# Developer Guide

**JWebSocket**

**Arduino Remote Control Demo**

**Dariel Noa Graverán**

1. **Overview**

Arduino Remote Control Demo is an application developed with the framework jWebSocket and Arduino hardware platform. The application is based in a plug-in, which sends and receives data of the micro-controller of Aduino´s Platform. The source code is structured in three parts: the web application controller, the jWebSocket server and the microcontroller of circuits of the Arduino Platform. The application uses an event-driven programming. It promotes the use of libraries like Raphael for the treatment of vector images on the web, jQuery for access and animations of the HTML elements and, RxTx library for send and receive data from USB port

1. **Infrastructure solution**

The solution is built following the model that EventPlugIn owns in jWebSocket Server EventPlugIn can listen to events that are released when the Micro-controller of Arduino´s circuit sends data to the server. The following is the diagram of components for a better understanding:

|  |
| --- |
| [Component Diagram](Images/Components_Diagram.png) |

*Ref. to Fig.1: Component Diagram*

**Description:**

* Arduino Remote Control Demo Application uses the libraries, JavaScript, jQuery and Raphael.
* Arduino Remote Control Demo application sends and receives information through JavaScript Client Library of jWebSocket.
* The jWebSocket Client communicates with jWebSocket Server through Websocket protocol.
* jWebSocket Server contains Arduino RemoteControlPlugIn, who receives and sends data to the controller application.
* The Arduino Connection component is contained in Arduino RemoteControlPlugIn. This component is who sends and receives data from the micro-controller of Aduino´s circuit directly.
* Micro-controller of Arduino´s circuit has connected a kit contains 4 LEDs, and a joystick.

1. **Requirements for use**

The application has the advantage of work on Linux, Windows and Mac OS, but for its total operation requires the following:

* OpenJDK 1.7 or higher.
* RxTx native libraries copied into the binaries folder of the Java Virtual Machine(JVM).
* Micro-controller of circuit of the Arduino hardware platform.
* Web browser with support for Websocket protocol.

In the xml file associated with the configuration of the Arduino RemoteControlPlugIn, the developer may must specify which port will be used for connecting the Micro-controller of Arduino´s circuit.

For controller packages application, jWebSocket Server, the native libraries for access to serial port and the containing program of Micro-controller of Arduino´s circuit, the developer should access the URL: <http://jwebsocket.org/download/>

1. **Modularization of the application**

**Server side**

|  |  |
| --- | --- |
| Solution Name | Arduino Remote Control Demo |
| Locationof the sources inthe SVN server: | <https://jwsdev.org:9443/svn/jWebSocket/branches/jWebSocket-1.0/jWebSocketPlugIns/jWebSocketArduinoPlugIn/> |
| SVNbranch: | jWebSocket-1.0 |
| Mavendependencies: | <dependency>  <groupId>org.jwebsocket</groupId>  <artifactId>jWebSocketServer</artifactId>  <version>1.0</version>  </dependency>  <dependency>  <groupId>org.rxtx</groupId>  <artifactId>rxtx</artifactId>  <version>1.2.7</version>  </dependency> |
| JAR module: | ArduinoRemoteControl-1.0.jar |
| Packagestructure | Base namespace  org.jwebsocket.plugins  [Ref. to Packet structure server](Images/Structure_Package_Server.PNG) |
| org.jwebsocket.plugins.arduino:  Contains the Arduino RemoteControlPlugIn which sends and receives data to the controller application. | |
| org.jwebsocket.plugins.arduino.connection:  It contains the elements necessary for establish the serial communication with the Micro-controller of Arduino´s circuitdirectly. | |
| org.jwebsocket.plugins.arduino.conecction.event:  In this package is the event who starts when the data is sent from the Micro-controller of Arduino´s circuit to the jWebSocket Server. | |
| org.jwebsocket.plugins.arduino.event.c2s:  the package contains to the definition of the events who start from the controller application when it is heard by jWebSocket Server. | |
| org.jwebsocket.plugins.arduino.event.s2c:  The package contains to the definition of the events who start the launch from jWebSocket Server when are heard by the application controller. | |
| org.jwebsocket.plugins.arduino.util:  The package contains utilities to make working with the application contains classes for the treatment of data that is sent from the microcontroller. | |

**Client Side**

The controller application, in addition to elements forming part of the HTML markup language, has a JavaScript structure, constituting the burden of implementation on the client. The JavaScript code is divided into several files that are presented to below:

* Init.js: The file contains methods for connecting to the jWebSocket server, the method for send a command from the controller to the Micro-controller of Arduino´s circuit and initializes the design elements that are part of the graphical interface of the controller application.
* jwsRemoteControl.js: Contains the functions of the demo application: functions that rely on design elements to show status of the LEDs, and another function that displays the coordinates of the joystick using an object created with the library Raphael.

