Bruno Miguel Vara Rodrigues

Licenciado em Ciências de Engenharia Electrotécnica e Computadores

Teleoperation of a Service Robot using a Mobile Device

Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e Computadores

Orientador: João Paulo Pimentão, Professor Auxiliar, FCT-UNL

Co-orientador: Pedro Alexandre Sousa, Professor Auxiliar, FCT-UNL

Júri:

Presidente: Prof. Doutor José Barata de Oliveira

Arguentes: Prof. Doutor Ricardo Jardim Gonçalves

Vogais: Prof. Doutor João Paulo Pimentão

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Teleoperation of a Service Robot using a Mobile Device

«Aquele que se empenha a resolver as dificuldades resolve-as antes que elas surjam.»

Sun Tzu

Agradecimentos

Aqui expresso os meus sinceros agradecimentos, a todos aqueles que me apoiaram durante toda esta jornada académica, e que ainda hoje fazem parte da minha vida.

Primeiramente, quero agradecer aos Professores Doutores **João Pimentão** e **Pedro Sousa**, sem esquecer toda a **equipa da Holos**, que sempre se mostraram disponíveis a ajudar por mais que fossem as dúvidas.

Aos meus colegas e grandes amigos: **Carlota Maçaneiro, Gonçalo Amaro, Flávio Páscoa, Filipa Matias, Júlio Oliveira e Rafaela Nunes**, o meu grande obrigado por me acompanharem este tempo todo.

Por último, e como os últimos são sempre os primeiros um especial obrigado á minha Família: **pai, mãe, irmã, avós e padrinho(tio)**.

Igualmente à família **Cristina-Martins** pelo acompanhamento que me deram nestes últimos anos, e que se sentiram igualmente orgulhosos pelo atingir dos meus objectivos.

Resumo

Teleoperação é um conceito que nasce com a rápida evolução da tecnologia, com um significado bastante intuitivo "operar á distância".

O primeiro sistema de Teleoperação foi criado em meados de 1950, onde eram manuseados produtos químicos. Os sistemas de controlo á distância estão presentes hoje em dia nos mais diversos tipos de aplicações.

Esta dissertação apresenta o desenvolvimento de uma aplicação móvel para efectuar a teleoperação de um robot de serviço móvel.

A aplicação integra-se num sistema distribuído de vigilância(resultado de um projecto de investigação QREN) e obrigou ao desenvolvimento de uma interface de comunicação entre o robot(resultado de outro projecto QREN) e o sistema de vigilância.

Foi necessário especificar uma linguagem de comunicação entre os dois sistemas, que foi implementada sobre uma *framework* de comunicação 0MQ (Zero Message Queue).

Para a realização dos testes da aplicação foram desenvolvidos três protótipos de suporte antes de se efectuar o teste sobre o robot.

Abstract

Teleoperation is a concept born with the rapid evolution of technology, with an intuitive meaning "operate at a distance."

The first teleoperation system was created in the mid 1950s, which were handled chemicals. Remote controlled systems are present nowadays in various types of applications.

This dissertation presents the development of a mobile application to perform the teleoperation of a mobile service robot.

The application integrates a distributed surveillance (the result of a research project QREN) and led to the development of a communication interface between the robot (the result of another QREN project) and the vigilance system.

It was necessary to specify a communication protocol between the two systems, which was implemented over a communication framework 0MQ (Zero Message Queue).

For the testing, three prototype applications were developed before to perform the test on the robot.

Keywords: Teleoperation, Surveillance, Mobile Application, 0MQ.

Index

Images Index

1

1 Introduction

1.1 Motivation

Robot, a common word used by everyone in general, it is a Slavic word(from Czech: *Robota*) which has the meaning of "compulsory work" or "forced work".

Over the centuries, people created mechanisms to imitate human body parts. In ancient Egypt and Greece, machines identical to today's robots were used and were controlled using mechanic and hydraulic mechanisms.

Many people tend to think that robots need to have a human appearance, however, a robot is considered as a machine that is given a task to perform (a washing machine, for instance, is by this definition a domestic robot, which has the task to wash our clothes). Yet, the idea of a humanoid is not excluded from research, Toyota and Honda are just examples of two well-known brands, which have been researching in these kinds of robots.

Robots have been improved along the time mainly due the technological advances of electronics and computer engineering.

Nowadays, the massive usage of robots is concentrated in industrial applications. Factories contain very hazardous environments (such as soldering or painting stations), where humans may become injured performing such tasks. Also productivity of humans is impaired by the fact that these are usually repetitive tasks, a robot can do such jobs significantly more quickly than a human worker, they don't need to rest and can work 24 hours.

According to the International Federation of Robotics, Service robots are robots that perform useful tasks for humans or equipment excluding industrial automation application. They can be either personal (e.g. a domestic servant robot, automated wheelchair, personal mobility assist robot, pet exercising robot) or pro-

fessional (used for a commercial task), usually operated by a properly trained operator (e.g. for cleaning public places, to do delivery in offices or hospitals, fire-fighting, rehabilitation or surgery robots in hospitals). In this context, an operator is designated to start, monitor and stop the intended operation of the robot. Although the number of service robots in use is rather small when compared to industrial robots, the market is increasing as the time goes by.

In some cases, service robots consist of a mobile platform on which one or several arms are attached and controlled in the same way as the arms of industrial robots. It should be noted then, that, contrary to their industrial counterparts, service robots do not have to be fully automatic or autonomous. In many cases these machines may even assist a human user or be teleoperated.

Service robots tend to replace humans in some situations, where the humans are promoted to managers of the robot, imagine for instance a night guard that is responsible to verify a perimeter. If there is an autonomous robot to do exactly the same activity and capable to alert the human in unusual events, it could replace the total human in this kind of scenario. Such things are now possible with current technology.

Figure 1.1 Industrial robot (Kuka Robotics), Service Robot (Holos), Humanoid (Honda)

Robotics evolution is considerate the third industrial revolution, it started at the end of the 19th century.

The term Teleoperation (the junction of *tele* - from the Greek, meaning "at a distance" - and operation), i.e. "working at a distance" appears in 1950, half a century later, when the first machine controlled by a human was created.

Teleoperation can be applied to all of the examples of robots given above (it was so on the beginning of the industrial robots and is the standard operation form of most of the service robots). What is then the concept of "at a distance"?

Well, distance can be measured in centimeters, meters or even kilometers. In fact regardless of the considered distance, when someone (lets say the master) is controlling his slave (the robot) we are in a teleoperation scenario.

Everyone has once in a lifetime, played with a remote controlled car or airplane, or watched a movie where someone tries to disarm a bomb using a remote controlled robot. Well these are also good examples of teleoperation.

Imagine a scenario where a fire alarm is triggered, wouldn't be better just by to send a remote controlled robot to the local and confirm this (it may be an hazardous situation)? There are robots specialized to perform such tasks, in dangerous environments, usually they are equipped with cameras that gives a visual feedback a GPS (or other sensors) to know its location, and also additional sensors that may provide further insights to the atmosphere in the location, better than a human could.

The evolution of technology is constantly increasing computational power (even Moore's law[1] that has been predicted to be ending soon keeps being true nowadays). Many may not realize it, but nowadays, on many people's pocket lies a powerful computer that has more computational power that your desktop computer had 5 years ago! When connected to a network, that smartphone or tablet can be used to do unbelievable things, for example control robots, appliances, or surveillance cameras regardless where you are. Forget those big remote, heavy computers and tons of monitoring screens, now it is possible to control anything with that small computer.

1.2 Research Scope

There are currently many technologies based on teleoperation (e.g.telesurgery, telerobotics, telepresence).

Teleoperation started at the Argonne National Laboratory around 1950, where the first master-slave manipulator was developed so that it could handle nuclear materials.

Today's teleoperated systems are much more sophisticated; the operator can actually feel that he is present at the teleoperated environment. This has been made possible since video and force feedbacks were introduced in teleoperation systems.

As every technology teleoperation is not an exception, there are issues and solutions. Most of the time we can't just focus on the solution but instead we can work on improvements that reduce such issues.

Time delay, for instance, is one of the biggest problems in remotely controlled systems, many teleoperation architectures have been developed to "fight" this issue, some of these architectures will be described in detail in further sections.

We have to face the reality; the more we want, more problems we will find, and more complex the new systems will become.

1.3 Aim and Objectives

The objective of this work is to **develop a mobile interface**, based on **Android® OS**, capable to control a mobile service robot from anywhere.

This robot is a prototype of an existing project called ServRobot[1], it will be integrated in another existing project called DVA[2].

Surveillance systems are part of the current mechanisms of the society for its protection against events that endanger people's health and goods. These systems have evolved; becoming less depended of humans and using more sensors and information processing and reasoning systems to detect potentially disrupting events.

In the DVA project (partially sponsored by the European Regional Development Fund and the Portuguese Government) a surveillance system based in geographic position of events and humans agents was developed to improve human-machine cooperation in surveillance. This system reduces the dependence on humans in the detection of events and benefits from the location of the events and of human agents to improve and accelerate the response to events.

Nevertheless, there are some tasks performed by humans that could be delegated to machines, such as: confirmation of events; access to areas dangerous to human health; mobile sensors' information; reconnaissance of areas. To respond to these gaps a partnership was established with the ServRobot project (also partially sponsored by the European Regional Development Fund and the Portuguese Government) in order to integrate the autonomous service robot, developed in this project, as an agent of the DVA system. ServRobot developed a service robot designed specially to be integrated in surveillance systems and it is composed by many types of sensors, gathering information about its environment. The use of this robot as an

[1] http://servrobot.holos.pt
[2] http://dva.holos.pt

agent enables the execution of tasks, hitherto performed only by humans, by the robot and minimizes human intervention in various hostile situations.

This dissertation is a cornerstone on the achievement of that objective, as it provides a mean for a controlling mobile device, connected to the DVA system, to be able to teleoperate the ServRobot.

1.4 Dissertation Structure

In the first chapter of this document a brief introduction to the dissertation's subject is provided.

The second chapter presents the state of the art in teleoperation. Applications, protocols and interfaces used in teleoperation systems are also explained in detail in this section.

Third chapter describes a research approach for this dissertation.

The developed communication protocol will be presented in fourth chapter, followed by development in chapter fifth. The application, prototypes, tests and results are described further in this chapter.

Sixth chapter includes all conclusions, knowledge contributions and future work that can be done in the scope of this dissertation.

----This page was left empty on purpose----

2

2 State of the Art

The purpose of this chapter is to present the current state of the art for the technologies that are core to this dissertation. The focus will be placed on the subjects of Human-Machine interaction, going then further into teleoperation of systems.

An overview of different applications is presented to allow a deep overview of intrinsic and requirements presented by different types of applications.

As remote control is the key issue, the communication protocols used play a key role both on the efficacy and efficiency of the overall system. That is the reason why a sub-chapter is devoted to them.

The final sub-chapter is used to present the development platform chosen and options taken.

2.1. Human-Machine Interaction [HMI]

In the end of XIX century with Industrial Revolution, artisanal methods started to being replaced by new production methods using machines. As the years passed by, the interaction between humans and machines became unavoidable.

Along these years many solutions for the interaction with machines were proposed and developed. The main goals were to reduce the risk of injury, fatigue, error and discomfort. Machines can work 24 hours a week without resting and as well improve productivity and the quality of the products that were previously handmade by humans.

The decrease on the cost of low power microcomputers and electronics, allowed these HMI technologies to start become more widespread and accessible.

There are several types of HMI. Acoustic (sound), Bionics, Optics (Light), Motion and Touch are the most common examples.

Acoustic or sound based technology focuses in voice recognition; it can be used to convert words to text or to provide commands to control certain devices.

Acoustic myography is another interesting area of HMI technology, which is basically measuring the acoustic properties of muscles as they contract[2].

