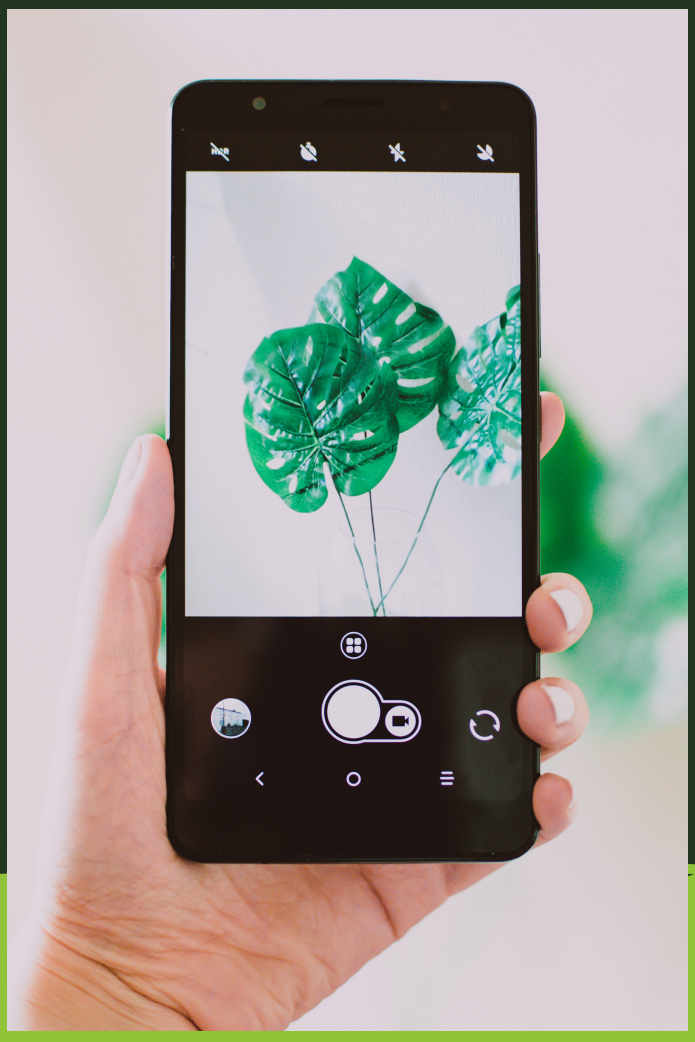
School Gardens for Future Citizens

eSchool Garden

FROM REAL TO VIRTUAL GARDENS



DEVELOPING INSTRUMENTS WITH SENSORS

PART II

ERASMUS+ 2018-1-ES01 KA201-050599

CONTENTS

eSGarden

EXERCISE 2

BY SARA BLANC
sablacla@upv.es

INTRODUCTION

Weather stations and other agriculture instruments created with sensors are directly related to your school garden since the data collected by these instruments shows how different weather and soil variables influence crop irrigation and crop health

Activities focused on developing instruments and software applications improve many soft competences in your students, such as mutual respect, collaboration, active listening, teamwork and project solving, among others.

Firstly, these activities include the use of digital tools and platforms, programming, understanding of the interoperability of machine-devices, database learning and data analysis and exploitation. Thus, the learning experience strengthens many skills and competences in the European DigCom framework:

- Information search and retrieval.
- Communication and collaboration in digital environments.
- Safety in digital environments.
- Creation of digital content.
- Problem solving.

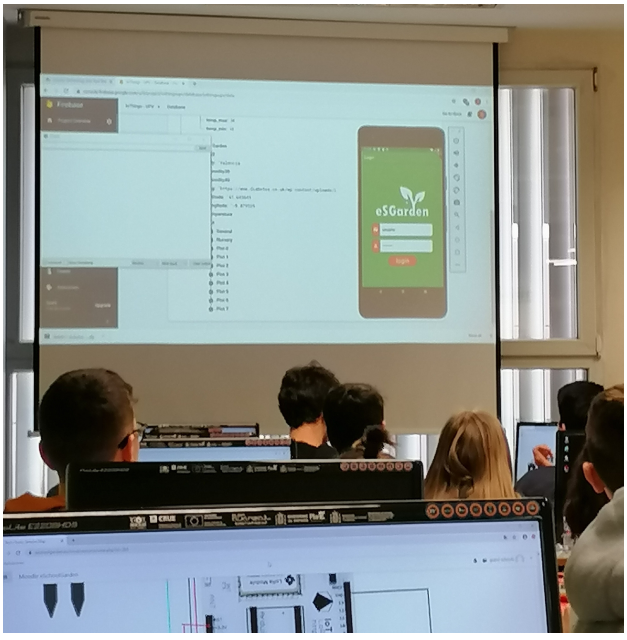
Secondly, environmental competences are also strongly connected with these activities. By learning in technology, the real final goal is directed at the efficient use of natural resources and the responsible use of water. For example, various cognitive items can be assessed at the end of the activity, such as the following:

- The student understands how much water is needed to irrigate a small garden.
- The student understands which elements influence the contamination of the environment.
- The student understands how pollution affects the garden.
- The student understands how the garden is dependent on weather conditions.

Socio-emotional and behavioural objectives are also developed around responsibility and active transformation in individual actions.

