

**FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS
FATEC PROFESSOR JESSEN VIDAL**

ARIOVALDO DE SOUZA JUNIOR

**INTERFACE FOR USER-ENVIRONMENT INTERACTION
USING OPEN SOURCE TOOLS AND INTERNET OF THINGS**

São José dos Campos
2012

ARIOVALDO DE SOUZA JUNIOR

**INTERFACE FOR USER-ENVIRONMENT INTERACTION
USING OPEN SOURCE TOOLS AND INTERNET OF THINGS**

Trabalho de Graduação apresentado à Faculdade de Tecnologia São José dos Campos, como parte dos requisitos necessários para a obtenção do título de Tecnólogo em Informática com Ênfase em Banco de Dados.

Orientador: MsC. Giuliano Bertoti

São José dos Campos
2012

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

SOUZA JUNIOR, Ariovaldo
Interface for User-Environment Interaction using Open Source Tools and Internet of Things.
São José dos Campos, 2012.
71f.

Trabalho de Graduação – Curso de Tecnologia em Informática com
Ênfase em Banco de Dados, FATEC de São José dos Campos: Professor Jessen Vidal, 2012.
Orientador: MsC. Giuliano Bertoti.

1. Áreas de conhecimento. I. Faculdade de Tecnologia. FATEC de São José dos Campos:
Professor Jessen Vidal. Divisão de Informação e Documentação. II. Título

REFERÊNCIA BIBLIOGRÁFICA –

SOUZA JUNIOR, Ariovaldo. **Interface for User-Environment Interaction using Open Source Tools and Internet of Things**. 202. 71f. Trabalho de Graduação - FATEC de São José dos Campos: Professor Jessen Vidal.

CESSÃO DE DIREITOS –

NOME DO AUTOR: Ariovaldo de Souza Junior

TÍTULO DO TRABALHO: Interface for User-Environment Interaction using Open Source Tools and Internet of Things

TIPO DO TRABALHO/ANO: Trabalho de Graduação / 2012.

É concedida à FATEC de São José dos Campos: Professor Jessen Vidal permissão para reproduzir cópias deste Trabalho e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste Trabalho pode ser reproduzida sem a autorização do autor.

Ariovaldo de Souza Junior
Endereço do Aluno
CEP 12237-828 – São José dos Campos - SP

Ariovaldo de Souza Junior

**INTERFACE FOR USER-ENVIRONMENT INTERACTION
USING OPEN SOURCE TOOLS AND INTERNET OF THINGS**

Trabalho de Graduação apresentado à
Faculdade de Tecnologia São José dos
Campos, como parte dos requisitos
necessários para a obtenção do título de
Tecnólogo em Informática com Ênfase em
Banco de Dados.

Antônio Egydio São Thiago Graça, MsC, FATEC

Fernando Masanori Ashikaga, MsC, FATEC

Giuliano Bertoti, MsC, FATEC

____/____/____

DATA DA APROVAÇÃO

Dedico o presente trabalho aos meus pais Ariovaldo de Souza e Luzia Aparecida de Souza, que tanto me ajudaram até aqui, sempre demonstrando muita paciência e agindo com amor.

AGRADECIMENTOS

Agradeço ao professor e orientador Giuliano Bertoti, pelo apoio e encorajamento contínuos pesquisa, aos demais professores, pelos conhecimentos transmitidos, aos meus pais e irmãs pelo amor e carinho, aos meus companheiros de jornada Ricardo Kramer de Oliveira Barros, Daniel Antônio Torres Cesário e Luis Guilherme Ungur que tanto ajudaram e motivaram durante o caminho. Agradeço também a Deus pela provisão durante o curso.

*“Whether you think you can, or you think you
can't - you're right.”*

Henry Ford

RESUMO

Todos os anos, desastres naturais de todos os tipos causam milhares de mortes e destruição ao redor do mundo. No Brasil o fenômeno natural que causa mais dano são as enchentes. Elas podem ser causadas por interferência humana, construções irregulares, acúmulo de lixo em bueiros, excesso de chuvas, dentre outras razões. Embora elas não possam ser evitadas na maioria dos casos, há medidas preventivas que podem ser tomadas para reduzir os impactos causados por este tipo de fenômeno. Um planejamento urbano mais elaborado, conservação das ruas de forma a evitar o acúmulo de lixo em bueiros e alertas de enchentes são alguns exemplos do que pode ser feito. Este trabalho visa desenvolver uma interface para interação do usuário com o meio ambiente, assim como seu monitoramento utilizando Python, ferramentas de HTML5 e o serviço de web Cosm para Internet das Coisas – alimentando seu banco de dados com os dados coletados pelo dispositivo Arduino. Como objetivo secundário, o desenvolvimento da interface foi mantido o mais simples possível, de forma a permitir sua reprodução por pessoas interessadas neste campo de estudos ao redor do mundo.

Palavras-Chave: Internet das Coisas, Arduino, meio-ambiente, monitoramento de enchentes, sistemas de alerta, interação humano-computador.

ABSTRACT

Every year natural disasters of all kinds cause thousands of deaths and destruction worldwide. In Brazil the natural phenomenon that causes more damage is flood. It can be caused by human interference, life irregular constructions, trash accumulation in sewers, excess of rain, and other reasons. Even though it cannot be avoided in most of the cases, there are preventive actions that can be taken in order to reduce the impacts caused by this kind of phenomenon. A better urban planning, conservation of the streets, so the water can run freely to the sewers, and flood alerts are some examples of what can be done. This work aims to develop an interface for user-environment interaction and monitoring utilizing Arduino, Python, HTML5 tools and the Cosm web service for Internet of Things - feeding its shared database with the data retrieved from the Arduino. As a secondary objective, the interface development has been kept simple allowing it to be reproduced by people interested in this field of study worldwide.

Keywords: Internet of Things, Arduino, environment, flood monitoring, alert systems, human-computer interaction.

SUMMARY

1- INTRODUCTION	11
1.1- Motivation	11
1.2- Objectives	12
1.2.1- General Objectives	12
1.2.2- Specific Objectives	13
1.3- Methodology	13
1.4- Work Structure	14
2- RELATED WORKS	15
2.1- Floods	15
2.2- Python Programming Language	16
2.3- The world connected through the internet	17
2.3.1- Internet of Things	17
2.3.2- Cosm.com	19
2.4- Data collecting hardware	20
2.4.1- Microcontrollers	20
2.4.2 – Arduino	22
2.4.3- Sensors	24
2.5- Web Applications	25
2.5.1- NoSQL Database - Couch DB	25
2.5.2- Programming languages for the web	28
2.5.2.1- HTML5	28
2.5.2.2- JavaScript	30
2.5.2.1- CSS3	31
2.5.3- Communication and data retrieving	33
3- SYSTEM DEVELOPMENT	36
3.1- Software Processes and Architecture	36
3.2- Socio-technical Systems and its Requirements	38
3.3- System Design and Software Development	39
3.4- The Environment Simulation Module	41
3.5- The Arduino device	42
3.6- The Main System	45
3.7- The Cosm component	48
3.8 - The Flood Alert component	49
4- Results And Discussion	51
4.1- Arduino Water Level Measuring device	51
4.2- The environment simulation module	53
4.3- System implementation	54
4.3.1- Main system	54
4.3.2- Arduino Capture	55
4.3.3- Flood Alarm component	56
4.3.4- Cosm component	57
5- FINAL CONSIDERATIONS	61
5.1- Contributions and Conclusions	61
5.1.1- Publication	63
5.2- Future Works	63
REFERENCES	65

1- Introduction

1.1- Motivation

Thousands of natural disasters cause massive destruction and deaths around the world annually. There are hurricanes (GOLDENBERG et al., 2001) (BUSINESS INSIDER, 2012), tsunamis (KANAMORI, 1972), earthquakes (SCHOLZ, 2002), volcanic eruptions (Werner-Allen et al., 2005), floods (CAMPOLO; ANDREUSSI; SOLDATI, 1999) (NICHOLLS; HOOZEMANS; MARCHAND, 1999) (SMITH; WARD, 1998) (THE TELEGRAPH, 2012), among others. These phenomena kill a great number of people and leave behind millions of dollars in prejudice. When it comes to the Brazilian scenery, it has few or none seismic or volcanic activity. There no registry that Brazil had been hit by a tsunami in the past. However, other types of natural phenomena become more frequent every year (DAVIS, 2009) (STOLTMAN; LIDSTONE; DECHANO, 2007).

It is speculated that the increase in the number of occurrences of certain types of disasters (like floods, hurricanes and fires) is a direct result of the global warming (COX et al., 2000) (HUGHES, 2000). The excessive emission of CO₂ provokes the greenhouse effect, which occurs when the infrared radiation that reaches our planet coming from the Sun cannot be reflected back to the space (STILLE; DAVIS; YOUNG JR, 2006).

Until not long ago there were no reports about climatic phenomena like hurricanes in Brazil. However, the number of occurrences is increasing, mainly in the South of Brazil (APOLO11, 2004) (STRAN; BOSELLI; ALENCAR, 2010), causing great destruction. In other regions of the planet, these phenomena leave behind a trail of destruction wherever they pass through (ARNOLD, 2005) (HUGHES, 2000) (THE TELEGRAPH, 2012).

However, the natural phenomenon that causes more destruction in Brazil is flood (STRAN; BOSELLI; ALENCAR, 2010). According to the Table I, from January to June, 2010, there were more than 789 occurrences of floods, of many types. In 2011, in Nova Friburgo, Rio de Janeiro, 429 people died, according to the civil defense, and 44 still missing, after a great flood made a hillside slid over many houses in Córrego Dantas, burying them. During this period it was possible to count more than 900 deaths in the highlands of Rio de Janeiro (R7 Notícias, 2011).

Even though that in many cases there is not a viable measure to avoid floods, there are some actions that can be taken in order to diminish the impact caused by it year after year. Among these actions are projects aiming to improve the runoff and draining of rain waters and the urban planning (JHA; BLOCH; LAMOND, 2012) (MIN, 2012) and alarm systems

(GLOBO.COM, 2011) (VEJA, 2011), which warn the population about an eminent flood in case of strong rains.

Table 1 – Map of natural accidents in Brazil (STRAN; BOSELLI; ALENCAR, 2010)

Type of events	Midwest	Northeast	North	Southeast	South	Total
Flood	16	199	74	164	336	789
Periods of drought	0	265	15	99	204	583
Hurricanes	0	0	0	0	4	4
Strong winds and hail	0	2	0	6	207	215
Mudslide	0	0	0	20	0	20
Marine erosion	0	3	0	2	0	5
Fluvial erosion	0	1	5	0	0	6
Uncontrolled migrations	1	0	0	0	0	1
Linear erosion	1	0	0	0	2	3
Dam disruption	0	1	0	0	0	1
Frost	0	0	0	5	0	5
Total	18	472	94	298	753	1.635

Source: Confederação Nacional de Municípios (2010)

This work aims to develop a simple and low cost solution that works as monitoring and alert system collecting data and feeding an Internet of Things (IoT) platform with it. This solution utilizes open source prototyping devices and the final version will be available in the web so people interested in environment monitoring of this kind can copy and implement it.

1.2- Objectives

This section will present the general objective of this work (1.2.1) as well the specific objectives (1.2.2).

1.2.1- General Objectives

This work general objective is to create a system composed by an electronic device that captures water levels readings and a system that stores this data in a NoSQL database, and then utilizes it in a local alert system. It also feeds the databases of the Cosm Internet of

Things (IoT) web service, being available for consultation in the web. It utilizes open source programming languages and prototyping hardware, aiming to reduce the implementation costs and also make it available to the general community.

1.2.2- Specific Objectives

The specific objectives are:

- Assemble a water level measuring device, with network transmission capability to transmit the collected data;
- Creation of a prototype that simulated a natural environment in order to test the electronic device that collects and transmits data;
- Modeling and creation of a database to store the collected data;
- Create a software to analyze the validity of the collected data, represent it graphically in a browser interface, emitting warning alerts in case the reading values come out of the acceptable range;
- Feed the Cosm IoT platform, which stores data collected from various types of devices and make it available for its users in real time;
- Create a Python component able to send data to the Cosm service and make it available to its community for utilization;

1.3- Methodology

A physical model that simulates a natural environment, like a river or specific part of a city, will be created and external variables will be applied to it making the water level vary. This model is necessary to observe the water level measurement prototype working and evaluate its efficiency in this kind of environment.

This prototype will be assembled utilizing an ultrasonic pulse sensor for distance measurement, a network board and the Arduino Mega project board (ARDUINO, 2012a), which utilizes a microcontroller that controls the data collecting device. The collected data will be transmitted by the network board through a CAT6 network cable, utilizing the HTTP protocol to exchange messages between the device and server (RICHARDSON; RUBY, 2007).

An application will collect this data and store it in a local database that will be modeled utilizing the CouchDB NoSQL platform (COUCHDB, 2012).

Among the other roles that this application will perform are the data consistency check, water level monitoring and data transmission to the Cosm IoT web service (COSM, 2012), which will make it available for utilization by the users of this service. It will also plot in a web browser application the graphical representation of the collected data, emitting an audible alert in case the risk of flood is detected.

The system performance in this simulated environment will be observed during fourteen days. From the data collected it will be possible to come to a conclusion regarding to the precision and stability, power source durability, and application viability on monitoring natural phenomena of this kind.

1.4- Work Structure

This work is divided in some fundamental parts, being this one its introduction, bibliographic revision (chapter 2) which brings the interpretation of important works published in areas related to this work, materials and methods (chapter 3) where the description of the methodology utilized in this work can be found. The discussion of the results (chapter 4) brings the results obtained and its interpretation and the final conclusions (chapter 5) closes the work, bringing what was observed and learnt with it.

2- Related works

With the increase in the number of natural disasters occurrences it is necessary to provide the communities with tools to prevent or at least diminish its effects, reducing the number of deaths caused by them. There are some solutions available, but most of them are quite expensive, being necessary a massive investment, making it not affordable for many cities.

This work aims to present a low cost alternative for flood disasters alert. To do this it will utilize open-source devices, programming languages and tools. This chapter will discuss about the nature of floods (2.1), Python as programming language (2.2), the utilization of the internet as a communication medium (2.3), data collecting hardware (2.4), and web applications (2.5).

2.1- Floods

Every year, hundreds of cases of flood are reported in Brazil (STRAN; BOSELLI; ALENCAR, 2010). Flood is a type of natural disaster that can cause great problems to entire communities and bring destruction of properties and in a higher level, lost of human lives (R7 NOTÍCIAS, 2012).

Figure 2.1 – Sewer drain obstructed by trash (TRIBUNA VARGINHENSE, 2012)



Source: Tribuna Varginhense (2012)

People living in urban zones directly influence some of the flood occurrences (Ashley; Ashley, 2007) (SZÖLLÖSI-NAGY; ZEVENBERGEN, 2005). The asphalt of the streets and the concrete around houses, sidewalks, and other types of construction makes it impossible for the soil to absorb the excess of water. Then the excess of water must run through storm sewers (Ashley; Ashley, 2007). However, the accumulation of trash (that was dispensed in an inappropriate way) in the drains (Figure 2.1) obstruct it blocking the water to run into the sewers (PACIONE, 2009).

There are situations when the rain volume is too high (SZÖLLÖSI-NAGY; ZEVENBERGEN, 2005). In these situations, even if the stormwater drainage and runoff structures are well dimensioned, the system overloads, exceeding its capacity, causing an overflow (JHA; BLOCH; LAMOND, 2012), and consequently, flood. In a different scenery, the riverbed flow capacity may not stand to the additional volume and then flood large regions.

Whether a flood has been caused by human influence or not, there are not much things to do when it happens. When a region is hit by flood it is necessary to evacuate the area as quickly as possible, in order to save lives. The existence of a flood alert system (IN360, 2012) (Veja, 2011) would allow people not only to save themselves, but also save some of their most important belongings.

The employment of electronic devices to provide data related to various types of readings by flood alert systems makes it possible to elaborate low cost collaborative solutions to monitor this kind of phenomenon (GERTZ; DI JUSTO, 2012). Moreover, it can provide data to researches concerning to climate modeling and other types of studies.

2.2- Python Programming Language

Python is an object oriented programming language, created by Guido van Rossum in 1991 and the name is a reference to the humor group Monty Python, which created the Monty Python Flying Circus.

It is a high level language, interpreted, and with an clean syntax. Its library collection covers a great range of implementation needs and its community on the internet is very active, creating modules, APIs and frameworks to talk to many applications (PYTHON, 2012). It was designed to be easy to read, utilizing code indentation instead blocks like Java or C or words like Fortran or Pascal.

Today the language covers from desktop applications to web and mobile applications. It is multi-platform, being interpreted in Mac, Microsoft Windows and Linux, among others, by its virtual machine. It is under the Python Software Foundation, an open source license, being free to utilized and distribute.

Two big projects that utilize Python in its implementation are YouTube (YOUTUBE, 2012) and the application server Zope (ZOPE, 2012). Great companies also utilize it in many of their components. It is being utilized as script language in 3D software Maya (MAYA, 2012), Blender (BLENDER, 2012) and others. Some Linux distributions added the language as an standard component, like Ubuntu (UBUNTU, 2012) and Red Hat (RED HAT, 2012).

Python was chosen for this work due its HTTP libraries that provide easy connectivity to web services and also because its libraries to work with strings. It was an important choice from the very beginning, in the testing stages, once great results were obtained in almost no time or research. It does not matter if it is a big or small project, Python is suitable for a great range of applications.

2.3- The world connected through the internet

The internet became over the years an important tool to spread information. It is both an utility and a medium (MIT TECHNOLOGY REVIEW, 2012) connecting people from all parts of the work. A new branch of application of the internet features is the Internet of Things (IoT), which consists in connecting devices that collect data and send it to be stored in online servers, being possible to make it available to other users. More information about IoT and Cosm, a web service that provides support to it, can be found in the next two sub-sections.

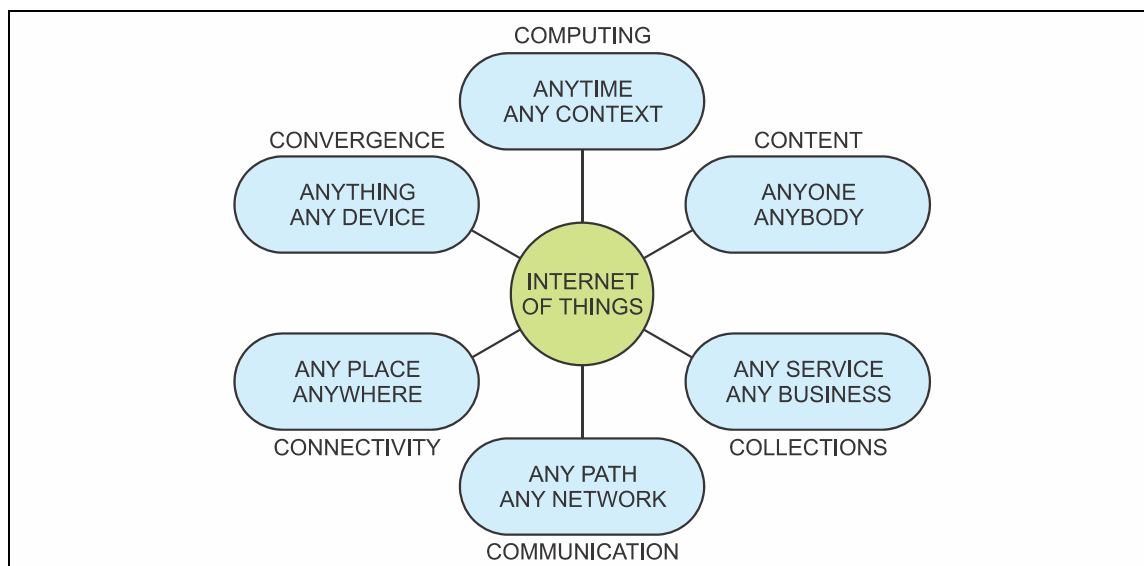
2.3.1- Internet of Things

There is today a great variety of devices that utilize technologies that allow its connection to information systems. A great variety of applications for this kind of devices can be found in the market. An example is a video camera in a pill shape, that when swallowed travels through all the digestive system taking and transmitting pictures that can be utilized for illnesses diagnostic. In a farm, automatic harvesters utilize GPS (Global Positioning Systems) to trace routes. Many types of sensors feed databases with the data collected, that are processed in order to provide useful information about the environment where it is placed in and how it works (FOX; MATSUMOTO; LAU, 2001) (MARGOLIS, 2012) (VAN DEN BROEKE; REIJMER; VAN DE WAL, 2004).

