Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Paulo Alexandre Fonseca Pinto

Telmo João Santos Moreira

Domótica como forma de redução de consumos de energia na ESTGV

Relatório de Projeto

Licenciatura em Engenharia Eletrotécnica

Orientador:

Professor António Alberto Ferreira

Mestre Paulo Jorge de Figueiredo Correia

Junho de 2023

RESUMO

A domótica juntamente com o sistema baseado na “internet das coisas” (IoT) permitiram a criação de métodos que ajudam na eficiência energética nos edifícios novos e nos de mais antigos.

O avançar dos anos exigiu uma mudança de atitude relativamente aos recursos energéticos, tornando-se assim, necessário criar condições nos edifícios que proporcionem conforto e bem-estar ao ser humano a baixo custo. No entanto, é necessário ter bastante atenção aos consumos energéticos, pois o consumo exagerado prejudica os recursos do planeta e da humanidade, que dependente bastante da energia, energia essa que para as exigências atuais da sociedade é escassa e com elevado custo, sendo uma das maiores preocupações da sociedade atualmente.

Neste documento serão analisados os temas referentes à domótica e à sua correlação com a “internet das coisas” como também a forma como estes permitem um maior conforto térmico sem a necessidade excessiva de consumos.

Efetuou-se a experiência de controlo sobre o aquecimento num gabinete baseado na comunicação sem fios sendo um sistema que controla vários componentes em simultâneo, componentes esses que interagem entre si de forma remota.

AGRADECIMENTOS

No presente relatório expõe-se a conclusão de uma etapa muito significativa e produtiva para nós. Foi um percurso marcado por intenso trabalho, algumas dúvidas, mas acima de tudo marcado por muita dedicação, sendo possível chegar aqui através do apoio e contributo direto ou indireto de várias pessoas, às quais queremos agradecer.

Começamos por agradecer ao Professor António Ferreira da Escola Superior de Tecnologia e Gestão de Viseu pela disponibilidade, orientação, ideias, sabedoria partilhada e pela paciência demonstrada em todos os momentos em que nos reunimos para debater o ponto de situação do projeto.

Agradecemos também ao Mestre Paulo Correia da Escola Superior de Tecnologia e Gestão de Viseu não só pelas ideias partilhadas, como pelas opiniões e críticas, pela colaboração no solucionar de dúvidas que foram surgindo e como pela ajuda disponibilizada na procura, aquisição e assemblagem do material para este projeto.

A todos os colegas e professores do Departamento de Engenharia Eletrotécnica da ESTGV, pela sua colaboração e ideias partilhadas para que este projeto fosse alcançado com sucesso.

Por fim às nossas famílias e amigos pelo incentivo e por serem modelos de coragem, como também pela ajuda na superação de obstáculos físicos e emocionais que foram surgindo ao longo da realização deste projeto.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

SIGLAS E ABREVIATURAS

AVAC	Aquecimento, Ventilação e Ar Condicionado
CH_PD	*Chip Power-Down*
ESTGV	Escola Superior de Tecnologia e Gestão de Viseu
GPIO	*General Purpose Input Output*
GND	*Ground*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HVAC	*Heating, Ventilating and Air Conditioning*
IDE	*Integrated Development Environment*
IFTTT	*If This Then That*
IoT	*Internet of things*
IP	*Internet Protocol*
LED	*Light-emitting diode*
MIT	Instituto de Tecnologia de Massachusetts
MQTT	*Message Queuing Telemetry Transport*
OSI	*Open Systems Interconnection*
PHP	*Hypertext Preprocessor*
PIR	*Passive Infrared*
PLC	*Programmable Logic Controller*
PRAES	Programa de Apoio a Edifícios mais Sustentáveis
PRR	Programa de Recuperação e Resiliência
RCCTE	Regulamento das Características de Comportamento Térmico dos Edifícios
RDBMS	*Relational database management system*
RST	*Reset*
SCE	Sistema de Certificação Energética dos Edifícios
SMS	*Short Message Service*
SOC	*System On a Chip*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
UPS	*Uninterruptible Power Supply*
URL	*Uniform Resource Locator*
WiFi	*Wireless Fidelity*

1. Introdução

Devido aos conhecimentos adquiridos durante toda a nossa formação académica juntamente com o conhecimento que temos sobre a área prendemo-nos sobre o motivo que leva ao excesso de consumo energético, qual o motivo desse feito e o que se pode mudar para resolver essa situação.

Existem vários motivos para um consumo excessivo, desde os materiais utilizados na construção dos edifícios, passando pelas suas arquiteturas chegando ao fator principal sendo esse o comportamento humano.

Cada pessoa tem uma sensação de conforto diferente, o que influencia bastante o consumo, sendo necessário descobrirem-se formas de ajudar todas as pessoas a sentirem-se confortáveis sem precisarem de pagar fortunas nas faturas.

Sabemos que o nosso projeto não é o mais eficaz, mas, dentro das limitações, conseguirmos tirar o máximo proveito do sistema desenvolvido.

1.1 Eficiência energética

A eficiência energética nos edifícios é de enorme importância quando se fala de sustentabilidade e redução do consumo de energia, consistindo em adotar medidas e tecnologias que permitam otimizar o uso de energia nos edifícios, reduzindo o consumo desnecessário e promovendo o uso eficiente dos recursos disponíveis.

Existem várias maneiras de melhorar a eficiência energética nos edifícios. Uma delas é através da melhoria do isolamento térmico, garantindo que as estruturas dos edifícios estejam bem isoladas para evitar perdas de calor no inverno e ganhos excessivos de calor no verão.

Outra medida importante é a utilização de sistemas de iluminação eficientes, como lâmpadas de LED, que consomem menos energia em comparação com as lâmpadas incandescentes e fluorescentes. Além disso, é essencial promover o uso consciente da iluminação, desligando as luzes quando não estão a ser utilizadas.

Os sistemas de aquecimento, ventilação e ar condicionado (HVAC) também desempenham um papel crucial na eficiência energética dos edifícios. A escolha de equipamentos eficientes e o uso adequado desses sistemas podem reduzir significativamente o consumo de energia. Por exemplo, a regulação adequada dos termostatos e a programação dos HVAC para desligar ou ajustar a temperatura quando não está ninguém no edifício ou no compartimento em questão.

Além disso, a utilização de energias renováveis, como a energia solar fotovoltaica ou a energia eólica, pode contribuir para a eficiência energética de um edifício. A instalação de painéis

solares no telhado pode gerar eletricidade limpa e reduzir a dependência da rede elétrica convencional.

A eficiência energética nos edifícios procura reduzir o consumo de energia, tornando os edifícios mais sustentáveis e economicamente mais viáveis. Através de medidas como o isolamento térmico, iluminação eficiente, sistemas HVAC adequados e o uso de energias renováveis, é possível diminuir o impacto ambiental dos edifícios e promover uma utilização mais racional dos recursos energéticos disponíveis.

É importante mencionar que as normas de eficiência energética podem variar de acordo com os países e regiões. Muitos governos têm estabelecido regulamentos e certificações específicas para promover a eficiência energética nos edifícios.

1.1.1 Eficiência energética em Portugal

Em Portugal, a eficiência energética dos edifícios é responsável pelo consumo de cerca de 30% da energia final [1]. O Decreto-Lei n.º 101-D/2020, de 7 de dezembro, estabelece os requisitos aplicáveis à conceção e renovação de edifícios com o objetivo de assegurar e promover a melhoria do respetivo desempenho energético e regula o Sistema de Certificação Energética dos Edifícios (SCE), transpondo para o ordenamento jurídico nacional as novas diretivas europeias.

O Programa de Recuperação e Resiliência (PRR) conta com 610 milhões de euros para aumentar a eficiência energética dos edifícios em Portugal, dos quais 300 milhões de euros se destinam ao setor residencial, 240 milhões de euros aos edifícios da Administração Pública e 70 milhões de euros a empresas de serviços [2].

Verificando estes factos, verificasse que em Portugal, a eficiência energética nos edifícios tem sido uma área de grande interesse e tem sido feito um esforço significativo para promover a melhoria da eficiência energética em todo o país. De seguida estão enumeradas algumas informações sobre a eficiência energética nos edifícios em Portugal:

1. Certificação Energética: Em Portugal, existe o Sistema de Certificação Energética de Edifícios (SCE), sendo uma certificação obrigatória para todos os edifícios no momento da sua construção, venda ou locação. O SCE classifica os edifícios com base na sua eficiência energética, atribuindo-lhes uma classificação numa escala de A+ (mais eficiente) a F (menos eficiente) e com isso fornecer aos proprietários e inquilinos informações sobre o desempenho energético do edifício e recomendações para melhorias.

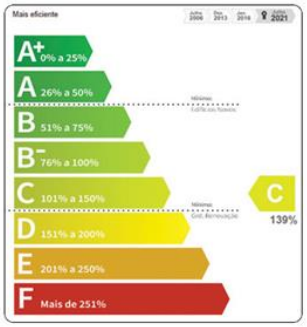

Figura 1 – Certificado Energético [3].

2. Programa de Eficiência Energética em Edifícios da ADENE: A Agência para a Energia (ADENE) é responsável pela implementação do Programa de Eficiência Energética nos Edifícios (RCCTE). Este programa estabelece requisitos mínimos para a eficiência energética nos edifícios novos e existentes, definindo padrões de isolamento térmico, sistemas de climatização e iluminação, entre outros. O objetivo é promover a construção e a reabilitação de edifícios energeticamente eficientes.
3. Incentivos e financiamento: O governo português tem implementado programas de incentivo e financiamento para estimular a melhoria da eficiência energética dos edifícios. O Fundo Ambiental disponibiliza apoios financeiros para projetos de eficiência energética e energias renováveis nos edifícios, tanto para particulares como para empresas. Além disso, existem linhas de crédito e programas de financiamento específicos para a reabilitação energética de edifícios.
4. Energias renováveis: Portugal tem investido no desenvolvimento e na promoção das energias renováveis em edifícios. A utilização de painéis solares fotovoltaicos para a produção de eletricidade e sistemas solares térmicos para aquecimento de água são cada vez mais comuns. Além disso, têm sido incentivados os sistemas de climatização e de aquecimento central que utilizam bombas de calor, que são mais eficientes do ponto de vista energético em comparação com sistemas tradicionais.
5. Reabilitação urbana: O país tem dado ênfase à reabilitação urbana, incentivando a melhoria da eficiência energética nos edifícios existentes. Programas como o Programa de Apoio a Edifícios mais Sustentáveis (PRAES) visam promover a reabilitação energética de edifícios, com apoio financeiro para a implementação de medidas de eficiência energética.

1.2 Motivações e Objetivos

Este projeto situa-se no domínio da eficiência energética que é impulsionada por várias motivações e objetivos, tanto a nível global como a nível nacional.

Uma vez que os custos da energia térmica e as consequências das emissões de CO2, estão cada vez mais elevados, o objetivo principal é reduzir estes dois fatores para uma melhoria não só a nível monetário como também a nível ambiental e segundo um estudo recente (Correia, Paulo "Controlo Baseado Em Tecnologias *IoT* Melhoria da Eficiência Energética e Conforto em Edifícios"), o controlo de temperaturas nos espaços fechados e devidamente automatizados, pode alcançar os 50% significando que podemos ter mais conforto com metade do consumo tornando os nossos objetivos reais.

1.3 Estrutura do relatório

Este relatório está estruturado em 5 capítulos e 14 anexos. Neste primeiro é efetuada um breve enquadramento do tema e do projeto, explicando as motivações que levaram à sua escolha e os objetivos que se pretendem alcançar com a elaboração do mesmo.

No **capítulo 2** falamos sobre a domótica e seus conceitos e também a forma de como esta está bastante ligada à eficiência energética.

No **capítulo 3** são apresentados com detalhe os vários componentes selecionados para a implementação do projeto e explicitados os motivos que justificaram as suas escolhas.

No **capítulo 4** é descrita a seleção e montagem dos componentes e dos processos necessários para a realização do sistema de aquisição e tratamento de dados. Sendo também abordados os procedimentos de teste efetuados para a implementação do sistema.

No **capítulo 5** são expostas as conclusões gerais respetivas ao projeto, aos conhecimentos adquiridos com o desenvolvimento do mesmo e feitas algumas considerações sobre áreas críticas a serem alvo de maior atenção em futuros projetos.

2. Domótica

A domótica é uma das formas de melhorar a eficiência energética e será através dela que iremos desenvolver o nosso projeto.

2.1 Definição

O termo "Domótica" é uma combinação das palavras "*Domus*" (casa) e "Robótica" (controlo automatizado de algo), sendo o conceito de controlo automatizado que otimiza todo o sistema, simplificando a vida diária das pessoas, realizando tarefas rotineiras e tediosas de forma automática e sem preocupações [4].

A domótica surgiu nos anos 80 e, nos últimos anos, tem sido objeto de investigação em várias áreas, como a iluminação, a eficiência energética e o desenvolvimento de novos materiais para construção, entre outras. A pesquisa aplicada na área da domótica utiliza diferentes abordagens, incluindo inteligência artificial, controlo ótimo e técnicas de otimização.

Inicialmente, a ideia da domótica estava principalmente focada no controlo da iluminação, condições térmicas e integração entre vários componentes. No entanto, graças à pesquisa e desenvolvimento realizados, a domótica agora oferece ainda mais benefícios. Além de apresentar circuitos que controlam, verificam e comparam diversas funções, todos com o objetivo de automatizar o ambiente doméstico, também permite o acesso a equipamentos por meio da Internet e a verificação do seu estado em tempo real e dessa forma, o controlo remoto deixa de ser uma utopia para se tornar numa realidade.

2.2 Funcionalidades

Ao planear a instalação de um sistema domótico, é crucial considerar as diferentes funcionalidades que se desejam implementar tendo em conta as necessidades específicas de cada utilizador. Essas funcionalidades podem ser categorizadas em quatro áreas principais: conforto, segurança, eficiência energética e comunicação, desempenhando essas funcionalidades um papel fundamental na gestão e otimização dos recursos disponíveis.

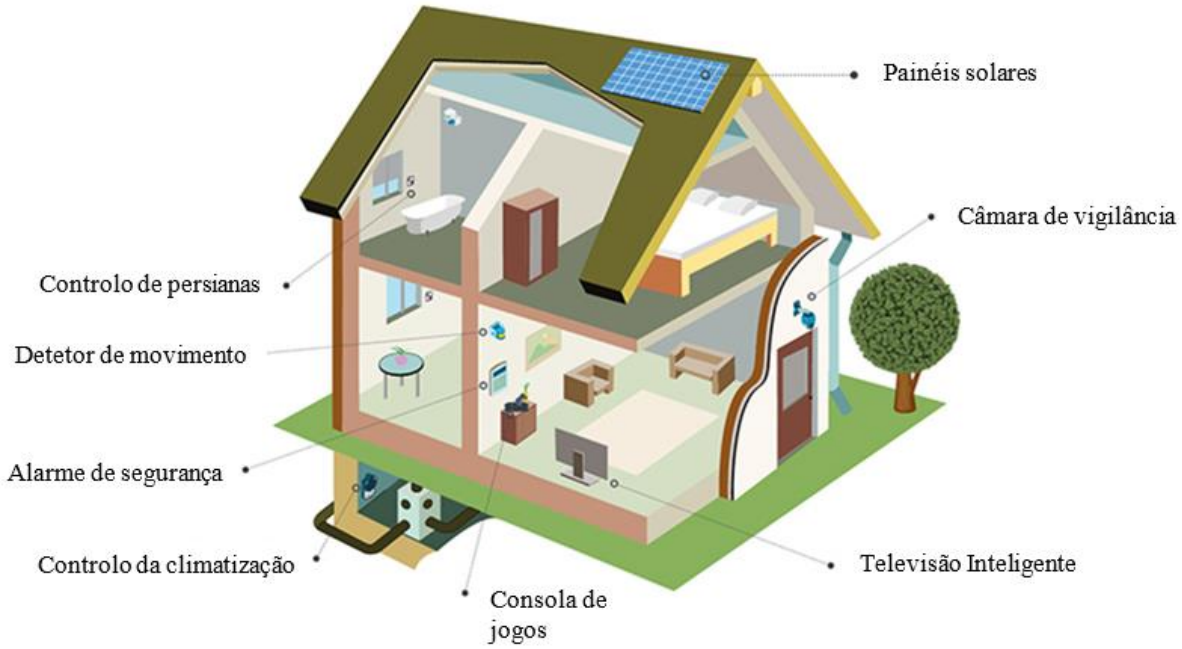

Figura 2 – Casa Inteligente.

2.2.1 Conforto

O objetivo principal é garantir o conforto estando relacionado com as instalações de aquecimento, ventilação e ar condicionado (AVAC). No entanto, também podem ser considerados nessa categoria todos os outros sistemas que possam contribuir para a comodidade e bem-estar. Algumas das funções aplicáveis nesse contexto incluem:

- Controlo do sistema de climatização;
- Controlo da iluminação;
- Automação da rega de jardins;
- Abertura e fecho automático de portas;
- Criação de cenários personalizados;
- Ativação automática de vários sistemas com base nos dados meteorológicos como o fecho de estores e janelas em caso de tempestade ou vento forte;
- Assistente de memória para lembretes, como tomar medicamentos, entre outros;
- Suporte para idosos, pessoas doentes ou com deficiências, proporcionando maior autonomia e acesso à telemedicina.
- Comando de aparelhos de entretenimento (vídeo, áudio e multimédia);

2.2.2 Eficiência energética

A instalação de um sistema domótico tem como um dos conceitos mais importantes a procura pela eficiência e otimização do consumo de energia. Isso não significa eliminar o consumo, mas sim geri-lo e racionalizá-lo, tendo como exemplo a utilização de uma *UPS* no caso da falha de energia. O objetivo é atender às necessidades domésticas com o menor custo possível. Algumas ações relacionadas à eficiência energética incluem:

- Aproveitamento de tarifas reduzidas;
- Deteção de fontes de perdas de energia;
- Tomada de medidas, como o ajuste das persianas para aproveitar a luz solar;
- Monitoramento e registo do consumo para otimizar a eficiência.

2.2.3 Segurança

A preocupação com a segurança está em constante crescimento e cada vez mais as pessoas colocam-na como prioridade máxima. No campo da segurança, algumas tarefas aplicáveis incluem:

- Iluminação de áreas de risco;
- Videovigilância tanto do exterior quanto do interior da residência;
- Criação de alarmes técnicos em situações de emergência ou anormais, tais como:
    - Inundação;
    - Vazamento de gás e água;
    - Falta de energia;
    - Deteção de fumo e fogo;
    - Eletrodomésticos, luzes, aquecedor ou fogão ligados;
    - Portas ou janelas abertas;
    - Alarme médico (monitoramento e diagnóstico remoto de sinais vitais).
    - Desligar automaticamente a água e o gás em caso de fugas;
    - Ativação automática dos serviços de segurança:
        - Alerta aos moradores por meio de SMS, e-mail ou ligação telefónica;
        - Contato com bombeiros e/ou polícia.