BRIEF DESCRIPTION

The activity is suitable for team or individual work as part of technology or informatics classroom projects.
The activity is recommended for students more than 15 years old. It contributes by encouraging students' creativity, motivating students through the use of different resources, and combining collaborative work and critical thinking. This activity is also an opportunity for teachers to get more out of the eTwinning platform



GOAL

The goal is to build a weather station and a soil humidity probe. This activity can be complemented with other tasks such as:
-Sensor data analysis and exploitation
-Creation of a blog
-3D printing for boxes and gadgets

LEARNING STEPS

*Program your activity **in a sequence of phases.** For example:*

- *Analysis of components, artefacts, gadgets and tools to build your weather station or your soil humidity probe.*
- *Health and safety training.*
- *Conceptual design of the 3D prototype.*
- *Implement Arduino code and debug the code in the classroom.*
- *Make a testing.*

MATERIAL

*In this exercise **we build a weather station and a soil humidity probe** using two **Arduino IoT 33 Nano** boards with WiFiNINA chip.*
Each Arduino microcontroller is the core of a node as follows:

***ID_NODE "01"** – Semantically identifies a general space dedicated to ambient measurements (the weather station).*
***ID_NODE "04"** – Semantically identifies a plot space; for example, one dedicated to growing onions (the soil humidity probe).*

You will need to attach several sensors to each board, such as those for ambient temperature and humidity, brightness, air quality, etc. Additionally, for the plot space, you can look for specific soil humidity sensors, of varying complexity, but include several because the crop humidity should be checked at minimum on the surface and at depth.

We recommend you the Arduino IDE software or the online Arduino development platform.

***Remember to download our basic project template for Arduino 33 IoT NANO boards:**.*

LEARNING STEPS:

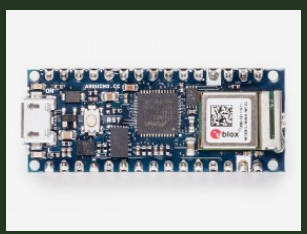
ANALYSIS OF COMPONENTS, ARTEFACTS, GADGETS AND TOOLS

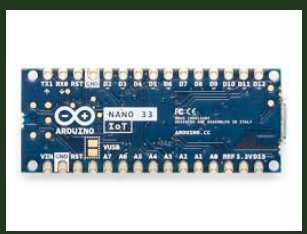
The first step in the design process is the selection of materials and resources according to your economic, security, and environmental safety criteria.

Carry out the following steps **to prepare the activity** with your students:

1. Design your nodes.
2. Choose your sensors and calibrate them.
3. Prepare your new database.
4. Define a control panel.

In the exercise we use ARDUINO IoT 33 Nano with Wi-Fi chip
https://www.arduino.cc/





DESIGN YOUR NODES

As a general overview, the simplest module requires the following components:

- A box to protect the electronic devices.
- The electronic kit is based on an Arduino microcontroller. The manufacturer offers several kits and an even wider variety can be found elsewhere on the market.
- A protoboard to connect the sensors to the microcontroller.
- A power bank to avoid using electrical cables for power supply. However, in the informatics classroom you can use the USB port for power.

CHOOSE YOUR SENSORS AND CALIBRATE THEM

Many analogue sensors require calibration. The alternative is to use digital sensors. Nevertheless, some analogue sensors are easy to calibrate by determining their load resistance and sensitivity; however, their behaviour may depend on temperature and ambient humidity.

There are some available libraries you can use, such as DHT spectrum, for example (https://www.arduinolibraries.info/libraries/dht-sensor-library)

There is also a DHT library for digital sensors.

```
DHT.read11 (<pin_assigned>);

int amb_temperature = DHT.temperature;
int amb-humidity = DHT.humidity;
```

On the other hand, there are simple analogue sensors with an easy calibration process, as is the case for soil humidity sensors.

A humidity reading of 0% equals completely dry soil, while 100% is equivalent to the sensor being practically submerged in water.
Use a glass of water to read the maximum value of your sensor, and set the minimum value with the sensor being very dry.

Analogue pins are read with the **analogRead**(<pin_assigned>) function. The Nano board returns a value between 0 and 1023 (10-bit analogue-to-digital converter), so you can carry out a fun and easy activity with your students.

PERCENTAGE

How can we know if the humidity is higher or lower than 50%?
For example, a completely dry sensor shows a value (v1), and the same sensor submerged in water shows another value (v2), with v1 > v2 (reverse logic). A reading equal to v1 corresponds to 0%, while a reading equal to v2 corresponds to 100%.

We have a range of (v2 - v1) values, so the humidity will be higher than 50% if the value is higher than v1 + ((v2 - v1) x 0.5).

NOTE - Take into account that many analogue sensors need to be calibrated according to the specifications detailed on the vendor's datasheets.

PREPARE YOUR NEW DATABASE

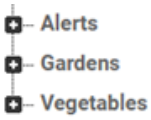
In Exercise 1, we already chose Google Firebase as the database cloud service. It is recommended that you are skilled in security rules and manage user permissions. Security rules define how users access to your database to both writing new data or read existing data.

```
For example:
"Gardens": {
".write": "auth.uid != null && auth.provider != null ",
".read": "auth.uid !=null",
    }
```

In the example, the write option is only permitted for identified users (e.g. teachers), through a login and password. However, anonymous users are only allowed to read existing data. Access to the app by clicking here.
If you already have some practice with the Firebase console, this document explains you how we have structured the database to connect it with the open source app available by clicking here. You could follow this model or adapt it to your future projects.

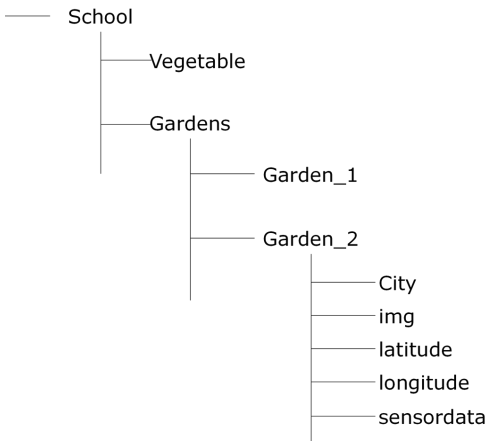
eSGarden Firebase Structure (an example)

Our higher level contains three collections:



Alerts: Sensor values outside the thresholds assigned in the app sensor settings. These alerts are based on checking the value of each new reading received from a sensor and comparing it against the thresholds. They are defined in the app.

