

REMOTE ROBOTIC CONTROL VIA INTERNET PROTOCOL

NASA Undergraduate Space Grant Fellowship

Jason Akagi
Department of Electrical Engineering
University of Hawaii at Manoa
Honolulu, HI 96822

ABSTRACT

The robotic remote control system is composed of 3 major sections: design and fabrication of the robot; communication between PCs and microprocessors; design and implementation of remote control user interface. The robotic hardware includes a Z-World RabbitCore 2000 microprocessor, a motor driving circuit, RF transmitter and receiver circuits, antennas, power regulation board, infrared sensor board, and chassis. Serial communication is used between the PC server and microprocessor. TCP/IP communication is used between the server and remote client. A graphical user interface was written for use on POSIX operating systems; it allows users full control over the target robot from any computer with internet access. Potential uses of this technology include ground-station control for unmanned missions in space exploration and robot and vehicle operation.

INTRODUCTION

In space exploration, NASA has expressed great interest in using rovers to explore the terrain of Mars. The Mars Exploration Rovers are the two rovers scheduled for a mission in 2003. Rovers will be controlled directly from Earth and will be used to gather and study soil samples. There appears to be great promise in the new generation of rovers in space. According to NASA, "Future rovers will maneuver to precision rendezvous with other surface vehicles such as the carriers. Eventually teams of robotic rovers will work together to build an infrastructure of robotic colonies, laying the groundwork for human visits and human bases [1]." Other schools interested in this technology are Santa Clara University and Tokyo Institute of Technology. Undergraduate students at the University of Hawaii of Manoa have been building autonomous robots for several years now for a competition called Micromouse. A Micromouse is a small robot that autonomously navigates to the center of a randomly constructed 16 x 16 cell maze. Walls are the only obstacles in the maze.

results in great precision for 90 degree turns. Building the robotic system was a time consuming process because precision was so important. The motor driving system uses a power-transistor circuit controlled by outputs from the robot's computer. The digital outputs have a timing sequence written in software to send pulses through the bipolar transistor circuit and drive the motors.

HARDWARE -- WIRELESS REMOTE CONTROL

A wireless remote control circuit is used so that the Micromouse robot wouldn't be constantly dragging a serial cable while running through the maze. The circuit was easy to build and the parts were from a kit bought from Reynolds Electronics. The transmitter and receiver operate at UHF at 433.92 MHz and are used for short range remote control systems [2]. The remote control modules can operate as far apart as 200 feet indoors. All I need to do is wirelessly connect the robot processor with a nearby transmitter; even a 30 feet range would probably be sufficient for my purposes. However, I would like to eventually experiment with a more powerful antenna and transceiver so that remote control can be done over much longer distances.

SOFTWARE DESIGN -- AUTONOMOUS TRACKING SOFTWARE

The software design consists of autonomous tracking for the robot, RS232 serial communication, and a graphical user interface written for Linux. The first programming goal for this project was to develop the tracking code to control the motion of the robot. Micromouse is designed to be a completely autonomous robot, but the purpose of manipulating it is because it is often useful to have manual control over its movement during testing. In order to prevent the robot from hitting walls in the maze, infrared sensors have been positioned to determine the location of maze walls. Depending on the position of walls, the robot avoids obstacles by making minute adjustments to its path. Turns are taken at 90 degrees at the intersections of the maze. Autonomous tracking makes the user interface much easier to develop. Rather than controlling each step of the motors, a command is used to tell the robot which direction should be taken. The command is processed onboard the robot's computer and a move is made to the next cell in the desired direction (North, East, South, West). Redundancy for collision avoidance was implemented in case there are inconsistencies with the maze path. If the robot is about

The objective of this project is to remotely control a robot at another school over the internet. My goal was to be able to move the robot through the maze using manual control and to be able to type in target coordinates and have the robot move to the new position by trying to traverse the shortest path.

HARDWARE DESIGN -- OVERVIEW OF REMOTE CONTROL SYSTEM

The following diagram depicts the hardware modules for the remote control system.

