SafeHouse

Luben Ivanchev

MSc Eng. Vision, Graphics and Interactive Systems
Master Thesis Project - 10th Semester 2009/2010

Department of Electronic Systems
Niels Jernes Vej 12, 9220 Aalborg
Aalborg University

3rd June 2010

DEPARTMENT OF ELECTRONIC SYSTEMS

VISION, GRAPHICS AND INTERACTIVE SYSTEMS

TITLE: SafeHouse
THEME: Control and Monitor of multiple sensors/actuators in multiple rooms
PROJECT PERIOD: February 1st 2010 to June 3rd 2010
PROJECT GROUP: 10grp1020
GROUP MEMBERS:

Luben Ivanchev

_____

SUPERVISOR: Lars Bo Larsen
NUMBER OF COPIES: 3
TOTAL PAGES: 74

Wireless platform for controlling sensors and actuators in multiple rooms.

SYNOPSIS:

This report documents the development of a system to control and monitor sensors and actuators in a house or office. The system developed helps a user to get and change statuses of components quickly and wherever s/he is through a web-based interface. The user can have the same control over the system from their home computer which acts as a local server to the system. The system was developed using Service Oriented architecture:

- A desktop application providing the services for the user and serving as a means of communication between any computer (user) and the micro controllers (sensors and actuators).
- A web service application, where the user can at a glance, get the status or change the status of sensors/actuators and define how the system works.
- A micro controller application, responsible for receiving requests from the desktop application, acts according to those requests and sends an answer back to the desktop application.

The desktop and web service applications have been developed in Java programming language and PHP, respectively. The micro controller application has been developed using C language.

This report presents the system development life cycle, from the analysis to user testing.

Table of Contents

Contents

List of Figures

Preface

This report and all of its contents are part of the 10th semester in Vision, Graphics and Interactive Systems at Aalborg University. It has been written by the project group VGIS 1020 during the spring semester of 2010.

The purpose of this master thesis is to develop a system to control and monitor sensors and actuators, either in houses or in offices. The theme of my project is: Wireless platform for controlling sensors and actuators in multiple rooms. Usability is vital to this project. It is aimed at the average user who has minimal computer literacy.

Each chapter and the corresponding sub chapters are marked by one or more consecutive numbers.

Report outline

The structure of my report is divided into 8 parts:

1. Introduction: where I present the goals of the system and usage scenarios.
2. Analyses: here I analyze all the different technologies and techniques that I thought useful for my project. For instance, I analyzed existing systems of smart homes, sensors and actuators, communication protocols.
3. Design: here I explain how I designed all of the components of my projects, either in a global view or in a more specific view.
4. Implementation: here I explain which problems I faced during the coding phase. I also explain the choices that I made to avoid them and which problems remained without a possible solution.
5. Testing: here I show the final tests that I made to my final program and which results I obtained.
6. Conclusion: I give in this part a brief conclusion of the entire project, comparing the early ideas of the project and the final solution.
7. Appendixes: in this section I included some tables, graphics and example files that are highly detailed and not so relevant to include in the previous parts of the report.
8. Bibliography: I give a list of references that I used to develop my project and to write this report.

Key Terms and definitions

User – This is the person who has this system installed in their property.

Open sensor board – Sensor board developed by Aalborg University. These are powered by 9 volt power supply or battery. They can have multiple sensors installed on them.

Sensors – Any components that can be attached to the sensor boards. These may be read/write such as lights or sirens or simply read such as temperature sensors or motion detectors.

Remote server – Is a machine that has public IP address that displays the data to the user. This remote server has the task of saving information and displaying it. It also can dispatch alarm signals to authorities.

Mobile server – Acts as the information highway between the data gathering sensor devices and the data displaying server.

Bluetooth – Wireless protocol for data transfer, weaker signal.

Radio Frequency – Wireless protocol for data transfer, stronger signal.

Rule - any input received from a sensor that triggers an action.

SOA (**service-oriented architecture**) - is a flexible set of design principles used during the phases of systems development and integration. A deployed SOA-based architecture will provide a loosely-integrated suite of *services* that can be used within multiple business domains.

Authors

The master thesis project *SafeHouse* was developed by the group members Sérgio Pedro and Luben Ivanchev of the group VGIS 1020 till the date of 14th of May of 2010. From that date, the group became two different groups: VGIS 1020 and VGIS 1027.

Thus, the following sections of this report are common to both members, even though there were further minor changes to them:

- Introduction.
- Analysis.
- Bibliography.
- Appendixes A and B.

Therefore, the remaining parts were designed and implemented individually by Luben Ivanchev, member of the group VGIS 1020.

Acknowledgments

I would like to thank all the people who helped me during the accomplishment of my project, and particularly:

- My supervisor Dr. **Lars Bo Larsen**, semester coordinator and project supervisor for his help, management and advice.

- Mr. **Ben Krøyer**, from the antennas lab, for his help and support regarding the micro controllers and external sensory.

- Ms. **Mette Billeskov**, semester secretary, for her support regarding the administrative issues.

- Mr. **Per Mejdal Rasmussen**, from the IT workshop, for his help and support concerning the desktop application of my project.

- Mr. **Jørgen Schiønning**, CEO of PDM Technology, for his advice and support for the master thesis regarding Service Oriented Architectures. Also, for this permission of using software libraries of PDM Technology in this project.

- Prof. **Frank Fizek** and Mr. **Morten Pedersen**, from the mobile devices lab, for lending us the necessary phones for this project.

1. Introduction

Sensors and actuators are a vital part of our everyday life. In nature, they are present in all life forms around us. Without them we could not survive even a second. Our eyes, our tact, our taste, our muscles, are all intrinsically part of what makes the human body. The big agent of this control is our brain, which responds to electrical signals sent from our body's sensors, sending electrical stimulus to our actuators (e.g. muscles).

Nevertheless, humans are also dependent from other kinds of sensors and actuators, the electronic or mechanical ones. For instance, when I put a dish cooking in the oven, there is a sensor to detect when the temperature exceeds the temperature that I established. After, this sensor sends a signal to the electrical plate (the brain of the oven) and this one sends a signal to the thermal plate (actuator) in order to shut down for a while, till the temperature drops down a bit.

In the last decades there has been a progressive development of **Smart Houses**. The idea is quite simple: instead of having small brains in each machine, the house has a unique brain to control and monitor all the sensory and actuators built in. Many companies have developed systems for this purpose and increasingly more prototypes are coming to expositions and fairs. There are many goals with these systems: some simply give the user a more centralized control of his house; others are more concerned with power saving; and many are concerned with security.

Figure 1 - Energy saving in a smart house. If the window is opened, the radiator turns off.

Some of these systems are intended to be controlled just at home with a built in panel in the wall where the user can control everything at home. Other systems are intended to be controlled wherever you are through a cell phone. The idea of my system is to have an easy-to-install system for controlling sensors and actuators in a house or office, through wireless communication that allows the user control his/her house or office wherever they are at any moment of the day.

The entire system is idealized to achieve a Service Oriented Architecture. In this way, the format of inter communication between applications should be XML.

The system is made in such a way that the development of one of the components does not interact with the development of another, i.e. each component is completely independent of any other, meaning that the only part that all the components have to have in common is the format of communication between them (input/output), i.e. the only thing that has to be ensured is that the format is readable and interpretable by each component.

1.1. Scenarios

At this point, I have introduced the fundamentals of my project. Nevertheless, all of this does not make sense if I do not explain a real user scenario. For instance, let us consider a common citizen, called Peter, who decided to implement in his house the *SafeHouse* system with the following equipment:

- For his bedroom: a temperature sensor, a magnet sensor (to detect if the windows is open or not), a light bulb and a heating system.
- For his living room: a heating system and a distance sensor (to detect if the window is open or not).

After setting up the system, Peter goes to work, already with the *SafeHouse* application installed on his home computer:

1. During the morning, Peter realized that he left the light of his bedroom turned on. Quickly, he opens the *SafeHouse* website and in few seconds, he turns off the light. Peter just has made his good action for the environment.
2. During the afternoon, Peter talks with some co-workers and they decide to have a party in that evening at his place. He knows that he has left the window of the living room opened but he did not turn on the heating. He knows that is not pleasant for the guests to be in the living room freezing. Since he cannot close the window by distance (he did not want to spend more money), he simply decides to turn on the heating in the living room.

Another possible scenario:

Peter and his family go on holiday. Peter activated his *Safehouse* system to ensure that his home is safe. While on holiday Peter wish to monitor the status of his home and be alerted if there is some intrusion into the house. Peter has set up email accounts that will immediately receive a notification if something is detected. Upon detection Peter can notify the authorities manually by calling them after receiving an email or automatically via the email service.

After introducing briefly this topic and explaining two possible scenarios, I will present the analysis of existing systems, SOA, mobile phones and communication protocols in the next section.

2. Analysis

In this section all theory behind my project is going to be described. This section will begin with a presentation of a **general diagram** for my system: which system components are involved in it and how the general interaction between them is.

Hereafter, I will introduce the **service oriented architecture**: how my system is able to work based on services in each application, rather than based on a fixed point-to-point application. In order to achieve the concretization of my project, I am going to make a brief introduction about **existing household systems**: which products are on the market and their potential for the future. I will also describe the system architecture choice.

In the next section I will introduce the **embedded devices** used in the system and which sensory they have incorporated and the data extracted from each sensor or actuator. Finally, in the last part I will talk about the different **communication protocols**, especially within the wireless communications.

I finish my analysis section with the **problem formulation** for this master thesis work, considering all the previous sub sections.

2.1. Overall System Architecture

As I have introduced above, the key idea of my system is to allow the user control his house or office in a glance through any web browser. Each user action has to be made through an entry point in the house, possibly a simple server running in a home computer. This means that the user can interact with his house anywhere in the world through his mobile browser.

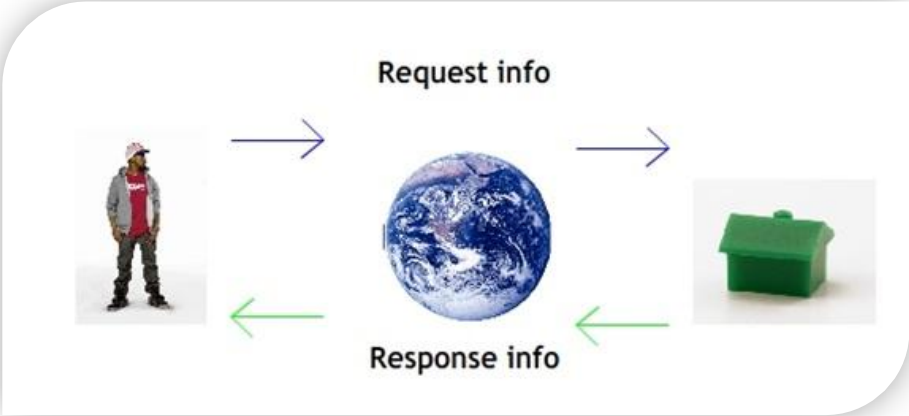

Figure 2 - The general idea of the system.

The most important idea of my system is to allow a user to, at anytime from anywhere in the world, request status information from his automated home. Upon a user request, the system responds with the desired data almost instantaneously. Of course, an entry point has to be considered to the system, i.e. somewhere; somehow the user needs to connect to this house. The system has to be, in this way, based on 3 point communication: The user (blue square), the house entry point (green square), and the sensors/actuators (red circles).

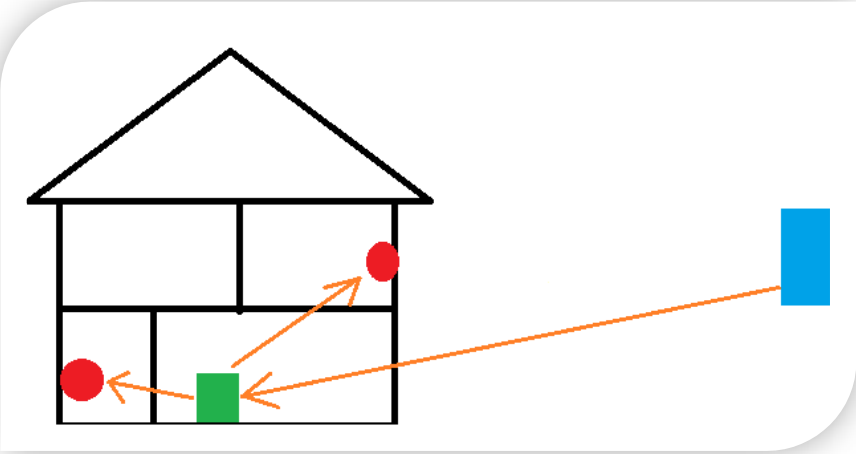

Figure 3 - Three point communication of the system

Slave-Master Architectural configuration:

Below I describe some possible architectural configurations that I considered. I list advantages and disadvantages of each system and present my final choice at the end.

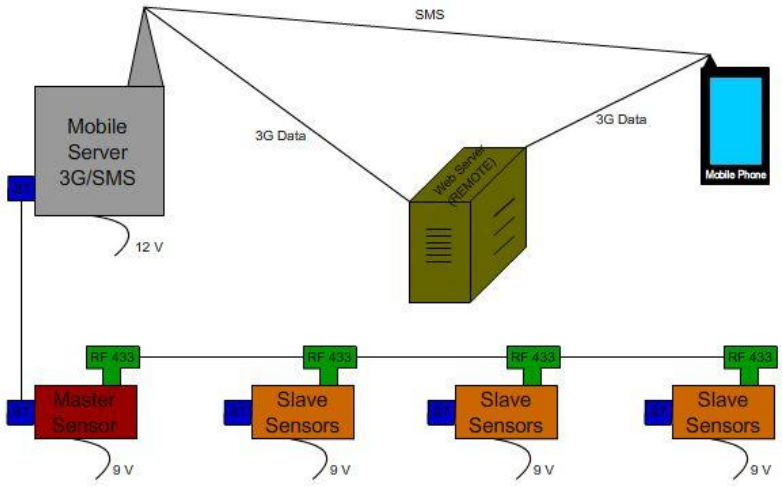

Figure 4 - First basic set up of the system.

The above diagram illustrates my initial idea. I intended to install radio frequency (RF) modules onto the open sensor board in order to facilitate the communication between the different sensors. The radio frequency transmits at 433MHz, which is the same frequency as RFID. This setup would mean there will be interference with any RFID system that is within range. However the use of RF module allows for greater distances between sensor boards and wall penetration. This setup also requires the sensor boards to be arranged such that one sensor board is master and all others are slaves. This brings about the problem that if the master were to fail all other sensor boards would be rendered useless until the master is repaired. In this setup the master communicates, using Bluetooth, with a mobile server which is installed locally (in the home) within range of the master sensor. The master sensor is responsible for sending all data from the sensors to the mobile server which, in turn, is responsible for sending all data to a remote server via 3G/4G or directly to a GSM mobile phone via SMS protocol. The same applies for data that is sent from the user to the sensor boards through the mobile server and then into the master sensor board which will delegate instructions to the lesser slave boards.

Mobile server Architectural configuration

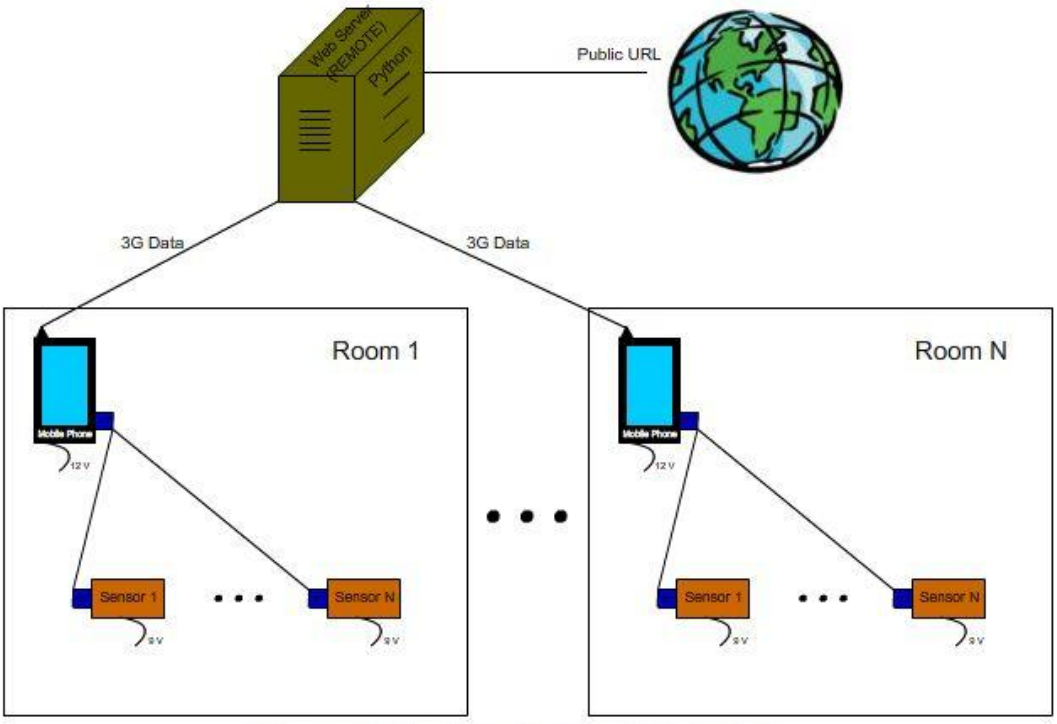

Figure 5 – Using mobile server/s in the system.

