UNIVERSIDADE
LUSÓFONA

Centralized System for Home Automation

Trabalho Final de Curso

Francisco Loureiro Castel-Branco (21701361)

www.ulusofona.pt

Francisco Loureiro Castel-Branco

Orientado por Pedro Arroz Correia Bonifácio Serra

Trabalho Final de Curso | LEI | 25 de Junho de 2018

Direitos de cópia

À Filipa

e a todos o que me acompanharam na aventura deste ano letivo.

Índice

Resumo

A presente dissertação pretende explorar o tema *Home Automation* através da conjugação da criação de módulos que funcionam como interruptores, que fazem de interface para ligar e desligar componentes eléctricos numa residência, e uma aplicação web que tem como finalidade a gestão da habitação, assim como controlo remoto dos mesmos componentes.

É possível acrescentar ou alterar os módulos dinamicamente (sem reiniciar o sistema), atribuir os diversos interruptores físicos (*relays*) e observar as alterações de estado de forma instantânea.

A aplicação prática deste sistema consiste na conjugação de várias bibliotecas da plataforma *NodeJS*, como *onoff*, *ExpressJS*, *dhcp* e *mongoose*. Desta forma é possível criar um sistema com todas as funcionalidades integradas na mesma aplicação, sem entrar em incoerências e dependências.

Palavras-Chave: Aplicação Web, interruptor, *MongoDB*, *NodeJS*, *relay*, biblioteca, *dhcp*, *relay*

Abstract

The present dissertation explores the topic of Home Automation through modules that enable the controlling of physical switches through a web application. These switches can turn on/off electrical components in a residence.

It is possible to add or change modules dynamically (without restarting the system), assigning the various physical switches (relays) and observing the status changes instantly.

The practical application of this system consists in the conjugation of several libraries of the NodeJS platform, such as *onoff*, *ExpressJS*, *dhcp* and *mongoose*. In this way it is possible to create a system with all the functionalities integrated in the same application, without inconsistencies nor dependencies.

Keywords: Web application, switch, *MongoDB*, *NodeJS*, *relay*, library, dhcp

1. Introdução

Este projeto foi uma proposta de tema para o Trabalho Final de Curso, cadeira da Licenciatura em Engenharia Informática da Universidade Lusófona de Humanidades e Tecnologias e visa explorar a informatização de uma habitação de forma a possibilitar o controlo da mesma de forma remota, sem comprometer aspetos visuais na mesma.

IoT (*Internet of Things*) é um tema bastante falado atualmente. Desde automatização de sistemas de rega até ao controlo pelo smartphone ou por voz, podemos então controlar o nosso ambiente através do dispositivo que temos mais próximo de nós.

Para isto, foi concebido um módulo-interruptor que se liga ao sistema central (**HACS**) e lhe são atibuídos os relés (*relays*) préviamente instalados em série com os disjuntores do quadro elétrico. Estes módulos funcionam por *IP* (*Internet Protocol*) e MQTT (*Message Queuing Telemetry Transport*) e estão isoladas da rede habitacional, o que permite ter maior controlo na segurança.

O sistema central foi desenvolvido sobre um *RaspberryPi* que permite a interface directa com circuitos integrados e com relés, através do *GPIO* (*General Purpose Input and Output*) e, visto que também contém uma porta *Ethernet*, pode corre um servidor DHCP para interagir com os módulos acima referidos.

2. Identificação dos Requisitos

Para serem identificados os requisitos, foi necessário estudar diferentes abordagens para o tema. As hipóteses consistiam em centralizar o controlo dos interruptores (*Figura 2.1, Modelo 2*), ou criar interruptores independentes do sistema de controlo dos *Relays* (*Figura 2.1, Modelo 1*).

