Cidade Inteligente

Luís Miguel de Sousa Oliveira - 2221441

Rodrigo Sousa Gomes - 2221455

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, junho de 2023

Índice

Lista de Figuras / Tabelas

Lista de siglas e acrónimos

API	Application Programming Interface
BLE	Bluetooth Low Energy
CSS	Cascading Style Sheets
ESTG	Escola Superior de Tecnologia e Gestão
HTML	HyperText Markup Language
IPL	Instituto Politécnico de Leiria
LED	Light Emitting Diode
MCU	Micro-Controller Unit
RPi	Raspberry Pi
SBC	Single-Board Computer

1. Introdução

Cidade Inteligente

Este é um site dedicado a fornecer informações e serviços para os seguranças de uma determinada cidade (cidade inteligente), sendo também uma plataforma para os residentes da cidade.

A cidade inteligente é um projeto inovador que visa melhorar a qualidade de vida dos seus habitantes através do uso das tecnologias IoT.

A segurança é uma preocupação essencial em qualquer cidade, e a implementação de uma cidade inteligente oferece vantagens significativas nesse aspeto. A utilização de tecnologias avançadas, como sensores e atuadores controláveis, possibilita um monitoramento mais eficiente e uma resposta mais rápida a incidentes. Além disso, a criação de uma plataforma interativa que envolve os residentes da cidade permite a participação ativa da comunidade na segurança pública. Ao fornecer informações e serviços relevantes, o site atua como um canal de comunicação entre os seguranças e os moradores, promovendo uma colaboração mais estreita e uma maior sensação de segurança permitindo o controle da cidade, sendo uma ferramenta valiosa para ajudar os seguranças a manter a ordem e a segurança em toda a área urbana.

O objetivo geral deste trabalho é desenvolver um site dedicado aos seguranças de uma cidade inteligente, fornecendo-lhes ferramentas e informações para melhorar a segurança e o controle da área urbana. Para atingir esse objetivo, os seguintes objetivos específicos foram estabelecidos:

1. Criar um **mapa interativo** da cidade que exiba informações relevantes em tempo real, como o estado da iluminação das ruas, camaras de vigilância, estado de semáforos, entre outros.

2. Utilizar sensores espalhados pela cidade para coletar diversos dados, como níveis de humidade, temperatura do ar, entre outros, e disponibilizar essas informações aos seguranças assim como aos cidadãos da cidade.

3. Permitir o controlo de atuadores, como alarmes, semáforos e luzes, por parte dos seguranças para auxiliar nas ações de segurança e manutenção da cidade.

4. Oferecer acesso ao histórico de informações sobre a cidade aos administradores, permitindo uma análise e investigação mais aprofundada caso necessário.

No desenvolvimento do site, foram empregues diversas tecnologias de desenvolvimento web, como **HTML**, **CSS** e **JavaScript**. **APIs** serão utilizadas para integrar dados provenientes de sensores e controlar atuadores, estes vão estar ligados a controladores. Além disso, serão adotadas práticas de segurança para proteger os dados e garantir a privacidade dos usuários.

Funcionalidades

A Plataforma Web oferece as seguintes funcionalidades:

- Mapa interativo da cidade
- Acesso aos dados obtidos através dos sensores
- Controlo de atuadores de forma manual ou automática (através da dashboard ou dos próprios Arduino e Raspberry)
- Acesso ao histórico
- Imagens em tempo real

2. **Arquitetura**

A cidade inteligente utiliza uma combinação de sensores e atuadores para melhorar a eficiência e a segurança em diferentes situações. No caso específico, são utilizados sensores no Arduino, como o photoresistor para detetar se é dia ou noite, e os sensores de temperatura e humidade para fornecer informações ambientais. Quando o sensor "avoid" no Raspberry Pi deteta a presença de um comboio, ele envia essa informação para o Arduino através da API e aciona o atuador LED RGB (no Arduino) para simular um semáforo, inicialmente acendendo na cor vermelha para indicar que os veículos devem parar. Após o tempo necessário, o semáforo muda para a cor verde, permitindo a passagem. Além disso, o sensor de luminosidade no Arduino é utilizado para verificar a intensidade luminosa do ambiemte. Nesse caso esta informação é enviada através da API para o RPI, e o atuador LED no Raspberry Pi é ativado, fornecendo iluminação nos postes da cidade. Essa integração entre os sensores e atuadores permite uma resposta automatizada e eficiente às condições do ambiente. Com base nos dados coletados, a cidade inteligente é capaz de adaptar-se às necessidades dos seus habitantes, garantindo a segurança no trânsito e uma iluminação adequada em diferentes momentos do dia. Passando toda a informação pela plataforma web apresentando as informações na dashboard.

O tipo de arquitetura do projeto trata-se de uma arquitetura “Device-to-Platform” sendo uma plataforma que não necessita de grande pré-processamento, nem utiliza outro tipo de tecnologia para a comunicação.