The above figure displays another architecture that I decided to consider. This setup uses Bluetooth to communicate between all sensors and mobile server. In this way, if one sensor board fails the others remain functioning. Multiple mobile servers are needed due to Bluetooth's inability to neither cover long distances nor penetrate walls. This architecture allows different rooms or areas to function independently from one another and remain functioning if one mobile server fails. This architecture is cumbersome due to the fact of all the *mobile phone server devices* that need to be installed per room. This architecture does offer the versatility of a low power consuming, long distance data transmitting and sensing device; however it is not practical for a domotic system. Both of the above systems require a remote server. The remote server is a very complex part of the project. Vital decisions need to be made as to its architecture and implementation. It has important tasks that include sending control information to sensors and receiving input from them. It also needs to have a user friendly interface that allows the user to view status, activate or deactivate sensors, and easily program schedules and rules for the system. All while maintaining a log of previously occurred events.

The remote server needs to be a machine with public access to a URL. This serves as the keeper of information and controller of sensor boards. It also has the important task of notifying the user of any sensor input.

The remote server will have the tasks of:

1- Storing information with time stamps for historic backup.
2- Displaying current status to the user.
3- Dispatching alarms to the user's mobile phone.

In the following picture is shown which options I have considered for the remote server.

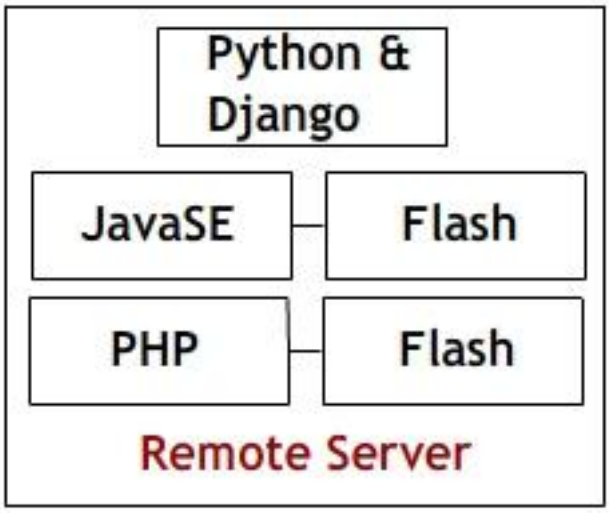

Figure 6 - Possible server implementations of the remote server. Python vs Java vs PHP.

For my project any of the languages are usable for the implementation however a decision needed to be made. In the end this configuration proved to be impractical because it would require a local server to communicate with a remote server. Instead having a local server would all together resolve the need for a remote server and avoid many legal and privacy concerns that the user might have.

Chosen SOA architecture design

The system needs to be designed in such a way that it adheres to the principles and practices of the SOA as described in section 2.2 below. To ensure a solid implementation SOA I had to make sure that the system could dynamically adapt to hardware changes and notify this to the user somehow. The diagram below shows how this is achieved. There are two GUIs presented below. Essentially they allow the user to perform identical interaction with system. However, the desktop GUI is always available and can be accessed directly from the local computer on which the software is installed. That is to say if the server is down the web GUI will not be accessible but the desktop GUI will.

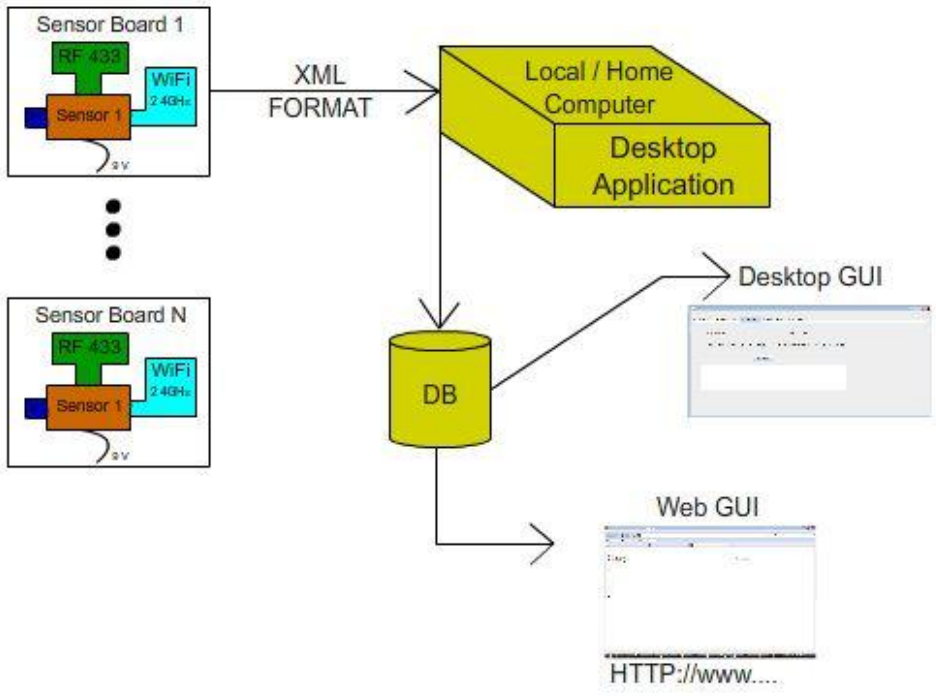

Figure 7 - SOA implementation for chosen architecture.

The system should be able to *automatically and dynamically*:

1. Detect any addition/removal of sensor board/s from the system.
2. Detect all sensors and their control methods.
3. Update the database.
4. Update the desktop/web GUIs.

All of these tasks need to be done on system startup and periodically while the program is in operation; throughout its life cycle.

To reach the completion of such system, I have to investigate first the different topics that are going to compose it. Moreover, before going through the physical components, I have to introduce the architectures for such systems. Specifically, in the next chapter I am going through Service Oriented Architecture and how it can influence the development of software.

2.2. Service Oriented Architecture

In traditional systems, the focus always stands in the development of the application for a specific propose, without taking into account other possible platforms or other kinds of communication. Moreover, the software model is in most cases separated from the business model. Thus, concern is given to design and develop the software components independently of the business components. Due to this splitting, most likely the developers design the system in an enclosed way, i.e. they implement a standalone application. In another words, even with a good result in the final product, any change in a block of the program will cause changes in the blocks depending on that one. This is really expensive for a company, which afterwards is going to have several costs in the system support.

With this comes the key idea of a Service-Oriented Architecture (SOA): **combine the software model with the business model** [5] [14]. In this way, each required business service will become a software service. All the blocks of an application are no longer enclosed together and inflexible for changes. Each service is independent and it can run wherever required with any kind of communication protocol. Moreover, each service can be reused in another business model or changed without compromising other services. No doubt, it is harder to achieve this when a team is doing its first project. But afterwards, a small modification in one of the services or the construction of a new one makes a brand new application ready to be used by another customer. This is really important in terms of costs: the flexibility to separate the services by their functionality. The XML format fully enters in the SOA logic (although it is not required). With a small change in a service, I have to add to the XML a couple of new fields to be understandable by that service, without compromising other services. This is really powerful and it can make a significant difference in the time spent in development and in terms of costs.

2.2.1. Key elements for a Service Oriented Architecture

In order to realize a Service Oriented Architecture, several rules have to be followed [11]:

- *Loose coupling*: A class or a structure in a software application has to have as little possible knowledge of other components in the system. This means that if a requirement change occurs, the cost of that change will be minimal.
- *Implementation neutrality:* Good software architecture should not be dependent on programming language details.
- *Flexible configuration:* One of the strongest principles of SOA. With this, different components can be bound together later in the project.
- *Persistence:* Services should endure long enough and have a correct way of handling exceptions.
- *Granularity:* This is strongly related to the combination of the software model with the business model. The services should not be seen in detailed interaction among themselves, but instead as a high-level view more related to the business model.

Respecting these rules will contribute to better software development. This does not mean that the software development will be faster in the initial stages, but eventually, it will become easier to correct or to add or change features.

So far, I have been introducing SOA and its rules, but in order to see a better picture of SOA architecture, I have to introduce two main concepts: the service provider and the service requester [29]. Obviously, the service provider is the one that provides all the services available to one or more service requesters. The crucial thing here is that the service provider can provide different kinds of services for different purposes without changing the structure inside of it. In this way, different service requesters (green triangles) will retrieve different data (orange arrows) from the service provider (blue hexagon). This diagram is relevant to my project because I will have only one service provider. However it is important to note that SOA may support multiple service providers as well as service requesters.

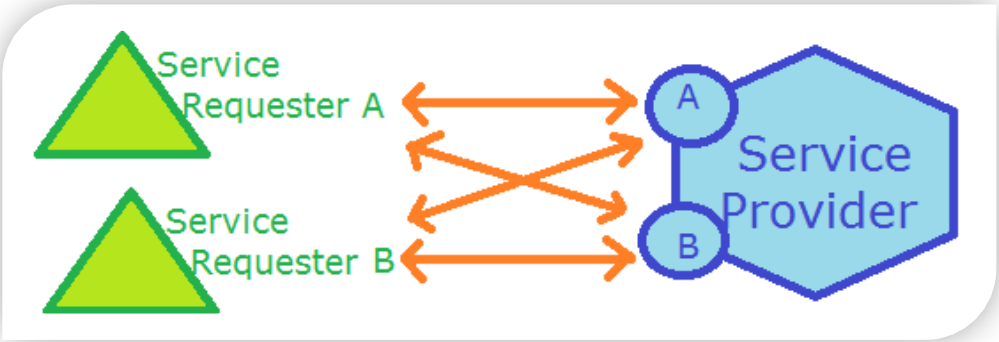

Figure 8 - Basic scheme of SOA architecture (Service provider and Service requesters)

In other hand, I have two kinds of services: simple and composite [24]. Simple services are the ones that take care of one particular action within the service provider. These simple services can be used independently or reused to make a composite service. Thus, a service requester can either make a simple request or a composite request. This also follows the idea how structured the XML is: an XML has always a root element. If it is a simple request, the root of this element is simply the name of the service. If it is a composite request, the root of the element has to be different and incorporate the different simple (or composite) sub-requests (upper picture).

There is another component intrinsic to SOA: the service registry. The service registry is where the service providers are going to store the services information for the service requesters. Before a service requester sends a request to the service provider, it should first find the service it is looking for. This service registry does not need to be stored in the same place where the service provider is running. This allows even more flexibility to the service providers to distribute the different services across different machines. In the end, when the service requester asks for a service, it retrieves the address where that service is running. Alternatively to this service registry, a service tunnel can be implemented. In this way, the service requester sends a request directly to a main entry point of the service provider, but afterwards this one

redirects the call to the location of the service. This is similar to how communication via proxy works.

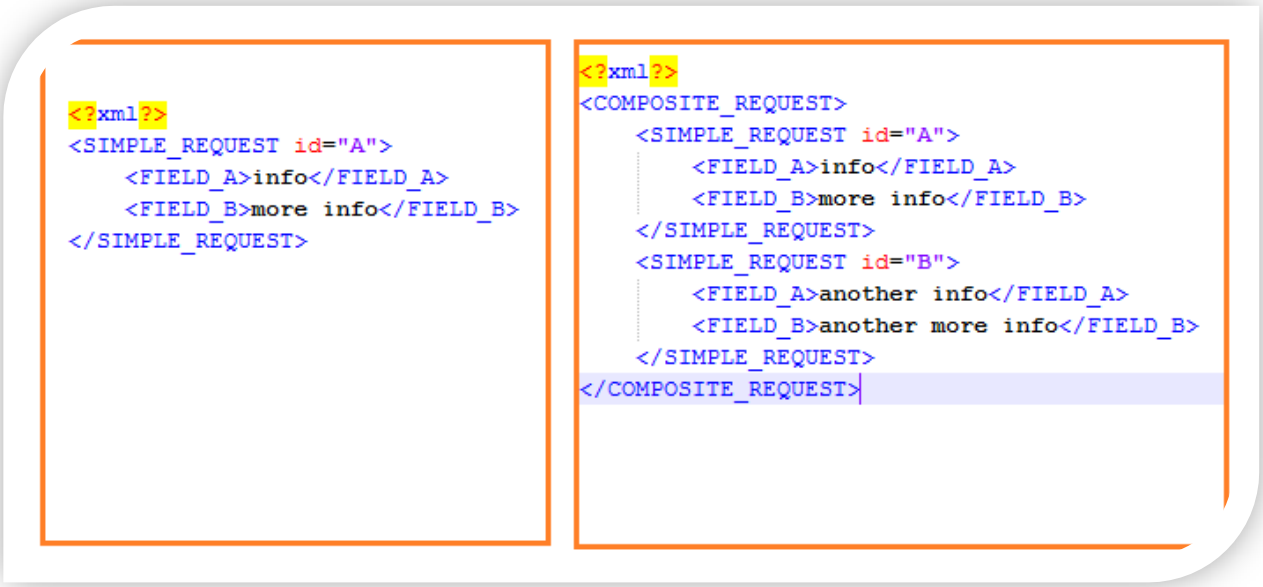

Figure 9 - XML scheme of a simple request (left) and a composite request (right).

Hereafter, SOA architecture is suitable to be implemented in any kind of solution. It provides a well-structured and flexible way of creating software and it allows the software to be constructed based on services. Thus, it makes sense to use SOA in *SafeHouse* system. But before progressing into component details, it is crucial to introduce the existing system within the household market and research field.

2.3. Existing Household Systems

The concept of Home Automation (Domotics) has existed for many years. Yet it is of recent years that the electronic components required for home automation have become cheaply available thus allowing systems to be more affordable to the average person. Still the market is predominantly targeted at upper class people (Meyer and Schulze 2006). A cheap alternative will be able to tap into a much larger customer group.

There are many standards for home automation devices. Yet most devices and systems cannot communicate between themselves [32]. This requires skilled professionals to configure this communication between these devices. This configuration of such a system is one time only and will require further configuration if the user wishes to modify his/her system.

The home automation market is rapidly growing and more and more people are installing smart home systems. The figure below lists some motivations for users to implement a smart home system.

Figure 10 - User motivations for installing smart home products [20].

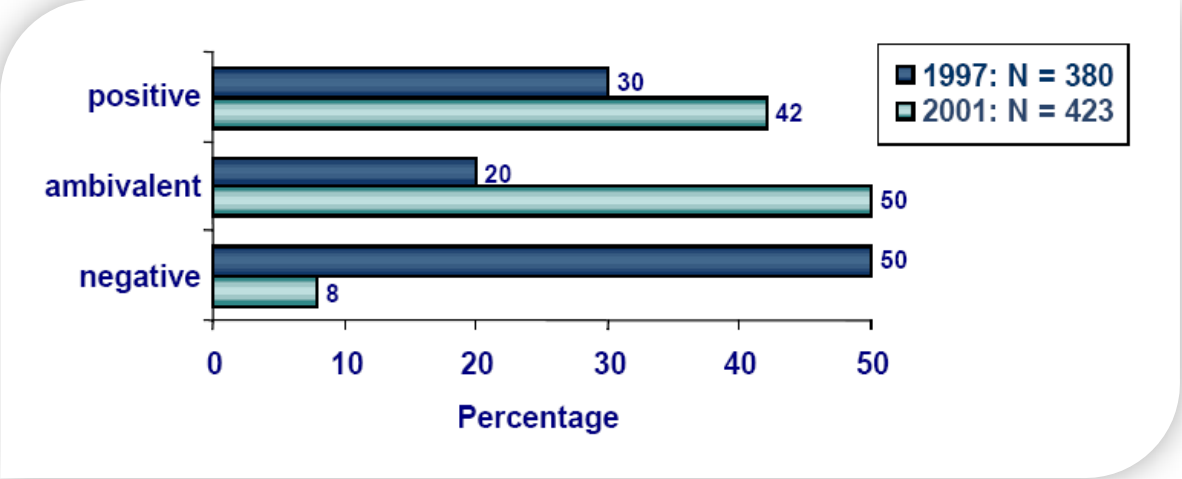

Figure 11 - User acceptance of smart home systems [20].

