Escola Superior de Tecnologia e Gestão

**Supermercado Inteligente**

Bruno Alexandre Lopes Ferreira

Ricardo Bento Santos

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, Junho de 2022

# Lista de Figuras

Figura 1 - Arquitetura IoT 2

Figura 2 - Enderçamentos IP Arquitetura 3

Figura 3 – Diagrama evento câmara PY 7

Figura 4 - Diagrama Evento Câmara PY 8

Figura 5 - Diagrama evento temperatura frigorifico PKT 9

Figura 6 - Imagem geral do cenário de testes 10

Figura 7 - Imagem PKT Cenário Físico 10

Figura 8 - Imagem output PY portas 21

Figura 9 - Imagem output PY câmara 22

# Lista de tabelas

Tabela 1 - Eventos 3

# Lista de siglas e acrónimos

|  |  |
| --- | --- |
| API  ESTG | Application Programing Interface  Escola Superior de Tecnologia e Gestão |
| IPLeiria | Instituto Politécnico de Leiria |
| REST  PKT  PY | Representational State Transfer  Packet Tracer  Python |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Índice

[Lista de Figuras iv](#_Toc106008966)

[Lista de tabelas v](#_Toc106008967)

[Lista de siglas e acrónimos vi](#_Toc106008968)

[1. Introdução 1](#_Toc106008969)

[2. Arquitetura 2](#_Toc106008970)

[3. Implementação 6](#_Toc106008971)

[4. Cenário de Teste 9](#_Toc106008972)

[5. Resultados obtidos 22](#_Toc106008973)

[6. Conclusão 25](#_Toc106008974)

[7. Bibliografia 26](#_Toc106008975)

[8. Anexos 27](#_Toc106008976)

# Introdução

Este relatório tem como propósito explicar detalhadamente todas as funcionalidades desenvolvidas no trabalho, cujo tema é **“Supermercado/Comércio Inteligente”**. A razão que nos incentivou a eleger este tema, foi o espaço livre, tanto para automação de tarefas, como para a passagem de informação (p.e nível de stock secção líquidos, pessoas na loja (…)) nos supermercados. Acreditamos que estas funcionalidades poderiam ser adaptadas em muitos estabelecimentos, num contexto em que se pretenda obter um **Supermercado Inteligente**.

Tratamos como objetivos gerias e principais a:

* Automação de tarefas (p.e abrir caixas consoante nº de pessoas na loja)
* Passagem de informação para o gerente e trabalhadores

E ainda, os objetivos específicos (relacionados com a resolução **ineficácias** dos supermercados):

* Gestão de pessoas na loja e no estacionamento
* Eficaz monitorização de temperatura na secção de congelados e frios
* Eficaz monitorização de stock
* Controlo automático do número de caixas abertas
* Sistema básico de segurança
* Proteção automática antifogo / antifumo

Tendo em conta todos estes aspetos mencionados, tentamos implementar possíveis soluções para os mesmos no nosso trabalho, que se encontra dividido em:

* Recolha de dados – PKT ou PY
* Comunicação dos dados – através REST API - métodos HTTP
* Tratamento e apresentação de dados – Website

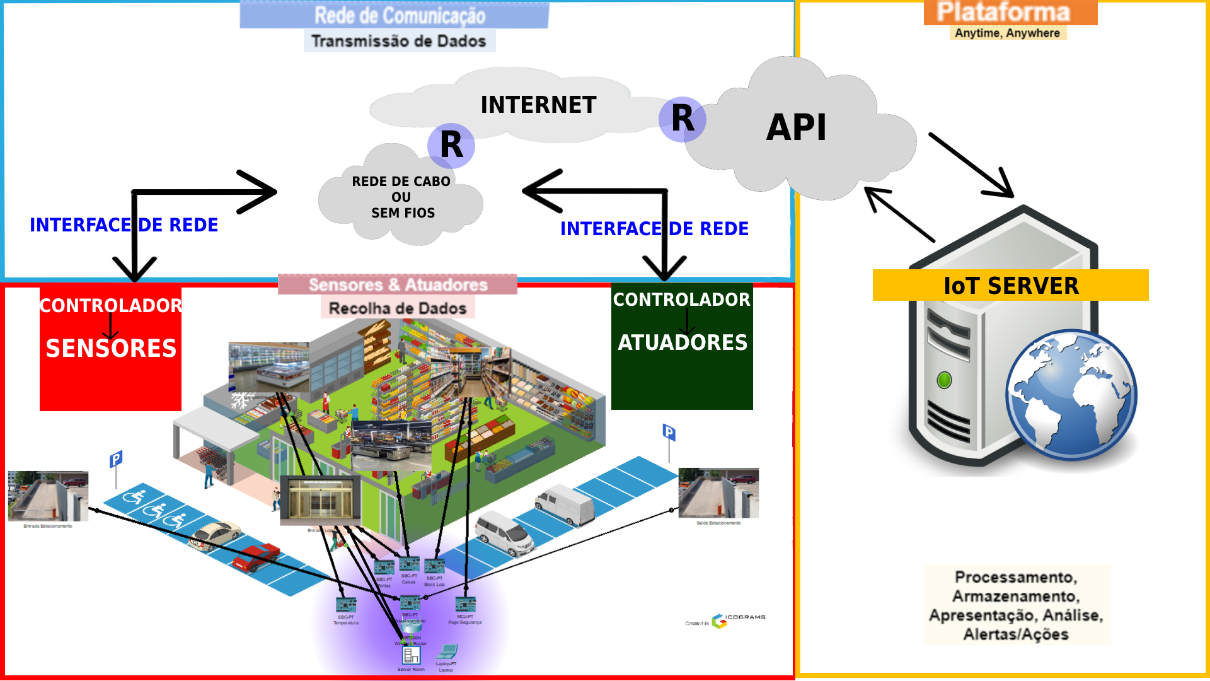
# Arquitetura

A arquitetura do nosso trabalho encontra-se dividido em três principais fases (como se pode ver pela figura).

Primeiramente dá-se a recolha de dados (simulada no PKT e no PY), que representa o nosso “Ambiente físico”, no mesmo inserem-se sensores que têm como função recolher dados e atuadores que recebem “instruções” provenientes dos microcontroladores (MCU e SBC).

De seguida, os microcontroladores estabelecem comunicação com uma rede sem fios (2ª fase), onde são responsáveis pela transmissão de dados obtidos na 1ª fase, enviados para o servidor na cloud (API) através dos métodos POST e GET.