In the beginning the internet allowed connection only of computers, that is, it was able to connect computers and servers to a global network. After sometime this network become available to mobile telephones and then to many other types of electronic devices. The next stage of this connectivity evolution is the IoT, being possible to connect almost everything to the internet, managing it through the worldwide computer network (RAUNIO, 2010).

Through the web, people and objects are connected at any moment, anywhere, utilizing, to anything and by anyone, virtually utilizing any path or network and any service (Figure 2.2). To make it possible, elements like convergence, computing, content, communication, connectivity and collections must be present (RAUNIO, 2010). It implicates on the intersection of digital and physical worlds (BAOYUN, 2009).

Figure 2.2 – Components that are part of the Internet of Things



Source: Adapted of Raunio (2010)

There are many tools available in the web able to process the collected data and apply it to the IoT. Among them are the Google Powermeter (GOOGLE POWERMETER, 2012), Sensorpedia (SENSORPEDIA, 2012), OpenSense (SEN.SE, 2012) and Cosm (COSM, 2012). All these services provide tools to process and publish data in a way that is possible to extract meaning from it.

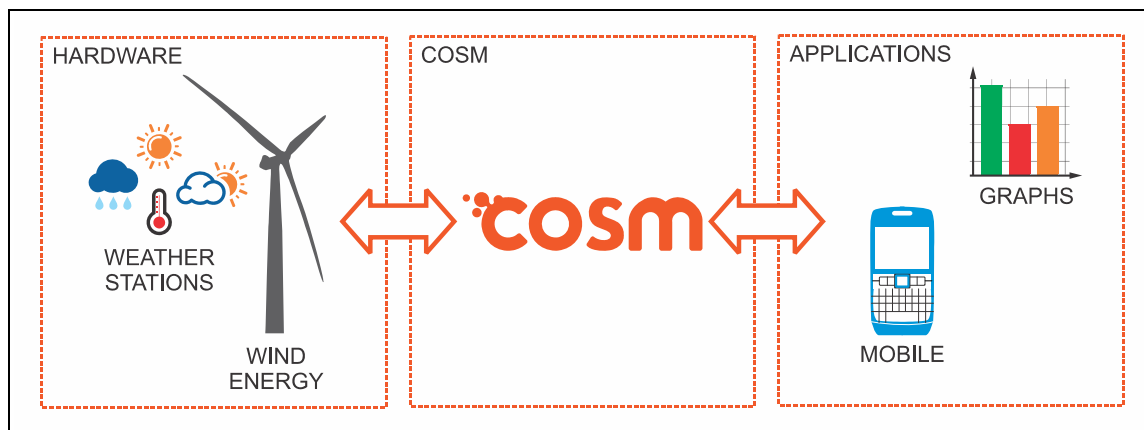
IoT is changing the way we interact with the objects around us. The number of devices that generate all sorts of data increases everyday and people are sharing it for free utilization (HERSENT; BOSWARTHICK.; ELLOUMI, 2011). The amount of data generated for these devices that is being stored in web servers grows continuously, and the tendency is that it grows even more (HUBER et al., 2000) (LYNCH, 2008). The creation and implementation of good tools for interpretation and visualization of this data is fundamental for efficient

employment of this information in environmental, social and industrial areas (RAUNIO, 2010).

2.3.2- Cosm.com

Cosm.com brings together people, devices, applications and the IoT (COSM, 2012). It is a free web service that provides storage, sharing and interpretation of data collected from various devices (Figure 2.3), through applications written utilizing its API (Application Programming Interface). It is a secure and scalable platform, that is, its capacity grows according to the demand (UCKELMANN; HARRISON; MICHAHELLES, 2011).

Figure 2.3 – Cosm and some of its interactions



Source: Adapted of Cosm (2012)

The creator of this service was Usman Haque, when he needed to manipulate data generated by sensors in interactive environments. It has been projected in 2008, receiving the name Pachube, and being accessible to the public in 2010. The name of the web service changed in 2012 and now it is called Cosm. Since it has been available to the general public, lots of people adopted it in order to connect, collect, interpret, and share data and applications that process it (COSM, 2012).

It is a collaborative tool, it means that anyone is able to connect a device and contribute with many projects already in progress or create a new one. What makes it possible is the fact it accepts many-to-many type of connections. The data from a determined user is available to others that can use it freely (UCKELMANN; HARRISON; MICHAHELLES, 2011).

Cosm API is very simple to use and its main function is to retrieve data generated by sensors and also data stored in web servers. It also provides methods for upload and download of information. The operation mode is based in the RESTful web service model (FIEDLING, 2000) (FAIRWEATHER; BRUMFIELD, 2011) (RICHARDSON; RUBY, 2007). The

complexity of a determined application is kept out of the system, retrieving and uploading the necessary data to the Cosm servers and all data processing is performed in the client side (COSM, 2012) (UCKELMANN; HARRISON; MICHAHELLES, 2011) (VASSEUR; DUNKELS, 2010).

Each application constructed corresponds, describing it in a simplified way, to a feed. These feeds are divided in data flows that can be generated by a particular sensor or device created to capture specific variations in the environment around it. The applications can be conceived to work with only one or many data flows at the same time and all communication, that is, request and response, is negotiated by HTTP protocol (GOURLEY et al., 2002) (VASSEUR; DUNKELS, 2010).

All these characteristics, combined to the easy utilization and access to the information makes Cosm a suitable platform for utilization in this work and at the moment it is being utilized in a great number of projects. It accepts a great variety of programming languages for application development (UCKELMANN; HARRISON; MICHAHELLES, 2011) and has a large and active community and the service aggregates lots of new projects everyday (VASSEUR; DUNKELS, 2010). Once the IoT field is growing a lot and in a fast way, it plays an important role in this medium numerous works have referenced it already (GERTZ; DI JUSTO, 2012) (OXER; BLEMININGS, 2009) (PAPARELLA; SIMKO, 2009) (PREMEAUX; EVANS, 2009).

2.4- Data collecting hardware

In order to obtain data about the environment in general and feed IoT platforms with it, it is necessary to utilize specific devices. The following sub-sections will describe some of them.

2.4.1- Microcontrollers

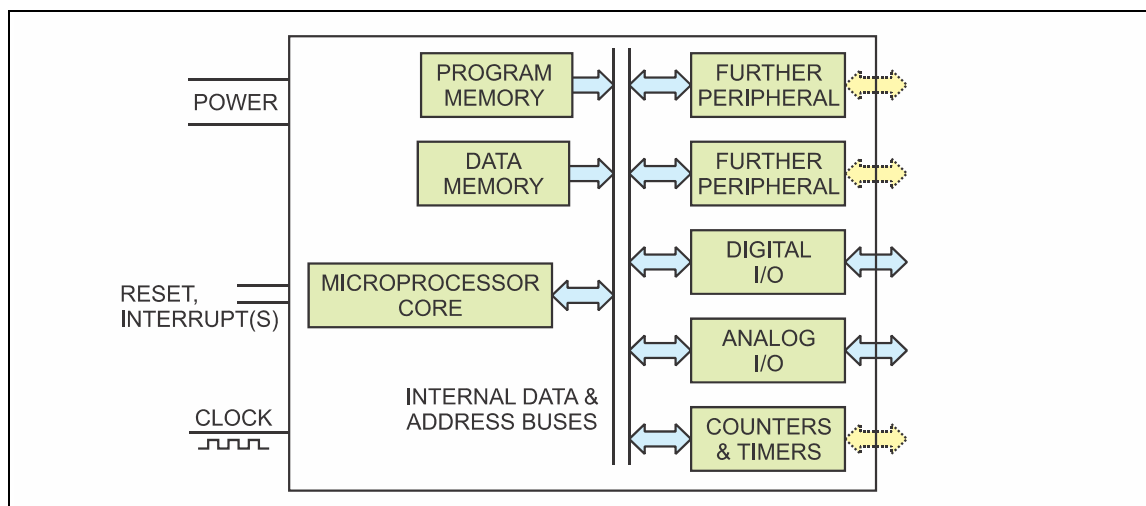
Since its first studies, the field of electronics had many significant changes. It is a result of studies promoted by many scientists and result of a very broad set of knowledge in related fields, like Physics, Chemistry and Mathematics (BRINDLEY, 2012).

Some discoveries were more significant once they represented a great advance in the technology. The thermionic valves, or electronic valves, employed at the beginning in telecommunications equipment, were replaced by transistors which reduced significantly the size of electronic devices (LEVINŠTEJN; SIMIN, 1998). Influenced by the need of miniaturization, the first integrated circuit (IC) has been conceived, invented by Jack Kilby in

June of 1958, an employee of Texas Instruments (TEXAS INSTRUMENTS, 2012). The IC allowed the combination of many transistors, creating components that could be employed in many applications. The IC invention allowed the creation of microcontrollers (IBRAHIM, 2003).

A microcontroller is a single chip microprocessor system. It has in and out parallel or serial ports, timers, internal and external interrupts and memory, all this integrated in a single component (IBRAHIM, 2003). They are programmed utilizing variations of C or Basic compilers, which translate the code to the Assembler language injecting this code into the targeted microcontroller (WILMSHURST, 2007). The Figure 2.4 presents a simplified schema of the interior of a microcontroller and its functions.

Figure 2.4 – Simplified scheme of a microcontroller (WILMSHURST, 2007)



Source: Adapted of wilmshurst (2007)

The main application for microcontrollers is for tasks automation. They are present in almost all electronic devices. From a remote control that change channels and the volume on a TV to the navigation instruments of a military aircraft (MACKENZIE, 1995). Its application is recommended also for hostile environments, like high temperature, radiation, bomb disarming, or any other adverse circumstance, where human presence or application of other types of devices won't be effective (IBRAHIM, 2003).

The development of the microelectronics revolutionized the computers (BRINDLEY, 2012). The integrated circuits, that eventually resulted in the creation of microprocessors and microcontrollers, had a main role on the development of uncountable number of electronic devices. However, write programs to run even the simplest task, like blink a led (light emitting diode) was something possible just to engineers of areas related to the electronics field or people with a great knowledge about this (OXER; BLEMININGS, 2009). The platform

Arduino (ARDUINO, 2012a) has been conceived in order to help breaking this barrier and allow people with almost no knowledge in electronics to step into this very restricted field. It is the theme of our next topic.

2.4.2 – Arduino

Arduino is a prototyping platform, which consists in a board composed by a microcontroller, an oscillator (a circuit that emits electronic signals repeatedly) and a 5 volts linear regulator, and it has been conceived by Massimo Banzi (Figure 2.5). Banzi always had interest in electronic experimentation and he was asked to create a prototyping platform that would be easy to learn and use, even for people with little or no knowledge of electronics. This work had as coauthor David Cuartielles. The idea had as base the Master studies of Hernando Barragan at Interaction Design Institute Ivrea (IDII), in Italy (ARDUINO, 2012a) (BANZI, 2009).

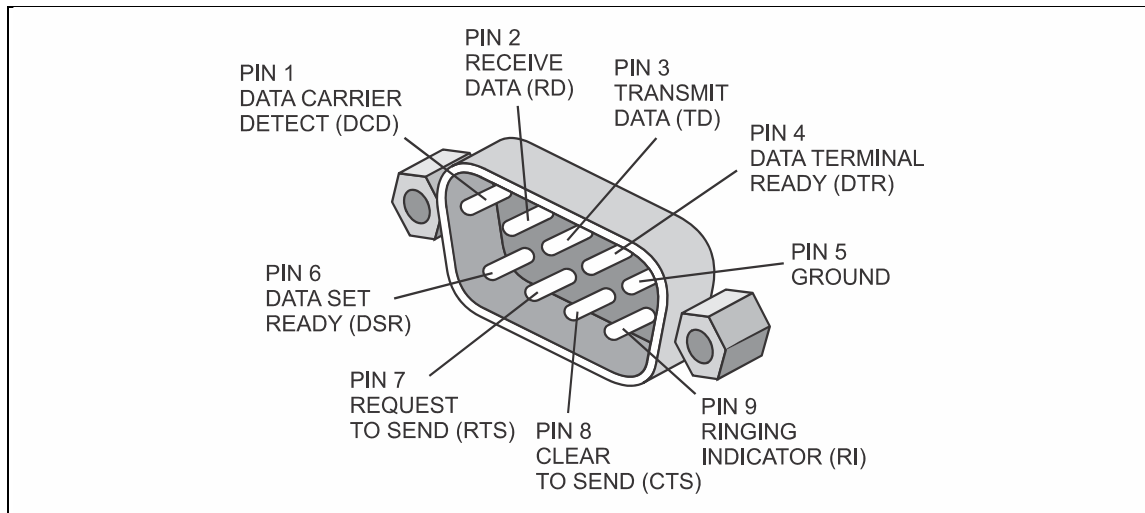
Figure 2.5 – Arduino UNO, the latest version of Arduino USB and first version distributed



Source: Adapted of Arduino (2012c)

The name “Arduino” refers to a historical character of the city of Ivrea. This project has been conceived to be multiplatform. It means that it can be utilized with various operating systems, like Microsoft Windows, Linux (various distributions) and Macintosh. It is programmable via USB port, which is a useful feature once the old serial ports (Figure 2.6) are not being found in some new motherboards manufactured today, and it is expected it to disappear. Moreover, it is based in open source software and hardware, being possible to be assembled by anyone, at no authorial rights costs (BANZI, 2009). Also it has a big community of users, which share experiences in many blogs and forums.

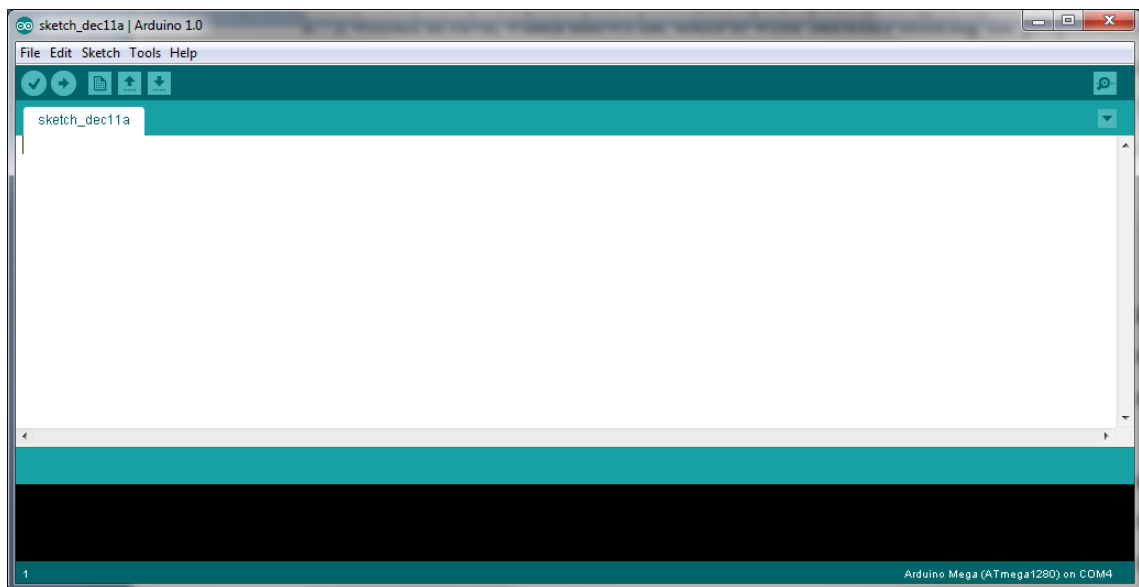
Figure 2.6 – Serial Port Scheme



Source: Adapted of Banzi (2009)

Another feature of Arduino is its IDE – Integrated Development Environment – (Figure 2.7), written in Java, which allows the users to write sketches utilizing the programming language named Processing (PROCESSING, 2012). Sketches are instructions that will be translated to the programming language C and delivered to the AVR-GCC compiler. It will perform the final translation to a language that the microcontroller can interpret (ARDUINO, 2012b).

Figure 2.7 – IDE for programming the Arduino’s microcontroller



Source: IDE screenshot (2012)

Due its ease of use, it is being applied to the most varied types of projects around the world. It is a low cost platform, and many electronic enthusiastic users are adopting this platform in their projects. One of the most notable examples of utilization of Arduino in environmental monitoring happened in Japan.

After the tsunami hit the Fukushima region, in March of 2011, many Japanese citizens, unsatisfied about the information disclosed by their government, decided to assemble Geiger counters coupled to Arduino boards and transmit the collected data to the internet. They have utilized the Cosm IoT web service, named Pachube at that time (COSM, 2012). This allowed people to quickly create a radiation map in Japan, providing information independent from the government.

With this and other initiatives this platform is proving its utility not only as a prototyping and learning tool, but also as an applicable solution for many problems in the world. From weather stations to electromagnetic interference (HUBER et al., 2000), Arduino is allowing many people to have access to a field before dominated only by engineers and highly skilled professionals.

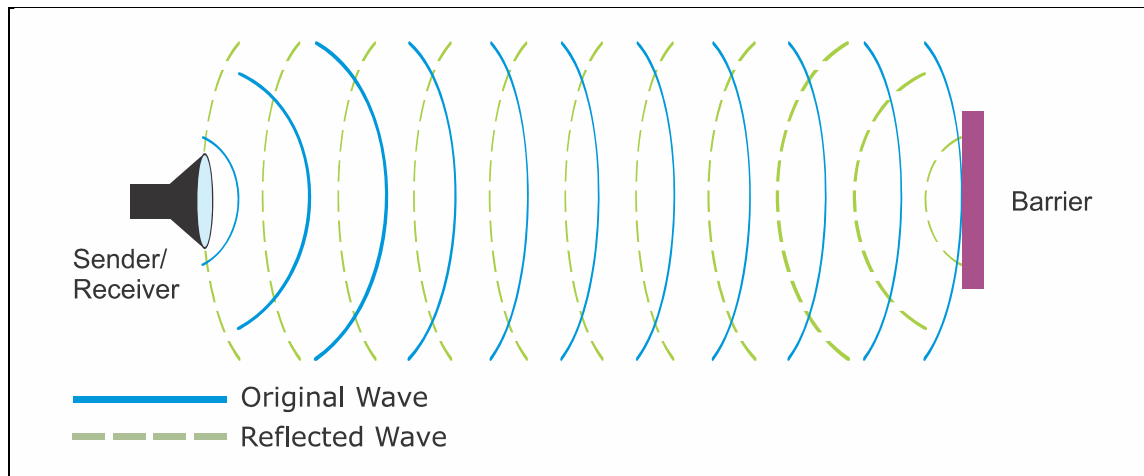
2.4.3- Sensors

A sensor is a device which converts physical stimulus to electric signals, to be read and interpreted. Due this characteristic, it can be considered as being an interface between the physical world and many electronic devices that depend on the interpretation of environmental phenomena (BKREV023). Some of these can save lives, like, for example, a device able to detect toxic gases presence in the air. Other can detect and measure heat, activating fire alert systems.

They can be of varied types and their applications are numerous. Some sensors can measure acceleration (DE SOUZA et al., 2010) (LAINE et al., 2001), shock (USUI et al., 2006) or vibration (BERNSTEIN et al., 1999). There are biosensors (MALMQVIST, 1993), chemical proprieties detectors (KAMEOKA; CZAPLEWSKI; CRAIGHEAD, 2004), electromagnetism (HUBER et al., 2000), and others. As there are different types of sensors, which cover different types of phenomena, for each phenomenon there are different technologies that can be applied in order to detect and measure it (LEVINŠTEJN; SIMIN, 1998).

One sensor which will be particularly useful to this work is the ultrasonic distance measurement sensor. This measurement is taken by sending a ultrasonic pulse that finds a barrier and is reflected back to the emitter which captures it (Figure 2.8). The distance is calculated based on the time the pulse takes to go and come back (BAXTER, 1996). Once it works by sound wave reflection it can be introduced into many types of environment, including water level measurement, which is an advantage, once it avoids some problems like oxidization due component immersion into the water (LEVINŠTEJN; SIMIN, 1998).

