

OpenIO Labs - System

Accessing the OpenIO Labs System

UG102 (v2017.1) April 3, 2017

Revision History

Date	Version	Revisions
3/4/2017	2017.1	<i>Initial release.</i>

Table of Contents

Revision History	2
Chapter 1: Introduction	4
Chapter 2: Logging In & Quick Tour	5
Logging In.....	5
Quick Tour.....	6
Chapter 3: IOIs and Capture	7
IOIs View.....	7
Capture View.....	14
Chapter 4: Main Control Tabs	19
Tags Tab.....	19
Report Tab.....	20
Search Tab.....	22
Widget Tab.....	23
Chapter 5: Example Use Case	25
Initial Setup And Configuration.....	25
Loading Our First ScriptML Script.....	26
Running the Script and Using Widgets.....	27
Additional Resources	31
References.....	31

Chapter 1: Introduction

This User Guide provides an overview of using the OpenIO Labs system. To follow the guide it is assumed that you have an active account with OpenIO Labs and that you can access the server and use the system accordingly. If you do not have an account, go to the OpenIO Labs web site (www.openiolabs.com) and sign up for a user account now.

This user guide is intended to guide you through the use of the OpenIO Labs system with the main emphasis on accessing the system through a Web browser. Some users will use the OpenIO Labs system to connect hardware devices and control hardware devices through the ScriptML interface. Although touched on here, this topic is covered in more detail in other User Guides e.g. Ref [1].

The structure of the OpenIO Labs system is covered in detail in Ref [2] and will only be summarised here. From the perspective of this User Guide, the system comprises edge devices (IOIs) connected to a server, the server with a range of databases and a Web browser that accesses the server and databases.

The IOIs and the OpenIO Labs server communicate using the industry standard LWM2M protocol (See Ref [3] for further details). The LWM2M protocol is responsible for managing the communications between the IOI and the server. These communications include Registration, Data Transfer, Control and Notification messages.

The user interacts with the OpenIO Labs system (including the server and the IOIs) by way of a browser interface. It is the user experience of accessing the OpenIO Labs system via a Web browser that this User Guide is focussed upon.

Chapter 2: Logging In & Quick Tour

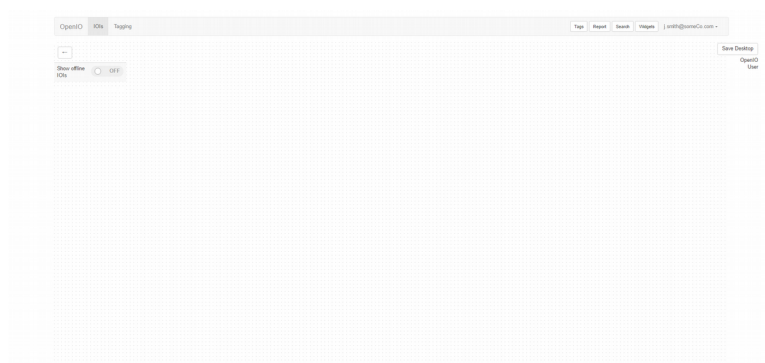
We will start this chapter with a quick introduction to the OpenIO Labs system and the steps to gain access to the system by a web browser, such as the Google Chrome browser..

Logging In

When you first register to use the OpenIO Labs system you will receive a user-id, password and server URL. To start, open your Chrome browser and enter the URL in the search bar. You should see a login page similar to the one shown below:



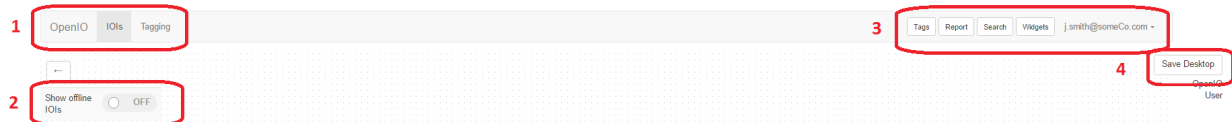
Enter the User Name and Password that you received in the registration e-mail, and click login. You will be greeted with the following screen.



If you see this screen you have successfully logged in to the OpenIO Labs system. In the following sections, we will guide you through accessing the different parts of the OpenIO Labs system via the browser interface.

Quick Tour

Starting with the initial log-in screen illustrated below, we will go through each of the key sections in turn that are presented at the top of the screen.



The use for each of these sections is described briefly in the table below, and in greater detail in following chapters.

Element	Description
1 – IOIs and Capture	These two selection buttons allow you to toggle between showing the IOIs or showing active Capture Sessions. The use of capture is described in detail later.
2 – Active selections	The contents for this section of the screen will depend on whether IOIs or Capture are selected. If the IOIs are selected, a list of active and in-active IOIs can be viewed. If capture is selected a list of active or inactive Capture Sessions is displayed.
3 – Main control tabs	The main control tabs allows you to display Tags that are active, view Reports of previously stored data collected, Search for specific stored data, Widgets used to process data and currently logged in user.
4 – Save Desktop	This button allows you to save the configuration of the desktop so that when you re-login you can resume where you left.

Table 1: Description of the main elements of the browser screen for the OpenIO Labs system

Chapter 3: IOIs and Capture

In this section we will look into the IOI and Capture views in more detail.

The IOIs are the edge devices that are connected to the OpenIO Labs system. Typically the IOIs are physical devices that allow you to inter-connect various sensors and actuators to the OpenIO Labs system. In addition, however, virtual IOIs (vIOI) can be attached to the server. The vIOI can be used to connect software applications to the OpenIO Labs system. Currently Matlab and Octave are supported as vIOIs.

Capture is used in the OpenIO Labs system to allow you to “tag” collected data. Multiple tags can be defined, and we have the concept of a Capture Session in which all data collected during that session can be retrieved at a later date for viewing or further processing.

IOIs View

We will now look at the IOIs tab that we saw in opening screen-shot. The IOIs tab is where you are able to view all of your active IOIs, details of the IOI and means to control the IOI.

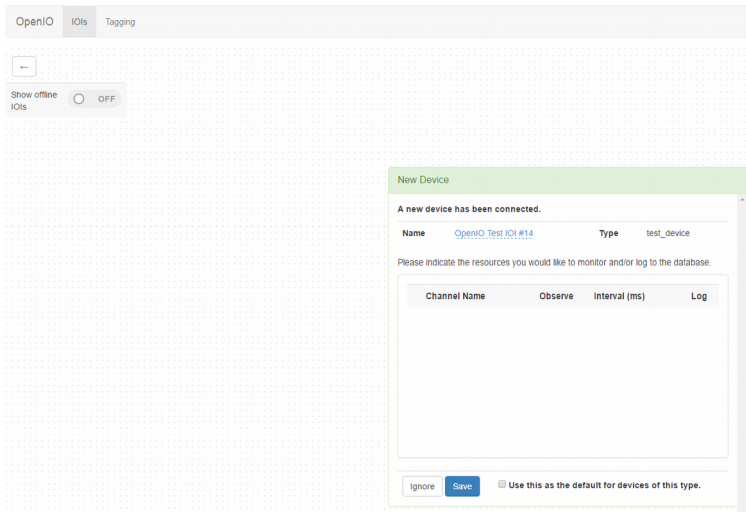
We will start with the assumption that you have just powered on an IOI and it has a connection to the Internet.

