departamento de eletrónica, telecomunicações e informática

Curso 8309 - Mestrado Integrado em Engenharia Eletrónica e Telecomunicações
Disciplina 41489 – Sistemas de Instrumentação Eletrónica
Ano letivo 2019/20

Relatório

Projeto 1 – Sistema para Controlo de um Motor DC

Autores:
84667 Hugo Micael Resende Leal
84774 Miguel Filipe Pereira de Freitas Carvalhosa
Turma P2 Grupo 9

Data 02/05/2020
Docente Pedro Nicolau Faria da Fonseca

Resumo: O presente relatório visa apresentar o trabalho desenvolvido durante o projeto 1 da unidade curricular Sistemas de Instrumentação Eletrónica, o qual consiste no desenvolvimento de um sistema para controlo de um motor DC com a placa chipKit Max32.

Introdução

O presente relatório tem como objetivo apresentar o trabalho desenvolvido e os resultados deste para o Projeto 1, Sistema para Controlo de um Motor DC, da unidade curricular de Sistemas de Instrumentação Eletrónica.

O projeto 1 consiste no desenvolvimento de um sistema que permita controlar a velocidade e sentido de rotação de um motor DC, com recurso a um microcontrolador. Este projeto enquadra-se na unidade curricular, pois permite aos alunos adquirirem competências no uso deste tipo de sistema, bem como as interfaces que permitem fazer o controlo deste e a respetiva leitura da velocidade e posição do veio. Permite ainda aos alunos adquirirem experiência na implementação de controladores do tipo proporcional-integrador (PI) em microcontroladores.

Este relatório segue uma abordagem "*top-down*". Primeiramente será apresentada uma descrição da arquitetura do sistema desenvolvido, com recurso a diagramas de blocos, descendo de seguida até à especificação pormenorizada do hardware e do software desenvolvidos.

Descrição do problema e objetivos

O problema apresentado aos alunos consiste no desenvolvimento de um sistema para controlo de um motor DC usando a placa de desenvolvimento *chipKit Max32* da Digilent. O principal objetivo é permite ao utilizador controlar o sentido e velocidade de rotação do motor. Assim, o sistema deve possuir uma interface com o utilizador, no computador, que permita a este definir o *setpoint*, tanto de velocidade como do sentido de rotação. A interface deve também mostrar ao utilizador a velocidade instantânea do motor, o sentido de rotação do veio e a posição angular do veio em relação a um valor fixo. O sistema deve ainda conter uma interface que permita controlar o sentido e velocidade do motor DC. A obtenção dos dados em questão é realizada através da leitura de um *encoder* com dois canais.

O problema tem as seguintes especificações:

- Comunicação entre o computador e o microcontrolador deve ser realizada através da UART;
- Velocidade de rotação deve variar entre 10 e 50 rpm, com o passo de 1 rpm;
- A energia entregue ao motor deve ser realizada com o uso de um sinal PWM, com a resolução de pelo menos 100 passos;
- A frequência do sinal PWM deve ser superior a 20kHz, de modo a evitar ruído acústico.
- Deve ser possível ao utilizador parar o motor colocando as rotações por minuto a 0;
- A interface com o utilizador deve mostrar a velocidade instantânea com resolução de 1 rpm;
- A interface com o utilizador deve exibir a posição angular do veio com a resolução de pelo menos 2 graus;
- A função de controlo deve ser implementada com recurso a um controlador Proporcional-Integrador (PI).

Arquitetura do Sistema

A arquitetura do sistema está representada com recurso a diagramas de blocos e divide-se em 3 níveis:

- **nível 0**, onde se representa o sistema visto como um todo, estando representados os sinais de entrada e saída da generalidade do sistema;
- **nível 1**, onde se apresenta o sistema dividido pelos seus blocos principais e as entradas e saídas de cada um destes;
- **nível 2**, onde se apresenta a estrutura interna do microcontrolador e os blocos de hardware e software utilizados, bem como as entradas e saídas de cada um.

Diagrama de blocos de nível 0

Na Figura 10, encontra-se o diagrama de blocos de nível 0 do sistema. Neste, o sistema é representado na sua totalidade como apenas um bloco.

Neste diagrama são representados os sinais de entrada e saída do sistema com as seguintes características:

- **S1**: Representa o *setpoint* da velocidade e sentido e rotação inseridos pelo utilizador;
- **S2**: Informação acerca do valor das grandezas físicas sobre as quais estamos a atuar;
- **S3**: Sinal digital UART.

Diagrama de blocos de nível 1

Na Figura 11, encontra-se esquematizado o diagrama de blocos de nível 1 do sistema. Neste encontram-se representados os seguintes blocos:

- **chipKit Max32**: onde se realizará o envio do sinal PWM à interface de potência, leitura dos encoders, implementação do controlador e a interface com o utilizador.
- **Interface de potência:** Corresponde ao circuito integrado L239D e é o bloco responsável por fazer a conversão do sinal PWM em tensão a ser aplicada ao motor.
- **Motor DC**: Corresponde sistema que queremos controlar. Recebe sinais em corrente da ponte H e, mediante estes, o motor roda a uma determinada velocidade e num certo sentido. Possui uma interface com encoders que gera dois sinais em tensão digitais em quadratura, a partir dos quais é possível ler a posição do veio, a velocidade do motor e o sentido de rotação.

No diagrama encontram-se ainda representados os seguintes sinais:

- **S4**: Dois sinais PWM com frequência de 20kHz e amplitude 3.3V, negados entre si.
- **S5**: Sinal em corrente aplicado ao motor que varia de acordo com o sinal PWM aplicado. Chega no máximo a 600mA de amplitude e 1.2A de pico.
- **S6**: Dois sinais digitais com amplitude 3.3V e fase em quadratura entre si.