Figure 2.8 – Ultrasonic distance measurement sensor



Source: Adapted of Levinštejn (1998)

The sensors are very necessary nowadays. They are employed in critical and/or extremely dangerous systems covering many purposes (e.g. nuclear plants) and without them, various technological advances won't be possible to be reached (LEVINŠTEJN; SIMIN, 1998). Sensor networks around the world helps on weather forecast, disaster prevention or radiation levels monitoring. Through them it is possible to feed a local or remote database system (COSM, 2012) (SEN.SE, 2012) and processing this data utilizing specific applications, being possible to reach a better understanding about many different types of phenomena.

2.5- Web Applications

Web applications are tools that can be accessed by the web. They are applied to many purposes and its main characteristic is to be available everywhere, being just necessary to have an internet connection, so the service can be accessed. In general they combine tools already implemented (like database systems) and programming languages. The next subsections will describe some of the tools utilized in this project.

2.5.1- NoSQL Database - Couch DB

NoSQL is exactly what can be read in the Word: 'no' and SQL. There are some specialists however that defines it as being 'not only SQL'. It is a database modeling practice which presents a different structure paradigm, abandoning the relational model. The term covers all database systems of this kind and there are already a variety of technologies that implement this paradigm (TIWARI, 2011).

The applications that utilize it are specially directed, balanced to cover different application areas. The utilization of this kind of tool has grown in the past few years and the tendency reveals that this growth will continue (INDEED, 2012). The job market is already offering positions to professionals of this area (Figure 2.9) once there are many companies developing with NoSQL platforms, for example Cassandra NoSQL (CASSANDRA, 2012), MongoDB (MONGODB, 2012), Redis (REDIS, 2012), Riak (RIAK, 2012) LucidDB (LUCIDDB, 2012), CouchDB (COUCHDB, 2012), among others.

Figure 2.9 – Job offers for NoSQL platforms increased over 400% in the last two years (INDEED, 2012)



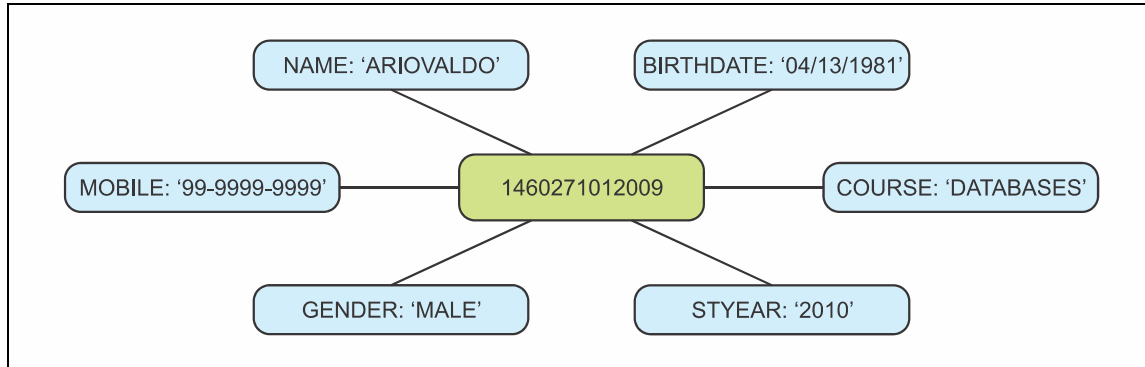
Source: Indeed (2012)

For this work it was chosen the CouchDB platform. It respects the NoSQL principles, not storing the data in tables and columns, but in JSON documents (Figure 2.10), which organizes it in pairs of attribute (key) and value, identified by a unique identifier. These documents have no size limit being possible to store even a whole webpage (TIWARI, 2011). It is adaptable to any application written utilizing a programming language that has libraries for HTTP and JSON. Moreover, it is an open source platform, being under the Apache license (COUCHDB, 2012). It has been developed as a project for creation of a new database platform, by Damien Katz, in 2005.

CouchDB is a RESTful (Representational State Transfer) (FIELDING, 2000) (COUCHDB, 2012) (GOURLEY et al., 2002) (RICHARDSON; RUBY, 2007) (SCHREIER, 2011) system, it means that it implements the HTTP protocol for GET, PUT, POST and DELETE transactions. It does not pack the data, it is utilized as is. Also, it utilizes JavaScript as querying language. It has been developed utilizing Erlang OTP, which is a programming language that offers concurrence features. Concurrence is useful when it is necessary to create

scalable databases without loss in availability and security (COUCHDB, 2012). Another characteristic that makes it promising is that it is faster than relational models, both for reading and writing transactions (TIWARI, 2011).

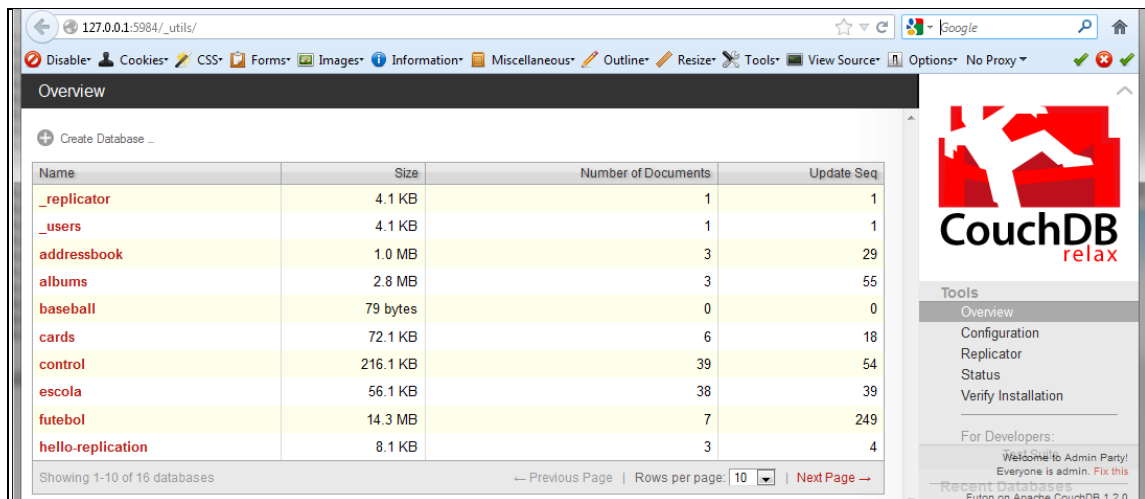
Figure 2.10 – Example of a JSON document



Source: Adapted of Tiwari (2011)

This platform is ideal for document oriented applications. It can be applied to email systems, Wiki websites (collaborative content), cataloging, project management, among others (TIWARI, 2011). It has a very intuitive management tool, Futon (Figure 2.11), and it allows programmers with very little knowledge about database to develop applications without much struggle (COUCHDB, 2012).

Figure 2.11 – The CouchDB management tool Futon



Source: CouchDB Futon screenshot (2012)

Despite this is a relatively new tool, it is being utilized in some projects (ATREV016) (ATREV017) (ATREV018) already. Many improvements are being implemented and this platform, as well other NoSQL models, is being adopted by companies (WPREV029) as a viable and worth alternative to relational databases.

2.5.2- Programming languages for the web

It is necessary to utilize specific programming languages inside of the web context. Some of them started as desktop languages and then incorporated new features for the web. An example of this is Java. Other were born inside of the web. The next sub-sections will bring some of the concepts of HTML5, which incorporates raw HTML with new tags, JavaScript and CSS3, and its specification still under development.

2.5.2.1- HTML5

Hypertext Markup Language (HTML) is a markup language utilized for content presentation in the web (W3C, 2012c). Has been conceived in 1990 by Tim Berners-Lee, at this time working at CERN, Switzerland, and become popular with the Mosaic browser, developed by NSCA (Figure 2.12). Today it is standardized by W3C (World Wide Web Consortium), where member organizations, full time teams, and the general public work together developing standards for the web (W3C, 2012a). The last full release of this language, version 4.01, has been released in December of 1999 (W3C, 2012c).

Figure 2.12 – Web browser Mosaic



Source: Favbrowser (2012)

In 1998 the W3C decided that they would close the version 4.01 of HTML and it would stop being developed (W3C, 2012a). It was believed that the future of web relied in XML (eXtensible Markup Language). The development works of a new specification, the

XHTML, a version of HTML that utilizes XML, started and became a W3C recommendation in May of 2001.

However, a group from a company named Opera didn't agree about adopting XML as being the future of the web. These individuals began to develop a conceptual specification that extended the HTML forms functionalities, and did it without breaking retroactive compatibility. This specification has been named Web Forms 2.0, being included in the HTML5 specifications later.

Workers from Mozilla foundation joined the Opera group and lead by Hixie Hickson, dedicated more time on the development of this specification, and named the new group WHATWG (Web Hypertext Application Technology Working Group) (WHATWG, 2012a). Hickson left Opera later to work for Google, where he dedicated full time on the development of HTML5, which at this time still named Web applications 1.0 (WHATWG, 2012a).

In 2006, Tim Berners-Lee recognized in a paper published in the MIT's digital magazine that the W3C maybe had been too optimistic about the XML functionalities: *"It is necessary to evolve HTML incrementally. The attempt to get the world to switch to XML, including quotes around attribute values and slashes in empty tags and namespaces all at once didn't work."* (DIG, 2012). Works for a new HTML specification started, and it was decided that the the previous works of WHATWG would constitute the basis of the new HTML version.

The W3C and the WHATWG group work in parallel on the development of HTML5. Due this fact there are two concurrent specifications. One is the official W3C version (W3C, 2012g), the other is the WHATWG group version (WHATWG, 2012b). Good ideas were included and the bad ones rejected independently of the source. The WHATWG group has created a website containing a more detailed description of the differences between the two versions (WHATWG, 2012c).

Thus, what HTML5 is? There is some confusion when it comes to define it exactly. Many technologies were included erroneously in its definition, being these just parallel works or independent specifications. An example of this is the Scalable Vector Graphics (SVG), which is an open standard for working with vector images. For some time it has been considered as a HTML5 component, but it is a graphic specification independent of W3C.

HTML5 utilizes CSS and JavaScript to format and deliver content, beyond the standard HTML that gained new tags for special behavior. The programming language today aims to target three main objectives: interoperability between browsers, error treatment

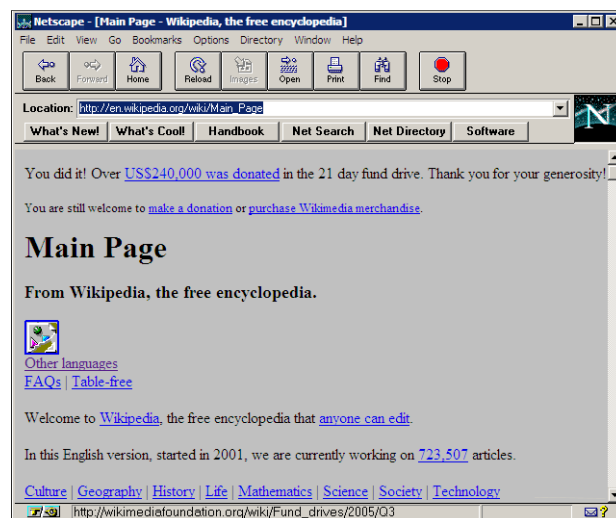
(characteristic seen for the first time in HTML) and the evolution of the language in a way that it becomes easier to develop web applications (BKREV028). This new version has the potential to replace technologies such as Adobe Flash (Adobe, 2012) to deliver videos and animations (FORBES, 2011), it can manage different medias, organize better the the page contents and other functionalities that didn't exist in the previous versions of HTML.

With so many new functions added, this new HTML standard became a very rich tool and perfectly employable in a great variety of applications for various platforms. The characteristic stands out however is the independence for proprietary platforms to deliver determined functions (ATREV014). The only one tool necessary then would be a browser that offers support to the new functionalities. It means that the need for various plug-ins disappear and it enhances the browser's usability (DAVID, 2010).

2.5.2.2- Java Script

JavaScript is a programming language for web browsers (CROCKFORD, 2009). Few web technologies had a so significant growth as it did. Many new adepts adhered to the language as soon it has been released in the middle 90's (CROCKFORD, 2009) (DANESH, 2007) (GOODMAN; MORRISON, 2007), being a part of the Netscape 2 web browser (Figure 2.13). It was developed by Ecma International, and the original name was EcmaScript. Today it has many of its API definitions specified by W3C (W3C, 2012b).

Figure 2.13 – Web browser Netscape 2 which was the first to offer support to JavaScript



Source: Wikipedia (2012)

Despite its name, this it has no real connection to the Java programming language (HEILMANN, 2006). JavaScript, as the second part of its name suggests, is a scripting language, utilized to add interactivity to the web pages from the client side. It updates the

contents dynamically through the DOM (Document Object Model) interface of the browsers (W3C, 2012b).

The language has been developed aiming to replace the Java applets that failed due being too heavy (CROCKFORD, 2009). Even not being extensively tested, it has been accepted really fast and applied to numerous websites, adding interactivity to them, ranging from a simply “good morning” message adjusted by the client machine’s clock to games and complex functions (DANESH, 2007).

As it was stated previously, JavaScript, being incorporated by the HTML5 specification, proposes to solve some problems such incompatibility between proprietary platforms and also different operating systems, mainly mobile platforms (KEITH, 2005). What was a platform developed in parallel and had to compete against proprietary scripting languages, now has been adopted as a standard aiming to promote higher compatibility. It ensures that end-users reach a better usability in general (W3C, 2012b).

JavaScript meets the market needs and is stable enough to be employed in complex applications. Its inclusion into the HTML5 specification will make it much more compatible to many mobile devices and browsers, and has potential to replace completely many proprietary solutions (DIG, 2012) (HEILMANN, 2006) (LAWSON; SHARP, 2011).

2.5.2.3- CSS3

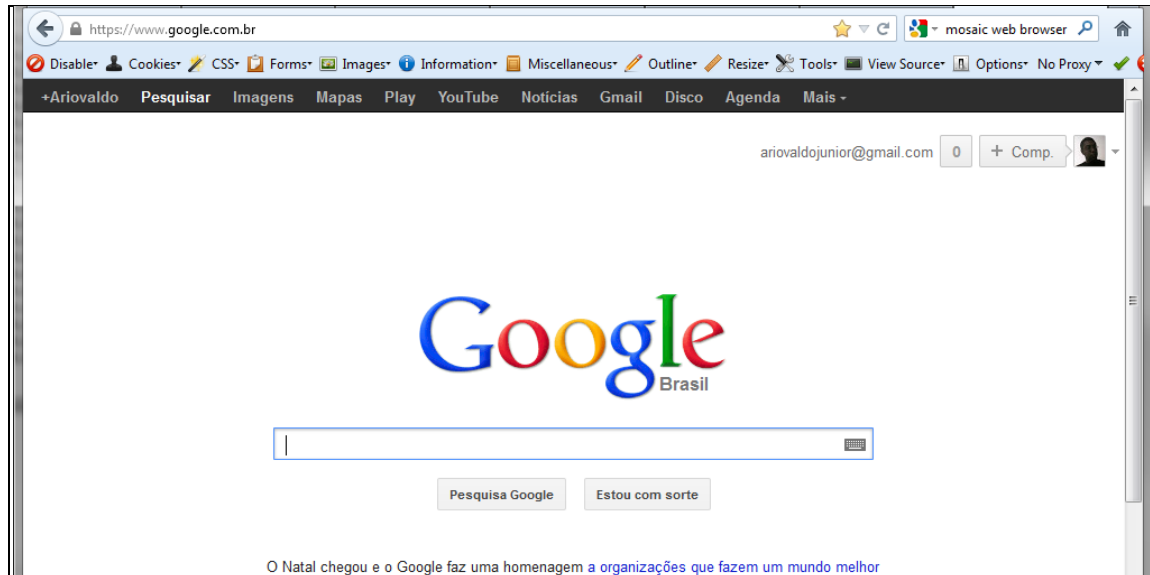
CSS (Cascading Style Sheets) is a style sheet language utilized to determine how the web pages will be presented in the browsers (Figures 2.14, 2.15) (GASSTON, 2011) (W3C, 2012c). This language has been incorporated to the HTML5 specifications, being the newest version named after CSS3, which still under development (GOLDSTEIN; WEYL; LAZARIS, 2011).

Its first version (CSS1) became an official W3C recommendation in December of 1996 (receiving a revision in June of 2008) and the second version has been released in June 2011 (W3C, 2012d). The version that is being named CSS3 is not finished yet and it is believed that more resources still being added, like shadows, round corners, transparency, and others, aiming to fulfill specific needs that its previous version didn’t cover, and do it without needing any other programming language (GOLDSTEIN; WEYL; LAZARIS, 2011).

Before the CSS development, when a design item was needed in a webpage (like round corners, gradient, etc), it was necessary to include images and other techniques, in order to produce the desired effect. Any kind of page layout formatting demanded a great amount of time and very often the final result was not what it was expected to be. Applying the style

sheets to the HTML document, in special the ones that follow the standards of CSS3, such problems were solved, allowing the development of lighter pages, with better code maintenance and lower number of images (GOLDSTEIN; WEYL; LAZARIS, 2011) (HOGAN, 2011).

Figure 2.14 – Google web page applying CSS



Source: Browser screenshot (2012)

Figure 2.15– Google web page without CSS



Source: Browser screenshot (2012)

The CSS3 role became more important (HOGAN, 2011) with the development of the HTML5 specifications. These two specifications are being worked in parallel and a constant review is necessary to know what is being formalized and what is being left out (GOLDSTEIN; WEYL; LAZARIS, 2011). It is expected that with the advances of these

specifications, along with JavaScript, a faster development of multiplatform dynamic applications for the web. It is also hoped that it will eliminate the need of proprietary plug-ins, which in many cases is an accessibility obstacle for the end user.

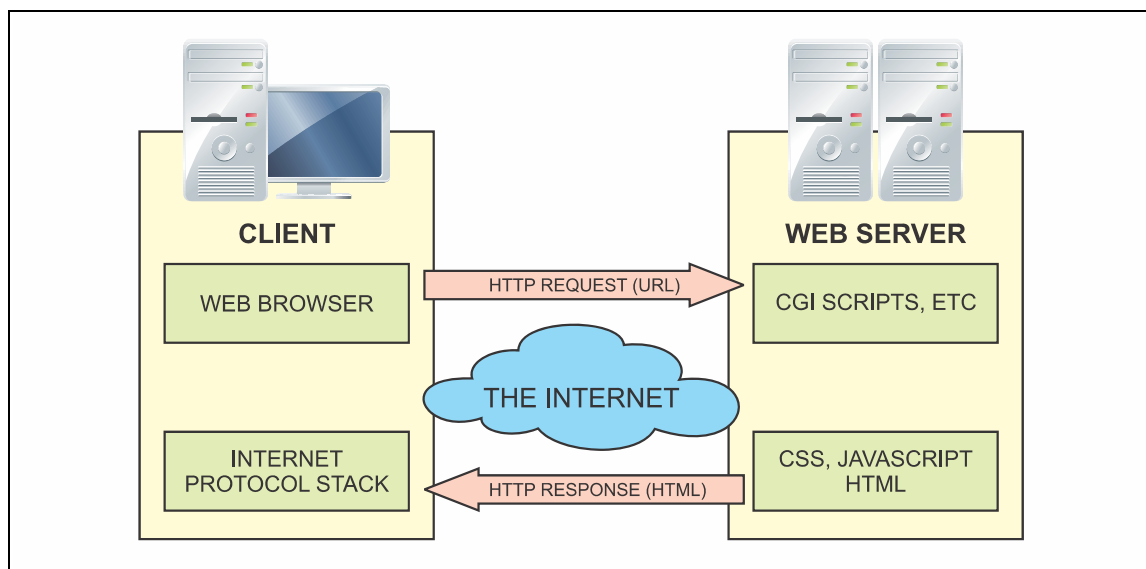
2.5.3- Communication and data retrieving

The Hyper Text Transfer Protocol (HTTP) is a protocol utilized to send and retrieve data. It is through it that the browser (or another kind of client-side application) and the web server communicate. It is the most utilized communication standard. Its specifications are kept and released by IETF (Internet Engineering Task Force) and the last stable version is the HTTP 1.1, being RFC 2061 its official document (IETF, 1997) (W3C, 1999).