The more a muscle contracts, the louder is the sound. It combines the natural conduction properties of the human body, such as the complex assemblage of bones to increase acoustic distinctiveness, for instance, if someone taps their fingers on the skin, this causes distinct forms of acoustic energy and causes waves to move across the skin, this allows any part of human body to potentially become part of the interface.

Bionic technology is also an impressive area, this area was inspired by scientific fiction, and it uses biological signals from our body to transmit the information. There are researches in this field regarding brain computer interfacing and myoelectric control, like moving a robotic arm just by reading the outputs of our muscles.

Optic based technology usually requires a camera, computer vision it is used to track and detect objects, also gestures to control certain devices. Laser and leds do also fit in this technology, as they are used to read bar codes, fingerprints, and many other applications[3].

Motion and Touch are presently two very used technologies, especially with the fast growth and dependency on smartphones and tablets by people in general.

Each of these devices has sensors capable of detecting motion; the gyroscope or the accelerometers are examples of sensors with these characteristics.

We have to take advantage of what today's technology gives to us. One thing is certain, technologies will begin to converge, devices will combine their functionalities, and new levels of sensor fusions will be created.

Human-Machine Interaction is currently advancing with one purpose, the Human does not have to understand the machine, but at the contrary, the machine must understand what the Human wants!

2.1.1. Dependability for Human-Machine Interaction

Dependability, it is collective term used to describe the performance and availability and its influencing factors: safety, availability, reliability, integrity and maintainability.[4]

Taking into the account of the factors mentioned above, it is easy to understand that there are some issues with respect to dependability.

How do we guarantee, for instance, that a machine is always available? Should we really trust on the given feedback? Is it reliable? These aspects are really important, because we know, that the machines are not perfect, and they are susceptible of technical failures.

The availability of a robot means that it has to be ready to execute a certain task, regardless of when his operator order it to do it.

The robot's reliability it is also a very important concept, it is somehow related with availability. It is expected that an available robot complete its task in a continued and satisfactory manner, this means that at the end of the day the operator must be able to trust the work that was done by his machine.

The absence of catastrophic consequences guarantee humans' safety. In a teleoperation scenario, the human is, most of the time, in a safe environment, moreover, that was why the first teleoperation mechanism was created for, the operator controls his teleoperated device/system at a distance to ensure his safety.

The maintainability of a robot means that it has to be possible to repair or replace faulty components without having to replace those that are still working, which sometimes it is not possible. That is is why systems' integration is still in constant research and development, the machine must be able to be modified over the time or to be repaired in case of failure.

These factors are related among themselves; many of these issues are solved or at least improved, introducing sensors (motion, light, sound, temperature, etc…) in the robot's system.

2.2. Teleoperation Overview

Many people see robots as machines that are human-like; these machines do not necessarily need to have a human appearance. Robots' main goal is replace the human effort, yet they might not perform their functions in the same way humans do, as long as the end result is the same.

Robotics became an area of interest of many people (from individual researchers to large companies). The first teleoperated system was created in the 50's by **Raymond C. Goertz** to handle dangerous materials in the laboratory; this system is represented in Figure 2.1[5].

Figure 2.1 Goertz using the first teleoperated system to handle dangerous materials.

In teleoperation there are two common words to describe whom controls and what is controlled.

This first, the *operator*, is the human who is responsible for controlling the robot. The environment where the operator is is called master's side. There he monitors and handles all the information that goes and comes from the robot.

The *teleoperator* is the machine (robot), which is being controlled[6]. Slave's side, is the given name to teleoperator's environment. The slave receives the orders and executes them, then feedbacks all necessary information to its operator.

Over the past years scientist and engineers have been trying to build a fully autonomous machine.

The word autonomous, which means «existing and functioning as an independent organism»[7], is something that has not yet been fully accomplished in robotics, in engineering the word autonomous is used when a system is functional almost without the human intervention[8].

When the operator is controlling the robot over a large distance, he does not have a visual contact with the slave. Thanks to the cameras installed in the remote side, or even in the robot itself, the operator receives a feedback allowing him to get an easier perception on how things are going in the remote environment.

The control commands are sent via electrical wire or wireless. Regardless of the communication media, delay is always introduced in the system[9].

In the situations where the master is not in the same environment of his slave, a transport delay is introduced in the control system.

Delay is one of the biggest obstacles in teleoperation, and one simply cannot get rid of it. Usually this delay grows proportionally to the distance between the

operator and the teleoperator. However there are other factors that might influence it like network traffic (electrical signals) or weather conditions (radio signals).

Nevertheless, there still exist *mechanical manipulators*, to control the machine the operator uses mechanically or hydraulically mechanisms, in this case both, operator and teleoperator, are present in the same environment and this entails no delay.

There is several control strategies used in telerobotics that can be classified into three classes: *Closed loop control* also known as *direct teleoperation, coordinated teleoperation* and *supervisory control*.

Consider as an example, that some individual is participating in a RC F1 (radio controlled Formula 1), he has a real-time feedback of the remote controlled car, and the operator's requests are being sent directly via radio signals. In this situation the delay is minimal and we are facing what we call closed loop control or direct teleoperation.

On coordinated teleoperation, the master controls the slave, but there is, on the slave's side, some kind of control loop to close the remote control system[8].

However, there is no autonomy at the slaves end. Since there is a considerable time delay in the transmission, sometimes the master may not have full control of the robot's actuators, and the loop needs to be closed in the slave's end to avoid undesirable situations due the time delays.

So far, have been mentioned opened and closed loops. The differences between them are the following:

Open loop: Control systems output is generated based on inputs.

Closed loop: This control system is also called Feedback control systems, the output is taken into consideration and corrections are made based on feedback[10].

As was said above, a totally autonomous system does not yet exist, but when the tasks are performed more or less autonomously we are facing supervisory control, the teleoperator receives high-level commands of his operator.

Usually in supervisory control, part of the control is delegated to the teleoperator. The human intervenes in the system only from time to time, to change the used algorithm when he believes it is necessary. The system processes the instruction and acts autonomously until further changes.

The communication delay does not have the same impact in the systems using this control scheme; since instability is avoided without interfering in feedback loop. This is due the fact that on supervisory control; the operator's action is limited to verify is the assigned task is performing as expected. He/she only tweaks some parameters and the device performs the tasks autonomously. Therefore no

detailed feedback on each individual action is required. The delay is therefore minimal.

2.2.1. Components of a Teleoperation System

Usually a teleoperation system is defined by five elements: The human operator, a master manipulator, a communication channel (Middleware), the slave and the environment.

Figure 2.2, represents the block system of a common teleoperation system.

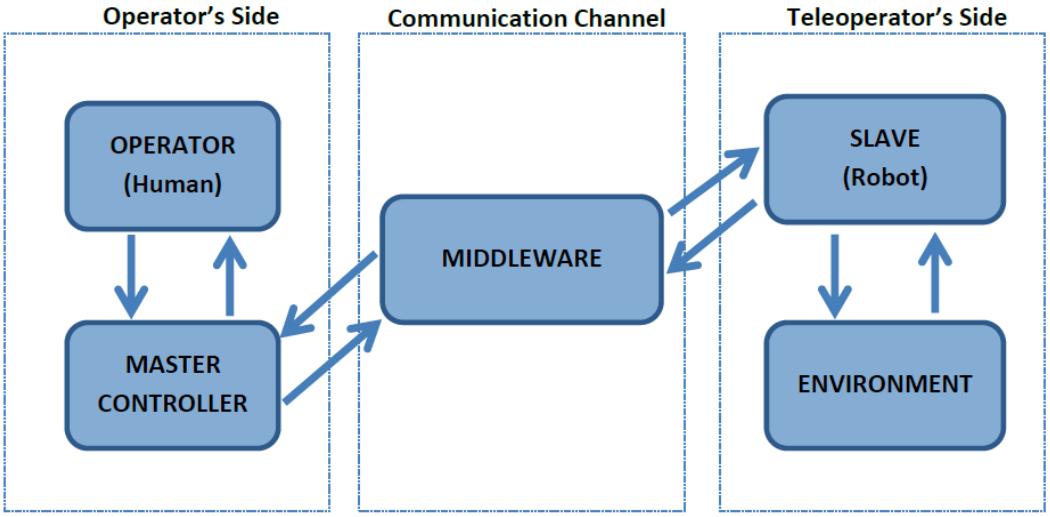

Figure 2.2 Representation of a Teleoperation system.

The operator communicates with the whole system using a master controller, which is most of the time a haptic device (see definition below), the orders pass through the communication channel until they reach the robot, in teleoperator's side, the salve gets information and interacts with the environment using its sensors or manipulators.

2.2.2. Haptic Devices

Haptic devices are usually some kinds of joysticks, these devices are used to control the teleoperator, influencing its position, in consequence the operator feels a reaction from the teleoperator, this is called force feedback.

This sensory information created by the remote side is very important features of telerobotics systems.

To further understand the concept, lets look at some examples. Suppose someone is controlling a mobile device using a smartphone, when the user presses the screen in the breaks button, the command is sent to the device and it takes an

action, as a reaction to that order in the operator side the smartphone may vibrate giving this sense feedback.

Another example is when someone is controlling a rover using a steering wheel, while turning left or right the wheel may create an opposite force to the operator's action or trembling, similar to what would happen if it were a real car.

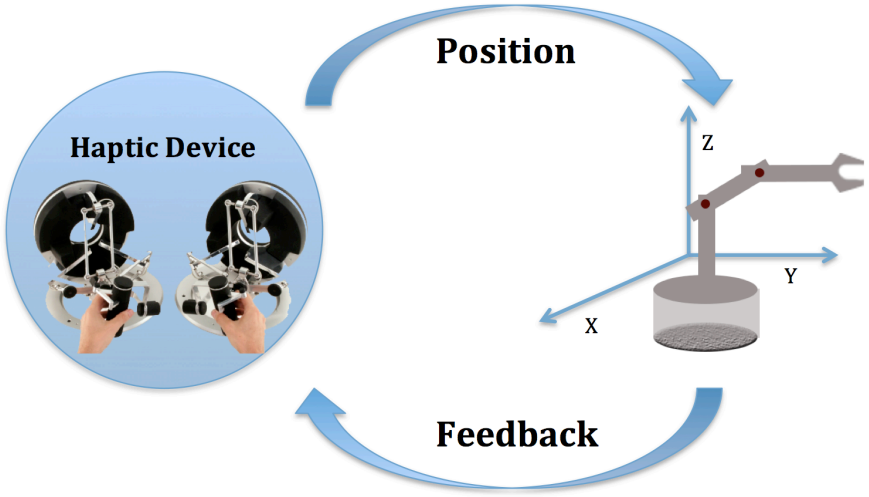

Figure 2.3 Example of Teleoperation using a haptic device.

2.2.3. Application Programming Interface

An application programming interface, common known as API, it is a conglomerate of programmed functions and it serves as an interface between the devices or systems. This API may be created in several programming languages.

Later in this thesis will be discussed the API and the communication protocol that was created.

2.3. Applications

Teleoperation can be applied to the most diversified fields, from the air to the sea; there are many kinds of applications to this technology.

Below in this chapter will be presented some applications covered by this technology.

2.3.1. Space Applications

Nowadays humans cannot live without satellites anymore. They are extremely important to global position systems, weather forecast and communications.

The first satellite sent to outer space was *Sputnik 1*, in 1957 by Soviet Union.

This was one of the first steps regarding space applications. In the present more sophisticated technologies are being used by the well-known spatial agencies.

With the evolution of telerobotics, uninhabited planets may now be explored without a need of a human crew.

NASA (National Aeronautics Spatial Agency) has sent two rovers to Mars in 2003. The mission was named Mars Exploration Rovers[11].

In the other hand, ESA (European Spatial Agency) with the cooperation of Roscosmos has two mission scheduled, one consisting of an Orbiter plus an Entry, Descent and Landing Demonstrator Module, to be launched in 2016, and the other, featuring a rover, with a launch date of 2018, these missions are integrated in Exo-Mars programme[12].

