MONITORING, 3D VISUALIZATION AND CONTROL OF AN IoT INFRASTRUCTURE VIA THE WEB.
CASE STUDY FOR A WASTE WATER HARVESTING SYSTEM.

by

GEORGIA ATSALI

Bachelor Degree in Automation Engineering, Alexander Technological Educational Institute of Thessaloniki, 2007

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2016

Approved by:

Assistant Professor
Spyros Panagiotakis

Abstract

Water is a vital resource for life. Nowadays, the availability of fresh water is a major issue which greatly affects the population. A domestic system which harvests, recycles and distributes for reuse the wastewater (rainfall or residential) has several benefits. The most important of them is the ability to make a building almost water self-sufficient when the public water supply is interrupted. The current work describes an IoT system that monitors, controls and manages the residential water infrastructure that was installed in a block of flats located at an urban area of Crete in southern Greece. This system harvests both residential and rainfall waste water, as black and grey, respectively. After the necessary treatment, the water is forwarded to be reused in toilet cisterns and for garden watering. This thesis presents the Internet of Things (IoT) infrastructure installed at the aforementioned premises but mainly focuses on the web-based client-side application that was developed for the remote real-time monitoring and control of the installed infrastructure. This application provides every tenant with the ability to monitor and control the water consumption of his house at anytime and from anywhere. The application was designed using the responsive Bootstrap framework in order to support a wide range of devices and screen sizes. A 3d scene of the whole building infrastructure was also developed and implemented in the application using the declarative abilities of x3dom react to mouse event clicks and to present in real time data information. The thesis also contributes to the decision-making functionality of the IoT infrastructure, introducing an intelligent controller developed and used for the characterization and separation of the rainwater runoffs as black or grey water.

Keywords: 3D SCADA, Fuzzy Logic, Real-time monitoring and control, X3D, X3DOM

Table of Contents

List of Figures

List of Tables

Chapter 1 - **Introduction**

Water is a vital resource for life. Nowadays, the availability of fresh water is a major issue which greatly affects the population growth in an area. This is directly attached to a number of factors such as urbanization, land use transformation, pollution and climate patterns changing. A domestic system which harvests, recycles and forwards for reuse wastewater; rainfall or residential, has several benefits. The most important of them is the ability to make a building almost water self-sufficient when the public water supply is interrupted. Following this idea, in a block of flats located at an urban area of Crete in southern Greece, a water management system was installed. This system consists of two basic sub-systems; the harvesting and treatment sub-system that collects all wastewater from the building infrastructure producing reusable water, and a distribution sub-system that manages the reuse of the treated water.

The harvesting pipe-line network collects the residential wastewater into two different tanks, to be treated separately as black and grey water. The runoffs of kitchen sinks, toilets and washing machines are considered black water. Grey water consists of the runoffs deriving from bathtubs, showers and washbasins. Although the separation of the residential wastewater is mostly straight forward, the outdoor harvesting part has its challenges. The roof runoff is affected by rainwater quality, roof and/or building features (e.g., length, material, location). Industrial pollution in urban areas, fossil fuels from vehicles and buildings, and/or agricultural activities (pesticides emission) in nearby rural areas can result in rainwater pollution. Additionally, dust, solids, and fecal deposits from birds and rodents, which accumulate on rooftops during dry periods, may also affect the quality of harvested rainwater. Therefore, the first-flush of the roof runoff water; i.e. the first water volume of the rainfall event, may contain pollutants at increased concentrations. An improved quality of the harvested water can be achieved by installing a device (or in this case a controlled valve) which will separate the first-flush water, from the remainder water collection. In current thesis a case study of an intelligent fuzzy logic controller has been implemented to handle the first flush event and it is described in detail in ***Chapter 4 - Rainwater harvesting system.***

On the other hand, the distribution sub-system consists of a pipe-line network which is controlled by an automatic event-based system, as well as an end-user user application. The former collects all monitoring data from the flow meters, tanks' levels and weather station's sensors, and decides if an action must be performed by actuators or users. It also initiates

scheduled processes, such as the watering of the garden located at the roof of the building. All the collected information is stored in a proper designed database. The user interface is a client-side web application for remote monitoring and control of the whole infrastructure. The latter was developed within the context of this thesis and is presented in detail in ***Chapter 5.***

Domènech and Saurí conducted a comparative study of the use of rainwater harvesting in residential buildings, integrating urban planning, social, environmental and economic aspects, in order to detect the challenges and scale up successfully the regulations and incentives. According to them user's motivations in installing such a system are mostly non pecuniary. Residents seemed satisfied with the idea of harvesting rainwater and its positive contribution to the environment and the ecosystem (in particular). Yet, a major concern was raised. Rainwater harvesting may offer numerous benefits but unless residents' knowledge about the system increases, the potential of such system will remain underrated and the risk of abandonment may become real. To that end, an integrated water management system providing an automatic harvesting system and a distribution system fully controlled by a user-friendly interface was considered as the potentially successful solution.

Furthermore, from a broader point of view, many infrastructures and industries use ICT (Information and Communications Technology) systems to remotely control processes and physical functions. These SCADA systems allow a physical system to collect data from sensors and control equipment located at remote areas. The visualization of this data can bring many benefits to control systems. It can assist users in analyzing and comprehending large volumes of data, and also detect patterns, clusters and outliers that are not obvious when non-graphical forms of presentation are used. Several studies argue that energy savings can be achieved by providing more frequent or real-time data to the decision-makers. However, most SCADA systems are still using proprietary technologies (Microsoft Silverlight, Adobe Flash, etc) which cannot run without cumbersome plug-ins or extensions, requiring a thick client installation which rises the user's cost.

On the other hand, Internet of Things (IoT) technology can take advantage of already installed devices or/and use a variety of low-cost and low-power internet-connected devices. Developing an IoT system with a web-based application as a UI, can lower the cost in both hardware and software needs, but also reduce the cost of using specialized personnel for maintenance and customization. The evolvement of JavaScript, CSS3 and HTML5 made web browsers capable to handle some of the workload of the servers. This allows developers to create more client-side web applications, to manage a greater part of the application within

the browser itself. This approach provides the application with direct control over the interface without the need to load the entirely page from the server.

Taking into account the aforementioned remarks, the user interface that was developed in the context of this thesis for our water management system, is an interactive web-based responsive application that provides a user-friendly and platform independent framework for monitoring and control of the infrastructure. The application is developed with open source technologies to provide both a modular and cost effective solution. It is designed to inform the user of the current state of every component of the system, the water consumption, historical analytics of the water consumption, the ability to create watering schedules for the roof garden, etc. This framework also provides role-based functionality, recognizing the logged-in user and displaying a personalized web-view of the application and appropriate information, e.g. when the tenant of an apartment logs-in, the application provides only the content for this apartment and current user's info. The application also provides an interactive page with a 3D scene of the building infrastructure for real-time monitoring and control of the IoT system. The 3D model constructed in the context of this thesis is unavoidable tailored to the needs of the specific installation. However, much care have been taken to keep it as generic and modular as possible, so it can fit any other installation or can be easily extended with new features. Our ultimate goal is with this real-time 3D web application to enable a new way on how big data of sensor networks are utilized and visualized. Definitely, with 3D, controlling and monitoring of grid networks become natural and user friendly, also more flexible and powerful, since any part of the installation is revealed and viewed.

EpIgrammatically, the contribution of this thesis can be summarized as follows:

- Creates a web-based SCADA UI for an IoT system.
- Provides a modular framework for real-time data acquisition and control with 3D visualization of all system's components.
- Provides a responsive 3D scene, for more intuitive control by human operators, and many valuable comfort features, such as zoom, navigation and different perspectives.
- Efficiently integrates various open source technologies to reduce cost, both for software products and hardware utilities, and
- Provides a framework for the seamless integration of X3Dom graphics with IoT and big data.

Chapter 2 - **Related Work**

Water is a vital resource for life. Nowadays, the availability of fresh water is a major issue which greatly affects the population. This is directly attached to a number of factors such as urbanization, land use transformation, pollution and climate patterns changing. The water demand particularly in urban areas has a common characteristic worldwide; it continues to grow over time. This creates a water scarcity problem which is a serious challenge to manage.

Based on [1] a domestic rainwater harvesting system has several benefits. Firstly the treated rainwater is consumed at, or near, the point of production, reducing the possibility of contamination during delivery, and of course lowers the publicly-supplied running costs. A significant amount of water demand can be replaced through the increase of rainwater harvesting system installations. Furthermore, the rainwater harvesting systems can be flexibly installed in both new and existing buildings. The building can be almost water self-sufficient when the public water supply is interrupted. So the rainwater harvesting systems can be used as water preventative systems to control water quantity.

In literature many studies have been performed about the benefits of harvesting, recycling and reuse of wastewater; rainfall or residential. In [2] Abdulla and Al-Shareef evaluate the reuse of harvesting rainwater in residential sectors of Jordan for potable water saving and provide recommendations regarding the improvement of the quality and quantity of harvested rainwater. They concluded that a maximum of 15.5 Mm^3/y of rainwater can be collected from the residential roofs of Jordan, but generally it can be used for secondary purposes. Rainwater harvesting can provide a water source to increase water supplies and also involve the public in water management, making the water management everybody's business.

Why IoT for water management

Many infrastructures and industries use ICT (Information and Communications Technology) systems to remotely control processes and physical functions. These systems, known as Supervisory Control and Data Acquisition (SCADA), allow a physical system to collect data from sensors and control equipment located at remote areas [3]. More specifically, these systems handle the data transfer between a central computer host, a number of Remote Terminal Units (RTUs) and/or Programmable Logic Controllers (PLCs), and the operator

terminals. A SCADA system gathers information, transfers it to a central site, alerts the home station that an incident has occurred (carrying out the necessary analysis and control) and displays the information in a logical and comprehensive form. SCADA systems can vary from relatively simple, e.g. a system that monitors environmental conditions of an office building, or very complex, e.g. a system that monitors all activities in a nuclear power plant [4].

Most developed water management systems are context specific, since they depend on many factors like the resources, climate, users etc. This means that the water saving solutions derived for one organization may not work for another [5]**.** A problem about information and communications technology (ICT) based water management systems is the lack of ICT standards which prevents an effective interoperability and also increases the cost of developing new products and its maintenance [6]. Currently, most ICT systems for water management are proprietary, where its producers have to support all production levels [6]. This difficulty of developing a complete system discourages both producers and buyers.

An IoT (Internet of Things) implementation in water management system can provide some benefits on this topic. IoT technology is widely accepted all around the world, mainly because it can take advantage of already installed devices or/and use a variety of low-cost and low-power internet-connected devices. The typical structure of an IoT system is shown in Figure **2**.**1**. An IoT system consists of a number of sensors and actuators which measure changes in the environment and control functions, respectively. These measurements (data) and commands are sent via a network to and from a main control system. This system also provides an application where data or alert messages can be displayed, and control commands can be sent. This type of visualization can provide the user with a better understanding and full control over several devices.

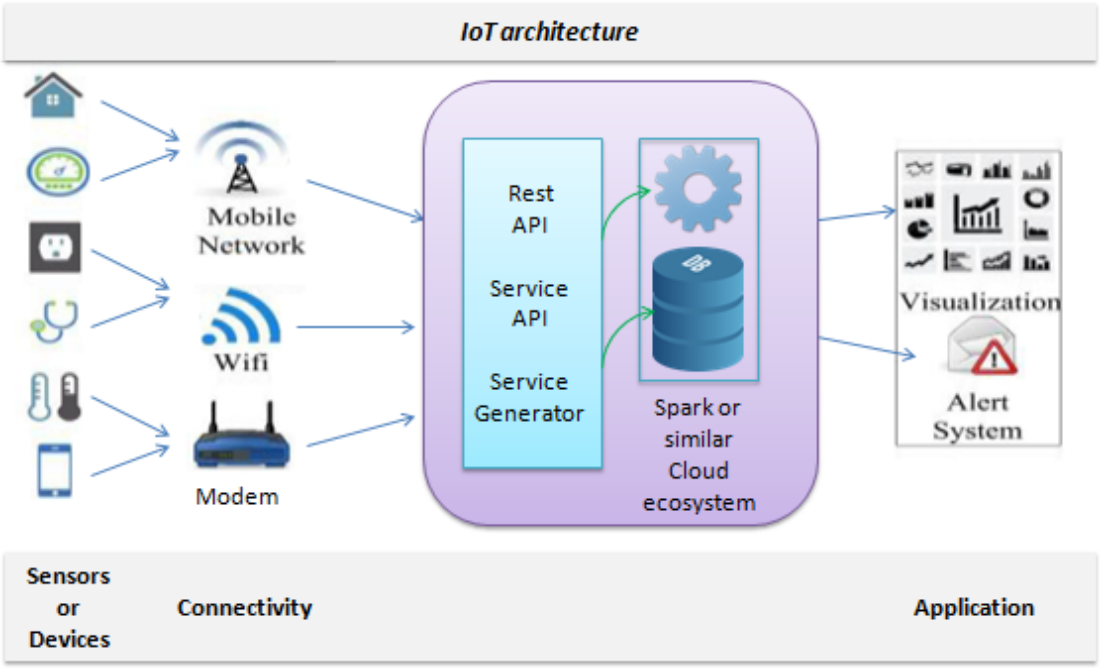


Figure 2.1 IoT architecture

According to [7] IoT systems can benefit water management due to several assets that they bring. The main benefits are:

Efficiency: they can provide real-time operational control to improve decision making and reduce operating costs. They can use real-time data retrieval from sensors and actuators to monitor and improve water management infrastructures, increasing the efficiency, by reducing the energy costs and minimizing human interventions.

Cost savings: IoT systems can improve the asset utilization (e.g., use water irrigation units without manual operation), the process efficiencies and productivity (e.g., remote monitoring of irrigation conditions).

Productivity: is a critical parameter affecting the efficiency and profitability of any company or organization. IoT allows real-time control, process optimization, service time reduction, resource conservations, and the capability to do all of this globally, reducing the difference between required and available skills and improving labor efficiency.

IoT builds on three pillars: Internet orientation, Thing orientation, and Knowledge orientation. We connect these pillars with the ability of objects to be identifiable (every "thing" identifies itself), to communicate (every "thing" communicates) and to interact (every "thing" interacts), either among themselves, or with the end-users or other entities in the network [7].

Authors in [8] propose an IoT-based architecture for monitoring and control of water utilities. They highlight all the aspects that an IoT system has to take under consideration, such as real-time monitoring, scalability, connectivity and security issues, as well as the support for dynamic environments.

According to [9] water management systems can benefit from internet information systems when they are used for decision-making and detailed tasks. At the same paper authors propose an interactive system which combines geo-informatics, enterprise information system, and cloud service for a water resource management.

Data Visualization

Data visualization, i.e. the presentation of data in a pictorial or graphical format, is now becoming properly appreciated due to the benefits it can bring to control systems. The visualization of data has the prospective to assist users in analyzing and comprehending large volumes of data, and also detect patterns, clusters and outliers that are not obvious when non-graphical forms of presentation are used. For this reason, data visualization will play an important part in a range of applied problems, including data mining and exploration, information retrieval and intelligence analysis [10].

Several studies in engineering areas argue that energy savings can be achieved by providing more frequent or real-time energy data to the decision-makers [11]. However, limited efforts have been done into the effective communication of this data. For a wider and complex indexing set, the actual visualization and arrangement of the individual graphical elements is a critical step in attracting the operator's attention [11].

In literature, visualization in wastewater treatment is often associated only to methods of processing data and doesn't discuss its on-site deployment. Research about the communication of data from waste water treatment Plants (WWTPs) to operators has been limited. Publications like [12] and [13], discuss general rules for the visualization of temporal data which may be transferable to wastewater practices.

Authors in [11] present a novel approach to precise and intuitive visualization of performance indicators, considering energy and process data on WWTPs. This approach deals with graphical and computational issues, but also takes into consideration the integration of heterogeneous data sources, computational requirements, as well as the variation of data

quality and quantity. With this approach, authors aim to support operators in evaluating and optimizing the performance of the plants.

Although remote data sensing and data processing supplement manual data recording and analyze practices, few visualization tools exist that collect data from dynamic resources and stream it in real-time to a field-realistic environment [14]. Authors in [14] present a real-time tracking and visualization framework, which contains real-time data collection, data processing, visualization, and an application for construction that simulates live and training environments. They conclude the view provided by the 3-dimentional display improved situational awareness of users and allowed camera views from multiple locations.

Real-Time data acquisition and monitoring

Although real-time sensor feeds can transform both decision making and environmental science, they are rarely used in real-time workflows, analyses and modeling tool chains [15]. In spite of the large benefits deriving from the use of these feeds, the complexity and the limited number of existing real-time platforms across these domains are the main obstacles to the adaptation of real-time data [15]. Usually, the data interactions are carried out through direct user requests to the system, with no or limited mechanisms to automatically notify the users of new readings or events as they occur [15]. Furthermore, such schemes are rarely designed to enable alerts or access to field-deployed sensors or actuators, thus they limit their use in decision making and control-centric applications [15]. Many of these systems are mainly designed for domain specific applications, limiting their use in a wider set of domains.

Another important obstacle that the end-user must face when using these systems is the need for him to implement and host locally these real-time systems [15]. This introduces deployment and maintenance complexities, in addition to deploying and maintaining field sensors and actuators, i.e. program the firmware and maintain the hardware [15].

Data platforms impose significant requirements in the form of system architectures, programming languages, operating systems, and sensing platforms, this makes their deployment labor intense, even for users who already maintain sensor networks for continuous data [15]. In order to reduce implementation overhead compliance to a physical protocol has been proposed. In the study of Díaz et al. 2013 [16], authors suggest that sensors should conform to a standard hardware interface, and particular to an Ethernet port.

Modern sensor systems can connect to the Internet via standard web protocols; this permits the use of web services as a perfect interoperability mechanism between sensors, actuators and decision support systems [15]. Embedded web-servers with advanced features (such as server push for event notifications or concurrent connections), can be implemented with no operating system support and very low memory capacity, thanks to efficient cross-layer TCP/HTTP optimizations, and therefore can run on tiny embedded systems [17]. Since embedded web-servers in an IoT system generally have fewer resources than web-clients (i.e. browsers, mobile phones, tablets) Asynchronous JavaScript and XML (Ajax) has proven to be a good way of transferring the server workload to the clients [18].

Web applications

Web technologies have experienced a rapid growth over the last years. In particular, the improvement of JavaScript, CSS3 and HTML5 enhanced web browsers' abilities. Richer web-based software solutions are available with an increasing range of functions. With responsive web design (RWD), a technology for displaying content for different screens without resizing it, developers are able to support a wider range of diverse devices with small effort [19].

With the evolvement of html5, css3 and javascript, web browsers can handle some of the workload of the servers. This allows developers to create more client-side web applications to manage a greater part of the application within the browser itself. This approach provide the application with direct control over the interface without the need to load the entirely page from the server. This can be performed with the help of Asynchronous JavaScript and XML (AJAX) communications which allow client-side web applications to request raw data from the server and update the page content from the browser, improving the response time, as it does not need the page to be refreshed [20].

Compared to classical native or desktop applications, a web based application has many advantages [21].

- **Accessibility** – With an internet connection a web-based application is accessible from anywhere at any time and from any device.
- **Scalability** – A web application can easily be scaled adding or removing pages of content without affecting the performance or the quality of any other page.

- **Portability** – A web application can run in any web browser on any OS without the need of re-compile. It's not restricted to a computer but it can run on any device that has a browser, e.g. a mobile phone or a tablet.
- **Maintenance** – A web application doesn't have to be installed in each client's computer thus maintenance it's easier. The updates are made through the server which sends the updated scheme instantly to all clients.
- **Controlled Access** – With user logins a web-based application can configure user groups or roles with different permissions. These permissions can be easily changed on the server.
- **Licensing** – The majority of web technologies like html, css, javascript, ajax etc are open source and free to use, resulting in a significant reduce of licenses needed.

SCADA systems

A Supervisory Control and Data Acquisition (SCADA) system collects information, transfers it to a central site, carries out any necessary analysis and/or control, and then displays the information at operator's screen [22]. The required control actions are then passed from the operator back to the process. According to [22] a typical SCADA system includes the following components:

- **Sensors** in the field or in a facility that measure **conditions**, e.g. water level or flow, temperature or humidity, etc.
- **Operating equipment** such as valves, pumps, etc. which can be controlled by activating actuators or relays.
- **Local processors** communicating with the sensors and the operating equipment. This includes the Programmable Logic Controller (PLC), Process Automation Controller (PAC), Intelligent Electronic Device (IED) and Remote Terminal Unit (RTU). A single processor can be responsible for dozens sensor inputs and operating equipment outputs.
- **Short range communications.** Usually the communication between the local processors, the sensors and the operating equipment is established using relatively short cables or wireless connections. These communications carry analog and discrete signals with electrical characteristics, such as voltage and current, or with other industrial communications protocols.

- **Host computers** used as the central point of monitoring and control. These computers are where human operators can supervise the process; review data, receive alarms and exercise control from the Human Machine Interface (HMI).
- **Long range communications.** Local processors and host computers are usually connected via leased phone lines, cellular packet data, satellite, etc.

Data Visualization in SCADA interfaces

Web technologies improved immensely after 2000, especially with the World Wide Web Consortium (W3C) which demonstrated a case study for standardizing specifications instead of proprietary products. The adaptation of standard and open specifications allows a healthy competition between organizations that can benefit the end user (e.g. a variety of browsers to select from). Also, crossed browser content can further open the competition area to mobile devices, such as smartphones and tablets, and benefit all users by lowering the cost of ownership. Opposed to W3C open standards, some SCADA vendors are still using proprietary technologies (Microsoft Silverlight, Adobe Flash, etc) which cannot run without cumbersome plug-ins or extensions, requiring a thick client installation which rises the user's cost [23].

It has been observed that most of those using W3C standards, were limited to 2D-models or pictures of technical components, to visualize the automation system process and real-time data. For example, Ecava develops products (e.g. IntegraXor Web SCADA [24]) which use SVG image library for the presentation of automation symbols. Rapid SCADA [25] is also a web based free and open source system software which uses 2D representation into the monitoring area; it also needs proprietary software for extended uses, such as automatic control module and charting components [26].

When data is provided in a 2D form, live shapes, processes, spatial cognition and other details are lost or displayed in a very abstract way. Considering the complexity of automation systems and the amount of corresponding data, a 3D-image is technically the best way to display these data [27]. A 3D representation has the advantage been more intuitive for human operators and have the ability to provide many valuable comfort features, such as zoom, navigation or/and different perspectives [28].

Following that concept many SCADA systems have been adapting 3D modeling to represent the HMI's, but most of them are not web based. GraphWorX™64 [29] provides the ability to create a 3D world but with the proprietary software Windows Presentation

Foundation. Another example is the RIVOPS (Remote Intuitive Visual Operations) console provided from an engineering firm called EATOPS (Ergonomics Applied to Operations). This console is a customized combination of software and hardware, which includes both 2d and 3d displays [30]. Scada3D Solutions [31] also provides custom made 3D models for any type of industrial company.

In order to enhance the human machine interaction, a 3D-monitoring page was developed that visualizes all the control and information units in real-time. It visualizes the technical components of water flow consumption information and the status of operating equipment in real time; these components have been installed on a four apartment building.

Local control unit

This section focuses on the HMI of the developed system and especially in the 3D model of the building infrastructure created to provide real-time monitoring and control to the end user. In order to supervise and control the waste water infrastructure, a real-time data responsive user interface developed. This interface should follow three basic specifications. Firstly, it had to be platform and device independent, so that the user won't be restricted to a specific operating system or screen size, respectively. Furthermore, the interoperability between the main control unit and the HMI should be considered in order to achieve real-time data transfer. The last requirement was the graphical representation of the whole building infrastructure, capable to show all data flow and control components in a comprehensive and easy to use way for the end user.

Reviewing the literature we discovered that most SCADA, visualization and process control systems use the OPC[1] software interface standard to provide data from the source to the client application in a standard way. OPC is based on the DCOM/COM, a platform dependent model developed by Microsoft for LAN connections. In order to reduce the cost of installing dedicated SCADA networks, researchers turned to Web-based SCADA systems. In [32] H.A. Abbas & A.M. Mohamed conducted a classification of all the different approaches to access an OPC DA server through the Internet, that were introduced in the literature, as well as their feasibility on the realization of these approaches. They concluded that none of these

[1] OPC stands for OLE (Object Linking and Embedding) for Process Control, also for Open Process Control [32].

approaches succeeded to achieve a real time behavior which is vital in the functionality of all SCADA systems.

Although the need for web-based HMI's is present, the combination between industrial operating equipment and web technologies doesn't seem to be the best combination. Furthermore it is evident that the limitations of closed protocols contradict our own specifications for open-free software. On the other hand, the utilization of single board computers has been documented in a number of monitoring and control works in a variety of fields, including environmental applications [33], [34], smart artificial vision systems [35] or smart appliances [36] and smart cities [37], with promising results. Authors in [33] present a wireless sensor network system which uses Raspberry Pi and Arduino for environmental monitoring applications. They note that such a design is useful in many monitoring and data collection applications, in the bases of a low-cost, scalable, easy to customize, easy to deploy, and easy to maintain system, but also has a major advantage to integrate all the system components under a single control board; database server, web server and control unit, which can be easily configured to run headless (i.e., without monitor, keyboard, or mouse).

Based on these concepts we developed a system which collects measurements of weather conditions, water flows and water levels, performs an automatic control and broadcasts a web application, with the collected information and the control interfaces via an internet connection, to any kind of device. The web server providing the web application is also installed on the single board computer, a Raspberry Pi.

Chapter 3 - **System Architecture**

In a block of flats located at an urban area of Crete in southern Greece, an Internet of Things (IoT) system was installed [38]. This system monitors, controls and manages the residential water infrastructure. It collects all water-waste from the four apartments located at the building, and from the outdoor surface (roof). The waste-water is dividing into two separate and airtight tanks based on its source (grey or black, see Figure 3.1). After the harvesting cycle, the grey and black water is filtered and promoted for reuse on garden watering and toilet flushes.