2.2.4 Comunicação

As comunicações têm sido alvo de grande investimento. Entre as diversas possibilidades abrangidas nessa área, destacam-se as iniciativas de supervisão, controlo e monitoramento remoto das instalações. Alguns exemplos dessas iniciativas são:

- Controlo e comando da habitação por meio de mensagens SMS;
- Controlo e comando da habitação utilizando uma aplicação para o telemóvel;
- Controlo e comando da habitação via TCP/IP, utilizando um navegador e tecnologias web.

2.3 Protocolos de comunicação

A comunicação requer a presença de um emissor, uma mensagem e um recetor. Um protocolo de comunicação define de maneira precisa o formato da mensagem a ser transmitida e as regras correspondentes de transmissão, incluindo sintaxe, semântica e sincronização.

O modelo OSI (*Open Systems Interconnection*), composto por sete camadas, conforme ilustrado na Figura 3, descreve as várias etapas pelas quais a informação passa até chegar ao seu destino.

No entanto, nem todos os protocolos utilizados na domótica seguem estritamente o modelo OSI. Alguns protocolos possuem apenas as camadas física, lógica e de rede, passando diretamente para a camada da aplicação (camada 7) [5].

Figura 3 – Modelo OSI [6].

No mercado atual, existem dois principais tipos de protocolos de comunicação, que podem ser subdivididos da seguinte forma [10]:

- Protocolos proprietários:
    - VIS;
    - X2D;
    - CAD;
    - entre outros.
- Protocolos normalizados ou abertos:
    - Na Europa:
        - BATIBUS;
        - EIBUS;
        - JBUS;
        - MODBUS;
        - D2B;
        - KNX;
        - PROFINET.
    - Nos Estados Unidos:
        - CEBUS;
        - X-10;
        - SMART HOUSE;
        - *LONWork*s.
    - No Japão:
        - HBS;

Os protocolos abertos são amplamente utilizados nas regiões geográficas mencionadas. No entanto, neste trabalho, apenas serão descritos alguns dos protocolos.

O protocolo X-10 é a tecnologia de comunicação mais antiga, económica e fácil de instalar, amplamente utilizada em ambientes domésticos. O protocolo KNX é o mais utilizado, confiável e robusto. O protocolo *LonWorks* é semelhante ao KNX e frequentemente comparado a ele. Além disso, existem sistemas que utilizam a tecnologia baseada em PLC.

2.4 *Internet of things* (IoT)

A *Internet* das Coisas (IoT) é a tradução direta da expressão em inglês "*Internet of Things*", em português designado de "Internet em Todas as Coisas". Todos utilizamos pelo menos um destes dispositivos: *smartphone*, *tablet* ou computador. No entanto, existem outros aparelhos que também se conectam à internet para realizar atividades específicas. Por exemplo, câmaras de segurança que estão online e permitem que o utilizador monitorize a casa remotamente ou vigie a loja quando estiver fechada. Outro exemplo são as *smart* TVs, podendo aceder a serviços como a *Netflix*, o *YouTube* ou o *Spotify*, sem a necessidade de as ligar ao computador ou telemóvel [7].

Figura 4 – Nuvem IoT [7].

2.4.1 Utilização da IoT

Atualmente, ainda não existe muita procura em se ter uma residência totalmente automatizada e dessa perspetiva, a Internet das Coisas pode não parecer particularmente relevante, no entanto essa procura tem vindo a aumentar nos últimos anos. Em 2022, 38% dos utilizadores individuais de Internet dispunham de algum equipamento de uso pessoal com acesso à Internet [8].

Apesar da palavra domótica levar ao pensamento de automação e casa, é errado pensar que esse conceito é aplicado apenas no ambiente doméstico, existindo aplicações além do lar em que esse conceito pode trazer benefícios de produtividade ou redução de custos de produção, como por exemplo:

- Hospitais e clínicas: os pacientes podem utilizar dispositivos que medem batimentos cardíacos ou pressão sanguínea, e os dados coletados serem enviados em tempo real para o sistema que controla os exames;

- Agropecuária: os sensores espalhados nas plantações podem dar informações bastante precisas sobre a temperatura, humidade do solo, probabilidade de chuvas, velocidade do vento e outras informações essenciais para o bom rendimento da plantação. De igual forma, sensores ligados aos animais conseguem ajudar no controlo do gado: um chip colocado na orelha do boi pode fazer o rastreamento do animal, informar o seu histórico de vacinas e assim por diante;
- Fábricas: a Internet das Coisas pode ajudar a medir, em tempo real, a produtividade das máquinas ou indicar quais setores da planta precisam de mais equipamentos ou suprimentos;
- Lojas: as prateleiras inteligentes podem informar, em tempo real, quando determinado produto está com falta de stock, qual produto está a ser menos vendido (exigindo medidas como reposicionamento físico ou criação de promoções) ou em quais horários determinados itens vendem mais (ajudando na elaboração de estratégias de vendas);
- Transporte público: os utilizadores podem saber, pelo *smartphone* ou em telas instaladas nos pontos de embarque, qual a localização de determinado autocarro. Os sensores também podem ajudar a empresa a descobrir que um veículo apresenta defeitos mecânicos, assim como saber como está o cumprimento dos horários;
- Logística: os dados dos sensores instalados nos camiões, contentores e até caixas individuais combinados com informações do trânsito, por exemplo, podem ajudar uma empresa de logística a definir as melhores rotas, escolher os veículos mais adequados para determinada área, decidir quais encomendas distribuir entre a frota ativa e assim por diante;
- Serviços públicos: os sensores nos caixotes do lixo podem ajudar a empresa de recolha do lixo a otimizar a recolha do lixo. Os carros podem ser ligados a uma central de monitoramento de trânsito para obter a melhor rota para aquele momento, assim como para ajudar o departamento de controlo de trânsito a saber quais estradas na cidade estão mais movimentadas naquele momento.

3. Escolha do material

3.1 Escolha do hardware

Inicialmente estava previsto o uso de uma *RaspBerry P*i para o desenvolvimento do projeto de forma mais autónoma e eficiente. Uma vez que estas não se encontraram disponíveis no mercado, obrigou-nos a encontrar outras soluções, como por exemplo a utilização do microcontrolador ESP8266 ESP-01.

3.1.1 ESP8266 ESP-01

O ESP8266 ESP-01 é um módulo *WiFi* baseado no *microchip* ESP8266, este módulo é um SOC (*System On a Chip*) autónomo que permite a conectividade via *WiFi* a um baixo custo para efetuar o controlo e/ou leitura de dados de dispositivos eletrónicos.

Existem várias versões do ESP8266, sendo possível ter até 9 GPIOs (*General Purpose Input Output*) e com isso permitirmos ao microcontrolador ter acesso à *internet*, no entanto podemos programar o ESP8266 para não ter acesso a uma rede *WiFi* e dessa forma agir apenas como um microcontrolador. Essa característica torna o ESP8266 muito versátil e o poder de economizar dinheiro e espaço em muitos projetos.

O ESP-01 é uma das versões mais simples do ESP8266, tendo apenas 8 pinos disponíveis. Os dois pinos de alimentação e um pino de reset. Uma das suas grandes vantagens prendesse ao seu tamanho compacto, medindo apenas 25,4 mm por 15,2 mm, tornando-o ideal para projetos que exigem uma placa de tamanho reduzido [9].

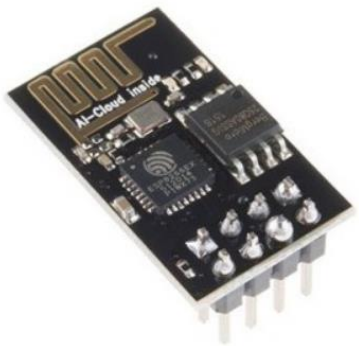

Figura 5 – ESP8266 ESP-01 [10].

O ESP-01 tem uma conectividade *WiFi* de 802.11 b/g/n, com uma antena embutida que tem um alcance de até 100 metros podendo ser programado pela linguagem de programação Lua ou o programa IDE do Arduíno [6]. Também tem suporte para protocolos como TCP/IP, HTTP e MQTT, tornando-o adequado para a construção de aplicações IoT (*Internet of Things*). No entanto, devido ao seu pequeno tamanho e reduzido número de pinos, será mais adequado para projetos simples, como sensores e atuadores.

3.1.1.1 ESP8266 ESP-01 *pinout*

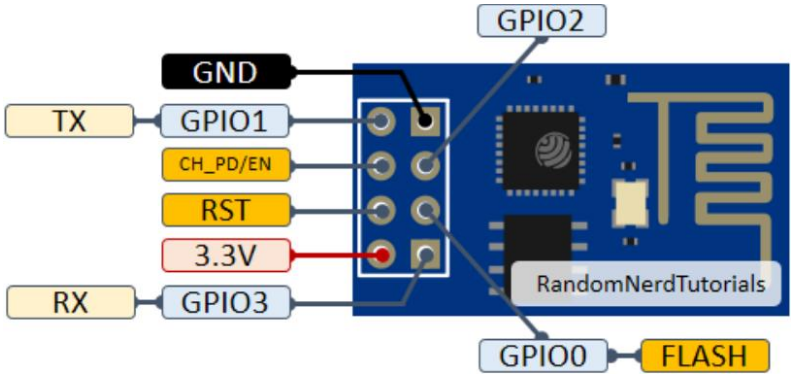

Figura 6 -ESP8266 ESP-01 *pinout* [12].

- GPIO0: Pino de entrada/saída digital

O pino GPIO 0 é um pino de entrada/saída configurável, o que significa que pode ser programado para funcionar como entrada ou saída de dados, dependendo das necessidades e suportando um sinal de nível lógico de 3,3V, portanto, é importante garantir que não são aplicadas tensões superiores a esse limite, pois isso pode danificar o módulo.

Durante a inicialização do ESP-01, o estado do pino GPIO 0 é muito importante, pois determina o modo de inicialização do módulo. Existem duas configurações principais:

- Modo normal: Quando o pino GPIO 0 está definido como alto (3,3V), o ESP-01 inicializa no modo normal e executa o *firmware* do utilizador. Nesse modo pode-se usar o pino GPIO 0 como um pino de entrada ou saída normal.
- Modo de programação: Quando o pino GPIO 0 está definido como baixo (0V), o ESP-01 entra no modo de programação permitindo o carregamento do *firmware* para o módulo por meio da porta TX ou RX. No modo de programação, o pino GPIO 0 deve ser mantido em nível baixo para que o módulo possa receber novos programas.

É importante mencionar que, ao utilizar o pino GPIO 0 como entrada ou saída, deve-se levar em consideração a sua função durante a inicialização. Se pretendermos usar o pino GPIO 0

como uma entrada/saída normal, é recomendável deixá-lo desligado ou ligado a um nível lógico alto (3,3V) durante a inicialização para evitar entrar acidentalmente no modo de programação.

- GPIO2: Pino de entrada/saída digital

Assim como o GPIO 0, o pino GPIO 2 pode ser configurado como entrada ou saída, dependendo das necessidades do projeto.

O pino GPIO 2 também suporta sinal de nível lógico de 3,3V, portanto, não se podem aplicar tensões superiores a esse limite para evitar danos ao módulo.

No entanto, ao contrário do pino GPIO 0, o pino GPIO 2 tem uma funcionalidade especial durante a inicialização do ESP-01, já que durante a inicialização, se o pino GPIO 2 estiver definido como baixo (0V), o ESP-01 entra no modo de inicialização do *bootloader*. Nesse modo, o ESP-01 aguarda comandos de atualização de *firmware* por meio da porta TX RX, portanto, é importante manter o pino GPIO 2 a um nível baixo se for pretendido realizar uma atualização de *firmware*.

Após a inicialização pode-se usar o pino GPIO 2 como um pino de entrada ou saída normal, assim como o GPIO 0. No entanto, é necessário ter cuidado ao usar o pino GPIO 2 como saída, pois ele está ligado a uma resistência *pull-up* interna de 10kΩ. Isso significa que, se usarmos o pino GPIO 2 como saída e fornecermos um sinal lógico baixo, pode ser necessário ligar uma resistência externa para ajudar a baixar o sinal de forma mais eficiente.

- RX: Pino de entrada do sinal

O pino RX é usado para receber dados na interface UART (*Universal Asynchronous Receiver-Transmitter*). É por meio deste pino que o módulo ESP-01 recebe comandos, dados ou informações enviadas pelo dispositivo nele ligado.

É importante verificar que o nível lógico do pino RX é de 3,3V. Portanto, ao se ligar o ESP-01 a outro dispositivo, é necessário nos certificarmos que o nível lógico seja de 3,3V para evitar danos.

Além disso o pino RX, é importante considerar a sua interação com o modo de inicialização e programação do ESP-01. Durante a inicialização, o pino RX é usado para receber comandos de atualização de *firmware*, quando combinado com os pinos GPIO 0 e GPIO 2 em configurações específicas. Portanto, é necessário ter-se cuidado ao usar o pino RX simultaneamente para receber dados de um dispositivo externo enquanto realiza operações de inicialização ou programação.

- TX: Pino de saída do sinal

O pino TX é usado para enviar dados na interface UART (*Universal Asynchronous Receiver-Transmitter*). É por este pino que o módulo ESP-01 envia comandos, dados ou informações para o dispositivo ligado a ele.

É importante observar que o nível lógico do pino TX é de 3,3V. Portanto, ao se ligar o ESP-01 a outro dispositivo, tem de se certificar de que o nível lógico seja de 3,3V para evitar danos ao módulo.

Assim como o pino RX, ao utilizar o pino TX, é importante considerar a interação com o modo de inicialização e programação do ESP-01. Durante a inicialização, o pino TX é usado para enviar respostas, dados ou informações durante o processo de inicialização ou programação do módulo. Portanto, é necessário ter cuidado ao usar o pino TX simultaneamente para enviar dados para um dispositivo externo enquanto se realizam operações de inicialização ou programação.

- CH_PD: Habilita ou desabilita o funcionamento

O pino CH_PD (*Chip Power-Down*) é usado para habilitar ou desabilitar o chip do ESP8266 ESP-01.

Para que o chip funcione corretamente, o pino CH_PD precisa ser mantido a um nível lógico alto (3.3V), ou seja, precisa ser ligado a uma fonte de alimentação adequada juntamente com o pino VCC.

Se o pino CH_PD for mantido a um nível lógico baixo (0V), o chip ESP8266 permanecerá desabilitado e não funcionará corretamente, logo é importante garantir que o pino CH_PD esteja sempre ligado a uma fonte de alimentação adequada para garantir que o chip esteja habilitado e funcione corretamente.

Além disso, é importante lembrar que o ESP8266 ESP-01 tem uma corrente máxima de operação limitada, portanto é necessário garantir que a fonte de alimentação forneça corrente suficiente para o chip e outros componentes que possam estar ligados.

- RST: Pino de Reset

O pino RST (*Reset*) do ESP-01 é responsável por reiniciar o módulo ESP-01.

Ao se aplicar um pulso de nível baixo (0V) no pino RST, o módulo ESP-01 é reiniciado e passa pelo processo de inicialização, retornando ao seu estado inicial. Esta ação pode ser útil em várias situações, quando é necessário redefinir-se o módulo para restaurar as configurações padrão ou para corrigir problemas de operação.

- GND: Pino de terra

O pino GND (*Ground*) do ESP-01 é o pino de referência de terra do módulo, fornecendo uma ligação comum para o potencial de terra, que é a referência elétrica para todos os outros componentes do circuito.

Este é ligado à terra do sistema e deve ser ligado a outros componentes que requerem referência comum de potencial, como outros dispositivos, sensores ou fontes de alimentação. É importante garantir que todos os componentes compartilhem o mesmo ponto de referência de terra para que os sinais elétricos sejam transmitidos corretamente e haja uma operação confiável do circuito.

No ESP-01, o pino GND é geralmente ligado à terra da fonte de alimentação, seja ela uma bateria, uma fonte de alimentação externa ou outro componente do circuito. É importante observar que o ESP-01 opera em um nível lógico de 3,3V, portanto, é necessário certificar-se de que a fonte de alimentação e os outros componentes estejam também alimentados a esse nível lógico para evitar danos ao módulo.

Ao projetar ou montar circuitos com o ESP-01, é recomendável ligarem-se todos os pinos GND relevantes aos pontos de referência de terra apropriados para garantir um funcionamento adequado e seguro do sistema.

- VCC: Pino de Alimentação

O pino VCC do ESP-01 é o pino de alimentação do módulo. Este fornece a tensão de alimentação necessária para o bom funcionamento do ESP-01.

O ESP-01 opera com uma tensão de alimentação de 3,3V. Portanto, o pino VCC deve ser ligado a uma fonte de alimentação de 3,3V para garantir o funcionamento adequado do módulo. É importante verificar que o ESP-01 não suporta uma tensão de alimentação de 5V, portanto, não se pode ligar diretamente a uma fonte de 5V, pois isso pode danificar o módulo.

Ao ligar-se o pino VCC, é necessário fornecer uma alimentação estável e adequada. Recomenda-se o uso de uma fonte de alimentação regulada de 3,3V ou um regulador de tensão de 3,3V para fornecer a tensão de alimentação ao ESP-01. A fonte de alimentação deve possuir a capacidade de fornecer corrente suficiente para suprir as demandas do módulo, especialmente durante transmissões *WiFi* ou outros períodos de alto consumo de energia.

Além disso, é importante mencionar que o pino VCC pode ser usado para fornecer energia a outros componentes do circuito, desde que estejam operando em uma tensão de 3,3V. No entanto, é necessário considerar a capacidade da fonte de alimentação em fornecer energia suficiente para todos os componentes ligados.

3.1.2 Controladores

3.1.2.1 *Shield* DHT11 ESP-01

O *Shield* DHT11 ESP-01 é um módulo IoT que juntamente com um microcontrolador ESP8266 permite medir a temperatura e humidade relativa do ar em tempo real e enviar esses dados para a Internet através da rede *WiFi* existente no local.

O *Shield* DHT11 ESP-01 permite o controlo automático de sistemas de aquecimento e refrigeração baseado na temperatura ambiente e na recolha de dados climáticos para a análise e previsão do tempo [13].

Figura 7 -*Shield* DHT11 [13].

Algumas vantagens do *Shield* DHT11 ESP-01 são:

- Baixo custo sendo uma opção acessível para medir a temperatura e a humidade do ar em projetos de IoT;
- Fácil utilização permitindo a sua integração em projetos de IoT;
- Precisão na medição é o suficiente para muitas aplicações;
- Ligação *WiFi* permite que os dados de temperatura e humidade sejam enviados para a *Internet*;
- Pouco consumo de energia sendo isso bastante importante para projetos que dependam de baterias.

Algumas desvantagens do *Shield* DHT11 ESP-01 são:

- Limitações do sensor sendo este não tão preciso quanto outros sensores de temperatura e humidade disponíveis no mercado;
- Ligação *WiFi* instável em alguns casos podendo afetar a confiabilidade dos dados coletados;

- Interface limitada: o *Shield* DHT11 ESP-01 tem uma interface limitada para ligação com outros dispositivos, o que pode ser um problema em projetos mais complexos;
- Dificuldade de leitura do sensor: o sensor DHT11 requer uma leitura precisa para fornecer dados precisos, e pode ser sensível a interferências elétricas.

