

WIRELESS CONTROL SYSTEM FOR AN INDUSTRIAL ROBOT

Bob Emmanuel Onyenike
Bachelor's thesis
Spring 2011
Degree Program in Information Technology
Oulu University of Applied Sciences

PREFACE

This Bachelor's Thesis is implemented as an extension to a wireless robot control system in the Computer Laboratory of Raahe School of Engineering and Business, Oulu University of Applied Sciences during spring 2011.

ACKNOWLEDGEMENTS

I would like to thank the management of the Computer Laboratory of Raahe School of Engineering and Business, Oulu University of Applied Sciences for the opportunity given to me to implement my bachelor's thesis.

Special thanks to my thesis supervisor Leo Ilkko for his supervising role, patience, advice, for giving me this interesting topic for my thesis and also for introducing me to the world of telerobotics.

Thanks to all fellow students who started with me at the beginning of my study year in September 2007, I appreciate their support, I also like to thank the students in other degree groups who are also great friends.

Special thanks to all the teachers of this great school, for their tutoring role and advices; Lea Hannila, Leo Ilkko, Risto Korva, Timo Vainio, Juha Rätty, Lauri Pirttiaho, Tiina Ovaska, the Rector of the school, all other teachers and members of staffs whose names are not mentioned you are always in my heart.

Finally I would like to thank my lovely family for their support, patient and understanding. I am grateful.

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu,
Tietotekniikan koulutusohjelma, Mobiiliteknologia

Tekijä: Bob Emmanuel Onyenike.

Opinnäytetyön nimi: Wireless Control System for Industrial Robot.

Työn ohjaaja: Leo Ilkko

Työn valmistumislukukausi ja -vuosi : Kevät 2011

Sivumäärä: 39 + liitteet 10

Tämä opinnäytetyö on langattoman ohjausjärjestelmän toteutus Mitsubishi MOVEMASTER RV-E3J-teollisuusrobotille, jossa on SONY EVI-D30-kamera antamassa reaaliaikaista videokuvaa robotin käyttäjälle. Työ tarkoitus on myös perehdyttää teknologioihin joita vaaditaan järjestelmän toteuttamiseen. Niitä teknologioita ovat tietoliikenne, multimedia, verkot, telerobotiikka ja ohjelmointi.

Tässä työssä käytettyjä ohjelmistotyökaluja ovat Qt-ohjelmointiympäristö Windows- ja Symbian-käyttöjärjestelmille. Toinen tässä opinnäytetyössä kehitetty sovellus toimii palvelimena Windows-käyttöjärjestelmässä ja toinen sovellus Symbian-käyttöjärjestelmässä mobiililaitteessa.

Perusajatuksena oli kehittää langaton ohjausjärjestelmä Mitsubishi RV-E3J-teollisuusrobotille ja ohjata robottia langattomasti mobiililaitteella. Mobiililaitesovelluksella voi lähettää komentoja teollisuusrobotille ja kontrolloida sen liikkeitä.

Tämän työn toisena perusajatuksena oli saada visuaalista palautetta robotilta lähettämällä videokuvaa matkapuhelimeen suoratoistona. Robotin käyttäjä voi ohjata matkapuhelimella kameraa ja sen avulla seurata reaaliaikaisesti robotin toimia.

Tärkeintä on se, että robotin liikkeet tapahtuvat turvallisesti. Liikkumisen on tapahduttava määritellyllä alueella eikä robotti saa aiheuttaa vahinkoa ympäristölleen.

Tämä opinnäytetyö on tehty Raahen tekniikan ja talouden kampuksella tietokonelaboratoriossa.

Asiasanat: Robotiikka, tiedonkeruu, viestintä

ABSTRACT

Oulu University of Applied Sciences
Bachelor Degree Program in Information Technology.

Author: Bob Emmanuel Onyenike.

Title of Bachelor's thesis: Wireless control system for industrial robot.

Supervisor: Leo Ilkko

Term and year of completion: Spring 2011

Number of pages: 39 + appendices 10

This thesis work is an implementation of a wireless control system for Industrial robot: a Mitsubishi MOVEMASTER RV-E3J industrial robot which has a SONY EVI-D30 camera device attached nearby to provide a real-time video feedback to the operator. The purpose is also to get acquainted with the related technologies needed for the implementation of this thesis work. The fields covered are in telecommunications, multimedia, networking, telerobotic and programming.

The software development tool used in this thesis work is the Qt Framework development tools for Windows and the Symbian platform. One application developed in this thesis work is acting as the desktop application which runs on the Windows operating system, while another application installed on a smartphone runs on the Symbian operating system.

The main idea of the project was to develop a wireless control system for the Mitsubishi RV-E3J industrial robot, and controlling the robot wirelessly from a smartphone. The smartphone's can send commands to the industrial robot to control its movement.

The second main idea of this thesis was to get a visual feedback from the standby camera to the smartphone through video streaming, the operator from the mobile phone can control the movement of the camera and at the same time get visual feedback from the camera which is positioned next to the robot and it transmits a real-time video output showing the robot's actions.

Most importantly it is very necessary for the robot's movement to be within a specific area and the working environment must be safe from an accidental damage caused by the robot.

This Bachelor's thesis work was done in the Computer Laboratory of Raahe School of Engineering and Business.

Keywords: [Robotics](#), [Data Collection](#), [Communication](#)

TABLE OF CONTENTS

PREFACE.....	1
ACKNOWLEDGEMENTS.....	1
TIIVISTELMÄ.....	2
ABSTRACT.....	3
TABLE OF CONTENT	4
SYMBOLS AND ABBREVIATION.....	6
1 INTRODUCTION.....	7
1.1 Tele-operation.....	7
1.2 Telepresence.....	7
1.3 Application of telerobotics.....	7
1.4 Major components of telerobotics.....	10
2 THE WORK ENVIRONMENT.....	11
2.1 Qt Framework.....	11
2.2 Qt version 4.7.....	11
2.3 Integrated Development Environment (IDE)	12
2.4 Smartphone dependencies.....	13
3 DEFINITION.....	14
3.1 Overview of the system.....	14
3.2 Interface between robot and operator.....	15
3.2.1 Workstation Interface.....	15
3.2.2 Mobile Interface.....	16
3.2.3 Difference between the interfaces.....	17
3.3 Video Control.....	18
4 IMPLEMENTATION.....	20
4.1 Overview of the implementation.....	20.
4.2 Workstation's application (WCSIRServer).....	21
4.3 Smartphone's application(WCSIRClient).....	24
4.4 Real-time video streaming.....	29

5 TESTING.....	30
5.1 Workstation to Robot and Camera connection testing.	30
5.2 Smartphone to workstation connection testing.	31
5.3 Testing the Video streaming.	33
6 POSSIBILITY OF FURTHER DEVELOPMENT.....	35
7 CONCLUSION.....	36
7.1 Design and development.....	36
7.2 Safety measure.....	37
8 LIST OF REFERENCES.....	38

SYMBOLS AND ABBREVIATIONS

AP	Access Point
API	Application Programming Interface
AT	Auto Tracing
COM	Serial Port
Desktop	Personal computer
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GSM	Groupe Spéciale Mobile, Global System for Mobile communications
HMH	Hazardous Material Handling
IT	Information Technology
PC	Personal Computer
RTSP	Real-time Streaming Protocol
RTP	Real-time Transport Protocol
SDK	Software Development Kit
TCP/IP	Transmission Control Protocol/ Internet Protocol
WCSIR	Wireless Control System for Industrial Robot
WCSIRClient	Thesis Application that runs on the Smartphone.
WCSIRServer	Thesis Application running on the Workstation connected to a Robot and Camera
Wi-Fi	Wireless Fidelity
Workstation	Computer

1 INTRODUCTION

Tele-robotics is the area of robotics concerned with the control of robots from a distance; mainly using wireless connections like Wi-Fi, Bluetooth or the internet. It is a combination of two major subfields, telepresence and teleoperation.

1.1 Tele-operation

Tele-operation is a process of getting work done from a distance. The work done can be anything while the distance can be a physical distance or just a change in scales when for example a surgeon is using a micro-manipulator technology to conduct surgery on a microscopic level.

Teleoperator is a device that is controlled remotely by a human operator. If such a device has the ability to perform autonomous work, it is called a telerobot. If the device operates completely alone, it is called a robot. The operators command has to correspond with the actions of the robot. In some operations where the distance can affect the wireless controlling of the robot, the robot is made to follow a specified path. For example, in a radio controlled aircraft at some points the device may need to take some independent decisions, such as obstacle avoidance (Wikipedia. 2011, Date of retrieval 31.03.2011).

1.2 Telepresence

Telepresence refers to a set of technologies which allows a person to feel as if they were present, to give the appearance that they were present, or to have an effect, through telerobotics at a place other than their true location.

Telepresence requires that the users' senses are provided with such stimuli as to give the feeling of being in that other location. Additionally, the users may be given the ability to affect the remote location. In this case, the user's position, movements, actions, voice, etc. may be sensed, transmitted and duplicated in the remote location to bring about this effect. Therefore, information may be travelling to both directions between the user and the remote location.

1.3 Application of telerobotics

Videoconferencing - A popular application is found in videoconferencing, a higher level of video telephony which deploys greater technical sophistication and improved fidelity of both video and audio than in traditional videoconferencing. Telepresence can be used to establish a sense of shared

presence or shared space among geographically separated members of a group.

Hazardous environments – In situations where humans are exposed to hazardous situations telerobotics is readily recognised as a suitable replacement. Mining, bomb disposal, rescue of victims from fire, toxic atmospheres, or even hostage situations, are some examples.

Space exploration – With the exception of the Apollo program most space exploration has been conducted with telerobotic space probes. Most space-based astronomy has been conducted with a telerobotic telescope. Recent noteworthy examples include the Mars Exploration Rovers (MER) and the Hubble Space telescope(Wikipedia. 2011, Date of retrieval 31.03.2011).

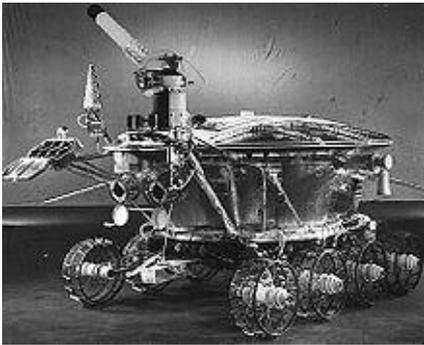


FIGURE 1: *The Soviet telerobotic vehicle Lunokhod-1 used to land on the moon between 1969 and 1977.*

Remote surgery - The possibility of being able to project the knowledge and the physical skill of a surgeon over long distances has many attractions. Thus, again there is a considerable research underway in the subject. (Locally controlled robots are currently being used for a joint replacement surgery as they are more precise in milling a bone to receive the joints.)

