OpenROV

How to create a software plugin for OpenROV Cockpit

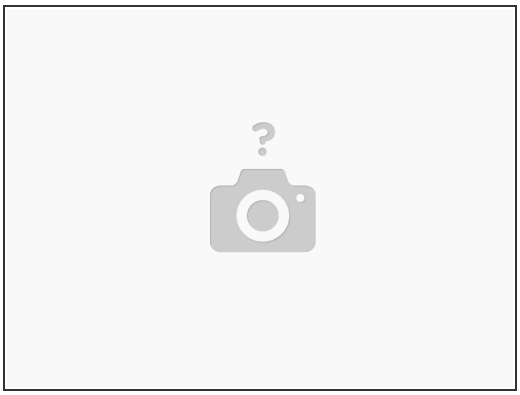
Walkthrough on creating a plugin focused on javascript. This walkthrough creates a rotating ROV on the screen that uses the navdata feed to have it mirror the physical orientation of the actual ROV.

Written By: Brian Adams

TOOLS:

- https://github.com/OpenROV/openrov-software (1)

Step 1 — How to create a software plugin for OpenROV Cockpit



- clone the OpenROV github repository
  - ***"git clone https://github.com/OpenROV/openrov-softw...

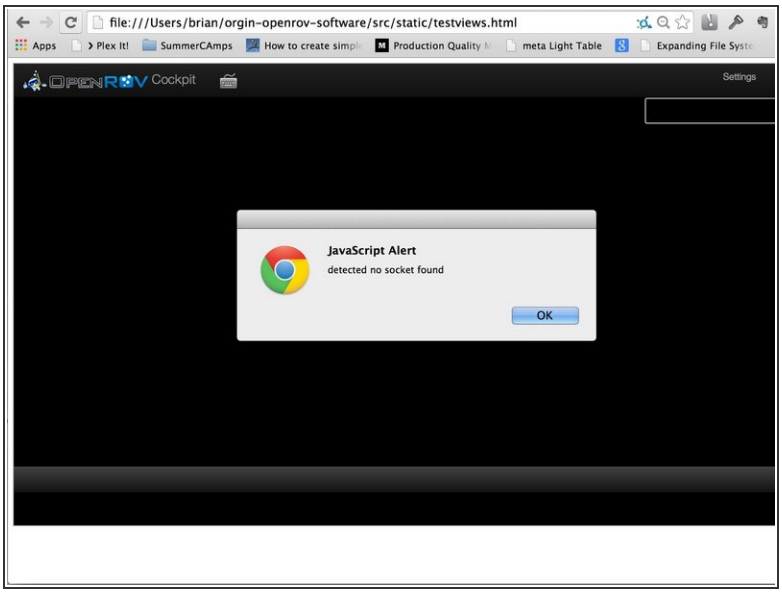
Step 2

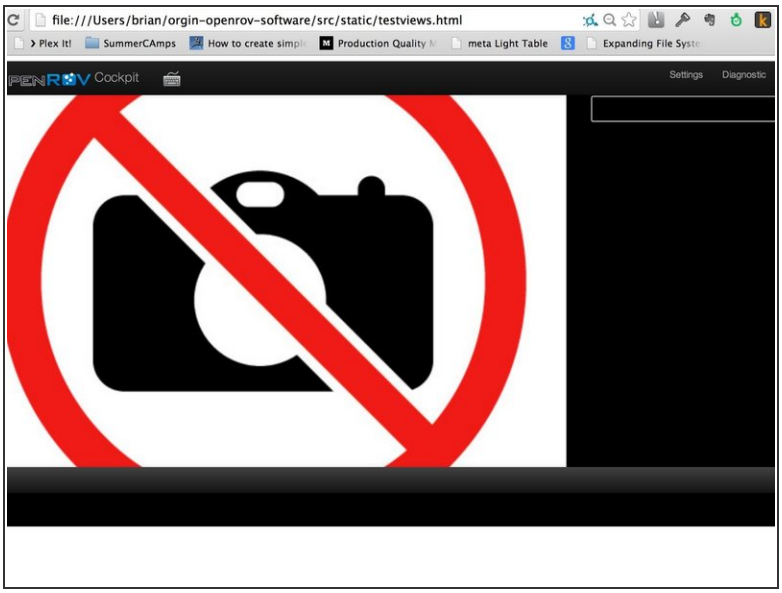
```
</script>
    <script>
        // load the template file, then render it with data
        var iframe = document.createElement("iframe");

            document.body.appendChild(iframe);

        var frameDoc = iframe.document;
            if(iframe.contentWindow)
                frameDoc = iframe.contentWindow.document; // IE
            // Write into iframe
            frameDoc.open();
            var scripts = [];
    /*   //     scripts.push('../lib/config.js');
            scripts.push('../plugins/arduinofirmwareupload/public/js/arduinofirmwareupload.js');
            scripts.push('../plugins/blackbox/public/js/blackbox.js');
            scripts.push('../plugins/capestatus/public/js/capestatus.js');
            scripts.push('../plugins/compass/public/js/compass.js');
           // scripts.push('../plugins/fpscounter/public/js/fpscounter.js');
            scripts.push('../plugins/googletalk_ipregistration/public/js/googletalk_ipregistration.js');
            scripts.push('../plugins/horizon/public/js/horizon.js');
            scripts.push('../plugins/photocapture/public/js/photocapture.js');
            scripts.push('../plugins/rovpilot/public/js/rovpilot.js');
            scripts.push('../plugins/telemetry/public/js/telemetry.js');
            scripts.push('../plugins/motor_diags/public/js/motor_diags.js');
