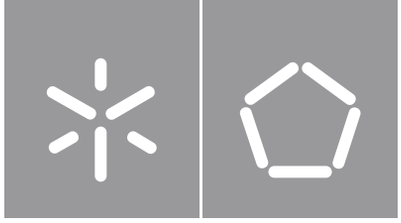
Universidade do Minho
Escola de Engenharia

Francisca Carolina Vigo Pereira Pinto

Conceção de uma Interface Homem-Máquina para uma máquina CNC

fevereiro de 2021

Conceção de uma Interface Homem-Máquina para uma máquina CNC
Francisca Vigo
UMinho | 2021



Universidade do Minho
Escola de Engenharia

Francisca Carolina Vigo Pereira Pinto

Conceção de uma Interface Homem-Máquina para uma máquina CNC

Dissertação de Mestrado em Engenharia Mecânica

Trabalho efetuado sob a orientação de
Professor Doutor Eurico Augusto Rodrigues de Seabra
Professor Doutor Luís Fernando Sousa Ferreira da Silva

fevereiro de 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Agradecimentos

Ao Professor Eurico Seabra e Professor Luís Ferreira da Silva, orientadores da minha dissertação, por todo o tempo e trabalho que investiram no meu projeto.

Um especial agradecimento ao Sr. Carlos Quelhas que me aceitou, de bom grado, para estagiar na sua empresa. A toda a equipa de trabalho na empresa Padrão Ortopédico, em especial ao Pedro, ao Ricardo e à Susana.

Agradeço ao meu orientador da empresa, engenheiro Joel Gomes (Joélio), por me aturar em todos os momentos que eu fiquei sem projeto por achar que era impossível e por me ter feito ver que eu conseguia. Ao Ivo, o meu porto seguro e melhor amigo que me acompanhou neste percurso com a máquina que tanto deu que fazer.

Claro que, nada seria possível sem a minha família, por isso, o maior "obrigada" aos meus pais, avó Mila, Beatriz e padrinho que, não só tornaram esta dissertação possível, como também todo o meu percurso académico.

Gostaria ainda de agradecer às pessoas que, direta ou indiretamente, ajudaram-me a chegar a este ponto da minha vida e a terminar este trabalho. Ao Zé e à família, obrigada por sempre me acolherem e me ouvirem. À Inês Estudante por estar sempre disponível quando precisei de ajuda. Aos meus tios, Sabina e Luís, por me hospedarem sempre que precisei de trabalhar na universidade.

Por fim, agradeço a todas as pessoas que me ajudaram ou se interessaram no meu projeto e me deram forças para o terminar.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

As próteses e ortóteses emergiram da área de terapia física. Enquanto prótese é um dispositivo aplicado externamente, usado para substituir total ou parcialmente um segmento de membro, a ortótese é um dispositivo usado para modificar as características estruturais e funcionais dos sistemas neuromuscular e esquelético. Uma das ortóteses existentes na área são os capacetes para tratamento de plagiocefalia posicional. Para a realização destes capacetes é necessária a tiragem do molde do crânio do paciente e produzir uma réplica, ou seja, um molde positivo do membro. Para isso, a empresa Padrão ortopédico utiliza uma série de etapas recorrendo a uma máquina CNC fabricada na própria empresa para maquinagem do molde.

A máquina CNC da Padrão Ortopédico padece de dois eixos principais, Z e X, sendo Z o eixo da ferramenta de corte (horizontal) e o X o eixo de movimento vertical. O bloco de poliuretano de baixa densidade é colocado num prato segundo o eixo rotativo A da CNC e o controlo da máquina seria promovido através do *software* "Grbl Controller" inserido num computador portátil ligado ao controlador Arduino da máquina.

Para o *design* da interface gráfica, o *software* previamente utilizado serviu como base para definir as funções necessárias a implementar como, "Envio de código e Maquinagem", "Manipulação Manual dos eixos", "Alteração das definições e parâmetros da Máquina" e "Comunicação Homem-Máquina".

Estas funções foram programadas na linguagem "Python3", com a utilização das bibliotecas *"*Tkinter*"* e *Pyserial*. O programa foi, por fim, implementado num *Raspberry* Pi4 (modelo b) como respetivo ecrã tátil de sete polegadas.

