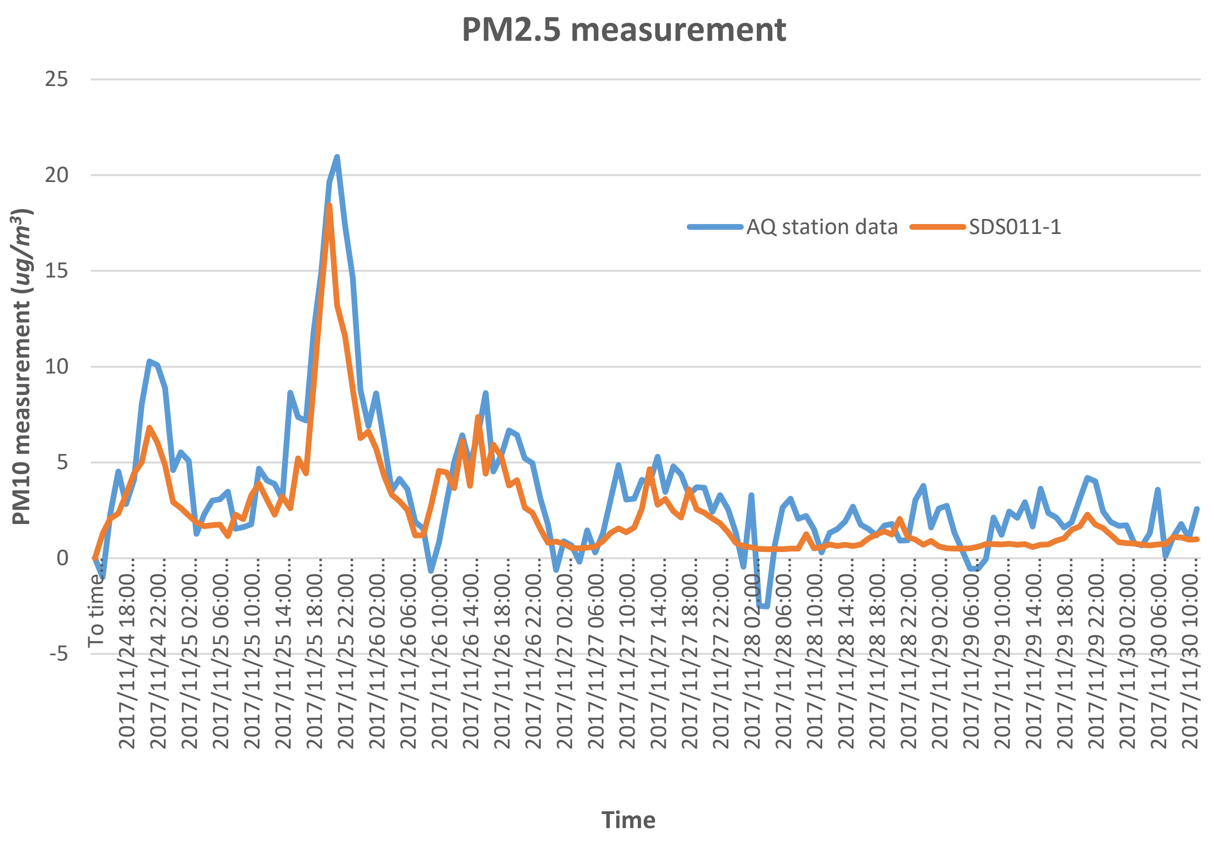


Figure 4.7. Comparison of PM2.5 measurement between SDS011-1 and air quality monitoring station in the field.

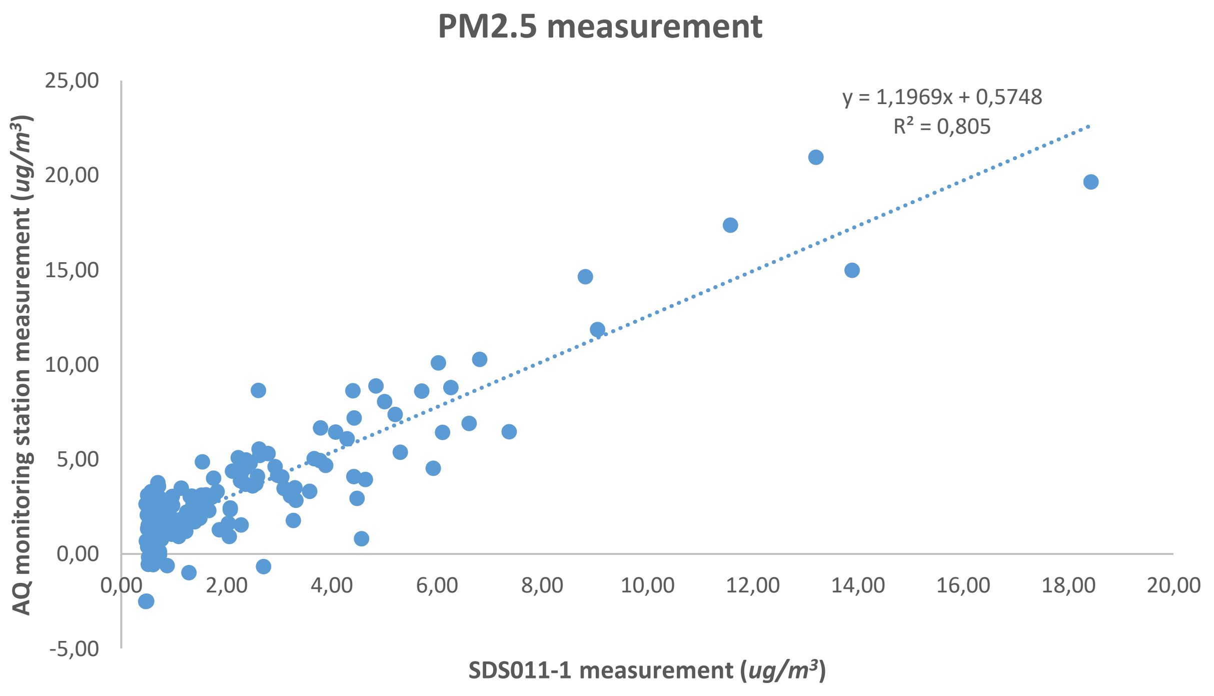


Figure 4.8. Correlation of PM2.5 measurement between the SDS011-1 and air quality monitoring station in the field.

