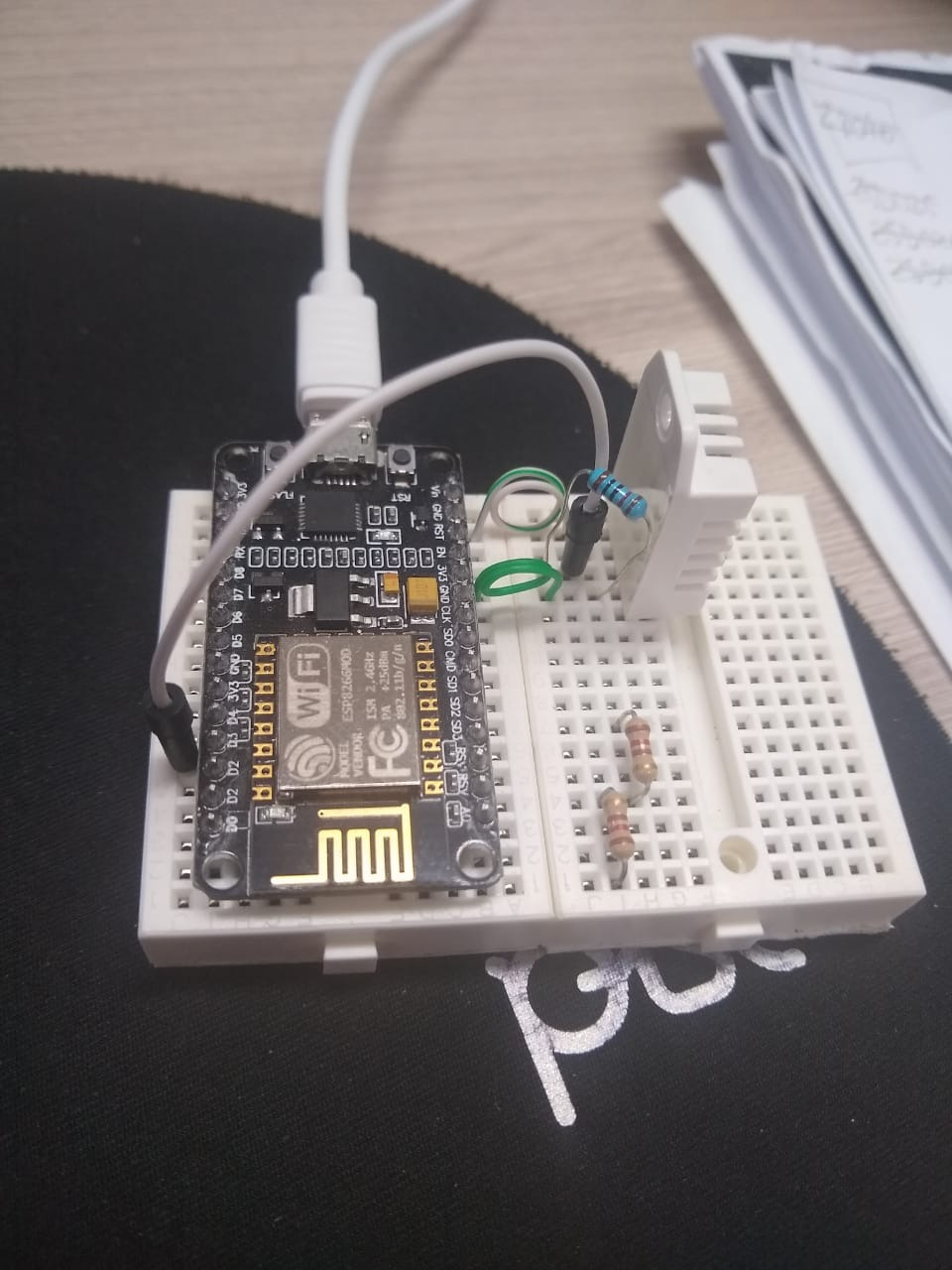
Trabalho Pratico de MQTT

Criar um dispositivo para medir temperatura e umidade.

