Projeto para a Disciplina de Integração de Sistemas de Informação (ISI)

None

PEQSPC

None

Table of contents

1. ProjetoISIv1 — Monitorização Industrial com ETL e IoT

Para entender todos os detalhes técnicos e de configuração do projeto, lê toda a documentação disponível no site gerado pelo **MkDocs**.

1.1 Introdução

O **ProjetoISIv1** é um trabalho prático da Unidade Curricular **Integração de Sistemas de Informação (ISI)** — Licenciatura em Engenharia de Sistemas Informáticos (2025/26).
O objetivo principal é aplicar e experimentar **ferramentas de ETL (Extract, Transform, Load)** em processos de integração de dados no contexto de **monitorização industrial com sensores IoT**.

1.2 Problema

A recolha de dados das estações de produção era feita manualmente e de forma isolada, o que impedia:

- A **monitorização em tempo real**;
- A identificação de **anomalias**;
- A integração e análise de métricas de **produção, paragem, stock e defeitos**.

1.3 Objetivos

- Centralizar dados de produção, stock e paragens num repositório único;
- Automatizar a recolha de dados via **sensores MQTT**;
- Normalizar e validar dados antes do armazenamento;
- Enriquecer as leituras com **dados externos via API**;
- Gerar relatórios automáticos e enviar por email;
- Visualizar indicadores em tempo real com **React + UI Builder + Grafana**.

1.4 Estrutura da Documentação

1.5 Sobre o Projeto

- **Nome dos Autores:** PEQSPC
- **Curso e unidade curricular** LESI-IPCA ISI(Integracao de Sistemas de Informaçao)
- **Ano letivo (2025/26)**
- **Link para o repositório Git** Github

2. Arquitetura Técnica do Projeto

A arquitetura técnica do **ProjetoISIv1** suporta todo o ciclo de recolha, transformação e visualização dos dados industriais.

2.1 Componentes Principais

2.1.1 1. Publisher (Simulador MQTT em Python)

- Gera dados de produção, stock, paragens e defeitos.
- Publica mensagens MQTT em tópicos por estação.

2.1.2 2. Broker MQTT — Eclipse Mosquitto

- Atua como intermediário entre sensores (publishers) e Node-RED (subscriber).
- Porta padrão: `1883`.

2.1.3 3. Node-RED (ETL)

- Recebe dados MQTT.
- Processa e acumula valores.
- Guarda resultados em **SQLite3**.
- Envia relatórios e aciona scripts Python.

2.1.4 4. Base de Dados SQLite3

- Armazena dados acumulados e históricos.
- Tabelas: `estacoes`, `dados_acumulados`, `relatorios`, `produtos`.

2.1.5 5. Enriquecimento via API

- Obtém **preços das peças** e cruza com dados de produção.
- Utiliza **expressões regulares (Regex)** para filtrar os produtos relevantes.

2.1.6 6. Python Reporting

- Gera relatórios automáticos sobre margens e desempenho.
- Envia por email.

2.1.7 7. Dashboard (React + Material UI)

- Exibe produção, paragens e eficiência em tempo real.
- Interface moderna criada com UI Builder.

2.2 Fluxo Global de Dados

```
graph TD
  A[Sensores Python] --> B[Broker MQTT (Mosquitto)]
  B --> C[Node-RED ETL]
  C --> D[SQLite3]
  D --> E[API Preços]
  D --> F[Python Reporting]
  F --> G[Email Automático]
  D --> H[Dashboard React]
```