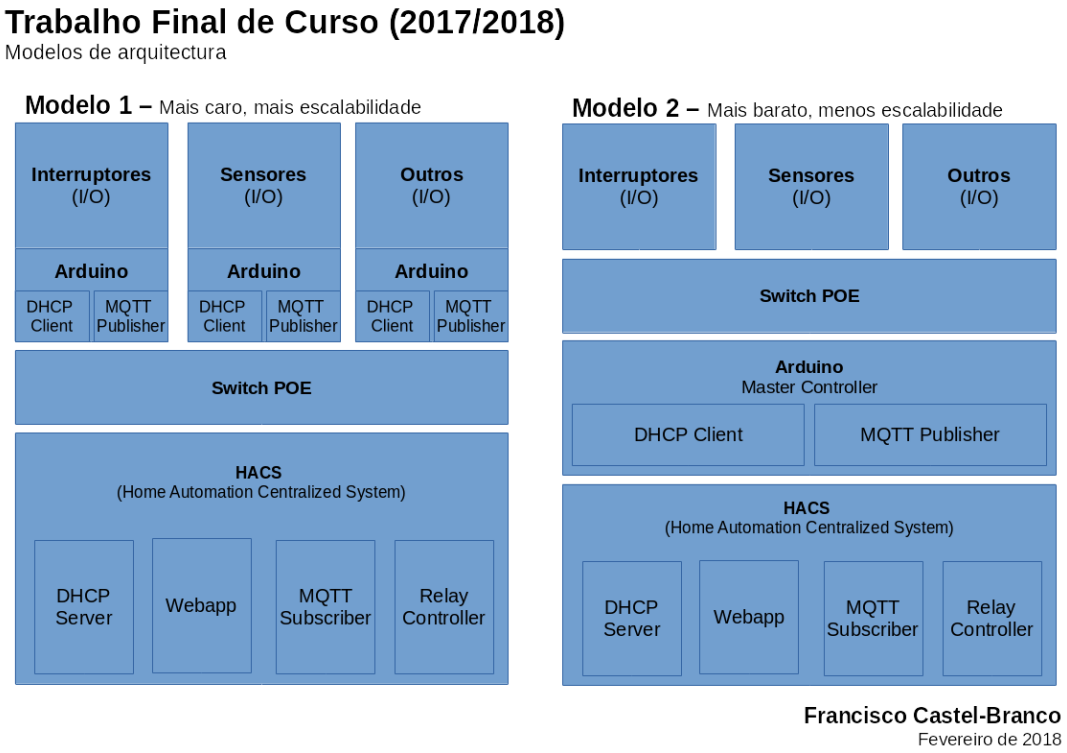

Figura 1: *Modelos de arquitectura estudados para desenvolvimento do projeto.*

Conforme estruturado na Figura 2.1, o sistema do Modelo 1, mais caro, permite maior escalabilidade pois podem ser acrescentados qualquer tipo de interruptor ou sensor à rede isolada do sistema. O sistema do Modelo 2, mais barato, apenas permite um limitado número de inputs, condicionando a localização e utilização destes periféricos. Uma vez que a escalabilidade é uma das principais preocupações dos sistemas *IoT*, foi escolhido o Modelo 1 para o desenvolvimento deste trabalho.

Foi escolhido o desenvolvimento sobre NodeJS para as funcionalidades do sistema por disponibilizar diversas bibliotecas que permitem a utilização de uma só linguagem de programação: Javascript. Desenvolvendo sobre NodeJS é possível

- utilizar o protocolo MQTT
- servidor DHCP
- comunicar com quase todas as bases de dados existentes atualmente no mercado

- controlar relays (utilizando um *RaspberryPi* ou semelhante);

Os interruptores são baseados no Arduino Leonardo, com POE (*Power Over Ethernet*) de standard **802.3af** em conjunto com os respectivos injetores e são programados na linguagem C/C++.

2.1. Lista de Requisitos

A

Tabela **1** apresenta os requisitos do sistema.

***Tabela 1**: Requisitos do sistema*

RF01	O sistema deve implementar um sistema de utilizadores e grupos
RF02	A aplicação web deve permitir ligar e desligar interruptores e visualizar qualquer alteração de forma instantânea.
RF03	O sistema deve guardar o estado dos interruptores de forma persistente.
RF04	O sistema deve alterar o estado dos interruptores mediante *input* do protocolo MQTT, após a atribuição dos botões aos interruptores pretendidos.
RF05	O utilizador deve conseguir consultar qual a primeira data de integração de um módulo num sistema.
RF06	Um utilizador que pertença ao grupo de administradores deve conseguir criar e eliminar utilizadores e/ou grupos de utilizadores.
RNF01	Desenvolver o sistema sob a plataforma NodeJS [1], utilizando a linguagem *JavaScript* [2], para o servidor e em C [3] para os módulos baseados em *Arduino Leonardo.*
RNF02	A base de dados persistente deverá ser MongoDB [4]
RNF03	O sistema deverá conseguir mostrar físicamente o resultado das alterações de estado dos interruptores.

