POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

www.dei.estg.ipleiria.pt

Systems Integration

Project

SmartH2O - Water Quality Sensor Platform

What is the project about? The overall approach.

The SmartH20 solution consists in the development of an integrated platform, that uses a sensors network to allow the quality monitoring of drinking water sources. There are a large number of parameters that can be used to monitor water quality, e.g. pH, dissolve oxygen, calcium, temperature, nitrate, chloride, ammonia, etc. Most of these parameters can be read from water sources by using sensors that are able to transmit this information remotely. Therefore, after measuring this parameters data can be transmitted over internet.

In this project context only some data regarding water quality parameters will be monitored. They are: pH, ammonia (NH_3) and chlorine (Cl_2). Consider the above scenario where some sensor nodes, with a unique ID are periodically reading the three parameters from a specific water source and sending readings to a sensor network gateway or Hub. Further, several applications will be developed to visualize, monitor and analyse this data and guaranty the water quality. If some of the parameters value are not in the recommended range, an alert will be triggered to notify the overall systems that the water is not fulfilling the recommended quality parameters defined in the SmartH2O monitoring platform.

In order to simplify the project, there is no need for using sensor network hardware. It is going to be replaced by a DLL that mimics the behaviour of the sensors nodes by pushing data directly to an application.

Summarizing, the aim of the SmartH2O platform is to develop a solution that collects data from sensors. These sensors acquire data about pH, ammonia (NH_3) and chlorine (Cl_2) of a water sample. Each sensor node represents a channel (PH, NH3 or CI2), therefore it has information about the channel pH, ammonia and chlorine.

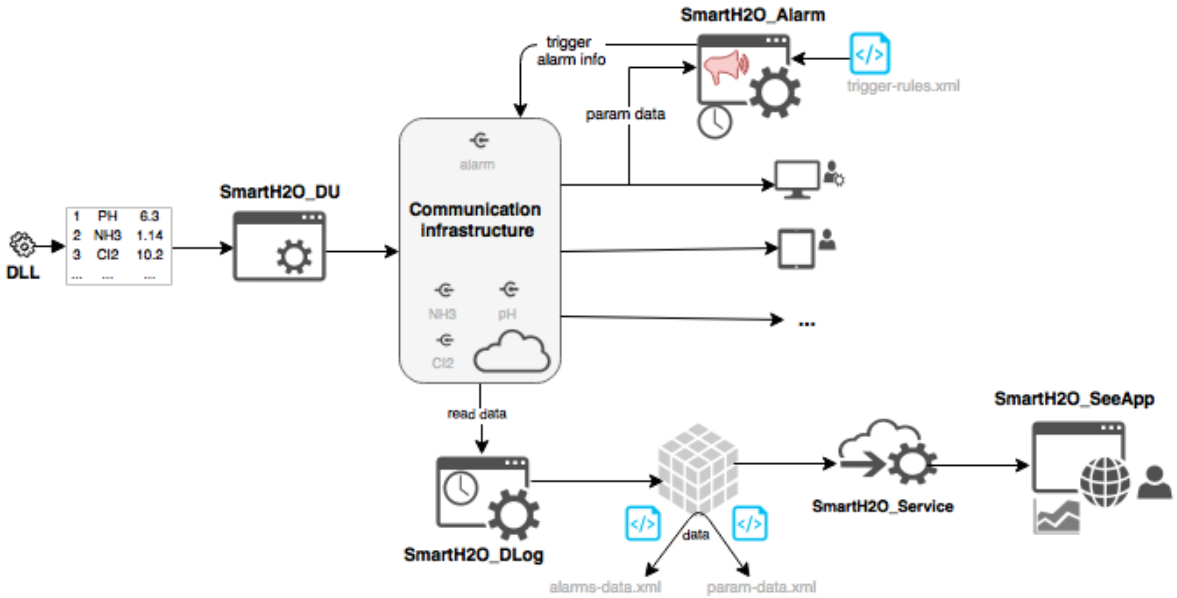

Figure 1 - Project components

The SmartH2O solution has several micro applications, each one with a specific behaviour. Figure 1 presents the several applications and their components. All the applications are interconnected. The integration platform solution must be chosen and implemented having in mind 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 implement now.