The controller application also uses JavaScript libraries in order to facilitate the work in some aspects; the libraries used are the following.

* jWebSocket.js: This library is responsible for establish the connection with the server controller application of jWebSocket.
* jwsEventsPlugIn.js: Extension of jWebSocket client library that defines the event model which is based on the Arduino Remote Control Demo.
* jwsCache.js: Library to manage the web browser cache.
* jQuery.js: JavaScript library has multiple functions, including access to DOM elements, effects, Ajax requests and event handling.
* Raphael.js: JavaScript library for the treatment of vector images on the Web, is used in the application in order to show the movements of the joystick.

1. **Source code structure**

**Server side**

The following figure shows in detail the directory structure containing the server code jWebSocket are shown including all java classes with which account the application. The classes are distributed in packets that were previously described, and which in turn are associated with directories shown below.

|  |
| --- |
| [Ref. to Folder structure server](Images/Server_Code_Tree_V2.PNG) |

*Ref. to Fig.2: Directory Structure Server*

The following figure shows the root directory of the application source code, as well as describing those directories and files for better understanding.

|  |
| --- |
| [Ref. to Server Root Directory](Images/Structure_Code_Server.PNG) |

*Ref. to Fig.3: Server root directory*

**Description**

* src: The directory is all the source code of the application, within him is the main/ java and then a directory structure that matches the one shown in Figure 2.
* target: The directory temporarily stores the compiled source code, its content is not included in the version control.
* pom.xml: Configuration File ArduinoRemoteControl module generated by Maven tool. This file defines among other features, dependencies, in this case and jWebSocket RxTx libraries.

**Client side**

The following figure shows where the source code distributed web application controller.

|  |
| --- |
| [Ref. to Directory structure of the client application](Images/Client_Code_Tree.PNG) |

*Ref. to Fig.4: Directory structure of the client application*

* 1. **packageStructure in the server side**

*org.jwebsocket.plugins.arduino* package contains the following class

|  |  |
| --- | --- |
| Class | Description |
| ArduinoRemoteControlPlugIn | This class represents the plug-in used to release and listen to events, which serve to communicate with the web application controller. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.ArduinoRemoteControlPlugIn | | |
| Type | **Method** | **Description** | |
| org.jwebsocket.plugins.arduino.connection.  ArduinoConnection | getArduinoConnection() | The method returns the object that connects directly to the Micro-controller of Arduino´s circuit. | |
| void | Initialize() | Inherited method that initializes the connection to the microcontroller circuit. | |
| void | processEvent(org.jwebsocket.eventmodel.event.C2SEvent aEvent, org.jwebsocket.eventmodel.observable.ResponseEvent aResponseEvent) | Inherited method that processes events that are launched from the application controller. | |
| void | sendJoystickPosition(java.lang.Integer aX, java.lang.Integer aY) | The method throws an event to the web application controller, which contains the coordinates of the joystick. | |
| void | sendLedState(java.lang.Boolean aBlue, java.lang.Boolean aRed, java.lang.Boolean aGreen, java.lang.Boolean aYellow) | This method raises an event to the controller application, which contains information on the status of the LEDs. | |
| void | setArduinoConnection(org.jwebsocket.plugins.arduino.connection.  ArduinoConnection aArduinoConnection) | The method changes the object that will connect to the microcontroller circuit. | |

