# Prototyping an End-to-End IOT System

Arunn Balasundram

University of Moratuwa

22-08-2019

# Outline

# Introduction

# Resource required

- Laptops with :
  - Arduino IDE installed.
  - MQTT client installed
- Hardware : ESP8266(NodeMCU),LEDs,LDR,10K resistors, Breadboard,Connectors.
- Setup a free cloud Node-RED account. https://cloud.ibm.com/ or https://fred.sensetecnic.com
- A WiFi network (or hotspot).

# ESP8266

- The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (micro controller unit).
- It is very useful in several IOT applications based in WiFi.
- There are several variations of ESP8266.


Figure 1: Variations of ESP8266

- It can be programmed with Arduino IDE.

# Programming ESP8266 using Arduino IDE

- Install the latest Arduino IDE.
- Add the Arduino core using following link.
  https://github.com/esp8266/Arduino
  Boards manager link:
  http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Install the ESP8266 drivers.

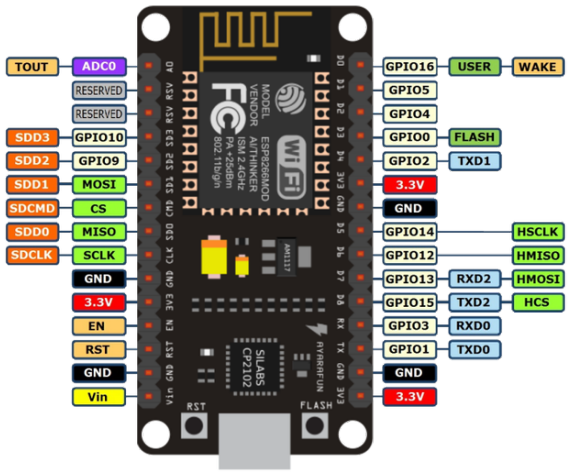



Figure 2: ESP12 breakout-NodeMCU

# Basic WiFi Applications

# Connecting to your WiFi Network

- Open a **WifiConnect.ino** Arduino Sketch from codes.
- Select board as "NodeMCU1.0"
- Edit the following lines above **setup()**.

```
#include <ESP8266WiFi.h>
const char* ssid     = "your-ssid";
const char* password = "your-password";
```

# Connecting to your WiFi Network

- Following lines inside **setup()** will connect ESP8266 to the specified WiFi Network.

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
```

- After ESP8266 connected to the network, obtained IP will be printed to the Serial.
- Load the program and Observe the serial monitor.

# Start your Own WiFi AP

- Open the **WifiApServer.ino** file from codes.
- Edit your credentials and load the program to ESP8266.
- Connect your AP from the Mobile phone and open **192.168.4.1** on your browser.

## Exercise

With the help of **WifiApServer.ino** example make a program to turn LED ON/OFF using a mobile phone.

# Scan all available networks

- Open the **WifiScan.ino** file from codes.
- Load the program to ESP8266.
- ```
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  Serial.print(WiFi.SSID(i));
  Serial.print(WiFi.RSSI(i));
  Serial.println((WiFi.encryptionType(i)
                  == ENC_TYPE_NONE)?" ":"*");
  ```
- Observe the output for details about all the available networks.
  ```

# Introduction to MQTT

# Communication protocols in Internet of things

- In the following section we will focus on following protocols:
  - MQTT
  - CoAP