Diagrama de blocos de nível 2

Na Figura 12, encontra-se representado o diagrama de blocos de nível 2 do sistema. Neste diagrama, o microcontrolador é representado pelos blocos de software e hardware que o constituem, nomeadamente:

- **UART1**: bloco para enviar e receber dados para o computador via UART.
- **Controlador**: bloco de software que implementa um algoritmo de um controlador proporcional-integrador que trata do controlo do motor DC para o *setpoint* desejado.
- **Gerador de PWM**: bloco para gerar um sinal PWM baseado em um *timer* e um *output compare module* do PIC32
- **Leitura do Encoder:** bloco de software que realiza a leitura do *encoder* do motor DC

Projeto e conceção

Atuação sobre a velocidade e sentido de rotação do motor

O motor utilizado neste projeto é um dispositivo que necessita de alguma potência para funcionar devidamente. Como o microcontrolador não consegue fornecer a potência que o motor necessita, é necessário utilizar algum dispositivo que consiga transformar os pequenos sinais do microcontrolador em sinais de potência para alimentar o motor.

O dispositivo utilizado neste projeto é o circuito integrado L293D, que contém duas pontes H. A ponte H pode ser vista como um conjunto de quatro interruptores controlados eletricamente que, consoante a combinação escolhida, permitem definir o sentido de rotação de um motor DC. Um modelo da ponte H encontra-se na Figura 1 - Modelo da ponte H.

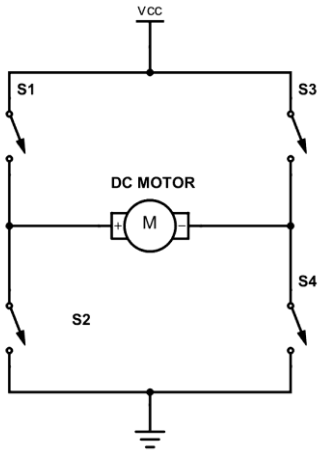

Figura 1 - Modelo da ponte H

Se os interruptores S1 e S4 estiverem fechados, flui uma corrente pelo motor num sentido. No entanto, se os interruptores S2 e S3 estiverem fechados, a corrente no motor flui no sentido oposto. Assim é possível alterar o sentido de rotação do motor. Também é possível travar o motor, fechando os interruptores S1 e S3 ou S2 e S4, uma vez que a queda de tensão aos terminais do motor será nula.

Para controlar a velocidade do motor utilizou-se uma técnica bastante vulgar que consiste no uso de sinais PWM.

A ponte H utilizada tem apenas duas entradas, em que cada entrada controla um lado da ponte (i.e. envia o sinal para o transístor de cima e o sinal negado para o transístor de baixo).

Se forem aplicados dois sinais PWM, um a cada lado da ponte, é possível controlar a velocidade de rotação do motor variando o *duty cycle* destes. De notar que os sinais devem ser simétricos, i.e., quando um sinal está em t_{on} o outro sinal está em t_{off} e vice-versa.

Na Figura 2 - Sinais PWM podem-se observar alguns exemplos dos sinais utilizados.

- No caso 1, ambos os sinais têm 50% de *duty cycle*. Neste caso, o motor está travado porque a média dos dois sinais ao longo de um período é nula. Durante a primeira metade do período o motor roda num sentido e durante a segunda metade do período o motor roda no sentido oposto. Como o motor tem uma característica de passa-baixo, estas comutações a alta frequência não são visíveis e o motor permanece parado.
- No caso 2, o sinal A tem um *duty cycle* superior ao do sinal B. Neste caso, o motor roda num sentido porque ao longo de um período, o sinal A está ativo durante mais tempo.
- No caso 3, o sinal A tem um *duty cycle* inferior ao do sinal B. Neste caso, o motor roda no sentido oposto porque ao longo de um período, o sinal B está ativo durante mais tempo.

Assim é possível alterar a velocidade de rotação do motor ajustando o *duty cycle* de ambos os sinais.

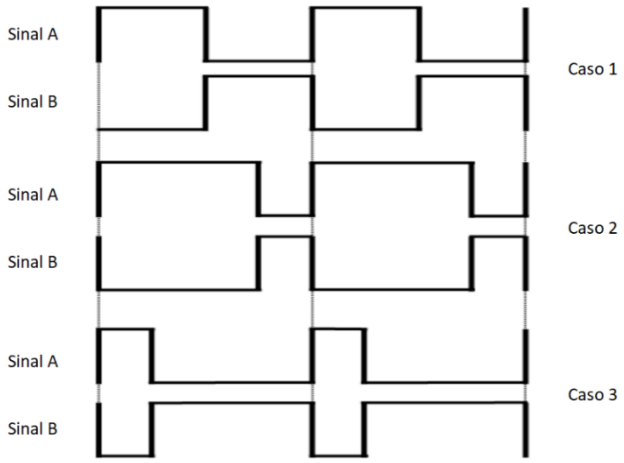

Figura 2 - Sinais PWM

Para gerar os sinais PWM decidiu-se gerar apenas um sinal PWM com o PIC32 e obter o sinal com a lógica invertida utilizando uma porta NOT. Esta decisão deve-se ao facto de ser mais simples inverter o sinal com hardware do que por software. Utilizou-se o circuito integrado HEF4011B, que possui quatro portas NAND, devido a ser o único IC possuído pelo grupo. Construiu-se uma porta NOT curto-circuitando ambas as entradas de uma porta NAND.

Assim o sinal A é obtido diretamente do pino do PIC32 e o sinal B é obtido passando o sinal A pela porta NOT. De notar que o atraso imposto pela porta NOT é no máximo 120ns, valor que é desprezável comparando com o período do sinal PWM.

Em termos de software, criou-se uma função que recebe como parâmetro de entrada um valor entre -127 e +127. Valores positivos fazem o motor rodar num sentido e valores negativos fazem o motor rodar no sentido oposto.