Por último, todos os dados recolhidos e enviados, serão processados, tratados e armazenados, de modo a serem dispostos num serviço web (Dashboard).



**GET**

**GET**

**GET**

**POST**

**POST**

**POST**

**SITE WEB**

Figura - Arquitetura IoT

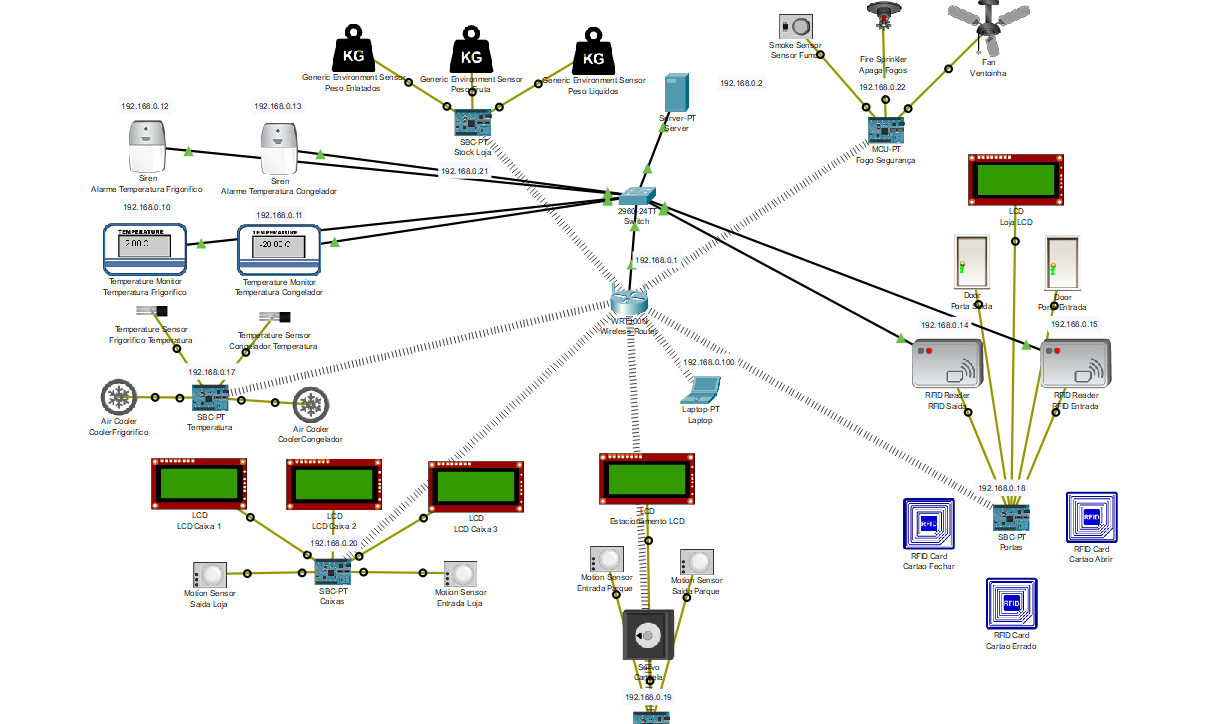
Sensores usados

* 2x Sensores de Temperatura
* 4x Sensores Movimentos
* 2x RFID Reader
* 1x Sensor de fumo
* 3x Generic Sensor (Simula sensor de peso)

Atuadores usados

* 2x Siren (Alarmes)
* 5x LCD
* 1x Sprinkler
* 1x Ventoinha
* 2x Portas
* 2x Cooler
* 1x Servomotor

**Endereçamentos IP**



192.168.0.19

192.168.0.18

192.168.0.20

192.168.0.17

192.168.0.11

192.168.0.10

192.168.0.12

192.168.0.13

192.168.0.21

192.168.0.100

192.168.0.1

192.168.0.2

192.168.0.22

Figura - Enderçamentos IP Arquitetura

ORIGEM DOS EVENTOS RESULTADO DOS EVENTOS

Tabela - Eventos

|  |  |
| --- | --- |
| PY:  Escolha do estado das portas através do PY através das teclas (POST) | PKT:  Alteração do estado da porta (fechada ou aberta) |
| PY:  Escolha do estado das portas através do PY através das teclas (POST) | DASHBOARD:  Alteração do estado exibido da porta (fechada ou aberta) |
| DASHBOARD: \*\*Apenas Gerente\*\*  Abrir porta de entrada ou saída (através dos botões POST com JavaScript para a API) | PKT:  Alteração do estado da porta (fechada ou aberta): |
| DASHBOARD: \*\*Apenas Gerente\*\*  Abrir porta de entrada ou saída (através dos botões POST com JavaScript para a API) | PY:  Tira foto automaticamente |
| PKT: (por exemplo)  Leitura da temperatura, POST com o valor, se o valor for superior ao limite tolerado | DASHBOARD:  Na Dashboard aparece que ocorreu um erro na página dos Erros e é também mostrado que há uma notificação por ver |
| PKT:  Abrir porta entrada ou saída - RFID card | PY:  Tira foto automaticamente |

Para complementar o nosso trabalho, optamos por implementar as seguintes funcionalidades:

* "Utilização de bibliotecas para apresentar gráficos dos dados em HTML/CSS/PHP/JavaScript"
  + O gráfico apresentado é sobre o valor do lucro, que é calculado através das vendas – compras, ou seja para obter mais um valor no gráfico é necessário enviar 2 POST’s um com o valor das vendas e outro com o valor das compras que depois é adicionado uma posição no gráfico.
* "Segurança dos dados aplicacionais (algoritmos de cifragem no código)"
  + A API encarrega-se de codificar \*TODOS\* os parâmetros recebidos pelo POST, que depois pode ser desencriptado através dos seguintes parâmetros:
* $iv = "1234567891011121";
* $key="nPIMLCuuYW";
* $cipher = "AES-128-CTR";
* "Utilização de um número de sensores e atuadores superior aos mínimos exigidos" - **2x atuadores diferentes a mais**
* "Plataforma web com pelo menos 3 utilizadores e com 3 privilégios diferentes" – **Cliente ; Trabalhador ; Gerente**
* "Criação de sensores, atuadores genéricos e personalizados no CPT" - **Sensores Peso**
* "Criação de scripts no CPT com o objetivo de controlar/alterar valores de Environment" - **Coolers Frigorifico e Congelador**
* "Criação de uma página com histórico das últimas imagens recebidas"
* “Controlo das imagens recebidas através de código PHP de modo a aceitar apenas as imagens com um tamanho máximo de 1000kB e com a extensão .jpg ou .png"
* "Outras funcionalidades e requisitos além dos anteriores, serão analisados caso a caso"
  + Javascript do site (sidebar animação tanto na dashboard como na página inicial ; relógio digital página cliente ; código de desconto do cliente; popups)
    - Animação da sidebar (exemplo ficheiro: PaginaInicial.html ; dashboard.php)
    - Relógio Digital (exemplo ficheiro: dashboard.php)
    - Código de Desconto (exemplo ficheiro: sensores-Dashboard.php)
    - Popups (exemplo ficheiro: erros.php ; paginaInicial.html)



