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

```
/**
 * LAMPADA DE LAVA
 * Objetivo
 *   Criacao de um sistema de controle de uma lampada de lava e que se comunica com um Broker
 *
 * Funcionalidades
 * - Perceber a presença de uma pessoa para ligar ou desligar seu sistema
 * - Captar a luminosidade do ambiente para regular a intensidade da sua luz RGB
 * - Monitorar a temperatura, evitando a quebra do material e danos ao circuito
 * - Configuracao da cor da luz RGB
 * - Ligar ou desligar o sistema
 * - Comunicacao com broker
 *
 * Funcoes do sistema
 *   void desligar_sistema ();
 *   void led_modo_automatico ();
 *   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);
 *
 * Autores do projeto
 *   @author Ana Beatriz
 *   @author Marcos Victor
 *   @author Mariana Aram
```

```
 *   @author Paula Talim
 *   @author Yago Garzon
 */

#include <Adafruit_NeoPixel.h>
#include <PubSubClient.h>
#include <arduino.h>
#include <WiFi.h>
#include <DHT.h>

/*** Definicoes para o MQTT ***/
#define TOPICO_SUBSCRIBE_SISTEMA "lampadalava/atuador/ligadesliga"
#define TOPICO_SUBSCRIBE_LED_AUTOMATICO "lampadalava/atuador/auto"
#define TOPICO_SUBSCRIBE_LED_MANUAL "lampadalava/atuador/ledcor"
#define TOPICO_PUBLISH_TEMPERATURA "lampadalava/sensor/temperatura"
#define TOPICO_PUBLISH_LUMINOSIDADE "lampadalava/sensor/luminosidade"
#define TOPICO_PUBLISH_MOVIMENTO "lampadalava/sensor/movimento"
#define ID_MQTT "IoT_PUC_SG_mqtt" //id mqtt (para identificação de sessão)
#define BROKER_MQTT "test.mosquitto.org"
#define BROKER_PORT 1883 // Porta do Broker MQTT

// Configuracao wifi
#define SSID     "AndroidAPT"
#define PASSWORD "nfwb6809"

/*** SENSORES E CONTROLES ***/
#define LED_COUNT 16 //Numero de pixels da led
#define SENSORTEMP 14 // Sensor de temperatura
#define SENSORLUZ 32 // Sensor de luminosidade
#define SENSORMOV 39 // Sensor de movimento
#define RELE 27 // rele
#define pin 12 // LED

/*** VARIAVEIS GLOBAIS ***/
int luminosidade;
float temperatura;
```

```
int movimento;
bool modo_automatico_ativado = true;
bool sistema_ligado = true;

unsigned long time_now;
unsigned long time_start_mov = 0;
unsigned long time_start_color = 0;
unsigned long time_start_sensor = 0;
unsigned long intervalo_pir = 300000;
unsigned long intervalo_led_cor = 50;
unsigned long intervalo_leitura_sensor = 1000;
int red = 0;
int green = 0;
int blue = 0;


// Objetos do MQTT
WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT passando o objeto
espClient


// Objeto dos sensores e atuadores
DHT dht(SENSORTEMP, DHT11);
Adafruit_NeoPixel strip(LED_COUNT, pin, NEO_GRB + NEO_KHZ800);// Cria um objeto de
pixel da fita de LED


/*** 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);

/**
 * Desliga sistema
 *
 * Configura o sistema para o modo desligado
```

```
 */
void desligar_sistema () {
  sistema_ligado = false;
  digitalWrite (RELE, HIGH); //liga o RELE
  strip.clear();// Defina todas as cores dos pixels como preto (desligado)
  strip.show(); // Atualize a fita de LED com as novas cores
}

/**
 * Led Modo Automatico
 *
 * Configura as cores da led quando estiver no modo automatico
 */
void led_modo_automatico () {
  // Controla o tempo de cada cor
  if (time_now - time_start_color >= intervalo_led_cor) {
    time_start_color = time_now;

    // Ajuste da luminosidade
    int limite;
    if (luminosidade > 3500) {
      intervalo_led_cor = 100;
      limite = 50;
    } else {
      intervalo_led_cor = 50;

      if (luminosidade <= 2000) {
        limite = 255;
      } else if (luminosidade  <= 2500) {
        limite = 200;
      } else if (luminosidade <= 3000) {
        limite = 150;
      } else {
        limite = 100;
      }
    }
```

```
    // Ajuste na cor
    if (blue < 0 || red < 0 || green < 0) {
      red = 0;
      green = 0;
      blue = 0;
    } else if (green < limite && blue == 0) {
      green ++;
    } else if (red > 0 && green == limite) {
      red --;
    } else if (red == 0 && blue < limite) {
      blue ++;
    } else if (red == 0 && green > 0) {
      green --;
    } else if (red < limite && blue == limite) {
      red ++;
    } else {
      blue --;
    }
    //Atualiza a cor
    strip.fill(strip.Color(red, green, blue));
    strip.show();
  }
}

/**
 Inicializa e conecta-se na rede WI-FI desejada
 */
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)
 */
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 CALLBACK
  Esta função é chamada toda vez que uma informação de
  um dos tópicos subescritos chega)
 */
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")) {
    time_start_mov = time_now;
    Serial.println("LED aceso mediante comando MQTT");

  } else if (msg.equals("D")) {
    time_start_mov = time_now + intervalo_pir + 1;
    desligar_sistema();
```

```
  Serial.println("Sistema desligado pelo MQTT");

} else if (msg.equals("A")) {
  modo_automatico_ativado = true;

} else {
  modo_automatico_ativado = false;

  if (msg.equals("1")) {
    // Amarelo
    red = 244;
    green =  255;
    blue = 0;
  } else if (msg.equals("2")) {
    // Laranja
    red = 255;
    green =  100;
    blue = 0;
  } else if (msg.equals("3")) {
    // Vermelho
    red = 255;
    green =  0;
    blue = 0;
  } else if (msg.equals("4")) {
    // verde
    red = 0;
    green = 255;
    blue = 0;
  } else if (msg.equals("5")) {
    // Azul
    red = 0;
    green = 0;
    blue = 255;
  } else if (msg.equals("6")) {
    // Roxo
    red = 119;
```

```
      green = 0;
      blue = 200;
    } else if (msg.equals("7")) {
      // Rosa
      red = 237;
      green =  48;
      blue = 207;
    }
    // Verifica se o sistema esta ligado para atualizar a cor
    if (sistema_ligado) {
      strip.fill(strip.Color(red, green, blue));
      strip.show();
    }
  }
}


/**
  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.