3. Enquadramento teórico e estado da arte

Frequentemente são lançados no mercado produtos nesta área. Luzes controladas por voz, controlo remoto, alterar o ambiente e muito mais. Neste capítulo iremos abordar alguns produtos e comparar com a abordagem pretendida nesta dissertação:

3.1. Belkin Wemo [4]

A empresa Belkin disponibiliza inúmeros dispositivos para tornar uma casa "inteligente":

- WiFi Smart Light Switch

Embora o conceito seja semelhante, este dispositivo opera sem fios e bateria(s).

Vantagens: Fácil aplicabilidade; fácil reposição

Desvantagens: Mais caro; é necessário a utilização de baterias; depende de um Wireless Access Point.

Solução proposta: Substituir os interruptores comuns por módulos alimentados por POE

- Insight Smart Plug

Possibilita a interação com tomadas elétricas.

Vantagens: Operar vários dispositivos que podem estar ligados à mesma tomada; boa solução para candeeiros de mesa

Desvantagens: Ocupa espaço visível, bastante caro

Solução Proposta: Alternar o estado da corrente diretamente no quadro elétrico, através de relés (*relays).*

3.2. Xiaomi Smart Home [6]

Também procura aumentar a inteligência de uma casa.

Vantagens: Permite ligar, desligar equipamentos e luzes;

Desvantagens: São acrescentos ao que já existe.

Solução proposta: Substituir o que já existe ou aplicar de raiz. Torna o sistema mais limpo e integrado à própria casa.

3.3. Logitech Harmony Elite [5]

Este comando permite controlar diversas atividades ou ações através de Bluetooth, Wi-Fi, Zigbee, Z-Wave ou por infravermelhos.

Vantagens: Controlo centralizado de diversos dispositivos independentes, substitui outros comandos à distância.

Desvantagens: Pouco prático para vários utilizadores ou controlo fora de casa; um comando por cerca de $280 [6].

Solução proposta: Webapp que pode estar disponível para o exterior através de VPN ou IP público e pode ser utilizada por diversos utilizadores simultâneamente.

3.4. EDP re:dy [7]

A EDP (Energias de Portugal) pretende, com esta solução, tornar a utilização energética doméstica mais eficiente. Consiste numa abordagem informatizada para calcular e monitorizar gastos e produções de energia numa habitação.

As suas principais funcionalidades são:

- Análise dos consumos de energia
- Monitorização da produção de energia solar
- Análise dos consumos associados ao carro elétrico

Contudo, também implementa a componente de automatização. Esta é obtida através de equipamentod *add-on*, como a *edp re:dy plug*.

Vantagens: Controlo centralizado de diversos dispositivos independentes, através de uma aplicação móvel que pode ser acedida em qualquer altura, com acesos à internet.

Desvantagens: *Add-ons* na habitação, Monitorização das habitações por entidades externas.

Solução proposta: Sistema interno, com acesso à internet apenas se o cliente assim o desejar.

É possível concluir que o maior erro das soluções disponíveis no mercado é não só o elevado custo, como também a fraca integração na casa.

A solução que esta dissertação apresenta tem por objetivo primário manter o visual de uma habitação igual, ou semelhante, ao de uma sem qualquer tipo de sistema inteligente, acrescentando um leque de funcionalidades e formas diferentes de alcançar os mesmos fim.

