Systems Integration

3rd year • 1st semester • Academic year 2017/2018

Project

AirMonit – Air Quality Sensors' Monitoring Platform

What is the project about? The overall approach.

The **AirMonit** solution consists in the development of an integrated platform that uses a sensors' network which allows the quality monitoring of atmospheric air. There are a large number of parameters that can be used to monitor air quality, e.g. Ground-level Ozone, Particulate Matter, Carbon Monoxide, Lead, Sulphur Dioxide, Nitrogen Dioxide etc. Most of these parameters can be read from surrounding air by using sensors that are able to remotely transmit this information. Therefore, after measuring this parameters data can be transmitted over the internet.

For the purpose of this project only some data regarding air quality parameters will be monitored. Namely: Nitrogen Dioxide (NO2) / *Dioxido de Azoto*, Carbon Monoxide (CO) / *Monóxido de Carbono* and Ozone (O3) / *Ozono*.

Consider the above scenario where some sensor nodes, with a unique ID, are periodically reading the three parameters and sending these readings to a sensor network gateway/Hub. Further, several applications will be developed to visualize, monitor and analyse these data and 'guaranty' the air quality. If some of the parameters values are not in the recommended range, an alert will be triggered to notify the overall systems that the air is not fulfilling the recommended quality parameters defined in the AirMonit monitoring platform.

In order to simplify the project, there is no need to use sensor network hardware. Instead, it is replaced by a DLL that mimics the behaviour of the sensors nodes by pushing data directly to an application.

Summarizing, the aim of the AirMonit platform is to develop a solution that collects data from sensors. These sensors acquire data about Nitrogen Dioxide (NO2), Carbon Monoxide (CO) and Ozone (O3). Each sensor node represents a channel (NO2, CO or O3) for a specific location (city).

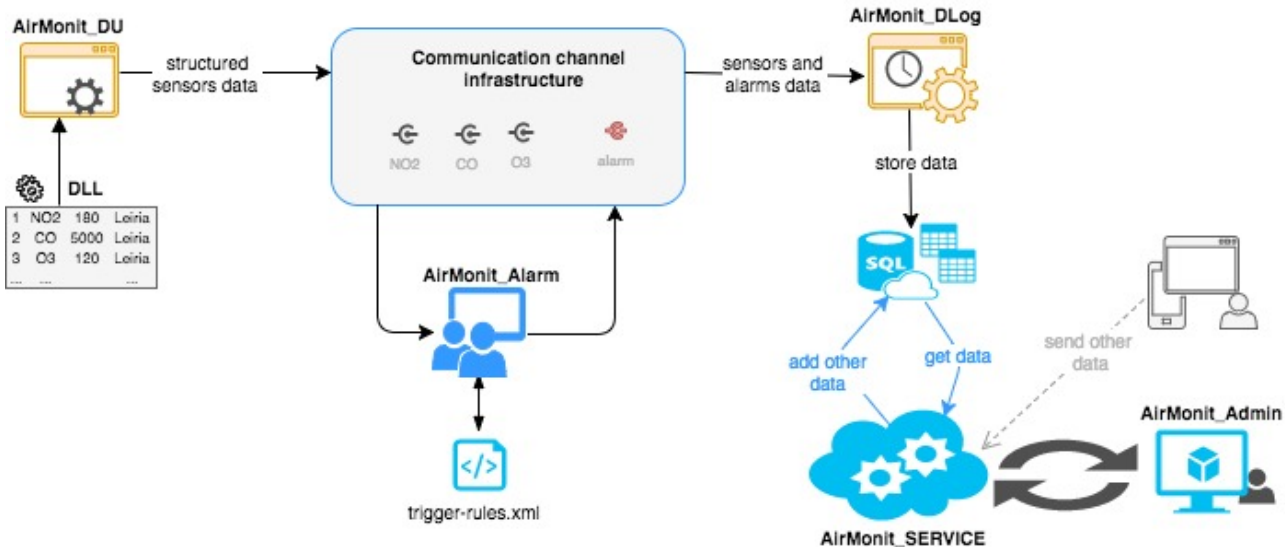

Figure 1 - Project components

The AirMonit solution has several micro applications, each one with a specific behaviour. Figure 1 represents the interaction of these applications and their components. All the applications are interconnected. The integration platform solution must be chosen and implemented considering that in a near future there could be dozens or hundreds of modules (different applications, e.g. mobile apps, web pages, etc.) sharing information. Only some applications are going to be implemented now.

