Documentação do Código

Objetos Inteligentes Conectados

Alunos: Victor Enrique Marinho Caetano e Leonardo Mosca Almeida

Professor: Willian Costa

Importações

O código inicia com a importação das bibliotecas ESP8266WiFi e PubSubClient, sendo estas responsáveis por estabelecer a comunicação de todo o software com a internet, permitindo a transmissão de informações através da rede, utilizando o NodeMcu ESP8266, que utiliza o módulo ESP-12 E, e utilização dos métodos de publish e subscribe do protocolo MQTT, respectivamente

```
//importação das bibliotecas para comunicação via MQTT
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

Definição dos tópicos MQTT

Este trecho define os tópicos que serão utilizados pelo protocolo MQTT e manipulados pelos métodos de publish e subscribe da biblioteca PubSubClient para recebimento e envio de dados através da rede, seus valores serão atualizados através da captura de dados das e armazenamento em variáveis que serão descritas mais a frente neste documento

```
#define TOPICO_PUBLISH_GAS "leitura_gas"
#define ID_MQTT "sensorgas_mqtt"
#define TOPICO_ALERT_GAS "alert_gas"
```

Declaração das variáveis contendo credenciais para autenticação

Neste ponto são declaradas variáveis para armazenar as credenciais que terão a função de autenticação em serviços terceiros, assim como o estabelecimento de um dos serviços utilizados, no caso o serviço de Broker MQTT utilizado, o mosquitto

```
const char* ssid = "Caetano 2G";
const char* password = "***************";
//declaração da variável que irá armazenar o MQTT Broker Service utilizado
const char* mqtt_server = "test.mosquitto.org";
```

Definição da variável para comunicação WiFi

Aqui foi definida a variável que irá estabelecer a comunicação entre os dispositivos via WiFi assim como esta será o principal agente responsável por utilizar os métodos e publish e subscribe do protocolo MQTT

```
WiFiClient espClient;
PubSubClient client(espClient);
```

Funções

Funções de reconexão

Estas são as funções responsáveis por tentar estabelecer uma comunicação com a rede WiFi e o serviço broker MQTT, respectivamente, quando há uma falha nessa tentativa ela é executada novamente após um determinado delay

- Função de reconexão com o MQTT

  Esta função estabelece a tentativa de conexão com o broker MQTT, onde em caso de falha, há uma nova tentativa de reconexão após um delay de dois segundos que é estabelecido na função, a situação da conexão é verificada através de um método chamado connected(), contido na biblioteca PubSubClient

```
void reconectMQTT(void) {
  if (!client.connected()){
    while (!client.connected()) {
      Serial.print("* Tentando se conectar ao Broker MQTT: ");
      Serial.println(mqtt_server);
      if (client.connect(ID_MQTT)) {
        Serial.println("Conectado ao MQTT");
      } else {
        Serial1.println("Falha ao tentar se reconectar com o broker");
        Serial.println("Havera nova tentativa de conexao em 2s");
        //nova tentativa de reconexão em 2 segundos
        delay(2000);
      }
    }
  }
}
```

- Função de reconexão com o WiFi

Esta função utiliza as variáveis declaradas anteriormente para se autenticar na rede especificada no código, onde realiza esse processo constantemente em função da variável que monitora a situação atual da conexão, e que estabelece um delay para que uma nova tentativa de conexão seja realizada, em caso de sucesso de conexão o ssid e o IP da rede são exibidos no monitor serial

```
void reconnectWiFi(void) {

  if (WiFi.status() == WL_CONNECTED)
    return;

  WiFi.begin(ssid, password);

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

  Serial.println();
  Serial.print("Conectado com sucesso na rede ");
  Serial.print(ssid);
  Serial.println("IP obtido: ");
  Serial.println(WiFi.localIP());
}
```

- Função de verificação das conexões

Essa função tem como principal objetivo chamar as duas funções que se conectam ao WiFi e ao Broker MQTT ao mesmo tempo, ela também pode ser utilizada para estabelecer uma primeira conexão de forma mais eficiente

```
void verifyConnection(void) {
  reconnectWiFi();
  reconectMQTT();
}
```

- Função de call-back

A função responsável pela obtenção e processamento de um tópico e seu payload, onde é estabelecida a mensagem de comunicação do envio do tópico, assim como a exibição do conteúdo contido em seu payload, o tópico se trata da variável declarada no início do código responsável por ser o que irá armazenar as informações durante o transporte das mesmas através do protocolo MQTT, assim como o conteúdo destas informações propriamente ditas

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Topic arrived[");
  Serial.print(topic);
  Serial.print("]");
  for (int i=0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
```

