UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Smart Home: economia e controle residencial através da internet

Pamela Taga, Wilian Franca Costa

Universidade Presbiteriana Mackenzie (UPM)
Rua da Consolação, 930 Consolação, São Paulo - SP, 01302-907 – Brazil

10919008181@mackenzista.com.br

Abstract. *This article describes the project that aims to demonstrate how the Internet of Things can be used in our day-to-day. Through the mqtt communication protocol, the user will be able to remotely control the lighting and have access to the temperature of their home or office.*

Resumo. *Este artigo descreve o projeto que tem como objetivo demonstrar como a Internet das Coisas pode ser utilizada no nosso dia a dia. Através do protocolo de comunicação MQTT, o usuário poderá controlar de forma remota a iluminação e ter acesso a temperatura de sua residência ou escritório.*

1. Introdução

Tecnologias para casas inteligentes não é algo novo, a primeira surgiu em 1975, uma plataforma de automação residencial – X10 – apta a enviar informações digitais por meio de rajadas de radiofrequência. Já em 1990, temos uma torradeira controlada pela internet. Com os avanços tecnológicos, a automação residencial deixou de ser algo para pessoas com alto poder aquisitivo e se tornou algo mais acessível. Um exemplo disso é o Google Home Smart Speaker que oferece recursos de controle remoto, proporcionando praticidade, segurança e sustentabilidade. Atualmente há diversas empresas que oferecem tecnologias para casa inteligente no Brasil, à Intelbras é uma delas.

Nesse artigo será mostrado um projeto onde o usuário poderá acender ou apagar uma lâmpada através do seu celular e verificar a temperatura daquele ambiente. É um projeto simples porém de grande importância, pois, através dele, podemos ter uma breve noção de como é a comunicação de objetos com a web através do protocolo MQTT.

2. Materiais e métodos

Para o desenvolvimento deste projeto, será utilizado os seguintes materiais:
01 – Placa ESP8266
01 – Cabo USB
03 – Jumpers Macho-Fêmea
01 – Sensor DHT22

