departamento de eletrónica, telecomunicações e informática

Curso 8309 - Mestrado Integrado em Engenharia Eletrónica e Telecomunicações

Disciplina 41489 – Sistemas de Instrumentação Eletrónica

Ano letivo 2021/2022

Relatório

Projeto 1 – Controlo de um Motor DC

Autores:

84953 Bruno Santos

84739 Manuel Silva

Turma P2 Grupo 9

Data 31/05/2022

Docente Pedro Fonseca

Resumo: Este relatório serve para mostrar e explicar o trabalho implementado no projeto de "Controlo de um motor DC". O objetivo era criar um sistema que fosse capaz de medir e controlar a velocidade de um motor DC, assim como medir o ângulo, parar o motor e mudar a sua direção de rotação. O microcontrolador envia um sinal de *PWM* (cujo *Duty-cycle* deverá ser regulado para que o motor atinja a velocidade desejada) para a ponte-H (L293NE) que controla o sentido de rotação do motor, e posteriormente o *encoder* gerará pulsos que serão lidos pelo microcontrolador, para o cálculo da velocidade a que o motor está. Face aos resultados que serão aqui apresentados, podemos concluir que os objetivos deste trabalho foram alcançados com sucesso, embora se pudesse melhorar o tempo de resposta transitória do controlador PI.

Introdução

Os motores DC estão muito presentes na nossa vida doméstica (através de vários eletrodomésticos, como por exemplo aspiradores e varinhas-mágicas), pelo que existe uma necessidade de termos na sua presença mecanismos de controlo para controlar o seu comportamento.

Como mencionado no resumo, o âmbito deste projeto é o *design* de implementação de um sistema de controlo para o motor DC SPG30E-60K, através da placa de desenvolvimento chipKIT MAX32. Para isso, será necessário fazer *design* e implementação de *Software, hardware e firmware*, algo que vai ser mostrado neste documento.

Descrição do problema e objetivos

O sistema deve ser capaz de ajustar a direção de rotação do veio e a sua velocidade (entre 10 e 50 RPM, com uma resolução de 1 RPM), através de uma interface gráfica, no computador (usando a UART). O sistema deve ainda poder parar (0 RPM) a qualquer momento.

Ao nível do controlo, através de um controlador PI, devemos gerar um sinal *PWM* com uma resolução de 100 níveis e uma frequência mínima de 20kHz, que será convertido (pelo microcontrolador) de um sinal *PWM* de baixa potência, para um sinal *PWM* com potência suficiente para que o motor rode. Isto é conseguido pela L293NE a 14V. Alimentamos a 14V a Ponte-H porque é a 14V que o motor consegue atingir a velocidade máxima estabelecida pelo fabricante (75 RPM).

Podemos então dividir os objetivos da seguinte forma:

- *Design* e implementação de Hardware (Ponte-H, Inversor *PWM*).
- Desenvolvimento de *Software* do sistema (Controlador PI, Timers, Interrupções).
- Desenvolvimento de uma interface gráfica.

