Filipe Dias Coelho da Cruz Nunes **Remote control of a robot via a handheld device using inertial sensor information**

Universidade do Minho
Escola de Engenharia

Filipe Dias Coelho da Cruz Nunes

Remote control of a robot via a handheld device using inertial sensor information

Outubro de 2011

Universidade do Minho
Escola de Engenharia

Filipe Dias Coelho da Cruz Nunes

Remote control of a robot via a handheld device using inertial sensor information

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de Mestre
em Engenharia Electrónica Industrial e Computadores

Trabalho realizado sob a orientação do
Professor Doutor Agostinho Gil Teixeira Lopes

Outubro de 2011

Nome: Filipe Dias Coelho da Cruz Nunes

Endereço electrónico: filnunes@gmail.com

Título dissertação:

Remote control of a robot via a handheld device using inertial sensor information

Orientador:

Professor Doutor Agostinho Gil Teixeira Lopes

Ano de conclusão: 2011

Designação do Mestrado:

Mestrado Integrado em Engenharia Electrónica Industrial e Computadores

Resumo

A interação entre o Homem e os veículos tende a evoluir de uma atuação direta para uma condução totalmente autónoma. Neste processo evolutivo, os investigadores acabam, gradualmente, com esta dependência através do uso de novos tipos de interação humano – computador e mediante novos dispositivos.

O objetivo principal deste estudo passou por criar uma aplicação para o iPod Touch que controlasse sem fios o robô do tipo carro pertencente ao Laboratório de Automação e Robótica da Universidade do Minho. Por outro lado, também foi decido criar uma interface com uma simples navegação entre menus e controladores intuitivos para o utilizador. Por fim, o interface teria que ser capaz de reproduzir o *streaming* de vídeo capturado de uma câmara montada na frente do robô.

O sistema foi composto por um iPod Touch a funcionar como controlador e um robô do tipo carro com um netbook incorporado a servir de mediador entre os dois. A conexão entre os dois dispositivos foi efectuada através de Wi-Fi no modelo cliente/servidor (para o iPod e netbook, respectivamente). Foram testados dois modos de controlo: a informação do acelerómetro presente no iPod ou barras deslocáveis desenhadas no ecrã sensível ao toque. Adicionalmente, a câmara Kinect foi instalada na frente do carro com o propósito de filmar a perspectiva do mesmo, permitindo a sua visualização no fundo do ecrã da aplicação do iPod. O sistema foi testado com quatro *setups* diferentes: resolução de imagens de 133x100 pixels (baixa resolução) e 240x180 pixels (alta resolução), numa rede pública e noutra privada, com 5 testes em cada, perfazendo um total de 20 testes.

Os resultados demonstraram que o sistema apresentou, na sua melhor performance, uma média de 8 frames por segundo, latência de 0,044 segundos e fácil condução do robô.

O objectivo desta dissertação foi atingido, demonstrando aptidão suficiente com um nível considerável de frames por segundo e a interação entre o utilizador e o carro quase em tempo real.

Palavras-chave: robô, iPod, *wireless*, controlo.

Abstract

The interaction between Man and vehicle tends to move from direct input to completely autonomous driving. In this transition researchers cut the dependency strings that tie the two together with new human computer interaction systems and devices.

The main purpose of this study was to create a wireless iPod Touch application to control a car-like robot from the Laboratory of Automation and Robotics of University of Minho. Another aim was to create an easy and enjoyable human interface with simple menu navigation and intuitive controllers for the user. Finally the interface would have to be capable of showing video streaming from a camera mounted on the forefront of the robot.

The system was composed of an iPod Touch acting as a wireless controller client and a car-like robot accommodating an intermediary netbook server between its microcontroller and the client application. The connection was effectuated through Wi-Fi and there were two controlling modes: with the iPod's accelerometer data or with touchable sliders drawn in the screen. Additionally a Kinect camera was installed in the forefront of the car with the aim to stream video in the car's perspective, watchable on the iPod's application background. The system was tested with four different setups: with image resolutions of 133x100 pixels (low resolution) and 240x180 pixels (high resolution), in a public and a private network, with 5 trials each, making up a total of 20 runs.

The experiments demonstrated that the system attained an average of 8 frames per second, latency of 0,044 seconds and easy robot maneuver in its best performance.

The goal of this dissertation has been accomplished, securely demonstrating aptitude in the private network with highly considerable frames per second as well as virtually providing real-time interaction between the user and the car.

Keywords: robot, iPod, wireless, control.

Acknowledgements

In the first place I would like to dedicate this thesis to my parents, for all the sacrifices they had to make that allowed me to be where I am today. I owe them much of what I have achieved and to them I dedicate my greatest acknowledgement.

I wish to thank my professor, Gil Lopes, not only for the tireless reviews, but also for all the support, guidance and enthusiasm throughout the development of this work.

To Tânia Simas, that kept me sane through all those nights of coding. For all the encouragement and advices about extra spaces, patience and always being there for me.

To my brother for the helpful reviews and tips.

Finally I would like to thank my friends for all the great times spent together during these last years and my colleagues from the laboratory for all the patience and suggestions.

Contents

List of Figures

List of Tables

Abreviations

MMP – **M**ulti**m**edia **P**layer

OS – **O**perating **S**ystem

HCI – **H**uman-**C**omputer **I**nteraction

FPS – **F**rames **P**er **S**econd

IP – **I**nternet **P**rotocol

TCP – **T**ransmission **C**ontrol **P**rotocol

UDP – **U**ser **D**atagram **P**rotocol

Chapter 1

Introduction

The world has been witness of countless changes and innovations on the field of technology since the invention of the Vacuum Tubes by Sir Frederick Williams and Tom Kilburn in the early 1940s.

Nowadays there is a different type of revolution in action. It is not particularly about the core technology itself, but the way engineers make of it. For decades, the personal computer was the great machine everybody wanted to have at home for entertainment, business or education. Then the laptop came along and quickly turned all attention to it, especially with its specifications getting better opposing to lower prices due to greater competition in a small time frame. As its size continued to decrease, the netbooks appeared, mostly to satisfy basic necessities, generally connected to Internet subjects.

In a recent past, the spotlight quickly turned from laptops and netbooks to handheld touchscreen devices, such as tablets, smartphones and multimedia players (MMP). Although much thinner and smaller than the former, these devices provide both regular users and developers with the latest technological innovations.

Until a new revolutionary technology is created, tablets and smartphones will continue to be the best solution in personal technology. This is, among others, the main reason why the development of applications for these devices greatly surpassed the software development for computers and became such hype among developers.

The goal of this thesis is to present another approach on building an application for a portable device, specifically designed to control a car like robot.

1.1 Background

The workgroup in which this project was developed is called Laboratory of Automation and Robotics (LAR) and belongs do the Industrial Electronics Department. A wide number of innovative projects take place in the laboratory, some during years, and some without an explicit end date being improved year after year by different students.

These projects cover a wide range of applications in the automation and robotics field which can go from an autonomous golf balls picking robot [1] to the MARy omnidirectional robot integrating an articulated arm with six degrees of freedom, vision processing, voice recognition and speech capabilities system [2], [3], [4], along with the university's robotic football team [5], [6], [7] and the omnidirectional wheelchair for disabled people [8].

These projects have three objectives in common: the possibility for students to develop a research career; the possible future usability or commercialization of the developed products; the demonstration of the laboratory's know how with pedagogical orientation.

With the aim of developing an application for a portable device, a robotic car named FormulaUM developed by the laboratory was chosen for our tests. This car would be driven and controlled by the portable device. This robotic car was built for autonomous driving challenges in order to participate in the annual national robotic event called "*Festival Nacional de Robótica*" (Robotics National Festival). The FormulaUM was originally built in 2010 and it has been upgraded ever since as new students take charge of the project.

1.2 Objectives

The work developed in this thesis will focus on several steps required to achieve the final goal of having a fully functional portable device application that connects wirelessly to the FormulaUM car robot and allows the user to control it, while receiving at the same time a video streaming of a camera mounted on the forefront of the car.

Initially the user will be presented with some choices to make, such as the type of input – either the accelerometer motion or a duo of sliders drawn in the screen – or network properties.

As a processing gateway between the application and the car there will be a Portuguese netbook (Magalhães), whose job is to control the connection, receive the motion data from the application and forward it to the microcontroller responsible for making the motors spin. It also provides for the video acquisition and streaming back to the application on the portable device.

The portable device and the netbook must establish a wireless connection following a server/client model. After the connection is successfully completed, the user will then be able to drive the car as long as the wireless connection exists. The user may also disconnect whenever he feels like it bringing the car to a secure full stop.

There will be some security features as well in case of a sudden connection loss.

1.3 Structure of the thesis

In the following chapter the background information required to understand the project will be provided in steps in order to fully demonstrate all the processes occurring in the system. Additionally both academic and commercialized solutions relevant to this study will also be presented.

The system implementation is described in the third chapter, focusing on the most important steps with detailed description regarding the programming of the devices, giving prominence to the portable device's application. The problems encountered in this phase, as well as solutions for those problems are also presented in this chapter.

The forth chapter explains the methodology employed in the tests with details on the device specifications and the test environment, as well as an explanation of the variables to evaluate.

The results of the experiments will be addressed separately in the fifth chapter in order to evaluate the performance of this application, whether referring to image quality, connection latency and other relevant variables resulting from the interaction between the user and the car. Additionally some relevant results from other studies will be compared to the ones obtained in this thesis.

The conclusions of the developed work and some considerations for future work will make the sixth and final chapter.

Chapter 2

Literature review

In order to achieve and understand the system introduced in the previous chapter, it is essential to isolate each point featuring its unique challenges and work to be developed. All these fragments of the whole scheme will be analyzed in the next sections of this chapter. The problems to solve related to those fragments and some work already published about the theme will be presented in the end of this chapter.

2.1 Remote control of a mobile device

The remote control of mobile devices, regardless of the purpose one is built for, does not follow a defined rule or standard. The logic behind each one of these systems depends on their own objectives, the platform for which the application is programmed to, the kind of mobile device to control and what type of control is required. Although this lack of strict guidance might be troublesome for a new project, there are some common steps in most of them, which will be discussed hereafter.

2.1.1 Control device

The control of a robot can be done from a manifold of devices. The choice of the perfect control device for a project depends on how the application will control and connect to the robot, what to show and inform the user and the mobility required for both ends.

Human – Computer Interaction (HCI) is an area of research aiming at the physical, psychological and theoretical aspects in the interaction between people and computer [9]. Although computer science and systems design are considered the core disciplines, HCI is a multi-disciplinary subject involving the "design, implementation and evaluation of interactive systems in the context of the user's tasks and work" [9]. Thus apart from the already stated skills, psychology, sociology, cognitive science and ergonomics are also required having the user as the main subject of interest. On the product side graphic design skills are almost mandatory for a visually attracting interface and business skills enables the understanding of the market's needs and desires [9].

HCI shows then that a human-computer interface has to be coherent all the way from the user point of view and utilization to technical specifications required by the project, passing by the task at hand. For a certain control system, this can be done in two ways: a mandatory controller will implicate one specific interface or vice versa, an interface is required and the controller is chosen accordingly.

Fong et al [10] presents a suite of remote driving tools to control the same mobile robot with different interfaces to the user. The authors first implemented a remote driving interface based on visual gestures: the robot's movement is controlled by the user with determined arm and body gestures recorded by a stereo vision system and processed by software. The control device in this case was clearly a computer and the interface was practically not needed. The authors claimed that this type of control could be easily used anywhere as long as the user was in line of sight with the cameras.

The second approach to reach the goal was with an application installed in a personal digital assistant (PDA), back in 2001 [10]. This time the equipment was a portable device, which does not limit the user as the previous type of control does. An application had been made dedicated to the experience and now the user could directly control the car pressing directional buttons on the PDA's screen.

The last approach refers to a web-based system that works on a web browser in which, similar to the PDA's control, an interface shows the user live video feed and enables direct control of the car through directional buttons [10]. Since this is browser based, it can be accessed through any of the standard personal devices such as laptops, tablets and smartphones.

In parallel to these common control devices and interfaces, Microsoft has developed a system in which the user wirelessly controls a quadcopter with Microsoft Surface [11]. It benefits from the wide area of control, enabling more accurate movements and the display of more information not compromising its usability.

Fong et al [10] inquired both novice and expert users that had experimented controlling a robot through a PDA interface and has concluded that the interface had high usability, robustness and performance. The study also informed that the "users reported that the interface enabled them to maintain situational awareness, to quickly generate commands and to understand at a glance what the robot was doing." [10].

