===========================================================================

**Process and visualize IoT data**

Miguel António - 2022237621 e Pedro Azevedo - 2022209667

====================================================================

Neste projeto, foi-nos proposto a criação de uma solução IoT completa desde a recolha de dados até ao seu processamento e visualização.

O código enviado juntamente com este report, dividi-se em 5 scripts de acordo com o que nos foi proposto realizar. O **Room.py** simula os dados recolhidos por sensores de uma sala e publica os dados no tópico “14\_ROOM\_DATA” no MQTT broker de modo a ser acedido pela **DPU.py (Data Processing Unit)**. Os dados enviados são em formato JSON:

room\_data = {

    "temperature": 20.0,

    "humidity": 50,

    "motion": 0,

    "ac\_working": 0,

    "hc\_working": 0,

    "sensor\_type": 1,

    "room\_number": ROOM\_NUMBER

}

(room\_number é definido no enunciado como sendo 5)

Ficheiro com os intervalos e os valores ideais:

[ideal\_conditions]

min\_temperature = 15.0

max\_temperature = 30.0

ideal\_temperature = 23.0

min\_humidity = 40

max\_humidity = 60

ideal\_humidity = 50

A **DPU.py**,lê um ficheiro “ideal.cfg”, que tem os intervalos a que a temperatura e a humidade devem respeitar e a temperatura e humidade ideal e processa os dados recebidos via MQTT e subscreve 2 tópicos: “14\_ROOM\_DATA” e “OTHER\_ROOMS”, sendo os dados do segundo tópico em formato JSON e simulados pelo professor. Este script armazena os dados processados e define vários alertas: para o caso da temperature ou da humidade estejam fora do intervalo definido por “ideal.cfg”; para o caso de os alarmes da AlarmConsole.py estejam ligados e seja detetado movimento; para o caso dos dados recolhidos não sejam do sensor do tipo 1 ou do tipo 2; para caso do AlarmConsole.py ligar/desliguar os alarmes; para o caso de não terem sido recebidos dados dos sensores por 15 segundos.

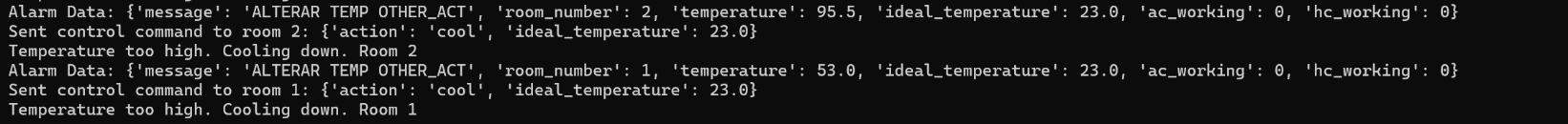
Caso não haja nenhum alarme, a DPU guarda os dados num measurement do InfluxDB “normal\_data” (em que as colunas são “temperature”, “humidity”, “motion”, “ac\_working”, “hc\_working”, “sensor\_type” e “room\_number”) e caso haja guarda em “eventos” (com apenas uma coluna designada “evento”, que é uma string da descrição do evento como está nos alarmes). Os alarmes são enviados para a **ControlCentral.py** usando TCP em para que esta comunique com a Room.py e OTHER\_ROOMS.

Caso a temperatura esteja fora do intervalo definido, o “ac\_working” vai ter que se ligar tomando o valor 1, e a temperatura em vez de ser gerada aleatoriamente vai ser gerada aumentando/diminuindo (conforme o necessário) 0.5 até chegar à temperatura ideal. O mesmo acontece com a humidade, em que caso a humidade esteja fora do intervalo definido, o “hc\_working” liga-se e toma o valor 1, e a humidade deixa de ser gerada aleatoriamente e é gerada aumentando/diminuindo (conforme o necessário) 5 até chegar á humidade ideal. Se os dados gerados pelo sensor do tipo 2 são em Fahrenheit, pelo que são convertidos para celsius e processados de

A **AlarmConsole.py** recebe comandos do utilizador para ligar ou desligar os alarmes de movimento e esta informação é enviada para o tópico “14\_ROOM\_DATA” usando MQTT para que a DPU os processe. Se o alarme for desligado os dados são descartados.

A **ControlCentral.py** recebe os alarmes do DPU e de acordo com o alarme realiza diferente tarefas.

- se os valores estiverem fora dos intervalos, caso seja a temperatura é enviada para o tópico “14\_ACT” a room\_number, temperatura ideal e ac\_working = 1, para que depois a Room.py adapte a forma como gera dados explicada assim. Caso seja a humidade, é enviada para o tópico “14\_ACT” a room\_number, humidade ideal e hc\_working = 1. Se o alarme for de não receber dados nos últimos 15 segundos é enviada uma mensagem para que se desconecte dessa sala. Os outros alarmes de movimento e da ativação dos alarmes de movimento também são recebidos.

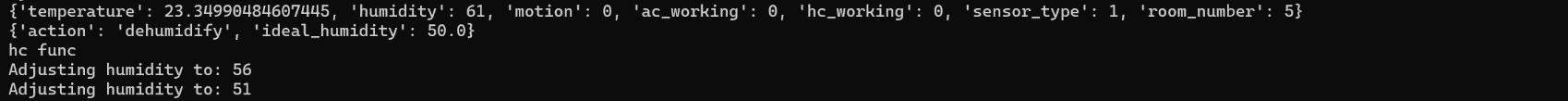
 O **MQTTdebugger.py** tem como propósito escrever todas as mensagens que foram transferidas pelo MQTT broker, serve como histórico e a cada mensagem é do tipo: [time]:[topic]:[message].

Execução do ControlCentral.py:

Uma imagem com texto, captura de ecrã, preto, Tipo de letra

Descrição gerada automaticamenteExecução do DPU.py :

(VALORES INSERIDOS representa um print para ver o que está a ser guradado na bade de dados do InfluxDB)

Execução do Room.py:

**Uma imagem com texto, captura de ecrã

Descrição gerada automaticamenteVisualização de Dados:**

Figura 1 – Visualização das Componentes da Humidade e Temperatura ao longo do tempo (série temporal) e num instante

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 2 - Visualização da Componente Motion ao longo do tempo (série temporal)

A partir da série tempooral da temperatura da Figura 1, observa-se que os como ao se gerarem valores aleatórios de [-0.1,0.5] a temperatura vai tender sempre a aumentar de maneira que há de chegar a altura em que vai ultrapassar o limite máximo (30ºC) pelo que a ControlCentral.py vai mandar informação para que a Room.py ajuste (ligando o AC) para o valor da temperatura ideal (23ºC). O mesmo acontece para a série temporal da humidade.

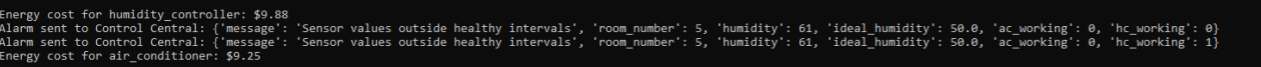
Dadas as restrições da escrita do relatório (máximo de 3 páginas) a visualização do custo, dos sensores e dos ac e hc será feita durante a defesa.

Visualização dos alarmes no InfluxDB:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteVisualização do custo de energia do AC/HC: