Faculdade de Engenharia da Universidade do Porto

Impressora 3D com LinuxCNC

João Filipe Oliveira Gonçalves

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Major Automação

Orientador: Professor Mário Jorge Rodrigues de Sousa

Outubro de 2014

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Impressora 3D com LinuxCNC"

foi aprovada em provas realizadas em 26-09-2014

o júri

Presidente Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Paulo Bacelar Reis Pedreiras
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor Mário Jorge Rodrigues de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - João Filipe Oliveira Gonçalves

Faculdade de Engenharia da Universidade do Porto

Resumo

A automatização dos processos industriais tem um papel fundamental na engenharia moderna, principalmente após a Segunda Guerra Mundial. A sistematização da manufatura tem como objetivo a criação de sistemas capazes de produzir em grandes quantidades de forma rápida, precisa e com o menor custo possível. A introdução das máquinas de Controlo Numérico Computorizado (CNC), nos anos 50, permitiu elevar ainda mais a fasquia ao nível da produção. Mais recentemente, a impressão a três dimensões tem assumido um papel fundamental na indústria da manufatura, ao revelar-se uma tecnologia extremamente versátil e economicamente rentável.

Neste contexto, o projeto teve como objetivo usar o LinuxCNC, *software open-source* de controlo de máquinas CNC convencionais de corte, para controlar uma impressora 3D. Assim, pretendeu-se dotar o *software* da capacidade de controlo de uma ferramenta de extrusão, algo que não faz nativamente. Para isso, foi essencial perceber o modo como o LinuxCNC funciona e como está organizado internamente.

Numa fase mais avançada do projeto, foi necessário proceder ao dimensionamento dos vários componentes que constituem uma impressora 3D. Por outras palavras, esta etapa consistiu em analisar o material disponível para a realização da dissertação e verificar a viabilidade deste face ao problema proposto. Neste particular, o projeto RepRap, que consiste na criação de impressoras 3D que se auto-replicam, também se revelou de extrema importância ao permitir integrar com LinuxCNC, os componentes responsáveis pelos movimentos e o componente relacionado com a extrusão.

No final, precedeu-se à configuração do sistema, implementação de um conjunto *scripts* e criação de uma interface gráfica que permitiram o controlo e monotorização da impressora 3D diretamente a partir do LinuxCNC.

Palavras-chave: Impressora 3D, LinuxCNC, RepRap, CNC.

Abstract

The automation of industrial processes has been a major goal of modern engineering, particularly after World War II. The objective of systematizing manufacture is to create systems that are able to produce in large quantities, quickly and with precision, but with less costs. The introduction of Computer Numerical Control (CNC), in the 1950s, allowed the enhancement of production rates. More recently, three dimensional printing has taken a decisive role in the manufacturing industry by revealing itself as an extremely versatile and economically profitable technology.

As such, the purpose of this project was to use LinuxCNC, an open-source software which allows for the controlling of a regular CNC cutting machines, in order to control a 3D printer. Thus, it was intended to give the software the ability to control an extrusion tool, something of which it isn't capable of by default. In order to accomplish it, understanding the way LinuxCNC works, and how is internally organized, was of fundamental importance.

At a later stage of the undertaking, it was necessary to proceed to the sizing of the many components which form a 3D printer. In other words, this phase was about analyzing the available resources to work on the dissertation, and to check its practicability in the proposed problem. In this particular, the RepRap project, which is about developing self-replicating 3D printers, showed its great importance by allowing the integration of the components responsible for the movements and the extrusion, with LinuxCNC.

At the final stages, the system configuration took place, by implementation of a set of scripts, as well a graphical interface which enabled the controlling and monitoring of the 3D printer, solely through LinuxCNC.

Keywords: 3D printer, LinuxCNC, RepRap, CNC

Agradecimentos

A todos os que direta ou indiretamente contribuíram para o meu sucesso académico, o meu muito obrigado.

Em particular, começo por agradecer ao Professor Mário de Sousa por todo o apoio, incentivo e disponibilidade demostrada durante a realização da dissertação.

Aos amigos que sempre estiveram presente deixo um abraço em forma de agradecimento pelo companheirismo, apoio e momentos de diversão.

Não podia deixar de dar uma palavra especial à minha namorada Joana pelo incentivo incondicional. Agradeço a paciência, apoio e carinho demostrado em todos os momentos.

Estou grato a toda a minha família por acreditarem em mim desde sempre e pela pronta ajuda em qualquer situação.

Por fim, o meu maior agradecimento vai para os meus pais por tornarem este meu sonho possível. Força, incentivo, apoio, encorajamento, confiança, paciência, compreensão, carinho, dedicação, preocupação. Obrigado por tudo.

Junto a minha irmã e o meu sobrinho ao agradecimento anterior e a todos eles dedico esta dissertação.

"Mastering others is strength.
Mastering yourself is true power"

Lao Tzu, Tao Te Ching

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

3D	Três dimensões
ABS	*Acrylonitrile Butadiene Styrene*
API	*Application Programming Interface*
ATX	*Advanced Technology eXtended*
CAD	*Computer-Aided Design*
CAM	*Computer-Aided Manufacturing*
CD	*Compact Disc*
CI	Circuito Integrado
CNC	Controlo Numérico Computorizado
DC	*Direct Current*
EMC	*Enhanced Machine Controller*
FEUP	Faculdade de Engenharia da Universidade do Porto
FIFO	*First In, First Out*
GUI	*Graphical User Interface*
HAL	*Hardware Abstraction Layer*
HP	*Hewlett Packard*
I2C	*Inter-Integrated Circuit*
ICSP	*In-Circuit Serial Programming*
IDC	*Insulation-Displacement Connector*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
I/O	*Input/Output*
ISP	*In-System Programming*
LCD	*Liquid-Crystal Display*
LED	*Light-Emitting Diode*
MIEEC	Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

NC	*Numerical Control*
NEMA	*Nation Electrical Manufactures Association*
NGC	*Next Generation Controller*
NML	*Neutral Message Language*
MPR	Milímetros Por Rotação
MOSFET	*Metal-Oxide-Semiconductor Field-Effect Transistor*
PC	*Personal Computer*
PID	*Proportional-Integral-Derivative*
PLC	*Programmable Logic Controller*
PPM	Passos por milímetro
PTFE	Politetrafluoretileno
PWM	*Pulse-Width Modulation*
OAC	*Open Architecture Controller*
RAM	*Random-Access Memory*
RTAI	*Real-Time Application Interface*
RTLinux	*Real-Time Linux*
TTL	*Transistor-Transistor Logic*
US	*United States*
USB	*Universal Serial Bus*

Capítulo 1

Introdução

A presente dissertação intitulada "Impressora 3D com LinuxCNC" surge no âmbito da unidade curricular Dissertação, presente no plano de estudos do Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC), da Faculdade de Engenharia da Universidade do Porto (FEUP). O objetivo da unidade curricular foca-se na investigação e desenvolvimento de um projeto científico, visando a aplicação dos conhecimentos adquiridos ao longo do percurso académico para a resolução de problemas complexos, na área de engenharia.

Neste capítulo inicial é elaborada a apresentação do tema de dissertação, juntamente com o seu enquadramento e motivação. Na fase seguinte serão expostos os principais objetivos e metas a atingir com o seu desenvolvimento. Posteriormente, será abordada a metodologia usada para a sua realização. Por fim, procede-se à descrição da estrutura do documento da dissertação com destaque para os principais capítulos que o constituem.

1.1 - Breve introdução histórica

Depois da devastação originada pela segunda grande guerra, o aumento da produtividade da indústria começou a ter um papel ainda mais fundamental que até então. John Parsons, considerado o pai da segunda revolução industrial, em conjunto com o Massachusetts Institute of Technology, introduziu nos anos 50 a primeira máquina NC (*Numerical Control*) com o objetivo de automatizar um conjunto de tarefas repetidas sistematicamente e que se pretendiam ser executadas de forma rápida, precisa e com menores custos [1, 2].

Com a evolução da tecnologia, nomeadamente com a integração do computador na indústria, abriu-se a possibilidade de estender as capacidades das máquinas NC, nomeadamente, através da incorporação de um componente computorizado dedicado ao controlo numérico, nascendo, assim, a máquina CNC (*Computer Numerical Control*) [1, 2]. Estas máquinas funcionam sobre um princípio subtrativo, isto é, a criação de um objeto dá-se através da maquinação sobre um material sólido existente até este adquirir a forma desejada.

Com o passar dos anos, a tecnologia CNC foi evoluindo de acordo com os sucessivos avanços tecnológicos. O aparecimento dos primeiros sistemas CAD (*Computer-Aided Design*) a três dimensões desencadeou o sucesso definitivo do controlo numérico.

Paralelamente, mas não alheio aos desenvolvimentos da máquina CNC, em 1983 foram dados os primeiros passos em direção à impressão a três dimensões por Charles 'Chuck' Hull, com a invenção da Estereolitografia [3]. Esta nova tecnologia aditiva consistia na conceção de um modelo tridimensional através da aplicação sucessiva de camadas de um designado material. Desta forma, não só era replicado o aspeto de um objeto, como também toda a sua funcionalidade. Assim, em vez de ser maquinado um material pré-existente para se obter o objeto pretendido, este era criado através da deposição do material de trabalho, evitando assim uma grande quantidade de desperdício.

Seguindo a tendência da evolução tecnológica, com os preços da tecnologia envolvida nos sistemas de impressão a três dimensões a diminuir e com o término da validade de algumas patentes [4], nestes últimos anos tem-se observado uma utilização exponencial deste tipo de soluções nas mais variadas áreas, desde uso militar para o desenvolvimento de armas, na criação de peças para industria automóvel, em várias frentes na medicina e até a nível alimentar. As potencialidades de utilização são infindáveis, o que permite antever, num futuro próximo, a completa democratização da tecnologia, o que permitirá aumentar a sua utilização, até mesmo a nível doméstico [5, 6]. Mais, esta já é considerada por muitos a origem da terceira revolução industrial [6-8], estimando-se que, com a entrada de novos e importantes *players* no mercado, a indústria relacionada com a criação de produtos em impressoras 3D atinja os 10,8 mil milhões de dólares em 2021 [9-11].

1.2 - Enquadramento e Motivação

O Controlo Numérico Computorizado obedece a um conjunto de processos sequenciais de modo a executar o conjunto de operações pretendidas. No fim da cadeia de ferramentas utilizadas, encontra-se o cérebro de um sistema CNC, o *software* de controlo. O primeiro programa de controlo a respeitar uma arquitetura "aberta" foi o EMC (*Enahanced Machine Controller*), que posteriormente mudou o nome para LinuxCNC. Esta trata-se de uma das soluções mais usadas para o controlo de máquinas industriais de fresagem e torneamento. Este *software* permite controlar uma máquina CNC, tanto a nível da movimentação como ao nível da ferramenta de trabalho

Por outro lado, face aos mais recentes desenvolvimentos, a impressora 3D já se tornou numa máquina de importância inegável para futuro da manufatura. Por essa razão, o uso de uma ferramenta familiar à indústria da manufatura - como é o LinuxCNC - para o controlo desta tecnologia aditiva mostra-se não só interessante, como de extrema utilidade. No entanto, o LinuxCNC não apresenta nativamente nenhum método de controlo de uma máquina de manufatura aditiva. Com isto, a dissertação apresentada procura estudar e colmatar essa lacuna de forma a controlar e monitorizar a impressora 3D desenvolvida.

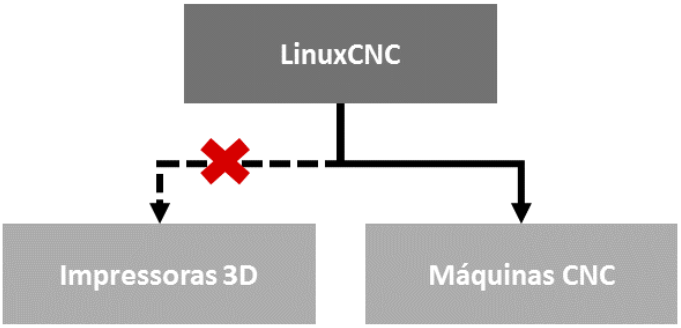


Figura 1.1 – LinuxCNC não está nativamente preparado para controlar impressoras 3D

1.3 - Objetivos

A principal finalidade da dissertação descrita neste documento é a criação de uma plataforma funcional que opera como impressora 3D, usando para o seu controlo o *software* LinuxCNC.

No entanto, no desenvolvimento de um projeto de engenharia é essencial o estabelecimento prévio de um conjunto de objetivos para atingir a meta estabelecida. De acordo com o enunciado e as metas propostas pelo orientador do projeto, destacam-se os principais objetivos da dissertação:

- Estudar dos componentes constituintes de uma impressora 3D, alternativas e viabilidade de utilização do material existente para a realização do projeto;
- Analisar e descrever o modo de funcionamento do LinuxCNC, nomeadamente a forma como está estruturado internamente;
- Configurar o LinuxCNC de modo a dotá-lo da capacidade de controlo e monitorização da ferramenta de extrusão;
- Integrar, testar e validar dos componentes constituintes da impressora 3D desenvolvida.

Alcançando cada um destes objetivos individuais, espera-se atingir a meta inicialmente proposta.

1.4 - Metodologia

Ao desenvolvimento de um projeto de engenharia deve estar associado uma metodologia bem definida desde o seu início de forma a maximizar o sucesso do mesmo.

A metodologia usada é baseada no processo de engenharia de sistemas, método cascata. Nesta metodologia, o principal objetivo é transformar as necessidades, requisitos e objetivos num produto final. No fim de cada nova fase dá-se o início de uma outra.

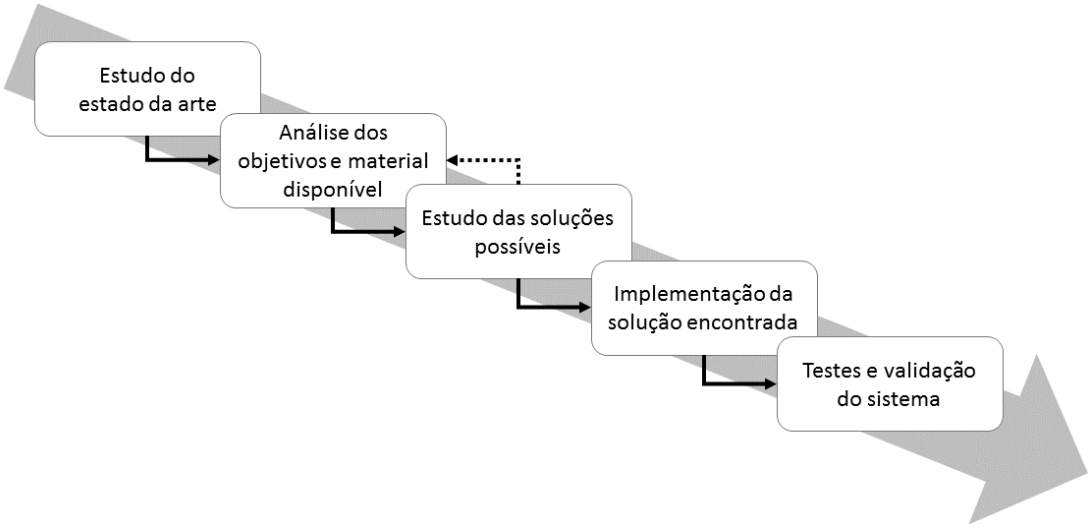


Figura 1.2 – Principais fases do projeto desenvolvido

Dado o problema inicial, é necessário realizar um estudo do estado da arte, de acordo com o sistema a desenvolver. Neste caso, procedeu-se à análise dos principais componentes que constituem o sistema de forma a compreender como deve ser efetuada a integração deste no desenho final.

Através da análise dos objetivos e do material disponível para o projeto, tendo já realizada a pesquisa de mercado, é possível estudar quais as soluções viáveis para resolver o problema proposto. Na fase do estudo das soluções possíveis, por vezes pode ser necessário regressar à fase anterior para verificar se a solução pensada respeita os objetivos e se maximiza o material disponibilizado.

Na fase final do projeto, procede-se à implementação da solução encontrada para o desenvolvimento do projeto e são feitos os testes e validações finais, essenciais a qualquer projeto de engenharia.

1.5 - Estrutura do documento

A organização do documento de dissertação é fundamental para a correta interpretação do trabalho desenvolvido ao longo do projeto. Neste sentido, na presente secção é apresentada a estrutura do documento e descrito o objetivo de cada um dos principais capítulos.

O documento apresenta 6 grandes capítulos. No primeiro capítulo, onde está inserida esta secção, é feita uma introdução através da exposição do enquadramento histórico, da motivação para o projeto, dos objetivos da dissertação e a metodologia utilizada para o seu desenvolvimento.

Por sua vez, o segundo capítulo - "Estado da arte" - pretende demostrar uma revisão bibliográfica dos principais elementos que constituem uma impressora 3D. Este capítulo encontra-se dividido em secções, cada uma das quais versa sobre o estudo de um dos componentes. No final também é feita a análise a um projeto de impressora 3D *open-source*.

O terceiro capítulo, intitulado "LinuxCNC", aborda as principais características e modo de funcionamento do *software* controlador exigido para a realização do projeto. Numa primeira

fase é feito um enquadramento do programa e, posteriormente, analisados os módulos constituintes.

O capítulo 4 corresponde ao "Dimensionamento da impressora 3D", onde é analisada e avaliada a viabilidade do material disponibilizado para o projeto. Nesta fase é discutido e comentado o dimensionamento de cada um dos componentes.

A integração e configuração da impressora 3D é exposta no capítulo 5. Este encontra-se dividido em 3 secção seguinte encontra-se descrita a solução proposta para realizar o controlo da ferramenta de extrusão pelo LinuxCNC. Por fim, são apresentados os testes finais realizados para demonstração do funcionamento dos módulos desenvolvidos.

As conclusões e propostas de trabalho futuro encontram-se expostas no capítulo 6, onde se faz uma análise critica às soluções implementadas e resultados obtidos.

Quase a finalizar o documento aparecem os anexos, que consistem num estudo mais aprofundado, realizado no contexto da dissertação, sobre motores passo-a-passo.

A lista de referências consultadas durante a dissertação encerram o documento.

Capítulo 2

Estado da arte

A prototipagem rápida, uma forma de manufatura aditiva, é nome que se dá à técnica usada pelas impressoras 3D para a criação de um objeto. O *modus operandi* consiste no depósito de um material sólido através de aplicações sucessivas de camadas desse mesmo material, de acordo com um modelo computacional previamente elaborado [12]. Trata-se de um tipo de manufatura alternativa à mais abundantemente conhecida manufatura subtrativa, que consiste na remoção de material sólido disponível para laborar.

Na verdade, a manufatura aditiva e subtrativa partilham os mesmos fundamentos básicos de funcionamento, muitos dos mesmos princípios e ferramentas, sendo a grande exceção a ferramenta de trabalho (corte/extrusão). Qualquer uma destas tecnologias exige um estudo multidisciplinar que vai desde a unidade de controlo para elaboração e interpretação do modelo tridimensional do objeto; a eletrónica capaz de reconhecer os comandos de controlo e atuarem nos motores e ferramentas, sendo estes últimos, por sua vez, responsáveis por fazer a máquina mover-se e criar o objeto.

Assim, a figura 2.1 apresenta de modo global as ferramentas e plataformas que formam a impressora 3D desenvolvida no projeto de dissertação descrito neste documento, bem como, de modo mais amplo, a maioria dos instrumentos presentes numa máquina CNC convencional [13]. A partir do esquema da figura 2.1 é feito um estudo do estado da arte dos componentes constituintes da impressora 3D desenvolvida, em que cada componente merece um subcapítulo onde é investigado e descrito o seu funcionamento, para além de serem analisadas alternativas no mercado que desempenham esse mesmo papel.

De acordo com o supracitado, este capítulo está dividido em 6 partes. De início é feita um estudo ao Controlo Numérico Computorizado, nomeadamente às ferramentas que tornam possível o controlo de uma máquina CNC (incluindo impressoras 3D), apresentando alternativas existentes no mercado. Na fase seguinte é apresentada a eletrónica de condução e controlo responsáveis por energizar os motores e controlar a ferramenta de trabalho, respetivamente. Posteriormente são analisados o tipo de motores mais usados para mover impressoras 3D, os motores passo-a-passo, sem deixar de serem avaliadas outras opções. No ponto 4 é feita uma descrição da ferramenta de trabalho, sendo que neste caso se trata de uma extrusora que expele termoplástico. Foi também elaborada uma análise ao projeto RepRap numa secção, que

visa o desenvolvimento de impressoras 3D *open-source*, e o modo com este projeto pode ajudar à realização da dissertação. Por fim, fez-se uma breve referência a projetos similares ao proposto na dissertação, denominados EMCRepRap.

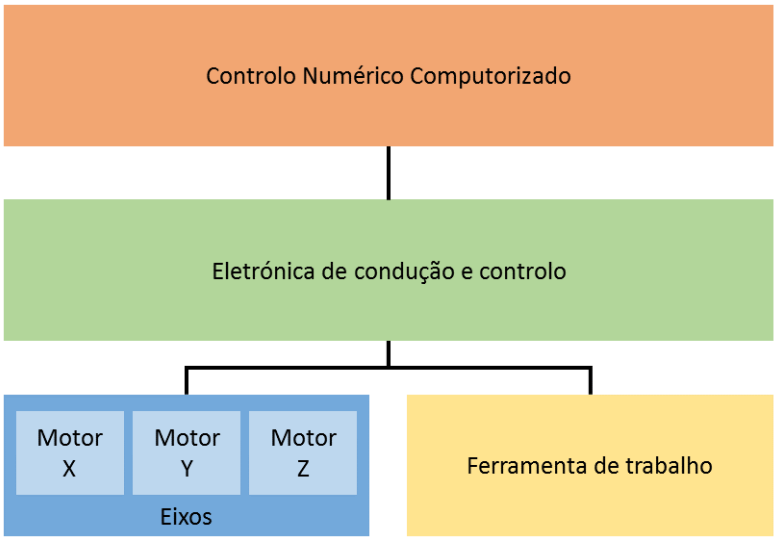


Figura 2.1 - Esquema base dos componentes de estudo de uma máquina CNC

2.1 - Controlo Numérico Computorizado

O Controlo Numérico Computorizado surgiu muito anos antes do aparecimento das impressoras 3D. Contudo, a introdução no mercado da máquina de impressão a três dimensões não alterou muito o panorama do controlo numérico já implementado, nomeadamente no que toca à cadeia de ferramentas necessárias para o controlo de máquinas numéricas. Isto porque, na verdade, tal como já mencionado, apesar de operarem sobre tecnologias diferentes – uma usa tecnologia aditiva e outra subtrativa -, uma impressora 3D respeita quase todas as características de uma vulgar máquina CNC [12, 14].

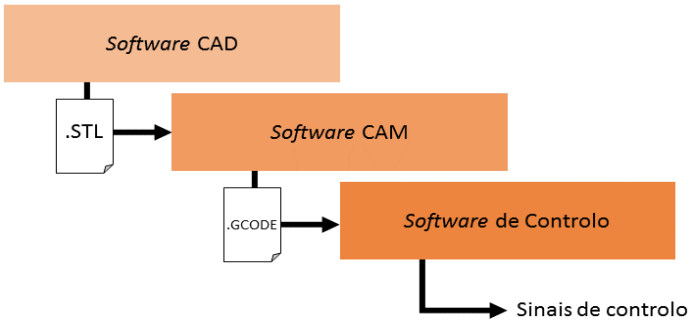


Figura 2.2 - Cadeia de software do Controlo Numérico Computorizado

Ao nível mais alto, ou seja, a nível computacional e de controlo, tanto as máquinas CNC como as impressoras 3D, respeitam a arquitetura de *software* esquematizada na figura 2.2.

Descrevendo a figura, o processo consiste usar um Desenho Assistido por Computador (*software* CAD) com a forma do objeto a criar, converte-lo num conjunto de movimentos e ações (software CAM), que posteriormente são interpretadas por um programa que envia sinais elétricos para a máquina, fazendo a interface com o mundo real (s*oftware* de controlo) [1].

2.1.1 - *Software* CAD/CAM

Na cadeia de ferramentas da arquitetura de um sistema de Controlo Numérico Computorizado, os programas de *Computer-Aided Design* (CAD) e *Computer-Aided Manufacturing* (CAM) fazem parte do topo da pirâmide e estão associados aos primeiros passos da prototipagem. Segundo Chennakesava R. Alavala em [15], um sistema CAD pode ser definido como uma ferramenta capaz, através dos recursos gráficos de um computador e técnicas sofisticadas, ajudar a desenvolver e analisar o modelo de uma objeto. Por sua vez, o mesmo autor define um sistema CAM como um sistema capaz de usar os recursos computacionais para planear, gerir e controlar as operações de um plano de fabrico. Por outras palavras, estes dois elementos da cadeia de *software* são os responsáveis pela criação do objeto e conversão deste em código a ser interpretado pela máquina, respetivamente.

Estes sistemas são bastante utilizados na indústria nas mais variadas áreas. No que toca ao controlo numérico, a função deste conjunto é possibilitar desenvolver código RS-274D (mais conhecido como G-code), a linguagem padrão para máquinas CNC, diretamente a partir da geometria do modelo do objeto [15]. Tipicamente o formato de ficheiro exportado pelo sistema CAD tem extensão STL (que deve o seu nome à tecnologia de manufatura aditiva *STeroLithography)*, por se tratar do formato mais abundantemente utilizado como *input* pelos programas de CAM para impressão a 3D [13].

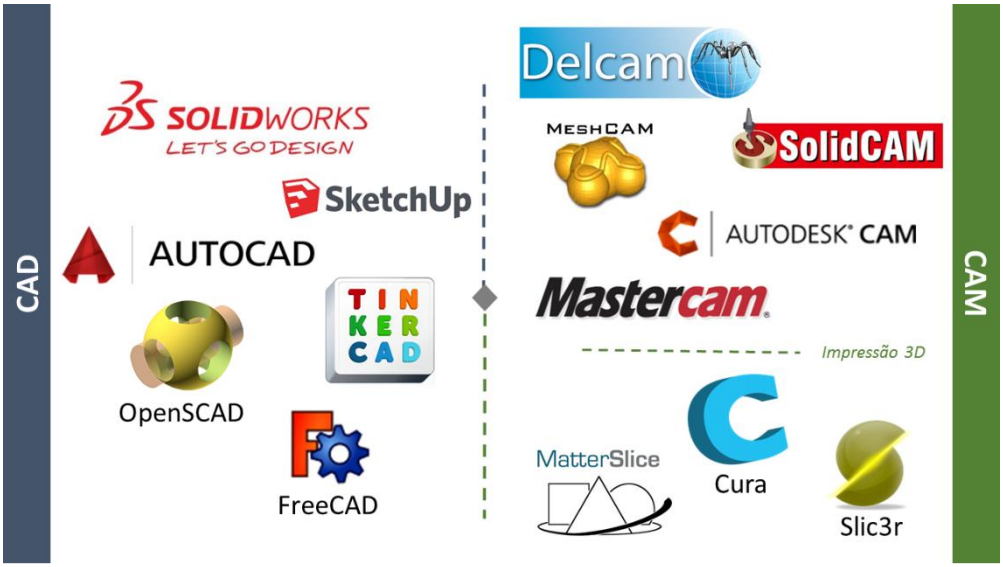


Figura 2.3 - Alguns exemplos de sistemas CAD e CAM

A utilização dos *software* CAM com impressoras 3D ou com máquinas CNC típicas é, a nível conceptual e do ponto de vista do utilizador, indiferenciada. Neste programa a geometria de um objeto é convertida num conjunto de movimentos e ações da ferramenta, que a máquina

deve seguir para criar esse mesmo objeto. Contudo, como o modo de criação da peça é diferente, o código que lhe dá origem também o será. Por essa razão, na realidade existem opções de *software* específicos para utilização com máquinas de extrusão ou máquinas de corte.

Dado que a impressão 3D é realizada através da aplicação de um dado material em camadas sucessivas, os *software* CAM para impressoras 3D são denominados de "*Slicers*", pois o modelo elaborado em CAD é cortado em "fatias", constituindo as camadas que formam o objeto. MatterSlice, Skeinforge, Cura, Slic3r são algumas das alternativas de programas CAM para impressoras 3D.

2.1.2 - *Software* de Controlo

O controlador é o cérebro de todo o sistema. Ele é o responsável por interpretar o código vindo do sistema CAM - convencionalmente trata-se do G-code - contendo as operações necessárias para imprimir um objeto e enviar para o mundo físico os sinais elétricos correspondentes. Tipicamente estes sinais são de passo/movimento, que faz o motor mover-se, e de direção, que diz qual o sentido do movimento.

Para o controlo numérico existem um sem-número de opções para *software* de controlo, no entanto, devido à natureza do projeto destaca-se a arquitetura *open-source*, a qual se iniciou com um projeto que finalmente deu origem ao LinuxCNC [16].

2.1.2.1 - LinuxCNC

O LinuxCNC, inicialmente designado por EMC - *Enhanced Machine Controller* -, desenvolvido pelo *US National Institute of Standards and Technology*, é um *software open-source* que implementa a capacidade de controlo numérico, em tempo-real, usando um computador convencional, para controlar uma grande variedade de máquinas autónomas, sendo o seu principal foco as máquinas CNC. Segundo a página oficial do projeto na internet [17], este oferece:

- Interpretador de linguagem de programação de máquinas, o RS-274 (mais conhecido como G-code);
- Um sistema de planeamento de movimentos em tempo-real;
- Operação de eletrónica de baixo nível como sensores e drivers de motor;
- Uma *breadboard* virtual, para criar uma configuração única para cada uso;
- Um *software* PLC programável com diagramas *ladder*;
- Várias interfaces gráficas para o utilizador;
- *Live-CD* e extensões tempo-real pré-compiladas;
- A possibilidade de controlar até 9 eixos e com suporte para várias interfaces;
- Capacidade de controlar servos (analógico ou PWM) em malha fecha pelo *software*, e motores de passo-a-passo;
- Movimento sincronizado dos eixos, controlo de velocidade constante, desvio com tolerância máxima, compensação de tamanho;
- Gerador de velocidade trapezoidal;
- Suporte para sistemas de não-cartesianos, incluindo robôs SCARA e PUMA.

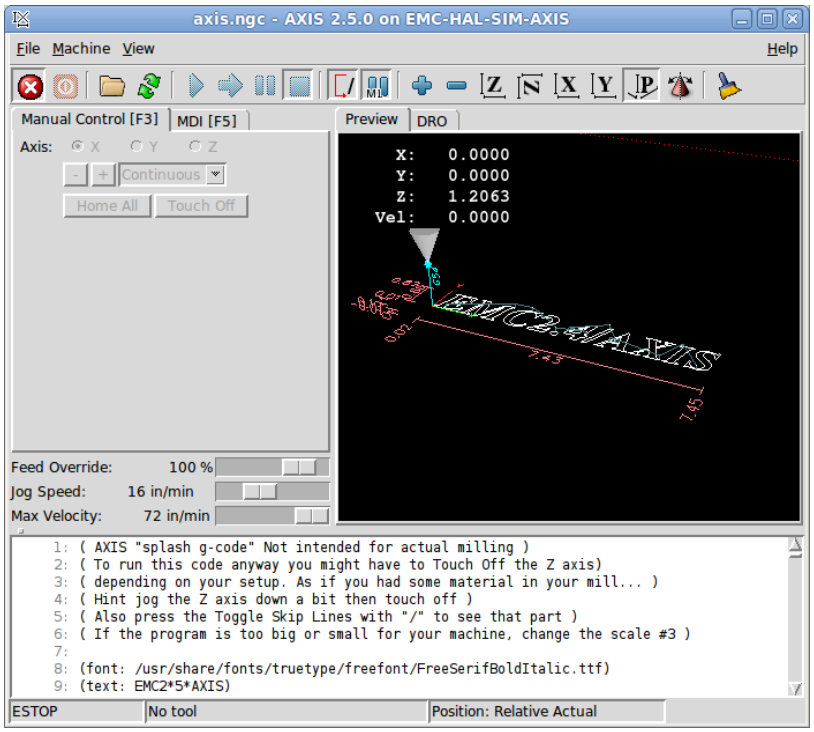


Figura 2.4 - Interface Axis: interface padrão do LinuxCNC

O LinuxCNC é "apenas" um sistema de controlo e por essa razão não oferece funções de desenho nem de geração de código a partir do modelo da peça. Para isso é necessário usar os programas CAD e CAM presentes na parte superior da cadeia de software do Controlo Numérico Computorizado, tal como foi analisado na secção 2.1.1. Mais ainda, este programa não apresenta funcionalidades pré-configuradas para o controlo direto de uma extrusora, sendo esse uma dos principais desafios da dissertação.