Military operation - The armed forces have an obvious interest in telerobotics since the application of robots in battlefield provides more efficient and life-saving combats.



FIGURE 2: *Justus security robot patrolling in Krakow.*

Other applications of telerobotics include Marine Remotely Operated vehicles (ROVs), which are widely used to work in water too deep or too dangerous for divers. They repair offshore oil platforms and attach cables to sunken ships to hoist them. They are usually attached by a tether to a control centre on a surface ship. The wreck of the Titanic was explored by an ROV, as well as by a crew-operated vessel (Wikipedia. 2011, Date of retrieval 31.03.2011).

Also, in Pipeline inspection small diameter pipes otherwise inaccessible for an examination can be viewed by using a pipeline video inspection.

Telerobotics is used also in remote manipulators which are used to handle radioactive materials.

Telerobotics is also used in education industries, entertainment, security and emergency management too.

Additionally, a lot of telerobotic research is being done in the field of medical devices, and minimally invasive surgical systems. With a robotic surgery system, a surgeon can work inside the body through tiny holes just big enough for the manipulator, with no need to open up the chest cavity to allow hands inside.

1.4 Major components of telerobotics

Two major components of Telerobotics are its visual and control applications. A remote camera provides a remote view from the robot. Placing the robotic camera in a perspective that allows intuitive control is a recent technique that although based in Science Fiction (Heinlein.1942) has not been fruitful as the speed, resolution and bandwidth has only recently been adequate to the task of being able to control the robot camera in a meaningful way. Using a head mounted display the control of the camera can be facilitated by moving or tilting the camera.

With this positioning the user can face some problems in the lagging response in movement of the robot and the visual representation because of the limitation of wireless controls.

This bachelor's thesis is about a telebot system of an industrial robot: a Mitsubishi Movemaster RV-E3J Industrial robot that is placed in a Laboratory room and is controlled from a Smartphone in this case a Nokia N97 phone.

2 THE WORK ENVIRONMENT

2.1 Qt Framework.

The two main applications in this thesis project are desktop and smartphone applications. They are implemented by using the Qt Framework application development tools, which are widely used for developing GUI and non-GUI application software. There are several applications and companies that are using this toolkit, e.g. Skype, Google Earth, Samsung, Nokia, Philips, Panasonic, Walt Disney animation Studio.

Qt is a cross platform application and a UI framework. It is produced by Nokia's Qt Development Framework division, which started after Nokia acquired a Norwegian company Trolltech, the original producer of Qt.

It includes a cross-platform class library, integrated development tools and a cross-platform IDE. Using Qt, you can write applications once and deploy them across many desktops, smartphones and embedded operating systems without rewriting the source code.

Qt uses standard C++ but it also makes use of a special code generator called Meta Object Compiler together with several macros to enrich the language. Qt can also be used together with other programming languages through a language binding. It runs on all major platforms and has extensive internationalization support(Wikipedia. 2011, Date of retrieval 31.03.2011). Here is a sample Qt Hello world application

```
#include <QtGui>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QLabel label("Hello, world!");
    label.show();
    return app.exec();
}
```

2.2 Qt version 4.7

During the development of this project I used the open source edition of the Qt Framework development tool. For a wireless control system project, the client application was developed to run on a smartphone, in this case the Nokia N97. While the desktop application was developed to run on the Windows platform,

the Qt versions I used are the Qt 4.7.1 version for Symbian and the Qt 4.7.1 version for Windows.

At the initial phase of the development Qt was working on an additional set of APIs that covers new features. This new API project is called the Qt Mobility API project. One of the features of this thesis project is the possibility of getting camera frames from the SONY EVI-3D camera into the desktop application running on the workstation. I downloaded and installed the Qt Mobility 1.1 version.

2.3 Integrated Development Environment(IDE)

The standard Qt's Integrated Development Environment called Qt Creator was used in this thesis project in developing the Smartphone and workstation applications. It includes a visual debugger and an integrated GUI layout and forms designer.

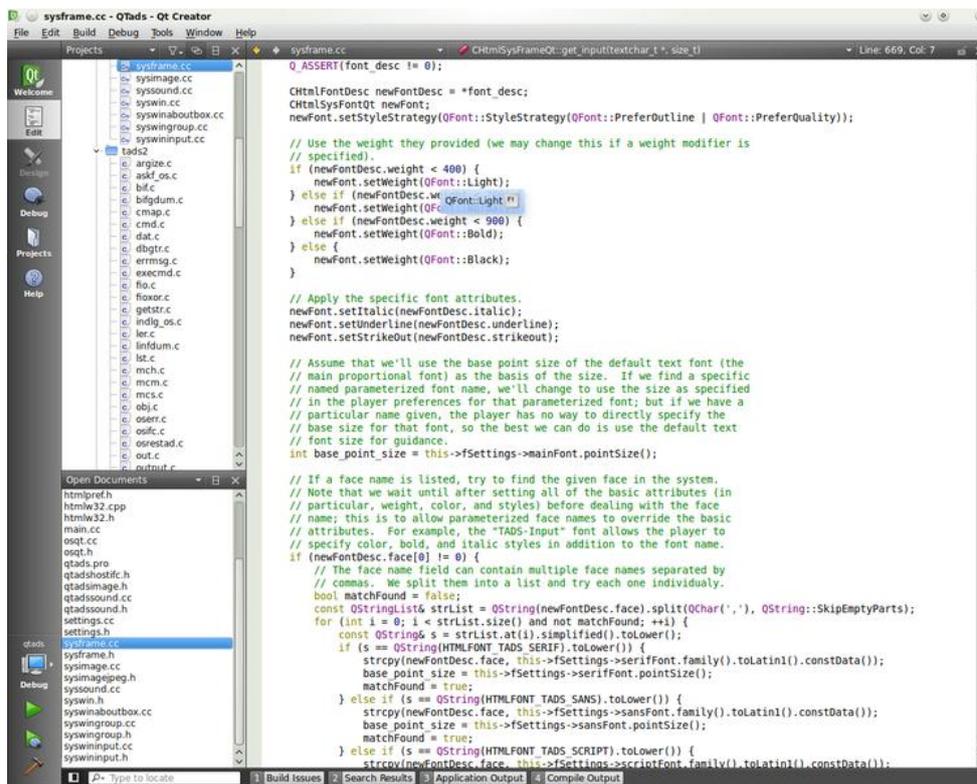


FIGURE 3: Qt Creator.

1.3 Smartphone dependencies.

On the smartphone there are some dependencies that need to be in place before a Qt application can be deployed on it. Applications are deployed to Symbian devices in .sis package files. Although some Symbian devices may already have a version of Qt installed on them, there need to be a way to ensure that an appropriate version of Qt is available for the application to use. Rather than deploy the required Qt libraries with the application, the preferred way to package Qt applications for deployment is to download and use the Smart Installer.

The Smart Installer makes sure that deployed applications have all the Qt dependencies they need to run on a device, performing the necessary updates when the user installs the application (Nokia 2011, Date of retrieval 31.03.2011).

3 DEFINITION

3.1 An overview of the system

As mentioned before in many cases the operator of a robot needs to see its actions, analyse them on the fly and then perform further control with possible corrections. Depending on the situation and environment the operator might not have a possibility to carry with him an original robot control unit (Teaching Box in case of the Mitsubishi Movemaster RV-E3J Industrial robot) and a visual output device such as a monitor of a laptop. Because of these constraints, a system that allows controlling the Movemaster robot through a Wireless channel from a smartphone and getting the visual display of the robot's movement was designed. In this case the operator will get the possibility to receive a visual feedback while using the interface on the workstation which is directly connected to a robot or through a remote control on the smartphone.

One of the necessary conditions for the successful system operation is the presence of an active wireless connection from the smartphone to TCP/IP connections. The wireless control system of the Mitsubishi Movemaster industrial robot works as follows: the operator using his/her smartphone establishes a TCP/IP Socket connection with the desktop application running on the workstation, which is directly connected to the Movemaster robot and the Sony EVI-D30 camera. After the connection has been established the operator is able to see on the display of his/her smartphone a video picture showing the Movemaster's movement that comes from a camera located in the working environment of the robot. Also the operator can control the Mitsubishi Movemaster's movement by using the controls of his/her smartphone. A diagram representing the WCSIR system operation is given on the Figure shown below.

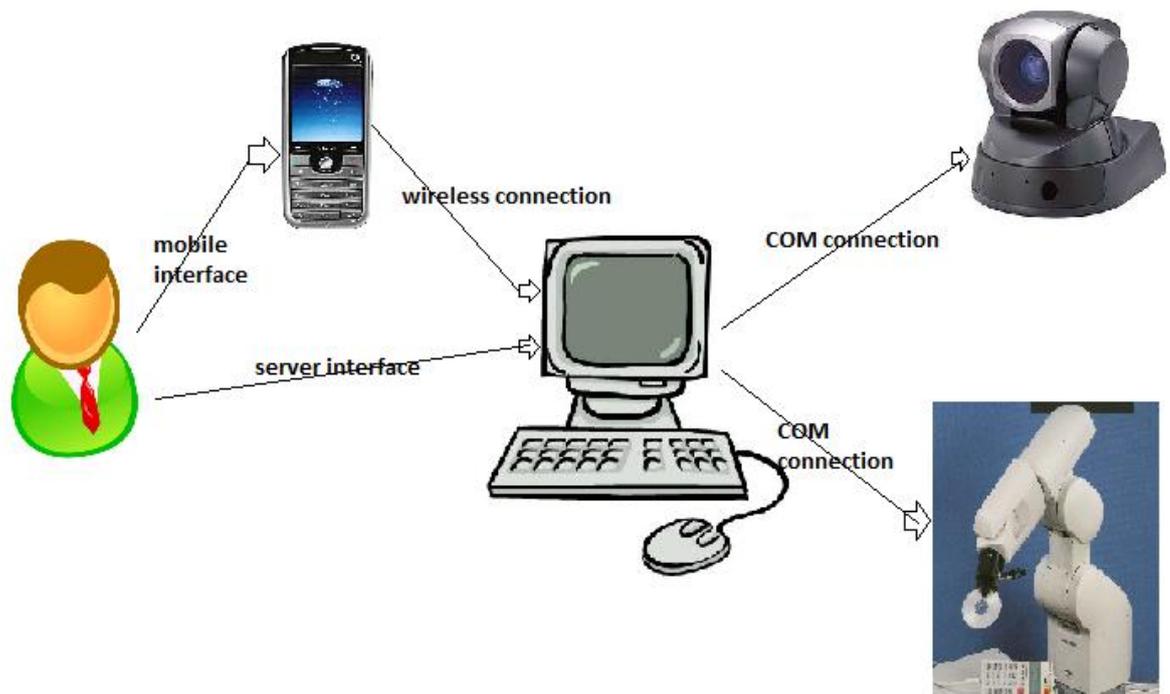


FIGURE 4: Diagram of the WCSIR system.

