Instituto Superior Técnico

Sistemas de Controlo Distribuído em Tempo-Real

MEEC

2019/2020 - 5º Ano, 1º Semestre

Real-Time Cooperative Decentralized Control of Illumination Systems

Grupo 20

Eduardo Pereira	79640
David Ribeiro	84027

Docentes:
Alexandre José Malheiro Bernardino
João Pedro Castilho Pereira Santos Gomes
Ricardo Adriano Ribeiro

10 de Janeiro de 2020

Conteúdo

Resumo

Este relatório mostra uma proposta para a minimização do consumo de energia da simulação de um escritório baseada num sistema de controlo distribuído. Cada posto de trabalho é composto por um micro-controlador, um LED e uma foto-resistência. O algoritmo *consensus* é utilizado no cálculo dos *duty cycles* a serem usados nos LED para um consumo mínimo de energia mas satisfazendo os valores de luminosidade em cada posto de trabalho. Foi também desenvolvido um controlador local para que o sistema consiga reagir a perturbações exteriores. Os arduinos comunicam entre si através do uso do *Can bus* e com uma aplicação desenvolvida em C++, permite ao utilizador enviar comandos para o sistema através de comunicação série.

Palavras-chave— controlo distribuído, consensus, tempo-real, eficiência energética, arduino

1 Introdução

Devido ao elevado consumo energético em escritórios para efeitos de iluminação, este projecto tem como objectivo apresentar uma solução mais viável tanto economicamente, como financeiramente, sem afectar a qualidade da iluminação disponível.

Neste trabalho o objectivo é desenvolver um sistema de iluminarias, com um sensor de luminância e o um sensor de presença associados a cada, que, respeitando os valores mínimos impostos pela norma EN 12464-1, optimize a energia consumida pelo sistema. Para tal é utilizado um sistema de controlo descentralizado, onde cada luminária terá uma referência de *duty cicle* independente, o qual influencia proporcionalmente a potência consumida. Estas referencias são calculadas a partir do algoritmo de controlo descentralizado *consensus*. Para garantir que a luminância é mantida ao longo do tempo, mesmo na presença de perturbações e/ou iluminação externa, luz do sol a variar ao longo do dia através de uma janela, por exemplo, é utilizado um controlador proporcional integral local em cada ponto, de forma a que a o valor de luminância seja o mais constante possível. Para a leitura do valor da luminância e respectiva actuação e controlo local será utilizado um arduino em cada ponto. Os arduinos comunicam entre si por *CAN bus*, e a um computador através de comunicação série.

2 Arquitetura do sistema

2.1 Circuito

Para simular as luminárias presentes num escritório, usa-se o circuito apresentado na Figura 1. Para se poder monitorizar a quantidade usa-se o circuito recetor também na Figura 1, através do uso de um LDR, resistência dependente de luz. O valor desta resistência em função da luz aplicada na resistência é dada por

$$LDR(x) = 10^{m*\log_{10} x+b}, \tag{1}$$

em que m e b representam respetivamente o declive da característica Iluminância/Resistência, que na escala logarítmica é linear, e a interseção do declive no eixo da resistência. Como o processo de fabricação dos LDR não é muito rigoroso, estes dois valores variam dentro de uma certa gama de valores fornecidos na folha de especificações do LDR. Por isso, os valores de m e b foram determinados de forma a se obter uma uma relação entre *duty cycle* e iluminância o mais linear possível, e que as leituras dos dois sensores fossem semelhantes para a mesma de luz.

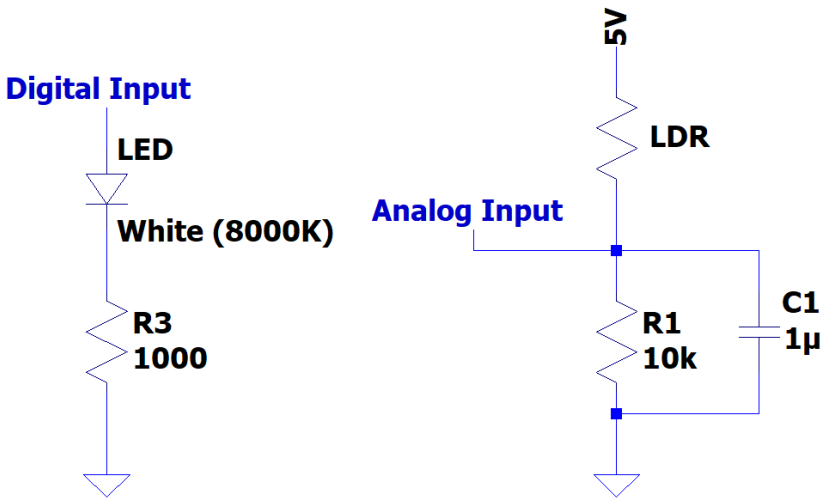

Figura 1: Esquemático do circuito emissor, à esquerda, e o circuito recetor à direita

