SYSTEMS INTEGRATION

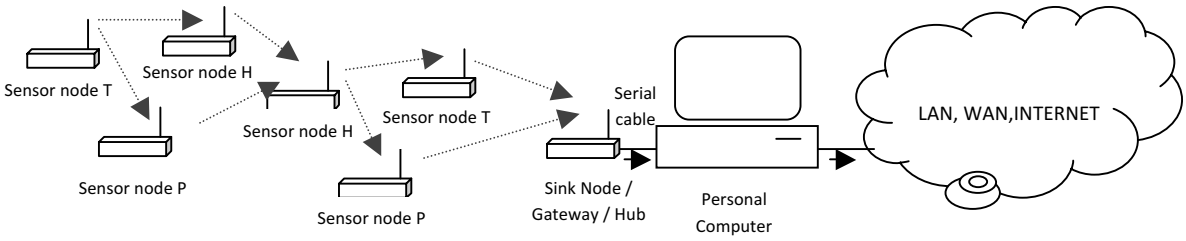
Computer Engineering - Academic Year 2015-2016

DATA INTEGRATION HUB

BACKGROUND

Today, the Internet of Things (IoT) is a reality. With the IoT some buzzwords came up such as Home Gateway, Home Hub, Big Data, Smart Homes, etc. Although the IoT represents the ability for things to communicate over Internet, there are other concepts such as data acquisition or sensor networks that pay an important role in the Internet of Things thematic.

Consider the following scenario where some sensor nodes (e.g. WaspMotes, each one with a unique ID) are periodically reading temperature, humidity and barometric pressure in a specific area and sending readings hop-by-hop to the sensor network gateway or Hub.



Suppose the sensor nodes use the following data message format:

Node ID	Channel	Data
Sender unique Node ID Example: 1	Data type Example: 'H' for humidity, 'T' for temperatue, etc	Value for the channel Example: 1 'T' **23.50**

Once data reaches the Hub, it could be stored in the local computer or in the cloud and then extract valuable information from that data for different purposes (e.g. raising alarms).

THE PROJECT

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 Sink Node by periodically pushing data directly to the desktop application.

The aim of the project is the development of a solution that collects data from sensors. In this project the data gathered by the sensors are from room division inside a house. The sensors acquire data about the Temperature (T), Humidity (H) and Barometric Pressure (P) of that room division. Each sensor node represents a channel (T, H or P), therefore it has information about the channel Temperature, Humidity and Pressure.

The whole solution to be implemented has several modules, each one identified in Figure 1. It is required to developed the **Sensor Data Controller** and the *Communication HUB*, **Alarming System**, **DBPersistence** and the **Service API** which provides the information required by the GUI client (*Service Client application*) where the final user can monitor the information gathered by the sensors.