The operation space of the robot must lie within safety limits in order to eliminate any possible danger for human beings as well as for the working environment. Thus, the operator should not be able to make an accidental damage by misusing controlling possibilities offered by WCSIR. However, this requires special precautions at the system design stage in accordance with the robot's specifications.

3.2 Interfaces between robot and operator

3.2.1 Workstation Interface

There are two interfaces between the operator and the robot existing in the WCSIR. The first one is a desktop interface; it gives the operator the possibility to control the robot from the workstation which is directly connected to the robot.

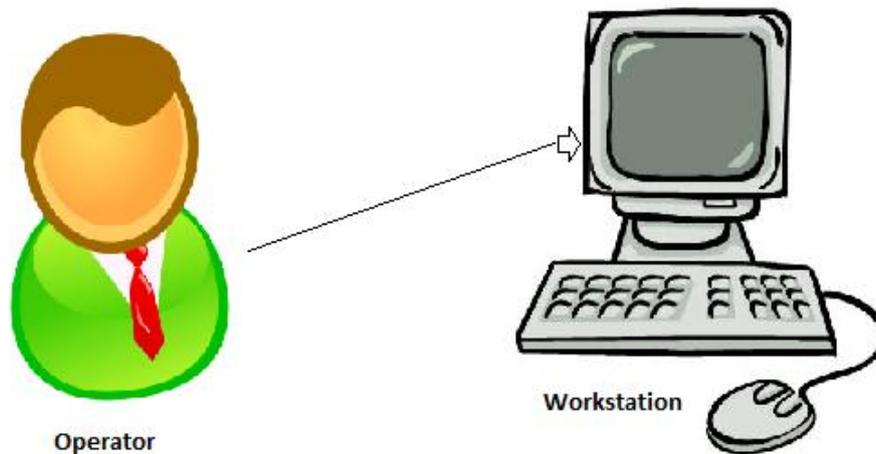


FIGURE 5: *Workstation interface*

3.2.2 Mobile Interface

The second interface is the mobile interface. This one gives the operator the possibility to interact with the robot and the camera by means of a smartphone using wireless connections. Interacting with the robot and the camera through its interface is very flexible since in order to control the robot the operator can choose any location within the mobile network accessibility. However, this interface has some limits.

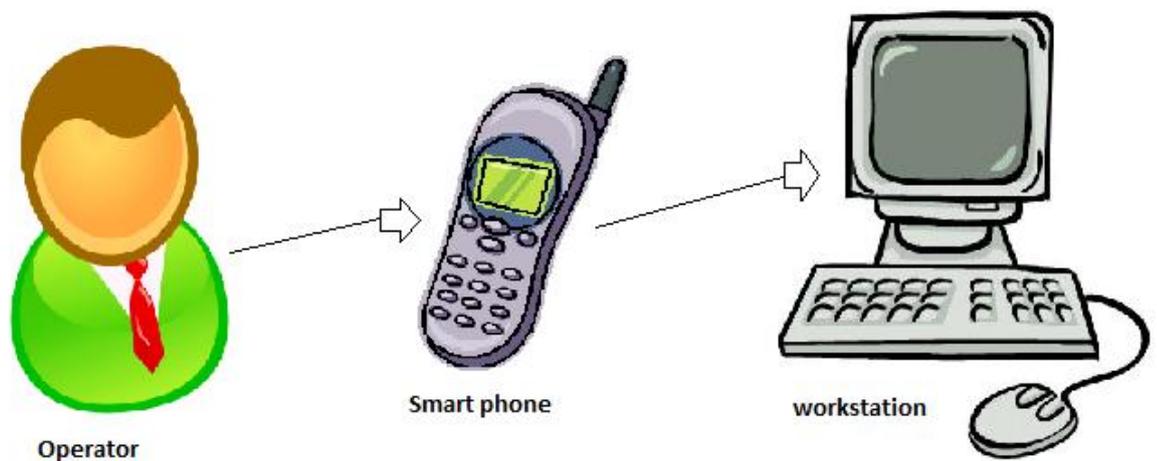


FIGURE 6: *Mobile phone interface*

3.2.3 Difference between the interfaces

The main limitation of the smartphone's interface is its small bandwidth, restricted CPU and small memory resources. The visibility of the robot offered by the smartphone interface is restricted by the screen size of the smartphone. The comparison between both interfaces is presented in the table below (Table 1).

	Workstation interface	Smartphone interface
Size	-	+
accessibility	-	+
Bandwidth	+	-
CPU power	+	-
Memory	+	-
Mobility	-	+
Robot visibility	+	-
Picture quality	+	-

TABLE 1: Workstation interface vs. Mobile interface.

From the table we can see that each interface has its own advantages and disadvantages. For example, the workstation interface leads in the area of resources but losses in mobility concerns. The operator should choose the proper interface base on the condition.

The extent of control over the robot also depends on the interface. Potentially the workstation interface can offer far better controlling possibilities than the smartphone interface, particularly in the area of feedback. The video signal processing requires a large amount of CPU power and memory availability which cannot be offered by the WCSIR mobile interface. So by using the workstation interface the operator may get a picture with a bigger resolution compared to the mobile interface, which means a more detailed robot image and therefore a better controlling scope.

However, the smartphone interface due to its characteristics cannot be replaced with the workstation's interface in certain cases (see Table 1). The main idea of this WCSIR system implies the use of the smartphone interface as a primary one offering the operator the maximum flexibility with accessing the robot and minimizing at the same time such constrains as location, portability and access

time. Nevertheless, both the workstation and mobile interface have to be coupled together supplementing each other and ensuring by this way the proper functionality of the entire Control System for the Industrial Robot.

3.4 Video control

The Video control plays one of the most crucial roles in telerobotics generally and in WCSIR system as one of its applications. In order to be able to control the robot in more efficient way, the robot operator has to have a visual feedback on the actions of the robot for safety reasons so that the operator can see an upcoming danger and take preventive measures.

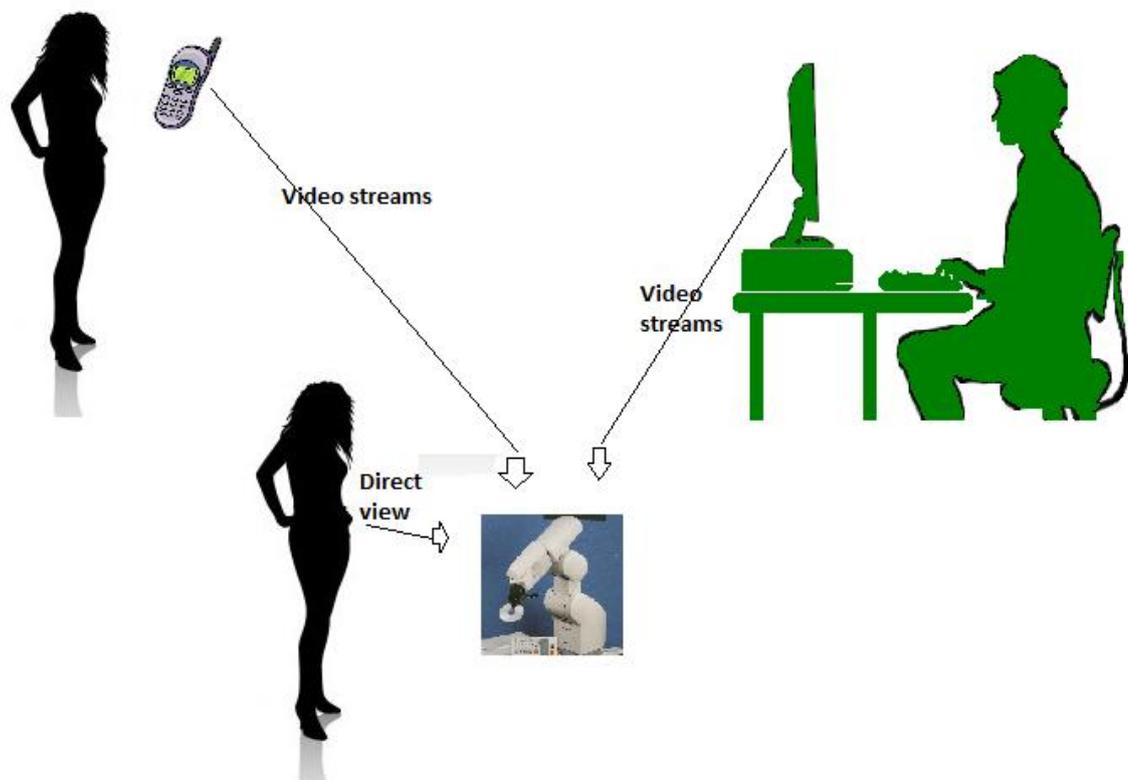


FIGURE 7: Several visual options to the operator.

The video control of the robot is implemented by using a camera, which is located close to the robot and it transfers a real-time video stream to the robot operator. Several features were kept in mind while choosing a suitable camera for this project: a sufficient picture resolution, moving and zooming capabilities along with an auto focusing feature and suitable video signal outlets. A video

camera Sony EVI-D31(Figure 8) was chosen for the WCSIR system development.



FIGURE 8: SONY EVI-D30

The camera has a 440 kilo pixel colour CCD image sensor and it produces a PAL colour encoding video signal. The pan/tilt range of this camera is 200 degrees in vertical and 50 degrees in horizontal planes respectively and the camera also has a 12x zoom. The camera allows transferring a real-time video stream to the display of operator's smartphone and gives the operator a visual feedback on the robot operations. The operator can follow the robot arm movement by enabling the Auto Tracking Mode, which has two options.

1. AT-PAN/TILT – It allows the following of the moving subject automatically by controlling the pan& tilt motors.
2. AUTO ZOOM- It automatically controls the zoom lens to ensure that the size of the subject remains constant (Sony 2011, Date of retrieval 31.03.2011).

Although Sony EVI-D31 meets the entire requirement in the existing system, any other camera can be used instead of depending on the working environment of the robot and other conditions. It only has to provide sufficient controlling capabilities and produce an ample image quality to make the observation of the robot suitable for the operator.