Figure 3.2 presents the architecture of the MCS which shows all the subsystems as well as the flow-meters and actuators installed at the building complex. Categorizing all the different components we have:

- **Piping network:** The waste water for every source, except the roof, is forwarded to be treated as black or grey according to the piping network installed. The roof rainwater cannot be treated solely as grey or black there for the direction of this pipe is controlled by a dedicated module on the main control system.
- **Wastewater treatment:** An autonomous, mostly analog, wastewater treatment system collecting the waste water through the pipes, perform the filtering treatment and fill the dedicated for reuse water tanks with the corresponding from grey and black water. This system provides the level values of the available recycled grey and black water to the main control system.
- **Sensors**: Throughout the infrastructure many flow-meters have been installed to make possible the measurement of the different consumptions at every part of the building. These measurements are collected, processed and forwarded to the MCS.
- **Actuators**: Every apartment has its own electro-valve providing its tenants with the option of selecting whether their apartment will consume fresh water or recycled grey water. To control the separation of the harvest rainwater of the roof; to be treated as black or grey, an electro-valve is also needed. A part of the roof accommodates a garden which is watered with black, grey and fresh water. In order to make the watering automatic three electro-valves were installed. All these actuators are controlled by a set of relays which are initiated by commands send from the MCS.

- **Weather station**: A weather station which consists of two parts, a set of sensors located at the roof of the building, and a data logger, which receives the sensors' measurements via a wireless connection and is located at the main control panel.
- **Main control system:** All the subsystems mentioned above as well as a user interface are controlled by a single board controller. This controller handles the events as well as the data flow from every part of the infrastructure, and also hosts a web-based application to provide tenants with an interactive remote control system.

The IoT system is equipped with a single board computer (Raspberry Pi), a data-logger (Arduino), a wireless weather station (Aercus Instruments WS3083) and several sensors and actuators. The Pi controller runs on a Linux minimal operating system capable of enhanced functionality. The source code for reading data from the sensors inside Arduino is written in C. For processing and submitting the information to the MySQL database a Python script is executed. The database as well as a Tomcat server are hosted on the controller and provide a web-based application for remote control. Figure 3.3 and Figure 3.4 display the hardware connections and the corresponding equipment, respectively.

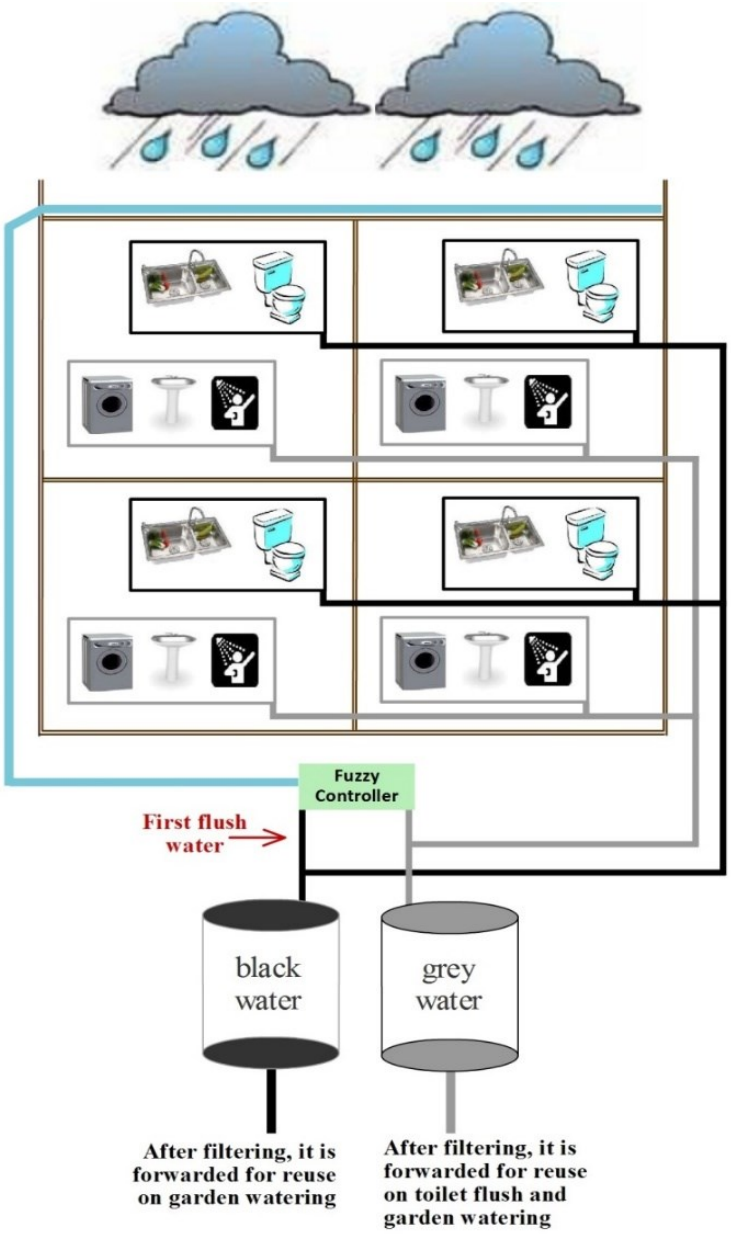


Figure 3.1 Waste-water harvesting system

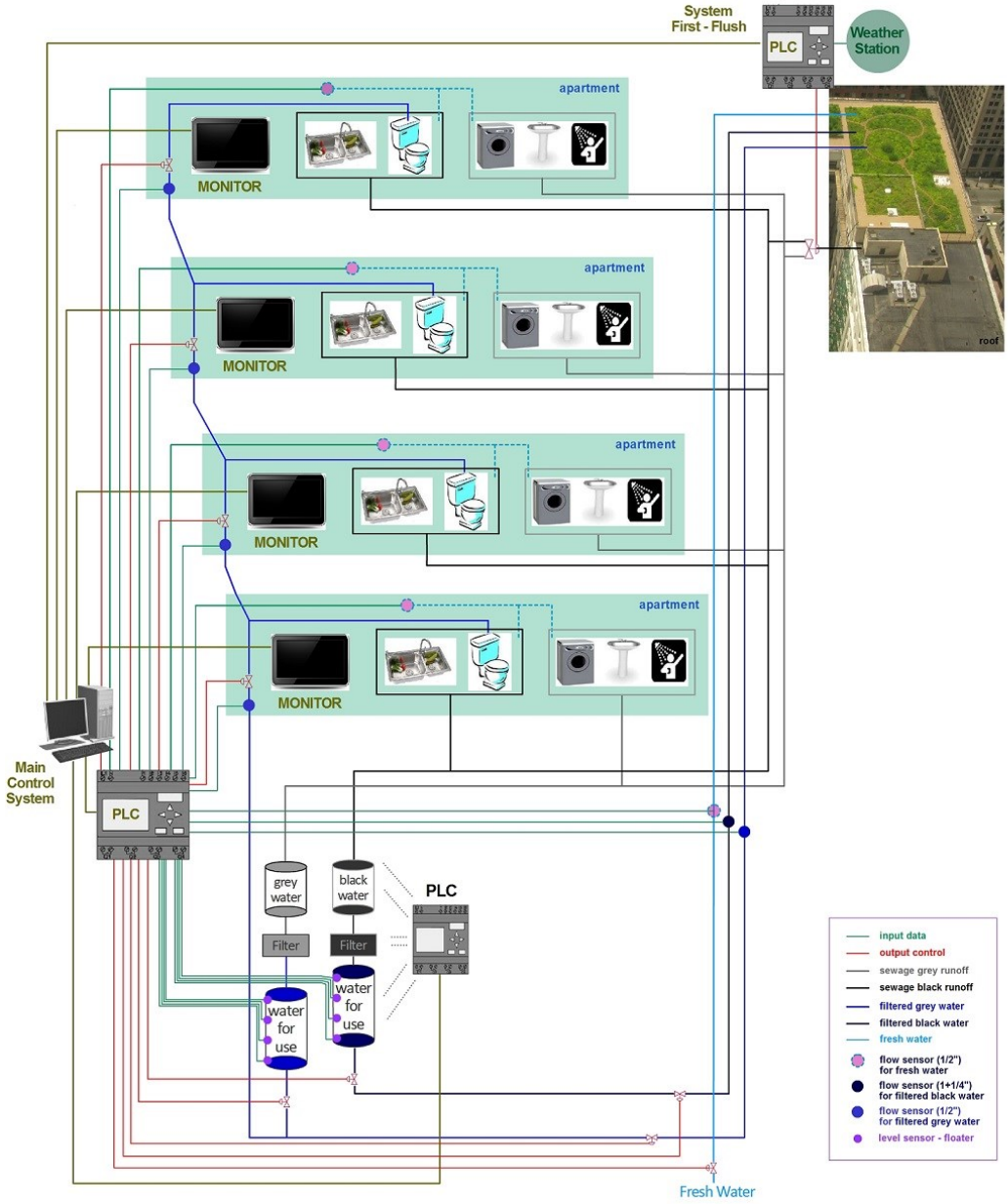


Figure 3.2 Main Control System's Architecture

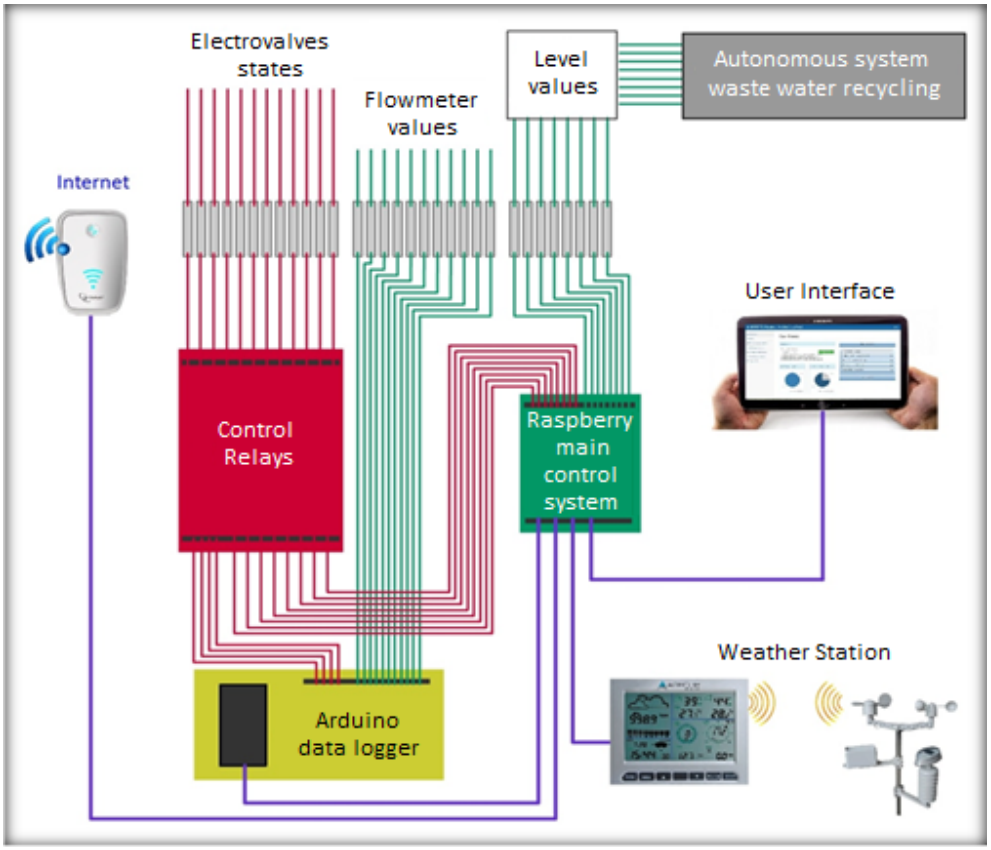


Figure 3.3 Hardware connections

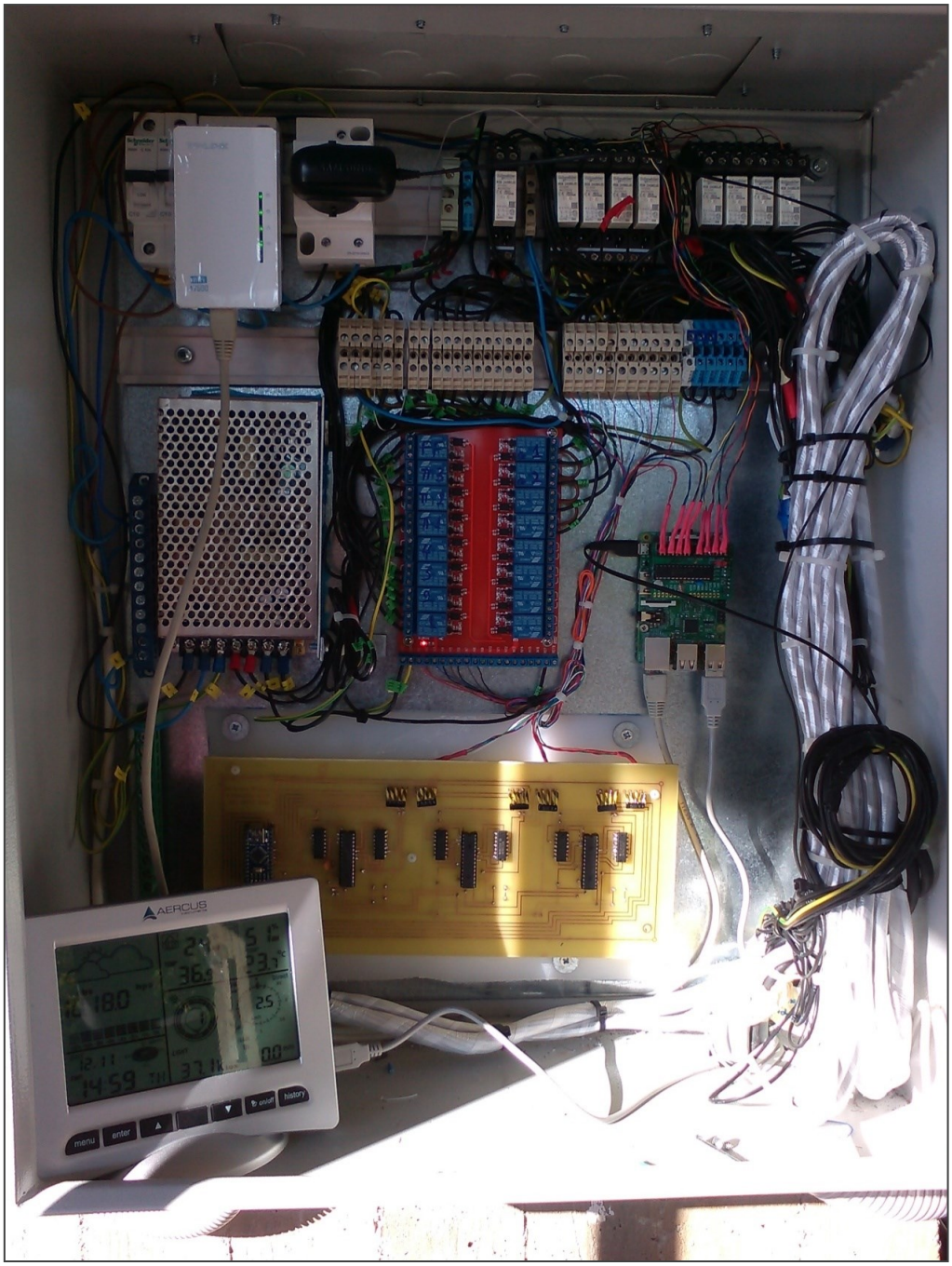


Figure 3.4 Hardware equipment

Main Control System (MCS)

The MCS is a Raspberry Pi single board computer which controls all parts of the building infrastructure and hosts a web application for providing tenants with interactive and personalized control. A Python script is continually running on the raspberry OS providing automatic control over the building. This control is based on events derive from real time data,

as well as scheduled functions. Pi also runs a server hosting a web application for the tenants to control their own part of the building. Three modules are handling the monitoring and control of this system. The first concerns the low level interaction with the system; sensors' data, actuators etc, the second the roof rainwater runoff harvesting system and the last the web application.

The first module hands the real-time data and initiates several sub-functions. The basic function on this module collects, processes (if needed) and saves the sensor data to the database, runs constantly. This function also checks few conditions in order to start the other sub-functions. When the level of the tanks are low a sub-function is called to cut all filtered water supplies and automatically enable the fresh water; this may prevent equipment failure, aridity at the toilet cisterns and/or other problems. The second module is initiated when the rain starts. This module handles the rainwater harvesting system which decides in which tank to forward the roof runoffs[2]. Whenever a change is made by this module to the actuators the corresponding database tables are updated so that the web application displays the correct status. The last function on this module is enabled when a message is sent from the web application. Every time a user makes a specific change, the application sends a message through a java web-socket to inform the system that a change has occurred. Depending on the message, different tasks are executed e.g. change water supply for a specific apartment to fresh, or submit new watering schedule.

The third module is a Web Application developed to provide control for individual users, i.e. tenants. This application was developed with a responsive template (http://getbootstrap.com/css/), so that the user won't be restricted to an operating system or device, it is web-based so that can be accessed remotely and provide personalized content based on the specific logged-in user. Specifically, when a user is logged in, the app displays his one apartment's data, status and control pages. Only the building's administrator (super) has full control over the flat complex[3].

[2] This module is discussed in detail in ***Chapter 4 - Rainwater harvesting system***

[3] This module is discussed in detail in ***Chapter 5 - User Interface - Web Application***

Database Architecture

In a large system like this, a database storing all information is necessary. The selected database for the current project an installed MySQL Server Community Edition which is an open-source relational database management system (RDBMS). For the web application a database called SiteDB was created to store all the information displayed at the application (consumption data, tank levels, starter status, etc.) and that derived from it (user's personally identifiable information, apartments' consumption profiles, etc.). Figure 3.5 shows the relation scheme between the tables of this database.

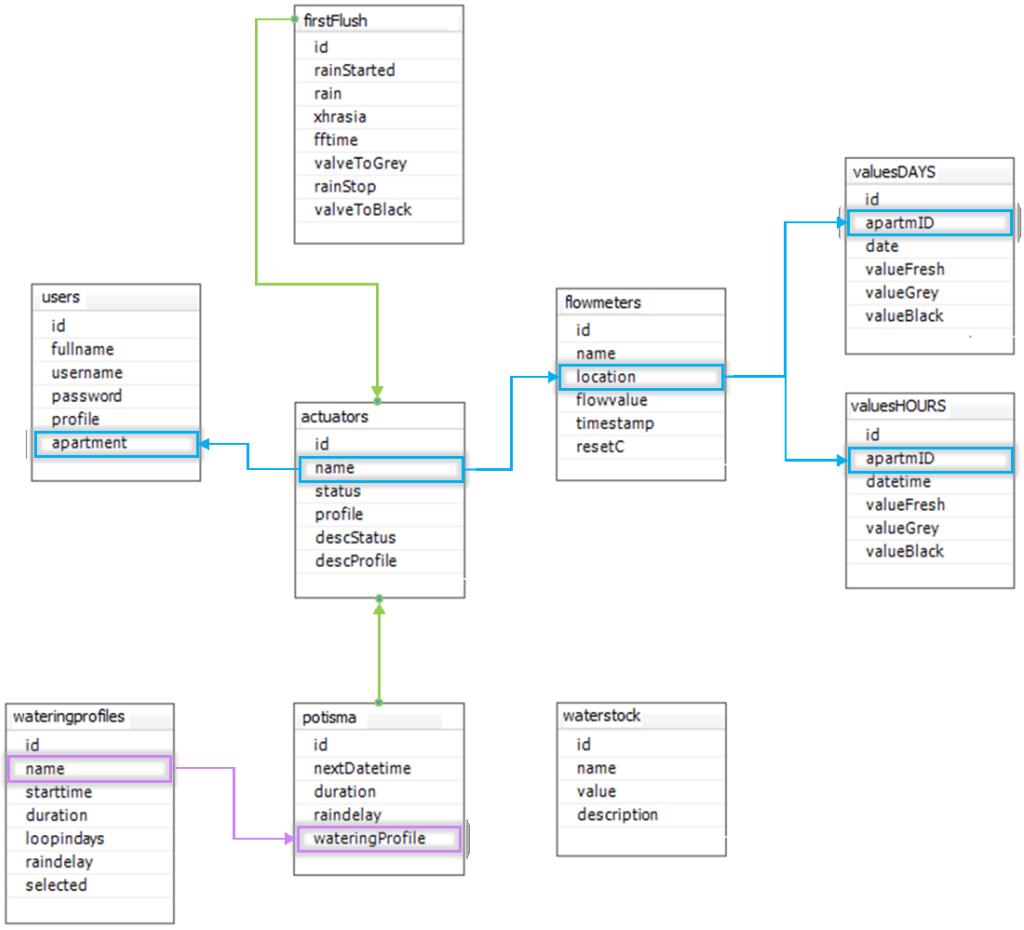


Figure 3.5 Database connections architecture

Chapter 4 - **Rainwater harvesting system**

The availability of fresh water is a major issue which greatly affects the population growth in an area. This is directly attached to a number of factors such as urbanization, land use transformation, pollution and climate patterns changing [39]. Many solutions have been proposed, with much interest in the use of roof-collected rainwater. Rainwater harvesting is an ecological and tolerable method of water management, resulting in the reduction of urban runoff and flooding, and of course water saving [40]. Using rainwater for various human needs retrenches the fresh water that would be used for fulfilling the same needs if not recycling it.

The roof runoff is affected by rainwater quality and, roof and/or building features (e.g., length, material, location). Industrial pollution in urban areas, fossil fuels from vehicles and buildings, and/or agricultural activities (pesticides emission) in nearby rural areas can result in rainwater pollution [41]. Additionally, dust, solids, and fecal deposits from birds and rodents, which accumulate on rooftops during dry periods, may also affect the quality of harvested rainwater [42]. Therefore, the first-flush of the roof runoff water; i.e. the first water volume of the rainfall event, may contain pollutants at increased concentrations. An improved quality of the harvested water can be achieved by installing a device (or in this case a valve) which will separate the first-flush water, from the remainder water collection [43].

At the building complex there was installed a roof rainfall harvesting system, which uses fuzzy logic to convert the direction of the runoff water, between two tanks [44]. The purpose is to separate the first flush rainwater from the remainder, so that can be treated as black and grey water respectively.

Analytically, to improve the quality of the roof harvested water, a solenoid valve has been installed at the runoff tube drainage and controlled with fuzzy logic to separate the first-flush water; i.e. the first water volume of the rainfall event (containing pollutants at increased concentrations), from the remainder harvest water. Considering the surface features (length, material, location - near highway) and some extended experimental data, we reached to a model which will receives two input variables, the rainfall intensity and the drought duration, and produces the time after which the first flush event must end.

The first input of the fuzzy controller is the rainfall intensity which is given in millimeters per hour (mm/h) in real time from a weather station located at the building. The second input is a bit more complex. In the Mediterranean area where the building is located, the duration between two rain events may vary from five minutes to three months. As a result

we cannot accept as dry period the time interval between two rain events. This could result to a wrong calculation of the first flush time. Considering this fact and the cement base of the roof surface, as drought duration we calculate the time passed between a “clean” roof surface, i.e. when the valve changed status as an indication of a completed first flush event, and the new rain event.

When a rain event starts, the fuzzy controller receives the values of these two inputs (rainfall intensity and drought duration) and returns a time value (in minutes - output) in which the current first flush event needs to be concluded.

For the design and analysis of the fuzzy controller we used Matlab/Simulink. The same principles were then applied with java coding, using the open source Java library jFuzzyLogic [45], so that it can be implemented to the larger wastewater management system of the building. Figure 4.1 shows the java class executing the FIS file which contains the Function Block interface with all the required elements.

```
public class ValveControl
{
    public static void main(String[] args) throws Exception
    {
        // Load FCL file
        String fileName = "holistic.fcl";
        FIS fis = FIS.load(fileName,true);
        if (fis == null)  //Error while loading the file
        {
            System.err.println("Can't load the file: '" + fileName + "'");;
            return;
        }

        //Get Fuzzy System
        FunctionBlock fuzzyBlock = fis.getFunctionBlock(null);

        //Set Inputs
        fuzzyBlock.setVariable("duration", 40);
        fuzzyBlock.setVariable("rainFall", 3);

        // Evaluation
        fuzzyBlock.evaluate();

        // Get output value
        double result = fuzzyBlock.getVariable("valveTime").defuzzify();

        System.out.println("Result: " + result);
    }
}
```

Figure 4.1 Java class handling the fuzzy logic consept

Fuzzy Logic Controller (FLC)

The first flush concept has been implemented using a Mamdani fuzzy logic controller. A fuzzy logic controller relates the output to the inputs using rules; i.e. a list with IF-AND/OR-THEN statements. The IF-AND/OR-part of the rules refers to the linguistic names of certain regions (fuzzy sets) of the input variables. Each input value belongs to these regions represented by a certain degree of membership [46]. The logical AND has been implemented with the minimum operator, and the THEN-part of the rules refers to values of the output variable. To acquire the output of the FLC, the degrees of membership of the IF-AND/OR-parts of all rules are evaluated, and the THEN-parts of all rules are calculated by the geometric center of the area (or center of gravity), under the membership functions and within the range of the output variable. Figure 4.2 presents the structure of some of the rules used in this FLC. Figure 4.3 illustrates an example of the calculation of the center of the gravity used for two rules WITH different weights (i.e. the last two rules on Figure 4.2).