Gardens: To give flexibility to your activities, we have considered the possibility of creating more than one garden for the same school. The structure is as follows:

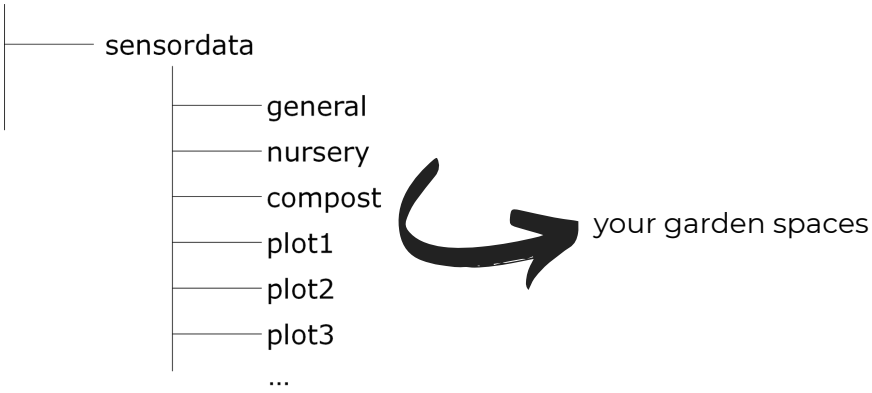


eSGarden Firebase Structure

Inside a “Garden” there are several items of metadata such as:

- City or garden location.
- An image URL representing your garden.
- The latitude
- The longitude

Sensordata: Data received from sensors.



Each space is indexed as follows:

- City: Field replicated from its parent.
- Data: Sensor measures.
- Items: Identify both Sensors and Actuators assigned to the space. The assignment is carried out with the “teacher” user role, with the privilege to add and remove gardens, plots and sensors.
- Name: Name shown in the app.
- Parent: Name of the garden replicated from its parent.
- Valve: Active means enabled. Max and min values represent operation thresholds. The valve is assigned to a unique sensor space.
- Vegetable: <only available in plots> Vegetable or crop in the plot.

eSGarden Firebase Structure

Items: The sensors currently available in our app are the following:

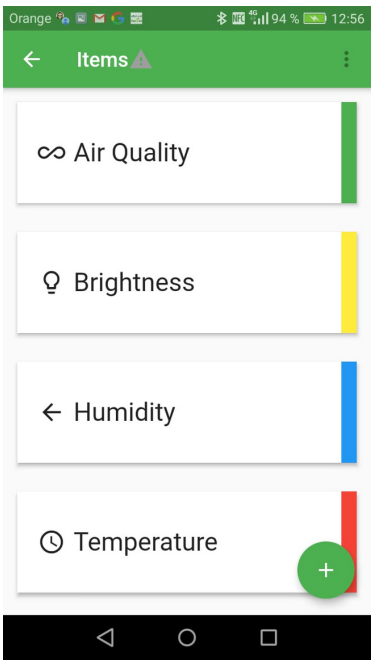
- Ambient humidity (not available in plots)
- Ambient temperature (not available in plots)
- Soil humidity (only plots)
- Soil temperature (only plots)
- Brightness (not available in plots)
- Air quality (CO2 by default – not available in plots)
- Noise (not available in plots)
- Wind (force and direction - not available in plots)
- Compost temperature (only plots)
- Compost humidity (only plots)

It must be taken into account that the sensors are mapped as indicated below; if a new sensor or actuator is added, it must be done by assigning it a number that is not already assigned. The default mapping is as follows:

sensor types

In our Arduino example, sensors are identified as follows:

TYPE 00: Brightness - 1 value
TYPE 01: Temperature - 1 value
TYPE 02: Humidity - 1 value
TYPE 03: PH - 1 value
TYPE 04: Relative noise - 1 value
TYPE 05: Air Quality - 1 value
TYPE 06: Rainfall - 1 value
TYPE 07: Wind - 2 values
TYPE 08: Soil temperature - 4 values
TYPE 09: Soil humidity - 4 values

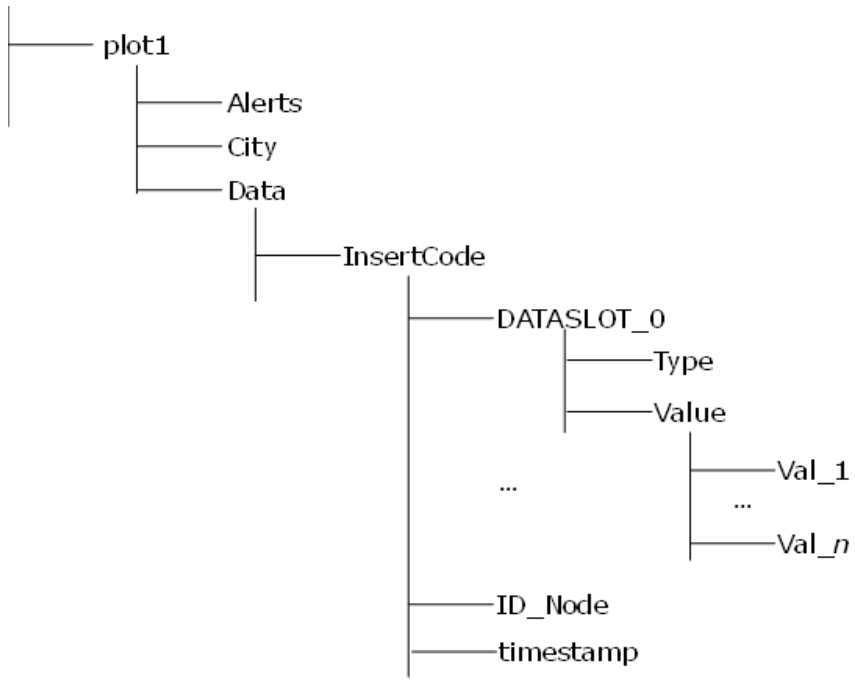


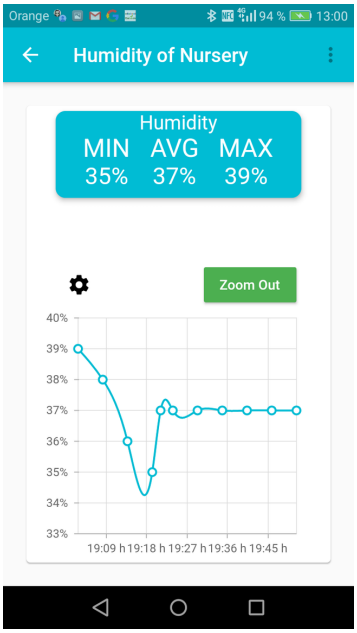
eSGarden Firebase Structure

Data: It corresponds to the insertion registers pushed from the sensor nodes (Arduinos). The structure is as follows:

- InsertCode: Insertion mark in Firebase.
- DATASLOT_i: A new push action in Firebase contains one or several measurements. These measurements are registered with the same timestamp. Each DATASLOT corresponds to one sensor.
  - Type: Sensor type. See Items mapping.
  - Value: One or several values*.
- ID_Node: Node identifier.
- Timestamp: Time mark in Firebase (reference: UNIX pattern).

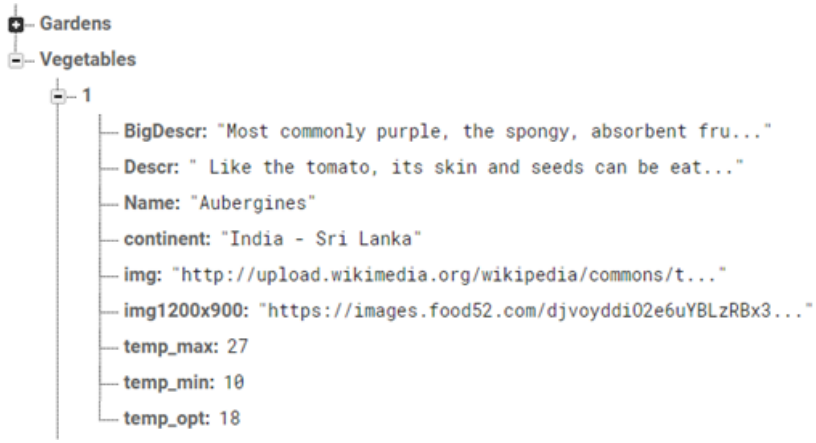
* Sensors can be uni or multi-parametric. A multi-parametric sensor takes a reading of various parameters. For example, the wind requires [Direction, Force]. Another example is the “soil humidity” or the “soil temperature”. These sensors are expected to return 4 values per reading: [measured at 20 cm deep, measured at 30 cm deep, measured at 40 cm deep, measured at 50 cm deep].





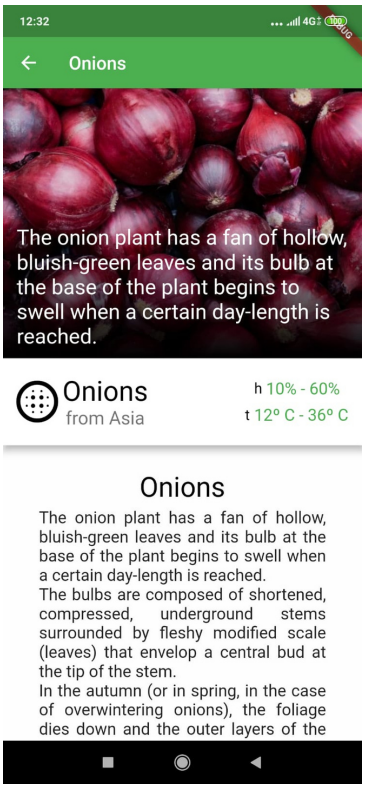
eSGarden Firebase Structure

Vegetable: This indicates the different crops that can be used in the application. We have introduced the most recommendable crops for a growing cycle coinciding with the school terms in Spain. In addition, they are also the most resistant crops and can, in general, give better results.



"Vegetables" are indexed in numerical order. Each has a structure that is used to describe it. This description is displayed on the app.

- BigDescr: General description; around 100 words.
- Descr: Short description.
- Name: Vegetable name.
- continent: Origin.
- img: Image URL.
- img1200x900: Image URL.
- temp_max: Maximum recommended temperature.
- temp_min: Minimum recommended temperature.



Additionally to data received from the sensors, the database also includes a .JSON object to describe our garden vegetables.

We can change, add, or modify this vegetable object directly in the IoT Cloud or by importing a suitable object. The expected format contains both metadata and descriptive information about each vegetable.

DEFINE A CONTROL PANEL

If you are skilled enough in programming, we offer to you the open source of our app. Download it and make the neccessary changes to adapt it to your needs (see next page).
Our app creates the Firebase structure explained above **automatically**.

If you are not skilled enough in programming, there are many IoT cloud platforms that integrates the database plus the control panel with mininum effort. For example, you can explore ThingSpeak or Ubidots.

Find a simple code example to connect your Arduinos IoT 33 NANO to these platforms in:

ThingSpeak

ubidots

We offer a detailed guide of excercises, including FCM notifications in Firebase and MQTT Wordpress link (only in Spanish) (click here).

DEFINE A CONTROL PANEL (AN APP IN VISUAL STUDIO)

To adapt our app to your needs, take into accont the following items.

The project is built with Android Studio. There are two initial requirements to open the project.

- The first is that you need to install Android Studio. You can download it from its official webpage: https://developer.android.com/studio.
- The second requirement is to install the Flutter plugin, needed to open our project files. You can get it by following this brief tutorial.

After you have installed Android Studio and Flutter, you will need to download the project from the GitHub repository.

Downloading is explained in this video (click here).

Next, you need to import the project into Android Studio. This is explained in this video (click here).

When you import the project, there may happen to be a lot of mistakes. See how to fix them in this video (click here).

Next, you must change the name of the package according to your school. You can do this by following this brief tutorial in this video (click here).

Take into account that our project uses a **Syncfusion package** for graphics. It is a licensed version, so you will receive a warning that it is a free trial.

Because our app allows two different roles:

1. students or general public, and
2. registered teachers ...

