António Pedro Pontinha Mendes

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Controlador Cognitivo para Estores e Luminosidade baseado em IoT e Base-de-Dados em Cloud

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador: Rui Santos-Tavares, Professor Auxiliar,
Universidade Nova de Lisboa

Júri

Presidente: Doutor Fernando José Almeida Vieira do Coito
Arguentes: Doutora Anikó Katalin Horváth da Costa
Doutor Rui Manuel Leitão Santos Tavares

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2017

Controlador Cognitivo para Estores e Luminosidade baseado em IoT e Base-de-Dados em Cloud

Para Vocês,
Família, Catarina e os que já não estão entre nós...

Agradecimentos

Agradeço ao Professor Rui Tavares por me ter proposto este desafio e ter sempre mostrado a sua disponibilidade para me ajudar.

Agradeço, de igual forma, à casa onde passei 5 anos da minha formação, a Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

Aos meus colegas de curso e amigos, os quais passaram comigo este tempo. Em especial ao Duarte Gonçalves pela sua disponibilidade e amizade, ao Dimo Naydenov por me mostrar que há sempre algo mais a aprender ou a melhorar e ao Filipe Perestrelo, que me ajudou no desenvolvimento da tese, por termos um projecto semelhante. A todos os outros, um obrigado também.

Ainda, agradeço à minha família mais próxima, em particular à minha Mãe, Pai e Irmã, que sempre me apoiaram e acreditaram em mim. Espero que tenham ficado orgulhosos de ter conseguido esta meta, porque sem o vosso apoio e companhia seria tudo mais difícil! Agradeço também, a todos os restantes membros da minha família, em particular à minha Madrinha Manuela e Prima Ana Lúcia pelo apoio e preocupação que sempre demonstraram. Ao meu tio Zé (está vingado tio) e tia Berta pela vossa ajuda e disponibilidade. E por fim, à minha Avó Materna Adelina, ao meu Padrinho Armando, ao meu avô António, à minha Avó Fernanda e ao meu Avô Pontinha que, apesar de já não estarem aqui, fizeram parte de mim e fizeram de mim (em particual a minha avó Adelina), a pessoa que sou hoje.

E por fim, mas não menos importante, agradeço a ti Catarina, pelo apoio, companhia, ajuda e empenho que me deste e por tornares esta caminhada mais fácil! Vou sentir saudades de te ir levar a casa depois das aulas. E ainda à tua família, que sempre me apoiou e acreditou em mim. Obrigado Alda, Carlos, Daniel, João e Dona Fátima (estará sempre connosco).

OBRIGADO A TODOS!

Resumo

Atualmente, o funcionamento da maioria dos estores elétricos, implica que o utilizador carregar num interruptor manual para abrir ou fechar o estore. Isto implica que, se o utilizador desejar, por exemplo, que o estore fique 50 % fechado, terá que aguardar junto do interruptor até que o estore se coloque na posição desejada. Ou seja, não é um processo rápido nem confortável caso o utilizador tenha vários estores.

Para criar um maior conforto ao consumidor, a solução proposta, através de uma abordagem *Internet-of-Things*, consistirá na realização de um sistema controlado por um microprocessador, que atuará no motor do estore e estará conectado a alguns sensores. Mas a grande novidade será a introdução de um sensor de distância colocado no estore que permitirá ao utilizador, através de uma aplicação *smartphone*, colocar o estore na posição desejável, no conforto do seu sofá ou fora da habitação.

Palavras-chave: Estores automatizados; *CLOUD*; *IoT*; Comunicação WiFi; *Android*

Abstract

Most of the automatic blinds nowadays, depends on the user to press a switch to open or close the blind. This implies that, if the user wants, for example, the blind to stay 50 % closed, the user has to keep pressing the switch until the blind is in the desired position. This means, if the user has multiple blinds, this is a slow and uncomfortable process.

So, to bring more comfort to the user, the designed solution, based on IoT approach, will consist in a system controlled by a microprocessor, that will act in the blind's motor and it will be connected to some sensors. But the big innovation it will be the use of a distance sensor, placed in the blind, that allows the user, through a smartphone app, to set the blind at the desired position, in the confort of your couch or outside the house.

Keywords: Automatic Blinds; Internet-of-Things; *Cloud*; WiFi Communication; Android

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Siglas

ADC	*Analog-to-Digital Converter*.
DC	*Direct Current*.
GPIO	*General Purpose Input/Output*.
HTTP	*Hypertext Transfer Protocol*.
IoT	*Internet of Things*.
IV	Infravermelhos.
KVL	*Kirchhoff Voltage Law*.
LDR	*Light Dependent Resistor*.
LED	*Light Emitting Diode*.
LPWAN	*Low-Power Wide Area Network*.
MQTT	*Message Queuing Telemetry Transport*.
OFDM	*Orthogonal frequency-division multiplexing*.
PCB	*Printed Circuit Board*.
SoC	*System on Chip*.
TJB	Transístor de Junção Bipolar.

UML *Unified Modeling Language*.

UNB *Ultra Narrow Band*.

UV Ultravioleta.

UWB *Ultra Wide Band*.

WLAN *Wireless Local Area Network*.

WPAN *Wide Personal Area Network*.

Capítulo 1

Introdução

1.1 Enquadramento

O sistema de um estore elétrico é composto pelo estore em si e um motor que controla a abertura e o fecho do estore. Este sistema veio ajudar ao conforto do utilizador possibilitando o controlo do estore com um simples pressionar de um interruptor para cima ou para baixo, dependendo da direção pretendida.

Mas, apesar disso, caso um utilizador tenha vários estores elétricos e queira colocar cada um com posições de abertura diferentes,terá de percorrer cada um dos interruptores, esperar que o estore atinja a posição desejada, e, posteriormente, realizar a mesma ação nos restantes. Isto é um processo demorado, pouco confortável e enfadonho para o utilizador.

1.2 Motivação

Com a introdução do IoT, houve uma revolução na área tecnológica. Segundo a Cisco, o IoT está presente nas mais variadas indústrias, desde a indústria da manufatura até à área do retalho e dos serviços financeiros [1]. Ainda, a Boston Consulting Group, estima que os se gastará cerca de **250 Biliões de Euros** em 2020 em IoT e, todas as indústrias deverão aumentar a sua aposta nesta área [2]. Por fim, o Gartner estima que em 2017, de todas as unidades IoT instaladas (cerca de **8380.6 milhões**), aproximadamente **63%** (**5244.3 milhões**) são associadas ao consumidor e, assim continuará até 2020 [3].

Na Figura 1.1 e na Tabela 1.1 encontram-se a estimativa de gastos e a estimativa de unidades de IoT, respectivamente.

Assim, com base nestes dados, seria uma oportunidade interessante desenvolver um sistema de controlo de um estore elétrico, baseado na tecnologia IoT, de forma a colmatar

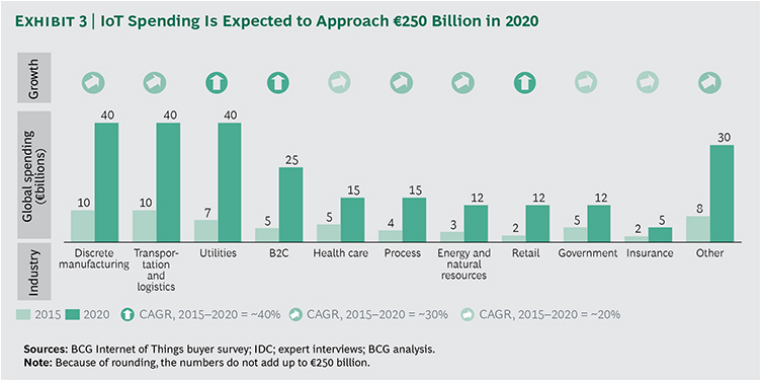

Figura 1.1: Estimativa de gastos em IoT nas diferentes áreas de negócio [2]

Tabela 1.1: Estimativa do número de dispositivos IoT ao longo dos próximos anos [3]

Category	2016	2017	2018	2020
Consumer	3963.0	5244.3	7036.3	12863.0
Business: Cross-Industry	1102.1	1501.0	2132.6	4381.4
Business: Vertical-Specific	1316.6	1635.4	2027.7	3171.0
Grand Total	**6381.8**	**8380.6**	**11196.6**	**20415.4**

IoT Units Installed Base by Category (Millions of Units)

a falta de conforto que os sistemas atuais fornecem ao consumidor.

Com a presença de alguns sensores e com um microprocessador, é possível realizar o controlo de um estore elétrico, com um simples pressionar de um botão numa aplicação *Smartphone*, no conforto da habitação ou fora desta, desde que exista ligação à Internet.

1.3 Estrutura da Dissertação

Esta dissertação, então, trata de propor um sistema que controle um estore elétrico através do uso de microprocessadores conectados a sensores de luminosidade e de temperatura para um controlo automático do estore e com um sensor de distância para solucionar a falta de conforto que o sistema atual fornece. Ainda, será desenvolvida uma interface (em *Android*) ao consumidor de forma a controlar o sistema.

Assim, este documento estrutura-se da seguinte forma:

1. Estado de Arte
2. Modelação do Sistema
3. Implementação
4. Conclusões e Trabalho Futuro

No Estado da Arte, são apresentadas as tecnologias analisadas para a implementação, bem como, um estudo de mercado, onde se analisa as principais funcionalidades dos produtos semelhantes ao implementado.

Na Modelação do Sistema, é modelado o sistema, de forma a facilitar a implementação. Nomeadamente, o sistema é decomposto em funções e, cada uma dessas funções, é retratada.

Na Implementação, é apresentado as técnicas, *software* e ligações elétricas para o desenvolvimento do prototipo.

Em Conclusões e Trabalho Futuro, são apresentadas algumas conclusões do trabalho desenvolvido, assim como aspetos a melhorar no seu desenvolvimento.

Capítulo 2

Estado de Arte

Nesta secção será feita uma descrição das tecnologias (comunicação sem fios, *blueto-oth*,...) utilizadas para realizar este projeto, bem como os sensores e *System on Chip* (SoC). De igual forma, serão descritos e comparados os sistemas semelhantes ao proposto. Os produtos avaliados são:

1. Levolux Microcontroller;
2. Nice WMS01ST;
3. Nice Nemo WSRT;
4. Somfy SUNIS IndoorWireFree™ RTS Sun Sensor;
5. Somfy THERMOSUNISWireFree™ RTS Light & Temperature Sensor;
6. Somfy Soliris;
7. Somfy Ondeis™WireFree RTS Rain & Sun Sensor;
8. Elero Lumero-868;
9. Elero Lumo-868;
10. Elero Aero-868;
11. Elero Sensero-868.

Desta forma, conseguimos ter a perceção dos sistemas existentes e das suas características.

Comecemos por analisar as tecnologias existentes que permitem a implementação da solução proposta.

2.1 Comunicação sem fios

Nesta secção são comparadas e apresentadas diferentes tecnologias de comunicação sem fios adequadas ao trabalho proposto. As tecnologias comparadas são:

- WiFi;
- *Bluetooth*;
- *Ultra Wide Band* (UWB);
- Zigbee.

A comparação das diferentes características permite escolher a que melhor se adequa ao trabalho proposto, considerando: distância, energia, facilidade de implementação.

Após a escolha da tecnologia, serão apresentados diferentes SoC's que permitem uma comunicação através dessa tecnologia.

2.1.1 Tecnologias sem Fios

Tipicamente, as tecnologias Zigbee e a *bluetooth* são utilizadas, normalmente, em comunicações com alcance máximo de 10 m, ou seja, *Wide Personal Area Network* (WPAN) [4], enquanto que as restantes, WiFi e UWB são utilizadas para comunicações de alcance máximo de 100 m *Wireless Local Area Network* (WLAN) [4].

Na Tabela 2.1, são apresentados alguns parâmetros de cada tecnologia, baseado em [5].

Tabela 2.1: Parâmetros de cada tecnologia

Tecnologia	WiFi	*Bluetooth*	UWB	Zigbee	LoRa	Sigfox
Frequência	2.4/5 GHz	2.4 GHz	3.1 - 10.6 GHz	868/915 MHz 2.4 GHz	433/868 MHz 915 MHz	868 MHz
Largura de Banda de Canal	22 MHz	1 MHz	500MHz - 7.5 GHz	0.3/0.6 MHz 2 MHz	-	200 KHz
Taxa de Sinal Máxima	54 Mb/s	1 Mb/s	110 Mb/s	250 Kb/s	50 Kb/s	100/600 bits
Corrente necessária para Transmissão	722.7 mA	102.6 mA	750 mA	74.1 mA	-	-
Corrente necessária para Recepção	709.5 mA	84.6 mA	750 mA	81 mA	-	-

Com o crescente uso de dispositivos IoT, novas tecnologias de comunicação foram criadas a pensar nestes dispositivos, que funcionam a bateria e utilizam comunicação sem fios a longa distância para realizar o seu objetivo. Estas tecnologias são descritas como *Low-Power Wide Area Network* (LPWAN) e serão analisadas especificamente a tecnologia LoRa e a tecnologia Sigfox [6].

LoRa significa *Long Range* e é uma tecnologia que suporta comunicações a grande distância e os seus principais objectivos é garantir que o dispositivo consuma o mínimo de energia possível permitindo assim que a bateria dure bastante tempo. A LoRa opera a

433, 868 e 915 MHz dependendo da região onde se encontra. A sua taxa de sinal máxima é de, aproximadamente, 50 Kb/s. De notar que, os dispositivos que utilizam esta tecnologia necessitam de um *gateway* específico desta tecnologia [7].

Relativamente à Sigfox , é uma tecnologia que ataca o problema das comunicações de uma forma bastante única: não é necessário estabelecer e manter conexões às redes. Através de uma modulação rádio *Ultra Narrow Band* (UNB), esta tecnologia opera a uma frequência de 868 MHz e usa cerca de 200 KHz da banda. Cada mensagem ocupa 100 Hz e a taxa de transmissão é de 100 ou 600 bits/s. Ao usar pouca frequência para a transmissão das mensagens, a bateria do dispositivo é poupada o que permite uma maior durabilidade. Um dispositivo ao querer enviar uma mensagem, o circuito de interface Sigfox acorda e a mensagem é enviada "uplink". Posteriormente, o dispositivo espera um curto período de tempo para verificar se existe dados a serem enviados "downlink". Sendo assim, como a verificação (*polling*) é feita do lado do dispositivo, esta tecnologia é bastante apropriada para aquisição de dados mas não se adequa em cenários "Command-and-Control"[7], [8].

Entre todas as tecnologias estudadas, conclui-se que a LoRa, e, especialmente a Sigfox, seriam uma interessante escolha para este projeto mas, a tecnologia escolhida será a WiFi visto que, na área doméstica, os pontos WiFi são predominantes além de que, os dispositivos móveis (*smartphones*) estão equipados com esta tecnologia e, comparativamente à tecnologia *Bluetooth*, a tecnologia WiFi tem uma distância de comunicação ser superior.

Apesar de, na Tabela 2.1, analisar-se que consome uma corrente superior comparativamente às restantes tecnologias, a tecnologia de Wifi suporta um grande número de dispositivos conectados ao mesmo tempo (até 250) , [4], [5]. Conclui-se, então, que esta tecnologia é a que melhor se adequa ao desenvolvimento deste projeto.

2.1.2 SoC's com Tecnologia WiFi

Tendo na Secção 2.1.1 chegado à conclusão que a tecnologia que melhor se adequa para a realização deste projeto é a tecnologia WiFi, é necessário realizar uma comparação de diferentes SoC's com esta tecnologia. Além disso, é necessário atuar sobre o motor do estore, obter valores dos sensores e processá-los para realizar o controlo do estore dependendo da temperatura e luminosidade. Sendo assim, estes dispositivos serão os pontos centrais do nosso sistema visto que são os SoC's que realizam a comunicação com todos os dispositivos. Como observado e concluído em 2.1.1, os SoC's a serem avaliados terão que comunicar através de WiFi. Sendo assim, as prioridades consideradas para a realização do projeto são as seguintes:

1. WiFi disponível no SoC;
2. Número de *General Purpose Input/Output* (GPIO)'s disponíveis;
3. Consumo energético do SoC;
4. Preço do SoC.

Assim, serão avaliados os SoC's presentes na Tabela 2.2:

Tabela 2.2: Comparação entre diferentes SoC's

Nome do SoC	CC3200MOD [9]	SPWF01SA [10]	GS1011 [11]	BCM43903 [12]	ESP8266 [13]
Alimentação	2.3V-3.6V	3.3V	3.0V-3.6V	3.0V-4.8V	3.0V-3.6V
Número de GPIO's Disponíveis	25	16	32	17	17
Corrente necessária para Transmissão*	229 mA	243 mA	-	-	140 mA
Corrente necessária para Receção*	59 mA	105 mA	-	-	56 mA
Preço	7.99€	19.20€	20.00$	7.30$	5.50€

*Valores correspondentes a comunicações WiFi 54 Mbps *Orthogonal frequency-division multiplexing* (OFDM)

Observando a Tabela 2.2, conclui-se que o ESP8266 será o processador mais adequado a utilizar visto que, comparativamente aos restantes, é o que consome um valor menor de corrente para realizar a comunicação, é o mais barato e também fornece uma quantidade de GPIO's suficiente para a realização do projeto.

2.2 Armazenamento de dados - *Cloud*

Neste tipo de projetos IoT, é interessante dispor sempre os dados disponíveis, quer seja, localmente ou remotamente, através de uma aplicação. Para isso, será utilizada uma base de dados na *Cloud* para guardar os dados obtidos dos sensores do sistema. Para isso, é necessário fazer uma comparação entre alguns dos fornecedores de *Cloud* existentes. Os fornecedores de *Cloud* possíveis serão:

- Microsoft Azure;
- Amazon S3 (AWS);
- Google Cloud;
- IBM

Todos os fornecedores apresentados têm um período de *trial*.

A Microsoft Azure oferece 170€ de crédito para novos clientes num prazo de 30 dias [14].

O período de *trial* da AWS é de 1 ano e oferece 5 GB de armazenamento, 20000 "Get Requests"e 2000 "Put Requests"[15].

A Google Cloud também oferece 12 meses gratuitos com, cerca de, 266€ para utilizar nos seus serviços[16].

Por fim, a IBM oferece um período de 1 mês gratuito. Neste período, poder-se-à enviar 200 MB de dados a partir do nosso processador, mensalmente, para a *Cloud*[17].

Devido a existir um protocolo entre a faculdade e a IBM que possibilita a sua utilização durante 6 meses (e são extensíveis durante mais 6 meses, indefinidamente) e também este serviço ter sido o único que já foi utilizado anteriormente, utilizar-se-à, então, este serviço de *Cloud*.

2.3 Medição de Distância

Nesta secção serão apresentados as tecnologias e sensores para medir distância.

2.3.1 Tecnologias

Para realizar a medição de distância são consideradas duas tecnologias diferentes: através da quantidade de luz que um determinado sensor recebe ou através de ultrassons.

Através da quantidade de luz são empregues dois elementos (detetor e recetor). Os emissores normalmente utilizados são *Light Emitting Diode* (LED) de infravermelhos visto que, ao contrário das lâmpadas de tungsténio, a luz é emitida numa banda mais curta e na zona dos infravermelhos no espetro de frequências e, dessa forma, o sistema torna-se mais robusto a interferências provenientes da luz solar [18]. Para os recetores, existem 4 formas distintas de detetar luz:

1. **Dispositivos fotocondutivos (fotoresistores):** Estes dispositivos chamam-se *Light Dependent Resistor* (LDR) visto que convertem as mudanças na luz incidente em resistência;

2. **Dispositivos fotovoltaicos (fotocélulas):** Estes dispositivos são chamados de células solares quando um conjunto destes são utilizados para gerar energia a partir do sol. O seu modo de funcionamento é bastante simples: quando exposto à luz, gera uma tensão de saída cuja magnitude é uma função da magnitude da luz incidente.

3. **Fotodíodos:** Dispositivos cuja corrente de saída é uma função da quantidade de luz incidente.

4. **Fototransístor:** Constituído por um transístor bipolar com a parte exterior transparente, permitindo assim que a luz atinja a junção base-colector do transístor. O seu *output* é uma corrente elétrica (semelhante ao fotodíodo), mas alterado por um ganho interno. Isto permite que o fototransístor seja mais sensível que o fotodíodo, em particular na zona dos infravermelhos, mas com um tempo de resposta mais lento.

Os sensores de ultrassons, compostos por um cristal piezoelétrico, ao ser-lhes aplicado uma tensão que varia, comporta-se como um transmissor e uma onda de ultrassons é produzida. O sensor também poderá funcionar como um recetor de ultrassons. A velocidade de transmissão é influenciada por alguns fatores, tais como, a temperatura e a humidade do ar. A velocidade de transmissão dos ultrassons é dada pela Equação 2.1 [18]:

$$v = 331,6 + 0,6\ m/s \tag{2.1}$$

Assim, observa-se que, a temperatura terá algum impacto na medição da distância e também o preço maior destes sensores comparados à tecnologia de medição de luz. Sendo assim, será utilizado um LED Infravermelhos (IV) com um recetor de IV.

2.3.2 Led de Infravermelhos

O LED de infravermelhos é utilizado no sistema de distância. Em particular, o LED funciona como emissor de infravermelhos e o LED escolhido é o TSAL6200 [19]. Uma das vantagens deste LED é a verticalidade com que o feixe de IV é emitido.

2.3.3 Sensor de Infravermelhos

O sensor de infravermelhos é, tal como o LED IV, utilizado no sistema de obtenção de distância trabalhando em conjunto. O seu objetivo é receber o feixe de IV proveniente do LED de forma a "calcular"a distância a que o estore se encontra do solo/parapeito. Foram avaliados 3 sensores diferentes:

- TSSP58038;
- Sharp GP2Y0A21YK;
- TSSP58P38;

O TSSP58038 não tem a capacidade para devolver um valor que simbolize distância. Só permite saber se foi ou não detetado um feixe de IV [20].

O Sharp GP2Y0A21YK permite já obter o valor da distância mas, devido ao seu preço (13.95$) foi excluído [21].

Por fim, o TSSP58P38 foi o sensor escolhido, devido ao seu baixo custo [22].

2.4 Medição de Luminosidade

O sensor de luminosidade é usado para obter o valor da luminosidade de forma a poder controlar o estore automaticamente dependendo desse valor e também registar esse valor na *Cloud* para consulta e estatísticas.

As tecnologias para medição de luminosidade encontram-se descritas na secção 2.3.1.

Um dos sensores analisado foi o BPW 21 que está adaptado para a sensibilidade do olho humano, ou seja, o seu funcionamento é semelhante ao do olho humano [23]. Mas, após análise e realização de alguns testes com o sensor, detetou-se que seria necessário um circuito de amplificação, o que iria causar um maior gasto de corrente.

Em contrapartida, escolheu-se utilizar um LDR em que, quando exposto a 10 Lux, a sua resistência é, aproximadamente 10 KΩ.

2.5 Medição de Temperatura

O sensor de temperatura, à semelhança do apresentado na Secção 2.4, servirá para o controlo do estore em função da temperatura da habitação. O valor da temperatura ao ser superior ou inferior ao nível definido pelo utilizador implica que o estore será, respetivamente, fechado ou aberto. De igual forma, os valores de temperatura também serão enviados e registados na *cloud*.

2.5.1 Tecnologias

Para realizar a medição de temperatura, existem vários princípios para tal [18]:

- Efeito Termoelétrico (termopares);
- Alteração Resistiva;
- Dispositivos semicondutores;
- Radiação;
- Termografia;
- Expansão Térmica;
- Frequência Ressonante;
- Sensibilidade dos dispositivos de fibra ótica;
- Mudança de cor;
- Mudança do estado do material.

Apenas serão avaliados os termómetros que se baseiam no princípio de alteração resistiva por facilidade de implementação e pelo preço que estes sensores apresentam.

Este princípio baseia-se no princípio físico de que a resistência de um material varia com a temperatura e estes dispositivos são, normalmente, conhecidos como termístores ou termómetros de resistência. O método mais comum de medição de resistência é através do uso de uma ponte *Direct Current* (DC). Apesar de se querer uma sensibilidade alta, através de uma tensão de excitação alta, tem que se escolher uma tensão de excitação ótima visto que teremos uma corrente proporcional à tensão. Isto implica que, a valores de corrente elevados, teremos um efeito de aquecimento no dispositivo, o que cria um erro na medição da temperatura porque altera o valor da resistência [18].

2.5.2 Termómetros de Resistência

Os termómetros de resistência baseiam-se no facto de os materiais variarem a sua resistência, segundo a Equação 2.2 [18]:

$$R = R_0(1 + a_1 T + a_2 T^2 + a_3 T^3 + ... + a_n T^n) \tag{2.2}$$

Sendo a Equação 2.2 não linear, é necessário linearizá-la para tornar conveniente a medição. Assim, desprezando todos os termos de ordem igual ou superior a 2 obtemos a Equação 2.3, segundo [18]:

$$R \approx R_0(1 + a_1 T) \tag{2.3}$$

2.5.3 Termístores

Estes dispositivos têm um coeficiente negativo de resistência, ou seja, a resistência diminui com o aumento da temperatura. Este comportamento é demonstrado pela Equação 2.4 [18]:

$$R = R_0 . e^{\beta(\frac{1}{T} - \frac{1}{T_0})} \tag{2.4}$$

Observa-se que, esta equação não é linear mas, em contrapartida, estes dispositivos têm pequenas dimensões e um custo reduzido, o que é perfeito para projetos IoT.

