OFICINA DE INTERNET DAS COISAS

DBSERVER DBLAB

A OFICINA

IMERSÃO NO **COTIDIANO** DO LAB	**TECNOLOGIAS** PARA INTERNET DAS COISAS
PROGRAMAÇÃO DE SISTEMAS EMBARCADOS	MONTAGEM DE **PROTÓTIPOS**

RESULTADOS

REFLEXÃO
SOBRE AS
POTENCIALIDADES
DAS TECNOLOGIAS
VIVENCIADAS

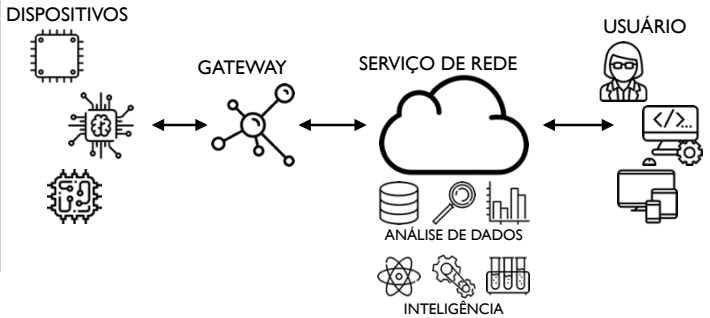
PALESTRA

UBILAB

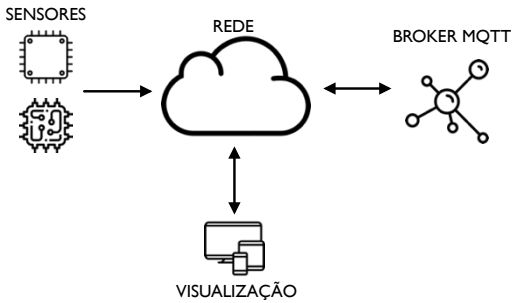
Prof. Eduardo Pellanda:

"Cidade que sente"

