

FIAP GRADUAÇÃO

I Agenda

- **Instalação node-red e primeiros testes**
- **Montado um Dashboard no node-red**
- **Montando um end-point**
- **Apresentação e uso do protocolo MQTT**

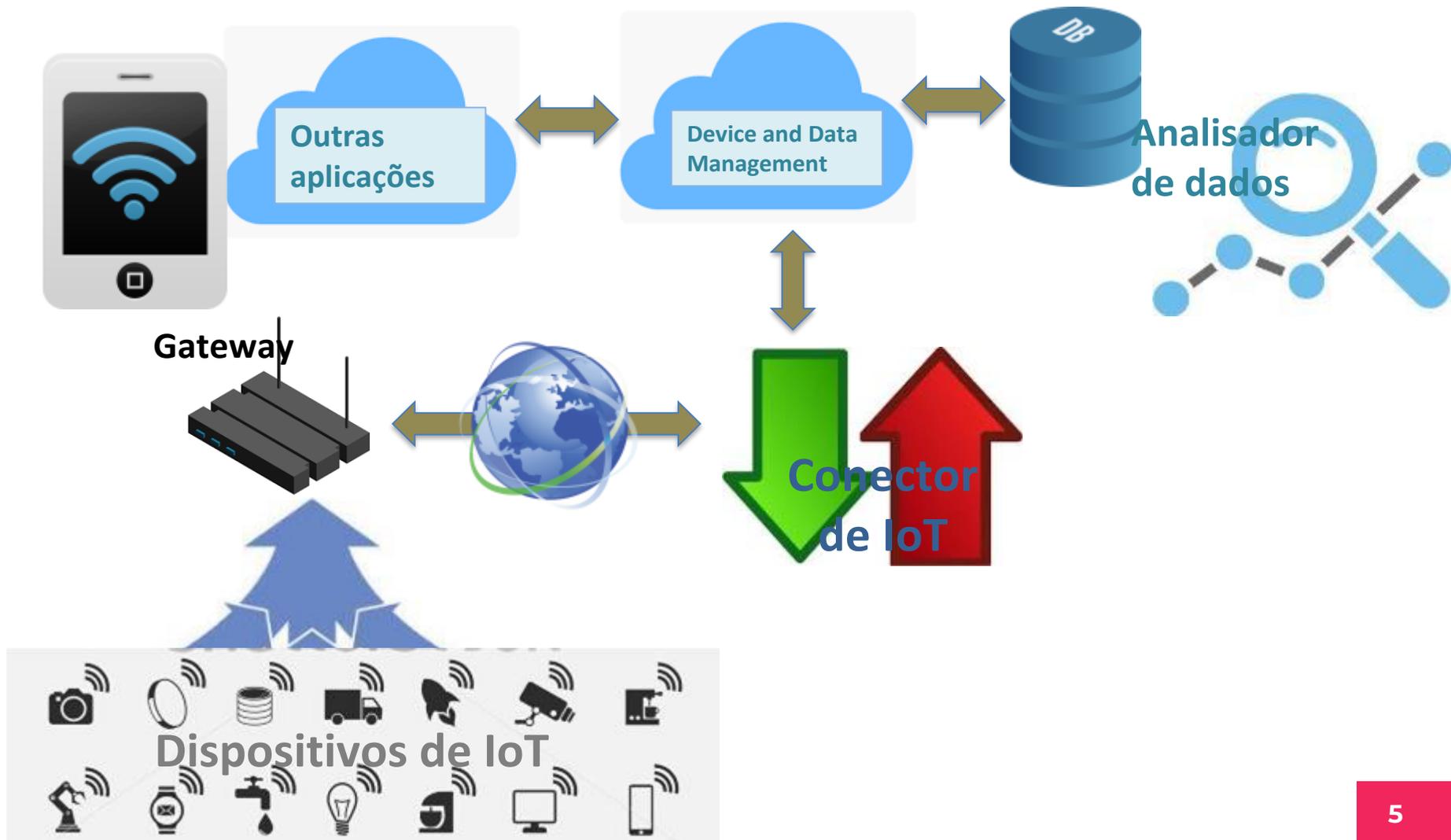
Conectando dispositivos a aplicações

- Agora que já exploramos as funcionalidades do Arduino e sua capacidade de conectar sensores e atuadores, vamos prosseguir conectando o Arduino a aplicações que fazem uso desse dispositivo
- Em primeiro lugar, vamos lembrar a arquitetura que usaremos para os dispositivos de IoT em geral se conectarem a suas aplicações

■ Arquitetura básica de implantação de IoT

- **Arquitetura de implantação** aqui fornecida é um desenho padrão para inspirar projetos reais a serem implementados, incluindo apenas os elementos fundamentais para a conectividade, sem detalhar soluções para problemas acessórios
- IoT envolve tantas tecnologias diferentes, permitindo tantas combinações diferentes, que projetos na área tendem a se tornar “Frankensteins”
 - Interoperabilidade: facilita a compatibilidade entre diferentes projetos de IoT
 - Modularidade: define módulos que podem ser criados separadamente ou ainda usados “off-the-shelf” (prateleira)
 - Compartilhamento de melhorias entre diferentes implementações

Arquitetura proposta



I Dispositivos de IoT (devices)

- Permitem a interação com o ambiente ao seu redor, seja capturando dados de sensores como executando comandos através de seus atuadores
- Cada funcionalidade no dispositivo pode ser considerado uma **Aplicação** (Endpoint Application)
 - Sensores de temperatura e luminosidade são aplicações diferentes dentro da mesma placa Arduino, por exemplo
 - Cada aplicação deve ser univocamente endereçável
 - Contexto embutido em vários padrões de comunicação como USB

■ Conector de IoT

- Gerenciam as mensagens que chegam de dispositivos ou são destinadas a eles, adaptando-as ao protocolo de cada dispositivo
 - Em uma arquitetura de implantação pode haver conectores diferentes para protocolos diferentes
 - São capazes de identificar e autenticar dispositivos
- Apesar de poder trabalhar com HTTP, em geral fazem uso de protocolos de aplicação mais simples ou mais adequados à IoT:
 - MQTT
 - WebSocket
 - CoAP
 - LoraWan

■ Device and Data Management

- Faz o gerenciamento remoto dos dispositivos e de seus dados, autorizando o acesso de outras aplicações.
 - Cadastra novos dispositivos e aplicações
 - Decide se um dispositivo anunciado pode ou não ser acrescentado à rede
 - Monitora se um dispositivo está disponível no momento
- Envia comandos de gerenciamento, como:
 - Inicialização e reinicialização
 - Desligamento
 - Atualização de firmware

■ Banco de dados e analisador de dados

- Armazena os dados vindos da aplicação, bem como os comandos que vão para os dispositivos
- Bancos de dados NoSQL são mais indicados, uma vez que a natureza das informações que são trocadas pelos dispositivos de IoT é muito diversa, podendo variar com o tempo
 - Ex: suponha que eu tenha uma tabela com os campos “Deviceld”, “Temperatura” e “Umidade”, mas tenha acabado de plugar um sensor de luminosidade...
- Faz sentido que os dados sejam monitorados por aplicações de análise de dados para um melhor aproveitamento

Gateway

- Faz a conexão de dispositivos que não tenha acesso direto à internet
 - Nem sempre é necessário
 - Quando necessário, realiza a conversão de protocolo entre os dispositivos de IoT e o conector de IoT
- O uso do IPv6 pelos dispositivos facilita a resolução do endereçamento do dispositivo, mas não é suficiente para resolver as mensagens específicas da aplicação
- Gerenciamento de múltiplos protocolos, especialmente com LAN's, PAN's e HAN's : Zigbee, Bluetooth, LoRa, Thread/6LoWPAN, etc.