No geral, o *Shield* DHT11 ESP-01 é uma opção interessante para projetos de IoT que exigem a medição da temperatura e humidade do ar, e oferece uma boa relação custo-benefício. No entanto, é importante considerar as limitações do sensor e as possíveis instabilidades da sua ligação WiFi ao escolher este módulo para um projeto específico.

O DHT11 possui um invólucro com quatro pinos: VCC, Data, NC (não ligado) e GND. Este sensor utiliza um termístor para medir a temperatura e um sensor capacitivo para medir a humidade.

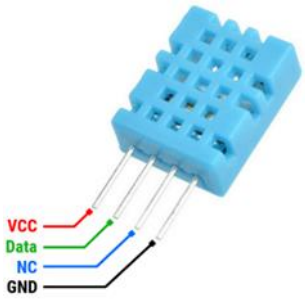

Figura 8 -DHT11 [14].

O sensor possui um pino VCC que deve ser ligado a uma fonte de tensão adequada para o seu funcionamento. Geralmente, esse pino é ligado a um dos pinos de alimentação do microcontrolador, que pode fornecer 3,3V ou 5V, dependendo do modelo. O pino VCC é responsável por fornecer energia ao circuito interno do sensor e permitir a sua comunicação com o microcontrolador.

O pino Data é o pino de dados do sensor, usado para transmitir as leituras de temperatura e humidade para o microcontrolador. Este comunica usando um protocolo de comunicação unidirecional, onde o microcontrolador solicita os dados e o sensor responde com as leituras.

O protocolo de comunicação unidirecional é um método simples e eficiente de troca de informações entre o sensor e o microcontrolador. O microcontrolador envia um sinal inicial para o sensor, que envia os dados em forma de impulsos. O microcontrolador lê os impulsos e converte-os em valores de temperatura e humidade. O sensor não recebe nenhum sinal do microcontrolador, apenas envia os dados quando solicitado.

O pino NC não é usado no sensor DHT11 e não precisa ser ligado a nada.

O pino GND é um dos pinos do sensor que serve para estabelecer uma referência comum de tensão entre o sensor e o microcontrolador, devendo ser ligado ao terminal de terra do microcontrolador para garantir o funcionamento correto do sensor.

Para utilizar o DHT11 com o ESP-01 é necessário ligar os pinos do sensor aos pinos correspondentes do módulo ESP-01.

Além da ligação física, também é necessário escrever o código para o ESP-01 ler os dados do sensor DHT11. Existem bibliotecas disponíveis para facilitar a leitura dos dados do sensor, como a biblioteca DHT.h, pois esta biblioteca fornece funções que estão preparadas para solicitar os dados do sensor e interpretá-los corretamente [14].

3.1.2.2 ESP-01S Relay v4.0

O ESP-01S *Relay* v4.0 é um tipo de módulo de relé que juntamente com o módulo *WiFi* ESP-01 é um relé de canal único numa pequena placa. É um módulo *WiFi* de baixo custo e fácil de usar, permitindo que o relé seja controlado remotamente usando um smartphone ou um computador.

A versão v4.0 deste módulo de relé tem um botão de reset que permite reiniciar o funcionamento do mesmo.

Tem uma entrada de 5V DC para a alimentação do circuito de comando. Já o circuito de potência tem uma corrente máxima de 10A, permitindo o seu uso para ligar/desligar vários dispositivos, como luzes, motores e eletrodomésticos [15].

Figura 9 – Shield relé ESP8266 ESP-01 [16].

Algumas vantagens do ESP-01S *Relay* v4.0 são:

- Baixo custo e acessível para automação residencial e controlo industrial.
- Fácil de usar, programar e pode ser integrado em várias plataformas e aplicativos IoT.
- Controlo remoto que permite o seu a sua manipulação através de um smartphone ou computador.
- Suporte MQTT que facilita a sua integração com outros dispositivos.
- Botão de reset facilita a manutenção e resolução de problemas.

Algumas desvantagens do ESP-01S *Relay* v4.0 são:

- Limitação do número de canais o que impede o controlo simultâneo de vários dispositivos
- Potência limitada não permitindo o seu uso com dispositivos que requerem mais potência.
- Limitação do alcance devido à qualidade da ligação *WiFi*.

No geral, o ESP-01S *Relay* v4.0 é uma opção econômica e fácil de usar para controlar dispositivos remotamente, mas suas limitações devem ser levadas em consideração ao decidir se é a melhor opção para um determinado projeto.

3.1.2.3 Sensor de movimento PIR HC-SR501

O sensor de movimento PIR HC-SR501 é um sensor de presença que deteta movimentos baseados na variação da luz infravermelha emitida pela radiação do corpo humano, sendo bastante utilizado em projetos que necessitem de detetar a entrada de uma pessoa numa determinada área. O sensor funciona através da deteção de energia na forma de radiação infravermelha o que permite a deteção de movimentos até uma distância de 7 metros. É possível ajustar a duração do tempo de espera para a estabilização e a sensibilidade do PIR através do potenciômetro localizado na placa.

A sigla PIR significa *Passive Infrared*, o que significa que o sensor deteta a radiação infravermelha emitida por objetos em movimento. Os sensores PIR são muito usados pois são altamente sensíveis a mudanças de calor no ambiente, o que permite detetar a presença de pessoas ou animais.

O HC-SR501 é um módulo compacto que contém o sensor PIR, circuitos de processamento de sinal e ajustes de sensibilidade [17].

O módulo possui três pinos principais:

- VCC (5V): Pino de alimentação para o módulo.
- GND (*Ground*): Pino de terra para o módulo.

- OUT: Pino de saída digital que fornece um sinal alto (*HIGH*) quando o movimento é detetado e um sinal baixo (*LOW*) quando nenhum movimento é detetado.

O módulo HC-SR501 também possui dois potenciômetros para ajustes:

- Sensibilidade: Permite ajustar a sensibilidade do sensor para detetar movimentos a diferentes distâncias. Girando o potenciômetro no sentido horário, a sensibilidade aumenta.
- Tempo de atraso: Define o tempo de espera após a deteção de movimento antes de o sinal de saída voltar a ser baixo. Girando o potenciômetro no sentido horário, o tempo de atraso aumenta.

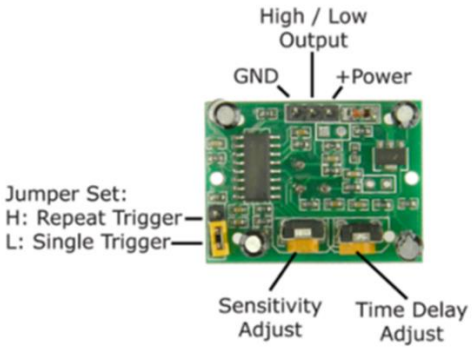

Figura 10 -PIR HC-SR501 *pinout.*

Ao detetar movimento, o sensor PIR aciona o sinal de saída, que pode ser ligado a um microcontrolador, para realizar ações específicas, como acender luzes, disparar um alarme ou gravar um vídeo.

O sensor PIR HC-SR501 é sensível ao calor e ao movimento, mas não deteta cores nem fornece informações sobre a forma ou tamanho do objeto em movimento. Portanto, é ideal para detetar a presença de pessoas ou animais, mas não é adequado para tarefas como reconhecimento facial ou deteção precisa de objetos.

Na Figura 11 estão as especificações do PIR HC-SR501.

Product Type	HC--SR501 Body Sensor Module
Operating Voltage Range	5-20VDC
Quiescent Current	<50uA
Level output	High 3.3 V /Low 0V
Trigger	L can not be repeated trigger/H can be repeated trigger(Default repeated trigger)
Delay time	5-300S(adjustable) Range (approximately .3Sec -5Min)
Block time	2.5S(default)Can be made a range(0.xx to tens of seconds
Board Dimensions	32mm*24mm
Angle Sensor	<110 ° cone angle
Operation Temp.	-15-+70 degrees
Lens size sensor	Diameter:23mm(Default)

Figura 11 - Especificações PIR HC-SR501.

3.2 Escolha do *software*

3.2.1 Arduíno IDE

O *software* Arduíno é uma plataforma de desenvolvimento *open-source* que permite a programação e o controlo de microcontroladores compatíveis, sendo um programa fácil de usar e bastante popular entre estudantes, entusiastas e profissionais em eletrónica.

O *software* Arduíno é baseado na língua de programação C/C++ e exibe uma interface gráfica intuitiva para escrever e carregar código nos dispositivos, permitindo que se criem projetos interativos, controlando sensores, atuadores e outros componentes eletrónicos.

Existem algumas versões do software arduíno disponíveis, mas a que mais se usa é o Arduíno IDE (*Integrated Development Environment*). O Arduíno IDE é uma aplicação para computador que fornece todas as ferramentas necessárias para escrever, compilar e carregar códigos nos dispositivos. Além do Arduíno IDE, também existem outras opções de *software* para programar placas Arduíno, como o Arduíno *Web Editor*, que permite programar diretamente no navegador, e outras IDEs de terceiros que oferecem recursos adicionais, como suporte a depuração avançada e integração com outras plataformas.

O IDE possui recursos como destaque de sintaxe, auto compilar, verificação de erros e uma biblioteca extensa com funções pré-definidas para facilitar a programação. Além disso, oferece uma interface de comunicação serial para monitorar e depurar o comportamento do dispositivo nele ligado.

Uma das principais vantagens do *software* Arduíno é a vasta comunidade de utilizadores e desenvolvedores que o suportam. Existe uma enorme quantidade de documentação, tutoriais e exemplos disponíveis online, o que facilita a aprendizagem e o desenvolvimento de projetos. Além disso, existem muitas bibliotecas de terceiros que podem ser adicionadas ao *software* para expandir as suas capacidades [18].

Escolhemos usar a versão para computador pois acaba por ser mais completa e intuitiva.

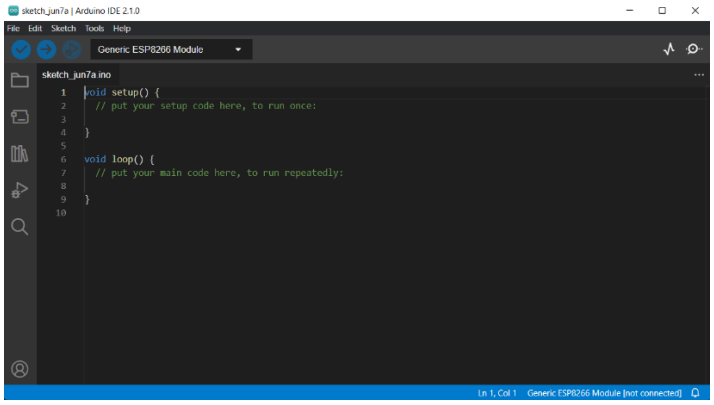

Figura 12 -Interface Arduíno IDE.

3.2.2 *Cayenne myDevices*

O *Cayenne myDevices* é uma plataforma de IoT que permite ligar, monitorar e controlar dispositivos IoT de forma maus acessível e intuitiva.

A proposta principal do *Cayenne myDevices* é simplificar o processo de desenvolvimento e implantação de projetos de IoT, oferecendo vários recursos para ligar e gerir dispositivos IoT, coletar dados, criar painéis de controlo personalizados e automatizar ações com base nos dados coletados, sendo uma solução adequada tanto para desenvolvedores experientes quanto para utilizadores com pouco conhecimento na programação [19].

As características mais importantes do *Cayenne myDevices* são:

- Conectividade: A plataforma suporta uma ampla gama de protocolos e dispositivos IoT populares, como Arduíno, *Raspberry Pi*, ESP8266, entre outros.
- Interface visual: O *Cayenne* tem uma interface gráfica intuitiva que permite criar e personalizar painéis de controlo para visualizar e interagir com os dados dos dispositivos ligados, adicionando *widgets*, gráficos, botões e outros elementos para exibir informações relevantes e controlar dispositivos remotamente.
- Automação: A plataforma permite criar regras e cenários de automação com base nas condições predefinidas, podendo-se configurar uma regra para ligar automaticamente uma lâmpada quando um sensor de movimento for acionado, como exemplo.
- Integração com serviços de terceiros: Permite a integração com uma variedade de serviços e plataformas populares, como Amazon Alexa, IFTTT e MQTT, alargando a funcionalidade da plataforma, integrando-a a outros sistemas e serviços.
- Gestão de muitos dispositivos: A plataforma oferece recursos para gerir e monitorar um elevado número de dispositivos IoT em simultâneo, sendo bastante útil em casos em que é necessário implantar e controlar uma rede de dispositivos em escala.

- Compartilhamento de projetos: O *Cayenne myDevices* têm a opção de compartilhar os projetos com a comunidade, permitindo que outras pessoas explorem e se inspirem em soluções criadas por outros.

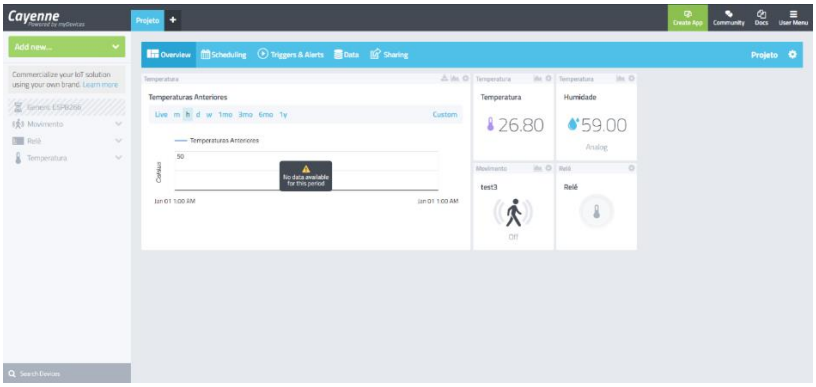

Figura 13 - Ecrã Principal do *Cayenne.*

3.2.3 Xampp

Inicialmente começamos por usar o Xampp.

O XAMPP é um *software* gratuito e de código aberto que facilita a criação e configuração de um ambiente de desenvolvimento *web* local combinando vários componentes-chave, incluindo o servidor web Apache, o sistema de gestão de uma base de dados MySQL, o interpretador de linguagem de script *PHP* e o servidor de aplicativos *Perl*. O nome XAMPP é um acrônimo que representa os sistemas operacionais compatíveis: X (para qualquer sistema operacional), *Apache*, *MySQL*, *PHP* e *Perl*.

A principal finalidade do XAMPP é fornecer aos desenvolvedores web um ambiente completo e pronto para uso nos seus computadores locais, permitindo que se desenvolvam e se testem aplicativos web antes destes serem implantados num servidor de hospedagem remoto, sendo software bastante utilizado por desenvolvedores de diferentes níveis de habilidade, desde iniciantes até profissionais experientes [20][21].

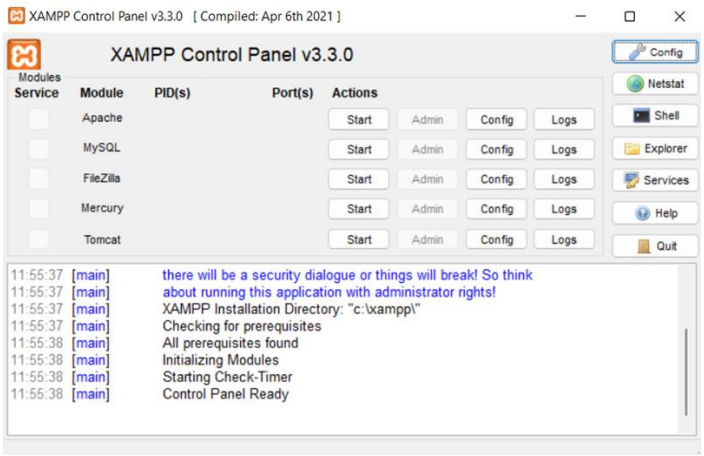

Figura 14 -Ecrã principal do XAMPP.

Os principais componentes do XAMPP são:

- *Apache*

O Apache é um dos principais componentes do XAMPP sendo responsável por fornecer o servidor web no ambiente de desenvolvimento local, desempenhando um papel fundamental na hospedagem e entrega de páginas da web aos clientes que acedam o servidor.

As principais funções e recursos do *Apache* no XAMPP são:

1. Servidor web: O *Apache* atua como um servidor web, recebendo solicitações HTTP dos clientes (navegadores da web) e fornecendo as melhores respostas sendo capaz de lidar com várias conexões em simultâneo e gerir solicitações de diferentes utilizadores.
2. Hospedagem de sites: O *Apache* permite que se alojem sites e páginas web localmente, podendo-se criar diretórios e colocar arquivos HTML, CSS, *JavaScript* e outros arquivos relacionados no diretório raiz do servidor Apache. Quando se faz um pedido para um URL específico, o *Apache* localiza o arquivo correspondente e envia-o como resposta.
3. Configuração flexível: O *Apache* oferece uma ampla gama de opções de configuração, permitindo que se personalize o comportamento do servidor *web* de acordo com as necessidades. Pode-se definir diretivas para controlar o acesso a arquivos e diretórios, configurar redireccionamentos, definir diretórios padrão e muito mais.
4. Suporte a módulos: O *Apache* é altamente modular e permite a integração de vários módulos adicionais para estender funcionalidades. Existem módulos disponíveis para suporte a tecnologias específicas, como *PHP, Perl, SSL* (para comunicação segura), autenticação de utilizadores, entre outros. No XAMPP, os módulos *PHP* e *Perl* já estão integrados ao *Apache*, facilitando o desenvolvimento de aplicativos *web* dinâmicos.

5. Registo de erros: O *Apache* regista informações detalhadas sobre erros e eventos ocorridos durante o processamento das solicitações, sendo útil para depuração e solução de problemas em aplicativos *web*, pois permite que se identifique e se corrijam problemas de forma mais eficiente.

- *MySQL*

O *MySQL* é um sistema de gestão de base de dados relacional (RDBMS) sendo um dos principais componentes do XAMPP desempenhando um papel crucial no armazenamento, na gestão e recuperação de dados para aplicativos *web* desenvolvidos em ambiente local.

As principais funcionalidades e recursos do *MySQL* no XAMPP são:

1. Armazenamento de dados: O *MySQL* permite que se crie e gerencie bases de dados para armazenar informações de forma organizada permitindo a criação de tabelas para armazenar dados específicos e definir a estrutura dos campos (colunas) e os tipos de dados que eles podem conter.
2. Consultas e manipulação de dados: O *MySQL* oferece uma linguagem de consulta poderosa chamada SQL (*Structured Query Language*). Com o SQL podemos executar consultas para recuperar, inserir, atualizar e excluir dados em tabelas de base de dados. Essas operações permitem a interação com os dados armazenados no base de dados de maneira eficiente.
3. Confiabilidade e desempenho: O *MySQL* é conhecido pela sua confiabilidade, desempenho e escalabilidade tendo sido projetado para lidar com grandes volumes de dados e cargas de trabalho intensas. No XAMPP podemos aproveitar esses recursos para desenvolver e testar aplicativos web que exigem acesso a dados confiáveis e rápidos.
4. Segurança: O *MySQL* possui recursos de segurança robustos para a proteção de dados onde podemos definir permissões de acesso granulares para utilizadores e tabelas, controlar o acesso a bases de dados e criptografar conexões para proteger informações confidenciais.
5. Integração com aplicativos web: O *MySQL* é amplamente utilizado em aplicativos web devido à sua compatibilidade com várias linguagens de programação, incluindo *PHP*. Com o XAMPP podemos integrar o *MySQL* a aplicativos *web PHP* local e aceder ao base de dados para armazenar e recuperar dados.

