Luís Gonzaga dos Reis Ramos

Licenciado em Ciências da Engenharia Mecânica

Desenvolvimento de um Sistema de Deslocamento Bi-axial para Aplicação em Túnel Aerodinâmico

Dissertação para Obtenção do Grau de Mestre em Engenharia Mecânica

Orientador:	Doutor Luís Miguel Chagas da Costa Gil, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente:	Doutor Daniel Cardoso Vaz, Professor Auxiliar da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa
Arguente:	Doutor Telmo Jorge Gomes dos Santos, Professor Associado da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa
Vogal:	Doutor Luís Miguel Chagas da Costa Gil, Professor Auxiliar da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2018

Luís Gonzaga dos Reis Ramos

Licenciado em Ciências da Engenharia Mecânica

Desenvolvimento de um Sistema de Deslocamento Bi-axial para Aplicação em Túnel Aerodinâmico

Dissertação para Obtenção do Grau de Mestre em Engenharia Mecânica

Orientador:	Doutor Luís Miguel Chagas da Costa Gil, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente:	Doutor Daniel Cardoso Vaz, Professor Auxiliar da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa
Arguente:	Doutor Telmo Jorge Gomes dos Santos, Professor Associado da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa
Vogal:	Doutor Luís Miguel Chagas da Costa Gil, Professor Auxiliar da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Setembro, 2018

Desenvolvimento de um Sistema de Deslocamento Bi-axial para Aplicação em Túnel Aerodinâmico

Aos meus pais

AGRADECIMENTOS

Agradeço ao meu orientador, o Professor Luís Gil, pela proposta do tema da presente dissertação. Este tema expandiu o meu conhecimento nas mais diversas áreas, e tornou-me uma pessoa mais polivalente. Tive a oportunidade de passar pelo processo de projeto, fabricação e teste de uma máquina, e terminar a dissertação tendo construído uma máquina funcional, com uma utilidade verdadeira.

Agradeço ainda, de modo geral, aos professores que me acompanharam ao longo do meu percurso académico, muitos dos quais possuem uma capacidade de ensinar e de cativar o aluno excecional.

Agradeço, por fim, aos meus pais e aos meus amigos por todo o apoio ao longo destes anos.

RESUMO

Neste documento é relatado o processo de construção de um engenho que, com recurso a duas mesas lineares e dois motores de passo a passo, se movimenta segundo dois eixos. A esta montagem foi desenvolvido e acoplado um suporte para um tubo de Pitot, mas podem ser facilmente adaptados outros instrumentos de medição. A utilização pretendida para a montagem é a medição de pressões/velocidades nos túneis aerodinâmicos da FCT.

Foram relatados vários aspetos relevantes ao desenvolvimento da montagem, como a escolha e adaptação de uma fonte de alimentação, o funcionamento dos *drivers* controladores dos motores passo a passo, a montagem dos sensores de fim de curso nos sítios devidos, o desenvolvimento de uma caixa para guardar e proteger os componentes e garantir uma interface com o utilizador, entre outros aspetos.

Foi também relatado o desenvolvimento do código para a placa controladora Arduino. Este *software* permite a inserção de diversas coordenadas na memória do Arduino e o posterior controlo do posicionamento da mesa através de dois botões que avançam para a posição seguinte ou voltam à anterior. Alternativamente, as posições pretendidas podem ser inseridas manualmente no computador, através da janela de comunicação *serial* do Arduino.

O funcionamento da montagem foi validado com sucesso através de um teste ao posicionamento da mesa, e através da medição de velocidades à saída de uma conduta, que permitiram determinar perfis de velocidade à saída da conduta, bem como valores de caudal.

Palavras Chave

- Mesa Linear
- Motores Passo a Passo
- Arduino
- Túnel Aerodinâmico

ABSTRACT

In this master thesis document the process of construction of a machine is described. This machine uses two linear modules and two stepper motors in order to move along two axis. A mount for a Pitot tube was made and coupled to this machine, but other measurement devices can easily be connected as well. The usage case scenario for this machine is pressure/velocity measurement in FCT wind tunnels.

Various aspects that were relevant to the development of the machine were described, such as the choice and customization of the power supply, the workings of the stepper motor drivers, the mounting of the end stop switches in the correct places, the development of an adequate case for the storage and protection of the components and placement of the user interface buttons, amongst other aspects.

Also described was the development of the software for the control devide, an Arduino. This software allows for the input of various coordinate values in the Arduino's memory, and later the control of the linear module's positioning through two buttons that advance it to the next position or return it to the previous position. Alternatively, a position value can be manually typed in the Arduino's serial monitor window.

The machine's correct working was validated by a test to the positioning of the linear modules, and also through the measurement of velocity values at the end of a duct, which allowed for the finding of velocity profiles at the end of the duct, as well as volume flow rate values.

Keywords

- Linear Module
- Stepper Motors
- Arduino
- Wind Tunnel

ÍNDICE DE CONTEÚDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

NOMENCLATURA

Símbolos latinos

$\boldsymbol{I}$	Corrente elétrica $[A]$
$\boldsymbol{L}$	Indutância $[H]$
$\boldsymbol{P}$	Potência $[W]$
$\boldsymbol{Q}$	Caudal volúmico $[m^3 \cdot s^{-1}]$
$\boldsymbol{R}$	Resistência elétrica $[\Omega]$
$\boldsymbol{Re}$	Número de Reynolds
$\boldsymbol{t}$	Tempo $[s]$
$\boldsymbol{T}$	Binário $[N \cdot m]$
$\boldsymbol{U}$	Diferença de potencial $[V]$
$\boldsymbol{V}$	Velocidade $[m \cdot s^{-1}]$

Símbolos gregos

μ	Viscosidade $[Pa \cdot s]$
ω	Velocidade angular $[rad \cdot s^{-1}]$
ρ	Massa volúmica $[kg \cdot m^3]$

Siglas

ATX	*Advanced Technology eXtended*
ADC	*Analog-to-digital converter*
DC	*Direct Current*

DEMI	Departamento de Engenharia Mecânica e Industrial
FCT	Faculdade de Ciências e Tecnologia
LED	*Light emitting diode*
MDF	*Medium density fiberwood*
NEMA	National Eletric Manufacturers Association
PC	*Personal Computer*
UNL	Universidade Nova de Lisboa
USB	*Universal Serial Bus*

1. INTRODUÇÃO

1.1. Contexto

O túnel aerodinâmico, inventado em 1871, foi inicialmente usado no ramo da aeronáutica no estudo do escoamento de ar sobre superfícies. Revelou-se um equipamento de grande utilidade, tendo sido conseguidos diversos avanços técnicos, mesmo antes do primeiro voo tripulado a motor (que só se viria a realizar em 1903).

Atualmente a utilidade dos túneis aerodinâmicos não se restringe à industria aeronáutica. São utilizados por construtores automóveis para estudar o escoamento sobre a carroçaria do automóvel e para avaliar o desempenho de componentes em condições climáticas adversas. São utilizados, também, no estudo do impacto do vento em edifícios e outras grandes construções, medindo efeitos de ressonância em estruturas altas, ou a poluição sonora perto de edifícios.

Com os rápidos avanços na tecnologia, a simulação computacional tornou-se uma ferramenta essencial para a dinâmica dos fluidos, e os túneis aerodinâmicos têm vindo a ser utilizados em conjunto com ferramentas computacionais, sendo frequentemente instrumentados com sistemas de aquisição e análise de dados, e a sua operação por vezes computorizada.

Estes sistemas, analógicos ou digitais, são normalmente instrumentos especializados, projetados para desempenhar uma função específica. Por esta razão têm, normalmente, um custo elevado.

Existem, no entanto, plataformas como o Arduino e o Rasbery Pi que se apresentam como placas de baixo custo com um elevado poder de processamento, e uma grande versatilidade de ligação com outros componentes, analógicos ou digitais. São, por isso, adequadas como alternativas a determinados sistemas mais caros.

1.2. Objetivos

O objetivo da dissertação pode-se resumir a:

- Projetar e construir uma montagem a ser colocada nos túneis aerodinâmicos existentes no Departamento de Engenharia Mecânica e Industrial da FCT-UNL, numa determinada secção destes, e capaz de deslocar um sensor (tubo de Pitot, anemómetro de fio quente, anemómetro laser-Doppler ou equipamento semelhante) para qualquer ponto na secção escolhida.

Existem, ainda, outros objetivos que se enquadram no principal. Estes são:

- Efetuar o controlo do posicionamento do sensor por via eletrónica, recorrendo a uma placa Arduino ou a um equipamento de aquisição digital da National Instruments.

- Utilizar motores passo a passo e mesas lineares adquiridas com a intenção de serem utilizados nesta montagem.
- Projetar uma base para a montagem com verificação e ajuste de nivelamento, de modo a garantir o correto posicionamento do sensor na secção do túnel.
- Garantir uma fácil interface com o utilizador.
- Testar a montagem, de modo a verificar o seu correto funcionamento.

1.3. Motivação

Este trabalho revelou-se uma oportunidade de acrescentar à secção de Dinâmica dos Fluidos e Termodinâmica Aplicada do DEMI da FCT-UNL um equipamento que tornaria os túneis aerodinâmicos da faculdade mais úteis, ao permitir efetuar medições que antes não eram possíveis ou eram complicadas de realizar.

Era também uma oportunidade de atualizar o túnel, ao acrescentar-lhe equipamento actual, que utilizaria componentes e *software* modernos.

As placas NI USB-6008 já tinham sido utilizadas noutros projetos do DEMI, nomeadamente em máquinas de impressão 3D, pelo que o seu funcionamento era conhecidos por vários alunos do departamento.

De um ponto de vista pessoal, o autor tem um elevado interesse por eletrónica, mecânica e projetos *do it yourself* de um modo geral, e o trabalho permitiu aprofundar conhecimentos nestas áreas.

1.4. Estrutura da dissertação

A presente dissertação encontra-se dividida em cinco capítulos.

No primeiro e presente capítulo aborda-se o contexto, objetivos, motivação e estrutura da dissertação.

No segundo capítulo é apresentada a revisão bibliográfica efetuada pelo autor, que começa com um resumo da história do túnel aerodinâmico e suas utilizações. É abordada a teoria de funcionamento do tubo de Pitot, um instrumento de medição de velocidade pontual num escoamento, e alguns aspetos importantes referentes à sua utilização.

São ainda analisadas neste capítulo as características e funcionamento de um motor passo a passo, bem como o *microstepping*, um modo de funcionamento do motor implementado pelo seu *driver* que permite atingir mais posições angulares do motor, bem como uma rotação mais suave.

No capítulo terceiro é descrito o desenvolvimento da montagem. São abordadas algumas características relevantes do equipamento utilizado, e é descrita a escolha e modificação da fonte de alimentação. É também descrita a fabricação de alguns elementos, como a caixa para os componentes e os apoios para os sensores de fim de curso, e são apresentadas as ligações elétricas entre componentes. É dedicado um sub-capítulo ao código do Arduino que foi desenvolvido para esta aplicação.

No quarto capítulo descrevem-se os testes a que a montagem foi sujeita. No primeiro foi medido o posicionamento das mesas lineares após esta ter percorrido uma malha de posições. No segundo são efetuados atravessamentos à saída de uma conduta. É descrito o procedimento experimental utilizado na medição dos vários valores de velocidade na conduta, e são apresentados os valores obtidos e o seu respetivo tratamento, que permitiu retirar conclusões sobre o escoamento estudado. Estas conclusões são apresentadas e discutidas.

O último capítulo contém a conclusão, onde é feita uma análise crítica do trabalho desenvolvido e da montagem em si, e onde são propostos trabalhos futuros relacionados com a montagem.

2. REVISÃO BIBLIOGRÁFICA

Na presente revisão bibliográfica abordam-se alguns temas relevantes para a dissertação.

Começa-se por descrever a história do túnel aerodinâmico e a sua utilização atual, como forma de apresentar a evolução e a utilidade deste equipamento.

A componente experimental da dissertação tem como único objetivo testar a montagem desenvolvida, e como tal abordou-se brevemente o funcionamento do tubo de Pitot, a correta utilização deste instrumento de medição e o cálculo de caudais por atravessamento.

É abordada, por fim, a teoria de funcionamento dos motores passo a passo, o componente que irá impor movimento à mesa linear, cujo complexo funcionamento é descrito em vários sub-sub-capítulos.

2.1. Breve história do túnel aerodinâmico

Após várias tentativas de voo tripulado falhadas, os pioneiros da aviação aperceberam-se da necessidade de estudar o escoamento de ar sobre a superfície de um planador num ambiente controlado.

Primeiro foram feitos estudos no exterior, em locais onde a velocidade do vento fosse estável e conhecida, e posteriormente construiram-se engenhos que colocavam o objeto em estudo em movimento de rotação, à volta de um eixo fixo ao solo, impondo-lhe uma velocidade pré-estabelecida [1]. Estas máquinas remontam ao séc. XVIII.

A invenção do túnel aerodinâmico é creditada a Frank H. Wenham (1824-1908), engenheiro naval britânico e membro da Sociedade Aeronáutica da Grã-Bretanha. O seu engenho, construído em 1871, consistia num ventilador colocado em rotação por um motor a vapor, que movia ar por uma passagem quadrada com cerca de 45 cm de lado e 3,5 m de comprimento.

Os testes conduzidos por Wenham no seu túnel permitiram vários avanços teórico-práticos, entre eles eles a definição de alongamento de uma asa[1], e a relação entre a força de sustentação e o alongamento.

Apesar destes avanços, o famoso voo tripulado a motor dos irmãos Wright só se realizaria em dezembro de 1903, 32 anos após a invenção do túnel aerodinâmico. Também os irmãos Wright recorreram a este equipamento, tendo construído dois túneis (na figura 2.1 observa-se a recriação de um deles). Recorrendo a sensores com um funcionamento semelhante a dinamómetros, efetuaram medições de sustentação em perfis alares relativamente a outros perfis alares calibrados, o que lhes permitia uma análise comparativa célere do desempenho dos perfis [1].

2.1.1 Uso atual

Atualmente os túneis aerodinâmicos são ferramentas de grande utilidade para diversos ramos de engenharia, não sendo o seu uso restringido apenas à aeronáutica.

[1] Alongamento: relação entre a envergadura e a corda de uma asa.

Figura 2.1: Recriação de um dos túneis aerodinâmicos construídos pelos irmãos Wright [2]

São utilizados pelos construtores automóveis não só para estudar o escoamento pela carroçaria do automóvel (recorrendo a modelos, ou a protótipos à escala real), mas também para avaliar o desempenho de componentes em condições climáticas diversas em túneis climáticos (figura 2.2), desenvolvidos exclusivamente para este efeito.

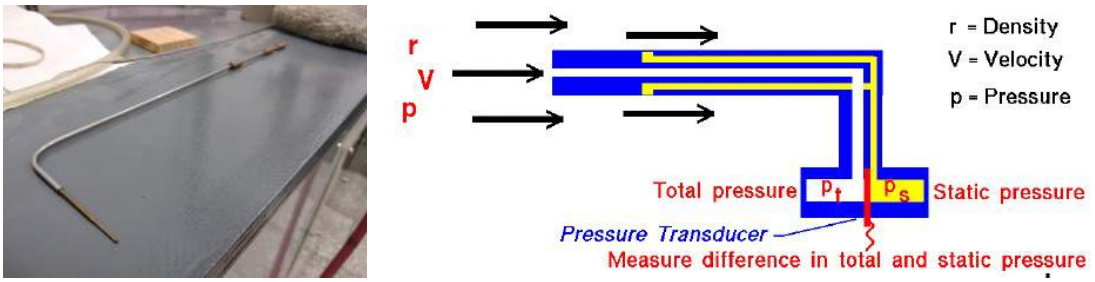
Figura 2.2: Teste em túnel climático [3]

São também utilizados no estudo do impacto do vento em construções, medindo o impacto de diferentes velocidades do vento na resistência estrutural do edifícios, medindo efeitos de ressonância ou desprendimento de vórtices em estruturas altas e pontes, medindo a poluição sonora perto de edifícios, entre outras utilizações [4].

Com os rápidos e consideráveis avanços informáticos, a simulação computacional tornou-se uma ferramenta essencial para a dinâmica dos fluidos. Os túneis aerodinâmicos são agora utilizados em conjunto com as ferramentas computacionais, validando a qualidade dos resultados obtidos por esta via. Para tal, os túneis são frequentemente atualizados com instrumentos de aquisição e análise de dados, e a sua operação é computorizada [5].

2.2. Tubo de Pitot e sua utilização

O tubo de Pitot, inventado pelo engenheiro francês Henri Pitot em 1732 e representado na figura 2.3a, é um instrumento utilizado para medir a velocidade do escoamento sobre uma área muito pequena (considerada pontual). Um instrumento semelhante é o tubo de Prandtl ou Pitot-estático, cuja representação se encontra na figura 2.3b. Na literatura usa-se frequentemente o nome "tubo de Pitot"para ambos os instrumentos.



(a) (b)

Figura 2.3: Tubo de Pitot (a) e representação esquemática do tubo de Prandtl [6] (b)

Ambos são tubos de pequeno diâmetro, posicionados de modo a ficarem alinhados com o escoamento. Os orifícios na parede do tubo de Prandtl, perpendiculares ao tubo central, medem a pressão estática, p_s. O orifício a montante mede a pressão total, ou pressão de estagnação p_0, porque a velocidade do fluido é reduzida até zero neste orifício. A pressão total é igual é a soma da pressão estática com a pressão dinâmica:

$$p_0 = p_s + p_d \tag{2.1}$$

Em vez de medir p_0 e p_s separadamente recorrendo a manómetros ligados à outra extremidade do tubo, pode-se recorrer a um transdutor diferencial de pressão e medir a pressão dinâmica diretamente (como representado na figura 2.3b).

A expressão 2.4, que permite calcular a velocidade a partir da pressão, é obtida da equação de Bernoulli para regime permanente, incompressível e sem atrito (equação 2.2), ao longo da linha de corrente da figura 2.4.

$$\frac{p_1}{\rho} + \frac{1}{2}V_1^2 + gz_1 = \frac{p_2}{\rho} + \frac{1}{2}V_2^2 + gz_2 = c^{te} \tag{2.2}$$

$$p_s + \frac{1}{2}\rho V^2 + \rho g z_s = p_0 + \frac{1}{2}\rho(0)^2 + \rho g z_0 \tag{2.3}$$

$$V = \sqrt{2\frac{(p_0 - p_s)}{\rho}} \tag{2.4}$$

2.2.1 Cuidados, vantagens e desvantagens

Uma desvantagem do tubo de Pitot é que este deve estar alinhado na direção do escoamento, que pode não ser conhecida. No entanto, para ângulos de desvio até 5°, os erros de medição são desprezáveis(figura 2.5).