* + Mensagens de erro
    - quando temperatura frigorifico > 5ºC
    - quando temperatura congelador > -18ºC
    - quando número de entrada da loja < número saída da loja
    - quando número de entrada no parque < número saída do parque
  + Tabela com status dos sensores e com a data do último funcionamento
  + Gerente tem acesso a interface de cliente

# Implementação

Integralmente, todos os algoritmos do nosso trabalho são modelados a partir do conceito exibido na Arquitetura, recolher dados, POST com os dados e por último a sua exibição. No entanto, os algoritmos de maior importância são os retratados na tabela de eventos, apesar de todos serem igualmente importantes para o bom funcionamento do Supermercado.

**FLUXOGRAMA EVENTO PYTHON CÂMARA**

Diagram

Description automatically generated

Figura – Diagrama evento câmara PY

**FLUXOGRAMA EVENTO PORTAS DASHBOARD – APENAS PARA GERENTE**

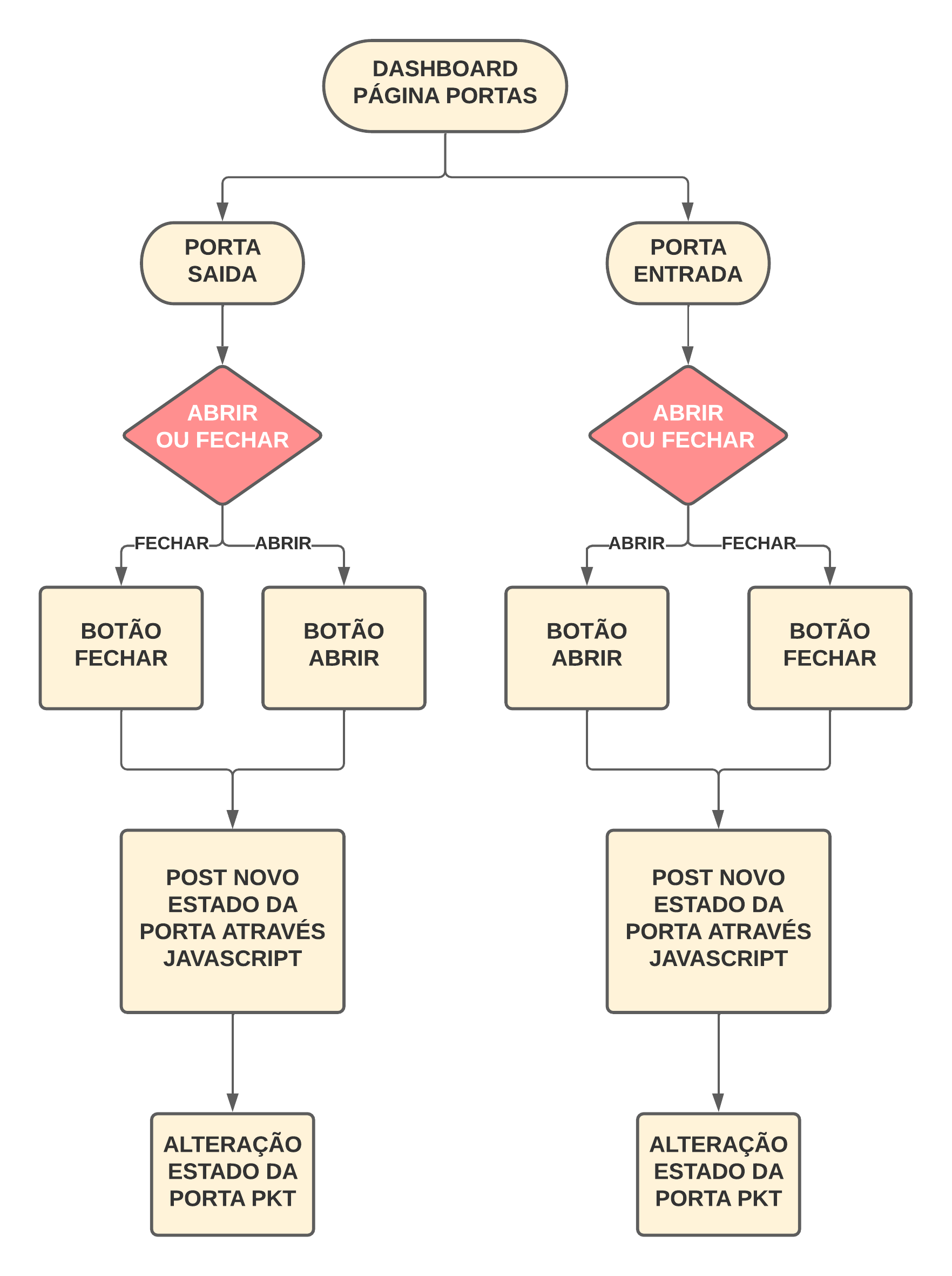
****

Figura - Diagrama Evento Câmara PY

**FLUXOGRAMA EVENTO LEITURA DE TEMPERATURA PKT**

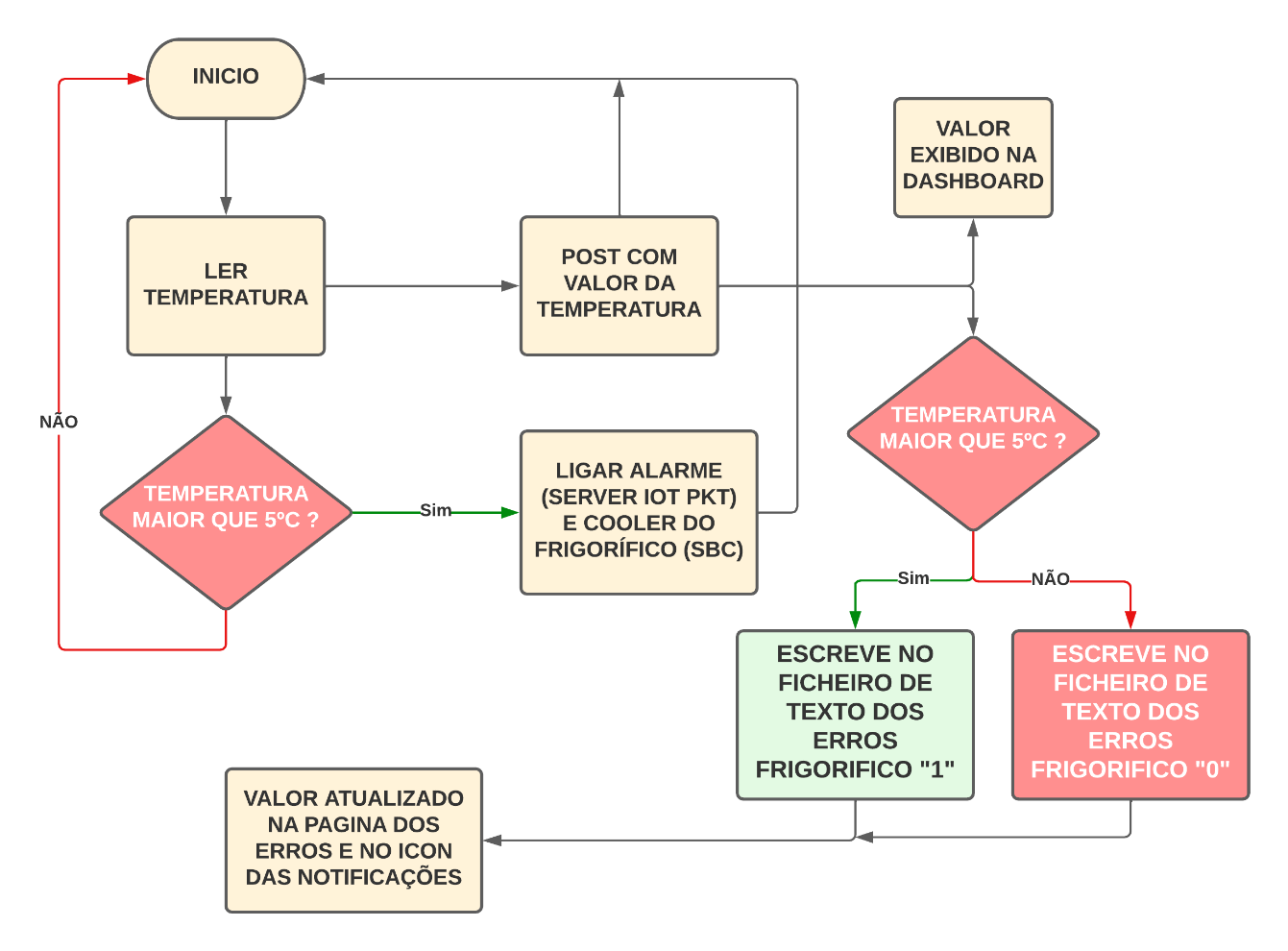
**EXEMPLO DO FRIGORÍFICO**

Figura - Diagrama evento temperatura frigorifico PKT

# Cenário de Teste

Para poder simular os sensores e atuadores usados no nosso supermercado inteligente, utilizamos o software “Cisco Packet Tracer” (figura 5 – ambiente simulação), a Dashboard criada por nós, de modo a podermos alterar o estado das portas através da mesma, e por último, o PY, onde conseguimos tirar fotos e, também, alterar o estado das portas.