SmartH2O Data-Uploader (SmartH2O_DU) and Communication channel infrastructure

The aim of the Data Uploader application (SmartH2O_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 **SmartH2O-DU application** parses the data generated by DLL and implements the required behaviours to make the water 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 formed format. A mechanism to implement the communication between the several systems must be considered. In Figure 1, this behaviour is represented by the **Communication infrastructure**.
Students need to decide the best approach to implement the Communication infrastructure, which is the basis of all data communication between all the applications ecosystem. In addition, consider that the following applications must work on a distributed architecture.

Alarms Application(SmartH2O_Alarm)

An application that is able to automatically trigger alarms/alerts if a specific water parameter is out of a safe range must be implemented. This application will be called SmartH2O_Alarm. The generated alarms created by the SmartH2O_Alarm must be sent to the communication infrastructure also. Therefore, by using the communication infrastructure data can be shared with the remaining modules.
The SmartH2O_Alarms application requires a XML file (*trigger-rules.xml)* with the pre-defined rules/conditions to trigger alarms. Each parameter (pH, ammonia and chlorine) 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, let's considered only the conditions equal, less than, greater than and between (=, <, > and between);
- A conditions can be active or inactive. This way at a specific time, one of the parameters can be ignored by the alarms applications.

The application data flow is the following:

- Each parameter data sent to the communication infrastructure (by the previous module) needs to be collected by the 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 SmartH2O platform of the poor water quality. This way the user can be notified and can take the appropriate procedures to avoid the consumptions of the "bad" water. 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, alarm condition (why it was triggered), etc.
- After triggering the alarm its data will be send to the communication 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.

SmartH2O Data-Logger (SmartH2O-DLog)

In this module students need to implement an application which tracks all the data that is transmitted via the communication infrastructure (parameters data and the generated alarms). All data available at the

communication infrastructure will be automatically store by the Data Logger application (SmartH2O_DLog) in two separated XML file: one to store the parameters data (*param-data.xml*) and other to store the information about the triggered alarms (*alarms-data.xml*). Thus, this micro application acts like a system log database, using XML files to store the data.

The XML structure of these two XML files must be specified by the students also.

SmartH2O Statistics-Visualizer

The SmartH2O Statistics-Visualizer module allows the SmartH2O platform users to visualize statistical information about the water quality parameters being monitored. The module implementation implies the development of a Web Service (SmartH2O_Service) and a graphic client application (SmartH2O_SeeApp) to allow the users to see daily and weekly statistics for every water quality parameter being monitored.

The web service (**SmartH2O_Service**) provides data from the two XML files maintained by the SmartH2O-DLog application. This information is summarized by the web service, making it available to client application. The data refers to the parameter water value (pH, ammonia and chlorine) and the alarms triggered by the platform. The mandatory services are the following, but other features may be considered:

- Get hourly summarized information (min, max and average values) for a parameter on a specific day;
- Get daily summarized information (min, max and average values) for a parameter on a specific threshold (between two dates);
- Get weekly summarized information (min, max and average values) for a parameter;
- Get daily alarms information;
- Get alarms information on a specific threshold (between two dates).

A Graphic User Interface (GUI) application, called **SmartH2O_SeeAPP**, 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:

- Show information about raised alarms;
- See alarms information that have been triggered between two dates;
- User can select to see the information about all water parameters, or only for some of them;
- Graphically see daily by hour statistics of each parameter;
- Graphically see weekly by day statistics of each parameter; and
- Other features can be implemented to improve users' iteration with the SmartH2O_SeeApp application and the overall platform data.

***

Guidelines

Groups must be formed by 3 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 is mandatory a project presentation (and oral discussion) that will occur in a later date, also published in the course assessment calendar.

Beside 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 help to develop). Summarising, the written report needs to include:

- All the implementation decision must be explained and justified;
- All the necessary credential (login + passwords if applicable) required to test the modules of the project must be presented;
- Must include the necessary steps to install/test the all project. It is recommended that student test all the given instruction to guaranty 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 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 include in this folder. The source code of each module should be in a different folder to clearly identify each project. For each project 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 **Bin**: folder with all the executables (exe files) of the project.
- Folder **Data**: database files. The database must have data, in order to be able to test the project.
- 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 will follow the next assessment weights.

Criterion	%
SmartH2O Data-Uploader + Communication channel	10
Alarms Application	17.5
SmartH2O Data-Logger	17.5
SmartH2O Statistics Visualizer (SmartH2O_Servcie)	20
SmartH2O Statistics Visualizer (SmartH2O_SeeApp)	20
Report	15

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 DLL like this:

```
SensorNodeDll.SensorNodeDlldll = new SensorNodeDll.SensorNodeDll();
```

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();
```