Bachelor's Thesis

Information Technology

2010

Vu Ba Tien Dung

DEVELOPING A SELF-BALANCING ROBOT WITH LEGO NXT MINDSTORMS

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology

Spring 2010 | 119

Slotte Vesa

Vu Ba Tien Dung

Developing a Self-balancing Robot with Lego NXT Mindstorms

The objective of this thesis is to develop an initial framework for a remote-controlled robot. The key issues of this thesis are twofold: first, developing a two-wheeled self-balancing robot which has many advantages compared to the traditional robot, and secondly, designing the communication protocols which are needed in the system.

The study covers the control system theory, robotic programming, and traditional network programming. After the research literature, this thesis presents the detailed designs of the two-wheeled self-balancing robot's control system and the communication protocols.

The results of the thesis are the system design and the demo source code. The implementation outcome proves the feasibility and the performance of the system design.

KEYWORDS:

Inverted pendulum, self-balancing robot, socket programming, protocol design

FOREWORD

This thesis would not be at the stage it is today if it was not for the assistance of the people I have around me. Firstly, I would like to thank my supervisor, Vesa Slotte. His knowledge, guidance and feedback throughout the thesis process have been invaluable for me. I would also like to thank Cisco Academy instructors at the Turku University of Applied Sciences for providing me with the resources and knowledge over the years to make me the engineer I am today, especially my CCNA teacher, Tero Virtanen, for all of his encouragement and advice. Finally, and most importantly, I would like to thank my mother for the her patience and support during my studies. .

Spring 2010, Turku

Vu Ba Tien Dung

TABLE OF CONTENTS

FIGURES

TABLES

NOTATION

θ (Theta)	Wheel's angle
τ (Tau)	Torque (Moment of force)
Ψ (Psi)	Body pitch angle (Tilt angle)
ω (Omega)	Angular velocity
K_d	Gain factor for differential component of PID controller
K_i	Gain factor for integral component of PID controller
K_p	Gain factor for proportional component of PID controller
API	Application Programming Interface
DSP	Digital Signal Processing
GUI	Graphic User Interface
IDE	Integrated Development Environment
I/O	Input/Output
J2ME	Java Platform, Micro Edition
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
NIO	Non-blocking Input/Output socket
PBL	Problem-based learning
PID	Proportional–integral–derivative controller
RTP	Real-time streaming protocol
RTPC	Real-time streaming control protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol

1 Introduction

1.1 Motivation

The importance of research into autonomous systems is due to their promises of changing the structure of society and creating a safe labour environment. In many people's opinions, an autonomous system such as a robot is simply an entertaining product. However, in the military, aeronautics and medicine, the implementation of advanced robotics has changed the concepts of defensive strategy, space exploration and surgery operation, respectively. Unfortunately, only a few literature papers about developing a remote controlled robot have been openly published.

Being inspired by the development and the benefits of remote-controlled robots, the author would like to develop a robot which can be remotely controlled through the Internet as his bachelor's thesis. The development project involves researching and developing a controlled dynamic robot with the support of local wireless technology and the Internet. Even though this thesis introduces only the very basic initial framework for developing a remote-controlled robot, the author hopes the readers can acquire enough knowledge about dynamic robots and robot communication design in order to recreate, extend, and improve the system by themselves.

The robot model which is used in the thesis project is the two-wheeled self-balancing robot. The author has selected that model not only because of its intriguing behaviour, but also because of its many advantages compared to the traditional three-wheel and four-wheel robots. These advantages will be discussed carefully in the following chapter.

The chassis of the robot is created using the LEGO NXT Mindstorms Development Kit because of its simplicity and flexibility for robot development in an educational environment.

1.2 Project Aim

The primary aim of this thesis project is to design the initial framework for creating a self-balancing two-wheeled robot, which can be controlled remotely by Internet users.

Moreover, the robot can be also controlled by local users with the aid of their computer or mobile phone. The project aim can be broken down into small objectives as follows :

- Implement a control system which is capable of balancing the robot. The control system also needs to be modified in order to offer some basic movements such as forward, backward and turn the robot around while maintaining the upright position.
- Design a scalable behaviour system into the robot. With the behaviour system, the robot can make its own decisions without human intervention.
- Develop a scalable communication system to enable communication between the robot and external human-controlled Bluetooth devices such as Bluetooth-enabled computer or mobile phone.
- Develop a scalable client server application to enable the remote users in the Internet monitor and control the robot.
- In detail document the architecture design and the implementation in order to reproduce it on different scales.

One of the most important objectives of the project is to make all the systems work together smoothly with the expected results. For example, the robot needs to maintain its balance while following the instructions from the external Bluetooth devices. Of course, it is still able to make its own decisions when it detects some collisions. This objective can be accomplished with the help of concurrency programming.

1.3 Author's contribution

In 1961, the first robot was created by General Motors Cooperation. From that time, it created an development explosion in autonomous systems over the world, especially in America and Japan.

However, the two-wheeled balancing robot had not been created until Dean Kamen introduced the Segway Personnel Transporter in 1999. With so many advantages over the traditional robot model, the two-wheeled balancing robot became a popular topic in academic autonomous systems' research. Therefore, many two-wheeled balancing robots have been developed.

Figure 1. Segway model i167 [1]

One of significant theoretical pieces of research in the two-wheeled balancing robots is NXTway-GS Model-Based Design by Yorihisa Yamamoto [2] which uses the LEGO Mindstorms Development Kit. In his work, Yorihisa Yamamoto has analyzed and documented in detail the equation of motions, physical dynamics, and also designed a state feedback control for the system. Unfortunately, his implementation was carried out with the MATLAB code generator which made it very hard for other researchers to approach the system. The main reason is because the MATLAB code generator produces the source code based on its knowledge about the equation of motions and the state equation. It does not approach the problem in a structured way that a human can comprehend. One of the successful implementations of the Yorihisa Yamamoto model is the Marvin NXT which has been developed by Johnny Rieper, Bent Bisballe Nyeng and Kasper Sohn in Spring 2009 [3].

In this thesis, the author will re-use the Model-based Controller design from Yorihisa Yamamoto to simplify his work, and also adapt the Marvin NXT source code into his own robot model. Java will be the primary language for this project due to its object-oriented capability and its vast amount of features.

The novel contribution of the author is designing and implementing the architecture in order to control the robot remotely from the Internet and the mobile phone. One of the possible reasons for this missing research is simply because most of researchers in autonomous system are not network application developers, and conversely, most of the researchers in network application development are not familiar with autonomous systems.

1.4 Thesis Overview

The thesis is divided into chapters. Each chapter will solve a specific problem in the project timeline. A brief overview of each chapter's content follows:

1.4.1 Chapter 1 - Introduction

This is the introductory chapter to the thesis, which provides the motivation of the author to his thesis topic, explains his aim for the project and how the thesis document is structured. In this chapter, the author also mentions the original works and what his novel contribution is.

1.4.2 Chapter 2 - Literature Review

This chapter covers some basic concepts about control systems and robot programming. Through this chapter, the readers can enhance their knowledge of autonomous systems and network protocols in order to fully understand the following chapters.

1.4.3 Chapter 3 - Architecture design of the project

This chapter addresses the solutions to the project problems in general . The necessary technologies, systems (robot, server, client, mobile phone) and sub-systems (control system, protocols) are explained in order to help the readers to have a good understanding of the system workflow.

1.4.4 Chapter 4 - Detailed design of the project

This chapter covers the design of each sub-system in detail. The control system of the self-balancing robot which has been developed by Yorihisa Yamamoto is discussed. The communication protocol designs between the robot-server, the robot-mobile phone and the server-client are explained. Real-time streaming protocol (RTP) is also mentioned as it is needed for remote monitoring.

1.4.5 Chapter 5 - Implementation and integration of the project

We have already discussed about the detailed design of the control system and each communication protocol. However, integrating all these sub-systems (control system, protocols) into specific systems such as robot, server, client and mobile phone is not an easy task. In this chapter, the design of each system's architecture is presented and discussed. Some interesting experiences in the implementation phase are also discussed in this chapter. The audience will gain knowledge in robot development and network application programming throughout this chapter.

1.4.6 Chapter 6 and 7 - Summary and Recommendation

This chapter provides a summary of all the information that has been presented throughout the thesis concluding with the achievements as well as future improvement that could be done.

2 Literature Review

The literature review chapter simply provides the readers with a review of some concepts they will regularly come across throughout the thesis. The chapter also provides basic knowledge to the readers who are not familiar with robot development, network programming and protocol design fields. Going through the concepts in this chapter will help the readers understand the following chapters easily.

2.1 Modern control system

The Control system or feedback system is the processing unit of a whole dynamic and unstable system.

In order to understand the concept of the control system, we need to go through a simple example, the thermostat system. The thermostat system is the most common control system people encounter. For instance, a chef is cooking a sauce in a big pan and needs to keep the pan at a certain temperature. He or she needs to use a thermostat to observe the temperature and respond to the change of the temperature by increasing or decreasing the heat power.

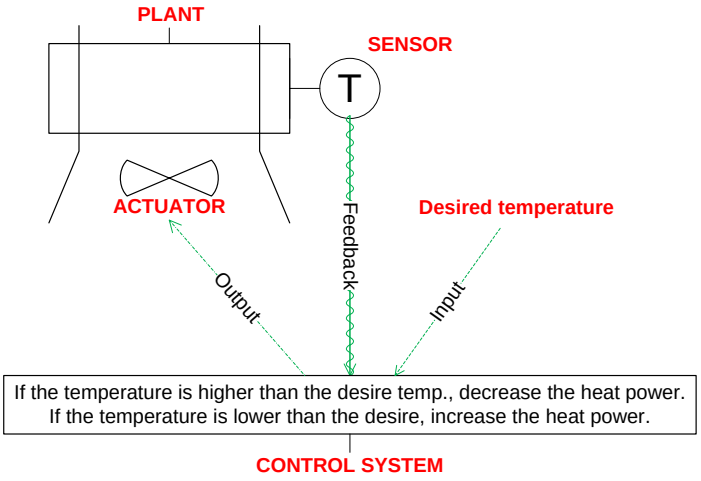


Figure 2. Thermostat control system

In Figure 2, the thermostat control system is described. The **control system** attempts to ensure the *desired temperature,* which is called the **input** value for the system being controlled, generally referred as the **plant**. The control system receives *feedback* from the **sensor system** which is the thermostat in this case. The control system makes its own calculation and reflects its **output** value to the power switch or the heating system which is known as the **actuator**. Of course, increasing or decreasing the heat source (actuator) can make the temperature change, and again the control system will receive a new feedback and make a new decision. That is the reason we call this feedback system a closed-loop feedback system.

In the history of control engineering, the first significant work in automatic control was awarded to James Watt for his speed control system of the steam engine. Other significant contributors in the early days of control engineering theory were Nyquist, Minorsky, Hazen and many others [4].

The three most popular feedback systems that are usually used as the introductory examples in discussions of the control systems are the thermostat system, the airplane guidance and the cart-pole. The thermostat system is used as the first example due to its simplest and commonest most people meet. The irplane guidance is perhaps the most complex one. Finally, the cart-pole is the most intriguing and interesting experiment in control system theory [5].

2.2 Two-wheeled balancing robot

The two-wheeled balancing robot is the industrial name of the cart-pole experiment in control systems or inverted pendulum problems in modern physics. In that system, a pole is free to move within one axis. On top of the pole is a cart that also moves in the same axis. Because of the weight of cart, the pole will obviously fall down if the pole does not move towards an appropriate direction.

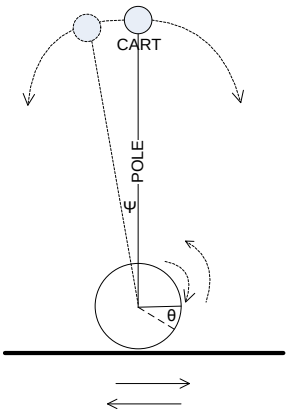


Figure 3. Cart-pole experiment

In order to maintain the cart-pole (*plant*) balance, a control system is needed. The control system collects the information about the different angle of the cart-pole compared to the equilibrium point ($\Psi=0^o$) and also the different angle of the wheel (*actuator*) compared to its equilibrium point ($\theta=0^o$). The control system can collect those data through the *sensor system* (gyroscope sensor, accelerometer and wheel encoder). The gyro sensor and accelerometer are used to measure the angle of cart-pole which is called body pitch angle or tilt angle. The wheel encoder is used to measure the angle of the wheel. Generally, the system can be described as a closed-loop control in Figure 4.

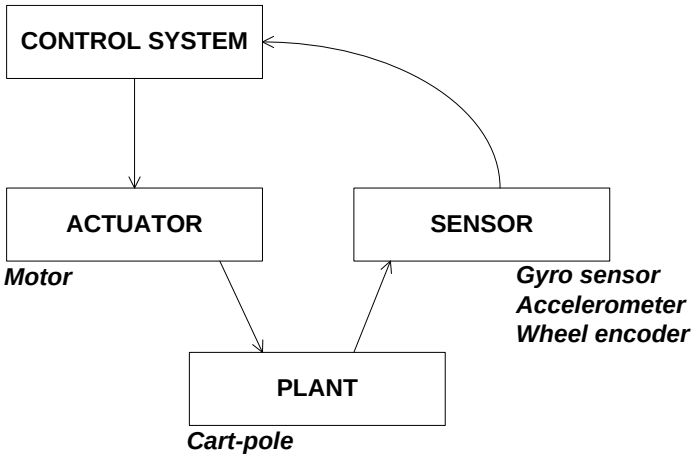


Figure 4. Cart-pole control system model

The first robot in which this cart-pole control system was applied was Legway, which was built by Steve Hassenplug on 14th October 2002 using the LEGO Mindstorms Education Kit. Since then, there have been many studies and projects on the two-wheeled self-balancing robot which used LEGO Mindstorms Education Kit or were constructed from the scratch. One of the most famous two-wheeled self-balancing robot projects which was started from the scratch was David P. Anderson‘s nBot in 2003.

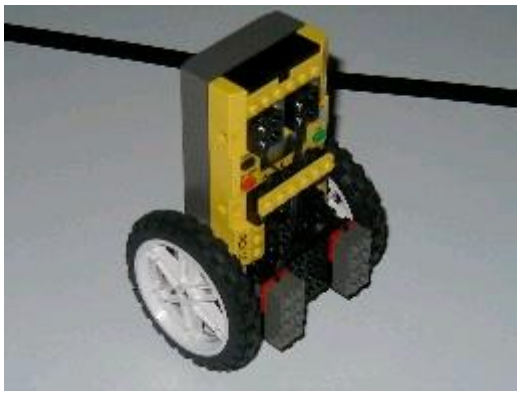


Figure 5. Steve Hassenplug's Legway (2002) [6]

Figure 6. David Anderson's nBot [7]

There were several reasons why the two-wheeled self-balancing robot became popular over the three and four-wheeled robots. Most of its advantages can be drawn from the article “SOHO security with mini self-balancing robot” written by Albert Ko, H.Y.K. Lau, T.L. Lau [8]. The advantages are listed below with the visualization of the robot copied from the above article:

- Small footprint: The physical area the self-balancing robot uses is very small. It allows the robot have a very small footprint and navigate among tight areas easily.
- Zero Turn Circle: The two-wheeled self-balancing robot is able to turn on the spot. Therefore, improving the capability of navigating through tight areas.
- Center of gravity: Within the same design as the three and four-wheeled robot, the center of gravity is shifted above the wheel area all the times. This allows the robot handle disturbances over the slope better than the traditional robot.

Figure 7. Small footprint [8]

Figure 8. Zero turn circle [8]

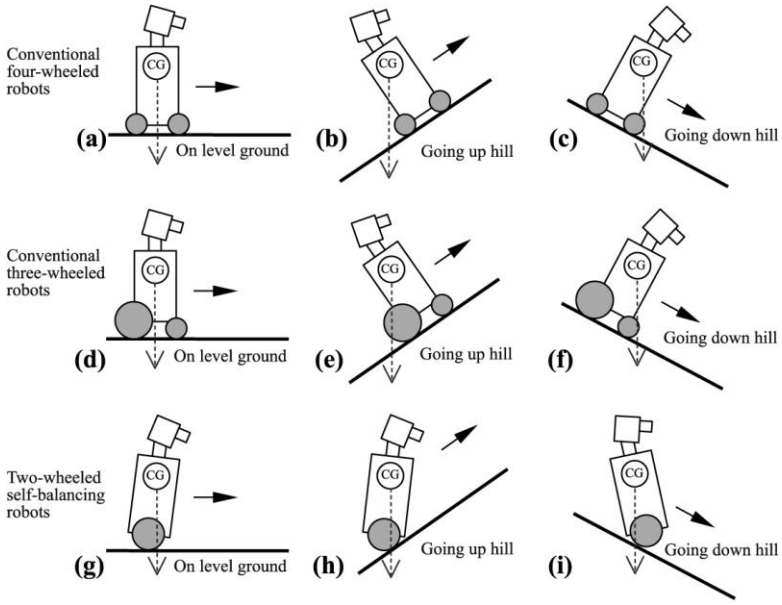


Figure 9. The center of gravity of the two, three, four wheeled robots over the slope [8]

2.3 Robot's behaviour-based architecture

Most robots are built with a microcontroller such as Atmel AVR, ARM or PIC as its brain. For instance, the LEGO NXT Mindstorms uses the ARM7 32-bit microcontroller. However, when most people start programming their robots, they approach the implementation task with some high level programming language such C, Java or Visual Basic. They think of the program workflow as a series of if-else, in which they check the states of the sensors in series and give a certain output. Finally, the program code becomes very complicated and very difficult to improve. Figure 10 visualizes the normal way of robot programming.

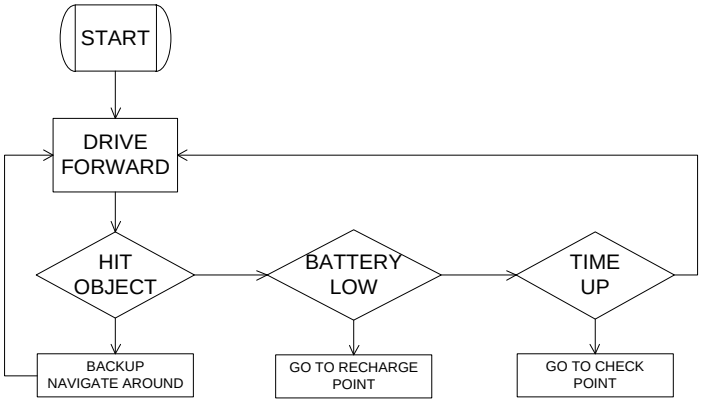


Figure 10. Structured programming visualized

In contrast, the robot is recommended to be programmed according to the behaviour-based architecture. In the behaviour-based architecture, there is an arbitration system which controls all the behaviours. When a behaviour meets the activation conditions, it can then trigger the arbitration system and take control of the system. Finally, the program code becomes interrupt-oriented and easier to understand. Figure 11 visualizes the behaviour-based architecture version of the program in Figure 10.

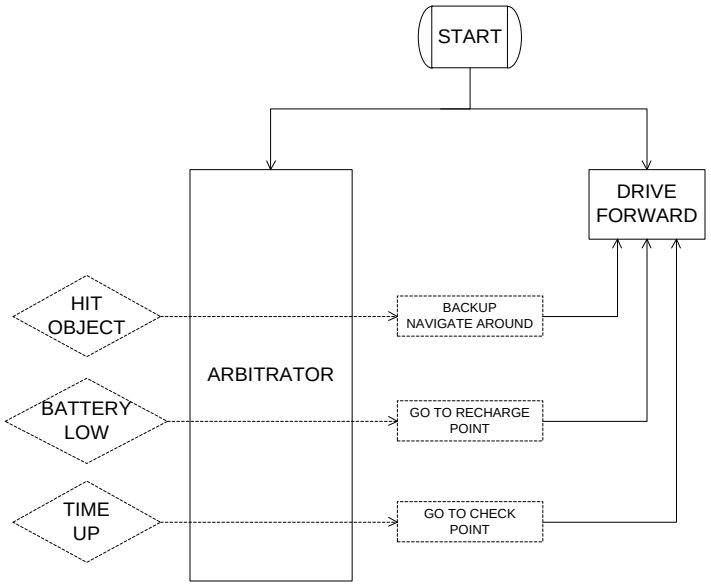


Figure 11. Higher priority behaviour suppresses other behaviours

2.4 LEGO NXT Mindstorms Programming

2.4.1 What is LEGO NXT Mindstorms?

Lego NXT Mindstorms is a programmable robotics kit released by Lego Cooperation in July 2006. The advantage of this development kit is that developers, researchers or students can easily develop and test their projects without having any tools or special skills of robotics building. The only skill they need is assembling the LEGO kit [9].

The basic intelligence is provided by the NXT Intelligent Brick. The NXT Brick is constructed with a 48Mhz, ARM7-TDMI 32-bit microcontroller, and a 4Mhz, AVR 8-bit microcontroller, a Bluetooth wireless communication (v2.1 compliant), an USB 2.0 port, 4 sensory input ports, 3 output servo motor ports, a 100x64-pixel LCD screen and a loudspeaker [10].

As a stand-alone system with a wide variety of sensors, the Mindstorms NXT is a great tool for learning about robotics programming. As a result of that, Turku University of Applied Sciences offers some problem-based learning (PBL) courses using the LEGO NXT Mindstorms Development Kit.

2.4.2 What is leJOS NXJ?

After publishing the LEGO NXT Mindstorms Kit, Lego released the firmware for the NXT Intelligent Brick as an Open source. Of course, that firmware only executes the program written in NXT-G language, which is an interactive programming language created by LEGO to provide some basic control of the LEGO NXT Kit.

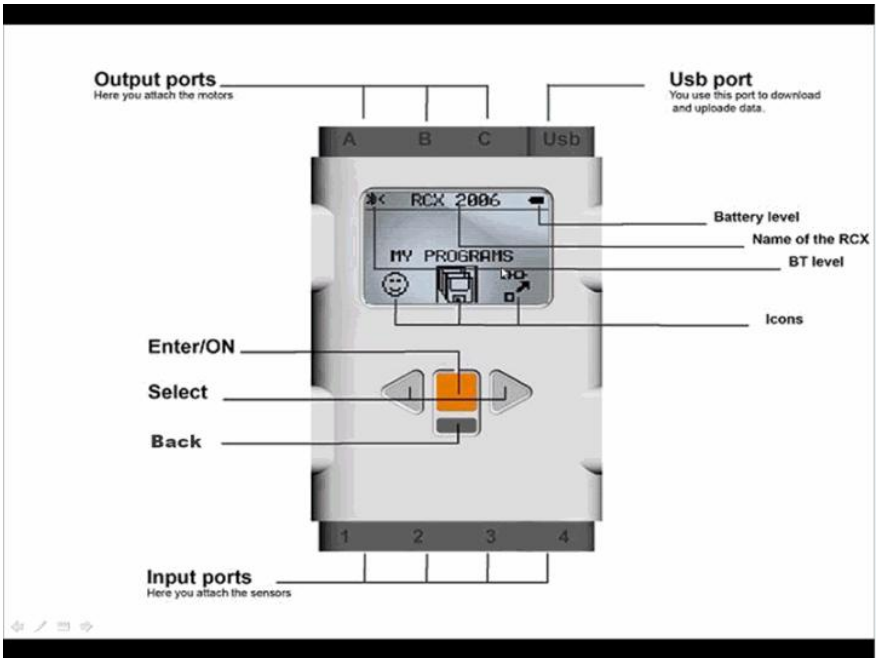


Figure 12. The NXT Intelligent Brick [11]

However, to extend the abilities of the NXT Brick to the maximum, many software teams have customized that firmware to enable other high-level programming languages such as C, Java, Python, MATLAB, Ruby, Ada and Lua run in the NXT Brick. leJOS NXJ is a customized firmware which is based on Java Virtual Machine developed by the leJOS team. With this customized firmware, the program which is written by Java language can run in addition to the NXT Brick.

2.4.3 Advantages of leJOS NXJ

There are many advantages of using the Java programming language on the LEGO system:

- Object-oriented programming: the program is constructed with the concepts of object and method which make the program natural and manageable.
- Java is an opensource language. It means that there will be a large number of contributors who improve and develop leJOS firmware.
- Easy implementation with some professional Integrated Development Environment (IDE) such as NetBeans or Eclipse.

- Concurrency programming is a primitive built-in in the language. It makes the language a perfect choice for building communication robot and complicated behaviour-based architecture.
- Bluetooth is fully supported in the language as the language has a long history of running in the mobile phone system as a Mobile Information Device applet (MIDlet).

2.4.4 Disadvantage of leJOS NXJ

Even though leJOS NXJ has numerous advantages, it still has a significant disadvantage compared to other firmware. That disadvantage is the high amount of memory utilization. This is an avoidable problem of the Java language because the program written by Java is only the bytecode program, not the machine language. It needs a Java Virtual Machine (JVM) to run. Therefore, besides the desired program, the JVM also runs and uses up a certain amount of memory.

2.5 Protocol design

A protocol is simply a specific language which two entities such as human or machine use to communicate with each other. In terms of telecommunications, a protocol is a set of standard rules for data representation, signaling, authentication and error control in order to send information (data) over a communication channel [12]. Designing new protocols promises offering new services and improving the current services. New protocols are being designed and proposed every month while old protocols are being revised every day. In order to make a new protocol compatible with the old system and old protocol, IEEE and IETF have chosen a framework for network communication protocols – the TCP/IP model, which was designed by the United States Department of Defense (DoD).

In this thesis, the author follows the recommended TCP/IP model closely while designing the protocols in order to make compatible and usable protocols.

2.5.1 TCP/IP model [13]

The TCP/IP model is a framework for network protocols which was created in 1970s by the U.S.A. DoD. It divides a communication session into five layers include physical layer, data link layer, networking layer, transport layer and application layer.

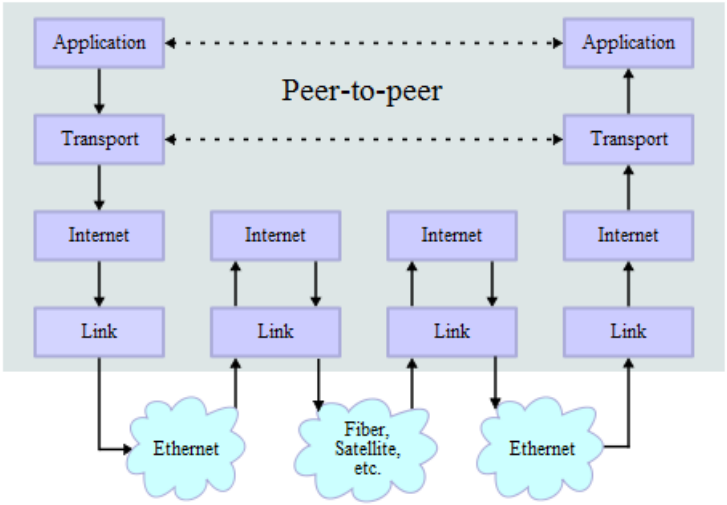


Figure 13. TCP/IP Stack Connections [14]

Figure 13 illustrates the layering model used with the Internet protocols.