Exemplo de Gateway: Philips Hue



Gateway:
Conversão entre
WiFi e Zigbee



Comunicação por Zigbee
(sem acesso ao WiFi)

■ Implementação do Gateway

- Dispositivos de IoT disponíveis no mercado podem ter seu próprio gateway (ex: Philips Hue) ou ainda se utilizar de computadores e smartphones (ex: smart bands Bluetooth)
- O gateway, no entanto, realiza funções simples, como autenticação do dispositivo, envio e recepção de mensagens do servidor com adaptação de protocolo, e algumas funções específicas de cada dispositivo.
- Dessa forma, a construção do gateway para o Arduino pode usar uma programação simples e visual, que explicita a origem e o destino das informações, ou seja
 - Que mensagens vêm do servidor para o dispositivo
 - Que mensagens vão do dispositivo para o servidor

■ Node-RED - <https://nodered.org/>

- Plataforma para programação visual de sistemas baseados em eventos
 - Executa como um servidor web na máquina hospedeira, que deve ter certas configurações mínimas de processamento e memória, como tablets e o Raspberry-Pi
- Contribuições da comunidade
 - Grande disponibilidade de módulos (bibliotecas) que executam diversas funções, elaborados por empresas e desenvolvedores voluntários

Node-RED e Node.js

- O Node-Red é um serviço escrito para Node.js que provê uma ferramenta visual para editar fluxos de mensagens, vindas de diferentes fontes, podendo ser processadas e mandadas para diferentes destinos, como uma conta de e-mail ou do Twitter
 - A ferramenta para edição dos fluxos roda no próprio browser
 - É possível exportar e importar fluxos no formato JSON usando o menu de opções
- Uma vez que a programação do Node.js é assíncrona, podemos pensar em programas em que todas as ações são acionadas por um evento gatilho.
 - Sendo assim, podemos pensar na programação do Node.js como fluxos de dados que iniciam a partir de algum evento, como o disparo de um temporizador, a requisição de um cliente de webservice ou um dado vindo do Arduino
- O Node-Red está disponível no IBM Bluemix e em outros provedores de Cloud Computing

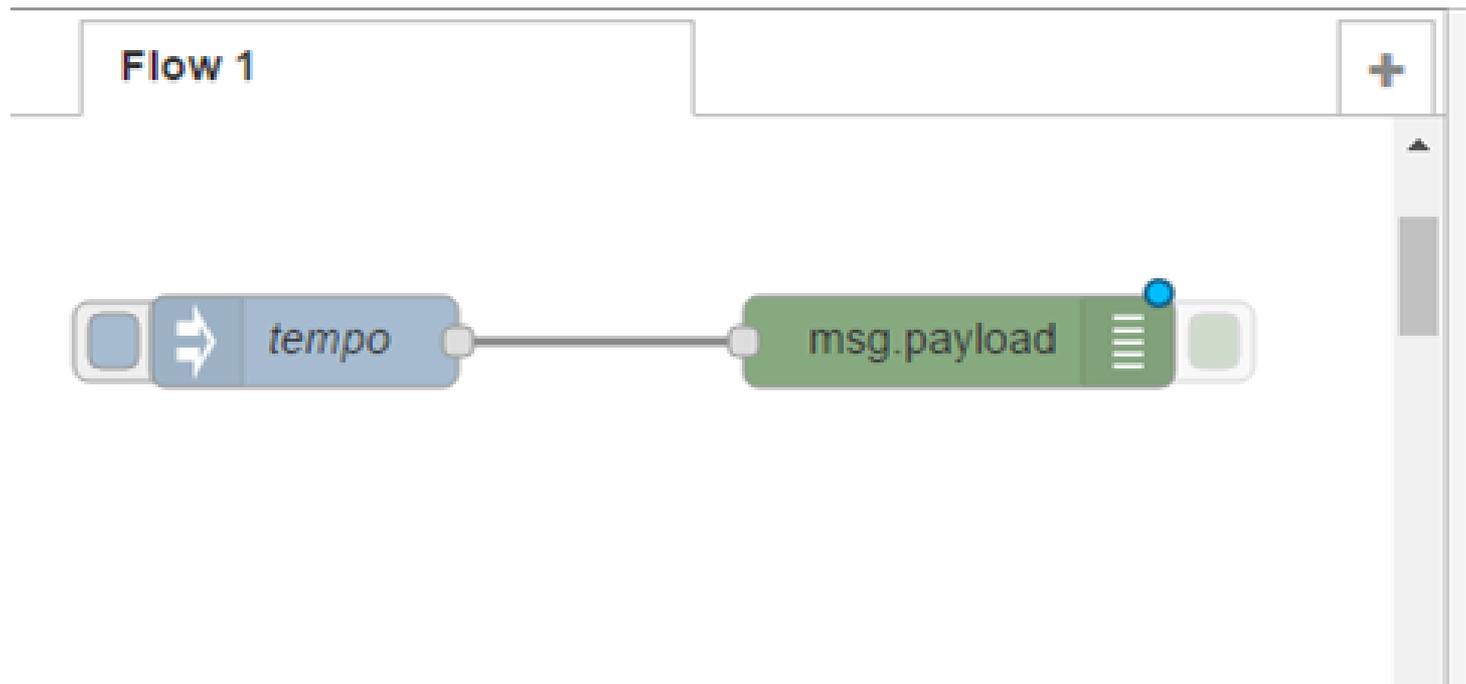
LAB1: Instalação local do Node-Red

- Faça a instalação do Node.js (versão LTS)
 - www.nodejs.org
- abra o cmd e digite:
 - `npm install -g --unsafe-perm node-red`
- Para acessar o serviço, após instalado, digite no cmd:
 - `node-red`
- Acessamos no browser:
 - <http://localhost:1880>

ref: <https://nodered.org/docs/getting-started/local>

LAB2: Primeiro fluxo (Flow) - Olá mundo!

- Inicialmente, ligar um nó de entrada do tipo “inject” a um nó do tipo “debug”, fazer um “Deploy” e acionar o injetor de dados
- Observar o resultado na tela de Debug
- Faça algumas alterações do inject, faça o deploy e analise o resultado



Desafio1

Monitor do clima tempo.

Faça o cadastro no site: Openweather

<https://openweathermap.org/>

Crie um token.

