



Projeto 08

Coisas da Casa – Resumo da Ópera

Jan K. S. – janks@puc-rio.br

ENG4051 – Projeto Internet das Coisas

Sensor de Luz



```
int leitura = analogRead(pino);  
int porcentagemLuz = map(leitura, 0, 4095, 0, 100);
```

Millis

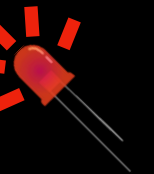
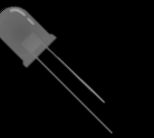
```
unsigned long instanteAnterior = 0;  
  
void loop () {  
    unsigned long instanteAtual = millis();  
    if (instanteAtual > instanteAnterior + 1000) {  
        Serial.println("+1 segundo");  
        instanteAnterior = instanteAtual;  
    }  
}
```

String

```
String texto1 = "Olá, mundo!";  
int numero = 100 * 2;  
String texto2 = String(numero);  
int numero2 = texto2.toInt() + 42;  
  
String texto3 = "aaa" + texto2;  
  
bool ehIgual = texto2 == texto3;  
bool comeceComOla = texto1.startsWith("Olá");  
  
char caracter = texto1[2]; // 'á'  
int totalCaracteres = texto1.length(); // 11  
  
String trecho = texto1.substring(0, 3); // "Olá"  
String trechoFinal = texto1.substring(5); // "mundo!"  
  
String texto4 = "  abc abc  \n";  
texto4.replace("ab", "AB"); // "ABc ABc"
```

LED

```
void setup () {  
    pinMode(pinoLED, OUTPUT);  
    digitalWrite(pinoLED, HIGH);  
}  
  
digitalWrite(pinoLED, LOW);
```



Serial

```
void setup () {  
    Serial.begin(115200); while(!Serial);  
}  
  
void loop () {  
    if (Serial.available() > 0) {  
        String texto = Serial.readStringUntil('\n');  
        Serial.println(texto);  
    }  
}
```



Botão

```
#include <GButton.h>
```

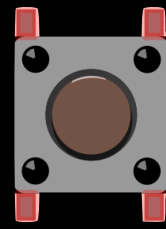
```
GButton botao(1);
```

```
void botaoPressionado (GButton& botaoDoEvento) {  
    Serial.println("Botão foi pressionado!");  
}
```

```
void botaoSolto (GButton& botaoDoEvento) {  
    Serial.println("Botão foi solto!");  
}
```

```
void setup () {  
    Serial.begin(115200);  
    botao.setPressHandler(botaoPressionado);  
    botao.setReleaseHandler(botaoSolto);  
}
```

```
void loop () {  
    botao.process();  
}
```



LED RGB

```
rgbLedWrite(RGB_BUILTIN, vermelho, verde, azul);
```



Sensor de Movimento

```
#include <GButton.h>
```

```
GButton sensor(21);
```

```
void movimento (GButton& sensor) {  
    Serial.println("Movimento detectado!");  
}
```

```
void inercia (GButton& sensor) {  
    Serial.println("Inércia detectada!");  
}
```

```
void setup () {  
    Serial.begin(115200);  
    sensor.setPressHandler(inercia);  
    sensor.setReleaseHandler(movimento);  
}
```

```
void loop () {  
    sensor.process();  
}
```



Botão com Passagem de Parâmetro

```
void minhaFuncao (int x) {  
    Serial.println(x);  
}
```

```
void setup () {  
    Serial.begin(115200);  
    botao.setPressHandler([](GButton &b){ minhaFuncao(42); });  
}
```