Para fornecer ao utilizador uma maior imersão no estado atual da cidade com base nos dados recebidos dos sensores e atuadores, existe também um mapa interativo que apresenta o estado atual dos atuadores, a altura do dia através do sensor de luminosidade, e botões para poder controlar os respetivos atuadores (apenas para seguranças e administradores).

Figura 1 - Mapa Interativo (Butões de Verificação)

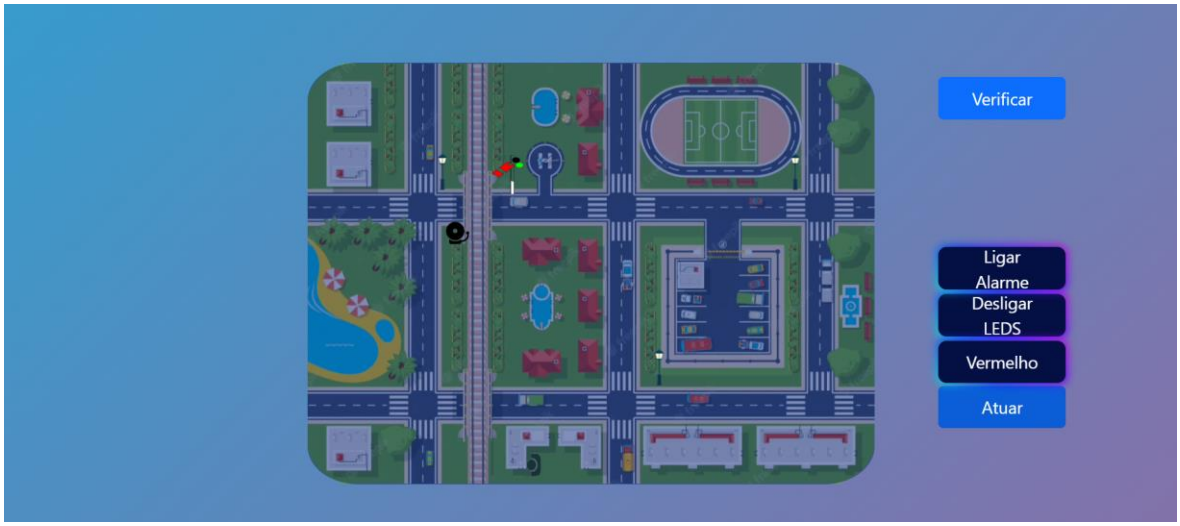

Figura 2 - Mapa Interativo (Atuadores)

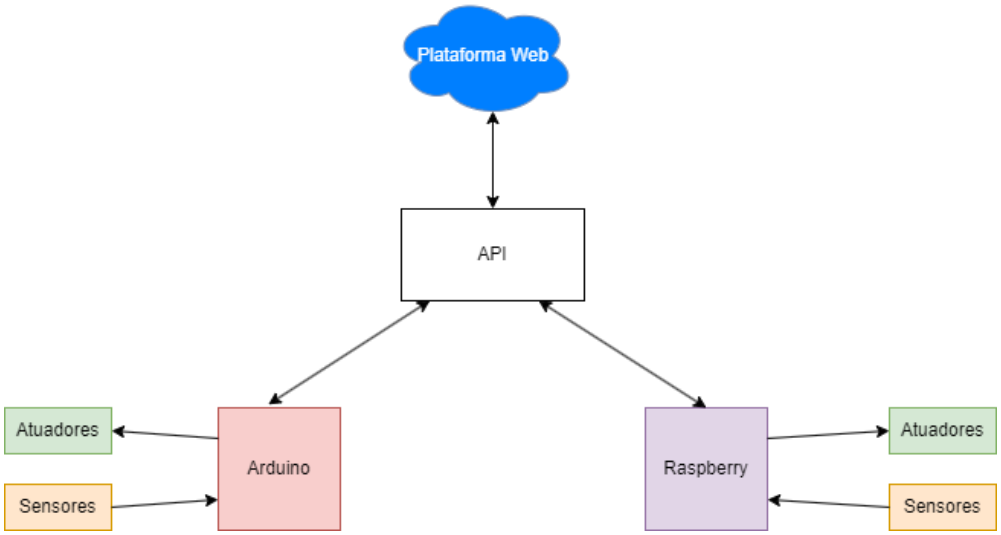

Figura 3 - Diagrama da Arquitetura do Projeto

Lista de equipamentos utilizados:

Arduino:

Sensores:

- **Photoresistor** (Verifica se é de dia ou de noite, o que irá permitir controlar a iluminação)

- **Temperatura** (Lê o valor da temperatura)

- **Humidade** (Lê o valor da humidade)

Aturadores:

- **LED RGB** (Simula o semáforo)

- **Buzzer** (Simula o alarme da cancela)

Raspberry:

Sensores:

- **Avoid** (Deteta a presença do comboio)

Aturadores:

- **LED** (Simula os postes de iluminação)

Para haver este controlo todo de forma rápida e eficaz todos os valores/informações passam pela API e são guardados num ficheiro, permitindo aos administradores aceder a estes mesmos valores através do histórico, e uma vez que o Arduino e o Raspberry Pi necessitam de comunicar, utilizam a API para ler os valores guardados nos ficheiros, temos na figura 2 uma tabela com os eventos e logo a seguir a explicação.

