

**TECHNICAL**

**REPORT**

Incase Project

INCASE Project is a research project meant to develop 3.0 revolution vision in smart technologies.

In this research project, we want to develop smart use of energy and find out if implementing a Smart building can lead to energy cost.

A smart building is a building that have been conceived to integrate and use avdnaced digital technologies in order to optimize several things sush as environnemental impact, daily tasks, safety…

The purpose of this report is to describe the conception and the technical specifications for the following mission:

“*Create an app to collect data from small sensors groups for a smart use of energy*”

**Table of content :**

[I - Project context: 3](#_Toc486897905)

[1) Smart use of energy : 3](#_Toc486897906)

[2) Sensor selection: 4](#_Toc486897907)

[3) Sensor network: 4](#_Toc486897908)

[4) Adapt the conception to a building: 6](#_Toc486897909)

[5) App architecture: 7](#_Toc486897910)

[II - Project developpement: 10](#_Toc486897911)

[1) Objectives: 10](#_Toc486897912)

[2) Arduino Developpement: 10](#_Toc486897913)

[3) App developpement: 11](#_Toc486897914)

[III - Next steps of project: 14](#_Toc486897915)

[1) Project results: 14](#_Toc486897916)

[2) Improvements: 14](#_Toc486897917)

[IV - Sources: 16](#_Toc486897918)

[V - Javadoc : 16](#_Toc486897919)

# I - Project context:

As mentioned previously we want to implement smart use of energy in a smart building. Since we don’t have already a smart building, the first step is to implement it and add to an existing building the electronic and informatic we need.

At this point, we need to detail what is the smart use of energy we want to make.

create small groups of sensors to mesure key values, collect them and analyse them to optimise energy use while providing confort.

To make a smart use of energy we have to optimise the use of energy while keeping people confortable in this smart handled environnement. This environnement could be inside or outside, but in our case, this environnement is meant to be in a smart building. Consequently we will assume that the environnement of our study, will be a simple room. Once the room is properly handled with this Smart energy, we can simply extend the principal to other rooms and then have a complete smart energetic building.

There are several ways we use energy in a room:

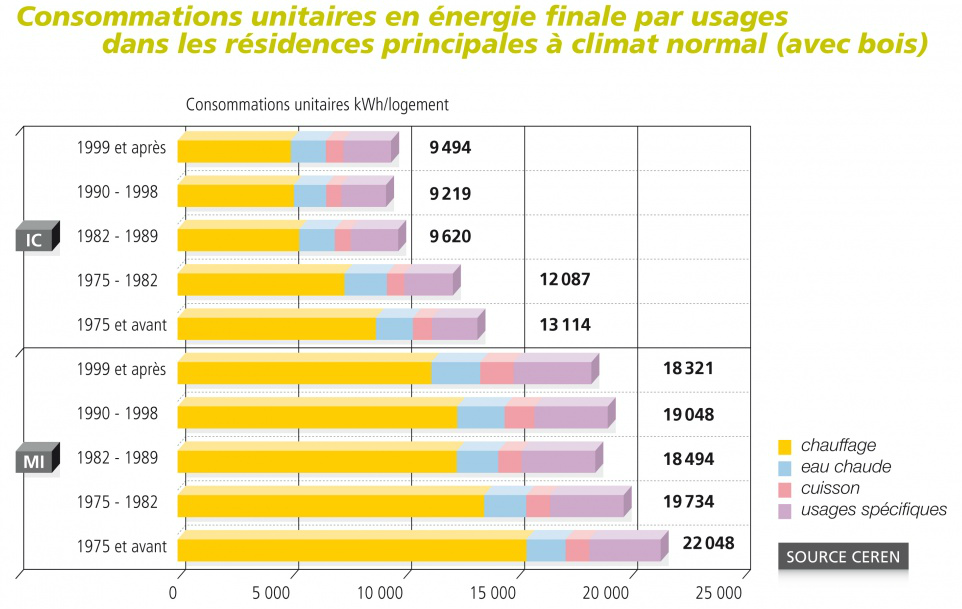


Figure 1: Energy consumption by usage, in houses, in tempered places.

IC: Collectif building - MI: Individual house - Heating – Warm water – Cooking – Other

On this graphic, we can see that the main use of energy is the heating. We will first focus on optimizing heating energy for this prototyping process. In order to optimize energy uses even more, solutions could be found in order to lower energy uses in the other usage. A smart building could optimize the lighting or the electrical tools (such as computers) consumtion for example.

Several studies have alredy tried to evaluate what parmeters make people comfortable1 in a room regarding heat. This mesurments led to calculate thermic comfort of people.