Abaixo está o link da imagem dos componentes obtidas pelo datasheet

ESP 8266

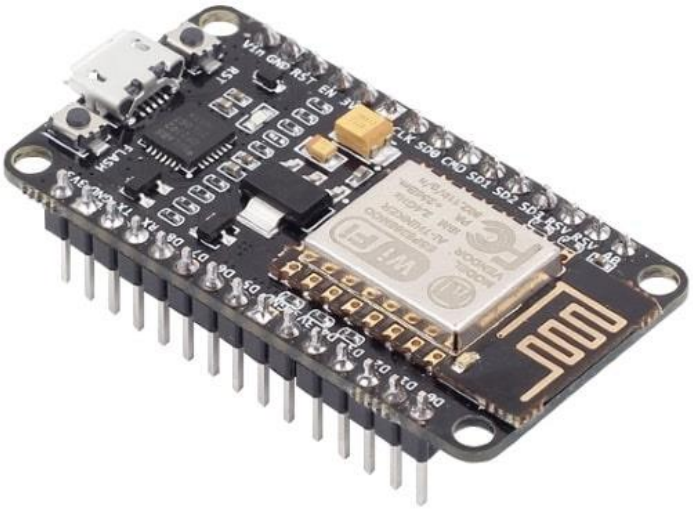


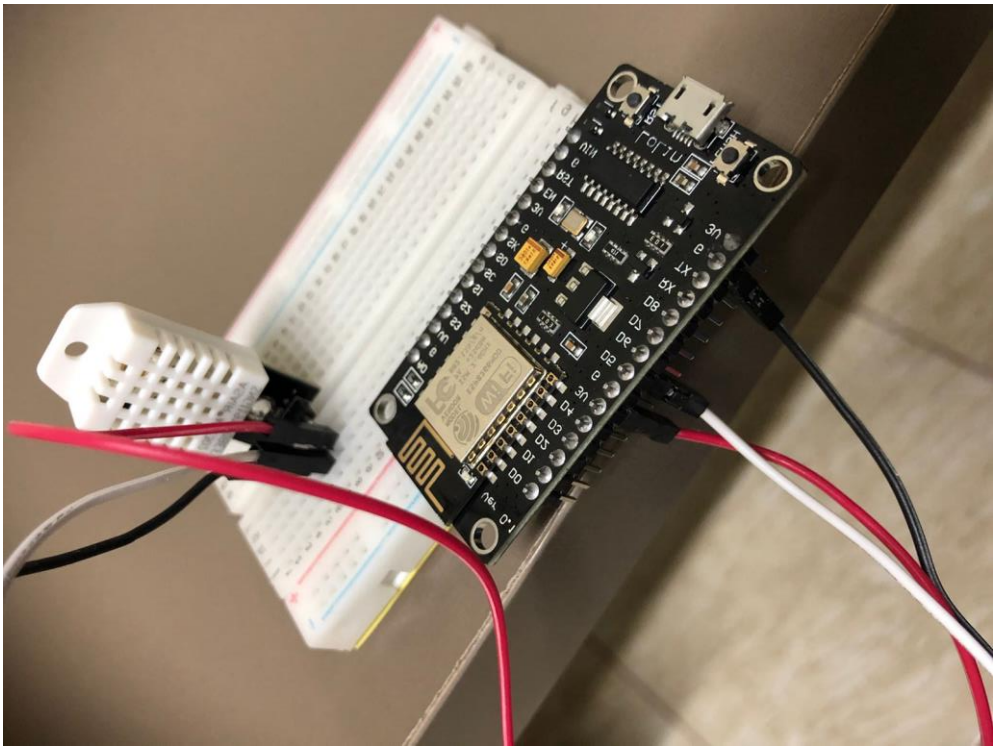
Figura 1: ESP8266 - Fonte: https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/?utm_source=google&utm_medium=organic&utm_campaign=shopping&utm_content=surfaces_across_google&gclid=CjwKCAiAm7OMBhAQEiwArvGi3DEJqqU6tXGORHwKMSDTHmpfjq0fZ6o7JsmI3Fbl6Kj3jue8gTXaGBoC2cEQAvD_BwE

Sensor DHT22

Figura 2: Sensor DHT 22 - https://www.alldatasheet.com/datasheet-pdf/pdf/1132459/ETC2/DHT22.html

Protótipo

Na imagem abaixo, podemos ver a montagem do protótipo pronto e como será utilizado cada componente citado acima.



A plataforma de prototipagem eletrônica utilizada será o ESP 8266 e o ambiente para desenvolvimento da programação será Arduino IDE

Especificações da placa ESP8266:

- Encapsulamento de 32 pinos QFN com 11.5mm por 11.5mm;
- Switch de Radiofrequência;
- Processadores de MAC e Baseband integrados;
- Gestão de Qualidade de Serviço (QoS) integrado;
- Interface I2S para aplicações de áudio;
- Regulador on-chip com baixo dropout para fornecimento de energia aos periféricos integrados;
- Arquitetura proprietária para geração de clock free-spurious;
- Engine integrada para criptografias WEP, TKIP, AES e WPA;
- Suporte aos protocolos 802.11 b/g/n;
- Wi-Fi Direct (P2P), soft-AP;
- Pilha de protocolo TCP/IP integrada com suporte a IPv4;
- Wi-Fi em frequência de 2.4GHz com suporte a WPA e WPA2;
- Potência de saída em +20dBm no modo 802.11b;
- Conversor ADC integrado com resolução de 10 bits;
- Suporte a uma variedade de antenas;
- Energia de consumo em modo sleep menor que 10uA;
- Tempo para sair de sleep e transmitir pacotes menor que 2ms;
- Potência de standby menor que 1.0mW;

Com o protótipo feito, é necessário fazer a programação mas, antes disso, para que o ambiente do arduino consiga se comunicar com a placa é necessário fazer as seguintes configurações:

1º Baixar o driver da placa: Dentro da IDE Arduino, acessar: arquivo – preferencias – copiar e colar este link: http://arduino.esp8266.com/stable/package_esp8266com_index.json em URLS adicionais para gerenciamento de placas – ok;
2º ir em: Ferramentas – placa – gerenciador de placas – digitar “ESP8266” – instalar- ok;

3º Selecionar a versão da placa: Acessar: Ferramentas – placa – ESP8266 Boards (3.0.2) NodeMCU 1.0 (ESP 12E Module);
4º Selecionar a porta: Ferramentas – Portas – COM4;
5º Baixar as seguintes bibliotecas: ESP8266WiFi.h, PubSubClient.h e DHT.h. Para isso, basta digitar o nome decada uma em: Sketch – incluir biblioteca – gerenciar bibliotecas – digitar o nome da biblioteca desejada – clicar em instalar e depois em ok

Agora é só copiar e colar o código abaixo e clicar em carregar

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

//DHT Area
#define DHTPIN D5 // set the pin that DHT is connected to
#define DHTTYPE DHT22 // set the DHT version
DHT dht(DHTPIN, DHTTYPE); // declares the DHT object
int t,h;