As mobility is seen as one of the most prioritized aspects of remote control, new smaller mobile devices are already seen as the best choice to use in these research cases. There are several tablets, smartphones and MMP devices available in the market nowadays ranging from affordable to very expensive prices. Besides the brand and type of equipment (with its own specifications), one other factor of great importance regarding the portable device is the operating system (OS) built in.

The OS available in those three groups of devices are basically Apple's iOS, Google's Android, Microsoft's Windows Phone 7, RIM's BlackBerry OS and Hewlett-Packard's webOS. Since this project involves the development of an application for a modern portable device, according to [12], the "'Very Interested' in Developing for Each Platform" graphic shows that iOS and Android, both phone and tablet versions, clearly dominate the developers' preference.

It was also presented in last April statistics about *Mobile Installed Base* in the United States of America, accounting for mobile phones, tablets and other connected media devices, where iOS was the number one mobile operating system, representing 44% of the whole market, followed by 28% from Android and 19% from RIM [13].

Another important factor on the choice of the OS was the availability of an iPod Touch 4th Generation (iOS) by the author. It should be noted that this iPod, although looking very similar to the iPhone, has more differences than the obvious lack of mobile phone network features, particularly lower RAM and some missing technological features. But more importantly, these different specifications do not reflect any real slowing in the processing of applications in general or this project application in particular. Giving the similarities, for the rest of this thesis all the projects, articles and other references concerning the iPhone, that do not depend on the different specifications between the two devices will be treated as valid for the iPod.

There are a large number of developers for this platform either as freelancers or working for software houses, for commercial or educational purposes or simply as a hobby. Due to the large amount of applications found in Apple's AppStore (official online place for downloading iOS applications) and by the number of articles already available, one can assume that a great deal of work has already been done for the iOS system. As an example of what has already been done with this device it was found that it can work as a wireless accelerometer for quantifying movement for all kinds of applications [14], [15], [16]; it can also serve the purpose of remote home automation and security control center as seen in [17], [18].

Even though the first generation iPod Touch was only first commercialized four years ago (2007), it was immediately accepted as the best MMP in the world at the time similarly to the following generations. Presently it stands as a very mature platform for applications and games development [19], [20], [21].

2.1.2 Types of control

Regarding this thesis' theme, the robot's movement depends solely on the controls sent remotely by the user. These movement variables are a translation of certain interactions between the user and the remote control's interface. According to the literature there has been an evolution on the types of control used over time, from which the most modern ones will be stated in an overview below.

Fong et al [10] experimented three types of controls: the first was through a PDA application where the user pressed speed and angle buttons in the touchscreen to steer and thrust the car; the second was through visual gesture monitoring using a stereo camera, raising the left arm lead to activating the control and with the movement of the right arm it was possible to control the robot like a joystick with two axis – up and down for forward and backward speed respectively, right and left to change the direction; lastly a web application able to be controlled from any browser with a control system like the PDA's was demonstrated, only with a mouse clicking, instead of touching the screen. They concluded that, except for the gesture system, those control types required low user training and allowed medium vehicle autonomy.

In another field of robotics [18], the authors controlled several aspects of home automation (domotics) from an iPad application. From the touchscreen the user controlled several aspects from home lighting to air conditioner temperature, making use of sliders that ranged from off for the lights and lowest temperature of the air conditioner, to the maximum value allowed of both applications. The authors also implemented buttons for various other functions and switches to turn on or off the alarms.

In the late years there has been a visible change of mentality regarding the usability and ergonomics in the user interface, greatly due to the Wii console, namely its wireless remote. The Wii Remote makes use of an accelerometer and an optical sensor to point to the television screen and gesture recognition to use in games [22]. This remote has been adopted in numerous projects, especially the ones where the wireless control of an object is required.

Neto et al describe how the Wii remote was used in demonstrations to disseminate technology among youth [23] in order to lure new students to engineering. The users were able to control a robot's arm with a ludic objective by rotating and translating the remote accordingly.

The Wii Remote was also used with the aim of controlling a robot movement [24]. The user had first to go through a training mode where a series of gestures were set in the recognition (controlling) mode. The author concluded that this approach was intuitive for a regular user, especially after defining its specific gestures.

Influenced by several racing games that recently appeared for portable devices, and because it is so natural for the user to imitate a wheel in its hands, some interfaces for portable devices adopted its own motion to control the desired object, which also require inertial sensors such as accelerometer(s) and/or gyroscope(s). For instance, Chua et al [25] explored four different interaction methods of tele-operating a robot. The fourth interaction design presented relied only on the accelerometer movement to send the steering and throttle commands to the robot. After a series of tests with 20 participants, they concluded that this interaction method might create some confusion to the user. Their second design implements sliders to control both direction and throttle, which return to the point of origin – the center of the bar with a null value – once the user lifts the fingers from the sliders.

Xing-Han et al [26] managed to control miniaturized car-like robots with accelerometers coupled to a microcontroller strapped to their wrists, thus using only hand movement.

Kubota et al [27] built an iPhone application to remotely control an omnidirectional robot, but in this case they used the accelerometer's x-axis to control the tilting of a webcam mounted on top of the robot.

2.1.3 Controlled device

On this theme any robot built to obey human orders is considered to be a controlled device. Terrestrial (cars alike) and aerial robots are the most common ones found on the literature [27], [28], [29], [30]. Nevertheless it was also found many other types of robots, from immobile stations for picking and placing objects [31] to other mobile robots replicating boats [32] or animals [33], all capable of receiving orders from a user. Thus it is possible to infer that the existence of any type of controllable robot is solely dependent on the human creativity and technology evolution [34].

In the latest years, researchers approached the theme with projects that, if successful, might lead to great revolutions in society. Wang and Ganjineh [35] developed an application for iPhone that allows the remote control (or monitoring) of an autonomous car developed in their university. This way, in search for the safest autonomous experience, they are able to control the car in case of failure.

The research on the topic will serve the automotive engineers in perfecting the so-called advanced driver assistance systems (ADAS), which might ultimately lead to fully autonomous transport vehicles on the daily life in the future [35]. Google's autonomous vehicle

[36] already goes beyond the previous system offering a fully autonomous solution demonstrated in real situations. Having driven about 300000 kilometers in city streets and highways, Google's over the edge technology packed car proved that with the continuous research breakthroughs and technology evolution it will be possible to consider these cars common in human life in the future [36].

2.1.4 Connection between the controlling device and the mobile device

The link between the controlling device and the device to control is essentially the path for data transfer. This can happen through two types of connection: wired and wireless.

Until the discovery of wireless data transfer, the only option regarding controlling electronic objects was through wires. Initially driven by the cellular telephony [37], nowadays everything tend to become wireless. This is due to many reasons, especially the possible physical distance from end to end and comfort for the user. But even today, there are factors that hinder the choice of wireless technologies for specific goals, such as fading, interference, data security threats and lower data rates [37], [38]. None of those factors affect noticeably this project and therefore the choice for wireless communication over wired communication is performed.

Wirelessly remote controlling an object can be done through more than one way, i.e., there is more than one type of wireless connection, each with its advantages and disadvantages over the others.

2.1.4.1 Bluetooth

Since the Bluetooth technology has appeared, many researchers switched from infrared remote controlling to this form of out-of-sight wireless communication. Wongwirat et al [39] successfully remotely controlled a rescue robot based on the Lego Mindstorms NXT from a computer via Bluetooth. The interface showed the values of the connection strength, which allowed the user to control the proper distance to the robot in order to avoid a sudden loss of signal and consequent disconnection. The visual approach was almost obligatory since

Bluetooth is known to have a small wireless range (maximum of 10 meters in some versions) [40].

Bluetooth is also very common nowadays on home automation systems and as an example, Lee and Choi [41] successfully created a prototype for a heater and air-conditioner controlled system by Bluetooth.

2.1.4.2 Zigbee

Zigbee is another type of wireless communication mostly addressed for sensor networks due to its low cost, high number of allowed devices, small data exchange and allowing unsupervised monitoring of a wide range of applications, among other reasons [42], [43]. With defined IEEE 802.15.4 standards, this solution had high adoption rate among industrial and military appliances, agriculture, animal and health monitoring and home automation. Because it is aimed at Personal Area Networks (PAN), this solution is often discriminated for robot controlling projects. Despite this Ding et al [44] claim to have built an omni-directional mobile robot integrating what the authors call a "remote-cerebellum" which responds to commands sent by a "remote-brain" – this being the computer. This connection was achieved with Zigbee by a wireless communication module.

2.1.4.3 Wi-Fi

In 1997 IEEE approved the 802.11 standard for Wireless Local Area Network (WLAN) in the band of 2.4GHz. Since then the IEEE approved eight amendments with speed and other miscellaneous improvements. The connection between the robot and the controller can be done in two different wireless modes: infrastructure mode (devices communicate indirectly through a central place) or ad-hoc (the network does not rely on pre-existing infrastructure setup and administration, allowing devices to create or connect to networks "on the fly") [45]. In both modes, connected devices are assigned an Internet Protocol (IP) addresses that uniquely identify them on the network. TCP/IP is the protocol suite that commands the data transfer through networks, created as an architectural model describing functions and components.

The management of the connections occurs on the transport layer protocol (belonging to the TCP/IP protocol suite) and there are two protocols to choose from: the Transmission

Control Protocol (TCP) and the User Datagram Protocol (UDP). The first one provides a reliable and ordered stream of bytes from the sender to the receiver and the latter offers a faster yet unreliable data transmission. Because TCP uses an acknowledge signal each time a transmission is performed, it is slower than UDP, which has no way of knowing the message arrived to the receiver. TCP and IP are put together in the TCP/IP allowing for large data transmission, split in packets by the IP. To guarantee the accurate delivery of data, TCP will handle the acknowledge signals between the connected ends, allowing for retransmission of lost data or its reorder [45].

The choice of which transfer protocol to use results subsequently on the balance between latency and reliability of the data transfer. While TCP is mostly used for applications requiring lossless information communication, such as the World Wide Web, e-mail, and file transferring, UDP is used when there is no concern about loss of information and the main goal is the speed of the data transfer with real time demands, found on video streaming, voice over IP (VoIP) and online gaming [45], [46], [47].

Since the first demonstration of a controlled robot over Wi-Fi in 1995 [48] many other researchers working on this topic have published information about it. Wenbin et al [49] presented an Internet based tele-control system for a wheeled mobile robot. The robot connects through a wireless Router (Buffalo WHR3-AG54 Air Station BroadBand Router) and the users can control it in a browser in any computer or in a control panel LabView based as long as it is connected to the Internet as well.

Due to the reliability of the TCP/IP protocol, Janssen [50] developed a system to wirelessly exchange messages between the robots and a server computer. From the experiments, the author claims the robots act as expected in a reliable fashion. Aroca et al [51] developed a remote control of a SCARA robot via TCP/IP over a Web Server accessible in any computer connected to the internet. On the other hand, UDP was the protocol adopted by Garg and Kappes [52] due to its connectionless nature, which enabled a higher throughput rather than with TCP/IP.

2.1.5 Video feed

Although not an essential component in these type of interfaces, several modern remote control systems are being built with some form of video streaming from a camera on-board of the controlled device to the controller, mainly when a physical distance from end to end exists.

A video feed is made of a quick sequence of digital still images, providing the effect of continuum movement. Digital images can be binary (black and white), grayscale (one byte per pixel), colour (three bytes per pixel). In the latter, the most common colour spaces are: RGB (Red, Green and Blue), YCbCr (intensity, blue and red chrominance) and HSV (hue, saturation, and value) or four-channel (RGB + alpha channel – value that reflects transparency for each pixel) [53] . All these channels are 8 bit long. Therefore, depending on the type of image and according to the previous statement, a pixel will be made of 1 byte up to 4bytes, as shown on figure 1. The total number of pixels of an image depends on its resolution.

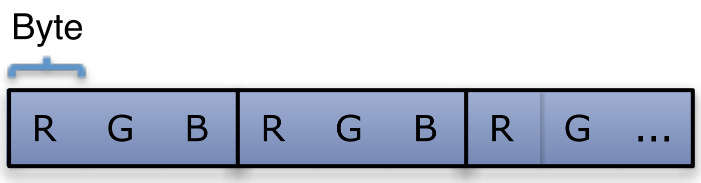


Figure 1 – RGB colour space format.

Any digital camera with USB connection can be instantly used to capture video frames by a computer. There are new approaches regarding this topic, such as the one from Ken [54], where the author streams video over Wi-Fi from an iPhone to any Apple Mac computer video program, or the use of Microsoft Xbox 360 new USB camera, the Kinect, on a computer [55]. These cameras vary on the previously explained image mode, resolution and frame rate.

