CAYENNE IOT WITH ESP8266

The ***Cayenne MQTT API*** is used to connect any device that you have with the Cayenne Cloud. After connecting your device, you can send data from your device to the Cayenne dashboard and display it using widgets. You may also receive commands from Cayenne, allowing remote control and automation of your devices.

There are three options for connecting with cayenne MQTT. They are:

1.**Option 1:** Use the Cayenne MQTT Libraries

2.**Option 2:** Use raw MQTT API functions

3.**Option 3:** Use HTTP to push MQTT data

Here i have opted for the first option because of the Arduino IDE. We have the MQTT Library is available directly through the Arduino IDE ***Library Manager***. But Here I attached with the repo if you need.

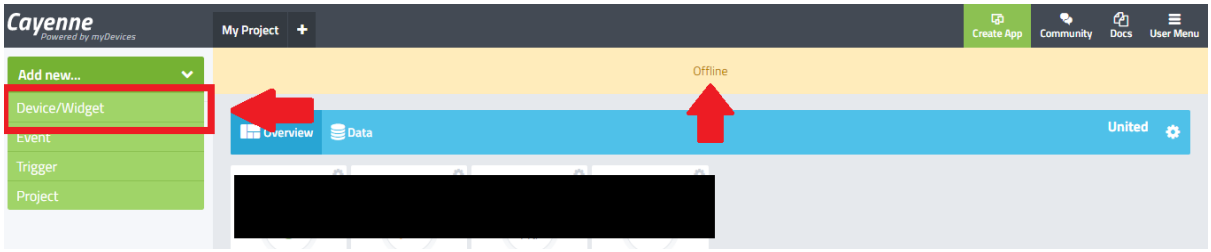
How to configure the cayenne dashboard with ESP8266?

Step 1:

I already signed in to the cayenne dashboard. Once you have created the cayenne account, make the sign-in. And now you reached at a window like this:

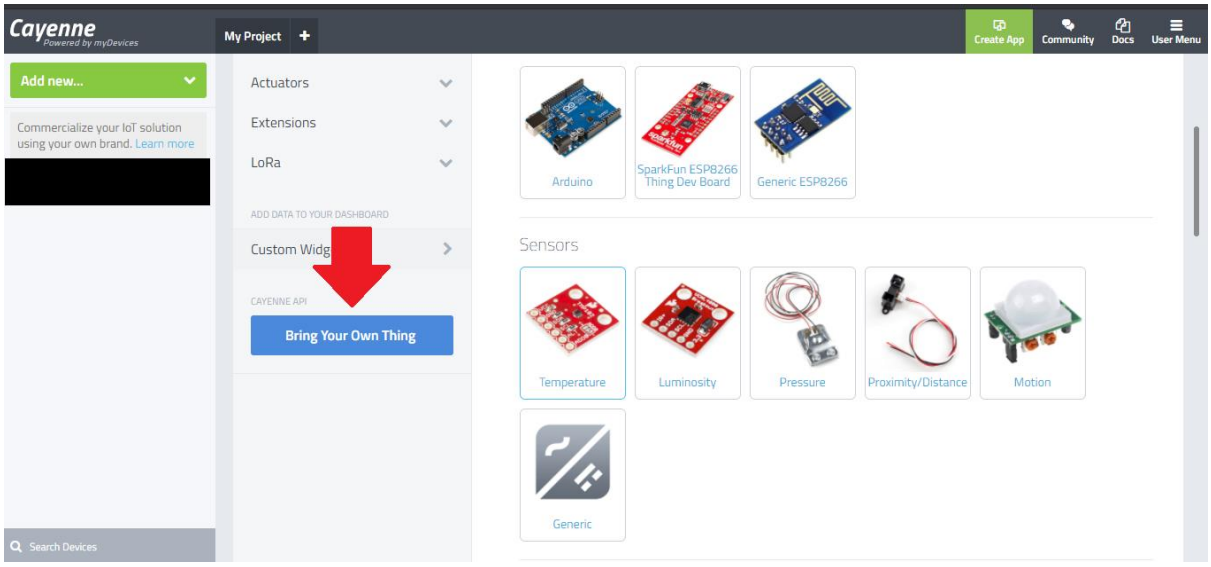
Click on ***Add new***

Notice that **offline** indication on top. If you are successfully made connection it will turn to **online.** From the drop-down list select **Device/Widget** option.