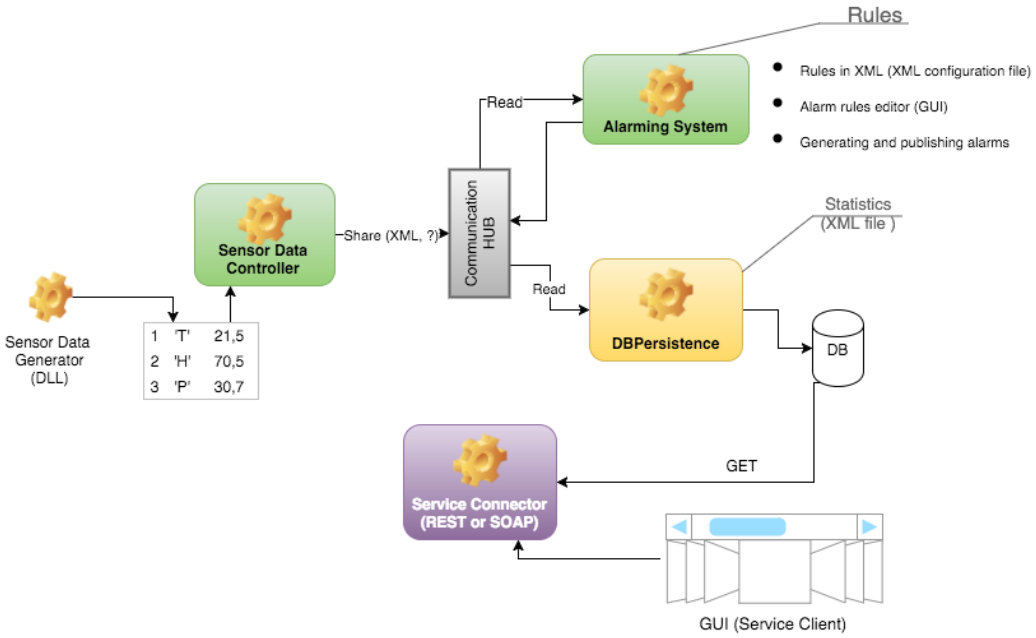


FIGURE 1 – COMPONENTS OF THE PROJECT

The proposed solution has several modules all interconnected. Therefore, it is recommended that for each group, a module should be assigned to each student.

The integration solution must be chose and implemented having in mind that in a near future there could be dozens or hundreds of modules sharing information and not only the *Sensor Data Controller*, *Alarming System* and *DBPersistence*.

MODULE 1 – SENSOR DATA CONTROLLER AND ALARMING MODULE

The aim of *Sensor Data Controller (SDC)* is to collect the data sent by the sensors and make it available to the remaining modules. In reality, it is the DLL that will push data into this module. The SDC module parses the data generated by DLL and implements the required behaviours to make them available to the remaining modules of the solution. Basically, 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 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 HUB.

The **Alarming** module must have a XML file where are defined the rules to generate alarms for the channels based on conditions that can be changed by the user at any time. This module must generate alerts automatically based on the conditions pre-defined in the XML file (for now consider only the conditions equal, less than and greater than). The alarms must also be shared among all modules. In this case there is no module to consume them but in a near future there will be. The *Alarming* module uses the information already parsed by SDC module to generate the alarms.

The Alarming module needs to have a GUI that allows the user to easily change the rules stored in the XML file.

MODULE 2 – DBPERSISTENCE

This module is responsible for storing the data in the database previously parsed by the SDC. Therefore, the *DBPersistence* module communicates with the Communication HUB to get the data previously sent by DLL (Module 1 – Sensor Data Controller).

The database structure (tables, relations, etc.) is totally up to you, but performance issues must be considered.

Additionally, this module needs to daily update statistical data about the channels, e.g. average values of each channel (parameters) per day. This statistical information must be daily updated and stored in a XML file.

MODULE 3 – SERVICE CONNECTOR

The *Service Connector* module represents a service (SOAP or REST) that connects to the database implemented in the previous module (*DBPersistence*) and returns the information about the channels that has been stored so far by the system.

The mandatory services are the following, but other features may be considered:

- Get the data for each channel in a specific time/date range;
- Average values for a specific day time/date range;

A Graphic User Interface (GUI) must be developed to show in a friendly user manner the data returned by the Service Connector. This application must have a feature that allows the user to export the information to an Excel file. The use of graphics (plot) to facilitate the analysis of the statistical information is recommended.

GUIDELINES

Groups must be formed by 3 students that can be from different practical classes. The name and number of the group members must be submitted in the courses page (moodle) until the end of November.

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 decision made in the implementation of the project and **clearly** identifying what each team member has implemented (module by module).

PROJECT DELIVERY

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

- 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 what module and features each team member has implemented.
  - All the implementation decision must be explained and justified;
  - All the necessary credential (login + passwords) 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 full operational;
  - A template for the report will be available in the course web page.
- 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	%
Module 1 – Sensor Data Controller + HUB	20
Module 1 – Alarming System	20
Module 2 – DBPersistence	20
Module 3 – Service Connector and GUI	20
Report	20

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

Initialize the DLL (data generation) like this:

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

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();
```