Development of Internet Based Control Architecture for an Unmanned Robotic Arm

Ryan Savio Rodrigues¹, Shilpa Choudhary².
¹Student, MSc. (Engg.), ²Asst Professor & Program Manager Centre for Embedded Systems Design, M. S. Ramaiah School of Advanced Studies, Bangalore.

Abstract
With the explosive growth of the Internet and all of its computing hardware/software technologies it is possible to conceive Robots Teleoperated over the Internet to be the next revolution in the world of Robotics. The Internet based teleoperations find great utility in applications that are dangerous, hostile, and inaccessible to humans and areas of work regardless of geographical locations. The Internet being a network infrastructure that is so matured and freely available connects people and allows access to devices across the globe. Also due to the growing awareness, technological support, data transmission protocols, ease of use and low cost the Internet has proved to be the most suitable infrastructure for worldwide access and information exchange.

The focus in this paper is to exhibit the design of open-ended control architecture for an unmanned Robotic Arm via the Internet. An attempt has been made to develop a prototype of the framework in a scalable, pluggable, generic, platform independent and interoperable manner using Web Services, the object oriented java and J2EE technologies. The system is designed to have a distributed Service centric architecture with the Web Service holding all the complex algorithms. Apart from this control architecture, the Web Service can also be consumed by systems built in other programming languages giving it the feature of being reusable and interoperable [1].

The functionalities implemented into this application allows authenticated users to exhibit physical motions of the Robotic Arm based on well defined positions at the selected speed. This motion is a result of the position interpolation conducted by the Web Service component. Once the coordinates have been registered into the system the user can also run the Robot in a continuous mode at defined speeds with the help of a well designed User Interface.

Key Words: Teleoperated, Internet, Web Services.

Abbreviations
IEAR Internet Enabled Autonomous Robot
J2EE Java2 Enterprise Edition
HTTP Hyper Text Transfer Protocol
TCP/IP Transport Control Protocol/Internet Protocol
JVM Java Virtual Machine
LAN Local Area Network
TINi Tiny Internet/Network Interface
CORBA Common Object Request Broker Architecture
JSP Java Server Pages
API Application Programming Interface
DOF Degree of Freedom
SDK Software Development Kit
MTK Dallas Semiconductor Microcontroller
toolkit
J2SE Java2 Standard Edition
CAN Controller Area Network
ROM Read Only Memory
SRAM Static Random Access Memory
IP Internet Protocol
DNS Domain Name Server
FTP File Transfer Protocol
XML Extended Markup Language
EJB Enterprise Java Bean
JMS Java Messaging Service
CGI Common Gateway Interface
SOAP Service Oriented Access Protocol
UDDI Universal Description Discovery and Integration
WSDL Web Service Definition Language

1. INTRODUCTION
1.1 Introduction to Robotic Teleoperations
New generation Robots have found applications in areas of work which are dangerous and hostile for humans. Due to this remote access, control and monitoring of Robotic systems have been an area of focus. Remote Sensing and Control in Robotics have found applications in areas like Remote Surgery in the world of medicine, Robots working in radioactive environments, undersea explorations, Space and Military. With the growing awareness, popularity and its enormous technological growth, the Internet has proved to be the future of distributed systems. It is now possible to conceive and develop distributed systems that could be controlled and monitored across the globe through the Web Browser.

1.2 Overview on the Internet
Over the years the Internet has matured leaps and bounds and is now a conceivable to think future of Tele-Robotics to be controlled and monitored over the Internet via the http protocol. With advances in microprocessor technologies, miniaturization has enabled a high degree of intelligence to be integrated into systems. This has allowed to integrate technologies like the Java JVM, the TCP/IP protocol to enable Ethernet connectivity and serial protocols to talk to serial devices.

1.3 Revolutionising Robotic Teleoperations
Due to these advances we are in a position to build distributed systems which collaborate with each other...
using various protocols like HTTP, TCP/IP, RS232, etc. This enables us to build distributed, platform independent, reusable, scalable applications that could be built once and run everywhere. With technologies like the Java JVM built into the microcontrollers and their various connectivity, it is now possible to build web applications that communicate with other Java technology components like Web Services via the http protocol which is the protocol of the web. This gives us the capabilities to build systems that control intelligent systems such as Robots through Web applications and other components like the serial communication channels.