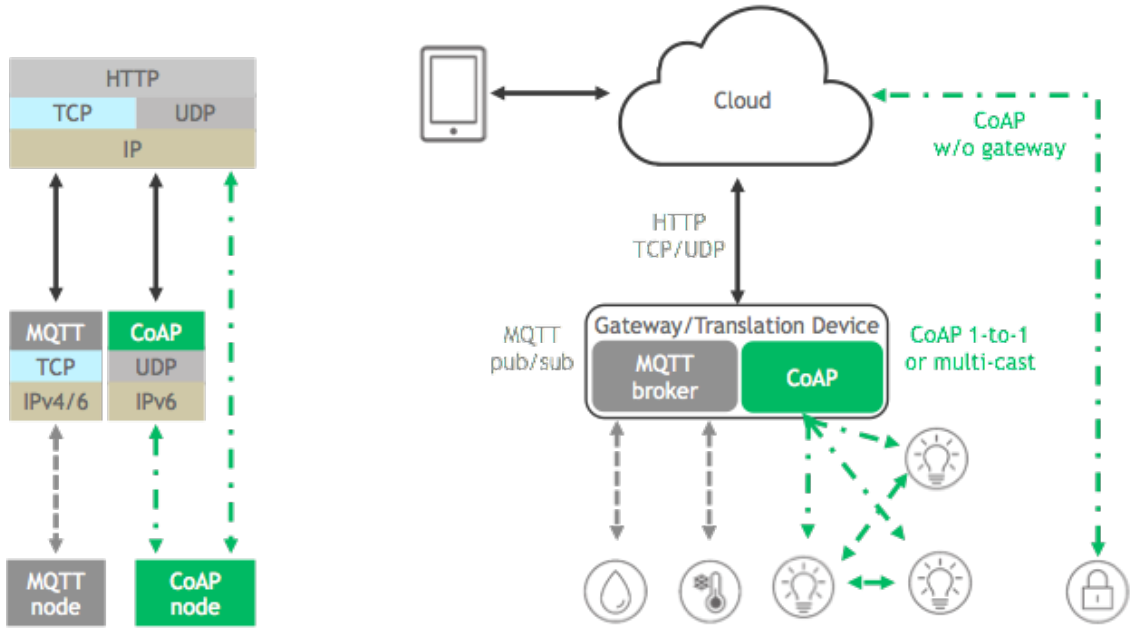



Figure 3: MQTT and CoAP in IOT

# MQTT

- MQTT in an extremely lightweight publish/subscribe messaging transport.
- It designed for constrained devices and low-bandwidth, high-latency or unreliable networks.

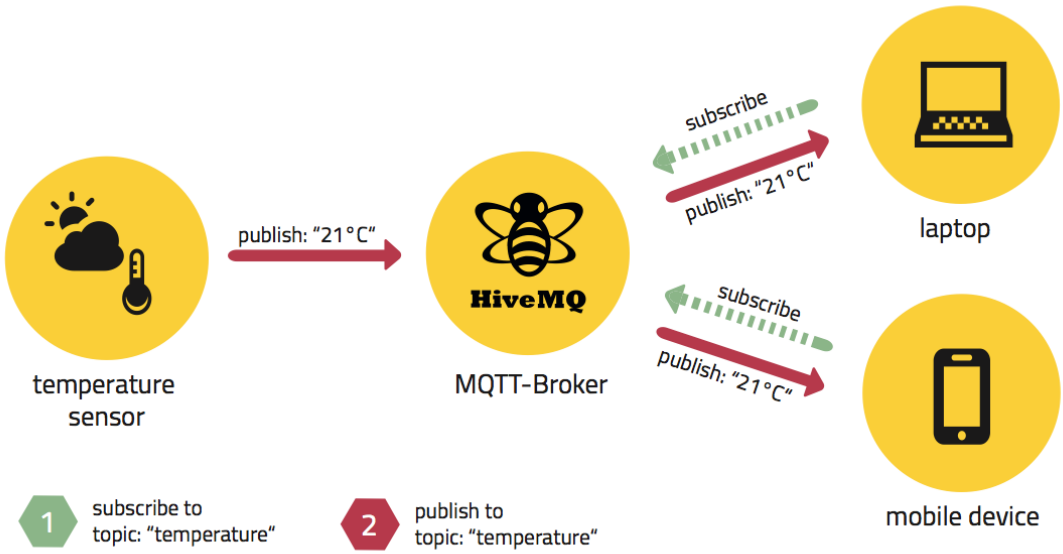



Figure 4: A simple MQTT architecture

# Sending data using MQTT

- Install a MQTT client (Eg:MQTTLens,MQTT.fx).
- We can test a simple MQTT communication using MQTTLens.
- Connect to following test MQTT broker.
  - Host : iot.eclipse.org
  - Port : 1883
- SUBCRIBE to topic : helloENTC
- PUBLISH some message to the topic : helloENTC

# MQTT on ESP8266

- Install the MQTT library for arduino. - https://github.com/knolleary/pubsubclient
- Open the basicMQTT Arduino Sketch.
- Edit the WiFi credentials on the top part of the code.
- Edit the MQTT broker details and select a topic for your group.