//Wifi area
const char* ssid = " "; // Set here SSID of your wifi network
const char* password = "  "; // Password of your wifi network
const char* mqtt_broker = "mqtt.eclipse.org"; // Change for your MQTT Broker prefered
WiFiClient nodemcuClient;
PubSubClient client(nodemcuClient);

void setup() {
  Serial.begin(9600);
  dht.begin(); // initialize dht
  Serial.println("\n\n**************************************");
  Serial.println(" Initializing IoT Termometer...");
  wifi_connect();
  client.setServer(mqtt_broker,1883); // Set your MQTT server and port
  pinMode(LED_BUILTIN, OUTPUT); // declares internal LED
}

void loop() {
  if (!client.connected()) {
    reconnect_MQTT();
  }
  DHT_readings();
  publish_MQTT();
  delay(5000);
}

// Function establishes connection with wifi network
void wifi_connect(){
  delay(10);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  digitalWrite(LED_BUILTIN,HIGH);
  Serial.println("**************************************\n\n");
}

//Seeks to re-establish the connection with the MQTT broker
void reconnect_MQTT(){
  while (!client.connected()) {
    client.connect("IoT_Termometer");
    Serial.println("Reconnecting MQTT...");
  }
}

// Function that takes dht readings and show results
void DHT_readings(){
  // Reading temperature or humidity takes about 250 milliseconds!
  h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  //float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

}

// Function publish readings in MQTT topic
void publish_MQTT(){
  // The true parameter persist information in topic
  client.publish("lab/temp", String(t).c_str(), true); // CHANGE THE TOPIC
  client.publish("lab/humid", String(h).c_str(), true); // CHANGE THE TOPIC

  // A series of prints to show readings
  Serial.println("\n---------------------------------------");
  Serial.println("Temperature and Humidity Readings: \n");
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));

  // Shows publication success in the MQTT topic blinking internal led
  digitalWrite(LED_BUILTIN, LOW);
  delay(200);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(200);
  digitalWrite(LED_BUILTIN, LOW);
  delay(200);
  digitalWrite(LED_BUILTIN, HIGH);
```

```
  delay(200);
}
```

O sensor DHT22 através do pino sinal dele, envia os dados para a placa ESP8266. A placa envia os dados via wifi para o servidor mqtt.eclipseprojects.io.
O MQTT (do inglês Message Queue Telemetry Transport) é um protocolo de comunicação entre máquinas (Machine to Machine - M2M). Ele utiliza dois conceitos: subscribed e publish. Um dispositivo pode ser "inscrito= subscribed" ou uma "publicação=publish". Quando tal dispositivo é um Publish, ele publica informações pré-estabelecidas. Já o dispositivo Subscribe recebe todas essas informações que estão sendo publicadas.
O Broker é um servidor que gere as informações aos inscritos e vice-versa, funcionando, então, como um intermediário entre Subscribers e Publishers. Neste projeto, o broker utilizado é o https://mqtt.eclipseprojects.io/ e porta: 1883
Após a configuração do aplicativo mqtt dashboard, é possível ver pelo smartphone, a temperatura e umidade do local.

5. Referências

BRASIL, ARDUINO. ESP8266 NODEMCU #04: CONFIGURANDO O PUBLISH DO MQTT (Termômetro IoT). Youtube, 21 DE JULHO DE 2020. Disponível em: < https://www.youtube.com/watch?v=vj2i1fSInx0&list=RDCMUChymjuTJ1BIsKai5RVNlF2A&index=4>.

BRASIL, ARDUINO. ESP8266 NODEMCU #05: CONFIGURANDO O SUBSCRIBE DO MQTT (TERMÔMETRO IoT). Youtube, 23 DE JULHO DE 2020. Disponível em: <https://www.youtube.com/watch?v=CvcTzarqQbI&list=RDCMUChymjuTJ1BIsKai5RVNlF2A&index=3>.

Curvello, André. Apresentando o módulo ESP8266. Disponível em: <https://www.embarcados.com.br/modulo-esp8266/>. Acesso em 11/11/2021.

GREENGARD, Samuel. The Internet of Things. Mit Press, 27 de março de 2015.

TUDO o que você precisa saber sobre Casa Inteligente e suas tecnologias.
INTERBRAS, 24 de novembro de 2020. Disponível em:
<https://blog.intelbras.com.br/casa-inteligente/>. Acesso em 31/08/2021