*org.jwebsocket.plugins.arduino.connection* package contains the following class:

|  |  |
| --- | --- |
| Class | Description |
| ArduinoConnection | Through this class plug-in can communicate with micro-controller circuits of the Arduino hardware platform, the class has methods to send data, and event listener using the data you send the circuit to the USB port. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.connection.ArduinoConnection | | |
| Type | **Method** | **Description** | |
| void | closePort() | Free USB port that is configured to communicate with the microcontroller. | |
| java.lang.Integer | getDataBits() | Gets the number of bits of data, which set up the port, the default is 8 bits. | |
| java.lang.Integer | getDebugRate() | Gets the port connection speed, the default is 9600 bps. | |
| java.lang.Integer | getParity() | Get parity with the connection to be made, the default value is 0, indicating no parity. | |
| java.lang.String | getPortName() | Gets the name of the USB port, which should be connected to the microcontroller circuit. | |
| java.lang.Integer | getStopBits() | Returns the stop bits associated with the connection via the USB port, the default value is 1. | |
| java.lang.Integer | getTimeOut() | Gets the time in milliseconds to wait for connection to the USB port, the default will be 2000 milliseconds. | |
| void | init() | Initializes the connection to the USB port, with all the settings needed to communicate. | |
| void | sendCommand(java.lang.String aCmd) | Sends data to the microcontroller circuit, this method sends text strings. | |
| void | sendCommand(java.lang.Integer aCmd) | Overloaded method that sends commands to the microcontroller circuit, which specializes in sending numeric data. | |
| void | serialEvent(gnu.io. SerialPortEvent aEvent) | Inherited method that is executed when the circuit sends data to the USB port, an event listener method that contains the data sent from the circuit. | |
| void | setDataBits(java.lang.Integer aDataBits) | Change the data bits to be sent by the USB port. | |
| void | setDebugRate(java.lang.Integer aDebugRate) | Change the speed of connection to the USB port. | |
| void | setParity(java.lang.Integer aParity) | Changes the parity. | |
| void | setPortName(java.lang String aPortName) | Change the port. | |
| void | setStopBits(java.lang.Integer aStopBits) | Change the stop bit. | |
| void | setTimeOut(java.lang.Integer aTimeOut) | Change time out conection. | |

*org.jwebsocket.plugins.arduino.connection.event* package contains the following class:

|  |  |
| --- | --- |
| Class | Description |
| DataIn | The class defines an event that is triggered when the micro-controller circuit sends data to the USB port. The main function of this event is to be heard by the plug-in ArduinoRemoteControlPlugIn, as it contains the data received from the circuit, which is then sent through another event to the Web application controller. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.connection.event.DataIn | | |
| Type | **Method** | **Description** | |
| org.jwebsocket.plugins.arduino.connection.event.DataIn | DataIn(java.lang.String aData) | Constructor for the class entering data to the event. | |
| java.lang.String | getData() | It can extract the event data. | |

*org.jwebsocket.plugins.arduino.event.c2s* package contains the following class:

|  |  |
| --- | --- |
| Class | Description |
| Command | Class that defines the event that launches the web application controller to send information to microcontroller circuit. This event is heard by the plug-in ArduinoRemoteControlPlugIn. |
| StartArduinoRemoteControl | Class is to define the event that is launched from the web application controller, when it starts. Once the event is played by the plug-in ArduinoRemoteControlPlugIn, is sent to the controller application status LEDs. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.event.c2s.Command | | |
| Type | **Method** | **Description** | |
| java.lang.Integer | getCmd() | Method that allows the plug-in ArduinoRemoteControlPlugIn extract the event data and then send them to the microcontroller circuit. | |

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.event.c2s.StartArduinoRemoteControl | | |
| Type | **Method** | **Description** | |
| The event does not have any functionality, since that will only serve to notify the application started controlling | | | |

*org.jwebsocket.plugins.arduino.event.s2c* package contains the following class:

|  |  |
| --- | --- |
| Class | Description |
| S2CJoystickPosition | Class that defines the event responsible for sending the application to the coordinates of the joystick controller. |
| S2CLedState | That class defines the event responsible for sending the application to the coordinates of the joystick controller. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.event.s2c.S2CLedState | | |
| Type | **Method** | **Description** | |
| org.jwebsocket.plugins.arduino.event.s2c. S2CLedState | S2CLedState(java.lang.Boolean aBlue, java.lang.Boolean aRed, java.lang.Boolean aGreen, java.lang.Boolean aYellow) | Class constructor allows to introduce the data will contain the event and then send it to the application controller. | |
| java.lang.Boolean | getBlue() | Returns the status of the blue LED on true and false off. | |
| java.lang.Boolean | getRed() | Returns the status of the red LED on true and false off. | |
| java.lang.Boolean | getGreen() | Returns the status of the green LED on true and false off. | |
| java.lang.Boolean | getYellow() | Returns the status of the yellow LED on true and false off. | |
| void | writeToToken(org.jwebsocket.token. Token token) | Inherited method that is called from the framework jWebSocket. Used to convert the data to Token, preparing the event for shipment. | |