Como a variação de iluminância faz variar o valor da foto-resistência, é preciso utilizar um circuito para que o arduino consiga medir esta variação de forma a obter depois a iluminância. Com o circuito recetor, verifica-se que a tensão medida na entrada analógica pode ser obtida utilizando um divisor de tensão é dada por

$$v = V_{cc} - i * LDR(x), \tag{2}$$

em que x é a iluminância em lux que o LDR capta, V_{cc} é novamente a tensão de alimentação do circuito e i é a corrente que passa no LDR, que por sua vez é dada pela soma da corrente na resistência R_1 e no condensador C_1, ou seja,

$$i = \frac{v}{R_1} + C_1 * \frac{\partial v}{\partial t}. \tag{3}$$

Ao substituir 3 em 2, resulta em

$$\dot{v} * \tau(x) = -v + V_{cc} * \frac{R_1}{R_1 + LDR(x)}, \tag{4}$$

em que $\tau(x)$ é dado por

$$\tau(x) = \frac{R_1 * LDR(x)}{R_1 + LDR(x)} * C_1. \tag{5}$$

Resolver a equação diferencial em 4, resulta em

$$v(t) = v_f - (v_f - v_i) \exp - \frac{t - t_i}{\tau(x_f)}, \tag{6}$$

em que v_f e v_i são os valores finais e iniciais respetivamente de v, t_i é o instante de tempo em que o valor de x muda e x_f é o valor final de x.

2.2 Caracterização do LDR

Para se saber a relação entre a iluminância e o *duty cycle* aplicado é necessário determinar os parâmetros m *e* b da equação (1). Para tal, foram medidos varios valores da tensão à saida do circuito recetor para diferentes valores de *duty cycle*. A partir da equação (divisor de tensão da Figura 1)

$$LDR = (\frac{V_{cc}}{v} - 1) * R_1 \tag{7}$$

é possível obter a resistência no LDR, e assim obter a iluminância a partir de

$$x = (\frac{LDR}{10^b})^{\frac{1}{m}}. \tag{8}$$

Como b é o *offset* entre a resistência e a iluminância seria necessário saber a que iluminância esta sujeito o LDR para se poder determinar. Como tal, este parâmetro foi apenas determinado de forma a que o conjunto tivesse valores similares para a mesma condição de luz. Para determinar m, com os pontos recolhidos, determinou-se a correlação

linear da iluminância x com o *PWM* (Pulse Width Modulation, é utilizado no lugar do *duty cicle*, é dado por numa palavra de um byte), para diversos valores de m (entre o mínimo e máximo possível dados pela *datasheet*), escolhendo-se no final o que apresentar maior correlação. Na Figura 2 pode-se observar um gráfico com a correlação obtida em função de m.

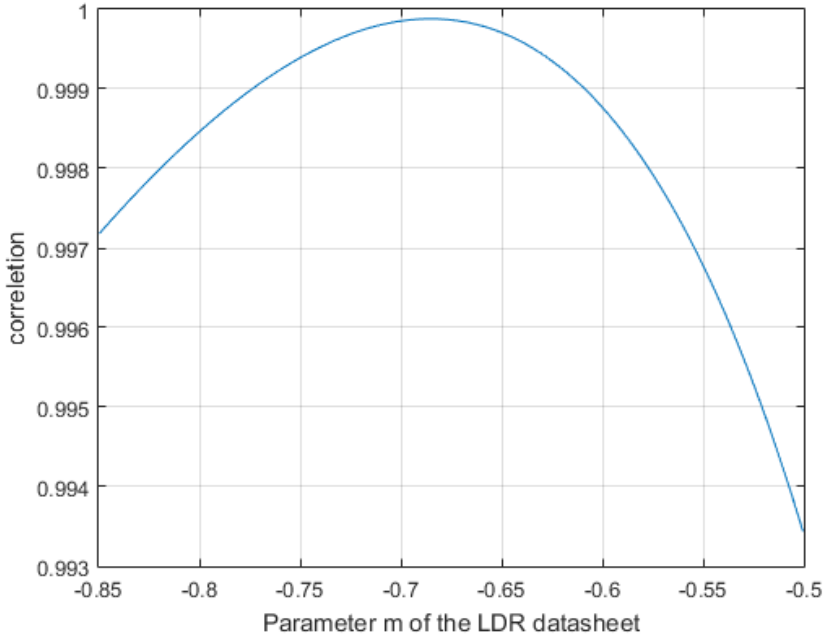

Figura 2: Determinação do m que melhor correlaciona linearmente a iluminância com o *PWM*.

2.3 Determinação do tempo de resposta

O sistema em causa é de primeira ordem, e pode ser descrito por

$$G(x) = \frac{K_0(x)}{1 + s\tau(x)}. \tag{9}$$

Para se determinar o tempo de resposta do circuito $\tau(x)$, em resposta a uma alteração ao *PWM* de referência, aplicaram-se diferentes *steps* (Figura 3 e traçou-se uma reta do tempo de resposta em função dos *PWM* aplicados.