Origem do Evento	Resultado do Evento	Tipo de Comunicação
Recebe um valor do sensor "Avoid"	Ativar o buzzer	SBC (Python) -> MCU
Recebe um valor do sensor "Avoid"	Altera a cor do LED (RGB)	SBC (Python) -> MCU
Recebe um valor do sensor "Avoid"	Altera o valor na dashboard	SBC (Python) -> Dashboard
Recebe um valor via "Bluetooth"	Envia uma imagem para a dashboard	SBC (Python) -> Dashboard
Clique no botão "Atuar Alarme"	Ativar o buzzer	Dashboard -> MCU
Clique no botão "Vermelho/Verde"	Altera a cor do LED (RGB)	Dashboard -> MCU
Clique no botão "Iluminação"	Ativar o LED	Dashboard -> SBC (Python)
Recebe um valor do sensor de luminosidade (Photoresistor)	Altera o valor na dashboard	MCU -> Dashboard
Recebe um valor do sensor de temperatura	Altera o valor na dashboard	MCU -> Dashboard
Recebe um valor do sensor de humidade	Altera o valor na dashboard	MCU -> Dashboard
Recebe um valor do sensor de luminosidade (Photoresistor)	Ativa o LED	MCU -> SBC (Python)

Figura 4 - Tabela de Eventos

1 - **SBC -> MCU** - SBC lê o valor do sensor avoid e faz POST desse estado (0/1) para o servidor assim como o estado do Semáforo, e o MCU faz um GET do estado do semáforo para alternar a cor do LED RGB.

2 – **SBC -> Web** - SBC lê o valor do sensor avoid e faz POST desse estado (0/1) para o servidor e também do valor do Semáforo. A Dashboard lê esse valor com um file_get_contents() e apresenta esse valor realizando as respetivas alterações.

3 - **Web -> MCU** - Botão (“Atuar Alarme” / “Semáforo”) do formulário da Dashboard faz um POST via API para o servidor, e o MCU faz um GET dos valores no servidor para ligar/desligar o buzzer / LED RGB.

4 - **Web -> SBC** - Botão (Iluminação) da Dashboard faz um POST via API (0 ou 1) para servidor, e o SBC faz um GET do valor no servidor para ligar/desligar o LED.

5 - **MCU -> Web** – MCU faz POST via API da temperatura/humidade e luminosidade para o servidor assim como o valor do LED e a Dashboard lê esse valor.

6 - **MCU -> SBC** - MCU faz POST do valor do LED para servidor, e o SBC faz um GET da desse valor via API, ligando e desligando o LED.

3. Implementação

O coração do nosso site é o mapa interativo da cidade, que permite visualizar diferentes áreas e pontos de interesse. Esse mapa é alimentado por uma série de **sensores** e **atuadores** estrategicamente distribuídos pela cidade, fornecendo informações em tempo real para os utilizadores. Existem três tipos de utilizador com diferentes permissões: **utilizador**, **administrador** e **segurança**.

Os **utilizadores** têm acesso a diversas informações relevantes, como por exemplo, a temperatura atual da cidade, que é obtida através de um **sensor de temperatura**. Esse recurso permite que os residentes planeiem as suas atividades de acordo com as condições climáticas. Além disso, temos um **sensor de humidade** que fornece informações sobre a humidade na cidade, sendo possível com isso identificar se está a chover ou não, auxiliando na preparação e na segurança dos cidadãos. Também disponibilizamos informações sobre a **luminosidade**, permitindo que os utilizadores tenham uma maior sensação de ambientação, sendo que o mapa simula o dia e a noite consoante o valor recebido.

Os **Seguranças** desempenham um papel crucial na segurança da cidade, tendo acesso a recursos adicionais que lhes permitam executar a sua atividade, como por exemplo as **luzes**. O facto do sistema estar automatizado permite que eles mantenham a segurança em tempo real sendo outro exemplo, no caso de o **sensor avoid** detetar um comboio, ativará o **buzzer** e colocará o **semáforo** na cor vermelha para garantir a segurança dos cidadãos.

Os **administradores**, por sua vez, possuem permissões especiais e têm acesso abrangente a todas as funcionalidades e recursos do sistema. Eles têm o controle total sobre as operações de segurança da cidade, permitindo-lhes tomar decisões estratégicas e implementar medidas de segurança de forma eficiente. Além disso, os administradores têm acesso privilegiado à gestão e monitoramento de todas as áreas da cidade inteligente, garantindo uma visão completa do sistema e facilitando a tomada de decisões informadas.

