A Web-Based Teleoperative Mobile Robotic System

Master of Engineering

in

Information Engineering

at

Massey University,

Albany, Auckland, New Zealand

Yi Wang

April 2006

ABSTRACT

With the rapid development of internet technology, it becomes real that human beings can access, modify and control a remote hardware device via internet connection. Such remote operations can replace the human to be present at a dangerous or unreachable place or can make as many as possible users to access the hardware in different places at a low cost.

The thesis research was aimed at developing a web based mobile robot control framework for education purpose. It should be composed of a mobile robot, Http server, dynamic user interface and video server. With it users can view and control the real robot via a normal web browser and can choose to run either simulation or the real robot. This is done by setting up operational parameters via a friendly GUI (graphic user interface). Users also can upload and compile their own C code to control the robot and get back the running results.

The main objectives of this thesis research are hardware upgrading for Nomadic Super Scout mobile robot and web based php programming. For the first objective, the onboard PC was replaced by a laptop that is remotely placed and connected to the robot control system via Bluetooth wireless. The Nserver for robot simulation was set up in the Linux operating environment. For the second objective, the software programming was focused on building a web control platform which should be user friendly. An Apache server was developed where PHP program was used for the user interface. The main advantage of using PHP is that it does not need to install or download any software or script to get access to the remote robot via a normal web browser on any operation like windows or Linux.

The web-based mobile robot system was tested using two different cases. One case demonstrated how the user specifies a set of motion parameters of the robot that is programmed to perform a wall-following behaviour. The other

demonstrated how the user uploads a collision avoidance program to run the robot that is placed among obstacles. Both case studies were performed in real environments and the results proved the success of the developed web-based robotic system.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude and thank to my supervisors Dr Peter Xu and Dr Mahammad A.Rashid for their constant supervision. Dr Peter Xu has provided me with all the resources I needed for this research. He has also given me the ideas to solve the problems I met in this project. Dr Rashid has taught me the attitude and method of the research.

I would like to thank ones who were involved and shared their time with me during the course of the project. Finally I will give my think to my family.

CONTENTS

List of Figures

Chapter 1 Introduction

1.1 Goal and Outline of the Thesis

The goal of this thesis research was to develop a web-based control platform by which every user can use normal web browser to access and control the Nomadic Super Scout robot at Massey University. One could either run the provided sample program by specifying a set of motion parameters or write and upload a program in C code to operate the robot via GUI. The robot movement status such as coordinators, sensor data, and images could be viewed and stored for further analysis.

This platform could be run at any computer operation system. Remote user might log onto the robot via any popular web browser over internet. It would not need to download or install any program or plug-in to control the robot.

This platform would serve as a test-bed for any one who is interested in the mobile robot. One could upload a program to and compile it on the server, do the testing runs and get back the running results in a file or displayed on the web browser.

This project can also help students who are interested in mobile robotics to learn the new technology and improve their problem solving skills. Making the Super Scout online will reduce the barrier to use the robot. It can meet different levels users' requirements. For an entry level user who just wants to be familiar with mobile robot, he or she can log onto the web and run sample programs to get the first hands-on experience. For those who are already familiar with this mobile robot system, they can write their program that use the sensing facilities to perform intelligent action, upload to and compile at the web to get the program started running. This can be achieved by any popular web browser over internet.

1.2 Upgrading the Robot Computer Hardware

The mobile robot to be used in this project is Nomadic Super Scout. It was out of order at the time the project began. It took about eight minutes to reboot its Linux system and the hard disk did not work properly.

Firstly, the main onboard PC was upgraded. Nomadic Super Scout communication, motion control and sensing are managed by an industrial PC mounted internally. The PC has a 233MHZ CPU, 32MB RAM and 10G hard disk. Actually this system should be able to run Linux system smoothly but why it takes so long to get Linux started? The hard disk is a bottleneck. This hard disk with a PCI adaptor card is inserted into the PCI slot instead of using the IDE slot directly. The reason it was designed like this may be to maximize the battery running time. If a standard PC hard disk is used, it would significantly reduce the robot running time with only two 12 Volt 17 amp lead-acid batteries.

A laptop computer is intended to replace the onboard computer. It will communicate with the robot controller wirelessly. The advantage of using a wireless laptop is that it is portable and the on-board batteries will be used by the robot only. Consequently the robot would run for a longer time between charging intervals.

What follows is to install a proper version of Linux on the laptop, setup the Linux environment suitable for the Super Scout, install the robot control software in the Linux operating system and connect the laptop to the robot controller. This is the most essential part of the hardware improvement.

1.3 Software Programming

Once the robot runs well, the efforts will go to build up a server based on Linux operating system for TeleScout system.

- Develop a Linux HTTP server which can communicate with the robot server;
- Build an image server for broadcasting the video stream;

- Develop a user interface which connects user with the robot host;
- Resolve security issues such as only a single user can run the robot;
- Provide a facility for users to upload and compile their own C code to control the robot and monitor its motion.
- Compile error messages if any, visualise the robot motion and generate a results report

In the thesis the design and implementation of the web-based mobile robot system, TeleScout system will be presented. The rest of the thesis is organised as follows.

An overview of previous and related work in web-based or telerobotics is given in Chapter 2. Chapter 3 describes the improvement of the robot control hardware and the setup for TeleScout system. The HTTP server setup and the development of the user interface are presented in Chapter 4. Chapter 5 gives two case studies. Chapter 6 concludes the thesis with open research issues.

Chapter 2 Telerobotics over Internet

2.1 Introduction

2.1.1 Definition of Telerobotics

A robot is a "reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks" [1].

A teleoperator is a machine that extends a person's sensing and/or manipulating capability to a location remote from that person [1]. It requires artificial sensors of the environment, a vehicle for moving them in the remote environment, and communication channels from and to the human operator. It may also include artificial arms or hands or other devices to apply forces and perform mechanical work on the environment.

A telerobot is an advanced form of a teleopertor the behaviour of which a human operator supervises through a computer intermediary [1]. A telerobot executes tasks based on the information received from the human operator plus its own artificial sensing and intelligence. A Telerobotic system enables humans to extend their action and intelligence to remote locations. Generally, internet based robots consist of teleoperators, robots, and the internet which is used as a communication channel. The internet is used as transmission media; hence no special communication channels are needed to be constructed. Due to the nature of the internet, the system can be accessed almost anywhere, as long as one can access the internet.

Telerobotic applications areas vary widely from performing maintenance function in hazardous environments, tele-education to homecare services.

Figure 2.1 is a basic Telerobotic model in which users send requests to the remote Http server by a normal web browser which is a program used to access the internet services and resources available through the World Wide Web [2] by a

user interface over internet. The user interface is at Web Server connected to the robot. The Web Server and the user interface translate the web requests to the robot controller and send the information from robot server to user by web browser. When using this system, remote users can use a normal web browser to operate the robot over internet, but sometimes they need to install some special plug-ins that depend on what the language to be used to program the interface part.

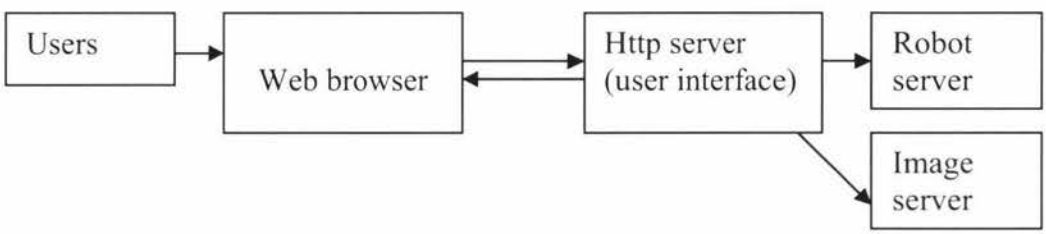


Figure 2- 1 The Basic Telerobotic Model

An internet controlled telerobotics system is likely to consist of a number of physical devices such as robot(s), sensors, and other actuators [3]. A framework of internet telerobotics has been implemented as a UWA telerobotic project started by Taylor [4]. It is used to control an industrial robot. Users can download a Java applet to control the robot and show robot image. The user interface was programmed as a downloadable Java applet to minimize the code size. The basic service required for telerobotics systems over the internet should include connectivity, user authentication, addressing, resource management, time delay estimation, and order request delivery [3].

2.1.2 Implementation of Web-Based Telerobots

There are mainly two methods to design the interface of the web-based Telerobotic system. The first is to let users to enter commands or parameters in a form and send to remote Web server via internet. The remote server will send those commands or parameters to the remote robot and will send the feed back from the robot to the users by an updated web page. The other way is to run program on the local computer's web browser, for example JAVA applet; the local computer creates immediate feedback.

The main advantage of the first method is that nothing is needed to install locally. A user can just use a normal web browser and low hardware requirement to control the robot via internet and get the feed back since all the scripts are executed at the remote web server. The disadvantage of this method is that the user has to stay idle while the request is transferred to the server and processed until the reply is transferred back.

The advantage of the second method is that the user interface is a web browser plug-in that is executed locally. It has fast response since it is executed locally but the disadvantage of this method is specific operation system and web browser versions needed and additional effort required by the user to install the software.

For example, a plugin can't be installed on the University computer so that the access to and control of any user interface of Telerobotic system written in JAVA applet are not possible. However there is no any problem to access the Telerobotic system developed in the first method. In this project, the first method was used for designing the user interface.

2.2 Issues in Telerobotics

There are a number of issues that must be addressed while developing a Telerobotic system. They are described as follows.

(1) Authentication Authentication is important for different level user to access different part of the remote robot. A user needs to get his/her user name and password first to get the right to access the robot. For some parts of the system like video part, it can be accessible to most of the users. But for some high level operation, a user needs permission to access for safety reason.

(2) Time delay management Time delay must be considered for a Telerobotic system. For example, user may operate the robot via internet by watching the real video stream. Due to the time delay, the robot may have already finished moving while the user is still seeing the robot moving. The NASA pathfinder [5,6] took about 11 minutes to send and receive information from the robot on Mars.

Researchers designed a simulation system in which they could move the robot and record all the commands and then send to the robot on Mars.

(3) Operation safety Operation safety is important for a telerobotic system, especially for robot wandering around which might collide with obstacles. Different levels of the system operation may be a way to avoid collision occurring.

(4) Feedback The feedback from the remote environment is important for users to know the current status of the robot. The feedbacks include video, audio and tactile information etc. Normally one or more video cameras are required to capture the live video stream and broadcast over the internet. Users can see the remote environment by the video and then operation the robot. Users can also download those sensor data via a web browser.

2.3 Review of Major Teleoperated Robots

The first "teleoperated robots" were developed over 30 years ago. Teleoperation began with very simple mock-ups in nuclear power plants [7,8]. Over the last 20 years, the development of intuitively operable teleoperation tools has continued to play an important role in the development of robotics in general. The basic objectives remain unchanged, even though the methods and technical limitations have been changing constantly [7, 9].

The Mars Pathfinder – sojourner which is the name given to the first robotic roving vehicle sent to the planet [5, 6]. Sojourner is a single vehicle as Figure 2-2 landed on Mars on July 4, 1997. It carried out many engineering, technology and science experiments.

NASA researchers can operate this vehicle by a simple user graphic interface as shown in Figure 2-3. There are command buttons, parameter input area on the left side of the interface and the Sojourner operator can also see the real image captured by the camera mounted on the Pathfinder although there is a huge delay to send the command and receive the feedback. The delay is about 11 minutes [6].

On the right side of Figure 2-2 is a screenshot showing the surface of Mars taken from the Lander IMP camera. Operator can goggle to see this 3D video and use a special joystick to move the model of Sojourner on the screen. The entire coordinates are recorded in sequence and would send to Sojourner to execute.

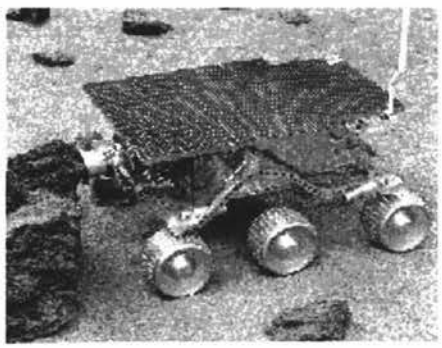


Figure 2- 2 The Sojourner, http://mars.jpl.nasa.gov/MPF/mpf/rover.html

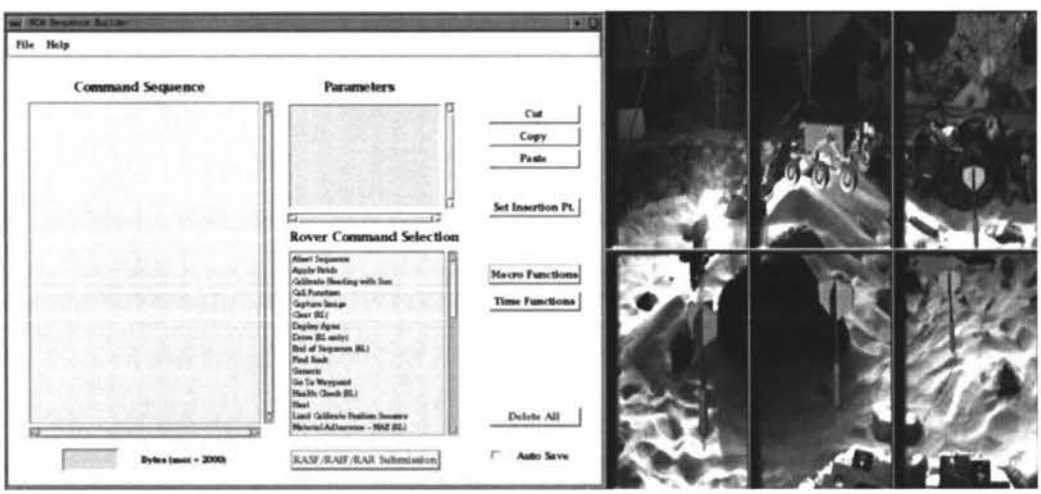


Figure 2- 3 The Sojourner GUI

The first internet –based teleportation is Mercury Project carried out by Goldberg in 1994[6]. This system allows WWW users to control and view the real robot. Remote users can move the robot arm fitted with a video camera around to find the buried artefacts in a sand box. There are 3 servers in this system called A, B and C. Server A stores the most recent images, server B is the database server and server C is used to operate the robot. When a client sends a request to the server, server A communicates with B which confirms if the client is an operator, If yes, server A will talk to C that allows the client to take control over the robot. The

user graphic interface is shown in Figure 2-4; the left side is a real time image which is clickable. The robot arm can move to any point on the image the remote user click on and a new local image will be returned to the user. There is also an operator password needed to operate this robot [2].

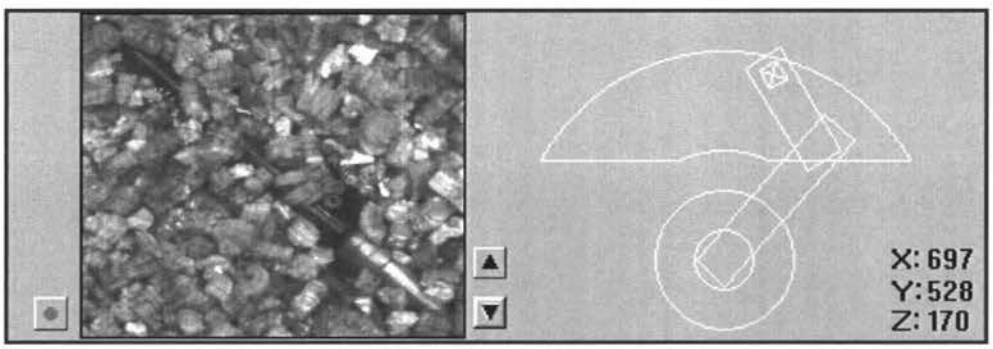


Figure 2- 4 The Mercury Project

The PumaPaint Project [9,10] is a tele-oporative robot online which allows WWW users to create their own paint works. About 25,000 unique-addressed machines downloaded the interface to produce about 500 canvases from June 1998 to March 2000[9]. The PumaPaint robot is equipped with 4 paint brushes and 4 jars of colour paint (red, blue, yellow and green) and also white paper is placed vertically in front of the robot.

The user interface is programmed via JAVA. Users have to download and install the JAVA applet to control and view the robot. As shown in Figure 2-5, a remote user can select one of the brushes by clicking the brush on the left-hand side of the interface. Then the remote user can use mouse point to draw anything in the middle of the blank area. At the same time, the robot can get the related coordinates and draw the same picture on the paper. The small window on the right-hand is the real broadcasting video stream captured by the camera.

Since only one user can really operate the robot online, the PumaPaint users can queue up their command online and check the queue size to determine how long they should wait for their turn. The system will automatically disconnect a remote user who is idle for a certain time.

The code of the interface is running on the client side. It is very efficient to run this applet since the JAVA code has already been compiled. This may not be good for users who are using a computer in places like university computing labs where unauthorised plug-ins and software are installed.

Figure 2- 5 http://pumapaint.rwu.edu/PumaPaint.html

The TeleGarden is an art installation that allows web users to view and interact with a remote garden filled with living plants. Members can plant, water, and monitor the progress of seedlings via the tender movements of an industrial robot arm [12]. The TeleGarden is developed by University of Southern California. Over 9000 members logged on this system in the first year when it was online since June of 1995

Figure 2-6 is the World Wide Web interface to the telegarden. Remote user can move the robot arm by clicking on the image at the left and right side of the interface. The left image is the entire view of the garden and the right image is the image captured by the camera on the robot arm.

At the bottom of the WWW user interface which is programmed via Common Gateway Interface Program (CGI) in C language. There are some programmed clickable buttons. Remote users can click on it to send command to the robot to do various planting operations. For example, there are two buttons at the user's interface – water and plant; a remote user can first find a free spot by checking the

image and the robot will automatically dig a hole into the ground and put a seed in that hole. To water, a remote user just needs to move the robot arm to the place where they want to water and then press the water button.

More than one remote user can access the robot at the same time since a fast industrial robot is used in TeleGarden [13]. Actually at one time point, a robot can only execute one users' command. But since the robot moves very fast and the operation like watering and planting has already been pre-programmed, more than one user can send commands that are stored in the server and will be executed one after another.

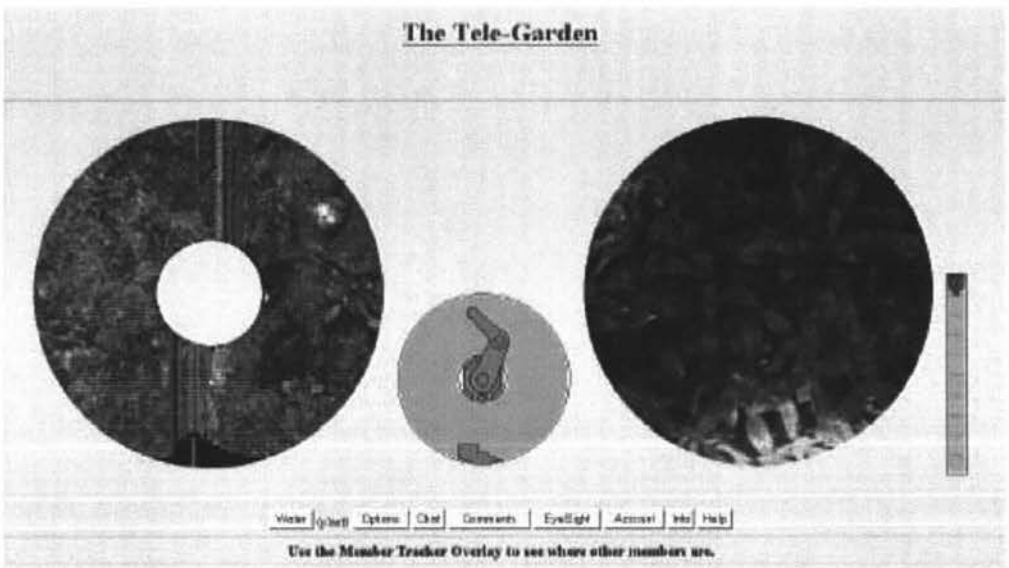


Figure 2- 6 The Tele-Garden User Interface

2.4 Remote Host Machine

For a teleoperation system over internet, efficient and effective communication between users and web server must be established. The command to control the remote hardware device from the users is sent to the remote server and the status of the hardware device and server are fed back to users.

Web server is a computer on the World Wide Web (connected to the Internet Backbone) that stores HTML documents that can be retrieved via a Web browser [14]. For a teleoperating system, the web server gets the request from remote

users, and then sends it to the hardware device. Once finishing up its responding, the web Server will generate related texts and graphic information back to the remote users. Different levels of connection also need to be established in this system. For a normal user, the basic right is to operate the robot to perform some basic operations. It is dedicated to those who are not very familiar with the robot but want to have a try. For higher level of operations on the robot, users need to get the higher level authorization to access the robot. This is also a way to prevent the hardware device from being damaged by unskilful operations.

The web server also connects to the robot controller to get the motion status and internal states of the robot which include robot coordinates and sensing results etc. The web server can feed the information back to remote users by generating the HTTP code over internet.

2.5 User Interface in CGI, PHP and JAVA for Telerobotic System

Telerobotic systems enable remote multiple users to access and control the robot through World Wide Web. One of the most important parts for a telerobotic system is to design a user interface shown in Figure 2-1. In most developed telerobotic systems, user interaction is still very computer oriented, since input to the robot is accomplished by filling in forms or selecting commands from a panel [15]. The existing interface in a teleoperation system lacks a functionality for bi-directional communication with the hardware device to be controlled. This includes sending commands and getting its status back. For some of the developed teleoperation systems, there is a camera mounted on the device so that remote users can view the device online and perform the remote operation.

The interface is usually programmed by a language like CGI, JAVA, and PHP which can communicate with the hardware device to be operated remotely over internet.

2.5.1 Common Gateway Interface (CGI)

Remote user uses web browser to send the http request to the web server that executes the CGI code associated with HTML codes at the server side. CGI results are sent back to remote user by HTML response. C/C++, Visual Basic and Perl are often used to write CGI script. CGI scripts are executed on the server side only when a user clicks to submit a fill-out form on the web browser.

The Telegarden project uses CGI to control the robot arm's movement. Remote users use mouse to click the image to generate the three dimensional coordinates that go to the web server. The CGI scripts at the web server are then executed to process the information from the client and send it to the robot daemon program to control the movement of the robot.

There are many other teleoperation systems using CGI scripts at the web server side like NASA Pathfinder Project [5] and Mercury Project [6].

2.5.2 Hypertext Preprocessor (PHP)

PHP(recursive acronym for "PHP: Hypertext Preprocessor") is a widely-used open source general- purpose scripting language that is especially suited for web development and can be embedded into HTML [16]. PHP is similar to CGI in that it is only executed at the server side. The difference is that PHP is embedded into HTML. The main advantage of PHP is that it has been optimized for fast response that is required for web-based teleoperation. Since it is executed on the server side, all the computer operation system with a web browser can access the PHP programming pages at the web server side.