Table 4.1 shows the general setting made to the fuzzy logic controller, as well as the membership factions' names and parameters.

```
IF  drought IS week1  AND  rainFall IS I75  THEN  valveTime IS delay0  WITH 1;
IF  drought IS week1  AND  rainFall IS I115 THEN  valveTime IS delay0  WITH 1;
IF  drought IS week1  AND  rainFall IS I55  THEN  valveTime IS delay2  WITH 1;
IF  drought IS week1  AND  rainFall IS I15  THEN  valveTime IS delay8  WITH 1;
IF  drought IS week1  AND  rainFall IS I15  THEN  valveTime IS delay12 WITH 0.2;
IF  drought IS week2  AND  rainFall IS I15  THEN  valveTime IS delay8  WITH 1;

IF  drought IS week2  AND  rainFall IS I95  THEN  valveTime IS delay0  WITH 1;
IF  drought IS week2  AND  rainFall IS I95  THEN  valveTime IS delay2  WITH 0.6;
```

Figure 4.2 Fuzzy system – Rules

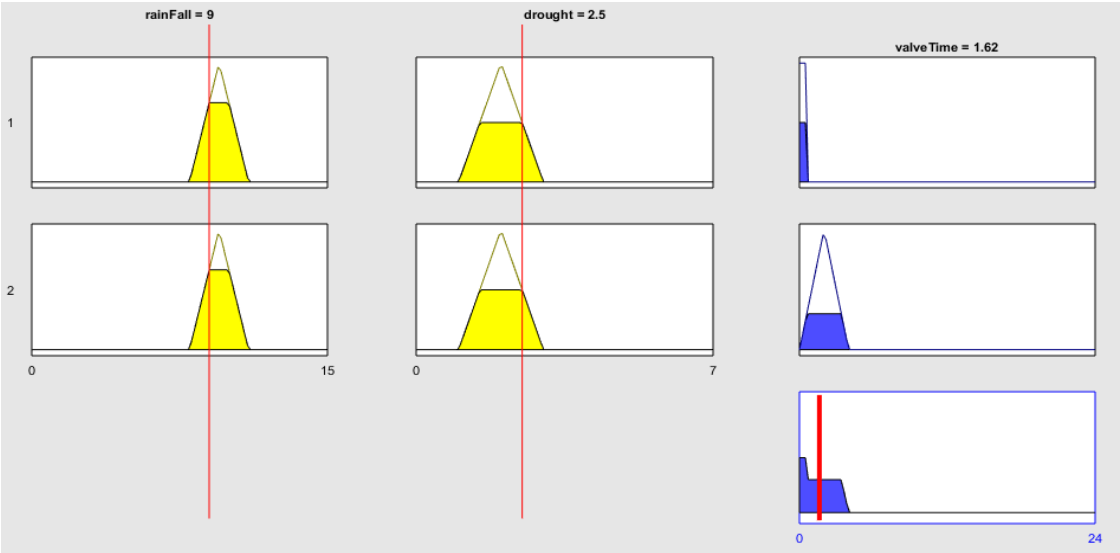


Figure 4.3 Fuzzy system - Center of Gravity

Variable	Name	Range	Number of member-ship functions
Input	rainFall	[0 15]	9
	drought	[0 7]	6
Output	valveTime	[0 24]	13
Defuzzification method:			Centre of area
Antecedent connective:			And (Minimum)
Implication:			Minimum
Minimum Degree of support:			1.00

Table 4.1 Fuzzy system - General Setting

It is worth noting that the shape and number of the membership functions of each fuzzy set, as well as the inference mechanism of the fuzzy logic, was selected based on trial-and-error methods, in a way that the region of interest of the input data is covered appropriately. The selected shapes of the membership functions, associated to the FLC linguistic variables, are piece-wise linear functions (triangular or/and trapezoidal). Table 4.2, Table 4.3 and Table 4.4 contain the analytic settings of each membership function and its fuzzy sets, and Figure 4.4, Figure 4.5 and Figure 4.6 present the graphical representation of these functions.

Membership function	Shape	Points
I08	trapezoid	0 ; 0 ; 0.5 ; 1.2
I15	triangle	0 ; 1.5 ; 3
I35	triangle	2 ; 3.5 ; 5
I55	triangle	4 ; 5.5 ; 7
I75	triangle	6 ; 7.5 ; 9
I95	triangle	8 ; 9.5 ; 11
I115	triangle	10 ; 11.5 ; 13
I135	triangle	12 ; 13.5 ; 15
I140	trapezoid	14 ; 15 ; 15 ; 15

Table 4.2 Fuzzy system - Input membership function rainfall

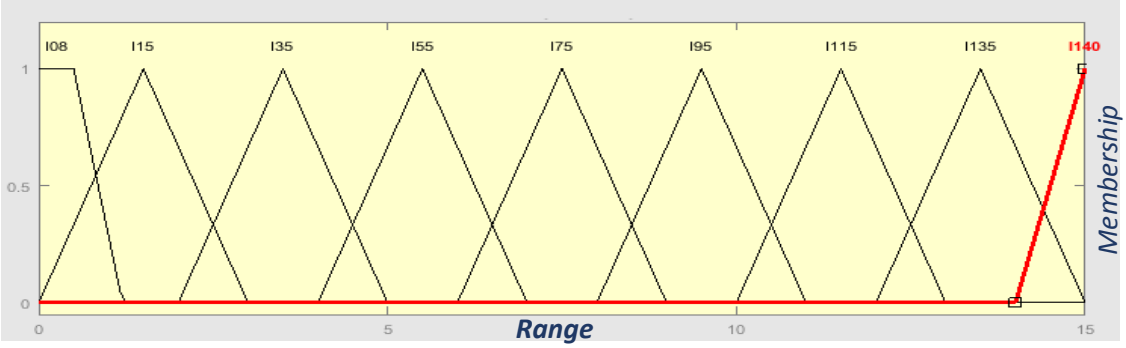


Figure 4.4 Fuzzy system - Input membership function rainfall

Membership function	Shape	Points
week1	trapezoid	0 ; 0 ; 1 ; 2
week2	triangle	1 ; 2 ; 3
week3	triangle	2 ; 3 ; 4
week4	triangle	3 ; 4 ; 5
week5	triangle	4 ; 5 ; 6
week6	trapezoid	5 ; 6 ; 7 ; 7

Table 4.3 Fuzzy system - Input membership function drought

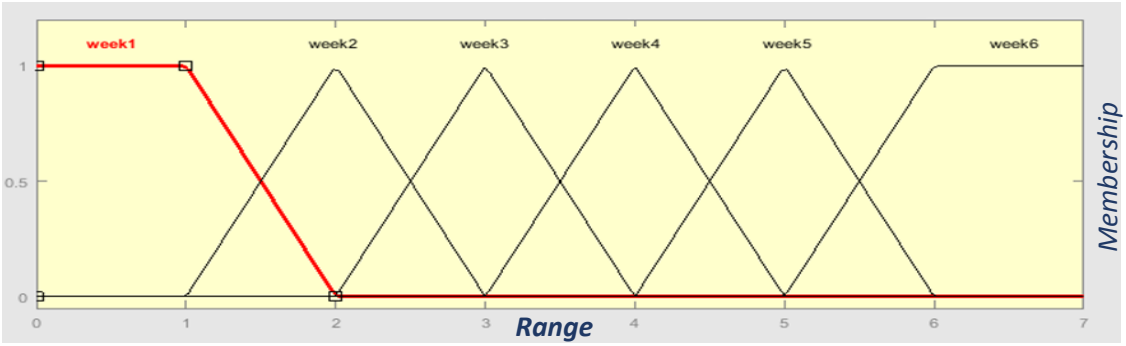


Figure 4.5 Fuzzy system - Input membership function drought

Membership function	Shape	Points
delay0	trapezoid	0 ; 0 ; 0.5 ; 0.7
delay2	triangle	0 ; 2 ; 4
delay4	triangle	2 ; 4 ; 6
delay6	triangle	4 ; 6 ; 8
delay8	triangle	6 ; 8 ; 10
delay10	triangle	8 ; 10 ; 12
delay12	triangle	10 ; 12 ; 14
delay14	triangle	12 ; 14 ; 16
delay16	triangle	14 ; 16 ; 18
delay18	triangle	16 ; 18 ; 20
delay20	triangle	18 ; 20 ; 22
delay22	trapezoid	20 ; 22 ; 24 ; 24
delay24	trapezoid	24 ; 24 ; 24 ; 24

Table 4.4 Fuzzy system - Output membership function valveTime

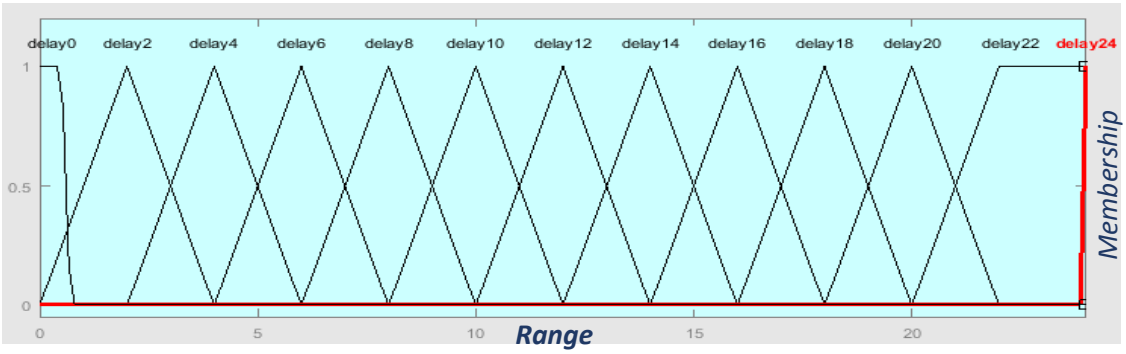


Figure 4.6 Fuzzy system - Output membership function valveTime

Figure 4.7 illustrates the control surface of the fuzzy logic controller. In more detail, it shows the connection between the inputs (rainFall and drought) and the output (valveTime) that derives from the base of knowledge, i.e. the rules used for this FLC. The vertical contour of the plot indicates the value of the output variable, and the two horizontal the values of the two inputs. This surface helps developers to understand the behavior of the controller.

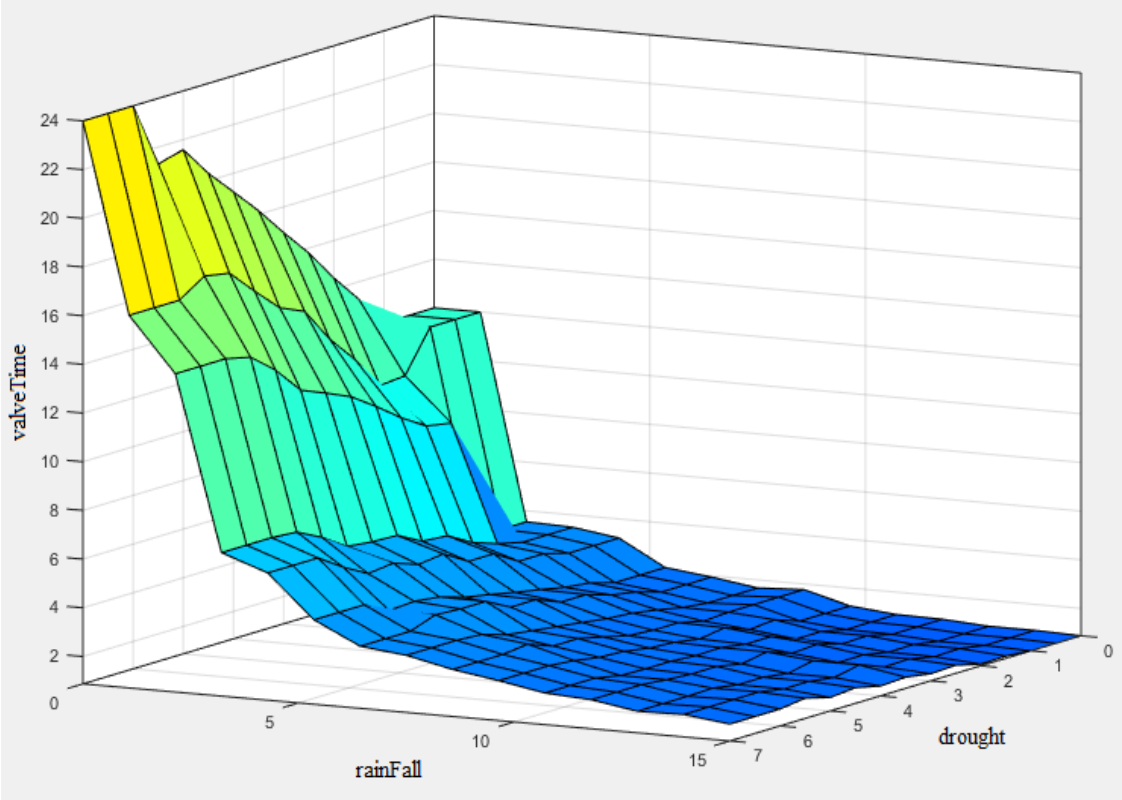


Figure 4.7 Fuzzy system - Control surface based on the rules

Rainfall Harvesting Algorithm

The fuzzy controller as part of a larger MCS, is initiated when a condition is true; in this case when rain starts. When the rain sensor receives indication of rain, the main system calculates the drought time and initializes the fuzzy controller with these two inputs. When the FLC returns the time value (output), the system forwards the first flush runoff towards the black water tank and starts a countdown. When the time passes (i.e. the first flush ends), it changes the valve's status towards the grey tank. When rainfall stops valve status returns towards black tank for the next rain.

All these actions are stored to a specific table in a database (Figure 4.8). Each row of this table consists of all relevant information regarding one full cycle of a rainfall event. It begins with a timestamp of the rain-start, and the rain quantity that is falling (submitted in mm per hour). Then the system calculates the time difference between today and the last date that the roof surface was clean, to get the drought time (in days), and it adds it to the dedicated column. The fuzzy system is initialized with those two input values and returns the time (in minutes) in which the first flush event will conclude and it's added to the row too. When this time elapses, the dedicated electro-valve changes its direction of the harvest water toward grey tank, so a timestamp of this action is also added to the table. When the rain stops, i.e. when the rain sensor of the weather station returns a value of 0mm/h, this cycle ends. A timestamp of the rain-stop event is added and the electro-valve returns towards the black tank adding the last value in the table and concluding the event. The later action prepares the electro-valve state for the next rainfall event.

In lines 1535 and 1536 (Figure 4.8) the column "valveToGrey" does not have values. This can happen when the rain stops before the first flush event is completed. In this case the draught time of the next rain event (line 1537) is calculated with the last completed event, i.e. when the roof was cleaned (line 1534).

id	rainStarted	rain	xhrasia	fftime	valveToGrey	rainStop	valveToBlack
1532	2016-02-11 05:00:00	4.5	16	5	2016-02-11 05:05:01	2016-02-11 05:25:00	2016-02-11 05:25:01
1533	2016-02-11 14:39:00	1.5	0	0	2016-02-11 14:39:01	2016-02-11 14:58:00	2016-02-11 14:18:01
1534	2016-02-13 18:13:10	3.7	2	3	2016-02-13 18:16:10	2016-02-13 19:33:10	2016-02-13 19:33:10
1535	2016-02-28 07:43:56	2.7	13	6	NULL	2016-02-28 07:47:41	2016-02-28 07:47:42
1536	2016-02-28 19:12:09	2.3	14	6	NULL	2016-02-28 19:14:17	2016-02-28 19:14:17
1537	2016-03-04 09:58:01	6.8	18	2	2016-03-04 10:00:01	2016-03-04 10:23:16	2016-03-04 10:23:16
1538	2016-03-06 22:29:45	3.1	2	3	2016-03-06 22:32:45	2016-03-06 22:47:12	2016-03-06 22:47:12
1539	2016-03-11 06:19:32	2.1	5	4	NULL	2016-03-11 06:21:41	2016-03-11 06:21:42

Figure 4.8 Log file – A full cycle run for a rain event

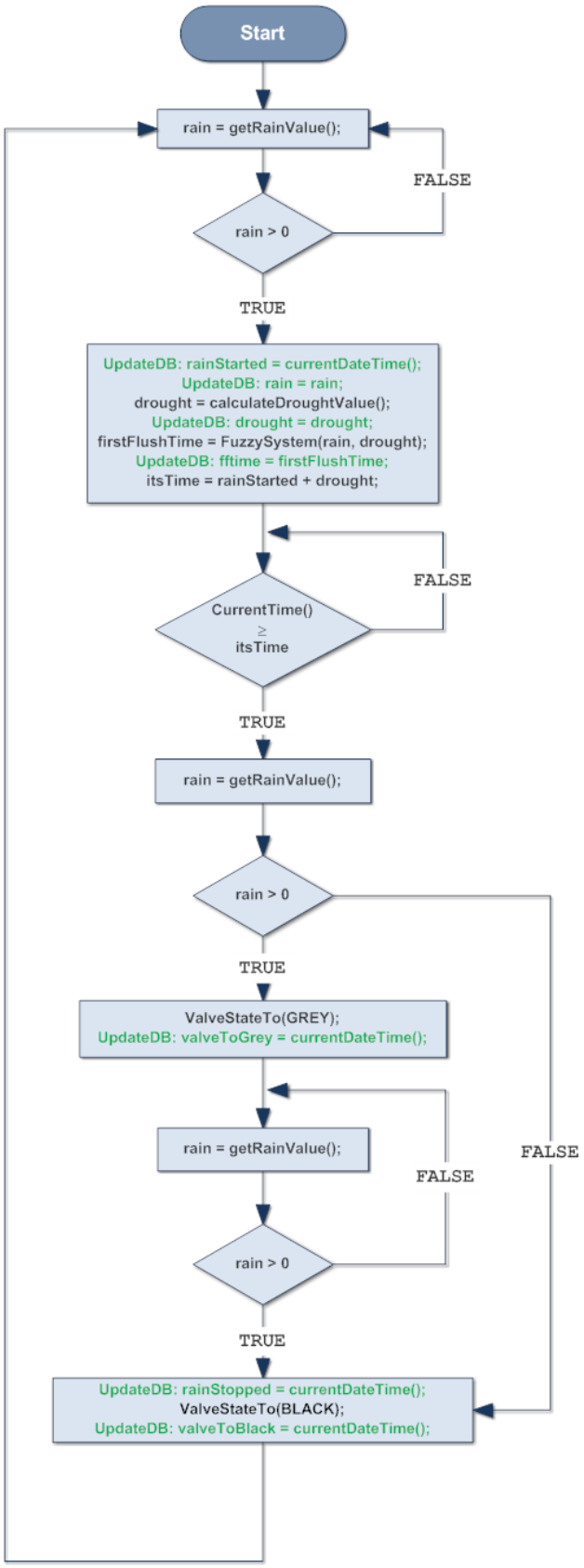


Figure 4.9 Logic chart of roof rainfall harvesting algorithm

Evaluation of the Harvesting System

Every system has to be evaluated to deduce the quality of its service. For this roof rainwater harvesting system an evaluation of the first flush event must be implemented. That means that both the Fuzzy Logic Controller and the actual water quality of the separated proportions must be evaluated. The former can be monitored through the event log (database table) where all the events of this sytem are submitted; Figure 4.8 shows an instance of that table. Studing the submitted data we can infer that the fuzzy implementation runs smoothly and with the expected results.

On the other hand there is no ICT way to test the quality of the separated water quantities. Thus in a period of a year (in which this IoT system is in use) a great amount of randomly selected water samples were collected for analysis. These samples were chemicaly analysed in order to be calculated the time after which the first flush event should have happened. These time intervals were then compared to the FLC's times and the results are juxtaposed in Table 4.5.

Studing these results we calculate a mean deviation of under half of a minute (0,43), that is an acceptable error value. Specifically because even when we expect the smallest time response of the FLC, i.e. when we have the smallest drought period and larger rain intensity, an extra half minute does not significantly affect the water balance in the treated black water tank. This system's feasibility lies in the simplicity of its implementation.

a/a	Drought (week)	Water runoff (lt)	Rain Intensity (mm/h)	**Experimental Estimation of time (min)**	**Fuzzy Logic Estimation of time (min)**	**Deviation (min)**
1	1	10	1,5	8,00	8,90	0,90
2	1	10	2,5	4,80	5,97	1,17
3	1	10	3,5	3,43	3,46	0,03
4	1	10	4	3,00	3,26	0,26
5	1	10	6	2,00	2,14	0,14
6	1	10	8	1,50	1,61	0,11
7	1	10	10	1,20	1,33	0,13
8	1	10	12	1,00	1,11	0,11
9	1	10	14	0,86	0,97	0,11
10	1	10	15	0,80	0,93	0,13
11	2	12	1,5	9,60	9,47	0,13
12	2	12	2,5	5,76	7,42	1,66
13	2	12	3,5	4,11	4,11	0,00

14	2	12	4	3,60	4,08	0,48
15	2	12	6	2,40	2,50	0,10
16	2	12	8	1,80	2,00	0,20
17	2	12	10	1,44	1,62	0,18
18	2	12	12	1,20	1,33	0,13
19	2	12	14	1,03	1,13	0,10
20	2	12	15	0,96	0,93	0,03
21	3	14	1,5	11,20	11,18	0,02
22	3	14	2,5	6,72	8,19	1,47
23	3	14	3,5	4,80	4,83	0,03
24	3	14	4	4,20	4,72	0,52
25	3	14	6	2,80	2,87	0,07
26	3	14	8	2,10	2,20	0,10
27	3	14	10	1,68	1,83	0,15
28	3	14	12	1,40	1,57	0,17
29	3	14	14	1,20	1,33	0,13
30	3	14	15	1,12	1,22	0,10
31	4	16	1,5	12,80	12,74	0,06
32	4	16	2,5	7,68	9,11	1,43
33	4	16	3,5	5,49	5,48	0,01
34	4	16	4	4,80	5,28	0,48
35	4	16	6	3,20	3,28	0,08
36	4	16	8	2,40	2,44	0,04
37	4	16	10	1,92	2,00	0,08
38	4	16	12	1,60	1,76	0,16
39	4	16	14	1,37	1,55	0,18
40	4	16	15	1,28	1,34	0,06
41	5	18	1,5	14,40	14,30	0,10
42	5	18	2,5	8,64	9,11	0,47
43	5	18	3,5	6,17	6,19	0,02
44	5	18	4	5,40	6,14	0,74
45	5	18	6	3,60	3,78	0,18
46	5	18	8	2,70	2,72	0,02
47	5	18	10	2,16	2,20	0,04
48	5	18	12	1,80	1,96	0,16
49	5	18	14	1,54	1,69	0,15
50	5	18	15	1,44	1,48	0,04
51	6	20	1,5	16,00	15,99	0,01
52	6	20	2,5	9,60	10,63	1,03
53	6	20	3,5	6,86	6,89	0,03

54	6	20	4	6,00	6,80	0,80
55	6	20	6	4,00	4,29	0,29
56	6	20	8	3,00	3,02	0,02
57	6	20	10	2,40	2,44	0,04
58	6	20	12	2,00	2,00	0,00
59	6	20	14	1,71	1,83	0,12
60	6	20	15	1,60	1,60	0,00

Table 4.5 Fuzzy controller's evaluation results

Chapter 5 - **User Interface - Web Application**

This application is an interactive web-based responsive user interface that provides a user-friendly and platform independent monitoring and control software. The application is developed with open source technologies to provide both a modular and cost effective solution. It is designed to inform the user of the current state of every component of the system, the water consumptions both present and past, the ability to create watering schedules for the roof garden etc. This framework recognizes the logged-in user and displays a personalized web-view of the application, e.g. when the tenant of apartment one logs-in the application provides the content for this apartment and current user's info. It also provides a real-time monitoring and interactive control page with a 3D scene of the building infrastructure.

User authentication

To every user has been assigned a username and password in order to access the platform. For the login page there has been implemented a *Form-Based Authentication* concept (Figure 5.1). When a user submits his credentials in the log in page (form), a servlet is evoked to verify the user's identity. It compares the submitted username and password against those saved in the database. If they exist (success) the user is forwarded to dashboard page and a cookie is saved to his computer. This cookie is then used by all pages to provide each user with a personalized content, and secure the application from unauthorized access. If his credentials are not correct (failure) or if the connection with the database is not established, a proper message is displayed (Figure 5.2 and Figure 5.3).

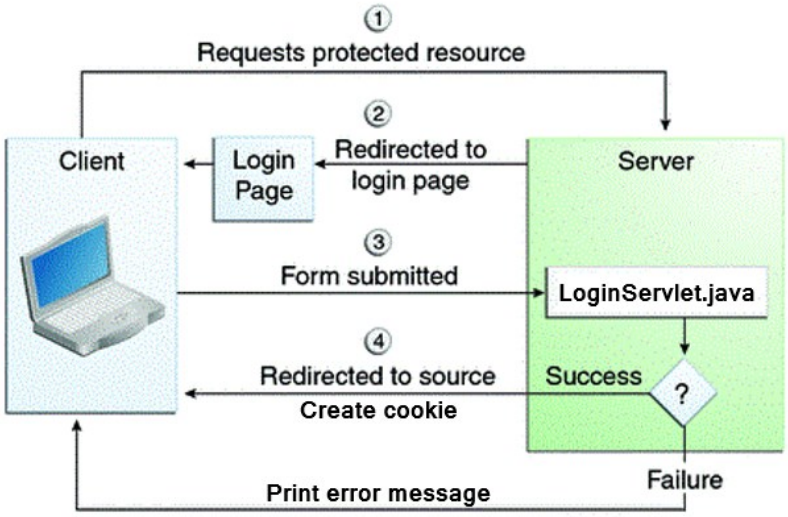


Figure 5.1 Authentication diagram

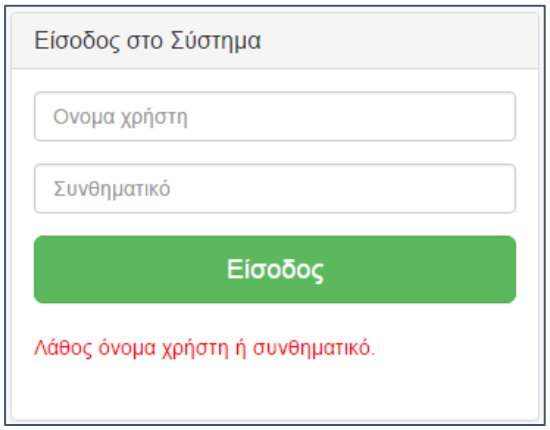


Figure 5.2 Error message - Wrong credentials

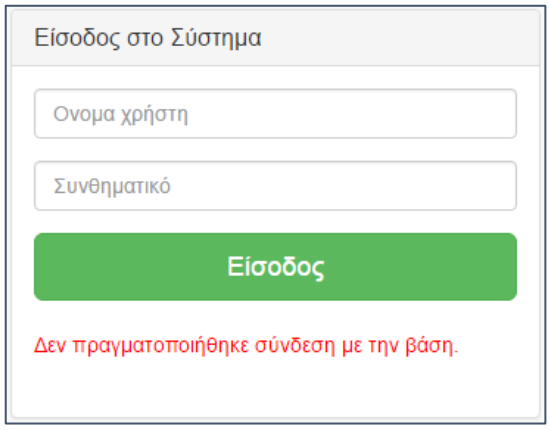


Figure 5.3 Error message - Unable to connect with the database

On every page of this web application there is a code part that checks for authorization before loading its content. This part of code searches for a specific cookie on user's computer. If the cookie exist then the user has logged in and the page concludes the loading process, providing the user with his personalized experience. If the cookie does not exist, i.e. if the user has not logged in, the page redirects the user to login page. This way even if a person knows a page URL he can't access a page without login first.

Roles and profiles

In the developed application there are three user roles: tenant, super (building administrator), and admin (system administrator). Although there are three roles, the content and interface of the tenant's role differs depending on the user's profile. That is because each tenant must have control over his own apartment and not the others. In this concept the tenant's menu has the same number of pages (i.e. tenant's role) but the content on some of this pages adjust based on the apartment that he lives and his information (i.e. his profile).