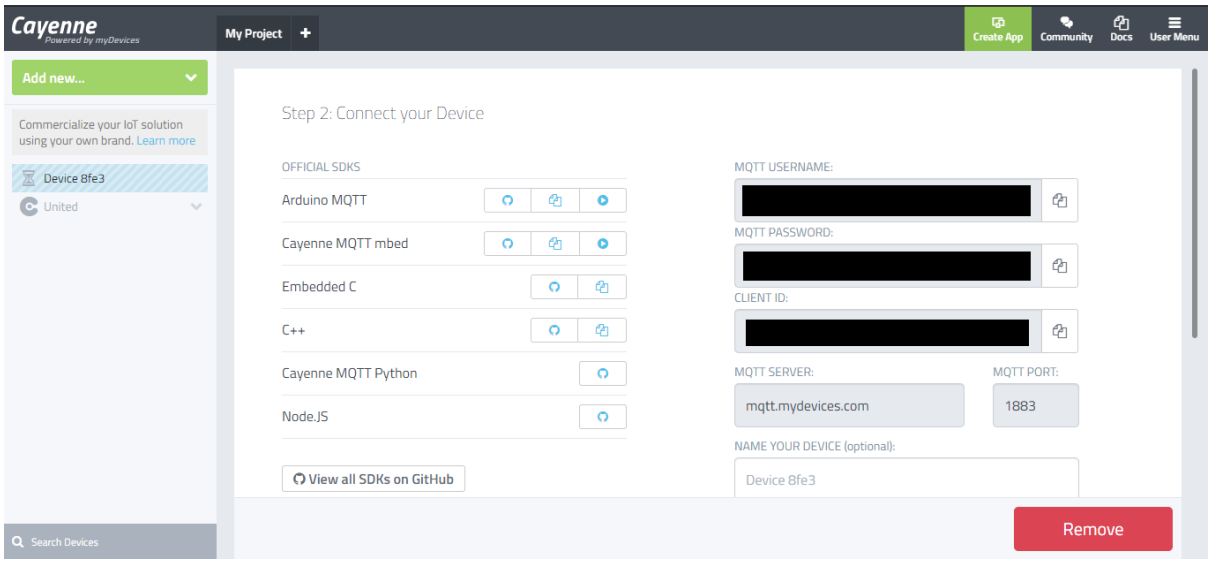
Step 2:

From the upcoming window just scroll down and you can see that **Bring Your Own Thing** option**.** Click it and do not select any other options from the panel.



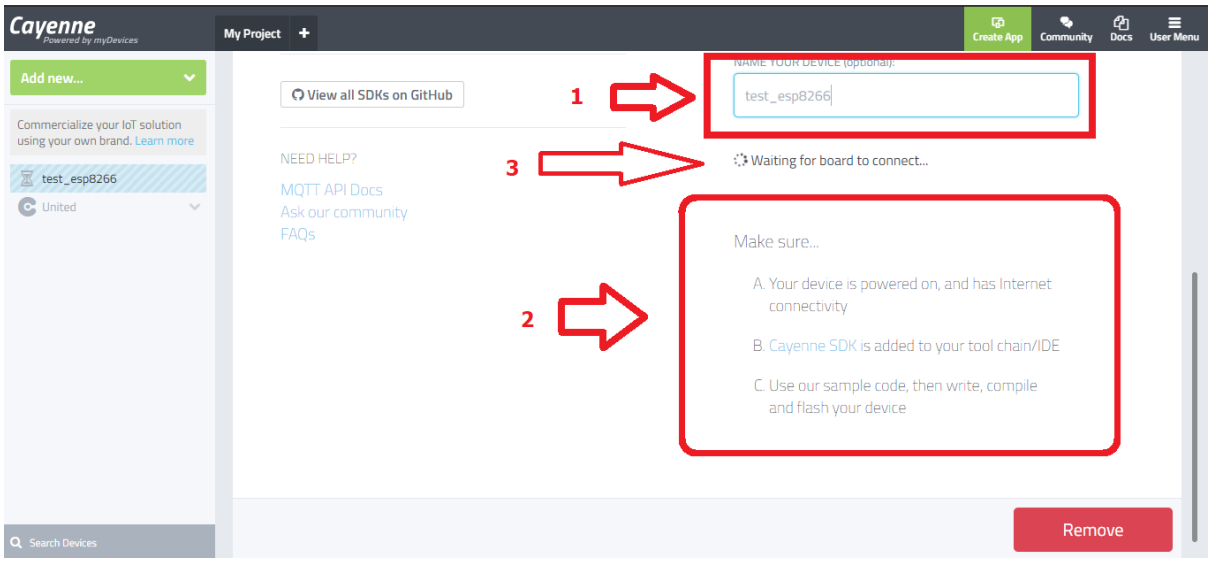
Step 3:

A new window with the credential option appears now. Copy **MQTT USERNAME**, **MQTT PASSWORD**, **CLIENT ID** from the given page for future use in source code.