- The physical layer at the bottom of the model specifies details of the transmission channel and the associated hardware. All specifications related to electrical properties, radio frequencies and signaling.
- The data link layer specifies details of the communication between the software directly running on top of the hardware (network interface driver). In this layer, the physical address of the system is used to distinguish the system physically.
- The network layer specifies communication between two systems across the Internet. This layer addresses the structure and format of the packets, how to address the systems and how to reach a certain system in the Internet. In this layer, the logical address ensures the uniqueness of the system. The Internet Protocol (IP) address is the most popular logical address being used on the Internet nowadays.
- The transport layer provides the communication from a network application on one computer to another computer. This layer provides the reliability of the

connection which is known as connection-oriented and connectionless protocol. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the two main protocols in this layer. TCP provides a reliable, connection-oriented connection through its three-way handshakes, acknowledgment and flow control mechanisms, while UDP provides a unreliable and connectionless connection. In this layer, the port number, which the application is using, helps the system distinguish the network applications.

- The application layer specifies details of the format and meaning of the messages that applications can exchange as well as the next expected behaviour of the protocol.

The significant advantage of the layering model is that the layers work independently of others. It helps the protocol designers focus only on one aspect of the communication at a given time.

In this thesis, the author focuses only on the transport layer and application layer. And thanks to the layering model, the implementation in the transport and application layer can work efficiently with other layers without any modifications.

2.6 Socket programming

Socket programming is a separate part of protocol designing. It is a way to implement the application and transport layer into a specific system. Therefore, it is hard to define which layer a socket belongs to.

A socket is one endpoint of a two-way communication link. Endpoint is defined as a combination of a port number and an IP address. Because of the flexibility of this definition, one computer can create two sockets itself and connect them. The only requirement is that those sockets need to be bound to two different port numbers. The sequence of socket connections will be described in the following paragraphs.

Generally, a server runs on a specific system and has a socket that is bound to a specific IP address and a specific port number. The server will start to wait for connection from a client. On the client side, the client makes a connection socket request to the server. If everything goes well, the server accepts the client connection and starts the communication through that socket.

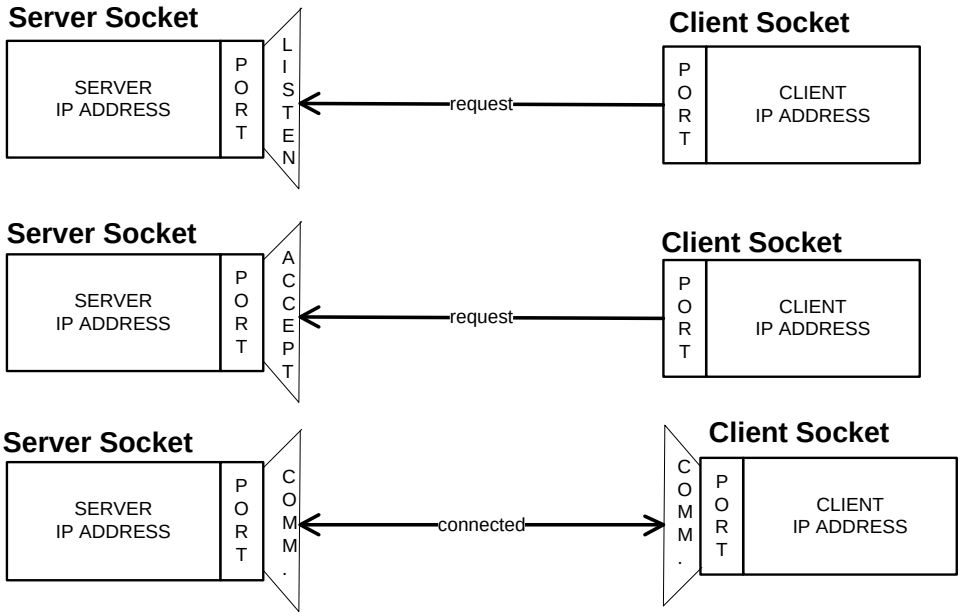


Figure 14. One server - one client socket connection

In Figure 14, the socket request and response processes are illustrated. The disadvantage of this model is that the server can only respond to one client request. After that, the server socket is unavailable and cannot respond to any other requests.

In Figure 15, the server accepts the connection. Upon the acceptance, the server acquires a new socket which is bound to the same local port and uses it to connect with the same client's endpoint. The ability to create multiple sockets which can bind to the same port enables the server to respond to multiple clients' requests concurrently.

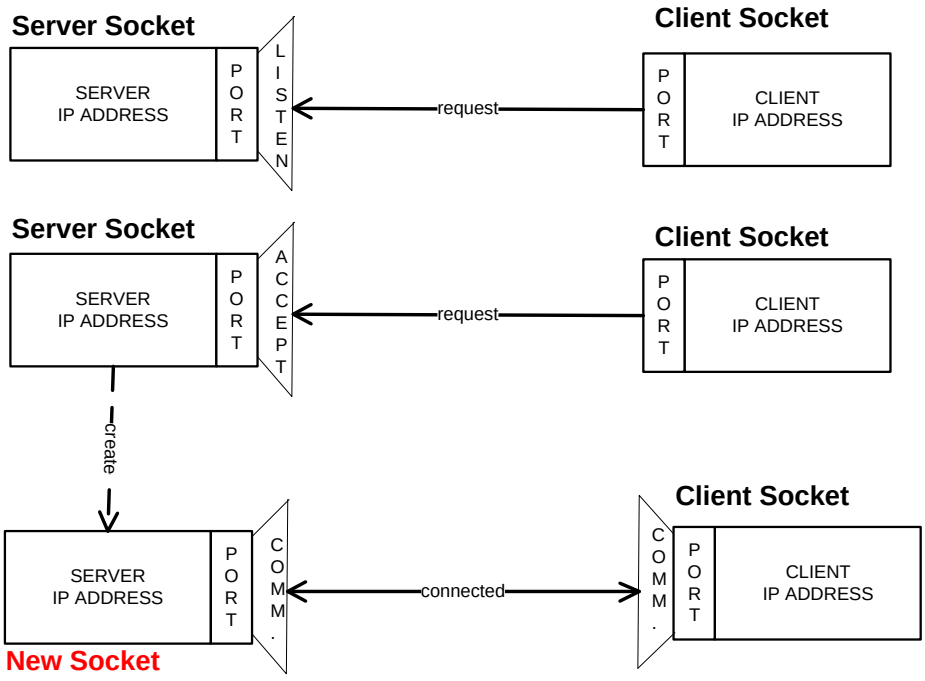


Figure 15. One server - multiple clients socket connection

After the connection is made, the sockets will supply the input and output stream (I/O) to the application. Through the I/O, the application then can send and receive messages between two systems.

Figure 16 summarizes the sequence of a socket connection in the client-server application model. In this model, upon the acceptance, the server socket creates another socket that is bound to the same port to communicate with the client socket. This ability enables the server to respond to multiple clients' requests. However, in order to handle multiple clients' requests, the server needs to implement concurrency programming. Concurrency programming will be discussed in the next section.

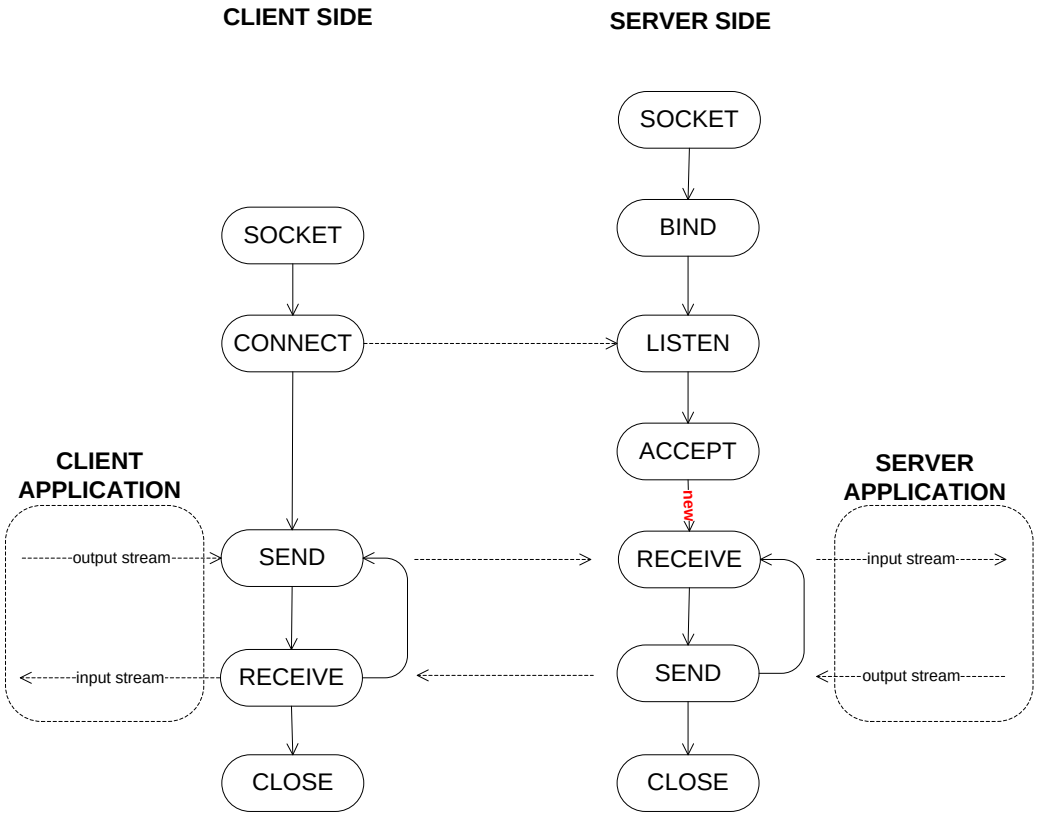


Figure 16. The sequence of socket functions and the data stream

2.7 Concurrency programming (Multi-threading programming)

The system with only one processing unit is supposed to execute one command at a given time. Only the systems that have multiple processors can truly execute multiple instructions concurrently.

It would be easy if the system focuses its attention on performing only one action at a time. However, in some situations, performing a great variety of operations in parallel is needed. That concept is called concurrency. Making a program that can handle multiple tasks at the same time is called *concurrency programming*, or sometimes *multi-threading programming*.

The Java programming language makes concurrency available and primitive through its language and Application Programmable Interfaces (APIs). Because leJOS NXJ runs on top of Java Virtual Machine, the robot program which is written with leJOS APIs can execute multiple tasks at the same time. Figure 17 gives an example of concurrency architecture through illustrating the concurrency architecture of the two-wheel self-balancing robot. The system needs to carry out three tasks at the same

time. Every task is carried out by one program thread. Therefore, the three threads work in parallel include: *balance controller*, *communication manager* and *arbitrator* (*collision avoid* and *movement control* sub-threads).

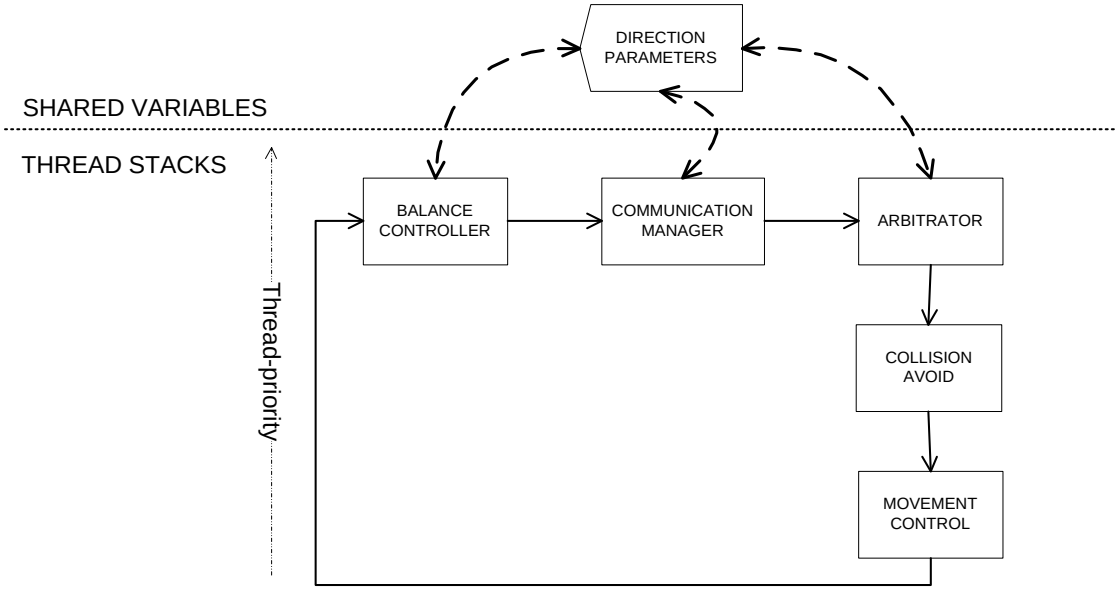


Figure 17. Concurrency architecture of the robot

In Figure 17, the problem occurs when the *balance controller, communication manager* and *arbitrator* threads share the same object. Indeterminate results may occur unless access to the shared object is managed properly.

For example, after the *communication manager* has received a command from the server, it updates the *direction parameters*. While the *communication manager* is updating the parameters list, the running time exceeds and it is suppressed. The *arbitrator* thread takes control and runs its *movement control* sub-thread. The *movement control* reads the parameters list which has not finished updating and executes it. The *movement control* can turn left, for instance, but 180^o instead of 30^o as it should be.

This problem is known as thread synchronization. The traditional solution is that the programmer needs to create a semaphore or a mutex for the object's access. When a thread has exclusive access a shared object, the object will be locked and the other threads need to wait until that thread unlocks the object. This mutual exclusion among the threads can be accomplished by using the negotiation algorithm of the mutex and the semaphore data structure.

Implementing the mutex is a time-consuming task. However, the Java programming language supports concurrency programming primitively. With Java APIs, creating a synchronized object without any knowledge about locking and monitoring object states is an approachable objective.

3 Architecture design

The architecture design in Figure 18 is one of possible solutions the author used to solve the thesis problem.

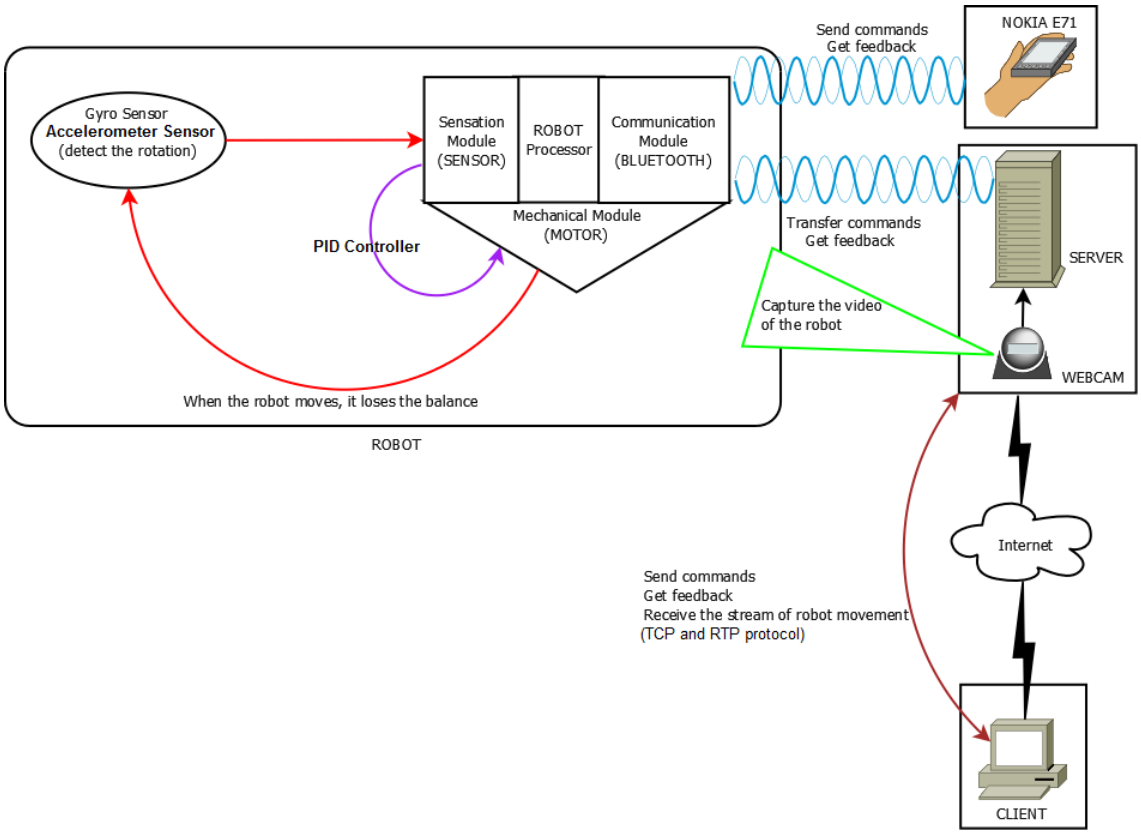


Figure 18. Architecture design

The robot is built using a LEGO NXT Mindstorms Development Kit. There are four important modules of the robot: the processor, the communication module, the sensory module and mechanical module (actuator).

- The communication module uses Bluetooth connection. The Bluetooth connection is a short-range local wireless connection. The range of the

connection is around 20 meters. With Java API, the Bluetooth connection is very easy to be established and transfer the data.

- The sensory module is constructed with an one-axis gyroscope sensor and an accelerometer sensor. Both of the sensors work together to give the control system accurate values of the tilt angle and the tilt velocity. Individually, the gyroscope sensor is used to measure the velocity in which the robot loses its balancewhen the accelerometer tells exactly the tilt angle compared to the equilibrium point.

- The mechanical module is constructed by two servo motors. Every servo motor is attached with a wheel encoder inside. The wheel encoder tells how much the motor has turned and how fast the motor is turning.

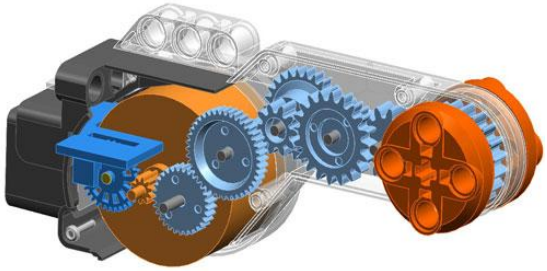


Figure 19. The LEGO NXT's servo motor [15]

The mobile phone uses Symbian S60 3rd OS. The phone also supports Java MIDP 2.0. With the support from J2ME (Java 2 Mobile Edition), the Bluetooth connection between the phone and the robot is easily implemented. The command will be sent from the phone to the robot and also the feedback will be returned. A simple and connectionless protocol should be implemented here. The main reason is because of the wireless communication's bandwidth, a small amount of traffic will be more efficient. The robot-mobile phone protocol will be discussed in the next chapter.

The server is equipped with a web camera and connected to the Internet. The server also has a Bluetooth dongle to make the Bluetooth connection with the robot.

- The protocol between the server and the robot has the same functions as the mobile and the robot. Therefore, a scalable and re-usable protocol should be designed.

- In order to receive the command from the client, a server-client protocol needs to be designed. A reliable protocol should be designed for this purpose. The main reason is because of the high possibility of the packet loss over the Internet. TCP is chosen as the transport protocol for this protocol.

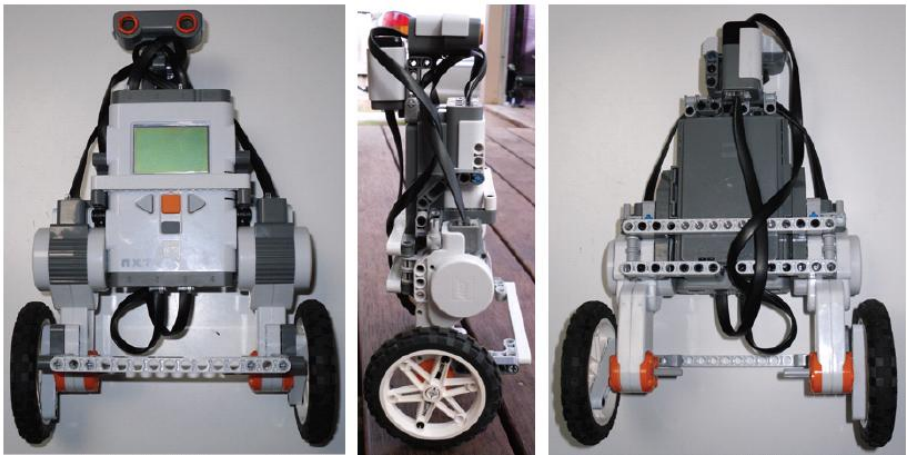
- The server needs to capture the raw data from the webcam and streams that information to the client, so that the client can monitor the behaviours of the robot remotely in order to give the correct instructions. Real-time Transport Protocol (RTP) will be used as the transport protocol and Joint Photographic Experts Group (JPEG) will be used as the streaming format. The reason of this implementation will be discussed in the next chapter.

The client application will receive the stream from the server to know the robot's position and navigate the robot remotely. To accomplish this requirement, the client applications need to make two connections to the server. The RTP connection is for streaming media and the TCP connection is for management information.

4 Detailed system design

4.1 Developing a self-balancing robot

Figure 20 is the image of the robot after construction. This construction follows the building instructions of Yorihisa Yamamoto. However, the accelerometer is added to make the tilt angle measurement easier.



a) front view b) side view c) back view

Figure 20. Two-wheel self-balancing robot built according to Yorihisa Yamamoto instructions [16]

4.1.1 Designing the control system

The physical structure of the robot makes it impossible to keep it in the upright position without an appropriate control system. The control system detects the degree in which the tilt changes and corrects the position of the robot by applying a suitable power to the motors.

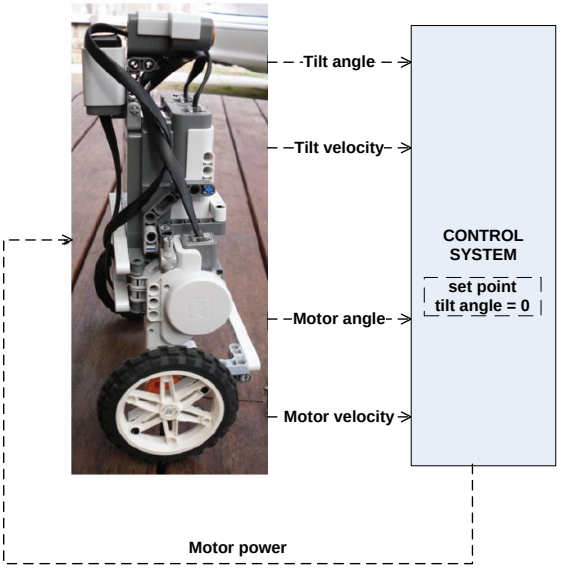


Figure 21. Inputs and outputs of the balance controller

One of the most commonly used close-loop feedback control system, the Proportional-Integral-Derivative (PID) controller will be used for solving this problem. The main reason is because the PID controller is suitable for this problem.

The output of the control system is the power of the motor. The value of the motor power is in pulse mode and contains errors. If the controller just simply sends this value to the motor, the robot will lose its balancing as soon as possible because of the lack of accuracy.

The PID controller contains three parts: proportion, integration, and derivation.

- After the controller calculates the desired value of motor power, the value can be smaller or larger than the accurate desired value and cannot be applied directly to the motor.

- The proportional component of the PID controller multiplies the value by a factor which is called the *proportional gain (K_p)*, so that the output is driven toward the desired value as quickly as possible. The proportional factor can be larger or smaller than one (1) depending on the accuracy of the calculation formula which is usually called the *state equation*.

- However, one problem with proportional component is that it always creates an offset from the desired value. The integral component is in place to integrate the error over time. It simply adds the sum of the errors and multiplies with a factor, which is called *integral constant (K_i)*. The integral factor is usually very small (less than 0.5). One effect of the integral component is causing the output value to oscillate around the set point until the error reduced to zero [17].

- The integral component oscillates the value and makes it take a longer time to settle to the desired value (error = 0). The differential component multiplies the derivative of the error by a gain factor *(K_d)*. The objective is to create a damping effect on the PI value.

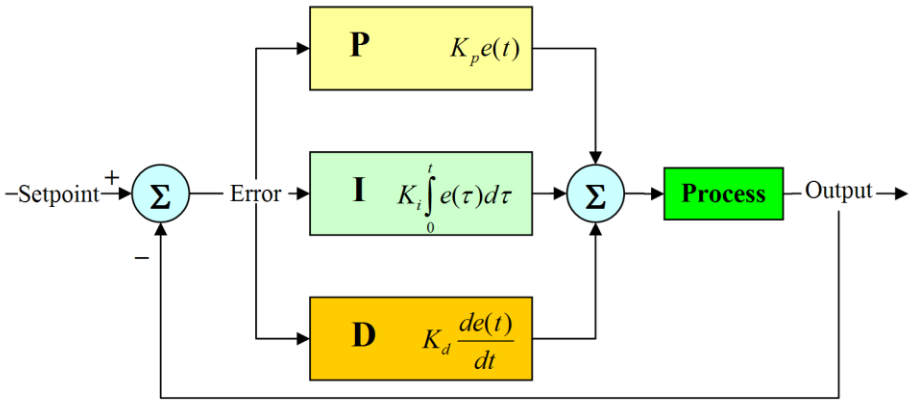


Figure 22. The PID controller [18]

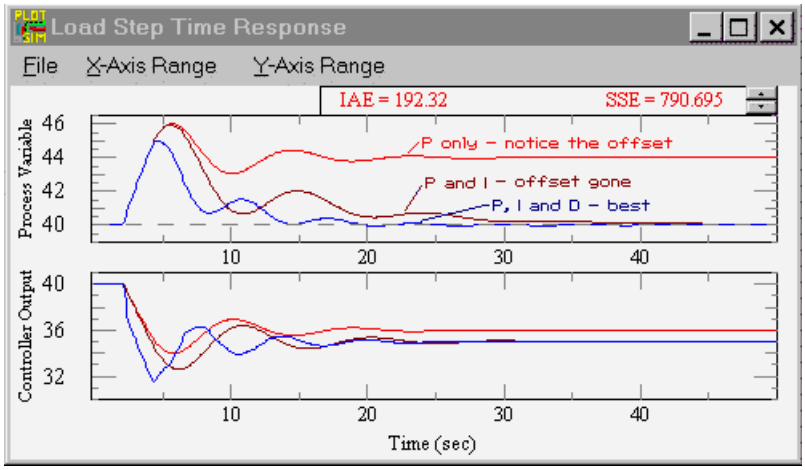


Figure 23. PID Tuning Plot [19]

Figure 23 illustrates the benefits of the PID controller.

- From the state equation, we obtain the output value 46, which has a large error with the desired value 40.
- After multiplying with the proportional gain factor (which is smaller than 1), the output value is decreased and remains near the desired value.
- The integral component oscillates the output value and reduces the error to zero.
- To settle the desired value quickly, the derivation gain factor is applied. It creates a damping effect on the PI value and enables faster convergence.

