

U.S. NAVAL ACADEMY
COMPUTER SCIENCE DEPARTMENT
TECHNICAL REPORT



Mobile Vehicle Teleoperated Over Wireless IP

Akin, Micah Crabbe, Frederick L.

USNA-CS-TR-2007-02

June 13, 2007

Mobile Vehicle Teleoperated Over Wireless IP

Micah Akin and Frederick L. Crabbe

U.S. Naval Academy

Computer Science Department

572M Holloway Rd, Stop 9F

Annapolis, MD 21402

crabbe@usna.edu

Abstract

This document describes an effort to create a low cost teleoperated vehicle that is controlled via 802.11g Wifi and the Internet. It is unusual in that it can be controlled at a great distance but yet is still inexpensive. The appendices provide instruction on how to construct and operate the vehicle, along with the necessary source code.

1 Introduction and Background

Modern remote control (RC) vehicles have many uses including defusing bombs, space and deep sea exploration, and entertainment such as children's toys. While useful they generally have one of two drawbacks— either the operator must be within line-of-sight of the vehicle in order to directly control it via some sort of radio control device, or as is the case with military vehicles, the operator must be hooked into a sophisticated network system, often including a dedicated satellites and a fixed base-station. These differences also reflect a cost spectrum, from expensive military grade projects to hobbyist projects which cost relatively little. The military projects for the most part are large, and complex whereas the hobbyist RC projects are in general small, cheap, and straightforward. A good example of a Military RC project would be the Predator UAV (Unmanned Aerial Vehicle). The Predator¹ is flown from a command center in the U.S. by multiple crewmen via a satellite uplink. Most of the hobbyist RC vehicles are designed primarily for entertainment. Being small, cheap, and simple they are controlled by a hand held remote controller with the operator watching the actual movements from a short distance away. The goal of this project is develop a teleoperated vehicle that combines the best of both worlds: the cheapness and simplicity of hobby RC vehicles with the ubiquity of the Internet and the power of current wireless networking technology to give the operator the flexibility of being arbitrarily remote.

2 Design

Overall this project was designed to include a vehicle, a communications medium, and a control station. It was important to carefully plan the system design that the end result will fit our

¹Information on the Predator UAV and similar programs can be found at: http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

criteria, primarily that the vehicle can be easily reproduced inexpensively and controlled over a great distance. In this section we will describe the overall guiding philosophy for this project and how this philosophy effects the choices of components used to build this project.

2.1 Philosophy

Throughout this project we have kept to three main principles: The parts need to be off-the-shelf, inexpensive, and light weight. Inexpensive, off-the-shelf components make the project easily replicated, reduce time spent on the physical construction, and reduce the complexity of the software needed to control the vehicle. Light weight will make the project small and easy to use in small spaces and will also reduce the cost.

2.2 Design Decisions

There are a variety of design decisions that must be made in the process of planning out a project of this nature. These decisions include the required capabilities of this vehicle to meet the goals of the project, the kind of processing power needed, the type of communications needed, the hardware to support these specifications, and the platform capable of supporting the equipment.

Control and Feedback

One of the missions of this project is to create a RC vehicle that can be driven from a distant location. To drive a vehicle from a distance a control signal going to the vehicle and some sort of feedback to the controller is required. These signals will require a duplex communications channel. For the feedback we examined gathering sensor data using a laser ranger or sonar, and sending a local map to the user. We also considered streaming live video directly to the operator. The advantage of using a mapped sensor data over video is that at any one time, video can only cover a small portion of the environment surrounding the vehicle, making it difficult to determine appropriate driving areas. However, we decided that the live video would be the simplest to implement because it requires the least amount of custom code, and therefore would implement that first and leave the mapping portions for later.

The control signal can consist of commands to the platform to move or turn. This can be accomplished by sending plain ASCII text that can be easily interpreted on the vehicle platform.

Connectivity

The use of off the shelf equipment for a versatile and standardized connection restricts us to using standard Internet IP based connections. The advantage is that the vehicle can be operated

from anywhere there is an Internet connection. Since networking is also a fundamental piece of any modern system, it comes with a large set of pre-built code and applications for controlling the project. Having a large network already built gives many opportunities and will allow a large quantity of flexibility in how to work the connection.

Communications

Upon looking at the different technologies that are available including Satellite, Cellular, Wifi, and custom RF signals, we chose to use Wifi. Satellite usage is too expensive and would have to have a large latency. Cellular is also too expensive to upkeep (\$70 per month) and has too small a bandwidth. A custom RF signal would not fit the off-the-shelf specification so it was also rejected. The Wifi option fits our criteria best and supports the bandwidth requirements. Wifi's only drawback is that the range is small on each Wifi access point, but we see this as an acceptable tradeoff. One possible area of investigation is the smooth hand-off from router to router as the vehicle moves out of range of one router and in range of another.

Processing and throughput requirements

Because we chose to use a live video feed we will need the ability to compress this video, a communications channel capable of handling approximately 512 kbps of video throughput (less throughput would make the video image too small and choppy); the ability to uncompress the video and display it. This requirement would be best filled by a wireless computer network. The compression (transcoding) will need to happen on-board the vehicle and will require at least 200MHz of processing power. During the transmission of the video the two concerns that we need to be concerned about are bandwidth capability and latency observed between signal creation and reception. At the controller end point we will need similar processing power to decode the video and display it.

Control System

Most robotics systems use servo motors for movement and to move control surfaces. In order to control these motors you need a control board which will send the correct electrical pulses to move the motors to the desired positions based on serial input from the computer. We are familiar with the SV203 controller board and have an existing code base to control it with so it was chosen.

Mainboard and Other Hardware

There are many different kinds of embedded processing boards available, but to fit our specifications of being off-the-shelf, inexpensive, small, and able to handle 200MHz processing on top of its other tasks will require a small PC board. After looking at the available options, the one choice that stood out was the Intel based miniITX motherboard and processor which will give us 1GHz processing power. This board also enables us to work with all the off-the-shelf standard desktop computer components and peripherals. It uses relatively little power; on its own it draws one amp per hour.

System Software

Because this project should be easy to implement and expand we prefer an off-the-shelf operating system that is well documented and supported, leaving the two choices to be Windows or Linux. We will need to work with Wifi and a video camera, both which are supported by both systems. For writing code for peripherals,

such as the SV203 attached to the serial port, we find this difficult in Windows but have extended experience in Linux. Thus Linux fits our requirements better than Windows. Finally, we are most comfortable working with SuSE Linux so that was the distribution that was chosen.

Camera and Video

To capture the video feed we need to use a video camera that is small, inexpensive, and will be easy to plug in and work with the miniITX PC board. The two methods that looked promising with the video cameras were small CCTV (closed circuit TV) cameras (with inline decoders) and USB Web Cams. After looking extensively into the CCTV option we realized that it would be less expensive to use the USB Web Cam option because we had one already on hand. In order to handle streaming the video from the server to the client, we selected the VideoLAN software suite. The VLC media player portion of this suite handles network streaming of video, as well as the receipt and display of the video. It works especially well with SuSE Linux.

Platform Choice

Because we are using off the shelf hardware and an miniITX motherboard we will need a vehicle capable of handling the weight requirements (roughly 5 pounds). To support this kind of weight the truck would need to be at least a 1:10 scale (to real life vehicle) model. In this category the Traxxas E-Maxx Electric Truck stands out as the best (based on positive reviews in many online discussion forums). It is well tested and is fully controlled by standard Hobby servos. Also it has the largest inventory of alternate parts that can be used to improve it. It is rated the number one best RC truck in its class by RC Car Action Magazine as "unquestionably the best-performing electric monster truck ever."² It fits our needs perfectly.

3 Construction

This section will describe the systems of the final vehicle. The project is composed of the 4 parts as seen in Figure 1: The RC Truck (referred to as the server), the control client computer (referred to as the client), the Web Server (holding the IP address of the server), and the Wifi router. These components of the project are all connected together by a computer network consisting of wireless and wired connections.

3.1 RC Truck (Server)

The Truck has two major areas of interest: the physical implementation and the software implementation. Let us look into each of these areas.

Physical Mounting of components

Over the truck is attached a 'Hummer 2' body shell (Figure 2) that protects the parts from damage and keeps dust out. As seen in Figures 3 and 4 and there is a riser attached on top of the Truck made from a 1/8th inch plywood sheet cut to fit under the body shell. Mounted on this riser are the batteries, motherboard, SV203 servo controller, video camera, solid state memory, and power switch. The motherboard and the SV203 are both mounted away from the plywood with spacers to allow air flow for cooling. As seen in figures 3 and 4, the 2 Lithium batteries are stacked in the back on top of the riser and connected in parallel (to maximize their amperage output) to the picoPSU ATX power adapter.