3. ETL PROCESSOS

```markdown

# 4. Processos ETL e Transformações

O **Node-RED** atua como o motor central de **ETL** (Extract, Transform, Load).

## 4.1 Estratégia

1. **Extração (Extract)**
   - Dados recolhidos de sensores MQTT.
   - Publicação periódica dos valores de produção, stock e paragem.
2. **Transformação (Transform)**
   - Normalização dos dados.
   - Cálculo de acumulados.
   - Validação e filtragem de outliers.
3. **Carregamento (Load)**
   - Inserção e atualização dos dados em **SQLite3**.
   - Geração de logs e relatórios.

## 4.2 Exemplos de Transformações

| Tipo | Operação | Descrição |
|---|---|---|
| Limpeza | Eliminar valores nulos | Ignorar mensagens sem payload válido |
| Normalização | Converter tipos | Garantir que todos os valores são `float` |
| Agregação | Somar produção acumulada | `acumulado += msg.payload.producao` |
| Regex | Extrair ID do produto | `/produto-(\d+)/` |
| Filtro | Ignorar paragens < 2s | `if (msg.payload.paragem >= 2)` |

## 4.3 Jobs Node-RED

Os fluxos Node-RED incluem:

1. **MQTT In → Function → SQLite Out**
2. **Function → Python Exec → Email Out**

Esses fluxos são documentados visualmente no ficheiro `flows.json` e explicados no relatório final.
```

5. Integração com API

Neste houve uma integraçao com uma API para integrar os preços da peças feitas nas estaçoes para adicionar ao relatorio.

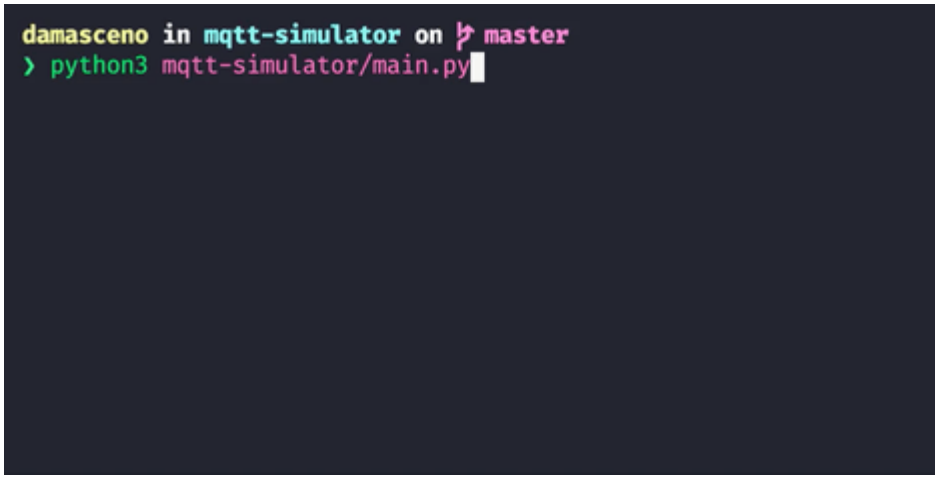
6. Base de Dados

7. Simulador de Sensores

7.1 MQTT Simulator

A lightweight, easy-to-configure MQTT simulator written in Python 3 for publishing JSON objects to a broker, simulating sensors and devices.

Features • Getting Started • Configuration • Main contributors



7.1.1 Features

- Lightweight and easy-to-configure simulator for publishing data to an MQTT broker
- Simple setup with a single JSON configuration file
- Publish data on predefined fixed topics
- Publish data on multiple topics that have a variable id or items at the end
- Simulated random variation of data based on configurable parameters
- Real-time event logging during simulation

7.1.2 Getting Started

Running using Python

Run the simulator with the default settings file (`config/settings.json`):

```
python3 mqtt-simulator/main.py
```

Or specify a custom settings file:

```
python3 mqtt-simulator/main.py -f <path/settings.json>
```

To install all dependencies with a virtual environment before using:

```
python3 -m venv venv
source venv/bin/activate
pip3 install -r requirements.txt
```

Running using uv

Run the simulator with uv, a fast Python package and project manager - no need to manually setup a virtual environment:

```
uv run mqtt-simulator/main.py -f <path/settings.json>
```

Running using Docker

Additionally, you can run the simulator via Docker using the provided `Dockerfile`.

Build the image:

```
docker build -t mqtt-simulator .
```

Run the container:

```
docker run mqtt-simulator -f <path/settings.json>
```

7.1.3 Configuration

See the configuration documentation for detailed usage instructions.

You can also check a full settings file example at: settings.json.