AirMonit Data-Uploader (AirMonit_DU) and Communication channel infrastructure

The aim of the Data Uploader application (AirMonit_DU) is to acquire the data from the sensors and make it available to the remaining modules. As mentioned above, to simplify this task, it is the DLL that will **push** the data to this module. The **AirMonit-DU application** parses the data generated by DLL and implements the required behaviours to make the air parameters data available to some of the remaining applications of the solution. Essentially, it needs to parse the data pushed by the DLL (See note 1 to find out the way the DLL is integrated in the project) and make that data available in a communication channel to all the remaining applications in a well-defined format. A mechanism to implement the communication between the several systems must be considered. In Figure 1, this behaviour is represented by the **Communication channel infrastructure**.
Students need to decide the best approach to implement the Communication channel infrastructure, which is the basis of all data sharing between all the applications ecosystem. In addition, consider that the following applications must work on a distributed architecture.

Alarms Application (AirMonit_Alarm)

An application that is able to automatically trigger alarms/alerts if a specific air parameter is out of a safe range must be implemented. This application will be called AirMonit_Alarm. The generated alarms created by the AirMonit_Alarm must also be sent to the communication channel infrastructure. Therefore, by using the communication channel infrastructure data can be shared with the remaining modules.
The AirMonit_Alarm application requires a XML file (*trigger-rules.xml)* with the pre-defined rules/conditions to trigger alarms. Each parameter (Nitrogen Dioxide, Carbon Monoxide and Ozone) can have several trigger rules/conditions defined in the XML file. Therefore, students must create the XML file to be used by this application, considering the following conditions:

- A parameter can have more than one alarm conditions;
- For now, consider only the conditions equal, less than, greater than and between (=, <, > and between);
- A condition can be active or inactive. This way at a specific time, one of the parameters can be ignored by the applications alarms;
- The AirMonit_Alarm application also allows the user to change the alarms' conditions persisted in the XML file.

The application data flow is the following:

- Each parameter data sent to the communication channel infrastructure (by the previous module) needs to be collected by this Alarms application, and then the application compares its values with the alarms rules/conditions store in the XML file (*trigger-rules.xml*) to see if it needs to trigger an alarm to notify the AirMonit platform of the poor air quality. Not all parameters need to be monitored by the alarms application. The user can choose to monitor only some parameters.
- The alarms must have information about the parameter they refer to, parameter value and data-time of the sensor reading, city to each sensor data refers to, alarm condition (why it was triggered), etc.
- After triggering the alarm its data must be send to the communication channel infrastructure.

The alarms application needs to have and on/off option to allow the users to temporarily inactivate the alarms application if necessary, e.g. when editing the XML file to change the trigger conditions.

AirMonit Data-Logger (AirMonit-DLog)

In this module students need to implement an application which tracks all the data that is transmitted via the communication channel infrastructure (parameters data and the generated alarms). All data available at the communication infrastructure will be automatically stored by the Data Logger application (AirMonit_DLog). Thus, this micro application acts like a system log database, where all the data transmitted along the infrastructure is stored. Additionally, the log database must also be prepared to store 'qualitative data' that will be provided by another service (ex. Information about the occurrence of a fire on a specific city, etc. which can influence the air quality also).
The database script (.sql file) representing the database structure must be specified and delivered by the students.

AirMonit Service and AirMonit Admin Application

The AirMonit Service and Admin application modules allow the AirMonit platform users to visualize statistical information about the air quality parameters being monitored. A Web Service (AirMonit_Service) must be implemented to make available an endpoint that will control the communication with the data log repository and a graphic client application (AirMonit_Admin) which allows the users to monitor daily and weekly statistics for every air quality parameter being monitored.