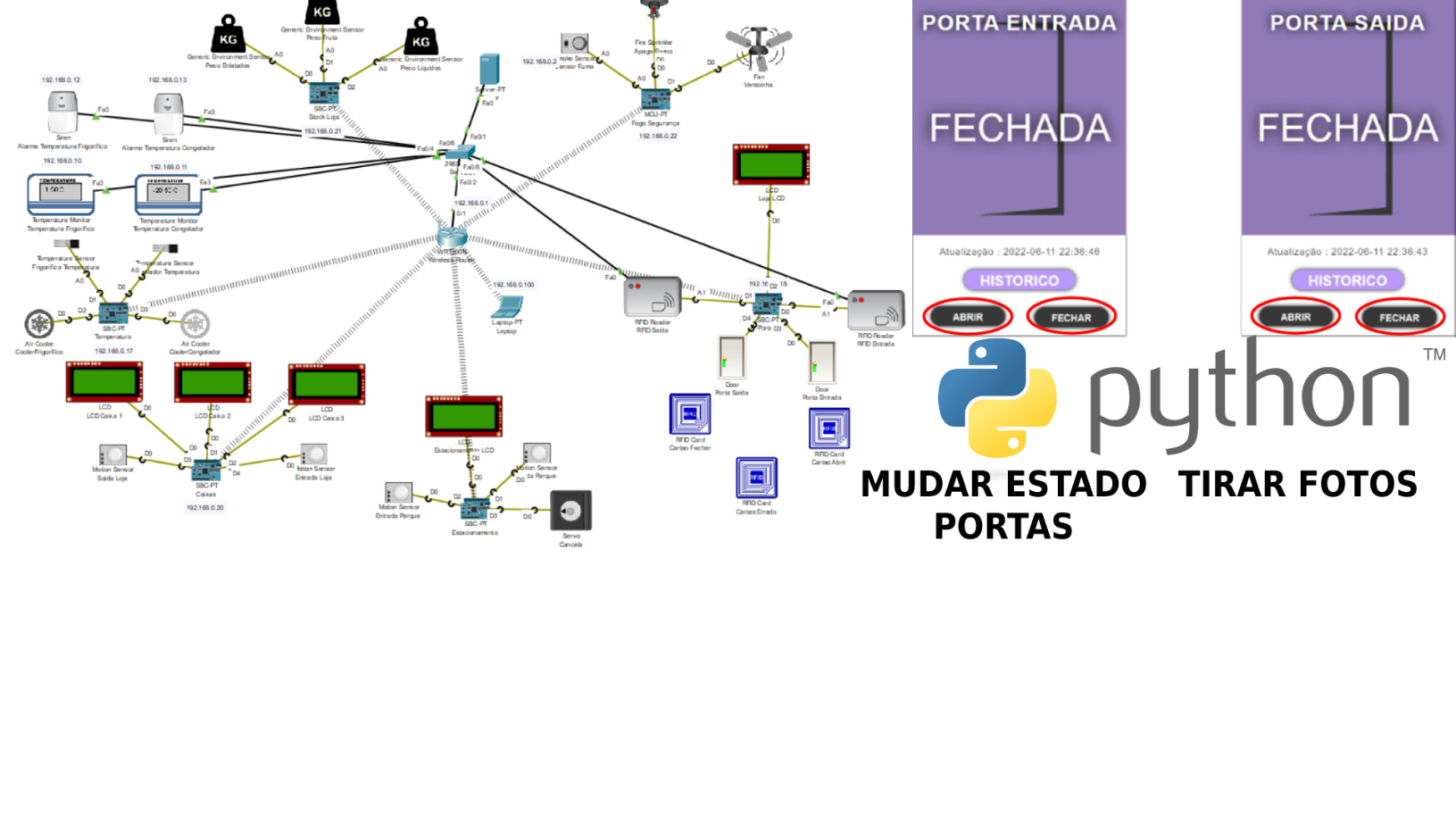


Figura - Imagem geral do cenário de testes

**NOTA:**

* Diagram

  Description automatically generatedNo PKT, criamos vários ambientes físicos para as diferentes localizações dos sensores como podemos ver pela figura.

Figura - Imagem PKT Cenário Físico

**SENSORES DE PESO (PKT)**

Todos os sensores de peso são adaptados dos “Generic Sensors”, visto que não nos era disponibilizado no PKT os mesmos. Tivemos de criar novas variáveis de ambiente (fruta, enlatados e líquidos) independentes que não têm qualquer tipo de efeito nas restantes, que se encontram na categoria “Other” das variáveis de ambiente.

* **Sensor Peso da Enlatados**

1. Encontra-se ligado ao SBC-PT “Stock Loja”: A0 (sensor) → D0 (SBC)
2. No generic sensor tivemos que definir a variavel lida para “PesoEnlatados”, com um valor máximo de 12 e mínimo de 0
3. Quando ligado, lê o valor da variável “PesoEnlatados” de 1 em 1 segundo, e envia o mesmo valor através de um POST para a API (juntamente com a data da leitura, o status do sensor “ON” e nome “PesoEnlatados”).
4. Esse valor é atualizado no respetivo ficheiro, e caso seja menor que 2 é mostrada a mensagem na dashboard “Necessário Repor” caso seja maior ou igual 2 a mensagem é “Em stock”

* **Sensor Peso de Frutas**

1. Encontra-se ligado ao SBC-PT “Stock Loja”: A0 (sensor) → D1 (SBC)
2. No generic sensor tivemos que definir a variável lida para “PesoFrutas”, com um valor máximo de 12 e mínimo de 0
3. Quando ligado, lê o valor da variável “PesoFrutas” de 1 em 1 segundo, e envia o mesmo valor através de um POST para a API (juntamente com a data da leitura, o status do sensor “ON” e nome “PesoFrutas”)
4. Esse valor é atualizado no respetivo ficheiro, e caso seja menor que 2 é mostrada a mensagem na dashboard “Necessário Repor” caso seja maior ou igual 2 a mensagem é “Em stock”

* **Sensor Peso Líquidos**

1. Encontra-se ligado ao SBC-PT “Stock Loja”: A0 (sensor) → D2 (SBC).
2. No generic sensor tivemos que definir a variável lida para “PesoLiquidos”, com um valor máximo de 25 e mínimo de 0.
3. Quando ligado, lê o valor da variável “PesoLiquidos” de 1 em 1 segundo, e envia o mesmo valor através de um POST para a API (juntamente com a data da leitura, o status do sensor “ON” e nome “PesoLiquidos”).
4. Esse valor é atualizado no respetivo ficheiro, e caso seja menor que 10 é mostrada a mensagem na dashboard “Necessário Repor” caso seja maior ou igual 10 a mensagem é “Em stock”.