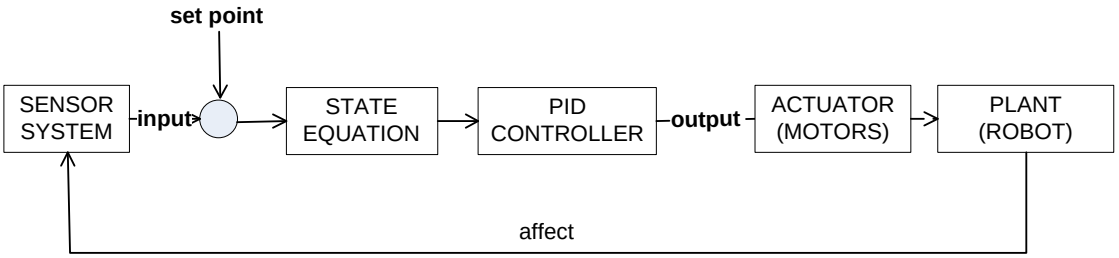


Figure 24. Control system for the two-wheeled self-balancing robot

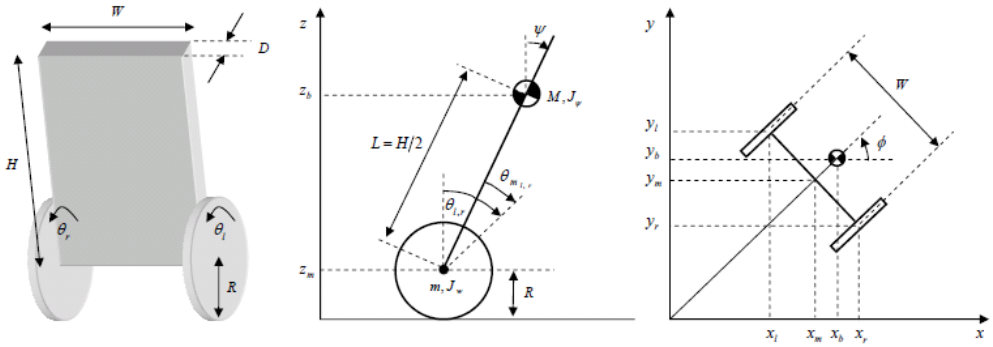
Figure 24 presents the overview of the control system for the two-wheeled self-balancing robot. The selection of the PID controller gain factors is a difficult task and requires the knowledge about Nyquist's theorem in feedback system design. The author also did not calculate the gain factors by himself, but simply choose the appropriate values through trial and error.

The gain factors which will be used in the thesis's implementation are: K_p=1.25 , K_i=0.25 , and K_d=0.10.

4.1.2 The plant model analyzed by Yorihisa Yamamoto [2]

Understanding the motion equations and retrieving the state-space equations of the two-wheel self-balancing robot are difficult tasks.

Fortunately, in the NXTway-GS documentation, Yorihisa Yamamoto provided a detailed analysis of the plant model. All the analysis of the two-wheel self-balancing robot motion equations and state-space equations are cited in the NXTway-GS documentation.



a) 3D view of the robot **b) side view** **c) plane view**

Ψ : body pitch angle $\Theta_{l,r}$: wheel angle $\Theta_{ml,r}$: DC motor angle

Figure 25. Side view and plane view of Two-wheel Self-balancing robot [2]

Table 1 presents the physical parameters of NXTway-GS.

Table 1. State-equation parameters for the plant model

Parameter (value)	Unit	Description
g = 9.81	*[m/sec^2]*	Gravity acceleration
m = 0.03	*[kg]*	Wheel weight
R = 0.04	*[m]*	Wheel radius
$J_W = mR^2/2$	*[kgm^2]*	Wheel inertia moment
M = 0.6	*[kg]*	Body weight
W = 0.14	*[m]*	Body width
D = 0.04	*[m]*	Body depth
H = 0.144	*[m]*	Body height
L = H/2	*[m]*	Distance of the center of mass from the wheel axle
$J_\Psi = M^2/3$	*[kgm^2]*	Body pitch inertia moment
$J_\theta = M(W^2 + D^2)/12$	*[kgm^2]*	Body yaw inertia moment
$J_m = 1x10^{-5}$	*[kgm^2]*	DC motor inertia moment
$R_m = 6.69$	*[Ω]*	DC motor resistance
$K_b = 0.468$	*[V sec/rad]*	DC motor back EMF constant
$K_t = 0.317$	*[Nm/A]*	DC motor torque constant
n = 1		Gear ratio
$f_m = 0.0022$		Friction coefficient between body and DC motor
$f_W = 0$		Friction coefficient between wheel and motor

The values of m, R, M, W, D, H are measured according to the author's robot model. The formulas of J_W, L, J_Ψ, J_θ are chosen from the appropriate physics inertia moment formulas. The values of R_m, K_b, K_t are collected from Ryo Watanabe's LEGO page [20]. The other values such as J_m, n, f_m and f_W are assumed according to Yorihisa Yamamoto's documentation.

In this thesis, the plant model will not be discussed any further. The main reason is that the plant model analysis of Yorihisa Yamamoto is so complicated, the readers need to have a strong background of mathematics and modern mechanic physics in order to understand and check the calculations. The background knowledge is beyond the scope of this thesis.

For simplifying the thesis, the author will only explain the idea behind the state equation calculations and the final result which will be applied to the robot. In case the reader

wants to clearly understand, all the detailed equations and calculations can be found in the References [2].

4.1.3 State Equation [2]

The main purpose of the state equation is to find out the formula which shows the relationship between the output motor power and the tilt angle, the tilt velocity, the motor angle and the motor velocity of the robot.

Until now, the author always mentions that we need to collect four items of information about the tilt angle, the tilt velocity, the motor angle and the motor velocity, but the reason for this selection has not been explained yet. Figure 26 explains the reason for this selection.

Let's assume that the robot is falling down towards the positive direction as described in Figure 26.

- The direction is considered as the positive angular direction of the system, so that $\Psi > 0^o$. The angular velocity ω_Ψ ($\ddot{\psi}$) is also larger than 0^o/s.
- Naturally, because the robot falls forward, the wheels will be turned backward. If the wheels do not have the original velocity, they will turn in the opposite direction to the robot's plant, we have $\theta_\Psi < 0^o$/s.
- Because of the rotation around the horizontal axle, the wheels create an angular velocity $\omega_{\theta\Psi}$. Even there is no net force applied to the system, a moment of force (torque) will be created, $\tau_{\theta\Psi}$, and be applied onto the horizontal axle.
- Due to the torque, the robot cannot maintain the static equilibrium and of course, falls down.

  > For an object to be in static equilibrium, not only must the sum of the forces be zero, but also the sum of the torques (moments) about any point [21].

- In order to make the system be in static equilibrium, we need to apply a certain power to the motors to make the wheels turn in the same direction with the tilt velocity, $\omega_{\theta l}$ ($\ddot{\theta}_l$) and $\omega_{\theta r}$ ($\ddot{\theta}_r$). The rotation of the wheels will create a torque τ_θ, which is in the opposite direction of $\tau_{\theta\Psi}$. If these two torques have the same magnitudes, the system will be in a temporary equilibrium.

$T_\theta = T_{\theta\psi}$

- At this moment, the problem is that the wheels have original velocity. This has already created a torque to the system. So , the next calculation has to take the original velocity into account.

$T_\theta + T_{\theta'} = T_{\theta\psi}$

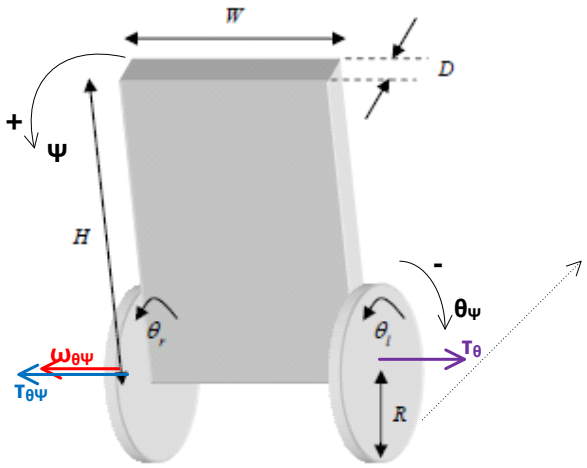


Figure 26. Angular velocity and Torque in the two-wheel self-balancing robot

According to the above explanation, the tilt angle, the tilt velocity, the wheel angle and the wheel velocity are the necessary parameters for the motor power calculation. The desired state equation should be in the below format:

$$a\psi + b\ddot{\psi} + c\theta + d\ddot{\theta} = F_{\theta'}$$

where $\theta = (\theta_l + \theta_r)/2$ and $\ddot{\theta} = (\ddot{\theta}_l + \ddot{\theta}_r)/2$

As we can see, the easiest approach is through the energy calculation, because most of the parameters are related to force.

In the NXT-GSway's documentation, even Yorihisa Yamamoto did not explain the idea behind those parameters, he finally proved that those four parameters are needed for the state equation. His final result is presented as the below equation. In the implementation, the author will re-use this result without making any changes.

$$34.1896 \times \psi + 2.8141 \times \ddot{\psi} + 0.8351 \times \theta + 1.0995 \times \ddot{\theta} = F_{\theta'}$$

4.2 Robot and Server communication

The protocol design has two parts: the sequence diagram of the protocol and the explanation with the communication message structures.

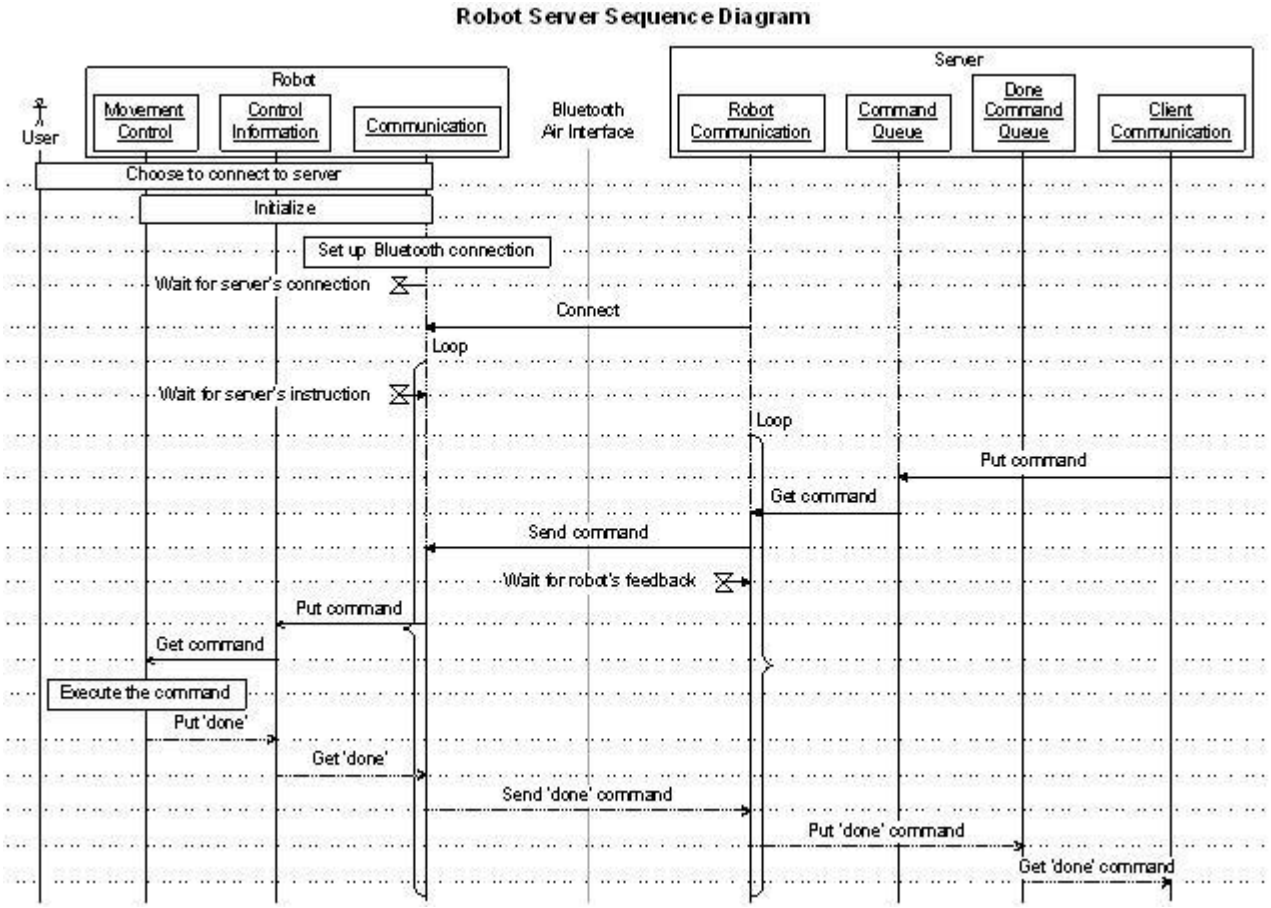


Figure 27. Robot-Server Communication Sequence Diagram

Figure 27 describes the sequence of the communication between the robot and the server.

- After the user starts the robot in ‘server mode’, the robot will start waiting for the connection from the server. The connection from the server will start the communication process.

- First of all, the robot will wait for the instruction message from the server. The server retrieves the command from the command queue, compiles the command into the instruction message and then sends it to the robot. The message’s structure is very simple. Figure 28 presents the structure of the message. The author chooses a simple message structure in order to limit the power consumption of the transferring process.

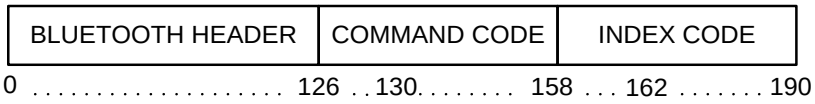


Figure 28. Robot-Server communication message structure

- The Bluetooth packet header is 126 bits long [22].
- The command code and index code are 4 bytes long, or 32 bits long.
- However, at this moment, the author only uses 3 bits for command code and 6 bits for index code. The other bits will be reserved for the future usage.

- After receiving the instruction message from the server, the robot will extract and read the command code, which contains the command from the server. The index code is simply a tracking code the server uses to track the command from multiple users. Table 2 presents the command code.

Table 2. The command code in the Robot-Server message

Command code	Description
0	Move forward
1	Move backward
2	Turn left
3	Turn right
4	Stop

- The communication thread will instruct the movement control thread to execute the command by using the synchronized *control information* object. After the movement control thread has finished its job which is specified in the control information object, it will notify with the communication thread also by set the flag ‘done’ in the synchronized control information object.

- After the flag ‘done’ is set, the communication thread will simply return the message it received from the server before to the server. This message has exactly the same content as the message it received from the server. At that time, the server’s communication thread is waiting for the return message. When it receives the return message with the same content it will consider the successful execution. If not, the server will only keep on waiting.

- After the command has been carried out , the robot will revert to the waiting mode. It will wait for another command while the server will try to send another command. The process will be looped forever until one system disconnects.

This protocol design has its own advantage and disadvantage. Its simplicity can save a certain amount of power for the robot. However, the lack of acknowledgement in transfering the data will create an unreliable communication if the communication channel is not stable.

Table 3 is one example of a communication session:

Table 3. Example of Robot-Server Communication

System	Content
SERVER	CONNECT
SERVER	SEND: *CODE: 0, INDEX: 1*
CLIENT	RECEIVE: *CODE: 0, INDEX: 1*
	(MOVE FORWARD IN 3 SECONDS)
CLIENT	SEND: *CODE: 0, INDEX: 1*
SERVER	RECEIVE: *CODE: 0, INDEX: 1*
	(SEND 'DONE' FEEDBACK TO THE CLIENT)
SERVER	SEND: *CODE: 4, INDEX: 2*
CLIENT	RECEIVE: *CODE: 4, INDEX: 2*
	(STOP MOVING)
CLIENT	SEND: *CODE: 4, INDEX: 2*
SERVER	RECEIVE: *CODE: 4, INDEX: 2*
	(SEND 'DONE' FEEDBACK TO THE CLIENT)

4.3 Robot and Mobile phone communication

The robot and mobile phone communication process is exactly the same as the robot and server communication process, except that the command will be sent from the mobile phone to the robot.

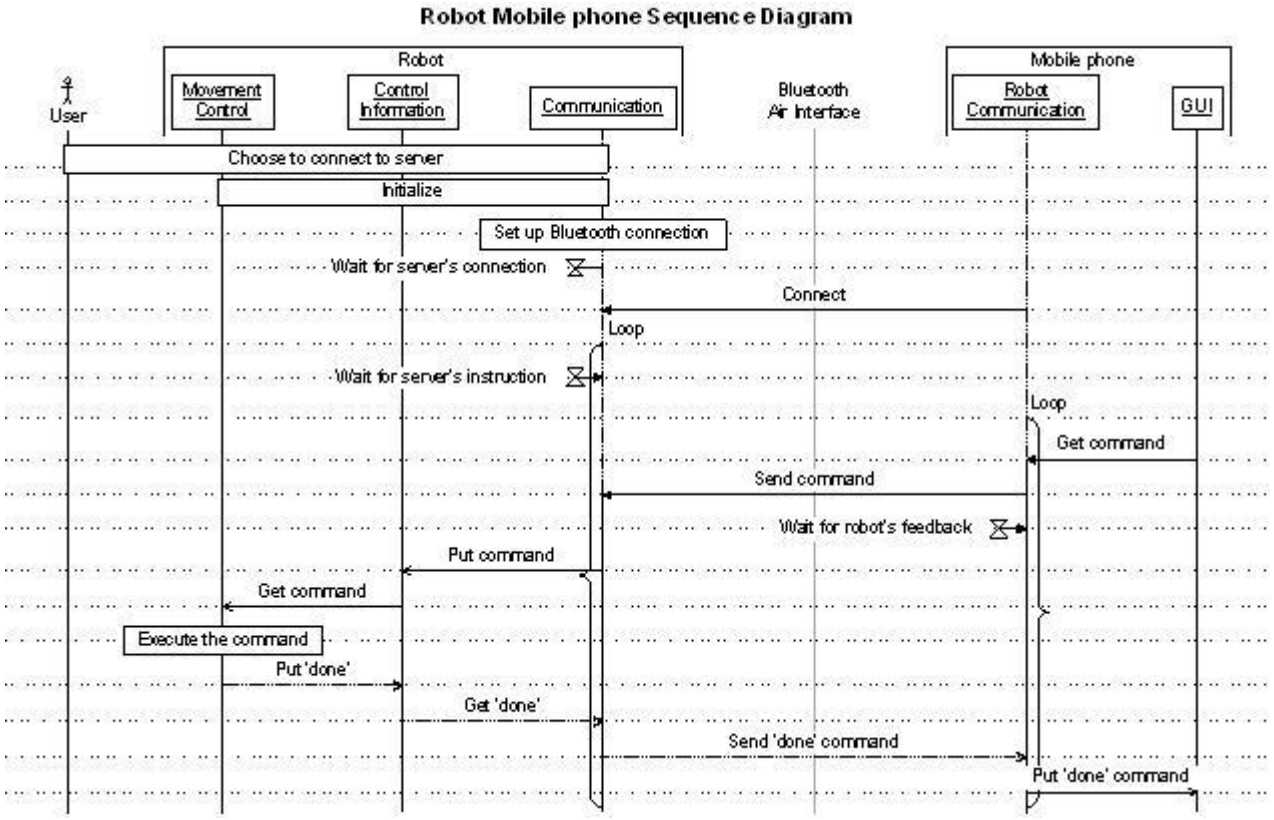


Figure 29. Robot-Mobile phone Communication Sequence Diagram

4.4 Robot and Client communication

As described in Figure 27, the command from the client needs to travel through the server in order to reach the robot. This section discusses two protocols the client needs in order to instruct the robot remotely through the server.

- Firstly, the real-time streaming protocol (RTP) is used for the server to distribute the live streaming webcam to the client's application. Even though the client cannot directly see the robot, by using this protocol, the client still has an opportunity to observe the environment where the robot is located.

- Secondly, a management protocol has to be designed to provide a connection between the server and the client. Through this protocol, the client can send its desired command to the server and the server can also utilize this channel to manage the basic parameters for the live streaming webcam traffic.

- Finally, the client-server protocol supports multiple streaming and management connections at the same time. However, only one client can have the robot controlled right at a given time.

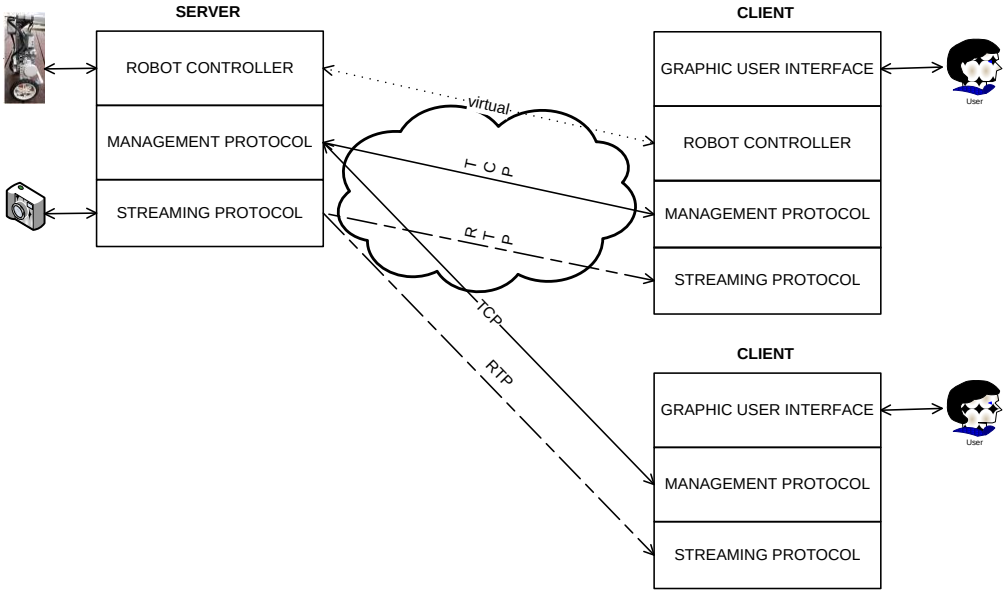


Figure 30. Robot-Client Communication Architecture

4.4.1 Real-time streaming protocol (RTP)

The real-time streaming protocol is not the protocol the author designed. However, a brief introduction of the protocol is given in this section.

The real-time streaming protocol (RTP) was developed by the Audio-Video Transport Working Group of the IETF and first published in 1996 as RFC 1889 and replaced by RFC 3550 in 2003 [23].

As we have already known, the voice and video packets' size is small compared to other commercial protocol such as File Transfer Protocol, Hypertext Transfer Protocol or many Windows network protocols. The reason for it is that the Digital Signal Processing (DSP) unit of audio and video's encoder just needs to digitize a small amount of media information such as 20 ms voice speech or 320x240 pixels raw video picture each processing time. However, due to the high sampling rate of the audio and video's DSP encoder, the amount of packets is relatively large. The busted stream of audio/video traffic will simply use up all the network resource. This is also the reason why the audio and video traffic is usually called *stream*.

RTP is designed to solve the data communication problems of audio and video streaming.

- Less overhead: RTP is a transport protocol which is created based on the UDP protocol. It allows less overhead communication compared to the TCP protocol. Thanks to this implementation, it reduces the latency of the packet as it does not include the three-way handshake and the acknowledgement mechanism. Without the acknowledgement mechanism, the sender does not need to wait for the acknowledged message everytime it sends a message to the client. Therefore, the transmission speed is increased.

- Guaranteed Order: RTP inherits the unreliable communication characteristic of UDP. It is not assured that the packets will reach the destination and also the order of the packets cannot be guaranteed. RTP modifies the UDP protocol by adding the sequence number to rearrange the messages in the end of the connection.

- Jitter Buffer: In addition, RTP defines a jitter buffer for the communication, it helps to eliminate the jitter in the communication. Jitter is simply the phenomenon when the packets reach the destination at different spaces of time. In a live streaming, the replay speed is changing variably, which is considered ‘annoying’.

- Payload format: RTP embeds the video/audio format information into the payload, which optimizes the transferred audio and video data over the network.

- Synchronized streams: RTP uses a sub-protocol called RTP Control (RTPC) to synchronize the time between the media streams. It helps one endpoint of the communication to have a better knowledge of what is happening at f the other endpoint.

With all of these advantages compared to TCP and UDP, the author has chosen RTP as the protocol to distribute the webcam traffic from the server to the client.

The RTP payload format for the video streaming and also the sequence diagram of the protocol are not discussed here. In case the readers want to research more about the protocol, the readers are referred to the RFC 3016 [24] document.

Figure 31 presents the overview architecture of the RTP protocol. On the server side, the raw data of the capture device will be encoded into RTP payload format for video streaming. The server will create a session manager with its own RTP port and connect with the client session through the client's IP and client's RTP port. On the client side, after receiving the streaming data from the server, the client will decode it into an appropriate format and display it to the player.

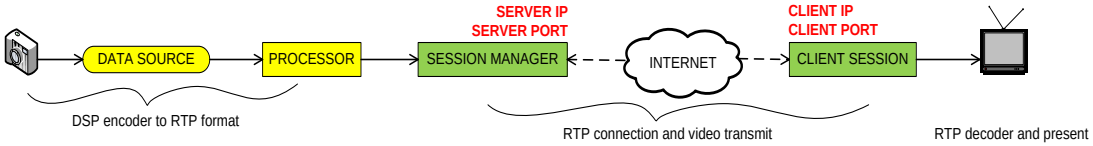


Figure 31. RTP Communication

Most difficult tasks of RTP communication are carried out on the server side. The only necessary task on the client side is building a session manager which connects to the server with the RTP locator:

rtp:// *server__ip_address* **:** *client_RTP_port* **/** *content-type* **/**

For example, if the server's IP address is 209.165.203.144 with RTP port 50000 and the client's RTP port is 50004, the RTP locator that the client needs to use is rtp://209.165.203.144:50004/video.

Note that there is a huge misunderstanding about the RTP port in the RTP locator. The server's RTP port is usually used in the locator because the server's port and client's port are usually the same. However, it does not mean that the RTP port in the locator is the server's port.

4.4.2 Server and Client Communication

The protocol design has two parts: the sequence diagram of the protocol and the explanation with the communication message structures.

The server and client communication is a scalable protocol, which is widely used over the Internet. A reliable transport protocol is needed. For that reason, TCP is the chosen transport protocol for this communication.

Figure 33 describes the sequence of the server-client communication in the application layer. For further reading about the transport protocol TCP, the RFC 793 document [25] is recommended.

- After the user has activated the server, the server will create a *server socket* and listen on port TCP 33333. The author uses port 33333 as it is his favorite number.

- The client application creates a *socket* and a connection to the *server socket*. Of course, the IP address of the server has to be known in advance. If there is no problem, the server will accept this connection and create a *new socket* to communicate with the *client socket*.

- After initializing the connection, the server will send the first message to the client to ask for authentication. The client will submit its username and password to the server a responding to the authentication request.

  - The server authenticates the client's username and password based on its own database. Upon the successful authentication, the server will send to the client the congratulation message and also grant the client a privilege level. There are 3 privilege levels for this system: monitor only (privilege level 2), control only (privilege level 5), and monitor and control (privilege level 15). The privilege level is also a part of information which the database contains.

  - Upon the fail authentication, the server will simply send a goodbye message to the client and interrupt the connection.

- Depending on the privilege level of the client, the server-client communication will display different behaviors.

  - Privilege level 15 (control right + monitor right): The monitor right will be discussed in detail in privilege level 2 due to the same behaviour. The control right allows the client to send the request commands to the server. The server's mission is receiving the commands from the client, importing the commands into the commands list so that the Robot communication thread can send them to the robot. After every requested command is completed, the server will send a feedback message to the client application.

- Privilege level 5 (control only): the server will listen to the commands from the client and also send the feedback to the client every time the command is completed.