Este site possui uma API que permite fazer requests. Leia a documentação:

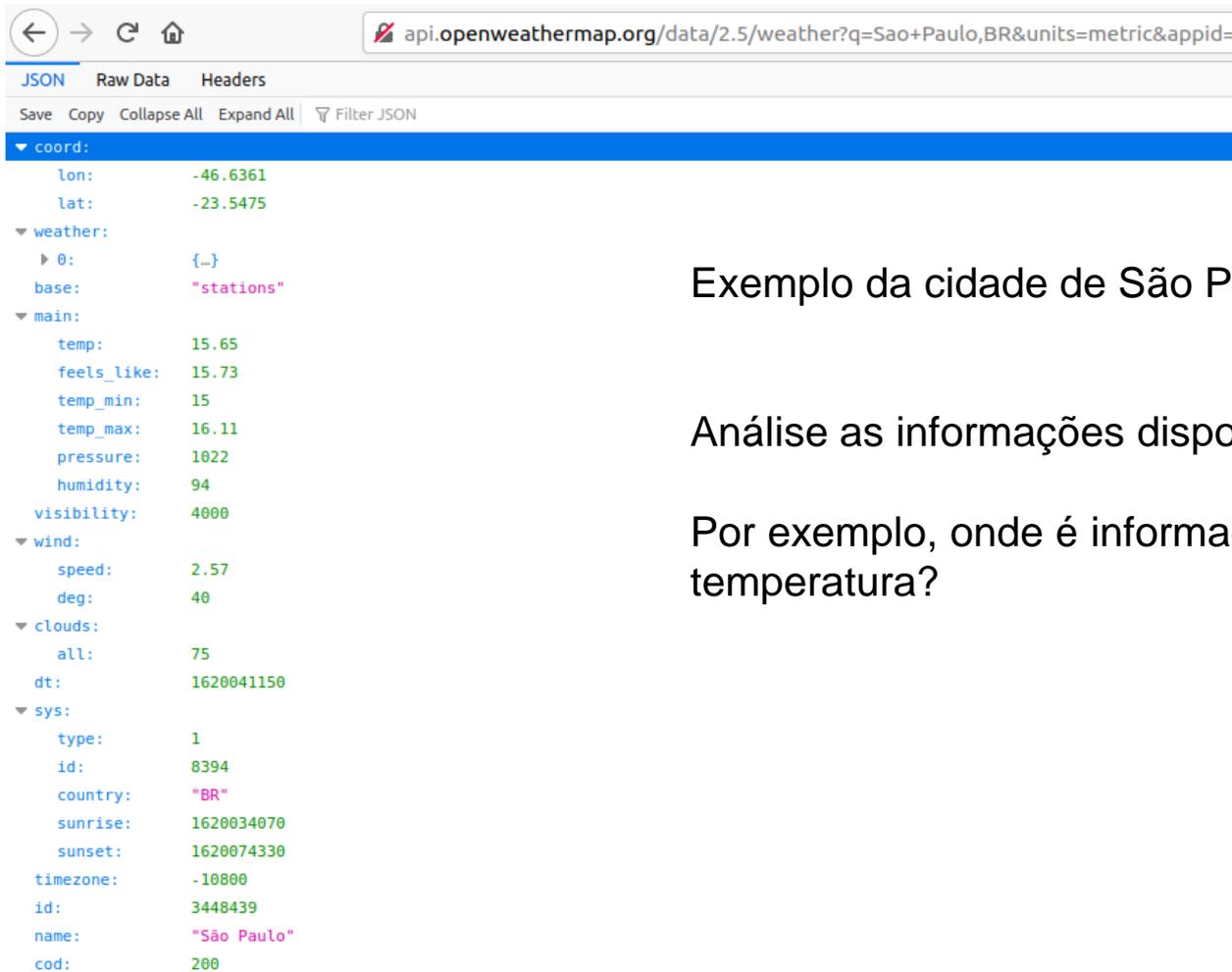
<https://openweathermap.org/current>

Crie uma URL que faz o request do tempo em alguma cidade de sua preferência. O resultado esperado é parecido com o a imagem.

```
api.openweathermap.org/data/2.5/weather?q={city name},  
{state code},{country code}&appid={API key}
```



Desafio1



```
api.openweathermap.org/data/2.5/weather?q=Sao+Paulo,BR&units=metric&appid=
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
▼ coord:
  lon: -46.6361
  lat: -23.5475
▼ weather:
  0: {}
  base: "stations"
▼ main:
  temp: 15.65
  feels_like: 15.73
  temp_min: 15
  temp_max: 16.11
  pressure: 1022
  humidity: 94
  visibility: 4000
▼ wind:
  speed: 2.57
  deg: 40
▼ clouds:
  all: 75
  dt: 1620041150
▼ sys:
  type: 1
  id: 8394
  country: "BR"
  sunrise: 1620034070
  sunset: 1620074330
timezone: -10800
id: 3448439
name: "São Paulo"
cod: 200
```

Exemplo da cidade de São Paulo.

Análise as informações disponibilizadas.

Por exemplo, onde é informada a temperatura?

Desafio1



Properties

Method: GET

URL: `api.openweathermap.org/data/2.5/weather?q=Sac`

Payload: Ignore

Enable secure (SSL/TLS) connection

Use authentication

Enable connection keep-alive

Use proxy

Return: a UTF-8 string

Name: Name

Sua URL

Desafio1

Análise o Debug do node-red e compare com o resultado obtido pelo navegador, as informações devem ser coincidentes. Algumas dessas informações não são muito relevantes e podemos filtrar.



Usando o node change, filtre apenas a “temp”



Usando o node function, escreva uma função que retorne os tópicos:
temperatura, temperatura min, temperatura max e velocidade do vento

Dica: Use o debug, e lembre a estrutura de um JSON

I Desafio1

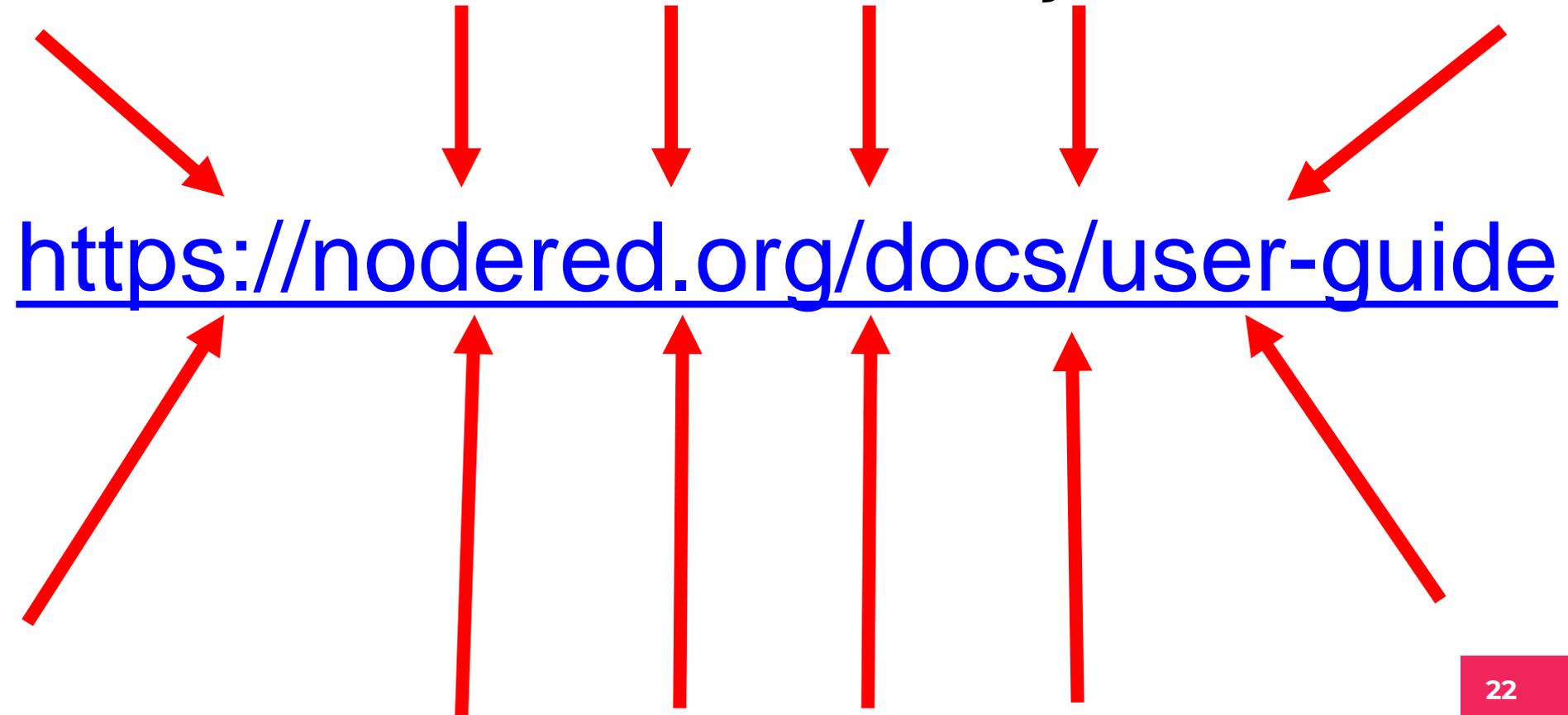
PAREEEEEEEEEEEEE!!!!!!!!!!!!