Para calibração da máquina foram ainda calculados os steps/mm de cada eixo, ajustados, através da nova interface, na CNC e realizados vários produtos de teste da fiabilidade do Sistema Interface–Máquina. Foi ainda, incorporado um fim de curso no eixo Z para garantir o *homing* no movimento horizontal da máquina.

Por fim, foi criado um livro de utilização da máquina que contempla asdimensões principais da mesma, os parâmetros ajustáveis e como proceder a uma maquinagem, passo a passo.

PALAVRAS-CHAVE: Molde, CNC, Interface, *Grbl*, *Python*.

Abstract

Prostheses and orthoses emerged from the area of physical therapy. While prosthesis is an externally applied device, used to replace part or all of a limb segment, orthosis is a device used to modify the structural and functional characteristics of the neuromuscular and skeletal systems. One of the existing orthoses in the area are helmets for the treatment of plagiocephaly. To make these helmets, it is necessary to remove the mold from the patient's skull and produce a replica, i.e., a positive mold of the limb. For this, the company Padrão Ortopédico uses a series of steps using a CNC machine manufactured in the company itself for machining the mold.

The CNC machine of Padrão Ortopédico has two main axes, Z and X, with Z being the cutting tool axis (horizontal) and X being the vertical movement axis. The low density polyurethane block is placed on a rotary plate along the secondary axis A of the CNC and "control" of the machine is promoted through the "Grbl Controller" software inserted in a portable computer connected to the machine's Arduino.

For the design of the graphical interface, the previously used software served as a basis for defining the necessary functions to be implemented, such as "Sending code and machining", "Manual manipulation of the axes", "Changing the settings and parameters of the Machine" and "Communication Machine man".

These functions were programmed in the Python3 language, using the *"Tkinter"* and Pyserial libraries. The program was finally implemented on a Raspberry Pi4 (model b) as its seven-inch touchscreen.

In order to calibrate the machine, the steps / mm of each axis were also calculated, adjusted, through the new interface, at the CNC and several products were tested for the reliability of the Interface – Machine System.

A limit switch on the Z axis was also incorporated to ensure homing in the horizontal movement of the machine.

Finally, a machine's manual was created that includes the main dimensions of the machine, the adjustable parameters and how to proceed with machining, step by step.

KEYWORDS: Mold, CNC, Interface, Grbl, Python.

Índice

Lista de figuras

Lista de tabelas

Lista de Equações

Lista de abreviaturas, siglas e acrónimos

Abreviatura	Significado
3D	Tridimensional
CAD	*"Computer Aided Design"*, Design assistido por computador
CAM	*"Computer-Aided Manufacturing"*, Manufatura assistida por computador
CNC	*"Computer Numerical "Control""*, Controlo Numérico Computorizado
CPU	*"Central Processing Unit"*, Unidade de processamento central
EEPROM	*"Electrically-Erasable Programmable Read-Only Memory"*, Memória programável para ler
FDM	*"Fused Deposition Modeling"*, Modelação por deposição
Gb	Gigabyte
GPU	*"Graphics Processing Unit"*, Unidade de processamento gráfico
GUI	*"Graphical User Interface"*, interface gráfica do utilizador
HDMI	*"High-Definition Multimedia Interface"*, interface de multimédia de alta definição
IDE	*"Integrated Development Environment"*, Ambiente de desenvolvimento integrado
LCD	*"Liquid Crystal Dislay"*, ecrã de cristal liquido
LED	*"Light-Emitting Diode"*, Díodo emissor de luz
MMI	*"Man Machine Interface"*, Interface homem-máquina
MVC	*""Model" – "View" – "Control""*, Modelo – visualizador - Controlo
NCK	*"Numerical "Control" Kernel"*, Controlo numérico Kernel
P.O.	Padrão Ortopédico
PLA	*"Polylactic Acid"*, Poliácido Lático
PLC	*"Power Line Communication"*, Comunicação via rede elétrica
RAM	*"Random Access Memory"*, Memória de acesso aleatória
Rx	*"Receive"*, Recetor
SD	*"Secure Digital"*, Digital seguro
TFT	*"Thin film transistor"*, Transístor de pelicula fina
Tx	*"Transmit"*, Transmissor

1. Introdução

Esta dissertação assenta no estudo, projeção e desenvolvimento de uma Interface Homem-Máquina para uma máquina CNC existente na empresa do estágio curricular Padrão Ortopédico do grupo Quelhas, Ribeiro & Cardoso Lda. No decorrer dos capítulos, será estabelecido como objetivo o estudo da CNC existente como base da projeção da interface, a determinação do equipamento físico a implementar, o *design* visual do *software* e a sua conceção em termos de programação.