Verificação Manual

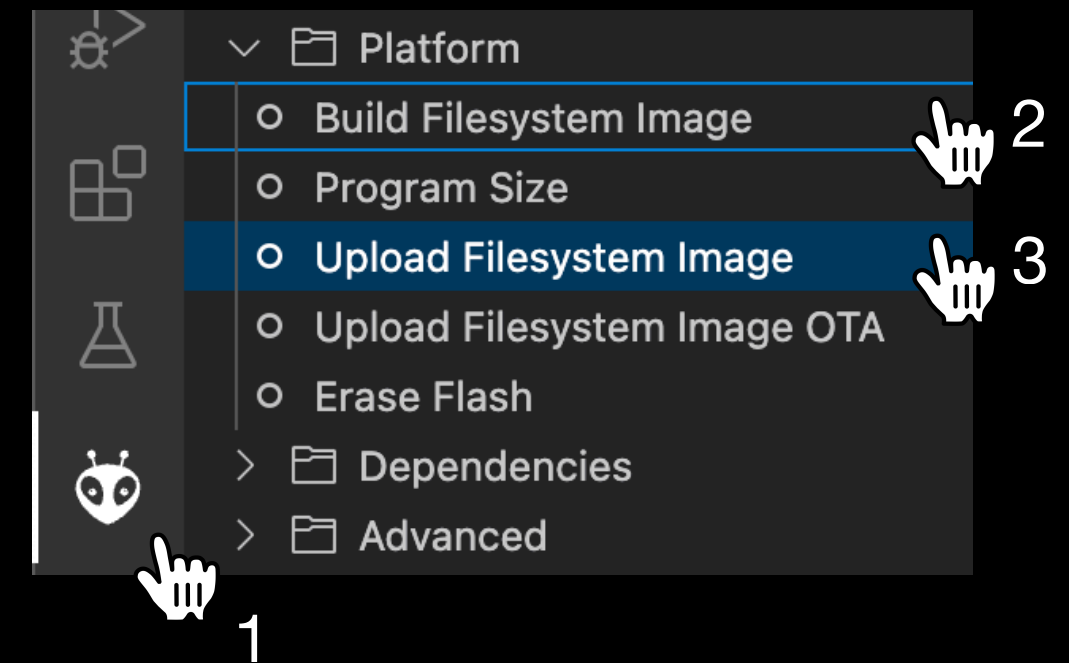
```
botao.isPressed();  
sensor.isPressed();
```

Setup

```
#include <LittleFS.h>
#include <ArduinoJson.h>

void setup() {
  if (!LittleFS.begin()) {
    Serial.println("LittleFS falhou!");
    while (true) {};
  }
}
```

Envio de Arquivos para ESP32



Leitura

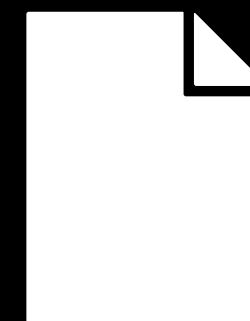
```
File arquivo = LittleFS.open("/arquivo.txt", "r");
if (!arquivo) {
  Serial.println("Arquivo falhou!");
  while (true) {};
}
String conteudo = arquivo.readString();
arquivo.close();
```

Escrita

```
File arquivo = LittleFS.open("/arquivo.txt", "w");
arquivo.println("IoT");
arquivo.print("PUC-Rio ");
arquivo.println(2020 + 4);
arquivo.close();
```

Arquivo JSON

```
JsonDocument dados;
deserializeJson(dados, arquivo); // leitura
serializeJson(dados, arquivo);   // escrita
```



Arquivos no LittleFS

WiFi

```
#include <WiFi.h>

void reconectarWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    WiFi.begin("NOME DA REDE", "SENHA DA REDE");

    Serial.print("Conectando ao WiFi...");
    while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      delay(1000);
    }
    Serial.print("conectado!\nEndereço IP: ");
    Serial.println(WiFi.localIP());
  }
}

void setup () {
  Serial.begin(115200); delay(500);

  reconectarWiFi();
}

void loop () {
  reconectarWiFi();
}
```



Rede WiFi

Serialização

```
JsonDocument dados;
dados["número"] = 12345;
dados["texto"] = "IoT";

String textoJson;
serializeJson(dados, textoJson);
serializeJson(dados, Serial);

String texto_json2 = "[10, 20, 30]";
JsonDocument lista;
deserializeJson(lista, texto_json2);
```