PHP is simple and straightforward to a new programmer. For example, the following codes are written in PHP and executed on server side. Users at the internet terminal can see "Hi, I'm a PHP script" instead of using a CGI program in C. The PHP code is intergraded with HTML codes. PHP can be used in most operation systems like Microsoft Windows, Mac OS X, RISC OS, Linux and UNIX. It is supported by all popular web servers today.

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
    echo "Hi, This is a PHP script!";
    ?>

  </body>
</html>
```

2.5.3 JAVA and JAVA Applet

JAVA is a high-level, object-oriented programming language which is similar to C++. JAVA has many features suited for web applications. It has a build-in application programming interface (API) that can handle graphics and user interfaces for creating applications or applets.

JAVA is platform independent and can be executed in any computer platform without rewriting the source code. Operation platform independent is also the main useful feature for a teleoperation system. Users all around the world can access the remote control interface on any operation platforms.

JAVA applet is a small application and can be executed on client side [17]. By using JAVA applets, a client can provide both a sophisticated interface and its own protocol to communicate with the server.

JAVA applet is a type of graphical component that can be displayed in the web browser. It is a rectangular area that can contain other components, such as buttons and text boxes [17]. It can display graphical elements such as images, rectangles, and lines, and it responds when the user clicks on the applet with a mouse. Users have to download and install the applet on the client side.

Some teleoperation system are using JAVA applet as user interface like PumaPaint project [9].

Chapter 3 Robot Hardware and Improvement

In this chapter the overall design, the hardware and software of the SuperScout robot will be presented. The hardware improvement made to the robot is as described.

3.1 Overall Design and Considerations

In order to build up a TeleScout system, a suitable http server, interface scripts between the http server and the proper control interface between the http server and the robot should be implemented.

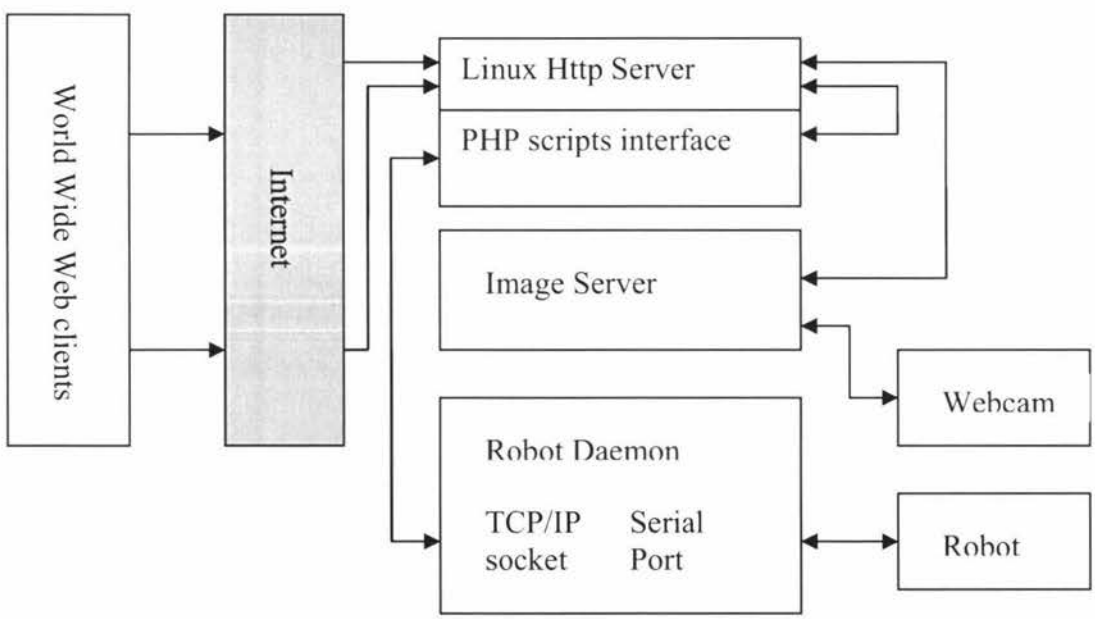


Figure 3- 1 System Architecture

The system architecture proposed is shown in Figure 3-1. The World Wide Web clients send requests to the Http server first. Red hat Linux 9 plus Apache are used to create this Http server. All applications including the homepage and the PHP scripts reside in the web server. Once WWW clients are verified by checking their correct user name and password, they can submit their quests to the robot server for controlling the mobile robot.

The dynamic script interface is programmed via PHP language. PHP is similar to CGI which is executed at the server side line by line but it is embedded within HTML codes. The PHP interface gets the requests for the robot being controlled from the clients and then sends those requests to the robot server. The robot server will translate those commands and send to the mobile robot by a serial cable. The Scout robot will execute the commands and send the results back to the user interface via the robot server. The user interface will generate the HTML code back to the internet client according to the PHP programming. The internet user can see the results on the web browser.

Linux Http server and the robot server run in the same computer. The robot-Nscout Super Scout robot is controlled by Nserver which is a program for controlling the robot in the robot server. The main task of the robot server is to send commands to get the sensor data from the robot. It has TCP/IP socket connection communicating with robot local host or IP address 127.0.0.1 on a specific port.

A standalone image server is also set up. A normal web cam is attached to capture the video stream of the mobile robot and broadcast it to the World Wide Web. Microsoft media encoder 9 is used in the project [18]. A separate computer is used to improve the broadcasting quality and reduce the load of the Http server.

3.2 Robot Controller

3.2.1 Introduction of Nomadic Super Scout Robot

Mobile robots are mainly used in the area of following path or avoid obstacles. For more sophisticated approaches, high level control strategies are required which involve calculation and programming [19]. A robot with the ability to sense the physical world and send the result back is needed for these researches. The Nomadic Scout is an ideal robot for navigational plans and control strategies.

The Nomadic Super Scout as shown in Figure 3-1 is a robot system with ultrasonic, tactile and odometry sensors. The robot is controlled by a high level

processor system which is a Pentium 233 MHZ industrial PC. The low level processor system is a Motorola MC68332. The low level system is connected with the high level PC system by a serial cable.

Nomadic Super Scout robot is mainly used for autonomous navigation researches. A number of different architectures for autonomous navigation have been proposed in the last two decades. They include hierarchical navigational architecture involving both high level (model & plan) layer, and low-level (sense & execute) layers; and behaviour-based architecture by combining simple behaviour-producing units into a complex behaviour [20]. More recently, fuzzy logic controller and neural networks have gained increasing attentions. Researchers can test and improve the control strategies on the Nomadic Super Scout robot.

Figure 3- 2 Super Scout Robot

3.2.2 The Nomadic Super Scout Robot Hardware Specification

The main parts of the robot are listed below and the hardware structure of the Super Scout robot is shown in Figure 3-2.

Host Serial Port: It is connected to a computer which is running client software. Default baud rate is 38400 baud, 8data bits, 1 stop bit, DCE.

"Joystick" Port: This allows user control of robot by moving the joy stick to different direction.

"Console" Port: This port offers a text-based interface to the Scouts command set (listed in the Language Reference Manual). It powers up at 9600 baud, 8 data bits, and 1 stop bit.

VGA Port: It is used to connect a VGA monitor.

Keyboard and Mouse Port: For plug-in mouse and keyboard to control the onboard PC.

Also there are powers on and power off buttons and status LEDS located at the side panel. The other hardware features as shown in Figure 3-3 are briefly described as follows.

Power System: Two 12V 17 AMP lead-acid batteries are used.

On Board PC: The PC is PCM 5862 Pentium 233 industrial PC with two serial ports that are used to connect with the low level robot controller.

Sensors: Two kinds of sensors are installed on Nomadic Super Scout system, bumper sensors and sonar sensors. The former is used to detect the collision and the latter is used to detect the obstacles around the robot. The bumper sensors are around the perimeter of Scout. The sonar sensors look like small mirrors around the robot shown in Figure 3-1. There are a total of 16 sonar sensors which are numbered from 0 to 15 with 0 being in front of the robot (the side opposite the batteries) and the rest counted in counter-clockwise.

Odometry: The scout keeps running integral of its current position in world coordinate after power on or the most recent movement command.

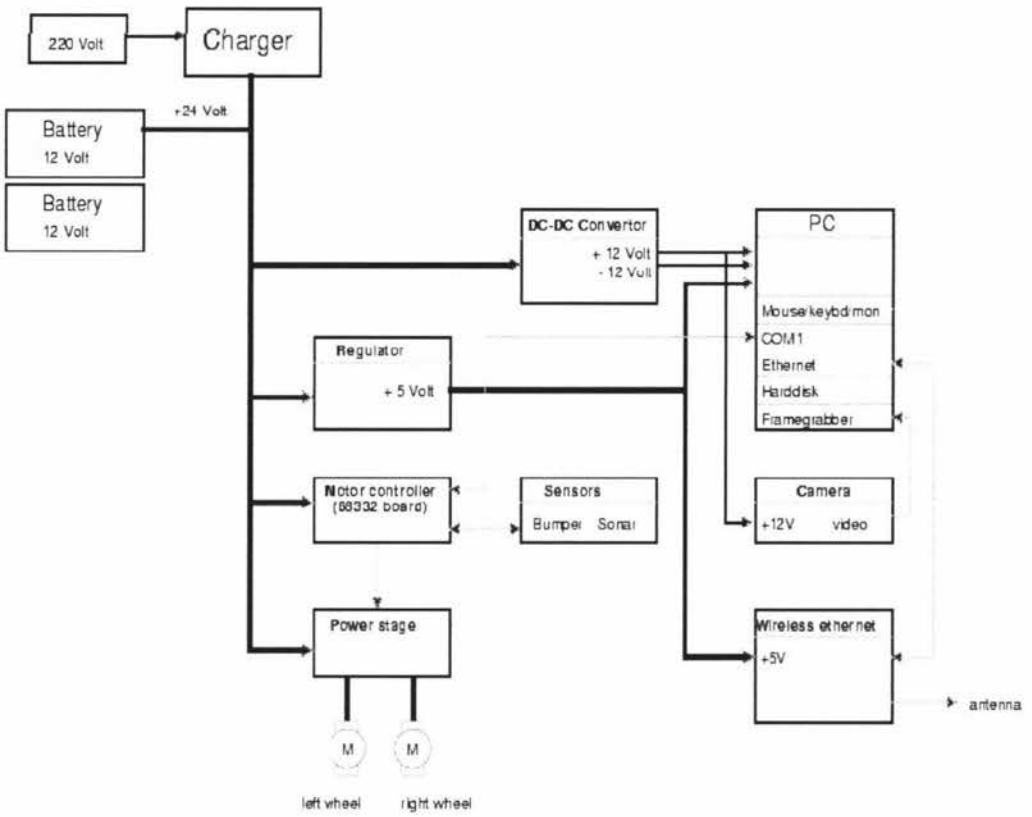


Figure 3- 3 SuperScout Hardware Structure [21]

3.3 Nomadic Super Scout Robot Software Environment

The control software for Super Scout robot is based on Linux system. Local user can program the robot in C. There is also a graphic interface for the robot which is Nserver. In Nserver user can easily switch the robot mode between real or simulation. If it is real, the command to the robot will be sent to the robot hardware directly. When in simulation mode, the robot commands only be executed in the Nserver.

3.3.1 The Graphic Interface

The operation system for the robot server is Linux system where Red Hat 6 Linux is used. Robot software has to be installed since it acts as the server for the Nomadic's Host Development environment. All the files needed for the robot can be found at http://nomadic.sourceforge.net/production/scout/.

The robot server's software framework for local control is shown is Figure 3-4. For safety reason, there are two modes that the Nserver communicates with the robot, simulate robot mode and real robot mode. Local user can change the operation from simulate mode to real mode or vice versa by selecting the mode at robot menu which is in the robot item window. The commands from the client program can be sent to either simulator or real robot by the mode selection in the graphic interface. The World Presentation between the Graphic Interface and Client Program is the physical world map in which the robot moves around. The Robot program is a session that provides TCP/IP connection with the real robot.

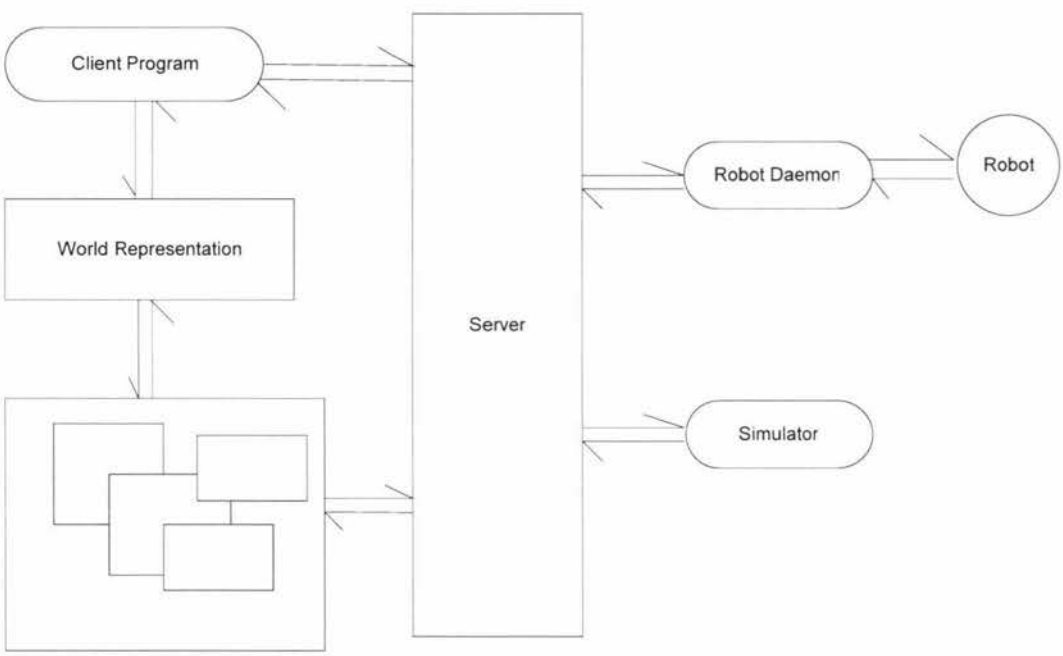


Figure 3- 4 Graphic Interface

Nserver, the main program to communicate with the robot, is different from the user interface for the TeleScout system. Internet users will send requests to the HTTP server first, and then the user interface (dynamic PHP scripts) will forward the requests to the robot server. The robot server can perform the robot operation commands like moving and sensing, and can send the coordinator data and sensing data back to the robot server according to the users' request. The user interface and the HTTP server will generate the HTTP code and send it back to the internet users.

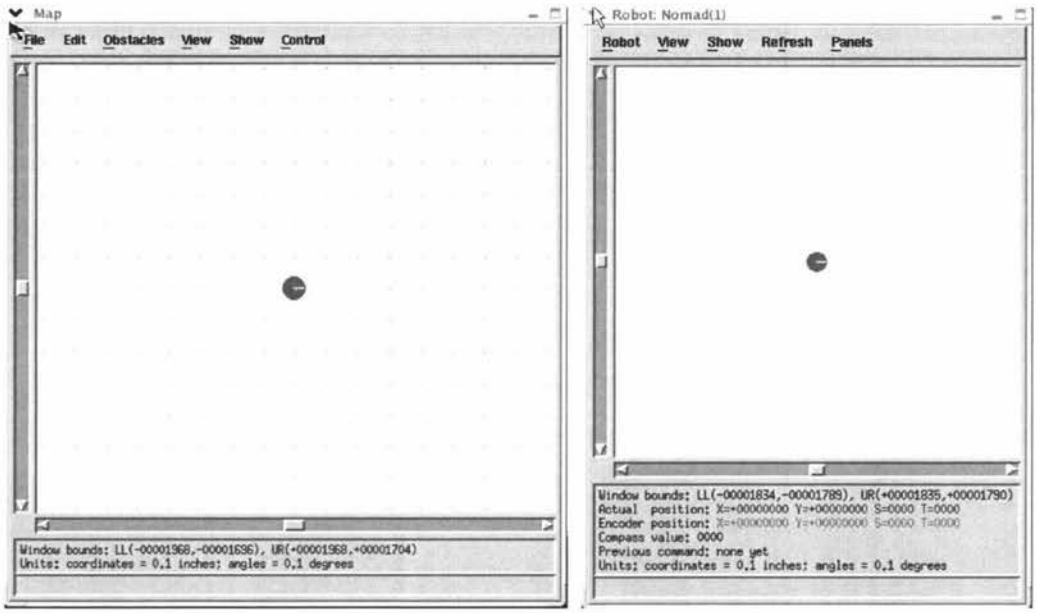


Figure 3- 5 Nserver

Local users can interactively control the robot in robot window in Figure 3-5. They can run the command centre by select Panel – command line at the menu from robot window. The command centre is shown in Figure 3-6. Local users can move the real robot by inputting commands into the command centre. They can initialize the robot, move the robot backward or forward, turn on and off the sonar sensors etc, once the robot is connected with the main control PC and is turned into real robot mode. This is a way to checking if the robot is successfully connected or not.

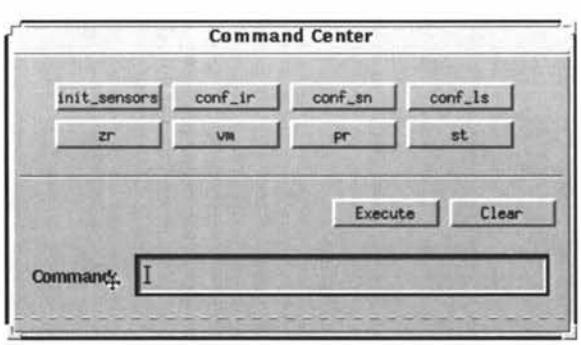


Figure 3- 6 Command Centre

At the bottom of the robot window is shown the information about current robot position which includes the robot coordinates, steer direction in degrees. There are many other functions available in this window. Local user can select to show the robot trace, the robot's actual position, and the robot's sensor data. These are the

functions also made available in the TeleScout system so that internet users can read and download robot's coordination data and sensor data on their web page. What the users need to do is to put the C code provided in the TeleScout Website in their own program in order to get the proper robot data. The details about this will be described in later chapters.

The map window allows the user to define and modify the map where the robot moves. The map can be defined before running the program or can be interactively modified when program is running according to the C program. The robot world is an abstract coordinate system and the dimensions are set in the world.setup file.

3.3.2 Nomadic Super Scout Robot Programming Concept

Figure 3-7 illustrates the programming concept for Nomadic Super Scout robot. Local users can program the robot by writing C program including Nclient.h header file and link the program with Nclient.o or Ndirect.o. There are two modes available for the robot – Direct Mode and Client Mode. As Shown in Figure 3-9, Application 1 is compiled and linked to Nclient.o which means it is in Client mode. The program will communicate with Nserver. User can select to run simulate or real robot in Nserver. While in Direct Mode, Application 2 is compiled and linked to Ndirect.o. The application will communicate with robot directly by the serial cable. Nserver has no effect in this application.

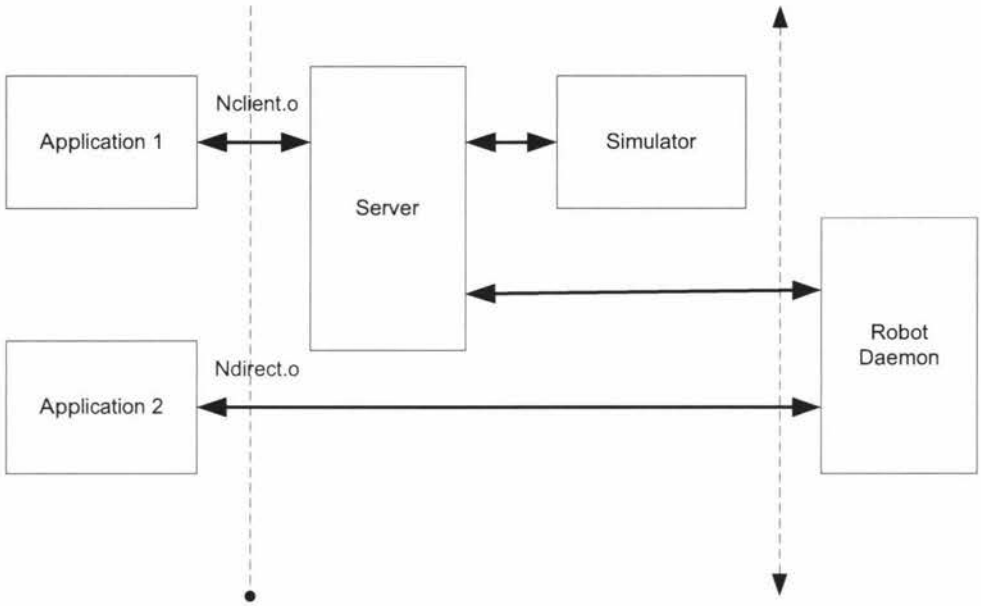


Figure 3- 7 Programming Concepts

The Direct Mode and Client Mode are fully compatible. But a very important thing here is that in Direct Mode, those functions provided by the server are not functional. For example, in Direct Mode, user can't draw obstacles at the map window in Nserver. Server commands [21] are only functional in Client Mode with Nserver being running.

In Client Mode, local user can change the robot running mode from simulation to real and from real to simulation in Nserver. Direct mode is always in real robot mode. The commands are sent to the robot by serial cable directly via robot program.

3.3.3 The Global Vectors

To navigate a mobile robot requires various sensors data and the robot's motion states while a program is being executed. For the Nomadic Super Scout Robot, the coordinate data, the robot speed, the sonar sensor data and the bump sensor data etc can be obtained by an application program through a global array, called "State Vector" [21]. The structures of these state vectors are listed in the Figure 3-10.

	Name	State Vector	Name
0	STATE_SIM_SPEED	[illegible]	SMASK_POS_DATA
...	...	...	...
17	STATE_SONAR_0	sonar data #0	SMASK_SONAR_0
18	STATE_SONAR_1	sonar data #1	SMASK_SONAR_1
19	STATE_SONAR_2	sonar data #2	SMASK_SONAR_2
...	...	...	...
32	STATE_SONAR 15	sonar data #15	SMASK_SONAR_15
33	STATE_BUMPER	bumper data	SMASK_BUMPER
34	STATE_CONF_X	x position	SMASK_CONF_X
35	STATE_CONF_Y	y position	SMASK_CONF_Y
36	STATE_CONF_STEER	steering angle	SMASK_CONF_STEER
...	...	...	...
38	STATE_VEL_RIGHT	translational velocity	SMASK_VEL_TRANS
39	STATE_VEL_LEFT	steering velocity	SMASK_VEL_STEER
...	...	...	...
41	STATE_MOTOR_STATUS	motor status	--
44	STATE_ERROR	error number	--

Figure 3- 8 State Vectors

3.3.4 Robot Commands

To program the robot includes mainly the following two steps. First is to establish the communication with the robot which can be in simulating mode or real mode. Initializing the robot is necessary for preparing a new robot running. Secondly is to send motion commands to move the robot in predefined motion and to send sensor commands to turn on the sensors and to get the sensor data back.

The Nomadic Super Scout Robot can be driven by following various commands which have already been defined.

Communication Command: To connect to server or directly to the robot by linking to Nclient.o or Ndirect.o.

Motion Commands: To move the robot at different speeds, for certain distance and in a direction.

Sensing Commands: To configure different sensor and obtain the readings. The sensor data will be stored in State Vectors.

Server Commands: To be active when a robot is in Client Mode and not active when robot is in Direct Mode. They include server status, the drawing of a robot and environmental map etc in Nserver.

3.4 Improvement of Robot Hardware

As mentioned in Chapter 1, one of the main tasks in this project is to upgrade the robot hardware. The problem with the existing robot hardware is that the main controller on-board PC is too old to run the robot control program in the latest version of Linux. It took about 8 to 10 minutes to get it started. Furthermore, the hard disk did not work properly and finally failed. Thus, the onboard PC must be upgraded first. The PC has the following specifications: 233MHZ CPU, 32MB RAM, and 10GB hard disk. As a matter of fact, if all the hardware had worked well, this PC should have been able to run Nserver without any problem as Linux does not require high-performance hardware.

The main reason for the PC working slowly is that the hard disk slows down the whole system. It is a mini hard disk (2.5inch) which is used in a laptop. The hard disk is fixed into a PCI card which is plugged into a PCI slot on the PC motherboard as seen in Figure 3-9. It might be due to a consideration of saving battery power since the robot is driven only by two 12 Volt 17 amp lead-acid rechargeable batteries. This mini hard disk does not use the ULTRA DMA33 hard disk connector on the motherboard.

The data transfer rate for a normal hard disk in ULTRA DMA33 mode is 33MB/S [22] and it is optimized for data transferring with low CPU load. The fastest data

transfer rate for a PCI device theoretically can be up to 132MB/S with 32-bit data path, 33MHz clock speed [23], but it will be a heavy load for the P233MHZ CPU.

A way to improve the robot system's performance is to upgrade the on-board PC first. For the TeleScout system, it would be better to detach this computer from the robot for saving the battery to maximize its running time which is crucial for all the autonomous mobile systems. It will be good for the users of the robot, too. With the existing on-board PC, users have to plug a VGA monitor, a mouse and a keyboard into the connectors atop the robot panel to compile and execute their programs. Before switching to the real robot mode, they have to unplug the above items in order for the robot to move around freely. This is of course very inconvenient.

3.4.1 Upgrade the Main Control PC

An IBM laptop computer was used to replace the onboard PC. It is IBM ThinkPad i1400 with Intel Celeron 500MHz CPU, 256MB RAM, 6GB HDD. The first consideration was that a laptop could be mounted on the top of the robot and the serial cable could be connected with the low level controller inside the robot. After upgrading the PC, it was found that a wireless connection between the PC and the robot low level controller would be established so that the robot would be totally free to move around.

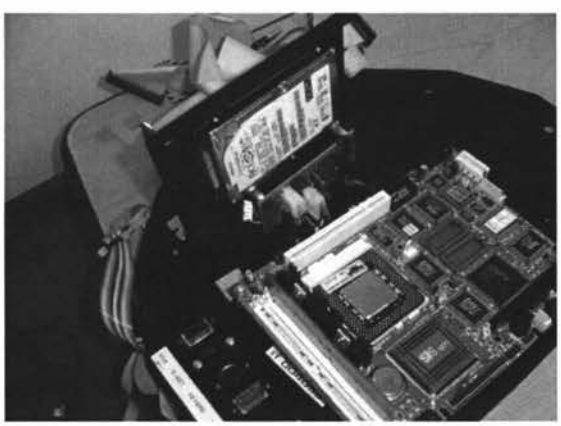


Figure 3- 9 Hard Disk

The next step is to set up software environment for the laptop. This includes installing Red Hat Linux and installing softwares specific to the Super Scout Robot.

The Scout operation system is based on Red Hat Linux 6.1. This Linux [24] was downloaded from internet and installed without any problem except the network drive. The laptop used an IBM 10M LAN network card which is too old to find its drive for Linux. A PCMICA network card was then secured for the installation.

The installation procedure was quite smooth. Most of the default minimal network installation selections were taken. A minimal installation used here was to minimize the system booting time and the system resources consumed at run time. For example, it is generally not necessary to install the printer support, the mail sending daemon or the game console mouse server [25].

To install the software package for the robot hardware controller, the scout-robot-1.3-1.i386.rpm package was downloaded from internet to usr/src/redhat/RPMS/i386 and the scout-robot-1.3-1.src.rpm package to /usr/src/redhat/SRPMS/. This was done by installing the binary package as follows:

```
# cd /usr/src/redhat/RPMS/i386
# rpm -i scout-robotd-1.3-1.rpm
```

The next software package needed to install is Nomadic's Host Develop Environment. It is the User Graphic Interface for local robot users. It is also required for remote internet users of TeleScout Systems. They need the Nserver program to control the robot and they need the object files and functions to compile their own C programs to control the robot.

The Host Develop Environment package was downloaded and uncompressed into a folder. There are three folders – server, sample and client. A license file has to be placed in the same folder with Nserver. Since the Nomad robot company no longer exists, there is an open license file downloadable at sourceforge.

The server folder contains an executed program Nserver, world.setup and robot.setup. The license file should be copied to this folder. World.setup sets up the window of the graphic interface of the Nserver. It also sets up the communication port which is used for a client to connect to the server and the default configuration file for robot. Robot.setup files set up the graphic parameters for the robot window and the simulation parameters they also set up the connection parameters for the robot.

The details for the setups in world.setup are,

```
[connect]
serv_port       = 7019;
[robots]
setup_files     = robot.setup;
```

The sere_port 7019 stands for that the program executed in the Nclient mode will communicate with the robot server at port 7019.

The connection part of robot.setup is as follows,

```
[connect]
conn_type       = serial    ; tcpip or serial
machine         = 127.0.0.1 ; for tcpip
tcpip_port      = 4000      ; for tcpip
serial_port     = 0         ; for serial: 0 = /dev/ttyS0, 1 = /dev/ttyS1, etc.
serial_baud     = 38400     ; for serial
retrans         = off
normal_timeout      = 10.0 sec
special_timeout     = 10.0 sec
server_delay    = 0.002 sec; the avg roundtrip delay between client and server
robot_delay     = 0.01 sec  ; the avg roundtrip delay between client and robot
```

The HTTP server will communicate with robot server via local host or IP address 127.0.0.1 which is the default host name and ip address for this computer. Since the web server and the robot controller are in the same computer so that only local host is used.

In the client folder are there an uncompiled sample program, the object files and head files for the robot programming.

After uncompressing these three folders (ie, Nserver, world.setup, robot.setup) to /Nscout, the installation process of the laptop's software envoriment was complete and ready to control the robot. It was tested by executing a simple program at the simulation mode and the simulated robot in the robot window was moving happily. This testing said the software installation had been successful.

3.4.2 Serial Connection of the Robot and the Control Laptop

The connecting the laptop to the robot controller is achieved by means of serial connection. The laptop has only one male com port but there are two com ports at the previous PC before being upgraded. One is male and the other is female. The serial cable is connected to the low level robot controller from the female connector to the female com port on the previous PC. Both ends of the serial cable are male. To connect from the laptop to the robot controller a male-to-female adaptor was used. However, it did not work out as desired. It was found out that operation commands were displayed in the Linux terminal window but timed out after a while. The commands were not sent from the laptop to the robot controller.

A further investigation showed that the port used in robot controller is a female port which is wired at DTE which is normally wired as male connector. Data rate of this port is 38400 baud, 8 data bits, 1 stop bit and no parity [25]. In order to find a possible solution for the connector, it requires an understanding of what DTE connection is and how the serial cable is connected.

RS232 is a voltage loop interface for two-way (full-duplex) communication represented by voltage levels with respect to system ground (common) [26]. DTE

is "Data Terminal Equipment" which connects to a network through a DCE device. DTE devices include Computers, Serial Printers and most devices that are not used to extend communications. **DCE** stands for "Data Communications Equipment" which provides a physical connection to a network and forwards traffic. DCE includes devices that are intended to plug into a DTE port and devices that extend communication like a modem.

The cable between DTE and DCE wires goes straight from PIN to PIN which is straight cable wiring. DCE to DCE and DTE to DTE wires are required to be crossed [27]. DCE to DCE wiring is called tail circuit cable and DTE to DTE wiring is called null modem cable. The female com port on robot controller and the female com port at the laptop are both DTE wiring, so that the two DTE devices may be connected easily.

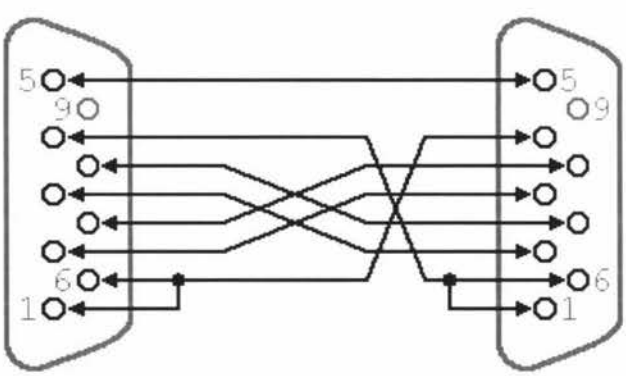


Figure 3- 10 Null Modem Wiring

A null modem cable was made as connected in Figure 3-10 and shown in Figure 3-11.

The cable was tested for connecting the robot controller with the laptop. At the robot Command Centre in the command line mode, a few robot commands were sent to the robot and the robot moved, which means the wiring of the cable is correct and the connection of the robot and the laptop is successful. The laptop can now control the robot by a serial cable which is wired as null modem with both sides male connector.

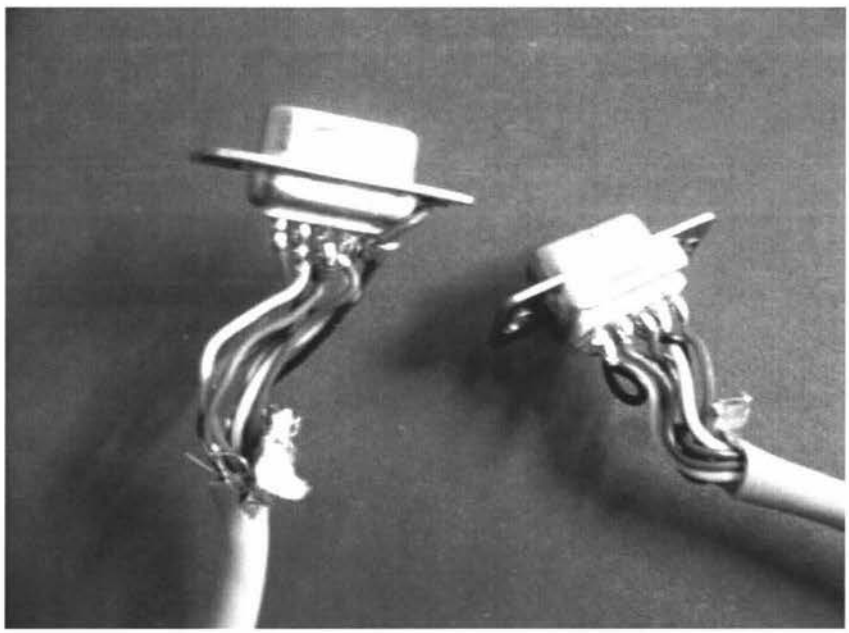


Figure 3- 11 The Null Modem Cable

So far so good but it still requires the laptop to be placed atop as there is a cable between the robot and the laptop. The next step is to find a wireless communication between the robot low level controller and the laptop.

3.4.3 The Wireless Communication between the Robot and Control PC

The wireless communication was to be placed between the PC and the low-level controller. the control PC was mounted inside the robot and another PC outside the robot could be connected with the inside PC via wireless network card before the hardware was upgraded,. In this case, the outside PC would be the HTTP server and the inside one be the pure robot control PC. Now that the laptop is for both control PC and HTTP server PC, the cable between the low-level controller and the PC must be wireless.

The solution was to use a "Bluetooth Aircable Serial" [28] as shown in Figure 3-12.

Figure 3- 12 Bluetooth Airecable Serial Adaptor

Bluetooth is a standard for short range low power, low cost wireless communication that uses radio technology [29] since in 1994 Ericsson [30] first carried out Bluetooth technology. Embedded Bluetooth capability has been widespread in various kinds of devices, such as PDA, data peripherals like cameras and keyboard, and audio peripherals like headsets and speakers etc.

The Bluetooth Airecable Serial Adaptor is manufactured by AIRCALBE INC [28]. It is a kind of embedded Bluetooth devices. There is no need to install any software to configure the wireless connection. Two Aircable Serial modules can be paired by a button which is accessible through a hole in the device as shown in Figure 3-12. Aircable Serial is Bluetooth 1.1 certificated and is compatible with all other Bluetooth 1.1 devices. It is also compliant with RS232 standard. What was used in the project is a DTE one with 9 pin female adaptor connect and a DCE one with 9 pin male adaptor.

The Aircable Serial Adaptors is in cable mode of configuration, that is, the pair of Bluetooth devices are connected together without any interference by any other Bluetooth device. The Aircable Serial Adaptors needs 5-15V DC power. A 5V DC power adaptor for the female one is connected with the laptop and 12V DC for the male one inside the robot is used as the DC power for the pair of Bluetooth adaptors.

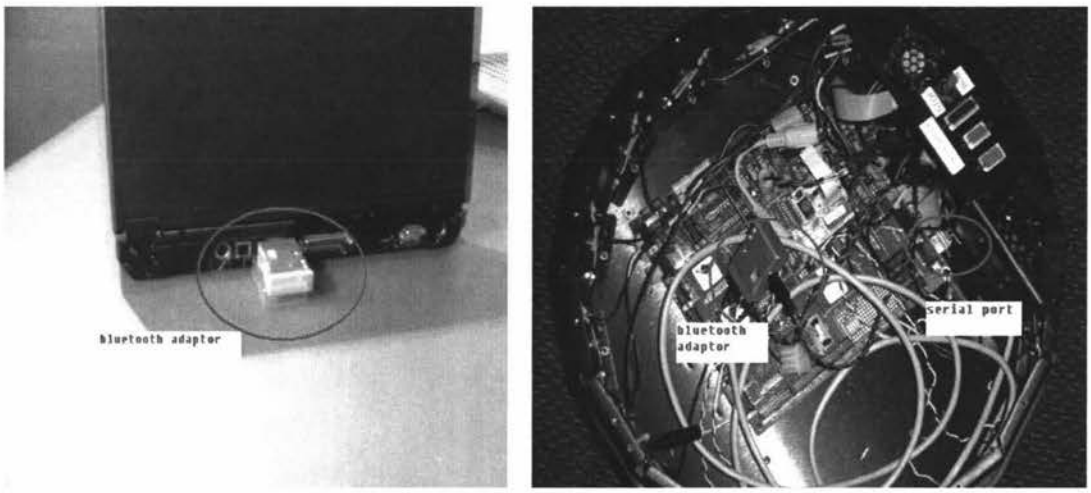


Figure 3- 13 Bluetooth Connection

The connection is shown in Figure 3-13. The procedure of the connection is as follows,

- Plug the female one into the com port at the laptop;
- Plug the male one into the com port at the robot controller;
- Press and hold the paring button for the female one for 5 seconds. The Aircable Serial module automatically configures itself and starts in the slave mode;
- Do the same thing on the male one at the robot side again;
- Short click the pairing button on the male one again to change it mode to master;
- The success of the connection between the female one and the male one can be checked by the status LED.

Initial control to the robot was performed by manual command at the Command Centre from the Nserver at the laptop. The result was that the robot did not respond at all. It seemed nothing wrong as the pair of devices were connected successfully as indicated by the LED status. The wireless connection between DTE and DCE devices is shown in Figure 3-14. It was found eventually that both sides of the wireless connection are DTE connection. The female one which is a DTE adaptor is connected with the laptop without problem, but the male one which is a DCE connector needs a null modem wiring to connect with the robot controller as shown in Figure 3-15. The easiest way was to connect one side of the

null modem wiring cable to the robot controller and to connect the male Aircable Serial Adaptor to the other side of the null modem cable. The testing run was successful because the robot responded the commands from the laptop and moved around freely without any external cable attached to it.

Up to this point, the hardware improvement has been completed successfully and the robot is ready for the TeleScout System.

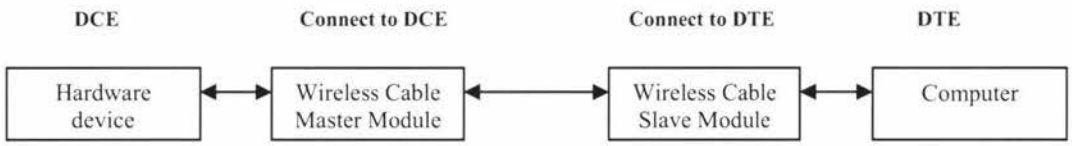


Figure 3- 14 Wireless Connection between DTE and DCE

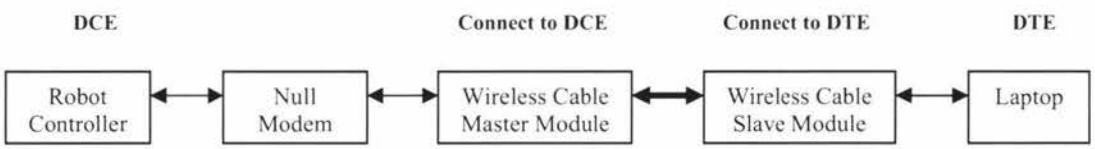


Figure 3- 15 Wireless Connection between DTE and DTE

Chapter 4 Server and User Interface for Web-Based Operation

4.1 Internet and Http server

Internet is a collection of thousands of network connected by dedicated special computers called router which is used to forward data packets between two networks[31][32]. Those individual networks belong to private owners who connect their network together for different purpose like long-term business or education.

Different computers in the internet are communicated by a standard protocol – Transmission Control Protocol /Internet Protocol. Every computer in the network has a unique IP address defined by TCP/IP protocol. So an internet user can find a specific computer by that computer's IP address. TCP is responsible for correct data delivering from client to server. Once transmission error is detected, like data lost, a retransmission will be triggered. IP is responsible for delivering the data from computer to computer in internet.

The World Wide Web is hypertext based, providing many kinds of internet services.
A World Wide Web user can retrieve documents, view video, view images, listen to sounds and voice over internet [33]. HTML ((HyperText MarkupLanguage) is the programming language on the World Wide Web and there are many links on HTML web pages for connecting different internet services.

For a teleportation system, internet provides the way for as many as possible users to access the hardware devices. They don't need to do the operation at the side of the hardware devices. They can just sit in front of the computer and click the mouse and wait for the result. It is a low cost solution for various users to access different hardware. The TeleScout System is a low cost, tele-education purposed system. Different users can access the Super Scout robot by different level. For

new users who are new to a mobile robot or new to Super Scout robot, they can just access the robot by internet and run sample programs to get the basic idea about what Super Scout robot is. For high level users, they don't have to have Linux operation system to compile their C code locally; they can write their C code and compile via internet; and they can run the program to see how the robot is running and get different sensor data.

4.2 Apache Server on Linux

Telescout is a telerobotic system for internet users to access Nomadic Super Scout robot over internet via a web browser. A web server is necessary for responding the internet users' HTTP requests. As the robot control onboard PC was replaced by a laptop, the laptop is the HTTP server and the robot controller for the internet operation. Since the robot operation system is based on Linux, the web server is Apache Server which is an open-source HTTP server for modem operation systems including Microsoft Windows and Linux [34].

Apache Server version 2.0 is used in Telescout system. This Apache Server package was integrated with Linux 9 software package. When installing the Linux, it can be easily installed just by selecting server installation. It can also be installed by downloading the free software package and installing it.

Before Apache, the most popular server software was the public domain HTTP ("daemon") httpd developed by Rod McCool at the National Centre for Supercomputing Application. After that, many Webmasters began to develop their own patches for this server software and they exchanged their improvement via email.

Finally the Apache project group appeared. The Apache web server software is managed jointly by a group of volunteers located around the world. The latest version of Apache Server is 2.2. Starting the software is quite easy in Linux. Once the Apache Server software package is installed, the httpd which is the Apache Hypertext Transfer Protocol (HTTP) server program will run as daemon to execute in the background to handle requests.

The HTTP service in Linux 9 is not started by default after installing it. The HTTP service has to be started manually shown as Figure 4-1 by selecting Main Menu – System Settings – Server Settings – Services and tick Httpd and closing this window.

Figure 4- 1 HTTP Service Setup

The next step is to setup the Apache Server shown as Figure 4-2. The default value can be used except changing virtual host – site configuration – default directory page to index.html. So the default home page of the Apache HTTP server is set to index.html in directory /var/www/html

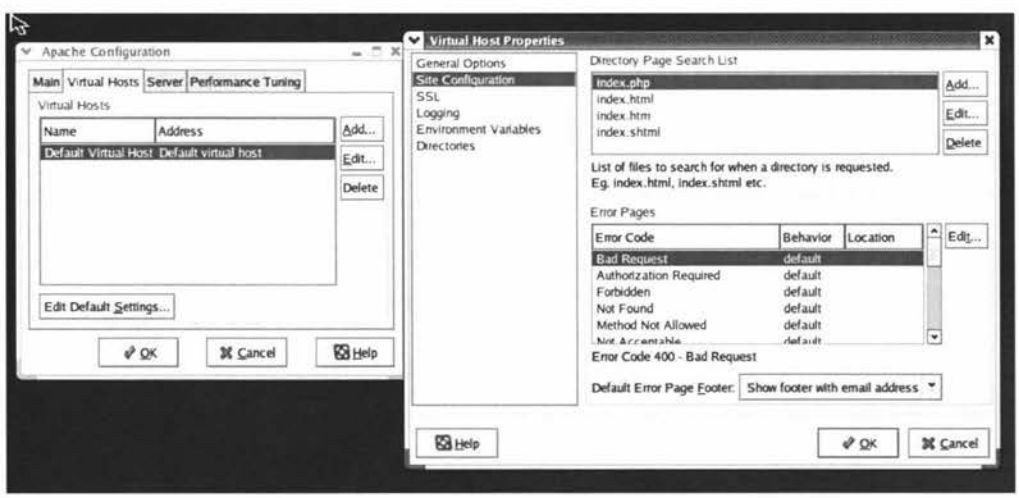


Figure 4- 2 Apache Server Setup

By now, the Apache HTTP Server is ready and it can be checked by entering 127.0.0.1 or local host to the address line in the web browser. The default

index.html page of Apache Server welcome information will be displayed in the web browser.

4.3 Image Server

The video broadcasting is necessary for a teleoperation system. The internet users will access the hardware via internet. They need to get the feedback to continue their operation via internet. The feedback will include data feed back which can be displayed or downloaded from the web browser, and live images of the movement of the hardware devices. For some entry level user who is not familiar with the hardware devices, they will be interested in the image broadcasting.

A dedicated Image Server is used in TeleScout system for improving the performance of the HTTP server and the Image server. The Apache HTTP is together with the robot server on the laptop computer. It is better off to take the image server out of it since the laptop is not powerful computer system for heavy load computation. A normal web cam is used as it is a low cost solution for the live video stream broadcasting. In the PumaPaint project, a JAVA video applet was used for video broadcasting.

4.3.1 Image Server Setup

Hardware setup

A PC with windows XP operation system is used to be the Image server.

Webcam is a real time camera whose images can be accessed by World Wide Web [35]. Logitech QucikCam for notebook is used for the live video capturing as shown in Figure 4-3.

Figure 4- 3 Logitech QucikCam for Notebook

Product Specifications

- Video capture: Up to 640x480 pixels (VGA CCD)
- Still image capture: Up to 1280x960 pixels, 1.3 mega pixels
- Frame rate: Up to 30 frames per second

USB connection is used to connect with PC. Installation is quite straightforward – just plug the webcam into a USB port of the image server and follow the window instructions to complete the installation.

Software setup

The webcams can be used in many applications in Windows System. For example, two MSN messenger users can use webcams to see each other from internet. This is a kind of point to point communication. Both sides of the webcam are equal. They should have the same application to receive the video stream. For using a webcam in teleoperation system, it is a bit different. It is a kind of broadcasting. There should be a video server connected with the webcam. There will be an application acting as server to respond the requests from World Wide Web. The webcam captures the image and broadcasts the video stream to the internet. The users, not only one user can watch the live video at the same time but they need different applications to receive the video stream depending on the image server application type. Like PumaPaint, a JAVA applet plug-in has to be installed in the web browser before watching the video.

Webcam is not only very popular and cheaper in the market to broadcast the live video stream, but also a very economic way to build up a mini monitor system.

There are many other live webcams broadcasting live video stream around the world. For example, a webcam at Scott Base [36] where the still image is updated every 15 minutes and eleven webcam broadcasting at Times Square [37].

The problem is that there is no server application sold with the normal webcamera. There is only a point to point webcam application in the installation CD. A free video broadcasting software - Microsoft Windows Media Encoder 9 [38] can be used to broadcast the live video stream. Windows Media Encoder is for audio and video capturing with different sound and video quality. The software package can be downloaded from www.microsoft.com freely.

Windows Media Encoder Features

Windows Media Encoder can provide the services listed as follows,

- Capture live video and audio content for local playback. Capture live content and encode it to a file for users to download and play it locally.
- Capture live video and audio for streaming. Capture live content and encode it to a Windows Media file for broadcasting over network.
- Convert film content to video and convert various video formats to Windows Media Format.
- Broadcast live video and audio content and push a stream to a publishing point on a server running Microsoft Windows Media Services.

4.3.2 Optimize the Broadcasting

The feature of the webcam the project requires is its ability to broadcast live video and audio content. There are two encoding formats that can be selected in this software. They are VBR and CBR encoding.

To reduce the bandwidth needed for video broadcasting, video is always compressed before transmission. The compression can be classified into variable bit rate(VBR) and constant bit rate (CBR) compression according to whether the output rate of the encoder is variable or constant [35]. VBR encoding is to encode the video or audio contents for downloading and playing locally or on a CD or DVD player. It is the most effective for encoding contents with a mix of simple

and complex data like a video with fast and slow scene. CBR encoding is to encode video or audio contents for network broadcasting. The bit rate remains constant within a small window of time set by the buffer size and it keeps the transfer rate constant no matter how the video picture changes. This is the advantage of CBR encoding for network broadcasting. The disadvantage of CBR encoding is that the compression quality is not constant since some of the contents are more difficult to compress than others. It results in that quality in some parts of CBR stream is lower than other parts. So CBR is the idea encode format for the live video stream broadcasting.

Since this is for live video broadcasting, the broadcasting delay and the video stream quality have to be well balanced. The buffer can be set in Windows Media Encoder to ensure the quality of live video stream broadcasting. Buffer is a temporary storage location for data information being sent or received [39].

A broadcast delay is the difference in the time between the point when live audio and video is encoded and when it is played back, and it is created primarily by the buffers that store digital media data [40].

When broadcasting real time contents, the contents in buffer change constantly. Real video data are sent to the buffer and the old data are sent to the end users by network. As shown in Figure 4-4, there is a five seconds' data buffer for video stream where data are sent to the buffer at zero seconds and sent out in five seconds.

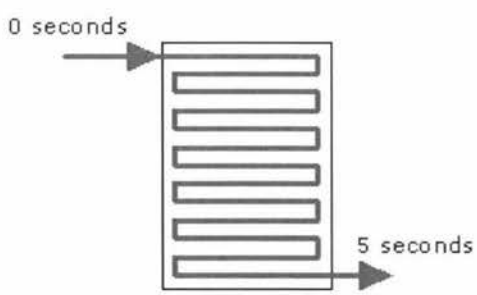


Figure 4- 4 A Five Seconds' Buffer in Video Stream

Buffers are very important for the live streaming media content broadcasting. It enables the Media Encoder to provide high quality video stream. But the buffer

size is bigger, the delay would be longer. As shown in Figure 4-5, the delay of the video stream is five seconds at the encode side plus five seconds at the server side and plus five seconds at the end user side, totally fifteen seconds. For the TeleScout system here, the delay has to be minimized since the end users would watch the live video and operate the robot. The end user can't control the robot well if he could only see the robot moving 10 second after inputting the commands. The only delay can be controlled in the project is the encoder delay so that minimizing the buffer size can reduce the broadcast delay. The smallest buffer size for Windows Media Encode 9 is one second.

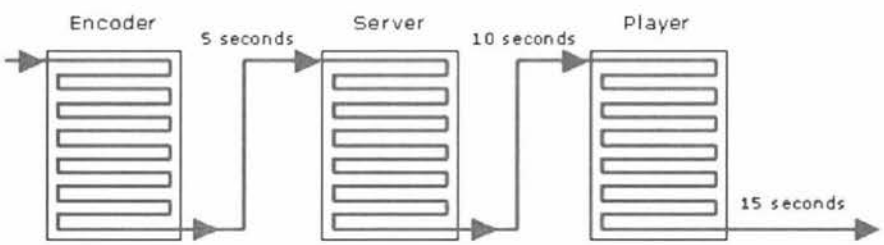


Figure 4- 5 Delay in Broadcasting

Therefore, Windows Media Encoder can be set up as follows,

- Start Windows Media Encoder and select new session and Broadcast a live event at the left side of Figure 4-5.
- In the next window select the Logitech QucikCam for notebook as the video device.
- Select Pull from encoder as broadcast method.
- As the right side of Figure 4-6, setup the broadcast connection. The HTTP port is 8080 and the IP address for the LAN connection.
- Setup the encoding option at the next window. Select VBR for Video and 282Kbps which is 29fps at 320x240 video sizes.

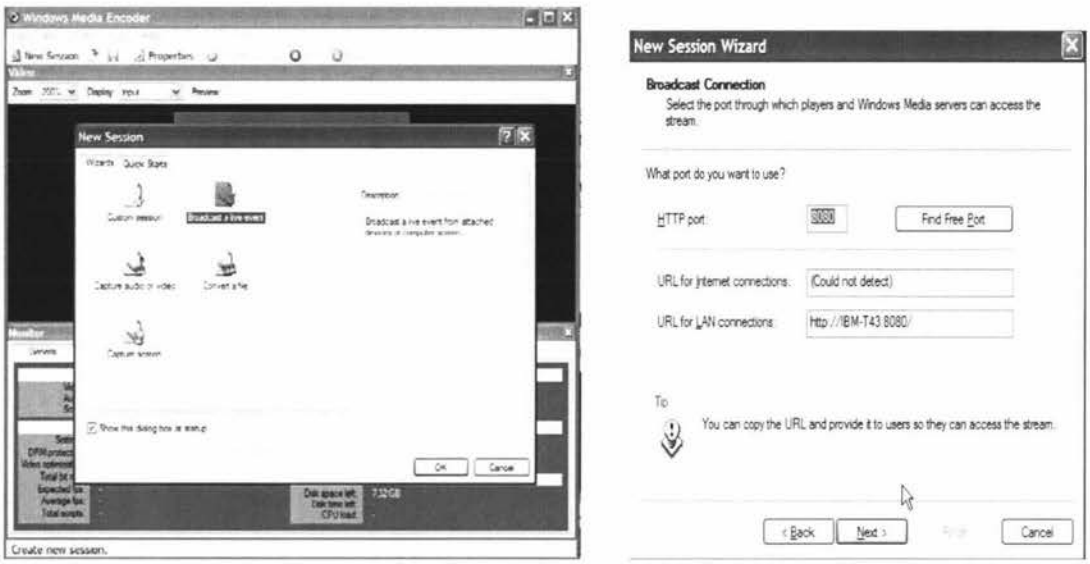


Figure 4- 6 Windows Media Encode Setup

The last step is to setup the Encoder buffer size to reduce the broadcasting delay as Figure 4-7.

- In Session Properties on the compression tab, click Edit custom encoding
- Click the bit rate tab
- In the buffer size, change the buffer size to 1 which is the smallest buffer size
- In the frame rate, change it to 20 fps

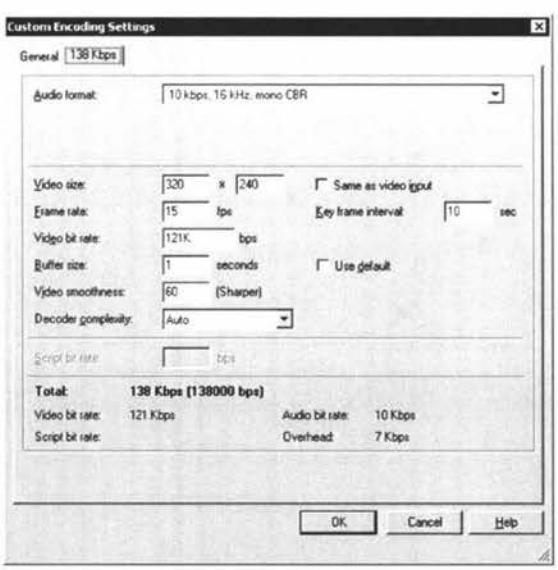


Figure 4- 7 Buffer Setting

Now the Windows Media Encoder is ready for broadcasting live video stream for

the Nomad Super Scout Robot over internet. The end user needs Windows Media Player 7 or later version to receive the video stream.

4.4 The User Interface

A user interface has to be implemented for a teleoperation system over internet. The remote user can communicate with the web server and the hardware devices via the interface. This includes sending commands and getting the feedback from the hardware devices. Since it is a teleoperation system, normally a visual option has to be attached to the user interface. Remote users can watch the hardware device online and perform the operation.

The developed user interface for the TeleScout system is shown in Figure 4-8. It is written in PHP scripts which are executed at the server side only. There is no need to install any plugin locally so that only a normal web browser is required to access the TeleScout System.

Different authorizations are given to different level users which include user name and password to gain the control of the robot.

Internet remote user can execute two sample programs – run robot and run sensor, once they are assigned a proper user name and password. Before running the programs, they have to specify a few input parameters, such as Max Velocity (at which the robot runs), Shortest Distance (which is the shortest distance between the robot and the obstacles) and the Running Time.

At the bottom right, there is a function for end users to upload their own C code to the HTTP server. They can use it to compile and run their own program. They can also choose either view image, sonar data or bumper data. Once anyone is chosen, there will be a sample code displayed at the bottom right. End users can simply copy and paste it into to their C program so that their program can be executed and remote HTTP server will show data they need.

The feedback from the robot is decided by what is specified in the program by the

remote user. Those sensor data can be downloaded from webpage so that they can be analysed for some purpose.

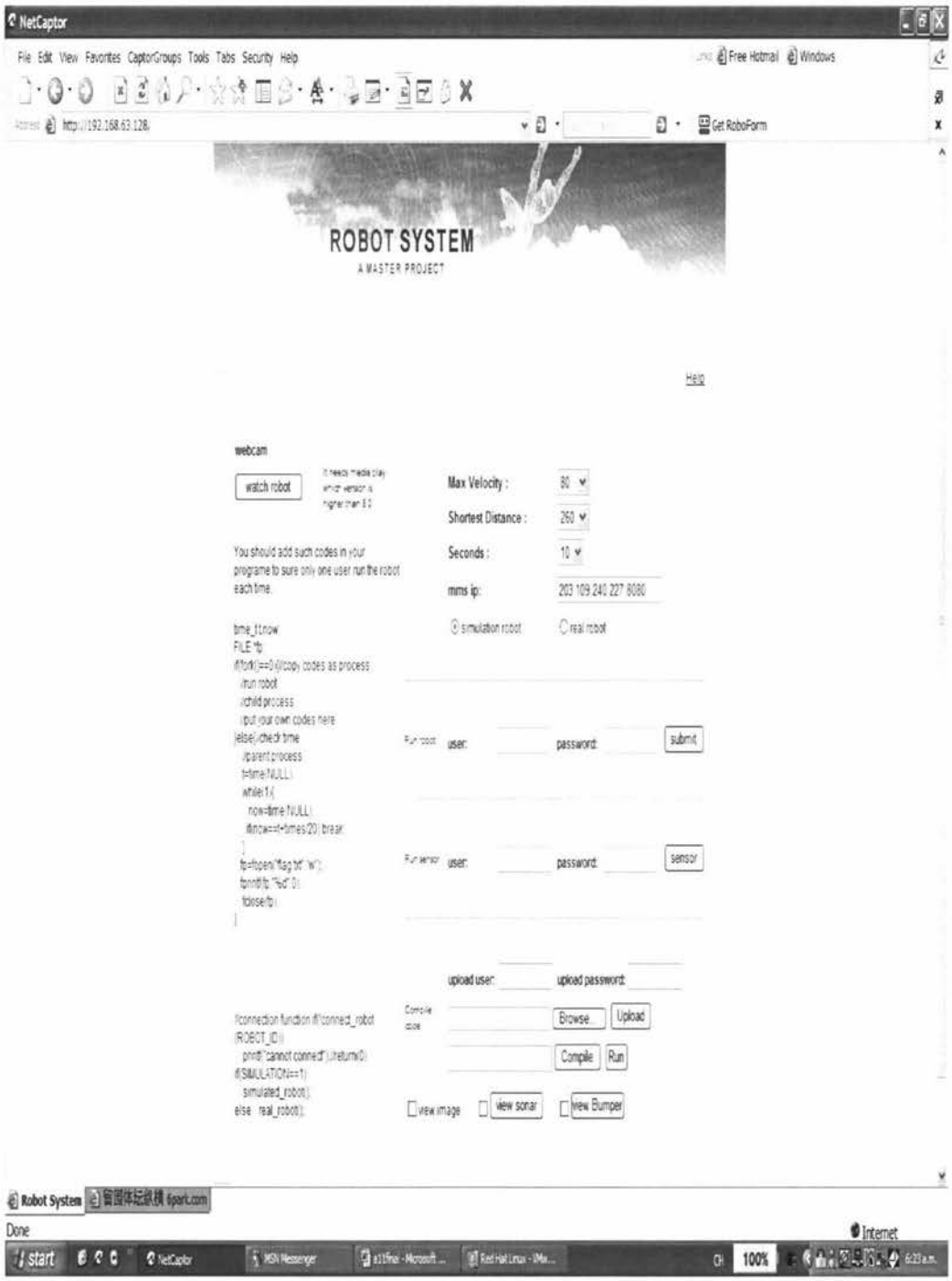


Figure 4- 8 User Interface of TeleScout System

At the left side of the interface the video system can be accessed via user name and password. Internet user can enter the user name and password to view the real time robot image over internet. There are sample codes available in the user

interface which must be copied to the C programs the internet user is programming. This part of code is to ensure only one user can operate the real robot at a time. These codes are already included in those two sample programs but internet user has to include these codes into their own programs by simply copying and pasting. These codes are to avoid two or more users sending commands to the real robot when it is in working.

4.5 Programming for user interface

4.5.1 Introduction

A user interface is required for a teleoperation system over internet. The remote user can communicate with the web server and the hardware devices by means of the interface. This includes sending commands and getting the feedback from the hardware devices. Since it is a teleoperation system, normally a visual monitor has to be attached to the user interface. Remote users can watch the hardware device online and perform the operation.

The most popular web server platforms with their integrated programming environments include Microsoft's internet Information Server with Active Server Pages, Netscape's Enterprise Server with Java, and Apache Software Foundation's Apache HTTP Server with Professional Home Page [35]. Apache HTTP server plus PHP web page development in Linux operation system is used in this project.

The robot interface, as shown in Figure 4-8, is programmed in Dreamweaver which is popular webpage design software from Macromedia [41] incorporating PHP code.

Dreamweaver has grown in popularity with professional web developers since its first release in late 1997 [42]. It has a powerful visual editor for creating and managing web pages for web developer without writing Hypertext Markup Language (HTML) code. The web developer can just simply drag and drop the web components like hyperlinks, hit counter or even web search to the web page without writing any HTML code. The HTML source codes are generated according to what the developer creates visually.

The first step is to user Dreamweaver to create the HTML home page for the HTTP server as shown in Figure 4-8. The basic elements of the home page comprise the text input boxes, buttons and select box. Simply dragging those text boxes and buttons to the HTML page, Dreamweaver will automatically generate HTML code.

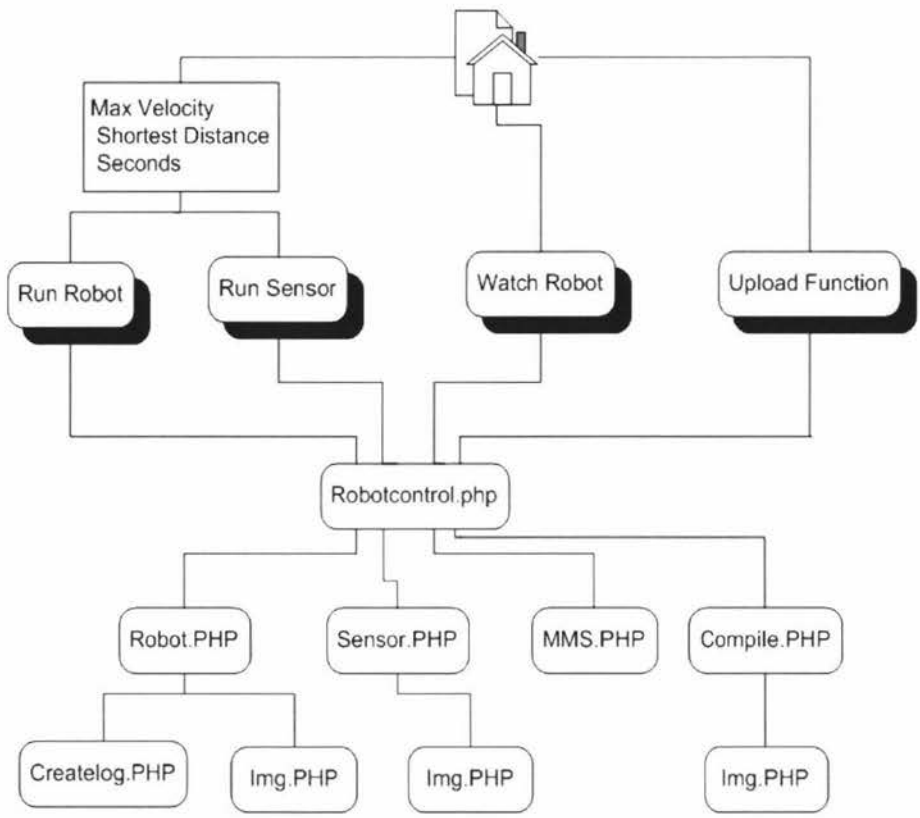


Figure 4- 9 The Site Map of TeleScout System

Figure 4-9 illustrates the site map of TeleScout system. There are 4 main functions linked to the homepage which are Run Robot, Run Sensor, Watch Robot and Upload Function. For Run Robot and Run Sensor, the programs have been already compiled. Remote users just supply the parameters and then run the robot by pressing Run Robot or Run Sensor button after the correct user name and password are used to login. Remote users can also use Upload Function to upload their own C program and run the real robot. The commands and the parameters are sent to the Robotcontrol.PHP, which check the user name and password first, and then send the parameters and commands received from the user interface to different function units.

Robot.PHP is the function for running a sample program for autonomous obstacle avoidance. It will recorder all the locations the robot visits and simulate the robot trajectory once the real robot will have completed its running. A total of five robot running logs can be stored in memory by executing Createlog.php. Remote user can choose either one to run after the simulation or the coordinates to be downloaded. The robot trajectory can be displayed by executing Img.PHP.

Sensor.PHP is a program to record the sonar sensor, bump sensor data of the Scout robot in the course of running. All the sensor data are downloadable.

Compile.PHP is a very practical function unit for the developed TeleScout system. Using it users can upload their C code and compile it at the remote HTTP server. When compiling is completed successfully, the program can be executed and the sensor data can be downloaded. If they want to download the sensor data, a block of sample code needs to be inserted into their program.

Img.PHP is a function for drawing the trajectory of the robot at remote user's web browse. It will be used by Robot.PHP, sensor.php and Upload.PHP.

4.5.2 Programming Techniques

Robotcontrol.PHP

Figure 4-10 presents the flow chart of Robotcontrol.PHP. Its function is to send the parameters to the sample programs – Run Robot and Run Sensor if user passes the user name and password check. So only one user can access it at a time as there is only one robot. Since this is a web based operation, there will definitely be a big chance more than one person trying to access the robot at the same time. Thus, a feature that prevents multiple users from accessing the robot has to be implemented. The feature in Robotcontrol.php is performed by checking whether the robot is free while someone sends control command to the robot. When the robot is free, the robot will execute that command, otherwise Robotcontrol.php will send a busy notice back to that user.

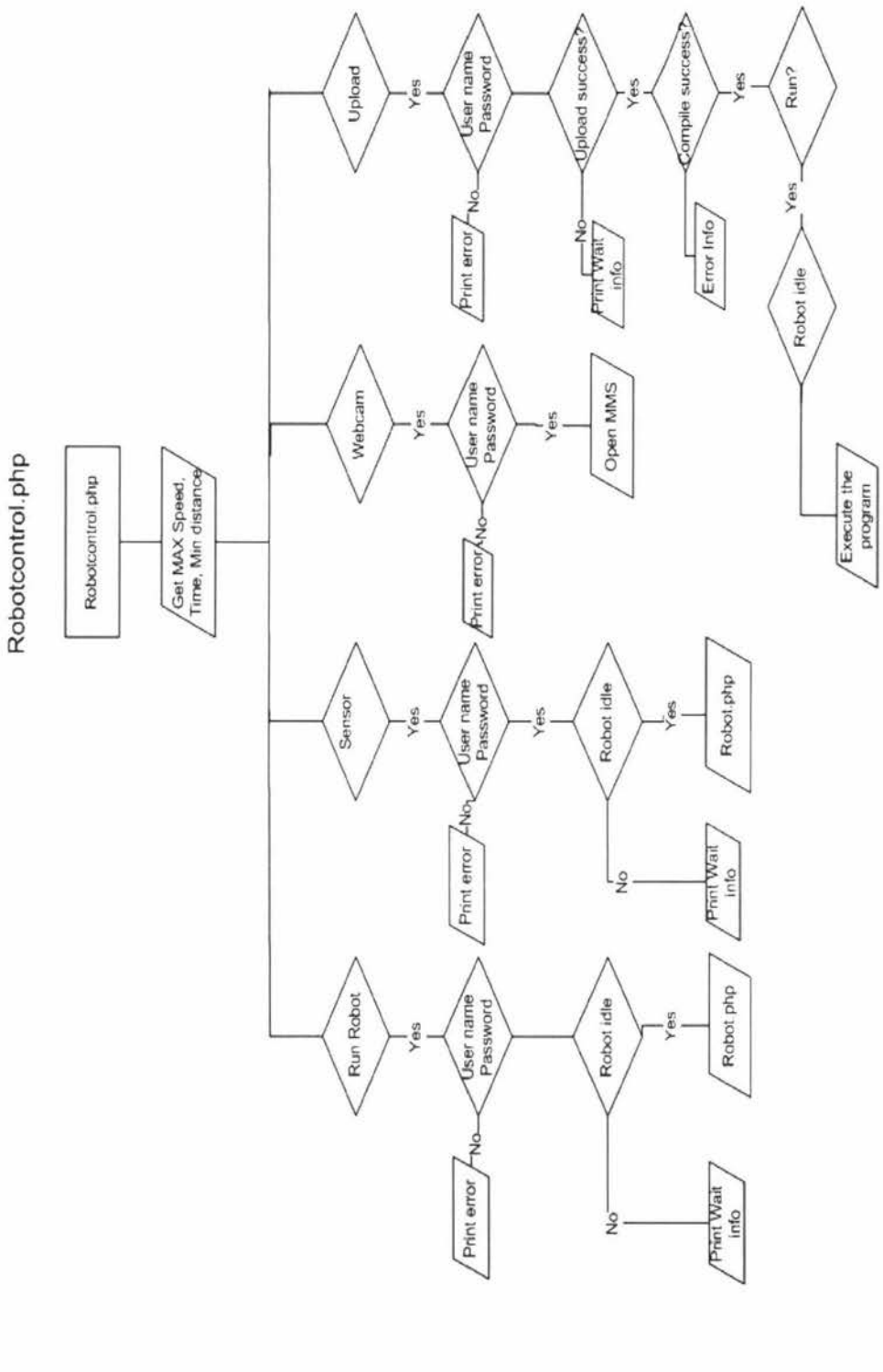


Figure 4- 10 Robotcontrol.PHP

The pseudo code for Robotcontrol.PHP is listed below. There is a special part of code being executed at the robot control program. It will set the flag as is the sign of the robot being busy or idle. Robotcontrol.PHP is the script to check the flag value whenever it receives a request for controlling the robot. A flag value can be either zero or one which means that the robot is idle or busy.