4. Método e planeamento

Com base na identificação de requisitos foi este o plano delineado:

1. Modelos de Arquitectura
2. Definir tecnologias a serem utilizadas para o sistema centralizado
    1. NodeJS (WebService)
    2. MongoDB (Base de Dados)
3. Definir dados relevantes a guardar na base de dados
    1. ID do módulo (ou MAC Address)
    2. Associação botões/funcionalidades
    3. Utilizadores

Assim é possível explorar diversas tecnologias de diferentes planos curriculares da Licenciatura em Engenharia Informática:

1. Sistemas Distribuidos

Para que o sistema seja robusto, é necessário distribuir o sistema por módulos de modo a prevenir falhas dispendiosas e também permite a sua escalabilidade.

Assim utilizamos **DHCP** para servir uma gama de **endereços IP** aos interruptores e sensores de modo a guardar numa base de dados a correspondência entre **MAC Address** e funcionalidades atribuídas, assim como os seus componentes.

2. Bases de dados

Foi opção utilizar uma base de dados **NoSQL**. É de todo pertinente abordar um tema atual com uma base de dados de tecnologia recente e de eficiente utilização.

3. Aplicação Web

A fim de ser viável para uso doméstico, é necessário criar uma interface leve e simples para qualquer pessoa poder utilizar. A forma mais simples é uma interface Web responsiva que não altera o *workflow* do utilizador quando utiliza qualquer equipamento com um browser.

4. POE (*Power Over Ethernet)*

Implementando **POE IEEE 802.3af** nos interruptores simplifica a instalação dos mesmos. Basta apenas um cabo de standard **IEEE 802.3** para realizar tarefas de comunicação com o controlador principal e de alimentação elétrica de 48v DC, que é bastante mais segura de manuseamento que norma 220v AC, que existe em Portugal.

5. Arduino Leonardo com placa de rede W5500

Em comparação com o Arduino Uno, o Leonardo comunica por USB directamente, sem ser necessário um SPI (Serial Peripheral Interface) independente do controlador.

O método de abordagem ao plano definido foi:

1. Desenvolvimento do Web Service com ações *dummy* que executem o máximo de funções possível (funções vazias, ou que retornem mensagens na consola) em conjunto com a base de dados e utilizadores;
2. Utilizar um servidor DHCP em conjunto com o Web Service para registar todas as ações dos módulos ligados ao sistema, assim como a sua disponibilidade (*online* ou *offline*);
3. Associar a informação obtida do servidor DHCP aos *relays* disponíveis
4. Criar ligações com lâmpadas, tomadas, fechaduras, etc

4.1.Modelos de dados

Os modelos de dados permitem compreender a estrutura para a qual o programa foi desenhado.

1. Os utilizadores contêm uma lista de grupos a que estão associados
2. Os grupos limitam a informação que pode ser acedida pelo utilizador
3. Os *relays* contêm uma lista de grupos a que estão associados
4. Ao acederem pela primeira vez ao sistema, os módulos completam alguma da informação na sua estrutura de dados. O utilizador tem que completar a configuração.
5. Os módulos contém um *array* de *relays*. Cada posição neste array corresponde ao número do botão desse módulo.

   Por exemplo: buttons[0] corresponde ao botão nº1 (primeria posição) e este elemento é um *relay*. Isto facilita a pesquisa por atribuição.

```
const relayModel = {
  name: {
    type: String,
    required: true,
    unique: true
  },
  state: {
    type: Boolean,
    required: true,
    default: false
  },
  rpiPin: {
    type: Number,
    required: true,
    default: -1
  },
  groups: {
    type: [{ type: Schema.Types.ObjectId, ref: 'Group' }],
    required: false,
    default: []
  }
};
```

SourceCode1: *Modelo de dados dos relays*

```
const relayModel = {
   name: {
     type: String,
     required: true,
     unique: true
   },
   state: {
     type: Boolean,
     required: true,
     default: false
   },
   rpiPin: {
     type: Number,
     required: true,
     default: -1
   },
   groups: {
     type: [{ type: Schema.Types.ObjectId, ref: 'Group' }],
     required: false,
     default: []
   }
 };
```

SourceCode2: *Modelo de dados dos grupos de utilizadores*

```
const userModel = {
  username: {
    type: String,
    required: true,
    unique: true
  },
  fullname: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  deletable: {
    type: Boolean,
    required: true,
    default: true
  },
  groups: {
    type: [{ type: Schema.Types.ObjectId, ref: 'Group' }],
    required: false
  }
};
```