Por se tratar de um requisito do projeto, esta ferramenta será alvo de análise mais detalhada no Capítulo 3.

2.1.2.2 - Alternativas no mercado

Em conformidade o próprio nome da Dissertação, o recurso ao LinuxCNC é inevitável para o projeto. Contudo, não se pode descorar o estudo das alternativas existentes no mercado passíveis de serem usadas para o controlo de uma máquina CNC, inclusive impressoras 3D.

Antes de mais é necessário fazer a distinção entre *software* usados por aqueles que utilizam as máquinas CNC como um *hobby*, ou projetos de pequena escala, e as alternativas usadas por profissionais. No entanto, apesar do LinuxCNC ser tipicamente inserido no primeiro grupo, e por essa razão apenas este conjunto foi alvo de análise, naturalmente não se descarta a sua utilização em ambientes industriais, havendo mesmo fabricantes que o integram nos seus produtos, como é o caso da Sirius Electronic Systems.

Mais, é importante não esquecer que apesar de mundos próximos, há uma tendência para separar o mundo das máquinas CNC do mundo das impressoras 3D e o *software* de controlo não é exceção. O controlo, no que toca à interpretação do código, é em tudo semelhante em ambos os casos. Ainda assim, ao longo do tempo, acompanhando a disseminação da tecnologia de

impressão a três dimensões, foram criadas ferramenta de controlo específicas para estas máquinas. A principal novidade residia no facto de além de controlarem os convencionais movimentos a três dimensões típicos de uma máquina CNC, também permitiam o controlo e monotorização da ferramenta de extrusão.

Em seguida são analisadas duas alternativas ao LinuxCNC: uma alternativa direta para controlo de máquinas CNC convencionais, o Mach3, e uma alternativa adaptada às impressoras 3D, o ReplicatorG.

Mach3

Dentro do número infindável de alternativas, as quais diferem relativamente pouco umas das outras, a mais usada por entusiastas do controlo numérico, de acordo com os resultados de um inquérito *online* realizado em 2012 é o Mach3 da Artsoft [18].

Desenvolvido como um projeto alternativo às versões iniciais do LinuxCNC, foi pensado para ser executado em ambiente Windows, tornando-se mais acessível a um maior número de pessoas, acabando por se tornar no *software* CNC mais popular.

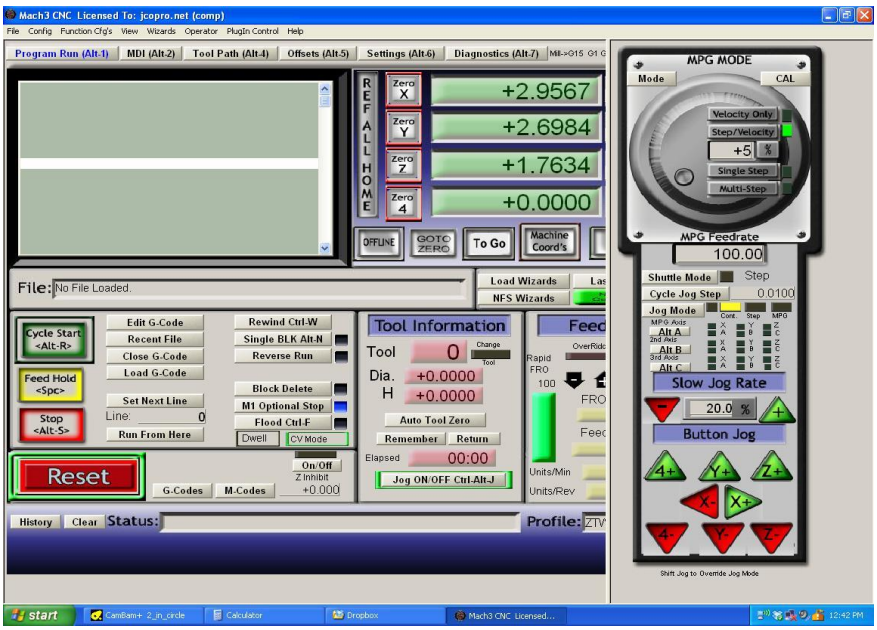


Figura 2.5 - Interface padrão Mach3 da ArtSoft

À imagem do LinuxCNC, torna um simples PC num controlador de máquinas CNC. Destaca-se pela simplicidade de integração com vários *hardware* e pela interface facilmente customizável, apesar de inicialmente um pouco confusa e distrativa. Do mesmo modo que o LinuxCNC, também não desempenha o papel de *software* de CAD e CAM, nem é possível controlar diretamente uma extrusora. No entanto a sua comunidade criou um *add-on*, que permite capacita-lo do controlo da tecnologia aditiva. De forma resumida, através da tabela 2.1, é feita a comparação do Mach3 com o LinuxCNC.

Estas duas opções, o LinuxCNC e Mach3, dominam um mercado que tem ainda dezenas e dezenas de outras outra opções, entre as quais, das que se seguem na lista de mais usadas, o TurboCNC, Eding CNC, USB CNC.

Tabela 2.1 - Comparativo Mach3 e LinuxCNC [17, 19]

	Mach3	LinuxCNC
Sistema Operativo	Windows 2000 ou superior, 32bit	Ubuntu 8.04 ou 10.04 LTS, 32bit
Nº Eixos	Até 6	Até 9
***Input* padrão**	RS-274D (G-code)	RS-274D (G-code)
Interface Gráfica	Customizável e com *wizards* opcionais	Várias alternativas e customizável
Interface de comunicação	Porta paralela ou *hardware* de terceiros	Porta paralela ou *hardware* de terceiros
Tipo de motor	Passo-a-passo e servomotores	Passo-a-passo e servomotores
Suporte extrusora	Não nativamente, mas possível através de *add-on* da comunidade	Não
Requisitos mínimos	1GHz CPU, 512MB RAM, Placa de vídeo com 32MB RAM	400MHz CPU, 256MB RAM
Preço	$175.00	Grátis
Observações	Grande suporte por ser bastante utilizado	Não requer instalação, problemático com portáteis

ReplicatorG

Software open-source, originalmente usado com as impressoras 3D criadas pela MakerBot, com a capacidade de controlar várias impressoras 3D *open-source*, mas também máquinas CNC genéricas [20]. A grande vantagem deste programa reside no facto de trazer integrado um *software* CAM, o Skeinforge, o que permite fazer *upload* direto do ficheiro .STL proveniente do *software* CAD sem necessidade de uma ferramenta intermédia. Com uma interface semelhante ao conhecido Arduino IDE, é multiplataforma, existindo versões para Linux, Windows e Mac.

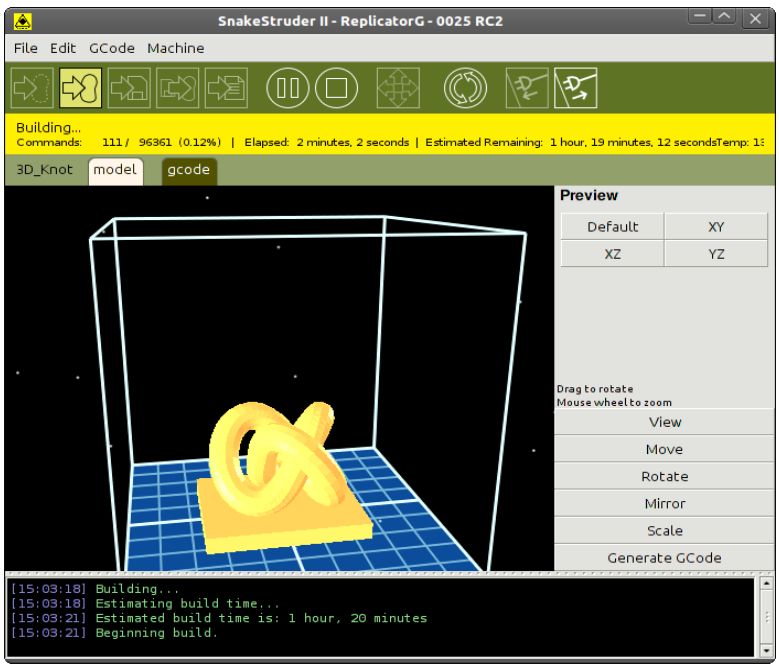


Figura 2.6 - Interface do ReplicatorG

Na verdade, para esta ferramenta funcionar é necessário que a eletrónica da máquina seja capaz de alojar um *firmware* que recebe o G-code enviado pelo ReplicatorG, usualmente debitando o código linha a linha. Devido à natureza das impressoras 3D, em que o controlo normalmente está na eletrónica, este tipo de soluções é bastante recorrente.

De entre as várias ferramentas existentes para executar esta tarefa, pode-se destacar o Proterface, RepSnapper, MatterControl [21].

2.2 - Eletrónica de condução e controlo

Nesta seção é abordado o segundo elemento da cadeia de componentes que constituem uma máquina CNC, a eletrónica de condução. No entanto, neste caso em particular, por se tratar de uma impressora 3D, junta-se ainda o estudo da eletrónica de controlo para a ferramenta de extrusão.

Na primeira parte é feita uma análise sobre o modo de funcionamento, os tipos, as estratégias e os modos de condução dos *drivers* dos motores. Note-se, contudo, que apenas é feita análise aos drivers de motores passo-a-passo, pois, como se poderá ver nos capítulos posteriores e na secção 2.3, são estes o tipo de motores utilizados no projeto.

Num segunda fase é explicado muito brevemente qual a utilidade e o porque da necessidade do uso da eletrónica de controlo em separado do controlo principal.

2.2.1 - Condução dos motores

O controlador numérico envia para o mundo físico os sinais elétricos que estimulam o movimento da máquina. Estes sinais, convencionalmente, tratam-se de sinais de passo e direção de rotação e são interpretados pelos *drivers* de motor, ou circuitos de condução, que por sua vez os converte em impulsos elétricos que são enviados para os terminais dos motores passo-a-passo, numa determinada sequência lógica, fazendo o motor mover-se [22, 23].

Os sinais de passo vêm sob a forma de impulsos elétricos e cada um destes impulsos deve fazer mover o motor um passo, ou seja, uma determinada distância definida pelas características físicas do motor. No que toca ao sinal de direção, ele aparece como um sinal "alto", tipicamente a ronda os 5V ou 3.3V, ou um sinal "baixo", normalmente a rondar os 0V. Conforme esta informação, é ditado o sentido do movimento do motor.

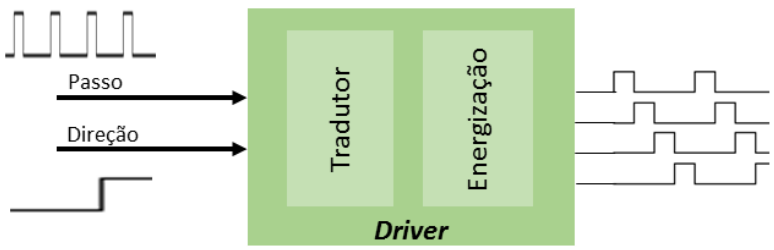


Figura 2.7 - Conversão feita no *driver* do motor

Tipicamente, de modo superficial, pode-se dividir um *driver* de motor de passo-a-passo em dois elementos: o tradutor e a energização, como mostra a figura 2.7. A função do tradutor no *driver* é interpretar os comandos de passo vindo do controlador e, juntamente com o sinal de direção, desencadear uma sequência lógica de excitação do motor. Por sua vez, na fase de energização é fornecida a corrente aos enrolamentos do motor de acordo com a sequência lógica anteriormente determinada. Assim, quando a sequência lógica determina que um enrolamento deve ser energizado, então, idealmente a corrente máxima (específica de cada *driver*) deve ser disponibilizada para esse enrolamento imediatamente, mantendo-se constante até nova ordem do tradutor para desligar esse enrolamento e energizar outro [23]. Deste modo dá-se origem às formas de onda à saída do *driver* da figura 2.7.

2.2.1.1 - Tipos de condução

A conversão é realizada de forma diferente tendo em conta o tipo de *driver* que se utiliza, que por sua vez depende do motor. Deste modo, a escolha pesada do circuito de condução deve ser feita em conjunto, como se de um pacote se tratasse, defende Austin Hughes, em 1990 [23].

Existem dois grandes tipos de circuito de condução: os unipolares e os bipolares, que dependem do tipo de enrolamento do motor passo-a-passo que se quer operar. Na secção 2.3.2, onde estão descritos os tipos de enrolamentos de um motor passo-a-passo, encontra-se informação sobre o modo como o circuito de condução opera e como influencia o comportamento do motor.

2.2.1.2 - Estratégias de condução

Outra das características definidoras de um *driver* é a sua estratégia de condução. De entre as várias opções existentes as mais estudadas são os *drivers* R/L e os *Chopper Drive*.
Os circuitos R/L, também conhecidos como *driver* de tensão constante, aplicam uma tensão em cada enrolamento para definir o passo. Contudo, é a corrente que passa nos enrolamentos que faz mover o motor e é possível obtê-la, pela Lei de Ohm, através dessa mesma tensão aplicada, da indutância L e resistência R dos enrolamentos. A resistência R determina a corrente máxima e a indutância L mostra a velocidade com que é capaz de alterar fluxo da corrente I no enrolamento [22, 23].

Já os circuitos *chopper* são um tipo de *driver* de corrente constante, ou seja, fornecem, sensivelmente, sempre a mesma corrente aos enrolamentos. Estes recebem uma tensão alta para que a corrente nominal seja alcançada rapidamente ($dI/dt = V/L$). Quando o limite superior do intervalo de valores corretos para a corrente é atingido, o fornecimento de energia é "cortado". Depois, pequenas mudanças no valor nominal fazem ativar/desativar o fornecimento de corrente para esta se manter num determinado nível [22, 23].

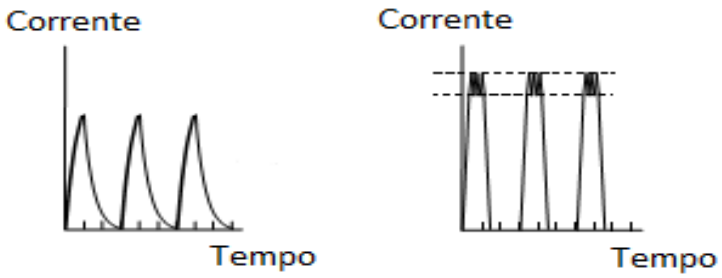


Figura 2.8 - Forma de onda do circuito R/L (esquerda) e c*hopper* (direita) [23]

A figura 2.8 mostra a forma de onda dos dois tipos de estratégias de condução para altas frequências. A forma de onda da direita tem uma aspeto mais semelhante à forma da onda ideal à saída de um *driver (figura 2.7),* podendo-se concluir a superior eficiência dos circuitos *chopper driver* face aos R/L para frequências altas (acima das 20KHz) de chegada de impulsos de passo [23, 24]. A baixas frequências os *chopper drivers* podem causar sons agudos indesejados, são menos eficientes a tensões baixas e perdem para os circuitos R/L na relação entre o preço e a eficiência [24].

2.2.1.3 - Modos de operação

Para lá da conversão dos sinais, também são os *drivers* que afetam o modo de operação dos motores, entre *full step*, *half step* e *microstep,* referenciados na secção 2.3.4, dentro do subcapítulo relativo aos motores passo-a-passo, e no anexo A2 mais pormenorizadamente.

2.2.2 - Controlo da extrusora

Por seu turno, a eletrónica de controlo é responsável por colmatar a lacuna do LinuxCNC para este projeto, ou seja, a impossibilidade de controlo direto e simples da ferramenta de extrusão, por este se tratar de um *software* criado para o controlo de máquinas CNC convencionais de corte.

Na verdade, apesar da uniformidade apresentada até este ponto no estudo apresentado neste documento, a necessidade de utilização de uma placa eletrónica dedicada ao controlo da extrusora é o primeiro facto que realmente distingue a tecnologia de impressão da tecnologia de corte. Isto acontece porque o valor da temperatura lido trata-se de uma grandeza analógica, que precisa de ser convertida para digital de modo a ser interpretado pelo controlador.

Porém, é importante não esquecer que ao controlador se pede que continue a coordenar os movimentos da máquina do mesmo modo que normalmente o faz. Por essa razão, a eletrónica de controlo da extrusora não pode estar alheia ao funcionamento do *software* de controlo original, pois é necessária uma sincronização entre movimentos e deposição de material. Por outras palavras, o componente responsável pelo controlo da extrusora tem de receber comandos do controlador e só aí deve efetivamente controlar os elementos constituintes da extrusora. Do mesmo modo, a eletrónica também fica responsável por comunicar ao controlador as informações necessárias para uma monotorização adequada.

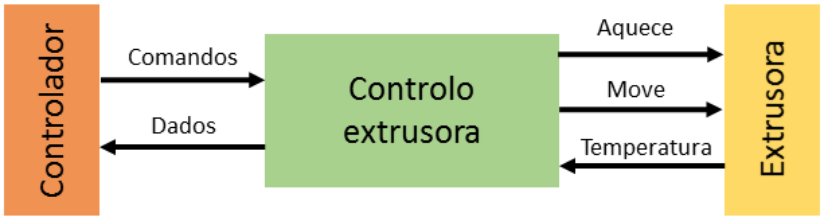


Figura 2.9 - Esquema controlo da extrusora

Neste caso em particular, tratando-se de uma impressora 3D que usa termoplástico como material de extrusão, depreende-se que é necessário haver um controlo da temperatura para que o plástico derreta para ser moldado. Mais, não só é preciso haver o controlo da temperatura, como também é necessário o controlador – tipicamente uma placa Arduíno ou similar – atuar sobre o alimentador de plástico (um motor) para que este puxe o plástico a uma determinada velocidade de acordo com as características da extrusora e do plástico usado. A figura 2.9 apresenta estes conceitos de forma esquemática.

Existem no mercado inúmeras opções para a condução de motores e para o controlo da extrusora é possível utilizar um dos múltiplos microcontroladores disponíveis. Todavia, com o passar do tempo foram aparecendo soluções próprias e convincentes que facilitam muito a escolha e integração destes dois componentes com o resto do sistema. Em concordância, atualmente verifica-se a existência de uma enorme quantidade de soluções estandardizadas e ao mesmo tempo bastante completas. Seguindo esta linha de pensamentos e sendo o projeto de dissertação aqui descrito um projeto que visa o desenvolvimento de uma impressora 3D, torna-se fundamental analisar as soluções de eletrónica de condução e controlo pré-construídas para este fim. O projeto de maior relevo nesta área é o projeto RepRap, analisado na secção 2.5 deste capítulo.

2.3 - Motores passo-a-passo

No fim da cadeia dos componentes principais que constituem uma máquina CNC ou impressora 3D estão os motores (em *ex aequo* com a ferramenta de trabalho). Estes elementos são responsáveis por fazer mover os eixos de acordo com os sinais recebidos dos *drivers*. Para esta tarefa, tipicamente são usados motores passo-a-passo ou servomotores. Por ser abundantemente mais utilizados [25], pelas vantagens elencadas no ponto 2.3.5, e por ter sido o tipo de motores usados, apenas foram analisados ao pormenor os motores passo-a-passo. No entanto, no fim desta secção encontra-se uma breve comparação com os servomotores.

Um motor passo-a-passo é um dispositivo eletromecânico que converte a excitação elétrica em movimentos mecânicos rotacionais. Uma rotação completa do eixo do motor é dividida em pequenos intervalos discretos iguais chamados de "passos", sendo que estes são resultado da aplicação de pulsos elétricos nos terminais do motor. Por outras palavras, o motor passo-a-passo gira com incrementos fixos sempre que é dado um impulso elétrico ao motor, ao contrário

de um motor DC padrão que gira continuamente [22, 26]. Como cada passo move o motor uma distância conhecida é possível determinar o posicionamento de um motor passo-a-passo, sem qualquer mecanismo de feedback, através da quantidade de pulsos aplicados aos seus terminais, numa determinada sequência que também determina o sentido de rotação [23]. Assim, sabendo-se o ângulo do passo é possível descobrir o número de passos necessários para realizar uma rotação completa, assim como encontrar a posição do motor através do número de passos dados, tal como se pode ver pela equação 2.1.

$$Nº\ de\ passos = \frac{Ângulo\ de\ rotação}{Ângulo\ de\ passo} \tag{2.1}$$

Ou seja, por exemplo, se um motor tiver um ângulo de passo de 20º para se dar uma volta completa de 360º são necessários 18 passos. Este facto é uma das mais notáveis vantagens no uso de um motor passo-a-passo, já que permite a sua utilização em aplicações *open-loop*, mesmo que seja necessário o conhecimento da posição do motor, bastando para isso ter em atenção aos impulsos que chegam ao motor.

Por outro lado, o ângulo de passo é determinado pelas características físicas do motor, nomeadamente pelo número de "dentes" do rotor e o número de fases do estator (no caso dos motores com "dentes"), segundo a equação 2.2.

$$Ângulo\ de\ passo = \frac{360^0}{Dentes\ do\ rotor * Fases\ do\ estator} \tag{2.2}$$

Provavelmente o motor mais usado atualmente trata-se do híbrido com 50 "dentes" e um estator de 4 fases [23]. Recorrendo-se às equações 2 e 1, respetivamente, descobre-se que o ângulo de passo típico é de 1,8º (= 360º/(50 * 4)) e o número de passos por rotação completa é de 200 (= 360º/1,8º).

2.3.1 - Tipo quanto à estrutura

Ao nível da sua estrutura os motores passo-a-passo podem ser classificados em três tipos, que operam de modo diferente. A figura 2.10 apresenta o interior de um motor passo-a-passo para facilitar a compreensão dos conceitos descritos de seguida. Assim, os motores passo-a-passo dividem-se em:

- **Íman permanente** (PM – *Permanent Magnet*) – motor sem dentes que possui um rotor de íman permanente A rotação do motor dá-se com o alinhamento do rotor com a polaridade do estator, esta ultima obtida através da corrente que passa aos seus enrolamentos. São motores de baixo custo e baixo resolução [22, 27];
- **Relutância Variável** (VR – *Variable Relutance*) – motor com dentes e com rotor não-magnetizado. Ao ser aplicada corrente nos enrolamento, um dos polos faz o rotor rodar, no enanto pela presença dos dentes este efetua o movimento aos poucos,

de modo a ficar alinhado com estator excitado. Design simples e custo moderado [22, 27];

- **Híbrido (H)** – junção dos dois conceitos anteriores ao ter um rotor magnetizado e com dentes. O íman permanente colocado no centro tem dois polos, positivo e o negativo, deslocados de modo que apareçam alternadamente aos olhos do estator. Do mesmo modo que no caso anterior, os dentes são colocados de maneira a que apenas encontre um local de equilíbrio. Design complexo e custo alto, no entanto apresenta alta resolução [22, 27];

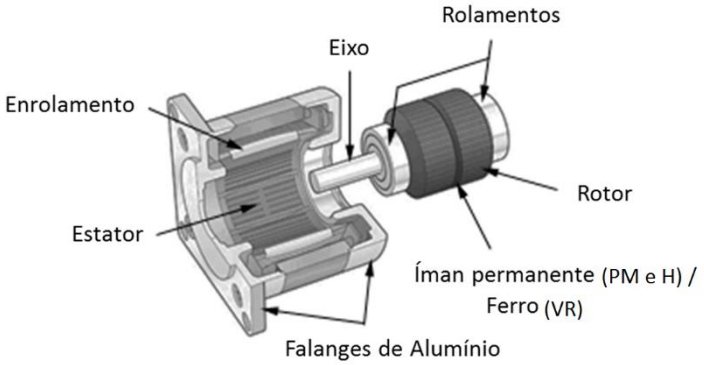


Figura 2.10 - Estrutura de um motor passo-a-passo [28]

Embora se trate de uma questão bastante relevante, não se achou pertinente, nesta fase do documento, fazer uma análise detalhada às características que distinguem estes três tipos de motores passo-a-passo. No entanto, no anexo A1 encontra-se um estudo aprofundado sobre esta matéria, realizado no contexto da dissertação.

2.3.2 - Tipo quanto aos enrolamentos e circuito de condução

Os motores passo-a-passo são também classificados quanto à configuração dos enrolamentos que percorrerem o estator, podendo ser unipolares ou bipolares, o que altera completamente os mecanismos de condução a usar.

2.3.2.1 - Unipolar

Tipicamente com 5 ou 6 fios, um motor unipolar tem dois dos fios colocados no centro de cada um dos enrolamento, podendo estes fios estarem ligados entre si. Ou seja, na verdade, cada fase tem dois enrolamentos, que são responsáveis pelo sentido positivo e negativo do movimento. Ou ainda, para o caso de um motor de 8 fios, é possível haver 2 fios por enrolamento.

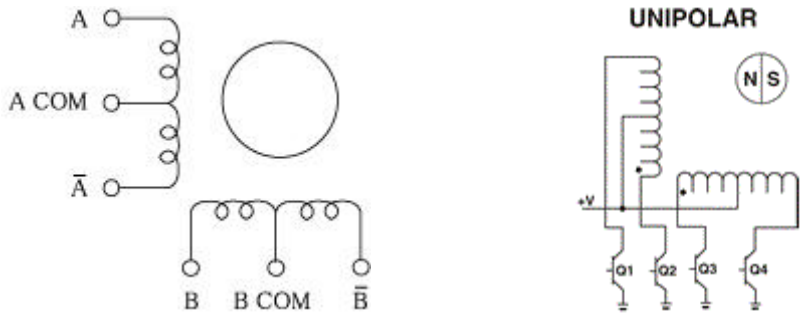


Figura 2.11 – Enrolamentos (esquerda) e circuito de condução (direita) do motor unipolar

Os fios presentes no meio dos enrolamentos, representados na figura 2.11 como A COM e B COM, normalmente encontram-se ligados à alimentação positiva e por sua vez as extremidades de cada enrolamento são ligadas à terra alternadamente para mudar a direção do campo magnético criado por esse enrolamento. Assim, como não é necessário inverter o sentido da corrente, o circuito de comutação pode ser feito de forma bastante simples, usando um transístor por enrolamento. Contudo, como o fio se encontra colocado no meio, apenas é utilizado metade de cada enrolamento levando a uma diminuição da capacidade de torque [22, 23, 29].

2.3.2.2 - Bipolar

Os motores bipolares têm, por norma, 4 fios que correspondem a dois pares/duas fases, cada um com um enrolamento. Nesta configuração não existe uma ligação direta com a alimentação como no caso dos motores unipolares. O que quer dizer que para se inverter a polaridade eletromagnética é necessário inverter a corrente nos enrolamentos, o que resulta num circuito de condução mais complexo.

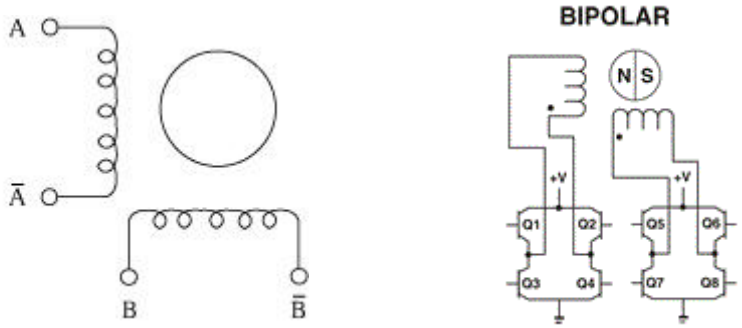


Figura 2.12 - Enrolamentos (esquerda) e circuito de condução (direita) do motor bipolar

A figura 2.12, do lado direito, mostra como é feita a condução dos motores bipolares, através de Pontes H. Este circuito permite ajustar a polaridade da energia aplicada nos terminais do motor através da comutação dos 4 interruptores – caso geral são transístores a desempenhar esse papel - presentes no esquema. Com isto é possível a corrente tomar tanto

valores positivos como valores negativos, levando a que seja, deste modo, praticável a alteração do sentido de rotação do motor. Em comparação com os motores unipolares, esta configuração consegue apresentar níveis de torque bastante superiores, mas como a complexidade do sistema de condução aumenta consideravelmente, os custos também crescem [22, 23, 29].

Conforme referenciado no subcapítulo 2.2, o tipo de enrolamentos de um motor influência de modo direto a escolha do *driver*. Então, há uma dependência que necessita de ser avaliada em conjunto para uma escolha adequada do componente certo. Assim, por exemplo, um motor com 4 fios, apenas deve ser conduzido com um *driver* bipolar, o que implica à partida uma diminuição do número de escolhas. Ou seja, existe, então, um compromisso entre os dois componentes que é essencial ter em atenção para o correto funcionamento e otimização do sistema. Juntamente com uma estratégia de condução com *chopper driver*, o uso de motores bipolares, e consequentemente *drivers* bipolares, é a que garante melhor eficiência para altas frequências de impulsos de passos, sendo por isso a configuração mais usado por entusiastas CNC [23].

2.3.3 - Classificação NEMA

Independentemente da estrutura e configuração dos enrolamentos dos motores passo-a-passo, estes são normalmente classificados quanto ao tamanho pela classe NEMA. NEMA é um acrónimo que significa *Nation Electrical Manufactures Association,* uma associação norte-americana que com esta classificação tenta padronizar os tamanhos, posição dos buracos para montagem, entre outros aspetos construtivos, dos motores passo-a-passo para que seja simples identificar a melhor opção para a aplicação desejada. Deste modo, um motor classificado como NEMA 23 tem praticamente todos os seus aspetos de construção definidos à partida, incluindo o diâmetro que será de 2.3".

Muitas vez esta classificação é mal interpretada levando a crer que quando maior a classificação NEMA, maior a capacidade de torque. Apesar de esta afirmação não corresponder à realidade, a verdade é que quanto maior o motor, maior a capacidade para tirar maior partido das suas dimensões para obter um torque superior.

2.3.4 - Modos de operação

Para lá da sua classificação quanto a aspetos físicos, os motores passo-a-passo, dependendo das suas configurações, podem operar essencialmente de 4 modos diferentes que estão relacionados com o modo como os solenoides são excitados:

- *Full step (one-phase on);*
- *Full step (two-phase on);*
- *Half step;*
- *Microstep*.

Estes modos de funcionamento, do mesmo modo que o tipo de motores quanto ao enrolamento, estão diretamente ligados com escolha do *driver*. Por outras palavras, e

exemplificando, apenas é possível colocar um motor a funcionar em modo *microstepping* se o *driver* tiver essa capacidade. Todavia, neste particular, em princípio, não existem limitações por parte do motor, apenas do lado do *driver* e, possivelmente, do controlador se não for capaz de enviar impulsos de passo com a frequência necessária para atuar no modo pretendido, à velocidade desejada.

O modo de operação escolhido para os motores afeta principalmente a sua resolução (ou precisão), velocidade e torque, sendo necessário um compromisso entre os três para se obter o melhor modo de funcionamento para o sistema desenhado. Quanto ao primeiro, vai alterar definitivamente o número de passos necessários para o motor completar uma rotação completa. Tendo em conta os modos de operação de um motor, a equação 2.1 dever ser atualizada para que possa refletir a diferenças em relação às diversas formas de operar.

$$Nº\ de\ passos = \frac{Ângulo\ de\ rotação}{Ângulo\ de\ passo} * Nº\ de\ divisões\ entre\ passos \tag{2.3}$$

Os modos *full step* não fazem nenhum divisão entre passo, o que significa que esse valor corresponderá a 1. Por sua vez, o modo *half step* divide cada passo em 2, ou seja, existem duas divisãoes por passo. Por fim, o modo *microstep* pode dividir os passos em vários intervalos, começando em 4, e continuando em 8, 16, 32, etc.

Assim sendo, se, por exemplo, o pretendido for executar uma volta completa de 360º, com um motor que tenha um ângulo de passo de 1.8º e esteja a trabalhar em modo *half step, resolvendo a equação* 2.3 o resultado é 400 passos, ou seja, é necessário o dobro dos passos para se atingir uma volta. Se o motor tivesse a operar em *microsteping* com 16 divisões, então o resultado seriam 3200 passos.

Uma vez mais, foi realizado um estudo detalhado sobre o modo de funcionamento dos motores passo-a-passo bastante relevantes para o uso correto deste componente no projeto, mas que decidiu-se deixar de fora do corpo principal do documento, estado presente no anexo A2.

2.3.5 - Utilidade e alternativas

Os motores passo-a-passo são usados nas mais variadas áreas, desde computadores a automóveis, devido a sua fantástica versatilidade de aplicação.

As suas principais características são:

- **Inexistência de escovas** – como não usam escovas para o seu funcionamento, a manutenção é menor e o número de falhas também;
- **Independência da carga** – giram com uma determinada velocidade independentemente da carga aplicada, desde que o torque seja menor que o do motor;
- **Posicionamento em malha aberta** – movem-se por passos, que são medidas discretas e determinadas. Se funcionar corretamente é sempre possível conhecer a posição do motor;

- **Capacidade de segurar carga parada** – desde que o torque seja respeitado, o motor é capaz de manter a carga na mesma posição sem gastos de energia;
- **Alteração de estado rápida** – tem uma excelente resposta ao arranque, parada e inversão de sentido.