We can list the following parameters influencing thermic confort:

* Air temperature
* Wall temperature
* Humidity
* Air movement speed

Others parameters can have an influence on thermic comfort. Some, such as cloths, weather, age… can’t be measured easily and consequently can’t easily be handled through an automated system. So, these parameters won’t be taken into account here.

Yet, others are easy to implement and can have a big impact on energy consumtion like daytime (thermic comfort is better at night with a lower temperature) and presence(we can stop heating when nobody is inside the room).

Once we now which informations we want to collect, we will have to think of how we want to gather them and display them on the Android app which is the main goal of the mission.

# II - Hardware:

## Sensor selection

In order to make precise mesurements on thermic comfort in rooms, we will take into account all the measurable parameters we mentioned earlier:

* Thermometer (Lego Mindstorm NXT + HDC1008 Temperature & Humidity Sensor)
* hymidity sensor (HDC1008 Temperature & Humidity Sensor)
* air flow sensor (Lego Mindstorm NTX).

We will also take into account daytime with a light sensor

All the above sensors use I²C communication protocol.

Une image contenant horloge, objet, chose

Description générée avec un niveau de confiance élevé

Figure : Scheme of measurement group

## Sensors implementation:

In order to run the sensors and collect their values we will use an Arduino. Other microcontroller could also work such as PIC microcontroller or Raspberry board.

PICs can be cheaper but don’t have a lot of pins wich may be an issue if other sensors are needed in order to implement other energy optimizations.

Raspberry are more expensive for features we won’t use in this project.

The Arduino we will use for the prototype is Arduino ATmega2560.

The Arduino µC will run a C++ class called Sensor (Sensor.h, Sensor.cpp).

The class Sensor has a *int digitalPin* value and a *int analogPin* value for the Arduino to read correct pins.

The *int cap* value allows the user to set a cap. When the function *read()* is called, if the read value is the above the cap, the *bool value* is directly set as true in order to express the fact that something was detected and the analog value from the sensor is stored into the *float* valiable *analogVa*l.

A sensor object can be instanciated with a default constructor or with a constructor asking for the analog pin, the digital pin, the cap.

The sensor used in the program are grouped in an array of sensors. This array is declared as a global variable and instantiated in the *setup()* function where all the sensors are declared, instantiated and then added to the array.

## Tests

For the software, constructors, getters and setters have been tested and are used in the Arduino code in order to instantiate two sensors.

The function *read()* of the *Sensor* class have not been tested with real sensors as no sensor have been implemented for the hardware part.

The sensor class have mostly been used to simulate values for the network tests.

# III - Network

Now that the basic hardware have been coded, we want to send this values to the App.

For this prototype, we are working in the context of one single room but we want to keep in mind that the final purpose is to extend the concept to a whole building. Then, we want to take in account how to transfer the data through one side of the building to an other. In this case, wires are not an option:

* The Android app will run on Android mobile devices. It would reduce a lot the mobility of the user.
* The infrastructure would be to big: integration of the project to a real building would not be easy.

## Wireless technology selection

Several wireless technologies exists and could be used to send the datas to the app.

