Monitoramento de ambientes com sensores utilizando MQTT

José Eduardo Batista do Nascimento

[1]Núcleo de Computação Eletrônica – NCE – Universidade Federal do Rio de Janeiro (UFRJ)
Prédio do CCMN - Bloco C, Caixa Postal: 2324 - CEP: 20.010-974

[2]Departamento de Ciência da Computação – Prédio do CCMN - UFRJ
Rio de Janeiro, Brasil

[3]Instituto de Matemática
Centro de Tecnologia – Rio de Janeiro, RJ – Brasil

eduardonascimento@poli.ufrj.br

Abstract. *This paper describes how to read a temperature sensor and publish the temperature in real time for monitoring.*

Resumo. *Este artigo visa descrever os passos necessários para implementar um sistema básico de monitoração em tempo real*

1. Dispositivos Utilizados

Para este trabalho foi utilizado um microcontrolador com interface de rádio embutido e suporte ao protocolo IEEE 802.11 mais conhecido como Wi-Fi chamado de ESP8266 integrada a um termistor, cuja resistência varia de acordo com a temperatura a qual o sensor está exposto. O módulo do sensor utilizado pode ser observado na figura abaixo.

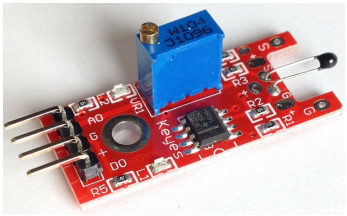

Figura 1. Módulo sensor para termistor

1.1. ESP8266

O ESP8266 é um microcontrolador do fabricante chinês Espressif que inclui capacidade de comunicação por Wi-Fi. Suas especificações são

– Wireless padrão 802.11 b/g/n
– Antena embutida
– Conector micro-usb
– Modos de operação: STA/AP/STA+AP
– Suporta 5 conexões TCP/IP
– Portas GPIO: 11
– GPIO com funções de PWM, I2C, SPI, etc

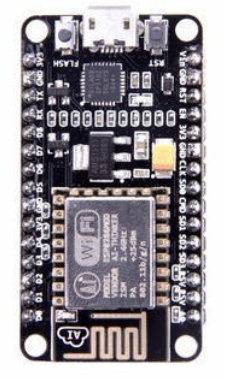

Figura 2. Placa NodeMCU utilizando ESP8266

– Tensão de operação: 4,5 9V
– Taxa de transferência: 110-460800bps
– Suporta Upgrade remoto de firmware
– Conversor analógico digital (ADC)

O NodeMCU possui antena embutida e conector micro-usb para conexão ao computador, além de 11 pinos de I/O e conversor analógico-digital.

1.2. Módulo Sensor Keyes-28

Esse módulo consiste de um NTC termistor comum, cuja resistência diminui com o aumento da temperatura. Suas características são:

- range de temperatura: -55C / +125C
- Pino digital que funciona como trigger
- Pino analógico
- Potenciômetro para regular threshold

2. Princípio de Funcionamento

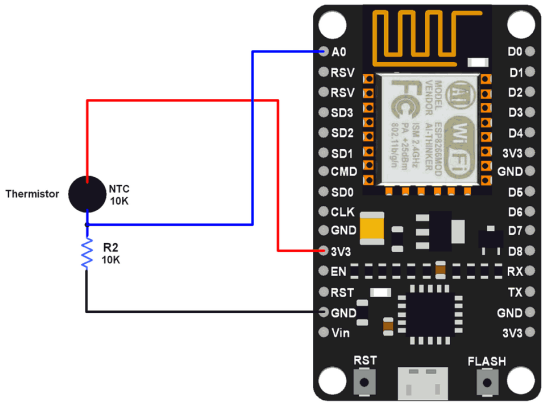

Figura 3. Placa NodeMCU utilizando ESP8266

O circuito descrito no vídeo que será disponibilizado segue a estrutura da figura acima. O NTC(Negative Temperature Coefficient) que é um resistor variável sensível a temperatura forma um divisor resistivo com um resistor de 10kΩ. Quando a temperatura varia, a tensão entre o NTC e o resistor irá variar. Este ponto está ligado a entrada analógica no microcontrolador que por sua vez, utiliza seu conversor AD de 10 bits para transformar a tensão medida num valor inteiro digital que posteriormente será utilizado para obtenção da temperatura. Podemos relacionar a resistência R do termistor com sua temperatura através da equação de Steinhart–Hart:

$$\frac{1}{T} = A + B\ln(R) + C[\ln(R)]^3 \tag{1}$$

Como sabemos a tensão de alimentação e o valor medido pelo microcontrolador, além do valor do resistor R2, é possível obter o valor de R.