- Privilege level 2 (monitor only): after the congratulation message, the server will send a message including the information about the RTP port the client has to use. After that, the server will stream the webcam information to the client's RTP port. There is a scenario we need to be concerned with at this privilege level. That is when a new client joins into the streaming group. The author decided that the server will stop all the current streaming connections and start all of them again with the new streaming client. The reason for this behaviour is because of the difficulty to 'hot-plug' a new streaming connection compared to restarting all the connections. When adapting to this scenario, the server will send a RTP changing request to the clients to ask them to restart their RTP connections.

- The communication will end when one side of the communication process sends a Goodbye message.

In order to make the protocol simple, the author uses only one message structure for all the messages. Figure 32 presents the client-server communication message structure.

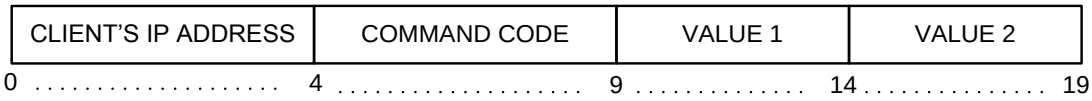


Figure 32. Client-Server communication message's structure

- Every message which is sent from the client to the server or from the server to the client will have 4 bytes of the client's IP address in the beginning. This information is duplicate in the IP address in the Layer 3 header, but it is needed for the application layer's management system.

- The command code is 4 bytes long and the value code is 20 bytes long. However, all the fields will have 1 byte longer for the delimiter code. An example will be shown later in Table 5 which explains the delimiter code.

- The command code with its appropriate value code is presented in Table 4.

Table 4. The command code in the Server-Client message

Command code	Value 1	Value 2	Description
USER			Server asks for the authentication
PASS	Username	Password	Client replies the authentication info
SUCC	Priv. level		Server sends the congratulation
BYET			Goodbye message
RTPI	RTP port		Servers sends the RTP client port
RTPC			Server requests RTP restarts
FORW	Index		Robot command: Move forward
BACK	Index		Robot command: Move backward
RIGH	Index		Robot command: Turn right
LEFT	Index		Robot command: Turn left
STOP	Index		Robot command: Stop
DONE	Index		Server sends the ‘done’ feedback

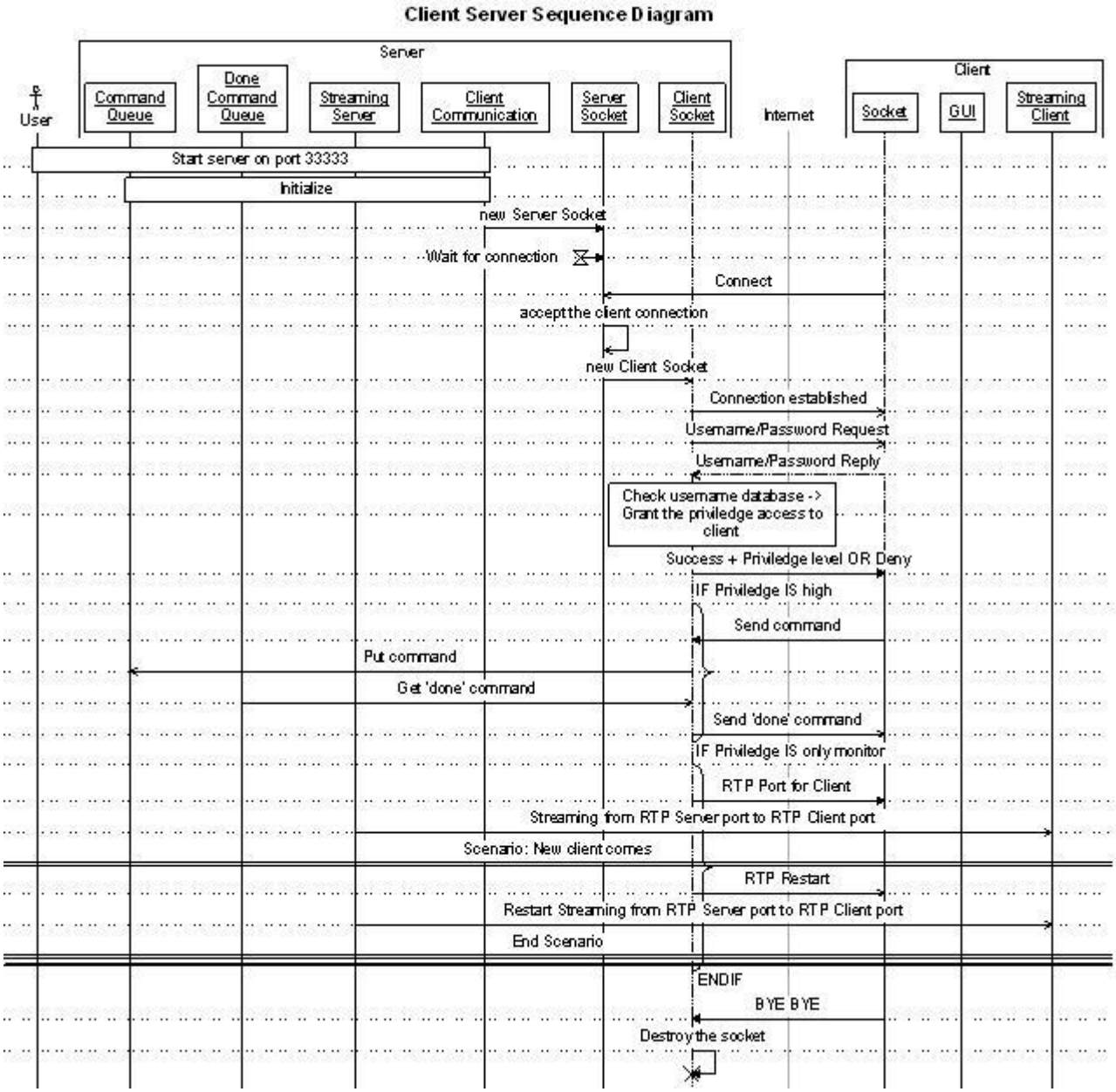


Figure 33. Server-Client Communication Sequence Diagram

Table 5 is one example of a communication session. In this example, the delimiter code is ~ (tilde). It is used to separate the command code and value codes from a unique string.

Table 5. Example of Robot-Client Communication

System	Content
CLIENT	CONNECT *(Server's IP address: 192.168.1.254* *Client's IP address : 192.168.1.10)*
SERVER	ACCEPT
SERVER	SEND: 192.168.1.10~USER
CLIENT	SEND: 192.168.1.10~PASS~tero~teropassword
SERVER	SEND: 192.168.1.10~SUCC~15
SERVER	SEND: 192.168.1.10~RTPI~50004
CLIENT	*START RTP SESSION WITH RTP LOCATOR* *rtp://192.168.1.254:50004/video/*
CLIENT	SEND: 192.168.1.10~FORW~1
CLIENT	SEND: 192.168.1.10~LEFT~2
CLIENT	SEND: 192.168.1.10~LEFT~3
SERVER	SEND: 192.168.1.10~DONE~1
SERVER	SEND: 192.168.1.10~RTPC
CLIENT	*RESTART RTP SESSION WITH RTP LOCATOR* *rtp://192.168.1.254:50004/video/*
SERVER	SEND: 192.168.1.10~DONE~2
CLIENT	SEND: 192.168.1.10~STOP~4
SERVER	SEND: 192.168.1.10~DONE~4
CLIENT	SEND: 192.168.1.10~BYET

5 Project Implementation and Integration

5.1 Robot systems' design

In order to make the robot self-balancing, a concurrency architecture is needed. The concurrency architecture of the robot program was described in Figure 17. The program needs three threads.

- Balancing Controller: The information about the tilt angle, tilt velocity, wheel angle and wheel velocity will be collected from the sensory module. The PID

controller will process the information and instruct the motors how they should move. The PID controller will be discussed in detail in the next chapter. This process will be re-done since the control system for this system is a closed-loop control system. This thread has access to the shared parameters list in order to listen to the movement's instructions from the Arbitrator thread.

- Communication Manager: The Bluetooth connection will be started and the data will be transferred in this thread. The command that the robot receives will be put into a shared parameters list, so that the Arbitrator thread can read and execute it.

- Arbitrator: The Arbitrator thread creates two sub threads which represent two behaviours of the robot include collision avoidance and movement control. The Arbitrator reads the command from the parameters list and writes some necessary instructions into the parameters list. The Balance controller thread reads the instructions and modifies the PID controller to move the robot according to the command.

All the features of this system were implemented, except for the collision avoidance behaviour. The main reason is because it was unnecessary for the robot at the time the thesis was written.

There were no difficult problems during the robot system development and the source code of robot system is presented in Appendix A.1. However, the most tricky task in developing this system is the robot's movements.

- In order to make the robot turn around, the movement control behaviour just needs to apply opposite offsets to the PID's output value of each motor. Because of the opposite value, the motor will turn in the opposite direction compared with the other one and it will make the robot turn around.

- Making the robot move forward and backward are tricky tasks. Thanks to the instructor's suggestions, the author applied many methods which can move the robot. The method which is used in the thesis is making an offset in the calculation of the motor angle. The idea behind this method is the differentiation from the equilibrium point of motor angle. If we imagine that if the robot sees the motor angle as different from the equilibrium point, the robot will tend to move

back to the equilibrium point to compensate for that differentiation. It then makes the robot move forward or backward.

5.2 Server systems' detailed design

Its main purpose is support the client to navigate the robot remotely. In order to solve that problem, the server is designed with five threads as shown in Figure 34.

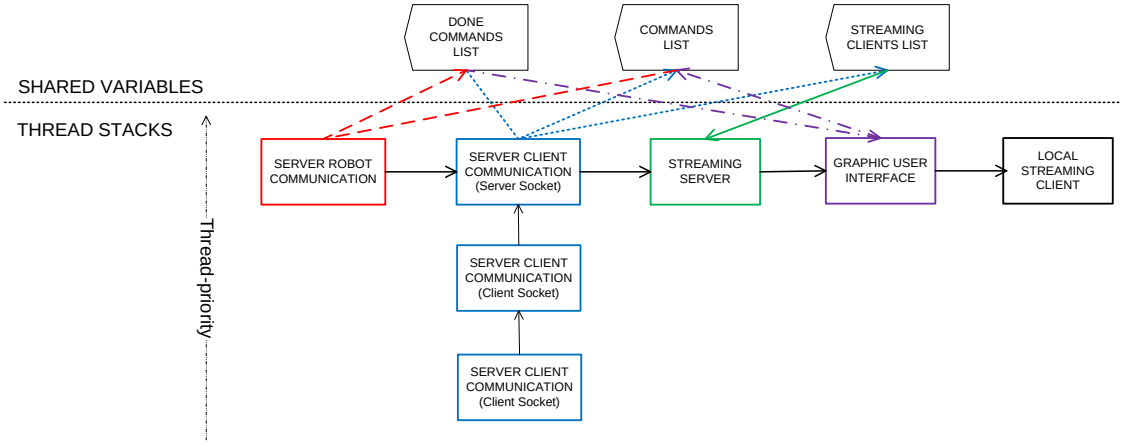


Figure 34. Server system’s concurrency architecture

- The Server Client Communication thread is used to make the server socket which listens on port 33333 and waits for the client’s connection. The Server Client communication thread will create a sub-thread for every client communication. This design ensures that the server can respond to multiple client’s requests at the same time.

- The Server client communication sub-thread will execute the authentication and command redistribution from the client to the robot as well as the feedback redistribution from the robot to the client. All of the tasks will be carried out through the shared variables *commands list* and *done commands list*.

- The Server robot communication thread will retrieve the command from the *commands list* and send it to the robot with an appropriate index code. By using the index code, the server will identify the feedback from the robot to enter it into the *done commands list*. At this moment, the implementation of the index code is not important. But it should be designed for future expansion. Imagine the case where the server grants multiple robot control rights to multiple clients. If the server sends a command to the robot and receives the feedback, it is hard

to say which client this feedback belongs to without a tracking code. And the tracking code in this situation is the index code.

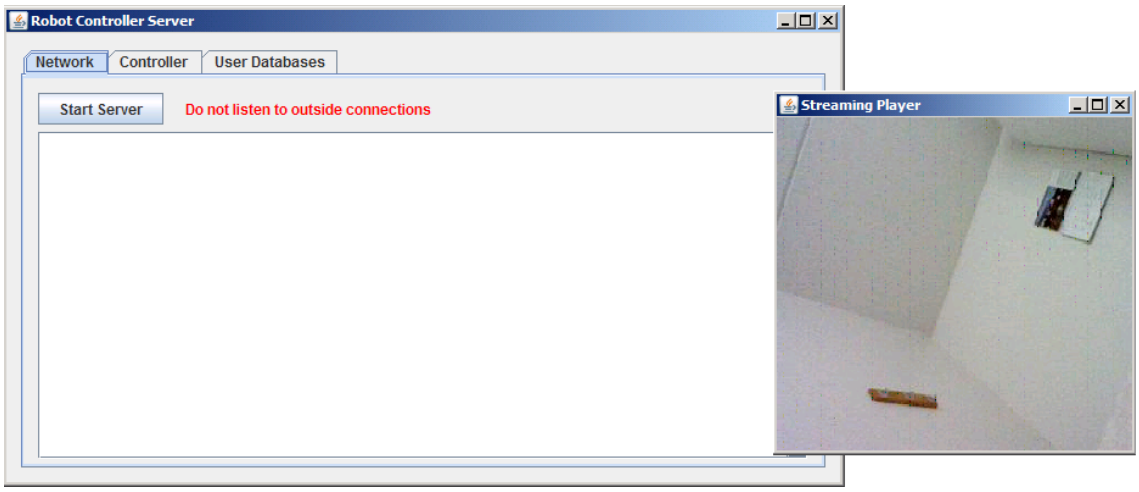
- The streaming server thread is a hard-working thread. Its missions are managing the RTP connections with the RTP clients, retrieving the raw data from the web camera, encoding the data into RTP JPEG streaming data, and sending the RTP data to all the clients in the *streaming clients list.* Every time a new client joins the *streaming clients list,* the streaming server thread will re-start all the current connections.

- The Graphic User Interface is also implemented as a separate thread. It also has access to the *commands list* and *done commands list* due to its ability to control the robot locally on the server.

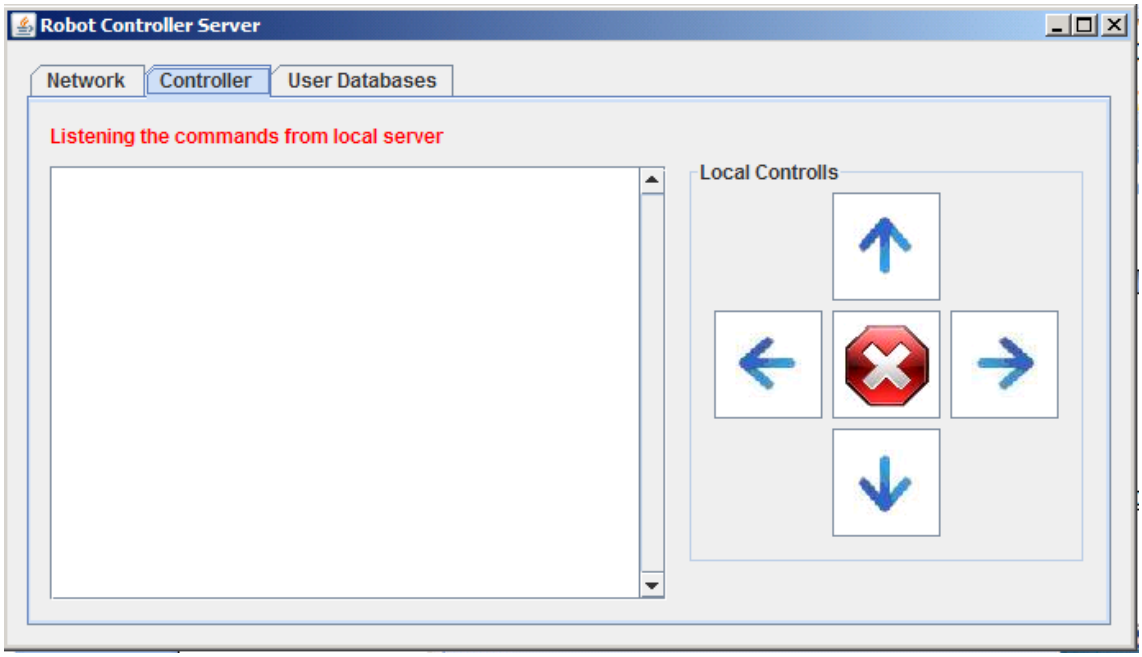
- The local streaming client is a thread which makes a RTP session from the server to itself in order to test the streaming server's output data and observe the robot locally.

All the features described above were implemented in the server system.

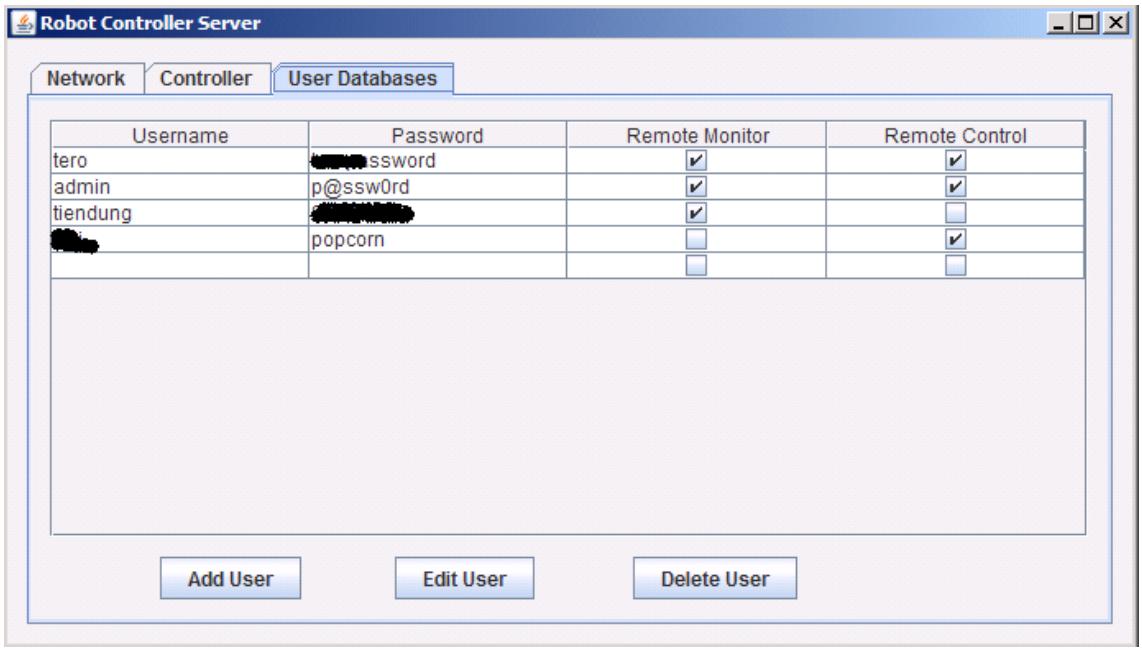
Every design requirement requires different knowledge areas and has different kind of difficulties. In Appendix A.2, the source code of the server is presented in detail with comments. The application's image is presented in Figure 35.



a) The network panel – List all the client's activities



b) The controller panel – List all the commands from client and local control



c) The user databases panel – List all the current users and privilege options

Figure 35. Server application's screenshots

Even though the readers can recreate the system by using the source code, the two most interesting development experiences of the implementation and integration phase will be discussed here.

- Multiple RTP clients: As mentioned before, multiple RTP clients capability leads to the problem of restarting the connections every time a new client comes in. The author had to accept this irritation because of the difficulty to implement the clone-able data source. In Figure 31, we have discussed about the RTP streaming connection between a server and a client. If we want multiple RTP streaming connections between a server and multiple client, we need to create a clone of the data source and create a thread which will stream the cloned data source to a client. The more clients join the group, the more threads we have and the more cloned data sources are created. It significantly impacts the performance of the system. Therefore, in order to maintain a good performance for the system, the author had to choose the solution in which one data source will be streamed to multiple clients, and when a new client comes, the streaming server thread will re-start all the connections.

- Non-blocking input/output (NIO) socket: Non-blocking I/O socket is a new concept of socket programming. The traditional socket programming, also the socket programming used in the thesis, has a great weakness. Whenever it is waiting for a connection or an expected input, it will block all other processes. This behaviour is known as blocking the I/O socket. In the thesis, when the user starts the server, the server socket is created an listens to port 33333. If at that time, there is no client connecting to the server, the server cannot stop listening and destroy the server socket by itself. The reason is because the server socket is still waiting for the client connection and will block the I/O, therefore, the destroy action will not be conducted. This behaviour of the traditional socket programming is considered annoying for the network application developers. In order to solve this problem, NIO should be used as it is a non-blocking I/O socket. However, the author has attempted to destroy the server socket at that time by creating a client connecting from the server to itself to help the server exit from the blocking mode. After that, the client connection terminates by itself and lets the server socket be destroyed by the normal method.

5.3 Client systems' design

The main purpose of the client system is to deliver the commands to the server, receive the feedbacks and the webcam streaming data from the server.

To accomplish its tasks, the client system is designed according to this architecture in Figure 36.

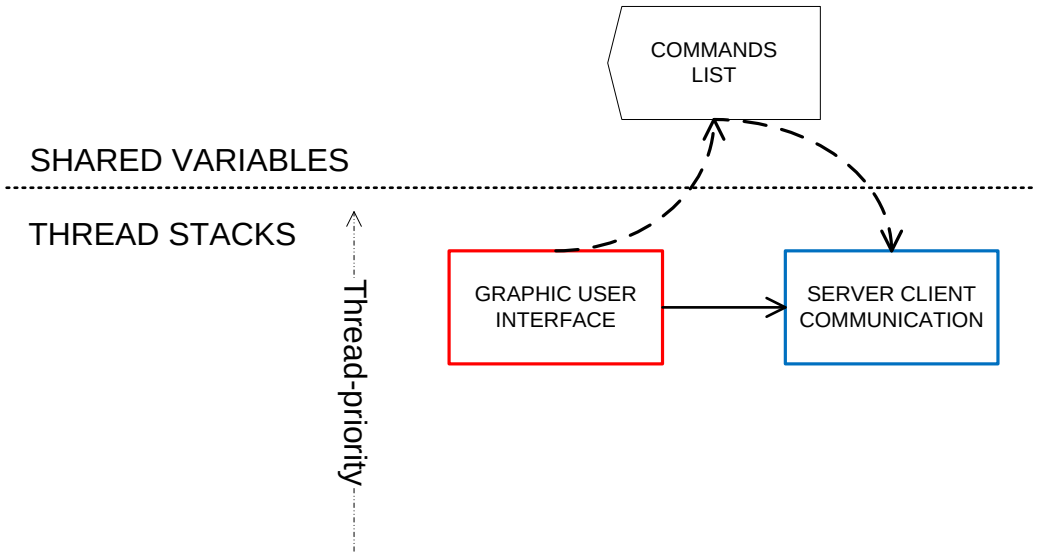


Figure 36. Client system's concurrency architecture

The graphic user interface thread is used to receive the command from the user and enter it into the *command lists*.

The server client communication retrieves the command from the commands list and sends it to the server. The feedback will be received from the server to acknowledge that the command is completed.

There were no problems in the development of the client system. The source code of client application can be found in Appendix A.3 with comments. The client application's image is presented in Figure 37.

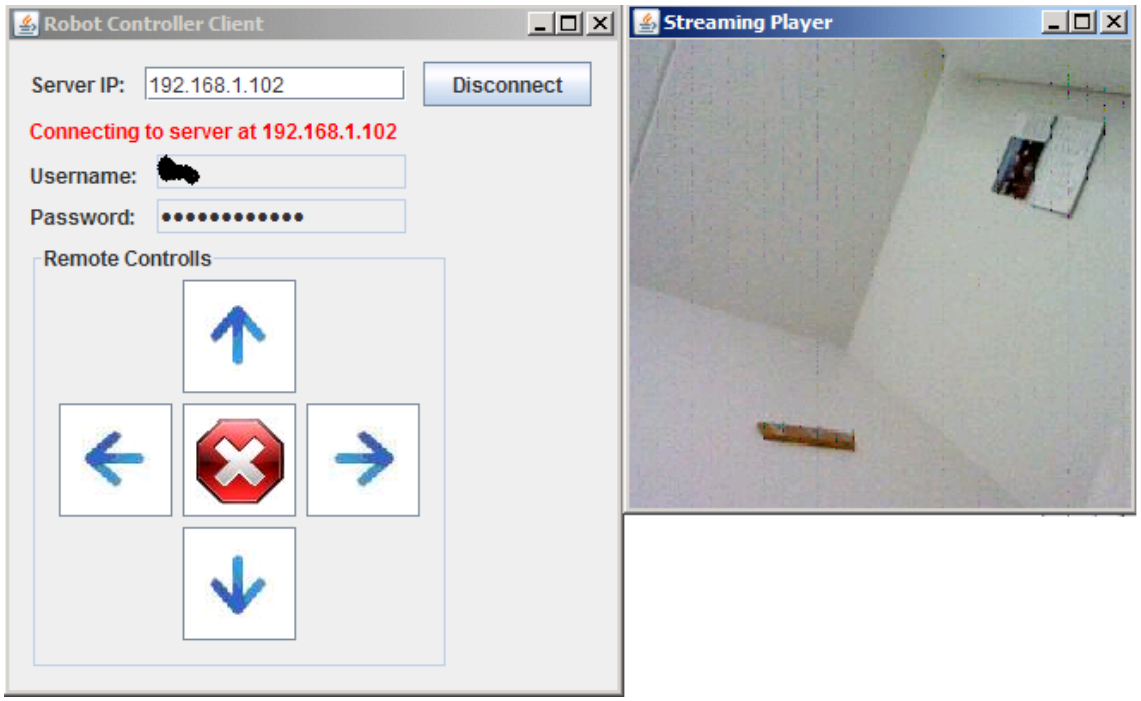


Figure 37. Client application's screenshot

5.4 Mobile phone systems' design

The mobile phone system's architecture is generally the same as the client system's. However, the design is simpler.

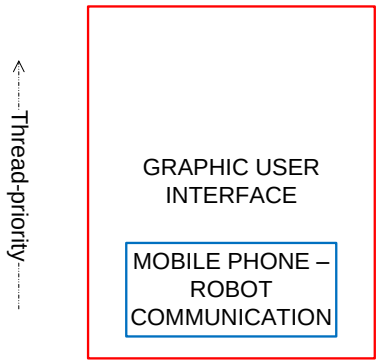


Figure 38. Mobile phone system's architecture

In order to limit the mobile phone's ability, the author designs only one thread which will send and receive one command and feedback every time for the mobile phone. The reason of this design is the limitation of the Bluetooth protocol as a half-duplex protocol in leJOS.

In the scope of the thesis, the author did not implement the mobile phone system due to duplications with other systems and lack of the developing time.

6 Summary

The overall architecture of the system has been presented in Chapter 3. It builds up a good foundation about how the system works and what parts the system need to be included. Through this chapter, the readers can have an overview of the system, they can then evaluate, adapt and expand the system's initial framework to their own system. For the readers who have no knowledge of embedded systems and network programming, this chapter is a good start for them to establish the necessary background in order to go through the following chapters.

The detailed designs about the two-wheeled self-balancing robot model and communication protocols have been discussed in Chapter 4. The control system which makes the robot balancing in its upright position is presented in an easy way, so that the audience without background of modern mechanic physics can easily understand it. The communication protocols are described and detailed examples are given . The readers can easily understand the communication sessions and the ideas behind the design. The readers can modify the protocols to their own needs.