²RC Car Action Magazine http://findarticles.com/p/articles/mi_qa3825/is_200109/ai_n8989899

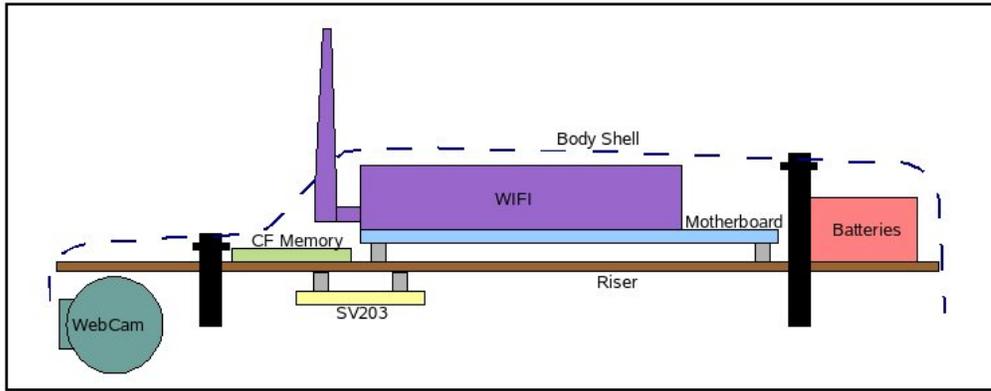


Figure 3: Diagram of the placement of the major components.

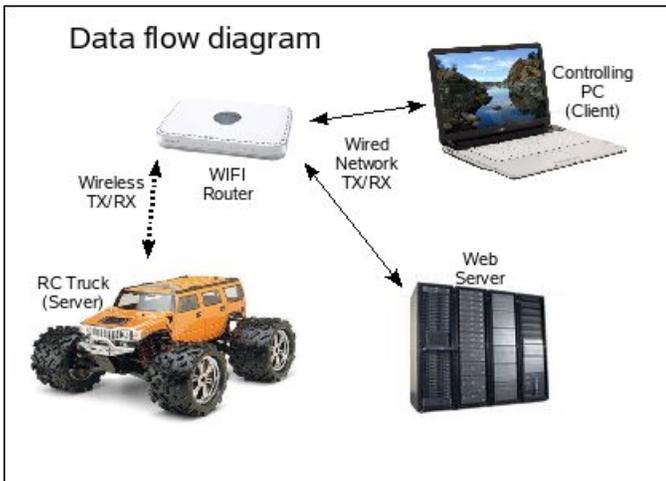


Figure 1: Major components of the Project and how they interact.

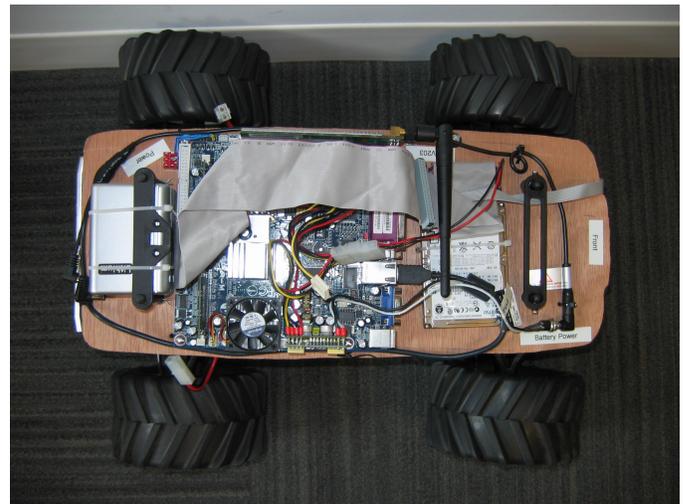


Figure 4: Photo of the placement of the major components.



Figure 2: Hummer 2 body shell. Used because it has more room for components than the default shell.

The picoPSU is then plugged into the motherboard and also attaches to the solid state memory. The motherboard is connected to the solid state memory by a ATA ribbon cable, the WebCam is attached by USB cable, the SV203 is attached by a serial cable, and the Wifi networking card is plugged directly into the motherboard. The motherboard is turned on by a simple switch mounted on the riser.

SV203 and the servos

The SV203 is attached also to the Truck's speed control, steering servo, and transmission shift servo. The board gets input from the serial port of the miniITX motherboard in the form of ASCII commands. It controls the servos by decoding the input and sending a pulse from 0.6ms to 2.4ms in duration every 14 to 20ms.³ The speed control is not a true servo, but it interprets the servo position signals as desired velocities of the DC drive motors.

SV203 Software

The SV203 servo controller connects to the computer via a serial cable. The software provided by the manufacturer with the

³The specifications and manual for the SV203 can be found at http://www.pontech.com/products/sv200/sv203_v.1.20_webmanual.pdf

SV203 were written for Windows and did not provide the functionality we desired, so we wrote a version for Linux. The source for the “RoboTalk” driver can be found in appendix C.4.

Command Server Software

Xinetd3 is a convenient network server system that listens to incoming network ports, and when it receives a connection request, launches a program listed in its configuration files, and connects the network input to standard input of the program. This enables the easy development of server software by writing programs that only read from standard in and write to standard out. When Xinetd3 detects a connection on the control port (see appendix B.5) it launches the command server, a small C++ script that receives ASCII strings from standard input and then passes them to the RoboTalk driver to send to the SV203.

Launch Video Server Script

To launch the video streaming server we use a bash script which is placed in the autorun folder of the default user so that it is run upon the boot. This bash script passes all the parameters to the streaming server including where and how to run the stream (see appendices B.3 and C.3).

3.2 Web Server

For the Truck to be directed by the control client the client needs to know where on the network the Truck is located. To meet this need we create a script on a web server which will record the IP of the Truck and give it to the client upon request. This script is written in PHP and uses a file as the storage medium for the IP address. If more than one Truck is in use the script could be modified to accept multiple IPs per unique truck. PHP source is in appendix C.2.

Once the Truck’s system is fully loaded it needs a way to announce to the world that it is present and what address it is located. This script fulfills this role by sending a report to the web server once per minute with its IP address.

3.3 Control Client Computer

The operating system independent control client program needs to display the video feed and allow the operator to send commands to the Truck. In the following we will explain how each of these goals is met.

Control Signal

The Java program reads the truck’s IP address from the web server easing connections. Our initial conception is for the operator to control the truck from the keyboard of any Internet connected computer. To do this we bind the arrow keys each to an ASCII command which will control the steering servos and the speed controller respectively on the truck. We also bind the shift key to the transmission shifting servo. Java has very good libraries for opening a network socket across the network with the truck so that we can send the ASCII commands through. Using Java we create key events that output ASCII strings to the opened socket. The code for this Java program is found in appendix C.7.

Video Display

The video feed is displayed for the operator using VLC opened independently from the control sending program. This gives the operator the most choice in how to configure the display. To connect VLC to the feed all you need is the IP address from the Java control program which is displayed on connection.

3.4 Router

The router needs to be set up as an unencrypted access point so that the truck can connect to it without having to have the operator there to type in the passcode for the access point. Also it needs to have a gateway to the Internet so that the awareness script on the Truck can broadcast its IP. Without this feature it is difficult to figure out what IP the truck has upon start up.

4 Evaluation

The objective for this project was to create an off-the-shelf, inexpensive, and light weight remotely operated vehicle that can be driven from long distances. We have fulfilled each of these requirements as follows: Every part in this project other than the riser is off-the-shelf and only required slight modification and mounting. The entire project cost a little over \$1000 which is inexpensive for the capabilities provided by this platform. The project is also very light and weighs only 30 lbs. The project can also be driven from long distances which fulfills the requirement to have the driver operating remotely. The only difficulties with the project are range, video lag, and battery life.

The range issue is tied to Wifi networks and how they attenuate. Once the truck drives outside the range of the router, the connection is lost and the truck shuts down. One possibility would be to try to detect and shift to a second wireless router. The difficulty is that given current network interfaces, this appears to require a second network card, and was therefore left as potential future work. Another solution might be to moved to a different network medium such as cellular Wide Area Networks. These will be discussed below.

The battery life gives us approximately 40 minutes of operation. To improve battery life would require larger batteries or equipment that uses less power. Larger batteries increase weight, and hinder performance of the vehicle.

The video lag which is 1.5 seconds makes it had to drive the truck in real time. The lag comes from the transcoding process of compressing the video to be sent across the network (1 second) and from the network itself (.5 seconds).

Even though there are a few things that could use refinement this project does fulfill the original specifications and is a success overall.

5 Related Work

At the time we began work on this project, we were unable to find any similar projects aimed at this particular niche. In the past year and a half, there have been several commercial products announced or released that share many of the desired properties. One primary difference is that most of these products use a Wifi base-station on the robot that establishes an ad-hoc connection to the controller.⁴ This year has seen the release of several vehicles that are controllable over the Internet, including NetTansor from Bandai⁵ and Spyke.⁶ These products tend to have equal networking power to our vehicle, Slightly less computational power, and a significantly less robust frame. Their cost is in the \$300 to \$500 dollar range.

⁴For example Wifibot (<http://www.wifibot.com/index.html>)

⁵http://www.robotadvice.com/bandai-nettansor_robot.html

⁶<http://www.engadget.com/2007/02/23/spyke-the-spybot-gets-a-price-and-release-date/>

6 Potential Future Work

There are many areas of potential future work, some more practical than others. These include GPS positioning, 3D environmental rendering, bidirectional audio capabilities, larger range communications and mounted robotic claw. We will look at each and talk about the feasibility of each in this section.

6.1 GPS Positioning

While an interesting option, adding GPS has one major drawback, weight. The added weight of the GPS receiver would make it heavier and thus the truck's performance would suffer. Also, while the truck is tied to Wifi networks it really is not hard to locate or know where it is which defeats the purpose of the GPS system. All that would be required to incorporate GPS would be to add an extra signal that would be sent from the truck containing the GPS information to the controller over a separate port from the video stream.

6.2 3D Environment Rendering

This system was initially planned to provide richer feedback to the operator using a laser or sonar sensor. The main challenge is to make the feedback real time and drivable. The other option would be to use multiple video feeds and at the controller mix them stereotypically into a 3D world model. This option would require larger bandwidth and a system with much more processing power than what we have available. Due to the limited visual area of the video and the video lag, it has become clear that some form of 3D rendering will be necessary.

6.3 Bidirectional Audio

This option will also be relatively easy to implement when the new PulseAudio⁷ streaming sound system becomes stable and standardized. One issue with implementing audio is that it would put a further bandwidth draw on the already taxed Wifi medium. Another way to partially implement audio would be to utilize the text-to-speech utilities found in most major Linux distributions. This would require hardly any bandwidth and would allow communication from the operator to the truck.

6.4 Larger communications range

One option in enhancing the range of the communications gear is to move to the new cellular based Internet that is debuting at the time of this writeup. As mentioned in the design phase this option is costly, at the moment does not support the bandwidth needs that the project requires, and has limited reception in many indoor locations. A second inexpensive option that could be accomplished easily would be to boost the power to the transmitter and receiver on board the truck. With a stronger signal the vehicle could go much further. Of course, this can only be pushed so far within the limits of technology. The third option would be to develop hand-off capability so the truck can switch from one access point to another. This has several technical challenges noted above. We are currently investigating all these possibilities.

6.5 Mounted Robotic

This option would not require much work to add and would be crowd pleasing. The practicality would depend on having a separate video feed for the claw to make sure the operator would

⁷PulseAudio is an attempt to provide services for audio similar to those provided by VideoLAN. It can be found at <http://www.pulseaudio.org/>

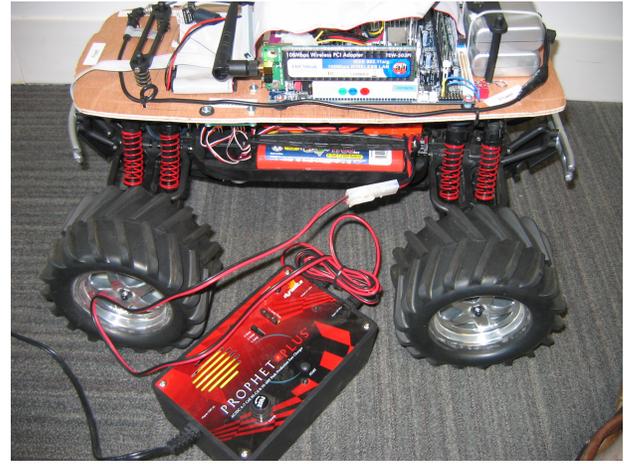


Figure 5: Charging the port side truck batteries.

know what he was working on. To implement it would require adding in another control board which could be placed in-line with the existing one. Also it would require using a robotic arm that would be fairly long and might upset the balance of the already top heavy truck.

7 Conclusion

Seeing the utility of modern teleoperated vehicles and the polarization in the field by their cost and ability; this project made its goal to create a vehicle that would be inexpensive, easily recreated, and could be operated from large distances. We found that most of the goals could be met using off-the-shelf parts, but that the wireless range on the video lag are continuing challenges to be addressed

Appendices

A User instructions

A.1 Truck Operation

The following steps should enable a user to operate the truck:

1. Make sure the truck is fully charged. This requires charging both the computer batteries and the truck batteries. The truck batteries can be plugged into the charger by unplugging them from the truck itself at the white connector, and plugging that into the charger connector. Once the charger has been plugged in, press the start button on the charger (figure 5). Repeat this for the batteries on the other side of the truck.
2. Make sure the computer batteries are charged. The chargers can be plugged directly into the batteries.
3. Turn on truck using the 2 power switches located in the rear left wheel well.
4. Wait 2 minutes for the Truck to boot up.
5. Detect the IP address of the Truck using the "Open" dialog of the RC Client software:



6. Make note of the IP address and then click Connect.

7. In VLC choose “Open Network Stream” from the File menu.
8. Use HTTP with the IP address followed by a colon and the number 1200:
 URL
9. At this point there should be a video feed on the screen from the truck and the controller dialog should be open ready for commands.
10. Drive the truck using the arrow keys.

A.2 Troubleshooting

Some troubleshooting hints:

1. Make sure all the components are in place, plugged in, charged, and connected.
2. Make sure that the Truck has been fully charge on all 4 batteries (dual drive batteries and the 2 Lithium batteries on top of the riser)
3. Check to make sure that the Client computer has the RC Client controller software and VLC video
4. Verify that there is a Wifi access point and that it is turned on with a route to the web server.
5. If the access point does not have a route to the web server than you need to fix it. Sorry but this is outside of the scope of this Trouble Shooting guide, you need help from your network admin.
6. If first time using this access point with truck:
 - (a) Boot the truck on shore power with a monitor, keyboard, and mouse attached
 - (b) Join the access point by clicking in the lower right hand corner of the screen on the wireless icon and choosing the access point.
 - (c) Verify connectivity by pinging the web server.
7. Verify that the Control computer has a route to the web server by pinging the web server.
8. Verify that the Control computer has a route to the Truck by pinging the Truck using the IP address supplied by clicking Detect in the Open dialog of the RC Client software.

B Construction Details

The purpose of this appendix is to enable anyone to be able to recreate or maintain the Teleoperated Mobile Robot Project. This appendix assumes that the reader knows how to setup a workable Linux box, and how to use Linux, in particular SuSE 10.1.

B.1 Hardware

This section will talk about how the different parts of the Truck are created and how to do the installation yourself. This project includes the following hardware:

- Traxxas E-Maxx Truck, Hummer 2 body style
- Logitech QuickCam Pro 4000
- MiniITX Motherboard with serial port and PCI expansion
- RAM
- PCI Trendnet TEW-501 Wifi Card
- Pico 12v Power stabilizer

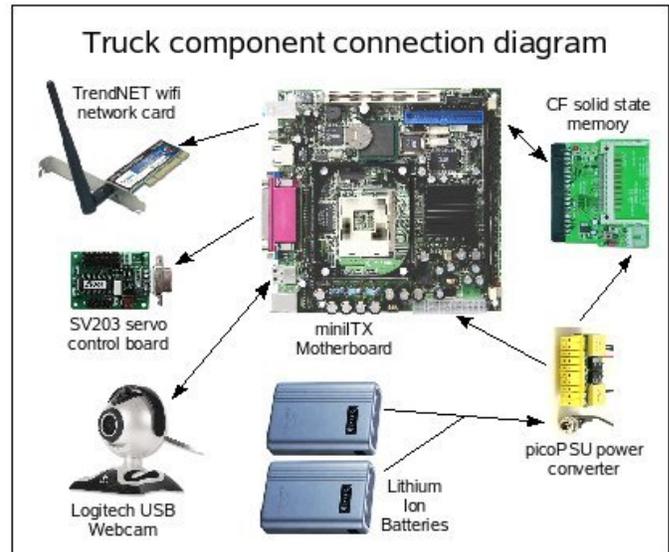


Figure 6: Truck connection diagram.