The Server, as stated before, distributes services to the web. It stores static websites or even complete applications of a company or e-commerce. The client is the application utilized by the user which sends, retrieves and interprets the results of various requests, and for this work, its definition is restricted to software. The most common client application is the web browser. The processing can take place whether in the client or server sides, or even in both.

The communication by this protocol occurs through request and response. The figure 2.16 illustrates the HTTP protocol communication between client and server.

Figure 2.16 – HTTP protocol communication



Source: Adapted of Gourley (2002)

The request is basically composed by four parts (GOURLEY et al., 2002):

- The method or action to be performed;
- The URI (Uniform Resource Identifier), which is the information about the resource that has been requested by the client;

- Information about the HTTP protocol version;
- Additional information that can be complementary to the previous information or not.

The method is applied to the object that is defined by the URI. It can be of many types, being the most common POST and GET. The POST method sends information in the body of the requisition to the web server. It is widely utilized to send forms that can enclose a great amount of data, being more indicated to data injection in the server. The GET method is the most utilized, and simply requests to the server to send back a resource and is the standard method, that is, in case no method is specified, it is the one that will be utilized (BKREV039). It sends the requested data via URI, being these visible in the address bar of the browser (GOURLEY et al., 2002) (IETF, 2012).

The resources allocated in a server have a name, from which the clients can identify and request it. The name of a determined resource is its URI. These are unique addresses that can contain information about the location of an image in a particular website (GOURLEY et al., 2002) (RICHARDSON; RUBY, 2007).

The information about the HTTP protocol are sent with the requests and are useful to identify in the Server which is the version supported by the browser. The additional information sent may contain data about the document types that can be accepted, page encoding supported, and other (GOURLEY et al., 2002).

The response of this protocol occurs after process the request, determining how the request will be replied. However, sometimes a request cannot be delivered by the server, for many reasons, like lack of access privileges, authentication issues, error in the server or resource not found. In this case the server can return an error code. These error messages can be customized (GOURLEY et al., 2002) (RICHARDSON; RUBY, 2007).

One of the most important components in an HTTP response is the information block that defines which kind of document is being sent to the browser, making it possible for it to present the contents in the correct way. It is coded by MIME (Multi Purpose Mail Extensions), a table of types that identifies the correct code to be utilized for each existing extension (IETF, 2012). For example, in case of files with html/htm extensions, the MIME type returned is text/html. Some very common MIME types are text/plain (ASCII files), application/zip (compressed files), image/gif (GIF format image) and image/jpeg (JPEG format image), among others (IETF, 2012) (Kozierok, 2005) (WILDE; PAUTASSO, 2011).

Around the world, servers, browsers, and applications connected to the web communicated between each other utilizing the HTTP protocol. Another protocol, SOAP,

which utilizes XML, is more utilized by companies and has its specification released by W3C. However it is not utilized as much as HTTP is (CHAPPELL, 2004) (MUELLER, 2002).

3- System Development

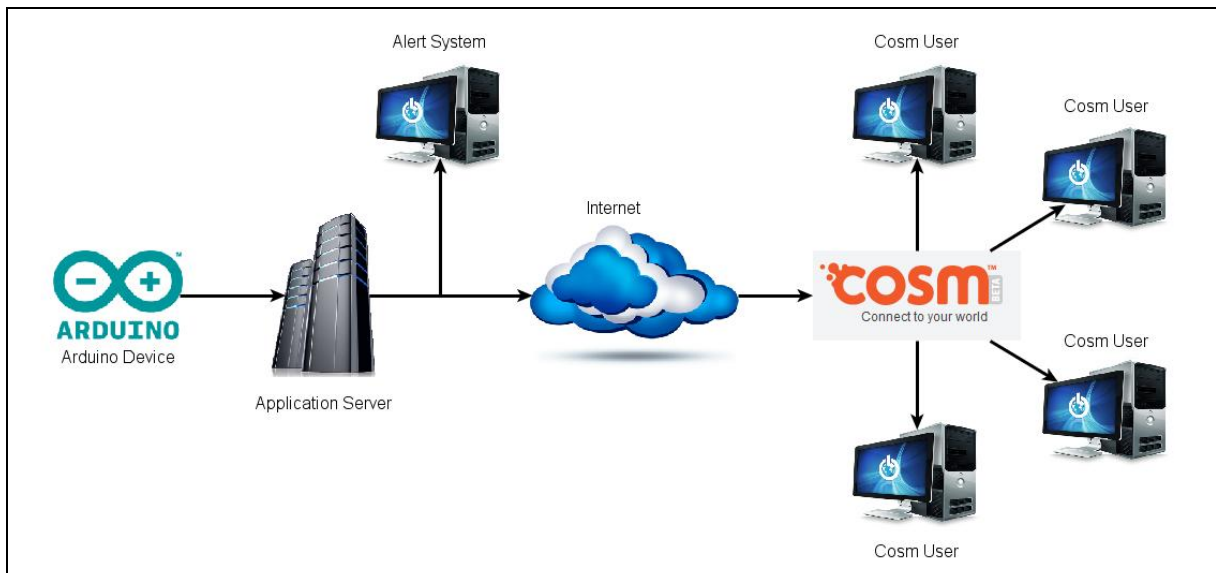
The objective of this chapter is to present the development of the system of this project. By system we can understand that it is “*a purposeful collection of interrelated components that work together to achieve some objective*” (SOMMERVILLE, 2007). All the stages conceived in order to reach out the results presented in chapter 4 are described here. It has been divided in two parts: hardware and software.

This section is divided in eight parts. The section 3.1 brings an overview of the Software Processes and Architecture. The section 3.2 describes the Socio-technical Systems and its Requirements. The System Design and Software Development are detailed in section 3.3 and The Environment Simulation Module in the section 3.4. The Arduino Device, the Main System, and the Flood Alert component are described in the sections 3.5, 3.6, and 3.7 respectively. Finally, the section 3.8 is about the Cosm component.

3.1- Software Processes and Architecture

The system’s main function consists in retrieve and store the data collected by an Arduino device and utilize it to feed an alert sub-system and also feed the Cosm IoT web service servers, being available to other people (Figure 3.1).

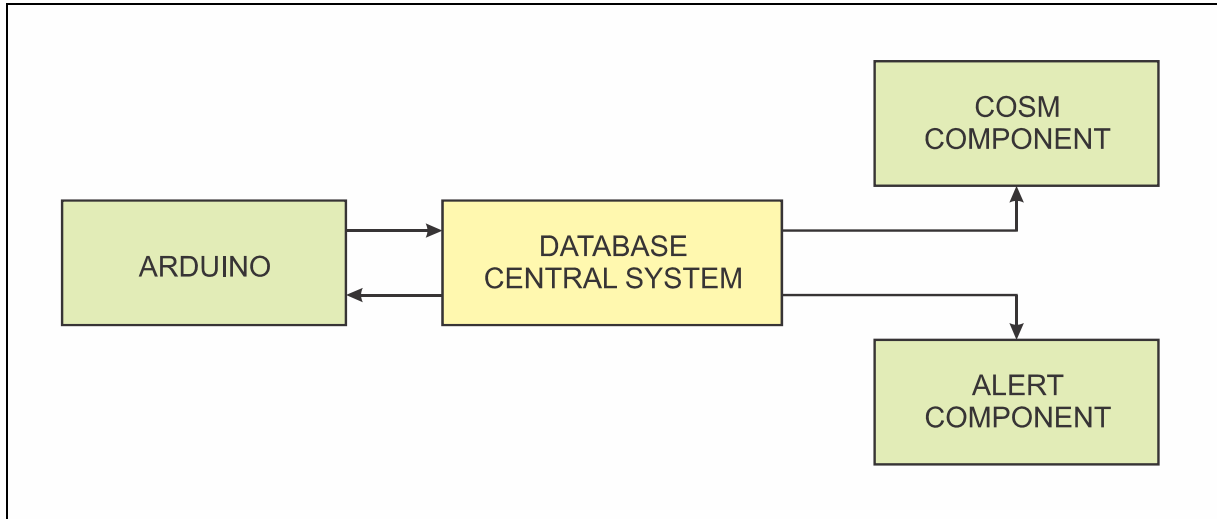
Figure 3.1 – Overview of the system



Each component of the system depends on the other to work properly. The system follows the repository model (SOMMERVILLE, 2007). This model is indicated in cases where the same data is utilized by several different components of the system, avoiding data redundancy. All data collected is evaluated for consistency and then stored in a local database.

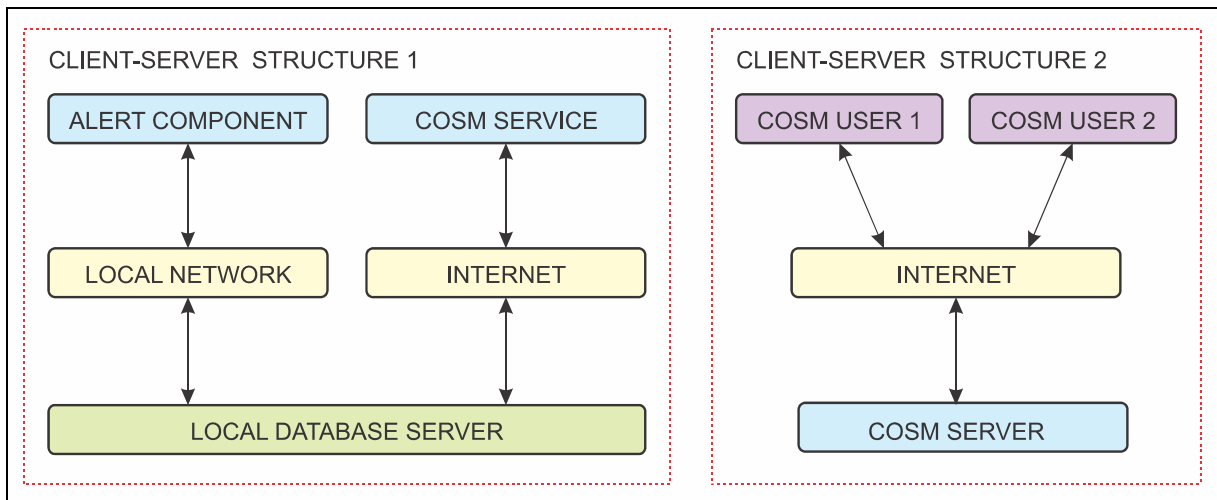
The information is retrieved from this database by the alert sub-system and also replicated to a database of the Cosm IoT web service when internet connection is available. Figure 3.2 illustrates this feature within the system.

Figure 3.2 - The data is stored in a database and utilized by other sub-systems



It also follows the specifications of a client-server model (ABOUT, 2012). Basically there are two client-server structures present in the system. The main system database server and the Cosm IoT web service server. These structures are illustrated in the Figure 3.3.

Figure 3.3 – The client-serve structures present in the system



The first structure shows the connection between the main server to both Cosm and alert sub-systems, that is, both sub-systems depend on the data stored in the main system database to work properly.

The second structure shows the connections between the Cosm IoT web service and eventual users. The utilization of the data stored in the service is an example of client-server model, where possible researchers would develop an interface to couple to the service, retrieve the data, and use it in their own projects.

The system has a central control. A component controls the data flow from the Arduino device to the database server and its utilization by the sub-systems. This central component monitors the system status assuring that the sub-systems only can work if it signalizes that it is running fine. It also provides the access to the sub-systems controls and tests through its operating interface.

As a whole it is a data processing system (BOURQUE; CLARK, 1992). The system behavior changes according to the variance in the input, or the failure on it, triggering other actions. Also the data stored changes after being processed by the Cosm component during the process of upload to the service databases.

3.2- Socio-technical Systems and its Requirements

The system modeled for this work is a socio-technical system composed by technical sub-systems (SOMMERVILLE, 2007). It is necessary human intervention and knowledge about the system in order to get the correct output from it. Another characteristic that indicates that it is a socio-technical system is the fact that it is affected by external events, once its main role is to monitor specific conditions of the environment, like water levels in a river or wetlands.

As an abstract functional requirement, required by the system to work properly, it needs to check the connection to the Arduino device connected to the local network. The device can be connected directly to the network adapter in the server or be accessed through a router or switch, using the IP 192.168.1.45. This is the IP that has been set on the microcontroller. After getting the response saying that the communication is working, the next step is to check the database availability, so the data can be stored in it. These two conditions must be met so the other sub-systems can perform their functions well.

During the system availability check data discrepancy is also assessed. Once the range of measurement is set according to the environment where it is installed, it is necessary to verify if the system is presenting invalid data. This phase also can uncover possible general failure in the system that can be generated by component malfunction or even complete destruction of it due to external natural events.

After installed the ideal up time of the system should be 100%, so issues related to power source availability and internet connection must be mitigated. Detailed information regarding system availability can be found in the modules descriptions in later sections of this chapter. The system should not present downtime unless it is necessary to interrupt it on

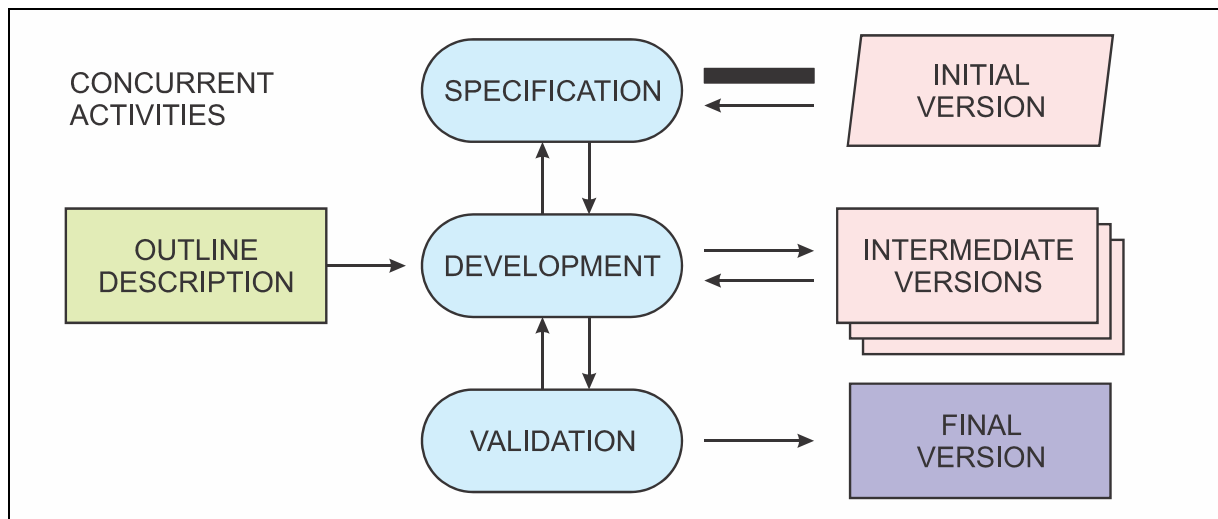
purpose for server maintenance, code update or hardware parts replacement. Any case other than these listed before must be investigated.

An internet connection is required for the Cosm component to work. If the option of synchronizing the data of the local server database with the Cosm IoT web service database is enabled in the main system panel and the internet connection fails, the sub-system will be interrupted automatically in order to prevent data modification and discrepancy. If it passes in the connection test, the next step is to check if the Cosm IOT web service is working properly. In case the service is not working properly the sub-system will be interrupted too.

3.3- System Design and Software Development

The system can be divided in two main parts: hardware and software. The software portion of it depends on the data collected by the hardware, so its sub-systems can work properly. It has functional and non-functional emergent properties. It is necessary to have the hardware and a local web server working to achieve any result, and it is the functional property of the system. As non-functional property it has a module which the main role is to send data to Cosm IoT web service servers, were it will be available for external use. A description of each part can be found in the Table 3.1.

Figure 3.4 – The Evolutionary Development Model (SOMMERVILLE, 2007)



Source: Adapted of Sommerville (2007)

The hardware has been conceived in order to create the necessary input for the system, based in the Arduino open-source prototyping platform. It means that it can be both a commercially available off-the-shelf solution or you can assembly your own hardware modules, once its schematic design is available for download in the project's webpage (ARDUINO, 2012c). It is a broadly utilized solution for carrying out experiments of multivariate purpose and nature (ARDUINO, 2012a).

The software developed followed an evolutionary approach (Figure 3.4) which consists in retrieving the data generated by the Arduino device and interpret it using its sub-systems. It has been planned and developed having in mind three major stages.

In the initial phase of the development the objective was to connect hardware and software and ensure that the input provided by the Arduino device was valid for utilization by the system. The first version of the main system was very simple, without graphical interface, and its function was to get and assess the input data and insert it in a non-relational database.

Table 3.1 – Description of the system modules

System Division by Modules		
Nature	Component	Description
Hardware	Environment Simulator	Composed by three tanks, a water pump and plumbing. It has been constructed to simulate the conditions of a natural environment where an Arduino device would be installed.
	Arduino Mega	This module has the microcontroller which is loaded with the basic instructions to translate the events that will become the system input. It is powered by a 9 volts battery.
	Ethernet Shield	It is coupled to the Arduino Mega module to give it the ability to communicate over a network connection.
	Ultrasonic Sensor	This sensor measures the distance from the device to the water surface emitting an ultrasonic pulse and calculating how much time does it take to hit an obstacle and travel back.
Software	Main System	It controls the data flow between the hardware and the two sub-systems, storing it in a database.
	Flood Alert Component	The flood alert component which monitors the input stored in the database triggering specific events when necessary.
	Cosm Service Component	This component sends the data stored in the local server database to an external database located in the Cosm IoT web service, where it will be available for other users.

From the beginning it was kept in mind that the main system would have to suffer several changes in order to accommodate the new features. They were added in order to supply the functionality needed by the side-components. To test the connection to the database and retrieve the stored data a small web application was built utilizing JavaScript and simple HTML.

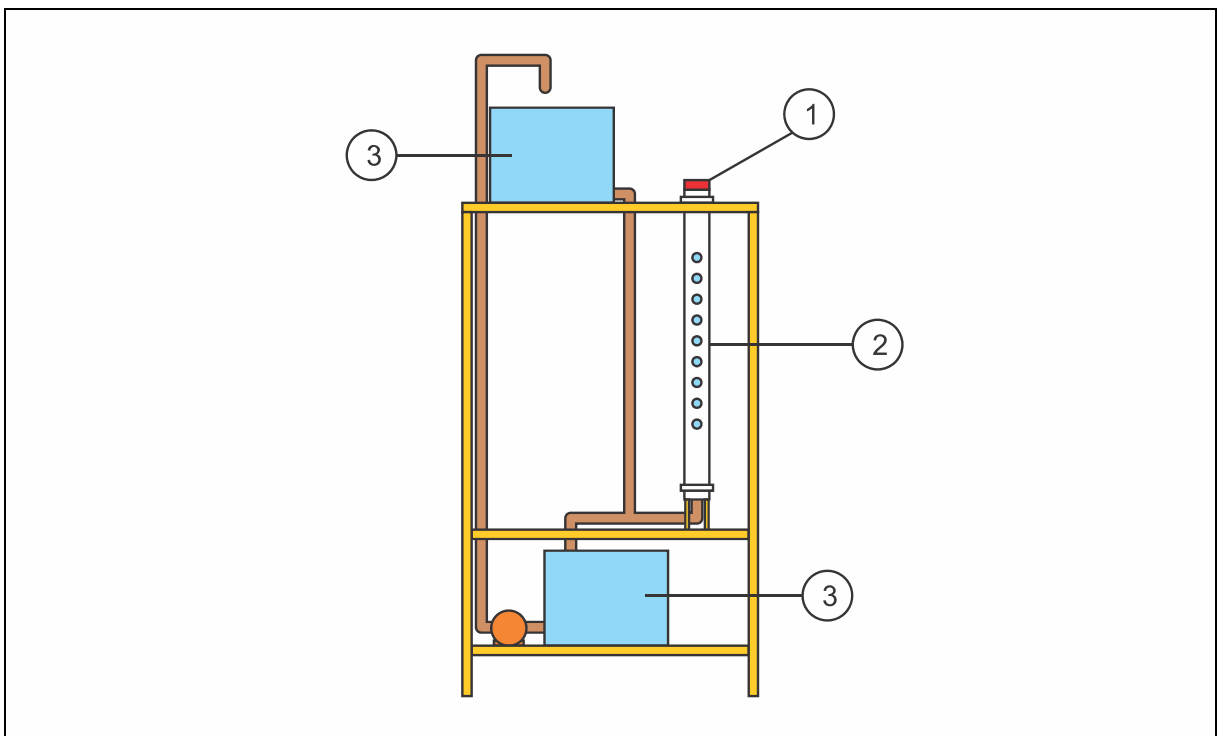
The second phase had as objective to create the alert sub-system and its interface. It was developed utilizing HTML5 elements like raw HTML for basic elements, JavaScript to add dynamic behavior and Cascade Style Sheets (CSS) to format the contents in order to deliver the information to the user in a more pleasant way. The main system also had to evolve to accommodate new features necessary to this sub-system, like controls for checking the elements utilized by it.

