University of Coimbra

Master's Thesis

Platform for the supervision of remote systems using low cost devices

Author:
Vitor Sousa
vhsousa@student.dei.uc.pt

Supervisor:
Professor Alberto Cardoso
alberto@dei.uc.pt

Master's in Informatics Engineering

Laboratory of Industrial Informatics and Systems
Department of Informatics Engineering

July 6, 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

University of Coimbra

Master's Thesis

Platform for the supervision of remote systems using low cost devices

Author:
Vitor Sousa
vhsousa@student.dei.uc.pt

Supervisor:
Professor Alberto Cardoso
alberto@dei.uc.pt

Jury:

President: Professor Fernando José BARROS

Vowels: Professor Jorge HENRIQUES
Professor Alberto CARDOSO

Master's in Informatics Engineering

Laboratory of Industrial Informatics and Systems
Department of Informatics Engineering

July 6, 2015

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

O desenvolvimento da experimentação on-line representa uma oportunidade para a criação de laboratórios remotos, abordando diversos temas em diversas áreas, especialmente em temas relacionados com cursos de engenharia.

O principal objetivo deste trabalho é o desenvolvimento de uma plataforma que integra, monitoriza e controla sistemas laboratoriais, através de uma forma uniforme de interação com diversos equipamentos e tecnologias. A arquitetura proposta considera a utilização de dispositivos de baixo custo, tais como os Raspberry Pi, num ambiente distribuído acessível através da internet e que permite a interação com dispositivos de redes de sensores sem fios (e.g. TelosB), placas de aquisição de dados (e.g. National Instruments) e redes de sensores virtuais. Os utilizadores têm a possibilidade de observar os sistemas laboratoriais através de uma web câmara.

Com o suporte da plataforma desenvolvida, foram criados cinco casos de estudo para testar e avaliar as suas funcionalidades e o seu desempenho. Os casos de estudo foram usados para a colaboração com algumas instituições e demonstrados em dois eventos: Human Sensor and Geographic Information Systems for Disaster Risk Management (HSenSIG) e a 3rd Experiment@International Conference - exp.at'15.

Os resultados obtidos através dos testes de benchmarking, monitorização da utilização e comentários dos utilizadores, dão-nos boas perspetivas para a sua utilização numa escala mais alargada, possibilitando uma maior qualidade à experiência de aprendizagem, para diferentes utilizadores.

Keywords: Experimentação On-line, Laboratórios Remotos, Integração de Sistemas, Redes de Sensores sem Fios, Aquisição de dados, Redes de Sensores Virtuais, Sistemas Distribuídos.

Abstract

The development of on-line experimentation represents an opportunity to create remote laboratories addressing several topics in different areas, specially on topics of engineering courses.

The main purposes of this work are the development of a platform that integrates, monitors and controls laboratory systems, by creating a uniform way of interaction with different equipment and technologies. The proposed architecture considers low-cost devices, such as the Raspberry Pi, in a distributed environment that is accessible over the Internet allowing the interaction with wireless sensor network devices (e.g. TelosB), data acquisition boards (e.g. National Instruments) and virtual sensor networks. The users can observe the laboratory systems through a web camera.

With the support of the developed platform, were created five study cases to test and evaluate its features and performance. They were used on the collaboration with some institutions and demonstrated on two events: Human Sensor and Geographic Information Systems for Disaster Risk Management (HSenSIG) and the 3rd Experiment@International Conference - exp.at'15.

The results obtained with the benchmarking testing, usage monitoring and feedback from users gives good perspectives for its usability at a larger scale, providing a higher quality of learning experience to different students.

Keywords: On-line Experimentation, Remote Laboratories, Systems Integration, Wireless Sensor Networks, Data Acquisition, Virtual Sensor Networks, Distributed Systems.

Acknowledgements

First of all I would like to express my gratitude to Professor Alberto Cardoso for giving me the opportunity to work alongside him and thank him for his guidance, patience and availability. Without those, this work would never get as far as it became.

To my colleagues who worked in parallel projects that were using the work presented on this thesis, I want to acknowledge my gratitude for their patience when things weren't always working as they should be.

To my beloved, who helped me through all of this time by providing shelter, food and patience when the work was coming in huge quantities and the deadlines were close, or through great reviews some of my written work. Definitely this was a fundamental and truly important piece to me.

I would also like to thank my friends for all the amiability and support given through all of these years together. Definitely I got by with a little help from my friends.

Last but not least, I want to thank my dad, mom, brothers and my awesome dog, *faísca*, who provided all the support, including financial, encouragement and comprehension during all of these years.

To all of you, thanks!

Contents

List of Figures

List of Tables

Abbreviations

ADC	**A**nalog-to-**D**igital **C**onverter
API	**A**pplication **P**rogramming **I**nterface
DAB	**D**ata **A**cquisition **B**oard
DAC	**D**igital-to-**A**analog **C**onverter
FIFO	**F**irst **I**n **F**irst **O**ut
FCFS	**F**irst **C**ome, **F**irst **S**erved
GPIO	**G**eneral **P**urpose **I**nput **O**utput
IP	**I**nternet **P**rotocol
JWT	**J**SON **W**eb **T**oken
NaN	**N**ot **a** **N**umber
NTP	**N**etwork **T**ime **P**rotocol
QA	**Q**uality **A**ssurance
RAM	**R**andom **A**ccess **M**emory
ROM	**R**ead-**O**nly **M**emory
RPi	**R**aspberry **Pi**
SMTP	**S**imple **M**ail **T**ransport **P**rotocol
SPOF	**S**ingle **P**oint **O**f **F**ailure
TCP	**T**ransport **C**ontrol **P**rotocol
URL	**U**niform **R**esource **L**ocator
VGI	**V**olunteered **G**eographic **I**nformation
VSN	**V**irtual **S**ensor **N**etwork
WSN	**W**ireless **S**ensor **N**etwork

Chapter 1

Introduction

This thesis corresponds to the research work done at the Laboratory of Industrial Informatics and Systems of the Department of Informatics Engineering within the Thesis/Project curricular unit of the Master in Informatics Engineering course at the University of Coimbra.

1.1 Motivation

The development of Information and Communication Technologies (ICT) enables many benefits and advances regarding all components of the educational process and play an increasingly fundamental role to support teaching and on-line learning in engineering courses. In particular, the development of on-line experimentation represents an opportunity to create remote laboratories addressing several topics in different areas, specially on topics of engineering courses.

Higher education institutions have to provide learning experiences that engage and addresses the needs of society in the twenty-first century [4]. As Swail [5] states, the "rules are changing and there is increased pressure on institutions of higher education to evolve, adapt, or desist". The transformation of teaching and learning on higher education is inevitable with the use of web-based communications technology [6]. The field of on-line learning, in general, should incorporate the potential of technology to address the challenges associated with, providing a high-quality learning experience in different educational contexts and using diverse technological supports and interfaces.

The recent large turnout to the Internet brought new means to share information and knowledge not only in terms of content but also as tools and platforms that allow educators to create innovative ways to support learning. A major outcome from this effort is the on-line laboratory paradigm. Remote laboratories provide to students the abilities to interact, in real time and with less access restrictions, with a laboratory system to perform practical experiences, visualizing and analyzing the dynamic behavior of a system.

Some laboratory systems were designed to be used on a restricted environment and do not provide any interface to collect data on a digital format, for further usage. Also, some of the experiments may only involve the manipulation of data collected from sensors, without any interaction with a real systems. Technologies such as Wireless Sensor Network (WSN) and Data Acquisition Board (DAB) provide the ability to interface with an analog system and convert the systems signals into digital data that can be sent through the Internet. Also the wireless sensor network, as a distributed framework, provides the abilities to integrate multiple actuators and sensors that are spatially distributed and connected trough nodes of a wireless network to a gateway. The data from those sensors, namely air temperature, humidity or luminosity, can be used for academic purposes, as presented on chapter 7.2, and can be emulated through a Virtual Sensor Network (VSN), removing the need of having a real WSN installed, when the accuracy of data may not be necessary.

After creating the means to retrieve data from laboratory systems or from sensors, we still have the problem of the amount of different accesses we have to make, using diverse communication protocols and different interfaces, as we can see in figure 1.1.

In this sense this thesis aims to contribute with a platform capable of unifying different interfaces into one that is accessible through the Internet.

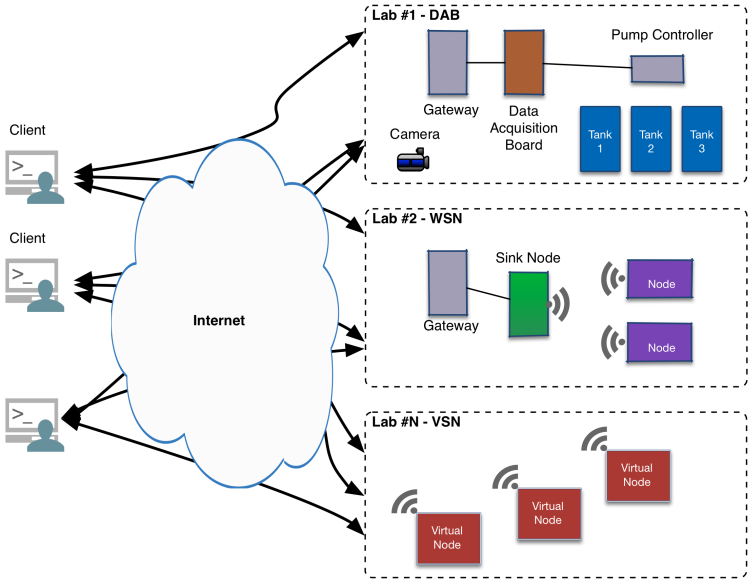


FIGURE 1.1: An example of different interfaces.

1.2 Objectives and Contributions

One of the objectives of this work is to contribute to existing knowledge regarding the remote laboratories paradigm by reactivating some of the systems available at the Laboratory of Industrial Informatics and Systems (LIIS) of the Department of Informatics Engineering (DEI) of the University of Coimbra (UC), making them remotely accessible for academic purposes and give a richer experience to courses that benefit from interaction with laboratory systems. Another objective corresponds to the development of an environment capable of unifying the access to different systems, which by nature the accesses are very distinct, and make them accessible through a remote environment using the Internet.

Based on the objectives, the main contributions of this work are:

1. The study of a set of platforms and technologies, presented on chapter 2, that enriches, improves, and allows to understand what are its main requirements.

2. On chapter 4, a set of main requirements for a platform that allows the integration of different devices and the remote access.

3. A fully detailed architecture with the explanation of the mechanisms for a platform of this sort, done on chapter 5.

4. Finally, on chapter 7, a set of applications that use the platform and help to understand, test and validate the approach and the scenarios where this platform can be applied.

It is expected that all the research, analysis and development work done on this thesis, contribute to the development of the general paradigm of remote laboratories.

1.3 Proposed Platform

With this work, we pretend to develop a platform that integrates, monitors and controls devices (e.g. WSN or DAB) and virtual environments (e.g. VSN), creating a unique way of interaction with different equipment and technologies. All the infrastructure is designed to be supported by low cost devices in a distributed environment, and accessible over the Internet, allowing the monitoring and control of remote systems. To complement this remote interaction, for each remote system is provided the possibility of having a web camera associated with.

To integrate some of the proposed devices, were developed drivers with abilities to communicate with WSN or DAB gateways. This solution was created with the intent to use specific devices such as the *National Instruments 6008* or *Crossbow TelosB*, meaning that to integrate new devices, new modules must be created. However, it is possible to created several instances of any of the integrated devices or virtual environments.

The interaction with the platform is made through a RESTful API that has endpoints to fetch recent data, fetch data from a database, authenticate a user, make a control request to a network, fetch information about a given network or fetch all the available networks.

For each solution, there are internal mechanisms that fetch and store data to a database for later usage, to alert the gateway administrator of any problem, to authenticate and to manage the control interaction. The authentication mechanism is mainly used to return a token that is considered on the control requests, to guarantee the authenticity of the user that is controlling the system. To provide an equal usage distribution of the solutions to all of the users, the control mechanism uses a First Come, First Served policy, with an established maximum threshold for individual usage time.

Any of the modules previously mentioned is configurable through a configuration file, that specifies all of the parameters that describe a network (e.g. Gateway IP).

1.4 Platform Use Cases

To test and evaluate the developed platform, the following 5 applications were created: (1) a remote laboratory for identification and control of nonlinear systems; (2) a remote laboratory for programming in Python, using a Raspberry Pi; (3) a Volunteered Geographic Information Service; (4) a remote laboratory for modeling and simulation of physiological processes; (5) and a Geographic Information System Web Platform.

The first application consists on a remote laboratory that allows carrying out remote experiments using a real laboratory system and visualize the process via a web camera. The experiments conducted on this remote laboratory included operations like monitoring systems, observing physical variables, systems identification, digital control of dynamic systems, network control systems and distributed control systems, considering remote controllers in a shared communication network. This application was used on an academic work of students from courses about identification and control of dynamical systems of a Master Degree on Electrical Engineering. The objective was to make accessible to students, a laboratory system that was in the LIIS so that the students could complete an academic work related with the intelligent control of a process, using fuzzy logic.

The second application is a remote laboratory that allows students to develop programs and run them remotely on a Raspberry Pi which has access to a WSN. The sensor nodes were used to measure air temperature, humidity and luminosity and each one was spatially distributed. This application was used on a CS2 course [7] of Informatics, Networks and Multimedia at the University of Azores, where the students had to develop a Python application that displays, for each node of the WSN, a descriptive statistical information about the data collected by sensors.

The third application is a web service that allows receiving and visualize information that was voluntarily sent by humans with their respective geographical location. It was created to be used on two sessions of the training school Human Sensor and Geographic

Information Systems for Disaster Risk Management (HSenSIG). The first session consisted on a simulacrum exercise where the volunteers contributed with information about buildings on fire, road blocks or people in danger, and this information was used by the Coimbra Civil Protection Authority to coordinate the firemen. The second session was a simulation of a real flooding in which was considered a model to control the three-tank system. Each tank, controlled via the developed platform, represented a location of the Mondego and Ceira rivers and the volunteers had to submit information warning of a possible flood. The information submitted on the second session was validated by verifying if the levels read by the sensors on the three-thank system, corresponded to situation that the user was submitting.

The fourth application consists on a web platform that allows the students to visualize and obtain data from an on-line experiment, supported by a three-tank laboratory system, that models, simulates and monitors a physiological process as the system of ingestion and excretion of a drug. It was designed to be used in courses about computational models of physiological processes and algorithms for diagnosis and self-regulation of the Master Degree on Biomedical Engineering, in a blended learning context.

Finally, the fifth application is a platform for sensor data visualization. The main functionalities are: (1) visualization of sensor measurements and sensor location on a map; (2) processing the measurements; (3) visualization of the processed results on a map. As presented on chapter 7.5, this web platform combines the data retrieval and processing functionalities of the platform, to build a visual environment that demonstrates the potentialities of a Geographic Information System (GIS).

1.5 Publications

Of all the work presented in this thesis, resulted in the following articles:

Alberto Cardoso, Vitor Sousa, Joaquim Leitão, Vitor Graveto and Paulo Gil. "*Demonstration of identification and control of nonlinear systems using a remote lab*". Proceedings of the 3rd Experiment@ International Conference – exp.at'15, pp. 97-98, June 2015;

Hélia Guerra, Alberto Cardoso, Vitor Sousa, Joaquim Leitão, Vitor Graveto and Luís Mendes Gomes. "*Demonstration of Programming in Python using a remote lab with Raspberry Pi*". Proceedings of the 3rd Experiment@ International Conference – exp.at'15, pp. 101-102, June 2015;

Alberto Cardoso, Daniel Osório, Joaquim Leitão, Vitor Sousa, Vitor Graveto and César Teixeira. "*Demonstration of modeling and simulation of physiological processes using a remote lab*". Proceedings of the 3rd Experiment@ International Conference – exp.at'15, pp. 103-104, June 2015;

Alexandra Ribeiro, Jorge Vieira, Vitor Sousa and Alberto Cardoso. "*Demonstration of GIS web-based platform for experimentation supported by geosensors in a WSN*". Proceedings of the 3rd Experiment@ International Conference – exp.at'15, pp. 137-138, June 2015;

Apart from these, it is expected the publication of other articles, at least in the journal International Journal of Online Engineering (iJOE).

1.6 Outline

The document is organized as follows: Chapter 2 presents an overview of the State of the Art in terms of remote laboratories, technologies such as WSN and a synthesis of all the addressed subjects. Chapter 3 explains all the development methodology, risk assessment and the work plan for the first and second semester. Chapter 4 contains all the requirements analysis done in order to define the necessary features to develop. Chapter 5 presents, in detail, the architecture, features and mechanisms developed. Chapter 6 presents all the functional and benchmarking tests that were done, in order to prove and guarantee the proper functioning and performance. Chapter 7 shows some applications that use the developed platform to control or monitor a remote system. On the final chapter of this thesis, Chapter 8, are presented some conclusions about the developed platform, including the accomplishments, setbacks and future work.

Chapter 2

State of the Art

2.1 Platforms for Remote Laboratories

Remote laboratories are systems based on real equipment, which allows users to perform experimental work through an Internet connection [8]. The provides the abilities of interaction in real time with a laboratory system to perform practical experiences, visualizing and analyzing the dynamic behavior of a system.

On this section, are described and analyzed two examples of remote laboratories services that contributed largely to the actual knowledge.

2.1.1 iLab

Created by Massachusetts Institute of Technology (MIT), the iLab [1] is a platform that aggregates several remote laboratories and makes them accessible through the Internet, allowing students, researchers and educators to carry out experiments from anywhere at any time. Sharing high cost or delicate equipment and minimizes the effort of the access to laboratory equipment are the main objectives of iLab. There are three categories of experiments available on this platform: (1) Batched experiments, where the entire course can be specified before the experiments begin (e.g. script); (2) Interactive experiments, where the students can monitor and control one or more aspects of the experiment; (3) Sensor experiments, where the user observes real-time data streams without influencing

the phenomena that are being measured. The remote laboratories are aggregated using the iLab Shared Architecture (ISA).

The ISA is a web service infrastructure that provides a unifying software framework that can support access to a variety of on-line laboratories, that can be globally distributed such as the users that access these remote laboratories through a single sign-on interface. This architecture is highly scalable in the sense that it minimizes the load involved on the Lab Server, decentralized in the sense that each organization manages its own users, secure by providing a mechanism for encrypted authorization and access control and compatible in the sense of it allows the interaction with commercial software like National Instruments LabView.

As we can see in figure 2.1, the iLab's architecture separates the on-line labs into three distinct modules connected by a web service. The modules are:

- A **Lab Server** that is operated by the laboratory owner and deals with the actual operation of the laboratory systems;
- The **Lab Client** that runs on the client's computer and provides the interface to operate the remote laboratory;
- The **Service Broker** which mediates the communication between the Lab Client and the Lab Server, providing storage and administrative services that are generic and can be shared by multiple laboratories within a single university.

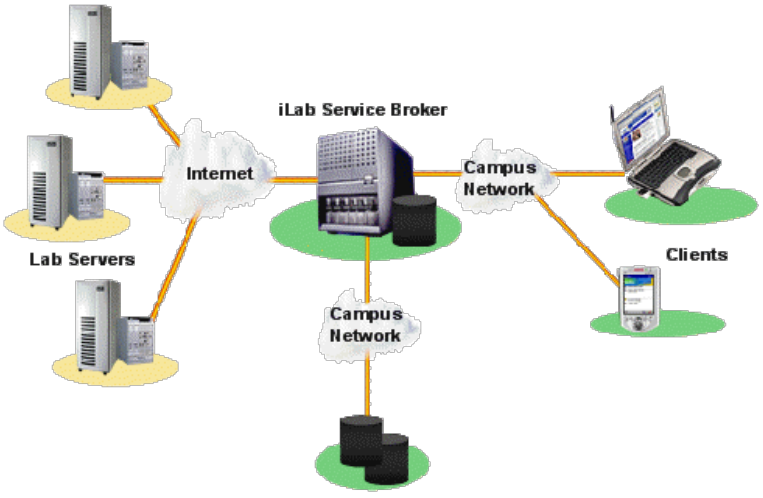


FIGURE 2.1: iLab Shared Architecture [1].

The protocol used to communicate between the servers of the ISA is the Simple Object Access Protocol (SOAP) that consists on eXtensible Markup Language (XML) information set and relies on other application layers (e.g. HTTP) for the message negotiation and transmission.

2.1.2 Go-lab

The Go-lab is a European collaborative project co-funded by the European Commission that creates an infrastructure to provide access to a set of on-line laboratories from worldwide renowned research organizations, such as European Space Agency (ESA, the Netherlands), European Organisation for Nuclear Research (CERN, Switzerland), Núcleo Interactivo de Astronomia (NUCLIO, Portugal), as well as several other universities and institutions. Similar to the previous example, the iLab, the Go-lab promotes an environment that can be used by universities, schools, instructors or students to extend regular learning activities with scientific experiments, conducted not only by teachers as a demonstration but also by students, giving them a real experience of scientific work.

To ease the integration of new labs, the Go-lab enables various levels of integration which can be chosen according to the possible resources a lab owner can invest: (1) Full integration; (2) Intermediary integration; (3) Low integration. The first level of integration consists on the standard client-server approach, implementing a Smart Device specification, that consists on a server gateway connected to the laboratory system with predefined interfaces (figure 2.2). The second level of integration consists on developing a Smart Gateway to interface with an already existing laboratory server that is connected to the laboratory system, and act as a Smart Device. The third level of integration consists on integrating an already existing client as an iFrame on the Go-lab page.

The communication between the Go-lab server and the Smart Devices and Smart Gateways is done using web sockets, enabling the server to push or pull information from the client asynchronously and only the meta-data information, plain text analytics, is fetched via HTTP GET request. As we can see in figure 2.2, there are a set of key components that interacts with the Smart Devices/Smart Gateways and provides the necessary mechanisms to control and monitor a laboratory. The authentication to use a remote laboratory is done using a token, that is generated by the booking system,

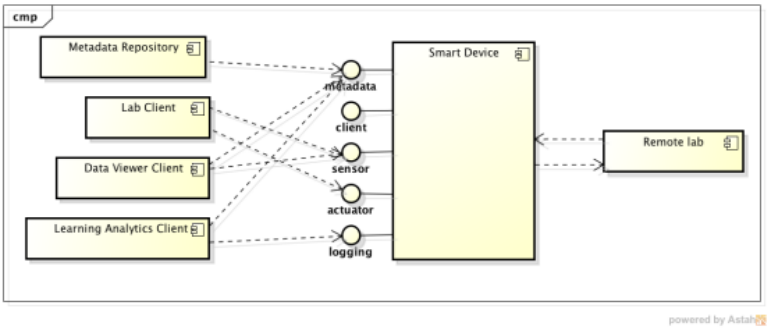


FIGURE 2.2: Go-lab UML Component for Smart Device services [2].

directly sent to the Smart Device/Smart Gateway. Those only contain logic to validate the token provided, only accepting or rejecting the authentication.

On the low integration level, the mechanisms used to interact with the remote laboratory are defined by the owner since this does not have any of the Smart Device or Smart Gateway integration.

2.2 Wireless Sensor Networks

Recent advances in wireless communications, digital electronics, and analog devices enabled the creation of sensor nodes that are low-cost and low-power to communicate untethered in short distances and collaborate as a group [9]. These spatially distributed autonomous devices can, using a set of sensors and actuators, be used to monitor and control a physical system, monitor environmental conditions such as air temperature, humidity, luminosity, etc. and cooperate to transfer data to a gateway. Due to their wireless capabilities they can be self-organized into clusters or collaborate together to complete a task that is issued by the users.

A wireless sensor is characterized by its small size, its ability to sense environmental phenomena through a set of transducers and a radio transceiver with autonomous power supply [10]. In figure 2.3 we can see an example of a WSN node which the main components are: (1) a photosynthetically active radiation sensor; (2) a total solar radiation sensor; (3) a temperature sensor (4) a humidity sensor; (5) a total of 18 expansion connectors (6) and a CC2420 radio and an internal antenna. On this example, the expansion connectors can be used as analog-to-digital (ADC) or digital-to-analog (DAC) converters to monitor or control analog systems.

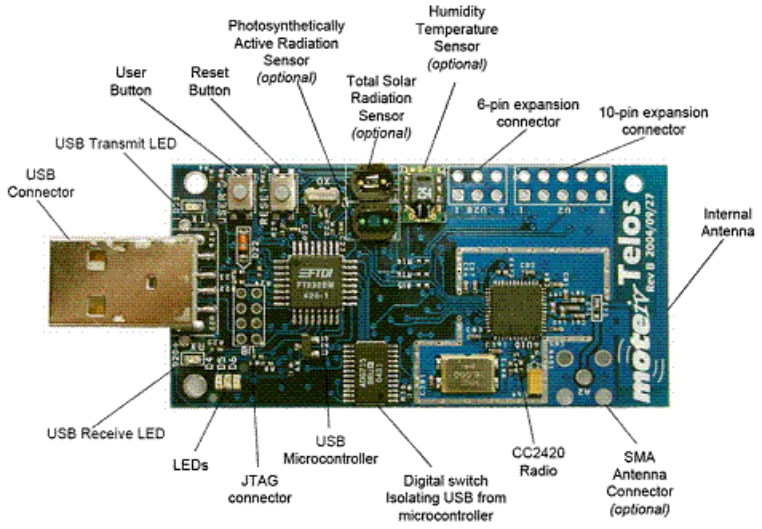


FIGURE 2.3: Example of a WSN node [3].

The low cost of each node is an important advantage of this type of networks, allowing an economical large-scale deployment of nodes. However, the low-cost together with the small dimensions, causes some limitations at levels like computing capabilities, storage capacity, communication and mainly at an energy level. An approach to overcome this problem, is for each node to use their small processing capacities to carry simple computations and transmit only the required data to a sink node that aggregates the data from all the nodes of the WSN and sends it to a remote server with higher processing capacities.

In terms of radio transceivers, currently there are mainly two types: (1) Carrier Sense Multiple Access (CSMA); (2) 802.15.4. CSMA is the cheaper and simpler radio that operates in a license free band (315/433/868/916 MHz) and has a bandwidth of 20 to 50 kilobits per second (kbps). The second type, 802.15.4, is a radio operating in 2.4 GHz band with 250 kbps of bandwidth, 5 times higher than the first type, but also more expensive. This type of radio offers the possibility of using an internal antenna making the nodes more practical, instead of having an external whip antenna.

The operating system is a software that manages all the underlying hardware components and resources of a device. The mainly used operating systems for WSN are: (1) TinyOS; (2) Contiki.

