Internet-based Control for the Intelligent Unmanned Ground Vehicle: Bearcat Cub

Masoud Ghaffari, Sugan Narayanan, Balaji Sethuramasamyraja, and Ernest L. Hall

Center for Robotics Research
University of Cincinnati
Cincinnati, Oh 45221-0072

ABSTRACT

Secure remote access with inter-operatability for operating a robot can be successfully achieved using the web services provided in the .NET framework. The complete design of the machine discussed in this paper is made on the .NET framework. The server which operates the robot is configured to IIS. The algorithm for obstacle detection is coded on a different server using the .NET framework. By using web services, the robot can be accessed by other servers. These web services are consumed by the server on which the robot executes. A proxy is created on this server. The whole control is given in the form of a series of web pages which can be accessed by any web browser. However in order to input parameters and control the robot, authentication is required. The user provides authentication credentials which are matched with the existing information on the data base. After authentication, the user proceeds further to control the robot. The security and reliability of remote access is provided by the components that come with the web services namely, SOAP, WSDL and Proxy.

Keywords: Robotics, .NET, remote control, tele-robotics, Internet-based control, unmanned ground vehicle

1. INTRODUCTION

Recent progress in Internet capabilities has made it easier to use as a reliable and widely accessible communication framework. Remote control via the Internet is a very young field of research that could have significant applications in the near future. Robotics, manufacturing, traffic control, space exploration, health care, disaster rescue, house cleaning, security, inspection and tele-presence are examples of such applications.

Since the first networked device, "Cambridge coffeepot," appeared on the Internet, a rapid enlargement of the WWW over the past several years has resulted in a growing number of tele-robotics sites and Web accessible devices. [1,2] devices.

Previous researchers have had different approaches to accessibility from the Internet. Availability for public users has been a goal for most projects, but others have focused on special user devices. For example, by 1995, Goldberg et al. had developed a tele-robotic system called the Tele-Garden by which WWW users are able to observe, plant and nurture life in a remote garden.[3] Likewise, Peterson, et al. developed a system for tele-pathology by the Internet. This system allows any Internet user to become a consultant for tele-pathology without the acquisition of specialized hardware or software.[4]

Rovetta, et al. used a mix of communication media for performing tele-surgery in 1995. Their work was based on a special user access and not a public access Internet.[5] Various other devices have become available over time, such as the Programmable Logic Control for a chemical experiment[6], Microscope [4,7,8], Blimp Space Browser[2], Nuclear Microprobe[9] and Web Camera.[10] In fact, Web cameras are the most common Internet connected devices.[1]

The merge of the Internet and manufacturing technologies has resulted in bridging of the gap between engineering technology (such as a rapid prototyping hardware system) and information systems to enable the remote control of engineering resources.[11,12] Wang et al. presented the concept of an Internet assisted manufacturing system for agile manufacturing practice.[13] In this system, a local user is able to introduce design specifications to a product information

system and the Central Network Server can generate complete CAD/CAPP/CAM/CAA files and control the remote FMS or CNC machines to accomplish the whole production process.

Tele-robotics is also an active branch in Internet connected devices. Schiling developed an educational inspection mobile robot for tele-diagnosis of malfunctions, tele-maintenance of machines, and tele-monitoring of remote sites by sensors and tele-operations of remote equipment, including robots.[15] In another experience, Winfield et al. developed a system that can control several robots from a Local Area Network simultaneously.[16]

All these systems are not yet commercially available. In fact, the limitation of bandwidth, safety, harmonizing the remote activities and time delays are the major concern of research in this area. In many situations, a human is needed to control these machines. In the existing set up, there are very few tools that offer a remote access to the robot, and its scope is also limited. Compatibility across platforms is not achievable. Timbuktu [TM], one of the tools used currently, also necessitates the need for software to be installed at both locations.

The purpose of this paper is to describe the design and development of an interface for remote control of the Bearcat III and Bearcat Cub robots via the Internet. Bearcat III and Bearcat Cub are two unmanned, intelligent, ground vehicles that have been developed in the Center for Robotics Research at the University of Cincinnati.