Devido à resposta lenta dos tubos que fazem a ligação aos manómetros, o tubo de Pitot não é útil em regime transitório [7]. Não é, igualmente, útil em gases a baixa velocidade devido às pequenas diferenças de pressão envolvidas. Por exemplo, para ar a 1 km/h, substituindo os

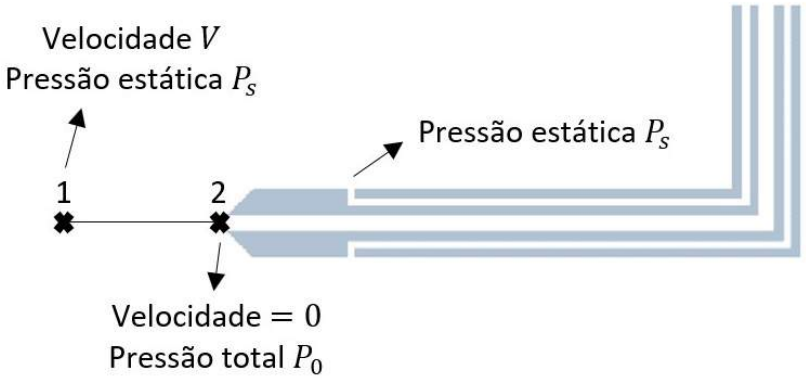


Figura 2.4: Diagrama da montagem de um tubo de Pitot

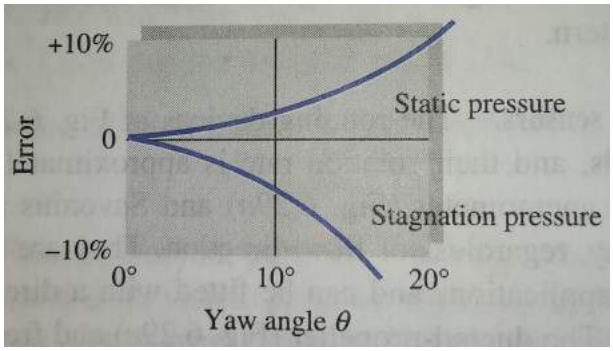


Figura 2.5: Erro de medição em função do ângulo de desvio [7]

valores devidos na equação 2.4, obtém-se $p_0 - p_s = 0{,}047\,\text{Pa}$, valor fora da resolução da maioria dos manómetros.

2.2.2 Medição de caudal por atravessamento

O caudal volúmico é a quantidade, em volume, de fluido que atravessa uma determinada superfície por unidade de tempo.

Supondo que a superfície S da figura 2.6a é uma malha imaginária pela qual o fluido atravessa sem resistência. Se a velocidade V variar, é necessária a integração na superfície elementar dA para o cálculo do caudal. Seja θ o ângulo entre V e n, que corresponde ao vetor normal a dA. O volume de fluido, representado na figura 2.6b, que atravessa dA no intervalo de tempo dt corresponde a:

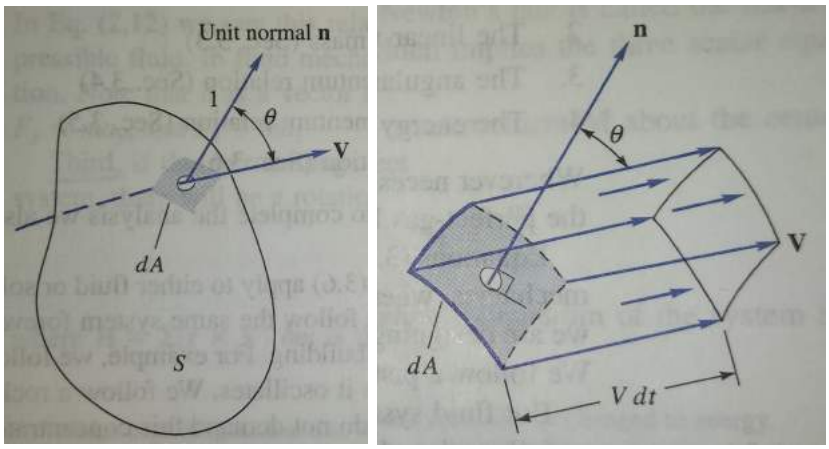
$$d\mathscr{V} = V\ dt\ dA\ cos\theta = (V \cdot n)\ dA\ dt \tag{2.5}$$

E o caudal volúmico Q que atravessa a superfície S é o integral de $d\mathscr{V}/dt$:

$$Q = \int_s (V \cdot n)\ dA = \int_s V_n\ dA \tag{2.6}$$

Em que o produto interno $V \cdot n$ é equivalente a V_n, a componente da velocidade normal a dA. O uso do produto vetorial permite que Q assuma valores positivos ou negativos, permitindo inferir o sentido do escoamento.

Numa medição de caudal por atravessamento, são medidos valores de velocidade V_i em pontos convenientemente selecionados de uma conduta ou canal. A superfície de passagem do fluido é dividida em áreas discretas, A_i, de modo a que a velocidade do fluido possa ser considerada



(a) (b)

Figura 2.6: Caudal volúmico através de uma superfície: unidade elementar de área dA (a) e elemento de volume de fluido que atravessa a unidade elementar num intervalo de tempo dt (b)

constante em qualquer ponto da área. O cálculo do caudal é, efetivamente, uma aproximação do integral da equação 2.6:

$$Q = \sum_i V_i A_i \tag{2.7}$$

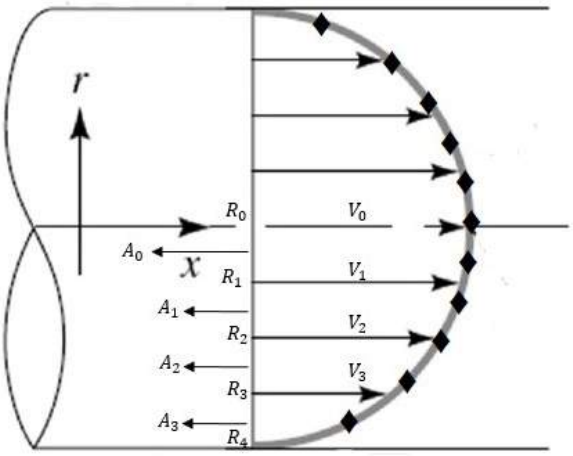


Figura 2.7: Determinação de caudal por varrimento

Na figura 2.7 observa-se um perfil de velocidades de um escoamento à saída de um tubo de secção circular, que foi dividido em coroas circulares, A_i, fazendo-se corresponder a cada coroa uma velocidade V_i, medida experimentalmente. Uma possível primeira aproximação (por excesso) do valor do caudal à saída do tubo será:

$$Q = \sum_i V_i \ \pi(R_{i+1}^2 - R_i^2) \tag{2.8}$$

2.3. Motores de passo a passo

Um motor passo a passo pode ser caracterizado como um dispositivo de potência constante, onde a potência é definida como o produto do binário pela velocidade(eq. 2.9). Este tipo de motores assume um determinado número de posições angulares distintas (duzentas, tipicamente), tendo uma precisão elevada no seu posicionamento.

$$P = T \times \omega \tag{2.9}$$

2.3.1 Motor ideal

Um motor passo a passo ideal não teria fricção mecânica e o seu binário seria proporcional ao número de espiras no estator multiplicado pela corrente que atravessa as referidas espiras. A sua única característica elétrica seria a indutância [2].

Num motor ideal, à medida que a sua velocidade se aproxima de zero, o seu binário aproximar-se-ia do infinito, e a velocidade infinita teria binário igual a zero. Como a corrente é proporcional ao binário, a corrente seria infinita a velocidade zero.

2.3.2 Motor real

Um motor real difere de um ideal por ter resistência não nula nas espiras. Adicionalmente, as placas de ferro estão sujeitas a saturação magnética, a perdas por corrente de Foucault e a perdas de histerese entre a variação de força de magnetização (corrente) e a indução magnética (densidade de fluxo magnético) [8].

A saturação magnética impõe um limite na proporcionalidade inversa entre a corrente e o binário, enquanto que a resistência das espiras e as perdas no ferro (por corrente de Faulcault e histerese) provocam o aquecimento do motor.

2.3.3 Curva velocidade-binário

Cada motor passo a passo tem especificado um valor máximo de corrente que não pode ser excedido. Como a corrente é inversamente proporcional à velocidade, para baixas velocidades a corrente tende para valores altos. Cabe ao *driver* do motor limitar o valor máximo de corrente. Como consequência, o binário fica também limitado até à *corner speed*, velocidade a partir da qual o valor de corrente começa a diminuir (figuras 2.8 e 2.9).

2.3.4 Perdas devido a *detent torque*

Uma característica que difere um motor passo a passo de outros motores (particularmene servomotores), é a sua capacidade de bloquear o rotor quando parado, até um certo binário [9]. Na figura 2.9 observa-se que para velocidade igual a zero, o valor do binário mantém-se igual ao valor máximo especificado, o que significa que um agente externo pode aplicar no rotor um binário inferior ao especificado sem que o veio altere a sua posição. O motor só assume este comportamento se estiver a ser alimentado.

Mesmo desligado, no entanto, o rotor possui uma resistência ao movimento, causada pelas atrações magnéticas internas. Ao rodar o veio de saída manualmente com o motor desligado é possível sentir uma resistência em forma pulsada. O binário necessário para vencer esta resistência é chamado *detent torque*.

[2] Indutância: propriedade de um condutor em que uma variação da corrente elétrica através dele induz uma força eletromotriz (diferença de potencial) neste.

Este binário representa uma perda quando o motor roda (independentemente do sentido), e a perda de potência devido a este efeito é proporcional à velocidade de rotação. Assim, esta perda deve ser subtraída à curva ideal. Na figura 2.10 observam-se as variações de potência e binário de um motor em função da sua velocidade, com e sem a subtração das perdas por *detent torque*.

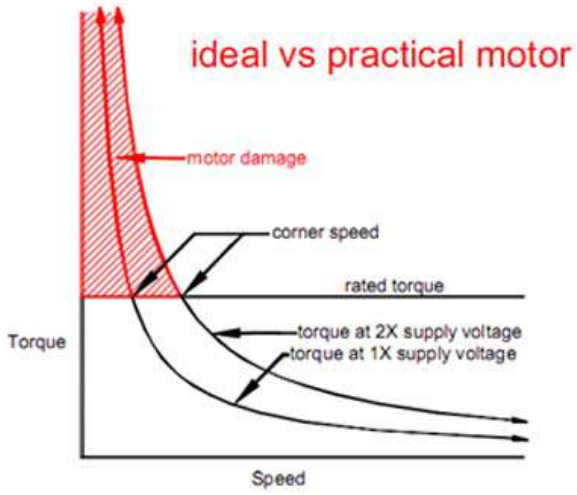


Figura 2.8: Curva velocidade-binário de um motor passo a passo [10]

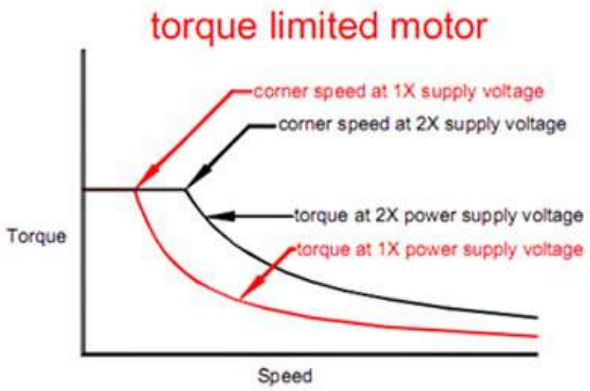


Figura 2.9: Curva de um motor limitado em corrente [10]

2.3.5 Escolha de valores de tensão e corrente

A potência e binário de um motor passo a passo é diretamente proporcional à tensão com que é alimentado [10], a partir da zona em que a corrente não é limitada pelo *driver* (figura 2.11). A tensão de funcionamento dos motores passo a passo não é indicada nas especificações, e deve-se sempre tomar em conta o aumento do calor dissipado resultante do aumento da tensão. A equação 2.10, obtida empiricamente [10] , sugere um limite máximo para a tensão da fonte de alimentação.

$$U_{\text{Max}} = 32 \times \sqrt{L} \tag{2.10}$$

Onde L é a indutancia do motor, normalmente indicada nas especificações deste.

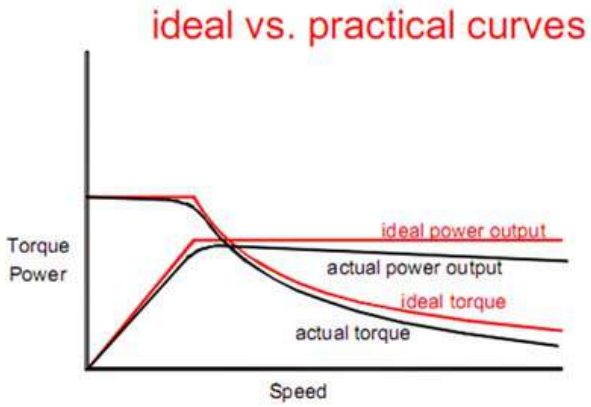


Figura 2.10: Curvas de funcionamento práticas de um motor passo a passo [10]

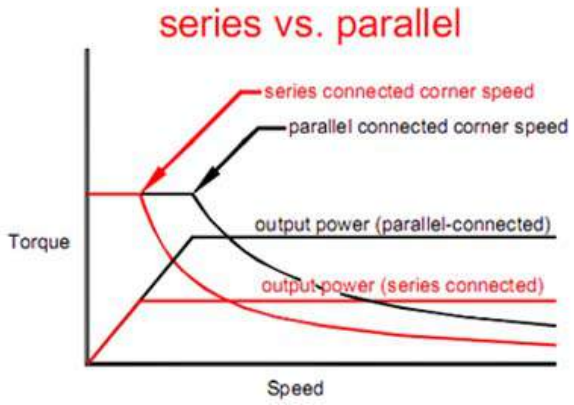


Figura 2.11: Influência do aumento de tensão no funcionamento de um motor passo a passo [10]

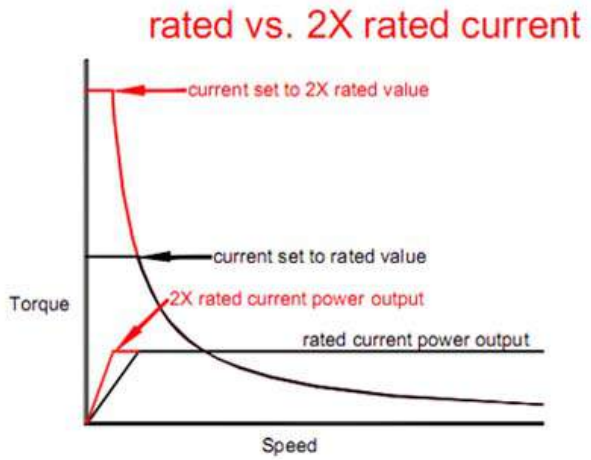


Figura 2.12: Influência do aumento do limite de corrente no funcionamento de um motor passo a passo [10]

O aumento da corrente limite aumenta o binário disponível a baixas velocidades de rotação. Na figura 2.12 observa-se o aumento da corrente limite para o dobro. O binário não iria, no entanto, duplicar, devido à saturação magnética [10]. O motor iria, também, dissipar consideravelmente mais calor. O valor de corrente máximo é sempre indicado nas especificações, e deve

ser respeitado.

2.3.6 Tipos de motores

Consoante o tipo de enrolamento, os motores passo a passo classificam-se em unipolares ou bipolares.

Os motores unipolares operam com dois enrolamentos e têm ligações elétricas ao centro de cada enrolamento. Consoante o modo como esta ligação ao centro é feita, os motores unipolares podem ser fabricados com cinco, seis ou oito fios de ligação, como representado na figura 2.13.

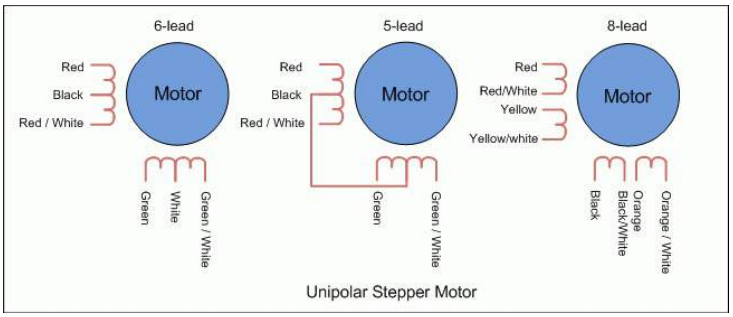


Figura 2.13: Tipos de ligações elétricas dos motores unipolares [11]

Os motores bipolares (figura 2.14) possuem também dois enrolamentos, mas não têm a ligação elétrica ao centro destes.

Os motores bipolares requerem drivers mais complexos, que incorporem circuitos *H-bridge* cuja função é reverter o sentido da corrente num enrolamento. Num motor unipolar, as secções do enrolamento funcionam alternadamente para atingir este efeito, não sendo necessários drivers tão complexos [11]. Como consequência, os motores bipolares disponibilizam mais binário que os unipolares [12].

Os motores unipolares de 6 e 8 fios, por não terem os enrolamentos ligados internamente, podem também funcionar como motores bipolares, se a sua ligação elétrica for feita à semelhança dos motores bipolares.

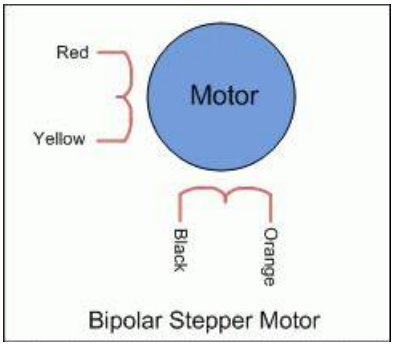


Figura 2.14: Esquema de ligações elétricas dos motores bipolares [11]

Em relação às suas dimensões, são disponibilizados motores passo a passo de diversos tamanhos e formatos para as mais diversas aplicações. No caso dos motores de tamanho médio e grande, existe uma norma criada pela NEMA, a associação de equipamento elétrico e fabricantes de equipamento de imagiologia médica dos Estados Unidos da América, que define vários formatos normalizados para motores passo a passo [13]. Um motor "NEMA 23"deverá ter, entre outras características normalizadas, uma face de montagem quadrada com arestas de 2,3 polegadas de comprimento.

2.4. Microstepping

2.4.1 Full stepping

A posição do veio de um motor passo a passo muda através da alteração do valor de corrente que atravessa cada um dos seus dois enrolamentos.

Os motores passo a passo têm tipicamente 200 passos, ou seja, conseguem assumir 200 posições radiais distintas sem recorrer a *microstepping*. Neste modo de funcionamento (*full stepping*), o valor de corrente em cada enrolamento alterna entre $+I_{\text{Max}}$ e $-I_{\text{Min}}$ de acordo com o diagrama de fase e o gráfico da variação de corrente da figura 2.15, onde I_{a} e I_{b} representa a corrente no enrolamento a e b, respetivamente.

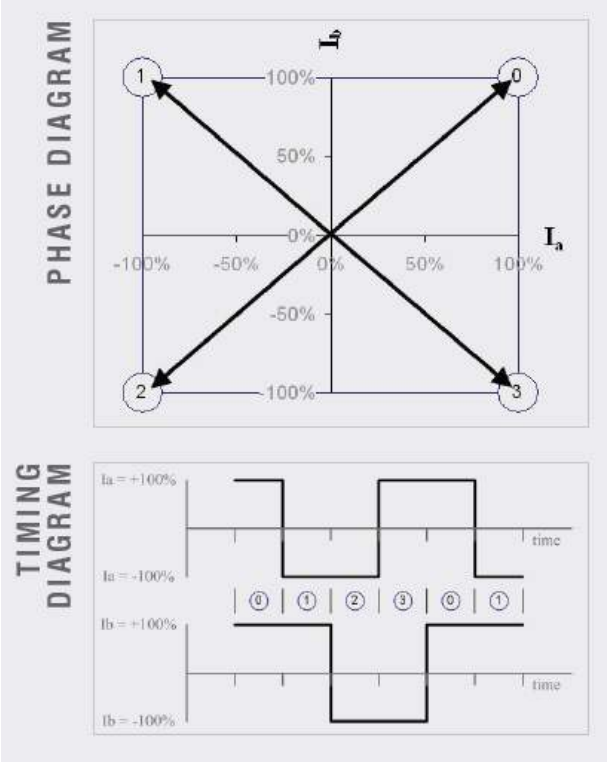


Figura 2.15: Diagrama de fase e variação da corrente no modo *full stepping* [14]

A sequência de quatro passos representada na figura 2.15 chama-se "*step cycle*" [14] (ciclo de passos). Um típico motor de 200 passos teria, portanto, de repetir este ciclo 50 vezes para atingir uma revolução completa, e o *driver* do motor teria apenas de alternar o sentido da corrente e não o seu valor em módulo, que permaneceria no máximo especificado.