2.5.4 Escolha do Sensor

Inicialmente não será adicionado nenhum sensor visto que este sensor não é considerado indispensável para o correto funcionamento e realização deste projeto.

Caso seja utilizado, o sensor de temperatura escolhido é o LM35 [24], visto que, a sua saída depende linearmente da temperatura a que se encontra exposto. A sua função de saída é dada pela Equação 2.5 [24]:

$$V_O = 10mV * T \tag{2.5}$$

Em que a tensão de saída V_O é expressa em miliVolt (mV) e a temperatura T é expressa em graus Centígrados (ºC). Isto significa que, por exemplo, aquando se obtém uma tensão de saída de 200 mV, o dispositivo encontra-se exposto a uma temperatura de 20 ºC. Ainda, à temperatura de 25 ºC, o efeito de auto-aquecimento é quase inexistente (0.08 ºC). Por fim, e devido a não ser necessário uma precisão grande na temperatura, este sensor apresenta uma precisão de 0.5 ºC.

2.6 Sistemas Comerciais

Nesta secção são apresentados sistemas que possibilitam um controlo automático de estores elétricos semelhantes ao protótipo desenvolvido. São apresentadas as suas principais funcionalidades e o modo de controlo de cada um dos produtos.

2.6.1 Microcontrolador Levolux

Este microcontrolador tem o objetivo de controlar várias zonas com diversos tipos de estores presentes num edifício através de vários sensores presentes.

Consegue controlar até 4 zonas com diferentes tipos de estore. Igualmente, este microcontrolador suporta a receção de sinais provenientes de 11 sensores lineares e também 2 anemómetros. Tem incorporado um relógio digital que permite ao utilizador selecionar o tipo de controlo, automático ou manual, através das horas introduzidas. O sistema contém um *display* que disponibiliza as configurações e o estado do sistema.

De seguida, são apresentados estes componentes:

- **Sensores de Luminosidade –** A cada um dos 4 grupos independentes, é possível utilizar sensores deste tipo, ou vários grupos partilharem o mesmo sensor. A intensidade da luz ao ultrapassar um valor pré-definido, causa o encerramento dos estores; se esse valor diminuir para um valor inferior ao pré-definido, os estores abrem.
- **Sensores de Temperatura** - Tem um comportamento semelhante ao sensor anterior.
- **Anemómetros/Sensores de Vento –** Sensores utilizados para estores exteriores, recolhem a velocidade do vento. Como anteriormente, existe um valor pré-definido para que os estores fechem de forma a protegê-los. Além dos sensores apresentados, este sistema contém uma tecnologia de *Sun Tracking* que calcula a posição do Sol e ajusta o ângulo dos estores.

Na Figura 2.1 é apresentado um sistema controlado pelo Microcontrolador Levolux.

2.6.2 Nice WMS01ST

Este produto é um transmissor rádio (433.92 MHz) em que, na sua composição, tem 2 sensores diferentes (luminosidade e temperatura) para determinar a posição do estore. Existe também outra versão (WMS01S) que utiliza, unicamente, um sensor de luminosidade. Estes transmissores são compatíveis, exclusivamente, com os motores disponibilizados pela empresa.

Dispõe de dois tipos de controlo: manual e automático. O controlo manual é realizado através do teclado. O modo automático é controlado através dos sensores instalados no transmissor. Este produto, ao ser composto por um sensor de luminosidade dianteiro e traseiro, permite assim dois tipos de configuração. Na Figura 2.2 é apresentado a composição deste produto.

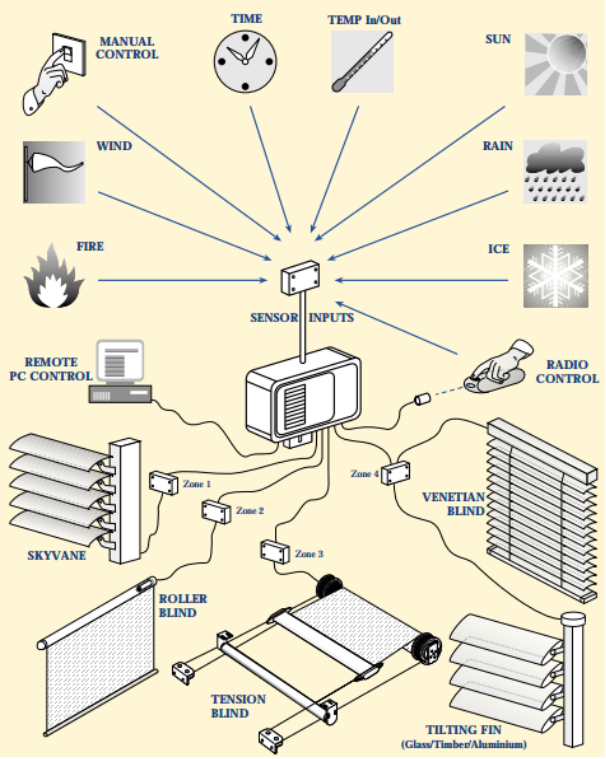

Figura 2.1: Controlo de diversos tipo de estores utilizando o *Microcontrolador Levolux* , adaptado de [25]

Figura 2.2: Composição do WMS01ST da NICE [26]

O transmissor pode ser colocado diretamente na janela através de uma ventosa (Figura 2.3) ou colocado em cima de um objeto no interior da divisão (Figura 2.4).

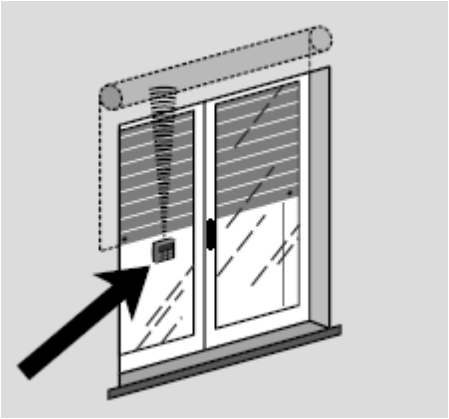

Figura 2.3: Transmissor colocado na janela [26]

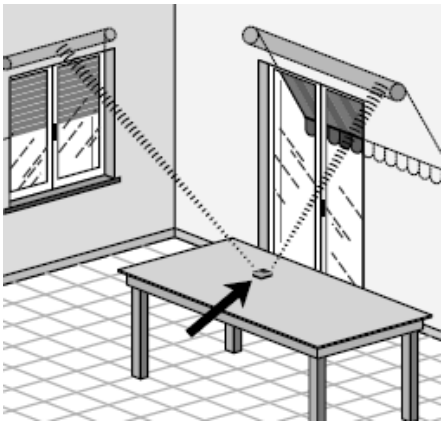

Figura 2.4: Transmissor colocado dentro de uma divisão [26]

No caso da Figura 2.3, são disponibilizados 3 modos de funcionamento.

1. O sensor de luminosidade traseiro é o único ativo. No caso de o estore estar aberto e o sensor detetar que o valor da luminosidade superou o nível definido durante um tempo determinado (ex: 5 minutos), o estore fechar-se-á até ao ponto onde o transmissor está posicionado e recolhe alguns centímetros para o sensor de luminosidade ficar exposto. Se o valor do sensor diminuir para um valor inferior ao definido, o estore irá abrir-se. Na figura 2.5 encontra-se o fluxograma deste modo.

Figura 2.5: Fluxograma do modo 1 do WMS01ST

2. Semelhante ao modo anterior com a diferença de o estore não abrir automaticamente. Terá de ser realizado manualmente através do teclado.

3. O sensor de luminosidade traseiro e o sensor de temperatura contribuem, ambos, para a tomada de decisão do transmissor, em que o valor da temperatura domina essa decisão. Se a temperatura for superior ao valor definido no sistema, este funcionará como descrito no 1º modo. Caso a temperatura baixe para valores inferiores

ao definido em algum ponto, o modo automático é cancelado e o estore é colocado na posição onde se encontrava com o valor da temperatura acima do definido. O controlo automático é reposto quando a temperatura atingir esse nível. Na figura 2.6 está representado o fluxograma deste modo.

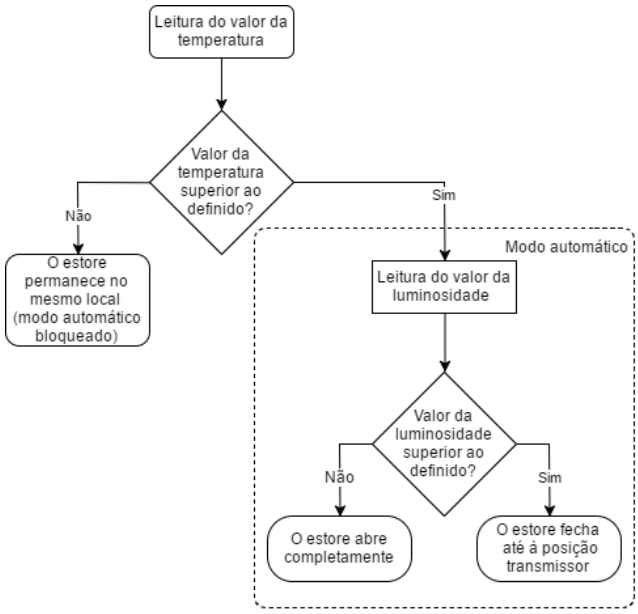

Figura 2.6: Diagrama de sequência do modo 3 do WMS01ST

Na figura 2.4, como observamos, o transmissor é colocado em cima de um objeto (ex: mesa). Neste caso, temos disponíveis mais 2 modos de funcionamento:

4. Este modo de configuração ativa exclusivamente o sensor de luminosidade traseiro. O transmissor tem definido um intervalo em que a luminosidade é ótima. Se a luminosidade ultrapassar o limite superior, o estore fechar-se-á até que a luminosidade atinja um valor dentro do intervalo definido. Se a luminosidade diminuir para valores inferiores ao limite inferior do intervalo, o estore abrir-se-á até que o valor se encontre dentro do intervalo. O intervalo poderá ser definido manualmente no transmissor. Na figura 2.7 está representado o fluxograma deste modo.

5. Os sensores que se encontrarão ativos serão o de luminosidade dianteiro e o de temperatura e, neste modo, o utilizador tem à sua disposição duas opções:

   a) Opção de Inverno

   b) Opção de Verão

   Cada uma destas opções deverá ser selecionada dependendo da estação do ano. O modo de funcionamento para ambas é semelhante visto que a diferença entre cada uma é se o estore abre ou fecha para o mesmo tipo de alteração detetada. A temperatura é novamente dominante sobre a luminosidade. Na opção de Verão,

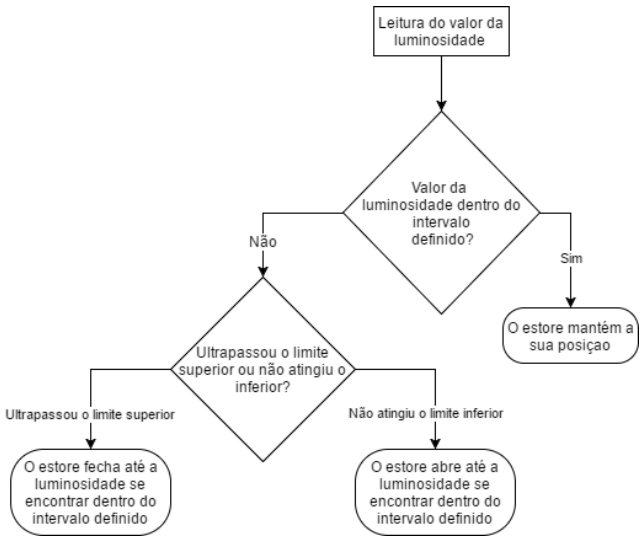

Figura 2.7: Fluxograma do modo 4 do WMS01ST

se o valor de temperatura ultrapassar o nível pré-definido o estore fecha-se por completo. Se este valor for inferior ao pré-definido, o transmissor tomará a decisão dependente do nível de intensidade e funciona da forma que foi descrita no modo 1. Caso a opção de Inverno for a escolhida, a diferença é que se o valor da temperatura for inferior ao pré-definido o estore ficará completamente aberto. Se o valor for superior, o transmissor terá o mesmo comportamento descrito anteriormente.

Para finalizar, todos os intervalos e níveis pré-definidos falados anteriormente, poderão ser escolhidos *à priori* pelo utilizador.

2.6.3 Nice NEMO WSRT

Este produto é um sensor climatérico que possui um transmissor rádio para comunicar com os motores tubulares presentes nos estores. Possibilita a tomada de decisão através da velocidade do vento, presença ou não de chuva e valor de luminosidade (existe uma versão deste equipamento que disponibiliza apenas a velocidade do vento e a luminosidade – NEMO WSCT). Na figura 2.9 está representado a composição do NEMO WSRT.

Para a velocidade do vento e para a luminosidade existe um limite em que o sensor transmitirá uma decisão para o motor. Se a velocidade do vento ultrapassar esse valor durante 3 segundos, todas as operações são recusadas no motor e o estore irá abrir-se. Quando a velocidade do vento diminuir, ao fim de 4 minutos o sensor permite que sejam enviados controlos manuais para o motor e o modo automático é restaurado.

Quanto à luminosidade, se durante 2 minutos o valor for superior ao definido, o estore irá fechar-se. Quando o valor diminuir, e assim se mantiver durante 15 minutos, o estore irá abrir-se.

O sensor de chuva deteta apenas a presença ou não de chuva e, assim, quando deteta chuva, o estore abrir-se-á.

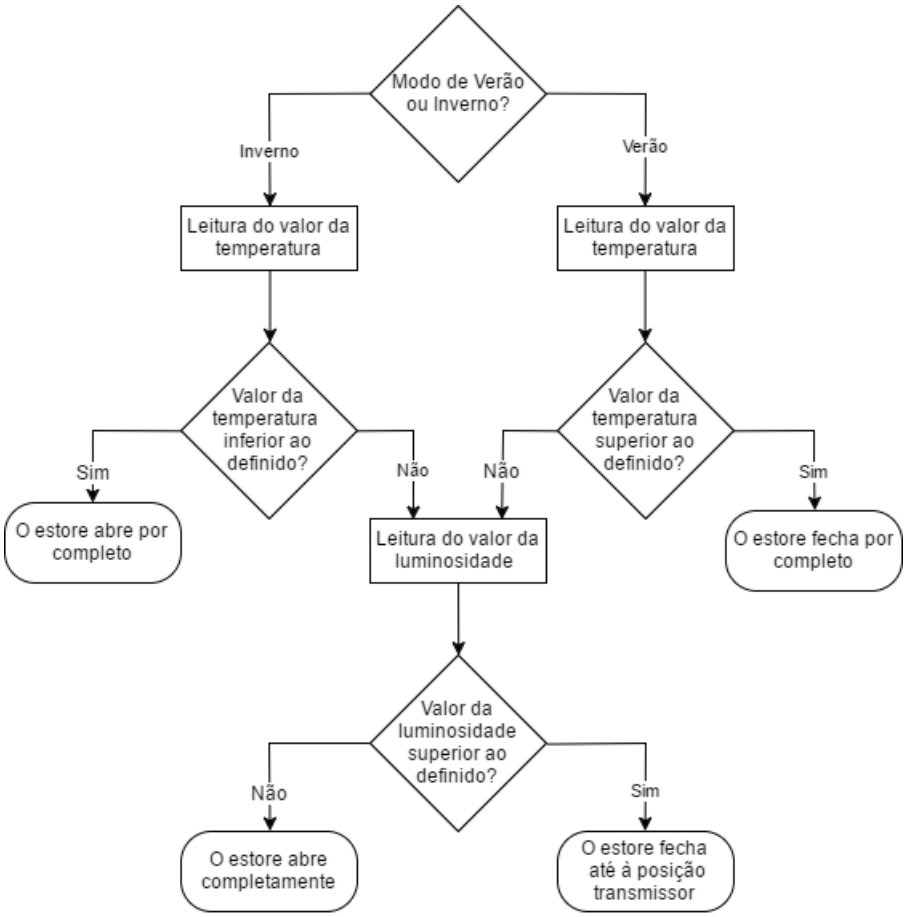

Figura 2.8: Diagrama de sequência do modo 5 do WMS01ST

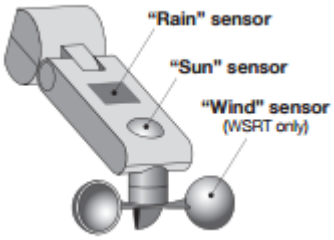

Figura 2.9: Composição do NEMO WSRT [27]

Na Figura 2.10 encontra-se o fluxograma deste sensor.

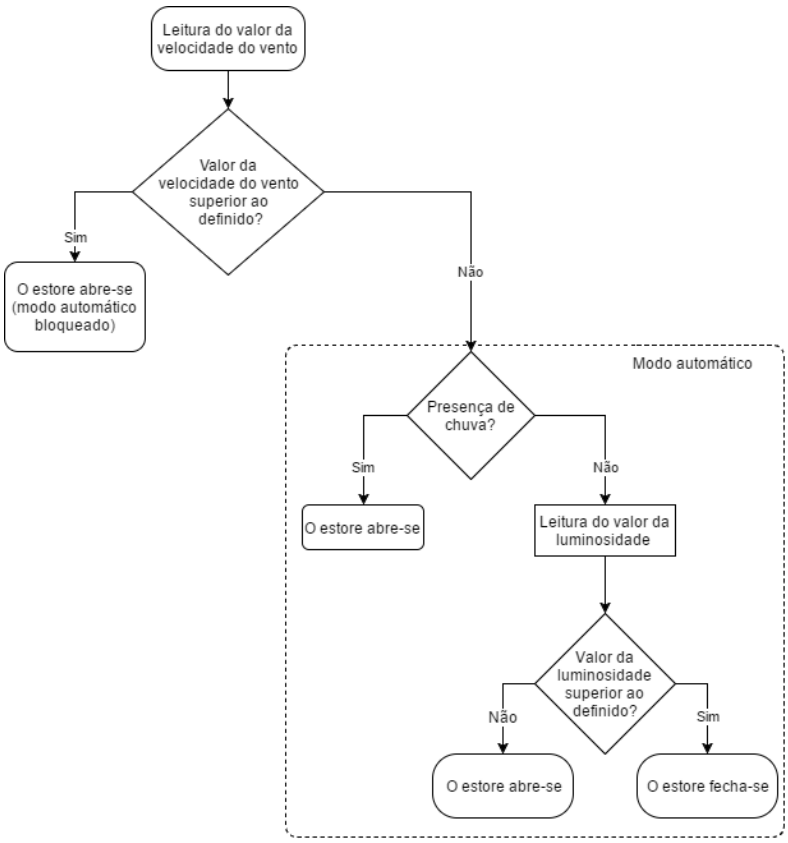

Figura 2.10: Fluxograma do NEMO WSRT

2.6.4 Somfy SUNIS Indoor WireFree™ RTS Sun Sensor

É um sensor/transmissor rádio sem fios para tecnologia RTS (Radio Technology Somfy® - 433.42 MHz) sem fios que comunica com os motores e os recetores desta empresa de forma a controlar estores/persianas dependendo da intensidade de luz detetada. Na Figura 2.11 está representado o transmissor. A funcionalidade deste produto é que permite assim proteger os objetos no interior de uma divisão dos raios Ultravioleta (UV). O seu funcionamento é simples e, apesar de ser semelhante a um dos modos apresentados em 2.6.2, a posição final, quando o estore fecha, não depende da posição do sensor.

É disponibilizado, então, um único modo de operação em que é definido um valor de *threshold*. Se o valor da intensidade da luz for superior ao valor de *threshold* durante 5 minutos, o transmissor envia uma ordem (baixar o estore) para o motor. Caso o valor da intensidade da luz seja inferior ao valor de *threshold* durante 30 minutos, o transmissor envia, então, uma ordem para o estore subir. De notar que o transmissor só envia uma ordem quando deteta uma variação na intensidade da luz. Este dispositivo tem um funcionamento semelhante ao modo 1 do WMS01ST (2.6.2) e, assim, o fluxograma associado encontra-se na Figura 2.5.

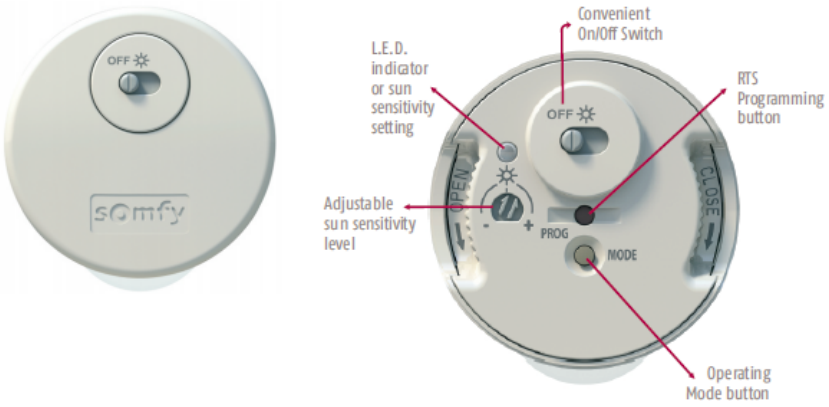

Figura 2.11: SUNIS Indoor WireFree™ RTS Sun Sensor [28]

2.6.5 Somfy THERMOSUNIS WireFree™ RTS Light & Temperature Sensor

Tal como o produto descrito anteriormente, trata-se de um sensor/transmissor de rádio sem fios para tecnologia RTS para controlar motores com esta tecnologia, motores esses que controlam estores. Em comparação com o produto em 2.6.4, este tem disponível um sensor adicional de temperatura e tem disponível 3 modos de funcionamento automático.

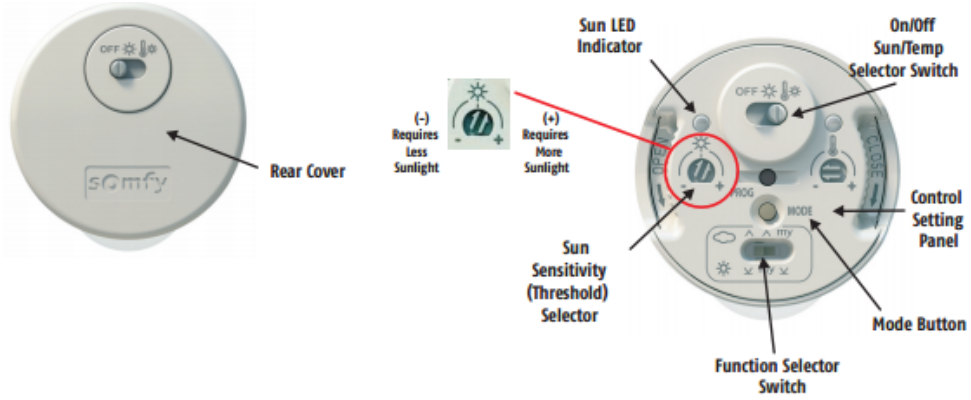

Figura 2.12: THERMOSUNIS WireFree™ RTS Light & Temperature Sensor [29]

Poderá ser utilizado unicamente a opção de intensidade de luz ou a opção conjunta de intensidade de luz e temperatura. Em semelhança com o produto descrito em 2.6.2, a decisão retirada através do valor de temperatura será prioritária à da intensidade da luz. Na Tabela 2.3 são apresentados os diferentes modos:

Cada um destes modos só funcionará se o valor de temperatura for superior ao valor de *threshold*, caso contrário, o estore permanecerá no mesmo sítio de forma a manter ou aumentar a temperatura.

Tabela 2.3: Modos disponíveis no THERMOSUNIS WIREFREE™ RTS LIGHT & TEMPERATURE SENSOR (baseado na tabela em [29])

	Modo 1	Modo 2	Modo 3
Intensidade da luz superior ao *threshold* durante 5 minutos	O estore baixa até á posição do sensor	O estore movimenta-se para a posição "*my*"*	O estore baixa até ao limite inferior
Intensidade da luz inferior ao *threshold* durante 30 minutos	O estore sobe até ao limite superior	O estore sobe até ao limite superior	O estore movimenta-se para a posição "*my*"*

* A posição "*my*" é uma posição definida no motor, ficando definida uma posição favorita. Esta opção é realizada através do próprio motor e não faz parte do funcionamento do THERMOSUNIS WireFree™ RTS Light & Temperature Sensor

2.6.6 Somfy Soliris

O SOLIRIS é um sensor de luminosidade e de vento e utiliza a mesma tecnologia apresentada anteriormente (RTS) para comunicar com os motores.

Figura 2.13: Sensor SOLIRIS RTS [30]

Para a correta tomada de decisão, este sensor usa valores de *threshold* para a luminosidade e vento. Estes valores podem ser ajustados através de dois potenciómetros. Se a velocidade do vento for superior à definida durante 2 segundos, é enviado um comando para o motor de forma a abrir o estore completamente e é bloqueado qualquer comando, manual ou proveniente da tomada de decisão a partir da luminosidade. Assim, não é causado qualquer estrago no estore devido a fortes rajadas de vento. A partir do momento em que a velocidade do vento diminui para um valor inferior ao de *threshold*, as ordens enviadas através de decisões envolvendo a luminosidade continuam bloqueadas durante 12 minutos, mas os controlos manuais estão disponíveis ao fim de 30 segundos.

Semelhante a vários produtos apresentados, se a intensidade da luz for superior ao valor de *threshold* durante 2 minutos, o estore fechar-se-á. Após o valor de luminosidade baixar para valores inferiores ao de *threshold*, é disparado um *timer* entre 15 e 30 minutos (dependo do que o utilizador tiver definido) em que ao fim desse tempo, é enviado um comando de abertura para o motor do estore. Como referido em [30], isto permite que o estore não esteja sempre a movimentar-se em dias nublados.

2.6.7 Somfy Ondeis™ WireFree RTS Rain & Sun Sensor

Sensor de chuva e de luminosidade com tecnologia RTS. Este sensor está equipado também com um painel solar, como observamos na figura 2.14, que permite recarregar a sua bateria. Utiliza um valor de *threshold* para o nível de chuva e de luminosidade.

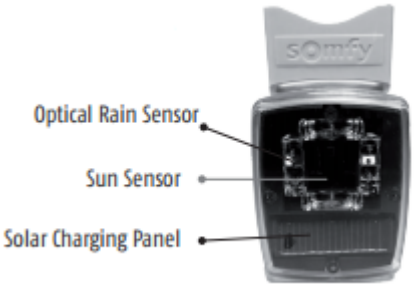

Figura 2.14: Ondeis™ WireFree RTS Rain & Sun Sensor [31]

Disponibiliza 6 tipos de controlo, mas apenas serão explicados 3 visto que os outros restantes só são aplicáveis a toldos exteriores, que não é o foco desta tese. Então, os modos disponibilizados são os seguintes:

1. **Modo de Chuva ("*Shutter/Screen Rain*")** - Se o valor medido de chuva for superior ao de *threshold*, será enviado um comando para o motor associado para baixar o estore. O sensor não enviará mais nenhum comando até que o valor da chuva seja inferior ao de *threshold* durante 5 minutos e volte a ultrapassá-lo.

2. **Modo de Chuva e de Luminosidade ("*Shutter/Screen Rain & Sun*")** – Modo semelhante ao anterior, mas com a adição da utilização do sensor de luminosidade. Este sensor, se detetar valores superiores ao de *threshold* durante 2 minutos, envia um comando de descida. Se este valor, durante 15 minutos for inferior ao de limite, é enviado um comando de subida para o motor do estore. Como já foi referido, este modo é semelhante ao anterior. Assim, se o sensor de *threshold* para chuva for ultrapassado, é enviado um comando para descer o estore e todas as decisões baseadas na luminosidade são bloqueadas até o sensor de chuva não detetar que choveu nos últimos 5 minutos.

3. **Modo de chuva com subida automática ("*Shutter/Screen Rain & Auto Up*")** – Igual ao 1º modo, mas se ao fim de 5 minutos não detetar chuva, envia um comando de subida para o motor do estore.

2.6.8 Elero Lumero-868

Trata-se de um sensor de luminosidade com um transmissor rádio sem fios compatível com todos os equipamentos da empresa com tecnologia *ProLine* e comunica através de sinais rádio de 868 MHz.

Figura 2.15: Lumero-868 [32]

Através de um valor de *threshold* definido, este aparelho controla o estore de forma a manter um nível de luminosidade constante. Se o valor da luminosidade for superior ao valor de *threshold* durante, aproximadamente, 5-7 minutos, este sensor irá enviar um comando para o estore de forma a descê-lo. Contrariamente, se o valor da luminosidade for inferior ao definido durante, aproximadamente, 15-17 minutos, o estore irá subir.

2.6.9 Elero Lumo-868

Tal como muitos dos produtos apresentados, este é novamente um sensor/transmissor sem fios e comunica através de ondas rádio com 868 MHz. Este produto apresenta uma característica nova fase a todos os produtos apresentados – um sensor de quebra de vidro. Além disso, apresenta, também, um sensor de luminosidade. Através disso, o produto é capaz de controlar os estores através de dois valores de *threshold*. Estes valores podem ser alterados através de dois potenciómetros correspondentes a cada *threshold*. O limite superior pode ser definido para pouca luz (dias nublados) até incidência direta no sensor. O limite inferior pode ser definido para agir assim que comece a anoitecer ou então que só aja quando anoiteça por completo. Ambos os *thresholds* podem ser desligados.

Na Figura 2.16 está representado o Lumo-868.

Se o valor de *threshold* superior for ultrapassado durante 3 minutos, é enviado um comando para descer o estore. Quando este tapar o sensor, o estore é recolhido ligeiramente para deixar o sensor exposto. Se o valor de luminosidade não ultrapassar este *threshold* durante 15 minutos, o estore é aberto.

Relativamente ao valor de *threshold*, inferior, se este não for atingido durante 15 minutos, o estore é fechado por completo e só poderá ser aberto novamente através de um comando manual ou de um temporizador presente noutros produtos.

O sensor de quebra de vidro funciona através da vibração causada pelo vidro a quebrar. Caso isto aconteça, é enviado um comando para o motor do estore de forma a fechar o estore por completo e todos os comandos automáticos são bloqueados.

Figura 2.16: Lumo-868 [33]

2.6.10 Elero Aero-868

Trata-se de um transmissor/sensor de vento e de luminosidade sem fios. Comunica usando ondas rádio da mesma frequência que ambos os produtos anteriores. Este aparelho tem 3 variantes:

1. **Aero-868**
2. **Aero-868 Plus:** Tem uma maior autonomia devido a uma bateria extra.
3. **Aero-868 AC:** Alimentação elétrica realizada através da rede elétrica.

O modo de funcionamento é igual para as 3 variantes e é semelhante ao comportamento apresentado em 1.3.3. Se o sensor de vento detetar uma velocidade superior à de *threshold*, o estore é aberto e todos os comandos são bloqueados durante 15 minutos. Se a luminosidade for superior ao valor de *threshold* durante 5-7 minutos, o estore é fechado. Caso contrário, se durante 15-17 minutos não ultrapassar esse valor, o estore permanece aberto completamente ou é enviado um comando para o abrir.

De notar que não é recomendado utilizar este sensor para controlo durante a noite devi-do à reserva de energia ser limitada.

Na Figura 2.17 está apresentado o Aero-868.

2.6.11 Elero Sensero-868

Sensor semelhante ao apresentado anteriormente 2.6.10, mas apresenta algumas funções adicionais e também a alimentação energética é feita, exclusivamente, pela rede elétrica. Sendo assim, poderá ser utilizado para proteção noturna, ao contrário do produto anterior. Este sensor é colocado no exterior do edifício.

Existem 2 variantes deste produto:

1. **Sensero-868 AC:** Contém um sensor de luminosidade e um sensor de vento.

Figura 2.17: Aero-868 [34]

2. **Sensero-868 Plus:** Contém ambos os sensores anteriores e um sensor de chuva.

Para definir os *thresholds*, existem 3 ou 4 potenciómetros (dependendo da versão do sensor) para regulação dos valores:

1. Luminosidade diurna
2. Luminosidade noturna
3. Vento
4. Chuva

Se o sensor de luminosidade detetar que o valor de *threshold* de luminosidade diurna foi ultrapassado durante 5-7 minutos, o estore é fechado, caso contrário, continua aberto. Caso o valor não atinja o *threshold* de luminosidade noturna, significa que anoiteceu e o estore irá baixar igualmente para permitir a privacidade.

Quanto ao sensor de vento, assim que o valor de *threshold* for ultrapassado, o estore é aberto e permanece assim durante 15-17 minutos e todos os comandos são bloqueados, manuais inclusive.

Por fim, o sensor de chuva tem um comportamento igual ao de vento à exceção de bloquear exclusivamente os comandos automáticos.

Na Figura 2.18 é apresentado o sensor Sensero-868 AC.

2.7 Apresentação do Trabalho Proposto

Após a análise dos sistemas comerciais (Secção 2.6) observa-se que existem algumas funcionalidades que são transversais a estes dispositivos (controlo por luminosidade, temperatura, entre outros). Apesar disso, conclui-se que não existe nenhum dispositivo que permita abrir/fechar os estores para uma determinada posição, personalizável de acordo com a vontade do utilizador. De igual forma, é proposto o desenvolvimento de um sistema que controle o estore automaticamente através de um sensor de luminosidade e

Figura 2.18: Sensero-868 AC [35]

de temperatura. Por fim, será disponibilizada uma *Cloud* em que serão gravados todos os dados para efeitos estatísticos e para consulta de dados fora da rede do microprocessador.

Na tabela seguinte (Tabela 2.4), é apresentado o custo de cada material e o preço total para a implementação deste projeto.

Tabela 2.4: Tabela do Preço Total de implementação do Projeto

Material	**Preço**	**Local de Compra (19/09/2017)**
2x MOD-WIFI-ESP8266-DEV	11.00 €	Olimex
TSSP58P38	0.98 €	Mouser
TSAL6200	0.42 €	Mouser
LDR	0.25 €	Robert Mauser
LM35	1.71 €	Robert Mauser
2x Relé 893-8991CFC12	6.60 €	Mouser
Restante Material (Resistências, potenciómetros, ...)	~ 5.00 €	Robert Mauser
Total	**25.96€**	

Capítulo 3

Modelação do Sistema

Neste capítulo são apresentados os modelos do sistema proposto para controlo autónomo e automático de estores elétricos. Inicialmente, é apresentado um esboço do sistema e um esquema de blocos de alto nível. Posteriormente, os casos de uso do sistema e a descrição de cada um. Seguidamente, são descritos os modelos estruturais e comportamentais para resposta aos diferentes casos de uso. Por fim, descreve-se cada uma das funções do sistema.

Ao

3.1 Diagrama de Blocos do Sistema

O objetivo desta tese é a realização de um sistema IoT que controle autonomamente um sistema de estores através de sensores de distância, luminosidade e temperatura e também que reaja a pedidos do utilizador. Todos os valores obtidos dos sensores serão guardados na *cloud* para armazenamento e estatística. Por sua vez, estes dados poderão ser acedidos através de uma aplicação móvel e é através desta aplicação que o utilizador pedirá ao sistema para colocar o estore na posição de fecho pedida (em %). De igual forma, poderá mudar o controlo do sistema de automático para manual. Na figura 3.1 está representado um diagrama de blocos do sistema proposto. Também está representado um diagrama de alto nível do funcionamento do sistema proposto na figura 3.2.

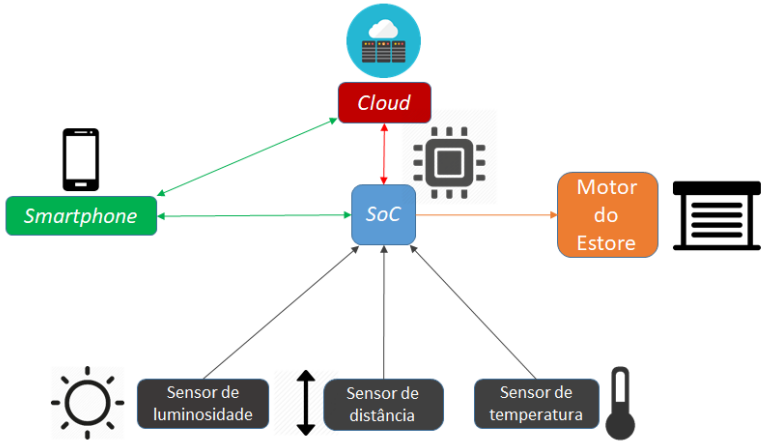

Figura 3.1: Blocos do Sistema Proposto

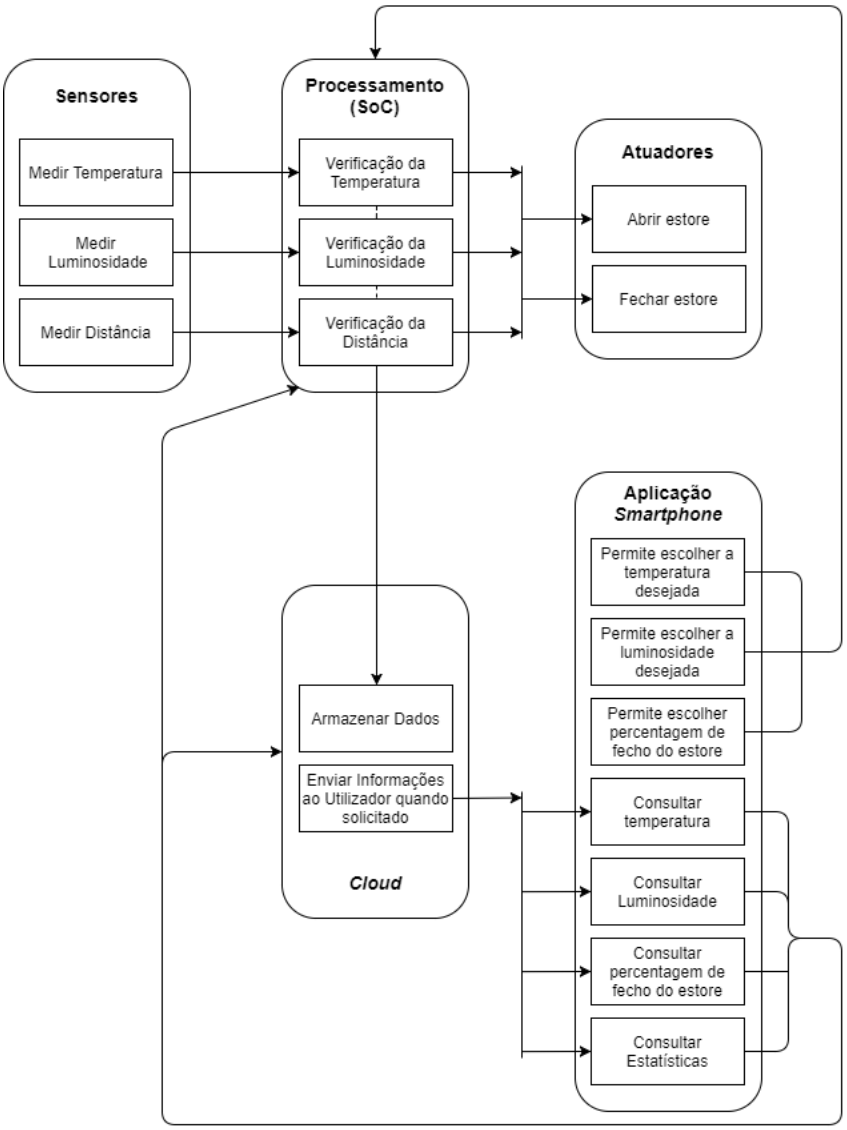

Figura 3.2: Esquema de alto nível repartido em blocos do sistema proposto

3.2 Casos de Uso

Para melhor compreensão e entendimento do funcionamento e comportamento do sistema, é necessário a modelação deste. Nesta secção são apresentados os casos de uso do sistema que visam ajudar a modelar o sistema.

Através das figuras 3.1 e 3.2, podemos concluir que existem 5 "atores"que compõem este sistema:

1. Utilizador;
2. Processador (SoC);
3. *Cloud*;
4. Sensores;
5. Motor do estore.

3.2.1 Utilizador

O utilizador, através da utilização da aplicação móvel presente no seu *smartphone*, poderá realizar as seguintes funções:

- Escolher a percentagem de fecho do estore;
- Requisitar informações do sistema, diretamente ao processador ou à *cloud*;
- Escolher os níveis de luminosidade e temperatura utilizados em modo de controlo automático;
- Ativar/Desativar o controlo automático.

Para a obtenção das informações de temperatura, luminosidade, distância e níveis desejados, o utilizador tem de pressionar o respetivo botão na aplicação. Dependendo da sua ligação WiFi, se estiver na mesma rede que o SoC, este pedido é feito diretamente ao SoC, caso contrário, as informações são obtidas através da *cloud*.

Também através da aplicação, é possível a escolha da percentagem de fecho do estore. Este valor é introduzido e, posteriormente enviado para o SoC.

Os valores dos níveis escolhidos podem ser alterados e, de forma semelhante à escolha da percentagem de fecho, são enviados para o processador.

Por fim, o utilizador poderá escolher entre ativar ou desativar o controlo automático do estore.

3.2.2 Processador (SoC)

O processador tem como objetivo obter a informação dos sensores, controlar o estore, receber comandos do utilizador e enviar informações para a *cloud* e para o utilizador. Sendo assim, as funções do processador são:

- Regular a temperatura;
- Regular a luminosidade;
- Obter a percentagem de fecho pedida pelo utilizador;
- Abrir ou fechar o estore;
- Enviar as informações dos sensores e do sistema para a *cloud* ou para a aplicação móvel;

Em modo autónomo/automático, os valores dos sensores de temperatura e luminosidade, também permitem ao processador regular os respetivos valores, através da abertura ou fecho do estore. Este controlo funciona através de um nível de prioridade em que, a luminosidade é prioritária face à temperatura, ou seja, para a o estore ser atuado dependendo da temperatura, a luminosidade tem que estar no nível desejado pelo utilizador.

O processador ao receber um pedido do utilizador para fechar o estore a um determinado nível, utiliza o sensor de distância para saber a posição aproximada do estore.

Aquando da mudança da posição do estore ou ao fim de determinado tempo (5 minutos será o tempo máximo que a *Cloud* não receberá informações), o processador enviará os dados para a *cloud*, ficando assim armazenados. Por solicitação da aplicação móvel/utilizador, o processador envia dados diretamente para este, caso o *smartphone* esteja ligado à mesma rede do processador no momento em que o pedido de informação do sistema é requisitado.

3.2.3 *Cloud*

A *cloud* tem como funções armazenar todos os dados ao nível dos sensores, gravar alterações ao estado do sistema (alteração do nível desejado de distância, temperatura ou luminosidade e alteração ao modo de controlo) e enviar informações para a aplicação móvel quando solicitado. Sendo assim, o utilizador estando afastado do sistema central consegue aceder a todas as informações.

3.2.4 Sensores

Os sensores têm como objetivo recolher alguns dados sobre o ambiente à volta (temperatura e luminosidade) e obter a distância a que o estore se encontra do solo/parapeito onde está colocado permitindo, assim, obter a posição relativa do fecho do estore. Os sensores são necessários para fazer as seguintes medições:

- Temperatura;
- Luminosidade;
- Distância.

3.2.5 Motor do estore

O motor é considerado um atuador. Movimenta o estore no sentido indicado pelo processador, por forma a obter o resultado desejado pelo utilizador.

3.2.6 Sistema Geral

Depois de analisados todos os "atores"do sistema, é necessário ter uma visão geral sobre o sistema e sobre os seus componentes. Assim, na figura 3.3 está representado os casos de uso do sistema.

3.3 Diagramas de Sequência do Sistema

Tendo sido analisado os casos de uso do sistema na secção anterior, é necessário, de igual forma, entender as relações e as comunicações entre os vários componentes do sistema. Para isso, será utilizada uma abordagem *Unified Modeling Language* (UML), mais propriamente, um diagrama comportamental de sequência UML.

Neste diagrama, a entidade mais à esquerda envia uma mensagem que despoleta uma sequência de comportamentos. Como observado anteriormente, o Utilizador será quem irá alterar alguns dos estados do sistema. Assim, para a correta modelação das relações entre os componentes do sistema, serão tomados em conta as seguintes entidades: Utilizador; Aplicação; *Cloud*; SoC; Sensores.

Na figura 3.4 está representado o diagrama de sequência do sistema.

3.4 Funções do Sistema

Na secção 3.3, foram modeladas as comunicações entre as diferentes entidades. Adicionalmente, é necessário ter a perceção de como o sistema funciona em tempo real. Para isso, o sistema será repartido num sistema de blocos/funções. De referir que, o sistema parte de um ponto inicial ao ligar-se o sistema, de seguida é executado um ciclo infinito e só termina quando o utilizador desliga o sistema. O diagrama de blocos das funções está representado na figura 3.5.

3.4.1 Função Aplicação

O bloco *Função Aplicação* tem o objetivo de fornecer uma interface ao utilizador através de uma aplicação móvel no seu *smartphone*. Através dessa aplicação, encontram-se