Below I list some negative arguments and problems with smart homes according to [20]:

- ✘ Smart homes are too expensive.
- ✘ Installation problems, no plug-and-play.
- ✘ Programming is too complicated.
- ✘ High repair costs.
- ✘ Privacy issues.
- ✘ Dominance of technology.
- ✘ Access rights, security issues.

Contrary to these problems I present a list, below, of guidelines of what a smart home should be according to [20]:

- ✓ Easy to use
- ✓ Understandable
- ✓ Affordable
- ✓ Easy to install, plug-and-play
- ✓ Easily expandable
- ✓ Time saving
- ✓ Ensure privacy and security
- ✓ Energy Saving

Problems still exist with smart home systems. This project aims to develop a solution that solves as many of these problems and, at the same time, adheres to the general guidelines of what a smart home should be.

The novelty of this field of research has led to the emergence of many companies [4], [12], [17], [18] who are researching and creating smart home systems or products. In this section I will describe some of these existing systems. This not all the products that exist but serve to demonstrate what are available and how this system will relate or differ to them.

The X10 standard is very popular and most (older) systems use it [6]. I will look at X10 in more detail later. In Europe the European Installation Bus is an improvement of the X10 standard and widely used on the continent. The are many others such as Echelon Lonworks, and National Instruments LabVIEW modules, all of which offer the possibility to add internet servers for remote web monitoring and control of a home. However, a trend is seen in newer systems which are more favour to use the newer wireless technologies. In industry many companies exist that provide home automation solutions. The communication technologies these companies use often differ from one to the next. This makes compatibility a problem. As mentioned earlier another problem is that hardware devices are manufacturer specific and there is no standardization either.

A study published by [2] is of relevance to this thesis. Their project aims at developing an online system that connects to a user's currently existing alarm system and allows the user to view the current status of the security system via TCP/IP. The user would also be able to arm and disarm his/her security system remotely from any computer or mobile device with an internet connection. Their problem formulation was that despite the existing companies there was a need to re-use current fully functional home systems instead of replacing them. In their study they created a "micro web-server" from a Modtronix SBC45EC board. The board contained a PIC18F452 microcontroller (similar to the one used for this project). Although their system is specifically designed to incorporate older security system it still has relevance to my proposed system because their general architecture is more or less the same as this project.

The authors of [15] propose a solution for controlling home appliances and providing home security using the Short Message Service (SMS) protocol and wireless technology. Their system relates to my initial consideration of using a mobile device as a server. The Home Application Control System is divided into 2 sub-sections all controlled from a local computer. One sub-section is for appliance control and the other is security control. A GSM modem acts as a mobile server and communicates with the remote user via the SMS protocol. The following picture shows their systems architecture.

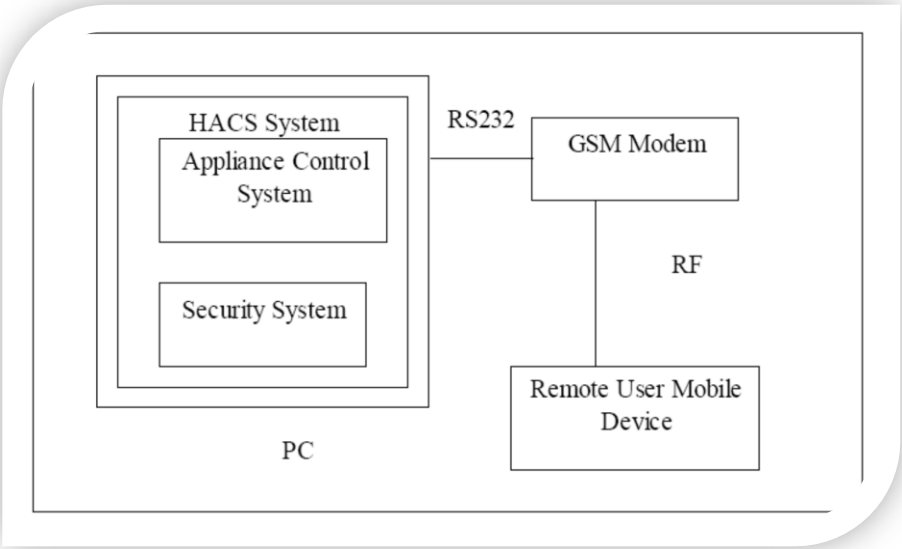

Figure 12 – System architecture based on a SMS protocol to interact with a remote use [3].

The paper however does not mention how communication is achieved between the PC and the appliances or the security system. However it is interesting to note they chose to use the SMS protocol to allow remote access.

Another solution is proposed by [3]. Their system requires a local computer to act as user interface, database and webserver. This is similar to the architecture I propose. The system requires the use of a master node that communicates between the computer and the other nodes. The master node communicates with the local PC via cable (RS 232) and uses Radio Frequency 433MHz to communicate with the slave nodes; this is shown in figure 9. I considered using the master/slave approach however a disadvantage to this set up is that all reliance is placed on the master node; if it fails all other nodes will cease to function. In [3] a SSL algorithm is used in order to ensure security on the webserver in addition to a user login and site certificate. This project used a PIC16F877 microcontroller which is similar to the microcontrollers I use.

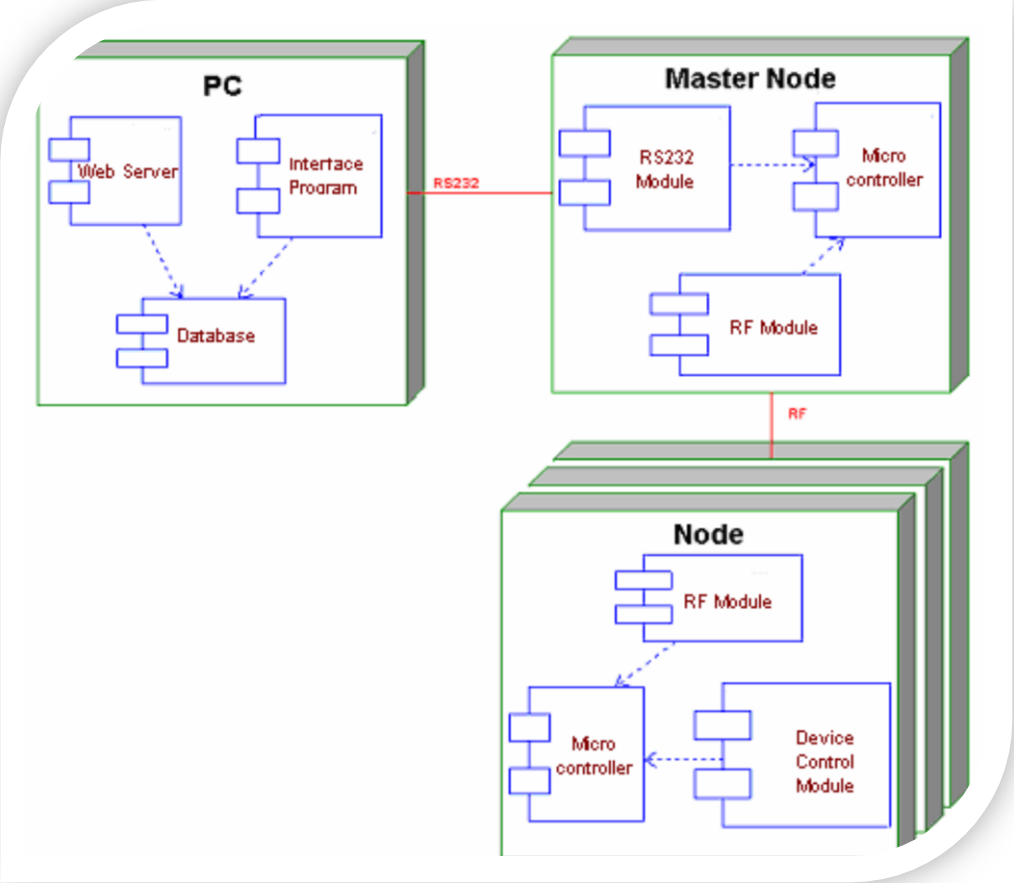

Figure 13 - System architecture proposed by [3].

A limitation of this system is in the addition of new devices to the home network. Currently devices are manually added by the user through the user interface. A simple handshaking protocol is used to do this. Here there is room for improvement. Value can be added to the existing product by automating this process. This will alleviate the user's responsibilities and make the system more plug-and-play. This author reported a penetration range of 100 meters of the RF medium in a concrete building and 200 meters in open ground. Another problem presented in this project is the failure of the system when power to the wall socket is terminated. Their solution is to use backup batteries or an uninterrupted power supply (UPS). This problem exists in all systems that require electricity, including this project. Solutions include backup batteries and generators. Using only backup batteries guarantees uninterrupted operation of the system until the battery runs out of power. Using a generator will ensure longer operation (until the diesel or petrol finishes), however the system will be inoperative from the time the power is cut until the generator is started. A combination of both is the best solution for extreme cases, however, costs are dramatically increased and consideration needs to be taken for the exhaust fumes of the generator and the considerable noise that it emits.

In [6] the X10 wired approach used in conjunction with the PIC16F877 microcontroller. Although I do not intend to use this wired approach it is still useful to note that the PIC series

of microcontrollers are well suited and popular for home automation applications. This means support and maintenance should be widely distributed and available.

According to [32] a service oriented approach which is event driven with web services is the best solution for a smart home system. The advantage of using web services is that different systems may vary in design and can function independently on different platforms, while still interacting with each other on a high level standard. Using a SOA approach allows services to support many various applications. Most importantly SOA approach allows autonomy, for the customer and even the device. Autonomy by definition means self-governance. It is used here in the sense that the smart house system can regulate itself and is independent of external forces. That is to say the home owner has full control of the system. This eases legal issues as well.

2.3.1. Conclusion

In the previous sub chapters I have described different companies and products in the market. Either through built in panels in the house or through remote applications over IP or GSM, all of them are intended for the same goal: give the user the power of controlling his house, either indoors or/and outdoors. Probably the tendency will be to turn houses more automated, independent of user actions. In the design part, I am going to consider these different solutions presented for the development of this work.

Now that I have presented briefly the SOA architecture and some existing systems, I am going through in more detail each component involved in my system, starting with the embedded devices.

2.4. Embedded Devices

As referred in the beginning of this report, sensors and actuators are everywhere. In fact, they are incorporated in the so called embedded devices. These kinds of devices are everywhere: in our washing machines, in our refrigerators, in our television sets, in our internet modem are some examples. Often we do not even realize the importance that they have in our daily life and without them, most of our daily tasks would be hard to realize. Let us just imagine what an oven would be without an embedded device to control the temperature inside: the oven would be so hot that would put my life in dangerous just in a matter of minutes, or probably melt. This is just simple example, but reflects how dependent we are of these useful devices.

Embedded devices have typically a microcontroller or a digital signal processor in their core. Moreover, they are ideal to perform a specific task with low power consumption. An interesting solution is the **open sensor board** (OSB) from Aalborg University and Technical University of Berlin (Department of Electronic Systems 2009), which provides an open source sensor hardware platform for development and integration with different kinds of sensors/actuators. Hereafter, I introduce the platform that I am going to work with:

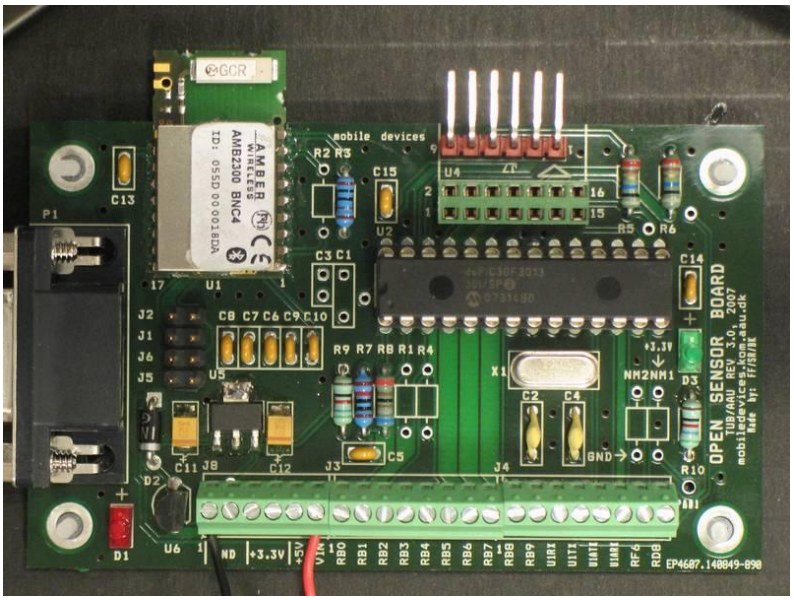

Figure 14 - Open sensor board with pic30f3013 and Bluetooth UART.

In order to develop code for it I use MPLAB IDE 8.2 and PicKit2 to program it. There are many sensors that can be put on the OSB. In the following sections I introduce each kind of component available for the OSB.

2.4.1. Distance Sensor

A distance sensor is very useful in many situations. A concrete example of a distance sensor in our daily life is the parking sensor in modern automobiles. This sensor gives an alarm each time that the car excessively nears an object. The alarm becomes more obvious the closer the

car is to the object. In a smart house, a distance sensor can be used in many situations: when someone gets close to a safe lock where one keeps valuable things, when someone enters in the house or simply as a motion sensor. The infra-red distance sensor is produced by Sharp, model name: GD2D12 [1]. This device gives as output a non-linear analog signal for distances between 10 and 80 cm. Since the PIC micro-controller has an analog-to-digital converter (ADC), my problem addresses the fact that distance sensor gives a non-linear signal. To better understand this, let us take a look in the following graph:

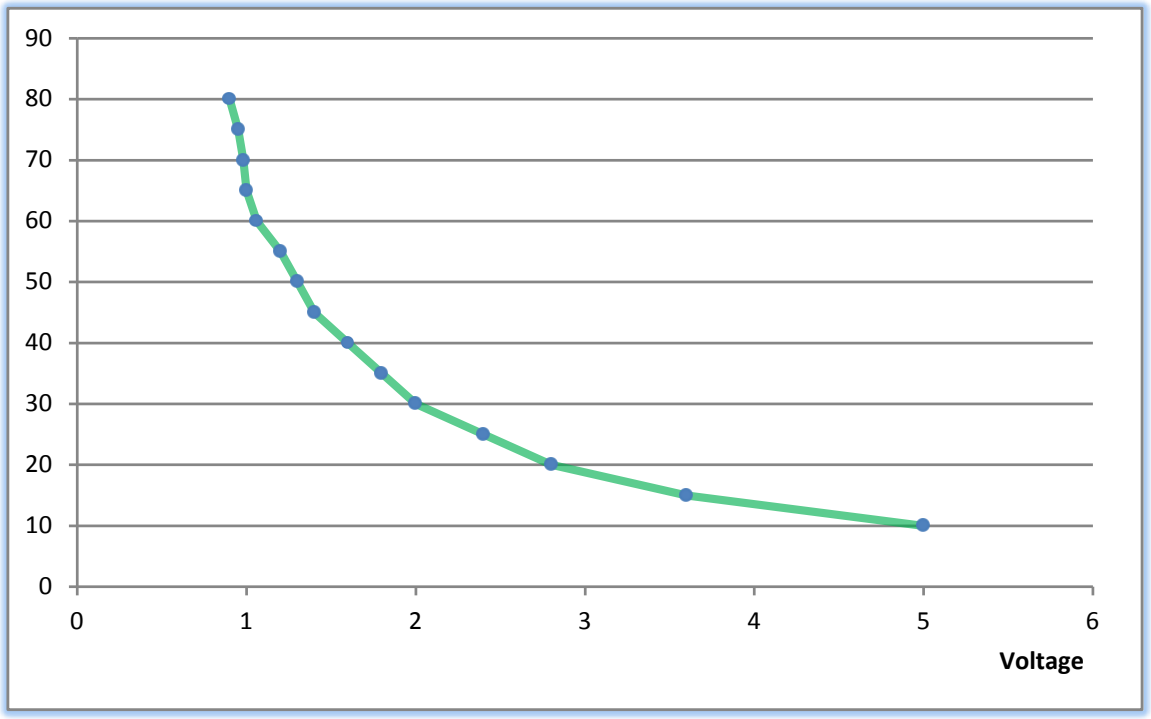

Figure 15 - Sharp GP2D12 Voltage vs. Distance (in cm) graph.