With the evolution of technology one day we may be expanding human race to other planets living as we were on Earth (this is the humble and optimistic opinion of the writer).

2.3.2. Military and Security Applications

The army invests large amount of money in military teleoperated vehicles and telerobotics. To collect information from the enemy, soldiers are exposed to hostile and very dangerous environments most of the time, therefore is better to send an unmanned vehicle (UV) to these reckon missions, human life is priceless.

«In the 1990s, along with the development of sensors and GPS technology, the themes for research in mobile robotics and in the emerging military Unmanned Ground Vehicles (UGVs) domain were dominated by positioning issues and experiments with various control architectures»[13].

These UGV are very useful on route clearing and land-mine detection.

«Unmanned Aerial Vehicles (UAVs) are well-recognized for their promises to achieve many powerful applications with reduced cost/danger associated with the absence of human pilots onboard: e.g., pesticide-spraying, landscape survey, entertainment/games, ad-hoc communication network, and surveillance/reconnaissance»[14].

US Air Force Predator is an example of a modern UAV, it is remotely piloted by radio or satellite with help of GPS it can fly autonomously.

2.3.3. Marine Applications

Marine remotely operated vehicles also known as ROVs are used to operate under dangerous underwater scenarios. ROVs are commonly used to repair off-shore oil platforms, but there are other applications, surveying, inspecting and very useful on oceanography.

A ROV is most of the time connected to the surface using an umbilical cable, which provides bilateral communication.

Operating a ROV brings more advantages; on long explorations better results are obtained instead of using divers to do the same job. Also these vehicles may dive under contaminated waters, in those cases where underwater environment is deemed too dangerous to human's health.

Hydrographic Institute from Portugal has Phatom S2 since 1985. This ROV is used to observe environments where there are sunken ships and aircrafts. Also to provide assistance to divers in their scientific investigations in the deep sea[15].

Ziphius is another example of a remote operated vehicle, but this drone is being commercialized as an entertainment product.

This is the first app-controlled aquatic drone, it was created by Azorean, a spin-off company of YDreams, it that plays augmented reality games and shows autonomous behaviors[16].

2.3.4. Forestry and Mining Applications

The main goal of these applications is related with exploration, especially on unknown environments such as forests and old mines. The robots working on these tasks must collect information of the place and take actions working almost autonomously.

Rescue robots do also fit on these kinds of applications, for instance, after an earthquake rescue teams must have a fast response and use these robots to check where humans are trapped.

2.3.5. Telesurgery

The point of this technology is not to replace doctors or surgeons but to make the machines able to assist them while their treating a patient.

Unlike humans, robots do not get tired, which is a big advantage on long surgeries, also robots are more precise and do not tremble.

Distinct robotic arms that are being controlled by the surgeon either at kilometers of distance or just a few meters of the patient generally compose these sur-

gical systems. If it is a surgery made at a considerable distance, on the remote side there are always a team to assist the patient in case of failure.

Telesurgeries are less evasive because surgeons operate through just a few small incisions[17].

2.3.6. Telepresence

Telepresence is part of teleoperation nowadays, the fact of the operator have a video and audio feedback while is operating a remote robot, make him to feel like he is present in the remote side. In some cases the operator may affect the remote location, his position, movements, actions, voice etc. may be sensed in the remote side.

Other example of telepresence is videoconference also known as teleconference, when people cannot reunite in one single place they may opt to meet thus this way.

2.4. Communication Protocols

In teleoperation systems we may consider two kinds of transmitted data between master and slave.

There are two common protocols used nowadays: TCP and UDP.

Transmission Control Protocol (TCP) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters.

Since TCP requires a reliable transmission, it means acknowledges must be sent to the both ends it costs retransmission and long timeouts, which is bad for real time teleoperation.

In the other hand is User Datagram Protocol (UDP), does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

UDP do not require any kind of acknowledge eliminating unnecessary waiting time, which seems to be better for real time applications such as teleoperation[18].

Researches have realized that TCP is not proper for teleoperation system, most of teleoperation systems employ UDP directly as their transport protocol[19].

2.4.1. The Trinomial Method

This is a rate-based protocol; it manages network congestion by adjusting the inter-packet gap (IPG) instead of the window size schema used by TCP.

This protocol controls the number of datagrams per second, according to the available bandwidth. Trinomial method is able to adapt itself to network congestion and the available bandwidth without affecting much the way the user teleoperates the robot.

Sending curve of this protocol is quite smooth and makes better use of the available bandwidth, thus obtaining excellent efficiency as compared to UPD or TCP protocols[20].

2.4.2. Real-Time Network Protocol

This protocol was specially created for a bilateral teleoperation. In teleoperation system, using this protocol, time delay can be produced through network device performance, end-to-end congestion control algorithms or implementation of the network stack in the host side.

This is a protocol that uses identification in the UPD/TCP headers as a means of informing real-time operating systems that the received packet fits the category of "real time," thereby giving maximum priority status to packets passing to application levels.

RTNP demonstrates that the overall time delay between the client and the server depends not only on the network but also on the software provided by the operating system[21].

2.4.3. Interactive Protocol

This is an IP-based protocol that takes advantage of both TCP and UDP protocols in order to improve responses in teleoperation systems.

It is a connection-oriented protocol that implements congestion and error control. In order to enhance efficiency, IRTP (Interactive Real-Time Protocol) protocol simplifies packet headers as much as possible so as to obtain a better relationship between the data sent by the application level (layer 7 of the OSI Model) and the protocol control information.

In addition, IRTP reconfigures itself so as to send two basic kinds of data which are present in a network control system: crucial data and real-time data.

The IRTP protocol uses the same control congestion algorithm as the trinomial method, wherein the IPG is decreased or maintained, but never increased.

Particularly, the trinomial control congestion algorithm detects congestion and maintains transmission rates in order to avoid more congestion.

However, the only means it uses for decreasing the transmission rate is through detection of a lost packet, a situation that is normally produced when the network is already congested[22].

2.4.4. Bilateral Transport Protocol

This paper[23] presents a new teleoperation protocol, called Bilateral Transport Protocol (BTP).

BTP is an end-to-end congestion control protocol.

The main goal of this protocol is to minimize the round trip time (RTT) while maximizing the transmission frequency. In order to do it, congestion control must be performed avoiding congestion signals.

Some consideration needs to be taken into account while designing a new communication protocol for teleoperation.

RTT in teleoperation is equivalent to the time spent sending an instruction from the master to the slave, therefore maximizing RTT is very important.

If the consumed bandwidth is increased, then congestion starts to be detected on the network, as consequence it increases RTT. A correct sending rate has to be determined in order to minimize RTT.

In teleoperation communications, two data flows must be taken in consideration, the one who goes from master to the slave, and the other with feedback information, from the slave to the master.

Each flow may have a different bandwidth, since they work in different directions.

In order to guarantee overall Internet performance, BTP separates these two flows, and follow TCP-friendly criteria, which ensure that TCP flows share the available network bandwidth in a friendly way.

Working at a minimum inter-arrival time (IAT) is highly recommendable.

Haptic devices with a touching sense require high frequency in their protocol loop. Minimizing IAT is somehow related to minimizing IPG, while there is not congestion present in the network.

For teleoperation, the rate-based approach is more convenient since packets are injected into the network in a smoother way.

This control system is executed via the periodic injection of packets into the network.

2.5. *Android Operating System*

Google® is leader in many markets; in 2005 in order to expand their business through the world of mobile devices Google has bought Android Inc. a small company from California.

Android is an Operating System (OS) based on Linux kernel, designed for mobile devices. It is an open source that made many manufacturers to adopt this new OS to their machines. Android's system architecture is represented bellow in Figure 2.4.

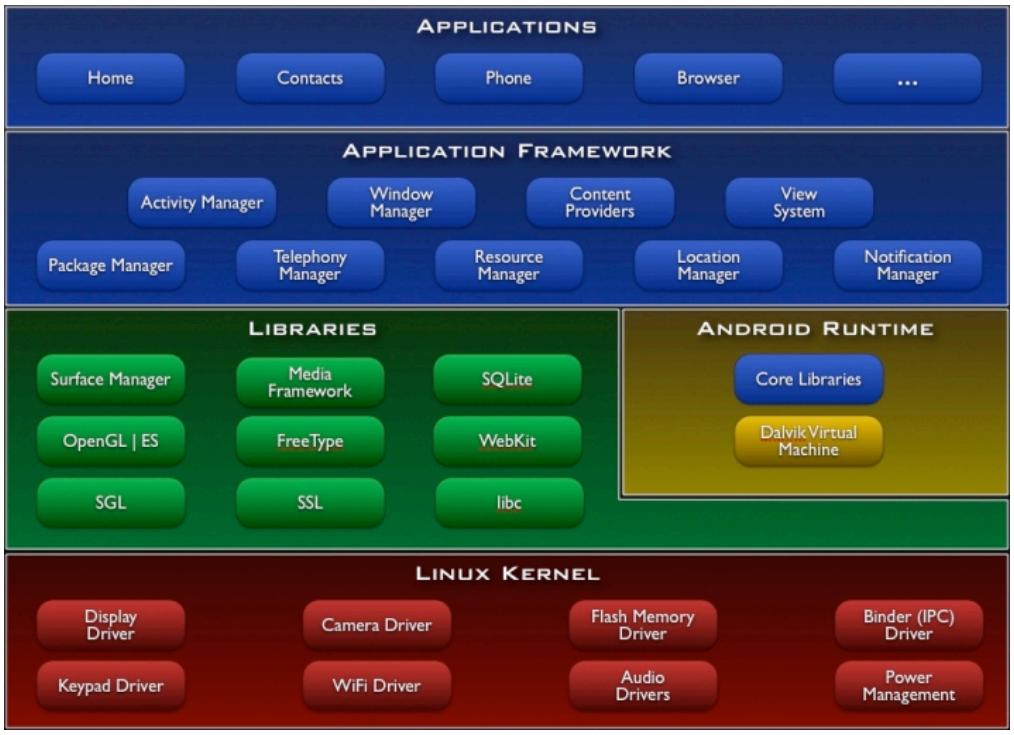

Figure 2.4 Android System's architecture[24].

In the present Android is leader in the market of mobile devices, in second comes iOS from Apple®, the third place is owned by Windows Phone from Microsoft®. Figure 2.5 depicts what as been said above.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q2 2014	84.7%	11.7%	2.5%	0.5%	0.7%
Q2 2013	79.6%	13.0%	3.4%	2.8%	1.2%
Q2 2012	69.3%	16.6%	3.1%	4.9%	6.1%
Q2 2011	36.1%	18.3%	1.2%	13.6%	30.8%

Figure 2.5. Mobile operating systems market share based on unit shipments.[25]

All the information below is taken from the official website[26], can be accessed by everyone.

2.5.1. Application Fundamentals

To develop applications is necessary to have some knowledge on programming, especially in Java.

The Android SDK (Software Development Kit) has the tools to compile the developer's code, along with any data and resource files. After compilation it is generated a file with an extension *.apk* - Android Package.

Like other Linux Systems, in Android OS each application has their own user ID that means each application is seen as a different user.

The permissions are given to one app based on its ID, this identification is only known by the System itself.

Each application has one or more Linux processes associated with it.

When one app is launched, Android starts the processes and finishes them when they are no longer needed or when the system must recovery memory.

In order to provide a secure environment Android Systems implement the principle of least privilege, this means each app, by default, can only access to the components that it requires to work, the app may only access to other part of the system if the permission is given.

2.5.2. Application Components

Depending on the objective, Android app components can be distinguished in four different types: Activities, Services, Content providers and broadcast receivers.

Activities: An activity is a single screen with a user interface. Consider a game consisting in several buttons. When a button is pressed it goes to another screen, meaning it migrated from one activity to another.

