Creating Remote Operating System to control Robot in remote area with Telematics

MSc. Karwan M. Kareem

Abstract- inventing and developing of remote control systems with tele-operated system have been an interesting field for producing new models of communications technologies that create high level of control of the operation of device, as a television set, from distance area. This needs a communication channel to transfer data between different electronic machines via using specific host, port number that is accepted by client and server side technologies.

Index Terms— socket programming, java thread, CORBA, artificial intelligent, tele-robot, RMI, remote control, tele-operation, buffered reader, buffered writer

1.0 INTRODUCTION

Remote control has been one of the best systems that control the science of sending, receiving and storing information by telecommunication devices. This is a new model of electronic system that able to control the transmission of useful data to and from a device in remote area. This research try to use new model technology like socket programming, which is the software abstraction used to represent the "terminals" of a connection between two machines, to develop remote operating system to control robot from distance area.

The architecture of the system contains client and server sides' component, which is implemented by using Java thread technology. Therefore, different users have opportunity to drive the robot via different machines at the same time without any problem. The structure of the system consists of seven classes that have been implemented by object oriented programming. These java classes interact together to control the robot via commands that send from client side, it's also able to control all directions, positions and behaviors of the robot. The system able to display information being sent to the system, and it's also able to display information being sent back from the server (position of device and proximity to objects).

1.1 Goals of the research:

The main aim of this research is to develop remote operating system to control robot in remote area with tele-operated system through using Java RMI, CORBA or Socket programming technology.

1.2 Objectives of the research:

1- The project should contain client and server side component.
2- The project should display information being sent to the server.
3- The project should display information being sent back from the server (position of device and proximity to objects).

1.3 Minimum Functionalities:

1- This robot should control by commands which send from client side application.
2- Driver should able to control direction, position of the robot in darken room.
3- Driver should able to control the behaviors of the robot via arrow keys on the keyboard.

2.0 Literature review:

2.1 Java Socket technology:

Java socket is a communication channel to transfer data between different computers via using specific host, port number that is accepted by client and server sides of the project. In this technology, port number is a key element which makes interaction between client and server side of the project. Bruce Eckel claims that *"The* socket is the software abstraction used to represent the "terminals" of a connection between two machines*"*.

2.2 Java thread technology:

A series of messages and data that have been exchanged between client and server side technologies as replies to each other. This technology can provide multi ports for same host. Therefore, several users can drive the robot via different machines at the same time without any problem.

2.3 Remote Operating System:

It's a system that able to control other system behaviors by a person from distance area at different place. The remote operating systems consists client and server side technologies which are interact each other by ultrasonic signals or by electrical signals transmitted via wire. Also called a machine used to control the operation of devise, as a television set, from distance area.

2.4 Telematics: Tele-operated system:

According to their degree of autonomy, robots can be classified into: repetitive operation tele-operated and autonomous or intelligent. Means tele-operation the case where a human operator, remotely located with respect to robot manipulates it. This is incompatible with a tele-operated robotic autonomy, understood as a case in which the control and decision making are performed by the robot thereof.

The limitations of these systems lie in the processing capacity of the signals and thus precision, and human-robot coordination. It is remarkable that the transmission delay plays an important role and should be considered in designing the control system. The design of man-machine interface is usually critical. For which usually lies with the operator tasks in decision making based on sensory information, experience and expertise. Current areas of research and development include:

a. The manipulator and mobile robot control.
b. The architecture of the remote tele-robot.
c. Processing, integration and fusion of the sensory system.
d. Interactive tasks planned and executed.
e. The graphical display of superimposed images.
f. Multi-sensor - balanced control.
g. Micro-mechanisms - control for the deployment of the instruments.
(Beniger, 1986, pp 12-34)

2.5 Class diagram:

It is one of the UML standard design languages. This modeling language is very valuable to design, develop and build complex computer application. This diagram illustrates the static structures of the project (to define the relationship between system entities such as users, things and data).

2.6 Sequence diagram:

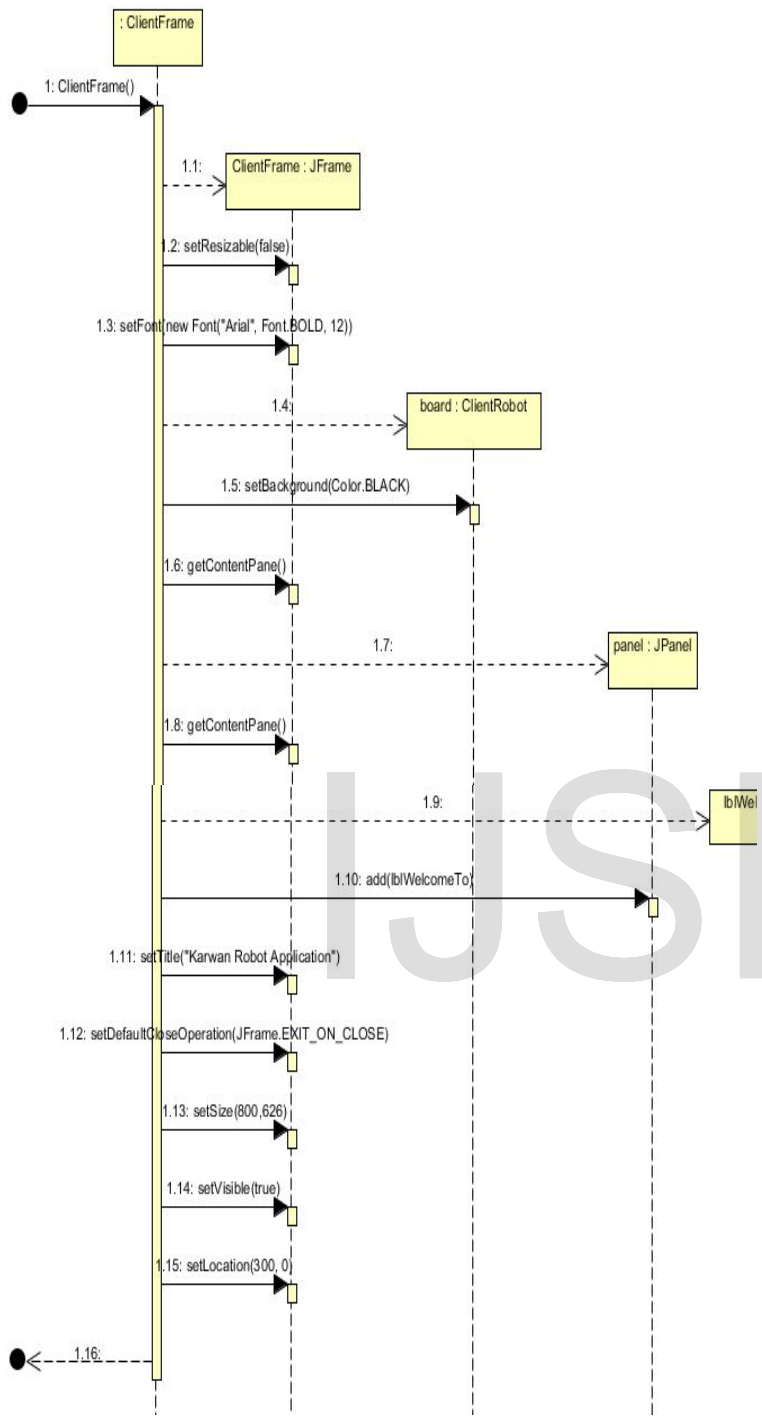
It is independent design programming-language. This diagram shows the relation between different objects in the same Use Case that help developer to define interaction between whole objects of the system**.** Class and sequence diagrams are very helpful for the developer in several aspects that include:

a. Define system requirements.
b. Define the interaction between system objects and project architecture.
c. Define the relationship between system entities and data structure.