- *PHP*

O *PHP* é uma linguagem de script amplamente utilizada para o desenvolvimento de aplicativos web dinâmicos. No XAMPP, o *PHP* tem um papel fundamental na criação e execução de páginas da web interativas e na interação com base de dados.

As principais funcionalidades e recursos do *PHP* no XAMPP são:

1. Geração de páginas dinâmicas: O *PHP* permite que sejam criadas páginas da web que podem exibir conteúdos dinâmicos. Isso significa que se conseguem criar páginas com informações diferentes com base em variáveis, dados da base de dados, interações do utilizador e outros fatores. Com o XAMPP, podemos escrever *scripts PHP* e executá-los no servidor *web Apache* para criar páginas dinâmicas.
2. Interação com a base de dados: *PHP* oferece recursos embutidos para interagir com as bases de dados, como *MySQL* permitindo estabelecer conexões com a base de dados, enviar consultas SQL para recuperar ou modificar dados e processar os resultados retornados, permitindo que a criação de aplicativos *Web* que armazenam e recuperam dados da base de dados, como informações do utilizador, conteúdo dinâmico e muito mais.
3. Manipulação de formulários e dados de entrada: O *PHP* facilita a recolha e o processamento de dados enviados por formulários *HTML*, podendo receber valores enviados por utilizadores, validar e filtrar esses dados para garantir a segurança, e utilizá-los nas páginas da web de acordo com as necessidades do aplicativo.
4. Suporte a funções e bibliotecas: O *PHP* possui uma ampla variedade de funções e bibliotecas incorporadas que tornam a programação web mais fácil e eficiente, oferecendo recursos para a manipulação de *strings*, gestão de arquivos, manipulação de datas e horas, manipulação de imagens e muito mais. Além disso, existem várias bibliotecas e estruturas adicionais disponíveis que podem ser usadas para desenvolver aplicativos mais complexos no XAMPP.
5. O *PHP* integra-se perfeitamente com outros componentes XAMPP, como *Apache* e *MySQL* e pode ser configurado para se comunicar com o servidor *web Apache* para processar solicitações e gerar respostas dinâmicas. Além disso, o *PHP* pode se ligar ao *MySQL* para executar operações da base de dados.

- *Perl*

O *Perl*, uma linguagem de programação de alto nível e uso geral, também está incluída no pacote XAMPP. Embora o *Perl* não seja tão utilizado para o desenvolvimento de aplicações web como o *PHP*, este é frequentemente usado para tarefas de script e automação.

As principais funcionalidades e usos do *Perl* no XAMPP são:

1. *Scripting* e automação: O *Perl* é conhecido pela sua capacidade de escrever *scripts* eficientes para a automação de tarefas permitindo manipulações avançadas de arquivos, expressões regulares e recursos de processamento de texto, tornando-o adequado para scripts que lidam com grandes volumes de dados, processamento de *logs*, extração de informações e muito mais.
2. Manipulação de dados: O *Perl* oferece uma variedade de recursos para manipulação de dados, como manipulação de cadeia de caracteres, processamento de arquivos, análise de texto e transformação de dados. Esses recursos são úteis para lidar com tarefas que envolvem manipulação e transformação de dados, como extrair informações de arquivos CSV, gerar relatórios e processar dados de entrada.
3. Desenvolvimento *web*: Embora o *Perl* não seja tão amplamente utilizado para o desenvolvimento de aplicações web como o *PHP*, este ainda pode ser usado para criar aplicações web. O *Perl* possui módulos e *frameworks* que permitem o desenvolvimento de aplicações web, embora a popularidade e o suporte da comunidade sejam menores em comparação com o *PHP*.
4. Administração do sistema: O *Perl* é frequentemente usado para tarefas de administração do sistema, como gerenciamento de servidores, processamento de *logs*, configuração e automação de tarefas relacionadas ao servidor. No XAMPP, o *Perl* pode ser usado para executar scripts que executam tarefas de administração e gerenciamento para o ambiente de desenvolvimento local.

O Xampp parecia ser a solução indicada para se prosseguir no projeto, no entanto no desenvolver do projeto iremos apresentar os motivos pela qual não seguimos com o Xampp.

3.2.4 *000Webhosting*

A *000webhost* é um fornecedor de hospedagem web gratuita que permite aos utilizadores a possibilidade de alocar websites sem qualquer custo financeiro, sendo uma plataforma para pessoas que pretendem criar um website básico sem terem de investir em serviços pagos.

Algumas das principais características da *000webhost* são:

- Hospedagem gratuita: Permite aos utilizadores criar e publicar nos seus websites sem qualquer custo inicial.
- Subdomínio: Pode-se criar um website usando um subdomínio do seu domínio principal. Como exemplo pode-se usar o *URL* "oseusite.000webhostapp.com".
- Armazenamento e largura de banda: A *000webhost* disponibiliza 1 GB de espaço de armazenamento e 10 GB de largura de banda mensal. Embora estes limites possam ser suficientes para websites pequenos, se tiver um site maior ou com mais tráfego, poderá considerar a atualização para um plano de hospedagem pago.

- Suporte a bases de dados *MySQL*: A *000webhost* permite criar e gerir uma base de dados *MySQL* para o seu website. Isto é útil se precisar de armazenar e recuperar dados de forma dinâmica.
- Suporte a *PHP*: O serviço de hospedagem suporta *PHP*, uma linguagem de programação popular para desenvolvimento web. Isto permite a criação de páginas web dinâmicas e interativas.
- Suporte ao cliente limitado: Como fornecedor de hospedagem gratuita, a *000webhost* oferece um suporte ao cliente limitado. Embora disponibilizem uma base de conhecimentos e fóruns comunitários para ajudar os utilizadores, existem limitações em termos de assistência personalizada.

Embora a *000webhost* ofereça hospedagem gratuita, existem algumas limitações quando comparados com os serviços de hospedagem pagos. Estas limitações podem incluir tempos de carregamento mais lentos do site, possíveis períodos de inatividade do servidor e menos funcionalidades avançadas [22].

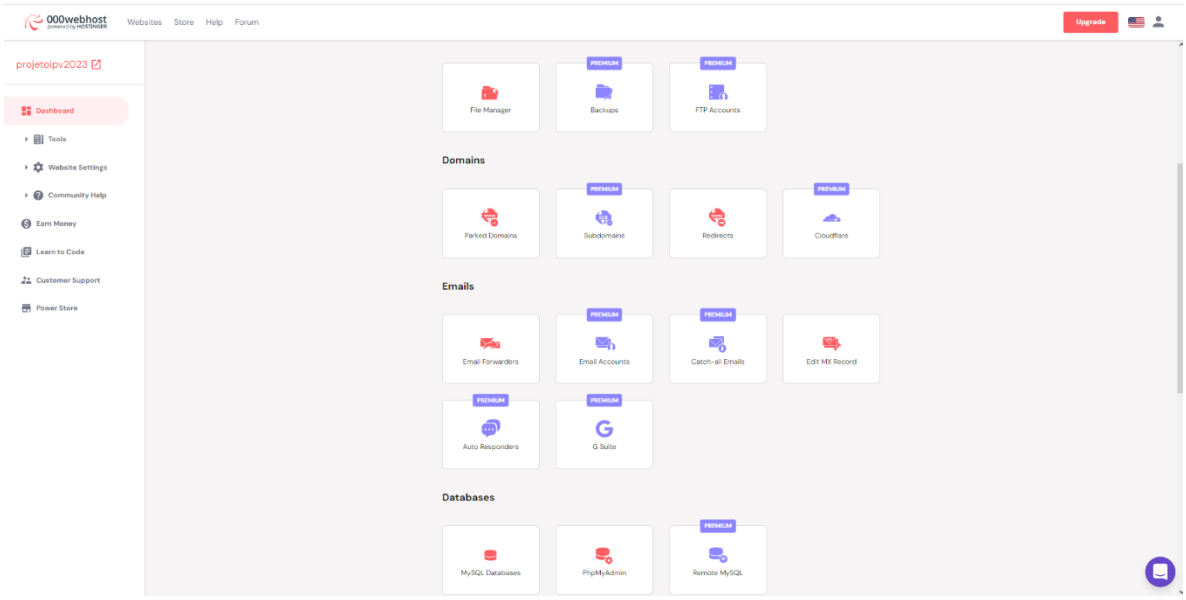

Figura 15 - *000webhost*.

3.2.5 MIT *App Inventor*

O MIT *App Inventor* é uma ferramenta online que permite criar aplicativos para telemóveis ou tablets diretamente do navegador da web sendo é uma ferramenta gratuita e de uso acessível, permite que qualquer pessoa crie aplicativos para dispositivos *Android*. Foi desenvolvido pelo MIT (Instituto de Tecnologia de *Massachusetts*) como uma ferramenta de código aberto para promover a programação e a educação em ciência da computação.

O MIT *App Inventor* utiliza uma abordagem baseada em blocos, também conhecida como programação visual, em que os utilizadores podem arrastar e soltar blocos de código de uma área de trabalho para criar a lógica do aplicativo.

O *App Inventor* possui uma interface gráfica que permite aos utilizadores projetar a aparência e a funcionalidade do aplicativo incluindo um conjunto de componentes pré-definidos, como botões, listas, sensores de movimento e GPS, que podem ser facilmente configurados e personalizados. Os utilizadores podem combinar esses componentes e definir as suas propriedades por meio de blocos de código visualmente organizados.

Uma das principais vantagens do MIT *App Inventor* é a sua acessibilidade.

O MIT *App Inventor* também possui recursos de depuração, emuladores de dispositivos Android e a capacidade de testar aplicativos diretamente em dispositivos móveis ligados. Isso permite que os desenvolvedores verifiquem o funcionamento correto de seus aplicativos antes de compartilhá-los ou disponibilizá-los na *Google Play Store*.

Além disso, o MIT *App Inventor* oferece suporte a recursos avançados, como acesso à base de dados, comunicação com servidores web e integração com outros serviços, permitindo que os aplicativos criados tenham funcionalidades mais complexas.

Desde o seu lançamento, o MIT *App Inventor* tem sido amplamente adotado em escolas, instituições educacionais e comunidades ao redor do mundo como uma ferramenta eficaz para ensinar programação e incentivar a criatividade na criação de aplicativos móveis. Ele desempenha um papel importante na promoção da alfabetização digital e na democratização do desenvolvimento de aplicativos para dispositivos móveis [23].

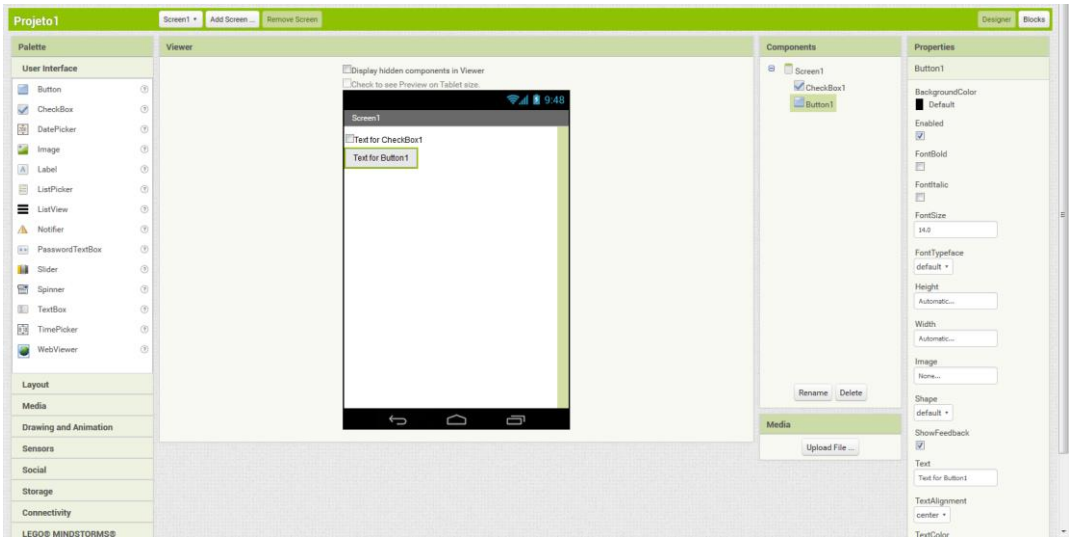

Figura 16 - Interface inicial da página *web.*

4. Sistema desenvolvido

A implementação do sistema engloba a seleção e a montagem dos componentes necessários para a realização do sistema de aquisição e tratamento de dados. Sendo também expostos adiante os procedimentos obrigatórios para a ligação dos dispositivos á rede *WiFi* e aos softwares verificando os valores dos sensores e ainda a criação da base de dados para armazenar os dados.

4.1 Instalação do Arduíno IDE

Inicialmente procedeu-se ao download do *software* Arduíno IDE no site oficial do mesmo.

Para se poder fazer a ligação ao ESP-01 foi necessário seguir os seguintes passos:

1. Abra o Arduíno IDE;
2. Ir a "*File*" e clicar em "*Preferences*";
3. Na janela "*Preferences*" aceder ao local "*Aditional Boards Manager URLs*" e colocar os seguintes endereços:

"http://arduino.esp8266.com/stable/package_esp8266com_index.json;
https://dl.espressif.com/dl/package_esp32_index.json"

4. Clicar em "OK";
5. Ir ao menu "*Tools*" e selecionar "*Board* > *Boards Manager*";
6. Na caixa de pesquisa é necessário procurar por "esp8266";
7. Irá aparecer uma placa denominada de "*esp8266 by ESP8266 Community*" e fazer a sua transferência (aconselhado transferir a versão mais atualizada possível);
8. Depois de transferido e instalado aceder ao menu "*Tools*>*Board*>esp8266" e selecionar a placa "*Generic ESP8266 Module*".

Seguidos estes passos já é possível através do computador testar os programas para mais tarde os poder testar.

4.1.1 Programar o ESP8266 ESP-01

Inicialmente para a programação do ESP-01 é necessário colocá-lo em modo de leitura para que se possa enviar o código.

Para isso foi necessário comprar um programador USB série para o ESP8266 ESP-01 e soldou-se um botão de pressão entre dois pinos (GPIO-0 e o GND) [23].

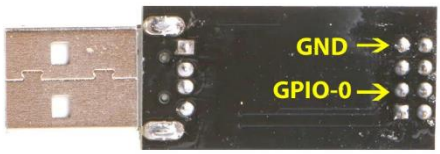

Figura 17 -Pinos para ligar o botão de pressão [24].

Depois de soldados, ao inserir a pen no computador é necessário pressionar o botão durante aproximadamente 1 segundo.

Efetuados esses processos e tendo já o programa Arduíno IDE pronto para enviar informação estavam as condições criadas para começarmos a enviar o código para o ESP-01 e testar o seu funcionamento.

4.1.2 Ligação á internet do módulo *WiFi* no Arduíno

Para ligar o módulo *WiFi* esp8266-01 à internet precisámos de utilizar no Arduíno a biblioteca:

```
#include <ESP8266WiFi.h>
```

Após a instalação da mesma, precisamos adicionar as seguintes linhas de código com os dados de acesso á internet:

```
// Informações da rede WiFi
char ssid[] = "NOMEDAREDE";  // nome da rede WiFi
char password[] = "SENHADAREDE";  // senha da rede WiFi
```

Seguidamente criámos uma função chamada "*connectWiFi*" para quando fosse chamada fizesse a ligação á internet:

```
void connectWiFi() {

  WiFi.mode(WIFI_OFF);

  delay(1000);

  //Esta linha esconde a visualização do ESP como hotspot wifi
  WiFi.mode(WIFI_STA);

  WiFi.begin(ssid, password);
  Serial.println("A aceder ao WiFi");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}
```

Por fim no *void setup* chamamos a função criada anteriormente da seguinte forma:

```
void setup() {
  // Inicialização da comunicação serial
  Serial.begin(115200);
  delay(100);
  connectWiFi();

}
```

4.2 Configuração no *Cayenne*

O *Cayyenne MyDevices* é uma plataforma de IoT, algo necessário para se conseguir simular os códigos e perceber na prática o se estavam a funcionar corretamente ou se precisavam de melhorias. Este utiliza o protocolo MQTT permitindo que se possa aceder às informações enquanto o servidor administra os dados.

De forma mais simples, para se ter um protocolo MQTT terá de existir um publicador que será responsável por publicar as mensagens num determinado tópico onde um assinante irá inscrever-se neste tópico para poder aceder à mensagem.

Como não há uma ligação direta entre o assinante e o publicador, para que essas mensagens aconteçam, o protocolo MQTT irá precisar de um gestor de mensagens.

O MQTT pode ser dividida nos seguintes termos:

- *Subscriber* (Subscrito) - pessoa que estará inscrita no tópico e irá ter o papel de recetor.
- *Publisher* (Publicador) - pessoa que estará responsável por ser o emissor e enviar os dados para um determinado tópico.
- *Broker* - será o intermediário para fazer uma ponte de comunicação entre o *Publisher* e o *Subscriber*, tornando-se responsável por receber, decifrar e enviar as mensagens.
- Tópico - será o endereço para onde os dados das mensagens serão enviados.
- *Client* (Cliente) - elemento que terá a capacidade de interagir com o *Broker*, podendo enviar e receber dados.
- Mensagem - pacote de dados trocados entre os clientes e o *Broker*.
- *Unsubscribe* - permite deixar de assinar um determinado tópico.
- *Payload* - conteúdo da mensagem que será enviada.

4.2.1 Ligação ao *Cayenne* pelo módulo *WiFi* no Arduíno

Para iniciarmos a sincronização necessinamos de adicionar a seguinte biblioteca ao código:

```
#include <CayenneMQTTESP8266.h>
```

Ao efetuar a ligação ao *Cayenne* necessitamos de declarar no código o destino para onde iremos enviar o respetivo código. Para isso necessitamos de três componentes, sendo elas:

- *MQTTUsername*: refere-se ao nome de utilizador da conta em que queremos que o código seja visível
- *MQTTPassword*: refere-se à palavra-passe da conta em que queremos que o código seja visível
- *ClientID*: refere-se ao dispositivo em que queremos que os dados sejam lidos ou enviados

Esses dados encontram-se na página do *Cayenne* quando queremos sincronizar um novo dispositivo. Na Figura 18 temos uma ilustração desses mesmos dados.