**Dispositivo:**

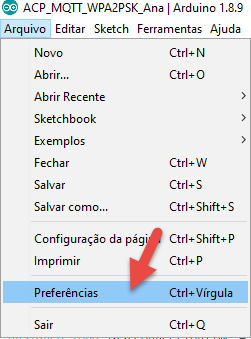


Microcontrolador: ESP8266

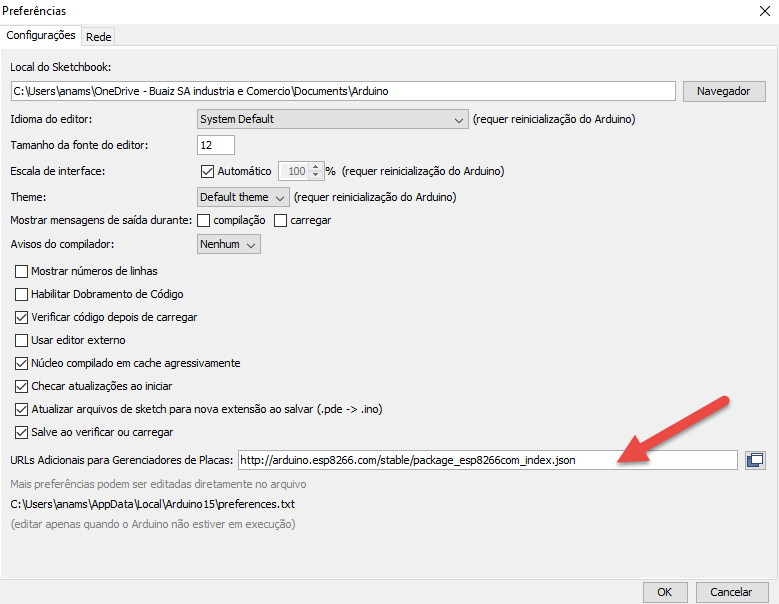
Sensor temperatura e umidade: DHT22

Conexão WiFi

**Instalar Placa microcontrolador no Arduino:**



**Inserir link para buscar pacote da placa do microcontrolador**



**Instalar Bibliotecas:**

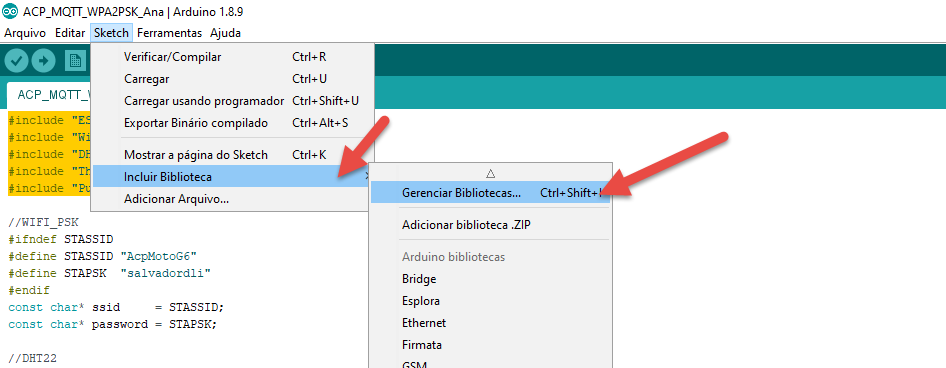
**#include "ESP8266WiFi.h"**

**#include "WiFiClient.h"**

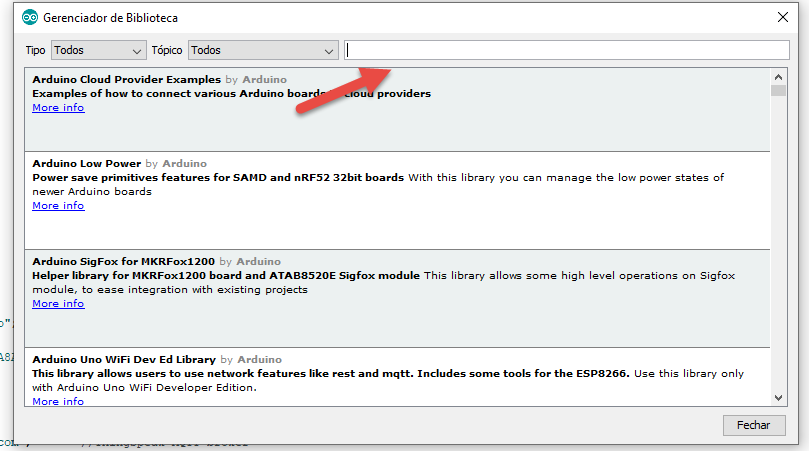
**#include "DHT.h"**

**#include "ThingSpeak.h"**

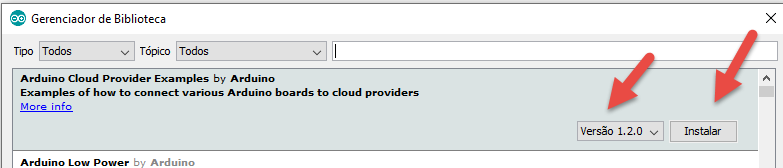
**#include "PubSubClient.h"**



**Inserir Biblioteca e pesquisar:**



**Escolher versão e instalar:**



Dados enviados para MQTT Broker

mqtt.thingspeak.com

O dispositivo foi programado usando Arduino, conforme abaixo:

#include "ESP8266WiFi.h"