Você já chegou em uma solução?????

Desafio1 - Solução

Sugestão: Está na documentação oficial!

SEMPRE LEIA A DOCUMENTAÇÃO OFICIAL!



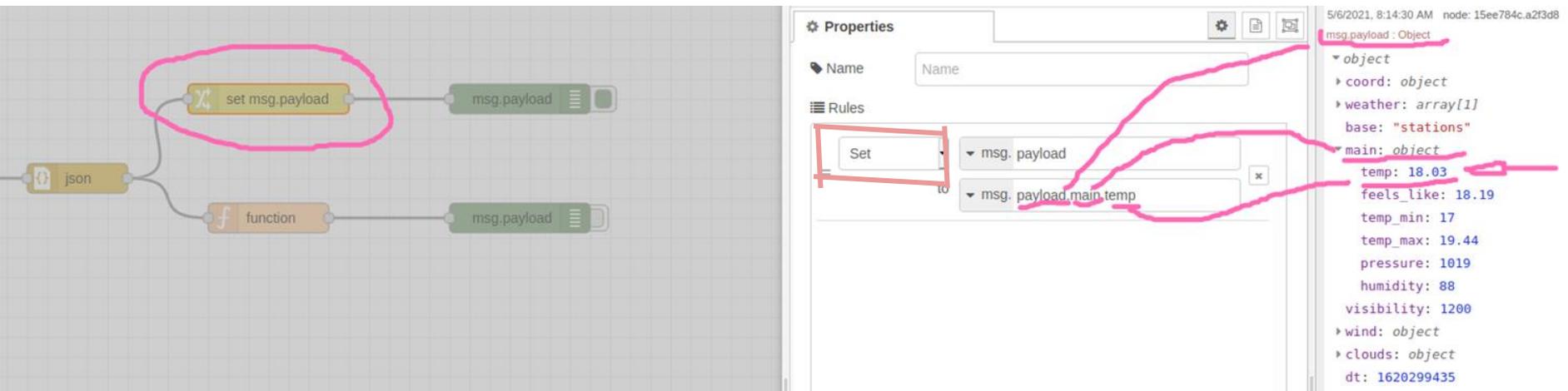
Desafio1 - Solução



- Realiza operações de modificações nos campos da mensagem, podendo modificar, acrescentar, apagar ou renomear um campo
 - Serve também para seus subcampos
 - Podem ser realizadas diversas operações consecutivas
- As operações que podem ser executadas são:
 - **Set**: cria ou modifica o valor de um campo
 - **Modify**: substitui o valor de um campo por outro
 - **Move**: renomeia um campo
 - **Delete**: apaga e remove um campo

Desafio1 - Solução

 **change** Usando o node change, filtre apenas a “temp”



The image shows a Node-RED workflow and its configuration. On the left, a workflow is visible with a 'json' node connected to two 'set msg.payload' nodes, which are both connected to 'msg.payload' nodes. The top 'set msg.payload' node is circled in pink. On the right, the configuration for the 'set msg.payload' node is shown. The 'Rules' section is highlighted with a red box, showing a 'Set' rule with the path 'msg.payload' and 'msg.payload.main.temp'. The 'Properties' section shows the 'Name' field. On the far right, the JSON output of the workflow is displayed, with the 'temp' value highlighted in pink.

```
msg.payload : Object
  object
  coord : object
  weather : array[1]
  base : "stations"
  main : object
    temp : 18.03
    feels_like : 18.19
    temp_min : 17
    temp_max : 19.44
    pressure : 1019
    humidity : 88
    visibility : 1200
  wind : object
  clouds : object
  dt : 1620299435
```

O **msg.payload** carrega todas a informações que vem da API, queremos apenas a temperatura.

Vamos analisar a estrutura do **msg.payload** para configurar corretamente.

O resultado fica: **payload.main.temp**

Desafio1 - Solução

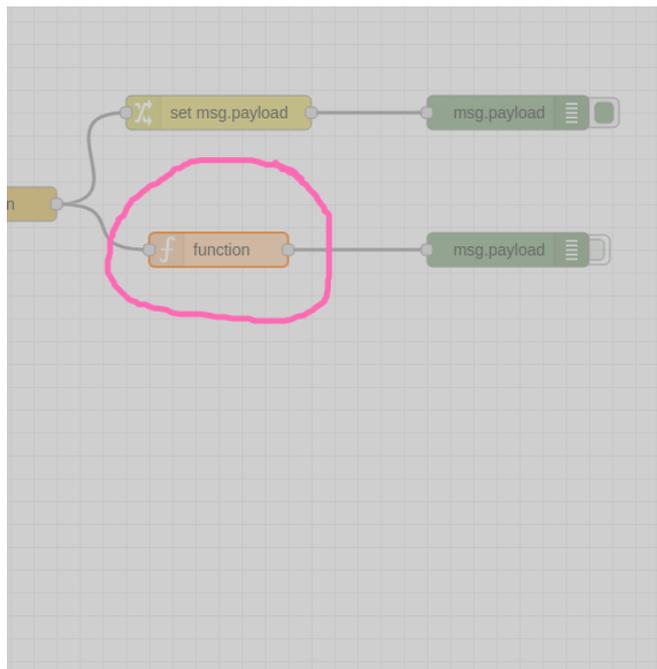


- Cria uma função JavaScript genéricas que podem manipular os campos da mensagem como se desejar.
 - A função do Javascript recebe o parâmetro msg, e pode retornar um ou mais objetos
 - Para o bom funcionamento do programa, é preciso que o(s) valor(es) retornados sejam a própria msg com as devidas modificações, ou null para interromper o caminho da mensagem
- O node Function implementa apenas o corpo da função.