Further, we compared the PM2.5 and PM10 measurement values among the three SDS011 sensors (Figures 4.9-11), and the results showed that there is very high correlation (R2⩾ 0.92) for both PM10 and PM2.5 among three SDS011 sensors. It seems that the SDS011 sensors have a better correlation on PM2.5 than PM10 among themselves (Table 4.2).

Table 4.2. Correlation coefficient for PM2.5 measurement among three SDS011 sensors.

R^2	SDS011-1	SDS011-2	SDS011-3
SDS011-1	1	0.97	0.99
SDS011-2	0.97	1	0.99
SDS011-3	0.99	0.99	1

Table 4.3. Correlation coefficient for PM10 measurement among three SDS011 sensors.

R^2	SDS011-1	SDS011-2	SDS011-3
SDS011-1	1	0.92	0.98
SDS011-2	0.92	1	0.96
SDS011-3	0.98	0.96	1

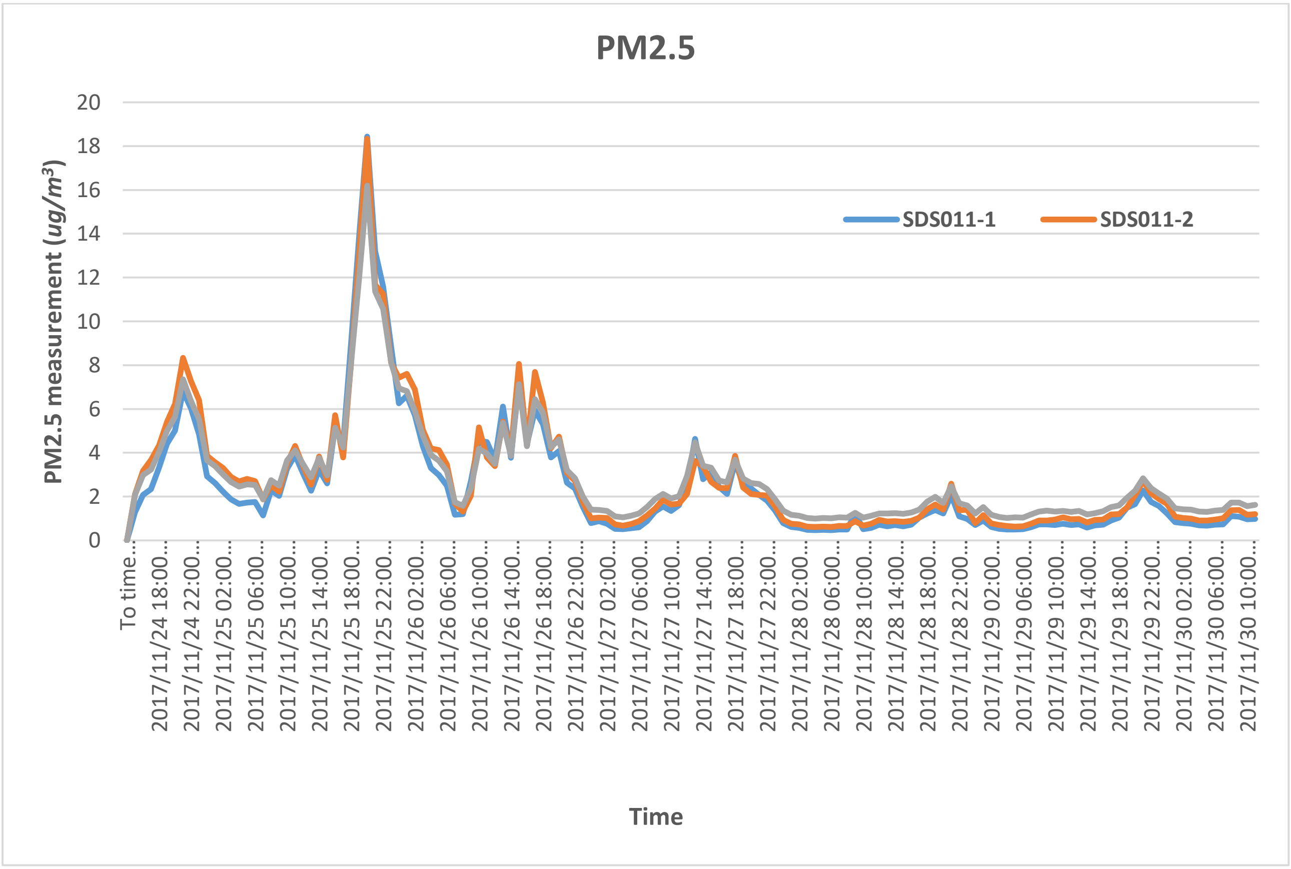


Figure 4.9. Comparison of PM2.5 measurement among three SDS011 sensor in the field.