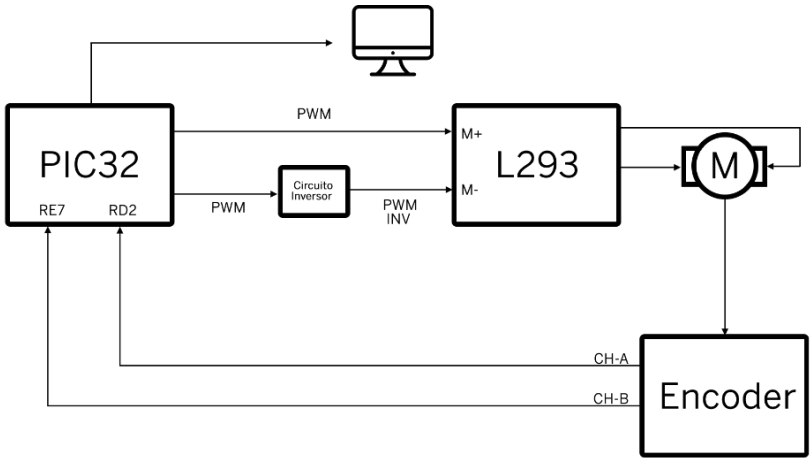

Figura 1- Esquema do Sistema de Controlo do motor DC

Projeto e conceção

Implementação do *Hardware*

Após a construção do diagrama de blocos, que nos serviu para entender a arquitetura do sistema e para saber como o vamos implementar, começámos pela análise e montagem do *hardware* que vamos usar. Em primeiro lugar, é fundamental conhecer o instrumento que vamos controlar: o próprio motor.

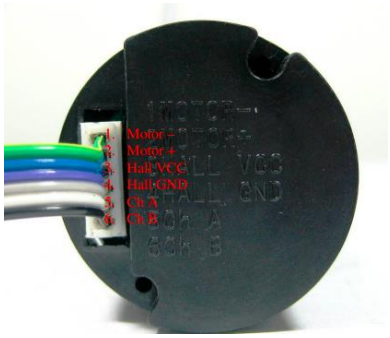

Figura 2- Esquema dos pinos do motor

A figura 2 mostra-nos que o motor contém 6 pinos. Nos pinos 1 (Motor-) e 2 (Motor+), vai entrar o sinal que vai controlar a velocidade e a direção do motor, 3 (Hall VCC) e 4 (Hall GND) são os pinos de alimentação do *encoder* de efeito de Hall, cujos impulsos gerados irão permitir ao utilizador saber o sentido de rotação do motor. Por fim, os pinos 5 e 6 são os canais de saída do *encoder.* Existe um atraso de fase de 90º entre os impulsos gerados, assim, se o Ch.B estiver atrasado em relação ao Ch.A, o utilizador saberá a partir dessa informação que o motor está a rodar no sentido dos ponteiros do relógio, caso contrário, o motor irá rodar no sentido oposto.

Como foi mencionado no parágrafo anterior os pinos Motor+ e Motor- estão ligados a um sinal que irá controlar o comportamento do motor. Para que isso aconteça utilizámos um dispositivo que faz a gestão desse sinal: o L293NE. Este dispositivo foi feito para fornecer correntes bidirecionais, assim é possível assegurar que o motor rode nos dois sentidos. A figura 3 ilustra como foi montado este circuito:

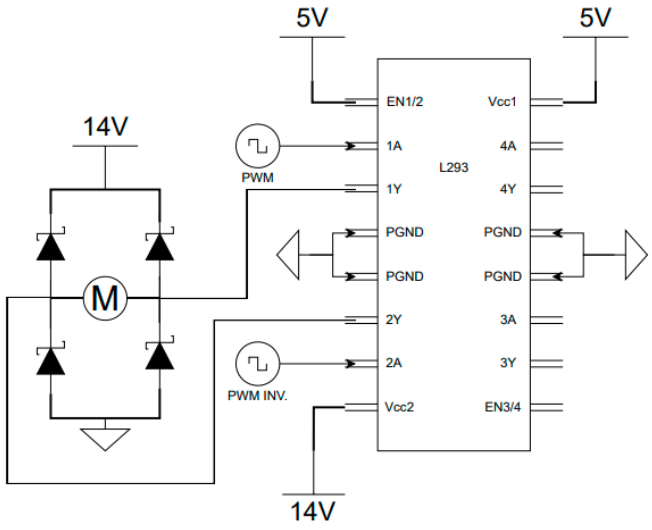

Figura 3- Esquema da Interface L293NE

É importante assinalar que este motor possui duas indutâncias, que são essenciais para a seu funcionamento, contudo, quando houver uma variação brusca de corrente, a tensão aos terminais das bobinas do motor aumenta consideravelmente, o que pode danificar o restante *hardware.*

Esta é a justificação para a utilização de dois díodos BA157 para cada saída da ponte-H. Assim sempre que a tensão da bobina atingir valores indesejáveis (V > 14 ou V < 0), um dos díodos ficará diretamente polarizado, limitando a esse valor de tensão. As características do BA157 são indicadas para esta aplicação, uma vez que é um díodo de *fast switching* (de acordo com o *datasheet*: *Maximum reverse recovery time* = 150 ns), e suporta correntes altas.