Once the IOI has successfully powered on, it will start by activating the IOI-client application. The IOI-client application will register the IOI with the OpenIO Labs server. The registration process is part of the LWM2M protocol (see Ref [3]) and is supported by both the IOI and the OpenIO Labs server.

Connecting an IOI

Once an IOI is registered with the OpenIO Labs server, you will see the following screen on your browser. The screen shows that a new device (IOI) has been connected to the system. The New Device window includes the following useful information:

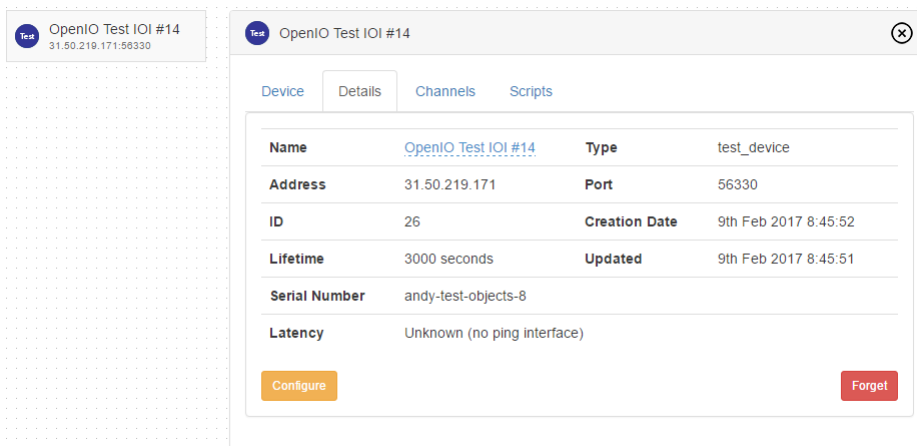
Name:	The name of the IOI device that has just been attached
Type:	A description of the type of IOI attached
Resources:	A list of resources that can be saved in the database (not present for this IOI)
Ignore/Save:	Two buttons that allow the IOI to be saved, or ignored by the system



Saving the IOI

In this instance, we want to save the IOI so that we can use it to collect data or perform a range of tasks. To do this, we click on the “Save” button.

When the IOI is saved, the browser view will change, and we will see a screen similar to this one.

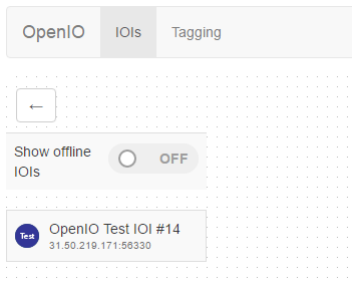


Here we see the active IOI present button (on the left) as well as the IOI information window (on the right).

The IOI information window allows you to view information on the Device, Details of the IOI, Channels on the IOI and Script management on the IOI. In the following sections, we will go through each of these in turn.

Closing the IOI Display

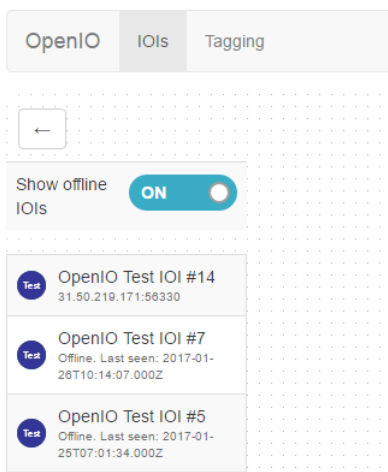
For the moment, we will close the IOI information window by clicking on the “X” in the top left corner. In doing so the main part of the browser window (on the top left) will look like this below.



You will notice that the IOIs tab has been selected (top middle), that we are not showing any offline IOIs but we can see the active IOI button that we have just connected to the system

Viewing Offline IOIs

If we now click on the button “Show offline IOIs” you may see a screen similar to this one (Note: if this is the first time you have used the system, there will be no off-line IOIs). Here we see the active IOI that we have just connected at the top of the list of buttons, and below that some off-line IOIs.