Services: A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. Imagine that some individual is listening to his/her YouTube® playlist, then the application is closed and another is launched, it is possible to keep listening to the music without blocking another app, to do so it is need to use a specific type of service.

Content Provider: A *content provider* manages a shared set of app data. You can store the data in the file system, a SQLite database, on the web, or any other persistent storage location your app can access. The easiest way to understand this

kind of component is giving this example: The user downloads Facebook® App from Google Play. He is asked if want to synchronize his contacts with his Facebook friends, name, profile pictures and so on. Through the content provider, Facebook can query or even modify the data from Contacts (if the content provider allows it).

Broadcast Receivers: Broadcast receiver is a component that responds to system wide broadcast announcements. There are many broadcasts originated by the system, low battery, screen turned off or even a captured picture, are examples of broadcasts. Also applications can initiate a broadcast. Consider that someone wants to watch a movie on his/her tablet, the website does not have any embedded media player compatible with the device, through a broadcast the application (in this case the browser) ask if there is any other application in the system capable to reproduce that video.

----This page was left empty on purpose----

3

3 Research Approach

In current market scenarios, where changes occur quickly, small projections for the future are hardly made. Teleoperated systems are now part of people's life, at work, at home, in every environment; it is possible to identify a remote system or at least a potential system that may be turned into a remote one.

This work was encouraged by the fast development especially of robotics but also by every technology in general.

In order to understand the goals for this research it is important to analyze the current state of the art in Teleoperation.

There are many possible ways to control a robot, using a computer for instance, a traditional radio remote control or any other haptic device.

In this masters dissertation we will take advantage of what a smartphone can give us, since it can be at the same time a haptic device, a computer and a telephone.

3.1 Problem Definition

In two distinct research projects, lead by Holos[1], and developed together with the Universidade Nova de Lisboa[2] and the Uninova[3] institute, two distinct products were developed:

- ServRobot - An autonomous mobile service robot for transport and surveillance missions.
- DVA - A fully distributed surveillance system based on geo-location and artificial intelligence for event detection, following and resource assignment.

The goal underlying this dissertation was to take advantage of both systems and to allow the robot to become an additional assignable resource that could be remotely teleoperated, in situations where the presence of a human may prove dangerous.

As most of the human resources in charge of the surveillance tasks are mobile, the robot control should also be made using a mobile device.

The task at hand was the development of an application that, integrated in the DVA system would allow the ServRobot to be remotely controlled using a mobile phone or a tablet.

This task however faces several problems. The robot is not prepared to be teleoperated, outside the line of view of the operator; the ServRobot is not in any way integrated with the DVA system, there is no application in DVA with the specific requirements that this one poses (short fast control messages and heavy visual feedback).

The underlying messaging structure of DVA, fully dependent on the JADE[4] platform might be inadequate for the message types that the task requires.

[1] http://www.holos.pt
[2] http://www.dee.fct.unl.pt
[3] http://rics.uninova.pt
[4] http://jade.tilab.com

So far, for this dissertation context, we have two different systems. They cannot communicate between them, but both provide advantages. However, by interconnecting both systems the overall functionality would increase significantly.

3.1.1 Requirements

The proposed communication protocol has to consider both systems; it will exchange information among them, but it is the protocol that must be adjusted to the systems and not the opposite.

Systems have their specific hardware and software that should not be changed by the fact we intend to integrate them. The protocol has to be designed in order to be adapted to such requirements.

To make sure it will work correctly on the real devices, some simulators had to be developed. This is required to predict the behavior of the real devices and as a safety measure, considering the cost of the system we will be teleoperating; also delays need to be assessed as we are talking about a teleoperation. Everything has to be very lightweight to improve the performance and minimize the latency.

The mobile application must respect DVA's context and design; it will be developed in Android OS, since it will work as a new module of DVA's application.

3.1.2 Definition of the Research Objectives

The main goal of this work will be to develop a mobile application capable of to be used for teleoperate a service robot (ServRobot) in the context of the integration of services between the ServRobot and the DVA surveillance system. The work will consider the development of a communication protocol between the systems to allow them to interoperate, as well as the requirement definition and implementation of the mobile app, fully integrated within the DVA infrastructure.

Within this main objective, it expected that the researcher not have only the working and functional application, but also some additional knowledge about robotics, telerobotics and telecommunications.

3.2 Research Methodology

The developed efforts referred in this thesis were originated by first understanding the basics concerning the specific topic this work are about. It started by a deep literature review about the domain and an analysis of the state of the art around it. When the basis had been understood, gaps and needs could then be identified, which leads to a problem.

Having a problem defined, a set of requirements and objectives could then be formulated. This allowed for the research hypothesis to be created.

The problem for this thesis is the lack of communication between the DVA's and the ServRobot's systems also the impossibility of teleoperate the ServRobot from a smartphone that uses DVA's mobile application.

For this, the integration of these two systems will create a whole new system architecture leaving behind the concept of two independent systems, and transforming it into two collaborative interoperable sub-systems.

After the research hypothesis had been establish, an user of the DVA's application should be able to check events that are triggered and teleoperate ServRobot to the location of the events. To finalize this work, a review about it has been made, where conclusions and future work have been established.

4

4 Communication Protocol

Inter-communication between two different systems is something that is very common nowadays. Regardless how this systems work, it is supposed that they can exchange messages and handle them on their own way but, at the end of the day, the way how these messages are exchanged must be transparent for the both ends.

As mentioned early in the introductory chapter, this master thesis covers two existing systems, the DVA and ServRobot.

Up to now both projects worked independently, so there was no communication between them.

DVA is a very robust surveillance system that handles many kinds of events, there is some autonomy but there is also the need of the human intervention from time to time.

ServRobot is service robot that has sets of sensors and some intelligence on it. To benefit of these characteristics, ServRobot will be integrated into DVA's system, acting as one more active agent.

A new communication protocol had to be developed in order to establish a proper communication between ServRobot and DVA, the details about the developed work will be discussed below in this chapter.

4.1 Architecture and Protocol Overview

To use the autonomous robot, developed in ServRobot's project, as a new agent in the DVA's surveillance system, it is necessary to define the architecture and a communication protocol. The integration of the robot in DVA's system could

provide new capabilities to this system, such as: send the robot to execute a mission; teleoperate the robot; or get robot's sensory information.

The integration architecture developed, depicted in Figure 4.1, has three primary nodes, **DVA** as the core node of the architecture, **ServRobot** representing the robot and the **Client** representing others devices that can interact with the robot.

The **DVA** node refers to the DVA's surveillance system. As the core of the architecture, it handles requests regarding registration messages, sensor and device list queries. Authentication and permission level requests are also managed by DVA.

The **ServRobot** node represents the robot as an operable device. The robot must first request a registration to DVA. DVA or other devices can subscribe ServRobot's sensors. Subscribing a sensor allows receiving its output's values updated ata specific frequency. It is also possible for the DVA to request execution of missions or remote control of the ServRobot. The robot can be in three different statuses: Idle, Mission, Teleoperation.

This master thesis is focused in mobile teleoperation of the robot through the DVA system, however, during it was also considered the hypothesis for the operator to subscribe to a certain sensor or to keep aware of the robot status periodically.

This architecture was projected to be scalable, allowing the integration with other systems (multiple robots and multiple teleoperation devices). The **Client** node represents a mobile device that interacts directly with a ServRobot by teleoperating it as a remote control.

A permission level was defined to control accesses between system' nodes. This permission level allows the definition of authorization to subscribe to a limited set of the sensors of the robot.

Depending on permission level, one client may only be allowed to do teleoperation, missions, or both. There are different classes of permission levels; they limit the client's freedom, regarding the actions it can perform.

The communication between ServRobot and DVA/Clients can be synchronous or asynchronous depending on what is being requested. For this master thesis purposes, in a teleoperation scenario it is clearly a synchronous communication. The orders or commands sent to the robot must be acknowledged in real time.

In the other hand, if the mobile device wants to keep posted about ServRobot's sensors it may subscribe them, if this is the case, the communications is asynchronous as the results came in a specific time interval.

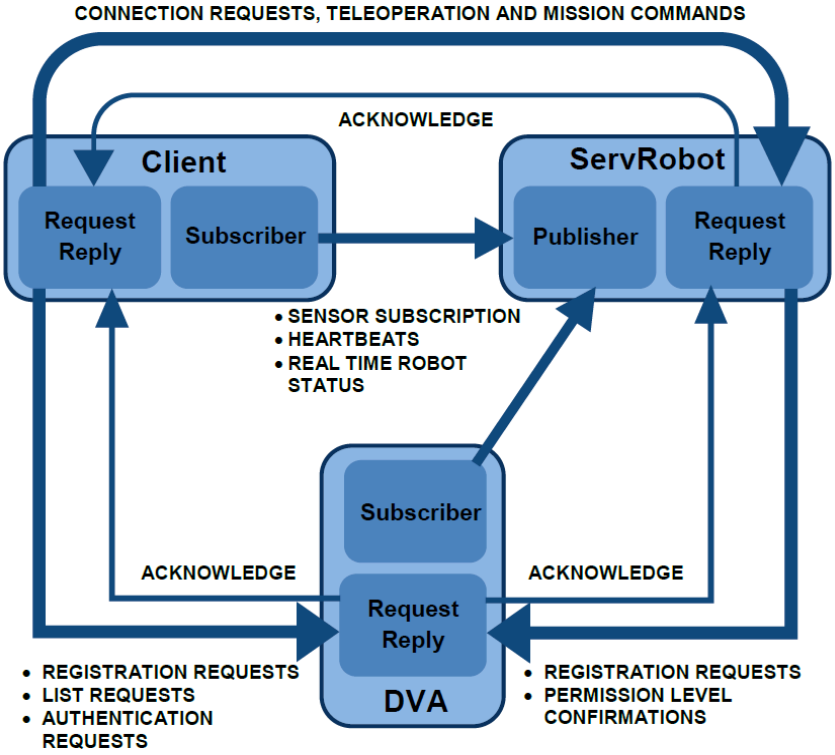

Figure 4.1 Designed Architecture

4.2 Messages' Structure

The messages exchanged using this protocol are **eXtensible Markup Language** (XML) based messages. This markup language defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Every message in the protocol is initiated with **<msg>** as its root tag, the next tag (msg's child tag) represents the message type, and it can have the following types: Emergency Message, Heartbeat Message, Registration Message, or Simple Message[27].

Emergency Message: This message type has the highest priority compared to the other types; it is used when an emergency stop it is needed.

Heartbeat Message: This message type it is used to keep devices aware of their connection between both ends, if a message of this type is absent after a defined time interval, it may indicate connection problems.

Registration Messages: This message type is used when a device wants to register in the system, a registration query is sent to the core of the system (**DVA**).

Simple Message: This message type covers the majority of the messages exchanged using the protocol, it is used in standard operations, such as missions, teleoperation and sensor messages.

From the types presented above, only simple message has different child tags, depending on the mode that is being used.

The XML message it is not complete yet, every message type has its own child tags, and the following tags are common to every message: **Destination Identification, Message identification, Source Identification, Timestamp and Session Identification** (with the exception of Heartbeat messages)**.**

Every message of the type **Simple Message** has also a child tag representing the data type that is being sent. The data types are divided in: **Robot Status** (<RobotStatus>), **Mission** (<Mission>), **Teleoperation** (<TeleOp>), **Sensor** (<Sensor>), **Device Subscription** (<DevSub>) and **Reply** (<reply>).

For the focus of this thesis, only the following data types are relevant: Teleoperation and device subscriptions.

Teleoperation is used in those cases where a client wants to control a certain robot, and device subscription is used if the client wants to keep posted of the outputs of a certain sensor in the robot's system.