Chapter 5 provides information about the implementation and integration phase. By reading this chapter, the readers will experience the technical difficulties and challenges when integrating all the separate protocols and sub-systems together.

In chapter 5, the final result of implementation and integration phase is clearly discussed. It proves that the overall and detailed designs in Chapter 3 and 4 are feasible and applicable.

7 Recommendation

There are numerous possibilities to improve this thesis project.

- Improve the PID's gain factors: The author achieves the PID's gain factors through estimation, trial and errors. As a result, the robot cannot perfectly balance in its upright position. The real experiment shows that the robot will drift

forward a little all the time. Owing to that reason, improving the PID's gain factors by applying Nyquist's theorem evaluation is recommended.

- Test and apply a better solution for moving the robot forward and backward: The current solution of moving the robot forward and backward works. However, the robot only moves a very short distance every time it receives a forward command. Many novel ideas can be tested and applied in order to find the best solution of moving the robot.

- Improve the streaming server: First of all, the streaming server should implement a component which can change the bit rate of the streaming data when the client receives an alarm about great latency. Secondly, the solution of restarting RTP connections when a new client joins the group should be avoided, a new solution should be designed and implemented.

With decent research and development in the future work, the current system can be improved and it could be possible to reach the ultimate goal of designing an initial framework for a remote-controlled robot through the Internet.

REFERENCES

[1] *Segway model i167*. [www-document]. Available at: http://segway.com/img/content/models/i167.jpg. Accessed on 12th April 2010.

[2] Yamamoto, Y., *NXTway-GS Model-Based Design-Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT*. Cybernet Systems Co., Ltd., 2008.

[3] Rieper, J., Bisballe Nyeng, B., and Sohn, K.. *Marvin - The Balancing Robot*. Aarhus University, 2009.

[4] Ogata, K., *Modern Control Engineering*, Prentice Hall, 5th Edition, 4th September 2009, pp-912.

[5] Byrnes, P. , *Protocol Management in Computer Networking*, Artech House Publishers, January 2000, pp-486.

[6] Hassenplug, S. , *Steve's Legway*, 2002. [www-document]. Available at: http://www.brickshelf.com/gallery/Hassenplug/LegWay/legway1s.jpg. Accessed on 12th April 2010.

[7] Anderson, D. , *nBot version 3*, 2003. [www-document]. Available at:http://www.geology.smu.edu/~dpa-www/robo/nbot/nbot_3.html. Accessed on 12th April 2010.

[8] Ko, A. , Lau, H.Y.K., and Lau, T.L.. "SOHO Security with Mini Self-Balancing Robots". *Industrial Robot: An International Journal*, 32(6):492-498, 2005.

[9] Kyle, J., Trevor, T., *Professional Microsoft Robotics Developer Studio*, Wiley Publishing, p.575, 2008.

[10] Wong, W., *Real Robots: Lego Mindstorm NXT*, 17th February 2007. [www-document]. Available at: http://electronicdesign.com/article/boards-modules-systems/real-robots-lego-mindstorm-nxt14920.aspx. Accessed on 14th April 2010.

[11] *The NXT Intelligent Brick*, 2004. [www-document]. Available at: http://www.sutree.com/upload/thumbnails/36641.gif. Accessed on 12th April 2010.

[12] Wikipedia, *Communication protocol.* [www-document]. Available at: http://en.wikipedia.org/wiki/Communications_protocol. Accessed on 14th April 2010.

[13] Comer, D.E., *Computer Networks and Internets*, Prentice Hall, 5th Edition, 28th April 2008, pp-768.

[14] Wikimedia, *TCP/IP Stack Connections* . [www-document]. Available at: http://upload.wikimedia.org/wikipedia/commons/c/c4/IP_stack_connections.svg. Accessed on 12th April 2010.

[15] *The LEGO NXT's servo motor* . [www-document]. Available at: http://www.active-robots.com/products/mindstorms4schools/nxt-accessories/9842-prod-02-inside-500.jpg. Accessed on 12th April 2010.

[16] Yamamoto, Y., *NXTway-GS Building Instructions of self-balancing two-wheeled robot built with LEGO Mindstorms NXT*. Cybernet Systems Co., Ltd., 2008.

[17] Wikipedia, *The PID Controller*. [www-document]. Available at: http://upload.wikimedia.org/wikipedia/commons/4/40/Pid-feedback-nct-int-correct.png. Accessed on 14th April 2010.

[18] Kyle, J., Trevor, T., *Professional Microsoft Robotics Developer Studio*, Wiley Publishing, p.594, 2008.

[19] *The PID Tuning Plot*. [www-document]. Available at: http://www.expertune.com/tutor.html. Accessed on 12th April 2010.

[20] Watanabe, R.. *Motion Control of NXTway(LEGO Segway)*. Waseda University, 2007.

[21] Wikipedia, *Torque*. [www-document]. Available at: http://en.wikipedia.org/wiki/Torque#Static_equilibrium. Accessed on 16th April 2010.

[22] Jain, R., *Wireless Personal Area Networks (WPANs)*, Washington University in Saint Louis, 2006.

[23] Wikipedia, *Real-time transport protocol*. [www-document]. Available at: http://en.wikipedia.org/wiki/Real-time_Transport_Protocol. Accessed on 16th April 2010.

[24] Networking Working Group, RFC 3016, *RTP Payload Format for MPEG-4 Audio/Visual Streams*, November 2000. [www-document]. Available at: http://tools.ietf.org/html/rfc3016. Accessed on 18th April 2010.

[25] Darpa Internet Program, *Transmission Control Protocol*, September 1981. [www-document]. Available at: http://tools.ietf.org/html/rfc793. Accessed on 18th April 2010.

[26] Sun Microsystems, *Capturing Time-Based Media with JMF*. [www-document]. Available at: http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/JMFCapturing.html. Accessed on 18th April 2010.

[27] Fischer, D., *Capture Video from Logitech QuickCam Pro 3000 Camera with Java JMF*, 23th January 2003. [www-document]. Available at: http://www.mutong.com/fischer/java/usbcam. Accessed on 18th April 2010.

Appendix A.1 – Robot System

The three files MotorController.java, GyroscopreSensor.java and BalanceController.java are cited from Marvin NXT source code [3].

Main.java Source Code

```
import lejos.nxt.*;
import lejos.robotics.subsumption.*;

public class Main {
    private static int mode;

    @SuppressWarnings("static-access")
    public static void main(String[] args) throws Exception
    {
        ControlInformation ci = new ControlInformation();

        getMode();

        // create Balance control thread
        BalanceController bc = new BalanceController(ci);

        // create behaviours of the robot
              Behavior movement = new BehaviorMovement(ci);

        Behavior[] bArray = {movement};
              Arbitrator arbitrator = new Arbitrator(bArray);

        Communication communication;
        if (mode == 1) communication = new RobotServerComm(ci);
        else communication = new RobotPhoneComm(ci);

        bc.start();
        communication.start();
        arbitrator.start();
    }

    public static void getMode()
    {
        int pressedButton;

        LCD.clear();
        LCD.drawString("Control mode", 0, 0);
        LCD.drawString("LEFT - Server", 0, 1);
        LCD.drawString("RIGHT - Phone", 0, 2);

        pressedButton = Button.readButtons();
        while (pressedButton == 0 || pressedButton == 1)
        {
            pressedButton = Button.readButtons();
        }

        if (pressedButton == 2)
            mode = 1;

        if (pressedButton == 4)
            mode = 2;

        if (pressedButton == 8)
            System.exit(0);

        LCD.clear();
    }
}
```

BalanceController.java [3] Source Code

```
import lejos.nxt.*;

public class BalanceController extends Thread
{
    private ControlInformation ci;

              // The PID control parameters
              private final double Kp = 1.25;//1.2;
              private final double Ki = 0.25;//0.25;
              private final double Kd = 0.10;//0.1;

              // Testing error contributions.
              private final double K_psi = 31.9978;
              private final double K_phi = 0.8703;
              private final double K psidot = 0.6;
              private final double K phidot = 0.02;

              private final int ACCEL_OFFSET = 6;

              public BalanceController(ControlInformation value)
              {
                            ci = value;
              }

              public void run()
              {
                            while (ci.getStartable() == false)
                            {}

                            double int_error = 0.0;
                            double prev error = 0.0;

                            MotorController motors = new MotorController(Motor.B,
Motor.A);
                            GyroscopeSensor gyro = new GyroscopeSensor(SensorPort.S3);
                            //GyroSensor accel = new GyroSensor(SensorPort.S2);

                            UltrasonicSensor accel = new UltrasonicSensor(SensorPort.S2);
                            int accel value = accel conversion(accel.getDistance());

                            LCD.clear();
                ci.setMovable(true);

                            while (!Button.ESCAPE.isPressed())
                            {
                                          double Psi = gyro.getAngle() + accel_value;
                                          double PsiDot = gyro.getAngleVelocity();

                                          double Phi = motors.getAngle() -
ci.getTiltAngle();
                                          double PhiDot = motors.getAngleVelocity();

                                          // Proportional Error
                                          double error = Psi * K psi + Phi * K phi +
PsiDot * K psidot + PhiDot * K phidot;

                                          // Integral Error
                                          int_error += error;

                                          // Derivative Error
                                          double deriv error = error - prev error;
                                          prev_error = error;

                                          // Power sent to the motors
                                          double pw = error * Kp + deriv error * Kd +
int_error * Ki;
```

```
                                        motors.setPower(pw + ci.getLeftMotorOffset() ,
pw + ci.getRightMotorOffset() );

                        // Delay used to stop Gyro being read to quickly. May need to be
increase or
                                        // decreased depending on leJOS version.
                                        delay(7);
                                }

                   System.exit(0);

              }

              public static void delay(int time)
              {
                                try
                                {
                                        Thread.sleep(time);
                                }
                                catch (Exception e) {}
              }

              public int accel conversion(int accel raw value)
              {
                                if (accel raw value == 0) return 0;

                                if (accel_raw_value > 0 && accel_raw_value < 50)
                                        return (-90 * accel_raw_value / 50) +
ACCEL OFFSET;

                                if (accel raw value > 200 && accel raw value < 256)
                                        return (90 * (255 - accel_raw_value) / 50) +
ACCEL_OFFSET;

                                return 0;
              }
}
```

GyroscopeSensor.java [3] Source Code

```
import lejos.nxt.*;
import lejos.nxt.addon.GyroSensor;

public class GyroscopeSensor
{
              private GyroSensor gyro;
              private double angle = 0.0;
              private int lastGetAngleTime = 0;
              private double offset = 0;

              public GyroscopeSensor(SensorPort input_port)
              {
                                gyro = new GyroSensor(input port);
                                gyro.setOffset(606);
                                calcOffset();
              }

              public void calcOffset()
              {
                                double offsetTotal = 0;

                                LCD.drawString("Calibrating Gyro", 0, 2);

                                for (int i = 0; i < 100; i++)
                                {
                                        offsetTotal += (double) gyro.readValue();
                                        try
                                        {
                                                Thread.sleep(4);
                                        }
```

```
                                        catch (InterruptedException e) {}
                        }

                        while (!Button.ENTER.isPressed())
                        {
                                        offset = Math.ceil(offsetTotal / 100);
                                        LCD.drawString("Calibration Done", 0, 4);
                                        LCD.drawString("offset: " + offset, 2, 5);
                                        LCD.drawString("Press Enter", 1, 6);
                        }

        }

        public double getAngleVelocity() // Unit: Psi/second
        {
                        return (double) gyro.readValue() - offset;
        }

        public double getAngle()
        {
                        int now = (int) System.currentTimeMillis();
                        int delta_t = now - lastGetAngleTime;

                        if(delta t != now) {
                                        angle += (getAngleVelocity() + offset) *
((double) delta t / 1000.0); // Psi' = Psi + (Psi/second) * (t' - t)
                        }
                        lastGetAngleTime = now;

                        return angle;
        }

}
```

MotorController.java [3] Source Code

```
import lejos.nxt.*;

class MotorController
{
        private Motor leftMotor;
        private Motor rightMotor;

        private double sin_x = 0.0;
        private final double sin_speed = 0.1;
        private final double sin_amp = 20.0;

        public MotorController(Motor leftMotor, Motor rightMotor)
        {
                        this.leftMotor = leftMotor;
                        this.leftMotor.resetTachoCount();

                        this.rightMotor = rightMotor;
                        this.rightMotor.resetTachoCount();
        }

        public void setPower(double leftPower, double rightPower)
        {
                        sin x += sin speed;
                        int pwl = (int) (leftPower + Math.sin(sin x) * sin amp);
                        int pwr = (int) (rightPower - Math.sin(sin_x) * sin_amp);

                        leftMotor.setSpeed(pwl);
                        if (pwl < 0)
                        {
                                        leftMotor.backward();
                        } else if (pwl > 0)
                                        {
                                                        leftMotor.forward();
                                        }
```

```
                                        else
                                        {
                                                leftMotor.stop();
                                        }

                        rightMotor.setSpeed(pwr);
                        if (pwr < 0)
                        {
                                        rightMotor.backward();
                        } else if (pwr > 0)
                                        {
                                                rightMotor.forward();
                                        }
                                        else
                                        {
                                                rightMotor.stop();
                                        }
                }

                public double getAngle()
                {
                        return ((double) leftMotor.getTachoCount() + (double)
rightMotor.getTachoCount()) / 2.0;
                }

                public double getAngleVelocity()
                {
                        return ((double) leftMotor.getRotationSpeed() + (double)
rightMotor.getRotationSpeed()) / 2.0;
                }

                public void resetMotors()
                {
                        leftMotor.resetTachoCount();
                        rightMotor.resetTachoCount();
                }

                public void stop()
                {
                        leftMotor.stop();
                        rightMotor.stop();
                }
}
```

BehaviorMovement.java Source Code

```
import lejos.robotics.subsumption.*;

public class BehaviorMovement implements Behavior
{
    ControlInformation ci;
    private int MOVE TIME = 2000;
    private int TURN TIME = 400;

    public BehaviorMovement(ControlInformation value)
    {
        ci = value;
    }

    public boolean takeControl()
    {
        return true;
    }

    public void suppress()
    {
        // info.setStatus(CS.NOTHING_STATUS);
    }
```

```
    public void action()
    {
        // in case the robot should be moving, check the desire information and move the
robot
        // if (info.isMovingBack() || info.isMovingForward() || info.isMovingLeft() ||
info.isMovingRight() || info.isMovingAround())
        // {
        //   info.setStatus(CS.BUSY STATUS);

        if (ci.getMovable())
        {
            if (ci.getMoving())
            {
                if (ci.getDirection() == RobotReturnProtocol.CODE FORWARD)
                {
                    for (int i=0; i<10; i++)
                    {
                        try {
                            Thread.sleep(10);
                        }
                        catch(InterruptedException e) { }
                        ci.increaseTiltAngle();
                    }
                }

                if (ci.getDirection() == RobotReturnProtocol.CODE BACKWARD)
                {
                    for (int i=0; i<10; i++)
                    {
                        try {
                            Thread.sleep(10);
                        }
                        catch(InterruptedException e) { }
                        ci.decreaseTiltAngle();
                    }
                }

                if (ci.getDirection() == RobotReturnProtocol.CODE_TURNLEFT)
                {
                    ci.decreaseLeftMotorOffset();
                    ci.increaseRightMotorOffset();

                    try {
                        Thread.sleep(TURN_TIME);
                    }
                    catch(InterruptedException e) { }

                    ci.resetLeftMotorOffset();
                    ci.resetRightMotorOffset();
                }

                if (ci.getDirection() == RobotReturnProtocol.CODE TURNRIGHT)
                {
                    ci.increaseLeftMotorOffset();
                    ci.decreaseRightMotorOffset();

                    try {
                        Thread.sleep(TURN TIME);
                    }
                    catch(InterruptedException e) { }

                    ci.resetLeftMotorOffset();
                    ci.resetRightMotorOffset();
                }



                try {
                    Thread.sleep(100);
                } catch (InterruptedException ex) { }

                ci.setMoving(false);
```

```
            }

        }
    }
}
```

ControlInformation.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class ControlInformation {
    private boolean bStartable;
    private boolean bMovable;
    private boolean bMoving;
    private int iDirection;
    private double dTiltAngle;
    private int iLeftMotorOffset;
    private int iRightMotorOffset;

    private final int TURN_OFFSET = 200;
    private final double ANGLE_OFFSET = 5; // motor angle offset
    private final int STOP = 4;

    public ControlInformation()
    {
        bStartable = false;
        bMovable = false;
        bMoving = false;
        iDirection = STOP;
        dTiltAngle = 0.0;
        iLeftMotorOffset = 0;
        iRightMotorOffset = 0;
    }

    public synchronized boolean getStartable()
    {
        return bStartable;
    }

    public synchronized void setStartable(boolean value)
    {
        bStartable = value;
    }

    public synchronized boolean getMovable()
    {
        return bMovable;
    }

    public synchronized void setMovable(boolean value)
    {
        bMovable = value;
    }

    public synchronized boolean getMoving()
    {
        return bMoving;
    }

    public synchronized void setMoving(boolean value)
    {
        bMoving = value;
    }
```

```
public synchronized int getDirection()
{
    return iDirection;
}

public synchronized void setDirection(int value)
{
    iDirection = value;
}

public synchronized double getTiltAngle()
{
    return dTiltAngle;
}

public synchronized void setTiltAngle(double value)
{
    dTiltAngle = value;
}

public synchronized void increaseTiltAngle()
{
    dTiltAngle = dTiltAngle + ANGLE OFFSET;
}

public synchronized void decreaseTiltAngle()
{
    dTiltAngle = dTiltAngle - ANGLE OFFSET;
}

public synchronized int getLeftMotorOffset()
{
    return iLeftMotorOffset;
}

public synchronized void increaseLeftMotorOffset()
{
    iLeftMotorOffset = TURN_OFFSET;
}

public synchronized void decreaseLeftMotorOffset()
{
    iLeftMotorOffset = -1 * TURN_OFFSET;
}

public synchronized void resetLeftMotorOffset()
{
    iLeftMotorOffset = 0;
}


public synchronized int getRightMotorOffset()
{
    return iRightMotorOffset;
}


public synchronized void increaseRightMotorOffset()
{
    iRightMotorOffset = TURN_OFFSET;
}

public synchronized void decreaseRightMotorOffset()
{
    iRightMotorOffset = -1 * TURN_OFFSET;
}

public synchronized void resetRightMotorOffset()
{
    iRightMotorOffset = 0;
}
```

```
}
```

Communication.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public abstract class Communication extends Thread {
    ControlInformation ci;

    public Communication(ControlInformation value)
    {
        ci = value;
    }

    public abstract void setupConnection();
    public abstract void closeConnection();
    public abstract void exchangeInformation();

    @Override
    public abstract void run();

}
```

RobotServerComm.java Source Code

```
import java.io.*;
import lejos.nxt.*;
import lejos.nxt.comm.*;

public class RobotServerComm extends Communication {
    private BTConnection btc = null;

    private InputStream is = null;
    private OutputStream os = null;
    private DataOutputStream dos = null;
    private DataInputStream dis = null;

    public RobotServerComm(ControlInformation value)
    {
        super(value);

    }

    public void setupConnection()
    {
        LCD.drawString("Waiting for connection", 0, 0);
        btc = Bluetooth.waitForConnection();

        if (btc != null)
        {
            dos = btc.openDataOutputStream();
            dis = btc.openDataInputStream();

            ci.setStartable(true);
            exchangeInformation();
        }
    }
```

```
    public void exchangeInformation()
    {
        byte exchangeInfo;
        RobotReturnProtocol rr = new RobotReturnProtocol();

        while (true)
        {
            try {
                // receive
                while (ci.getMoving() == true)
                {
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException ex) { }
                }

                exchangeInfo = dis.readByte();
                rr.setReturnInfo(exchangeInfo);

                ci.setDirection(rr.getCode());
                if (rr.getCode() != RobotReturnProtocol.CODE_STOP) ci.setMoving(true);


                LCD.drawInt(rr.getCode(), 0, 0);
                LCD.drawInt(rr.getValue2(), 0, 1);

                // send
                while (ci.getMoving() == true)
                {
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException ex) { }
                }

                exchangeInfo = rr.getReturnInfo();
                dos.writeByte(exchangeInfo);
                dos.flush();

            } catch (IOException ex) { }

            try {
                Thread.sleep(100);
            } catch (InterruptedException ex) { }
        }
    }

    public void closeConnection()
    {
        try {
            dis.close();
            dos.close();
            btc.close();
            btc = null;

        } catch (IOException ex) {  }

        ci.setStartable(false);

    }

    public void run()
    {
        setupConnection();
        exchangeInformation();
    }

}
```

RobotPhoneComm.java Source Code

```
import lejos.nxt.*;
import lejos.nxt.comm.*;

public class RobotPhoneComm extends Communication {
    private BTConnection btc = null;

    public RobotPhoneComm(ControlInformation value)
    {
        super(value);

        setupConnection();
        exchangeInformation();
    }

    public void setupConnection()
    {
    }

    public void closeConnection()
    {
    }

    public void exchangeInformation()
    {
    }

    public void run()
    {
        LCD.drawInt(20, 2, 2);
    }


}
```

RobotReturnProtocol.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class RobotReturnProtocol {
    private int value2;
    private int code;

    public static int CODE_FORWARD = 0;
    public static int CODE_BACKWARD = 1;
    public static int CODE_TURNLEFT = 2;
    public static int CODE_TURNRIGHT = 3;
    public static int CODE_STOP = 4;

    public static int MINIMUM_COMMAND = 1;
    public static int MAXIMUM_COMMAND = 50;

    public RobotReturnProtocol()
    {
        value2 = 1;
        code = -1; // 0 = forward, 1 = backward, 2 = left, 3 = right
    }

    public int getValue2()
    {
        return value2;
    }

    public void setValue2(int value)
```

```
    {
        value2 = value;
    }

    public int getCode()
    {
        return code;
    }

    public void setCode(int value)
    {
        code = value;
    }

    public byte getReturnInfo()
    {
        if (code == CODE_FORWARD) return (byte) (value2 - 101); // value from -100 to -
51
        else if (code == CODE BACKWARD) return (byte) (value2 - 51); // value from -50
to -1
        else if (code == CODE_TURNLEFT) return (byte) (value2); // value from 1 to 50
        else if (code == CODE_TURNRIGHT) return (byte) (value2 + 50); // value from 51
to 100
        else return 120;
    }

    public void setReturnInfo(byte value)
    {
        if (value >= -100 && value <= -51)
        {
            code = CODE FORWARD;
            value2 = value + 101;
        }

        if (value >= -50 && value <= -1)
        {
            code = CODE_BACKWARD;
            value2 = value + 51;
        }

        if (value >= 1 && value <= 50)
        {
            code = CODE_TURNLEFT;
            value2 = value;
        }

        if (value >= 51 && value <= 100)
        {
            code = CODE_TURNRIGHT;
            value2 = value - 50;
        }

        if (value == 120)
        {
            code = CODE_STOP;
            value2 = 0;
        }
    }



}
```

Appendix A.2 – Server Application Source Code

The two files MediaPanel.java and StateHelper.java are cited from the Sun Microsystems' website [26].

The file DeviceInfo.java is cited from David Fischer's Java Programming Examples [27].

Main.java Source Code

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

/**
 *
 * @author Vu Ba Tien Dung
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws FileNotFoundException, IOException,
UnknownHostException, InterruptedException {
        BlockingQueue<Protocol> protocolQueue = new LinkedBlockingQueue<Protocol>();
        BlockingQueue<Protocol> doneProtocolQueue = new LinkedBlockingQueue<Protocol>();
        BlockingQueue<StreamClient> streamClientQueue = new
LinkedBlockingQueue<StreamClient>();

        ServerSideForm sf = new ServerSideForm(protocolQueue);
        sf.setLocation(10,50);
        sf.setVisible(true);

        Thread thServerClientComm = new Thread(new ServerClientComm(protocolQueue,
doneProtocolQueue, streamClientQueue, sf.getTextAreaNetwork(),
sf.getTextAreaController(), sf.getLabelController(), sf.getLabelStart()));
        Thread thServerRobotComm = new Thread(new ServerRobotComm(protocolQueue,
doneProtocolQueue, sf.getTextAreaNetwork(), sf.getTextAreaController()));
        Thread thStreamingServer = new Thread(new StreamingServer(streamClientQueue));
        Thread thLocalWebCam = new Thread(new LocalWebCam(streamClientQueue));

        thServerClientComm.start();
        thServerRobotComm.start();
        thStreamingServer.start();
        thLocalWebCam.start();


    }

}
```

LocalWebCam.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.media.MediaLocator;
import javax.swing.JFrame;

/**
 *
 * @author Vu Ba Tien Dung
```

```
 */
public class LocalWebCam  implements Runnable {

    private BlockingQueue<StreamClient> streamClientQueue;
    private int QUEUE_SIZE;

    public LocalWebCam(BlockingQueue<StreamClient> q)
    {
        streamClientQueue = q;
        QUEUE_SIZE = 0;
    }

    @SuppressWarnings({"static-access"})
    public void run()
    {
        JFrame mediaFrame = null;
        MediaPanel mediaPanel = null;

        String strIPAddress, url;
        int index;
        MediaLocator ml;

        while (true)
        {
            try {
                if (streamClientQueue.size() != QUEUE SIZE)
                {
                    QUEUE_SIZE = streamClientQueue.size();
                    if (mediaFrame != null) mediaFrame.dispose();
                    if (mediaPanel != null) mediaPanel.dispose();

                    strIPAddress = InetAddress.getLocalHost().toString();
                    index = strIPAddress.indexOf('/');
                    strIPAddress = strIPAddress.substring(index + 1);

                    url = "rtp://" + strIPAddress + ":50002/video";

                    ml = new MediaLocator(url);
                    if (ml != null) {
                        mediaFrame = new JFrame("Streaming Player");
                        mediaFrame.setLocation(700, 50);
                        mediaPanel = new MediaPanel(ml.toString());
                        mediaFrame.add(mediaPanel);
                        mediaFrame.setSize(300, 300);
                        mediaFrame.setVisible(true);
                    }
                }

                Thread.currentThread().sleep(10);

            }
            catch (UnknownHostException ex) {
                Logger.getLogger(LocalWebCam.class.getName()).log(Level.SEVERE, null,
ex);
            }            catch (InterruptedException ex) { }

        }

    }
}
```

MediaPanel.java [26] Source Code

```
import java.awt.BorderLayout;
import java.awt.Component;
import java.io.IOException;
import java.net.URL;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.MediaLocator;
```

```
import javax.media.NoPlayerException;
import javax.media.Player;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * MediaPanel.java
 *
 * Created on Apr 6, 2010, 4:05:12 AM
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class MediaPanel extends javax.swing.JPanel {

    Player mediaPlayer;

    /** Creates new form MediaPanel */
    public MediaPanel(URL mediaURL) {

        initComponents();

        setLayout( new BorderLayout() ); // use a BorderLayout

        try {

            // Create a JMF player to play the media specified in the URL:
            mediaPlayer = Manager.createRealizedPlayer( new MediaLocator(mediaURL) );

            // Get the components for the video and the playback controls:
            Component video = mediaPlayer.getVisualComponent();

            if ( video != null )
                add( video, BorderLayout.CENTER ); // add video component

            // Start the JMF player:
            mediaPlayer.start(); // start playing the media clip