2. PROJECT DESIGN

In order to access the robot remotely with a secure connection that is platform- and device-independent, the .NET framework provides an effective and ideal solution by using the concept of web services.[17,18] The applications of web services provide a safe and secure connection with the robot at one end and the process or operation at the other end of the connection. The framework also does not confine itself to computers and makes the whole operation inter compatible across devices. Thus, using the .NET framework in ASP.NET, the code for operation of a robot can be programmed on a different server, which can be accessed using the web services protocols using a web interface to remotely access it.

Remote access of Bearcat Robot III was performed using web services in the .NET framework. The pages were designed using ASP. NET with SQL server as the back end. Web services were then consumed from a different server into this server. The pages were integrated together and were hosted on the web.

2.1 Web Server

A web server is a software tool which manages (hosts) web pages and makes them available to browsers, either through a local network or through the Internet. Physically web servers and the client machines can be on same machine or separated miles apart. However this does not make any difference in terms of access. There are many web servers available in the market today. Apache, IIS (Internet Information Services), Enterprise Server by IPlanet are a few. ASP. NET runs on IIS. In this case, the Web server is located in the Robotics Lab, located in Room 551 Baldwin Hall, University of Cincinnati, OH. It can be accessed by any machine existing on the UC (University Of Cincinnati, OH) network. Since IP masquerading is done, the IP address of a machine is different outside UC's network. Hence, the page transition address would vary.

2.2 Dynamic Web Pages

Dynamic web pages are those which keep changing based on time or user or other changes on client side. Static pages are those in which, the contents always remain the same. HTML (Hyper Text Markup Language) code is typed directly into an editor and saved as a HTML file. In dynamic pages, however, contents are more dynamic and personalized in nature. There are two ways in which dynamic pages can be created- client side and server side.

2.2.1 Client Side Dynamic Pages

In the client side model, the web pages are created by the browser to which the modules are attached. The HTML code is sent to the browser along with a separate file which has a set of instructions which is referenced by the HTML page. However these instructions can also be found embedded in the code. These are used by the browser to generate HTML page. Hence here it is dynamically done on request. The HTML page thus generated is sent back to browser. So, when a user types a URL (Uniform Resource Locator), the web server looks up the URL and locates the HTML page corresponding to it. In some cases, the web server also looks up for the instructions file associated with the HTML. The web server sends the HTML and the instructions back to the browser across the network. The module within the browser

then processes the instructions and then returns it as HTML within the .HTML page. There is only one page that will be returned to the browser. The HTML is then returned to the browser, where the page is displayed. This is shown in Fig 1.

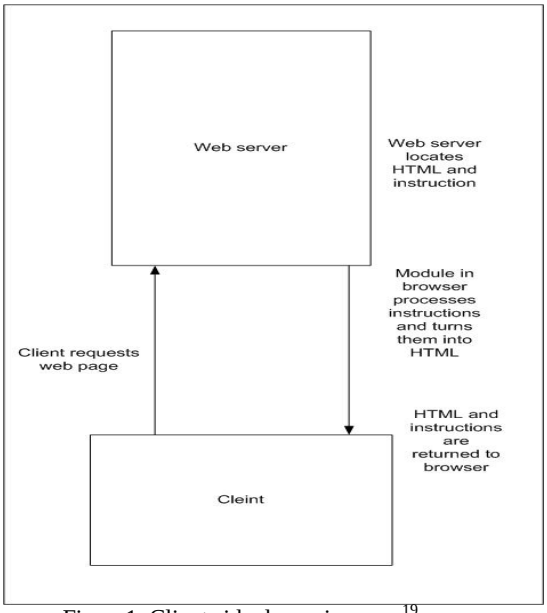

Figure1. Client side dynamic pages[19]

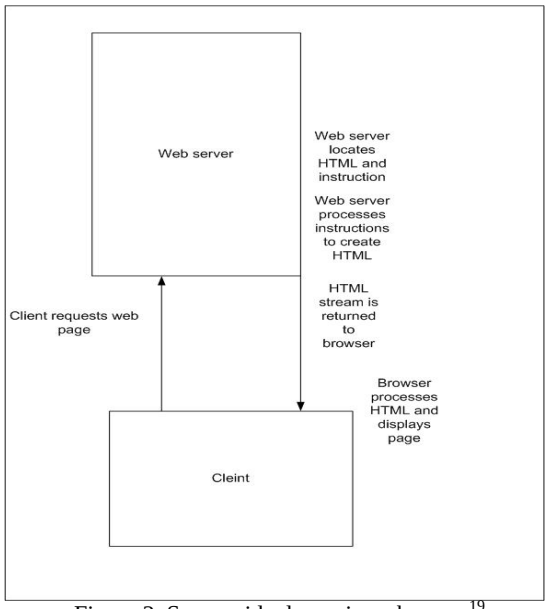

Figure 2. Server side dynamic web pages[19]

2.2.2 Server Side Dynamic Pages

Client side technologies take a long time to download, especially with a separate file of instructions. Also since this interpretation is done by the browser, each browser may interpret it in a different way. Another disadvantage is, since it is a coding issue on the client side, it is completely undesirable.