Função setup

Uma das funções essenciais para a implementação do código no microcontrolador, nela são estabelecidas a taxa de transferência em bits da transmissão serial, assim como o estabelecimento do servidor MQTT que será utilizado, o qual foi atribuído a uma variável declarada no início do código, a porta utilizada para a envio e recebimento dos dados e a chamada da função de call-back

Logo após isso são estabelecidas as entradas, analógicas e digitais, onde estão conectados, os sensores e atuadores do projeto, assim como suas respectivas funções INPUT ou OUTPUT, através do método pinMode()

```
void setup() {
  //definição do serial, server MQTT utilizado e a porta de comunicação
  Serial.begin(115200);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  //definição dos pinos do circuito que irão receber ou enviar informações
  pinMode(A0, INPUT);
  pinMode(D2, OUTPUT);
  pinMode(D8, OUTPUT);
}
```

Função de loop

A função principal do código, tem como principal utilidade o processamento de todas as informações estabelecidas e a utilização dos métodos de publish e subscribe e envio dessas informações com a utilização dos tópicos, assim como acionamento do hardware disponível que possui a função de atuador no sistema e no circuito, que neste caso, se trata do Buzzer

Primeiro é realizada a declaração das variáveis char que poderão compor conteúdos de payloads, logo após ocorre a chamada da função que irá tentar

estabelecer a conexão com o WiFi e o MQTT Broker Service, a verifyConnection(), a qual foi declarada previamente no código

Então é feita a leitura dos dados obtidos pelo sensor através da porta analógica A0, onde estas são exibidas no monitor serial, e também através do método sprintf(), armazenadas concatenadas ao array de char responsável pelo armazenamento da quantidade, em partículas por milhão, de gás presente no ambiente, a qual será enviada por meio de um dos tópicos, em diferentes pontos no escopo da função

Por último, a estrutura condicional, que será responsável por verificar a quantidade retornada pela leitura dessas informações a partir do sensor, e se caso estas ultrapassarem um determinado valor, uma série de ações serão acionadas, dentre elas, o acionamento do LED, o acionamento do Buzzer e o envio de notificação para o usuário em um tópico através do método publish e caso contrário, todos os elementos serão desativados, incluindo o LED e o Buzzer e uma informação sobre a normalização da quantidade de gás no ambiente será enviada através do mesmo tópico, com o método publish, porém, utilizando um payload com conteúdo diferente do anteriormente citado na estrutura, o envio das informações com a quantidade de partículas por milhão que é retornada para o usuário através do protocolo MQTT é realizado com um delay de um segundo estabelecido no código

```
void loop(){
  //declaração das variáveis de resposta padrão que serão utilizadas como payloads em chamadas de funções publis
  char ppm_str[10] = {0};
  char alert_str[20] = {"GÁS VAZANDO"};
  char empty[30] = {"NÍVEL DE GÁS NORMAL"};
  //chamada da função verifyConnection() para se conectar ao WiFi e ao MQTT Broker
  verifyConnection();
  //obtenção e leitura dos dados do sensor de gás MQ-2
  int sensor = analogRead(A0);
  sprintf(ppm_str,"%i", sensor);
  Serial.println(sensor);
  //publish da leitura do gás através do protocolo MQTT
  client.publish(TOPICO_PUBLISH_GAS, ppm_str);

  if(sensor >= 150){
    //alteração do estado do LED alertando o usuário sobre o vazamento de gás
    digitalWrite(D2, HIGH);
    //publish no tópico alertando o usuário sobre o vazamento de gás
    client.publish(TOPICO_ALERT_GAS, alert_str);
    Serial.println(alert_str);
    //acionamento do estado do LED alertando o usuário do vazamento de gás
    tone(D8, 1000, 500);
  }
  else{
    //alteração do estado do LED alertando o usuário sobre a normalização do nível de gás no ambiente
    digitalWrite(D2, LOW);
    //publish no tópico de alerta informando o usuário sobre a normalização do nível de gás no ambiente
    client.publish(TOPICO_ALERT_GAS, empty);
    Serial.println(empty);
  }
  //delay de 1 segundo para retomada do processo de leitura e publish dos dados nos tópicos
  delay(1000);
}
```