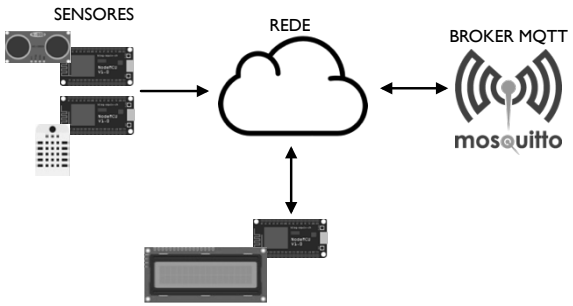
ARQUITETURA PARA IOT



A OFICINA

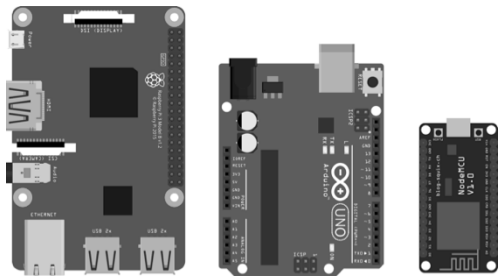


A OFICINA



COMPONENTES

PLATAFORMAS PARA
DESENVOLVIMENTO DE PROTÓTIPOS



RASPBERRY PI
LINUX

ARDUINO

ESP8266
WI-FI

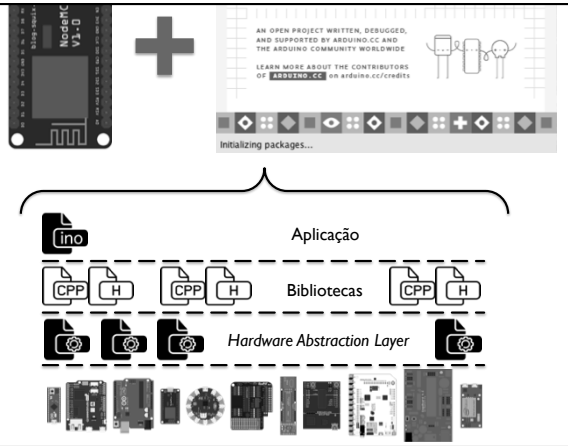
A OFICINA

PLATAFORMAS PARA
DESENVOLVIMENTO DE PROTÓTIPOS

ESP8266
WI-FI

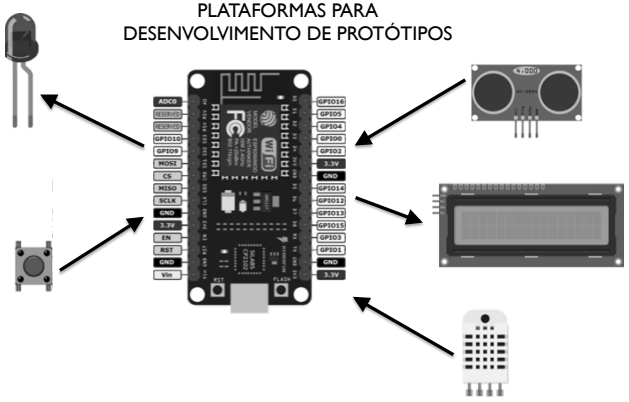
AMBIENTE ARDUINO

A OFICINA



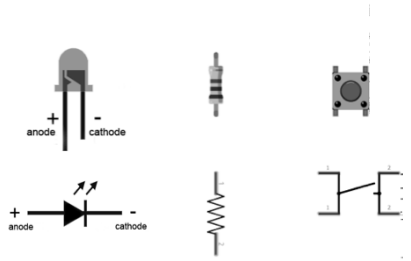
A OFICINA

PLATAFORMAS PARA
DESENVOLVIMENTO DE PROTÓTIPOS



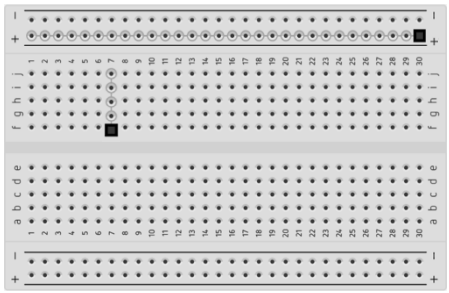
A OFICINA

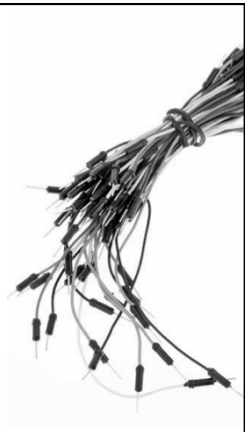
LED, RESISTOR e BOTÃO



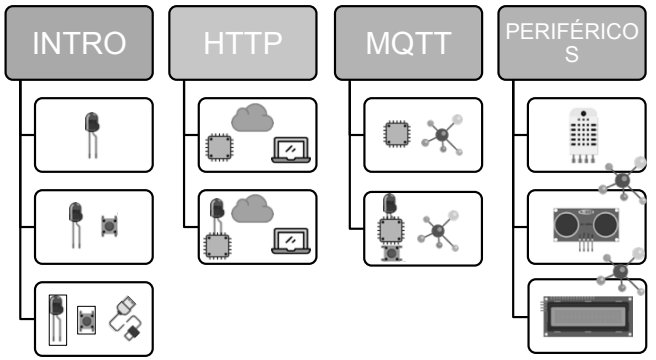
A OFICINA

PROTOBOARD

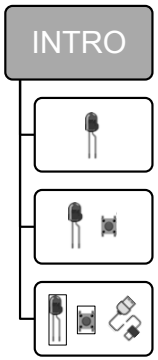




VAMOS COMEÇAR!