In the third phase it was developed the Cosm component that has as function to send data to the Cosm IoT web service databases utilizing its API. It is coupled to the main system, being accessible through its panel, causing it to change once again.

3.4- The Environment Simulation Module

In order to test the Arduino device that collects data for the system it was necessary to construct a physical model able to simulate the changes in the water level. It represents what could occur in a real environment where the device would be installed, like a river, a lake or wetlands. The model is represented in the Figure 3.5.

Figure 3.5 – The physical model for experiments



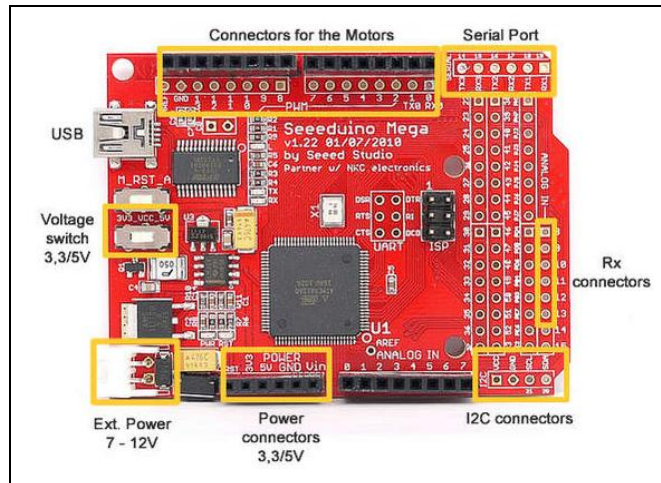
This model consists of three parts as illustrated. The device that measures the water levels is inside of a container (1) installed at the top of a PVC tube of 100 mm (2). The water level inside of this tube will change responding to the changes of water flow. Three lamps were placed behind the tube, making it possible to visualize the water level variation.

Attached to it are two tanks (3), one full of water which, when liberated by a shutoff valve, raises the water level inside of the container in a controlled way. Another tank, which is empty at the beginning of the tests, collects the water when necessary being controlled by another shutoff valve, lowering its level inside the system. Both tanks are connected by a water pump that send the water collected in the lower tank to the higher one so the system can be reset to perform tests continuously. The variation in the levels simulates the changes in a natural environment.

3.5- The Arduino device

In order to take the measures of water levels inside of the physical model an electronic device has been assembled from individual manufactured modules commercially available ready for use. It utilizes an Arduino Mega like board, an Ethernet module and an Ultrasonic Distance Sensor module.

Figure 3.6 – Board following the Arduino Mega specifications, assembled by Seeedstudio



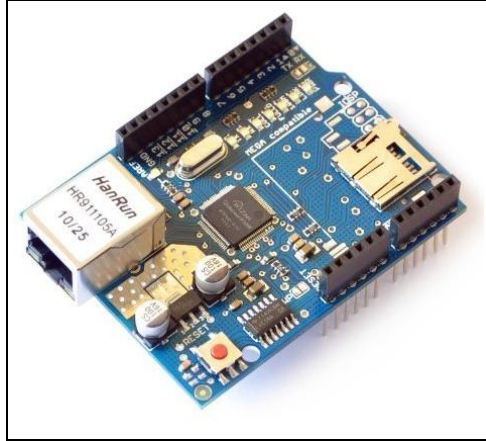
Source: Seeed (2012)

The the board assembled following the Arduino Mega (ARDUINO, 2012a) specifications (Figure 3.6) was manufactured by Seeedstudio (SEEED, 2012). The board has the following specifications: microcontroller ATmega 2560 of 16MHz, selectable 5V/3.3V operation, 70 Digital IO ports, 16 Analog inputs, 14 PWM (power with modulation) outputs, 4 Hardware serial ports. This board can be powered by an USB port, 9 volts battery or AC to DC adaptor.

This module is the device's brain. The instructions that control its behavior were transfered to the ATmega 2560 microcontroller utilizing the Arduino IDE version 1.0. These instructions consist in a code able to bring up a web server that loops continuously, requesting the information collected by the ultrasonic range measurement module and informing the

distances read by it. This data is transmitted by HTTP protocol over the network when requested by an application installed in the server that will collect it, check if it is valid, and then store it into a local database for later utilization.

Figure 3.7 – Ethernet Shield W5100



Source: Arduino (2012b)

The Ethernet Shield W5100 Wiznet for Arduino (Figure 3.7) was assembled according to the specifications found in the Arduino website (ARDUINO, 2012b). It has an operating voltage of 5VDC (supplied by the Arduino Board), Ethernet controller W5100 with 16K internal buffer, 10/100Mb connection speed. It is connected to the Arduino board through a SPI (Serial Peripheral Interface) port. It utilizes a RJ45 standard connector and CAT5/6 cable to connect to a computer or router. For this project a CAT6 cable was utilized.

It connects the Arduino Mega module to the server. The IP number set for it was 192.168.1.45 and it uses the HTTP port (80) for communication. Once it fits perfectly on the Arduino board it leaves the connectors free for utilization by other devices, like the ultrasonic sensor utilized in this project. Another feature it has is the Micro-SD module integrated on it, that can be utilized for storage or even accommodate a heavier web server in it.

The Ultrasonic Distance Sensor module (Figure 3.8), assembled by Seeedstudio, has a measurement range from 3 centimeters to 4 meters. It requires a 5VDC power supply (in this case it is supplied by the Arduino board) and has a dual transducer.

Figure 3.8 – Ultrasonic range measurement module assembled by Seeedstudio



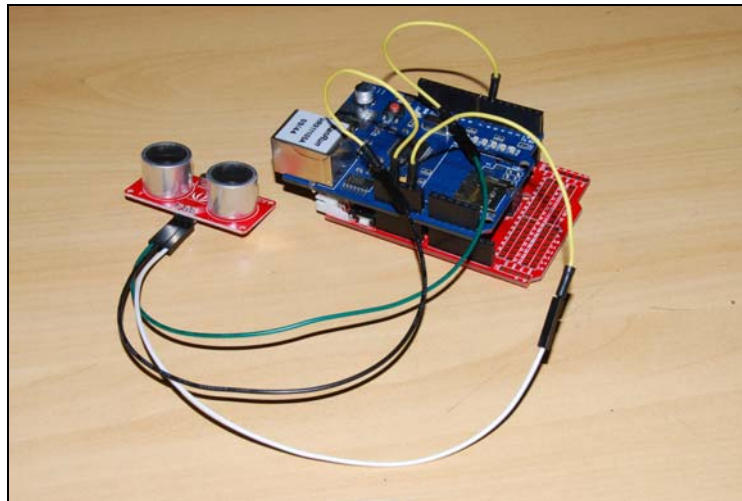
Source: Seeed (2012)

It works sending pulses that hit an obstacle and come back to the device and the distance is given by the time it takes to travel all the way to go to and come back from it. It has two pulse emitters/receivers, which helps to deal with the environment interferences. For example, if a fly land in front of one of them, blocking the signal, the other can continue giving the correct readings (test it).

All the three electronic devices were assembled in order to constitute the water level detector as illustrated in the Figure 3.9. The Ethernet shield has been coupled to the Arduino Mega board and the ultrasonic range measurement module was connected to the ground, VCC and pin 6 ports of the Ethernet module. Then it has been connected to the server by a CAT 6 cable to provide the inputs utilized by the system.

To avoid data loss in case of system downtime due to power source failure, a backup cell must be provided, so the main source can be replaced without stopping the device.

Figure 3.9 – The three devices assembled in order to compose the water level detector



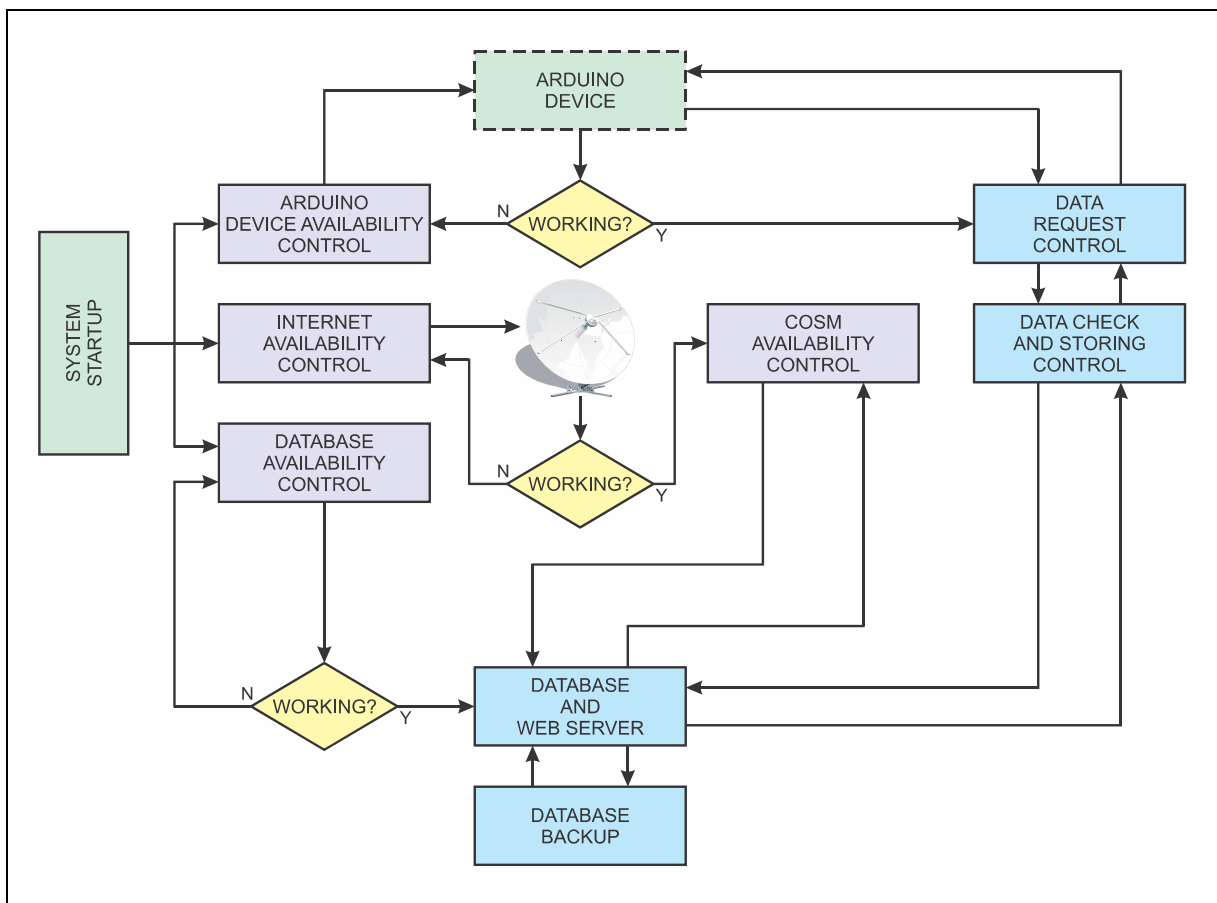
Due to its modular characteristics it is relatively easy to increase the functionalities of the Arduino device. Other modules could be coupled to the system and provide new types of data, like temperature or humidity. On the server side this additional data could be added with no much effort to the non-relational database. A simple change in the model of the JSON document processed by the main application would be enough to add this extra data.

Another great advantage of using pre-assembled modules is that the hard job on projecting, testing and approving to production for these parts is already done. In some cases a small adjustment is necessary, but in most of the cases it is not necessary too much knowledge in electronics in order to assemble different modules and get results from it.

3.6- The Main System

The main system is the core of the whole system. It retrieves the data from the Arduino device and stores it into the database, being available for the two sub-systems. It evolved with the development of the other components of the system incorporating new features when it was necessary. The final version of it is composed by the items described in the Table 3.2 and the relationship between these components is demonstrated in the Figure 3.10. All the programming languages, ready-to-use solutions, and other technologies utilized in this project were applied to it keeping in mind two factors: low cost and maintainability.

Figure 3.10 – Relationship between the components in the main system



It has four variables to store the communication status. The variables are `arduino_status`, `couchdb_status`, `internet_status` and `cosm_status`, all pre-set to the Boolean value “False”. The connectivity to the internet and Cosm IoT web service are not mandatory, once the user can choose to send the data later. After all the tests conclude it will be asked if the user wants to schedule a re-check in 5 minutes.

The Arduino Device Availability Control is a component that checks if the device is available. Once the user starts the tests it will be the first one to run. It will send a request to the Arduino Ethernet board and if it returns the code 200, the status icon will be set to green

and the variable `arduino_status` will be set to 'True'. Also the button that activates the Arduino Capture (Data Check and Storing Control) and the Flood Alarm sub-system buttons.

In case it returns a different code, it will set the status icon to red, the variable `arduino_status` remains 'False', and it will not be activate the Arduino Capture and Flood Alarm buttons. This control is also utilized every time that the system sends a data request to the device and a timeout occurs. It will freeze the system to avoid data loss due its corruption. When it is operational again it triggers another system startup routine to check the availability of the other components and update the availability variables according to their status.

The Data Request Control sends the data request to the Arduino device delivering it to the the Data Check and Recording Control. If it cannot establish connection with the device it sets the availability status to 'False' and triggers the Arduino Device Availability Control to monitor the system until it becomes operational again.

Table 3.2 – Main system components in the final version

Main System Components	
Arduino Device Availability Control	Checks if there is communication with the Arduino device component and trigger an specific alert in case it becomes unavailable for whatever reason.
Data request control	Connects to the Arduino device sending a data request
Data Check and Storing Control	Checks if the data received is ok and then stores it.
Database Availability Control	This component checks the database availability when the software is being loaded.
Database Backup Control	Performs a security copy of all data stored in the local database.
Database and Web Server	Implemented utilizing the non-relational database system CouchDB. Store all the data gathered by the Arduino device and complementary information in the format of JSON documents.
Internet Availability Control	Integrated with the Cosm component, checks the internet connection. The result returned determines if the sub-system will be available for use or not.
Cosm Availability Control	Checks whether the Cosm Iot web service is available or not.

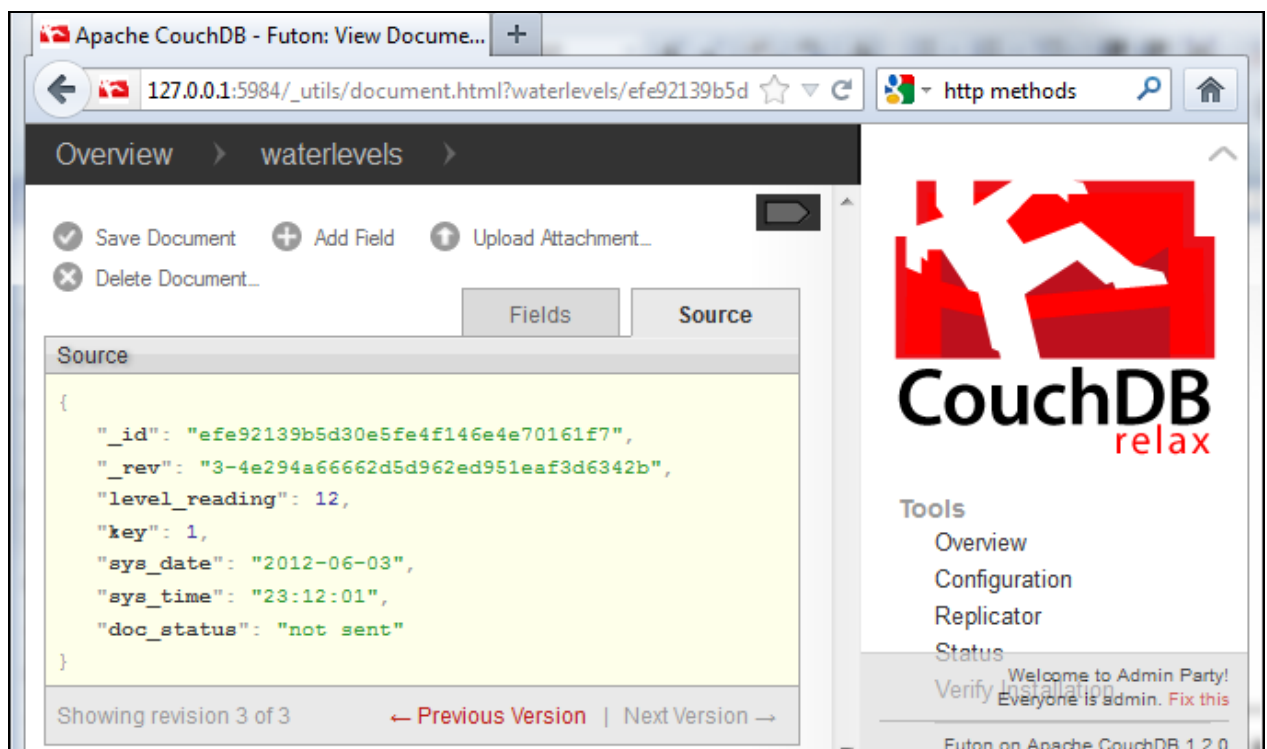
The Data Check and Storing Control has as main function to check if the input value from the Arduino device is valid (within an acceptable range) so it can be packed with the other information to compose a JSON document. This document is inserted into the database where it receives an ID and revision code. The DBMS then sends a confirmation saying that

the data has been added successfully. The data then is ready for use by the sub-systems. If the system cannot insert it into the database, it will trigger the Database Availability Control to run, freezing the system until it is available again.

In case the input has an invalid value, like a too high peak in the signal, it will be stored in a buffer. If the next five readings are within the same range, the documents for these five values are created and the alert sub-system handles it. If less than five readings return with the discrepant value this data is erased and an average value based on the last 10 readings before the freeze takes its place being stored into the database.

The Database and Web Server are controlled by the CouchDB NoSQL DBMS. CouchDB is a non-relational database system that stores the data utilizing documents instead of tables. Each JSON document stores the data requested from the Arduino device plus system date and time, status of the document (if it has been sent to Cosm or not), an auto-increasing key value that work as index, the document id and revision that are generated by the CouchDB DBMS. The index value is also stored in a separated document and updated every time a new JSON document is created, making it easy to set a new index on the startup. An example of JSON document generated by the system and stored into the 'waterlevels' database inside CouchDB is showed in the Figure 3.11.

Figure 3.11 – JSON document stored in the database.



Source: Screenshot

In order to communicate with the database it has been utilized a CouchDB library designed for Python 2.7. This library provides all the HTTP 1.1 methods and other functions necessary to access, retrieve, and recording data into the database. This library is easy to understand and apply. First it is necessary to paste its contents into the lib folder where the Python installation files are located in and add an import line in the code to make its functions available.

The Database Availability Control checks the connection to the database in order to prevent data loss. In case the connection to the database fails the `couchdb_status` variable is set to 'False' until it is available again, triggering a complete checkup after the connection with the database is established. During this freezing time the readings from the Arduino device are stored into a buffer variable waiting for the connection to the database to be available again. As soon it is working again, the data is stored and the buffer emptied.