# Connect to a MQTT broker

- In setup() following lines set the server and callback functions.

```
client.setServer(mqttserver, 1883);
client.setCallback(callback);
```

- The reconnect() function initiate the connection process.

```
if(client.connect("clientID")) {
    Serial.println("connected");
    client.publish("outTopic", "hello world");
    client.subscribe("inTopic");
}
```

- Once connected **client.loop()** will keep the connection alive.

# PUBLISH and SUBCRIBE to topics

- **client.publish("outTopic", "hello world");** will PUBLISH the message "hello world" to topic "outTopic".
- **client.subscribe("inTopic");** will SUBCRIBE to the messages to the topic "inTopic".
- The callback() function will be triggered during incoming messages.

```
void callback(char* topic, byte* payload, unsigne
Serial.print("Message arrived [");
Serial.print(topic);
}
```

# Understanding the functionality

- The basicMQTT script send a hello message with the time every two seconds to the outTopic.
- The inbuilt LED is controlled based on the value PUBLISHed to inTopic. (Eg: if 1 is published ON the inbuilt LED)

**Exercise**

Modify **basicMQTT** example to send light level every 5 seconds and ON/OFF the LED connected to pin5 if "on"/"off" is published to inTopic. (Select separate topics which is unique for your group)

# Activity: Sending the WiFi Strength to MQTT

- Modify the WiFiScan.ino example to send the signal strengths to a MQTT topic.
- You can use a JSON message format to send the WiFi signal strengths.
  Eg: {"id":"001","Wifi1":25,"Wifi2":35,"Wifi3":36}
- Different group can collect data from different places and send to same MQTT topic so that all groups can collect data from all other group.
- This signal strength dataset will be used for the project which will be described later on this course.

# Using Node-RED in IOT

# Introduction to Node-RED

- Node-RED is a virtual tool to wiring together hardware devices, APIs and online services in a easy way. https://nodered.org
- You can run Node-RED in wide range of devices and cloud services.
- Flows can be generated by wiring up and configure nodes.

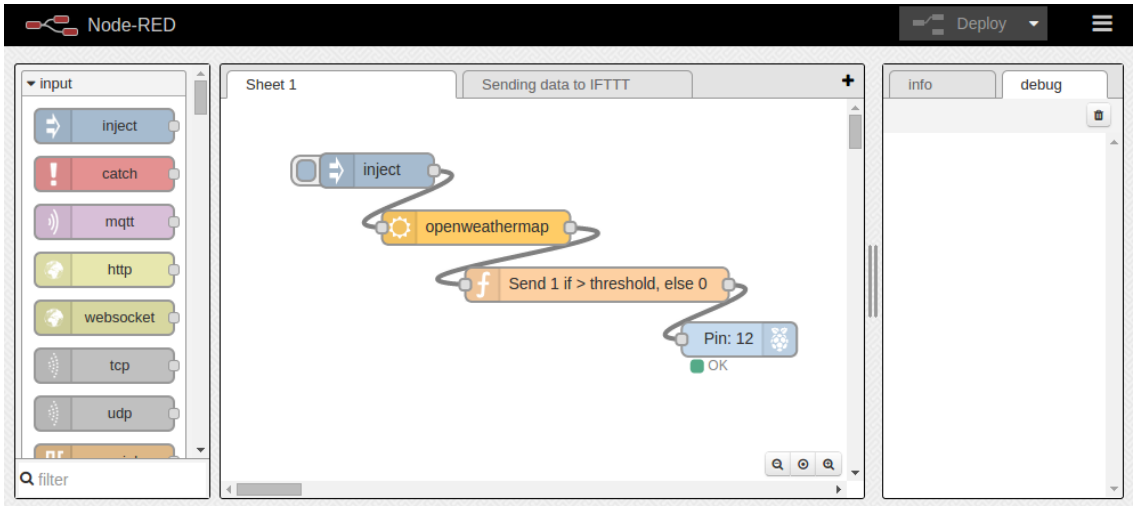



Figure 5: An example Node-RED flow

# A smart light system using Node-RED

- Using Node-RED dashboard visualize data(Graphs,Gauges,Text - outputs) or Get input (Buttons,Switch,Slider - inputs).