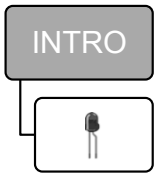


INTRODUÇÃO

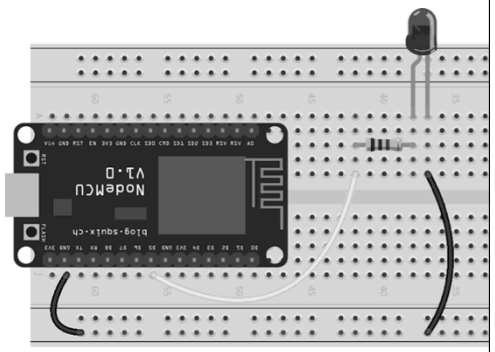


- desenvolvimento
- Componentes básicos
- Funções básicas

LED

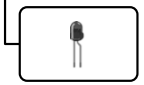


Acionamento de um LED pela placa



LED

INTRO

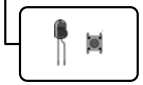


Acionamento de um LED pela placa

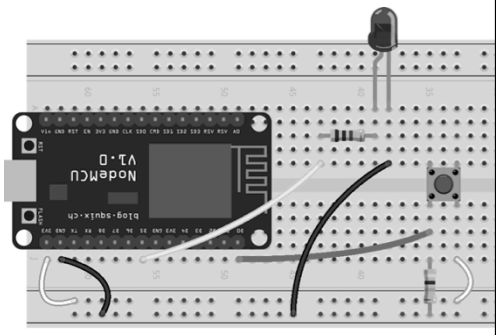
```
#define LED_PIN D5
#define DELAY_TIME 1000

void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(DELAY_TIME);

  digitalWrite(LED_PIN, LOW);
  delay(DELAY_TIME);
}
```

LED E BOTÃO

INTRO

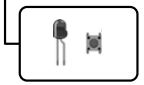


Botão para acionar o LED



LED E BOTÃO

INTRO

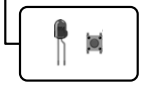


Botão para acionar o LED

```
#define LED_PIN D5
#define BUTTON_PIN D0

void setup()
{
  pinMode(LED_PIN, OUTPUT);

  /* TODO: Configure o pino do botão como uma entrada. */
}

void loop()
{
  bool buttonState = digitalRead(BUTTON_PIN);

  digitalWrite(LED_PIN, buttonState);

  delay(10);

}
```

LED E BOTÃO

INTRO



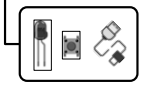
Botão para acionar o LED

```
#define LED_PIN D5
#define BUTTON_PIN D0

void setup()
{
  pinMode(LED_PIN, OUTPUT);

  pinMode(BUTTON_PIN, INPUT);
}

void loop()
{
  bool buttonState = digitalRead(BUTTON_PIN);

  digitalWrite(LED_PIN, buttonState);

  delay(10);
}
```

PORTA SERIAL

INTRO

Comunicação serial com o PC

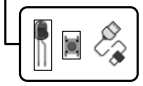


```
#define LED_PIN D5
#define BUTTON_PIN D0

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);

  Serial.begin(9600);

  Serial.println("DBServer | DBLab | Procergs");
  Serial.println("Oficina prática de Internet das Coisas");
}

void loop()
{
  bool buttonState = digitalRead(BUTTON_PIN);
  digitalWrite(LED_PIN, buttonState);

  /* TODO: Adicione mensagens pela porta serial
   *   para indicar o estado do botão e LED. */

  delay(100);
}
```

PORTA SERIAL

INTRO

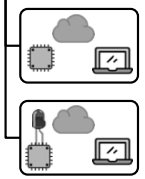
Comunicação serial com o PC



```
#define LED_PIN D5
#define BUTTON_PIN D0

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);

  Serial.begin(9600);

  Serial.println("DBServer | DBLab | Procergs");
  Serial.println("Oficina prática de Internet das Coisas");
}

void loop()
{
  bool buttonState = digitalRead(BUTTON_PIN);
  digitalWrite(LED_PIN, buttonState);

  if(buttonState == HIGH)
    Serial.println("O LED está ligado.");
  else
    Serial.println("O LED está desligado.");

  delay(100);
}
```

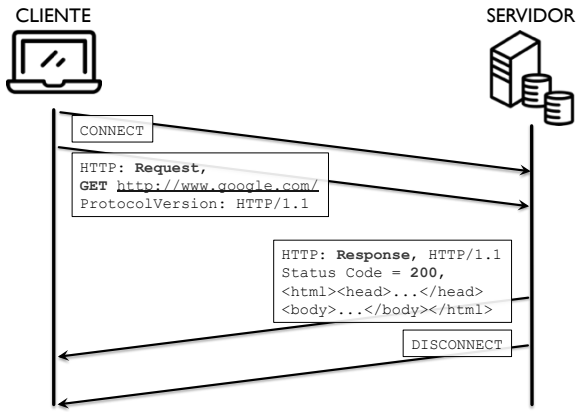
COMUNICAÇÃO HTTP

HTTP

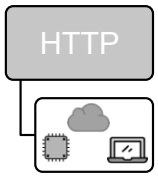


- Serviço HTTP
- Conectividade Wi-Fi
- Controle de um LED pela Internet

COMUNICAÇÃO HTTP



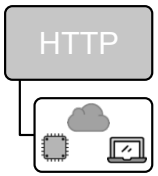
CLIENTE WI-FI



Serviço HTTP via Wi-Fi

```
#include <ESP8266WebServer.h>
#include "wifi.h"

ESP8266WebServer server(80);

WFclass wifi;

void setup(void)
{
  Serial.begin(9600);

  wifi.setup();

  server.on("/", handleRoot);

  server.begin();
}

void loop(void)
{
  server.handleClient();
}

void handleRoot()
{
  server.send(200, "text/plain", "Ola Procergs! ;)");
}
```

