

# B: Raspberry Pi Python Client

## Prerequisites:

- [Installing the Java Runtime Environment](#)
- [Installing Ignition](#)
- [Installing the following MQTT Modules](#)
  - MQTT Distributor
    - v3.1.0 or greater if using Ignition 7.9.X
  - MQTT Engine
    - v3.1.0 or greater if using Ignition 7.9.X
- Downloading the [Sparkplug Sample Code](#) onto a development system

## Overview:

Sparkplug is an open source project developed by Cirrus Link Solutions which shows how devices or projects can be enabled to communicate with MQTT Engine and Ignition. This example will show how data can be published via MQTT from an emulated device running on a development machine. In addition, it will show how devices or projects can be controlled by writing to tags in Ignition. It will also show the caveats associated with establishing /ending an MQTT session and ensuring that the tag values in Ignition are valid.

## Raspberry Pi Python Client:

This tutorial assumes:

- Ignition is running and in active trial mode or using a purchased license.
- MQTT Distributor is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- MQTT Engine is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- You have a [Raspberry Pi 2 Model B](#) running Raspbian Linux.
- You have a [Pibrella I/O board](#).

In order for this example to run you must have [Python](#) and [pip](#) installed on your Raspberry Pi. These should be installed by default.

Use pip to install the [protobuf](#) and [paho-mqtt](#) dependencies with the following commands:

```
sudo pip install protobuf
sudo pip install paho-mqtt
```

Also, let's set up some directories to run in:

```
mkdir ~/dev
mkdir ~/dev/client_lib
```

With the Sparkplug sample code downloaded onto your development machine, copy the following files to the Raspberry Pi:

```
sparkplug_b/raspberry_pi_examples/python/example.py
client_libraries/python/*
```

NOTE: The following instructions copy the files to the directories we created above.

For **Windows** the WinSCP application (<https://winscp.net/eng/download.php>) can be installed to connect and transfer files to the Raspberry Pi:

For **Linux/Mac**, the following commands can be used to copy each file:

```
scp sparkplug_b/raspberry_pi_examples/python/example.py pi@[RASP_PI_IP_ADDRESS]:~/dev/
scp client_libraries/python/* pi@[RASP_PI_IP_ADDRESS]:~/dev/client_lib/
```

Example if the Raspberry Pi IP address is 192.168.1.100 (also the default password for the 'pi' user in Raspbian is 'raspberrypi'):

```
scp sparkplug_b/raspberry_pi_examples/python/raspberry_pi.py pi@192.168.1.100:~/tmp/
scp client_libraries/python/* pi@192.168.1.100:~/dev/client_lib/
```

This example assumes the MQTT Server running is MQTT Distributor running with its default configuration. Since the IP address of the machine running the MQTT server (MQTT Distributor) is dependent on your network setup, the serverUrl must be changed in the main application file:

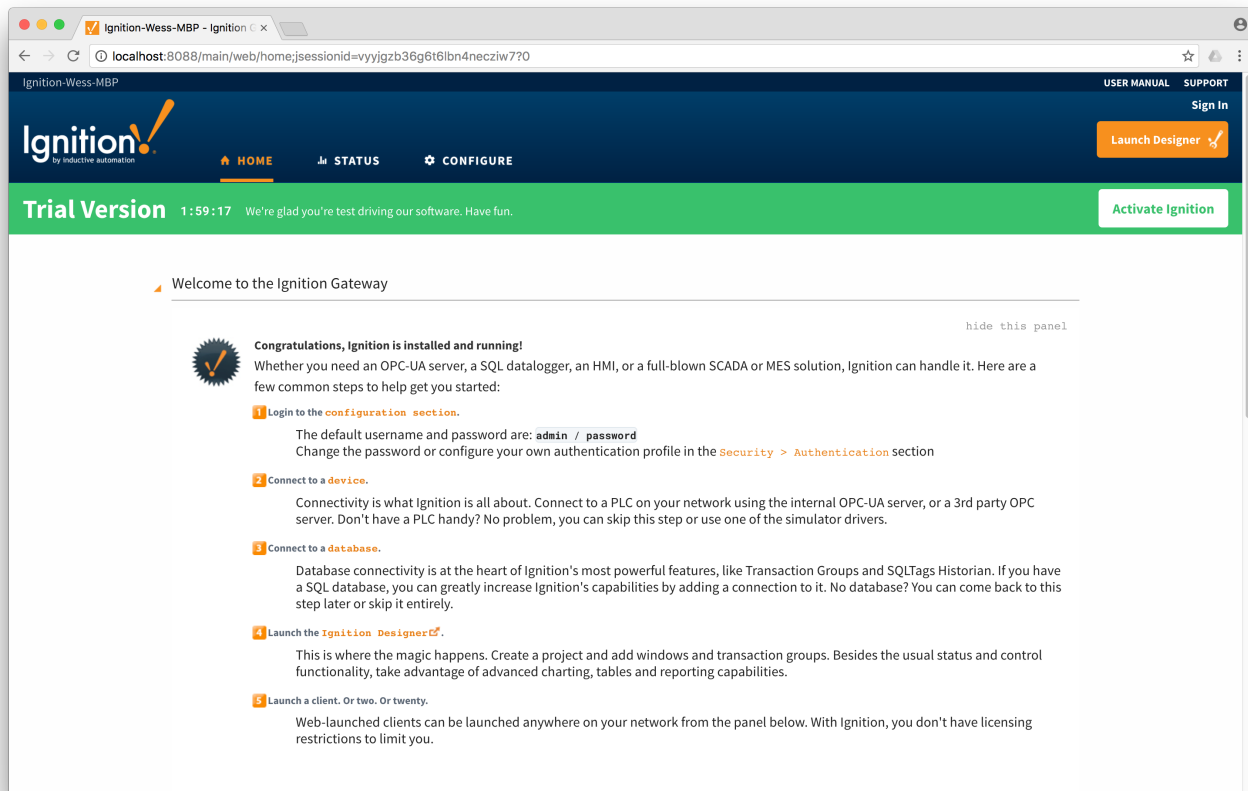
```
raspberrypi.py
```

Also, if not using MQTT Distributor you may need to modify the username and password. For simplicity this example does not use or support TLS over MQTT without modifications.

With the above steps completed, run the application with the following commands:

```
cd ~/dev/  
python example.py
```

At this point, the application will start, connect to the MQTT server, publish a Edge Node Birth Certificate, publish a Device Birth Certificate, and begin periodically reporting random data values to Ignition via MQTT Engine. This can be verified via Ignition Designer. Using a Web Browser, browse to the Ignition Gateway on your Ignition Gateway. If it is running on your development machine, that is: <http://localhost:8088>. You should see this:



Near the upper right corner, click 'Launch Designer'. This will open the following window after downloading the .jnlp file and executing it. Note the default username/password is admin/password. Type those into the appropriate fields and click 'Login'.



The image shows a login window for Ignition Designer by Inductive Automation. The window has a dark blue background with a hexagonal pattern. At the top, the logo "Ignition designer" is displayed, with "by inductive automation" in smaller text below "Ignition". An orange wrench icon is positioned between "Ignition" and "designer". Below the logo, a horizontal line separates the header from the main content area. On the left side of the main area, there is a text block: "Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying [license agreement](#)". To the right of this text, there are two input fields: "Username" and "Password". Below these fields are two buttons: "Login" and "Cancel". At the bottom of the window, there is a footer bar. On the left side of the footer, it says "Copyright © 2003–2016 Inductive Automation All rights reserved." On the right side of the footer, there is the Inductive Automation logo, which consists of a gear icon and the text "inductive automation".

Ignition designer  
by inductive automation

Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying [license agreement](#)

Username

Password

Login Cancel


Copyright © 2003–2016 Inductive Automation All rights reserved. inductive automation

This will bring you to a new Window where you can select an Ignition Project or create a new one. Create a new project by giving it a name and clicking 'Create New Project'.