#include "WiFiClient.h"

#include "DHT.h"

#include "ThingSpeak.h"

#include "PubSubClient.h"

//WIFI\_PSK

#ifndef STASSID

#define STASSID "XXXXXXXX"

#define STAPSK "XXXXXXX"

#endif

const char\* ssid = STASSID;

const char\* password = STAPSK;

//DHT22

#define DHTPIN 0 //GPIO-0 D3 do nodemcu

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

//ThingSpeak MQTT Channel

char\* mqttUserName = "TSArduinoMQTTDemo"; //Use any name.

char\* mqttPass = "PY1NGB3H4NU0WH3E"; //Change to your MQTT API Key from Account > MyProfile.

const char\* writeAPIKey = "OJE9RGQWNK9A8A6X"; //ThingSpeak Account Write API Key

long channelID = 898347; //ThingSpeak Channel ESP8266\_MQTT

WiFiClient client;

PubSubClient mqttClient(client); // Initialize the PubSubClient library.

const char\* server = "mqtt.thingspeak.com"; //ThingSpeak MQTT broker

static const char\* alphanum ="0123456789"

"ABCDEFGHIJKLMNOPQRSTUVWXYZ"

"abcdefghijklmnopqrstuvwxyz"; // For random generation of client ID.

unsigned long lastConnectionTime = 0;

const unsigned long postingInterval = 15L \* 1000L; // Grava a cada 15 seconds.

//=============================================================================

void setup()

{

//Serial setup

Serial.begin(115200);

delay(500);

pinMode(12, OUTPUT);

pinMode(13, OUTPUT);

//DHT Setup

Serial.println();

Serial.println("================================");

Serial.println(F("DHT22 conectado!"));

Serial.println("================================");

dht.begin();

delay(500);

//DHT End

//Wifi Setup

Serial.print("Conectando SSID '");

Serial.print(ssid);

Serial.println("'.");

//WPA\_PSK

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

//WPA\_PSK End

//Wifi Report

Serial.print("SSID '");

Serial.print(ssid);

Serial.println("' conectado!");

Serial.printf("Hostname: %s\n", WiFi.hostname().c\_str());

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

Serial.printf("Mask: %s\n", WiFi.subnetMask().toString().c\_str());

Serial.printf("Gateway: %s\n", WiFi.gatewayIP().toString().c\_str());

Serial.printf("Mac address: %s\n", WiFi.macAddress().c\_str());

Serial.println("================================");

//Wifi End

//MQTT Setup

mqttClient.setServer(server, 1883); //MQTT Broker details.

//MQTT End

}

//=============================================================================

void reconnect()

{

char clientID[9];

// Loop until reconnected.

while (!mqttClient.connected()) {

Serial.print("Contectando ao Broker MQTT...");

// Generate ClientID

for (int i = 0; i < 8; i++) {

clientID[i] = alphanum[random(51)];

}

clientID[8]='\0';

// Connect to the MQTT broker.

if (mqttClient.connect(clientID,mqttUserName,mqttPass)) {

Serial.println();

Serial.print("Broker MQTT conectado (Channel):");

Serial.println(channelID);

Serial.println("================================");

}

else {

Serial.print("failed, rc=");

// Print reason the connection failed.

// See https://pubsubclient.knolleary.net/api.html#state for the failure code explanation.

Serial.print(mqttClient.state());

Serial.println(" try again in 5 seconds");

delay(5000);

}

}

}

//=============================================================================

void mqttPublishFeed() {

float t = dht.readTemperature(); // Le temperatura.

float h = dht.readHumidity(); // Le umidade.

float hic = dht.computeHeatIndex(t, h, false);; // Le HIC.

// Create data string to send to ThingSpeak.

String data = String("field1=") + String(t) + "&field2=" + String(h) + "&field3=" + String(hic);

int length = data.length();

const char \*msgBuffer;

msgBuffer=data.c\_str();

Serial.println(msgBuffer);

// Create a topic string and publish data to ThingSpeak channel feed.

String topicString = "channels/" + String( channelID ) + "/publish/"+String(writeAPIKey);

length = topicString.length();

const char \*topicBuffer;

topicBuffer = topicString.c\_str();

mqttClient.publish( topicBuffer, msgBuffer );

lastConnectionTime = millis();

}

//=============================================================================

void loop() {

// Reconnect if MQTT client is not connected.

if (!mqttClient.connected()) {

reconnect();

}

// Call the loop continuously to establish connection to the server.

mqttClient.loop();

// If interval time has passed since the last connection, publish data to ThingSpeak.

if (millis() - lastConnectionTime > postingInterval) {

mqttPublishFeed(); // Publish three values simultaneously.

}

}

Site para visualização dos dados:

<https://thingspeak.com/channels/898347>