Figure 4.2 depicts an example of an emergency message. Note that this message type does not require any command, when the destination receives a message like this, stops everything and stands idle until further orders.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg>
    <EmergencyMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
        <SessionID>1425</SessionID>
    </EmergencyMsg>
</msg>
```

Figure 4.2 Example of an Emergency Message

Figure 4.3 presents an example of a heartbeat message, this is the shortest message exchanged using this protocol.

Note that **Session ID** is not present, this kind of message does not require any extra security, and it is needed only to aware about the connection status of a certain device.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg>
    <HBMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
    </HBMsg>
</msg>
```

Figure 4.3 Example of a Heartbeat Message

Figure 4.4 shows an example of a registration message; every device that wants to interact with the system must first register.

These messages are sent directly to the core of the system, **DVA,** it handles and manages the registrations, and also the permission levels.

```xml
<?xml version="1.0" encoding="UTF-8"?
<msg>
    <RegMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
        <SessionID>1425</SessionID>
        <DeviceLabel>Tablet BQ Curie 2</DeviceLabel>
        <DeviceType>TeleOp</DeviceType>
        <IpAddr>192.168.1.48</IpAddr>
    </RegMsg>
</msg>
```

Figure 4.4 Example of a Registration Message

Next, three examples of messages that are used in teleoperation mode will be presented.

Assuming that the client and the robot are already registered in the system the client (mobile device) must request, to his future slave (the robot) to establish a connection, Figure 4.5 depicts an example of the start teleoperation request.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg>
    <SimpleMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
        <SessionID>1425</SessionID>
        <TeleOp>
            <StartTeleOp></StartTeleOp>
        </TeleOp>
    </SimpleMsg>
</msg>
```

Figure 4.5 Example of Start teleoperation query.

After the connection being established, the client can start to send orders to his slave, such orders set a certain speed (in m/s) and turn rate (in radians).

In Figure 4.6 is presented an example of the structure of the message that is sent to the robot.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg>
    <SimpleMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
        <SessionID>1425</SessionID>
        <TeleOp>
            <SetTeleOp>
                <Speed>0.5</Speed>
                <TurnRate>0.56</TurnRate>
            </SetTeleOp>
        </TeleOp>
    </SimpleMsg>
</msg>
```

Figure 4.6 Example of a Teleoperation command.

When the client wants no longer to keep teleoperating the robot, it must close the connection to his slave, releasing the connection. This allows the robot to be used in other operation modes or by other devices.

To stop teleoperation the client must send a message similar to that presented below in Figure 4.7.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg>
    <SimpleMsg>
        <DestinationID>70:1A:04:F9:81:D6</DestinationID>
        <MessageID>124</MessageID>
        <SourceID>56:2B:04:E4:56:D8</SourceID>
        <Timestamp>12:30:56</Timestamp>
        <SessionID>1425</SessionID>
        <TeleOp>
            <StopTeleOp></StopTeleOp>
        </TeleOp>
    </SimpleMsg>
</msg>
```

Figure 4.7. Example of a Stop Teleoperation message.

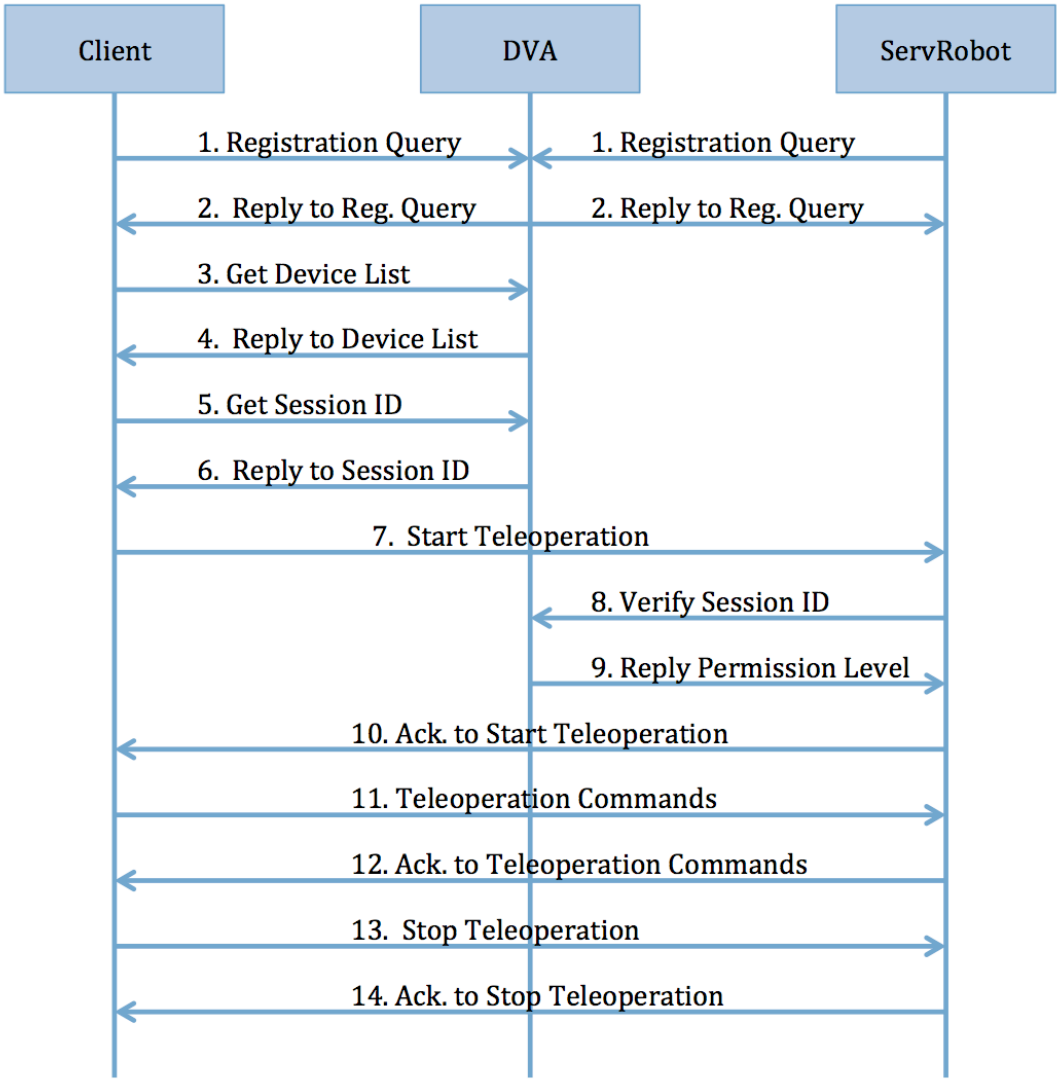

Figure 4.8. Diagram of the messages exchanged during a teleoperation session.

In the diagram presented above in Figure 4.8, it is presented the message flow that is exchanged since the client registration until the release of the connection with his slave.

4.3 The chosen middleware

The middleware that has been chosen to exchange message was **Zero Message Queue** also known by **ZeroMQ** , **ZMQ or** ØMQ.[28]

ZMQ is a high-performance asynchronous messaging library aimed at use in scalable distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ØMQ system can run without a dedicated message broker[29].

ZMQ has several patterns and various combinations between its socket types, however in this master thesis, only **Request-Reply**[30] and **Publish-Subscribe**[31] were used.

Request-Reply was used in the most cases, every time a teleoperation message is sent, this pattern is in used, in other words, on every sent request to the robot (e.g. set teleoperation speed and turn rate) a reply is expected, an error is reported if a reply is not received within a timeout.

Publish-Subscribe patter is used in some cases (e.g. heartbeats or periodic sensor update), every time a client wishes to be aware of the robot status, it must subscribe a pair "ip:port" of the slave in order to receive that information in an agreed time frequency.

4.4 Message Handling

At the first stage, some tests have been run in order to test the communication protocol, before the implementation of the Android application two simple **Java applications** were created, one to send messages and other to receive. In both cases messages were validated against a XML schema using **Xerces**[32].

Xerces is Apache's collection library. It was included in the project because it was useful to have a **Simple API for XML** parser (SAX parser), since the exchanged messages have a XML format, with Xerces it is easier to parse, validate, serialize and manipulate those messages.

Another feature, a **Java Architecture for XML Binding** (JAXB) were used to marshal and unmarshal the exchanged messages. Marshelling is similar to a serialization, it is the process of translating data object into a format that can be stored,

and when sent, e.g. over the network, it can be later reconstructed in its original state or another, depending on the computer environment.

The advantages of using XML Schema and JAXB is that a set of classes that perform otherwise difficult tasks is generated automatically, saving some development work and time.

At the end of the first stage, after a few tests were run, the conclusion was good, the message protocol was working flawlessly.

The real problems came when the developed java code had to migrate to the Android application. Unfortunately JAXB is not included in Android by default, to add it will cost much space in the application.

Also Xerces library had to be tricky before it works in Android Dalvik's virtual machine, after a long search, was found a **Xerces-for-Android** library, that same library was downloaded [33] and compiled into a .**jar** file.

Having the Xerces working on Android (which was good, because at least the validation part would work without doing it manually), the marshaling and unmarshaling was made using **DOM for Java** (dom4j) library[34].

Summary, migrating the java code developed in the first stage to the Android application caused some problems, but at the end everything was solved.

----This page was left empty on purpose----

5

5 Development

The application's main purpose is to allow a human security agent of DVA, through a smartphone or a tablet (running the DVA app), to be able to control the ServRobot remotely using his mobile device's accelerometers. This application is being developed under Android OS, and for that reason, the programming language used for the development is Java.

There are some reasons why Android OS was chosen at the first place, first it is open source and has lot of support, the second reason is that DVA is also written in Java, and that makes easier to integrate both apps.

This application could run separated of the DVA's app, but some issues would come up, especially at permission and authorization levels. For these reasons, the app will be integrated in the DVA's app as a new module (activity).

Figure 5.1, shows an illustration of the possibilities of the application usage.

The DVA mobile app has already a robust authentication and login system, and since the developed protocol (described in chapter 4) needs to deal with privileges it would be an advantage to use the already implemented login system to set privileges for each user.

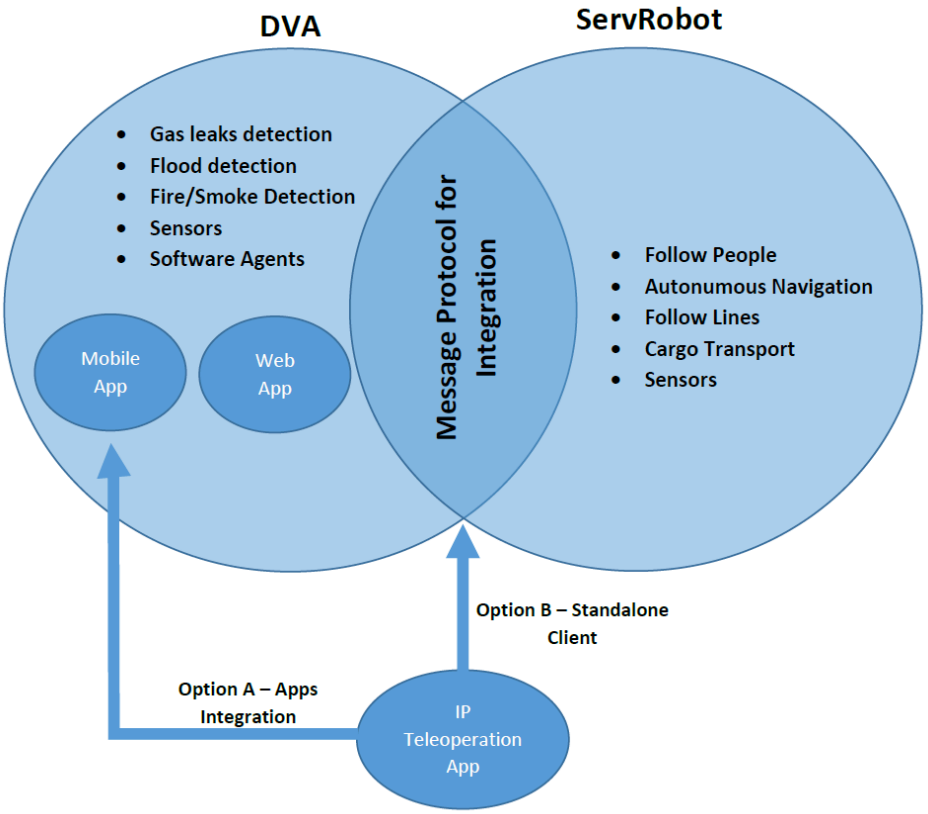

Figure 5.1. Possible options for the developed mobile application

5.1 Developed Application

Just to review the concepts, an activity, as explained in the first chapter, is one active screen of the application. This application has five distinct activities: **Main Screen**, **Calibration Activity**, **Authentication Activity**, **Instructions Activity** and **Teleoperation Activity.**

We must not misunderstand Authentication Activity with Login Activity from DVA's app. Next during this chapter each activity will be presented in detail.

5.1.1 Main Screen

There is not much to tell about this activity, it can be seen as the "main menu", it is composed by four buttons and has no complex code behind it. Each button links to another activity, Figure 5.2, depicts Main Screen activity.

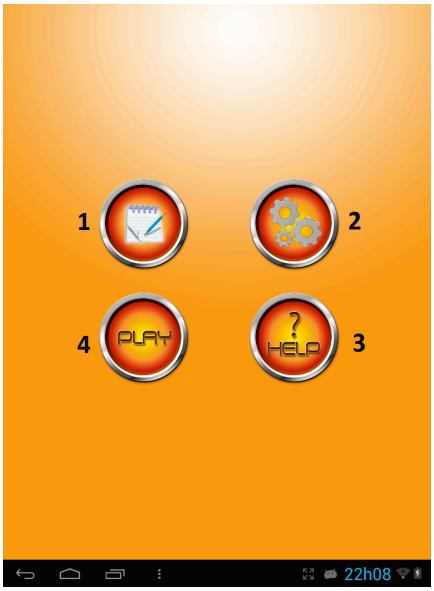

Figure 5.2. Main Screen Activity

Buttons 1, 2, 3, 4 open Authentication Activity, Calibration Activity, Instructions Activity and Teleoperation Activity respectively.

5.1.2 Instructions Activity

The purpose of the Instructions activity is to explain to the user how to use the application.

Starting with the authentication process, is explained in detail how to successfully authenticate a device, in the next steps is explained how to calibrate and control a teleoperator. Figure 5.3 depicts the activity mentioned above.

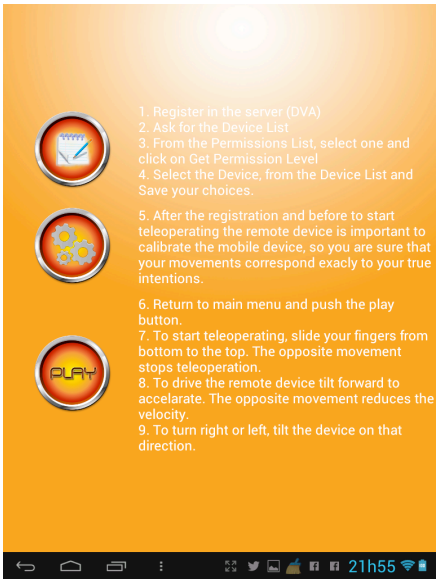

Figure 5.3. Instructions Activity

5.1.3 Authentication Activity

As it was presented in the introduction of this chapter, this application was developed and prepared to be used either as a standalone client, or integrated in the existing DVA's application. Therefore, using this application as standalone client the user has to request for a registration.

To proceed with the registration request, there are several fields and options that must be filled before pressing the registration button.

- Mandatory Fields:
- Server IP address and port;
- Device Label, can be any text, it describes the device (e.g. BQ Curie 2).
- Mandatory Options:
- Device Type (Robot, Remote Control or Teleoperation).

With all fields correctly filled, by using the library DOM for Java, the application generates a XML string.

Before sending this string via ZMQ socket, it is tested against a XSD schema, if valid then the request is sent, if not an exception is thrown.

The application is made for this exception never occur, however is a good practice to test every XML message (sent and received) against the XSD schema in use.

After the registration is completed (with app integration this part is skipped) the user is allowed to ask for the device list.

When the reply with the device list is received, the childs inside a <Device> tag is read and a string "deviceType:deviceLabel@ipAddr(IdTag)" is created and put into List as shown in Figure 5.4.