When a tenant logs in to the application, the menu buttons and by extension the pages that he can access, are a personalized selection of this user's profile. Each tenant can access information about his own flat, and of course control the type of water his apartment uses and its usage profile too. On the other hand, super can access all available information for every apartment and generally most parts of the system. This gives him detailed knowledge over the water consumption of the building complex. Super can also control all apartments and tenants' access credentials, so that he can overwrite water selection if needed or change credentials for a new tenant. Admin has all super's access plus a full control over the roof watering system.

Currently at the roof garden some scientific experiments are being conducted on how different types of water (black, grey, and fresh) affects plats' life cycle. After these experiments are concluded building administrator (super) will inherit these permissions, so that he can change the current watering schedule with another from the provided list or even create a new watering schedule. For this point forward we will assume that both administrators have the same permissions.

In Figure 5.4 and Figure 5.5 we can observe the differences between tenant's menu and administrator's. In this example the tenant lives at the first apartment so on his menu the information and controls are restricted to apartment 01. For each tenant the menu adjusts accordingly.

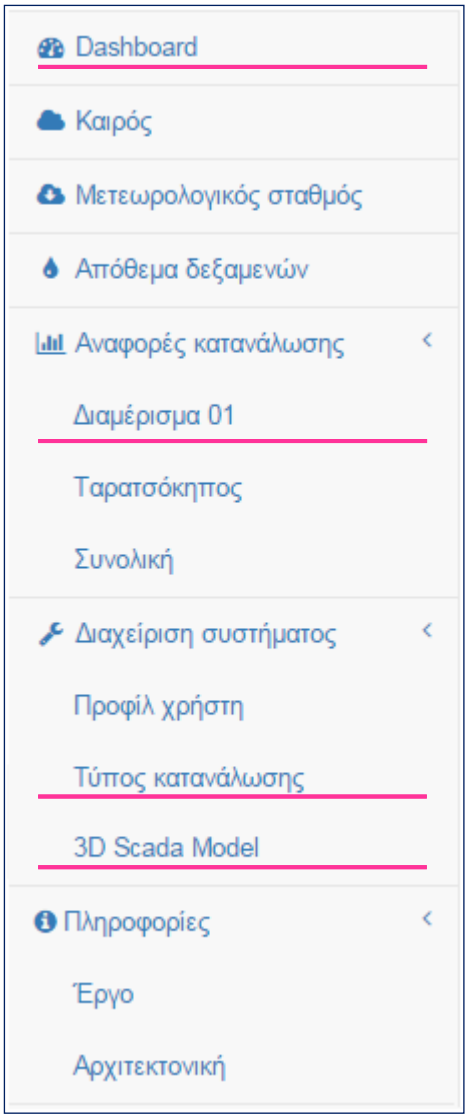


Figure 5.4 Tenant's menu

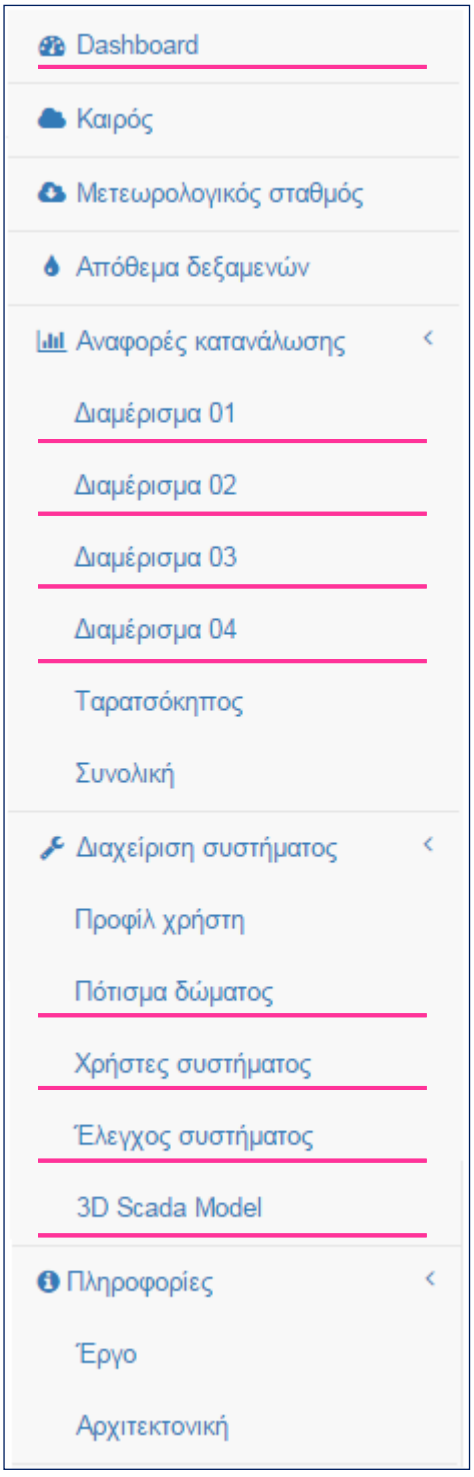


Figure 5.5 Administrator's menu

It’s obvious that administrator’s permissions exceed those of tenant’s. So when administrator selects to cuts off the grey water supply to the building, its only logical that tenant can not enables it, not even to his own apartment. On this concept when the administrator disables the supply to the building on the tenant’s control pages the selection menu its disabled too, and a message is displayed to contact his building super for further information.

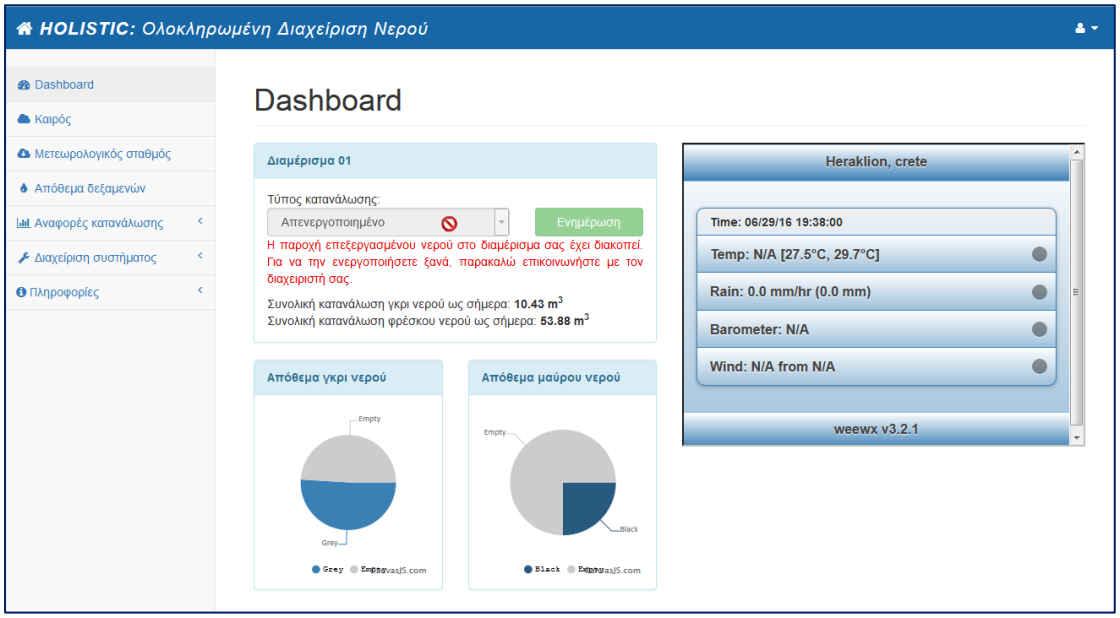


Figure 5.6 Disabled tenant's control - Dashboard

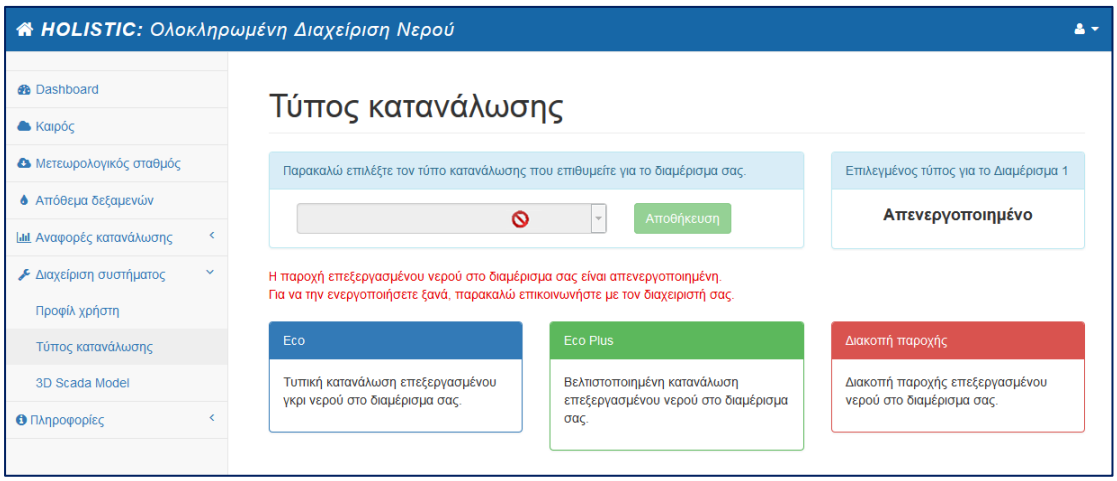


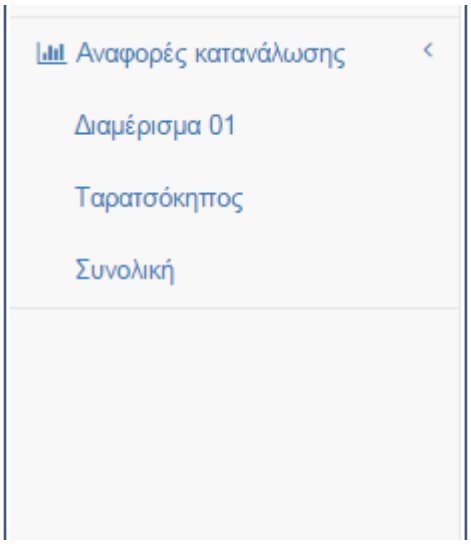
Figure 5.7 Disabled tenant's control - Usage type

Consumption Reports

It is important for every water management system to provide its users with the ability to see and compare the consumptions of a water reuse system. In every line of water supply, for both fresh and recycled water (grey and black), flowmeters have been installed and all water flows are measured and saved in a database. The application provides a series of reports highlighting the consumption of water (fresh, grey, black) for the entire residential complex and for each part separately.

In administrator's menu there are reports of consumption for every apartment of the building complex, the roof garden and of course a report of the total sum. Opposed to administrator's menu, in each tenant's menu only three reports are displayed. The report of his apartment, the roof garden and the total consumption of the complex. Figure 5.8 displays the different menus with the report pages provided for a tenant and an administrator.

Tenant's Menu



Administrator's Menu

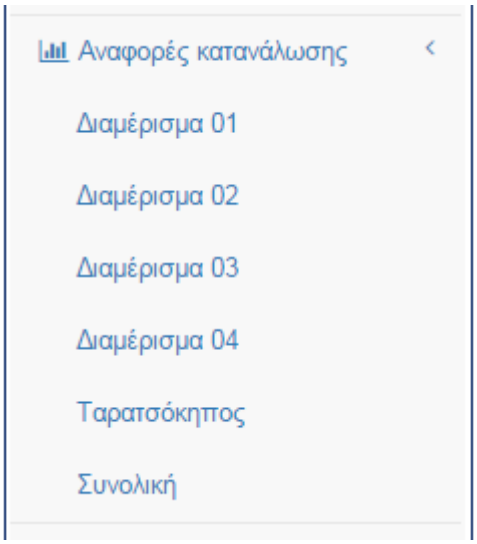


Figure 5.8 Consumption report pages

In every report page there are four charts of water consumption, covering a specific date. The default date is the current. Every time a user opens a report page the application retrieves the current date-time, runs a query to the database to get the desired values and then displays them to the charts accordingly.

The application provides the user with the ability to select a date that he prefers, from a specific panel on the report page (Figure 5.9). When the selected date is submitted, a servlet is evoked, to retrieve the data from the DB and present them to the proper charts. The first

chart shows the daily consumption in hours (Figure 5.10), the second the weekly consumption (Figure 5.11), the third the monthly (Figure 5.12) and the last the yearly (Figure 5.13).

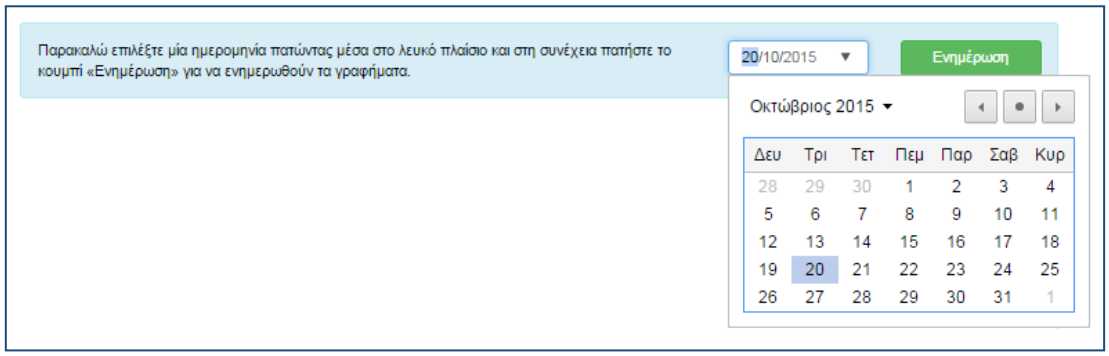


Figure 5.9 Consumption report - Date selection

In every chart referring to an apartment two lines are displayed on it. A grey line for the grey water consumption, and a blue line for the fresh water. In garden and total charts a black lines is displayed too. This line represents the consumption of recycled black water of the garden watering system. At the bottom of each of these charts there is a sum of the individual values of every line (different water consumption). For example under the weekly chart of apartment one, there would be two values in cubic meters. The first value would be the total amount of fresh water that this apartment consumed in this week, and the second the total amount of grey water.

The information displayed to these report pages offers the users a detailed picture of the water consumption of in all parts of the system. This provides them with a better understanding of the gain that this particular water management system offers.

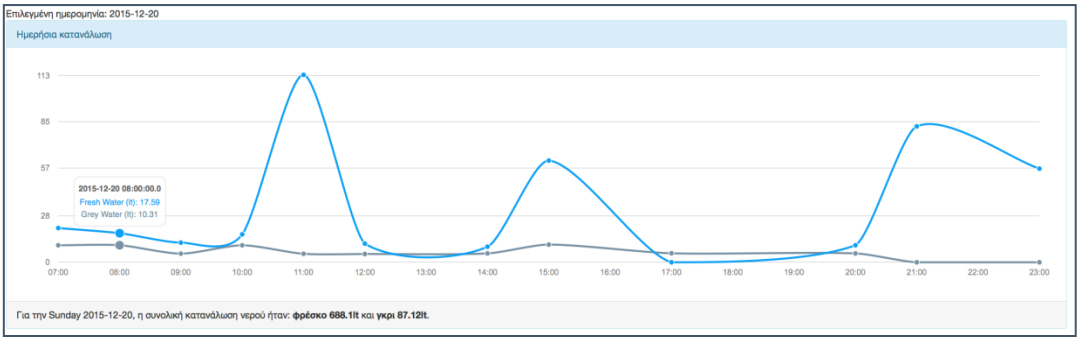


Figure 5.10 Consumption report - Daily chart

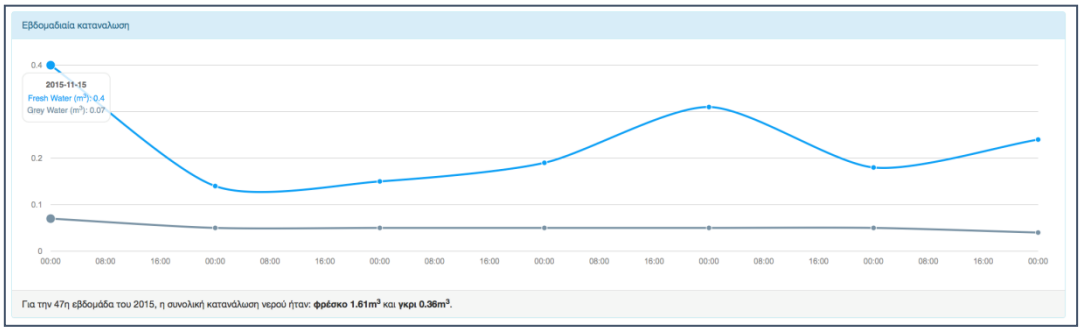


Figure 5.11 Consumption report - Weekly chart

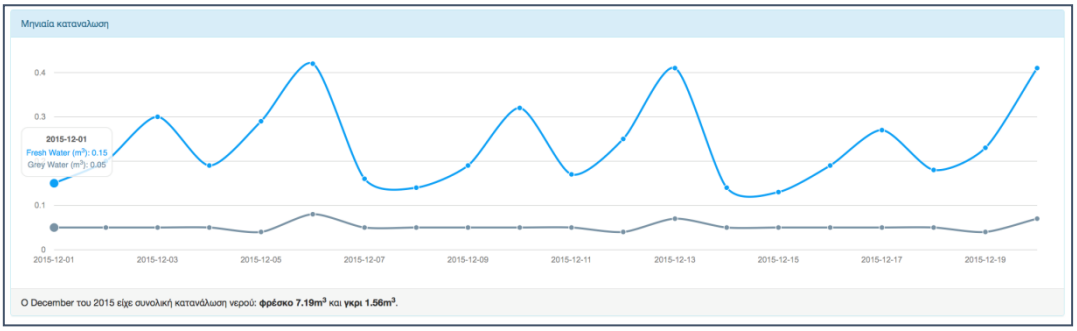


Figure 5.12 Consumption report - Monthly chart

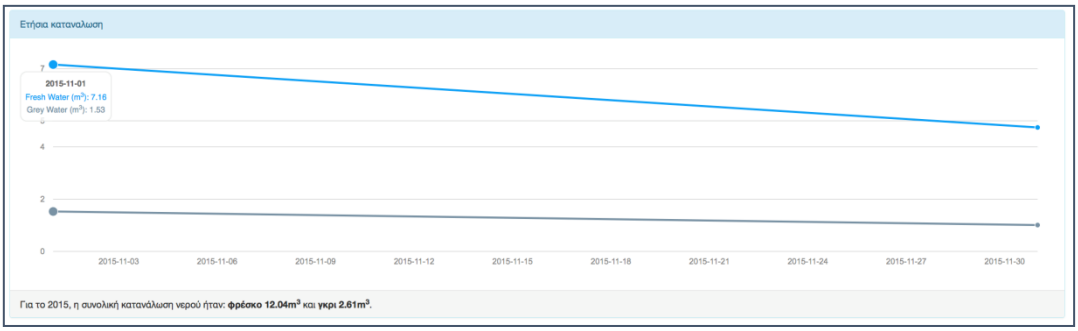


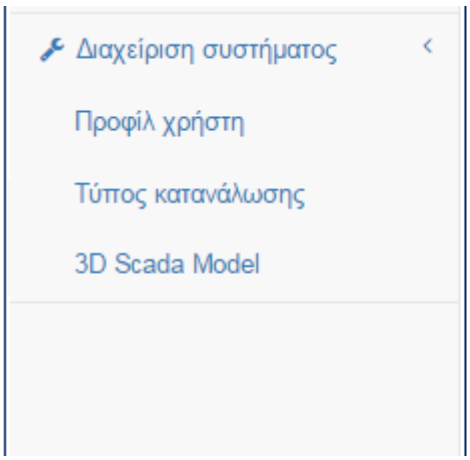
Figure 5.13 Consumption report - Yearly chart

System Control

This application provides building tenants with the ability to control and monitor the MCS. Browsing a user through the application' pages he can be informed about the current state of the system actuators and make changes according to their permission status. To control the different water supply in each part of the building there have been installed electro-valves. To handle these actuators a series of pages have been developed. Based on the role and profile of each user a different menu is displayed to his application instance. Figure 5.14 displays the differences of a tenant's menu and an administrator's. Both roles can change their profile information from "Profile page" (Figure 5.16). Administrator can also change all tenants' information from the "Users' Information" page provided to his menu (Figure 5.17).

Tenant's control over the system's actuators and usage, is limited to his apartment, so only two control pages are displayed to his menu, the "Usage type" (Figure 5.21) and "SCADA" page. In contrast, administrator's menu provides him with three control pages. The first handles the "Watering schedules" of the roof garden, and the others the actuators and the usage of the apartments.

Tenant's Menu



Administrator's Menu

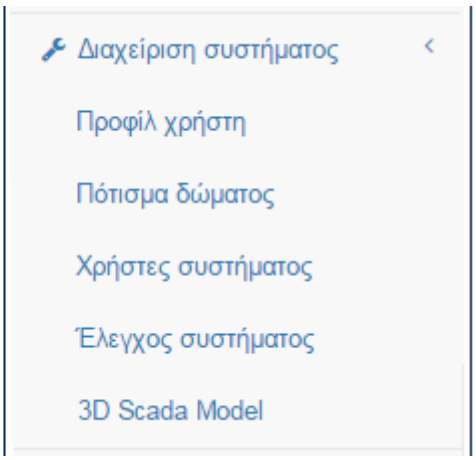


Figure 5.14 Control Menus

Users' information

Every user can change his profile information, i.e. his full name, username and password, from the provided "Profile page" on the application (Figure 5.16). Administrators, as we mentioned earlier, have more permissions than simple users and in this case the ability to overwrite or reset all tenants' information. This can be performed when an existing tenant lives and a new one arrives, or when a tenant forgets his password, or in any other case it is deemed necessary by the administrator. Figure 5.17 displays in a single table[4] all users' information from where administrator can access and change them.

All submitted information is saved to the database (table: Users) through a servlet. This servlet also updates all information on the page according to the new data. From this table the login page retrieves the username and password to authenticate the user and provide him with the personalized experience.

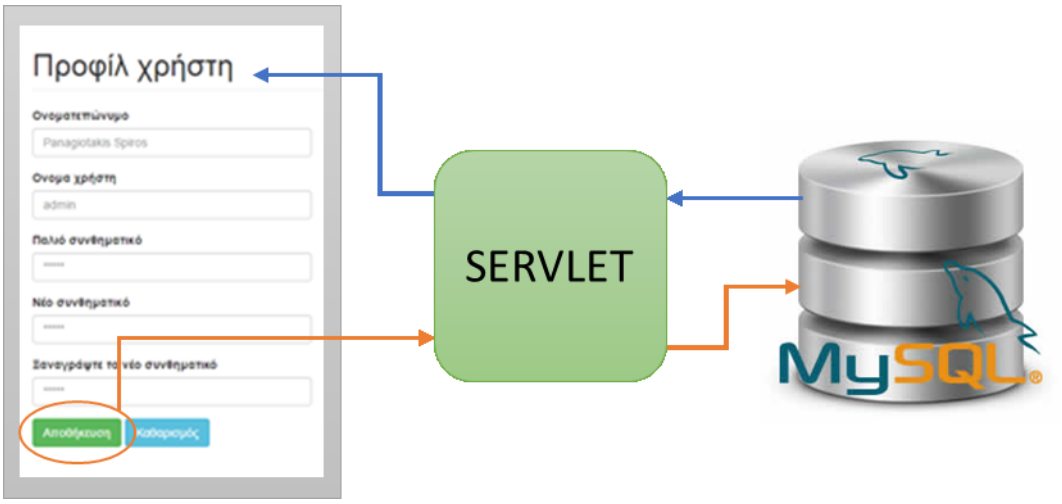


Figure 5.15 User's information - Flow diagram

[4] All users are displayed on this table, except system administrator. Building administrator can't access or overwrite system administrator's information.

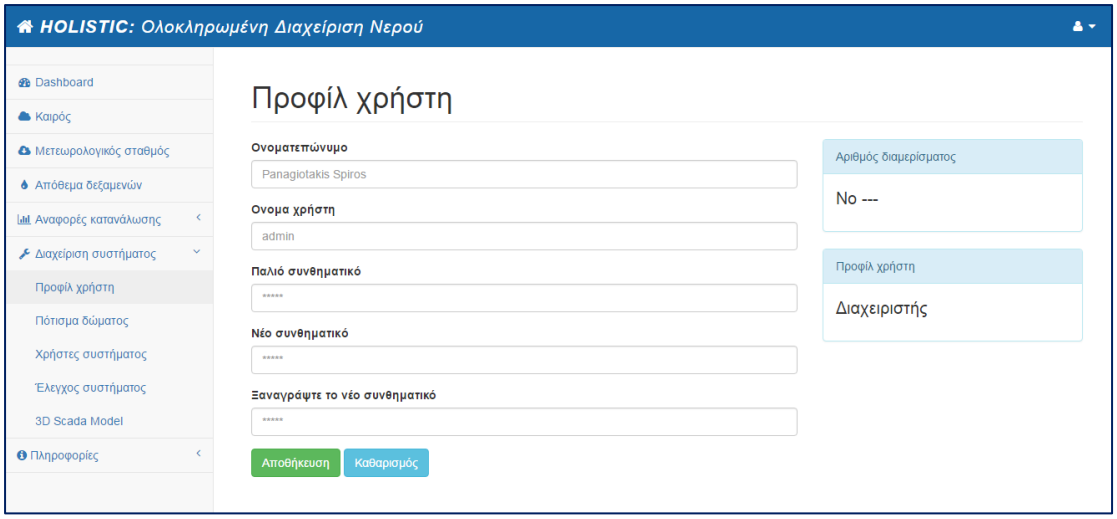


Figure 5.16 Control Menu - Profile page

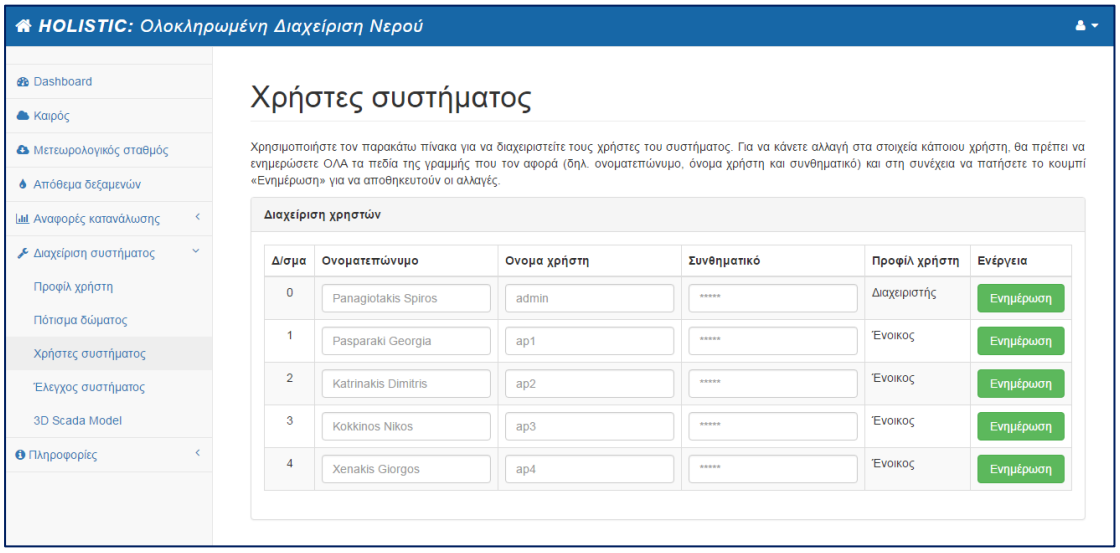


Figure 5.17 Control Menu - Users' information

Watering schedules

On the rooftop of the residential complex a garden has been placed the watering of which uses all three different water types, fresh water, recycled grey and black and is done automatically by the system. Watering page displayed in Figure 5.19 gives administrators the ability to select from a list different schedules for watering the garden. When the administrator wants to change or create new schedule he can visit the page displayed in Figure 5.20. Every schedule consists of a name, a start-time, a duration value in minutes, a frequency value of watering in days and an optional selection of rain delay. The latter is an option that

when the weather station reports rain, it delays the watering depending on the rain value fallen. If a significant portion of water has fallen from the last watering the system will minimize the watering duration or even skip a watering. This action can protect the plants from excessive watering, and reduce the water consumption, especially during the winter.