Assim, estes são recomendados para o uso em situações que requeiram grande precisão no posicionamento, com erro pequeno e não cumulativo, e onde sejam necessárias grandes acelerações e desacelerações com baixo torque. No entanto, deve ser evitado o seu uso quando se pretendem elevadas velocidades e/ou torques muito elevados, pois em ambos os casos essas grandezas podem superar a atração entre o rotor e a estator e então sair fora de controlo, saltando passos [22, 23, 26, 27].

Outro tipo de motores bastante utilizados para este tipo de aplicações são os servomotores. Estes dispositivos têm tipicamente acoplado um sensor de posição para feedback, logo, atuam sobre uma lógica de malha fechada, ou seja, recebem um sinal de controlo indicando a posição desejada, verificam a sua posição atual e movem-se de acordo com a posição pretendida [30, 31]. No entanto, tipicamente, isso não altera o controlo a jusante já que esse mecanismo é feito internamente no dispositivo. A tabela 2.2 mostra um comparativo entre esta solução e os motores passo-a-passo.

Tabela 2.2 - Comparativo entre motores passo-a-passo e servomotores [31, 32]

	Motor passo-a-passo	**Servomotor**
Velocidade	Baixa	Alta
Torque (velocidade baixa/alta)	Alto/Médio	Baixo/Alto
Precisão	Alta	Muito Alta
Durabilidade	Boa	Média
Controlo	*Open-loop*	*Closed-loop* (com *encoder*) interno. Externamente semelhante
Manutenção	Baixa	Alta
Custo	Baixo	Baixo (mas maior)
Eficiência energética	Baixa	Média

Pela análise das características dos motores passo-a-passo, percebe-se o porquê de esta solução ser bastante utilizada pelos entusiastas CNC. Além de ser uma alternativa barata, consegue um posicionamento preciso sem mecanismo de *feedback* o que simplifica o seu controlo. Mais, a baixa manutenção e a alta fiabilidade são também pontos positivos a ter em conta. Como, e principalmente no caso da impressão 3D, não se pretendem velocidades extraordinariamente altas para não afetar a qualidade de impressão, e o torque necessário está apenas relacionado com o peso da própria estrutura da máquina, os motores passo-a-passo são uma boa opção.

2.4 - Eixos e Ferramenta

Por último, tal como indicado na figura 2.1, no início do capítulo 2, são apresentados os eixos mecânicos, responsáveis pela trajetória física da impressora 3D, e a ferramenta de trabalho. Os primeiros definem as dimensões da máquina e o volume de impressão, através dos três eixos cartesianos, X, Y e Z. Estes três eixos são tipicamente definidos da seguinte forma: X para movimentos laterais, Y para movimentos de frente/trás e Z para movimentos cima/baixo [21].

Por sua vez, o papel da ferramenta de trabalho é manipular o material que constituirá o objeto e aplicar ações de extrusão (ou corte), segundo os movimentos dos eixos.

2.4.1 – Estrutura mecânica

A estrutura física da máquina inclui a plataforma de suporte e a mesa de trabalho, além dos óbvios eixos cartesianos que definem as movimentações possíveis da máquina. O estudo da plataforma de suporte e da mesa de trabalho fogem um pouco ao âmbito do projeto, contudo, é importante não deixar de fazer referência à importância do desenho do sistema.

Na verdade, o modo como a plataforma é projetada, ou seja, a disposição física das peças que constituem a impressora 3D, ou ainda o arranjo mecânico, altera consideravelmente as características da mesma, o que as diferentes soluções na parte superior da cadeia de componentes, nomeadamente na eletrónica e motores a usar [33]. Algumas das configurações existentes são:

- **XYZ** – A ferramenta move-se nas três dimensões e a mesa de trabalho fica parada;
- **XZ-Y** – A ferramenta move-se no plano XZ e a mesa sobre o eixo Y;
- **Z-XY** – A ferramenta move-se apenas sobre o eixo Z e a mesa no eixo X e Y;
- **XY-Z** - A ferramenta move-se no plano XY e a mesa no eixo Z.

2.4.1.1 - Eixos

Para qualquer uma destas configurações, o movimento da máquina é feito através de movimentos lineares ao longo do volume da máquina (obtido pela conjugação dos 3 eixos). A fonte de movimento é um motor passo-a-passo, que gera movimento rotacional, logo, é necessário um mecanismo que permita transformar a rotação do eixo do motor em movimento linear. Para isso existem, entre outros, dois métodos: através de uma correia ou através de um parafuso sem fim [21].

As correias têm as vantagens de oferecer capacidade para velocidades altas, movimentação precisa e baixo custo. No entanto, não é capaz de suportar grandes cargas e com o tempo a ressonância aumenta e o oscilar também, o que pode levar a correia a soltar-se [34].

Os parafusos sem-fim são umas das soluções mais usadas para a movimentação linear de máquinas CNC. Como a razão de conversão é de 1:1, ou seja o eixo do motor faz rodar diretamente o parafuso, tem um bom aproveitamento do movimento do motor, consegue movimentar cargas mais expressivas e tem boa precisão. Todavia, em relação à velocidade fica aquém do método anterior e o desgaste pode resultar no aumento de ressonância, levando-o a vibrar [34].

2.4.2 - Ferramenta de trabalho

Por último, a ferramenta de trabalho apresenta-se como responsável por moldar/depositar o material de modo a converter a peça de um simples modelo computacional num objeto real e palpável.

Como seria de esperar, a ferramenta de trabalho representa uma grande diferenciação entre uma máquina CNC típica, de corte, e uma impressora 3D. De acordo com o que foi analisado no subcapítulo 2.2, nomeadamente na secção "Controlo da extrusora", a ferramenta de trabalho e o seu controlo são radicalmente diferentes nos dois casos. Aliás, esta ferramenta é a ponta do icebergue, ou seja o que é efetivamente visível, que distingue a manufatura aditiva da manufatura subtrativa já elencadas.

2.4.2.1 - Extrusora

No caso da impressão 3D a ferramenta relevante é a extrusora. Este elemento deve ser capaz de se autoalimentar com filamentos de plástico e de o derreter para, posteriormente, ser depositado na mesa de trabalho. Normalmente é divido em duas partes chamadas de *Cold End* e *Hot End* (figura 2.13) [21].

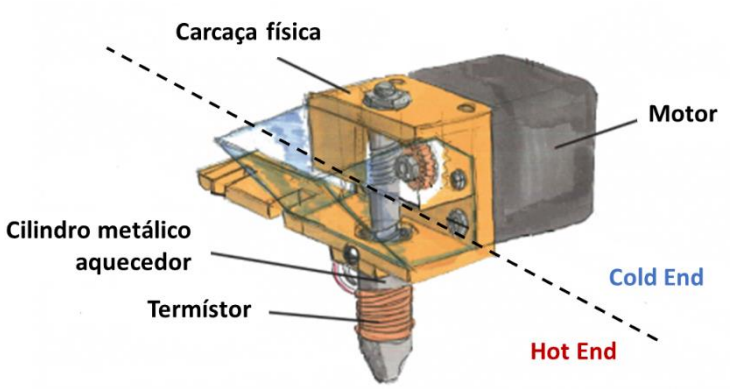


Figura 2.13 - Esquema de uma extrusora típica para impressão 3D

A parte fria da extrusora corresponde ao mecanismo que alimenta a parte quente com plástico à medida que este derrete. Esta tarefa é realizada com um ou mais motores, dependendo do modelo de extrusora que se utilize. Os mais populares na comunidade RepRap são *Wade's Geared Extruder*, *Greg's Hinged Extruder* e *Greg's Wade's Reloaded Extruder [21].*

Por outro lado, a *Hot End* deve ser capaz de derreter o plástico e depositá-lo sobre a mesa de trabalho. Para isso este componente conta com um pequeno cilindro de metal que aquece para derreter o plástico e um termístor, que lê a temperatura para que o controlo da extrusora a mantenha constante. O controlo consegue fazer variar a temperatura variando a tensão aplicada no aquecedor [21].

2.5 - Projeto RepRap

Nesta secção é feita a análise ao projeto RepRap, que contribui para a dissertação com a eletrónica necessária para a condução dos motores e controlo da extrusora. Este estudo foi dividido em três parte, começando pela apresentação do projeto, onde é feita a descrição do projeto RepRap. De seguida é analisada a eletrónica do projeto por se relevar essencial para a dissertação. Por fim, apesar do grande enfase dado a este projeto, também são enumerados, no final da secção, outros projetos de impressoras 3D que de algum modo influenciaram a indústria da impressão a três dimensões.

2.5.1 - Apresentação do projeto

O projeto RepRap, diminutivo de *Replicating Rapid-prototyper*, consiste no desenvolvimento de uma impressora 3D *open-source* capaz de se auto-replicar, através da impressão de objetos em plástico. Foi criada por Adrian Bowyer, com o primeiro aparecimento em 2004 e batizada com o nome Darwin (figura 2.14). Tornou-se na primeira impressora 3D de baixo custo e a sua aparição deu início a uma revolução no mundo da prototipagem rápida [35]. Com o lançamento dos modelos Medel, Huxlei e Prusa, criados por outros membros da comunidade, a impressora RepRap rapidamente se tornou na mais usada entre a comunidade de entusiastas segundo um inquérito de 2012 presente no artigo [8].

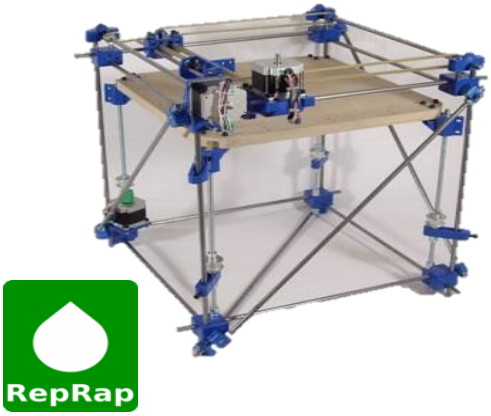


Figura 2.14 - Primeira impressora 3D RepRap, modelo Darwiw [36]

Como o próprio nome indica, o grande objetivo da impressora RepRap é ser capaz de produzir a maioria dos componentes que a constituem para criar outra impressora RepRap, e assim, deste modo, aumentar exponencialmente a disponibilidade e acessibilidade desta tecnologia a todos. Aliás, o projeto RepRap, segundo um estudo realizado [37], é apontado como um investimento bastante atrativo devido ao seu objetivo e modo como alterou o paradigma da impressão a três dimensões. Mais, segundo Joshua Pearce, professor na Universidade Tecnológica do Michigan, este projeto, se bem aplicado, pode ajudar a poupar milhares de euros [11, 38].

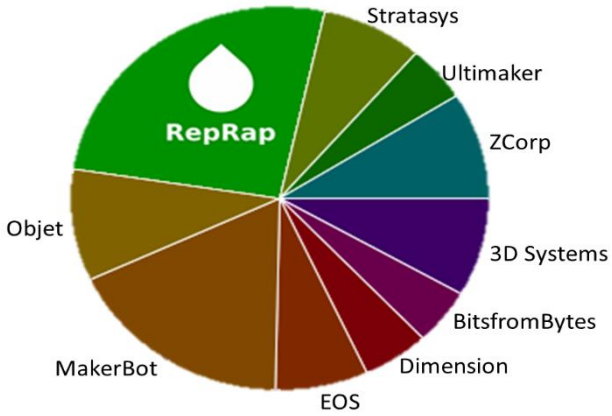


Figura 2.15 - Resultados do inquérito para aferir qual a impressora 3D mais usada [8]

De acordo com o apurado, conta com uma grande comunidade que ajuda no estudo, construção e integração da impressora, assim como disponibiliza todos o *software* e *firmware* necessários gratuitamente, incluindo o modelo das peças da impressora para a autorreplicação Com isto, o número de modelos cresceu desmedidamente, assim como levou ao aparecimento de projetos paralelos, como mostra a árvore cronológica [39] apresentada na página *web* do projeto.

Na verdade, o projeto RepRap é mais do que a eletrónica de condução e controlo, é uma solução completa para impressão a três dimensões. Segundo a sua página oficial, as impressoras RepRap são constituídas essencialmente por 4 componentes principais [21, 40]:

- **Cadeia de *software*** - conjunto de ferramentas responsáveis pela criação, conversão para código e interpretação do objeto a imprimir. Dividido em 3 partes: *software* CAD, *software* CAM, *firmware* da eletrónica;
- **Eletrónica** – conjunto de dispositivos que conduzem, controlam e fazem a máquina mover-se. Dividido em 5 partes: controlador principal, controlador da extrusora, *drivers* de motor, fins-de-curso e cama de impressão quente;
- **Corpo mecânico** – plataforma que sustenta a máquina e a permite mover. Constituído pelos eixos e plataforma de suporte;
- **Extrusora** – responsável pela alimentação e por derreter o plástico a ser depositado na mesa. Contém 3 partes principais: o motor que puxa o filamento de plástico, o termístor para ler a temperatura e a ponta que aquece o plástico.

Ora, tendo em atenção a figura 2.1, no início do capítulo 2, que mostra o esquemático dos componentes que constituem uma máquina CNC, ou uma impressora 3D, chega-se à conclusão, como seria de esperar, que partilham sensivelmente a mesma arquitetura. Assim, no contexto da dissertação, o objetivo é usar a eletrónica que o projeto RepRap dispõe para a condução dos motores e o controlo da extrusora.

2.5.2 - Electrónica RepRap

Com a evolução do projeto RepRap desenvolveram-se novas versões dos componentes de eletrónica, em que cada novo modelo foi trazendo uma nova geração com as melhorias necessárias para o projeto progredir. Uma das gerações mais marcantes para o projeto foi a Geração 3, juntamente com o modelo Medel. Apesar de já antiga – data de 2008 –, é das gerações mais documentadas, muito por culpa da estratégia da empresa MakerBot em lançar uma impressora 3D, a Cupcake, que usava eletrónica muito similar à usada no modelo Medel da RepRap [41]. Tendo isto em consideração e também face à disponibilidade do material existente para o projeto, a análise realizada parte deste princípio, ou seja, o estudo assume esta geração como alvo principal. Na verdade, analisar todas as outras gerações seria algo bastante complexo e pouco relevante para o projeto, por isso apenas se focou na geração com relevo para o projeto de dissertação.

Tanto esta como outras gerações da eletrónica são constituídas por um conjunto de placas eletrónicas que interligadas entre si formam os componentes principais da impressora RepRap, como como se observa na figura 2.16.

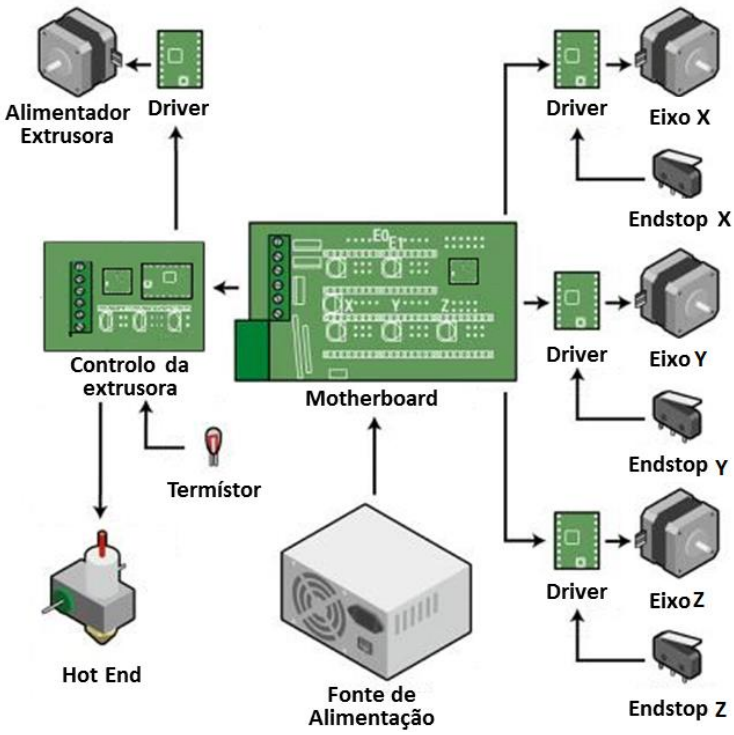


Figura 2.16 - Esquema de funcionamento da eletrónica RepRap [42]

2.5.2.1 - *Motherboard* RepRap

Trata-se do coração do sistema RepRap, o centro de controlo. Aqui é interpretado o G-code e enviados comandos para os respetivos componentes afetados.

Na sua versão 1.2 este dispositivo é comandado por um chip Atmega644p - um Sanguino, uma derivação do Arduíno, sendo ambos compatíveis -, possui 3 conectores IDC para ligar a 3

drivers de motor; uma conexão RS485 para comunicar com a placa de controlo da extrusora; possui 4 conectores I2C para ligar periféricos, como por exemplo um ecrã LCD; um conector serie para programação da *motherboard* e comunicação com o *host software*; um *slot* para cartões de memória para colocar G-code; uma ligação para uma fonte de alimentação ATX e botão *on/off* para controlo simples do todo o sistema e para paragens de emergência [43].

Apesar de desempenhar um papel crítico numa impressora RepRap convencional, no contexto deste projeto este é um elemento obsoleto. Como o controlo da extrusora é efetuado noutra placa separada, o papel do LinuxCNC é suprimir a necessidade da utilização da *motherboard* já que o controlo dos movimentos é totalmente realizado por ele.

2.5.2.2 - *Drivers* dos motores

Cada uma desta *boards* permitem controlar um motor passo-a-passo responsável por fazer mover um dos 3 eixos. Também são capazes de receber dados de dois fins-de-curso (mecânicos ou óticos*),* através de ligação RJ45, para limitar o caminho passível de ser percorrido em cada eixo. Na sua versão 2.3 o componente principal trata-se do *chip* Allegro A3982. Possuem também um potenciómetro que permite controlar a corrente que passa para os enrolamentos do motor; uma interface de comunicação IDC com a *motherboard* de onde recebe os sinais de direção e passo; uma saída com 4 pinos para um motor-a-passo e um conector para alimentação [44]. O *chip* Allegro A3982 é um *driver* de motor passo-a-passo completo, com tradutor de sinais de passo e direção para sinais elétricos. Foi desenhado para funcionar com motores bipolares em *full* e *half step*, sendo capaz de fornecer ao motor até 2A por enrolamento e funcionar a 35V [45].

2.5.2.3 - Fins-de-curso óticos

Tendo em conta não existirem mecanismos de *feedback*, a posição de motor, a cada nova utilização é desconhecida. Para que exista uma referência da posição do motor é necessário ajustar a posição manualmente ou então utilizar fins-de-curso, que são dispositivos que acionam quando é atingido um limite do eixo. Assim, no início de cada impressão, o motor precisa de recuar até atingir este limite que quando ativo representa que o eixo está na posição de origem. Podem ser usados limites mecânicos, que são acionados quando o carro toca nesse dispositivo, ou óticos, que dependem de um sensor que determina o fim do eixo.

Para as impressoras RepRap são tipicamente utilizados um fins-de-curso ótico por eixo, contundo, pode também ser usados 6, ou seja 2 por eixo, para garantir que o limite de cada eixo nunca é ultrapassado, mesmo que na presença de erros no código na definição das dimensões do eixo.

A versão 2.1 usa o fins-de-curso ótico H21LOB, aliado a um LED para *debug* / sinalização de limite ativo e um conector RJ45 para ligar ao *driver* do motor responsável pelo eixo correspondente [46].

2.5.2.4 - Controlador da extrusora

O componente responsável pelo controlo da extrusora junta numa só placa um Atmega168 (um clone Arduíno) para programar o comportamento do motor e do aquecedor do plástico; um sensor de temperatura para usar com um termístor padrão de 100K; dois *drivers* de motor DC

para controlar dois motores DC ou um motor passo-a-passo, capazes de fornecer 2A aos enrolamentos dos motores; três *drivers* MOSFET (14A/12V) para controlar os aquecedores do plástico, aquecedores da mesa de impressão, ventoinhas, etc; conexão RS485 para comunicar com a *motherboard* e para alimentação; conector ICSP para programação mais simples; conector IDC para ligar *encode* magnético rotativo; um conector I2C para ligar periféricos; uma ligação série para programação ou ligação direta com o PC e ainda conectores servos e outros extras para aumentar ainda mais as possibilidades de customização da máquina final [47].

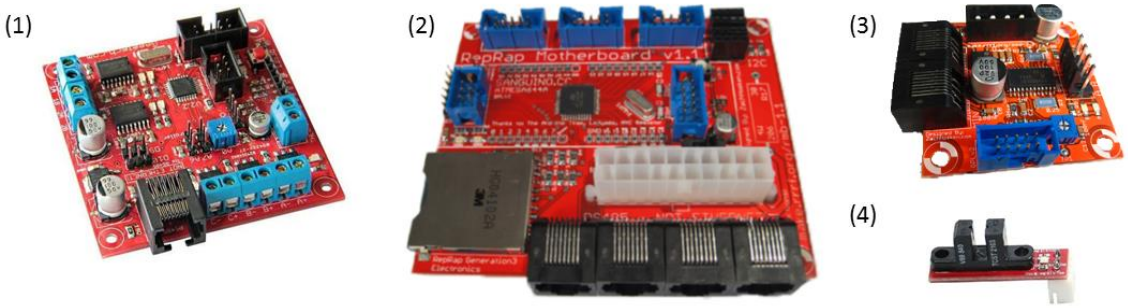


Figura 2.17 - Geração 3 RepRap com (1) Controlador da extrusora v2.2, (2) *Motherboard v1.2*, (3) *Driver* de motor v2.3 e (4) Fins-de-curso óticos v2.1

Face às gerações anteriores, a Geração 3 conta com *drivers* do motor mais pequenos, poderosos e baratos; o processador principal sofreu uma melhoria passando para um atmega644p (Sanguino) vindo de um Atmega168 (Arduíno); passou a existir um controlador próprio para a extrusora, um Atmega168 que controla os componentes sem ser necessário sobrecarregar o processador principal; os *drivers* de motores passaram de dois L297/L298 para um chip A3982 da Allegro, que é mais barato, pequeno e aquece menos que os antecessores e também apresenta pela primeira vez uma interface RS485. Em comparação com as gerações posteriores, toda esta tecnologia é ultrapassada, contudo a sua utilização é ainda viável devido à grande utilização no passado que levou à criação de uma comunidade forte com muita documentação. Uma análise mais profunda sobre as gerações do RepRap pode ser feita através da análise da tabela comparativa entre os diferentes conjuntos de componentes eletrónicos apresentados [48].

Além deste 4 componentes em cima detalhados, existe um conjunto de periféricos que podem ser adicionados à impressora de modo a aumentar as suas capacidades, entre os quais se destacam os ecrãs LCD para monotorização da impressão e os *encoders* magnéticos rotativos para controlar diretamente a velocidade do motor da extrusora.

Por fim, é importante ressalvar que a documentação existente na página oficial do projeto RepRap [36] é quase toda ela proveniente de um esforço da comunidade e que, apesar de ser uma página de edição colaborativa, se encontra relativamente bem organizada. Mais, algumas das fontes de informação mais precisas e fiáveis, como o *website* do criador e do fabricante das placas dos componentes da impressora, à data já não se encontram disponíveis, tornando-se, em muitos casos, o site oficial do projeto RepRap a única fonte de dados disponível.

2.5.3 - Impressoras 3D alternativas

O projeto RepRap despoletou uma pequena revolução na indústria da impressão 3D, o que levou naturalmente, e tratando-se este de um projeto *open-source*, a que rapidamente se desenvolvessem alternativas ao projeto original de criar uma impressora 3D *low-cost*. Já sem o objetivo da autorreplicação, os sistemas subsequentes mantinham a meta de democratizar a utilização da impressora 3D por todos tornando-a simples, acessível e barata, a somar à óbvia utilidade de um sistema de impressão a três dimensões, mesmo para uso doméstico.

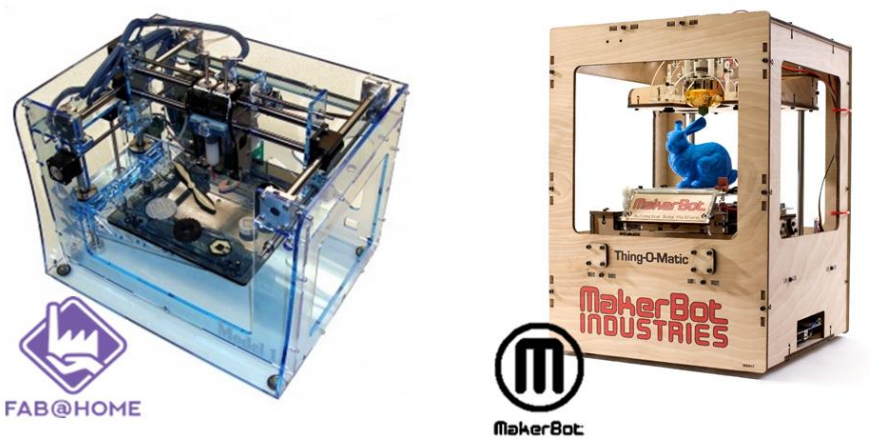


Figura 2.18 - Modelo 1 da Fab@Home (esquerda) e Cupcake da MakerBot (direita) [49, 50]

2.5.3.1 - Fab@Home

Um dos primeiro projetos a aparecer foi o Fab@Home com o seu primeiro modelo, chamado simplesmente de Model 1. Igualmente *open-source*, tinha como principal objetivo levar a fabricação aditiva para a casa de todos, tornando a impressora 3D num bem desejável. Com origem em 2006, criado por Hod Lipson e Evan Malone, apoiava-se essencialmente na colaboração de uma grande comunidade, tendo inclusive recebido o prémio *Popular Mechanics Breakthrough Award [50].*

2.5.3.2 - MakerBot

Por sua vez, a empresa MakerBot tornou-se uma das mais bem sucedidas, comprovado pelos resultados do inquérito presentes na figura 2.15, ao ser o segundo projeto mais usado, segundo os resultados desse inquérito. Também detém a página *web* Thingiverse, uma dos mais usados para partilha de objetos para impressão. O seu primeiro modelo chamava-se Cupcake, lançada em 2009, e usava eletrónica muito similar à Geração 3 do RepRap o que impulsionou o sucesso do mesmo. Os seus mais recentes modelos, fazem parte da linha Replicator e usam o *software* ReplicatorG para o controlo da impressora [49].

De modo a melhor compreender o que destinguem os primeiros modelos da Fab@Home e da Makerbot, foi feita uma comparação apresentada na tabela 2.3.

Tabela 2.3 - Comparativo do modelo origem da Fab@Home e MakerBot [49, 50]

	Fab@Home	MakerBot
Modelo	Model 1	Cupcake
Tecnologia de impressão	Deposição por seringa	Extrusão de termoplástico
Tamanho de impressão	18,6x16x18 cm	10x10x13 cm
Tipo material	Vários, devido à tecnologia de impressão	Plásticos tipo ABS, PLA, etc.
Custo Material	Muito baixo	Baixo
Velocidade	Fraca	Fraca
Plataforma física	Fraca	Aceitável
Características	(+) Baixo preço (+) Baixo custo do material (+) Variedade de material possível de usar (-) Lento (-) Pode ser lento a endurecer o material (-) Requer montagem (-) Fraca qualidade de impressão	(+) Relativo baixo preço (+) Baixo custo do material (+) Altamente customizável (+) Usa eletrónica RepRap (Geração 3) (+) Grande comunidade (-) Lento (-) Requer montagem (-) Qualidade de impressão aceitável

Na verdade, é praticamente impossível referenciar todas as impressoras 3D existentes devido à enorme quantidade e variedade. Na sua grande maioria foram surgindo projetos derivados dois primeiros e, por essa razão, optou-se por apenas analisar os dois projetos que de certo modo mais contribuíram para o sucesso desta tecnologia.

Para uma análise mais profunda, nomeadamente versando modelos de outras empresa, vale a pena consultar a literatura onde são realizados estudos de mercado mais extensos [40, 42].

2.6 - EMCRepRap

Por fim, para finalizar o capítulo "Estado da arte", procurou-se descrever o projeto EMCRepRap cujo objetivo é bastante semelhante ao objetivo desta dissertação.

Segundo a análise feita até este ponto, é possível concluir que não existe uma forma direta de usar o LinuxCNC para o controlo de uma impressora 3D, nomeadamente para o controlo da extrusora. Porém, embora muito tímidos, já foram efetuados no passado alguns avanços no sentido de criar uma ponte entre estes dois mundos aparentemente separados.

Dada a relevância, tanto do *software* LinuxCNC, como do projeto RepRap, naturalmente surgiram entusiastas que tentaram colmatar esta lacuna usando este dois sistemas. Assim, alguns membros da comunidade LinuxCNC – que procuravam um modo de usar um *software* de controlo que lhes era familiar para controlar uma impressora 3D – e outros membros da comunidade RepRap – que procuravam encontrar um controlador mais capaz que a *motherboard* RepRap – criaram um projeto apelidado de EMCRepRap. O nome do projeto advém

da junção de EMC (*Enhanced Machine Controller)*, o antigo nome do LinuxCNC, com RepRap, do projeto homónimo.

Este projeto consistia na utilização do LinuxCNC para controlar uma impressora 3D, tirando partido do projeto RepRap, nomeadamente a sua eletrónica, para condução dos motores e controlo da extrusora [51].

A ideia por detrás deste projeto é a mesma defendida ao longo deste documento, ou seja, o LinuxCNC deve ser capaz de controlar os movimentos da impressora 3D do mesmo modo como faz para uma máquina CNC tradicional, usando para isso a linguagem típica de programação de máquinas, o G-code.

Contudo, no caso de uma impressora 3D é também fundamental criar uma forma de controlar a extrusora ao mesmo tempo que são emitidas ordens de movimentação da máquina. Para isso, são adicionados ao G-code, funções especiais sob a forma de M-codes, que permitem alargar a funcionalidades de uma máquina CNC nas mais variadas vertentes. Neste caso em particular, é através deles que são desencadeadas ordens de extrusão para a placa de controlo da extrusora [51].

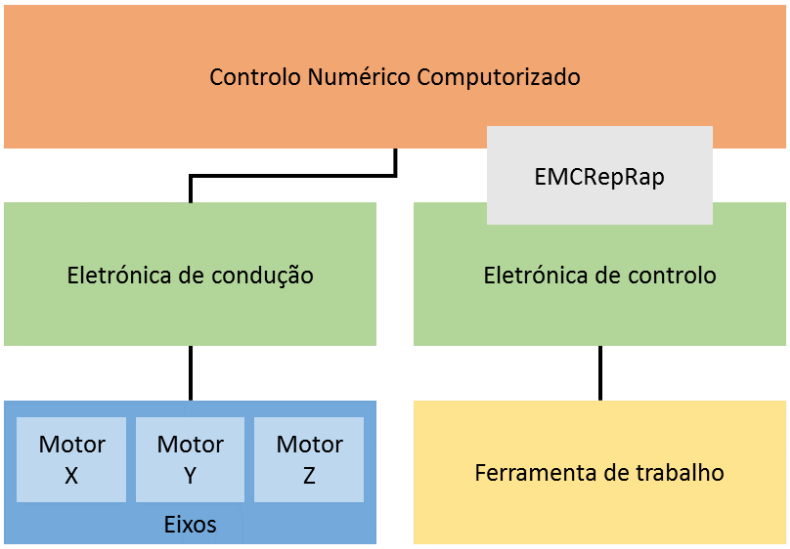


Figura 2.19 - Esquema da figura 2.1 com o enquadramento do EMCRepRap

A juntar a isto, o projeto também incentiva a criação e utilização de um conjunto de *scripts* e ligações que permitem estabelecer a conexão entre o LinuxCNC e a placa controladora da extrusão. A meta é substituir por completo a necessidade de utilizar a *motherboard* do projeto RepRap [51].

No entanto, a documentação deste sub-projeto é quase nula, havendo menos de uma dezena de utilizadores no total que participaram nele. Mais, a partilha de informação é escassa e já antiga, datando-se de 2010 a última contribuição na página *web* [51] dedicado a quem procura este tipo de soluções.

Capítulo 3

LinuxCNC

No presente capítulo será efetuada uma análise ao LinuxCNC, desde a sua origem, modo de funcionamento e modo como está organizado internamente. Perceber como funciona e como está organizado trata-se de uma etapa fundamental para o projeto, não só por ser de um dos objetivos da dissertação, como também pelo facto de ser o *software* de controlo utilizado com a impressora 3D desenvolvida.