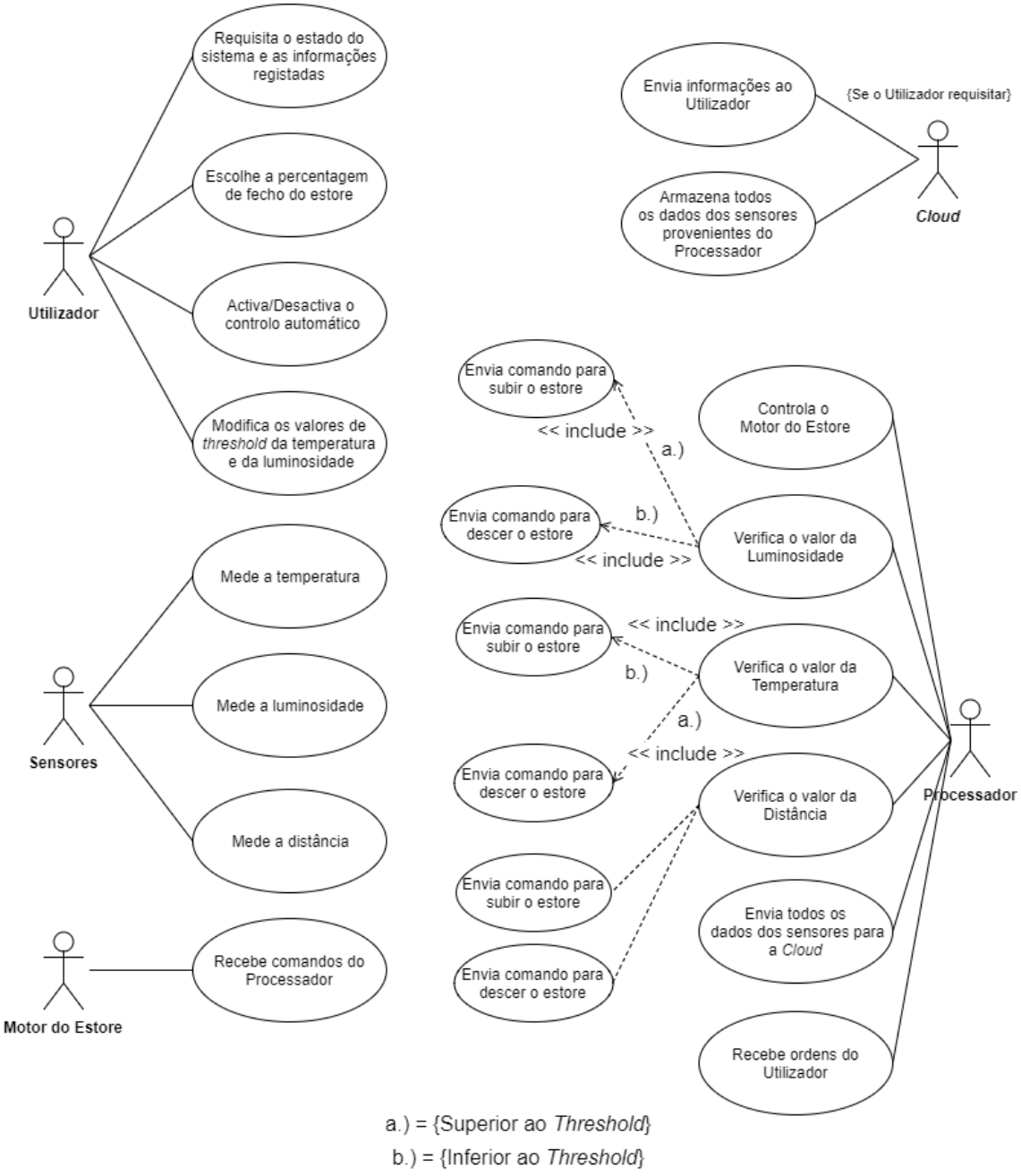

Figura 3.3: Diagrama de casos de uso do sistema

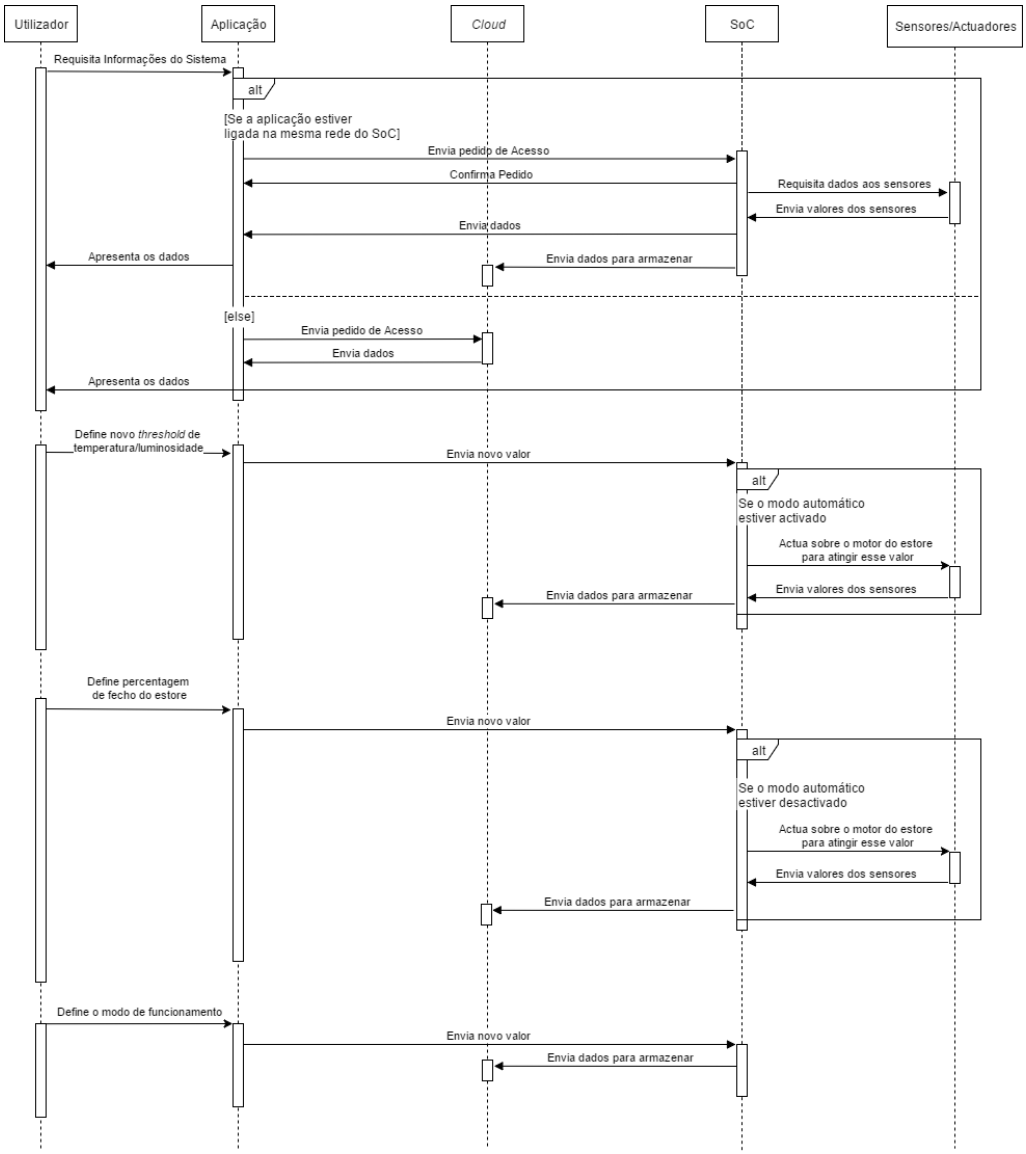

Figura 3.4: Diagrama de sequência do sistema

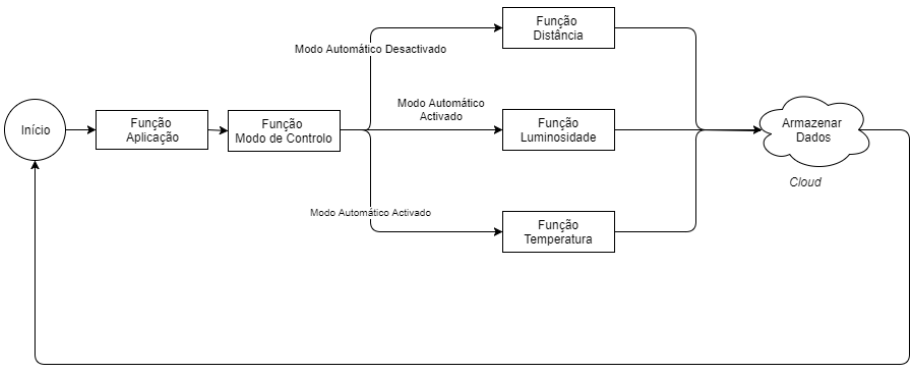

Figura 3.5: Diagrama de blocos contendo as funções do sistema

disponíveis vários botões para controlar o sistema e, também, um botão para obter informações do sistema. A aplicação será composta por várias atividades que conterão os seguintes componentes:

- Caixa de texto para definir o *threshold* da temperatura - Permite ao utilizador escolher o valor da temperatura e enviá-lo para o SoC através de um botão;
- Caixa de texto para definir o *threshold* da luminosidade - Permite ao utilizador escolher o valor da luminosidade e enviá-lo para o SoC através de um botão;
- Caixa de texto para definir a percentagem de fecho do estore - Permite ao utilizador escolher a percentagem de fecho do estore e enviá-lo para o SoC;
- Botão de escolha do modo de controlo - Permite ao utilizador escolher o modo de controlo do estore (manual ou automático);
- Botão para obtenção de dados do Sistema - Permite ao utilizador visualizar os valores dos sensores obtidos e as alterações realizadas ao sistema.

Na figura 3.6 está representado o fluxograma desta função.

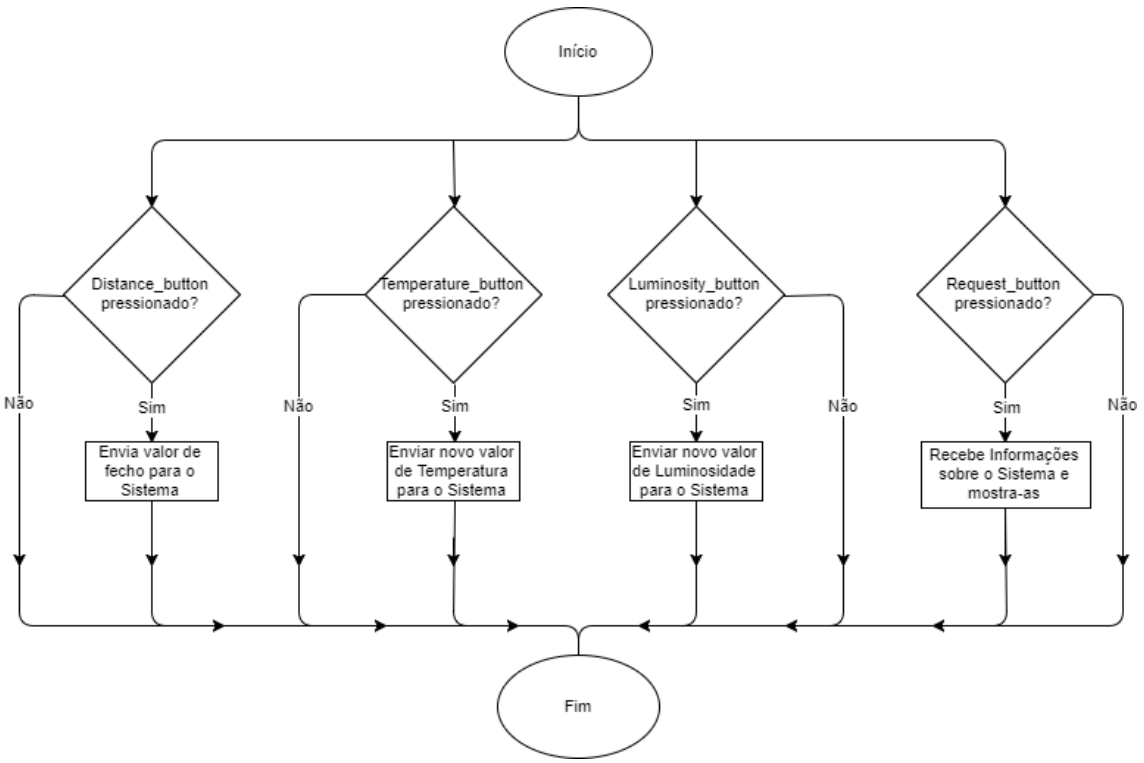

Figura 3.6: Fluxograma do funcionamento da função aplicação

3.4.2 Função Modo de Controlo

O bloco *Função Modo de Controlo* é o responsável pela escolha do controlo do motor do estore, automático ou não automático. Através da aplicação, o utilizador seleciona o modo que preferir. Posteriormente, o SoC avalia o modo selecionado e toma a sua decisão baseado nessa escolha. Na figura 3.7 está representado o fluxograma desta função.

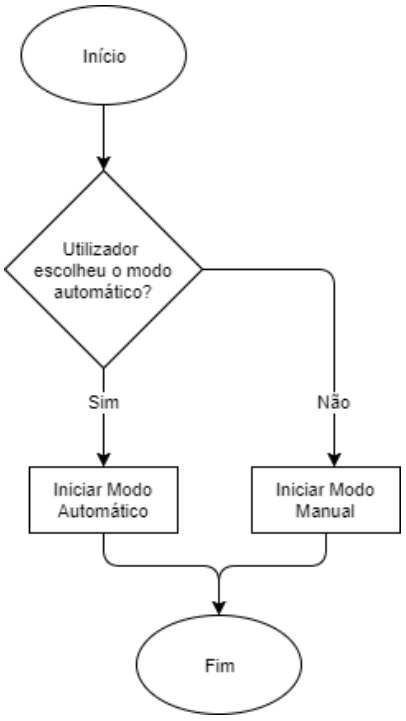

Figura 3.7: Fluxograma da função para a seleção do modo de controlo

3.4.3 Função Distância

O bloco *Função Distância* tem a função de controlar o estore dependendo da percentagem de fecho do estore que o utilizador introduzir na aplicação. Posteriormente, o SoC recebe esse valor de percentagem, converte-o para um valor inteiro para poder ser comparado com o valor lido do sensor e avalia a posição atual do estore para calcular se o estore terá que subir ou descer. De notar que, este modo só é realizado se o o modo de controlo automático (Secção 3.4.2) estiver desativado. Por fim, o SoC armazenará a informação mais recente na *Cloud*. Na figura 3.8 está representado o fluxograma desta função.

3.4.4 Função Luminosidade

O bloco *Função Luminosidade* é responsável pelo controlo da luminosidade na habitação. Para isso, o SoC necessita de controlar o estore. Através de um nível de *threshold* definido na aplicação, o SoC tentará atingir esse valor. Caso esta função realize alguma alteração ao nível do estore e depois de atingir a luminosidade pretendida, durante 5

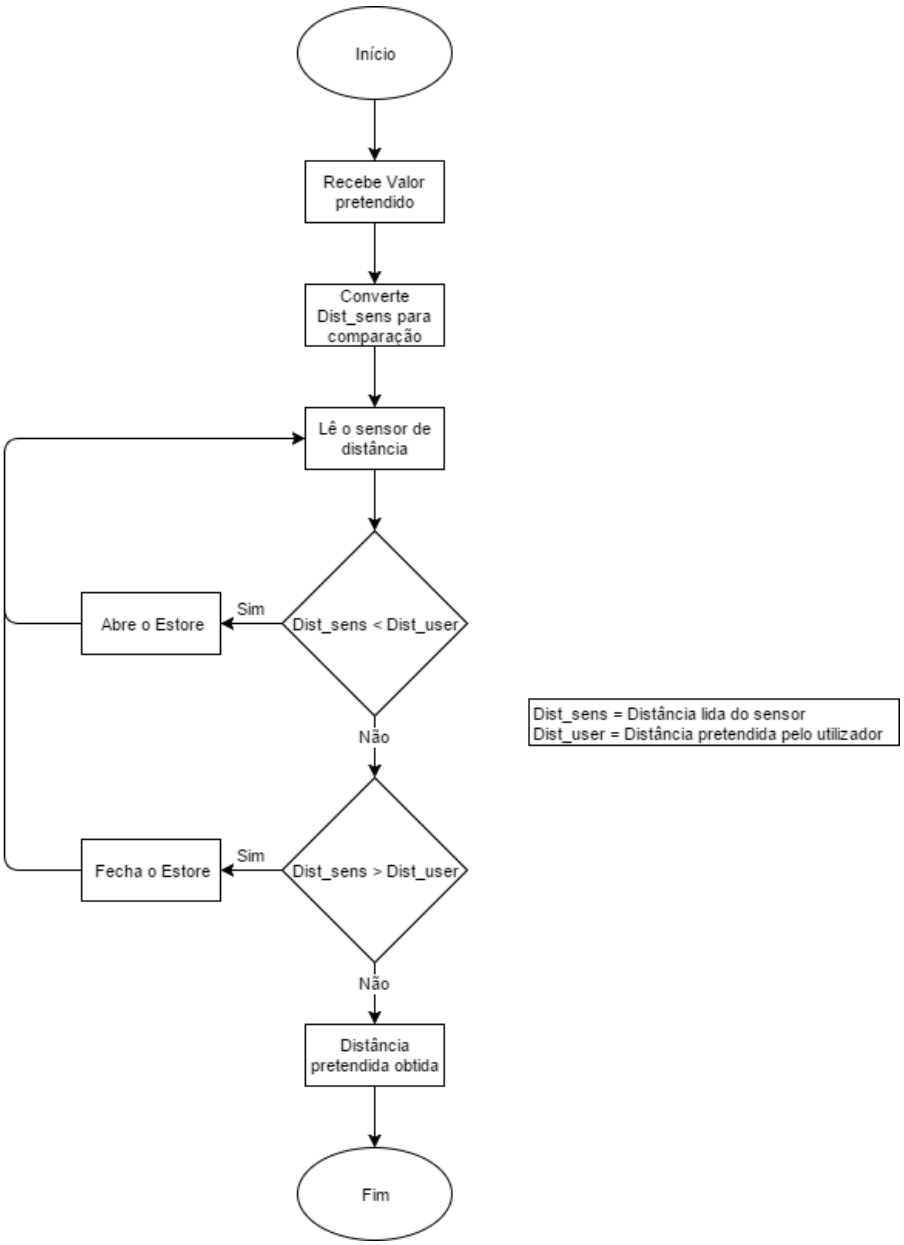

Figura 3.8: Fluxograma da função para obtenção da distância pretendida pelo utilizador

minutos, as funções que compõe o controlo automático não estarão ativadas de forma a não desgastar o uso do estore, ou seja, o modo automático entrará num período de *stand-by*. Caso também, a luminosidade não atinja o valor desejado ao final de um tempo determinado (30 segundos) o que implica que o estore se encontra num dos seus limites (completamente aberto ou fechado), o SoC não enviará mais comandos para subir ou descer o estore visto que se encontra completamente aberto ou completamente fechado. Na figura 3.9 está representado o fluxograma desta função.

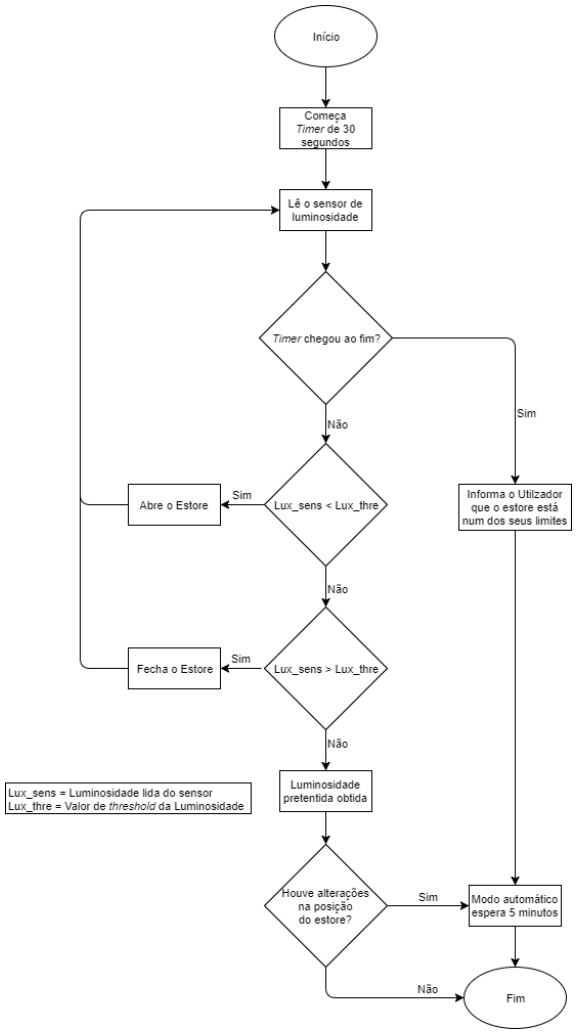

Figura 3.9: Fluxograma da função para obtenção da luminosidade pretendida pelo utilizador

3.4.5 Função Temperatura

O bloco *Função Temperatura* é o responsável pelo controlo da temperatura dentro da habitação. O seu funcionamento é semelhante ao controlo da luminosidade (Secção 3.4.4). Através do nível de *threshold* escolhido na aplicação e enviado para o SoC, este tentará manter a temperatura selecionada através do controlo do estore. Na figura 3.10

está representado o fluxograma desta função.

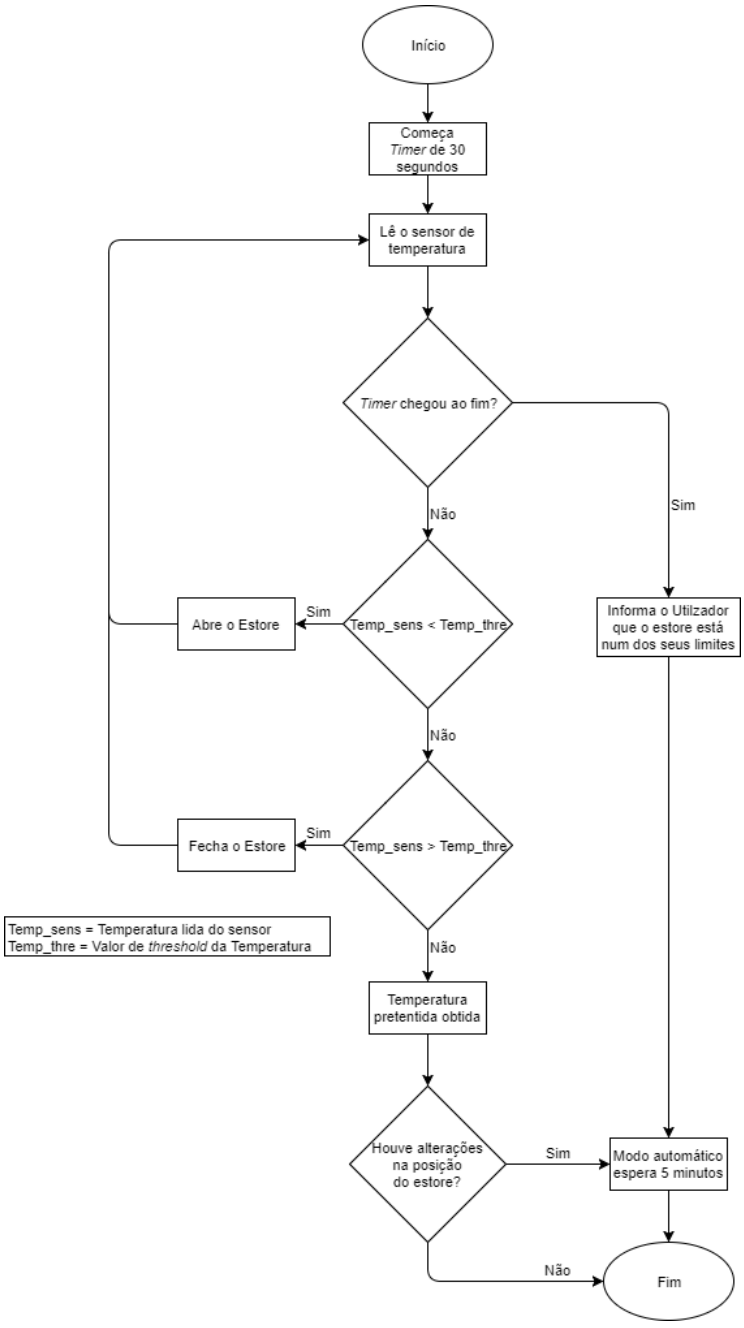

Figura 3.10: Fluxograma da função para obtenção da temperatura pretendida pelo utilizador

3.4.6 Armazenamento de dados na *Cloud*

Ao observarmos a figura 3.5, observamos que existe um bloco "Armazenar Dados". Os dados serão guardados na *Cloud* em três situações distintas.

1. Se o utilizador alterar algum dado do sistema, ou seja, se alterar o modo de controlo,

os *thresholds* ou o valor da distância pretendida através da aplicação;

2. Se, ao final de algum tempo, ou seja, quando um *Timer* de 10 minutos chegar ao fim, o que significa que a última vez que a base de dados foi atualizada foi há mais de 10 minutos;

3. Se o SoC alterar a posição do estore através de uma das suas funções.

Capítulo 4

Implementação

Neste capítulo, será apresentado o modo de implementação do projeto. Ou seja, os circuitos e todos os seus componentes e o código desenvolvido.

Começar-se-á pela alimentação do ESP8266, como o programar e, posteriormente as ligações criadas para comunicar com os sensores.

Também será explicada a função de cada ESP8266, visto que teremos dois microprocessadores deste tipo, um para dentro da habitação, outro para ser colocado no estore, na parte exterior.

Para uma maior facilidade entre a identificação dos *pins* entre os circuitos e os esquemáticos apresentados ao longo deste capítulo, na Figura **??** encontra-se o esquemático do ESP8266 presente nas *breadboards*.

Figura 4.1: Esquemático do ESP8266 presente nas restantes figuras

4.1 Circuito de Alimentação e Programação dos ESP8266

Começou-se por montar o circuito de funcionamento dos dois ESP8266. Soldou-se os módulos aos *pins* que os acompanhavam e analisou-se um possível circuito de alimentação e de programação.

Após a análise do *Datasheet* do ESP8266 [13] e como verificado na tabela 2.2, constata-se que, no máximo, este microprocessador, consumirá, aproximadamente 300 mA (com WiFi conectado e processamento de dados) e a tensão de alimentação adequada é de 3.3 V. Assim sendo, e de forma a não sobrecarregar o ESP8266 com uma tensão superior ao permitido, foi utilizado um regulador de tensão (LD1117V33, [36]) de 3.3 V e 800 mA de corrente máxima à saída do conector USB type-B fêmea presente na *breadboard*. Posteriormente, foi feita a ligação do V_{CC} (3.3V) ao *pin* 1 (V_{CC}) do ESP8266 e da referência ao *pin* 2 do ESP8266 (GND).

De forma a comunicar com o PC para ser programado, foi utilizado um Arduino UNO com as portas TX e RX ligadas às portas correspondentes no ESP8266. À saída do *pin* 4 do ESP8266 (RX), foi criado um divisor de tensão com 3 resistência de igual valor (1000 Ω) para, novamente, não utilizar uma tensão superior à que o ESP8266 suporta.

No Arduino UNO, o *pin* RST foi ligado ao *pin* GND para não programar o processador presente no Arduino.

Segundo o *datasheet* do ESP8266 [13], para ser possível programar o próprio, é necessário colocar o microprocessador em modo FLASH. Isto implica que, os *pins* 19(MTDO), 20 (GPIO0) e 21 (GPIO2) tenham que ser conectados de uma maneira específica. As configurações destes *pins* para selecionar o modo desejável no ESP8266, incluindo o modo FLASH. Os modos de configuração encontram-se apresentados na Tabela 4.1.

Tabela 4.1: Modos de Configuração do ESP8266

	MTDO	**GPIO0**	**GPIO2**
SDIO	1	X	X
UART	0	0	1
FLASH	0	0	1

Por fim, ainda é necessário ligar o *pin* 13 (RESET) a GND e retirá-lo. De igual forma, é necessário ligar o *pin* (`CH_EN`) a V_{CC}. Também foi adicionado um condensador de 100 µF para estabilizar o sinal da fonte.

Na Figura 4.2 e Figura 4.3 está representado o circuito criado (criado no *software* Fritzing) e o esquemático correspondente.

O ambiente de desenvolvimento escolhido foi o Arduino IDE de modo a facilitar a programação. Este processo foi realizado para 2 microprocessadores visto que teremos um processador central para ligação à *Cloud* e ligação aos sensores de temperatura e luminosidade e que atua no motor do estore e um processador que se encontrará no exterior da habitação, colocado no estore e que terá ligação ao processador central e onde se conectará o sensor de distância.