Desafio1 - Solução



Usando o node function, escreva uma função que retorne os tópicos: temperatura, temperatura min, temperatura max e velocidade do vento



Properties

Name:

Setup | On Start | **On Message** | On Stop

```

1
2 return msg;
    
```

função

retorno da função

```

5/6/2021, 8:14:30 AM node: 15ee784c
msg.payload: Object
  object
  coord: object
  weather: array[1]
    base: "stations"
  main: object
    temp: 18.03
    feels_like: 18.19
    temp_min: 17
    temp_max: 19.44
    pressure: 1019
    humidity: 88
    visibility: 1200
  wind: object
  clouds: object
    dt: 1620299435
  sys: object
    timezone: -10800
    id: 3448439
    name: "São Paulo"
    cod: 200

5/6/2021, 8:22:44 AM node: 15ee784c
msg.payload: number
18.03
    
```

Desafio1 - Solução

Note que recebemos as informações em `msg.payload` e que damos return em `msg`. Queremos as informações:

```
temp
temp_min
temp_max
speed
```

Temos algumas soluções possíveis. Uma delas, vamos criar uma saída `msg` que vai exibir os dados filtrados mantendo o formato de um objeto Java Script (“chave”:valor):

```
msg.payload = {
  "chave": valor,
  "chave": valor,
  "chave": valor,
  "chave": valor }
```

return msg;

[template](#)

```
msg.payload = {
  "temperatura": msg.payload.main.temp,
  "min": msg.payload.main.temp_min,
  "max": msg.payload.main.temp_max,
  "vento": msg.payload.wind.speed
}
```

return msg;

[Resultado](#)

Desafio1 - Solução



Usando o node function, escreva uma função que retorne os tópicos: temperatura, temperatura min, temperatura max e velocidade do vento

The screenshot shows the Node-RED interface. On the left, a workflow is visible with a 'function' node (circled in pink) connected to a 'msg.payload' node. The 'function' node is selected, and the 'Edit function node' dialog is open. The dialog shows the 'On Message' tab with the following code:

```

1 msg.payload = {
2   "temperatura": msg.payload.main.temp,
3   "min": msg.payload.main.temp_min,
4   "max": msg.payload.main.temp_max,
5   "vento": msg.payload.wind.speed
6 }
7
8 return msg;

```

On the right, the 'debug' console shows the output of the function:

```

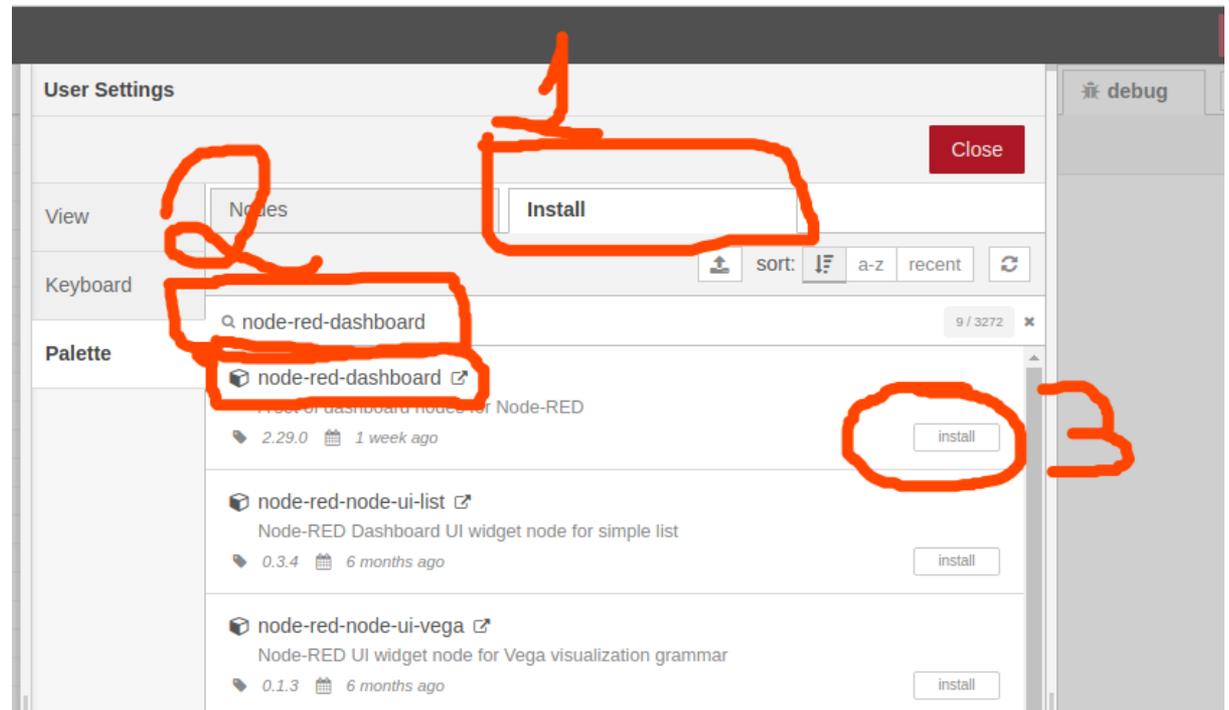
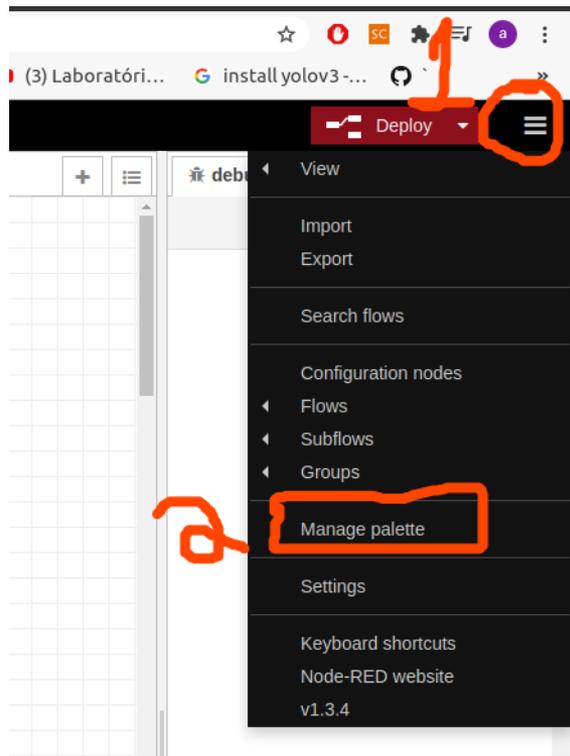
5/6/2021, 4:07:20 PM node: 3d1e21ad.1cf2fe
msg.payload : Object
  object
    temperatura: 27.48
    min: 26.67
    max: 28
    vento: 5.66

```

Orange arrows point from the code in the function editor to the corresponding values in the debug console output.

Dashboard - Instalar lib

Primeira coisa, vamos instalar os nodes para dashboard, caso não já não tenha instalado.



node-red-dashboard

Dashboard - Instalar lib

Primeira coisa, vamos instalar os nodes para dashboard, caso não já não tenha instalado.

The screenshot shows the Node-RED interface with a search for 'node-red-dashboard'. A modal dialog is open, warning that some nodes have dependencies that cannot be automatically resolved and may require a restart of Node-RED. The dialog has three buttons: 'Cancel', 'Open node information', and 'Install'. The 'Open node information' and 'Install' buttons are circled in orange. A large orange number '1' is drawn over the search results. The search results list several nodes, including 'node-red-dashboard' (version 2.29.0, 1 week ago) and 'node-red-node-ui-list' (version 0.3.4, 6 months ago). A large orange text overlay at the bottom reads 'LEIA A DOCUMENTAÇÃO'.