Posto isto, desta dissertação resulta o equipamento final para que o operador da CNC da Padrão Ortopédico possa comandar a máquina de uma forma mais fácil e com menor probabilidade de ocorrência de erros.

1.1. Enquadramento

As máquinas CNC de fresagem são um utensílio recorrente nas grandes industrias de próteses e ortóteses sendo uma necessidade constante neste meio a produção de moldes positivos dos membros dos utentes. A máquina desenvolvida na empresa Padrão ortopédico está equipada com dois sistemas de eixos principais e um rotativo, sem capacidade para ser controlada na raiz, mas sim, através de um computador portátil com um *software open-source*.

Os técnicos da empresa carecem de conhecimento em órgãos de máquinas e controlo de máquinas industriais, sendo um entrave para os mesmos utilizarem programas complexos de envio de código para a CNC, podendo pôr em risco a veracidade da peça obtida, o estado da máquina ou o próprio técnico.

Assim, o pretendido é equipar a máquina com uma interface homem-máquina amigável ao utilizador, simples e intuitiva sem nunca pôr em causa o tempo, a precisão e exatidão do trabalho.

1.2. Objetivo

Genericamente, esta dissertação incide na conceção de uma interface Homem-Máquina para a CNC da empresa Padrão Ortopédico facilmente manuseada por qualquer técnico da empresa (sem qualquer tipo de formação em máquinas industriais). Sendo que, inicialmente, não existe qualquer informação sobre a máquina em si, é necessária a aquisição de dados físicos (mecânicos e eletrónicos) e dados no âmbito do *software* utilizado para traduzir e controlar os movimentos da CNC. Após a recolha desta informação,

é possível a caracterização das limitações da mesma e, assim, as especificações necessárias para a idealização da base da interface.

Ao basear a nova interface a partir de *softwares* prévios, sem a necessidade de adquirir um controlador existente no mercado que, reivindica a compra de sistemas mecânicos compatíveis com o mesmo, é possível obedecer a um dos objetivos para com a empresa que é o baixo custo de conceção, evitando também a remodelação de alguns equipamentos já adquiridos para a máquina.

É importante referir que este trabalho não está centrado no estudo da máquina e no seu controlador, mas sim na possibilidade de comunicar com o mesmo sem necessidade de enviar códigos que não são intuitivamente visíveis para um operador inexperiente.

1.3. Estrutura da dissertação

A dissertação divide-se em cinco capítulos expressos ordenadamente segundo a execução de cada etapa do trabalho. Obtendo-se a distribuição a baixo descrita.

- No **Capítulo 1**, é expressa a introdução ao tema, onde se promove o enquadramento do tema e a descrição dos objetivos.
- No **Capítulo 2**, abordam-se os temas teóricos para contextualizar a leitura da dissertação. O leitor poderá usufruir de uma breve revisão bibliográfica sobre o trabalho realizado na empresa Padrão Ortopédico para a concessão de moldes positivos para a manufatura de próteses e ortóteses, bem como o ponto de situação das máquinas CNC nesta área da ortopedia.
- No **Capítulo 3**, é realizado um estudo da máquina CNC fabricada na Padrão ortopédico, desde o estudo os componentes mecânicos e dos eixos de transmissão de movimento, até à execução detalhada de um molde positivo de um membro inferior na própria máquina. Por fim, é possível contemplar uma lista de limitações da CNC e algumas propostas de melhoria para essas mesmas limitações.
- No **Capítulo 4**, dá-se início ao Design, programação e conceção da interface homem-máquina. Começando pela aquisição dos componentes físicos a implementar; das funções que o operador espera encontrar, seguindo-se do posicionamento gráfico das mesmas e, por fim, a programação do *software*. Após a descrição destes tópicos, são ainda especificadas algumas alterações realizadas na CNC e nos seus componentes para que, seja possível a implementação e verificação do equipamento concebido no final do capítulo.