Below is a minimal configuration file that connects to the `mqtt.eclipseprojects.io` broker and publishes data to the `/place/roof` and `/place/basement` topics. The simulator generates `temperature` variations based on the provided parameters:

```
{
  "BROKER_URL": "mqtt.eclipseprojects.io",
  "TOPICS": [
    {
      "TYPE": "list",
      "PREFIX": "place",
      "LIST": ["roof", "basement"],
      "TIME_INTERVAL": 8,
      "DATA": [
        {
          "NAME": "temperature",
          "TYPE": "float",
          "MIN_VALUE": 20,
          "MAX_VALUE": 55,
          "MAX_STEP": 3,
          "RETAIN_PROBABILITY": 0.5,
          "INCREASE_PROBABILITY": 0.6
        }
      ]
    }
  ]
}
```

7.1.4 Sensores Simulados

Sensor	Tipo de dado	Unidade	Intervalo de emissão	Exemplo de payload
Produção	Peças por minuto	pcs/min	5 s	{ "estacao": 1, "producao": 58 }
Paragem	Tempo parado	segundos	evento	{ "estacao": 1, "paragem": 15 }
Stock	Nível de stock	%	10 s	{ "estacao": 1, "stock": 73 }
Qualidade	Peças defeituosas	%	10 s	{ "estacao": 1, "defeitos": 4 }

7.2 Configuration

The MQTT Simulator configuration consists of 3 main sections: **Broker**, **Topics**, and **Data**.

Quick Navigation:
Broker settings • Topics settings • Data settings

You can also check a full settings file example at: settings.json.

7.2.1 Broker settings

The **Broker settings** section is located at the root level of the JSON configuration file and defines the fundamental MQTT connection parameters:

```
{
    "BROKER_URL": "mqtt.eclipse.org",
    "BROKER_PORT": 1883,
    "TOPICS": [
        ...
    ]
}
```

Key	Type	Default	Description
`BROKER_URL`	string	localhost	The broker URL where the data will be published
`BROKER_PORT`	number	1883	The port used by the broker
`PROTOCOL_VERSION`	number	4	Sets the paho.mqtt.client `protocol` param. Version of the MQTT protocol to use for this client. Can be either `3` (MQTTv31), `4` (MQTTv311) or `5` (MQTTv5)
`TLS_CA_PATH`	string	None	Sets the paho.mqtt.client.tls_set `ca_certs` param. String path to the Certificate Authority certificate file
`TLS_CERT_PATH`	string	None	Sets the paho.mqtt.client.tls_set `certfile` param. String path to the PEM encoded client certificate file
`TLS_KEY_PATH`	string	None	Sets the paho.mqtt.client.tls_set `keyfile` param. String path to the PEM encoded client private keys file
`AUTH_USERNAME`	string	None	Sets the paho.mqtt.client.username_pw_set `username` param. Username to authenticate with
`AUTH_PASSWORD`	string	None	Sets the paho.mqtt.client.username_pw_set `password` param. Password to authenticate with
`CLEAN_SESSION`	bool	True	Sets the paho.mqtt.client `clean_session` param. Boolean that determines the client type. This property is ignored if `PROTOCOL_VERSION` is `5`.
`RETAIN`	bool	False	Sets the paho.mqtt.client.publish `retain` param. If set to true, the message will be set as the “last known good”/retained message for the topic
`QOS`	number	2	Sets the paho.mqtt.client.publish `qos` param. Quality of service level to use
`TIME_INTERVAL`	number	10	Time interval in seconds between submissions towards the topic
`TOPICS`	array\	None	Specification of topics and how they will be published

7.2.2 Topics settings

The **TOPICS** key is a list. Each topic entry is an `object` containing parameters that define how topics will be structured and published:

```
{
    "TYPE": "multiple",
    "PREFIX": "place",
    "RANGE_START": 1,
    "RANGE_END": 2,
    "TIME_INTERVAL": 25,
    "DATA": [
        ...
    ]
}
```