À entrada do L293 estão ligados dois sinais *PWM*, com um desfasamento de 180º entre eles. Para conseguir esse atraso de fase, recorremos ao transístor TN0702 (DMOS-FET). Este FET é indicado para esta aplicação uma vez que, tem uma tensão de *threshold* baixa (0.5V < Vth < 1V), ou seja, a tensão VGS mínima para que o transístor conduza é igualmente baixa. Além disso, este é um transístor rápido com tempos de subida e descida baixos (ambos 20 ns).

O FET foi montado numa configuração fonte-comum, como se pode ver na figura 4:

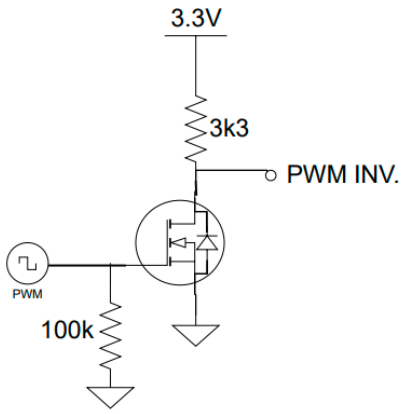

Figura 4- Circuito Inversor

O *PWM* gerado pelo módulo OC1 da PIC32 é ligado à porta do transístor, e a sua versão invertida é retirada do dreno. A resistência de *pull-up* usada foi de 3k3 Ω, já que ao alimentar este circuito a 3.3 V, a corrente de dreno seria 1 mA (de acordo com o *datasheet* é a condição para um valor típico de Vth de 0.8V). A resistência de 100 kΩ foi usada para que a porta do transístor fique no nível lógico '0' quando o *PWM* é desligado.

Implementação do *Software*

A parte mais desafiante deste projeto foi a implementação de *Software* e *Firmware*. Começamos por definir os timers.

Para assegurarmos a frequência do *PWM* que nos é pedida (20 kHz) utilizamos o Timer 2, para definirmos a frequência de amostragem utilizamos o Timer 3. Vamos utilizar o Timer 5 e Timer 4 para melhorar os resultados do nosso sistema, como iremos abordar mais abaixo.

O nosso objetivo seguinte foi conseguir obter a velocidade à qual o motor estava a girar. A nossa abordagem inicial foi criar uma variável (que estava dentro de uma rotina de interrupção) que incrementa sempre que há um impulso no *encoder*. O cálculo da rotação é depois feito multiplicando esta variável por 60 (para ter rotações em minutos), multiplicando pela frequência de amostragem (10 Hz), dividido pelo número de pulsos numa volta. Escolhemos 10 Hz, para a frequência de amostragem, uma vez que se a frequência for muito maior prejudicaria o correto funcionamento do controlador PI.

Com o cálculo das rotações funcional, prosseguimos para o controlador PI. A nossa estratégia para a sintonia do controlador PI foi aumentar a constante proporcional até o controlador ficar instável (Kp = 2.4) e usamos metade desse valor. Quanto à integral, fomos aumentando gradualmente até obtermos o melhor desempenho do controlador.

Uma das desvantagens do método da contagem de pulsos é que para velocidades maiores induz um erro que é notável. A solução que decidimos implementar foi contar o tempo entre cada pulso do *encoder*. utilizamos Este método só é viável neste projeto, uma vez que estamos a usar velocidades relativamente baixas. Se as velocidades fossem superiores o melhor método seria o anterior, sendo essa a sua grande vantagem.