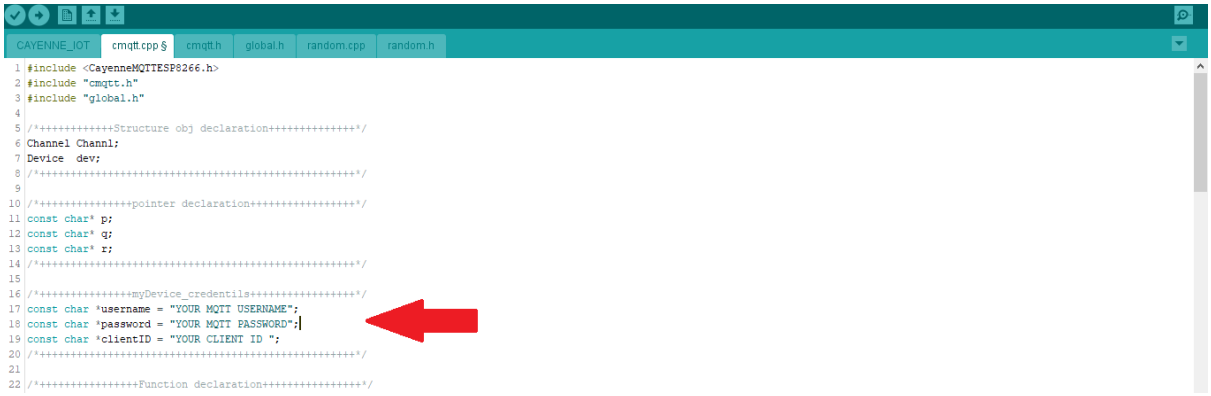


Step 4:

Change the name of your device in dashboard [Give a name that you wish to] in indication **[1].** In indication **[2]** they have suggested us to connect our ESP8266 controller with the cayenne console for further operation or procedure completion. So, let's do it:



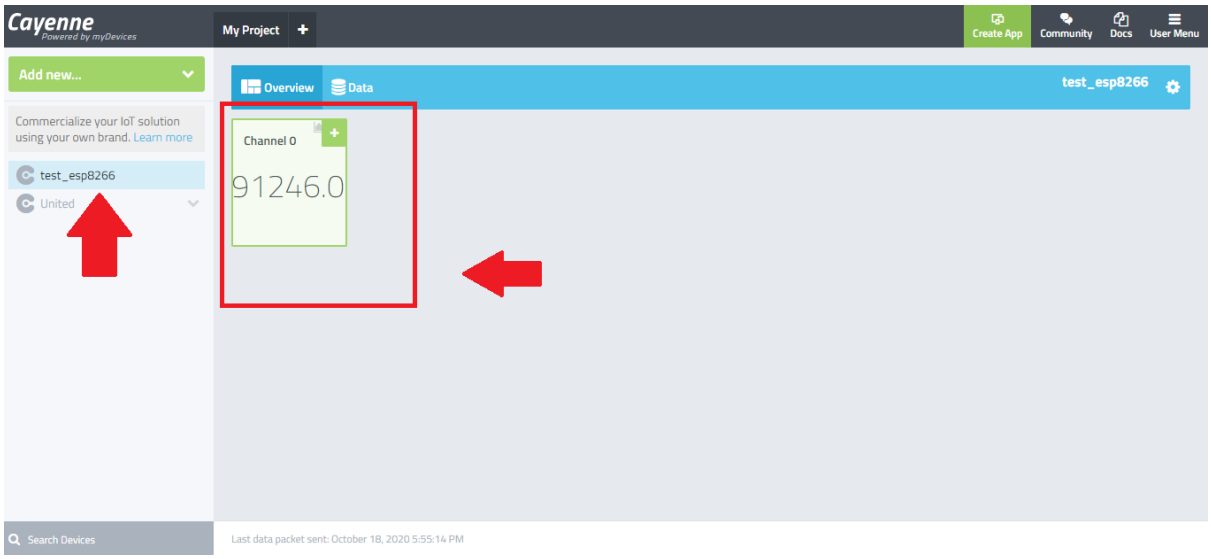
In indication **[3]** you can see that the dashboard waits to connect the ESP8266.So paste the credential that you have copied from **Step 3** in the source code section of ***cmqtt.cpp***.