SourceCode3: *Modelo de dados dos utilizadores*

```
const moduleModel = {
  name: {
    type: String,
    required: true,
    unique: true
  },
  firstAttatched: {
    type: String,
    required: true,
    default: new Date()
  },
  buttons: {
   type: [{type: Schema.Types.ObjectId, ref: 'Relay' }],
   required: true,
   default: []
  },
  buttonCount: {
    type: Number,
    required: true,
    default: -1
  }
};
```

SourceCode4: *Modelo de dados dos módulos*

5. Resultados

Apenas os utilizadores autorizados podem aceder à aplicação:

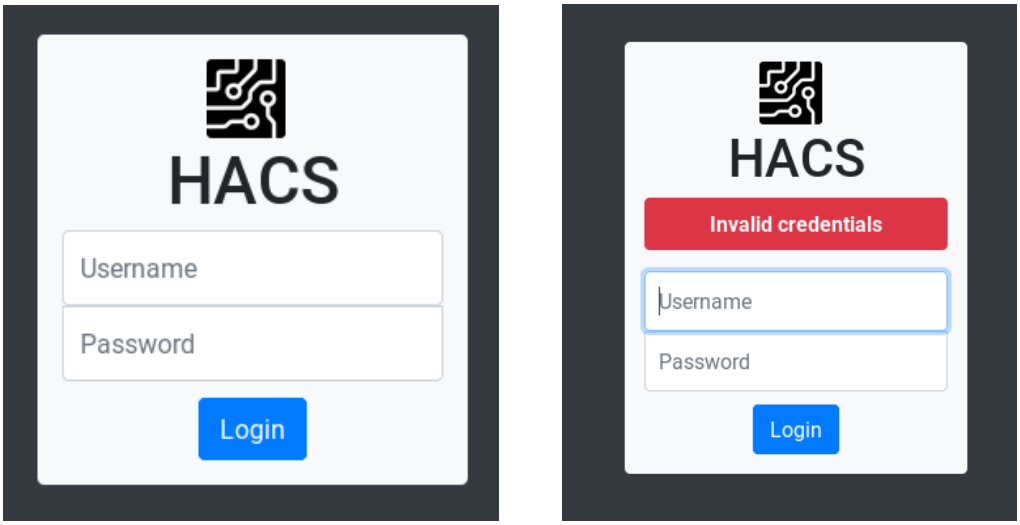

Figura 2: *Método de autenticação*

É possível alterar o estado dos *relays* através da aplicação web e, como podemos verificar, a aplicação é *responsive*:

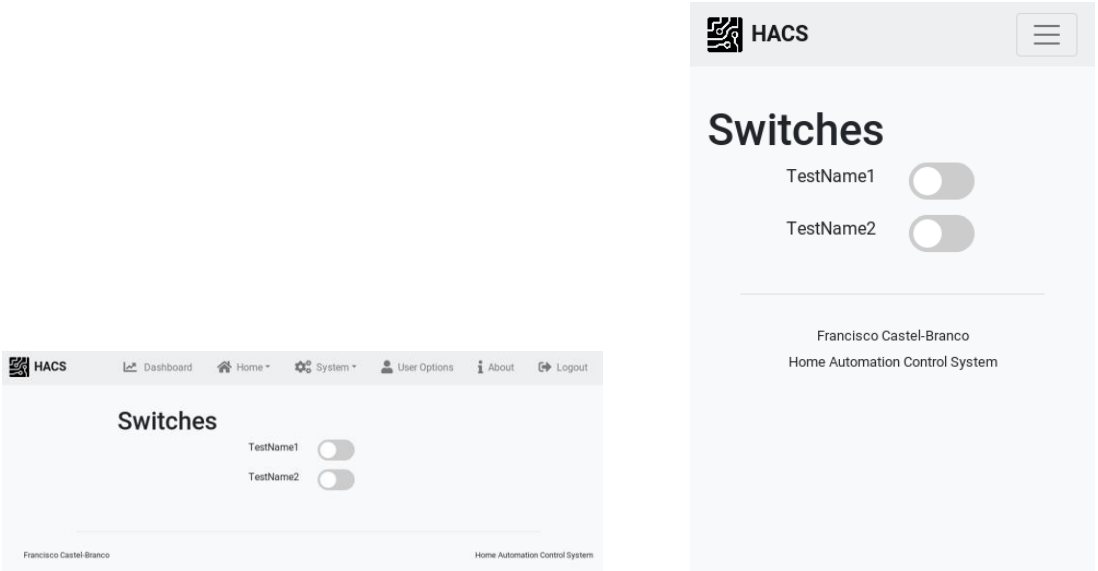

Figura 3: *Método de alteração do estado dos interruptores*

Nas Figuras 4, 5, 6 e 7 podemos verificar a forma como é possível gerir os módulos:

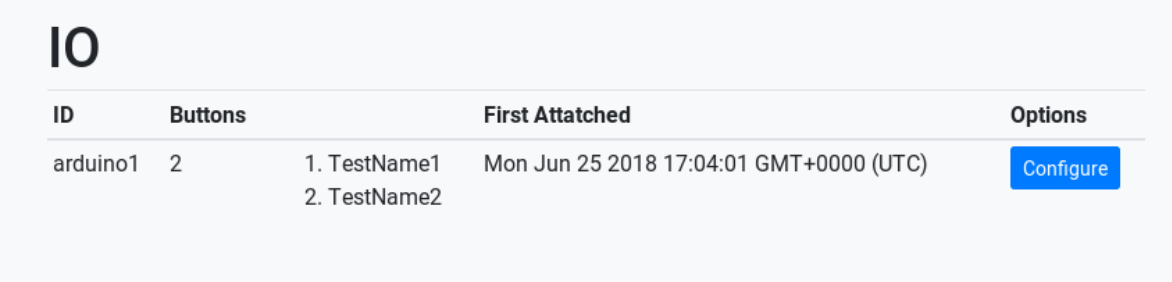

Figura 4: *Gestão de módulos.*

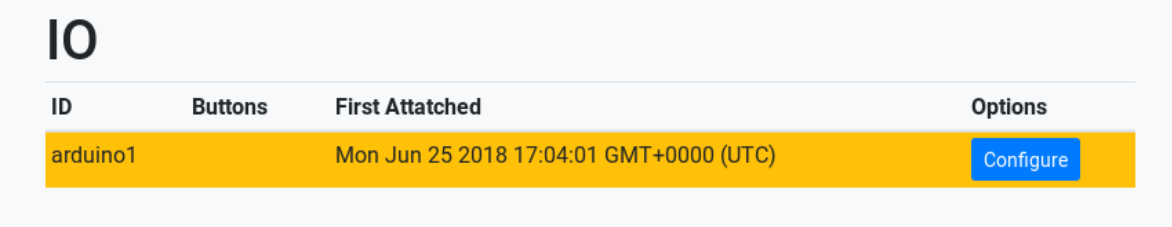

Figura 5: *Gestão de módulos com um novo módulo no sistema.*

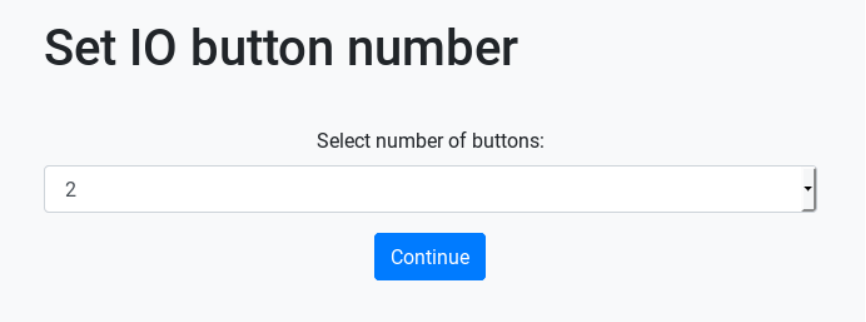

Figura 6: *Configuração inicial do número de botões que um módulo tem. Só é configurado uma vez.*