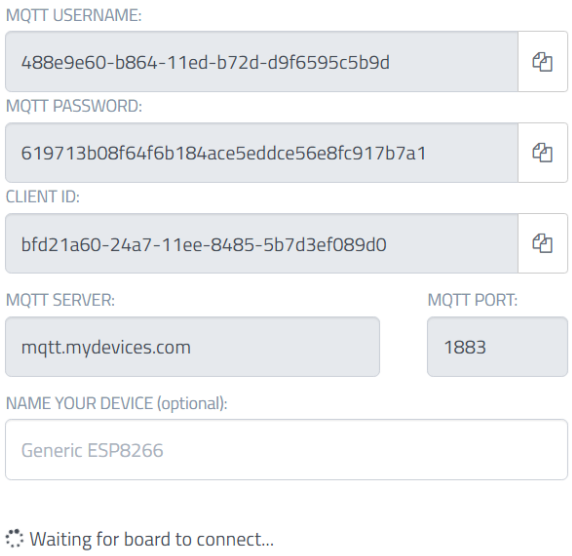

Figura 18 – Dados de sincronização para o *Cayenne*.

O código a colocar-se para fazer a sincronização é o seguinte:

```
char username[] = "488e9e60-b864-11ed-b72d-d9f6595c5b9d";
char mqtt_password[] = "619713b08f64f6b184ace5eddce56e8fc917b7a1";
char clientID[] = "3340b530-c259-11ed-b72d-d9f6595c5b9d";
```

Necessitamos também de acrescentar a seguinte linha de código no void setup:

```
void setup() {
  Cayenne.begin(username, mqtt_password, clientID, ssid, password);
}
```

O processo será finalizado quando o ESP estiver ligado à internet e a página da Figura 18 atualizar para a página da Figura 19.

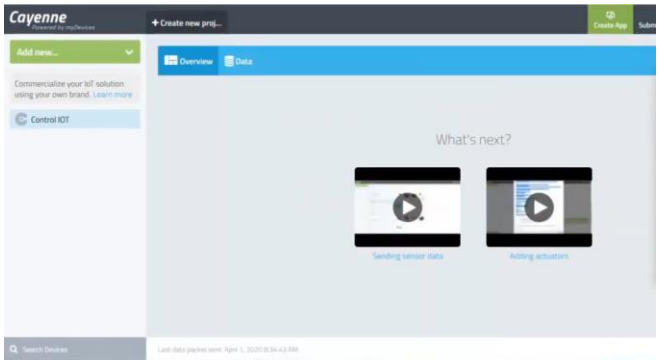

Figura 19 - Página inicial *Cayenne*.

4.2.2 Programação do sensor de temperatura/humidade

Começamos por simular o funcionamento do sensor de temperatura.

Inicialmente demos conta do principal erro do sensor que por ser de gama mais barata acaba por ter muitos erros, sendo um deles a medição da temperatura onde acrescentava mais 5ºC ao alor real da temperatura. Esse erro foi corrigido através do código onde foi pedido que o valor apresentado no *Cayenne* fosse o valor lido pelo sensor e a esse valor retirar 5 unidades e com isso conseguimos ter a temperatura real. Para confirmar o valor medido usou-se um aparelho de medida onde através dele conseguimos calibrar melhor o sensor de temperatura, e para isso utilizamos o seguinte código na programação do módulo *Wifi* no sensor de temperatura/humidade:

```
float temperature = dht.getTemperature();
float humidity = dht.getHumidity();
float temperature1=temperature-5;
Cayenne.celsiusWrite(1, temperature1);
Cayenne.virtualWrite(2, humidity, "rel_hum", "p");
```

4.2.3 Programação do relé

O relé também teve uns percalços, pois ao início não estávamos a conseguir comandar o relé através do *Cayenne* e estávamos a usar o botão de reset como botão ON/OFF.

Conseguiu-se perceber do problema do botão do reset e através de umas funcionalidades no *Cayenne* conseguimos resolver o problema e deixar o relé manipulável através do *Cayenne*.

Para conseguirmos controlar o relé, primeiro temos de declarar no código as suas informações como estão nas linhas de código abaixo descritas:

```
// Informações do shield relé
#define RELAY_PIN 0  // Pino do ESP8266 ligado ao pino de controle do relé
#define VIRTUAL_CHANNEL_RELAY 0  // Canal virtual do Cayenne para o relé
```

Tendo já definidas as informações temos de configurar o relé sendo isso possível através do *pinMode* que nos permite manipular o estado do relé através de impulsos únicos, como mostrado no código seguinte:

```
// Configuração do pino do relé
pinMode(RELAY_PIN, OUTPUT);
digitalWrite(RELAY_PIN, HIGH); // Desliga o relé inicialmente
```

4.2.4 Sensor de movimento

No sensor de movimento existiram as maiores dificuldades, pois o sensor de movimento não trazia nenhum módulo de encaixe rápido, logo tivemos de fazer as ligações elétricas de forma manual.

Um dos problemas surgiu logo quando verificamos no *datasheet* do sensor de movimento que a sua alimentação teria de ser de 5v, o que não coincidia com a alimentação do ESP-01 que era de 3,3v, logo tivemos de usar duas fontes diferentes.

Como o sinal enviado pelo sensor de movimento era de 5v, então usamos uma resistência de 100Ω para reduzir a tensão para os 3,3v para que o ESP-01 consiga ler os valores e ao mesmo tempo não tenha danos colaterais.

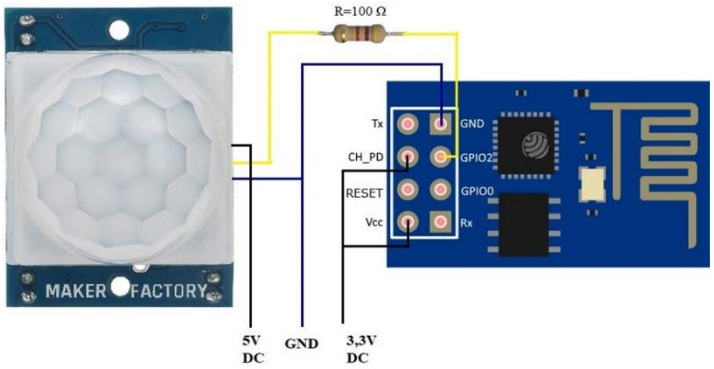

Figura 20 – Ligações elétricas do sensor de movimento.

Fizemos um código, mas não tínhamos acesso às leituras certas do sensor no *Cayenne*, começamos por alterar o código de programação, mas nada funcionava, até que começamos a regular os potenciómetros de sensibilidade e de *delay*.

Feitos esses ajustes começamos a ter as leituras no *Cayenne* mas não eram as leituras reais tendo muito tempo de atraso nas leituras e por vezes tinha leituras inexistentes.

Verificamos o código para procurar a raiz do problema até que começamos a medir tensões no sensor e no ESP-01 e nos deparamos com tensões diferentes daquelas que estávamos á espera e apercebemo-nos de um erro que cometemos, pois usávamos um transformador de telemóvel de 5 V para alimentar o sensor e usávamos as tensões do arduíno para alimentar os 3,3 V do ESP. Como as massas não eram as mesmas, os valores de tensão alteravam entre eles o que nos provocava os erros que estávamos a ter.

Decidimos colocar todas as tensões no arduíno e tudo ficou a funcionar.

Fizemos outro teste para verificar o historial de funcionamento do sensor e deparámo-nos com os seguintes dados da Figura 21.

Esperávamos resultados exatos entre 0 e 1 significando que o sensor estava a detetar ou não movimento e como ilustrado na Figura 21 esse não foi o caso.

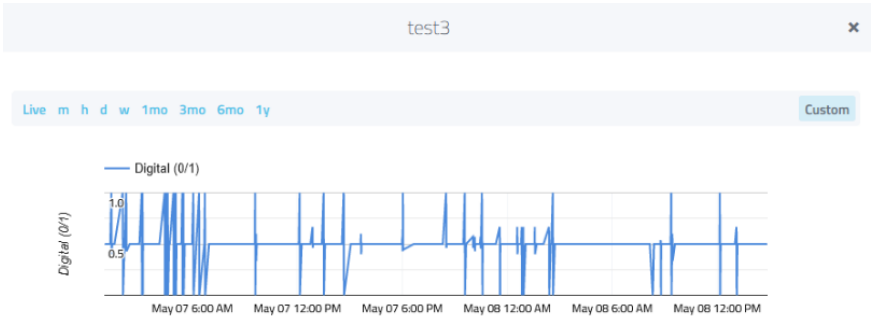

Figura 21 – Histórico do sensor de movimento.

Enquanto usávamos o *Cayenne* deparamo-nos com vários problemas de ligação ao *website* e como não queríamos falhas na ligação começamos a explorar outras opções.

Utilizamos o Xampp, tínhamos já efetuado a ligação do ESP-01 com a aplicação quando surgiu a ideia de se fazer o controlo do sistema mesmo sem a ligação à *internet* e devido a esse fator o Xampp não nos iria ajudar no desenvolvimento do projeto e devido a esse fator abandonamos o projeto no Xampp e abordamos novas opções.

A opção encontrada será descrita nos processos em baixo referidos.

4.3 Transmissão das leituras para a base de dados

As leituras dos dispositivos (rele/sensor temperatura) são transmitidos para a base de dados através do módulo *WiFi* ESP8266-01.

Para efetuar essa transportação de dados foi necessário instalar a biblioteca:

```
#include <ESP8266HTTPClient.h>.
```

Esta permitiu-nos utilizar as funções `GET()` e `POST()` para enviar solicitações *HTTP* para um servidor *web*.

Nos anexos 1,4,5,11 e 12 podemos verificar o código utilizando estas funções.

4.3.1 Funções de atualização da base de dados

Para atualizar o estado do relé na base de dados sempre que este é ligado ou desligado, foi necessário desenvolver uma função que nos permitisse chamá-la sempre que fosse precisa.

Função "*atualizarDB*"

```
void atualizarDB() {

  String postData1 = "estadorele=" + String(!digitalRead(RELAY_PIN));
// Guarda em string o estado do relé

  HTTPClient http1;
  http1.begin(wifiClient, URL);
  http1.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpSendCode = http1.POST(postData1); // Envia os dados da string para o
ficheiro PHP
  http1.end();  //fecha a ligação
}
```

4.4 Base de dados num servidor online

Para o desenvolvimento deste projeto precisámos de criar uma base de dados permitindo desta forma guardar as informações do sensor de temperatura/humidade, do relé, e dos horários que criados. Optámos por desenvolver uma base de dados que esteja sempre online (requer acesso á internet) uma vez que criada localmente (através do xampp) chegámos á conclusão de que esta só seria controlada se os dispositivos estivessem ligados á mesma rede local, ou seja na mesma rede *WiFi*, impedindo o acesso a ela a partir de qualquer lugar. O módulo de *WiFi* não havendo internet também não se conseguia de ligar a ela.

Tendo uma base online temos a vantagem de podermos controlar o funcionamento dos dispositivos a partir de qualquer lugar desde que existe uma ligação á internet. Desta forma chegámos ao site 000webhost que nos possibilitou através do pelo uso da ferramenta *web PHP MyAdmin* desenvolver a base de dados em *MySQL* online gratuitamente.

4.4.1 *PHP MyAdmin*

O PHP MyAdmin é um software desenvolvido em *PHP* que tem como objetivo gerenciar base de dados por meio de um navegador. Ele oferece suporte a várias operações no *MySQL* e *MariaDB*, como criação e exclusão de bases de dados, tabelas e campos, além de permitir a execução de consultas SQL diretamente pela interface. Por ser constantemente atualizado, essa ferramenta é uma opção confiável para a administração de base de dados em *MySQL*.

4.4.2 Criação das tabelas

Depois de aceder á base de dados já criada, construímos 2 tabelas com os nomes “dht11” e “Horario”.

A tabela “dht11” destina-se a guardar os dados do sensor de temperatura/humidade e do relé, enquanto a “Horario” destina-se a guardar os horários criados através da aplicação *Android* ou do *website*.

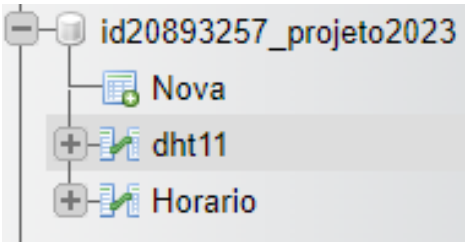

Figura 22 - Tabelas da base de dados.

4.4.3 Estrutura da tabela “dht11”

Na criação desta tabela, começamos por criar 5 variáveis com as seguintes funções:

- “ID”: Esta variável é de auto incremento, o que significa que sempre que forem adicionados novos dados, um ID será atribuído automaticamente. Esse ID serve para identificar cada conjunto de informações de forma única.
- “temperature3”: É utilizada para armazenar o valor da temperatura enviado pelo sensor. É por meio dessa variável que podemos acompanhar e registar as leituras de temperatura.
- “humidity1”: Outra variável importante é a “humidity1”, que guarda o valor da humidade enviada pelo sensor. Ela permite-nos monitorar e armazenar os níveis de humidade no ambiente em que se encontra o sensor.
- “estadorele”: A variável “estadorele” é responsável por guardar o estado do relé. Essa informação indica se o relé está ligado ou desligado.
- “datetime”: Por fim, temos a variável “datetime”, que é utilizada para armazenar a data e hora da última atualização dos dados. Essa informação é valiosa para acompanhar quando as informações foram atualizadas pela última vez e garantir a precisão dos registos.

	#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido
☐	1	**ID** 🔑	int(11)			Não	*Nenhum*
☐	2	**temperature3**	int(11)			Não	*Nenhum*
☐	3	**humidity1**	int(11)			Não	*Nenhum*
☐	4	**estadorele**	int(11)			Sim	*NULL*
☐	5	**datetime**	datetime			Não	current_timestamp()

Figura 23 - Estrutura da tabela “dht11”.

4.4.4 Estrutura da tabela “Horário”

Na criação da tabela “Horário”, foram definidas 5 variáveis com as seguintes funções:

- “id”: Esta variável é de autoincremento, assim como na tabela dht11.
- “Data”: Nesta variável, armazenamos a data inicial de funcionamento do horário.
- “Hora”: Aqui, guardamos a hora inicial de funcionamento do mesmo.
- “DataFim”: Utilizamos esta variável para armazenar a data final de funcionamento do horário.
- “HoraFim”: Aqui, guardamos a hora final de funcionamento do mesmo.

	#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra
☐	1	id	int(11)			Não	*Nenhum*		AUTO_INCREMENT
☐	2	Data	date			Não	*Nenhum*		
☐	3	Hora	time			Não	*Nenhum*		
☐	4	DataFim	date			Não	*Nenhum*		
☐	5	HoraFim	time			Não	*Nenhum*		

Figura 24 - Estrutura da tabela “Horário”.

4.4.5 Transferência de dados entre o módulo *WiFi* e a base de dados

Com o objetivo de transferir dados entre o módulo *WiFi* e a base de dados foi necessário criar ficheiros *PHP* designados por *“DataInserir”, “FuncHorario”, “GetDataRele”, “login”, “logout”, “protected_page”, “read”, “readHorario”, “test_data”, “test_dataRele”, “update”, “update2”.*

Estes ficheiros funcionam como meio de transporte de dados obtido pelo Arduíno enviando-os para a base de dados e vice-versa. Permitindo desta forma que os dispositivos utilizados tenham acesso a todas as informações necessárias para o seu funcionamento correto.

Os códigos dos ficheiros são expostos do Anexo 3 ao 14.

4.4.6 Acesso à base de dados

Para aceder à base de dados é necessário utilizar a plataforma de hospedagem para o *website*, www.000webhost.com, pois foi a plataforma que optamos para hospedar o nosso *website*, por ser um serviço gratuito e suficiente para o desenvolvimento do projeto.

4.5 Criação de um *website*

Decidimos criar um website para caso não seja possível utilizar a aplicação. Neste podemos fazer as mesmas funcionalidades. Para isso recorremos ao site 000webhosting que apesar desta ter algumas limitações como por exemplo um máximo de dez mil interações por dia, é o suficiente para o funcionamento do projeto. Na criação do site, começamos, assim como na aplicação por criar um sistema de login para a proteção do mesmo.

O *URL* do nosso site é: https://projetoipv2023.000webhostapp.com/protected_page.php.

4.5.1 Login no *website*

Na Figura 25 mostra o painel de login, no qual é necessário inserir o nome de utilizador (Projeto) e a Senha (Projeto) para termos acesso ao painel principal.

O código *PHP* desta página encontra-se no Anexo 6.

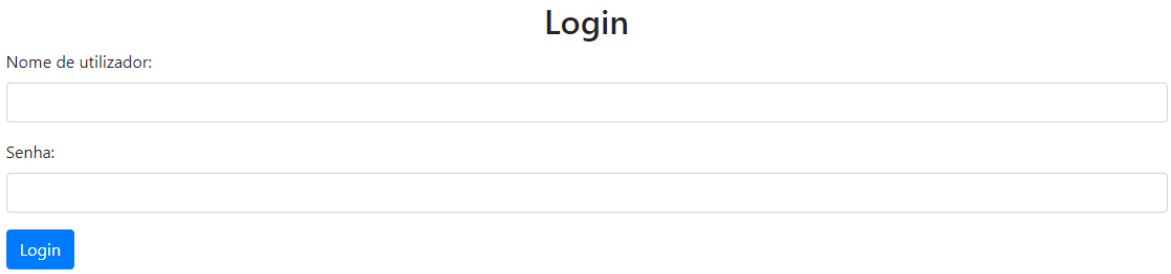

Figura 25 - Sistema de *Login* do *website*.

4.5.2 Painel principal

No painel principal podemos inserir novos horários de funcionamento, assim como apagar os horários já existentes. No final não nos podemos esquecer de efetuar o *Logout*, caso contrário a "página" estará sempre disponível para qualquer um que utilize esse dispositivo.

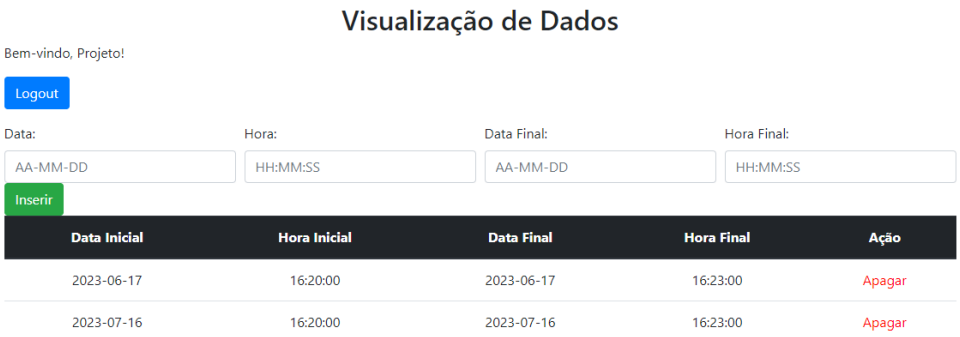

Figura 26 - Painel principal do *website*.

Podemos inserir novos horários de funcionamento, preenchendo os campos *"Data", "Hora", "Data Final", "Hora Final"* da forma correta, caso contrário não funcionará, e finalizando pressionado a opção "Inserir", como ilustrado na Figura 27.

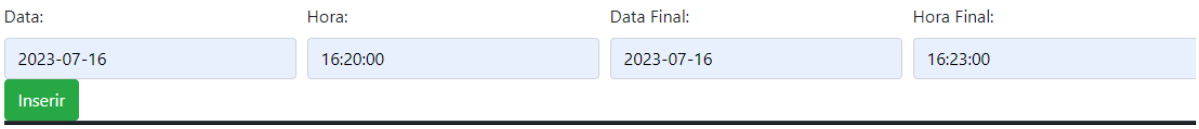

Figura 27 - Inserção de novos horários.