... we have included a security code in registration. This is the “Teachers Code” which is fixed within flutter code. **Change it in file lib > screens > authenticate > register.dart**
Now, you have the software environment ready, but you need a Firebase Database project to communicate with the app.

How can you link your Firebase new project with our app in Visual Studio?

Firebase setup: We explained in previous exercises the structure that our app will create in Google Firebase. Therefore, you will need to create a new Firebase Project with a Realtime Database.

Vegetables are not created by the app. Add the collection, which is available from the GitHub repository too (click here).

Before continuing, you have to set the same name you used to change the app package. To do this, go to your Firebase Account and follow the instructions in this video to embedd the package name in the code and, this video to link the app to your project.

TESTING

Now, all the requirements needed to start up have been accomplished, but you still need a device to run the app.

There are two different options to test the app: using your personal phone or creating a virtual device on your PC.

If you choose the first option, you only need to configure your phone to enable USB Debugging. You may have to unlock developer actions to configure the USB debugging mode to active. It is easy to do but different for each device.

On the other hand, if you want to run the app on your PC, just follow the brief tutorials in two videos that explain how to set up an emulator (click here 1) ... (click here 2).

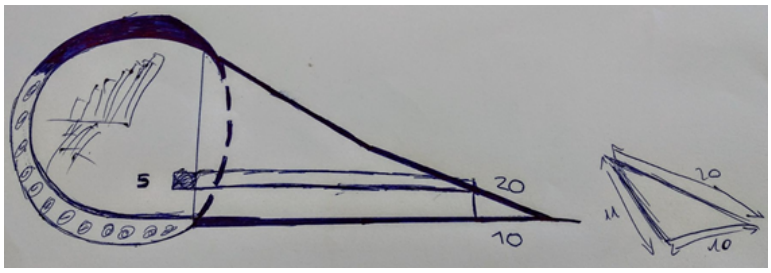
HEALTH AND SAFETY TRAINING

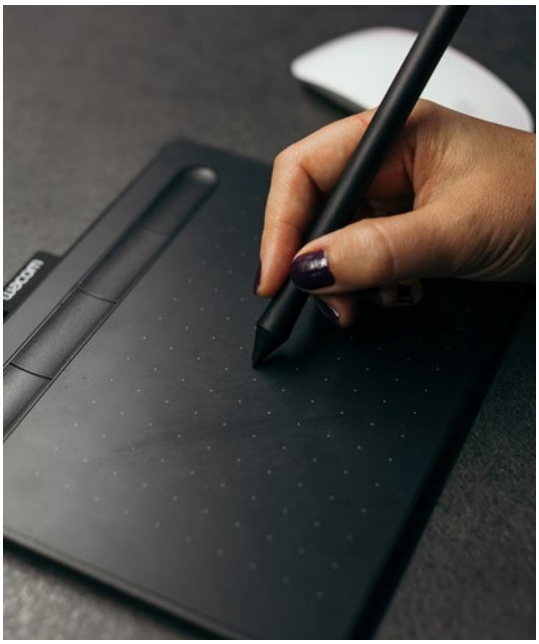
Explain to your students the most important recommendations for sitting in front of a computer. They need to learn about avoiding bad postures; for example:

- Sit correctly in the chair and regulate the height of the table and chair so the screen is in front of your eyes. Your eyes should be level with the top edge of the screen.
- Keep your back upright and supported by the back of the chair.
- Keep the wrists of both hands straight, without twisting or bending. Do wrist exercises if you spend many hours in front of the computer.
- Adjust the distance between table and chair. Keep the elbow of the arm that handles the mouse flexed. Relax your shoulders.
- Feet flat on the ground. Legs at a 90° angle. Rest every hour. Get up and take a little walk!

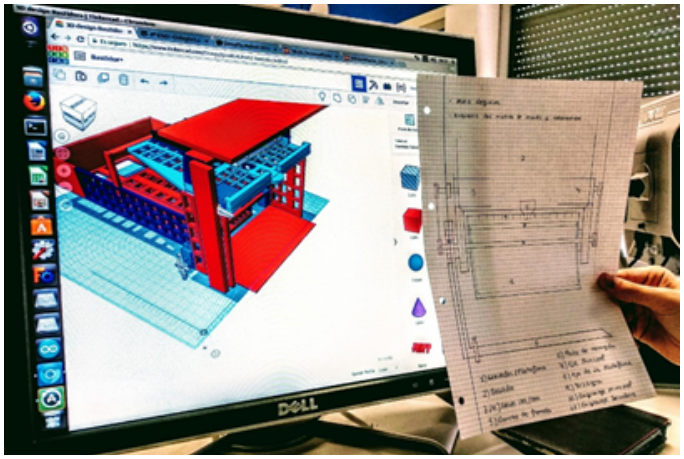
CONCEPTUAL DESIGN OF THE 3D PROTOTYPE

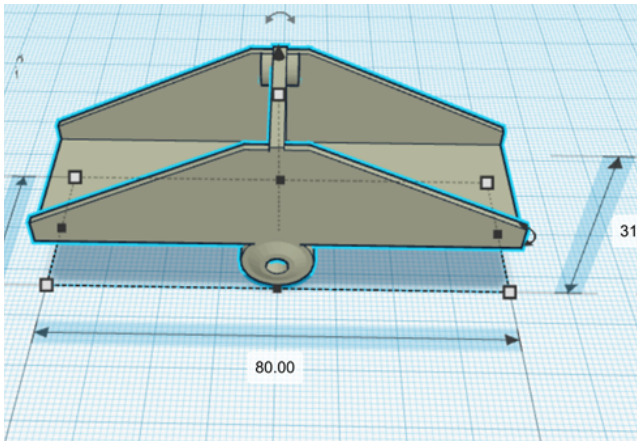
Design the final product in a MODULAR FORM; that is, design the different parts of our final product (robot). How? Make sketches with pencil and paper or a graphics tablet.