3. MQTT

O protocolo de rede da Internet é o TCP/IP. Desenvolvido com base na pilha TCP/IP, o MQTT (Message Queue Telemetry Transport) tornou-se o padrão para comunicações de IoT. O MQTT é um protocolo de rede leve e flexível que oferece o equilíbrio ideal para os desenvolvedores de IoT. O protocolo leve permite a implementação em hardware de dispositivo altamente restringido e em redes de largura da banda limitada e de alta latência. Sua flexibilidade possibilita o suporte a diversos cenários de aplicativo para dispositivos e serviços de IoT.

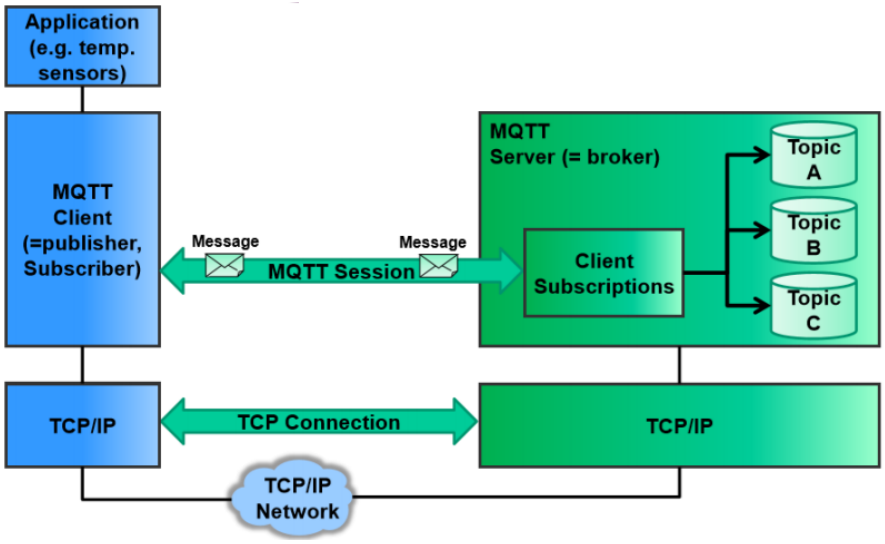

Figura 4. Diagrama de relações dentro do universo MQTT

3.1. Cliente MQTT

Um cliente MQTT pode ser qualquer dispositivo como um microcontrolador ou um PC rodando um server que utiliza a bibliotecas MQTT. Clientes podem ser tanto publicadores, quanto assinantes. Clientes assinam tópicos para publicar e receber mensagens.

3.2. Broker

O Broker tem como principal função receber, filtrar e encaminhar as mensagens aos clientes. O Broker comanda os tópicos e recebe assinaturas dos clientes.

3.3. Sessões, assinaturas e tópicos

Uma sessão é identificada como o estabelecimento de conexão entre um cliente e um broker. Diferente de uma sessão, uma assinatura estabelece uma conexão ou melhor um vínculo entre um cliente e um tópico. Além disso, tópicos permitem clientes trocarem informações dentro de um campo semântico e são baseados no padrão publish/subscribe.

3.4. Modelo Publish-Subscribe

Alternativa ao tradicional modelo cliente-servidor, o Publicador e Assinante não precisam se conhecer(pelo ip ou port). Além disso, o Publicador e Assinante não precisam estar sincronizados e o Intermediário(Broker) conhece tanto o publicador, quanto o assinante.

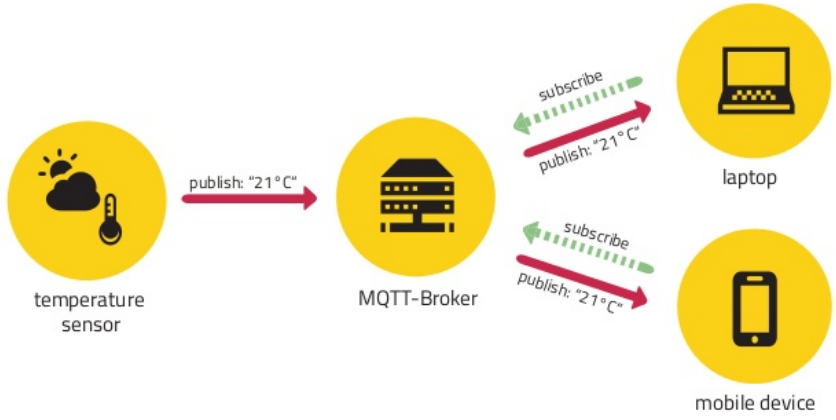

Figura 5. Esquema de comunicação entre o sensor de temperatura e o Broker através do protocolo MQTT

