**Anderson Bezerra Calixto - Sistemas Distribuídos**

**Vídeos da implementação:** [**https://www.youtube.com/watch?v=GgpRmmKVMMI**](https://www.youtube.com/watch?v=GgpRmmKVMMI)

**e** [**https://www.youtube.com/watch?v=fV51YOV4-UY**](https://www.youtube.com/watch?v=fV51YOV4-UY)

**Repositório no github:** [**https://github.com/andersonbr/mqtt**](https://github.com/andersonbr/mqtt)

Arquiteturas implementadas:

**Socket Cliente/Servidor (em Python) - no diretório “socket” no github**

O servidor guarda uma lista com os estados dos clientes conectados. Quando um cliente desconecta, é retirado dessa lista.

O servidor recebe a descrição do equipamento com um comando DESC, seguido da descrição. O estado é enviado com um comando ESTADO, seguido de seu estado.

O servidor cria duas threads de serviços socket, um na porta 9999, para comunicação com os equipamentos, e outra na porta 3001, para fornecer um serviço web que será a forma de gerenciamento, que responde no protocolo HTTP (implementado manualmente o básico)

Clientes (equipamentos) vão se conectar ao servidor e enviar informações em texto plano, com comandos definidos para o servidor, informando sua descrição e estado atual.

Os clientes recebem do servidor um comando MOD, seguido de um novo estado, para ser modificado em seu valor interno.

**MQTT (pub/sub) - na raíz do repositório no github, implementado em Javascript (nodejs)**

1. Manager (Node.JS com client MQTT) – arquivo: manager.js
   1. Subscriptions
      1. presence
         * Faz com que o manager saiba quem será monitorado/gerenciado
         * Recebe os parâmetros atuais dos equipamentos, os tipos de parâmetros, etc.
      2. die
         * Faz com que o manager saiba quem desconectou.
   2. Publishers
      1. whoalive
         * Força o envio da mensagem “presence” dos equipamentos conectados, uma espécie de ping.
      2. eqps/ID\_DO\_EQUIPAMENTO
         * Envia modificação de valores dos parâmetros dos equipamentos
2. Nodes (5 elementos)
   1. Subscriptions
      1. eqps/ID\_DO\_EQUIPAMENTO
         * Fica pronto para receber modificações dos dados vindo do manager.
      2. whoalive
         * Fica disponível para receber solicitação do “ping”, que será respondido com um “presence”
   2. Publishers
      1. presence
         * Informará os parâmetros atuais do equipamento
      2. die
         * Publish “implícito”, definido em forma de “Last Will and Testament (LWT)”, que será enviado em caso de conexão perdida com o MQTT Broker (Mosquitto)
3. Broker utilizado: Mosquitto

No manager, foi disponibilizada uma interface Web, com uso do framework express, que fornece recursos de webserver, para fornecer arquivos estáticos (index.html, scripts javascript e estilos CSS do diretório /static)

Ao carregar a página, é enviado um whoalive, para receber de todos os equipamentos um “presence”.

Componentes HTML e CSS são utilizados para representar os dados dos equipamentos, com 3 tipos implementados:

* switch: para chaves (on/off);
* info: para informativos do tipo sensor;
* ranger: para definição de um valor dentro de um intervalo (ex: potenciômetros, etc)

Tecnologias utilizadas na interface Web

Frontend:

* HTML
* Javascript
* CSS
* jQuery
* Bootstrap
* Socket.io

Backend

* Node.JS
* Express
* MQTT.js

Para implementação física foi utilizado o ESP8266, com firmware NodeMCU, que utiliza a linguagem LUA.

Os parâmetros utilizados no protocolo MQTT (publishers e subscribers) são análogos aos virtuais em NodeJS.

Ambiente de equipamento virtual:

* Docker com imagem node:latest

Mais detalhes nos vídeos informados no início deste documento