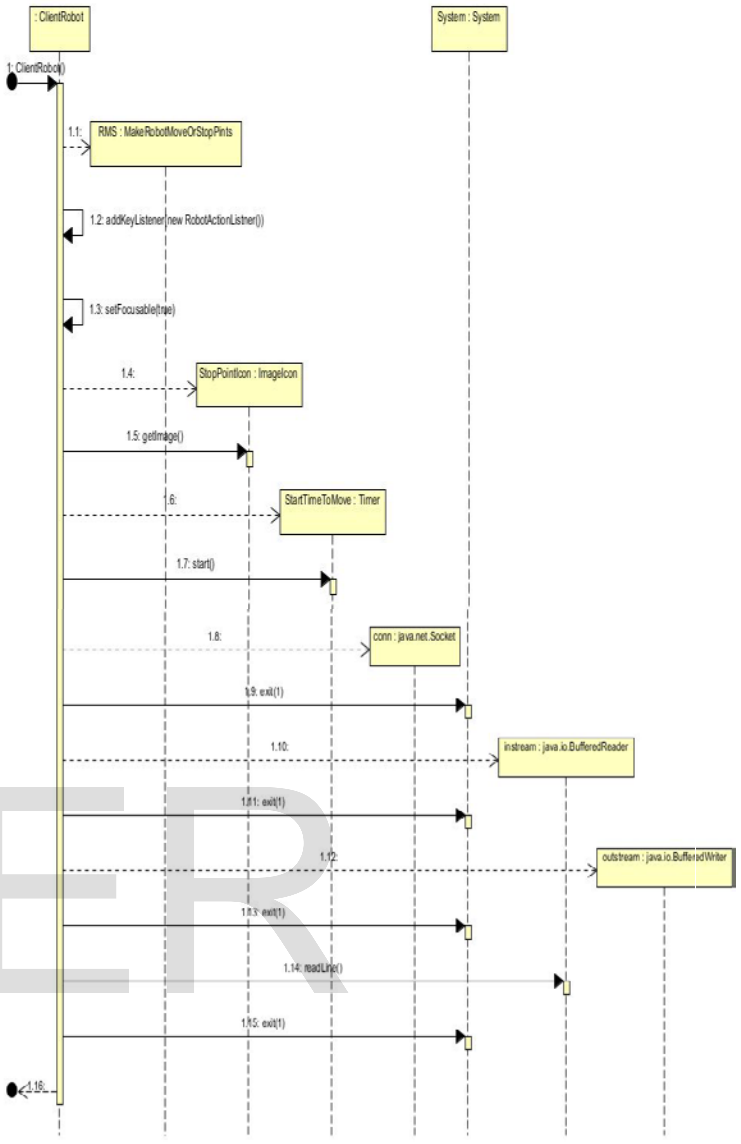
3.0 Design and implementation:

In this part of the research, the researcher tries to design a high level architectural of the application "client side and server side of the robot" that include sequence and class diagrams, with implement all JAVA programming classes which are used to build the project.

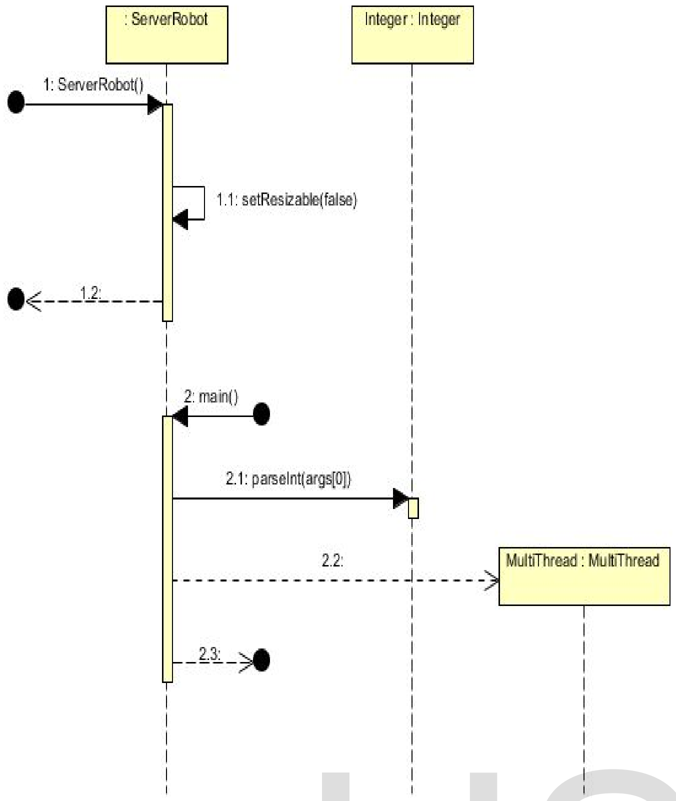
3.1 A high level architectural view of the application - Sequence Diagrams:



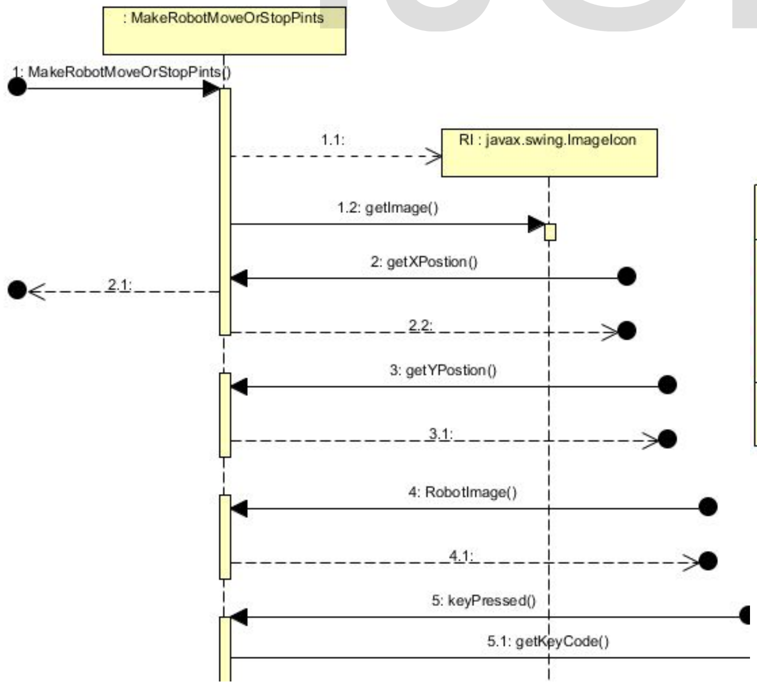
(Figure 1: ClientFrame.Java - Sequence Diagram)

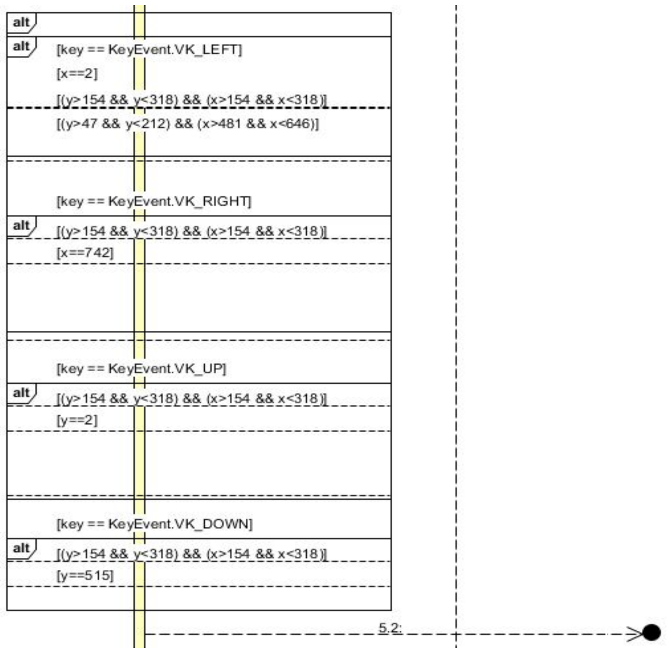


(Figure 2: ClientRobot.java - Sequence Diagram)



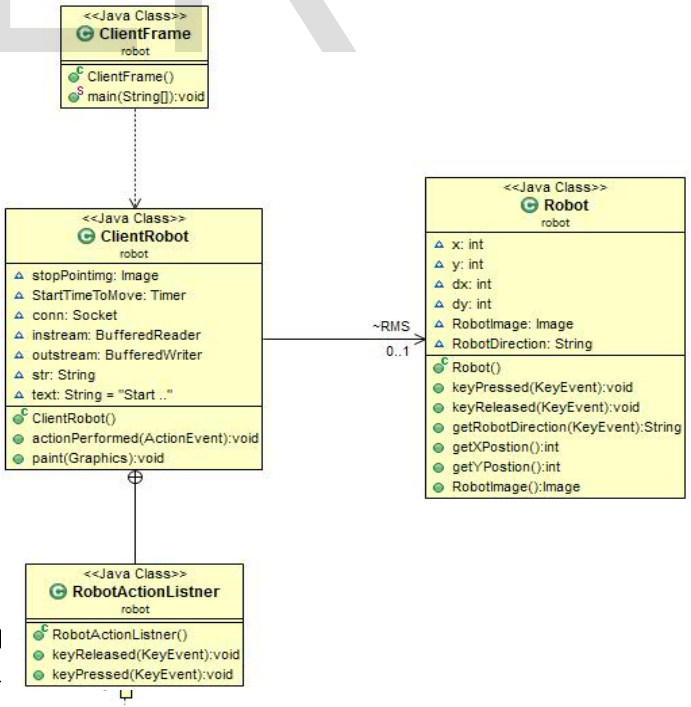
(Figure 3: ServerRobot.java - Sequence Diagram)