```
{
        Get max speed; simulate time and the min distance from HTML
form;
        Case
                Run robot;
                        User name and password;
                        Robot idle; yes?
                        Include robot.php
                Run sensor
                .....
                Upload
                        User name and password;
                        Upload successfully; yes?
                        Compile successfully; yes?
                        Robot idel; yes?
                        Execute the program
                Webcam
                .....
```

There is a special part of code being executed at the robot control program. It will set the flag which is the sign of the robot being busy or idle. Robotcontrol.PHP is the script to check the flag value whenever it receives a request for controlling the robot. A flag value can be either zero or one with zero meaning that the robot is idle and can be accessed and one meaning that the robot is busy and a notice will be sent back to the remote users.

The flag value will be set to zero initially in the robot control program before the

robot starts to run. The robot process will duplicate itself and hence there will be two same robot processes running and stopping at the same time and the server side. One is the main process and the other is the child process. Those two processes are exactly the same. The main process accumulates the robot running time when the robot just begins to run. When the robot running time >= the robot running time the user set, the value of the flag will be reset to one by the child process and simultaneously the robot stops running. Consequently, at any time the robot is running, the value of the flag is zero which means the robot is busy. When the robot is idle, the value of the flag is one which means the robot is ready for receive any commands. These parts of codes have been already included in those two sample programs – Run Robot and Run Sensor. But when users want to run their own programs, they need to upload them to the server. The program must contain the part of code listed at left side of Figure 4-8 to avoid more than one command being sent to the robot when the robot is running.

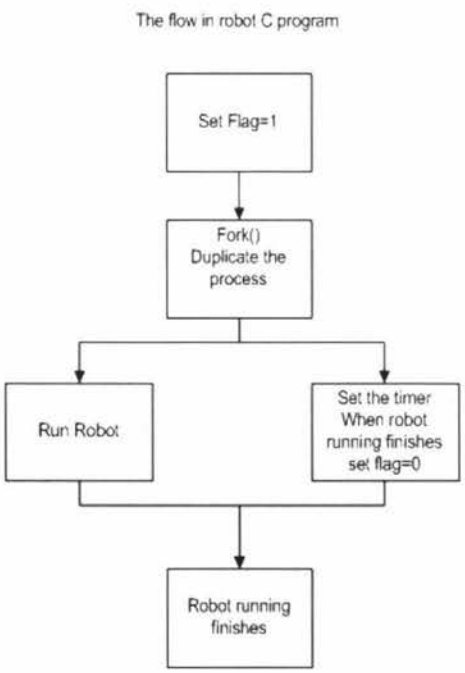


Figure 4- 11 The Fork Process

The flow chart for the flag process and its code is listed below,