TinyOS is an open-source component-based operating system for devices with low power specifications. It has a component-based programming model, codified by the NesC language [11], a dialect of C, that allows to interact with the operating system through

three types of components: (1) Commands; (2) Events; (3) Tasks. Commands are instructions that the developer want to give to the device, such as fetching data from one of the sensors. Each Command creates a new Task and a Callback that is called when the task has ended. Tasks are non-preemptive and run in a First In First Out (FIFO). A typical application is about 15K in size, of which the base operative system is about 400 bytes; the largest application, a database-like query system, is about 64 K bytes [12]. The latest release is the 2.1.2 and brought the support for IPv6 (6lowPAN) through a module called Berkeley Low-power IP stack (BLIP).

Contiki [13] was developed by Adam Dunkels at the Swedish Institute of Computer Science in 2004 and first released in 2005. Just like the TinyOS, this is an open-source operating system that was designed to run on low specs devices, such as the nodes of a WSN, that is highly portable and event-driven with optional preemptive multi-tasking. When compared to other WSN operating systems, Contiki had the advantage of bringing the support of network protocols like the IPv6 (6lowPAN), RPL and CoAP, a lightweight threading, called protothreads, and a kernel prepared to a wide range of architectures and dynamically linked code. A typical configuration uses 2 kilobytes of RAM and 40 kilobytes of flash memory and the programming language used is C with a set of adjustments to meet the Contiki specifications. There are several projects, such as the ICT FP7 GINSENG, that uses Contiki as the main operative system for the nodes of the network, but with major adaptations. The ICT FP7 GINSENG [14] aims to create WSN that meet demanding performance specifications and that can be applied in industrial environments where a real-time operation is critical. The latest release of Contiki is the 2.7, launched at November 15, 2013.

2.3 Data Acquisition Systems

Data acquisition (DAQ) is the process of collecting signals that measure physical conditions and converting them into digital numeric values that can be manipulated by a computer. Typically the data acquisition hardware includes sensors that convert the physical parameters into electrical signals, a signal conditioning circuitry to convert sensor signals into a form that can be converted to digital values, and a ADC which converts conditioned signals into digital values. Sometimes it is necessary to amplify or filter the signal from the transducer because it may not be suitable for the DAQ hardware that

is being used. The hardware provides the necessary interface to connect to a computer (e.g. USB, Parallel, Serial, etc.) or slot cards to (S-100 bus, PCI, etc.) connect in the motherboard.

On the figure 2.4 we can see an example of a DAQ board from National Instruments, model USB-6008, that has USB connection to the computer, 8 analog inputs, and 2 analog outputs.

FIGURE 2.4: Example of a DAQ board.

Most of the boards allow two types of signaling: (1) Single-ended; (2) Differential. Single-ended signaling is the simplest method of transmitting electrical signals over wires. It resumes to two wires, one carrying the electrical signal, and the other wire is connected to a reference voltage, usually ground. The differential signaling consists on sending the electrical signal twice, one per wire. The circuit that receives the two signals responds to the electrical difference between the two electrical signals, instead of the difference between a single wire and a reference voltage.

2.4 Synthesis

At this chapter was shown an overview of the state of the art in terms of remote laboratories, wireless sensor networks and data acquisition systems, complementing the proposed approach to the creation of a platform to monitor and control remote systems.

Being this work directly related with remote laboratories, was studied two approaches in order to understand what were the main problems that they found and, if the case, understand what was the approach that they took to overcome it. The main conclusions drawn was that there is a huge problem in creating a unique way to communicate with laboratory systems and each of the examples takes an approach that is best suitable for their scenario. Fortunately, there is a huge motivation into creating a unifying mechanism, but it is still far from becoming a standard.

There are several scenarios of laboratory systems and remote laboratories. The most common scenarios are: (1) a remote process (e.g. PCS 327) identification and control; (2) Control of a remote system (e.g. three-tank system); (3) Sensor data analysis. WSN and DAB has specificities that make them adequate for the certain scenarios. For example, on the first scenario the sampling rate of approximately 1 second, provided by a WSN, may not be sufficient making the usage of a DAB more efficient on an identification and control environment. On the other hand, the DAB is cable dependent boards and do not provide the spatially distributed character, making the WSN more suitable for the third scenario.

Concluding, we propose a platform that integrates all of this devices shown on this chapter, creating a good environment to cover most of the scenarios seen in remote experimentation.

Chapter 3

Methodology

3.1 Development Methodology

At the start of the project, there was an idea of what to develop and the objective was that each week that idea would be tweaked until it was according to the thesis objectives. To match this type of iterative and incremental workflow, we choose the agile software development methodology SCRUM. It's principal aspect is the recognition that something unexpected, as a new requirement, can surge during the course of the project.

This methodology was adapted to a laboratory environment where a real client does not exist, neither does a development team. Mainly, there was two people involved on the roles: (1) The supervisor, Professor Alberto; (2) The student, Vitor. The project supervisor took the roles of product owner and scrum master and the student was the development team.

Instead of having a daily scrum, we had a weekly scrum where the scrum master and the development team met and discussed what was done on the previous sprint. After the meeting, it was planned what was necessary to be done on the next sprint, and so on until the end of the project.

3.2 Work plan

In this section are shown the activity plans for both of the semester of the internship. This was a compilation of all the features and requirements that were analyzed on each of the weekly scrum meetings.

3.2.1 First Semester

The Gantt chart of the activity plan for the first semester is described in figure 3.1. On

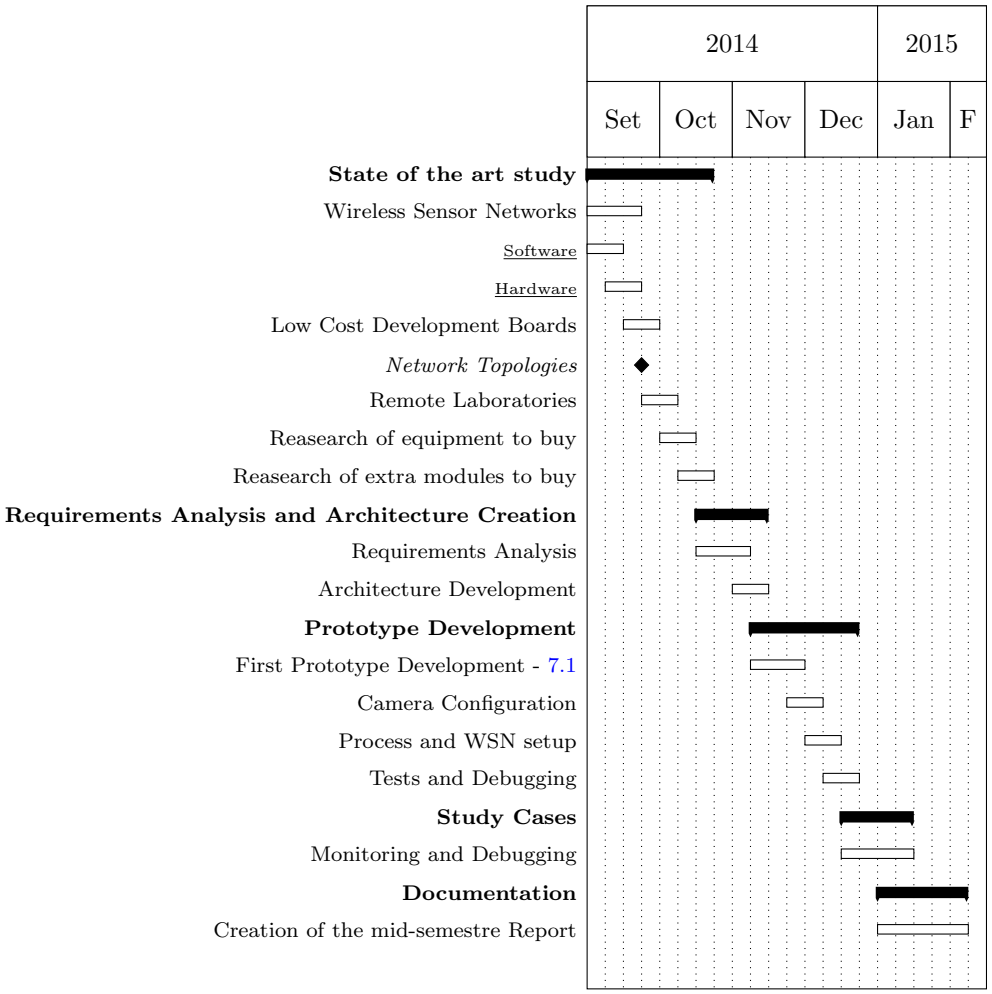


FIGURE 3.1: First semester Gantt chart.

the first semester was done a study of the state of the art to understand what technologies are adequate and work had been done, as well the creation of a first prototype that is shown at the section 7.1.

3.2.2 Second Semester

The Gantt chart of the activity plan for the second semester is described in figure 3.2. The second semester was mainly dedicated to the development of the platform. After

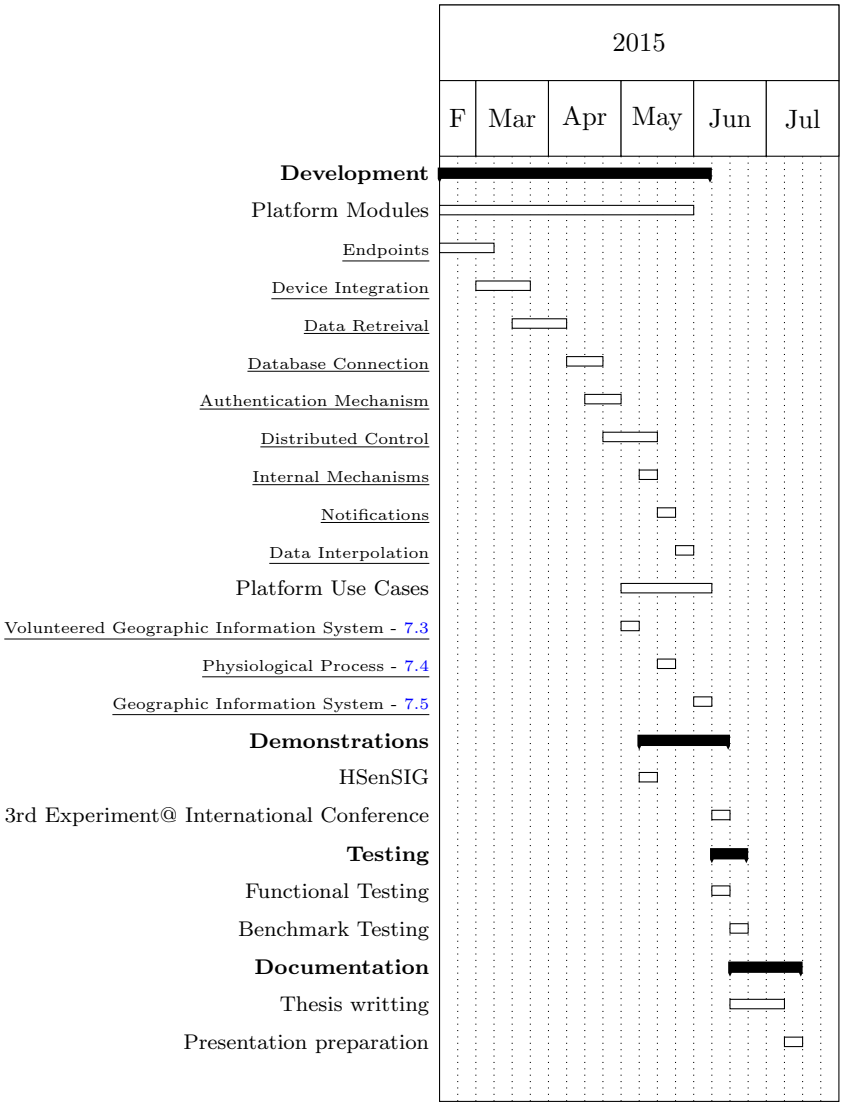


FIGURE 3.2: Second semester Gantt chart.

the platform was in a stable version, the efforts were dedicated to the development of the use cases to be presented at the summer school, Human Sensor and Geographic Information Systems for Disaster Risk Management (HSenSIG), and at the 3rd Experiment@International Conference - exp.at'15. Lastly were done the functional and benchmarking tests and written all the documentation.

3.3 Risk Assessment

At this section we discuss some risks for the project, as well as a mitigation plan for each one of them.

3.3.1 Devices Integration

Since the whole project consists on the integration of multiple devices, not completing this feature would result on a complete failure of the project. We created some Gateway Drivers that communicate with the existing devices according to their specifications (e.g. TCP sockets).

3.3.2 Heavy Data Fetch

Storing one sample every second, 24/7, results on a large quantity of data that is stored on the database. Since users can fetch data directly from the database, there is the possibility of fetching large quantities of data (e.g. 5GB) and stall the whole system. We adopted a timeout mechanism to the database queries that aborts a query if it is taking too long.

3.3.3 Platform Use Cases

Developing use cases to the platform results in less time spent in developing features and can compromise the proposed objectives. By creating a template, using existing technologies like Bootstrap or AngularJS, we are able to speed up the process of creating a web-accessible remote laboratory and mitigate this risk.

Chapter 4

Requirements Specification

4.1 Functional Requirements

Functional requirements describe the operations that a software system should be able to perform. In this section are presented and described the main functional requirements for the platform.

4.1.1 API

In table 4.1 we have the API functional requirements, that defines the necessary endpoints to monitor, authenticate and control a network.

Table 4.1: API functional requirements definition.

#	Name	Description
1	List available networks	A user can fetch a data of all the gateways available on the platform.
2	List network information	A user can fetch information about a network that is available on the platform.
3	Fetch network recent data	A user can fetch the most recent data retrieved from a specified network.
4	Fetch network device recent data	A user can fetch the most recent data retrieved from a specified network and device

#	Name	Description
5	Fetch network database data from a date	A user can fetch database data, from a network that has storage options, since a specified date.
6	Fetch network database data until a date	A user can fetch database data, from a network that has storage options, until a specified date.
7	Fetch network database data between two dates	A user can fetch database data, from a network that has storage options, between two specified dates.
8	Fetch network database data from a device	A user can fetch database data, from a network that has storage options, specifying the device.
9	Fetch network database data specifying the sampling rate	A user can fetch database data, from a network that has storage options, specifying the sampling rate.
10	Fetch network database data with interpolation	A user can fetch database data, from a network that has storage options, creating an interpolation of the values retrieved.
11	Fetch network database data with multiple options	A user can fetch database data, from a network that has storage options, specifying multiple options (e.g sampling rate and the device id)
12	Authenticate to control	Using valid credentials on a specified network, a user can authenticate himself.
13	Control a network	Using a valid token, a user can send control actions to a network.
14	Access to a network camera	A user can access to a camera from a specified network

4.1.2 Internal Server

The table 4.2 has the some of the main functional requirements to the basic functioning of the server.

TABLE 4.2: Internal Server functional requirements definition.

#	Name	Description
1	Command-line settings	The server settings are able to be modified through command line parameters
2	Settings file	The server settings are able to be modified by loading a settings file
3	WSN Integration	The server is able to monitor and control a WSN
4	DAB Integration	The server is able to monitor and control a DAB
5	VSN Integration	The server is able to monitor a VSN
6	Load networks	The server is able to load a set of networks from their respective files
7	Fetch data from network gateway	The server is able to fetch data from a network gateway
8	Store data from network gateway	The server is able to store the fetched data into a database
9	Recover from lost connection to a gateway	The server is able to recover a lost connection to a gateway (DAB or WSN)
10	Alert network administrator of failure	The server is able to alert the responsible administrator of a particular network, that a failure occurred.
11	Generate authentication token	The server is able to authenticate and generate a control token
12	First Come, First Served control	The server is able to manage the control on a First Come, First Served order

4.2 Non-functional Requirements

Non-functional requirements specify criteria that judge the operation, rather than specific behaviors of a system. These are requirements that have no visible impact on end users but improves the overall quality of the interaction experience. At this section are described two non-functional requirements, scalability, and robustness, that are considered the most important to this project, due to the expected heavy user interaction.

4.2.1 Scalability

Scalability is the ability of a system to handle a growing amount of work. There are two types of scalability: (1) Vertical; (2) Horizontal. The vertical scaling consists on adding more resources to a computer, typically involving the addition of memory, hard disk or more CPUs. The horizontal scaling consists on adding more computers to a distributed software.

Having the possibility to run more instances of the platform on different computers, results in better response times on low and high demand requests. To meet this requirement, it is necessary to create a system that is capable of having multiple instances working as one. This brings some problems: (1) In terms of data storage and data fetching, it is necessary the creation of means to differentiate the instance that does the sampling and store, from the instance that only does the sampling and does not store, surpassing the problem of having duplicated keys on the database; (2) The control policy must be done by multiple machines with different states, creating the need of a stateful control mechanism that saves the control action to a database, accessible by other machines.

In terms of scalability, the objective with this platform is to build a system that is capable of running on a standalone or distributed mode, allowing both horizontal and vertical scalability.

4.2.2 Robustness

Robustness can be defined as the ability of a software system to cope with unanticipated events, invalid inputs, corrupted stored data, and so on, during its runtime. If a software system includes interaction with hardware system, it should consider its problems too, like equipment damage or loss of power.

Independent systems, software and hardware, can generate faulty situations where there is no possibility of automatically restart or repair. For example, WSN are very propitious to faulty situations where one or all nodes of the network fail and there is no data incoming from the gateway or the sink node. On this case, the platform should handle the situation by generating values indicating that there is no connection to the WSN, such as not a number (NaN) values. When the connection from the nodes or the gateway resumes, the system should be capable of reconnecting and resuming normal operations. Other than that, the platform itself can generate some buggy situations and, itself should be able to recover from the faulty situation.

On a remote laboratory environment, sometimes the experiments are slow and take time. It is important that the system stays operational most of the time, not contributing to the user having to restart his experiment.

4.3 Synthesis

At this chapter was shown the main requirements, functional and non-functional, that are mainly considered for the development of the platform. Having a set of requirements gives a good notion of what needs to be done and how long it might take to do it.

Functional requirements helped to define what are the main functionalities of the API, explaining what are the kind of endpoints should be considered by the system, and the internal server, by specifying internal functionalities. The non-functional requirements, from a developer point-of-view, helped to understand how the architecture should be defined in order to accomplish them, as explained on each one of them.

There is a set of other requirements that are equally important but not considered on this chapter. For example, at the non-functional requirements, scenarios like backup, security or efficiency. These are as important as the requirements defined on the previous sections, but due to the small size of this work in terms of usage and development time, it would be an excessive overhead. Due to lack of time, it was impossible to exploit all the requirements in order to build a platform with greater quality.

With this set of requirements, it is possible to build a stable environment on a small scale and keep developing features improving the overall service.

Chapter 5

Platform Architecture

5.1 Overview

After analyzing the state of the art devices and platforms for remote laboratories, we tried to define an architecture capable of accomplishing what was proposed. In this chapter, each of the modules is described, demonstrating what are their main features and their connection with the other modules.

In figure 5.1 we can see an overview of the defined architecture that is composed by seven interconnected modules: (1) Authentication Mechanism (AM); (2) Gateway Handler Module (GHM); (3) Database Connector Module (DBCM); (4) RESTful API; (5) Stateful Control Mechanism (SCM); (6) Heatmap Module (HM); (7) Mail Module (MM).

Each of these modules has distinct functionalities, but some of them depend on each other to complete tasks. For example, the Stateful Control Mechanism depends on several modules like the RESTful API, the Database Connector Module, and the Gateway Handler Module.

All of the modules are separated from each other with the exception of the Database Connector Module, Stateful Control Mechanism, and Mail Module that are inside the Gateway Handler Module, but accessible from the other modules.

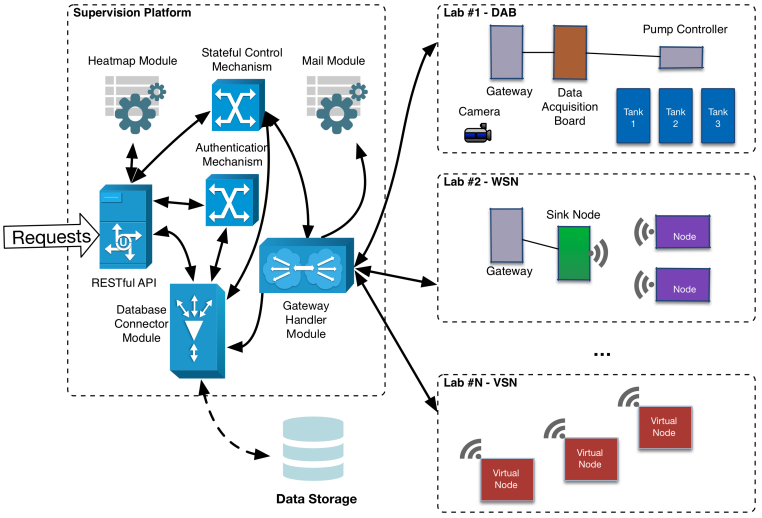


FIGURE 5.1: Supervision Platform Architecture.

On the following sections will be explained in detail the functionalities of each module and of the internal components of the platform.

5.2 Technologies Used

At this section are described the main software technologies that were used to develop the architecture of the platform.

5.2.1 Flask

Flask is a web development micro-framework for Python with a small core and easy-to-extend philosophy.

The core version of Flask is a HTTP server with a routing system. Making a request to a URL (e.g. /networks/), will result on the execution of a method on the server side and it is possible to pass parameters to the function just by adding them to the request. If the parameters added do not meet the server requirements, the request will be aborted. This routing system automatically order routes by their complexity, meaning that it is possible to declare routes in a arbitrary order and they will work as expected. The only criteria is that the declared routes must be unique or the request will redirected to a canonical URL, usually a 404 Not Found.

For each incoming request Flask creates a new thread to respond. This results on a non-blocking server allowing to respond to several requests simultaneously.

The idea of Flask is to build a good foundation for all web applications and to everything else is added by extensions.

5.2.2 Pyro

Python Remote Objects (Pyro) is a remote method invocation library, for Python, that enables building applications where objects can communicate each other over the network, with minimal programming effort. It takes care of locating the right object on the right computer and makes possible to use normal Python method calls, with almost every possible parameter and return value type. Written in Python, Pyro runs on normal Python 2.x, Python 3.x, IronPython or Pypy and runs on any operating system. It is also possible to define timeouts on network communications to prevent a call blocking forever and supports automatic reconnection to servers in case of interruptions.

Pyro works as client-server architecture, where on the server are defined the methods to be invoked and it keeps waiting for client connections on a certain port and registry. The client connecting to the server, using the same Pyro library, can invoke methods that are defined on the server side.

There is a lightweight native client library available for .NET and Java, called Pyrolite. Pyro is now on the version 4.

5.2.3 JSON

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format. It was derived from the ECMAScript Programming Language Standard. JSON defines a small set of formatting rules for the portable representation of structured data [15]. It can represent four primitive types: (1) Strings; (2) Numbers; (3) Booleans; (4) Null, and two structured types: (1) Dictionary; (2) List.

5.2.4 JWT

JSON Web Token (JWT) is a compact mean of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted [16].

5.2.5 MongoDB

MongoDB is a cross-platform, NoSQL, document-oriented database, with the purpose of being flexible and scalable [17]. The main reason why MongoDB moves away from the relation model is to make scaling out easier. A document-oriented database replaces the concept of row with a more flexible model called document, that makes it possible to represent complex hierarchical relationships with a single record. There is no predefined schema allowing to the document's keys and values from different types or sizes. Without a fixed schema, adding and removing fields as needed becomes easier. In general, this makes development faster as developers can quickly iterate and try multiple modules for the data and choose the best one to pursue.

MongoDB can also run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. These features are:

1. **Sharding**: The abilities to scale horizontally through shards of the database. A shard consists on a partition of data from the database that is held on a separate server. This is divided into ranges, based on the shard key, and distributed across multiple shards.

2. **Replication**: This consists on making two or more copies of the data sets. Each set can be either primary or secondary replica at any time. The primary replica performs all writes and reads by default, while the secondary replica will maintain a copy of the primary one. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary set should become primary.

A document-oriented structure favors JSON-like documents, making MongoDB a good solution to the integration of data in certain types of applications.

5.2.6 Synthesis

At this section were shown the main technologies that are used to support the creation of the platform.

One of the main objectives of this work is making it accessible over the Internet using a unique address. Technologies such as Flask, can create a web API capable of accepting requests (e.g. GET or POST) and start an action on the server side, like getting data from a WSN. Essentially we only need the routing mechanism from Flask and since the rest of the features only comes through extensions, we get less overhead and a more efficient RESTful server.

The simplicity of Pyro made possible to create drivers as fast and simple as possible, investing more developing time on the platform features instead on drivers to the devices (e.g. DAB) as we will see on the next section.

Using an standard data interchange format such as JSON, when compared with other data interchange format like XML, will result in a more efficient parsing and better compatibility with other programming languages. This means that it is possible to access, export or import data from/to the platform in most of the programming languages. Since we are on a remote experimentation environment, it is useful for users to have the ability to interact with the platform in different languages (e.g. Matlab or Javascript) without any restriction. With JWT we get all of these benefits and the necessary mechanisms for a stateless authentication.

Devices that does monitoring (e.g. WSN or DAB) tend to generate large quantities of data and storing that, can lead to some problems. The MongoDB will provide the necessary horizontal scalability through the features of replication and sharding and also, since it is a document-oriented database, we can store directly JSON data without any additional parsing or processing. This results on a more efficient data store and retrieval.

Concluding, we now have the necessary basis to create an Internet-accessible platform, capable of efficient and fast device driver creation and integration, with standard data interchange formats and an efficient database system.

5.3 Authentication Mechanism

The Authentication Mechanism is used to generate tokens to authenticate a user on control requests and they are only valid for the network they were generated. In figure 5.2 we can see a flowchart with the steps necessary to create a new token.

