GESTÃO E CONTROLO DE UMA LINHA DE PRODUÇÃO

André Filipe Gomes Marques

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2020

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores

Candidato: André Filipe Gomes Marques, Nº 1170466, 1170466@isep.ipp.pt
Orientação científica: Carlos José Ribeiro Campos, crc@isep.ipp.pt

Empresa: Indmei

Orientador: Manuel Vieira Silva, mvieirasilva@sapo.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
12 de julho de 2020

Agradecimentos

Agradeço à minha família pelo seu apoio, especialmente aos meus pais que batalharam muito para me oferecer uma educação de qualidade. À minha irmã pelo apoio incondicional e pela compreensão da minha ausência enquanto eu me dedicava à realização deste trabalho.

Ao meu orientador Eng. Carlos Campos pelo apoio, disponibilidade, e colaboração prestada durante a realização deste projeto.

Aos meus professores, pelos conselhos e ensinamentos que me permitiram apresentar o melhor desempenho na formação profissional ao longo do curso.

Aos meus colegas e amigos que me acompanharam pelos incentivos e companheirismo demonstrado durante o curso.

Resumo

No trabalho efetuado foi proposto que se projetasse e implementasse uma aplicação que proporcionasse um melhor controlo das diferentes linhas de produção, sobretudo nas horas sem supervisão. Pretendia-se que o sistema encurtasse a necessidade de controlo de produção e supervisão, redução do tempo de paragem das linhas, e aumento do controlo de qualidade, com o objetivo de melhorar o rendimento da produção.

O projeto foi dividido em três partes principais, a primeira consistia na conversão do sistema previamente existente para funcionar numa máquina com sistema operativo baseado em Linux.

A segunda parte consistiu na realização de uma interface Web que disponibilizasse, em tempo real, os dados de saída do sistema de controlo de produção para os funcionários e para o supervisor.

A última fase correspondeu à implementação de uma interface gráfica web que permitisse visualizar os dados de forma mais simpática e organizada, que permitisse a sua melhor compreensão por parte dos responsáveis.

O resultado deste trabalho foi bastante positivo, a empresa passou a disponibilizar de uma ferramenta de controlo, muito mais amigável e fácil de interagir. Os funcionários passaram a ter acesso aos dados de produção em tempo real.

Palavras-Chave

JavaScript, Python, MySQL, Back-end, Front-end, JQUERY, JSON.

Abstract

In the work carried out it was proposed to design and implement a system that would provide a better control of the different production lines, it was intended that the system would shorten the need for production control and supervision, reduce the downtime of the lines, and increase the control quality, with the objective of improving production yield.

The project was divided into three main parts, the first consisted of converting the previously existing system to work on a machine with a Linux-based operating system.

The second part consisted of the realization of a web interface that made available, in real time, the output data of the production control system to the employees and the supervisor.

The last phase corresponded to the implementation of a graphical web interface that would allow data to be viewed in a more sympathetic and organized way, which would allow its better understanding by those responsible.

The result of this work was very positive, the company started to offer a control tool, much more friendly and easy to interact. Employees now have access to production data in real time.

Keywords

JavaScript, Python, MySQL, Back-end, Front-end, JQUERY, JSON

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

ASCII – American Standard Code for Information Interchange

DCL – Data Control Language

DDL – Data Definition Language

DML – Data Manipulation Language

TCL – Transaction Control Language

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

XML – eXtensible Markup Language

XHR – XMLHttpRequest

JSON – JavaScript Object Notation

USB – Universal Serial Bus

SQL – Structured Query Language

1. INTRODUÇÃO

Atualmente, para ter sucesso no mundo empresarial é necessário que toda a informação recolhida do ceio de uma empresa seja organizada e explorada ao máximo, no sentido tentar melhorar o rendimento e qualidade das suas linhas de produção. Para isso é preciso ter acesso à produção de todas as linhas e de forma mais individualiza, associada a cada posto de trabalho. Este trabalho foi realizado num contexto empresarial, na empresa Indmei, dedicada ao fabrico de meias com caraterísticas especificas e de alta qualidade.

Neste capítulo é feita a apresentação e contextualização do trabalho realizado no âmbito da unidade curricular Projeto/Estagio da Licenciatura em Engenharia Eletrotécnica e de Computadores no ano 2019/2020. Neste capítulo são apresentados os principais objetos do projeto e a calendarização do mesmo.

1.1. CONTEXTUALIZAÇÃO

A Indmei é uma empresa do ramo têxtil que se foca na produção e acabamento de meias, e, por isso, possui diversos postos de trabalho que necessitam de constante supervisão e controlo.

A Indmei apresentou uma proposta de projeto/estagio que incide, fundamentalmente, no desenvolvimento de um *software* que permite efetuar uma melhor visualização e análise da

informação recolhida nos postos de trabalho.Com o intuito de fazer a melhor gestão da qualidade de produção de seis postos de trabalho a empresa necessita de melhorar o sistema análise e visualização dos dados recolhidos pelo sistema de controlo já existente. Para tal, surgiu a necessidade de implementar uma aplicação Web que faça uma melhor gestão dos dados já armazenados na base de dados da empresa e futuros.

Perspetivava-se que durante a realização deste projeto, iráia surgir a oportunidade de trabalhar com tecnologias de desenvolvimento Web, tais como HTML, JavaScript, CSS, e até Python.

1.2. OBJETIVOS

O principal objetivo deste projeto é melhorar a qualidade da supervisão de seis linhas de produção e implementar um sistema web que permita apresentar a informação recolhida durante o processo de produção de forma amigável e organizada ao utilizador. Dada a complexidade do objetivo, sentiu-se a necessidade de o subdividir em múltiplas tarefas mais simples, tais como:

- Estudo do modo de funcionamento e configuração de base de dados MySQL
- A conversão do sistema de recolha de dados para o sistema operativo Linux
- Estudo de algumas ferramentas usadas para desenvolvimento Web (JavaScript, JQuery, Python, …)
- Desenvolvimento de uma interface gráfica para display de dados em tempo real
- Desenvolvimento da aplicação Web para analises estatísticas da informação, e ajuda na tomada de decisões

No final espera-se obter uma aplicação que facilite a análise e compreensão dos dados de produção recolhidos, de forma a identificar, o mais rápido possível, potenciais problemas das linhas de produção. Com tudo isto espera-se melhorar a qualidade e o rendimento dos seis postos de trabalho.

1.3. CALENDARIZAÇÃO

Para uma melhor gestão do tempo disponível para a realização foi concebida a calendarização da figura 1, ainda na fase de planeamento. Esta inclui tanto as semanas para aprendizagem de conceitos como para o desenvolvimento efetivo do trabalho. Tal como podemos ver no mês de março foi iniciada a primeira parte do projeto com o estudo da linguagem c++, assim como do sistema já implementado na Indmei, e gradualmente foi feita a conversão para Linux.

O mês de abril deu início à segunda fase o projeto, que corresponde ao estudo de algumas linguagens de programação tal como, python, html, css, e JavaScript. Neste mês foi também desenvolvida a interface gráfica destinada às linhas de produção e supervisores.

A terceira fase do trabalho começou no início do mês de maio e prolongou-se até ao final da terceira semana de junho. Esta correspondeu ao desenvolvimento da aplicação web destinada à apresentação e analise estatística dos dados de produção.

Id	Descricão	inicio	fim	Duração	março				abril					maio						junho					
					1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	6	1	2	3		
1	Estudo da linguagem de programaçao c++	01/mar	07/mar	1 semana																					
2	Estudo da linguagem SQL	01/jul	07/mar	1 semana																					
3	Estudo do código c++ do sistema exixtente	01/mar	14/mar	2 semanas																					
4	adaptaçao do codigo para rodar em linux	01/mar	28/mar	4 semanas																					
5	Estudo da linguagem python	29/mar	04/abr	1 semana																					
6	Estudo do sistema Flask	29/mar	11/abr	2 semanas																					
7	Estudo da linguagem JavaScript	12/abr	18/abr	1 semana																					
8	Estudo da ferramenta Jquery	12/abr	25/abr	2 semanas																					
9	Implementaçao da ineterface grafica para supervisores	29/mar	02/mai	5 semanas																					
10	Desemvolvimento da aplicaçao Web	03/mai	20/jun	7 semanas																					

Figura 1 Calendarização do projeto

1.4. ORGANIZAÇÃO DO RELATÓRIO

No primeiro capítulo é feita uma breve introdução ao projeto realizado no âmbito da unidade curricular de Projeto/Estágio do 3º ano da licenciatura em engenharia eletrotécnica e de computadores (LEEC), assim como a contextualização do problema a resolver, calendarização, e apresentados os principais objetivos propostos.

No capítulo seguinte, 2, analisa-se a arquitetura do sistema, onde são apresentadas as diferentes tecnologias utilizadas e são explicadas as interações entre estas. É também, feito o enquadramento destas no sistema em geral.

No capítulo 3, são expostos os principais algoritmos, etapas, e processos utilizados no desenvolvimento do projeto, assim como uma descrição detalhada dos mesmos.

No capítulo seguinte, 4, procede-se à mostra e analise dos resultados obtidos no trabalho.

Por fim, no último capítulo, 5, são retiradas as principais conclusões e apresentadas perspetivas que poderão ser abordadas para desenvolvimentos futuros.