```
{
        Code for robot control
        ……
        ……
        Set flag = 1;  // to inform other users that the robot is busy
        Duplicate the robot process by fork();
        Run robot;
        Set the flag = 0; // to inform the other users that the robot is free
}
```

Robotcontrol.php also runs a program designed to run before executing user's commands. It is to reset the Nserver robot mode to simulation. Since user can select real robot or simulation robot at the user interface. There is a chance that the robot hangs up when robot server mode is in simulation while user uploads and runs a program in real mode.

Compile.PHP

This is a very practical and useful function in the TeleScout system. High level remote user who is familiar with C programming can upload their C programs to compile and execute at the HTTP server. The operation interface is shown in Figure 4-12. User can select view image, view sensor and view bump. Once clicking on the button, the sample code for this function will be displayed as presented in Figure 4-13. User also needs to check the checkbox for showing a related button for displaying the sonar data.

Those codes are used to display different sensor data on the output HTML page. Once one of them is selected, the sample code will be displayed at the bottom and remote user can copy those codes into their C program in order to get the sensor

date from the webpage. For example, if a user needs to display the sonar data, the sonar checkbox is activated first and the view sonar is then pressed, and the sample code is copied into the application program.

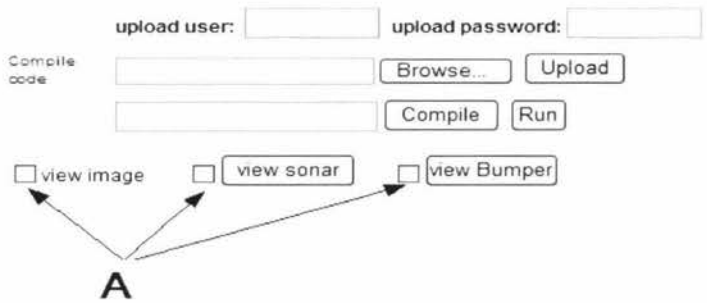


Figure 4- 12 The Upload Function

Figure 4- 13 View Bumper Sample Code

Remote user first writes the C program locally, then enters a proper user name and password, then browses the file and finally presses upload. The file will be uploaded to the remote server. Figure 4-14 presents some information displayed after the program has been uploaded. Users can know their uploaded file status from the debug information. On the left side of Figure 4-15 is the successfully uploaded information and the right side is the failed information.

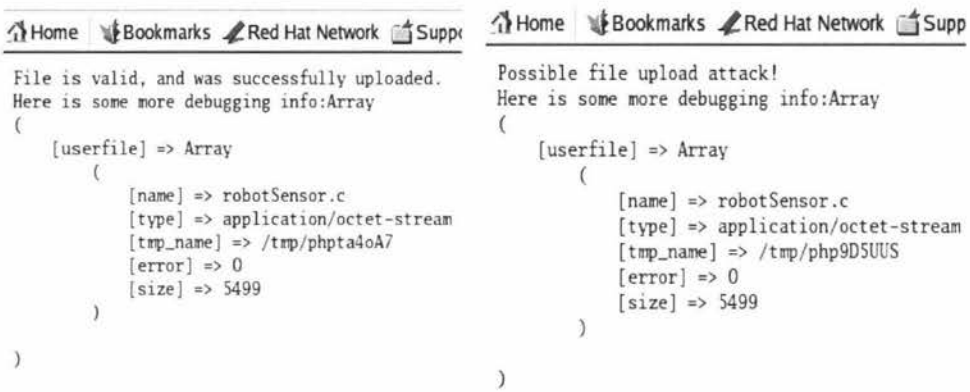


Figure 4- 14 Upload Information

The next step is to go to second line – compile code. The same name as the uploaded file is entered. For example, if the uploaded file name is robotsensor.c, the name entered in the compile line is robotsensor. By pressing Compile button, the program will be compiled at the server. If the compile fails the remote server will send local user the compile error message. There will be an error log hyperlink next to the compile failure information. The challenge in this session is how to give the end user a notice that the compile is unsuccessful. This is common for a C program, which usually needs to compile more than once before getting a correct executable program. User relies on the error message to check the program.

Since the uploaded file is saved in a folder on the server harddisk, if the C program is all correct, the compiled one will be created in that specific folder. However, there is another problem to be considered that if the first time the program was successfully compiled and later the user modifies it. the newly compiled file will not be created.

The solution for this problem is shown in Figure 4-15. The script will go to the destination folder to delete the output file with same name as the current file. When the php script receives the compile command, it will get the file name and go to check if there is such output file with that file name. If that file exists, delete

it then do the compile. After compiling, check if that file exists again to confirm if the file has been compiled successfully. The compiling message will be sent to the user.

There is an error log link at the HTML page when the compiling is unsuccessful. The error log file is saved in Apache error log folder. The Apache Server error log records any errors that occur in the course of processing requests [43]. The C program compiling error message is recorded in that file, too. A hyperlink is made to this error log file, and by downloading and checking the log file, user can know the problems in the program. The source program needs to be revised locally and is uploaded and compiled until the user sees the "compile successful" information.

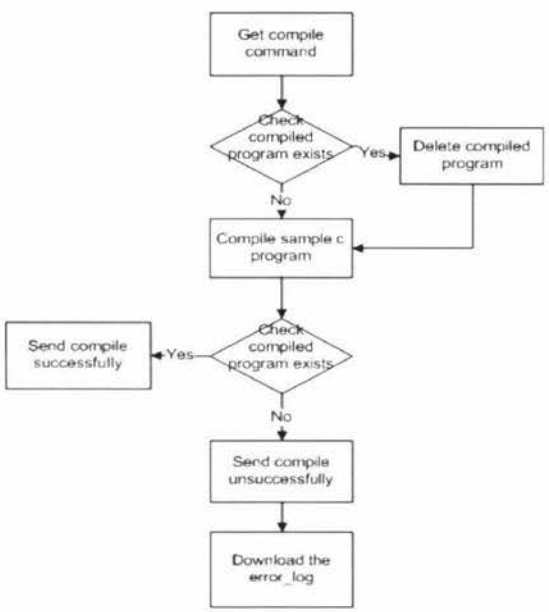


Figure 4- 15 Compile Message

The last step is to run the program. It can be executed locally. The file name needs to be entered with arguments given in the text box followed by pressing run button. Compile.PHP will generate a report HTML page to the end user, as shown in Figure 4-16. By the time now user can press the View Image to see the robot trajectory, and press View Sonar or View Bumper to see all sonar or bumper sensor data.

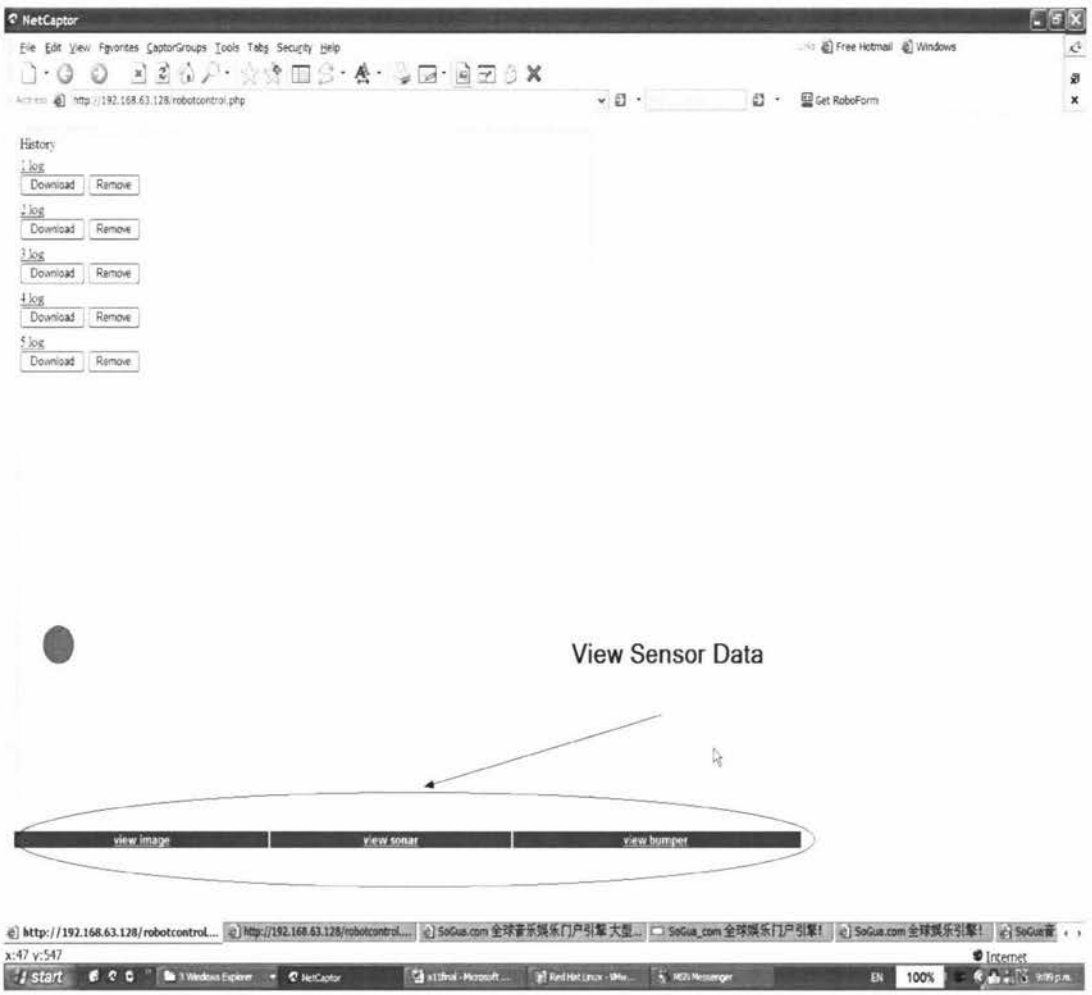


Figure 4- 16 Output HTML Page for Upload Function

Chapter 5 Experimental Case Studies

5.1 Experiment Setup

The client PC and the robot are located in a different computer labs. It has the following features, Operation system: Windows XP, Web browser: IE 6 and Windows Media Player

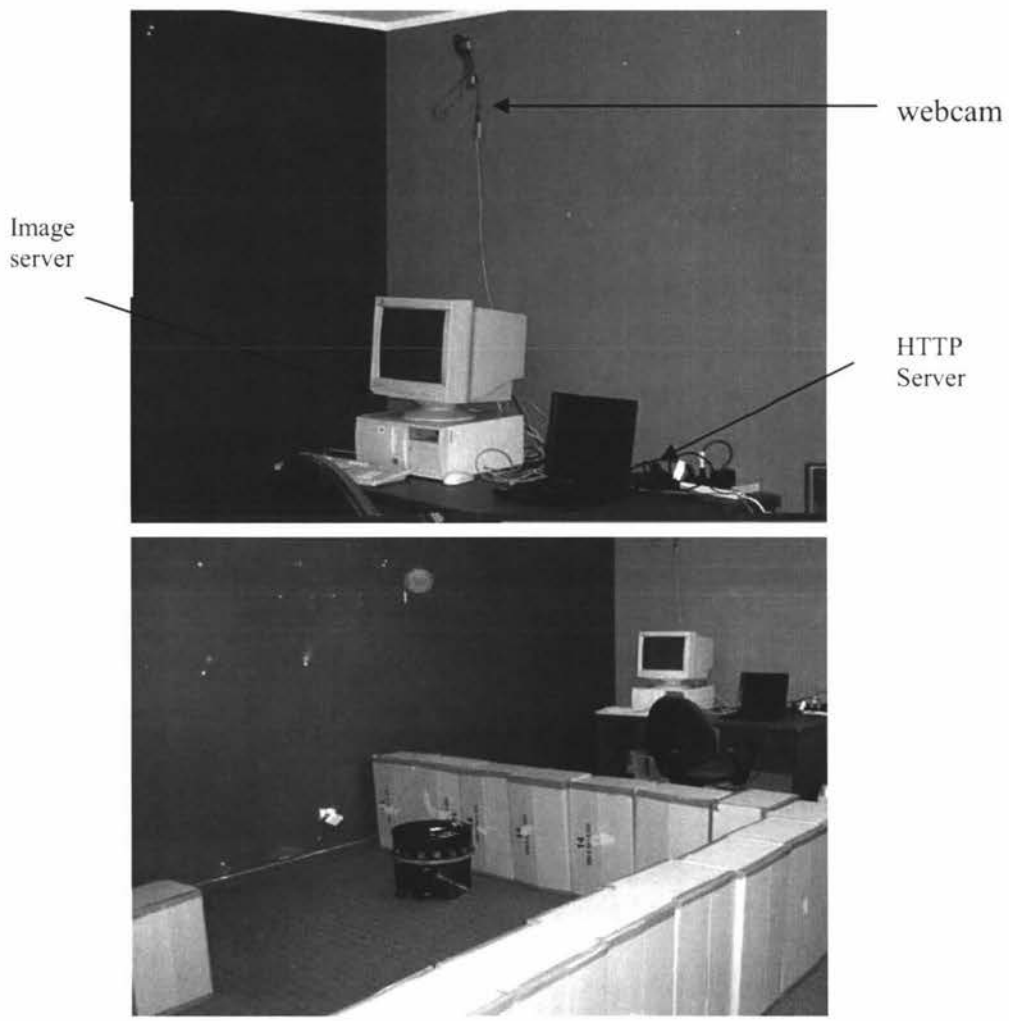


Figure 5- 1 The Lab Image of the Robot and Operating Environment

The mobile robot and its operating environment are shown in Figure 5-1. There are two computers: the laptop for the web server and robot controller and the computer for the image server. Carton boxes are used for the barrier fence.

5.2 The Sample Programs

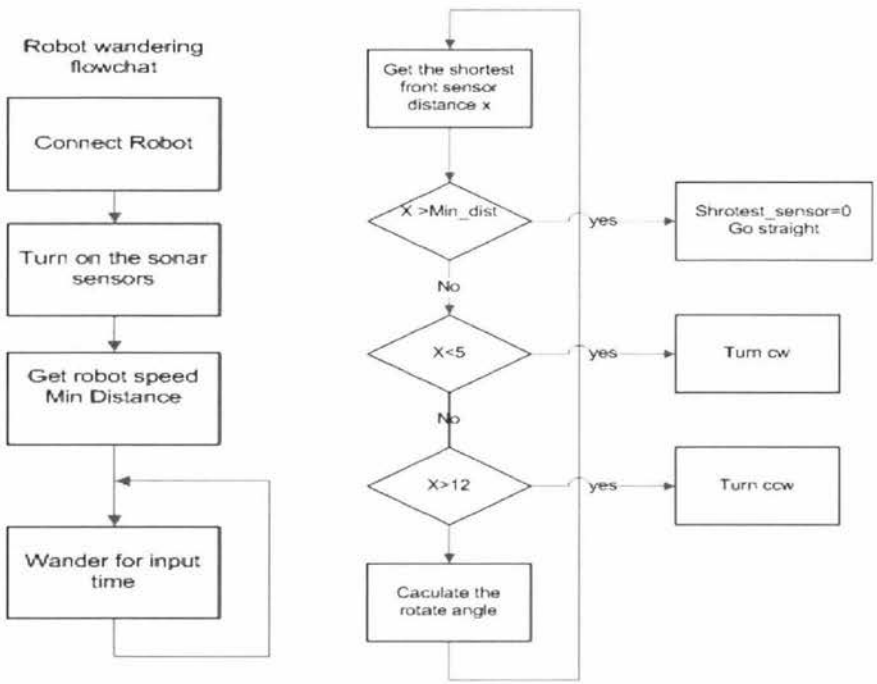


Figure 5- 2 Robot Wandering Program Flowchart

There are two sample programs that can be selected to run at the user interface. One is to run robot and the other is to run sensor. Figure 5-2 is the flow chart for the first sample program for autonomous obstacle avoidance. The robot will wander around, detect the obstacles and pass around them within a minimum distance. Users can specify the maximum speed the robot travels, the minimum distance between the obstacle and the robot and the time period the robot will run. The robot will turn on all 16 sonar sensors first and get the frontal 5 sensors data, and then compare these data with the minimum distance specified by the user. If all the sensor data are bigger than the minimum distance which means there is no obstacle in front of it, the robot will go straight. If any sensor data are smaller than the minimum distance, the robot will check the direction of the obstacle and turn counter-clockwise or clockwise to avoid any collision until all the 5 frontal sensor data are bigger than the minimum distance.

Users can get the robot coordinators by running the robot program and they can also get back sonar sensor data, bump sensor data by running the sensor program.

5.3 Experimental Case 1

In this case, a client computer with Redhat Linux 9 operation system is used to access and control the robot in simulation mode. Figure 5-3 shows the parameters provided for this simulation. Run Robot function will be selected and submitted to the server to perform the simulation in the Nserver.

Since this is only a simulation run, the real robot won't move. The simulation robot in Nserver will move according to what has been specified by the parameters supplied by the user in the user interface.

The operating field where the robot moves is shown at the left side of Figure 5-4, and the moving trace of the robot is given at the right side. The robot travels in a constant clearance to the wall and fence.

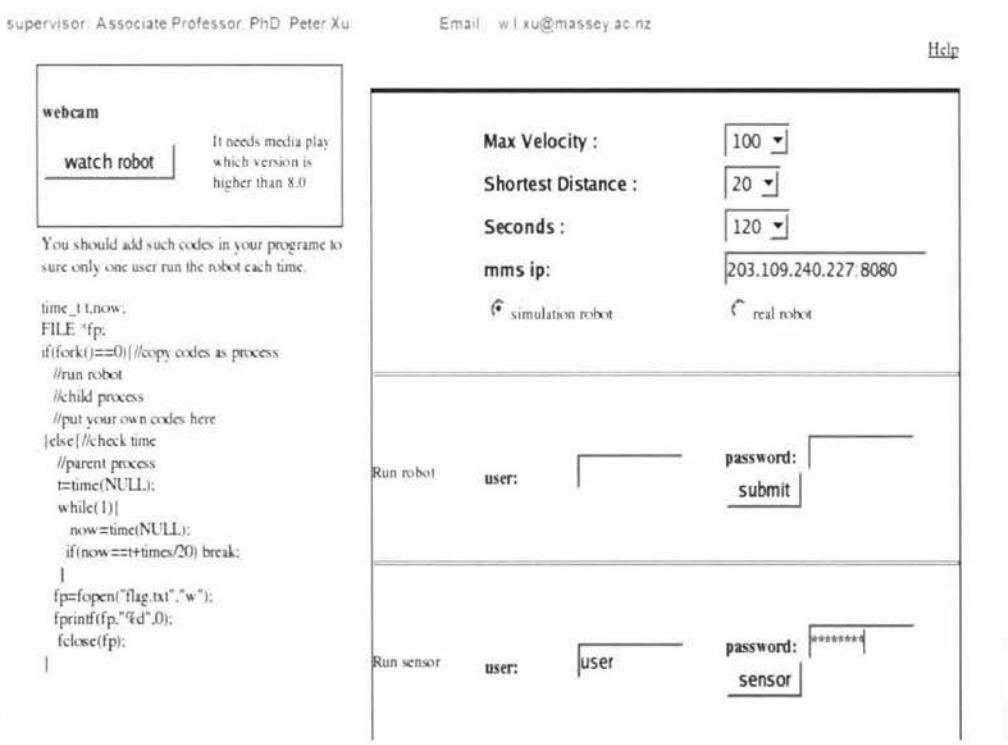


Figure 5- 3 Simulation Parameters

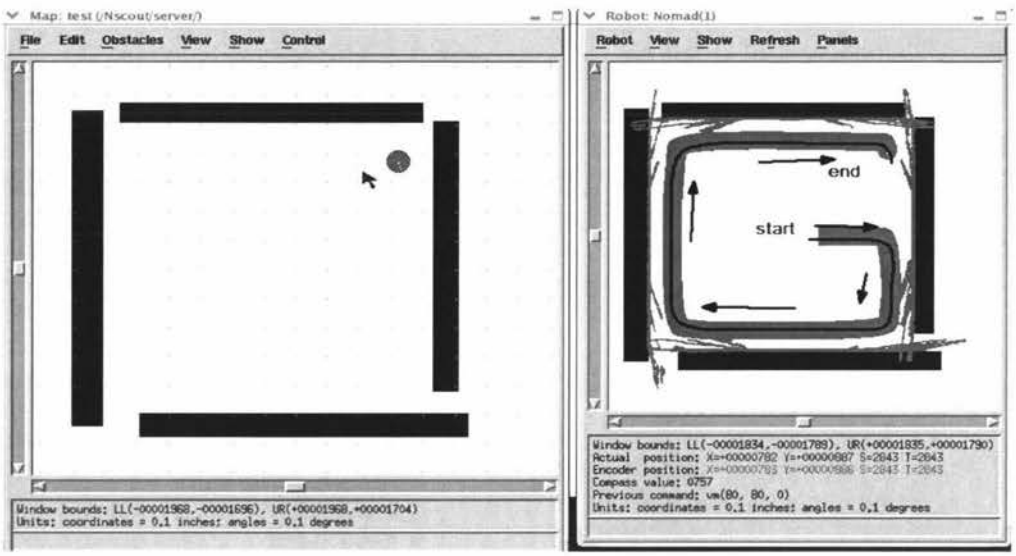
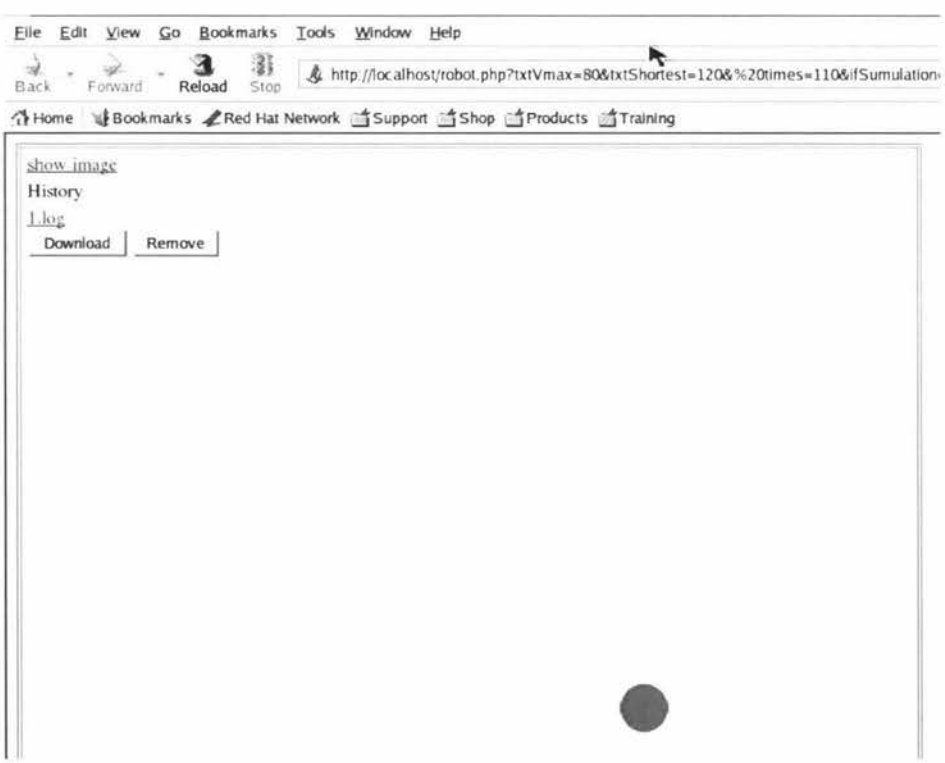


Figure 5- 4 The Robot Trace from Nserver

Figure 5-5 presents the result generated by the HTTP to the remote user.



The main result page includes show image and the log files. User can review the robot movement by clicking on 1.log. The trace of the robot movement is the same as the one shown in Figure 5-4. User can download the robot coordinates by

clicking the download button in a HTML page. Remote user can simple copy those coordinate data for further analysis.

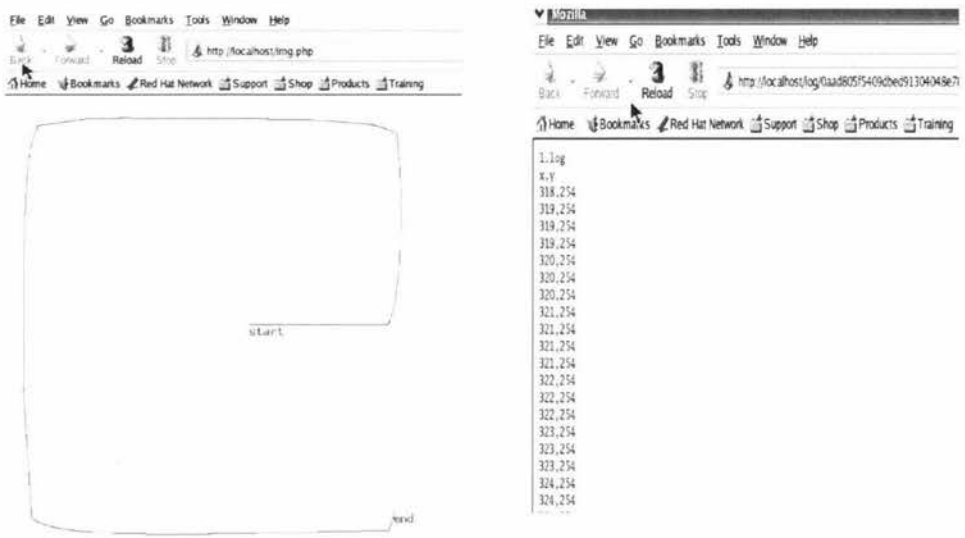


Figure 5- 5 Simulation Result Page

The second part of the case study is to use the same map to perform a Run Sensor simulation. The parameters specified are the same as those of Run Robot above.

Figure 5-6 shows the sonar data in Nserver on the remote user screen. All the small dots represent the measured distance of the obstacle which is smaller than the minimum distance set. In this simulation, the minimum distance between the sonar sensor and the obstacle is 120 inch.

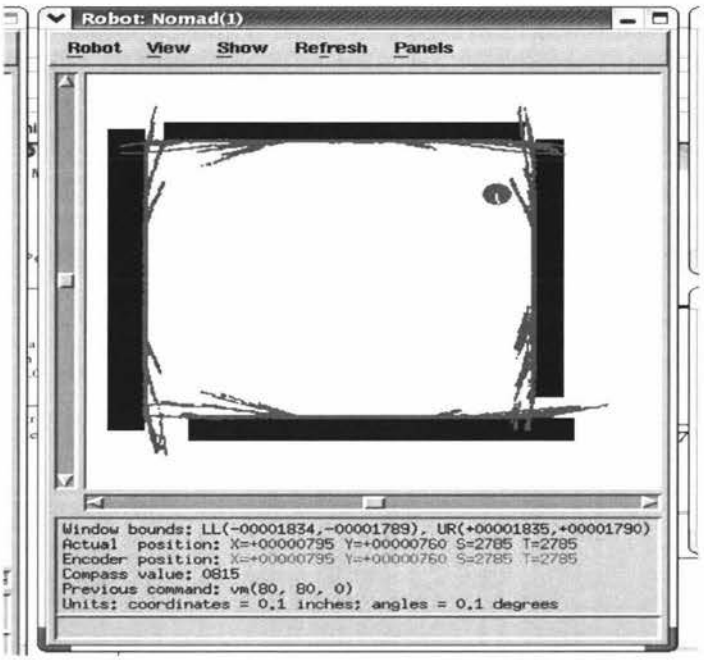
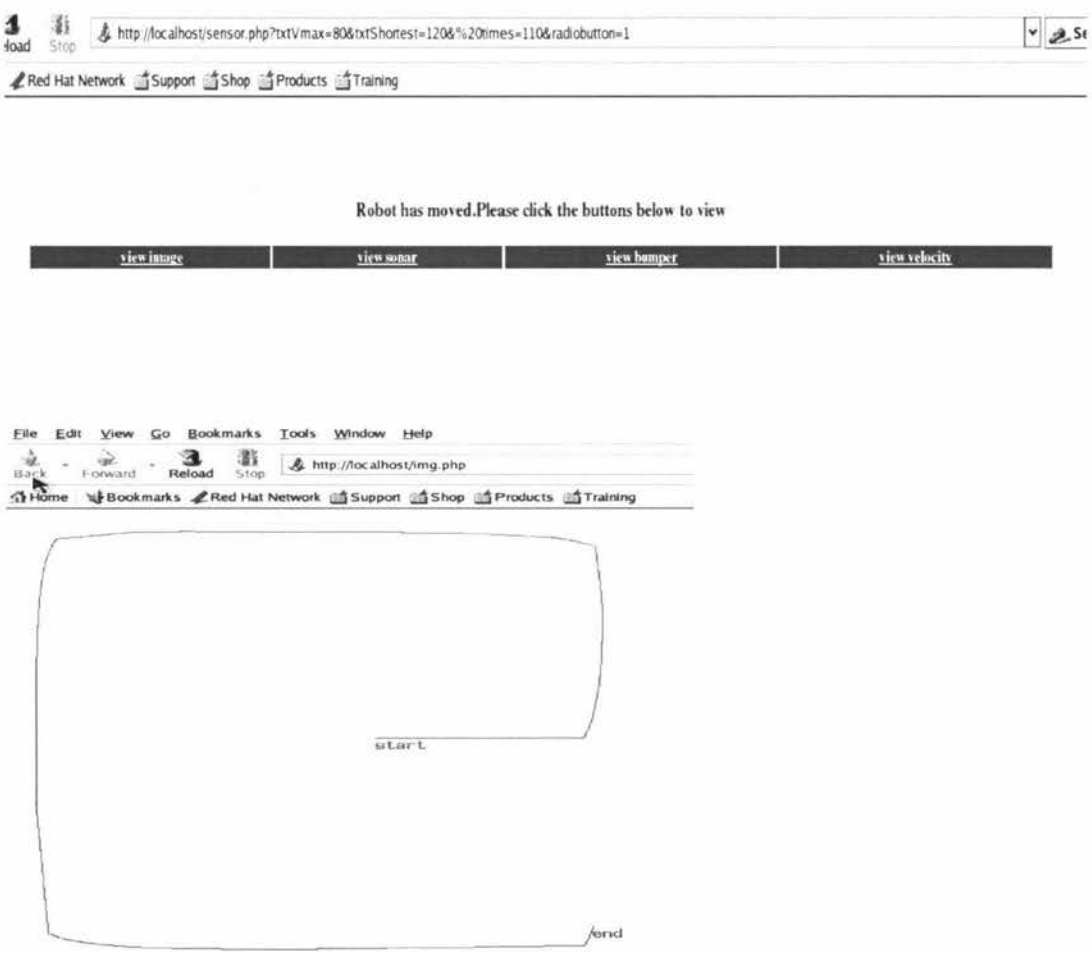


Figure 5- 6 Sonar Data on Nserver

Figure 5-7 is the HTML page generated by the HTTP server to the remote user.



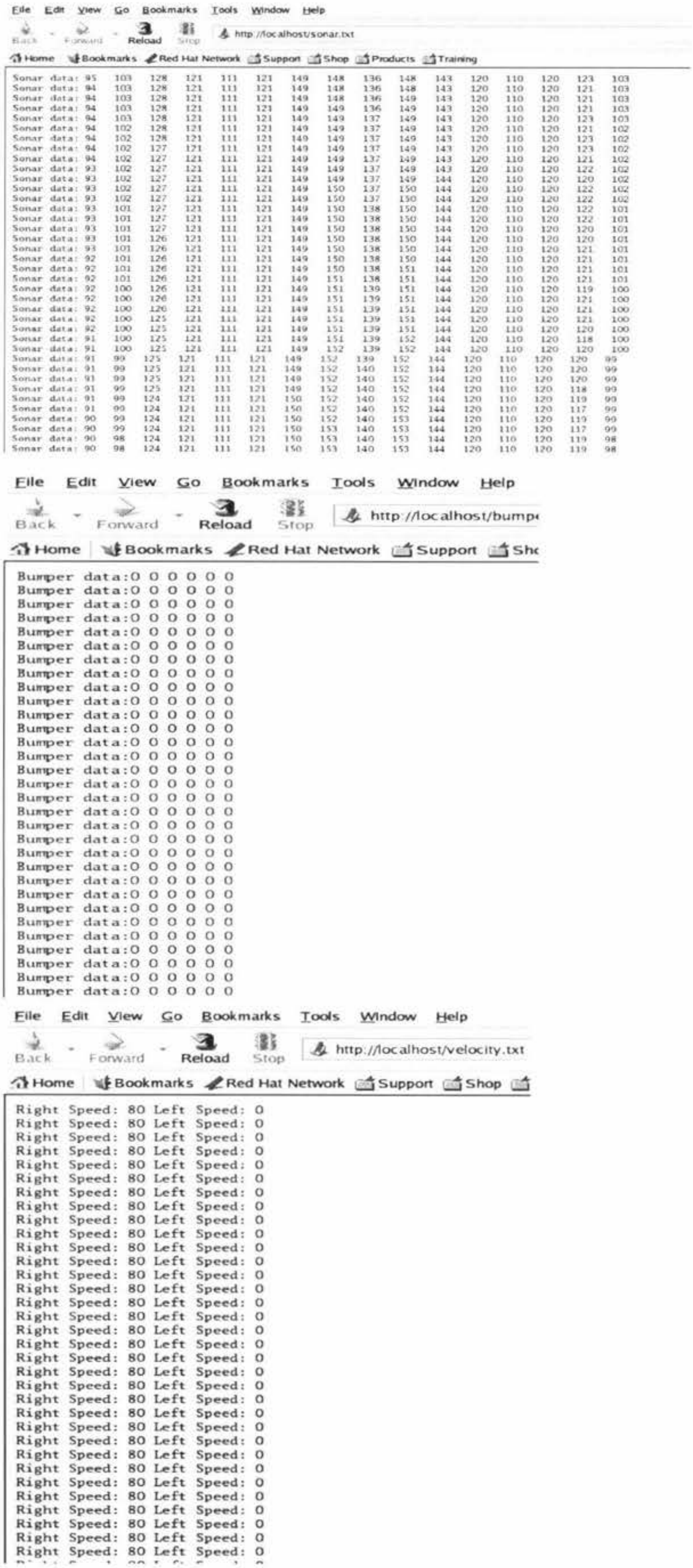


Figure 5- 7 Run Rensor Result Page

By clicking the hyperlink of View Image, View Sonar, View Bumper and View Velocity, the related data will be displayed in a newly opened html page. User can download those data for further analysis.

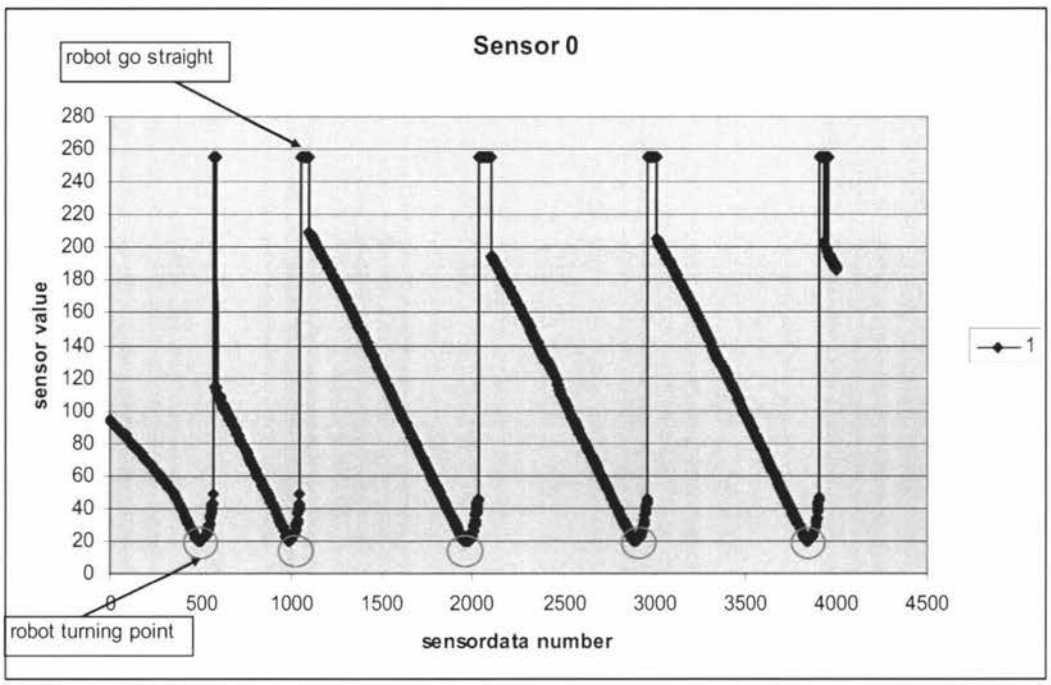


Figure 5- 8 Sensor 0 Data Chart

Figure 5-8 shows the data log of sensor 0 which is the main turning reference of Super Scout robot. The circles indicate where the robot turns. The minimum distance for turning is 20 inch. At the top of the Figure 5-8, there are five groups of sonar data equal 255. This means there were no obstacles in front of the robot and the robot goes straight forward.

5.4 Experimental Case 2

The task for this experiment is to upload a C program, make modification to get the feedback results, and compile and execute it.

Suppose that a remote user has the top and has a C program ready for controlling Nomad Super Scout Robot locally. An imaginary scenario is that the user can't compile and run this program directly on the real robot. However, the work can be done by using the Telescout system remotely.

The first step is to make a modification to the C program. The server will compile this file and execute it in the same way as is done locally. However, the user has to add some code to prevent other user from accessing the robot when the program is in running. The two sample programs – Run Robot and Run Sensor have already incorporated the code into their program. As a result, only one user can access the real robot and other will get a notice that the robot is busy. In addition, the user needs to get the sensor data from the remote server. A special block of codes have to be inserted into the program to retrieve the sensor data via a web browser, otherwise the user can only see the live video stream.