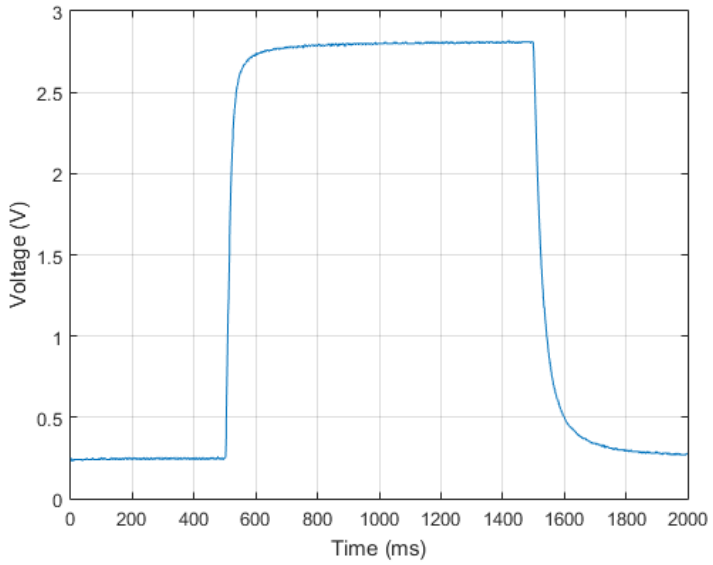

Figura 3: Na figura pode-se observar um *step* crescente aos 500 ms e um negativo aos 1500 ms.

Na Figura 4 pode-se observar os tempos de resposta em função do PWM. À esquerda tem-se *step* de 0 para o valor presente no eixo horizontal, e à direita o inverso.

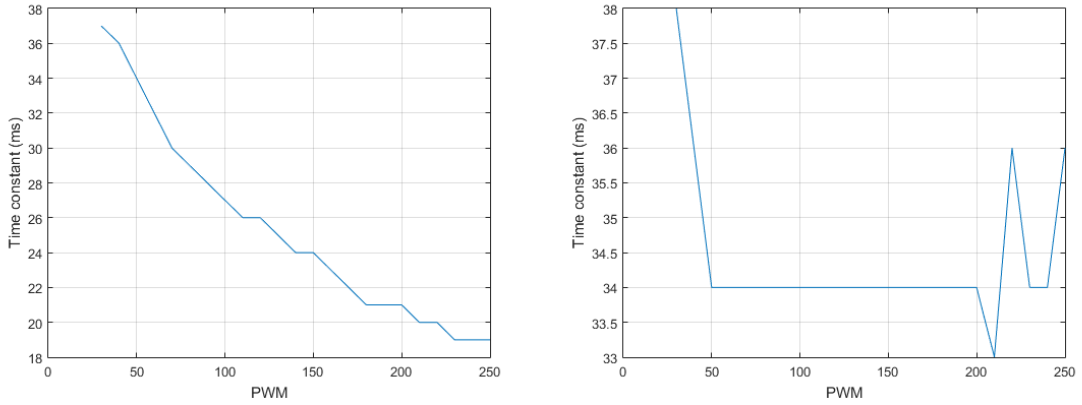

Figura 4: Tempo de resposta em função do PWM.

Assim assumiu-se que quando o *step* é crescente a constante de tempo varia linearmente com o PWM, e quando é decrescente se mantém constante.

3 Inicialização do sistema

Quando se inicializa o sistema cada elemento envia uma mensagem para reiniciar o sistema, de forma a que o sistema seja imediatamente calibrado. O sistema desenvolvido sabe o número de elementos que serão inseridos na rede, mas poderia não saber, para tal bastaria adicionar um vector com o numero máximo de elementos que o sistema aceita, e quando se faz a calibração guardar a informação dos elementos que de facto estão na rede, e os que não estão. Esta solução obrigaria a alocar memória para o numero máximo de elementos que o sistema poderia receber, mas como se verá na secção das comunicações, não seria um problema visto que o numero máximo de elementos permitidos é baixo.

Após a inicialização do sistema é necessário determinar a relação entre o PWM e a iluminância. A relação entre ambas é linear, mas a luz de um LED vai influenciar a leitura dos LDRs vizinhos, portanto é necessário também determinar os coeficientes de acoplamento. A iluminância do sistema é dada por

$$\begin{bmatrix} L_1 \\ ... \\ L_N \end{bmatrix} = \begin{bmatrix} k_{11} & ... & k_{1N} \\ ... & & ... \\ k_{N1} & ... & k_{NN} \end{bmatrix} \begin{bmatrix} d_1 \\ ... \\ d_N \end{bmatrix} + \begin{bmatrix} o_1 \\ ... \\ o_N \end{bmatrix}, \tag{10}$$

onde L_i, d_i, o_i são a iluminância, PWM e iluminância externa ao sistema na secretária i, e k_{ij} o ganho de iluminância em i em função do PWM do LED j.

Assim para se determinar a relação entre PWM e a iluminância é necessário determinar os coeficientes k_{ij} e as iluminâncias externas e_i. Para tal, cada elemento lê o valor de iluminância com todos os LEDs desligados para obter a sua iluminância externa o_i, e posteriormente, à vez cada elemento liga o seu LED com PWM máximo, todos os elementos fazem a leitura da iluminância, e o obtém o valor do coeficiente por

$$k_{ij} = \frac{L_i - o_i}{PWM_{max}}. \tag{11}$$

onde j é o elemento com o LED ligado. A organização deste processo pode ser observado no fluxograma da Figura 5.