2. ARQUITETURA DO SISTEMA

Neste capítulo é feita a apresentação e descrição das tecnologias utilizadas no desenvolvimento do projeto, assim como a explicação para a escolha das mesmas. Todas as tecnologias utilizadas são gratuitas (*open source*) mas apresentam muita robustez e, algumas, elevado nível de complexidade. Este aspeto permite reduzir os custos de desenvolvimento do projeto mantendo a sua qualidade.

A componente de recolha de dados já estava implementada e precisou apenas de uma conversão para o seu correto funcionamento em sistema operativo Linux. Esta recolha é feita através de um Arduíno que está constantemente a escrever para a porta serie do PC. Posteriormente o processamento e validação dos dados é realizado por um aplicativo escrito em c++.

Na figura seguinte está representado o diagrama do sistema implementado, onde podemos ver as diferentes tecnologias utilizadas e algumas das suas interações, este foi pensado ainda na fase de planeamento e mais tarde corroborado pela fase de analise dos resultados.

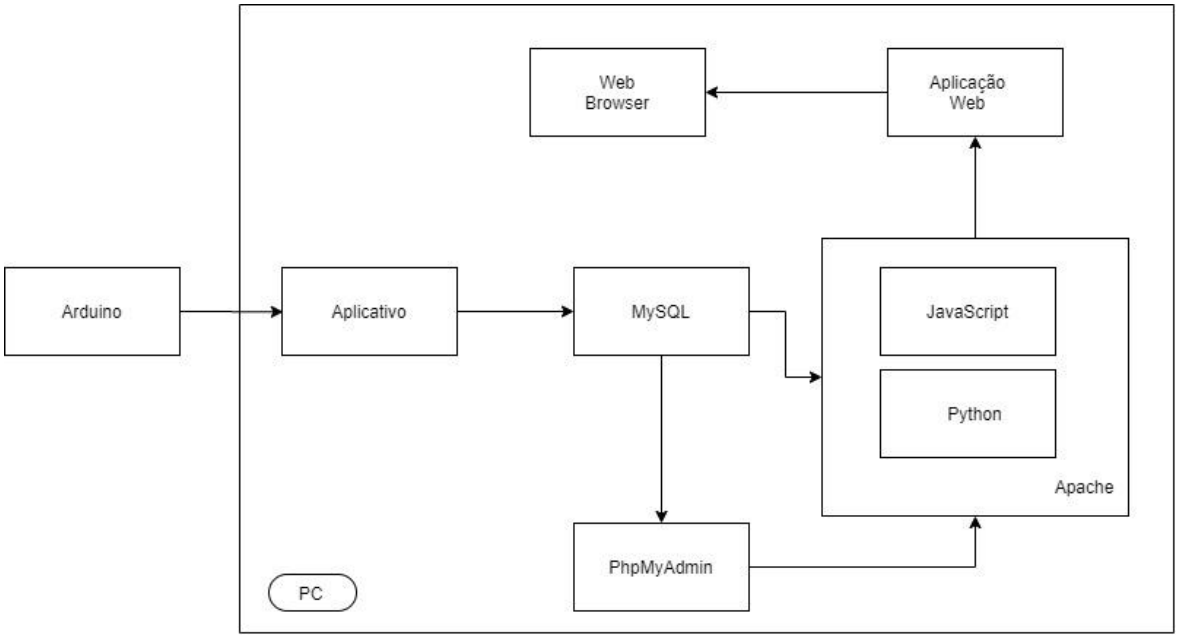

Figura 2 Esquema geral do sistema

Tal como podemos verificar na figura 2, a base de dados MySQL encontrasse no centro do diagrama, tornando evidente a sua importância no sistema. Numa primeira fase os dados recolhidos pelo Arduíno são enviados para a porta USB do PC e posteriormente são processados e validados pelo aplicativo, que é responsável por inserir as entradas de informação na base de dados. Posteriormente a aplicação web comunica com o MySQL para obter os dados relativos à produção, esta comunicação realizasse através de uma *application programming interface* (API) do Mysql para a linguagem Python.

2.1. VERSÃO ANTERIOR

Como referido anteriormente, o sistema previamente instalado na Indmei estava mais focado na captura e armazenamento dos dados. Este sistema recorre a um arduíno para fazer a recolha da informação. Cada posto de trabalho é constituído por um *switch* que é acionado e, posteriormente, lido pelo arduíno que manda a informação para o PC através de uma ligação USB. No computador existe um aplicativo escrito em c++ que faz a leitura da porta USB e valida os dados. Se os dados estiverem corretos o aplicativo armazena-os numa base de dados.

A versão antiga apresentava também uma interface gráfica escrita em PHP e HTML que permitia a visualização dos dados de produção do dia.

Contudo, esta versão apresentava alguns problemas, tais como:

1. Apenas era possível visualizar os dados do próprio dia, logo não era possível efetuar qualquer tipo de analise estatística.
2. Os dados estavam organizados em blocos de quatro horas.
3. O sistema estava implementado em Windows.

Na realização deste projeto, foi necessário fazer uma conversão do aplicativo para efetuar a leitura da porta série em sistemas UNIX. A interface gráfica foi totalmente descartada e procedeu-se ao desenvolvimento de uma nova, tal como irá ser visto no capítulo 3.

2.2. MySQL

O MySQL é um sistema de gestão de base de dados relacional (RDBMS) que adota maioritariamente um modelo de cliente-servidor, em que o cliente gera os pedidos e o servidor processa esses pedidos e constrói as respetivas respostas, e a Structured Query Language (SQL). Os comandos SQL dividem-se quatro grandes grupos, Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL), e Transaction Control Language (TCL). Neste projeto os grupos mais utilizados são o DDL, uma vez que esta lida com a estrutura e descrição dos dados dentro da base de dados, e o DML que, faz a manipulação dos dados, este inclui os comandos mais conhecidos, tais como SELECT, INSERT, UPDATE, etc.. [1]

2.3. Python

Python é uma das linguagens de programação mais interessantes e usadas a nível mundial. De acordo com as tabelas de classificação da TIOBE, python é a terceira linguagem mais popular, ficando apenas atrás de c e java. As principais vantagens desta linguagem em comparação a outras incluem:

- É uma linguagem interpretada, o que significa que não precisa de ser compilada para instruções máquina de baixo nível
- O Python Package Index (PyPI) contêm vários módulos de terceiros que a tornam capaz de interagir com a maioria das outras linguagens e plataformas

- Extensas librarias de suporte: oferece uma grande biblioteca padrão que inclui áreas como protocolos de internet, operações de string, ferramentas de desenvolvimento Web e interfaces de sistema operacional. Muitas tarefas de alto uso já foram escritas na biblioteca padrão o que reduz significativamente o comprimento do código a ser escrito. [2]

Outra vantagem do Python é a sua grande flexibilidade de programação Web, graças a todas as estruturas (frameworks) que tornam o desenvolvimento Web rápido e fácil, como por exemplo Django e Flask. Uma framework é um pacote que contém pacotes e módulos que fornecem cetras funcionalidades que podem ser alteradas de forma seletiva para criar um software específico de aplicação. [3]

2.3.1. FLASK

Uma Web Application Framework é uma coleção de módulos e bibliotecass que possibilitam aos desenvolvedores escrever aplicações com maior nível de abstração em relação a detalhes de baixo nível, tal como protocolos e gestão de threads.

Flask é uma Web Framework desenvolvida por Armin Roonacher, que lidera uma equipa internacional de entusiastas Python chamados Poocco. Esta é baseada no Wekzeg web server gateway interface (WSGI) kit e no mecanismo de modelo Jinja2.

```
<html>
    <head>
        <title>{{ title }}</title>
    </head>
    <body>
        <h1>Hello {{ username }}</h1>
    </body>
</html>
```

Figura 3 Modelo Jinja2 [4]

O Jinja2 é um mecanismo de modelo para Python usado para criar HyperText Markup Language (HTML), eXtensible Markup Language (XML), ou outros formatos de marcação, que são retornados ao usuário por meio de uma resposta HyperText Transfer Protocol (HTTP). Um modelo contém variáveis que são substituídas pelos valores passados quando

o modelo é renderizado, tal como mostra a figura 3. As variáveis são uteis com dados dinâmicos. [4]

Para que a FlaskApp funcione de forma correta é preciso que os ficheiros sigam a organização apresentada na figura 4.

```
/app
   -/app.py
    /templates
       -/index.html
       -/404.html
```

Figura 4 Estrutura FlaskApp [4]

2.4. JAVASCRIPT

JavaScript é uma linguagem de script que roda do lado do cliente, ou seja, ao invés de rodar remotamente em servidores na internet, o JavaScript tem como característica principal correr os programas localmente, sendo estes executados pelo navegador web. Assim sendo, o JavaScript fornece às páginas web a possibilidade de programação, transformação e processamento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HMTL e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas, o que torna a experiência do utilizador mais interessante e interativa.

JavaScript é uma linguagem de thread única, o que significa que possui uma única Call Stack e, portanto, apenas tem a capacidade de fazer uma coisa de cada vez.