Essas funcionalidades combinadas proporcionam uma plataforma poderosa e abrangente para manter a ordem e a segurança em toda a cidade inteligente

Para a segurança dos dados disponibilizados na plataforma, e para o bom funcionamento da mesma, é essencial abordar a hierarquia de permissões como fizemos anteriormente, sendo esta tabela, sendo que esta tabela serve para reforçar o que for referido na página anterior:

Nível de Permissão	Utilizadores	Seguranças	Administradores
Acesso ao mapa	✔	✔	✔
Informações dos Sensores	✔	✔	✔
Controle de Atuadores	✘	✔	✔
Acesso à foto de segurança	✘	✔	✔
Histórico	✘	✘	✔
Monitoramento completo	✘	✘	✔

Sendo assim, agora vamos avançar para a implementação de POSTs, GETs e algoritmos no contexto da integração com dispositivos Raspberry Pi (RPI) e Arduino.

Como foi visto anteriormente o Arduino e o Raspberry Pi implementam os seguintes POSTs e GETs:

ARDUINO

POST (uso da função "post2API" desenvolvida nas aulas laboratorias)

TEMPERATURA

post2API(enviaNomeTemp, enviaTemVal , datahora);

HUMIDADE

post2API(enviaNomeHum, enviaHumVal, datahora);

LUMINOSIDADE

post2API(enviaNomeLum, enviaLumVal, datahora);

LED

post2API(enviaNomeLed , enviaLedVal, datahora);

GET

BUZZER

clienteHTTP.get("/ti/api/api.php?nome=alarme");

LED RGB

clienteHTTP.get("/ti/api/api.php?nome=semaforo");

RASPBERRY PI

POST

SEMÁFORO (LED RGB)

post2API((enviaNomeSem, enviaSemVal)

AVOID

post2API(enviaNomeAvo, enviaAvoVal)

GET)

LED

(através da API)

E para facilitar a compreensão dos algoritmos iremos apresentar 2 fluxogramas o do Arduino e do RPI.

Já vimos anteriormente quais os sensores e atuadores ligados ao Arduino e RPI assim como os respetivos **POSTs** e **GETs** logo irei apresentar os fluxogramas de forma simplificada e clara.

A figura seguinte representa o fluxograma do Arduino:

Onde faz get do valor do Semáforo e buzzer e consoante essas valores altera a cor do LED RGB e liga/desliga o buzzer.

Também faz a leitura dos valores do sensores e faz POST desses valores.

