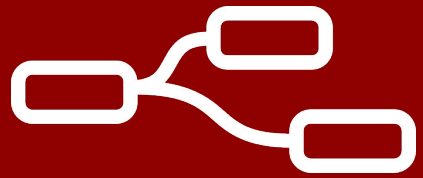
Desenvolvimento de aplicações integrando serviços web, fontes de dados e dispositivos IoT com o uso do

Node-RED



Jordano R. Celestrini
jordanorc@gmail.com

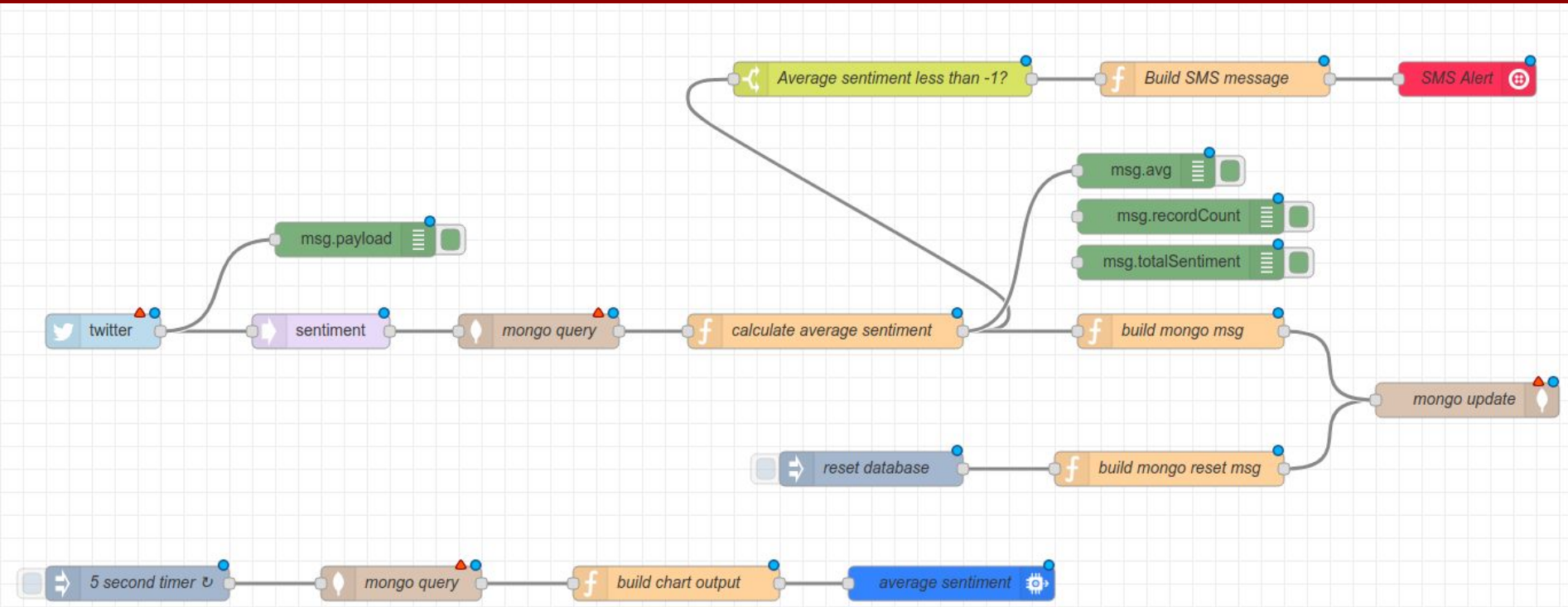
Agenda

- **Introdução e instalação**
  - O que é o Node-RED
  - Breve histórico
  - Conceitos do Node-RED
  - Instalação e execução
  - A interface do Node-RED
  - *Hands on*: HelloWorld - Escrevendo o seu primeiro fluxo
- **JSON e MQTT**
  - O que é o JSON
  - O que é o MQTT
  - MQTT no Node-RED: usando o Eclipse Mosquitto
- **Explorando nós do Node-RED**
  - Nós principais do Node-RED
  - Executando requisições HTTP
  - Adicionando novos nós à paleta
  - *Hands on*: Criando um robô com Telegram
- **Dispositivos e Dashboards**
  - Conectando o Node-RED com dispositivos
  - SensorTag
  - Miband
  - Introdução ao Dashboard
  - Nós do Dashboard
  - *Hands on*: Criando um dashboard para exibição de dados
- **Tópicos avançados**
  - Instalando e configurando o Node-RED no Raspberry
  - Integração com IBM Watson
  - *Hands on*: Criando um Chatbot utilizando o Watson Conversation

O que é o Node-RED

- Ferramenta para criação de aplicações IoT
- Simplifica o desenvolvimento “ligando” de blocos de código para executar tarefas.
- Faz uso de uma abordagem de programação visual que permite aos desenvolvedores conectar blocos de códigos predefinidos, conhecidos como "nós", para executar uma determinada tarefa.
- Os nós conectados, geralmente uma combinação de nós de entrada, nós de processamento e nós de saída, quando ligados juntos, formam um "fluxo".