Para conceber o método de o Timer 4 para poder contar o tempo entre cada impulso, sempre que há um impulso no canal do *encoder.* O valor do registo TMR4 é depois colocado outra vez a 0 para começar a contar o impulso seguinte. A expressão para o cálculo da velocidade é:

$$rpm = \left(\frac{60}{\frac{newTMR}{freqPrescaller}} * 420\right)$$

Sendo que *newTMR* é o valor do registo TMR4, ou seja, o tempo entre cada impulso, freqPrescaller é a frequência à saída do divisor de frequência do timer, e 420 é o número de impulsos numa rotação do veio do motor.

O Timer 5 é utilizado para o refrescamento da interface ~~gráfica.~~ de utilizador

Resultados

A figura 5 representa a resposta do motor a três velocidades diferentes: 10, 30 e 50 RPM. As retas horizontais representam os *SetPoints*.

Estes valores é com ou sem controlador PI?

Começa com velocidade=10?

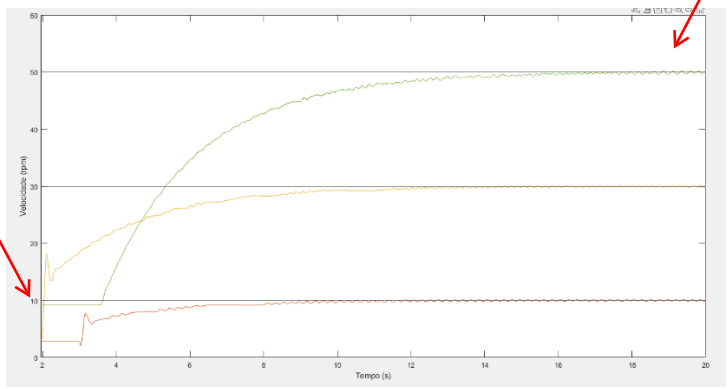

Figura 5- Resposta do Sistema a 10 RPM (Vermelho), 30 RPM(Amarelo), 50 RPM(Verde)

Na figura 6 está representado o gráfico da resposta do motor a 30 RPM quando é sujeito a uma perturbação de carga. com ou sem controlador?

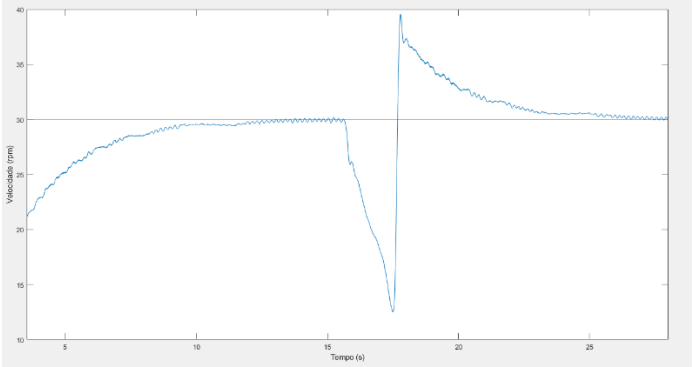

Figura 6 - Resposta do Sistema para 30 RPM quando sujeito a perturbação de carga

Os impulsos do *encoder* com o motor a rodar nos dois sentidos, estão representados na figura 8. A figura que se segue representa o *PWM* gerado pela PIC32 (amarelo) e a sua versão invertida (azul).