2.4.2 Half stepping e outros modos de microstepping

Se o *driver* utilizado for capaz de alterar a magnitude da corrente que percorre os enrolamentos, pode ser aplicado o *microstepping*. O modo *half stepping*, cujo diagrama de fase e gráfico da variação da corrente em cada enrolamento no tempo está representado na figura 2.16, efetivamente divide cada passo em dois. Verifica-se, neste modo que $I_{\text{a,b}} \in \{I_{\text{Max}},\ 0,\ -I_{\text{Max}}\}$.

Na figura 2.17 estão representados dois modos diferentes de empregar $1/4$ *stepping*. As posições angulares do veio do motor são as mesmas em ambos os modos, mas a potência disponibilizada diverge, estando relacionada com o comprimento do fasor no diagrama de fase, como demonstrado nas equações 2.11 e 2.12.

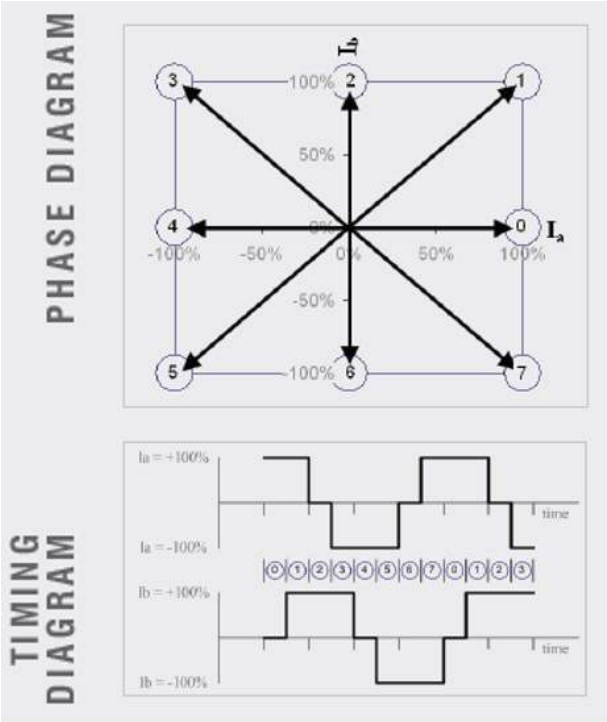


Figura 2.16: Diagrama de fase e variação da corrente no modo *half stepping* [14]

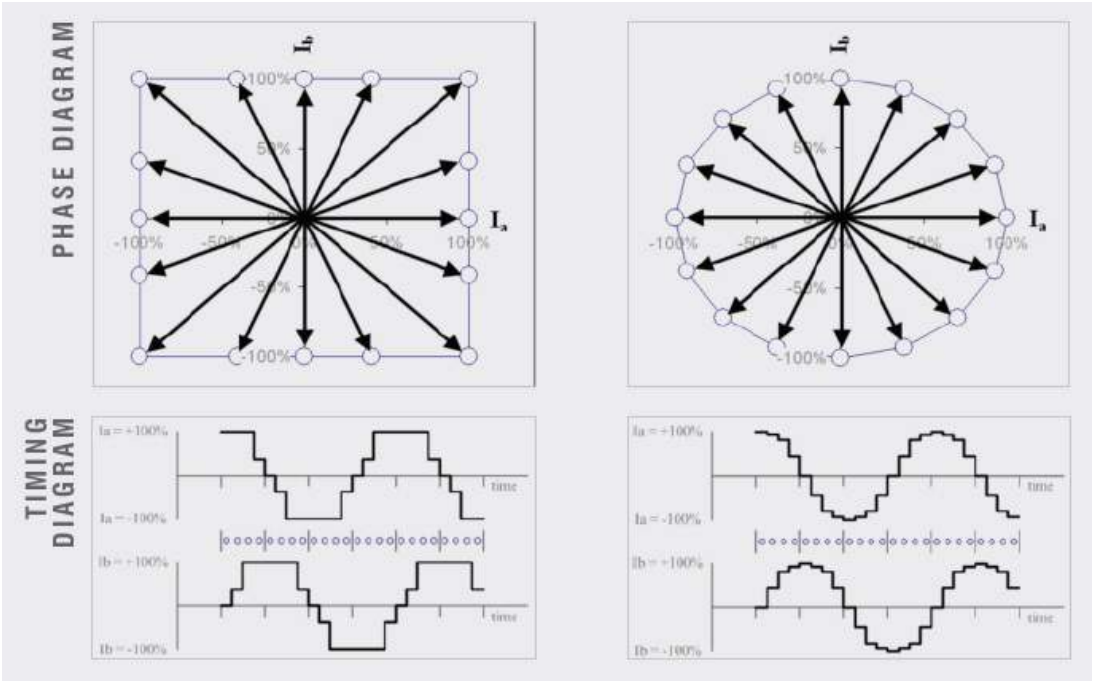


Figura 2.17: Diagrama de fase e variação da corrente no modo 1/4 *step* [14]

A potência disponibilizada a um dado momento é calculada pela equação 2.11, onde R é o valor de resistência de um enrolamento (os enrolamentos devem possuir a mesma resistência).

$$P_{motor} = I_{\mathrm{a}}^2 \times R + I_{\mathrm{b}}^2 \times R \tag{2.11}$$

Na equação 2.12 é calculado o comprimento de um fasor, e substituída a potência, demons-

trando a relação entre o comprimento de um fasor e a potência do motor.

$$L_{fasor} = \sqrt{I_{\mathrm{a}}^2 + I_{\mathrm{b}}^2} = \sqrt{\frac{P_{motor}}{R}} \tag{2.12}$$

Verifica-se que, dos dois modos de *microstepping* da figura 2.17, o segundo caso resultará num movimento mais suave, pois a potência permanece constante (e o binário também, já que este é diretamente proporcional à corrente, como indicado na equação 2.9).

3. DESENVOLVIMENTO DO SISTEMA DE DESLOCAMENTO

3.1. Motores passo a passo

De acordo com as especificações presentes no anexo A, os motores passo a passo Phidgets 57STH56 utilizados têm uma indutância de 4,5 mH, e a corrente máxima que o motor está projetado para consumir é 2,8 A. É possível, então, calcular a sua potência disponibilizada máxima, recorrendo à relação empírica da equação 2.10:

$$P_{\max} = U_{\text{Max}} \times I_{\max} = 32 \times \sqrt{L} \times I_{\max} = 190\,\text{W} \tag{3.1}$$

3.2. *Drivers* dos motores passo a passo

Ao contrário de outros tipos de motores de aplicações e dimensões semelhantes (motores DC de escovas e servomotores, por exemplo), os motores passo a passo têm mecanismos de controlo complexos. Parte da teoria por detrás destes mecanismos foi abordada no sub-capítulo 2.17.

Um *driver* de um motor passo a passo é um circuito que tem como entrada (*input*) sinais digitais que contêm informação relativa ao movimento que se pretende do motor, e como saída (*output*) os próprios fios de ligação dos enrolamentos do motor, sendo que o driver deve controlar corretamente o valor e o sentido da corrente nos enrolamentos de modo a atingir o movimento pretendido.

Os *drivers* utilizados na presente montagem (um para cada motor) são uns PiBot *stepper motor driver* rev. 2.2 (figura 3.1), cujo principal componente é um circuito integrado Toshiba TB6600HG, um circuito integrado de aplicação específica em drivers de motores passo-a-passo bipolares. A documentação do driver está disponível no anexo B. As suas principais características são:

- Tensão de entrada de 8 a 40 V DC.
- *Microstepping*: 1/1, 1/2, 1/4, 1/8, 1/16.
- Controla motores com um consumo de corrente nominal até 4,12 A.
- Potenciómetro de ajuste que limita a corrente máxima que o driver disponibiliza ao motor.
- LED de diagnóstico.
- Dissipador de calor incluído.

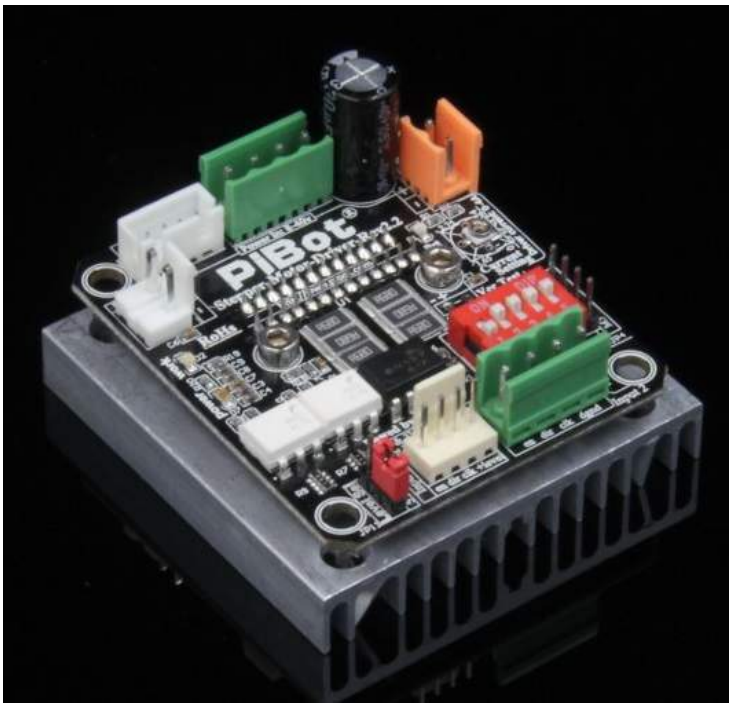


Figura 3.1: Driver PiBot stepper motor driver rev. 2.2

- Proteção contra sobre-aquecimento.

As entradas digitais estão preparadas para receber o seguinte tipo de impulsos:

en (*enable*) Se o nível lógico for alto, permite o movimento do motor. Tem que ser alto para o motor se movimentar. Uma vez parado, pode-se bloquear o veio mantendo o nível lógico alto. Neste estado o veio do motor suporta, sem deslizar, um determinado binário. Este deverá estar descrito nas especificações do motor, e no caso dos motores utilizados na presente montagem é de $12\,\mathrm{kg \cdot cm}$.

dir (*direction*) Consoante o nível lógico é alto ou baixo, o veio roda no sentido horário ou anti-horário.

clk (*clock*) O motor avança um passo (ou fração de um passo, consoante o modo de *microstepping*) a cada subida de nível lógico. O tempo que o nível lógico se mantém alto não é relevante, e nada acontece na descida para o nível lógico baixo, apenas a subida importa.

dgnd (*digital ground*) Ligação ao negativo da placa controladora.

O circuito integrado dos drivers tem uma potência dissipada máxima de 40 W [15], o que significa que os dois drivers podem dissipar até 80 W de calor. De modo a evitar o seu sobre-aquecimento, decidiu-se integrar uma ventoinha para criar convecção forçada junto aos dissipadores de calor dos drivers.

3.3. Fonte de alimentação

Foi necessário escolher uma fonte de alimentação com as seguintes especificações:

- Tensão de saída entre 8 e 40 V DC, de acordo com as especificações do driver, descritas na secção 3.2.

- Valor de intensidade corrente à saída adequado. Os dois motores têm um consumo de corrente máximo combinado de 5,6 A e os restantes componentes da montagem têm um consumo muito baixo em relação aos motores, mas o valor da fonte deverá ser mais alto, por forma a evitar um funcionamento em esforço.

A escolha recaiu para uma fonte de alimentação de um computador *desktop* que, por se encontrar desatualizado, já não era utilizado. Uma fotografia da fonte e as especificações desta encontram-se nas figuras 3.2 e 3.3, respetivamente. Escolhendo o rail de 12 V, o único acima do mínimo de 8 V, verifica-se que a fonte disponibiliza 18 A neste, valor acima do mínimo especificado.

Figura 3.2: Fonte de alimentação escolhida

Figura 3.3: Características da fonte de alimentação escolhida

As fontes de alimentação dos computadores *desktop* são, quase na sua totalidade, fontes comutadas que seguem a norma *ATX*, introduzida pela Intel em 1995 e atualizada em 2000 para a norma *ATX12V* [16]. Apesar dos diversos tipos de fichas, que ligam a diversos componentes dentro de um computador, as principais tensões disponibilizadas por este tipo de fontes são apenas três: 3,3 V, 5 V e 12 V. Uma tensão disponibilizada por uma fonte é habitualmente denominada por *rail* (p. ex. "Esta fonte disponibiliza 18 amperes no *rail* de 12 volts").

A principal diferença que a norma *ATX12V* introduziu foi a diminuição da corrente disponibilizada nos *rails* de 3,3 V e de 5 V e o aumento da corrente disponibilizada no *rail* de 12 V, fruto de avanços nos componentes, que passaram a requerer quase exclusivamente 12 V para o seu funcionamento.

Por disponibilizarem elevados valores de corrente nestes três *rails*, as fonte de alimentação anteriores à norma *ATX12V* são frequentemente modificadas para fontes de alimentação de bancada, onde são eliminados os fios e são afixados conectores banana à caixa, conseguindo-se dar utilidade a um componente de outro modo obsoleto. Esta conversão está bem documentada *on-line*.

A conversão para fonte de bancada seguiu os seguintes passos:

1. Cortaram-se as fichas originais com um alicate de corte (Figura 3.4a).

2. Furou-se a caixa em quatro sítios num engenho de furar e afixou-se quatro conectores banana fêmea (Negativo, 3,3 V, 5 V e 12 V).

3. Agruparam-se os fios de acordo com as suas cores:

Preto	Negativo
Laranja	3,3 V
Vermelho	5 V
Amarelo	12 V
Verde	Fio que dá a indicação à fonte para ela sair do modo *stand-by* quando o computador é ligado.

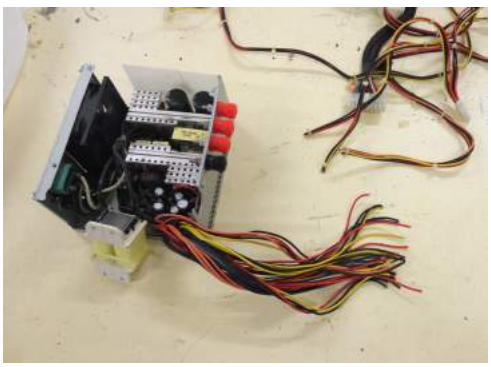
4. Isolou-se a ponta dos restantes fios que não foram utilizados nesta aplicação e guardaram-se dentro da caixa de modo a não interferir com a ventoinha.

5. Soldou-se o fio verde a um negativo, de modo a impedir a fonte de entrar em modo *stand-by*, já que nesta aplicação este modo não é desejado.

6. Como as fontes comutadas requerem uma pequena carga para estabilizarem os valores de tensão de saída [17], soldou-se uma resistência de 10 Ω e 10 W de poder de dissipação entre um fio vermelho (5 V) e um negativo. Fixou-se a resistência a um dissipador de calor dentro da caixa, garantindo o arrefecimento desta (Figura 3.4c).

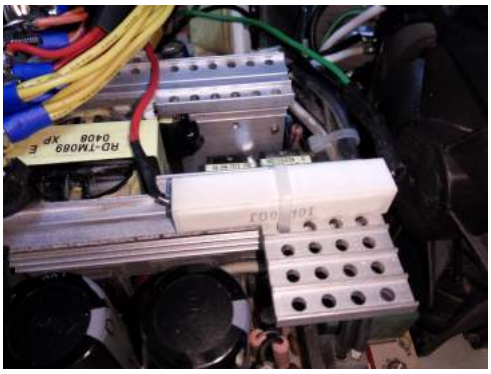
7. Soldaram-se os fios a olhais de acordo com a sua cor, e afixara-se os olhais ao conectores banana, tendo o cuidado de utilizar manga termo-retrátil nas ligações expostas (Figura 3.4b).

8. Arrumados os fios de modo a não interferirem com a ventoinha, fechou-se a caixa e verificou-se o correto funcionamento, medindo as tensões de saída com um multímetro. Com uma caneta de acetato escreveram-se as tensões de saída por baixo do respetivo terminal (Figura 3.4d).



(a)

(b)



(c)

(d)

Figura 3.4: Modificação da fonte de alimentação

3.4. Placa NI USB-6008 e *Software* LabVIEW

Para controlar o posicionamento das mesas lineares e para efetuar a interface com o utilizador planeou-se recorrer, inicialmente, a uma placa NI USB-6008 (figura 3.5) e ao *software* LabVIEW, ambos produtos da empresa National Instruments.

A placa NI USB-6008 apresenta-se como um "*Bus-powered Multifunction DAQ USB device*" [18]. Esta placa dispões de oito entradas analógicas, duas saídas analógicas e doze entradas/saídas digitais. A resolução do conversor analógico-digital (ADC) das entradas analógicas é de 11 *bits*, ou 12 *bits* se a ligação às entradas for do modo diferencial.

O software LabVIEW, "*Laboratory Virtual Instrument Engineering Workbench*", é a interface onde o utilizador programa a linguagem "G", uma linguagem não-textual, dividida em duas janelas.

A primeira e a principal chama-se "*block diagram*", e tem este nome por ser semelhante a um diagrama de blocos. O programador escolhe os blocos necessários para executar a função que pretende e define as relações e ligações entre os vários objetos/blocos.

A segunda janela, o "*front panel*", serve de interface com o utilizador final e permite inserir comandos ou instrumentos virtuais como botões, luzes avisadoras, potenciómetros, termómetros, etc.

Estes produtos foram recomendados por terem sido utilizados anteriormente com sucesso em aplicações semelhantes de controlo de motores passo-a-passo. A placa foi adquirida, e foi adaptado um software já existente, desenvolvido por um aluno do DEMI, para verificar o seu correto funcionamento.

Verificou-se de imediato que, ao diminuir o intervalo entre os pulsos a ser enviados para a entrada clk do driver (ou seja, ao aumentar a frequência do sinal desta entrada), de modo a aumentar a velocidade de rotação do motor para a gama necessária, este começava a saltar passos, ou seja, o driver do motor não estava a receber os impulsos devidos. Esta deficiência no comando causa fortes vibrações e um movimento muito instável. Acresce ainda que os motores passo-a-passo não dispõem de mecanismo de *feedback*, e como tal não há modo de compensar posteriormente os passos não dados.

Após discussão com o autor do software e alguma pesquisa, concluiu-se que a limitação é do sistema operativo Windows. A placa USB-6008, apesar de ter um microcontrolador [18], funciona ligada a um computador, com o LabVIEW aberto. O código é armazenado no computador, e a execução deste é feita pelo computador, e não no microcontrolador. Apesar do elevado poder de processamento de qualquer PC atual, o sistema operativo Windows não está dedicado apenas ao LabVIEW. Como consequência, a melhor resolução possível ronda o 1 ms [19].