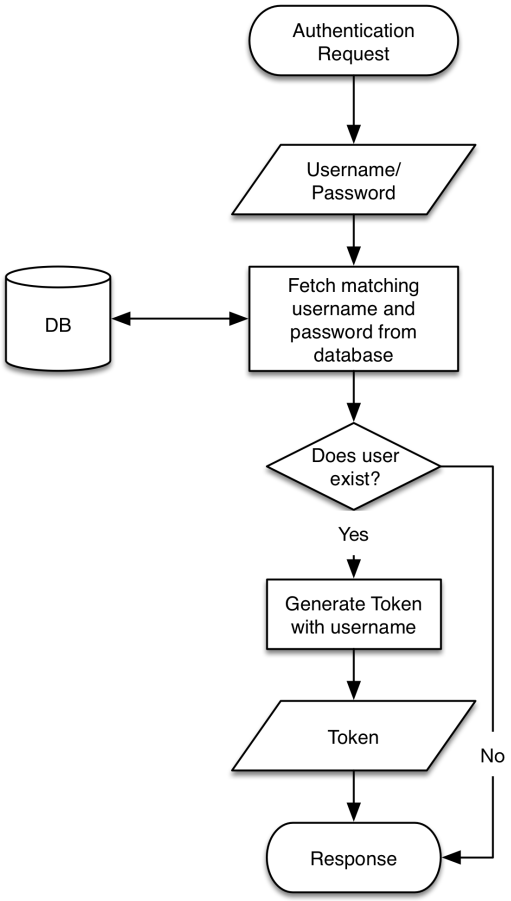


FIGURE 5.2: Authentication Mechanism Flowchart.

The process starts with a user request to create an authentication token for a network, providing a username and a password. Next it is verified if the pair, username, and password, exists on the database. If a user is returned, meaning that the credentials are valid, a JWT token is created with the username encapsulated and returned to the user.

Otherwise, the mechanism will return a response to the user alerting that the credentials provided are wrong.

5.3.1 JWT Creation

To create a JWT token, it is necessary two parameters: (1) An expiration date; (2) A secret key to encrypt the encapsulated data. The expiration date consists on a number of seconds that the token will expire in. By default, it is set to 1200 seconds, but it is configurable on the respective Network Configuration File. The encryption key is hard-coded on the platform and it is used to encrypt the encapsulated data.

Those parameters are used to instantiate a *TimedJSONWebSignatureSerializer* that creates the token by dumping the information, in this case a JSON with the username and the network (e.g. "username":"xpto", "network":"liis_wsn01).

5.3.2 JWT Validation

To validate a token it is used the same package as before but now instantiated only with the secret key and then loading the token. When loading a token, there are three possible results: (1) A JSON format data with the username; (2) A SignatureExpired exception, meaning that the token has already expired; (3) A BadSignature exception, which indicate that the token is invalid.

5.4 Gateway Handler Module

The Gateway Handler Module is the module that integrates the networks with the remaining functionalities of the platform. For each network loaded by the platform, a new instance of the GHM is created and stored on a list that is available at runtime, but is deleted when the server restarts or shuts down.

In figure 5.3 we can see the architecture of the Gateway Handler Module, of an instance that is connected to a WSN.

There are six key components on this module:

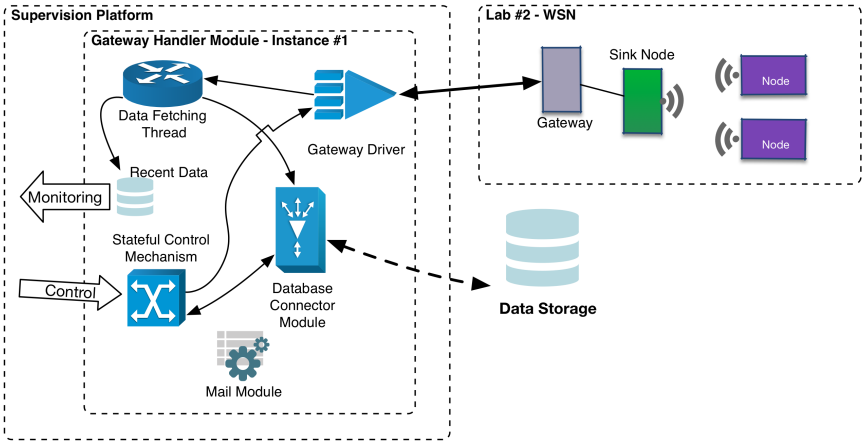


FIGURE 5.3: Gateway Handler Module architecture.

1. **Data Fetching Thread**: Access the gateway driver to fetch the data from the network devices and does the pre-fetching, the database storing and the recent data storing.

2. **Gateway Driver**: Connects to the respective network gateway to send or receive data. Each driver is made specifically to a network.

3. **Database Connector Module**: Manages the database accesses to fetch or store data. Only works when pre-fetching is active.

4. **Stateful Control Mechanism**: Manages the control interaction of a network.

5. **Actuate Method**: Method invoked to send a control signal to a gateway driver.

6. **Recent Data Storage**: Variable containing the most recent values fetched from the network. Only works with pre-fetching.

7. **Mail Module**: Manages the email sending.

At the startup procedure of the Gateway Handler Module, it is loaded a file that defines the configurations of the network this module will handle, for example, what type of Gateway Driver should be used and the configurations for the Database Connector Module. The remaining configuration parameters can be seen in section 5.7.

5.4.1 Gateway Drivers

Gateway Drivers provides a direct connection with the network gateways to send and receive data. As we can see on the figure 5.4, each driver must have two fundamental methods: (1) Process; (2) Actuate, and a connection to the gateway (e.g. TCP).

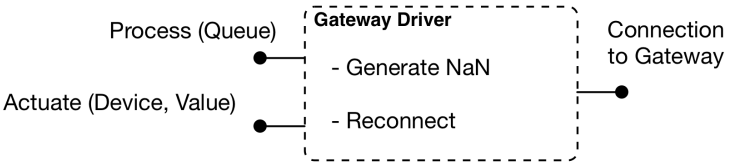


FIGURE 5.4: Gateway Driver architecture.

The first method, *Process(Queue)*, is invoked by the GHM to get data from the gateway, process and return it on a JSON format. If the pre-fetching mechanism is active it is passed a queue, with the values that were collected, to be processed and if the queue is empty the response will be filled with NaN. The second method, *Actuate(Device, Value)*, is called by the Stateful Control Mechanism and sends commands to the gateway with the device and the actuation voltage.

The connection to the gateway must be implemented by the developer along with the necessary mechanisms to restart the connection in case of internal driver failure. When the GHM cannot connect to the driver, it will try to restart the driver, as many times as defined on the network configuration file, and if the error persists it will try to send an email to the network responsible alerting the situation.

Each of the drivers has to be created according the specifications of the gateway that it is meant to be connected with. In this case, we developed three drivers for three different gateways: (1) a Contiki IPv6 Wireless Sensor Network; (2) a National Instruments 6008 DAQ board; (3) a Virtual Sensor Network.

Node Interaction Module

This driver was developed to be used with a Contiki IPv6 Wireless Sensor Network. The communication between the driver and the gateway is made via a TCP socket to send or receive a stream of bytes that represents an action (e.g. control action) or values that

were read from a device. The received values are then parsed and then converted from bytes to the voltage signal according to the sensor specifications.

Although the TCP connection guarantees packet delivery, the Contiki IPv6 WSN does not have any mechanism to guarantee that the data will be sent at the sampling rate. When the driver is trying to get data from the gateway and does not receive anything on the sampling rate period, the return message will be filled with NaN values.

The values returned for this type of network are:

- **Temperature**: Measures temperature in the air (Celsius).
- **Humidity**: Measures the humidity in the air (Percentage).
- **Photosynthetically Active Radiation (PAR)**: Measures the spectral region of the solar radiation that is visible to the human eye (Lux).
- **Total Solar Radiation (TSR)**: Measures all the spectral region of the solar radiation (Lux),
- **Battery**: Voltage of the node's battery (Volt).
- **Internal Temperature**: Internal Temperature of the Microprocessor (Celsius).
- **adcX**: The X represent the number of the ADC that the value was read (e.g. adc0) (Volt).

Board Interaction Module

This driver allows the interaction with a Pyro server that is connected to a National Instruments 6008 DAQ board. The Pyro server uses the PyDAQmx driver to communicate with the board and has two remote methods: (1) *read_all()*; (2) *execute_task(output, number of samples, value).*

The Pyro methods are remotely called by this driver in order to fetch data or actuate on the board. If any error occurs during a remote call, NaN values are returned for all of the inputs of the board.

The values returned for this type of network are:

- **aiX**: The X represents the number of the channel where the value was collected (e.g. ai1). The values are returned in Volt.

Virtual Interaction Module

The Virtual Interaction Module driver is different from the other two, previously explained, drivers. It does not connect to any server but instead it is a virtual environment that simulates a Wireless Sensor Network.

It is possible to add new nodes and assign them sensors, via the configuration file explained in section 5.7. To create a trustworthy dataset for the sensor values, it was calculated the mean and standard deviation for each hour of the day, based on approximately 3600 values that were fetched by a real Wireless Sensor Network. To calculate the return value of the virtual sensor, it is used the data generated to the hour of the server and is added or subtracted to the mean, a random value inside of the standard deviation.

On this driver, there is no option of actuation. The method exists, but it just returns an error to the client saying that the network with this driver is not controllable.

There are three available sensors:

- **Temperature**: Measures temperature in the air (Celsius).
- **Humidity**: Measures the humidity in the air (Percentage).
- **PAR**: Measures the spectral region of the solar radiation that is visible to the human eye (Lux).

5.4.2 Data Fetching Thread

Data Fetching Thread is the mechanism that collects data from a network, using the gateway driver and has two operating modes: (1) Direct; (2) Pre-fetching. The first mode consists on collecting the data from the network when there is a request for data. The second mode, pre-fetching, is done by running a thread at the sampling rate and collecting automatically the data from the gateway, and storing it on a variable, recent data, that can be accessed from the outside of the Gateway Handler Module to fulfill,

for example, */fetch/recent* requests. At the rest of the time, the thread is at a sleep state and only wakes up to collect new data.

For both modes, the fetched data is parsed and converted into a JSON format and added the timestamp of the moment it was collected. Only one of the modes works simultaneously, meaning that either the direct mode or the pre-fetching mode can be active at the same time and the pre-fetching mode is the only mode that is able to store data into the database.

5.4.3 Database Connector Module

This module manages all the connections to and from the database. For each Gateway Handler Module is instantiated a new Database Connector Module to manage all of the database accesses, reads, and writes, for that network.

As we can see in figure 5.5, this module is composed by a set of collections, a data queue, a storing thread and a MongoDB driver.

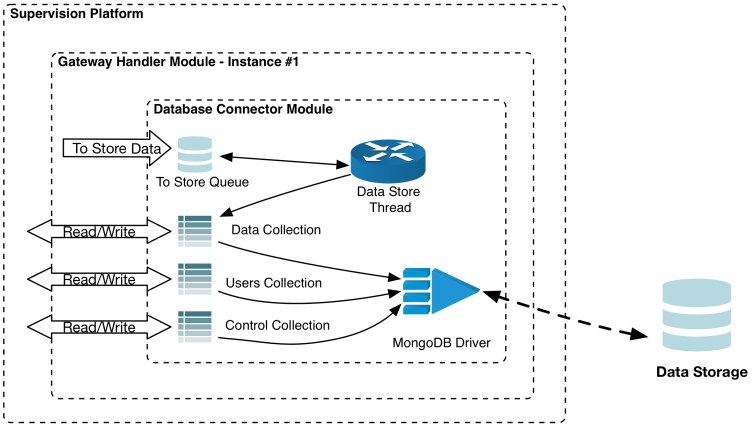


FIGURE 5.5: Database Connector Module architecture.

The users, data and control collections are the only means to have direct interaction with the database driver. At the startup of this module they are loaded from the database and stored into variables that can be later accessed, from outside of the module, to make find, insert or update operations directly to the database.

To Store, Queue is used by the Data Fetching Thread that sends sets of JSON data to be stored on the database. Using a queue to hold data while it is not saved into the

database, allows the pre-fetching thread to be non-blocking. This means that it can collect and queue data continuously even if the connection to the database is down or slow.

When there is data on the To Store Queue, the Data Store Thread awakes and starts retrieving it and storing to into the data collection, that is directly connected to the database driver, Pymongo. The data is only removed from the queue if the insert operation does not fail.

5.4.4 Stateful Control Mechanism

This mechanism manages the requests to control a network with a First Come, First Served policy. To send control actions to a network, a user must provide: (1) A JWT Token, generated by the Authentication Mechanism; (2) The network that he wants to control; (3) The device he wants to control; (4) The control value in percentage. In figure 5.6 we can see the workflow of the Stateful Control Mechanism.

The first step is to verify if the provided token is valid, using the JWT validation method from the Authentication Mechanism and if the token is not valid, a response is sent alerting the user that the token has expired or is invalid. The next step is to convert the provided value from percentage to voltage. In the control section of a network configuration file, it can be set the maximum and minimum input voltage of all its devices. Those values are used to convert from a percentage to a voltage using the equation at 5.1.

$$actuate\_voltage = \frac{(input\_value \times max\_v - min\_v)}{100} + min\_v \tag{5.1}$$

A response message is returned alerting the user that the input exceeds the limits of the system if the calculated actuation voltage is not inside or equal to the maximum/minimum specified voltage. Otherwise, the operation continues.

If all previous conditions are valid, then it is searched on the Control Collection, the record that describes the user that is actuating, not the one that has made the request. The record comes with the username, a timestamp of when the actuation period started,

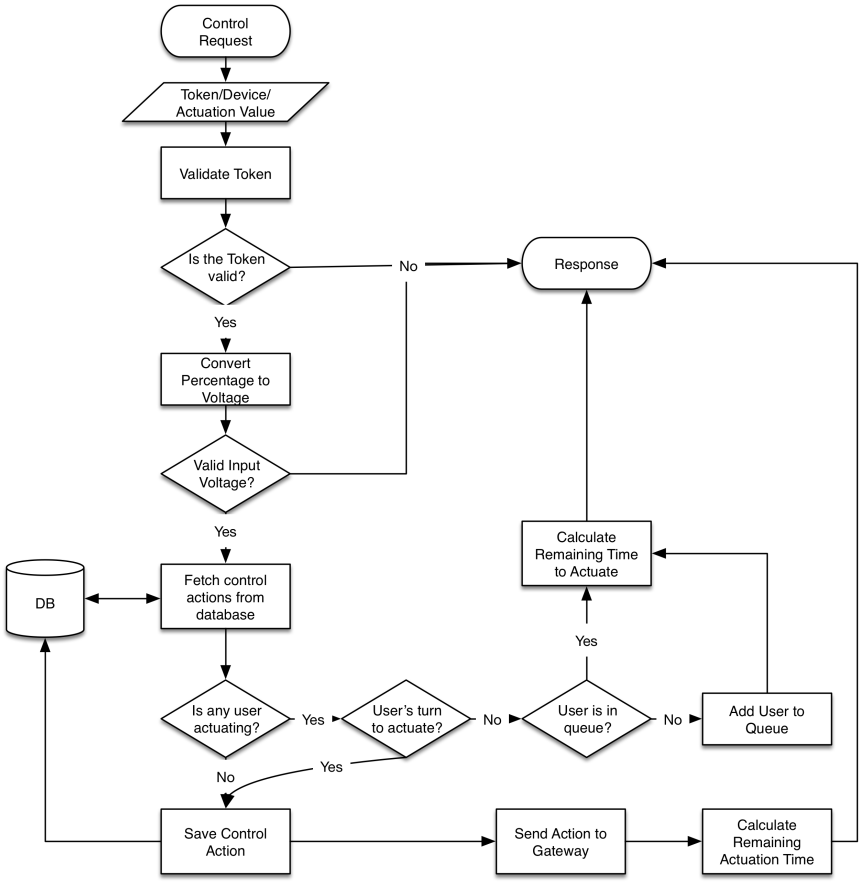


FIGURE 5.6: Stateful Control Mechanism Flowchart.

waiting flag that defines if the user is on the waiting list or actuating, and the email to alert the user that is time period of actuation has come.

If the record does not come empty and the time period of actuation is still valid, it is verified if the user that made the request is already in queue and he is added if he is not in queue. The response for both cases, if he is in queue or not, will be the time left until it is his turn to actuate.

Otherwise, meaning that the user that made the request is able to actuate, a control action is created and stored to the database. Using the converted voltage and the device that the user provided it is sent an action to the Gateway Driver method *Actuate(Device, Value)* and then sent a response to the user with the remaining time that he has to actuate.

Storing the control actions to the database is one of the reasons that makes this mechanism stateful and able to do distributed controlling, but there is still a problem. When

the actuation is made using different servers at the same time, it is possible that the clocks of the servers are out of sync. The difference can be of 1 millisecond, 1 minute or even they can be in different timezones. To overcome this problem and instead of relying on the server's internal clock, it is used an Network Time Protocol (NTP) pool to guarantee that all of the servers use the same timestamp when verifying if the user period of actuation is over, calculating all the return values and adding an action record to the database.

5.4.5 Mail Module

This module handles all the requests to send email notifications. As we can see in the figure 5.7, it is very similar to the Database Connector Module, by using the same queue and thread mechanism, allowing a non-blocking interaction. The content of each message that is stored on the To Send Queue is: (1) Subject; (2) Message Body; (3) Destination email.

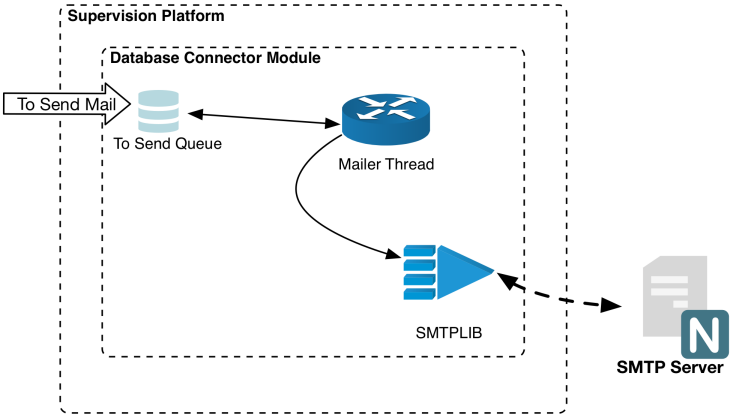


FIGURE 5.7: Mail Module architecture.

The mailer thread awakes every 30 seconds to check if there are any new messages to be sent. If there is any message, it is processed and sent via the Python SMTPLIB library to the Simple Mail Transport Protocol (SMTP) server address that is provided via the Network Configuration File. Otherwise, if there is no SMTP server address, none of the messages is sent.

This module is mainly used to send notifications about the start of the control period, to send emails with the content fetched from the database and to send error messages

to the administrator of a network. content fetched from the database and to send error messages to the administrator of a network.

5.5 RESTful API

At this section are described the main endpoints of the RESTful API and the data format of the inputs.

5.5.1 Formats

The formats of the input data are:

- <**date**>: Date's format must be YYYY-mm-dd%20HH:MM:SS. The %20 must be present in order to separate the date from the hours. It is possible to ignore values (e.g. 2015-03-16 or 2015-03-16%2011:00);
- <**samplingrate**>: Interval in seconds between each timestamp of the collected data. (Integer);
- <**deviceid**>: ID of a network device. (String);
- <**network**>: Specifies the network that you want to retrieve data from. (String);
- <**email**>: Used for large database data fetches. (String);
- <**heatmap**>: Name of the sensor to generate the data interpolation. (String);

5.5.2 Endpoints

The API is constituted by several endpoints. Each endpoint returns a response on a JSON format containing a key that is related to the endpoint (e.g. data) and three other keys that define if the request was successful or not. The main keys are:

- **status**
  - **200** - The operation was successful;

  - **400** - Bad request;
  - **401** - Unauthorized;
  - **403** - Forbidden;
  - **405** - Method not allowed;

- **success**: This parameter is a boolean that determines if the search was completed successfully or not;
- **message**: In case that the parameter success is false, this will show up with a message that explains what has gone wrong.

On the following subsections are explained the created endpoints and their respective request input parameters and response keys.

/networks

Returns a list with all the networks attached to the platform

/camera/<network>/

If this network has a camera attached, you can see the video through this endpoint. If no camera is attached, then a white screen is presented.

/info/<network>/

This endpoint returns all the information related with the provided network name. The added response keys are:

- **info**
  - **number-of-nodes** The number of devices available on the network;
  - **ids** A list with the ids of the devices present on the network;
  - **sampling-time** The period (in seconds) in which the network collects data;
  - **units** The units for each type of data returned (temperature, humidity, etc.);

- **general** General info about the network such as:
  - **admin** Email of the network administrator;
  - **master** Boolean that determines if this is a master or a slave instance of the platform;
  - **type** The network type (e.g. VSN, WSN or DAB);
  - **camera** Address of the camera. Null if no camera is attached;

/authenticate/<network>/

This endpoints authenticates and returns a token to control a network. It must receive two parameters in a JSON format:

```
{
"username":"XPTO",
"password":"1234"
}
```

The added response key is:

- **token** Token that allows user to control a network.

/control/<network>/<token>/<deviceid>/<value>/

The added response keys are:

- **time_remaining** The time left to actuate or time left to the turn of the user.

/fetch/recent/<network>/

This endpoint returns the data that was most recently collected by the provided network. The optional parameter is:

- **deviceid** ID of the device where data was collected.

The added response keys are:

- **data** A dictionary, where the keys are the device ids, with data collected from each device. The IDS are specific to the type of network that results came from.

- **ids** A list of devices that where data returned, was collected from.

/fetch/database/<network>/from/<date>

This endpoint returns data since the provided <date>.

The optional parameters are:

- **deviceid** ID of the device where data was collected.
- **samplingrate** Time between each sample in seconds;
- **heatmap** Interpolate data based on the sensor provided from this parameter;
- **email** Email to send large query results.

The added response keys are:

- **data** A dictionary, where the keys are the device ids, with data collected from each device. The IDS are specific to the type of network that results came from;
- **ids** A list of devices IDS where data was collected from;
- **heatmap** URL to the image created based on the interpolation of data.

/fetch/database/<network>/until/<date>

This endpoint returns data until a specific <date>. The optional parameters are:

- **deviceid** ID of the device where data was collected;
- **samplingrate** Time between each sample in seconds;
- **heatmap** Interpolate data based on the sensor provided from this parameter;
- **email** Email to send large query results.

The added response keys are:

- **data** A dictionary, where the keys are the device ids, with data collected from each device. The IDS are specific to the type of network that results came from;
- **ids** A list of devices that where the data returned, was collected from;
- **heatmap** URL to the image created based on the interpolation of data.

/fetch/database/<network>/between/<date1>/<date2>

This endpoint returns data between <date1> and <date2>. The optional parameters are:

- **deviceid** ID of the device where data was collected;
- **samplingrate** Time between each sample in seconds;
- **heatmap** Interpolate data based on the sensor provided from this parameter;
- **email** Email to send large query results.

The added response keys are:

- **data** A dictionary, where the keys are the device ids, with data collected from each device. The IDS are specific to the type of network that results came from:
- **ids** A list of devices that where the data returned, was collected from;
- **heatmap** URL to the image created based on the interpolation of data.

5.6 Heatmap Module

The Heatmap Module is a script that generates an image based on the interpolation of data, fetched from the database, belonging to spatially distributed devices, like WSN's or VSN's.

This module uses *gdal_grid* to create a grid based on the provided coordinates. The grid nodes are filled with values from the interpolation of the data fetched, and then the values are converted to colors using a palette of reds, creating thus an image that is

saved to the downloads folders. To prevent the accumulation of images in the downloads folder, each image is saved with a JWT token that expires 300 seconds after the image was created and is deleted by the Downloads Garbage Collector.

The response from this module is a link to download the image generated.

5.7 Network Configuration File

The Network Configuration File specifies all the parameters necessary to the mechanisms and modules of the platform. Each file is in a JSON format and represents a network and its name corresponds to the name of the file. When the server starts, all of the files, inside a specified folder, are loaded. This folder can be set via the command-line interface or by default is used, or created, one called "networks" that is inside the platform folder.

On the following list is shown and described all the supported parameters.

```
{
"info":{
        "type": The type of Gateway Driver. (WSN, DAB, VSN)
        "master": Defines if this is a Master or a Slave
        Gateway Handler Module (Boolean),
        "camera": Link to the remote camera (URL),
        "admin": Email address of the network administrator
},
"connections" :{ Defines the configurations for all the connections
        "dispatcher": { Connections settings of the Gateway Driver
                "version": Version of the Gateway (IPv6, NI-6008, ...),
                "ip": IP address of the gateway,
                "port": PORT of the gateway,
                "registry": "REGISTRY of the gateway,
                "connection-mode" : Connection mode (static or automatic)
        },
        "database": { Defines the configurations for the
                Database Connection Module
                "version": Version of the database that is
                being used (Mongo-3.0),
                "ip": IP address of the database,
                "port": PORT of the database,
                "database": The name of the database,
                "username" : Username credential to the
                database access (liis),
                "password": Password credential to the
                database access (liismongo)
        },
        "mail": { Defines the configurations for the Mail Module
                "from": Address from where the emails
                are sent (liis-lab.dei.uc.pt),
                "SMTP": SMTP server of the "from" email
                address ()smtp.dei.uc.pt)
        }
},
"control":{ Defines the configurations for the
        Stateful Control Mechanism
        "type": The type of controller (FCFS, Booking...),
        "expire": Token Expiration Time (1200),
        "max-v": Maximum voltage of DAC (2.5),
        "min-v": Minimum voltage of DAC(0),
```

```
42          "timeout": Control Timeout for each user (200),
43          "cancellable": Set device that is being controlled to
44          minimum voltage when the timeout has been reached,
45          "users": Location of the CSV file with the users able to control
46 },
47 "supervise":{ Defines the configurations for the Data Fetch Thread
48          "min-v":Minimum voltage of DAC(-2.5),
49          "max-v":Maximum voltage of ADC (2.5),
50          "nn": Number of nodes. Only works with the WSN,
51          "ts": Sampling rate for the pre-fetching,
52          "max-retries": Number of retries before alerting the administrator,
53          "nodes": Defines the nodes for the VSN. (e.g.
54          [{"temperature":1 , "humidity":10, "par":2000},
55          {"temperature":7 , "humidity":15, "par":1000},
56          {"temperature":2.5, "humidity":5, "par":32}])
57          "pre-fetch": Defines if the pre-fetching is active or not (Boolean),
58          "db-store": Defines if the database storage is active or not (Boolean),
59          "storage": The collection where the data fetched is stored.
60 }
61 }
```

As we can see in figure 5.8, the server will load every file and start a Gateway Handler Module for each one, passing the configurations in order to start all of the mechanisms and modules configured.