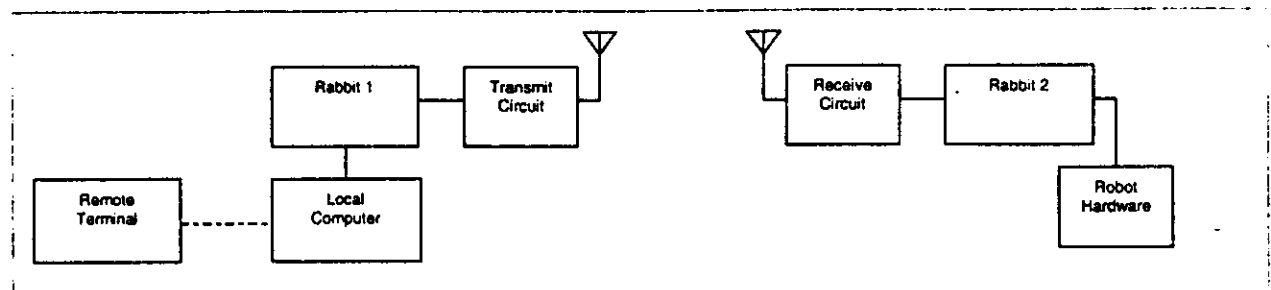


Figure 1: Diagram of remote control system modules

Starting from the left side of the diagram is a remote computer running a UNIX terminal application. It is connected to a local computer server by remote login. The local machine then sends serial data to a microprocessor, called Rabbit 1. Using a wireless transmitter, data is transmitted to a receiver circuit. The signal is decoded and sent to a second processor called Rabbit 2. Finally, the robot's microprocessor instructs the robot to move. As you can see, the robot is just a small piece of the entire system.

HARDWARE -- ROBOTICS

The hardware implementation involved designing and building robotic equipment, wireless remote control circuits, serial communication test circuitry, and setting up a secure Linux server to connect to the robot. I decided to use this year's Micromouse as the remote controlled robot. The reason for this choice is that I wrote the autonomous-tracking software for the robot and I thought it would be interesting to see how the robot operates under remote control. Aluminum sidings were used to build the chassis, which housed motors that were purchased from Eastern Air Devices. The design of the frame allows the robot to move backward as easily as it can move forward. Also, by centering the motors, the robot can pivot around a single point. This

crash, it can readjust itself to get back on course. This cannot correct all situations but it does improve the reliability of tracking.

SOFTWARE -- LINUX SSH SERVER / UNIX CLIENT

The purpose of this project is to be able remotely control a robot or any similar type of electronic device via the internet. Transfer Control Protocol/ Internet Protocol (TCP/IP) enable computers to network with others from remote locations via the internet regardless of operating system or software. I decided upon a Linux operating system rather than MacOS or Windows systems because I wanted a programming friendly interface. Initially, I thought that TCP/IP socket programming would be necessary to accomplish this. But I was able to find a simple alternative solution without socket programming experience. In order to provide a secure connection between computers over the internet, a Linux SSH server was configured with a Mandrake Linux 8.2 distribution. The server allowed remote SSH login from any internet-enabled computer via telnet or SSH. Connections are initiated by login with a username and password in a UNIX terminal window. The user interface program is run off of the Linux server once a user signs in.

SOFTWARE - SERIAL COMMUNICATION

RS232 serial protocol was used for a local desktop computer to send commands to the robot microprocessor. Dynamic-C which is used to program the RCM2000 microprocessor has a library with several serial functions. This made it very easy to write the driver for the RCM2000. On the server side, I decided to work with the Linux operating system to communicate with my microprocessor because all computer devices are treated as files. Therefore, writing to an attached serial device is like writing to any other computer file. This allows command signals to be sent to the microprocessor. The serial port can be directly controlled from the command line or from an executable application. Below are two examples of writing to the serial port of a Linux machine.