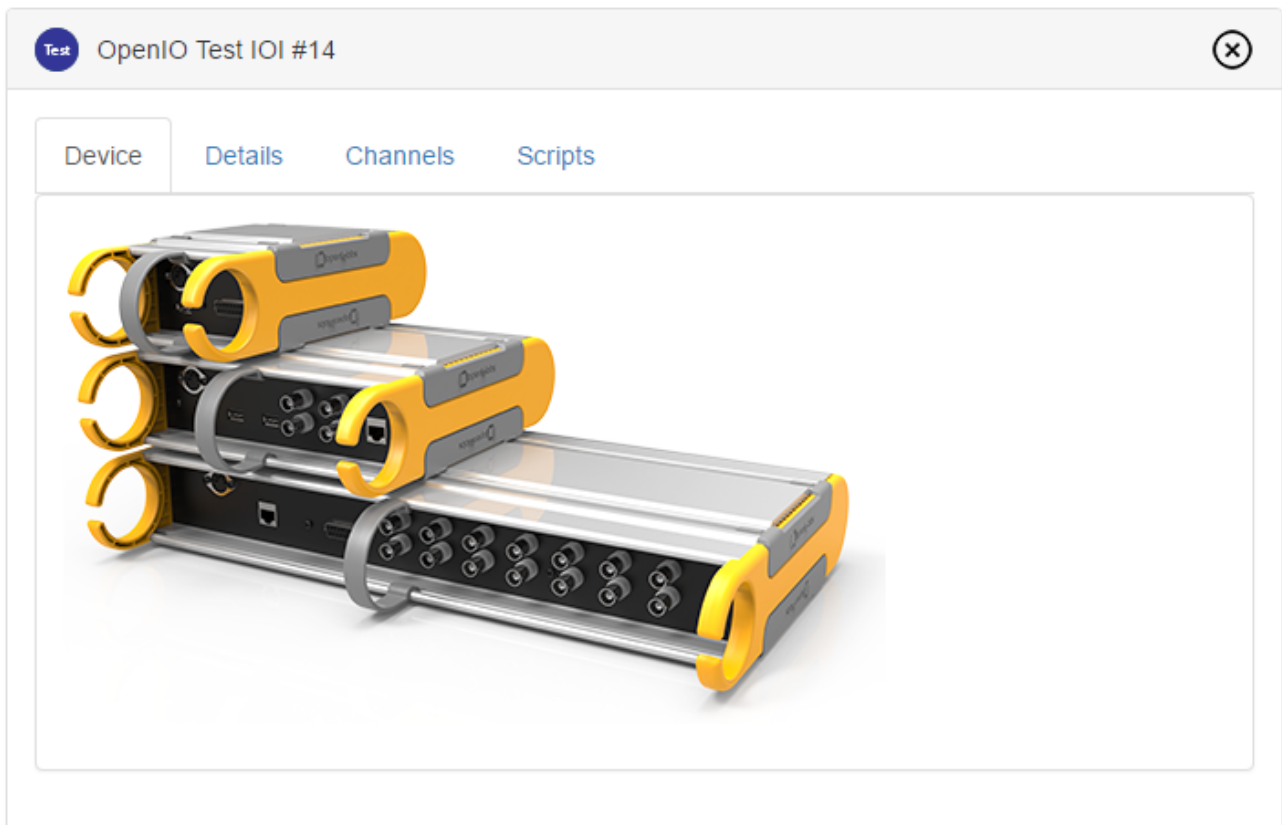


Off-line IOIs are your IOIs that have been connected to the OpenIO Labs system at some time in the past, and remembered by the server.

The reason why you may want to access these off-line IOIs, is that you may want to access and process the data that they previously collected. By clicking on either an active or an off-line IOI you will re-open the IOI Information Window that we saw previously.

IOI – Device Tab

Assuming that we have just clicked on the active IOI button, we will re-open the IOI Information Window. By clicking on the Device tab of this window we will see the following on the browser.



The image that we have on the browser indicates the device that is selected (active or off-line). The image will change from IOI to IOI and is intended to provide a simple pictorial representation of the IOI that you are using.

IOI – Details Tab

Next, we can click on the Details tab. You will see the following window on your browser. This window provides a summary of the details associated with this IOI. The meaning for each of these fields is given below the Details tab image below.

Test

OpenIO Test IOI #14

✕

Device

Details

Channels

Scripts

Name	OpenIO Test IOI #14	Type	test_device
Address	31.50.219.171	Port	56330
ID	23	Creation Date	8th Feb 2017 12:43:42
Lifetime	3000 seconds	Updated	8th Feb 2017 17:39:22
Serial Number	andy-test-objects-8		
Latency	Unknown (no ping interface)		

Configure

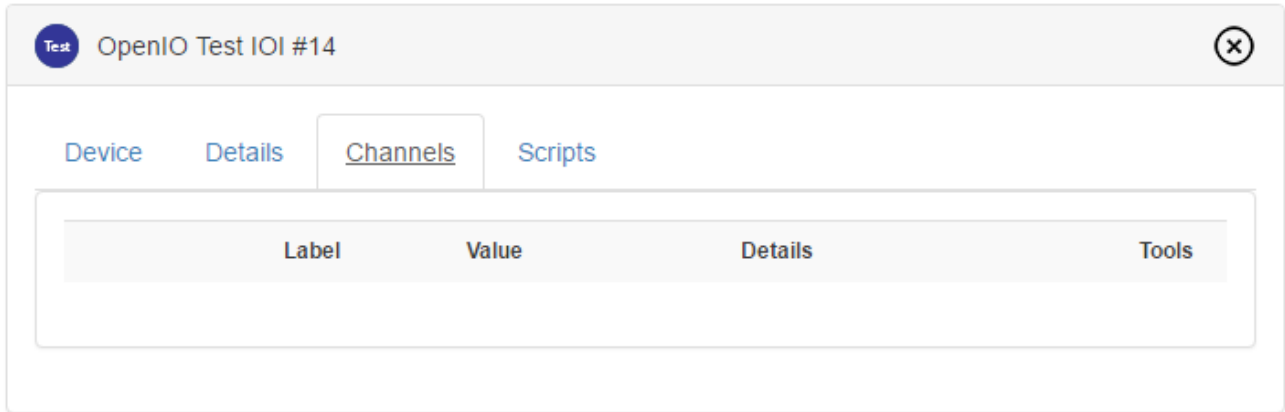
Forget

Field	Description
Name	This is the name of the IOI. It starts with a name assigned by the server, but by clicking on the name hyper-link in blue, you can edit the name to make it more meaningful to you. There are no-limits on how many times you change the name, but the server will only remember the last name that you assigned.
Address	The address is the IP address of the IOI as seen by the server. If you are on a local network, this will be different to the IP address of the IOI locally.
ID	The ID of the device is assigned by the server to allow the server to keep track of the different devices that are either active or off-line.
Lifetime	The Lifetime for the device is how long the device will remain active before a re-registration is required. If the device is actively collecting data this will occur automatically.
Serial Number	The Serial Number is the unique identifier that is assigned to every IOI and vIOI.
Latency	If Latency is configured for the IOI, this is the time taken for a periodic "ping-test". The Latency is useful if the IOI is operating over a poor Internet connection and can be used to indicate that there maybe network issues that should be resolved.
Type	The Type field defines the type of the IOI.
Port	The PORT field is the UDP port that the server sees from the IOI. This is an external port and not the port used locally (necessarily). The knowledge of the Port may help debug issues to do with Firewalls etc.
Creation Date	The creation data is the date when the device first registered with the server (or re-registered if the Forget button was used).
Updated	The Updated field indicates the last time that the device re-registered.

Summary of the fields in the Details Tab

IOI – Channels Tab

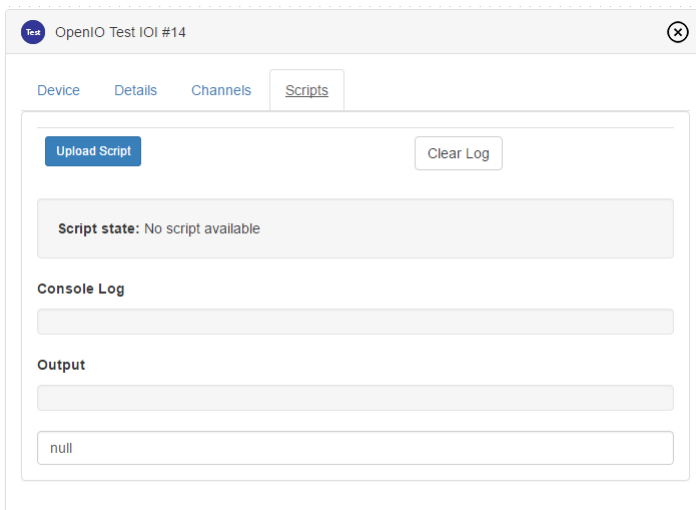
Now we can click on the Channels Tab and we will see the following Window in the browser.



The channels tab will contain details of the IOI objects that it supports. If the IOI is a fixed purpose IOI (i.e. one hard-coded as a DAC, say) then we will see this here. In this instance, this IOI uses ScriptML scripts for its operation, and consequently the channels tab is empty until a script is uploaded.

IOI – Scripts Tab

The IOI that we have registered with the server supports the uploading and execution of ScriptML scripts. As a consequence we will see the Scripts Tab visible. If we click on the Scripts Tag we will see the following window on the browser.