Open/Create Project

**Ignition!**  
by inductive automation

**Create New**

 New Project

**Open Recent**

**New Project Setup**

Project Name  ✓

Project Title

Authentication Profile


Default Database

Default Tags Provider

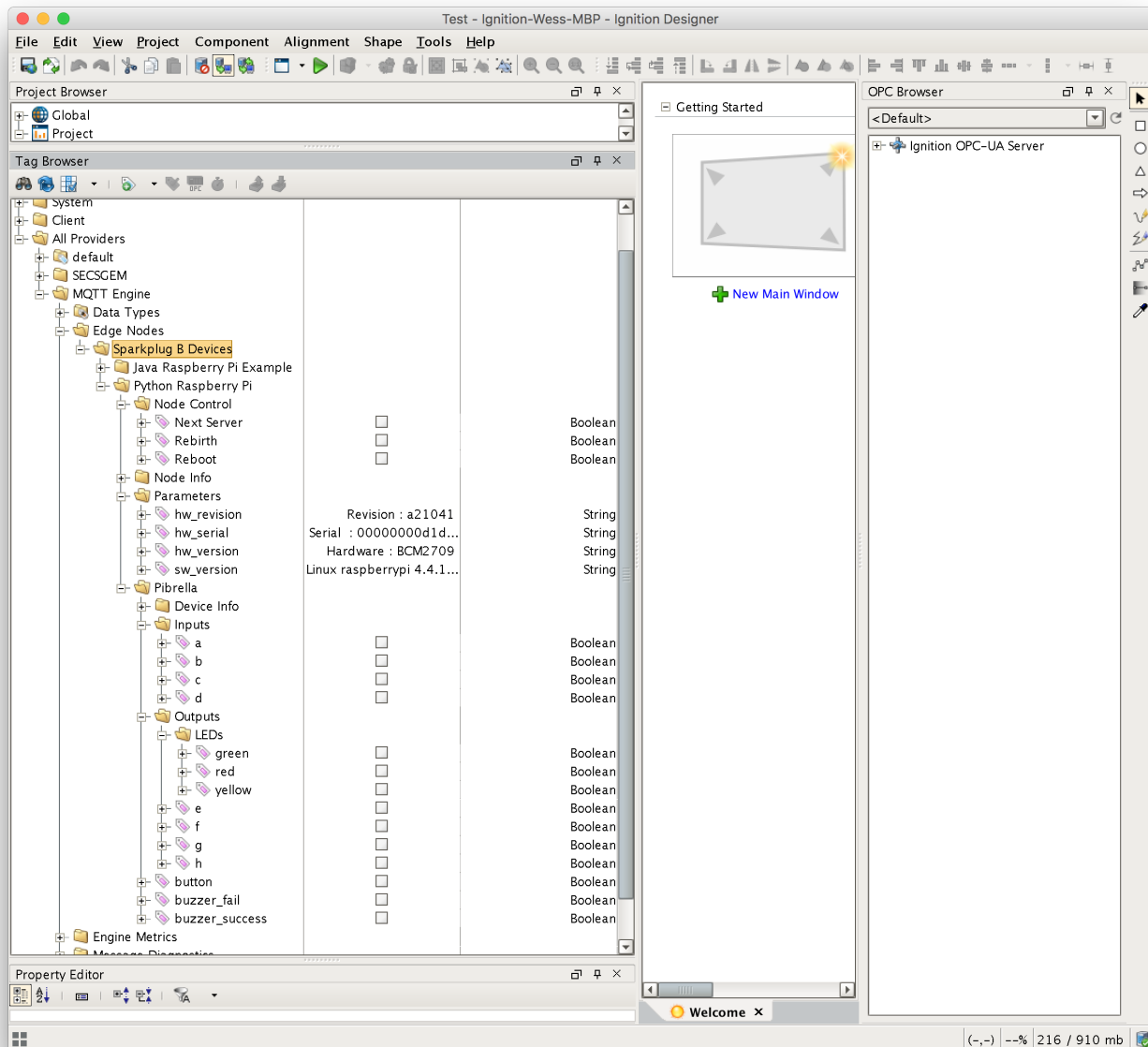
Project Template

Description

Create New Project



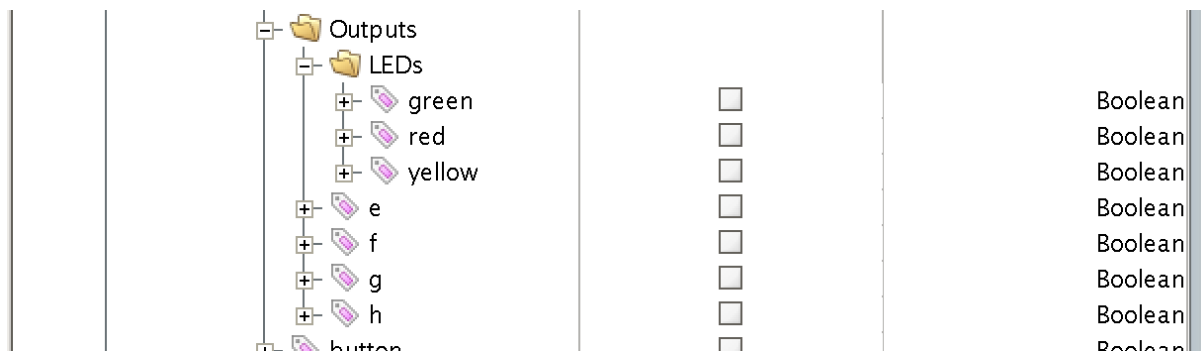
Now you should be in designer. In the left hand side of the main window is a 'Tag Browser' window. In it, expand 'All Providers -> MQTT Engine -> Edge Nodes -> Sparkplug B Devices -> Python Raspberry Pi -> Pibrella'. You should see the following



You will see that MQTT Engine saw a new device attach to the MQTT Server and publish a Birth Certificate. As a result, MQTT Engine created the Ignition Tags shown above. These are also dynamically updated as the values change. You can also write to the outputs after you [Enable Device Writes from Ignition](#). This can be done by putting designer into read/write mode. Do so by clicking this button in the menu to enable read/write mode:



Then you can change any of the values on the outputs here:



When you click on one of the outputs, you should see the output on the Pibrella board change to reflect the state. Also, if any inputs change on the Pibrella you should see them update in the Tag Browser in the Ignition Designer. If you are not seeing the output and input values update to reflect the change you have made, make sure MQTT Engine is not configured to block outbound device tag writes as described [here](#).