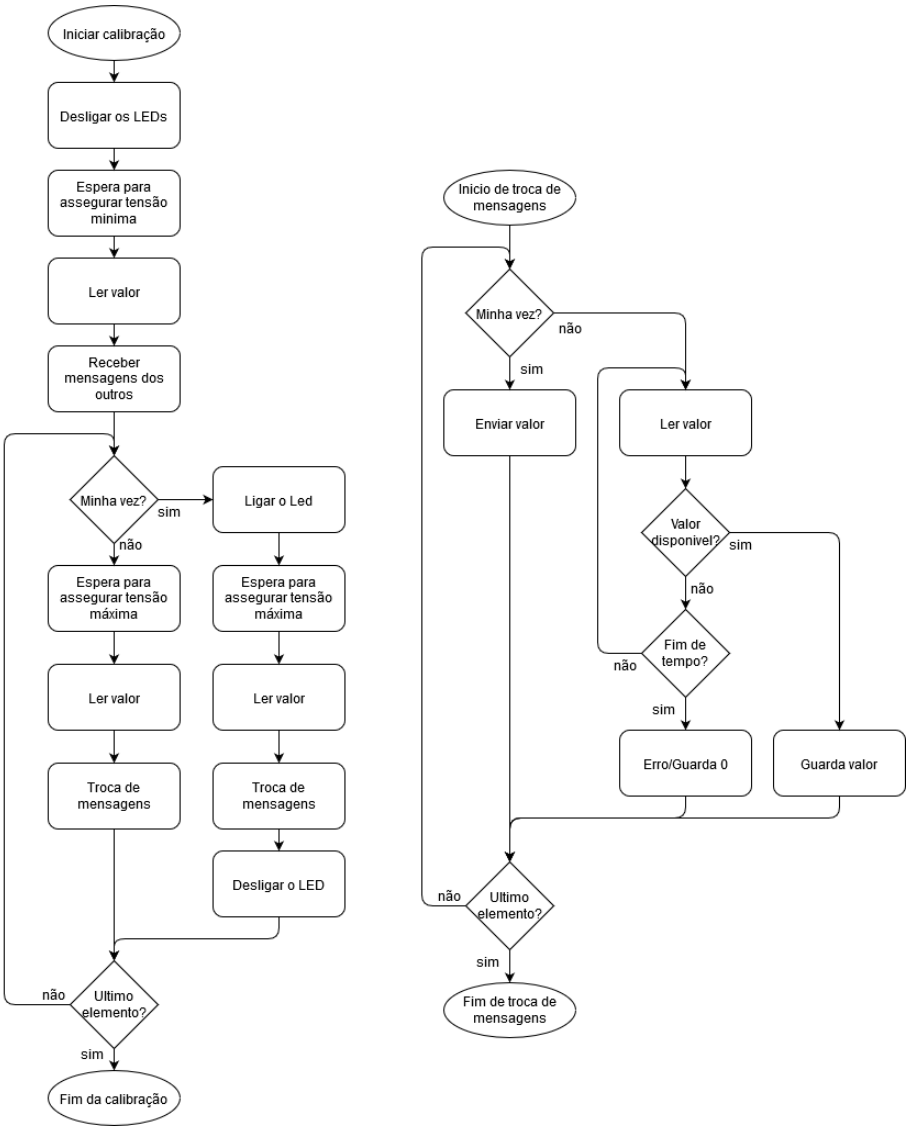

Figura 5: Fluxograma do processo de calibração.

No fluxograma da Figura 5 pode-se notar que os coeficientes são passados para todos os elementos da rede. Este processo não é necessário ao bom funcionamento do programa, apenas é feito para o utilizador poder ter noção dos valores dos coeficientes de todo o sistema, e não ter acesso a apenas um nó. Para remover esse procedimento será necessário adicionar um tempo de espera para garantir que todos os elementos leram os valores, ou então os elementos que não ligaram o LED enviam um mensagem de verificação em como a leitura já foi efetuada. Todas estas abordagens são síncronas.

4 Controlo

4.1 Local

Cada elemento tem um controlador local para manter o valor da iluminância desejada, e ser capaz de atuar perante perturbações externas. Na seguinte Figura pode-se observar o diagrama de blocos do controlador local.

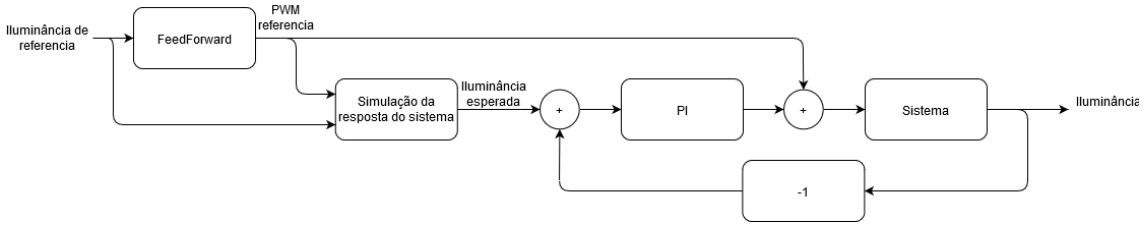

Figura 6: Esquema de controlo local de cada elemento.