O que é o Node-RED



Breve histórico

- Criado e 2013 pela IBM
- Função: conectar dispositivos e hardware de maneira simples a serviços web e outros softwares
  - Um tipo de “cola” para IoT
- É opensource - https://github.com/node-red/node-red
- Com o passar do tempo, transformou-se em uma ferramenta de programação IoT para os mais variados propósitos
- Possui uma comunidade muito ativa

Breve histórico

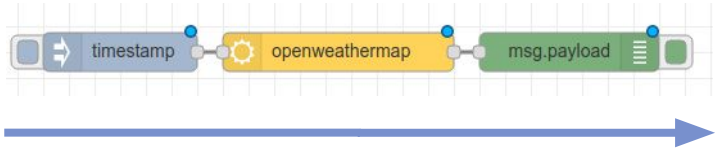
- Instalação simples (`sudo npm install -g --unsafe-perm node-red`)
- Vasta documentação
  - Tutoriais: https://cookbook.nodered.org/
  - Guia de usuário: https://nodered.org/docs/user-guide/
  - Tutorial para criação de nós: https://nodered.org/docs/creating-nodes/
  - Documentação da API: https://nodered.org/docs/api/
  - Tutorial para desenvolvedores colaboradores: https://nodered.org/docs/developing/

Breve histórico

- Muitos exemplos de fluxos prontos:
  - https://flows.nodered.org/?type=flow
- Muitos nós disponíveis (https://flows.nodered.org/?sort=downloads&type=node):
  - Dashboard
  - Redes sociais / mensageiros: Twitter, Facebook, Slack, Telegram
  - Integração com outros serviços: IBM Watson, Openwhisk
  - Banco de dados: MySQL, CouchDB, MongoDB, Azure SQL

Conceitos do Node-RED

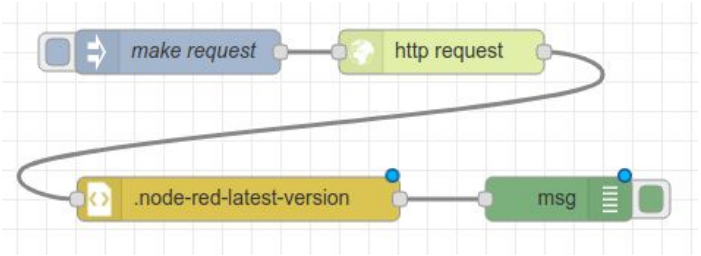
- **Fluxos**
  - No Node-RED, programas são fluxos compostos por uma coleção de nós conectados para trocar mensagens.
  - Cada nó tem um propósito bem definido
  - Tecnicamente, um fluxo consiste em uma lista de objetos JavaScript que descrevem os nós e suas configurações




```
[
    {
        "id": "31e10b93.612544",
        "type": "openweathermap",
        "z": "7dc53b7c-69de-45a4-b600-a41b4440979b",
        "wtype": "current",
        "city": "Goiabeiras",
        "country": "BR",
        "language": "pt",
        "x": 450,
```

Conceitos do Node-RED

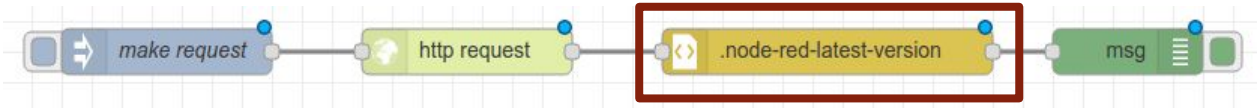
- **Mensagens**
  - As mensagens transmitidas entre nós no Node-RED são, por convenção, objetos JavaScript chamados **`msg`**, consistindo em um conjunto de propriedades nomeadas
  - Geralmente contêm uma propriedade **`msg.payload`** com a carga útil (*payload*) da mensagem.
  - Os nós podem anexar outras propriedades a uma mensagem, que pode ser usada para transportar outras informações para o próximo nó no fluxo.




```
{
   "_msgid": "248d9834.0f5668",
   "topic": "",
   "payload": [
       "v0.20.4"
   ],
   "statusCode": 200,
   "headers": {},
   "responseUrl": "https://nodered.org/",
   "responseCookies": {}
}
```

Conceitos do Node-RED

- **Nós**
  - São o principal bloco de construção dos fluxos no Node-RED.
  - Quando um fluxo está em execução, as mensagens são geradas, consumidas e processadas pelos nós.