Installing 'node-red-dashboard'

Before installing, please read the node's documentation. Some nodes have dependencies that cannot be automatically resolved and can require a restart of Node-RED.

Cancel Open node information Install

Keyboard

Palette

node-red-dashboard

node-red-dashboard

A set of dashboard nodes for Node-RED

2.29.0 1 week ago install

node-red-node-ui-list

Node-RED Dashboard UI widget node for simple list

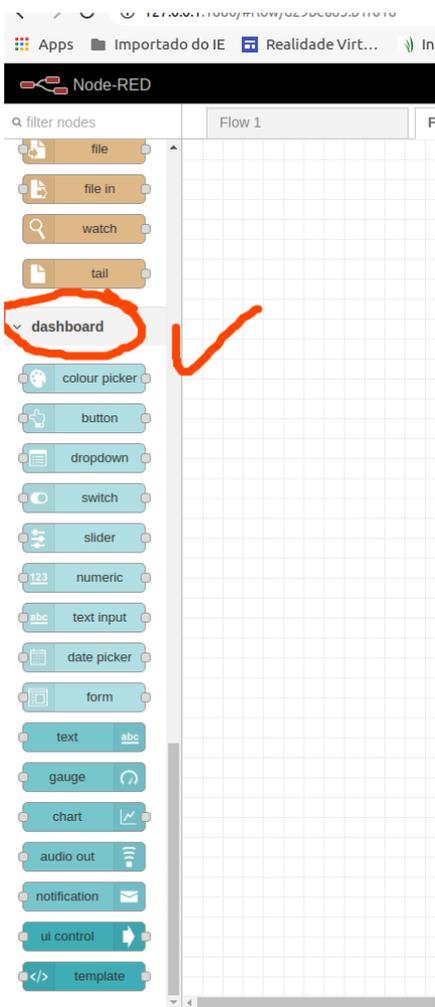
0.3.4 6 months ago install

node-red-node-ui-vega

Node-RED UI widget node for Vega visualization grammar

LEIA A DOCUMENTAÇÃO

Dashboard - Instalar lib

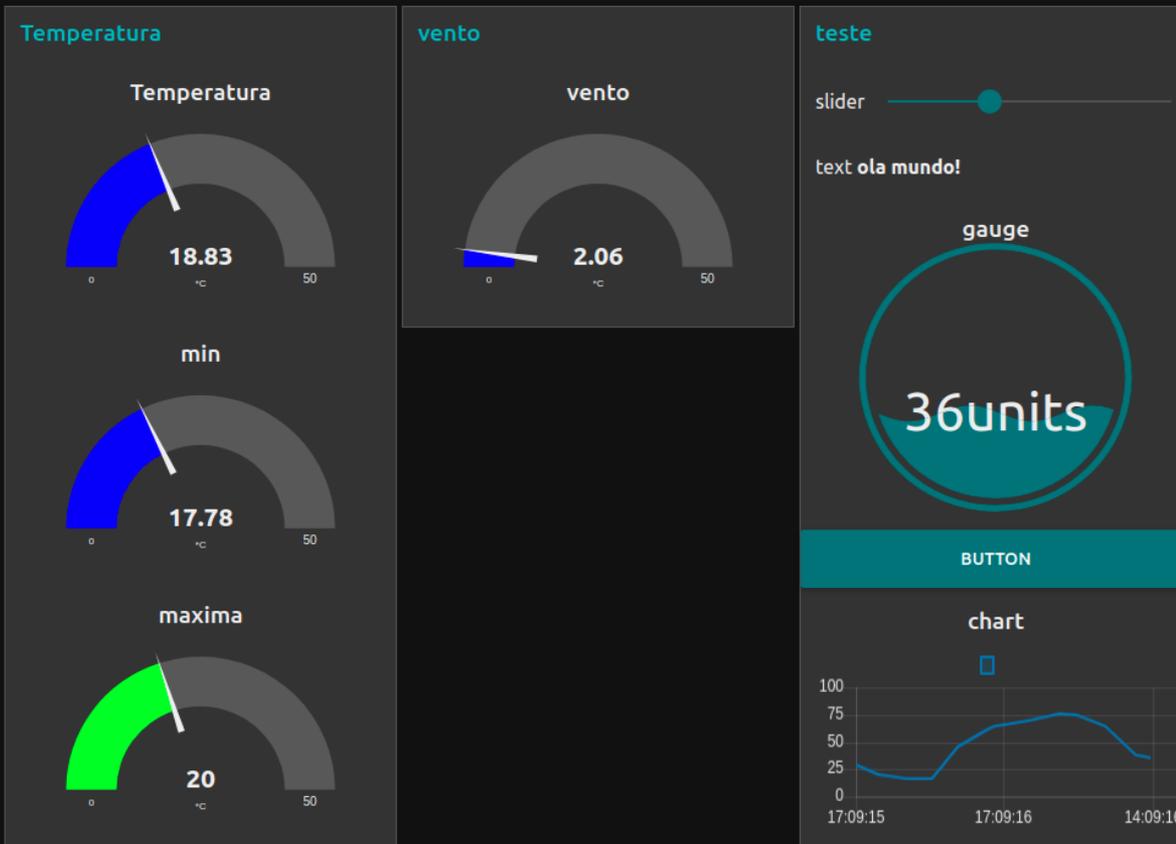


Show! Vamos usar

Dashboard - Como elaborar...

