



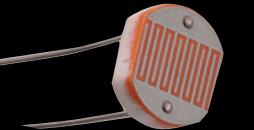
Projeto 08

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 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 comecaComOla = 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"
```

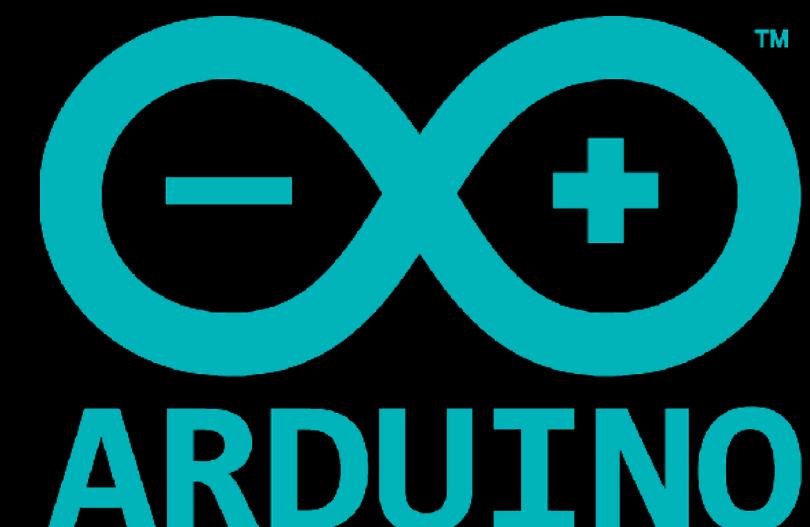
LED

```
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);  
    }  
}
```



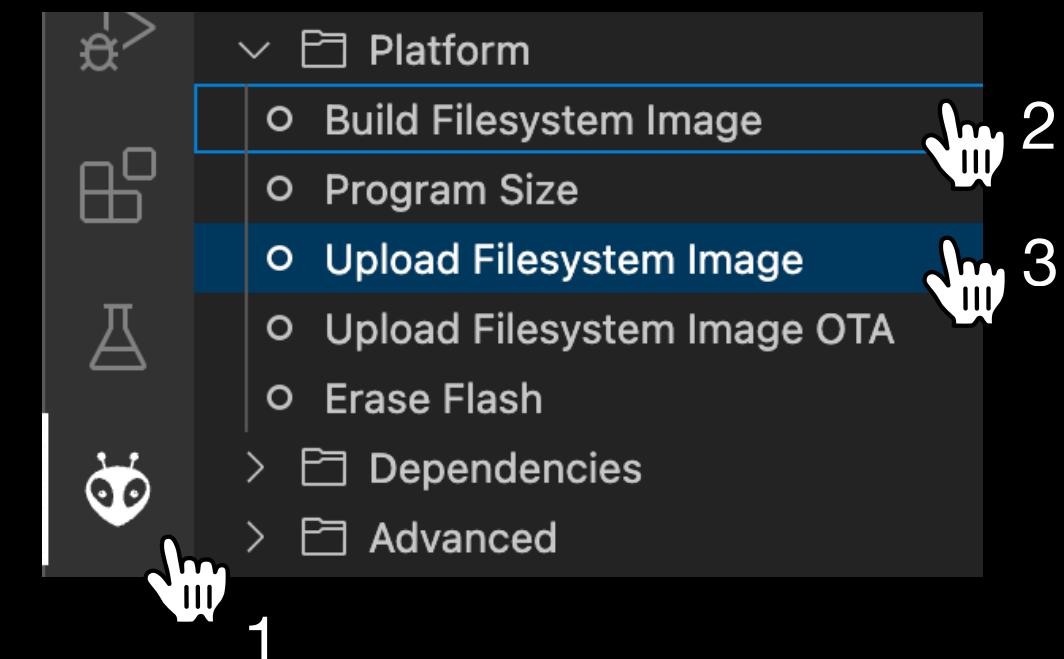


Setup