Fez-se uma montagem experimental para determinar o intervalo entre pulsos menor em que o motor não saltasse passos. Configurou-se o driver para não usar microstepping, marcou-se a posição inicial do veio com uma caneta de acetato, e configurou-se o software para dar várias voltas completas ao motor. Para intervalos abaixo dos 20 ms começa-se a verificar que a posição final do motor não coincide com a inicial. A esta velocidade, calculou-se o tempo que a mesa levaria a percorrer todo o seu curso, de acordo com a equação 3.2. Note-se que 20 ms entre pulsos, sem microstepping, significa que o motor roda $\frac{1}{200}$ de volta em 20 ms. Como o passo do fuso da mesa é 2 mm, isto também significa que a mesa percorre $\frac{1}{200} \times 2\,\text{mm}$ em 20 ms. Logo, o tempo de movimento da mesa é dado por:

$$t_{\,20\text{ms}} = 700\,[mm] \times \frac{20}{1000 \times 60}\,[min] \times \frac{200}{2}\,\left[\frac{1}{\text{mm}}\right] = 23{,}3\,\text{min} \tag{3.2}$$

Este intervalo de tempo não é aceitável, e o funcionamento do motor a esta velocidade é pouco suave. Decidiu-se, portanto, utilizar o Arduino Uno como placa controladora, já que este executa o código na própria placa (não precisa de estar ligado a um computador). Assim, não

apresenta a limitação da placa NI USB-6008. Como benefício adicional o Arduino é uma ordem de grandeza mais barato que a placa da National Instruments.

Figura 3.5: Placa NI USB-6008 [20]

3.5. Arduino e respetivo código

O Arduino é uma plataforma de *hardware* e *software open-source* que se apresenta como uma solução para controlo digital de objetos físicos. A sua linguagem de programação é uma variante das linguagens C e C++.

A placa utilizada na montagem é a Arduino Uno, cujo microcontrolador é um Atmel ATmega 328p. A placa apresenta 14 entradas/saídas digitais e 6 entradas analógicas (que podem ser configuradas como entradas ou saídas digitais), cujo conversor analógico-digital (ADC) tem uma resolução de 10 bits. Seis das saídas digitais têm a capacidade de modulação por largura de pulso (PWM, *pulse with modulation*). O microprocessador funciona a 16 MHz.

O arduino dispõe funções capazes de contar micro-segundos. A função delayMicroseconds(), cuja função é colocar o programa em pausa por um determinado número de microsegundos, de acordo com a documentação do arduino [21], é muito precisa para intervalos a partir dos 3 microsegundos. Calculou-se na equação 3.3 que, para a velocidade máxima especificada do motor, deve ser enviado para a entrada clk do driver um pulso a cada 139,5 μs. O Arduino não apresenta, portanto, as limitações que a placa NI USB-6008 com o LabVIEW apresenta. Por esta razão, foi escolhido como placa controladora para a presente montagem.

$$2150\,[rpm] \times \frac{200}{6 \times 10^7} = 0,007167\,[\text{passos por } \mu s] \tag{3.3}$$

$$t_{\text{entre pulsos}} = 139{,}5\,\mu\text{s por passo} \tag{3.4}$$

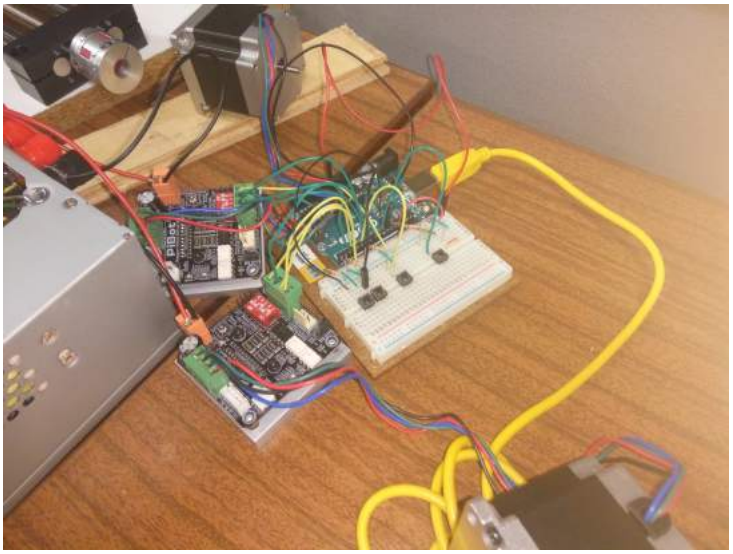


Figura 3.6: Montagem de teste dos componentes

Na figura 3.6 observa-se o exemplo de uma montagem de teste com o Arduino, onde se recorre ao uso de uma *breadboard*, uma placa perfurada muito útil na fase de prototipagem e testes, pois permite a troca rápida de componentes. O diagrama do circuito eletrónico finalizado encontra-se no anexo C.

3.5.1 Código

No desenvolvimento do código, começou-se por definir o que era pretendido deste:

- Ao ligar a fonte de alimentação, a mesa deve deslocar-se para a origem, onde estão colocados sensores de fim de curso. Ao atingir estes sensores os motores devem parar, finalizando o processo de inicialização.

- O utilizador pode inserir uma matriz no início do código com as coordenadas dos pontos para onde se pretende que a mesa se desloque, no formato $\{x_1, y_1, x_2, y_2, ...\}$. Neste modo, devem estar disponíveis dois botões para avançar para a posição seguinte ou retroceder para a anterior. Este modo de utilização será denominado "modo manual".

  Se o utilizador inserir uma posição que sai dos limites das mesas, ou se colocar um número ímpar de pontos, deve-se acender uma luz de erro e o programa deve parar.

- Alternativamente a este modo, deve estar disponível um "modo serial", que, utilizando a janela serial do programa do Arduino, permite ao utilizador inserir manualmente as coordenadas da posição desejada. Em vez de inserir as coordenadas, o utilizador poderá, igualmente, inserir a distância e direção a percorrer a partir do ponto onde a mesa se encontra.

  Se o utilizador inserir uma posição que sai dos limites da mesa, esta não deve avançar e deve ser mostrada uma mensagem de erro na janela serial.

Atendendo a estes requisitos, foi desenvolvido pelo autor o código que se encontra no anexo D. Note-se que cada funcionalidade acima descrita não corresponde, necessariamente, a uma só função. O código final tem 8 funções:

setup Função do Arduino presente em qualquer programa. Ao ligar o Arduino, corre apenas uma vez.

loop Função do Arduino presente em qualquer programa. O que estiver contido nesta função corre em *loop* até se desligar o arduino.

mainManualCalc Calcula a direção e os passos que os motores devem efetuar para ir para a posição seguinte na matriz de coordenadas que o utilizador inseriu no início do código.

inicializacao Coloca os motores a rodar na direção da origem, e pára quando os dois sensores de fim de curso forem ativados.

modoManual Movimenta a mesa para a posição seguinte ou para a anterior, consoante o *input* do utilizador.

recvWithEndMarker Recebe o que o utilizador escreve na janela serial.

parseData Identifica o tipo de movimento que o utilizador pretende (no modo serial) e divide o que foi inserido anteriormente em duas variáveis, xx e yy.

showParsedData Desloca os motores para a posição pretendida no modo serial.

As figuras 3.7 e 3.8 representam a utilização dos dois modos de funcionamento. O utilizador pode escolher o modo de funcionamento através do interruptor que se encontra no topo da caixa para os componentes, e que se pode observar na figura 3.9.

Na figura 3.7 observam-se várias matrizes de posições inseridas no início do código do Arduino. O programa aceita o upload de apenas uma, mas podem ser inseridas outras matrizes como comentários, de modo a agilizar a sua posterior utilização.

Na figura 3.8 observa-se o funcionamento do modo serial, em que, através da janela de comunicação serial disponível no computador, se inseriu, em primeiro lugar, o comando "30, 12"e em segundo lugar o comando "+46, -2". O software analisa o input e calcula o número de passos para cada motor, o sentido de rotação dos motores e a posição final da mesa.

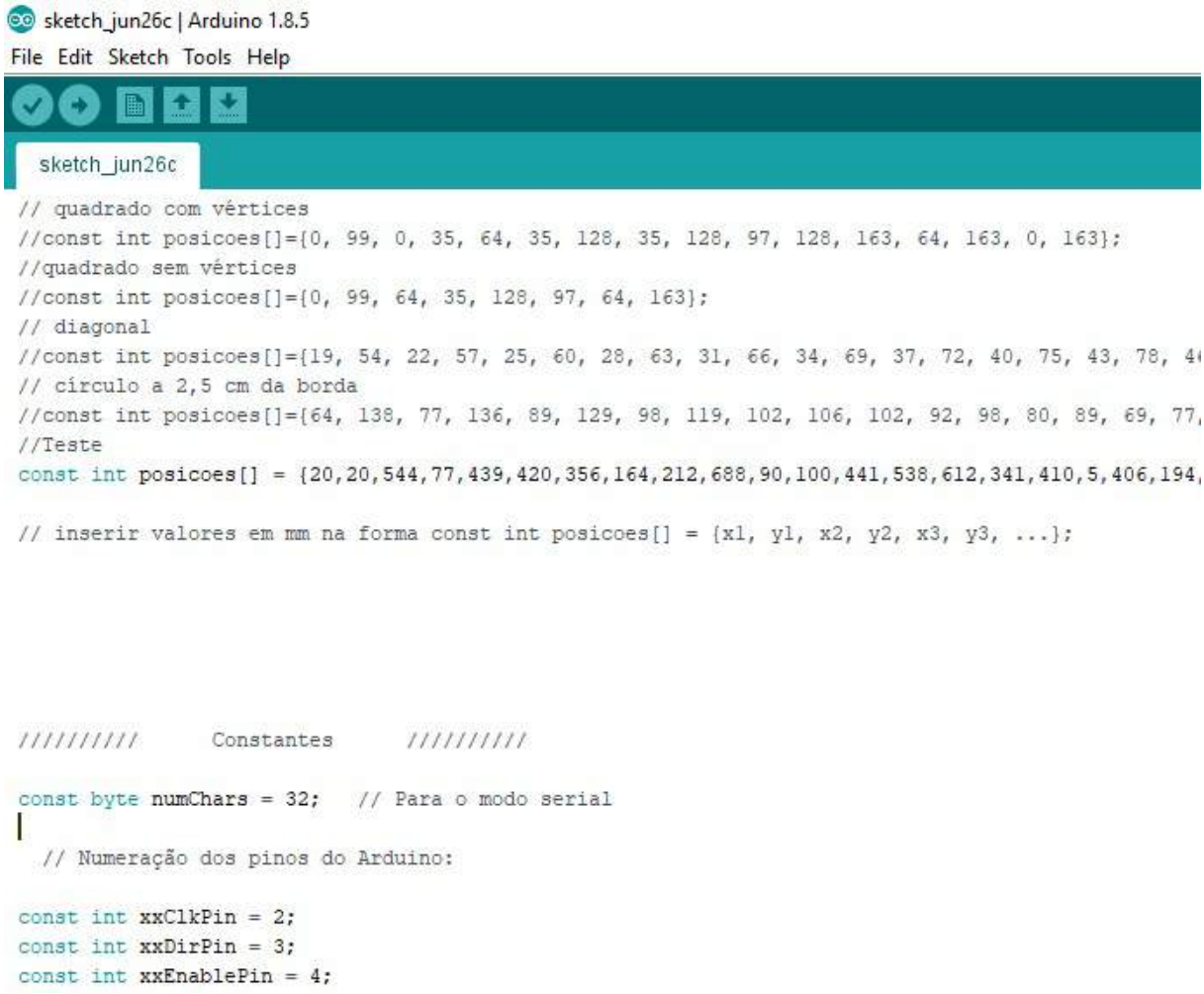


Figura 3.7: Demonstração do modo de funcionamento "Modo Manual"

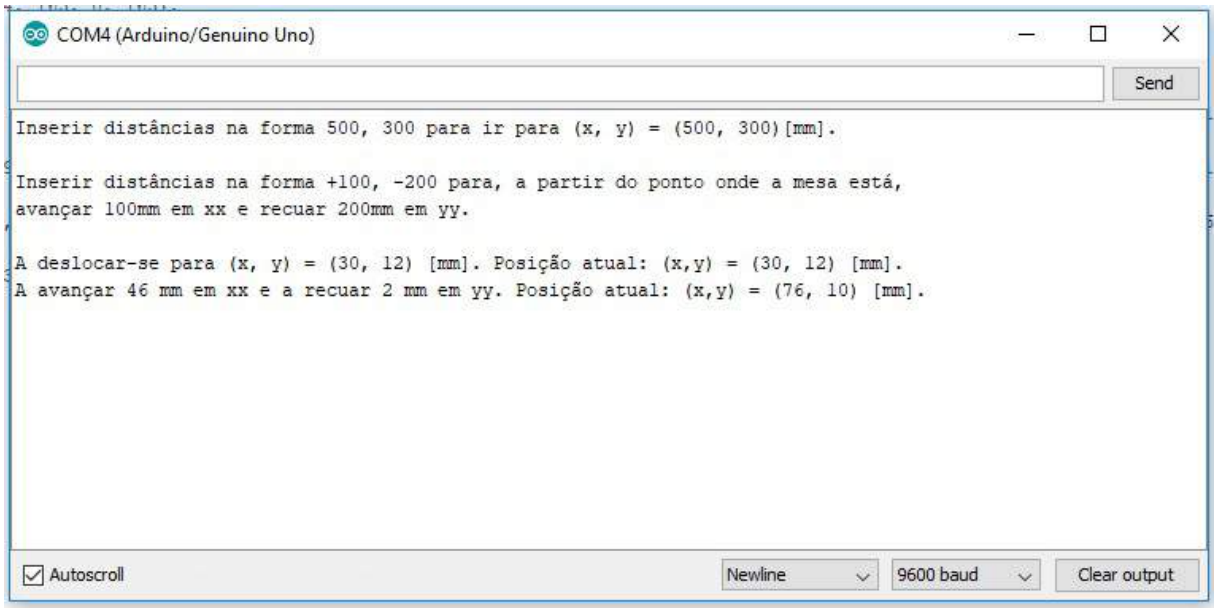


Figura 3.8: Demonstração do modo de funcionamento "Modo Serial"

Uma característica importante do código é o modo como a onda quadrada que será enviada para a entrada *clk* do driver (descrita no sub-capítulo 3.2) é gerada. No Arduino, o modo mais simples de gerar um sinal periódico de onda quadrada (para ligar e desligar um LED repetidamente, por exemplo) recorre à função *delay()*. O código abaixo mostra um exemplo de utilização desta função:

```
void loop() {
  digitalWrite(LED, HIGH);   // Acende o LED (HIGH representa +5 V na saída do Arduino)
  delay(1000);               // Espera um segundo (1000 ms)
  digitalWrite(LED, LOW);    // Apaga o LED (LOW representa 0 V)
  delay(1000);               // Espera outro segundo
}                            // Começa do início (função loop)
```

Esta função tem a desvantagem de colocar o programa em pausa durante o intervalo do *delay*, não sendo possível ler sensores, efetuar cálculos matemáticos ou manipular entradas ou saídas [22]. Recorrendo a esta função não seria possível, por exemplo, movimentar a mesa linear até ao sensor de fim de curso.

Em alternativa, utiliza-se um temporizador mais complexo, que recorre á função *millis*, que, quando chamada, devolve o número de milissegundos decorridos desde que o código foi iniciado no Arduino [23]. O código abaixo representa um exemplo de utilização deste temporizador em que uma onda quadrada é gerada até um sensor se ativar:

```
void loop() {

 pulso = 250;  // Meio período da onda quadrada, em milissegundos

 millisAtual = millis();  // Milissegundos decorridos desde que se iniciou o código

 if(sensor == 0){  // Se o sensor não foi ativado

  if(millisAtual - millisAnterior > pulso){
   millisAnterior = millisAtual;
   estadoClk = (estadoClk == LOW) ? HIGH : LOW;
   digitalWrite(saidaClk, estadoClk);

    // Se já passaram mais que 250 ms (pulso), muda o valor do estado da
    // saída digital Clk (se é HIGH passa para LOW, se é LOW passa para HIGH)

  }
 }

  else{  // Se  sensor foi ativado, o motor pára
         // (o estado da saída Clk deixa de variar e permanece no LOW)
   estadoClk = LOW;
   digitalWrite(saidaClk, estadoClk);
  }
 }
}
```

O detalhe acima exposto representa uma pequena parte do código, mas reflete processo de escrita deste. É, também, um bom exemplo da versatilidade do Arduino: para o mesmo

fim (alternar uma saída digital entre 0 V e 5 V), o utilizador tem disponível um método muito simples e fácil de compreender, mas mais limitado, e outro mais complexo mas que permite outras funcionalidades.

O código, na sua íntegra, ocupa 17 páginas de texto. Devido à sua dimensão, foi ponderada a sua inclusão como anexo do presente documento. No entanto, dada a importância deste *software* no funcionamento da máquina, e dada a sua elevada complexidade, o autor decidiu não o omitir do documento. Uma otimização do código que o tornasse mais curto e eficiente seria possível, mas os constrangimentos temporais não possibilitaram uma aprendizagem mais aprofundada das linguagens C e C++.

3.6. Caixa para os componentes

Foi necessário agrupar os *drivers*, a fonte de alimentação e o Arduino numa caixa, que deveria também incluir os botões de interface com o utlizador e as ligações aos restantes componentes. Para o efeito, foi adaptada uma caixa previamente desenvolvida por um colega.

Trata-se de uma caixa em madeira MDF (*Medium Density Fiberwood*) de 3 mm de espessura, cujas faces foram desenhadas no *software* SolidWorks, cortadas a laser nas instalações do FabLab na FCT, encaixadas umas nas outras e coladas com cola de madeira para reforço estrutural.

Além de proteger os componentes, a caixa foi equipada com uma ventoinha de 80 mm para promover a circulação de ar dentro da caixa, particularmente junto aos *drivers*, que dissipam uma potência considerável (até 80 W, como visto em 3.2). Na figura 3.10d observa-se a grelha da ventoinha, bem como a entrada USB do Arduino. Na tampa superior foram colocados os botões de posição anterior, posição seguinte, seletor de modo de funcionamento, *reset* do Arduino, e ainda o LED indicador de erro (figura 3.9), devidamente identificados. Na parte de trás encontram-se quatro conectores para os dois motores e os dois sensores de fim de curso, igualmente identificados, bem como a parte de trás da fonte de alimetação (figura 3.10a). Nas figuras 3.10b e 3.10c visualiza-se a disposição dos componentes no interior da caixa.

3.7. Sensores de fim de curso

Foram utilizados dois sensores de fim de curso, ambos posicionados na origem de modo a que, na inicialização, a mesa se desloque até esta e lá permaneça até próxima instrução.

Os sensores de fim de curso utilizados recorrem a um *microswitch* que, quando premido, fecha o circuito entre a respetiva entrada entrada do arduino e o negativo.

Foi necessário criar um meio de fixação entre o sensor e a mesa linear. Para o efeito, e à semelhança da caixa para os componentes (sub-capítulo 3.6), desenharam-se os apoios em SolidWorks, tendo sido posteriormente cortados a laser numa placa de MDF nas instalações do FabLab da FCT.

Na figura 3.11 está representado o modelo em SolidWorks do apoio do sensor do eixo dos xx e na figura 3.12 representam-se os sensores montados e já ligados à caixa. Note-se que o posicionamento dos sensores é ajustável através das duas ranhuras em cada um dos apoios, dada a necessidade de ajustar o ponto de ativação dos sensores *in situ*.