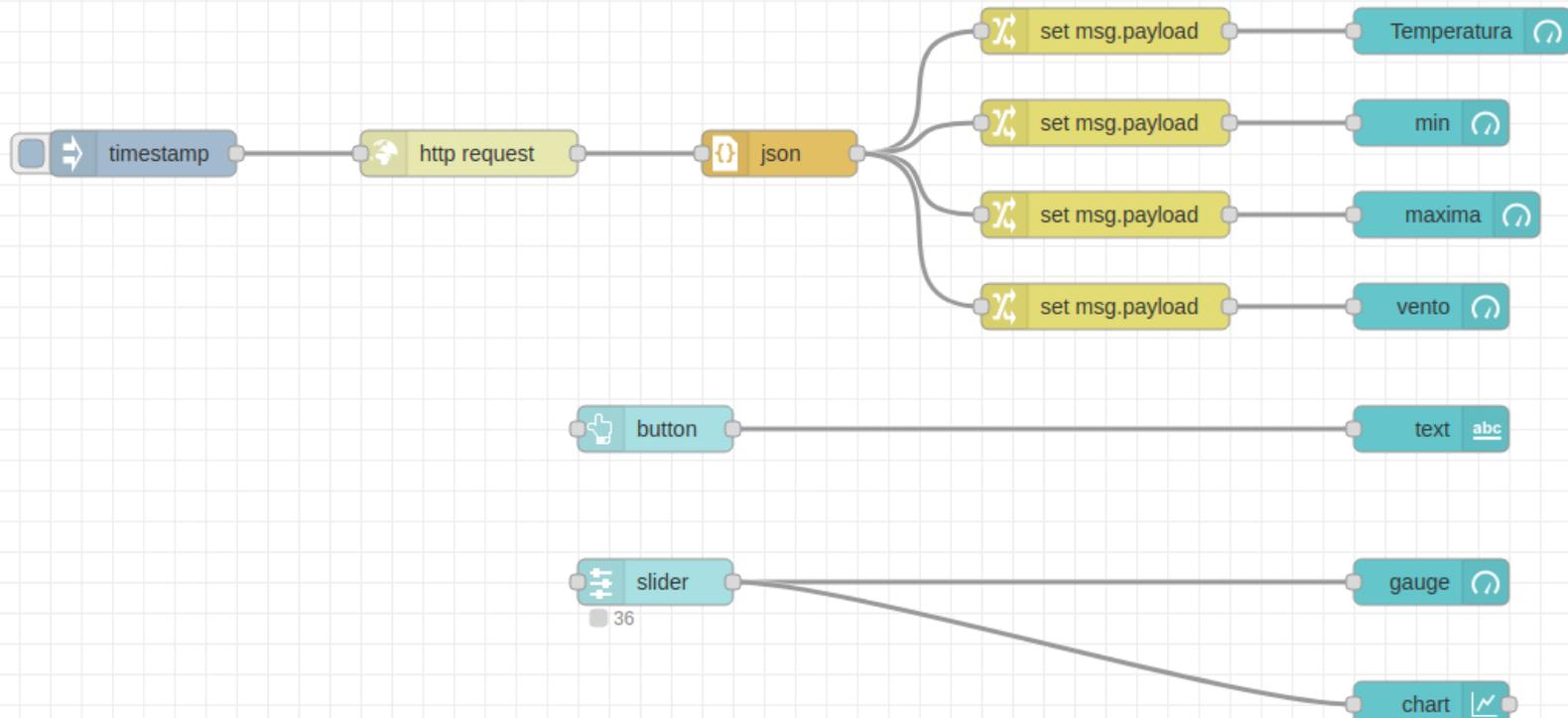
Home

Apenas exemplo...

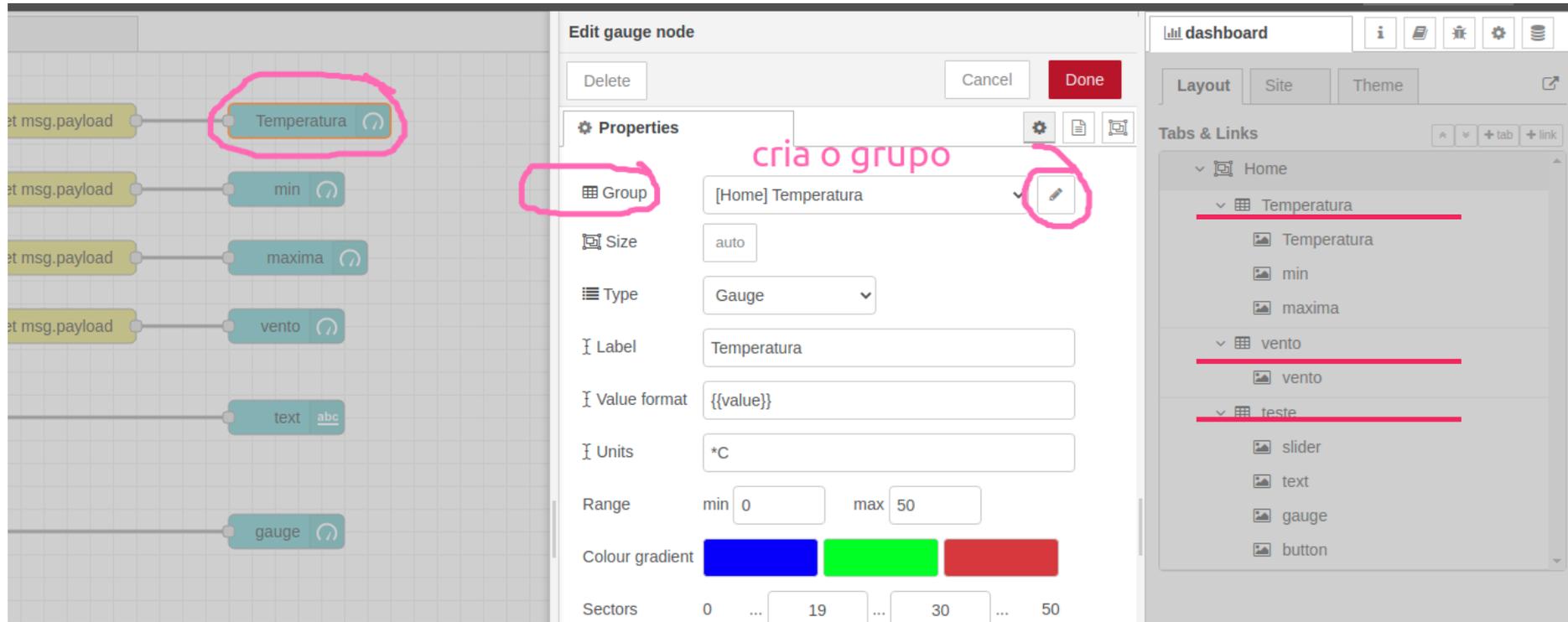


Dashboard - Como elaborar...

Monte o flow...



Dashboard - Como elaborar...



cria o grupo () e edite os campo como achar mais conveniente, neste exemplo eu criei 3 grupos. Temperatura, vento, teste

I Desafio 2

O dashboard de exemplo do professor está muito feio...mas é um bom começo para criar novos dashboards, e bemmm mais bonitos.

Edite/Crie de forma criativa um dashboard para exibir as informações da API Open Weather Map.

O mínimo esperado:

Fazer a busca de pelo menos duas cidades diferentes da sua preferência, exibindo pelo menos as informações:

- Temperatura atual;
- Temperatura mínima;
- Temperatura máxima;
- Velocidade do vento;
- Humidade relativa;
- Sensação térmica.

A temperatura atual das duas cidades (ou mais) devem ser exibidas no mesmo gráfico (chart) e atualizado a cada 5 segundos.