Inicialmente é feita uma pena introdução focando-se nas arquiteturas de controlo "abertas", depois este capítulo é dividido em 2 partes. Na primeira é feito descrito o modo de funcionamento do LinuxCNC, nomeadamente o modo como os seus módulos e processos internos interagem. No fim é feita uma breve menção ao sistema de ficheiros que o LinuxCNC utiliza para a configuração da máquina.

Desde a criação da máquina CNC a evolução tem sido enorme e cada vez mais a manufatura tem um papel fundamental na sociedade e na economia global. Contudo, segundo Yoram Koren [52] e Atul Gupta e Venu Uppuluri Srinivasa [53], houve, de algum modo, uma estagnação na evolução dos sistemas de manufatura. De acordo com as duas publicações, ao longo do tempo assistiu-se a uma utilização cada vez mais intensiva de protocolos independentes, que, naturalmente, contribuiu para a maior complexidade de integração dos equipamentos de diferentes fabricantes e diferentes épocas.

A necessidade de uma arquitetura "aberta" para máquinas CNC era imperativa para a sua evolução. Assim, começara a ser dados os primeiros passos para a criação de uma arquitetura que permitisse a fácil reconfiguração dos sistemas de manufatura, as quais se apelidaram de OAC's, *Open Architecture Controllers* [53].

Segundo o IEEE, um OAC define-se como "um sistema 'aberto' capaz de permitir a execução de aplicações devidamente implementadas, independentemente do vendedor da plataforma, que interagem com outras aplicações do sistema e de forma consistente com o utilizador" [52, 54].

De acordo com Oshiro [55] citado em [53], um OAC deve de ter:

- **Interações** – para comunicar com os outros sistemas;
- **Interoperabilidade** – por forma a garantir a funcionalidade de um componente independentemente do vendedor;
- **Portabilidade** – facilidade de transferir uma aplicação de um ambiente para outro;
- **Escalabilidade** – capacidade de expansão das funcionalidades do sistema.

Logo, um OAC deve ser flexível tanto a nível de *hardware* como a nível de *software*, ou seja, deve possibilitar a integração de outros sistemas de controlo de uma forma estandardizada, assim como permitir o uso de aplicações e algoritmos independentes. Estas e outras vantagens de um OAC, nomeadamente para projetos como a dissertação descrita neste documento, são destacadas por Fisher [56].

A primeira solução OAC, tinha o nome de EMC, *Enhanced Machine Controller*, proposta pelo *US National Institute of Standards and Technology*, e conseguia utilizar um simples PC com Linux para efetuar controlo numérico, respeitando as diretrizes do programa *Next Generation Controller* [16, 57]. Posteriormente, foi desenvolvida uma nova versão desta solução, o EMC2, até que em 2011 o projeto mudou o nome para LinuxCNC [17].

O LinuxCNC apresenta-se como um programa *open-source* capaz de transformar um computador convencional num controlador de máquinas CNC. As suas características e funções gerais estão descritas no subcapítulo 2.1.2.1, aquando do estudo do "Estado da arte" desta ferramenta. Por essa razão, parte-se do princípio que chegado a este ponto já foi feita uma leitura a essa secção, assim não será repetida essa mesma informação neste capítulo. Por sua vez, feita uma análise mais detalhada ao modo de funcionamento do LinuxCNC e às suas bases definidoras.

De referir que apesar de já se encontrar na versão 2.6.2, a versão analisada é a versão 2.5.4, pois foi a versão utilizada para a realização do projeto. No entanto não são esperadas grandes alterações face ao que é aqui exposto.

3.1 - Princípios de funcionamento

De modo a tirar o máximo partido da ferramenta e de entender como a usar corretamente é essencial perceber a arquitetura do sistema, os principais módulos que o constituem e quais as suas interações entre si, o que pode ser analisado na figura 3.1.

Os cinco grandes componentes constituintes do LinuxCNC [58] são:

- Uma camada que permite a reconfiguração do hardware no LinuxCNC sem necessidade de recompilar (HAL);
- Controlo de movimentos (EMCMOT);
- Controlo de entradas e saídas discretas (EMCIO);
- Executor e coordenador de tarefas (EMCTASK);
- Interfaces gráficas.

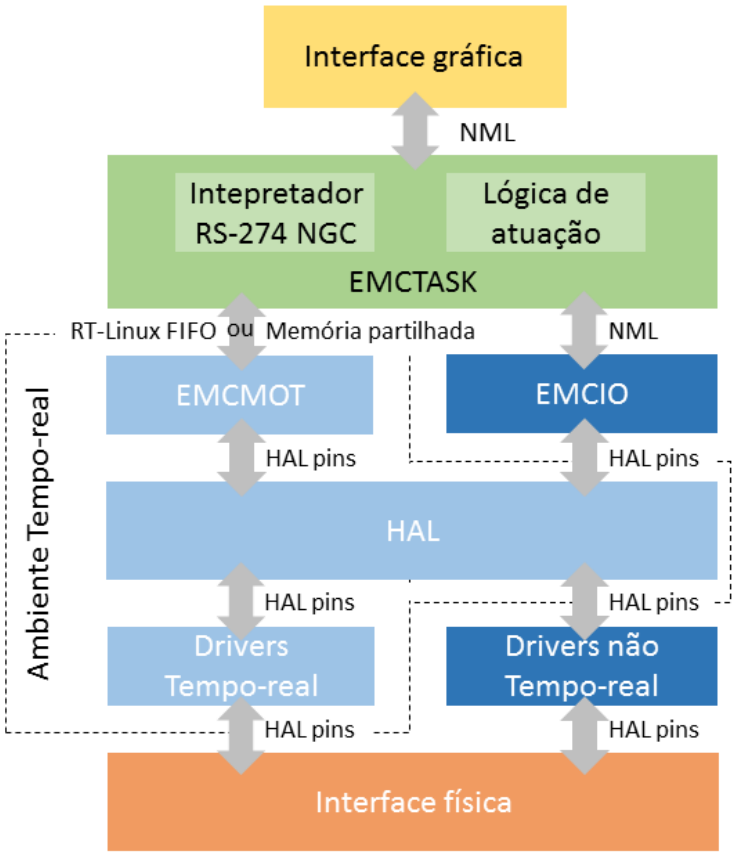


Figura 3.1 - Arquitetura de funcionamento LinuxCNC

Tal como se pode verificar, existe uma hierarquia de camadas e funções que o LinuxCNC respeita para o controlo de uma máquina. Na verdade, este *software* segue uma orientação típica do controlo de máquinas CNC, obedecendo, através da sua organização interna, aos mesmos princípios elencados por Suh S.H., et. Al. no livro *Theory and Design of CNC Systems [1]*

Do mesmo modo, e como este se trata de um *software* criado sempre tendo em atenção às definições de um OAC, a interligação entre camadas dá-se, não só através dos pinos HAL e melhoria partilhada, como por NML – *Neutral Message Language*. Esta tecnologia trata-se de uma API uniforme cujo objetivo é maximizar a portabilidade dos *software*, escondendo o protocolos específicos de cada linguagem. O artigo de Jonh Michaloski, et Al., [59] apresenta um estudo mais pormenorizado sobre esta linguagem.

3.1.1 - *Hardware Abstraction Layer* (HAL)

Ao mais alto nível, o HAL trata de permitir a definição, especificação e interligação de um conjunto de *building blocks* para a configuração de um sistema complexo. Trata-se de uma camada - simula uma espécie de *breadboard* - colocada entre os módulos EMCMOT/EMCIO e os drivers do *hardware*, em ambiente tempo-real. Como o próprio nome indica, a ideia deste módulo é tentar padronizar as ligações do *hardware* com os módulos de controlo, de maneira

a que a configuração seja semelhante para qualquer *hardware* que lhe seja ligado, tornando mais fácil atingir, então, um dos principais requisitos dos OAC's, a interoperabilidade.

O utilizador não necessita de saber como cada bloco funcional opera, estes são tratados como caixas negras com um determinado número de pinos de entrada e saída. O *designer*, durante a fase de conceção do sistema, apenas tem de decidir que componentes utilizar e como os interligar. As ligações são efetuadas criando sinais HAL que, seguindo a analogia da *breadboard*, são como fios que ligam os componentes, e devem ser do mesmo tipo que o pinos aos quais ligam. O anteriormente descrito é explicado no manual LinuxCNC, num documento específico para detalhar o funcionamento do HAL [60].

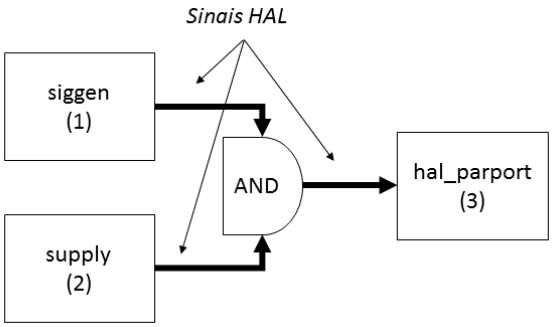


Figura 3.2 - Exemplo básico do funcionamento do HAL

Exemplificando (figura 3.2), se o utilizador quiser usar um AND lógico com um gerador de sinais (1) e um determinado valor constante (2), para mostrar a junção dos sinais na porta paralela (3), não necessita de saber como é que cada um destes módulos funciona, apenas precisa de os invocar e ligar corretamente (HAL *signals*) para obter o resultado desejado.

Conforme a aplicação, o utilizador apenas tem de especificar algumas características do seu sistema e o LinuxCNC trata de adaptar os módulos escolhidos por ele para essa mesma configuração.

Além disto, o HAL permite a criação de *drivers* de *hardware* e módulos de *software* que podem ser implementados e executados em tempo-real. Nesse caso, a criação de módulos é suportada por duas ferramentas, o *Comp* e o *Classiclandder*, que permitem, respetivamente, facilitar a criação de módulos mais complexos escritos em C e o desenvolvimento de módulos de lógica *ladder*. Também é possível criar módulos em Python, mas em ambiente não-tempo-real, ao contrário dos exemplos imediatamente supracitados [60].

3.1.2 - Gestão dos movimentos (EMCMOT)

Ainda dentro do ambiente tempo-real, o EMCMOT é responsável pelo controlo dos movimentos. Como seria de esperar, no contexto do projeto, estará neste módulo a origem dos movimentos nos 3 eixos cartesianos da impressora 3D.

Segundo um estudo feito ao LinuxCNC [57], este componente executa periodicamente tarefas de planeamento de trajetórias, cálculos da cinemática direta e inversa e por fim gera a saída pretendida para os motores responsáveis pelo movimento. Este último ponto é resultado

do estudo da posição do eixo, da análise da próxima trajetória e da interpolação entre a posição atual e a próxima posição na trajetória.

Mais, os parâmetros únicos de cada máquina, indispensáveis para efetuar os cálculos, como por exemplo, o número e tipo de eixos, fatores de escala, máxima velocidade e aceleração, são obtidos através do sistema de ficheiros criado durante a configuração do sistema. Neste grupo entra também os limites de comprimento impostos no *software*, os limites de comprimento efetivos do *hardware* e os detetores da posição de origem [57].

Recordando a figura 3.1, para receber comandos ou enviar o estado da máquina, o EMCMOT comunica com o EMCTASK através memória partilhada ou mecanismo FIFO do RTLinux. Não é usado NML, pois este requer C++, mas o controlo dos movimentos está limitado ao C para fornecer estruturas genéricas e maximizar a portabilidade para outros sistemas operativo de tempo-real, conforme se confirma por colaboradores do projeto LinuxCNC [61]. Mais uma vez são respeitados os princípios dos OAC's, principalmente a portabilidade.

Sob a forma de um módulo HAL, é possível criar suporte específico para um determinado tipo de *hardware,* evitando assim alterar o código nuclear do sistema. Assim, através dos pinos do HAL, o EMCMOT interage com módulos tempo-real, como são exemplos os *drivers* de *hardware* e compensadores PID [57].

3.1.3 - Módulos de entradas e saídas (EMCIO)

Trata-se de um controlador de entradas e saídas (I/O), constituído por uma hierarquia de controladores subordinados para controlar a ferramentas principal da máquina, assim como ferramentas auxiliares integradas e definidas pelo utilizador - como o LinuxCNC não está nativamente preparado para controlar uma impressora 3D, apesar de extrusora ser a sua ferramenta principal, é definida como um novo subsistema. Este novo subsistema deve ser definido separadamente do ficheiro HAL, mas deve ser invocado por ela através do comando *loadusr*. Este conceito torna-se mais claro no capítulo 5, mais precisamente na secção 5.2.3.1, onde este módulo é efetivamente usado.

Como a arquitetura do controlador I/O é específica para cada máquina, dependendo da complexidade do sistema, podem existir mais do que um ficheiro de configuração, contendo as interfaces específicas de cada subsistema. Estes subsistemas vêm, na sua grande maioria, já pré-configurados bastando para isso os invocar se necessário. Conforme visto na secção sobre o HAL, podem ser criados subsistemas alternativos utilizando o *ClassicLadder* para criar um módulo PLC – *Programmable Logic Controller* – ou através do *Comp* que permite construir, compilar e instalar módulos HAL através do código fonte. Obviamente, obedecendo sempre às mesmas regras, as ligações contidas nos ficheiros são feitas através de sinais HAL do mesmo tipo que o módulo a serem usados.

São feitas referências a estes ficheiros no ficheiro principal de configuração do HAL, a ser lido no início do programa [57, 61] .

3.1.4 - Interpretador lógico (EMCTASK)

A função do módulo EMCTASK é interpretar a linguagem de programação típica das máquinas CNC, o RS-274NGC, uma versão ligeiramente modificada da RS-274D - mas conhecida como G-code - pelo projeto *Next Generation Controller* (NGC), sem alterar os seus princípios

básicos. Além disso, tem também um processo interno responsável por manipular comandos e pela lógica de atuação.

Este elemento está diretamente ligado à interface gráfica do utilizador, onde recebe comandos diretamente do operador e envia dados de monotorização. Esta interação é realizada por mensagens NML.

Por outro lado, também monitoriza e coordena as sub-rotinas dos módulos EMCMOT e EMCIO de acordo com os comandos que lhe são pedidos [57].

3.1.5 - Interfaces gráficas

Por fim, na camada mais alta dos componentes que constituem o LinuxCNC, está a interface gráfica do utilizador ou GUI – *Graphical User Interface* -, onde o operador pode, de modo simples e intuitivo, monitorizar o estado da máquina e enviar comandos para a camada EMCTASK. Em alternativa, também pode ser usado o terminal para executar o LinuxCNC.

De origem, o LinuxCNC apresenta várias interfaces de utilizador alternativas, entre as quais se destacam:

- AXIS;
- Touchy;
- NGCGUI;
- Mini;
- TkLinuxCNC;
- Keystick.

O AXIS é a interface padrão do LinuxCNC e, ao mesmo tempo, é também a opção mais avançada e completa. As suas principais vantagens são ter uma janela de pré-visualização do G-code com grafismo a três dimensões e permitir expandir a interface com outros módulos para adaptar às necessidades, adicionando módulos criados pelo utilizador ou *widgets* já existentes no programa [62].

A interface com o utilizador está tipicamente no mesmo ambiente que o resto dos elementos do controlador, contudo o LinuxCNC permite ser controlado remotamente, através de uma das suas interfaces gráficas compatíveis por meio de mensagens NML, podendo mesmo, em alguns casos, ser usado outro sistema operativo que não o Linux [57].

A camada HAL possui também interfaces gráficas simples dedicadas à monitorização dos seus processos, o HALScope e o HALMeter. Estas servem para fazer os ajustes e calibrações necessárias ao conjunto de ligações feitas na camada HAL. Por sua vez, o *ClassicLadder*, usado para editar os diagramas *ladder*, também tem, como seria de esperar, a sua própria interface gráfica [57, 60].

3.1.6 - Ambiente tempo-real

De modo a obter um controlo em tempo-real dos movimentos das máquinas, é necessário uma plataforma com capacidade computacional para realizar tarefas em tempo-real. Para isso, o LinuxCNC requer o uso de extensões tempo-real sobre o *kernel* Linux padrão [58].

O tempo é um fator crucial. O controlador da máquina precisa de interpretar os códigos G-code, calcular o movimento necessário para cada motor de cada eixo, enviar os comandos,

seguir a execução das tarefas, tudo isto respeitando limites temporais muito restritos. Por outras palavras, espera-se que um comando do *software* de controlo tenha uma reposta pronta a cálculos algo complexos. Ainda assim, usar um computador para controlar uma máquina CNC (ou impressora 3D) é um desperdício de recursos, tendo em conta os requisitos mínimos referidos na página da internet do controlador [17]. Na verdade, qualquer PC moderno tem a capacidade de controlar um sem-número de eixos de uma máquina. Contudo, um PC convencional não planeia a execução de tarefas de modo a potenciar os seus recursos para este tipo de tarefas. Logo, o objetivo é concentrar os recursos do PC nas tarefas ligadas ao controlo da máquina e dar menos prioridade às tarefas acessórias para estes sistemas.

Assim, para adereçar esta questão, o LinuxCNC faz uso de extensões de tempo-real RTAI, Xenomai ou RTLinux para modificar o modo como o *kernel* padrão do Linux opera, ou melhor, modificar o algoritmo responsável pela distribuição de recursos computacionais pelas várias tarefas. Contudo, em vez de usar um sistema apenas com função tempo-real (sistemas usualmente caros), ao usar uma extensão tempo-real permite que coexistam os dois tipos de ambientes (figura 3.3) [63, 64].

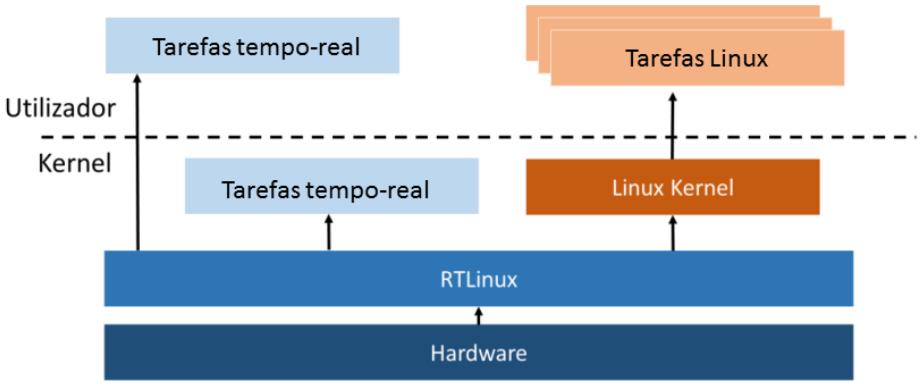


Figura 3.3 - Esquema exemplificativo do RTLinux

Como afirma o artigo de Victor Yodaiken, em 1999 [64], a classificação de prioridade das tarefas num sistema operativo tempo-real é relativamente diferente do Linux convencional. No caso do último mencionado a latência depende de tudo o que está a ser executado pelo sistema, causando impressibilidade no cumprimento das tarefas. Por sua vez, num sistema de tempo-real, a latência apenas depende das tarefas com prioridade igual ou superior à que está a ser executada no momento. Então, o RTLinux atinge uma performance de tempo-real ao eliminar as fontes de imprevisibilidade causada pela tentativa de se obter uma boa distribuição entre todos os processos. Pode-se considerar, de forma simplificada, que o *kernel* RTLinux encontra-se situado algures entre o *kernel* Linux normal e o hardware. Deste modo, a performance tempo-real é obtida fazendo com que apenas os processos relacionados com o RTLinux sejam processados devido à alta prioridade, permitindo uma precisão na ordem das dezenas de microssegundos, sendo que é obviamente limitado pelos recursos de *hardware*.

3.1.7 - Drivers internos e Interface física

A ligação do LinuxCNC com o mundo real é realizada através de *drivers*, tempo-real e não-tempo-real, que são implementados na camada HAL. A nível interno, este contém um conjunto de *drivers* que permite cobrir uma grande quantidade de *hardware* de diferentes empresas para fazer a interface física, que vão desde dispositivos Mesa Electronics, Vital Systems, até Pico Systems, entre outros. No entanto, de todos, o mais simples e mais usual trata-se do uso da porta paralela presente nos computadores mais antigos [60].

A porta paralela tem duas configurações possíveis para o driver do HAL: como *output* para se obter 12 saídas e 5 entradas, ou como *input*, para se dispor de 13 entradas e 4 saídas [60, 65].

3.2 - Sistema de ficheiros

A configuração de uma máquina no LinuxCNC é realizada através de um conjunto de ficheiros de texto de fácil compreensão ao olho humano (figura 3.4). Estes ficheiros podem ser lidos e alterados com um editor de texto comum para modificar os valores neles contidos de modo a adaptar a configuração à máquina que se pretende integrar.

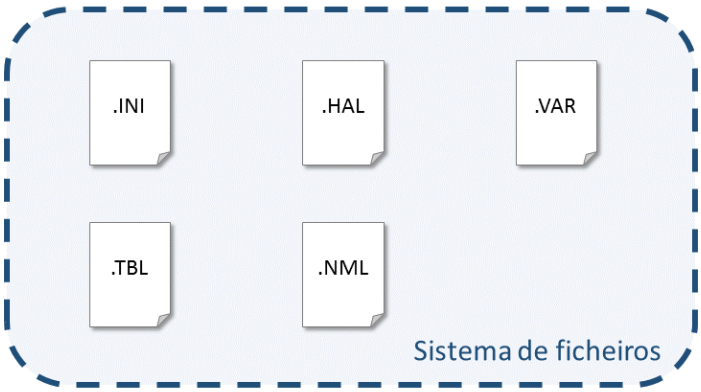


Figura 3.4 - Ficheiros de configuração do LinuxCNC

No início do programa cada um destes ficheiros é lido para dar conhecimento ao LinuxCNC das propriedades do sistema, sendo que, alguns destes documentos são também modificados durante o seu funcionamento [66]. Os principais ficheiros de configuração são:

- **INI** - trata-se do principal ficheiro do programa, onde estão presentes as propriedades específicas de cada máquina;
- **HAL** - ficheiro que contém todas as configurações necessárias para fazer a ponte entre os comandos e a interface física, como explicado na secção 3.1.1. Normalmente existem mais de que um ficheiro para dividir a máquina em módulos;
- **VAR** - esta é a forma que o *software* tem de guardar valores entre sessões de utilização do LinuxCNC. Usado principalmente para configurar offsets a cada nova execução;

- **TBL** – guarda a informação sobre cada ferramenta (apenas para ferramentas de corte);
- **NML** – usado para configurar os canais de comunicação internamente e externamente, como analisado em 3.1.

Apesar da fácil interpretação, é importante salientar que criar cada um destes ficheiros de raiz seria uma tarefa bastante complexa que afastaria a maioria dos utilizadores. Então, de modo a simplificar bastante a configuração correta de uma máquina o LinuxCNC dispõem de duas ferramentas que geram os ficheiros de configuração conforme os dados inseridos sobre a máquina, um com o nome *Stepconf*, para o uso de motores passo-a-passo, e *Pncconf*, para se usar com produtos Mesa I/O, tanto para servomotores como para motores passo-a-passo. Mais, aquando da criação dos ficheiros de configuração, é também criado um ficheiro com extensão .stepconf que pode ser utilizado para modificar os parâmetros da máquina, através dos programas supracitados, sem a necessidade de alterar nada diretamente nos ficheiro de configuração [62]. Contudo, naturalmente que configurações mais complexas que que destoam do modelo normal de uma máquina CNC – tal como no caso da dissertação em que se pretende criar um módulo que permita o LinuxCNC controlar uma impressora 3D -, não podem ser totalmente definidas por estes *wizards* já que contêm opções limitadas.

Além do já referido, o LinuxCNC trás consigo vários sistemas pré-configurados com algumas das configurações mais comuns utilizados com este *software*. Assim, se necessário, basta fazer pequenas alterações às propriedades base, através dos ficheiros de configuração, para adaptar à máquina que se pretende integrar.

Dentro do conjunto de ficheiros necessários para a configuração do LinuxCNC, destaca-se o ficheiro INI.

3.2.1 - Ficheiro de configurações INI

O Ficheiro INI é ficheiro mais relevante de todos, pois é nele que estão presentes as principais parâmetros e definições da máquina, bem como as ligações entre os outros ficheiros de configuração e quando os usar.

Mais, através do ficheiro INI é possível ampliar as capacidades do LinuxCNC para lá da configuração base necessária para o controlo de uma máquina CNC simples. Este ficheiro encontra-se dividido em secções, onde estão definidas as variáveis do sistema.

Para isso, é necessário conhecer a própria estrutura do ficheiro e a função de cada secção [66]. As secções são:

- **EMC** – onde estão definidas algumas informações gerais, como o nome da máquina e a versão do documento.
- **DISPLAY** – nesta secção estão definidas as variáveis que modificam a interface gráfica do utilizador. Não só se pode escolher aqui qual a interface pretendida, como também alterar alguns parâmetros da mesma. Também é nesta secção que se invoca os painéis de controlo adicionais criados pelo utilizador.
- **FILTER** – o LinuxCNC permite que sejam adicionados filtros para manipular, automaticamente, os ficheiros que são carregados para o programa. O resultado dos filtros, definidos nesta secção, serão tratados como G-code para o *software* o conseguir interpretar.

- **EMCMOT** – nesta fase do ficheiro é onde está definido o período base em que LinuxCNC verifica se é necessário gerar um sinal de passo. Também estão definidas outras variáveis temporais usadas pelo EMCMOT.
- **TASK** – aqui especificado o nome e definido tempo de ciclo que o módulo TASK executa. Atualmente estas duas variáveis não são, usualmente modificadas.
- **HAL** – esta segmento é responsável por invocar o ficheiro HAL que contém as ligações internas entre pinos HAL, próprias de cada máquina. Pode também ser definida uma variável que invoca um outro ficheiro HAL contendo ligações de pinos associados a painéis de controlo extra criados pelo utilizador. A invocação das interfaces gráficas do HAL, para que estas executem ao mesmo tempo que a inteface principal, também é definida pelas variáveis desta secção.
- **HALUI** – podem ser executados comandos manuais de G-code usando uma interface gráfica do HAL, o HALUI.
- **TRAJ** – trata-se da secção mais complexa. Nesta secção estão definidos os parâmetros principais do planeamento da trajetória. Para lá da cinemática também é aqui definido o número de eixos, o seu nome, a velocidade e aceleração máximas de cada eixo.
- **AXIS_n – a** cada um dos eixos será destinada uma secção destas. Neste espaço são atribuídos os parâmetros individuais de cada eixo, nomeadamente, o tipo (linear ou angular), a velocidade e aceleração máximas, o tamanho do eixo e a posição de origem (*home*). Os parâmetros aqui definidos estão condicionados ao valor das varriavés definidas na secção [TRAJ].
- **EMXIO** – por fim, este segmento está relacionado com a camada EMCIO e nesta fase são definidos o ciclo em que o módulo executa, qual o ficheiro TBL a usar pelo LinuxCNC, bem como outras configurações relacionadas com a ferramenta de trabalho.

Capítulo 4

Dimensionamento da impressora 3D

O objetivo deste capítulo é descrever o processo de dimensionamento e análise dos componentes que constituem a impressora 3D com LinuxCNC, tendo em conta os requisitos do sistema e o material disponível para o projeto. Este estudo encontra-se decomposto em 5 partes que visam o arranjo mecânico, a movimentação linear, os circuitos de condução, a extrusora e a análise da viabilidade do controlador. Note-se, no entanto, que nesta fase não se pretende demonstrar as soluções, nem resultados obtidos, mas sim discutir e comentar os componentes usados para o posterior desenvolvimento e configuração da impressora 3D, descritos no capítulo 5.

Assim, no primeiro ponto deste capítulo é examinada a orientação estrutural da máquina. De seguida, parte-se para a análise dos eixos cartesianos, o dimensionamento dos motores. A somar a isto, nas secções seguintes, é feita a avaliação dos *drivers* e do componente de controlo da extrusora. Por último, verifica-se os requisitos do controlador e a sua capacidade para ser o controlador do sistema.

Conforme previamente indicado, no começo do projeto já existia bastante material disponível para a sua realização. Assim, optou-se pela análise e integração sempre que possível, dos componentes existentes no laboratório, nomeadamente o suporte físico, os eixos e motores e, de igual modo, a eletrónica do projeto RepRap.

4.1 - Arranjo mecânico

Entende-se por arranjo mecânico a disposição física dos eixos que definem o volume máximo de impressão. A estrutura física da máquina, mais propriamente o modo como é montada, altera substancialmente o dimensionamento dos outros componentes. Porém, a otimização da estrutura de impressão não é um objetivo do projeto e por essa razão foi escolhida a configuração XZ-Y mostrada na figura 4.1. Trata-se de uma das configurações mais usadas na montagem de impressoras 3D, inclusive os afamados modelos Mendel, Huxley e Prusa da impressora RepRap [33].

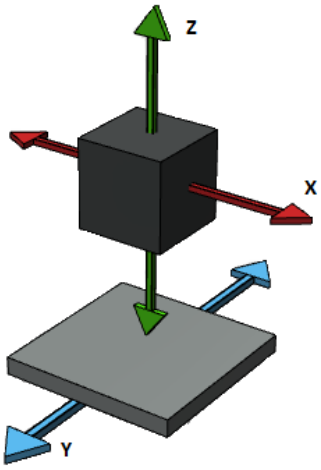


Figura 4.1 - Movimentos de impressora 3D com configuração XZ-Y.

Como é fácil de perceber, a denominação XZ-Y está relacionada com a disposição e sentido dos movimentos da extrusora (representado na figura 4.1 pelo bloco cinza escuro) - no plano XZ e arrumação e movimentos da mesa de trabalho no eixo Y.

Segundo a configuração escolhida, o eixo Y apenas terá de suportar o peso da mesa de trabalho. Por sua vez, o eixo Z terá de suportar o peso da extrusora, o seu próprio peso e o peso do seu motor. Por último o eixo Y tem de ser capaz de vencer o peso do eixo e motor Z, somado ao peso da extrusora.

4.2 - Movimentos lineares

As três dimensões exigidas pela máquina são alcançadas por meio de movimentos lineares no sentido dos três eixos cartesianos. No entanto, são utilizados motores que geram movimento rotativo, ou seja, é necessário converter a rotação produzida pelos motores em movimentos lineares. Para isso, os motores foram acoplados a mecanismos com parafuso sem-fim, que, ao rodar faz mover o carrinho presente no sistema. A figura 4.2 mostra o mecanismo usado no projeto para os movimentos lineares dos 3 eixos.

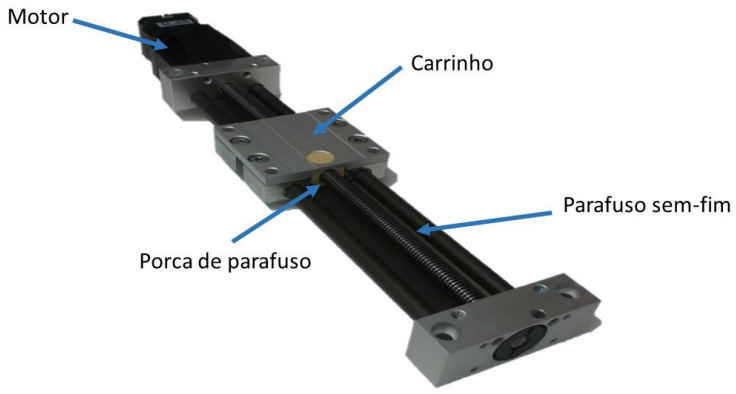


Figura 4.2 - Mecanismo usado no projeto responsável pelo movimento linear

Assim, é essencial perceber as grandezas envolvidas no estudo da movimentação linear e sua condução de modo a maximizar as capacidades da impressora e obter um comportamento correto e previsível. Qualquer que seja a configuração usada, o motor escolhido e o circuito de condução eleito, o estudo de um movimento linear começa sempre pela escolha do tipo de sistema linear e pela avalização da carga a que vai estar sujeito.