When a new schedule is selected, a servlet is evoked which calculates the date and time of the next watering and the proper details are saved to the dedicated table in the database. This table always contains the next date and time of the watering event the duration of it, whether or not a rain delay must be applied and the schedule's name. From this table the main control system's algorithm handles the watering and SCADA page shows the watering event. To notify the MCS of the change on watering schedule a message is send to it by the servlet through a web-socket. Figure 5.18 displays the work flow on which the data travel.

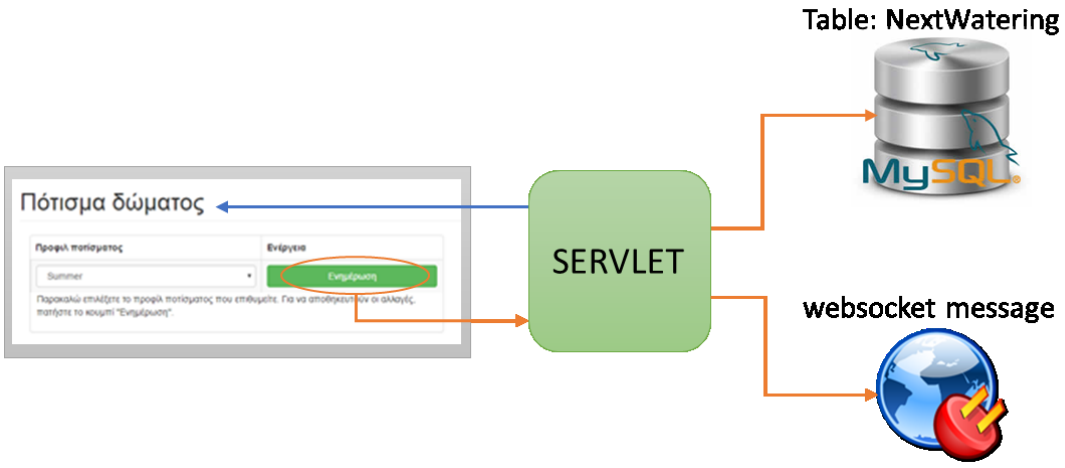


Figure 5.18 Watering schedule selection flow chart

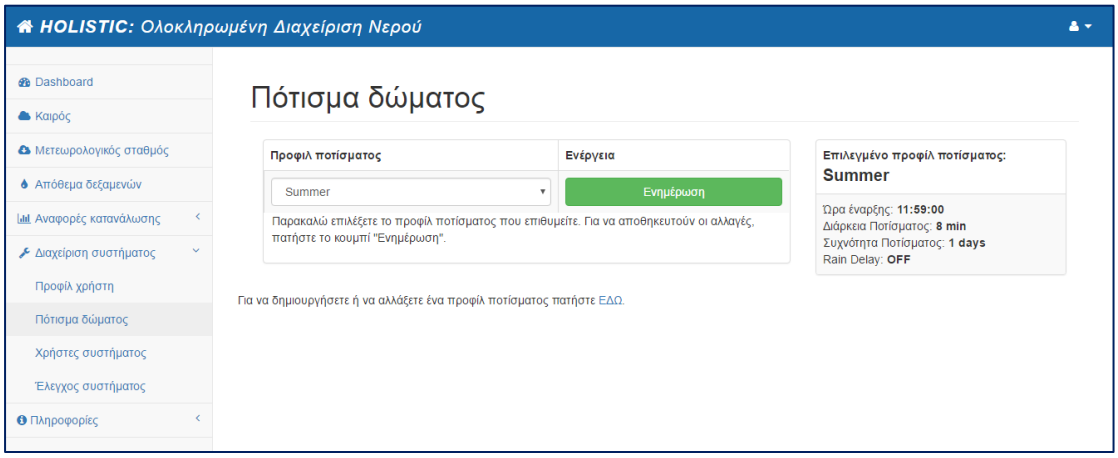


Figure 5.19 Control Menu - Roof garden watering schedule

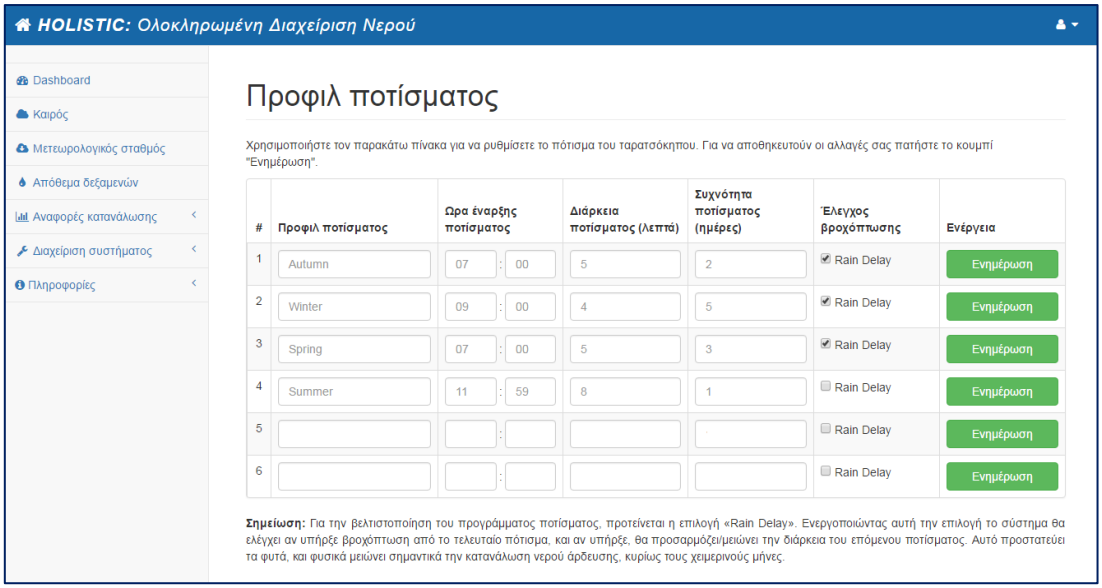


Figure 5.20 Control Menu - Create/Change roof watering schedule

Usage control

Tenants have limited control over the system, in fact a tenant can only control his own apartment (Figure 5.21). He can either activate the supply of recycled grey water to his apartment, selecting also the usage type, or disable it (Figure 5.22). There are two different usage types provided. The first “Eco” simply forwards the grey water to his toilet cistern. The second type “Eco Plus” supplies grey water and performs a delay when reloads it, to save even more water.

The system monitors all flowmeters installed in the building for measuring the consumptions and save them to the database for further use. When “Eco Plus” usage type is selected for an apartment, an extra thread of monitoring is enabled to observe this flowmeter’s values. When the flowmeter receive indication that the toilet cistern is pressed the electrovalve suppling water turns off and a timer starts a countdown. When the time is up, i.e. when the cistern has emptied, the electrovalve opens again to fill the toilet cistern. This prevents extra water to be lost while the cistern empties its content.

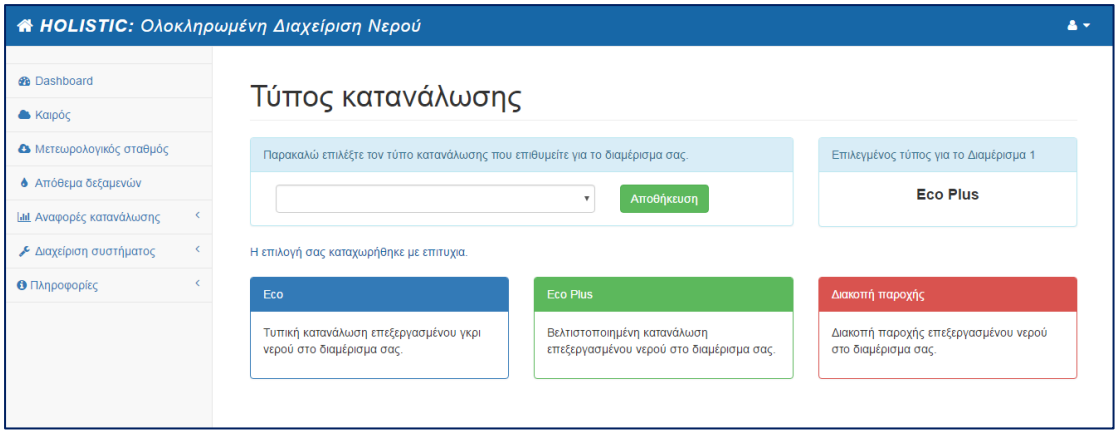


Figure 5.21 Control Menu - Usage type page

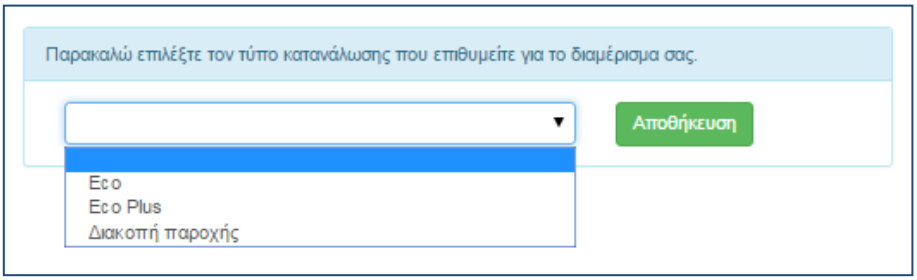


Figure 5.22 Control Menu - Usage type page – Options

On the other hand administrators have the ability to overwrite tenants' selections of their apartments and/or disable some or all parts of the system (Figure 5.23). When an administrator disables the supply of recycled water to the whole building, all actuators' status changes and fresh water supply is automatically enabled. This change can NOT be altered from a tenant; only from an administrator. While this recycled water supplies are turned off, a message suggesting to contact the super is displayed to all tenants control pages when they login to the application (Figure 5.6, Figure 5.7).

All actuators' status and selected usage profiles are saved to a specific table in the database. This table is updated every time a tenant or an administrator makes a change on them. At the same time a message describing the change it is sent through a web-socket to the MCS for notification.

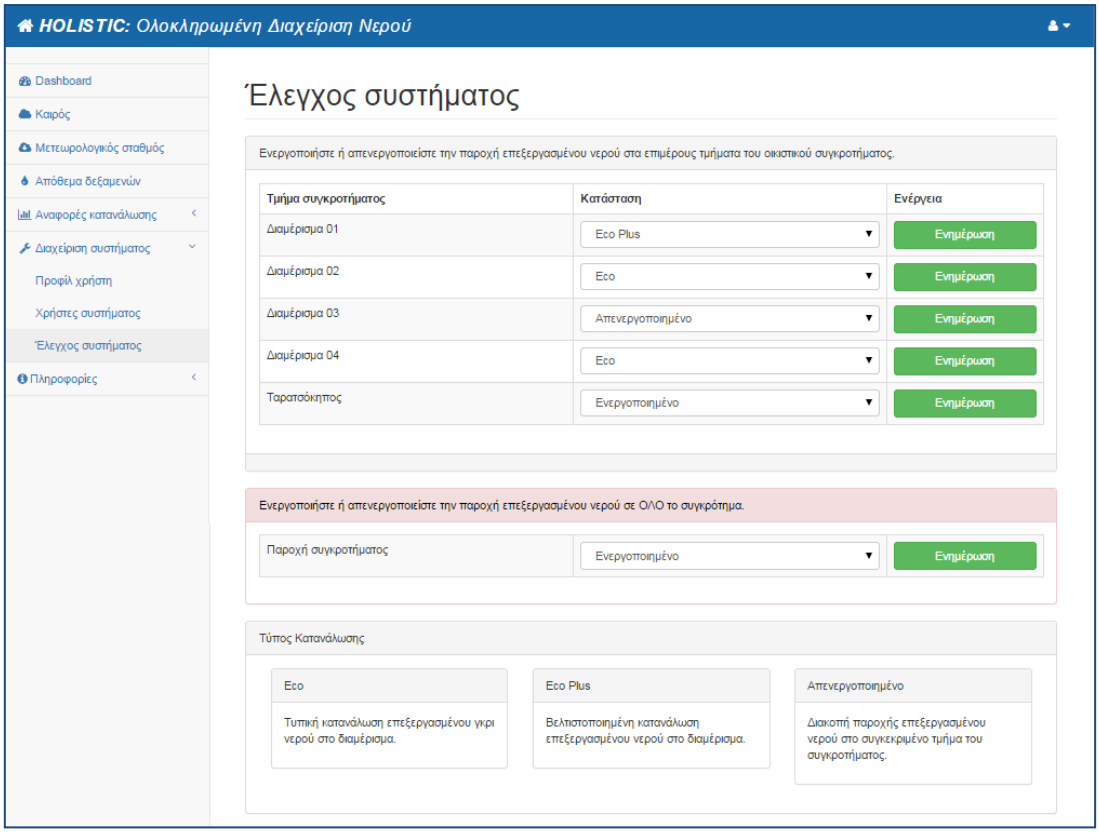


Figure 5.23 Control - Whole System

3D scene

The appearance of interactive 3D scenes on the Web was widely spread mostly after the development of WebGL [47]. WebGL enables the display of such scenes without the need of special hardware or software configurations at the end user's device. Such 3D scenes can thus be displayed on smartphones, tablets, laptops, and desktop PCs only with the use of conventional browsers [48]. There is no need for plug-ins or other additional software [49], [50]. The JavaScript-based open-source framework X3DOM is based on this technology and enables declarative 3D in HTML5 [51]. Same as the integration of SVG for 2D graphics, X3DOM extends the HTML Document Object Model (DOM) with additional (X)3D objects. Using these declarative DOM objects, a web developer can define both a complete 3D scene and its runtime behavior, without low-level JavaScript or WebGL programming [48].

To visualize all the components installed at the building, a 3D scene of the facility was created (Figure 5.24 through Figure 5.28). The displayed components are represented by scene nodes. Some of these nodes may be dynamically attached to mouse-event listeners. This restricts user's control access to specific components, i.e. when a tenant signs in, he can

only control his apartment. The main control unit reports events to indicate changes or process initiations (i.e. garden watering or roof harvesting); it also provides water flow values. Processes such as garden watering and roof harvesting, but also user's changes on electrovalves' states are animated actions, e.g. the change on an electrovalve state will show a rotating wheel.

The creation of a 3D scene for monitoring and control purposes requires the following information: 3D models and metadata of the components, as well as position and orientation vectors [28]. By developing a 3D scene with Autodesk 3ds Max Design, the aforementioned requirements are met. 3ds Max Design is software developed by Autodesk Inc. and provides a 3D modeling, animation, and rendering solution to architects, designers, engineers and visualization specialists. 3ds Max is a powerful tool for creating realistic graphics [52]. It also allows designers to install plug-ins to extend its abilities. Exploiting this ability, the InstantExport plug-in was installed at 3ds Max Design. This provided the capability to export the designed scene as an xhtml file, comprising with all the necessary html code and the x3d scene. This plug-in was designed by InstantReality and it is an x3d exporter for 3ds Max. It exports XML-based x3d models as well as directly HTML/XHTML files [53]. These models can be easily included to any web based application and the objects displayed in it can be modified in the same manner as any DOM object [54].