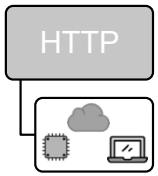
CLIENTE WI-FI



Serviço HTTP via Wi-Fi

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

class WFclass
{
    public:
      WFclass();
      void setup();

    private:
      /* TODO: Informe aqui o nome e senha da rede
       *   Wi-Fi disponivel. */
      const char* c_ssid = "";
      const char* c_pass = "";

      int status;
};
```

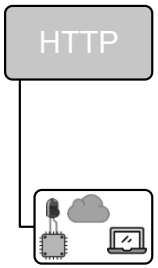
CLIENTE WI-FI



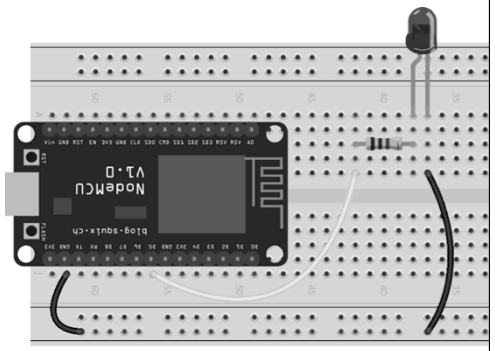
Serviço HTTP via Wi-Fi

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>

class WFclass
{
    public:
      WFclass();
      void setup();

    private:
      const char* c_ssid = "DBDevices";
      const char* c_pass = "!mP@db@dM";

      int status;
};
```

WEB SERVER E LED

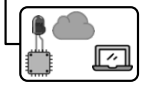


Controle de um LED pela web



WEB SERVER E LED

HTTP

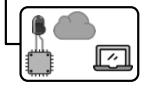


Controle de um LED pela web

```
#define LED_PIN D5

ESP8266WebServer server(80);
WFclass wifi;

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  Serial.begin(9600);

  wifi.setup();

  server.on("/", handleRoot); /* Rota raiz. */

  /* TODO: adicione callbacks para as rotas '/on' e '/off'. */

  server.begin();
}

void handleRoot()
{
  String webString =
  "<!DOCTYPE HTML><html><head>"
  "<meta name='apple-mobile-web-app-capable' content='yes'>"
  "<style>body{background-color:#4285F4;font-size:60px;font-famil
```

WEB SERVER E LED

HTTP

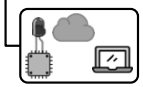


Controle de um LED pela web

```
#define LED_PIN D5

ESP8266WebServer server(80);
WFclass wifi;

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  Serial.begin(9600);

  wifi.setup();

  server.on("/", handleRoot);
  server.on("/on", handleLedOn);
  server.on("/off", handleLedOff);

  server.begin();
  Serial.println("Servidor HTTP rodando...");
}

void handleRoot()
{
  String webString =
  "<!DOCTYPE HTML><html><head>"
  "<meta name='apple-mobile-web-app-capable' content='yes'>"
```

WEB SERVER E LED

HTTP



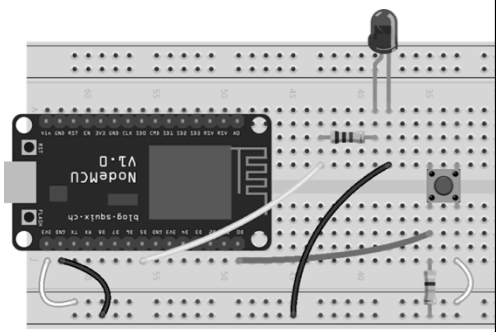
Controle de um LED pela web

```
void handleLedOn()
{
  digitalWrite(LED_PIN, HIGH);
  handleRoot();
}

void handleLedOff()
{
  digitalWrite(LED_PIN, LOW);
  handleRoot();
}
```