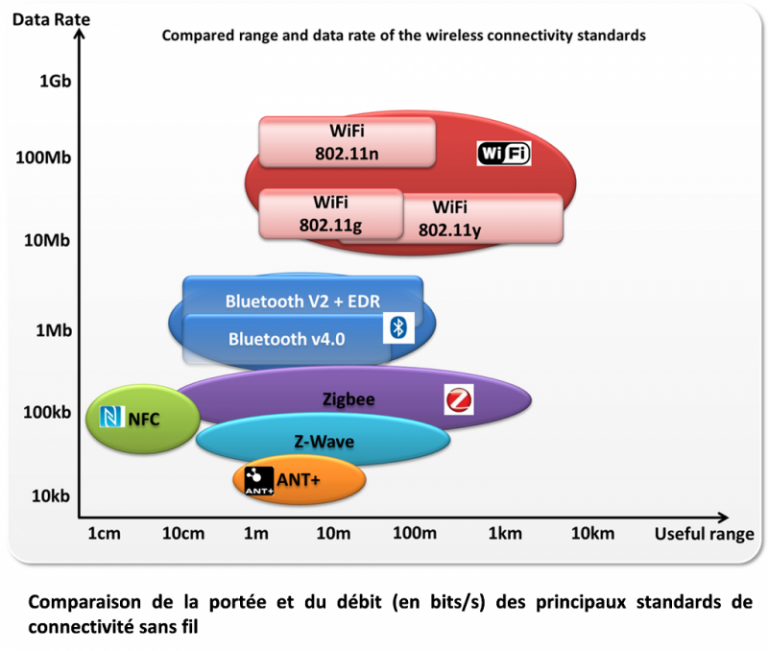
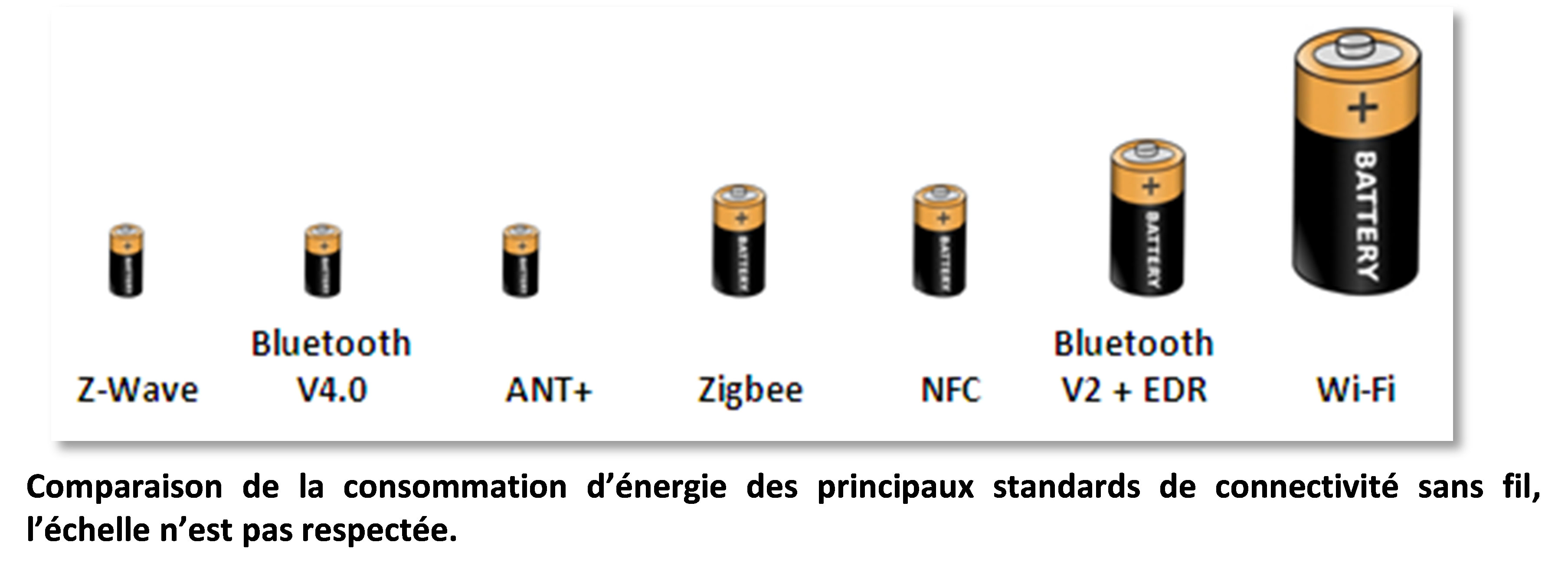


Figure 4: Comparison between wireless communication technologies battery needs.

Figure 34: Comparison between wireless communication technologies data rate and useful range.

In our case, measuring temperature, air flow and humidity don’t require a very high data rate. Yet, it would be ideal if the range could be long enough to reach any point in the smart building.

Both the Bluetooth and Zigbee wireless technology easily match the project range needs to cover a big room or a big hall. Yet, the Bluetooth is easier to implement and to use with the mobile devices on which the Android app will run : the zigbee technology is not always available on Android device. Consequently the Bluetooth is a better choice in our case.

## Communication protocol

In order to send correctly the data we had to think of a communication protocol to send the different informations across the Bluetooth network.

As we don’t know what kind of data can be sent in this Bluetooth network, we add to take into account different basics data types such as byte, int, float, char and bool. All these data types also have different length when converted into bytes that may result into wrong query reading and so wrong.

The protocol is organized as follow:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Query type | Number of values | Type of 1st value | 1st value | Type of 2nd value | 2nd  value | … |

Query type can be:

* 0 for an empty query that doesn’t expect any reply. Its purpose is for testing the connection.
* 1 to ask the liste of sensors. The expected answer is the number of sensors and for each of them its address and value.
* 2 to ask for the value of one specific sensor identified by address. The expected answer is its value.

The data type is encoded as follow:

* byte = 1
* int = 2
* char = 3
* float = 4
* boolean = 5

## Network of sensors

We have already conceived a small network of sensors connected with an Arduino and communicating with I²C protocol. Now, the data we have collected should be sent by Bluetooth to the App.

We will add to this network a new component in order to allow the Arduino to communicate directly with the Android app: the Bluetooth chipset RN-42 from Sparkfun.