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.event.s2c.S2CJoystickPosition | | |
| Type | **Method** | **Description** | |
| org.jwebsocket.plugins.arduino.event.s2c. S2CJoystickPosition | S2CJoystickPosition(java.lang.Integer aX, java.lang.Integer aY) | Class constructor allows to introduce the data will contain the event and then send it to the application controller. | |
| java.lang.Integer | getmX() | Allows to obtain the x value of the event stored position. | |
| java.lang.Integer | getmY() | Allows to obtain the y value of the event stored position. | |
| void | writeToToken(org.jwebsocket.token. Token token) | Inherited method that is called from the framework jWebSocket. Used to convert the data to Token, preparing the event for shipment. | |

*org.jwebsocket.plugins.arduino.util* package contains the following class:

|  |  |
| --- | --- |
| Class | Description |
| JoystickProgram | Class that contains functions to give treatment to the data being sent from the circuit, referring to the coordinates of the joystick. |
| LedsProgram | Class that contains functions to give treatment to the data being sent from the circuit, concerning the status of the LEDs. |

Detailed description of the methods:

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.util.JoystickProgram | | |
| Type | **Method** | **Description** | |
| java.lang.Integer[] | refineValue(java.lang.Integer aX, java.lang.Integer aY) | The method limit the coordinate values ​​to send to the controller application. | |
| java.lang.Integer[] | treatValues(java.lang.String aData) | By this method it is possible to transform the data sent by the micro-controller regarding the coordinates, returns an array, where positions 0 y1 are the values ​​of x, y respectively. | |

|  |  |  |
| --- | --- | --- |
| Class: org.jwebsocket.plugins.arduino.util.LedsProgram | | |
| Tipo | **Método** | **Descripción** | |
| java.lang.Boolean[] | parseLedState(java.lang.Integer aState) | The method transforms the data sent by the Micro-controller of Arduino´s circuit with respect to the state of the LEDs, as a parameter an integer between 0 and 15, the number indicates that LED is on and / or reflected off a binary number 4 bits, the return of the method is an array with 4 positions representing each of the LEDs, in the order blue, red, green and yellow on true, false off. | |

* 1. **Source code structure in the client side**

The web application controller, consists of elements of web programming. This refers to HTML pages, CSS files and JavaScript files. Then be shown the functions held by each one of the JavaScript file containing the weight of the client application programming:

|  |  |  |
| --- | --- | --- |
| File: init.js | | |
| Type | **Function** | **Description** | |
| void | connect() | Opens the connection to the server jWebSocket, and generates the plug-in ArduinoRemoteControl, which will launch the events. | |
| void | disconnect() | Close the connection to the jWebSocket server. | |
| void | Init() | Initializes the design elements and then calls the connect () method described above. | |
| void | initCanvas() | Create an SVG image associated with an HTML element, to represent the movements of the joystick, this method uses the Raphael library. | |
| void | registerEvents() | Records events of the elements of level design mouse (onClick). | |
| void | sendCommand(aCmd) | Function that supports plug-in generated in the controller application to launch to the server jWebSocket the Command event. The ACMD is an integer parameter, used the numbers 49, 50, 51 and 52 for driving the LEDs blue, red, green and yellow respectively. | |
| void | startArduinoRemoteControl() | Makes a function call registerEvents () and throws to the server StartArduinoRemoteControl jWebSocket the event, telling the server that started the application controller. | |

|  |  |  |
| --- | --- | --- |
| File: jwsRemoteControl.js | | |
| Tipo | **Función** | **Descripción** | |
| void | changePosition(aX , aY) | The function takes care of the application display controller joystick movements. | |
| void | changeledsStatus(aBlue, aRed, aGreen, aYellow) | The role is responsible for representing the state of the LEDs, using CSS classes and attributes. Receives 4 Boolean parameters represent the state of each of the LEDs. | |