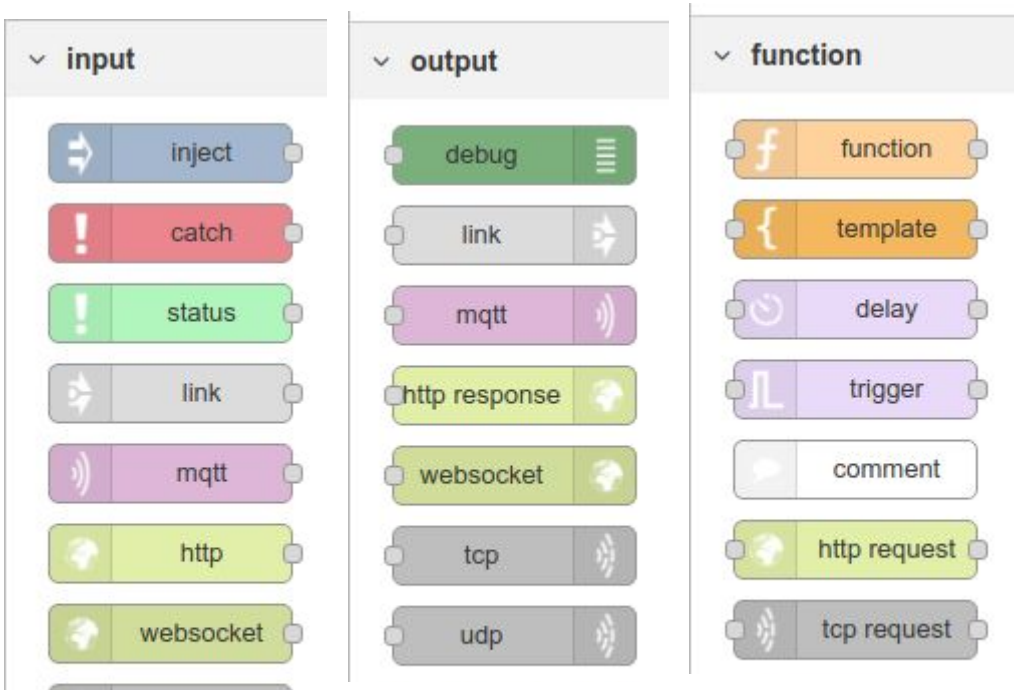

```
{
   "_msgid": "6cbbcaa4.0a9cc4",
   "topic": "",
   "payload": "<!DOCTYPE html>\n<html
lang=\"en\">\n<head>\n\t<meta...",
   "statusCode": 200,
   "headers": {
   },
   "responseUrl": "https://nodered.org/",
   "responseCookies": {}
}
```


Seletor html - `.node-red-latest-version`


```
{
   "_msgid": "248d9834.0f5668",
   "topic": "",
   "payload": [
      "v0.20.4"
   ],
   "statusCode": 200,
   "headers": {},
   "responseUrl": "https://nodered.org/",
   "responseCookies": {}
}
```

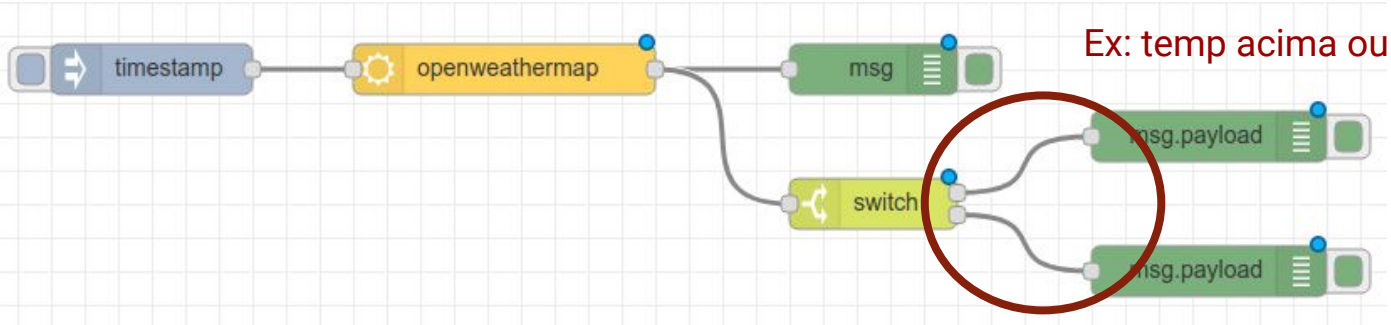
Conceitos do Node-RED

- **Nós**
  - 3 tipos básicos
  - **Nós de entrada** - gera mensagens para nós de recebimento de dados.
  - **Nós de saída** - consome mensagens, por exemplo, para enviar dados para um serviço externo
  - **Nós de processamento** - nós que processam dados de alguma forma, emitindo mensagens novas ou modificadas.
  - 2 categorias adicionais
  - **Nós de credenciais**
  - **Nós criados por usuários**



Conceitos do Node-RED

- **Fios**
  - Os fios definem as conexões entre os nós de entrada e saída em um fluxo.
  - Conectam a extremidade dos nós de saída às entradas dos nós de recebimento de dados.
  - É possível conectar mais de um nó à extremidade do nós de saída usando fios.
  - Quando vários nós são conectados a um terminal de saída, as mensagens são enviadas para cada nó conectado, na ordem em que foram conectadas à saída.