Leitura do encoder

O motor utilizado neste projeto possui um encoder acoplado ao seu veio. Este encoder tem dois sensores de efeito de hall desfasados de 90º que produzem duas ondas quadradas desfasadas de 90º entre si, com duty cycle de 50%, amplitude de 5V e frequência variável consoante a velocidade de rotação do motor. Sabe-se do *datasheet* que, por cada volta completa do motor, o encoder produz 180 pulsos em cada saída. No entanto, após verificação prática descobriu-se que este valor está errado, sendo o valor correto 420 pulsos.

Os sinais provenientes do encoder têm uma amplitude de 5V. No entanto, as entradas do microcontrolador apenas suportam tensões até 3.3V. Para resolver este problema, construiu-se um divisor resistivo para converter os 5V do encoder para 3.3V suportados pelo PIC32. Para além de converter os níveis lógicos, este circuito também resolve outro problema presente neste tipo de encoder: o facto de utilizar uma configuração *open-drain* significa que é necessário inserir uma resistência de *pull-up* para evitar que o pino fique a "flutuar" (i.e. sem estar ligado a nada), o que faria com que o sinal fosse afetado pela mais pequena interferência. Uma das resistências do divisor resistivo é então utilizada também como resistência de *pull-up*.

Sabe-se que a relação entre a tensão de entrada e de saída de um divisor resistivo é a seguinte:

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2} \quad \leftrightarrow \quad \frac{R_2}{R_1 + R_2} = \frac{V_{out}}{V_{in}}$$

onde R_1 é a resistência entre V_{in} e V_{out} e R_2 é a resistência entre V_{out} e a referência.

Sabendo que $V_{out} = 3.3V$ e $V_{in} = 5V$ e escolhendo $R_2 = 10k\Omega$, obteve-se $R_1 = 5.1k\Omega$. Como não se possuía este valor, utilizou-se o valor mais próximo, $R_1 = 5.6k\Omega$, o que resulta na tensão de saída $V_{out} = 3.2V$, que é um valor aceitável.

Para aumentar a resolução da medição, decidiu-se medir não apenas cada pulso, mas cada transição do sinal de um dos canais. Assim verificam-se 840 transições do sinal de um dos canais, tendo a medição o dobro da resolução do que apenas contando o número de pulsos.

Para garantir que todas as transições do encoder são contadas, decidiu-se ligar um dos canais do encoder a um pino digital com *Change Notice Interrupt*. Isto permite que seja gerada uma interrupção sempre que ocorre uma mudança do sinal nesse pino. O segundo canal do encoder foi ligado a um pino digital do microcontrolador também configurado como entrada.

Na Figura 3 - Sinais de saída do encoder encontra-se uma representação dos sinais de saída do encoder, retirada do *datasheet* do mesmo. Como se pode verificar, quando o motor se encontra a rodar no sentido 1, no flanco ascendente do sinal A, o sinal B tem o valor 0. No entanto, quando o motor se encontra a rodar no sentido 2, no flanco ascendente do sinal A, o sinal B tem o valor 1. Assim é possível determinar o sentido de rotação, analisando o canal B sempre que o

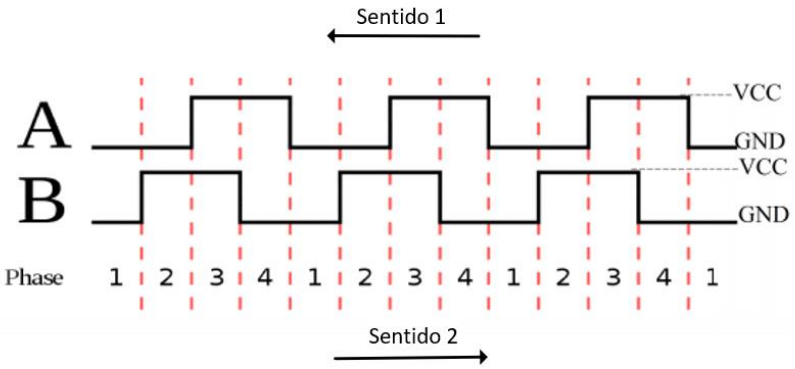

Figura 3 - Sinais de saída do encoder

canal A transita de estado.

Em termos de software, criou-se uma rotina de serviço à interrupção que é executada sempre que o canal A muda de estado. Dentro da rotina verifica-se o sentido de rotação, através de diversas condições, e altera-se o valor de um contador. Este contador serve mais tarde para calcular a velocidade de rotação e é incrementado se o motor estiver a rodar no sentido 1 ou é decrementado se o motor estiver a rodar no sentido 2.

Da mesma forma, também se verifica a posição angular do veio do motor. Sabe-se que cada volta do veio corresponde a 840 transições do canal A, logo facilmente se conclui que cada transição do canal A corresponde a 0.43°. Para determinar a posição angular do veio, apenas é necessário incrementar (ou decrementar, consoante o sentido) a posição em 0.43° a cada passagem pela rotina de serviço à interrupção. De notar que este incremento deve ser realizado de uma forma circular para ter valores cíclicos entre 0 e 360°.

Cálculo da velocidade

O contador do encoder é constantemente alterado quando são geradas as interrupções. No entanto, pretende-se obter amostras da velocidade num regime periódico fixo de modo a mais tarde implementar um controlador.

Para isso configurou-se um timer para gerar eventos periódicos a uma frequência de amostragem fixa definida à priori. Para garantir que se obtém uma amostra assim que o timer termina a contagem e aproveitando o restante tempo para realizar outras operações, decidiu-se utilizar uma interrupção gerada pelo timer. Na rotina de serviço à interrupção determina-se o número de pulsos gerados pelo encoder no período de amostragem, comparando o valor do contador do encoder atual com o valor no início do período. Sabendo que ocorreu um determinado número de pulsos num intervalo de tempo fixo e conhecido, facilmente se calcula a velocidade em rotações por minuto.