           // scripts.push('../plugins/touchcontroller/public/js/gamecontroller.js');
            scripts.push('../plugins/touchcontroller/public/js/touchcontroller.js');
            scripts.push('../plugins/diveprofile/public/js/diveprofile.js');
            scripts.push('../plugins/tankcontrol/public/js/tankcontrol.js');
            scripts.push('../plugins/flybywire/public/js/flybywire.js');
*/
            var styles = [];
            styles.push('../plugins/touchcontroller/public/css/style.css');


            frameDoc.writeln(new EJS({url: '../views/index.ejs'}).render({title: "testing", styles:style
            frameDoc.close();
            //autoResize(iframe);

           iframe.height= "720px"; //360px";
           iframe.width= "1280px";//"567px"; //
       // new EJS({url: 'views/index.ejs'}).update('test', {title: "testing", styles:{}, scripts:{}});
    </script>


</body>
<!--  <frameset cols="250,*">
```

- Use your favorite text editor to open /src/static/testviews.html
- We will start with a clean workspace, so we need to comment out the other plugins that are used.
- Commend out all of the script.push(...) lines about midway down the document. You can use the /* ... */ to comment out multiple lines easily.
- Save the file

- Note there is a commandline option in the comments for how to launch chrome in a special file access mode. You will need to close any running instances of node, open a command window, and then start chrome in this special mode.
  - OSX: /Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome --allow-file-access-from-files
    - If you get issues with getting a profile lock, shut down any currently running instances of chrome
- From Chrome, do a file open, and select the testviews.html file you edited above. If you get "Cross origin requests" JS error try running a HTTP server, e.g. `python -m SimpleHTTPServer 8000` and access through http://localhost:8000/src/static/testviews.html.