Json

```
#include <ArduinoJson.h>
```

```
JsonDocument dados;
dados["número"] = 12345;
dados["texto"] = "IoT";
```

```
String meuTexto = dados["texto"];
int meuNumero = dados["número"];
```

```
JsonDocument lista;
lista.add(10);
lista.add(20);
for (unsigned int i = 0; i < lista.size(); i++) {
  int elemento = lista[i];
  Serial.println(elemento);
}
```



Servidor

```
#include <WebServer.h>
#include <uri/UriBraces.h>
```

```
WebServer servidor(80);
```

```
void setup () {
  reconectarWiFi();
```

```
  servidor.on("/inicio", HTTP_GET, pagina1);
  servidor.on("/contato", HTTP_GET, pagina2);
  servidor.on("/contato", HTTP_POST, tratarDados);
  servidor.on(UriBraces("/parametros/{}/{ }"), HTTP_GET, pagina3);
  servidor.begin();
```

```
}
```

```
void loop () {
  reconectarWiFi();
  servidor.handleClient();
}
```

Tratamento de Dados POST

```
void tratarDados () {
  String email = servidor.arg("email");
  String mensagem = servidor.arg("mensagem");
  // faz alguma coisa com esses dados...
  // redireciona para uma outra página
  servidor.sendHeader("Location", "/inicio");
  servidor.send(303);
}
```

Página Simples

```
void pagina1 () {
  servidor.send(200, "text/html", "Bem-vindo!");
}
```

Página com HTML

```
void pagina2 () {
  File arquivo = LittleFS.open("/pagina.html", "r");
  if (!arquivo) {
    servidor.send(500, "text/html", "Erro no HTML");
    return;
  }
  String html = arquivo.readString();
  arquivo.close();
  html.replace("{{nome}}", "Jan");
  servidor.send(200, "text/html", html);
}
```



Servidor Web

Página Com Parâmetros

```
void pagina3 () {
  String texto = servidor.pathArg(0);
  int numero = servidor.pathArg(1).toInt();
  servidor.send(200, "text/html", "Dados ok");
}
```

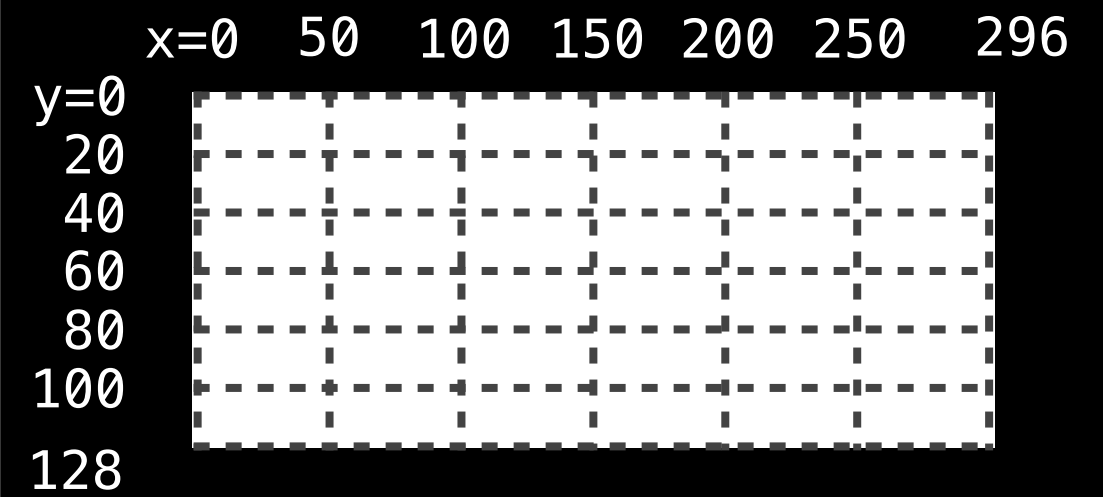
Setup

```
#include <GxEPD2_BW.h>
#include <U8g2_for_Adafruit_GFX.h>

U8G2_FOR_ADAFRUIT_GFX fontes;
GxEPD2_290_T94_V2 modeloTela(10, 14, 15, 16);
GxEPD2_BW<GxEPD2_290_T94_V2, GxEPD2_290_T94_V2::HEIGHT> tela(modeloTela);

void setup() {
  tela.init();
  tela.setRotation(3);
  tela.fillScreen(GxEPD_WHITE);
  tela.display(true);

  fontes.begin(tela);
  fontes.setForegroundColor(GxEPD_BLACK);
}
```