Change the **ssid** and **password** of your WIFI router in ***global.h***. Then compile and upload the source code. After the uploading, restart the device once again and Wait for the dashboard to change to home page when the ESP8266 make connection.

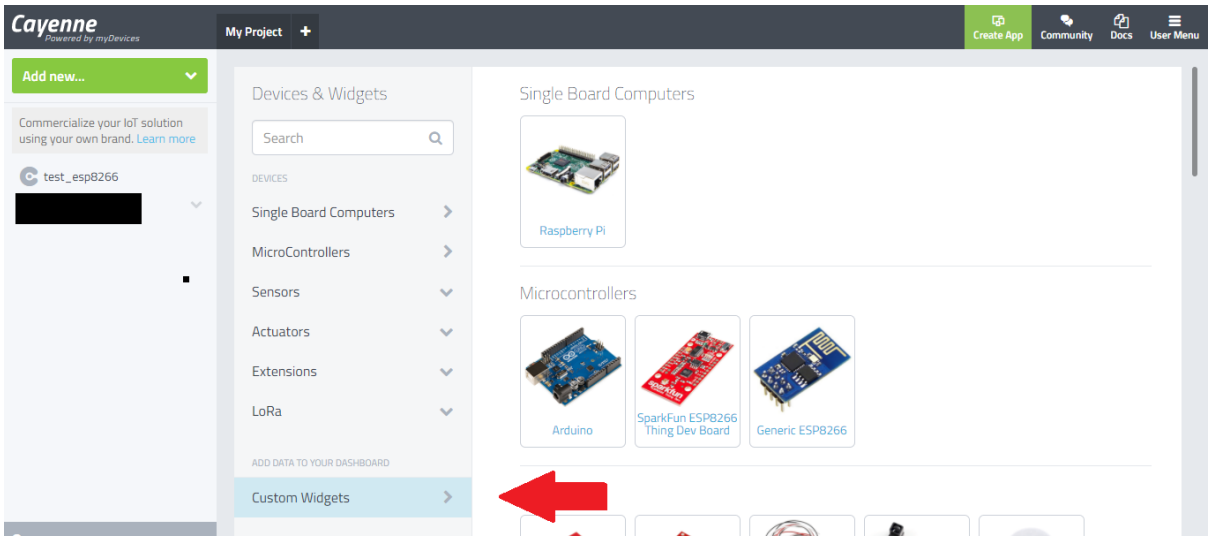
Step 5:

After the connection is successfully established the dashboard would change and goes to the home window. There we can see the device name that I created on the left side and a default sensor widget on the centre. We can customize it with our requirements.



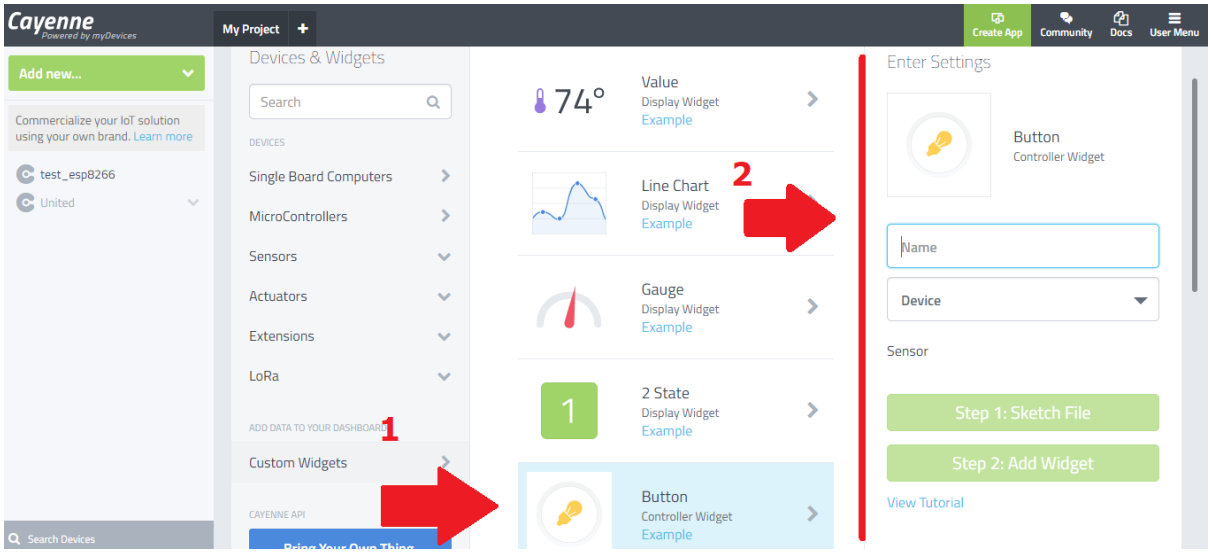
How to customize the widgets?