        } // end try

        catch ( NoPlayerException noPlayerException ) {
            System.err.println( "No media player found" );
        } // end catch

        catch (CannotRealizeException ex) {
                ex.printStackTrace();
        } // end catch

        catch ( IOException iOException ) {
            System.err.println( "Error reading from the source" );
        } // end catch
    }

    public MediaPanel(String mediaURL) {
        initComponents();

        setLayout( new BorderLayout() ); // use a BorderLayout

        try {

            // Create a JMF player to play the media specified in the URL:
            mediaPlayer = Manager.createRealizedPlayer( new MediaLocator(mediaURL) );

            // Get the components for the video and the playback controls:
            Component video = mediaPlayer.getVisualComponent();

            if ( video != null )
                add( video, BorderLayout.CENTER ); // add video component
```

```
            // Start the JMF player:
            mediaPlayer.start(); // start playing the media clip

        } // end try

        catch ( NoPlayerException noPlayerException ) {
            System.err.println( "No media player found" );
        } // end catch

        catch (CannotRealizeException ex) {
                ex.printStackTrace();
        } // end catch

        catch ( IOException iOException ) {
            System.err.println( "Error reading from the source" );
        } // end catch
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
    }// </editor-fold>//GEN-END:initComponents

    public void dispose() {
        mediaPlayer.stop();
        mediaPlayer.close();

    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables

}
```

ServerRobotComm.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import lejos.pc.comm.*;
```

```
/**
 *
 * @author Vu Ba Tien Dung
 */
public class ServerRobotComm implements Runnable {
    private BlockingQueue<Protocol> protocolQueue;
    private BlockingQueue<Protocol> doneProtocolQueue;

    private javax.swing.JTextArea textareaNetwork;
    private javax.swing.JTextArea textareaController;

    private static final String ROBOT_NAME = "NXT";
    private static final String ROBOT_ADDRESS = "00:16:53:FF:FF:0A";
    private NXTComm nxtComm;
    private InputStream is = null;
    private OutputStream os = null;
    private DataOutputStream dos = null;
    private DataInputStream dis = null;

    public int CODE_FORWARD = 0;
    public int CODE_BACKWARD = 1;
    public int CODE_TURNLEFT = 2;
    public int CODE_TURNRIGHT = 3;
    public int CODE_STOP = 4;
    private final String FORWARD = "~FORW";
    private final String BACKWARD = "~BACK";
    private final String TURNLEFT = "~LEFT";
    private final String TURNRIGHT = "~RIGH";
    private final String STOP = "~STOP";

    private int commandIndex;

    public ServerRobotComm(BlockingQueue<Protocol> q1, BlockingQueue<Protocol> q2 ,
javax.swing.JTextArea network, javax.swing.JTextArea controller)
    {
        protocolQueue = q1;
        doneProtocolQueue = q2;

        textareaNetwork = network;
        textareaController = controller;

        commandIndex = 1; // from 1 to 50
    }

    @SuppressWarnings("static-access")
    public void run() {
        try
        {
            System.out.println("Start " + Thread.currentThread().getName());

            setupConnection();

            while (true)
            {
                if (protocolQueue.size() > 0)
                {
                    sendRobot(protocolQueue.take());
                }

                Thread.currentThread().sleep(10);
            }

            // Protocol value = protocolQueue.take();
            // protocolQueue.add();
        }
        catch (Exception e) {
            System.out.println(Thread.currentThread().getName() + " " + e.getMessage());
        }
    }

    public void sendRobot(Protocol value)
    {
        RobotReturnProtocol rr = new RobotReturnProtocol();
```

```
        if (value.getCommand().equals(FORWARD.substring(1))) rr.setCode(CODE FORWARD);
        if (value.getCommand().equals(BACKWARD.substring(1))) rr.setCode(CODE BACKWARD);
        if (value.getCommand().equals(TURNLEFT.substring(1))) rr.setCode(CODE_TURNLEFT);
        if (value.getCommand().equals(TURNRIGHT.substring(1)))
rr.setCode(CODE_TURNRIGHT);
        if (value.getCommand().equals(STOP.substring(1))) rr.setCode(CODE STOP);
        rr.setValue2(commandIndex);

        try {
            dos.writeByte(rr.getReturnInfo());
            dos.flush();
       } catch (IOException ex) {}

        System.out.println("send info = " + rr.getReturnInfo());

        if (value.getIPAddress() != null)
        {
            textareaController.setText( textareaController.getText() + "\nDeliver the
command " + value.getCommand() + " from " + value.getIPAddress() + " to robot." );
            textareaController.setCaretPosition(textareaController.getText().length());
        }
        else
        {
            textareaController.setText( textareaController.getText() + "\nDeliver the
command " + value.getCommand() + " from local server to robot." );
            textareaController.setCaretPosition(textareaController.getText().length());
        }

        byte receive;

        try {
            receive = dis.readByte();
            rr.setReturnInfo(receive);
            if (rr.getValue2() == commandIndex)
                if (value.getIPAddress() != null)
                {
                    textareaController.setText( textareaController.getText() + "\nThe
command " + value.getCommand() + " ( " + value.getValue1() + " ) from " +
value.getIPAddress() + " is done." );

textareaController.setCaretPosition(textareaController.getText().length());
                    doneProtocolQueue.add(value);
                }
                else
                {
                    textareaController.setText( textareaController.getText() + "\nThe
command " + value.getCommand() + " ( " + value.getValue1() + " ) from local server is
done." );

textareaController.setCaretPosition(textareaController.getText().length());
                }

            commandIndex++;
            if (commandIndex > RobotReturnProtocol.MAXIMUM_COMMAND)
                commandIndex = RobotReturnProtocol.MINIMUM_COMMAND;
        } catch (IOException ex) { }
    }

    private void setupConnection() throws NXTCommException
    {
        try
        {
            nxtComm = NXTCommFactory.createNXTComm(NXTCommFactory.BLUETOOTH);
        }
        catch (NXTCommException e) { }

        // robot information initialization
        NXTInfo nxtInfo = new NXTInfo();
        nxtInfo.name = ROBOT NAME;
        nxtInfo.deviceAddress = ROBOT_ADDRESS;

        boolean opened = false;
```

```
        while (!opened)
        {
            opened = nxtComm.open(nxtInfo);
        }

        is = nxtComm.getInputStream();
        os = nxtComm.getOutputStream();
             dos = new DataOutputStream(os);
             dis = new DataInputStream(is);
    }

    public void closeConnection()
    {
        try {
            dis.close();
            dos.close();
            is.close();
            os.close();
            nxtComm.close();
        } catch (IOException ex) { }
    }
}
```

RobotReturnProtocol.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class RobotReturnProtocol {
    private int value2;
    private int code;

    public static int CODE_FORWARD = 0;
    public static int CODE_BACKWARD = 1;
    public static int CODE_TURNLEFT = 2;
    public static int CODE_TURNRIGHT = 3;
    public static int CODE STOP = 4;

    public static int MINIMUM_COMMAND = 1;
    public static int MAXIMUM_COMMAND = 50;

    public RobotReturnProtocol()
    {
        value2 = 1;
        code = -1; // 0 = forward, 1 = backward, 2 = left, 3 = right
    }

    public int getValue2()
    {
        return value2;
    }

    public void setValue2(int value)
    {
        value2 = value;
    }

    public int getCode()
    {
        return code;
    }

    public void setCode(int value)
    {
```

```
        code = value;
    }

    public byte getReturnInfo()
    {
        if (code == CODE_FORWARD) return (byte) (value2 - 101); // value from -100 to -
51
        else if (code == CODE BACKWARD) return (byte) (value2 - 51); // value from -50
to -1
        else if (code == CODE_TURNLEFT) return (byte) (value2); // value from 1 to 50
        else if (code == CODE_TURNRIGHT) return (byte) (value2 + 50); // value from 51
to 100
        else return 120;
    }

    public void setReturnInfo(byte value)
    {
        if (value >= -100 && value <= -51)
        {
            code = CODE FORWARD;
            value2 = value + 101;
        }

        if (value >= -50 && value <= -1)
        {
            code = CODE BACKWARD;
            value2 = value + 51;
        }

        if (value >= 1 && value <= 50)
        {
            code = CODE_TURNLEFT;
            value2 = value;
        }

        if (value >= 51 && value <= 100)
        {
            code = CODE_TURNRIGHT;
            value2 = value - 50;
        }

        if (value == 120)
        {
            code = CODE_STOP;
            value2 = 0;
        }
    }



}
```

ServerClientComm.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.util.concurrent.BlockingQueue;
import java.net.*;
import java.io.*;

/**
 *
 * @author Vu Ba Tien Dung
 */
public class ServerClientComm implements Runnable {
    private final int MIN_PORT = 50000;
    private int port_index;
```

```
    private BlockingQueue<Protocol> protocolQueue;
    private BlockingQueue<Protocol> doneProtocolQueue;
    private BlockingQueue<StreamClient> streamClientQueue;

    private javax.swing.JTextArea textareaNetwork;
    private javax.swing.JTextArea textareaController;
    private javax.swing.JLabel labelController;
    private javax.swing.JLabel labelStart;

    private ServerSocket serverSocket;
    private boolean listening;

    public ServerClientComm(BlockingQueue<Protocol> q1, BlockingQueue<Protocol> q2,
BlockingQueue<StreamClient> q3, javax.swing.JTextArea network, javax.swing.JTextArea
controller, javax.swing.JLabel label1, javax.swing.JLabel label2) throws
UnknownHostException, InterruptedException
    {
        protocolQueue = q1;
        doneProtocolQueue = q2;
        streamClientQueue = q3;

        StreamClient value = new StreamClient();
        port index = 2;

        value.setClientAddress(InetAddress.getLocalHost());
        value.setPortNumber(MIN_PORT + port_index);
        port_index = port_index + 2;
        streamClientQueue.put(value);

        serverSocket = null;
        listening = false;

        textareaNetwork = network;
        textareaController = controller;
        labelController = label1;
        labelStart = label2;
    }

    @SuppressWarnings("static-access")
    public void run() {
        try
        {
            System.out.println("Start " + Thread.currentThread().getName());

            // Protocol value = protocolQueue.take();
            // protocolQueue.put();
            while (true)
            {
                while (!listening)
                {
                    if (labelStart.getText().indexOf("Do not listen") != -1)
                    { }
                    else
                        listening = true;

                    Thread.currentThread().sleep(10);
                }

                try
                {
                    serverSocket = new ServerSocket(33333);
                } catch (IOException e) {
                    System.err.println("Could not listen on port: 33333.");
                    System.exit(-1);
                }

                while (listening)
                {
                    new Thread(new ServerClientCommThread(protocolQueue,
doneProtocolQueue, streamClientQueue, serverSocket.accept(), textareaNetwork,
textareaController, labelController, port_index+=2)).start();
                    if (labelStart.getText().indexOf("Do not listen") != -1)
```

```
                        listening = false;

                    Thread.currentThread().sleep(10);
                }

                serverSocket.close();
            }
        }
        catch (Exception e) {
            System.out.println(Thread.currentThread().getName() + " " + e.getMessage());
        }
    }

    public boolean isListening()
    {
        return listening;
    }

    public void setListening(boolean value)
    {
        listening = value;
    }

}
```

ServerClientCommThread.java Source Code

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.StringTokenizer;
import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Vu Ba Tien Dung
 */
public class ServerClientCommThread implements Runnable {
    private final int MIN_PORT = 50000;
    private int port_index;

    private Socket socketClient;
    private BlockingQueue<Protocol> protocolQueue;
    private BlockingQueue<Protocol> doneProtocolQueue;
    private BlockingQueue<StreamClient> streamClientQueue;
    private int QUEUE SIZE;

    private javax.swing.JTextArea textareaNetwork;
    private javax.swing.JTextArea textareaController;
    private javax.swing.JLabel labelController;

    private UserDatabase userdata;
    private int userPriviledge;
    private int rtpPort;
    private boolean bStreaming;

    private final String BYEBYE = "~BYET";
    private final String PASSWORD = "~PASS";
    private final String SUCCESS = "~SUCC";
    private final String FORWARD = "~FORW";
    private final String BACKWARD = "~BACK";
    private final String TURNLEFT = "~LEFT";
    private final String TURNRIGHT = "~RIGH";
    private final String STOP = "~STOP";
```

```
    private final String USERNAME = "~USER";
    private final String RTP INIT = "~RTPI";
    private final String RTP CHANGE = "~RTPC";
    private final String DONE = "~DONE";

    private final char delimeter = '~';

    private final int PRIVILEDGE HIGHEST = 15;
    private final int PRIVILEDGE NORMAL = 5;
    private final int PRIVILEDGE_LIMITED = 2;


    public ServerClientCommThread(BlockingQueue<Protocol> q1, BlockingQueue<Protocol>
q2, BlockingQueue<StreamClient> q3, Socket client, javax.swing.JTextArea network,
javax.swing.JTextArea controller, javax.swing.JLabel label, int port_index) throws
FileNotFoundException, IOException
    {
        socketClient = client;
        this.port index = port index;
        bStreaming = false;
        QUEUE_SIZE = 0;

        protocolQueue = q1;
        doneProtocolQueue = q2;
        streamClientQueue = q3;

        textareaNetwork = network;
        textareaController = controller;

        userPriviledge = 0;
        rtpPort = 0;
        labelController = label;
    }

    @SuppressWarnings({"static-access"})
    public void run()
    {
        PrintWriter out = null;
        BufferedReader in = null;
        String inputString = null;
        String outputString = null;

        if
(!socketClient.getInetAddress().toString().equals(socketClient.getLocalAddress().toStrin
g()))
        {
            textareaNetwork.setText( textareaNetwork.getText() + "\nClient on " +
socketClient.getInetAddress() + " connecting to the server." );
            textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
        }

        try {
            out = new PrintWriter(socketClient.getOutputStream(), true);
            in = new BufferedReader(new
InputStreamReader(socketClient.getInputStream()));

            while (true) {
               // Processing the client request
               if (in.ready())
                    inputString = in.readLine();
               if (inputString != null)
               {
                    // BYEBYE command
                    if (inputString.startsWith(BYEBYE)) {
                        textareaNetwork.setText( textareaNetwork.getText() + "\nClient
on " + socketClient.getInetAddress() + " send BYE message to the server." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                        if (userPriviledge >= PRIVILEDGE NORMAL)
                            labelController.setText("Listening the commands from local
server");
                        break;
                    }
```

```
                    // PASSWORD command
                    if (inputString.startsWith(PASSWORD)) {

                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String strCommand = st.nextToken();
                        String strUsername = st.nextToken().trim();
                        String strPassword = st.nextToken().trim();
                        userdata = new UserDatabase("userdatabase.txt");

                        int i;
                        for (i = 0; i < userdata.rowCount(); i++) {
                            if (strUsername.equals(userdata.getUser(i).getUsername())) {
                                if
(strPassword.equals(userdata.getUser(i).getPassword())) {
                                    if (userdata.getUser(i).getRemoteMonitor() &&
userdata.getUser(i).getRemoteControl())
                                    {
                                        if (labelController.getText().indexOf("local
server") != -1)
                                        {
                                            userPriviledge = PRIVILEDGE_HIGHEST;
                                            outputString = SUCCESS + delimeter +
Integer.toString(PRIVILEDGE_HIGHEST) + delimeter + Integer.toString(0);
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nClient on " + socketClient.getInetAddress() + " get
highest control priviledge to the server." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                                            labelController.setText("Listening the
commands from " + socketClient.getInetAddress() + " (Username: " +
userdata.getUser(i).getUsername() + ")");

                                        }
                                        else
                                        {
                                            userPriviledge = PRIVILEDGE_LIMITED;
                                            outputString = SUCCESS + delimeter +
Integer.toString(PRIVILEDGE_LIMITED) + delimeter + Integer.toString(0);
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nClient on " + socketClient.getInetAddress() + " get
monitor permission." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                                        }
                                    }
                                    else if (!userdata.getUser(i).getRemoteMonitor() &&
userdata.getUser(i).getRemoteControl()) {
                                        if (labelController.getText().indexOf("local
server") != -1)
                                        {
                                            userPriviledge = PRIVILEDGE_NORMAL;
                                            outputString = SUCCESS + delimeter +
Integer.toString(PRIVILEDGE_NORMAL) + delimeter + Integer.toString(0);
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nClient on " + socketClient.getInetAddress() + " get
control priviledge to the server." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                                            labelController.setText("Listening the
commands from " + socketClient.getInetAddress() + " (Username: " +
userdata.getUser(i).getUsername() + ")");
                                        }
                                        else
                                        {
                                            userPriviledge = PRIVILEDGE_LIMITED;
                                            outputString = SUCCESS + delimeter +
Integer.toString(PRIVILEDGE_LIMITED) + delimeter + Integer.toString(0);
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nClient on " + socketClient.getInetAddress() + " get
monitor permission." );
```

```
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nAll the video streaming will be restarted in some
seconds." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                                            }
                                        } else if (userdata.getUser(i).getRemoteMonitor() &&
!userdata.getUser(i).getRemoteControl()) {
                                            userPriviledge = PRIVILEDGE LIMITED;
                                            outputString = SUCCESS + delimeter +
Integer.toString(PRIVILEDGE_LIMITED) + delimeter + Integer.toString(0);
                                            textareaNetwork.setText(
textareaNetwork.getText() + "\nClient on " + socketClient.getInetAddress() + " get
monitor permission." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                                        }

                                        // exit the loop after finding the correct answer
                                        break;
                                    }
                                }
                            }

                            if (i == userdata.rowCount())
                            {
                                outputString = BYEBYE + delimeter + Integer.toString(0) +
delimeter + Integer.toString(0) ;
                                textareaNetwork.setText( textareaNetwork.getText() +
"\nClient on " + socketClient.getInetAddress() + " failed in authentication." );

textareaNetwork.setCaretPosition(textareaNetwork.getText().length());
                            }

                            out.println(outputString);
                        }

                        // FORWARD command
                        if (inputString.startsWith(FORWARD) && userPriviledge >=
PRIVILEDGE NORMAL)
                        {
                            Protocol value = new Protocol();
                            StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                            String strCommand = st.nextToken();
                            String strIndex = st.nextToken();
                            String strExtra = st.nextToken();

                            value.setIPAddress(socketClient.getInetAddress());
                            value.setCommand(strCommand);
                            value.setValue1(strIndex);
                            value.setValue2(strExtra);

                            protocolQueue.add(value);
                            textareaController.setText( textareaController.getText() +
"\nClient on " + socketClient.getInetAddress() + " send FORWARD command." );

textareaController.setCaretPosition(textareaController.getText().length());
                        }

                        // BACKWARD command
                        if (inputString.startsWith(BACKWARD) && userPriviledge >=
PRIVILEDGE NORMAL)
                        {
                            Protocol value = new Protocol();
                            StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                            String strCommand = st.nextToken();
                            String strIndex = st.nextToken();
                            String strExtra = st.nextToken();

                            value.setIPAddress(socketClient.getInetAddress());
                            value.setCommand(strCommand);
```

```
                        value.setValue1(strIndex);
                        value.setValue2(strExtra);

                        protocolQueue.add(value);
                        textareaController.setText( textareaController.getText() +
"\nClient on " + socketClient.getInetAddress() + " send BACKWARD command." );

textareaController.setCaretPosition(textareaController.getText().length());
                    }

                    // LEFT command
                    if (inputString.startsWith(TURNLEFT) && userPriviledge >=
PRIVILEDGE_NORMAL)
                    {
                        Protocol value = new Protocol();
                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String strCommand = st.nextToken();
                        String strIndex = st.nextToken();
                        String strExtra = st.nextToken();

                        value.setIPAddress(socketClient.getInetAddress());
                        value.setCommand(strCommand);
                        value.setValue1(strIndex);
                        value.setValue2(strExtra);

                        protocolQueue.add(value);
                        textareaController.setText( textareaController.getText() +
"\nClient on " + socketClient.getInetAddress() + " send TURNLEFT command." );

textareaController.setCaretPosition(textareaController.getText().length());
                    }

                    // RIGHT command
                    if (inputString.startsWith(TURNRIGHT) && userPriviledge >=
PRIVILEDGE_NORMAL)
                    {
                        Protocol value = new Protocol();
                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String strCommand = st.nextToken();
                        String strIndex = st.nextToken();
                        String strExtra = st.nextToken();

                        value.setIPAddress(socketClient.getInetAddress());
                        value.setCommand(strCommand);
                        value.setValue1(strIndex);
                        value.setValue2(strExtra);

                        protocolQueue.add(value);
                        textareaController.setText( textareaController.getText() +
"\nClient on " + socketClient.getInetAddress() + " send TURNRIGHT command." );

textareaController.setCaretPosition(textareaController.getText().length());
                    }

                    // STOP command
                    if (inputString.startsWith(STOP) && userPriviledge >=
PRIVILEDGE_NORMAL)
                    {
                        Protocol value = new Protocol();
                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String strCommand = st.nextToken();
                        String strIndex = st.nextToken();
                        String strExtra = st.nextToken();

                        value.setIPAddress(socketClient.getInetAddress());
                        value.setCommand(strCommand);
                        value.setValue1(strIndex);
                        value.setValue2(strExtra);

                        protocolQueue.clear();
```

```
                        protocolQueue.add(value);
                        textareaController.setText( textareaController.getText() +
"\nClient on " + socketClient.getInetAddress() + " send TURNLEFT command." );

textareaController.setCaretPosition(textareaController.getText().length());
                    }

                    inputString = null;
                }

                // Processing the server response
                if (userPriviledge == 0)
                {
                    userPriviledge = 1;
                    outputString = USERNAME + delimeter + Integer.toString(0) +
delimeter + Integer.toString(0);
                    out.println(outputString);

                }
                else if ((rtpPort == 0) && (userPriviledge == PRIVILEDGE HIGHEST ||
userPriviledge == PRIVILEDGE_LIMITED))
                {
                    rtpPort = MIN PORT + port index;

                    StreamClient value = new StreamClient();
                    value.setClientAddress(socketClient.getInetAddress());
                    value.setPortNumber(rtpPort);

                    streamClientQueue.add(value);
                    bStreaming = true;
                    QUEUE SIZE = streamClientQueue.size();

                    try {
                        Thread.currentThread().sleep(10);
                    } catch (InterruptedException ex) {}

                    outputString = RTP_INIT + delimeter + Integer.toString(rtpPort) +
delimeter + Integer.toString(0);
                    out.println(outputString);
                }

                if (bStreaming)
                    if (QUEUE_SIZE != streamClientQueue.size())
                    {
                        QUEUE_SIZE = streamClientQueue.size();
                        outputString = RTP CHANGE + delimeter + Integer.toString(0) +
delimeter + Integer.toString(0);
                    }

               if (doneProtocolQueue.peek() != null)
                   if (doneProtocolQueue.peek().getIPAddress() ==
socketClient.getInetAddress())
                   {
                    try {
                        Protocol value = doneProtocolQueue.take();
                        outputString = DONE + delimeter + value.getCommand() + delimeter
+ value.getValue1();
                    } catch (InterruptedException ex) {

Logger.getLogger(ServerClientCommThread.class.getName()).log(Level.SEVERE, null, ex);
                    }

                   }

                // Take a rest a bit so the other thread can run
                try {
                    Thread.currentThread().sleep(20);
                } catch (InterruptedException ex) {}
            }

            out.close();
            in.close();
            socketClient.close();
```

```
        }
        catch (IOException ex) { }
    }
}
```

Protocol.java Source Code

```
import java.net.InetAddress;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class Protocol {
    private InetAddress IPAddress;
    private String strCommand;
    private String strValue1;
    private String strValue2;

    public Protocol()
    {
        IPAddress = null;
        strCommand = null;
        strValue1 = null;
        strValue2 = null;
    }

    public void setIPAddress(InetAddress value)
    {
        IPAddress = value;
    }