```
#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 reconnectarWiFi() {
  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);

  reconnectarWiFi();
}

void loop () {
  reconnectarWiFi();
```



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);
}
```

O JSON

```

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

WebServer servidor(80);

void setup () {
    reconnectWiFi();
    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 () {
    reconnectWiFi();
    servidor.handleClient();
}

```

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

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!");
}

```



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.fillRect(0, 0, 296, 128, GxEPD_WHITE);
    tela.display(true);

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

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!
```

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

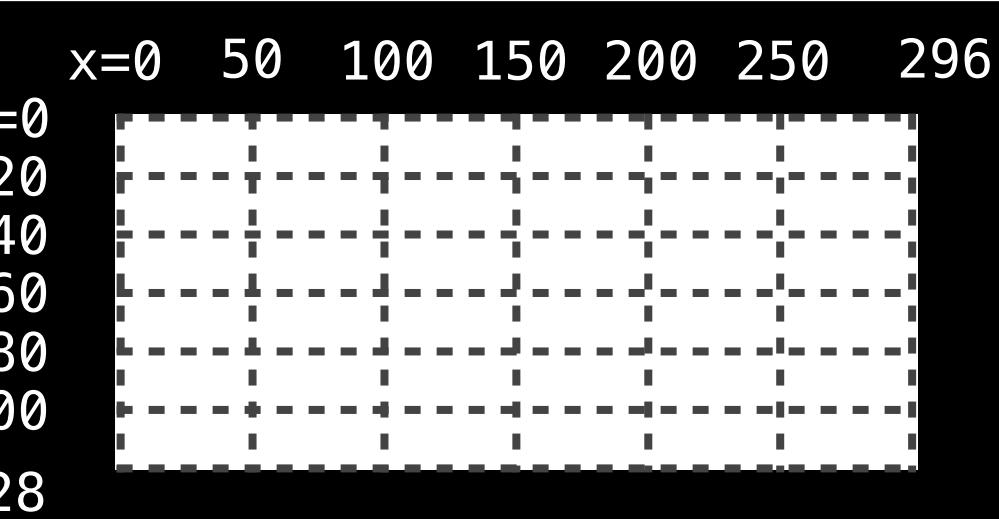
```
u8g2_font_helvB24_te
u8g2_font_helvB18_te
u8g2_font_helvB14_te
u8g2_font_helvB12_te
```

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!
```

Display ePaper
2.9" WeAct



```

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

WiFiClient conexao;
MQTTClient mqtt(1000);

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

  reconnectWiFi();
  conexaoSegura.setCACert(certificado1);

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

  reconnectMQTT();
}

void loop() {
  reconnectWiFi();
  reconnectMQTT();
  mqtt.loop();
}

```

Setup

Reconectar

```

void reconnectarMQTT() {
  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);
}