The program used in this experiment is for robot wandering. It has been compiled successfully locally. The codes added for session control are displayed in area A in Figure 5-9.

To make the sensor data to be displayed at HTML page, View Image, View Sensor and View Bumper should be selected. Pressing view sonar and view bumper button, there are two separate windows to show the sample code. User can add those sample codes to their C programs.

he next step is to type in the user name and password, to browse the program and upload it. The success of uploading window will be displayed in a HTML page. In this case, the file name is runrobot.c, and 'runrobot' must be typed in Compile box followed by pressing Compile button. The C program will be compiled on the remote server and a notice will be returned to the end user in a HTML page.

The final step is to run this program in command line mode. In this case, the command to run the robot is 'runrobot 100 25 120 0'. The parameters followed by runrobot are different according to individual user's programs.

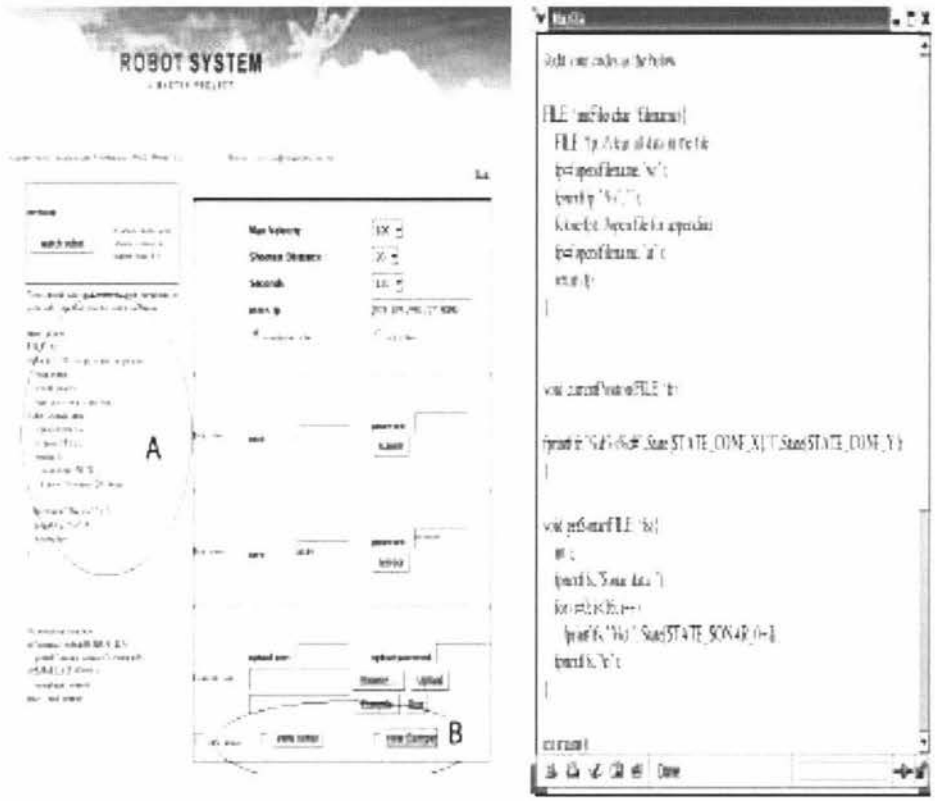


Figure 5- 9 Sample Code

The number 100 means max speed is 100 inc/sec; the minimum distance to the obstacle is 25 inches, 120 means the robot running time is 120 seconds and 0 means the robot is in real robot mode. The code of the program is listed in appendix.

The running result is reported in a HTML page shown in Figure 5-10. At the bottom of the report page, there are buttons for View Image, View Sonar and View Bumper. Clicking on those hyperlinks, user can get the corresponding data that are displayed in user's own web browser. Figure 5-11 shows those output pages.

From the output page, remote user can get the robot trace in image format, the robot sonar sensor and bumper sensor data in txt file format. For different levels of users they can choose to what kind of sensor date they want to get from the remote server.

Figure 5- 10 The Report Page

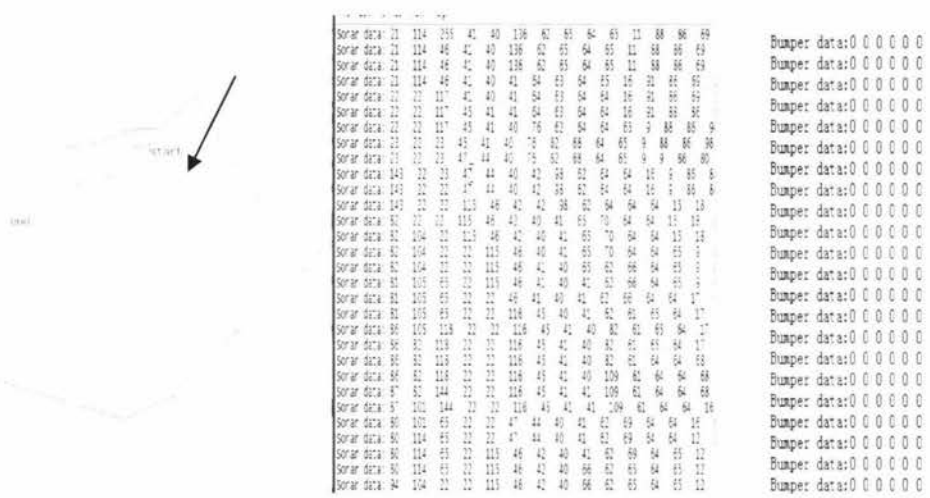


Figure 5- 11 The Output Page

Figure 5-12 shows the logged data of sonar sensor 0. Compared with those in Figure 5-8, the sonar readings are not regular. The largest sonar data in this case is at point A which is 150 inch. There are five turning points at points denoted by B. Those values are between 20 and 25 which are close to the specified minimum distance of 25 inches. Actually there are many turning points when the robot wandered around. When the robot intended to pass the obstacle, it stops and turns ccw or cw to find a way, which is indicated by the sonar data pattern denoted by point C and d.

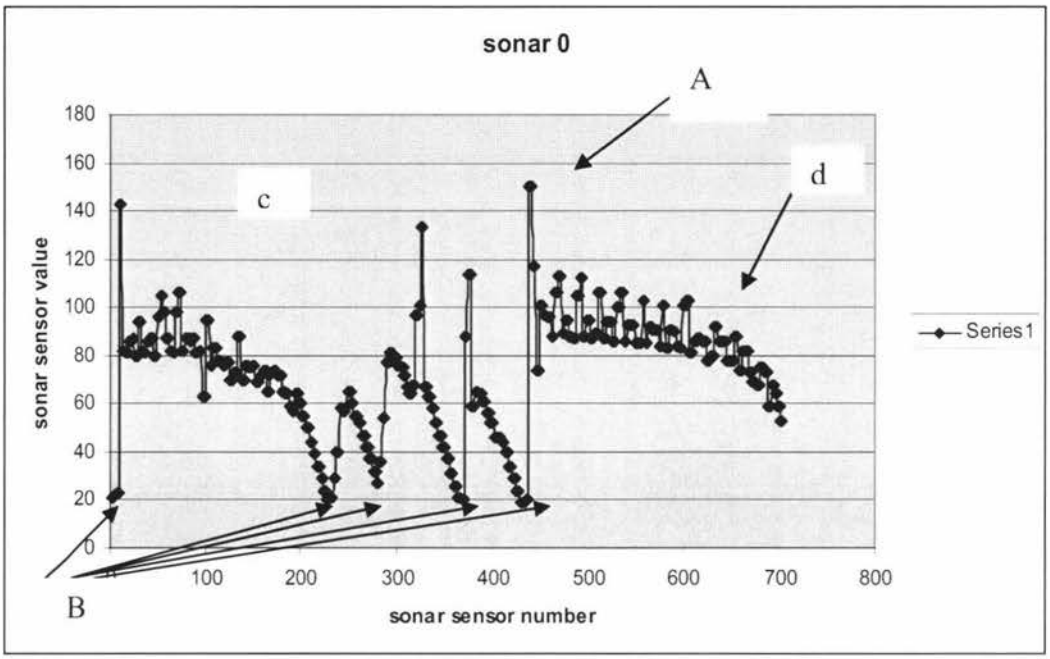


Figure 5- 12 Sonar Sensor 0 Data Chart for Compiler Function

User can also view the live video stream of the robot in windows media player. The video part of Telescout system is totally independent of the robot system. The video format is in VBR format, 282Kbps at 320x240 video sizes. The video quality is good enough to see the robot wandering around.

Chapter 6 Conclusions and Future Research

6.1 Conclusions

The experimental case studies have shown that that the Telescout system performed well. World wide users can now control the Super Scout robot located in the laboratory of Massey University. Its web address is HttP://it002069.

The Super Scout robot hardware was upgraded first. The on-board PC was removed and replaced by a laptop as the main high-level controller for the robot. Bluetooth connection between the robot low-level controller and the laptop was established.

Apache web server on Redhat Linux system was set up. PHP scripts were mainly used for the user interface programming, compared to the previous teleoperation system using CGI or JAVA. Several algorithms have been implemented to make TeleScout system more stable, such as the flag function that is used to ensure only user to control the robot, the unique upload function that allows user to upload, compile, debug and run users' own program at the remote server.

A common webcam was used as the video broadcasting device. The video system is a separate module which can be used in many other places.

Two case studies showed the procedure of how to use this Telescout system. The results from the case studies were analysed and presented properly.

6.2 Future Research

The further work should be concentrated upon improving the advanced controllers in order to improve the robot navigational performance. More and practical sample programs can be added to the TeleScout system to meet the needs of various levels of users.

The real time image capturing system should be improved. More cameras with more functions like auto zooming and focusing should be added to the TeleScout system.

The TeleScout system could be extended to include more robots that are controlled by different users at a time. Some other different types of robots may be considered to be incorporated into the TeleScout system.

6.3 Existing Problems

The robot collided once with the barrier fence at the end of the robot running session. When the server was sending a finish commands to the robot, it still got to move for another one second approximately. If the robot is too close to the barrier fence at the time the stoping commands arrives, there will be a big chance the robot crashes onto the barrier fence.

To solve this, reduction of the robot running speed should be reduced and the minimum distance specified between the robot and the obstacle should be increased.

References

[1] Sheridan, T. B. (1992): Telerobotics, automation and human supervisory control. Cambridge,MA: MIT Press.

[2] Steven Gentner, Nick Rothenberg, Carl Sutter, Jeff Wiegley http://www.usc.edu/dept/raiders/interface.html visited on 08/2005

[3] B. Dalton, "A distributed Framework for Internet Telerobotics", http://telerobot.mech.uwa.edu.au/Information/MITChapterBJAD.pdf, visited May 2006.

[4] K. Taylor and J.Trevelyan. Australia's telerobot on the web. In 26th International Symposium on Industrial Robotics, pages 39-44, Singapore, October 1995

[5] Mars Pathfinder Microrover Telecom Team , http://mars.jpl.nasa.gov/MPF/rovercom/rovintro.html visited on 06/2005

[6] Jet Propulsion Laboratory , http://mars.jpl.nasa.gov/MPF/mpf/rover.html, visited on 06/2005

[7] K. Goldberg, M. Nascha, "Beyond the Web: Excavating the Real World Via Mosaic", 1994, http://www.usc.edu/dept/raiders/paper/

[8] R.S. Mosher, "Force reflecting eletrohydraulic servo manipulator," Electro-Tech, Dec, 1960.

[9] K. Goldberg, M. Mascha, S. Gentner. N. Rothenberg, C. Sutter, and J. Wiegley,(1995) "Desktop teleoperation via the World Wide Web," Proceedings of IEEE International Conference on Robotics and Automation, Part Vol. 1, pp.654-9.

[10] M. R. Stein, "The PumaPaint Project", Journal of Autonomous Robots, Fall 2003

[11] C. Madden and M. Stein, "A Robotic Device for Three-Dimensional Manipulation over the Internet", Proceedings of the 28th Biennial Mechanisms and Robotics Conference (MECH) October 2004

[12] goldberg@ieor.berkeley.edu, http://queue.ieor.berkeley.edu/~goldberg/garden/Ars/ visited on 06/2005

[13] The global.com http://www.treasurecrumbs.com/pixel/goldberg/faq.html, visited on 06/2005

[14] viatlnet.com , http://www.vitalnet.net/h/support_glossary.html visited on 10/2005

[15] Marin, R.; Sanz, P.J.; Nebot, P.; Wirz, R.;Industrial Electronics, IEEE Transactions on Volume 52, Issue 6, Dec. 2005 Page(s):1506 - 1520 Digital Object Identifier 10.1109/TIE.2005.858733

[16] Sara McMains Carlo S´equin Charles Smith Paul Wright, 2000 "Internet-based Design and Manufacturing" http://www.ucop.edu/research/micro/99_00/99_106.pdf

[17] [27] Herbert Schildt, "Java 2 : a beginner's guide"

[18] Microsoft, http://www.microsoft.com/windows/windowsmedia/forpros/encoder/default.mspx Visited on 11/2005.

[19] KT Keung & WL Xu "Software Environment and Intelligent Navigation. of. Nomad Super Scout Mobile Robot" 2000, <http://www.massey.ac.nz/~wlxu/research/undergraduate/under4.htm >

[20] UNIVERSITE LIBRE DE BRUXELLES , http://iridia.ulb.ac.be/Projects/flar.html visited on 09/2006

[21] Nomadic Technology, "Nomadic scout user's manual, 12 July, 1999", http://nomadicic.sourceforge.net/production/scout/scout_user-1.3.pdf, visited April 2004.

[22] Pcguild.com "http://www.pcguide.com/ref/hdd/if/ide/modesUDMA-c.html" visited on 10/2005

[23] http://www.sundance.com/edge/files/pci.htm visited on 10/2005

[24] www.redhat.com visited on 09/2005

[25] http://nomadic.sourceforge.net/production/scout/upgrade.html visited on 10/2005

[26] http://www.bnoack.com/index.html?http&&&www.bnoack.com/data/RS232-port.html visited on 05/2005

[27] "RS232 CONNECTION THAT WORKS" < www.bb-elec.com/bb-elec/literature/ tech/faq_rs232_connections_work.pdf>

[28] http://www.aircable.net/serial.html visited on 10/2005

[29] McDermott-Wells, P.;Potentials, IEEEVolume 23, Issue 5, Dec 2004-Jan 2005 Page(s):33 - 35 Digital Object Identifier 10.1109/MP.2005.1368913

[30] www.ericsson.com visited on 10/2005

[31] Mercier, C.D.; Hembree, S.D.;Industry Applications Magazine, IEEE Volume 4, Issue 6, Nov.-Dec. 1998 Page(s):8 - 15 Digital Object Identifier 10.1109/2943.730753

[32] http://www.pcwebopedia.com/TERM/R/router.html visited on 06/2005

[33] Hirsch, P.M.;Computer Applications in Power, IEEE Volume 8, Issue 3, July 1995 Page(s):25 - 29 Digital Object Identifier 10.1109/67.392022

[34] Ben Laurie and Peter Laurie ,1999, "Apache : the definitive guide"

[35] Adaotptive digital technology, http://www.adaptivedigital.com/services/serv_definitions.htm, visited on 01/2006

[36] Antarctic weather station , http://www.antarcticanz.govt.nz/education/2568, visited on 01/2006

[37] Earthcam.com , http://www.earthcam.com/usa/newyork/timessquare/, visited on 01/2006

[38] Microsoft, http://www.microsoft.com/windows/windowsmedia/9series/encoder, visited on 10/2005

[39] http://www.adaptivedigital.com/services/serv_definitions.htm visited on 08/2005

[40] Microsoft, http://www.microsoft.com/windows/windowsmedia/howto/articles/BroadcastDelay.aspx#bufferingstreamingmediadata, visited on 02/2006

[41] www.macromedia.com, visited on 11/2005

[42] Copeland, D.R.; COrbo, R.C.; Falkenthal, S.A.; Fisher, J.L.; Sandle, M.N.; IT Professional Volume 2, Issue 2, March-April 2000 page(s):20-27 Digital Object Identifier 10.1109/6294.839363

[43] http://httpd.apache.org/docs/1.3/logs.html, visited o1/2006

APPENDICES

A1. Robotcontrol.php