As you can see, the graph is not easy to understand in terms of output (voltage). Instead of making a case-by-case interpretation of the signal, i.e. if output is 3V means this or 1V means that, a linearization of the signal using 1st Grade Linear Regression is used. However, the application of a linear regression to the entire graph, results in huge deviations from the real output. For more detailed information, the reader can refer to Appendix A.

Instead of applying a linear regression over the entire graph, the graph is divided into three sections: voltage between 0.9 and 1.4 Volts; voltage between 1.4 and 2.4 Volts; and voltage between 2.4 and 5.0 Volts. Finally, a linear regression is applied to each section and the following graph is obtained:

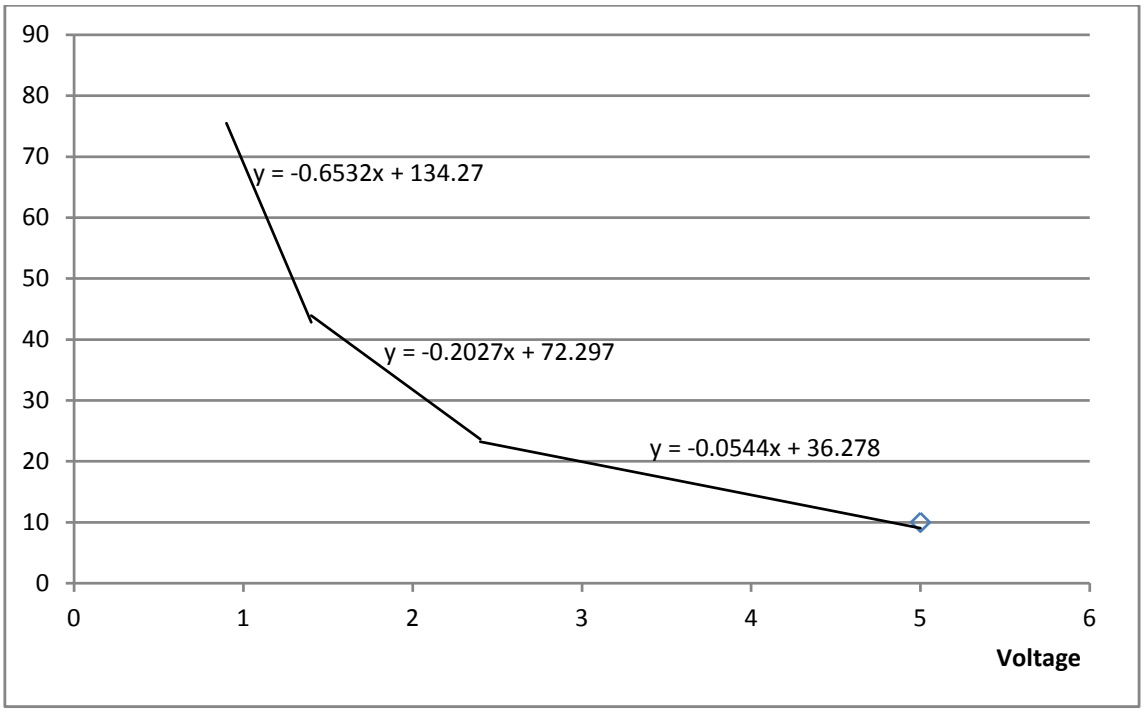

Figure 16 - Linearization of the GP2D12 signal (volts vs. cm).

The application of the above formulas gives values that are sufficiently close to the real distance and serve the purposes of this project.

When comparing the real values and this approximation, we can see that the error margin is low enough to be considered in this project. Let us remember that the primary goal of this project with the distance sensor is to detect "someone nearby something" and even a maximum error of 5cm will not make a big difference in a smart house implementation. On other hand, this linearization cannot be used in a parking mechanism or a space shuttle for the obvious reasons mentioned above. In general distance sensors are always giving different results depending on the room temperature and supply voltage. However, in most cases it is useful, since the operating temperature of this component is between -10 and 60 degrees Celsius.

Another idea could be to make a 2nd grade linearization. With this, an accurate approximation of the results with the real values can be obtained. Nevertheless, this solution is not sustainable for a micro controller, which does not have a powerful processor and will take time to make that calculation. Finally, the values can be stored in a table, which will be quick but very sensitive to changes, i.e. if I change the distance reading on the sensor with another one, I have to store all the new values again.

A last approach would be to simply set the distance sensor to **true** or **false**, meaning that the value is true if someone is nearby (in the range of the sensor) and false otherwise.

In the next part of this chapter I am going to talk about another important component in my system: the temperature sensor.

2.4.2. Temperature Sensor

A temperature sensor is perhaps one of the most useful sensors that we can have anywhere. In the project *SafeHouse* we are going to use the LM35DZ [7].

Figure 17 - The temperature sensor LM35DZ.

Unlike the distance sensor, this temperature sensor gives a linear response, more precisely 10mV per degree Celsius, working in temperatures between 0ºC and 50ºC. In the following graphic is represented the relationship between temperature and voltage. As you can see it is fairly easy to obtain the temperature.

In the next chapter I are going through another sensory implemented on the open sensor board, concretely a green led, a buzzer and a magnet sensor.

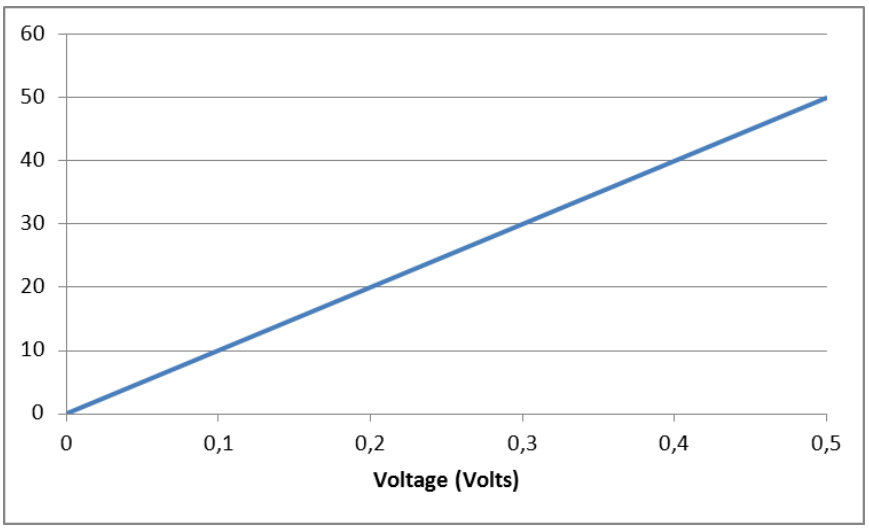

Figure 18 - Relationship between voltage and temperature (ºC) in the LM35DZ.

2.4.3. Other Sensors

Besides the temperature and distance sensors, I have decided also to use other components. One of them is the magnet sensor. This component gives a high-low output signal, meaning that the signal is in low output when a metal is in contact with it and in high output otherwise. This magnet sensor can be used near entrances, such as doors or windows, to detect if that entrance is open or close.

I have also decided to simulate a normal light bulb and an alarm. Instead of using real components, like the ones that I have at my homes, I use small green LEDs and a buzzer. The principle is the same and it works for my purpose of providing a proof of concept in the end of this project.

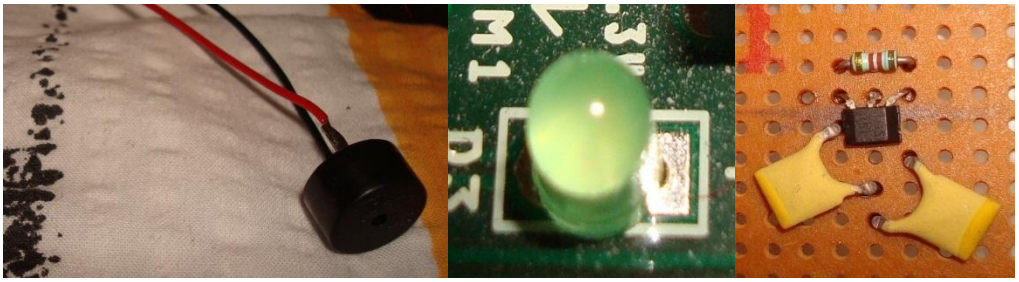

Figure 19 - Other components. From the left to the right: a buzzer, a green led and a magnet sensor.

In the next chapter I introduce mobile devices, where I intend to implement a user-friendly and quick application to access the house.

2.5. Mobile Devices

It is well known that mobile devices are becoming increasingly powerful. As time progresses the division between mobile phones and computers is becoming smaller. It is possible to develop applications for mobile devices as we do for computers and the operating systems for mobile phones have increased in number and robustness. In this way, it makes sense to think about a collaborative platform between cell phones and other devices around them, particularly to access a house and control it. Nevertheless, one of the main limitations of mobile devices is exactly the different platforms that we can find for them. Another limitation is processing power.

The evolving smartphone market

Global Share by Platform	Q3 2007 Units	Share	Q3 2008 Units	Share	Q3 2009 Units	Share
Symbian	21,219,390	68.1	18,583,060	46.6	19,064,288	46
RIM BlackBerry	3,298,090	10.6	6,051,730	15.2	8,703,262	21
Apple iPhone	1,107,460	3.2	6,899,010	17.3	7,459,939	18
Microsoft Windows Mobile	3,797,360	12.2	5,425,470	13.6	3,647,081	8.8
Google Android	0	0	0	0	1,450,544	3.5
Other (Palm, Linux)	1,733,940	5.6	2,890,830	7.3	1,243,323	3
Total	31,156,240	100	39,850,100	100	41,444,104	100

Global Share by Maker	Q3 2007 Units	Share	Q3 2008 Units	Share	Q3 2009 Units	Share
Nokia	16,025,690	51.4	15,485,690	38.9	16,577,642	40
RIM BlackBerry	3,298,090	10.6	6,051,730	15.2	8,703,262	21
Apple iPhone	1,107,460	3.6	6,899,010	17.3	7,459,939	18
HTC	850,400	2.7	2,308,210	5.8	2,072,205	5
Motorola	2,058,500	6.6	2,313,930	5.8	out of top five	
Other	7,816,100	25.1	6,791,530	17	6,631,057	16
Total	31,156,240	100	39,850,100	100	41,444,104	100

Figure 20 - The evolving smartphone market from 2007 till 2009 (McLean 2009).

On one hand, we have a wide spread platform for most Nokia devices – Symbian - in its most popular versions: S40 and S60. On the other hand, we have different operating systems per manufacturer, which does not help to choose a platform to work with. Besides, a developing team has to concentrate effort in learning each platform instead of focusing in the development itself. Soon, the need for a hardware-independent framework was needed. In this matter, Sun© has integrated its Java virtual machine into mobile devices, the so called MIDP (Mobile Information Device Profile) or JSR 118 [10]. This virtual machine was one of the first attempts to unify the development for mobile devices. Because of this, Java comes naturally as a good choice for mobile development.

In other hand, the world-wide corporation Google© has introduced its first operating systems for smart phones, the Android OS. Following-up the launching of the operating systems, Google also has launched the Android SDK, a Java-based framework for Android OS

development [34]. With this comes another advantage for programming in Java. Nevertheless, the SDK is still in development and thus, I have decided to neglect Android as a possible solution. Besides, the Android devices available were limited to few numbers at the time of this thesis.

For Nokia devices with Symbian S60 operating system [21], different options for the development were available: the native Symbian C++ [26], the brand new Qt framework [9], the J2ME (Java 2 Micro Edition) and Python for S60 [21].

If a developer wants to go deep in the capabilities of his mobile device, s/he should use Symbian C++. With this language, it is possible that the developer can do anything with his mobile device, respecting the device's limitations, obviously. Of course, with the use of a more powerful language one loses portability of code, meaning that if an application for Symbian S60 5th edition is developed, perhaps it will have problems porting to a S60 3rd edition. Another disadvantage is the time to develop an application for Symbian C++. Generally speaking, I have to be more aware of device details, instead of concentrating in the core components of the application. For instance, in the case of my thesis, this would become unsustainable in terms of time available vs. time to develop.

Another option could have been the Qt platform. This platform is a way of Nokia giving a simpler access to the device functionalities. With this platform, the time of programming is reduced, compared to the native Symbian C++, since it avoids a lot of complicated details regarding the GUI rendering. At the time of this project, Nokia was still launching platform components and documentation. Because of that, I have decided not to explore a completely new platform for this project.

On another hand, there is Python. Python is one of the easiest languages to use in mobile phone programming. There are uncountable frameworks and development kits based on Python and no exception goes for mobile devices. There is the so called pyS60 in different versions depending on the edition of the Symbian. With python an application can be created quickly and effortlessly. However, as referred above, python comes in different version, meaning that for each edition of Symbian a different development kit has to be used. This is a big disadvantage when considering porting an application to a wide-range of devices.

Finally, there is J2ME. This platform uses Java as programming language and it has the advantages and disadvantages of a high-level programming language. On one hand, the development of code for a high-level language has proven to be faster. A proof of this fact is the job trends comparing the four different platforms I have discussed (following in Figure 21). An obvious disadvantage is that with Java you do not have access to low-level resources, but normally this situation can be over turned in some way. The biggest advantage of Java goes exactly for the range of Java enabled devices [28].

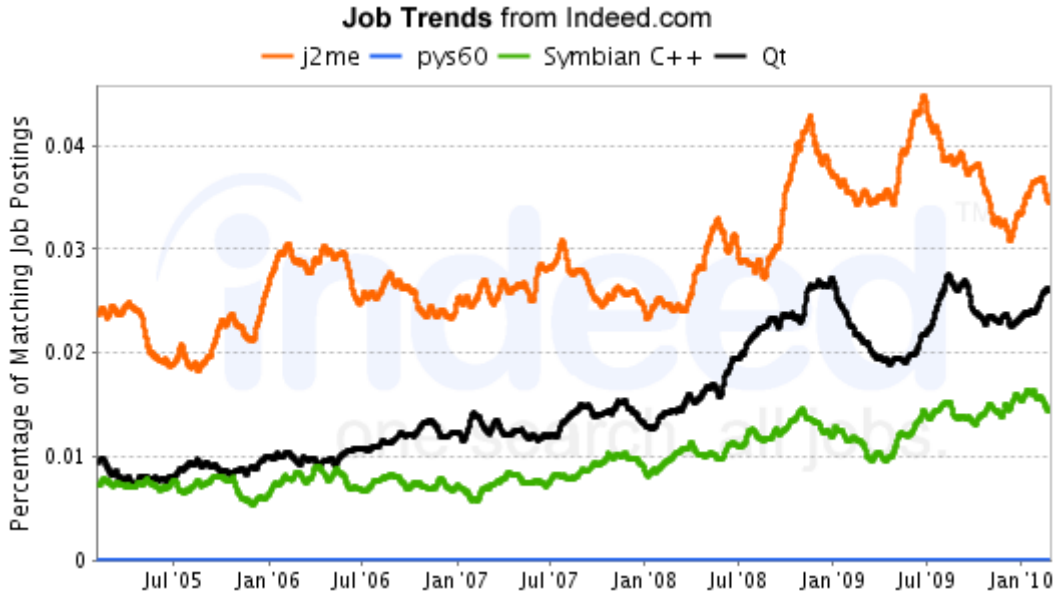

Figure 21 - Job trends for different mobile development platforms (Indeed 2010).

2.5.1. Conclusion

Considering what I have discussed before, I have decided to use J2ME. It presents a good balance between GUI control and performance, meaning that it is quite simple to make a user interface and at the same time the application runs fast.

At this point, I have introduced my initial idea for my system and presented some of the existing systems. Hereafter, I have introduced a flexible and elegant architecture (SOA) and introduced the: embedded devices and mobile devices. However, I have chosen not to use a mobile device in this project. In the next section I am going through how the system could communicate within components, i.e. which wireless protocols I have available and which ones are more suitable for my thesis work.

2.6. Communication protocols

Since the beginning of the internet in the 1970's many things have evolved; how we connect between each other has also changed. And in the last decade we have seen a great evolution of wireless communications. Wireless communications have revolutionized our lives and the way the internet is accessed. Of course the existence of wirelessly can only be attributed to the IP protocol. The emergence of IPv6 and LTE will allow for mobile devices to be assigned IP numbers as well as wireless high speed internet, respectively.

Thus, the communication protocols can be divided into different sections: by type (wired or wireless) and by range (short and long range). This section discusses different types of protocols and finally concludes with the chosen protocol.