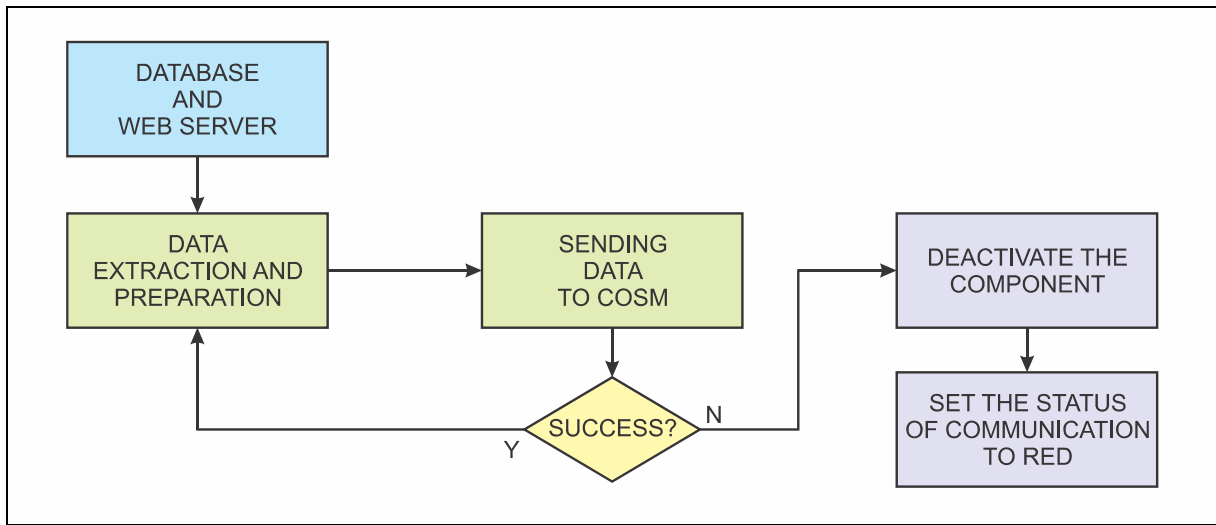
The Database Backup Control generates a backup copy of all data stored in the database. It is a functionality projected to liberate space in the server disk or simply have a security copy of the data collected during the system utilization.

The Internet Availability Control checks if there is an internet connection available. Once the system was designed having in mind that it could be utilized in a place with no internet connection the alert sub-system would run without any problem. If it succeeds the Cosm Availability Control will be available to run, otherwise the `cosm_availability` will be automatically set to 'False', setting the both 'Internet' and 'Cosm' status icon to red and keeping the option to upload data to the Cosm IoT web service unavailable.

If the internet connection is available, the Cosm Availability Control tests the connection to the Cosm IoT web service. The `cosm_availability` status variable will be set to 'True' in case the test succeeds and the status icon will become green. The Cosm component button that activates the broadcasting will remain unavailable until the connection to the system becomes available.

3.7- The Cosm component

The Cosm component is the component that prepares the data stored in the database to be sent over the internet to the Cosm service databases. This control extracts the necessary information from the JSON document stored in the CouchDB and sends it to the Cosm IoT web service. Once it receives the confirmation that the data had been transmitted successfully it changes the status of the document's key 'doc_status' 'not sent' to 'sent' informing the system that it is not necessary send this document anymore.

Figure 3.12 – The Cosm component scheme

In case it fails to send the data due a connection timeout error, it will deactivate the sub-system and disable its activation button, triggering the Internet and Cosm Availability Controls from the main system to check the connection status until it is available again. If the problem is not in the connection the Cosm Availability Control will perform the necessary tests to check if the problem is in the Cosm service, liberating the sub-system to for data broadcasting after the connection is restored. This sub-system scheme is illustrated in the Figure 3.12.

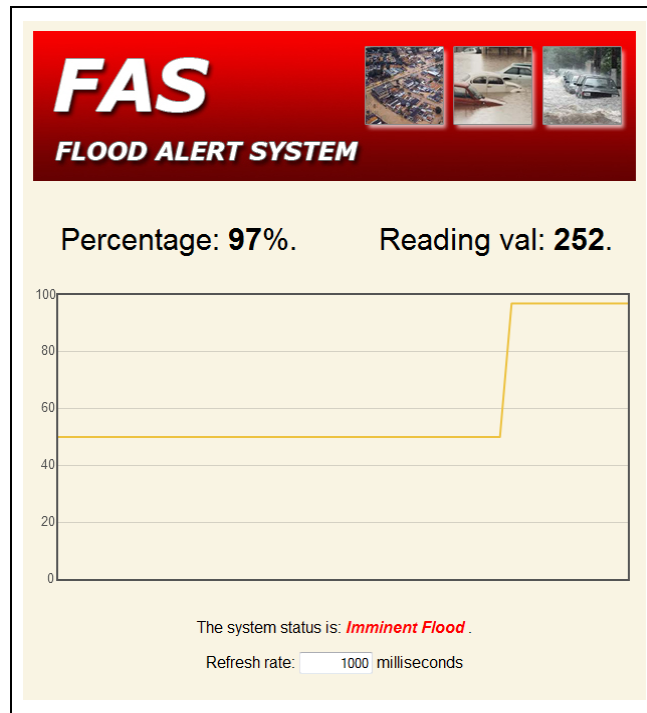
3.8 – The Flood Alert component

This sub-system is a browser-based application that operates utilizing the inputs produced by the Arduino device stored in the database. The application is accessible by the main system interface that enables it for utilization if the Arduino Device and the Database Availability Controls confirm that these components are working. It was implemented in HTML5, that is, the interface utilized raw HTML in its structure, JavaScript to add dynamic behavior and CSS3 for content formatting.

The JavaScript libraries jQuery and Flot were utilized to plot the dynamic graphic, which will be updated following the rate specified by the user directly in the interface. The maximum reading value limit is set by the user in the main system interface. The minimum reading value limit is not set because it is not important to this system in particular.

The maximum and minimum reading registered will be displayed in the interface, informing the environment situation during the system utilization. When the water level reaches a critical value (which is set in the main system and displayed in this interface) an alarm sound will be played, warning about it.

Figure 3.13 – The Flood Alert component scheme



Source: Screenshot

The water level status can be found over the graphic plot. The status is set to ok if the reading range is from 0 to 65% of the critical value set previously. From 65% to 95% it is set as “Danger”. Above that it is the status is set to “Imminent flood”.

The other elements in the interface are only for design purposes, with no dynamic behavior whatsoever. The scheme for the Flood Alert component can be visualized in the Figure 3.13.

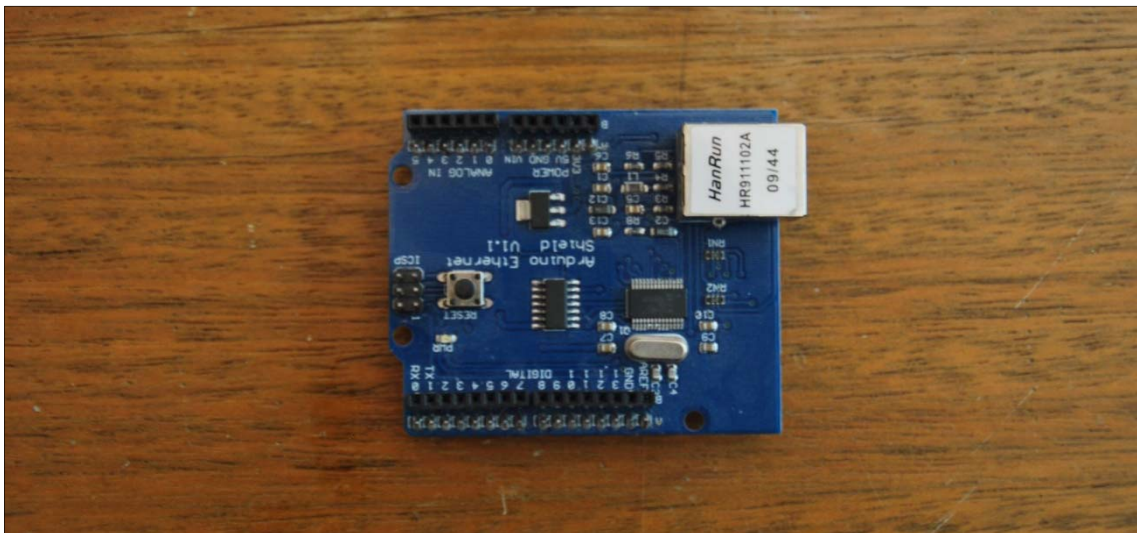
4- Results and discussion

From the scope definition for this work until its final presentation it was necessary to go through several steps. Each one revealed a particular set of results. In this section the results obtained during the development of this work are presented and discussed. Section 4.1 is about the Arduino Water Level Measuring device and the 4.2 is about the Environment Simulation module to which it has been coupled. The section 4.3 discuss about the Main System and its components, as well about the Flood Alert and Cosm components.

4.1- Arduino Water Level Measuring device

There were two versions of the Arduino Water Level Measuring device. The first version presented a problem in the network module. The network module ENC2860 (Figure 4.1) chosen in the beginning due its lower cost. However, its project is not supported by the Arduino project, it means that this module must have its own libraries to work properly and at the time it was utilized in this project, the final version for this library was not concluded, presenting several compatibility issues and malfunction.

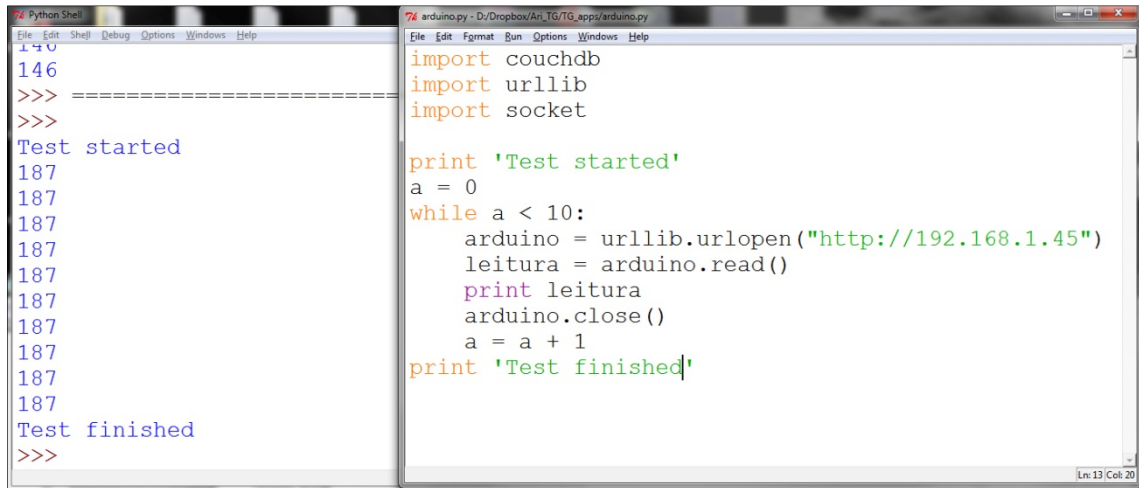
Figure 4.1 – The Ethernet module ENC28J60 that was discarded from this project due compatibility issues.



Once it was not possible to find a suitable code or modify the library aiming to make it work, it has been replaced by the network module W5100, which is fully supported by the Arduino project. It worked perfectly right in the first attempt.

A very simple application was utilized for testing purposes and it shows that the device is able of measuring distances. The data retrieved by this device showing it is working is shown in the Figure 4.2.

Figure 4.2 – Data retrieved from the Arduino WLM device



The figure consists of two side-by-side screenshots. The left screenshot shows a Python Shell window with the following text: '140', '146', '>>> =====', '>>>', 'Test started', '187', '187', '187', '187', '187', '187', '187', '187', '187', 'Test finished', '>>>'. The right screenshot shows an Arduino IDE window with the following code: 'import couchdb', 'import urllib', 'import socket', 'print \'Test started\'', 'a = 0', 'while a < 10:', ' arduino = urllib.urlopen(\'http://192.168.1.45\')', ' leitura = arduino.read()', ' print leitura', ' arduino.close()', ' a = a + 1', 'print \'Test finished\''. The status bar at the bottom right of the IDE shows 'Line 13 Col 20'.

After the initial tests, the device was left on during a period of three months, powered by a 9 volts alkaline battery, being connected to the computer every week to check if it still working properly. It could be observed that it still working just by checking the power led, however, it was also necessary to validate the inputs.

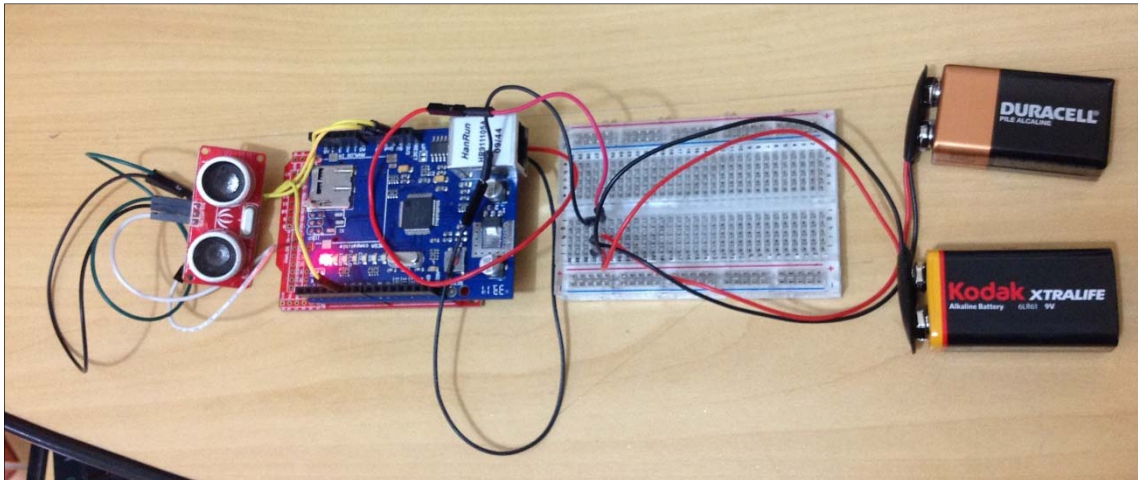
Figure 4.3 – Battery level after three months of use



This test showed that the device is stable, working continuously with low power consumption. The test was interrupted because it gave enough data to assume that it would continue working for more time, once the battery tester indicated that the energy level indicated it still green (Figure 4.3).

However, in order to avoid interruption in the data feed, it would be recommendable to change the batteries after this period. To do that without turning the device off it is necessary to have two battery connectors assembled in parallel. The new battery is connected to the free connector first, then the old one can be removed (Figure 4.4).

Figure 4.4 – Batteries assembled in parallel to prevent the device to turn off



The tests performed with the device returned good results, spending a low amount of energy to continue working and providing correct inputs even after a long period of time working continuously.

4.2- The environment simulation module

It was necessary to construct an Environment Simulation module, so the Arduino Water Level Measurement device could be tested. Once the shutoff valve was open, it filled the tube, raising the water level. It was half full, then the system was adjusted to this mark (set to zero), for testing purposes.

Every time more water was allowed to go into the testing tube, it was necessary to wait two minutes for the perturbation in the water surface to cease. In 10 tests performed, the measurement precision varied from -3 mm to +4 mm, for different variations in the water levels taken having the zero mark in the tube as reference point. The Table 4.1 shows the results obtained in the tests.

Table 4.1 – Water level measurement and variation precision

Test number	1	2	3	4	5	6	7	8	9	10
Mark in the tube (in mm)	0	50	100	150	200	250	300	350	400	450
Device reading (in mm)	0	51	99	14	202	251	304	351	398	447

The measurement difference detected during the tests represents an acceptable variation; in the real environment it won't represent a significant difference.

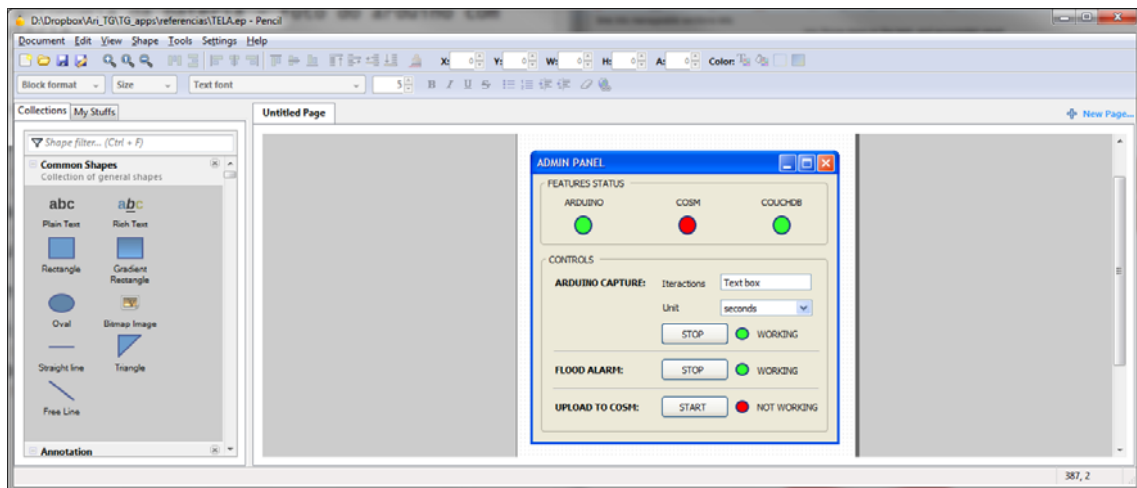
4.3- System implementation

After check that the device and the testing module worked properly providing the necessary data, the implementation began. The following sections will show the results obtained for each component of the system.

4.3.1- Main system

During the Main System panel implementation it was necessary to implement temporary false components and check if it performed well. The free software Pencil (reference here) was utilized to design the first version of the interface (Figure 4.5)

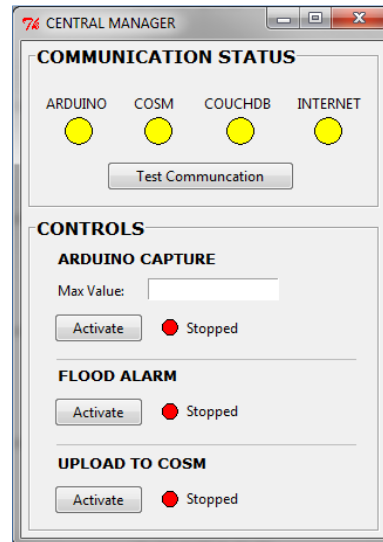
Figure 4.5 – First version of the interface designed with the free software Pencil



The final version needed one more status control to check the internet connection. Some components had their names changed to better represent it. Some layout adjustments were applied in order to distribute the components better and the final version of this interface can be seen in the Figure 4.6.

Despite it is represented only the conceptual design and the final version, the Main System had to go through several steps until its conclusion. Once each component starts a new thread, it had to be designed in order to not interfere in the construction of the interface. For example, the communication tests didn't have a "Test Communication" button in the beginning. All the tests were loaded at the system startup. It prevented the application interface to be drawn until all the tests conclude, and because of of that a button has been added to activate it.

Figure 4.6 – The final version of the main panel



As soon as the controls (Arduino Capture, Flood Alarm, and Upload to Cosm) are activated they trigger the events they are related to, and the only control that requires a input from the user to work according to what is necessary is the Arduino Capture, which needs the refresh rate and unity to be specified. The results presented by each component are in the following sections.

4.3.2- Arduino Capture

This control sends a HTTP request to the Arduino device, pack the data retrieved adding more information and stores it in the CouchDB. A two weeks continuous test was performed utilizing the control and it retrieved and stored the data flawlessly during the whole period. The tests showed that it was consistent, retrieving the data at the intervals specified in the main application by the user. The figure 4.7 shows the data stored in the database during this test.

Figure 4.7 – Data stored in the CouchDB by the Arduino Capture Control.