The Scripts window has a number of different areas as shown. The meaning of these different buttons and fields is as follows:

Field	Description
Upload Script	This button is used to upload a ScriptML script. For details on ScriptML and how it is used refer to UG401 (Ref [4]).
Clear Log	This button is used to clear the console logs shown below.
Script State	This field records the current state of the scripts.
Console Log	ScriptML console output. Used for displaying general reporting information.
Output	Output field used for presenting outputted data to the user

Summary of the fields in the Scripts Tab

Capture View

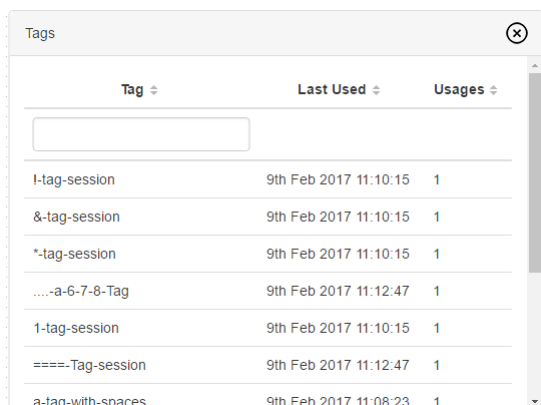
We will now look at the Capture tab, Tags, Tag Sessions and review their operation.

Use of Capture in The OpenIO Labs System

Before looking at the various ways that Capture is used in the OpenIO Labs system, it is worth looking at an overview of what capture is and why it is useful.

Capture, as the name suggests allows you to create tags, and have those tags associated with the data that is collected by the system.

First the structure of a Tag. A Tag can be any (in general) sequence of ASCII printable characters with the exception of a space (the browser will convert a space into a dash '-'). The following screen shows some examples of acceptable tags.



Tag	Last Used	Usages
l-tag-session	9th Feb 2017 11:10:15	1
&-tag-session	9th Feb 2017 11:10:15	1
*-tag-session	9th Feb 2017 11:10:15	1
....-a-6-7-8-Tag	9th Feb 2017 11:12:47	1
1-tag-session	9th Feb 2017 11:10:15	1
====-Tag-session	9th Feb 2017 11:12:47	1
a-tag-with-spaces	9th Feb 2017 11:08:23	1

The tags are intended to have some relevance to you, some form of memorable word or phrase could be used to allow you to identify the data that is associated with that specific tab.

We will see later that you may have multiple Tags associated with a given set of data, or with with a Capture Session (sessions are explained a little later).

If we first consider why you might want to have multiple tags. If you imagine that you are performing some form of experiment. Maybe this experiment takes a week to complete and that on each day, different aspects of the experiment are being studied. In the course of your experiment, you may find a particularly exciting result and you don't want to lose that result. So we are starting to see where capture may help here. Consider the following table:

Tag	Use
j_smith	You may want to tag all data that you collect with your name or equivalent. This may be particularly helpful if your organisation uses multiple instances of the OpenIO Labs system on the same server.
project_honey_monster	You may want to use the project name as the overall collection of the data collected during the experiments for this project.
experiment_21	We can imagine that you may want to use this tag for the whole duration of an experiment.
monday_26_jan_2017	You can use tags as a way to divide the work between the different days that you worked.
afternoon_session	Alternatively, you can sub-divide the tags in to smaller periods.
interesting_observation	If you find a specific interesting result collected in the data, you can tag that data as something that could be reviewed at a later date.

Example Use of Multiple Tags

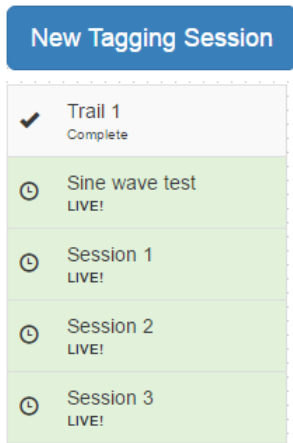
Capture Sessions

To use the tags that we have defined, we can define something called a capture session. A capture session can be thought of as the duration of a specific experiment or project. The capture session is a collection of Tags that you may wish to associate in some manner. You can use multiple capture sessions concurrently and each capture session can use the same or different tags.

A capture session is associated with one or more data sources. We will see later that you can associate one or more data sources with the capture session.

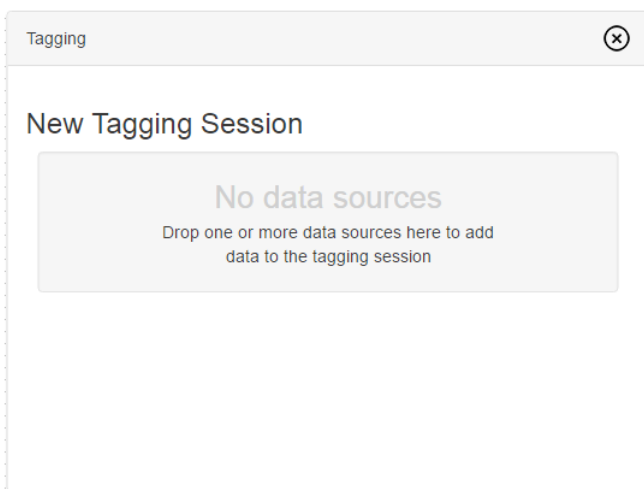
Capture View Details

To use capture and Capture Sessions, click on the Capture View icon on the top level of the browser window to the left-hand-side. You should see something similar to this (note if this is the first time a Capture Session has been created, only the New Capture Session button will be visible).



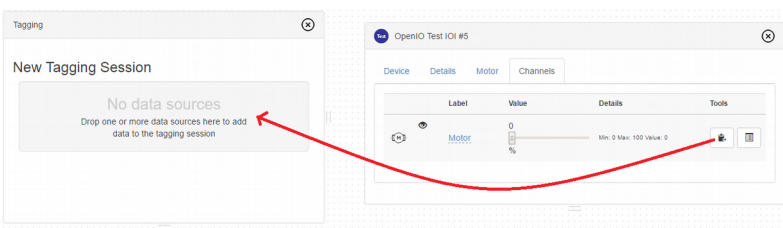
Here we see the New Capture Session button (in blue) and below that a number of Capture Sessions. The Capture Session in white with a tick is completed and closed. The other Capture Sessions are active and live as indicated by the icon in the image.

To create a new Capture Session, simply click on the New Capture Session button, and you will see a new pop-up appear on the screen like this one.