Microsoft Kinect has revolutionized the digital camera domain, by combining an RGB camera with a depth sensor (infrared camera) in the same structure, which facilitated the distance detection to an object from a short distance (less than a meter) up to 3,5 meters. The direct consequence of this setup is the computational requirements that are much smaller. Its setup is greatly simplified and it can be operated in environments with varying illumination conditions [56].

The use of video streaming allows remote controlling of the object without a line of sight and its greatest uses are in the military, rescue and salvation fields, where the action might put one's life in danger. In [57], the author states that apart from the basic requirement of being built in such a way that is able to survive hostile environments, the robots must provide a video feed to the rescuers.

The unmanned aerial vehicle (UAV) is also where the video streaming is almost mandatory. With tasks like surveillance and monitoring, Ettinger et al [58] used the camera not only to see where the vehicle was heading, but also to create an automatic stability system based on algorithms applied on the footage. As described, these video systems on modern UAV projects are used for more than visually informing the controller, as the footage can be used to estimate the aircraft's attitude using a specific algorithm [59].

Comparing different kinds of interfaces, Fong et al [10] state that driving a robot from a remote control in an irregular or unexplored terrain is easier and better with the visual help of a video streaming.

In summation, most authors agree that video feed from the robot to the controller is well suited for mobile robots, especially those in which it is possible to lose sight of.

2.2 Commercial Solutions

After the success of several research projects about remote controlling a robot from a mobile device, some commercial products are already found in the market. Some research reveals that it was fast adopted by toy companies, but no results were found for serious applications though. It is difficult to say what company started this new trend of toys but one of the first successes was Parrot with its AR.Drone.

2.2.1 Parrot's AR.Drone

The wireless products manufacturer company Parrot released in 2010 the AR.Drone, a radio controlled quadcopter capable of being wirelessly controlled by an application on the iPhone, iPod Touch and iPad (and in the future on other non-apple devices). Originally distributed as a game, it was quickly adopted by the developers' community since the SDK was free and the system's characteristics unique in the market.

The drone presents two built-in cameras, which can be viewed in the controlling device screen. An ultrasound altimeter allows it to hover and it is compatible with augmented reality flying games and multiplayer interaction. It also features a 3-axis accelerometer, a 2-axis gyroscope and a single axis yaw precision gyroscope, which constitute the stability control [60].

Due to a dedicated Wi-Fi network created by the drone, the controlling device can receive a live streaming from both cameras, achieving a resolution of 640x480 pixels at 15 frames/s on the front one and 60 frames/s on the bottom one. This product was awarded the 2010 CES Innovations Award for Electronic Gaming Hardware [61].

2.2.2 AppToyz's Appcopter and AppRacer

Following the concept of the previous system, AppToyz offers a radio controlled helicopter and car, both much cheaper than the AR.Drone but with less hardware. Both toys interact with an infrared peripheral attached to the iPhone or iPod controllable by an application downloadable from AppStore and the display will only show basic control buttons [62].

2.2.3 Griffin's Helo TC

The company Griffin offers another helicopter driven through an iPhone application. It is very similar to AppToyz's Appcoter, lacking a lot of hardware comparing to the AR.Drone and the connection being performed through an infrared peripheral. This product is due to enter the market in the Christmas season, this year [63].

2.2.4 iHelicopters' iPhone Controlled Wall Climbing Car iW500

iHelicopter has recently released a car able to be driven on the wall due to a sealed vacuum between itself and the wall. It is controlled by an iPod, iPhone or iPad with a docked infrared transmitter adapter and an application downloadable from the Appstore. It uses the gyroscope motion to control the car and a slider to control the throttle [64].

Chapter 3

Experimental Setup

This chapter focuses on the actual system proposed in this thesis, with the description of all its subparts and details of the techniques used.

The setup of this system includes the controlling device, the iPod Touch 4th Generation, the laboratory's car-like robot FormulaUM and the netbook placed on it as a processing gateway, the Magalhães. This netbook is connected to the microcontroller that controls both the stepper motor for the front wheels and the DC motor for the rear wheels. There is also the Microsoft Kinect camera in the front of the vehicle, whose function is solely to capture video.

First the robot is turned on and afterwards the netbook runs the server program. When the computer is ready to accept connections the iPod application is launched. There will be several screens with options displayed in a friendly manner that the user will have to go through before it can successfully connect. From this point on the whole system acts as a cycle until the user disconnects or the connection breaks and it disconnects automatically.

The main cycle consists, firstly, of a frame captured by the Kinect camera being transferred in packets (due to TCP specifications) from the computer to the iPod until the whole frame is available with the last packet arrival. After this the iPod transfers the controller's data back to the computer, which is redirected to the microcontroller, which in turn actuates on the motors accordingly.

3.1 Processing Gateway and Peripherals Setup (server side)

Magalhães (Magellan in English) is the name of the netbook that is used by the robotic car FormulaUM. It was granted from its makers, the Portuguese company JP Sá Couto. Made originally for children in a deal with the Portuguese government, it has the processing potential to do the required tasks as well as the small form factor which enables an ergonomic fitting in the server side setup.

This netbook runs the Linux Operating System and the server software was developed in C++ language using the Code::Blocks integrated development environment (IDE) [65]. First the computer UART connection is initialized with the microcontroller at 9600 baudrate and sends the value for the gain (the standard value is 4, but can be easily changed before running the program); it then sets up the connection with the Kinect camera (figure 2) and allocates the necessary space for the images, through the ofxkinect library; afterwards it sets up the TCP connection by creating a TCP/IP socket, binding it to the chosen port (standard is 50000) finishing in a listening state waiting for incoming connections. At this point the program will block until the iPod connects with it.

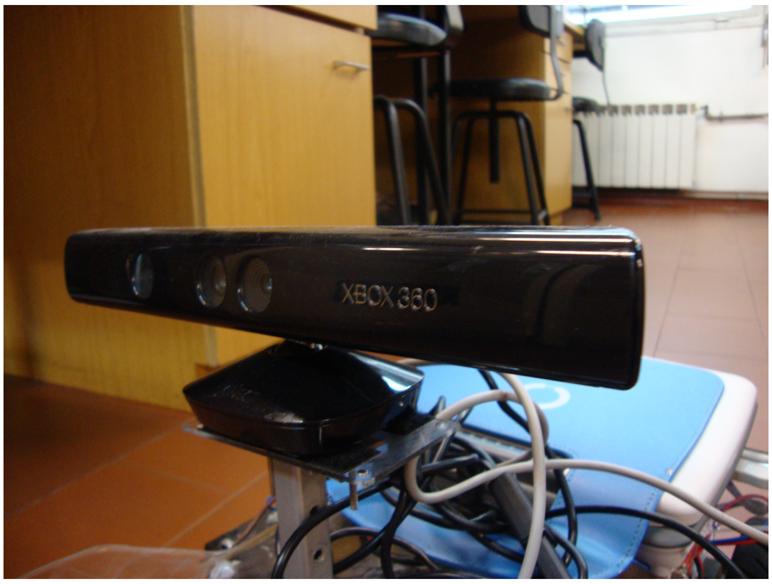


Figure 2 - Xbox 360's Kinect camera.

The mobile platform composed by the car, the Kinect camera and the netbook is presented in figure 3.

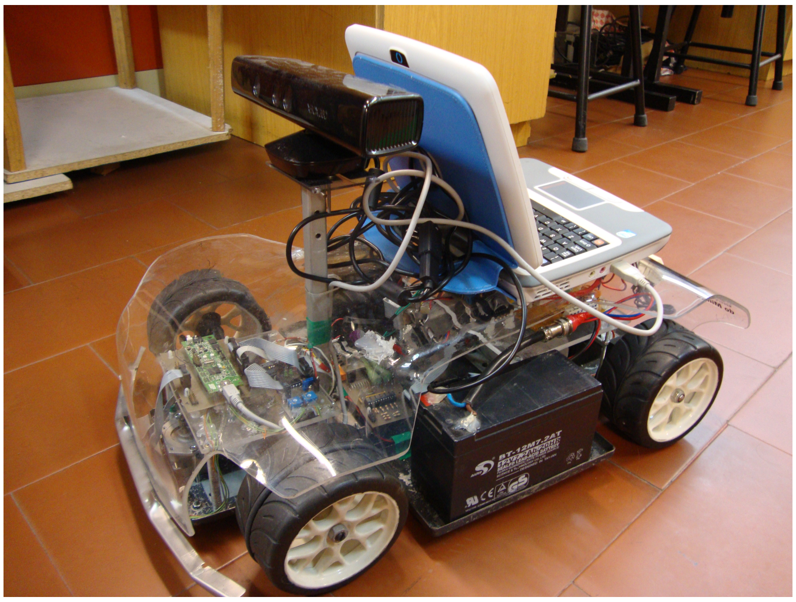


Figure 3 - The mobile platform.

3.2 iPod Setup (client side)

The iOS programs must be built in a complete Apple environment, which means an Apple computer or laptop and Apple's compiler: Xcode. For this project it was used a MacBook Pro 2.3GHz Intel Core i5 with Mac OS X 10.6.8 and Xcode 4.0.1. The program was written in Objective-C. The iPod touch running the final application is shown in figure 4.

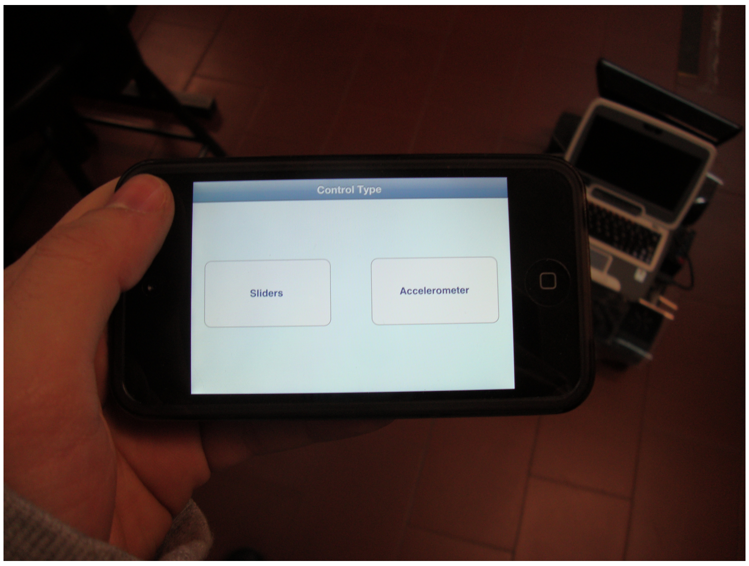


Figure 4 - The iPod application.

Since the graphical interface was a significant topic in this research, particular importance was given to its construction. From the start of the project it was decided that the program would run in landscape mode, i. e. with the iPod facing the user horizontally. This way there is more space in the screen for the slider controls, the video feed feels more natural in a widescreen format and the handling of the device more ergonomic.

According to [66], the average finger tap is 88x88 pixels (44x44 points in Apple's Retina Display format) thus the buttons should not be smaller than that size. In fact, since this application did not need many buttons in each screen, it was chosen to make the operation buttons 190x100 pixels in a rectangular form with round edges, creating equilibrium between the screen geometry and screen size, and at the same time, maintaining the graphical interface familiar to regular iOS users. Operation and program flow buttons, text fields, labels, screen titles, warnings, sliders and activity indicator symbols (which indicate that an action is underway) establish the bundle of graphical objects present throughout the program.

The application developed for the iPod Touch has a total of four screens, each with a specific function. A basic methods diagram of this setup is shown in Figure 5. In the following subchapters, the screens will be exhaustively studied and its respective method diagrams will be displayed on figures 6, 8, 15 and 18 to help the reader visualize the flow of the program. At the same time the triggered methods will be mentioned when appropriate after the action that triggers it. A list of the methods is presented in the Annex section.

The screens are completed with a top bar enclosing its title in the center and a program flow button on the left, which leads the user to the previous screen. Occasionally, an extra program flow button will appear on this same bar on the right, forwarding the user to the next screen; this only happens when strictly necessary and is thoroughly explained in the following subchapters. Finally the application has a programmed protection against automatic locking and screen light dimming, so the screen can be untouched for an infinite amount of time without any threat to the car control.