**SENSORES DE TEMPERATURA E COOLERS (PKT)**

Ambos os sensores de temperatura leem variáveis de ambiente criadas por nós, a “TemperaturaCongelador” e “TemperaturaFrigorifico”, que não influenciam as restantes.

* **Sensor Temperatura Congelador e Cooler Congelador**

1. Sensor e Cooler ligados ao SBC-PT “Temperatura”

Sensor: A0 (sensor) → D0 (SBC)

Cooler: D0 (sensor) → D3 (SBC)

1. O SBC lê por analogRead(D0) o valor de 0 a 1023 que o sensor vai enviar, depois a partir de uma função chamada lerTemperatura irá devolver o valor da mesma em graus.
2. De seguida é realizado um POST para a API de 2 em 2 segundos, com o nome do sensor da temperatura do congelador “TemperaturaCongelador”, o valor lido, a data atual e o status do sensor “ON”, cujo valor vai ser atualizado na dashboard.
3. Se o valor da temperatura passado pelo POST for maior do que -18ºC, a API, irá escrever o valor 1 no ficheiro de texto das Notificações TemperaturaCongelador, que irá atualizar as notificações da Dashboard e página dos erros
4. Se a temperatura for maior do que -18ºC, o que significa que os produtos estão numa temperatura que prejudica a sua qualidade, o Cooler do congelador é ligado através da função digitalWrite(D3, HIGH).
5. No servidor IOT também existe uma condição que quando a temperatura do congelador passar dos -18ºC, a sirene é ligada. Quando a temperatura desce dos -18ºC o Cooler do coneglador é desligado e a sirene também.

* **Sensor Temperatura Frigorifico e Cooler Frigorifico**

1. Sensor e Cooler ligados ao SBC-PT “Temperatura”

Sensor: A0 (sensor) → D1 (SBC)

Cooler: D0 (sensor) → D2 (SBC)

1. O SBC lê por analogRead(D1) o valor de 0 a 1023 que o sensor vai enviar, depois a partir de uma função chamada lerTemperatura irá devolver o valor da mesma em graus.
2. De seguida é realizado um POST para a API de 2 em 2 segundos, com o nome do sensor da temperatura do congelador “TemperaturaFrigorifico”, o valor lido, a data atual e o status do sensor “ON”, cujo valor vai ser atualizado na dashboard.
3. Se o valor da temperatura passado pelo POST for maior do que 5ºC, a API, irá escrever o valor 1 no ficheiro de texto das Notificações TemperaturaFrigorifico, que irá atualizar as notificações da Dashboard e página dos erros
4. Se a temperatura for maior do que 5ºC, o que significa que os produtos estão numa temperatura que prejudica a sua qualidade, o Cooler do frigorifico é ligado através da função digitalWrite(D2, HIGH).
5. No servidor IOT também existe uma condição que quando a temperatura do congelador passar dos 5ºC, a sirene é ligada. Quando a temperatura desce dos 5ºC o Cooler do frigorifico é desligado e a sirene também.