Sabendo que o encoder gera 840 pulsos por rotação, o número de pulsos gerados corresponde a $\frac{pulsos}{840}$ rotações.

Sabendo ainda que os pulsos são gerados num intervalo de tempo fixo, h, então o número de rotações por minuto equivale a $\frac{pulsos \times 60}{840 \times h} \rightarrow RPM = \frac{pulsos \times 60 \times F_a}{840}$.

Esta operação é executada na rotina de serviço à interrupção por se tratar de uma operação simples. No entanto, sempre que é gerada uma interrupção, ativa-se uma *flag* para impor no ciclo infinito o período de amostragem.

Interface com o utilizador via UART

O projeto em questão possui uma forte componente de interação com o utilizador, e para tal, é essencial o desenvolvimento de uma interface com o mesmo. Esta interface deve ser capaz de:

- mostrar ao utilizador os valores de velocidade instantânea em rpm, sentido de rotação e posição do veio em graus
- adquirir valores de *setpoint* entre 10 e 50 rpm, sentido de rotação e uma ordem de paragem do motor que equivale a adquirir um valor de 0 rpm.

Para o primeiro ponto, após termos feito a aquisição e o cálculo dos valores de velocidade instantânea e posição angular, realizamos apenas uma impressão destes valores no terminal via a UART1. Visto que interessa ao utilizador ter conhecimento destes valores de forma contínua, esta impressão é executada a cada iteração do ciclo de controlo.

Para o segundo ponto, os valores de *setpoint* inseridos pelo utilizador são adquiridos usando uma rotina de serviço à interrupção da UART1, de modo a proceder-se à alteração do setpoint apenas quando desejado pelo utilizador, evitando parar a execução do programa em todas as iterações do ciclo de controlo. Os valores inseridos pelo utilizador de interesse são: entre 10 e 50 rpm, -10 e -50 rpm e 0 rpm. Sinal positivo de rpm indica rotação num sentido e o sinal negativo indica rotação no sentido oposto. Assim, sempre que o utilizador insere um caracter do teclado, é gerada uma interrupção e o programa vai para a rotina de interrupção para ser efetuada a leitura deste. Nesta rotina, o programa realiza um primeiro processamento, se o valor lido é um dígito ou carater "-", este é guardado numa estrutura de dados baseada numa *stack*. Quando a *stack* se encontrar cheia ou o utilizador inserir a tecla "*enter*", uma *flag* é ativada. Esta *flag* tem a função de indicar que os dados estão prontos para processamento ao mesmo tempo que impede a rotina de interrupção de guardar mais dados na estrutura de dados.

Assim que os dados inseridos pelo utilizador se encontrem prontos para processamento, o programa executa a função "*print_UI*". Nesta função são efetuados o *parse* e a validação dos dados inseridos na *stack*. Em primeiro, é retirado da

stack o primeiro carater a ser processado e de seguida é verificado se é dígito ou sinal. Caso seja dígito e o único elemento presente na *stack*, é posteriormente verificado se este dígito corresponde ao número zero, único *setpoint* válido para um número com estas características, e de seguida, este é enviado para o controlador. Se a stack não estiver vazia, o programa verifica as condições análogas às anteriores. Verifica se é um dígito e se a stack está vazia. Caso isto se confirme, o programa faz a concatenação dos dois algarismos e verifica se este está entre os valores de 10 e 50, e de seguida envia para o controlador. Por fim, caso a stack, neste ponto, ainda contenha elementos, o algoritmo que desenvolvemos verifica-se se este elemento corresponde ao carater "-". Em caso afirmativo, os elementos previamente retirados são concatenados, e se o valor final estiver dentro da gama de -10 a -50 rpm, este é enviado para o controlador.

Para as operações anteriormente descritas, a *stack* tem implementada uma função para adicionar um elemento a esta, "*push*", outra para retirar um elemento desta e devolvê-lo "*pop*" e uma última para verificar o estado desta (vazia, completamente cheia ou com espaço livre para adicionar elementos).

Todos os casos que não valores de 0 rpm, entre 10 e 50 rpm e de -10 a -50 rpm, correspondem a valores de *setpoint* inválidos e, portanto, se identificados pela função darão origem a uma mensagem de erro no terminal.

Na Figura 4, encontra-se um fluxograma que resume de forma gráfica o código efetuado para o segundo ponto descrito.