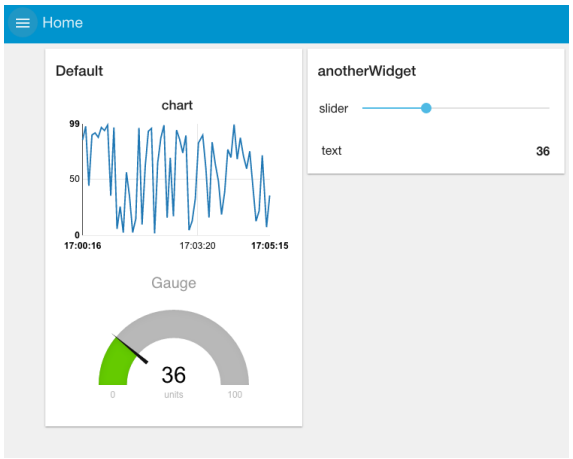


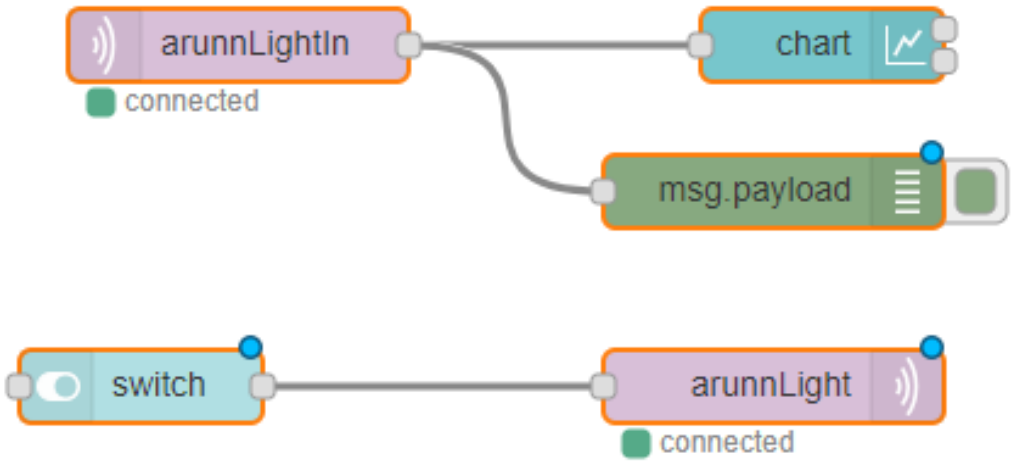

Figure 6: A simple Node-RED dashboard

## Exercise

Develop a system using Node-RED and ESP8266 which can visualize the light level of the lab in a Graph/Gauge, On/OFF a LED using a switch and Control the brightness level of LED using a slider.

# Light control system using Node-Red

- Start by creating following flows.

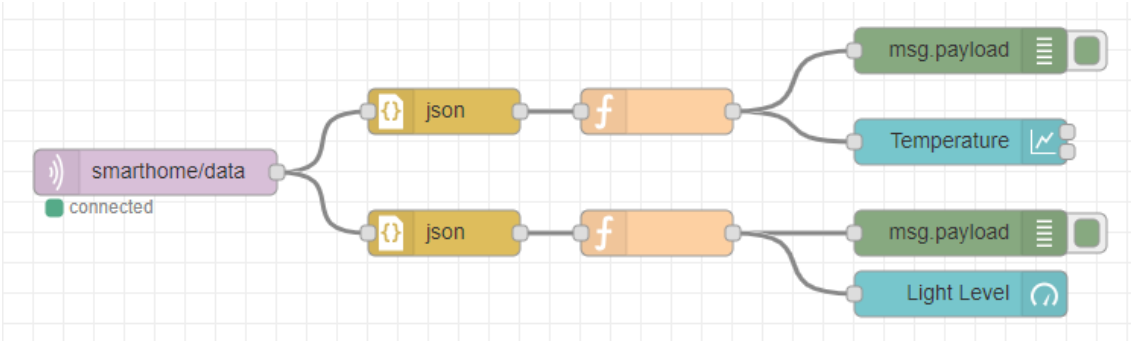



- Configure the MQTT credentials.
- Open the Dashboard UI.
- Test the system using MQTTLens.