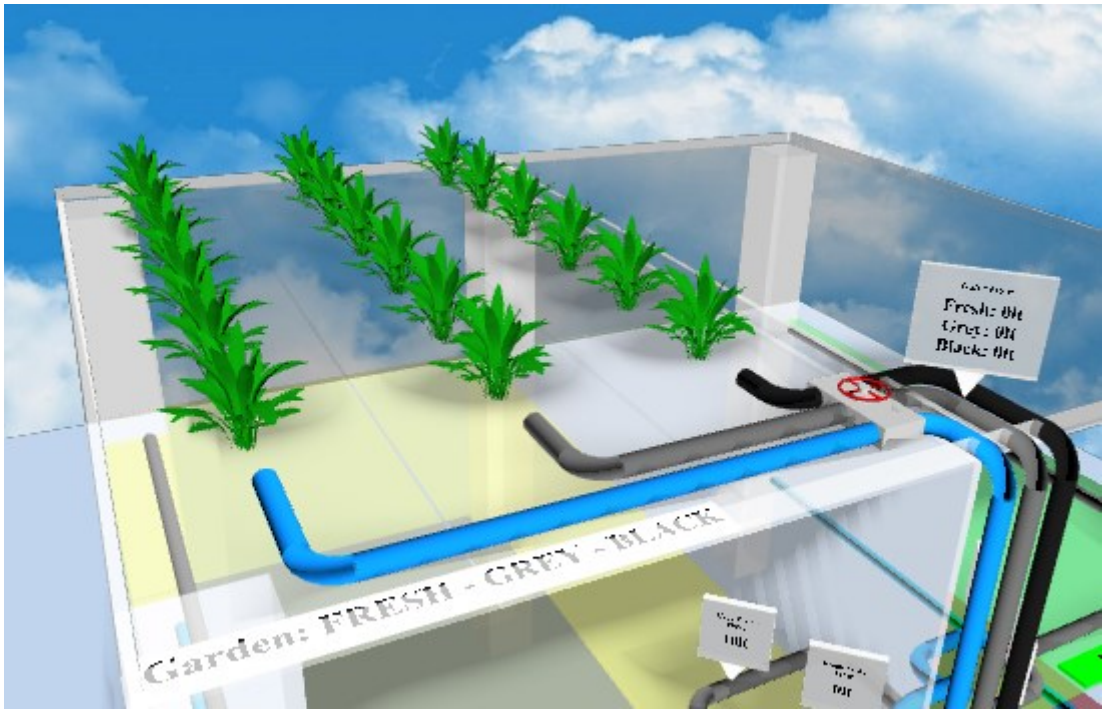


Figure 5.24 3D Scene – Roof garden

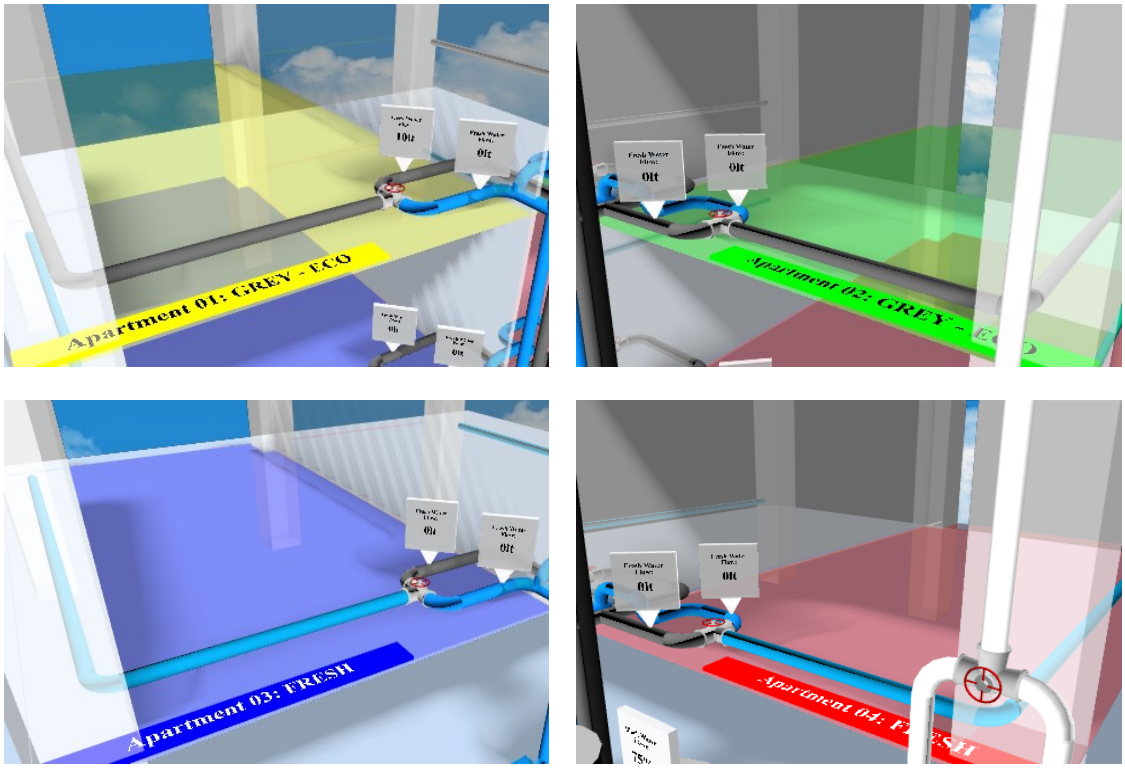


Figure 5.25 3D Scene – Four apartments

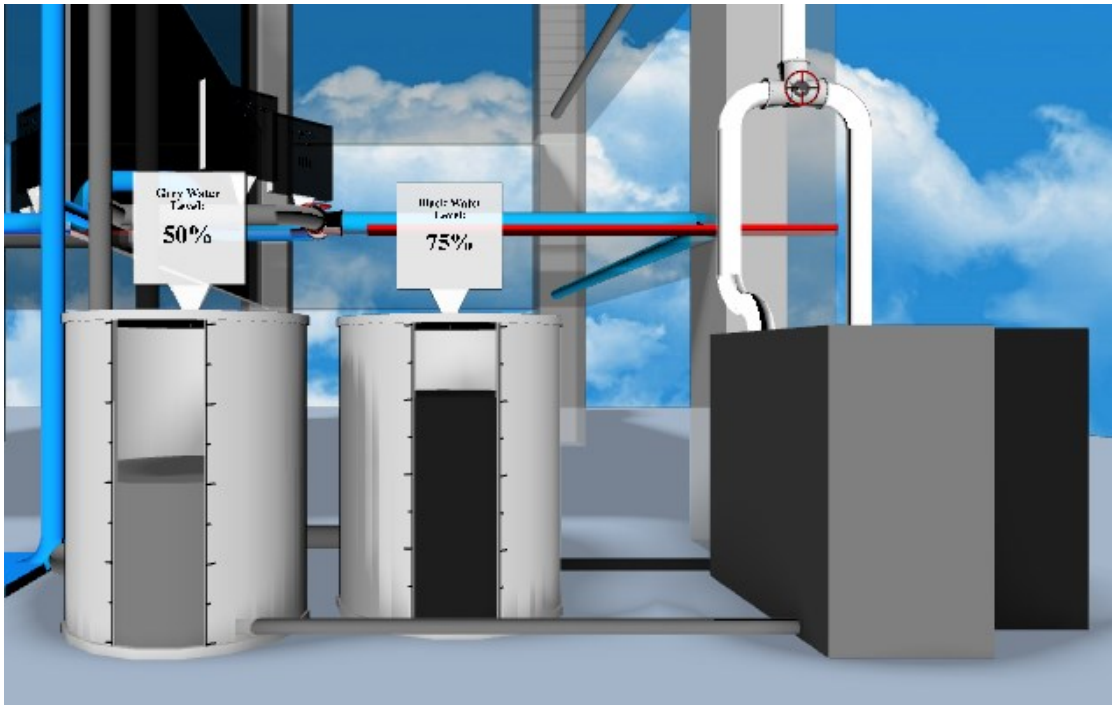


Figure 5.26 3D Scene – Tanks with the treated grey and black water

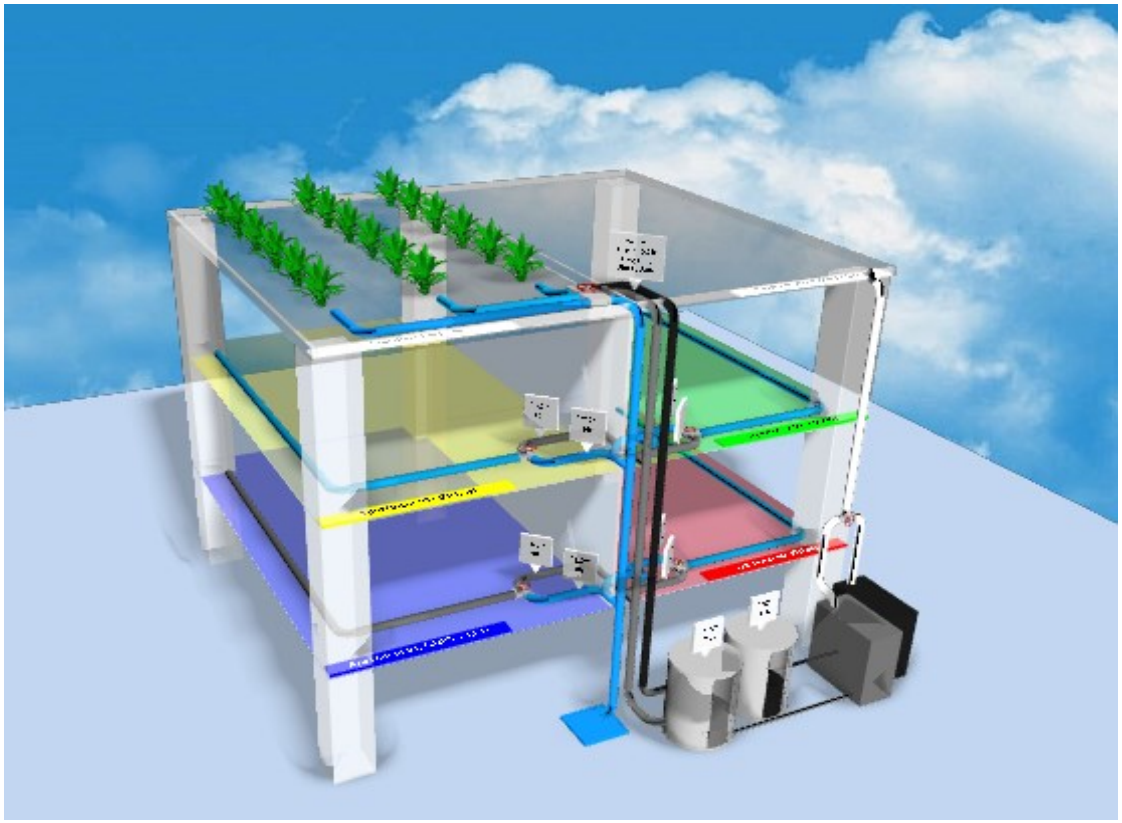


Figure 5.27 3D Scene of the facility – Southeast corner

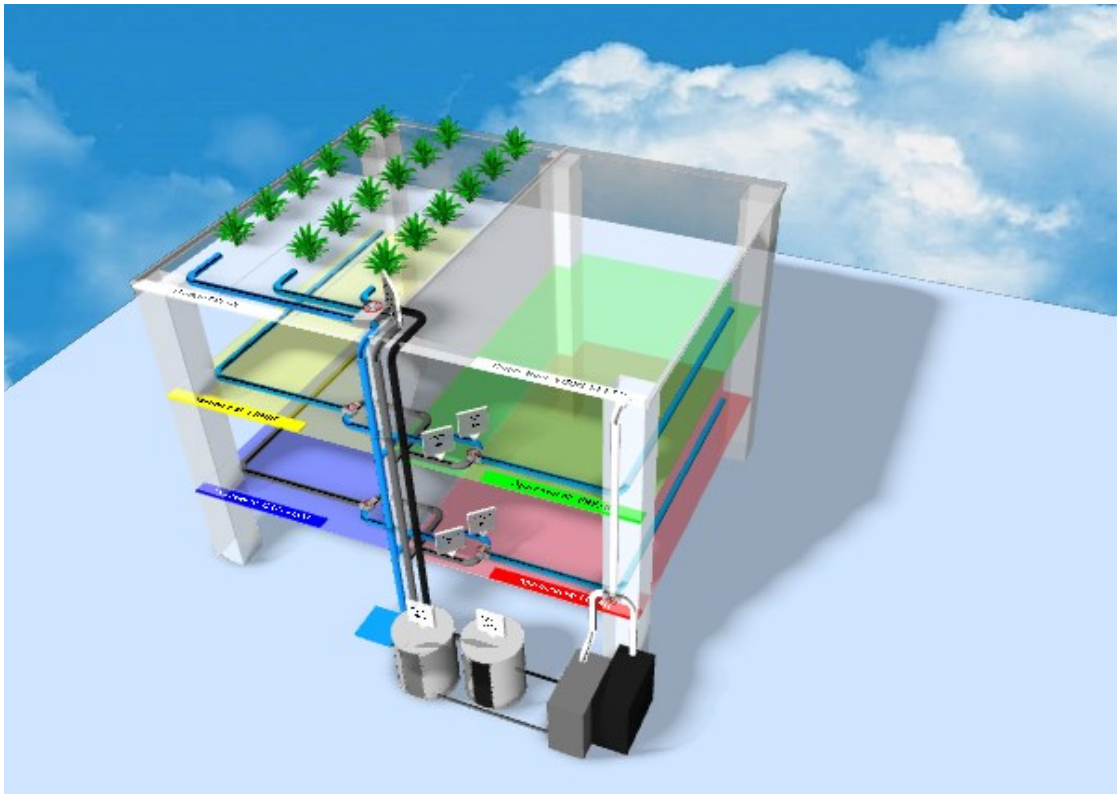


Figure 5.28 3D Scene of the facility – Southwest corner

Real time data acquisition

In order to keep the user updated at all times with the changes occurring at every part of the building infrastructure, a SCADA page was created with an X3D scene. In order to provide a UI with continually updated information, the first time this page loads, it initiates a time interval (T.I.) executing four functions every three seconds. Every one of these functions when called executes a series of tasks in order to acquire and update with the latest values the corresponding areas of the SCADA page (if changes occurred).

```
window.setInterval(mainCallback, 1000*3);  //interval = 3sec

function mainCallback()
{
    CheckForMainChanges();
    CheckForWateringChanges();
    CheckForFirstFlushChanges();
    CheckForFlowmeterChanges();
}
```

Figure 5.29 Continually running functions

Every one of those four functions follows the same principals. A javascript sends an HTTP request through Ajax engine to execute a java servlet. This servlet makes a connection with the database and with a set of SQL queries retrieves the data. The servlet then reforms the data, to a structural form of a JSON object and sends it as a response. The returned JSON information is processed in the javascript and forwarded to the page, updating the corresponding areas. This work flow is displayed in Figure 5.30. In some cases a graphical animation is also displayed throw javascript functions.

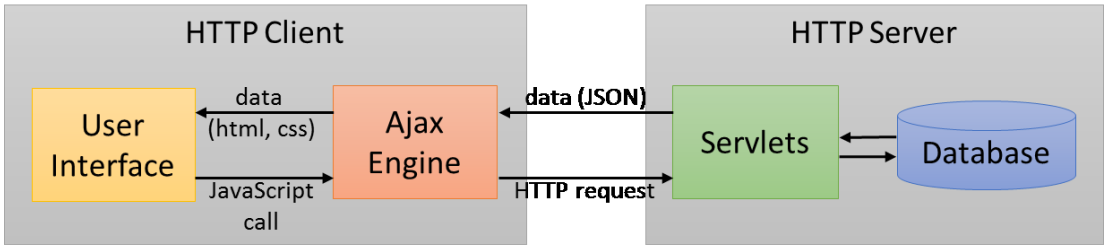


Figure 5.30 Real-time work flow

When the SCADA page is loaded for the first time a servlet called "GetScadaValues.java" is executed to return the actuators' states, the apartments' usage profiles and the tanks' levels values in order to initialize the 3D scene. These values are returned with the form of a JSON object. This object has the form displayed in Figure 5.31 and is assigned to a global variable, so that it can be later compared and determine if a change has occurred.


```
JSON{
    "actuators":[   {"profile":"3","name":"ap1","status":"0"},
                    {"profile":"3","name":"ap2","status":"0"},
                    {"profile":"3","name":"ap3","status":"0"},
                    {"profile":"3","name":"ap4","status":"0"},
                    {"profile":"0","name":"mainblack","status":"0"},
                    {"profile":"0","name":"maingrey","status":"0"},
                    {"profile":"0","name":"roof","status":"0"},
                    {"profile":"0","name":"gardenF","status":"1"},
                    {"profile":"2","name":"gardenB","status":"1"},
                    {"profile":"2","name":"gardenG","status":"1"}
            ],
    "tanks":[       {"level":"0.25","name":"black"},
                    {"level":"0.25","name":"grey"}
            ]
    }
```


Figure 5.31 JSON response from GetScadaValues.java

```
function initialize()
{
    $.get('GetScadaValues', function(res)
    {
        objJSON = JSON.parse(res);
        var i = 0;
        while(i<4)
        {
            updateTexts(i+1, objJSON.actuators[i].status);
            i++;
        }

        if(objJSON.actuators[9].profile == 1 && objJSON.actuators[8].profile == 1)
            updateTexts(0,1);
        else
            updateTexts(0,2);

        var element = document.getElementById("TankLiquidBlack");
        changeSize(objJSON.tanks[0].level, element);
        var tankText = document.getElementById("TankTextEditBlack");
        tankText.setAttribute("string", (objJSON.tanks[0].level*100)+"%");

        element = document.getElementById("TankLiquidGrey");
        changeSize(objJSON.tanks[1].level, element);
        tankText = document.getElementById("TankTextEditGrey");
        tankText.setAttribute("string", (objJSON.tanks[1].level*100)+"%");

        firstflush("white");
        zeroFlowmeters();

    });
}
```

Figure 5.32 SCADA 3d scene initialization function

The first function in the T.I. executes the same java servlet used to initialize the SCADA scene. The returned data is now compared with the previous (saved at the global variable); if there is a change, the new data are forwarded to update the corresponding areas of the scene and this JSON is saved as previous to the global variable.

```
function CheckForMainChanges()
{
  $.get('GetScadaValues', function(resNew)
    {
        if(resNew !== resPrev)
        {
            initialize();
            resPrev = resNew;
        }
    });
}
```

Figure 5.33 Time interval's 1st function

The second function in the time interval, checks if a watering event is enabled and running. The function calls via Ajax a servlet that returns a json object with two values, the next scheduled watering date-time and its duration.

```
function InitializeWatering()
{
    $.get('GetWateringValues', function(wres)
    {
        var objJSON = JSON.parse(wres);
        startTime = new Date(objJSON.nextDatetime);
        var duration = objJSON.duration;
        endTime = new Date(objJSON.nextDatetime);
        endTime.setMinutes(endTime.getMinutes() + duration);
    });
}
```

Figure 5.34 SCADA 3d scene watering function initialization

Then a javascript calculates if at the current date-time (now) the garden is being watered, i.e. if the current date-time is inside the interval of the watering; if now is greater than next-datetime value and less than next-datetime plus the duration of the watering.

$$nextDatetime < now < nextDatetime + duration$$

If this condition is true, the function `potisma()` is called to handle the animation of the watering in the 3D scene.

```
function CheckForWateringChanges()
{
    InitializeWatering();

    if(oldStartTime !=  startTime)          ///why after the 1st time??
    {
        var now = new Date();
        if(now > startTime && now < endTime && potismaStarted===false)
        {
            potismaStarted = true;
            updateJSONobject();
        }

        if(potismaStarted === true && now > endTime)
        {
            potismaStarted = false;
            updateFlowmeters(0, 0, 0, 0);
        }
        oldStartTime = startTime;
    }

    potisma(potismaStarted);
}
```

Figure 5.35 Time interval's 2nd function

The third function executing in the T.I., checks if a first flush event is in progress. The "GetFirstFlushValues.java" servlet is requested to return information of the database "FirstFlush" table. Specifically, this servlet reads the last row of the DB table, i.e. the latest rain event, and returns two values, the "valveToGrey" and the "rainStop" value. Figure 5.36 displays an instance of the "FirstFlush" DB table.

id	rainStarted	rain	xhrasia	fftime	valveToGrey	rainStop	valveToBlack
1532	2016-02-11 05:00:00	4.5	16	5	2016-02-11 05:05:01	2016-02-11 05:25:00	2016-02-11 05:25:01
1533	2016-02-11 14:39:00	1.5	0	0	2016-02-11 14:39:01	2016-02-11 14:58:00	2016-02-11 14:18:01
1534	2016-02-13 18:13:10	3.7	2	3	2016-02-13 18:16:10	2016-02-13 19:33:10	2016-02-13 19:33:10
1535	2016-02-28 07:43:56	2.7	13	6	NULL	2016-02-28 07:47:41	2016-02-28 07:47:42
1536	2016-02-28 19:12:09	2.3	14	6	NULL	2016-02-28 19:14:17	2016-02-28 19:14:17
1537	2016-03-04 09:58:01	6.8	18	2	2016-03-04 10:00:01	2016-03-04 10:23:16	2016-03-04 10:23:16
1538	2016-03-06 22:29:45	3.1	2	3	2016-03-06 22:32:45	2016-03-06 22:47:12	2016-03-06 22:47:12
1539	2016-03-11 06:19:32	2.1	5	4	NULL	2016-03-11 06:21:41	2016-03-11 06:21:42

Figure 5.36 Database table for rainwater harvesting system

Every row on this table encompasses all the stages, from the start until the end of a precipitation event. As Figure 5.36 shows the second column has the timestamp of the rain-start event, columns three through five have the inputs and the output value of the Fuzzy Controller, column six has the timestamp of the FirstFlush-valve when the direction of the water runoff turns towards the grey tank, i.e. when the first flush event ends. Columns seven and eight have the timestamps of the rain-stop event and the state change of the FirstFlush-valve towards the black tank, respectively. When the rain stops before the first flush event ends then the sixth column's cell is null (see lines 1535 and 1536 in Figure 5.36).

After the two values are returned from the servlet, a javascript performs checks to ascertain if a rain event is in progress and at what stage. There are four possible combinations:

- IF both "valveToGrey" and "rainStop" have values then the event is completed, thus we wait for a new event.
- IF both "valveToGrey" is empty and "rainStop" has value then the event is completed and we wait for a new event.
- IF both "valveToGrey" and "rainStop" don't have values then the event has started and the rain runoff is forwarded towards the black tank.
- IF "valveToGrey" has a value but "rainStop" is empty then the event has started and the rain runoff is forwarded towards the grey tank.

Figure 5.38 shows the javascript conditions. The result is a specific string message which is set as an argument in the "firstFlush()" function which displays a graphical representation in the 3D scene.

```
function CheckForFirstFlushChanges()
{
    InitializeFurstFlush();

    if(waitFor == 'newEvent')
        firstflush('white'); //arg can be: white(no rain),
    else if(waitFor == 'valveToGrey')          black(first flush water) or
        firstflush('black');
    else                                       grey(remaining water)
        firstflush('grey');
}
```

Figure 5.37 Time interval's 3rd function

```
function InitializeFurstFlush()
{
    $.get('GetFirstFlushValues', function(response)
    {
        var flushJSON = JSON.parse(response);
        var valveToGrey = flushJSON.FFvalues[0].valveToGrey;
        var rainStop = flushJSON.FFvalues[0].rainStop;

        if(valveToGrey !== "EMPTY")
        {
            if(valveToBlack !== "EMPTY")
                waitFor = 'newEvent';
            else
                waitFor = 'rainStop';
        }
        else
        {
            if(rainStop !== "EMPTY")
                waitFor = 'newEvent';
            else
                waitFor = 'valveToGrey';
        }
    });
}
```

Figure 5.38 SCADA 3d scene FirstFlush function initialization

The last function in the time interval, checks the flow-meters' values and displays them to the corresponding areas of the 3D scene. There are eleven flow-meters installed at the building complex returning consumption information for the four apartments and the garden, i.e. all the locations using treated water. Same as the other functions, after an http request a java servlet is executed and a JSON object returns the relevant data. The structure of this JSON object is shown in Figure 5.39 and the code parsing this JSON in Figure 5.41.

```
JSON{
        "flowmeters":[    {"name":"flowFresh","location":"ap1","value":"0"},
                          {"name":"flowGrey","location":"ap1","value":"13"},
                          {"name":"flowFresh","location":"ap2","value":"18"},
                          {"name":"flowGrey","location":"ap2","value":"0"},
                          {"name":"flowFresh","location":"ap3","value":"0"},
                          {"name":"flowGrey","location":"ap3","value":"0"},
                          {"name":"flowFresh","location":"ap4","value":"0"},
                          {"name":"flowGrey","location":"ap4","value":"0"},
                          {"name":"flowFresh","location":"garden","value":"6.2"},
                          {"name":"flowGrey","location":"garden","value":"6"},
                          {"name":"flowBlack","location":"garden","value":"5.8"}
                      ]
        }
```


Figure 5.39 JSON object - Flow-meters data

```
function updateFlowmeters(D, fresh, grey, black)
{
    if(D===0)
    {
        var gardenFresh = document.getElementById("FlowTextEditGardenF");
        gardenFresh.setAttribute("string", "Fresh: "+fresh+"lt");
        var gardenGrey = document.getElementById("FlowTextEditGardenG");
        gardenGrey.setAttribute("string", "Grey: "+grey+"lt");
        var gardenBlack = document.getElementById("FlowTextEditGardenB");
        gardenBlack.setAttribute("string", "Black: "+black+"lt");
    }
    else
    {
        var apGrey = document.getElementById("FlowTextEditD"+D+"G");
        apGrey.setAttribute("string", grey+"lt");
        var apFresh = document.getElementById("FlowTextEditD"+D+"F");
        apFresh.setAttribute("string", fresh+"lt");
    }
}
```

Figure 5.40 Javascript function displays flowmeters' values

```
function CheckForFlowmeterChanges()
{
    $.get('GetFlowmeterValues', function(resFlow)
    {
        var objJSON = JSON.parse(resFlow);
        console.log(resFlow);

        for(var i=0; i<11; i++)
        {
            if(objJSON.flowmeters[i].location == "garden")
            {
                if(objJSON.flowmeters[i].name == "flowFresh")
                    gfresh = objJSON.flowmeters[i].value;
                else if(objJSON.flowmeters[i].name == "flowGrey")
                    ggrey = objJSON.flowmeters[i].value;
                else
                    gblack = objJSON.flowmeters[i].value;
            }
            else
            {
                if(objJSON.flowmeters[i].name == "flowFresh")
                {
                    apfresh = objJSON.flowmeters[i].value;
                    var location = objJSON.flowmeters[i].location;
                    var D = location.substr(location.length - 1);
                    updateFlowmeters(D, apfresh, apgrey, 0);
                }
                else
                {
                    apgrey = objJSON.flowmeters[i].value;
                    var location = objJSON.flowmeters[i].location;
                    var D = location.substr(location.length - 1);
                    updateFlowmeters(D, apfresh, apgrey, 0);
                }
            }
        }
        if(gfresh!=0 || ggrey!=0 || gblack!=0)
        {
            updateFlowmeters(0, gfresh, ggrey, gblack);
        }
    });
}
```

Figure 5.41 Javascript function parsing the flowmeters' JSON

Real time monitoring

In the previous paragraph it was disused how the application retrieves the data from the database; here it will be presented the graphical display of the changes at the 3D scene. For the initialization of the 3D scene, i.e. the display of the selected usage profiles of the apartments and the garden (Figure 5.42), but also the tank levels' values (Figure 5.43), the `initialize()` function is executed. This function includes two sub-functions that display the new data changes to the scene, the `updateTexts()` function (Figure 5.44) that updates all the textual information of the selected usage profiles of each apartment, the garden and also the presentage of the valiable recycled water for reuse. `changeSize()` fuction () handles the graphical representation of the water levels display inside the tanks.

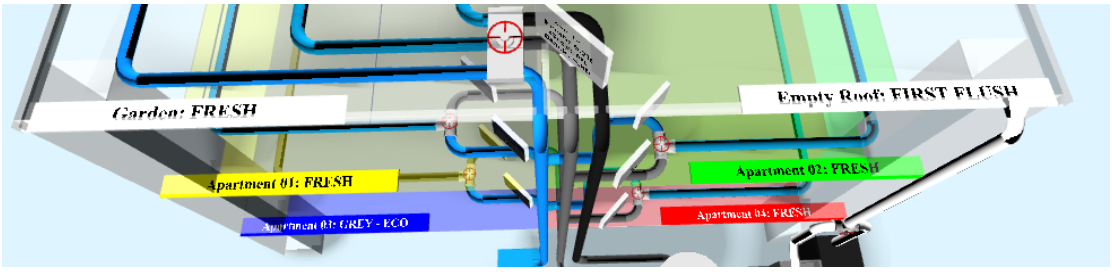


Figure 5.42 Usage profiles

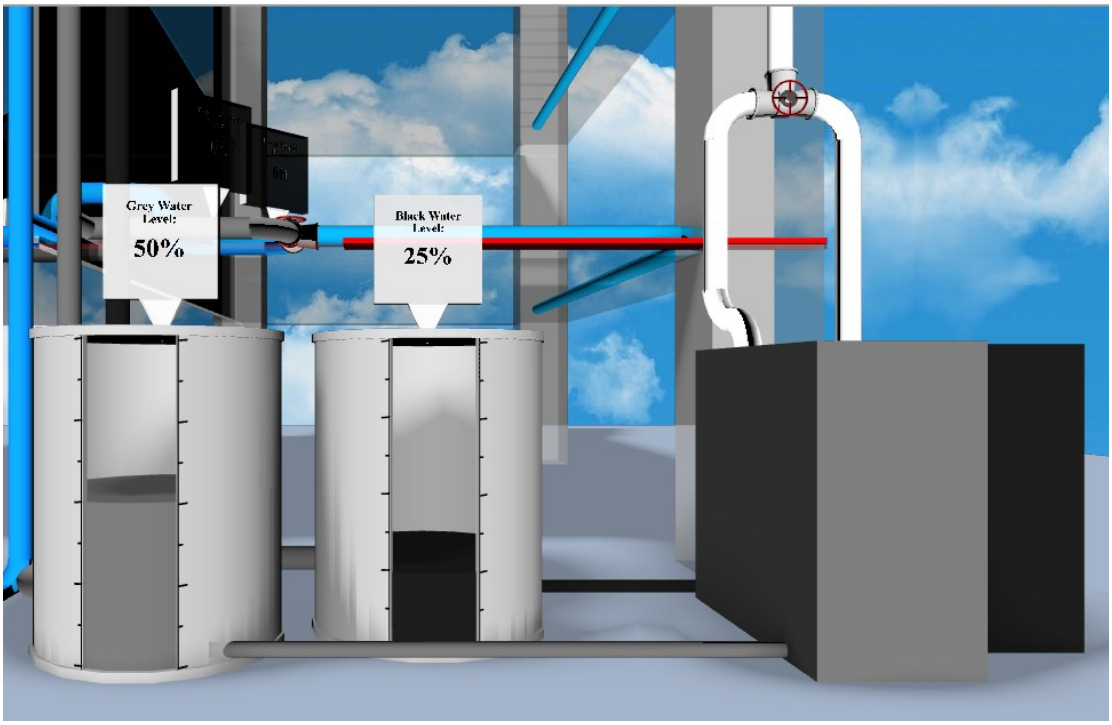


Figure 5.43 Tanks' recycled water levels

The `updateTexts()` function (Figure 5.44) takes two input arguments: the number of the apartment ('0' for the garden) and the valve state. Depending on the data that needs to be updated the x3dom element of the specific text is called to display the textual information. Figure 5.45 displayes the stracure of the textual element of the apartment one.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//update texts and colors | D=number of apartment(0=garden), valState=status for apartment(profile for garden)
////////////////////////////////////////////////////////////////////////////////////////////////////
function updateTexts(D, valState)
{
    if(D==0)
    {
        if(valState == 2)
        {
            document.getElementById("TextEditGarden").setAttribute('string', 'Garden: FRESH');
            var groupG = document.getElementById('TubeGardenGrey').getElementsByTagName('Material');
            for(var i=0; i<groupG.length; i++)
             { groupG[i].setAttribute('diffuseColor','0.031373 0.596078 0.815686');}
            groupG = document.getElementById('TubeGardenBlack').getElementsByTagName('Material');
            for(var i=0; i<groupG.length; i++)
             { groupG[i].setAttribute('diffuseColor','0.031373 0.596078 0.815686');}
        }
        else
        {
            document.getElementById("TextEditGarden").setAttribute('string', 'Garden: FRESH - GREY - BLACK');
            var groupG = document.getElementById('TubeGardenGrey').getElementsByTagName('Material');
            for(var i=0; i<groupG.length; i++)
             { groupG[i].setAttribute('diffuseColor','0.392157 0.392157 0.392157');}
            groupG = document.getElementById('TubeGardenBlack').getElementsByTagName('Material');
            for(var i=0; i<groupG.length; i++)
             { groupG[i].setAttribute('diffuseColor','0.090196 0.090196 0.090196');}
        }
    }
    else
    {
        if(valState == 0)
        {
            document.getElementById('TextEditD'+D).setAttribute('string', 'Apartment 0'+D+': FRESH');
            var groupD = document.getElementById('TubesD'+D).getElementsByTagName('Material');
             for(var i=0; i<groupD.length; i++)
              { groupD[i].setAttribute('diffuseColor','0.031373 0.596078 0.815686');}
        }
        else
        {
            if(objJSON.actuators[D-1].profile == 1)
                document.getElementById("TextEditD"+D).setAttribute('string', 'Apartment 0'+D+': GREY - ECO');
            else
                document.getElementById("TextEditD"+D).setAttribute('string', 'Apartment 0'+D+': GREY - ECO PLUS');

            var groupD = document.getElementById('TubesD'+D).getElementsByTagName('Material');
            for(var i=0; i<groupD.length; i++)
             { groupD[i].setAttribute('diffuseColor','0.392157 0.392157 0.392157');}
        }
     }
};
```

Figure 5.44 Javascript function updating all texts in the 3D scene