Fontes de Símbolos

u8g2_font_open_iconic_all_4x_t



https://github.com/olikraus/u8g2/wiki/fntpic/u8g2_font_open_iconic_all_4x_t.png

Fontes de Texto

u8g2_font_helvB24_te
u8g2_font_helvB18_te
u8g2_font_helvB14_te
u8g2_font_helvB12_te

Textos

```
fontes.setFont( u8g2_font_helvB24_te );
fontes.setFontMode(1);
fontes.setCursor(x, y);
fontes.print("Meu texto");

tela.display(true); // SEMPRE CHAMAR NO FINAL!
```

Desenhos

```
tela.drawLine(x1, y1, x2, y2, cor);

tela.fillCircle(x, y, raio, cor);
tela.drawCircle(x, y, raio, cor);

tela.fillRect(x, y, comprimento, altura, cor);
tela.drawRect(x, y, comprimento, altura, cor);

tela.fillTriangle(x1, y1, x2, y2, x3, y3, cor);
tela.drawTriangle(x1, y1, x2, y2, x3, y3, cor);

tela.display(true); // SEMPRE CHAMAR NO FINAL!
```

Display ePaper
2.9" WeAct



Setup

```
#include <WiFi.h>
#include <WiFiClient.h>
#include "certificados.h"
#include <MQTT.h>
```

```
WiFiClient conexao;
MQTTClient mqtt(1000);
```

```
void setup() {
  Serial.begin(115200);
  delay(500);
```

```
  reconnectarWiFi();
  conexaoSegura.setCACert(certificado1);
```

```
  mqtt.begin("mqtt.janks.dev.br", 1883, conexao);
  mqtt.onMessage(recebeuMensagem);
  mqtt.setKeepAlive(10);
  mqtt.setWill("tópico da desconexão", "conteúdo");
```

```
  reconnectarMQTT();
```

```
}
```

```
void loop() {
  reconnectarWiFi();
  reconnectarMQTT();
  mqtt.loop();
}
```

Reconectar

```
void reconectarMQTT() {
  if (!mqtt.connected()) {
    Serial.print("Conectando MQTT...");
    while(!mqtt.connected()) {
      mqtt.connect("SEU ID", "LOGIN", "SENHA");
      Serial.print(".");
      delay(1000);
    }
    Serial.println(" conectado!");

    mqtt.subscribe("topico1"); // qos = 0
    mqtt.subscribe("topico2/+/parametro", 1); // qos = 1
  }
}
```



Recebimento

```
void recebeuMensagem(String topico, String conteudo) {
  Serial.println(topico + ": " + conteudo);
}
```

Envio

```
mqtt.publish("topico", "conteúdo"); // retain = false, qos = 0
mqtt.publish("topico2/1234/parametro", "conteúdo 2", false, 1);
```

mqtt → { topic: "tópico1/10", payload: "conteúdo" }

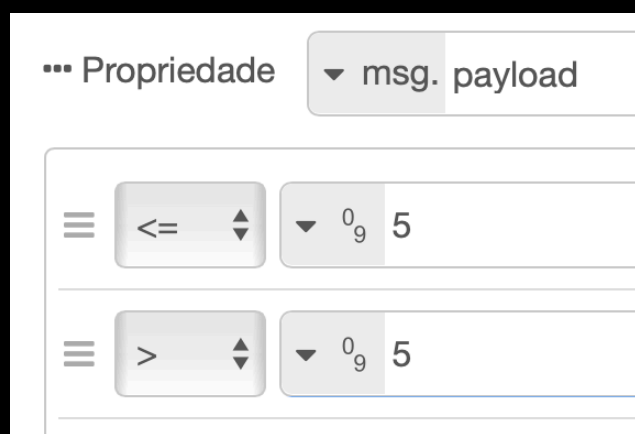
{ topic: "tópico2", payload: "conteúdo" } → mqtt