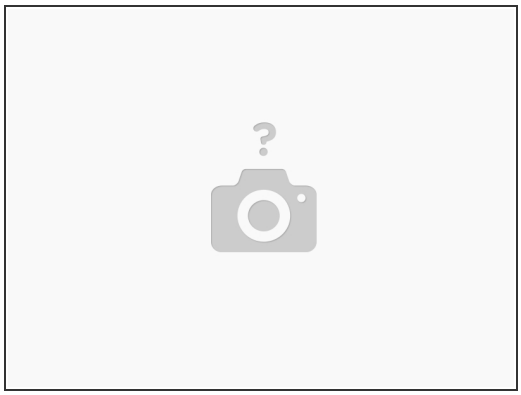
Step 3





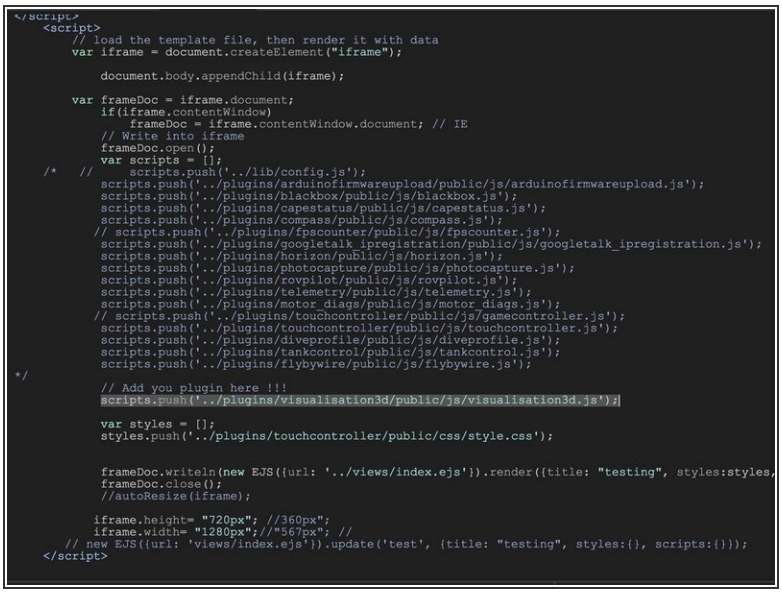
- Once you open up the file you should get a blank canvas with a warning dialog that socket.io could not be found. Just dismiss the message.
- You are now in a sandbox environment where you can modify all of the browser code and see it actually work without needing a node process connected to it.

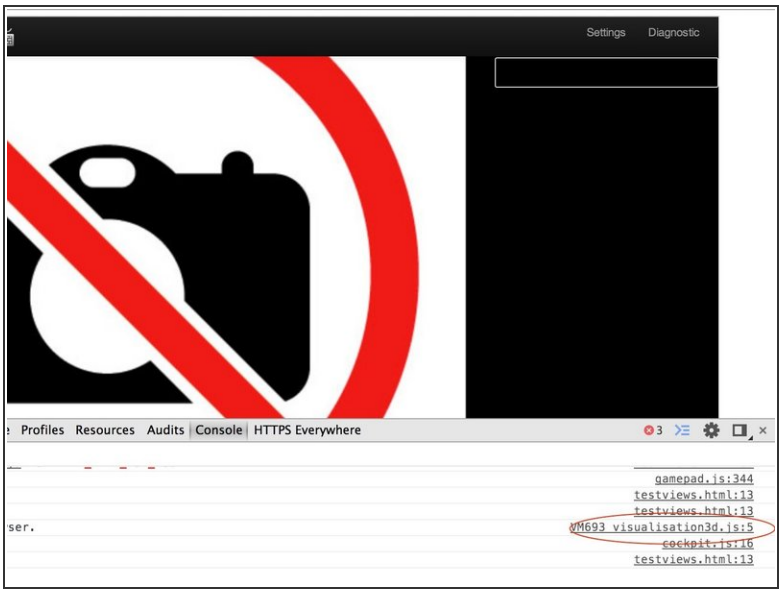
Step 4



- Now we need to create our base files for our new plugin. The easiest way to do this it to clone an existing plugin. I recommend starting with /src/plugins/example folder in to a new one called /src/plugins/visualisation3d. [br][code]cp -r src/plugins/example src/plugins/visualisation3d[/code]
- Inside the new folder, the convention is to name your javascript file that runs in the browser with the same name as your folder. So rename /src/plugins/visualisation3d/public/js/example.js to visualisation3d.js [br][code]mv src/plugins/visualisation3d/public/js/example.js src/plugins/visualisation3d/public/js/visualisation3d.js[/code]