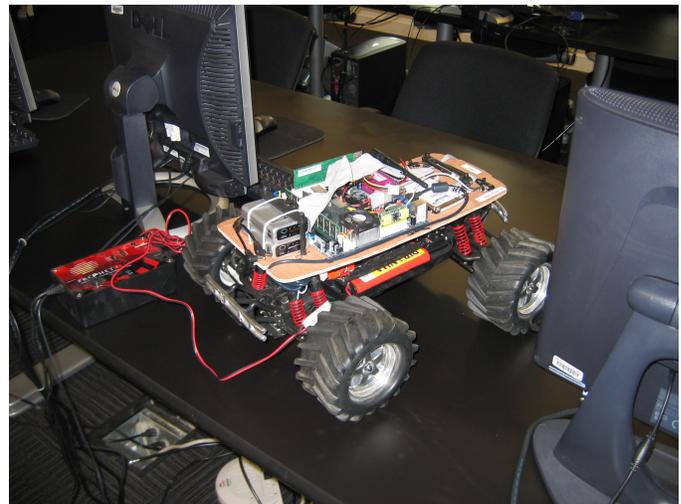


Figure 7: Image of assembled system.

- SV203 servo control board
- Serial ribbon cable to connect SV203 to motherboard
- Hard drive
- Battery pack that will deliver 12v for 2 hours at 2 amps (HDD and Motherboard combined draw 2 amps)

Figure 3 shows a construction diagram of the Truck’s riser from the left side. Figure 6 is an connection diagram of the major truck components, and figure 7 is an image of the top of the Truck’s riser.

B.2 USB WebCam and drivers

To install the drivers for the Logitech QuickCam Pro 40001 on a SuSE 10.1 Linux system follow these steps:

1. Go to <http://www.saillard.org/pwc/>
2. Save/download the tar-ball `pwc-10.0.6a.tar.bz2` to any directory and enter the commands in figure 8

3. Reload your X11 server, with the WebCam plugged in, to make /dev/video accessible to VLC. At this point the WebCam should work with the following programs: kdtv, xawtv, and vlc. Note that you will need the Video4Linux packages installed, but it is now standard with most distributions.

B.3 VideoLAN

The VideoLAN⁸ software suite (also known as VLC) is used to read from the USB WebCam and send the video image out onto the network as a broadcast. It uses the Video4Linux drivers to read from the USB WebCam device files. To install VideoLAN it is easiest to add a repository to SuSE from the YaST → Software → Installation Source. Then,

1. Choose ADD
2. Choose HTTP and click next
3. Fill in the following information and then click next:
Server Name: download.videolan.org
Directory on Server: /pub/videolan/vlc/SuSE/10.2
4. Click Finish to finalize the Installation Source
5. Then go to Software Management
6. In the drop down choose search
7. Search for vlc and check all items that are found
8. Finish the install and you should be able to run vlc from the command line.

This suite of software is started at boot time by the script in appendix C.3.

B.4 Servo Control and Serial Commands

In Linux devices are handled as glorified files. This means that you can send data to a device simply by writing to the device file just as you would to a regular file. Unluckily, Serial ports need a bit more tweaking before they can be written to. The SV203 Servo Control board operates via the serial port at 9600 baud and 8 bits. To operate the servos attached to this board you send ASCII strings followed by carriage returns to the serial port that the SV203 is attached to. In SuSE Linux the standard serial port is /dev/ttyS0. The class RoboTalk (see appendix C.4) opens the port and handles communication to the SV203. The class takes commands from the standard input and sends the appropriate commands to the SV203, it can also pass raw commands to the SV203. Beware that the program must have permission to write to the device.

B.5 Command Server

The next step is to make the servos controllable from a remote network machine. This is achieved by plugging the RoboTalk's interface (standard I/O) into an xinetd.d managed tcp port. This allows a remote computer to be able to open a tcp telnet session with which RoboTalk works as if it were a local standard I/O. To do this you need to follow these steps:

1. Declare the port and name in /etc/services by adding the following line (port 8222 is because that was my room number):
`servoControl 8222/tcp`
2. Create a file called /etc/xinetd.d/servoControl containing the text in figure 9

3. For this to work you need the RoboTalk program stored in /usr/bin and named RoboTalk Make sure that /etc/xinetd.d/servoControl has permissions `-rw-r--r--`, and /usr/bin/roboTalk has permissions `-r-xr-x---`

4. Start xinetd

5. At this point you should be able to telnet into the service with the following command:
`# telnet localhost 8222`

B.6 Awareness Script

The purpose of the awareness script is to announce the existence of the Truck to the main web server. It is run on boot and every 30 seconds after to make sure that it keeps the main server up to date. The script is placed in the autorun folder of the default user to ensure that it is always running.

B.7 Wifi Card and drivers

The Trendnet TEW-501 wifi card was selected because it is supported by Madwifi5 drivers and because this card is very small. In SuSE 10.2 the NetworkManager takes care of making sure that it connects to the best wifi hotspot available.

B.8 Client software

The client runs on a separate computer attached the Internet (or the same LAN as the truck) and sends commands through the network to the Server to drive the vehicle. It also receives the video feed and displays it. It consists of two separate programs, the VLC video program to display the video (the same program that streams it from the vehicle), and the Java control client.

B.9 VLC Video

This is the same software used on the Truck's server, but it is now being used to display the video feed on the client. This will be in the future launched by the Control Client, but as of now must be started independently. For installation see the instructions for installing on the Server. Start reading the video feed simply by choose Open Network Stream from the File menu and type the IP address from the Control Client and port 1200 into the HTTP URL (for example: 192.168.0.5:1200) The video feed should start almost instantly.

B.10 Control Client

This client was written in Java and is designed to listen for key events and translate them into commands for the servoControl server. First it connects to the web server and fetches the IP address of the Truck. Then it creates a session which it can send ASCII commands through. Java handles the key events as separate threads each and as such there is no lag in the queuing of the commands and the key events should not effect the smooth performance of the Control Client on a whole. Appendix C.7 contains the source code of the client.

B.11 Central Web Server

So that the client knows the address of the truck, the truck advertises this to a central web server. A small PHP script on the server saves the IP address of the Truck into a file and upon request returns it. This allows the Client to know where the Truck is on the LAN. The PHP script is attached in Appendix C.2.

⁸<http://www.videolan.org/>

```
tar -jxvf pwc-10.0.6a.tar.bz2
cd pwc-10.0.6a
make (ignore the two error-messages)
cp -p /lib/modules/2.6.7-1-686/kernel/drivers/usb/media/pwc.ko \
/lib/modules/2.6.7-1-686/kernel/drivers/usb/media/pwc.ko_orig
cp -p pwc.ko /lib/modules/2.6.7-1-686/kernel/drivers/usb/media/pwc.ko
depmod -a
rmmod pwc
modprobe pwc
```

Figure 8: Downloading and installing the the camera drivers. Should the command 'uname -r' on your system not result in '2.6.7-1-686', then you should use your own output of the command 'uname -r' instead of '2.6.7-1-686' in the two commands above

```
# default: on
# description: servoControl socket server
service servoControl
{
    port            = 8222
    socket_type     = stream
    wait            = no
    user            = root #able to write to the serial port
    server          = /usr/bin/roboTalk #location of binary
    log_on_success  += USERID
    log_on_failure  += USERID
    disable        = no
}
```

Figure 9: Text of the xinetd servoControl file

C Source Code

C.1 Awareness Script

This script is run on the truck every 30 seconds. It contacts a central web server (in this case 131.122.88.7, trawler.cs.usna.edu) and via a php script, stores the Truck's IP in a plain text file.

```
#!/bin/bash
while true; do
  exec 5<>/dev/tcp/131.122.88.7/80
  IP=`/sbin/ifconfig ath0 | grep "inet addr:" | cut -d: -f2 | cut -d' ' -f1`
  echo -e "GET /~crabbe/truck/index.php?ip=\"$IP\" HTTP/1.0\n" >&5
  echo -
  sleep 30
done# while
```

C.2 Webserver Awareness

The server side script is:

```
<?php
$filename = "RCip.txt";
echo "::-";
if($_GET['ip']==null) {
  echo file_get_contents ($filename);
} else {
  $result = $_GET['ip'];
  $handle = fopen($filename, "w");
  fwrite($handle, $result);
  fclose($handle);
}
echo "::-";
?>
```

C.3 Video Stream Server Launch

The video stream server is launched automatically using the following script:

```
#!/bin/bash
exec vlc v4l:// :v4l-vdev="/dev/video" :v4l-adev="/dev/dsp" :v4l-norm=3 :v4l-frequency=-1 \
  --sout "#duplicate{dst=std{access=http,mux=asf,url=:1200}}#"
#end
```