Key	Type	Description	Required
`TYPE`	string	It can be `"single"`, `"multiple"` or `"list"`	yes
`PREFIX`	string	Prefix of the topic URL, depending on the `TYPE` it can be concatenated to `/<id>` or `/<item>`	yes
`LIST`	array\	When the `TYPE` is `"list"` the topic prefix will be concatenated with `/<item>` for each item in the array	if `TYPE` is `"list"`
`RANGE_START`	number	When the `TYPE` is `"multiple"` the topic prefix will be concatenated with `/<id>` where `RANGE_START` will be the first number	if `TYPE` is `"multiple"`
`RANGE_END`	number	When the `TYPE` is `"multiple"` the topic prefix will be concatenated with `/<id>` where `RANGE_END` will be the last number	if `TYPE` is `"multiple"`
`CLEAN_SESSION`	bool	Overwrites the broker level config value and applies only to this Topic	no
`RETAIN`	bool	Overwrites the broker level config value and applies only to this Topic	no
`QOS`	number	Overwrites the broker level config value and applies only to this Topic	no
`TIME_INTERVAL`	number	Overwrites the broker level config value and applies only to this Topic	no
`PAYLOAD_ROOT`	object	The root set of params to include on all messages	optional
`DATA`	array\	Specification of the data that will form the JSON to be sent in the topic	yes

7.2.3 Data settings

The key **DATA** inside TOPICS is a list. Each data entry is an `object` containing parameters that define individual data properties and how values are simulated:

```
{
    "NAME": "temperature",
    "TYPE": "float",
    "INITIAL_VALUE": 35,
    "MIN_VALUE": 20,
    "MAX_VALUE": 55,
    "MAX_STEP": 0.2,
    "RETAIN_PROBABILITY": 0.5,
    "RESET_PROBABILITY": 0.1,
    "INCREASE_PROBABILITY": 0.7,
    "RESTART_ON_BOUNDARIES": true
}
```

Key	Type	Description	Required
`NAME`	string	JSON property name to be sent	yes
`TYPE`	string	It can be `"int"`, `"float"`, `"bool"`, `"math_expression"` or `"raw_values"`	yes
`INITIAL_VALUE`	same that is returned according to `TYPE`	Initial value that the property will assume when the simulation starts. If not specified: random for `"int"`, `"float"` or `"bool"`, and determined by other parameters for `"math_expression"` or `"raw_values"`	optional
`RETAIN_PROBABILITY`	number	Number between 0 and 1 for the probability of the value being retained and sent again	optional, default is `0`
`RESET_PROBABILITY`	number	Number between 0 and 1 for the probability of the value being reset to `INITIAL_VALUE`	optional, default is `0`
`MIN_VALUE`	number	Minimum value that the property can assume	if `TYPE` is `"int"` or `"float"`
`MAX_VALUE`	number	Maximum value that the property can assume	if `TYPE` is `"int"` or `"float"`
`MAX_STEP`	number	Maximum change that can be applied to the property from a published data to the next	if `TYPE` is `"int"` or `"float"`
`INCREASE_PROBABILITY`	number	Number between 0 and 1 for the probability of the next value being greater than the previous one	optional, default is `0.5` (same probability to increase or decrease). Only valid if `TYPE` is `"int"` or `"float"`
`RESTART_ON_BOUNDARIES`	bool	When true and the value reaches `MAX_VALUE` or `MIN_VALUE` the next value will be the `INITIAL_VALUE`	optional, default is false. Only valid if `TYPE` is `"int"` or `"float"`
`MATH_EXPRESSION`	string	Math expression written in a *Pythonic* way. Also accept functions from Math modules	if `TYPE` is `"math_expression"`
`INTERVAL_START`	number	Minimum value that the `MATH_EXPRESSION`'s variable `x` can assume	if `TYPE` is `"math_expression"`
`INTERVAL_END`	number	Maximum value that the `MATH_EXPRESSION`'s variable `x` can assume	if `TYPE` is `"math_expression"`
`MIN_DELTA`	number	Minimum value that can be added to the `MATH_EXPRESSION`'s variable `x` from a published data to the next	if `TYPE` is `"math_expression"`
`MAX_DELTA`	number	Maximum value that can be added to the `MATH_EXPRESSION`'s variable `x` from a published data to the next	if `TYPE` is `"math_expression"`
`INDEX_START`	number	The index to start publishing from the `VALUES` array	optional, default is `0`. Only valid if `TYPE` is `"raw_values"`
`INDEX_END`	number	The index to end publishing from the `VALUES` array	optional, default is `len(values) - 1`. Only valid if `TYPE` is `"raw_values"`
`RESTART_ON_END`	bool		

