# Exclusive Online Robotics: Robotic Arm Implementation and Testing

**James Carroll, Tyler Thomas**

**Draft #1**

**June 2, 2014**

**CEN 4935 Senior Software Engineering Project**

**Instructor: Dr. Janusz Zalewski**

**Computer Science & Software Engineering Programs**

**Florida Gulf Coast University**

**Ft. Myers, FL 33965**

# 1. Introduction

This XOR Robotic Arm (XRA) is a web application for telerobotic control of a robotic arm over the Internet. The specific requirements and design description for the XRA are contained in [1] and [2]. This document covers the details of the implementation and testing of the XRA software.

The XRA implementation has been done with a strict policy of platform-independence, so that both developers and users of the XRA can use practically any operating system any platform. This is a major divergence from the previous projects [3, 4, 5, 6, 7], which all utilized Microsoft's development tools and Windows Embedded CE. In addition to eliminating any restrictions on platform, the XRA is also free from any proprietary interests, making it easier to use, replicate, and extend.

The rest of the paper will cover the materials used for the XRA, the code that makes it work, and the testing procedures used to ensure every requirement has been met.

## 1.1 Bill of Materials

| | |
|---|---|
| AL5A robotic arm + SSC-32 Serial Servomotor Controller | $280 |
| USB-to-Serial adapter | $10 |
| TI Beagleboard-xM | $150 |
| Logitech C920 USB Webcam | $75 |
| **Total** | $515 |

**Table 1 – Bill of Materials**

Note that the major expenses, the AL5A and the Beagleboard-xM, were selected because of their availability at the authors' institution. There are, however, much more affordable options for low-power robotic arms and development boards, like the OWI-535 and the Raspberry Pi. Of course, using a different robotic arm would most likely require a few changes to the server-side code. It

is also possible to just run the server-side programs on any old PC, which would eliminate the need to purchase another system. Lastly, the Logitech C920 was selected because of its hardware compression in both the MJPEG and H.264 formats; however, there are much less expensive USB webcams with hardware compression. What this all means is that it is very possible to recreate this project with under $100 in hardware costs. Of course, the software is all 100% free.

### 1.2 Tools and Software

Server-side software

- Node.js, with the following packages installed
    - Socket.io
    - SerialPort
- Nginx
- MJPG-Streamer (for online MJPEG streaming)
- SSH daemon (typically enabled by default) for remote access

Client-side software

- Modern web browser
    - e.g. Chromium or Firefox
- SSH client
    - e.g. built-in `ssh` utility (Linux, Mac OS), PuTTY
- SFTP client
    - e.g. WinSCP, Filezilla, Cyberduck
- Text Editor
    - e.g. Sublime Text, Notepad++, Vim, Emacs

## 2. Implementation

The best starting point for understanding how the XRA operates under the hood is in the xor_radp.js file. Run file will open up a socket that listens for connecting controllers, and initializes communication with the SSC-32 device. The first 13 lines of this program in Listing 1 do just this.

**Listing 1: Snipped from xra_device_port.js**

```
var radpSocket = require('socket.io').listen(8000);
var SerialPort = require('serialport').SerialPort;

/* Serial port connection to the SSC-32 */
var ssc = {
  port : new SerialPort('/dev/ttyUSB0', {
    baudrate : 115200
  }),
};
```

To run the xor_radp.js program, Node.js is invoked and passed the file name.

```
  [root@apollo ~]# node xor_radp.js
```

The `radpSocket` listens on port 8000, because port 80 is already bound to by the HTTP server. In order to serve the Socket.io files to the client, a TCP proxy pass-through is needed so that requests for socket.io on port 80 are passed on to port 8000. This is done in Nginx by adding the lines red in Listing 2 to the nginx.conf file. This file is typically located at /etc/nginx/nginx.conf.

**Listing 2: Snipped from nginx.conf**

```
http {
  ...
  server {
    ...
    location /socket.io/ {
      proxy_pass http://localhost:8000;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
```

```
    }
    ...
  }
  ...
}
```

The last component to describe is the control panel. The control panel is a web page that is served to the user upon directing their browser to the IP address or URL of the server. The full listing for the control panel web page is in the Appendix.