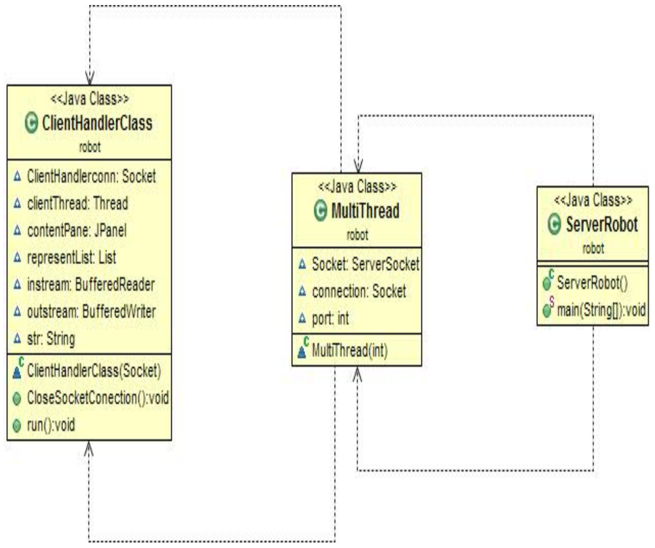

alt
alt
[key == KeyEvent.VK_LEFT]
[x==2]
[(y>154 && y<318) && (x>154 && x<318)]
[(y>47 && y<212) && (x>481 && x<646)]
[key == KeyEvent.VK_RIGHT]
alt
[(y>154 && y<318) && (x>154 && x<318)]
[x==742]
[key == KeyEvent.VK_UP]
alt
[(y>154 && y<318) && (x>154 && x<318)]
[y==2]
[key == KeyEvent.VK_DOWN]
alt
[(y>154 && y<318) && (x>154 && x<318)]
[y==515]
5.2:

(Figure 4: MakeRobotMoveOrStopPaints.java - Sequence Diagram)

3.2 A high level architectural view of the application - Class Diagrams:



(Figure 5: Class diagram - client side of the robot)



(Figure 6: Class diagram – server side of the robot)

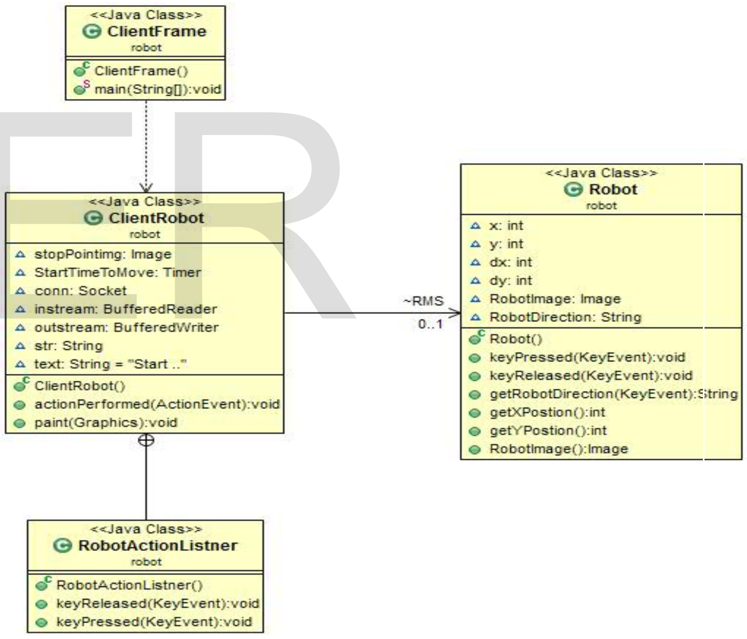
3.3 Server and client side implementation:

The structure of the project consists of seven classes. These java classes interact together to develop remote robot which can be controlled by driver from distance area. The robot was developed by Client- server technology which was implemented by Java Socket technology. Java socket is a communication channel to transfer data between different computers via using specific host, port number that is accepted by client and server sides of the project. In this technology, Port number is a key element which makes interaction between client and server side of the project. Bruce Eckel claims that "The socket is the software abstraction used to represent the "terminals" of a connection between two machines".