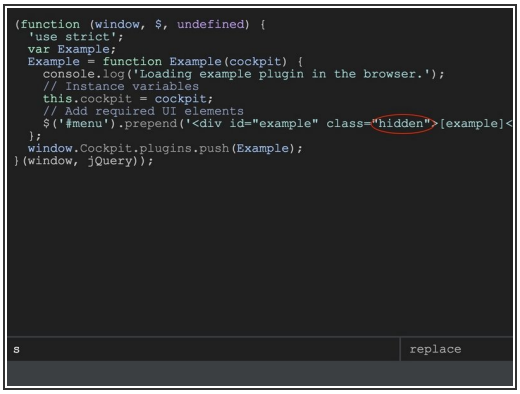
Step 5

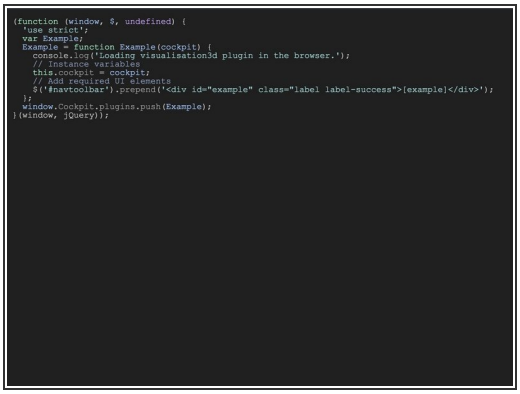




- Test the new plugin by adding it to the test harness.
- Go back to the textviews.html file you edited in the first step. Right below the lines you commented out before, you are going to add a new push command to push your plugin it to the array of registered plugins.
- Add the line as shown in the image. scripts.push('...
- Save you file. Reload the chrome tab that was displaying the file. Nothing obvious has changed.
- Open up Chrome developer tools however, and you should see a message that says the plugin example has loaded, and on the right hand side you should see the file listed is you visualisation3d.js file.