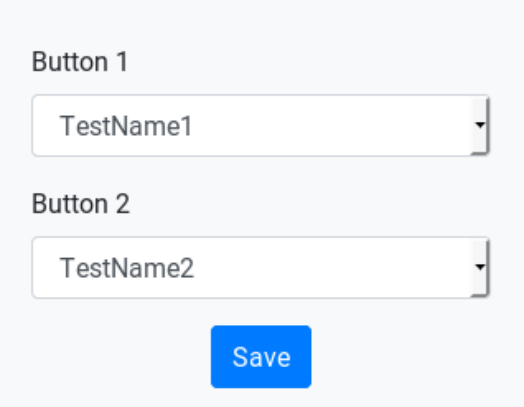

Figura 7: *Configuração dos botões do módulo. Referem os relays da Figura 3*

Na Figura 8 podemos verificar que o protótipo é algo semelhante a um interruptor comum:

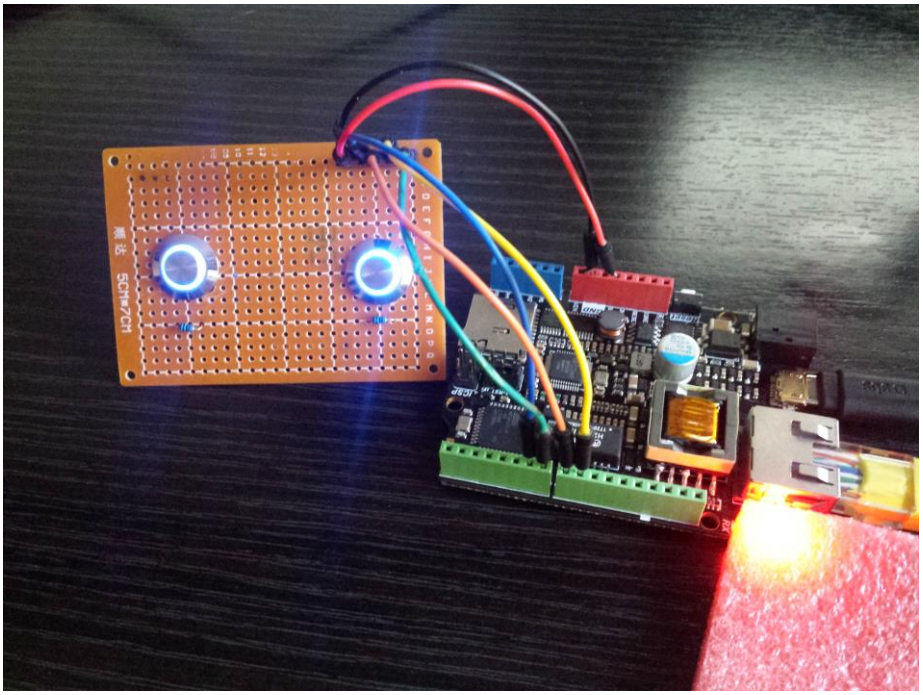

Figura 8: *Configuração dos botões do módulo. Referem os relays da figura 4*

Como é possível verificar pelas figuras acima, o método de configuração de um módulo (figura 8) é bastante simples.

6. Conclusão e trabalho futuro

Este trabalho permitiu a exploração na área de IoT. Um conceito simples mas de bastante complexa implementação. Foi necessário configurar desde o *backend* (*Firewall*, servidor DHCP, servidor HTTP e base de dados) até à interface gráfica. Esta última teria de ser o mais intuitiva possível.

No final deste trabalho passou a ser possível configurar módulos *plug and play* que serviam de substituto aos interruptores comuns das habitações, que era o principal objetivo. Assim passou a ser possível não só controlar luzes da forma convencional (através do módulo) como também controlar estas ações através de uma interface gráfica que também disponibiliza informações "ao vivo" da casa.

É possível, também, alterar o número de botões de uma determinada divisão. Assim que o módulo for alterado, podemos configurar cada botão de acordo com o que for desejado pelo utilizador.