4. Zerynth

O Zerynth é o middleware para dispositivos inteligentes, aplicativos IoT e Industry 4.0. Em outras palavras, é o "Android para o mundo embarcado", suportando aplicações em manufatura, varejo, robótica, automação residencial e todos os outros setores do mercado onde a IoT desempenhará o papel principal.
Esse middleware roda sobre o browser e permite programar os mais variados microcontroladores de 32 bits interface Wi-Fi embutida e o ESP8266 está nessa lista. Os projetos são programados em python. Para usá-lo com ESP8266 é necessário instalar uma flash ROM customizada, disponilizada pelo próprio fabricante.

5. O código

Figura 6. Logo do middleware utilizado neste trabalho

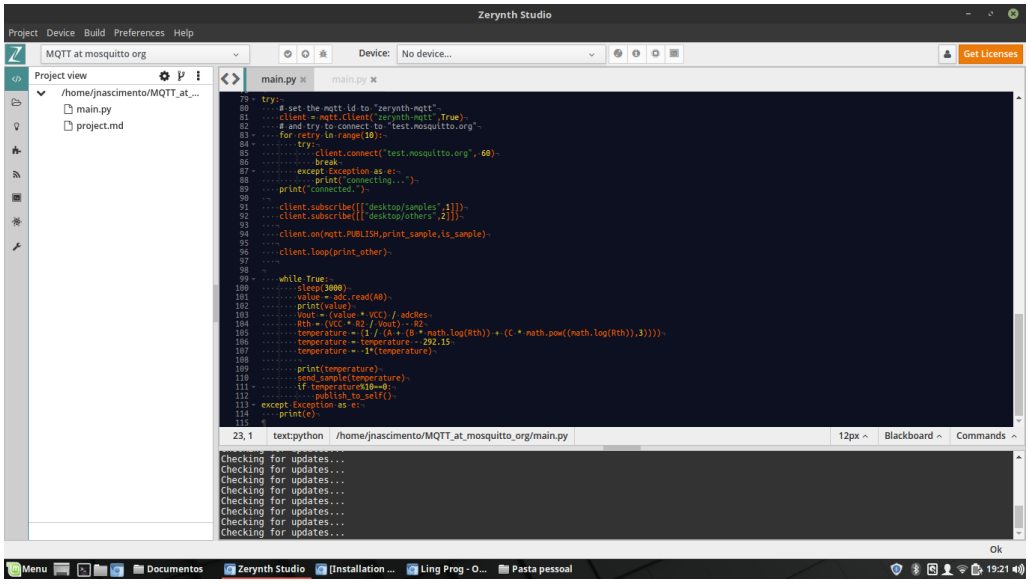

Figura 7. Captura de tela do editor

```
while True:
        sleep(3000)
        value = adc.read(A0)
        print(value)
        Vout = (value * VCC) / adcRes
        Rth = (VCC * R2 / Vout) - R2
        temperature = (1 / (A + (B * math.log(Rth)) + (C * math.
   pow((math.log(Rth)),3))))
        temperature = temperature - 292.15
        temperature = -1*(temperature)
```

.

O código acima é responsável por calcular a resistência equivalente do termistor e posteriormente calcular sua temperatura.

```

def send_sample(obj):
    print("publishing: ", obj)
    client.publish("temp/random", str(obj))

 print(temperature)
        send_sample(temperature)
        if temperature%10==0:
            publish_to_self()
except Exception as e:
```

```
11        print(e)
```

.

A função sendsample() é responsável por publicar os valores de temperatura para o broker.

O código completo pode ser acessado neste link: github

Referências

[1] Zerinth Middleware. *Installation Guide*, https://docs.zerynth.com/latest/index.html

[2] O protocolo MQTT, *Leve e simples. Perfeito para IoT e sistemas embarcados*, https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/mqtt/, Jose E.B. Nascimento, Matheus Molin, Luis Octávio, Rio de Janeiro, Universidade Federal do Rio de Janeiro, 2018

[3] Things flow Python, *Streaming dataflow library for IoT applications. Program at a higher level, with reusable "things"*, https://github.com/mpi-sws-rse/thingflow-python, Author: jfischer

[4] Mosquitto, *hosts a publicly available Mosquitto MQTT server/broker.*, https://test.mosquitto.org/,