**INST. FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS**

**Disciplina:** Sistemas Distribuídos

**Curso:** Sistemas de Informação **Turno:** Noturno **Data:** 25/01/2023

**Professor:** Charles Tim Batista Garrocho

**Nome:** Anderson Gabriel e Fabricio Augusto

**RELATÓRIO TRABALHO FINAL SISTEMAS DISTRIBUÍDOS**

**Proposta de Trabalho**

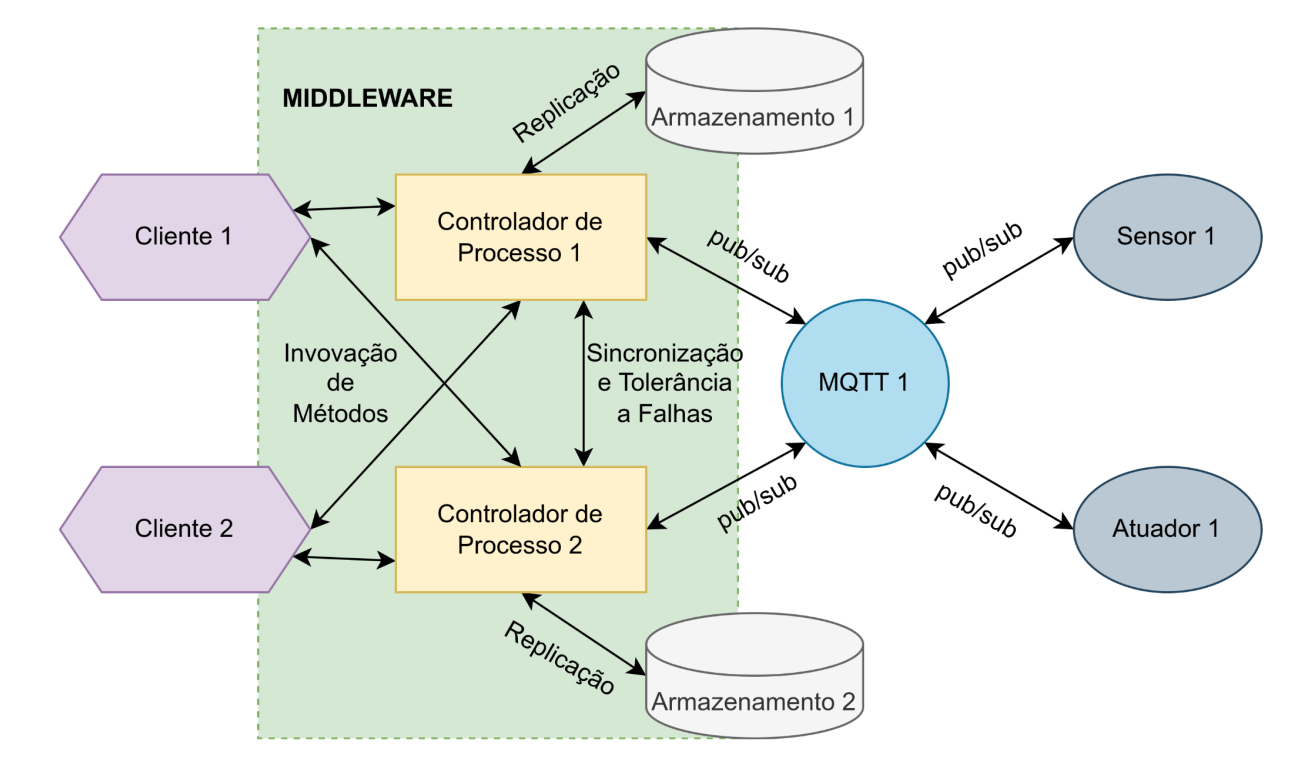
O objetivo desse trabalho é aplicar os conhecimentos adquiridos durante o semestre da matéria de Sistemas Distribuídos. A ideia é implementar um middleware para fazer integração de sensores e atuadores MQTT e uma aplicação Cliente RPC. A figura abaixo mostra a proposta apresentada pelo Professor:

Figura 1

O trabalho foi realizado simulando um carro de Formula 1, onde dispõe de sensor (Sensor – Figura 1) para medir a temperatura dos pneus, além de uma central (Controlador de processo – Figura 1) que gerencia essas temperaturas e informa ao piloto através de um visor presente no volante(Atuador – Figura 1), informando ao piloto para ir para os boxes. Além desse dispositivo, a equipe (Cliente – Figura 1) consegue através de uma solicitação a central, enviar uma mensagem personalizado para o piloto.