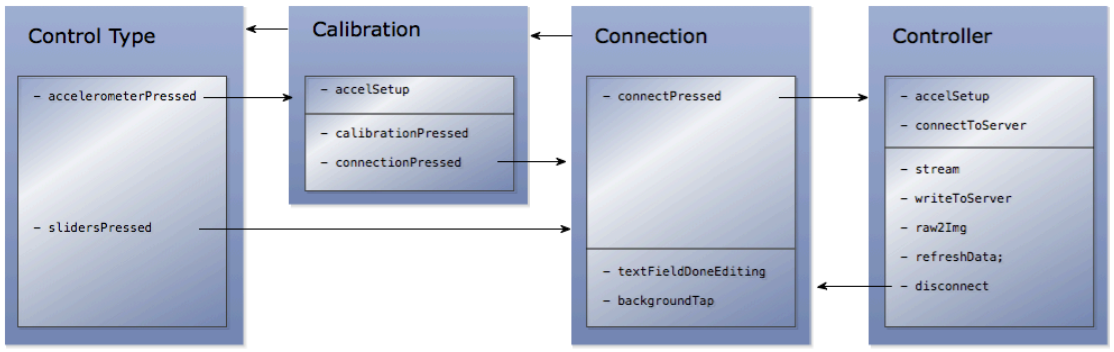


Figure 5 - iPod general setup method diagram.

3.2.1 Control Type Screen

The way the user interacts with the car can be done through two alternative controls: the sliders or the accelerometer. This choice is done in the first screen, the Control Type screen (showed in figure 7), which affects the logic of the afterwards screens: if the sliders are chosen (-*slidersPressed*), the application goes directly to the connection screen, but if the accelerometer mode is chosen *(-accelerometerPressed),* a calibration of the sensor will be required, thus forwarding to the Calibration Screen.

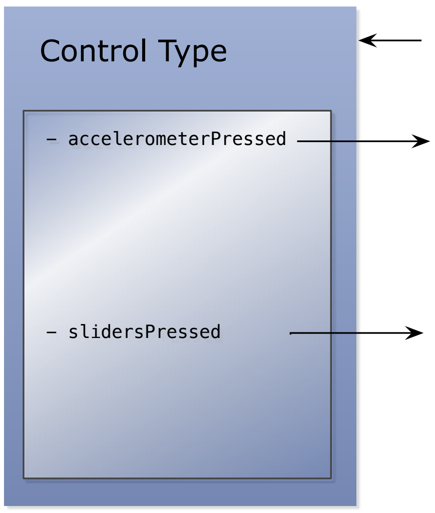


Figure 6 - Control Type screen method diagram.

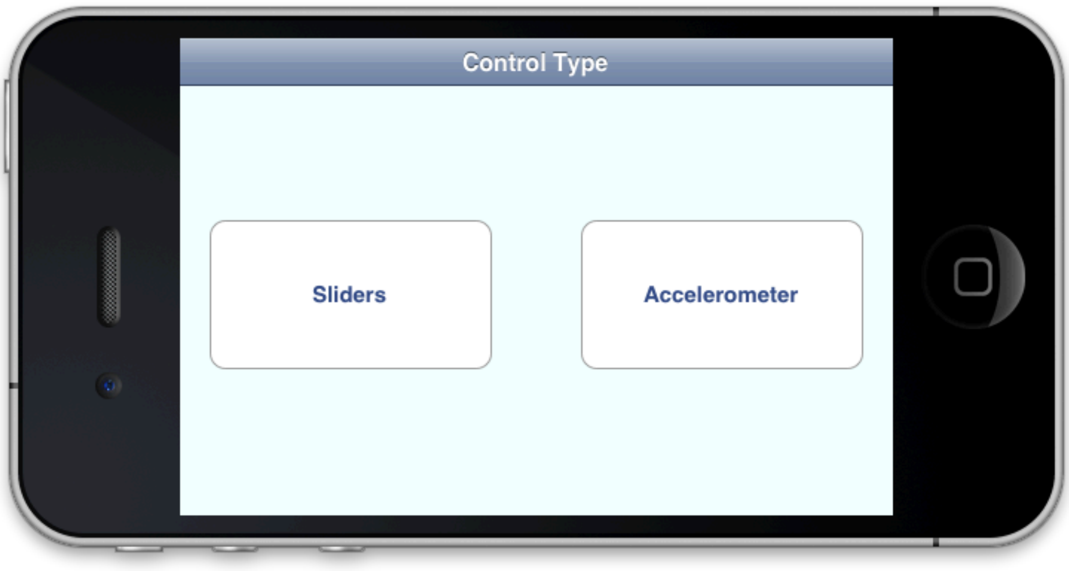


Figure 7 - Control Type screen.

3.2.2 Calibration Screen

This screen appears, as described above, only when the accelerometer mode is chosen. Right after the view appears, the program sets up and starts the accelerometer (-*accelSetup*) with data updates with a frequency of 1/60 second.

It then allows the user to hold the device in a comfortable position and, after pressing the calibration button *(-calibrationPressed)*, it records the offset of the X-axis and mark that position as the null position, which means the car will receive a 0 value for torque if the device is not tilted from that point on.

In this screen it was decided to have a different approach referring to the way it is forward to the next screen: because the first calibration might not mean that the calibrated position is ideal to the user, the forwarding to the Connection screen does not happen automatically after pressing the operation button, similarly to the previous screen. A new program flow button appears instead in this screen, on the right side of the top bar; firstly showing up disabled, it becomes enabled after pressing the calibration button for the first time. This way the user can calibrate repeatedly until the device's position becomes perfect and can advance to the next phase of the setup after pressing the Connection button on the top bar (-*connectionPressed*).

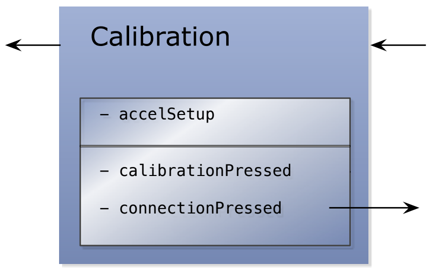


Figure 8 - Calibration screen method diagram.

Figures 9 and 10 demonstrate the two phases of the Calibration screen.

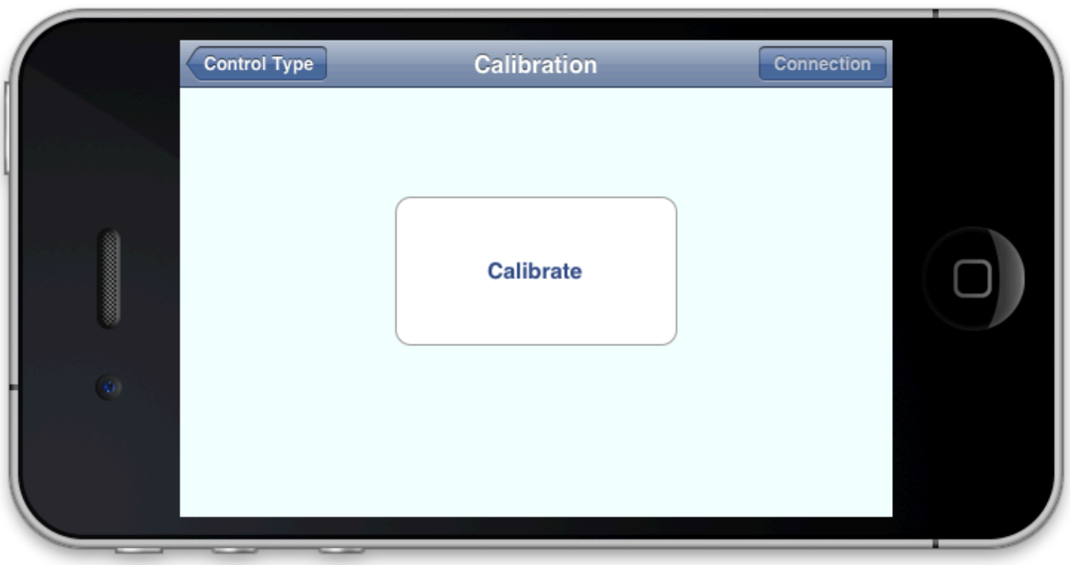
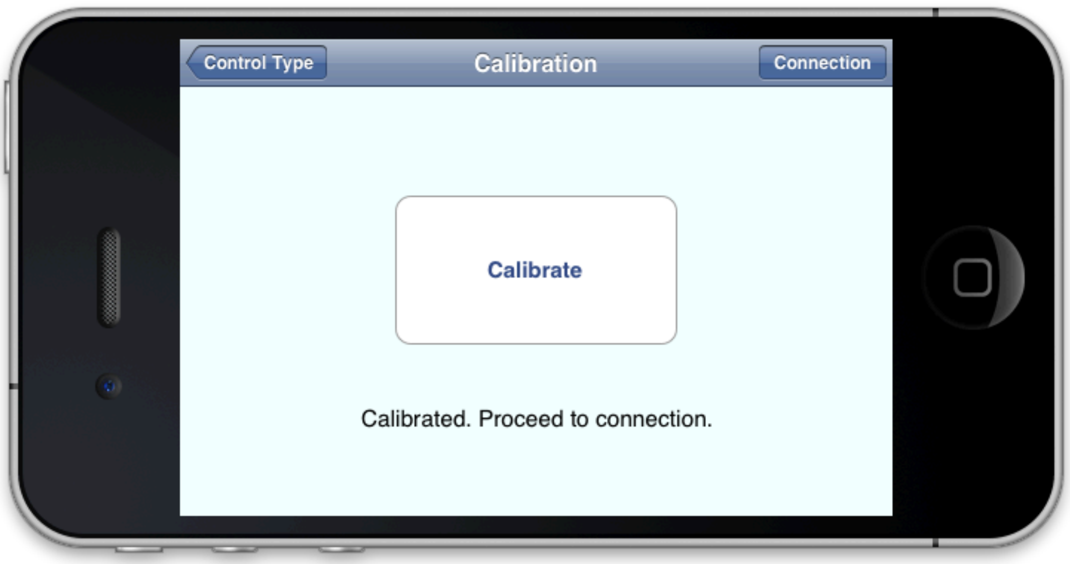


Figure 9 - Calibration screen before pressing the "Calibrate" button.



Calibrated. Proceed to connection.

Figure 10 - Calibration screen after pressing the "Calibrate" button.

3.2.3 Connection Screen

The Connection Screen is the last before the screen where the user actually controls the car. This screen is displayed in figure 11.

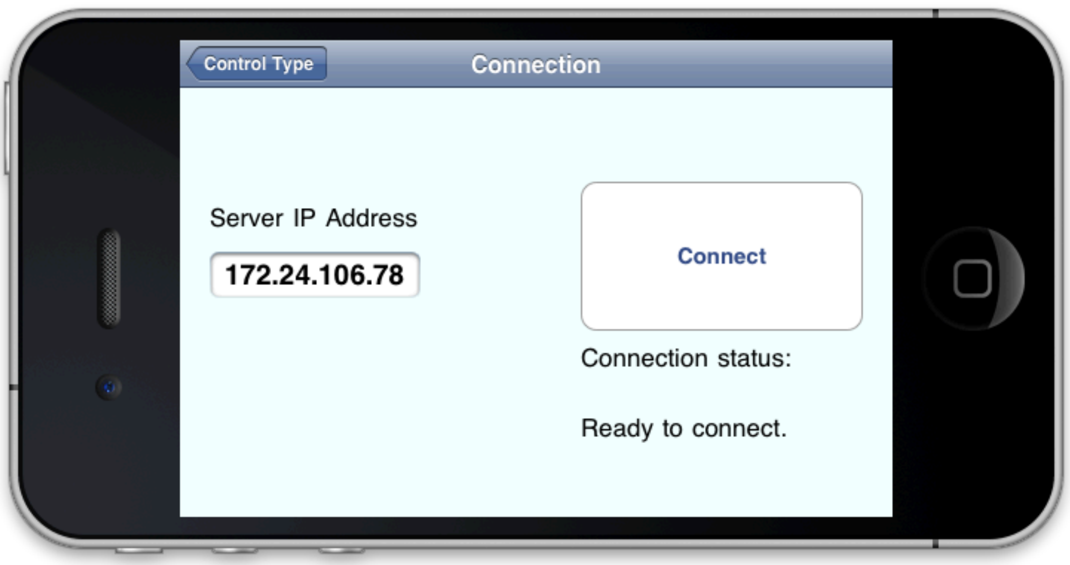


Figure 11 - Connection screen.

In this screen the user has to set the IP from the server in a textbox duly noticed and then press the connect button. This IP has to be checked in the netbook every time it reconnects to the network, for it is a dynamic IP. This screen was designed in a way that, when the user presses the textbox to input a new IP, a keyboard appears from below and the whole view rises the necessary number of pixels, so that the text box is always visible to the user. This can be perceived in figure 12.