Additionally, there are different 3D software such as Tinkercad, FreeCAD, Fusion 360, etc. Connect the software with a 3D printer to make the weather station wrapper box or pieces for a sensor (such as a pluviometer).





There are pages, such as thingiverse.com, hosting hundreds of thousands of 3D designs or models that other people have made, which you can download and adapt to your needs.

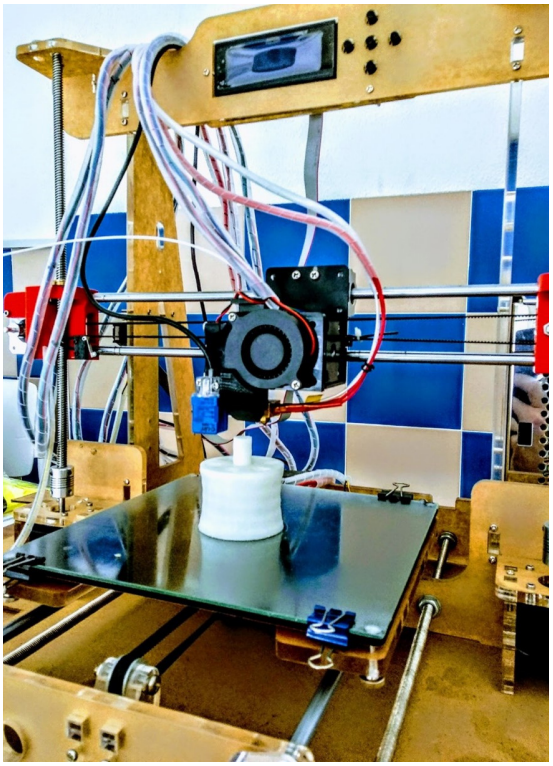
Once the 3D design has been made in a computer program, we must export our design to the file format with **.stl** extension.

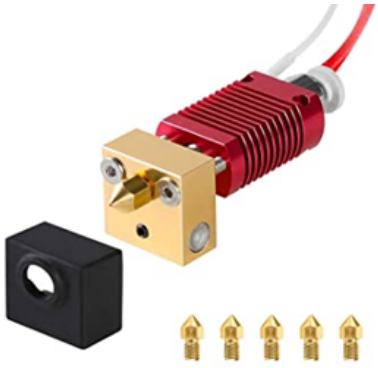
3D printing is a technology, specifically 3D printers that perform additive manufacturing where a three-dimensional object is created by superimposing successive layers of certain materials, mostly plastics. This object is an object previously designed by us using 3D modelling and / or design software.

MATERIAL

Using an FDM printer, the basic material required to print a model is:

3D printer: *This device is composed of several pieces. It is important to understand at least two of them. First, a heated bed where the molten filament is deposited in different XYZ coordinates to form the end object. Second, the dot end and extruder, which reaches high temperatures. It has a nozzle through which the molten filament comes out. The diameter of the filament can be of various sizes (0.2mm, 0.3mm, 0.4mm, etc.), depending on its purpose, the most common size being 0.4mm.*