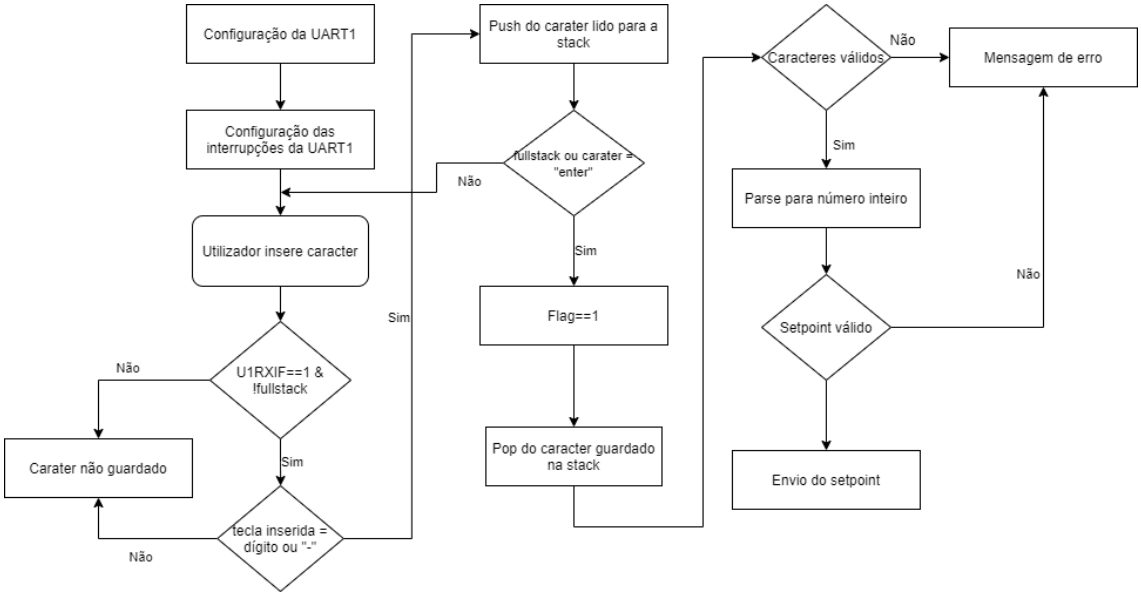

Figura 4 - Fluxograma da receção e validação de um setpoint

Controlador PI

Para colocar o motor no *setpoint* desejado pelo utilizador é necessário um controlador. Para este efeito, e seguindo as especificações, desenvolvemos um controlador do tipo proporcional-integrador (PI).

Partindo do modelo contínuo do controlador $Gr(s) = K * \left(1 + \frac{1}{sTi}\right)$, em que K é o ganho e Ti corresponde à constante de integração do erro, é possível obter a expressão para este controlador no modelo discreto, obtendo-se a expressão $Gr(q) = \frac{s0+s1*q^{-1}}{1+q^{-1}}$, no qual $s0 = K * (1 + \frac{h}{Ti})$ e $s1 = -K$, e em que h é o intervalo de amostragem. A função de controlo deste controlador é dada por $u(k) = \frac{s0+s1*q^{-1}}{1-q^{-1}} * \big(r(k) - y(k)\big)$, em que r(k) corresponde ao *setpoint* e y(k) ao sinal à saída, conforme o esquematizado na Figura 5. A subtração do sinal r(k) pelo sinal y(k), pode ser definido como o sinal de erro (e(k)), assim, fazendo $e(k) = r(k) - y(k)$, e substituindo na expressão anterior do controlador, obtemos uma nova função de controlo definida por $u(k) = \frac{s0+s1*q^{-1}}{1-q^{-1}} * e(k)$. Desenvolvendo esta última, chegamos à expressão usada no algoritmo para o sinal de controlo: $u(k) = u(k-1) + s0 * e(k) + s1 * e(k-1)$, computacionalmente bastante eficiente. Os resultados foram obtidos com recurso à referência bibliográfica [8], onde se encontra também a demonstração completa dos resultados matemáticos obtidos.

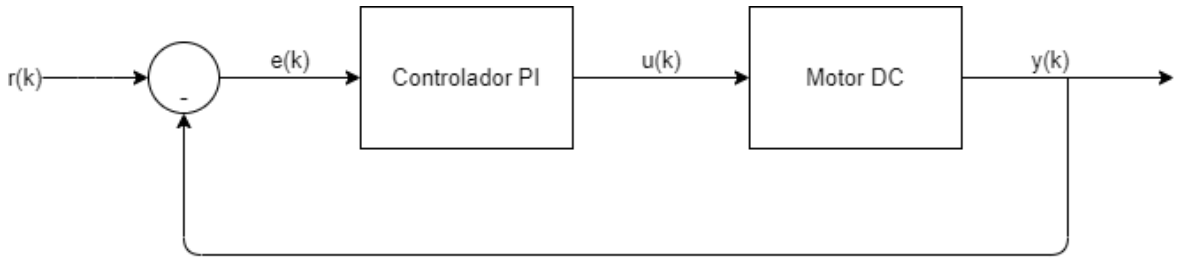

Figura 5 – Representação geral do controlador implementado

O algoritmo segue os seguintes pontos de desenvolvimento:

- definição dos parâmetros do controlador: h, K e Ti;
- declaração das variáveis para o sinal de controlo (em k e k-1), referência, erro (em k e k-1) e saída a 0;
- cálculo dos parâmetros s0 e s1;
- em ciclo infinito, igualar o sinal de referência ao *setpoint* inserido pelo utilizador, igualar o sinal de saída à velocidade e sentido de rotação lidos do motor e cálculo do sinal de erro em k;
- cálculo do sinal de controlo e igualar os sinais de controlo e erro em k-1 aos sinais de controlo e erro em k, respetivamente;
- Saturar o sinal de controlo. Sendo o número de passos do sinal PWM de 256, o sinal de controlo é saturado positivamente em 127, que corresponde a um dos sentidos de rotação e negativamente em -127, o outro sentido de rotação.
- Enviar sinal de controlo como argumento da função que gera os sinais PWM a serem enviados para a ponte H.

É, então, necessário obter os parâmetros h, K e Ti para a melhor resposta possível. Com melhor resposta possível compreende-se que a saída consegue chegar ao *setpoint* desejado ajustando o parâmetro K, e de seguida o erro desta em regime estacionário tem de ser nulo, para o *setpoint* ser exatamente o definido pelo utilizador, através do ajuste do parâmetro Ti. Visto que quanto maior o termo integrador pior será o regime transitório, ou seja, este faz com que o sistema em questão demore mais tempo a estabilizar no *setpoint* desejado. Assim, o parâmetro Ti, deve ser dimensionado de modo a garantir que o erro em estado estacionário seja nulo e que o sistema tenha uma resposta rápida o suficiente em regime transitório. Estes foram obtidos através de experimentação e análise da resposta do sistema aos diversos parâmetros aplicados.

Resultados e Análise

Devido à situação atual de pandemia do Covid-19, as aulas presenciais estão suspensas e os alunos não têm acesso ao laboratório para desenvolver e testar o sistema nas condições ideais. No entanto, foram feitos todos os possíveis para montar e testar os circuitos e o sistema em casa com as condições que o grupo possui.