In addition, java thread is another technology which was used to develop the project. This technology can provide multi ports for same host. Therefore, numerous users can drive the robot via different machines at the same time without any problem. This robot can control by commands which send from client side application. Driver can control direction, position of the robot in darken room which include three stop-points in different position. The robot cannot pass through stop-points, but it can go around them. Moreover, the robot cannot pass through the walls of the darken room.

Driver can control the behaviors of the robot via arrow keys on the keyboard. When the robot changes its location, direction in the room, its send back the messages to client. Each message includes direction, position of the robot. Current position is represented by (x, y) points on the client handler screen. When driver presses right arrow key on the keyboard, then client sends direction, position messages to server. After, server side will accept client message, and also it will send back the message to client side. This process will be continues until the driver changes, releases the arrow key. Client side of the project consists of four classes (figure 7).

This part of the research will explain that what is the component of these classes? How they interact each other. First class is Robot.java, which care about behaviors, structure of the robot. This class include its instance variables which are x, y, dx and dy. These variables represent position of the robot, and also RobotDirection is another variable that is responsible for the direction of the robot. Furthermore, this class return (x, y) position, robot direction and icon image of the robot. This class is responsible for that where the robot can go or not? For example, robot cannot pass through stop-points, but it can go around them.

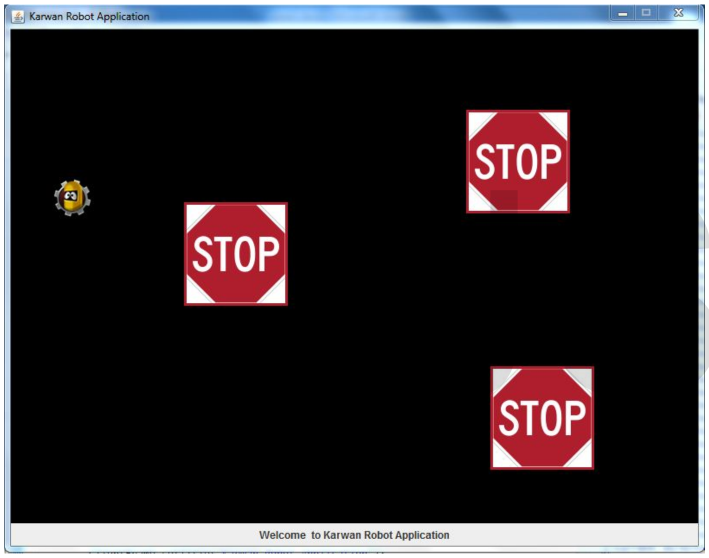


(Figure 7: Class diagram - client side of the robot)

Each stop point has four bounds, each bound has specific x or y position on the screen. When the bound of the robot encounters one of the bounds of the stop points, then the robot must go back five pixels. Therefore, robot object cannot path through stop-point objects. Furthermore, the darken room has four bounds which were represented by x or y position on the screen. When the robot encounters one of them, then it must go back 15 pixels that dependent of the robot directions, positions.

- First stop-point area, bounds: ((y>154 && y<318) && (x>154 && x<318)).
- Second stop-point area, bounds: ((y>47 && y<212) && (x>481 && x<646)).
- Third stop-point area, bounds: ((y>345 && y<509) && (x>509 && x<673)).
- Darken room bounds are: x==2, x==742, y==2 and y==515.

Second class is ClientFrame which includes client frame method, main method. This class consists of one frame which includes one Jpanel, to make the darken room, and also it obtains new board object from ClientRobot class to represent robot icon, three stop-point icons. Look (figure 8). Generally, ClientFrame class is graphic user interface class which makes darken room that provides three stop-point icons, one robot icon for user.