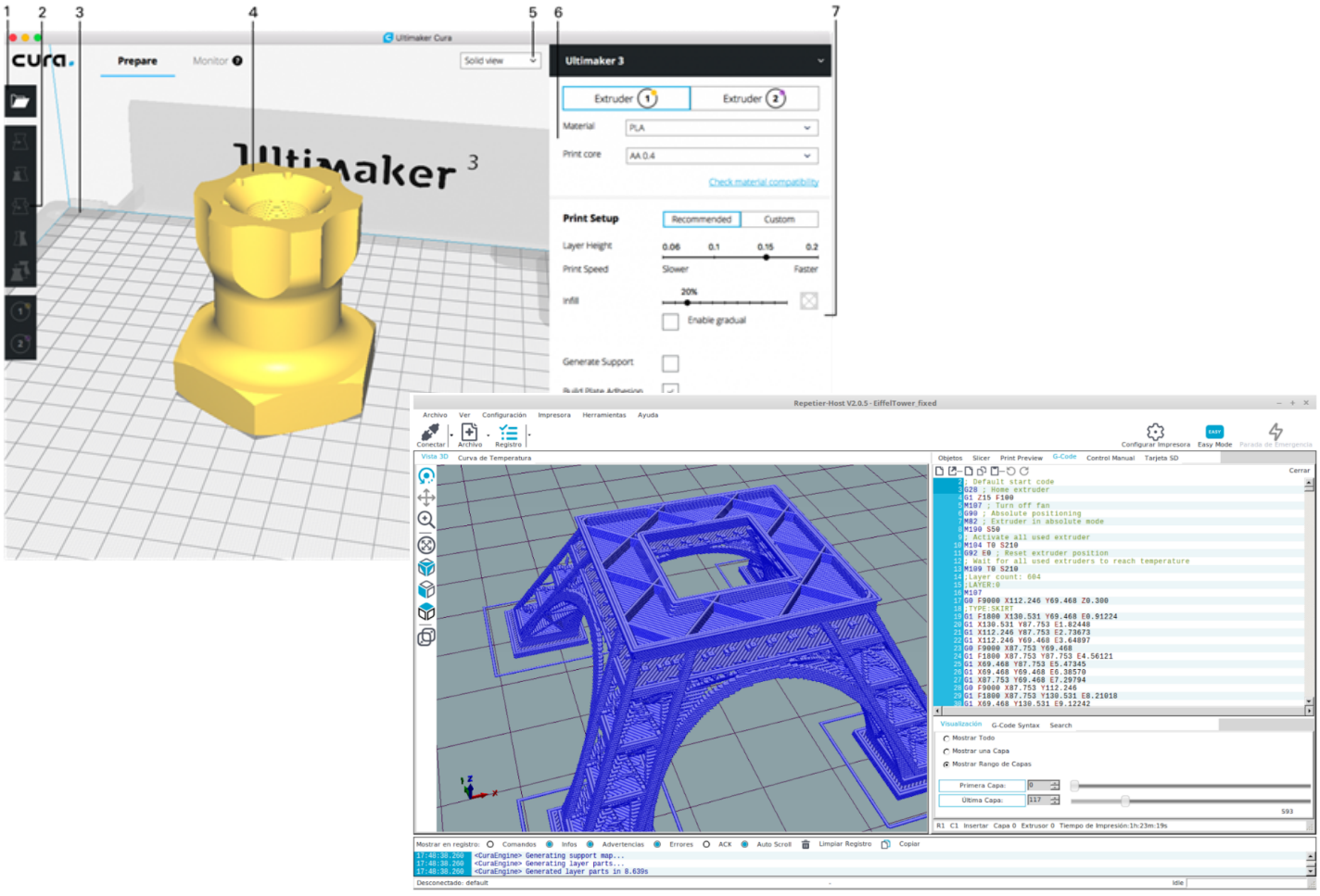




3D filament: *Resistant thermoplastic material that withstands high temperatures well. It tends to bend and warp in a similar way to hard rubber until it eventually breaks. Its extrusion temperature is between 220–270* **℃**, *but this depends on the type of filament. There are various types of filament for 3D printers: ABS, PET (food use), Flexible and PLA (the most common). They are sold in spools of between 500 grams and 1000 grams, normally with a thickness of 1.75mm.*

MATERIAL

Software: *You need specific software for 3D printing to configure the different parameters of the printer and 3D printing and, to create, delete or modify parts of the object to be printed. We will obtain a g-code file. Some examples of 3D printing programs are UltiMaker Cura and Repetier-Host, among others. Additionally, some 3D design / modelling platforms, such as Tinkercad, give you the possibility to send your design directly from the program to a dedicated printing company who can print it for you.*



g-code and g-code file. *G-code is the programming language most widely used in computer numerical control and 3D printers. The g-code file is the way to export the design of your object. It contains instructions that are understandable to your 3D printer.*

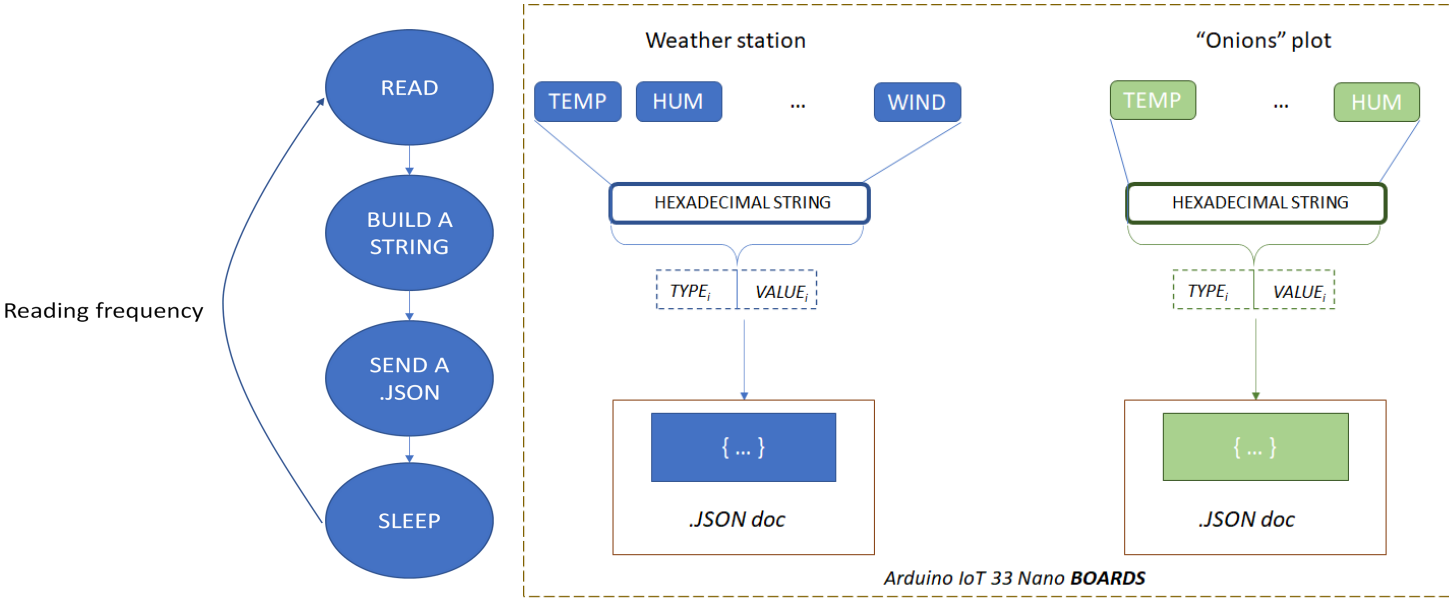
IMPLEMENT ARDUINO CODE AND DEBUG THE CODE

In **our example** for Arduino 33 IoT NANO boards, the Arduino microcontroller reads and sends sensors information cyclically. Each cycle allows our node to make a reading of one or several sensors. The time elapsed between two reading cycles of the same sensor string is known as a reading frequency.

Note that there are no reliability mechanisms at network level, such as acknowledgement ACK. Thus, nodes assume a non-critical transmission. We assume that in the event of a message being lost, the system will not suffer any serious consequences.

We assume that you read all your sensors in your node in the cycle.

The general idea is depicted in this figure:



First, your Arduino builds a string in hexadecimal format. The string contains a list of data read from the attached sensors in a reading cycle.

Depending on the sensor, you need to adapt the format from hexadecimal to decimal or float. In our code, we use a maximum of 5 characters per reading.

Reading with 5 characters
2, 35, 21.35, 1.24, 120, 1023, 345.1

Reading with more than 5 characters
345.24, 1000.4

The .JSON object is built using specific functions of ArduinoJson library. You can find more information at https://arduinojson.org/.

The string is composed of the addition of each sensor value preceded by its type. The types are pre-defined and identified as numeric.