Here I have gone for ***a digital sensor-actuator*** system. I have 3 devices that are **led**, **bulb**, and a **fan** as actuators. For setting up these widgets we need to click on ***Add new*** ⟶ ***Devices/Widgets*** from the home menu**.** From the next window [given below] clicks on ***Custom Widgets***.

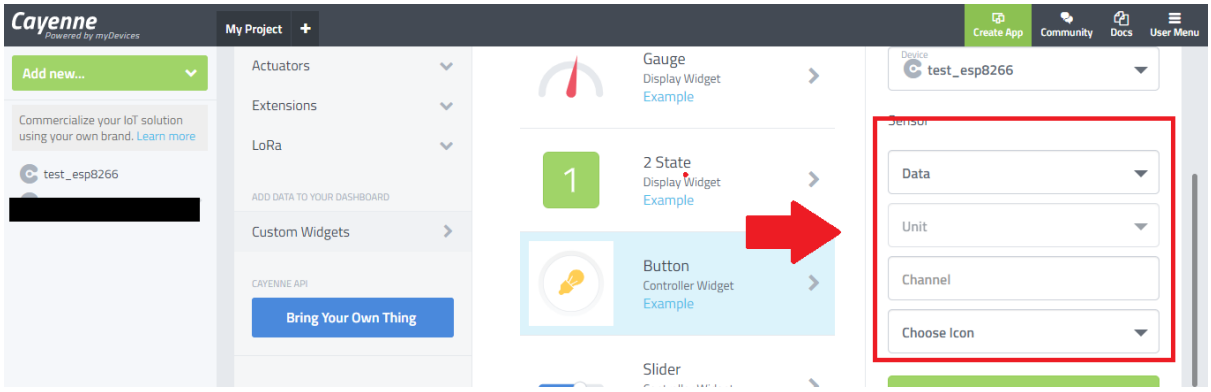


Now we can see two options labelled: Click on first one [**Button**] for setting up a ***digital actuator*** here I say my first widget is **led**. Next a sub window will appear on the right side there we need to setup the name as:

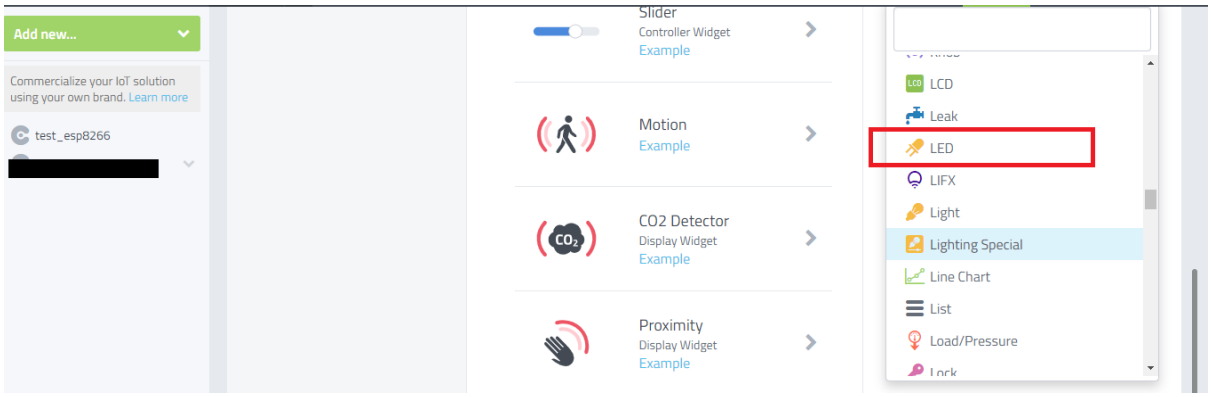
1. Name : LED
2. Device : test_esp8266 **[Select your device name]** .