BOTÃO E LED PELA WEB

HTTP

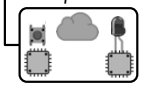
Botão controlando LED pela web



BOTÃO E LED PELA WEB

HTTP

Botão controlando LED pela web

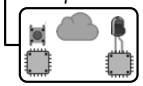


```
void setup()
{
  Serial.begin(9600);

  pinMode(BUTTON_PIN, INPUT);
  digitalWrite(BUTTON_PIN, LOW);

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  wifi.setup();

  webservice.setHttpHandler("/led", httpHandler);
  webservice.start();
}
```

BOTÃO E LED PELA WEB

HTTP

Botão controlando LED pela web

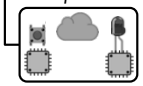


```
void loop()
{
  webservice.handleClient();

  int buttonState = digitalRead(BUTTON_PIN);
  if(buttonState != currentButtonState)
  {
      ...

      currentButtonState = buttonState;

      String state;
      if(buttonState == HIGH) state = "on";
      else state = "off";

      webserviceClient.sendRequest(state, host);
  }
}
```

BOTÃO E LED PELA WEB

HTTP

Botão controlando LED pela web

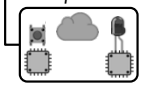


```
void httpHandler()
{
  String state;

  webservice.httpHandler(state);

  if(state == "on")
    digitalWrite(LED_PIN, HIGH);
  else if(state == "off")
    digitalWrite(LED_PIN, LOW);

  Serial.println("Led está agora " + state);
  Serial.println();
}
```

BOTÃO E LED PELA WEB

HTTP

Botão controlando LED pela web

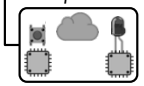


```
void ESP8266WebserviceClient::sendRequest(String &state,
                                          const char *host)
{
    wifiClient.connect(host, port);

    Serial.print("Enviando requisição para o host ");
    Serial.println(host);

    // Cria-se URI para requisição.
    String url = "/led?state=" + state;

    Serial.print("GET URL: ");
    Serial.println(url);

    wifiClient.print(String("GET ") + url + " HTTP/1.1\r\n" +
                     "Host: " + host + "\r\n" +
                     "Connection: close\r\n\r\n");

    wifiClient.print(String("GET ") + url + " HTTP/1.1\r\n" +
                     "Host: " + host + "\r\n" +
                     "Connection: close\r\n\r\n");
}
```

BOTÃO E LED PELA WEB

HTTP

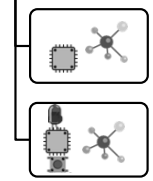
Botão controlando LED pela web



```
#define LED_PIN D5
#define BUTTON_PIN D0

/* TODO: Coloque aqui o endereço IP do host conten
 *   o LED que queremos acionar. */
const char *host = "";

bool currentButtonState = LOW;

...

void setup()
{
  Serial.begin(9600);

  pinMode(BUTTON PIN, INPUT);
```

COMUNICAÇÃO MQTT

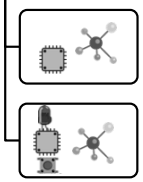
MQTT

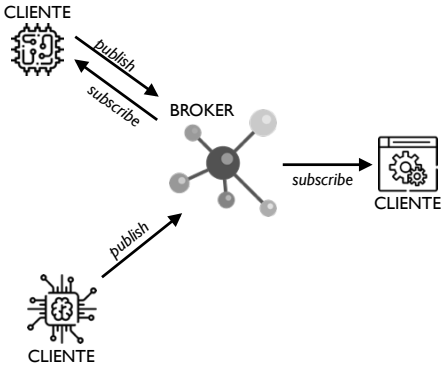


- Arquitetura *publish/subscribe*
- Publicação de mensagens em um *broker* público

COMUNICAÇÃO MQTT

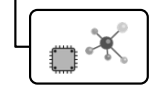
MQTT





MQTT

MQTT

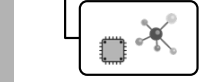


Cliente MQTT simples

```
WFclass wifi;
ESP8266PubSubClient mqttClient;
int value = 0;
long lastTimeMsg = 0;

void setup()
{
  Serial.begin(9600);
  wifi.setup();
  mqttClient.setup();
  mqttClient.connect();
}

void loop()
{
  if(!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();

  long now = millis();
  if (now - lastTimeMsg > 2000) {
    lastTimeMsg = now;
    ++value;
    String message("Hello World! #");
    message += String(value, DEC);
    mqttClient.publish(MQTT_OUT_TOPIC, message.c_str());
    Serial.println("Mensagem publicada: " + message);
  }
}
```