Podemos também eliminar horários já existentes, simplesmente clicando na opção "Apagar" como mostrado na Figura 28.

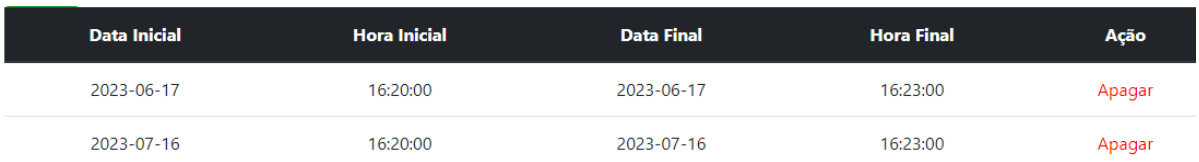

Figura 28 - Eliminação de horários.

4.6 Aplicação para telemóvel

Para o desenvolvimento da nossa aplicação Android, utilizamos a plataforma MIT *App Inventor* pois esta permite que pessoas com pouco conhecimento de programação criem os seus próprios aplicativos para Android. Ele usa uma interface gráfica baseada em blocos, tornando o desenvolvimento de aplicativos acessível e intuitivo. Os utilizadores arrastam e ligam blocos para criar funcionalidades e comportamentos. O *App Inventor* oferece recursos para testar, depurar e exportar aplicativos para dispositivos Android. É uma ferramenta prática que estimula a criatividade e a inovação.

4.6.1 Programação do ecrã de *login*

Para que haja uma pequena proteção contra possíveis invasões na nossa aplicação começamos por criar um sistema de login através do MIT *App Inventor*, sendo esta feita através de "blocos". Na Figura 29 podemos verificar que se não inserirmos o nome de utilizador (Projeto) e palavra-passe (Projeto) correta, irá mostrar um aviso de dados incorretos, não permitindo o acesso á aplicação.

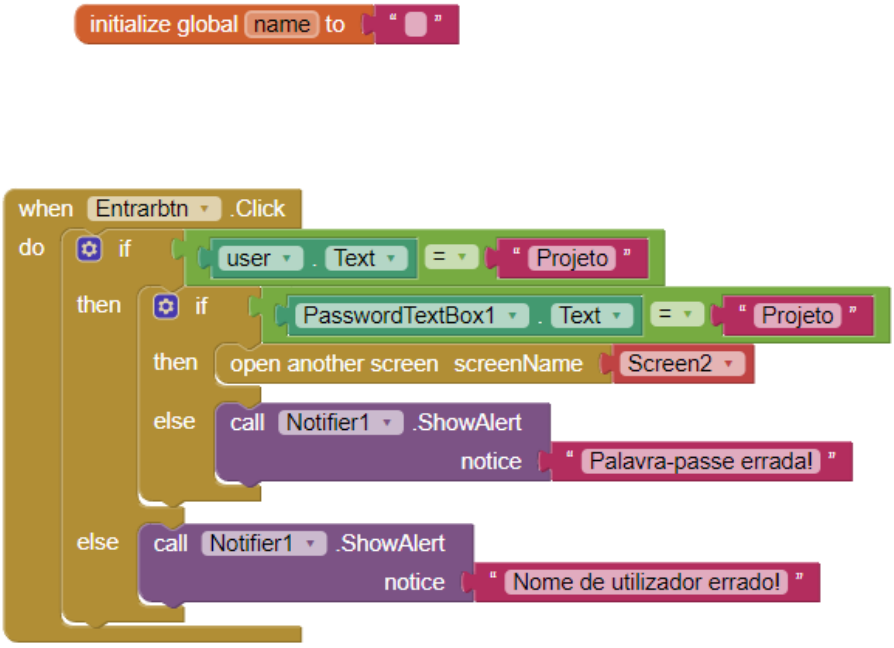

Figura 29 - Código do ecrã de *Login*.

4.6.2 Login na aplicação

Na Figura 30 revelamos o painel de *login* da aplicação no telemóvel. Neste após inserir o nome de utilizador e a palavra-passe carregamos no botão “Entrar” para termos o total acesso á aplicação.

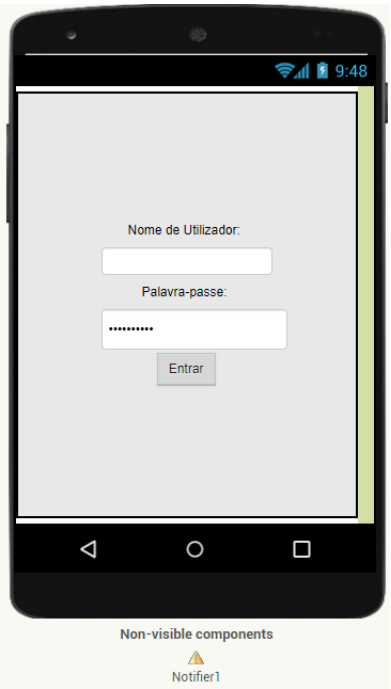

Figura 30 - Sistema de *Login* da aplicação.

4.6.3 Programação do ecrã do painel principal

Para termos acesso aos vários conteúdos foi necessário criar formas de acesso às tabelas da base de dados. Para isso recorremos á função do *webviewer* da aplicação do MIT *App Inventor*, que após carregarmos no botão dedicado a essa opção, reencaminha para o *URL* que foi predefinido. Nesse *URL* encontra-se um código desenvolvido em linguagem *PHP*, guardado num ficheiro no servidor online (000webhost).

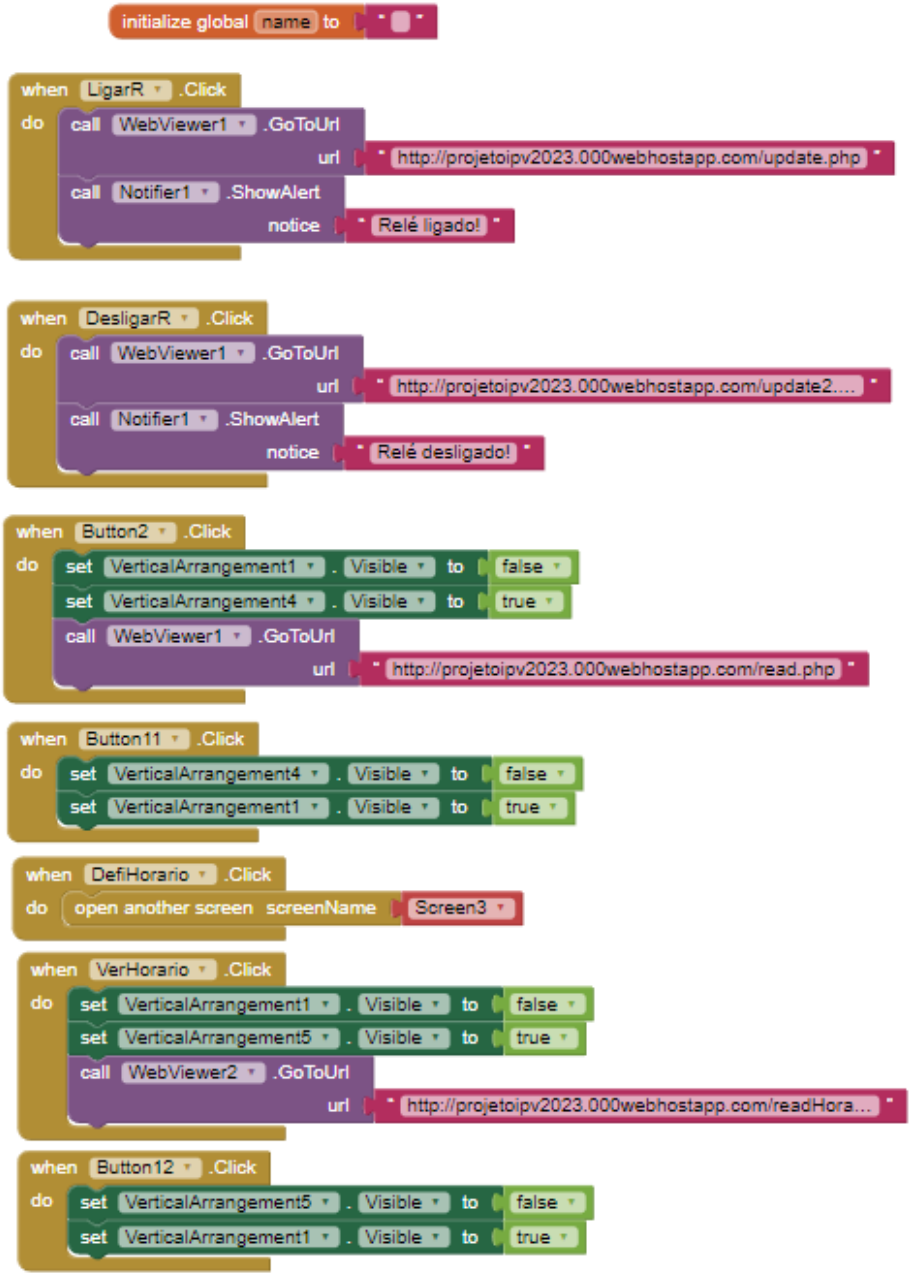

Figura 31 - Código do painel principal.

4.6.4 Funcionalidades da aplicação

No desenvolvimento do painel principal começámos por criar 5 botões de interação, sendo 2 deles utilizados para o controlo do relé (ligar/desligar), e os restantes para o acesso á *URL* predefinida tendo as seguintes funções:

- "Definir horário": Abre um novo ecrã, possibilitando a criação de horários.
- "Ver horários": Dá acesso a um novo ecrã, mostrando os horários existentes.
- "Visualizar dados": Permite aceder a um novo ecrã, revelando o estado dos dispositivos.

Figura 32 - Painel principal da aplicação.

4.6.5 Programação do ecrã de criação de horários

Para o funcionamento dos botões neste ecrã, foi necessário recorrer á função "*AfterDateSet*" do MIT *App Inventor*, esta permitiu que quando o botão for pressionado abra um *pop-up* permitindo escolher a data/hora pretendida, seguidamente mostra os dados escolhidos através das funções "*FormateDate*" e "*FormatTime*".

Para o botão "Guardar" enviar os dados para a base de dados recorremos á função "*join*". Esta faz a ligação *URL* predefinida que por sua vez armazena os dados na base de dados através de um código em linguagem *PHP*. Mais uma vez o ficheiro que contém esse código encontra-se guardado no servidor online (000webhost).

Se o botão voltar for ativado, através da função "*open another screen screenName*" regressamos ao ecrã 2 (ecrã principal).

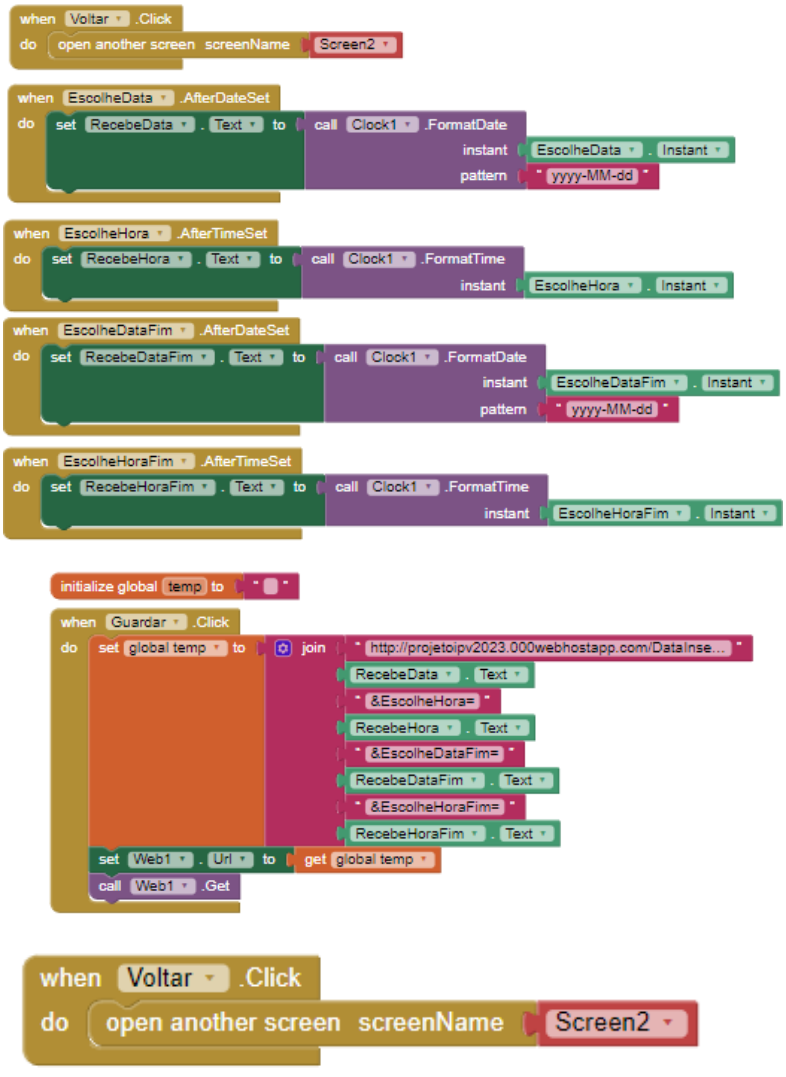

Figura 33 - Código em blocos da criação de horários.

4.6.6 Criação de horários

Na opção de “Definir horário”, podemos, como mostrado na Figura 34, criar horários de funcionamento como por exemplo do aquecimento da escola. Neste ecrã criámos 6 botões, 2 para definir a data e hora inicial, 2 para a data e hora final do nosso horário, o botão de “Guardar” para guardar os dados escolhidos na base de dados e o “Voltar” para regressar ao ecrã principal. Quando a hora atual estiver dentro do intervalo do horário, o relé será ligado, caso contrário desligado.

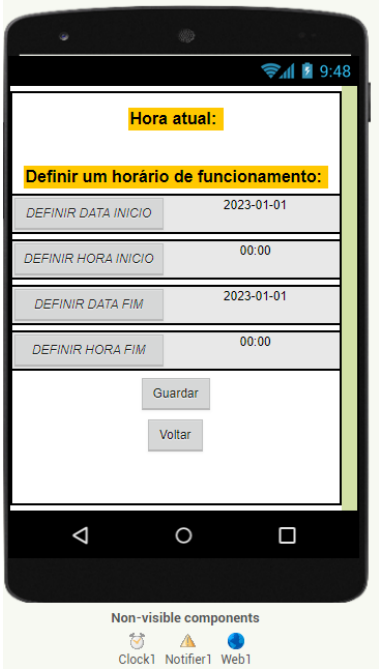

Figura 34 - Definição de horários.

4.6.7 Programação do ecrã “Ver horários”

Na programação deste ecrã começámos por remover a visibilidade do ecrã principal e dar ao ecrã “Ver horário” recorrendo á função “*set… .Visible to …*”. Para ser possível visualizar os dados armazenados na base de dados no servidor online, recorremos á função “*GoToURL*”, e nesta colocámos o *URL* do ficheiro *PHP* onde se encontra o código desenvolvido (o ficheiro encontra-se guardado também no servidor online).

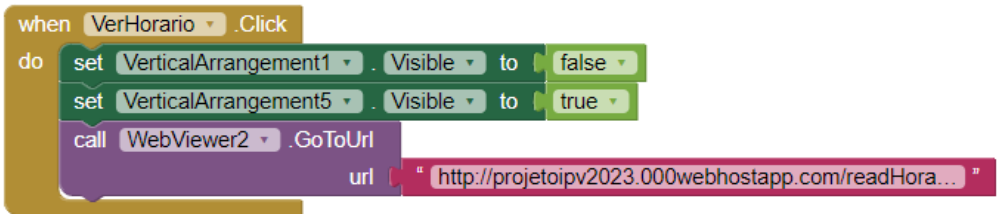

Figura 35 - Código em blocos do ecrã “Ver horários”.

4.6.8 Visualização de horários

Na opção de Ver horários, podemos como diz o mesmo verificar todos os horários criados tendo também a opção de os apagar. Carregando no botão “Voltar” regressamos ao ecrã principal.

Figura 36 - Visualização dos horários.

4.6.9 Programação do ecrã “Visualizar dados”

Assim como na programação do ecrã “Ver horários” neste ecrã começámos por remover a visibilidade do ecrã principal (*VerticalArrangement1*) e dar ao ecrã “Visualizar dados” (*VerticalArrangement5*) recorrendo á função “*set… .Visible to …*”. Para ser possível visualizar os dados armazenados na base de dados no servidor online, recorremos á função “*GoToURL*”, e nesta colocámos o *URL* do ficheiro *PHP* onde se encontra o código desenvolvido (também guardado no servidor online). Se o botão “Voltar” (*Button12*) for acionado remove a visibilidade do ecrã “Visualizar dados” e dá ao ecrã principal através da mesma função usada anteriormente.

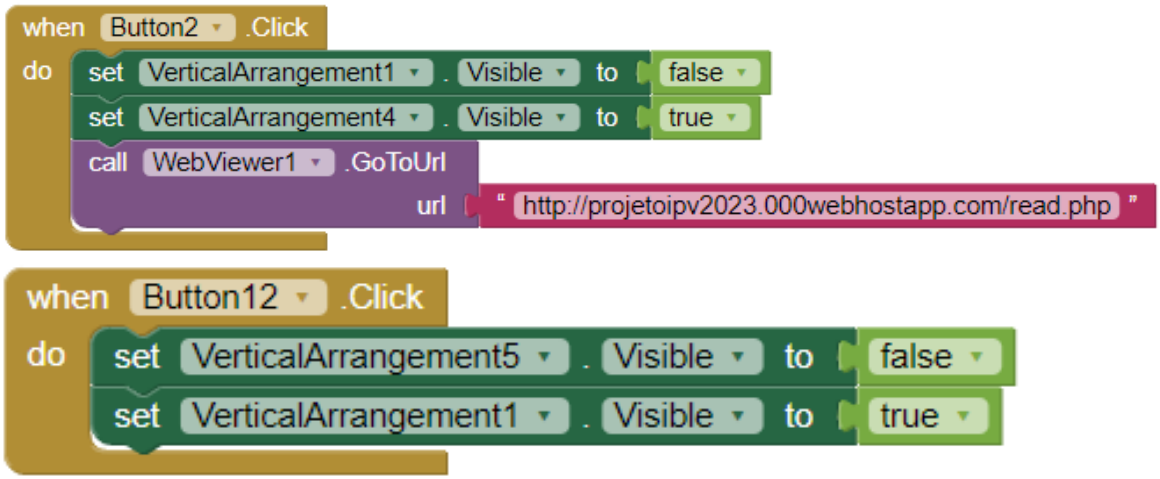

Figura 37 - Código em bloco do ecrã “Visualizar dados”.