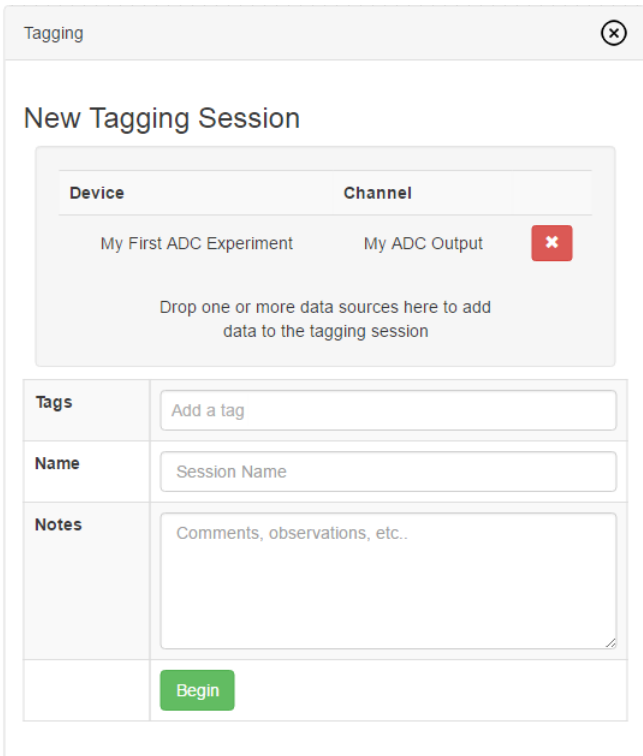
This indicates that we are about to start a Capture Session, but first, we must supply some data sources that we can use for the Capture Session.

To do this we will need to have an active IOI from which we will use as the data source.



As we can see in the screen shot above, we have a data source for which we can see the channels tab active. To start the Capture Session, we must drag the data source into the Capture Session window area that is shaded in grey, and as indicated with the arrow.

When we do this, we will see the following in the browser.



The screenshot shows a browser window titled 'Tagging' with a close button. Inside, the 'New Tagging Session' form is displayed. At the top, there is a table with two columns: 'Device' and 'Channel'. The 'Device' column contains 'My First ADC Experiment' and the 'Channel' column contains 'My ADC Output'. To the right of the 'Channel' column is a red square button with a white 'x'. Below this table is a grey shaded area with the text 'Drop one or more data sources here to add data to the tagging session'. Below the grey area are three input fields: 'Tags' with a placeholder 'Add a tag', 'Name' with a placeholder 'Session Name', and 'Notes' with a placeholder 'Comments, observations, etc..'. At the bottom of the form is a green 'Begin' button.

We can see the data source that we are using (at the top), The Tags entry area, the Capture Session name and some notes.

Although we are showing a single data source here, we can add multiple data sources to the Capture Session.

Next we can add some tags. In line with the use of the tags mentioned previously, a number of tags are added. Next we add a name for the Capture Session, and finally some notes that explain what the Capture session is about and the type of data collected etc.

The screen will now look something like this.

Tagging

New Tagging Session

Device

Channel

My First ADC Experiment

My ADC Output

×

Drop one or more data sources here to add data to the tagging session

Tags

smith

project_honey_monster

experiment_21

monday_26_jan_2017

afternoon_session

interesting_observation

Add a tag

Name

My First Tagging Session

Notes

This is the first use of the `IQI` using an ADC and a ScriptML script written in Python

Begin

To start the Capture Session, simply click the Begin button. At which point the Capture Session is added to the active and inactive capture sessions.

With the Capture Session active, we can continue with the experiment as required. Once you are happy that all of the data is collected, and that you no longer need the capture session, you can click on the Capture Session Icon to open a screen similar to the one below.

Tagging Session

×

Name

My First Tagging Session

Data Sources

ADC

My ADC Output

Tags

smith

project_honey_monster

experiment_21

monday_26_jan_2017

afternoon_session

interesting_observation

Started

2017-02-09T12:53:36.000Z

Ended

LIVE!

End Session

By clicking on the End Session button, the Capture Session is closed. You will be asked whether you wish to proceed or not.

Chapter 4: Main Control Tabs

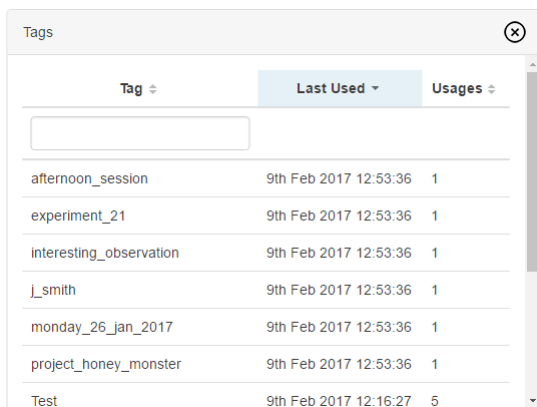
Having reviewed how we can connect an IOI, collect data and create Tags and Capture Sessions, we can now look at the Main Control Tabs that are on the right hand-side at the top of the browser window. These tabs include:

- Tags
- Report
- Search
- Widgets

We will go through each of these in turn to review their purpose and how we can use them.

Tags Tab

The first tab to consider is the Tags tab. If we click on the Tags tag, we will see a window similar to the one shown below.



Tag	Last Used	Usages
afternoon_session	9th Feb 2017 12:53:36	1
experiment_21	9th Feb 2017 12:53:36	1
interesting_observation	9th Feb 2017 12:53:36	1
j_smith	9th Feb 2017 12:53:36	1
monday_26_jan_2017	9th Feb 2017 12:53:36	1
project_honey_monster	9th Feb 2017 12:53:36	1
Test	9th Feb 2017 12:16:27	5

Here we can see the Last Used tags that we recently entered into our Capture Session. The purpose for the Tags tab is so we can locate which Tags we have used in the case where we have forgotten. We can sort the Tags alphabetically, based on last usage, based on number of uses, or we can search for the tag. The search option (the empty box below the Tag button) is handy if we can remember a part of the tag we are looking for. Enter a partial list of characters you remember and the server will list all of the matching Tags.

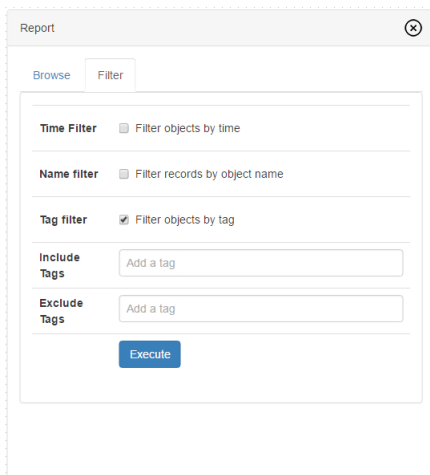
Once we have identified the Tags that we want to use, we can now consider the Report Tab and using the Tag identifier to recover data from the report.

Report Tab

Now that we know about Tags and Capture Sessions, and assuming that we have collected some data, we can now start to review and export the data, if required.

Within the OpenIO Labs system, a Report is one of the best ways of retrieving the data. A Report is basically a set of criteria with which the database is queried, and the resulting matches presented to the user. From the Report view you can view the data, plot the data, download the data, depending on the type of data that has been recorded.