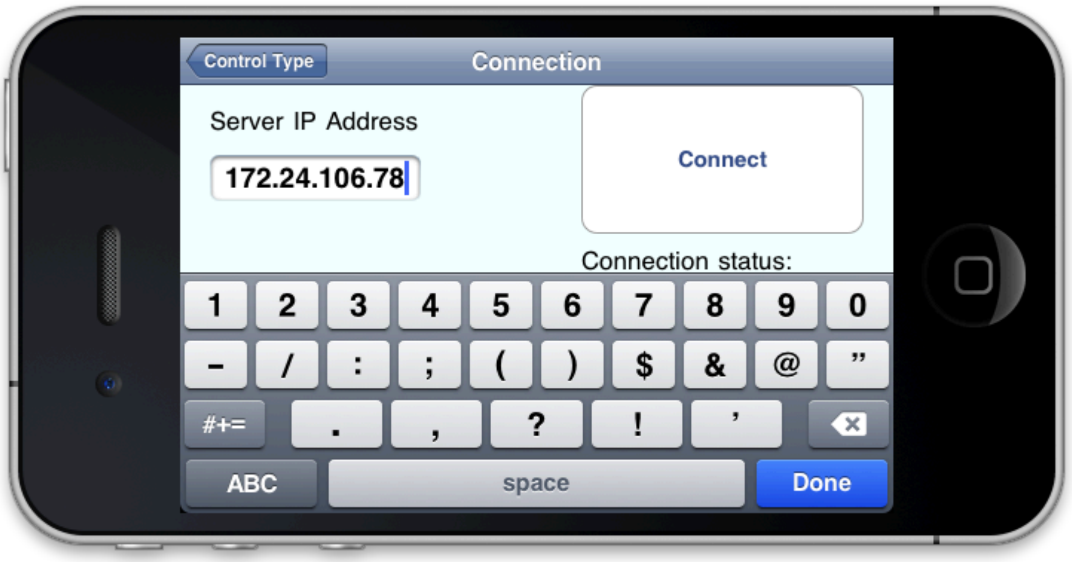


Figure 12 - Connection screen with the server IP textbox highlighted.

Whenever the textbox is dismissed the view goes back to its original position, which can be done by pressing the keyboard's *done* button (*-textFieldDoneEditing*), another button or background of the view (*-backgroundTap*).

After the user has pressed the connect button (*-connectPressed*), the control screen will appear without controls or image, just a label stating "Connecting..." and Apple's spinner (a moving sign which users will understand as an action taking place and to wait for something to happen in the screen) in the center of the screen and the disconnect button on the left end of the top bar; at the same time the code responsible to connect to the server runs in the background. It is possible to cancel this connecting phase by pressing the disconnect button. This phase is shown in figure 13.

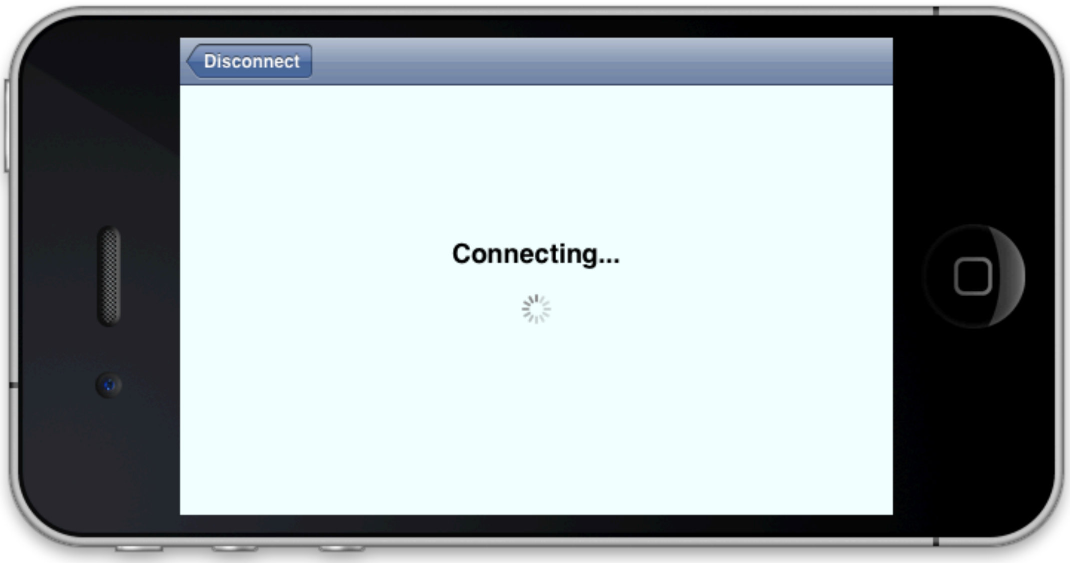


Figure 13 - Control screen while trying to connect to the netbook.

If the connection is successful then the application will keep in that screen with its associate controls, but if not it will go back to the Connection Screen with the alert showed in the figure 14. The user has then to dismiss the alert pressing the Dismiss button, which will reenable the IP text box and all the buttons.

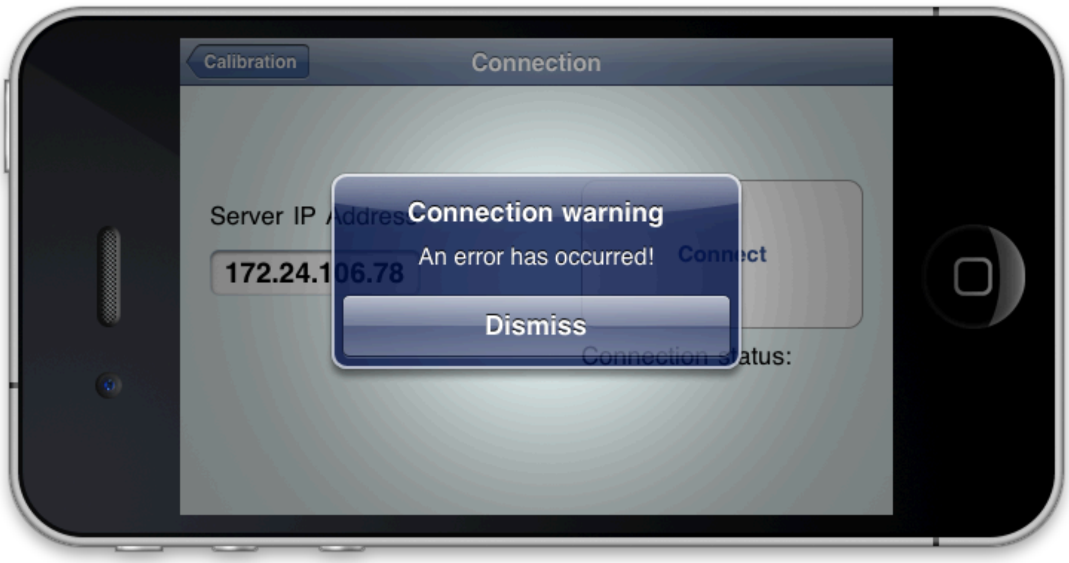


Figure 14 - Connection screen showing a connection warning, after a failed connection.

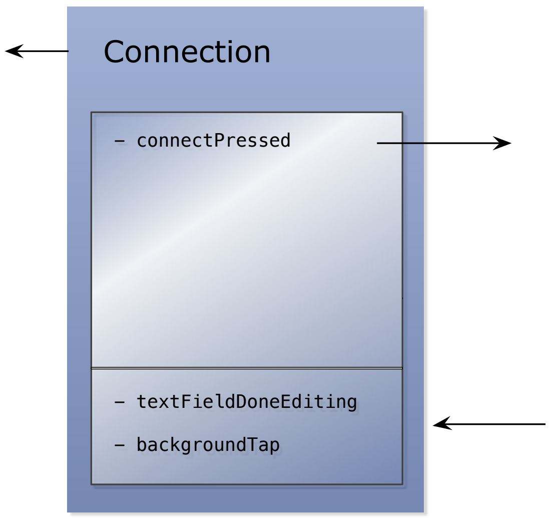


Figure 15 - Connection screen method diagram.

3.2.4 Control Screen

After all this screen's initial setting up (especially the accelerometer setup (-*accelSetup*) with the same characteristics as in the Calibration screen) and a successful connection (after the method *-connectToServer*), the Control screen appears. Depending on which of the control types was chosen, this screen will be displayed differently: with the accelerometer selected it will only show the images coming from the camera (*-raw2Image*), for there is no reason for any extra objects to be shown. On the other hand with the sliders elected there will be, apart from the background images, two sliders, one on each side; the left one in a vertical position controlling the torque, and the other horizontally controlling the angle of turn of the car. These two screen displays are showed respectively in figures 16 and 17.

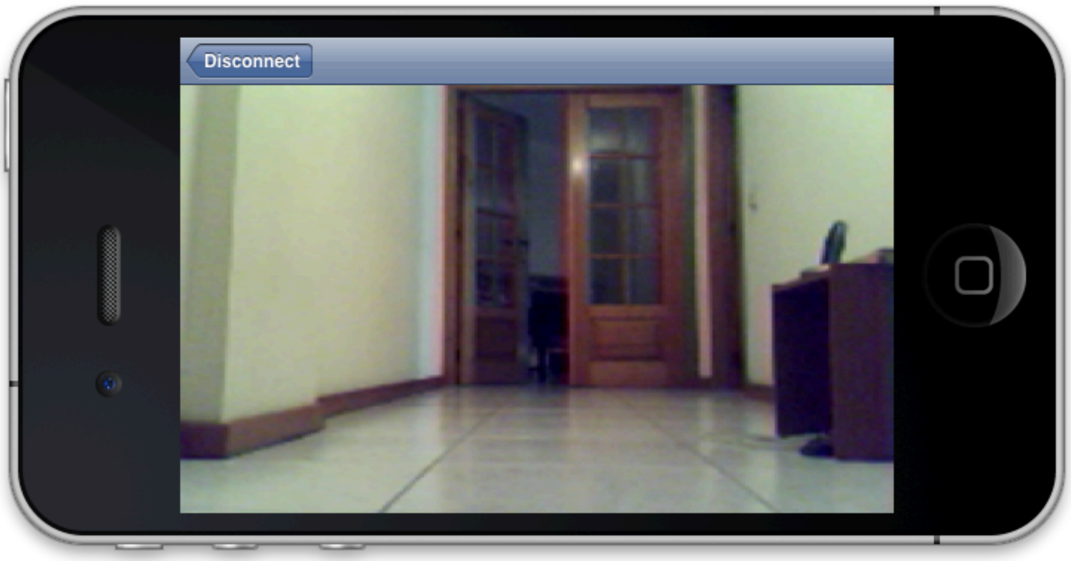


Figure 16 - Control with the accelerometer.

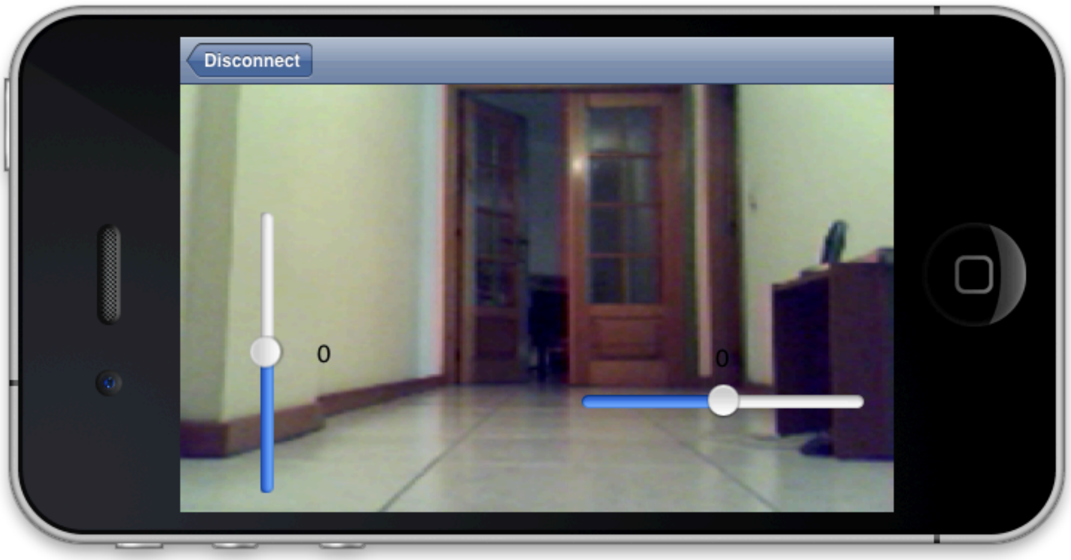


Figure 17 - Control with the sliders.