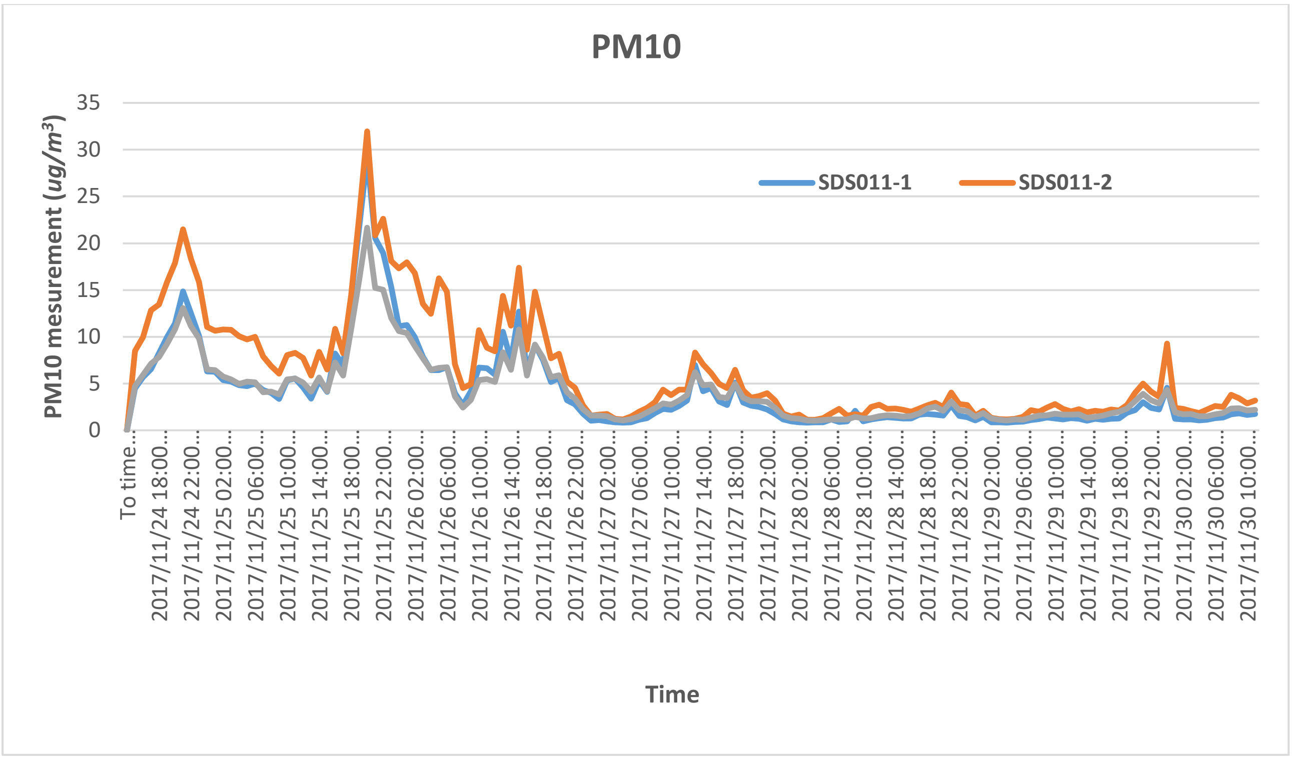


Figure 4.10. Comparison of PM10 measurement among three SDS011 sensor in the field.

Lessons learned

The SD logger shield for the Arduino was relatively cheap and easy to use. There were only a few modifications that needed to be made to the code developed for the Arduino WiFi shield and the logger chip does not require all of the extra components that the Arduino WiFi shield needed. The code for the logger shield can be found in Appendix A.

During one-week testing period in the field, SDS011 sensors showed very satisfactory results for both PM2.5 and PM10 measurement. From both field and lab testing, we can see that the SDS011 sensors are not so sensitive with PM concentration at low levels.

4.2 Wemos

The Wemos D1 Mini as platform to connect the PM sensors to the hackAIR platform has been tested by the pilot partner BUND. After the firmware for the Arduino platform was finished the firmware for the Wemos D1 mini was developed until December 2017. Already in November BUND received the first test sketches of the firmware, which already performed reliably. The official firmware was uploaded by the end of November.

BUND has completed a first test of the Wemos code in the first week of December. The objective of the test was to ensure that both the firmware and the Wemos D1 mini hardware reliably connect the sensors to the hackAIR platform.

The hardware test followed these steps:

1) Assembled 8 Wemos D1 mini, soldering the long 1x8 header pins to the board
2) Uploaded the Wemos firmware from the hackAIR library to the Wemos D1 mini, with dedicated token codes from the hackAIR library.
3) Connected both the PM sensor SDS011 and the humidity sensor DHT022 to the board with jumper wires
4) Powered the Wemos via USB
5) Check for Wi-Fi created by Wemos & if available connect to Wi-Fi
6) Log Wemos into Wi-Fi and disconnect from Wemos
7) Register activity of Wemos on the hackAIR platform

BUND found that, of 8 Wemos boards tested, 2 did not perform as expected, not creating a WiFi after being programmed with the firmware. BUND reckons this is either due to some programming mistake by them or to some problems with the specific Wemos boards. Statistically 25% of the Wemos boards did not work straight out of the box. Even though this is a high number, BUND still overall was satisfied by the easiness and reliability of the Wemos. Most of the boards logged into Wi-Fi systems with ease and consistently connected to the website.

It has to be added that the connection to the humidity sensor could not be tested as the website does not indicate humidity values. They are used in the background algorithms but not displayed. So BUND did not have any indication of the performance of the humidity sensor. The PM sensor however produced PM data and submitted it to the hackAIR platform, and there is no indication why the humidity sensor shouldn't work the same way.

4.3 PSOC

NILU tested the PSOC sensor platform and further tests will be carried out in the future. However, test results are not available at this point.

4.4 Card board sensor

The cardboard sensor is a hands-on exercise that is easy to set-up. The sensor has been tested by TEI by comparing its measurements (blobs detected) with measurements of a DYLOS laser particle counter. The results show that correlation between the hackAIR card board sensor and the DYLOS is quite promising. Further testing will be carried out in the near future.

Table 4.4 Correlation test series of hackAIR card board sensor and DYLOS.

