Home Automation Remote Control

Nikhil Sharma

This document describes the usage and design of the home automation remote control system. The remote is a web interface primarily designed for mobile phones, and it can be viewed on any major web browsers that support Jquery mobile.

**Content of the folder**

* assets.txt – A list of URLs where the assets were taken from in order to produce this prototype.
* documentation.doc – You’re reading it now
* webclient/index.html – The entry page for the web client. The entire webclient needs to be hosted for the solution to work.
* webclient/config.json – The configuration file is used to define the user interface of the remote, and set the server where each request needs to be sent to.
* <server-url>/living-room/status.json – This is the sample data returned by the server when requesting the status of the components.
* <server-url>/living-room/update.json – This is the sample data returned by the server when making a request to control the components.
* js/jquery/jquery.js
* js/jquery/jquery.mobile-1.4.3.min.css
* js/jquery/jquery.mobile-1.4.3.min.js : Those files are taken from jquerymobile.com and they are required libraries to run the solution.
* js/remotecontrol.js: This contains the code for initializing the remote and handling server interaction through Jquery.
* components/base/basecomponent.js: Contains common functions and template definition used by each separate components
* components/curtains/asset/close-curtains.png
* components/curtains/asset/closed-curtain-icon.png
* components/curtains/asset/curtains-icon.jpg
* components/curtains/asset/open-curtains.png
* components/curtains/asset/opened-curtain-icon.png: Images used by the curtains control component.
* components/curtains/curtaincomponent.js: User interface component for curtains
* components/curtains/curtainscontroller.js: Server controller for curtains
* components/heat/asset/Farm-Fresh\_temperature\_5.png
* components/heat/asset/heat-down.png
* components/heat/asset/heat-up.png
* components/heat/asset/panel.jpg: Images used by the temperature control component
* components/heat/font/digital-7.ttf: Font used for the digital panel of the temperature control component
* components/heat/heatcomponent.js: User interface component for temperature
* components/heat/heatcontroller.js: Server controller for temperature
* components/light/asset/Light\_bulb\_icon.png
* components/light/asset/lightswitch.jpg
* components/light/asset/lightswitchoff.jpg: Images used by the light control component
* components/light/lightcomponent.js: User interface component for light
* components/light/lightcontroller.js: Server controller for light

**External code**

The three files in js/jquery folder are taken from external sources.

The object navbarHandler in components/base/basecomponent.js was adapted from external sources.

**Usage**

The remote is designed as a Jquery mobile web component, with three pages: One to control the light, one for the heat and one for the curtains. Each page is accessible by pressing on an icon from the navigation bar.

Each page has a customized control designed to provide a pleasing user experience. That is why the light control looks like an actual light switch that can be found in a house, and the heat control looks like a thermostat. For the curtain control, producing an animated curtain seemed a bit “overkill”, so we simply have a visual slider that can be controlled with two buttons.

**Configuration**

The remote is configurable through a config.json file shown as follow:

{ "url":"<server-url>/living-room",

"controls": [

{ "type":"light",

"title":"Light",

"hook":"lamp\_001"

},

{ "type":"heat",

"title":"Temperature",

"hook":"thermostat\_001"

},

{ "type":"curtains",

"title":"Curtains",

"hook":"window\_001"

}

]

}

Through the configuration file, the layout of the remote control (order of the buttons in the navigation bar) as well as the interaction with the server can be defined. The url is where the following calls will be made:

* status.json (GET): To get the status of the elements (light / heat / curtains)
* update.json (POST): To interact with those elements

Each control has a “type”, which can be either “light”, “heat”, or “curtains”. The type has to match the component created. Also, as more components are created, new types can be used.

The “title” is the text shown at the top of the visual display for each component. (It can be “Hallogen lamp”, “Window to the garden”, “Fridge temperature”…)

The hook is an ID recognized by the server. That ID will be sent for each update.json request. (the hookID is passed as a parameter and should be recognized by the server to know which real life component to interact with).

The remote can be configured with more than one of each component.

**Server API**

Note that each API calls sent to the server are logged via the JavaScript debugger, for informational purposes. They can be viewed via Firebug or through Chrome’s JavaScript console.

Status:

<server-path>/status.json (GET)

Returns:

Result=”success”

Controls= an array of statuses. For each status, the server’s hookID is returned, the type of the component, and extra parameters depending on the component.

For light, status=1 or 0 depending if the lights are currently on or off

For temperature, “temperature” is the current room temperature. “setTemperature” is the temperature that we’re trying to reach, and “unit” is the unit of measure of the temperature (Fahrenheit, Celsius, or Kelvin).

For window, the status is a number between 0 (closed curtains) and 1 (wide opened curtains).

Update:

<server-path>/update.json (POST)

The request accepts the hookID, and parameters depending on the component.

For light, “turn\_light=true” turns the lights ON, “turn\_light=false” turns the lights off.

For temperature, “setTemperature” is set to an integer, which is the value of the temperature. The unit is assumed to be the same as the one returned by the server (or Fahrenheit by default).

For curtains, “action=open\_curtains” to start opening the curtains, “action=close\_curtains” to start closing them. The curtains will move until they reach the maximum closed/opened or until the “set\_curtains” request is received. “set\_curtains” is set to a decimal number between 0 and 1, defining the final position of the curtain.

**Error handling**

A red error message will show if the remote cannot communicate with the server. The message is not intrusive like a popup, and goes away automatically after 5 secs. Some error messages also popup if the components are misconfigured or not implemented correctly.

Note that some error messages will not show in the UI if the interface is broken, but they can still be viewed in the JavaScript console.

**Custom components**

In order to create a new component, the light/heat/curtains component can be used as example. Basically, we need two objects:

* The component object, which defines the user interface
* The controller object, which decides how to send requests to the server depending on the commands received by the user interface

There are templates inside the basecomponent.js file that defines the interface of each object, and contains comments for each methods to be defined. Note that the user interface is produced using Jquery mobile, so the components will have to be designed the same way.

The components and controller components were separated for flexibility of implementation (one dev can work on component and the other can work on the controller), and also to ensure that the server calls aren’t just mapping to user interaction. For instance, the user will continuously press a button to increase the temperature, but the server should not receive continuous calls to go from 70, 71, 72, 73 degrees… Also, the server doesn’t need to be notified immediately when someone presses the up button for temperature, but that is not the case for the curtains components. The curtains should move as soon as the user presses the button, so a server request will be sent immediately on touch.

**Release notes:**

* The user interface currently shows a “Home” button which doesn’t do anything. It’s really just a bug, but this could also be a placeholder for a new feature! An extra view will show the plan of a house, and the user taps on the room they want to control.
* The remote is configured to interact with the mock server in <server\_url>, which obviously will not persist any changes.
* The buttons in the curtain controller display a blue glow that I wasn’t able to remove. It’s supposed to show which button has focus.
* I didn’t check the copyrights on images, so for an actual product those will have to be redesigned by XD. The list of links where they were taken from are listed in assets.txt

**Technical notes:**

The main technology used is Jquery mobile.

The solution can be run locally. It was tested on a Mac in Firefox and Google chrome.