```
while (j < list.size()) {
    Element elem = (Element) list.get(j);
    Node node = elem.selectSingleNode("IdTag");
    String idTag = node.getText();
    node = elem.selectSingleNode("DeviceLabel");
    String deviceLabel = node.getText();
    node = elem.selectSingleNode("DeviceType");
    String deviceType = node.getText();
    node = elem.selectSingleNode("IpAddr");
    String ipAddr = node.getText();
    dev_list.add(deviceType
            + " = " + deviceLabel
            + "@" + ipAddr + "(" + idTag + ")");
    j++;
}
```

Figure 5.4. While cycle to read childs inside a <Device> tag.

Posteriorly, this **List dev_list** is shown in a dropdown menu in the application, so that the user may choose which device he wants to interact with.

The choices are saved in the Android's Shared Preferences to be loaded later when the user starts a communication with the specific device.

Yet, the authentication process is not complete. The user must specify which permission level he needs.

Before requesting the permissions the user must select them in the dropdown menu and press the Get Permission Level button.

Figure 5.5, presents the layout of the authentication activity.

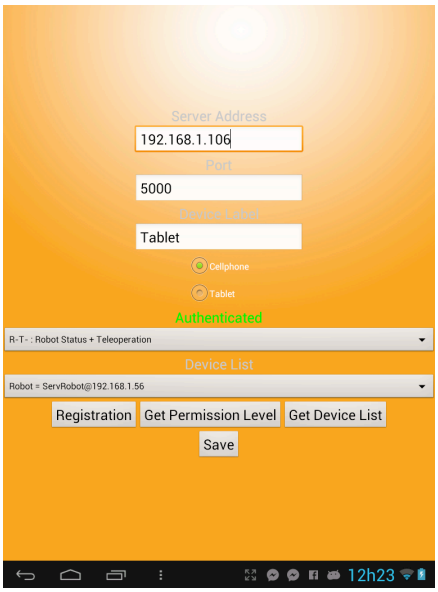

Figure 5.5. Authentication Activity Layout.

5.1.4 Calibration Activity

This activity is one of the most important activities of the application. It is really important to calibrate the device before start a teleoperation session because depending of the reference point the behavior of the controlled device may not be the same.

For example, if someone is laid down the accelerometer has a certain values, the values are not the same if the individual is standing in a vertical position, it is therefore necessary to set the reference x, y, z points.

The accelerometer has a very bad behavior; the reason is that even if the device is standing still on the table, the accelerometer's coordinates change their values.

In this activity the user may test if the values made by his movements pleases him. To calibrate the user may change the force speed, by force it means if the de-

vice is moved forward the speed step changes in a soft way (e.g. not from 0 to maximum speed just by moving the device 1 mm forward).

The same can be done regarding the radius step (e.g. turn rate angle) and maximum delay between the sent messages to the teleoperator.

At the beginning the maximum delay was not taken into account, however the development has gone further, this delay became relevant. It's useless to send messages every 100 ms if the remote device takes 500 ms to handle each arrived message.

Figure 5.6 depicts calibration activity.

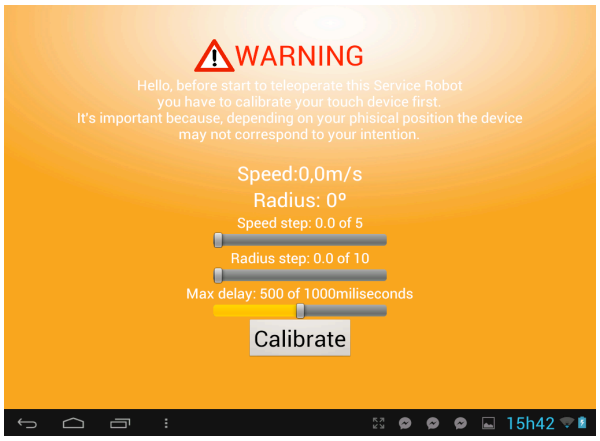

Figure 5.6. Calibration activity

When the user presses calibrate button, his preferences are saved in Android's shared preferences and loaded when the user starts the teleoperation session.

5.1.5 Teleoperation Activity

This activity is the heart of the application; after all processes of registration and authentication are complete the user can finally start a teleoperation session.

In the **onCreate** the first steps to load the preferences that were saved before, such as chosen device, its IP address, port, etc. Posteriorly, the application tries to establish a connection with the device through a ZMQ connect method. If failed an error message is popped up, if the connection succeeds and if the ServRobot's camera is turned on, an image is shown in the screen, very useful when the operator does not have a clear view of its teleoperator.

The video feedback is given through an URL that is passed to the teleoperation device to play, to avoid network congestion, it is not feasible to provide the video feed using the ZeroMQ protocol, also it was decided to supply this video with a low throughput in terms of frames per second (FPS), however, it is enough

for a good perception of the environment. This URL is obtained, using the message protocol, through the command **GetSensorInfo**.

To start to send control commands to the ServRobot, the user slides his finger from bottom to upper, and an XML message is sent, containing the command **StartTeleOp.** Immediately the accelerometer starts to read its values, which is why users must calibrate their device before start teleoperation otherwise some nonsense values are sent to the teleoperator.

```
if (speed < 0) {
    speed = 0;
}
if (speed > 10) {
    speed = 10;
}

if (radius < -30) {
    radius = -30;
}
if (radius > 30) {
    radius = 30;
}
speed = speed * (float) 0.05;
```

Figure 5.7. Code used to limit the values read from the accelerometer.

To avoid dangerous speeds, the application is limited to send the maximum value of 0.5 meters per second; this value was already being used in radio teleoperation. Also the turn rate is limited to 30º per second or ±0.52 radians per second (the messages contain the turn rate in radians), these are the maximum values accepted by ServRobot's driving motors.

As said before, the accelerometer is really hard to control, at the minimum movement or even standing still it reads a different value from its axis.

This was a problem, as an event was triggered on every small change on the axis and the control messages were sending using this event. This issue not only flooded the network with useless messages but also bombed the teleoperator with tons of messages.

The solution was pretty easy, to avoid picking in all the values a median filter was implemented, a counter and a maximum delay (as explained in 5.1.4) were introduced.

On every 20 counts or after a delay, a message is sent; the reason of the number twenty was based on some tests that were run in the device (Tablet BQ Curie 2 Dual Core 1Ghz). Every 500 milliseconds the device could only count 17 times.

Also the current value is always compared against the last one, if it is equal, then the message is not sent. Figure 5.8 shows the code that does what have been presented above.