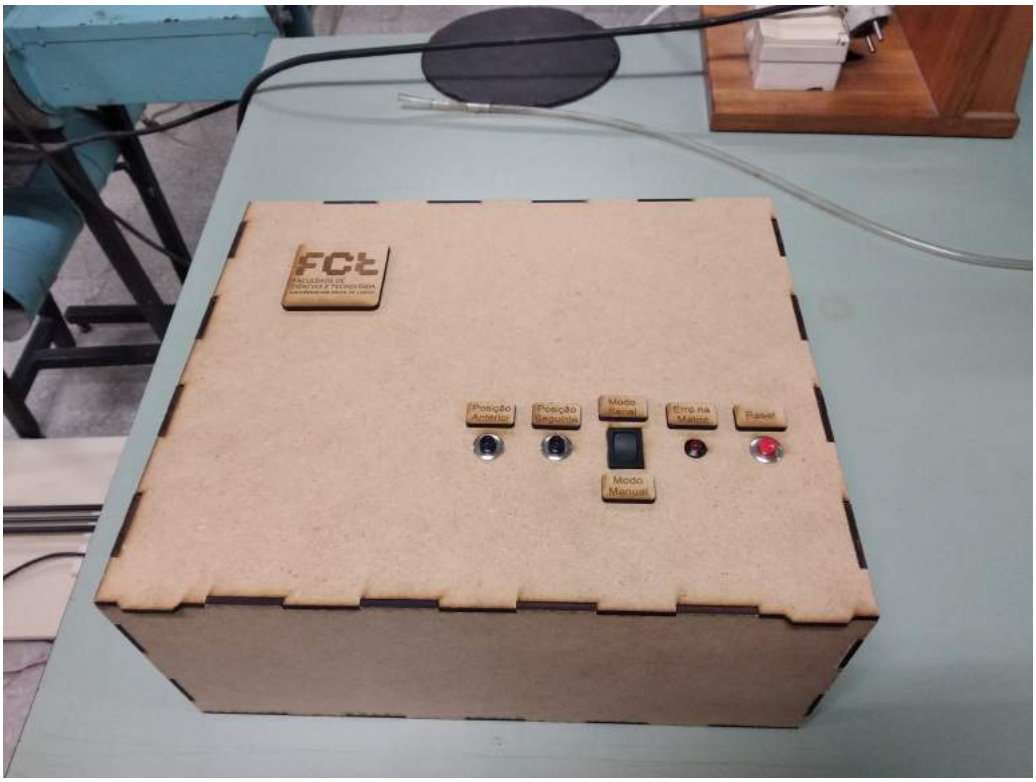
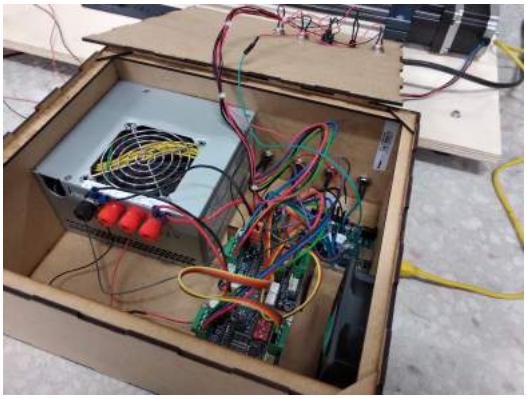


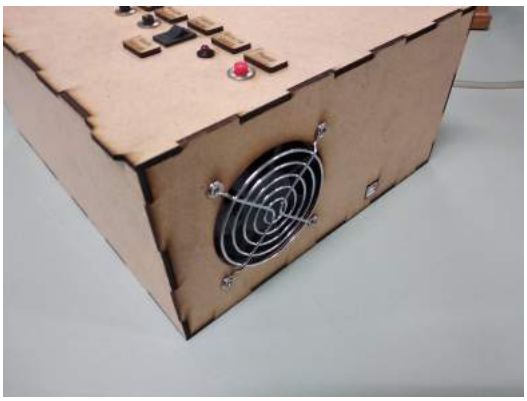
Figura 3.9: Caixa - vista do topo

(a)



(b)

(c)



(d)

Figura 3.10: Caixa - vista do interior, da parte de trás e da lateral

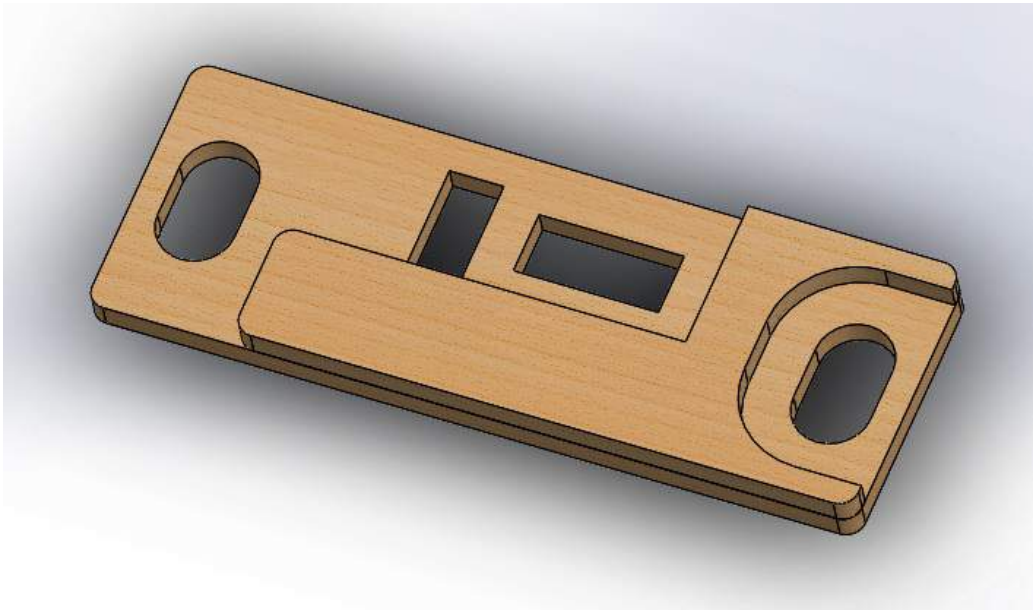


Figura 3.11: Modelo em SolidWorks de um apoio para um sensor de fim de curso

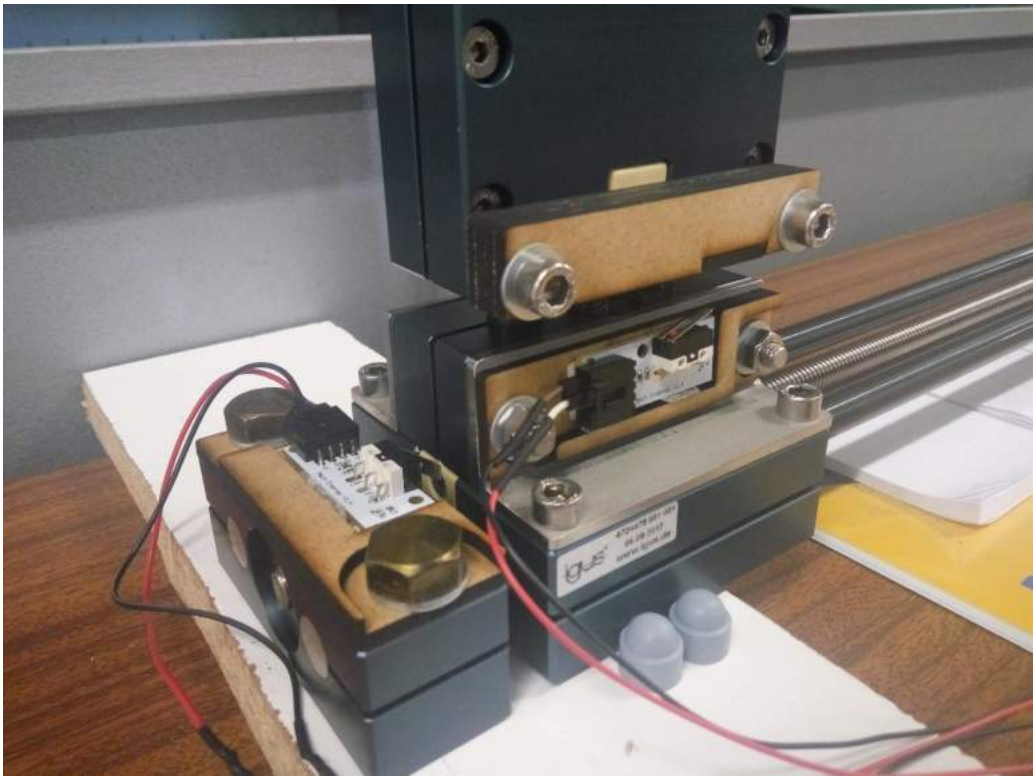


Figura 3.12: Sensores de fim de curso montados e a funcionar

3.8. Ligação entre a mesa linear e o tubo de Pitot

Foi acoplado à mesa linear um tubo em aço que atua como extensão para o tubo de Pitot, permitindo a sua inserção dentro do túnel aerodinâmico, mas mantendo a mesa e todos os seus componentes fora deste, com o objetivo de minimizar a interferência com o escoamento. Para o efeito, será necessário abrir uma ranhura vertical com 700 mm de altura (deslocamento máximo da mesa) numa das janelas do túnel, permitindo o movimento do tubo. Esta não foi aberta porque, devido a constrangimentos temporais, não se adaptou a montagem ao túnel (será preciso também apoio oficinal para a construção de uma estrutura de suporte adequada). Na figura 3.13 observa-se o tubo, que é ajustável em inclinação e extensão.

Foram também desenvolvidos acessórios que efetuam a ligação entre o Pitot e o tubo referido anteriormente. Na figura 3.14 observam-se vários acessórios desenvolvidos no software SolidWorks e impressos numa impressora 3D. A escolha do acessório indicado dependerá das necessidades do utilizador. Os acessórios cilíndricos da figura 3.14b permitem encaixar o Pitot dentro do tubo de extensão, e passar o tubo de silicone dentro deste.

Esta montagem experimental apresenta diversas possibilidades de utilização. Certamente que, alterando o instrumento de medição para um anemómetro de fio quente ou laser-Doppler, terão de ser fabricados novos acessórios.

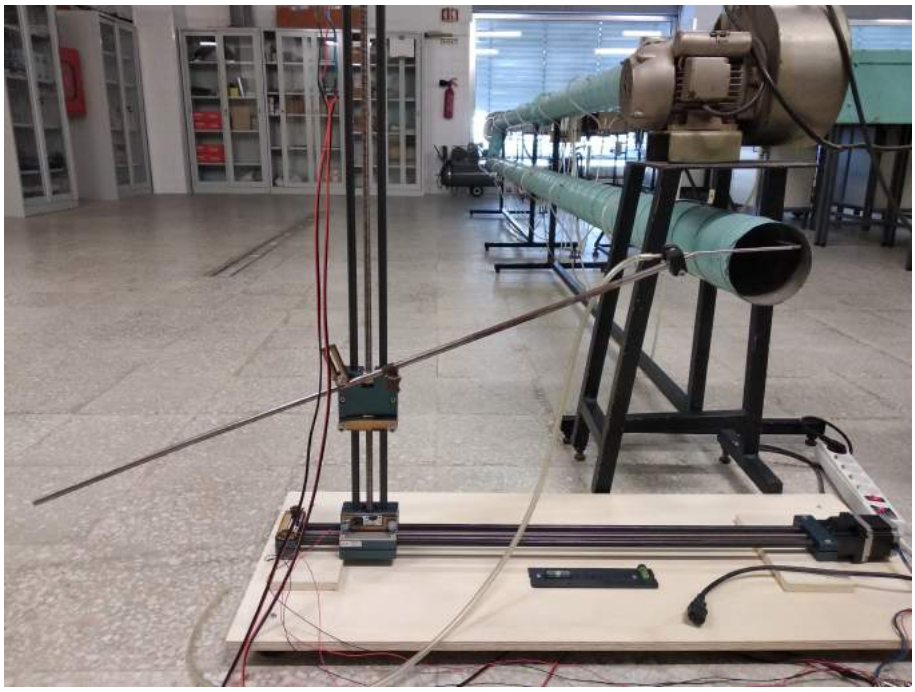


Figura 3.13: Detalhe do tubo em aço

(a)

(b)

Figura 3.14: Acessórios de ligação do Pitot ao tubo de extensão

4. TESTES EXPERIMENTAIS

A componente experimental da presente dissertação pretende testar o correto funcionamento da máquina desenvolvida bem como demonstrar a sua utilidade. Para o efeito, foi decidido que se iria realizar um teste ao posicionamento da mesa e que também se iriam realizar dois atravessamentos à saída de uma conduta circular, ligada a um ventilador.

No teste ao posicionamento, fez-se com que a mesa percorresse 53 pontos distintos e aleatórios, e verificou-se com um paquímetro a posição da mesa nos dois eixos, relativamente à origem, no início da malha de posições, a meio e no final.

No primeiro atravessamento ir-se-ia utilizar um tubo de Pitot para medir vários valores de velocidade ao longo de um diâmetro da conduta, de modo a definir um perfil de velocidades à saída desta e também calcular o caudal volúmico.

O segundo atravessamento seria um circulo concêntrico com a conduta, e permitiria meramente confirmar que a velocidade não se altera ao longo do círculo.

A experiência foi feita em colaboração com o colega João Chambel, que está a desenvolver um leitor de pressão diferencial integrado com o LabVIEW, permitindo-lhe testar a sua montagem.

4.1. Teste ao posicionamento

Os motores passo a passo, ao contrário dos servomotores, não dispõem de mecanismo de *feedback*, e como tal é importante garantir que a posição da mesa corresponde à posição efetivamente pretendida dela. A origem é o único ponto da mesa em que é dado *feedback* ao software sobre a posição desta, e este ponto só é considerado na inicialização do programa. Numa malha de posições, existindo algum erro no posicionamento de um ponto, esse erro na posição vai-se propagar a todos os pontos subsequentes.

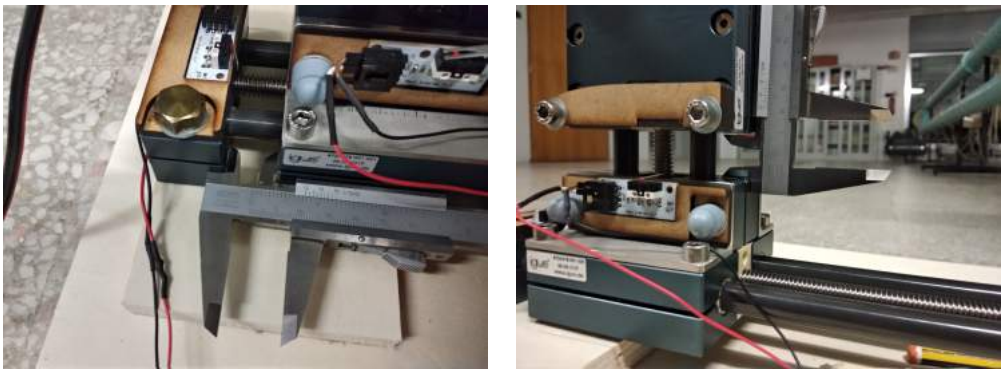
De modo a eliminar a existência de algum erro no software ou falha no hardware, tal como a existência de escorregamento no acoplamento entre o motor passo a passo e a mesa linear ou algum salto de passos da parte de um driver ou de um motor passo a passo, efetuou-se o seguinte teste:

1. Recorrendo a uma função do *software* Excel, geraram-se 25 pares de números aleatórios entre 0 e 700, que representam coordenadas de posição da mesa. Colocou-se, adicionalmente, o ponto $(x, y) = (20, 20)[mm]$ como primeira e última coordenadas desta malha de posições.

2. Inseriram-se as coordenadas no código do Arduino e fez-se *upload* deste.

3. Avançou-se para o primeiro ponto (20, 20) e confirmou-se com um paquímetro o deslocamento em xx e yy (figura 4.1).

4. Percorreram-se as 25 coordenadas aleatórias e na última (20, 20) verificou-se novamente o deslocamento.



(a) (b)

Figura 4.1: Medição do deslocamento da mesa do eixo dos xx (a) e yy (b)

5. Efetuou-se o regresso ao início, passando novamente por todas as coordenadas. Verificou-se o deslocamento no final.

Os valores de deslocamento medidos com o paquímetro foram iguais no primeiro ponto, no 27º e no último (53º): 20,4 mm em xx e 20,3 mm em yy. O deslocamento é ligeiramente superior a 20 mm devido ao ajuste do ponto de ativação do sensor de fim de curso.

O facto de não existir diferença mensurável na posição da mesa após esta ter percorrido 53 posições distintas comprova a precisão dos motores passo a passo, e afasta a possibilidade de existir algum problema com o *software* ou com os componentes da montagem capaz de causar desvios na posição da mesa.

4.2. Atravessamentos

4.2.1 Objetivos da experiência

- Efetuar a montagem do sistema de deslocamento bi-axial e de um tubo de Pitot de modo a que este percorra vários pontos ao longo da saída de uma conduta, que está acoplada a um ventilador.
- Efetuar a montagem do leitor de pressão diferencial integrado com o LabVIEW de modo a adquirir os valores de pressão necessários.
- No primeiro ensaio, registar 31 valores de pressão dinâmica ao longo de um diâmetro da saída da conduta.
- No segundo ensaio, registar 18 valores de pressão dinâmica ao longo de um círculo concêntrico com a própria conduta.
- Relativamente ao primeiro ensaio, determinar o perfil de velocidades à saída da conduta, e calcular o caudal volúmico.
- Relativamente ao segundo, efetuar uma análise estatística que permita determinar a qualidade dos valores registados.

4.2.2 Equipamento e procedimento

Equipamento

- Sistema de deslocamento bi-axial
- Ventilador assíncrono monofásico "vortice" de potência nominal $P = 460\,\mathrm{W}$
- Tubo de Pitot e tubo de ligação em silicone para este
- Acessório para ligar o tubo de Pitot ao tubo de extensão da mesa linear (conforme descrito em 3.8)
- Leitor de pressão diferencial Omega PX154 integrado com o LabVIEW
- Computador para registo dos valores de pressão e velocidade

Procedimento experimental

1. Colocar o sistema de deslocamento bi-axial no chão, junto ao ventilador
2. Montar o leitor de pressão diferencial.
3. Ajustar o nivelamento da base recorrendo aos pés ajustáveis e aos níveis de bolha (figura 4.2)

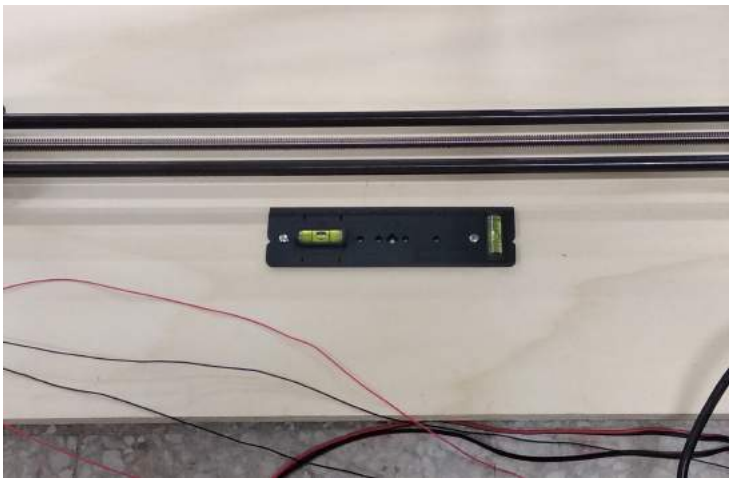


Figura 4.2: Verificação do nivelamento da base

4. Deslocar o Pitot ao longo da aresta de saída da conduta circular. Fazer os ajustes necessários para que, em qualquer ponto da aresta, o Pitot fique muito próximo, mas não entre em contacto, com a aresta. Garante-se assim o paralelismo entre o plano em que a mesa se movimenta e o plano da saída do tubo do ventilador, e minimiza-se o erro abordado em 2.2.1.
5. Deslocar o Pitot de modo a descobrir a posição da saída da conduta no plano da mesa linear. Com esta referência, criar uma matriz com as coordenadas de 31 pontos equidistantes ao longo de um diâmetro a 45° relativamente ao chão. O perfil de velocidades deverá ser igual ao longo de qualquer diâmetro, o ângulo de 45° permite demonstrar a capacidade da mesa se movimentar em dois eixos.