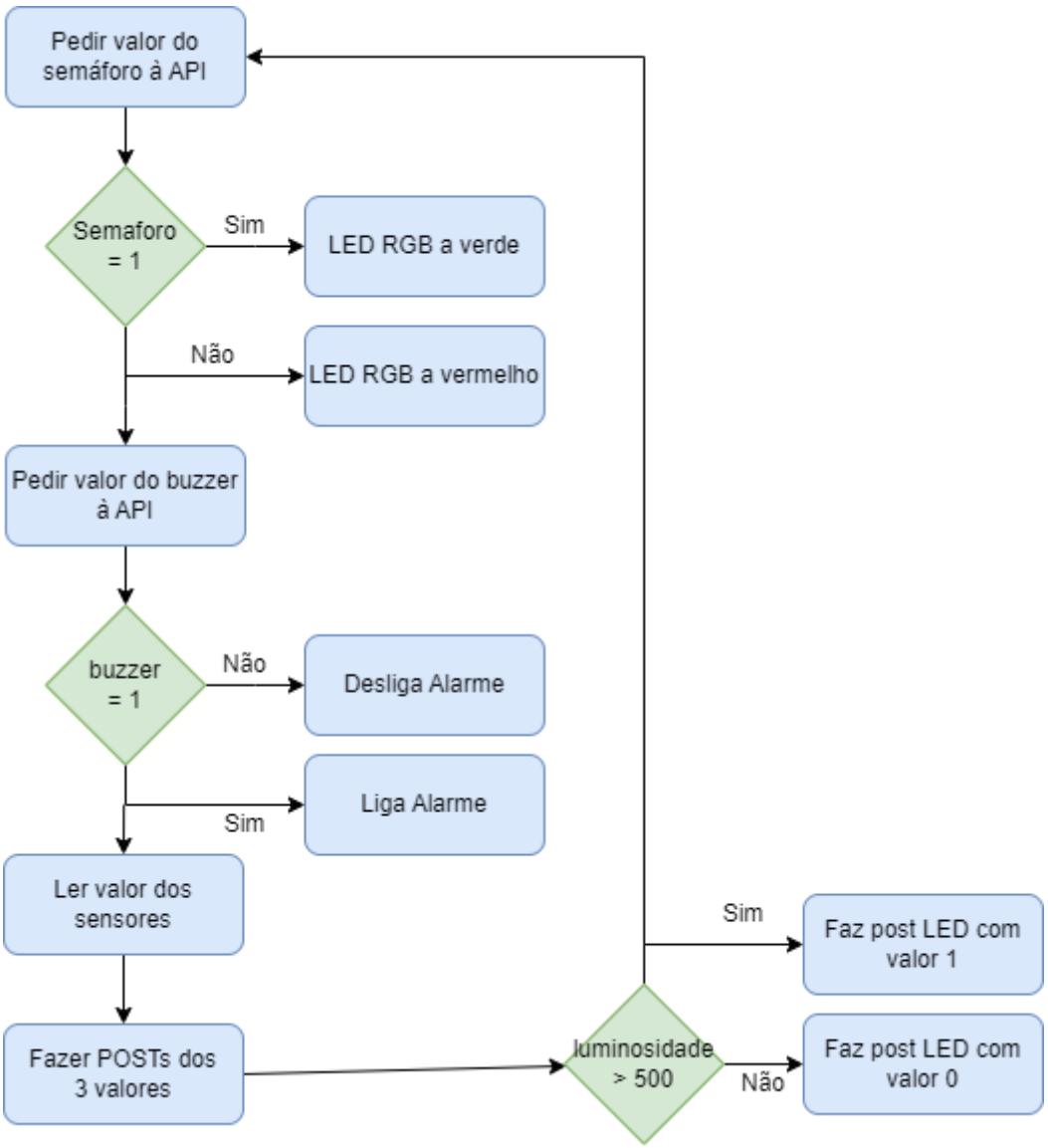

Figura 5 - Fluxograma do Arduino

E a figura seguinte mostra o fluxograma do funcionamento do RPI:

Onde faz GET do valor do LED e consoante esse valor desliga e liga o respetivo LED, também lê o valor do sensor Avoid e faz POST desse valor.

Uma vez que RPI está equipado com Bluetooth quando este detetar um dispositivo irá enviar a imagem.

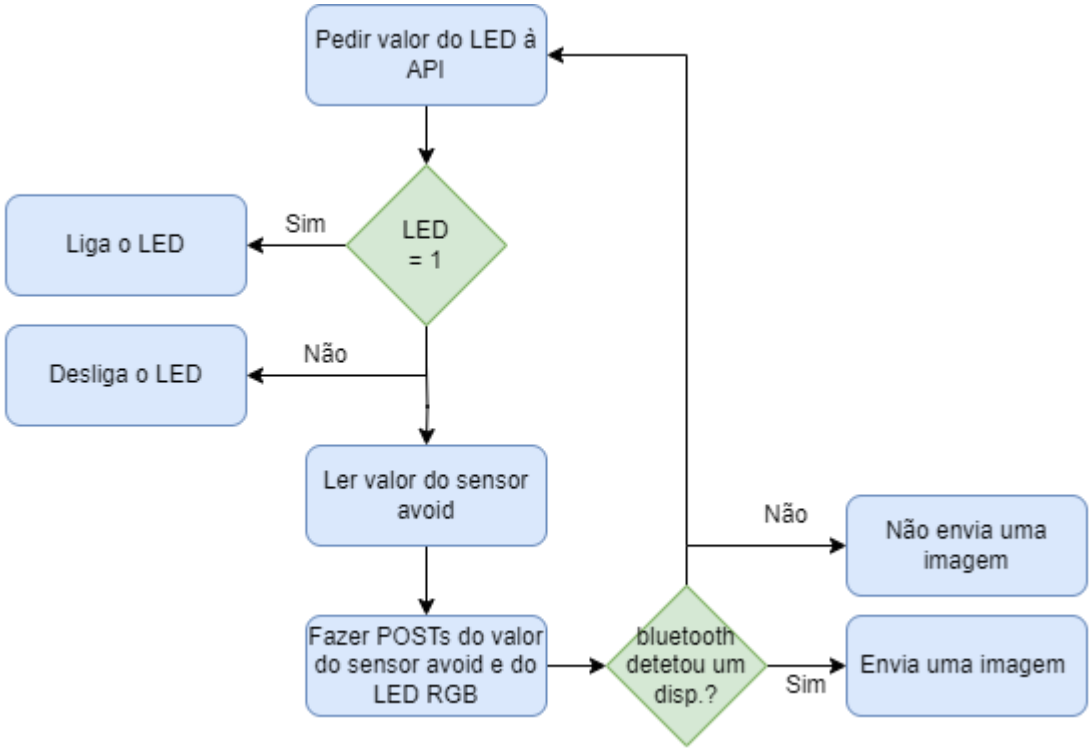

Figura 6 - Fluxograma do Raspberry Pi

4. Cenário de Teste

Uma vez que o projeto está divido em 3 partes, plataforma web, RPI e Arduino vamos explicar de forma mais detalhada e técnica a aplicação de cada um assim como o respetivo código!

Para trabalhar com **Arduino** foi usado o Arduino IDE, e este (o Arduino) faz requisições à API para obter valores do semáforo (LED RGB) e do alarme (buzzer), também lê diversos valores de sensores como temperatura, humidade e luminosidade. Esses valores são então enviados para a API através de requisições POST.

Falando dos pinos temos:

O pino **photocellPin (A0)** é usado para ler a intensidade da luz ambiente através de uma fotocélula.

O **pino do sensor de humidade/temperatura (0)** é usado para obter o valor da temperatura assim como de humidade.

Os pinos **redPin, greenPin e bluePin (11, 10, 9, respetivamente)** são usados para controlar um LED RGB, onde cada pino corresponde a um componente de cor do LED.

O **pino buzzer (8)** é utilizado para controlar um buzzer ativo.