```
<Transform id='TextD100' translation='-47.876137 61.688843 56.825386' >
        <Transform id='TextD10' translation='0 0.7 0' rotation='-1 0 0 1.55' >
                <shape>
                        <Text id='TextEditD1' string="Apartment 01: ">
                                <FontStyle style="BOLD" size="2.7"/>
                        </Text>
                        <Appearance>
                                <Material  diffuseColor='0 0 0'/>
                        </Appearance>
                </shape>
        </Transform>
        <Shape>
                <Appearance id='WireColor_25' >
                        <Material diffuseColor='1.000000 1.000000 0.000000' />
                </Appearance>
                <IndexedFaceSet id='TextD1-GEOMETRY' solid='false' coordIndex='0 1 2 -
                        <Coordinate point='-21.500000 0.000000 2.000000, -21.500000 0.
                        <Normal vector='0.000000 -1.000000 0.000000, 0.000000 -1.00000
                </IndexedFaceSet>
        </Shape>
</Transform>
```

Figure 5.45 X3DOM text element

Roof garden watering event

The watering of the roof garden starts according to the scheduled date-time. Although the date-time is saved to the DB it can be changed from the web application at any time, while some other user observes the 3d model. Consequently it is important to check for unpredicted changes to watering schedule. To this end a function (Figure 5.35) checking these changes has been added to the interval, to run every three seconds. To update the graphical representation of the roof garden at the 3D scene after reciving the new information the `potisma()` function (Figure 5.49) is executed.

One input argument is given to the `potisma()` function. This argument “tells” if the watering is on or off (true or false). If the watering is off the displayed result sould be of Figure 5.46. If the watering is on then the displayed preview would be as Figure 5.47 or as Figure 5.48 depending on the usage profile, i.e. if the watering uses all three types of water or just one.

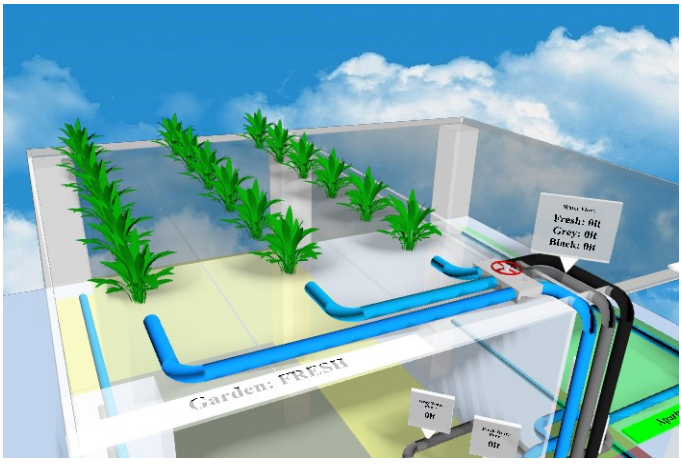


Figure 5.46 Roof garden watering – OFF

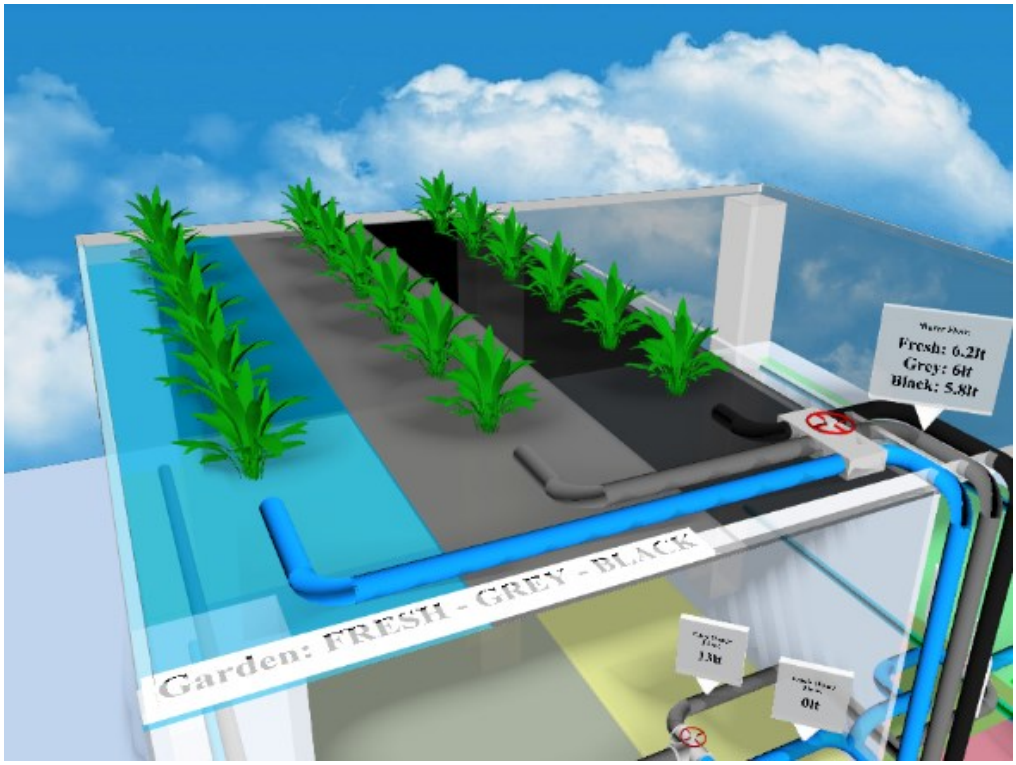


Figure 5.47 Roof garden watering – ON, with Fresh, Grey and Black water

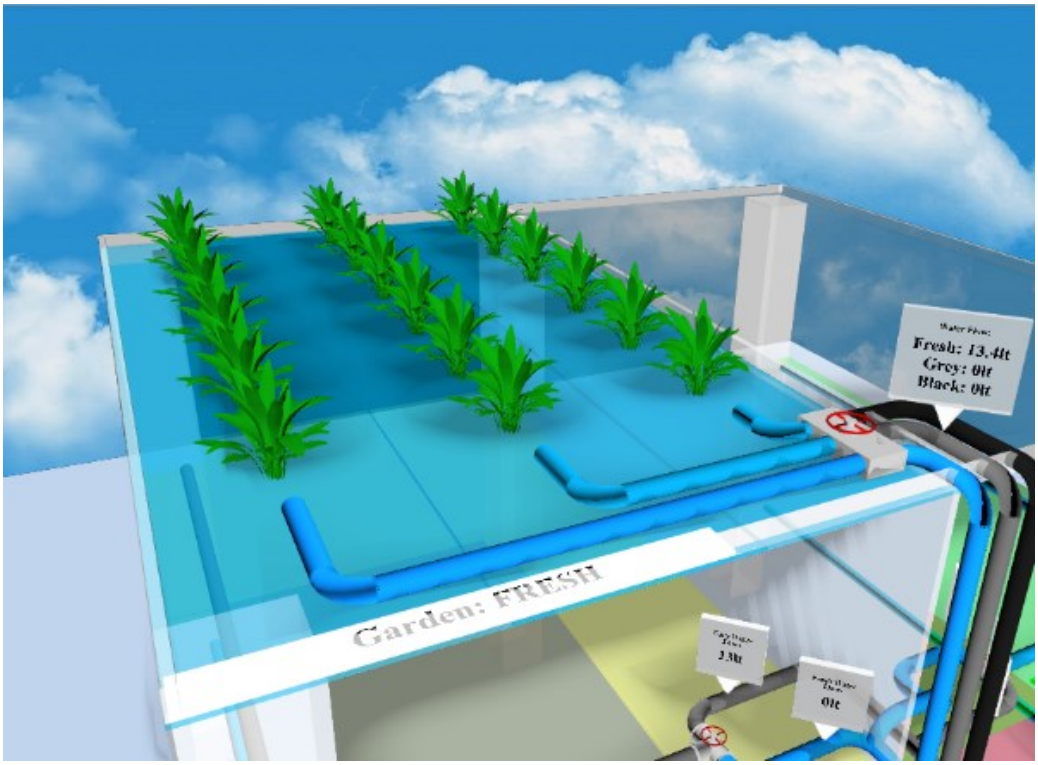


Figure 5.48 Roof garden watering – ON, only with Fresh water

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//Roof Garden Watering preview - set color to garden floors
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
function potisma(onoff)
{
    if(onoff === true)
    {
        if(objJSON.actuators[9].profile == 1 && objJSON.actuators[8].profile == 1)  //if its enabled
        {
            var fresh = document.getElementById('FloorGardenFresh').getElementsByTagName('Material');
            var grey  = document.getElementById('FloorGardenGrey').getElementsByTagName('Material');
            var black = document.getElementById('FloorGardenBlack').getElementsByTagName('Material');
            for(var i=0; i<fresh.length; i++)
                {
                    fresh[i].setAttribute('diffuseColor','0 0.6 0.8');
                    fresh[i].setAttribute('transparency','0.5');
                    grey[i].setAttribute('diffuseColor','0.4 0.4 0.4');
                    grey[i].setAttribute('transparency','0.2');
                    black[i].setAttribute('diffuseColor','0 0 0');
                    black[i].setAttribute('transparency','0.5');
                }
        }
        else
         {
            var fresh = document.getElementById('FloorGardenFresh').getElementsByTagName('Material');
            var grey  = document.getElementById('FloorGardenGrey').getElementsByTagName('Material');
            var black = document.getElementById('FloorGardenBlack').getElementsByTagName('Material');
            for(var i=0; i<fresh.length; i++)
                {
                    fresh[i].setAttribute('diffuseColor','0 0.6 0.8');
                    fresh[i].setAttribute('transparency','0.5');
                    grey[i].setAttribute('diffuseColor','0 0.6 0.8');
                    grey[i].setAttribute('transparency','0.5');
                    black[i].setAttribute('diffuseColor','0 0.6 0.8');
                    black[i].setAttribute('transparency','0.5');
                }
        }
    }
    else
    {
        var fresh = document.getElementById('FloorGardenFresh').getElementsByTagName('Material');
        var grey  = document.getElementById('FloorGardenGrey').getElementsByTagName('Material');
        var black = document.getElementById('FloorGardenBlack').getElementsByTagName('Material');
        for(var i=0; i<fresh.length; i++)
            {
                fresh[i].setAttribute('diffuseColor','1 1 1');
                fresh[i].setAttribute('transparency','0.8');
                grey[i].setAttribute('diffuseColor','1 1 1');
                grey[i].setAttribute('transparency','0.8');
                black[i].setAttribute('diffuseColor','1 1 1');
                black[i].setAttribute('transparency','0.8');
            }
    }
}
```

Figure 5.49 Javascript function updating the watering event

Rainfall harvesting event

The rainfall harvesting process starts when the rain sensor is triggered. When this happens on the 3d model the empty roof floor changes color to black (Figure 5.50). This means that the rain started, forwording the water direction to the black water tank, coloring as black. This was designed to emphasize that the first rain water is treated as black according to the concept of the first flush event. After a specific time (***Rainfall Harvesting Algorithm***) when the roof surface is considered cleaned, it changes color to grey (Figure 5.51) as do the tubes, to display the rection of the rainfall to the grey water tank. This is shown until the rain stops.

All these graphical representations are handled from the `firstflush()` function (Figure 5.52). This function takes as an argument a string with a single word. The possible cases would be “white” if there is no rian, “black” if the rain has started but the first flush event hasn’t concluded yet, and grey when the first flush event has ended but the rain not. Do display all these color changes the fuction gets a specific element by its id , e.g. “FloorRoof”, and changes its diffuseColor attribute.

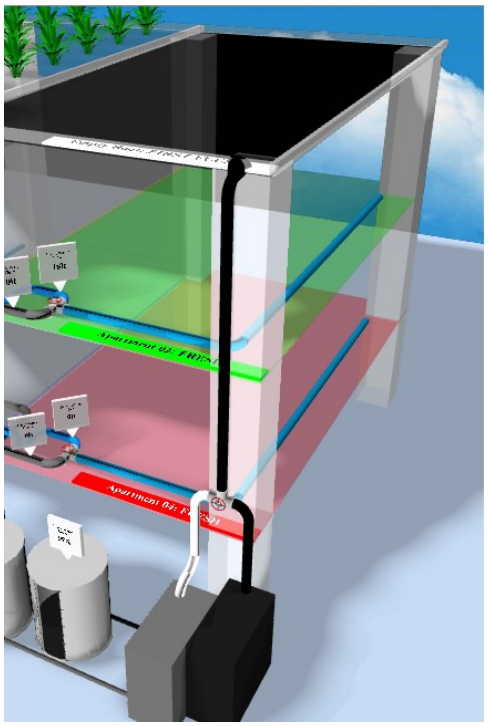


Figure 5.50 Water harvesting as black

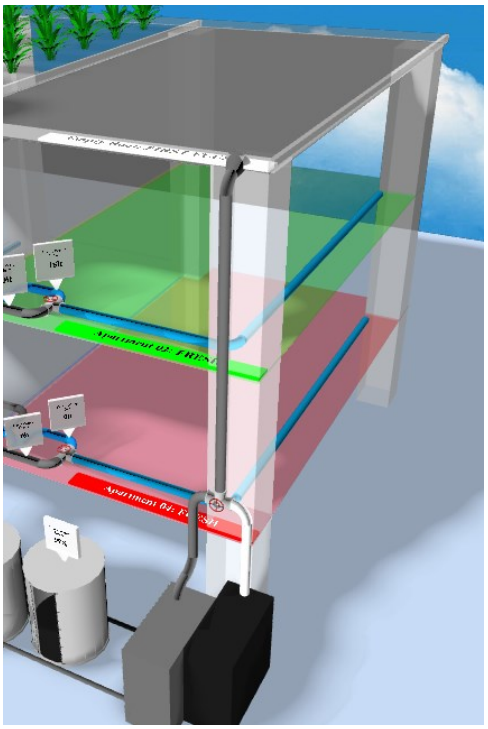


Figure 5.51 Water harvesting as gre

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//First Flush preview - set color to tubes
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
function firstflush(goto) //goto can be white(no rain), black(first flush water) or grey(remaining water)
{
    if(goto === "black")
    {
        //------------------
        var ff = document.getElementById('FloorRoof').getElementsByTagName('Material');
        for(var i=0; i<ff.length; i++)
            {
                ff[i].setAttribute('diffuseColor','0 0 0');
                ff[i].setAttribute('transparency','0.5');
            }

        var fftubes = document.getElementById('TubesRoof').getElementsByTagName('Material');
        for(var i=0; i<fftubes.length; i++)
            { fftubes[i].setAttribute('diffuseColor','0 0 0');}
        //------------------
        var fftubesblack = document.getElementById('TubesRoofBlack').getElementsByTagName('Material');
        for(var i=0; i<fftubesblack.length; i++)
            { fftubesblack[i].setAttribute('diffuseColor','0 0 0');}

        var fftubesgrey = document.getElementById('TubesRoofGrey').getElementsByTagName('Material');
        for(var i=0; i<fftubesgrey.length; i++)
            { fftubesgrey[i].setAttribute('diffuseColor','1 1 1');}
    }
    else if (goto === "grey")
    {
        //------------------
        var ff = document.getElementById('FloorRoof').getElementsByTagName('Material');
        for(var i=0; i<ff.length; i++)
            {
                ff[i].setAttribute('diffuseColor','0.4 0.4 0.4');
                ff[i].setAttribute('transparency','0.2');
            }

        var fftubes = document.getElementById('TubesRoof').getElementsByTagName('Material');
        for(var i=0; i<fftubes.length; i++)
            { fftubes[i].setAttribute('diffuseColor','0.4 0.4 0.4');}
        //------------------
        var fftubesgrey = document.getElementById('TubesRoofGrey').getElementsByTagName('Material');
        for(var i=0; i<fftubesgrey.length; i++)
            { fftubesgrey[i].setAttribute('diffuseColor','0.4 0.4 0.4');}

        var fftubesblack = document.getElementById('TubesRoofBlack').getElementsByTagName('Material');
        for(var i=0; i<fftubesblack.length; i++)
            { fftubesblack[i].setAttribute('diffuseColor','1 1 1');}
    }
    else
    {
        //------------------
        var ff = document.getElementById('FloorRoof').getElementsByTagName('Material');
        for(var i=0; i<ff.length; i++)
            {
                ff[i].setAttribute('diffuseColor','1 1 1');
                ff[i].setAttribute('transparency','0.8');
            }

        var fftubes = document.getElementById('TubesRoof').getElementsByTagName('Material');
        for(var i=0; i<fftubes.length; i++)
            { fftubes[i].setAttribute('diffuseColor','1 1 1');}
        //------------------
        var fftubesblack = document.getElementById('TubesRoofBlack').getElementsByTagName('Material');
        for(var i=0; i<fftubesblack.length; i++)
            { fftubesblack[i].setAttribute('diffuseColor','1 1 1');}

        var fftubesgrey = document.getElementById('TubesRoofGrey').getElementsByTagName('Material');
        for(var i=0; i<fftubesgrey.length; i++)
            { fftubesgrey[i].setAttribute('diffuseColor','1 1 1');}
    }
}
```

Figure 5.52 Javascript fuction updating the rainfall harvesting system

Flowmeters measurements

As a tenant uses the water supply (fresh or filtered) at his apartment the water flow measurements of his consumption are displayed in real-time to the corresponding text area in the 3d scene. As it was mentioned in the previous paragraph every three seconds the system gets the newest mesurments of all the flows of the building, to display these values to the 3D scene the `CheckForFlowmeterChanges()` fuction (Figure 5.41) calls the `updateFlowmeters()` (Figure 5.53). This function as does the `updateTexts()` mentioned before updates the textual information of all the flowmeters values in the 3d scene in real time.

```
function updateFlowmeters(D, fresh, grey, black)
{
    if(D===0)
    {
        var gardenFresh = document.getElementById("FlowTextEditGardenF");
        gardenFresh.setAttribute("string", "Fresh: "+fresh+"lt");
        var gardenGrey = document.getElementById("FlowTextEditGardenG");
        gardenGrey.setAttribute("string", "Grey: "+grey+"lt");
        var gardenBlack = document.getElementById("FlowTextEditGardenB");
        gardenBlack.setAttribute("string", "Black: "+black+"lt");
    }
    else
    {
        var apGrey = document.getElementById("FlowTextEditD"+D+"G");
        apGrey.setAttribute("string", grey+"lt");
        var apFresh = document.getElementById("FlowTextEditD"+D+"F");
        apFresh.setAttribute("string", fresh+"lt");
    }
}
```

Figure 5.53 Javascript function updating the flowmeters' values

The tenant may see four possible combinations of flow mesurments when he has selected to consume treated grey water at his toilet cistern. So, when someone at his apartment uses the toilet cistern, he can see a value (greater than zero) on the grey floemeter's text area in the 3D scene where his apartmen is displayed (Figure 5.56). If a faucet is also use, e.g. at his kitchen, then a value on his fresh water floemeter's text area in the 3D scene will appear too (Figure 5.57). When the treated grey water has filled the cistern, the value of the corresponding text area shows a zero value (as the grey water does not continue to run through the pipeline) and the fresh water text may (Figure 5.54) or may not (Figure

5.55) show a zero fresh water consumption depending on whether the faucet has been closed or not, respectively.

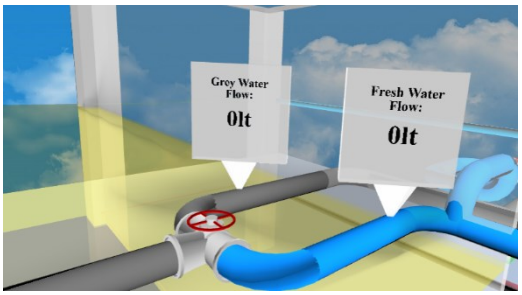


Figure 5.54 Flowmeters' values - Treated water selected - No usage

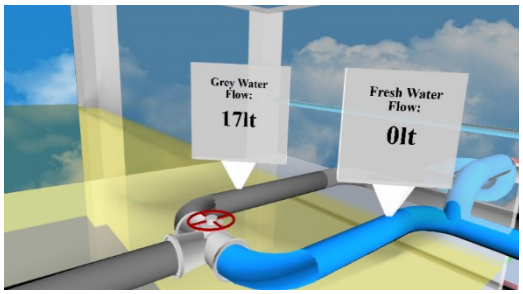


Figure 5.56 Flowmeters' values - Treated water selected - Grey is used

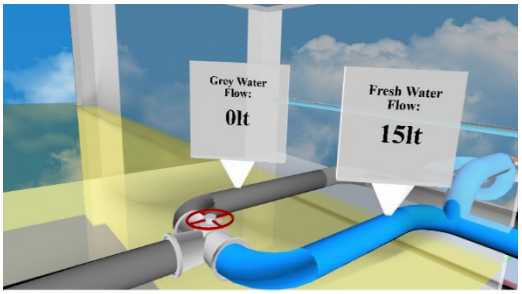


Figure 5.55 Flowmeters' values - Treated water selected - Fresh is used

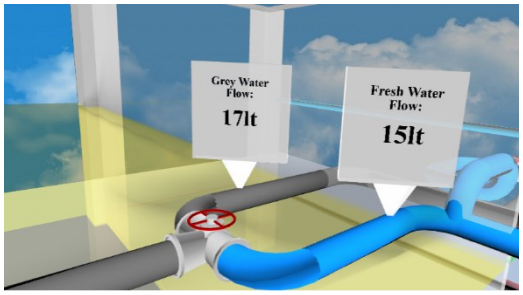


Figure 5.57 Flowmeters' values - Treated water selected - Grey and Fresh is used

On the other hand, when the tenant has selected to consume only fresh water at his apartment, a zero value will be displayed at the grey flowmeter's text area, and a non-zero (Figure 5.59) or a zero value (Figure 5.58) will displayed at the fresh water's flowmeter text area, depending on whether a faucet (or the cistern) is at use or not, respectively.

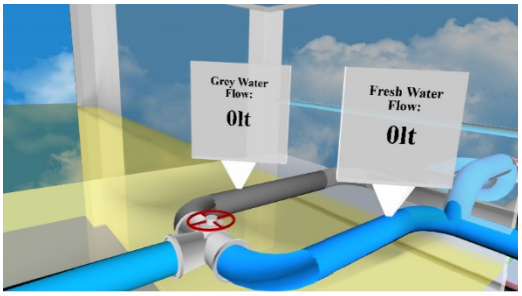


Figure 5.58 Flowmeters' values - Fresh water selected - No usage

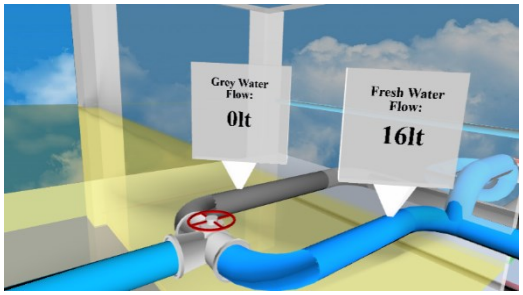


Figure 5.59 Flowmeters' values - Fresh water selected - Fresh is used

User action listeners

Every scada platform must has the ability to provide some kind of control over the system to the user. On this 3D model of the house there are five actuators (electrovalves) which the user can press to enable or disable the usage of recycled water to that area. As it was mentioned in ***Roles and profiles*** only administrators is permitted to access all parts of the system. Following the same principals on the 3D model, administrators have full access over the actuators but tenants are restricted to their apartment.

To handle these actions at the 3D model, javascript listeners have been assigned to every actuator's graphical representation; binding them by their ID name. To manage the different user permissions, the javascript code retrieves the saved cookie (created at the login) in order to recognize the user and enable only the listeners which this user controls.

When a user presses an electrovalve at the 3D model, an animation of a turning valve handle is displayed, the appropriate texts are updated, and a function handling the DB update is called. This function sends, via Ajax post method, these changes to a servlet in order to update the actuators' table in the database.

```
function checkCookie()
{
    var name = 'userid' + "=";
    var mynull = 10;
    var ca = document.cookie.split(';');
    for(var i = 0; i <ca.length; i++)
    {
        var c = ca[i];
        while (c.charAt(0)==' ') {
            c = c.substring(1);
        }
        if (c.indexOf(name) == 0) {
            return c.substring(name.length,c.length);
        }
    }
    return mynull;
}
```

Figure 5.60 Javascript fuction checks user's identity