O facto desta linguagem receber e enviar pedidos através da rede para servidores remotos, sem ter a necessidade de recarregar a totalidade da página conforme as entradas do usuário foi um elemento decisivo quanto à sua escolha para este projeto.[5]

2.4.1. JQUERY

JQuery é uma biblioteca JavaScript rápida, pequena e rica em recursos que simplifica o processo de passagem e manipulação de documentos HTML, eventos, animação e AJAX. Tem uma API fácil de usar que funciona em quase todos os navegadores.[6]

O principal motivo da utilização desta tecnologia no projeto é a simplicidade de execução dos pedidos AJAX (asynchronous JavaScript and XML). Estes pedidos possuem grande

importância porque permitem a troca de dados entre o front-end e o back-end sem usar alguma variedade de linguagem no lado do servidor para injetar os dados necessário no JavaScript e sem ter de navegar para uma página nova sempre que for preciso atualizar o conteúdo.[7]

```
$.ajax({
        url: "/getconf",

        success: function (result) {
              console.log(result.postos)
              n_postos = result.postos
              n_turnos = result.turnos
              $('#n_postos').append(n_postos)
              $('#n_turnos').append(n_turnos)
        } })
```

No estrato de código acima podemos ver a estrutura básica de um pedido AJAX, neste caso a propriedade "url" contém o destino para onde o pedido (request) é enviado. A propriedade "success" permite chamar uma função caso o pedido tenha sido bem-sucedido. Para esta função são passados três argumentos: o conteúdo retornado pelo servidor; o objeto jqXHR (XMLHttpRequest); e uma string que descreve o estado do pedido.[8]

2.4.2. JSON (JAVASCRIPT OBJECT NOTATION)

JSON é uma formatação leve de troca de dados, é fácil de ler e escrever para o programador, e fácil de interpretar e gerar para máquinas. Está baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro – 1999. JSON é em formato de texto e completamente independente da linguagem, pois usa convenções que são idênticas às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Python, Perl e muitas outras. Estas propriedades fazem com que o JSON seja um formato ideal para troca de dados.[9]

O formato do conteúdo pode mudar de acordo com a estrutura dos dados. As duas estruturas mais usadas são: [9]

- Um objeto, que é um conjunto desordenado de pares nome/valor. O objeto começa com uma chave de abertura ("{") e termina com uma chave de fechamento ("}"), cada nome é seguido por ":" e os pares nome/valor são separados por virgula, tal como pode ser observado na figura 5. Este tipo de estrutura foi o mais utilizado no desenvolvimento do projeto.

- Um *array*, sequencia de dados listados de forma ordenada numa lista. Não existe o mapeamento nome/valor.

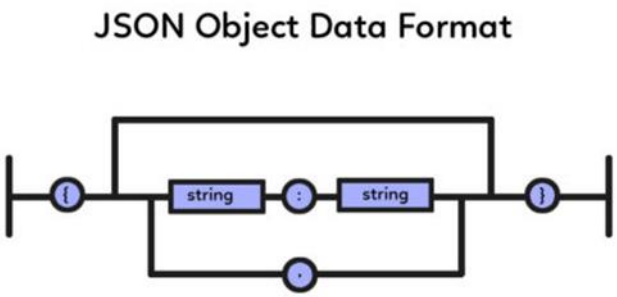

Figura 5 Estrutura de um objeto JSON

3. DESENVOLVIMENTO DO SISTEMA

Como referido anteriormente, a realização deste projeto foi escalonada em três fases distintas. Numa primeira fase procedeu-se à conversão do código do sistema de aquisição de dados para poder ser executado numa máquina com sistema operativo Linux. Na segunda fase foi desenvolvida uma aplicação gráfica para visualização dos dados de produção, número de peças produzidas e tempo de paragem, em tempo real. Por último, foi desenvolvida uma aplicação Web para efetuar cálculos estatísticos e permitir uma visualização mais amigável dos dados adquiridos ao longo do tempo.

3.1. BASE DE DADOS

A base de dados é um dos elementos mais importantes do sistema, pois permite guardar grandes quantidades de dados que são depois acessados pela aplicação desenvolvida. Neste projeto foi usado um total de nove tabelas, no entanto seis apresentam a mesma estrutura, uma vez que apenas armazenam os dados recolhidos pelos seis postos de trabalho. As restantes tabelas armazenam os valores de configuração e controlo das linhas de produção.

Cada tabela deve ser constituída pro um certo número de Colunas, e cada coluna tem o seu próprio tipo de dados associado. Os tipos de dados numéricos mais usados no projeto estão apresentados na seguinte tabela:

Tabela 1 Tipos de dados da base de dados

Type	Memoria (bytes)	Valor mínimo com sinal	Valor máximo com sinal	Valor mínimo sem sinal	Valor máximo sem sinal
tinyInt	1	-128	127	0	255
SmallInt	2	-32768	32767	0	65535
MediumInt	3	-8388608	8388607	0	16777215
INT	4	-2147483648	2147483647	0	4294967295

Na tabela 2, está representada a estrutura seguida pelas seis tabelas que armazenam os registos da produção dos postos de trabalho. Cada uma destas tabelas é constituída por nove colunas de permitem a correta identificação dos dados.

Tabela 2 Estrutura da tabela "DataX"

Data1	
Coluna	Tipo da Variável
Id_turno	Tinyint (3)
Id_bloco	Tinyint (3)
Nr_peças período	Smallint (5)
Total_peças_bloco	Smallint (5)
Tempo_paragem_total	Smallint (5)
Hora _inico_periodo	Time
Hora_primeria_peca	time
Hora_ultima_peca	time
Data	date

Descrição das colunas:

1. Id_turno – identificação do turno, serve para identificar o registo
2. Nr_peças_periodo – número de peças produzidas por período, a duração do período pode ser definida n tabela de controlo, neste trabalho é de 60 minutos.
3. Total_pecas_bloco – número de peças produzidas por turno
4. Tempo_paragem_total – representa o tempo total (em segundos) por período
5. Hora_inicio_periodo – contém a hora de início do período, serve para identificar o registo

6. Hora_ultima_peca - regista a hora de produção da última peça, serve para efetuar o cálculo do tempo de paragem da produção entre dois períodos diferentes
7. Data – contém a data do registo e serva para o identificar

A tabela “Def_turnos”, tabela 3, guarda as informações relativas ao horário dos turnos de trabalho e atribui um número de identificação (id) a cada turno inserido. Cada turno contém uma hora de início e hora de fim.

Tabela 3 Estrutura da tabela "Def_turnos"

Def_Turnos	
Coluna	**Tipo de variável**
id_turno	Tinyint (3)
Hora_inicio	Time
Hora_fim	time

Na tabela 4 está representada a constituição da tabela de controlo. O controlo serve para: identificar o tempo de cada período; a duração de cada bloco; o tempo de paragem de referência por período e por turno; e a referência do número de peças produzidas por período.

Tabela 4 Estrutura tabela "Controlo"

Controlo	
coluna	**Tipo de variável**
Data_info	Varchar (25)
duracao_periodo	Smallint (5)
Duraçao_bloco	Tinyint (3)
Ref_cont_tempo_paragem	Tinyint (3)
Ref_tempo_paragem	Tinyint (3
Ref_producao_periodo	Smallint (5)

Por fim, os dados de utilizador estão armazenados na tabela “users”, que segue a estrutura da tabela 5. A coluna “id” contém a informação do número de identificação de cada utilizador, a coluna “username” guarda o nome de utilizador, e a coluna password guarda a codificação sha256 da palavra passe.

Tabela 5 Estrutrura tabela "Users"

Users	
Coluna	**Tipo de variável**
id	**Tinyint (3)**
username	**Varchar (30)**
password	**Varchar (100)**

3.2. AQUISIÇÃO DE DADOS

A aquisição dos dados é feita através de um Arduíno que comunica com o computador através de uma ligação USB. Sempre que um posto produz uma nova peça o Arduíno envia para o PC um número inteiro correspondente ao número do posto, por exemplo, se o posto número 6 produzir uma peça é enviado o número 6 pela conexão USB. Posteriormente o aplicativo escrito em c++ deteta que existe informação nova na porta série e procede à sua interpretação e validação, caso a informação esteja correta os dados são atualizados e inseridos numa base de dados, tal como podemos verificar no diagrama da figura 6.

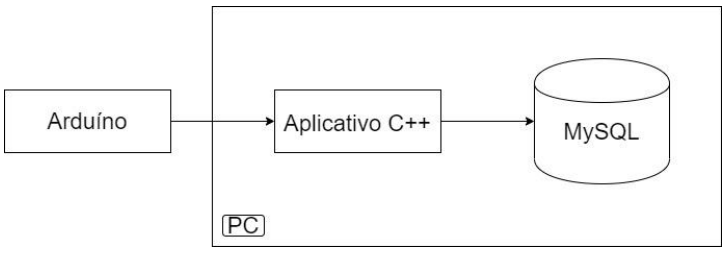

Figura 6 Estrutura de um objeto JSON

Nesta fase do projeto é necessário converter o código do aplicativo c++ para que este rode num sistema Linux. Os principais problemas encontrados nesta fase foram: o código ter sido

construído com base na API do Windows (WinAPI), e o facto de a comunicação série se processar de forma completamente diferente no sistema Linux. Por estas razoes foi necessário proceder ao estudo da WinAPI e da comunicação série em sistema operativo Linux.