```
//Collects samples and puts in the array
speed_buffer[counter] = speed;
radius_buffer[counter] = radius;
counter++;

//Timer difference between current tyme and the time of last sent msg
long timeDif = System.nanoTime() - lastUpdate;
if ((counter >= 20) || (timeDif >= MAX_DELAY)) {
    double speed_aux = Utils.median(speed_buffer);
    double radius_aux = Utils.median(radius_buffer);

    //Updates TextViews with the current values
    radius_TextView.setText("Radius: " + (int) radius_aux + "°");
    speed_TextView.setText("Speed:" + df.format(speed_aux) + "m/s");
    counter = 0;

    //Checks if the current msg is equal to the last, if yes does not send
    if (!lastResult.equals(df.format(speed_aux) + "#" + (int) radius_aux)) {
        acknowledge = teleOp.SetAngleAndSpeedQuery((float) speed_aux, (float) Math.toRadians(radius_aux));
    }
    lastUpdate = System.nanoTime();
    lastResult = df.format(speed_aux) + "#" + (int) radius_aux;
}
```

Figure 5.8. Code that showing the filtering process

When the operator wants to terminate the communication with the teleoperator, he can slide the finger from upper to bottom and the control message to stop is sent, however if the user wishes to restart the teleoperation, he may at any time do it.

The ZMQ socket used for this process it is only killed when the teleoperation activity is closed.

If the connection is lost, in the robot side there is a mechanism that prevents damage to the robot, by stopping the robot, thus avoiding crashes or dangerous situations. In these situations an alert is popped up in the mobile device (after three sending attempts) that informs that the connection has been lost.

5.2 Tests and Results

Since the beginning of this dissertation, some things had to be taken into account. What if tests could not be run in the real devices? This thesis was being developed in parallel with two other that would provide, the capability of controlling the robot in from the DVA infrastructure.

Given the possibility that the work on those areas might not be concluded in time to be able to do the tests under real circumstances, some simulators have been developed in order to test the current development, get performance results and analyze them afterwards

5.2.1 DVA Simulator

This simulator was needed in particular for the development and test of registry, permissions and device list requests functions.

Every device, client tablets or smartphones, service robots (if more than one) have first to be registered in the DVA system before it is be able to interact with each other.

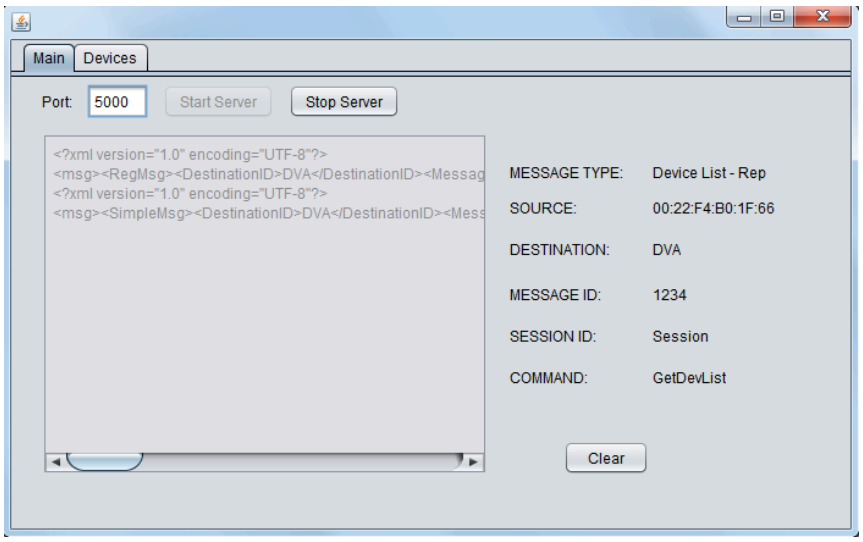

Figure 5.9. Dva Simulator main tab view.

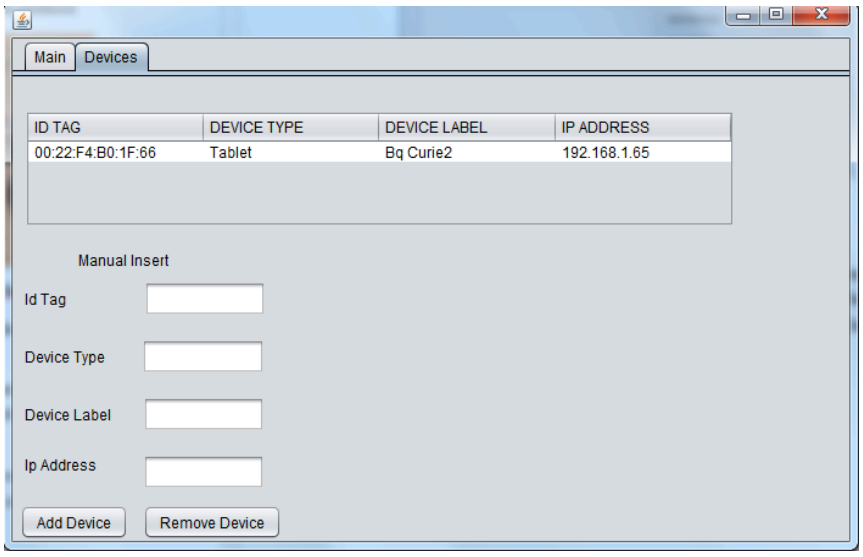

Figure 5.10. Dva Simulator devices tab view.

Figure 5.9 and Figure 5.10, depicts the two tabs of the DVA simulator. The main tab is used to start the server, listening a given port, and also to stop it.

The log window shows all incoming requests and parses all the elements of the message header after that shows them on the right side.

The second tab, Devices, has a table with all current devices connected with the system. For simulations tests, it is also possible to add fake devices manually, but of course it is expected that if a client tries to interact with one of those devices (fake ones) he will get no response back and a timeout will be triggered.

5.2.2 ServRobot Simulator: Raspberry Pi Approach

A small rover was adapted and developed in order to test the application and the communication protocol. It's true that this small robot is much more limited than ServRobot, it doesn't have the huge set of sensors, data acquisition boards or the fast CPU that ServRobot has, but for teleoperation and simulation results it's close enough.

PiBot, (Raspberry Pi + Robot) depicted in Figure 5.11, it is composed by parts of a disassembled remote Nikko Car, a Raspberry Pi, L298N Dual Bridge, a Webcam and an external power bank.

Figure 5.11. PiBot, an adapted remote controlled car to simulate ServRobot.

Nikko Car's parts, contains a set of wheels, a stepper motor and a servomotor.

It is controlled by a Raspberry Pi (RPi) revision 2. This RPi was overclocked at 1Ghz because it was needed a fast CPU to handle and threat burst request and reply messages.

RPi's I/O pins are connected to a L298N Dual Bridge, which allows controlling PiBot's motors using pulse width modulation (PWM) signals and digital input signals.

The stepper motor was controlled by a PWM signal, in the other hand servo received a digital signal. Note that, using the developed protocol to teleoperation messages, a speed (m/s) and a turn rate angle (radians) are expected.

Speed can be easily simulated by setting a percentage (0-100) to a PWM signal, unfortunately, servos cannot read angles and this one was only set up to turn left and right.

Finally, the camera was mounted to give a visual feedback. Its video is transmitted as MJPEG stream and showed up in the mobile application.

As RPi runs a 32 bit Java Virtual Machine, a software to perform the above actions was created. ZMQ, Xerces and JAXB biddings were used to handle received XML messages.

After running a few tests, using a Raspberry Pi, appeared to not be a good option to work as a controller. The processes the of validation of the incoming XML messages, unmarsheling the elements of the message, actuate over the motors and send back the reply showed to be too much, even at 1 GHz of CPU, RPi is way too much slower than a data acquisition boards installed in ServRobot.

Sometimes the delay between request-response went up to 3 seconds. This is unacceptable regarding a teloperation scenario.

5.2.3 ServRobot Simulator: Software Approach

This simulator was developed due the limitations of PiBot. Using this simulator is possible to test all the components of the architecture: message protocol, application (client) and ServRobot(simulated).

ServRobot's simulator has four windows as shown below in Figure 5.12, the first window is a Log which prints all incoming teleoperation messages. The second window it is a side view of ServRobot, the wheels spin at a certain speed, depending on the sent requests. Below this window there is the top view, this view illustrates the turn rate angle that is also being requested.

Finally the last window, fourth, simulated a manual control, it permits to register the ServRobot at DVA's system (or simulator), also to start the ServRobot's listening port, for the requests, and at last to tests the turn rate and the speed manually, without any request from a client.

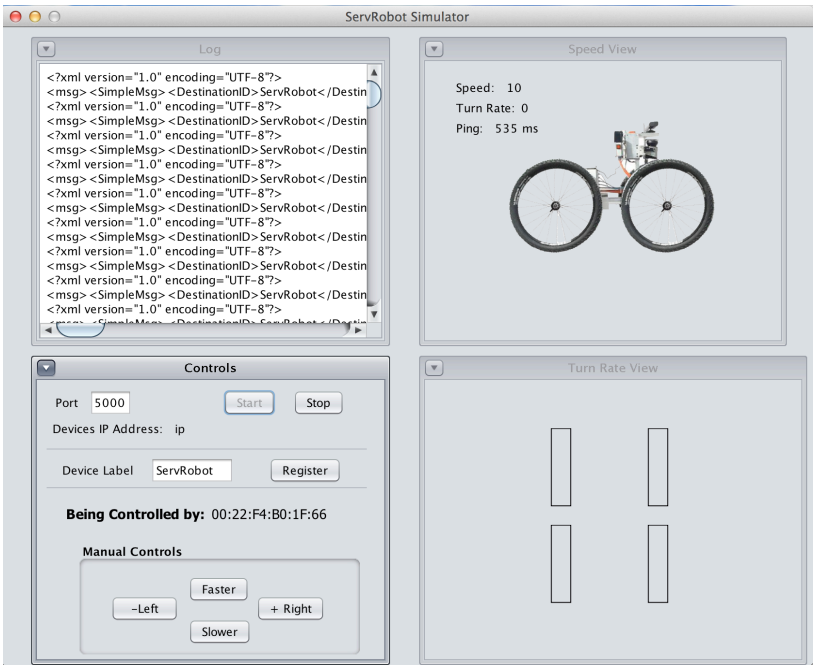

Figure 5.12. ServRobot's software simulator.

With this simulator the request-reply pattern used on the developed protocol fulfill all the specifications, especially regarding the delays. The delays are much smaller, less than 1s counting from the request until received the reply back.

This simulator is running under OS X at 2.4 GHz processor with 8 GB of RAM. It has three running threads one for the spin and turns the wheels, other to receive the incoming requests and the last one to measure the latency between the two sources (client – ServRobot).

5.2.4 ServRobot

So far tests and results were being taken from simulators, when this protocol was tested in the real device, the expected results were more or less of a mix of all the simulations.

In fact, the developed protocol and application worked properly, the whole architecture was expected to communicate harmoniously without any fails, and it actually did!

In the second chapter it was mentioned that most of the communication protocols used in teleoperation, are UDP based and those does not need an acknowledge to every command that is sent.

Against all these, as said in the fourth chapter, teleoperation commands are sent via ZMQ sockets using Request-Reply, which means that to every request, a reply is expected. The choice was made based on ZMQ lightweight and fast transmission characteristics, yet it proved to be not best choice.

Even with ServRobot's processors power, a request-reply pattern introduces a considerable delay in the teleoperation system, so that we thought it would be better to change the pattern of request-reply to publish-subscriber, but for speed and turn rate commands only. The commands would be published in a topic and the ServRobot would read the values from that topic.

Nevertheless, there are more things we have to take into the account; the mechanical limitations of a 60 kg prototyped robot are something that we cannot put apart.

It has a really good CPU as said, which allows to process requests with a reasonable speed and worked fine. However, it still has some delay in the processing, this is the result of the hardware limitations, and the operator has to be aware of these.

As a result of the tests performed, we can conclude that the commands should be differential instead of absolute; in other words, when the mobile device is turned right, the angle should increment slowly by one incremental value until it reaches the limit ($30^{o}/s$), instead of trying to set the robot's wheels angle as a mimic of the absolute angle of the device in relation to the floor. In fact, the more pronounced the angle, the higher should the increment be.

----This page was left empty on purpose----

6

6 Conclusions

In this chapter the conclusions for the work contained in this dissertation are presented, and also a prospective about the future work is discussed here.

This work aims to provide a contribution to the research that is being made in DVA and ServRobot focused towards integration of a mobile service robot within a distributed surveillance system, with the aim of adding further surveillance, information acquiring and eventually remote operation within the surveillance system DVA.

In the chapter 4 a communication protocol was presented in order to integrate both systems, allowing them to exchange information.

Chapter 5 presented the development and results. In the development, the communication protocol was put in practice, as expected in the tests and results; the presented architecture and communication protocol are fully operational.

However, by operation it does not mean flawless. Some issues, specially regarding time delay were between commands and feedback were higher than expected, and some considerations were provided towards enhancing the full system operation.

6.1 Knowledge Contributions

The work made for this dissertation, presents mechanisms for system integration also shows a very detailed research in the telerobotics fields.

The proposed architecture and communication protocol follows the research being made about two projects, DVA and ServRobot, and it is functional.

Mobile applications are becoming more and more popular, allowing to do things we had no idea that was possible.

The developed application was build, allowing DVA and ServRobot to communicate taking profit of the advantages of the both projects, like registering in the DVA Surveillance System and teleoperating the ServRobot to attend on triggered events.

During the development stage, three prototypes were produced to help in the tests and results evaluation stage.

The DVA simulator was particularly relevant, especially when there was no possibility to connect to the real server.