No código do Arduino são definidos nomes para os diferentes tipos de dados que serão enviados para a API, como temperatura, humidade, luminosidade, semáforo, LED e buzzer.

O Arduino também estabelece uma conexão Wi-Fi e aguarda até que a conexão seja estabelecida. Em seguida, é inicializado um objeto HttpClient para realizar as requisições HTTP à API. Dentro do loop principal (void loop()), o código realiza uma requisição GET à API para obter o valor do semáforo e do alarme. Esses valores são armazenados nas variáveis recebeSemVal e recebeBuzVal, respetivamente. Em seguida, o código realiza leituras dos sensores de temperatura, humidade e luminosidade. O valor da luminosidade é lido através do pino photocellPin, enquanto os sensores de temperatura e humidade (DHT11 ou DHT22) através do pino 0. Após a leitura dos sensores, o código chama a função

post2API() para enviar os valores coletados para a API. Dependendo do nome do valor a ser enviado (como LED), o código realiza algumas operações adicionais. Por exemplo, se o nome for "led", ele verifica se o valor é igual a 1 e, em seguida, envia "ON" ou "OFF" para a API em vez de 0 ou 1. Em seguida, o código verifica os valores recebidos do semáforo e do alarme. Se o valor do semáforo for igual a 1, o LED RGB é configurado para a cor verde. Caso contrário, o LED RGB é configurado para a cor vermelha. O buzzer é ligado ou desligado com base no valor recebido do alarme. Por fim, há um atraso de 5 segundos antes do código retornar ao início do loop principal.

Agora a nível do **RPI** este realiza uma série de ações, como deteção de obstáculos usando um sensor (Avoid), controlo de um LED, captura e envio de imagens para uma plataforma usando Bluetooth Low Energy (BLE).

Explicando este código mais profundamente, temos o seguinte. Primeiro importa-se as bibliotecas necessárias, como requests para realizar requisições HTTP, time para lidar com o tempo, RPi.GPIO para controlar os pinos GPIO do Raspberry Pi, datetime para trabalhar com datas e horas, cv2 para capturar imagens da webcam e bluepy.btle para realizar a comunicação BLE. Em seguida, são definidas algumas variáveis, como endURL que contém a URL da API para enviar os valores do LED, e outras variáveis para armazenar os nomes e valores dos dados que serão enviados para a API. A função post2API() é responsável por realizar uma requisição POST para a API com o nome e valor passados como parâmetros. Ela utiliza a biblioteca requests para enviar os dados. Dentro do loop principal (while True), o código realiza a leitura do sensor de obstáculos conectado ao pino GPIO 8 do Raspberry Pi. O valor lido é armazenado na variável enviaAvoVal. Dependendo do valor lido, é exibida uma mensagem indicando se o obstáculo (nesse caso, um comboio) foi detetado ou não. Em seguida, os valores de enviaAvoVal e enviaSemVal são enviados para a API utilizando a função post2API(). Após isso, o código faz uma requisição GET para a URL especificada em endURL para obter o valor do LED. Se a requisição for bem-sucedida, o valor recebido é armazenado na variável valorRecebido. Dependendo desse valor, o LED conectado ao pino GPIO 3 do Raspberry Pi é ligado ou desligado. Em seguida, é definida uma classe Procura que herda da classe bluepy.DefaultDelegate. Essa classe é usada para lidar com a descoberta de dispositivos BLE. No método handleDiscovery(), quando um dispositivo é descoberto

com um sinal RSSI (Received Signal Strength Indication) acima de um determinado limite, a captura de imagem é realizada usando a biblioteca cv2 a partir de uma webcam. A imagem capturada é então enviada para a API por meio de uma requisição POST. Por fim, o código utiliza a classe bluepy.Scanner() para realizar a varredura de dispositivos BLE e chama a função scan() para iniciar a varredura. Os dispositivos encontrados são tratados pela classe Procura. O código possui um loop infinito que só é interrompido quando ocorre uma exceção do tipo KeyboardInterrupt, que é disparada quando o utilizador pressiona CTRL + C. Nesse caso, o código realiza algumas tarefas de limpeza e encerra a execução. Os URLs, endereços IP, pinos GPIO e demais configurações precisam de ser ajustados de acordo com o ambiente e requisitos específicos.

Quando o Raspberry Pi (RPI) deteta a presença de um dispositivo Bluetooth próximo, ele interpreta isso como a entrada de um segurança no gabinete. Como resultado, algumas ações são desencadeadas, incluindo o envio de uma foto da pessoa para a dashboard. Isso proporciona um recurso adicional de segurança e monitoramento.

O processo de comunicação e controlo dos sensores e atuadores entre o Raspberry Pi e o Arduino, é bastante simples.

Sensor Avoid (Avoidance Sensor): O Raspberry Pi lê o estado do sensor avoid conectado a ele por meio do pino GPIO. O valor lido é enviado para a API utilizando a função post2API() com o nome "avoid". O Arduino, por sua vez, pode fazer uma requisição GET à API para obter o valor enviado pelo Raspberry Pi. Com base nesse valor, o Arduino controla o LED RGB. Se o valor for 1 (deteção do objeto), o Arduino aciona o LED RGB para a cor vermelha. Caso contrário, se o valor for 0 (sem deteção do objeto), o LED RGB muda para a cor verde.