3.2.1. COMUNICAÇÃO. SÉRIE

A comunicação série, em Linux, realiza-se através de um sistema semelhante ao de leitura e escrita para um ficheiro, isto é, as portas serie são tratados como se fossem um ficheiro e apresentam nomes como "/dev/ttys0", "/dev/ttys1". Neste caso, a comunicação série com o Arduíno acontece sempre na porta "/dev/ttyUSB0". Para efetuar a comunicação entre o aplicativo e as portas série foi usada a tecnologia termios, o que corresponde a uma API para entradas e saídas de terminal para sistemas Unix. A estrutura do programa para realizar as operações de entradas e saídas serie com a ajuda de termios é a seguinte:[10]

i. Abrir o dispositivo série através da chamada standard do sistema Unix "open(2)".

ii. Configuração dos parâmetros e propriedades da comunicação serie com a ajuda de funções e estruturas de dados especificas de termios.

iii. Uso das funções standard do sistema Unix "read(2)" ou "write(2)", para ler e escrever para a porta serie.

iv. Fechar o dispositivo série através da chamada standard "close(2)"

No processo de desenvolvimento do código de leitura da porta serie começou-se por criar uma classe denomina "serial" que recebe por argumento uma string com o nome da porta serie que se pretende utilizar, neste caso "/dev/ttyUSB0". De seguida, procede-se à abertura do ficheiro recorrendo à chamada standard "open(2)", tal como mostrado no extrato de código abaixo:[10]

```
int serial_port = open("/dev/ttyUSB0", O_RDWR);
```

Na figura 7 está representado o fluxograma da função "ConnectComPort". Nesta função começa-se por criar a variável "com1" do tipo "serial" e logo de seguida executa-se o método "com1.com_conection()" que é responsável pela abertura do ficheiro da porta serie e que

retorna o correspondente descritor de arquivo. Posteriormente é chamado o método "com1.conf_conection" que recebe o descritor por argumento. Neste método procede-se à configuração dos parâmetros da comunicação serie utilizando as estruturas da API termios, como apresentado no extrato de código abaixo.

```
tty.c_cflag &= ~PARENB;
tty.c_cflag &= ~CSTOPB;
tty.c_cflag |= CS8;
tty.c_cflag |= CREAD | CLOCAL;

tty.c_lflag &= ~ICANON;
tty.c_lflag &= ~ECHO;
tty.c_lflag &= ~ECHOE;
tty.c_lflag &= ~ECHONL;
tty.c_lflag &= ~ISIG;

tty.c_iflag &= ~(IXON | IXOFF | IXANY);
tty.c_iflag &=
~(IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL);

tty.c_oflag &= ~OPOST;
tty.c_oflag &= ~ONLCR;

tty.c_cc[VMIN] = 18;
tty.c_cc[VTIME] =10;
```

Neste fragmento de código é possível observar os cinco grandes campos da estrutura termios, o "tcflag_t c_iflag" que é responsável pelos modos de entrada; o "tcflag_t c_oflag" que atua ao nível dos modos de saída; o "tcflag_t c_cflag" que atua sobre os modos de controlo; o "tcflag_t c_lflag" responsável pela configuração dos modos locais; e o "cc_t c_cc[NCCS]" encarrega dos carateres especiais.

As configurações com maior importância são as que incluem os seguintes parâmetros:

- "~PARENB" – desabilita o bit de paridade
- "~CSTOPB" – implementa um stop bit na comunicação
- "CS8" – define o tamanho da máscara para 8 bits
- "~ICANON" – ativa funcionamento no modo não canônico
- "VMIN" – define o número mínimo de caracteres lidos no modo não canónico, neste caso são 18
- "VTIME" -define o tempo máximo em decimas de segundo da leitura no modo não canónico, neste caso 1 segundo (10 decimas de segundo). O que significa que ao fim de um segundo é retornado o conteúdo lido na porta serie independentemente do número de caracteres ter atingido o valor definido em "VMIN" ou não.

A definição da baud rate de leitura é conseguida através da chamada de “cfsetispeed()”, esta deve ser igual à que foi definida nas configurações do arduíno, neste caso 9600 bits por segundo.

Como se pode verificar no fluxograma da Figura 7, após serem definidas as configurações da comunicação série o programa entra num ciclo infinito. Neste ciclo o programa está constantemente a ler dados da porta serie por meio do método “com1.read_data”, e sempre que encontra dados novos estes são processados e validados pelo restante sistema. O método “com1.read_data” usa a chamada standard “read(2)” que escreve os dados lidos numa variável global chamada “read_buffer” e retorna o número de bytes lidos. Se o valor retornado for inferior a zero significa que houve um erro de leitura e os dados são rejeitados. [11]

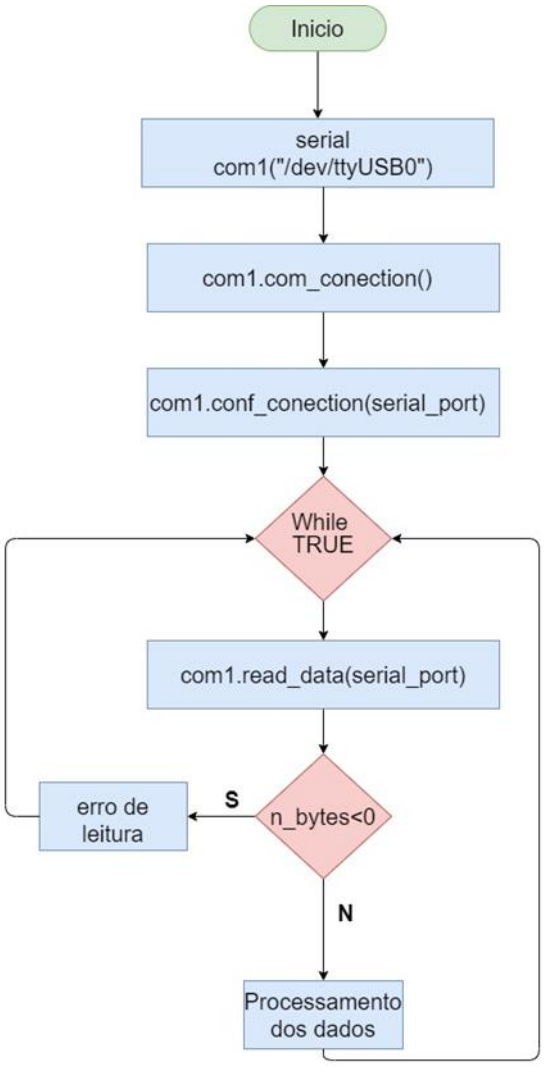

Figura 7 Fluxograma da função ConnectComPort ()

3.3. INTERFACE GRÁFICA PARA SUPERVISIONAMENTO

Nesta fase, a Indmei propôs a elaboração de uma interface gráfica que permitisse ver os dados de produção em tempo real. Foi sugerido que implementasse uma página Web que mostrasse o número de peças e o tempo total de paragem por hora para o turno atual.

Como referido anteriormente, para o desenvolvimento em back-end foi escolhida a linguagem de programação Python, para o front-end optou-se por utilizar JavaScript.

Na figura 8, observa-se das diferentes interações entra as tecnologias usadas. Os ficheiros escritos em Python apresentam três tarefas de extrema importância: a gestão e manutenção do servidor, feita com recurso à tecnologia referida anteriormente Flask; a leitura e processamento dos dados armazenados nas bases de dados; e a passagem do conteúdo para as tecnologias de front-end. O JavaScript é utilizado para atribuir características dinâmicas e atualizar a informação da página html.

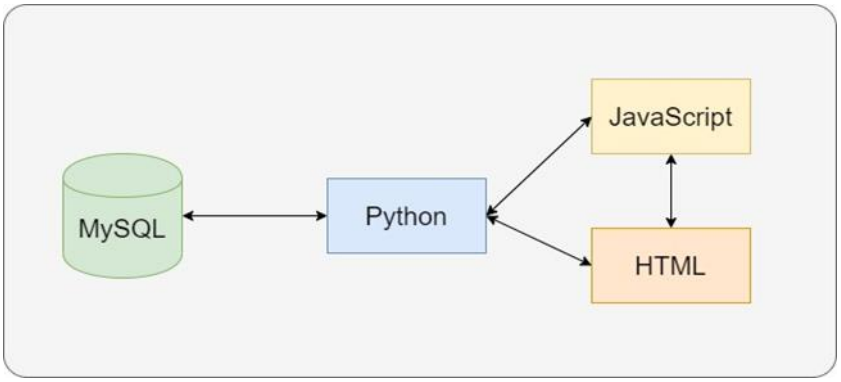

Figura 8 Diagrama de interações da interface gráfica

A microestrutura flask necessita que os ficheiros sigam a estrutura apresentada na figura 9, o programa principal "interface.py" tem de estar dentro do diretório principal "/interface"; os ficheiros html modelo devem ser colocados numa pasta com o nome "templates"; e os ficheiros do tipo js (javaScript) e css dentro do diretório "static".