A conversão do movimento rotativo em movimento linear exige que o motor imprima um torque superior ao torque necessário para mover o eixo. Por sua vez, este último, para conseguir mover o carrinho com a carga, tem de superar a inércia da carga somada à inércia do parafuso sem-fim [67-69].

$$T_{total} = T_{fricção} + T_{aceleração} \tag{4.1}$$

Decompondo,

$$T_{fricção} = \frac{F + \mu_s * P}{2\pi * e} * p \tag{4.2}$$

$$T_{aceleração} = \left(J_{carga} + J_{parafuso} + J_{motor}\right) * \alpha \tag{4.3}$$

Por sua vez, as inércias de carga e de parafuso são obtidas através das expressões,

$$J_{parafuso} = \frac{\pi * l * \rho * r^4}{2} \tag{4.4}$$

$$J_{carga} = m * \left(\frac{p}{2\pi}\right)^2 \tag{4.5}$$

Por fim, a aceleração angular pode ser obtida através da seguinte equação 4.6 [68, 69], obtida através da conversão da velocidade angular, em rad/s^2, em rotações por minuto (rpm) e que, neste caso, tendo em conta o passo (ou *pitch*) do prego, se pode transformar em velocidade linear, em m/s.

$$\alpha = \frac{d\omega}{dt} \simeq \frac{\omega_f - \omega_0}{t_f - t_0} \simeq \frac{\omega_f}{t_f} \tag{4.6}$$

Sendo que

$$1rad/s = \frac{2\pi}{60}\, rpm \quad \text{e} \quad v = p * \frac{rpm}{60}\ (m/s) \tag{4.7}$$

então,

$$\alpha \simeq \frac{v}{t} * \frac{2\pi}{p} \tag{4.6}$$

onde:

T – torque (N.m);
F – força externa no sentido contrário ao do movimento aplicada à carga (N);
P – peso da carga (N);
p – passo do parafuso (m/rev);
ρ – densidade do parafuso (kg/m^3);
e – eficiência do parafuso;
μ_s - coeficiente de atrito estático;
g – aceleração gravítica (m/s^2);
J – inércia (kg/m^2);
l – tamanho da viagem (m);
r – raio do parafuso (m);
α – aceleração angular (rad/s^2);
ω – velocidade angular (rad/s);
t – tempo de aceleração (s);
v – velocidade linear (m/s).

Como se pode verificar pela equação 4.3, é necessário conhecer a inércia do rotor do motor para o cálculo do torque necessário para determinada aplicação. Note-se, portanto, que existe aqui um compromisso complexo entre o motor e o eixo, já que, para conhecer a inércia do rotor, é necessário à partida escolher um motor. Ora, só depois de escolhido o motor se pode verificar o torque final e averiguar se o mesmo tem capacidade para conduzir o eixo, ou seja, não é possível fazer uma análise em fases distintas. Caso o motor não seja capaz de conduzir o eixo, é necessário repetir o processo até se encontrar um motor que satisfaça a necessidade.

Para o projeto recorreu-se aos veios mecânicos com parafuso sem-fim do tipo ACME e carrinho móvel do fabricante Igus, modelo SLW-1080. Os eixos usados partilham praticamente as mesmas características físicas. A tabela 4.1 mostra alguns dos atributos dos eixos cartesianos.

Tabela 4.1 - Características físicas dos 3 eixos cartesianos e forças a que estão sujeitos [70-72]

	Eixo X	**Eixo Y**	**Eixo Z**
Tamanho (mm)	750	750	250
Diâmetro (mm)	10	10	10
Avanço/*pitch* (mm/rev)	2	2	2
Massa (kg)	2,2 (=0,7+0,2*7,5)	2,2 (=0,7+0,2*7,5)	1,2 (=0,7+0,2*2,5)
Constituição	Aço	Aço	Aço
Densidade (kg/m^3)	~7850	~7850	~7850
Porca de parafuso	Plástico / PTFE	Plástico / PTFE	Plástico / PTFE
Coeficiente de atrito estático	~0,04	~0,04	~0,04
Eficiência	~65%	~65%	~65%
Força externa contrária ao movimento - F (N)	-	-	34,5 [=(1,2+0,32+2)*9,8)]
Peso a que o eixo está sujeito – P (N)	34,5 [=(1,2+0,32+2)*9,8)]	9,8 (=1*9,8)	-

Na tabela 4.1 também estão presentes as variáveis de força externa contrária ao movimento (F) e o peso a que o eixo está sujeito (P). Estes valores estão relacionados com a disposição física do eixo. Recordando o que foi dito na secção 4.1, onde é apresentado o arranjo mecânico, o eixo X está sujeito ao peso do eixo e motor Z, mais o peso da extrusora, sendo que estas forças têm o seu vetor perpendicular ao sentido do movimento. No caso do eixo Y, este apenas está sujeito ao peso da mesa de trabalho, não havendo nenhuma força no sentido oposto ao do movimento, como no caso anterior. Por sua vez, o eixo Z, por esta numa posição vertical, no pior caso, ou seja, quando pretender subir, tem de suportar o seu próprio peso, o peso do seu motor e peso da extrusora, sendo que todas estas forças funcionam contra o sentido do movimento. De acordo com o material disponível para ao projeto da dissertação, a massa da mesa de trabalho estima-se ser menor que 1kg e o peso da extrusora será não superior a 2kg.

Quanto aos motores disponíveis para a realização da dissertação, a tabela 4.2, apresenta as características principais relevantes para esta etapa, incluindo a massa usada para o cálculo da força externa contrária ao movimento e peso a que os eixos estão sujeitos.

Tabela 4.2 – Características relevantes dos motores disponíveis para o projeto, para o cálculo do torque [73, 74]

	Eixo X	**Eixo Y**	**Eixo Z**
Fabricante	Nanotec	Nanotec	Igus
Modelo	ST4118D1804	ST4118D1804	MOT-AN-S-060-005-042-L-A-AAAA
Massa (kg)	0,5	0,5	0,32
Inércia rotor (kg.m^2)	$1{,}02 * 10^{-5}$	$1{,}02 * 10^{-5}$	$0{,}8 * 10^{-5}$
Torque (N.m)	0,8	0,8	0,5

No que toca à velocidade linear máxima e tempo de aceleração para atingir a velocidade máxima que o sistema deve ser capaz de demonstrar, não foram requeridos nenhum valor específico para essas grandezas, nesta fase do projeto. Assim, tentou-se encontrar valores não muito exigentes, mas que não comprometessem a finalidade da máquina, apenas para o cálculo do torque, de modo a descobrir se os motores são capazes de mover a impressora.

Antes de mais á necessário perceber que o controlador LinuxCNC gera impulsos que imprimem um perfil de velocidade trapezoidal, ou seja, durante o movimento do eixo este passa por um tempo de aceleração, mais tempo de velocidade constante (velocidade máxima) e por fim um tempo de desaceleração. Se for considerado, para simplificação dos cálculos, que estes três tempos têm o mesmo valor, então, a velocidade máxima será 1,5 vezes maior a velocidade média o eixo leva a mover-se de uma ponta a outra [75]. Assim, sabendo que, por exemplo, os eixos maiores que têm 750mm, considerando que seriam necessários 10 segundos para chegar de um lado ao outro do eixo, então a velocidade média seria de 0,075m/s, logo, a velocidade máxima atingida é de 0,1125 m/s (=1,5*0.075). Do mesmo modo, se atribui o valor de 3,33s (=10/3) ao tempo de aceleração até à velocidade máxima, por os três tempos serem iguais.

Usando a equação 4.1, incluindo as que dela derivam, os valores obtidos de torque para cada um dos eixos é apresentado na tabela 4.3.

Tabela 4.3 – Cálculo do torque necessário para mover os três eixos

	Eixo X	**Eixo Y**	**Eixo Z**
Aceleração angular (m/s^2)	106,14	106,14	106,14
Inércia da carga (kg/m^2**)**	$3{,}57 * 10^{-5}$	$1{,}01 * 10^{-5}$	$3{,}57 * 10^{-5}$
Inércia do parafuso kg/m^2**)**	$1{,}47 * 10^{-12}$	$1{,}47 * 10^{-12}$	$0{,}49 * 10^{-12}$
Torque de aceleração (N.m)	$1{,}12 * 10^{-3}$	$1{,}09 * 10^{-3}$	$0{,}89 * 10^{-3}$
Torque de fricção (N.m)	$0{,}68 * 10^{-3}$	$0{,}19 * 10^{-3}$	$1{,}69 * 10^{-2}$
Torque total (N.m)	$1{,}80 * 10^{-3}$	$1{,}29 * 10^{-3}$	$1{,}78 * 10^{-2}$
Margem de segurança	x1,4		
Torque total recomendado (N.m)	$2{,}51 * 10^{-3}$	$1{,}80 * 10^{-3}$	$2{,}49 * 10^{-2}$

Na realidade, deve-se manter uma margem de segurança de 40%, isto é, o torque do motor deve ser 1,4 vezes maior que o torque da aplicação [76]. Ou seja, tendo em atenção ao torque de cada motor, na tabela 4.2, nos três casos verifica-se claramente a condição

$$T_{motor} > T_{eixo} \tag{4.7}$$

o que significa que os motores disponíveis podem ser usado para integração com cada um dos três veios responsável pelos movimentos nos três sentidos. Ainda assim, não se pode esquecer que o próprio mecanismo de suporte e acoplamento provoca atrito que afeta estes cálculos. Assim, a margem de segurança deve ser, sempre que possível, ainda maior.

Para o fornecimento de energia ao motor é necessária a utilização de uma fonte de tensão continua. Esta alimenta os *drivers* que por sua vez alimentam o motor. Ou seja, é necessário conhecer o componente responsável pela condução dos motores, mais concretamente as suas limitações técnicas, para avaliar corretamente as curvas velocidade-torque do motor.

4.3 - Circuitos de condução

Como já abordado no capítulo 2.2, os circuitos de condução são responsáveis por receber os sinais de passo e direção do controlador e convertê-los numa sequência lógica de impulsos. Desta forma, os *drivers* fornecem corrente aos enrolamentos dos motores com o intuito de os fazer mover.

Inicialmente é feita a análise da viabilidade de utilização dos drivers disponíveis e o estudo da curva velocidade-torque tendo em conta a tensão e torque a que o motor está sujeito.

A forma como o fazem altera o modo de funcionamento dos motores como foi referido no capítulo 2.3. Ou seja, também é a eletrónica de condução que introduz os conceitos *full step, half step e microstep*.

4.3.1 - Análise face às características dos motores

No dimensionamento de um sistema de movimento linear elétrico, um aspeto igualmente importante é escolher o circuito de condução para cada um dos eixos. Neste caso, verificou-se se o material disponível era capaz de cumprir essa missão. No fundo, o que se pretende é que a corrente fornecida pelo *driver* aos enrolamentos do motor seja suficiente para este funcionar corretamente. Depois de verificada esta condição é necessário selecionar a fonte de alimentação do circuito, que, juntamente com o torque a que o motor é sujeito, definem a velocidade máxima de cada eixo.

Tabela 4.4 - Características do motor segundo os fabricantes [73, 74]

	Eixo X	**Eixo Y**	**Eixo Z**
Fabricante	Nanotec	Nanotec	Igus
Modelo	ST4118D1804	ST4118D1804	MOT-AN-S-060-005-042-L-A-AAAA
Ângulo de passo	1,8º	1,8º	1,8º
Tensão máxima (V)	-	-	60
Tensão nominal (V)	5,4	5,4	3,15
Corrente nominal (A)	1,8	1,8	1,8
Resistência/fase (Ω)	3,0 ± 15%	3,0 ± 15%	1,75 ± 10%
Impedância/fase (mH)	7,0 ± 20%	7,0 ± 20%	3,30 ± 20%

4.3.1.1 - Corrente nominal

Segundo a tabela 4.4, os três motores usados nesta máquina têm uma corrente nominal de 1,8 A. Assim, como a disposição do eixo não influencia a escolha do circuito de condução, podem ser usados três drivers iguais.

Encontrada a corrente nominal do motor, o próximo passo foi descobrir se a eletrónica de condução disponível para o projeto é capaz de imprimir no motor essa corrente nominal.

Conforme analisado no capítulo 2 deste documento e de acordo com o material disponível, a eletrónica RepRap apresenta 3 *boards* para conduzir os 3 motores dos 3 eixos. Estas placas contêm, entre outros componentes, um chip Allegro A3982. Este dispositivo trata-se de um *chopper driver* de motores passo-a-passo bipolares capazes de fornecer até ±2 A e 35V ao motor (Figura 4.3), através da comutação dos transístores que constituem as duas pontes H [45] no seu interior. Para além disso, a *board* RepRap conta com um potenciómetro para ajustar a corrente que sai do driver e assim, empiricamente, colocar a corrente no nível ideal.

Characteristic	Rating	Units
Load Supply Voltage	35	V
Logic Input Voltage	–0.3 to 7	V
Sense Voltage	0.5	V
Reference Voltage	4	V
Logic Supply Voltage Range	3.0 — 5.5	V
Output Current	±2	A

Figura 4.3 - Especificações do *driver* Allegro A3982 retiradas da sua *datasheet* [45]

Dado a corrente de saída ser 2A, superior à corrente necessária para conduzir os motores dos 3 eixos (1,8S), este trata-se de um *driver* ajustado a esta aplicação.

Note-se ainda o facto de este driver reconhecer um sinal positivo a partir dos 3V até aos 5,5V. Ou seja, a tensão lógica dos sinais de controlo deve respeitar este intervalo e, como se poderá ver na secção 4.5.1, isto verifica-se.

4.3.1.2 - Curvas torque-velocidade

Ainda na fase de dimensionamento dos circuitos de condução, por sim foi avaliada a velocidade máxima que o motor pode atingir tendo em conta o torque que lhe é aplicado e a fonte de tensão a que está sujeito.

Para este exercício de análise, as figuras 4.5 e 4.6 apresentam as curvas torque-velocidade dos motores do eixo X, Y (figura 4.5) e Z (figura 4.6), segundo os seus fabricantes.

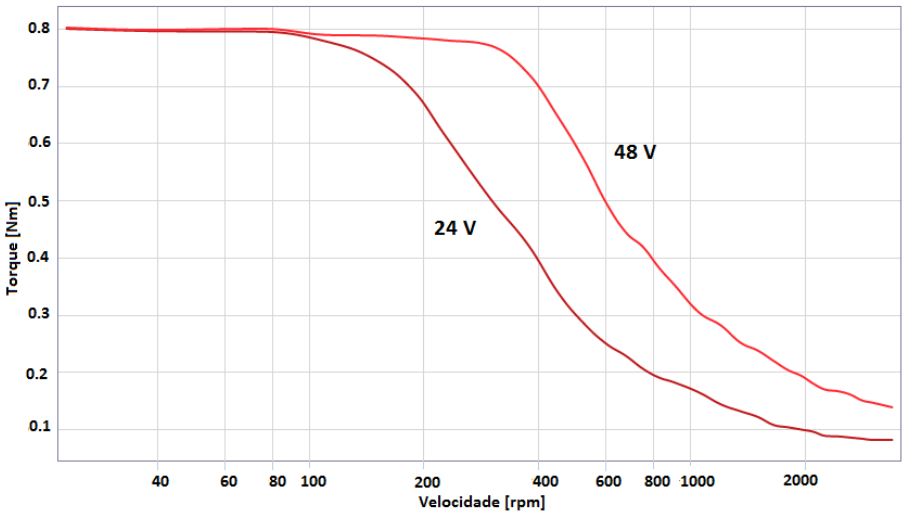


Figura 4.4 - Curvas Torque-Velocidade do motor ST4118D1804, Nanotec [77]

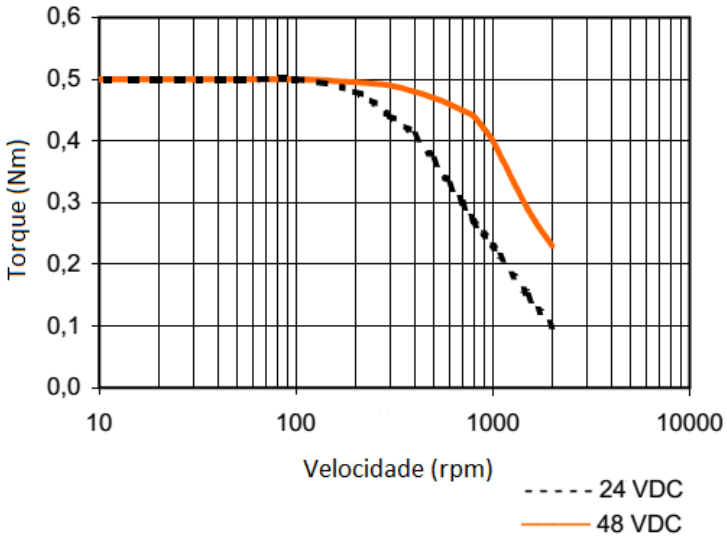


Figura 4.5 - Curvas Torque-Velocidade do motor MOT-AN-S-060-005-042-L-A-AAAA, Igus [74]

Pela observação dos gráficos das figuras 4.5 e 4.6, consegue-se perceber que quanto maior a tensão aplicada no motor, maior torque/velocidade o motor é capaz de suportar. Contudo, note-se que as tensões aqui apresentadas são de uma ordem de grandeza perto de 10x maior que a tensão nominal indicada na *datasheet* do motor. Segundo a literatura, nomeadamente Stoelting, H.D, et. Al., em 2008 [22], na prática este rácio deve estar compreendido entre 1:5 e 1:10. A razão pela qual isto acontece foi, em parte, aferido no capítulo 2.2.1. Relembrando, a tensão alta nos terminais do motor faz com que a corrente que passa no enrolamento atinga rapidamente o seu valor nominal. O valor da tensão nominal tipicamente presente nas *datasheets* é simplesmente calculado através da Lei de Ohm, com corrente nominal e a resistência de cada enrolamento.

Na verdade, embora a tensão máxima que o *driver* pode suportar seja de 35V, este não foi alimentado com esse valor de tensão, por duas razões. Primeiro, porque a fonte de tensão disponível no laboratório é uma fonte de alimentação ATX que fornece 12V. Segundo, porque o *driver* está integrado uma placa impressa com outros componentes que têm podem ter limites de tensão bem inferiores. Pela primeira razão apontada, e como é sugerido pela página *web* oficial do projeto RepRap [21] a utilização de fonte de tensão de 12V, não fui ponderada a utilização de outra fonte.

Como se pode verificar, os gráficos 4.5 e 4.6 não apresentam os valores da curva velocidade-troque de 12V, o que dificulta uma análise concreta. Mais ainda, o facto de se ter obtido valores de torque dos eixos bastante menores que os assinalados nas figuras, torna a avaliação ainda mais incerta. Na verdade, com estes resultados, apenas através de experimentação prática é possível determinar a velocidade máxima de cada eixo. Essa análise é realizada no capítulo 5.

4.3.2 - Análise face aos modos de funcionamento

O modo de funcionamento do motor está intrinsecamente ligado ao *driver* escolhido, mais propriamente ao modo como o *driver* alimenta o motor.

O *chip* A3982 da Allegro, presente na placa de condução da eletrónica RepRap, trata-se de um driver de motores passo-a-passo bipolares que consegue funcionar em modo *full step* e *half*

step. Deste modo seria possível escolher entre maior velocidade/torque ou uma maior resolução, respetivamente. Segundo a ficha técnica [45] deste componente a escolha entre os dois modos de funcionamento é efetuada através do pino denominado MS1. Se esta porta for alimentada com um sinal lógico baixo, o *driver* funciona em modo *full step*. Por outro lado, se ao pino MS1 chegar um sinal alto, então este opera no modo *half step*.

Apesar da versatilidade apresentada pelo *driver*, o esquemático da placa de condução RepRap utilizada no projeto mostra que esse mesmo pino já se encontra pré-ligado ao sinal VCC, ou seja, à fonte de alimentação 5V, que em termos lógicos significa um sinal alto. A figura 4.6 pretende demonstrar o supracitado.

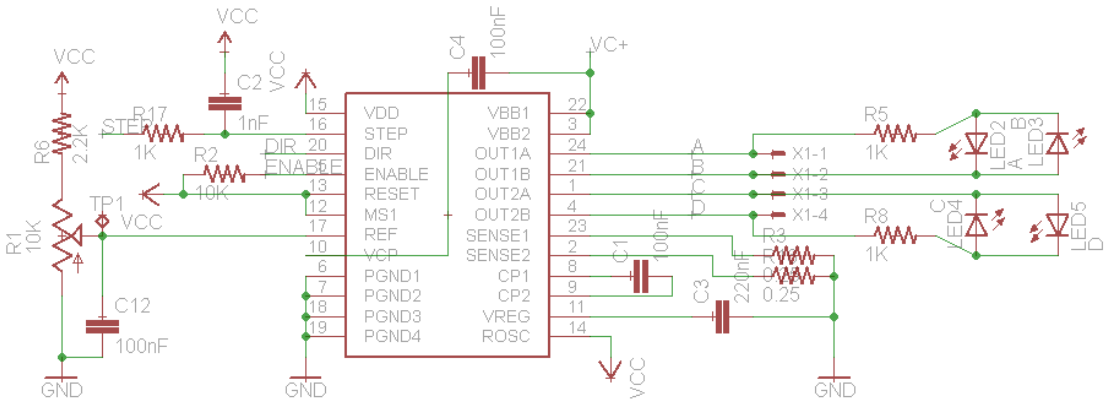


Figura 4.6 - Esquema do driver A3982 na placa de condução RepRap [44]

Neste modo, obtém-se o dobro da resolução, o que aumenta consideravelmente a precisão do motor, assim como uma rotação mais suave. Todavia, tal como se pode analisar pelo estudo realizado sobre o motores passo-a-passo presente no anexo A2, o modo de funcionamento em *half step* faz diminuir a capacidade de torque em cerca de 30%. Com uma maior resolução, é necessário um maior esforço por parte do controlador para gerar o dobro dos impulsos que o modo *full step*.

4.4 - Extrusora e o seu controlo

De acordo com o referido no Capitulo 2.4, a extrusora é dividida essencialmente em duas partes, o *cold end* e o *hot end*. A primeira inclui a estrutura física e o motor, enquanto que a segunda é constituída pelo cilindro de aquecimento e pelo termístor.

Para este projeto foi disponibilizada uma extrusora pré-montada com estrutura física de madeira, segundo o modelo Wade's Greared Extruder [21], um dos modelos mais utilizados nas impressoras 3D RepRap. No entanto, é essencial ressalvar que não foi possível obter qualquer informação sobre os aspetos técnicos individuais de cada componente, já que a ferramenta foi adquirida antes do início da dissertação, em forma de *kit*, que não descriminava os seus elementos constituintes. Por essa razão as características de cada componente só puderam ser conhecidas de forma empírica.

A componente eletrónica de controlo é fundamental para ligar o LinuxCNC à extrusora. A razão pela qual a sua utilização é essencial prende-se com o facto de o LinuxCNC não estar preparado nativamente para controlar diretamente uma impressora 3D. Então, utilizando o material disponível para o projeto, o controlo da impressora foi realizado na placa RepRap versão 2.2, criada exclusivamente para essa tarefa.

Esta placa controla a temperatura do cilindro de aquecimento e a velocidade do motor alimentador de filamento de plástico através de um clone Arduíno, o Atmega168. Este pequeno microcontrolador de 8 *bits* possui 16KB de memória *flash* ISP.

4.4.1 - Condução do motor da extrusora

Para a condução do motor integrado na extrusora, a placa impressa RepRap possui dois *drivers* de motores DC. Estes tratam-se de dois Allegro A3949, que conduzem de acordo com a lógica de abertura e fecho dos transístores da única ponte H que possuem no seu interior. O objetivo seria conduzir, com os dois *drivers*, um motor DC que puxasse o filamento num sentido e o outro motor DC para puxar no sentido inverso.

Contudo, motor presente na estrutura da extrusora trata-se de um motor passo-a-passo. Tipicamente, este tipo de motores requerem circuitos de condução com duas pontes H para a sequenciação lógica de energização dos seus enrolamentos.

A figura 4.7 mostra a diferença entre os *drivers* para motores passo-a-passo e *drivers* para motores DC, de acordo com o afirmado.

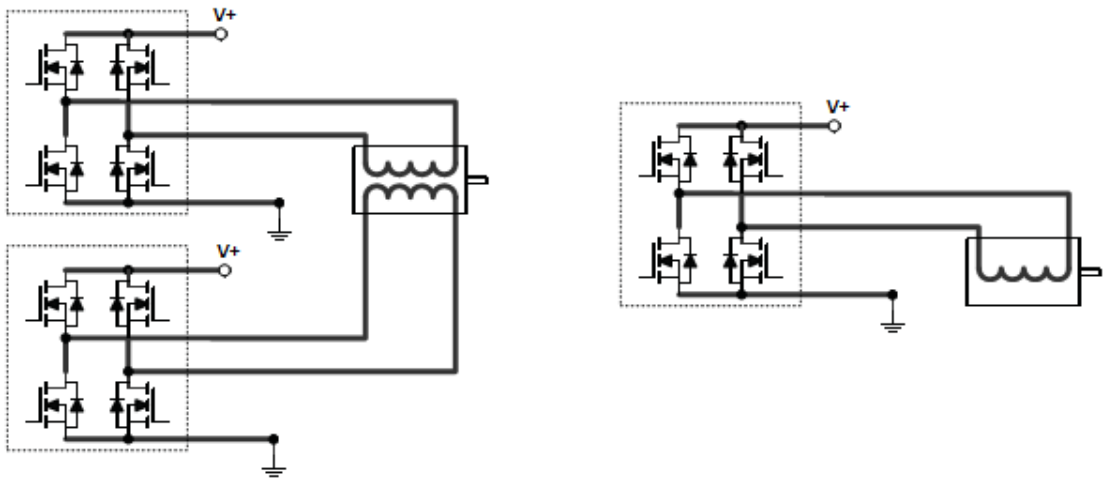


Figura 4.7 - Ligação dos drivers aos motores passo-a-passo bipolares (esquerda) e motores DC (direita)

Na realidade, para conduzir um motor passo-a-passo com recurso a *drivers* de motores DC, basta utilizar dois *drivers* deste tipo. Como cada um deles possui uma ponte H, ao serem usados 2, obtêm-se o circuito necessário para alimentar o motor adequadamente. A figura 4.7 serve para demonstrar isso mesmo.

Ainda assim, é importante ter em atenção que um *driver* para motores passo-a-passo como o Allegro A3982, possui internamente a lógica necessária para conjugar as duas pontes de forma correta. Assim sendo, a simples conjugação de dois *drivers* de motores DC, como os Allegro

A3949, não é suficiente. Todavia, é possível usar o Atmega168 existente na placa RepRap para executar uma rotina de controlo a jusante que permita obter o resultado desejado. Por outras palavras, como a comutação dos transístores das pontes H de cada um dos *drivers* é controlado pelo Atmega, programado com um *firmware* apresentado no capítulo 5, mais propriamente na secção 5.2.2.

4.4.2 - Leitura e definição da temperatura

A placa RepRap conta ainda com três MOSFET's NIF5003, denominados por A, B e C, sendo que o segundo está ligado a um pino PWM e os outros dois a pinos digitais. Segundo a sua ficha técnica, estes MOSFET's são capazes de operar sobre 42V e 14A, bem acima das condições de funcionamento da placa, fixada nos 12V da fonte de alimentação. Tipicamente são usados para ligar ao Atmega168 uma válvula para ajudar no controlo do fluxo de plástico, um aquecedor cilíndrico para derreter o plástico e ainda uma ventoinha para facilitar a solidificação do plástico, respetivamente. No projeto desenvolvido, apenas foi usado o segundo dos elementos referidos, por se tratar do único disponível e essencial. Naturalmente, o cilindro aquecedor usa o MOSFET B para se obter um controlo por PWM.

Por sua vez, o Atmega168, responsável pela regulação da temperatura, necessita de um mecanismo de *feedback* que lhe indique constantemente a temperatura a que o cilindro se encontra, de modo a atuar da melhor forma. Para isso foi usado um termístor encostado ao cilindro aquecedor. Estes são dispositivos elétricos que mudam a sua resistência termicamente, ou seja, o valor da resistência é alterado conforme a temperatura a que estão sujeitos. Assim, através da alteração dos valores da resistência, e consequentemente, da corrente aos seus terminais, é possível determinar a temperatura do cilindro. Mais uma vez, não existe informação sobre o termístor disponível, no entanto, tipicamente as extrusoras RepRap são acompanhadas por termístores NTC de 100kΩ a 25ºC [78].

Note-se que, originalmente, a *board* de controlo da extrusora foi criada para receber comandos de extrusão da *motherboard* RepRap. Esta, dava início ao processo de controlo por parte do microcontrolador da placa de extrusão. Por sua vez, este seria programado com um *firmware* adequado à versão de eletrónica usada, disponibilizado pelo projeto RepRap. No entanto, como se pretende que o LinuxCNC substitua por completo a *motherboard*, é necessário cria a ligação física entre o LinuxCNC e a placa de controlo e desenvolver e adaptar *scripts* dos dois lados. Uma vez mais, este último tópico é abordado com maior detalhe no capítulo 5.

4.5 - Viabilidade do controlador

Por fim, uma vez dimensionados os eixos, motores e *drivers,* no final da cadeia encontra-se a análise da viabilidade do LinuxCNC para controlar a impressora 3D, face ao computador atribuído para realizar esta tarefa. Nesta fase foram realizados testes à porta paralela para garantir o seu funcionamento e testes de latência para verificar se o computador escolhido cumpria os requisitos temporais impostos pelo LinuxCNC.

O computador atribuído para assumir a tarefa de controlador foi um HP, com processador Intel Core 2 Quad Q9300 a 2.5Ghz com 6MB de Cache L2, 4GB de RAM. Segundo os requisitos mínimos do *software* LinuxCNC, este PC tem a capacidade de o executar.

4.5.1 - Porta paralela

Originalmente, esta interface de comunicação entre um computador e um periférico externo, era usada para controlar uma impressora convencional. Atualmente a sua utilização foi bastante reduzida com a introdução dos cabos USB.

A ligação com o exterior dá-se através de conector de 25 pinos que tipicamente estava dividida em 3 grupos de pinos: 8 pinos de dados (D), 4 de controlo (C) e 5 de estado (S), além dos 7 de *ground*, como se pode observar na figura 4.8. Contudo, nos anos 90 foi introduzida a porta paralela que permitia que o grupo de pinos de dados pudesse ser usado tanto como *input* como output.

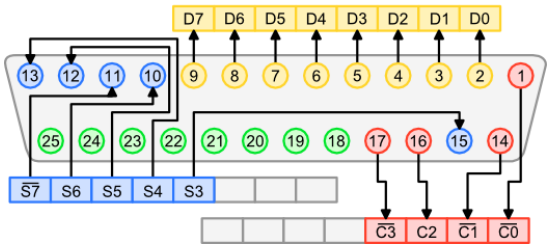


Figura 4.8 - Porta paralela DB25, interface fêmea [65]

Para verificar se o computador conseguia aceder e enviar comandos através da porta paralela, recorreu-se a uma pequena aplicação, o Parallel Port Tester, desenvolvida especificamente para testar a porta com o LinuxCNC, diretamente através do HAL [79].

Deste modo comprovou-se o funcionamento da porta paralela e verificou-se que o sinal de saída alto tinha o valor de 3,3V.

4.5.2 - Teste de latência

O LinuxCNC disponibiliza uma ferramenta com o nome de *Latency Testm* ou teste de latência, visível na figura 4.9, para testar se o PC onde está instalado é capaz de cumprir com os requisitos temporais para o controlo de uma máquina.

O teste consiste em por o computador à prova, usando a maior quantidade possível de recursos do CPU, para verificar qual a pior latência que ele atinge.

Segundo o manual do LinuxCNC [66], se o valor *Max Jitter* da Base thread for menor que 20 microsegundos, a performance do *software* será boa. Se esse mesmo valor for maior que 20 e menor que 50 microsegundos, o PC poderá ser usado embora o uso de *microstepping* possa estar condicionado. Se o resultado for maior que 50 microsegundos, o computador não será uma boa escolha para o controlo.

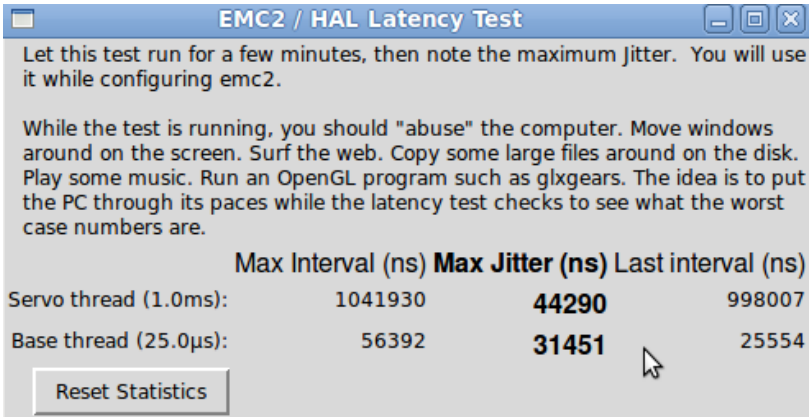


Figura 4.9 - Resultados do teste de latência LinuxCNC

Tal como se pode verificar, o resultado obtido foi de 31451 nanosegundos, o que, segundo o supracitado, significa que o PC cumpre os requisitos temporais, embora o uso de *microsteps* seja limitado. Como já se verificou na secção anterior, neste projeto desenvolvido os motores funcionaram em modo *half step*, por essa razão, o computador escolhido apresenta as características necessárias para ser o controla da impressora.