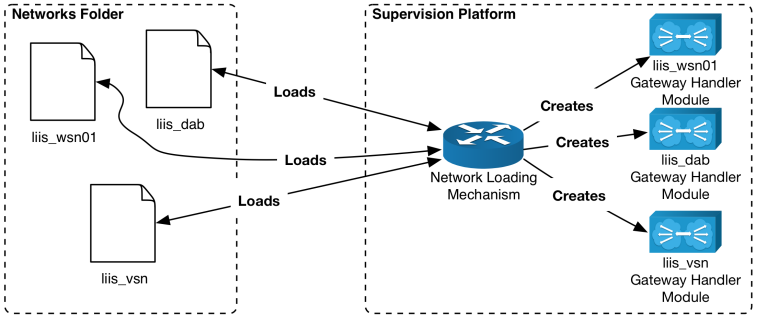


FIGURE 5.8: Network Loading Mechanism.

After the configurations are loaded, the handler is stored on a dictionary, allowing it to be accessed by the RESTful API, at any time.

5.8 Camera

Directly, the platform does not provide any support to stream audio or video from a camera. Instead, it is possible to define a URL to a camera that has those abilities and the user accesses directly to it thus removing the huge overhead of transmitting live video and audio.

If the URL provided, on the Network Configuration File, produces snapshots of the camera live video, there is the possibility to the */camera* endpoint of a network and see

a feed with one image per second, as we can see in figure 5.9. Otherwise, the user has to access directly to the camera feed.

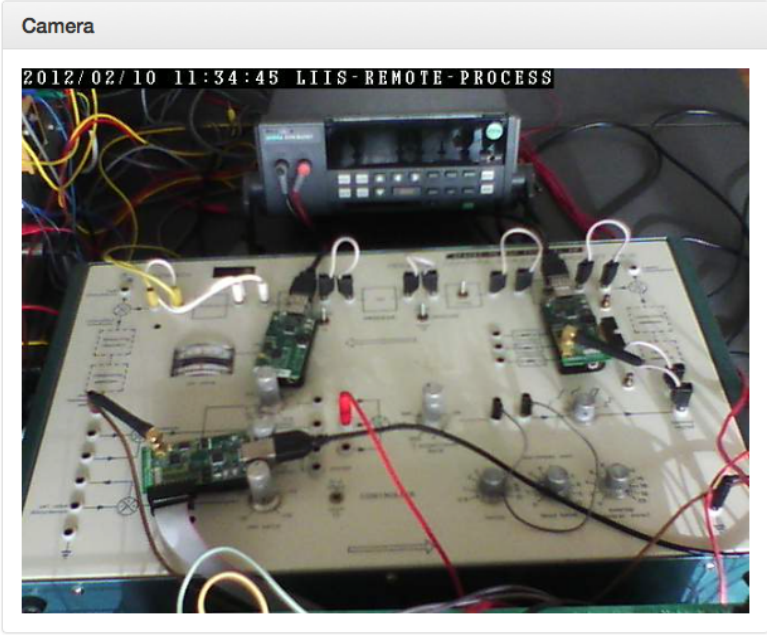


FIGURE 5.9: Snapshot of a camera feed via the platform

5.9 Internal Mechanisms

At this section are described the internal mechanisms of the platform which act as a complement of the previously explained modules.

5.9.1 Command-line interface

Through the command-line interface, when starting the platform, it is possible to set some minor configurations of the server such as:

- **-h** or –**help**: Displays the usage of the command-line interface.
- **-p** or –**port**: Defines the port that the server will run on. The input must be a valid integer.
- **-t** or –**threadtimeout**: Defines the thread timeout for the large database searches. The input must be a valid integer.
- **-n** or –**networksfolder**: Defines where is the folder with the network configuration files. The input must be a valid path.
- **-c** or –**configurationfile**: Defines where is the configuration file. The input must be a valid path. This configuration file does not override any of the previous commands.

5.9.2 Exceptions and alerts

Sometimes unpredicted conditions are met or the communication with other systems fails. Besides making a log of every error that occurs, the platform can send, using the Mail Module, email alerts to the administrator of a network, alerting of an error. For example, if the connection to a gateway is lost and the handler could not reconnect on a number of retries defined at the Network Configuration File, an alert is sent to the administrator, while the system continues to reconnect.

5.9.3 Downloads Garbage Collector

This mechanism consists on a thread that wakes every hour to check if there is any file to be deleted. Files, like the heatmap, when are created the name assigned is a JWT token with an expiration time. What this thread does is to verify if there is any file on the downloads folder that has expired and delete it, if so. This prevents the server from accumulating files that are not used and only occupies hard drive space.

5.10 Horizontal Scalability

By running more instances of the platform, we will have more computational power, meaning that the responses to high and low demand requests is faster than having only one instance. To scale horizontally the platform, a set of tools and adaptations had to be used and made. In the figure 5.10 we can see the distributed architecture of the platform. The scenario presented consists on three instances of the platform that are running on different servers and connected, all of them, to the same databases and gateways.

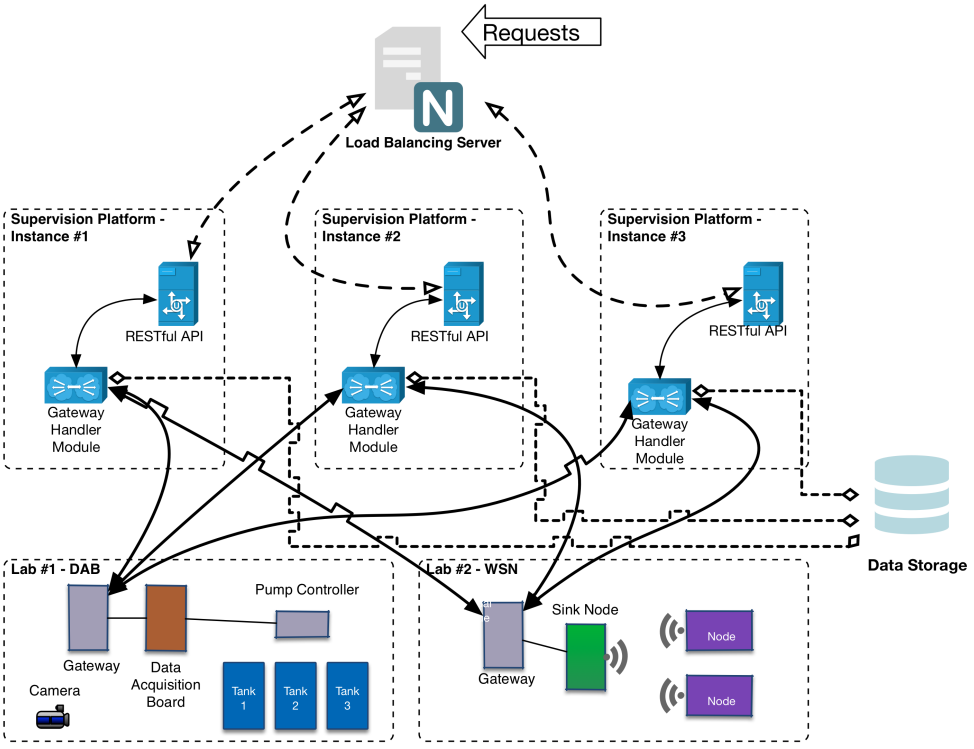


FIGURE 5.10: Horizontal Scalability architecture.

Basically this architecture consists on replications of one platform instance with minor tweaks: (1) Only one of the servers must be configured, at the Network Configuration File, as master and the rest must be slaves. This means that the master server is the only one that can store data into the database and the rest of the servers can only do pre-fetch, preventing from having duplicated data on the database; (2) Another obligatory condition is that all of the instances must be connected to the same gateways and databases, to the operations be coherent on all of the instances and to the Stateful Control Mechanism work properly; (3) To make the access to the platform instances transparent to the user, it is used a Load Balancing Server, Nginx, that distributes the requests between the available instances. This setup is done by adding the IP address of the server to a pool of servers on the Nginx configuration file.

5.11 Synthesis

At this chapter, was shown and described the architecture of the supervision platform, that allows the integration and access to Wireless Sensor Networks, Data Acquisition Boards and Virtual Sensors Networks data from one RESTful API.

The technologies shown on the first section are crucial to the architecture of the supervision and control platform. Flask provides the routing mechanism to receive and respond to requests, using a unique interface and, complementing flask with a JSON data interchange format, results on a more efficient and compatible approach. Based on JSON, the JWT tokens provide the functionalities to do stateless authenticate, resulting on fewer accesses to the MongoDB database. The simplicity of Pyro enables the creation of Gateway Drivers very quickly through remote method invocations.

Following that, it is explained the how the Authentication Mechanism works and how the JWT are created, validated and used.

The Gateway Handler Module is one of the key modules to integrate multiple gateways simultaneously by creating one handler for each of the Networking Configuration File that is loaded on the startup procedure of the platform. The Gateway Handler Module internal mechanisms, such as the Database Connector, the Stateful Control Mechanism, the Data Fetching Thread and the Gateway Drivers are instantiated for each of the networks, providing a completely separated environment for each network loaded that, in

case of failure, will not disturb the workflow of the other handlers. The Gateway Drivers handles the communication with the network gateway and has all the mechanisms to convert the data gateway specs to JSON. This data is then used by the Data Fetching Thread, directly or pre-fetched, to serve the users requests and to store it to the database. The database connection is made using the Database Connection Module that provides a queue/thread to process and store data from other modules, allowing them to do non-blocking store. There is also the users and control collections that are used by the Stateful Control Mechanism that provides the abilities to do distributed control over a network, on a First Come First Served policy, by saving each action of the control into a database.

Using a RESTful API, it provides a unique interface to do data access, control and authenticate on all of the networks.

There are other small modules that complement all of the architecture. To communicate, via email, any situation, it was created the Mail Module that can be invoked from any of the Gateway Handler Modules on the platform. The Heatmap Module has the abilities to interpolate data from the database and create an image based on the values from spatially distributed devices such as the Wireless Sensor Network.

Finally, it is explained how it is possible to scale horizontally the platform. This is done by replicating several instances of the platform and using a load balancing server to distribute the requests between all of the instances, however, there are some restrictions. Only one of the instances can store data to the database, otherwise there will be duplicated entries of data on the database.

Chapter 6

Tests and Results

6.1 Functional Tests

The objective of software quality assurance (QA) is to assure sufficient planning, reporting, and control to affect the development of software products which meet their contractual requirements [18]. Functional testing is a quality assurance (QA) process that bases its tests cases on the specifications of a software that is being tested and consists on feeding inputs and checking whether the outputs are equal or not to an expected result. Since this is a black box type of testing, the internals of the software that is being tested are rarely considered, such as on the following cases.

6.1.1 Tools Used

To perform and monitor the tests, it was used some tools that make the functional testing process, as on this case, easier and efficient.

The tool PAW was used to make all of the API functional testings and the command line was used always to monitor the logs from the platform.

PAW

PAW is an HTTP request builder that enables the quick creation of HTTP requests with headers, URL parameters, JSON form URL-encoded or a multi-part body. It is

also possible to get, for each request made, a response in various formats (e.g. JSON) and save all of the requests for later usage.

On the figure 6.1 we can see a screen-shot of PAW with a request to fetch all of the networks from the platform. The response then appears in JSON format, as expected.

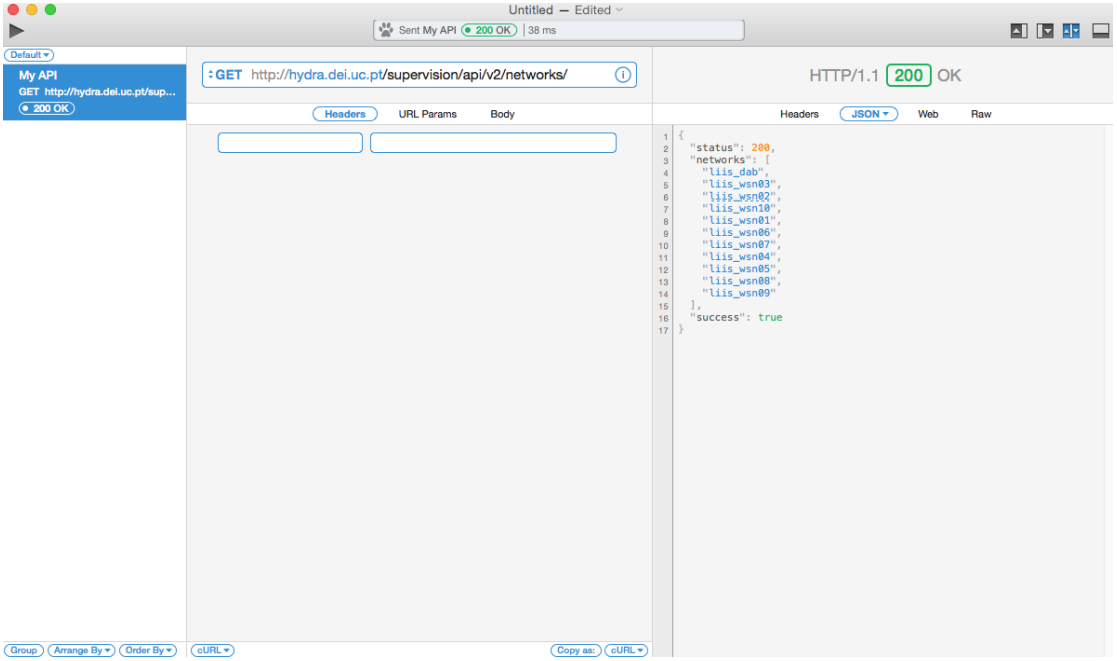


FIGURE 6.1: Screen-shot of PAW.

This tool is only available for a Mac OS X operating system and has a free trial of 30 days and each license costs $29.99.

Command line

The command line was used to access via SSH the server that was running the platform in order to monitor the logs. The command *'tail -f out.log'* shows the last logs that were written to the file, as in the figure 6.2.

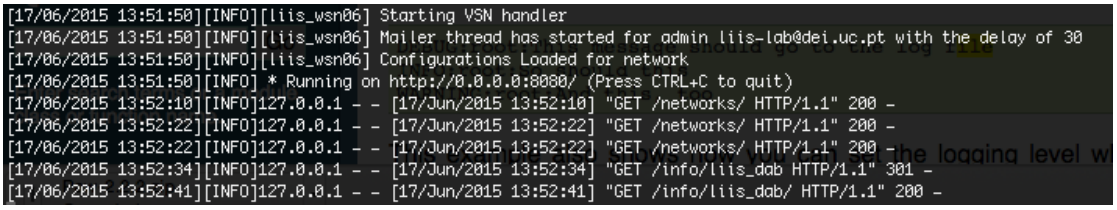


FIGURE 6.2: Screen-shot of OS X command line.

6.1.2 Tests Definition

For each test it was defined the test condition, providing a input if necessary, and the expected result from the platform.

API Tests

At the Appendix A.1 are shown the set of tests that ensure the proper implementation and functioning of the API. It covers most of the endpoints and they were tested with and without all of their parameters. On the expected results row are the keys and values of a JSON format based dictionary that is expected to be returned from the endpoint at the given condition.

Back-end Tests

The set of tests shows on the table A.2 ensure the proper functioning and implementation of the main internal functions. From the 1st to the 8th test the server was rebooted each time in order to load the new configurations. The 9th and 10th test were made by forcing the restart of the dispatcher that was connected to the Wireless Sensor Network.

6.1.3 Tests Execution

Some of these tests were done as the platform was being developed, in order to detect and fix some bugs, but they were all revised as this document was being written.

In order to simulate a real-time situation, the tests were done on a production version of the platform that was being used at the same time by other projects.

All of the API requests were made using PAW (described at 6.1.1) via an external network different from the server where the platform was running. The rest of the tests were made through the terminal.

As the tests were being executed the server logs were being monitored to detect any error or exception, meaning that the test had not been successfully completed.

6.1.4 Results and analysis

This section shows and discusses the results of the previously defined tests. The results are available at the Appendix B and the main objectives of this tests were to:

- Test and prove the proper functioning of the API
- Test and prove the correct operation of the platform internals
- Test and validate all of the main requirements.

As shown on the results, all of the executed tests have a passed status. By using *agile development* environment, every week new features had to be implemented and after the development of the feature, a functional test was made to guarantee the proper functioning. When the functional test had a "pass", the feature was done and ready for the production mode.

Several users detected bugs and reported them via email, with a set of steps to reproduce the error and if the functional test for that situation existed, they re-ran to verify if the pass status had changed. Otherwise, the set of steps were reproduced in order to detect the bug, fix it and create a new functional test to guarantee that the problem would not appear anymore.

Almost everyone has experienced a project that was declared "done" and then continued for weeks or months afterward [19]. By ensuring a "passed" on a functional test allowed to prove that a feature was done and ready for usage.

6.2 Benchmarking Tests

The basic principle of benchmarking is to collect a measure for a number of systems and compare the measured value of the system under assessment to these values. This allows us to decide if the system is better, equally good, or worse compared to the benchmarking base [20].

On this section is benchmarked and analyzed the performance of the platform with different workloads. The comparison is made between two operating modes of the platform, the standalone which consists on having only one instance of the platform, and

the distributed that has 2 instances running on different servers with a load balancer to distribute the workload between those.

6.2.1 Tools used

In order to ease the benchmarking process, there are tools to generate workload to a RESTful web service or to do a constant monitoring of the systems hardware resources. These tools generate enough data that allows to make some conclusions and analysis of the platform's performance.

Bench-Rest

Bench-rest is a Node.js client module for load testing or benchmarking RESTful API [21]. This tool can be used via the command-line or using Javascript. The parameters that can be defined are the number of iterations (-n) and the amount of concurrent operations or clients (-c). As an example, with 1000 iterations and 50 clients, the output for this test would be:

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent
    /liis_wsn01

errors:  0
stats:  { totalElapsed: 6456.860515996814,
main: { meter:
{ mean: 155.44475690949898,
count: 1000,
currentRate: 213.32138588726016,
'1MinuteRate': 11.993337805601524,
'5MinuteRate': 2.479281926757393,
'15MinuteRate': 0.831022799265474 },
histogram:
{ min: 271.00808000564575,
max: 400.7492479979992,
sum: 315039.55147928,
variance: 236.4249771854178,
mean: 315.03955147928,
stddev: 15.376117103658446,
count: 1000,
median: 315.31109300255775,
p75: 323.02037300914526,
p95: 335.63698898777363,
p99: 368.85343126177787,
p999: 400.74374668300163
}
}
}
```

The main keys of the output dictionary are:

- **errors**: Number of errors that occurred.
- **stats.totalElapsed**: The elapsed time for the entire run, including all the setup. (milliseconds)
- **stats.main.meter.mean**: Average number iterations per second
- **stats.main.meter.count**: Number of iterations completed
- **stats.main.meter.currentRate**: Number of iterations per second at this moment. This is manly useful to monitor progress.
- **stats.main.meter.1MinuteRate**: Number of iterations per second for the last minute. This is only relevant if more than one minute has passed.
- **stats.main.histogram.min**: The minimum time any iteration took (milliseconds)
- **stats.main.histogram.max**: The maximum time any iteration took (milliseconds)
- **stats.main.histogram.mean**: The average time any iteration took (milliseconds)
- **stats.main.histogram.p75**: The amount of time that 95% of all iterations completed within (milliseconds)

htop

The htop is a native tool for the operating systems based on the UNIX to do system monitoring and process view.

As we can see on the figure 6.3, it shows a frequently updated list of the processes that are running on a computer, with the respective CPU, memory, and swap usage. The list can be ordered by any of the parameters available. For the benchmarking tests, this tool is helpful to monitor the memory and CPU usage of the process that is being tested.

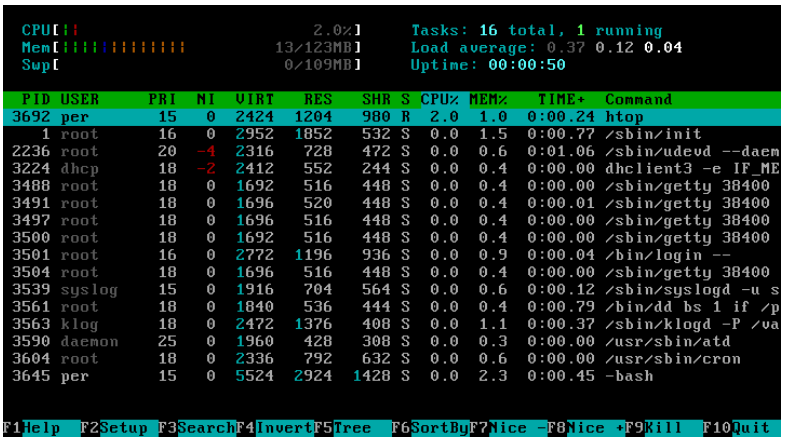


FIGURE 6.3: Screen-shot of htop.

6.2.2 Experimental Setup

This section shows and describes all the related aspects that make the benchmarking tests possible.

Platform Instance 1

The first instance is running on a machine with these hardware specifications:

- **CPU**: 4x Intel(R) Core(TM)2 CPU Q6600 @ 2.40GHz
- **RAM**: 4134 MB
- **Operating System**: Ubuntu 15.04 LTS

Platform Instance 2

The second instance is running on a machine with these hardware specifications:

- **CPU**: 8x Intel(R) Core(TM) i7 CPU 950 @ 3.07GHz
- **RAM**: 6102 MB
- **Operating System**: Ubuntu 14.04.2 LTS

Tests Definition

At the table 6.1 are shown the different configurations used during the benchmarking tests. The tests are focused on varying the number of requests and see how the response times vary and then conclude if the distributed version has better performance in terms of response times, since there is a bigger number of instances processing the requests. On every test, the number of concurrent operations was 50.

TABLE 6.1: Parameters of the benchmarking tests.

Workload Type	**Number of Requests**	**Endpoint**
Small	1000	/fetch/recent/
Small	5000	/fetch/recent/
Small	10000	/fetch/recent/
Moderate	1000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=3600
Moderate	5000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=3600
Moderate	10000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=3600
Heavy	1000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=600
Heavy	5000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=600
Heavy	10000	/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/ ?samplingrate=600

6.2.3 Standalone Scenario

This scenario consists on a standalone instance of the platform serving all the requests, as we can see on the figure 6.4.

FIGURE 6.4: Experimental setup for the standalone scenario.

Only the first instance is used and there is no load balancer to redirect the requests. It is expected higher response times on all workload conditions.

6.2.4 Distributed Scenario

As we can see on the figure 6.5, there is two instances of the platform running simultaneously. The load is being distributed by an apache web server with a 50/50 criteria.

By adding a second instance with higher hardware specifications, it is expected to achieve lower response times than the standalone benchmarking.

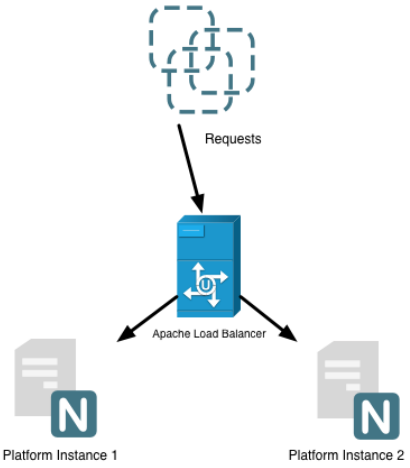


FIGURE 6.5: Experimental Setup for the distributed scenario.

6.2.5 Results and analysis

On this section are presented and analyzed the results of the tests made on both scenarios in order to verify if there was an increase of performance by using a distributed platform scenario versus a standalone platform scenario. The results can be found in Appendix C.

Figure 6.6 shows the total execution times of each test on a standalone scenario. For the smaller workload tests, the execution times were lower when compared to the other tests on the same scenario. As for the moderate workload type, the results suggest that at higher number of requests, the execution times start to escalate quickly in relation to the lower number of requests tests. Probably this is related to the number of errors that occurred since for each error the client and the server have to wait until the timeout to guarantee there was no response. As for the heavy workload type tests, we can see that, as expected, the overall execution times were higher than the other tests. However, it was not possible to complete the test with 10000 requests from the heavy workload type tests. The test generated such a high amount of data, approximately 500 MB on each of the 50 concurrent requests, that the first instance wasn't able to handle all the process of request, process and respond.

Similar to the results of the standalone scenario, the distributed scenario also had the quick escalation of the execution times, when compared to the tests with a lower number of requests. The same reason applies to this, meaning that this is an overall problem

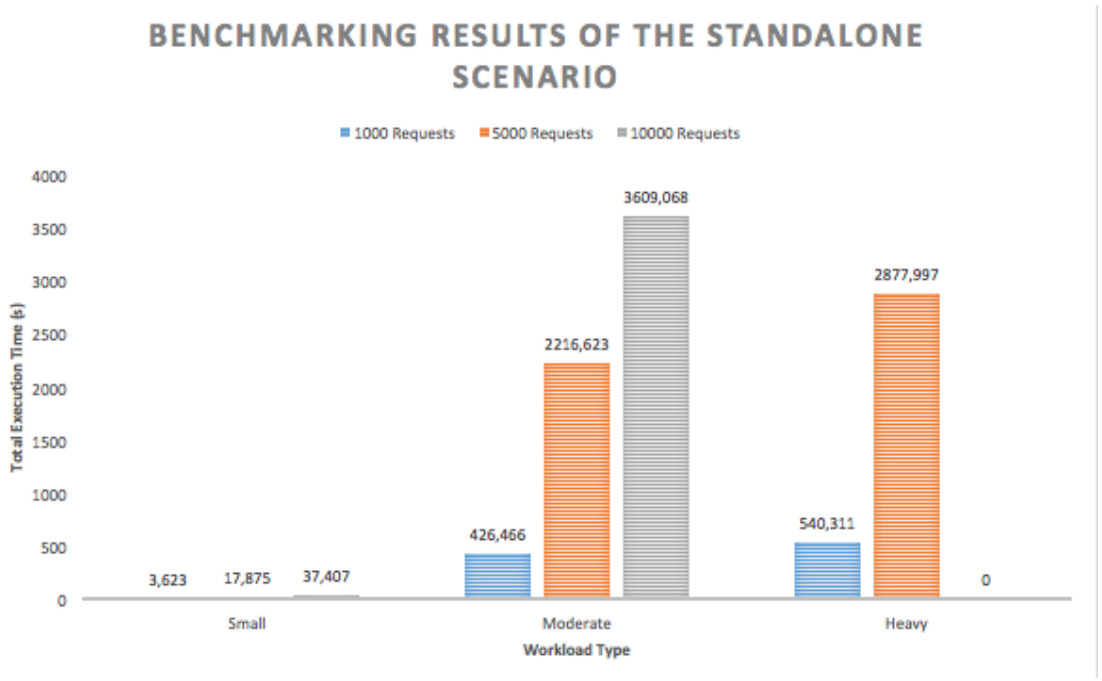


FIGURE 6.6: Benchmarking results of the standalone scenario.

of the HTTP protocol and not from the platform itself. As we can see in figure 6.7 by distributing the load of the requests between two instances of the platform, running different machines, we were now able to complete all the tests with lower execution times and, based on the results, reduce the number of errors from 7447 to 387.

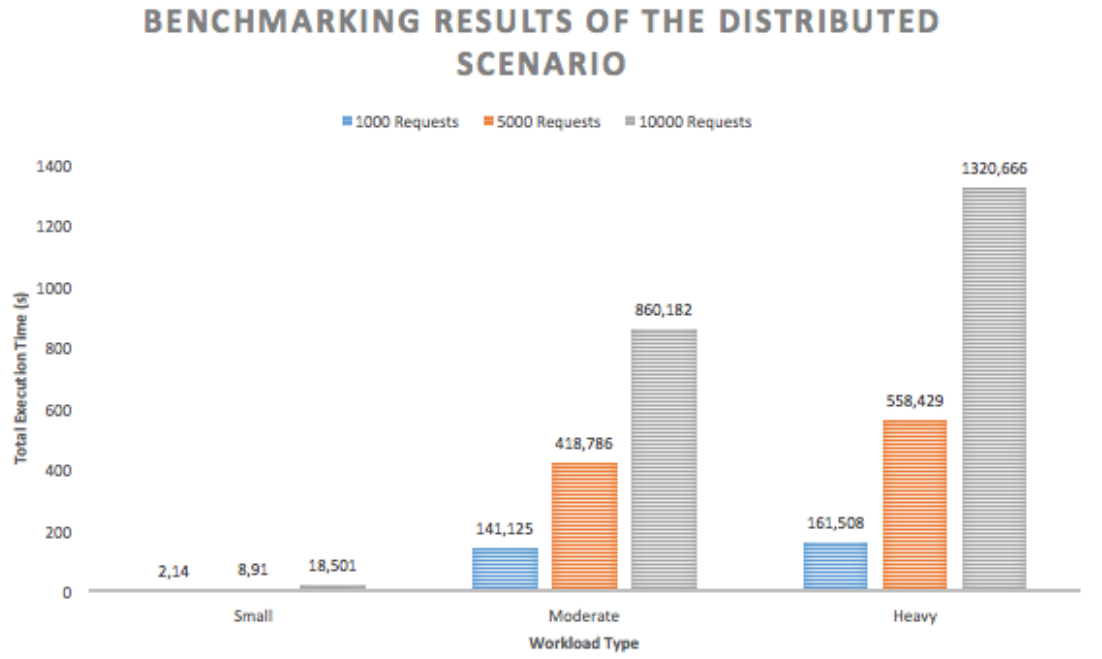


FIGURE 6.7: Benchmarking results of the distributed scenario.

Figure 6.8 shows a direct comparison between the results of the execution times of both scenarios. As we can see, the execution times from the distributed scenario are far lower than the ones from the standalone scenario. What causes the low response times is the capabilities of the distributed version to distribute the tasks between two instances of the platform. With that, if one of the instances is overloaded processing a request, the other one can still process as many requests come, but only if it is not overloaded. In fact, the distributed version can respond 2 times faster for the small workload type, 4 times for the moderate and 5 times for the heavy, having a lower number of errors per test. The main type of errors occurred were HTTP 504 (Gateway timeout) due to the fact that the server may be overloaded and could not respond to the request in timely meaning that with more platform instances fewer requests will overload fewer servers.

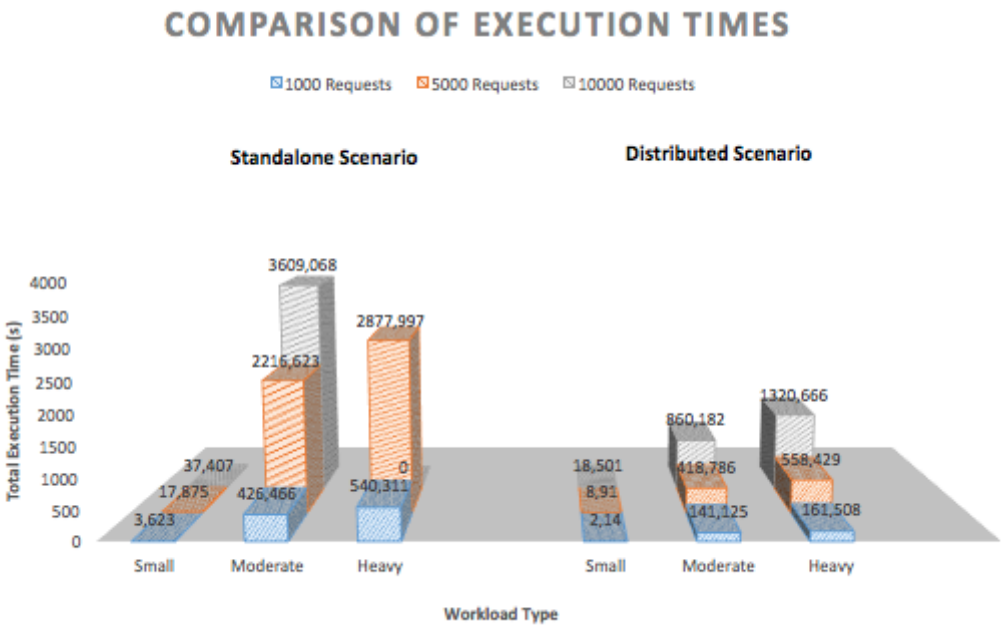


FIGURE 6.8: Comparison of the benchmarking results between the standalone and the distributed scenarios.

These results are according to what was expected because we are not only adding one more instance of the platform, but increasing twice as many processing capabilities and hardware specifications.

During the execution of the benchmarking tests, it was detected that the server could not keep the demand of 50 requests for a 1 GB data request, generated from the endpoint */fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=60*. The heavy workload type tests had to be tweaked, by increasing the sampling rate of the heavy workload type testing to 600 seconds. Also, during the execution of the heavy and moderate workload type tests on the standalone scenario, the machine could not handle all of the intensive CPU work and overheated to a point that it restarted. The tests had to be re-executed after the machine was fixed with a temporary solution.

Chapter 7

Platform Use Cases

On this chapter are described the use cases created to test, demonstrate and validate the platform. They were used on the collaboration with some institutions and demonstrated on two events: Human Sensor and Geographic Information Systems for Disaster Risk Management (HSenSIG) and the 3rd Experiment@International Conference - exp.at'15.

7.1 Remote Laboratory for identification and control of nonlinear systems

This application consists on a remote laboratory that allows carrying out remote experiments using a real laboratory system that is at the Laboratory of Industrial Informatics and Systems (LIIS) of the Department of Informatics Engineering of the University of Coimbra. The experiments include operations like monitoring systems, observing physical variables, systems identification, digital control of dynamic systems, network control systems, and distributed control systems considering remote controllers in a shared communication network.

Using a web platform to interact with the remote laboratory, students can visualize and obtain data in real time from the remote system. In general terms, the remote experiment can be used for: (1) identification of the system model; (2) control of the nonlinear system. For the first case, it is possible to send a signal to the input of the remote system, and then observe and record the resultant response of the system. In the

second situation, the remote system can be controlled considering a remote controller interacting in real time with the laboratory system, developed, for example, in Matlab.

The remote monitoring and control of the system was implemented using a WSN, where the sensor and actuator are connected through nodes of the wireless sensor network to a gateway that provides data to the main platform. All of the remote lab is based on a client-server architecture where the connection from the client to the server is done via socket and from server to the physical process is done through a WSN as in figure 7.1. This application does not have any integration with the platform for the control and monitoring of remote systems since this was a prototype version to better understand the main requirements to develop a complete version. Instead, this uses standard TCP communication between the server and the gateway of the WSN and only one gateway is supported. The communication from the front-end to the back-end, is sent plain text messages with values of actions or commands to start or shut down the supervision.

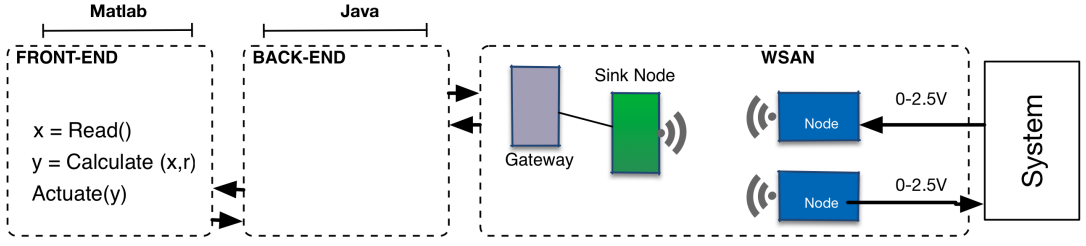


FIGURE 7.1: Architecture of the remote laboratory for identification and control of nonlinear systems.

This application surged through the need of using a laboratory system on an academic work. The objective was to make accessible to students a laboratory system, so that they could make one of the courses work related to intelligent control of a process, using fuzzy logic

After the usage of this application, and in a way to receive feedback from the users, it was given a questionnaire with 5 questions to evaluate the importance of using remote laboratories on academic works and how to improve the experience. Through figure 7.2 we can see that some of the students considers that remote experimentation is not so relevant on the context of a learning process. Complementing the results with the written feedback, it was possible to verify that the students who responded that this application was not so relevant had issues using it.

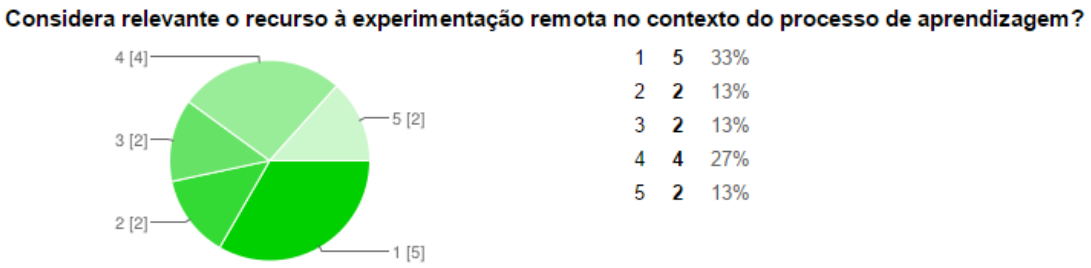


FIGURE 7.2: Answers about the importance of remote experimentation.

As for the relevancy of using a way to see what is happing on the process (figure 7.3), 60% of the students told that the visual stimulus contributed to a better experience on the interaction with this remote experimentation. As by directing our visual attention to objects of high value in a scene, we make the best use of the high-resolution space of our retina and information processing resources of our brain [22].

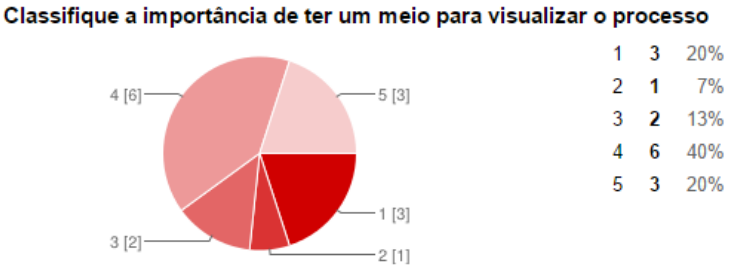


FIGURE 7.3: Answers about the importance of using a way to visualize the experiment.

Although the answers were positive, the visual stimulus of this experiment was weak. Besides the remote process and the nodes, the only thing that the users could see were the measured values from the multimeter and the LED lights from WSN nodes (figure 7.4)

This application was submitted and accepted, as a demonstration on the 3rd Experiment@International Conference.

7.2 Remote Laboratory for Programming in Python using a Raspberry Pi

This application consists on a remote laboratory that allows students to develop programs and run them remotely on a Raspberry Pi which has access to data from a WSN.

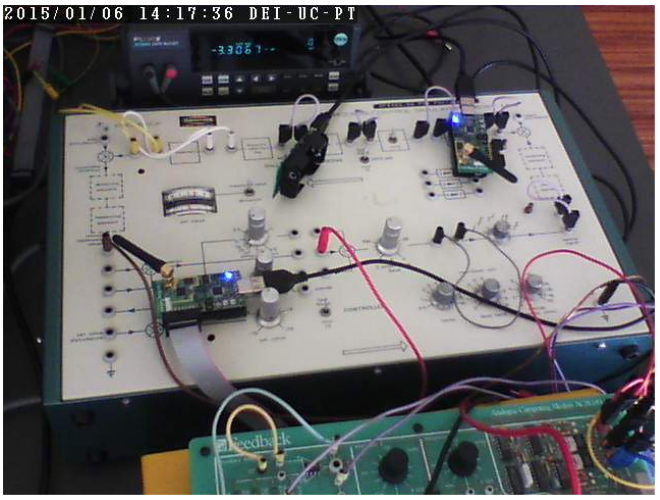


FIGURE 7.4: Camera view of the remote laboratory for identification and control of nonlinear systems.

The CS2 course [7], where this application was used, provides basic knowledge on elementary data structures and algorithms and some of the projects involve the development of programs to run remotely on a Raspberry Pi. The sensor nodes of the WSN were used to measure temperature, humidity and luminosity and each one was distributed on different locations of the LIIS. The objective was to simulate real conditions like direct sun exposure, at the interior of a house and a low light environment. The task of the students was a group project to develop a small Python application that displays, for each node of the WSN, a descriptive statistical information about the data collected by sensors. All the data and results should be saved in Comma Separated Values (CSV) files for later usage.

Concerning the process of interaction with the remote laboratory, the students had to register on a Moodle platform where was available a tutorial with information about the basics of programming in Python, guidelines to test the experimental setup and a simple program to exemplify how to perform basic operations like retrieval and processing data. A Python module with a set of functions to fetch data from the WSN was developed so that the students only had to develop the core part of the program without worrying on understanding how the interaction with the platform is made. To the access to the remote laboratory, a management system was provided by the platform, based on a First Come, First Served policy, with the establishment of a maximum threshold for individual usage time.

To carry out the project of a registered student, the procedure for testing the code involves the following steps:

1. Login to the platform;
2. Submission of a file with the program in Python;
3. Waiting for the turn to run the users code;
4. Program execution with visualization, in real time, of the results shown in a Raspberry Pi monitor;
5. Download of a file with data recorded.

After a user's program execution and if it did generate any file, it is always available through the main menu of the platform.

Regarding the architecture, some servers were used to separate the load. The front-end was developed on a Model View Controller (MVC), a framework for implementing user interfaces. It divides a given software into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user. As we can see on figure 7.5, the MVC framework was used to create all of the interface, including file submission and download, camera visualization and web socket communication to the back-end server that is running on a Raspberry Pi. Here the program syntax will be checked for errors, if it is a Python file and the user that submitted a file has already something running, all the conditions for execution are ok, the file will be queued. When available, a consumer thread will start to execute the files that are in queue and alert, in real-time through web sockets, the front-end of the execution status. If a user is not on-line when his program execution ends, an email will be sent with the output.

The module provided to the students has all the necessary mechanisms to fetch data from the endpoint */fetch/recent/* of the platform through a function *getdata()* invocation. Only the documentation of the module was provided to the students and the module is installed on the Raspberry Pi. To execute a student's program, the consumer thread does a system call for *$ python student_file.py* and since the module is installed on the system, if there is calls to *getdata()*, the module with the mechanisms to fetch data

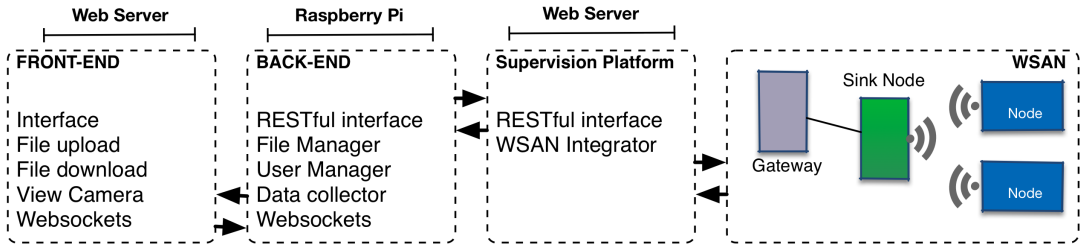


FIGURE 7.5: Architecture of the Remote Laboratory for Programming in Python.

from the platform endpoint will start to work. Besides *getdata()*, there is a *plotdata()* function, that receives a fixed format dictionary, to plot data into the *PiTFT* Raspberry Pi screen (figure 7.6).

FIGURE 7.6: Camera view of the Remote Laboratory for Programming in Python.

All of the interactions with this application were stored and counted for statistical purposes. During the group project, the total number of requests was 8933 and of those, 2370 were file uploads 520 file downloads. As we can see in figure 7.7, of the total number of uploads, there was a success rate of 72% and 28% of the submissions had errors.

There were 3 types of possible errors: (1) Syntax Error; (2) Not a Python File; (3) Execution Time Limit Exceeded. In figure 7.8 we can see that the biggest number of errors was syntax errors. Since the users could not test their scripts on their computers, the application sometimes was used as a debugger, raising this number. With 41%, the Execution Time Limit Exceeded (TLE) error was the second biggest type of error. After analyzing some of the submitted projects, and with some feedback that was received via email, it was possible to understand that this error happened because many students

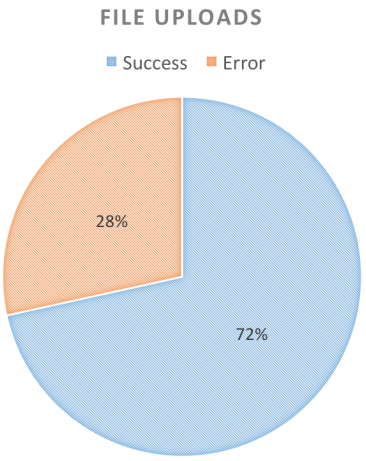


FIGURE 7.7: File upload statistics of the remote laboratory for programming in Python.

tried to create a script that collected data during various times of the day, but in only one submission. This caused the consumer thread to abort, after the timeout period, the execution of the program execution, returning a TLE error to the user. Last, with 7%, there was some users trying to submit a non-python file.

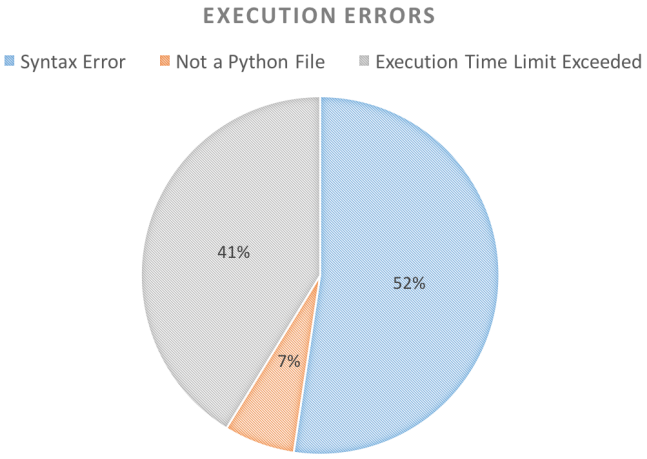


FIGURE 7.8: Execution errors statistics of the remote laboratory for programming in Python.

Comparing with the results obtained on the previous years, this application contributed to a higher percentage of students enrolled in the projects and an increase of the percentage of positive grades. To be more specific, in the previous academic year, 2013/2014, the course had a total of 23 students that completed all the evaluation processes and this academic year, 2014/2015, there was a total of 32 students. The number of registrations on all of the projects in 2013/2014 was 31,08% of 74 students and this year the numbers raised to 41,56% of 77 students. Once again, this validates the use of real and interactive

experiments contributes to a high-quality learning experience and motivates students to participate on the projects.

This application was submitted and accepted, as a demonstration on the 3rd Experiment@International Conference.

7.3 Volunteered Geographic Information Service

This application consists on a web platform that allows to receive and visualize information that was voluntarily sent by humans with their respective geographical location. It was created to be used on two sessions of the training school Human Sensor and Geographic Information Systems for Disaster Risk Management (HSenSIG). The two sessions had different objectives: (1) Simulacrum exercise by Coimbra Civil Protection Authority with Volunteered Geographic Information (VGI) collection for Damage Assessment; (2) Simulation of the integration of human and physical sensors for Disaster Management.

For the first case was built a web environment where users could submit information about an occurrence that they were seeing, including a description, number of victims, a photo, phone number, geographic location and intensity of the watched phenomena. That information was then submitted to a server and displayed on a map in real-time (figure 7.9). This map was on a computer at the command post of the Coimbra Civil Protection, and the instructions for the firemen were given based on the voluntary contributions.

The second scenario was the simulation of a flooding using the three tank laboratory system with a model that controlled the level of each tank representing three locations of the Mondego and Ceira rivers: (1) Penacova; (2) Cabouco at Ceira River, Coimbra; (3) Parque Verde, Coimbra. The first tank was only considered on the control scenario but not as a possible location for the volunteers. Similar to the first scenario, the volunteers had to submit information about the status of the river where they were if a flood was occurring. To simulate the environment, it was created a page for each location (figure 7.10) that was placed running on a Raspberry Pi with a monitor in two different locations of the Department of Informatics Engineering, where the event occurred. This simulated the real situation where the volunteer would be at the location.

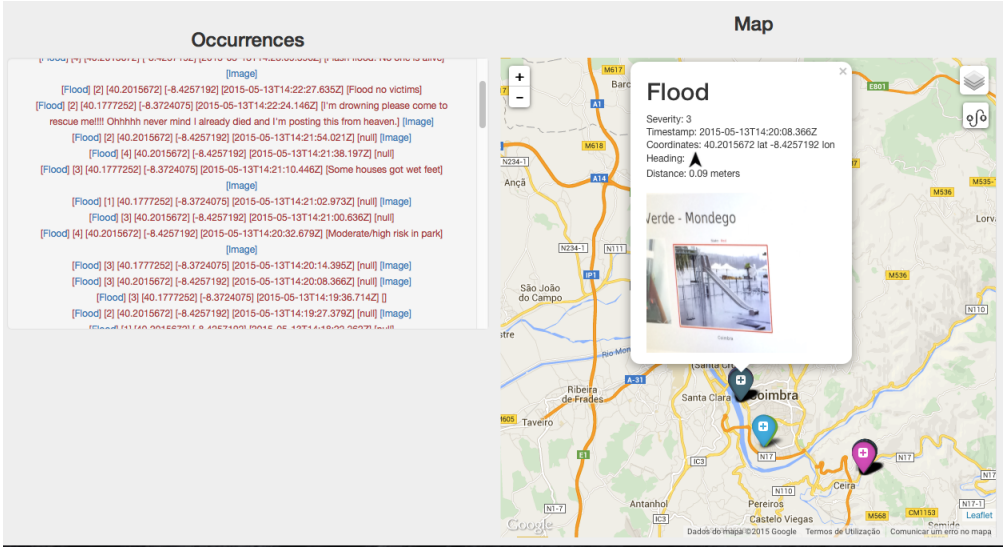


FIGURE 7.9: Visualization of the volunteered geographic information service map occurrences.

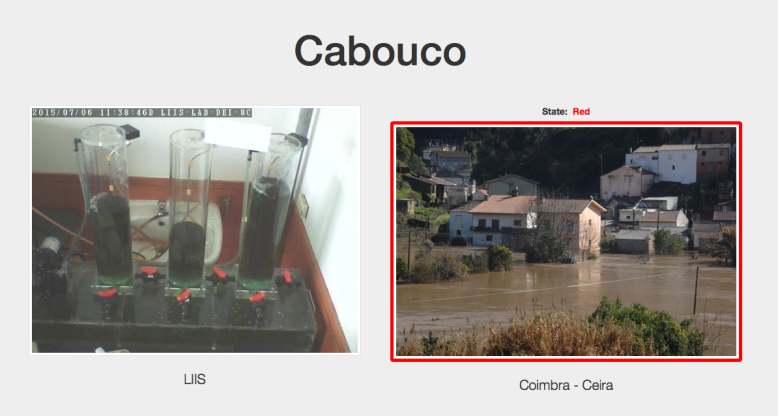


FIGURE 7.10: Volunteered Geographic Information service web camera and location visualization.

As the levels of the tanks ascend, the state presented on the web page changed, according to a system-defined threshold and then the volunteers started to contribute saying that there was a flood at their location. To simulate a real geographic position, the users had to select their location based on a list of default locations.

When the contribution is submitted it is verified the validity of the information. The verification consists, as seen in figure 7.11, on checking with the supervision platform if, according to the respective tank level, there is a flood occurring. If the submitted information is invalid it is stored as invalid and is not shown on the map of figure 7.9. All of the supervision is done through the developed platform.

Although subjective, all the received feedback was very encouraging. There is a lot

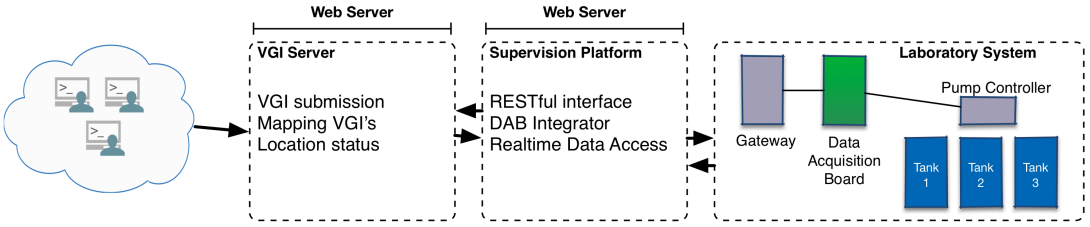


FIGURE 7.11: Volunteered Geographic Information service architecture.

of interest by the Coimbra Civil Protection Authority to develop a system like this, with some improvements like web semantics and techniques of pattern recognition for the received images from the volunteers. One of the situations that surprised the Civil Protection, was the ability to see what the volunteers were seeing through the sent images. This proved to be very efficient on recognizing locations and streets without knowing the name.

7.4 Remote Laboratory for Modeling and Simulation of Physiological Processes