 */
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_SISTEMA);
      MQTT.subscribe(TOPICO_SUBSCRIBE_LED_AUTOMATICO);
      MQTT.subscribe(TOPICO_SUBSCRIBE_LED_MANUAL);
    } else {
      Serial.println("Falha ao reconectar no broker.");
      Serial.println("Havera nova tentativa de conexao em 2s");
      delay(2000);
    }
```

```
  }
}

/**
  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());
}

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

void setup() {
  Serial.begin(9600); //Monitor serial
  pinMode(SENSORLUZ, INPUT); //DEFINE O PINO COMO ENTRADA
  pinMode(RELE, OUTPUT); //DEFINE O PINO COMO SAIDA
  pinMode(SENSORMOV, INPUT); //DEFINE O PINO COMO ENTRADA
  dht.begin();

  //COMANDOS LED
  strip.begin(); // Inicialize a fita de LED
  strip.clear(); // Defina todas as cores dos pixels como preto (desligado)
  strip.show(); // Atualize a fita de LED com as cores definidas

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

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

void loop() {
  char temperatura_str[10] = {0};
  char luminosidade_str[10] = {0};

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

  time_now = millis();

  //Controla o tempo de leitura dos sensores
  if (time_now - time_start_sensor >= intervalo_leitura_sensor) {
    time_start_sensor = time_now;
    if (sistema_ligado) {
      // LUMINOSIDADE
      // Leitura do fotoresistor
```

```
  luminosidade = analogRead(SENSORLUZ);
  // Exibe a luminosidade lida nomonitor serial
  Serial.println();
  Serial.print("Valor do sensor de luminosidade = ");
  Serial.println(luminosidade);
  // Publica a luminosidade lida no broker
  sprintf(luminosidade_str, "%d", luminosidade);
  MQTT.publish(TOPICO_PUBLISH_LUMINOSIDADE, luminosidade_str, 1);
  // INFORMA O VALOR DO SENSOR DE TEMPERATURA
  temperatura = dht.readTemperature();
  // Testa se retorno é valido, caso contrário algo está errado.
  if (!isnan(temperatura) && temperatura > 0) {
    // Exibe a temperatura no monitor serial
    Serial.print("\nValor do sensor de Temperatura = ");
    Serial.print(temperatura);
    Serial.println();
    // Publica a temperatura no broker
    sprintf (temperatura_str, "%.01f", temperatura);
    MQTT.publish(TOPICO_PUBLISH_TEMPERATURA, temperatura_str, 1);
  }
} else {
  // Publica 0 caso o sistema estiver desligado
  MQTT.publish(TOPICO_PUBLISH_LUMINOSIDADE, "0", 1);
  MQTT.publish(TOPICO_PUBLISH_TEMPERATURA, "0", 1);
}
//INFORMA O VALOR DO SENSOR DE MOVIMENTO
movimento = digitalRead(SENSORMOV);
Serial.print("Movimento: ");
if (movimento == HIGH) {
  time_start_mov = time_now;
  MQTT.publish(TOPICO_PUBLISH_MOVIMENTO, "1", 1);
  Serial.println("1");
} else {
  MQTT.publish(TOPICO_PUBLISH_MOVIMENTO, "0", 1);
  Serial.println("0");
```

```
    }
  }
  // Verifica se o sistema esta ligado
  if (time_now - time_start_mov <= intervalo_pir) {
    sistema_ligado = true;
    // Controla o aquecedor
    if (temperatura <= 35) {
      // Liga o aquecedor
      digitalWrite (RELE, LOW);
    } else if (temperatura >= 60) {
      // Desliga o aquecedor
      digitalWrite (RELE, HIGH);
    }
    // Controla a led
    if (modo_automatico_ativado) {
      led_modo_automatico();
    } else {
      strip.fill(strip.Color(red, green, blue));
      strip.show();
    }
  } else {
    desligar_sistema();
  }
  MQTT.loop();
}
```