Date/Time		hackAIR card board sensor (Blobs detected)	DYLOS (μg/m3)		Dylos Pure
			No Humidity	Humidity	
08.11.2017	Day 1	232,6	40,5946	105,14	95
09.11.2017	Day 2	209,7	39,74	93,1902	93
10.11.2017	Day 3	279,2	53,4139	119,6472	125
21.11.2017	Day 4	132	34,1849	95,7178	80
23.11.2017	Day 6	300,7	69,2245	224,6334	162
24.11.2017	Day 7	326,1	77,7707	195,9821	182

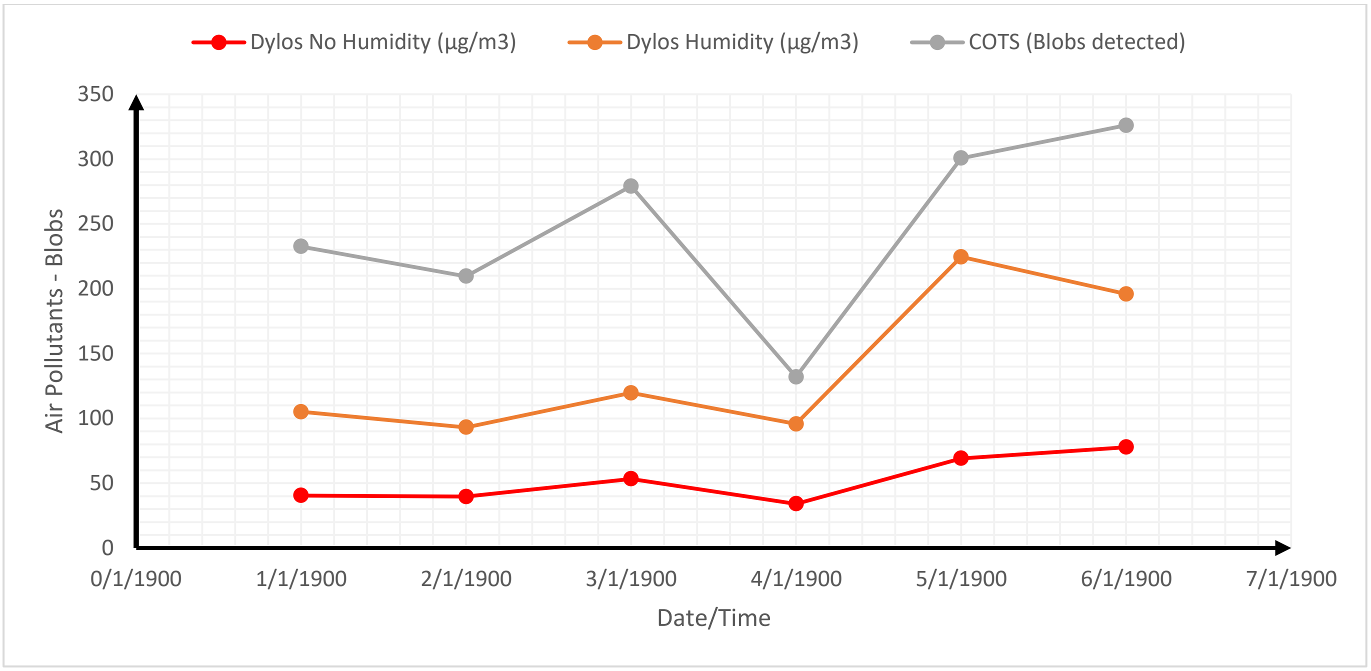


Figure 4.11 Correlation test series of hackAIR card board sensor (COTS) and DYLOS

4.5 hackAIR platform

The hackAIR platform has been tested by both NILU and BUND. It is currently updated every Friday by DRAXIS. In the time between each update, the hackAIR consortium is asked to test the functionalities and to report bugs and questions at the hackAIR Jira. The functionalities that are currently available seem to work well.

Our testing results and the current development are promising.

hackAIR partner VUB is currently also testing the hackAIR platform with regard to privacy. The results will be compiled in Deliverable D7.4-Intermediate pilot implementation and evaluation report.

4.6 hackAIR mobile app

The hackAIR mobile app is currently available for Android systems only. It has been updated regularly with the request to the hackAIR consortium to test all available functionalities and to report bugs and questions at the hackAIR Jira (jira.draxis.gr/secure/Dashboard.jspa).

hackAIR partner VUB is currently also testing the hackAIR mobile app with regard to privacy. The results will be compiled in Deliverable D7.4-Intermediate pilot implementation and evaluation report.

4.7 Sensor casing

4.7.1 Initial requirements

In preparation of the pilots, the hackAIR project has first undertaken individual research on requirements of sensor casing. The technical partners made the following list of requirements for the casing.