If we click on the Report Tab we will see a window similar to the one below.

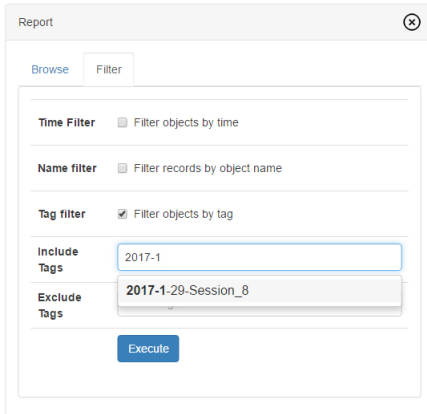


From the Report Tab, there are two views. A Browse view and a Filter view.

The Browse view allows you to view all of the Reports as a time sequence with the latest report presented first. You can scroll through the reports based on time, until you find the report that you are interested in.

Once you have found the report you are looking for you can view it locally, or you can download the report for further processing.

The alternative way of viewing reports is to use the Filter View. If we imagine that we collected some data in the form of a 3D plot, and that we knew that the Tag included the date 2017-1 we can start to write the Tag into the Include Tags box. The server will then show all Tags that match the partial Tag. Next you select the Tag and click on the Execute button.



Report

Browse Filter

Time filter ☐ Filter objects by time

Name filter ☐ Filter records by object name

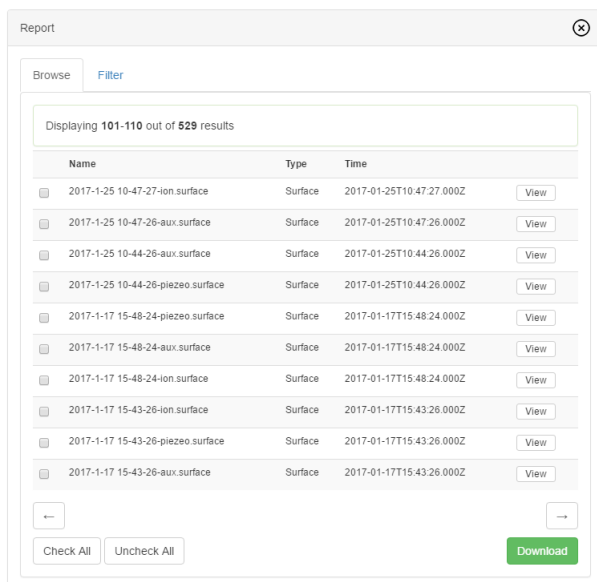
Tag filter ☒ Filter objects by tag

Include Tags

Exclude Tags

Execute

The server will then switch to the Browse view and present a list of all of the reports that use the tag in question, as shown below.



Report

Browse Filter

Displaying 101-110 out of 529 results

Name	Type	Time	
<input type="checkbox"/> 2017-1-25 10-47-27-ion.surface	Surface	2017-01-25T10:47:27.000Z	View
<input type="checkbox"/> 2017-1-25 10-47-26-aux.surface	Surface	2017-01-25T10:47:26.000Z	View
<input type="checkbox"/> 2017-1-25 10-44-26-aux.surface	Surface	2017-01-25T10:44:26.000Z	View
<input type="checkbox"/> 2017-1-25 10-44-26-piezo.surface	Surface	2017-01-25T10:44:26.000Z	View
<input type="checkbox"/> 2017-1-17 15-48-24-piezo.surface	Surface	2017-01-17T15:48:24.000Z	View
<input type="checkbox"/> 2017-1-17 15-48-24-aux.surface	Surface	2017-01-17T15:48:24.000Z	View
<input type="checkbox"/> 2017-1-17 15-48-24-ion.surface	Surface	2017-01-17T15:48:24.000Z	View
<input type="checkbox"/> 2017-1-17 15-43-26-ion.surface	Surface	2017-01-17T15:43:26.000Z	View
<input type="checkbox"/> 2017-1-17 15-43-26-piezo.surface	Surface	2017-01-17T15:43:26.000Z	View
<input type="checkbox"/> 2017-1-17 15-43-26-aux.surface	Surface	2017-01-17T15:43:26.000Z	View

← →

Check All Uncheck All [Download](#)

Next you can either View the specific report, or select the Report by clicking the Check-box on the left and click Download to copy the report to the local disk store.

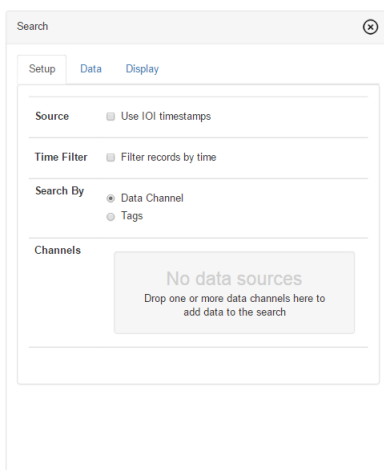
In this instance, we click on the View tab and the 3D-surface map for the collected data is displayed as shown below.



Search Tab

The next tab in the Main Controls Tab is the Search Tab. The search tab is used as an alternative to the Reports tab and allows you to search for data that has been recorded on to the server, view the data and if required download the data.

The Search tab is accessed by clicking on the tab itself. You will see a screen similar to the one below.

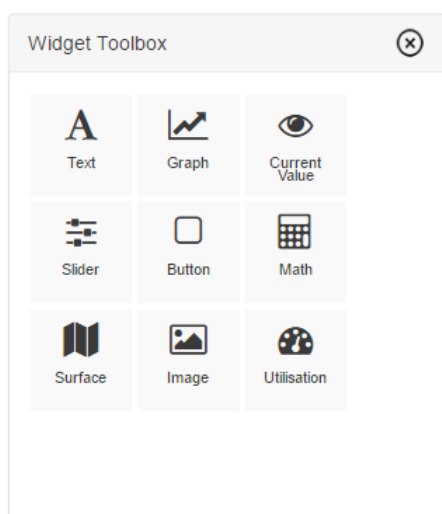


From this window we can search based on the data timestamps, Data Channel and Tags. Once the correct data source has been loaded, the data can be pulled from the database and viewed on the Data tab or displayed on the Display tab.

Widget Tab

The final tab that we will consider is the Widget tab. The widget tab provides a collection of tools that are used to help provide data viewing or additional data processing functions.

The main Widget window is illustrated below.



The purpose for each of the Widgets is considered in detail in the table below.

Widget	Purpose
Text	This widget is used to display text that was collected from the IOI devices or from a vIOI.
Graph	This widget provides a simple 2D graphing mechanisms to display data as it is received by the server.
Current Value	The Current Value widget allows you to see the current value of a specific data stream from the IOI to the server.
Slider	The Slider widget is used to allow the user to set a value that can be used by the IOI, for instance to change a threshold for some specific purpose such as an alarm state.
Button	The Button widget is used as a type of Boolean ON/OFF function that can be used to control a device or sensor.
Math	The Math widget is used to provide some simple mathematical processing of the received data.
Surface	The Surface widget is used to plot 3D data received from the IOI.

Image	The Image widget is used to display an image received from the IOI from e.g. a camera.
Utilisation	The Utilisation widget is used to allow you to understand how much a specific device is used.

Summary of the Widgets Available from the Widget Tab

In this section we won't go into the detail of how the Widgets are used. Instead in the following Chapter we will consider a typical usage scenario to show how all of the pieces of the system work together including the Widget Tab.

Chapter 5: Example Use Case

We will finish this User Guide by stepping through an example based on a specific type of device. In this instance we will be using an OpenIO Labs GP-2 IOI with a Digital-to-Analogue Converter (DAC) connected to an Analogue-to-Digital-Converter (ADC). We will use this to simulate a motor control scenario, where we are observing the drive voltage applied to the motor over time.

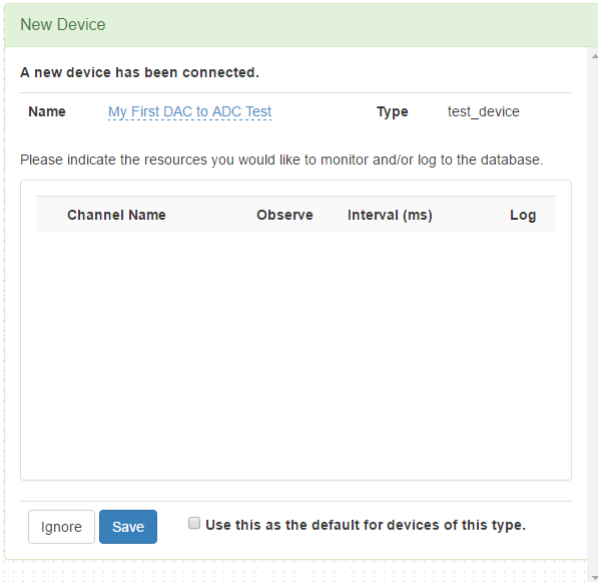
Initial Setup And Configuration

In this example we will be using a Maxim MAX518 8-bit DAC and an Analog Devices AD7991 12-bit ADC. In addition, we will be using a script to drive a signal to the DAC using an I2C interface. The analogue output voltage from the DAC is connected to the analogue input of the ADC.

The DAC is an 8-bit device, so the digital input will range between 0 and 255. The ADC is a 12-bit device, and so the digital output will range between 0 and 4095. Both devices use a 5 Volts power supply, so a DAC set to 255 level will correspond to +5 Volts, and similarly for a 5 Volt input, the ADC will measure a digital value of 4095.

We will correctly wire up the ADC and DAC such the voltages and the I2C clock and data lines are correctly attached to the GP-2 IOI. One point, for the GP-2 with the I2C, we need to ensure that we are using a 2.2k Ohm pull-up resistor on the clock and data lines.

When we power on the device, the IOI will start the client application and we will see the following on the Browser screen. Note we have amended the name of the Device from the server provided default, to something more meaningful.



New Device

A new device has been connected.

Name	My First DAC to ADC Test	Type	test_device
------	--------------------------	------	-------------

Please indicate the resources you would like to monitor and/or log to the database.

Channel Name	Observe	Interval (ms)	Log
--------------	---------	---------------	-----

☐ Use this as the default for devices of this type.

When we are happy with the name and we have the correct device we can click the Save button. The device is now active and registered with the server.

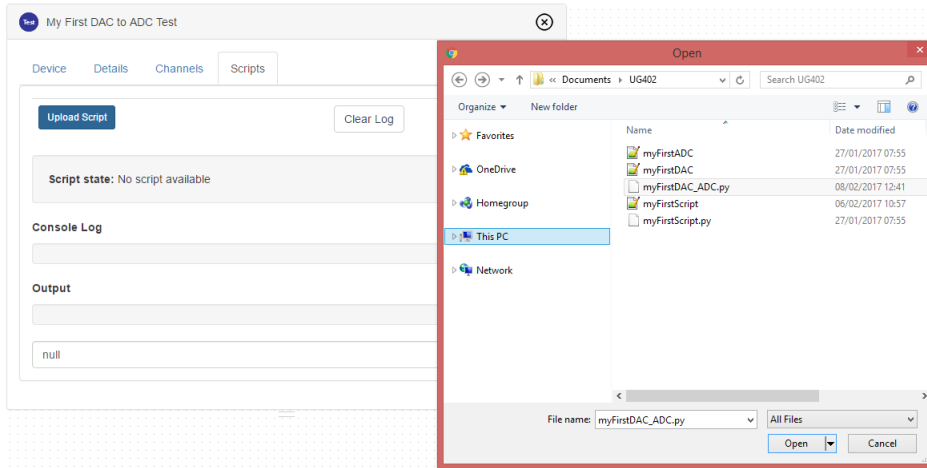
Loading Our First ScriptML Script

In this instance we are going to use ScriptML to drive the DAC, to extract data from the ADC, and to push the data up to the server.

ScriptML was created by OpenIO Labs to allow the users full control of the devices attached to the IOIs but using a language of their choice. Currently OpenIO Labs supports C and Python, although other languages such as Java, C++ and Golang can be easily added.

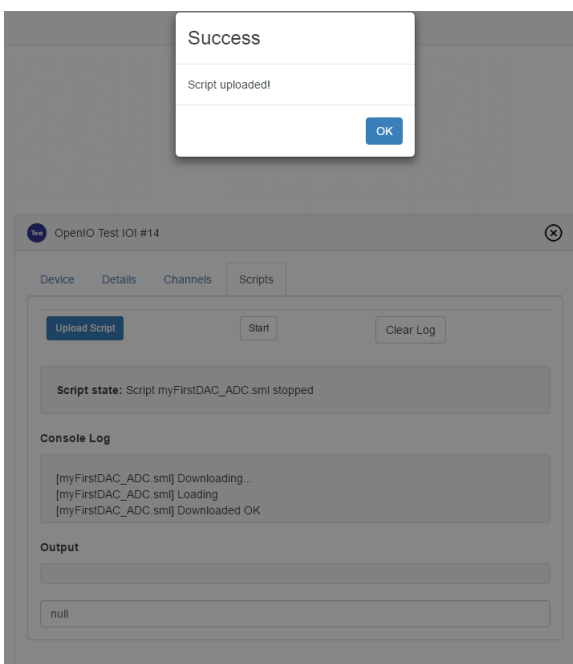
The philosophy behind ScriptML is that the user writes a script that implements the application they need for their experiment, but using a language of their choice. The script is then uploaded to the server where it is converted from the source language into the ScriptML meta language which is then downloaded to the IOI where it can be executed. For more details on ScriptML and how it can be used refer to Ref [1, 4].

When we clicked on the Save button, the Device window appeared. We can now select the Scripts tab, and then the Upload Scripts button, from where we can select our first script to run as shown below.



