/\*

Projeto Coffee IOT (V 1.0)

Validando Wifi e MQTT

Hardware envolvido:(NodeMCU ESP8266)

Referencia do Codigo: https://www.arduino.cc/

\*/

//Inclui as bibliotecas de ESP8266WiFi e PubSubClient ao projeto

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

//Cria constantes do tipo char,

//ssid: nome da rede wifi,

//password: senha da rede wifi,

//mqtt\_server: endereco de rede do broker mqtt

const char\* ssid = "Zhone\_AD62";

const char\* password = "znid307908706";

const char\* mqtt\_server = "iot.eclipse.org";

//Instanciamento da classe WiFiClient, objeto espClient

WiFiClient espClient;

//Instanciamento da classe PubSubClient, objeto client

//recebe como argumento o objeto espClient

PubSubClient client(espClient);

//Classe responsável por conectar o NodeMCU 8266 a rede wifi

void setup\_wifi(){

delay(100);

Serial.println();

Serial.print("Conectando-se a rede WiFi: ");

Serial.println(ssid);

//Classe WiFi utiliza o método begin que recebe como argumento o

//nome da rede wifi e a senha do wifi

WiFi.begin(ssid, password);

//Enquanto o status da classe WiFi for diferente de WL\_CONNECTED o

//NodeMCU 8266 irá tentar se conectar a rede wifi

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

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Conectado a Rede Wifi ");

Serial.print("( ");

Serial.print(ssid);

Serial.println(" )");

Serial.print("Endereco IP do seu Arduino: " );

Serial.println(WiFi.localIP());

}

//Classe responsável por conectar nossa aplicação ao broker MQTT

void reconnect(){

//Enquanto o objeto client atraves do metodo connected for diferente de 0

while(!client.connected())

{

Serial.println("Estabelecendo conexão com o servidor MQTT");

//Ao estabelecer conexão com o broker MQTT o objeto client chama a método //connect que recebe o nome do cliente MQTT

if(client.connect("ESP8266Client2608"))

{

Serial.print("Conectado ao Servidor MQTT: ");

Serial.println(mqtt\_server);

//O objeto client chama o método subscribe que recebe o tópico ao qual //estamos inscritos

client.subscribe("gdgfoz/coffeeiot");

//Caso aconteça alguma falha na conexão com o broker haverá uma nova //tentativa de conexão a cada 5 segundos

}else{

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

Serial.println(client.state());

Serial.println("Nova tentativa em 5 segundos");

delay(5000);

}

}

}

//Classe responsável por receber as mensagens enviadas pelo broker MQTT

//A classe callback recebe três argumentos,

//topic : tópico que está enviando a mensagem,

//message: mensagem enviada,

//length : tamanho da mensagem

void callback(String topic, byte\* message, unsigned int length){

Serial.print("[Mensagem Recebida] Topico: ");

Serial.print(topic);

Serial.print(" Mensagem: ");

String messageTemp;

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

Serial.print((char)message[i]);

messageTemp += (char)message[i];

}

}

//Classe obrigatória da plataforma Arduino

void setup() {

//Chama a classe setup\_wifi para iniciar a conexão wifi

setup\_wifi();

//Objeto client (MQTT) usa o método setServer para se conectar ao Broker

//Método setServer recebe como argumentos o endereço do Broker e a porta

client.setServer(mqtt\_server, 1883);

//Objeto client (MQTT) usa o método setCallback (MQTT) e recebe como //argumento a classe callback

client.setCallback(callback);

}

//Classe obrigatória da plataforma Arduino

//A classe loop inicia um loop infinito na aplicação

void loop() {

//Se o ESP8266 não estiver conectado irá chamar a classe reconnect() para a //conexão ao Wifi

if(!client.connected())

{

reconnect();

}

if(!client.loop())

client.connect("ESP8266Client2608");

client.loop();

}