In the WCSIR system the control over camera movement such as pan/tilt can be performed by using the menu list's actions programmed in the smartphone's application. These command buttons options are also in the desktop application.

4. IMPLEMENTATION

4.1. Overview of the implementation

The two main applications implemented between the operator and the robot existing in this system are the desktop application named “WCSIRServer” and the smartphone application named “WCSIRClient”. The WCSIRServer gives the operator the possibility to control the robot from the workstation which is directly connected to the robot and the camera. The “WCSIRClient” gives the operator a possibility to interact with the robot and the camera by means of a Smartphone by using a TCP/IP socket connection.

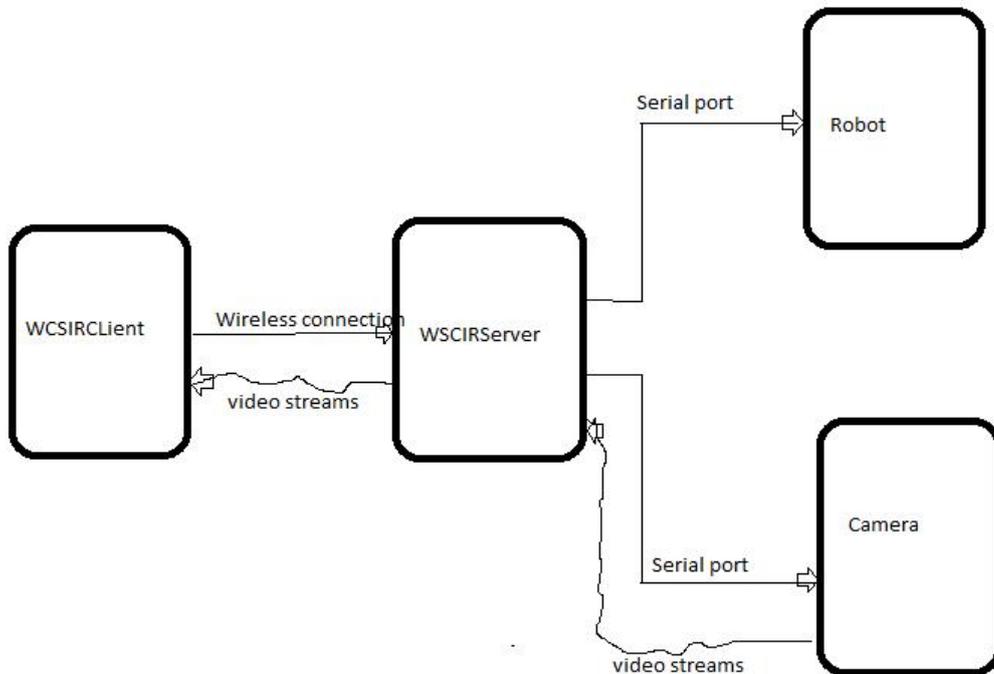


FIGURE 9: *Wireless Control System for an Industrial Robot.*

The Figure below shows how the operator communicates with the robot and the camera from the smartphone interface. The operator uses the smartphone’s WCSIRClient application to send commands to the workstation which are then re-transmitted from the WCSIRServer application through the opened serial COM ports to the robot and the camera. The Figure below(Figure 10) shows how the communication between the user and the robot and the camera are sent.

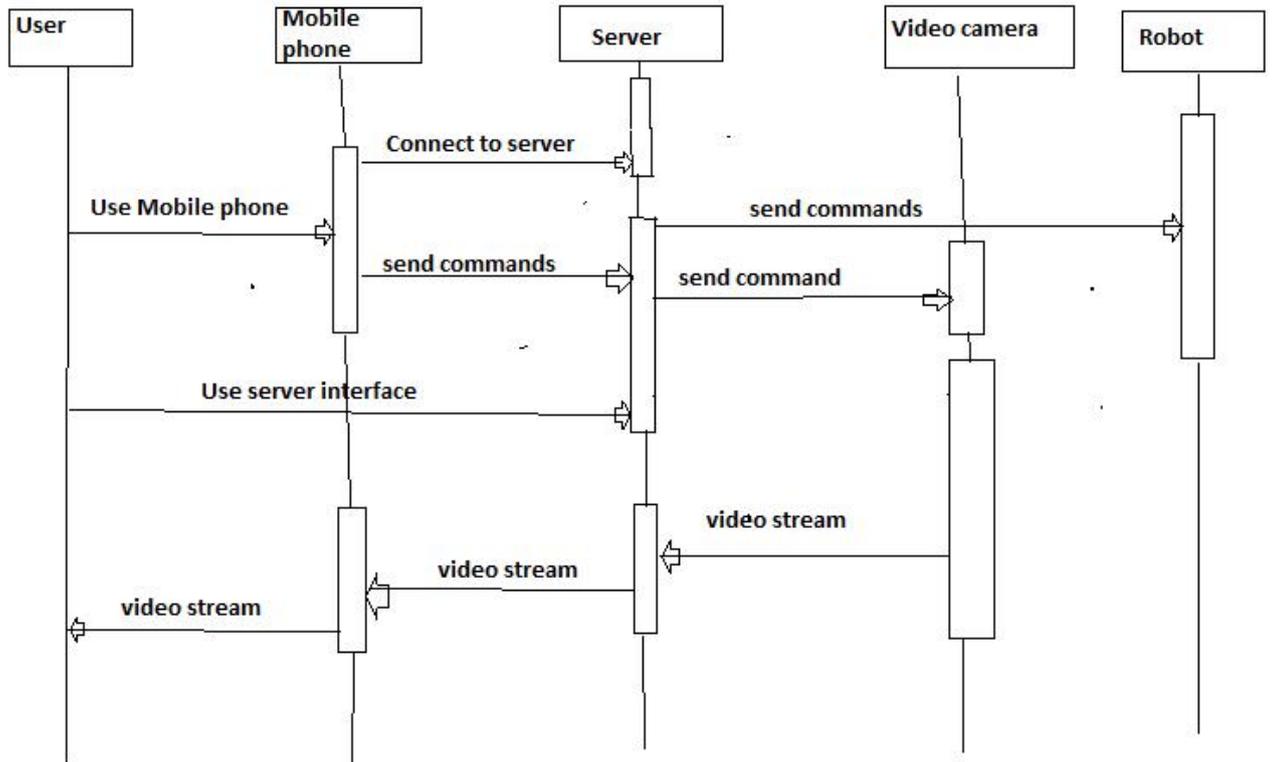


FIGURE 10: Sequence diagram of the WCSIR.

Both the Workstation and the Smartphone's application of the system were developed by using Nokia's Qt Framework development tools for the Windows and the Symbian platform. The powerful Qt SDK provides great tools for developers who are targeting at several platforms. The subsequent development or alteration of the WCSIR permits easy building, debugging and deploying of the applications not only for the workstation but also for smartphones powered by the Symbian platform.

4.2 Workstation's application(WCSIRServer)

There are two hardware devices that communicate with the WCSIRServer application; the video camera Sony EVI-D31 and the Mitsubishi MOVEMASTER RV-E3J industrial robot which are connected to the workstation through serial computer ports. They are used as command channels for both the camera and the robot. The WCSIRServer application runs and opens a COM socket connection to the robot and the camera.

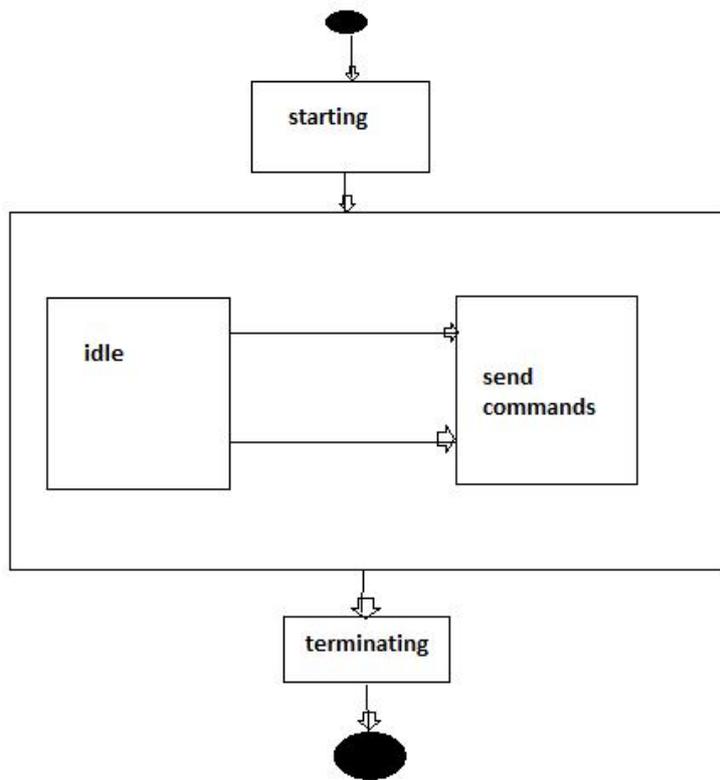


FIGURE 12: State diagram of the WCSIRServer application.

Through the command channel of the robot, the Mitsubishi Movemaster receives commands and the definition of the positions from the WCSIRServer application to perform the required actions.

The commands the operator can give to the robot through the user interface are restricted and designed according to safety specifications. The camera command channel serves the abilities of camera manipulation such as the pan/tilt, movement and zooming.

The functionality of this part of the WCSIR system implemented as the WCSIRServer application is written in a Qt C++ programming language. The sequence diagram below(Figure 11) shows the communications from the WCSIRServer application running on the workstation to the robot and camera.