- Posteriormente a todos os tópicos anteriormente enunciados, no **capítulo 5**, estão expressas as conclusões retiradas de todo o estudo, projeto e conceção realizados, bem como a definição de melhorias e trabalhos futuros.

2. Produção de moldes positivos no fabrico de próteses e ortóteses e máquinas CNC

Este capítulo dá início ao projeto adjacente a toda a dissertação. Foram esmiuçados os conceitos teóricos que permitem a compreensão do projeto e as bases onde este assenta.

As próteses e ortóteses emergiram da área de terapia física. O desenvolvimento nestes dois campos foi consequência de três eventos históricos, nomeadamente, a primeira guerra mundial, a segunda guerra mundial e a epidemia de pólio (Lusardi, 1994).

Antes e aquando a primeira guerra mundial, a produção de uma prótese era realizada por ferreiros e soldadores, sendo estas constituídas principalmente de madeira e metal. Já na segunda guerra mundial, profissionais de terapia física, próteses e ortóteses eram aliados dos médicos, aumentando assim o desenvolvimento científico na área (Lusardi, 1994).

Após a epidemia do pólio, com a gravidade e variedade de casos de paralisia, iniciou-se um estudo aprofundado, considerando campos como distrofia muscular e desarticulações, reabilitação física, recuperação cirúrgica e paralisia cerebral (Lusardi, 1994).

Hoje em dia, cada tipo de prótese e ortótese é fabricada de uma forma diferente e utilizando técnicas e materiais consoante a necessidade do paciente, sendo o objetivo principal o seu bem-estar (Lusardi, 1994).

A prótese é um dispositivo aplicado externamente, usado para substituir total ou parcialmente um segmento de membro ausente ou deficiente. Exemplos comuns são as próteses transtibiais e transfemorais para o membro inferior e as próteses transradial ou transumeral para o membro superior (Healy et al., 2018).

A ortótese é, por sua vez, um dispositivo aplicado externamente, usado para modificar as características estruturais e funcionais dos sistemas neuromuscular e esquelético. Exemplos comuns são suportes para tornozelo, talas de pulso, ortóteses de pé e tornozelo, calçados personalizados (Lusardi, 1994).

Na tabela 2.1 e 2.2 é possível observar os tipos gerais de próteses e ortóteses existentes.

Tabela 2.1 - Ortóteses e próteses dos membros inferiores (Quintero-Quiroz, 2017)

Próteses	AK	Próteses acima do joelho
	BK	Próteses abaixo do joelho
	TT	Próteses transtibiais
	TF	Próteses transfemorais
Ortóteses	FO	Ortóteses de pé

Tabela 2.1 - Ortóteses e próteses dos membros inferiores (cont.) (Quintero-Quiroz, 2017)

Ortóteses	AFO	Ortóteses de tornozelo
	KO	Ortóteses de joelho
	KAFO	Ortóteses joelho-tornozelo-pé
	HpO	Ortóteses de anca
	HKO	Ortóteses anca-joelho
	HKAFO	Ortóteses anca-joelho-pé
	SIO	Ortóteses sacroilíacas

Tabela 2.2 - Próteses e ortóteses dos membros superiores. (Jin et al., 2015; Zenie, 2016)

Próteses	Próteses de desarticulação do ombro
	Próteses trans-humerais
	Próteses trans-humerais curtas
	Próteses trans-humerais longas
	Próteses desarticulação do cotovelo
	Próteses transradiais
	Próteses transradiais curtas
	Próteses transradiais longas
	Próteses de desarticulação do pulso
	Próteses transcarpais
	Próteses falangeais
Ortóteses	Ortóteses crânio-cervicais
	Ortóteses mão-pulso
	Ortóteses do ombro
	Ortóteses do cotovelo
	Ortóteses cervicais-torácicas
	Ortóteses lombares

2.1. Capacetes para tratamento de plagiocefalia posicional

Entende-se por plagiocefalia a assimetria do perímetro craniano, habitualmente com planificação occipital simétrica ou assimétrica, designada por "plagiocefalia posicional ou postural" (PP). A cabeça pode adotar a configuração de paralelogramo ou trapezoidal (Aguiar, n.d.).