MQTT

MQTT

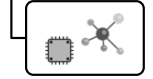


Cliente MQTT simples

```
ESP8266PubSubClient::ESP8266PubSubClient()
{
  pubSubClient = new PubSubClient(wifiClient);
  byte mac[6];
  WiFi.macAddress(mac);
  deviceID = String(mac[0],HEX)+String(mac[1],HEX)+String(mac[2],HEX)+
             String(mac[3],HEX)+String(mac[4],HEX)+String(mac[5],HEX);
}

void ESP8266PubSubClient::setup()
{
  pubSubClient->setServer(mqtt_server, mqtt_server_port);
}

void ESP8266PubSubClient::setCallback(MQTT_CALLBACK_SIGNATURE)
{
  pubSubClient->setCallback(callback);
}

boolean ESP8266PubSubClient::publish(const char* topic,
                                     const char* payload)
{
  return pubSubClient->publish(topic, payload);
}
```

MQTT

MQTT

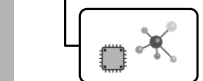


Cliente MQTT simples

```
void ESP8266PubSubClient::connect()
{
  // Repete até ter uma conexão efetuada.
  while (!pubSubClient->connected()) {
    Serial.print("Conectando-se ao broker MQTT...");
    // Tenta conexão.
    if (pubSubClient->connect(deviceID.c_str())) {
      Serial.println(" conectado!");

      /* TODO: quando o cliente conseguir se conectar, enviar
       *   a mensagem "ESTOU VIVO" para o tópico de saída. */

    } else {
      Serial.print(" falhou, rc= ");
      Serial.print(pubSubClient->state());
      Serial.println(" Tentando novamente em 5s...");
      delay(5000);
    }
  }
}
```

MQTT

MQTT

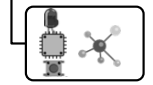


Cliente MQTT simples

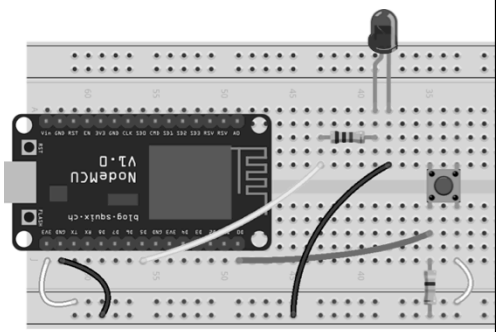
```
void ESP8266PubSubClient::connect()
{
  // Repete até ter uma conexão efetuada.
  while (!pubSubClient->connected()) {
    Serial.print("Conectando-se ao broker MQTT...");
    // Tenta conexão.
    if (pubSubClient->connect(deviceID.c_str())) {
      Serial.println(" conectado!");

      pubSubClient->publish(MQTT_OUT_TOPIC, "ESTOU VIVO!");

    } else {
      Serial.print(" falhou, rc= ");
      Serial.print(pubSubClient->state());
      Serial.println(" Tentando novamente em 5s...");
      delay(5000);
    }
  }
}
```

MQTT

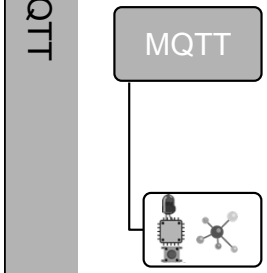
MQTT



Cliente MQTT pub/sub

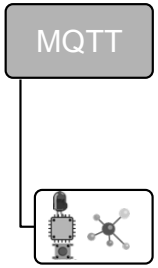


MQTT



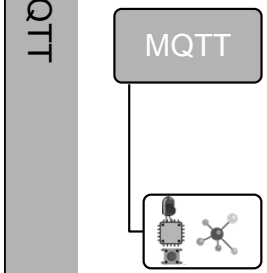

```
#define LED_PIN D5
#define BUTTON_PIN D0

WFclass wifi;
ESP8266PubSubClient mqttClient;

int currentButtonState = LOW;

void setup()
{
  Serial.begin(9600);

  pinMode(BUTTON_PIN, INPUT);
  digitalWrite(BUTTON_PIN, LOW);

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  wifi.setup();
  mqttClient.setup();
  mqttClient.connect();

  mqttClient.setCallback(callback);
}
```