6. Criar uma matriz com 19 pontos de um círculo concêntrico com a conduta e a 2,5 cm de distância da aresta. Colocar as matrizes no início do programa do Arduino e fazer *upload* do programa.

7. Registar os valores de velocidade ao longo da diagonal. Para cada ponto, registar o valor de velocidade recorrendo ao LabVIEW e carregar no botão "posição seguinte".

8. Registar os valores de velocidade ao longo do círculo.

Na figura 4.3 observa-se a montagem experimental.

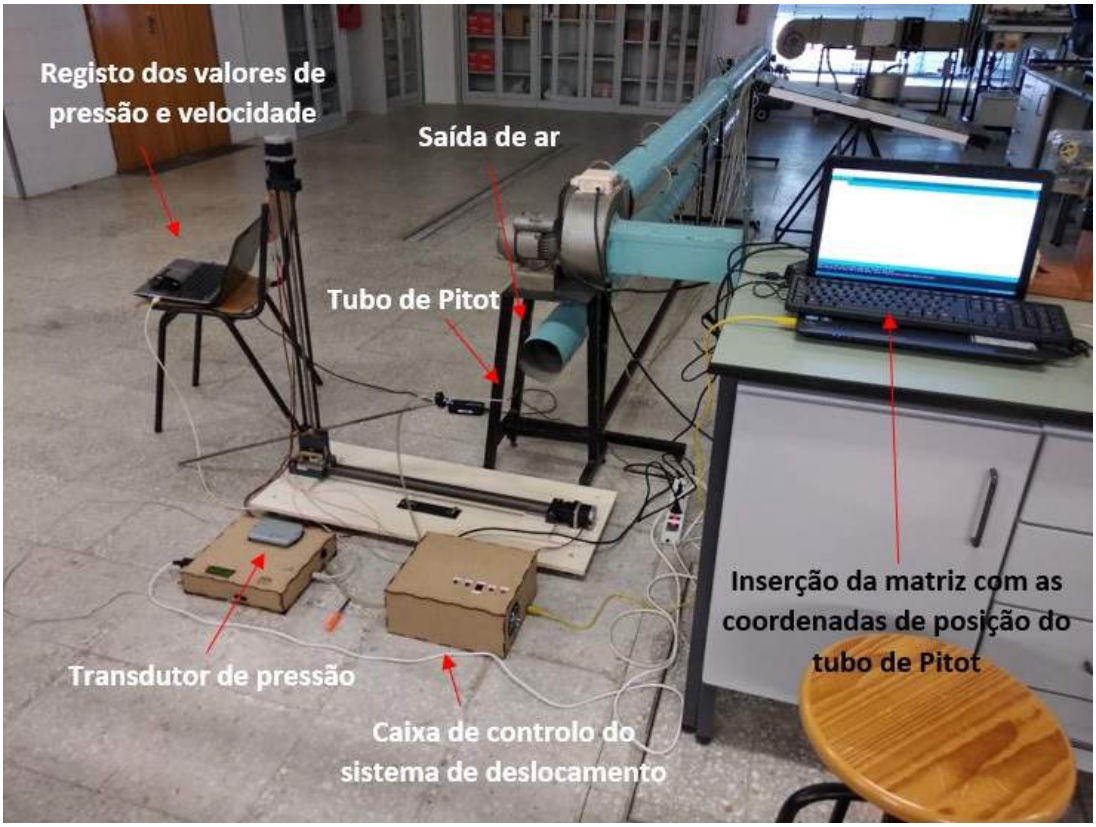


Figura 4.3: Montagem experimental

4.2.3 Resultados

Atravessamento diagonal

Nesta medição foram registados os valores da velocidade do escoamento para 31 pontos equidistantes ao longo da linha de diâmetro da figura 4.4. Para cada ponto, o leitor de pressão diferencial foi configurado para efetuar 250 medições em 5 segundos (o que corresponde a uma frequência de aquisição de 50 Hz). A média destes valores é calculada, e convertida em velocidade, aplicando a equação 2.4, automaticamente, no programa desenvolvido em LabVIEW. Na tabela 4.1 observam-se os valores de velocidade para cada ponto, e na figura 4.6 a sua representação gráfica. A esta representação gráfica foi adicionada uma interpolação polinomial de modo, que corresponde a uma aproximação do perfil de velocidades à saída da conduta.

A forma do perfil de velocidades da figura 4.6 sugere um escoamento turbulento, dada a rápida ascensão dos valores de velocidade junto da parede da conduta. Na parede da conduta, de acordo com a teoria da camada limite, a velocidade do escoamento é nula. Na realidade o

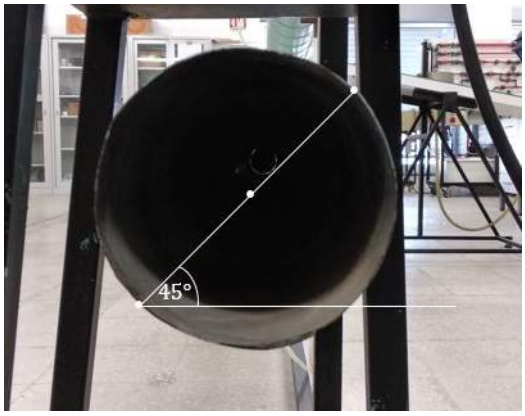


Figura 4.4: Linha percorrida pelo tubo de Pitot no atravessamento diagonal

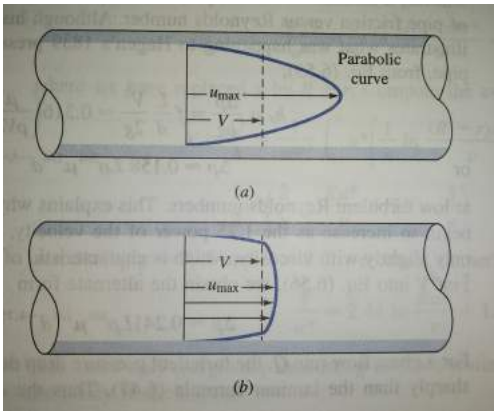


Figura 4.5: Perfil de velocidades para escoamento laminar (a) e turbulento (b) [7]

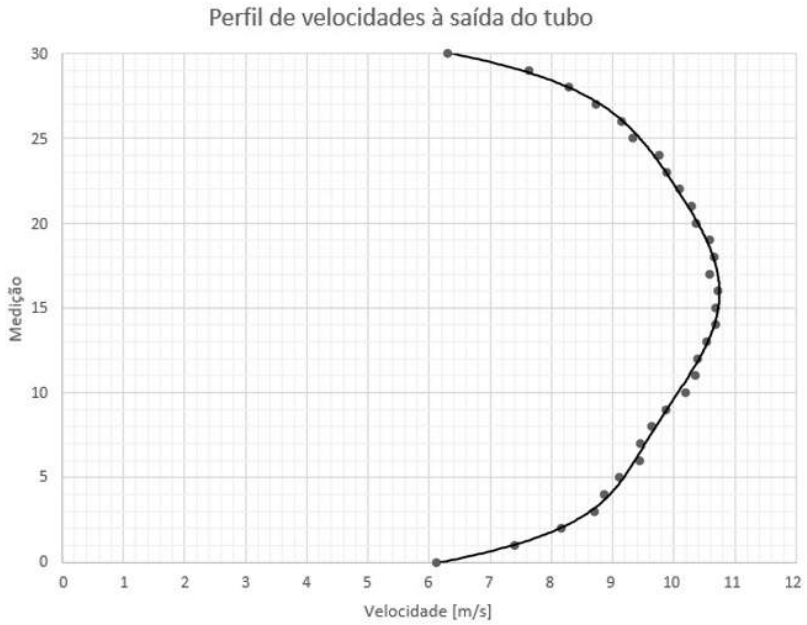


Figura 4.6: Perfil de Velocidades à saída do tubo

Tabela 4.1 Registo dos valores de velocidade para a medição diagonal

Diâmetro do círculo	D = 126 mm
Intervalo entre pontos de medição	d = 4,2 mm

Medição	Distância radial [mm]	Velocidade do escoamento [m/s]
0	0	6,12264
1	4,2	7,4035
2	8,4	8,16214
3	12,6	8,70312
4	16,8	8,86516
5	21	9,10469
6	25,2	9,4442
7	29,4	9,45674
8	33,6	9,63991
9	37,8	9,87376
10	42	10,1922
11	46,2	10,3489
12	50,4	10,3913
13	54,6	10,5413
14	58,8	10,6799
15	63	10,6781
16	58,8	10,7202
17	54,6	10,5888
18	50,4	10,6562
19	46,2	10,5823
20	42	10,3677
21	37,8	10,2892
22	33,6	10,0992
23	29,4	9,88141
24	25,2	9,75774
25	21	9,33562
26	16,8	9,1453
27	12,6	8,72166
28	8,4	8,28365
29	4,2	7,63343
30	0	6,2993

Tabela 4.2 Cálculo do caudal

i	Raio $R_i[m]$	Velocidade $V_i[m/s]$	Área $A_i[m^2]$	$V_iA_i[m^3/s]$
0	0	10,6781	5,542E-05	5,918E-04
1	0,0042	10,70005	1,663E-04	1,779E-03
2	0,0084	10,56505	2,771E-04	2,927E-03
3	0,0126	10,52375	3,879E-04	4,082E-03
4	0,0168	10,4656	4,988E-04	5,220E-03
5	0,021	10,27995	6,096E-04	6,267E-03
6	0,0252	10,08148	7,204E-04	7,263E-03
7	0,0294	9,869555	8,313E-04	8,204E-03
8	0,0336	9,669075	9,421E-04	9,109E-03
9	0,0378	9,60097	1,053E-03	1,011E-02
10	0,042	9,220155	1,164E-03	1,073E-02
11	0,0462	9,00523	1,275E-03	1,148E-02
12	0,0504	8,71239	1,385E-03	1,207E-02
13	0,0546	8,222895	1,496E-03	1,230E-02
14	0,0588	7,518465	1,607E-03	1,208E-02
15	0,063	6,21097		
$Q = \sum_{i=0}^{14} V_i\ A_i = 0{,}114\,22\,\mathrm{m^3/s} = 114{,}218\,\mathrm{L/s}$				

tubo de Pitot não é pontual, e não é possível efetuar uma medição de velocidade imediatamente após a parede. No entanto, o primeiro e o último valor de velocidade é cerca de metade do valor máximo. Na figura 4.5 pode-se comparar a forma típica do perfil de velocidades de um escoamento laminar (a) e turbulento (b), à saída de um tubo circular.

Na tabela 4.2 efetuou-se o cálculo do caudal de acordo com o método descrito em 2.2.2. Dada a simetria do círculo, para cada raio foram registados dois valores de velocidade. Os valores da coluna "Velocidade"correspondem à média destes dois valores.

Uma vez calculado o caudal, é possível calcular a velocidade média do escoamento: $\overline{V} = \frac{Q}{A}$. Com este valor, pela equação 4.1, calculou-se o número de Reynolds do escoamento. Os valores da massa volúmica e viscosidade foram retirados de tabelas das propriedades do ar a $20°C$, e d corresponde ao diâmetro da saída do tubo.

$$Re_d = \frac{\rho \overline{V} d}{\mu} = \frac{1,205 \times 9.160 \times 0,126}{1,80 \times 10^{-5}} = 7,7 \times 10^4 \tag{4.1}$$

Este elevado número de Reynolds confirma a hipótese apresentada anteriormente, e o escoamento é, efetivamente, turbulento (a transição de regime laminar para turbulento ocorre a $Re_{d,crit} \approx 2300$ [7].

O autor achou interessante tentar obter um perfil de velocidades de um escoamento com mais baixo número de Reynolds, que tivesse uma forma mais aproximada ao perfil da figura 4.5(a). Para o efeito, repetiu-se a experiência nos mesmos moldes, mas fechando a entrada do ventilador, que se observa na figura 4.7, e que tinha ficado completamente aberta na primeira medição. A entrada foi sendo fechada progressivamente enquanto os valores de pressão recolhidos se encontravam dentro da resolução do transdutor de pressão. Os resultados obtidos nesta experiência estão sintetizados nas tabelas 4.3, 4.4 e na figura 4.8.

Tabela 4.3 Registo dos valores de velocidade para a segunda medição diagonal

Diâmetro do círculo	D = 126 mm
Intervalo entre pontos de medição	d = 4,2 mm

Medição	Distância radial [mm]	Velocidade do escoamento [m/s]
0	0	1,52797
1	4,2	2,50828
2	8,4	2,73038
3	12,6	2,87996
4	16,8	2,88184
5	21	3,02282
6	25,2	3,09434
7	29,4	3,17933
8	33,6	3,23589
9	37,8	3,31949
10	42	3,38231
11	46,2	3,39311
12	50,4	3,45794
13	54,6	3,50921
14	58,8	3,55415
15	63	3,58483
16	58,8	3,60337
17	54,6	3,6211
18	50,4	3,5866
19	46,2	3,58636
20	42	3,5553
21	37,8	3,51328
22	33,6	3,46823
23	29,4	3,36112
24	25,2	3,29234
25	21	3,22375
26	16,8	3,13499
27	12,6	2,98855
28	8,4	2,90411
29	4,2	2,65972
30	0	2,07487

Figura 4.7: Admissão do ventilador

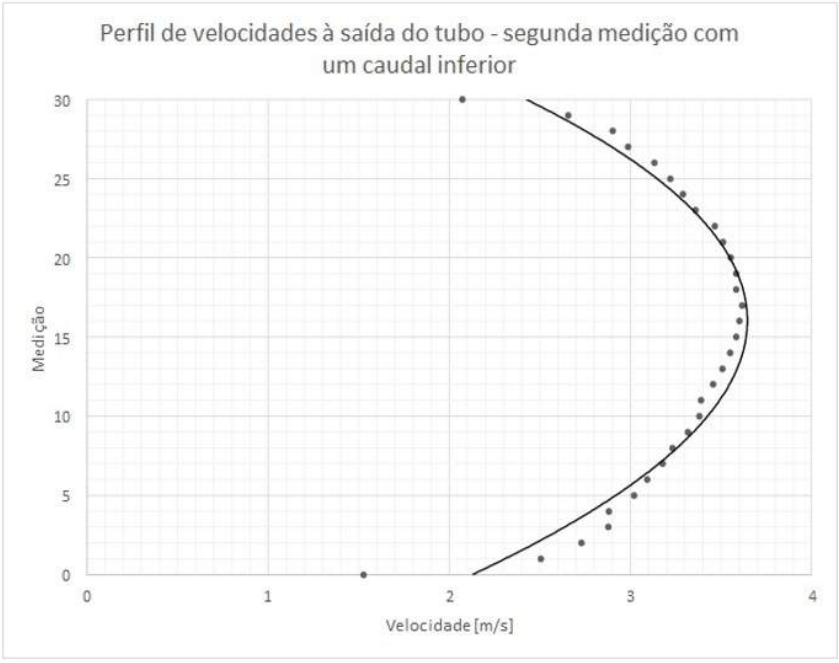


Figura 4.8: Perfil de velocidades à saída do tubo na segunda medição

Tabela 4.4 Cálculo do caudal para a segunda medição diagonal

i	Raio $R_i[m]$	Velocidade $V_i[m/s]$	$A_i = \pi(R_{i+1}^2 - R_i^2)$	$V_i A_i[m^3/s]$
0	0	3,58483	5,542E-05	1,987E-04
1	0,0042	3,57876	1,663E-04	5,950E-04
2	0,0084	3,565155	2,771E-04	9,879E-04
3	0,0126	3,52227	3,879E-04	1,366E-03
4	0,0168	3,489735	4,988E-04	1,741E-03
5	0,021	3,468805	6,096E-04	2,115E-03
6	0,0252	3,416385	7,204E-04	2,461E-03
7	0,0294	3,35206	8,313E-04	2,786E-03
8	0,0336	3,270225	9,421E-04	3,081E-03
9	0,0378	3,19334	1,053E-03	3,362E-03
10	0,042	3,123285	1,164E-03	3,635E-03
11	0,0462	3,008415	1,275E-03	3,835E-03
12	0,0504	2,934255	1,385E-03	4,065E-03
13	0,0546	2,817245	1,496E-03	4,215E-03
14	0,0588	2,584	1,607E-03	4,153E-03
15	0,063	1,80142		
$Q = \sum_{i=0}^{14} V_i \ A_i = 0{,}038\,597\,\mathrm{m}^3/\mathrm{s} = 38{,}5967\,\mathrm{L/s}$				

Com estes dados, calculou-se o novo número de Reynolds:

$$Re_d = \frac{\rho \overline{V} d}{\mu} = \frac{\rho \times \frac{Q}{A} \times d}{\mu} = \frac{1,205 \times \frac{0,038597}{\pi \times 0,063^2} \times 0,126}{1,80 \times 10^{-5}} = 2,7 \times 10^4 \tag{4.2}$$

O escoamento mantém-se em regime turbulento, o que justifica as pequenas diferenças no perfil de velocidades, e nas condições desta montagem experimental não foi possível obter um escoamento laminar, pois a resolução do transdutor de pressão não é suficiente para medir velocidades muito baixas.

Medição circular

Nesta medição foram registados os valores de velocidade do escoamento ao longo de um círculo distanciado de 2,5 cm da aresta do tubo. Foram registados valores ao longo de 18 pontos equidistantes. Na tabela 4.5 estão registados os valores de velocidade.

No tratamento destes valores foram calculados alguns parâmetros estatísticos relevantes, os quais estão indicados na tabela 4.6. No cálculo da incerteza $t \times S_m$ o valor t foi obtido da tabela t Student, bi-caudal, para 17 graus de liberdade ($n-1$ medições) e um intervalo de confiança de 95%. Este intervalo de confiança permite afirmar que:

$$u \ \epsilon \ [\overline{u} - t \times S_m; \ \overline{u} + t \times S_m] \iff u \ \epsilon \ [9.40365; \ 9.56499][m/s]$$
$$\text{Onde } u \text{ é a velocidade do escoamento a 2,5 cm da aresta do tubo.} \tag{4.3}$$

O último valor da tabela apresenta o erro relativo, uma representação adimensional da qualidade da medição.

Tabela 4.5 Registo dos valores de velocidade para a medição circular

Medição	Velocidade do escoamento [m/s]
0	9,64306
1	9,58879
2	9,59406
3	9,71151
4	9,65863
5	9,69816
6	9,61467
7	9,53749
8	9,45387
9	9,39155
10	9,27826
11	9,22372
12	9,21492
13	9,29517
14	9,39806
15	9,36631
16	9,4894
17	9,56018

Tabela 4.6 Análise estatística dos valores de velocidade da medição circular

Média $\overline{u}$ $[m/s]$	9,48432
Desvio padrão σ $[m/s]$	0,16290
Desvio padrão da média S_m $[m/s]$	0,03840
Incerteza $t \times S_m$ $[m/s]$	0,08067
Erro relativo [%]	0,85055

4.2.4 Discussão dos resultados

Em ambas as medições o sistema de deslocamento bi-axial desempenhou a sua função de forma célere, e sem percalços. O posicionamento correto do tubo de Pitot foi particularmente difícil, dada a dificuldade de medir pontos de referência num círculo, e dado o facto da saída do tubo estar amolgada e ter que ser ajustada com um alicate. Ainda assim, os resultados obtidos são de boa qualidade: na medição diagonal, o perfil de velocidades é simétrico e tem uma forma expectável, e na medição circular observa-se uma precisão muito boa em relação aos valores adquiridos, dado que o erro relativo calculado é inferior a 1%.