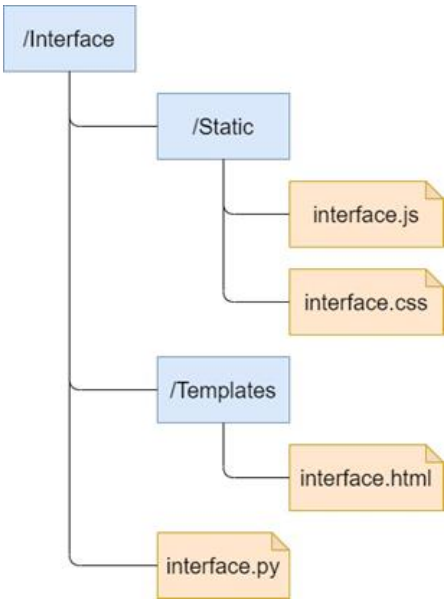

Figura 9 Mapa de diretorios da aplicação

3.3.1. DESENVOLVIMENTO BACK-END

No desenvolvimento desta aplicação o servidor apenas precisou de dois "routes", o primeiro para proceder à renderização do modelo interface.html e o último para obter os dados que são visualizados na página html, tal como mostra o código abaixo.

```
app = Flask(__name__)
@app.route("/")
@app.route("/home")
def home():
   return render_template('interface.html')

@app.route("/getData")
def getData():
   posto = request.args.get('posto')
   mycursor = mysql.connection.cursor()
   turnos = infogeral()
   c_turno = get_today_turno(turnos)
   pecas, tempo, horas = get_today_data(c_turno,
posto)
   return jsonify({'turno':c_turno['turno'],
'pecas':pecas, 'tempo':tempo,
'hora':str(c_turno['inicio']), 'h':horas})

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000,
debug=True)
```

A função “getData()” tem com principais tarefas recolher os dados de produção do turno atual, para o posto que é passado por argumento; e organizar esses dados por hora. O primeiro passo para que tal aconteça é estabelecer uma conexão à base de dados, o que requer o módulo flask_mysqldb. No extrato de código abaixo pode-se verificar como são definidas as configurações dessa conexão.

```
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'indmei'
app.config['MYSQL_PASSWORD'] = '*********'
app.config['MYSQL_DB'] = 'indmei'

mysql = MySQL(app)
```

Como mostra a figura 10, após a ligação ser estabelecida, é executada a função “infogeral()” que recolhe a informação relativa a cada turno da tabela “def_turnos”, recorrendo à seguinte query:

```
Select * from def_turno
```

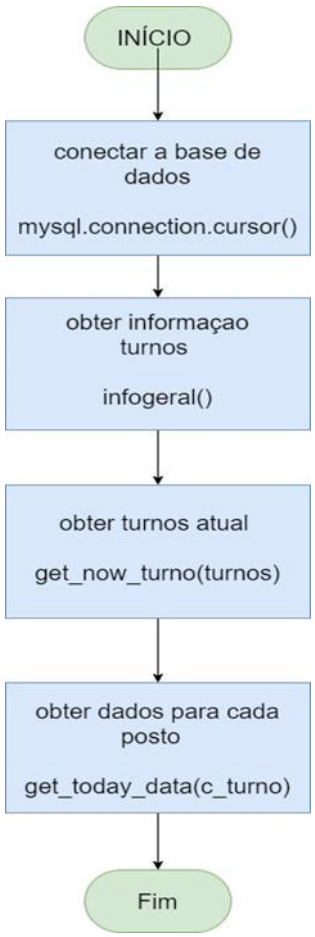

Figura 10 Fluxograma da função "getData()"

Consequentemente essa informação é passada para a função "get_now_turno()" que com base na hora do sistema determina o turno instantâneo. De seguida, procede-se à chamada de "get_today_data()" que seleciona a informação para o turno pretendido e para o dia atual. Por fim os dados são retornados num objeto JSON, seguindo a seguinte estrutura:

```
 {
'turno':c_turno['turno'],
'pecas': pecas,
'tempo': tempo,
}
```

- **"turno"** – contém os detalhes do turno selecionado (hora de início, hora do fim, e nome do turno)
- **"pecas"** – contém um vetor com o número de peças para cada hora do turno
- **"tempo"** – contém um vetor com o tempo total de paragem da linha por cada hora do turno

3.3.2. DESENVOLVIMENTO FRONT-END

Para efetuar a visualização dos dados em tempo real, a Indmei propôs que a página html fosse composta por seis tabelas, e cada tabela devia apresentar pelo menos oito linhas (uma linha para cada hora do turno) e duas colunas (a primeira para o numero de peças produzidas nessa hora, e a segunda para o tempo total de paragem nessa hora).

Na figura 11 está representado o fluxograma do ficheiro "interface.js", como podemos observar assim que a página html carrega é executado um ciclo "for" com a duração igual ao número de postos existentes, neste caso seis. Durante cada ciclo são executadas duas funções, "drawtables()" e "updateData()". Na primeira procede-se ao desenho da tabela para o posto correspondente, e na segunda é efetuado um pedido AJAX que obtém os dados a partir do processo definido em back-end "/getData" e atualiza as tabelas correspondentes.

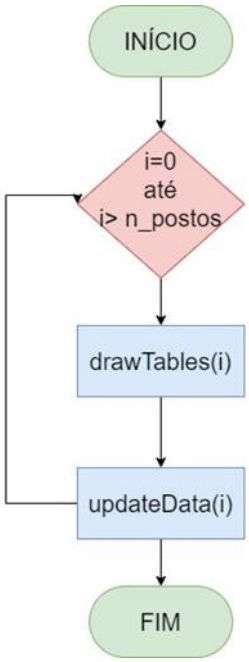

Figura 11 Fluxograma do ficheiro "interface.js"

```
$.ajax({
        url: "/getData",
        data: {
              'posto': posto,
        },
        success: function (result) {
                  …
              $('#Tbody'+posto).empty();
              $('#Tbody'+posto).append(rowDados)
        }
  })
```

Pela interpretação do extrato de código apresentado acima, percebe-se que para efetuar o pedido AJAX foi utilizada a tecnologia JQuery (mencionada no capítulo 2) para facilitar a troca de dados.

3.4. APLICAÇÃO WEB PARA ANÁLISE DE DADOS

Nesta última fase do projeto foi desenvolvida uma interface que permite a visualização de grandes quantidades de dados de forma mais organizada e percetível.

3.4.1. SISTEMA DE AUTENTICAÇÃO

O método de autenticação por JSON Web Token (JWT) foi o escolhido para fazer a gestão do sistema de login. O JWT é um pedaço de texto com algumas informações codificadas que permitem identificar o usuário para o qual este foi gerado. Para o correto funcionamento é necessário efetuar os seguintes passos:

i. O utilizador deve introduzir o seu nome de utilizador e palavra passe
ii. Verificar que estes estão corretos no interior da aplicação Flask
iii. Gerar o JWT com a identificação do utilizador codificada
iv. Enviar o JWT para o utilizador
v. Sempre que o utilizador efetuar um *request* para a aplicação deve enviar o JWT gerado anteriormente. Assim é possível verificar que o utilizador que enviou o JWT é o mesmo para o qual este foi gerado.

O último passo é muito importante, pois quando o servidor recebe um JWT valido pode verificar a informação codificada no seu interior e validar o utilizador.

Para implementar este sistema é preciso importar o módulo JWT para o interior do ficheiro onde a aplicação está definida. Posteriormente é necessário definir uma "secret_key" que é usada para assinar o token e garantir que este foi gerado pela aplicação. Por norma a "secret_key" deve ter 256 bits de tamanho, neste projeto, de cada vez que a aplicação é iniciada, gera-se uma nova "secret_key" aleatória, usando a seguinte instrução:

```
app.config['SECRET_KEY'] =
''.join(choice(string.ascii_uppercase +
string.ascii_lowercase  + string.digits) for _ in
range(256))
```

Após a verificação dos dados inseridos pelo utilizador (username e password) cria-se o objeto JWT usando a função "jwt.encode()", onde são passados por argumento a identificação do usuário, o tempo de duração do token (neste projeto é de 30 minutos), e a "secret_key", tal como no extrato de código seguinte:

```
token = jwt.encode({'user': user, 'exp':
datetime.utcnow()+timedelta(minutes=20)},app.conf
ig['SECRET_KEY'])
```

A verificação do token é feita numa função à parte que deve ser chamada em todos os "routes" que se quer proteger. Nessa função o token é descodificado usando a chamada "jwt.decode()" que recebe por argumentos o próprio token e a "secret_key" e retorna o "payload", tal como mostra o extrato de código abaixo. O "payload" é o nome técnico para a informação que está contida no token (id do usuário).

```
@app.route('/dashboard')
@token_required
def dashboard():
  return render_template('dashboard.html')
```

Os dados de utilizador estão guardados numa base de dados chamada "Users" que contém apenas a tabela "usersLog" com duas colunas (nome de utilizador e password). As passwords são guardadas com recurso à encriptação sha256. No código da aplicação a comparação da palavra passe introduzida com a que esta guardada na base de dados é feita da seguinte forma:

```
sha256_crypt.verify(inserted_password, password)
```

3.4.2. ALGORITMOS DE RECOLHA DE DADOS

No desenvolvimento do programa existem dois algoritmos de recolha de dados que são crucias para o funcionamento da aplicação. O primeiro faz o retorno dos dados de produção para o dia e postos passados por argumento; e o segundo retorna a informação relativa à produção por hora e para o dia atual.