This chipset has features such as a command mode that allows several actions such as rebooting the chipset independently from the Arduino, an implemented server protocol and library…

The Bluetooth chipset communication pins are pluged on Arduino on TX-1 and RX-1 (Chipset TX goes to Arduino’s RX and chipset RX goes to Arduino’s TX)

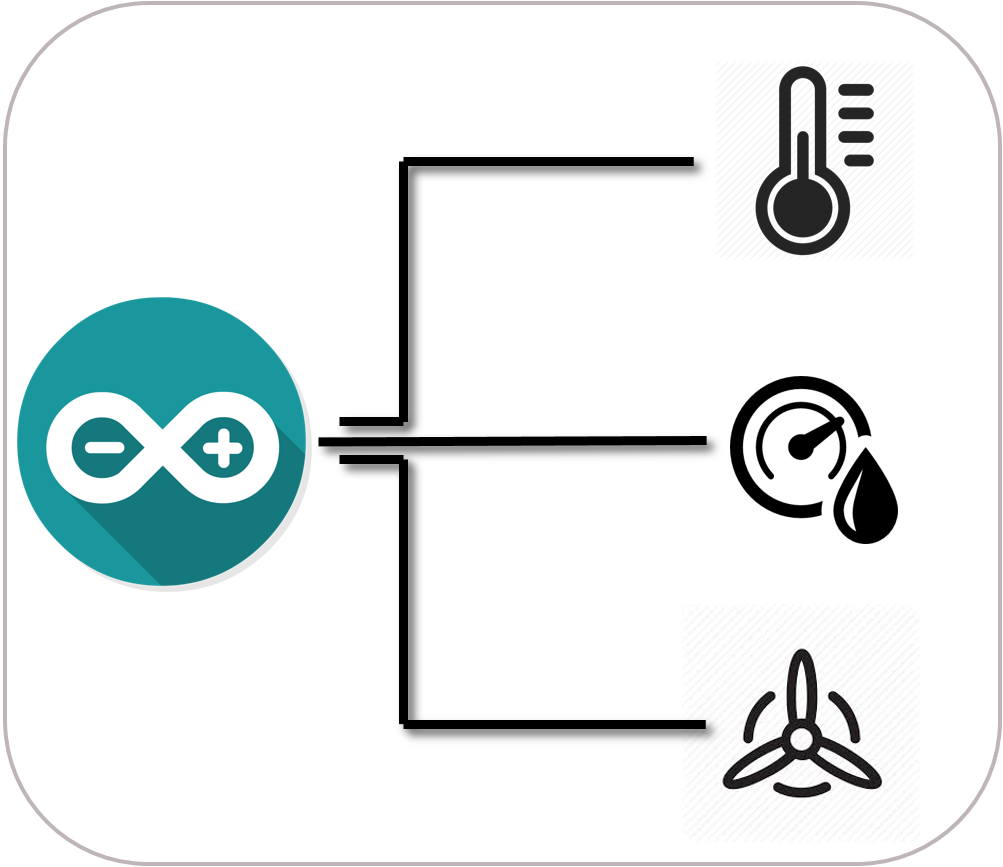
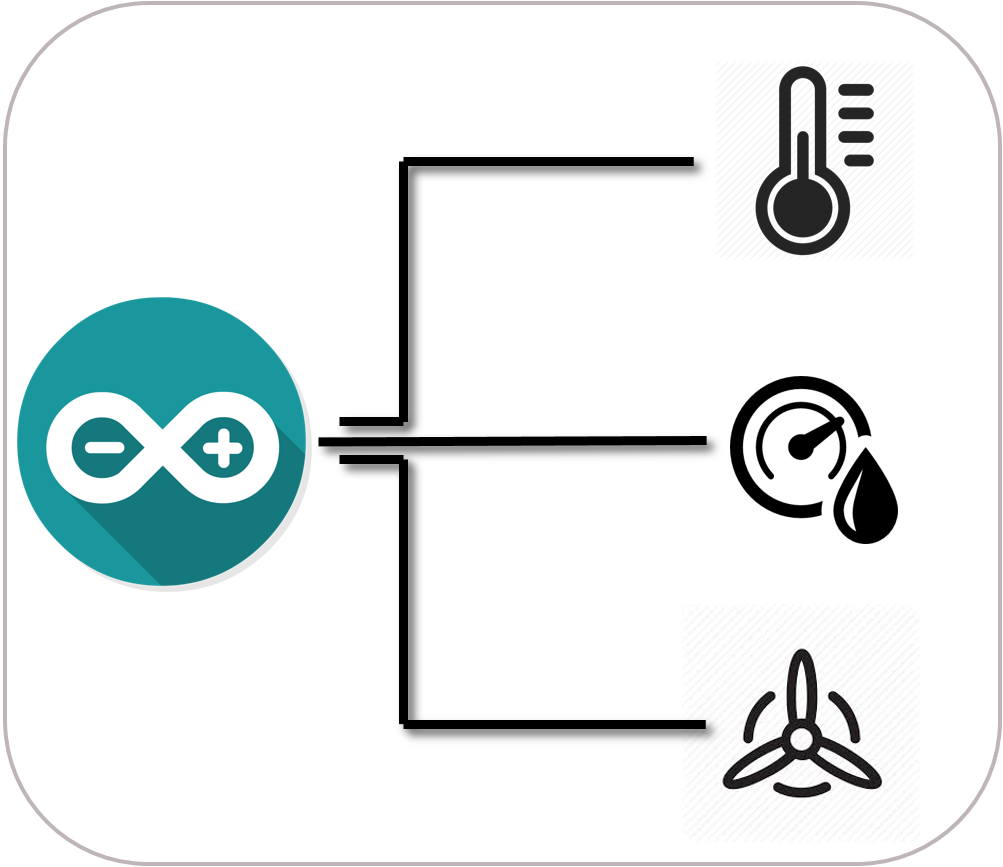
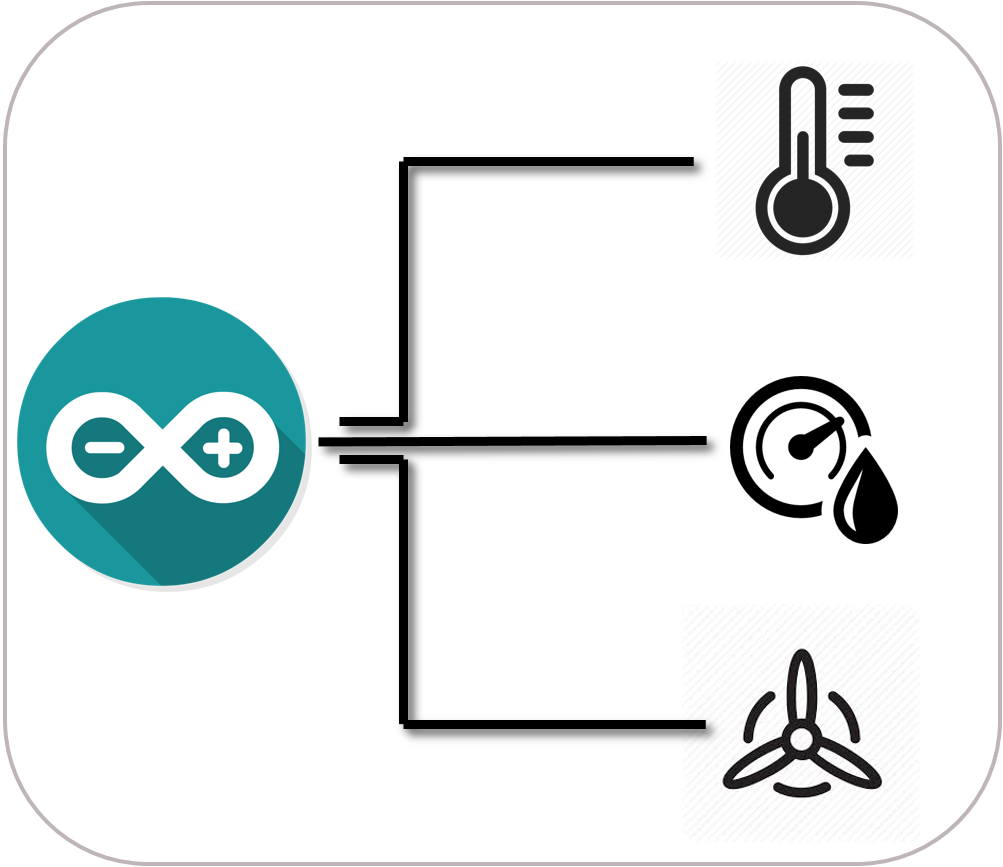