With server side coding, the HTML source is set to web server where all the processing is done, before the page is sent back to the browser. Here, the page is logically hidden on the server and most browsers can read it in a consistent way. This is shown in Fig. 2.

2.3 Front End

The pages are created using VB.NET in ASP. NET. Since it is server side scripting, the code is compiled on the IIS, the web server for .NET. The VB code and the ASP code are embedded in a single page. Before the actual coding, all the required name spaces and libraries required for the code are called in using an import statement. The code is written in a modular approach with several sub routines, which will be called by the objects as required. The modules start with sub followed by the name of the module. The appropriate event handlers are also called in the module by creating an instance. These pages are referred as web forms in .NET terminology. All web forms have the extension of .aspx. Thus all web forms which are compiled by the IIS, are web pages hosted by that IIS. Since it is a server side scripting, the web forms can be seen across the browsers consistently.

2.4 Back End

Microsoft SQL server is used as the back end where the tables are created on the data base. There are two tables, one for registered user information and other for existing inventory. The structures of the field and data type are provided in the table design. An authenticated access is provided to the database. A secured connection is established with the data base through the ODBC provided by .NET framework. All the information given when a new user signs up is captured in this data base. Whenever the user logs in, the credentials are authenticated with the server and ensure further progress to the other pages. The users are managed using the enterprise manager in SQL server.

2.5 Data providers

With the .NET framework, there are two data providers[20]: 1. OLEDB data provider 2. SQL data provider. The OLEDB provider is for connecting to any OLEDB compatible data store like Access, Excel, Microsoft SQL server, Oracle etc.

The SQL client data provider is specifically for Microsoft SQL server. The SQL client data provider provides much faster access to SQL server than the OLEDB equivalent. Each data provider has its own connection object class. The OLEDB data provider has an oledbconnection class and SQL client data provider has sqlconnection class. These classes are found in System.data.OLEDB. and System.data.sqlclient name spaces respectively. So depending on the data base, the appropriate name space should be chosen.

2.3 Data Grid

A data grid is very helpful in displaying data from the database in a tabular format. However, to display the information, data grid needs to be bound with the database. The data grid emits the HTML needed for data bound HTML tables. To do this, first a data reader with some database is obtained. Once the data is in the data reader, the data grid requires few lines of code to be bound with it. These lines of code indicate where the data should come from, which is commonly referred as the data source. The data binding property should also be notified here. One of the main advantages here is that the code involved is very simple and also very efficient. Essentially, a connection is created with the database and the connection is opened. Then the data grid's data source property is set to the required data reader. Finally the binding takes place using `databind` method. A significant feature in this approach is that, it isolates the code completely from the content. Unlike conventional ASP, there is no mixing of HTML table and data reader output syntax. For the given case, it is assumed that the inventory table resides in the same database as the user information table. Hence the binding is with the same server and the data source is from the same database.

2.4 Algorithm for Obstacle Detection

All authenticated users will be able to perform the obstacle detection test. Assuming for three types of obstacles-circular, linear, rectangular, the algorithm was devised. Since the intent is to demonstrate web services, all the functionality of the existing system are not incorporated in the algorithm.

In a given two-dimensional system, the robot is assumed to move for the pre specified points. The user specifies the start and end co-ordinates. The position (x, y co-ordinates) and type of obstacle is also specified.

- **Circular obstacle**: For a circular obstacle, the system checks if the path of the robot will interject the circle. This can be checked by knowing the co-ordinates of center of circle and the radius of the circle. There is an incremental loop which progressively checks if the co-ordinates of robot along its path to the destination will collide within the spread of circle. The position of robot is incremented every time within the loop.