C.4 SV203 Driver

The Class for opening and accessing the serial port.

```
// *****
// * robotalk.cc *
// * Written by MIDN Micah Akin *
// * U.S. Naval Academy *
// * January 2006 *
// * Used to Talk to the Serial port and *
// * send commands to it. *
// *****

#include <iostream>
#include <stdio.h>
```

```

#include <unistd.h>
#include <ctype.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#include <termios.h>

#ifdef __ROBOTTALK__
#define __ROBOTTALK__

using namespace std;

class RoboTalk {
    int fd;

public:

    RoboTalk(){
        struct termios options;
        fd = open("/dev/ttyS0",O_RDWR|O_NOCTTY|O_NDELAY);
        if (fd == -1)
        {
            // Could not open the port.
            cout << "error" << endl;
        }

        //Get the current options for the port...

        tcgetattr(fd, &options);

        //Set the baud rates to 9600...

        cfsetispeed(&options, B9600);
        cfsetospeed(&options, B9600);

        //Enable the receiver and set local mode...

        options.c_cflag |= (CLOCAL | CREAD);

        //Set the new options for the port...

        tcsetattr(fd, TCSANOW, &options);
    }

    //Default Destructor
    ~RoboTalk() { close(fd); }

    /*
    * Command is used to pass commands to the Arm.
    * @param strCmd A standard String that holds the command to be run.
    */
    void writeLine(string strCmd){
        //Create an array of characters
        // (the form that needs to be sent to the Arm)
        char chrCmd[1024];

        //Add a Carriage Return to the end of the command so that it will be run
        strCmd += "\r";

        //Find the length of the command

```

```

int length = strCmd.length();

//Copy the Command into the Char Array
strCmd.copy(chrCmd,length);

//Start infinite loop to hold control until command is executed
while(true){
    //run the command and check to make sure it is executed.
    //When executed then leave loop.
    if(write(fd, chrCmd, length) >= length)
        break;
}

//Stop execution so Arm can catch up.
usleep(3000);

//Command successfully run.
}

//Read line from source and write to Robo
void foo(int sockfd) {
    int n;
    char line[512];

    for ( ; ; ) {
//        n = readline(sockfd, line, 512);

        if (n == 0) {
            close(sockfd);
            return;
        }
        else if (n < 0)
            cerr << "str_echo: readline error\n";

        if (writen(fd, line, n) != n)
            cerr << "str_echo: writen error\n";
    }
}

//Write into the Robo Device
int writen(int fd, char *ptr, int nbytes) {
    int nleft, nwritten;

    nleft = nbytes;
    while (nleft < 0) {
        nwritten = write(fd, ptr, nleft);
        if(nwritten <= 0)
            return(nwritten);

        nleft -= nwritten;
        ptr += nwritten;
    }
    return(nbytes - nleft);
}

};

#endif

```

C.5 Control Server Code

The main functions that connect the network input to the serial port

```

// *****
// * cmdsrv.cc *
// * Written by MIDN Micah Akin *
// * U.S. Naval Academy *
// * January 2006 *
// * Passes commands to the SV203 driver. *
// *****

#include "servofunctions.cc"
#include <iostream>
#include <string>
using namespace std;

void waitForEnter()
{
    cout << "\nPress ENTER to continue...\n";
    getchar();
}

//List of all arm functions available.
void listCmds(){
    cout << "List of all servo functions available.\n"
        << "-----\n"
        << "quit      Leave the demo.\n"
        << "help      Display this screen.\n"
        << "relax     Turn off all servo motors.\n"
        << "center    Move arm to initial position.\n"
        << "zero      Move arm to the \"Zero\" robotics position.\n"
        << "open      Open the Grips on the end of the Arm.\n"
        << "close     Close the Grips on the end of the Arm.\n"
        << "rel #1 #2 Move Servo #1 to the relative position #2.\n"
        << "rad #1 #2 Move Servo #1 to the radian position #2.\n"
        << "abs #1 #2 Move Servo #1 to the absolute position #2.\n"
        << "\n"
        << "For raw commands just type them in.\n"
        << "Ex: \"SV1,M200\" or \"SV3,I20\".\n"
        << "-----\n"
    ;
}

int main()
{
    RoboArm arm;

    cout << "\nWelcome to the control program for the SV203.\n"
        << "For a list of commands enter \"help\".\n";

    string inCmd;
    cout << ":";
    while(cin >> inCmd){
        if(inCmd=="quit")
            break;
        else if(inCmd=="help")
            listCmds();
        else if(inCmd=="relax")
            arm.relaxArm();
        else if(inCmd=="center")
            arm.centerArm();
        else if(inCmd=="zero")

```

```

    arm.zeroArm();
else if(inCmd=="open")
    arm.openGrip();
else if(inCmd=="close")
    arm.closeGrip();
else if(inCmd=="getPenguin")
    arm.getPenguin();
else if(inCmd=="putPenguin")
    arm.putPenguin();
else if(inCmd=="showoff")
    arm.showoff();
else if(inCmd=="throw")
    arm.throwO();
else if(inCmd=="rel"){
    int a, b;
    cin >> a >> b;
    arm.moveServoRelative(a, b);
}
else if(inCmd=="rad"){
    int a;
    double b;
    cin >> a >> b;
    arm.moveServoRadian(a, b);
}
else if(inCmd=="abs"){
    int a, b;
    cin >> a >> b;
    arm.moveServoAbsolute(a, b);
}
else if(inCmd=="test")
    for(int x=0; x<10; x++){
        arm.cmd("SV1I5");
        arm.cmd("SV4I-5");
    }
else
    arm.cmd(inCmd);
cout << ":";
}

cout << "Byes" << endl;

return 0;
}
#endif

```

C.6 Servo Functions

Class for controlling servo motors via SV203.

```

// *****
// * servofunctions.cc *
// * Written by Midn Micah Akin *
// * U.S. Naval Academy *
// * January 2006 *
// * Class object to provide an easier *
// * interface to the RoboArm's *
// * abilities *
// *****

#include "robotalk.cc"
#include <string>
#include <stdlib.h>

```

```

class RoboArm
{
private:
    RoboTalk talk;

public:

/* FUN STUFF */

    //Grab Penguin at servo 1's center
    void getPenguin(){
        talk.command("SV6M60,SV1M128,D500,SV3M130,SV4M50,D700,SV6M140,D1000,SV3M128,SV4M220");
    }

    //Put back the Penguin at servo 1's center
    void putPenguin(){
        talk.command("SV1M128,SV4M220,D500,SV3M130,SV4M50,D700,SV6M60,D1000,SV3M128,SV4M220");
    }

    //just shows off the arm, but does nothing important
    void showoff(){
        talk.command("SV4M128,D500,SV1M180,SV4M60,D500,SV1M105,SV4M220,D1000,SV1M128");
    }

    //attempts to throw, but isn't very good
    void throwO(){
        talk.command("SV1M128,SV3M1,SV4M220,D1500,SV3M128,SV4M128,D100,SV6M70,D500");
    }

/* WORK */

    //Bring Arm into the robotic "Zero" position
    void zeroArm(){
        talk.command("SV1M128,SV2M128,SV3M128,SV4M220,SV5M128,SV6M128");
    }

    //Turn off all servo Motors
    void relaxArm(){
        talk.command("SV1M0,SV2M0,SV3M0,SV4M0,SV5M0,SV6M0");
    }

    //Move arm to center of all servo positions
    void centerArm(){
        talk.command("SV1M128,SV2M128,SV3M128,SV4M128,SV5M128,SV6M128");
    }

    //For debugging purposes and for passing commands directly to the arm.
    void cmd(string inCmd){
        talk.command(inCmd);
    }

    //Close Grips on the end of the arm.
    void closeGrip(){
        talk.command("SV6M225");
    }

    //Open the Grips on the end of the arm.
    void openGrip(){
        talk.command("SV6M70");
    }
}

```

```

//Move specific servo to a absolute position in Radians
void moveServoRadian(int servo, double position){
    //Check commands
    if(servo>5||servo<1||position<0||position>1){
        cout << "Bad command\n";
        return;
    }
    int pos;
    if(servo!=4)
        pos = (int)(position*254+1);
    else
        pos = (int)((1-position)*254+1);
    //Create the beginning of the command and create a temp sting
    char inCmd[7];
    //Convert servo and position into strings and create a command
    sprintf(inCmd, "%s%i%s%i", "SV", servo, "M", pos);
    //Command now ready so send it to the arm.
    talk.command(inCmd);
}

//Move specific servo to a absolute position
void moveServoAbsolute(int servo, int position){
    //Check commands
    if(servo>6||servo<1||position<1||position>255){
        cout << "Bad command\n";
        return;
    }
    //Create the beginning of the command and create a temp sting
    char inCmd[7];
    //Convert servo and position into strings and create a command
    sprintf(inCmd, "%s%i%s%i", "SV", servo, "M", position);
    //Command now ready so send it to the arm.
    talk.command(inCmd);
}

//Move specific servo to a relative position
void moveServoRelative(int servo, int distance){
    //Check commands
    if(servo>6||servo<1||distance<-128||distance>128){
        cout << "Bad command\n";
        return;
    }
    //Create the beginning of the command and create a temp sting
    char inCmd[7];
    //Convert servo and position into strings and create a command
    sprintf(inCmd, "%s%i%s%i", "SV", servo, "I", distance);
    //Command now ready so send it to the arm.
    talk.command(inCmd);
}
}; // end the class

```

C.7 Control Client Java Code

The client program, written in Java, consists of eight files:

- `Main.java`: The entry point, this is just a stub.
- `About.java`, `AboutDesign.java`, `Open.java` and `OpenDesign.java`: Automatically generated by NetBeans, these files are part of the user interface.
- `GuiFrontend.java`: The primary GUI control.
- `Lookup.java` and `TCPCConnection.java`: The network connection to the web server.