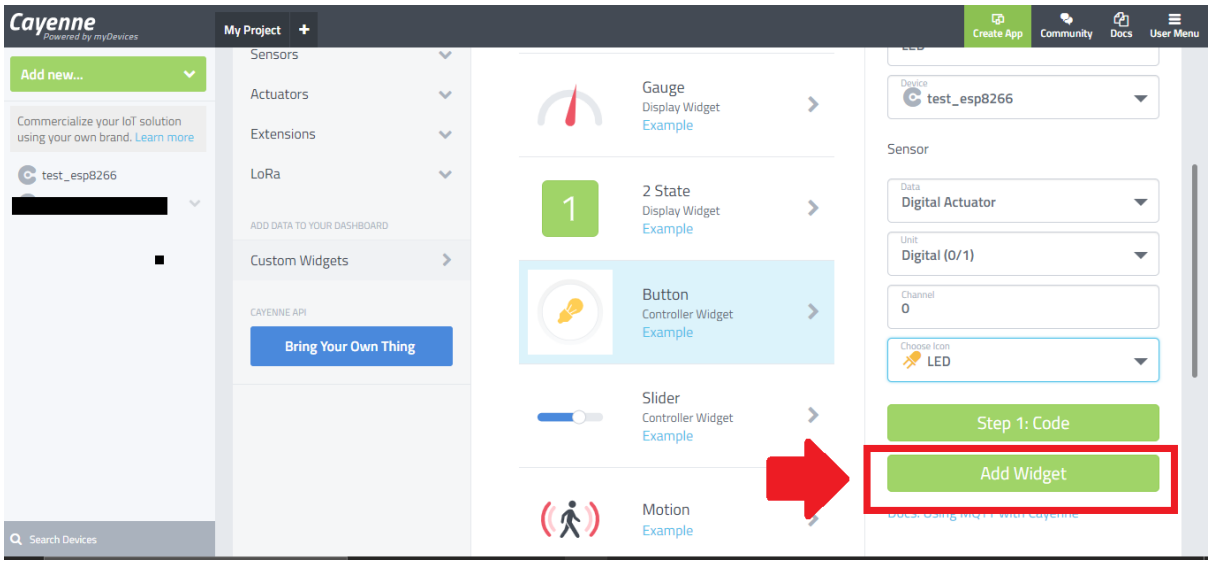
3. Data : click & select **Digital Actuator**
4. Unit : click & select **Digital (0/1)**
5. Channel : type 0 For the led **[for the bulb :1 , fan : 2]**



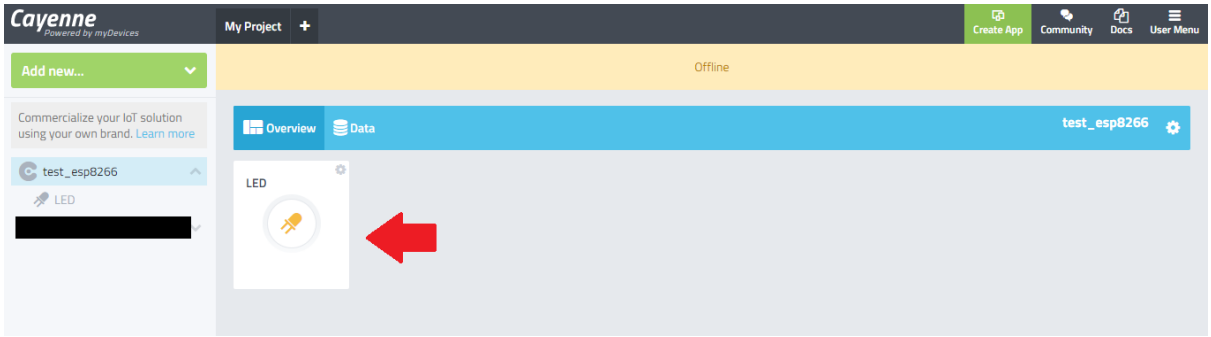
6. Choose Icon : From the drop down list please scroll down and select **LED** icon as shown.