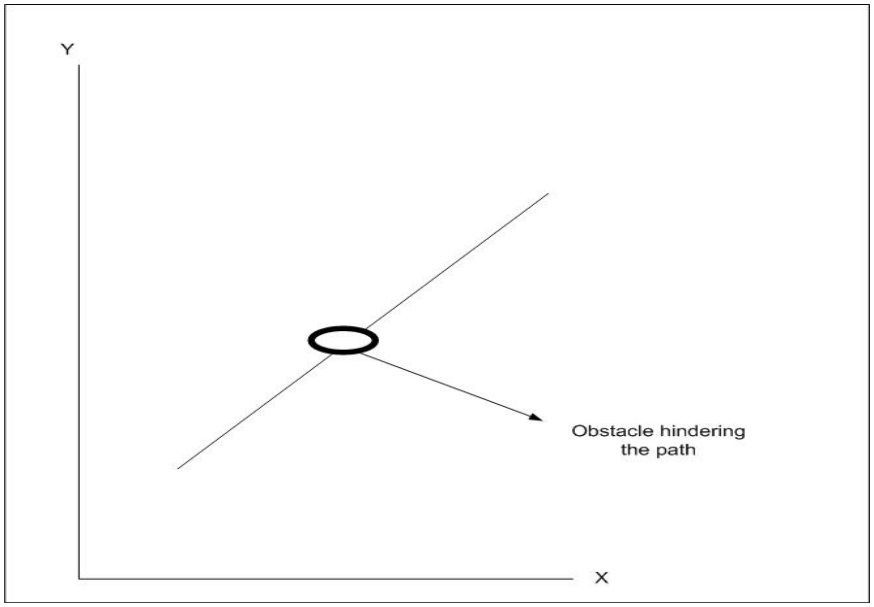

Figure 3. Circular obstacle

- **Linear obstacle**: For a linear obstacle, the system checks based on position and length of the line. The incremental loop increments the co-ordinate positions of the robot until it reaches the final destination. On its

path, if it encounters any co-ordinate of the obstacle (co-ordinates from beginning to end of obstacle), then it prompts that an obstacle has been encountered.

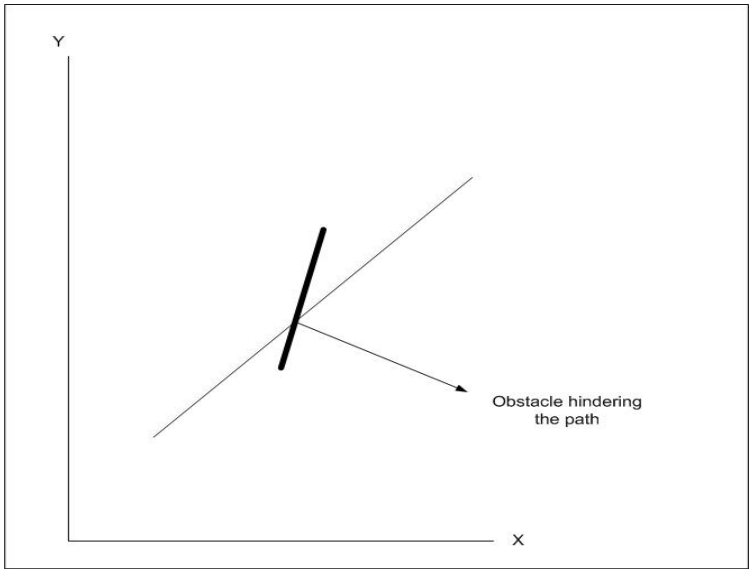

Figure 4. Linear obstacle

- **Rectangular obstacle**: For a rectangular obstacle, it is very similar to a linear obstacle. However here co-ordinates for all the four corners of the rectangle will be taken into account. As the position of robot gets incremented using the incremental loop, it progressively keeps checking for the obstacle. In this case, the whole enclosed area between the four points is checked for and when the path is obstructed by any of the co-ordinates inside, it prompts that the obstacle has occurred.