* 1. **Programing the micro-controller**

Programming the Micro-controller of Arduino´s circuit, is another important aspect in the process of application development. The codes are developed with an Arduino IDE is part of the hardware platform. The IDE allows you to create programs and then upload them to the circuit. Before this process must be configured in the IDE, the port will be connected to the circuit and the circuit model. An Arduino program consists of two main functions:

* setup(): This function typically sets the digital pin mode, in or out, in the case of the application, set pins 4, 7, 8 and 12 digital output, set the baud rate to 9600 bps and configure the analog pins 0 and 1 for the coordinates of the joystick.
* loop(): This function is performed as a cycle, it is practically the entire program that will control the LEDs and will monitor the joystick.

In addition to the aforementioned functions, the program contains other to be described below:

|  |  |  |
| --- | --- | --- |
| Fichero: rc\_demo.pde | | |
| Type | **Function** | **Description** | |
| void | changeLedState() | The function works with Boolean variables associated with each LED which prepares a variable with an integer from 0 to 15 and then sent via the USB port jWebSocket server. | |
| int | treatValue(int data) | Because the values ​​produced by the joystick is very sensitive, it is necessary to treat jWebSocket sent to the server, this function takes care of the problem. | |
| void | changePosition(int x, int y) | The function is executed when the joystick changes position jWebSocket sent to the server through the USB port the coordinates of the joystick, it sends a text string containing the values ​​x, and the text string is identified by start the character 'J'. | |

1. **Libraries and tools used**

* jWebSocket Framework, LGPL.
* Arduino Hardware platform, CC license (Creative Commons).
* jQuery JavaScript library, GPL v2.
* Raphael library, MIT License.
* RxTx library, GPL v2.1
  1. **Maven Configuration**

The ArduinoRemoteControl module has pom.xml file, where in addition to other parameters define the version of the module and dependencies which must include the module. Dependencies are an important aspect in projects created with Maven. You then see how they are configured units for better understanding.

|  |
| --- |
| [Ref. to Dependencies in the pom.xml file](Images/POM_file_Dependences.PNG) |

*Ref. to Fig.5: Dependencies in the pom.xml file*

The bookstore operated by the last unit shown in Figure 5, depending on the repository may not be available, if this happens you can change the version of the library by 2.1.7, this would not alter the operation of the application. However, it should be noted that by changing the version of the library will also change the version of the native library copied into the / bin folder of JDK.

1. **Hardware**

For remote control of devices, you must have a micro-controller circuit of the Arduino platform, the application was developed on the microcontroller Arduino Mega ADK, however you can use any other circuit of the platform Arduino Duemilanove, Arduino Diecimila or similar. Arduino circuits have a connection to the computer via the USB port, but older models are connected by the RS232 port.

To jump-start the application you must have, besides the micro-controller circuit the following:

* Four LEDs, blue, red, green and yellow.
* A joystick with two connections (x, y).
* Un cable USB.
* A module TinkerKit (Optional).

For more information on Micro-controller of Arduino´s platform, visit the site: <http://arduino.cc> .

1. **Continuous Improvement**

The server console jWebSocket recorded in all logs, related to what happens, eg when you throw an event, and when listening to an event, to achieve faster data processing on the server, the developer can go to server configuration and disable these logs. In the Web application is similar, by default the logs are disabled, because these events display all the console application, is costly.

1. **Configuración de la aplicación**

The application is subject to the settings that you can make the working environment jWebSocket, however you need to specify which port is connected to the Arduino circuit and for this we must access the file located at:

|  |
| --- |
| *$JWEBSOCKET\_HOME/conf/EventsPlugIn/rc-application/app-plugins/rc.xml* |

In rc.xml file, specifying all the details that has the plug-in ArduinoRemoteControlPlugIn, to configure the port should take into account that operating system will work. In Windows the port would be COM0, ....., COM4, on Linux it may be /dev/tty/USB0, /dev/tty/USB1 or similar and probably MacOS /dev/tty.usbserial-1B1.

|  |
| --- |
| [Ref. to Port settings in the rc.xml file](Images/PlugIn_Port_Configuration.PNG) |

*Ref. to Fig.6: Port settings in the rc.xml file*