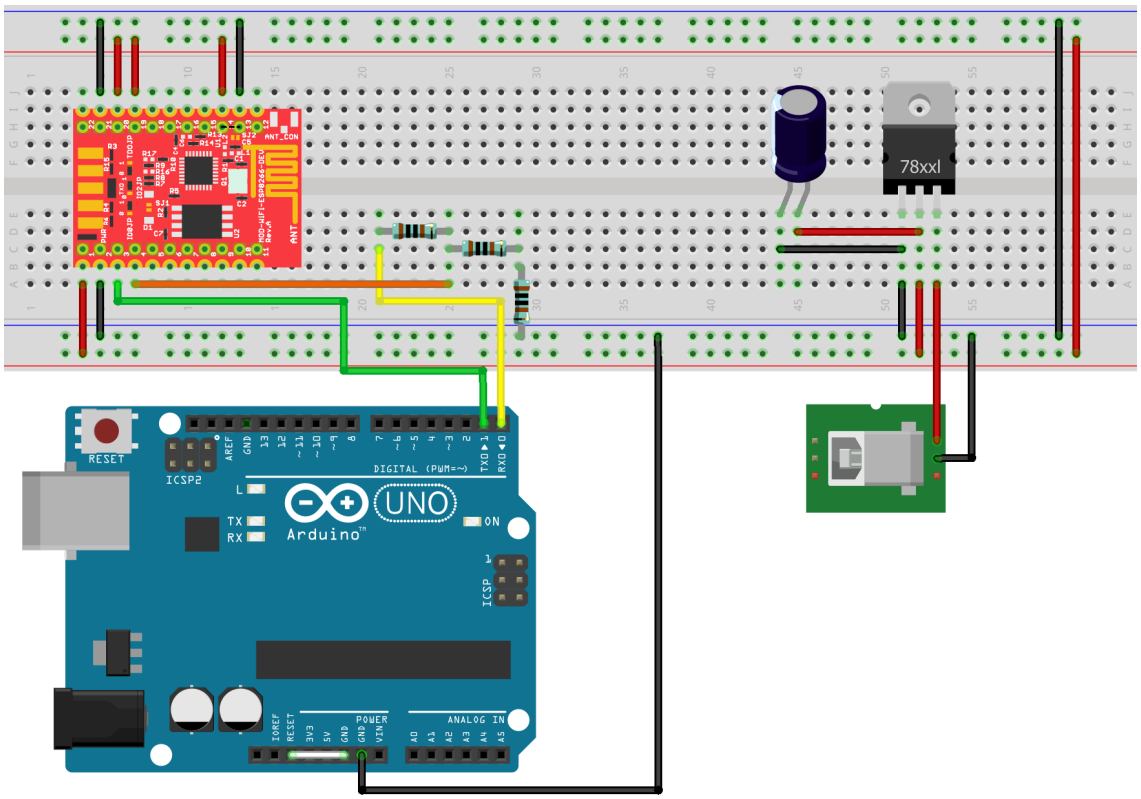

Figura 4.2: Circuito de Alimentação e Programação do ESP8266

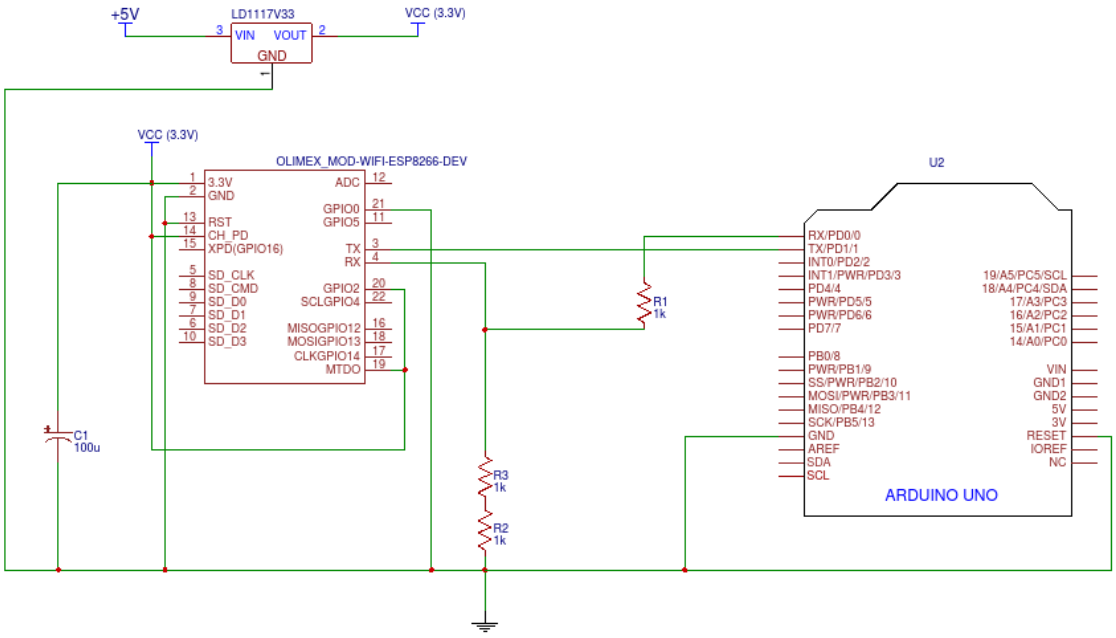

Figura 4.3: Esquemático de Alimentação e Programação do ESP8266

4.2 Processador Central

Este processador encontrar-se-á dentro da habitação conectado a uma rede WiFi pertencente ao utilizador e será responsável pelas seguintes funções:

- Controlo do estore através do input do utilizador na aplicação e dos valores obtidos nos sensores;
- Ligação ao sensor de temperatura;
- Ligação ao sensor de luminosidade;
- Receber e enviar informações para o processador presente no exterior;
- Ligação à *Cloud* para armazenamento de Dados.

De seguida, cada uma das funções serão descritas como foram implementadas. O código associado a este processador encontra-se disponível no Anexo I.

4.2.1 Controlo do estore através do *input* do utilizador na aplicação e dos valores obtidos nos sensores

Esta função tem como objetivo implementar a modelação realizada nas secções 3.1, 3.3, 3.4 e 3.6. O utilizador através da aplicação seleciona o modo desejado. Caso seja selecionado o modo automático, o processador avalia os valores dos sensores de temperatura e de luminosidade e atua nos motores dependendo desses valores.

Para atuar sobre os motores, é necessário utilizar relés. Os relés utilizados são de 12 V, mas serão ativados através de uma bateria de 9 V. Em testes em laboratório, analisou-se que, a 7 V, estes relés consomem cerca de 100 mA e ficam ativos. E visto que o ESP8266, através dos seus *pins* digitais, só fornece, no máximo 20 mA e 3.3 V [13], é necessário utilizar um Transístor de Junção Bipolar (TJB), como um interruptor, que permite amplificar a corrente presente na sua base, e colocar essa corrente amplificada no coletor. Assim, através da colocação de uma resistência na base, será possível dimensionar a corrente do coletor e ativar o relé.

O dimensionamento é baseado através do *Datasheet* do transístor. Analisando o *Datasheet*, observa-se que, a tensão entre a base e o emissor necessária para o transístor conduzir corrente da base para o coletor de $V_{\mathrm{be_{ON}}} = 0.6V$, sendo o ganho intrínseco do transístor de $\beta = 100$, a tensão de saturação entre o coletor e o emissor de $V_{\mathrm{ce_{SAT}}} = 0.6V$ quando a corrente no coletor $I_{\mathrm{c}} = 150mA$ e a corrente na base $I_{\mathrm{b}} = 15mA$.

Assim, e sabendo que a corrente para ativar o relé é de $I_{\mathrm{c}} = 100mA$ e que

$$I_{\mathrm{c}} = \beta . I_{\mathrm{b}} \tag{4.1}$$

é necessário dimensionar a resistência da base.

Para isso, utiliza-se a *Kirchhoff Voltage Law* (KVL) entre a base do transístor e o emissor, obtendo:

$$-3.3 + R_{\mathrm{b}}.I_{\mathrm{b}} + V_{\mathrm{be_{ON}}} = 0 \tag{4.2}$$

Assim, e utilizando a Equação 4.1, obtém-se $I_{\mathrm{b}} = 11\,\mathrm{mA}$ e que $R_{\mathrm{b}} = 2.7\,\mathrm{k\Omega}$. Visto que, o ganho intrínseco do transístor pode variar e de forma a poupar no consumo energético, alterou-se experimentalmente a $\mathrm{R_b}$ até do relé deixar de ativar. Esse valor escolhido foi de 20 kΩ.

Para implementar esta função, é necessário o seguinte material:

- 1 Bateria de 9 V;
- 2 Transístores 2N2222A [37];
- 4 Resistências de 10 kΩ;
- 2 Díodos 1N4007 [38];
- 2 Relés de 12 V 899-1C-F-C-12VDC [39].

Os relés escolhidos são compostos por 5 terminais, conforme a figura seguinte.

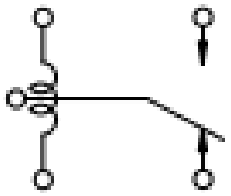

Figura 4.4: Esquemático do relé 899-1C-F-C-12VDC [39]

Os relés, são ativados quando aos terminais das suas bobines, é aplicada a tensão necessária para tal. Assim, o interruptor altera a sua posição. Para ativar o sentido do motor que se quer ativar, será necessário ativar o *pin* digital pretendido e, assim, o interruptor mudará a sua posição e ligará a função pretendida. De notar que, só um relé poderá estar ligado visto que, caso contrário, o motor do estore não funcionará corretamente, permitindo até, que o equipamento deixe de funcionar. Utiliza-se um díodo para proteger o transístor de um pico de corrente aquando da mudança de posição do interruptor.

Então, os *pin* 11 e 22 (GPIO 5 e GPIO 4, respetivamente) encontram-se conectados à base de um transístor através de uma resistência de 20 kΩ, ativando ou desativando o relé correspondente.

Assim, o circuito para esta função é o apresentado na Figura 4.5.

Para uma melhor compreensão, na figura 4.6 apresenta-se o esquemático da Figura 4.5.

O código associado ao controlo do estore encontra-se na Listagem I.4

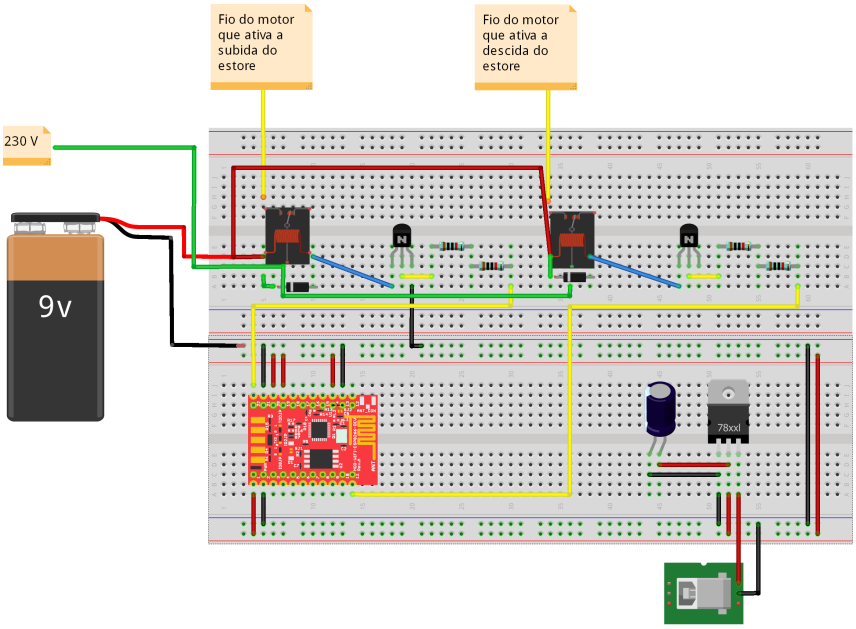

Figura 4.5: Circuito para ativação do motor

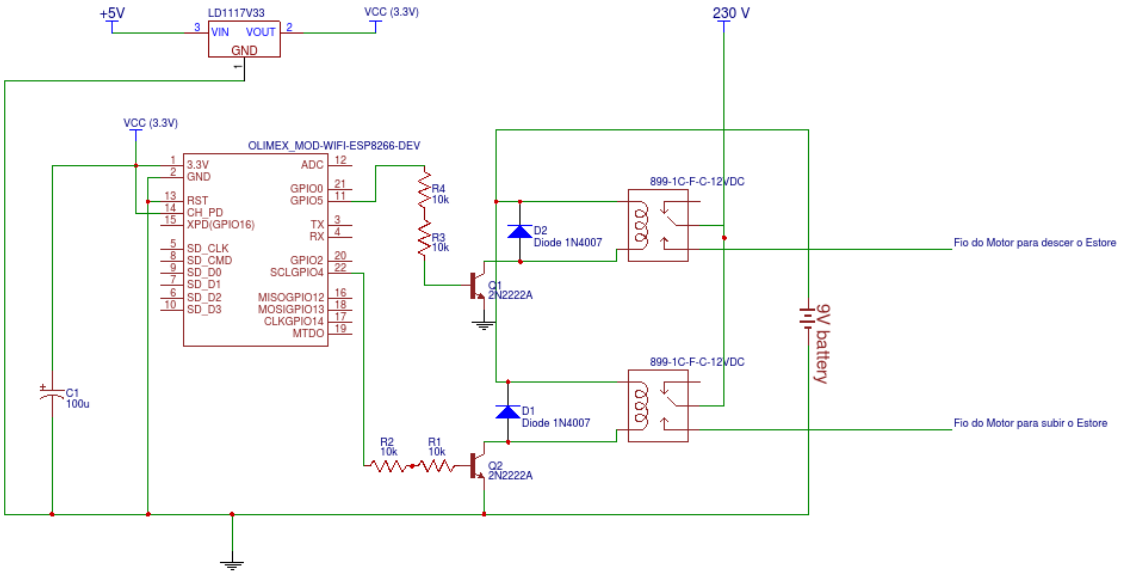

Figura 4.6: Esquemático para ativação do motor

4.2.2 Ligação ao sensor de temperatura

O *datasheet* do sensor de temperatura LM35 [24] indica que, no *pin* de output do sensor, a tensão obtida é proporcional à temperatura (10 mV = 1 ºC), ou seja, por um aumento de 1ºC na temperatura, a saída do sensor aumentará 10 mV. Também nos é indicado que, caso não se use uma resistência à saída, apenas nos é permitido obter valores entre 2 °C e 150 °C, o que, para este projeto, é o indicado.

Na figura 4.7, está representada a relação entre a tensão de saída do sensor e a temperatura medida por este.

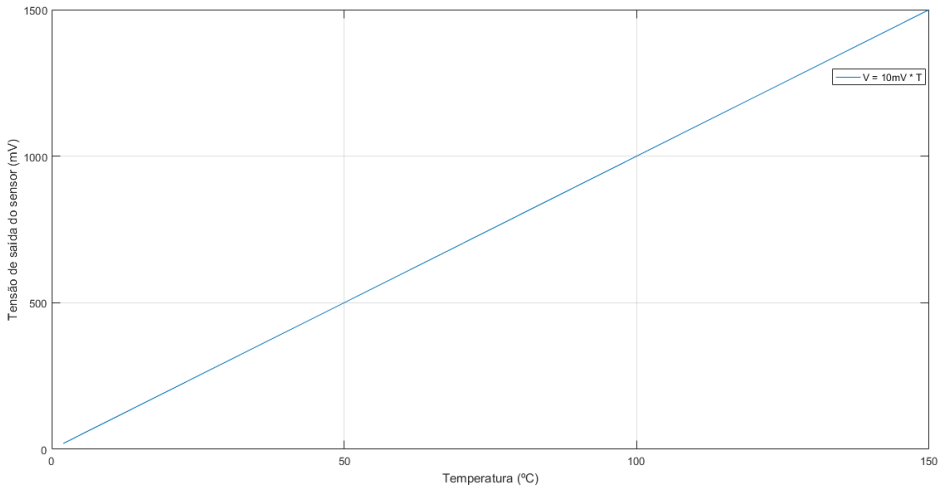

Figura 4.7: Gráfico da Relação da Temperatura com a Tensão de saída do sensor LM35

Tendo de se obter valores de tensão a partir do sensor, é necessário conectar a saída do sensor ao *pin* 12 [ADC]do ESP8266. Após testes realizados, foi concluído que o *pin* ADC do microprocessador utilizado, mede valores entre 0 e 1.164 V. Assim, o circuito utilizado é o representado na Figura 4.8.

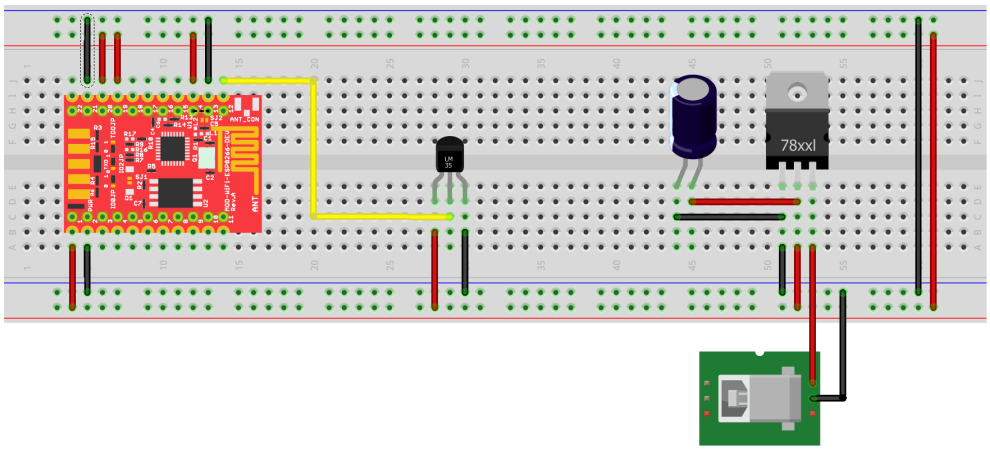

Figura 4.8: Circuito de ligação do ESP8266 ao sensor LM35

Para um melhor entendimento, na figura 4.9 apresenta-se o esquemático da Figura 4.8.

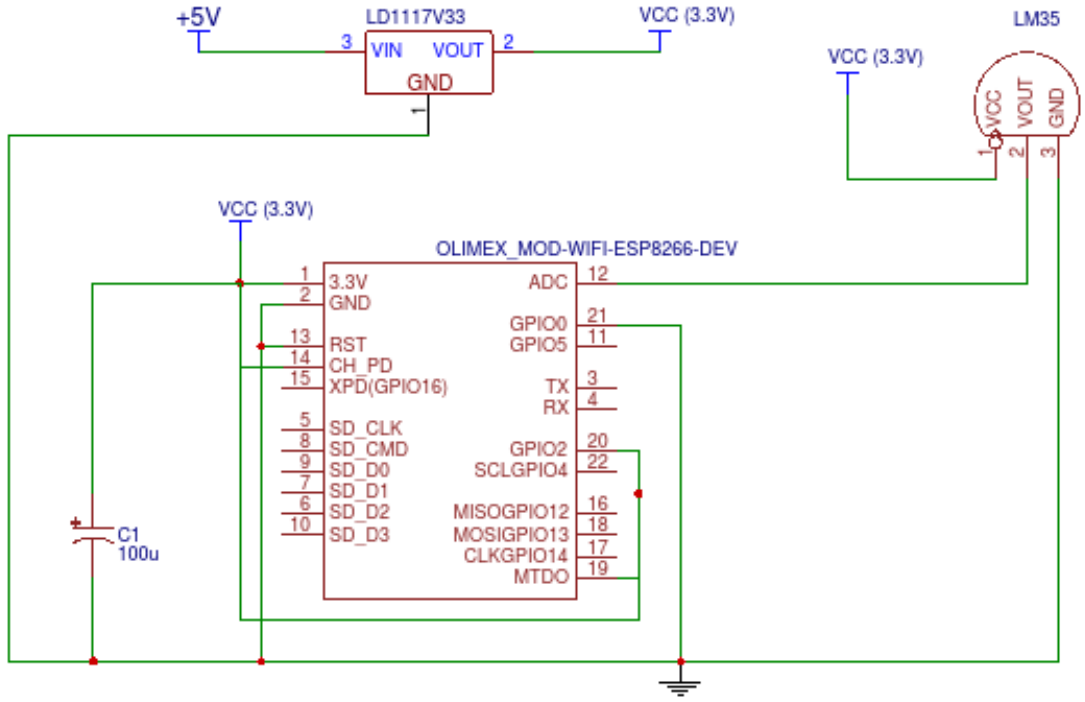

Figura 4.9: Esquemático do Circuito de ligação do ESP8266 ao sensor LM35

4.2.3 Ligação ao sensor de Luminosidade

O sensor é uma foto-resistência em que a 10 Lux tem uma resistência 5 kΩ-10 kΩ e que, essa resistência, varia com a intensidade da luz. Em série, estará uma resistência de 20 kΩ, composta pela série de 2 resistências de 10 kΩ. A um dos terminais do sensor encontra-se ligada a resistência de 20 kΩ e, entre a resistência e a foto-resistência, conecta-se o *pin* 12 do ESP8266. É necessária a resistência de 20 kΩ porque o *pin* ADC só consegue processar sinais até cerca de 1.1V. Caso a foto-resistência detete que se encontra num local muito escuro, a tensão será alta e ultrapassará o valor de 1.1 V. Para isso, a nível de código de programação, considerar-se-à 4 níveis de intensidade da luz, representados na tabela seguinte.

Tabela 4.2: Tabela de Níveis de luminosidade correspondentes ao valor do *pin* ADC

	0-255	**256-511**	**512-767**	**768-1024**
Baixa	-	-	-	X
Média Baixa	-	-	X	-
Média Alta	-	X	-	-
Alta	X	-	-	-

Na Figura 4.10 está representado o circuito para obtenção de valores de luminosidade.

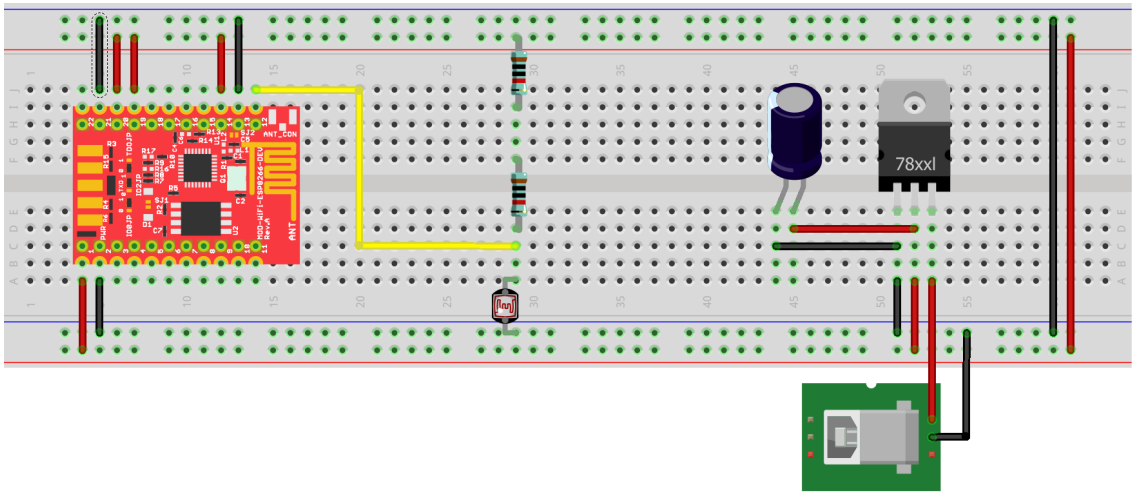

Figura 4.10: Circuito de ligação do ESP8266 ao sensor de Luminosidade

Na Figura 4.11 está representado o esquemático do presente na Figura 4.10.

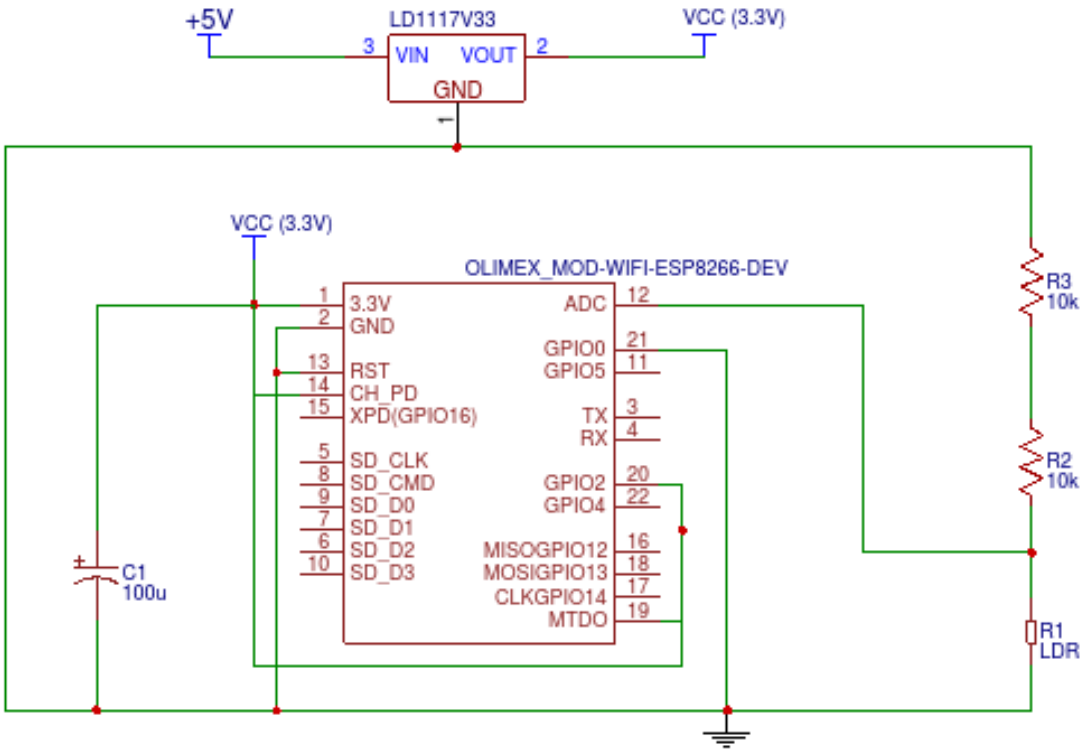

Figura 4.11: Esquemático do Circuito de ligação do ESP8266 ao sensor de Luminosidade

4.2.4 Ligação do Sensor de Temperatura e de Luminosidade ao *pin* ADC

Como observado nas secções 4.2.2 e 4.2.3, os sensores de temperatura e de luminosidade terão que ser ligados ao *pin* ADC. Como constatado em [**esp8266**], o microprocessador dispõe, apenas, um *pin* deste tipo. Então, é necessário realizar a multiplexagem dos

dois sinais.