O desenvolvimento do trabalho foi realizado em python, além de usar um servidor baseado em eventos (MQTT) e um servidor baseado em objetos (RPYC) para montagem e elaboração do código. Conforme script abaixo:

Central.py

*from email.quoprimime import header\_decode*

*from multiprocessing.connection import Client*

*import turtle*

*import time*

*import paho.mqtt.client as paho*

*import time*

*import random*

*import paho.mqtt.client as mqtt*

*import rpyc*

*#hostname*

*broker="localhost"*

*#port*

*port=1883*

*#time to live*

*timelive=60*

*delay = 0.01*

*def on\_connect(client, userdata, flags, rc):*

*print("Connected with result code "+str(rc))*

*client.subscribe("carro")*

*def on\_message(client, userdata, msg):*

*temp = msg.payload.decode()*

*print(str(temp))*

*if int(temp) > 110:*

*message= "Cuidado, temperatura alta. Vá para os box's"*

*print("Recebendo")*

*ret= client.publish("volante", message)*

*time.sleep(5)*

*def on\_message\_equipe(client, userdata, msg):*

*ret= client.publish("volante", msg)*

*time.sleep(5)*

*client = mqtt.Client()*

*client.connect(broker,port,timelive)*

*client.on\_connect = on\_connect*

*client.on\_message = on\_message*

*client.on\_message\_equipe = on\_message\_equipe*

*client.loop\_start()*

*time.sleep(5)*

*class MeuServico(rpyc.Service):*

*def exposed\_message\_pilot(self, message):*

*on\_message\_equipe(client, self, message)*

*from rpyc.utils.server import ThreadedServer*

*t = ThreadedServer(MeuServico, port=18866)*

*t.start()*

O sensor faz a simulação de um medidor de temperatura presente nos pneus dos carros, onde o mesmo envia dados para a central que determina se é hora de ir ou não para os boxes. Conforme script abaixo:

Sensor.py

*from email.quoprimime import header\_decode*

*from multiprocessing.connection import Client*

*import turtle*

*import time*

*import paho.mqtt.client as paho*

*import random*

*import paho.mqtt.client as mqtt*

*#hostname*

*broker="localhost"*

*#port*

*port=1883*

*#time to live*

*timelive=60*

*delay = 0.01*

*def on\_connect(client, userdata, flags, rc):*

*print("Connected with result code "+str(rc))*

*client.subscribe("central")*

*def on\_message(client, userdata, msg):*

*print(msg.payload.decode())*

*client = mqtt.Client()*

*client.connect(broker,port,timelive)*

*client.on\_connect = on\_connect*

*client.on\_message = on\_message*

*client.loop\_start()*

*while True:*

*temp = random.randint(105,115)*

*ret= client.publish("carro", str(temp))*

*print("Publicou" + str(temp))*

*time.sleep(5)*

O volante possui um painel responsável por informar o piloto quando a central determinar que é o momento de ir para os boxes, além disso, é possível a equipe enviar uma mensagem para o piloto e exibi-la através deste painel. Conforme script abaixo:

Volante.py

*from email.quoprimime import header\_decode*

*from multiprocessing.connection import Client*

*import turtle*

*import paho.mqtt.client as paho*

*import time*

*import paho.mqtt.client as mqtt*

*import random*

*#hostname*

*broker="localhost"*

*#port*

*port=1883*

*#time to live*

*timelive=60*

*delay = 0.01*

*def on\_connect(client, userdata, flags, rc):*

*print("Connected with result code "+str(rc))*

*client.subscribe("volante")*

*def on\_message(client, userdata, msg):*

*print(msg.payload.decode())*

*client = mqtt.Client()*

*client.connect(broker,port,timelive)*

*client.on\_connect = on\_connect*

*client.on\_message = on\_message*

*client.loop\_forever()*

A equipe possui uma conexão com a central, onde através dela é possível enviar uma mensagem para o volante do piloto. Conforme script a seguir:

Equipe.py

*import rpyc*

*from pynput.keyboard import Key, Listener*

*proxy = rpyc.connect('localhost', 18866, config={'allow\_public\_attrs': True})*

*print('1 - Enviar mensagem para o piloto')*

*print('2 - Sair ')*

*op = input()*

*while(op != '2'):*

*if(op == '1'):*

*message = input()*

*proxy.root.message\_pilot(message)*

*print('1 - Enviar mensagem para o piloto')*

*print('2 - Sair ')*

*op = input()*

**Conclusão**

Sistemas distribuídos nos mostra como é possível integrar dois ou mais estilos arquitetônicos em um mesmo sistema, integrandos de forma onde o middleware controla toda a operação e entrega para o cliente ou atuador apenas o necessário sem que os mesmos tenham custo com esse processamento.

**Link repositório GitHub:**

https://github.com/AndersonGC/goPitStop