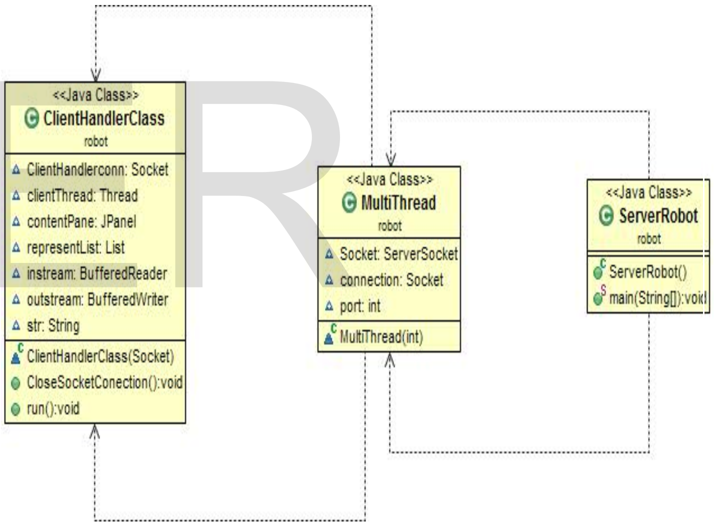
(Figure 8: ClientFrame – Darken room)

Third class is ClientRobot class which extends Jpanel, implements ActionListener. This class includes Robot-class that is responsible for the robot behaviors. Moreover, ClientRobot class includes paint method which draws three stop-point icons on the darken room, and also it draws robot icon by getting (x, y) position of the robot from robot object. Stop-point icons have static position, but robot icon has dynamic (x, y) position that assist robot icon to move around the room by using arrow keys.

When user press arrow keys, robot changes its (x, y) position. Moreover, ClientRobot class sends current (x, y) position of the robot to serverRobot class by using writes method (outstream.write (text)). This method writes current (x, y) position of the robot on the new (Bufferedwritter) object which use local host, 8888 port number to make connection with server side object via using java socket technology.

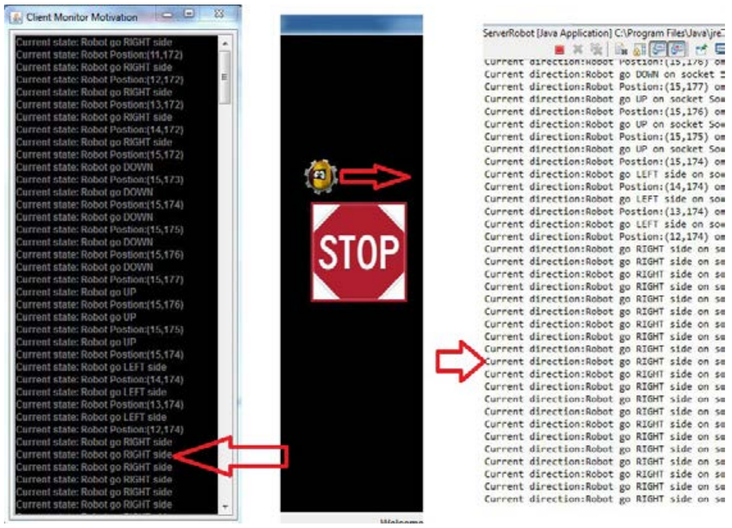
Server side of the project consists of three classes. Look (figure 9). First class is serverRobot which extends JFrame, includes portNumber instance variable that is passed by new MaltiThread object. MaltiThread is a second class of the server side. This class defines its constrictors which are port number, serverSocket and socket objects to establish server socket, and listen for client.

This class can provide multi-port numbers for clients that make a good traffic between client and server sides. Therefore, numerous clients can interact with server to drive the robot without any problem. Third class is clientHandlerClass which extends JFrame, implements Runnable. This class defines its instance variables which are socket, thread, BufferedReader and BufferedWiter, and also it includes socket and thread constructors to build multiply connection between sereverRobot and clientHandlerClass classes.



(Figure 9: Class diagram – server side of the robot)

Moreover, BufferedReader objects reads client message by using InputStreamReader object, but BufferedWriter object response client message via using OutputStreamWriter object. Client handler class defines new JList box to print current value of InputStreamReader object. This class includes Run () method which print Current value of "InputStreamReader" on client handler list box by using "str" instance variable. Consequently, all states of the robot are printed on the client handler screen. Look (figure 10).