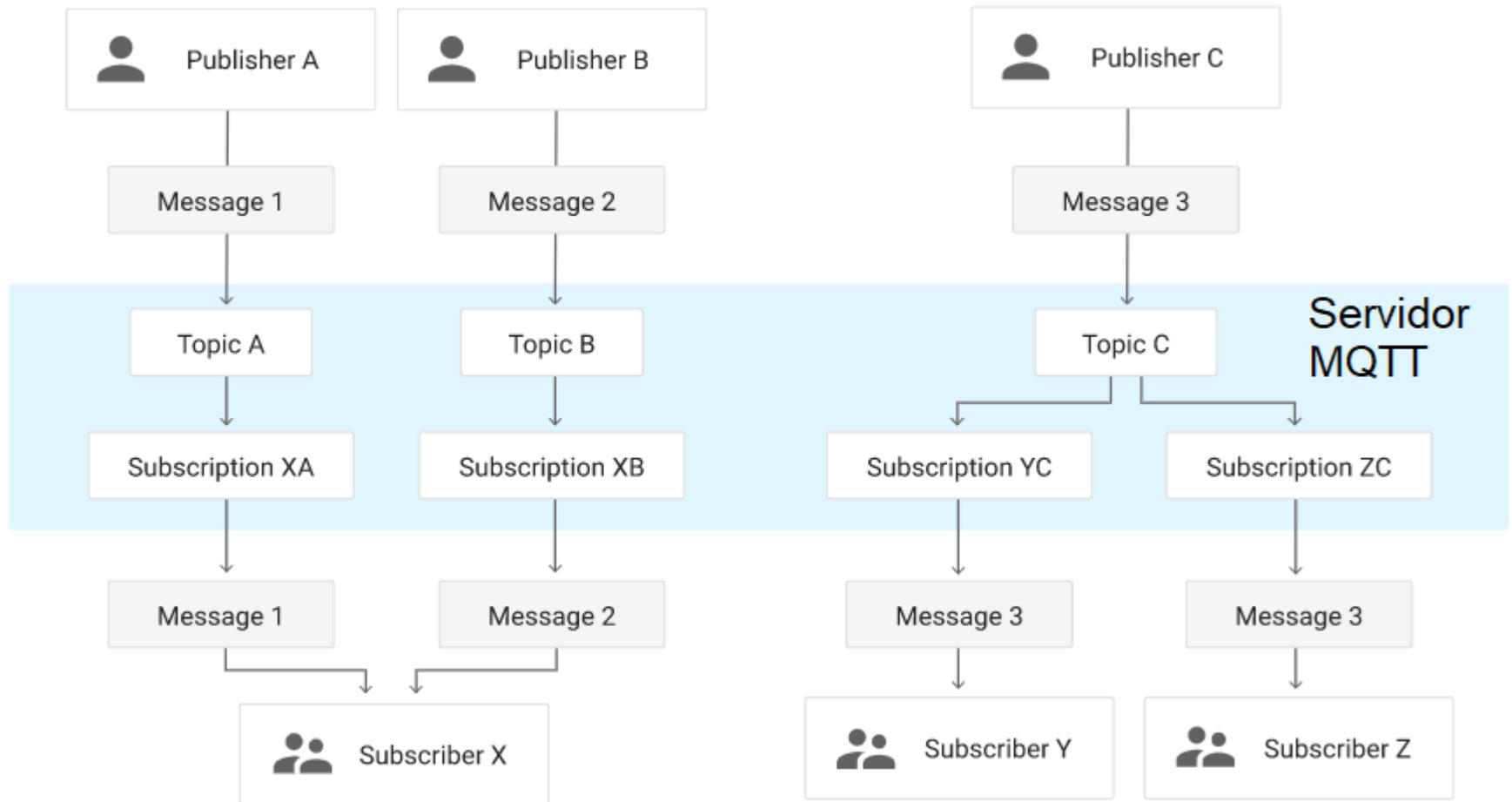
MQTT – MQ Telemetry Transport

- Protocolo muito simples para publicação e recebimento de mensagens, apropriado para dispositivos com alta latência e baixa largura de banda de comunicação
- A leveza do protocolo, que usa cabeçalhos de poucos bytes, o torna adequado para a comunicação de objetos no cenário da internet das coisas
- Um servidor MQTT, também conhecido como *broker*, faz o papel de servidor, que gerencia as mensagens publicadas, enviando-as aos clientes que se inscreveram para recebê-las
- Os clientes MQTT são as pontas da comunicação, podendo enviar ou receber mensagens através das operações:
 - **Publish**: um cliente MQTT publica uma mensagem com determinado tópico
 - **Subscribe**: um cliente se cadastra no servidor para receber cópias de mensagens com determinado tópico

Tecnologias de comunicação para IoT

		3G		WiFi	
		HTTPS	MQTT w. SSL	HTTPS	MQTT w. SSL
Receive	msg/hour	1,708	160,278	3,628	263,314
	%battery/msg	0.01709	0.00010	0.00095	0.00002
	msg delivery	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
Send	msg/hour	1,926	21,685	5,229	23,184
	%battery/msg	0.00975	0.00082	0.00104	0.00016

Ilustração do protocolo MQTT



Usando o MQTT – conectando a um servidor

- O protocolo MQTT pode ser testado facilmente empregando simples aplicativos para celular
- Após a instalação do programa cliente basta configurar a conexão com o servidor MQTT, também chamado de “Message Broker”, fornecendo seu endereço IP ou URL, na porta padrão 1883
- Para uso em teste, um servidor público pode ser empregado, tais como:
 - `iot.eclipse.org`
 - `test.mosquitto.org`
 - `dev.rabbitmq.com`
 - `broker.mqttdashboard.com`
- Também é possível instalar e configurar o próprio servidor MQTT
 - No caso de uso em uma rede local, com poucas conexões, o servidor mosquitto (`mosquitto.org`) é o mais apropriado, por consumir poucos recursos, podendo ser executado em plataformas de IoT como Raspberry Pi
 - Para uso no ambiente da Cloud Computing, onde podemos esperar milhões de conexões e necessitamos de escalabilidade, precisamos de um servidor do tipo RabbitMQ (`www.rabbitmq.com`)

■ Desafio 3 - Testando com um webclient

- Alguns servidores MQTT permitem o uso do protocolo WebSocket para o transporte da mensagem
 - Com isso, é possível criar clientes da Web que enviam e recebem mensagens através de um servidor MQTT
- Para testar: <http://www.hivemq.com/demos/websocket-client/>

■ Desafio 3 - Testando com um webclient

Crie um chat, com **publish** e **subscribe**.

Sugestão: O tópico pode ser juntando o primeiro nome com o último sobrenome, em “*camelCase*”.

- Para enviar uma mensagem a alguém, use o tópico daquela pessoa

exemplo:

```
arnaldoAVianaJr/+/teste/#
```

```
arnaldoAVianaJr/fiap/teste2
```

■ Desafio 4 - Pesquisa e Resposta :)

1. QoS (Quality of Service) – indica o nível de verificação de recebimento de mensagens pelo servidor (publish) ou pelo cliente (subscribe)
 - **QoS 0:** *(sua resposta aqui..)*
 - **QoS 1:** *(sua resposta aqui..)*
 - **QoS 2:** *(sua resposta aqui..)*
2. Caracteres coringa na subscrição de tópicos:
 - a. Níveis múltiplos (multi-level): o caractere #
 - *(sua resposta aqui.. de exemplos)*
 - b. Nível simples (single-level): o caractere +
 - *(sua resposta aqui.. de exemplos)*

Desafio 5 - Cliente MQTT no Node-RED

Node-RED

filter nodes

network

- mqtt in
- mqtt out
- http in
- http response
- http request
- websocket in
- websocket out
- tcp in
- tcp out
- tcp request
- udp in
- udp out

Edit mqtt in node

Delete Cancel Done

Server Add new mqtt-broker... 

Topic Topic

mqtt in > Add new mqtt-broker config node

Cancel Add

Connection Security Birth Message Will Message

Server broker.hivemq.com Port 1883

Enable secure (SSL/TLS) connection

Client ID Leave blank for auto generated

Keep alive time (s) 60 Use clean session

Use legacy MQTT 3.1 support

Desafio 5 - Cliente MQTT no Node-RED

The screenshot displays a Node-RED workflow on a grid background. At the top, a blue 'inject' node with the payload 'Chegueiiii!!!' is connected to a purple MQTT client node labeled 'arnaldoAViana/fiap/teste1'. The MQTT node has a green 'connected' indicator and a speaker icon. Below this, another purple MQTT client node with the same label and 'connected' status is connected to a green 'msg.payload' node. A blue arrow points from the text 'Crie o nome do seu tópic!' to the MQTT client nodes. In the bottom right, a 'debug' console window shows the following log entry:

```
30/07/2021 12:33:47 node: 13f66d94.415972
arnaldoAViana/fiap/teste1 : msg.payload : string[13]
"Chegueiiii!!!"
```

Crie o nome do seu tópic!



Na próxima aula...

Laboratório: Conectando o Arduino com o Node-Red