**PARQUE DE ESTACIONAMENTO (PKT)**

É composto pela entrada e pela saída, na entrada encontram-se um servomotor (simula a cancela), um sensor de movimento e um LCD, e na saída, um sensor de movimento.

* **Entrada Parque** 
  1. Sensores e atuadores ligados ao SBC-PT “Estacionamento”

Sensor de movimento: D0 (sensor) → D2 (SBC)

LCD: D0 (sensor) → D0 (SBC)

Servomotor: D0 (sensor) → D3 (SBC)

* 1. Através da função digitalRead(D2) conseguimos saber se o sensor foi ativado ou não pois esta função devolve HIGH ou LOW. (é ativado quando há movimento).
  2. Se o sensor de movimento for ativado é realizado um GET de modo a saber a quantidade de cliente.es que entrou no parque, de seguida é aumentado esse valor por 1
  3. Depois é realizado um POST para a API de 6 em 6 segundos (o sensor de movimento apenas deteta movimento de 5 em 5 segundos) com o nome do sensor de movimento “EntradaParque”, o valor atrás obtido, a data atual, e o status do sensor “ON”.
  4. E o valor passado no POST será atualizado na Dashboard
  5. **NUM JAVASCRIPT À PARTE NO SBC**
     + São realizados dois GET à API do valor de carros que entraram no parque e a quantidade de carros que saíram do parque, para calcular o total de carros no parque
     + Breve verificação se os GETS correram bem (status == 200)
     + Verifica-se se o valor de carros que entrou é maior ou igual ao número de carros que saiu, se tal acontecer é guardado o numero de carros no parque na variável “total” (entrada - saída)
     + Se a condição não se verificar é escrito no LCD “Chamar Gerente” (saídas maiores que entradas – erro) e servomotor fechado customWrite(D3, -127).
     + Se total for menor que 50 é escrito no LCD “Parque Livre” e o servomotor vai estar aberto customWrite(D3, 110).
     + Se total for igual a 50 é escrito no LCD “Parque Ocupado” e o servomotor vai estar fechado customWrite(D3, -127).
* **Saida Parque** 
  1. Sensores e atuadores ligados ao SBC-PT “Estacionamento”

Sensor de movimento: D0 (sensor) → D1 (SBC)

* 1. Através da função digitalRead(D1) conseguimos saber se o sensor foi ativado ou não pois esta função devolve HIGH ou LOW. (é ativado quando há movimento).
  2. Se o sensor de movimento for ativado é realizado um GET de modo a saber a quantidade de clientes que saiu do parque, de seguida é aumentado esse valor por 1
  3. Depois é realizado um POST para a API de 6 em 6 segundos (o sensor de movimento apenas deteta movimento de 5 em 5 segundos) com o nome do sensor de movimento “SaidaParque”, o valor atrás obtido, a data atual, e o status do sensor “ON”.
  4. E o valor passado no POST será atualizado na Dashboard
* **NOTAS**
  1. Na API antes de os valores do parque serem escritos é calculado o total de pessoas no parque, se esse número for menor que 0 é escrito nas Notificações/PessoasParque o valor “1”, ou seja ocorreu um erro e este vai ser mostrado na página de Erros e nas Notificações, caso esteja dentro do normal (maior que 0) é escrito “0”
  2. Na dashboard é calculado o número de clientes no parque (entradas - saídas), se este valor for menor que 0 é mostrada a mensagem de erro, se for menor que 50 e maior que 0, é mostrada a mensagem “Parque Livre”, se for igual a 50 é mostrada a mensagem “Parque Ocupado”

**LOJA ENTRADA E SAIDA (PKT)**

Na entrada e saída da loja foi usado em cada, uma porta, um RFID reader, um sensor de movimento, e adicionalmente em comum um LCD para o estado da loja, mais 3 LCD’S para o estado das caixas e por último os RFID cards.

* **Entrada da Loja**

1. Porta e RFID reader ligados ao SBC-PT “Portas”

Porta: D0 (sensor) → D4 (SBC)

RFID reader: A1 (sensor) → D1 (SBC)

Sensor de movimento ligado ao SBC-PT “Caixas”: D0 (sensor) → D3 (SBC)

1. Como criamos cartões distintos para abrir e fechar as portas, tivemos de configurar o RFID reader da entrada para apenas reconhecer cada um destes id’s e consoante o valor passado para o SBC este reagir de forma diferente.

RFID card abrir id: 1001 ; valor SBC = 8

RFID card fechar id: 1002 ; valor SBC = 0

1. No inicio do loop é logo realizado um GET para saber se o estado da porta se encontra bem, caso for diferente do atual, atualiza para o certo.
2. Depois ao passar os cartões do RFID reader obtemos ações diferentes, mas o algoritmo em si é o mesmo, primerio é lido o valor do RFID reader através de “cartao = analogRead(0)” de meio em meio segundo.
3. De seguida é realizado um GET à API para saber se o estado da porta é igual à ação pretendida por parte do utilizador, caso seja mantém o estado da porta igual e não envia POST. Caso seja diferente, realiza POST com o novo estado da porta, juntamente com a data atual e nome “PortaEntrada”, e atualiza o estado da porta através do customWrite.
4. Como o sensor de movimento encontra-se ligado a um SBC diferente o seu código é distinto. Através da função digitalRead(D4) sabemos se o sensor foi ativado ou não, pois esta função devolve HIGH ou LOW.
5. Se o sensor de movimento for ativado é realizado um GET de modo a saber a quantidade de clientes que entrou na loja, de seguida é aumentado o valor por 1.
6. Depois é realizado um POST para a API de 6 em 6 segundos (sensor de movimento deteta movimento de 5 em 5 segundos) com o nome do sensor de movimento “ClientesEntrada” , valor obtido, a data atual e o status do sensor “ON”

* **Saida da Loja**

1. Porta e RFID reader ligados ao SBC-PT “Portas”

Porta: D0 (sensor) → D4 (SBC)

RFID reader: A1 (sensor) → D1 (SBC)

Sensor de movimento ligado ao SBC-PT “Caixas”: D0 (sensor) → D3 (SBC)