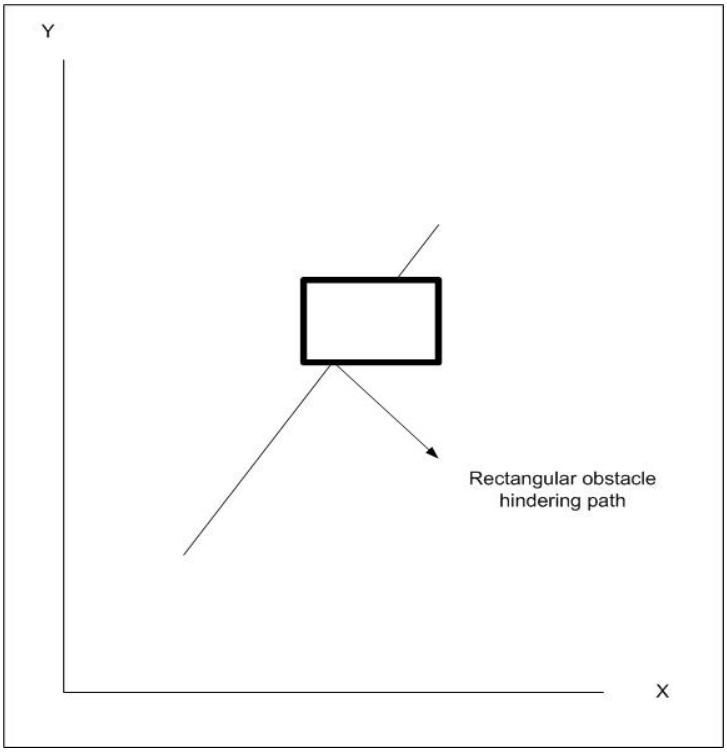

Figure 5. Rectangular obstacle

2.5 Web Services

A web service, as mentioned before, is extremely useful in remotely accessing by a secured means. The actual algorithm for obstacle detection is on a different server. The server on which robot is running, remotely accesses that server. By this, the code is not exposed to the user. Only the required privileges are given and the appropriate parameters are passed. Also the web services can be consumed by several users and hence it is not restricted to a single robot. Web services are quite different from web forms. Web services have the extension .asmx. This extension tells IIS to use aspnet_isapi.dll and lets ISAPI filter knows that a web service is going to be defined. Unlike the web forms, web services do not have embedded HTML in it. The coding is similar to object oriented programming (OOP) model, where different levels of access rights (public, private, protected), concepts of inheritance are specified.

A web service definition contains four essential parts

- Procession directive
- Name spaces
- Public class
- Web-callable methods

2.5.1 Processing directive

The processing directive appears on top of ASP. NET source file indicating settings or constraints that should be applied to any object generated from that file. It tells the compiler the language it was written and the name of the class defined.

2.5.2 Namespaces

Similar to .aspx files, it is possible to import the required logic or commands from other files existing in the libraries or name spaces. This can be seen in the code as
Imports System.Web.Services

It is required to import these name spaces in minimum, as it contains all the necessary classes for web services to handle network connection issues and other OS related tasks.

2.5.3 Public Class

Public class acts as the container for the methods of the web service. The name of this class is the name of the web service and should therefore correspond to the class value specified in the processing directive.

Public Class circleobstacle
..
End Class

The object classes defined are those which are going to be exposed over the web. This will help to make remote method calls over the Internet to the server, which will look like method calls from same machine.

2.5.4 Web Methods

Methods that are exposed over the web are web (callable) methods. A web service will expose one or more methods. There can be methods which need not be exposed to consumption. They do not have the web method declaration before the name of the function. Hence, "<WebMethod()> Public Function circleobstacle(ByVal r As Integer, ByVal c1 As Integer, ByVal c2 As Integer, ByVal x1 As Integer, ByVal y1 As Integer, ByVal x2 As Integer, ByVal y2 As Integer)" as string as seen on the code, are web callable methods. The parameters for web methods can be customized. For example web methods can be customized to hold the cached results for a particular period. So when the consumer is requesting a web service, the results that were cached before will be returned.
For the given problem, three web services are designed; one for each type of obstacle. In each web service, the web methods comprise of the algorithm which pass on the parameters for the function when consumed.
The web services can be accessed over the internet similar to the web forms. By typing in the address of the web service (which ends with .asmx), it fetches the web service on the browser by taking a HTTP request. In the .asmx page, on clicking WSDL, it expands to list the WSDL functionalities associated with that web service.
Values for the required parameters are provided and when they are invoke the web services, a request is made to the web service to execute the web service method with the value entered being passed. In the web service, based on the

algorithm provided, for the given values, the results are obtained on the browser. However these results are in XML format, which can be consumed. On the results page, there are a few name spaces which make reference to SOAP. The <types> element defines the data type that web service expects to receive and return after completion.
Now, having created the web services, the functionality could be used in the web forms which appear in the browser as web pages. Hence the web services need to be consumed for this purpose.