Arduino/Wi-Fiand Ethernet as well as Wemos

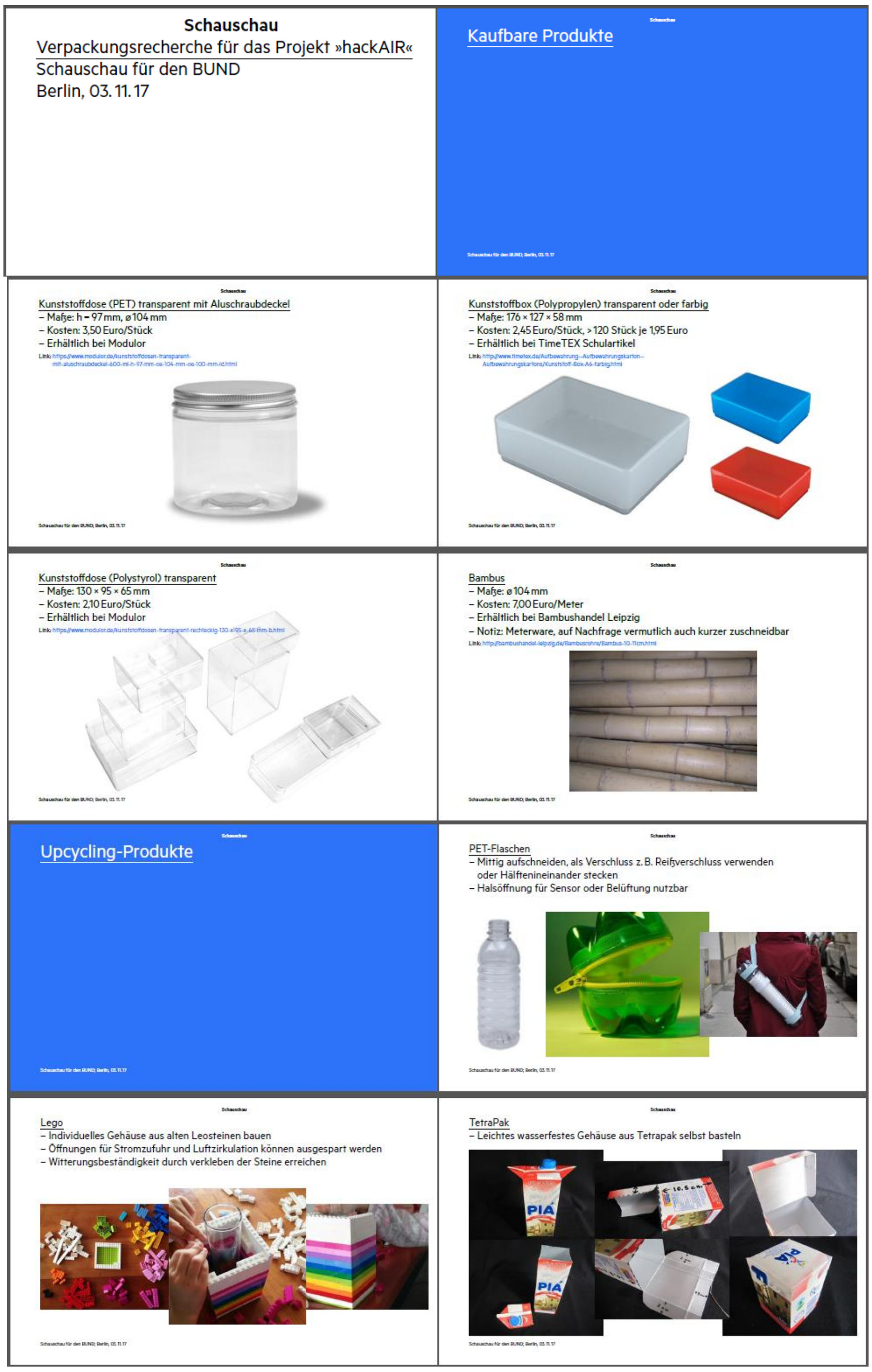
- The used materials color should be bright (white or any other) in order to avoid temperature concentration. You may use plastic for best temperature repel. Any user must be able to open the case and access equipment. Avoid metals due to the WiFi existence.
- Any opening / holes should be properly protected from rain drops.
- A place for holding the case at the balcony should be designed.
- Power and Ethernet cords positions should be carefully designed at the lower part of the case.
- A place for solar cell power bank may be designed at the top of the case.
- Any electronic equipment (arduino / ESP / Ethernet module) should be placed at the higher place in the case and a hole must be designed above them in order to guide hot air to the environment and not capture it in the case.
- The minimum size for placing the electronic equipment is (height: 6cm, cross section: 10cmx10cm).
- The sensor should be placed at least 6-7cm below the electronics.
- Free air paths should be ensured for both the inlet and outlet of the sensor.
- Air inlet of the sensor should be guided through a chamber that a user could fill with any dehydrating material (salt – rice – silica gel etc).
- At the lower part of the case two holes must be open to ensure proper ventilation and enable possible water concentration disposal.

PsoC

- The used materials color should be bright (white or any other) in order to avoid temperature concentration. You may use preferable rubber or any other flexible material. Any user must be able to open the case and access equipment. Avoid metals due to the Bluetooth existence.
- Any opening / holes should be properly protected from rain drops.
- A place for attaching the case (with scratch or any holder) at any wearable clothes or travel bags should be designed.
- Consider that PSoC is significantly smaller than arduino (height: 3cm, cross section: 6cm x 6cm)
- A power bank / solar cell power bank should be fitted at the top of the inside of the case.
- Any electronic equipment (PSoC, power bank) should be placed at the higher place in the case and a hole must be designed above them in order to guide hot air to the environment and not capture it in the case.
- The sensor should be placed at least 2-3cm below the electronics.
- Free air paths should be ensured for both the inlet and outlet of the sensor.
- Air inlet of the sensor should be guided through a chamber that a user could fill with any dehydrating material (salt – rice – silica gel etc).
- At the lower part of the case two holes must be open to ensure proper ventilation and enable possible water concentration disposal.

4.7.2 Product research

Based on the requirements set out by the project, BUND contracted a creative lab to research a small series of alternatives that match the list of requirements to a substantial degree. The creative lab also developed options based on sustainability and recycling perspectives due to the nature of BUND as environmental NGO. The creative lab submitted the following list of options (in German, Figure 4.11):



Schauschau
Verpackungsrecherche für das Projekt »hackAIR«
Schauschau für den BUND
Berlin, 03. 11. 17