Para isso, foi utilizado um *multiplexer* de 8 bits. Esse *multiplexer* é o CD74HC4051E [40]. Este foi o *multiplexer* escolhido devido a ter um R_{ON} baixo (cerca de 7 Ω). Na figura 4.12 está representado os as funções dos *pins* do *multiplexer*.

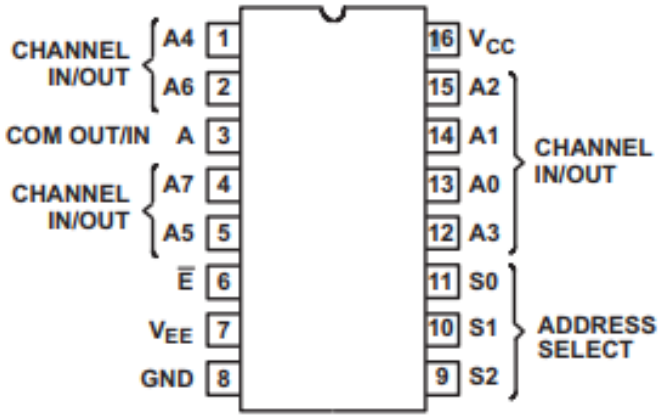

Figura 4.12: Funções dos *Pins* do CD74HC4051E [40]

Como observado na Tabela 4.3, o pin 6 (ENABLE) é *active Low*, o que significa que o *multiplexer* apenas selecionará um dos canais se este *pin* se mantiver em *LOW*. Consequentemente, a tensão que está à entrada do canal ativo, será disponibilizada à saída do *pin* 3 (COM OUT/IN).

Tabela 4.3: Tabela de Verdade do CD74HC4051E [40]

PINS DE ENTRADA				**CANAL ATIVO**
ENABLE	**S_2**	**S_1**	**S_0**	
L	L	L	L	A0
L	L	L	H	A1
L	L	H	L	A2
L	L	H	H	A3
L	H	L	L	A4
L	H	L	H	A5
L	H	H	L	A6
L	H	H	H	A7
H	X	X	X	NENHUM

H = sinal lógico elevado (*HIGH*)
L = sinal lógico baixo (*LOW*)
X = *Don't Care*

Observando a Tabela 4.3, e tendo em conta que é necessário, apenas, que dois sensores se conectem ao mesmo *pin*, escolheu-se utilizar o canal A0 (*pin* 13) e o canal A1 (*pin* 14) para o sensor de temperatura e de luminosidade, respetivamente. Isto implica que S_1 e S_2 estejam ligados à massa e, S_0 se encontre conectado a um *pin* digital que será alterado

para LOW ou HIGH quando se queira o sensor de temperatura ou o de luminosidade, respetivamente.

Assim, o circuito completo é o apresentado na Figura 4.13.

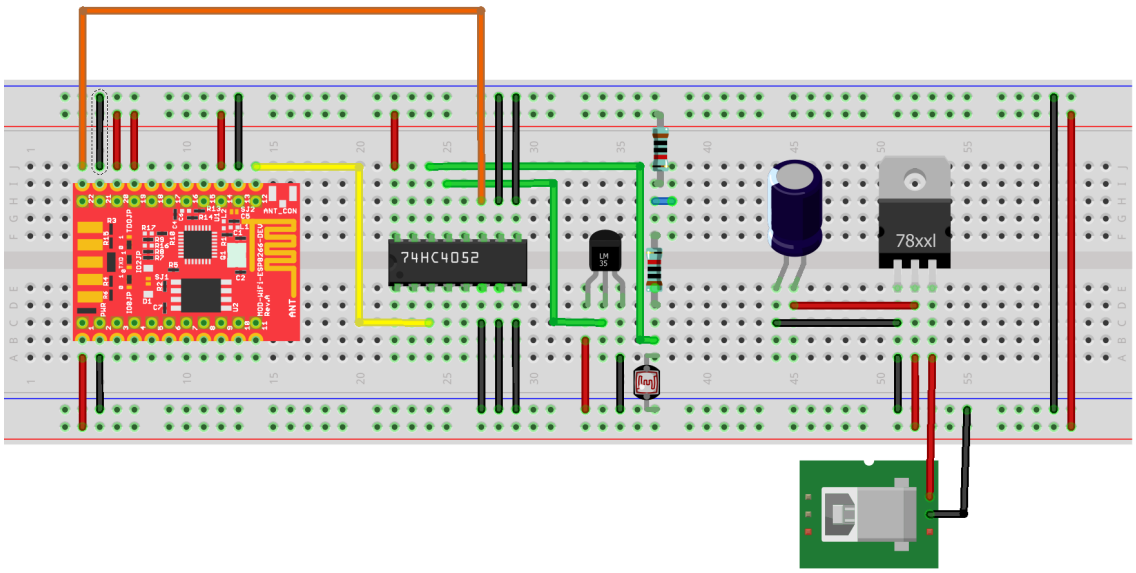

Figura 4.13: Circuito utilizando o *multiplixer* para seleção de um dos sensores

Para uma melhor perceção do circuito, na figura 4.14 apresenta-se o seu esquemático.

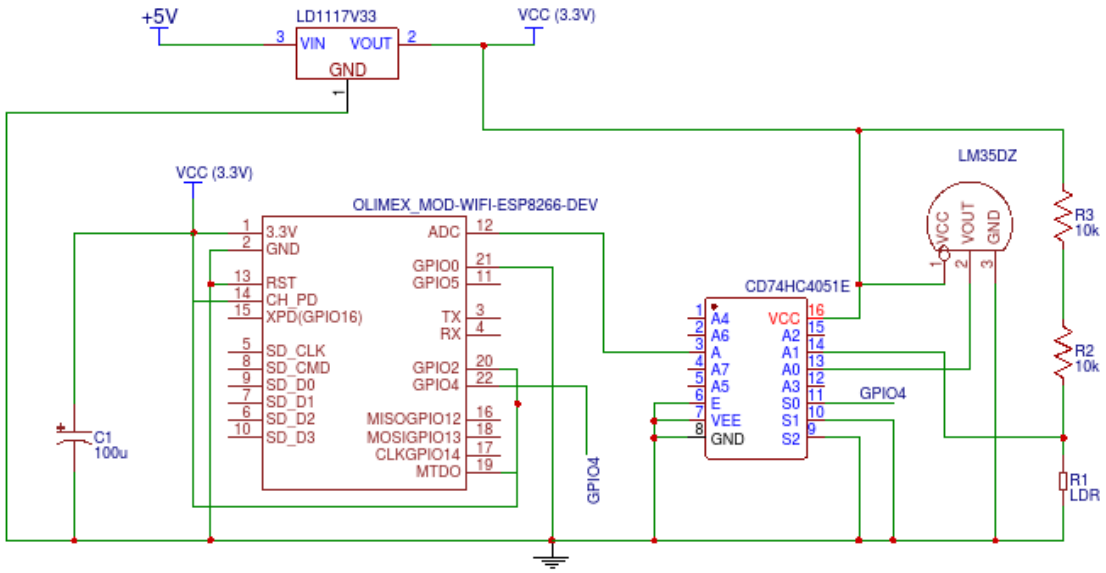

Figura 4.14: Esquemático do Circuito utilizando o *multiplixer* para seleção de um dos sensores

O código associado à obtenção de valores dos sensores encontra-se na Listagem I.3.

4.2.5 Receber e enviar informações para o processador presente no exterior

A ideia geral será que o processador central, visto que estará próximo do motor do estore, poderia ser alimentado através da rede elétrica a que o motor estaria ligado, e assim permaneceria sempre ligado. Em contrapartida, o processador que se encontra no exterior, no estore, utilizará algum modo de poupança de energia.

No ESP8266, está disponível um modo que permite desligar o WiFi mas continuar a processar dados e será este o modo utilizado. Na Figura 4.15, encontra-se representado o esquema de comunicação entre os dois processadores.

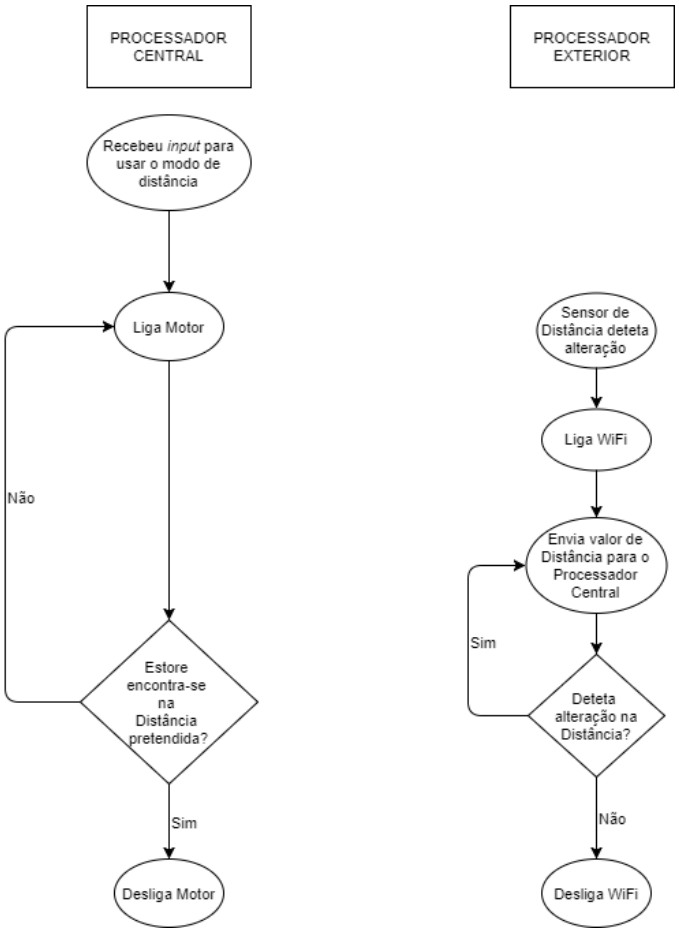

Figura 4.15: Esquema da comunicação entre ambos os Processadores

4.2.6 Ligação à *Cloud* para Armazenamento de Dados

A ligação à *Cloud* da IBM é feita através do protocolo *Message Queuing Telemetry Transport* (MQTT) que enviará uma mensagem com todos os dados disponíveis no SoC central (distância, temperatura, luminosidade, *thresholds* e modo automático).

O formato da mensagem é o seguinte:

"Name": "18FE34D81E46",

"cm": 0,

"LumT": 0,
"TempT": 0,
"DistT": 0,
"LumValue": 0,
"TempValue": 0,
"DistValue": 0,
Em que cada entrada tem o seguinte significado:

- "Name"representa o ID do dispositivo e corresponde ao *MAC address*;
- "cm"representa o modo de controlo ativado em que, 0 significa controlo manual (distância) e 1 representa controlo automático (temperatura e luminosidade);
- "LumT"representa o *threshold* da luminosidade e pode ter valores de 0-3;
- "TempT"representa a o *threshold* da temperatura e pode obter valores entre 0 e 100;
- "DistT"representa o valor da percentagem de fecho do estore pretendido pelo utilizador;
- "LumValue"representa o nível calculado pelo sensor de Luminosidade e pode obter valores de 0-3;
- "TempValue"representa o valor de temperatura obtido pelo sensor de temperatura.
- "DistValue"representa o valor de distância medido pelo sensor de distância;

O código associado a esta função encontra-se na Listagem I.1.

4.2.7 Resumo das ligações aos GPIO's do Processador Central

De forma a facilitar o entendimento de quais os *pins* utilizados para compor o sistema, na Tabela 4.4 está descrito as ligações utilizadas.

Tabela 4.4: Tabela com Ligações dos sensores aos respetivos *pins* do Processador Central

GPIO	Sensor
4	*Pin* de Seleção S0 do *Multiplexer*
5	Relé para ativar motor (subida do estore)
11 (ADC)	Saída *Multiplexer*
13	Relé para ativar motor (descida do estore)

4.3 Processador Exterior

Como já referenciado na secção 4.2.5, este processador estará colocado no estore com um sensor e um LED de IV que, em conjunto, compõe um sensor de distância.

Esta será a única função deste processador, recolher os valores deste sensor e enviá-los para o processador central. Para tal, é utilizado o sensor TSSP58P38 para receber sinais IV com frequência de, aproximadamente, 38.5 kHz. Então, é necessário criar um circuito que consiga modular o sinal de IV proveniente do LED IV (TSAL6200). Assim, foi utilizado um Timer 555 [41] ajustado para esta frequência e funcionará no modo de de vibrador astável (oscilador). O circuito para este modo está representado na Figura 4.16 e as funções de cada *pin* poderão ser consultadas no *datasheet* [41].

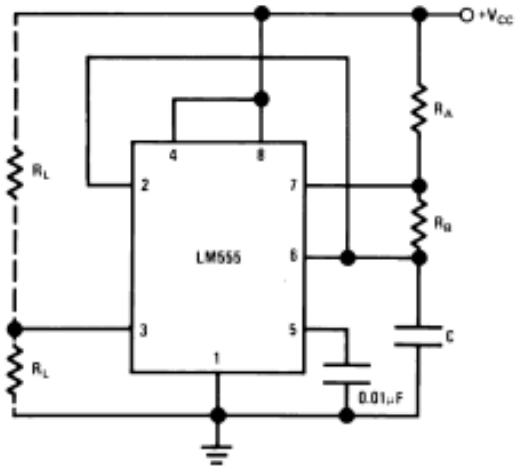

Figura 4.16: Esquemático do Timer 555 para funcionamento de oscilador

Através do *datasheet* obtemos as seguintes equações:

$$f = \frac{1}{T} = \frac{1.44}{(R_A + 2R_B)C} \tag{4.3}$$

$$t_1 = 0.693(R_A + R_B)C \tag{4.4}$$

$$t_2 = 0.693(R_B)C \tag{4.5}$$

$$T = t_1 + t_2 \tag{4.6}$$

Onde:

f representa a frequência;

T representa o período

R_A representa a resistência entre V_{CC} e o *pin* 7 do Timer 555;

R_B representa a resistência entre o *pin* 6 e o *pin* 7 do Timer 555;

C representa o condensador ligado entre o *pin* 6 e GND.

t_1 representa o tempo que o sinal de saída se encontrará a *HIGH*;

t_2 representa o tempo que o sinal de saída se encontrará a *LOW*;

Com estas equações, conseguimos obter o valor dos componentes R_A, R_B e C para colocar o Timer 555 a funcionar a uma frequência de 38.5 kHz (de modo a ter em conta a incerteza dos componentes visto que o sensor de IV funciona a 38 kHz).

Colocando $R_A = 1\,k\Omega$, $C = 4.7\,nF$ e $f = 38.5\,kHz$ obtemos a partir da Equação 4.3 que:

$$38500 = \frac{1.44}{(1000 + 2.R_B)4.7\,nF}(=) \tag{4.7}$$

$$R_B = 3479\,\Omega \tag{4.8}$$

Com este valor de R_B, obtém-se os seguintes valores para T_1 e para T_2:

$$t_1 = 14.59\,\mu s \tag{4.9}$$

$$t_2 = 11.33\,\mu s \tag{4.10}$$

Ainda, observa-se que, o sinal de saída apresenta um *duty cycle* equilibrado, como se pode observar na Equação 4.11:

$$D = \frac{t_1}{t_1 + t_2} x100 = 56.29 \tag{4.11}$$

De notar que, a entrada de alimentação do Timer 555 será ligada ao GPIO 4 do ESP8266 de forma a ativar o Timer 555 quando for necessário.

É utilizado um transístor à saída do Timer 555 visto que, como os GPIO's do ESP8266 só conseguem emitir 20 mA no máximo, esta corrente não permite que o LED tenha um alcance grande. Assim, com o transístor, essa corrente é amplificada.

Na Figura 4.17, apresenta-se o esquemático completo para este processador.

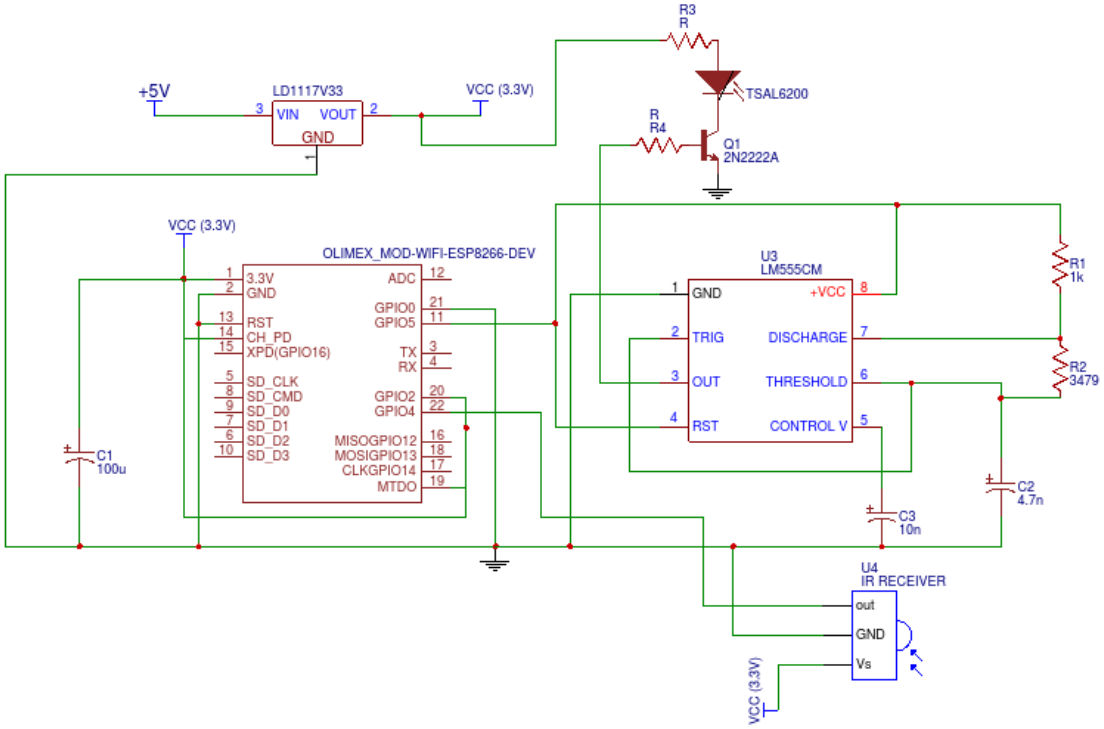

Figura 4.17: Esquemático do circuito do Processador Exterior

São utilizados 3 potenciómetros: um para limitar a corrente no LED diretamente e ser mais fácil controlar o *range* que a onda de IV se propaga, outro para controlar a corrente de base a ser amplificada pelo transístor e, por fim, um potenciómetro para afinar a frequência a que o Timer 555 disparará.

Por fim, no ESP8266 é colocado o GPIO5 a *HIGH* durante cerca de 200 µs e, posteriormente, analisará quanto tempo o *pin* GPIO4, conectado ao sensor de IV, se encontrou a *LOW*. Assim, e baseado no *datasheet* do sensor, quanto maior for o tempo que este *pin* se encontrou a *LOW*, mais próximo um objeto se encontra, ou seja, mais próximo o estore se encontra de estar fechado.

O código associado a este Processador encontra-se no Anexo II.

4.4 Aplicação *Smartphone*

Nesta secção são apresentados todos os ecrãs na aplicação *Smartphone* desenvolvida. O programa para desenvolvimento da aplicação foi o *Android Studio*.

As atividades desenvolvidas foram as seguintes:

1. *Main Activity*;
2. *Options Activity*;
3. *Control Mode Activity*;

4. *Update Thresholds Activity*;
5. *Informations Activity*;
6. *Cloud Info Activity*;
7. *Device Info Activity*.

O código associado à aplicação *Smartphone* encontra-se no Anexo III.

4.4.1 *Main Activity*

Esta atividade é a atividade onde a *app* é iniciada. Contém uma caixa de texto com instruções de utilização da *app* e permite aceder às outras atividades através de um menu lateral.

Na Figura 4.18 está representada o ecrã desta atividade e na Figura 4.19 apresenta-se o menu de troca de atividades.

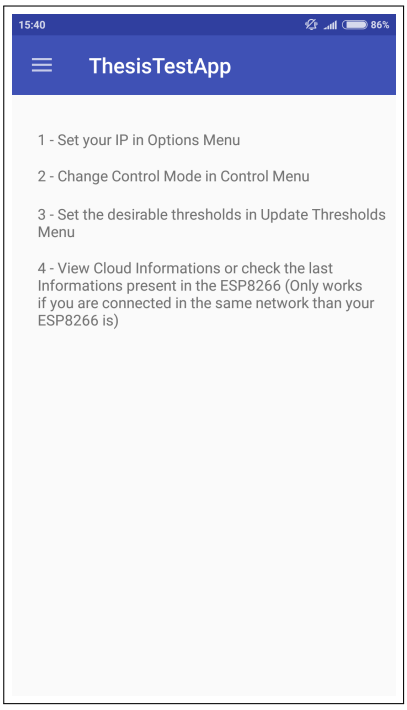

Figura 4.18: Ecrã da Actividade *Main Activity*

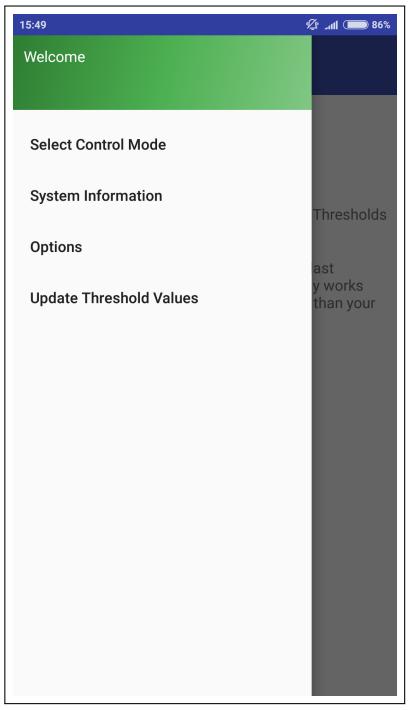

Figura 4.19: Menu que permite aceder a outras actividades

4.4.2 *Options Activity*

Nesta atividade está disponível 2 caixas de texto para *input* do utilizador. Aqui, na caixa de "IP", o utilizador coloca o IP do seu *router* para poder aceder ao ESP8266 caso não esteja na mesma rede. Este IP é o utilizado pelas atividades *Control Mode* e *Update Thresholds*.

No caso da caixa de texto "Port", é o número do porto utilizado. O protocolo *Hypertext Transfer Protocol* (HTTP) utiliza o porto 80, por isso, o utilizador terá que colocar esse valor.

Por fim, o botão "Save"tem a função de guardar estas informações numa classe chamada "*Shared Preferences*". Esta classe permite que todas as atividades tenham acesso à mesma informação e, caso a aplicação seja fechada, guarda essa mesma informação para um próxima utilização.

De igual forma, através de um deslize de um dedo no lado esquerdo do ecrã, é aberto o menu que permite aceder às outras atividades, semelhante ao apresentado na Figura 4.19.

Na figura 4.20 está representada esta atividade.

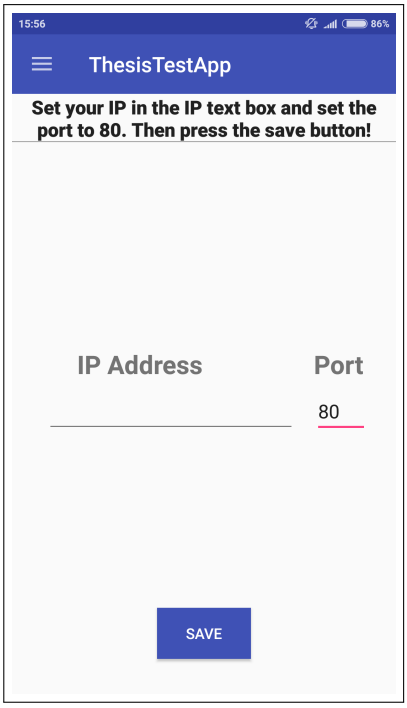

Figura 4.20: Ecrã da actividade *Options Activity*

4.4.3 *Control Mode Activity*

Nesta atividade é exposto um botão onde é possível alterar o modo de controlo do sistema (Manual ou Automático). O utilizador ao pressionar no botão, é enviado uma mensagem para o IP gravado na atividade "*Options Activity*". Ao utilizar-se o protocolo HTTP, o URL será o seguinte:

http://"ipAddress":"portNumber"/control_mode

Onde:

"ipAddress" - IP colocado na atividade "*Options Activity*;

"portNumber" - Porto colocado na atividade "*Options Activity*;

control_mode - *String* que o microprocessador reconhece como o utilizador quer alterar o modo de controlo do estore.