```
var ElectroValveGarden = document.getElementById('GroupElectroValveGarden');
if(mycookie<3)
    ElectroValveGarden.addEventListener('mousedown', gardenValvePressed, false);
function gardenValvePressed()
{
    var left = document.getElementById('timeRight');
    animation(left,'Garden');
    if(objJSON.actuators[9].profile == 1)
        ubdateDB(0, 2);
    else
        ubdateDB(0, 1);
};
```

Figure 5.61 Javascript assigns eventListener if the user is admin

Chapter 6 - **Conclusion and Future work**

This thesis is part of a larger ICT work developed for an IoT system installed at a four apartment building at Iraklion, Crete, Greece [38]. This IoT system is a complete water management system consisting of three subsystems for harvesting the wastewater, treating it and then distributing it for reuse. The harvesting system consists mostly of directional pipe lines steering the wastewater into two different tanks, to be treated separately as grey or black. Part of the harvesting system is alslo the intelligent system developed to separate the black from the grey water of the rainfall wastewater from the roof's surface. This part of the harvesting system was designed and implemented with a fuzzy logic controller which handles the separation between the two water quantities. The controller uses the concept of the first flush event to calculate the time after which this event is considered expired and thus the water is cleaner, i.e. grey.

After the treatment of the collected wastewater, the filtered grey and black water is forwarded into two tanks to be reused. The grey water can be consumed for both roof garden watering and by apartments' toilet cisterns. The treated black water can be used only for garden watering, mostly for health safety issues.

For providing monitoring and control over the distributing system and the water consumptions of the whole building infrastructure, a web application was developed with a 3D Scene of the building working as a Human to Machine Interface. This application provides personalized content depending on the signed-in user. So, a tenant can monitor and control only his apartment, while the building administrator can monitor and control the whole building infrastructure, even overwriting the tenant's decisions if he deems necessary, e.g. with a low level in the grey water reserve, the administrator can overwrite the selection of all apartments' usages turning them to fresh.

This application provides detailed consumption reports of every part of the building infrastructure, for all three water types (fresh, treated grey and black). The user can select a specific date to study the daily and weekly provided charts, or oversee the summarized consumptions through the monthly and yearly charts. This can motivate tenants to continue using the treated water, as they can observe both the economic and ecological benefits. The updated page with the current treated water reserves and two pages with the current weather conditions of the surrounding area of the building and a weather forecast page included on the application, can also provide information to the users facilitating their

decision on the water type selection. The water type selection page provides the user with two predefined profiles for optimized usage, but also with the option to disable, if wished, the treated water supply from his apartment.

To extend further the applications capabilities, but also to provide a more comprehensive spatial view tool with all the necessary control capabilities, a real time monitoring and control Human to Machine Interface was created with a 3D visualization of the building infrastructure, and added to the web app. This extension is a SCADA user interface from which the end user monitors, in real time, all that is happening to the building infrastructure, but also controls and interacts with its parts. This is a rich interface environment since it's equipped with an interactive 3D scene that doesn't need any additional software to be installed for it to be displayed. A 3D representation is more intuitive for human operators since it provides many valuable comfort features, such as zoom, navigation or viewing of different perspectives [28].

There has been observed that most SCADA UIs, using W3C standards, are limited to 2D-models or pictures of technical components, in visualizing the automation system process and real-time data. When data is provided in a 2D form, live shapes, processes, spatial cognition and other details are lost or displayed in a very abstract way. Considering the complexity of automation systems and the amount of corresponding data, a 3D-image is technically the best way to display these data [27]. On the other hand real-time sensor feeds can transform both decision making and environmental science, they are rarely used in real-time workflows, analyses and modeling tool chains [15]. Modern sensor systems can connect to the Internet via standard web protocols; this permits the use of web services as a perfect interoperability mechanism between sensors, actuators and decision support systems [15]. Embedded web-servers with advanced features (such as server push for event notifications or concurrent connections), can be implemented with no operating system support and very low memory capacity, thanks to efficient cross-layer TCP/HTTP optimizations, and therefore can run on tiny embedded systems [17]. Since embedded web-servers in an IoT system generally have fewer resources than web-clients (i.e. browsers, mobile phones, tablets), Asynchronous JavaScript and XML (Ajax) has proven to be a good way of transferring the server workload to the clients [18].

Current work takes advantage of these features and provides an open source web-based framework which is both a modular and a cost effective solution, as a Human to Machine Interface product; in contrast to the majority of the existing proprietary desktop

software. The designed framework can be easily modified and installed to a variety of IoT systems covering the need of a HMI, both for local and remote real-time monitoring and control. Adding more automated control functions, such as water leakage detection or electrical energy consumption monitoring, there can be created a complete standalone solution for any building infrastructure.

References

[1] Z. Li, F. Boyle and A. Reynolds, "Rainwater harvesting and greywater treatment systems for domestic application in Ireland," *Desalination,* vol. 260, no. 1-3, pp. 1-8, 2010.

[2] F. A. Abdulla and A. Al-Shareef, "Roof rainwater harvesting systems for household water supply in Jordan," *Desalination,* vol. 243, pp. 195-207, 2009.

[3] G. A. Cagalaban, J.-g. Song, S. Jung and S. s. Kim, "Software Vulnerability Design and Approaches for Securing SCADA Control Systems," *International Journal of Smart Home,* vol. 3, no. 1, pp. 49-56, 2009.

[4] M. Choras, A. Flizikowski, R. Kozik and W. Holubowicz, "Decision Aid Tool and Ontology-Based Reasoning for Critical Infrastructure Vulnerabilities and Threats Analysis," in *Critical Information Infrastructures Security: 4th International Workshop*, Bonn, Germany, 2009.

[5] P. Vijai and B. Sivakumar P., "Design of IoT Systems and Analytics in the context of Smart City," in *2nd International Conference on Intelligent Computing, Communication & Convergence*, Bhubaneswar, Odisha, India, 2016.

[6] T. Robles, R. Alcarria, D. Martín, A. Morales, M. Navarro, R. Calero, S. Iglesias and M. López, "An Internet of Things-based model for smart water management," in *28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, Canada, 2014.

[7] T. Robles, R. Alcarria, D. Martín, M. Navarro, R. Calero, S. Iglesias and M. López, "An IoT based reference architecture for smart water management processes," *JoWUA,* vol. 6, no. 1, pp. 4-23, 2015.

[8] C. Turcu, C. Turcu and V. Gaitan, "An Internet of Things Oriented Approach for Water Utility Monitoring and Control," in *Proceedings of the 6th WSEAS European Computing Conference*, Prague, Czech Republic, 2012.

[9] S. Fang, L. Xu, H. Pei, Y. Liu, Z. Liu, Y. Zhu, J. Yan and H. Zhang, "An Integrated Approach to Snowmelt Flood Forecasting in Water Resource Management," *IEEE Transactions on Industrial Informatics,* vol. 10, no. 1, pp. 548-558, 2014.

[10] J. J. Hilda, C. Srimathi and B. Bonthu, "A Review on the Development of Big Data Analytics and Efective Data Visualization Techniques in the Context of Massive and Multidimensional Data," *Indian Journal of Science and Technology,* vol. 9, no. 27, pp. 1-13, 2016.

[11] C. M. Thürlimann, D. J. Dürrenmatt and K. Villez, "Energy and process data processing and visualisation for optimising wastewater treatment plants," *Water Practice and Technology,* vol. 10, no. 1, pp. 10-18, 2015.

[12] N. Andrienko and G. Andrienko, Exploratory Analysis of Spatial and Temporal Data, A Systematic Approach, Germany: Springer-Verlag Berlin Heidelberg, 2006.

[13] W. Aigner, S. Miksch, W. Müller, H. Schumann and C. Tominski, "Visualizing Time-Oriented Data - A Systematic View," *Computers and Graphics,* vol. 31, no. 3, pp. 401-409, 2007.

[14] T. Cheng and J. Teizer, "Real-time resource location data collection and visualization technology for construction safety and activity monitoring applications," *Automation in Construction,* vol. 34, pp. 3-15, 2013.

[15] B. P. Wong and B. Kerkez, "Real-time environmental sensor data: An application to water quality using web services," *Environmental Modelling & Software,* vol. 84, pp. 505-517, 2016.

[16] L. Díaz, A. Bröring, D. McInerney, G. Libertá and T. Foerster, "Publishing sensor observations into Geospatial Information Infrastructures: A use case in fire danger assessment," *Environmental Modelling & Software,* vol. 48, pp. 65-80, 2013.

[17] S. Duquennoy, G. Grimaud and J.-J. Vandewalle, "The Web of Things: interconnecting devices with high usability and performance," in *6th IEEE International Conference on Embedded Software and Systems*, HangZhou, Zhejiang, China, 2009.

[18] D. Guinard, V. Trifa, F. Mattern and E. Wilde, "From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices," in *Architecting the Internet of Things*, Springer Berlin Heidelberg, 2011, pp. 97-129.

[19] M. Stangl, J. Pielmeier, C. Berger, S. Braunreuther and G. Reinhart, "Development of a Web Based Monitoring System for a Distributed and Modern Production," *Procedia CIRP,* vol. 52, pp. 222-227, 2016.

[20] J. D. Walker and S. C. Chapra, "A client-side web application for interactive environmental simulation modeling," *Environmental Modelling & Software,* vol. 55, pp. 49-60, 2014.

[21] A. Syberfeldt, I. Karlsson, A. Ng, J. Svantesson and T. Almgren, "A web-based platform for the simulation–optimization of industrial problems," *Computers & Industrial Engineering,* vol. 64, pp. 987-998, 2013.

[22] R. J. Robles and T.-h. Kim, "Security Encryption Scheme for Communication of Web Based Control Systems," in *Signal Processing and Multimedia*, Germany, Springer Berlin Heidelberg, 2010, pp. 317-325.

[23] IntegraXor, "Ecava," Ecava, 2016. [Online]. Available: http://www.ecava.com/what-is-web-scada/ . [Accessed 25 October 2016].

[24] "SVG IMAGE LIBRARY," ECAVA, [Online]. Available: https://www.ecava.com/svg-image-library/. [Accessed 20 11 2016].

[25] "Rapid SCADA," Rapid SCADA, 2016. [Online]. Available: http://rapidscada.org/download-all-files/. [Accessed 31 October 2016].

[26] "Rapid SCADA," 2016. [Online]. Available: http://rapidscada.org/download-all-files/. [Accessed 25 October 2016].

[27] R. Agrusa, V. G. Mazza and R. Penso, "Advanced 3d visualization for manufacturing and facility controls," in *2nd Conference on Human System Interactions*, Catania, Italy, 2009.

[28] S. Feldhorst, M. Fiedler, M. Heinemann, M. t. Hompel and H. Krumm, "Event-based 3D-Monitoring of Material Flow Systems in Real-Time," in *8th IEEE International Conference on Industrial Informatics (INDIN)* , Osaka, Japan, 2010.

[29] "iCONICS," ICONICS, Inc., 2016. [Online]. Available: http://www.iconics.com/Home/Products/HMI-SCADA/GENESIS64/GraphWorX64.aspx#.WBcunS195hF. [Accessed 31 October 2016].

[30] "RIVOPS - Remote Intuitive Visual OperationS," Remote Intuitive Visual Operations, 2006-2016. [Online]. Available: http://www.rivops.com/technology/. [Accessed 31 October 2016].

[31] "3d SCADA Solutions," Scada3D Solutions, [Online]. Available: http://scada3d.es/en/home/. [Accessed 2016 October 2016].

[32] H. A. Abbas and A. M. Mohamed, "Review on the Design of Web Based SCADA Systems Based on OPC DA Protocol," *International Journal of Computer Networks (IJCN),* vol. 2, no. 6, pp. 266 - 277, 2011.

[33] S. Ferdoush and X. Li, "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications," *Procedia Computer Science,* vol. 34, pp. 103-110, 2014.

[34] J. Yang, C. Zhang, X. Li, Y. Huang, S. Fu and M. F. Acevedo, "Integration of wireless sensor networks in environmental monitoring cyber infrastructure," *Wireless Networks,* vol. 16, no. 4, pp. 1091-1108, 2010.

[35] B. Murovec, J. Perš, R. Mandeljc, V. S. Kenk and S. Kovačič, "Towards commoditized smart-camera design," *Journal of Systems Architecture,* vol. 59, no. 10, pp. 847-858, 2013.

[36] B. Fabian and T. Feldhaus, "Privacy-preserving data infrastructure for smart home appliances based on the Octopus DHT," *Computers in Industry,* vol. 65, no. 8, pp. 1147-1160, 2014.

[37] F. Leccese, M. Cagnetti and D. Trinca, "A smart city application: a fully controlled street lighting isle based on Raspberry-Pi card, a ZigBee sensor network and WiMAX," *Sensors,* vol. 14, pp. 24408-24424, 2014.

[38] M. Kalochristianakis, D. Katrinakis, G. Atsali, A. Malamos, T. Manios and S. Panagiotakis, "HOLISTIC: An IoT system for residential water recycling based on open source technologies," in *International Conference on Telecommunications and Multimedia*, Heraklion, Crete, Greece, 2016.

[39] C. Vialle, C. Sablayrolles, M. Lovera, S. Jacob, M. C. Huaue and M. Montrejaud-Vignolesa, "Monitoring of water quality from roof runoff: Interpretation using multivariate analysis," *Water Research,* vol. 45, no. 12, p. 3765–3775, June 2011.

[40] R. Farreny, T. Morales-Pinzon, A. Guisasola, C. Taya, J. Rieradevall and X. Gabarrell, "Roof selection for rainwater harvesting: Quantity and quality assessments in Spain," *Water Research,* vol. 45, no. 10, p. 3245–3254, May 2011.

[41] A. Rouvalis, C. Karadima, I. V. Zioris, V. A. Sakkas, T. Albanis and J. Iliopoulou-Georgudaki, "Determination of pesticides and toxic potency of rainwater samples in western Greece," *Ecotoxicology and Environmental Safety,* vol. 72, no. 3, p. 828–833, March 2009.

[42] W. Ahmed, F. Huygens, A. Goonetilleke and T. Gardner, "Real-Time PCR Detection of Pathogenic Microorganisms in Roof-Harvested Rainwater in Southeast Queensland, Australia," *Aapplied and Environmental Microbiology,* vol. 74, no. 17, p. 5490–5496, July 2008.

[43] C. B. Mendez, B. J. Klenzendorf, B. R. Afshar, M. T. Simmons, M. E. Barrett, K. A. Kinney and M. J. Kirisits, "The effect of roofing material on the quality of harvested rainwater," *Water Research,* vol. 45, no. 5, p. 2049–2059, February 2011.

[44] G. Atsali, D. Katrinakis, S. Panagiotakis, D. Athanasaki, M. Kalochristianakis, T. Manios and A. Malamos, "First Flush Rainwater Harvesting Application with Fuzzy Logic Control," in *International Symposium on Ambient Intelligence and Embedded Systems*, Heraklion, Crete, Greece, 2016.

[45] P. Cingolani and J. Alcalá-Fdez, "jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming," *International Journal of Computational Intelligence Systems,* vol. 6, no. 1, pp. 61-75, March 2013.

[46] N. J. Schouten, M. A. Salman and N. A. Kheir, "Energy management strategies for parallel hybrid vehicles using fuzzy logic," *Control Engineering Practice,* vol. 11, no. 2, p. 171–177, February 2003.

[47] D. Seo and J. Lee, "Direct hand touchable interactions in augmented reality environments for natural and intuitive user experiences," *Expert Systems With Applications,* vol. 40, no. 9, pp. 3784-3793, 2013.

[48] T. N. Eicke, Y. Jung and A. Kuijper, "Stable dynamic webshadows in the X3DOM framework," *Expert Systems with Applications,* vol. 42, no. 7, pp. 3586-3609, 2015.

[49] M. Olbrich, T. Franke and P. Rojtberg, "Remote visual tracking for the (mobile) web," in *The 19th international conference on Web3D Technology*, Vancouver, BC, Canada, 2014.

[50] C. Stein, M. Limper and A. Kuijper, "Spatial data structures for accelerated 3d visibility computation to enable large model visualization on the web," in *The 19th international conference on Web3D technology*, Vancouver, BC, Canada, 2014.

[51] A. Aderhold, K. Wilkosinska, M. Corsini, Y. Jung, H. Graf and A. Kuijper, "The common implementation framework as service – Towards novel applications for streamlined presentation of 3d content on the web," in *Design, User Experience, and Usability: User Experience Design for Diverse Interaction Platforms and Environments, Third International Conference, DUXU 2014*, Heraklion, Crete, Greece, 2014.

[52] "3ds Max Design," Autodesc, Inc., 2016. [Online]. Available: http://www.autodesk.com/education/free-software/3ds-max-design. [Accessed 19 September 2016].

[53] "3ds Max Export," Fraunhofer-Gesellschaft, [Online]. Available: http://www.x3dom.org/documentation/tutorials/3ds-max-export/. [Accessed 19 September 2016].

[54] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo and J. Blat, "3D graphics on the web: A survey," *Computers & Graphics,* vol. 41, pp. 43-61, June 2014.

[55] K. Beihang, "W3C - HTML & CSS," 2016. [Online]. Available: https://www.w3.org/standards/webdesign/htmlcss. [Accessed 16 September 2016].

[56] T. Parisi, "HTML5: A New Visual Medium," in *Programming 3D Applications with HTML5 and WebGL*, O'Reilly Media, 2013, p. 300.

[57] S. Ringe, R. Kedia, A. Poddar and S. Patel, "HTML5 Based Virtual Whiteboard for Real Time Interaction," in *4th International Conference on Advances in Computing, Communication and Control*, 2015.

[58] D. Mitrović, M. Ivanović, Z. Budimac and M. Vidaković, "Radigost: Interoperable web-based multi-agent platform," *Journal of Systems and Software,* vol. 90, pp. 167-178, 2014.

[59] "The Java EE5 Tutorial," Oracle and/or its affiliates, September 2010. [Online]. Available: http://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html. [Accessed 19 September 2016].

[60] S. Rakhtala and E. S. Roudbari, "Application of PEM Fuel Cell for Stand-alone Based on a Fuzzy PID Control," *Bulletin of Electrical Engineering and Informatics,* vol. 5, no. 1, pp. 45-61, March 2016.

[61] S. F. Rezeka, A.-H. Attia and A. M. Saleh, "Management of air-conditioning systems in residential buildings by using fuzzy logic," *Alexandria Engineering Journal,* vol. 54, p. 91–98, March 2015.

[62] A. Saffioti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Computing,* vol. 1, no. 4, pp. 180-197, December 1997.

[63] P. T. Roseline, N. Ganesan and C. J. Tauro, "A Study of Applications of Fuzzy Logic in Various Domains of Agricultural Sciences," in *International Conference on Current Trends in Advanced Computing*, Bengaluru, Karnataka, 2015.

[64] S. M. Rahman and N. T. Ratrout, "Review of the Fuzzy Logic Based Approach in Traffic Signal Control: Prospects in Saudi Arabia," *Journal of Transportation Systems Engineering and Information Technology,* vol. 9, no. 5, p. 58–70, October 2009.

[65] C. G. Amza and D. T. Cicic, "Industrial Image Processing Using Fuzzy-Logic," in *25th DAAAM International Symposium on Intelligent Manufacturing and Automation*, Vienna, Austria, 2014.

[66] T. Korol, "A fuzzy logic model for forecasting exchange rates," *Knowledge-Based Systems,* vol. 67, pp. 49-60, September 2014.

[67] V. Vella and W. Lon Ng, "Improving risk-adjusted performance in high frequency trading using," *Expert Systems With Applications,* 11 February 2016.

[68] A. Fernández, V. López, M. José del Jesus and F. Herrera, "Revisiting Evolutionary Fuzzy Systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Systems,* vol. 80, p. 109–121, May 2015.

[69] C. M. Salgado, S. M. Vieira, L. F. Mendonça, S. Finkelstein and J. M. Sousa, "Ensemble fuzzy models in personalized medicine: Application to vasopressors administration," *Engineering Applications of Artificial Intelligence,* vol. 49, pp. 141-148, March 2016.

[70] P. Domingues, P. Carreira, R. Vieira and W. Kastner, "Building automation systems: Concepts and technology review," *Computer Standards & Interfaces,* vol. 45, p. 1–12, March 2016.

[71] J. Victor and A. D. Correia, "Wuzzy: A Real-Time Fuzzy Control Tool And Its Application," in *Algorithms and Architectures for Real-Time Control 1997*, Vilamoura, Portugal, 1997.

[72] L. Domènech and D. Saurí, "A comparative appraisal of the use of rainwater harvesting in single and multifamily buildings of the Metropolitan Area of Barcelona (Spain): social experience, drinking water savings and economic costs," *Journal of Cleaner Production,* vol. 19, pp. 598-608, 2011.

[73] A. Gómez, D. Cuiñas, P. Catalá, L. Xin, W. Li, S. Conway and D. Lack, "Use of Single Board Computers as Smart Sensors in the Manufacturing Industry," in *The Manufacturing Engineering Society International Conference, MESIC*, Barcelona, 2015.

[74] F. C. Delicato, P. F. Pires, L. Pirmez and L. F. R. d. C. Carmo, "A flexible web service based architecture for wireless sensor networks," in *The 23rd International Conference on Distributed Computing Systems Workshops*, Washington, 2003.

[75] M. Limper, Y. Jung, J. Behr, T. Sturm, T. Franke, K. Schwenk and A. Kuijper, "Fast, progressive loading of binary-encoded declarative-3d web content," *IEEE Computer Graphics and Applications,* vol. 33, no. 5, pp. 26-36, 2013.

[76] P. Melin and O. Castillo, "A review on the applications of type-2 fuzzy logic in classification and pattern recognition," *Expert Systems with Applications,* vol. 40, no. 13, p. 5413–5423, October 2013.

[77] B. S. J. Costa, C. G. Bezerra and L. A. H. G. de Oliveira, "Java fuzzy logic toolbox for industrial process control," in *XVIII Congresso Brasileiro de Automática*, Bonito-MS, 2010.

[78] E. Sierra, A. Hossian, P. Britos, D. Rodriguez and R. Garcia-Martinez, "Fuzzy Control For Improving Energy Management Within Indoor Building Environments," in *Electronics, Robotics and Automotive Mechanics Conference*, Morelos, Mexico, 2007.

[79] M. E. Raynor and P. Wilson, "Beyond the dumb pipe: The IoT and the new role for network service providers," *The Internet of Things in telecom,* 2 September 2015.

Appendix A - Technologies used in the web application

HTML - CSS

Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) are two of the core technologies used to build web pages. HTML supply the structure of the page and CSS the visual layout. Along with graphics and scripting, HTML and CSS are the basic components for building web pages and web applications [55]. There is a variety of devices supporting these technologies and with the development of html5 and responsive templates these technologies are spreading to event more.

HTML5 is the 5th version of HTML. It offers richer media support, enchases the user-application-server interactions, supports game development, 3d graphical models and much more. With HTML5, the web browser has become a capable platform which can run sophisticated applications that can compete with native code both in features and performance [56].

Javascript

Javascript is an object-oriented language which can interact with HTML code, providing sites with dynamic content. Javascript can be directly embedded to the html file or saved as a separate file with the extension .js. The latter can be reused on multiple pages. Javascript has the advantage on running in client's side, locally. This can provide user with a more responsive experience as it reacts to user actions more quickly. This language can be used to develop client-side scripts as well as server-side programs [57].

JSON

JavaScript Object Notation (JSON) is a format for data interchange. It has a simpler syntax and it is less data-fraught than XML. It is also more restrictive than XML as it supports a smaller number of data formats. XML can include audio, video, photos and other formats. This can be a threat as it can include an executable file. JSON is easy for a person to read and write, as well as for a machine to parse and generate. JSON is lightweight so it is faster at serialization and deserialization of data [57].

AJAX

Asynchronous JavaScript and XML (AJAX) is a technique for creating dynamic and fast web pages. Ajax provides developers with the ability to communicate with the server, requesting row data and updating parts of the page without the need to reloading the whole page. This improves the response time to client-side applications as the data is loaded in the background and then displayed on the webpage [20]. Figure 0.1 and Figure 0.2 present the difference between traditional communication of a web application and Ajax.

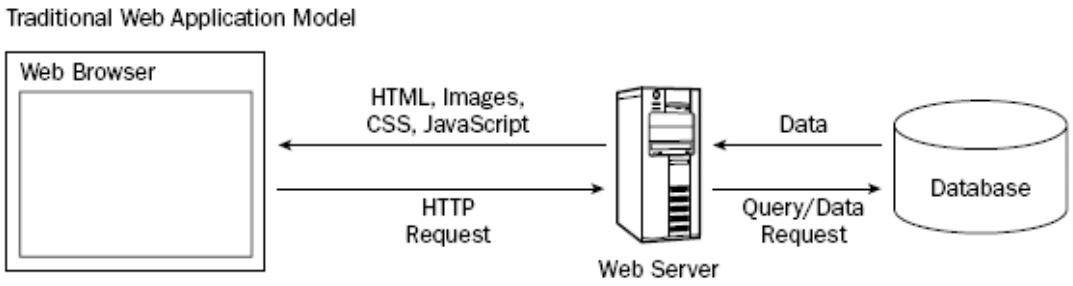


Figure 0.1 Traditional Web Application Model

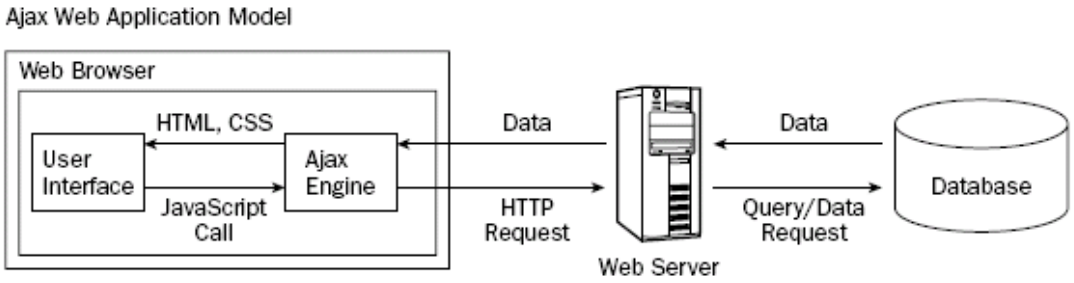


Figure 0.2 AJAX Web Application Model

Web Sockets

Web Socket is a protocol enabling a full-duplex communication over a single TCP connection between web clients and servers. This type of connection provides simultaneous, two way exchange of text data and binary information over the same channel. Modern web browsers support web sockets protocol [58].

Java Servlets

A servlet is a Java class which is used to extend servers capabilities that host applications. These particular applications are accessed by a request-response programming model to provide a dynamic and user-oriented content. Servlets can respond to any type of requests but they are commonly used by web servers handling http request-response [59].

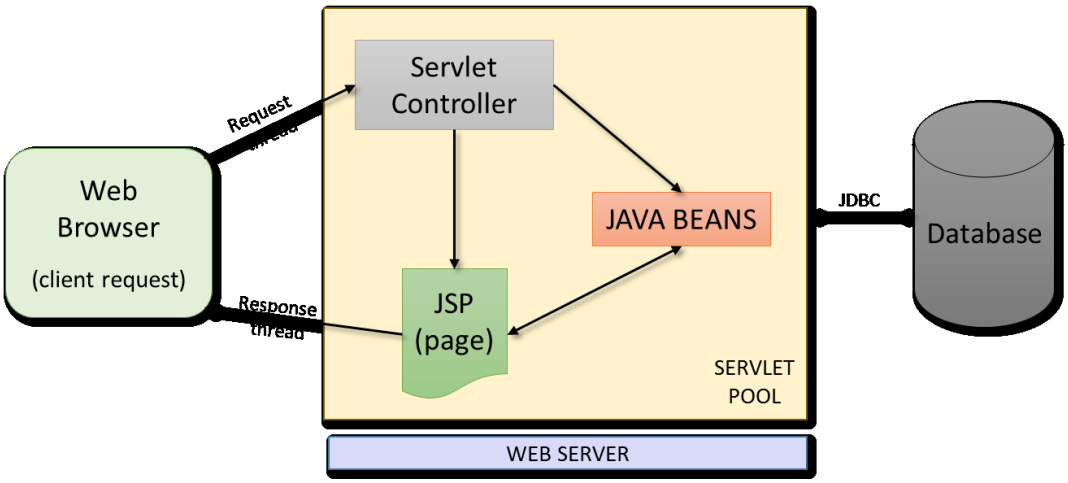


Figure 0.3 Java servlet configuration diagram

Fuzzy Logic

Fuzzy reasoning is a powerful and straightforward means of problem solving, which is designed on the basis of human experience [60] and provides a controller without the requirement of a mathematical model [61]. An important problem with first flush method is the need to cope with the large amount of uncertainty, which is inherent of natural environments. Fuzzy logic's features make it a suitable tool to address this problem [62].

Fuzzy logic (FL) has emerged as an important part of expert system which has proved to provide solution to real life problems [63]. It has been successfully applied in a variety of fields ranging from engineering, to financial sector, from medicine, to agriculture and many more, for over twenty years.

Saffiotti in [62] reviews some of the possible uses of fuzzy logic in the field of autonomous navigation in real-world environments. He focuses on four issues: how to design behavior-producing modules; how to coordinate the activity of several such modules; how to use the sensors' data; and how to integrate low-level execution with high-level reasoning.

Fuzzy logic approaches for traffic signal control were reviewed in [64]. The results of these applications indicate better performance than traditional traffic signal controls, especially during heavy volume and uneven traffic conditions.

An automatic analysis of X-ray images was performed in [65] using fuzzy logic techniques, for the inspection process of industrial manufacturing systems. The fuzzy reasoning was applied to decide whether the object, displayed on the image, was defected or not.

For forecasting exchange rates, a predictive model with the use of fuzzy logic theory was created in [66]. To evaluate its effectiveness it was then tested with data from times of prosperity and during financial crisis, with high efficiency rates.

In [67] the authors introduce type-2 fuzzy sets in intelligent trading algorithms, in order to improve the risk-adjusted performance with the minimum increase of the design and computational complexity.

Hybrid systems, which integrate both the management of uncertainty and inherent interpret-ability of fuzzy rule based systems, with the learning and adaptation capacity of evolutionary optimization, have become popular in the last years. A review of the progression of evolutionary fuzzy systems has been done in [68], analyzing their taxonomy and components.

A fuzzy c-means clustering algorithm was implemented in [69] to find subgroups of patients, and then develop a fuzzy model for each subgroup to achieve personalized healthcare for individual patients of ICU, fitted to the individual rather than the “average” patient.

A study of fuzzy logic integrations with expert systems which has been applied in the field of agricultural sciences is presented in [63] along with a review of a few of applications.