Figure 5: Scheme of the sensor network

The Bluetooth port has to be defined in the code as a global constant: Serial1 in our case.

## The Bluetooth network is implemented as follow:

In the *setup()* function The Bluetooth communication is started with baud rate defined to 115200.

In *loop()*, the function *handleQuery()* is called. This function reads the query and call the appropriate function(s) depending on the query.

Several other functions have been added to the Arduino code in the main function in order to use the command mode of the chipset (cf. RN-42 manual).

The function StartCommandMode() activates the commande mode.

The function EndCommandMode() desactivates the commande mode.

The function rebootBT() forces the chipset to reboot.

## Tests

The chipset RN-42 have been tested after being added to the network with the functions CommunicationTest. This function reads the Bluetooth entry and displays it in the standard Serial port. It also reads the standard Serial entry in order to send it on the Bluetooth network.

The software Putty was run on a computer with Bluetooth connection in order to send information for the function to read and to verify the values sent by the function.

# IV – Software:

## App design:

We will detail in this part how the Android app was conceived and what was planned to be implemented.

The Android app aims to connect to the different Arduinos via Bluetooth and to display on the screen the measured temperature humidity and air flow.

The priority is to have a synthetic view of the different Arduinos and sensors the Android device is linked with. In this condition, organizing the datas in a list seem to be the most appropriate organization.

Ideas for displaying the datas:

1. Organizing the datas in a list of Arduinos which, when clicked, open a new view with the linked sensors.

|  |  |
| --- | --- |
| + | - |
| Allows to display a bigger variety of informations and could allow to change the names of sensors or the details of the analysis results (update frequency or delay for an average value) | Does not provide an easy global view on all the datas. |

1. Organizing the datas in a list of Arduinos with a submenu of the linked sensors.

|  |  |
| --- | --- |
| + | - |
| Provide a global view on all the datas and the network organization. | Can be complicated to circulate between the different datas because of congested informations on screen. |

1. Organizing the data in several list of sensors with a mention of the Arduino. Each list groups sensors under categories (temperature, humidity, air flow) and correspond to one view.

|  |  |
| --- | --- |
| + | - |
| Allow the user to sort the data around one specific topic. | Does not provide an easy global view on all the datas. |

All these organizations have different goals but could make the data easier to read under different circonstances. Therefor, all of these configurations could be implemented as compatible with each other:

The second organization could be a default global organization of the data as a main global view.

A button in this default global view allow the user to swith from this network mapping organization to a topic organization similar to the third view (and vice- versa). Each of the data list should correspond to one type of data and be organized a one view. Switching between the lists could be done by sliding the view from side to side or with tabs.

When a sensor is clicked a view similar to the first organization should be displayed. From this view, changing parameters and customization should be possible.

In order to add new datas to the default list, we need to create a Bluetooth connection between the Android device and the Arduino device

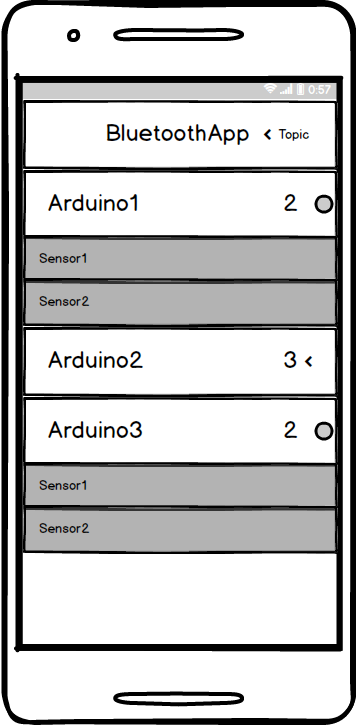
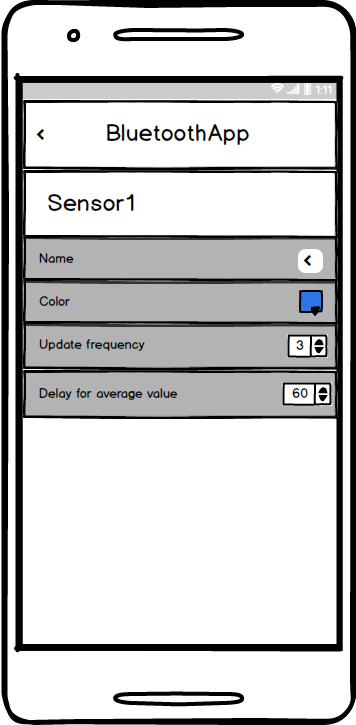
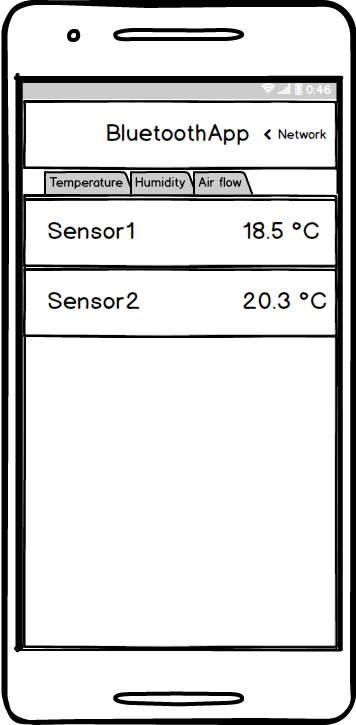
Handling the Bluetooth connection requires a new view in order to display the scan result and the paired devices. . Accessing this Bluetooth activity from the main activity will be done by clicking a Bluetooth button.