1.4 The Control Architecture
This paper focuses on building a framework using the Java J2EE technologies used so widely in the E-commerce world to collaborate with embedded Java ingrained into intelligent microcontrollers. This paper uses the microcontroller DS80C400 board and the JoinMax Smart Arm Robotic Arm as a platform over which this framework resides and functions. This takes a very distributed and pluggable approach of building web components which are user friendly through the development of dynamic web pages. This enables the application to be deployed on a centralized http server located at one location and invoked through the Web Browser from another part of the globe. This is a thin client architecture which avoids the hassles of installing the software on each of the client machines thus giving more flexibility to the approach. Over the years the Java J2EE framework has proved to be the most mature framework in the world of E-commerce and it is this technology that gives us the ability to build a robust architecture. This paper also makes use of the technologies like WEB SERVICES and Java communication APIs that gives us the ability to develop a SECURE (through the use of Authentication), GENERIC (through the use of web services), PLATFORM INDEPENDENCE (as Java by its nature can be deployed and run on any platform), SCALABILITY (Complex algorithms and core functionality of the control system built in the form of a Web Service), PLUGGABILITY (The architecture by its design is loosely coupled which gives us the flexibility of plugging in/out components) architecture that could be used for other applications besides TELEROBOTICS over Internet.

2. LITERATURE REVIEW
2.1 Contributions
Several Contributions have been made in the field of teleoperations and remote access via the internet, using the Java programming language, J2EE enterprise framework and Web Services. Significant efforts have been made in the research of remote access of robots.

Authors Masoud Ghaffari, Sugan Narayanan, et al [2] discuss how remote access can be successfully achieved using the web service provided in the .NET framework. Also author Maki K. Habib [3] discusses an Internet based monitoring and control systems framework that would allow several users to simultaneously view and control distributed devices, developed in the Java language and accessible via the Web Browser.

Investigators Maja Matijasevic, Kimon P. Valavanis, et al [4] present a framework for multi User Distributed virtual environments which is used for teleoperation of a mobile robot over the Internet. Authors Huosheng Hu, Lixiang Yu, et al [5] have discussed a new generation networked telerobotic system that combines network technology with intelligent mobile Robots that can be controlled via the Internet.

In their research Kang Hee Lee, Soo Hyun Kim et al [6] discuss the design of an Internet based robot system which is insensitive to unpredictable Internet time delay problem in transmission. The idea is to obtain the predicted position of the robot and operate the mobile robot using direct control. The predictable position is based on the time at which the users command reaches the robot system. Also authors M. Farajmamadi, J. Gu, Max Meng et al [7] have described an Internet based wireless mobile robot architecture. The system consists of a mobile robot, a server workstation, radio frequency communication circuit and host software. The user controls the mobile robot using a user friendly interface and receives a real time feedback through a camera server.

Investigators Ken Goldberg, Billy Chen, et al [8] describe a system that allows a distributed group of users to simultaneously teleoperate an industrial robot arm via the Internet. In their paper authors Kuk-Hyun Han, Shin Kim, et al [9] describe the implementation of an internet-based personal robot with novel direct internet control architecture which is insensitive to the inherent internet time delay.

2.2 Summary of the literature review
There has been an exorbitant amount of research conducted in the area of Teleoperations of Robotic Systems via the Internet. All these approaches though well researched on and proven through an implementation path have a lot of scope for improvements.

The objective of this paper has been to build a control architecture to overcome the above mentioned gaps in the designs. The design implemented in this paper is purely a Java based design, which uses the middleware J2EE framework and is based on the Service Oriented Architecture using Web Services. The uniqueness of this design is to implement the complex algorithms into Web Services that can be consumed not only by the control system built in this paper but also be reused by multiple such systems and also systems built using other programming languages thus exhibiting the features like REUSABILITY and INTEROPERABILITY. Web Services is a fairly new and mature technology in the world of E-commerce that makes calls to remote services using SOAP requests and responses embedded into XMLs and transported via the http protocol. This makes the design have a GENERIC approach which is interoperable and hence avoids the use of technologies like CORBA. Since the complex and core algorithms are built into Web Services and the entire architecture can be broken down into independently deployable Web Applications, this gives the design a PLUGGABLE and SCALABLE
approach as additional features and capabilities can be added to this control system in the form of newly built Web Services.

With the advances in miniaturization of microprocessor technologies, network microcontrollers have higher processing and integration capabilities. These microcontrollers have the Java JVM built into it and also protocols like the TCP/IP stack, 1 wire and CAN embedded into it. This gives it the capability of installing Web Servers with Web Applications deployed, enables it to be integrated to the outside world through the Ethernet connectivity and have it configured to have its own IP address. This gives the design the feature of having the controlled process (Robot) and its controller work as an independent entity thus exhibiting the feature of LOOSE COUPLING. Thus giving the entire architecture a very DISTRIBUTED and SCALABLE approach.

3. PROBLEM DEFINITION

3.1 Problem definition

With the advent and growth of the Internet Technologies, demand has been growing ever since to have embedded systems enabled by Internet. There has been a growing requirement to have teleoperated systems like robots in areas where human reachability is posed with a definite challenge like Mines, Nuclear Reactors, Micro Surgery, Space and Deep sea explorations, Home and Industrial usages like Tele-teaching, Tele-maintenance, Tele-expertise, Tele-production, etc.

3.2 Problem Statement

To prove the concept of remotely controlling an intelligent system like a Robotic Arm via the Internet using Web Services and Java/J2EE technologies.

3.3 Main Objective

1) Build a Teleoperated Robotic System which can be controlled over the Internet.
2) Only authenticated users should be allowed to control the Robot.
3) Build a hardware infrastructure on which the software architecture designed and built could reside and operate on, that would comprise of:
   a) Robotic Arm that would have 6 DOF powered by servo motors which could be controlled by an external control system
   b) Control system that could be connected to a LAN system which would have the capability of hosting web applications and be recognised over the internet by an IP assigned to it.
4) Build the software system comprising of loosely coupled components, deployed on systems/products from various technologies (mechanical, electronics, networking, etc) that would communicate to each other using commonly known and matured protocols like RS232,Http, SOAP, etc.

3.4 Methodology adopted to meet the objective

This paper is basically a prototype for an open ended Robot control architecture designed using the Java/J2EE technology so widely used for E-commerce and the Service Oriented Architecture Web Services.

This paper is built over a hardware platform which comprises of an integration of:
1) JoinMax Robotic Arm Assembly kit
2) DSTINIM400 Network Micro Evaluation Board
3) DSTINS400 Socket Evaluation Board
4) Serial and Ethernet Connectivity
5) TINI SDK, MTK SDK, J2SE, J2EE and Java2 comm. APIs.
6) Intel Celeron Laptop computer to communicate with each other using the Http, SOAP, TCP/IP and Serial RS232 protocols.

These are all integrated to form a homogeneous environment for the software architecture designed to reside and operate. The computer hosts the web application which provides the user interfaces and authentication. It also hosts the Web Service component which does all the path interpolation based on the user inputs of position and speed.

The computer system then communicates these interpolated position information to the embedded web application deployed on the Tynamo Embedded Web Server hosted on the TINI controller board. This communication takes place across the Internet through the Ethernet connectivity and protocol inbuilt into the TINI board.

The TINI microcontroller then interprets this position information into a byte stream recognizable by the servomotor controller board which is then fed through the serial connectivity via the RS232 port which results in physical motion of the Robotic Arm.

4. REQUIREMENTS

4.1 Functional Requirements

1) The system should be able to authenticate users requesting control of the Robot.
2) The mechanical assembly (Robotic Arm) should have 6 degree of freedom to enable it to provide services like pick and place.
3) The user should be able to operate the Robot via a web browser.
4) The user should be able to operate each of the Robot DOF individually so that it would provide him with the flexibility of programming the Robot to achieve various tasks.
5) The user should be able to store the positions and operate the Robot in the autonomous mode.
6) The user should be able to operate the Robot at various speeds.
7) The user should be able to stop/interrupt when the Robot is running in the autonomous mode.