    public void setCommand(String value)
    {
        strCommand = value;
    }

    public void setValue1(String value)
    {
        strValue1 = value;
    }

    public void setValue2(String value)
    {
        strValue2 = value;
    }

    public InetAddress getIPAddress()
    {
        return IPAddress;
    }

    public String getCommand()
    {
        return strCommand;
    }

    public String getValue1()
    {
        return strValue1;
    }

    public String getValue2()
    {
```

```
        return strValue2;
    }
}
```

ServerSideForm.java Source Code

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.TableModel;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * ServerSideForm.java
 *
 * Created on Apr 5, 2010, 8:13:42 PM
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class ServerSideForm extends javax.swing.JFrame {
    private final String FORWARD = "FORW";
    private final String BACKWARD = "BACK";
    private final String TURNLEFT = "LEFT";
    private final String TURNRIGHT = "RIGH";
    private final String STOP = "STOP";
    private int command_index;

    /** Creates new form ServerSideForm */
    public ServerSideForm(BlockingQueue<Protocol> q) throws FileNotFoundException,
IOException {
        initComponents();
        initUsernameDatabase();

        protocolQueue = q;
        command index = 0;
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        panelMain = new javax.swing.JPanel();
        tabbedMain = new javax.swing.JTabbedPane();
        panelNetwork = new javax.swing.JPanel();
        buttonStart = new javax.swing.JButton();
        labelStart = new javax.swing.JLabel();
        scrollpaneNetwork = new javax.swing.JScrollPane();
        textareaNetwork = new javax.swing.JTextArea();
        panelController = new javax.swing.JPanel();
        scrollpanelController = new javax.swing.JScrollPane();
```

```
        textareaController = new javax.swing.JTextArea();
        labelController = new javax.swing.JLabel();
        panelLocalControll = new javax.swing.JPanel();
        buttonUp = new javax.swing.JButton();
        buttonRight = new javax.swing.JButton();
        buttonStop = new javax.swing.JButton();
        buttonLeft = new javax.swing.JButton();
        buttonDown = new javax.swing.JButton();
        panelUser = new javax.swing.JPanel();
        scrollpanelUser = new javax.swing.JScrollPane();
        tableUser = new javax.swing.JTable();
        buttonAdd = new javax.swing.JButton();
        buttonEdit = new javax.swing.JButton();
        buttonDel = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Robot Controller Server");
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT CURSOR));
        setName("mainFrame"); // NOI18N

        tabbedMain.setName("Network"); // NOI18N

        buttonStart.setText("Start Server");
        buttonStart.setToolTipText("start and stop the service");
        buttonStart.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonStartActionPerformed(evt);
            }
        });

        labelStart.setForeground(new java.awt.Color(255, 0, 0));
        labelStart.setText("Do not listen to outside connections");


scrollpaneNetwork.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL SCROLLBAR
ALWAYS);

scrollpaneNetwork.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICAL_SC
ROLLBAR_ALWAYS);

        textareaNetwork.setColumns(20);
        textareaNetwork.setEditable(false);
        textareaNetwork.setRows(5);
        scrollpaneNetwork.setViewportView(textareaNetwork);

        javax.swing.GroupLayout panelNetworkLayout = new
javax.swing.GroupLayout(panelNetwork);
        panelNetwork.setLayout(panelNetworkLayout);
        panelNetworkLayout.setHorizontalGroup(

panelNetworkLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelNetworkLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panelNetworkLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                    .addComponent(scrollpaneNetwork,
javax.swing.GroupLayout.PREFERRED SIZE, 632, javax.swing.GroupLayout.PREFERRED SIZE)
                    .addGroup(panelNetworkLayout.createSequentialGroup()
                        .addComponent(buttonStart)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(labelStart)))
                .addContainerGap())
        );
        panelNetworkLayout.setVerticalGroup(

panelNetworkLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelNetworkLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panelNetworkLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASEL
INE)
```

```
                    .addComponent(buttonStart)
                    .addComponent(labelStart))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(scrollpaneNetwork, javax.swing.GroupLayout.PREFERRED_SIZE,
268, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        tabbedMain.addTab("Network", panelNetwork);


scrollpanelController.setVerticalScrollBarPolicy(javax.swing.JScrollPane.VERTICAL_SCROLL
BAR ALWAYS);

scrollpanelController.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTICA
L_SCROLLBAR_ALWAYS);

        textareaController.setColumns(20);
        textareaController.setEditable(false);
        textareaController.setRows(5);
        scrollpanelController.setViewportView(textareaController);

        labelController.setForeground(new java.awt.Color(255, 0, 0));
        labelController.setText("Listening the commands from local server");

        panelLocalControll.setBorder(javax.swing.BorderFactory.createTitledBorder("Local
Controlls"));

        buttonUp.setBackground(new java.awt.Color(255, 255, 255));
        buttonUp.setIcon(new javax.swing.ImageIcon(getClass().getResource("/up.JPG")));
// NOI18N
        buttonUp.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonUpActionPerformed(evt);
            }
        });

        buttonRight.setBackground(new java.awt.Color(255, 255, 255));
        buttonRight.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/right.JPG"))); // NOI18N
        buttonRight.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonRightActionPerformed(evt);
            }
        });

        buttonStop.setBackground(new java.awt.Color(255, 255, 255));
        buttonStop.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/stop.JPG"))); // NOI18N
        buttonStop.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonStopActionPerformed(evt);
            }
        });

        buttonLeft.setBackground(new java.awt.Color(255, 255, 255));
        buttonLeft.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/left.JPG"))); // NOI18N
        buttonLeft.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonLeftActionPerformed(evt);
            }
        });

        buttonDown.setBackground(new java.awt.Color(255, 255, 255));
        buttonDown.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/down.JPG"))); // NOI18N
        buttonDown.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonDownActionPerformed(evt);
            }
        });
```

```
        javax.swing.GroupLayout panelLocalControllLayout = new
javax.swing.GroupLayout(panelLocalControll);
        panelLocalControll.setLayout(panelLocalControllLayout);
        panelLocalControllLayout.setHorizontalGroup(

panelLocalControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelLocalControllLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(buttonLeft, javax.swing.GroupLayout.PREFERRED SIZE, 66,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(panelLocalControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addComponent(buttonDown, javax.swing.GroupLayout.PREFERRED_SIZE,
66, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(panelLocalControllLayout.createSequentialGroup()

.addGroup(panelLocalControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                            .addComponent(buttonUp,
javax.swing.GroupLayout.PREFERRED_SIZE, 66, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(buttonStop,
javax.swing.GroupLayout.PREFERRED SIZE, 66, Short.MAX VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(buttonRight, 0, 66, Short.MAX_VALUE)))
                .addContainerGap())
        );
        panelLocalControllLayout.setVerticalGroup(

panelLocalControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelLocalControllLayout.createSequentialGroup()
                .addComponent(buttonUp, javax.swing.GroupLayout.PREFERRED SIZE, 66,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(panelLocalControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
                    .addComponent(buttonLeft, javax.swing.GroupLayout.PREFERRED SIZE,
66, javax.swing.GroupLayout.PREFERRED SIZE)
                    .addComponent(buttonRight, javax.swing.GroupLayout.PREFERRED SIZE,
66, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(buttonStop, javax.swing.GroupLayout.PREFERRED_SIZE,
66, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(buttonDown, javax.swing.GroupLayout.PREFERRED SIZE, 66,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout panelControllerLayout = new
javax.swing.GroupLayout(panelController);
        panelController.setLayout(panelControllerLayout);
        panelControllerLayout.setHorizontalGroup(

panelControllerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelControllerLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(panelControllerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelControllerLayout.createSequentialGroup()
                        .addComponent(scrollpanelController,
javax.swing.GroupLayout.DEFAULT_SIZE, 380, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(panelLocalControll,
javax.swing.GroupLayout.PREFERRED SIZE, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(labelController))
                .addContainerGap())
```

```
        );
        panelControllerLayout.setVerticalGroup(

panelControllerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelControllerLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(labelController)

.addGroup(panelControllerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(panelControllerLayout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(panelLocalControll,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(panelControllerLayout.createSequentialGroup()
                        .addGap(11, 11, 11)
                        .addComponent(scrollpanelController,
javax.swing.GroupLayout.DEFAULT SIZE, 272, Short.MAX VALUE)))
                .addContainerGap())
        );

        tabbedMain.addTab("Controller", panelController);

        tableUser.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null}
            },
            new String [] {
                "Username", "Password", "Remote Monitor", "Remote Control"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class, java.lang.Boolean.class,
java.lang.Boolean.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        scrollpanelUser.setViewportView(tableUser);
        tableUser.getColumnModel().getColumn(0).setResizable(false);
        tableUser.getColumnModel().getColumn(1).setResizable(false);
        tableUser.getColumnModel().getColumn(3).setResizable(false);

        buttonAdd.setLabel("Add User");
        buttonAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonAddActionPerformed(evt);
            }
        });

        buttonEdit.setText("Edit User");
        buttonEdit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonEditActionPerformed(evt);
            }
        });

        buttonDel.setText("Delete User");
        buttonDel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonDelActionPerformed(evt);
            }
```

```
        });

        javax.swing.GroupLayout panelUserLayout = new
javax.swing.GroupLayout(panelUser);
        panelUser.setLayout(panelUserLayout);
        panelUserLayout.setHorizontalGroup(

panelUserLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelUserLayout.createSequentialGroup()

.addGroup(panelUserLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(panelUserLayout.createSequentialGroup()
                        .addGap(79, 79, 79)
                        .addComponent(buttonAdd)
                        .addGap(57, 57, 57)
                        .addComponent(buttonEdit)
                        .addGap(60, 60, 60)
                        .addComponent(buttonDel))
                    .addGroup(panelUserLayout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(scrollpanelUser,
javax.swing.GroupLayout.DEFAULT_SIZE, 632, Short.MAX_VALUE)))
                .addContainerGap())
        );
        panelUserLayout.setVerticalGroup(

panelUserLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelUserLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(scrollpanelUser, javax.swing.GroupLayout.DEFAULT SIZE,
263, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(panelUserLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
                    .addComponent(buttonAdd)
                    .addComponent(buttonEdit)
                    .addComponent(buttonDel))
                .addContainerGap())
        );

        tabbedMain.addTab("User Databases", panelUser);

        javax.swing.GroupLayout panelMainLayout = new
javax.swing.GroupLayout(panelMain);
        panelMain.setLayout(panelMainLayout);
        panelMainLayout.setHorizontalGroup(

panelMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelMainLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(tabbedMain, javax.swing.GroupLayout.DEFAULT SIZE, 657,
Short.MAX_VALUE))
        );
        panelMainLayout.setVerticalGroup(

panelMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
panelMainLayout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(tabbedMain, javax.swing.GroupLayout.PREFERRED_SIZE, 344,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addContainerGap())
        );

        tabbedMain.getAccessibleContext().setAccessibleName("Network");
        tabbedMain.getAccessibleContext().setAccessibleParent(this);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addComponent(panelMain, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.DEFAULT SIZE, Short.MAX VALUE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(panelMain, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.DEFAULT SIZE, Short.MAX VALUE)
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void buttonStartActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonStartActionPerformed
        if (buttonStart.getText().equals("Start Server"))
        {
            buttonStart.setText("Stop Server");
            labelStart.setText("Listening to outside connections on port 33333");
        }
        else
        {
            buttonStart.setText("Start Server");
            labelStart.setText("Do not listen to outside connections");

            try {
                Socket tempSocket = new Socket(InetAddress.getLocalHost(), 33333);
                tempSocket.close();
            } catch (IOException ex) {
                Logger.getLogger(ServerSideForm.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }

    }//GEN-LAST:event buttonStartActionPerformed

    private void buttonLeftActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonLeftActionPerformed
        if (labelController.getText().indexOf("local server") != -1)
        {

            Protocol value = new Protocol();
            value.setCommand(TURNLEFT);
            value.setValue1(Integer.toString(command_index++));

            protocolQueue.add(value);

        }
    }//GEN-LAST:event_buttonLeftActionPerformed

    private void buttonDownActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonDownActionPerformed
        if (labelController.getText().indexOf("local server") != -1)
        {

            Protocol value = new Protocol();
            value.setCommand(BACKWARD);
            value.setValue1(Integer.toString(command_index++));

            protocolQueue.add(value);
        }

    }//GEN-LAST:event buttonDownActionPerformed

    private void buttonAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonAddActionPerformed
        User user = new User();
        userdata.addUser(user);

        EditUserDialog edit = new EditUserDialog(this, true,
userdata.getUser(userdata.rowCount() - 1));
        edit.show();
```

```
        tableUser.setModel(new javax.swing.table.DefaultTableModel(
            userdata.getObject(),
            new String [] {
                "Username", "Password", "Remote Monitor", "Remote Control"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class, java.lang.Boolean.class,
java.lang.Boolean.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });

        try {
            userdata.saveToFile();
        } catch (IOException ex) {
            Logger.getLogger(ServerSideForm.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }//GEN-LAST:event buttonAddActionPerformed

    private void buttonEditActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonEditActionPerformed
        // TODO add your handling code here:
        if (tableUser.getSelectedRow() < userdata.rowCount() &&
tableUser.getSelectedRow() >= 0)
        {
            EditUserDialog edit = new EditUserDialog(this, true,
userdata.getUser(tableUser.getSelectedRow()));
            edit.show();

            tableUser.setModel(new javax.swing.table.DefaultTableModel(
                userdata.getObject(),
                new String [] {
                    "Username", "Password", "Remote Monitor", "Remote Control"
                }
            ) {
                Class[] types = new Class [] {
                    java.lang.String.class, java.lang.String.class,
java.lang.Boolean.class, java.lang.Boolean.class
                };
                boolean[] canEdit = new boolean [] {
                    false, false, false, false
                };

                public Class getColumnClass(int columnIndex) {
                    return types [columnIndex];
                }

                public boolean isCellEditable(int rowIndex, int columnIndex) {
                    return canEdit [columnIndex];
                }
            });

            try {
                userdata.saveToFile();
            } catch (IOException ex) {
                Logger.getLogger(ServerSideForm.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }//GEN-LAST:event_buttonEditActionPerformed
```

```
    private void buttonDelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonDelActionPerformed
        // TODO add your handling code here:
        if (tableUser.getSelectedRow() < userdata.rowCount() &&
tableUser.getSelectedRow() >= 0)
        {
            userdata.removeUser(tableUser.getSelectedRow());

            tableUser.setModel(new javax.swing.table.DefaultTableModel(
                userdata.getObject(),
                new String [] {
                    "Username", "Password", "Remote Monitor", "Remote Control"
                }
            ) {
                Class[] types = new Class [] {
                    java.lang.String.class, java.lang.String.class,
java.lang.Boolean.class, java.lang.Boolean.class
                };
                boolean[] canEdit = new boolean [] {
                    false, false, false, false
                };

                public Class getColumnClass(int columnIndex) {
                    return types [columnIndex];
                }

                public boolean isCellEditable(int rowIndex, int columnIndex) {
                    return canEdit [columnIndex];
                }
            });

            try {
                userdata.saveToFile();
            } catch (IOException ex) {
                Logger.getLogger(ServerSideForm.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }//GEN-LAST:event buttonDelActionPerformed

    private void buttonUpActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonUpActionPerformed
        if (labelController.getText().indexOf("local server") != -1)
        {
            Protocol value = new Protocol();
            value.setCommand(FORWARD);
            value.setValue1(Integer.toString(command index++));

            protocolQueue.add(value);
        }

    }//GEN-LAST:event buttonUpActionPerformed

    private void buttonRightActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonRightActionPerformed
        if (labelController.getText().indexOf("local server") != -1)
        {

            Protocol value = new Protocol();
            value.setCommand(TURNRIGHT);
            value.setValue1(Integer.toString(command_index++));

            protocolQueue.add(value);
        }
    }//GEN-LAST:event_buttonRightActionPerformed

    private void buttonStopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonStopActionPerformed
        if (labelController.getText().indexOf("local server") != -1)
        {

            Protocol value = new Protocol();
```

```
            value.setCommand(STOP);
            value.setValue1(Integer.toString(command index++));

            protocolQueue.clear();
            protocolQueue.add(value);
        }
    }//GEN-LAST:event buttonStopActionPerformed

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton buttonAdd;
    private javax.swing.JButton buttonDel;
    private javax.swing.JButton buttonDown;
    private javax.swing.JButton buttonEdit;
    private javax.swing.JButton buttonLeft;
    private javax.swing.JButton buttonRight;
    private javax.swing.JButton buttonStart;
    private javax.swing.JButton buttonStop;
    private javax.swing.JButton buttonUp;
    private javax.swing.JLabel labelController;
    private javax.swing.JLabel labelStart;
    private javax.swing.JPanel panelController;
    private javax.swing.JPanel panelLocalControll;
    private javax.swing.JPanel panelMain;
    private javax.swing.JPanel panelNetwork;
    private javax.swing.JPanel panelUser;
    private javax.swing.JScrollPane scrollpaneNetwork;
    private javax.swing.JScrollPane scrollpanelController;
    private javax.swing.JScrollPane scrollpanelUser;
    private javax.swing.JTabbedPane tabbedMain;
    private javax.swing.JTable tableUser;
    private javax.swing.JTextArea textareaController;
    private javax.swing.JTextArea textareaNetwork;
    // End of variables declaration//GEN-END:variables

    private UserDatabase userdata;
    private BlockingQueue<Protocol> protocolQueue;

    private void initUsernameDatabase() throws FileNotFoundException, IOException {
        userdata = new UserDatabase("userdatabase.txt");

        tableUser.setModel(new javax.swing.table.DefaultTableModel(
            userdata.getObject(),
            new String [] {
                "Username", "Password", "Remote Monitor", "Remote Control"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class, java.lang.Boolean.class,
java.lang.Boolean.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
    }

    public javax.swing.JTextArea getTextAreaNetwork()
    {
        return textareaNetwork;
    }

    public javax.swing.JTextArea getTextAreaController()
    {
        return textareaController;
    }
```

```
    public javax.swing.JLabel getLabelController()
    {
        return labelController;
    }

    public javax.swing.JLabel getLabelStart()
    {
        return labelStart;
    }
}
```

EditUserDialog.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * EditUserDialog.java
 *
 * Created on Apr 6, 2010, 2:49:01 AM
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class EditUserDialog extends javax.swing.JDialog {

    /** Creates new form EditUserDialog */
    public EditUserDialog(java.awt.Frame parent, boolean modal, User user) {
        super(parent, modal);
        initComponents();
        dialogUser = user;
        initDialogField();

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jCheckBox1 = new javax.swing.JCheckBox();
        jCheckBox2 = new javax.swing.JCheckBox();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("User Information Dialog");
        setAlwaysOnTop(true);

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Input the new
information of the user:"));

        jLabel1.setText("Username:");
```

```
        jLabel2.setText("Password:");

        jLabel3.setText("Remote Monitor");

        jLabel4.setText("Remote Control");

        jButton1.setText("OK");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Exit");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jCheckBox1))
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jButton1)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jCheckBox2)))
                            .addGap(26, 26, 26)
                            .addComponent(jButton2))
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2))
                        .addGap(34, 34, 34)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(jTextField2)
                            .addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT SIZE, 136, Short.MAX VALUE))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT SIZE, Short.MAX VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
```

```
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED SIZE,
javax.swing.GroupLayout.DEFAULT SIZE, javax.swing.GroupLayout.PREFERRED SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(jCheckBox1))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jCheckBox2))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT SIZE, Short.MAX VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT SIZE, Short.MAX VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED SIZE,
javax.swing.GroupLayout.DEFAULT SIZE, javax.swing.GroupLayout.PREFERRED SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton2ActionPerformed
        // TODO add your handling code here:
        this.dispose();
    }//GEN-LAST:event jButton2ActionPerformed

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        dialogUser.setUsername(jTextField1.getText());
        dialogUser.setPassword(jTextField2.getText());

        dialogUser.setRemoteMonitor(jCheckBox1.isSelected());
        dialogUser.setRemoteControl(jCheckBox2.isSelected());

        this.dispose();
    }//GEN-LAST:event_jButton1ActionPerformed

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                EditUserDialog dialog = new EditUserDialog(new javax.swing.JFrame(),
true, dialogUser);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
```

```
                dialog.setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JCheckBox jCheckBox1;
    private javax.swing.JCheckBox jCheckBox2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration//GEN-END:variables
    protected static User dialogUser;

     private void initDialogField() {
        jTextField1.setText(dialogUser.getUsername());
        jTextField2.setText(dialogUser.getPassword());

        if (dialogUser.getRemoteMonitor())
            jCheckBox1.setSelected(true);
        else
            jCheckBox1.setSelected(false);

        if (dialogUser.getRemoteControl())
            jCheckBox2.setSelected(true);
        else
            jCheckBox2.setSelected(false);
    }
}
```

UserDatabase.java Source Code

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import javax.swing.JOptionPane;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class UserDatabase {
    protected ArrayList<User> userList;
    protected String filename;
    public UserDatabase()
    {
        userList = new ArrayList<User>();

        User user1 = new User("tiendung", "86270885", true, false);
        User user2 = new User("yini", "popcorn", true, true);

        userList.add(user1);
        userList.add(user2);
    }
```

```
public UserDatabase(String strFilename) throws FileNotFoundException, IOException
{
    filename = strFilename;
    userList = new ArrayList<User>();

    FileReader freader = new FileReader(filename);
    BufferedReader inputFile = new BufferedReader(freader);

         String filedata;
    filedata = inputFile.readLine();

    for (int i=0; i<Integer.parseInt(filedata); i++)
    {
        User user = new User();

        user.setUsername(inputFile.readLine());
        user.setPassword(inputFile.readLine());

        if (inputFile.readLine().equals("true"))
            user.setRemoteMonitor(true);
        else
            user.setRemoteMonitor(false);

        if (inputFile.readLine().equals("true"))
            user.setRemoteControl(true);
        else
            user.setRemoteControl(false);

        userList.add(user);
    }

    inputFile.close();
}

public void saveToFile() throws IOException
{
    FileWriter fwriter = new FileWriter(filename);
         PrintWriter outputFile = new PrintWriter(fwriter);

    outputFile.println(rowCount());

    for (int i=0; i<rowCount(); i++)
    {
        outputFile.println(userList.get(i).getUsername());
        outputFile.println(userList.get(i).getPassword());
        outputFile.println(userList.get(i).getRemoteMonitor());
        outputFile.println(userList.get(i).getRemoteControl());
    }

    outputFile.close();
}

public User getUser(int index)
{
    return userList.get(index);
}

public void setUser(int index, User user)
{
    userList.set(index, user);
}

public void addUser(User user)
{
    userList.add(user);
}

public void removeUser(int index)
{
    userList.remove(index);
}
```

```
    public Object[][] getObject()
    {
        Object[][] object = new Object[userList.size()+1][4];

        for (int i=0; i<userList.size(); i++)
        {
            object[i][0] = userList.get(i).getUsername();
            object[i][1] = userList.get(i).getPassword();
            object[i][2] = userList.get(i).getRemoteMonitor();
            object[i][3] = userList.get(i).getRemoteControl();
        }

        return object;
    }

    public int rowCount()
    {
        return userList.size();
    }
}
```

User.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class User {
    protected String username;
    protected String password;
    protected boolean remotecontrol;
    protected boolean remotemonitor;

    public User()
    {
            remotecontrol = false;
            remotemonitor = true;
            username = "";
            password = "";
    }

    public User(String strUsername, String strPassword, boolean bRemoteMonitor, boolean
bRemoteControl) {
        username = strUsername;
        password = strPassword;
        remotecontrol = bRemoteControl;
        remotemonitor = bRemoteMonitor;
    }

    public String getUsername()
    {
        if (username.length() > 0) return username;
        else return " ";
    }

    public String getPassword()
    {
        if (password.length() > 0) return password;
        else return " ";
    }

    public boolean getRemoteControl()
    {
        return remotecontrol;
    }
```

```
    public boolean getRemoteMonitor()
    {
        return remotemonitor;
    }

    public void setUsername(String strUsername)
    {
        username = strUsername;
    }

    public void setPassword(String strPassword)
    {
        password = strPassword;
    }

    public void setRemoteControl(boolean bRemoteControl)
    {
        remotecontrol = bRemoteControl;
    }

    public void setRemoteMonitor(boolean bRemoteMonitor)
    {
        remotemonitor = bRemoteMonitor;
    }
}
```

StreamingServer.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

import java.util.concurrent.BlockingQueue;
import java.io.*;
import java.io.IOException.*;
import java.util.*;

import javax.media.*;
import javax.media.control.*;
import javax.media.datasink.*;
import javax.media.format.*;
import javax.media.protocol.*;
import javax.media.rtp.*;
import java.awt.*;
import java.net.InetAddress;
import javax.media.rtp.rtcp.*;
import com.sun.media.rtp.*;

/**
 *
 * @author Vu Ba Tien Dung
 */
public class StreamingServer implements Runnable {

    private BlockingQueue<StreamClient> streamClientQueue;
    private int QUEUE SIZE;
    private StreamClient[] streamClientArray;

    private String defaultVideoDeviceName = "vfw:Microsoft WDM Image Capture";
    private String defaultVideoFormatString = "size=640x480, encoding=rgb,
maxdatalength=230400";

    private CaptureDeviceInfo captureVideoDevice = null;
    private VideoFormat captureVideoFormat = null;

    private boolean bStreaming;
    private Processor processor;
    private RTPManager rtpMgrs[];
```

```
    private final int SERVER PORT = 50000;

    public StreamingServer(BlockingQueue<StreamClient> q)
    {
        streamClientQueue = q;
        QUEUE_SIZE = 0;
        bStreaming = false;
        initCaptureDevice();
    }

    public void initCaptureDevice()
    {
        Vector deviceListVector = CaptureDeviceManager.getDeviceList(null);
        if (deviceListVector == null)
        {
            System.out.println("... error: media device list vector is null, program
aborted");
            System.exit(0);
        }
        if (deviceListVector.size() == 0)
        {
            System.out.println("... error: media device list vector size is 0, program
aborted");
            System.exit(0);
        }

        for (int x = 0; x < deviceListVector.size(); x++)
        {
            // display device name
            CaptureDeviceInfo deviceInfo = (CaptureDeviceInfo)
deviceListVector.elementAt(x);
            String deviceInfoText = deviceInfo.getName();

            // display device formats
            Format deviceFormat[] = deviceInfo.getFormats();
            for (int y = 0; y < deviceFormat.length; y++)
            {
                // search for default video device
                if (captureVideoDevice == null)
                if (deviceFormat[y] instanceof VideoFormat)
                if (deviceInfo.getName().indexOf(defaultVideoDeviceName) >= 0)
                {
                    captureVideoDevice = deviceInfo;
                    System.out.println(">>> capture video device = " +
deviceInfo.getName());
                }

                // search for default video format
                if (captureVideoDevice == deviceInfo)
                if (captureVideoFormat == null)
                if
(DeviceInfo.formatToString(deviceFormat[y]).indexOf(defaultVideoFormatString) >= 0)
                {
                    captureVideoFormat = (VideoFormat) deviceFormat[y];
                    System.out.println(">>> capture video format = " +
DeviceInfo.formatToString(deviceFormat[y]));
                }
            }
        }
        System.out.println("... list completed.");
    }


    @SuppressWarnings("static-access")
    public void run() {
        try
        {
            System.out.println("Start " + Thread.currentThread().getName());

            while (true)
            {
                if (streamClientQueue.size() != QUEUE_SIZE)
                {
```

```
                    QUEUE SIZE = streamClientQueue.size();

                    if (bStreaming) stopStream();

                    streamClientArray = new StreamClient[streamClientQueue.size()];
                    streamClientArray = streamClientQueue.toArray(new StreamClient[0]);

                    for (int i = 0; i<streamClientArray.length; i++)
                        System.out.println(Thread.currentThread().getName() + " " +
streamClientArray[i].getClientAddress() + " " + streamClientArray[i].getPortNumber());

                    createStream();
                }

                Thread.currentThread().sleep(10);
            }
        }
        catch (Exception e) {
           System.out.println(Thread.currentThread().getName() + " " + e.getMessage());
        }
    }

    public void createStream()
    {
        processor = null;
        StateHelper sh = null;

        try
        {
            processor = Manager.createProcessor(captureVideoDevice.getLocator());
            sh = new StateHelper(processor);
        }
        catch (IOException e) { }
        catch (NoProcessorException e) {  }

        // Configure the processor
        if (!sh.configure(10000)) System.exit(-1);

        // Set the output content type and realize the processor
        // processor.setContentDescriptor(new
FileTypeDescriptor(FileTypeDescriptor.MSVIDEO));

        // Get the track control objects
        TrackControl track[] = processor.getTrackControls();
        boolean encodingPossible = false;

        // Go through the tracks and try to program one of them to output JPEG RTP data
        for (int i = 0; i < track.length; i++)
        {
            if (!encodingPossible && track[i] instanceof FormatControl)
            {
                if (((FormatControl)track[i]).setFormat(new
VideoFormat(VideoFormat.JPEG RTP)) == null)
                    track[i].setEnabled(false);
                else
                    encodingPossible = true;
            }
            else
                track[i].setEnabled(false);
        }

        if (!encodingPossible)
        {
            sh.close();
            System.exit(-1);
        }

        if (!sh.realize(10000)) System.exit(-1);

        // get the output of the processor
        DataSource source = processor.getDataOutput();
```

```
        //SessionManager rtpsm = new com.sun.media.rtp.RTPSessionMgr();
        PushBufferDataSource pbds = (PushBufferDataSource) source;
        PushBufferStream pbss[] = pbds.getStreams();

        rtpMgrs = new RTPManager[pbss.length];
        rtpMgrs[0] = RTPManager.newInstance();

        SessionAddress localAddr;
        SessionAddress destAddr[];
        SendStream sendStream;

        try
        {
            //System.out.println("IP: " + streamClientArray[0].getClientAddress() + "
port: " + streamClientArray[0].getPortNumber());
            //System.out.println("length = " + streamClientArray.length);


            localAddr = new SessionAddress(InetAddress.getLocalHost(),
              SERVER PORT);
            rtpMgrs[0].initialize(localAddr);

            destAddr = new SessionAddress[streamClientArray.length];
            for (int i=0; i<streamClientArray.length; i++)
            {
                destAddr[i] = new
SessionAddress(streamClientArray[i].getClientAddress(),
streamClientArray[i].getPortNumber());
                rtpMgrs[0].addTarget(destAddr[i]);

                System.out.println("Created RTP session to: " +
streamClientArray[i].getClientAddress() + " " + streamClientArray[i].getPortNumber());
            }

            sendStream = rtpMgrs[0].createSendStream(source, 0);
            sendStream.start();

            // start the processor
            processor.start();
            bStreaming = true;
        }
        catch (Exception  e) {  }
    }

    public void stopStream() throws InvalidSessionAddressException
    {
        processor.close();
        bStreaming = false;

        // for (int i=0; i<QUEUE_SIZE; i++)
        // {
        //     SessionAddress destAddr = new
SessionAddress(streamClientArray[i].getClientAddress(),
streamClientArray[i].getPortNumber());
        //    rtpMgrs[0].removeTarget(destAddr, "bye bye");
        // }

        rtpMgrs[0].dispose();

        System.out.println("just cleaned");
    }
}
```