4.6.10 Verificação da temperatura/humidade e estado do relé

Na opção de “Visualizar dados”, conseguimos verificar os dados que o sensor de temperatura e humidade envia, o estado do relé (ligado ou desligado) e também a hora e data da última atualização. Neste ecrã adicionamos o botão de “Voltar” para regressar ao painel principal se assim for desejado.

Figura 38 - Visualização do estado dos dispositivos.

4.7 Sensor de temperatura e relé no mesmo microcontrolador

O ESP-01 é pequeno e compacto sendo bastante limitado, no entanto foi possível agrupar o sensor de temperatura e o rele num só ESP-01.

A ideia surgiu na necessidade de falhar a internet e não haver forma de se conseguir controlar o relé de forma automatizada e através da junção dos dois dispositivos, mesmo que falhe a internet estes irão estar permanentemente em contacto não colocando em causa o conforto e o bem-estar das pessoas.

Desta forma conseguimos criar uma temperatura máxima desejada, sendo que quando a temperatura chega a esse valor o relé será automaticamente desligado, mesmo sem o acesso á

internet. O código para esta ideia está presente no Anexo 1 e abaixo revelamos um excerto sobre a máxima temperatura permitida.

```
if (temperature1 <= 26) { // Se a temperatura for inferior a 26 ativa os
horários

  HTTPClient httpfetch1;
  httpfetch1.begin(wifiClient, URL2); //Ligação com a base de dados para
verificar os horários
  httpfetch1.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpFetchCode1 = httpfetch1.GET();
  httpfetch1.end();  //fecha a ligação

delay(5000);
} else {

      digitalWrite(RELAY_PIN, HIGH); // Desliga o relé
      atualizarDB();
      delay(2000);
}
```

Neste caso se a temperatura do relé for superior a 26ºC o relé irá desligar mesmo se na criação dos horários a ação definida for diferente.

Na Figura 39 está representado o esquema de ligação do ESP-01 juntamente com os dois *shields*.

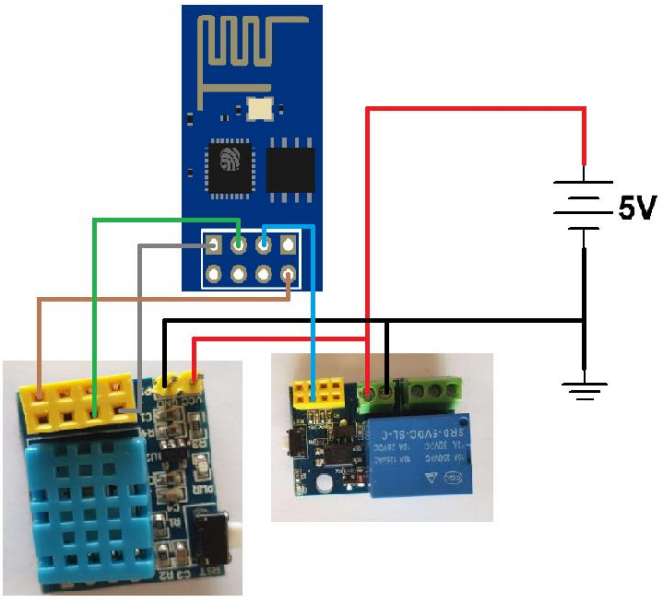

Figura 39 – Esquema sensor de temperatura mais relé

Os *shields* são alimentados com 5VDC, no entanto o ESP-01 é alimentado com 3,3VDC o que iria necessitar de uma fonte de alimentação externa, no entanto como os *shields* têm a tensão de 3,3VDC nos terminais acabamos por alimentar o ESP a um dos *shields*, neste caso usamos o *shield* DHT11 como alimentação do ESP-01.

O pino de controlo do relé é o GPIO0 enquanto que o pino de controlo do *shield* DHT11 é o GPIO2.

5. Conclusão

Com a realização deste projeto, obtivemos um amplo conhecimento sobre várias vertentes, desde as funcionalidades do ESP-01 e as suas diversas capacidades, também como a domótica, a programação em linguagem C++, *PHP* e *HTML* sendo estes últimos conhecimentos a nível das linguagens de programação que usamos quer nos códigos para o ESP como a elaboração do *website* e da aplicação para telemóvel.

Verificamos que com este projeto as vantagens são enormes no ponto de vista da redução do consumo, pois permite o controlo dos aparelhos e desligá-los quando ninguém está a usar pois por vezes deixam-se ligados para manter a sala quente ou por esquecimento e desta forma mais automatizada conseguimos controlar os possíveis esquecimentos e permitindo que não existam excessos de consumo.

O resultado deixa-nos satisfeitos, não só por se tratar de uma vitória de um projeto concluído, como pelo trabalho ardo-o e pelo trabalho em equipa a que nos propusemos fazer mesmo no meio das dificuldades que a vida traz a cada um de nós.

Este projeto não é algo que fica parado no tempo, podendo ser aperfeiçoado no futuro devido às novas aspirações do ser humano no que toca ao conhecimento da domótica e da necessidade de melhorar a eficiência energética para um menor gasto e maior conforto.

Existem várias ideias em mente que poderão ser alcançadas como o acrescentar de uma electroválvula eletrónica nos aquecedores permitindo o controlo quer físico como á distância. Também vemos como opção o acesso facilitado à mudança das credenciais da rede de *internet* por parte do consumidor permitindo ao cliente mudar as credenciais da rede de *internet* através da aplicação do telemóvel. Essa ideia foi explorada por nós e apesar de não termos conseguido colocar a ideia em prática acreditamos que seja possível ser feito, mas através de outro microcontrolador.

REFERÊNCIAS

[1] DGEG. "*Eficiência Energética*", https://www.dgeg.gov.pt/pt/areas-setoriais/energia/eficiencia-energetica/edificios/ (acedido em 22 de junho de 2023).

[2] Diário de Notícias. "*Mais de 122 milhões de euros para tornar edifícios mais sustentáveis*", https://www.dn.pt/dinheiro/mais-de-122-milhoes-de-euros-para-tornar-edificios-mais-sustentaveis-15948510.html (acedido em 22 de junho de 2023).

[3] Deco Proteste. "*Certificado energético: o que é, onde pedir e qual o preço*", https://www.deco.proteste.pt/dinheiro/comprar-vender-casa/noticias/certificado-energetico-que-e-onde-pedir-qual-preco (acedido em 22 de junho de 2023).

[4] Wikipédia. "*Domótica*" https://pt.wikipedia.org/wiki/Dom%C3%B3tica (acedido em 23 de junho de 2023).

[5] Alura. "*O modelo OSI e suas camadas*" https://www.alura.com.br/artigos/conhecendo-o-modelo-osi (acedido em 3 de julho de 2023).

[6] Maisimoes. "*Padrões de Comunicação*" http://masimoes.pro.br/site/redes/02_ModProt/1.2-modelos-2.htm (acedido em 14 de julho de 2023).

[7] Infowester. "*O que é Internet das Coisas (IoT)*?" https://www.infowester.com/iot.php (acedido em 14 de julho de 2023)

[8] Anacom. "*Utilização da Internet das Coisas (IoT - Internet of Things) 2022*" https://www.anacom.pt/render.jsp?contentId=1737942 (acedido em 4 de julho de 2023)

[9] MasterWalker. "*Como usar com Arduino – Módulo WiFi ESP8266 ESP-01*" https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-wifi-esp8266-esp-01 (acedido em 10 de junho de 2023)

[10] Botnroll. "*Módulo Serie WiFi ESP8266 ESP-01*" https://www.botnroll.com/pt/esp/1021-modulo-serie-wifi-esp8266.html (acedido em 4 de julho de 2023)

[11] Eletrogate. "*Adaptador DIP para ESP8266 ESP-01 com DHT11*" https://www.eletrogate.com/adaptador-dip-para-esp8266-esp-01-com-dht11 (acedido em 15 de maio de 2023)

[12] Random Nerd Toturials "*ESP8266 Pinout Reference: Which GPIO pins should you use?*" https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/ (acedido em 5 de abril de 2023)

[13] Makerhero. "*Módulo Sensor de Temperatura e Umidade DHT11 para ESP8266 ESP-01*" https://www.makerhero.com/produto/modulo-sensor-de-temperatura-e-umidade-dht11-para-esp8266-esp-01/ (acedido em 12 de abril de 2023)

[14] Circuit Geeks. "*DHT11 & DHT22 Humidity and Temperature Sensor with Arduino*" https://www.circuitgeeks.com/arduino-dht11-and-dht22-sensor-tutorial/ (acedido em 10 de abril de 2023)

[15] BotnRoll. "*ESP-01S Relay v1.0*" https://www.botnroll.com/img/cms/ESP-01S%20Relay%20v1.0%20Usage.pdf (acedido em 13 de abril de 2023)
[16] BotnRoll. "*Shield Relé ESP8266 ESP-01*" https://www.botnroll.com/pt/ethernet-wi-fi/3561-shield-rel-com-esp8266-esp-01.html (acedido em 15 de abril de 2023)
[17] BotnRoll. "*HC-SR501 PIR Montion Detector*" https://www.botnroll.com/img/cms/HC-SR501-ETC.pdf (acedido em 5 de maio de 2023)
[18] Makehero. "*O que é IDE Arduino?*" https://www.makerhero.com/blog/o-que-e-ide-arduino/ (acedido em 15 de julho de 2023)
[19] Pplware. "*myDevices Cayenne: O mundo do IoT no teu Raspberry PI*" https://pplware.sapo.pt/linux/mydevices-cayenne-o-mundo-do-iot-no-teu-raspberry-pi/ (acedido em 15 de julho de 2023)
[20] Techtudo. "O que é XAMPP e para que serve" https://www.techtudo.com.br/noticias/2012/02/o-que-e-xampp-e-para-que-serve.ghtml (acedido em 20 de junho de 2023)
[21] Escola Ninja. "*O que é: XAMPP*" https://blog.escolaninjawp.com.br/glossario/o-que-e-xampp/ (acedido em 22 de junho de 2023)
[22] RockContent. "*Saiba agora como hospedar um site no 000webhost*" https://rockcontent.com/br/blog/000webhost/ (acedido em 14 de julho de 2023)
[23] Zeroseuns. "*MIT App Inventor 2: O Que é e Como Funciona?*" https://zeroseuns.info/aplicativos/mit-app-inventor-2/ (acedido em 16 de julho de 2023)
[24] Autodesk Instructables. "USB to ESP-01 Adapter Board Modification", https://www.instructables.com/USB-to-ESP-01-Board-Adapter-Modification/ (acedido em 07 de junho de 2023).

Anexos

ANEXO 1 – DEMOSTRAÇÃO DO CÓDIGO DO RELÉ/DHT11

```
#define CAYENNE_PRINT Serial  // Utiliza a porta serial para imprimir as
informações do Cayenne
#include <CayenneMQTTESP8266.h>
#include <DHTesp.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

String URL = "http://projetoipv2023.000webhostapp.com/test_dataRele.php";
String URL1 = "http://projetoipv2023.000webhostapp.com/GetDataRele.php?id=2";
String URL2 = "http://projetoipv2023.000webhostapp.com/FuncHorario.php";
String URLsensor = "http://projetoipv2023.000webhostapp.com/test_data.php";

DHTesp dht;

// Informações da rede WiFi
char ssid[] = "NOMEDAREDE";  // nome da rede WiFi
char password[] = "SENHADAREDE" ;  // senha da rede WiFi

// Servidor Cayenne
char username[] = "488e9e60-b864-11ed-b72d-d9f6595c5b9d";
char mqtt_password[] = "619713b08f64f6b184ace5eddce56e8fc917b7a1";
char client_id[] = "3340b530-c259-11ed-b72d-d9f6595c5b9d";

WiFiClient wifiClient;

// Informações do shield relé
#define RELAY_PIN 0  // Pino do ESP8266 ligado ao pino de controle do relé
#define VIRTUAL_CHANNEL_RELAY 0  // Canal virtual do Cayenne para o relé

// Informações do sensor dht11
#define DHTPIN 2        // pino de dados do sensor
#define DHTTYPE DHT11   // tipo do sensor

// Pino do botão
#define BUTTON_PIN 9  // Pino do ESP8266 ligado ao botão físico

int buttonState = HIGH;  // Estado atual do botão
int lastButtonState = HIGH;  // Estado anterior do botão
String payload1;

int temperature3 = 0;
int humidity1= 0;
```

```
void setup() {
  // Inicialização da comunicação serial
  Serial.begin(115200);
  delay(100);
  connectWiFi();
  dht.setup(DHTPIN, DHTesp::DHT11);
  // Ligação à rede WiFi
  Cayenne.begin(username, mqtt_password, client_id, ssid, password);

  // Configuração do pino do relé
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH); // Desliga o relé inicialmente

  // Configuração do pino do botão
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Serial.println("Setup completo.");
}

void loop() {
  // Recebe as mensagens do Cayenne
  Cayenne.loop();
  delay(2000);
  float temperature = dht.getTemperature();
  float humidity = dht.getHumidity();
  float temperature1=temperature-5;
  Cayenne.celsiusWrite(1, temperature1);
  Cayenne.virtualWrite(2, humidity, "rel_hum", "p");

  // Lê o estado atual do botão
  buttonState = digitalRead(BUTTON_PIN);

  // Verifica se o botão foi pressionado (transição de HIGH para LOW)
  if (buttonState == LOW && lastButtonState == HIGH) {
    // Inverte o estado do relé
    digitalWrite(RELAY_PIN, !digitalRead(RELAY_PIN));
        atualizarDB();

    // Atualiza o estado do relé no Cayenne
    Cayenne.virtualWrite(VIRTUAL_CHANNEL_RELAY, digitalRead(RELAY_PIN));
  }

  // Armazena o estado atual do botão para comparação na próxima iteração
  lastButtonState = buttonState;

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }
```

```
 String postData = "temperature1=" + String(temperature1) + "&humidity=" +
String(humidity);

  HTTPClient http;
  http.begin(wifiClient,URLsensor);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpCode = http.POST(postData); // Verdadeiro
  String payload1 = http.getString();

   payload1 = "";

  if(httpCode > 0) {
      String payload1 = http.getString();
    }


  http.end();  //Fecha ligação
delay(2000);


if (temperature1 <= 26) { // Se a temperatura for inferior a 26 ativa os
horários

  HTTPClient httpfetch1;
  httpfetch1.begin(wifiClient, URL2); //Ligação com a base de dados para
verificar os horários
  httpfetch1.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpFetchCode1 = httpfetch1.GET();
  httpfetch1.end();  //fecha a ligação

delay(5000);
} else {

      digitalWrite(RELAY_PIN, HIGH); // Desliga o relé
      atualizarDB();
      delay(2000);
}

  HTTPClient httpfetch;
  httpfetch.begin(wifiClient, URL1); //Ligação com a base de dados para
atualizar o estado do relé
  httpfetch.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpFetchCode = httpfetch.GET();

  if (httpFetchCode == HTTP_CODE_OK) {

    String payload = httpfetch.getString();
```

```
    payload.trim();

    if (payload == "0")
    {
      digitalWrite(RELAY_PIN, HIGH); // Desliga o relé
    }
    else if (payload == "1")
    {
      digitalWrite(RELAY_PIN, LOW); // Liga o relé

    }

  }

  httpfetch.end();  //fecha a ligação

delay(2000);

}

//Função criada para atualizar o estado do rele na base de dados quando este é ligado/desligado através do caynne ou da temperatura ser superior á estipulada
void atualizarDB() {

  String postData1 = "estadorele=" + String(!digitalRead(RELAY_PIN));
// Guarda em string o estado do relé

  HTTPClient http1;
  http1.begin(wifiClient, URL);
  http1.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpSendCode = http1.POST(postData1); // Envia os dados da string para o ficheiro PHP
  http1.end();  //fecha a ligação
}

void connectWiFi() {

  WiFi.mode(WIFI_OFF);

  delay(1000);

  //Esta linha esconde a visualização do ESP como hotspot wifi
  WiFi.mode(WIFI_STA);

  WiFi.begin(ssid, password);
  Serial.println("A aceder ao WiFi");
```

```
  while (WiFi.status() != WL_CONNECTED) {



    delay(500);
  }
}

// Função para lidar com as mensagens recebidas do Cayenne
CAYENNE_IN(VIRTUAL_CHANNEL_RELAY) {

  int value = getValue.asInt();

  if (value == 0) {
    digitalWrite(RELAY_PIN, HIGH); // Desliga o relé
    atualizarDB();
  } else if (value == 1) {
    digitalWrite(RELAY_PIN, LOW); // Liga o relé
    atualizarDB();
  }
}
```

ANEXO 2 – DEMOSTRAÇÃO DO CÓDIGO DO SENSOR DE MOVIMENTO

```
#include <CayenneMQTTESP8266.h>
#define CAYENNE_PRINT Serial
#define CAYENNE_DEBUG


// Informações da rede WiFi
char ssid[] = "NOMEDAREDE";  // nome da rede WiFi
char wifiPassword[] = "SENHADAREDE" ;  // senha da rede WiFi

// Servidor Cayenne
char username[] = "488e9e60-b864-11ed-b72d-d9f6595c5b9d";
char password[] = "619713b08f64f6b184ace5eddce56e8fc917b7a1";
char clientID [] = "d07cbcf0-c410-11ed-b72d-d9f6595c5b9d";

#define SENSORPIR_PIN 2 // Não use os pinos digitais 0 ou 1, pois eles entram
em conflito com o uso do Serial // Neste caso, o sensor está ligado ao GPI02
#define VIRTUAL_PIN 3

void setup()
{
 pinMode(SENSORPIR_PIN, INPUT); // INPUT mode - Para receber informação do
sensor
 Serial.begin(9600);
 Cayenne.begin(username, password, clientID, ssid, wifiPassword); //Para
inicialização da coneção via MQTT
}

void loop()
{
 Cayenne.loop();
 checkSensor(); // Verificação do sensor
}

int previousState = -1;
int currentState = -1;
unsigned long previousMillis = 0;
void checkSensor() // O sensor é verificado a  cada 250 milisegundos
{
 unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= 250) {
  currentState = digitalRead(SENSORPIR_PIN);
 if (currentState != previousState) {
 Cayenne.virtualWrite(3 , currentState, "digital_sensor", "d");
```

```
 Cayenne.virtualWrite(4 , currentState, "digital_sensor", "d"); //Configuração
de mais que um canal virtual, para ter vários Widgets (formas gráficas de
representação dos valores adquiridos)
 Cayenne.virtualWrite(5 , currentState, "digital_sensor", "d");

 previousState = currentState;
 }
 previousMillis = currentMillis;
 }
}
```

ANEXO 3 – CÓDIGO DO FICHEIRO *"DATAINSERIR"*

```
<?php


$dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234",
"id20893257_projeto2023");
if(!$dbc) {
die("DATABASE CONNECTION FAILED:" .mysqli_error($dbc));
exit();
}
$db = "id20893257_projeto2023";
$dbs = mysqli_select_db($dbc, $db);
if(!$dbs) {
die("DATABASE SELECTION FAILED:" .mysqli_error($dbc));
exit();
}
$EscolheData = mysqli_real_escape_string($dbc, $_GET['EscolheData']);
$EscolheHora = mysqli_real_escape_string($dbc, $_GET['EscolheHora']);
$EscolheDataFim = mysqli_real_escape_string($dbc, $_GET['EscolheDataFim']);
$EscolheHoraFim = mysqli_real_escape_string($dbc, $_GET['EscolheHoraFim']);

$query = "INSERT INTO Horario (Data, Hora, DataFim, HoraFim)
VALUES ('$EscolheData', '$EscolheHora', '$EscolheDataFim', '$EscolheHoraFim')";


if(mysqli_query($dbc, $query)){
echo "Novos dados inseridos com sucesso";
}
else{
echo "ERRO: Não foi possivel executar". $query." ". mysqli_error($dbc);
}
mysqli_close($dbc);
?>
```