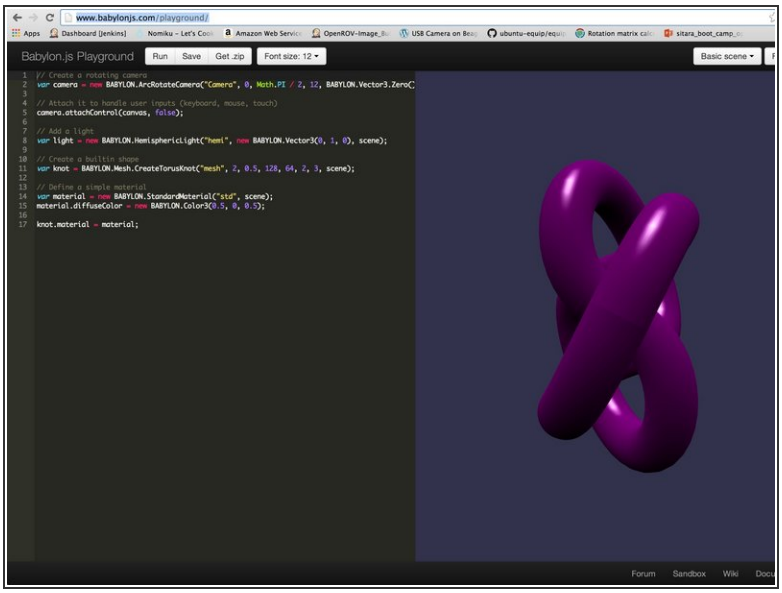
Step 6

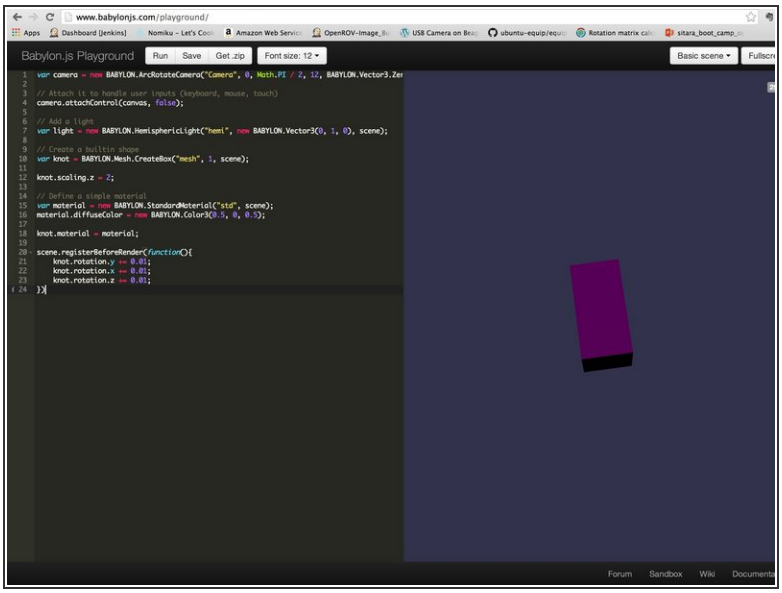




- Now that all of the plumbing for the new plugin is in place, it is time to start changing the interface around.
- Begin by opening up your visualisation3d.js file and delete the "hidden" class from the div tag being inserted in the code.
- And since the background is black, we should also change the font color to something that 'pops'. Since we are using bootstrap.js we can use one of there special classes. In this case where you removed hidden, replace it with 'label label-success'
- While we are at it, go ahead and change the log message so that is kicks out the name of your new plugin instead of 'example'.
- And finally, clean up all of the rest of the references of Example with the name of your plugin.
- Now reload the page in chrome and you should see a green 'example' at the top of the page.