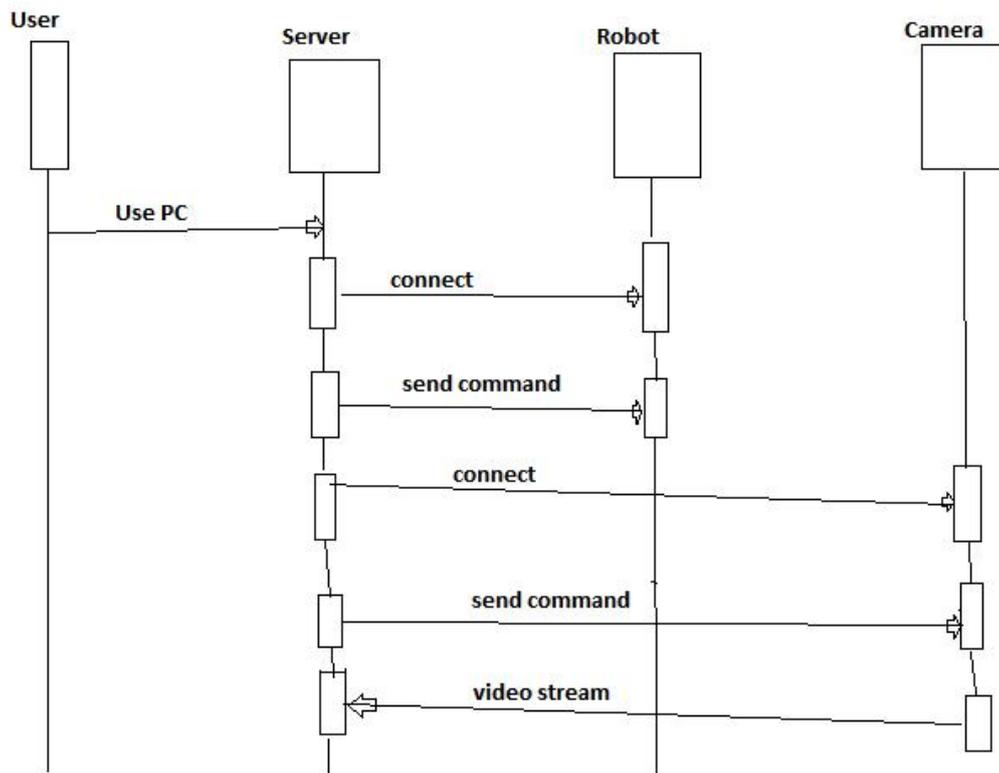


FIGURE 11: Sequence diagram of the workstation interface.

After the application is launched on the workstation, it tries to establish a socket connection to the serial COM ports associated with the robot and the camera device, and it also listens to a dedicated port (13001) for receiving a socket connection from the smartphone. The operator can start sending a command to the robot and the video camera through serial COM ports, Then a serial port socket connection is established.

When the application serial COM connection with the attached camera is opened, it starts searching for video streams from the camera and then displays the video frames on its main view. The user also has the possibility to send movement's commands to the robot and the camera by using the buttons on the view.

The figure below shows the interface's view of the WCSIRServer application.

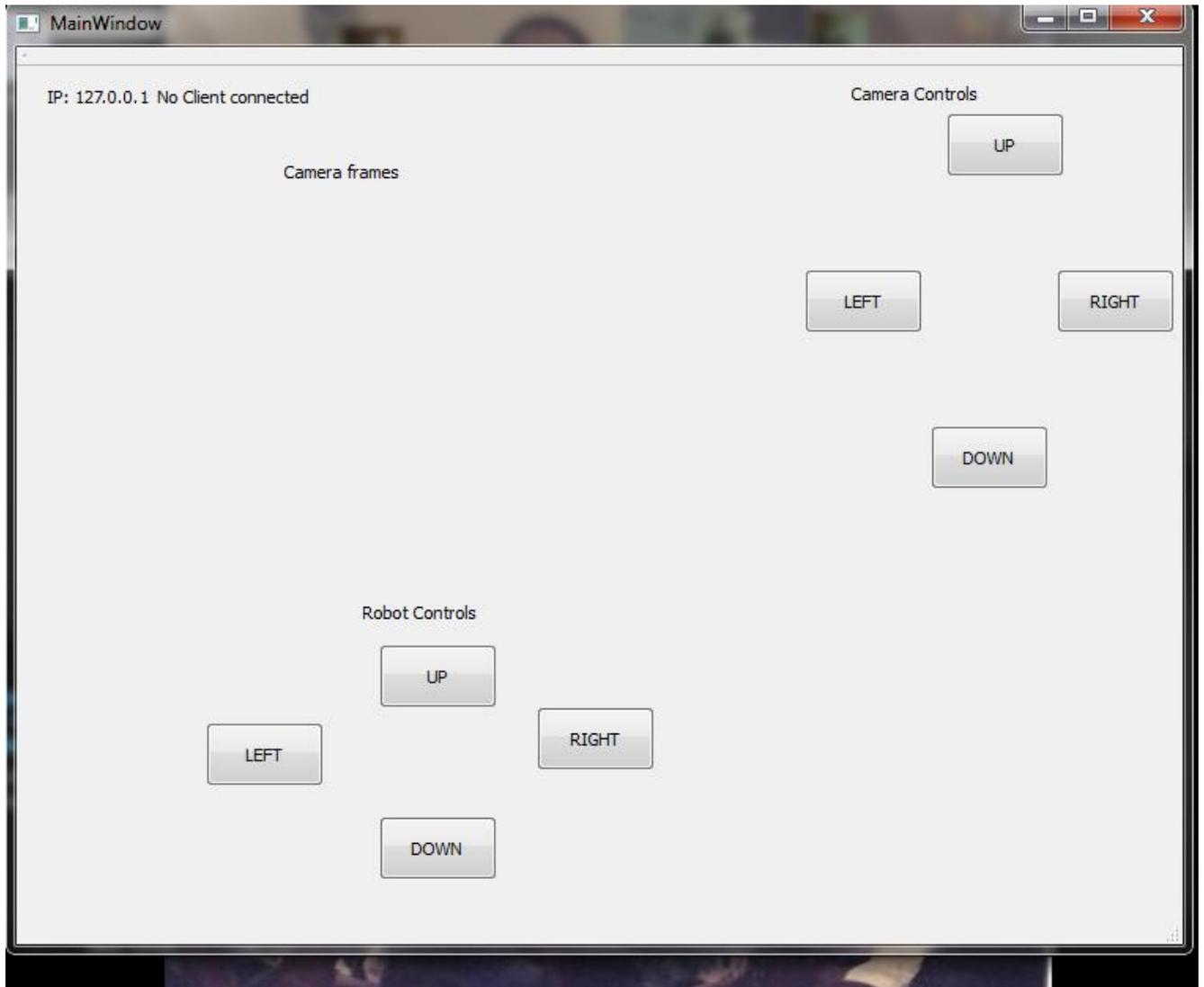


FIGURE 13: *The WCSIRServer Application running on the workstation.*

4.3 Smartphone's application(WCSIRClient)

The smartphone controlling the robot and camera devices was the main idea of this thesis work. It works by having a developed WCSIRClient application running on the mobile phone establishing a TCP/IP socket connection to the WCSIRServer application launched on the workstation through a wireless channel and then sending the required commands to control the movement of both the camera and the robot. On the Symbian powered Nokia phones the menu list items are generally called the left and right softkeys. In the WCSIRClient application these menu items are programmed to provide the option of sending commands from the smartphone to the desktop application, which then re-transmit these commands to the robot and the camera. The WCSIRClient application functions as an extended arm of the desktop's WCSIRServer application, which needs to be launched and connected to the

robot and camera before the mobile phone's WCSIRClient application can become useful.

The sequence diagram below shows the communication sequence between the WCSIRClient application running on the smartphone and the WCSIRServer application running on the workstation.

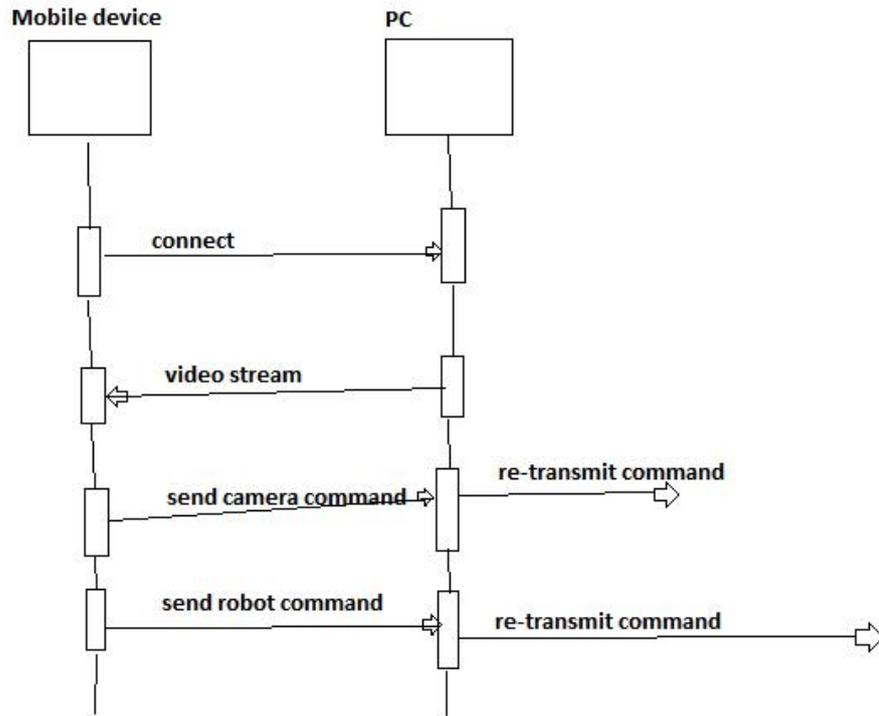


FIGURE 14: Sequence diagram of the mobile phone interface.

Programming the WCSIRClient application was done by using the Qt Framework application development tools. The codes were written in Qt C++ and then compiled for the targeted Symbian platform.

When the WCSIRClient application is launched, the user can see the connection state of the application from the top right-hand position of the application, for example “Not Connected”(Figure 15). The operator also has the menu option “Connect to Server” which allows him to establish a TCP/IP Socket connection to the desktop’s WCSIRServer application running on the workstation,



FIGURE 15: Menu options of the smartphone's application.

Then the user specifies the IP address of the WSCIRServer's workstation and optionally a user name. When he clicks on the "OK" button the WCSIRClient application establishes a TCP/IP socket connection with the WSCIRServer application running on the specified IP address.



FIGURE 16: The connection view of the WCSIRClient application.

When the connection is successful, the operator can see the “connected” message on the top position of the client application.