The view will be organized as follow:

The paired devices (the Bluetooth devices that have already been connected) as a first list of device. The already paired Arduinos the user willneed to add to the displayed datas will most likely already be used and so displaying them in first place would make the app easier to use.

* A button allow the user to start scanning the network for new devices. These new devices will be displayed under this button as a list.
* The Bluetooth devices will display their name, address, and a button to select this device to add to the main activity displayed datas.
* When at least one device is selected a button appears above the paired device list. When this button is clicked the main activity starts again and the selected Arduinos are displayed in the Arduino list and the submenu of linked sensor is available.

Figure 6: Scheme for the main menu



Une image contenant capture d’écran, ciel

Description générée avec un niveau de confiance élevé

Figure : class diagram (cf. App developpement)

## App developpement:

### Classes:

The app has several classes specificly designed in order to model the different elements used in this project:

The Sensor class: it echoes to the Arduino sensor class their content are similar.

The AccessPoint: it is equivalent to the sensor network. It contains a BluetoothDevice object that stands for the RN-42 chipset, and a list of sensors. This class also contains function in order to implement the Bluetooth communication. We can list the followings functions:

* CreateSocket() creates a specific socket to this device in order to star connection.
* PairDevice() handle the pairing process which is necessary in order to exchange data via Bluetooth.
* UnpairDevice()
* ConnectDevice() starts the connection with the bluetooth Chipset
* DisconnectDevice() breaks the connection with the device.

Other classes implement the network part, such as BluetoothCommunication, ArduinoQuery and Arduino Data.

BluetoothCommunication initiates the queries send it and reads the Bluetooth network for the answer and analyses it in function of the routine function (QueryEmpty(), QueryContent, QuerySensor()).

The BluetoothReceiver class handles the Bluetooth reception. It is used in order to scan the network with the discovery process of the BluetoothConnectionActivity that calls its onReceive() function whan a new device is found.

The ViewHolders ans Adapters are classes that are used in order to display a list of data in a ListView. They all follow the basic implementation of ViewHolders and Adapters exept for SensorAdapter() we will mention later.

The app is organized in two activities: MainActivity and BluetoothConnection Activity.

### MainActivity:

MainActivity goal is to display the connected Arduinos and the linked sensors. At the first lauch of MainActivity, a toast message proposes the user to activate the Bluetooth if it is not activated. Using the Bluetooth functionalities without Bluetooth will cause a crash of the activity.

The activity has one ArrayList of AccessPoints that are displayed in the ListView SensorListView. The ArrayList is generated with the BluetoothConnectionActivity that returns a selection of AccessPoints with an Intent. This Intent is handled with the function onNewIntent() and receiveSelection().

For the display of AccessPoints are different in the BluetoothConnectionActivity, the AccessPoints have a specific ViewHolder called AccessPointMainViewHolder in this activity.

The SensorListView is actually a ListView of AccessPoints. This ListView elements are updated with an optimized internal Java asynchronous process that calls the SensorAdapter method getView() to create the ListView content.

When generated with SensorAdapter, the adapter connects to the AccessPoint and queries for the list of its sensors. Based on the answer, the list of sensor is displayed dynamically under its AccessPoint with the function GetSensorView(). The ViewHolder used for sensors is called SensorViewHolder.

Consequently, this process of query works like an update.

On every display of access point, a button allows the user to show or hide the sensor linked to the clicked access point.

### BluetoothConnectionActivity:

BluetoothConnectionActivity goal is to display the paired divices, scan the network for new Bluetooth devices and display them on screen.

The Bluetooth devices are organized into two lists. One for the paired devices and another one for the new devices.

The list of paired device is initialized at the Activity creation with the methode setPairedDevices().

The list of new devices is created with the discovery() process od BluetoothAdapter. Each time the discovery method finds a new result, it calls the onReceive() method of the Bluetooth receiver with an Intent. If the Intent value is ACTION\_FOUND we add the device to the newAccessPointList();

The displayed devices can be selected from the BluetoothConnectionActivity screen to display the datas on the main screen. When selected by clicking on the “+” button, the devices are added to a list to be sent to the MainActivity with an intent. The device can be deselected by clicking the “-“ button that replaces the “+” button. As the access point object is sent by intent to the MainActivity, AccessPoint and Sensor are classes that have to be implemented as parcelable.

## Tests

The last part to be added in this version of the app is the network part including modifications to the AccessPoint class in createSocket() and to the BluetoothCommunication class in order to add the ArduinoCommunication and ArduinoData systems. Therefor this network version of AccessPoint BluetoothCommunication, ArduinoCommunication and ArduinoData haven’t been tested.

All the ListViews have been tested and work properly but the network part can have unexpected consequences on their display.

The pairing process and the discovery process work properly.