This application consists on a web platform that allows users to visualize and obtain data from an on-line experiment, supported by a three-tank laboratory system, that models, simulates and monitors a physiological process as the system of ingestion and excretion of a drug. It was designed to be used in courses about computational models of physiological processes and algorithms for diagnosis and self-regulation of the Master Degree on Biomedical Engineering, in a blended learning context.

In general terms, the remote experiment can be used for the following purposes: (1) identification of the system model; (2) control of the nonlinear system. For the first case, it is possible to send a input signal to the remote system and observe and record the resultant response of the system. For the second situation, the remote system can be controlled considering a local controller with parameters defined by the user or a remote controller interacting in real time with the three-tank lab system.

In order to supervise the three-tank lab system, it is connected to a DAB. As shown in figure 7.12, we can see that the interaction between the front-end and the system is

made through the API of the platform, taking advantage of the RESTful interface to authenticate, control and monitor the system.

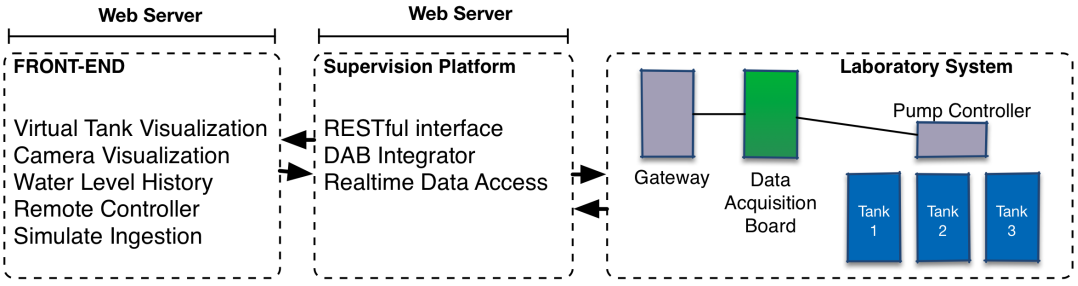


FIGURE 7.12: Architecture of the Remote Laboratory for Modeling and Simulation of Physiological processes.

To provide real-time data to the user, the front-end is fetching from the platform at a fixed sampling rate. This data is used to feed the virtual tank visualization (figure 7.13) and the graph with the history of the water level of all tanks (figure 7.14). Also, the users can observe the dynamic behavior of the system through a Web camera. To administrate the access to the remote lab, a management system is provided by the platform, based on a First Come, First Served policy, with the establishment of a maximum threshold for usage time for each individual user. All of the access management is done by the Stateful Control Mechanism.

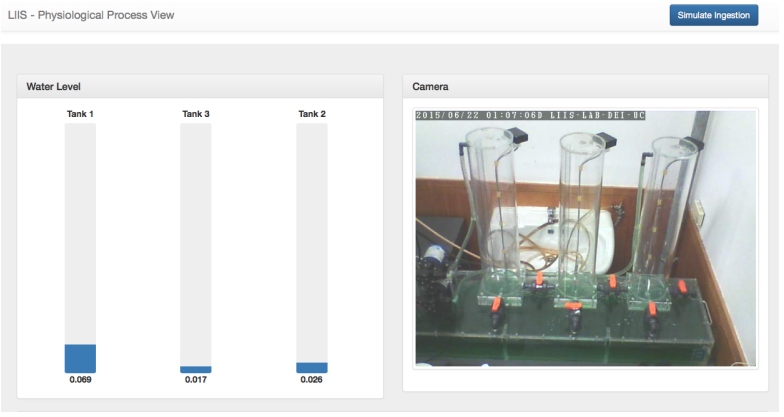


FIGURE 7.13: Virtual tank and web camera visualization.

When the user access the web platform, if he has permissions and is the first in the queue, an initial setup is made that consists on setting the water on each tank to a system defined level. After that, the user can simulate the ingestion of the drug by clicking on the ingestion button (figure 7.13). The web platform will start to apply the

model to the tank system, calculating the actuation and controlling the two available pumps on the tank 1 and 3.

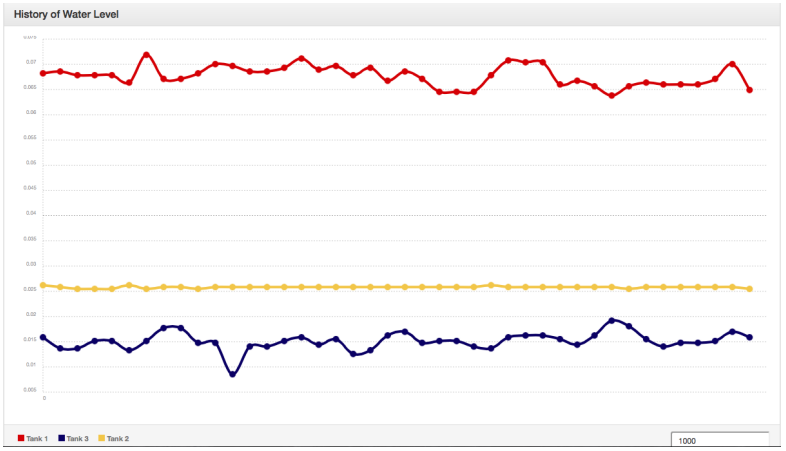


FIGURE 7.14: Water level history graph.

This application is based on a thesis of a Masters in Biomedical Engineering entitled *Modeling Physiological Systems - A computational and hybrid approach.* Also, this application was submitted and accepted, as a demonstration on the 3rd Experiment@International Conference.

7.5 Geographic Information System Web Platform

This application is a platform for sensor data visualization. The main functionalities are: (1) visualization of sensor measurements and sensor location on a map; (2) processing the measurements; (3) visualization of the processed results on a map. As shown in figure 7.15, this web platform combines the data retrieval and processing functionalities of the platform, to build a visual environment that demonstrates the main functionalities of a Geographic Information System (GIS).

Data used on this application, namely air temperature, humidity, and luminosity is being fetched by stationary wireless sensor network in a testbed that has an artificially assigned geographic location, is spatially distributed over a geographic area and is connected to the supervision platform. The endpoints used were:

- ***/fetch/info/***: Get network information for the main page

- ***/fetch/recent with device identification***: Get sensor data for each of the nodes on the map. This was also used to present data in real-time on the map or on the graph analysis
- ***/fetch/database with heatmap parameter***: Get time-series for the historic page and generate the interpolation images for the map

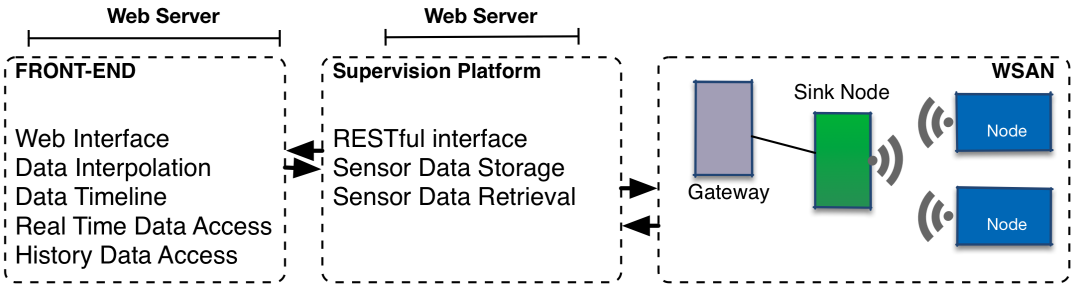


FIGURE 7.15: Geographic Information System Web Platform architecture.

This application provides, for each node, a location on a map and it is possible to view the time-series data, graphically as in figure 7.17. For each node, the user can choose the sensor from which he wants to view the time-series. The information that is displayed on the map (figure 7.16) is being fetched in real-time, from the selected node.

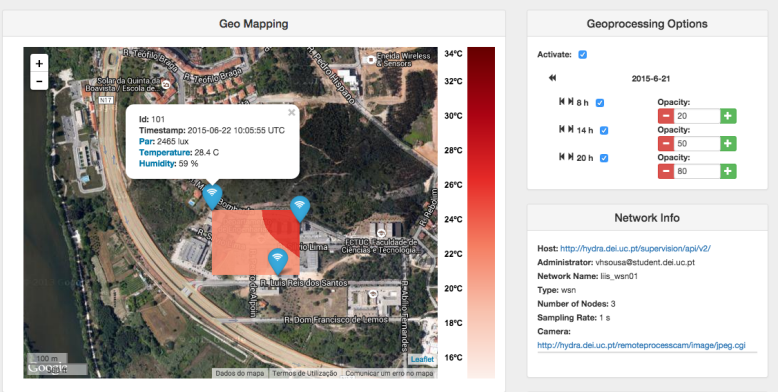


FIGURE 7.16: Interpolation surface based on the temperature data from a WSN.

One other feature is that a user can activate an option to create an interpolation surface based on temperature values from all of the nodes on the network. When this option is activated, 3 layers are created at different times of the day and it is possible to activate or deactivate any of the layers. For each layer, it is also possible to define what time of the day or change the day when the data was collected. After all the parameters are set, the output image of the interpolation surface is shown on the map.

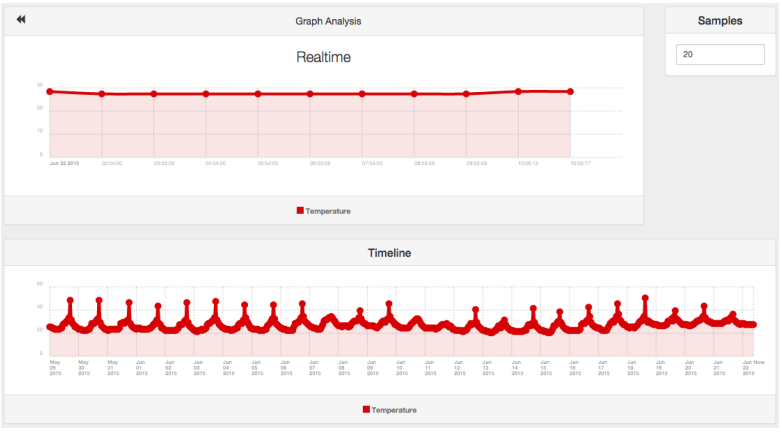


FIGURE 7.17: History and real-time data of a WSN.

For each of those variables, there was the possibility to see a timeline of the values fetched from 30 days ago and navigate through each day seeing the values with a sampling rate of 1 hour (figure 7.18).

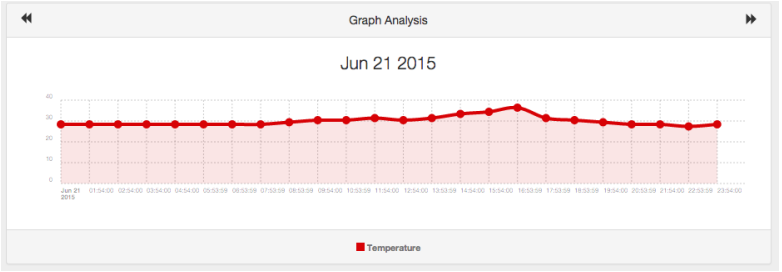


FIGURE 7.18: WSN data from a specific day.

This application was submitted and accepted, as a demonstration, on the 3rd Experiment@International Conference.

Chapter 8

Conclusions

This chapter presents the main conclusions of all the work discussed on this thesis. On the first section are described the main accomplishments achieved by done by developing this work. Next follows the setbacks and limitations of the developed platform. The third section talks about some new features that will be developed in the future. Finally, are presented some minor comments about the experience obtained during the course of the internship.

8.1 Accomplishments

With this work we created a platform that integrates, monitors and controls different technologies, such as Wireless Sensor Networks, Virtual Sensor Networks and Data Acquisition Boards, through an Internet accessible RESTful API, using low-cost devices such as the Raspberry Pi. It also includes features such as: (1) Automatic fetch and storing of data; (2) Stateless authentication via JWT token; (3) Distributed control with a First Come, First Served policy; (4) Email notifications; (5) Data interpolation.

Through the benchmarking tests, we could see on chapter 6.2, that the platform created is scalable to a higher number of users, providing an overall better quality of experience on the interaction with the remote systems.

A big achievement was the creation of a unique and transparent way of interaction with devices that have different characteristics and interfaces. There are several approaches

to the laboratory paradigm, but it is very difficult to reach consensus on the creation of a standard way to communicate with different devices through one interface. With this work, we present one approach to the problem by creating several drivers that can communicate with standard devices.

Another achievement was the rehabilitation of some devices that were inoperable due to missing controllers or complex interfaces. Now, it is possible to have a remote interaction with that equipment, making them accessible to a larger number of users.

With this work, we also managed to produce something that can contribute to higher-quality academic projects, by providing students the ability to do remote experimentation with real systems, in real-time and with fewer access restrictions.

8.2 Setbacks

The integration of different software technologies (e.g. drivers) causes a major overhead when retrieving the data from the gateways. This overhead adds latency to the requests and makes the sampling rate lower than direct access conditions, namely, outside of the platform. For example, this overhead is not noticed on a WSN since it has a relatively higher sampling rate, but on a DAB it is noticeably higher.

8.3 Future Work

There are still features to be developed that increases the potentialities of the platform and reduce the cost of implementing a gateway, such as:

- **Arduino support**: Create a Gateway driver that connects to an Arduino Board;
- **Geosensors support**: Create a Gateway driver that connects to sensors that have Geolocation capabilities;
- **Booking control policy**: Implement an alternative to the First Come, First Server control policy. The Booking control policy allows the users to choose a specific time and day to conduct an experiment;

- **Remote turn on/off**: Associate to a network gateway a relay that, remotely, can turn on or off the system.

8.4 Final Thoughts

These past few months of the internship have been very rewarding, allowing me to interact with different software and hardware technologies. It was a great joy being able to do some of the physical connections and then integrate the hardware with the software, allowing a remote interaction with something that was not designed for that.

It was also very rewarding being able to build something that can be used as the main tool in different projects, allowing students to explore and carry on experiments on systems that had restricted access, such as remote laboratory system.

Bibliography

[1] MIT. The ilab project. URL https://wikis.mit.edu/confluence/display/ILAB2/Home. Last access on July 1, 2015.

[2] Go-Lab. Specifications of the lab owner and cloud services (initial) – m21 revision. URL http://www.go-lab-project.eu/sites/default/files/files/deliverable/file/Go-Lab%20D4.1%20revision.pdf. Last access on July 1, 2015.

[3] Yacine CHALLAL. Anatomie d'un noeud capteur. URL http://moodle.utc.fr/file.php/498/SupportWeb/co/Module_RCSF_35.html. Last access on July 1, 2015.

[4] D Randy Garrison and Norman D Vaughan. *Blended learning in higher education: Framework, principles, and guidelines*. John Wiley & Sons, 2008.

[5] Watson Scott Swail. Higher education and new demographics: Questions for policy. *Change: The Magazine of Higher Learning*, 34(4):14–23, 2002.

[6] Frank Newman, Lara Couturier, and Jamie Scurry. *The future of higher education: Rhetoric, reality, and the risks of the market*. John Wiley & Sons, 2010.

[7] ACM and IEEE. *Computer Science Curriculum 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.*, 2013.

[8] Fernando Coito and Luís Brito Palma. A remote laboratory environment for blended learning. In *Proceedings of the 1st International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '08, pages 69:1–69:4, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-067-8. doi: 10.1145/1389586.1389667. URL http://doi.acm.org/10.1145/1389586.1389667.

[9] Cauligi S Raghavendra, Krishna M Sivalingam, and Taieb Znati. *Wireless sensor networks*. Springer, 2006.

[10] Paolo Baronti, Prashant Pillai, Vince WC Chook, Stefano Chessa, Alberto Gotta, and Y Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards. *Computer communications*, 30(7):1655–1695, 2007.

[11] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, 49(4):41–51, July 2014. ISSN 0362-1340. doi: 10.1145/2641638.2641652. URL http://doi.acm.org/10.1145/2641638.2641652.

[12] J. Polastre R. Szewczyk K. Whitehouse A. Woo D. Gay J. Hill M. Welsh E. Brewer P. Levis, S. Madden and D. Culler. Tinyos: An operating system for sensor networks. URL http://www.cs.berkeley.edu/~culler/papers/ai-tinyos.pdf. Last access on July 1, 2015.

[13] The contiki os. URL http://contiki.sourceforge.net/docs/2.6/. Last access on July 1, 2015.

[14] Tony O'donovan, James Brown, Felix Büsching, Alberto Cardoso, José Cecílio, Jose Do Ó, Pedro Furtado, Paulo Gil, Anja Jugel, Wolf-Bastian Pöttner, Utz Roedig, Jorge Sá Silva, Ricardo Silva, Cormac J. Sreenan, Vasos Vassiliou, Thiemo Voigt, Lars Wolf, and Zinon Zinonos. The ginseng system for wireless monitoring and control: Design and deployment experiences. *ACM Trans. Sen. Netw.*, 10(1), December 2013. ISSN 1550-4859. doi: 10.1145/2529975. URL http://doi.acm.org/10.1145/2529975.

[15] Douglas Crockford. The application/json media type for javascript object notation (json). 2006.

[16] Michael Jones, Paul Tarjan, Yaron Goland, Nat Sakimura, John Bradley, John Panzer, and Dirk Balfanz. Json web token (jwt). 2012.

[17] Kristina Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

[18] Kurt F. Fischer. Software quality assurance tools: Recent experience and future requirements. *SIGSOFT Softw. Eng. Notes*, 3(5):116–121, January 1978. ISSN 0163-5948. doi: 10.1145/953579.811110. URL http://doi.acm.org/10.1145/953579.811110.

[19] Amr Elssamadisy and Jean Whitmore. Functional testing: A pattern to follow and the smells to avoid. In *Proceedings of the 2006 Conference on Pattern Languages of Programs*, PLoP '06, pages 27:1–27:13, New York, NY, USA, 2006. ACM. ISBN 978-1-60558-372-3. doi: 10.1145/1415472.1415504. URL http://doi.acm.org/10.1145/1415472.1415504.

[20] Klaus Lochmann. A benchmarking-inspired approach to determine threshold values for metrics. *SIGSOFT Softw. Eng. Notes*, 37(6):1–8, November 2012. ISSN 0163-5948. doi: 10.1145/2382756.2382782. URL http://doi.acm.org/10.1145/2382756.2382782.

[21] Jeff Barczewski. Bench-rest, 2015. URL https://github.com/jeffbski/bench-rest. Last access on July 1, 2015.

[22] Binbin Ye, Yusuke Sugano, and Yoichi Sato. Influence of stimulus and viewing task types on a learning-based visual saliency model. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 271–274, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2751-0. URL http://doi.acm.org/10.1145/2578153.2578199.

Appendix A

Tests definition tables

A.1 API Tests

Table A.1: API functional tests definition.

#	Endpoint	Input	Expected result
1	/networks	-	**"status":**200 **"success":**true **"data":** <a list with all the network names>
2	/info/<network>/	method: **GET** network:**liis_wsn01**	**"status":**200 **"success":**true **"data":** <dictionary>
3	/fetch/recent/<network>/	method: **GET** network:**liis_wsn01**	**"status":**200 **"success":**true **"data":** <dictionary with fetched data>
4	/fetch/recent/<network>/	method: **GET** network:**liis_wsn11**	**"status":**400 **"success":**false **"message":** "Invalid Network liis_wsn11

#	Endpoint	Input	Expected result
5	/fetch/recent/<network>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** deviceid:**101**	**"status":**200 **"success":**true **"data":** <dictionary with fetched data from node 101>
6	/fetch/recent/<network>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** deviceid:**105**	**"status":**400 **"success":**false **"message":** "Bad Request. No data for this Node ID
7	/fetch/database/ <network>/ from/<date>/	method: **GET** network:**liis_wsn01** date:**2015-06-16**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-06-16>
8	/fetch/database/ <network>/ from/<date>/	method: **GET** network:**liis_wsn01** date:**2015-12-16**	**"status":**404 **"success":**false **"message":** "No results found"
9	/fetch/database/ <network>/ from/<date>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date:**2015-06-16** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-06-16 but only from node 101>
10	/fetch/database/ <network>/ from/<date>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date:**2015-06-16 14:00** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-06-16 at 14h00 with results only from node 101>

#	Endpoint	Input	Expected result
11	/fetch/database/ <network>/ from/<date>/ ?deviceid=<deviceid> &samplingrate=<sr>	method: **GET** network:**liis_wsn01** date:**2015-06-16 14:00** deviceid:**101** sr:**3600**s	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-06-16 at 14h00 with results only from node 101 and with a sampling rate of 1 hour(3600s) >
12	/fetch/database/ <network>/ from/<date>/	method: **GET** network:**liis_wsn01** date:**2015-05-16 14:00**	**"status":**400 **"success":**false **"message":** Your query is taking too long. Since you did not provide an email, your query will be ditched
13	/fetch/database/ <network>/ from/<date>/ ?email=<email>	method: **GET** network:**liis_wsn01** date:**2015-05-16 14:00** email:vhsousa@me.com	**"status":**400 **"success":**false **"message":** Your query is taking too long. The results will be sent to your email, as soon as they are gathered
14	/fetch/database/ <network>/ from/<date>/ ?heatmap=<ht>	method: **GET** network:**liis_wsn01** date:**2015-06-16** ht:**temperature**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-06-16> **"heatmap":** <url for the heatmap image>
15	/fetch/database/ <network>/ until/<date>/	method: **GET** network:**liis_wsn01** date:**2015-03-29**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-03-29>

#	Endpoint	Input	Expected result
16	/fetch/database/ <network>/ until/<date>/	method: **GET** network:**liis_wsn01** date:**2015-02-29**	**"status":**404 **"success":**false **"message":** "No results found"
17	/fetch/database/ <network>/ until/<date>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date:**2015-03-29** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-03-29 but only from node 101>
18	/fetch/database/ <network>/ until/<date>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date:**2015-03-29** **14:00** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-03-29 at 14h00 with results only from node 101>
19	/fetch/database/ <network>/ until/<date>/ ?deviceid=<deviceid> &samplingrate=<sr>	method: **GET** network:**liis_wsn01** date:**2015-03-29** **14:00** deviceid:**101** sr:**3600**s	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-03-29 at 14h00 with results only from node 101 and with a sampling rate of 1 hour(3600s) >
20	/fetch/database/ <network>/ until/<date>/	method: **GET** network:**liis_wsn01** date:**2015-03-29** **14:00**	**"status":**400 **"success":**false **"message":** Your query is taking too long. Since you did not provide an email, your query will be ditched

#	Endpoint	Input	Expected result
21	/fetch/database/ <network>/ until/<date>/ ?email=<email>	method: **GET** network:**liis_wsn01** date:**2015-03-29 14:00** email:vhsousa@me.com	**"status":**400 **"success":**false **"message":** Your query is taking too long. The results will be sent to your email, as soon as they are gathered
22	/fetch/database/ <network>/ until/<date>/ ?heatmap=<ht> &samplingrate=<sr>	method: **GET** network:**liis_wsn01** date:**2015-03-29** ht:**temperature** sr:**3600s**	**"status":**200 **"success":**true **"data":** <list with the database entries since 2015-03-29> **"heatmap":** <url for the heatmap image>
23	/fetch/database/ <network>/between/ <date1>/<date2>/	method: **GET** network:**liis_wsn01** date1:**2015-06-10** date2:**2015-06-11**	**"status":**200 **"success":**true **"data":** <list with the database entries between date1 and date2>
24	/fetch/database/ <network>/between/ <date1>/<date2>/	method: **GET** network:**liis_wsn01** date1:**2014-06-10** date2:**2014-06-11**	**"status":**404 **"success":**false **"message":** "No results found"
25	/fetch/database/ <network>/between/ <date1>/<date2>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date1:**2015-06-10** date2:**2015-06-11** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries between date1 and date2 but only from node 101>

#	Endpoint	Input	Expected result
26	/fetch/database/ <network>/between/ <date1>/<date2>/ ?deviceid=<deviceid>	method: **GET** network:**liis_wsn01** date1:**2015-06-10 14:00** date2:**2015-06-11 14:00** deviceid:**101**	**"status":**200 **"success":**true **"data":** <list with the database entries between date1 and date2 with results only from node 101>
27	/fetch/database/ <network>/between/ <date1>/<date2>/ ?deviceid=<deviceid> &samplingrate=<sr>	method: **GET** network:**liis_wsn01** date1:**2015-06-10** date2:**2015-06-11** deviceid:**101** sr:**3600**s	**"status":**200 **"success":**true **"data":** <list with the database entries between date1 and date2 with results only from node 101 and with a sampling rate of 1 hour(3600s) >
28	/fetch/database/ <network>/between/ <date1>/<date2>/	method: **GET** network:**liis_wsn01** date1:**2015-06-01** date2:**2015-06-10**	**"status":**400 **"success":**false **"message":** Your query is taking too long. Since you did not provide an email, your query will be ditched
29	/fetch/database/ <network>/between/ <date1>/<date2>/ ?email=<email>	method: **GET** network:**liis_wsn01** date1:**2015-06-01** date2:**2015-06-10** email:vhsousa@ me.com	**"status":**400 **"success":**false **"message":** Your query is taking too long. The results will be sent to your email, as soon as they are gathered

#	Endpoint	Input	Expected result
30	/fetch/database/ <network>/between/ <date1>/<date2>/ ?heatmap=<ht> &samplingrate=<sr>	method: **GET** network:**liis_wsn01** date1:**2015-06-10** date2:**2015-06-11** ht:**temperature** sr:**3600s**	**"status":**200 **"success":**true **"data":** <list with the database entries between date1 and date2> **"heatmap":** <url for the heatmap image>
31	/authenticate/ <network>/	method: **POST** network:**liis_dab** username:**vhsousa** password:**liis2014**	**"status":**200 **"success":**true **"token":** <JWT>
32	/authenticate/ <network>/	method: **POST** network:**liis_dab** username:**vhsousa** password:**liis2015**	**"status":**401 **"success":**false **"message":** Unauthorized. Invalid Credentials
33	/control/<network>/ <token>/ <deviceid>/ <value>/	method:**POST** network:**liis_dab** token:**<JWT>** deviceid:**ao0** value:**100**	**"status":**200 **"success":**true **"message":** Actuated successfully **"time_remaining":** x
34	/control/<network>/ <token>/ <deviceid>/ <value>/	method:**POST** network:**liis_dab** token:**<JWT>** deviceid:**ao0** value:**120**	**"status":**400 **"success":**false **"message":** Input exceeds limits of the device: 5 v **"time_remaining":** x

A.2 Back-end Tests

TABLE A.2: Back-end functional tests definition.

#	Test Condition	Input	Expected result
1	Add a new Virtual Sensor Network	vsn_test configuration file	Server loads the new network and it's available through the endpoint /info/vsn_test
2	Add a new Wireless Sensor Network	wsn_test configuration file	Server loads the new network and it's available through the endpoint /info/wsn_test
3	Add a new Data Acquisition Board	dab_test configuration file	Server loads the new network and it's available through the endpoint /info/dab_test
4	Start a new platform instance changing the default port through the command-line parameters	port: 9823	Server passes the startup procedure without any error or exception
5	Start a new platform instance changing the default port through the server configuration file	port: 9823	Server passes the startup procedure without any error or exception
6	Start a new platform instance changing the default fetch-thread timeout through the command-line parameters	thread_timeout: 120	Server passes the startup procedure without any error or exception
7	Start a new platform instance changing the default fetch-thread timeout through the server configuration file	thread_timeout: 120	Server passes the startup procedure without any error or exception

#	**Test Condition**	**Input**	**Expected result**
8	Connection Lost with a Wireless Sensor Network	-	Email is sent to the administrator alerting that the network is down
9	A WSN reconnects to the platform	-	All the mechanisms resumes normal operation

Appendix B

Tests results

B.1 API Tests

B.1.1 Test 1

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/networks
- **Response**