Key	Type	Description	Required
		When true and the index of the `VALUES` array reaches `INDEX_END`, the next index will be `INDEX_START`. Otherwise, the param will become inactive and won't be sent after reaching `INDEX_END`	optional, default is false. Only valid if `TYPE` is `"raw_values"`
`VALUES`	array\	The values to be published in array order	if `TYPE` is `"raw_values"`
`VALUE_DEFAULT`	object	The default value params used or overwritten by params in `VALUES`	optional, default is `{}`. Only valid if `TYPE` is `"raw_values"` and `VALUES` is an array\

> ***NOTE:*** Access math_expression.md file for more explanations and a example of `TYPE: "math_expression"`.

7.3 MQTT Simulator - Data type `math_expression`

For general information on how to configure the MQTT Simulator see the README.md file.

For `TYPE: "math_expression"` we need five required configuration parameters: `MATH_EXPRESSION`, `INTERVAL_START`, `INTERVAL_END`, `MIN_DELTA` and `MAX_DELTA`. Each of these have some notes:

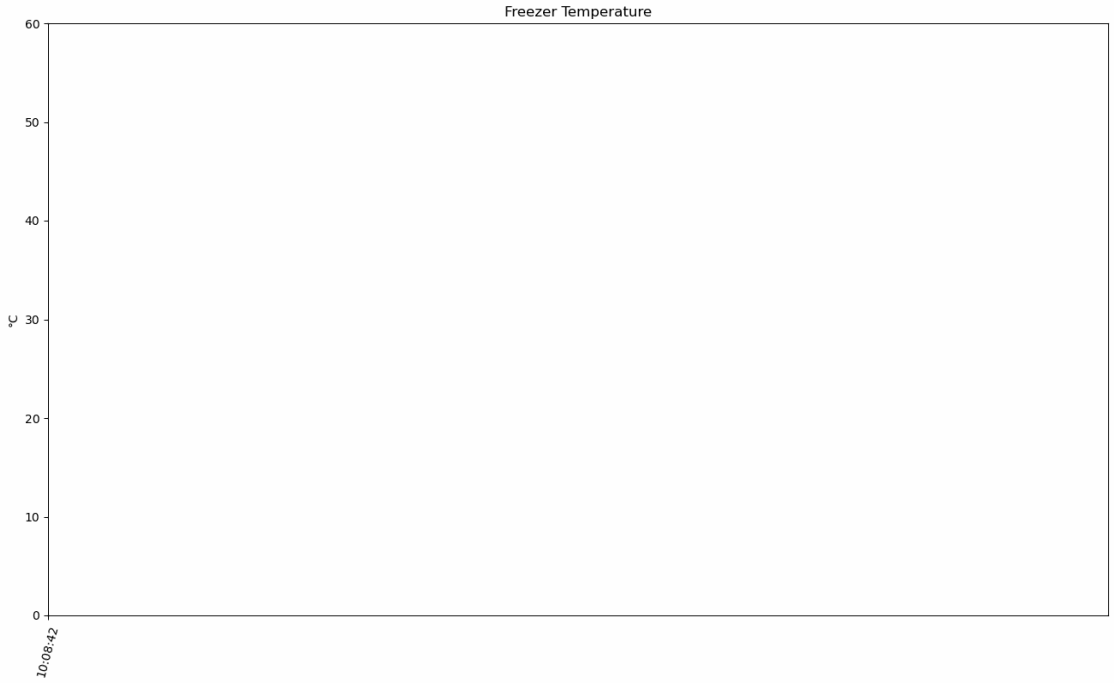
```
{
    ...
    "TYPE": "math_expression",
    "MATH_EXPRESSION": "x**2",
    "INTERVAL_START": 30,
    "INTERVAL_END": 40,
    "MIN_DELTA": 0.1,
    "MAX_DELTA": 0.2,
    ...
}
```