4.2 Hardware Requirements

1) The system should have a mechanical assembly (Robotic Arm) which would act as the controlled mechanical (embedded) device.
2) The Robot should have a control system of its own.
3) The control system should have the connectivity to be plugged onto the internet via the RJ45 cable and jack.
4) The control system should have a serial, CAN, 1 wire, etc connectivity to control the servo controller.
5) The control system should have additional connectivity to add enhancements like the mobile mechanism, vision system and area detection using proximity sensors.
6) Have a wireless connectivity so that it could be free from physical connections for mobility.

4.3 Software Requirements

1) The user should be able to operate the Robot via a web browser implying that the robotic system should be controlled via a web application.
2) The web application should be deployed and hosted on a web server so that the program could run on the control system and is only invoked via the web browser over the internet.
3) All the complex position definition and speed control algorithms should be implemented into a Web Service and hosted onto an Application Server as components to be consumed by multiple servers through proxies created on them.
4) The control system should have an operating system to configure the system, ftp and deploy web servers, application programs and web application, run them on the server and also debug them.
5) The control system should control the Robot by a known and mature protocol like the RS232.
6) The servomotor controller should have an inbuilt firmware to interpret the byte stream from the controller into appropriate signals for motor movements.

5. SYSTEM ARCHITECTURE

Over the years, Java has proven to be the most matured language when it comes to the Internet. It has a range of middleware technologies like EJB (Enterprise Java Beans), XML (External Markup Language), JMS (Java Messaging Service), Web Services, etc that have maturity in terms of its capabilities. Even the web servers and application servers like WebSphere, WebLogic, Tomcat, Apache Web Server, etc have all contributed well in terms of stable server technologies to achieve complex tasks in the world of E-commerce.

This range of middleware Java Technologies can be taken advantage of even in the embedded space. This framework exhibits the use of web services which works on the http protocol and can be consumed not just by the java applications, but also by other technologies like .NET which can make use of these web services. This design exhibits how Java and Web Services can be integrated together to work in tandem to achieve the task of teleoperating a robot over the Internet.

5.1 Hardware Architecture

Hardware system architecture as shown in figure 1 comprises of

1) Process Server – Windows XP machine hosting the IBM WebSphere
   a) Application Server – hosting the Process Module (Web Application).
2) Service Provider – Windows XP machine hosting the IBM WebSphere
   a) Application Server – hosting the Service Module (Web Services).
3) Robot Controller – Dallas Semiconductor DS80C400 board hosting the
   a) Tynamo Embedded Web Server – hosting the Control Module (Web Application).
4) Servo Controller – 16 channel servo controller which communicates to external control systems using the serial protocol – which interprets byte streams into physical motor motion.
5) Robotic Arm – SMARTARM mechanical assembly having 6 DOF.

The hardware components used in this design are off-the-self products like the SMARTARM Robot Assemble Kit and the Dallas Semiconductor Network Controller DS80C400 based controller board. The hardware architecture is developed by integrating and configuring these components together to host internet based web services and communicate with each other using the http and serial protocols.

In this design the Robot Controller interacts with the 16 channel serial servo controller inbuilt into the Robotic Arm via the RS232 serial port using the Java serial comm. API. The Robot controller is the DS80C400 board which has the following capabilities.

1) Serial Port to interface to the Robotic Arm.
2) Ethernet connectivity through the RJ45 connector.
3) Capability to configure the system to have an IP address so that it could be located over the internet.
4) Inbuilt java JVM to program, deploy and host Java web components so that the system could be controlled via a web browser.
The Robot Controller is configured to enable it over the Internet by installing the SLUSH operating system. The Robot Controller sends out a stream of bytes to the servo controller which it interprets into physical motion. The byte format is as in Table 1 below.

| Start byte = Servo number, Servo position, 0xFF 0x00-0xFE 0x00-0xFE |

Table 1: Byte Stream format [10]

The Robot Controller is configured to enable it over the Internet by installing the SLUSH operating system. SLUSH is used to configure the control system by assigning it an IP address, gateway IP and DNS. In order to bring an embedded system onto the Internet it is needed to assign a static IP address so that external machines could locate and ping to it. To do so a LAN infrastructure is setup and configured so that multiple computers and control systems could connect onto a single infrastructure to communicate with each other using TCP/IP, HTTP, FTP, RS232 and other protocols.