Sensor de Luminosidade (LDR - Light Dependent Resistor): O Arduino lê o valor da luminosidade do sensor LDR. Esse valor é enviado para a API utilizando a função post2API() com o nome "luminosidade". O RPI faz uma requisição GET à API para obter o valor de luminosidade enviado pelo Arduino. Com base nesse valor, o RPI pode controla um atuador(LED). Se o valor de luminosidade for acima de um determinado limiar, o RPI liga o LED. Caso contrário, se o valor de luminosidade estiver abaixo do limiar, o LED é desligado.

Sensor do Semáforo (Traffic Light Sensor): Como foi visto anteriormente o RPI efetua o post do valor do sensor Avoid e do valor do semáforo e o Arduino realiza uma requisição GET à API para obter o estado do semáforo. O valor do semáforo recebido é utilizado para controlar o LED RGB no Arduino (semáforo verde/vermelho).

Mas também temos botões na Dashboard havendo um controlo dos atuadores através destes!

O sistema possui três botões na dashboard. Um deles controla o buzzer do Arduino, permitindo ligá-lo ou desligá-lo. Quando esse botão é pressionado, a API envia o valor correspondente (1 para ligado, 0 para desligado) para o Arduino. O Arduino, então, faz uma requisição GET à API para obter o estado do buzzer. Com base nesse valor recebido, o Arduino controla o buzzer. Se o valor for 1, o Arduino aciona o buzzer, produzindo um som. Por outro lado, se o valor for 0, o Arduino desativa o buzzer, interrompendo o som. O segundo botão na dashboard controla o LED do Raspberry Pi, que está associado à luminosidade. Ao ser pressionado, esse botão envia um valor específico para a API, indicando se o LED deve ser ligado ou desligado. O Raspberry Pi realiza uma requisição GET à API para obter esse valor. Com base no valor recebido, o Raspberry Pi controla o LED. Se o valor for positivo (indicando ligar), o LED é ativado, iluminando o ambiente. Caso contrário, se o valor for zero (indicando desligar), o LED é desativado. O terceiro botão na dashboard controla o LED RGB, que funciona como um semáforo e está conectado ao Arduino. Quando esse botão é pressionado, a API envia um valor específico para indicar o estado do semáforo (por exemplo, 1 para verde, 0 para vermelho). O Arduino realiza uma requisição GET à API para obter o estado do semáforo. Com base nesse valor, o Arduino controla o LED RGB. Se o valor for 1, o LED RGB é acionado para exibir a cor verde, indicando que o semáforo está verde. Por outro lado, se o valor for 0, o LED RGB é configurado para exibir a cor vermelha.

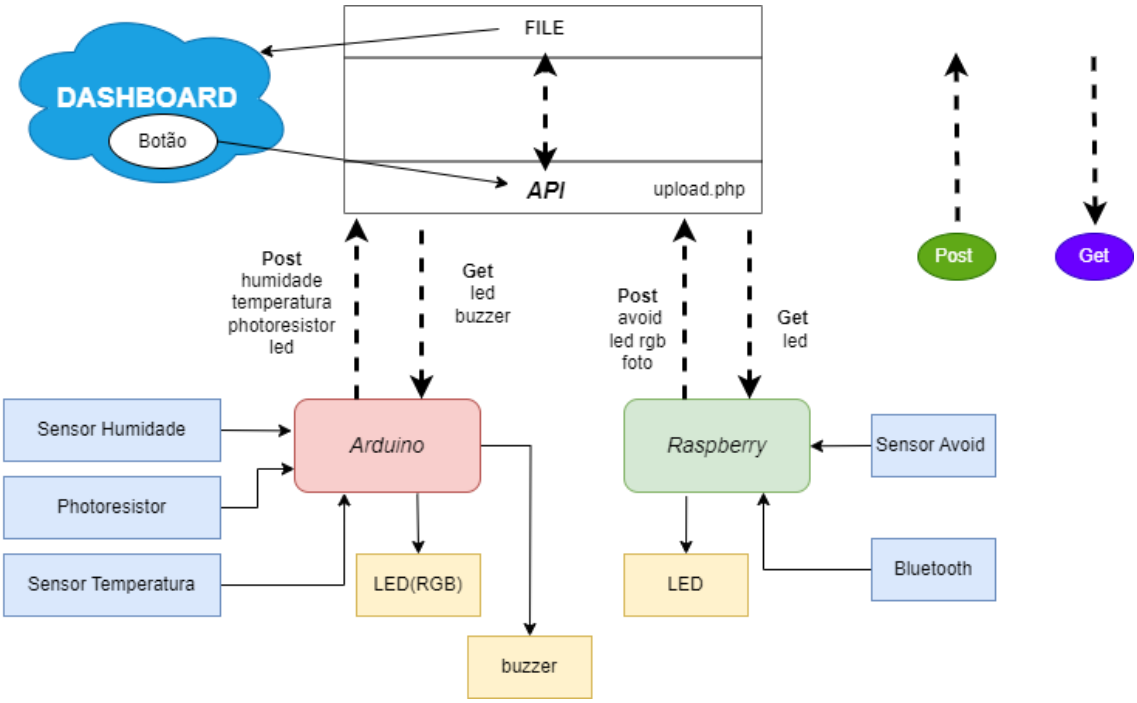

Figura 7 - Diagrama do Cenário de Teste do Projeto

5. Resultados obtidos

Depois de testarmos as funcionalidades da aplicação, chegamos á conclusão que todas elas funcionavam como pretendíamos. Os resultados obtidos nos testes podem ser vis