Schauschau
Kaufbare Produkte
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Kunststoffdose (PET) transparent mit Aluschraubdeckel
– Maße: h – 97 mm, ø 104 mm
– Kosten: 3,50 Euro/Stück
– Erhältlich bei Modulor
Link: https://www.modulor.de/kunststoffdosen-transparent-mit-aluschraubdeckel-600-ml-h-97-mm-oe-104-mm-oe-100-mm-id.html
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Kunststoffbox (Polypropylen) transparent oder farbig
– Maße: 176 × 127 × 58 mm
– Kosten: 2,45 Euro/Stück, > 120 Stück je 1,95 Euro
– Erhältlich bei TimeTEX Schulartikel
Link: http://www.timetex.de/Aufbewahrung--Aufbewahrungskarton--Aufbewahrungskartons/Kunststoff-Box-A6-farbig.html
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Kunststoffdose (Polystyrol) transparent
– Maße: 130 × 95 × 65 mm
– Kosten: 2,10 Euro/Stück
– Erhältlich bei Modulor
Link: https://www.modulor.de/kunststoffdosen-transparent-rechteckig-130-x-95-x-65-mm-b.html
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Bambus
– Maße: ø 104 mm
– Kosten: 7,00 Euro/Meter
– Erhältlich bei Bambushandel Leipzig
– Notiz: Meterware, auf Nachfrage vermutlich auch kurzer zuschneidbar
Link: http://bambushandel-leipzig.de/Bambusrohre/Bambus-10-11cm.html
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Upcycling-Produkte
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
PET-Flaschen
– Mittig aufschneiden, als Verschluss z. B. Reißverschluss verwenden oder Hälftenineinander stecken
– Halsöffnung für Sensor oder Belüftung nutzbar
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
Lego
– Individuelles Gehäuse aus alten Leosteinen bauen
– Öffnungen für Stromzufuhr und Luftzirkulation können ausgespart werden
– Witterungsbeständigkeit durch verkleben der Steine erreichen
Schauschau für den BUND; Berlin, 03. 11. 17

Schauschau
TetraPak
– Leichtes wasserfestes Gehäuse aus Tetrapak selbst basteln
Schauschau für den BUND; Berlin, 03. 11. 17

Figure 4.11. Screenshot of the options for sensor casing.

4.7.3 Product testing

BUND then selected several options as well a small series of other alternatives. The following options were seen as viable contenders and were used for test assemblies. The most important outcomes of the analysis are summarized in the table below:

Option	Overall verdict
1 – Round box with alum. lid	Too small for sensor and Wemos/Arduino, difficult to drill holes into the bottom and to keep distance between hole and hardware.
2 – square box	Large, space for all electronics, but electronics need extra fixing in box and it opens quite easily, not easy to attach hooks to attach box.
5 – PET-bottle solution	Surprisingly straightforward, easy to assemble. Good design with sensor outlets at lower opening and hardware further up in bottle. Extra plus: upcycling product, not extra casing needed.
7 – Tetra-Pak carton	Also surprisingly straightforward. But long-term durability was questionable. Tetra-pak loses its protection where cut open.

4.7.4 Outcome and suggested casing

These initial testing results formed the base for the final assembly rounds that involved several test casings. The consortium member BUND arrived at the conclusion that the DIY solution of the PET-bottle not only performed very well with regard to the requirements set out by the consortium but also was an upcycling solution that could be assembled with household goods. The following four pictures explain a DIY casing solution that can easily be assembled by any citizen.

Alternatives of course may include other DIY options. Also options like electric boxes or the sewage pipe used by the luftdaten.info project were analyzed and to some citizens might be the preferred option.

Also the consortium members will undertake further analysis as to what casing options perform best in housing the mobile PSOC sensor system.

5 Conclusions

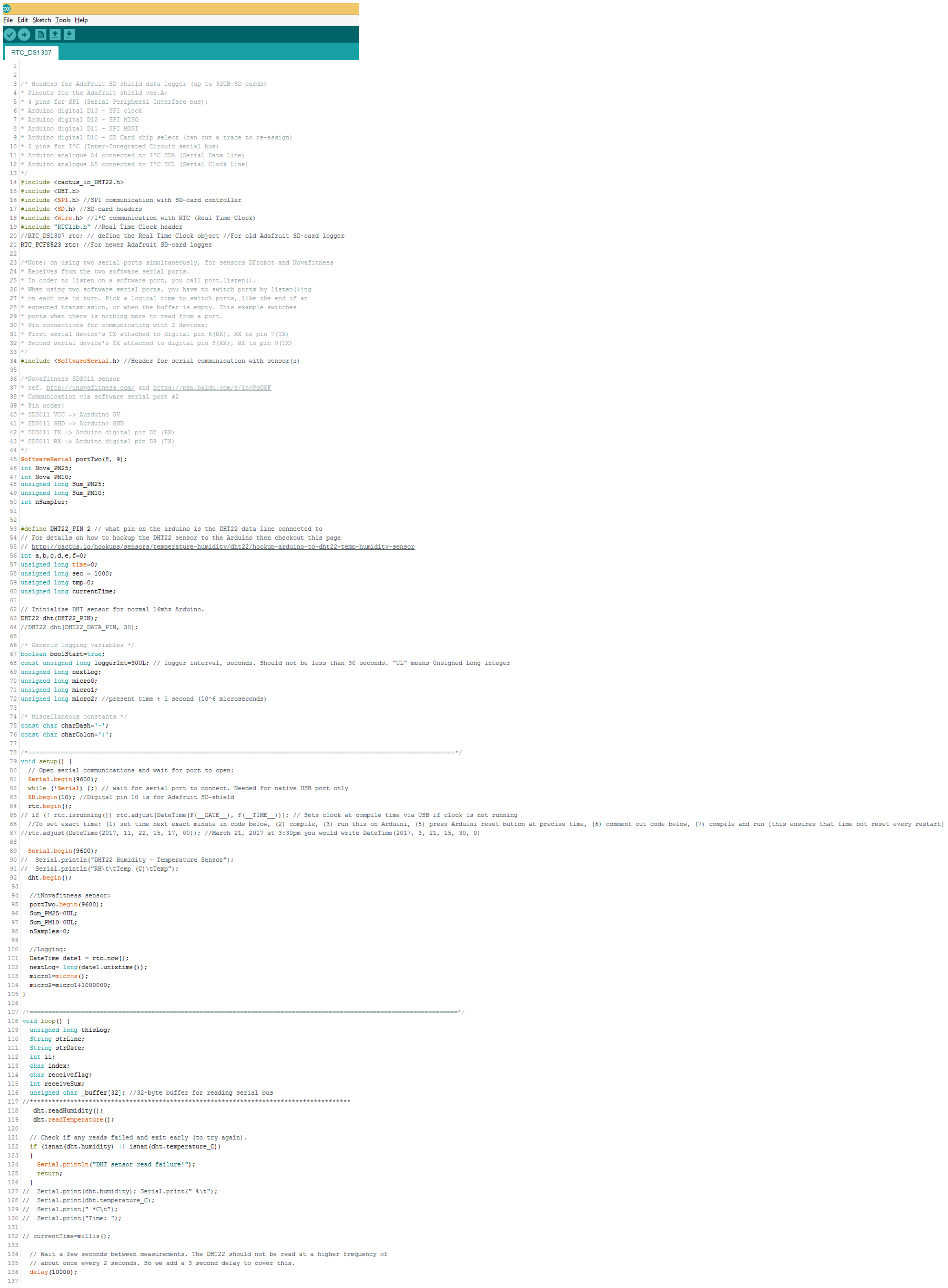
After its finalization, all training material will be easily accessible on the hackAIR web pages.