O crânio de um lactente tem uma grande plasticidade, é pouco rígido, pelo que não é difícil perceber o quanto é sensível às forças externas de pressão (Aguiar, n.d.).

Durante a travessia do canal de parto, o crânio de um recém-nascido sofre deformações significativas transitórias, representadas na figura 2.1, tornando, assim, possível a sua travessia. Independentemente disso, outras forças externas, pós-natais ou até pré-natais, podem causar deformação do crânio, não tão transitórias (Aguiar, n.d.).

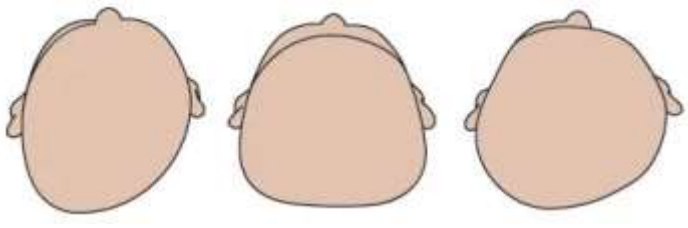


Figura 2.1 - Exemplificação da condição plagiocefalia posicional (Neves, 2015).

A utilização de capacetes está indicada apenas para casos graves, com deformidades superiores a 20 mm, avaliadas com um craniómetro, instrumento com o qual se medem os diâmetros do crânio. A sua utilização remonta a 1979, mas, após um período de quase esquecimento e não-aceitação, foi de novo adotado (Aguiar, n.d.).

Trata-se de uma ortótese craniana dinâmica. O capacete vai agir com uma contraposição de forças. Realiza um apoio nas áreas proeminentes, contendo o seu crescimento, deixando as áreas achatadas livres para crescerem, direcionando o próprio crescimento do crânio do bebé (Aguiar, n.d.).

O capacete é utilizado entre os 3 e 18 meses, devendo ser utilizado 23 horas por dia durante 3 a 5 meses, com consultas periódicas de reavaliação (Aguiar, n.d.).

Os capacetes para o tratamento plagiocefalia posicional, como o observado na figura 2.2, são um tipo de ortótese fabricada para promover o alinhamento, crescimento ou inibição de crescimento do crânio dos bebés que nascem com deformações cranianas.

Este tópico é essencial ao estudo da máquina CNC pois esta foi inicialmente projetada para a produção dos mesmos na empresa Padrão Ortopédico dado que, a sua concessão manual era bastante desconfortável para o paciente, passando por retirar o molde de gesso a partir da própria cabeça do bebé. Por

esta razão, também se tornava difícil para o técnico fazer e retirar o molde com as medições necessárias para o fabrico do capacete tendo este, na maioria das vezes, de ser ajustado constantemente.

Figura 2.2 - Capacete para tratamento de plagiocefalia posicional (Chang, 2020).

A realização do molde a partir da máquina CNC é vantajosa a vários níveis, tais como: é possível a produção de um molde positivo do crânio do bebé sem necessidade de utensílios como gesso, lixas, espátulas, brocas, não sendo necessário sujeitar o paciente a todo um processo moroso e repetidas vezes, no caso de ser necessário retirar novas medidas. O facto de a camada de gesso ser uma sobre espessura do crânio do bebé também levaria a cálculos desnecessários, ao contrário do molde positivo produzido na CNC.

Por fim, o molde pode ser utilizado várias vezes e o ficheiro eletrónico fica guardado para sempre que for necessário, não sendo este destruído como o de gesso. Evitando, assim, fazer o utente passar mais do que uma vez pelo processo.

2.1.1. Necessidade de produzir moldes positivos no fabrico

Toda a informação descrita no presente capítulo sobre a produção de moldes de próteses e ortóteses, bem como, de tópicos relacionados, foi adquirida pelos relatos e declarações de uma amostra de dez técnicos de saúde empregados na empresa Padrão Ortopédico consultados entre 2020 e 2021.

O objetivo principal de uma prótese ou ortótese é então, a melhor eficiência no que toca à substituição de um membro completo ou na realização de uma função específica do membro em questão, incluindo, no membro artificial, a melhor comodidade do paciente. Nesta vertente da saúde, qualquer incómodo da pessoa amputada é prejudicial para a sua vida. É inimaginável, segundo a amostra consultada de técnicos de saúde deste campo, um paciente ter lesões, mesmo estas sendo menores - como calos e escoriações, como consequência da utilização da prótese.