O controlador local procura que a variável de saída siga uma dada referencia de entrada. Para tal o controlador *feedforward* relaciona a variável de saída com a de atuação do sistema. Assim o controlador responde à iluminância de referencia da entrada, determinando um valor de PWM que gere esse valor de iluminância à saída. O controlador *feedforward* contudo não tem qualquer capacidade para lidar com perturbações, colocando sempre o mesmo valor à saída para uma dada referencia. Este bloco é totalmente dependente da calibração que foi feita, podendo assim apresentar erro estático em caso de diferenças em relação aos valores determinados inicialmente.

Para resolver os problemas do controlador *feedforward* adiciona-se um controlador de *feedback*. Este controlador compara o valor de iluminância à saída com o valor esperado, e utiliza esse valor de erro para determinar como deve atuar sobre o sistema. Para este efeito foi escolhido um controlador proporcional integral (PI). Como o sistema é discreto é necessário utilizar uma transformação, neste caso Tustin, pois não tem atrasos e é a melhor a lidar com ruído. Foi adicionada uma janela de *anti-windup*, para prevenir que o erro acumulado do integrador fosse demasiado alto, o que pode acontecer devido a uma variação brusca na referencia, ou à referencia estar fora do alcance do sistema. Para diminuir os erros de quantificação (conversor A/D tem uma resolução de 1024 bits para um alcance 5V) poderia-se adicionar um filtro passa baixo, fazendo a média das ultimas amostras.

Por fim o simulador de resposta do sistema utiliza a equação (6) para determinar o valor de iluminância esperada ao longo do tempo. Este bloco permite que o valor do erro calculado no controlador de *feedback* seja mais perto do real, visto que o sistema não tem uma resposta imediata à alteração de referência.

4.2 Geral

O algoritmo *consensus* é utilizado para minimizar o consumo de energia, satisfazendo os valores de luminosidade pedidos em cada luminária simulada. Trata-se de um problema de minimização e pode ser descrito por

$$\begin{aligned} \min_{d} \quad & c^T d \\ \text{s.t.} \quad & d \in C \end{aligned}, \tag{12}$$

em que

$$d = [d_1 \ \cdots \ d_N]^T \tag{13}$$

e

$$c = [c_1 \ \cdots \ c_N]^T \tag{14}$$

são respetivamente os *duty-cycles* associados a cada LED e os custos associados a cada luminária e

$$C : \{d : \forall i, \ 0 \leq d_i \leq 100 \ e \ k_i^T d \geq L_i - o_i\} \tag{15}$$

representa a *feasible region*, L_i, o_i e k_i são respetivamente o limite inferior de luminosidade do nó i, a luminosidade externa medida no nó i e

$$k_i = [k_{i1} \ \cdots \ k_{ii} \ \cdots \ k_{iN}]^T \tag{16}$$

que representa a influência do nó i nos outros nós. O algoritmo *consensus* permite a separação a função do problema de minimização, ou seja, cada nó precisa apenas dos seus próprios valores de L, o e k para calcular a solução ótima. Para obter esta solução, o algoritmo ADMM (*alternated direction method of multipliers*, que usa o método *augmented Lagrangian*, é aplicado em cada iteração para que o nó i possa obter os valores de *duty-cycle*. A minimização distribuída usando o *augmented Lagrangian* baseia-se nos seguintes cálculos em cada nó:

1. $d_i(t+1) = argmin_{d_i \in C_i}\{c_i^T d_i \ + y_i^T(t)(d_i - \overline{d_i}(t)) + \frac{\rho}{2}||d_i - \overline{d_i}(t)||_2^2\}$
2. $\overline{d_i}(t+1) = \frac{1}{N}\sum_{j=1}^{N} d_j(t+1)$
3. $y_i(t+1) = y_i(t) + \rho(d_i(t+1)) - \overline{d_i}(t+1))$

em que t é o número da iteração, $\overline{d_i}$ é uma cópia da solução média de todos os nós e ρ é um parâmetro do método. Baseado em (2), a iteração $d_i(t+1)$ é dada por

$$d_i(t+1) = argmin_{d \in C}\{\frac{1}{2}\rho d_i^T d_i - d_i^T z_i(t)\} \tag{17}$$

com $z_i(t) = \rho\overline{d_i}(t) - c_i - y_i(t)$

Este algoritmo permite calcular a solução para este problema de otimização com restrições de forma descentralizada. Esta descentralização permite uma poupança em termos de memória nos arduinos, já que cada arduino precisa apenas de calcular a sua solução.

5 Comunicações

O sistema é composto por uma rede de arduinos ligados entre si por *Can bus*, e um dos quais, *Hub*, está ligado a um computador por porta serie. O utilizador poderá monitorizar e dar ordens ao sistema a partir de uma aplicação de computar, a qual recebe e envia toda a informação para o sistema através do *Hub*. Cada arduino guarda as suas informações, exceto os *buffers* com a informação da iluminância e *PWM* ao longo do ultimo minuto, as quais são enviadas constantemente para o computador.

5.1 Comunicação entre arduinos

A comunicação entre os vários elementos da rede é feita através de *Can bus* (Controller Area Network). A comunicação é descentralizada, e todos elementos se encontram ligados à rede, sendo a transmissão de mensagens feita através de *Broadcast*, cabendo a cada elemento da rede decidir através do identificador da mensagem se a lê ou não. Não existe limitação no numero de elementos ligados à rede, sendo possível adicionar ou remover elementos facilmente à rede. O controlo de erros é feito por CRC (Cyclic Redundancy Check), sendo a probabilidade de erro não detatado inferior a $4.7 * 10^{-11}$.