```
<?
        //session_start();
        $user=$_REQUEST["user"];
        $password=$_REQUEST["password"];
        $webcamuser=$_REQUEST["webcamuser"];
        $webcampwd=$_REQUEST["webcampwd"];
        $uploaduser=$_REQUEST["uploaduser"];
        $uploadpassword=$_REQUEST["uploadpassword"];
        $webcam=$_REQUEST["webcam"];
        $run=$_REQUEST["Submit"];
        $action=$_REQUEST["action"];
        $prom=$_REQUEST["prom"];
        $sensor=$_REQUEST["sensor"];
        $suser=$_REQUEST["suser"];
        $spassword=$_REQUEST["spassword"];



        $vmax=$_REQUEST["txtVmax"];
        $shortest=$_REQUEST["txtShortest"];
        $times=$_REQUEST["times"];
        //$times*=32;
        $ifSumulation=$_REQUEST["radiobutton"];
        $log=$_REQUEST["log"];

        if($run){//check if run robot submit
                if($user=="user" && $password=="password"){
                $handle=fopen("flag.txt","r");
                $flag=fgets($handle);
                //print("flag".$flag);
                if($flag==0){
                        print("<html><body>\n");
                        print("<script language='javascript'>\n");
                        print("alert('The robot is running.Please wait');");
                        $l="robot.php?txtVmax=$vmax&txtShortest=$shortest&
times=$times";

                        print("location='$l';\n");
                        print("</script>\n");

                        print("</body></html>");

                        //include("robot.php");
                }else{
                   print("The robot is running, try late please.");
```

```
                }
                }else{
                   print("Wrong password or user!");
            }
           }

           if($sensor){
                   if($suser=="user"&&$spassword=="password"){
                           $handle=fopen("flag.txt","r");
                   $flag=fgets($handle);
                   //print("flag".$flag);
                   if($flag==0){

                                   print("<html><body>\n");
                                   print("<script language='javascript'>\n");
                                   print("alert('The robot is running.Please wait');");

$l="sensor.php?txtVmax=$vmax&txtShortest=$shortest&
times=$times&radiobutton=$ifSumulation";

                                   print("location='$l';\n");
                                   print("</script>\n");

                                   print("</body></html>");
                                   //include("sensor.php");
                   }else{
                      print("The robot is running, try late please.");
                   }
                   }else{
                   print("Wrong password or user!");
        }
           }

           if($webcam){//if submit to requir webcam
                   if($webcamuser=="user" && $webcampwd=="user"){
                           $ip=$_REQUEST["mmsip"];
                           print("<script language=javascript>");
                           print("location='mms.php?mmsip=$ip'");
                           print("window.close();");
                           print("</script>");
                   }else{
                      print("Wrong password or user!");
              }
           }
//echo "user:$uploaduser  pwd:$uploadpassword";

           if($action){
                   if($uploaduser=="user" && $uploadpassword=="password"){
                           switch($action){
                                   case "Upload":
                                           $uploaddir = 'uploads/';
```

```
                                        $uploadfile = $uploaddir .
basename($_FILES['userfile']['name']);
                                        echo '<pre>';
                                        if
(move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
                                           echo "File is valid, and was successfully
uploaded.\n";
                                        } else {
                                           echo "Possible file upload attack!\n";
                                        }
                                        echo 'Here is some more debugging info:';
                                        print_r($_FILES);
                                        print "</pre>";
                                        break;
                                case "Compile":
                                        if (!file_exists("uploads/$prom.c")) {
                                                echo "The file $filename does not
exists.Check if you uploaded";
                                        }
                                        if (file_exists("uploads/$prom")) {
                                                unlink("uploads/$prom");
                                        }

                                        $info=shell_exec("gcc -o uploads/$prom
uploads/$prom.c Nclient.o -lm");
                                        if (file_exists("uploads/$prom")) {
                                                print("Compile sucessful");
                                        }else{
                                                print("Compile error");
                                        }
                                        break;
                                case "Run":
                                        $temp=explode(" ",$prom);
                                        $prom1=trim($temp[0]);
                                        if (!file_exists("uploads/$prom1")) {
                                                echo "The file $filename does not
exists.Check if you uploaded or compiled";
                                        }else{
                                                $robot=$prom;
                                                include("compile.php");
                                        }
                        }
                }else{
                        print("Check your user name and password!");
                }
        }

?>
```

A2.Compile.php

```
<?php
session_start();


        if(!$robot) $robot="robot";//define the file name as robot if it does not
defined

        function changeScale($pos){//change scale
                $aPos=explode("#",$pos);
                $size=count($aPos);
                $size--;
                //$lx=$rx=$ty=$by=0;
                for($i=0;$i<$size;$i++){
                        $pos1=explode("|",$aPos[$i]);
                        $pos1[1]=str_replace("#", ",$pos1[1]);
                        if(!$lx)$lx=$pos1[0];
                        if(!$rx)$rx=$pos1[0];
                        if(!$ty)$ty=$pos1[1];
                        if(!$by)$by=$pos1[1];

                        if($pos1[0]>$rx)
                                $rx=$pos1[0];
                        if($pos1[0]<$lx)
                                $lx=$pos1[0];
                        if($pos1[1]<$by)
                                $by=$pos1[1];
                        if($pos1[1]>$ty)
                                $ty=$pos1[1];
                }
                $w=$rx-$lx;
           $h=$ty-$by;

           if($w==0){
                   $w=0.0001;
           }
           if($h==0){
                   $h=0.0001;
           }

           if($w>$h){
              $factor=500/$w;
           }else{
              $factor=500/$h;
           }
                //if($w!=0)$ws=500/$w;
                //if($h!=0)$s=$w/$h;
                //if($s!=0)$fh=500/$s;
```

```
        for($i=0;$i<$size;$i++){
                $pos1=explode("|",$aPos[$i]);
                $pos1[1]=str_replace("#", ",$pos1[1]);
        $nx=($pos1[0]-$lx+70)*$factor;
                $ny=($pos1[1]-$by+70)*$factor;
                $newpos.=ceil($nx)."|".ceil($ny)."#";
        }
        //echo "lx:$lx<br>";
        //echo "rx:$rx<br>";
        //echo "ty:$ty<br>";
        //echo "by:$by<br>";
        //echo "ws:$ws<br>";
        return $newpos;


}

function setFile($txt,$filesNum){
        $i=1;
        while(true){

if($_SESSION["$i.log"]==""||$_SESSION["$i.log"]==null){
                        break;
                }
                $i++;
        }

        if($i<=$filesNum){
                $_SESSION["$i.log"]=$txt;
        }else{
                for($i=1;$i<=$filesNum;$i++){
                        $_SESSION["$i.log"]="";
                }
                $_SESSION["1.log"]=$txt;
        }

}




$vmax=$_REQUEST["txtVmax"];
$shortest=$_REQUEST["txtShortest"];
$times=$_REQUEST["times"];
$times*=32;
$log=$_REQUEST["log"];

print("<html><body>");
if(!$log){
```

```
                $filename=trim("./exec/$robot $vmax $shortest $times");
                if($action=="Run") $filename=trim("./uploads/$robot $vmax
$shortest $times");

                $pos=exec($filename);//run robot c prom

                $pos2=changeScale($pos);
                $f=fopen("positon.txt","r");
                $pos=fread($f,filesize("positon.txt"));
                fclose($f);
                $pos2=changeScale($pos);
                $_SESSION["trance"]=$pos2;
                setFile($pos2,5);


        }else{
                $pos2=$_SESSION["$log.log"];
        }

        $aPos=explode("#",$pos2);
        $size=count($aPos);
        $size--;        //transfter x|y#x|y to Ax=x1, Ay=x2  here is counting how
many coordinates




        print("<input type='hidden' name='hidIndex' value='0' id='hidIndex'>\n");

        for($i=0;$i<$size;$i++){
                $pos1=explode("|",$aPos[$i]);
                $pos1[1]=str_replace("#", ",$pos1[1]);

                print("<input type='hidden' name='hidX$i' value='".$pos1[0]."'
id='hidX$i'>\n");
                print("<input type='hidden' name='hidY$i' value='".$pos1[1]."'
id='hidY$i'>\n");
        }
        print("<div id='LayerBall' style='position:absolute; width:43px;
height:43px; z-index:1; left: 300px; top: 200px'><img src='ball.gif' width='41'
height='41'></div>");
        print("<script language=javascript>
                        function move(){
                                var x,y,index,i;


                                index=document.getElementById('hidIndex');
//save index of position

                                i=parseInt(index.value);
```

```
                                px='hidX'+i;
                                py='hidY'+i;
                                x=document.getElementById(px);
                                y=document.getElementById(py);

                                posX=parseInt(x.value);
                                posY=parseInt(y.value);
                                //if(posX==''){
                                //
                                //}

        document.getElementById('LayerBall').style.left=posX;

        document.getElementById('LayerBall').style.top=posY;
                                index.value=i+1;
                                window.status='x:'+posX+' y:'+posY;
                        }

                        function time1(){
                                //var x=0;
                                //var num=new Array();
                                //for(i=0;i<".$size.";i++){
                                        //num[i]=i+2;
                                //}

