PONTÍFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Ensino Superior

Engenharia de Computação

LÂMPADA DE LAVA

Cloud

Grupo: Ana Beatriz, Marcos Vitor, Mariana, Paula, Yago

Belo Horizonte, 2023

Objetivo: Demonstrar a conexão do código do hardware com o broker

Ferramentas utilizadas

- Arduino IDE – para o desenvolvimento do código
- Broker – onde o código irá se conectar
- Hardware – onde será executado o código

Programa

```
#include <arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define PIN_LED 13

/* Definicoes para o MQTT */
#define TOPICO_SUBSCRIBE_LED         "topico_liga_desliga_led"
#define TOPICO_PUBLISH_TEMPERATURA   "topico_sensor_temperatura"

#define ID_MQTT "IoT_PUC_SG_mqtt"

// Roteador celular
const char* SSID = "nome"; // SSID / nome da rede WI-FI que deseja se conectar
const char* PASSWORD = "senha"; // Senha da rede WI-FI que deseja se conectar

const char* BROKER_MQTT = "test.mosquitto.org";
//const char* BROKER_MQTT = "broker.emqx.io";
int BROKER_PORT = 1883; // Porta do Broker MQTT


//Variáveis e objetos globais
WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT passando o objeto
espClient

long numAleatorio;

/* Prototypes */
void initWiFi(void);
void initMQTT(void);
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void reconnectMQTT(void);
void reconnectWiFi(void);
void VerificaConexoesWiFIEMQTT(void);
```

```
/*****************
* IMPLEMENTACOES *
******************/

/**
  Inicializa e conecta-se na rede WI-FI desejada
  Parâmetros: nenhum
  Retorno: nenhum
*/
void initWiFi (void) {
  delay(10);
  Serial.println("------Conexao WI-FI------");
  Serial.print("Conectando-se na rede: ");
  Serial.println(SSID);
  Serial.println("Aguarde");

  reconnectWiFi();
}

/**
  Inicializa parâmetros de conexão MQTT(endereço do
  broker, porta e inicializa a função de callback)
  Parâmetros: nenhum
  Retorno: nenhum
*/
void initMQTT(void) {
  MQTT.setServer(BROKER_MQTT, BROKER_PORT);   //informa qual broker e porta
deve ser conectado
  MQTT.setCallback(mqtt_callback);            //atribui função de callback
(função chamada quando qualquer informação de um dos tópicos subescritos
chega)
}

/**
  Função de callback
  esta função é chamada toda vez que uma informação de
  um dos tópicos subescritos chega)
  Parâmetros: nenhum
  Retorno: nenhum
*/
void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  String msg;

  /* obtem a string do payload recebido */
  for (int i = 0; i < length; i++) {
    char c = (char)payload[i];
    msg += c;
  }
```

```
  Serial.print("Chegou a seguinte string via MQTT: ");
  Serial.println(msg);

  /* toma ação dependendo da string recebida */
  if(msg.equals("L")) {
    digitalWrite(PIN_LED, HIGH);
    Serial.println("LED aceso mediante comando MQTT");

  } else if (msg.equals("D")) {
    digitalWrite(PIN_LED, LOW);
    Serial.println("LED apagado mediante comando MQTT");

  } else {
    Serial.println("Não identificou comando MQTT recebido.");
  }
}

/**
  Reconecta-se ao broker MQTT (caso ainda não esteja conectado ou em caso de a
conexão cair)
  em caso de sucesso na conexão ou reconexão, o subscribe dos tópicos é
refeito.
  Parâmetros: nenhum
  Retorno: nenhum
*/
void reconnectMQTT(void) {
  while (!MQTT.connected()) {
    Serial.print("* Tentando se conectar ao Broker MQTT: ");
    Serial.println(BROKER_MQTT);

    if (MQTT.connect(ID_MQTT)) {
      Serial.println("Conectado com sucesso ao broker MQTT!");
      MQTT.subscribe(TOPICO_SUBSCRIBE_LED);

    } else {
      Serial.println("Falha ao reconectar no broker.");
      Serial.println("Havera nova tentativa de conexao em 2s");
      delay(2000);
    }
  }
}

/**
  Verifica o estado das conexões WiFI e ao broker MQTT.
  Em caso de desconexão (qualquer uma das duas), a conexão
  é refeita.
  Parâmetros: nenhum
```

```
  Retorno: nenhum
*/
void VerificaConexoesWiFIEMQTT(void) {
  if (!MQTT.connected())
    reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

  reconnectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

/**
  Reconecta-se ao WiFi
*/
void reconnectWiFi (void) {
  //se já está conectado à rede WI-FI, nada é feito.
  //Caso contrário, são efetuadas tentativas de conexão
  if (WiFi.status() == WL_CONNECTED) {
    return;
  }

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI

  while (WiFi.status() != WL_CONNECTED)  {
    delay(100);
    Serial.print(".");
  }

  Serial.println();
  Serial.print("Conectado com sucesso na rede ");
  Serial.print(SSID);
  Serial.println("\nIP obtido: ");
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(9600); //Enviar e receber dados em 9600 baud
  delay(1000);
  Serial.println("Disciplina IoT: acesso a nuvem via ESP32");
  delay(1000);
  // programa LED interno como saida
  pinMode(PIN_LED, OUTPUT);

  digitalWrite(PIN_LED, HIGH);
  delay(1000);
  digitalWrite(PIN_LED, LOW);    // apaga o LED

  // GERANDO TEMPERATURA COMO UM NÚMERO ALEATÓRIO
  // inicializa o gerador de números aleatórios.
  // um pino analógico desconectado irá retornar um
```

```
  // valor aleatório de tensão em analogRead()
  randomSeed(analogRead(0));

  /* Inicializa a conexao wi-fi */
  initWiFi();

  /* Inicializa a conexao ao broker MQTT */
  initMQTT();
}

void loop () {

  // cria string para temperatura
  char temperatura_str[10] = {0};

  /* garante funcionamento das conexões WiFi e ao broker MQTT */
  VerificaConexoesWiFIEMQTT();

  // gera um valor aleatório de temperatura entre 10 e 100
  numAleatorio = random(10, 101);

  // formata a temperatura aleatoria  como string
  sprintf(temperatura_str, "%dC", numAleatorio);

  /*  Publica a temperatura */
  MQTT.publish(TOPICO/PUBLISH/TEMPERATURA, temperatura_str);

  Serial.print("Gerando temperatura aleatoria: ");
  Serial.println(temperatura_str);

  /* keep-alive da comunicação com broker MQTT */
  MQTT.loop();

  /* Refaz o ciclo após 2 segundos */
  delay(2000);
}
```