Começou-se por testar o *encoder* do motor. Após montar o circuito para conversão de níveis lógicos, ligou-se o motor a uma fonte de tensão de 12V e visualizaram-se os sinais do *encoder* com recurso a um analisador lógico. Apesar de não ser a melhor forma de visualizar os sinais, foi a única forma possível dado que os alunos não têm acesso a um osciloscópio. Na Figura 6 - Sinais produzidos pelo encoder encontram-se os sinais obtidos, que estão de acordo com o esperado.

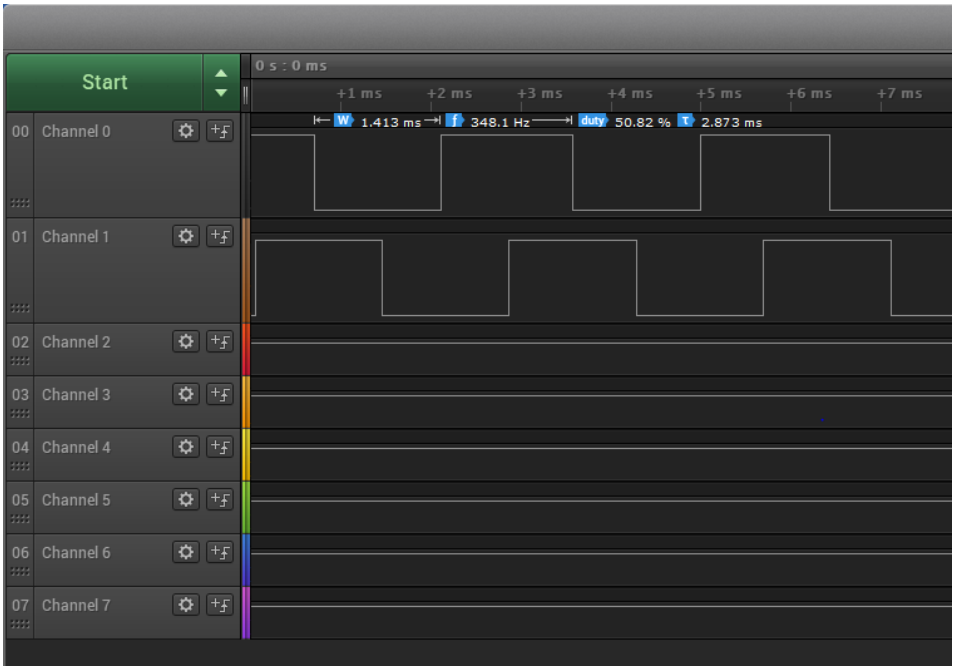

Figura 6 - Sinais produzidos pelo encoder

De seguida procedeu-se ao teste da ponte H e do motor. Mais uma vez, os sinais PWM gerados pelo microcontrolador foram visualizados com recurso a um analisador lógico e encontram-se na Figura 7 - Sinais PWM enviados para a ponte H. Pode-se observar que os sinais têm a frequência desejada e que têm a lógica invertida entre si.

Testou-se a ponte H enviando sinais com todos os valores possíveis de *duty cycle* e observou-se o motor a aumentar gradualmente a velocidade em ambas as direções, como o esperado.

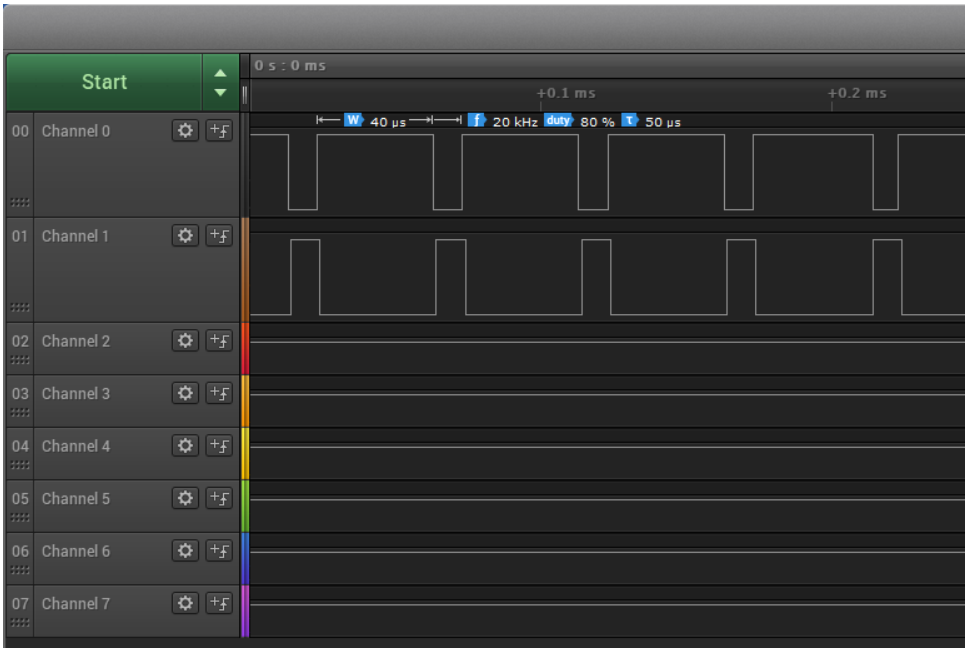

Figura 7 - Sinais PWM enviados para a ponte H

O teste do ajuste do *setpoint* foi realizado imprimindo no terminal o valor calculado para verificar se estava de acordo com a sequência de teclas pressionadas pelo utilizador. Testaram-se várias situações de *setpoints* inválidos para validar o correto funcionamento do módulo. Na Figura 8 - Valores de setpoint introduzidos encontra-se uma imagem do terminal onde se verificam alguns dos valores introduzidos.