- `MATH_EXPRESSION`:
  - The `MATH_EXPRESSION`'s variable **must** be defined as `x`.
  - Any *Pythonic* expression is valid, so, for instance, if you declare it as `x**2` or `math.pow(x,2)` the generated function will be the same.
- `INTERVAL_START` and `INTERVAL_END`:
  - These parameters will works as the function domain, restricting the value that `x` can reach.
  - When the variable `x > INTERVAL_END`, the function will be evaluated, and then the variable will be reset to `x=INTERVAL_START`. So keep in mind that the real interval is `[INTERVAL_START, INTERVAL_END+MAX_DELTA)`.
- `MIN_DELTA` and `MAX_DELTA`:
  - It is possible to set both with the same value, in this case, it is expected that the curves are more similar between the "loops", and may be identical if `RETAIN_PROBABILITY = 0`.

7.3.1 Example 1 - Freezer Temperature

In the example below the `MATH_EXPRESION` = $2x²+1$, `INTERVAL_START = 0`, `INTERVAL_END = 5`, `MIN_DELTA = 0` and `MAX_DELTA = 0.5`, so it is expected that the generated values are between 1 and 61.5, and the curves should be slightly different.

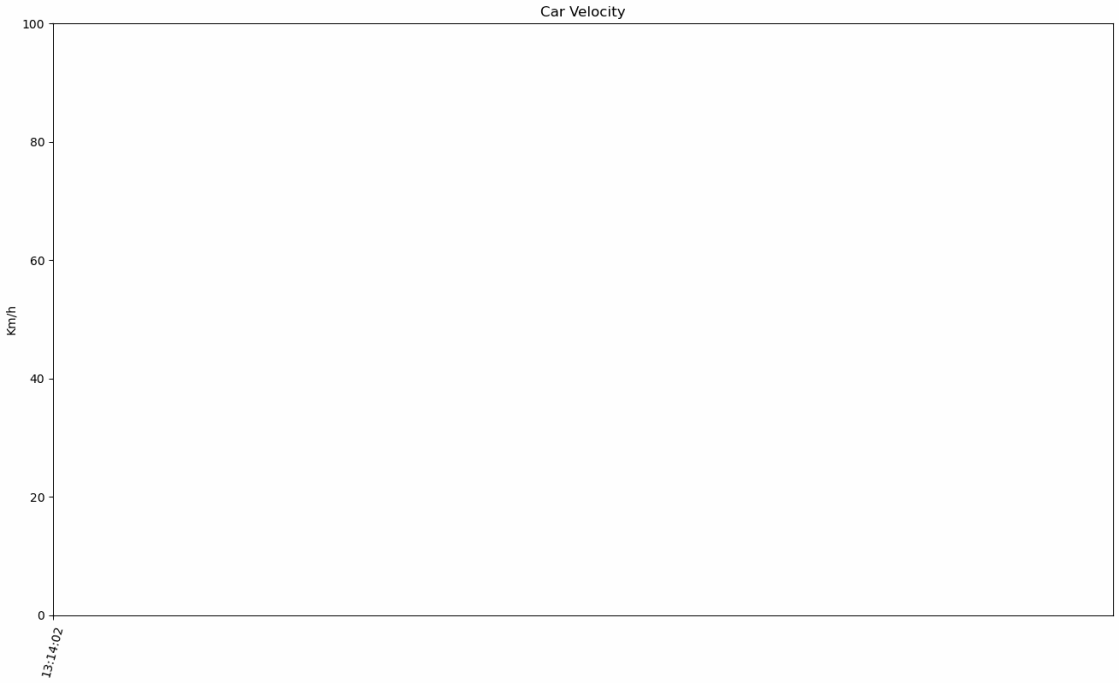
```
{
    "TYPE": "single",
    "PREFIX": "freezer",
    "TIME_INTERVAL": 6,
    "DATA": [
        {
            "NAME": "temperature",
            "TYPE": "math_expression",
            "RETAIN_PROBABILITY": 0.1,
            "MATH_EXPRESSION": "2*math.pow(x,2)+1",
            "INTERVAL_START": 0,
            "INTERVAL_END": 5,
            "MIN_DELTA": 0.5,
            "MAX_DELTA": 0.5
        }
    ]
}
```