The user can press the button to go to the previous screen in the top bar, which in this case is labeled "disconnect" (*-disconnect*), disconnecting the iPod from the netbook and putting the car to a stop; on the iPod this button will bring the application back to the Connection screen with the warning presented in figure 19. If the server closes the connection, the iPod application will go back to the connection screen displaying the warning showed on figure 20. The remaining methods will be explained in the subchapter 3.4, for they belong to the system cycle after connecting section.

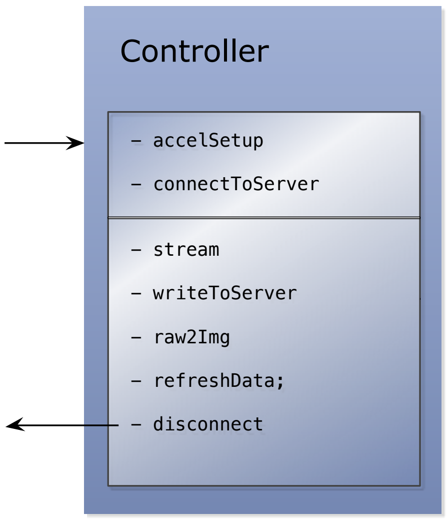
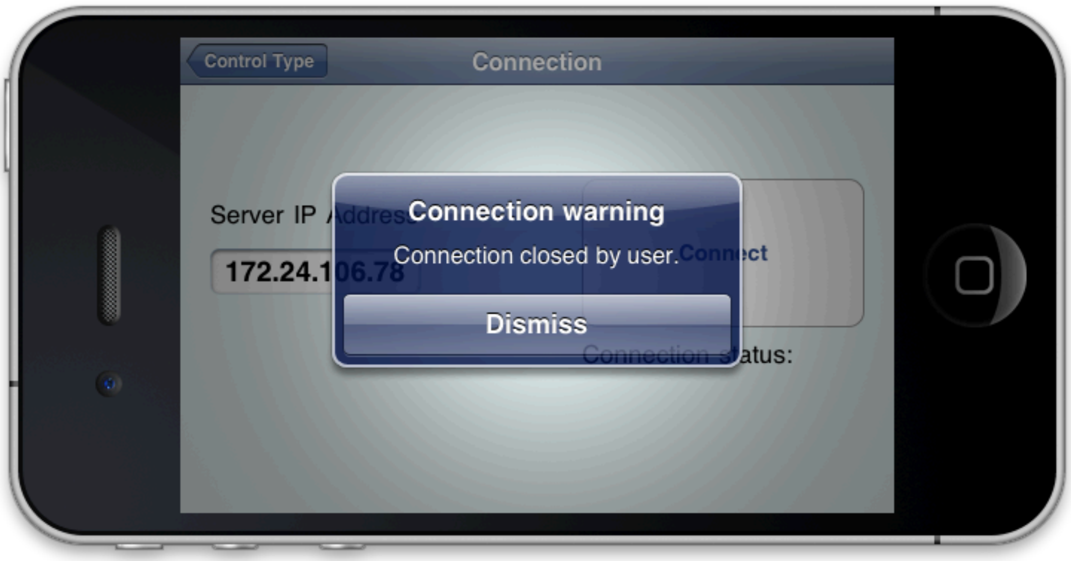


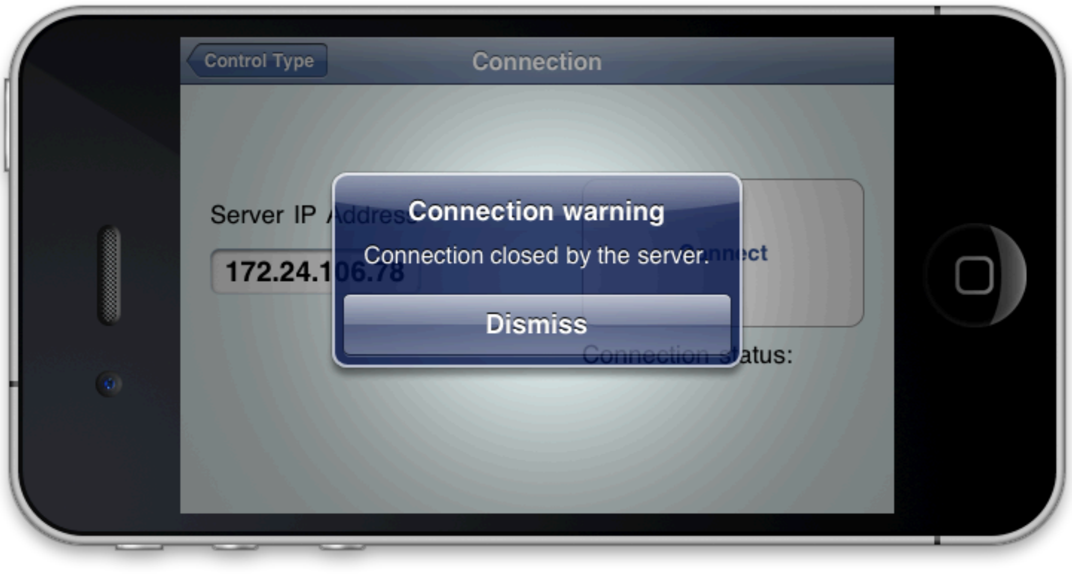
Figure 18 - Controller screen method diagram.



Connection warning

Connection closed by user.

Figure 19 - Connection warning when connection closed by the user.



Connection warning

Connection closed by the server.

Figure 20 - Connection warning when connection closed by the server.

3.3 Connection

The netbook is set up for the connection as the server in TCP/IP mode and binds to the predefined port number 50000. Immediately after, it enters a frozen state waiting for an incoming connection.

On the other hand the iPod is set to be the client, configuring input and output streams to receive and send data through it. After pressing the connect button on the Connection screen, the iPod tries to connect to the server on the given server IP address and already given port number. If successful, the system enters in the main cycle; if not, the iPod returns to the previous screen showing a warning, while the server remains waiting for a connection.

There is no need for extra programming for errors check, because the TCP protocol already implies reliability due to an automatic and hidden acknowledge message reply system.

The iPod manages the connection with events called to a method (-*stream*) (much like interruptions) and from a total of 7 events available in NSStream.h, 5 are used in this application: NSStreamEventNone happens after one minute trying to connect, and the server not being responsive; NSStreamEventOpenCompleted is fired when one of the two streams (income and outcome) has been opened, which in this matter means the connection was successful; NSStreamEventHasBytesAvailable means that there are bytes incoming from the connection (image data in this case); NSStreamEventErrorOccurred and NSStreamEventEndEncountered lead the connection to the same disconnect state, the first one due to an error external to the application and the second one because the connection was closed either by the iPod or the netbook.

3.4 System Cycle after connecting

After the successful connection of the two devices, the system enters in the main program cycle. This part is schematized in figure 21.

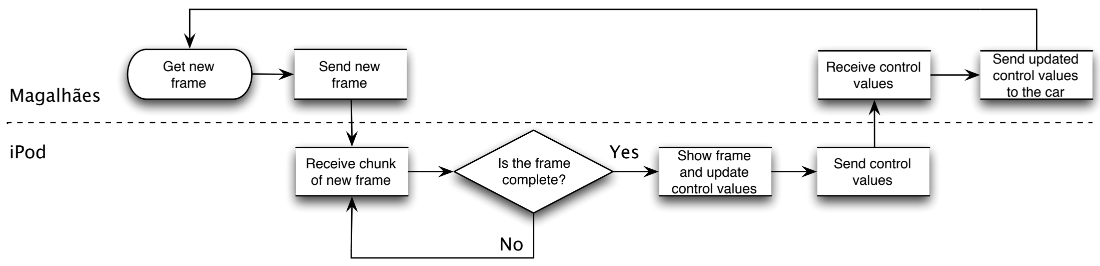


Figure 21 - Main program cycle.

Once the connection has been successful, the netbook grabs a new frame from the Kinect camera and after some processing (explained in the next subchapter) forwards it through the Wi-Fi connection to the iPod. Since the protocol used is TCP/IP, the data sent will be separated in packets, therefore the same reception event will be triggered several times (-*stream* with the event *NSStreamEventHasBytesAvailable*) and the bytes received in each event will be added to the image buffer, until the total number of bytes received reaches the one defined for the image.

Once this happens the iPod will undergo some image processing (*-raw2Image* - explained in 3.5) in order to properly display it on its screen. While this happens the netbook waits for a reply, which comes, right after the picture is shown on the iPod, as the form of the updated values of the controls (*-writeToServer*).

Finally the netbook will transfer those values to the microcontroller, which will in turn make the FormulaUM's motors actuate accordingly, and then restart the whole cycle by obtaining a new frame from the Kinect.

3.5 Video feed

The first attempt at creating the video feed was through UDP protocol, because this would allow a quicker data transfer disregarding the general quality of the feed (frames loss, duplicate frames, or other general UDP problems), which is well suited for video feed. This seemed not to work well, since the netbook would send the data without any response from the iPod, and on the other end the iPod seemed to not receive any data. Therefore an attempt at TCP protocol was made, which provoked a slower, but working communication.

The Kinect camera grabs RGB frames of 640x480 pixels, which proved to be too much for the iPod's screen and the TCP connection. In order to solve this issue, several steps stand between the frame acquiring by the Kinect and its display on the iPod's screen:

1) In the netbook, the original frame goes through resizing from 640x480 pixels to 133x100 pixels or 240x180 pixels (resolution predefined before running the system) and then a horizontal mirroring function (cvFlip()), because during the tests it was found that the image would reach the iPod horizontally inverted otherwise;

2) Once the whole frame is retrieved by the iPod in a string of characters fashion, the RGB image was not drawn accordingly to the frame sent by the netbook (due to Xcode's peculiarities), so in order to fix it, it undergoes a cyclic modification to add an alpha channel after the third byte of each pixel. To this alpha channel was given the null value in all pixels since the chosen mode for creating the image according to the data buffer ignored the Alpha's value. It was then proved to work with the added alpha channel, displaying the content correctly. This transformation is schematized in figure 22.

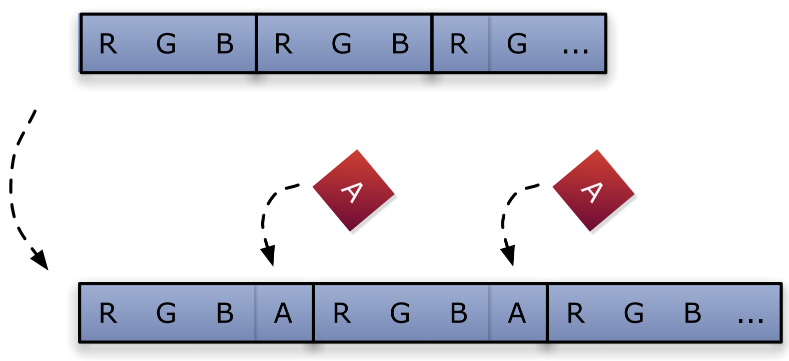


Figure 22 - RGB to RGBA process.

3) Since the frame is, at this point, still in its raw format (a string of chars), it is created an image from scratch with the data provided (and pre-chosen resolution).

4) The image created is finally stretched to 480x276 pixels, filling the iPod's background, allowing a good visualization by the user.

3.6 Control

Whether with the accelerometer or with the sliders, after showing the frame on the screen, the iPod updates and converts the control values to a string of chars to send it afterwards to the netbook. This string is formatted in the following way: "torqueValue angleValue" and both torque and angle values have a range of 61: from -30 to 30.

The netbook will receive this string and split it in two signed variables, one for the torque value and another for the angle value. The program then checks whether the values are negative or positive applying simple comparisons with if/else cases. Independently of these comparisons results, the netbook will have two send instances through USB to the microcontroller for both motors:

- For the torque values, the first instance will be the signal of the values. The '+' sign will indicate that the movement will be forward and the '-' sign will tell the microcontroller that the movement will be backward. For the angle values, the same logic applies, as 'l' will mean the wheels will be turned to the left and 'r' to the right.

- The second instance will indicate the value to apply on the rear wheels for the torque or the front wheels for the angle.

For the torque value, while positive it will move forward, as with the negative signal the car will go backwards and the zero value makes the car naturally come to a stop. The angle value, if positive will make the wheels turn to the car's right and to the other side when with the negative sign. In this case, the zero value will make the front wheels face parallel to the car.

In an attempt to make this project relatively universal, a library will be created to act as an intermediate from the netbook to the car. This library will contain the standard data transfer functions to the microcontroller in which to send a command from the netbook, one will only have to use the send data function twice, like previously explained. This way any remote control that can connect to the netbook, will easily and in the same fashion control the car.

3.7 Disconnect

The last part requiring explanation is the disconnecting of the Wi-Fi connection. There are two ways for this to happen, which are explained as follows.