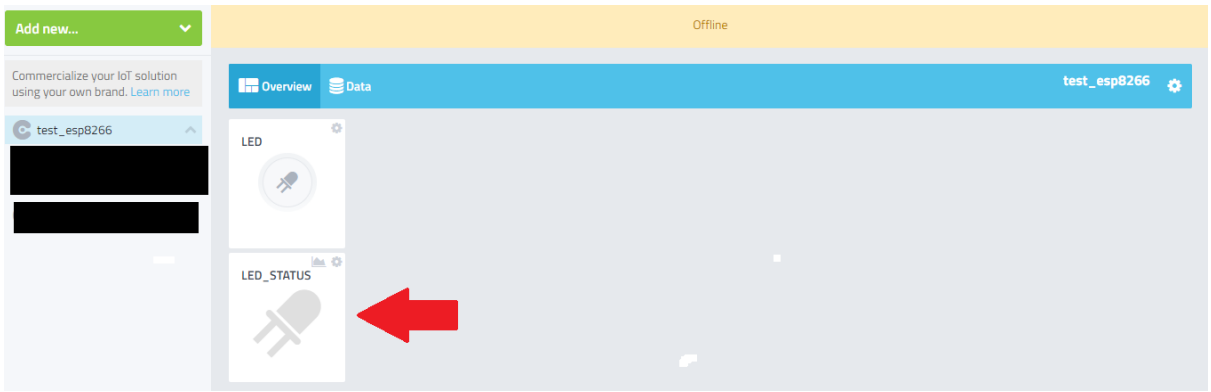
After all selections, you can see below the selected settings and the **Add Widget** button now appears.



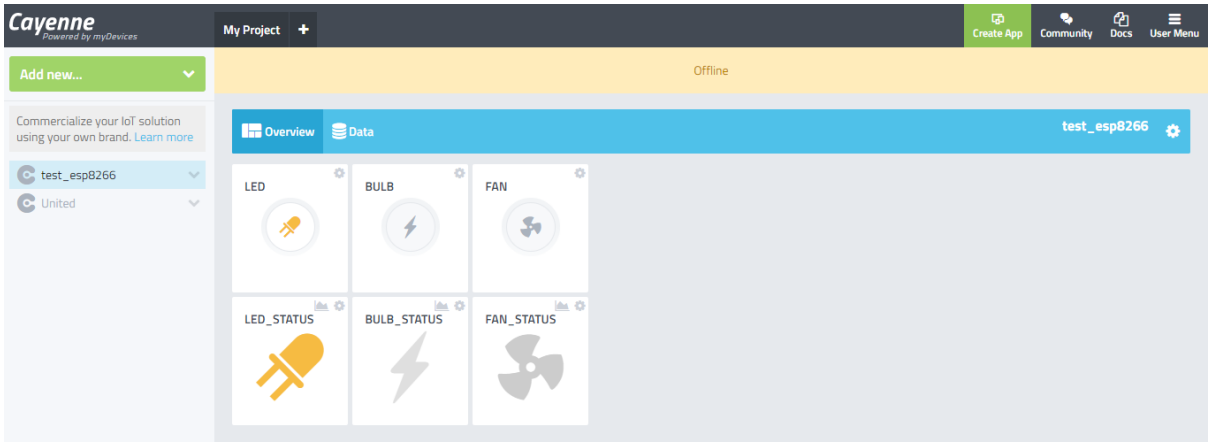
Click ***Add Widget*** and it will guide us to the home page where our newly created widget is displayed.



Below this LED widget we can create its own **status** widget for knowing whether it is ON/OFF. We can place it just below the LED widget. It can be created in a way same as we created the actuator. From ***Custom Widgets*** select **2 state** options and goes for the right-side window then edit only the **Data** portion. Change it from **Digital Actuator** to **Digital Sensor**. And click ***Add Widget***. And see the difference.



Like this we can create our all other three gadgets and their widgets in the dashboard.