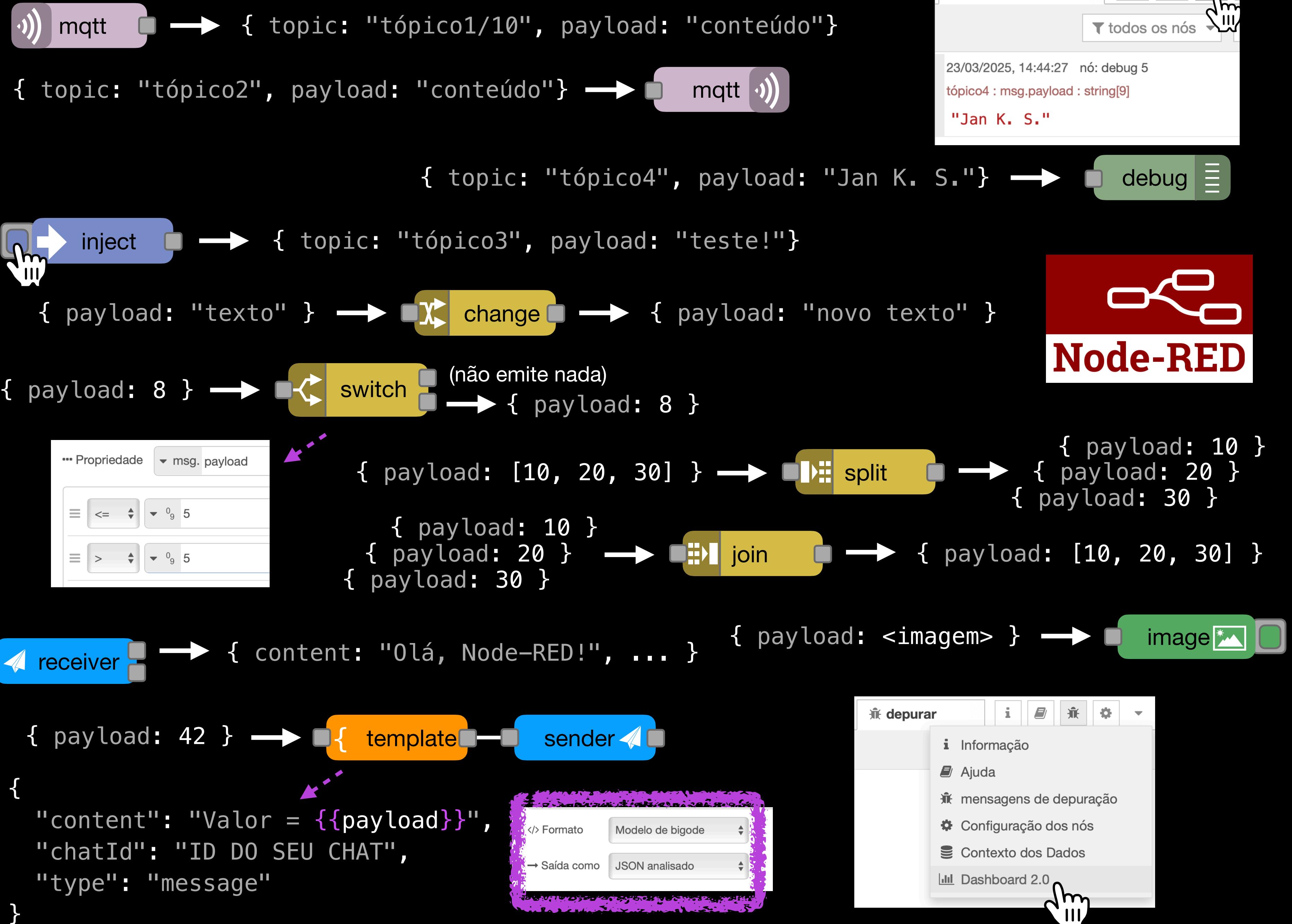
```

Envio

```

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

```



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);
```

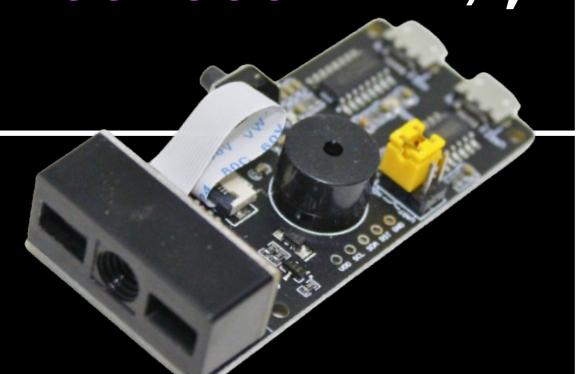
```
void setup() {
    tela.init();
    tela.setRotation(3);

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

Setup – Leitor

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

    Serial1.println(~M0091001.);
    delay(100);
    Serial1.println(~M0021001.);
    delay(100);
    Serial1.println(~M00B00014.);
}
```