In this example, we are choosing myFirstDAC_ADC.py which is the Python script that will drive the DAC and collect data from the ADC and subsequently push this to the server.

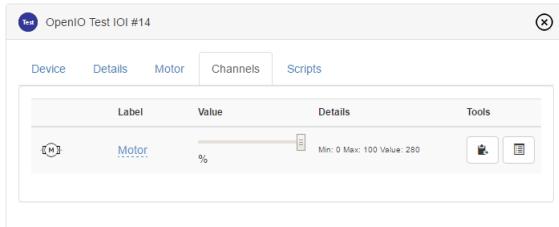
If we click Open, the Script will be uploaded, translated into ScriptML and then pushed down to the GP-2 device. We should see something like this on the browser.



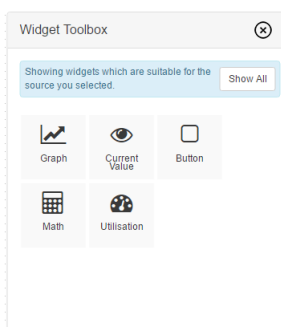
After clicking OK, we can now start the script by clicking the Start button.

Running the Script and Using Widgets

Once the script is running we can click on the Channels tab where we will see a screen as follows.

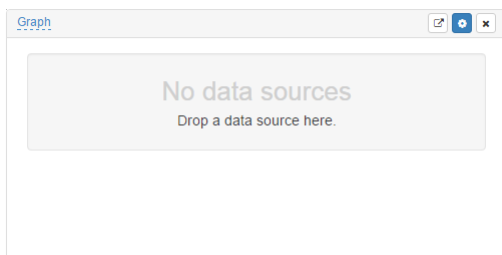


Next we click on the Widget button on the channels tab, or the Widget tab in the Main Control Tab section. We should see something like this.



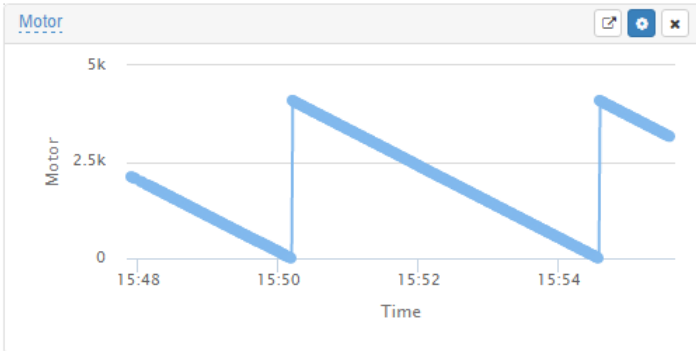
Using the Graph Widget

First we are going to use the Graph Widget, so click on this and we should see something like this.



We now need to add the data source to the Graph Widget, and also adjust the time-axis. To add the data source, go back to the window with the Device window with the Channels Tab, and drag the second icon from the right (looks like a clipboard) and drop it into the Graph Widget Data Source area. Next click on the Graph Widget settings icon and change the Data Window length field from 10 seconds to 1000 seconds.

If we let the system run for a few minutes, we will have collected sufficient points to see the DAC reach its maximum level and then reset back to zero. The output from the ADC is collected within the script and pushed to the server for display in the Graph Widget. We should see something like this.

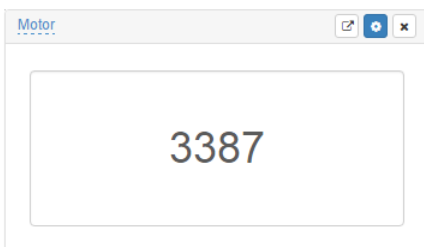


This is showing the output from the ADC driven by the DAC.

Using the Current Value Widget

Next to get a display of the current value as it changes over time, go back to the Widget window we saw earlier and click on the Current Value icon. A Current Value window will appear. As before, drop the data source from the Channels tab in the Device window into the data source section of the Current Value widget.

We should now see something like this, with the value changing slowly over time.

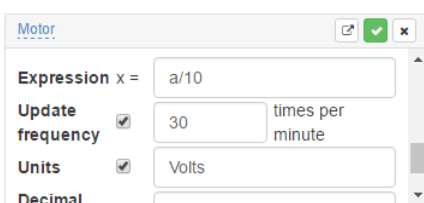


Using the Math Widget

Next we can explore using the Math Widget. With the Math Widget we can apply simple mathematical expressions to the data.

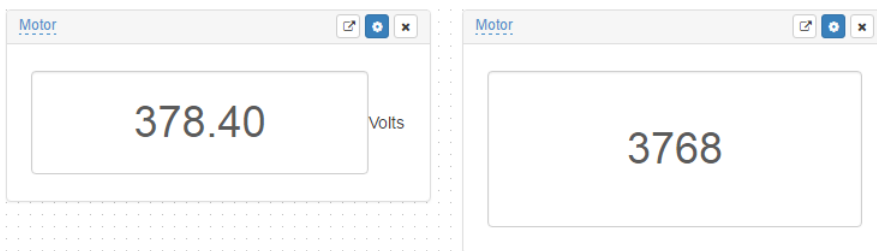
As before click on the Math Widget from the Widget window. When the Math Widget appears, drop the data source into the data source area in the Math Widget as we did for the Graph Widget.

Next click on the setup icon on the Math Widget and adjust the settings as we see below.



Here we have added an expression $a/10$ where "a" is the value from the ADC. We also set the update frequency to 30 times a minute (once every 2 seconds), set the Units to Volts and the number of decimal places to 2.

When we click the green tick we will see something like this, with the Current Value widget along-side.

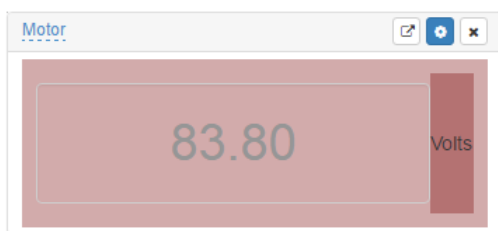


Here we can see the Math widget on the left and the Current Value widget on the right. The Math widget is showing a value of the current value divided by 10 as we desired.

Finally, we can setup an Alarm, we will assume that we want an alarm to occur whenever the Math widget value drops below 100.0 Volts.

To do this, click on the settings icon again, and scroll down to the Alarms field. For the first entry select the "less-than" symbol and for the second field a value of 100.0.

This will cause the Math widget to raise an alarm whenever the signal goes below 100 Volts. We can see this happening as shown below, where the Math widget will flash red whenever the value it calculates drops below 100.



Additional Resources

References

The following documents and references are cited within this guide:

1. UG402 – ScriptML Examples in C and Python
2. UG101 – OpenIO Labs System Architecture
3. LWM2M – See https://en.wikipedia.org/wiki/OMA_LWM2M
4. UG401 – ScriptML System Overview