As mensagens são limitas a tem um tamanho máximo de 134 bits em modo standard, dos quais 5 bytes são para o identificador e 8 bytes para o conteúdo. Assim foi definido o seguinte protocolo de comunicação

Mensagem	Descrição	ID de destino	memória (byte)
r	Informação para reiniciar o sistema	BROADCAST	1
q	Informação para correr o consensus	BROADCAST	1
g X	Pedir o parametro X ao arduino ID	ID	3
Xjvalue	Parametro X e o respetivo valor do elemento j	BROADCAST	X - 1, ID - 1, value - 1 a 6
Ijvalue	Iluminância do elemento j	ID	I - 1, ID - 1, value - 1 a 6
djvalue	PWM do elemento ID	ID	d - 1, ID - 1, value - 1 a 6
sXvalue	Alterar X para o valor recebido	ID	s - 1, X - 1, value - 1 a 6
Hj	Identificador do HUB	BROADCAST	H - 1, ID - 1 a 7

Onde cada elemento apenas aceita as mensagens com o seu identificador (*buffer* 1 de receção) ou as de *Broadcast* (ambos os *buffers* de receção). A caractere *X* representa um de vários parâmetros disponíveis para cada elemento, desde ocupação ('o'), iluminância de referência ('r'), energia gasta ('e'), ou erro de visibilidade ('v'). O caractere j representa o identificador do emissor da mensagem. Isto é necessário pois o *HUB* com o identificador da

mensagem apenas sabe identificar que a mensagem é para ele, não sabendo a sua origem. Essa identificação é feita através de um caractere, o que aumenta o numero de endereços disponíveis de 10 para mais de 200. Por fim o *value* tem disponíveis 6 bytes para poder escrever o seu valor.

Cada mensagem em modo standard tem no máximo 134 bits, sendo que a velocidade de transmissão é de 1 Mbps consegues-se obter assim um numero mínimo aproximado de 7463 mensagens por segundo. Sabendo que o sistema tem de correr a 100 Hz, tem-se que o numero máximo de mensagens por período é de 74. Como em cada período cada elemento tem de enviar os seus valores de iluminância e *PWM* ao *HUB*, e pode ser necessário que cada elemento envie uma mensagem numa dada altura a pedido $g\ X\ T$, tendo assim que cada elemento (exceto o *HUB*) tem de poder enviar pelo menos 3 mensagens por período. Assim tem-se que o numero máximo de elementos é de 24 elementos. Na prática será ainda menor pois é necessário tempo para o controlo local, calculo de parâmetros, bem como enviar e receber mensagens pela porta serie na comunicação com a aplicação.

5.2 Comunicação entre Computador e sistema

A comunicação do computador com o sistema de arduinos é baseado na biblioteca Boost Asio, que permite implementar funções de I/O assincronamente, através de uma aplicação desenvolvida em C++. A função `async_write` permite a escrita de uma string que contém o comando desejado num *buffer*, que por sua vez é enviado para um dos arduinos para que o comando possa ser processado. A função `async_read_until` permite a leitura da string que o arduino envia, que é escrita noutro *buffer* diferente e esta string é enviada para a consola. Para facilitar as comunicações entre o computador e o sistema, os cálculos necessários para certos comandos são todos feitos nos arduinos e assim só é preciso enviar a string com a resposta desejada para cada comando, a custo de um uso da memória quase total dos arduinos. É que os *buffers* de dados são alocados na aplicação devido a falta de memória nos arduinos. A comunicação é realizada por porta série, e em caso da aplicação não conseguir abrir a porta, retorna uma mensagem de erro para a consola. Foram também implementadas condições de *break* para alguns comandos, para garantir que não chegam strings não desejadas à consola da aplicação. A utilização de funções assíncronas permite o envio de comandos para os arduinos e leitura de mensagens sem que um comprometa o outro, sendo feitas assincronamente. Por outro lado, isto implica que sejam protegidas as estruturas de dados bem como a leitura e escrita na porta série, para não haver tentativas de acesso em simultâneo, levando a erros de leitura ou escrita.

No desenvolvimento da aplicação foram usadas *threads* para definir a tarefa de escrita e de leitura, para permitir que as funções fossem assíncronas, podendo enviar ou receber mensagens do arduino sem uma ordem especifica. É usado o máximo *baud rate* de 250000 para que as comunicações possam ser o mais rápido possível.

6 Resultados

No controlo local foram observadas as resposta do sistema para os diferentes componentes do mesmo. Todos os testes foram feitos utilizando duas referencias fixas, 10 lux para o estado *low* e 40 lux para o estado *high*.