3.7.1 Willful disconnection

The iPod application has a button for this purpose, in which when pressed by the user closes the connection and returns to the Connect Screen.

On the netbook side, once that button is pressed, the zero value for torque is sent to the microcontroller to stop the speed motor and then returns to the "awaiting for connection" state. The user can then reconnect if he will.

3.7.2 Abnormal / sudden disconnection

There is the possibility that at some point during car maneuvers, the connection might have problems, e.g., moving away from the hotspot and losing the connectivity, thus disconnecting both ends. When this happens the netbook will get an error resulting from a “peek” function, where it checks whether there is a receptor or not every time the cycle restarts. This error will lead to sending the null value to the microcontroller, making the car stop and returning to the “awaiting for connection” state. On the other end the iPod will have either the NSStreamEventEndEncountered event or the NSStreamEventErrorOccurred event fired, which will terminate the connection and return to the previous screen (*-disconnect*).

Chapter 4

Methodology

In order to evaluate the system proposed in this thesis, some experiments were undergone. These experiments will be described after the hardware and software environment description.

The remote controller was an iPod Touch 4G released on September 1, 2010 with updated iOS 4.3.3 operating system version. It measures 111 × 58.9 × 7.2 mm weighing 101 g. Its display is a LED-backlit LCD with a 2:3 aspect ratio and a resolution of 960x640 pixels at 326 ppi. It has a 1Ghz ARM Cortex-A8 processor, 8GB of storage and IEEE 802.11 b/g/n connectivity. Among many other technical specifications not directly influencing this thesis, the iPod contains a three-axis gyroscope and an accelerometer.

The controlled device was the car-like robot FormulaUM previously built in the laboratory. Although the car is not the centerpiece of this thesis, it is of tremendous complexity, as it was originally built from scratch for autonomous driving competitions. For this reason its size has to obey the rules, thus measuring: 55 cm x 36 cm x 35 cm having the weight of 14.6 Kg with the netbook on top of it. On the front it has a 24V bipolar stepper motor to control the steering and on the rear a 120W DC motor to control the car's torque and direction. Reverse is achieved through a relay that inverts the motor's polarity. There is also a magnetic break with a peak current of 2A. The brain of the car is the Atmel 8-bit microcontroller AT90USB128 with USB interface. It works at a frequency of 8MHz and from the six bi-directional ports available, one is used for the stepper motor and another for controlling the DC motor, the breaks, the relay and the cooling fan. A third port is prepared for signal acquisition from an encoder for the DC motor, limit switchers for the stepper motor, and line sensor. Finally the car is powered by two lead-acid batteries that support 10A, the car's peak current.

The intermediate between the iPod and the car was a small Portuguese netbook, Magalhães, version 1.1 with Intel Atom N270 dual processor at 1.60GH – 32 bit and 1Gb of memory, running on Ubuntu 10.10 (2.6.35–28 generic-pae Kernel). It has two USB 1.1 ports and its network adaptor has IEEE 802.11 b/g connectivity. It connects to the car's microcontroller through one of the USBs and to the Kinect through the other.

The platform created will be tested in two different network environments:

- One public network (for students and teachers from the university) with unknown characteristics;

- One private network constituted of an Asus WL-520gC router operating in the 2.4GHz band with 125 High Speed Mode and a throughput of 54 Mbps.

On both of these network environments two image resolutions sent by the netbook will be evaluated:

- 133x100 pixels (considered low resolution);

- 240x180 pixels (considered high resolution).

For each of the four test setups, five trials will be conducted in order to achieve a credible average. On the whole 20 runs will be effectuated by a volunteer student so to evaluate the experience as objective as possible.

The aspects under evaluation are:

- The number of frames per second (FPS) achieved by the netbook; with the function ofGetFrameRate() from the Openframeworks API, the program automatically displays the number of FPS recorded by the Kinect camera. Notice that this number does not represent the maximum FPS allowed by the camera, but the real FPS allowed by the network, because a new frame is only acquired after the wireless transmition of data between the two devices takes place.

- The latency of the communication between the iPod and the netbook; the main cycle has two data transfers: the frame from the netbook to the iPod and the control data the other way around. Since the frame's data is much bigger than the control data, the transfer time of the first will be measured. This will be done on the iPod as follows: a NSData variable collects the system time right before receiving the first packet of data; and after receiving all packets attaining the limit of the image buffer a NSTimeInterval variable is created returning the time interval between that variable and the first NSData variable.

- The volunteer's evaluation regarding the control of the car; the scale ranges from 1 – almost impossible to drive, 2 – very difficult to drive, 3 – average driving conditions/drivable, 4 – relatively easy to drive, to 5 – very enjoyable to drive.

Through these experiments it will be possible to verify if the system is both stable and reliable, thus achieving the proposed goal. The results will be presented and analyzed in the following chapter and summarized in a table, before the conclusion chapter.

Chapter 5

Analysis And Discussion Of Results

This section presents the results of the experiments described in the previous chapter. Throughout the trials it was possible to identify the superior and the deficient characteristics of the system, as well as advantages and disadvantages of this platform comparing to similar solutions. The results and analysis will be presented following.

5.1 Tests Results

5.1.1 Public network

The tests in this network revealed that it was only possible to get up to 4 frames per second and some times not even a frame per second, especially in the case of high image resolution. Table 1Table 2 present the results from both sets of experiences in this network setup.

Table 1 - Results obtained with 133 x 100 resolution video feed on public network.

Trial	**Frames Per Second**	**Connection Latency (seconds)**	**User's Evaluation**
1	4	0,068	3

2	4	0,072	2
3	4	0,104	1
4	3	0,082	2
5	4	0,108	1

Table 2 - Results obtained with 240 x 180 resolution video feed on public network.

Trial	**Frames Per Second**	**Connection Latency (seconds)**	**User's Evaluation**
1	3	0,272	2
2	3	0,218	1
3	3	0,235	1
4	3	0,312	2
5	3	0,272	1

The average number of FPS achieved with the lowest resolution was 4 and the connection latency was of 0,086 seconds. When in high resolution the average number of FPS was 3 and the connection latency was of 0,259 seconds.

Albeit these results seem propitious for an enjoyable and easy control, these tests also revealed that due to the high latency between the devices and random network glitches, the driving presented very difficult or, especially in high-resolution almost impossible, according to the user.

5.1.2 Private network

These tests were effectuated as proof of concept, so to understand the capabilities of the system in a near-perfect environment. Table 3Table 4 show the results obtained from the experiences on this network.

Table 3 - Results obtained with 133 x 100 resolution video feed on private network.

Trial	Frames Per Second	Connection Latency (seconds)	User's Evaluation
1	7	0,043	5
2	8	0,044	5
3	7	0,046	5
4	8	0,048	5
5	8	0,043	5

Table 4 - Results obtained with 240 x 180 resolution video feed on private network.

Trial	Frames Per Second	Connection Latency (seconds)	User's Evaluation
1	5	0,129	4

2	4	0,131	4
3	4	0,142	4
4	4	0,141	4
5	4	0,138	4

When in low resolution the system achieved 8 FPS and a connection latency of 0,044 seconds, while in high resolution it was registered 4 FPS and a connection latency of 0,136 seconds.

The tests demonstrated that in the private network the frames appeared fluently on the iPod with little latency between the iPod instructions and the FormulaUM actuation. Apart from the obvious differences in image quality and frames per second, a little more latency was perceived by the user when in high resolution, albeit still highly drivable.

5.2 Results Summary

The average data resulting from the tests are summarized in the table 5.

Table 5 - Average of the results from the trials.

Test	**Network**	**Resolution**	**Frames Per Second**	**Connection Latency (seconds)**	**User's Evaluation**
1	Public	133 x 100	4	0,086	3
2	Public	240 x 180	3	0,259	1

3	Private	133 x 100	8	0,044	5
4	Private	240 x 180	4	0,136	4

5.3 Studies Comparisons

Overall with a minimum latency value of 44ms, this system beats the one from [51], demonstrating an average time of 600ms from the PDA command until the robot's execution. For this minimum latency value, the image data transmission comprised 39900 bytes (100 x 133 x 3 channels) resulting in a total of 906Kbytes/s, which surpassed the maximum transfer of 33 bytes/sec between the robots in [50] and the network throughput of 20Kbytes/sec in [48]. These results also show that this solution greatly surpasses the throughput of Bluetooth and Zigbee [40].

On the other hand when presented with the worst setup scenario – high image resolution feed and on the public network – the system proved inefficient with very difficult handling. It achieved 500Kbytes/s, 52% of the maximum speed achieved with the previous setup.

In [24] the author concluded that, according to several users, the Wiimote controller was easy, intuitive and fun to operate with when controlling a robot, although with the problem that if it was trained for right hand gestures, it would not work correctly for left-handed people. Whereas the Wiimote requires both wrist and arm movement as well as great coordination and understanding of the right gestures for the task, the PDA interface presented in [10] required little understanding of the controlling process with reported high usability from both novice and expert users. The controller presented in this thesis follows this last interface and was reported by the test user as intuitive with easy menu navigation and robot maneuver, thus befitting the purpose explored in Chapter 2.

Chapter 6

Conclusions and Future Work

The goal of this thesis was to make an application for a mobile device capable of controlling the Laboratory's car-robot FormulaUM with an easy and intuitive interface.

The background information on Chapter 2 was undeniably helpful for the entire planning and construction of the application. Not only indicated the best interfaces available for the kind of device in question, but also, because of the extensive research, it was possible to understand that one of the most important aspects in a project like this is the interaction between the user and the mobile robot. The type of connection between the mobile device and the FormulaUM was another topic that deserved plenty of attention as the success of the project depended greatly on the choice of the right networking characteristics for this interaction.

In chapter 3, the theory learned on the previous chapter was put into action on building the iPod/netbook interface and an emphasized look over the iPod application was given. All interface screens from the iPod were demonstrated and it was also described the cycle of operations of that interface.

A series of experiences were conducted to determine the system's viability and compare the output in different network conditions. In these experiences the number of frames per second, connection latency between command and actuation of the motor and user opinion were the variables to evaluate. The methodology behind these experiments was described in Chapter 4.

In the previous chapter, the results were posted and discussed, with positive outcomes from the private network tests.

The results from the experiments demonstrate that the goal of this dissertation has been achieved, with a solid proposition for remote controlling a car-like robot with a mobile device application. Within public network, the system did not show its true potentialities, due to connectivity latency and glitches. However it demonstrated secure aptitude in the private network achieving 4 fps with high image resolution and 8 fps with low image resolution and an interaction between the user and the car in almost real-time. These results depended merely on the network characteristics and may not be achieved with different network configurations.

6.1 Future Work

Although the goal of this dissertation has been achieved, it is known by the author that this project has its flaws and that some modifications or add-ons both in the technology and the software would be of great benefit.

Firstly, the project lacks a solid car-like robot piece with optimized software running on the microcontroller, which slew down and hardened the difficulty on working on it. Since the microcontroller on the car was still not programmed to deliver car status at the time (e.g., battery conditions, lights warnings, et cetera), the iPod application lacked some feedback information from the car, which would be an added value to the final product.

The greatest problem in this project was probably the latency of the network that in some experiments compromised the handling of the car. Because the network protocol chosen was TCP/IP, due to its intrinsic characteristics, the data was obliged to get from one end to the other regardless of how slow the connection could be. Since the latency was almost due to the large size of the image data, if the image and the controls data were separated into two different streams, the handling of the car could have been flawless: the driving commands would be sent via TCP and the video feed via UDP.

With more time in hand, a compression/decompression algorithm would probably have solved most of the issues experimented during the public network tests, for with less data to transfer, faster the communication would be.

Extra features can be implemented in the future, such as a Proportional, Integral and Derivative (PID) controller to achieve a near-optimal control over the car with encoder feedback applied to the motor [67], [68]; and instead of displaying the original image, the final product could have some image processing either on the netbook with the image processing library openCV as demonstrated in [53] or directly on the iPod [69]. This way it could be used for rescue missions and dangerous places for man to go to. Finally the server could be programmed as a web service, making its control available in any device with an Internet connection and a web browser [70].

Bibliography

[1] L. Pacheco*, et al.*, "Mobile Robot for Autonomous Golf Balls Picking," presented at the Controlo'2008 - 8th Portuguese Conference on Automatic Control, UTAD, Vila Real, Portugal, 2008.

[2] M. Gonçalves, "Visão por computador aplicada a plataforma omnidireccional para robô de serviços em casa," Electronics Engineering Department, Universidade do Minho, Guimarães, 2010.

[3] J. Rodrigues, "Plataforma omnidireccional para robô de serviços em casa," Electronics Engineering Department, Universidade do Minho, Guimarães, 2010.