Depois de concluído o estudo deste capítulo, a figura 4.10 apresenta, de modo simplificado, o esquema da impressora 3D dimensionada.

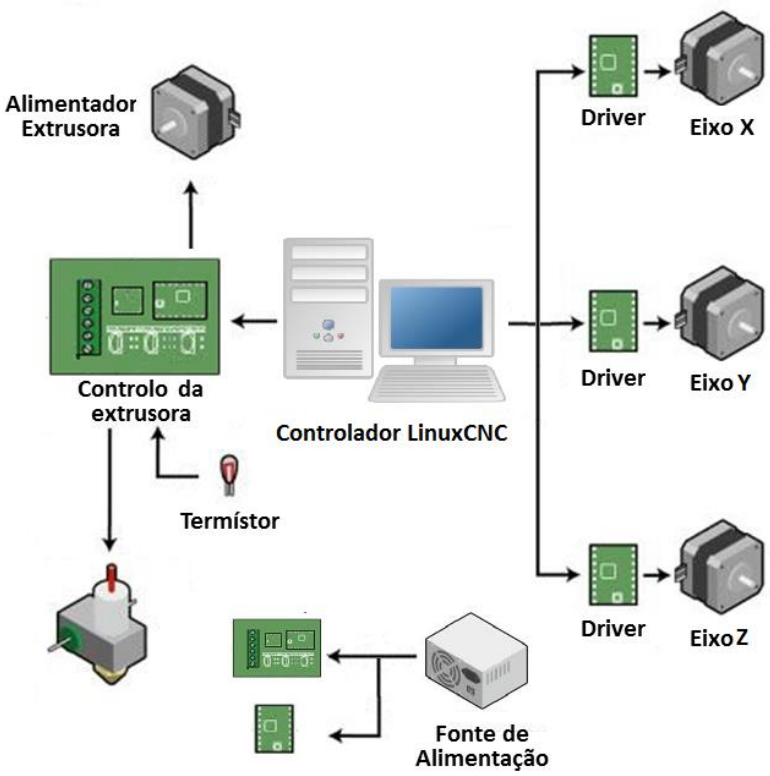


Figura 4.10 - Esquema simplificado da impressora 3D dimensionada

Capítulo 5

Desenvolvimento da Impressora 3D - myPrinter

No presente capítulo são descritos os procedimentos efetuados para configurar a impressora 3D, incluindo as ligações entre os componentes dimensionados no capítulo anterior, a configuração do controlador e a solução encontrada para controlar a impressora 3D com recurso ao LinuxCNC.

O desenvolvimento do projeto foi dividido em duas partes essenciais que neste documento tem reflexo em dois subcapítulos distintos: as movimentações da máquina nas três dimensões e o controlo e integração da ferramenta de extrusão com o LinuxCNC. No final são apresentados os testes realizados para validar o sistema.

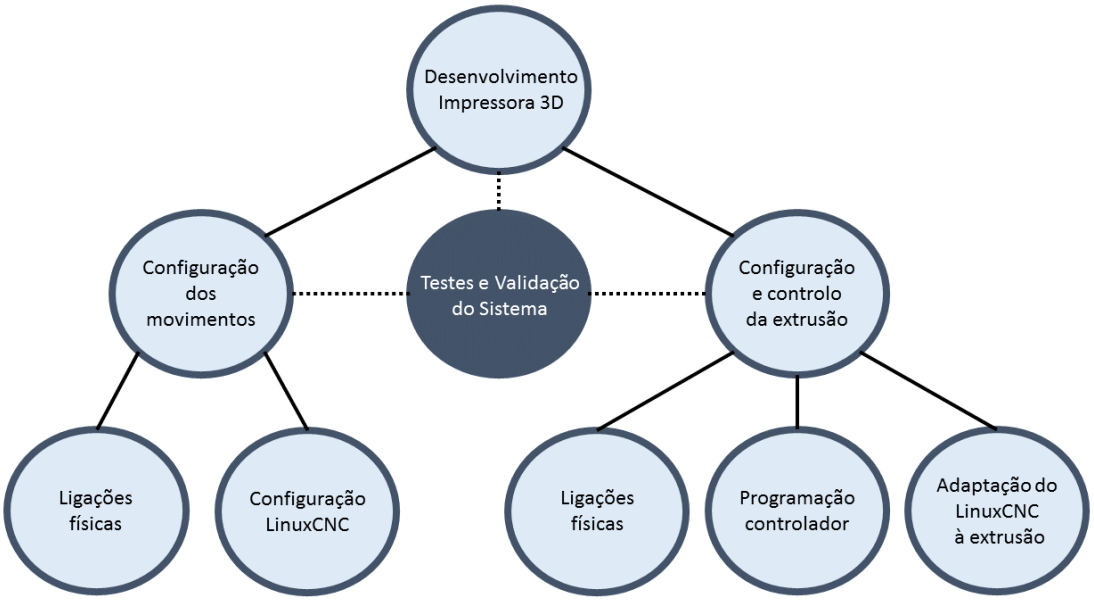


Figura 5.1 – Esquema das secções que constituem o capítulo 5

Na primeira parte deste capítulo é feita a exposição das principais tarefas e análises realizadas com o intuito de controlar os movimentos da impressora 3D, incluindo a ligação dos motores aos *drivers*, dos *drivers* ao PC controlador e por fim a configuração deste. Na secção seguinte são descritos os mecanismos usados e os métodos desenvolvidos para controlar a ferramenta de extrusão. Mais ainda, também nessa secção é apresentada a forma como foi feita a ligação e monotorização da extrusora com o LinuxCNC, através da placa controladora de extrusão do projeto RepaRap.

Para validar o sistema são executados alguns testes simples para verificar se a máquina desenvolvida tinha o comportamento desejado.

5.1 - Configuração dos movimentos

Na secção inicial do capítulo 5 é apresentado o trabalho realizado e as análises efetuadas para permitiram a integração e configuração da impressora 3D, quanto aos movimentos nas três dimensões.

Inicialmente, a configuração dos movimentos do sistema consistiu em integrar os motores com os *drivers* presentes nas placas de condução RepRap e as ligações destes ao controlador, através da porta paralela. No final da secção é demostrada a configuração do LinuxCNC face às características dos eixos da impressora 3D.

5.1.1 - Ligação motores-condução-controlo

Para a integração dos motores com os drivers foi inicialmente necessário descobrir os pares de polos dos motores de forma a garantir que os enrolamentos eram corretamente energizados pelos motores. Apesar das fichas técnicas dos motores apresentarem essa informação, na prática, para um maior rigor e certeza, foi realizado um pequeno teste experimental utilizando um multímetro para ler a resistência entre os terminais de cada motor. Todos os motores usados no projeto, tratando-se de motores passo-a-passo bipolares, possuem 4 fios, o que significa que existem dois pares ligados entre si, como mostra a figura 5.2.

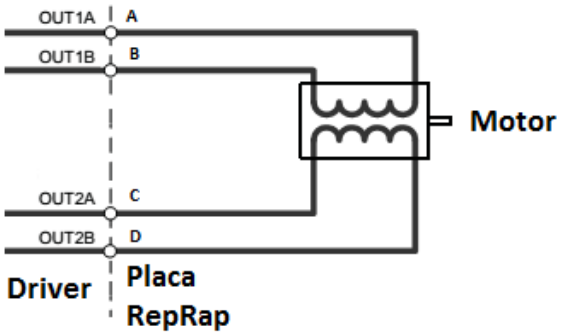


Figura 5.2 - Esquema da ligação dos motores dos eixos à placa de condução RepRap.

Como seria de esperar, a resistência entre os fios correspondentes às extremidades de um enrolamento (OUT1A/OUT1B e OUT2A/OUT2B) é muito menor face aos outros dois fios (OUT1x com OUT2x), pois os primeiros estão fisicamente ligados. Assim, através da medição da resistência de um dos fios face aos outros três, foram encontrados os pares de enrolamentos do motor.

A ligação dos fios dos motores aos terminais dos *drivers* Allegro A3982 foi realizada pela interface de quatro pinos que as das placas de condução RepRap, segundo o seu esquemático [44], e os pares de polos definidos, respeitando, mais uma vez, a figura 5.1.

Por outro lado, a conexão dos *drivers* ao PC controlador efetuou-se através da interface IDC exibida pelas *boards* RepRap, apresentada no fragmento do seu esquemático, na figura 5.3.

Os pinos STEP e DIR são utilizados para transmitir ao *driver* os sinais de passo e direção, respetivamente, e foram ligados aos pinos correspondentes da porta paralela - definidos no controlador, conforme pode ser analisado na secção 5.1.2. Por sua vez, o pino 5, denominado ENABLE, recebe um sinal do controlador, igualmente através da porta paralela, para dar início à energização dos motores. Este sinal é ativo ao nível baixo, ou seja, o *driver* só entra em modo de funcionamento quando lhe é aplicado nesse pino uma tensão próxima dos 0V. Como se pretende que os três eixos entrem em funcionamento simultaneamente, o mesmo sinal proveniente do controlador foi partilhado pelas três placas de condução.

Em contrapartida, os pinos MAX e MIN são utilizados para sinalizar ao controlador que foi acionado um *limitswitch*, ou seja, que o sistema chegou a um dos seus limites espaciais. Dado que estes dispositivos não foram usados no projeto, estes pinos não são conectados a nenhum sinal.

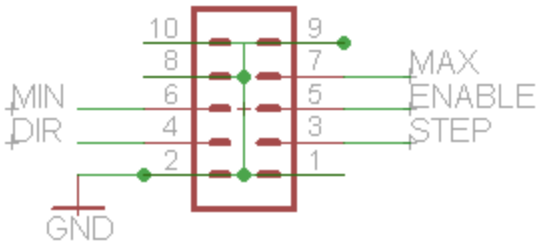


Figura 5.3 - Interface IDC de entrada da placa do *driver*

5.1.2 - Configuração LinuxCNC

De acordo com o exposto no capítulo 3, a configuração do LinuxCNC é feita através de um conjunto de ficheiro que definem a máquina a controlar. Para facilitar a criação destes ficheiros, o *software* disponibiliza uma pequena ferramenta, o *Stepconf*, que permite, através de uma interface gráfica, especificar de forma simples as características de uma máquina CNC de corte que utilize motores passo-a-passo para o seu posicionamento. Note-se que, apesar de este perfil referido não corresponder ao perfil da máquina desenvolvida neste projeto (obviamente trata-se de uma impressora 3D e não de uma máquina de corte), o importante é

serem gerados os ficheiros de configuração essenciais ao LinuxCNC, de modo a que este envie para a impressora os sinais de passo e direção.

As secções seguintes demonstram a configuração do controlador segundo os parâmetros solicitados pelo *Stepconf* e os atributos dos componentes constituintes da impressora 3D

5.1.2.1 - Configuração dos pinos da porta paralela

Dos dezassete pinos configuráveis da porta paralela doze são pinos de saída e cinco são de entrada, quando configurada como *output*. Os pinos de entrada não foram utilizados pois, em concordância com o que já foi exposto nos capítulos anteriores, para o controlo dos motores passo-a-passo não está prevista a utilização de qualquer tipo de mecanismo de feedback para conhecer a sua posição. Além disso, não foram utilizados *limitswitchs* para deteção dos limites físicos de viagem da máquina.

Por outro lado, no caso dos pinos de saída, o sistema desenvolvido apresenta três drivers de motores e cada um destes deve receber dois sinais, um de passo e outro de direção. Mais, devem também admitir um sinal de *enable* para entrarem em funcionamento, sendo que este último pode e deve ser partilhado pelos três. Então, são necessários no mínimo sete pinos para a sua correta configuração.

A figura 5.4 mostra como foram configurados os pinos da porta paralela de acordo com o supracitado, através da interface do configurador.

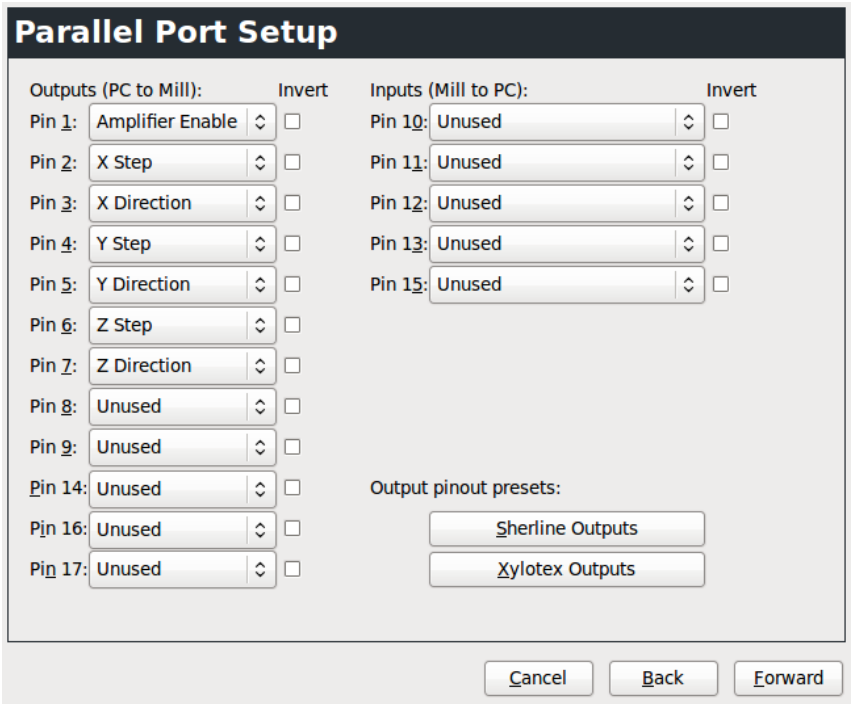


Figura 5.4 – Interface S*tepconf* de configuração da porta paralela

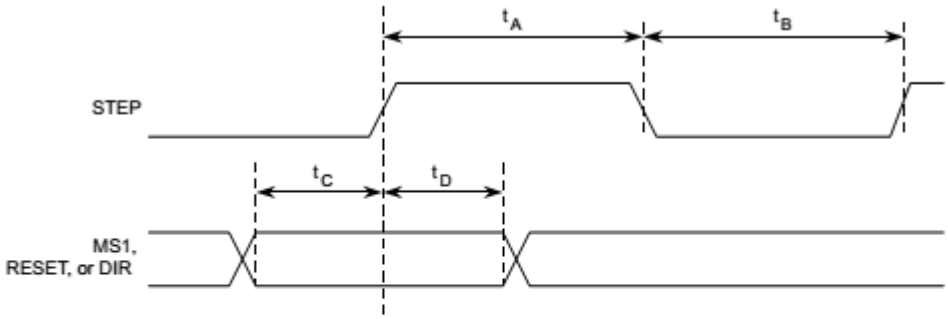
Dado que o LinuxCNC não permite o controlo nativo de ferramentas aditivas, não foi configurado nenhum pino da porta paralela relativo à extrusora.

5.1.2.2 - Características dos *drivers*

De modo a respeitar as restrições temporais impostas pelos circuitos de condução para que estes traduzam corretamente os sinais de passo e direção que recebem, foi necessário fornecer ao LinuxCNC os seguintes atributos:

- ***Step time*** – por quanto tempo o sinal de passo fica ativo;
- ***Step space*** - o tempo entre sinais de passo;
- ***Direction hold*** - por quanto tempo o sinal de direção é mantido depois de uma mudança de direção;
- ***Direction setup*** – quanto tempo antes de uma mudança de direção depois do último passo.

Conforme previsto na ficha técnica do *diver* Allegro A3982 [s2], cujo fragmento relevante para esta secção se exibe figura 5.5, este conta com tempo de passo, ou seja o seu *step time*, de 1µs. Por sua vez, como o *step space* corresponde ao tempo mínimo entre sinais de passo, então, o tempo mínimo que o driver passa em sinal baixo, representado na figura 5.5 pelo símbolo t_B, corresponde a esse mesmo valor e está definido com 1µs. Por outro lado, no que toca ao *direction hold e direction setup,* estes encontram correspondência nos tempos da *datatasheet* apelidados de *setup time* e *hold time*, logo têm uma restrição temporal na ordem dos 200ns.



Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figura 5.5 - Restrições temporais do driver A3982 da Allegro

Por último, ainda em relação as restrições temporais, também é solicitado pelo *Stepconf* o valor do *Base Period Maximum Jitter*. Esta variável deve ser definida com o tempo obtido no teste de latência. Recorde-se o resultado obtido no teste de latência encontra-se no capítulo 4 e foi de 31451ns.

5.1.2.3 - Configuração dos eixos

Na fase de configuração dos eixos cartesianos procedeu-se ao cálculo do número de passos necessários para fazer mover o motor um milímetro linear, foram definidas as dimensões físicas do trajeto de cada eixo e por fim, através de testes empíricos, foram atribuídos valores para a velocidade e aceleração de cada eixo, individualmente.

Relembrando o que foi estudado no capítulo 2.3 sobre motores passo-a-passo, o número de passos para o motor girar uma volta completa é obtido através da equação 2.3.

$$Nº\ de\ passos = \frac{Ângulo\ de\ rotação}{Ângulo\ de\ passo} * \ Nº\ de\ divisões\ entre\ enrolamentos \tag{2.3}$$

Como se pode comprovar pelo capítulo 4, os três eixos apresentam parafusos sem-fim com as mesmas características físicas. Em relação aos motores, os três possuem ângulos de passo iguais. Mais ainda, foram usados três *drivers* iguais para a condução dos três motores. Assim sendo, o estudo sobre o número de passos por milímetro é aplicável aos três eixos.

Os motores passo-a-passo selecionados apresentam um ângulo de passo de 1.8º. E como já foi analisado, os *drivers* propostos funcionam em modo *half step*. Logo,

$$Nº\ de\ passos = \frac{360^0}{1{,}8^0} * \ 2 = 400 \tag{5.1}$$

Por sua vez, os parafusos sem-fim eleitos apresentam uma distância entre dentes, ou *pitch*, de 2mm. Deste modo o MPR, milímetros por revolução é obtido por

$$MPR\ = \frac{1}{pitch} = \frac{1}{2} = 0{,}5 \tag{5.2}$$

De seguida foi calculado o produto entre o número de passos obtidos na equação 5.2 e o valor do MPR, aqui denominado por PPM, passos por milímetro.

$$PPM\ =\ \text{Nº de passos} * \text{MPR}\ =\ 400 * 0{,}5 = 200 \tag{5.3}$$

Dado a razão de conversão ser 1:1, a rotação completa do rotor do motor implica a rotação completa do parafuso sem-fim, o que significa que o número de passos se mantem. Assim, o valor final é de 200 passos por milímetro.

Em relação ao tamanho do trajeto, localização inicial e velocidade e aceleração de cada eixo, a tabela 5.1 mostra os valores definidos e de seguida é explicado cada um dos parâmetros.

Tabela 5.1 – Valores de configuração dos eixos X, Y e Z

	Eixo X	Eixo Y	Eixo Z
Início do eixo (m)	0,1	0,1	0
Fim do eixo (m)	0,6	0,6	-0,2
Localização inicial (m)	0,15	0.15	-0,05
Velocidade máxima (m/s)	0,013	0,02	0,01
Aceleração máxima (m/s^2)	1,8	2,1	1,5

Os dois primeiros parâmetros da tabela 5.1 referem-se ao trajeto máximo de cada um dos eixos. Assim, o tamanho do trajeto pode ser simplesmente deduzido subtraindo o valor da posição máxima no final e início do eixo. Na prática, estão a ser definidos os limites de trajetória impostos pelo *software* para cada eixo. Na verdade, tal como já se viu, a estrutura mecânica dos eixos X e Y possuem uma trajetória total de 0.75m e a do eixo Z têm um comprimento de 0.25m. No entanto, devido às características de montagem, optou-se por apenas usar um trajeto de 0.5m para os eixos X e Y, e 0.2 para o eixo Z. Mais ainda, deste modo fica assegurada uma margem de segurança para evitar que os carrinhos de cada eixo forcem o seu movimento contra o limite físico da estrutura, por algum erro do sistema.

A terceira linha da tabela 5.1 mostra a posição de origem de cada eixo. Este parâmetro é importantíssimo para a máquina conhecer a sua posição constantemente. Sabendo a posição em que inicia o seu movimento, o controlador é capaz de saber exatamente a posição da máquina tendo em conta o tendo em conta o número de passos por milímetro já calculados. Deste modo, pode até prever eventuais quebras dos limites de trajetória e impedir que a máquina comece a maquinar. Para o seu correto funcionamento, a cada nova utilização da máquina, é fundamental colocar o eixo na posição *home*, ou seja posição inicial prevista no *software*. Segundo o configurador, este valor deve-se encontrar dentro dos valores das duas primeiras linhas da tabela 5.1, não podendo ser iguais aos limites do intervalo. Assim, optou-se por deixar uma margem de 0.05m para cada eixo em relação ao seu início.

Por fim, os dois últimos valores prendem-se com a velocidade e aceleração máxima que cada eixo consegue alcançar. De acordo com a documentação do LinuxCNC [62], o dimensionamento da velocidade e aceleração deve ser realizado experimentalmente com o auxílio de uma pequena ferramenta que o configurador disponibiliza. Para este teste começou por se aplicar valores relativamente baixos para a velocidade, verificando o comportamento do motor a cada novo incremento. A velocidade máxima foi definida pela maior velocidade cujo motor não bloqueava ou perdia passos, deixando uma margem de segurança. Depois de definida a velocidade máxima, o procedimento para encontrar a aceleração máxima foi idêntico.

No final da definição da máquina pelo configurador *Stepconf*, o LinuxCNC deu origem a um conjunto de ficheiros que refletem os parâmetros definidos. De salientar, no entanto, que durante a configuração da porta paralela, o pino responsável pelo sinal de *Enable*, não foi invertido. Como apontado na secção 5.1.1, a sua inversão é essencial para o correto

funcionamento dos *drivers*, dado que estes só entram em modo de condução quando chega um sinal de nível baixo ao seu pino *enable*. De forma a corrigir este sinal foi necessário editar o ficheiro HAL onde estão definidos cada um dos pinos da porta paralela. Assim, foi necessário adicionar ao ficheiro HAL a seguinte linha:

```
setp parport.0.pin-01-out-invert 1
```

Este ultimo ajuste permitiu concluir a configuração da impressora 3D quanto aos movimentos.

5.2 - Configuração e controlo da extrusora

A segunda parte da configuração da impressora 3D desenvolvida tem como objetivos a integração da ferramenta de extrusão com a eletrónica de controlo RepRap e a posterior ligação desta ao PC controlador com o LinuxCNC. Pretende-se, assim, documentar o trabalho desenvolvido nesta vertente, analisar as propostas de projetos similares e apresentar a solução encontrada para projeto.

Este estudo foi dividido em 3 partes fundamentais: as ligações necessárias entre os componentes que constituem este subsistema, o desenvolvimento do *firmware* e programação do controlador de extrusão e, por fim, a integração do LinuxCNC com a extrusora e o seu controlo.

5.2.1 - Ligação extrusora-controlo-LinuxCNC

Do mesmo modo que para os motores dos eixos, também o motor passa-a-passo usado como alimentador de filamento da extrusora se trata de um motor bipolar. Assim, usando a mesma técnica descrita no ponto 5.1.1, rapidamente se obteve os dois pares de fios que constituem cada um dos enrolamentos. O esquema da figura 5.6 demonstra a ligação feita dos quatros fios do motor à placa RepRap de controlo da extrusão.

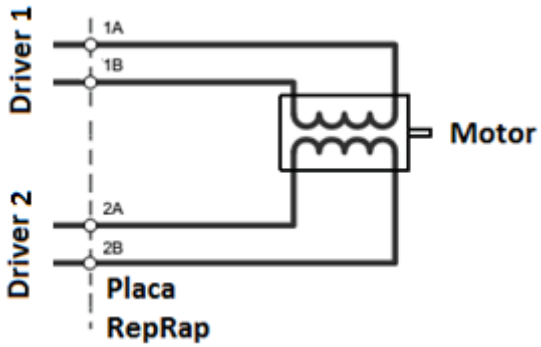


Figura 5.6 - Esquema da ligação do motor da extrusora à placa de controlo RepRap

Por outro lado, o termístor e o cilindro aquecedor foram ligados às portas correspondentes na mesma placa. Tanto para um caso como para o outro, a sua ligação não obedece a nenhum requisito especial já que não é necessário respeitar nenhuma polaridade.

Por outro lado, apesar do controlo da extrusora ser efetivamente realizado na placa RepRap, é essencial garantir que LinuxCNC possui controlo sobre o *timing* de extrusão, de modo a esta estar em sincronia com os movimentos da máquina. Ou seja, é necessário uma ligação entre o controlador mestre (LinuxCNC) e o controlador escravo (placa de controlo RepRap).

Como a porta paralela é usada pelo LinuxCNC para o controlo dos movimentos, a solução encontrada foi efetuar a ligação do PC controlador com o conector série da placa de controlo da extrusão através de porta série RS-232 de 9 pinos do computador. No entanto, os níveis lógicos da porta paralela variam entre ±12V, enquanto que a placa espera receber sinais lógicos TTL, ou seja entre 0 e 5V - na verdade, os níveis lógicos TTL, tipicamente, partem do princípio que níveis inferiores a 0,8V se trata de um 0 e níveis superiores a 2V são considerados um 1 lógico.

Para realizar a conversão dos sinais foi usado o CI MAX232N. A utilização deste componente é relativamente simples, bastando seguir as indicações da sua ficha técnica, nomeadamente o esquema que apresenta onde circuito típico de funcionamento [80]. Assim, o circuito montado obedece ao esquema da figura 5.7, onde se pode observar que foram utilizados condensadores de 1µF.

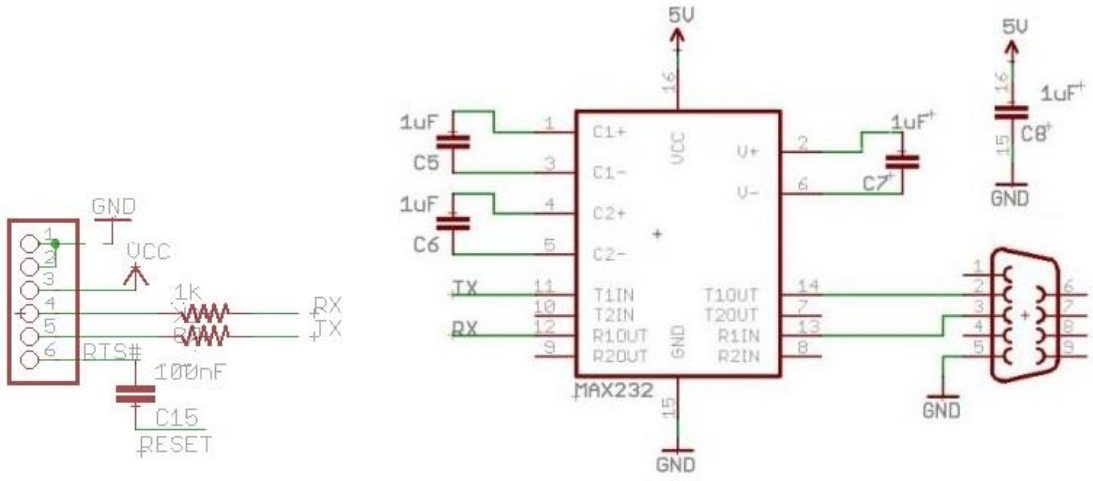


Figura 5.7 - Circuito com CI MAX232N, usado para conversão de sinal da porta série

Esta mesma figura mostra ainda a entrada série TTL de 6 pinos existente na placa RepRap.

A comunicação serie usa fundamentalmente 2 fios para ligar a dois pinos diferentes em cada um dos dispositivos que se quer conectar, um para transmitir informação (TX) e outro para receber (RX). Por essa razão, é importante ressalvar que o pino RX da placa controladora deve ligar ao pino TX do PC, e vice-versa. Além disso, os dois componentes devem partilhar a mesma terra (GND).

5.2.2 - Programação do controlador da extrusora

Como se pode analisar pela figura que demonstra a ligação do motor à placa RepRap (figura 5.6), e como já havia sido esclarecido no capítulo 4, nomeadamente na secção 4.4, esta placa possui dois *drivers* Allegro A3949, desenhados para controlar dois motores DC. Contudo, estes pode ser usados para conduzir um motor passo-a-passo, desde que seja aplicado o devido controlo a jusante.

Para isso, a eletrónica RepRap de controlo possui um microcontrolador Atmega168. Recorde-se que o Atmega168 trata-se de um clone de Arduíno e, por essa razão, a programação deste microcontrolador, presente na placa RepRap, realiza-se com auxílio de um programador AVR. Este periférico permite usar o *software* Arduíno para carregar programas para o *chip do* Atmega, fazendo uso da porta USB do lado do PC e da interface ISP do lado da placa de controlo da extrusora RepRap.

O projeto RepRap disponibiliza todo o código necessário, denominado RepRap *3rd Generation Firmware*, para programar o Atmega168, no contexto da criação de uma impressora 3D RepRap. Todavia, o código apresentado parte do princípio que o controlo da impressora 3D é efetuado pela *motherboard* RepRap, mas, na realidade, o objetivo nesta fase do projeto é usar o LinuxCNC para substituir por completo a utilidade da placa-mãe. Por essa razão, naturalmente se chegou à conclusão que o *firmware* RepRap não poderia ser simplesmente carregado para o controlador.

Assim sendo, foi necessário não só modificar, adaptar e implementar código para corresponder a esta nova configuração, como também aplicar um protocolo de comunicação série entre a placa e o PC controlador.

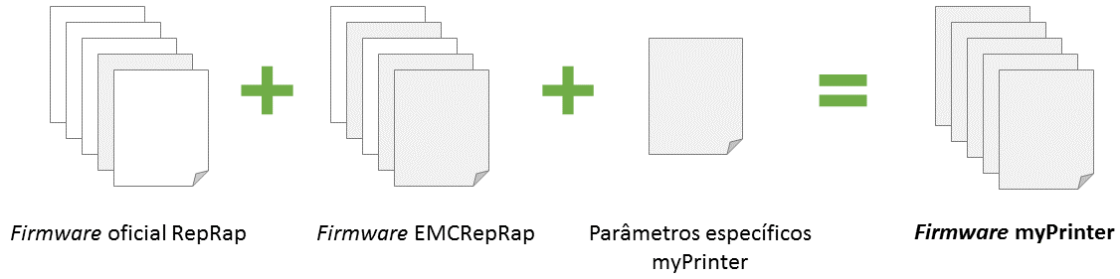


Figura 5.8 - *Firmware* myPrinter nasce da adaptação e modificação de código de diferentes fontes

De acordo com que foi exposto no capítulo 2.6, este problema foi endereçado por um ínfimo número utilizadores do LinuxCNC e do projeto RepRap, nomeadamente com a criação do subprojeto EMCRepRap. Uma análise ao código partilhado por alguns membros desta comunidade revelou que este estava adaptado ao uso original de dois motores DC, e não a motores passo-a-passo como o pretendido para este caso. No entanto, apesar de, recorde-se, não poder ser utilizado o código original RepRap para o controlo da extrusora, este contempla a possibilidade de utilização de um motor passo-a-passo como alimentador da extrusora. Segundo estes prossupostos, para realizar o controlo da extrusora corretamente focou-se em compreender e interpretar o *firmware* para o controlador de extrusão EMCRepRap e o *firmware*

original da geração 3 RepRap para, mais concretamente, se encontrar uma forma de adaptar o código das duas fontes e fazer as modificações necessárias (figura 5.8).

5.2.2.1 - Estruturação do código do controlador

Os códigos de configuração são, tipicamente, divididos em vários ficheiros, como se fossem módulos de um sistema. Neste caso em particular, foram divididos segundo os elementos que constituem a extrusora, somando-lhes os ficheiros com o código necessário para a comunicação série com o LinuxCNC.

No projeto de dissertação a estruturação do código foi efetuada tendo por base, maioritariamente, dois subprojetos EMCRepRap [81, 82]. No entanto, foram feitas algumas modificações sempre que necessárias, nomeadamente no que toca ao motor, seguindo-se, nesse caso, o exemplo do *firmware* original RepRap.

- **Communication.ino**

Este ficheiro contém o código responsável por interpretar os comandos do LinuxCNC. Faz a gestão dos pacotes de dados recebidos, invoca as funções presentes nos outros ficheiros e envia os resultados obtidos da *query* pedida para o protocolo de comunicação série (SimplePacket.cpp).

Tendo em conta um dos subprojectos EMCRepRap, foram retirados os comandos sem significado para o projeto. Assim, neste caso em particular, é capaz de receber 7 comandos diferentes:

- Ligar/desligar extrusora;
- Ler/definir temperatura;
- Ler/definir velocidade do motor;
- Estado da comunicação.