To test the behavior two prototypes of ServRobot were made (PiRover and a software approach).

Also two papers[27][28] were published in the scope of this dissertation.

6.2 Future Work

As mentioned earlier, during the validation the aim was to assess the functionality of the mobile application and the communication protocol.

Its performance was not very high, and for that reason, higher delays were presented in the communication. To eliminate, or at least, decrease those delays another communication pattern for speed and direction commands has to be adopted, which means we need to migrate from ZMQ Request-Reply to ZMQ Publish-Subscribe. And at the same time, reduce the high traffic created by the fast outputs and high instability of the accelerometers. Some results have already been achieved by the introduction of mean and avoid the processing of redundant messages algorithms.

ZMQ Publish-Subscribe is somehow an approach to a connectionless protocol, such as User Datagram Protocol (UDP), and for this specific case it is only needed to send a command and ServRobot must perform the action, no feedback is expected.

Also, the mobile application can be even richer in details. So far, it only allows to register and remote control the ServRobot, but new functionalities can be added.

Accessing ServRobots sensors, like temperature, smoke or any other can be an asset, the communication protocol is prepared to it, but the application has no implementation for this.

It could also be interesting , if the application had a way to subscribe to a sensor with a time frequency, allowing to the operator to monitorize it, which could be

very relevant on taking conclusions while the robot is being teleoperated, for example on recognition of events such as fire.

6.3 Concluding Remarks

The concept of automation has evolved far from what many people may even think. Even in the ancient civilizations, engineers of that time attempt to build self-operating machines.

The industrial revolution was a trigger for the fast growing of robotics and its all derived areas. In the early past (in comparison with ancient times), mechanical and hydraulic mechanisms were used, nowadays, with the advanced of the electronics, robots are becoming more and more efficient and self-controlled, showing somehow certain autonomy.

Machines are now part of our lives, they are spread in so many fields; medicine, space exploration, military and marine applications are just few examples.

The machines make our lives simpler and risk free (or at least risk reduced), this led to a large investment in robotics along the years.

In this dissertation two biggest areas were presented, focused on two developed project results: Surveillance Systems (DVA) and Service Robots (ServRobot). The dissertation was focused in merging and integrating the two existing systems, which did not interact with each other in the past.

With the integration of DVA and ServRobot it was possible to take advantage of the best of what the individual systems could give us.

A communication protocol and a brand new architecture were developed in order to make this integration possible.

The way it was developed, it allow anyone, which is using a mobile device to check the events (gas leaks, fire, floods) triggered by DVA and teleoperate ServRobot to those sites using the very same mobile device.

Using the accelerometer it is possible to control the speed and the direction of the ServRobot, the feedback is given by camera and displayed in the screen of the device, with this the ServRobot can be remotely controlled, act like an active agent of DVA's Surveillance system, keeping humans far from danger in hazardous situations.

----This page was left empty on purpose----

7

7 References

[1] "Moore's Law." [Online]. Available: http://www.mooreslaw.org/.

[2] Grass, "Acoustic Myography (AMG)," 2004.

[3] J. Cannan and H. Hu, "Human-Machine Interaction (HMI): A Survey," 2011.

[4] F. do A. Gurgel, "Glossário de Engenharia Industrial," 2009.

[5] L. Basañez and R. Suárez, "Teleoperation," pp. 449–468, 2009.

[6] W. T. Townsend, D. Wilkinson, and B. Zenowich, "Teleoperator System With Master Controller Device and Multiple Remote Slave Devices," US2012/0041599A1, 2012.

[7] "Online Dictionary," 2014. [Online]. Available: http://dictionary.reference.com/browse/autonomous.

[8] S. Lichiardopol, "A Survey on Teleoperation," 2007.

[9] N. Chopra, M. W. Spong, and R. Lozano, "Synchronization of bilateral teleoperators with time delay," *Automatica*, vol. 44, no. 8, pp. 2142–2148, Aug. 2008.

[10] R. N. Silva, "Supporting Text in the course of Control Theory," 2012.

[11] NASA, "Mars Exploration Rovers." [Online]. Available: http://www.nasa.gov/redplanet/mer.html.

[12] ESA, “ExoMars Programme.” [Online]. Available: http://exploration.esa.int/mars/46048- programme-overview/.

[13] P. Appelqvist, J. Knuuttila, and J. Ahtiainen, “Development of an Unmanned Ground Vehicle for Task-Oriented Operation - Considerations on Teleoperation and Delay,” 2007.

[14] D. Lee, A. Franchi, P. R. Giordano, H. Son, and H. . Bulthoff, “Haptic Teleoperation of Multiple Unmanned Aerial Vehicles over the Internet,” *IEEE Int. Conf. Robot. Autom.*, pp. 1341–1347, 2011.

[15] “Portuguese Hydraulic Institute.” [Online]. Available: http://www.hidrografico.pt/rov-remotely- operated-vehicle.php.

[16] “Ziphius the first app-controlled drone.” [Online]. Available: http://myziphius.com/.

[17] S. Kumar, “Introduction to Telesurgery,” 2008.

[18] A. S. Tanenbaum and D. J. Wetherral, *Computer Networks*, 5th ed. 2011, pp. 541–552.

[19] Y. Zhou, M. Meng, H. Liang, L. Sun, Z. Xu, and K. Shen, “TFRC-PROBE: A Transport Protocol for Teleoperation Systems of Mobile Robots,” *IEEE Int. Conf. Inf. Acquis.*, pp. 1492–1496, 2006.

[20] P. X. Liu, M. Q.-H. Meng, P. R. Liu, and S. X. Yang, “An end-to-end transmission architecture for the remote control of robots over IP networks,” *Mechatronics, IEEE/ASME Trans.*, vol. 10th, no. 5, pp. 560 – 570, 2005.

[21] Y. Uchimura and T. Yakoh, “Bilateral robot system on the real-time network structure,” *Ind. Electron. IEEE Trans.*, vol. 51th, no. 5, pp. 940 – 946, 2004.

[22] L. Ping, L. Wenjuan, and S. Zengqi, “Transport layer protocol reconfiguration for network-based robot control system,” *Networking, Sens. Control. 2005. Proceedings. 2005 IEEE*, pp. 1049 – 1053, 2005.

[23] R. Wirz, R. Marin, M. Ferre, J. Barrio, J. M. Claver, and J. Ortego, “Bidirectional Transport Protocol for Teleoperated Robots,” *Ind. Electron. IEEE Trans.*, vol. 56, no. 9, pp. 3772 – 3781, 2009.

[24] “eLinux Website.” [Online]. Available: http://elinux.org/.

[25] “Mobile OS Market Share.” [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[26] “Android.” [Online]. Available: , http://developer.android.com/guide/components/fundamentals.html.

[27] B. Dias, B. Rodrigues, J. Claro, J. P. Pimentão, P. Sousa, and S. Onofre, "Architecture and Message Protocol Proposal for Robot's Integration in Multi-Agent Surveillance System," *Rough Sets Curr. Trends Soft Comput.*, vol. 8536, pp. 366–373, 2014.

[28] J. Claro, B. Dias, B. Rodrigues, J. P. Pimentão, P. Sousa, and S. Onofre, "Autonomous Robot Integration in a Surveillance System," *Power Electron. Motion Control Conf. Expo. (PEMC), 2014 16th Int.*, pp. 713–719, 2014.

[29] "ZeroMQ Official Website." [Online]. Available: http://rfc.zeromq.org/spec:23.

[30] "ZeroMQ Request-Reply." [Online]. Available: http://rfc.zeromq.org/spec:28/REQREP.

[31] "ZeroMQ Publisher-Subscriber." [Online]. Available: http://rfc.zeromq.org/spec:29/PUBSUB.

[32] "Xerces." [Online]. Available: http://xerces.apache.org/.

[33] "Xerces for Android." [Online]. Available: http://gc.codehum.com/p/xerces-for-android/.

[34] "Dom for Java." [Online]. Available: http://dom4j.sourceforge.net.

----This page was left empty on purpose----

8

8 Glossary

Expression	Abbreviation	Description
Android OS	-	Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google.
Activity	-	An activity is a single screen with a user interface.
Application Programming Interface	API	An API specifies a software component in terms of its operations, their inputs and outputs and underlying types.
Delay	-	Delay is the technical term for the delayed signals in electronic circuits, usually the sound delay in satellite transmissions.
DVA	-	Advanced Surveillance System based on Agents is a project developed by Holos, co-financed by QREN
eXtensible Markup Language	XML	XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
Framework	-	Is an abstraction that unites common code among multiple software projects by providing a generic functionality.

Global Positioning System	GPS	GPS is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.
Human-Machine Interface	HMI	HMI is the part of the machine that handles the human–machine interaction.
Humanoid	N/A	Humanoid is a machine (robot) that has an appearance resembling a human being.
Industrial Robots	-	An automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either, fixed in place or mobile for use in industrial automation applications.
International Federation of Robotics	IFR	IFR is a professional non-profit organization established in 1987 to promote, strengthen and protect the robotics industry worldwide.
Java	-	Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.
Middleware	-	Middleware is computer software that provides services to software applications beyond those available from the operating system.
Mobile Application	APP	A mobile app is a computer program designed to run on smartphones, tablet computers and other mobile devices.
Moore's Law	-	Moore's Law is the observation that, over the history of computing hardware, the number of transistors in a dense integrated circuit doubles approximately every two years.
Operator	-	An individual that controls a remote device or machine.
Remote Operated Vehicle	ROV	ROV is a tethered underwater vehicle.
Robot	-	A robot is an automatic mechanical device.

Service Robots	-	Service robots assist human beings, typically by performing a job that is dirty, dull, distant, dangerous or repetitive, including household chores.
ServRobot	-	ServRobot is an autonomous service robot developed by Holos.
Teleoperation	-	Teleoperation indicates operation of a machine at a distance.
Teleoperator	-	Teleoperator is the remote controlled machine, also known as slave.
Telepresence	-	Telepresence refers to a set of technologies which allow a person to feel as if they were present, to give the appearance of being present.
Telesurgery	-	Telesurgery is a surgery performed by a surgeon at a distance.
Transmission Control Protocol	TCP	TCP is one of the core protocols of the Internet protocol suite(IP). It provides reliable, ordered and error-checked delivery of a stream of octets between programs running on computers connected to the network.
Unmanned Aerial Vehicle	UAV	An UAV, commonly known as a drone, is an aircraft without a human pilot aboard.
Unmanned Ground Vehicle	UGV	An UGV, is a vehicle that operates while in contact with the ground and without an onboard human presence.
User Datagram Protocol	UDP	UDP is one of the core members of the Internet protocol(IP). It uses a simple connectionless transmission.
Zero Message Queue	0MQ	0MQ is a high-performance asynchronous messaging library aimed at use in scalable distributed or concurrent applications.