Key	Value
"fa9c6059e35a79a47e8abe96010000a0" ID: fa9c6059e35a79a47e8abe96010000a0	{rev: "1-8760c4659e78480f502e855d47c86970"}
"fa9c6059e35a79a47e8abe96010002bb" ID: fa9c6059e35a79a47e8abe96010002bb	{rev: "1-9da038bdfdd32bfe9bd07a83e689c1f"}
"fa9c6059e35a79a47e8abe9601000690" ID: fa9c6059e35a79a47e8abe9601000690	{rev: "1-920317e97c87892cf8aa8cb40c71a2a1"}
"fa9c6059e35a79a47e8abe9601000c1d" ID: fa9c6059e35a79a47e8abe9601000c1d	{rev: "1-a4971a441e844359ae790ec6277656"}
"fa9c6059e35a79a47e8abe960100151d" ID: fa9c6059e35a79a47e8abe960100151d	{rev: "1-93b1cb0caa15e4664bca2070aebc7cdcf"}
"fa9c6059e35a79a47e8abe9601001943" ID: fa9c6059e35a79a47e8abe9601001943	{rev: "1-6c49aa6bc74e401092d1e349dee61725"}
"fa9c6059e35a79a47e8abe96010019d1" ID: fa9c6059e35a79a47e8abe96010019d1	{rev: "1-f0c8b980b542408dbf956d52a40799cf"}
"fa9c6059e35a79a47e8abe9601001d45" ID: fa9c6059e35a79a47e8abe9601001d45	{rev: "1-5767d9ba037399c96938a1699175908f"}
"fa9c6059e35a79a47e8abe9601002b35" ID: fa9c6059e35a79a47e8abe9601002b35	{rev: "1-919ea4fcd8222e2f9462913bc0177eba"}

As stated before, a longer test would be necessary in order to assay the reliability of the Flood Alarm component in a longer period of time; however the results presented after the two weeks test show that the system is stable, working according to its specifications.

4.3.4- Cosm component

During the implementation of the Cosm component some Python APIs indicated in the Cosm website were tested, however no one could provide the functions needed by this project. Because of this, the component that connects and uploads the data from the application's database to the Cosm IoT web service has been implemented from the scratch, utilizing the RESTful interface provided by the service. The httplib and urllib libraries of Python were utilized in the implementation.

Figure 4.9 – The map function utilized by Futon to filter the data

The screenshot shows the Apache CouchDB Futon interface for a database named 'waterlevels'. The 'View Code' section displays a JavaScript map function:

```
function(doc) {
  if (doc.distance) {
    emit(doc._id, doc.distance);
  }
}
```

Below the code is a 'Run' button and a language dropdown set to 'javascript'. A warning message states: "Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application."

The results table shows the following data:

Key	Value
"fa9c6059e35a79a47e8abe96010000a0"	"257"
"fa9c6059e35a79a47e8abe96010002bb"	"280"
"fa9c6059e35a79a47e8abe9601000690"	"280"
"fa9c6059e35a79a47e8abe9601000c1d"	"280"
"fa9c6059e35a79a47e8abe960100151d"	"279"

On the right side, the CouchDB logo and 'relax' text are visible, along with a sidebar containing links to 'Tools' (Overview, Configuration, Replicator, Status, Verify Installation) and 'Recent Databases' (addressbook).

The data uploaded to the Cosm account created for this work was collected during the two weeks test. The data was copied from the Futon application after being filtered by a map function (Figure 4.10) and then copied and pasted to a TXT file that has been treated by a small Python application used to clean the unnecessary data.

Once the data was retrieved from CouchDB to form a package of 50 inputs and sent to the Cosm IoT web service, it was necessary to change its status from "not sent" to "sent". The Figures 4.10 and 4.11 shows the data status in the database before and after being processed by the component.

Figure 4.10– The data status before being processed by the component

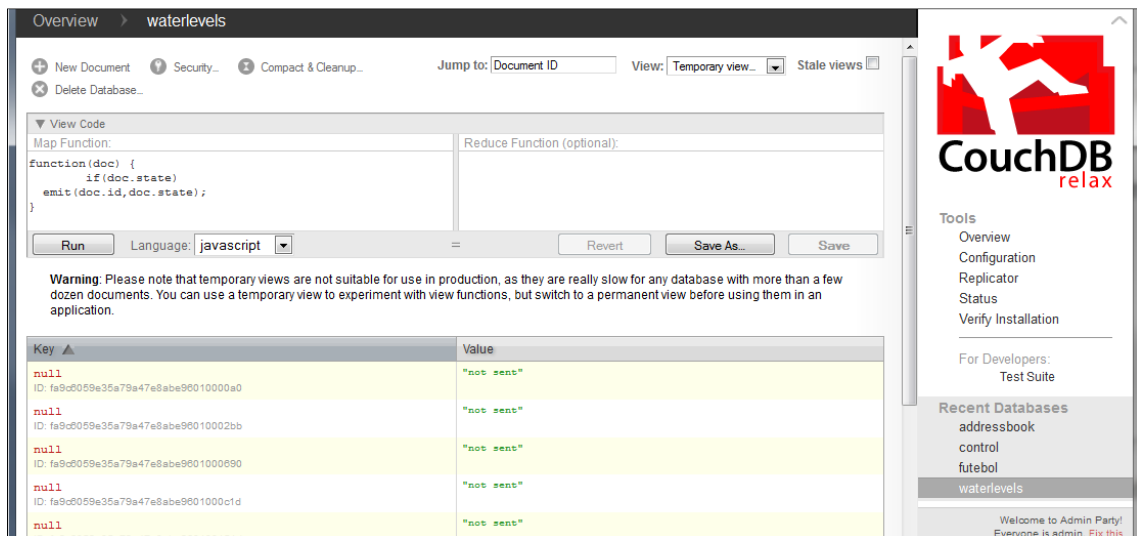
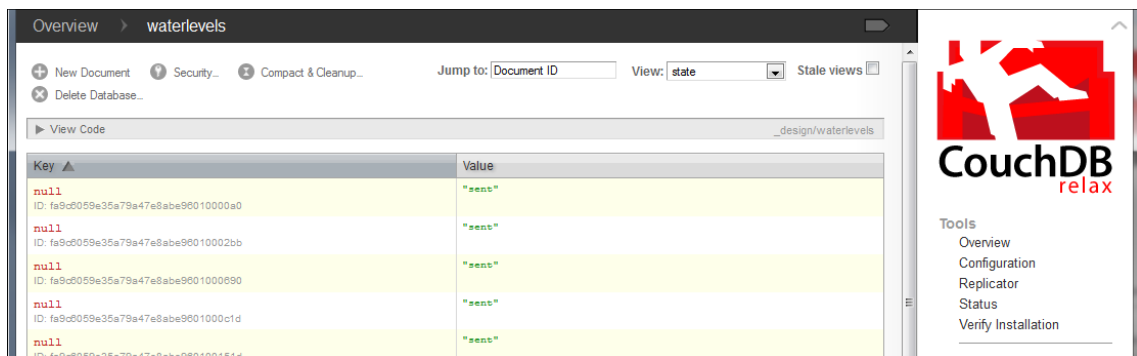
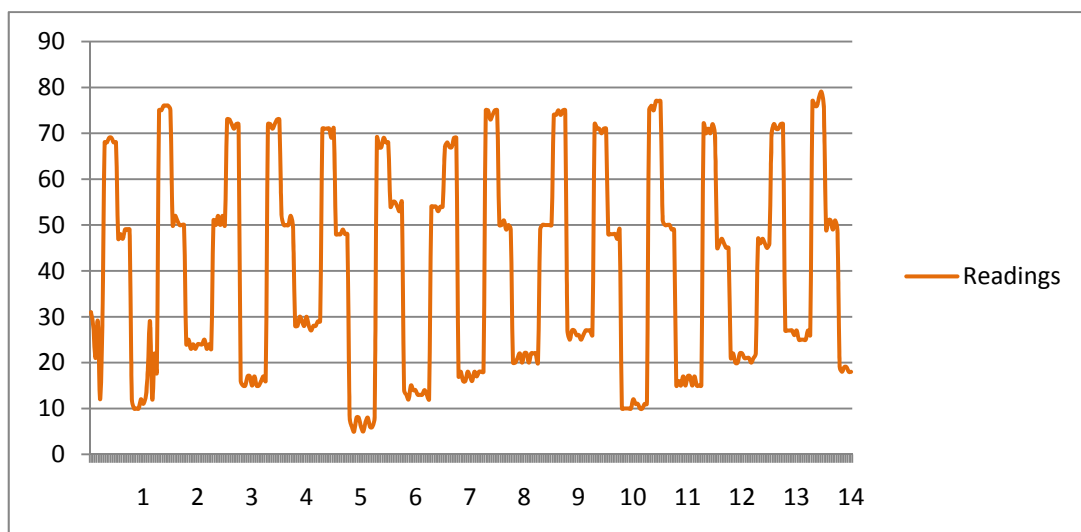


Figure 4.11– The data status after being processed by the component



After this treatment it was pasted into the Microsoft Excel 2007 software in order to plot the graphic the data collected. The Figure 4.9 shows the graphic created by the software.

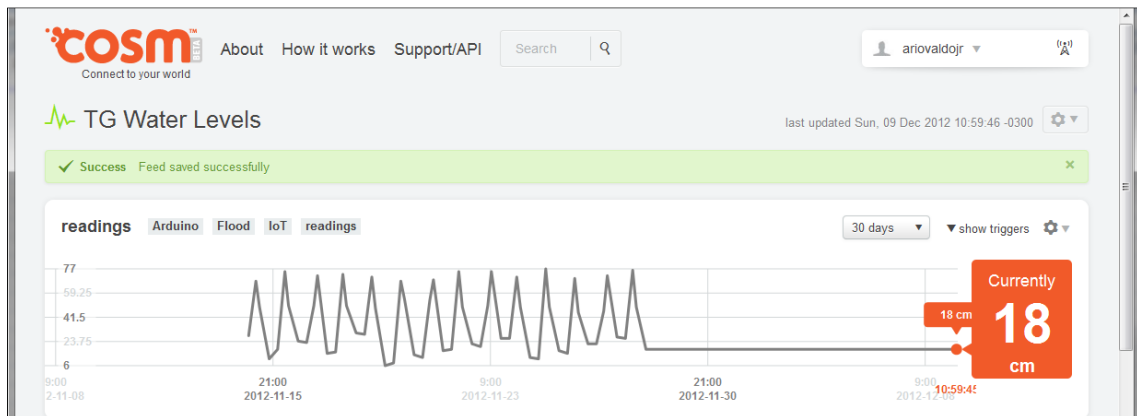
Figure 4.12 – Graphic representing the data retrieved from the database



This graphic was plot for comparison purposes. The Figure 4.13 shows the interface provided by the service plotting a very similar graphic, showing that the Cosm component

worked accordingly to the inputs. The interface provided by Cosm simplifies the graphic, grouping similar points, providing a simple plot. It is not possible to change the period of to another different from what is offered. In this case, a plot for 30 days was chosen, and the data corresponds to the 14 days test period.

Figure 4.13 - Graphic plot by the Cosm IoT web service showing that the data transmission was successful



Once the tool proved to work properly, its code has been posted in the Cosm IoT web service forum (Figure 4.14), being a contribution to the community and an alternative tool for users interested in historical data broadcasting.

Figure 4.14 – Topic created in the Cosm forum, sharing the Cosm component code for implementation by users that utilize Python

The screenshot shows a forum post on the Cosm IoT web service. The post is titled 'Sending Datapoints using Python' and is by user 'ariovaldojr' on December 9th, 2012. The post content includes a Python code snippet for uploading data to the Cosm IoT service. The code uses the urllib and http libraries to send a POST request to the Cosm API. The post also includes a sidebar with a list of forum topics and a search bar.

Sending Datapoints using Python

By ariovaldojr on December 9th, 2012
It worked! Successful projects and code go here.

Dear fellows,

I'm working with Arduino + HTML5 + CouchDB + Python in my final project in college. I tried to find out a good solution to post datapoints in Cosm, however I didn't find anything ready to use. So I came up with this solution:

```
import urllib, httpplib

pachube_api_key = 'YOUR KEY GOES HERE'
pachube_feed = 'http://api.cosm.com/v2/feeds/##FEEDNUMBER##/datastreams/##STREAMNAME##/datapoints.xml'

eeml = "XML CODE HERE"

# upload data
headers = {"X-PachubeApiKey": pachube_api_key}
conn = httpplib.HTTPConnection("www.cosm.com")
conn.request("POST", pachube_feed, eeml, headers)
response = conn.getresponse()
conn.close()
```

I hope this code helps someone, any doubts, please write me :)

Ari

cosm forum [view all topics](#)

ariovaldojr

- About Cosm
- IoT FAQ
- Ways to use Pachube
- Quickstart
- Tutorials/Libraries
- Hardware
- API Documentation
- Press Coverage
- Meetups
- Support
- My account
- Log out

Search

Search this site:

Pachube.blog

- Introducing CosmJS - Cosm Javascript Library
- Visualight LED bulb lets you

This component worked perfectly for the Cosm IoT web service. No input collected was neither lost during the transmission nor transmitted more than one time. When the upload activity was restarted, it uploaded the data correctly, creating the packages it was supposed to. Also it proved to be reliable for utilization during capture process and also in another moment, in sceneries where internet connection is not available (due interruption or non-existence) or even when the user wants to follow a schedule to upload the data to the service.

5- Final Considerations

This work presented a system composed by an electronic device that captures water levels readings and a system that stores this data in a NoSQL database, and then utilizes it in a local alert system. It also feeds the databases of the Cosm Internet of Things web service, being available for consultation in the web. It utilized open source programming languages and prototyping hardware, aiming to reduce the implementation costs and also make it available to the general community. This chapter is divided in two sections. The section 5.1 brings the contributions and conclusions. Section 5.2 points out possible future works that could have the present project as start point.

5.1- Contributions and Conclusions

The contributions reached by this work are:

- It presented a open source prototyping tool Arduino as a viable platform and its utilization;
- A Water Level measuring device was assembled, having data transmission over the network capability;
- A prototype that simulated a natural environment was created in order to test the electronic device that collects and transmits data;
- A database has been modeled and implemented in order to store the collected data;
- It was created an application that analyses the validity of the collected data, represent it graphically in a web interface and emit warning alerts in case the measurement comes out of the acceptable range;
- It was possible to create a component that utilizes the data collected to feed the Cosm IoT platform, which stores data collected from various types of devices and make it available for its users in real time;
- A Python solution to send historical data to the Cosm IoT web service was created and made available to the Cosm community for utilization.

From these contributions it is possible to conclude that:

- The objective of creating a relatively low cost solution to monitor flood occurrences is possible to be reached;
- The module for environment simulation proved to be a reliable source of inputs for this work;

- Arduino prototyping platform is a viable alternative even for definitive projects, once it showed to be very stable;
- Due to the nature of the data collected for this project, CouchDB was indicated for the project, working in key-value pairs;
- The Python programming language proved to be a useful language, able to produce quickly yet consistent results. For a project this one, which dealt almost only with data in string format, this language was ideal, once its tools to handle strings allow a great flexibility for coding;
- Python web libraries (httplib and urllib) proved to work well with the Cosm IoT web service RESTful interface, being able to be implemented 'as is', with no need of further adjustments;
- The application, both as whole and particular components, worked well, under a variety of circumstances, proving to be reliable to be utilized under the purpose for which it was created.

The following experiences were obtained along of the development of this work:

- To work with the Arduino platform added knowledge about hardware components, in special microcontrollers and sensors, and the importance of its utilization in intelligent systems to capture data able to provide a better understanding about environment phenomena;
- Knowledge about NoSQL document oriented databases as well as JSON document structures. The experience obtained utilizing CouchDB as the database platform for this project covered many features of the application, like, map functions, RESTful interfaces, JavaScript, the Python API for CouchDB, and others;
- The advantages of JSON documents over XML documents being utilized as interface between applications, allowing them to communicate to each other properly, and due its structure, process the information faster.
- Better knowledge of Python as programming language, mainly about how to handle strings and how to apply the native web libraries to interact with RESTful applications (like Cosm and CouchDB). A better understanding about implementation of desktop interfaces and its architecture design was acquired as well;
- The construction of the physical model for environment simulation helped to understand better basic concepts of physics applied to the model in order to achieve the results desired. The construction of the model itself also demonstrated to be a challenge. It was necessary

to gather information about how to handle determined tools and plumbing techniques to handicraft it;

- It was necessary to cover topics related to HTML5 in order to implement the Flood Alarm sub-system, which monitored the variations in the data retrieved from the Arduino Water Level Measuring device. For the implementation it was utilized JavaScript to add dynamic behavior to the application, CSS3 for layout design, and raw HTML;
- Problems with APIs and hardware showed that sometimes it is necessary to make course corrections along the project aiming to achieve results faster. It was necessary to discard a network module for the Arduino Water Level Measuring device due to project incompatibility. It was needed to disregard the utilization of two Python APIs for the Cosm IoT web service once they were made for upload data generated at the moment, it didn't cover historical data. Once it was identified the service support was contacted and it was recommended to use HTTP libraries and connect direct to the service APIs. The problem was solved utilizing the Python libraries "httplib" and "urllib".

5.1.1- Publication

A short paper about this work was published in the 14th SICT – Simpósio de Iniciação Científica e Tecnológica organized by Fatec SP:

SOUZA JUNIOR, A.; BERTOTI, G. "Arquitetura de Hardware e Software Open Source para o Monitoramento de Enchentes". Boletim Técnico da Faculdade de Tecnologia de São Paulo, v.34, p. 101, 2012. ISSN 1518-9082.

At this time, once it was not complete, a section about the Cosm sub-system was not included and it creates an opportunity for a possible future additional publication.

5.2- Future Works

The contributions reached by this work do not close the doors for new researches related to this theme, instead creates opportunities for possible future works:

- Recreate the device utilizing a wireless network module in order to provide better usability of the Arduino Water Level Measurement device;
- Perform tests in the real environment, recording the data in a Micro SD card for posterior utilization, or even in real time utilization, utilizing the wireless network module;
- Adapt the system to receive inputs from multiple sources instead of just one;
- Adapt the Flood Alert sub-system to work online, so it could be monitored by distance and by multiple users;

- Utilize the device expansion capabilities to implement other features that would make it work as a weather station, capturing more data and by doing it, provide data to be utilized to improve the understanding of the environment behavior in a particular region;
- Create a desktop application that retrieves the data from the Cosm IoT web service, working in a similar way as the Flood Alert sub-system, or in case of additional features, as an interface for the weather station;
- Utilize the Raspberry Pi to construct a capture unity to store the collected data in the CouchDB;
- Redo the work utilizing the MongoDB platform for data storage, once it utilizes a similar paradigm as CouchDB and compare the platforms point the advantages and disadvantages of each one.

REFERENCES

- ABOUT. *Introduction to Client Server Networks*. Available at <http://compnetworking.about.com/od/basicnetworkingfaqs/a/client-server.htm> Accessed in 10/12/2012.
- ADOBE. *Adobe Flash Player*. Available at <http://get.adobe.com/br/flashplayer/?promoid=JZEFT> Accessed in 12/10/2012.
- APACHE COUCHDB. *Apache CouchDB*. Available at <http://couchdb.apache.org/> Accessed in 03/22/2012.
- APOLO11. *Furacão Catarina: Ciclone ou Furacão Extratropical?* Available at http://www.apolo11.com/furacao_catarina.php Accessed in 12/01/2012.
- ARDUINO. *Arduino boards*. Available at <http://arduino.cc/en/Main/Boards> Accessed in 03/21/2012 (c).
- ARDUINO. *Arduino development IDE*. Available at <http://arduino.cc/en/Main/Software> Accessed in 03/21/2012 (b).
- ARDUINO. *Arduino*. Available at <http://arduino.cc> Accessed in 03/03/2012 (a).
- ARNOLD, C. *El Niño: Stormy Weather for People and Wildlife*. New York: Houghton Mifflin Harcourt, 2005.
- ASHLEY, R.; ASHLEY, R. M. *Advances in urban flood management*. London: Taylor & Francis, 2007.
- BANZI, M. *Getting Started With Arduino*. Sebastopol: O'Reilly Media, Inc., 2009.
- BAOYUN, W. *Review on internet of things*. **Journal of Electronic Measurement and Instrument**. Vol. 23, n. 12, December 2009.
- BASHAM, B.; SIERRA, K.; BATES, B. *Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam*. Sebastopol: O'Reilly Media, Inc., 2011.
- BAXTER, L. K. *Capacitive Sensors: Design and Applications - Volume 1 de IEEE Press Series on Electronics Technology*. Piscataway: John Wiley & Sons, 1996.
- BERNSTEIN, J.; MILLER, R.; KELLEY, W.; WARD, P. *Low-noise MEMS vibration sensor for geophysical applications*. **Journal of Microelectromechanical Systems**. Vol. 8, n. 4, p. 433-438, December 1999.
- BLENDER. *Blender*. Available at <http://www.blender.org/> Accessed in 12/10/2012.
- BOURQUE, L. B.; CLARK, V., A. *Processing Data: The Survey Example*. California: Sage, 1992.
- BRINDLEY, K. *Starting Electronics*. Oxford: Elsevier, 2012.
- BRODIE, M.; WELTZIEN, E.; ALTMAN, D.; BLENDON, R. J.; BENSON, J. M. *Experiences of Hurricane Katrina Evacuees in Houston Shelters: Implications for Future Planning*. **American Journal of Public Health**. Vol 96, n. 5, p. 1402-1408, May 2006.
- BRYANT, R. E.; KATZ, R. H.; LAZOWSKA, E. D. *Big-data computing: Creating revolutionary breakthroughs in commerce, science, and society*. **Computing Research Association**. December 2008.