A extensão .ino é a extensão típica usada pelo Arduino IDE, e usa uma linguagem bastante similar ao C.

- **Configuration.h**

Neste bloco de código são configurados alguns aspetos da comunicação, como por exemplo, a velocidade da comunicação série ou ainda o tempo máximo da espera por pacote. Do mesmo modo, aqui são definidos os aspetos técnicos do motor, como o tipo, o modo de funcionamento e o número de passos por revolução. Aqui também são declarados pinos de comunicação com o Atmega168, nomeadamente o RX e TX para ligar com o LinuxCNC, os pinos de passo e direção para comunicar aos *drivers,* os pinos do cilindro de aquecimento e do termístor. Todos estes parâmetros tiveram de ser adaptados à impressora 3D myPrinter.

A extensão deste ficheiro é .h, ou seja, trata-se de um ficheiro *header*, onde são declaradas as funções que normalmente são partilhadas por várias parte do código. A sua utilização é também bastante usual em programação com linguagem C.

- **ExtruderController.ino**

Trata-se do ficheiro principal do *firmware*. Aqui são invocados todos os outros ficheiros contendo as funções para ler e atuar sobre os componentes da extrusora. Apresenta também a lógica de atuação nos *drivers* do motor passo-a-passo, implementado tendo em conta o *firmware* RepRap e modificações específicas da máquina desenvolvida.

- **Heater.h**

Trata-se do ficheiro onde estão definidas as funções relacionadas com a temperatura. Este, está responsável por ler e atuar efetivamente sobre os pinos do termístor e do cilindro aquecedor, respetivamente. Para garantir resultados precisos são lidas 5 temperaturas e feita a média entre elas sempre que é pedido o valor atual da temperatura. Neste ficheiro é declarado o ficheiro Thermistor.h.

- **Motor.h**

Por sua vez, o fichiero Motor.h é onde estão definidas a funções relacionados com o motor. Da mesma forma que no caso anterior, também é aqui que os pinos do motor são ativos. A formação dos passos é obtida através da ativação de *interrupts* no Atmega168, no entanto a lógica de condução é realizada no ficheiro ExtruderController.ino. Como o motor passo-a-passo não usa mecanismo de *feedback,* o valor de leitura é o mesmo que o de escrita.

- **SimplePacket.cpp e SimplePacket.h**

Protocolo de comunicação série *standard* em C++. Divide as mensagens em pacotes que ambos os dispositivos ligados, por comunicação série, reconhecem. Neste caso em particular, o protocolo permite o envio de comandos e de dados por parte do LinuxCNC e da placa controladora RepRap, respetivamente.

- **Thermistor.h**

Este ficheiro contém a *lookup table* do termístor. O seu objetivo é traduzir o valor da resistência medido, para Kelvin's. Este ficheiro é gerado através da execução de um *script python* num PC, bastando para isso introduzir as características do termístor. Este script é fornecido pelo projeto RepRap.

Dado que não são conhecidas as características do termístor do projeto, o ficheiro Thermistor.h foi criado usando as propriedades de um termístor usual nas impressoras RepRap.

Após a compreensão, adaptação, efetuados modificações necessárias e posterior implementação do *firmware* desenvolvido, foi concluída a programação do controlador da extrusora. No entanto, apenas foi possível efetivamente testa-lo depois de ser implementada a capacidade ao LinuxCNC de comunicar com placa controladora da extrusora RepRap.

5.2.3 - Adaptação do LinuxCNC à tecnologia de extrusão

Finalmente, uma vez realizada a programação do Atmega168 da placa RepRap com o *firmware* desenvolvido, a última grande tarefa do projeto consistiu em efetivar a integração da ferramenta de extrusão com o LinuxCNC. Para atingir este objetivo foi necessário

implementar um conjunto de funcionalidades adicionais para permitir que o LinuxCNC envie comandos de leitura e escrita da temperatura e velocidade do motor, através da porta série.

Para esta etapa do projeto voltou a ser fortemente estudada a solução encontrada pelos subprojetos EMCRepRap. De forma resumida, do lado do PC controlador deve ser implementado um conjunto de *scripts* que permitem a comunicação entre o LinuxCNC e o *firmware* desenvolvido. Mais ainda, foram também desenvolvidos *scripts* para adaptar a saída do *software* CAM, o G-code, de modo a que seja possível este contemplar as funcionalidades extras requeridas para a comunicação série.

Assim, a análise dos *scripts* foi dividida em duas fases. A primeira contempla a comunicação e monitorização do LinuxCNC em relação à placa RepRap. De seguida foram analisados os *scripts* a jusante do LinuxCNC, entre este e o *software* CAM.

5.2.3.1 - Comunicação e monotorização

De acordo com o que foi mencionado em cima, inicialmente o estudo recaiu sobre o conjunto de ficheiros EMCRepRap que permitem a comunicação e monitorização da ferramenta de extrusão através do LinuxCNC.

- **RepRapSerialComm.py**

Módulo que permite a comunicação com o controlador da extrusora, pela porta série do PC, e vice-versa. Neste ficheiro dá-se a criação dos pacotes de comunicação, de acordo com o protocolo série. Possui um elevado número de mecanismos de deteção de erros.

Dado o script disponível, apenas foi necessário especificar a porta pela qual a comunicação é feita e qual o ser *baudrate*.

- **repstrap-commtest.py**

Trata-se apenas de um *script* de teste para verificar a comunicação entre o LinuxCNC e a placa RepRap. Além disso também lê a temperatura do termístor para garantir que a sua ligação está correta. Naturalmente, usa o módulo RepRapSerialComm.py para este teste.

Quando foi efetuado este teste, o resultado obtido para a temperatura foi de 40ºC, à temperatura ambiente. Como é evidente, este resultado está muito longe da temperatura ambiente real verificada. A razão para este valor tão desfasado da realidade prende-se com o facto da *lookup table*, usada para converter a resistência do termístor em valores de temperatura, não ser a mais adequada. Como já foi visto, as características técnicas do termístor eram desconhecidas e por essa razão não foi possível definir corretamente a tabela de conversão.

- **repstrap-extruder.py**

Este é o ficheiro principal do conjunto de *scripts*. A sua primeira função é mapear os pinos HAL associados ao controlo e monotorização da extrusora. Só deste modo o LinuxCNC é capaz de interpretar os dados que recebe e enviar comandos de controlo, pela porta série. Mais ainda, este *script* é também responsável pela própria conversão da informação, como mostra a figura 5.9.

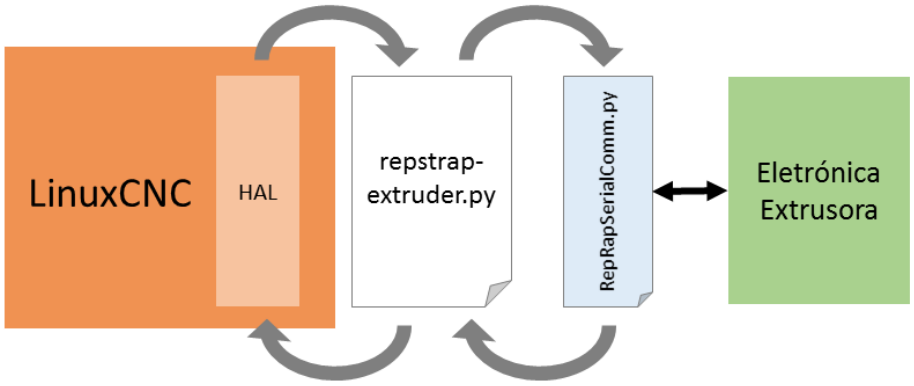


Figura 5.9 - Cadeia desencadeada pelo *script* repstrap-extruder.py

Também neste módulo foram definidos os comandos que são enviados para a placa da extrusora. Como visto nas secção anterior, são 7 os comandos utilizados.

Do mesmo modo que no caso do *script* RepRapSerialComm.py, é apenas necessário garantir que a porta série especificada no código corresponde à que está a ser efetivamente usada.

- **repstrap-extruder.hal**

Tal como já mencionado em várias ocasiões, o HAL comporta-se como uma *breadboard* virtual. Assim, o objetivo deste ficheiro é simplesmente ligar os pinos criados no ficheiro anterior e os pinos usados para controlar/monitorizar a máquina. Estes últimos pinos são usado pelo ficheiro repstrap-extruder-pyvcp para criar uma interface entre os pinos e o utilizador.

Para que seja possível esta união destes sinais é essencial invocar o *script* repstrap-extruder.py neste ficheiro HAL, através do comando *loadusr*.

De acordo com os subprojectos EMCRepRap estudados, é também neste ficheiro que se define o número de passo do motor por metro cúbico de filamento (PPMC). Para efeitos de teste, foi usado o valor pré-definido.

- **repstrap-extruder.pyvcp**

Finalmente, o LinuxCNC permite criar painéis gráficos para adicionar funcionalidades extra à interface gráfica AXIS, nomeadamente para atuar sobre entradas e saídas (I/O) especiais.

Assim, no ficheiro repstrap-extruder-pyvcp foi definido um conjunto de interfaces gráficas, como botões e sinalizadores, que atuam e monitorizam diretamente na impressora 3D, a partir do LinuxCNC.

Baseada nos exemplos da documentação do LinuxCNC e EMCRepRap, a figura 5.10 apresenta a interface criada para o projeto.

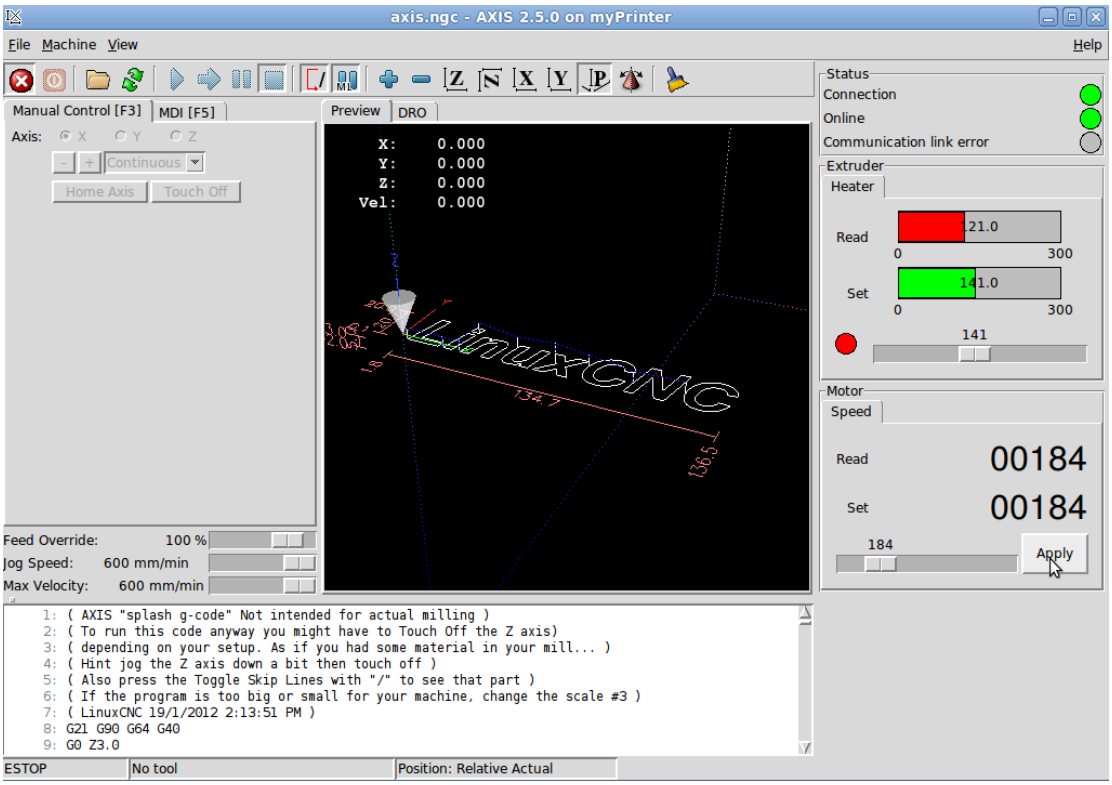


Figura 5.10 - Interface gráfica final do projeto myPrinter com painel de controlo e monitorização da ferramenta de extrusão

Recordando a figura 2.4 que apresenta a interface AXIS do LinuxCNC verifica-se que foi lhe foi adicionado um painel do lado direito da interface gráfica padrão do LinuxCNC. Este novo painel encontra-se dividido em 3 partes fundamentais:

- Na parte superior do painel é exibido o estado da impressora: se conectada, se ligada e se tem algum erro de comunicação; No caso da Figura 5.24, o sinal verde nos dois primeiros e a ausência de cor no terceiro, mostra que a impressora 3D está pronta a maquinar.
- De seguida é apresentado o controlo e monitorização da temperatura. A barra vermelha representa a temperatura lida e a barra verde mostra a temperatura definida. A barra cinza em baixo serve para modificar a temperatura manualmente e, por sua vez, o LED, quando aceso (cor vermelha) indica que o cilindro de aquecimento está ligado. No caso da figura 5.10, a temperatura foi definida para 141ºC, no entanto, como essa temperatura ainda não foi atingida (estão a ser lidos 121ºC), então o cilindro ainda está a ser alimentado até alcançar a temperatura pretendida;
- No fundo, o painel ostenta o controlo do motor. Igualmente com uma barra deslizante é possível definir manualmente a velocidade do motor.

De modo a que LinuxCNC reconheça o painel desenvolvido foi necessário instanciar estes ficheiros no ficheiro original de configurações do LinuxCNC, o ficheiro INI. Para isso, bastou adicionar a linha de código aqui apresentada:

```
[AXIS]
PYVCP = repstrap-extruder.pyvcp
```

Do mesmo modo, para o reconhecimento do ficheiro repstrap-extruder.hal, foi necessário apenas modificar, também no ficheiro INI, o valor da variável POSTGUI_HALFILE dentro da secção [HAL] de modo a que esta aponte para *script* correto.

5.2.3.2 - Configurações a jusante para a automatização da impressão

Com o painel integrado na interface do LinuxCNC e com todas as configurações efetuadas para estabelecer a ligação entre o PC controlador e a ferramenta de extrusão, foi necessário conferir ao LinuxCNC a capacidade de ativar a extrusão e definir a temperatura automaticamente, como o faz com uma ferramenta de corte. Ou seja, o LinuxCNC deve ser capaz imprimir um objeto, bastando para isso que lhe seja fornecido um ficheiro do tipo G-code.

Na verdade, como foi visto no capítulo 2, os *software* CAM ao traduzirem um ficheiro CAD para código que o software de controlo perceba (G-code), além do conjunto de movimentos que a máquina tem de efetuar, definem também os momentos em que as ferramentas atuam. Este conceito é valido tanto para impressoras 3D como para máquinas CNC convencionais. No entanto, para criar exatamente o mesmo objeto modelizado em CAD, o código resultante do processamento do *software* CAM será completamente diferente se especificado para corte ou para impressão. No primeiro caso será removido o material em excesso, por isso os movimentos da máquina serão maioritariamente à volta do objeto. Por outro lado, no caso da extrusão, os movimentos estarão sensivelmente confinados ao volume da peça. Isto pode ser observado de modo simplificado na figura 5.11.

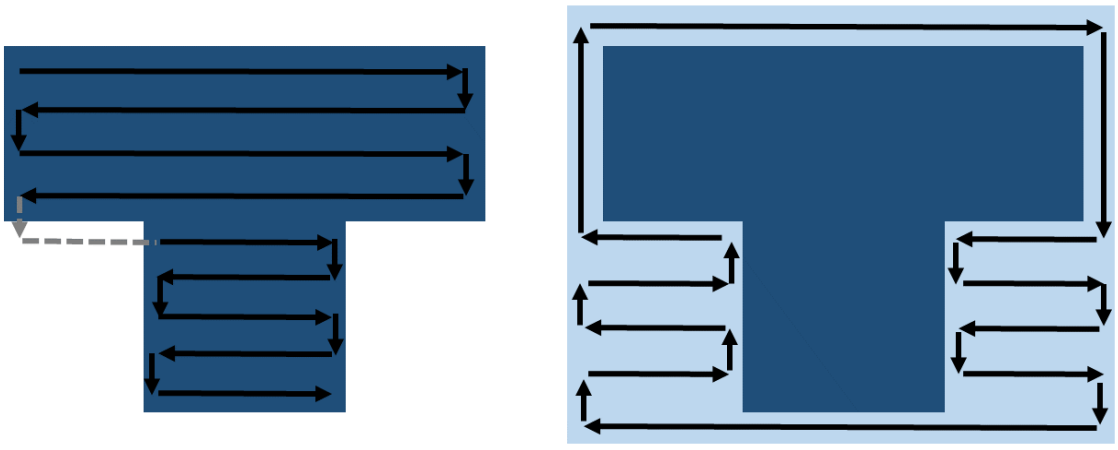


Figura 5.11 - Exemplo básico da diferença dos movimentos de uma máquina de corte e uma máquina de extrusão

Mais ainda, no caso da impressão, é também necessário o *software* CAM conhecer o tipo de material que a ferramenta imprime para ser definida, por exemplo, a temperatura de derretimento do termoplástico. Mas não só, também devem ser fornecidos outros parâmetros como a altura de cada camada de impressão, grossura de impressão, velocidade de impressão, e muito outros. Por outras palavras, o *software* de CAM necessita de conhecer para qual o tipo de ferramenta e material que a máquina usa, de modo a criar o ficheiro G-code adequado.

No entanto, no LinuxCNC o desencadeamento da impressão não se dá de forma tão simples como no caso de uma ferramenta de corte. Na verdade, para além dos comandos típicos de movimentação, o ficheiro G-code deve conter também outros comandos, denominados M-codes que permitem adicionar um conjunto de capacidades extra durante a maquinação.

Neste caso em particular, é através deste M-codes que é definida a temperatura de extrusão, e se dá a ativação do motor alimentador que inicia a criação do objeto. De modo mais prático, o que acontece é que o interpretador do LinuxCNC vai lendo linha a linha o código do objeto, quando encontra um M-code, procura na sua pasta de trabalho o ficheiro correspondente ao M-code invocado, que define os procedimentos que deve efetuar nesse momento.

Normalmente, o primeiro passo do ficheiro CAM é começar a aquecer o cilindro de aquecimento para quando for acionada a extrusão, a plástico se encontre na temperatura ideal para ser moldado. Neste caso, seria invocado quando é invocado o M-code correspondente à escrita da temperatura, o M105. Entretanto, o controlador RepRap fica responsável por garantir que essa temperatura é atingida e quando isso acontecer dá sinal ao LinuxCNC para avançar. Depois de mover a ferramenta para a posição onde se dá o início da impressão, o LinuxCNC encontrar outro M-code que envia um comando à placa RepRap para dar Início à extrusão, ou seja, o motor alimentador deve começar a mover-se.

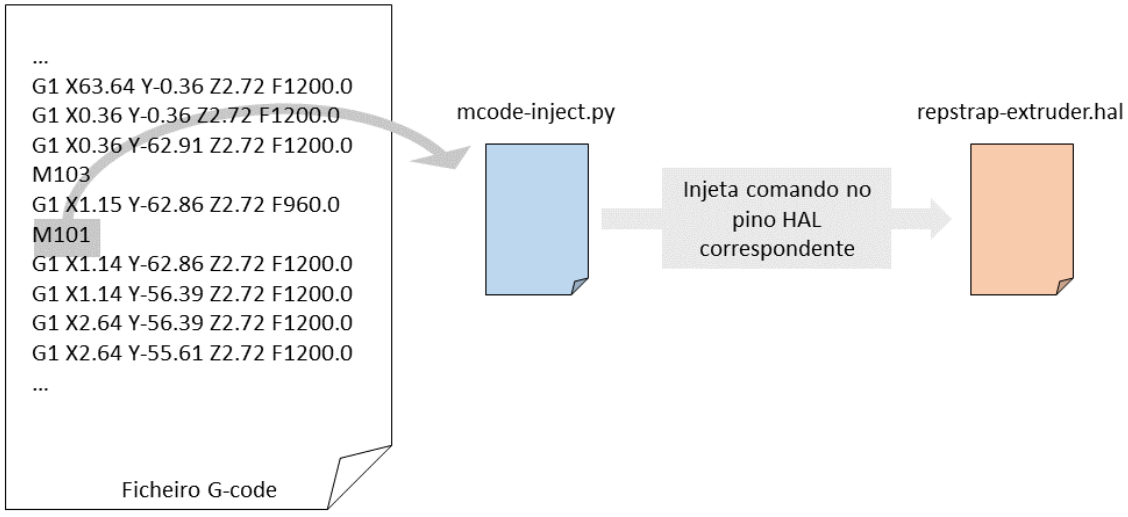


Figura 5.12 – Etapas do funcionamento dos M-codes na impressora 3D desenvolvida

Como demonstra a figura 5.12, sempre que é invocado um M-code, este chama um *script* implementado, denominado mcode-inject.py, cuja função é injetar no pino HAL correspondente o comando que se quer transmitir ao controlador de extrusão RepRap, através

do *script* repstrap-extruder.py e, consequentemente, pelo protocolo de comunicação implementado (ver figura 5.9).

5.3 - Testes finais realizados e validação

Em qualquer projeto de engenharia é essencial proceder-se a um conjunto de testes para validar o sistema desenvolvido. Neste caso em particular, os testes resumem-se a duas avaliações fundamentais: verificar se os eixos se movem conforme o pretendido e perceber se o sistema é mesmo capaz de controlar a extrusão através do LinuxCNC, de forma manual e automática. Na verdade, os testes realizados foram bastante simples, serviram apenas para verificar se o objetivo de dotar o LinuxCNC da capacidade de controlar uma impressora 3D tinha sido cumprido, sem grandes preocupações com a calibração final da máquina.

Cada eixo foi testado individualmente quando à correta movimentação, de acordo com o que é pedido pelo controlador. Inicialmente, o teste consistiu em mover o eixo num determinado sentido e avaliar se de facto este se movia a distância exigida pelo controlador. Para isso procedeu-se à atuação sobre os eixos de forma manual[1], através da interface AXIS.

Para o teste dos eixos, a primeira análise realizada foi descobrir a distância que o eixo linear se movia quando chega um passo aos terminais do motor. Como já foi exposto na secção 5.1.2.3, os três eixos partilham as mesmas características físicas, ou seja, partilham o menos número de passos por milímetro, que, obtido através da equação 5.3 é de 200 passos por milímetro. Isto significa que cada passo faz mover o eixo 0.005mm (=1/200), o que torna impraticável fazer ter uma avaliação super rigorosa sobre o movimento de cada eixo.

No entanto, na verdade, nunca foram estabelecidas metas em relação à precisão da máquina e, por essa razão, apenas se fizeram testes relativamente básicos para verificar se o sistema se movia corretamente. Com o auxílio de uma régua convencional, mediu-se a posição inicial da máquina e acionou-se o movimento manual no LinuxCNC para que cada um dos eixos se movesse 100mm. No final do movimento de cada um dos eixos, foi medida a sua nova posição e verificou-se que cada um dos eixos moveu a distância exigida pelo controlado. Quanto aos requisitos temporais, este foram verificado através da velocidade imprimida pelo controlador, ou seja, mediu-se o tempo que os eixos demoravam a percorre o trajeto de 100mm e calculou-se a velocidade para efeitos de comparação. Também neste caso se obteve sensivelmente os valores esperados para cada um dos eixos. Por outro lado, a aceleração não foi alvo de análise.

Numa segunda fase, o LinuxCNC foi carregado com um programa G-code, já existente na sua libraria, e voltou-se a testar se realmente os eixos efetuavam o movimento esperado, mas desta vez automaticamente. Os resultados obtidos foram igualmente satisfatórios, mas neste caso comprova-se que o controlador é de facto capaz de sincronizar perfeitamente os movimentos dos eixos para um correto posicionamento da ferramenta de trabalho.

[1] Por manual entende-se que não foi carregado nenhum ficheiro com objeto para imprimir, apenas foram utilizados os comandos da interface AXIS de movimentos no sentido positivo e negativo.

Uma vez mais se ressalva que através destes procedimentos foi feita uma avaliação um tanto ou pouco superficial pois o objetivo nesta fase era perceber apenas se o controlador era capaz de mover os eixos.

Para a validação do sistema quando à extrusão, inicialmente usou-se o painel desenvolvido e integrado na interfase AXIS, para o teste manual da ferramenta. O painel permite aplicar uma temperatura ao cilindro de aquecimento manualmente, até um limite de 300ºC, e, no mesmo painel é possível verificar a temperatura do cilindro a cada instante. Como já foi explicado no capítulo 4, não havia conhecimento sobre as propriedades do termístor usado para medir a temperatura e, por essa razão, a temperatura ambiente estava bastante desfasada da realidade – à temperatura ambiente o termístor registava 40ºC. Assim, nesta fase, apenas se avaliou se a temperatura definida no painel do LinuxCNC era posteriormente lida pelo termístor. Ou seja, na prática, se fosse definida uma temperatura de 200ºC, espera-se que o termístor quando ler 200ºC (ainda que incorretos) desligue automaticamente o fornecimento de energia ao cilindro aquecedor. Este prossuposto foi de facto verificado.

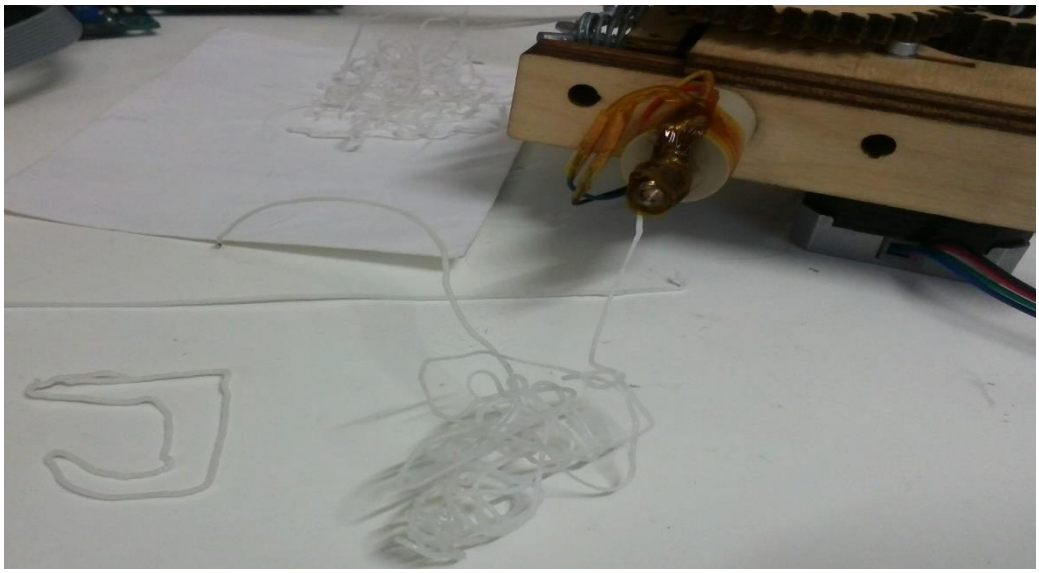


Figura 5.13 – Teste com extrusora a imprimir termoplástico

Por fim, para o teste e validação final, pretendia-se que o sistema se comportasse como uma impressora 3D funcional, em que apenas lhe é passado um ficheiro G-code e este automaticamente começa a imprimir a peça. Para isso, foi utilizado um ficheiro .stl descarregado da internet [83], com a forma de uma *Bitcoin*. Antes de mais, foi necessário converter este ficheiro .stl, num ficheiro interpretável pelo LinuxCNC, o G-code. Neste caso, foi utilizado o *software* CAM *Skeinforge*, bastantemente utlizado pala comunidade RepRap para a geração de G-code adaptado à impressão a três dimensões. Para a conversão do ficheiro é necessário configurar um conjunto enorme de parâmetros de impressão, no enquanto, foram usadas valores pré-definidas, isto porque, mais uma vez, apenas se pretende demonstrar o conceito, não se pretende uma calibração refinada da máquina. Após carregar o ficheiro para a interface do LinuxCNC e colocar a máquina em funcionamento, esta automaticamente

estabelece a temperatura e, quando o código exige que a extrusora comece a imprimir, o motor desse componente começa a rodar. Simultaneamente, os eixos movem-se de acordo com o que o trajeto visível na plataforma AXIS.

Para efeitos práticos, e tendo em conta o estabelecido e pretendido para esta fase do projeto, considera-se o sistema testado e validade.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste último capítulo são sumarizados os principais resultados obtidos na dissertação, sendo estes analisados face aos objetivos inicialmente propostos. Também serão descritas algumas perspetivas de trabalhos futuros para dar seguimento ao projeto apresentado.

6.1 - Conclusão

Analisados aos resultados obtidos na dissertação, uma vez terminado o projeto e, de acordo com os objetivos estabelecidos inicialmente, comprova-se que estes foram atingidos com êxito.

Através dos estudos e análises efetuadas ao longo do projeto foi possível perceber a extrema relevância do trabalho realizado na dissertação, não só pelo facto de envolver o desenvolvimento de uma impressora 3D - uma tecnologia muito em voga nos dias de hoje - mas também porque permitiu criar um modo de controlar esta impressora 3D através do LinuxCNC, que não está preparado de origem para controlar ferramentas aditivas. Destaca-se este último ponto devido ao reduzido número de projetos nesta área – e mesmo estes, num total que não ultrapassa a dezena, apresentam muito pouca informação documentada.

Recordando o que foi apresentado inicialmente, o principal objetivo da dissertação consistia em usar o *software* LinuxCNC para controlar uma plataforma funcional que operasse como uma impressora 3D. No final pretendia-se que um computador convencional, equipado com o controlador LinuxCNC, fosse capaz de coordenar os movimentos de um conjunto de três eixos e, simultaneamente, estar apto a gerir uma ferramenta de extrusão. Unidos, estes dois grupos formam a impressora 3D desenvolvida.

Na verdade, o objetivo principal foi dividido em *milestones* que foram alcançados e tornaram-se vitais para o sucesso do projeto. Assim, para assegurar o posicionamento dos eixos foi necessário proceder ao dimensionamento dos eixos físicos, dos motores, dos circuitos de condução e posterior configuração do LinuxCNC para que este controlasse os movimentos da impressora 3D pela porta paralela. Por outro lado, ao nível da extrusão, a impressão do termoplástico foi garantida através da modificação e adaptação de *firmware* do controlador

externo da extrusora. Do mesmo modo, também foi implementado um conjunto de *scripts* que permitiram a comunicação do LinuxCNC com o controlador da extrusora, através da porta série, sendo que, por fim, foi criado um painel virtual aliado à interface gráfica do controlador, que permite a monitorização e controlo da temperatura e da velocidade do motor da extrusora, diretamente a partir do LinuxCNC.

Para o *hardware* de interface do LinuxCNC com os motores e extrusora foi usada a eletrónica base do projeto RepRap. Esta provou ser uma excelente opção para a condução e controlo dos componentes da máquina, tal como seria de esperar pois foi desenhada para o controlo de impressoras 3D. Na verdade, o projeto prevê o uso de uma placa, denominada *motherboard*, responsável pelo controlo da impressora, no entanto, o LinuxCNC substituiu o seu papel na totalidade.

Por fim, perante os testes realizados e os resultados obtidos constata-se que os módulos em que o sistema se divide encontram-se a funcionar de acordo com o pretendido. Ao nível da movimentação, os três eixos cartesianos responsáveis pelos movimentos nas três dimensões respondem corretamente às ordens de posicionamento do controlador. Por sua vez, a extrusão mostra-se responsiva ao desencadear ações necessárias para a impressão do termoplástico. Para automatizar o processo, foi definido um conjunto de M-codes – comandos especiais colocados no meio do G-code, gerado pelo *software* CAM – que, ao serem invocados, davam origem ao depósito do plástico derretido na posição deseja.

Em suma, a figura 6.1 demonstra esquematicamente os principais componentes que constituem a impressora 3D desenvolvida. Note-se que se trata de uma versão contextualizada da figura 2.1, presente no início do capítulo 2 - Estado da arte, depois de encontrada a solução para o problema proposto.