7.3.2 Example 2 - Car Velocity

In the example below the `MATH_EXPRESION` = $\sqrt{75x}$, `INTERVAL_START = 0`, `INTERVAL_END = 100`, `MIN_DELTA = 10` and `MAX_DELTA = 10`, so it is expected that the generated values are between 0 and 87. As `REATIN_PROBABILITY = 0` and the `MIN_DELTA` and `MAX_DELTA` are identicals, the curves must be identicals.

```
{
    "TYPE": "single",
    "PREFIX": "car",
    "TIME_INTERVAL": 6,
    "DATA": [
        {
            "NAME": "velocity",
            "TYPE": "math_expression",
            "RETAIN_PROBABILITY": 0,
            "MATH_EXPRESSION": "(75*x)**(1/2)",
            "INTERVAL_START": 0,
            "INTERVAL_END": 100,
            "MIN_DELTA": 10,
            "MAX_DELTA": 10
        }
    ]
}
```


Car Velocity
100
80
60
40
20
0
Km/h
13:14:02

8. Dashboard

9. Relatórios e Email

10. Pré-requisitos e Instalação

Esta secção descreve as dependências necessárias e o processo de instalação do sistema de **Monitorização Industrial com IoT e ETL**, incluindo a configuração dos componentes MQTT, Node-RED, SQLite, React UI e automação com Python.

10.1 Pré-requisitos

Antes de executar o projeto, certifica-te de que tens as seguintes ferramentas instaladas:

10.1.1 Sistema Base

- **Python 3.10+** — para scripts de simulação, integração com API e envio de emails
- **Node.js 18+** — para execução da interface React e Node-RED
- **npm ou yarn** — gestor de pacotes JavaScript
- **SQLite3** — para armazenamento local dos dados processados
- **Mosquitto MQTT Broker** — para gerir a comunicação entre sensores (publishers) e consumidores (subscribers)
- **Git** — para controlo de versões e clonagem do repositório

10.1.2 Bibliotecas Python

Instala as bibliotecas necessárias:

```
pip install paho-mqtt requests smtplib sqlite3
```

10.1.3 Dependências Node.js

```
npm install @mui/material @emotion/react @emotion/styled axios
```

10.1.4 Node-RED

Instala o Node-RED globalmente:

```
npm install -g node-red
```

Depois, inicia com:

```
node-red
```

10.1.5 Comandos Git

Geeks git Commands

10.1.6 Gerar Documentação com MkDocs

```
pip install mkdocs mkdocs-material
mkdocs serve
```

11. Troubleshooting

Problema	Causa provável	Solução
Node-RED não liga	Porta ocupada	Muda a porta: `node-red -p 1881`
MQTT não conecta	Broker offline	Reinicia Mosquitto: `sudo systemctl restart mosquitto`
SQLite bloqueado	Conflito de acesso	Fecha outras conexões e tenta novamente
Emails não enviados	Autenticação falhou	Verifica SMTP e permissões “App Password”

12. Conclusão e Trabalhos Futuros

O **ProjetoISIv1** demonstrou a aplicação prática de conceitos de **Integração de Sistemas de Informação** com recurso a ferramentas de ETL, comunicação IoT e visualização web.

12.1 Resultados

- Automação da recolha de dados via sensores simulados;
- Processamento em tempo real com Node-RED;
- Armazenamento persistente em SQLite;
- Geração de relatórios automáticos;
- Visualização em dashboard web.

12.2 Trabalhos Futuros

- Migrar a base de dados para **SQL Server**;
- Usar docker para todo o sistema (Mosquitto, Node-RED, UI, DB);
- Adicionar predições com **IA/ML**;
- Criar alertas em tempo real (SMS/Telegram).