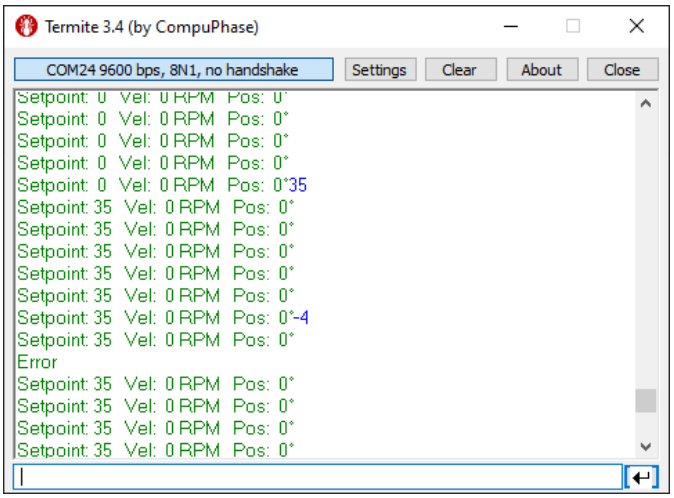

Figura 8 - Valores de setpoint introduzidos

Por fim procedeu-se ao teste do controlador. Este teste engloba todos os módulos desenvolvidos e valida o funcionamento do sistema como um todo. Testaram-se vários valores de *setpoint* e verificou-se um correto ajuste da velocidade do motor consoante o valor introduzido. Também se verificou que os valores da posição do veio estavam de acordo com os valores reais.

Em termos da performance do controlador, verificou-se que a resposta do sistema tem um tempo de subida razoável, na ordem dos 3 segundos, um ligeiro *overshoot* e um tempo de estabelecimento de cerca de 2 segundos.

A performance do sistema poderia ser ainda melhor fazendo um ajuste mais rigoroso dos parâmetros do controlador.

Na Figura 9 - Interface com o utilizador encontra-se uma imagem do terminal onde se pode observar a interface com o utilizador quando o sistema está em funcionamento.

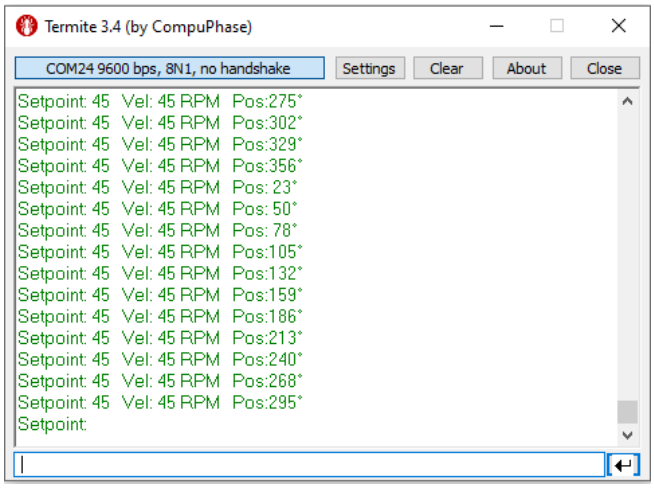

Figura 9 - Interface com o utilizador

Por fim, foi gravado um pequeno vídeo a demonstrar o funcionamento do sistema, que pode ser consultado no seguinte link: https://www.youtube.com/watch?v=MJg8bpmxOFU

Conclusão

Após análise dos resultados obtidos, verifica-se que o sistema desenvolvido cumpre com os requisitos presentes no problema proposto.
O trabalho desenvolvido teve impacto positivo na medida em que deu aos alunos competências no trabalho com interfaces de potência do género da ponte H, leitura de *encoders* de motores DC e implementação de algoritmos de controlo em microcontroladores. Permitiu ainda o desenvolvimento de boas práticas de programação em C e em microcontroladores e, findado este projeto os alunos têm módulos base já configurados e prontos para utilização noutros projetos.
Assim, conclui-se que o projeto foi finalizado com êxito.

Referências

[1] Pedro Fonseca. “Guia para a redação de relatórios”. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2012.

[2] Pedro Fonseca. “Lab assigments guide”. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2020.

[3] Pedro Fonesca, “Sistemas de Instrumentação Eletrónica” apontamentos. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2020.

[4] Pedro Fonseca, Paulo Pedreiras. “Position, Speed and Acceleration”, slides aulas teóricas “Sistemas de Instrumentação Eletrónica”. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2020.

[5] Pedro Fonseca, Paulo Pedreiras. “Power Interfaces”, slides aulas teóricas “Sistemas de Instrumentação Eletrónica”. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2020.

[6] Cytron Technologies. User manual: MO-SPG-30E-XXXK. Electronic Publication. Maio, 2011.

[7] Texas Instruments. Datasheet: L239X, Quadruple Half-H drivers. Electronic Publication. Janeiro, 2016.

[8] Alexandre Mota, “Sistemas e Sinais Discretos”, apontamentos. Departamento de Eletrónica, Telecomunicações e Informática. Universidade de Aveiro. 2019.