[4] D. Oliveira, "Braço robótico manipulador para aplicação em robô de serviços," Electronics Engineering Department, Universidade do Minho, Guimarães, 2010.

[5] C. Machado*, et al.*, "Robot Football Team from Minho University."

[6] F. Ribeiro*, et al.*, "Robot Orientation with Histograms on MSL," *RoboCup'2011 International Symposium,* 11 de Julho de 2011 2011.

[7] G. Lopes*, et al.*, "Catadioptric system optimisation for omnidirectional Robocup MSL robots," *RoboCup'2011 International Symposium,* 11/07/2011 2011.

[8] F. Ribeiro. (2007) Cadeira de Rodas Omnidireccional. *Robótica*. 50-51.

[9] A. J. Dix, *Human-computer interaction*. Harlow, England; Munich [u.a.]: Pearson, 2007.

[10] T. Fong*, et al.*, "Advanced Interfaces for Vehicle Teleoperation: Collaborative Control, Sensor Fusion Displays, and Remote Driving Tools," *Autonomous Robots,* vol. 11, pp. 77-85, 2001.

[11] E. Havir. (2011, 24/10/2011). *Controlling the AR Drone with Surface*. Available: http://blogs.msdn.com/b/surface/archive/2011/01/27/controlling-the-ar-drone-with-surface.aspx

[12] Appcelerator and IDC, "Q3 2011 Mobile Developer Report," 2011.

[13] comScore. (2011, 24/10/2011). *comScore Data on Apple iOS Featured at WWDC*. Available: http://blog.comscore.com/2011/06/comscore_data_on_apple_ios.html

[14] R. LeMoyne*, et al.*, "Implementation of an iPhone as a wireless accelerometer for quantifying gait characteristics," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 3847-3851.

[15] R. LeMoyne*, et al.*, "Implementation of an iPhone for characterizing Parkinson's disease tremor through a wireless accelerometer application," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 4954-4958.

[16] Y. Fujiki, "iPhone as a physical activity measurement platform," presented at the Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems, Atlanta, Georgia, USA, 2010.

[17] S. R. Das*, et al.*, "Home automation and security for mobile devices," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, 2011, pp. 141-146.

[18] V. M. Salerno and G. Scata, "An iOS Application to Manage Home Automation."

[19] D. Bell. (2011, 24/10/2011). *Best portable video players (PVP)*. Available: http://reviews.cnet.com/best-pvps/

[20] M. Spoonauer. (2010, 24/10/2011). *Apple iPod touch (2010) Review*. Available: http://www.laptopmag.com/review/mp3/apple-ipod-touch-2010.aspx

[21] (2010 - 2011, 24/10/2011). *Apple iPod touch 32GB (4th Generation) - Customer Reviews*. Available: http://www.amazon.com/Apple-iPod-touch-32GB-Generation/product-reviews/B001FA1O18/ref=dp_top_cm_cr_acr_txt?ie=UTF8&showViewpoints=1

[22] (2011, 24/10/2011). *Wii controllers*. Available: http://www.nintendo.com/wii/console/controllers

[23] P. Neto*, et al.*, "Attracting Students to Engineering: Using Intuitive HRIs for Educational Purposes Trends in Intelligent Robotics." vol. 103, P. Vadakkepat*, et al.*, Eds.: Springer Berlin Heidelberg, 2010, pp. 250-257.

[24] A. Başçetinçelik, "WiiRobot: controlling robots with Wii gestures," 2009.

[25] C. W. L. K. Chua*, et al.*, "Interactive methods of tele-operating a single unmanned ground vehicle on a small screen interface," presented at the Proceedings of the 6th international conference on Human-robot interaction, Lausanne, Switzerland, 2011.

[26] W. Xing-Han*, et al.*, "A hand-gesture-based control interface for a car-robot," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 4644-4648.

[27] N. Kubota*, et al.*, "Multifeatured visualization and navigation in tele-operation of mobile robots," in *Robotic Intelligence In Informationally Structured Space (RiiSS), 2011 IEEE Workshop on*, 2011, pp. 85-92.

[28] A. Birk*, et al.*, "The IUB Rugbot: an intelligent, rugged mobile robot for search and rescue operations," in *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*: IEEE Press, 2006.

[29] E. Altug*, et al.*, "Control of a quadrotor helicopter using visual feedback," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 72-77 vol.1.

[30] M. Quigley*, et al.*, "Semi-autonomous human-UAV interfaces for fixed-wing mini-UAVs," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, pp. 2457-2462 vol.3.

[31] M. Becker*, et al.*, "GripSee: A Gesture-Controlled Robot for Object Perception and Manipulation," *Autonomous Robots,* vol. 6, pp. 203-221, 1999.

[32] P. X. Liu*, et al.*, "Remote control of a robotic boat via the Internet," in *Information Acquisition, 2005 IEEE International Conference on*, 2005, p. 6 pp.

[33] W. Neubauer, "A spider-like robot that climbs vertically in ducts or pipes," *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on,* vol. 2, pp. 1178-1185 vol.2, 12-16 Sep 1994 1994.

[34] P. Bentley and D. Corne, *Creative evolutionary systems*. San Francisco, CA.; San Diego, CA: Morgan Kaufmann ; Academic Press, 2002.

[35] M. Wang and T. Ganjineh, "Remote controlling an autonomous car with an iPhone," 2009.

[36] E. Guizzo. (2011, 25/10/2011). *How Google's Self-Driving Car Works*. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[37] D. Tse and P. Viswanath. (2005). *Fundamentals of wireless communication*. Available: http://www.myilibrary.com?id=95190

[38] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *Wireless Communications, IEEE,* vol. 12, pp. 12-26, 2005.

[39] O. Wongwirat*, et al.*, "A prototype development of ET rescue robot by using a UML," in *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on*, 2009, pp. 1-5.

[40] N. Baker, "ZigBee and Bluetooth strengths and weaknesses for industrial applications," *Computing & Control Engineering Journal,* vol. 16, pp. 20-25, 2005.

[41] L. Kwang Yeol and C. Jae Weon, "Remote-controlled home automation system via Bluetooth home network," in *SICE 2003 Annual Conference*, 2003, pp. 2824-2829 Vol.3.

[42] P. Baronti*, et al.*, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications,* vol. 30, pp. 1655-1695, 2007.

[43] P. Kinney, "ZigBee Technology: Wireless control that simply works," presented at the Communication Design Conference, 2003.

[44] R. Ding*, et al.*, "Design and Realization of Omni-directional Mobile Robot Body Based on Zigbee Technology," in *Embedded Computing, 2008. SEC '08. Fifth IEEE International Symposium on*, 2008, pp. 207-211.

[45] C. M. Kozierok, *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*: No Starch Press, 2005.

[46] Y. Amir*, et al.*, "An Overlay Architecture for High-Quality VoIP Streams," *Multimedia, IEEE Transactions on,* vol. 8, pp. 1250-1262, 2006.

[47] A. H. J. Lakkakorpi, and J. Ruutu, "Measurement and Characterization of Internet Gaming Traffic," *Nokia Research Center, NOKIA GROUP, Finland*.

[48] K. Goldberg*, et al.*, "Desktop teleoperation via the World Wide Web," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, 1995, pp. 654-659 vol.1.

[49] W. Wenbin*, et al.*, "Internet-based tele-control system for wheeled mobile robot," in *Mechatronics and Automation, 2005 IEEE International Conference*, 2005, pp. 1151-1156 Vol. 3.

[50] O. Janssen, "Wifi interface for mobile robots," 2008.

[51] R. V. Aroca*, et al.*, "Scara robot controller using real time linux," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, 2007, pp. 1-6.

[52] S. Garg and M. Kappes, "An experimental study of throughput for UDP and VoIP traffic in IEEE 802.11b networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 2003, pp. 1748-1753 vol.3.

[53] G. R. Bradski and A. Kaehler, *Learning OpenCV*. Farnham; Cambridge: O'Reilly, 2008.

[54] Ken. (2008, 24/10/2011). *Sneak Preview: iPhoneCam*. Available: http://macdaddyworld.com/2008/01/12/sneak-preview-iphonecam/

[55] V. Frati and D. Prattichizzo, "Using Kinect for hand tracking and rendering in wearable haptics," in *World Haptics Conference (WHC), 2011 IEEE*, 2011, pp. 317-321.

[56] I. Oikonomidis*, et al.*, "Efficient model-based 3d tracking of hand articulations using kinect," 2011.

[57] R. Murphy*, et al.*, "Potential Tasks and Research Issues for Mobile Robots in RoboCup Rescue

RoboCup 2000: Robot Soccer World Cup IV." vol. 2019, P. Stone*, et al.*, Eds.: Springer Berlin / Heidelberg, 2001, pp. 339-344.

[58] S. M. Ettinger*, et al.*, "Vision-guided flight stability and control for micro air vehicles," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 2002, pp. 2134-2140 vol.3.

[59] T. D. Cornall*, et al.*, "Aircraft attitude estimation from horizon video," *Electronics Letters,* vol. 42, pp. 744-745, 2006.

[60] (2011, 24/10/2011). *AR.Drone - Technologies*. Available: http://ardrone.parrot.com/parrot-ar-drone/en/technologies

[61] CES. (2010, 03/10/2011). *2010 Innovations Honorees*. Available: http://www.cesweb.org/awards/innovations/2010honorees.asp

[62] (2011). *Apptoyz - Products*. Available: http://www.apptoyz.com/products/

[63] (2011, 24/10/2011). *HELO TC: iOS Controlled RC*. Available: http://www.griffintechnology.com/helotc

[64] (2011, 24/10/2011). *iHelicopters*. Available: http://www.ihelicopters.net/shop/wall-climbing-car/iphone-controlled-wall-climbing-car-iw500/)

[65] (2011, 24/10/2011). *Code::Blocks*. Available: http://www.codeblocks.org

[66] A. Inc., "iOS human interface guidelines," 2011.

[67] W. Long and J. Ackermann, "Robustly stabilizing PID controllers for car steering systems," in *American Control Conference, 1998. Proceedings of the 1998*, 1998, pp. 41-42 vol.1.

[68] A. Datta*, et al.*, *Structure and synthesis of PID controllers*: Springer, 2000.

[69] L. G. Fagundes and R. Santos, "Development of Computer Graphics and Digital Image Processing Applications on the iPhone," in *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2010 23rd SIBGRAPI Conference on*, 2010, pp. 34-45.

[70] G. Mester, "Wireless sensor-based control of mobile robots motion," in *Intelligent Systems and Informatics, 2009. SISY '09. 7th International Symposium on*, 2009, pp. 81-84.

Annexes

METHOD SUMMARY

	Method	Description	Input	Output
Control Type view controller	slidersPressed	Method called when Sliders button is pressed - pushes Connection view controller	(id) sender	IBAction
	accelerometerPressed	Method called when Accelerometer button is pressed – pushes Calibration view controller	(id) sender	IBAction
Calibration view controller	accelSetup	Sets up the accelerometer with 1/60 update frequency and starts accelerometer data acquisition	void	void
	calibrationPressed	Method called when Calibrate button is pressed – calibrates the X-axis	(id) sender	IBAction
	connectionPressed	Method called when Connection button is pressed – pushes Connection view controller	(id) sender	IBAction
Connection view controller	textFieldDoneEditing	Method called when Done button is pressed – makes the keyboard hide	(id) sender	IBAction
	backgroundTap	Method called when anything but the keyboard is pressed – makes the keyboard hide	(id) sender	IBAction
	connectPressed	Method called when Connect button is pressed – pushes Controller view controller	(id) sender	IBAction
	alertView: clickedButtonAtIndex	Alert appears with an informative message about the connection status	(UIAlertView *) alertView (NSInteger) buttonIndex	void
Controller view controller	connectToServerUsingCFStream: portNo	Method automatically called – establishes the connection to the selected address (urlStr) and port number (portNo)	(NSString *) urlStr (uint) portNo	void
	disconnect	Method called when Connect button is pressed – pushes back Connection view controller	void	void
	writeToServer	Method automatically called – writes the data (buf) on the network output stream	(const uint8_t *) buf	NSInteger
	stream:handleEvent	The delegate receives this message when a given event has occurred on a given stream	(NSStream *) theStream (NSStreamEvent) streamEvent;	void
	raw2Img	Method automatically called – takes the image data (rawData) and creates a UIImage with the predefined resolution	(unsigned char *) rawData	UIImage
	accelSetup	Sets up the accelerometer with 1/60 update frequency and starts accelerometer data acquisition	void	void
	refreshData	Refreshes the accelerometer values	void	void