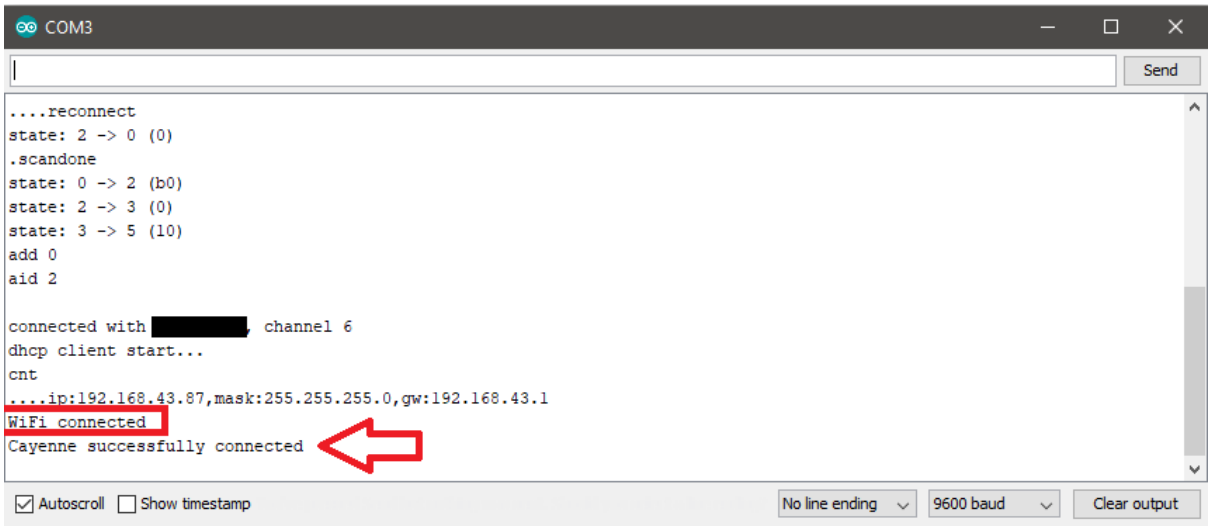


How to debug in source code?

I have already written the ***subscribe*** **(IN HANDLERS)** and ***publish*** **(OUT HANDLERS)** functions in the source code. You can find it in ***cmqtt.cpp*** file.

Initially, we get connection status if it's successfully connected to the WIFI router followed by the cayenne mqtt connection status.

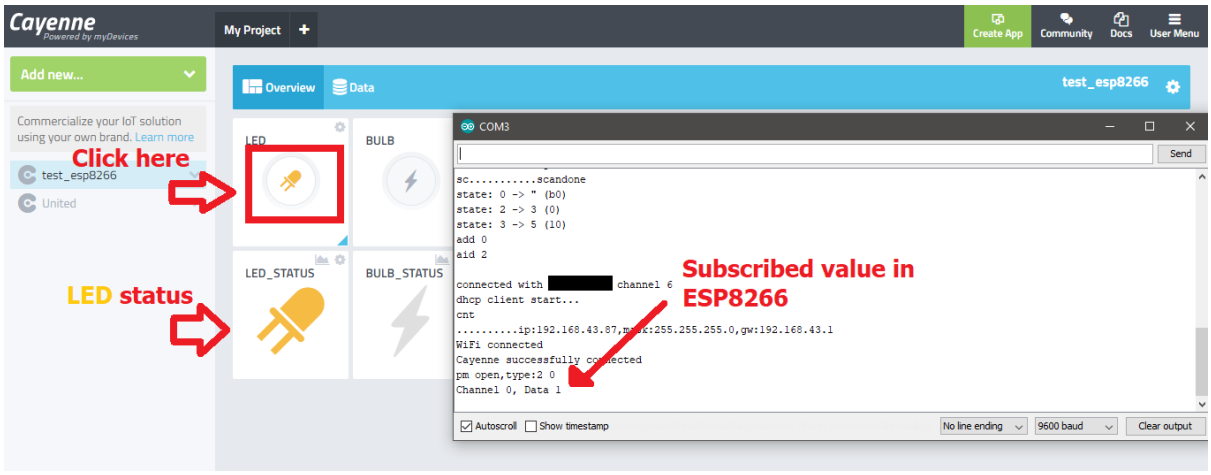
For me, I have successfully connected with cayenne mqtt so I get success status.



I have placed my devices:

- LED on **[channel 0]**
- BULB on **[channel 1]**
- FAN on **[channel 2]**

For example, when I click the **LED** button which belongs to the zeroth channel on the dashboard. The serial monitor shows its debug output as the channel where **LED** placed and digital input (**1/0**) that I have given. The serial print means I just turned ON the LED through channel zero. And also, we can see that its **status** has been changed on the dashboard.



If I click once again, it will be ***turned OFF*** and you can observe its response on the dashboard as well as in the serial monitor.

The ***subscribed*** value can be used for further extension of hardware/peripheral interfaces in ESP8266 GPIO because it's in digital state for an iot concept.

Similarly, you can try other gadgets and its status in the dashboard along with its debugged value.

THANK YOU!!!