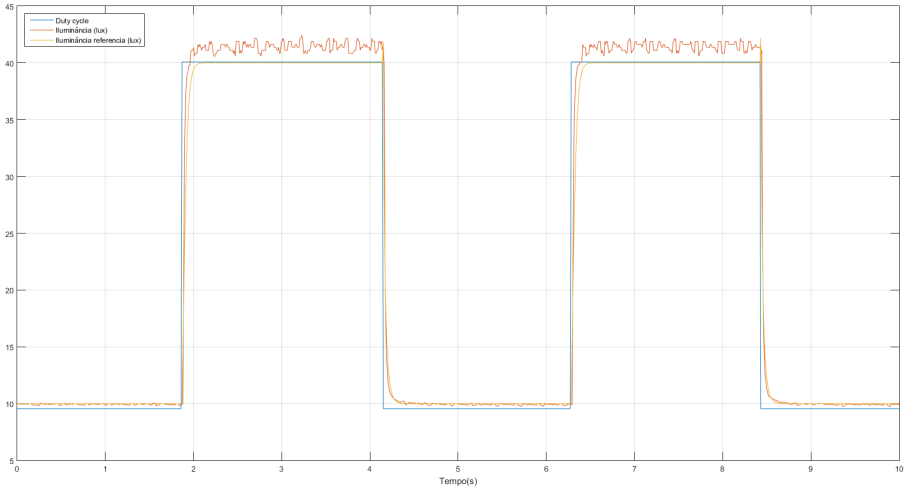

Figura 7: Resultados obtidos com o controlador *feedforward*.

Na Figura 7 pode-se observar a resposta do sistema utilizando apenas um controlador *feedforward*. Pode-se notar que o sinal de saída tem um desvio em relação à referencia. Isto deve-se a um calibração defeituosa. É também possível notar oscilações no valor da iluminância lida, as quais poderiam ser atenuadas utilizando um filtro passa baixo que fizesse a média das últimas leituras. O tempo de resposta obtido foi em média de 20 ms.

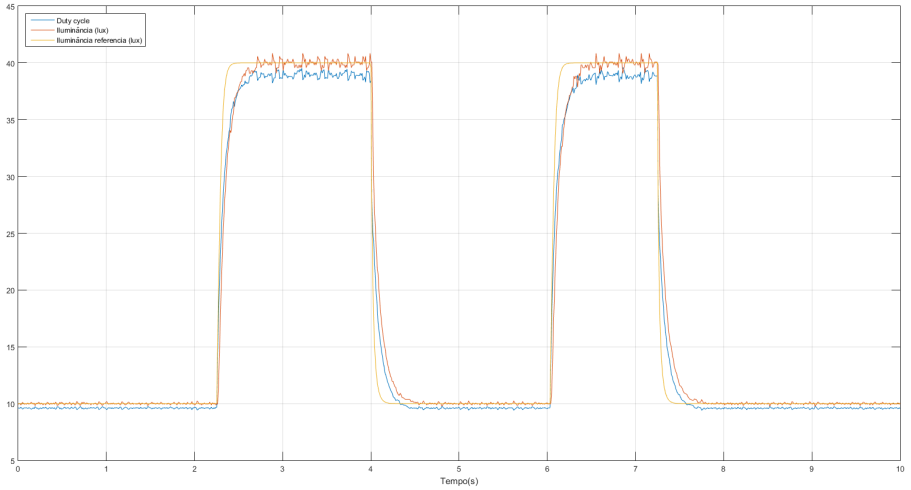

Figura 8: Resultados obtidos com o controlador *feedback*.

Na Figura 8 pode-se observar a resposta do sistema utilizando apenas um controlador por *feedback*. A resposta do sistema não apresenta erro estático, sendo capaz de seguir a referencia fornecida, contudo não é capaz de acompanhar os tempos dos transitórios, tendo um tempo de resposta de cerca de 55 ms.

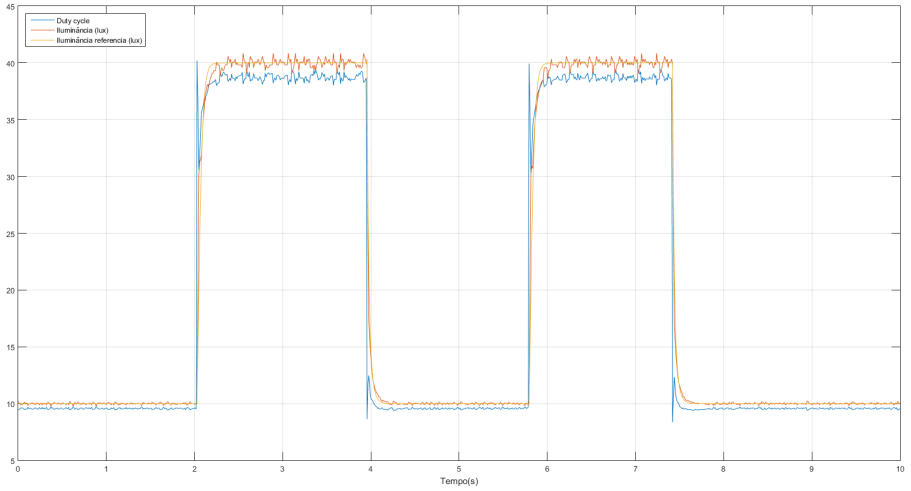

Figura 9: Resultados obtidos com o controlador completo.

Por fim o resultado obtido com o controlador completo pode ser visto na Figura 9.