5.2 Software Architecture

As seen in Fig 2 and Fig 3 the software architecture which is the heart of the system comprises of

1) Client Web Browser – This is a software application shipped by Microsoft (Internet Explorer) or the Netscape browser through which the client connects to web applications over the internet using the http protocol.
2) Process Module – Web Application – serves client requests
3) Service Module – Web Service – hosting the position interpolation and speed control algorithms
4) Robot Control Module – Web Application – interprets the position information from the process module into byte streams to control the robot based on position inputs from the service module.
This software architecture/framework is implemented around the following Java technologies:

1) Application Servers – An application server is a component-based product/software that resides in the middle-tier of a server-centric architecture. It provides middleware services and state maintenance, along with data access and persistence. Java application servers are based on the Java™ 2 Platform, Enterprise Edition (J2EE™).

2) Web Servers – A Web application runs within a Web Server/container. The Web server/container provides the runtime environment through components that provide naming context and life cycle management. Some Web servers may also provide additional services such as security and concurrency management.

3) Web Application – A Web application is a dynamic extension of a Web Server. There are two types of Web applications:
   a) Presentation-oriented: A presentation-oriented Web application generates Web pages containing various types of markup language (HTML, XML and so on) in response to requests.
   b) Service-oriented: A service-oriented Web application implements the endpoint of a fine-grained Web service. Service-oriented Web applications are often invoked by presentation-oriented applications.

4) Java Servlet – Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications without the performance limitations of CGI programs.
5) Java Server Pages – Java Server Pages (JSP) technology is the Java platform technology for delivering dynamic content to web clients in a portable, secure and well-defined way. The Java Server Pages specification extends the Java Servlet API to provide web application developers with a robust framework for creating dynamic web content on the server using HTML and XML templates and Java code which is secure, fast and independent of server platforms.

6) Web Services – Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.
The basic platform is XML plus HTTP. HTTP is a ubiquitous protocol, running practically everywhere on the Internet. XML provides a meta language in which one can write specialized languages to express complex interactions between clients and services or between components of a composite service. Behind the facade of a web server, the XML message gets converted to a middleware request and the results converted back to XML.

The full-function web services platform can be thought of as XML plus HTTP plus SOAP plus WSDL plus UDDI. SOAP is a protocol specification that defines a uniform way of passing XML-encoded data. In also defines a way to perform remote procedure calls (RPCs) using HTTP as the underlying communication protocol.

UDDI provides a mechanism for clients to dynamically find other web services. Using a UDDI interface, businesses can dynamically connect to services provided by external business partners. A UDDI registry is similar to a CORBA trader, or it can be thought of as a DNS service for business applications. A UDDI registry has two kinds of clients: businesses that want to publish a service (and its usage interfaces), and clients who want to obtain services of a certain kind and bind programmatically to them. SOAP is layered over HTTP and assumes that requests and responses are UDDI objects sent around as SOAP messages.

WSDL provides a way for service providers to describe the basic format of web service requests over different protocols or encodings. WSDL is used to describe what a web service can do, where it resides, and how to invoke it.

7) Java Serial Communication - The Java Communications 3.0 API is a Java extension that facilitates developing platform-independent communications applications for technologies such as Smart Cards, embedded systems, and point-of-sale devices, financial services devices, fax, modems, display terminals, and robotic equipment. As shown in the figures 2 and 3 the design is structured into components that are reusable, modular, loosely coupled, portable and scalable for future enhancements. This software architecture uses the Enterprise Java and Web Services which can be broken down as follows

1) Process Module - This is a Java web application deployed onto the IBM WebSphere application server to serve client requests. To do so it hosts the user interface that the client uses to interact with the system in the form of JSP pages. This application is further broken down into

a) User Login - This is the login JSP page that authenticates the user before getting in the system user interface to interact with the system. Here the system presents the user with the login screen to enter the user/pswd credentials. These credentials are then passed on to the Authentication Module as seen in Fig 3.