5. CONCLUSÃO

A montagem foi concluída e é funcional. Os objetivos impostos inicialmente foram cumpridos, e qualquer utilizador da máquina pode, com facilidade, efetuar medições de velocidade numa malha de posições (até 100) e ao longo de um caminho tão complexo quanto for necessário para a experiência. Pode, também, inserir coordenadas de forma expedita através da janela serial do Arduino, sem ser necessário criar e fazer upload de uma matriz de posições.

É importante notar o baixo custo da montagem, particularmente após a tomada de decisão de utilizar um Arduino em detrimento da placa NI USB-6008. As mesas lineares e os motores passo a passo têm um custo considerável, mas os restantes componentes são comuns e baratos. Recorrendo a equipamentos como a máquina de corte laser (que trabalha não só com MDF mas com outros tipos de madeira, cortiça, cartão, acrílico, etc.) e a impressora 3D, foi possível criar, com algum engenho, componentes muito úteis a um preço muito baixo. A prototipagem também é facilitada com estes equipamentos, já que a fabricação das peças é muito rápida.

A utilização do Arduino requereu a familiarização com a sua linguagem de programação e com a sua eletrónica, mas revelou-se uma plataforma muito versátil e capaz, em parte porque usa uma linguagem baseada em C e C++, duas linguagens estabelecidas no ramo da informática, e em parte devido à extensa documentação disponível *on-line*.

Um defeito da montagem é a sua estabilidade, e deve-se ao modo como as mesas estão montadas. Uma base metálica, em conjunto com reforços metálicos na mesa linear de baixo (xx), poderiam reduzir este problema, mas nunca eliminá-lo, porque o peso do motor de passo a passo que controla o eixo dos yy vai sempre estar na parte superior da montagem, e como este se move, não pode ser apoiado. No entanto, o registo de valores com a mesa em movimento nunca foi um objetivo, e após chegar ao ponto pretendido a mesa permanece imóvel.

A ligação entre o tubo que serviu de extensão para o Pitot e a mesa linear pode também ser melhorada, fabricando uma fixação mais resistente e com um ajuste angular mais amplo e fácil de ajustar pelo utilizador. Uma fixação adequada teria que ser desenvolvida em alumínio, recorrendo a uma fresadora CNC. Tal fixação não foi desenvolvida dada a falta de apoio oficinal em tempo útil.

5.1. Trabalhos futuros

Um trabalho futuro será a adaptação da montagem aos dois túneis aerodinâmicos existentes no Laboratório de Mecânica de Fluidos e Termodinâmica Aplicada da FCT-UNL. Devido a constrangimentos temporais foi decidido não efetuar esta adaptação, que é um trabalho oficinal e não iria valorizar tanto a dissertação, em detrimento de outros aspetos mais vocacionados para a engenharia. Adicionalmente, à data da elaboração da presente dissertação, o túnel aerodinâmico aberto não se encontrava ainda em funcionamento.

Os aspetos negativos mencionados anteriormente (estabilidade da montagem e ligação entre a mesa e o tubo de extensão) podem ser colmatados ou eliminados em trabalhos de fabricação futuros, recorrendo a materiais como o alumínio e ferramentas de oficina.

BIBLIOGRAFIA

[1] NASA, *Whirling Arms and the First Wind Tunnels*. URL: https://www.grc.nasa.gov/www/k-12/WindTunnel/history.html (acedido em 30/01/2018).

[2] Wright Brothers Aeroplane Co., *The 1901 Wright Wind Tunnel*. URL: http://www.wright-brothers.org/Adventure%7B%5C_%7DWing/Hangar/1901%7B%5C_%7DWind%7B%5C_%7DTunnel/1901%7B%5C_%7DWind%7B%5C_%7DTunnel.htm (acedido em 30/01/2018).

[3] Daimler, *Climatic wind tunnel, Mercedes-Benz plant Sindelfingen*.

[4] Wikipedia, *Wind Engineering*. URL: https://en.wikipedia.org/wiki/Wind%7B%5C_%7Dengineering (acedido em 01/02/2018).

[5] Miguel Angel Gonzalez, *Wind Tunnel Designs and Their Diverse Engineering Applications*. 2013, p. 228, ISBN: 9789535110477. DOI: http://dx.doi.org/10.5772/3403. URL: http://www.intechopen.com/books/wind-tunnel-designs-and-their-diverse-engineering-applications%7B%5C%%7D5Cnwww.intechopen.com.

[6] N. Hall, *Pitot-Static Tube*. URL: https://www.grc.nasa.gov/www/k-12/airplane/pitot.html (acedido em 04/09/2018).

[7] F. M. White, *Fluid Mechanics*, 4º. McGraw-Hill, 1999, pp. 175, 385–389.

[8] D. COLLINS, *Hysteresis loss and eddy current loss: What's the difference?*, 2018. URL: https://www.motioncontroltips.com/hysteresis-loss/.

[9] ——, *Detent torque and holding torque*, 2016. URL: https://www.motioncontroltips.com/faq-whats-the-difference-between-detent-torque-and-holding-torque/.

[10] Gecko Drive Motor Controls, *Step Motor Basics*. URL: https://www.geckodrive.com/support/step-motor-basics.html (acedido em 27/08/2018).

[11] George, *Unipolar Stepper Motor vs Bipolar Stepper Motors*, 2012. URL: https://www.circuitspecialists.com/blog/unipolar-stepper-motor-vs-bipolar-stepper-motors/ (acedido em 28/08/2018).

[12] D. Khatik, *What is the difference between - bipolar & unipolar stepper motor?* URL: https://www.quora.com/What-is-the-difference-between-bipolar-unipolar-stepper-motor (acedido em 28/08/2018).

[13] Nema, "NEMA ICS 16 - Industrial Control and Systems Motion/Position Control Motors , Controls , and Feedback Devices", p. 187, 2001.

[14] inc. Zaber Technologies, "ZABER Microstepping Tutorial", pp. 1–5, 2014.

[15] Toshiba, *Toshiba TB6600HG*. 2016.

[16] Intel, "ATX12V Power Supply Design Guide", pp. 1–44, 2003.

[17] S. E. Ltd., *What is Minimum Load?* URL: https://www.sunpower-uk.com/glossary/what-is-minimum-load/.

[18] National Instruments, *User Guide NI USB-6008/6009*, 2012.

[19] ——, *Minimum elapsed time resolution*. URL: https://forums.ni.com/t5/LabVIEW/Minimum-elapsed-time-resulution/td-p/135554 (acedido em 04/06/2018).

[20] ——, *USB-6008*. URL: http://www.ni.com/pt-pt/support/model.usb-6008.html (acedido em 20/09/2018).

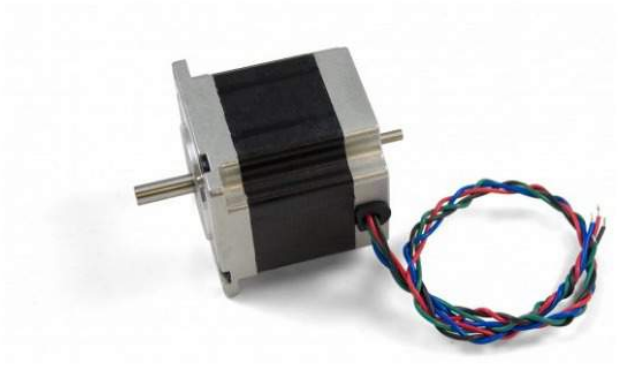
[21] Arduino, *delayMicroseconds()*. URL: https://www.arduino.cc/reference/en/language/functions/time/delaymicroseconds/ (acedido em 03/07/2018).

[22] ——, *delay()*. URL: https://www.arduino.cc/reference/en/language/functions/time/delay/ (acedido em 29/08/2018).

[23] ——, *millis()*. URL: https://www.arduino.cc/reference/en/language/functions/time/millis/ (acedido em 29/08/2018).

Anexos

A. Especificações do motor passo a passo



57STH56 NEMA 23 Bipolar Precision Gearless Stepper

ID: 3330_0

RoHS COMPLIANT

This precision bipolar stepper with rear shaft has a 0.9° step angle and 11.2 kg·cm of torque at low speeds.

$28.00

Quantity Available: 382

Qty	Price
5	$26.88
10	$25.76
25	$25.20
50+	...

– 1 +

ADD TO CART

Description | Connection & Compatibility | **Specifications** | Resources | Other Steppers

Product Specifications

Motor Properties	
Motor Type	Bipolar Stepper
Manufacturer Part Number	57STH56-2804MB
Step Angle	0.9°
Step Accuracy	± 5 %
Holding Torque	12 kg·cm
Rated Torque	11.2 kg·cm
Maximum Motor Speed	2150 RPM
Acceleration at Max Speed (w/1067 Motor Controller)	800000 1/16 steps/sec²
Electrical Properties	
Recommended Voltage	12 V DC
Rated Current	2.8 A
Coil Resistance	900 mΩ
Phase Inductance	4.5 mH
Physical Properties	
Shaft Diameter	1/4"
Rear Shaft Diameter	3.9 mm
Mounting Plate Size	NEMA - 23
Weight	695 g
Number of Leads	4
Wire Length	300 mm

B. Documentação dos *drivers*

What is it

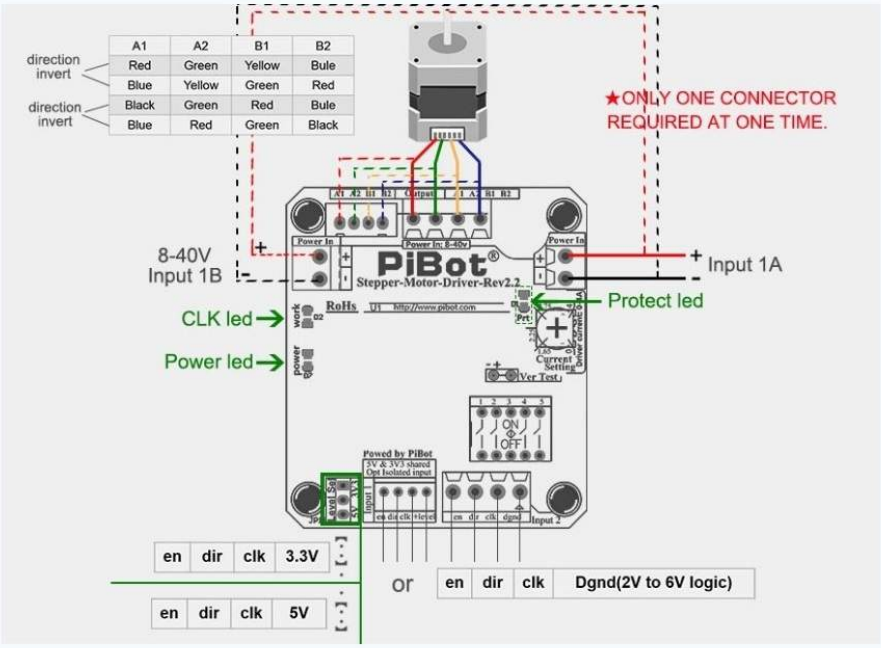
A variety style signal supply logic input selectable, which compatible with lots of MCU and other controller board, such as Mach3 USBCNC Raspberry Pi and so on.

- Quick connector and Normal connector suitable for hobby and factory.
- Wide power input DC 8 - 40V.
- Signal supply logic in Quick connector (en dir clk "3.3V or 5V") optical isolated to the IC.
- Signal supply logic in Normal connector (en dir clk dgnd "2V to 6V logic") with high speed direct to IC.
- Microstepping: 1/1, 1/2A, 1/2B, 1/4, 1/8, 1/16(default).
- 0 - 4.5A(4.12A) ~ (from Toshiba6600 manual) output current can be accurate set by read the voltage of voltmeter.
- Diagnosis LED available.
- Heatsink available.
- Reserve the M3 M2 M1 TQ input pins for program control.

How it works

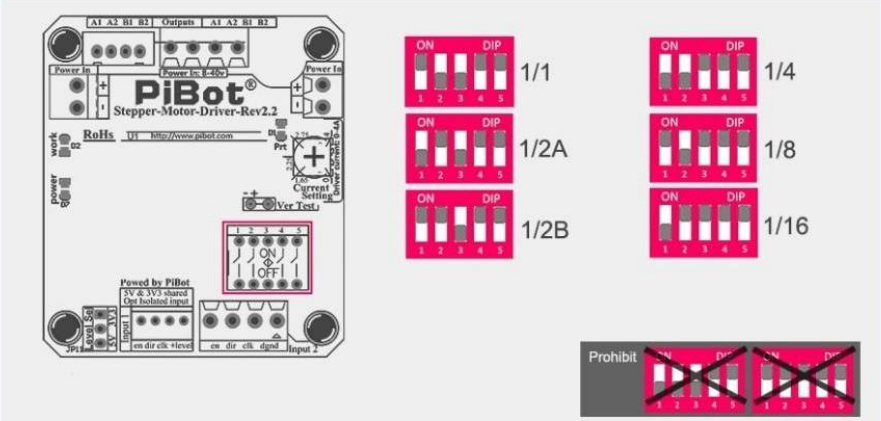
Input and Output

Normal connector and Quick connector is parallel. ONLY ONE CONNECTOR REQUIRED AT ONE TIME.



Micro-stepping Setting

This picture shows how you can change the micro-stepping settings of the driver.



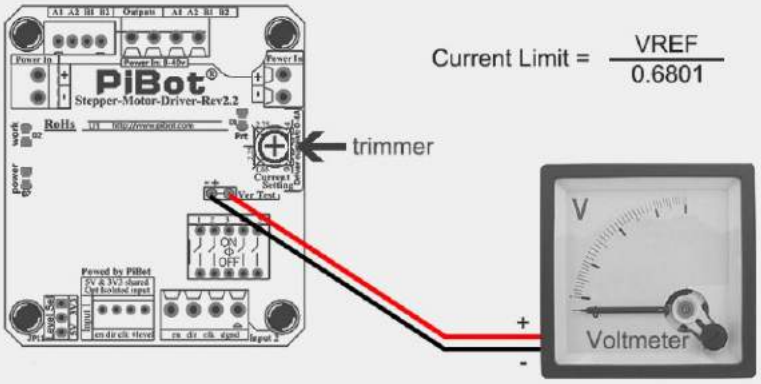
Output Limit current Adjust

You can approximate the current limit by the position of the marked point on the trimmer. DO NOT EXCEED YOUR MOTOR'S CURRENT RATING.

OR the current limit can be adjusted by measuring VREF and turning the trimmer. Connect the [+] of the voltmeter to VREF and the [-] lead to GND and read the value.

Current Limit = VREF ÷ 0.6801

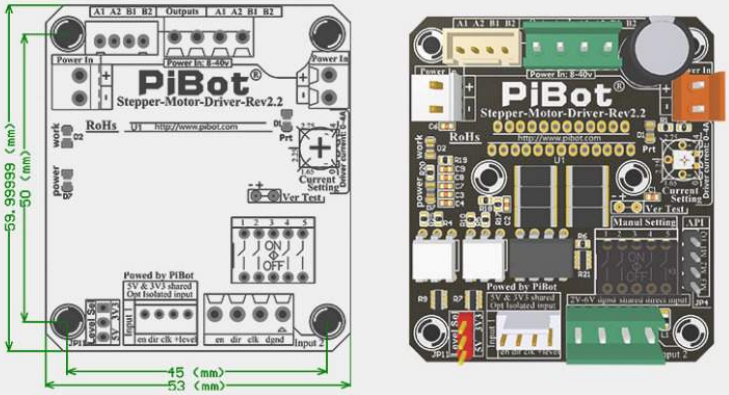
So for example the Max current is: 2.81V ÷ 0.6801 = 4.1317A



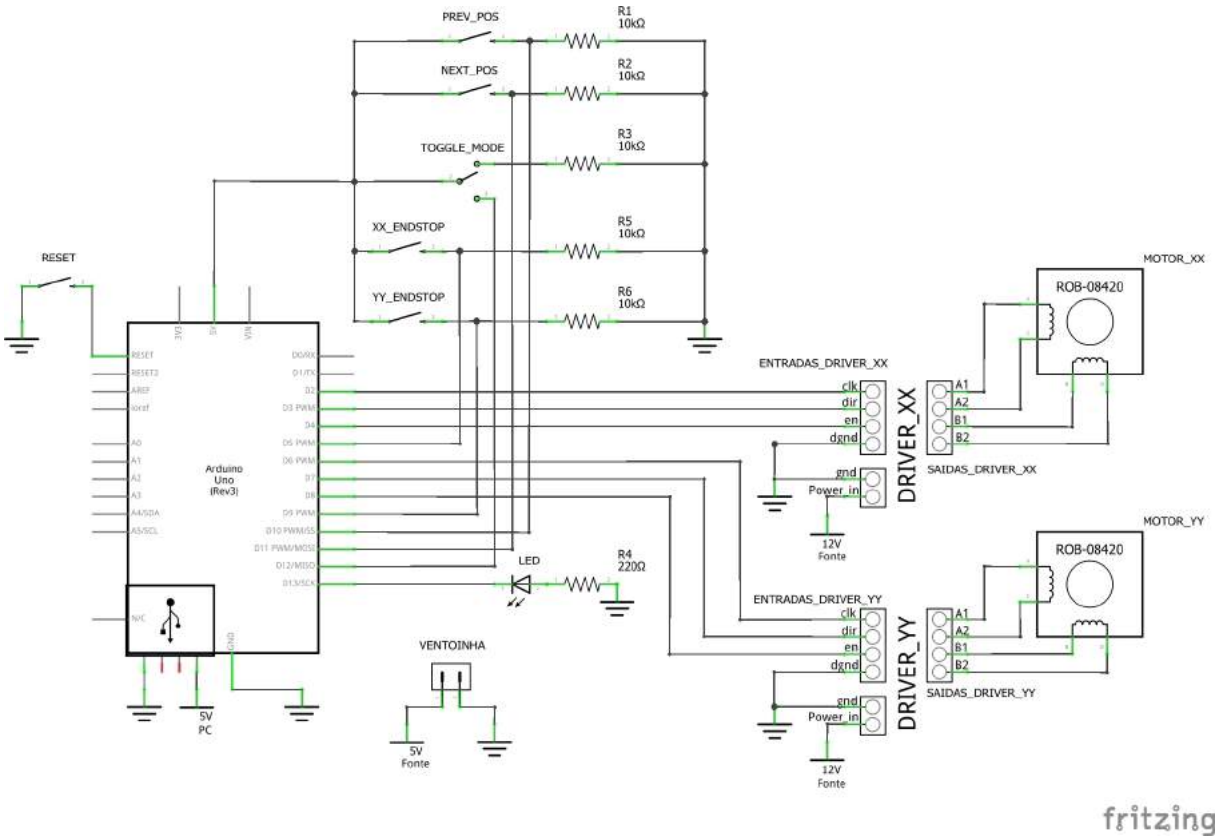
Specification

- Driver Chip: Toshiba6600
- Power Input: DC 8 - 40V
- Supply Logic in Quick Connector: 3.3V or 5V
- Supply Logic in Easy Connector: 2V to 6V
- Drive Current: 0 - 4.5A(4.12A)
- Microstepping: 1/1, 1/2A, 1/2B, 1/4, 1/8, 1/16(default)
- Size of Driver Board: 53mm × 60mm
- Hole Diameter: 3mm
- Mounter Hole: 45mm × 50mm