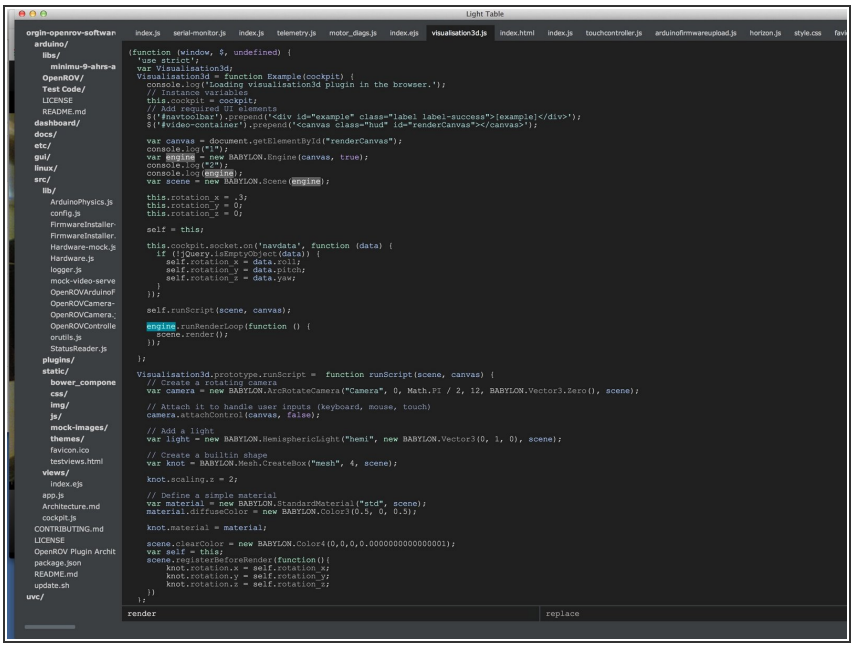
Step 7





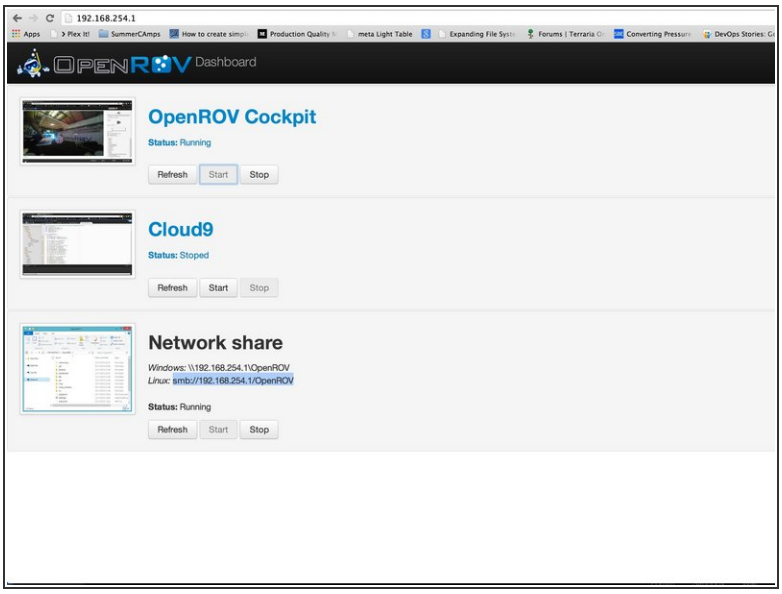
- Now the fun part. To replace the example innards with a call to your working plugin code.
- For this plugin we have chosen to use a third party library that makes 3d in javascript easier. We are using babylon.js.
- If you go to http://www.babylonjs-playground.com/ you get a tool that lets you dial in your code with an online browser.
- We have an image of the modified code we are using that presents more of an ROV shape ;-)
- You can now download the code which gives a .html file and a .js file which we will place in to the plugin.

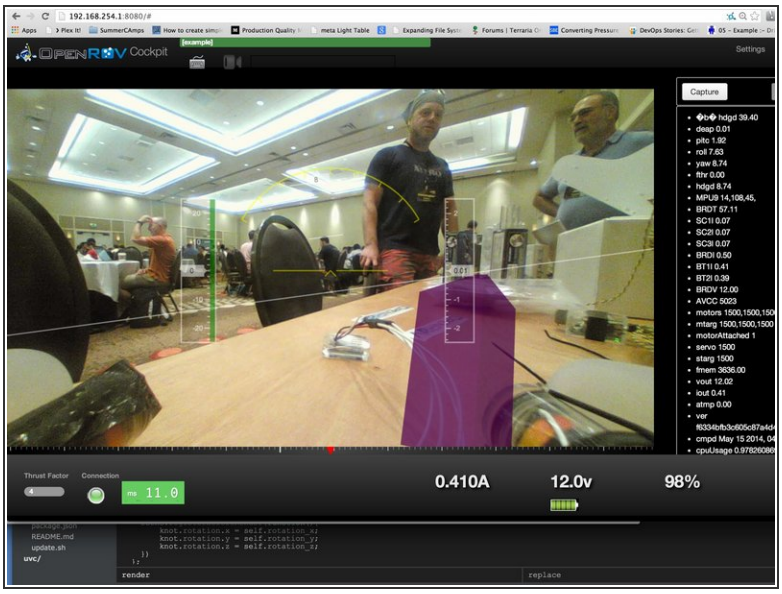
Step 8



- Copy the babylon.js file in to the same /public/js folder as your plugin. All .js files will automatically be loaded.
- The example code from the site simply adds a canvas to the page. We are going to do the same thing in the plugin, but we have to decide where to put it.
- Plugins have some references for a concept of the cockpit, global event emitter, etc.
- The fully updated plugin code picture is here.

Step 9





- To get the plugin on to the ROV, go to it default web address (usually 192.168.254.1). From the dashboard start the file sharing. Login to the ROV. Copy the folder for your plugin to the plugin folder on the ROV. Restart the cockpit process.
- Woot! There is our purple take on an ROV swirling around with real telemetry.
- Things to still do: Convert the degrees to radian which is what the library uses to get "real" orientation
- Add some love to the 3d model to make it look more like an ROV