The web service (**AirMonit_Service**) provides data from the database files maintained by the AirMonit-DLog application. This information is summarized by the web service, making it available to client applications. The data refers to the parameter air value (Nitrogen Dioxide, Carbon Monoxide and Ozone) and the alarms triggered by the platform. Although other features may be considered, the mandatory services are the following:

1. Get hourly summarized information (min, max and average values) for a parameter on a specific day, in a city specified by the user or all cities;
2. Get daily summarized information (min, max and average values) for a parameter on a specific threshold (between two dates/time) in a city specified by the user or all cities;
3. Get daily alarms information by city;

In order to enrich data quality, the AirMonitor_Service must also be prepared to store uncommon events ('qualitative data') as well as additional quantitative data like, e.g. temperature values (among others), coming from other applications which may, in the future, be integrated with this project. As an example, the temperature is the measured temperature surrounding the user, while uncommon events could be, e.g. industrial gas leak, fire, tornado, etc. This ad-hoc information could be useful to explain (or not) temporary high values for specific pollutants (example: wild fire and Carbon monoxide raising).

4. Add additional air quality data provided by a user representing uncommon events and temperature (e.g.: in Leiria city that was a fire today at 12PM, in Lisbon, a gas leak was detected at 15PM, 23º in Coimbra at 2:23PM, etc.).

A user is only able to upload this information when he/she identifies herself (no need for registered users). Hence, this kind of data records follow, at least, the following structure:
User_ID, Uncommon_event_description, Temperature_in_celcius_degrees, City_name

A Graphic User Interface (GUI) application, called **AirMonit_Admin**, needs to be developed to show in a friendly user manner the data returned by the previous web service. Therefore, the following features are mandatory:

- Allows the user to choose if the data will be shown to a specific city or all cities;
- Show information about raised alarms;
- See alarms information that have been triggered between two dates;
- Show uncommon events data provided by the users;
- Graphically see daily by hour statistics of each parameter per day (selected by the user).

*** The end ***

Guidelines

Groups must be formed by 4 students, which can be from different practical classes. The name and number of the group members must be submitted in the courses page (http://ead.ipleiria.pt - Moodle) until the end of October.
The project delivery must be performed until the date published in the official assessment calendar. It has a mandatory project presentation (and oral discussion) that will occur in a later date, also published in the course assessment calendar.

Besides the source code of the entire solution, it is also mandatory to deliver a **written report** explaining all the decisions made regarding the project implementation and **clearly** identifying what each team member has implemented (which application has helped to develop). Summarising, the written report needs to include:

- All the implementation decisions, explained and justified;
- All the necessary credentials (login + passwords if applicable) and configurations required to test the modules of the project;
- The necessary steps to test the whole project. It is recommended that students test all the given instructions to guarantee that the project is fully operational.

A template for the report will be available in the course web page.

Project Delivery

Students must guaranty that all the following material is included in the CD/DVD/pen when delivering the project, and with the following organization structure:

- Text file (**identification.txt**) with all the information about the team members (number, name and e-mail) and course name, etc.
- Folder **Project**: all the source code of each module must be included in this folder. The source code of each module should be in a different folder to clearly identify each project. For each project, you must include: source files and other resource (files, executables, dll's, etc.)
- Folder **Report**: the written report (DOC or PDF) explaining all the decision that were took by the team members to implement the project. Don't forget that in the end of the report it is mandatory to identify which features has each team member implemented.
- Folder **Data**: SQL server database files (.mdf and .ldf). The database must have data, in order to be able to test the project. Also, database script must be included (.sql file).
- Folder **Other**: other files required by the project that were used by the team members but were not included in the previous folders.

Absolute paths to folders and files should be avoided.

Project Assessment

The project evaluation criteria are as follows:

Criterion	% [0-100]	[0-20]
AirMonit Data-Uploader + Communication channel	10	2
AirMonit Alarm application	17.5	3.5
AirMonit Data-Logger	17.5	3.5
AirMonit Service	20	4
AirMonit Admin application	20	4
Written report	15	3

Any questions about the project should be addressed to the professor.

Other information

The project of the *SensorNodeDll* can be downloaded from Moodle and be included in your solution. After including it in the solution, you can:

Instantiate the DLL like this:

```
AirSensorNodeDll.AirSensorNodeDlldll = new AirSensorNodeDll.AirSensorNodeDll();
```

Initialize the DLL (data generation) like this:

```
dll.Initialize(AnyMethodName, delay_in_ms);
```

The parameter is a reference for an application method that will be called by DLL when new data is available. That method must follow the following signature:

```
public void AnyMethodName(string str)
```

where *str* is the message string generated by the sensor/DLL. Remember that DLL uses the push approach.

Stop DLL (stop the generation of sensor data) like this:

```
dll.Stop();
```