# Sending multiple parameters using JSON

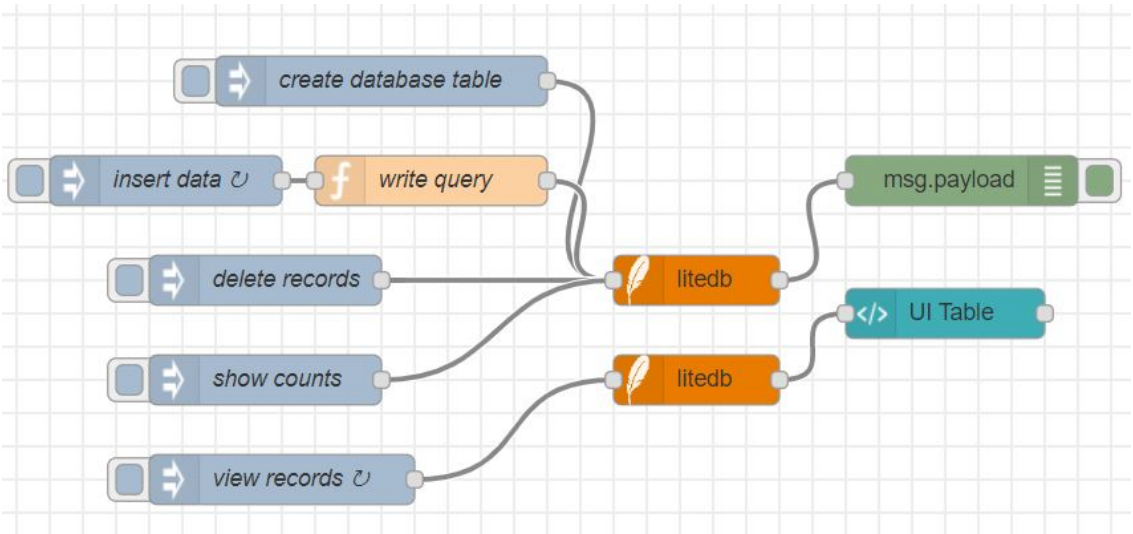
- Using the JSON message format we can send multiple values to the MQTT.
  {"temp":27,"humid":85}
- These values can be obtained using JSON and Function blocks in Node-RED using following function.

```
var value = msg.payload.Temp;
msg.payload = value;
return msg;
```



# Storing Data in Node-Red

- In built SQLite node can be used to store some data in Node-RED permanently.
- Export the sample flow using **store.json**



# Storing Data in Node-Red

- Using the flow shown in the last slide we can
  - create table
  - Insert data
  - Delete the old data
  - Obtain the number of data
  - Visualize the data in a tabular format.
  - Export the sample flow using **store.json**
- Basic SQL quarries and JavaScript skills are required to complete the above task.
- Following **slides** giving some basic guidelines on the above.

# Introduction to CoAP

# CoAP

- CoAP is a request/response based protocol similar to HTTP.
- Is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things.
- It is based on a REST model and support methods such as GET, PUT, POST, and DELETE.
- CoAP is designed to easily translate to HTTP for simplified integration with the web.