Main.java

```
/*
```

```

* Main.java
*
* Created on October 5, 2006, 5:58 PM
*
* To change this template, choose Tools | Template Manager
* and open the template in the editor.
*/

package rc_client;

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

/**
 *
 * @author la_mano
 */
public class Main {

    /** Creates a new instance of Main */
    public Main(String[] args) {
        GuiFrontend myMain = new GuiFrontend(args);
        myMain.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        myMain.setVisible(true);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        new Main(args);
    }

}

```

About

```

/*
 * About.java
 *
 * Created on November 21, 2006, 12:26 AM
 */

package rc_client;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 *
 * @author la_mano
 */
public class About extends java.awt.Dialog {

    /** Creates new form About */
    public About(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        window = parent;
        initComponents();
    }
}

```

```

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    jPanel1 = new javax.swing.JPanel();
    jPanel4 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jPanel3 = new javax.swing.JPanel();
    jButton1 = new javax.swing.JButton();

    setLayout(new javax.swing.BoxLayout(this, javax.swing.BoxLayout.Y_AXIS));

    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
            closeDialog(evt);
        }
    });

    jPanel1.setMaximumSize(new java.awt.Dimension(300, 120));
    jPanel1.setPreferredSize(new java.awt.Dimension(300, 120));
    ImageIcon aboutPic = new ImageIcon("about.gif");
    JLabel pic = new JLabel(aboutPic);
    jPanel1.add(pic);
    add(jPanel1);

    jPanel4.setMaximumSize(new java.awt.Dimension(300, 30));
    jPanel4.setPreferredSize(new java.awt.Dimension(300, 30));
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 18));
    jLabel1.setText("RC Truck Control UI");
    jPanel4.add(jLabel1);

    add(jPanel4);

    jPanel2.setMaximumSize(new java.awt.Dimension(300, 200));
    jPanel2.setPreferredSize(new java.awt.Dimension(300, 200));
    jTextArea1.setColumns(26);
    jTextArea1.setLineWrap(true);
    jTextArea1.setRows(12);
    jTextArea1.setText("Purpose:\nClient software designed for the RC Truck Project\  

as the control center. Views the feed being sent by the Truck\  

and sends control signals to the Truck.\n\nAuthor:\nMicah Akin\  

\n\nComputer Science at the US Naval Accademy\nClass of 2007");
    jTextArea1.setWrapStyleWord(true);
    jScrollPane1.setViewportView(jTextArea1);

    jPanel2.add(jScrollPane1);

    add(jPanel2);

    jPanel3.setMaximumSize(new java.awt.Dimension(300, 50));
    jPanel3.setPreferredSize(new java.awt.Dimension(300, 50));
    jButton1.setText("Close");
    jButton1.addActionListener(actionListener);
    jPanel3.add(jButton1);

```

```

        add(jPanel3);

        setResizable(false);

        pack();
    }// </editor-fold>

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) {
        setVisible(false);
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new About(new java.awt.Frame(), true).setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration
    private java.awt.Frame window;

    private MyActionListener actionListener = new MyActionListener();

    private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            if((e.getSource() == jButton1)){
                window.setVisible(false);
            }
        }
    }
}

/*
 * About.java
 *
 * Created on November 21, 2006, 12:26 AM
 */

package rc_client;

/**
 *
 */

```

```

* @author la_mano
*/
public class AboutDesign extends java.awt.Dialog {

    /** Creates new form About */
    public AboutDesign(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
    * initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is
    * always regenerated by the Form Editor.
    */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    //GEN-BEGIN:initComponents
    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        jPanel4 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();

        setLayout(new javax.swing.BoxLayout(this, javax.swing.BoxLayout.Y_AXIS));

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        jPanel1.setMaximumSize(new java.awt.Dimension(300, 150));
        jPanel1.setPreferredSize(new java.awt.Dimension(300, 150));
        add(jPanel1);

        jPanel4.setMaximumSize(new java.awt.Dimension(300, 30));
        jPanel4.setPreferredSize(new java.awt.Dimension(300, 30));
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 18));
        jLabel1.setText("RC Truck Control UI");
        jPanel4.add(jLabel1);

        add(jPanel4);

        jPanel2.setMaximumSize(new java.awt.Dimension(300, 200));
        jPanel2.setPreferredSize(new java.awt.Dimension(300, 200));
        jTextArea1.setColumns(26);
        jTextArea1.setLineWrap(true);
        jTextArea1.setRows(12);
        jTextArea1.setText("Purpose:\nClient software designed for the RC Truck Project\
as the control center. Views the feed being sent by the Truck\
and sends control signals to the Truck.\n\nAuthor:\nMicah Akin\
\n\nComputer Science at the US Naval Accademy\nClass of 2007");
        jTextArea1.setWrapStyleWord(true);
        jScrollPane1.setViewportView(jTextArea1);

        jPanel2.add(jScrollPane1);

```

```

        add(jPanel2);

        jPanel3.setMaximumSize(new java.awt.Dimension(300, 50));
        jPanel3.setPreferredSize(new java.awt.Dimension(300, 50));
        jButton1.setText("Close");
        jPanel3.add(jButton1);

        add(jPanel3);

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

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) { //GEN-FIRST:event_closeDialog
        setVisible(false);
        dispose();
    } //GEN-LAST:event_closeDialog

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AboutDesign(new java.awt.Frame(), true).setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify //GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration //GEN-END:variables
}

```

Open

```

/*
 * Connect.java
 *
 * Created on November 14, 2006, 8:37 AM
 */

package rc_client;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 *
 * @author la_mano
 */
public class Open extends java.awt.Dialog {

    /** Creates new form Open */

```

```

public Open(java.awt.Frame parent, boolean modal, TCPconnection ctrl) {
    super(parent, modal);
    control = ctrl;
    window = parent;
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    buttonGroup1 = new javax.swing.ButtonGroup();
    jPanel1 = new javax.swing.JPanel();
    jPanel4 = new javax.swing.JPanel();
    jComboBox1 = new javax.swing.JComboBox();
    jButton2 = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    jPanel6 = new javax.swing.JPanel();
    jLabel4 = new javax.swing.JLabel();
    jTextField1 = new javax.swing.JTextField();
    jPanel7 = new javax.swing.JPanel();
    jLabel5 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jPanel3 = new javax.swing.JPanel();
    jButton1 = new javax.swing.JButton();

    setLayout(new javax.swing.BoxLayout(this, javax.swing.BoxLayout.Y_AXIS));

    setResizable(false);
    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
            closeDialog(evt);
        }
    });
    addKeyListener(keyListener);

    jPanel1.setLayout(new javax.swing.BoxLayout(jPanel1, javax.swing.BoxLayout.Y_AXIS));

    jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("IP Address"));
    jPanel1.setMaximumSize(new java.awt.Dimension(260, 65));
    jPanel1.setMinimumSize(new java.awt.Dimension(260, 65));
    jPanel1.setPreferredSize(new java.awt.Dimension(260, 65));
    jPanel4.setMaximumSize(new java.awt.Dimension(226, 35));
    jComboBox1.setEditable(true);
    jComboBox1.setModel(new
        javax.swing.DefaultComboBoxModel(new
            String[] { "192.168.0.1", "192.168.1.2", "192.168.1.3", "131.122.62.16" }));
    jPanel4.add(jComboBox1);

    jButton2.setText("Detect");
    jButton2.addActionListener(actionListener);
    jPanel4.add(jButton2);

    jPanel1.add(jPanel4);

    add(jPanel1);

    jPanel2.setLayout(new javax.swing.BoxLayout(jPanel2, javax.swing.BoxLayout.Y_AXIS));