MQTT



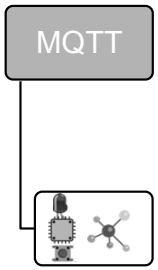

```
void callback(char *topic, byte *payload, unsigned int length)
{
  char message[length+1];
  memcpy(message, payload, length);
  message[length]=0;

  Serial.print("Mensagem recebida: ");
  Serial.println(message);

  if(!strncmp(message, "on", 2))
  {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("O LED está ligado!");
  }
  else if(!strncmp(message, "off", 3))
  {
    digitalWrite(LED_PIN, LOW);
    Serial.println("O LED está desligado!");
  }
}
```

MQTT




```
void loop()
{
  if(!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();

  int buttonState = digitalRead(BUTTON_PIN);
  if(buttonState != currentButtonState) {
    delay(50);
    buttonState = digitalRead(BUTTON_PIN);
    if(buttonState != currentButtonState)
    {
      currentButtonState = buttonState;

      String message("Meu botão está ");
      if(buttonState == LOW)
        message += String("desligado.");
      else
        message += String("ligado.");

      mqttClient.publish(MQTT_OUT_TOPIC, message.c_str());

      /* TODO: publique o estado do botão no tópico de entrada,
       *   com mensagens "on" e "off". */

      Serial.println("Mensagem publicada: " + message);
    }
  }
  delay(10);
}
```

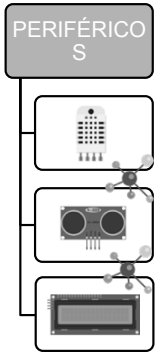
MQTT




```
void loop()
{
  if(!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();

  int buttonState = digitalRead(BUTTON_PIN);
  if(buttonState != currentButtonState) {
    delay(50);
    buttonState = digitalRead(BUTTON_PIN);
    if(buttonState != currentButtonState)
    {
      currentButtonState = buttonState;

      String message("Meu botão está ");
      if(buttonState == LOW)
        message += String("desligado.");
      else
        message += String("ligado.");

      mqttClient.publish(MQTT_OUT_TOPIC, message.c_str());

      if(buttonState == LOW) mqttClient.publish(MQTT_IN_TOPIC, "off");
      else mqttClient.publish(MQTT_IN_TOPIC, "on");

      Serial.println("Mensagem publicada: " + message);
    }
  }
  delay(10);
}
```

MQTT

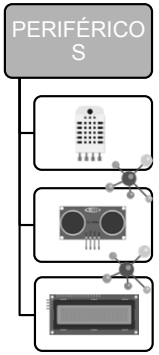
PERIFÉRICOS



- Integração com periféricos de entrada e saída
- Comandos e mensagens via MQTT

MQTT

PERIFÉRICOS



SENSORES:

```
#define MQTT_OUT_TOPIC "dblab/hands-on/mqtt/display"

mqttClient.publish(MQTT_OUT_TOPIC, msg);
```

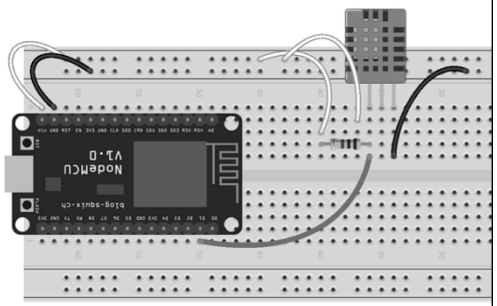
DISPLAY:

```
#define MQTT_IN_TOPIC "dblab/hands-on/mqtt/display"

pubSubClient->subscribe(MQTT_IN_TOPIC);
```

PERIFÉRICOS

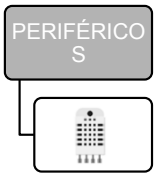
PERIFÉRICOS

Sensor de temperatura e umidade



PERIFÉRICOS

PERIFÉRICOS



Sensor de temperatura e umidade

```
#include <DHT.h>

#define DHTPIN D1
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  Serial.begin(9600);

  dht.begin();

  wifi.setup();
  mqttClient.setup();
}
```