Desenho do Código de Barras

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

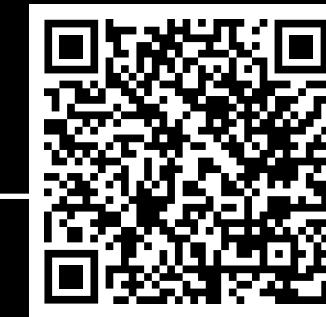
tela.display(true);
```



Desenho do QR Code

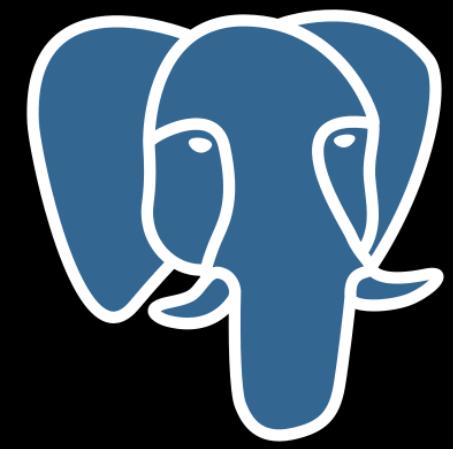
```
qrcode.setScale(2);
qrcode.draw("https://google.com", x, y);

tela.display(true);
```



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);
        }
    }
}
```



PostgreSQL

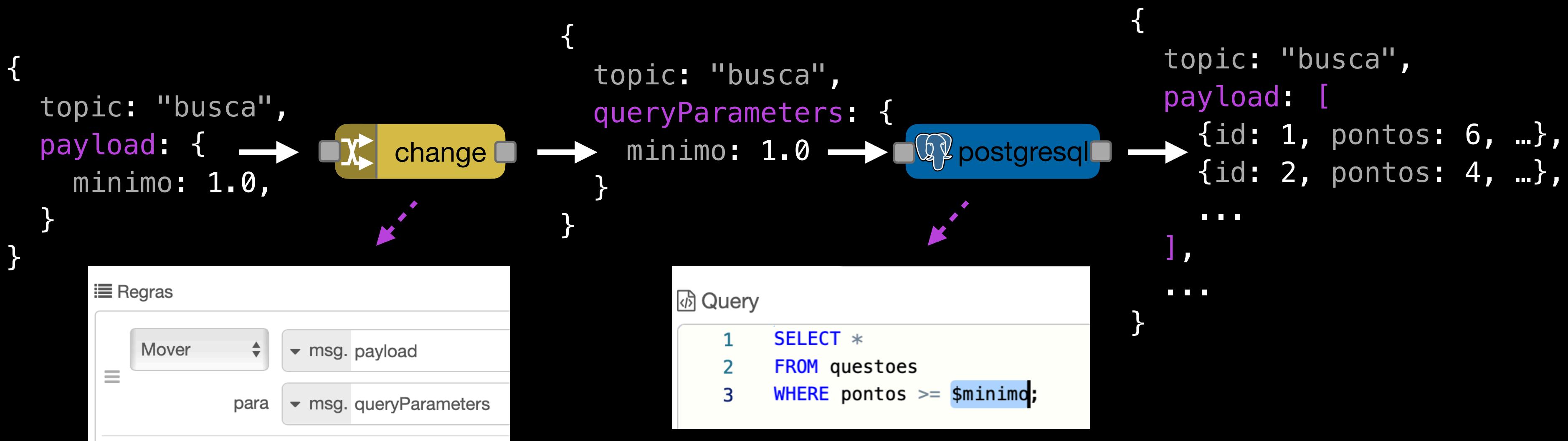
Inserção de Dados

```
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

```
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;
```



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;
```

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
```

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

The screenshot shows the Grafana interface for managing template variables. A dropdown menu is open under the 'Select variable type' label, with 'Text box' selected. Other options visible in the dropdown include 'None' and 'Select variable type'. Below the dropdown, there are fields for 'Name' (set to 'nomeCliente') and 'Label' (set to 'Nome do Cliente').