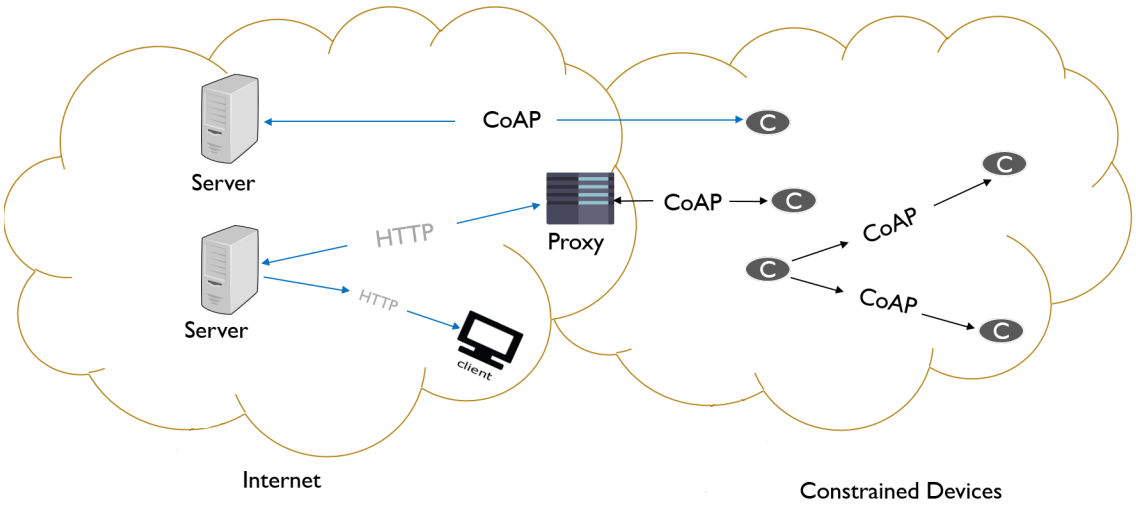



Figure 7: A simple CoAP architecture

# Communication in CoAP

- CoAP follows a client/server model.
- Clients make requests to servers, servers send back responses.
- CoAP use resources at server to exchange data.// Eg : /tempearte /light

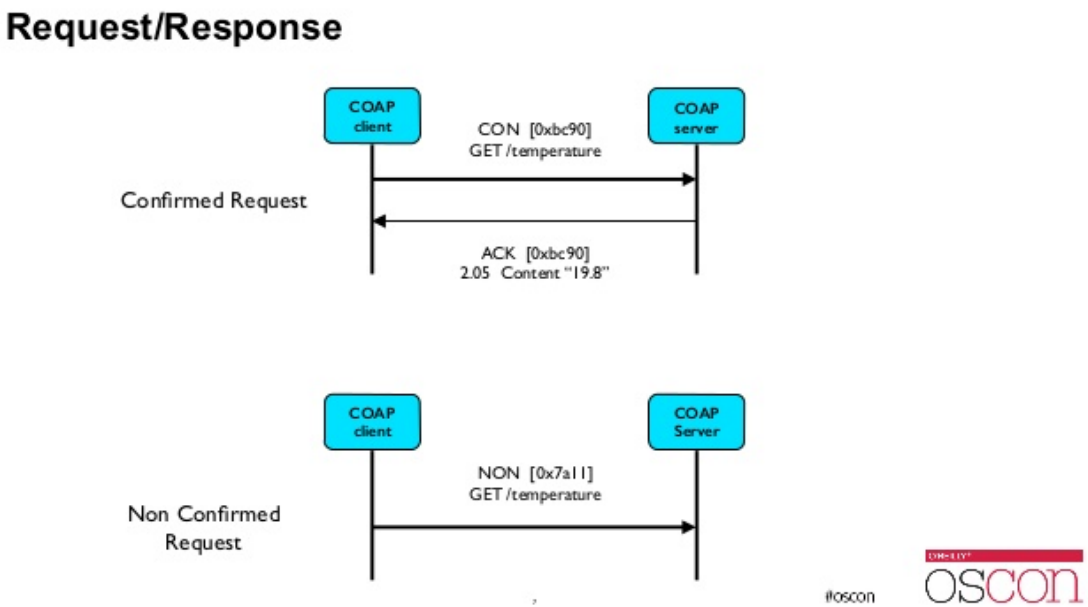



Figure 8: CoAp message transfer

# CoAP on ESP8266

- CoAp is very useful in exchanging data using constrained devices.
- We can implement CoAP server and client in a ESP8266.
- Following library implement a lightweight CoAP implementation in ESP8266. https://github.com/automote/ESP-CoAP

# CoAP client

- Open the **client.ino** file.
- Edit your WiFi Credentials.
- **coap.start()** in the setup() will start the coAP client.
- **coap.get(ip,port,"light")** will send a GET request with payload "light".
- Similarly you can send other kinds of request to a server.

# CoAP Server

- Open the **server.ino** file.
- Edit your WiFi Credentials.
- Following line show how to assign callback functions and resources to the CoAP server.
  **coap.server(callbacklight, "light");**
- If any requests are directed to the particular resource then the respective callback function will be triggered.

# CoAP application

- Using the above two example we can build a simple system to control the LED in a **ESP** from another **ESP** module.
- In the server side we are using **light** is the resource for the LED control.
- Also we can check the status of the LED using the same resource.
- Using the payload we can send several commands to the server **ESP** module.
- The callback function will handle these controls.

# Addtional Reading

- Additional Node-RED Topics
- JSON
- MQTT
- CoAP
- SQLite Nodered