Na figura 12 está representado o fluxograma da função "get_data_day()". Esta carrega grande importância pois é a base que suporta quase todas as funcionalidades da aplicação. Este algoritmo recebe por argumento o dia e o posto para os quais deve recolher a informação presente na base de dados. O primeiro passo a efetuar é iniciar a conexão à *database* através do modulo "mysql.connection.cursor()", tal como já foi visto no capítulo 3.2.1. De seguida o programa entra num ciclo que permite recolher os dados para todos os turnos, o comando SQL utilizado foi o seguinte:

```
Select total_pecas_bloco, tempo_paragem_total
from data%s where id_turno=%s and data='%s'" %
(posto, turno, dia)
```

Após a execução deste comando verifica-se se foi retornado algum conteúdo, no caso de a resposta não conter informação as variáveis "pecas" e "tempo" são igualadas a zero; caso a resposta apresente conteúdo é efetuado um novo ciclo que faz a soma das peças produzidas e do tempo de paragem e guarda os valore nas variáveis "pecas" e "tempo". Por último, estas variáveis são adicionadas às listas "total_pecas" e "total_tempo" utilizando a seguinte operação:

```
total_pecas.append(int(pecas))
total_tempo.append(int(tempo))
```

No final do algoritmo estas listas devem apresentar a informação para cada um dos turnos.

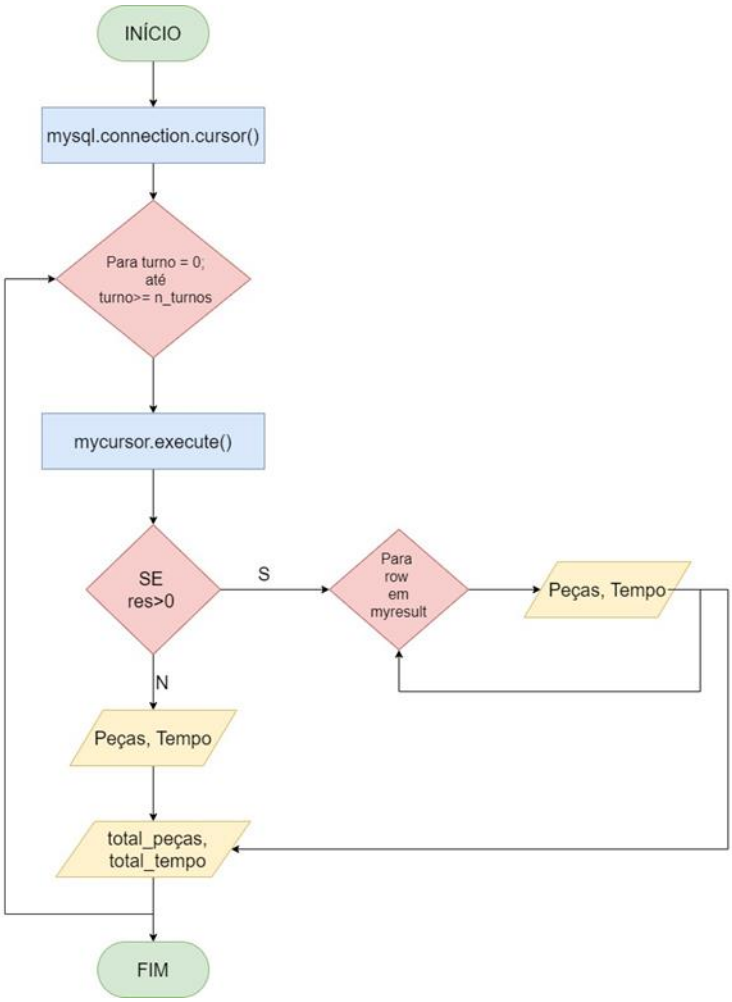

Figura 12 Fluxograma do algoritmo "get_data_day()"

O segundo algoritmo, representado pela figura 13, recebe apenas o número do posto por argumento. Neste algoritmo é executado um ciclo “for” que percorre todas as 24 horas do dia e preenche uma lista com os dados de produção para cada hora. O comando SQL utilizado para recolher a informação da base de dados foi:

```
Select nr_pecas_periodo, tempo_paragem_total from
data%s where hora_inicio_periodo = '%s' and
data='%s'" % (posto, i, today)
```

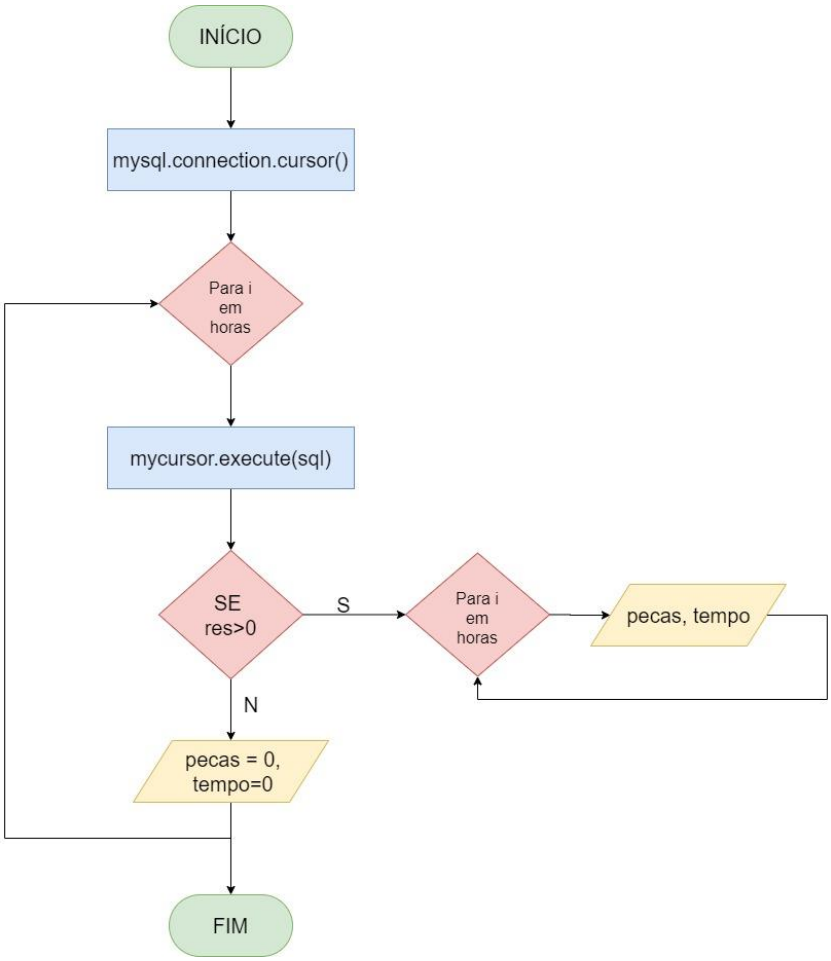

Figura 13 Fluxograma do algoritmo "get_data_hour()"

3.5. VISUALIZAÇÃO DOS DADOS

Para facilitar a observação da informação pela parte do cliente foi usado o JavaScript com o intuito de tornar as páginas mais dinâmicas e interativas. A informação é apresentada maioritariamente em tabelas ou gráficos. A atualização do conteúdo é feita através de pedidos AJAX, tal como foi explicado e exemplificado no capítulo 3.2.2.. Para a apresentação em gráficos foi usada a libraria “Chart.js”.

A API “Chart.js” permite desenhar os diferentes tipos de gráficos dentro de um elemento do tipo canvas em HTML5. Neste projeto foram utilizados dois tipos de gráficos: de barras, e de linha. Uma das características mais interessantes desta libraria é que os gráficos são do tipo responsivo, ou seja, adaptam-se ao espaço disponível na página.

O primeiro passo para desenhar um gráfico passa por criar um elemento canvas na página html, para tal usa-se o código seguinte:

```
<canvas id="mychart" style="display:
inline;"></canvas>
```

De seguida, no ficheiro de JavaScript, usasse o método "getElementById()" para obter o elemento canvas onde se pretende desenhar o gráfico. Consequentemente tem de ser criada uma variável do tipo "new chart" que recebe por parâmetros o elemento canvas e um objeto que contém os dados e as configurações do gráfico, tal com mostra o código abaixo:

```
bigctx = document.getElementById("mychart");
bigchart= new Chart (bigctx, bigchartinfo);
```

Para o desenho do gráfico é preciso passar o objeto "bigchartinfo" que deve conter as seguintes propriedades:

1. "type" – especifica o tipo de gráfico que deve ser desenhado (exemplos: barras, linha)

2. "labels" - contém um vetor de *strings* que especifica o nome das instâncias a ser comparadas no gráfico

3. "datasets" – é um objeto que deve conter no mínimo os dados que se pretende introduzir no gráfico e uma "label" que especifica o tipo de dados

```
var bigchartinfo = {
            type: 'bar',
            data: {
                labels: ['semana 1','semana 2'],
                datasets: [{
                            data: [250,300],
                            label: "Posto 1"
                           },
                           {
                           data: [300,260],
                           label: "Posto 2"
                           }]
                }
```

4. ANÁLISE DOS RESULTADOS

4.1. INTERFACE DE SUPERVISIONAMENTO DE LINHA

Na figura 14, está apresentado o resultado da interface gráfica com destino a controlo e supervisionamento das seis linhas de produção. Nesta interface é possível observar os dados de produção em tempo real. O turno é detetado automaticamente, de acordo com a hora do sistema, e os dados são atualizados de três em três segundos, por meio de um pedido AJAX. Todas as colunas apresentam uma linha verde que indica o período do momento. Nesta fase foram cumpridos todos os objetivos e requisitos que foram propostos pela empresa. Para implementar o estilo da página html foi usado um ficheiro de código CSS escrito manualmente e sem recurso a nenhuma biblioteca ou API.