PERIFÉRICOS

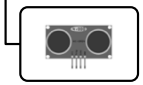
PERIFÉRICOS

Sensor de temperatura e umidade

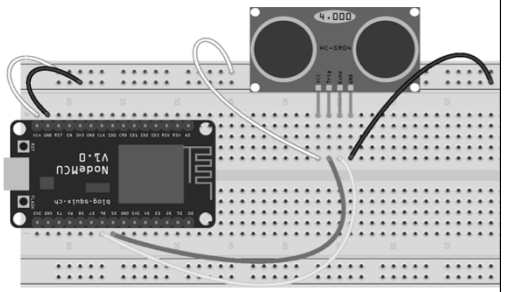
```
void loop()
{
  if (!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();

  float hum = dht.readHumidity();
  float temp = dht.readTemperature();
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.print(" %, Temp: ");
  Serial.print(temp);
  Serial.println(" Celsius");

  char humidity[MESSAGE_MAX_SIZE] = "";
  char temperature[MESSAGE_MAX_SIZE] = "";

  snprintf (temperature, MESSAGE_MAX_SIZE, "T: %02dC", (int)temp);
  snprintf (humidity, MESSAGE_MAX_SIZE, "H: %02d%%", (int)hum);

  mqttClient.publish(MQTT_OUT_TOPIC, humidity);
  mqttClient.publish(MQTT_OUT_TOPIC, temperature);
}
```

PERIFÉRICOS

PERIFÉRICOS

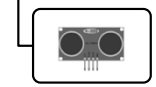


Sensor de distância



PERIFÉRICOS

PERIFÉRICOS



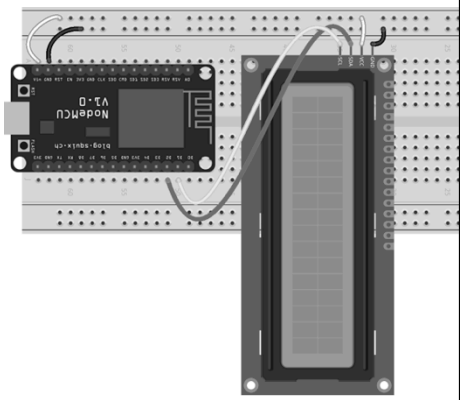
Sensor de distância

```
#include <hcsr04.h>

#define TRIG_PIN D5
#define ECHO_PIN D6

HCSR04 hcsr04(TRIG_PIN, ECHO_PIN, 20, 4000);

...


void loop()
{
  if (!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();

  long now = millis();
  if (now - lastTimeMsg > 500)
  {
    lastTimeMsg = now;

    int distance = hcsr04.distanceInMillimeters();
    char msg[MESSAGE_MAX_SIZE] = "";
    snprintf (msg, MESSAGE_MAX_SIZE, "Dist: %ldmm", distance);
    Serial.println(msg);
    mqttClient.publish(MQTT_OUT_TOPIC, msg);
  }
}
```

PERIFÉRICOS

PERIFÉRICOS

Display de cristal líquido



PERIFÉRICOS

PERIFÉRICOS

Display de cristal líquido

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
  Serial.begin(9600);

  wifi.setup();
  mqttClient.setup();
  mqttClient.connect();

  mqttClient.setCallback(callback);

  lcd.begin();
  lcd.backlight();
  lcd.print("DBLAB HANDS-ON");
}

void loop()
{
  if (!mqttClient.connected()) mqttClient.connect();
  mqttClient.loop();
}
```

PERIFÉRICOS

PERIFÉRICOS

Display de cristal líquido

```
void callback(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  char msg[length+1];
  memcpy(msg, payload,length);
  msg[length]=0;
  Serial.print(msg);

  if(!strncmp(msg, "I", 1)) {
    clearLcd(strlen(msg));
    lcd.setCursor(0, 0);
  }
  else if(!strncmp(msg, "H", 1)) {
    clearLcd(strlen(msg));
    lcd.setCursor(8, 0);
  }
  else if(!strncmp(msg, "D", 1)) {
    clearLcd(strlen(msg));
    lcd.setCursor(0, 1);
  }

  lcd.printstr(msg);
}
```

https://github.com/dbserver/dblab/tree/master/hands-on/iot

DBSERVER DBLAB

Obrigado!

Medium

dblab@dbserver.com.br

DBSERVER DBLAB