1. Como criamos cartões distintos para abrir e fechar as portas, tivemos de configurar o RFID reader da saida para apenas reconhecer cada um destes id’s e consoante o valor passado para o SBC este reagir de forma diferente.
   * 1. RFID card abrir id: 1001 ; valor SBC = 8
     2. RFID card fechar id: 1002 ; valor SBC = 0
2. No inicio do loop é logo realizado um GET para saber se o estado da porta se encontra bem, caso for diferente do atual, atualiza para o certo.
3. Depois ao passar os cartões do RFID reader obtemos ações diferentes, mas o algoritmo em si é o mesmo, primerio é lido o valor do RFID reader através de “cartao = analogRead(1)” de meio em meio segundo.
4. De seguida é realizado um GET à API para saber se o estado da porta é igual à ação pretendida por parte do utilizador, caso seja mantém o estado da porta igual e não envia POST. Caso seja diferente, realiza POST com o novo estado da porta, juntamente com a data atual e nome “PortaEntrada”, e atualiza o estado da porta através do customWrite.
5. Como o sensor de movimento encontra-se ligado a um SBC diferente o seu código é distinto. Através da função digitalRead(D3) sabemos se o sensor foi ativado ou não, pois esta função devolve HIGH ou LOW.
6. Se o sensor de movimento for ativado é realizado um GET de modo a saber a quantidade de clientes que saiu da loja, de seguida é aumentado o valor por 1.
7. Depois é realizado um POST para a API de 6 em 6 segundos (sensor de movimento deteta movimento de 5 em 5 segundos) com o nome do sensor de movimento “ClientesSaida” , valor obtido, a data atual e o status do sensor “ON”

* **NOTAS**

1. LCD loja ligado ao SBC – PT “Portas”: D0 (sensor) → D2 (SBC)
2. Atualiza a mensagem do LCD consoante o estado das portas, ou seja é realizado dois GET, um para saber o estado da porta da entrada e outro para a porta da saída
3. Caso tenham sido bem sucedidos e o estado da porta for igual em ambos (p.e aberta – aberta; fechada – fechada), atualiza o estado da loja para o estado em comum das portas, caso sejam distintos mete a mensagem “Chamar Gerente”
4. LCD Caixas ligados ao SBC – PT “Caixas”

LCD Caixa 1: D0 (sensor) → D0 (SBC)

LCD Caixa 2: D0 (sensor) → D1 (SBC)

LCD Caixa 3: D0 (sensor) → D2 (SBC)

1. O código de atualização da mensagem nos LCD’s encontra-se todo no mesmo JavaScript.
2. Primeiramente é realizado 2 GET, um para saber o numero de pessoas que entraram na loja e outro para saber o numero de pessoas que sairam da loja
3. Caso tenham sido bem sucedidos é calculado o numero de pessoas na loja (total = entradas – saidas)
4. Se este valor é menor que 0 ou vazio (“”) é escrita a mensagem em todos os LCD’s “Chamar Gerente”, caso seja maior que 0 e menor ou igual a 10, é escrito no LCD da Caixa 1 “Aberta” e nas restantes “Fechada”, caso seja menor ou igual a 20 e maior ou igual a 10 mantém-se a mensagem no LCD da Caixa 1 e na Caixa 3 e muda a da Caixa 2 para “Aberta”, caso seja maior que 20 todos os LCD’s mostram a mensagem “Aberta”
5. Nota: o algoritmo descrito em cima é feito na dashboard para poder mostrar as mesmas mensagem na parte das Caixas. Adicionalmente a API também faz o mesmo apenas para saber se o valor de clientes da loja é menor que 0, caso seja menor que 0, escreve no ficheiro Notificações/PessoasLoja o valor “1”, caso esteja dentro do normal escreve “0”, para depois mostrar na página de erros e nas notificações o sucedido.

**SENSOR DE FUMO (PKT)**

Na secção de segurança dentro da loja implementamos um sensor de fumo, um sprinkler e uma ventoinha, que reagem consoante o valor lido pelo sensor de fumo

1. Sensor de Fumo e atuadores ligados ao MCU “Fogo Segurança”

Sensor de Fumo: A0 (sensor) → A0 (SBC)

Sprinkler: D0 (sensor) → D0 (SBC)

Ventoinha: D0 (sensor) → D1 (SBC)

1. Como podemos ver pelo anexo, primeiramente lemos o valor do sensor de fumo e enviamos um POST para API, de modo a atualizar o valor na Dashboard
2. De seguida para conseguirmos controlar os atuadores no PKT temos varias condições, se o fumo for menor ou igual a 5 acontece nada, caso seja maior que 5 e menor ou igual a 15, liga-se a ventoinha na velocidade 1, caso seja maior que 15 e menor ou igual a 40 liga-se a ventoinha na velocidade 2 e caso seja maior que 40 liga-se o sprinkler e a ventoinha a nivel 2.

**DASHBOARD**

Na dashboard apenas temos uma forma de interagir com o cenário de teste, que passa por abrir ou fechar tanto a porta de entrada como a de saída.

1. Primeiro criamos dois botões apenas visíveis para o Gerente, que quando clicados é executado um código de JavaScript, que se encarrega por enviar um POST para a API com a ação que se pretende realizar, o nome da porta e a data atual. Código descrito na figura 3

**PYTHON**

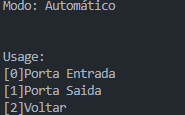
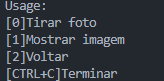
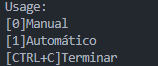
* **Portas**:
  1. Usa-se a mesma base do try except que se usa na figura 2. É perguntao ao user a ação que pretende realizar.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Imagem output PY portas

* 1. Em todas as opções que envolvem alções será feito um POST para a API, com a ação a realizar, a data atual e o nome da porta envolvida. Cujo estado será atualizado tanto na Dashboard como no PKT
* **Câmara**: Código descrito na figura 3



0

1

Figura - Imagem output PY câmara

**NOTAS:**

* Existem dois ficheiros distintos responsáveis para receber o POST com as fotos, o “uploadLog” e “uploadRecentImage”.
* O “uploadLog” recebe a foto com um nome diferente do “uploadRecentImage”, normalmente o nome da foto que o “uploadLog” recebe vem com um número concatenado à frente de modo a poder guardar as imagens para fazer o histório.
  + Imagem 1 – Webcam0
  + Imagem 2 – Webcam1
  + Etc
* Enquanto que o “uploadRecentImage” é responsável por receber a imagem mais recent cujo nome é constante, sempre “Webcam”.
* Esta distinção foi feita de modo a conseguir exibir o histórico das imagens.
* Ambos os ficheiros fazem um controlo do tamanho da imagem máximo 1000KB e do tipo de imagem, só deixam passar se for .png e .jpg

# Resultados obtidos

**TEMPERATURA (PKT)**

* No cenário do PKT configuramos as variáveis da temperatura do congelador e do frigorífico para passar do limite tolerado às 14h00 até às 16h00. O SBC continua a enviar os dados de 2 em 2 segundos para a API como planeado, e o mesmo ativa o cooler do frigorifico e do congelador para baixar a temperatura e o servidor IoT liga os alarmes como previsto. Na dashboard aparece que acontece dois erros e quando clicado neles, é mostrado que a temperatura está demasiado alta tanto no congelador como no frigorifico. Em ambos o caso a data de atualização do último valor encontra-se certa.