Na Figura 4.21 está representado o ecrã desta atividade.

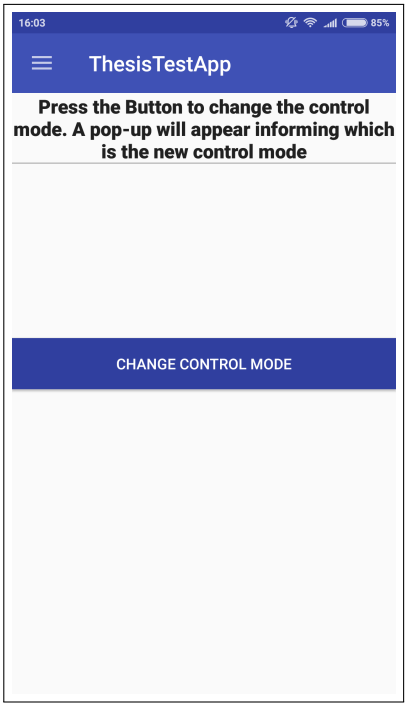

Figura 4.21: Ecrã da atividade *Control Activity*

Então, é criado um *pop-up* na aplicação em que informa o utilizador se a mensagem foi enviada ou não com sucesso. Caso seja verdade, é apresentada a mensagem que a mensagem foi enviada, e que se espera a resposta do microprocessador. Na figura 4.22 está representado este *pop-up*.

Figura 4.22: *Pop-up de sucesso de mensagem enviada*

Após chegar a mensagem do microprocessador, é apresentado qual foi o modo ativado. Na Figura 4.23 e na Figura 4.24 estão representados os 2 casos possíveis.

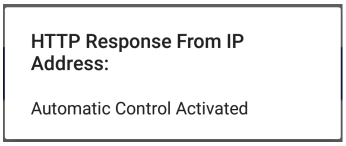

Figura 4.23: Controlo Automático ativado

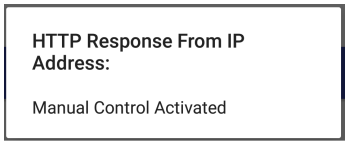

Figura 4.24: Controlo Manual ativado

4.4.4 *Update Thresholds Activity*

Nesta atividade são apresentados 3 caixas de texto (Luminosidade, Temperatura e Distância). Estas podem ser alteradas pelo utilizador em que escreve o valor pretendido.

A caixa referente a luminosidade poderá obter valores de 0-3 que representam os 4 modos explicitados na Tabela 4.2 em que 0 corresponde a Luminosidade Alta e 3 corresponde a Luminosidade Baixa.

A caixa de texto referente à temperatura poderá obter valores entre 0-50 ºC.

Por fim, a caixa de distância poderá obter valores de 0-100 %. O valor de 0% significa que o utilizador quer o estore aberto na totalidade e o valor de 100% que o oposto, o estore completamente fechado.

Estes valores, são enviados para o microprocessador de uma forma semelhante à apresentada na subsecção 4.4.3. A mensagem enviada tem o seguinte formato:

http://"ipAddress":"portNumber"/threshold_update/
"luminosityThreshold"; "temperatureThreshold";distanceValue

Onde:

"ipAddress" - IP colocado na atividade "*Options Activity*;

"portNumber" - Porto colocado na atividade "*Options Activity*;

threshold_update - *String* que o microprocessador reconhece como o utilizador quer alterar os valores dos *thresholds* da luminosidade, temperatura e distância;

luminosityThreshold - Valor que o utilizador colocou na caixa de texto referente à luminosidade;

temperatureThreshold - Valor que o utilizador colocou na caixa de texto referente à temperatura;

distanceValue - Valor que o utilizador colocou na caixa de texto referente à distância.

Os valores são separados por ";" de forma a facilitar a decomposição da mensagem do lado do microprocessador.

Após o envio da mensagem, é aberto um *pop-up* a indicar que a mensagem foi enviada, e, em caso de sucesso, é apresentada uma mensagem de que os valores foram alterados com sucesso.

Na Figura 4.25 e na Figura 4.26, estão representados os ecrãs desta atividade.

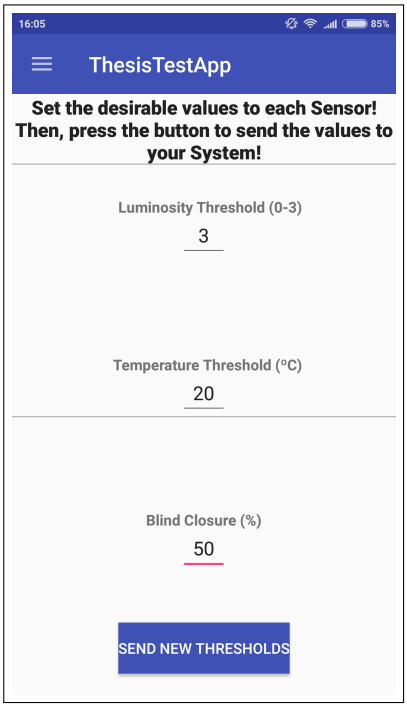

Figura 4.25: Ecrã da Atividade *Main Activity*

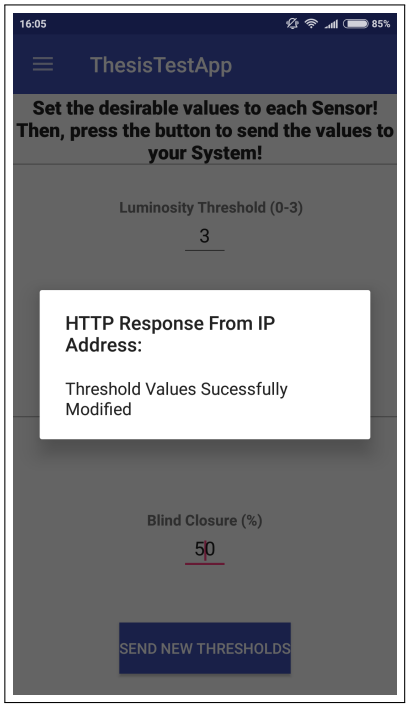

Figura 4.26: Menu que permite aceder a outras atividades

4.4.5 *Informations Activity*

No ecrã desta atividade são apresentados dois botões para selecionar as informações que se pretende observar:

- Informações presentes na *Cloud*;
- Informações obtidas através do microprocessador.

Na Figura 4.27 está representado o ecrã desta atividade.

4.4.6 *Cloud Info Activity*

Nesta atividade são apresentadas as informações gravadas na *Cloud* através de gráficos e de *Gauges*. Para isso, utilizou-se uma *Web View* na aplicação que permite aceder ao *site* escolhido.

A *Cloud* IBM Bluemix contém uma ferramenta (*Nodered*) que permite construir os gráficos através das mensagens recebidas. Para obter essas informações, configura-se a *Web View* para o URL respetivo da nossa aplicação no *NodeRed*, neste caso, **https://noderedthesis.eu-gb.mybluemix.net/ui** e obtém-se o apresentado na Figura 4.28, Figura 4.29 e Figura 4.30.

Estes gráficos são compostos por nós, conectados entre si e dispostos da forma apresentada na Figura 4.31. Dentro de cada um dos nós, existe código programado para retirar o valor que o nó necessita (ex: nó "temperature"retira o valor da temperatura enviado na mensagem).

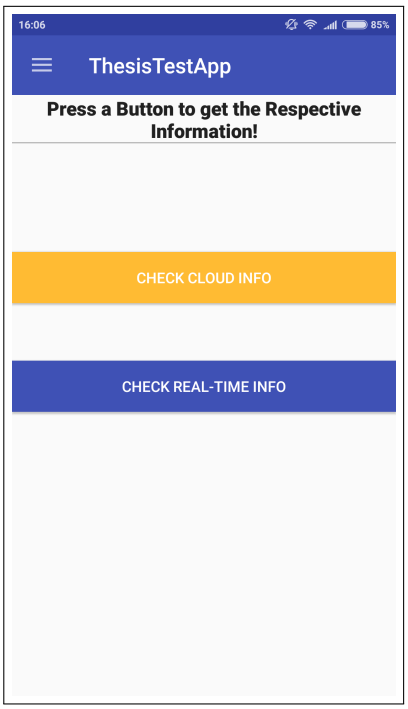

Figura 4.27: Ecrã da atividade *Informations Activity*

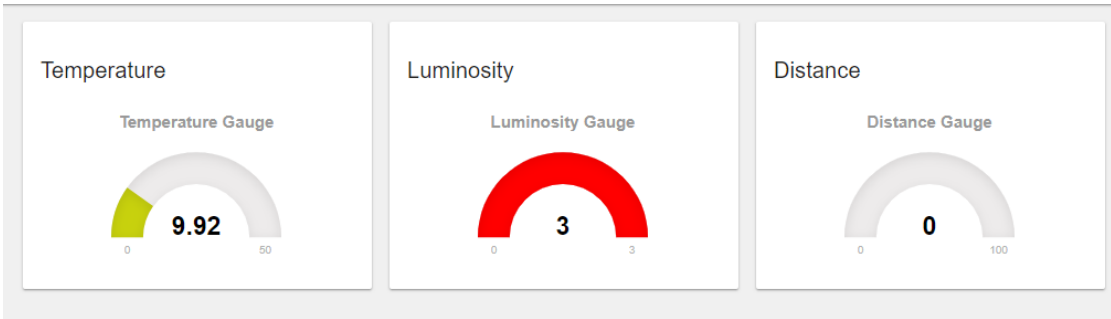

Figura 4.28: *Gauges* apresentados na atividade *Cloud Info Activity*

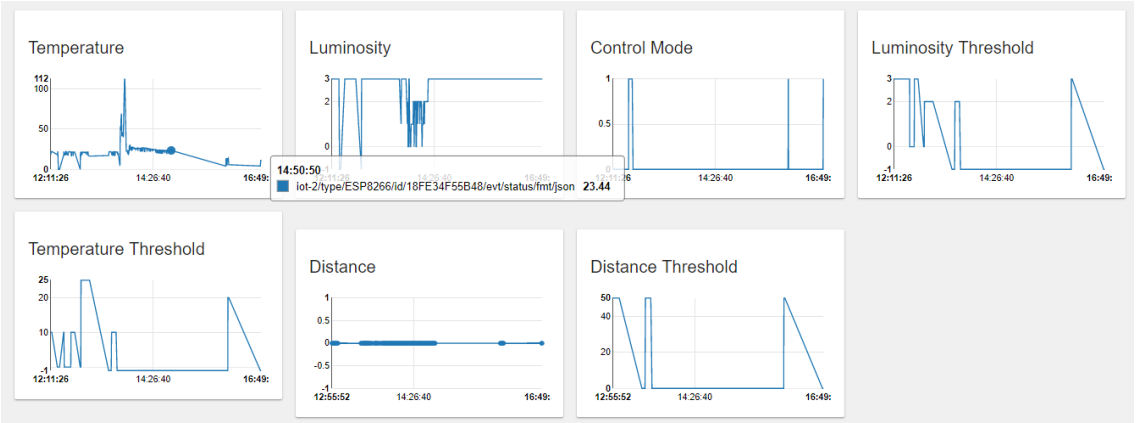

Figura 4.29: Gráficos apresentados na atividade *Cloud Info Activity*

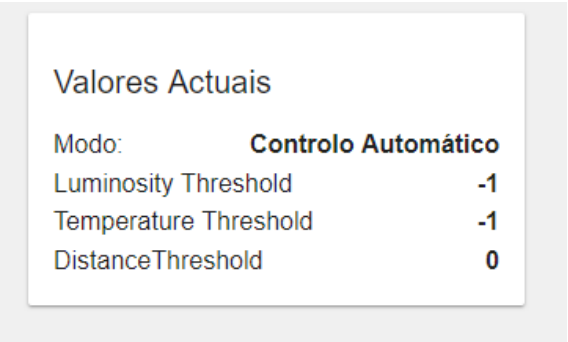

Figura 4.30: Estado da última atualização na *Cloud* nas variáveis de controlo apresentado na atividade *Cloud Info Activity*

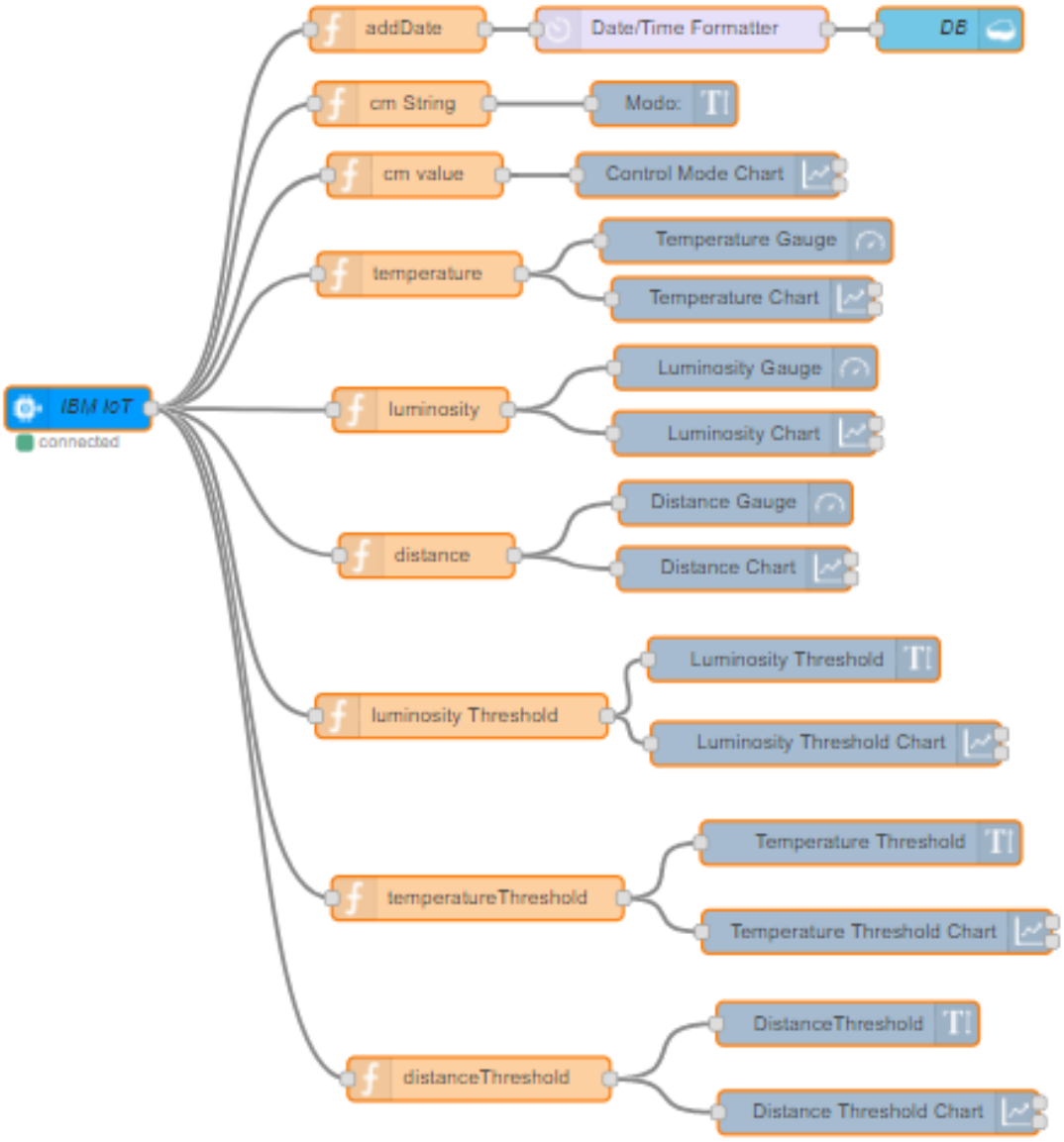

Figura 4.31: Configuração dos nós na ferramenta *Node-RED*

4.4.7 *Device Info Activity*

Por fim, esta atividade contém várias caixas de texto e um botão *Refresh* para obter as informações mais recentes do microprocessador e colocá-las nas respetivas caixas.

O método é semelhante ao utilizado nas outras atividades, mas com uma diferença no IP. A mensagem enviada é a seguinte:

http://"ipAddress":"portNumber"/information

Onde:

"ipAddress" - IP 192.168.1.102 o que implica que, apenas se conseguirá obter informações diretamente do microprocessador se o *Smartphone* estiver conectado à mesma rede do microprocessador;

"portNumber" - Porto 80;

information - *String* que o microprocessador reconhece como o utilizador querer obter informações diretamente.

Ao receber a mensagem com as informações, a *app* decompõe a mensagem recebida e coloca nas respetivas caixas.

Na Figura 4.32, está representado o ecrã desta atividade.

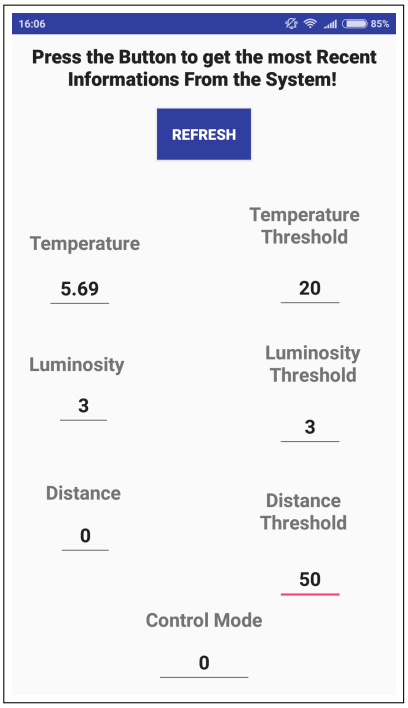

Figura 4.32: Ecrã da atividade *Device Info Activity*

Capítulo 5

Conclusões e Trabalho Futuro

Nesta secção são apresentadas algumas conclusões e observações sobre o sistema implementado. Ainda são apresentadas algumas melhorias que podem ser realizadas e algumas indicações para desenvolvimento de trabalho futuro.

5.1 Discussão de Resultados

Detetou-se uma diferença de aproximadamente 40 mV entre o *ground* do processador central e do *ground* do sensor de temperatura, ou seja, se a tensão entre os *pins* de saída e *ground* do sensor de temperatura for de 250 mV (25ºC), o ADC lê 210 mV (21ºC). Este problema poderá ser explicado pelo facto de ser utilizada uma *breadboard* (resistência interna) como suporte.

O ESP8266, ao ter o WiFi ligado e fazer leituras ao *pin* do ADC, ao final de alguns segundos, os valores obtidos neste *pin* são inflacionados em cerca de 20-30 mV. Após algumas pesquisas em documentação associada ao ESP8266, concluiu-se que é um problema generalizado presente em toda a comunidade que utiliza este processador.

O facto de o ESP8266 ter disponível apenas 1 *pin* analógico (ADC), foi necessário a utilização de um *multiplexer*. Estes circuitos integrados têm uma resistência interna associada que faz variar em alguns mV a tensão no *pin* ADC. Apesar disso, o *multiplexer* utilizado tem uma resistência interna bastante baixa que pouco afeta a tensão referida.

O sensor de luminosidade utilizado (LDR), quando exposto a luminosidade baixa, tem uma resistência muito elevada (superior a 100 kΩ) e, devido ao divisor de tensão utilizado, a tensão no *pin* ADC neste caso será bastante superior ao máximo suportado (1.1 V).

Após diversos testes, constatou-se que o sensor de distância (LED de IV + Sensor de IV) não é fiável o suficiente. Como foi indicado, é avaliado o tempo que o sensor

IV tem o seu *output* a *LOW* baseado na proximidade do objeto (quanto mais próximo, maior o tempo que o *pin* se encontra a *LOW*) para obter um valor de distância. Mas, este tempo varia bastante para pequenas movimentações. Notou-se que, com movimentações de afastamento (distância a aumentar), o tempo diminui não linearmente, estabilizando posteriormente.

O WiFi demora cerca de 4 segundos a conectar-se a uma rede. Este tempo impossibilita uma maior rapidez no envio do valor da distância do processador exterior para o processador central. Uma das principais características do processador exterior é a eficiência energética, o que é obtido por desligar o WiFi quando não utilizado, mas, torna a comunicação mais lenta por ter que aguardar pela conexão à rede. Foi desenvolvida a solução em que o WiFi fica permanentemente ligado.

O processador central tem uma grande sobrecarga de conexões (*app*, *Cloud* e processador exterior). Caso o processador central necessite de publicar na *Cloud* poucos milissegundos depois de ter recebido/enviado informação para *app* ou para o outro processador, esta publicação não será bem sucedida. Foram colocados alguns *delays* no programa para contornar este problema, criando assim, alguma lentidão nas comunicações.

Os relés funcionam corretamente, apesar de a corrente de ativação ser inferior aos 100 mA referidos. Assim, desta forma, o circuito total gastará menos corrente, o que é uma vantagem.

Algumas das funções mencionadas no capítulo 3 não puderam ser implementadas devido ao processador central ser limitado e ficar bastante sobrecarregado facilmente, não conseguindo efetuar todas as funções a tempo, como já foi referenciado.

De referir que, este sistema foi construído com um gasto económico baixo (Tabela 2.4) o que permite que, caso este sistema seja produzido e vendido, se obtenha lucro a partir das vendas do produto porque os sistemas semelhantes ainda têm um preço alto.

5.2 Trabalho Futuro e Melhorias

Na secção 5.1 foi apresentado alguns problemas que o sistema desenvolvido apresenta. Assim, na presente secção, apresenta-se algumas soluções que visam solucionar estes problemas.

Alguns dos problemas referenciados podem ser resolvidos através da criação de uma *Printed Circuit Board* (PCB) com todos os componentes e o SoC escolhido.

Recomenda-se também que se utilize um novo SoC devido às limitações que este apresenta ao nível do *pin* ADC e do WiFi. E, também, pelo facto de os *pins* digitais apenas conseguirem fornecer 20 mA de corrente máxima. Também seria interessante utilizar um processador que processe diferentes ações ao mesmo tempo (processamento paralelo).

Sugere-se também a troca do LDR por um sensor de luminosidade digital, bem como, o sistema de distância (LED IV + recetor IV) por um sensor que devolva diretamente distância (ex: Sharp GP2Y0A21YK).

A nível do desenvolvimento da aplicação, é sugerido a criação de uma atividade de *Login*, com ligação a uma base de dados (na própria *Cloud* da IBM, por exemplo), em que cada utilizador poderia adicionar, alterar e remover os sistemas de estores em sua posse.

Também seria interessante criar um sistema que fosse capaz de controlar vários estores, em simultâneo. Ou seja, evoluir o sistema criado para um sistema em que, existisse de igual forma um processador central capaz de atuar sobre vários estores em que, cada um desses estores, teria associado um processador conectado a um sensor de luminosidade (semelhante ao circuito do processador exterior).

De forma a melhorar o consumo energético do circuito, sugere-se também a adição de células fotovoltaicas ligadas à bateria do processador exterior de forma a recarregar a bateria.

Recomenda-se o teste deste sistema a controlar diretamente um motor de um estore visto que não foi possível testá-lo devido ao preço de um equipamento destes.

Abordando a segurança do sistema, é necessário criar um método de segurança para efetuar a comunicação com o SoC. Por exemplo, criar um método de autenticação baseado no MAC *Address* do *Smartphone* que enviou o comando desejado.

Por fim, relativamente ao processador exterior e à sua comunicação WiFi e baixo consumo energético, recomenda-se que seja desenvolvido código em que o processador exterior processe dados mas o WiFi esteja desligado. Aquando de uma alteração no valor da distância obtido pelo respetivo sensor, o processador deverá ativar o WiFi, enviar os valores de distância até o estore ficar na posição desejada e, de seguida, desligar o WiFi e aguardar novamente por nova alteração do valor de distância. Assim, e visto que este processador deverá ser alimentado a uma bateria, obtém-se um consumo energético mais baixo.

Bibliografia

[1] Cisco. *Internet of Things*. URL: https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html?stickynav=3 (acedido em 16/09/2017).

[2] Boston Consulting Group. *Winning in IoT: It's All About the Business Processes*. URL: https://www.bcgperspectives.com/content/articles/hardware-software-energy-environment-winning-in-iot-all-about-winning-processes/ (acedido em 16/09/2017).

[3] Gartner. *Gartner Says 8.4 Billion Connected "Things"Will Be in Use in 2017, Up 31 Percent From 2016*. URL: http://www.gartner.com/newsroom/id/3598917 (acedido em 16/09/2017).

[4] J. S. Lee, Y. W. Su e C. C. Shen. "A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi". Em: *IECON Proceedings (Industrial Electronics Conference)*. 2007, pp. 46–51. ISBN: 1424407834. DOI: 10.1109/IECON.2007.4460126.