Numa fase futura, gostaria de melhorar a gestão destes módulos, acrescentar funcionalidades, como alarmes que disparam as luzes de dada divisão e são desligados através de um botão, ou sensores de proximidade que permitem o *stand by* dos módulos de modo a poupar energia, e também o possível desenvolvimento de uma, ou mais, aplicações móveis nativas para receber notificações *push* de modo a alertar o utilizador dado um determinado evento. Se possível, também gostaria de estudar as possibilidades comerciais de um produto desta natureza.

Bibliografia

Internet Documents

Internet Encyclopedia

[1]

https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers

[Accessed Apr. 22, 2018]

Internet Encyclopedia

[2]

https://en.wikipedia.org/wiki/Internet_of_things [Accessed Apr. 22, 2018]

Internet Blog

[3]

https://blog.nationwide.com/9-wifi-home-automation-apps/ [Accessed Apr. 28, 2018]

Belkin Wemo - Home Automation

[4]

http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/

[Accessed Apr. 28, 2018]

Logitech Harmony Elite

[5]

https://www.pcmag.com/review/349593/logitech-harmony-elite

[Accessed May 4, 2018]

Análise Xiaomi Smart Home

[6] H. Cura, “pplware”, August, 2017

https://pplware.sapo.pt/gadgets/analise-kit-xiaomi-smart-home-casa-inteligente/

[Accessed May 4, 2018]

Glossário

DHCP	*Dynamic Host Configuration Protocol* – Atribui dinâmicamente um endereço IP a um dispositivo
MAC Address	M*edia Access Control Address* – Endereço único por interface de rede
SQL	*Simple Query Language* – Linguagem utilizada nas bases de dados mais comuns
NoSQL	Tipo de base de dados que não utiliza SQL
POE	*Power Over Ethernet* – Protocolo que transporta energia elétrica através de cabos de rede
IEEE	*Institute of Electrical and Electronics Engineers* – Organização de engenheiros que também normaliza interfaces, nomenculaturas, valores padrão, etc.
IoT	*Internet Of Things* – Conceito que interliga vários tipos de dispositivos para aumentar ou facilitar o quotidiano
RF	Requisito Funcional
RNF	Requisito Não Funcional
Relay	Interruptor que é accionado mediante uma corrente, neste caso de baixa voltagem. Fecha o circuito de 220v através da energia de 5v do RaspberryPi
RaspberryPi	Microcomputador de arquitectura ARM.
Arduino	Microcontrolador (não corre um sistema operativo, apenas corre código-máquina, compilado de C)
C	Linguagem de programação
NodeJS	Framework de desenvolvimento
HTTP	HyperText Transfer Protocol, protocolo de comunicação utilizado pelos browsers como Mozilla Firefox e Google Chrome

Anexos

I. Manual técnico da aplicação

Nota: Este manual é referente à instalação sobre a distribuição *Raspbian* (GNU/*Linux*) Stretch, à data desta dissertação.

Para instalar esta aplicação, é necessário:

- NodeJS v8.1.x+ e NPM (Node Package Manager) respetivo
- MongoDB v3.x+
- RaspberryPi 2 (ou superior)

Nota Importante: É imperativo que a base de dados esteja ativa no porto default antes da execução de qualquer programa abaixo referido

1. Aceder à pasta **hacs-webapp/** e executar o comando **npm install** para instalar as dependências do projeto.

2. Carregar as definições de teste da base de dados utilizando o comando **npm run setup all**.

Nota: É possível apagar as informações da base de dados com o comando **npm run setup clean**.

3. Abrir os portos 3000 (Webserver), 1883 (MQTT) e 67 (DHCP).

Nota: O RaspberryPi deve estar configurado para aceder à rede privada por WiFi. A porta de rede é utilizada para servir os módulos. É necessário ligar esta porta a um *switch* para isolar estes componentes.

4. Correr a aplicação com o comando **npm start** . É **obrigatório** correr com direitos de administrador (*root*). Se assim não for, a aplicação fecha, devolvendo um erro. Isto deve-se ao servidor DHCP.

Para aceder à aplicação web, [IP]:3000 através de uma máquina da rede privada a que o RaspberryPi está ligado (Wifi na nota do ponto 3). **Utilizador: "admin", Password: "admin"**