2.6.1. X10 Industry Standard (Wired communication)

No project on home automation would be complete without mentioning X10 [6]. The X10 standard was developed in 1975 in Scotland by Pico Electronics in order to facilitate communication between electronic devices in home automation. It works *mainly* using the currently existing home wiring. The reason I say *mainly* is because a radio protocol exists that translates wireless radio packets to ordinary wired packets and sends them over the electrical wiring. This was the first technology of its kind and over the years higher bandwidth technologies emerged; most of which too use the homes existing wiring. Communication is achieved by modulating a pulse over the wiring from one connected device to another. The presence of a pulse indicates a binary 1 and absence a 0. Despite the newer faster technologies on the market today the X10 standard is still popular because it is wide spread, inexpensive and parts are available worldwide. All of which, no doubt, are characteristics of a formula for success.

A huge advantage of using such a system is that no addition wiring is needed, except for new hardware. However, there are certain significant drawbacks to this and similar but faster standards. Among the disadvantages, there are:

- ✘ Takes approximately 1 sec to transmit command data.
- ✘ It has poor propagation in split-phase electrical distribution (can be fixed by installing a capacitor between the phase wires).
- ✘ Affected by line noise from other devices.
- ✘ Receives interference from other X10 signals in the building (X10 lacks data encryption and signals may be used by devices with duplicate addresses).

It is also an old technology and, if trends follow the way they have been, will soon be replaced by wireless technology. I have chosen to use a wireless technology. It is however interesting to know this method works and is inexpensive.

2.6.2. RFID Standard (Wireless communication)

Among the wireless communications, we have Radio Frequency Identification [33]. This technology has many different proposes, but one of the most known is to identify objects, i.e. replacing the old fashion barcode. RFID systems are composed by three different components:

- The transponder, which carries the object identifying data, i.e. this component carries all the information belonging to one specific product.
- The transceiver, which reads and writes tag data. It means that this component can retrieve information from the transponder or even change that information.
- Records database, where information about the transponders is written. For instance, before the transceiver approves a certain transponder, it has to check first this database.

The RFID tag can transmit data up to 1000 meters, depending on its type. For instance, the most typical and cheap type is the passive one, which has a range of 10 meters. This one is completely "turned off" in absence of a transceiver. Also, a RFID module has the advantage of passing through obstacles, such as walls and cabinets. However, the RFID require to be designed for a specific application and they are sensible to environmental changes. Also, the mobile device platform made available for this project does not incorporate a RFID module, which will limit the usage of RFID in this project. Mainly because of this reason I have decided to not use RFID and go for other solution.

2.6.3. Wireless Personal Area Networks (WPANs)

The WPANs are intended for communication in short range. Among different standards, there are three familiar standards: Bluetooth, Wi-Fi (IEEE 802.11) and Zigbee (IEEE 802.15.4).

Bluetooth [16] is a wide-spread technology and present in the most of the mobile devices. Indeed, it has made its own revolution within the short range radio frequencies. First, it transmits at reasonable speed for short range, up to 721 Kb per second. Secondly, it is reasonably cheap, allowing many manufacturers of computers and cell phones to implement one Bluetooth chip on the device. Bluetooth, however, has difficulties in transmitting radio frequencies through walls or other obstacles.

One of the technologies promising to replace Bluetooth is Zigbee, which is affirmed to be cheaper and simpler than Bluetooth. Unfortunately, Zigbee is not the most common in the devices I have used in this project, so it was neglected from this thesis work.

Wi-Fi has revolutionized the wireless networks. It allows a user to connect to a LAN and consequently to the internet. Moreover, Wi-Fi has the advantage of being spread worldwide

and widely used communication standard. Wi-Fi has a range of transmission, up to 30m indoors and 90m outdoors, operating in ranges between 2.4GHz and 5.0GHz.

Finally, in terms of power consumption [27] and referring only to Bluetooth and Wi-Fi, Bluetooth has a clear advantage over Wi-Fi, consuming almost 10 times less power than Wi-Fi. This fact influences not only ecological issues but also the user electricity bill at the end of the month. Hereafter, Bluetooth becomes the natural choice to use in the sensor boards, since these have to plug-in to the house's electric grid.

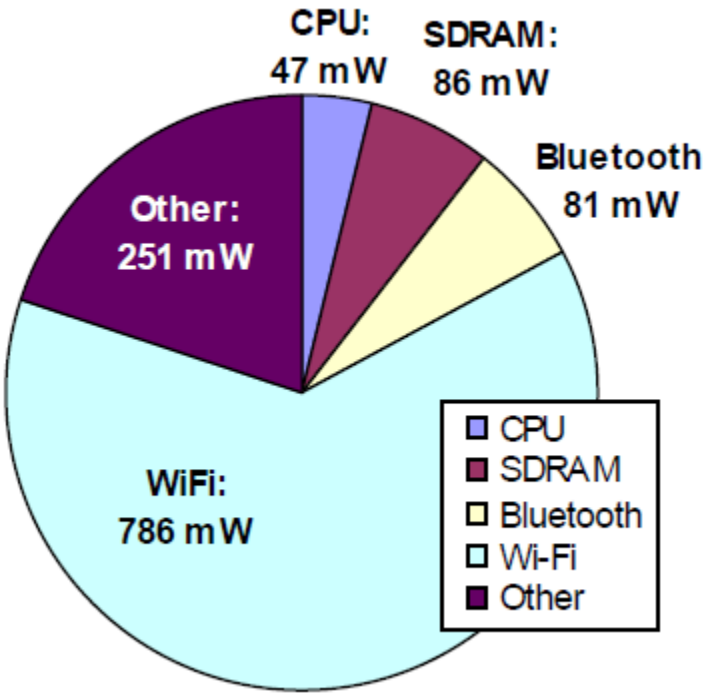

Figure 22 - Power breakdown for a connected mobile device in idle mode [27].

Neither Wi-Fi nor Bluetooth allow a user to access his house from anywhere in the world. This only can be achieved wirelessly using GSM communications (which is the standard for mobile communications) or wired with a local gateway to the internet.

2.6.4. 3G Mobile Communications (Wireless Communication)

As said before, 3G allows a device, mobile or computer, to connect to the internet from anywhere in the world, as long as there is an antenna and network provider signal. Being the standard in mobile communications, it allows for outdoor communication with the user's house. However, the use of 3G should be strictly limited to outdoors, since it has extra costs (data costs) to the user, unless the user has a 3G package with his network provider.

2.6.5. Conclusion

Let us remember that in this master thesis work, there are 3 main components: embedded devices, house entry point (a desktop server) and the web service. Bluetooth will be used to communicate between local desktop server and the sensor board.

During this analysis section I have discussed all the topics of the *SafeHouse* project: I have introduced the system, SOA and existing systems. After that, I have talked about embedded and mobile devices and finally in this last section about communication protocols. Hence, I present the problem formulation of my project.

2.7. Problem formulation

In section 2.2, the SOA architecture was described. This architecture allows us to compose a software system based on services. In the case of controlling a house, each service will be represented by an action that can be taken on a sensor in the house. This service can be requested by any device as long as it knows how to request it. The SOA architecture is further reflected in that the web service is aware of SOA services available and changes appropriately as stated by the SOA service. Similarly the mobile phone interface will do the same. Thus, it is important to define that the entire system is based on SOA architecture; it does not matter if we are talking about an embedded device, a mobile phone or a desktop application.

With consideration to these facts, the problem is formulated as follows:

The system should be as user friendly and intuitive as possible. It does not matter where, when and how the user accesses their home electronically. The user should be able to access the system installed in the house and either simply view the status of the system and its components or change it in a way that is relevant to him/her. The system should keep all users information private. *SafeHouse* should give the user the power of control; the ability to change the system in his/her own way without requiring any expert knowledge.

Hereafter, the main question of this master thesis is “How can a smart house system be made based on services capable of being accessed from anywhere in this world?” The novelty of this research is the design of a product system that is cost effective, energy efficient and easy to install, maintain and operate.

3. Design

In the previous chapter I have discussed different topics, from Service Oriented Architecture to Communication Protocols. I have analyzed different systems and technologies important for this master thesis. Finally, I made my problem formulation.

Here are the topics that will be covered in the design section:

- SOA architecture applied to the *SafeHouse* system.
- The different applications implementing SOA: sensor board, desktop application and the mobile application.
- The GUI Design for the mobile application.
- Management of Rules/Restrictions for the house.

Hence, this chapter will start to refer to the **Architectural Choice** I have made for my project: which components are in my system from one point to the other. Following this idea, I will introduce the **Communication protocol** I have integrated in my solution. At this point, I will introduce the **configuration of the system**, i.e. the procedures in order to install the application from scratch.

Hereafter, I will introduce the three main applications of the system: **sensor board application, desktop application** and **Web service application**. At the end of the Design chapter, the reader will have a deep knowledge and understand how the system works.

3.1. Architectural Choice

The most important idea of my system is to allow a user, at anytime from anywhere in the world, request status information from his automated home. Upon a user request, the system responds with the desired data almost instantaneously.

In section 2.2 I have discussed the power and elegancy of a Service Oriented Architecture. This architecture can be applied to any system I need, but for that I need to focus on services and not only on objects or classes as I usually do. Nevertheless, in the section 2.3 I have introduced some of the existing systems for smart homes. Analyzing those systems and knowing that I intend to implement a SOA for smart homes, I have decided to follow three-point architecture:

1. The embedded layer, i.e. where the components are connected to a micro controller and to a communication interface, such as Bluetooth UART.
2. The application layer running on an entry point of the house. This is the mean of connection between the user (web browser) and the embedded layer.
3. The presentation layer running on a cell phone where the user has the power, in a glance, to retrieve the current status of the house components and change them.

The most powerful argument of this type of architecture is that I do not need to configure again application or presentation layer if the user wants to change components in the embedded layer. Obviously, the embedded layer has to be configured again, but that is a small change that does not affect the rest of the system. The application layer has more than one service running at each time, at least two: one to communicate with the embedded layer and one for each presentation layer that connects to the system. To understand this scheme of three layers better, let us consider the following diagram:

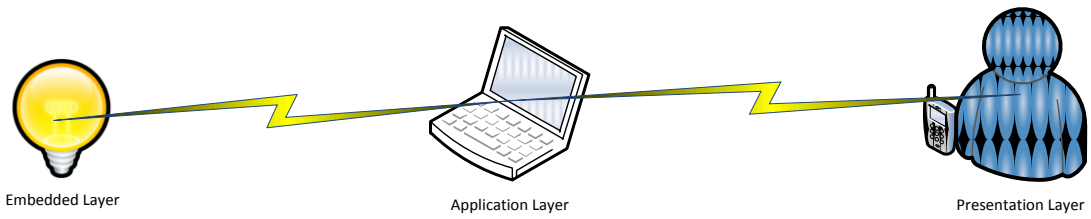

Figure 23 - Three layers architecture of the SafeHouse system: The lamp on the left represents the embedded layer; the laptop on the middle represents the application layer; and the user on the right represents the presentation layer.

The main actor of this system is the user and to allow him/her easy access to their house, the user has an application installed in his/her home computer, able to communicate with different communication standards. But this application is not standard for any specific house, meaning that the user can communicate with different houses and not change anything in the application. The key for this to happen is exactly SOA: since the architecture is based on services, the user retrieves the information of his house at the first place. With this the user can have control of two or more different houses without changing any programming detail on his device application. Moreover, the user can change or check in a glance any status of a

component in his house. This will become clear in the GUI Design and web service application sections later.

In other hand, I have the application layer: this layer is specific for each house and provides the services for the user in order to change or check status from the system. Also, it provides an extra service for a management menu on the desktop where the application layer is running. This layer will become clearer in section 3.3 and 3.6.

Finally, I have as well the embedded layer: this layer is responsible to receive the information from the application layer, make the changes in the components or retrieve status and send back and send an answer back to the application layer.

Hereafter, and in a house point of view, I intend that each embedded layer represents one division of the house, each application layers represents a house and each presentation layer one of the tenants/owners of the house. In the next chapter I are going through the communication protocol in the SafeHouse system: how the different layers interact with each other without going to a deep level of each application, meaning that the focus will be on the format of inter connection of applications.

3.2. Communication Protocol

I have introduced my architecture in a general point of view in the previous section. But a question remains till here: how each layer communicates with the other layers? Each format is used to represent the information passed back and forth? This section describes the design of the communication protocol.

It contains two different sub sections: Embedded-Application layers communication and Application-Presentation layers communication. The reason for this segregation is because the information changed between each other is different and there are specific details from each side to deal with.

3.2.1. Embedded-Application Layers communication

The embedded layer possesses different components integrated to the micro controller. Each of those components has specific properties: for instance, a representation for a led should be a Boolean (on/off) but a representation for the temperature sensor should be a float. Other thing to take in consideration: there are components which the user cannot change the status, he can only retrieve; for example, the led can be changed its status but not the temperature sensor (the user can change the temperature only by turning on or off the heating or opening or closing the window). Following this logic, the application layer has to send in the beginning a query to retrieve which components the embedded layer possesses and which methods can be used of each of those components. Also, it retrieves only primitive type of data: Boolean, Float, Integer, etc. The figure 20 shows the first information changed between the layers.

The reader can see some properties of each component:

1. The attribute **id** of the component: an identification tag for that component in that specific embedded layer.
2. The attribute **id** of the type: this goes to identify which kind of data can be retrieved from this component, either a Boolean or a Float.
3. Inside the node **METHODS** there are two possible child-nodes: **GET** and/or **SET**. If GET appears in that node means that the application layer can retrieve the status of that component. If SET appears in the node means that the application layer can change the status of that node according to the type retrieved previously. For this project the information will not be encrypted due to time constraints.

```
Application Layer -> Embedded Layer
-----------------------------------
<QUERY/>


Embedded Layer -> Application Layer
-----------------------------------
<COMPONENTS>
    <COMPONENT id="LED0">
        <TYPE id="bool"/>
        <METHODS>
            <GET/>
            <SET/>
        </METHODS>
    </COMPONENT>
    ...
    <COMPONENT id="TEMP0">
        <TYPE id="float"/>
        <METHODS>
            <GET/>
        </METHODS>
    </COMPONENT>
</COMPONENTS>
```

Figure 24 - First XML exchange between embedded and application layers.

The application layer, after storing this information, can send commands to the embedded layer. These commands can be either GET or SET. Moreover, the commands can be sent separately or together, meaning that the application can send a simple request (one command) or a composite request (more than one command).For a further understanding of these two concepts the reader can review section 2.2 about SOA. In the Figures 25 and 26 is shown the different types of requests.

Nevertheless, if in some case the XML is not valid or not recognized by the embedded layer, this last one sends a Non-Acknowledgement back to the application layer. The same principle applies for the rest of the system. In the next section I will present the communication between application and presentation layers.

```
Simple Requests
---------------
<SET id="component_id">value</SET>

<GET id="component_id"/>


Simple Answers
--------------
<SET id="component_id">value</SET>

<GET id="component_id">value</GET>
```

Figure 25 - Simple Requests from Application Layer and Simple Answers from Embedded Layer.

```
Composite Requests
------------------
<REQUEST>
    <SET id="id_1">value</SET>
    ...
    <GET id="id_N"/>
</REQUEST>

Composite Answers
-----------------
<ANSWER>
    <SET id="id_1">value</SET>
    ...
    <GET id="id_N">valueN</GET>
</ANSWER>
```

Figure 26 - Composite Requests from Application Layer and Composite Answers from Embedded Layer.

3.2.2. Application-Presentation Layers communication

The way how the application and presentation layers communicate between each other is in many senses similar to the previous section. Special attention goes for the security in the requests and answers between these two layers. For instance, each request from the presentation layer has to incorporate a pair username-password in order for the application layer to not accept every single request from the “outside” world. Without this, the system would be pruned to failure.