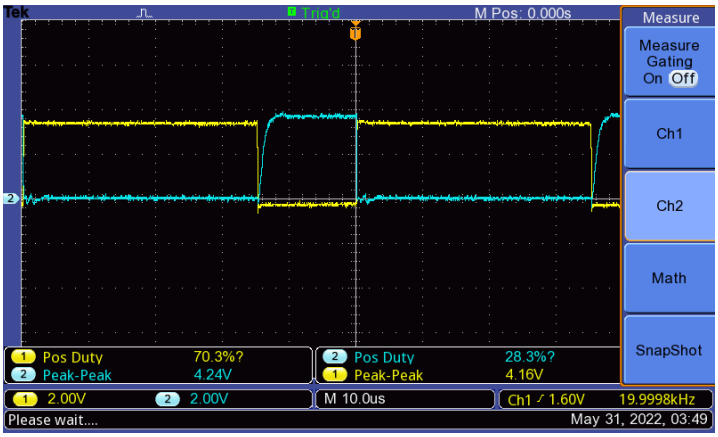

Figura 7 – PWM invertido e não-invertido

Os impulsos do *encoder* com o motor a rodar nos dois sentidos, estão representados na figura 8.

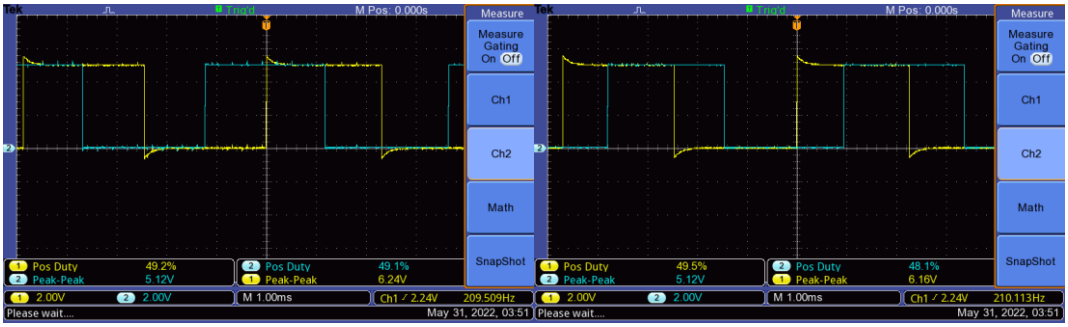

Figura 8 – Impulsos no enconder (Ch.A e Ch.B)

Análise dos Resultados

Como foi apresentado na figura 5, a velocidade do motor atinge um valor final muito próximo do valor definido no *setpoint.* Contudo a resposta transitória do controlador é lenta, o que se deve à incapacidade do controlador PI reagir à derivada do erro (para melhores resultados poderíamos utilizar um controlador PID, embora isso não seja exigido neste trabalho). Apesar disso, o tempo do regime transitório do controlador PI poderia ser melhorado se usássemos um critério de seleção das constantes proporcional e integral mais robusto, ou simplesmente diminuindo a constante de integração, sendo que isso poderia levar a que a resposta do motor apresentasse alguma sobre-elevação de RPM. Não havendo quaisquer especificações sobre o tempo transitório, optamos por uma constante de integração mais alta.

A tabela um contém os tempos da reposta transitória do motor para os três valores de velocidade mencionados anteriormente, assim como o valor do *ripple* em estado estacionário

Setpoint(RPM)	Tempo Transitório (s)	*Ripple* em estado estacionário (RPM)
10	9	0.15
30	11	0.21
50	14	0.30

Tabela 1- Resposta do Controlador PI para várias velocidades

Para comprovarmos o bom funcionamento do controlador, realizamos o teste de aplicação de uma perturbação de carga no motor. Naturalmente, ao fazer isso o valor da velocidade do motor varia abruptamente. Quando esse efeito desaparece, o controlador PI deverá ser capaz de corrigir o valor da velocidade, como é possível ver na figura 6, onde usamos o exemplo de uma velocidade de 30 RPM.

Na figura 7 mostramos que o circuito inversor de *PWM* funciona corretamente, embora seja de notar que existe algum tempo de subida na versão invertida. Podemos ainda verificar que a especificação da frequência do sinal de PWM a 20 kHz foi cumprida.