We learned that the data transfer with the Arduino board via WiFi or Ethernet was quite unstable. Better results have been achieved by using SD card. This would provide the users with the opportunity to access their own data, but on the other hand, it would not be suitable for uploading real-time data to the hackAIR platform. Here we need further discussion within the project consortium.

Nevertheless, first testing results indicate that the data of the Arduino boards might be of relatively good quality. However, further testing trials of all sensors used in hackAIR are currently carried out with results to be expected by beginning of 2018.

BUND has undertaken quite some effort to decide for an appropriate casing for the Arduino/Wemos sensor platform. They seem to have found a suitable solution. However, we still require a case solution for the portable PSOC platform.

Appendix A – Code for Arduino SD card approach



```
/* Headers for AdaFruit SD-shield data logger (up to 32GB SD-cards)
* Pinouts for the Adafruit shield ver.A:
* 4 pins for SPI (Serial Peripheral Interface bus):
* Arduino digital D13 - SPI clock
* Arduino digital D12 - SPI MISO
* Arduino digital D11 - SPI MOSI
* Arduino digital D10 - SD Card chip select (can cut a trace to re-assign)
* 2 pins for I²C (Inter-Integrated Circuit serial bus)
* Arduino analogue A4 connected to I²C SDA (Serial Data Line)
* Arduino analogue A5 connected to I²C SCL (Serial Clock Line)
*/
#include <cactus_io_DHT22.h>
#include <DHT.h>
#include <SPI.h> //SPI communication with SD-card controller
#include <SD.h> //SD-card headers
#include <Wire.h> //I²C communication with RTC (Real Time Clock)
#include "RTClib.h" //Real Time Clock header
//RTC_DS1307 rtc; // define the Real Time Clock object //For old Adafruit SD-card logger
RTC_PCF8523 rtc; //For newer Adafruit SD-card logger

/*Note: on using two serial ports simultaneously, for sensors DFrobot and Novafitness
* Receives from the two software serial ports.
* In order to listen on a software port, you call port.listen().
* When using two software serial ports, you have to switch ports by listen()ing
* on each one in turn. Pick a logical time to switch ports, like the end of an
* expected transmission, or when the buffer is empty. This example switches
* ports when there is nothing more to read from a port.
* Pin connections for communicating with 2 devices:
* First serial device's TX attached to digital pin 6(RX), RX to pin 7(TX)
* Second serial device's TX attached to digital pin 8(RX), RX to pin 9(TX)
*/
#include <SoftwareSerial.h> //Header for serial communication with sensor(s)

/*Novafitness SDS011 sensor
* ref. http://inovafitness.com/ and https://pan.baidu.com/s/1nvPqOXF
* Communication via software serial port #2
* Pin order:
* SDS011 VCC => Aurduino 5V
* SDS011 GND => Aurduino GND
* SDS011 TX => Arduino digital pin D8 (RX)
* SDS011 RX => Arduino digital pin D9 (TX)
*/
SoftwareSerial portTwo(8, 9);
int Nova_PM25;
int Nova_PM10;
unsigned long Sum_PM25;
unsigned long Sum_PM10;
int nSamples;


#define DHT22_PIN 2 // what pin on the arduino is the DHT22 data line connected to
// For details on how to hookup the DHT22 sensor to the Arduino then checkout this page
// http://cactus.io/hookups/sensors/temperature-humidity/dht22/hookup-arduino-to-dht22-temp-humidity-sensor
int a,b,c,d,e,f=0;
unsigned long time=0;
unsigned long sec = 1000;
unsigned long tmp=0;
unsigned long currentTime;

// Initialize DHT sensor for normal 16mhz Arduino.
DHT22 dht(DHT22_PIN);
//DHT22 dht(DHT22_DATA_PIN, 30);

/* Generic logging variables */
boolean boolStart=true;
const unsigned long loggerInt=30UL; // logger interval, seconds. Should not be less than 30 seconds. "UL" means Unsigned Long integer
unsigned long nextLog;
unsigned long micro0;
unsigned long micro1;
unsigned long micro2; //present time + 1 second (10^6 microseconds)

/* Miscellaneous constants */
const char charDash='-';
const char charColon=':';

/*=====================================================================================================================*/
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {;} // wait for serial port to connect. Needed for native USB port only
  SD.begin(10); //Digital pin 10 is for Adafruit SD-shield
  rtc.begin();
// if (! rtc.isrunning()) rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // Sets clock at compile time via USB if clock is not running
  //To set exact time: (1) set time next exact minute in code below, (2) compile, (3) run this on Arduini, (5) press Arduini reset button at precise time, (6) comment out code below, (7) compile and run [this ensures that time not reset every restart]
//rtc.adjust(DateTime(2017, 11, 22, 15, 17, 00)); //March 21, 2017 at 3:30pm you would write DateTime(2017, 3, 21, 15, 30, 0)

  Serial.begin(9600);
//  Serial.println("DHT22 Humidity - Temperature Sensor");
//  Serial.println("RH\t\tTemp (C)\tTemp");
  dht.begin();

   //iNovafitness sensor:
   portTwo.begin(9600);
   Sum_PM25=0UL;
   Sum_PM10=0UL;
   nSamples=0;

   //Logging:
   DateTime date1 = rtc.now();