```
{
 "status":200,
 "networks":["liis_dab","vsn_test","liis_wsn03","liis_wsn02","liis_wsn10","
    liis_wsn01","liis_wsn06","liis_wsn07","liis_wsn04","liis_wsn05","
    liis_wsn08","liis_wsn09"],
 "success":true
}

```


- **Status** Passed

B.1.2 Test 2

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/info/liis_wsn01/
- **Response**


```
{
"status": 200,
"info": {
        "units": {
                "par": "lux",
                "temperature": "C",
                "battery": "volt",
                "internal-temperature": "C",
                "humidity": "%",
```

```
                "adc": "volt",
                "tsr": "lux"
        },
        "number-of-nodes": 3,
        "ids": ["102","103","101"],
        "sampling-time": 1,
        "general":"{\"admin\":\"vhsousa@student.dei.uc.pt\",\"master\":true,
    \"camera\":\"http://hydra.dei.uc.pt/remoteprocesscam/image/jpeg.cgi\",\"
    type\":\"wsn\"}"
},
"success": true
}
```


- **Status** Passed

B.1.3 Test 3

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
- **Response**

{"status": 200, "data": {"102": {"battery": 3.036, "timestamp": "2015-06-17 16:00:12 UTC", "internal-temperature": 27.3, "adc6": 0.004, "adc7": 0.001, "adc2": 1.547, "adc3": 1.586, "adc0": 1.522, "adc1": 1.53, "par": 1971.0, "temperature": 29.4, "humidity": 45.0, "tsr": 417.0}, "103": {"battery": 3.077, "timestamp": "2015-06-17 16:00:12 UTC", "internal-temperature": 26.2, "adc6": 0.006, "adc7": 0.0, "adc2": 1.494, "adc3": 1.464, "adc0": 1.58, "adc1": 1.511, "par": 1458.0, "temperature": 29.4, "humidity": 45.0, "tsr": 383.0}, "101": {"battery": 3.065, "timestamp": "2015-06-17 16:00:12 UTC", "internal-temperature": 34.1, "adc6": 0.001, "adc7": 0.001, "adc2": 1.501, "adc3": 1.475, "adc0": 1.546, "adc1": 1.579, "par": 4827.0, "temperature": 41.4, "humidity": 29.0, "tsr": 509.0}}, "ids": ["102", "103", "101"], "success": true}

- **Status** Passed

B.1.4 Test 4

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn11/
- **Response**

{"status":400,"message":"Invalid Network liis_wsn11","success":false}

- **Status** Passed

B.1.5 Test 5

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/?deviceid=101

- **Response**

{"status": 200, "data": {"battery": 3.063, "timestamp": "2015-06-17 16:01:04 UTC", "internal-temperature": 35.5, "adc6": 0.001, "adc7": 0.001, "adc2": 1.578, "adc3": 1.514, "adc0": 1.546, "adc1": 1.551, "par": 4791.0, "temperature": 42.4, "humidity": 28.0, "tsr": 507.0}, "success": true}

- **Status** Passed

B.1.6 Test 6

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/?deviceid=105

- **Response**

{"status":400,"message":"Bad Request. No data for this Node ID","success":false}

- **Status** Passed

B.1.7 Test 7

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-06-16/

- **Response**

{"status":200,"data":[{"date":"2015-06-16 00:00:00 UTC","data":{"102":{"par":885,"temperature":24.4,"battery":3.026,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":21.7,"humidity":62,"adc6":0.004,"adc7":0.001,"adc2":1.518,"adc3":1.544,"adc0":1.511,"adc1":1.512,"tsr":53},"103":{"par":799,"temperature":25.4,"battery":3.07,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":21.7,"humidity":61,"adc6":0.005,"adc7":0.001,"adc2":1.451,"adc3":1.505,"adc0":1.569,"adc1":1.486,"tsr":44},"101":{"par":836,"temperature":22.4,"battery":3.038,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":16.3,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.542,"adc3":1.558,"adc0":1.509,"adc1":1.524,"tsr":43}},"ids":["102","103","101"]},{"date":"2015-06-16 00:00:01 UTC","data":{"102":{"par":872,"temperature":24.4,"battery":3.029,"timestamp":"2015-06-16 00:00:00 UTC","internal-temperature":22.8,"humidity":62,"adc6":0.004,"adc7":0.001,"adc2":1.491,"adc3":1.469,"adc0":1.511,"adc1":1.503,"tsr":53},"103":

- **Status** Passed

B.1.8 Test 8

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-12-16/

- **Response**

- **Status** Passed

{"status":404,"message":"No results found","success":false}

B.1.9 Test 9

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-06-16/?deviceid=101

- **Response**

{"status":200,"data":[{"date":"2015-06-16 00:00:00 UTC","data":{"101":{"par":836,"temperature":22.4,"battery":3.038,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":16.3,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.542,"adc3":1.558,"adc0":1.509,"adc1":1.524,"tsr":43}}},{"date":"2015-06-16 00:00:01 UTC","data":{"101":{"par":811,"temperature":22.4,"battery":3.026,"timestamp":"2015-06-16 00:00:00 UTC","internal-temperature":16.6,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.534,"adc3":1.546,"adc0":1.518,"adc1":1.519,"tsr":42}}},{"date":"2015-06-16 00:00:02 UTC","data":{"101":{"par":744,"temperature":22.4,"battery":3.036,"timestamp":"2015-06-16 00:00:01 UTC","internal-temperature":16.3,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.508,"adc3":1.492,"adc0":1.509,"adc1":1.503,"tsr":51}}},{"date":"2015-06-16 00:00:03 UTC","data":{"101":{"par":738,"temperature":22.4,"battery":3.033,"timestamp":"2015-06-16 00:00:02 UTC","internal-temperature":16.1,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.57,"adc3":1.49,"adc0":1.506,"adc1":1.484,"tsr":51}}},{"date":"2015-06-16 00:00:04 UTC","data":{"101":

- **Status** Passed

B.1.10 Test 10

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-06-16%2014:00/?deviceid=101

- **Response**

{"status":200,"data":[{"date":"2015-06-16 14:00:00 UTC","data":{"101":{"par":4168,"temperature":30.4,"battery":3.046,"timestamp":"2015-06-16 14:00:00 UTC","internal-temperature":24.2,"humidity":47,"adc6":0.001,"adc7":0.001,"adc2":1.567,"adc3":1.469,"adc0":1.53,"adc1":1.55,"tsr":481}}},{"date":"2015-06-16 14:00:01 UTC","data":{"101":{"par":4168,"temperature":30.4,"battery":3.046,"timestamp":"2015-06-16 14:00:01 UTC","internal-temperature":24.2,"humidity":47,"adc6":0.001,"adc7":0.001,"adc2":1.577,"adc3":1.502,"adc0":1.526,"adc1":1.541,"tsr":481}}},{"date":"2015-06-16 14:00:02 UTC","data":{"101":{"par":4168,"temperature":30.4,"battery":3.051,"timestamp":"2015-06-16 14:00:02 UTC","internal-temperature":24.2,"humidity":47,"adc6":0.001,"adc7":0.004,"adc2":1.562,"adc3":1.6,"adc0":1.523,"adc1":1.541,"tsr":481}}},{"date":"2015-06-16 14:00:03 UTC","data":{"101":{"par":4156,"temperature":30.4,"battery":3.038,"timestamp":"2015-06-16 14:00:03 UTC","internal-temperature":24.5,"humidity":47,"adc6":0.001,"adc7":0.001,"adc2":1.551,"adc3":1.566,"adc0":1.524,"adc1":1.539,"tsr":481}}},{"date":"2015-06-16 14:00:04 UTC","data":{"101":

- **Status** Passed

B.1.11 Test 11

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-06-16%2014:00/?deviceid=101&samplingrate=3600

- **Response**

- **Status** Passed

{"status":200,"data":[{"date":"2015-06-16 14:00:00 UTC","data":{"101":{"par":4168,"temperature":30.4,"battery":3.046,"timestamp":"2015-06-16 14:00:00 UTC","internal-temperature":24.2,"humidity":47,"adc6":0.001,"adc7":0.001,"adc2":1.567,"adc3":1.469,"adc0":1.53,"adc1":1.55,"tsr":481}}},{"date":"2015-06-16 15:00:00 UTC","data":{"101":{"par":4266,"temperature":31.4,"battery":3.048,"timestamp":"2015-06-16 14:59:59 UTC","internal-temperature":25.6,"humidity":49,"adc6":0.001,"adc7":0.001,"adc2":1.609,"adc3":1.564,"adc0":1.533,"adc1":1.562,"tsr":482}}},{"date":"2015-06-16 16:00:00 UTC","data":{"101":{"par":4852,"temperature":38.4,"battery":3.053,"timestamp":"2015-06-16 15:59:59 UTC","internal-temperature":31.8,"humidity":38,"adc6":0.001,"adc7":0.001,"adc2":1.553,"adc3":1.566,"adc0":1.541,"adc1":1.552,"tsr":513}}},{"date":"2015-06-16 17:00:00 UTC","data":{"101":{"par":3851,"temperature":36.4,"battery":3.06,"timestamp":"2015-06-16 17:00:00 UTC","internal-temperature":31.8,"humidity":39,"adc6":0,"adc7":0.001,"adc2":1.498,"adc3":1.476,"adc0":1.545,"adc1":1.569,"tsr":433}}},{"date":"2015-06-16 18:00:00 UTC","data":{"101":{"par":3521,"temperature":29.4,"battery":3.038,"timestamp":"2015-06-16 17:59:59 UTC","internal-

B.1.12 Test 12

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-05-16%2014:00/
- **Response**

{"status":400,"message":"Your query is taking too long. Since you did not provide an email, your query will be ditched","success":false}

- **Status** Passed

B.1.13 Test 13

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-05-16%2014:00/?email=vhsousa@me.com
- **Response**

{"status":400,"message":"Your query is taking too long. The results will be sent to your email, as soon as they are gathered","success":false}

- **Status** Passed

B.1.14 Test 14

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/from/2015-06-16/?heatmap=temperature
- **Response**

{"status":200,"heatmap":"hydra.dei.uc.pt/supervision/api/v2/download/eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNDU1NDY3OCwiaWF0IjoxNDM0NTU0Mzc4fQ.e30.CQiwa9WOxcLbZWF6N0[{"date":"2015-06-16 00:00:00 UTC","data":{"102":{"par":885,"temperature":24.4,"battery":3.026,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":21.7,"humidity":62,"adc6":0.004,"adc7":0.001,"adc2":1.518,"adc3":1.544,"adc0":1.511,"adc1":1.512,"tsr":53},"103":{"par":799,"temperature":25.4,"battery":3.07,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":21.7,"humidity":61,"adc6":0.005,"adc7":0.001,"adc2":1.451,"adc3":1.505,"adc0":1.569,"adc1":1.486,"tsr":44},"101":{"par":836,"temperature":22.4,"battery":3.038,"timestamp":"2015-06-15 23:59:59 UTC","internal-temperature":16.3,"humidity":68,"adc6":0.001,"adc7":0.001,"adc2":1.542,"adc3":1.558,"adc0":1.509,"adc1":1.524,"tsr":43}},"ids":["102","103","101"]},{"date":"2015-06-16 01:00:00 UTC","data":{"102":{"par":878,"temperature":24.4,"battery":3.026,"timestamp":"2015-06-16 00:59:59 UTC","internal-temperature":21.7,"humidity":62,"adc6":0.004,"adc7":0.001,"adc2":1.459,"adc3":1.542,"adc0":1.508,"adc1":1.494,"tsr":51},"103":{"par":823,"temperature":25.4,"battery":3.07,"timestamp":"2015-06-16 00:59:59 UTC","internal-

- **Status** Passed

B.1.15 Test 15

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-03-29/
- **Response**

{"status":200,"data":[{"date":"2015-03-28 15:12:29 UTC","data":{"102":{"par":878,"temperature":22.4,"battery":2.972,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":21.1,"humidity":61,"adc6":0.004,"adc7":0,"adc2":1.528,"adc3":1.54,"adc0":1.516,"adc1":1.519,"tsr":46},"103":{"par":1104,"temperature":22.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":19.2,"humidity":61,"adc6":0.007,"adc7":0.001,"adc2":1.516,"adc3":1.525,"adc0":1.556,"adc1":1.581,"tsr":159},"101":{"par":4675,"temperature":39.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":32.4,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.498,"adc3":1.486,"adc0":1.548,"adc1":1.583,"tsr":487}},"ids":["102","103","101"]},{"date":"2015-03-28 15:12:30 UTC","data":{"102":{"par":836,"temperature":22.4,"battery":3.031,"timestamp":"2015-03-28 15:12:29 UTC","internal-temperature":20.3,"humidity":61,"adc6":0.005,"adc7":0.001,"adc2":1.514,"adc3":1.572,"adc0":1.511,"adc1":1.511,"tsr":46},"103":

- **Status** Passed

B.1.16 Test 16

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-02-29/
- **Response**

{"status":400,"message":"Invalid first date format","success":false}

- **Status** Passed

B.1.17 Test 17

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-03-29/?deviceid=101
- **Response**

{"status":200,"data":[{"date":"2015-03-28 15:12:29 UTC","data":{"101":{"par":4675,"temperature":39.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":32.4,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.498,"adc3":1.486,"adc0":1.548,"adc1":1.583,"tsr":487}}},{"date":"2015-03-28 15:12:30 UTC","data":{"101":{"par":4669,"temperature":39.4,"battery":3.058,"timestamp":"2015-03-28 15:12:29 UTC","internal-temperature":32.7,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.573,"adc3":1.502,"adc0":1.544,"adc1":1.547,"tsr":487}}},{"date":"2015-03-28 15:12:31 UTC","data":{"101":{"par":4675,"temperature":39.4,"battery":3.068,"timestamp":"2015-03-28 15:12:30 UTC","internal-temperature":33.2,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.584,"adc3":1.518,"adc0":1.546,"adc1":1.555,"tsr":487}}},{"date":"2015-03-28 15:12:32 UTC","data":{"101":{"par":4681,"temperature":39.4,"battery":3.053,"timestamp":"2015-03-28 15:12:31 UTC","internal-

- **Status** Passed

B.1.18 Test 18

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-03-29%2014:00/?deviceid=101
- **Response**

{"status":200,"data":[{"date":"2015-03-28 15:12:29 UTC","data":{"101":{"par":4675,"temperature":39.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":32.4,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.498,"adc3":1.486,"adc0":1.548,"adc1":1.583,"tsr":487}}},{"date":"2015-03-28 15:12:30 UTC","data":{"101":{"par":4669,"temperature":39.4,"battery":3.058,"timestamp":"2015-03-28 15:12:29 UTC","internal-temperature":32.7,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.573,"adc3":1.502,"adc0":1.544,"adc1":1.547,"tsr":487}}},{"date":"2015-03-28 15:12:31 UTC","data":{"101":{"par":4675,"temperature":39.4,"battery":3.068,"timestamp":"2015-03-28 15:12:30 UTC","internal-temperature":33.2,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.584,"adc3":1.518,"adc0":1.546,"adc1":1.555,"tsr":487}}},{"date":"2015-03-28 15:12:32 UTC","data":{"101":{"par":4681,"temperature":39.4,"battery":3.053,"timestamp":"2015-03-28 15:12:31 UTC","internal-temperature":32.7,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.545,"adc3":1.553,"adc0":1.556,"adc1":1.601,"tsr":487}}},{"date":"2015-03-28 15:12:33 UTC","data":{"101":{"par":4681,"temperature":39.4,"battery":3.065,"timestamp":"2015-03-28 15:12:32 UTC","internal-

- **Status** Passed

B.1.19 Test 19

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-03-29%2014:00/?deviceid=101&samplingrate=3600
- **Response**

{"status":200,"data":[{"date":"2015-03-28 15:12:29 UTC","data":{"101":{"par":4675,"temperature":39.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":32.4,"humidity":32,"adc6":0.001,"adc7":0.001,"adc2":1.498,"adc3":1.486,"adc0":1.548,"adc1":1.583,"tsr":487}}},{"date":"2015-03-28 16:12:29 UTC","data":{"101":{"par":2716,"temperature":23.4,"battery":2.98,"timestamp":"2015-03-28 16:12:28 UTC","internal-temperature":17.2,"humidity":57,"adc6":0.002,"adc7":0.001,"adc2":1.535,"adc3":1.552,"adc0":1.517,"adc1":1.519,"tsr":420}}},{"date":"2015-03-28 17:12:29 UTC","data":{"101":{"par":2197,"temperature":21.4,"battery":3.031,"timestamp":"2015-03-28 17:12:28 UTC","internal-temperature":14.9,"humidity":65,"adc6":0.001,"adc7":0.001,"adc2":1.508,"adc3":1.508,"adc0":1.509,"adc1":1.505,"tsr":386}}},{"date":"2015-03-28 18:12:29 UTC","data":{"101":{"par":1409,"temperature":19.4,"battery":3.026,"timestamp":"2015-03-28 18:12:28 UTC","internal-temperature":13.8,"humidity":67,"adc6":0.001,"adc7":0.001,"adc2":1.456,"adc3":1.522,"adc0":1.506,"adc1":1.487,"tsr":219}}},{"date":"2015-03-28 19:12:29 UTC","data":{"101":{"par":762,"temperature":18.4,"battery":2.98,"timestamp":"2015-03-28 19:12:29 UTC","internal-

- **Status** Passed

B.1.20 Test 20

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-04-29%2014:00/
- **Response**

{"status":400,"message":"Your query is taking too long. Since you did not provide an email, your query will be ditched","success":false}

- **Status** Passed

B.1.21 Test 21

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-04-29%2014:00/?email=vhsousa@me.com
- **Response**

{"status":400,"message":"Your query is taking too long. The results will be sent to your email, as soon as they are gathered","success":false}

- **Status** Passed

B.1.22 Test 22

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/until/2015-03-29/?heatmap=temperature&samplingrate=3600
- **Response**

{"status":200,"heatmap":"hydra.dei.uc.pt/supervision/api/v2/download/eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNDU2Mjc4MSwiaWF0IjoxNDM0NTYyNDgxfQ.e30.czC7Y1OD4b_JBrp0A7L_8iKfGlA2sk.png","data":[{"date":"2015-03-28 15:12:29 UTC","data":{"102":{"par":878,"temperature":22.4,"battery":2.972,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":21.1,"humidity":61,"adc6":0.004,"adc7":0,"adc2":1.528,"adc3":1.54,"adc0":1.516,"adc1":1.519,"tsr":46},"103":{"par":1104,"temperature":22.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-temperature":19.2,"humidity":61,"adc6":0.007,"adc7":0.001,"adc2":1.516,"adc3":1.525,"adc0":1.556,"adc1":1.581,"tsr":159},"101":{"par":4675,"temperature":39.4,"battery":3.063,"timestamp":"2015-03-28 15:12:28 UTC","internal-

- **Status** Passed

B.1.23 Test 23

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-10/2015-06-11/
- **Response**

{"status":200,"data":[{"date":"2015-06-10 00:00:00 UTC","data":{"102":{"par":909,"temperature":27.4,"battery":3.033,"timestamp":"2015-06-09 23:59:59 UTC","internal-temperature":25.4,"humidity":59,"adc6":0.004,"adc7":0.001,"adc2":1.472,"adc3":1.557,"adc0":1.509,"adc1":1.497,"tsr":51},"103":{"par":805,"temperature":28.4,"battery":3.075,"timestamp":"2015-06-09 23:59:59 UTC","internal-temperature":25.4,"humidity":57,"adc6":0.005,"adc7":0.001,"adc2":1.484,"adc3":1.581,"adc0":1.573,"adc1":1.498,"tsr":44},"101":{"par":775,"temperature":27.4,"battery":3.041,"timestamp":"2015-06-09 23:59:59 UTC","internal-temperature":21.1,"humidity":61,"adc6":0.001,"adc7":0,"adc2":1.525,"adc3":1.516,"adc0":1.516,"adc1":1.512,"tsr":51}},"ids":["102","103","101"]},{"date":"2015-06-10 00:00:01 UTC","data":{"102":{"par":854,"temperature":27.4,"battery":3.033,"timestamp":"2015-06-10 00:00:00 UTC","internal-

- **Status** Passed

B.1.24 Test 24

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2014-06-10/2014-06-11/
- **Response**

{"status":404,"message":"No results found","success":false}

- **Status** Passed

B.1.25 Test 25

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-10/2015-06-11/?deviceid=101
- **Response**

{"status":200,"data":[{"date":"2015-06-10 00:00:00 UTC","data":{"101":{"par":775,"temperature":27.4,"battery":3.041,"timestamp":"2015-06-09 23:59:59 UTC","internal-temperature":21.1,"humidity":61,"adc6":0.001,"adc7":0,"adc2":1.525,"adc3":1.516,"adc0":1.516,"adc1":1.512,"tsr":51}}},{"date":"2015-06-10 00:00:01 UTC","data":{"101":{"par":830,"temperature":27.4,"battery":3.033,"timestamp":"2015-06-10 00:00:00 UTC","internal-temperature":20.8,"humidity":61,"adc6":0.001,"adc7":0.001,"adc2":1.535,"adc3":1.555,"adc0":1.52,"adc1":1.525,"tsr":42}}},{"date":"2015-06-10 00:00:02 UTC","data":{"101":{"par":793,"temperature":27.4,"battery":3.043,"timestamp":"2015-06-10 00:00:01 UTC","internal-

- **Status** Passed

B.1.26 Test 26

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-10%2014:00/2015-06-11%2014:00/?deviceid=101
- **Response**

{"status":200,"data":[{"date":"2015-06-10 14:00:00 UTC","data":{"101":{"par":4241,"temperature":29.4,"battery":3.046,"timestamp":"2015-06-10 14:00:00 UTC","internal-temperature":23.4,"humidity":53,"adc6":0.001,"adc7":0.001,"adc2":1.585,"adc3":1.511,"adc0":1.522,"adc1":1.557,"tsr":474}}},{"date":"2015-06-10 14:00:01 UTC","data":{"101":{"par":4254,"temperature":29.4,"battery":3.043,"timestamp":"2015-06-10 14:00:01 UTC","internal-temperature":23.4,"humidity":53,"adc6":0.001,"adc7":0.002,"adc2":1.578,"adc3":1.502,"adc0":1.526,"adc1":1.542,"tsr":474}}},{"date":"2015-06-10 14:00:02 UTC","data":{"101":{"par":4241,"temperature":29.4,"battery":3.043,"timestamp":"2015-06-10 14:00:02 UTC","internal-temperature":23.4,"humidity":53,"adc6":0.001,"adc7":0.001,"adc2":1.536,"adc3":1.566,"adc0":1.522,"adc1":1.535,"tsr":474}}},{"date":"2015-06-10 14:00:03 UTC","data":{"101":{"par":4241,"temperature":29.4,"battery":3.053,"timestamp":"2015-06-10 14:00:03 UTC","internal-

- **Status** Passed

B.1.27 Test 27

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-10/2015-06-11/?deviceid=101&samplingrate=3600
- **Response**

{"status":200,"data":[{"date":"2015-06-10 00:00:00 UTC","data":{"101":{"par":775,"temperature":27.4,"battery":3.041,"timestamp":"2015-06-09 23:59:59 UTC","internal-temperature":21.1,"humidity":61,"adc6":0.001,"adc7":0,"adc2":1.525,"adc3":1.516,"adc0":1.516,"adc1":1.512,"tsr":51}}},{"date":"2015-06-10 01:00:00 UTC","data":{"101":{"par":769,"temperature":26.4,"battery":3.043,"timestamp":"2015-06-10 00:59:59 UTC","internal-temperature":20.3,"humidity":63,"adc6":0.001,"adc7":0.002,"adc2":1.526,"adc3":1.516,"adc0":1.519,"adc1":1.529,"tsr":51}}},{"date":"2015-06-10 02:00:00 UTC","data":{"101":{"par":817,"temperature":25.4,"battery":3.033,"timestamp":"2015-06-10 02:00:00 UTC","internal-temperature":19.2,"humidity":65,"adc6":0.002,"adc7":0.001,"adc2":1.485,"adc3":1.443,"adc0":1.512,"adc1":1.506,"tsr":40}}},{"date":"2015-06-10 03:00:00 UTC","data":{"101":{"par":848,"temperature":25.4,"battery":3.033,"timestamp":"2015-06-10 02:59:58 UTC","internal-temperature":19.2,"humidity":66,"adc6":0.001,"adc7":0.001,"adc2":1.475,"adc3":1.561,"adc0":1.508,"adc1":1.503,"tsr":42}}},{"date":"2015-06-10 04:00:00 UTC","data":{"101":{"par":775,"temperature":24.4,"battery":3.038,"timestamp":"2015-06-10 04:00:00 UTC","internal-

- **Status** Passed

B.1.28 Test 28

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-01/2015-06-10/
- **Response**

{"status":400,"message":"Your query is taking too long. Since you did not provide an email, your query will be ditched","success":false}

- **Status** Passed

B.1.29 Test 29

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-01/2015-06-10/?email=vhsousa@me.com
- **Response**

{"status":400,"message":"Your query is taking too long. The results will be sent to your email, as soon as they are gathered","success":false}

- **Status** Passed

B.1.30 Test 30

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-06-10/2015-06-11/?heatmap=temperature&samplignrate=3600
- **Response**

{"status":200,"heatmap":"hydra.dei.uc.pt/supervision/api/v2/download/eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNDU2MjQ2NSwiaWF0IjoxNDM0NTYyMTY1fQ.e30.LC_31lFK6ZjY5TOh5eu3.
[{"date":"2015-06-10 00:00:00 UTC","data":{"102":{"par":909,"temperature":27.4,"battery":3.033,"timestamp":"2015-06-09 23:59:59 UTC","internal-
temperature":25.4,"humidity":59,"adc6":0.004,"adc7":0.001,"adc2":1.472,"adc3":1.557,"adc0":1.509,"adc1":1.497,"tsr":51},"103":
{"par":805,"temperature":28.4,"battery":3.075,"timestamp":"2015-06-09 23:59:59 UTC","internal-
temperature":25.4,"humidity":57,"adc6":0.005,"adc7":0.001,"adc2":1.484,"adc3":1.581,"adc0":1.573,"adc1":1.498,"tsr":44},"101":
{"par":775,"temperature":27.4,"battery":3.041,"timestamp":"2015-06-09 23:59:59 UTC","internal-
temperature":21.1,"humidity":61,"adc6":0.001,"adc7":0,"adc2":1.525,"adc3":1.516,"adc0":1.516,"adc1":1.512,"tsr":51}},"ids":["102","103","101"]},{"date":"2015-06-10 01:00:00 UTC","data":
{"102":{"par":915,"temperature":27.4,"battery":3.033,"timestamp":"2015-06-10 00:59:59 UTC","internal-
temperature":25.4,"humidity":59,"adc6":0.004,"adc7":0.001,"adc2":1.53,"adc3":1.558,"adc0":1.514,"adc1":1.52,"tsr":51},"103":

- **Status** Passed

B.1.31 Test 31

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/authenticate/liis_dab/
- **Response**