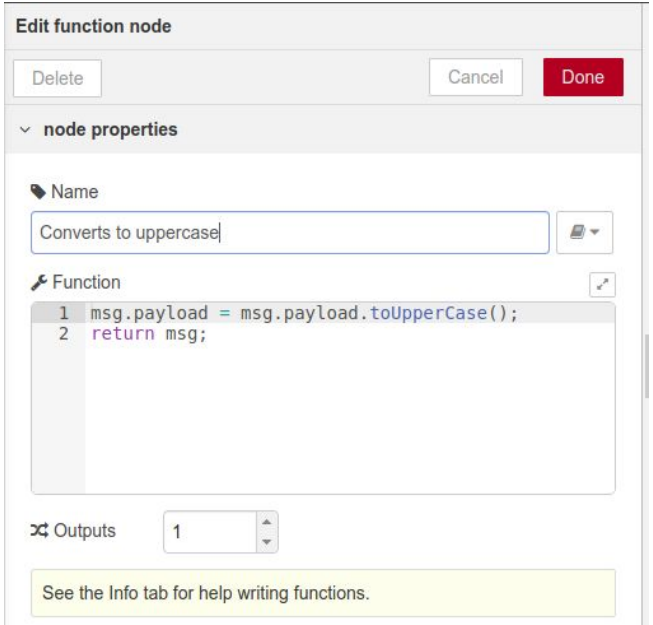
Ex: temp acima ou abaixo de 30º

Conceitos do Node-RED

- **Contexto**
  - As mensagens, até agora, eram a única maneira de obter dados dentro e fora dos nós.
  - Todavia, há uma exceção a essa regra -> `context`
  - O `context` está disponível apenas para os nós de função
  - É usado para armazenar dados na memória que podem ser acessados posteriormente
  - É importante para os nós que precisam manter um índice, contar ou somar dados nas mensagens.
  - Existem 2 tipos de contexto
    - Contexto local (`context`) - que pode ser acessado apenas pelo nó que criou o índice
    - Contexto do fluxo (`flow`) - que pode ser acessado por todos os nós de um fluxo
    - Contexto global (`global`) - que pode ser acessado por todos os nós de uma instância

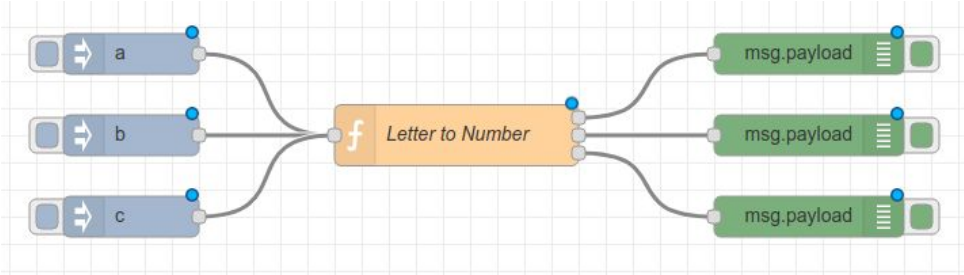
Conceitos do Node-RED

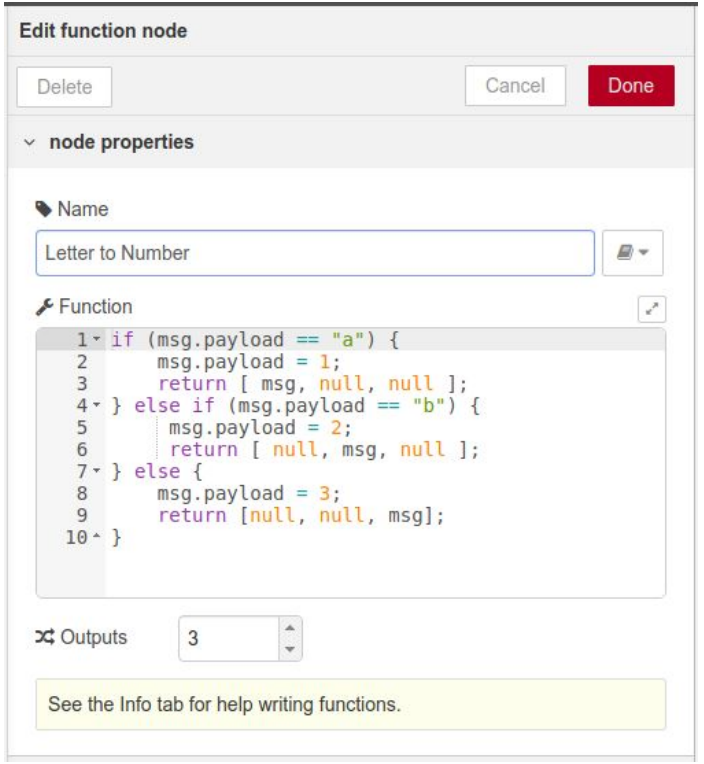
- **Nós de função**
  - Os nós de função são o “canivete suíço” do Node-RED
  - Podem ser utilizados quando não há um nó dedicado especificamente à tarefa que se deseja executar
  - São escritos em Javascript por meio de um editor de código disponível no Node-RED