# III - Next steps of project:

## Project results:

Arduino and Android are able to communicate.

Arduino returns linked sensors informations to Android support when asked.

C++ Class to handle sensors on Arduino.

The first (main) screen of the app dynamic displays access points (Arduinos) selected to be displayed in a second screen that can be accessed from main screen via a Bluetooth button. This display of data is similar to the network organization mentioned in this report.

In this Bluetooth section are displayed the list of paired Bluetooth devices that have already been used. Under this list a button starts the discovery of new devices.

The displayed devices have a button in order to add/remove theme to a list of selected devices. When something is in this list a button appears above the list of paired sensors. This button sends the selected devices to be displayed to the main screen. Several devices can be sent at one time.

During the initialization of these new devices, the app initiate a Bluetooth connection in order to collect necessary informations from the Arduino.

When the main activity restarts, the displayed access points have a small button that allows the submenu of sensor to be showed or hidden. The value of sensors is updated each time the listview is updated.

## Improvements:

Key changes:

* The Arduino developpement hasn’t been tested yet with real sensors to take values from.
* The Arduino code should be modified in order to read properly the queries from ArduinoQuery class, that has recently been added to the app.
* There are little security in the app about the Bluetooth : conditions must be added to check if the Bluetooth is still activated each time the app uses Bluetooth.
* The Arduino should be able to store the measured datas until the app asks for it.
* The datas collected by the app should be stored in a file.
* Some mathematical models should be implemented to calculate statistics ans analysis about thermic comfort and used energy.

The rest of the suggestions mainly gives way to make the app more user-friendly:

* The other organizations of datas could be implemented.
* A spinner could be added to show if the discovery process is still active. This process is asynchronous and the app still usable while this process is on and nothing warns the user that the process has finished.
* In the paired devices there is indicators to say if the device is reachable or can be connected.
* Storing the list of sensors displayed during the last use of the app could be a usefull tool for the user.

## Network extention to a Building

We mentioned above that the Bluetooth matches the needs for the prototype. None the less, it probably won’t be enough for a whole building because of obstacles like walls that lower the range.

Data should be collected with small mesurement units centralizing all the measurement from one room. These small measurement units could be easily installed in several rooms to take measurements in all the smart building with always the same format and the same transfert system, the same cost…

In order to be installed all over a building these sensor groups need to be as discrete as possible and to integrate seemlessly into the environnement: big sensors in a room can give people the feeling they are under surveillance, which is something to avoide.

Other architectures could be better options to match the bigger proportions of a building. For example, it would be possible to collect all the data on a serveur. This server could be local or part of a cloud system. It is important to mention that paying for a cloud service brings security to the data ( duplication on several physical servers, permanente access, encryption… ) and don’t imply maintenance. Moreover, the cost of a cloud server depends directly on the services and on the quantity of the informations that are communicated.

In order to extend the concept to a whole building, measurement units has to be installed in at least the main rooms of the building.

As we decided to use a Bluetooth technology, it is possible to connect the Android device running the app to severals Arduino terminal to collecte and send the datas. Yet, the Bluetooth device range is nerfed because of walls and obstacles. It might work for a small building the size of a house but such an architecture propably can not work with a complete building with several floors. In this case, the datas has to be sent to a dedicated servers (that can be physical or in the cloud): an architecture with a server is mandatory in case of a big building to equip.

Once all the devices have sent their informations on a server there are two possibilities:

* An administrator user loads all the informations from all the devices on his Android device and generate a global report with all the energetic informations to help him optimize the consumption of the energy.
* The server is programmed in order to analyse the collected data and send the report to the Android device when asked.

Moreover, the app could also be a way to handle all the sensor of the building wherever the user is, with a server architecture. It also allows a very precise analysis of the data at any moment.

# IV - Sources:

1: - <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3545575/>

- <http://www.ecoconso.be/fr/Le-confort-thermique-encore-mieux>

- [https://www.energieplus-lesite.be/index.php?id=10250#Temperature](https://www.energieplus-lesite.be/index.php?id=10250)

- [https://www.energieplus-lesite.be/index.php?id=11016#c17973](https://www.energieplus-lesite.be/index.php?id=11016)

Figure : <http://reseaux-chaleur.cerema.fr/consommation-denergie-dans-les-batiments-chiffres-cles-2013>

Figure 3: <http://www.wi6labs.com/2016/03/16/quelle-technologie-radio-pour-les-objets-connectes-premiere-partie/>

Figure 4: <http://www.wi6labs.com/2016/03/16/quelle-technologie-radio-pour-les-objets-connectes-deuxieme-partie/>

# V - Javadoc :

To access the doc, double click on the file below. Your explorer will open where the file is stored.

Unzip it with right click on the file in the explorer > Unzip file .

Double click on index file to open the doc.