Client Monitor Motivation
STOP
ServerRobot [Java Application] C:\Program Files\Java\jre

```
Current state: Robot go RIGHT side
Current state: Robot Postion:(11,172)
Current state: Robot go RIGHT side
Current state: Robot Postion:(12,172)
Current state: Robot go RIGHT side
Current state: Robot Postion:(13,172)
Current state: Robot go RIGHT side
Current state: Robot Postion:(14,172)
Current state: Robot go RIGHT side
Current state: Robot Postion:(15,172)
Current state: Robot go DOWN
Current state: Robot Postion:(15,173)
Current state: Robot go DOWN
Current state: Robot Postion:(15,174)
Current state: Robot go DOWN
Current state: Robot Postion:(15,175)
Current state: Robot go DOWN
Current state: Robot Postion:(15,176)
Current state: Robot go DOWN
Current state: Robot Postion:(15,177)
Current state: Robot go UP
Current state: Robot Postion:(15,176)
Current state: Robot go UP
Current state: Robot Postion:(15,175)
Current state: Robot go UP
Current state: Robot Postion:(15,174)
Current state: Robot go LEFT side
Current state: Robot Postion:(14,174)
Current state: Robot go LEFT side
Current state: Robot Postion:(13,174)
Current state: Robot go LEFT side
Current state: Robot Postion:(12,174)
Current state: Robot go RIGHT side
Current state: Robot go RIGHT side
Current state: Robot go RIGHT side
Current state: Robot go RIGHT side
Current state: Robot go RIGHT side
Current state: Robot go RIGHT side
```

```
Current direction:Robot Postion:(15,176) on
Current direction:Robot go DOWN on socket S
Current direction:Robot Postion:(15,177) on
Current direction:Robot go UP on socket So
Current direction:Robot Postion:(15,176) on
Current direction:Robot go UP on socket So
Current direction:Robot Postion:(15,175) on
Current direction:Robot go UP on socket So
Current direction:Robot Postion:(15,174) on
Current direction:Robot go LEFT side on so
Current direction:Robot Postion:(14,174) on
Current direction:Robot go LEFT side on so
Current direction:Robot Postion:(13,174) on
Current direction:Robot go LEFT side on so
Current direction:Robot Postion:(12,174) on
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
Current direction:Robot go RIGHT side on se
```

(Figure 10: client handler screens)

4.0 Conclusion:

It is a fact that remote control system is one of the best systems to sending, processing, receiving and storing information from a device in remote area via telecommunication tools. Using socket programming, java thread and OOP programming are also having a major impact on improving Tele-robotic systems.

In closing, new model technology like socket programming is *used to* develop the system that able to control the robot. Different users have ability to drive the robot via different machines at the same time. In the system, seven java classes interact together to control the robot via commands that send from client side. The system can control directions, positions and behaviors of the robot. It's able to display information being sent to the system, and it's also able to display information being sent back from the server (position of device and proximity to objects).

REFERENCE

[1] Amoroso, A. (2007) *Cyber security*. London: Silicon Press. [Online]. Available at:<http://library.books24x7.com.libaccess.hud.ac.uk/toc.aspx?site=MVJ2L&bookid=15450>. [Accessed 23 NOV 2011].

[2] Allen, G., Jenkins, T. (2012) *Distributed Objects - Sockets, Java RMI, and CORBA*. (Unpublished lecture). University of Huddersfield.

[3] Allen, G., Jenkins, T. (2012) *Concurrency Concepts and Java Threads*. (Unpublished lecture). University of Huddersfield.

[4] Bell, D. (2003). *UML basics: An introduction to the Unified Modeling Language*. Available: <http://www.ibm.com/developerworks/rational/library/769.html/>. (Accessed on 29 June 2015.)

[5] Croft, David Wallace. (2004) *Advanced Java game programming*. Berkeley, Calif: Apress.

[6] Foster, James C. (2005) *Sockets, Shellcode, Porting, and Coding: Reverse Engineering Exploits and Tool Coding for Security Professionals*. Rockland: Syngress Publishing.

[7] Harbour, Jonathan S. (2007) *Beginning Java Game Programming*. (2nd Edition). Boston: Course Technolgy.

[8] Micro-mechanisms - control for the deployment of the instruments. (Beniger, 1986, pp 12-34).

[9] Oaks, S., Wong, H. (2006) *Java threads: understanding and mastering concurrent programming*.

[10] Oracle web site (2012) *All about Sockets*. [Online]. Available at <http://docs.oracle.com/javase/tutorial/networking/sockets/index.html > [Accessed at: 10-04-2012].

[11] W. Miller, R., Williams, A. (2001) *Java sockets*. [Online]. Available at:
< http://www.ibm.com/developerworks/java/tutorials/j-sockets/j-sockets-pdf.pdf > [Accessed at: 9-04-2012].

MSc. Karwan M. Kareem
Karwanmus@yahoo.com
University of Sulaimani
Faculty of Physical and Basic Education
Department of Computer Science