{ topic: "tópico4", payload: "Jan K. S." } → debug

inject → { topic: "tópico3", payload: "teste!" }

{ payload: "texto" } → change → { payload: "novo texto" }

{ payload: 8 } → switch (não emite nada) → { payload: 8 }



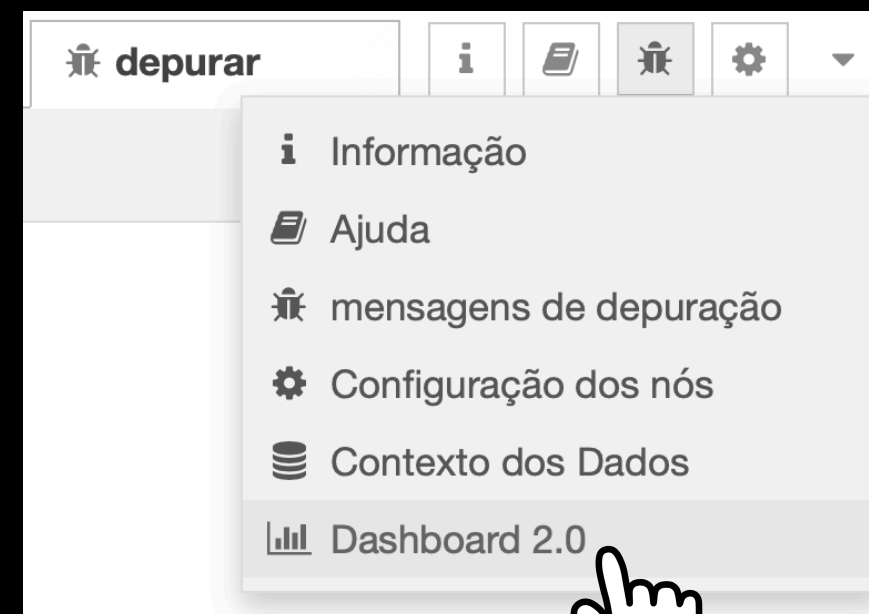
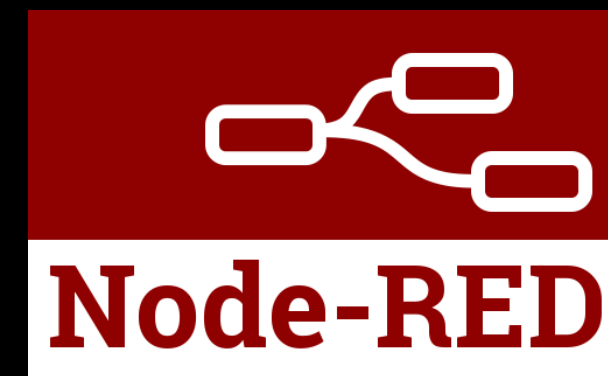
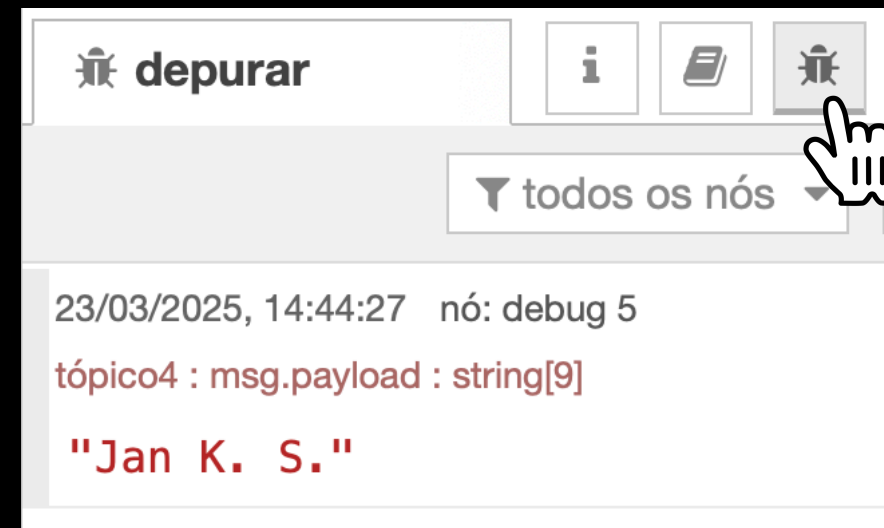
{ payload: [10, 20, 30] } → split → { payload: 10 }
{ payload: 20 }
{ payload: 30 }