**Listing 3: Snipped from xra.html**

```
<!-- snip -->

<script>
$(function() {
  var radp = io.connect('http://onlinerobotics.net:8000');

  var countdownTimeoutID = -1;
  var countdownDateTime = new Date(0,0,0,0,5);

  radp.on('connect', function(s) {
    /* Update status labels */
    $('#connection-status').text('Connected!');

    /* Start the countdown timer */
    var secondsString = countdownDateTime.getSeconds().toString();
    if (secondsString < 10)
      secondsString = '0' + secondsString;

    $('#countdown').text(countdownDateTime.getMinutes()
      + ':' + secondsString + ' remaining');

    countdownTimeoutID = setInterval(function() {
      countdownDateTime.setSeconds(countdownDateTime.getSeconds()-1);

      var secondsString = countdownDateTime.getSeconds().toString();
      if (secondsString < 10)
        secondsString = '0' + secondsString;

      $('#countdown').text(countdownDateTime.getMinutes()
        + ':' + secondsString + ' remaining');
    }, 1000 /* countdown timer interval */);
  });

  radp.on('disconnect', function() {
    clearTimeout(countdownTimeoutID);
    $('#countdown').text('');
    $('#connection-status').text('Disconnected');
```

```
  $('#connection-status').css('color', 'red');
});

$('#rotation').slider({
  slide: function(event, ui) {
      radp.emit('cmd', {
        'servoNum':0,
        'angle':ui.value*180/100
    });
  }
});

$('#shoulder').slider({
  slide: function(event, ui) {
      radp.emit('cmd', {
        'servoNum':1,
        'angle':ui.value*180/100
    });
  }
});

$('#elbow').slider({
  slide: function(event, ui) {
      radp.emit('cmd', {
        'servoNum':2,
        'angle':ui.value*180/100
    });
  }
});

$('#wrist').slider({
  slide: function(event, ui) {
      radp.emit('cmd', {
        'servoNum':3,
        'angle':ui.value*180/100
    });
  }
});

$('#claw').slider({
  slide: function(event, ui) {
      radp.emit('cmd', {
        'servoNum':4,
        'angle':ui.value*180/100
    });
  }
});

radp.on('status', function(msg) {
  $('#status').text(msg);
});
```

```
  radp.on('fdbk', function(msg) {
    $('#rotation').slider("value",100*msg.servo0/180);
    $('#shoulder').slider("value",100*(msg.servo1-40)/140);
    $('#elbow').slider("value",100*(msg.servo2-15)/165);
    $('#wrist').slider("value",100*msg.servo3/180);
    $('#claw').slider("value",msg.servo4);
  });
});
</script>

<!-- snip -->
```

The code in Listing 3 contains the JavaScript code behind the control panel. This code is rather concise, as JavaScript provides rich APIs for creating event-driven applications, and jQuery has been used as a simple way to reference and manipulate page elements to create a dynamic web page.

The first line of this script tries to establish a socket connection with the RADP after the page has loaded. The RADP port number has been specified by appending ":8000" to the server IP or URL. The next line of code sets up a function to be called when the socket is successfully established. This "on-connect" event will notify the user that they have successfully connected by updating the "connection-label" element.

### 2.1 System-specific Implementation Details

Arch Linux has been selected as the OS of choice for this particular instance of the XRA, as it offers a "bare-bones" installation of Linux and comes with no graphical desktop environment. Arch uses a powerful system management daemon called *systemd* [8] that makes it easy to create persistent background services. *systemd* has been leveraged in order to make both the Robotic Arm Device Port and video streamer into system services that start automatically at boot time.

System services are created by writing .service files in the `/usr/lib/systemd/system/` directory. Listings 4 and 5 content the contents of the radp.service and video-streamer.service.

**Listing 4: radp.service**

```
[Unit]
Description=Robotic Arm Device Port
```

```
Requires=nginx.service
Before=nginx.service

[Service]
Type=simple
PIDFile=/var/run/xra.pid
Restart=always
StandardError=syslog
SyslogIdentifier=RADP
User=root
WorkingDirectory=/root
ExecStart=/usr/bin/node /root/xor_radp.js

[Install]
WantedBy=multi-user.service
```

**Listing 5: video-streamer.service**

```
[Unit]
Description=MJPEG Stream on port 8080

[Service]
Type=simple
PIDFile=/var/run/mjpg_streamer.pid
ExecStart=/usr/bin/mjpg_streamer -i "input_uvc.so -r 320x240 -f 10"

[Install]
WantedBy=multi-user.target
```

The following command will enable these services so that they start automatically at boot time.

```
[root@apollo ~]# systemctl enable radp.service
[root@apollo ~]# systemctl enable video-streamer.service
```

# 3. Testing

## 3.1 Test Plan

The testing procedures outlined in this section will ensure that the software that has been delivered meets the requirements set forth in the SRS [1]. Each of these requirements will have a specific testing procedure that will test the compliance of the software.