3.3. Configuration of the system

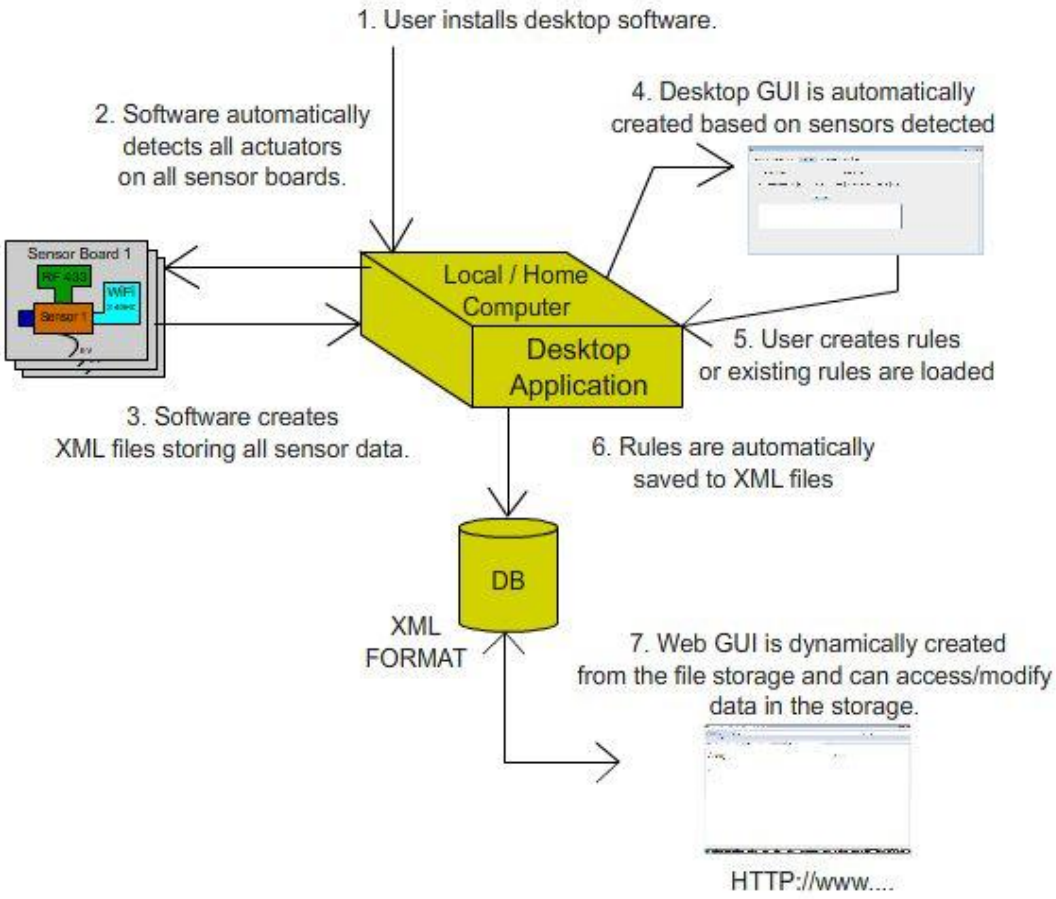

Figure 27 – System configuration

The diagram above serves to show how the system is configured by the user assuming the sensor boards are physically installed. User interaction is required in step 1 to initiate the software; and again in step 5 to create rules. Rules can also be loaded from memory. *Once activated the rules will hold true until a user-defined action is detected upon which a user-defined output will occur.* Theoretically the systems life cycle never ends and the program is constantly iterating through an infinite loop. In other words if the desktop application is not running the system will not be working. However the system is built such that it will automatically resume the last defined configuration upon running the application.

3.4. Sensor Board Application

The sensor board application or embedded layer is responsible to manage the different components integrated on the embedded device, the microcontroller pic30f3013. The sensor board application was design as part of another project and does not fall part of this project in a significant way.

Its services were however utilized in the creation of this final project so will be very concisely described below.

Essentially it consists of byte code that is interpreted from Assembler to C/C++ into XML strings that are sent via Bluetooth to a connecting Bluetooth device.

Thus the sensor board provides a service to all Bluetooth connecting devices as long as the requester of the service knows how to access the service.

For the requirements for this project the desktop application connects to the sensor board application and further processes the information received.

3.5. Desktop Application

To understand what the system needs to do we need to understand the concept of the *rule.* I have defined a rule in the introduction section and will define it again below.

3.5.1. Rules

The creation of the rules remains an area where the combinations of possible rules are only limited by the number of sensors involved. Needless to say there are many possibilities. The system is built such that any sensors installed will be detected along with their associated methods. These will be added to a list and in the GUI, allowing the user to use the sensors in his/her configuration. All sensors should be aptly named to ease usage of the system.

Rules creation process:

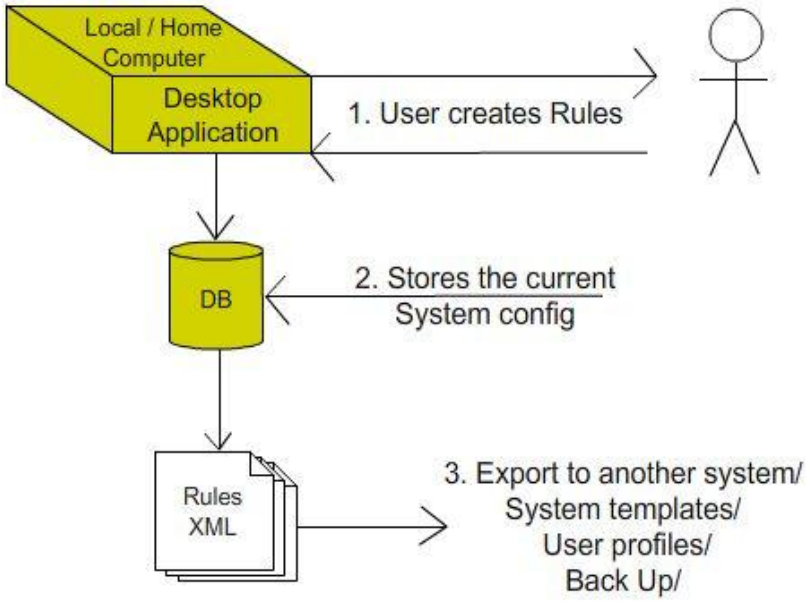

Figure 28 – Rule creation process.

Rules created by the user via the GUI are saved in the file/database system. These rules are saved in XML format and can be exported to another system as the diagram illustrates. The system automatically loads any rules from files placed in the appropriate folder. Even though the new rules are the loaded the user still has to activate the desired rules.

3.5.1.1 Rules Problem:

I define a rule as: “any input received from a sensor that triggers an action”.

It is a condition that if satisfied performs a certain task. In this case the rule only includes one input action for which it performs one output action. Simply put a rule tells the system what to do if a condition has been met.

Variables in a rule that need to be defined:

The input/trigger sensor.

What the input is/what happens to indicate input.

If input occurs what device is the action device.

What to do with action device.

For example: If motion detected then start alarm.

This can be broken down further: If (action) then (action). One action can be broken down further into two: (object) and (status). (Object) being the sensor and (status) being the status of the sensor. Again, to repeat, this rule accepts only one trigger and performs one output action.

So I can say:

IF (object) IS (status) THEN (object) IS (status).

This might make more sense:

IF (motion) IS (detected) THEN (alarm) IS (on).

So I can say one rule consists of two objects and two statuses. These are the four elements with which I can create the simplest most basic rule. The four elements to the rule are important to understand the execution of the rule and how it is possible to further expand the rules to create numerous, compound, multiple input, multiple output, timed rules.

This creation process is further documented below.

3.5.1.2 Compound rules

Although this system is only configured to handle single rules (one-input and one-output) it can easily be configured to handle multiple-inputs and multiple-outputs. This section presents a possible implementation for compound rules.

The example from above:

IF (object) IS (status) THEN (object) IS (status).

We can compound this like so:

IF (object) IS (status) THEN (object) IS (status).

AND/OR

IF (object) IS (status) THEN (object) IS (status).

AND so it repeats.

Similarly this could be done to handle one input-multiple outputs.

IF (object) IS (status) THEN (object) IS (status).

AND/OR

THEN (object) IS (status).

AND so it repeats or vice versa.

It is at this point that it is crucial to note that implement this functionality will complicate the design of the GUI. More buttons and options will need to be added that will increase the cognitive load on the user; further requiring more training of the user.

It is also at this point that I should mention that theoretically the user can still create multiple-input and multiple-output rule with this project as is. This requires the user to infer these rules and unfortunately requires some level of intellect to be used. The practical implementation of compound rules is explained below in the rule execution section where it is most relevant and better understandable.

3.5.1.3 Rule execution

When the system is activated it essentially enters into an infinite loop. Essentially a quality found all around us that has made computers what they are today. The infinite loop that runs in this program consists of a *rule-queue*. This loop is constantly checking a list of rules as read line-by-line from a list in a text XML file. Rules can be added and removed on-the-fly, in real time to this text file. Special consideration is given in the rule creation process to ensure that duplicate rules are not created. This loop is illustrated in figure below:

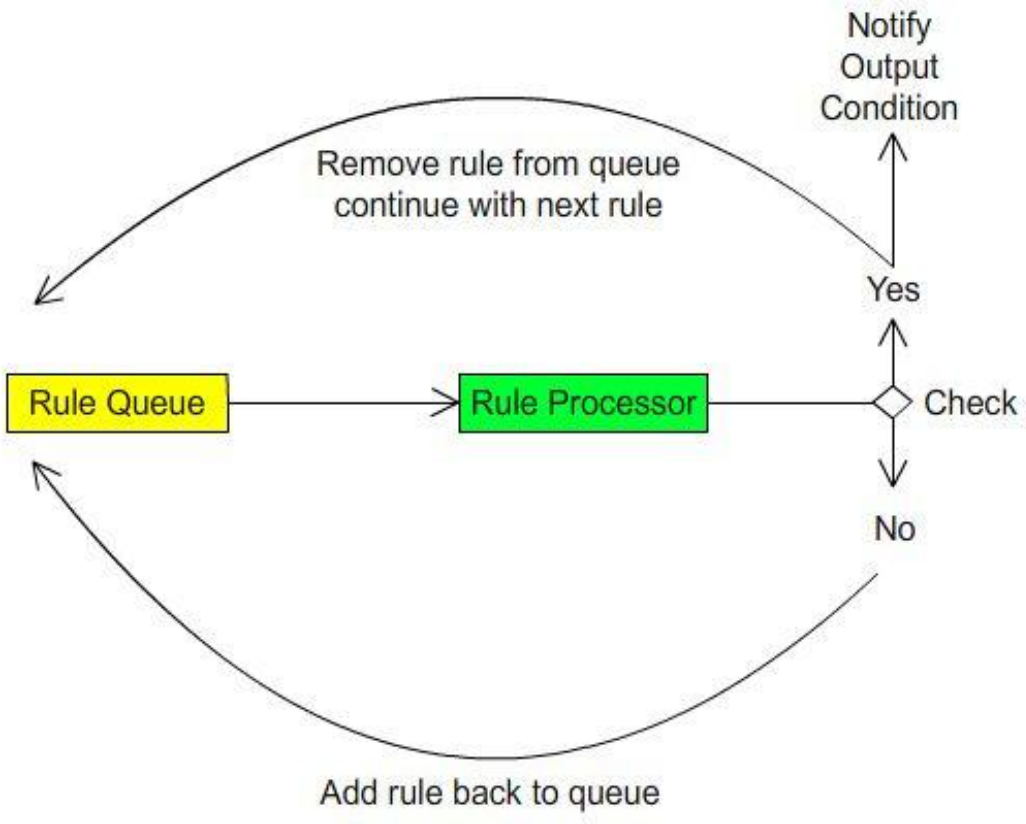

Figure 29 – Rule execution process.

As stated above the rules are executed sequentially. A *rule processor* is used to check to see if a condition is true or not. If nothing has happened the next rule is executed. If a condition is met then the output action should be performed, the user should be notified by email, the web application should display some notification, the rule should be removed from the *rule queue* and saved to be used again later if the user wishes to reset the rule. The system executes the rules sequentially which allows for compound rules to be created. A rule can be created multiple times such that the same precondition can have multiple different output actions and vice versa.

3.5.2. Desktop Application Logic

The Desktop application is designed such that it is as automated as possible; requiring minimal user input. This is especially true for the hardware installation of the sensor boards and their accompanying sensors. All the user needs to do is start the program. The program then should take a small amount of time to detect all the sensors, create all the necessary start up files and create the GUI for the user.

System execution

In more detail below are the steps of the system execution:

1. Start software
2. Auto discover sensor boards
3. For each sensor board detect each sensor
4. For each sensor detect each method
5. Create a file of all sensor boards
6. Create a file of all sensors
7. Create a file of which sensors are on which sensor board
8. Create GUI based on available sensors.
9. User programs “rules” consisting of start and stop conditions/actions/outputs/timers/alarms
10. System begins operation
11. Constantly checking to see if rule is broken
12. Constantly checking to see if hardware is added/removed/failure
13. Handle “broken rules” as pre-defined by the user
14. Handles new hardware by adding it to the GUI
15. Handles removals/failure by notifying the user as alarm
16. User needs to reset the “rule” if it is “broken”

Alarms

We define an alarm as “the output action that occurs when a *rule* is *broken”.* For example in a security situation if intrusion is detected then some kind of notification should be sent to someone. Email notification has been chosen to be used to notify the user/s. The user of this system has the option of adding numerous email addresses to which notifications would be sent. Notifications can also be seen live in the web GUI. This process can further be expanded to support SMS notification. SMS notification is not implemented or discussed any further.

Auto setup

Auto setup is handled dynamically as much as possible. The user still needs to initiate the system and define “rules” for execution. However, the auto setup designed in the system should handle all the tasks required to allow the user to use the software as easily as possible. That means automatically detecting the hardware and automatically creating the GUI and saving user settings.

File structure

List checkers and file checkers are used extensively to ensure the data stored is current, precise and duplicate-free. Erroneous data can trigger false alarms or even completely corrupt program execution. All XML files need to be well-formed. Data stored in the file system is also used to update components in the desktop user GUI as well as the web GUI. This project uses text files to store data however it can be improved by using a database such as SQL. This will improve data access and efficiency.

3.5.3. Desktop Application GUI

The desktop application GUI proposes serious difficulties and pitfalls that need to be considered. In designing this GUI I need to conform to GUI semiotics that is commonly used. This increases the probability that the user will be familiar with certain elements and graphic layout in the GUI.

Screens

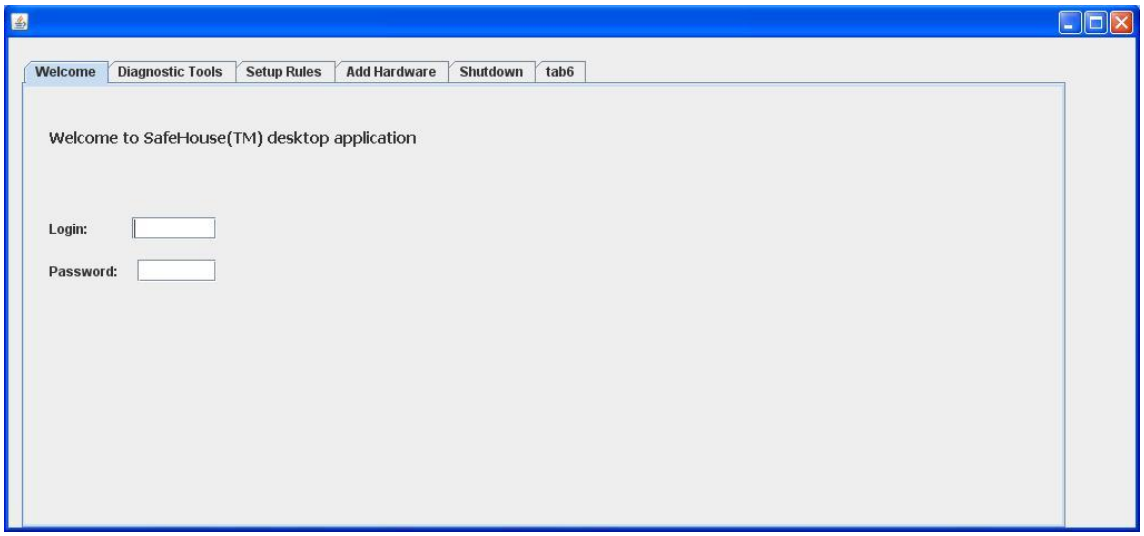

Figure 30 – Desktop login screen

The GUI is structured using a tab layout. This type of layout is frequently seen in modern GUIs. This is good because the user should be familiar with it. Following standards the user is greeted with a log-in screen. After successful login the other Tabs of the GUI should become active. After login the system will begin the auto-detect process.