[5] G. Reiter. *Wireless connectivity for the Internet of Things*. 2014. URL: http://www.ti.com/lit/wp/swry010/swry010.pdf?DCMP=ep-con-wcs-cmtech{\%}7B{\%}5C{\&}{\%}7DHQS=ep-con-wcs-cmtech-bn-whip-en (acedido em 04/06/2017).

[6] M. Bor e U. Roedig. "LoRa Transmission Parameter Selection". Em: (). URL: http://eprints.lancs.ac.uk/85515/4/lora{_}tps{_}r1342.pdf.

[7] A. Augustin, J. Yi, T. Clausen e W. Townsley. "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things". Em: *Sensors* 16.9 (2016), p. 1466. ISSN: 1424-8220. DOI: 10.3390/s16091466. URL: http://www.mdpi.com/1424-8220/16/9/1466.

[8] *Sigfox*. URL: https://www.sigfox.com/en/sigfox-iot-technology-overview (acedido em 10/07/2017).

[9] *CC3200MOD SimpleLink™ Wi-Fi® and Internet-of-Things Module Solution, a Single-Chip Wireless MCU*. URL: http://www.ti.com/lit/ds/symlink/cc3200mod.pdf (acedido em 05/06/2017).

[10] *SPWF01SA SPWF01SC Serial-to-Wi-Fi b/g/n intelligent modules*. 2017. URL: http://www.st.com/content/ccc/resource/technical/document/datasheet/ba/8c/b2/64/02/bc/4e/05/DM00102124.pdf/files/DM00102124.pdf/jcr:content/translations/en.DM00102124.pdf (acedido em 05/06/2017).

[11] *GS1011 Ultra Low-Power Wireless System-on-Chip Datasheet*. URL: http://www.semiconductorstore.com/pages/asp/DownloadDirect.asp?sid=1496663440639 (acedido em 05/06/2017).

[12] Cypress. *WICED™ IEEE 802.11 a/b/g/n SoC with an Embedded Applications Processor*. 2016. URL: http://www.cypress.com/file/298221/download (acedido em 05/06/2017).

[13] *ESP8266EX Datasheet*. URL: http://bbs.espressif.com/ (acedido em 05/06/2017).

[14] *Microsoft Azure*. URL: https://azure.microsoft.com/pt-pt/ (acedido em 07/06/2017).

[15] *Amazon Simple Storage Service (S3) - AWS*. URL: https://aws.amazon.com/s3/ (acedido em 07/06/2017).

[16] *Google Cloud*. URL: https://cloud.google.com/ (acedido em 07/06/2017).

[17] *IBM Bluemix*. URL: https://www.ibm.com/cloud-computing/bluemix/pt (acedido em 07/06/2017).

[18] R. Langari. *Measurement and Instrumentation*. 2012. ISBN: 9780123819604. DOI: 10.1016/C2009-0-63052-X. arXiv: arXiv:1011.1669v3.

[19] *TSAL6200 - High Power Infrared Emitting Diode, 940 nm, GaAlAs, MQW*. URL: https://www.vishay.com/docs/81010/tsal6200.pdf (acedido em 08/06/2017).

[20] *TSSP58038 - IR Receiver Module for Light Barrier Systems*. URL: https://www.vishay.com/docs/82479/tssp58038.pdf (acedido em 08/06/2017).

[21] *GP2Y0A21YK/GP2Y0D21YK*. URL: https://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf (acedido em 08/06/2017).

[22] *TSSP58P38 - IR Detector for Mid Range Proximity Sensor*. URL: https://www.vishay.com/docs/82476/tssp58p38.pdf (acedido em 08/06/2017).

[23] *BPW 21*. 2014. URL: http://www.mouser.com/ds/2/311/BPW21,Lead(Pb)FreeProduct-RoHSCompliant-318296.pdf (acedido em 07/06/2017).

[24] *Datasheet LM35 Precision Centigrade Temperature Sensors*. URL: http://www.ti.com/lit/ds/symlink/lm35.pdf (acedido em 26/08/2017).

[25] *Levolux Microcontroller*. URL: levolux.com/wp-content/uploads/2015/03/ElecControlSystems1.pdf (acedido em 10/01/2017).

[26] *Manual de Instruções WMS01ST*. URL: http://www.niceforyou.com/sites/default/files/upload/manuals/IST345.4862.pdf (acedido em 18/01/2017).

[27] *Manual de Instruções NEMO WSRT*. URL: http://www.niceforyou.com/sites/default/files/upload/manuals/IST275R01.4862.pdf (acedido em 20/01/2017).

[28] *SUNIS Indoor WireFree™ RTS Sun Sensor*. URL: https://www.somfysystems.com/file.cfm/Sunis{_}Indoor{_}Sensorhighres.pdf?contentid=200130 (acedido em 30/01/2017).

[29] *Manual de Instruções THERMOSUNIS WireFree™ RTS Light & Temperature Sensor*. URL: https://www.somfysystems.com/file.cfm/ThermoSunis{_}instructions.pdf?contentid=153650 (acedido em 02/02/2017).

[30] *Manual de Instruções SOLIRIS RTS*. URL: https://www.somfysystems.com/file.cfm/soliris{_}RTS{_}24V.pdf?contentid=275930 (acedido em 03/02/2017).

[31] *Manual de Instruções Ondeis™ WireFree RTS Rain & Sun Sensor*. URL: https://www.somfysystems.com/file.cfm/Rain{_}Sensor{_}Instructions{_}A4{_}print.pdf?contentid=313389 (acedido em 03/02/2017).

[32] *Lumero-868*. URL: https://www.elero.com/en/products/control-systems/lumero-868/ (acedido em 14/02/2017).

[33] *Lumo-868*. URL: https://www.elero.com/en/products/control-systems/lumo-868/ (acedido em 15/02/2017).

[34] *Aero-868/Aero-868 Plus/Aero-868 AC*. URL: https://www.elero.com/en/products/control-systems/aero-868-plus/ (acedido em 16/02/2017).

[35] *Sensero-868 AC/Sensero-868 AC Plus*. URL: https://www.elero.com/en/products/control-systems/sensero-868-ac-plus/ (acedido em 20/02/2017).

[36] *LD1117V33 Datasheet*. 2013. URL: http://www.mouser.com/ds/2/389/ld1117-974075.pdf (acedido em 26/08/2017).

[37] *Datasheet Transístor 2N2222A*. URL: http://www.onsemi.com/pub/Collateral/P2N2222A-D.PDF (acedido em 26/08/2017).

[38] *Datasheet Díodo Retificador 1N4007*. URL: http://www.vishay.com/docs/88503/1n4001.pdf (acedido em 26/08/2017).

[39] *Datasheet 899-1C-F-C-12VDC*. URL: http://www.mouser.com/ds/2/378/200982919182111463-7392.pdf (acedido em 26/08/2017).

[40] Schs122l –. *Datasheet CDx4HC405x, CDx4HCT405x High-Speed CMOS Logic Analog Multiplexers and Demultiplexers*. 1997. URL: http://www.ti.com/lit/ds/symlink/cd74hc4051.pdf (acedido em 27/08/2017).

[41] *LM555 Timer*. 2000. URL: http://www.ti.com/lit/ds/symlink/lm555.pdf (acedido em 02/09/2017).

ANEXO I

Código associado ao Processador Central

Listagem I.1: Envio da Mensagem para a *Cloud*

```
void sendToCloud() {
    if (!client.connected()) {
      Serial.print("Reconnecting client to ");
      Serial.println(server);
      while (!client.connect(clientId, authMethod, token)) {
        Serial.print(".");
        delay(500);
      }
      Serial.println();
    }

    String payload = "{\"d\":{\"Name\":\"18FE34D81E46\"";
    payload += ",\"cm\":";
    payload += control;
    payload += ",\"LumT\":";
    payload += thresholdLuminosity;
    payload += ",\"TempT\":";
    payload += thresholdTemperature;
    payload += ",\"DistT\":";
    payload += thresholdDistance;
    payload += ",\"LumValue\":";
    payload += luminosityValue;
    payload += ",\"TempValue\":";
    payload += temp;
```

```
25      payload += ",\"DistValue\":";
26      payload += distanceValue;
27      payload += "}}";
28
29      Serial.print("Sending payload: ");
30      Serial.println(payload);
31
32      if (client.publish(topic, (char*) payload.c_str())) {
33        Serial.println("Publish ok");
34      } else {
35        Serial.println("Publish failed");
36      }
37  }
```

Listagem I.2: Comunicação com o Exterior

```
1   void receivedInfo() {
2
3     // Wait until the client sends some data
4     Serial.println("new client");
5     while (!wifiClient.available()) {
6       delay(1);
7     }
8
9     // Read the first line of the request
10    String req = wifiClient.readStringUntil('\r');
11
12    // Match the request
13    if (req.indexOf("/control_mode") != -1) {
14      if (control == 0)
15        control = 1;
16      else
17        control = 0;
18    }
19    else {
20      if (req.indexOf("/information") != -1)
21        val_infor = 1;
22
23      else {
24        if (req.indexOf("/threshold_update/") != -1)
25          thresh = 1;
```

```
26
27          else {
28            if (req.indexOf("/distance/") != -1)
29              dist = 1;
30
31            else {
32              Serial.println("invalid request");
33              wifiClient.stop();
34              return;
35            }
36          }
37        }
38      }
39
40      if (thresh) {
41        String auxSize = "/threshold_update/";
42        int size = auxSize.length();
43        req1 = req.substring(req.indexOf("/threshold_update/")
44      + size);
45
46        int indexComma = req1.indexOf(";");
47        String req2 = req.substring(req.indexOf(";") + 1);
48
49        int indexSpace = req2.indexOf(" ");
50        t1 = req1.substring(0, indexComma);
51
52        indexComma = req2.indexOf(";");
53        t2 = req2.substring(0, indexComma);
54        t3 = req2.substring(indexComma + 1, indexSpace);
55        thresholdLuminosity = t1.toInt();
56        thresholdTemperature = t2.toInt();
57        thresholdDistance = t3.toInt();
58        aux = "Threshold Values Sucessfully Modified";
59      }
60      else {
61        if (dist == 1) {
62          String auxDist = "/distance/";
63          int sizeDist = auxDist.length();
64          req1 = req.substring(req.indexOf("/distance/")
65      + sizeDist);
```

```
66          distanceValue = req1.toInt();
67          aux = "Distance␣Updated";
68        }
69        else {
70          if (val_infor == 1) {
71            aux = String(temp, 2) + ";";
72            aux = aux + String(luminosityValue) + ";";
73            aux = aux + String(distanceValue) + ";";
74            aux = aux + String(control) + ";";
75            aux = aux + String(thresholdLuminosity) + ";";
76            aux = aux + String(thresholdTemperature) + ";";
77            aux = aux + String(thresholdDistance);
78          }
79          else {
80            if (control == 0) {
81              aux = "Manual␣Control␣Activated";
82            }
83            else {
84              if (control == 1) {
85                aux = "Automatic␣Control␣Activated";
86              }
87            }
88          }
89        }
90      }
91
92      wifiClient.flush();
93
94      String httpResponse;
95      String httpHeader;
96      // HTTP Header
97      httpHeader = "HTTP/1.1␣200␣OK\r\nContent-Type:␣";
98    httpHeader += "text/html;␣charset=UTF-8\r\n";
99      httpHeader += "Content-Length:␣";
100     httpHeader += aux.length();
101     httpHeader += "\r\n";
102     httpHeader += "Connection:␣close\r\n\r\n";
103     httpResponse = httpHeader  + aux + "␣";
104     // Send the response to the client
105     wifiClient.print(httpResponse);
```

```
106     delay(1);
107     Serial.println("Client disconnected");
108     if (val_infor == 1 || dist == 1) {
109       val_infor = 0;
110       thresh = 0;
111       dist = 0;
112       return;
113     }
114     thresh = 0;
115     dist = 0;
116     delay(500);
117     sendToCloud();
118     os_timer_disarm(&mTimer);
119     os_timer_arm(&mTimer, TIMECLOUD, true);
120   }
121 }
```

Listagem I.3: Obtenção dos valores dos sensores de luminosidade e temperatura

```
1
2  void readAnalogSensor() {
3    yield();
4    digitalWrite(MuxSelectStatePin, LOW);
5    delay(1);
6    int sensorValue = 0;
7    for (int i = 0; i < 10; i++) {
8      sensorValue  = sensorValue + analogRead(ADC_PIN);
9      delay(5);
10     yield();
11   }
12   temp = sensorValue * 1.1165 / 1023.0 * 10.0;
13   Serial.println(temp);
14   yield();
15   delay(100);
16   yield();
17   digitalWrite(MuxSelectStatePin, HIGH);
18   delay(1);
19   sensorValue = 0;
20   for (int i = 0; i < 10; i++) {
21     sensorValue  = sensorValue + analogRead(ADC_PIN);
22     delay(5);
```

```
  }
  if (sensorValue < 256)
    luminosityValue = 0;
  else {
    if (sensorValue < 512)
      luminosityValue = 1;
    else {
      if (sensorValue < 768)
        luminosityValue = 2;
      else {
        luminosityValue = 3;
      }
    }
  }
  yield();
  Serial.println(luminosityValue);
  delay(100);
}
```

Listagem I.4: Código Principal do Processador Central

```
void loop() {

  if (_timeout) { //Check If timer has been triggered
    sendToCloud();
    _timeout = false;
  }
  if (_timeoutADC) { //Check If timer has been triggered
    readAnalogSensor();
    _timeoutADC = false;
  }
  // Check if a client has connected
  wifiClient = wifiServer.available();
  if (wifiClient) {
    receivedInfo();
  }

  //readAnalogSensor();
  yield();

  if (control == 1) {
```

```
21      if (luminosityValue > thresholdLuminosity) {
22        Serial.println("SOBE");
23        digitalWrite(relayPin2, LOW);
24        delay(10);
25        digitalWrite(relayPin1, HIGH);
26        delay(10);
27      }
28      else {
29        if (luminosityValue < thresholdLuminosity) {
30          Serial.println("DESCE");
31          digitalWrite(relayPin1, LOW);
32          delay(10);
33          digitalWrite(relayPin2, HIGH);
34          delay(10);
35        }
36        else {
37          if (temp > thresholdTemperature + 5) {
38            Serial.println("DESCE");
39            digitalWrite(relayPin1, LOW);
40            delay(10);
41            digitalWrite(relayPin2, HIGH);
42            delay(10);
43          }
44          else {
45            if (temp < thresholdTemperature - 5) {
46              Serial.println("SOBE");
47              digitalWrite(relayPin2, LOW);
48              delay(10);
49              digitalWrite(relayPin1, HIGH);
50              delay(10);
51            }
52            else {
53              Serial.println("PARA");
54              digitalWrite(relayPin1, LOW);
55              delay(10);
56              digitalWrite(relayPin2, LOW);
57              delay(10);
58            }
59          }
60        }
```

```
61      }
62    }
63    else {
64      if ( distanceValue < thresholdDistance - 5) {
65        Serial.println("DESCE");
66        digitalWrite(relayPin1, LOW);
67        delay(10);
68        digitalWrite(relayPin2, HIGH);
69        delay(10);
70      }
71      else {
72        if ( distanceValue > thresholdDistance + 5) {
73          Serial.println("SOBE");
74          digitalWrite(relayPin2, LOW);
75          delay(10);
76          digitalWrite(relayPin1, HIGH);
77          delay(10);
78        }
79        else {
80          Serial.println("PARA");
81          digitalWrite(relayPin1, LOW);
82          delay(10);
83          digitalWrite(relayPin2, LOW);
84          delay(10);
85
86        }
87      }
88    }
89  }
```

ANEXO II

Código Associado ao Processador Exterior

Listagem II.1: Código Associado ao Processador Exterior

```
#include <ESP8266WiFi.h>

#define irLedPin 4          // IR Led on this pin
#define irSensorPin 5        // IR sensor on this pin

const char* ssid = "ZON-89F0";
const char* password = "07208adfac73";

const char* host = "192.168.1.102";

unsigned long timeFinal = 0;
unsigned long timeStart = 0;
unsigned long distance = 0;
unsigned long lastDistance = -50;
int change = 0;


void sendToServer() {

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
```

```
25    const int httpPort = 80;
26    if (!client.connect(host, httpPort)) {
27      Serial.println("connection␣failed");
28      return;
29    }
30
31    // We now create a URI for the request
32    String url = "/distance/";
33    url += String(distance);
34
35    Serial.print("Requesting␣URL:␣");
36    Serial.println(url);
37
38    // This will send the request to the server
39    client.print(String("GET␣") + url + "␣HTTP/1.1\r\n" +
40                 "Host:␣" + host + "\r\n" +
41                 "Connection:␣close\r\n\r\n");
42    unsigned long timeout = millis();
43    while (client.available() == 0) {
44      if (millis() - timeout > 10000) {
45        Serial.println(">>>␣Client␣Timeout␣!");
46        client.stop();
47        return;
48      }
49    }
50
51    // Read all the lines of the reply from server and print them
        to Serial
52    while (client.available()) {
53      String line = client.readStringUntil('\r');
54      Serial.print(line);
55    }
56
57    Serial.println();
58    Serial.println("closing␣connection");
59  }
60
61  void setup() {
62    Serial.begin(115200);
63    delay(10);
```

```
64
65    // We start by connecting to a WiFi network
66
67    Serial.println();
68    Serial.println();
69    Serial.print("Connecting to ");
70    Serial.println(ssid);
71
72    /* Explicitly set the ESP8266 to be a WiFi-client, otherwise,
          it by default,
73       would try to act as both a client and an access-point and
             could cause
74       network-issues with your other WiFi-devices on your WiFi-
             network. */
75    WiFi.mode(WIFI_STA);
76    WiFi.begin(ssid, password);
77
78    while (WiFi.status() != WL_CONNECTED) {
79      delay(500);
80      Serial.print(".");
81    }
82
83    Serial.println("");
84    Serial.println("WiFi connected");
85    Serial.println("IP address: ");
86    Serial.println(WiFi.localIP());
87
88
89    attachInterrupt(digitalPinToInterrupt(irSensorPin), SensorHigh
          , HIGH );
90    pinMode(irSensorPin, INPUT);
91    pinMode(irLedPin, OUTPUT);
92  }
93
94  void loop() {
95    digitalWrite(irLedPin, HIGH);
96    timeStart = micros();
97    while (micros() - timeStart < 200000) {
98    }
99    digitalWrite(irLedPin, LOW);
```

```
100   delay(20);
101   if (change) {
102     lastDistance = distance;
103     distance = ((timeFinal - timeStart) / 200000.0) * 100.0;
104     if ((distance - lastDistance < -2 ) && (distance -
          lastDistance > 2))
105       sendToServer();
106     yield();
107     change = 0;
108     delay(200);
109   }
110 }
111
112 void SensorHigh()
113 {
114   timeFinal = micros();
115   change = 1;
116
117 }
```

ANEXO III

Código Associado à Aplicação *Android*

Listagem III.1: Código representativo de um envio de uma mensagem para o processador central (Utilizado pelas actividades *Update Thresholds*, *Control Mode* e *Device Info*)

```
1     /**
2      * Description: Send an HTTP Get request to a specified ip
          address and port.
3      *
4      * @return The ip address' reply text, or an ERROR message
          is it fails to receive one
5      */
6     public String sendRequest(String urlString) throws
         IOException {
7           String serverResponse = "ERROR";
8           System.out.println(urlString);
9           try {
10              HttpURLConnection urlConnection = (
                   HttpURLConnection) (new URL(urlString).
                   openConnection());
11              if (urlConnection.getResponseCode() ==
                   HttpURLConnection.HTTP_OK) {
12
13                  InputStream inputStream = null;
14                  inputStream = urlConnection.getInputStream()
                       ;
15                  BufferedReader bufferedReader =
```

```
16                              new BufferedReader(new
                                    InputStreamReader(inputStream));
17                      serverResponse = bufferedReader.readLine();
18
19                      inputStream.close();
20                  }
21              }
22              catch (MalformedURLException e) {
23                  e.printStackTrace();
24                  serverResponse = e.getMessage();
25              } catch (IOException e) {
26                  e.printStackTrace();
27                  serverResponse = e.getMessage();
28              }
29              // return the server's reply/response text
30              return serverResponse;
31          }
32
33      /**
34       * An AsyncTask is needed to execute HTTP requests in the
             background so that they do not
35       * block the user interface.
36       */
37      private class HttpRequestAsyncTask extends AsyncTask<Void,
            Void, Void> {
38
39          // declare variables needed
40          private String requestReply, ipAddress, portNumber;
41          private Context context;
42          private AlertDialog alertDialog;
43          private String parameter;
44          private String parameterValue;
45
46          private boolean erro = false;
47
48
49          /**
50           * Description: The asyncTask class constructor. Assigns
                 the values used in its other methods.
51           *
```

```
52          * @param context    the application context, needed to
                create the dialog
53          * @param ipAddress  the ip address to send the request
                to
54          * @param portNumber the port number of the ip address
55          */
56         public HttpRequestAsyncTask(Context context, String
               ipAddress, String portNumber) {
57             this.context = context;
58
59             alertDialog = new AlertDialog.Builder(this.context)
60                     .setTitle("HTTP␣Response␣From␣IP␣Address:")
61                     .setCancelable(true)
62                     .create();
63
64             this.ipAddress = ipAddress;
65             this.portNumber = portNumber;
66
67         }
68
69         /**
70          * Name: doInBackground
71          * Description: Sends the request to the ip address
72          *
73          * @param voids
74          * @return
75          */
76         @Override
77         protected Void doInBackground(Void... voids) {
78             alertDialog.setMessage("Data␣sent,␣waiting␣for␣reply
                   ␣from␣server...");
79             if (!alertDialog.isShowing()) {
80                 alertDialog.show();
81             }
82            try {
83                 requestReply = sendRequest("http://" + ipAddress
                       + ":" +
```

```
84                          portNumber+"/" + threshold + "/" +
                               luminosityThreshold + ";" +
                               temperatureThreshold + ";" +
                               distanceValue);
85                  System.out.println("http://" + ipAddress + ":" +
86                          portNumber+"/" + threshold + "/" +
                               luminosityThreshold + ";" +
                               temperatureThreshold + ";" +
                               distanceValue);
87              } catch (IOException e) {
88                  erro = true;
89                  e.printStackTrace();
90              }
91
92
93              return null;
94      }
95
96
97      }
98
99      public class InputFilterMinMax implements InputFilter {
100
101         private int min, max;
102
103         public InputFilterMinMax(int min, int max) {
104             this.min = min;
105             this.max = max;
106         }
107
108         public InputFilterMinMax(String min, String max) {
109             this.min = Integer.parseInt(min);
110             this.max = Integer.parseInt(max);
111         }
112
113         @Override
114         public CharSequence filter(CharSequence source, int
               start, int end, Spanned dest, int dstart, int dend) {
115             try {
```

```
116                 int input = Integer.parseInt(dest.toString() +
                        source.toString());
117                 if (isInRange(min, max, input))
118                     return null;
119             } catch (NumberFormatException nfe) { }
120             return "";
121         }
122 
123         private boolean isInRange(int a, int b, int c) {
124             return b > a ? c >= a && c <= b : c >= b && c <= a;
125         }
126     }
127 }
```

Listagem III.2: Código para tratamento de recepção de mensagem aquando da obtenção de valores diretamente do Processador

```
1         /**
2          * Name: onPostExecute
3          * Description: This function is executed after the HTTP
               request returns from the ip address.
4          * The function sets the dialog's message with the reply
               text from the server and display the dialog
5          * if it's not displayed already (in case it was closed
              by accident);
6          */
7         @Override
8         protected void onPostExecute(Void aVoid) {
9             if(erro) {
10                alertDialog.setTitle("ERROR");
11                alertDialog.setMessage("No address associated
                      with hostname!");
12                if(!alertDialog.isShowing())
13                {
14                    alertDialog.show(); // show dialog
15                }
16            }
17            else {
18                try{
19                    alertDialog.setMessage(requestReply);
20                    System.out.println(requestReply);
```

```
21                      String[] a = requestReply.split(";");
22                      if(a.length == 7) {
23                          for (int i = 0; i < a.length; i++) {
24                              if (i == 0)
25                                  temperature.setText(a[i]);
26                              if (i == 1)
27                                  luminosity.setText(a[i]);
28                              if (i == 2)
29                                  distance.setText(a[i]);
30                              if (i == 3)
31                                  controlMode.setText(a[i]);
32                              if (i == 4)
33                                  lumThresh.setText(a[i]);
34                              if (i == 5)
35                                  tempThresh.setText(a[i]);
36                              if (i == 6)
37                                  distThresh.setText(a[i]);
38                              if(alertDialog.isShowing())
39                              {
40                                  alertDialog.hide(); // hide
                                        dialog
41                              }
42                          }
43                      }
44                      else{
45                          alertDialog.setMessage(requestReply);
46                          if(!alertDialog.isShowing())
47                          {
48                              alertDialog.show(); // show dialog
49                          }
50
51                      }
52                  }
53                  catch (IndexOutOfBoundsException e){
54                      alertDialog.setMessage(requestReply);
55                      System.out.println(requestReply);
56                      if(!alertDialog.isShowing())
57                      {
58                          alertDialog.show(); // show dialog
59                      }
```

```
60                  }
61
62              }
63
64              erro = false;
65          }
```