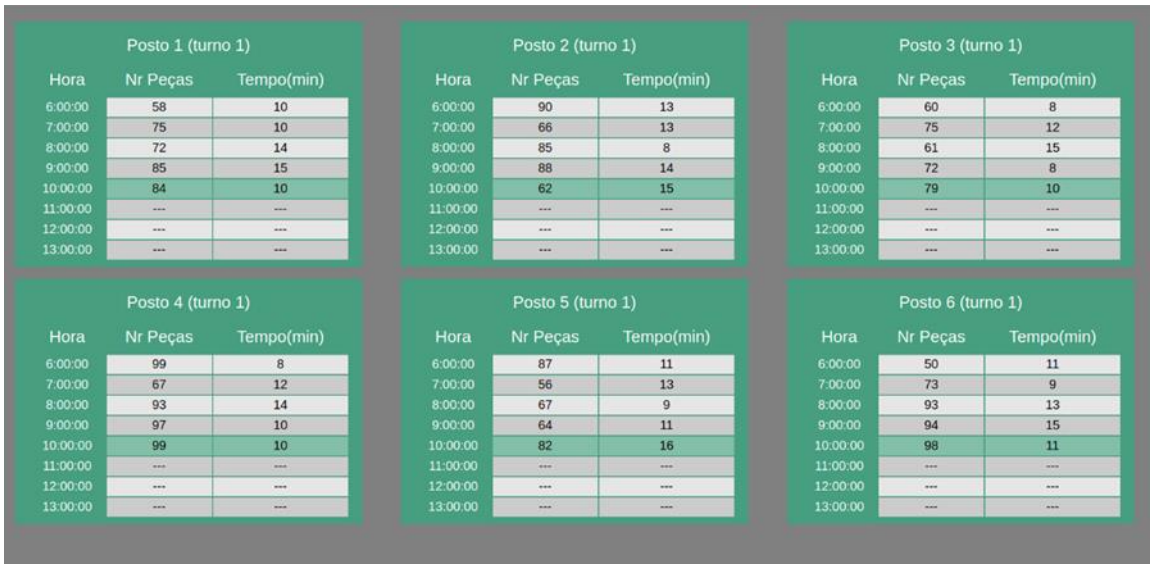

Figura 14 Interface gráfica de supervisionamento

4.2. APLICAÇÃO WEB PARA ANÁLISE ESTATÍSTICA

Nesta fase do projeto foi implementada uma aplicação Web com várias páginas html que apresentam diferentes tipos de informação. Quando a aplicação é iniciada é necessário efetuar o login, figura 15. Como foi referido no capítulo anterior o sistema de autenticação e autorização é baseado em JWT e expira a cada 30 minutos.

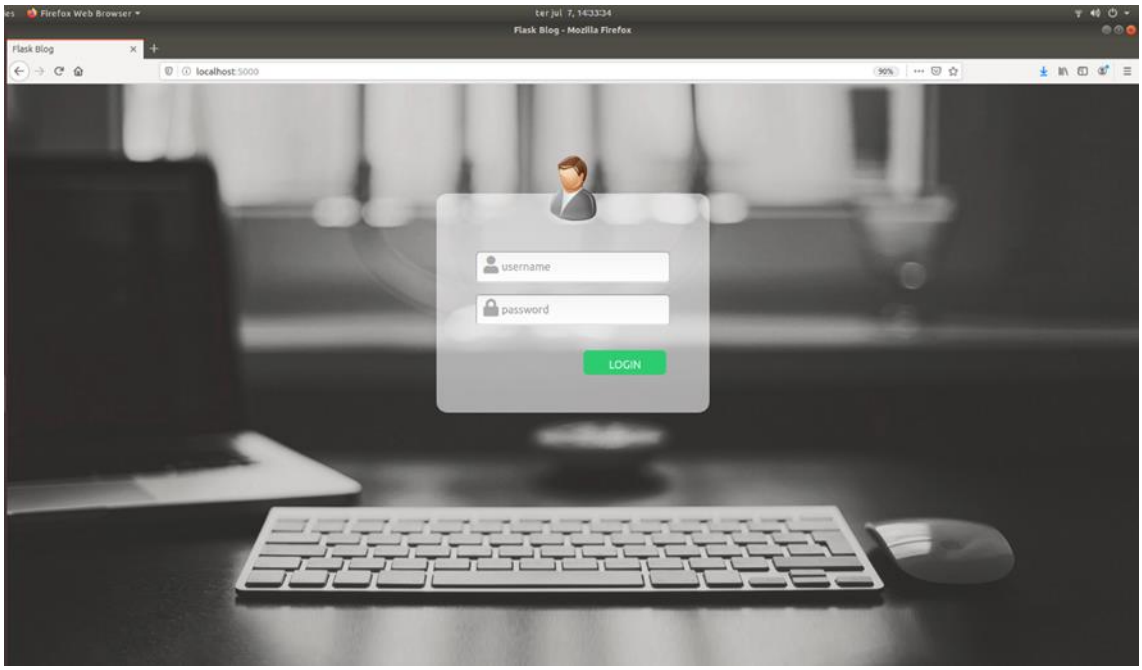

Figura 15 Página de login

O utilizador após efetuar o login é redirecionado para a página principal, designada por "home", apresentada na figura 16. Todas as páginas da aplicação têm um menu fixo na parte superior que facilita a navegação pelas diferentes funcionalidades do sistema.

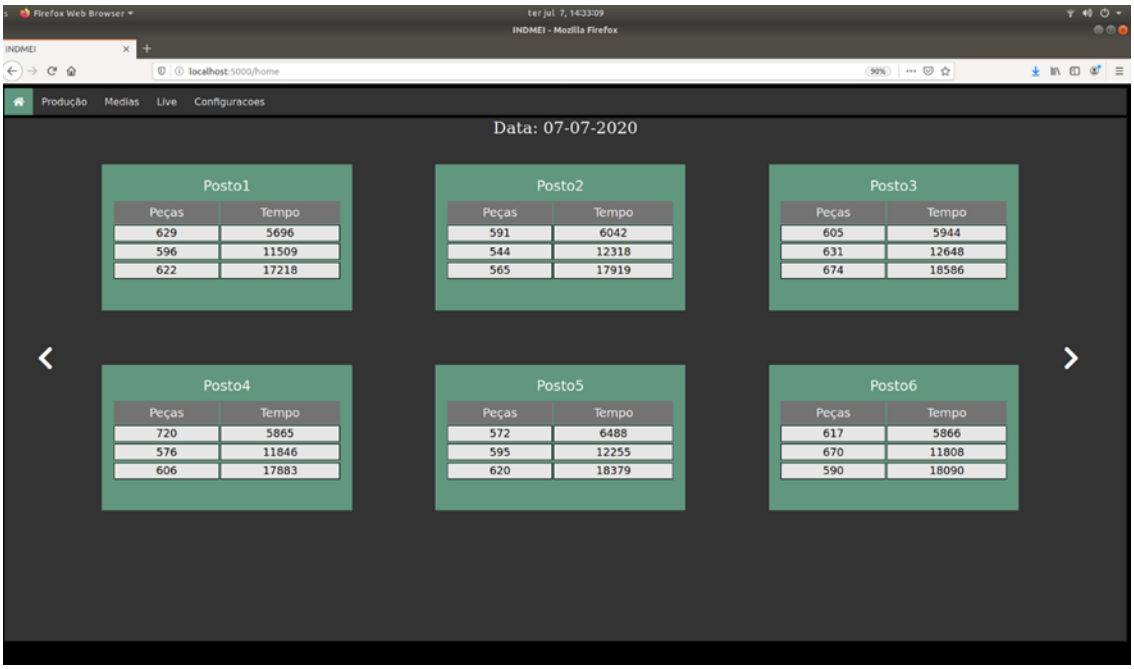

Figura 16 Página "home"

Nesta página o utilizador consegue visualizar dois tipos de informação: o número diário total de peças produzidas e o total diário de tempo de paragem (em minutos). Esta informação está dividia por turno, ou seja cada linha da tabela corresponde a um turno diferente. As duas setas laterais permitem mudar a informação apresentada para o dia desejado.