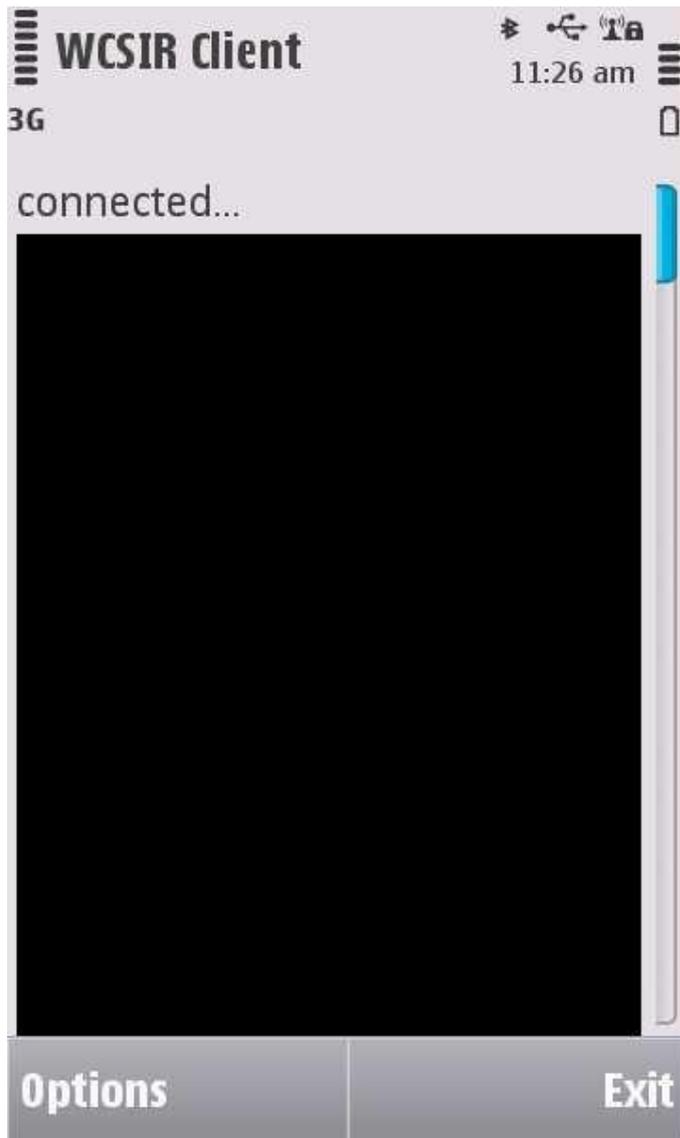


FIGURE 17: Connected state of the WCSIRClient application.

During the design of the smartphone's WCSIRClient application the following considerations were kept in mind. The robot control menu needs to have a simple layout and naming so that it can easily be understood by the operator. In the menu's list of items there are menu items for both the robot control and the camera control and they are uniquely identified so the operator will not be confused. In case of invariable robot operations, the predefined positions and movements were used in order to simplify the operator's input opportunities.

4.4 Real-time video streaming

The real-time visual representation of the robot actions was a key attribute of this thesis work because most of the operator's control decisions are based on visual information originated from the video camera which is located nearby the robot. The camera transfers a real-time analogue signal either on a separated video(S-Video) or on a composite video stream through a S-Video connector to the Hauppauge Win TV PCI bus card with a TV-tuner plugged into the workstation. This allows capturing a real-time output video stream from the camera for further processing. In this WCSIR system the S-Video is used because it has a better quality video rendering appearance since the brightness and colour information that are found together on the composite video do not have to be decoded in the S-Video (Musijenko 2006). From the video-capture card the signal goes to the WRSIRServer application running on the workstation. The WCSIRClient application establishes a TCP/IP socket connection wirelessly to the WRSIRServer application running on the workstation. The video signals are then re-transmitted from the WRSIRServer application to the WCSIRClient application running on the Smartphone.

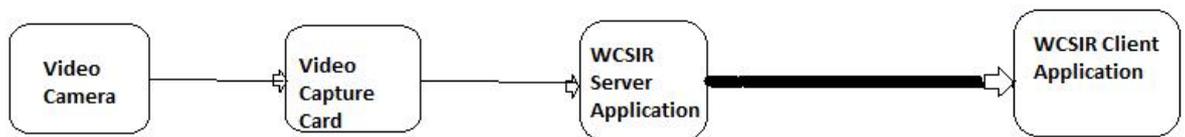


FIGURE 18: Video streaming

The operator has the possibility of tilting the camera's positions from the mobile phone by selecting from the options menu list the required command to send to the camera specifying the direction of the movement, for example move left, move right, move up or move down(Figure 15).

5 TESTING

The Agile testing method was adopted in this thesis project. It is a software testing practice that follows the principle of agile software development. It focuses on an on-going testing against a newly developed code until quality software is produced. The Agile testing is built upon the philosophy that testers need to adapt to rapid deployment cycles and changes in testing patterns. The testing of the different applications and functionalities was a very crucial part in this thesis project and thorough testing was carried out during the implementation phase. During the testing period the following functionalities were tested as they were being developed; a socket connection, a video streaming, sending commands from the desktop application to the camera device, sending commands from the workstation to the robot, sending commands from the Smartphone to the robot and sending commands from the Smartphone to the camera.

Qt creator has a debugging output view where it is possible for developers to get debugging messages. During development of these applications it was used extensively for verifying socket connections, commands sent from the smartphone to the robot and camera and also for checking the video streams coming from the camera.

5.1 A Workstation to a Robot and Camera connection testing

When the WCSIRServer application is launched, it tries to open a serial port connection to the robot and the camera and when the COM port is inaccessible, a pop-up dialog box is displayed notifying the operator of this error connection. The operator should always check the serial port cables connecting the workstation and the robot and the camera beforehand to ensure proper connection.

After a successful COM port connection to the robot and the camera this socket connection remains running until the application is closed. This testing should be done on the safety guideline for operating the robot.

5.2 A Smartphone to a Workstation connection testing

When the application running on the Smartphone is launched, the TCP/IP connection state to the workstation can be seen at the top position of the application. During the launch the application is “not connected”(Figure 15), but when the operator from the menu option connects using the IP address of the workstation running the WCSIRServer application to the workstation, a TCP/IP socket connection request is sent and when it is successful the operator can see the state of the connection. The diagram below(Figure 19) shows a case where the application is launched and not connected and then the operator makes a socket connection to the workstation’s IP address and gets a successful connection.

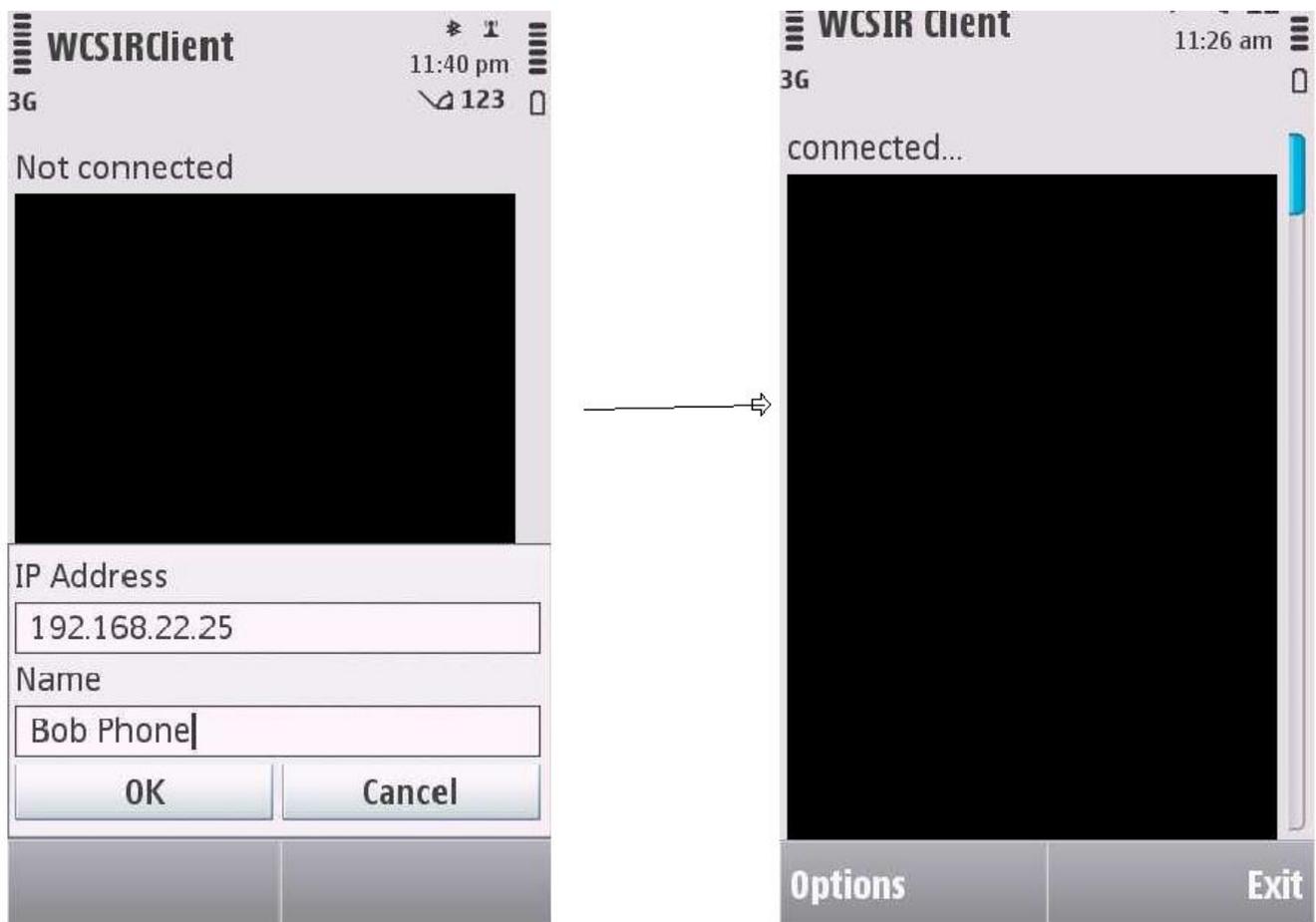


FIGURE 19: Connecting to the WCSIRServer application on the workstation.

Getting commands from the Smartphone to the robot and camera is done by sending the specified command to the WCSIRServer application which then re-transmits the command request to the robot. This is tested by checking from

the WCSIRServer application during the debugging if the commands sent from the smartphone are actually received. When the TCP/IP wireless connection is established between the WCSIRClient application on the Smartphone and the WCSIRServer application on the workstation, the WCSIRClient application can start sending commands to the WCSIRServer application. In the options menu of this application on the Smartphone there are specific commands for controlling the movement of the robot and camera. These commands are “Robot up”, “Robot down”, “Robot left”, “Robot right”, “Camera up”, “Camera down”, “Camera left” and “Camera right”, they send commands separately to the robot and to the camera(Figure 20).



FIGURE 20: Menu options on the WCSIRClient application

5.3 Testing the video streaming

During the implementation phase of this thesis work, a live video stream was tested by several ways as it is very important to get the video output on the WCSIRServer application which will retransmit the video frames to the WCSIRClient application running on the Smartphone. At the beginning the video streams obtained from the camera were tested by using the HP MediaSmart webcam software which captures the video stream from the video card. The video quality produced was of good quality and had a relatively small delay.