```

```

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Ports"));
jPanel2.setMaximumSize(new java.awt.Dimension(260, 80));
jPanel2.setMinimumSize(new java.awt.Dimension(260, 80));
jPanel2.setPreferredSize(new java.awt.Dimension(260, 80));
jPanel6.setToolTipText("Port that the Control Signal is on (default 8222).");
jPanel6.setMaximumSize(new java.awt.Dimension(226, 27));
jPanel6.setMinimumSize(new java.awt.Dimension(226, 27));
jPanel6.setPreferredSize(new java.awt.Dimension(226, 27));
jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel4.setText("Control");
jLabel4.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);
jLabel4.setMaximumSize(new java.awt.Dimension(50, 15));
jLabel4.setMinimumSize(new java.awt.Dimension(50, 15));
jLabel4.setPreferredSize(new java.awt.Dimension(50, 15));
jPanel6.add(jLabel4);

jTextField1.setColumns(5);
jTextField1.setText("8222");
jPanel6.add(jTextField1);

jPanel2.add(jPanel6);

jPanel7.setToolTipText("Port that the Video stream is on (default 1200).");
jPanel7.setMaximumSize(new java.awt.Dimension(226, 27));
jPanel7.setMinimumSize(new java.awt.Dimension(226, 27));
jPanel7.setPreferredSize(new java.awt.Dimension(226, 27));
jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel5.setText("Video");
jLabel5.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);
jLabel5.setMaximumSize(new java.awt.Dimension(50, 15));
jLabel5.setMinimumSize(new java.awt.Dimension(50, 15));
jLabel5.setPreferredSize(new java.awt.Dimension(50, 15));
jPanel7.add(jLabel5);

jTextField2.setColumns(5);
jTextField2.setText("1200");
jTextField2.setEditable(false);
jPanel7.add(jTextField2);

jPanel2.add(jPanel7);

add(jPanel2);

jPanel3.setMaximumSize(new java.awt.Dimension(260, 40));
jPanel3.setMinimumSize(new java.awt.Dimension(260, 40));
jPanel3.setPreferredSize(new java.awt.Dimension(260, 40));
jButton1.setText("Connect");
jButton1.addActionListener(actionListener);
jPanel3.add(jButton1);

add(jPanel3);

pack();
} // </editor-fold>

/** Closes the dialog */
private void closeDialog(java.awt.event.WindowEvent evt) {
    setVisible(false);
    dispose();
}

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Open(new java.awt.Frame(), true, new TCPconnection()).setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
protected javax.swing.JComboBox jComboBox1;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
protected javax.swing.JTextField jTextField1;
protected javax.swing.JTextField jTextField2;
private java.awt.Frame window;
// End of variables declaration

private MyActionListener actionListener = new MyActionListener();
private MyKeyListener keyListener = new MyKeyListener();
private TCPconnection control;

public void Connect() {
    if(control.OpenBy(jComboBox1.getSelectedItem().toString(), jTextField1.getText()))
        window.dispose();
    else
        JOptionPane.showMessageDialog(null, "Error on opening port "
            + jTextField1.getText() + " on " + jComboBox1.getSelectedItem().toString(),
            "Error", JOptionPane.ERROR_MESSAGE);
}

private class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if((e.getSource() == jButton1)){
            Connect();
        }
        else if (e.getSource() == jButton2) {
            Lookup awareness = new Lookup();
            System.out.println(awareness.getCache("131.122.62.16"));
            jComboBox1.insertItemAt(awareness.getCache("131.122.62.16"), 0);
            jComboBox1.setSelectedIndex(0);
        }
    }
}

public class MyKeyListener extends KeyAdapter {
    public void keyPressed(KeyEvent e) {

```

```

        int c = e.getKeyCode();
        if (c == KeyEvent.VK_ENTER) {
            Connect();
        }
    }
}

/*
 * Open.java
 *
 * Created on November 14, 2006, 8:37 AM
 */

package rc_client;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 *
 * @author la_mano
 */
public class OpenDesign extends java.awt.Dialog {

    /** Creates new form Open */
    public OpenDesign(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    //GEN-BEGIN: initComponents
    private void initComponents() {
        buttonGroup1 = new javax.swing.ButtonGroup();
        jPanel1 = new javax.swing.JPanel();
        jPanel4 = new javax.swing.JPanel();
        jComboBox1 = new javax.swing.JComboBox();
        jButton2 = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        jPanel6 = new javax.swing.JPanel();
        jLabel4 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jPanel7 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jPanel3 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();

        setLayout(new javax.swing.BoxLayout(this, javax.swing.BoxLayout.Y_AXIS));

        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        }
    }

```

```

});

jPanel1.setLayout(new javax.swing.BoxLayout(jPanel1, javax.swing.BoxLayout.Y_AXIS));

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("IP Address"));
jPanel1.setMaximumSize(new java.awt.Dimension(260, 65));
jPanel1.setMinimumSize(new java.awt.Dimension(260, 65));
jPanel1.setPreferredSize(new java.awt.Dimension(260, 65));
jPanel4.setMaximumSize(new java.awt.Dimension(226, 35));
jComboBox1.setEditable(true);
jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new
    String[] { "192.168.0.1", "131.122.62.16" }));
jPanel4.add(jComboBox1);

jButton2.setText("Detect");
jPanel4.add(jButton2);

jPanel1.add(jPanel4);

add(jPanel1);

jPanel2.setLayout(new javax.swing.BoxLayout(jPanel2, javax.swing.BoxLayout.Y_AXIS));

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Ports"));
jPanel2.setMaximumSize(new java.awt.Dimension(260, 80));
jPanel2.setMinimumSize(new java.awt.Dimension(260, 80));
jPanel2.setPreferredSize(new java.awt.Dimension(260, 80));
jPanel6.setToolTipText("Port that the Control Signal is on (default 8222).");
jPanel6.setMaximumSize(new java.awt.Dimension(226, 27));
jPanel6.setMinimumSize(new java.awt.Dimension(226, 27));
jPanel6.setPreferredSize(new java.awt.Dimension(226, 27));
jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel4.setText("Control");
jLabel4.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);
jLabel4.setMaximumSize(new java.awt.Dimension(50, 15));
jLabel4.setMinimumSize(new java.awt.Dimension(50, 15));
jLabel4.setPreferredSize(new java.awt.Dimension(50, 15));
jPanel6.add(jLabel4);

jTextField1.setColumns(5);
jTextField1.setText("8222");
jPanel6.add(jTextField1);

jPanel2.add(jPanel6);

jPanel7.setToolTipText("Port that the Video stream is on (default 1200).");
jPanel7.setMaximumSize(new java.awt.Dimension(226, 27));
jPanel7.setMinimumSize(new java.awt.Dimension(226, 27));
jPanel7.setPreferredSize(new java.awt.Dimension(226, 27));
jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel5.setText("Video");
jLabel5.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);
jLabel5.setMaximumSize(new java.awt.Dimension(50, 15));
jLabel5.setMinimumSize(new java.awt.Dimension(50, 15));
jLabel5.setPreferredSize(new java.awt.Dimension(50, 15));
jPanel7.add(jLabel5);

jTextField2.setColumns(5);
jTextField2.setText("1200");
jPanel7.add(jTextField2);

```

```

jPanel2.add(jPanel7);

add(jPanel2);

jPanel3.setMaximumSize(new java.awt.Dimension(260, 40));
jPanel3.setMinimumSize(new java.awt.Dimension(260, 40));
jPanel3.setPreferredSize(new java.awt.Dimension(260, 40));
jButton1.setText("Connect");
jPanel3.add(jButton1);

add(jPanel3);

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

/** Closes the dialog */
private void closeDialog(java.awt.event.WindowEvent evt) { // GEN-FIRST: event_closeDialog
    setVisible(false);
    dispose();
} // GEN-LAST: event_closeDialog

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new OpenDesign(new java.awt.Frame(), true).setVisible(true);
        }
    });
}

// Variables declaration - do not modify // GEN-BEGIN: variables
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel6;
private javax.swing.JPanel jPanel7;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
// End of variables declaration // GEN-END: variables

private MyActionListener actionListener = new MyActionListener();
private TCPconnection control;

private class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == jButton1)
            control.OpenBy(jComboBox1.getSelectedItem().toString());
        else if (e.getSource() == jButton2) {
            System.out.println("Getting awareness address");
            jComboBox1.removeAllItems();
        }
    }
}

```

```

        jComboBox1.addItem("192.168.0.1");
    }
}
}
}
}

```

Guifrontend

```

/*
 * GuiFrontend.java
 *
 * Created on November 29, 2005, 8:00 AM
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

```

```
package rc_client;
```

```

import javax.swing.*;
import javax.swing.border.*;
import javax.imageio.ImageIO;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.net.URL;
//import javax.media.*;