A figura 17, mostra a página “produção” que está encarregue de mostrar a informação de produção semanal de todos os postos. Para tal, são geradas 6 tabelas com os sete dias da semana. Nestas tabelas é possível observar os seguintes dados:

1. Número de peças e tempo total de paragem por turno, por dia
2. Total de peças e tempo de paragem dos três turnos, por dia
3. Total de peças produzidas e tempo de paragem de cada turno, por semana
4. Total de peças e tempo de paragem de todos os turnos, por semana

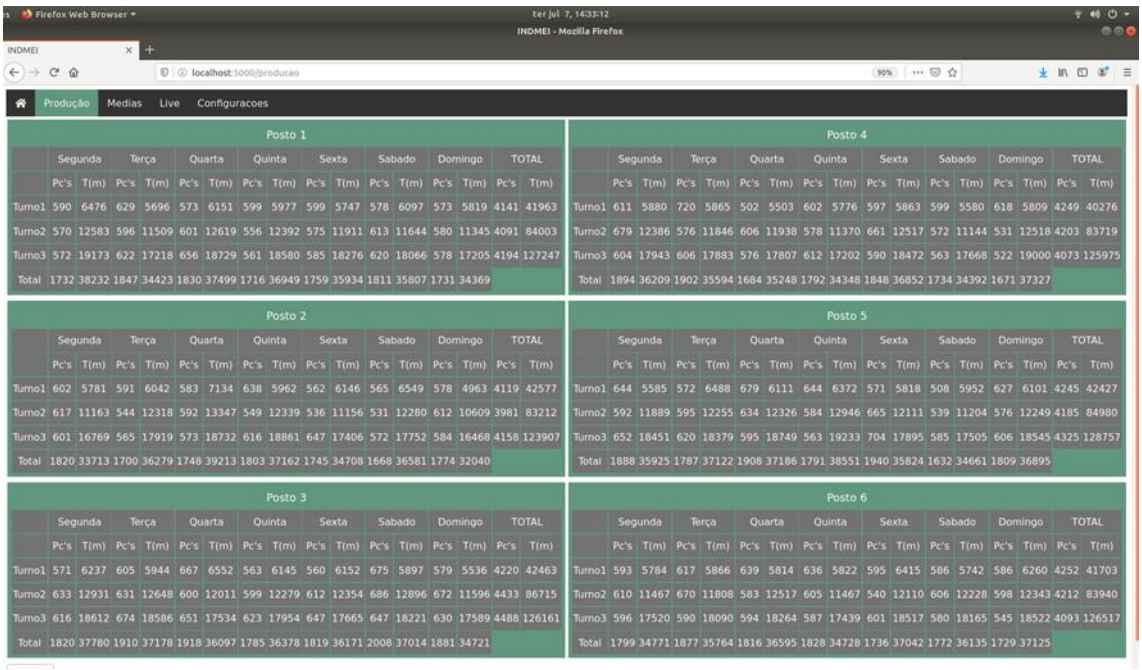

Figura 17 Pagina Produção

Na figura 18, está apresentada a página designada por “dashboard”. Nesta os dados são apresentados de forma mais amigável e interativa, por meio de gráficos de barras.

No primeiro gráfico, e também o maior, é apresentado o número total de peças produzidas por cada posto. Dependendo do input escolhido pelo utilizador os dados podem ser apresentados por semana ou por mês, o número de semanas apresentadas no gráfico pode ser incrementado usando o *slider* implementado em cima.

Nos outros gráficos, mais pequenos, podem ser visualizados os dados de produção de cada turno para cada posto de trabalho. Dependendo do input escolhido os dados podem corresponder aos últimos 7 dias; 15 dias; ou último mês. No contentor mais à direita é apresentado o total de todos os turnos de cada linha de produção.

Figura 18 Página "Dashboard"

Na página representada na figura 19 é possível fazer uma análise mais detalhada ao ritmo de produção ao longo do dia, isto porque para cada posto é apresentado um gráfico que mostra o número de peças produzido por hora em tempo real. Isto permite que os picos de produção sejam facilmente identificados pelo utilizador. Para complementar também são apresentados os seguintes dados:

1. Total de peças produzidas por posto até ao momento

2. Média de peças produzidas por hora

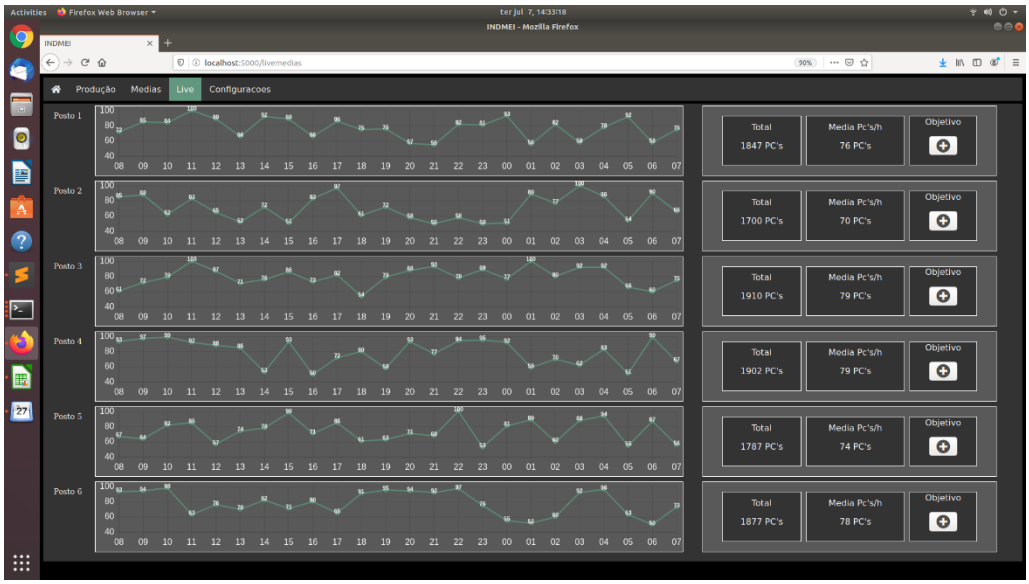

Figura 19 Pagina "Live"

5. CONCLUSÕES

Este trabalho foi realizado no âmbito de um estágio na área da engenharia eletrotécnica e computadores, proporcionada pela Indmei, empresa dedicada ao fabrico de meias.

Com a conclusão deste projeto foi possível obter uma aplicação funcional, sem licenças de *software*, para controlo e análise de dados de produção. Todas as exigências e objetivos propostos do início do trabalho foram atingidos com sucesso.

A implementação deste sistema permite que a administração tenha mais controlo e mais informação sobre as linhas de produção, e torna a análise dos dados mais simples, o que aumenta a eficiência das tomadas de decisão relativas aos postos de trabalho.

Este sistema permite que os próprios operários de linha tenham mais consciência do seu ritmo de trabalho, pois podem ver a produção na interface gráfica de supervisionamento na nav de fabrico, e podem se comparar aos outros postos de trabalho.

Durante a execução do trabalho surgiram algumas dificuldades devido à falta de conhecimentos, nomeadamente sobre leitura dos dados recebidos na porta série em sistema UNIX e a programação em JavaScript, especialmente na troca de informação entre o do

servidor e o módulo do cliente. No entanto, estas dificuldades foram ultrapassadas com o estudo das respetivas matérias.

Este projeto permitiu adquirir conhecimentos na área das linguagens WEB, criação e gestão de sistemas de informação. Por outro lado, ao realizar o trabalho em contexo empresarial, permitiu ter uma maior envolvência com a indústria, percebendo a sua dinâmica e modos de funcionamento.

Em projetos futuros poder-se-á implementar mais módulos de análise estatística, assim como:

- Previsões estatísticas com base nos dados armazenados
- Criação de padrões diários que permitam a deteção automática de picos ou quebras de trabalho
- Criação automática de *backups* da base de dados

Referências Documentais

[1] MySQL—MySQL 8.0 Reference Manual :: 1.3.1 What is MySQL?, 2020

[2] Invensis Technologies. 2020. Benefits Of Python Over Other Programming Languages - Invensis Technologies. [online] Available at: <https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/>.

[3] Pythonbasics.org. 2020. What Is Flask Python - Python Tutorial. [online] Available at: <https://pythonbasics.org/what-is-flask-python/> [Accessed 7 July 2020].

[4] Medium. 2020. Jinja2 Explained In 5 Minutes!. [online] Available at: <https://codeburst.io/jinja-2-explained-in-5-minutes-88548486834e> [Accessed 7 July 2020].

[5] EDUCBA. 2020. How Javascript Works? | Understanding On How Javascript Functions?. [online] Available at: <https://www.educba.com/how-javascript-works/> [Accessed 7 July 2020].

[6] js.foundation, J., 2020. Jquery. [online] Jquery.com. Available at: <https://jquery.com/> [Accessed 7 July 2020].

[7] MDN Web Docs. 2020. *Ajax*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX> [Accessed 7 July 2020].

[8] js.foundation, J., 2020. *Jquery.Ajax() | Jquery API Documentation*. [online] Api.jquery.com. Available at: <https://api.jquery.com/jQuery.ajax/> [Accessed 7 July 2020].

[9] Json.org. 2020. *JSON*. [online] Available at: <https://www.json.org/json-en.html> [Accessed 7 July 2020].

[10] Weis, O., 2020. Serial Communication Programming Tutorial [Eltima Software]. [online] Eltima Software. Available at: <https://www.eltima.com/serial-programming-for-windows-and-linux.html> [Accessed 7 July 2020].

[11] Man7.org. 2020. Termios(3) - Linux Manual Page. [online] Available at: <https://man7.org/linux/man-pages/man3/termios.3.html> [Accessed 7 July 2020].