   nextLog= long(date1.unixtime());
   micro1=micros();
   micro2=micro1+1000000;
}

/*=====================================================================================================================*/
void loop() {
  unsigned long thisLog;
  String strLine;
  String strDate;
  int ii;
  char index;
  char receiveflag;
  int receiveSum;
  unsigned char _buffer[32]; //32-byte buffer for reading serial bus
//****************************************************************************************
   dht.readHumidity();
   dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(dht.humidity) || isnan(dht.temperature_C))
  {
    Serial.println("DHT sensor read failure!");
    return;
  }
//  Serial.print(dht.humidity); Serial.print(" %\t");
//  Serial.print(dht.temperature_C);
//  Serial.print(" *C\t");
//  Serial.print("Time: ");

// currentTime=millis();

  // Wait a few seconds between measurements. The DHT22 should not be read at a higher frequency of
  // about once every 2 seconds. So we add a 3 second delay to cover this.
  delay(10000);


//****************************************************************************************
// Temperatur and humidity measurement


  micro0=micro1;
  micro1=micros();
  if (micro1<micro0) {// In case the counter overflows (gets automatically zeroed) approx every 70 minutes.
    micro2=micro1+1000000; //reset second target
  } else if (micro2<=micro1){ // if 1 second passed
    // Read iNovafitness sensor:
    // Step1: read 10 bytes from serial bus
    index=0;
    while (portTwo.available() && index != 10) {
      _buffer[index] = portTwo.read();
      index++;
    }
    while (portTwo.read() != -1) {}
    // Step2: Check the package integrity
    if (_buffer[0] == 0xAA && _buffer[1] == 0xC0 && _buffer[9] == 0xAB) {
      receiveSum = 0;
      for (int i = 2; i < 8; i++) {receiveSum += _buffer[i];}

      if ((receiveSum & 0xFF) == _buffer[8]) { //package is valid
        Nova_PM25 = ((_buffer[3] << 8) + _buffer[2]);
        Nova_PM10 = ((_buffer[5] << 8) + _buffer[4]);
        Sum_PM25 += (long)Nova_PM25;
        Sum_PM10 += (long)Nova_PM10;
        nSamples += 1;
        //Serial.print(Nova_PM25);
        //Serial.print("\t"); // tab space
        //Serial.print(Nova_PM10);
        //Serial.print("\t"); // tab space
        //Serial.print(Sum_PM25);
        //Serial.print("\t"); // tab space
        //Serial.print(Sum_PM10);
        //Serial.print("\t"); // tab space
        //Serial.println(nSamples);
      }
    }
    micro2=micro2+1000000; // next log in 1 second

    DateTime date1 = rtc.now(); //read time from Real Time Clock (RTC)
    thisLog=long(date1.unixtime()); //convert time to integer seconds
    if (nextLog <= thisLog){

      strDate=String((int)date1.year());
      strDate += charDash;
      strDate += twoDigit((int)date1.month());
      strDate += charDash;
      strDate += twoDigit((int)date1.day());
      strDate += ' ';
      strDate += twoDigit((int)date1.hour());
      strDate += charColon;
      strDate += twoDigit((int)date1.minute());
      strDate += charColon;
      strDate += twoDigit((int)date1.second());

      File dataFile = SD.open("datalog.txt", FILE_WRITE); //Open file
      if (dataFile) { // file is available, write to it:
        if (boolStart){ //Write column header
          // "\n"=new line, "\t"=tab
          dataFile.println("\nDate\ttime\t\tPM2.5\tPM10\thum\ttemp");
          Serial.println("Date\ttime\t\tPM2.5\tPM10\thum\ttemp");

          boolStart=false;
        }
        // make a string for assembling the data to log:
        if (nSamples==0) nSamples=-1; //prevent divide by zero if there is no data. Outputs negative value instead
        strLine = strDate;
        strLine += "\t";
        strLine += (float)Sum_PM25/(float)(10*nSamples);
        strLine += "\t";
        strLine += (float)Sum_PM10/(float)(10*nSamples);
        strLine += "\t";
        strLine += (dht.humidity);
        strLine += "\t";
        strLine += (dht.temperature_C);
        strLine += "\t";
//      strLine += nSamples;

        dataFile.println(strLine);


        dataFile.close();
        Serial.println(strLine); // print to the serial port too:
     //   Serial.print("\t");
      } else { // file isn't open, pop up an error:
        Serial.println("error opening datalog.txt");
      }



      nextLog=nextLog+loggerInt; //logger interval
      micro1 = micros(); //reset timer for Shinyei
      Sum_PM25=0UL;
      Sum_PM10=0UL;
      nSamples=0;
    }//endif date=nextLog
  }//endif micro2<micro1 (1 second logger-interval)
}

/*========================================================================================================================*/
String twoDigit(int ii){
  // Converts an integer to a right-justified two-character string left-padded with zeros, e.g. 1 => "01"
  // This function is used to output date/time, e.g. months, minutes, hours, minutes, seconds
  String string2=String(ii);
  if (ii<10){
    return String("0" + string2); //pad with leading zero
  } else {
    return string2;
  }
}
```