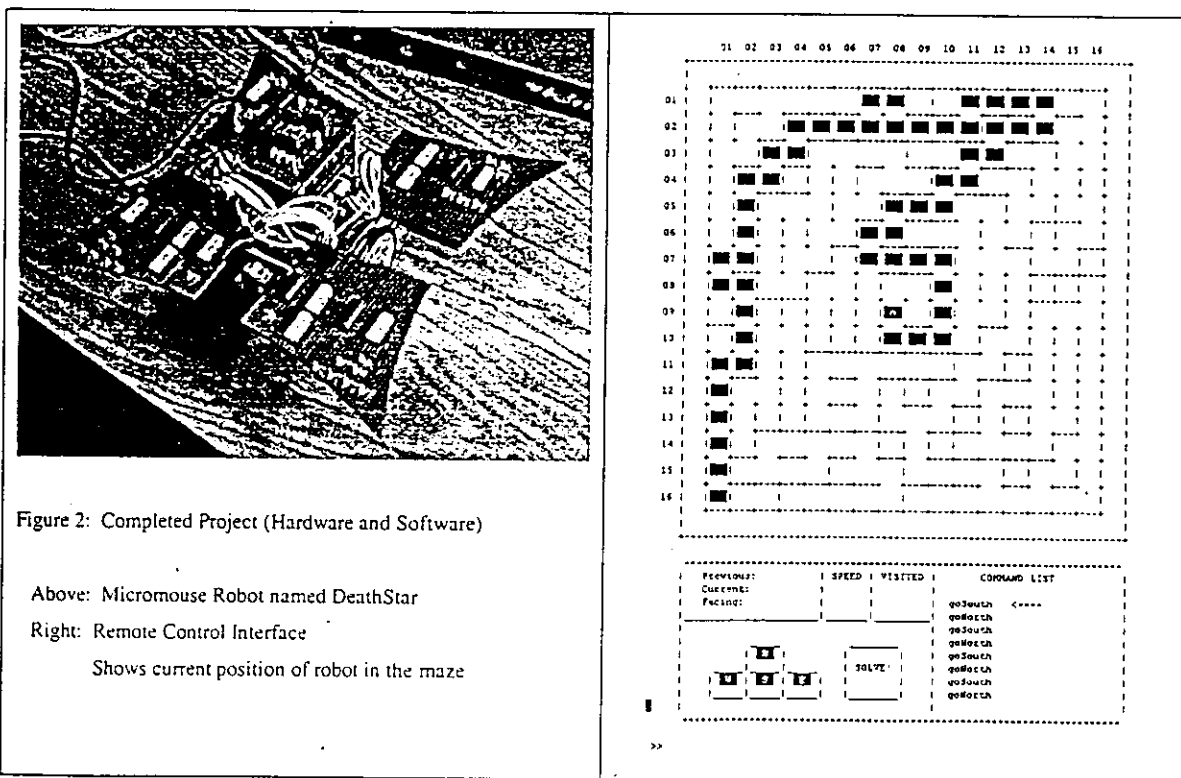
Command-line echo "hello serial port" > /dev/ttyS0

Application fd = open ("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
 fcntl (fd, F_SETFL, 0);
 write (fd, "hello serial port", 18);

It took a lot more time to program the Linux end because I had a little trouble with the serial communication settings. Some research was required to get me started for the Linux serial interface. After writing the programs to control the serial ports of the microprocessor and the Linux computer, I attached a PC laptop running Windows 2000 for testing. I ran the HyperTerminal application and configured the serial settings to match with the settings of the other processors. Baud rate was set to 19200 kb/s and character size was 8 data bits with no parity and a single stop bit (19200 baud, 8N1). Using HyperTerminal, I was able to send serial data and watch the response from the microprocessor and Linux computer in the transmit/receive window.

SOFTWARE – REMOTE CONTROL USER INTERFACE

The graphical user interface for controlling the Micromouse robot was one of the most challenging parts of this project. The application was written in the C programming language and was targeted for a Linux operating system. However, I was able to port the program to other POSIX systems such as MacOS X, and HP-UX on Wiliki. The figure below shows the Micromouse robot and the remote control program.



When the user executes the program, single keystrokes can make the robot move West ('w'), South ('k'), East ('l'), and North ('I'). The 'q' key ends the program and the 's' key will automatically make the robot solve the maze using its onboard solving code. The 'c' key will cause the program to prompt integer values of target coordinates in the form 'x y'. This command (which can be abbreviated by typing the 'a' key) will make the robot move to the cell at column x, and row y. As the robot moves through the actual maze, the program will display the current cell that the robot is in.

Another feature of the remote control interface is that it can also display the current speed of the Micromouse. The speed can be changed by pressing the number keys from '1' to '5', where '5' is the fastest. Also a history of commands list is displayed in the program. This program is very easy to use and serves its purpose well.

CONCLUSION

Designing and building the remote robotic control system was a comprehensive engineering project that taught me many new things about programming and working with electronic hardware. The programs that were written for this project can be used for other space related applications. I think that the experience would be very useful for other projects at the University of Hawaii campus such as the Cubesat program. I am currently working with the Data and Command Handling team. We will design the software for a satellite ground-station, which is similar to remote control of a robot over the internet. Rather than using the internet as communication lines, we will be using the satellite's antenna and a transceiver. This was a challenging project, but I was able to attain my goal of constructing a system that can receive commands over the internet. Maybe some day this will be used on a larger scale such as remote control of Mars rovers.

ACKNOWLEDGMENTS

I would like to thank the Hawaii NASA Space Grant Consortium for funding my project and giving me the opportunity to conduct this research. I would also like to thank Dr. Wayne Shiroma for serving as an advisor for this research project, and Dr. Tep Dobry for helping me in the design process.

References

- [1] <http://mars.jpl.nasa.gov/technology/rovers/>
- [2] <http://www.rentron.com/remote.htm>