StreamClient.java Source Code

```
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
```

```
 *
 * @author Vu Ba Tien Dung
 */
public class StreamClient {

    private InetAddress iaClient;
    private int iPortNum;

    public StreamClient() throws UnknownHostException
    {
        iaClient = InetAddress.getLocalHost();
        iPortNum = 0;
    }

    public void setClientAddress(InetAddress value)
    {
        iaClient = value;
    }

    public void setPortNumber(int value)
    {
        iPortNum = value;
    }

    public InetAddress getClientAddress()
    {
        return iaClient;
    }

    public int getPortNumber()
    {
        return iPortNum;
    }
}
```

StateHelper.java [26] Source Code

```
import javax.media.*;


 public class StateHelper implements javax.media.ControllerListener {

     Player player = null;
     boolean configured = false;
     boolean realized = false;
     boolean prefetched = false;
     boolean eom = false;
     boolean failed = false;
     boolean closed = false;

     public StateHelper(Player p) {
               player = p;
               p.addControllerListener(this);
     }

     public boolean configure(int timeOutMillis) {
               long startTime = System.currentTimeMillis();
               synchronized (this) {
                   if (player instanceof Processor)
                             ((Processor)player).configure();
                   else
                             return false;

                   while (!configured && !failed) {
                             try {
                                 wait(timeOutMillis);
                             } catch (InterruptedException ie) {
                             }
                             if (System.currentTimeMillis() - startTime > timeOutMillis)
                                 break;
```

```
                }
            }
            return configured;
    }

    public boolean realize(int timeOutMillis) {
            long startTime = System.currentTimeMillis();
            synchronized (this) {
                player.realize();
                while (!realized && !failed) {
                        try {
                            wait(timeOutMillis);
                        } catch (InterruptedException ie) {
                        }
                        if (System.currentTimeMillis() - startTime > timeOutMillis)
                            break;
                }
            }
            return realized;
    }

    public boolean prefetch(int timeOutMillis) {
            long startTime = System.currentTimeMillis();
            synchronized (this) {
                player.prefetch();
                while (!prefetched && !failed) {
                        try {
                            wait(timeOutMillis);
                        } catch (InterruptedException ie) {
                        }
                        if (System.currentTimeMillis() - startTime > timeOutMillis)
                            break;
                }
            }
            return prefetched && !failed;
    }
    public boolean playToEndOfMedia(int timeOutMillis) {
            long startTime = System.currentTimeMillis();
            eom = false;
            synchronized (this) {
                player.start();

                while (!eom && !failed) {
                        try {
                            wait(timeOutMillis);
                        } catch (InterruptedException ie) {
                        }
                        if (System.currentTimeMillis() - startTime > timeOutMillis)
                            break;
                }
            }
            return eom && !failed;
    }

    public void close() {
            synchronized (this) {
                player.close();
                while (!closed) {
                        try {
                            wait(100);
                        } catch (InterruptedException ie) {
                        }
                }
            }
            player.removeControllerListener(this);
    }

    public synchronized void controllerUpdate(ControllerEvent ce) {
            if (ce instanceof RealizeCompleteEvent) {
                realized = true;
            } else if (ce instanceof ConfigureCompleteEvent) {
                configured = true;
            } else if (ce instanceof PrefetchCompleteEvent) {
```

```
                    prefetched = true;
                } else if (ce instanceof EndOfMediaEvent) {
                    eom = true;
                } else if (ce instanceof ControllerErrorEvent) {
                    failed = true;
                } else if (ce instanceof ControllerClosedEvent) {
                    closed = true;
                } else {
                    return;
                }
                notifyAll();
    }
}
```

DeviceInfo.java [27] Source Code

```
/*************************************************
 * File: DeviceInfo.java.java
 * created 24.07.2001 21:44:12 by David Fischer, fischer@d-fischer.com
 */

import java.awt.Dimension;

import javax.media.*;
import javax.media.control.*;
import javax.media.format.*;
import javax.media.protocol.*;


public class DeviceInfo
{

            public static Format formatMatches (Format format, Format supported[] )
            {
                        if (supported == null)
                                    return null;
                        for (int i = 0;  i < supported.length;  i++)
                                    if (supported[i].matches(format))
                                                return supported[i];
                        return null;
            }


            public static boolean setFormat(DataSource dataSource, Format format)
            {
                        boolean formatApplied = false;

                        FormatControl formatControls[] = null;
                        formatControls = ((CaptureDevice)
dataSource).getFormatControls();
                        for (int x = 0; x < formatControls.length; x++)
                        {
                                    if (formatControls[x] == null)
                                                continue;

                                    Format supportedFormats[] =
formatControls[x].getSupportedFormats();
                                    if (supportedFormats == null)
                                                continue;

                                    if (DeviceInfo.formatMatches(format,
supportedFormats) != null)
                                    {

            formatControls[x].setFormat(format);
                                                formatApplied = true;
                                    }
                        }

                        return formatApplied;
```

```
		}


		public static boolean isVideo(Format format)
		{
				return (format instanceof VideoFormat);
		}


		public static boolean isAudio(Format format)
		{
				return (format instanceof AudioFormat);
		}


		public static String formatToString(Format format)
		{
				if (isVideo(format))
						return videoFormatToString((VideoFormat)
format);

				if (isAudio(format))
						return audioFormatToString((AudioFormat)
format);

				return ("--- unknown media device format ---");
		}


		public static String videoFormatToString(VideoFormat videoFormat)
		{
				StringBuffer result = new StringBuffer();

				// add width x height (size)
				Dimension d = videoFormat.getSize();
				result.append("size=" + (int) d.getWidth() + "x" + (int)
d.getHeight() + ", ");

				/*
				// try to add color depth
				if (videoFormat instanceof IndexedColorFormat)
				{
						IndexedColorFormat f = (IndexedColorFormat)
videoFormat;
						result.append("color depth=" + f.getMapSize() +
", ");
				}
				*/

				// add encoding
				result.append("encoding=" + videoFormat.getEncoding() + ",
");

				// add max data length
				result.append("maxdatalength=" +
videoFormat.getMaxDataLength() + "");

				return result.toString();
		}


		public static String audioFormatToString(AudioFormat audioFormat)
		{
				StringBuffer result = new StringBuffer();

				// short workaround
				result.append(audioFormat.toString().toLowerCase());

				return result.toString();
		}
}
```

Appendix A.3 – Client Application Source Code

The file MediaPanel.java is cited from the Sun Microsystems' website [26].

Main.java Source Code

```
/**
 *
 * @author Vu Ba Tien Dung
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        ClientSideForm cf = new ClientSideForm();
        cf.setLocation(50,50);
        cf.setVisible(true);


    }

}
```

ClientSideForm.java Source Code

```
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.media.MediaLocator;
import javax.swing.JFrame;
import java.io.*;
import java.net.*;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * ClientSideForm.java
 *
 * Created on Apr 14, 2010, 8:50:04 AM
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class ClientSideForm extends javax.swing.JFrame {

    Socket clientSocket = null;
    Thread clientServerComm = null;
    Connecting connecting;
    private BlockingQueue<Protocol> protocolQueue;

    private final String FORWARD = "~FORW";
    private final String BACKWARD = "~BACK";
    private final String TURNLEFT = "~LEFT";
```

```
    private final String TURNRIGHT = "~RIGH";
    private final String STOP = "~STOP";

    /** Creates new form ClientSideForm */
    public ClientSideForm() {
        initComponents();
        connecting = new Connecting();

        protocolQueue = new LinkedBlockingQueue<Protocol>();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        label1 = new javax.swing.JLabel();
        textfieldServerIP = new javax.swing.JTextField();
        buttonConnect = new javax.swing.JButton();
        labelConnect = new javax.swing.JLabel();
        panelRemoteControll = new javax.swing.JPanel();
        buttonUp = new javax.swing.JButton();
        buttonRight = new javax.swing.JButton();
        buttonStop = new javax.swing.JButton();
        buttonLeft = new javax.swing.JButton();
        buttonDown = new javax.swing.JButton();
        labelUser = new javax.swing.JLabel();
        labelPassword = new javax.swing.JLabel();
        textfieldUsername = new javax.swing.JTextField();
        textfieldPassword = new javax.swing.JPasswordField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Robot Controller Client");

        label1.setText("Server IP:");

        textfieldServerIP.setText("127.0.0.1");
        try
        {
            int index = InetAddress.getLocalHost().toString().indexOf("/");

textfieldServerIP.setText(InetAddress.getLocalHost().toString().substring(index+1));
        }
        catch (UnknownHostException ex) { }

        buttonConnect.setText("Connect");
        buttonConnect.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonConnectActionPerformed(evt);
            }
        });

        labelConnect.setForeground(new java.awt.Color(255, 0, 0));
        labelConnect.setText("Not connect anywhere");


panelRemoteControll.setBorder(javax.swing.BorderFactory.createTitledBorder("Remote
Controlls"));

        buttonUp.setBackground(new java.awt.Color(255, 255, 255));
        buttonUp.setIcon(new javax.swing.ImageIcon(getClass().getResource("/up.JPG")));
// NOI18N
        buttonUp.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonUpActionPerformed(evt);
            }
        });
```

```
        buttonRight.setBackground(new java.awt.Color(255, 255, 255));
        buttonRight.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/right.JPG"))); // NOI18N
        buttonRight.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonRightActionPerformed(evt);
            }
        });

        buttonStop.setBackground(new java.awt.Color(255, 255, 255));
        buttonStop.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/stop.JPG"))); // NOI18N
        buttonStop.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonStopActionPerformed(evt);
            }
        });

        buttonLeft.setBackground(new java.awt.Color(255, 255, 255));
        buttonLeft.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/left.JPG"))); // NOI18N
        buttonLeft.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonLeftActionPerformed(evt);
            }
        });

        buttonDown.setBackground(new java.awt.Color(255, 255, 255));
        buttonDown.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/down.JPG"))); // NOI18N
        buttonDown.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buttonDownActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout panelRemoteControllLayout = new
javax.swing.GroupLayout(panelRemoteControll);
        panelRemoteControll.setLayout(panelRemoteControllLayout);
        panelRemoteControllLayout.setHorizontalGroup(

panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelRemoteControllLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(buttonLeft, javax.swing.GroupLayout.PREFERRED_SIZE, 66,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
                    .addGroup(panelRemoteControllLayout.createSequentialGroup()

.addGroup(panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING, false)
                            .addComponent(buttonStop, 0, 0, Short.MAX_VALUE)
                            .addComponent(buttonUp,
javax.swing.GroupLayout.PREFERRED SIZE, 66, Short.MAX VALUE))
                        .addGap(6, 6, 6)
                        .addComponent(buttonRight, 0, 66, Short.MAX_VALUE))
                    .addComponent(buttonDown, javax.swing.GroupLayout.PREFERRED_SIZE,
66, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );
        panelRemoteControllLayout.setVerticalGroup(

panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(panelRemoteControllLayout.createSequentialGroup()
                .addComponent(buttonUp, javax.swing.GroupLayout.PREFERRED_SIZE, 66,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
```

```
                    .addComponent(buttonRight, javax.swing.GroupLayout.PREFERRED SIZE,
66, javax.swing.GroupLayout.PREFERRED SIZE)
                    .addGroup(panelRemoteControllLayout.createSequentialGroup()

.addGroup(panelRemoteControllLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
                            .addComponent(buttonLeft,
javax.swing.GroupLayout.PREFERRED SIZE, 66, javax.swing.GroupLayout.PREFERRED SIZE)
                            .addComponent(buttonStop,
javax.swing.GroupLayout.PREFERRED_SIZE, 66, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(buttonDown,
javax.swing.GroupLayout.PREFERRED SIZE, 66, javax.swing.GroupLayout.PREFERRED SIZE)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        labelUser.setText("Username:");

        labelPassword.setText("Password:");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(labelConnect)
                    .addComponent(panelRemoteControll,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                                .addComponent(label1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                .addComponent(textfieldServerIP,
javax.swing.GroupLayout.PREFERRED_SIZE, 152, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(labelUser)
                                    .addComponent(labelPassword))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(textfieldPassword,
javax.swing.GroupLayout.DEFAULT_SIZE, 149, Short.MAX_VALUE)
                                    .addComponent(textfieldUsername,
javax.swing.GroupLayout.DEFAULT SIZE, 149, Short.MAX VALUE))))
                        .addGap(10, 10, 10)
                        .addComponent(buttonConnect)))
                .addGap(28, 28, 28))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(label1)
                    .addComponent(textfieldServerIP,
javax.swing.GroupLayout.PREFERRED SIZE, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(buttonConnect))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(labelConnect)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(labelUser)
                    .addComponent(textfieldUsername,
javax.swing.GroupLayout.PREFERRED SIZE, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.PREFERRED SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(labelPassword)
                    .addComponent(textfieldPassword,
javax.swing.GroupLayout.PREFERRED SIZE, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(panelRemoteControll,
javax.swing.GroupLayout.PREFERRED SIZE, javax.swing.GroupLayout.DEFAULT SIZE,
javax.swing.GroupLayout.PREFERRED SIZE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void buttonLeftActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonLeftActionPerformed
        // TODO add your handling code here:
        Protocol value = new Protocol();
        value.setCommand(TURNLEFT.substring(1));

        protocolQueue.add(value);
}//GEN-LAST:event buttonLeftActionPerformed

    private void buttonDownActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonDownActionPerformed
        // TODO add your handling code here:
        Protocol value = new Protocol();
        value.setCommand(BACKWARD.substring(1));

        protocolQueue.add(value);
}//GEN-LAST:event_buttonDownActionPerformed

    private void buttonConnectActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonConnectActionPerformed
        if (buttonConnect.getText().indexOf("Dis") == -1)
        {
            try {
                // TODO add your handling code here:
                clientSocket = new
Socket(InetAddress.getByName(textfieldServerIP.getText().trim()), 33333);
                if (clientSocket != null) {
                    connecting.setConnecting(true);
                    buttonConnect.setText("Disconnect");
                    labelConnect.setText("Connecting to server at " +
textfieldServerIP.getText().trim());

                    textfieldUsername.setEditable(false);
                    textfieldPassword.setEditable(false);
                    User user = new User();
                    user.setPassword(new String(textfieldPassword.getPassword()));
                    user.setUsername(textfieldUsername.getText());

                    clientServerComm = new Thread(new ClientServerComm(clientSocket,
connecting, user, this, this.protocolQueue));
                    clientServerComm.start();
                }
            } catch (IOException ex) {}
        }
        else if (buttonConnect.getText().indexOf("Dis") != -1)
        {
            connecting.setConnecting(false);
```

```
            buttonConnect.setText("Connect");
            labelConnect.setText("Not connect anywhere");

            textfieldUsername.setEditable(true);
            textfieldPassword.setEditable(true);
        }
    }//GEN-LAST:event buttonConnectActionPerformed

    private void buttonUpActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_buttonUpActionPerformed
        // TODO add your handling code here:
        Protocol value = new Protocol();
        value.setCommand(FORWARD.substring(1));

        protocolQueue.add(value);
    }//GEN-LAST:event_buttonUpActionPerformed

    private void buttonRightActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonRightActionPerformed
        // TODO add your handling code here:
        Protocol value = new Protocol();
        value.setCommand(TURNRIGHT.substring(1));

        protocolQueue.add(value);
    }//GEN-LAST:event buttonRightActionPerformed

    private void buttonStopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event buttonStopActionPerformed
        // TODO add your handling code here:
        Protocol value = new Protocol();
        value.setCommand(STOP.substring(1));

        protocolQueue.add(value);
    }//GEN-LAST:event buttonStopActionPerformed

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClientSideForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton buttonConnect;
    private javax.swing.JButton buttonDown;
    private javax.swing.JButton buttonLeft;
    private javax.swing.JButton buttonRight;
    private javax.swing.JButton buttonStop;
    private javax.swing.JButton buttonUp;
    private javax.swing.JLabel label1;
    private javax.swing.JLabel labelConnect;
    private javax.swing.JLabel labelPassword;
    private javax.swing.JLabel labelUser;
    private javax.swing.JPanel panelRemoteControll;
    private javax.swing.JPasswordField textfieldPassword;
    private javax.swing.JTextField textfieldServerIP;
    private javax.swing.JTextField textfieldUsername;
    // End of variables declaration//GEN-END:variables

    public javax.swing.JButton getButtonConnect()
    {
        return buttonConnect;
    }

    public javax.swing.JLabel getLabelConnect()
    {
        return labelConnect;
    }
```

```
    public javax.swing.JTextField getTextUsername()
    {
        return textfieldUsername;
    }

    public javax.swing.JTextField getPassword()
    {
        return textfieldPassword;
    }

    public javax.swing.JTextField getIPAddress()
    {
        return textfieldServerIP;
    }

}
```

Connecting.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class Connecting {

    boolean bConnecting;

    public Connecting()
    {
        bConnecting = false;
    }

    public synchronized void setConnecting(boolean value)
    {
        bConnecting = value;
    }

    public synchronized boolean getConnecting()
    {
        return bConnecting;
    }
}
```

Protocol.java Source Code

```
import java.net.InetAddress;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class Protocol {
    private InetAddress IPAddress;
    private String strCommand;
```

```
    private String strValue1;
    private String strValue2;

    public Protocol()
    {
        IPAddress = null;
        strCommand = null;
        strValue1 = null;
        strValue2 = null;
    }

    public void setIPAddress(InetAddress value)
    {
        IPAddress = value;
    }

    public void setCommand(String value)
    {
        strCommand = value;
    }

    public void setValue1(String value)
    {
        strValue1 = value;
    }

    public void setValue2(String value)
    {
        strValue2 = value;
    }

    public InetAddress getIPAddress()
    {
        return IPAddress;
    }

    public String getCommand()
    {
        return strCommand;
    }

    public String getValue1()
    {
        return strValue1;
    }

    public String getValue2()
    {
        return strValue2;
    }
}
```

User.java Source Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class User {
    protected String username;
    protected String password;
    protected boolean remotecontrol;
    protected boolean remotemonitor;

    public User()
```

```
    {
            remotecontrol = false;
            remotemonitor = true;
            username = "";
            password = "";
    }

    public User(String strUsername, String strPassword, boolean bRemoteMonitor, boolean
bRemoteControl) {
        username = strUsername;
        password = strPassword;
        remotecontrol = bRemoteControl;
        remotemonitor = bRemoteMonitor;
    }

    public String getUsername()
    {
        if (username.length() > 0) return username;
        else return " ";
    }

    public String getPassword()
    {
        if (password.length() > 0) return password;
        else return " ";
    }

    public boolean getRemoteControl()
    {
        return remotecontrol;
    }

    public boolean getRemoteMonitor()
    {
        return remotemonitor;
    }

    public void setUsername(String strUsername)
    {
        username = strUsername;
    }

    public void setPassword(String strPassword)
    {
        password = strPassword;
    }

    public void setRemoteControl(boolean bRemoteControl)
    {
        remotecontrol = bRemoteControl;
    }

    public void setRemoteMonitor(boolean bRemoteMonitor)
    {
        remotemonitor = bRemoteMonitor;
    }
}
```

MediaPanel.java [26] Source Code

```
import java.awt.BorderLayout;
import java.awt.Component;
import java.io.IOException;
import java.net.URL;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.MediaLocator;
import javax.media.NoPlayerException;
import javax.media.Player;
```

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * MediaPanel.java
 *
 * Created on Apr 6, 2010, 4:05:12 AM
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
public class MediaPanel extends javax.swing.JPanel {
    Player mediaPlayer;

    /** Creates new form MediaPanel */
    public MediaPanel(URL mediaURL) {
        initComponents();

        setLayout( new BorderLayout() ); // use a BorderLayout

        try {

            // Create a JMF player to play the media specified in the URL:
            mediaPlayer = Manager.createRealizedPlayer( new MediaLocator(mediaURL) );

            // Get the components for the video and the playback controls:
            Component video = mediaPlayer.getVisualComponent();

            if ( video != null )
                add( video, BorderLayout.CENTER ); // add video component

            // Start the JMF player:
            mediaPlayer.start(); // start playing the media clip

        } // end try

        catch ( NoPlayerException noPlayerException ) {
            System.err.println( "No media player found" );
        } // end catch

        catch (CannotRealizeException ex) {
                ex.printStackTrace();
        } // end catch

        catch ( IOException iOException ) {
            System.err.println( "Error reading from the source" );
        } // end catch
    }

    public MediaPanel(String mediaURL) {
        initComponents();

        setLayout( new BorderLayout() ); // use a BorderLayout

        try {

            // Create a JMF player to play the media specified in the URL:
            mediaPlayer = Manager.createRealizedPlayer( new MediaLocator(mediaURL) );

            // Get the components for the video and the playback controls:
            Component video = mediaPlayer.getVisualComponent();

            if ( video != null )
                add( video, BorderLayout.CENTER ); // add video component

            // Start the JMF player:
            mediaPlayer.start(); // start playing the media clip

        } // end try
```

```
        catch ( NoPlayerException noPlayerException ) {
            System.err.println( "No media player found" );
        } // end catch

        catch (CannotRealizeException ex) {
                ex.printStackTrace();
        } // end catch

        catch ( IOException iOException ) {
            System.err.println( "Error reading from the source" );
        } // end catch
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );
    }// </editor-fold>//GEN-END:initComponents

public void dispose() {
        mediaPlayer.stop();
        mediaPlayer.close();

    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables

}
```

ClientServerComm.java Source Code

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.StringTokenizer;
import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.media.MediaLocator;
import javax.swing.JFrame;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vu Ba Tien Dung
 */
```

```
public class ClientServerComm implements Runnable {

    private BlockingQueue<Protocol> protocolQueue;

    Socket clientSocket;
    Connecting connecting;
    User userInfo;
    ClientSideForm clientForm;
    PrintWriter out;
    BufferedReader in;

    private final String BYEBYE = "~BYET";
    private final String PASSWORD = "~PASS";
    private final String SUCCESS = "~SUCC";
    private final String FORWARD = "~FORW";
    private final String BACKWARD = "~BACK";
    private final String TURNLEFT = "~LEFT";
    private final String TURNRIGHT = "~RIGH";
    private final String STOP = "~STOP";
    private final String USERNAME = "~USER";
    private final String RTP_INIT = "~RTPI";
    private final String RTP_CHANGE = "~RTPC";
    private final String DONE = "~DONE";

    private final char delimeter = '~';

    private final int PRIVILEDGE_HIGHEST = 15;
    private final int PRIVILEDGE_NORMAL = 5;
    private final int PRIVILEDGE LIMITED = 2;

    private int rtpPort;
    JFrame mediaFrame;
    MediaPanel mediaPanel;

    private int userPriviledge;
    private int commandIndex;

    public ClientServerComm(Socket socket, Connecting value, User user, ClientSideForm
cf, BlockingQueue<Protocol> q)
    {
        clientSocket = socket;
        connecting = value;
        userInfo = user;
        clientForm = cf;

        rtpPort = 0;
        userPriviledge = 0;
        commandIndex = 0;
        mediaFrame = null;
        mediaPanel = null;

        protocolQueue = q;

    }

    @SuppressWarnings("static-access")
    public void run()
    {
        String inputString, outputString;
        inputString = null;
        outputString = null;

        try {

            out = new PrintWriter(clientSocket.getOutputStream(), true);
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            while (connecting.getConnecting()) {

                // Processing the input commands
                if (in.ready()) inputString = in.readLine();

                if (inputString != null)
```

```
                {
                    // BYEBYE command
                    if (inputString.startsWith(BYEBYE)) {
                        connecting.setConnecting(false);
                        clientForm.getButtonConnect().setText("Connect");
                        clientForm.getLabelConnect().setText("Not connect anywhere");
                        clientForm.getTextUsername().setEditable(true);
                        clientForm.getPassword().setEditable(true);
                    }

                    // USER command
                    if (inputString.startsWith(USERNAME)) {
                        outputString = PASSWORD + delimeter + userInfo.getUsername() +
delimeter + userInfo.getPassword();
                        out.println(outputString);

                        System.out.println("password: " + userInfo.getPassword());
                        System.out.println("in: " + inputString);
                        System.out.println("out: " + outputString);
                    }

                    // SUCCESS command
                    if (inputString.startsWith(SUCCESS))
                    {
                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String str = st.nextToken();
                        str = st.nextToken();

                        userPriviledge = Integer.parseInt(str);
                    }

                    // RTP INIT command
                    if (inputString.startsWith(RTP INIT))
                    {
                        StringTokenizer st = new StringTokenizer(inputString,
Character.toString(delimeter));
                        String str = st.nextToken();
                        str = st.nextToken();
                        rtpPort = Integer.parseInt(str);

                        String url = "rtp://" +
clientForm.getIPAddress().getText().trim() + ":" + rtpPort + "/video";
                        System.out.println(url);
                        MediaLocator ml = new MediaLocator(url);
                        if ( ml != null ) // Only display if there is a valid URL
                        {
                            mediaFrame = new JFrame( "Streaming Player" );
                            mediaFrame.setLocation(400, 50);
                            mediaPanel = new MediaPanel( ml.toString() );
                            mediaFrame.add( mediaPanel );
                            mediaFrame.setSize( 300, 300 );
                            mediaFrame.setVisible( true );
                        }
                    }

                    // RTP CHANGE command
                    if (inputString.startsWith(RTP CHANGE))
                    {
                        mediaPanel.dispose();
                        mediaFrame.dispose();

                        String url = "rtp://" +
clientForm.getIPAddress().getText().trim() + rtpPort + "/video";
                        MediaLocator ml = new MediaLocator(url);
                        if ( ml != null ) // Only display if there is a valid URL
                        {
                            mediaFrame = new JFrame( "Streaming Player" );
                            mediaFrame.setLocation(400, 50);
                            mediaPanel = new MediaPanel( ml.toString() );
                            mediaFrame.add( mediaPanel );
                            mediaFrame.setSize( 300, 300 );
                            mediaFrame.setVisible( true );
```

```
                        }
                    }

                    // DONE command
                    if (inputString.startsWith(DONE))
                    { }

                    inputString = null;
                }



                // Processing the output information
                if (protocolQueue.isEmpty() == false)
                {
                    Protocol value = protocolQueue.take();

                    if (userPriviledge >= PRIVILEDGE NORMAL)
                    {
                        if (value.getCommand().equals(FORWARD.substring(1)))
                            outputString = FORWARD + delimeter +
Integer.toString(commandIndex++) + delimeter + Integer.toString(0);

                        if (value.getCommand().equals(BACKWARD.substring(1)))
                            outputString = BACKWARD + delimeter +
Integer.toString(commandIndex++) + delimeter + Integer.toString(0);

                        if (value.getCommand().equals(TURNLEFT.substring(1)))
                            outputString = TURNLEFT + delimeter +
Integer.toString(commandIndex++) + delimeter + Integer.toString(0);

                        if (value.getCommand().equals(TURNRIGHT.substring(1)))
                            outputString = TURNRIGHT + delimeter +
Integer.toString(commandIndex++) + delimeter + Integer.toString(0);

                        if (value.getCommand().equals(STOP.substring(1)))
                            outputString = STOP + delimeter +
Integer.toString(commandIndex++) + delimeter + Integer.toString(0);

                        out.println(outputString);
                    }
                }

                // Take a rest a bit, so the other threads can run
                Thread.currentThread().sleep(20);

            }

            if (mediaPanel != null) mediaPanel.dispose();
            if (mediaFrame != null) mediaFrame.dispose();

            outputString = BYEBYE + delimeter + Integer.toString(0) + delimeter +
Integer.toString(0);
            out.println(outputString);

            out.close();
            in.close();
            clientSocket.close();

        } catch (IOException ex) {}
        catch (InterruptedException et) {}
    }

}
```