Conceitos do Node-RED

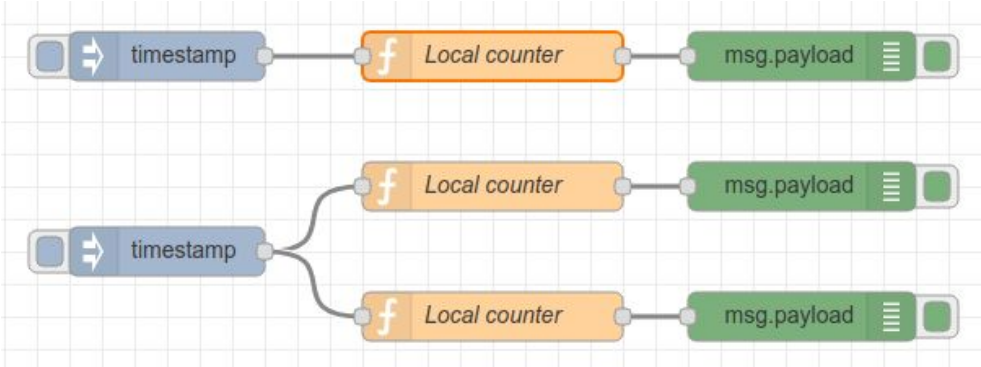
- **Nós de função**
  - Os nós de função também podem retornar múltiplos valores.

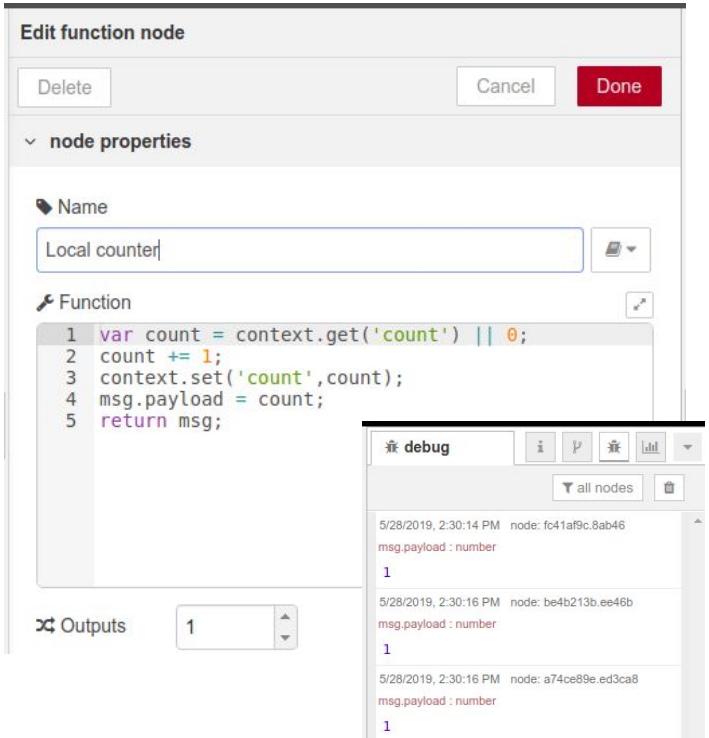




Conceitos do Node-RED

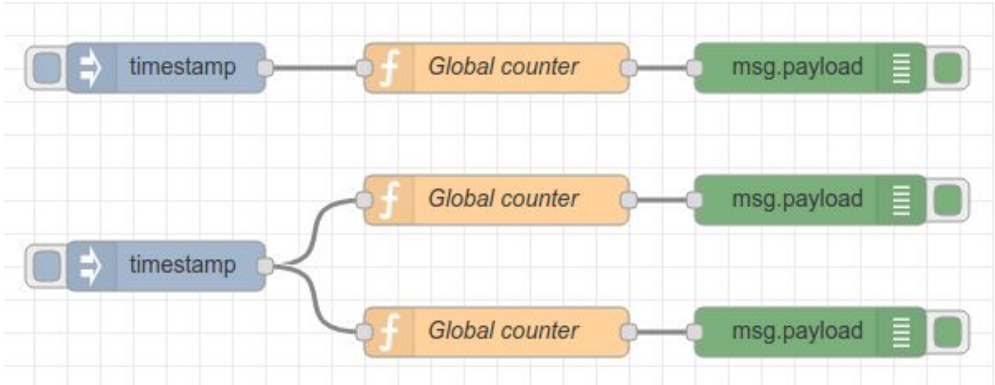
- **Nós de função - utilizando o contexto local (`context`)**