2.6 Schematic Layout and Step-by-Step Operation

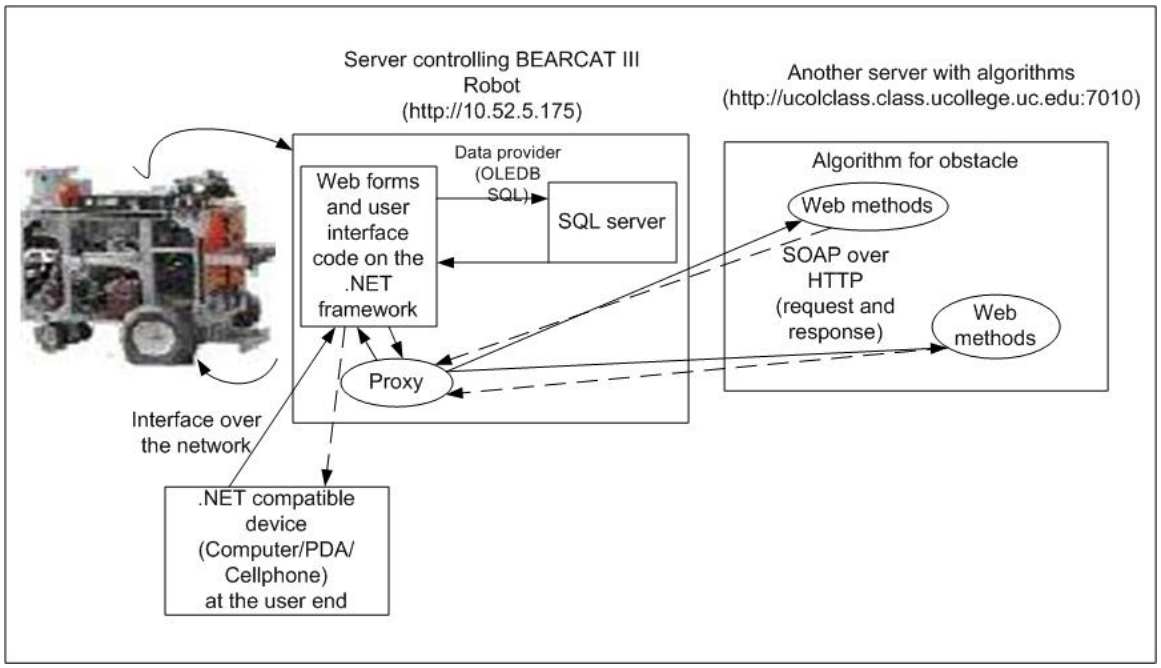

Figure 6. Schematic layout of operation

The request the user sends is given in solid arrows and the result sent back shown in broken arrows.

1. The user with a .NET compatible device, logs on to the BEARCAT III server using his/her credentials through the web interface.
2. The information provided is checked with the existing data stored in the SQL server. The connection between them is made through the data provider (OLEDB SQL connection). If the user is new, an option for signing up is provided. The users' information is stored on the SQL server.
3. On correct authentication, the user proceeds to the page where the data grid on the same server (http://10.52.5.175) is displayed.
4. The existing BRAT code is also called from the remote location.
5. On moving further, the web services are invoked. The .NET framework interacts with the proxy created on the BEARCAT III server (10.52.5.175). This was created by consuming the web services on the other server (http://10.85.2.251).
6. The proxy interacts with the web callable methods on the remote server. Parameters are sent back and forth via SOAP over HTTP.
7. The remote server (http://10.85.2.251) uses the parameters provided by the user to the proxy and inputs them to the algorithm for obstacle avoidance.
8. After computing the results are sent back as HTTP response in XML. This is being consumed by the proxy on the BEARCAT III server.
9. The proxy passes the results to the web forms.
10. The web forms display them to the user as a part of the application.

Thus the user finally gets the result, by using the algorithm on a remote server as a web interface in his/her device.