**STOCK DA LOJA (SENSORES PESO PKT)**

* No cenário do PKT configuramos as variáveis do peso dos enlatados, frutas e líquidos para passar do limite tolerado por cada um às 15h00 de forma linear. O SBC continua a enviar os dados de cada um dos pesos de 1 em 1 segundo para a API como planeado. Quando os valores se encontram abaixo do limite previsto, na Dashboard é mostrada a mensagem em cada uma das secções “Necessário Repor”. Caso se encontrarem acima do limite é mostrada a mensagem “Em Stock”. Em ambos o caso a data de atualização do último valor encontra-se certa.

**PARQUE DE ESTACIONAMENTO (PKT)**

* No cenário PKT temos a capacidade de aumentar e diminuir o número de pessoas que entraram e saíram do parque através dos sensores de movimento. Primeiramente, quando ambos os valores se encontram a 0, o SBC comunica essse valor, a API recebe o valor e a Dashboard mostra essa informação juntamente com a mensagem “Parque Livre” assim como o LCD do PKT (servomotor posição aberta). Caso aumentemos o valor de entrada para 1, as mensagens de “Parque livre” são mantidas e o valor de entrada na Dashboard aumenta para 1 (O mesmo acontece para a saida) (servomotor posição aberta). Caso aumentemos só o valor de saida para 1 e mantenhamos o valor de entrada a 0, esta informação é atualizada na Dashboard que passa a mostrar a mensagem de erro na página de erros e o LCD do PKT mostra a mensagem “Chamar Gerente” (servomotor posição fechada). Por último em situações extremas em que se encontram 50 pessoas no parque, o LCD mostra a mensagem “Parque Ocupado” (servomotor posição fechada).

**LOJA (PKT / DASHBOARD / PY)**

* No cenário PKT temos a capacidade de aumentar e diminuir o número de pessoas que entraram e saíram da loja através dos sensores de movimento de 6 em 6 segundos. Primeiramente, quando ambos os valores se encontram a 0, o SBC comunica essse valor, a API recebe o valor e a Dashboard mostra a informação do número total de clientes (entradas – saida), e que todas as caixas se encontram fechadas (assim como os LCD’s). Caso aumentemos apenas o valor da entrada para 10 mostra apenas a caixa 1 aberta nos LCD’s e na Dashboard (que também mostra que há dez clientes na loja). Para todas as outras possibilidades foi verificado que aconteceu o planeado (descrito no cenário de teste).
* Tanto na entrada da loja como na saida da loja é necessário usar os cards para fechar ou abrir as portas. Foi verificado que o estado das portas apenas são alterados caso os cartões sejam os de ID certo. Quando uma ação para alterar o estado da porta acontece, o SBC comunica o novo estado à API, cujo estado será atualizado na Dashboard. No caso do código PY e dos butões da Dashboard para abrir e fechar as portas o mesmo acontece, e adicionalmente a porta no PKT também atualiza. Por último o programa para tirar fotos do PY funcionva igualmente bem, caso a opção de modo automático esteja selecionada, quando se abre e fecha a respetiva porta é tirada uma foto que depois irá ser colocado na página das fotos na Dashboard, assim como no histórico das mesmas.
* Por último verificou-se também que quando ambas as portas se encontram abertas é escrito no LCD “Loja Aberta”, se ambas estiverem fechadas “Loja Fechada” e no caso de terem um estado diferente mostra a mensagem “Chamar Gerente”

**SENSOR DE FUMO (PKT)**

* No cenário do PKT configuramos a variável de “Fumo” para tomar valores extremos de uma forma linear começando a subir às 08h00 até às 15h00 onde atinge o máximo, e de seguida começa a estabilizar para um valor normal até às 18h00. Durante este período o MCU envia os valores lidos do Fumo para a API, que são mostrados na Dashboard, às 11h00 o valor do fumo encontra-se a 10 a ventoinha é ligada na velocidade 1, ao 12h00 o valor encontra-se entre 15 e 30 é liga a ventoinha na velocidade 2 e às 13h00 quando atinge o valor 40, liga a ventoinha na segunda velocidade e o sprinkler, e à medida que vai descendo vai desligando os atuadores consoante o valor.

# Conclusão

Apesar de sentirmos que o nosso tema poderia ter sido desenvolvido mais extensamente (controlo de Stock – Segurança da Loja), acreditamos que na solução criada, todos os objetivos foram realizados, e esta seria necessária para o bom funcionamento de um supermercado numa situação real.

Concluindo, consideramos que o trabalhado desenvolvido coincidiu com os nossos gostos sobre a tecnologia, especialmente a parte de automatização de tarefas e criação do Website.

# Bibliografia

Não foi necessário qualquer tipo de investigação para realizar o relatório.

# Anexos

Todos os blocos de código inseridos no trabalho encontram-se como “Objetos” que podem ser abertos com recurso a ficheiros externos, encontrados na mesma pasta do Relatório, apenas adicionamos aqui a forma como encriptamos e desencriptamos valores.

$iv = "1234567891011121";

$key="nPIMLCuuYW";

$cipher = "AES-128-CTR";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*ENCRIPTAR VALORES\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ivlen = openssl\_cipher\_iv\_length($cipher);

$ciphervalor = openssl\_encrypt($\_POST['valor'], $cipher, $key, $options=0, $iv);

$ciphernome = openssl\_encrypt($\_POST['nome'], $cipher, $key, $options=0, $iv);

$cipherhora = openssl\_encrypt($\_POST['hora'], $cipher, $key, $options=0, $iv);

$cipherstatus = openssl\_encrypt($\_POST['status'], $cipher, $key, $options=0, $iv);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*DESENCRIPTAR VALORES\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Exemplo para temperatura do frigorifico:

$valor\_frig = file\_get\_contents("API/Files/TemperaturaFrigorifico/valor.txt");

$valor\_frig = openssl\_decrypt($valor\_frig, $cipher, $key, $options=0, $iv);