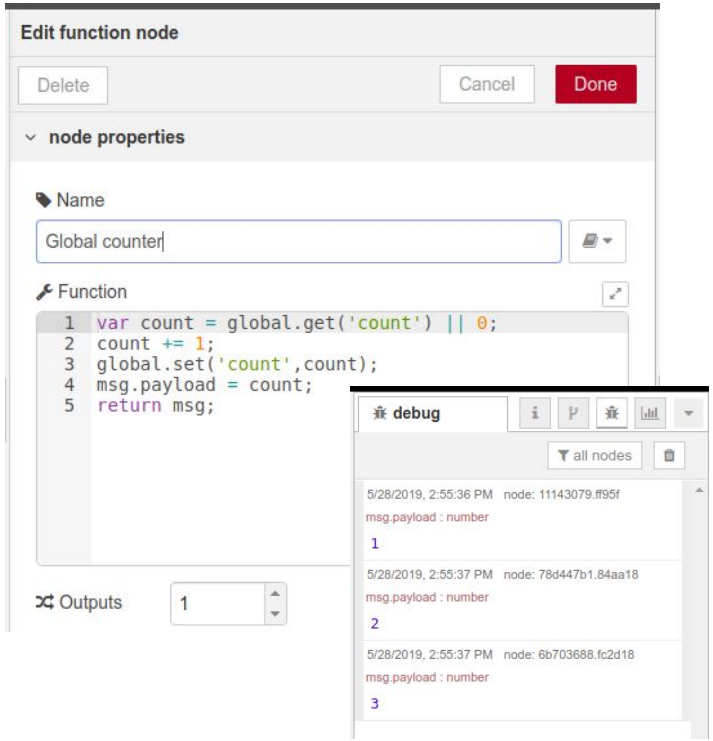




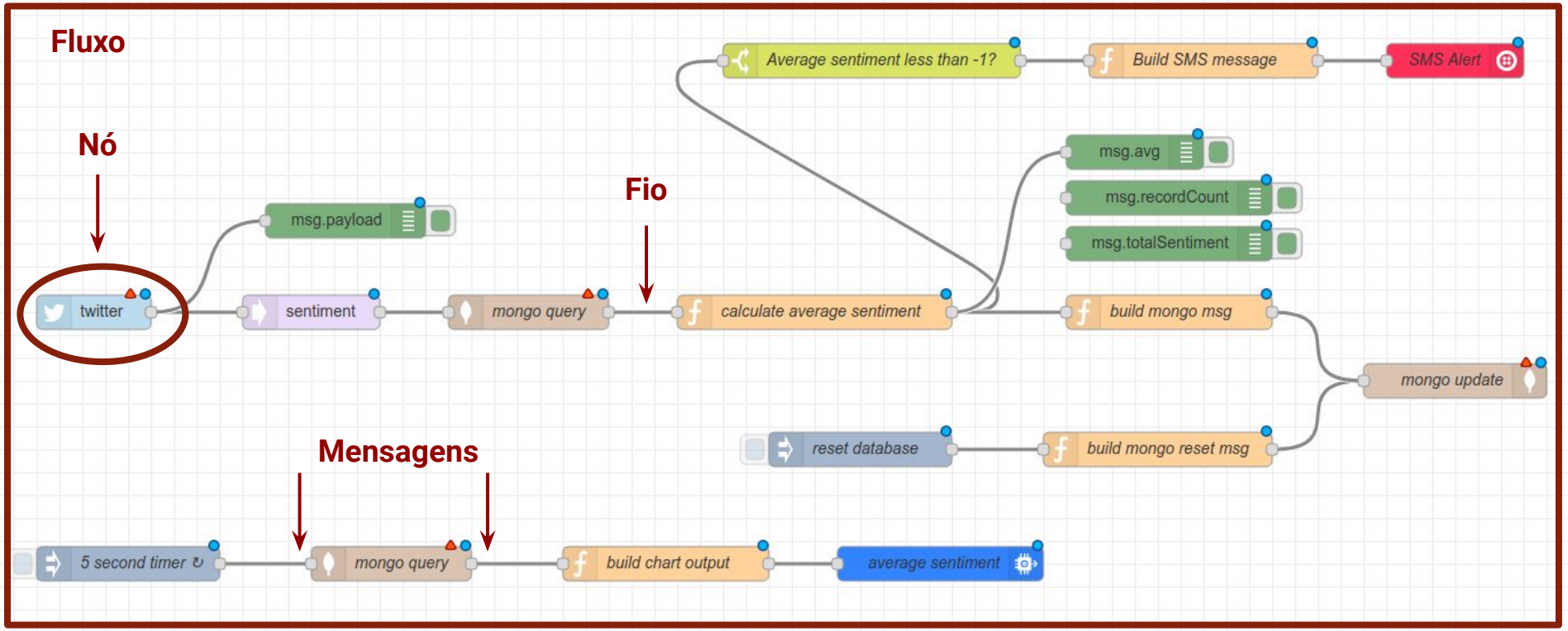
Conceitos do Node-RED

- **Nós de função - utilizando o contexto global (`global`)**





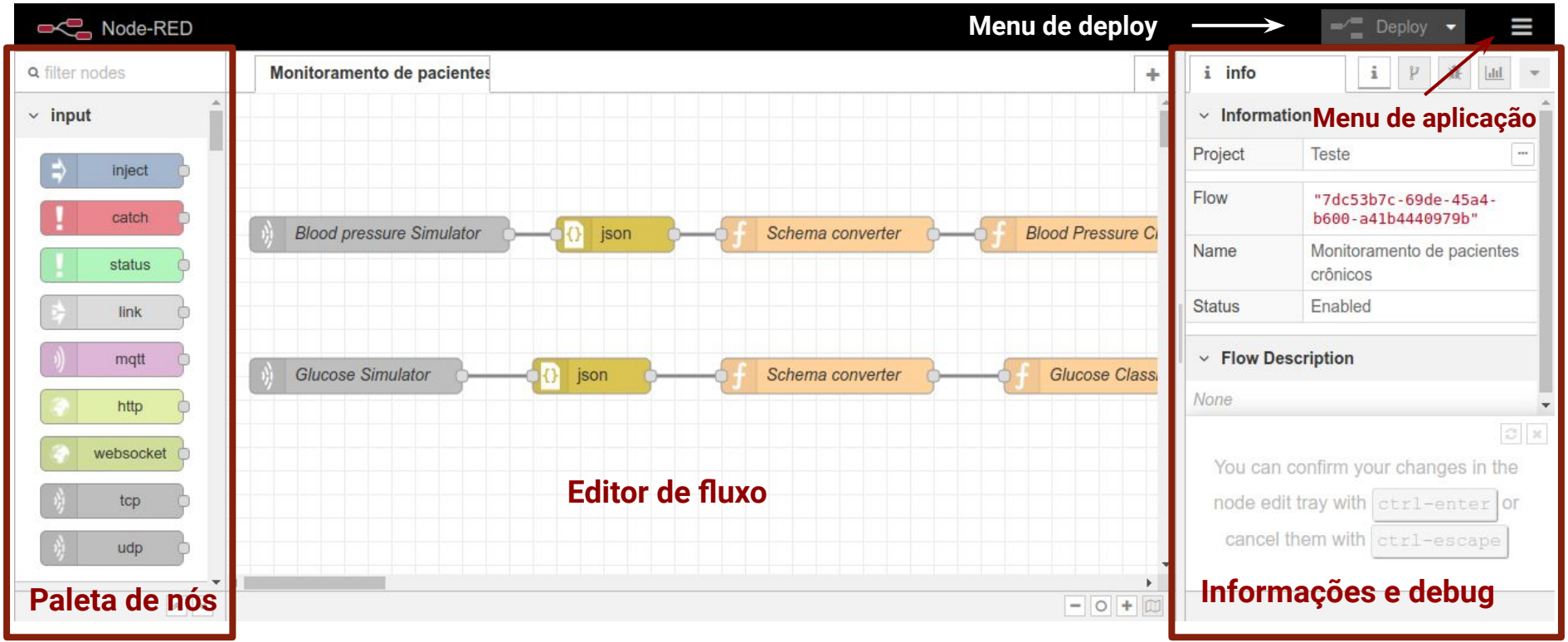
Conceitos do Node-RED



Instalação e execução

- Instalação Local
  a. Instalar Node.js (10.x LTS)
     - `sudo apt install curl build-essential`
     - `curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -]`
     - `sudo apt install nodejs`
  b. Instalar Node-RED
     - `sudo npm install -g --unsafe-perm node-red`
- Instalação docker
  a. `docker run -it -p 1880:1880 --name nodered nodered/node-red-docker`

A interface do Node-RED



Hands on: Hello World

1. Instalar o Node.js
2. Instalar o Node-RED
3. Executar os exemplos - https://github.com/jordanorc/curso-node-red/
    a. Primeiro Exemplo - `HelloWord.json`
    b. Utilizando nós de entrada e saída - `InputOutput.json`
    c. Comunicação com páginas web - `NodeVersion.json`
    d. Exemplo de uso do switch - `SwitchExample.json`
    e. Nós de função - String toUpperCase() - `StringUpperCase.json`
    f. Testando contexto - `Context.json`

O que é o JSON

- JSON -> JavaScript Object Notation
- Utilizado para representar um objeto JavaScript como String.
- É comumente usado por APIs da web para retornar dados.
- Elementos básicos do JSON.
  - `{` e `}` - delimita um objeto.
  - `[` e `]` - delimita um array.
  - `:` - separa chaves (atributos) de valores.
  - `,` - separa os atributos chave/valor.

O que é o JSON

- Os tipos de dados básicos do JSON são:
  - **string** - separados por aspas (duplas ou simples). Ex. "Brasil" ou 'Brasil'
  - **número** - sem aspas e pode ser **inteiro** ou **real**. Ex. 1 (inteiro) ou 23.454 (real)
  - **booleano** - tipo lógico normal, pode assumir valores **true** ou **false**.
  - **nulo** - valor para representar nulo. Ex. { "nome" : null }
  - **object**: É um conjunto de pares nome/valor.
  - **array**: utilizados para elementos ordenados.

O que é o JSON

- JSON no Node-RED
  - Usualmente as mensagens trocadas pelos nós seguem o formato JSON
  - Se o JSON estiver representado como String é necessário realizar uma conversão antes de utilizá-lo.
  - O Node-RED fornece um nó para converter string -> json -> string