While other systems may require more complex testing methods, such as batch scripts, the XRA project can mainly be tested by visual inspection or manual interaction. Other testing, which has been applied, was used to gauge the ease of use. This subsection outlines how each of the requirements will be tested, and the following subsection will display which tests have been passed, and which have failed (and why).

The great majority of requirements have been put in place to promote user ease. It is with that motivation that several volunteer testers were chosen with varying degrees of technological literacy. Upon any troubling interaction by a user, the portion of the XRA software under question was closer analyzed and improved upon until no usability problems were found.

Other than the requirements which are obviously satisfied upon use of the client side RACP by a single user, some requirements called for the access of the robotic arm over more than one machine. In the case that the device port was already bound to one RAC, inspection by a second user as to whether requirements relating to display and socket connection were made.

## 3.2 Test Results

| | |
|---|---|
| 3.1.1 | The RADP shall communicate with an exterior servo controller. |
| Passed | xor_radp.js uses the Node-SerialPort library to communicate with the external SSC-32. |

| | |
|---|---|
| 3.2.1.1 | The control panel shall be served to the operator in the form of a web page. |
| Passed | Directing a web browser to OnlineRobotics.net/xra will retrieve the control panel |

| | |
|---|---|
| | from the XRA server. |

| | |
|---|---|
| 3.2.1.2 | The control panel should contain a live video stream of the robotic arm. |
| Passed | A video stream of the arm is displayed at the top of the control panel. |

| | |
|---|---|
| 3.2.2.1 | The RAC should connect to the RADP as soon as it is loaded. |
| Passed | In order to conform to requirement 3.2.3.2, if another client is connected to the RADP, the current client fails to connect. In all other cases, it connects right away. |

| | |
|---|---|
| 3.2.2.2 | The RAC shall contain five slider elements for controlling the arm's five joints. |
| Passed | Each slider element also has a label at each end indicating which physical orientations correspond to each slider extreme |

| | |
|---|---|
| 3.2.2.3 | Each of the sliders in requirement 3.2.2.2 shall be used to adjust the angle of one of the arm's joints by transmitting the desired angle value. |
| Passed | The data sent with a command emission contains both information relating to which servo is being controlled and to which angle it will be set. |

| | |
|---|---|
| 3.2.2.4 | Upon connecting to the RADP, the RAC sliders shall update to reflect the positions of the arm's joints. |
| Passed | In the interest of protecting the hardware, the arm is set to a predetermined position (entirely vertical) upon a disconnect to minimize joint stress. Upon connection, the RAC sliders are set to reflect this position. |

| | |
|---|---|
| 3.2.2.5 | Commands sent from the RAC shall conform to the XSMF. |
| Passed | Every message sent to the RADP has a target identifier and value field for joint angle. |

| | |
|---|---|
| 3.2.3.1 | The RADP shall listen for connecting RACs on a specific port. |
| Passed | The RADP listens on port 8000. |

| | |
|---|---|
| 3.2.3.2 | The RADP shall only accept one RAC connection at a time. |
| Passed | When the RADP is in use, additional connections are rejected and prompted with the message “The XRA is currently in use. Please try again later.” |

| | |
|---|---|
| 3.2.3.3 | The RADP shall convert XOR Commands received from a RAC into device specific commands for the robotic arm controller. |
| Passed | This is done in xor_radp.js in the “on-command” callback function. |

| | |
|---|---|
| 3.2.3.4 | The RADP shall send any status changes to the connected RAC using XOR Feedback in XSMF. |
| Passed | Status changes are sent with a target identifier of “status,” with a value containing the status update message. |

| | |
|---|---|
| 3.2.3.5 | The RADP should notify any rejected RACs using XOR Feedback containing a reason for the rejection. |
| Passed | When a connection is rejected, a “status” message is displayed on the control panel below the video feed. |