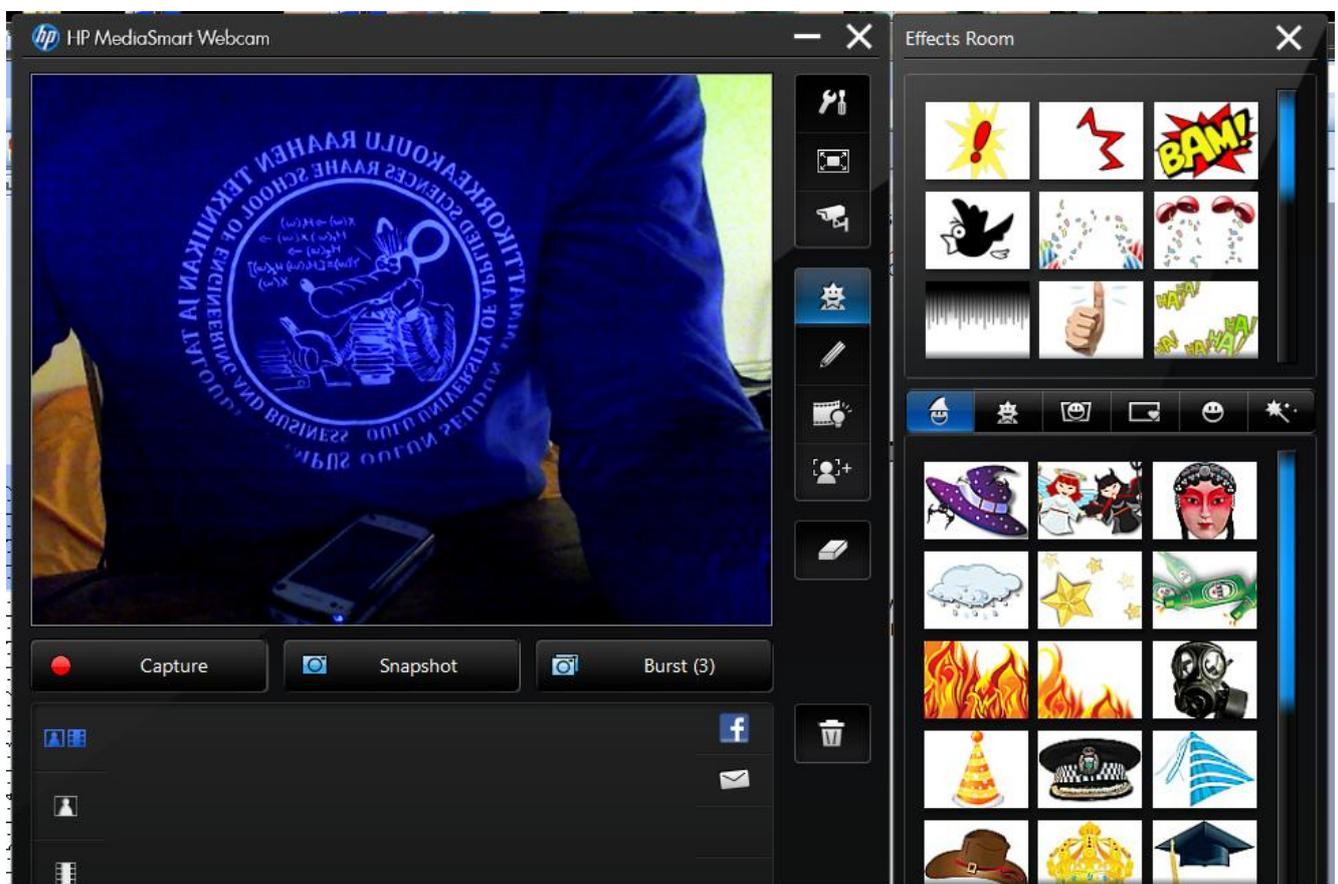


FIGURE 21: Running an HP MediaSmart Webcam application.

At the beginning of this thesis work getting video streams on a Qt application was not possible until Nokia Qt Development Framework division released the Qt Mobility API, it is an additional set of APIs which are not included in the Qt SDK and they support playing and streaming media contents. After this additional feature had been installed, the Qt SDK was used in developing

these applications. The relevant codes for displaying media contents were added to the source code and compiled video frames were received on the WCSIRServer application and also on the WCSIRClient application.

These led to the conclusion that for easy playing of media content on a Qt application, the Qt Mobility API should be installed into the Qt SDK. At the time of implementation the Qt Mobility API version released was the Qt Mobility 1.1.1.

6 POSSIBILITY OF FURTHER DEVELOPMENT

This Wireless Control System for an Industrial Robot can be further developed to these directions:

- The system can be developed to support the addition of several robots and to control them too.
- The system can be developed to include the addition of multiple cameras positioned at strategic locations to provide more dimensional views to the robot.
- The Mobile application can be made to directly establish a serial COM connection to the robot and the camera.

7 CONCLUSION

7.1. DESIGN AND IMPLEMENTATION

The main software technology used for this bachelor's thesis is the Qt Framework development tools. The hardware section of this project consists of several devices that should work together providing intended behaviour and compatibility to the entire system. Understanding the different devices and technologies makes this project an interesting topic for a software developer.

While designing and implementing this Wireless Control System project, the initial plans changed several times, initially the plan was to use a wireless access point to provide the wireless channel from the workstation to the smartphone but it was later discovered that wireless TCP/IP connections can be made directly from the smartphone to the workstation as long as there is no firewall preventing the connection. These changes were made to improve the general performance of the entire system and also to cut financial expenses.

The implementation of the project was quite a challenging and educational task, which gave me deeper understanding of Qt application's development tools on the main platforms used in this project. The tools used for testing, debugging and design were also better understood.

At the initial phase of this project while investigating the Qt Framework's API to be used in the development, it was discovered that the Qt Framework does not yet have support for Camera API which provides the possibility of getting video frames on its applications from a camera device. This caused a delay in the development because an alternative approach was sought, Later Qt Development Frameworks released a new set of APIs called the Qt Mobility API which has support for getting camera device frames(Nokia 2011, Date of retrieval 31.03.2011). In this project the camera device is attached to the Windows enabled workstation used to run the WCSIRServer application.

7.2 SAFETY MEASURES

Safety measures play one of the most important roles in this Wireless Control System for Industrial robot system design. If safety measures are not taken into account, the Industrial robot can be dangerous to its environment and mostly to people. The robot's movements can become unpredictable and may result severe consequences. The safe and proper handling of the robots movements was considered at every phase of the WCSIR development from the beginning of the design phase to the conclusion phase.

The robot's movements were designed without the need of eliminating the possibility of collision between the robot and other peripheral devices close-by. The position definition was made considering the operational limits of the robot and with reference to the technical specification of the robot. All robot's movements lie within specific limits and are safe for the operator and the spectators and also for the working environment. The robot's movement were tested in the step mode at its lowest speed to ensure a proper operation. The Server and mobile application's command to the robot was designed to be easily understood and identified so that accidents can be reduced. All these measures increased the overall safety of the system.

All work with the RV-E3J robot was made in accordance with the Movemaster's safety manual and also considering other conditions such as atmosphere, electromagnetic fields, static electricity, humidity, temperature and air pressure(Movemaster, 1994).

8. REFERENCES

1. Heiniein R. 1942. Waldo (Short story).
USA: Doubleday.
2. MOVEMASTER. 1994.
Mitsubishi Industrial robot reference manual. RV-E3J.
Mitsubishi Electronics Europe GMBH. Industrial Automation.
3. Musijenko E. 2006, Wireless control system for industrial robot.
Raahe, Finland.
Oulu University of Applied Science.
4. Nokia Corporation. 2011. Qt.
Date of retrieval 30.03.2011
<http://qt.nokia.com/>
5. Nokia Corporation. 2011. Qt Framework's API.
Date of retrieval: 31.03.2011
<http://doc.qt.nokia.com/4.7/index.html>
6. SONY. 2011. Commands list.
Date of retrieval: 31.03.2011
<http://bssc.sel.sony.com/Professional/docs/manuals/evid30commandlist1-21.pdf>
7. Wikipedia. 2011. Agile testing,
Date of retrieval 30.03.2011
http://en.wikipedia.org/wiki/Agile_testing
8. Wikipedia. 2011. Telerobotics,
Date of retrieval 31.03.2011
<http://en.wikipedia.org/wiki/Telerobotics>
9. Wikipedia. 2011. Qt(Framework),
Date of retrieval 31.03.2011
[http://en.wikipedia.org/wiki/Qt\(framework\)](http://en.wikipedia.org/wiki/Qt(framework))

9. APPENDICES

APPENDIX 1: SOURCE CODE OF A CLASS ACTING AS A TCP SERVER

APPENDIX 2: SOURCE CODE OF MAINWINDOW CLASS FOR THE
WCSIRSERVER

APPENDIX 3: SOURCE CODE OF CLIENT'S MAINWINDOW CLASS

APPENDIX 4:: CLIENT'S LOGIN DIALOG CLASS

APPENDIX 1

SOURCE CODE OF A CLASS ACTING AS A TCP SERVER

```
#include <QTcpServer>
#include <QListWidget>
#include <QHostAddress>
#include <QLabel>
#include <QNetworkInterface>
#include <QTcpSocket>
/*
   A server class used for receiving socket connections from smartphone's app
*/
class ChatServer : public QTcpServer
{
    Q_OBJECT
public:
    ChatServer(QObject* parent, QLabel* notice);
signals:
    void robotupRequested();
    void robotdownRequested();
    void robotleftRequested();
    void robotrightRequested();
    void cameraupRequested();
    void cameradownRequested();
    void cameraleftRequested();
    void camerarightRequested();
protected:
    void incomingConnection(int socketId);

private slots:
    // slots for handling socket data and state
    void readClient();
    void ClientDisconnected();
    void ClientConnected();

private:
    QHostAddress iHostAddress;
    QLabel* messageBoard;
    QTcpSocket *iTcpSocket ;
};
```

```

ChatServer::ChatServer(QObject* parent, QLabel* notice)
    :QTcpServer(parent)
{
    //UI component for displaying notifications
    messageBoard = notice;

    // 13001 is the dedicated port for this application
    int port = 13001;
    iHostAddress = QHostAddress(QHostAddress::LocalHost);

    /*
    Server listening for incoming connections
    on the it's IP address and port and notifies the UI
    */
    if( !this->listen(iHostAddress, port ))
    {
        messageBoard->setText("Error in starting server!");
        return;
    }
    messageBoard->setText(QString("IP: %1 Please connected client ")
        .arg(iHostAddress.toString()));
}
//server responds to incoming connections
void ChatServer::incomingConnection(int socketId)
{
    qDebug()<<"incomingConnection...";
    QTcpSocket* socket = new QTcpSocket(this);
    socket->setSocketDescriptor(socketId);
    connect(socket, SIGNAL(connected()), this, SLOT(ClientConnected()));
    connect(socket, SIGNAL(readyRead()), this, SLOT(readClient()));
    connect(socket, SIGNAL(disconnected()), this,
    SLOT(ClientDisconnected()));
    connect(socket, SIGNAL(disconnected()), socket, SLOT(deleteLater()));
}
//The Server responds to incoming data received from client
void ChatServer::readClient()
{
    if (((QTcpSocket*)sender())->canReadLine())
    {
        QString receivedCommand(((QTcpSocket*)sender())->readLine());
        /*
        identifies each command from client and then makes
        the specific request to robot or camera
        */
    }
}

```

```

*/
if(receivedCommand.toInt() == 1){
    emit robotupRequested();
}
if(receivedCommand.toInt() == 2){
    emit robotdownRequested();
}
if(receivedCommand.toInt() == 3){
    emit robotleftRequested();
}
if(receivedCommand.toInt() == 4){
    emit robotrightRequested();
}
if(receivedCommand.toInt() == 5){
    emit cameraupRequested();
}
if(receivedCommand.toInt() == 6){
    emit cameradownRequested();
}
if(receivedCommand.toInt() == 7){
    emit cameraleftRequested();
}
if(receivedCommand.toInt() == 8){
    emit camerarightRequested();
}
}
}
//SLOT to handle when the clients are disconnected
void ChatServer::ClientDisconnected()
{
    messageBoard->setText(QString("IP: %1 No Client connected ")
        .arg(iHostAddress.toString()));
}
//when client is conencted the UI is updated with the new connection state
void ChatServer::ClientConnected()
{
    messageBoard->setText(QString("Client connected..."));
}
}

```

APPENDIX 2

SOURCE CODE OF MAINWINDOW CLASS FOR THE WCSIRSERVER

```
#include <QMainWindow>
#include "chatserver.h"
#include <QTcpSocket>
namespace Ui {
    class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    //SLOTS to send commands to the robot and camera
    void on_upCamera_clicked();
    void on_leftCamera_clicked();
    void on_rightCamera_clicked();
    void on_downCamera_clicked();
    void on_upRobot_clicked();
    void on_leftRobot_clicked();
    void on_rightRobot_clicked();
    void on_downRobot_clicked();
private:
    Ui::MainWindow *ui;
    ChatServer* chatServer;
};

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //server instance is created
    chatServer = new ChatServer(this, ui->connectionNotice);
    connect(chatServer, SIGNAL(robotupRequested()), this,
    SLOT(on_upRobot_clicked()));
```

```

    connect(chatServer, SIGNAL(robotleftRequested()), this,
    SLOT(on_leftRobot_clicked()));
    connect(chatServer, SIGNAL(robotrightRequested()), this,
    SLOT(on_rightRobot_clicked()));
    connect(chatServer, SIGNAL(robotdownRequested()), this,
    SLOT(on_downRobot_clicked()));
    connect(chatServer, SIGNAL(cameraupRequested()), this,
    SLOT(on_upCamera_clicked()));
    connect(chatServer, SIGNAL(cameradownRequested()), this,
    SLOT(on_downCamera_clicked()));
    connect(chatServer, SIGNAL(cameraleftRequested()), this,
    SLOT(on_leftCamera_clicked()));
    connect(chatServer, SIGNAL(camerarightRequested()), this,
    SLOT(on_rightCamera_clicked()));
    QVBoxLayout *layout = new QVBoxLayout;

    //UI item to display the video frames from camera device
    Phonon::MediaObject *media = new Phonon::MediaObject();
    Phonon::VideoWidget *vwidget = new Phonon::VideoWidget();
    Phonon::createPath(media, vwidget);

    //gets video frames and start playing
    media->setCurrentSource(
        Phonon::MediaSource("");
    media->play();
    layout->addWidget(vwidget);
    ui->widget->setLayout(layout);
}

```

APPENDIX 3

SOURCE CODE OF CLIENT'S MAINWINDOW CLASS

```
#include <QLabel>
#include <QTcpSocket>
#include <Phonon/MediaObject>
#include "loginview.h"
#include <QMenuBar>
#include <QAction>
#include <QVBoxLayout>
#include <QUrl>
#include <Phonon/VideoWidget>

namespace Ui {
    class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    virtual ~MainWindow();
    void displayVideoStream();

public slots:
    void connectToServer();
    void moveRobotDown();
    void moveRobotUp();
    void moveRobotLeft();
    void moveRobotRight();
    void moveCameraDown();
    void moveCameraUp();
    void moveCameraLeft();
    void moveCameraRight();
    void setNotice();
    void setNoticeDisconnect();
    void forceStop();
    void readServer();

private:
    Ui::MainWindow *ui;
```

```

    QLabel *note;
    QTcpSocket *tcpSocket ;
    LoginView *loginV;
    Phonon::MediaObject *media;
};

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("WCSIR Client");
    tcpSocket = loginV->getSocket();
    this->menuBar()->addAction("Connect to Server", this,
    SLOT(connectToServer()));
    this->menuBar()->addAction("Robot Up", this, SLOT(moveRobotUp()));
    this->menuBar()->addAction("Robot Down", this,
    SLOT(moveRobotDown()));
    this->menuBar()->addAction("Robot Left", this, SLOT(moveRobotLeft()));
    this->menuBar()->addAction("Robot Right", this, SLOT(moveRobotRight()));
    this->menuBar()->addAction("Camera Up", this, SLOT(moveCameraUp()));
    this->menuBar()->addAction("Camera Down", this,
    SLOT(moveCameraDown()));
    this->menuBar()->addAction("Camera Left", this,
    SLOT(moveCameraLeft()));
    this->menuBar()->addAction("Camera Right", this,
    SLOT(moveCameraRight()));
    this->menuBar()->addAction("ROBOT STOP!", this, SLOT(forceStop()));
    media = new Phonon::MediaObject();
    Phonon::VideoWidget *vwidget = new Phonon::VideoWidget();
    Phonon::createPath(media, vwidget);

    note = new QLabel("Not connected", this);
    QVBoxLayout *mainLayout = new QVBoxLayout(this);
    mainLayout->addWidget(note);
    mainLayout->addWidget(vwidget);
    this->centralWidget()->setLayout(mainLayout);

    loginV = new LoginView(this);
    connect(loginV, SIGNAL(serverConnected()), this, SLOT(setNotice()));
    connect(loginV, SIGNAL(serverDisconnected()), this,
    SLOT(setNoticeDisconnect()));
    connect(loginV, SIGNAL(cleintReadyy()), this, SLOT(readServer()));
}

```

```

void MainWindow::moveRobotUp()
{
    tcpSocket->write(QString("1").toAscii() + "\n" );
}
void MainWindow::moveRobotDown()
{
    tcpSocket->write(QString("2").toAscii() + "\n" );
}
void MainWindow::moveRobotLeft()
{
    tcpSocket->write(QString("3").toAscii() + "\n" );
}
void MainWindow::moveRobotRight()
{
    tcpSocket->write(QString("4").toAscii() + "\n" );
}
void MainWindow::moveCameraUp()
{
    tcpSocket->write(QString("5").toAscii() + "\n" );
}
void MainWindow::moveCameraDown()
{
    tcpSocket->write(QString("6").toAscii() + "\n" );
}
void MainWindow::moveCameraLeft()
{
    tcpSocket->write(QString("7").toAscii() + "\n" );
}
void MainWindow::moveCameraRight()
{
    tcpSocket->write(QString("8").toAscii() + "\n" );
}
void MainWindow::setNotice()
{
    note->setText("connected...");
    displayVideoStream();
}
void MainWindow::setNoticeDisconnect()
{
    note->setText("Not connected...");
}

```

APPENDIX 4

CLIENT'S LOGIN DIALOG CLASS

```
/*
A Dialog class for connecting to the server.
*/
#include <QDialog>
#include <QTcpSocket>
#include <QLineEdit>
#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QLabel>
#include <QPushButton>

class LoginView: public QDialog
{
    Q_OBJECT
public:
    LoginView(QWidget *parent);
    ~LoginView();
private slots:
    void connectToServer();
    void hostConnected();
    void hostDisconnected();
signals:
    void serverConnected();
    void serverDisconnected();
    void cleintReadyy();
private:
    QTcpSocket *tcpSocket ;
    QLineEdit *ipAddress;
};

LoginView::LoginView(QWidget *parent): QDialog(parent)
{
    /*
constructs and set the UI components
*/
    QVBoxLayout *layout = new QVBoxLayout(this) ;
    QLabel *ipAddre = new QLabel("IP Address", this);
    QLabel *name = new QLabel("Name", this);
```

```

ipAddress = new QLineEdit("127.0.0.1",this);
QLineEdit* userName = new QLineEdit("My Phone",this);
QPushButton* okButton = new QPushButton("OK", this);
connect(okButton, SIGNAL(clicked()), this, SLOT(connectToServer()));
QPushButton* cancelButton = new QPushButton("Cancel", this);
connect(cancelButton, SIGNAL(clicked()), this, SLOT(close()));
layout->addWidget(ipAddress);
layout->addWidget(name);
layout->addWidget(userName);
QHBoxLayout *hLayout = new QHBoxLayout ;
hLayout->addWidget(okButton);
hLayout->addWidget(cancelButton);
layout->addLayout(hLayout);
tcpSocket = new QTcpSocket(this);
connect(tcpSocket, SIGNAL(connected()), this, SLOT(hostConnected())) ;
connect(tcpSocket, SIGNAL(disconnected()), tcpSocket,
SLOT(deleteLater()));
    connect(tcpSocket, SIGNAL(disconnected()), this,
SLOT(hostDisconnected())) ;
    connect(tcpSocket, SIGNAL(readyRead()), this, SIGNAL(cleintReadyy()));
}
void LoginView::connectToServer()
{
    tcpSocket->abort();
    tcpSocket->connectToHost(ipAddress->text(), 13001);
}
void LoginView::hostConnected()
{
    this->hide();
    emit serverConnected();
}
void LoginView::hostDisconnected()
{
    qDebug()<< "server disconnected..." ;
    emit serverDisconnected();
}

```