Para provarmos que a mudança de direção do motor é feita corretamente, analisamos os impulsos do *encoder* que, como referido anteriormente, têm um atraso de 90º entre eles. A primeira imagem da figura 8 apresenta os impulsos do *encoder* (com velocidade de 30 RPM) a rodar no sentido dos ponteiros do relógio. Ao carregar na tecla 'd' na interface gráfica, os impulsos do canal que estava adiantado 90º, fica agora atrasado (e vice-versa), como pode ser visto na segunda imagem.

Conclusões

Ao analisarmos os resultados obtidos, podemos dizer que atingimos os principais requisitos com sucesso:

- Implementamos circuitos adequados para a implementação deste sistema;
- Conseguimos regular a velocidade do motor dentro da gama requerida (10-50 RPM);
- Implementamos o sistema de controlo, recorrendo a um controlador PI;
- O erro estacionário do sistema é inferior ao que é exigido nas especificações (1 RPM);
- Conseguimos visualizar na interface gráfica o valor atual da velocidade em RPM e da posição angular. A interface gráfica permite também parar e mudar a direção do motor;
- Conseguimos interagir com o instrumento de medida disponível: o *encoder*. O que nos permitiu determinar o valor da velocidade e do ângulo.

Apesar disso, achamos que existem pontos a ser melhorados. Como já referido, a resposta transitória do controlador PI poderia ser melhorada, caso tivéssemos implementado um método de sintonia mais indicado do que aquele que foi descrito acima.

Referências

[1] "Eletronic Instrumentation Systems Laboratory projects guide", Electronic Instrumentation Systems

[2] Mota, Alexandre, "Controladores PID", Beta Version 0.1, 2018.

[3] Cytron Technologies (2011). Datasheet SPG30E-60K.

[4] Texas Instruments (2016). Datasheet L293.

[5] Vishay (2002). Datasheet BA157.

Anexo

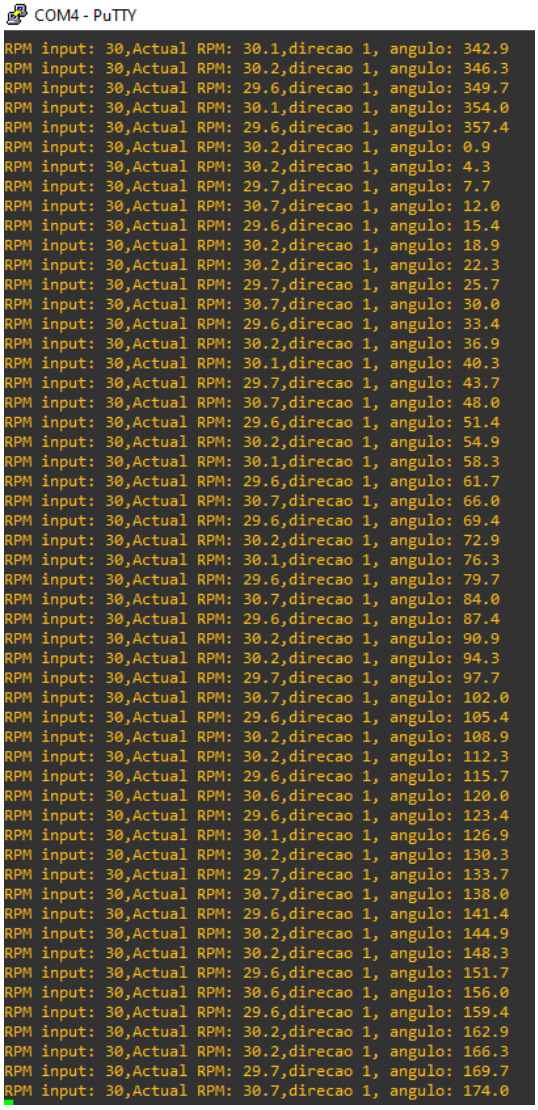

Figura 8 - Interface gráfica quando é colocado um setpoint de 30 RPM