Neste é possível notar um *overshoot* no *duty cycle* quando se dá a mudança de referência que não acontecia no controlador por *feedback*, pois o controlador por *feedforward* altera instantaneamente. Pode-se notar também que a resposta é mais rápida em relação ao controlador por *feedback* (cerca de 30 ms de tempo de resposta), acompanhando o valor de iluminância esperada ao longo do transitório.

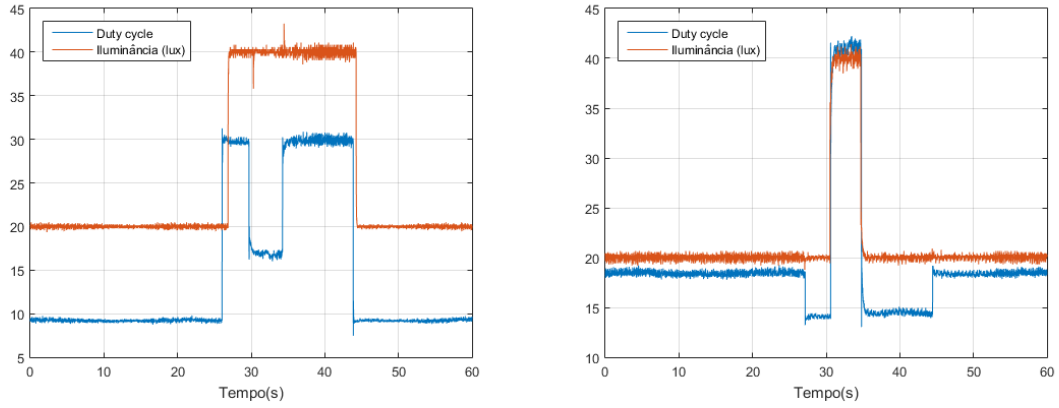

Figura 10: Variação do *duty cycle* e da iluminância com o tempo.

Os resultados para o controlador geral podem ser vistos na Figura 10. Nele é possível observar várias mudanças de referencia em dois elementos ao longo do tempo, e que o sistema é capaz de se adaptar, alterando a iluminância disponível num, sem intervir com a iluminância do outro. É possível notar alterações ao *duty cycle* de um elemento, sem que este altere a sua iluminância, devido à iluminância que lhe passou a ser disponibilizada pelo outro elemento. Isto deve-se ao facto de o controlo distribuido feito pelo consensus calcular referencias novas para o *duty cycle* de ambos que permita obter a iluminância desejada em ambos os pontos, sem que a alteração de uma provoque uma perturbação na outra. Nota final para o facto de o *duty cycle* na figura da esquerda estar ligeiramente avançado em relação à iluminância. Isto acontece pois alguns pontos são perdidos, durante a comunicação entre arduinos, ficando o *buffer* que guarda as iluminâncias lidas no ultimo minuto com mais pontos que o *buffer* que guarda o valor do *duty cycle*.

7 Conclusão

A solução apresentada mostra uma simulação de um sistema de controlo distribuído de forma a minorar o consumo de energia num escritório. É de reforçar a importância que a minimização de energia apresenta num caso real, quer seja por razões monetárias ou razões ambientalistas. A aplicação baseada na biblioteca Boost Asio apresenta uma inovação de comunicação assíncrona com o arduino, visto que a maioria de aplicações usando as bibliotecas da Boost são baseadas em sockets. O sistema está preparado para lidar com mais arduinos, sendo que a implementação de mais nós mas acabou por não ser experimentado.

O controlo do sistema é robusto, sendo capaz de lidar com variações na ocupação das secretárias, calculando novas referencias a partir do algoritmo consensus, e mantendo a iluminância praticamente inalterada.

Existem uns quantos pontos possíveis a melhorar no sistema completo. Um problema a resolver na aplicação será garantir que o comando de escrita e leitura dos arduinos sejam exclusivos, ou seja, que os comandos não se sobreponham, que pode ser resolvido com a utilização de *mutexs*. Outra melhoria seria não ter de definir o numero de arduinos do sistema, para facilitar a adição ou remoção de elementos. Finalmente, como o valor de iluminância exterior varia ao longo de um dia, principalmente em espaços com luminosidade exterior, para garantir o menor custo possível deveriam ser feitas calibrações periodicamente para garantir que os coeficientes são os mais corretos, não deixando apenas a cargo do controlador local, avaliando por exemplo o valor médio das correções do controlador local ao final de um certo intervalo de tempo.

8 Divisão do trabalho

O trabalho desenvolvido foi distribuído pelos diversos elementos da forma que a seguir se apresenta.

Divisão do trabalho desenvolvido:

- David Ribeiro - controlador local e global, comunicações entre elementos da rede, aplicação e comunicação serie, calibração e inicialização;
- Eduardo Pereira - controlador local, aplicação e comunicação serie, calibração e inicialização.

Divisão das secções do relatório:

- David Ribeiro - introdução, caracterização do LDR, determinação do tempo de resposta, controlador local, comunicações entre elementos da rede, calibração e inicialização, resultados, conclusão;
- Eduardo Pereira - resumo, introdução, circuito, controlador global, aplicação e comunicação serie, conclusão.

Referências

[1] Electronic, Lida Optical &, Cds Photoconductive Cells Gl5528, 2014, `https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf`

[2] Alexandre Bernardino, Solution of Distributed Optimization Problems: The consensus algorithm,2018