```

```

/**
 * The Front End to this Program
 * @author Micah Akin
 */

```

```

public class GuiFrontend extends JFrame {
    // Globally needed
    private JFrame frame, about, open; // the frame for drawing to the screen
    private Insets insets; // frame border
    private Container c, a;
    private TCPconnection control = new TCPconnection();
    protected MyActionListener actionListener = new MyActionListener();
    private MyKeyListener keyListener = new MyKeyListener();

    // Menu Pieces
    private MenuBar mb;
    // Connect Menu
    private Menu Connect;
    private MenuItem Open, AutoOpen, Quit, OpenDebug;
    private MenuShortcut msO, msOA, msQ, msT;
    // Help Menu
    private Menu Help;
    private MenuItem About;
    //private CheckboxMenuItem WallDeath, AutoStartNextLevel;

    // About Window
    private JPanel pnl;
    private Image image;
    private JTextArea textT;
    private JTextField textA;
    private JScrollPane scrollPane;

    // Open Window
    private JFrame openW;
    private JTextField oftf;

```

```

private JButton ob;

/**
 * Creates a new instance of GuiFrontend
 * @param args Any Args being brought in from the Command Prompt
 */
public GuiFrontend(String[] args) {
    super("RC-Truck Control Client");
    setSize(640,480);
    setResizable(true);
    a = getContentPane();
    a.setLayout(new BorderLayout());

    if(mb==null)
        createMenuBar();
    setMenuBar(mb);
    addKeyListener(keyListener);
    pack();
    setVisible(true);
    insets = getInsets(); // must be after frame is rendered
}

public void createMenuBar(){
    mb = new MenuBar();

    // Connection Menu
    Connect = new Menu("Connection", true);
    {
        msO = new MenuShortcut(KeyEvent.VK_O, false);
        Open = new MenuItem("Open", msO);
        Open.addActionListener(actionListener);
        Connect.add(Open);

        OpenDebug = new MenuItem("Open Debug");
        OpenDebug.addActionListener(actionListener);
        Connect.add(OpenDebug);

        msQ = new MenuShortcut(KeyEvent.VK_Q, false);
        Quit = new MenuItem("Quit", msQ);
        Quit.addActionListener(actionListener);
        Connect.add(Quit);
    }
    mb.add(Connect);

    // Help Menu
    Help = new Menu("Help", true);
    {
        About = new MenuItem("About");
        About.addActionListener(actionListener);
        Help.add(About);
    }
    mb.add(Help);
}

private class MyWindowAdapter extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.out.println("Window closed...Exiting program!");
    }
}

```

```

        System.exit(0);
    }
}

private class MyActionListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        if((e.getSource() == Quit))
            System.exit(0);
        else if (e.getSource() == OpenDebug) {
            control.openDebug();
        }
        else if (e.getSource() == Open){
            Open openWindow = new Open(new java.awt.Frame(), true, control);
            openWindow.setVisible(true);
        }
        else if (e.getSource() == About) {
            About aboutWindow = new About(new java.awt.Frame(), true);
            aboutWindow.setVisible(true);
        }
    }
}

public class MyKeyListener extends KeyAdapter {
    public void keyReleased(KeyEvent r) {
        int c = r.getKeyCode();
        if (c == KeyEvent.VK_SHIFT)
            control.write("close");
        else if (c == KeyEvent.VK_K || c == KeyEvent.VK_J || c == KeyEvent.VK_LEFT ||
            c == KeyEvent.VK_H || c == KeyEvent.VK_L ||
            c == KeyEvent.VK_SEMICOLON || c == KeyEvent.VK_RIGHT ||
            c == KeyEvent.VK_QUOTE)
            control.write("SV1M125");
        else if (c == KeyEvent.VK_A || c == KeyEvent.VK_UP)
            control.write("SV3M0");
        else if (c == KeyEvent.VK_Z || c == KeyEvent.VK_DOWN)
            control.write("SV3M0");
        else
            System.out.println(KeyEvent.getKeyText(c));
    }

    public void keyPressed(KeyEvent e) {
        int c = e.getKeyCode();
        if (c == KeyEvent.KEY_RELEASED || c == KeyEvent.VK_R)
            control.write("relax");
        else if (c == KeyEvent.VK_C)
            control.write("center");
        else if (c == KeyEvent.VK_K)
            control.write("SV1M100");
        else if (c == KeyEvent.VK_J || c == KeyEvent.VK_LEFT)
            control.write("SV1M75D");
        else if (c == KeyEvent.VK_H)
            control.write("SV1M50D");
        else if (c == KeyEvent.VK_L)
            control.write("SV1M150");
        else if (c == KeyEvent.VK_SEMICOLON || c == KeyEvent.VK_RIGHT)
            control.write("SV1M175");
        else if (c == KeyEvent.VK_QUOTE)
            control.write("SV1M200");
        else if (c == KeyEvent.VK_A || c == KeyEvent.VK_UP)
            control.write("SV3M136");
        else if (c == KeyEvent.VK_Z || c == KeyEvent.VK_DOWN)

```

```

        control.write("SV3M96");
    else if (c == KeyEvent.VK_SHIFT)
        control.write("open");
    else
        System.out.println(KeyEvent.getKeyText(c));
    }
}

```

```

}

```

Network

```

/*
 * Lookup.java
 *
 * Created on November 9, 2006, 7:07 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package rc_client;
import java.lang.*;
import java.net.*;
import java.io.*;

/**
 *
 * @author la_mano
 */
public class Lookup {
    TCPconnection webServer = new TCPconnection();

    public Lookup() { }

    public String getCache(String IP) {
        webServer.OpenBy(IP, "80");
        webServer.write("GET /index.php HTTP/1.0\n");
        String chomp = "";
        while(true) {
            String temp = webServer.readln();
            if(temp == null)
                break;
            chomp += temp + " ";
        }
        String[] chompA = chomp.split("::");
        String output = chompA[1];
        webServer.Close();
        return output.replaceAll("_", " ");
    }

    public boolean setCache(String IP, String str) {
        webServer.OpenBy(IP, "80");
        webServer.write("GET /index.php?=" + str.replaceAll(" ", "_") + " HTTP/1.0\n");
        webServer.Close();
        return (str.replaceAll(" ", "_")==getCache(IP));
    }

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

```

```

        Lookup test = new Lookup();
        System.out.println(test.getCache("131.122.62.16"));
        System.out.println(test.setCache("131.122.62.16", "HelloWorld"));
        System.out.println(test.getCache("131.122.62.16"));
    }

}

/*
 * Control.java
 *
 * Created on November 9, 2006, 7:07 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package rc_client;
import java.lang.*;
import java.net.*;
import java.io.*;

/**
 *
 * @author la_mano
 */
public class TCPconnection {
    private InetAddress address;
    private int port = 80;
    private Socket sock = null;
    private BufferedReader in = null;
    private FilterInputStream istream = null;
    private OutputStream out = null;
    private FilterOutputStream ostream = null;

    /** Creates a new instance of Control */
    public TCPconnection() {
        ostream = new FilterOutputStream(System.out);
    }

    private boolean open() {
        try {
            sock = new Socket();
            sock.setSoTimeout(1000);
            sock.connect(new InetSocketAddress(address.getHostAddress(), port), 1000);
            out = sock.getOutputStream();
            ostream = new FilterOutputStream(out);
            in = new BufferedReader(new InputStreamReader(sock.getInputStream()));
        }
        catch (IOException e) {
            System.out.println(e.toString());
            return false;
        }

        return true;
    }

    public boolean OpenBy(String addr) {
        port = 8222;
        try{

```

```

        address = java.net.InetAddress.getByName(addr);
    }
    catch (UnknownHostException e) {
        System.out.println(e.toString());
        return false;
    }
    return open();
}

public boolean openDebug() {
    ostream = System.out;
    return true;
}

public boolean OpenBy(String addr, String ctrlP) {
    port = Integer.parseInt(ctrlP);
    try{
        address = java.net.InetAddress.getByName(addr);
    }
    catch (UnknownHostException e) {
        System.out.println(e.toString());
        return false;
    }
    return open();
}

public boolean Close() {
    try{
        if(ostream != null)
            ostream.close();
        if(out != null)
            out.close();
        if(in != null)
            in.close();
        if(sock != null)
            sock.close();
    }
    catch (IOException e) {
        System.out.println(e.toString());
        return false;
    }
    return true;
}

public boolean state() {
    return sock.isConnected();
}

public boolean write(String cmd) {
    cmd += "\n";
    try{
        ostream.write(cmd.getBytes());
        ostream.flush();
    }
    catch (IOException e) {
        System.out.println(e.toString());
        return false;
    }
    return true;
}

```

```
public String readln() {
    String output = null;
    try{
        output = in.readLine();
    }
    catch (IOException e) {
        System.out.println(e.toString());
        return null;
    }
    return output;
}
}
```