3. RESULTS AND CONCLUSION

The constraints and limitations relating to Timbuktu [TM] were eliminated by utilizing the full potential of the .NET framework. With Timbuktu [TM], a remote access to the desktop of the host machine was provided. The two machines were interfacing across the subnet using the IP addresses. Also communication was via TCP/IP. Timbuktu [TM] also necessitated the client and host to be on same platform. By using web services, a secure and reliable transfer of information was achieved. The web callable methods exposed only the required part of the algorithm to the client machine. By using SOAP over HTTP, the data packets were secured. Inter-operatability was also achieved by using the .NET framework, which crosses the boundaries of computers. The communicating platforms between the various devices and servers also become insignificant by incorporating the features of .NET framework.

The robot, connected to the server, was controlled by any device at a remote location located inside the network of University of Cincinnati, OH. It successfully utilized the services of the web methods designed for obstacle avoidance that was coded on a different server across the network. Alternatively, the BRAT [TM] code used on BEARCAT III was also accessible via the web interface. On installing IIS to the Bearcat Cub, a remote access to this robot via the .NET framework was also achieved. As the code is present in a centralized location, it can be called by any number of other robots connected to that server. Narayanan (2003) has explained more details of this project.[18]

REFERENCES

1. Available: http://dir.yahoo.com/Computers_and_Internet/Internet/Devices_Connected_to_the_Internet/
2. H. Hu, et al., "Internet-based robotic systems for teleoperation," *Assembly Automation*, 21, 2, pp. 143-151, 2001.
3. E. Paulos and J. Canny, "Ubiquitous tele-embodiment: applications and implications," *Int. J. Human-Computer Studies*, 46, pp. 861-877, 1997.
4. I. Petersen, et al., "Telepathology by the Internet," *Journal of Pathology*, 191, pp. 8-14, 2000.
5. S. You, et al., "A low-cost internet-based telerobotic system for access to remote laboratories," *Artificial Intelligence in Engineering*, 15, pp. 265-279, 2001.
6. O. Frederik, et al., "Remote Monitoring and Control of Electrochemical Experiments via the Internet Using "Intelligent Agent" Software," *Electroanalysis*, 11, 14, pp. 1027-1032, 1999.
7. J. Mansfield, et al., "Development of a System to Provide Full, Real-time Remote Control of a Scanning Electron Microscope across the Second Generation Internet: The Teaching SME," *Microscopy and Microanalysis*, 6, pp. 31-41, 2000.
8. C. Baur, et al., "Robotic nanomanipulation with a scanning probe microscope in a networked computing environment," *J. of Vacuum Society*, 15, 4, pp. 1577-1580, 1997.
9. C. Churms, et al., "The remote control of nuclear microprobes over the Internet," *Nuclear Instruments and Methods in Physics Research B*, 158, pp. 124-128, 1999.
10. B. Prihavec and F. Solina, "User interface for video observation over the internet," *J. of Network and Computer Applications*, 21, pp. 219-237, 1998.
11. F. Tay, et al., "Distributed rapid prototyping – a framework for Internet prototyping and manufacturing," *Integrated Manufacturing System*, 12, 6, pp. 409-415, 2001.
12. C. Wang, et al., "Implementation of remote robot manufacturing over Internet," *Computers in Industry*, 45, pp. 215-229, 2001.
13. Z. Wang, et al., "Architecture for Agile Manufacturing and Its Interface with Computer Integrated Manufacturing," *J. of Material Processing Technology*, 61, pp. 99-103, 1996.
14. T. Kesavadas and H. Subramanium, "Flexible virtual tools for programming robotic finishing operations," *Industrial Robot*, 25, 4, pp. 268-275, 1998.
15. K. Schilling, "Telediagnosis and teleinspection potential of telematic techniques," *Advances in Engineering Software*, 31, pp. 875-879, 2000.
16. Winfield, and O. Holland, "The application of wireless local area network technology to the control of mobile robots," *Microprocessors and Microsystems*, 23, pp. 597-607, 2000.
17. E.L. Hall, "Intelligent Robot Trends and Predictions for the .Net Future," *SPIE Con. Proc.*, Boston, 2001.
18. S. Narayanan, "Application of Web Services for Remote Access of Bearcat III Robot Using the .NET Framework," MS Thesis, University of Cincinnati, 2003.
19. R. Birdwell, et al., "Beginning ASP. NET using VB.NET," Wrox Press Ltd., 2001.
20. http://www.15seconds.com/issue/010530.htm