b) Authentication Module - Here the user's credentials are cross-checked against what is available in the database (in our case a properties file as seen in Fig 3).

c) User Interface - These are JSP pages that accept user inputs and invoke the appropriate modules of the process component. This is the main page of the application which accepts position and speed definitions. These positions are bundled into a 2 dimensional ArrayList and passed on to the Process Component.

d) Process Component - Processes the user inputs before passing them to the Web Service. As seen in Fig 3, the process component interprets each of the positions which are defined as integer values into an array of bytes shown in Table 1. Thus creating a 3 dimensional ArrayList. This is then converted into 2D Array of XSOBytes which is in the form that the Web Service understands.

e) Communication Component - This module takes care of finding and invoking the appropriate Web Service which is an XML request over http.

2) Service Module - This is the Web Services component that implements the position interpolation and speed control. Based on the selected speed and input positions, the intermediate positions are interpolated. The interaction with the Web Services is in the form of SOAP requests and responses. Here the Web Service calculates all the intermediate positions and adds another dimension to the Array of XSOBytes as seen in figure 3. This is then passed on back to the Process Component of the Process Module. There it is then converted back to a multidimensional ArrayList which is in the form recognized by the Robot Control Module.

3) Robot Control Module - This is a web application which accepts requests from the process module. This can further be broken down into

a) Robot Control Process Component - which is a servlet deployed on the Dynamo embedded web server that serves http requests. The interpolated positions from the Web Services are passed onto as input to the web application. Here each of the 7 set of byte arrays (which defines a position) are stripped out from the ArrayList in a loop (for each of the 10 positions) and passed onto the servomotor controller via the communication component using the Java comm. API.

b) Robot Control Communication Component - The process component makes a call to this communication component, which passes on the streams of bytes to the 16 channel servomotor drive. This is achieved using the JAVA Serial Comm. API. The drive then interprets the servomotor parameter as indicated by the last byte in the array as seen in Table 1 into a particular degree of rotation of the servomotor indicated by the second byte in the array.

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

A demonstration of the designed Robotic Control system was conducted through the Web Browser for moving the Robot through a sequence of motions and
performing an autonomous motion/task. The following conclusions are thereafter deduced.

1) Only authenticated users could gain access to the system based on the credentials defined in the authenticator properties file.

2) The user could define a numeric position for each of the 7 servos that constitute a physical position of the Robotic Arm and see it actually move to those coordinates thus making it easy for the user to define a set of positions to achieve a particular task.

3) The user could enter non-valid and out of range positions and see the system prompt him for valid inputs.

4) The user could run the Robot in an autonomous mode at different speeds and see the difference in physical motion of the Robotic Arm.

5) As seen from the values gathered in the previous section there is a lesser difference between two intermediate position/servo values for motions with slower speed thus indicating that the interpolation logic would calculate larger number of intermediate positions for slow speeds and vice versa to achieve a smooth and well defined path.

6) It was seen that the XP machine hosting the process component and the TINI controller board hosting the control component have been configured to recognize each other’s IP addresses over an Ethernet connectivity, thus making it a single homogeneous Robotic system that could communicate to each other over the internet. Due to this Ethernet connectivity they can also be wirelessly connected thus enabling physical independence of the Robotic Arm from the XP machine thus making it to be a mobile if required.

7) The XP machine hosting the service component can be configured to be recognized as a separate node on a LAN network. It can be independently run and connected to by several Robotic systems. This gives the architecture the ability of centralizing all its complex algorithms into Web Services that could be reusable by multiple robotic systems which is independent of platform and language.

6.2 Future Work

Enhancements: Level 1:

1) Adding Mobility to the system.
2) Collision detection and Avoidance.

Enhancements: Level 2:

1) Vision Capabilities.
2) Path finding capabilities.
3) Collaboration with other robots.

REFERENCES

[1] Crossplatform - Using C#, Java Webservices and JMS together by Armoghan Asif - accessed on 19/2/2007 at 8:00pm.


[7] M. Farajmandi and J. Gu, Max Meng, Peter X. Liu and Yangquan Chen “Internet Based Wireless Mobile Robot Network” – accessed on 19/02/2007 at 8:00 pm.