ANEXO 4 – CÓDIGO DO FICHEIRO *"FUNCHORARIO"*

```
<?php
// Configurações da base de dados
$servername = "localhost";
$username = "id20893257_projeto1";
$password = "*Qwerty1234";
$dbname = "id20893257_projeto2023";

// Ligação com a base de dados
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Falha na ligação com a base de dados: " . $conn->connect_error);
}

// Obtenção do valor atual da variável na tabela
$sql = "SELECT Data, Hora, DataFim, HoraFim FROM Horario WHERE id <> 1";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $estadoRele = 0; // Valor padrão inicial

    while ($row = $result->fetch_assoc()) {
        $dataDefinida = $row["Data"];
        $horarioDefinido = $row["Hora"];
        $dataFimDefinido = $row["DataFim"];
        $horarioFimDefinido = $row["HoraFim"];

        // Obter data e hora atual
        $dataAtual = date("Y-m-d");
        $horarioAtual = date("H:i");

        // Adicionar 1 hora ao horário atual
        $horarioAtual = date("H:i", strtotime($horarioAtual . " +1 hour"));

        if ($dataAtual == $dataDefinida && $horarioAtual >= $horarioDefinido &&
($horarioAtual <= $horarioFimDefinido && $dataAtual == $dataFimDefinido)) {
            $estadoRele = 1; // Atualiza o valor se a condição for verdadeira para algum horário
        }
    }
```

```
    // Atualizar o valor na tabela
    $sqlUpdate = "UPDATE dht11 SET estadorele = $estadoRele WHERE ID = 2";
    if ($conn->query($sqlUpdate) === TRUE) {
        echo "Valor atualizado com sucesso para $estadoRele.";
    } else {
        echo "Erro ao atualizar o valor: " . $conn->error;
    }
} else {
    echo "Nenhum resultado encontrado na tabela.";
}

$conn->close();
?>
```

ANEXO 5 – CÓDIGO DO FICHEIRO *"GETDATARELE"*

```
<?php

$hostname = "localhost";
$username = "id20893257_projeto1";
$password = "*Qwerty1234";
$database = "id20893257_projeto2023";

$conn = mysqli_connect($hostname, $username, $password, $database);

if (!$conn) {
    die("Ligação falhada: " . mysqli_connect_error());
}

// Verifica se o parametro "id" foi passado no URL
if (isset($_GET['id'])) {
    $id = $_GET['id'];

    // Executa a SQL query para receber dados baseados no parametro recebido
    $sql = "SELECT * FROM dht11 WHERE ID='$id'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        // Os dados foram encontrados
        $row = $result->fetch_assoc();
        $data = $row['estadorele']; // Coluna onde queremos ir buscar os dados

        echo $data; // Retorna os dados recebidos
    } else {
        // Os dados nao foram encontraods
        echo "Dados não encontrados";
    }
} else {
    // o parametro "id" não foi passado pelo URL
    echo "Parâmetro 'id' não recebido";
}

$conn->close();
?>
```

ANEXO 6 – CÓDIGO DO FICHEIRO *"LOGIN"*

```
<?php
session_start();

// Função para verificar se o utilizador está logado
function isUserLoggedIn()
{
    return isset($_SESSION['username']);
}

// Função para realizar o login
function login($username, $password)
{


    if ($username === 'Projeto' && $password === 'Projeto') {
        $_SESSION['username'] = $username;
        return true;
    }

    return false;
}

// Função para realizar o logout
function logout()
{
    // Limpa todos os dados da sessão
    session_unset();
    // Destroi a sessão
    session_destroy();
}

// Verificar se o formulário de login foi enviado
if (isset($_POST['login'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Chamar a função de login
    if (login($username, $password)) {
```

```
        // Redirecionar para a página protegida após o login bem-sucedido
        header("Location: protected_page.php");
        exit();
    } else {
        $loginError = "Nome de utilizador ou senha inválidos.";
    }
}

// Verificar se o utilizador está logado
if (isUserLoggedIn()) {
    // Se o utilizador já estiver logado, redirecione para a página protegida
    header("Location: protected_page.php");
    exit();
}
?>

<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <link                                                                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>

<body>
    <div class="container">
        <h2 class="title text-center">Login</h2>
    </div>
    <div class="container">
        <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
            <div class="form-group">
                <label for="username">Nome de utilizador:</label>
                <input    type="text"    class="form-control"    id="username"    name="username"
required>
            </div>
            <div class="form-group">
                <label for="password">Senha:</label>
```

```
            <input type="password" class="form-control" id="password" name="password" required>
        </div>
        <button type="submit" class="btn btn-primary" name="login">Login</button>
        <?php if (isset($loginError)) : ?>
            <p class="text-danger"><?php echo $loginError; ?></p>
        <?php endif; ?>
      </form>
   </div>

   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js" integrity="sha384-fbbOQedDUMZZ5KreZpsbe1LCZPVmfTnH7ois6mU1QK+m14rQ1l2bGBq41eYeM/fS" crossorigin="anonymous"></script>
</body>

</html>
```

ANEXO 7 – CÓDIGO DO FICHEIRO *"LOGOUT"*

```
<?php
session_start();

// Função para realizar o logout
function logout()
{
    // Limpa todos os dados da sessão
    session_unset();
    // Destroi a sessão
    session_destroy();
    // Redireciona para a página de login após o logout
    header("Location: login.php");
    exit();
}

// Chamar a função de logout
logout();
?>
```

ANEXO 8 – CÓDIGO DO FICHEIRO *“PROTECTED_PAGE”*

```
<?php
session_start();

// Função para verificar se o utilizador está logado
function isUserLoggedIn()
{
    return isset($_SESSION['username']);
}

// Função para realizar o logout
function logout()
{
    // Limpa todos os dados da sessão
    session_unset();
    // Destroi a sessão
    session_destroy();
}

// Verificar se o utilizador não está logado
if (!isUserLoggedIn()) {
    // Redirecionar para a página de login
    header("Location: login.php");
    exit();
}

// Verificar se o formulário de logout foi enviado
if (isset($_POST['logout'])) {
    logout();
    // Redirecionar para a página de login após o logout
    header("Location: login.php");
    exit();
}

// Verificar se o formulário de inserção foi enviado
if (isset($_POST['insert'])) {
    $dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234",
"id20893257_projeto2023");
```

```
    if (!$dbc) {
        die("Ligação falhada: " . mysqli_error($dbc));
        exit();
    }

    $data = $_POST['data'];
    $hora = $_POST['hora'];
    $datafim = $_POST['datafim'];
    $horafim = $_POST['horafim'];

    // Realizar a inserção na base de dados
    $query = "INSERT INTO `Horario` (`Data`, `Hora`, `DataFim`, `HoraFim`) VALUES ('$data', '$hora', '$datafim', '$horafim')";
    mysqli_query($dbc, $query);

    // Fechar a ligação com a base de dados
    mysqli_close($dbc);

    // Redirecionar para a página atual para evitar envio repetido do formulário
    header("Location: " . $_SERVER['PHP_SELF']);
    exit();
}

// Verificar se o parâmetro 'delete' está presente na URL
if (isset($_GET['delete']) && !empty($_GET['delete'])) {
    $deleteId = $_GET['delete'];

    // Ligação com a base de dados
    $dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234", "id20893257_projeto2023");

    if (!$dbc) {
        die("Ligação falhada: " . mysqli_error($dbc));
        exit();
    }

    // Executar a query de exclusão
    $query = "DELETE FROM `Horario` WHERE id = $deleteId";
    mysqli_query($dbc, $query);

    // Fechar a ligaçãocom a base de dados
```

```
    mysqli_close($dbc);

    // Redirecionar para a página atual para evitar envio repetido do formulário
    header("Location: " . $_SERVER['PHP_SELF']);
    exit();
}
?>

<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <link                                                                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <style>
        .telmo {
            margin-top: 50px
        }

        .delete-link {
            color: red;
        }
    </style>
</head>

<body>
    <div class="container telmo">
        <h2 class="title text-center">Visualização de Dados</h2>
        <p>Bem-vindo, <?php echo $_SESSION['username']; ?>!
            <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
                <button type="submit" class="btn btn-primary" name="logout">Logout</button>
            </form>
        </p>
        <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
            <div class="form-row">
                <div class="col-md-3">
                    <label for="data">Data:</label>
```

```
            <input type="text" class="form-control" name="data" id="data" placeholder="AA-MM-DD">
          </div>
          <div class="col-md-3">
            <label for="hora">Hora:</label>
            <input type="text" class="form-control" name="hora" id="hora" placeholder="HH:MM:SS">
          </div>
          <div class="col-md-3">
            <label for="datafim">Data Final:</label>
            <input type="text" class="form-control" name="datafim" id="datafim" placeholder="AA-MM-DD">
          </div>
          <div class="col-md-3">
            <label for="horafim">Hora Final:</label>
            <input type="text" class="form-control" name="horafim" id="horafim" placeholder="HH:MM:SS">
          </div>
          <div class="col-md-3 mt-3 mt-md-0">
            <button type="submit" class="btn btn-success" name="insert">Inserir</button>
          </div>
        </div>
      </form>
      <table class="table text-center">
        <thead class="thead-dark">
          <tr class="thead-dark">
            <th>Data Inicial</th>
            <th>Hora Inicial</th>
            <th>Data Final</th>
            <th>Hora Final</th>
            <th>Ação</th>
          </tr>
        </thead>
        <tbody>
          <?php
          $dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234", "id20893257_projeto2023");

          if (!$dbc) {
            die("Ligação falhada: " . mysqli_error($dbc));
            exit();
```

```
            }

            $query = "SELECT * FROM `Horario` where id<>1";
            $res = mysqli_query($dbc, $query);

            while ($data = mysqli_fetch_array($res)) :
            ?>
               <tr>
                  <td><?php echo $data['Data']; ?></td>
                  <td><?php echo $data['Hora']; ?></td>
                  <td><?php echo $data['DataFim']; ?></td>
                  <td><?php echo $data['HoraFim']; ?></td>
                  <td><a class="delete-link" href="?delete=<?php echo $data['id']; ?>" onclick="return confirm('Tem certeza que deseja apagar este item?')">Apagar</a></td>
               </tr>
            <?php endwhile; ?>
         </tbody>
      </table>
   </div>

   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>
   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js" integrity="sha384-fbbOQedDUMZZ5KreZ...

</body>

</html>
```

ANEXO 9 – CÓDIGO DO FICHEIRO *“READ”*

```
<?php

$dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234",
"id20893257_projeto2023");

if(!$dbc) {
die("Ligação falhada:" .mysqli_error($dbc));
exit();
}
$db = "id20893257_projeto2023";
$dbs = mysqli_select_db($dbc, $db);
if(!$dbs) {
die("DATABASE INCORRETA:" .mysqli_error($dbc));
exit();
}

$query = "SELECT * FROM `dht11`";
$res = mysqli_query($dbc,"SELECT * FROM `dht11`");
$row = mysqli_fetch_array($res);



if(mysqli_query($dbc, $query)){


}
else{
echo "ERRO: Não foi possivel executar". $query." ". mysqli_error($dbc);
}
?>
<!doctype html>
<html lang="en">
   <head>
   <meta charset="utf-8">
   <link                                                                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
```

```
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">

</head>
<Style>
   .telmo{margin-top:50px}
</Style>

<body>
  <div class="container telmo">
    <h2 class="title text-center">Visualização de Dados</h2>
  </div>
<div class="container telmo">

  <table class="table text-center">
    <thead class="thead-dark">
      <tr class="thead-dark">
        <th>Temperatura</th>
        <th>Humidade</th>
        <th>Estado do Relé</th>
        <th>Última Atualização</th>
      </tr>
    </thead>
    <tbody>
      <?php while($data = mysqli_fetch_array($res)): ?>
        <tr>
          <td><?php echo $data['temperature3']; ?></td>
          <td><?php echo $data['humidity1']; ?></td>
          <td><?php if($data['estadorele']==0){echo "Desligado";}else{echo "Ligado";} ?></td>
          <td><?php echo $data['datetime']; ?></td>
        </tr>
      <?php endwhile; ?>
    </tbody>
  </table>
  </div>
    <script   src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
    <script              src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"
integrity="sha384-
```

```
fbbOQedDUMZZ5KreZpsbe1LCZPVmfTnH7ois6mU1QK+m14rQ1l2bGBq41eYeM/fS"
crossorigin="anonymous"></script>
</body>

</html>

<?php
mysqli_close($dbc);
?>
```

ANEXO 10 – CÓDIGO DO FICHEIRO *"READHORARIO"*

```
<?php
$dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234", "id20893257_projeto2023");

if (!$dbc) {
    die("Ligação falhada: " . mysqli_error($dbc));
    exit();
}

$db = "id20893257_projeto2023";
$dbs = mysqli_select_db($dbc, $db);

if (!$dbs) {
    die("DATABASE INCORRETA: " . mysqli_error($dbc));
    exit();
}

if (isset($_GET['delete'])) {
    $id = $_GET['delete'];
    $deleteQuery = "DELETE FROM `Horario` WHERE `id` = $id";
    if (mysqli_query($dbc, $deleteQuery)) {

    } else {

    }
}

$query = "SELECT * FROM `Horario`";
$res = mysqli_query($dbc, $query);
$row = mysqli_fetch_array($res);

// Adicionando cabeçalhos de controle de cache
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Sat, 1 Jan 2000 00:00:00 GMT");
?>

<!doctype html>
<html lang="en">
```

```
<head>
    <meta charset="utf-8">
    <link                                                                    rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
    <style>
        .telmo {
            margin-top: 50px
        }

        .delete-link {
            color: red;
        }
    </style>
</head>

<body>
    <div class="container telmo">
        <h2 class="title text-center">Visualização de Dados</h2>
    </div>
    <div class="container telmo">
        <table class="table text-center">
            <thead class="thead-dark">
                <tr class="thead-dark">
                    <th>Data Inicial</th>
                    <th>Hora Inicial</th>
                    <th>Data Final</th>
                    <th>Hora Final</th>
                    <th>Ação</th>
                </tr>
            </thead>
            <tbody>
                <?php while ($data = mysqli_fetch_array($res)) : ?>
                    <tr>
                        <td><?php echo $data['Data']; ?></td>
                        <td><?php echo $data['Hora']; ?></td>
                        <td><?php echo $data['DataFim']; ?></td>
                        <td><?php echo $data['HoraFim']; ?></td>
```

```
                <td><a class="delete-link" href="?delete=<?php echo $data['id']; ?>" onclick="return confirm('Tem certeza que deseja apagar este item?')">Apagar</a></td>
            </tr>
            <?php endwhile; ?>
        </tbody>
    </table>
</div>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js" integrity="sha384-fbbOQedDUMZZ5KreZpsbe1LCZPVmfTnH7ois6mU1QK+m14rQ1l2bGBq41eYeM/fS" crossorigin="anonymous"></script>
</body>

</html>

<?php
mysqli_close($dbc);
?>
```

ANEXO 11 – CÓDIGO DO FICHEIRO *"TEST_DATA"*

```
<?php

$hostname = "localhost";
$username = "id20893257_projeto1";
$password = "*Qwerty1234";
$database = "id20893257_projeto2023";

$conn = mysqli_connect($hostname, $username, $password, $database);

if (!$conn) {
	die("Ligação falhada: " . mysqli_connect_error());
}

echo "Ligação coma base de dados bem sucedida<br>";

if(isset($_POST["temperature1"]) && isset($_POST["humidity"])) {

	$t = $_POST["temperature1"];
	$h = $_POST["humidity"];

		$sql = "UPDATE dht11 SET temperature3 = '$t', humidity1 = '$h', datetime = CURRENT_TIMESTAMP WHERE id='2'";

	if (mysqli_query($conn, $sql)) {
		echo "\nNovos dados guardados com sucesso";
	} else {
		echo "Erro: " . $sql . "<br>" . mysqli_error($conn);
	}
}

?>
```

ANEXO 12 – CÓDIGO DO FICHEIRO *"TEST_DATARELE"*

```
<?php

$hostname = "localhost";
$username = "id20893257_projeto1";
$password = "*Qwerty1234";
$database = "id20893257_projeto2023";

$conn = mysqli_connect($hostname, $username, $password, $database);

if (!$conn) {
	die("Ligação falhada: " . mysqli_connect_error());
}

echo "Ligação á base dados bem-sucedida<br>";

if(isset($_POST["estadorele"])) {

	$r = $_POST["estadorele"];

		$sql = "UPDATE dht11 SET estadorele = '$r', datetime = CURRENT_TIMESTAMP WHERE id = '2'";


	if (mysqli_query($conn, $sql)) {
		echo "\nNovos dados guardados com sucesso.";
	} else {
		echo "Erro: " . $sql . "<br>" . mysqli_error($conn);
	}
}

?>
```

ANEXO 13 – CÓDIGO DO FICHEIRO *"UPDATE"*

```
<?php

$dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234", "id20893257_projeto2023");

if(!$dbc) {
die("Ligação com a base de dados falhada:" .mysqli_error($dbc));
exit();
}
$db = "id20893257_projeto2023";
$dbs = mysqli_select_db($dbc,$db );
if(!$dbs) {
die("DATABASE INCORRETA:" .mysqli_error($dbc));
exit();
}

$estadorele = mysqli_real_escape_string($dbc, $_GET['estadorele']);

$query = "Update dht11 SET estadorele='1', datetime = CURRENT_TIMESTAMP where id='2'";

if(mysqli_query($dbc, $query)){
echo "Dados atualizados com sucesso";
}
else{
echo "ERRO: Não foi possivel executar". $query." ". mysqli_error($dbc);
}
mysqli_close($dbc);
?>
```

ANEXO 14 – CÓDIGO DO FICHEIRO *"UPDATE2"*

```
<?php

$dbc = mysqli_connect("localhost", "id20893257_projeto1", "*Qwerty1234", "id20893257_projeto2023");
if(!$dbc) {
die("Ligação com a base de dados falhada:" .mysqli_error($dbc));
exit();
}
$db = "id20893257_projeto2023";
$dbs = mysqli_select_db($dbc,$db );
if(!$dbs) {
die("DATABASE INCORRETA:" .mysqli_error($dbc));
exit();
}

$estadorele = mysqli_real_escape_string($dbc, $_GET['estadorele']);

$query = "Update dht11 SET estadorele='0', datetime = CURRENT_TIMESTAMP where id='2'";

if(mysqli_query($dbc, $query)){
echo "Dados atualizados com sucesso";
}
else{
echo "ERRO: Não foi possivel executar". $query." ". mysqli_error($dbc);
}
mysqli_close($dbc);
?>
```