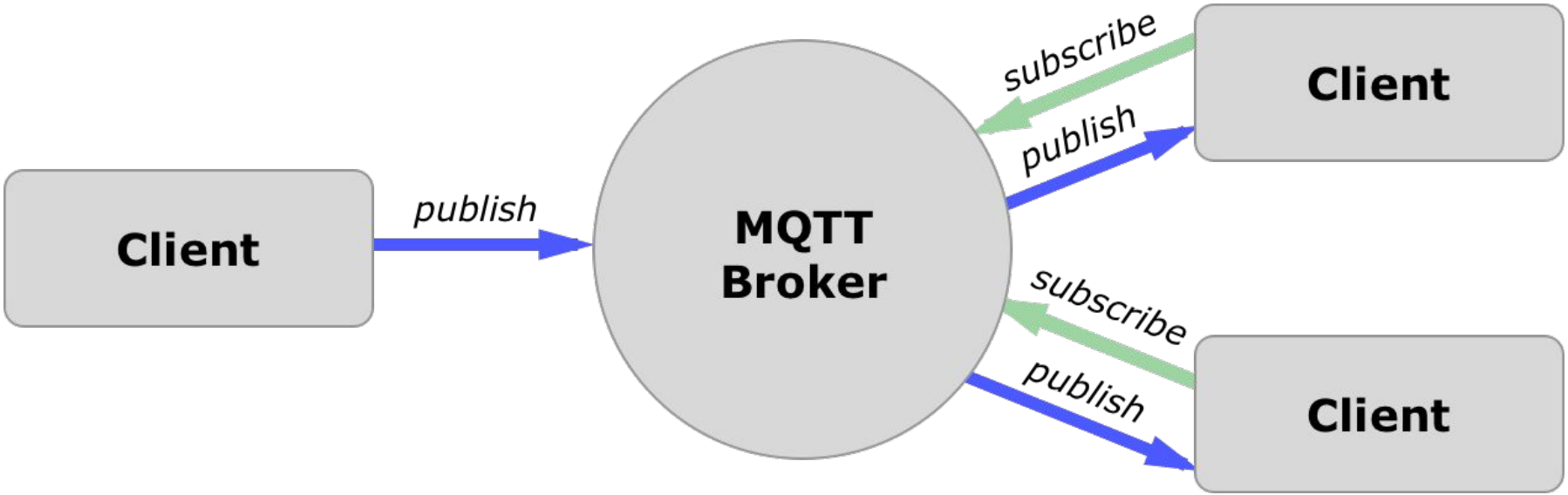
O que é o MQTT

- É um protocolo comunicação Machine-to-Machine (M2M), ou máquina a máquina
- Inventado e desenvolvido inicialmente pela IBM no final dos anos 90
- Era utilizado para conectar sensores em pipelines de petróleo a satélites
- Permite a comunicação entre pares de maneira assíncrona
- Por ser leve, permite a implementação de clientes em hardware de capacidade restrita
- Sua flexibilidade possibilita o suporte a diversos cenários de aplicativo para dispositivos e serviços de IoT.

O que é o MQTT

- Terminologia
  - **Client** - pode ser qualquer dispositivo que se conecte a um *broker* - de um microcontrolador para um servidor.
  - **Broker** - é responsável por rotear, filtrar e distribuir mensagens para as partes apropriadas (clientes).
  - **Topic** - cada mensagem publicada por um cliente é enviada para um tópico específico no *broker*.
  - **QoS** - O Quality of Service especifica como o aplicativo deve tentar publicar mensagens em um *broker* MQTT.

MQTT no Node-RED: Eclipse Mosquitto



O que é o MQTT

- **QoS - Quality of Service**
  - O MQTT especifica os seguintes três níveis de entrega de mensagens:
  - QoS 0: Enviar apenas uma vez.
    - Método *fire* and *forget*
    - Nível de mensagem padrão que não garante a entrega de mensagens.
    - Por exemplo, o envio de dados do sensor em QoS 0 pode resultar em perda de medidas. É incomum, mas pode acontecer.

O que é o MQTT

- **QoS - Quality of Service**
  - QoS 1
    - Entregue pelo menos uma vez.
    - Esse nível garante que a mensagem seja entregue, mas não impõe a entrega única.
  - QoS 2
    - Entregue exatamente uma vez: este nível garante a entrega correta.
  - Um caso de uso comum para QoS 1 e QoS 2 é a distribuição da configuração inicial ou das informações do canal para os dispositivos recém-conectados.

O que é o MQTT

- **Retain**
  - É possível definir que uma mensagem seja retida
  - Ela será recebida toda vez que um novo assinante começar a ouvir em um canal.
  - Pode ser muito útil para definir opções de inicialização ou configuração.
- **Last Will**
  - Permite saber se um dispositivo ainda está conectado a um *broker*.
  - Para isto, é preciso configurar uma mensagem para ser enviada quando o dispositivo for desconectado do servidor por um período de tempo pré-especificado.

Eclipse Mosquitto - Instalação

- Disponível nos repositórios do Ubuntu por padrão
- Instalando o server
  - `sudo apt install mosquitto`
- Instalando o client
  - `sudo apt-get install mosquitto-clients`
- Inscrever-se em um tópico
  - `mosquitto_sub -t "test"`
- Publicar em um tópico
  - `mosquitto_pub -m "message from mosquitto_pub client" -t "test"`

Hands on: Node-RED e MQTT

1. Instalar o Mosquitto
2. Testar pub/sub
3. Criar um aplicativo no Node-RED que publique mensagens em um tópico no MQTT
4. Criar um aplicativo no Node-RED que inscreva-se para receber mensagens publicadas em um tópico MQTT