Remember, from character 2 till the end, the string represents tuples as follows:

TYPE OF SENSOR *2 characters*	SENSOR READ **VALUES** *The number of characters depends on the type of sensor*

Each value read by a sensor occupies **5 characters** in our example. Additionally, there are some sensors which send **more than one value per reading.** For example, wind requires two values (force and direction), and soil humidity requires various profiles per reading (a profile equals a certain depth).

Remember our semantic annotation to identify sensor types:

sensor types

In our Arduino example, sensors are identified as follows:

TYPE 00: Brightness - 1 value
TYPE 01: Temperature - 1 value
TYPE 02: Humidity - 1 value
TYPE 03: PH - 1 value
TYPE 04: Relative noise - 1 value
TYPE 05: Air Quality - 1 value
TYPE 06: Rainfall - 1 value
TYPE 07: Wind - 2 values
TYPE 08: Soil temperature - 4 values
TYPE 09: Soil humidity - 4 values

The last action in our cycle is a .JSON frame building as follows:

```
{
"DATASLOT_0"
: { "Type" : 5, "Value": [ 74 ] },
"DATASLOT_1"
: { "Type": 1, "Value": [ 24] },
"DATASLOT_2"
: { "Type": 2, "Value": [ 0 ] },
"DATASLOT_3"
: { "Type": 6, "Value": [ 250 ] },
"ID_Node" : 1,
"timestamp" : 1605662192815
}
```

Data sensor transformation in Arduino

The string is composed of the addition of each sensor value preceded by its type. The types are pre-defined and identified as numeric.

Besides reading sensor values, the Arduino should also provide suitable data formats. Arduino IoT 33 Nano contains an analogue to digital converter of 10 bits. Therefore, data read from analogue sensors has a range between 0 and 1023.

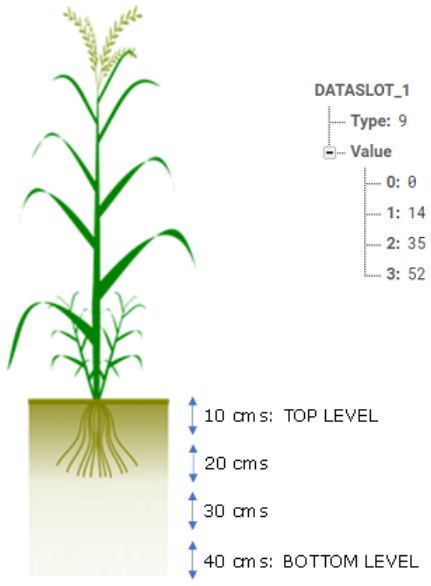
The Arduino microcontroller must send data as expected in the app representation. Examples:

- **Humidity**: Read data must be transformed into a percentage between 0% and 100%. The function “map” can help you with transforming data (1).
- **Temperature** °C: The eSGarden app uses Centigrade.
- **Wind**: It is a multiparametric sensor with two values: force + angle.

TYPE: Wind	Force	Angle
%02X	%05X	%05X

- **Soil temperature or humidity:** In the case of multiparametric probes, which can measure the temperature or humidity at different depth levels (or profiles), values must be sent following a unique type.

TYPE: Soil humidity	Top level	Intermediate	Intermediate	Bottom level



DATASLOT_1
Type: 9
Value
0: 0
1: 14
2: 35
3: 52

And the .JSON object built in Arduino:

“DATASLOT_1”

:{“Type”: 9, “Value”: [0, 14, 35, 52]},

The identification of our garden spaces

Arduino Nano boards can manage analogue and digital sensors. We use one board per garden space (weather station, compost machine, seed nursery, and each plot). However, you can replicate the space identifier ID_NODE) within several boards if you need more than one Arduino Nano in the same space.

Nano boards are identified with a code which represents the garden space. The identifier is defined in “send_params.h” as follows:

```
#define ID_NODE "01" //01 identifies the school weather station
```

or

```
#define ID_NODE "04" //01 identifies our plot of onions
```

The number assigned to each ID_Node is predefined in your semantic annotation. Thus, if you are using the eSGarden example, you must use the following identifiers:

```
"Testing" => 00 [not used]
"General" => 01
"Nursery"  => 02
"Compost" => 03
"Plot 1" => 04
"Plot 2" => 05
"Plot 3" => 06
"Plot 4" => 07
"Plot 6" => 08
"Plot 7" => 09
```

If you need more plots, add them in the string variable defined into “send_params.h” code (“Testing” was included for development testing purposes):

```
String PLOTLIST[] = {"Testing", "General", "Nursery", "Compost", "plot 1", "plot 2",
"plot 3", "plot 4", "plot 5", "plot 6", "plot 7"};
```

Firebase connection and data interaction

In our previous practice exercise (Part I), remember the step dedicated to explaining our code lines to connect the Arduino WiFiNINA chip to your school network and the available functions for Firebase connection and interaction.

MAKE A TESTING

Some issues should be tested in the real garden space. Use the following checklist as a resource:

- Wi-Fi access point is out of range. We suggest a solution in Part III.
- The connectors come loose. In this event you can solder the latest version of your diagrams.
- The box is not waterproof. Try introducing insulators for boxes.
- Battery problems. You can replace the power bank with a set of batteries. We recommend that you investigate inducing the microcontroller into a low-power mode during idle or sleep time.

YOUR ACTIVITY:

ADAPT THE EXAMPLE

You are ready to implement code with your own toolkits. Remember these ideas:

1. Organise your gardens, both the real and the virtual.
2. Design sketches, papel draws and 3D models to integrate the kit and sensors.
3. Program the code in the Arduinos, prepare the database and the control panel to observe data.
4. Make the prototype useful at the garden. A weather station or a soil humidity probe are good projects to be true engineers.

Next exercise ...

In the next exercise we offer you a practical activity to implement a long rage connection using LoRa RF technology.

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
sepie
SERVICIO ESPAÑOL PARA LA INTERNACIONALIZACIÓN DE LA EDUCACIÓN

Project name: eSGarden – School Gardens for Future Citizens
Number: 2018-ES01-KA201-050599

Erasmus+