| | |
|---|---|
| 3.4.1 | Any tools used for developing the XRA should be free and open-source. |
| Passed | Arch Linux, Node.js, and its packages are all free and open-source. |

| | |
|---|---|
| 3.4.2 | Any communication between the RAC and RADP should conform to the XSMF. |
| Passed | Every message sent to or from the RADP is `emit()`’ed with a target identifier followed by target values. |

| | |
|---|---|
| 3.5.1.1 | The XRA shall favor platform-independent software. |
| Passed | The control panel works on Windows, Mac OS, Linux, and mobile platforms. The XRA source code can be modified via SFTP and any text editor. |

| | |
|---|---|
| 3.5.2.1 | The XRA should be accessible at all times, except when taken down for maintenance. |

| | |
|---|---|
| Passed | Note: There is no visibility at night. However, a relay attached to a lamp could be used to turn on a light whenever an operator connects to the RADP. |

## 4. Conclusion

The goal of this project was to create a telerobotics web application for controlling a robotic arm. Simply put, this goal has been accomplished. Furthermore, there is no need to take this statement at face-value; the project is hosted at OnlineRobotics.net for everyone to see.

In summary, the following goals have been accomplished for this project.

- Create a cross-platform web application, accessible from anywhere in the world with an Internet connection.
- Use exclusively open-source tools so that anyone may view, use, and extend the XRA source code without paying a dime.
- Leverage techniques and technology to create highly maintainable and reliable software.

At the time of this writing, the XRA is in a very early stage. There are practically limitless extensions, modifications, and applications for this software. To name a few…

- Implement a queue so that additional clients can wait for their turn to control the arm
- Have multiple arms that can be controlled by multiple clients
- Add more widgets to the control panel, like network stats, uptime, a client queue, etc.
- Implement a login feature so that only credentialed user can connect to the RADP
- Design a hardware controller that interprets the arm movements of the operator in order to control the arm
- Establish an online community so that others can host their own telerobotic devices at OnlineRobotics.net
- Extend the software to be able to control other types of robotics systems

With the prevalence of Internet-driven technology, anybody should be able to have their own online telerobotic systems. As long as the XOR project remains free and open, the may one day be a reality.

## 5. References


[1] J. Carroll and T. Thomas. *Exclusive Online Robotics: Robotic Arm - Software Requirements Specification*. http://itech.fgcu.edu/ 24 Apr. 2009

[2] J. Carroll and T. Thomas. *Exclusive Online Robotics: Robotic Arm - Software Design Description*. http://itech.fgcu.edu/ 28 Apr. 2009

[3] Daboin, Carlos. *Robotic Arm Connectivity User Manual Project*. http://itech.fgcu.edu/ 13 Aug. 2009

[4] LaForge, Robert. *AL5A Robotic Arm Project: Web-Based Control with Spatial Awareness and Intuitive Manipulation.* http://itech.fgcu.edu/ N.d.

[5] Saldivar, Adrian and Kyle Rosier. *Remote Software Updater for eBox 2300 and Windows CE6*. http://itech.fgcu.edu/ 4 April 2012

[6] Fernandez, Arnold and Victor Fernandez. *Robotic Arm Remote Control Maintenance User Manual.* http://itech.fgcu.edu/ 18 April 2013

[7] Baquero, Abraham. *Robotic Arm Remote Control*. http://itech.fgcu.edu/ 8 Nov. 2013

[8] Wiki. *systemd*. https://wiki.archlinux.org/index.php/systemd/ 27 May 2014

[9] Surg, Ann. *Transcontinental Robot-Assisted Remote Telesurgery: Feasibility and Potential Applications*. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1422462/ Apr. 2002

[10] United States Department of Justice. *Vanguard Robot Assessment*. https://www.ncjrs.gov/pdffiles1/nij/204637.pdf 4 July 2004

[11] Something Interesting Digital. *Sid – Office Robot*. http://sidigital.co/sid

[12] Goldberg, Ken. *The Telegarden*. http://www.ieor.berkeley.edu/~goldberg/garden/Ars/

[13] Joyent, Inc. *Node.js* http://nodejs.org/

# Appendix

All of the source code for the XRA project is hosted online at http://git.io/xra.