{ payload: 10 }
{ payload: 20 }
{ payload: 30 } → join → { payload: [10, 20, 30] }

receiver → { content: "Olá, Node-RED!", ... } { payload: <imagem> } → image

{ payload: 42 } → template → sender

```
{  
  "content": "Valor = {{payload}}",  
  "chatId": "ID DO SEU CHAT",  
  "type": "message"  
}
```



Setup – Códigos

```
#include <GxEPD2_BW.h>
#include <BarcodeGFX.h>
#include <QRCodeGFX.h>
```

```
GxEPD2_290_T94_V2 modeloTela(10, 14, 15, 16);
GxEPD2_BW<GxEPD2_290_T94_V2, GxEPD2_290_T94_V2::HEIGHT> tela(modeloTela);
BarcodeGFX codigoBarras(tela);
QRCodeGFX qrcode(tela);
```

```
void setup() {
  tela.init();
  tela.setRotation(3);

  tela.fillScreen(GxEPD_WHITE);
  tela.display(true);
}
```

Desenho do Código de Barras

```
codigoBarras.setScale(2);
codigoBarras.draw("7896065880069", x, y, altura);

tela.display(true);
```



Desenho do QR Code

```
qrcode.setScale(2);
qrcode.draw("https://google.com", x, y);

tela.display(true);
```



Setup – Leitor

```
void setup() {
  Serial.begin(115200); delay(500);
  Serial1.begin(9600, SERIAL_8N1, 47, 48);

  Serial1.println("~M00910001.");
  delay(100);
  Serial1.println("~M00210001.");
  delay(100);
  Serial1.println("~M00B00014.");
}
```



Resposta do Leitor

```
void loop() {
  if (Serial1.available() > 0) {
    String texto = Serial1.readStringUntil('\n');
    texto.trim();
    if (texto.length() > 5) {
      Serial.println("Resposta do leitor: " + texto);
      long long code = strtoll(texto.c_str(), nullptr, 10);
    }
  }
}
```



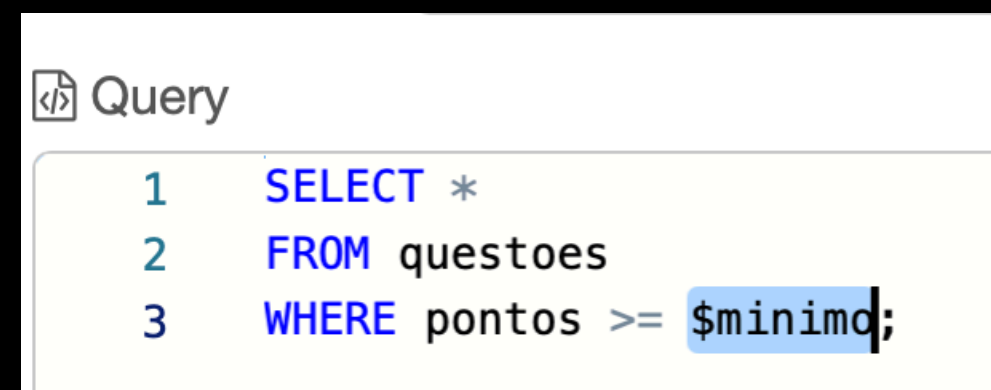
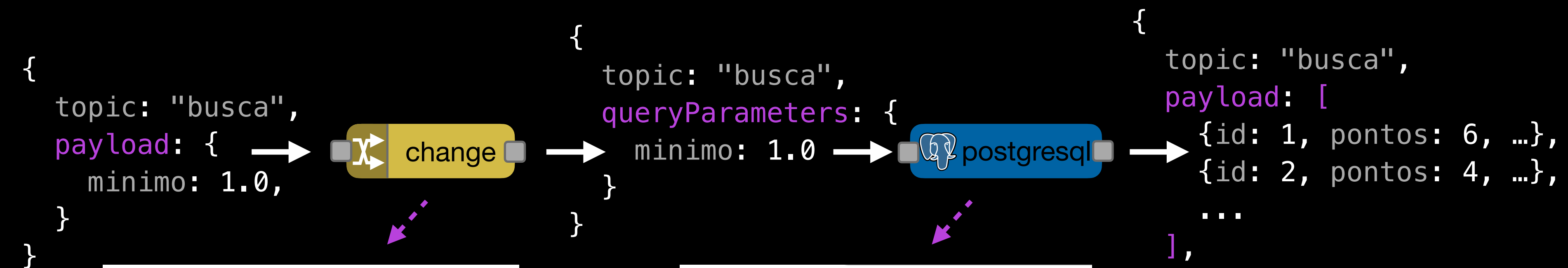
Inserção de Dados

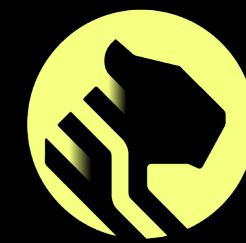
```
INSERT INTO provas (id, nome, inicio) VALUES
(1, 'P1 de Programação', '2024-04-22 17:00:00'),
(2, 'P2 de Programação', '2024-05-17 17:00:00');
```

Busca de Dados

```
SELECT *
FROM questoes;
```

```
SELECT numero, pontos, enunciado
FROM questoes
WHERE pontos >= 2.0 AND id = 1
ORDER BY id_prova ASC, numero ASC;
```





Timescale

Busca com Janelamento

```
SELECT
  time_bucket('1 hour', data_hora) AS time,
  AVG(luz) AS media_luz,
  SUM(movimento) AS soma_movimento
FROM dados
WHERE data_hora > NOW() - INTERVAL '1 day'
GROUP BY time
ORDER BY time ASC;
```

Outras Funções de Janelamento

```
time_bucket('1 hour', data_hora) AS time
first(temperatura, time) AS primeira_temperatura
last(temperatura, time) AS ultima_temperatura

time_bucket_gapfill('1 hour', data_hora) AS time
```



Grafana

Filtro de Tempo do Grafana

```
SELECT
  time_bucket('1 minute', data_hora) AS time,
  AVG(luz) AS media_luz
FROM dados
WHERE $__timeFilter(data_hora)
GROUP BY time
ORDER BY time ASC;
```

```
SELECT saldo
FROM clientes
WHERE nome = '$nomeCliente'
```

The screenshot shows the Grafana configuration interface for a template variable. On the left, a table lists the variable 'nomeCliente' with a value of 'None'. A red arrow points from this table to the configuration panel on the right. The configuration panel has a 'Select variable type' dropdown set to 'Text box'. Under the 'General' section, the 'Name' field is set to 'nomeCliente' and the 'Label' field is set to 'Nome do Cliente'.

Nome do Cliente	Enter variable value
	None

Select variable type
Text box

General

Name
The name of the template variable. (Max. 50)
nomeCliente

Label
Optional display name
Nome do Cliente