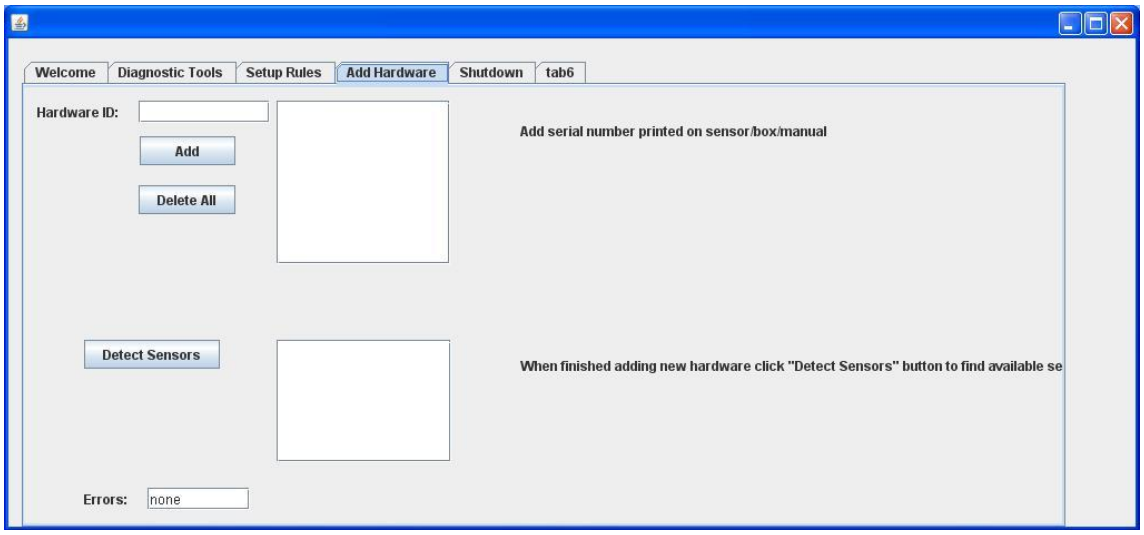

Figure 31 – Desktop hardware setup screen.

Even though hardware detection should be automatic it can also be added manually through the “add hardware” tab.

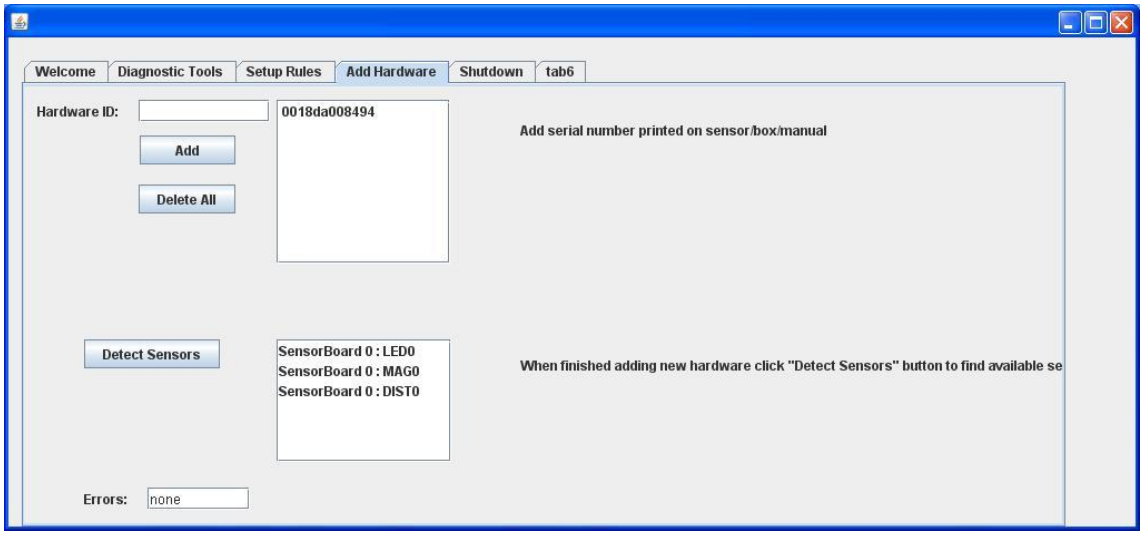

Figure 32 – Desktop hardware setup screen with data.

First the user needs to add the hardware ID of the sensor boards then the program detects what is available of which sensor board. This process is used only for demonstration and testing purposes.

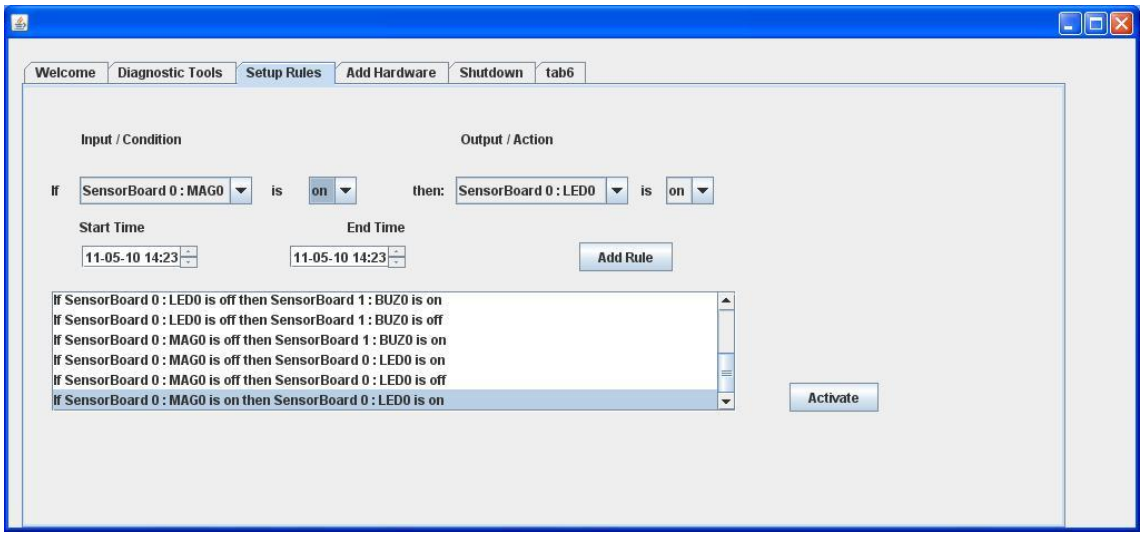

Figure 33 – Desktop rule setup screen

The “setup rules” tab is the most important. The problem is 2 fold. Firstly how the rules should be created and secondly how the user should be given control of the rules in a way that is simple and self-explanatory yet, at the same time, offer the full robustness of system capabilities.

This desktop GUI is not the primary user interface however can be used to configure the system manually if troubleshooting is needed. This GUI still remains relatively simple and easy to use.

3.6. Web application

The purpose of the web application is to be the central point of access to the system for the user. It is depicted in the screen shots below.

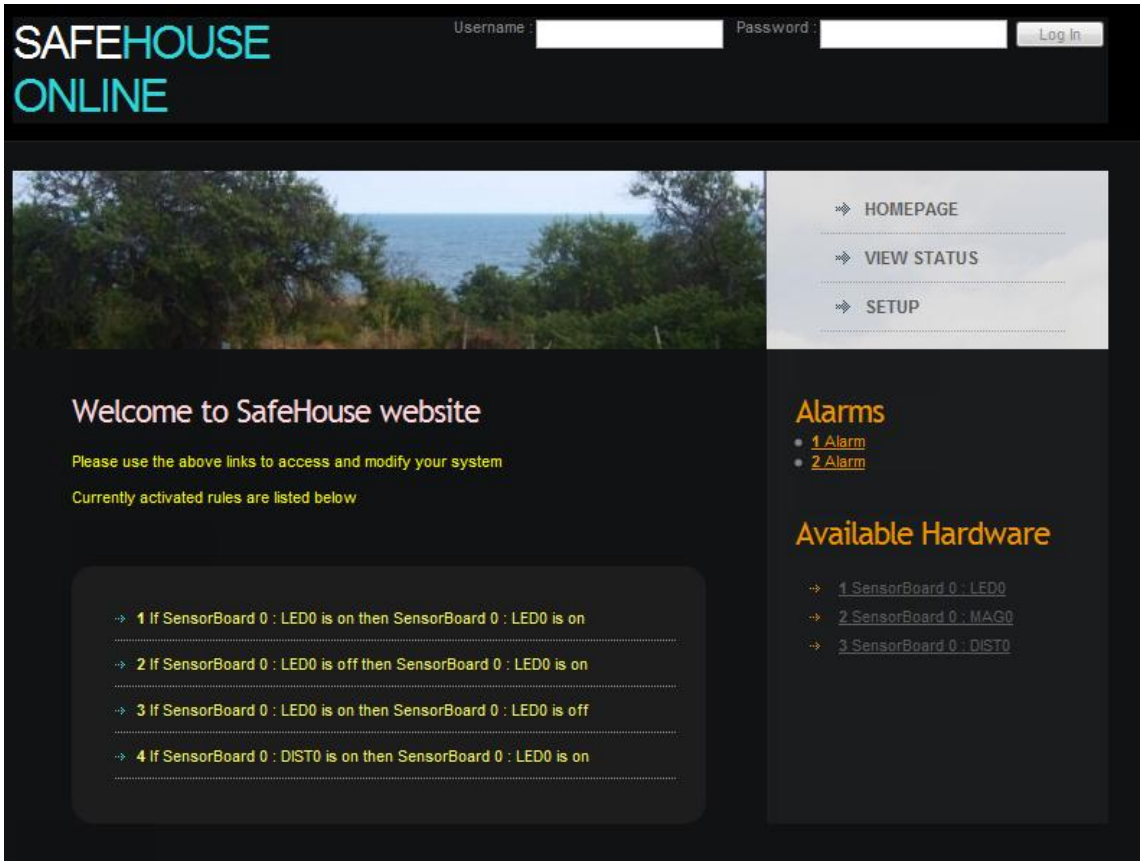

Figure 34 – Web home screen.

Figure 34 is the first page the user sees once they are logged in. If not logged in the user will not see any information detailing sensor status or alarms.

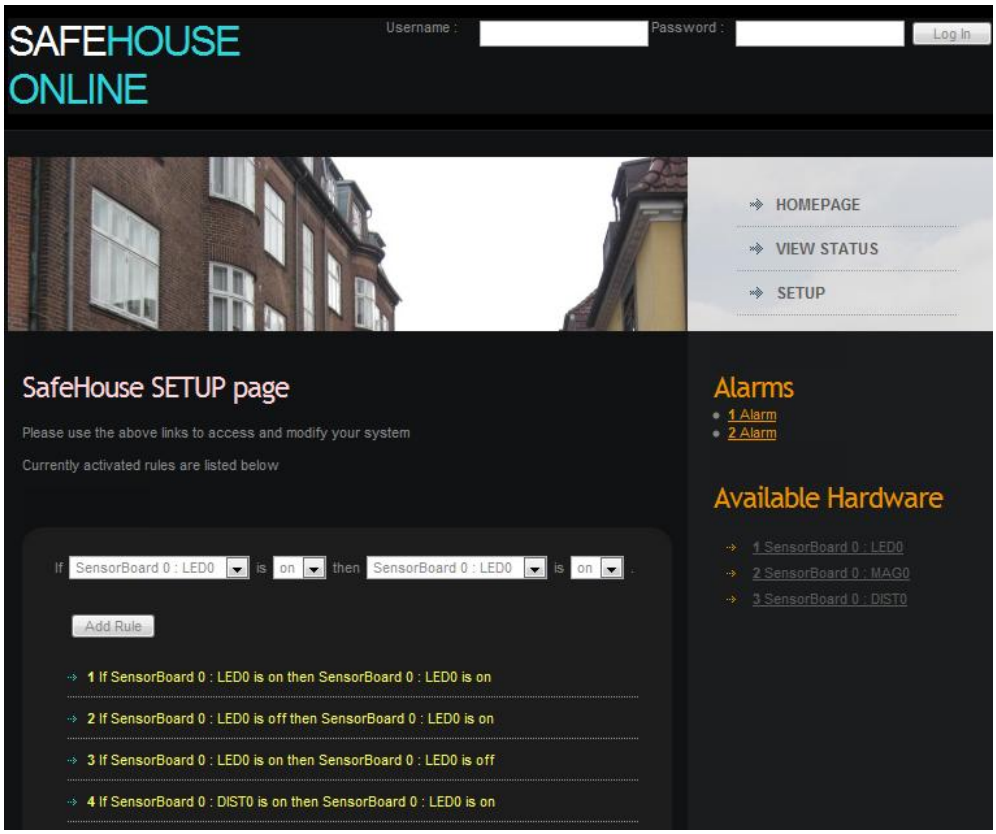

Figure 35 – Web setup rules screen.

Figure 35 shows the setup rules page on the web GUI. The purpose of this web page is to allow access to u user from any browser anywhere.

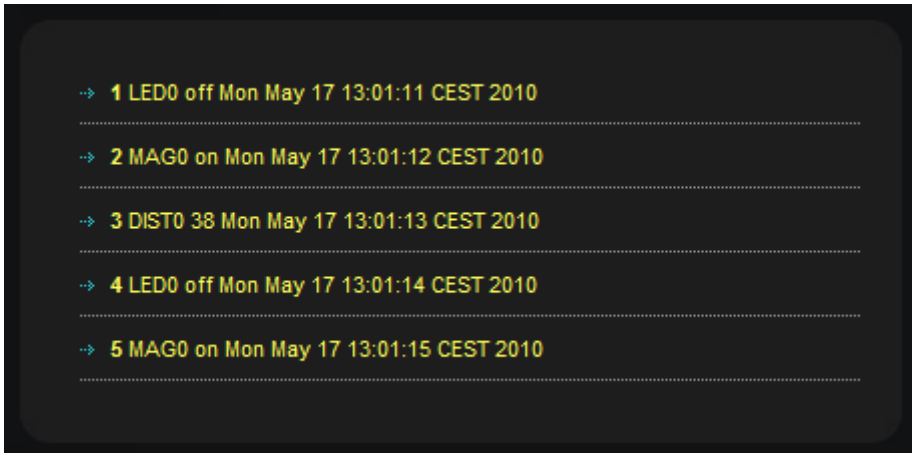

Figure 36 – Web view of notifications.

Figure 36 shows a log of information with a time stamp. This log expands as the system runtime increases and only select information should be shown to the user. The user should also be allowed to request information from a certain day.

Conclusion

There is no limit when it comes to designing GUIs. There are many solutions to the same problem. The purpose of this Web GUI it to demonstrate how the third layer (the web service) in the SOA architecture accesses the information initially sent by the first layer (the sensor board service), through the second layer (the desktop layer). This it achieves.

Implementation

XAMPP

XAMPP Apache server was used to host the PHP code.

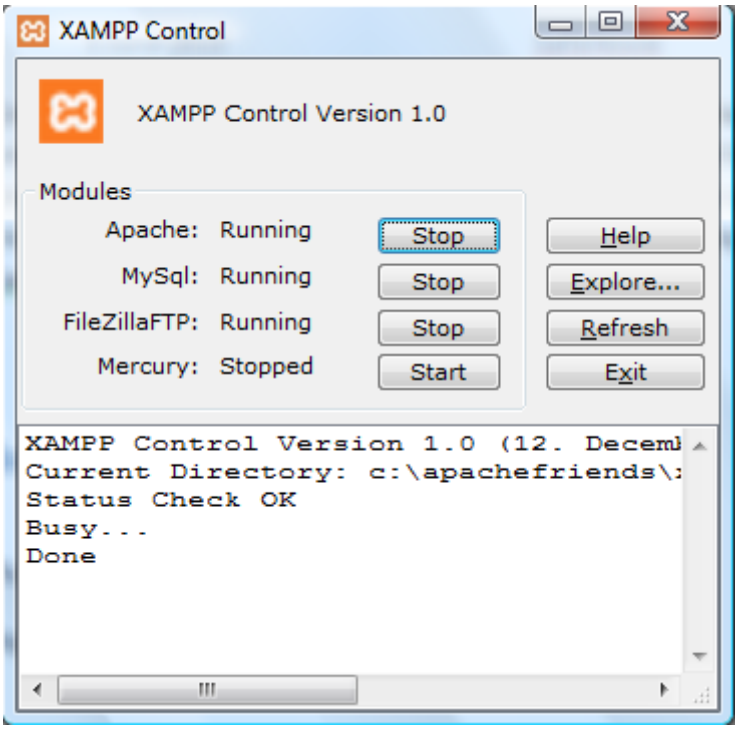

Figure 37 – XAMPP control panel.

This provides an efficient way to create a localhost that can open port to allow external access to the hosting machine. For this project only localhost was used for demonstration purposes and no external port was opened.