Figure Dimension



C. Esquema de ligações



D. Código implementado no Arduino

```
const int posicoes[] = {};

// inserir valores em mm na forma const int posicoes[] = {x1,
y1, x2, y2, x3, y3, ...};




//////////     Constantes     //////////

const byte numChars = 32;   // Para o modo serial

  // Numeração dos pinos do Arduino:

const int xxClkPin = 2;
const int xxDirPin = 3;
const int xxEnablePin = 4;
const int xxEndstopPin = 5;
const int yyClkPin = 6;
const int yyDirPin = 7;
const int yyEnablePin = 8;
const int yyEndstopPin = 9;
const int prevPosPin = 10;
const int nextPosPin = 11;
const int toggleModePin = 12;
const int errorLedPin = 13;


  // Características do equipamento:

const int pulse = 250;

const int passoMesa = 2;    // Passo do fuso das mesas é de
2mm

const int stepping = 2;     // Os drivers estão no modo half
stepping (2)
                            // mudar para 4 para 1/4
stepping, 16 para 1/16, etc.
```

```
const int passosPorRot = 200; //  Num de passos que o motor
leva para completar uma rotação



//////////     Variáveis     //////////

// Aqui inicializam-se as variáveis que vão ser lidas e/ou
modificadas por mais que uma função
// ou cujo valor tem de ser guardado entre iterações de uma
função
// As restantes são inicializadas na sua respetiva função

int numCoordenadas = 0;     // Inicialização das variáveis
long xxImpulsos[50];        // relacionadas com o
long yyImpulsos[50];        // modo manual
int coordenada = -1;        //

int mainManualCalcDone = 0;   // Para correr a função
mainManualCalc apenas uma vez. Como ela contém
                              // um ciclo for não pode ficar
no setup, pois só iria correr uma iteração deste ciclo

int xxClkState = LOW;
int xxDirState = LOW;
int yyClkState = LOW;
int yyDirState = LOW;
int errorLedState = LOW;

unsigned long currentMicros = 0;        // Variáveis
auxiliares necessárias
unsigned long xxPreviousMicros = 0;     // para a geração do
sinal clk
unsigned long yyPreviousMicros = 0;     //

int xxOrigem = 0;     // Variáveis relacionadas com o modo de
inicialização.
int yyOrigem = 0;     // Tomam o valor de 1 quando os motores
atingem a origem,
                     // terminando assim a inicialização
```

```
char receivedChars[numChars];    // Variáveis relacionadas
char tempChars[numChars];        // com o modo serial
int xx = 0;                      //
int yy = 0;                      //
int modo = 0;                    //
boolean newData = false;         //
int xxPosAtual = 0;              //
int yyPosAtual = 0;              //

int prevToggleMode = 0;



void setup() {

  Serial.begin(9600);     // Abre uma porta serial

  pinMode(xxClkPin, OUTPUT);
  pinMode(xxDirPin, OUTPUT);
  pinMode(xxEnablePin, OUTPUT);
  pinMode(xxEndstopPin, INPUT);
  pinMode(yyClkPin, OUTPUT);
  pinMode(yyDirPin, OUTPUT);
  pinMode(yyEnablePin, OUTPUT);
  pinMode(yyEndstopPin, INPUT);
  pinMode(prevPosPin, INPUT);
  pinMode(nextPosPin, INPUT);
  pinMode(toggleModePin, INPUT);
  pinMode(errorLedPin, OUTPUT);

  digitalWrite(xxEnablePin, HIGH);    // Mantém o motor
sempre energizado. Serve para
  digitalWrite(yyEnablePin, HIGH);    // manter o rotor
bloqueado quando o motor está parado

  Serial.println("Inserir distâncias na forma 500, 300 para
ir para (x, y) = (500, 300)[mm].");
  Serial.println();
  Serial.println("Inserir distâncias na forma +100, -200
para, a partir do ponto onde a mesa está,");
```

```
  Serial.println("avançar 100mm em xx e recuar 200mm em yy.");
  Serial.println();

}



void loop() {

  int toggleModeState = digitalRead(toggleModePin);   // Esta
variável é definida aqui para ir verificando se
                                                      // o
utilizador mudou de modo de utilização

  if(toggleModeState != prevToggleMode){          // Leva a
mesa à origem se o utilizador mudar o modo
    xxOrigem = 0;                                 // de
utilização
    yyOrigem = 0;
    prevToggleMode = toggleModeState;
  }

  if(mainManualCalcDone == 0){
    mainManualCalc();
  }

  if(xxOrigem == 0 || yyOrigem == 0){       // Corre a função
inicialização
    inicializacao();                          // até a mesa
atingir a origem
  }

  else if(toggleModeState == HIGH){
    modoManual();
  }

  else{
    recvWithEndMarker();
    if (newData == true) {
        strcpy(tempChars, receivedChars);       // This
temporary copy is necessary to protect the original data
```

```
        parseData();                                // because
strtok() used in parseData() replaces the commas with \0
        showParsedData();
        newData = false;
    }
  }
}



void mainManualCalc() {

  int numPosicoes = sizeof(posicoes)/sizeof(posicoes[1]);  //
Devolve o num de elementos no vetor posições
  numCoordenadas = numPosicoes/2;

  for(int i = 0; i < numPosicoes;
i++){                                                    //
Erro - se o utilizador tentar sair dos limites
    if(posicoes[i] > 700 || posicoes[i] < 0 || numPosicoes %
2 != 0 || numPosicoes > 100){ // da mesa linear, se colocar
um número ímpar de
      errorLedState =
HIGH;
       // posições ou se colocar mais de 100 posicões
      digitalWrite(errorLedPin,
errorLedState);                                          //
(limite de memória), a função pára de correr
      mainManualCalcDone =
1;
  // e acende o led de erro

return;
                        //

}
                          //

}
                            //
```

```
  int xxCoord[numCoordenadas] = {0};      // Coordenadas em
xx para onde se deseja ir
  int yyCoord[numCoordenadas] = {0};
  long xxDist[numCoordenadas] = {0};       // Distância em xx
a percorrer para ir para essa coordenada
  long yyDist[numCoordenadas] = {0};
  long xxNumImpulsosArr[numCoordenadas] = {0};    // Num de
impulsos a dar ao motor para ir para essa coordenada
  long yyNumImpulsosArr[numCoordenadas] = {0};

  for(int i = 0; i < numCoordenadas; i++){

    xxCoord[i] = posicoes[2*i];
    yyCoord[i] = posicoes[2*i+1];

    if(i == 0){
      xxDist[0] = xxCoord[0];
      yyDist[0] = yyCoord[0];
    }

    else{
      xxDist[i] = xxCoord[i]-xxCoord[i-1];
      yyDist[i] = yyCoord[i]-yyCoord[i-1];
    }

    xxImpulsos[i] = (2 * xxDist[i] * stepping * passosPorRot)
/ passoMesa;
    yyImpulsos[i] = (2 * yyDist[i] * stepping * passosPorRot)
/ passoMesa;
  }
  mainManualCalcDone = 1;
}



void inicializacao() {

  currentMicros = micros();

  if(xxOrigem == 0){
```

```
    xxDirState = HIGH;                      // Garante que
motor roda na direcção certa
    digitalWrite(xxDirPin, xxDirState);    // para levar a
mesa à origem

    if(digitalRead(xxEndstopPin) == LOW){                //
Temporizador
      if(currentMicros - xxPreviousMicros > pulse){      //
fornece uma onda quadrada
        xxPreviousMicros = currentMicros;                //
em que cada subida
        xxClkState = (xxClkState == LOW) ? HIGH : LOW;    //
corresponde a um passo
        digitalWrite(xxClkPin, xxClkState);              //
do motor
      }
    }

    else{
      xxClkState = LOW;
     digitalWrite(xxClkPin, xxClkState);
      xxOrigem = 1;
    }
  }

  if(yyOrigem == 0){

    yyDirState = HIGH;                      // Garante que
motor roda na direcção certa
    digitalWrite(yyDirPin, yyDirState);    // para levar a
mesa à origem

   if(digitalRead(yyEndstopPin) == LOW){
     if(currentMicros - yyPreviousMicros > pulse){
       yyPreviousMicros = currentMicros;
       yyClkState = (yyClkState == LOW) ? HIGH : LOW;
       digitalWrite(yyClkPin, yyClkState);
     }
    }

    else{
```

```
      yyClkState = LOW;
      digitalWrite(yyClkPin, yyClkState);
      yyOrigem = 1;
    }
  }
}



void modoManual() {

  int posSwitchState = 0;

  unsigned long xxPreviousMicros = 0;
  unsigned long xxImpulsosCont = 0;
  unsigned long yyPreviousMicros = 0;
  unsigned long yyImpulsosCont = 0;

  if(digitalRead(nextPosPin) == HIGH){
    posSwitchState = 1;
  }

  else if(digitalRead(prevPosPin) == HIGH){
    posSwitchState = 2;
  }

  else{
    return;
  }

 switch(posSwitchState){
    case 1:

      if(coordenada < numCoordenadas-1){

        coordenada++;

        xxDirState = (xxImpulsos[coordenada] > 0) ? LOW :
HIGH;
       digitalWrite(xxDirPin, xxDirState);
```

```
       yyDirState = (yyImpulsos[coordenada] > 0) ? LOW :
HIGH;
       digitalWrite(yyDirPin, yyDirState);

       while(xxImpulsosCont < abs(xxImpulsos[coordenada]) ||
yyImpulsosCont < abs(yyImpulsos[coordenada])){

         if(xxImpulsosCont < abs(xxImpulsos[coordenada])){

            currentMicros = micros();

           if(currentMicros - xxPreviousMicros >
pulse){       // fornece uma onda quadrada
             xxPreviousMicros =
currentMicros;                // em que cada subida
             xxImpulsosCont++;
             xxClkState = (xxClkState == LOW) ? HIGH :
LOW;    // corresponde a um passo
             digitalWrite(xxClkPin,
xxClkState);              // do motor
            }
          }

         if(yyImpulsosCont < abs(yyImpulsos[coordenada])){

            currentMicros = micros();

           if(currentMicros - yyPreviousMicros >
pulse){       // fornece uma onda quadrada
             yyPreviousMicros =
currentMicros;                // em que cada subida
             yyImpulsosCont++;
             yyClkState = (yyClkState == LOW) ? HIGH :
LOW;    // corresponde a um passo
             digitalWrite(yyClkPin,
yyClkState);              // do motor
            }
          }
        }
      }
      break;
```

```
    case 2:

      if(coordenada > 0){

        xxDirState = (xxImpulsos[coordenada] > 0) ? HIGH :
LOW;
       digitalWrite(xxDirPin, xxDirState);

        yyDirState = (yyImpulsos[coordenada] > 0) ? HIGH :
LOW;
       digitalWrite(yyDirPin, yyDirState);

       while(xxImpulsosCont < abs(xxImpulsos[coordenada]) ||
yyImpulsosCont < abs(yyImpulsos[coordenada])){

         if(xxImpulsosCont < abs(xxImpulsos[coordenada])){

            currentMicros = micros();

           if(currentMicros - xxPreviousMicros >
pulse){       // fornece uma onda quadrada
             xxPreviousMicros =
currentMicros;                // em que cada subida
             xxImpulsosCont++;
             xxClkState = (xxClkState == LOW) ? HIGH :
LOW;    // corresponde a um passo
             digitalWrite(xxClkPin,
xxClkState);              // do motor
            }
          }

         if(yyImpulsosCont < abs(yyImpulsos[coordenada])){

            currentMicros = micros();

           if(currentMicros - yyPreviousMicros >
pulse){       // fornece uma onda quadrada
             yyPreviousMicros =
currentMicros;                // em que cada subida
             yyImpulsosCont++;
```

```
            yyClkState = (yyClkState == LOW) ? HIGH :
LOW;    // corresponde a um passo
            digitalWrite(yyClkPin,
yyClkState);              // do motor
           }
         }
        }
        coordenada--;
      }
      break;
  }
}



//////////      Modo Serial     //////////

void recvWithEndMarker() {    // Recebe e guarda o que foi
inserido pelo utilizador

    static byte ndx = 0;
    char endMarker = '\n';
    char rc;

    while (Serial.available() > 0 && newData == false) {
        rc = Serial.read();

        if (rc != endMarker) {
            receivedChars[ndx] = rc;
            ndx++;
            if (ndx >= numChars) {
                ndx = numChars - 1;
            }
        }
        else {
            receivedChars[ndx] = '\0'; // terminate the string
            ndx = 0;
            newData = true;
        }
    }
}
```

```
//============

void parseData() {          // Identifica o tipo de movimento
que o utilizador pretende
                            // e divide o que foi inserido em
duas variáveis (x e y)

  if(receivedChars[0] == '+' || receivedChars[0] == '-'){
    modo = 0;
  }
  else{
    modo = 1;
  }

    char * strtokIndx; // this is used by strtok() as an index

    strtokIndx = strtok(tempChars,",");      // get the first
part
    xx = atoi(strtokIndx);                // convert it to an
integer

    strtokIndx = strtok(NULL, ","); // this continues where
the previous call left off
    yy = atoi(strtokIndx);     // convert this part to an
integer
}


//============

void showParsedData() {         // controla o movimento dos
motores

  unsigned long xxPreviousMicros = 0;
  unsigned long xxImpulsosZero = (abs(xx) * stepping *
passosPorRot) / passoMesa;     // Num de impulsos para o
primeiro tipo de movimento
  unsigned long xxImpulsosUm = (abs(xx - xxPosAtual) *
stepping * passosPorRot) / passoMesa;       // Num de
impulsos para o segundo tipo de movimento
```

```
  unsigned long xxImpulsosCont = 0;
  unsigned long yyPreviousMicros = 0;
  unsigned long yyImpulsosZero = (2 * abs(yy) * stepping *
passosPorRot) / passoMesa;
  unsigned long yyImpulsosUm = (2 * abs(yy - yyPosAtual) *
stepping * passosPorRot) / passoMesa;
  unsigned long yyImpulsosCont = 0;
  int xxPosAtualBackup = 0;
  int yyPosAtualBackup = 0;

  switch(modo){
    case 0:

      xxPosAtualBackup = xxPosAtual;    // Guarda os valores
da posição anterior, para, caso o utilizador
      yyPosAtualBackup = yyPosAtual;    // tente sair dos
limites da mesa, se repor com os valores anteriores

      xxPosAtual = xxPosAtual + xx;
      yyPosAtual = yyPosAtual + yy;

      if(xxPosAtual < 0 || xxPosAtual > 700 || yyPosAtual < 0
|| yyPosAtual > 700){   // Se o utilizador tentar sair dos
limites
        Serial.println("Erro - fora dos limites da mesa
linear.");                    // da mesa o case statement
pára antes dos
        xxPosAtual =
xxPosAtualBackup;
   // motores rodarem e é mostrada uma mensagem
        yyPosAtual =
yyPosAtualBackup;
   // de erro
        break;
       }

      if(xx < 0){
        Serial.print("A recuar ");
        Serial.print(abs(xx));
        Serial.print(" mm em xx ");
       }
```

```
      else if(xx == 0){
        Serial.print("A manter a posição em xx ");
      }

      else{
        Serial.print("A avançar ");
        Serial.print(abs(xx));
        Serial.print(" mm em xx ");
      }

      if(yy < 0){
        Serial.print("e a recuar ");
        Serial.print(abs(yy));
        Serial.print(" mm em yy. ");
      }

      else if(yy == 0){
        Serial.print("e a manter a posição em yy ");
      }

      else{
        Serial.print("e a avançar ");
        Serial.print(abs(yy));
        Serial.print(" mm em yy. ");
      }

      xxDirState = (xx > 0) ? LOW : HIGH;
     digitalWrite(xxDirPin, xxDirState);

      yyDirState = (yy > 0) ? LOW : HIGH;
     digitalWrite(yyDirPin, yyDirState);

     while(xxImpulsosCont < xxImpulsosZero || yyImpulsosCont
< yyImpulsosZero){

       if(xxImpulsosCont < xxImpulsosZero){

          currentMicros = micros();

         if(currentMicros - xxPreviousMicros > pulse){
```

```
// fornece uma onda quadrada
           xxPreviousMicros = currentMicros;
// em que cada subida
           xxImpulsosCont++;
           xxClkState = (xxClkState == LOW) ? HIGH : LOW;
// corresponde a um passo
          digitalWrite(xxClkPin, xxClkState);
// do motor
          }
        }

       if(yyImpulsosCont < yyImpulsosZero){

         currentMicros = micros();

         if(currentMicros - yyPreviousMicros > pulse){
// fornece uma onda quadrada
           yyPreviousMicros = currentMicros;
// em que cada subida
           yyImpulsosCont++;
           yyClkState = (yyClkState == LOW) ? HIGH : LOW;
// corresponde a um passo
          digitalWrite(yyClkPin, yyClkState);
// do motor
          }
        }
      }

     Serial.print("Posição atual: (x,y) = (");
     Serial.print(xxPosAtual);
     Serial.print(", ");
     Serial.print(yyPosAtual);
     Serial.println(") [mm].");

   break;

   case 1:

     if(xx < 0 || xx > 700 || yy < 0 || yy >
700){                  // Se o utilizador tentar sair dos
limites
```

```
       Serial.println("Erro - fora dos limites da mesa
linear.");    // da mesa, o case statement pára antes

break;
// dos motores rodarem e é mostrada uma mensagem de erro
      }

      Serial.print("A deslocar-se para (x, y) = (");
      Serial.print(xx);
      Serial.print(", ");
      Serial.print(yy);
      Serial.print(") [mm]. ");

      xxDirState = (xx - xxPosAtual > 0) ? LOW : HIGH;
     digitalWrite(xxDirPin, xxDirState);

      yyDirState = (yy - yyPosAtual > 0) ? LOW : HIGH;
     digitalWrite(yyDirPin, yyDirState);

     while(xxImpulsosCont < xxImpulsosUm || yyImpulsosCont <
yyImpulsosUm){

       if(xxImpulsosCont < xxImpulsosUm){

         currentMicros = micros();

         if(currentMicros - xxPreviousMicros > pulse){
// fornece uma onda quadrada
           xxPreviousMicros = currentMicros;
// em que cada subida
           xxImpulsosCont++;
           xxClkState = (xxClkState == LOW) ? HIGH : LOW;
// corresponde a um passo
          digitalWrite(xxClkPin, xxClkState);
// do motor
         }
        }

       if(yyImpulsosCont < yyImpulsosUm){

         currentMicros = micros();
```

```
          if(currentMicros - yyPreviousMicros > pulse){
// fornece uma onda quadrada
            yyPreviousMicros = currentMicros;
// em que cada subida
            yyImpulsosCont++;
            yyClkState = (yyClkState == LOW) ? HIGH : LOW;
// corresponde a um passo
           digitalWrite(yyClkPin, yyClkState);
// do motor
          }
        }
      }

      xxPosAtual = xx;
      yyPosAtual = yy;

      Serial.print("Posição atual: (x,y) = (");
      Serial.print(xxPosAtual);
      Serial.print(", ");
      Serial.print(yyPosAtual);
      Serial.println(") [mm].");
  }
}
```