                                window.setInterval('move()',45);
                        }
                </script>
                ");
?>
<table width="750" height="750" border=1>
<tr valign=top><td>
<table>
<!--<tr><td><a href="img.php" target="_blank">show image</a></td></tr>-->
<tr><td>History</td></tr>
<?
for($i=1;$i<=5;$i++){
        if($_SESSION["$i.log"]!=""||$_SESSION["$i.log"]!=null){
                print("<form action='createlog.php' method=post
target='_blank'><tr>
                <td><a href='robot.php?log=$i'>$i.log</a>
                <br><input type=hidden name=log value=$i>
                <input type=submit name=createlog value='Download'>
                <input type=submit name=removelog value='Remove'>
                </td>
                </tr></form>");
        }
}
?>

</table>
```

```
</td></tr>


</table>
<?php
        print("<script language=javascript>time1();</script>");


//----------------------------------------
        $viewimage = $_REQUEST["checkImage"];
        $viewbumper = $_REQUEST["checkBumper"];
        $viewvelocity = $_REQUEST["checkVelocity"];
        $viewsonsar = $_REQUEST["checkSonsar"];
?>
<TABLE  WIDTH="75%" BORDER="0"><TR align=center
BGCOLOR="#6600FF">
<?php if($viewimage){ ?>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="img.php"
target="_blank"><font face=arial size=2 color=white><b>view
image</b></font></a></TD>
<?php
}
        if($viewsonsar){
?>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="sonar.txt"
target="_blank"><font face=arial size=2 color=white><b>view
sonar</b></font></a></TD>
<?php
}
        if($viewbumper){
?>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="bumper.txt"
target="_blank"><font face=arial size=2 color=white><b>view
bumper</b></font></a></TD>
<?php
}
        if($viewvelocity){
?>
        <!--<TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="velocity.txt"
target="_blank"><font face=arial size=2 color=white><b>view
velocity</b></font></a></TD> --></TR></TABLE>

<?php
}
        print("</body></html>");
```

```
?>
```

A3.Createlog.php

```
<?
session_start();
$log=$_REQUEST["log"];
$pos=$_SESSION["$log.log"];
$removelog=$_REQUEST["removelog"];
$createlog=$_REQUEST["createlog"];

$sid=session_id();
$filename="log/$sid$log.txt";
if($createlog){
  $pos="$log.log\rx,y\r".$pos;
  $pos=str_replace("#","\r",$pos);
  $pos=str_replace("|",",",$pos);

  $f=fopen($filename,'w');
  fwrite($f,$pos);
  print("Close the window when you download and save it!");
  print("<script language=javascript>
          location='log/$sid$log.txt';
       </script>");
  echo $pos;

}

if($removelog){
  if(file_exists($filename)){
    if(unlink($filename)){
       print("removed the log file");
    }else{
       print("removed unsuccessful");
    }
  }else{
    print("The log file does not exist!");
  }
}

?>
```

A4.Img.php

```
<?
session_start();

	//drawline function----------------------------------------------------
	function drawLine($points_x,$points_y){

		$img = @imagecreate(750, 750)
			or die("Cannot Initialize new GD image stream");
		$background_color = imagecolorallocate($img, 255, 255, 255);
		$linecolor = imagecolorallocate($img, 233, 14, 91);
		$size=count($points_x);
		for($i=0;$i<$size-1;$i++){
			imageline($img, $points_x[$i], $points_y[$i],
$points_x[$i+1], $points_y[$i+1], $linecolor);
			//echo "x:$points_x[$i]  y:$points_y[$i]<br>";
		}
		imagestring($img, 5, $points_x[0], $points_y[0],  "start",
$linecolor);
		imagestring($img, 5, $points_x[$size-1], $points_y[$size-1],
"end", $linecolor);
		return $img;
	}



//drawline

	header("Content-type: image/png");
	//$img=$_REQUEST["img"];
	$pos=$_SESSION["trance"];
    //str 2 array----------------------------
	$aPos=explode("#",$pos);
	$size=count($aPos);
	$size--;
	for($i=0;$i<$size;$i++){
		$pos1=explode("|",$aPos[$i]);
		$x[$i]=$pos1[0];
		$y[$i]=str_replace("#", ",$pos1[1]);
	}
    //---------------------------------------
	$img1=drawLine($x,$y);

	imagepng($img1);
	imagedestroy($img1);


?>
```

A5.mms.php

```
<html>
<body BGCOLOR=#CBCDDE>
<p align=center>
<object id="wmp" classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95"
width="336" height="400">
<param name="filename" value="mms://<?echo $_REQUEST['mmsip']?>">
<param name="PlayCount" value="1">
<param name="AutoStart" value="1">
<param name="ClickToPlay" value="1">
<param name="DisplaySize" value="0">
<param name="EnableFullScreen Controls" value="1">
<param name="ShowAudio Controls" value="1">
<param name="ShowStatusBar" value="1">
<param name="ShowDisplay" value="1">
<param name="AutoSize" value="1">
<param name="EnableContext Menu" value="1">
</object>
</p>
</body>
</html>
```

A6.Sensor.php

```
<?php
session_start();
        function changeScale($pos){//change scale
                $aPos=explode("#",$pos);
                $size=count($aPos);
                $size--;
                //$lx=$rx=$ty=$by=0;
                for($i=0;$i<$size;$i++){
                        $pos1=explode("|",$aPos[$i]);
                        $pos1[1]=str_replace("#", ",$pos1[1]);
                        if(!$lx)$lx=$pos1[0];
                        if(!$rx)$rx=$pos1[0];
                        if(!$ty)$ty=$pos1[1];
                        if(!$by)$by=$pos1[1];

                        if($pos1[0]>$rx)
                                $rx=$pos1[0];
                        if($pos1[0]<$lx)
                                $lx=$pos1[0];
                        if($pos1[1]<$by)
```

```
                        $by=$pos1[1];
                    if($pos1[1]>$ty)
                        $ty=$pos1[1];
                }
                $w=$rx-$lx;
                $h=$ty-$by;
                if($w>$h){
                    $factor=500/$w;
                }else{
                    $factor=500/$h;
                }

                for($i=0;$i<$size;$i++){
                    $pos1=explode("|",$aPos[$i]);
                    $pos1[1]=str_replace("#", ",$pos1[1]);
                $nx=($pos1[0]-$lx+70)*$factor;
                    $ny=($pos1[1]-$by+70)*$factor;
                    $newpos.=ceil($nx)."|".ceil($ny)."#";
                }

                return $newpos;


        }

        $vmax=$_REQUEST["txtVmax"];
        $shortest=$_REQUEST["txtShortest"];
        $times=$_REQUEST["times"];
        $times*=32;
        $shortest=$_REQUEST["txtShortest"];
        $ifSumulation=$_REQUEST["radiobutton"];

        $filename=trim("./exec/sensor $vmax $shortest $times $ifSumulation");

        exec($filename);//run robot c prom
        $f=fopen("positon.txt","r");
        $pos=fread($f,filesize("positon.txt"));
        fclose($f);
        $pos2=changeScale($pos);
        $_SESSION["trance"]=$pos2;
?>
<HTML>
<HEAD>
<script language=javascript>
function changeColor(td,color){
    td.bgColor=color;
}
</script>
<TITLE>sensor robot</TITLE>
</HEAD>
```

```
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<P> </P><P> </P><P align=center><font face=arial
size=3><b>Robot has moved.Please click the buttons below to
view</b></font></P>
        <TABLE align=center WIDTH="75%" BORDER="0"><TR align=center
BGCOLOR="#6600FF">
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="img.php"
target="_blank"><font face=arial size=2 color=white><b>view
image</b></font></a></TD>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="sonar.txt"
target="_blank"><font face=arial size=2 color=white><b>view
sonar</b></font></a></TD>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="bumper.txt"
target="_blank"><font face=arial size=2 color=white><b>view
bumper</b></font></a></TD>
        <TD onmouseover="changeColor(this,'#CC3300')"
onmouseout="changeColor(this,'#6600FF')"><a href="velocity.txt"
target="_blank"><font face=arial size=2 color=white><b>view
velocity</b></font></a></TD> </TR></TABLE>
</BODY>
</HTML>
```

A7.Robot.php

```
<?php
session_start();


        if(!$robot) $robot="robot";//define the file name as robot if it does not
defined

        function changeScale($pos){//change scale
                $aPos=explode("#",$pos);
                $size=count($aPos);
                $size--;
                //$lx=$rx=$ty=$by=0;
                for($i=0;$i<$size;$i++){
                        $pos1=explode("|",$aPos[$i]);
                        $pos1[1]=str_replace("#", ",$pos1[1]);
                        if(!$lx)$lx=$pos1[0];
                        if(!$rx)$rx=$pos1[0];
                        if(!$ty)$ty=$pos1[1];
                        if(!$by)$by=$pos1[1];

                        if($pos1[0]>$rx)
                                $rx=$pos1[0];
```

```
                if($pos1[0]<$lx)
                        $lx=$pos1[0];
                if($pos1[1]<$by)
                        $by=$pos1[1];
                if($pos1[1]>$ty)
                        $ty=$pos1[1];
        }
        $w=$rx-$lx;
    $h=$ty-$by;
    if($w>$h){
      $factor=500/$w;
    }else{
      $factor=500/$h;
    }
        //if($w!=0)$ws=500/$w;
        //if($h!=0)$s=$w/$h;
        //if($s!=0)$fh=500/$s;
        for($i=0;$i<$size;$i++){
                $pos1=explode("|",$aPos[$i]);
                $pos1[1]=str_replace("#", ",$pos1[1]);
        $nx=($pos1[0]-$lx+70)*$factor;
                $ny=($pos1[1]-$by+70)*$factor;
                $newpos.=ceil($nx)."|".ceil($ny)."#";
        }
        //echo "lx:$lx<br>";
        //echo "rx:$rx<br>";
        //echo "ty:$ty<br>";
        //echo "by:$by<br>";
        //echo "ws:$ws<br>";
        return $newpos;


}

function setFile($txt,$filesNum){
        $i=1;
        while(true){

if($_SESSION["$i.log"]==""||$_SESSION["$i.log"]==null){
                        break;
                }
                $i++;
        }

        if($i<=$filesNum){
                $_SESSION["$i.log"]=$txt;
        }else{
                for($i=1;$i<=$filesNum;$i++){
                        $_SESSION["$i.log"]="";
                }
                $_SESSION["1.log"]=$txt;
```

```
		}

	}



	$vmax=$_REQUEST["txtVmax"];
	$shortest=$_REQUEST["txtShortest"];
	$times=$_REQUEST["times"];
	$times*=32;
	$log=$_REQUEST["log"];

	print("<html><body>");
	if(!$log){

		$filename=trim("./exec/$robot $vmax $shortest $times");
		if($action=="Run") $filename=trim("./uploads/$robot $vmax
$shortest $times");

		$pos=exec($filename);//run robot c prom

		$pos2=changeScale($pos);

		$_SESSION["trance"]=$pos2;
		setFile($pos2,5);
	}else{
		$pos2=$_SESSION["$log.log"];
	}

	$aPos=explode("#",$pos2);
	$size=count($aPos);
	$size--;	//transfter x|y#x|y to Ax=x1, Ay=x2  here is counting how
many coordinates



	print("<input type='hidden' name='hidIndex' value='0' id='hidIndex'>\n");

	for($i=0;$i<$size;$i++){
		$pos1=explode("|",$aPos[$i]);
		$pos1[1]=str_replace("#", ",$pos1[1]);

		print("<input type='hidden' name='hidX$i' value='".$pos1[0]."'
id='hidX$i'>\n");
		print("<input type='hidden' name='hidY$i' value='".$pos1[1]."'
id='hidY$i'>\n");
	}
```

```
	print("<div id='LayerBall' style='position:absolute; width:43px;
height:43px; z-index:1; left: 300px; top: 200px'><img src='ball.gif' width='41'
height='41'></div>");
	print("<script language=javascript>
			function move(){
				var x,y,index,i;


				index=document.getElementById('hidIndex');
//save index of position

				i=parseInt(index.value);
				px='hidX'+i;
				py='hidY'+i;
				x=document.getElementById(px);
				y=document.getElementById(py);

				posX=parseInt(x.value);
				posY=parseInt(y.value);
				//if(posX==''){
				//
				//}

	document.getElementById('LayerBall').style.left=posX;

	document.getElementById('LayerBall').style.top=posY;
				index.value=i+1;
				window.status='x:'+posX+' y:'+posY;
			}

			function time1(){
				//var x=0;
				//var num=new Array();
				//for(i=0;i<".$size.";i++){
					//num[i]=i+2;
				//}

				window.setInterval('move()',45);
			}
		</script>
		");
?>
<table width="750" height="750" border=1>
<tr valign=top><td>
<table>
<tr><td><a href="img.php" target="_blank">show image</a></td></tr>
<tr><td>History</td></tr>
<?
for($i=1;$i<=5;$i++){
	if($_SESSION["$i.log"]!=""||$_SESSION["$i.log"]!=null){
```

```
                print("<form action='createlog.php' method=post
target='_blank'><tr>
                <td><a href='robot.php?log=$i'>$i.log</a>
                <br><input type=hidden name=log value=$i>
                <input type=submit name=createlog value='Download'>
                <input type=submit name=removelog value='Remove'>
                </td>
                </tr></form>");
        }
}
?>

</table>
</td></tr>


</table>
<?
        print("<script language=javascript>time1();</script>");
        print("</body></html>");



?>
```

A8.Robotsensor.c

```
/*
 *
 * example program demonstrating simple motion control and sensing on
 * Scout, ScoutII, and SuperScout mobile robots.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

#include "Nclient.h"

#ifndef ALL_SENSORS
#define ALL_SENSORS 0
#endif

#define ROBOT_ID  1
```

```
#define ROTATION_CONSTANT    0.118597  /* inches/degree (known to 100
ppm) */

#define RIGHT(trans, steer)  (trans +
(int)((float)steer*ROTATION_CONSTANT))
#define LEFT(trans, steer)   (trans - (int)((float)steer*ROTATION_CONSTANT))

#define scout_vm(trans, steer)   vm(RIGHT(trans, steer), LEFT(trans, steer), 0)
#define scout_pr(trans, steer)   pr(RIGHT(trans, steer), LEFT(trans, steer), 0)


FILE *initFile(char *filename){
        FILE *fp;
        //clear all data in the file
        fp=fopen(filename,"wt");
        fprintf(fp,"%s","");
        fclose(fp);
        //open file for appending
        fp=fopen(filename,"at");

        return fp;
}

void currentPositon(int v,int steer,FILE *fr){
        fprintf(fr,"%d%s%d#",State[STATE_CONF_X],"|",State[STATE_CONF
_Y]);
}

void getSenor(FILE *fs){
        int i;
        fprintf(fs,"Sonar data: ");
        for (i=0; i<16; i++)
                fprintf(fs,"%ld   ", State[STATE_SONAR_0+i]);
        fprintf(fs,"\n");
}

void getBumper(FILE *fb){
        int i;
        fprintf(fb,"Bumper data:");
        for ( i = 0 ; i < 6 ; i ++ )
        if (State[STATE_BUMPER] & ( 1L << i ) )
                fprintf(fb,"1 ");
        else
                fprintf(fb,"0 ");
        fprintf(fb,"\n");
}

void getVelocity(FILE *fv){
        fprintf(fv,"Right Speed: %ld Left Speed: %ld\n",
State[STATE_VEL_RIGHT],State[STATE_VEL_LEFT]);
}
```

```
static void turn_front_sonar_on(void)
{
 /* remainder are irrelevant */
 int sn_order[16] = {0, 2, 15, 1, 14, 3, 13, 4, 12, 5, 11, 6, 10, 7, 9, 8};

 /* turn on 5 front sonars and set them to fire in certain order */
 conf_sn(15, sn_order);

 conf_tm(5);
}


/* monitor the active sonar as specified in parameter "order"
 * returns the sonar which detects the closest object and is less
 * than min_dist */
static short shortest_front_sonar(long *state, short min_dist,short shortest)
{
 short i, /*shortest,*/ shortest_sonar;

 // shortest = 255;
 shortest_sonar = 0;

 i = 17;
 do
 {
  if (state[i] < shortest)
  {
   shortest = (short)state[i];
   shortest_sonar = i - 16;
  }
  //printf("%d: %d ", i - 17, (int)state[i]);
  i++;
  if (i == 20)
  {
   i = 31;
  }
 } while (i < 33);

 if (shortest > min_dist)
 {
  shortest_sonar = 0;
 }

 //printf("\n");
 return(shortest_sonar);
}
```

```
static void wander(short track_dist,short vmax,short shortest,FILE *fs,FILE
*fr,FILE *fb,FILE *fv)
{

 short del_angle;
 short s_v = 0;
 //short vmax = 100;
 short shortest_front;
 static short cwturncounter;
 static short ccwturncounter;
 static short lastdir;

 shortest_front = shortest_front_sonar(State, track_dist,shortest);

 if (shortest_front == 0)
 {
  scout_vm(vmax, 0);
  ccwturncounter = cwturncounter = lastdir = 0;
  currentPositon(vmax,0,fr);
  getSenor(fs);
  getBumper(fb);
  getVelocity(fv);
  return;
 }

 if (shortest_front < 5)
 {
  /* check for getting stuck */
  if ((cwturncounter > 5) && (ccwturncounter > 5))
  {
   shortest_front = (shortest_front + 3);
  }
  else
  {
   shortest_front = -(shortest_front + 3);
  }
  if (lastdir == 0)
  {
   cwturncounter += 1;
  }
  lastdir = 1;
 }
 if (shortest_front > 12)
 {
  shortest_front = shortest_front - 12;
  if (lastdir == 1)
  {
   ccwturncounter += 1;
  }
  lastdir = 0;
 }
```

```
 del_angle = shortest_front*225;
 s_v = del_angle/4;
 scout_vm(0, s_v);
 currentPositon(0,s_v,fr);
 getBumper(fb);
 getSenor(fs);
 getVelocity(fv);
}


int main(int argc, char *argv[])//argv[1] is vmax,argv[2] is shortest
{
//*************************************************************
*

        FILE *fp,*fs,*fr,*fb,*fv;//fs: sonar file;fr:robot postion file;fb:bumper
file;fv:velocity file
        time_t t,now;
        int i,times,SIMULATION;
        short vmax,shortest;
        vmax=(short)atoi(argv[1]);
        shortest=(short)atoi(argv[2]);
        times=atoi(argv[3]);
     SIMULATION=atoi(argv[4]);

        fr=initFile("positon.txt");
        fs=initFile("sonar.txt");
        fb=initFile("bumper.txt");
        fv=initFile("velocity.txt");



//*************************************************************
*


 /* connect to the robot (in this example, to a ScoutII robot using the
  * Ndirect.o library) */
//  connect_robot(1, MODEL_SCOUT2, "/dev/ttyS0", 38400);



//SERV_TCP_PORT=7019;
//strcpy(SERVER_MACHINE_NAME, "localhost");

if(!connect_robot(ROBOT_ID))
        printf("cannot connect");//return(0);

if(SIMULATION==1)
 simulated_robot();
```

```
else
 real_robot();


 /* turn on the front sonars */
 turn_front_sonar_on();

 gs();
 fp=fopen("flag.txt","w");
 fprintf(fp,"%d",1);
 fclose(fp);
 /* the loop */
 if(fork()==0){//copy codes as process
   //run robot
   //child process
   i=0;
   while (i<times){
    wander(20,vmax,shortest,fs,fr,fb,fv);
    i++;
   }
 }else{//check time
   //parent process
   //printf("start/n");

   t=time(NULL);
   while(1){
     now=time(NULL);
     if(now==t+times/20) break;
   }
   fp=fopen("flag.txt","w");
   fprintf(fp,"%d",0);
   fclose(fp);
   //printf("end/n");
 }

 fclose(fr);
 fclose(fv);
 fclose(fb);
 fclose(fs);

 return 0;
}
```

A9.Robot.c [21]

```
/
 *
 * example program demonstrating simple motion control and sensing on
 * SuperScout mobile robots.
```

```
 *
 */

#include <stdio.h>
#include <stdlib.h>

#include "Nclient.h"

#define ROTATION_CONSTANT    0.118597  /* inches/degree (known to 100
ppm) */

#define RIGHT(trans, steer)  (trans +
(int)((float)steer*ROTATION_CONSTANT))
#define LEFT(trans, steer)   (trans - (int)((float)steer*ROTATION_CONSTANT))

#define scout_vm(trans, steer)   vm(RIGHT(trans, steer), LEFT(trans, steer), 0)
#define scout_pr(trans, steer)   pr(RIGHT(trans, steer), LEFT(trans, steer), 0)



static void turn_front_sonar_on(void)
{
 /* remainder are irrelevant */
 int sn_order[16] = {0, 2, 15, 1, 14, 3, 13, 4, 12, 5, 11, 6, 10, 7, 9, 8};

 /* turn on 5 front sonars and set them to fire in certain order */
 conf_sn(15, sn_order);

 conf_tm(5);
}


/* monitor the active sonar as specified in parameter "order"
 * returns the sonar which detects the closest object and is less
 * than min_dist */
static short shortest_front_sonar(long *state, short min_dist)
{
 short i, shortest, shortest_sonar;

 shortest = 255;
 shortest_sonar = 0;

 i = 17;
 do
 {
  if (state[i] < shortest)
  {
   shortest = (short)state[i];
   shortest_sonar = i - 16;
  }
  printf("%d: %d ", i - 17, (int)state[i]);
```

```
    i++;
    if (i == 20)
    {
     i = 31;
    }
  } while (i < 33);

  if (shortest > min_dist)
  {
   shortest_sonar = 0;
  }

  printf("\n");
  return(shortest_sonar);
}



static void wander(short track_dist)
{
  short del_angle;
  short s_v = 0;
  short vmax = 100;
  short shortest_front;
  static short cwturncounter;
  static short ccwturncounter;
  static short lastdir;

  shortest_front = shortest_front_sonar(State, track_dist);

  if (shortest_front == 0)
  {
   scout_vm(vmax, 0);
   ccwturncounter = cwturncounter = lastdir = 0;
   return;
  }

  if (shortest_front < 5)
  {
   /* check for getting stuck */
   if ((cwturncounter > 5) && (ccwturncounter > 5))
   {
    shortest_front = (shortest_front + 3);
   }
   else
   {
    shortest_front = -(shortest_front + 3);
   }
   if (lastdir == 0)
   {
    cwturncounter += 1;
```

```
    }
    lastdir = 1;
  }
  if (shortest_front > 12)
  {
    shortest_front = shortest_front - 12;
    if (lastdir == 1)
    {
      ccwturncounter += 1;
    }
    lastdir = 0;
  }
  del_angle = shortest_front*225;
  s_v = del_angle/4;
  scout_vm(0, s_v);
}



int main(void)
{
  /* connect to the robot (in this example, to a ScoutII robot using the
   * Ndirect.o library) */
  connect_robot(1, MODEL_SCOUT2, "/dev/ttyS0", 38400);

  /* turn on the front sonars */
  turn_front_sonar_on();

  gs();

  /* the loop */
  while (1)
  {
    wander(20);
  }

  return 0;
}
```