Anexos

Anexo 1: Diagramas da arquitetura do sistema

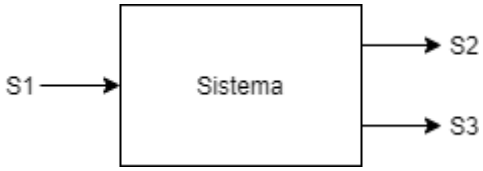

Figura 10- Arquitetura nível 0 do sistema

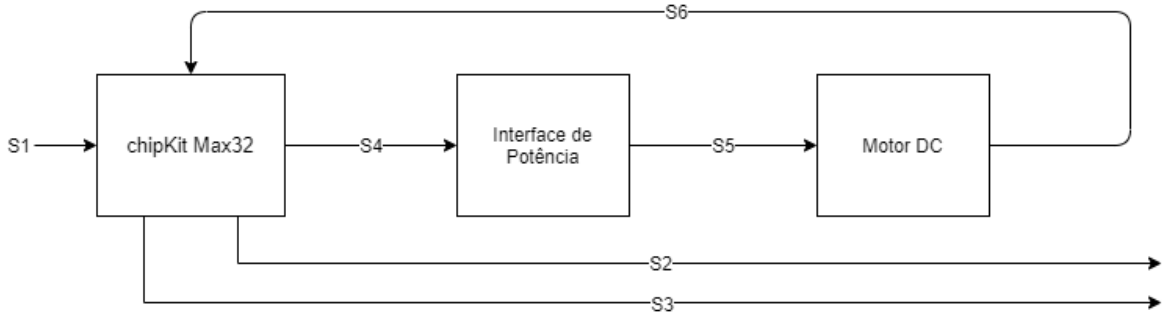

Figura 11 - Arquitetura nível 1 do sistema

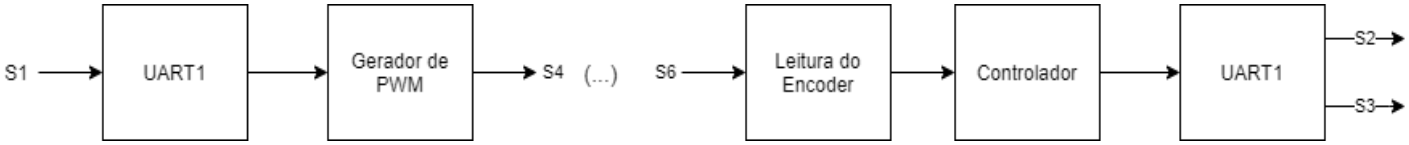

Figura 12 - Arquitetura nível 2 do sistema

Anexo 2: Esquema elétrico do sistema

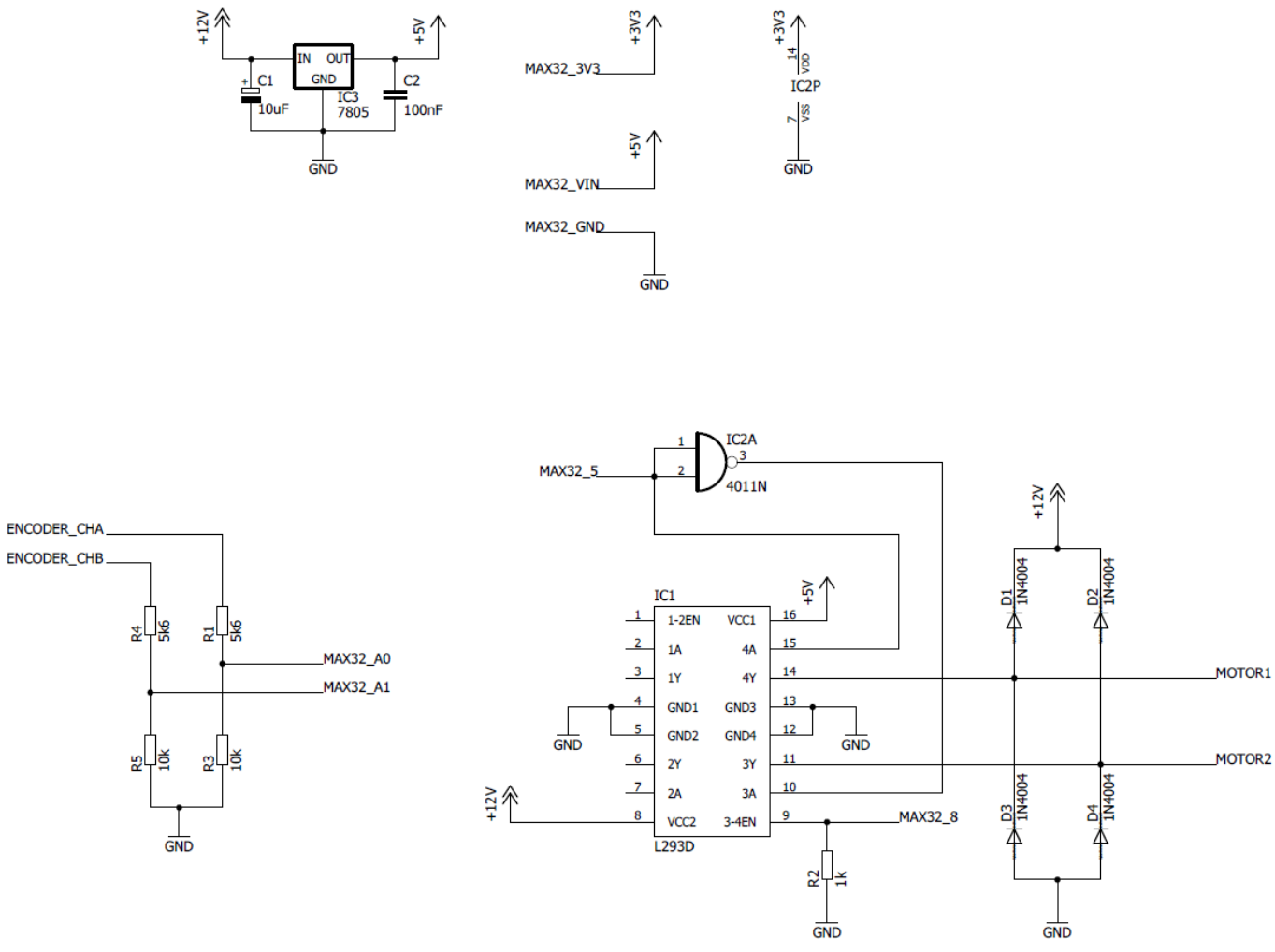

Figura 13 - Esquema elétrico do sistema