- BUSINESS INSIDER. *New York And New Jersey Wake Up To Catastrophic Damage*. Available at <http://www.businessinsider.com/hurricane-sandy-map-2012-10>. Accessed in 12/01/2012.
- CAMPOLO, M.; ANDREUSSI, P.; SOLDATI, A. *River flood forecasting with a neural network model*. **Water Resources Research**. Vol. 35, n. 4, p. 1191-1999, April 1999.
- CASSANDRA. *Cassandra*. Available at <http://cassandra.apache.org/> Accessed in 03/21/2012.
- CHAPPELL, D. *Enterprise Service Bus*. Sebastopol: O'Reilly Media, Inc., 2004.
- COSM. *Cosm*. Available at <https://cosm.com/> Accessed 03/03/2012.
- COSM. *Cosm*. Available at <https://cosm.com/> Accessed in 03/03/2012.
- COUCHBASE. *Case Studies*. Available at <http://www.couchbase.com/library?type=Case+Studies> Accessed in 03/26/2012.
- COUCHDB. *Apache CouchDB*. Available at <http://couchdb.apache.org/> Accessed in 03/22/2012.
- COX, P. M.; BETTS, R. A.; JONES, C. D.; SPALL, S. A.; TOTTERDELL, I. J. *Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model*. **Nature**. Vol. 408, p. 184-187, November 2000.
- CROCKFORD, D. *JavaScript: The Good Parts*. Sebastopol: O'Reilly Media, Inc., 2008.
- DANESH, A. *JavaScript in 10 Simple Steps or Less*. Indianapolis: Wiley Publishing, Inc., 2007.
- DAVID, M. *HTML5: Designing Rich Internet Applications*. Oxford: Elsevier, Inc., 2010.
- DAVIS, L.A. *Natural Disasters*. New York: Infobase Publishing, 2009.
- DE SOUZA, M.; CARVALHO, D. D. B.; BARTH, P.; RAMOS, J. V.; COMUNELLO, E.; VON WANGENHEIM, A. *Using Acceleration Data from Smartphones to Interact with 3D Medical Data*. In: 23RD SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI), 2010, Gramado. **Anais...** Gramado, 2010.
- DIG. *Reinventing HTML*. Available at <http://dig.csail.mit.edu/breadcrumbs/node/166> Accessed in 03/26/2012.
- FAIRWEATHER, I.; BRUMFIELD, A. *LabVIEW: A Developer's Guide to Real World Integration*. Boca Raton: CRC Press, 2011.
- FAVBROWSER. *NCSA Mosaic*. Available at <http://www.favbrowser.com/images/mosaic-browser.jpg> Accessed in 12/10/2012.
- FIEDLING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation (Doctor of Philosophy in Information and Computer Science) - University of California, Irvine, 2000.
- FORBES. *The death of Flash and the quiet triumph of open standards*. Available at <http://www.forbes.com/sites/timothylee/2011/09/28/the-death-of-flash-and-the-quiet-triumph-of-open-standards/> Accessed in 03/26/2012.
- FOX, C. G.; MATSUMOTO, H.; LAU, T. A. *Monitoring Pacific Ocean seismicity from an autonomous hydrophone array*. **Journal of Geophysical Research**. Vol. 106, n. B3, p. 4183-4206, 2001.

GASSTON, P. *The Book of CSS3: A Developer's Guide to the Future of Web Design*. San Francisco: No Starch Press, 2011.

GERTZ, E.; DI JUSTO, P. *Environmental Monitoring with Arduino*. Sebastopol: O'Reilly Media, Inc., 2012.

GLOBO.COM. **Teresópolis começa a receber Sistema de Alerta e Alarme contra enchentes**. Available at <http://in360.globo.com/rj/noticias.php?id=21298> Accessed in: 03/03/2012.

GOLDENBERG, S. B.; LANDSEA, C. W.; MESTAS-NUÑEZ, A. M.; GRAY, W. M. *The Recent Increase in Atlantic Hurricane Activity: Causes and Implications*. **Science**. Vol 293, p.474-478, July 2001.

GOLDSTEIN, A.; WEYL, E.; LAZARIS, L. *HTML5 & CSS3 in the Real World*. Collingwood: SitePoint Pty, Limited, 2011.

GOODMAN, D.; MORRISON, M. *JavaScript Bible*. Indianapolis: Wiley Publishing, Inc., 2007.

GOOGLE POWERMETER. *Google PowerMeter: A Google.org Project*. Available at <http://www.google.com/powermeter/about/> accessed in 03/20/2012.

GOURLEY, D.; TOTTY, B.; SAYER, M.; AGGARWAL, A.; REDDY, S. *HTTP: The Definitive Guide*. Sebastopol: O'Reilly Media, Inc., 2002.

HEILMANN, C. *Beginning JavaScript with DOM Scripting and Ajax: From Novice to Professional*. New York: Apress, 2006.

HERSENT, O.; BOSWARTHICK, D.; ELLOUMI, O. *The Internet of Things: Key Applications and Protocols*. New York: John Wiley & Sons, 2011.

HOGAN, B. P. *HTML5 and CSS3: Develop with Tomorrow's Standards Today*. Sebastopol: O'Reilly Media, Inc., 2011.

HUBER, R.; BRODSCHELM, A.; TAUSER, F.; LEITENSTORFER, A. *Generation and field-resolved detection of femtosecond electromagnetic pulses tunable up to 41 THz*. **Applied Physics Letters**. Vol. 75, n. 22, p. 3191-3193, April 2000.

HUGHES, L. *Biological consequences of global warming: is the signal already apparent?* **Trends in Ecology & Evolution**. Vol. 15, n. 2, p. 56-61, February 2000.

IBRAHIM, D. *Microcontroller Projects in C for the 8051*. Burlington: Elsevier, 2003.

IETF. *Hypertext Transfer Protocol -- HTTP/1.1*. Available at <http://tools.ietf.org/html/rfc2616> Accessed in 03/27/2012.

IN360. **Teresópolis começa a receber Sistema de Alerta e Alarme contra enchentes**. Available at <http://in360.globo.com/rj/noticias.php?id=21298> Accessed in 03/17/2012.

INDEED. **Uma busca. Todos os empregos**. Available at <http://www.indeed.com.br/> Accessed in 03/21/2012.

JHA, A. K.; BLOCH, R.; LAMOND, J. *Cities and Flooding: A Guide to Integrated Urban Flood Risk Management for the 21st Century*. Washington DC: World Bank Publications, 2012.

KAMEOKA, J.; CZAPLEWSKI, D. A.; CRAIGHEAD, H. G. *Polymeric Nanowire Chemical Sensor*. **Nano Letters**. Vol. 4, n. 4, p. 671-675, April 2004.

- KANAMORI, H. *Mechanism of tsunami earthquakes. Physics of the Earth and Planetary Interiors*. Vol. 6, n. 5, p. 346-359, 1972.
- KEITH, J. *DOM Scripting: Web Design with JavaScript and the Document Object Model*. New York: Apress, 2005.
- KOZIEROK, C. M. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. San Francisco: No Starch Press, 2005.
- LAINE, J.; TAPALIAN, C.; LITTLE, B.; HAUS, H. *Acceleration sensor based on high-Q optical microsphere resonator and pedestal antiresonant reflecting waveguide coupler. Sensors and Actuators A: Physical*. Vol. 93, n. 1, p. 1-7, August 2001.
- LAWSON, B.; SHARP, R. *Introducing HTML5*. Berkeley: New Riders, 2011.
- LEVINŠTEJN, M. E.; SIMIN, G. S. *Transistors: From Crystals to Integrated Circuits*. Singapore: World Scientific, 1998.
- LUCIDDB. *LucidDB*. Available at <http://www.luciddb.org/html/main.html> Accessed in 03/22/2012.
- LYNCH, C. *Big data: How do your data grow? Nature*. Vol. 455, p.28-29, September 2008.
- MACKENZIE, S. *The 8051 Microcontroller*. Englewood Cliffs: Prentice-Hall, Inc., 1995.
- MALMQVIST, M. *Biospecific interaction analysis using biosensor technology. Nature*. Vol. 361, n. 6408, p. 186-187, January 1993.
- MARGOLIS, M. *Arduino Cookbook*. Sebastopol: O'Reilly Media, Inc., 2012.
- MAYA. *Autodesk Maya*. Available at <http://usa.autodesk.com/maya/> Accessed in 12/10/2012.
- MERELO, J. J.; FERNANDES, C. M.; MORA, A. M.; ESPARCIA A. I. *SofEA: a pool-based framework for evolutionary algorithms using Couch*. In: PROCEEDINGS OF THE FOURTEENTH INTERNATIONAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE COMPANION (GECCO COMPANION '12), 2012, New York. *Anais...* New York, 2012.
- MEYER, J. *The Essential Guide to HTML5: Games to Learn HTML5 and JavaScript*. New York: Apress, 2010.
- MIN. *Obras de Drenagem para prevenção de enchentes*. Available at <http://www.integracao.gov.br/obras-de-drenagem> Accessed in 12/01/2012.
- MIT TECHNOLOGY REVIEW. *How the Internet Became Boring*. Available at <http://www.technologyreview.com/view/428087/how-the-internet-became-boring/> Accessed in 12/10/2012.
- MONGODB. *MongoDB*. Available at <http://www.mongodb.org/> Accessed in 03/22/2012.
- MUELLER, J. *Special Edition Using Soap*. New York: Que Publishing, 2002.
- NICHOLLS, R. J.; HOOZEMANS, F. M. J.; MARCHAND, M. *Increasing flood risk and wetland losses due to global sea-level rise: regional and global analyses. Global Environmental Change*. Vol. 9, n. 1, p. S69-S87, October 1999.
- OXER, J.; BLEMINGS, H. *Practical Arduino: Cool Projects for Open Source Hardware*. New York: Apress, 2009.
- PACIONE, M. *Urban geography: a global perspective*. Abingdon: Routledge, 2009.

- PAPARELLA, M. S.; SIMKO, E. S. *Current Topics in Technology: Social, Legal, Ethical, and Industry Issues for Computers and the Internet*. Boston: Cengage Learning, 2009.
- PETRUZZELLIS, T. *The aAlarm, Sensor & Security Circuit Cookbook*. New York: TAB Books, 1993.
- PREMEAUX, E.; EVANS, B. *Arduino Projects to Save the World*. New York: Apress, 2009.
- PROCESSING. *Processing*. Available at <http://www.processing.org> Accessed in 12/10/2012.
- PYTHON. *About Python*. Available at <http://www.python.org/about/> Accessed in 12/10/2012.
- R7 NOTÍCIAS. **Mais de 900 pessoas morreram e quase 350 ficaram desaparecidas**. Available at <http://noticias.r7.com/rio-de-janeiro/noticias/confira-a-cobertura-completa-das-chuvas-no-rio-20110113.html> Accessed in: 03/03/2012.
- R7 NOTÍCIAS. **Sobe para 414 o número de desalojados em Teresópolis**. Available at <http://noticias.r7.com/rio-de-janeiro/noticias/sobe-para-414-o-numero-de-desalojados-em-teresopolis-20120407.html> Accessed in: 04/12/2012.
- RAUNIO, B. *The Internet of things*. Stockholm: .SE, 2010.
- RED HAT. *Red Hat Linux*. Available at <http://www.redhat.com/> Accessed in 12/10/2012.
- REDIS. *Redis*. Available at <http://redis.io/> Accessed in 03/22/2012.
- REVISTA VEJA. **Tecnologia melhorou reação do Rio à enchente**. Available at <http://veja.abril.com.br/noticia/brasil/tecnologia-melhorou-a-resposta-da-cidade-a-enchente> Accessed in: 03/03/2012.
- RIAK. *Riak*. Available at <http://docs.basho.com/riak/latest/> Accessed in 03/22/2012.
- RICHARDSON, L.; RUBY, S. *RESTful Web Services*. Sebastopol: O'Reilly Media, Inc., 2007.
- RICHARDSON, L.; RUBY, S. *RESTful Web Services*. Sebastopol: O'Reilly Media, Inc., 2007.
- SCHOLZ, C. H. *The Mechanics of Earthquakes and Faulting*. Cambridge: Cambridge University Press, 2005.
- SCHREIER, S. *Modeling RESTful applications*. In: PROCEEDINGS OF THE SECOND INTERNATIONAL WORKSHOP ON RESTFUL DESIGN (WS-REST '11), 2011, New York. **Anais...** New York, 2011.
- SEED. *Seed: Open Hardware Facilitator*. Available at <http://www.seeedstudio.com/> Accessed in 10/12/2012.
- SEN.SE. *Feel. Act. Make sense*. Available at <http://open.sen.se/> Accessed in 03/20/2012.
- SENSORPEDIA. *Sensorpedia: Explore. Contribute. Share*. Available at <http://www.sensorpedia.com/> Accessed in 03/20/2012.
- SLIWKANICH, T.; SCHNEIDER, D.; YONG, A.; HOME, M.; BARBOSA, D. *Towards scalable summarization and visualization of large text corpora*. In: PROCEEDINGS OF THE 2012 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA (SIGMOD '12), 2012, New York. **Anais...** New York, 2012.
- SMITH, K.; WARD, R. *Floods: physical processes and human impacts*. Chichester: John Wiley & Sons, 1998.

- SOMMERVILLE, I. *Software Engineering*. New York: Addison-Wesley, 2007.
- STILLE, D. R.; DAVIS R.; YOUNG JR, T. E. *The Greenhouse Effect: Warming the Earth*. Bloomington: Capstone PressInc, 2006.
- STOLTMAN, J. P.; LIDSTONE, J.; DECHANO, L. M. *International Perspectives on Natural Disasters: Occurrence, Mitigation, and Consequences*. Dordrecht: Kluwer Academic Publishers, 2007.
- STRANZ, E.; BOSELLI, G.; ALENCAR, A. **Desastres naturais no Brasil - Análise da portarias de Situação de Emergência e Estado de Calamidade Pública de 2003 a 2010**. Confederação Nacional de Municípios. Brasília, 2010. Available at http://www.cnm.org.br/index.php?option=com_docman&task=doc_download&gid=134&Itemid=4. Accessed in: March, 3rd 2012.
- SZÖLLÖSI-NAGY, A.; ZEVENBERGEN, C. *Urban Flood Management: Introduction - 1st International Expert Meeting on Urban Flood Management*. London: Taylor & Francis, 2005.
- TEXAS INSTRUMENTS. *The Chip that Jack Built*. Available at <http://www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml> Accessed in 03/21/2012.
- THE TELEGRAPH. *Superstorm Sandy: flood, wind damage and travel chaos in New York City*. Available at <http://www.telegraph.co.uk/news/picturegalleries/worldnews/9643442/Superstorm-Sandy-flood-wind-damage-and-travel-chaos-in-New-York-City.html> Accessed in 12/01/2012.
- THIELEN, J.; BARTHOLMES, J.; RAMOS, M.-H.; DE ROO, A. *The European Flood Alert System – Part 1: Concept and development*. **Hydrology and Earth System Sciences**. Vol. 5 n. 1, p. 257-287, 2008.
- TIWARI, S. *Professional NoSQL*. Piscataway: Wiley, 2011.
- TRIBUNA VARGINHENSE. **Bueiros entupidos: eleições passam, sujeira fica**. Available at <http://www.tribunavarginhense.com.br/2012/10/10/bueiros-entupidos-eleicoes-passam-sujeira-fica/> Accessed in 10/15/2012.
- UBUNTU. *Ubuntu*. Available at <http://www.ubuntu.com/> Accessed in 12/10/2012.
- UCKELMANN, D.; HARRISON, M.; MICHAHELLES, F. *Architecting the Internet of Things*. Berlin: Springer, 2011.
- USUI, K.; KISAKA, M.; OKUYAMA, A.; NAGASHIMA, M. *Reduction of external vibration in hard disk drives using adaptive feed-forward control with single shock sensor*. In: 9TH IEEE INTERNATIONAL WORKSHOP ON ADVANCED MOTION CONTROL, 2006, Istanbul. **Anais...** Istanbul, 2006.
- VAN DEN BROEKE, M.; REIJMER, C.; VAN DE WAL, R. *Surface radiation balance in Antarctica as measured with automatic weather stations*. **Journal of Geophysical Research**. Vol. 109, p. 1-16, 2004.
- VASSEUR, J. P.; DUNKELS, A. *Interconnecting Smart Objects with IP: The Next Internet*. Burlington: Elsevier Inc., 2010.
- VEJA. **Tecnologia melhorou reação do Rio à enchente**. Available at <http://veja.abril.com.br/noticia/brasil/tecnologia-melhorou-a-resposta-da-cidade-a-enchente> Accessed in 03/17/2012.

- W3C. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. Available at <http://www.w3.org/TR/CSS21/> Accessed in 03/26/2012 (e).
- W3C. *Cascading Style Sheets, level 1*. Available at <http://www.w3.org/TR/REC-CSS1/> Accessed in 03/26/2012 (d).
- W3C. *HTML & CSS*. Available at <http://www.w3.org/standards/webdesign/htmlcss.html> Accessed in 03/26/2012 (c).
- W3C. *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. Available at <http://www.w3.org/TR/html5/> Accessed in 03/27/2012 (g).
- W3C. *HTTP - Hypertext Transfer Protocol*. Available at <http://www.w3.org/Protocols/> Accessed in 03/26/2012 (f).
- W3C. *JavaScript script APIs*. Available at <http://www.w3.org/standards/webdesign/script.html> Accessed in 03/25/2012 (b).
- W3C. *World Wide Web Consortium*. Available at <http://www.w3.org/> Accessed in 03/25/2012 (a).
- WERNER-ALLEN, G.; JOHNSON, J.; RUIZ, M.; LEES, J.; WELSH, M. *Monitoring volcanic eruptions with a wireless sensor network*. In: Second European Workshop on Wireless Sensor Networks, 2005, Istanbul. *Anais...* Istanbul, 2005. 1212f.
- WHATWG. *HTML Living Standard*. Available at <http://www.whatwg.org/specs/web-apps/current-work/multipage/> Accessed in 03/27/2012 (b).
- WHATWG. *Welcome to the WHATWG community*. Available at <http://www.whatwg.org/> Accessed in 03/25/2012 (a).
- WHATWG. *What are the various versions of the spec?* Available at http://wiki.whatwg.org/wiki/FAQ#What_are_the_various_versions_of_the_spec.3F Accessed in 03/27/2012 (c).
- WHITE, M. *Information anywhere, any when: The role of the smartphone*. **Business Information Review**. Vol. 27, n. 4, p. 242-247, January 2011.
- WIKIPEDIA. *Netscape Navigator 2.02*. Available at http://en.wikipedia.org/wiki/File:Netscape_2.02.png Accessed in 12/10/2012.
- WILDE, E.; PAUTASSO, C. *REST: From Research to Practice*. London: Springer, 2011.
- WILMSHURST, T. *Designing Embedded Systems with PIC Microcontrollers: Principles and applications*. Oxford: Elsevier, 2007.
- WILSON, J. *Sensor Technology Handbook*. Oxford: Elsevier, 2005.
- YAN, L. *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems*. Boca Raton: Taylor & Francis, 2008.
- YOUTUBE. *YouTube*. Available at <http://www.youtube.com> Accessed in 12/10/2012.
- ZINN, C. *Building a Faceted Browser in CouchDB Using Views on Views and Erlang Metaprogramming*. **Lecture Notes in Computer Science**. Vol. 6816, p. 104-121, January 2011.
- ZOPE. *Zope*. Available at <http://www.zope.org/> Accessed in 12/10/2012.