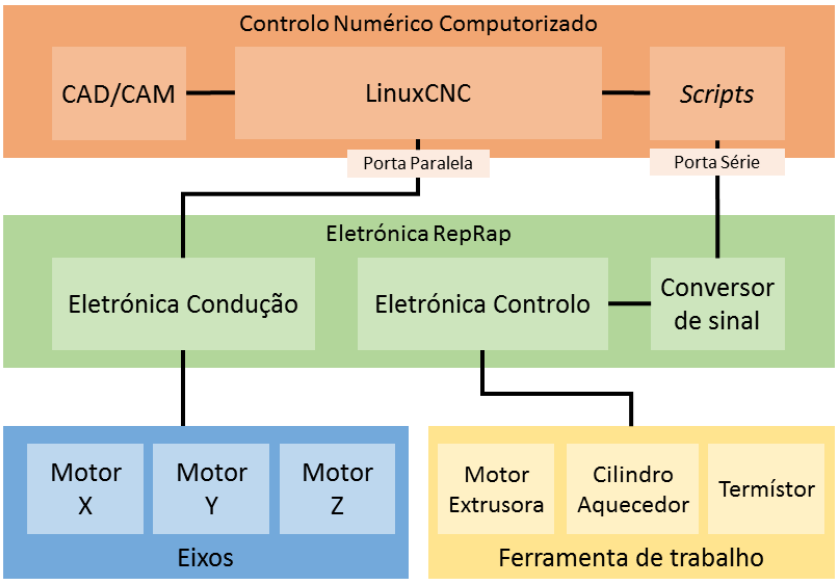


Figura 6.1 - Esquema dos componentes base da Impressora 3D desenvolvida.

6.2 - Desenvolvimentos futuros

O fim da dissertação não significa o fim deste projeto. Apesar dos objetivos definidos para a dissertação terem sido atingidos, futuros desenvolvimentos poderão levar o projeto a um novo patamar.

A criação de uma plataforma que funciona como impressora 3D, controlada através do LinuxCNC não é por si só um fim. Desde o início da dissertação ficou patente qual o objetivo do projeto, para lá do contexto da dissertação. A intenção é no futuro se integrar o compilador MatIEC – um compilador para linguagem de programação de PLC's definidas pelo IEC 61131-3, mais concretamente para as linguagens ST, IL e SFC - com LinuxCNC.

Assim, o presente documento, além de demonstrar o projeto elaborado nesta dissertação, pode servir como ponto de partida para a fase seguinte do projeto, nomeadamente para perceber como funciona o LinuxCNC, como está internamente organizado e como interage com a plataforma criada.

Num caráter mais imediato, o próximo passo será a montagem da máquina, que consiste disposição física dos eixos sobre a plataforma física de suporte e acoplamento da extrusora ao eixo Z, e a calibração final da mesma. No entanto, é importante salientar e sublinhar que a montagem da máquina não estava na ordem de trabalhos do projeto, por se tratar de um aspeto puramente técnico de assemblagem dos componentes que, repita-se, foram testados e validados individualmente. Os resultados obtidos mostram que, uma vez testados individualmente os módulos do sistema, a sua montagem final permitirá obter objetos impressos a partir do modelo interpretado pelo LinuxCNC. Para isso, serão apenas necessárias calibrações circunstanciais resultantes da disposição dos eixos e da extrusora face a estes.

Em suma, o trabalho desenvolvido na dissertação, apesar de concluído com sucesso, tem uma grande margem de progressão. Na sua grande maioria, este progresso está relacionado com a fase seguinte do projeto, fora dos objetivos da dissertação, mas que contribui para o objetivo final que se pretende alcançar.

Anexos

Anexo A – Estudo do modo de funcionamento dos motoros passo-a-passo

A.1 - Definição quanto à sua estrutura

Ao nível da sua estrutura os motores passo-a-passo podem ser classificados em três tipos, que operam de modo diferente:

- Íman permanente (PM);
- Relutância Variável (VR);
- Híbrido (H).

A.1.1 - Íman Permanente

Este tipo de motores têm na sua estrutura ímanes permanentes, sendo uma das suas características definidoras o facto de não possuir dentes. Tenreiro Machado, em 1994 [27], explica que o rotor é magnetizado alternadamente nos seus polos norte e sul o que resulta no movimento num determinado sentido. Quando a corrente passa por um dos enrolamentos, um polo do estator fica com polaridade Norte e o seu par fica com a polaridade inversa, fazendo o motor se fixar numa posição. Posteriormente, quando outro enrolamento ganha energia, e consequentemente o seu par fica com polaridade oposta, faz o motor mover-se à procurar de estabilidade magnética (figura A.1) [84]. São motores de baixo custo e de baixa resolução com ângulo de passos entre 6º e 45º [22].

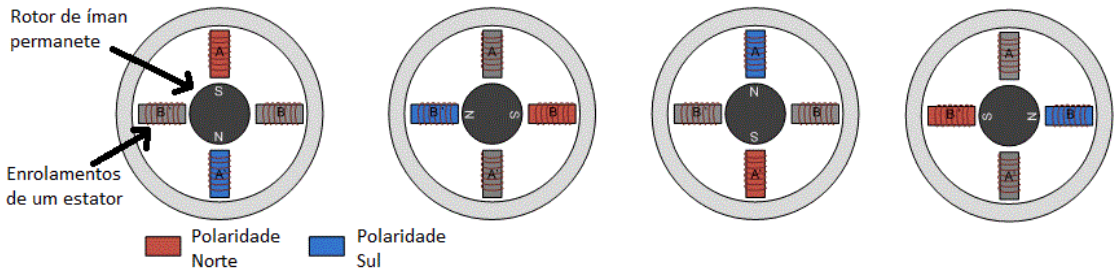


Figura A.1 - Motor passo-a-passo de ímanes permanentes [84]

A.1.2 - Relutância Variável

Ao contrário do motor de íman permanente, este tipo de motor contém dentes ao longo do rotor não-magnetizado, assim como o estator também apresenta dentes que encaixam nos dentes do rotor (figura A.2). O funcionamento deste é, tal como no caso anterior, baseado na energização dos enrolamentos do estator alternadamente, contudo, pela presença dos dentes, a força magnética faz o rotor rodar aos poucos, de modo a ficar alinhado com estator excitado, afirma Tenreiro Machado [27]. Tipicamente o ângulo de passo está entre 1.8° e 30°, tem um custo moderado e um design simples [22, 84].

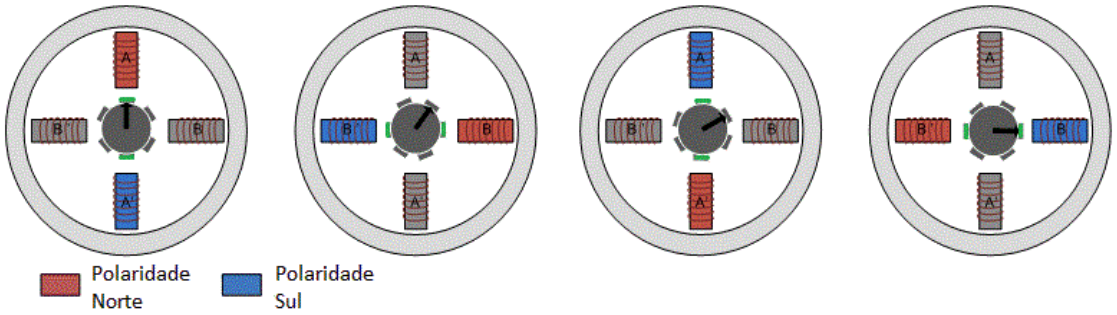


Figura A.2 - Motor passo-a-passo de relutância variável [84]

A.1.3 - Híbrido

Por fim, o motor híbrido apresenta uma combinação das alternativas anteriores ao conter um rotor magnético com dentes (figura A.3). O íman permanente colocado no centro tem dois polos, positivo e o negativo, com o mesmo número de dentes – que determina o número de passo por rotação completa – mas deslocados de modo que apareçam alternadamente aos olhos do estator. Do mesmo modo que no caso anterior, os dentes são colocados de maneira a que apenas encontre um local de equilíbrio, estando os outros dentes deslocados face à posição desejada [27, 84]. Quando o estator muda a sua polaridade os dentes do rotor com polaridade positiva movem-se de forma a ficarem alinhados com os dentes do estator com energia negativa. Com a junção destas duas ciências, este tipo de motor é capaz de operar com grande resolução, tendo um ângulo de passo entre 0.36° e 15°. No entanto tem um design complexo e são soluções caras [22].

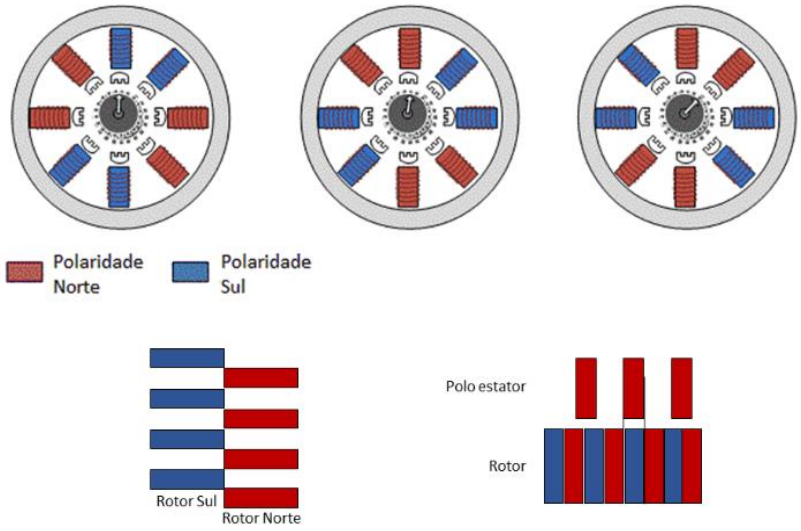


Figura A.3 - Motor passo-a-passo híbrido [84]

A.2 - Definição quanto ao modo de funcionamento

Para lá da sua classificação quanto a aspetos físicos, os motores passo-a-passo, dependendo das suas configurações, podem operar essencialmente de 4 modos diferentes que relacionado com o modo como os solenoides são excitados:

- *Full step (one-phase on);*
- *Full step (two-phases on);*
- *Half step;*
- *Microstep.*

A.2.1 - *Full step (one-phase on)*

Neste modo, apenas um enrolamento recebe energia de cada vez, por essa razão se dá o nome de *one-phane on*.

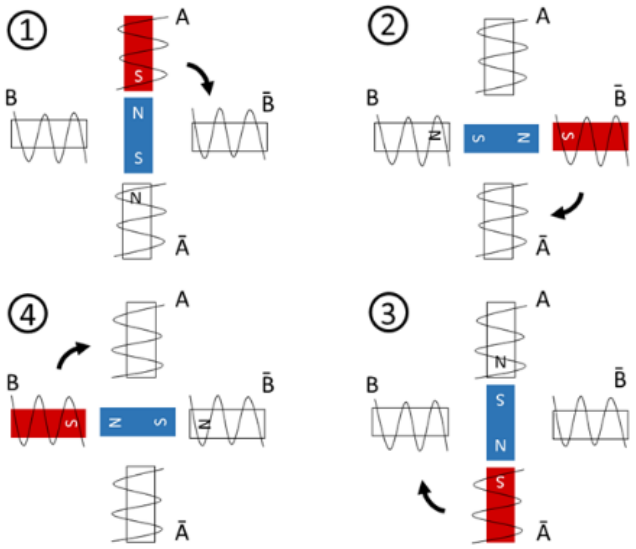


Figura A.4 - Funcionamento em *full step (one-phase on)*

A figura A.4 ilustra a sequência típica de um motor de 2 fases. No passo 1 a fase A é energizada, o que leva o rotor a ficar na magneticamente preso na sua direção, com os polos de energia opostas a encontrarem-se. No ponto 2 a fase A é desligada e a fase B é excitada fazendo com que o rotor gire de acordo com as leis da física. No passo 3 a energia volta ao enrolamento A, mas com a corrente a fluir de forma contrária ao que aconteceu em 1, levando a que o motor dê meia volta. Por fim, o ponto 4 é semelhante ao 3, mas para a fase B, e posteriormente volta ao ponto 1. Neste modo de funcionamento, apenas se tira partido de 25% dos enrolamentos na configuração unipolar e 50% na configuração bipolar, o que significa que o torque será, obrigatoriamente inferior à capacidade do motor tem de atingir [22, 23, 26, 27].

A.2.2 - *Full step (two-phases on)*

Por oposição ao *full step (one-phase on)*, e tal como o nome sugere, os dois enrolamentos recebem sempre energia. Contudo, o seu funcionamento não difere muito da versão anterior, tal como a figura A.5 demonstra. Em suma, o resultado final aparentemente é semelhante, o motor dá o mesmo número de passos para fazer uma rotação completa, com a mesma resolução. Todavia, ao ter os dois enrolamentos energizados, no modo bipolar o aproveitamento dos enrolamentos é total e mesmo no modo unipolar esse aproveitamento passa para o dobro, ou seja 50%, comparando com o modo *on-pahse on*. Este simples facto, segundo a literatura, leva a que se consiga um torque de cerca de 40% superior que na versão supracitada [22, 23, 26, 27].

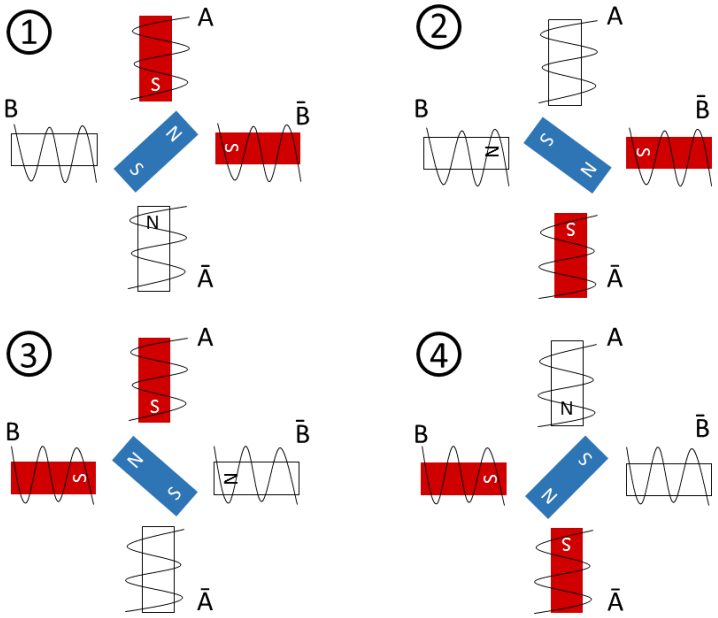


Figura A.5 - Funcionamento em *full step (one-phase on)*

A.2.3 – *Half Step*

Esta versão combina o modo de funcionamento das duas anteriores, ou seja, intercala a entre a excitação de um e dois enrolamentos. Naturalmente que o torque será menor que no caso anterior - na ordem dos 30% menor - já que por vezes apenas um enrolamento tem corrente, aliás, é mesmo necessária a utilização de um circuito de condução capaz de gerir a quantidade de corrente a passar nas bobines para que o torque se mantenha constante durante o funcionamento do motor (figura A.6). No entanto, esta é uma solução abundantemente utilizada pois permite que um motor passo-a-passo obtenha o dobro da resolução, sem alterar nenhuma aspeto relativo ao motor em si, o que permite movimentos mais suaves [22, 23, 26, 27].

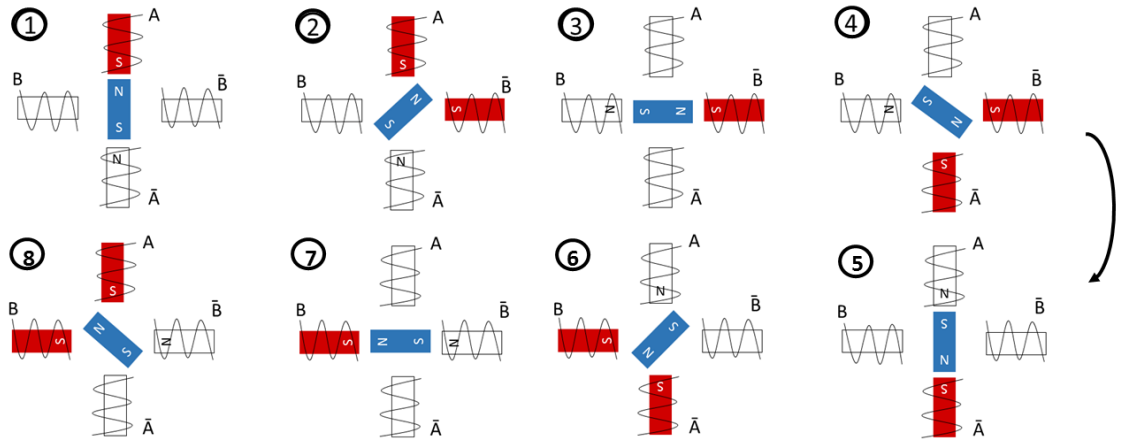


Figura A.6 - Funcionamento em *half step*

A.2.3 – *Microstep*

Finalmente, o modo de operação em *microstep* consiste em dividir, em intervalos ainda mais pequenos, a distância entre as duas fases. O princípio de funcionamento baseia-se em fornecer continuamente corrente elétrica aos enrolamentos mas, em vez de cada um destes estar completamente energizado, o circuito de condução divide a corrente pelos dois enrolamentos de forma pesada, resultando em pequenas divisões consoante essa mesma distribuição de energia (figura A.7).

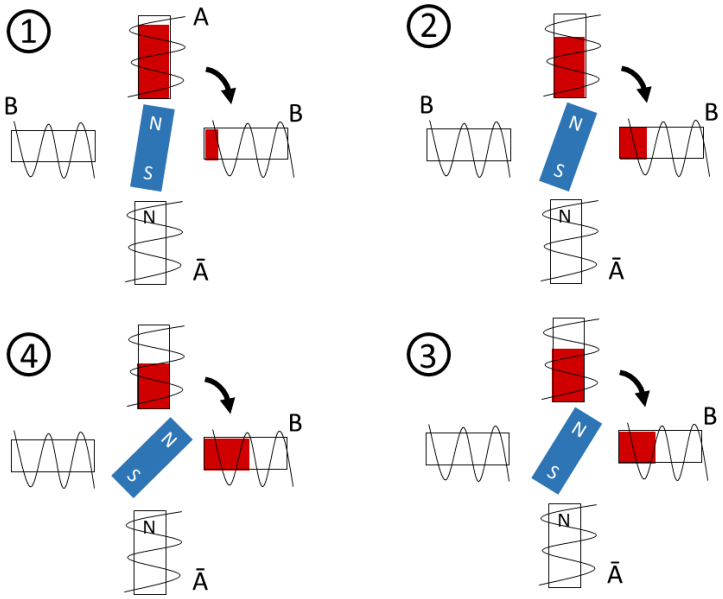


Figura A.7 - Funcionamento em *microstep*

Tal como na passagem de *full step* para *half step*, a resolução aumenta, neste caso em fatores de 8, 16, 32 ou 64, mas em contra partida o torque diminui consideravelmente. Contudo, é bastante usado quando um grande nível de precisão é essencial e, em contra partida, velocidade elevada não é um fator decisivo [22, 23, 26, 27].

Referências

[1] S. H. Suh, S. K. Kang, D. H. Chung, and I. Stroud, *Theory and Design of CNC Systems*. Springer, 2008, 1-10.

[2] Y. Koren, "Computers: Computer-based machine-tool control: The best place to inject a computer in numerical control is locally, not as a controller of many systems," *Spectrum, IEEE,* vol. 14, pp. 81-83, 1977.

[3] 3D Systems. *The Journey of a Lifetime*. Disponível em http://www.3dsystems.com/30-years-innovation. Acessado em Julho 2014.

[4] C. Mims. Quartz. *3D printing will explode in 2014, thanks to the expiration of key patents.* Disponível em http://qz.com/106483/3d-printing-will-explode-in-2014-thanks-to-the-expiration-of-key-patents/. Acessado em Agosto 2014.

[5] E. Ferreira, "Impressão 3D, revolução de equipa: Emanuel Ferreira at TEDxOporto," ed. Youtube: TEDx Talks, 2014.

[6] A. Council and M. Petch, *3D Printing: Rise of the Third Industrial Revolution*. Gyges 3D, 2014, capitulo 1.

[7] N. Gersbenfeld, "How to Make Almost Anything: The Digital Fabrication Revolution," *Foreign Affairs,* vol. 91, 2012.

[8] J. Moilanen and T. Vadén. Statistical Studies of Peer Production. *Another industrial revolution?* Disponível em http://surveys.peerproduction.net/2012/05/manufacturing-in-motion/. Acessado em Julho 2014.

[9] T. McCue. (2013) 3D Printing Stock Bubble? $10.8 Billion By 2021. *Forbes*. Disponível em: http://www.forbes.com/sites/tjmccue/2013/12/30/3d-printing-stock-bubble-10-8-billion-by-2021/

[10] The Economist. *A third industrial revolution.* Disponível em http://www.economist.com/node/21552901. Acessado em Agosto 2014.

[11] H. Kelly. CNN. *Study: At-home 3-D printing could save consumers 'thousands'*. Disponível em http://whatsnext.blogs.cnn.com/2013/07/31/study-at-home-3-d-printing-could-save-consumers-thousands/. Acessado em Agosto de 2014.

[12] Y. Yan, S. Li, R. Zhang, F. Lin, R. Wu, Q. Lu, *et al.*, "Rapid Prototyping and Manufacturing Technology: Principle, Representative Technics, Applications, and Development Trends," *Tsinghua Science and Technology,* vol. 14, pp. 1-12, 2009.

[13] C. K. Chua, K. F. Leong, and C. S. Lim, *Rapid Prototyping: Principles and Applications*. World Scientific, 2010, capítulo 2.

[14] T. Owad. (2014, 17 de Março) CNC Machining vs 3D Printing. *Make:* . Disponível em: http://makezine.com/magazine/make-ultimate-guide-to-3d-printing/cnc-machining-vs-3d-printing/

[15] C. R. Alavala, *CAD/CAM: Concepts and Applications*. PHI Learning, 2008, capítulo 1.

[16] F. M. Proctor and J. Micahloski, "Enhanced Machine Controller Architecture Overview," *Nation Institute os Standards and Technology*, 1993.

[17] LinuxCNC. Disponível em http://linuxcnc.org/. Acessado em Junho de 2014.

[18] B. Warfield. CNCCookbook. *CNC Control Market Shares: What Are the Most Popular Controls?* Disponível em http://blog.cnccookbook.com/2012/08/02/cnc-control-market-shares-what-are-the-most-popular-controls/. Acessado em Junho 2014.

[19] Mach3. Disponível em http://www.machsupport.com/software/mach3/. Acessado em Junho 2014.

[20] ReplicatorG. *ReplicatorG is a simple, open source 3D printing program.* Disponível em http://replicat.org. Acessado em Julho 2014.

[21] RepRap. *RepRap Options.* Disponível em http://reprap.org/wiki/RepRap_Options. Acessado em Julho 2014.

[22] H. D. Stoelting, E. Kallenbach, and W. Amrhein, *Handbook of Fractional-Horsepower Drives*. Springer, 2008.

[23] A. Hughes, *Electric Motors and Drives,* in *Fundamentals, Types and Applications*, Newnes, Ed., 3 ed Oxford, 1990, pp. 305-335.

[24] J. W. Valvano. Stepper motor and driver selection [Online]. Disponível em: http://users.ece.utexas.edu/~valvano/Datasheets/StepperSelection.pdf.

[25] J. F. Kelly, *3D Printing: Build Your Own 3D Printer and Print Your Own 3D Objects*. Que, 2013, 10-11.

[26] Solarbotics. Stepper Motor Basics [Online]. Disponível em: http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf.

[27] A. J. Tenreiro Machado, *Motores passo a passo: controlo e modos de funcionamento*. Porto. Publindústria, 1994.

[28] *Hybrid-Type Stepping Motors : Expansion in Line-Up of Small-Diameter and Composite-Type Models.* Disponível em https://www.minebea.co.jp/english/press/2012/1186854_5482.html. Acessado em Junho de 2014.

[29] J. W. Douglas. *Control of Stepping Motors.* Disponível em http://homepage.cs.uiowa.edu/~jones/step/. Acessado em Junho de 2014.

[30] Nation Instruments. *Motor Fundamentals.* Disponível em http://www.ni.com/white-paper/3656/en/. Acessado em Junho 2014.

[31] Techno Inc. Choosing Between Stepper and Servo Motors [Online]. Disponível em: http://www.techno-isel.com/Tic/H834/PDF/H834P041.pdf.

[32] F. G. Brites and V. P. d. A. Santos. Universidade Federal Fluminense. Motor de Passo [Online].

[33] RepRap. *Category: Mechanical arrangement*. Disponível em http://reprap.org/wiki/Category:Mechanical_arrangement. Acessado em Julho 2014.

[34] Joshua. CNC 4 everyone. *Linear Drives*. Disponível em http://www.cnc4everyone.com/hardware/linear-drives/. Acessado em Julho 2014.

[35] R. Jones, P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer*, et al.*, "RepRap – the replicating rapid prototyper," *Robotica,* vol. 29, pp. 177-191, 2011.

[36] RepRap. Disponível em http://reprap.org/. Acessado em Julho 2014.

[37] B. T. Wittbrodt, A. G. Glover, J. Laureto, G. C. Anzalone, D. Oppliger, J. L. Irwin*, et al.*, "Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers," *Mechatronics,* vol. 23, pp. 713-726, 9// 2013.

[38] (2012, Novembro) A paradigm shift. *Industrial Engineer*. 12.

[39] E. Gilloz. RepRap. *RepRap Family Tree*. Disponível em http://reprap.org/mediawiki/images/e/ec/RFT_timeline2006-2012.png [Imagem]. Acessado em Junho 2014.

[40] D. Culler, N. Anderson, and S. Ames, "Design and construction of a rapid prototyping machine: A breakdown of the machine sub-systems used to learn multi-disciplinary engineering skills," in *ASEE Annual Conference and Exposition, Conference Proceedings*, 2009.

[41] MakerBot. [Online]. Disponível em: http://downloads.makerbot.com/support/pdf/Cupcake/Docs/Cupcake%20Build%20Sequence%20-%20Batches%201-9.pdf.

[42] B. Evans, *Practical 3D Printers: The Science and Art of 3D Printing*. Apress, 2012, 7-18.

[43] RepRap. *Motherboard 1.2*. Disponível em http://reprap.org/wiki/Motherboard_1.2. Acessado em Junho 2014.

[44] RepRap. *Stepper Motor Driver 2.3*. Disponível em http://reprap.org/wiki/Stepper_Motor_Driver_2.3. Acessado em Junho 2014.

[45] Allegro. A3982: DMOS Stepper Motor Driver with Translator [Online]. Disponível em: http://www.allegromicro.com/en/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A3982.aspx.

[46] RepRap. *OptoEndstop 2.1*. Disponível em http://reprap.org/wiki/OptoEndstop_2.1. Acessado em Junho 2014.

[47] RepRap. *Extruder Controller*. Disponível em http://reprap.org/wiki/Extruder_Controller_2_2. Acessado em Junho 2014.

[48] RepRap. *Comparison of Electronics*. Disponível em http://reprap.org/wiki/Comparison_of_Electronics. Acessado em Junho 2014.

[49] J. Heathcote, "Improving the Makerbot 3D Printer," Project Report, School of Computer Science, University of Manchester, 2012.

[50] fab@home. *The Project*. Disponível em http://www.fabathome.org/index.php?q=node/2. Acessado em Julho 2014.

[51] RepRap. *EMCRepRap*. Disponível em http://reprap.org/wiki/EMCRepRap. Acessado em Agosto 2014.

[52] Y. Koren, "Open-Architecture Controllers for Manufacturing Systems."

[53] A. Gupta and V. U. Srinivasa, "Black Boc fo Machine Tools; Based on Open CNC Architecture Control Systems," *Control System an Instrumentation,* vol. 03, 2012.

[54] G. Pritschow, Y. Altintas, F. Jovane, Y. Koren, M. Mitsuishi, S. Takata*, et al.*, "Open Controller Architecture – Past, Present and Future," *CIRP Annals - Manufacturing Technology,* vol. 50, pp. 463-470, 2001.

[55] O. T. Oshiro, "A parallel control architecture to ultra-precision machine tool," Douturamento, Escola de Engenharia de São Carlos, Universidade de São Paulo, Brasil, 1998.

[56] G. E. Fisher, "Rapid system prototyping in an open system environment," in *Rapid System Prototyping, 1994. Shortening the Path from Specification to Prototype. Proceedings., Fifth International Workshop on*, 1994, pp. 213-219.

[57] T. Staroveški, D. Brezak, and T. Udiljak, "LinuxCNC – the enhanced machine controller: application and an overview," *Technical Gazette,* vol. 20, 2013.

[58] LinuxCNC. User Manual V2.5 [Online]. Disponível em: http://linuxcnc.org/docs/2.5/pdf/LinuxCNC_User_Manual.pdf.

[59] W. P. Shackleford, F. M. Proctor, and J. Michaloski. National Institute of Standards and Technology. The Neutral Message Language: A Model and Method for Message Passing in Heterogeneous Environments [Online].

[60] LinuxCNC. HAL Manual V2.5 [Online]. Disponível em: http://linuxcnc.org/docs/2.5/pdf/LinuxCNC_HAL_Manual.pdf.

[61] LinuxCNC. *EMC Components*. Disponível em http://wiki.linuxcnc.org/cgi-bin/wiki.pl?EMC_Components. Acessado em Julho de 2014.

[62] LinuxCNC. Getting Started V2.5 [Online]. Disponível em: http://linuxcnc.org/docs/2.5/pdf/LinuxCNC_Getting_Started.pdf.

[63] B. Rison. FSM Labs, Inc. Getting Started with RTLinux [Online].

[64] V. Yodaiken. 5th Linux Conference Proceedings. The RTLinux Manifesto [Online].

[65] A. Greensted. The Lab Book Pages. Disponível em http://www.labbookpages.co.uk/electronics/parallelPort.html. Acessado em Junho 2014.

[66] LinuxCNC. Integrator Manual V2.5 [Online]. Disponível em: http://linuxcnc.org/docs/2.5/pdf/LinuxCNC_Integrator_Manual.pdf.

[67] R. Moheimani, *Mechatronic Systems 2004.* Elsevier Science & Technology Books, 2005.

[68] Orientalmotor. *Motor Sizig Tools.* Disponível em http://orientalmotor.com/support/motor-sizing.html. Acessado em Julho de 2014.

[69] P. D. W. TilaKaratna, B. Shirinzadeh, and G. Aliei, "Model development and system identification of a cartesian manipulator using a laser-interferometry based measurement system," *Mechatronic Systems,* vol. 1, 2004.

[70] Igus. DryLin Driver engineering [Online]. Disponível em: http://www.igus.pt/_Product_Files/Download/pdf/66_gl2_e_dryl_sht_rz.pdf.

[71] R. Miklosovic. Enginnering Basics [Online]. Disponível em: http://academic.csuohio.edu/cact/frank/eec440/.

[72] Parker. System Calculations [Online]. Disponível em: http://www.compumotor.com/catalog/cataloga/a60-a62.pdf.

[73] Nanotec. Stepping Motor - ST4118D1804 [Online]. Disponível em: http://en.nanotec.com/fileadmin/files/Datenblaetter/Schrittmotoren/ST4118/D/ST4118D1804.pdf.

[74] Igus. stepper motor [Online]. Disponível em: http://www.igus.eu/_wpck/pdf/global/Motordatasheet_EN.pdf.

[75] MSI Tec. Move Profile [Online]. Disponível em: http://www.msitec.com/pdf/move_profile.pdf.

[76] ABB. The Motor Guide [Online]. Disponível em: http://www04.abb.com/global/seitp/seitp202.nsf/0/12c3580f179a9d58c125761f0057ca5c/$file/motor+guide+gb+02_2005.pdf.

[77] Nanotec. Nanotec. *ST4118 – Stepper Motor - Nema 17.* Disponível em http://en.nanotec.com/products/250-st4118-schrittmotor-nema-17/. Acessado em Julho 2014.

[78] RepRap. *Thermistor*. Disponível em http://reprap.org/wiki/Thermistor. Acessado em Julho 2014.

[79] G. Vdwalle. LinuxCNC. *Parallel Port Tester*. Disponível em http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Parallel_Port_Tester. Acessado em Abril 2014.

[80] Texas Instruments. MAX232, MAX232I - Dual EIA-232 Drivers/Receivers [Online]. Disponível em: http://www.ti.com/lit/ds/symlink/max232.pdf.

[81] S. Wong. GitHub. hrepstrap [Online]. Disponível em: https://github.com/sam0737/hrepstrap.

[82] J. Casainho. Project Hosting (Google Developers). casainho-projects [Online]. Disponível em: https://code.google.com/p/casainho-projects/source/browse/trunk/emcrepstrap/.

[83] J. Casainho. MakerBot Thingiverse. Bitcoin for keychain [Online]. Disponível em: http://www.thingiverse.com/thing:29275.

[84] N. Agnihotri. Engineers Garage. *Stepper Motors or Step Motors*. Disponível em http://www.engineersgarage.com/articles/stepper-motors. Acessado em Junho de 2014.