NETBEANS IDE

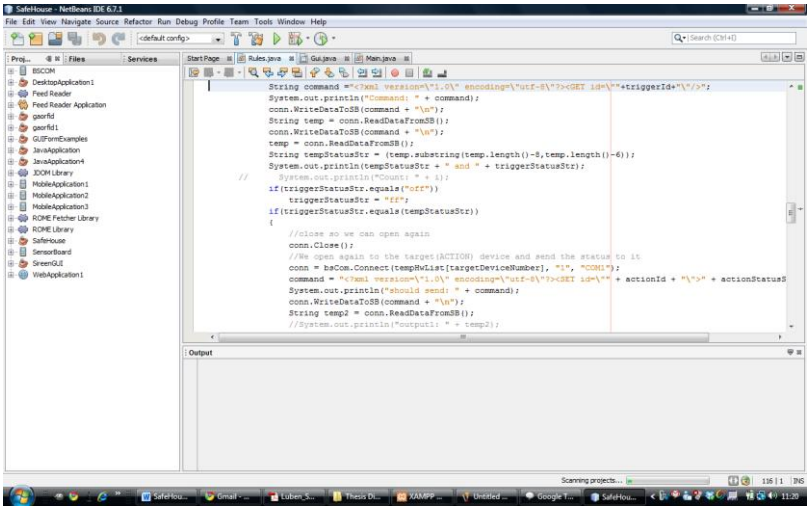

Figure 38 – NetBeans IDE

The NetBeans IDE was used to develop the java code and desktop GUI. NetBeans offers a good interface for designing GUIs in java. The drag-and-drop interface is a very powerful tool for designing GUIs, especially when development time is limited. Eclipse is an alternative interface however it does not offer a GUI designer as powerful as NetBeans and it is for this reason that NetBeans was chosen.

Code Design

In order to realize this project it had to be programmed somehow. This section describes what programming languages were used and how they were connected.

Java was used to create the desktop application. The java program stores all of its information in XML and plain text files. These files are constantly being read and written to. This acts as the main suppository of data or brain. Once this data is processed it will become information which is useful to the user. Any presentation layer that reads this data has to process it in a way that can be considered *relevant* to the user. There are many ways to interpret this information which gives the designers of the presentation layer much flexibility during their design phase. For example the course granularity and high density of the data allows it to be entirely *dumped* on the users' displays. However it can and should be *cut-down* to display that which is considered relevant. I have chosen to design and implement the presentation layer as a web service.

In order to create this service I choose to use Apache web server and PHP to aid in the hosting and HTML creation. As stated above this presentation layer has to interact with the lower application level. There are 2 ways of doing this. One way would be to give the PHP code

access to the already existing Java code and functions. The other is achieved by giving the PHP access to the data files. The former way has the benefits of not having to duplicate code in PHP however requires significant research into the coordination between the Java runtime and PHP server. Instead the latter, less elegant, yet; failsafe method was used. This would not only ensure a timely completion of the project that would be seamlessly integrated, bug-free but most importantly achieve the goals that this project outlines.

Thus the Apache server was used to create a local PHP server.

On this server the Java application was given read/write privileges to a folder. This folder would be the brain of the system. This folder is also accessed by the PHP code that processes and displays the data in a way I have considered *relevant*. An open port will give secure external access to the PHP/HTML webpages that would display the information to the user (and allow user input to the system).

Coding problems

According to sources [35], [36] there is a way to call java classes and functions from PHP. This would greatly simplify and speed up the implementation of the PHP code but would require an amount of unknown research that could not be risked at time of implementation. Unfortunately this means that some code is duplicated on the PHP side that already exists on the Java side. This method of programming is somewhat redundant and not efficient however because of time limitations this method was used as it is guaranteed to work.

Testing

A certain amount of user testing was possible to be achieved during the duration of the project. This section describes how the testing was carried out in detail.

Goals of testing

The purpose of this testing was to deduce to what level this project achieve its goals with respect to the degree of measure of how intuitive the GUI is.

Testing procedure

3 participants were asked to interact with the program.

2 Male and 1 female.

Participants are in the age group from 22 up to and including 24.

It was explained to participants that their personal details would be kept confidential.

Participants were told that the system they were using is intended for home automation purposes.

It was explained to them, in layman terms, what the sensors were and what they do.

They were given the task of creating some rules and activating the system.

The users' interactions were recorded via a screen capture program.

Findings

In this sample gender did not influence the findings.

Because the test group was small no conclusive findings can be presented here but an initial result was found and is described below.

All users managed to understand the task and achieve it. However considerable time was spent in describing the concept of a rule to them and the sensor names.

User 1:

Setup 3 rules and activate the system in 2:20.

User 2:

Setup 3 rules and activate the system in 1:43.

User 3:

Setup 3 rules and activate the system in 1:55.

Conclusion

The findings, however sufficient, still show that the users managed to complete the tasks.

User feedback revealed that users found the GUI simple and intuitive.

Project Conclusion

This project set out to achieve certain goals and mentioned in the beginning.

It managed to reach the primary goal of creating a wireless platform for controlling sensors and actuators in multiple rooms.
The sensors can be controlled and monitored.
The system was developed using SOA.
A SOA-based web service application was created.
Usability is vital to this project. The test results, however limiting, still show positive feedback toward usability.
It would manage to give the user access from a mobile web service if a port were open. This remains an area for improvement.
Another area for improvement would be to add data encryption to the transmitting data.

During the life time of this project the following lessons were re-learnt:

Timely planning is essential to project success.

Constant changing of ideas can be unproductive.

There are many ways to get a job done.

There is no substitute for hard work.

Consideration and respect needs to be given to all members involved.

Co-ordination is vital between all members.

Conflict should be resolved in a manner that is work appropriate.

This project is a noble attempt at creating a home automation system. Though it is a step in the right direction it is still far too simple to compare with industry products made by companies such as Sony or Microsoft. It still manages to demonstrate a certain level of understanding and knowledge.

Appendixes

Appendix A

As I have stated in section 2.4.1., I have gone through some solutions to solve the problem of linearization with the distance sensor Sharp GP2D12. The first approach I took in account is represented in the following picture. I have used a 1st grade linear regression in the output signal. With this the signal becomes linear, but pruned to huge deviations as I can see. Only around 1V and 3,75V the distance respects the real value.

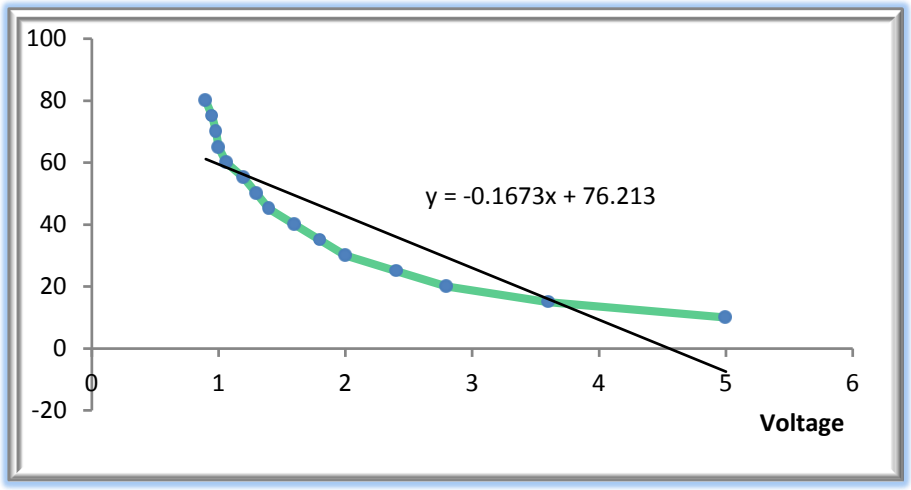

Figure 39 - Sharp GP2D12 non-linear and linear graph.

As referred in that section, I could have used a 2nd grade regression and with this the problem will become solved, or at least, the output would be really approximated to the real distance value. However, for a microcontroller as the pic30f3013 this would have become a heavy solution for a light CPU.

Thus, I had two options: either I store the values on the EPPROM of the microcontroller or I make sub-linearization. The first solution lacks of completeness: it is almost impracticable to make a table with every single input value. With this, I have divided my output into three different intervals and applied a 1st linear regression (as stated in that section). With this I achieve quite approximate values, as I can see in the following table.

Without having the necessity of being 100% precise, I have decided to use this last option.

Volts (x100)	Distance (in cm	Linear Distance (in cm	Absolute Error (in cm
90	80	75,482	4,518
95	75	72,216	2,784
98	70	70,2564	0,2564
100	65	68,95	3,95
106	60	65,0308	5,0308
120	55	55,886	0,886
130	50	49,354	0,646
140	45	43,919	1,081
160	40	39,865	0,135
180	35	35,811	0,811
200	30	31,757	1,757
240	25	23,222	1,778
280	20	21,046	1,046
360	15	16,694	1,694
500	10	9,078	0,922
			1,81968

Figure 40 - Comparison betIen different output data in the Sharp GP2D12. The green cell is the average of the absolute error.

Appendix B

In section 2.5 was referred that I have used libraries developed previously during Sérgio Pedro's internship at PDM Technology. In this appendix I am going through which libraries I am using: their intent and why I have used in this project.

1. XML Library

The main propose of this library is to make the XML easy to be parsed and easy to transform into an object (XMLObject). Then, the object can be acceded in a simple way and the developer can perform different actions over it: create, update and delete. Let us say that I receive a string representing the XML called **received**. Thus, I can parse that string by simply doing:

```
Parser parser = new Parser(received);
```

```
XMLObject xmlObject = parser.getXMLObject();
```

Of cmyse, the string has to respect the XML format, i.e. the XML has to be valid otherwise an InvalidXMLException will be thrown. At this point, I have my XML object loaded. Let us say that I receive another string, I parse it into a new XML object called **xmlObjectUpdate** and now I want to update my XML object in memory. To do that I simply make:

```
XMLNode rootNode = xmlObject.getRoot(); //to get the root of the XML
```

```
XMLNode rootNodeUpdate = xmlObjectUpdate.getRoot();
```

```
rootNode.updateNode(rootNodeUpdate);
```

Other actions can be taken like creation or deletion of elements in the xml. Since the *SafeHouse* system uses XML as format of inter connection between components, I have decided to integrate this library into my project.

2. BSCOM Library

Different frameworks implement in different ways the communication methods. Some frameworks, like J2ME make a quite standard way to use connectivity, but others not that much, like Android SDK. Following-up the idea of unifying different communication standards, the intent of this library was to make a flexible and easy way to create either a client or a server. With this, the developer can use either Internet (3G/Wi-Fi) or Bluetooth without taking care of implementation details. In the following code I explain how to open different kind of connections. For instance, for a client to open a connection and send/receive data the procedure is really simple:

```
BSCom bsCom = new BSCom(); //object containing the main methods

BSConnection conn = bsCom.Connect(ip,port,name); //opens connection

conn.WriteData("Hello");

String received = conn.ReadData();
```

To use Internet or Bluetooth the procedure is simple:

17. If the developer wants to use Internet, it has to give the ip and port of the server and leave name null.
18. If the developer wants to use Bluetooth, it has to give the mac address (ip), port and the name of service.

For a server the procedure is similar, except that instead of using Connect, it has to use the method Accept.

Since I have used during this project Bluetooth, 3G and Wi-Fi, it made all the sense to incorporate this library into my application.

Bibliography

[1] Acroname, Inc. "Sharp IR Rangers Information." *Acroname Robotics.* 2010. http://www.acroname.com/robotics/info/articles/sharp/sharp.html (accessed February 2010).

[2] Adão, Helder, Rui Antunes, and Frederico Grilo. "Web-Based Control & Notification for Home Automation Alarm Systems." *World Academy of Science, Engineering and Technology*, no. 37 (2008): 152-156.

[3] Alkar, Ali Ziya, and Umit Buhur. "An Internet Based Wireless Home Automation." *IEEE Consumer Electronics* 51, no. 4 (2005): 1169-1175.

[4] AMX. *AMX.com - It's Your World Take Control.* 2010. http://www.amx.com (accessed March 2010).

[5] Bieberstein, Norbert (et al). *Executing SOA: a practical guide for the service-oriented architecture.* Upper Saddle River, N.J.: IBM Press / Pearson, 2008.

[6] Burroughs, Jon. *X-10® Home Automation Using the PIC16F877A.* Microchip Technology Inc., 2010.

[7] Corporation, National Semiconductor. "LM35 - Precision Centigrade Temperature Sensor." *National Semiconductor.* 2010. http://www.national.com/mpf/LM/LM35.html#Overview (accessed February 2010).

[8] Department of Electronic Systems, Aalborg University. *Mobile Devices: opensensor.* 2009. http://mobiledevices.kom.aau.dk/projects/00/ (accessed February 2010).

[9] Forum Nokia. *Getting Started with the Nokia Qt SDK.* Nokia Corporation, 2010.

[10] Group, JSR118 Expert. *Mobile Information Device Profile for Java 2 Micro Edition.* Sun Microsystems Inc. and Motorola Inc., 2002.

[11] Huhns, M. P., and M. N. Singh. "Service-oriented computing: key concepts and principles." *IEEE Internet Comput.*, January 2005: 75-81.

[12] Inc., SmartLabs. *INSTEON - Wireless Home Control Solutions for Lighting, Security, HVAC, and A/V Systems.* 2010. http://www.insteon.net/ (accessed March 2010).

[13] Indeed. "j2me, pys60, Symbian C++, Qt Job Trends | Indeed.com." *Indeed.com.* 2010. view-source:http://www.indeed.com/jobtrends?q=j2me,+pys60,+Symbian+C%2B%2B,+Qt&l= (accessed March 2010).

[14] Josuttis, Nicolai M. *SOA in practice.* Beijing: O'Reilly, 2007.

[15] Khiyal, Malik (et al). "SMS Based Wireless Home Appliance Control System." *Issues in Informing Science and Information Technology* 6 (2009): 887-894.

[16] Klingsheim, André N. *J2ME Bluetooth Programming.* Master Thesis, Bergen: University of Bergen, 2004.

[17] Ltd, Living Control. *Living Control : Welcome.* 2009. http://www.livingcontrol.com/ (accessed March 2010).

[18] Ltd, Total AV Control. *Total AV Control - Home Automation - Multiroom.* 2009. http://www.totalhomecontrol.co.uk/ (accessed March 2010).

[19] McLean, Prince. "AppleInsider | Canalys Q3 2009: iPhone, RIM taking over smartphone market." *AppleInsider.* November 2009. http://www.appleinsider.com/articles/09/11/03/canalys_q3_2009_iphone_rim_taking_over_smartphone_market.html (accessed February 2010).

[20] Meyer, S., and E. Schulze. "Smart Homes and the Aging User, Trends and Analysis of Consumer Behavior." *Symposium Domotics and Networking.* Miami: Berlin Institute for Social Research, 2006.

[21] Nokia Corporation. *PyS60 Library Reference.* Nokia Corporation, 2008.

[22] "S60 5th Edition: What's New for Developers." Nokia, October 2008.

[23] Osipov, Maxim. "Home Automation with ZigBee." *The 8th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN).* St. Petersburg: Springer-Verlag, 2008. 263-270.

[24] Papazoglou, Mike P. "Service-Oriented Computing: Concepts, Characteristics and Directions." *Fourth International Conference on Web Information Systems Engineering.* Rome: IEEE Computer Society, 2003. 3.

[25] Pedro, Sérgio. *Orion Mobile.* Internship Report, Aalborg: Aalborg University/PDM Technology, 2010.

[26] Pérez, F. (et al). "Symbian C++ Application Programming Overview." ITACA Institute, Polytechnic University of Valencia, 2007.

[27] Pering, T. (et al). "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces." *4th Int. Conf. Mobile Systems, Applications and Services (MobiSys).* 2006. 220-232.

[28] Regnier, Thibaut. "MIDP telephones benchmark." *Le Club des utilisateurs Java.* n.d. http://www.club-java.com/TastePhone/J2ME/MIDP_Benchmark.jsp (accessed March 2010).

[29] Schmidt, M. -T. (et al). "The Enterprise Service Bus: Making service-oriented architecture real." *IBM Systems Journal* 44, no. 4 (2005): 781–798.

[31] Team, Fast Track. *Welcome to SmarterHome.* 2009. http://smarterhomesystems.com/ (accessed March 2010).

[32] Warmer, Cor (et al). "Web services for integration of smart houses in the smart grid." *Grid-Interop.* Denver, 2009. 207-214.

[33] Weis, Stephen August. *Security and Privacy in Radio-Frequency Identification Devices.* Master Thesis, Massachusetts: Massachusetts Institute of Technology, 2003.

[34] Xuguang, Huang. "An Introduction to Android." Dababase Lab. Inha Univeristy, November 2009.

[35] IBM, Information center, http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=/com.ibm.etools.mft.doc/ac69011_.htm

[36] Nold, M., Soeterbroek, j., PHP & Java http://www.phpbuilder.com/columns/marknold20001221.php3