```
1       {
2       "status": 200,
3       "token": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNjAxOTk0NSwiaWF0IjoxNDM2
    MDE2MzQ1fQ.eyJ1c2VybmFtZSI6InZoc291c2EiLCJzeXN0ZW0iOiJsaWlzX2RhYiJ9.dS3
    qmQHr-H7PbN8yiy48Nb1acWlH6-XPJup8x2eg9Wk",
4       "success": true
5       }
6
```


- **Status** Passed

B.1.32 Test 32

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/authenticate/liis_dab/
- **Response**


```
1       {
2       "status": 401,
3       "message": "Unauthorized. Invalid Credentials",
4       "success": false
5       }
6
```


- **Status** Passed

B.1.33 Test 33

- **Request**: http://hydra.dei.uc.pt/supervision/api/v2/control/liis_dab/eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNjAxOTk0NSwiaWF0IjoxNDM2MDE2MzQ1fQ.eyJ1c2VybmFtZSI6InZoc291c2EiLCJzeXN0ZW0iOiJsaWlzX2RhYiJ9.dS3qmQHr-H7PbN8yiy48Nb1acWlH6-XPJup8x2eg9Wk/ao0/100/
- **Response**

```
1  {
2  "status": 200,
3  "time_remaining": 120,
4  "message": "Actuated successfully",
5  "success": true
6  }
7
```

- **Status** Passed

B.1.34 Test 34

- **Request** http://hydra.dei.uc.pt/supervision/api/v2/control/liis_dab/eyJhbGciOiJIUzI1NiIsImV4cCI6MTQzNjAxOTk0NSwiaWF0IjoxNDM2MDE2MzQ1fQ.eyJ1c2VybmFtZSI6InZoc291c2EiLCJzeXN0ZW0iOiJsaWlzX2RhYiJ9.dS3qmQHr-H7PbN8yiy48Nb1acWlH6-XPJup8x2eg9Wk/ao0/120/
- **Response**

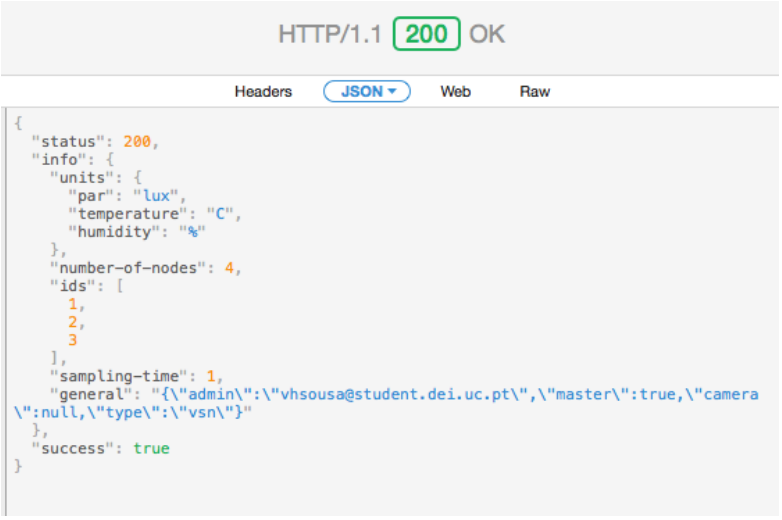
```
1  {
2  "status": 400,
3  "message": "Input exceeds limits of the device: 5 v",
4  "success": false
5  }
6
```

- **Status** Passed

B.2 Back-end Tests

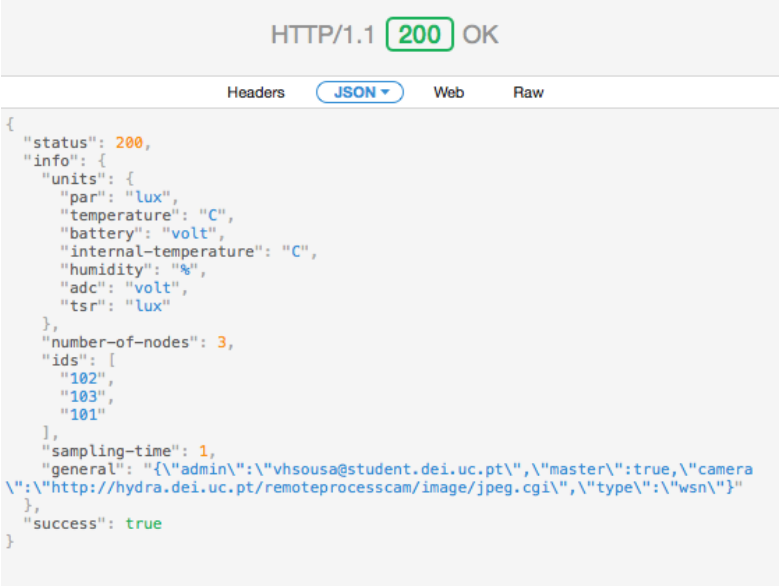
B.2.1 Test 1

- **Condition** Add a new Virtual Sensor Network
- **Result** The network vsn_test is now accessible at /info/vsn _test/
- **Status** Passed



B.2.2 Test 2

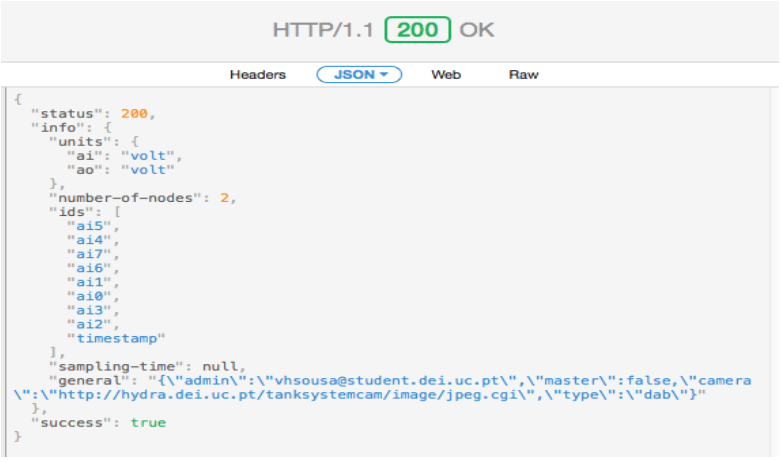
- **Condition** Add a new Wireless Sensor Network
- **Result** The network wsn_test is now accessible at /info/wsn _test/



- **Status** Passed

B.2.3 Test 3

- **Condition** Add a new Data Acquisition Board
- **Result** The network dab_test is now accessible at /info/dab _test/
- **Status** Passed



B.2.4 Test 4

- **Condition** Start a new platform instance on port 9823 through a command line parameter
- **Result** Using the parameter '-p' as in 'python restful-server.py -p 9823', it is possible to change the default port of the server.

```
17/06/2015 21:10:35][INFO][liis_wsn06] Configurations Loaded for network
17/06/2015 21:10:35][INFO] * Running on http://0.0.0.0:9823/ (Press CTRL+C to quit)
17/06/2015 21:10:36][INFO][liis_wsn01] No value from node 101
```

- **Status** Passed

B.2.5 Test 5

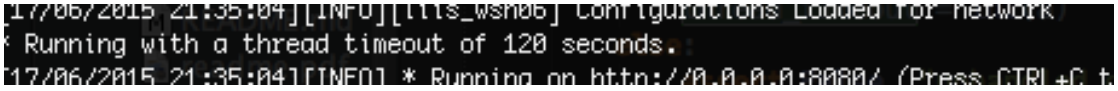
- **Condition** Start a new platform instance on port 9823 through the configuration file.
- **Result** Using the parameter '-c' as in 'python restful-server.py -c conf.json', it is possible to load a JSON configuration file with "port", "networks" and "thread_timeout" parameters.

```
17/06/2015 21:20:01][INFO] * Running on http://0.0.0.0:9823/ (Press CTRL+C to quit)
```

- **Status** Passed

B.2.6 Test 6

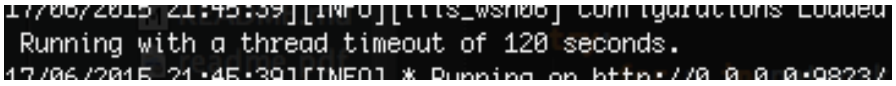
- **Condition** Start a new platform instance with a thread timeout of 120 seconds, through a command line parameter
- **Result** Using the parameter '-t' as in 'python restful-server.py -t 120', it is possible to change the default thread timeout of the server.



- **Status** Passed

B.2.7 Test 7

- **Condition** Start a new platform instance on port 9823 through the configuration file.
- **Result** Using the parameter '-c' as in 'python restful-server.py -c conf.json', it is possible to load a JSON configuration file with "port", "networks" and "thread_timeout" parameters.



- **Status** Passed

B.2.8 Test 8

- **Condition** Connection with a Wireless Sensor Network dispatcher is lost
- **Result** After 5 connection retries, the administrator of the network is alerted of the situation
- **Status** Passed

<liis-lab@dei.uc.pt>
Para: Vitor Sousa
[Network IPv6 liis_wsn01] Connection Problem

Hello Admin,

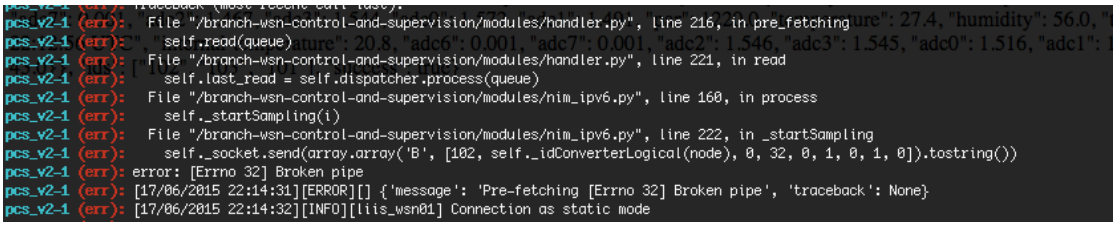
The connection to the WSN has reached a maximum of 5. Please check with is wrong.

IP: 10.3.3.112
PORT: 12345

Thanks

B.2.9 Test 9

- **Condition** A Wireless Sensor Network dispatcher reconnects to the platform
- **Result** All mechanisms resume normal operations.



- **Status** Passed

Appendix C

Benchmarking results

C.1 Standalone scenario

C.1.1 Test 1 - Small 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Progress [========================================] 100% 0.0s conc:0 278/s

errors:  0
stats:  { totalElapsed: 3623.9584990143776,
main: { meter:
        { mean: 278.4362447692208,
        count: 1000,
        currentRate: 280.5756198841005,
        '1MinuteRate': 0,
        '5MinuteRate': 0,
        '15MinuteRate': 0 },
        histogram:      { min: 107.12453600764275,
                max: 235.10389000177383,
                sum: 172715.6273112893,
                variance: 226.03482449181942,
                mean: 172.71562731128932,
                stddev: 15.034454579126555,
                count: 1000,
                median: 172.93098299205303,
                p75: 179.38070376217365,
                p95: 196.2859849512577,
                p99: 222.77655490398408,
                p999: 235.0972281517686 } } }
```

C.1.2 Test 2 - Small 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
```

```
Progress [========================================] 100% 0.0s conc:0 281/s

errors:  0
stats:  { totalElapsed: 17875.07683700323,
main: { meter:
        { mean: 281.38720237948974,
        count: 5000,
        currentRate: 261.4361577804727,
        '1MinuteRate': 63.01883053763066,
        '5MinuteRate': 13.888227217351368,
        '15MinuteRate': 4.706411437356729 },
        histogram:      { min: 111.96197098493576,
                max: 404.978128015995,
                sum: 880444.8200371265,
                variance: 198.40592786378832,
                mean: 176.08896400742532,
                stddev: 14.085663912779841,
                count: 5000,
                median: 175.36499550938606,
                p75: 184.98664101213217,
                p95: 195.91386535614728,
                p99: 205.6593801492453,
                p999: 404.07545330938706 } } }
```

C.1.3 Test 3 - Small 10000

```
$ bench-rest -n 10000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    recent/liis_wsn01/
Benchmarking 10000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Progress [========================================] 100% 0.0s conc:0 269/s

errors:  0
stats:  { totalElapsed: 37407.43246102333,
main: { meter:
        { mean: 268.97820660803893,
        count: 10000,
        currentRate: 282.79778296000205,
        '1MinuteRate': 118.57265589302867,
        '5MinuteRate': 29.658024032300002,
        '15MinuteRate': 10.279735685126555 },
        histogram:      { min: 98.42391300201416,
                max: 398.8617199957371,
                sum: 1851112.0583595932,
                variance: 283.76143016405484,
                mean: 185.1112058359593,
                stddev: 16.845219801595196,
                count: 10000,
                median: 185.3311125189066,
                p75: 195.8551039993763,
                p95: 215.50707319527862,
                p99: 234.4049980250001,
                p999: 247.2208455658853 } } }
```

C.1.4 Test 4 - Moderate 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/
    between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:0 2/s

errors:  669
```

```
stats:  { totalElapsed: 426466.55399897695,
main: { meter:
        { mean: 2.3449852984289605,
        count: 1000,
        currentRate: 936.0600267127926,
        '1MinuteRate': 1.0969616278097978,
        '5MinuteRate': 1.0051339143632672,
        '15MinuteRate': 0.5423435426144453 },
        histogram:      { min: 31.72358199954033,
                max: 60242.00378400087,
                sum: 21317791.763157994,
                variance: 748841089.9756408,
                mean: 21317.791763157995,
                stddev: 27364.960989843214,
                count: 1000,
                median: 1434.532360509038,
                p75: 60006.77843025327,
                p95: 60009.81715217531,
                p99: 60086.77107515067,
                p999: 60241.98389444789 } } }
```

C.1.5 Test 5 - Moderate 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/
    between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:0 2/s

errors:  1605
stats:  { totalElapsed: 2216623.083884001,
main: { meter:
        { mean: 2.2557894059318215,
        count: 5000,
        currentRate: 2.100654814331804,
        '1MinuteRate': 1.317787551048485,
        '5MinuteRate': 1.8375989132390127,
        '15MinuteRate': 1.863273139263995 },
        histogram: { min: 371.6240490078926,
                max: 60216.55198299885,
                sum: 108423109.92340195,
                variance: 740305591.3735102,
                mean: 21684.62198468039,
                stddev: 27208.557318856696,
                count: 5000,
                median: 1267.428599998355,
                p75: 60007.54934449494,
                p95: 60041.39262700081,
                p99: 60054.14178850591,
                p999: 60081.61743149805 } } }
```

C.1.6 Test 6 - Moderate 10000

```
$ bench-rest -n 10000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 10000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/
    between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:5 2/s

errors:  2453
stats:  { totalElapsed: 3609068.7201870084,
main: { meter:
```

```
        { mean: 2.7709615406996497,
        count: 10000,
        currentRate: 2.687812584202406,
        '1MinuteRate': 1.4121227832264576,
        '5MinuteRate': 2.237393873332537,
        '15MinuteRate': 2.4315233342847793 },
        histogram:      { min: 330.832987010479,
                max: 60213.842245996,
                sum: 179052716.36869335,
                variance: 642207299.6660855,
                mean: 17905.271636869336,
                stddev: 25341.809321082135,
                count: 10000,
                median: 1439.621718004346,
                p75: 60006.761581502855,
                p95: 60008.292285099626,
                p99: 60014.27010460794,
                p999: 60035.83111399594 } } }
```

C.1.7 Test 7 - Heavy 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/
    between/2015-05-01/2015-05-02/?samplingrate=600
Progress [========================================] 100% 0.0s conc:5 2/s

errors:  628
stats:  { totalElapsed: 540311.0297130048,
main: { meter:
        { mean: 1.850890280452474,
        count: 1000,
        currentRate: 7.63161063992038,
        '1MinuteRate': 4.121376290591109,
        '5MinuteRate': 1.8320994205997052,
        '15MinuteRate': 0.8741931918775679 },
        histogram: { min: 129.72210198640823,
                max: 67223.52881500125,
                sum: 26343525.693567663,
                variance: 727408243.5086714,
                mean: 26343.525693567663,
                stddev: 26970.50691975721,
                count: 1000,
                median: 4493.075258508325,
                p75: 60007.46653249115,
                p95: 60013.81241939366,
                p99: 64860.256787961465,
                p999: 67223.52638803223 } } }
```

C.1.8 Test 8 - Heavy 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/
    database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/
    between/2015-05-01/2015-05-02/?samplingrate=600
Progress [========================================] 100% 0.0s conc:5 2/s

errors:  2092
stats:  { totalElapsed: 2877997.7980590165,
main: { meter:
        { mean: 1.7373823180035426,
        count: 5000,
```

```
        currentRate: 6.278463965099324,
        '1MinuteRate': 2.0884575863034276,
        '5MinuteRate': 2.00471697426522,
        '15MinuteRate': 1.7348056191123928 },
        histogram: { min: 3.2220929861068726,
                max: 60143.657187998295,
                sum: 142280741.97609693,
                variance: 722465966.7057348,
                mean: 28456.148395219385,
                stddev: 26878.72702911607,
                count: 5000,
                median: 6438.0112404972315,
                p75: 60007.2398544848,
                p95: 60009.11733900309,
                p99: 60012.94824192226,
                p999: 60066.92654612556 } } }
```

C.1.9 Test 9 - Heavy 10000

```
Did Not Finish
```

C.2 Distributed scenario

C.2.1 Test 1 - Small 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent
    /liis_wsn01/
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Progress [========================================] 100% 0.0s conc:0 475/s

errors:  0
stats:  { totalElapsed: 2140.134290009737,
main: { meter:
        { mean: 474.9320083658196,
        count: 1000,
        currentRate: 658.0085559253455,
        '1MinuteRate': 0,
        '5MinuteRate': 0,
        '15MinuteRate': 0 },
        histogram: { min: 11.01552301645279,
                max: 317.43131598830223,
                sum: 99178.30727088451,
                variance: 2436.194935583726,
                mean: 99.17830727088452,
                stddev: 49.35782547462688,
                count: 1000,
                median: 84.588982000947,
                p75: 130.2738462537527,
                p95: 183.06794265061606,
                p99: 267.09811198800804,
                p999: 317.4301757513285 } } }
```

C.2.2 Test 2 - Small 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Progress [========================================] 100% 0.0s conc:0 568/s

errors:  0
stats:  { totalElapsed: 8910.53682500124,
main: { meter:
        { mean: 568.1689988677684,
        count: 5000,
        currentRate: 508.4395353783525,
        '1MinuteRate': 45.84653265154609,
        '5MinuteRate': 9.477468378684593,
        '15MinuteRate': 3.176723153992152 },
        histogram: { min: 2.7009589970111847,
                max: 485.5687710046768,
                sum: 431546.09162795544,
                variance: 5850.268973978615,
                mean: 86.30921832559109,
                stddev: 76.48705102158544,
                count: 5000,
                median: 118.46559999883175,
                p75: 163.60698150098324,
                p95: 182.31061576008796,
                p99: 216.33568752139817,
                p999: 433.076977509231 } } }
```

C.2.3 Test 3 - Small 10000

```
$ bench-rest -n 10000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Benchmarking 10000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/recent/liis_wsn01/
Progress [========================================] 100% 0.0s conc:0 549/s

errors:  0
stats:  { totalElapsed: 18501.65299001336,
main: { meter:
        { mean: 549.2979316979639,
        count: 10000,
        currentRate: 352.3060242648794,
        '1MinuteRate': 124.28932612955519,
        '5MinuteRate': 27.397618379701562,
        '15MinuteRate': 9.284812454075418 },
        histogram: { min: 2.592501014471054,
                max: 618.7567619979382,
                sum: 901367.1941169202,
                variance: 6634.934451308326,
                mean: 90.13671941169203,
                stddev: 81.4551069688594,
                count: 10000,
                median: 57.9816310107708,
                p75: 170.40583249926567,
                p95: 196.81019430458545,
                p99: 216.84557927280665,
                p999: 249.05417836061125 } } }
```

C.2.4 Test 4 - Moderate 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:3 11/s

errors:  10
stats:  { totalElapsed: 141125.61739900708,
main: { meter:
        { mean: 7.0871650342808135,
        count: 1000,
        currentRate: 5.3647302815218465,
        '1MinuteRate': 4.586150236580643,
        '5MinuteRate': 2.523544487812693,
        '15MinuteRate': 1.0102536323810112 },
        histogram:{ min: 152.72400000691414,
                max: 120105.47211900353,
                sum: 5928349.952686518,
                variance: 295913272.5156277,
                mean: 5928.349952686518,
                stddev: 17202.129883117024,
                count: 1000,
                median: 759.8622290194035,
                p75: 932.4883162528276,
                p95: 60223.38605868071,
                p99: 60978.20882558554,
                p999: 120105.45228691954 } } }
```

C.2.5 Test 5 - Moderate 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:24 2/s

errors:  47
stats:  { totalElapsed: 418786.22589698434,
main: { meter:
        { mean: 11.942531885555692,
        count: 5000,
        currentRate: 2.5810580839774535,
        '1MinuteRate': 6.157131295016485,
        '5MinuteRate': 8.254208096898411,
        '15MinuteRate': 4.335227729857532 },
        histogram: { min: 145.0069130063057,
                max: 120097.92607998848,
                sum: 19421799.267213047,
                variance: 219458941.41870704,
                mean: 3884.359853442609,
                stddev: 14814.146665221964,
                count: 5000,
                median: 385.84690798819065,
                p75: 748.0849217474461,
                p95: 60257.51644394696,
                p99: 103135.06078876945,
                p999: 120013.16534028479 } } }
```

C.2.6 Test 6 - Moderate 10000

```
$ bench-rest -n 10000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Benchmarking 10000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=3600
Progress [========================================] 100% 0.0s conc:44 8/s

errors:  126
stats:  { totalElapsed: 860182.7290740013,
main: { meter:
        { mean: 11.628400419573383,
        count: 10000,
        currentRate: 2.850427469952005,
        '1MinuteRate': 5.073493899662898,
        '5MinuteRate': 9.250925967706149,
        '15MinuteRate': 6.79299784571018 },
        histogram: { min: 1.9229399859905243,
                max: 126747.89685499668,
                sum: 41138739.17031655,
                variance: 257270398.13791355,
                mean: 4113.873917031655,
                stddev: 16039.650810971963,
                count: 10000,
                median: 630.2924530059099,
                p75: 895.819596260786,
                p95: 60484.60631704926,
                p99: 103365.98573436736,
                p999: 120016.28803542662 } } }
```

C.2.7 Test 7 - Heavy 1000

```
$ bench-rest -n 1000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Benchmarking 1000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Progress [========================================] 100% 0.0s conc:5 2/s

errors:  15
stats:  { totalElapsed: 161508.92374899983,
main: { meter:
        { mean: 6.193191524757363,
        count: 1000,
        currentRate: 3.214154312866782,
        '1MinuteRate': 4.6709132054035605,
        '5MinuteRate': 2.516717737959244,
        '15MinuteRate': 1.0094662204583655 },
        histogram: { min: 15.233238011598587,
                max: 120110.44702798128,
                sum: 6528580.207422644,
                variance: 392528853.8976216,
                mean: 6528.580207422644,
                stddev: 19812.340949459292,
                count: 1000,
                median: 821.2134099751711,
                p75: 1145.4326097518206,
                p95: 60650.43591144681,
                p99: 120015.75932596475,
                p999: 120110.4445516533 } } }
```

C.2.8 Test 8 - Heavy 5000

```
$ bench-rest -n 5000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Benchmarking 5000 iteration(s) using up to 50 concurrent connections

Progress [========================================] 100% 0.0s conc:24 7/s

errors:  43
stats:  { totalElapsed: 558429.6055970192,
main:
{ meter:
{ mean: 8.955164054423522,
count: 5000,
currentRate: 33.67553400219773,
'1MinuteRate': 4.785332421066812,
'5MinuteRate': 6.829210106435755,
'15MinuteRate': 4.016999048248094 },
histogram:
{ min: 210.24665799736977,
max: 120059.37963500619,
sum: 26201967.756153792,
variance: 293016351.1343462,
mean: 5240.393551230758,
stddev: 17117.720383694385,
count: 5000,
median: 890.6764625012875,
p75: 1284.5379507541656,
p95: 60775.01846369505,
p99: 120011.65967198342,
p999: 120014.26740372449 } } }
```

C.2.9 Test 9 - Heavy 10000

```
$ bench-rest -n 10000 -c 50 http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Benchmarking 10000 iteration(s) using up to 50 concurrent connections

flow:  http://hydra.dei.uc.pt/supervision/api/v2/fetch/database/liis_wsn01/between/2015-05-01/2015-05-02/?samplingrate=600
Progress [========================================] 100% 0.0s conc:5 2/s

errors:  146
stats:  { totalElapsed: 1320666.0195119977,
main:  { meter:
        { mean: 7.573026189011743,
        count: 10000,
        currentRate: 3.785801300594897,
        '1MinuteRate': 2.1264375727622173,
        '5MinuteRate': 5.087058166678599,
        '15MinuteRate': 5.095478785905252 },
        histogram: { min: 1.954048991203308,
                max: 120421.08014401793,
                sum: 64302869.60345107,
                variance: 367279160.1360143,
                mean: 6430.286960345107,
                stddev: 19164.528695901037,
                count: 10000,
                median: 1561.4685180038214,
                p75: 2313.7137604877353,
                p95: 62516.00978614389,
                p99: 120012.70189100862,
                p999: 120059.41709774289 } } }
```