Teste	Resultado
Raspberry Pi lê o estado do sensor Avoid	O Raspberry Pi leu o estado do sensor avoid corretamente
RPI envia o valor para a API	A dashboard faz get do valor e o histórico lê o valor do ficheiro
Altera o valor na dashboard/histórico	O valor foi atualizado corretamente assim como as respetivas alterações
O valor do sensor altera o valor do semáforo	Esse valor foi alterado
O RPI faz POST desse valor (semáforo)	Valor guardado com sucesso no ficheiro.
Alterações na dashboard...	Alterações feita com sucesso
Arduino faz get do valor	Get feito com sucesso
Altera o estado do LED RGB	LED RGB alternado com sucesso (verde/vermelho)
Arduino lê valor do sensor temperatura	Valor lido com sucesso
Arduino faz post desse valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações (como por exemplo o gráfico)
Arduino lê valor do sensor humidade	Valor lido com sucesso
Arduino faz post desse valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações
Arduino lê valor do sensor luminosidade	Valor lido com sucesso
Arduino faz post desse valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações
Usa esse valor para alterar o valor do LED	Valor do LED alterado corretamente
O RPI faz get desse valor	Get feito com sucesso
Muda o estado do LED	Alterna o estado do LED
Clique no botão "Atuar Alarme"	O buzzer foi ativado após o clique no botão "Atuar Alarme"
Faz post para a API do valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações
Arduino faz get do valor	Get feito com sucesso
Ativar/Desligar buzzer	O buzzer foi ativado/desativa após o clique no botão "Alarme"
Clique no botão "Vermelho/Verde"	A cor do LED foi alterada conforme o botão "Vermelho/Verde"

Faz post para a API do valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações
Arduino faz get do valor	Get feito com sucesso
Altera o estado do LED RGB	LED RGB alternado com sucesso (verde/vermelho)
Clique no botão "Iluminação"	O LED foi ativado após o clique no botão "Iluminação"
Faz post para a API do valor	post feito com sucesso
Atualização desse valor na dashboard e histórico	Valor atualizado e as respetivas alterações
RPI faz get do valor	Get feito com sucesso
Altera o estado do LED	LED ligado / desligado com sucesso

Figura 8 - Tabela dos Testes Realizados

EXTRAS

OUTROS REQUISITOS (até 15%)		
Pedidos assíncronos		✔
Segurança dos dados aplicacionais		✘
Autenticação dos pedidos HTTP na API		✘
Comunicação bluetooth no SBC enquadrada no tema de projeto		✔
Integração de sensores e atuadores virtuais		✘
Utilização de um número de sensores/atuadores reais superior	4+3	✔
Apresentar gráficos dos dados em HTML/CSS/PHP/JavaScript		✔
Plataforma web com 3 utilizadores e com 3 privilégios diferentes	2+1	✔
Utilização de outras bibliotecas Python no SBC		✘
Criação de uma página com histórico das últimas imagens recebidas		✘
Controlo das imagens recebidas		✘
Outras funcionalidades		✔

6. Conclusão

Neste projeto, foi desenvolvido um site dedicado aos seguranças de uma cidade inteligente, com o objetivo de melhorar a segurança e o controle da área urbana. Foram estabelecidos objetivos específicos para alcançar esse objetivo, e as seguintes funcionalidades foram implementadas: mapa interativo da cidade, acesso a informações e dados provenientes de sensores, controlo de atuadores como luzes, acesso ao histórico e imagens em tempo real. A solução desenvolvida demonstra o potencial da implementação de uma cidade inteligente para melhorar a segurança e a qualidade de vida dos habitantes.

Embora a solução desenvolvida tenha alcançado todos os objetivos enunciados, há pontos a serem melhorados. Por exemplo, ter implementado mesmo uma câmara para enviar a foto de quem entra no gabinete. Além disso, é importante considerar a implementação de medidas de segurança adicionais para proteger os dados da plataforma e garantir a privacidade dos utilizadores, sendo um tópico muito importante nos dias de hoje. Em resumo, a implementação de um site dedicado aos seguranças de uma cidade inteligente demonstrou-se eficaz para melhorar a segurança e o controle da área urbana, conseguindo integrar uma plataforma agradável ao utilizador.

7. Bibliografia (Webgrafia)

W3Schools - https://www.w3schools.com/

Arduino - https://www.arduino.cc/