Cada prótese é concebida para um e só um paciente, para cada caso específico e para um tempo específico. Variáveis como a idade, o peso, o crescimento da pessoa, as cicatrizações da amputação, entre outras, são cruciais para delinear o fabrico correto da prótese ou ortótese. Para não falar que, um dos fatores mais influentes é o preço da mesma. Por ser um trabalho minucioso e que requer a utilização de uma grande quantidade de produtos, materiais e ferramentas, adicionando ao facto de não ser realizada uma produção em série, o fabrico de uma prótese é um processo caro, que leva muitas vezes à diminuição de qualidade dos equipamentos adquiridos pelo cliente.

Assim, as perguntas pertinentes no início da conceção de uma prótese ou ortótese são estas: que tipo de membro artificial é necessário? Que zonas do coto é que têm de ser protegidas para garantir a melhor comodidade? Qual o orçamento para isso?

A primeira e a segunda pergunta são respondidas, substancialmente, com o tipo de amputação que o paciente apresenta. A terceira pergunta depende do que a pessoa está disposta a ceder, pessoalmente ou através de um intermediário (como por exemplo, na maior parte dos casos, seguradoras).

Na questão de comodidade, é necessário recordar que cada prótese é uma prótese, é única e pessoal, a melhor forma de conceber um encaixe perfeito para cada pessoa, é exatamente a partir do molde do coto da pessoa.

Claro que, é impossível produzir uma prótese ou ortótese eficaz diretamente na superfície do coto do paciente para não falar que obrigar alguém a deslocar-se diariamente, até ao fim do processo, para proceder à tiragem de medidas e para experimentar encaixes de prova, é desnecessário e danoso financeiramente. Posto isto, ao longo dos anos foram implementados no fabrico de próteses e ortóteses, vários processos de tiragem de molde do coto do paciente, assim, durante a conceção do membro, os técnicos têm sempre um exemplar quase exato dos limites da amputação e dos membros necessários para a fixação.

Os moldes são essenciais para, aquando o procedimento e criação do encaixe prostático, os técnicos de saúde terem noção das dimensões e medidas adequadas ao paciente, das áreas onde devem criar alívio e das áreas onde adicionam pressão para distribuir a força da fixação da prótese e do seu peso, sem comprometer que, ao longo de um dia de utilização, o paciente se sinta dorido ou mesmo magoado nas áreas sensíveis do coto.

O processo de produção de moldes positivos é relativamente simples sendo, na sua generalidade, realizado um negativo do membro onde vai ser fixada a prótese ou ortótese que, através de um *software* ou por enchimento manual, espelha-se no molde positivo, onde são de seguida realizadas as correções necessárias.

2.1.2. Processo manual do fabrico de próteses

O fabrico de um membro artificial começa então pela aquisição de todas as características fisiológicas do individuo (como na ficha de utente apresentada na figura 2.3) que possam influenciar o tipo de prótese e processo a realizar. Toda a informação descrita neste texto foi retirada da consultoria de técnicos de próteses e ortóteses da empresa Padrão Ortopédico bem como imagens adjacentes ao tema.

FOLHA DE RECTIFICAÇÃO DE MOLDE

NOME DO AMPUTADO_____ DATA__-__-__
DIREITO/ESQUERDO_____ MOLE/MEDIO/RIJO_____
ADUÇÃO/ABDUÇÃO_____ ORTOPROTÉSICO_____
FLEXÃO/EXTENSÃO_____ Nº SAPATO _____
COMPRIMENTO _____ ALTURA DO SALTO_____
NÍVEL ISQUIÁTICO AO_____ SAPATO COM/SEM_____
CENTRO DO TROCÂNTER_____ ÍSQUION AO CHÃO_____
ML ÓSSEA_____ GOLO_____ PRATOS TIBIAIS AO CHÃO_____
ÂNGULO ÍLIO-FEMURAL_____
ML MOLE_____
ML LONG_____
ML MOLE GOLO_____
DISTÂNCIA BI-SUB-TROCANTÉRICA_____
REDUÇÃO EM TERMOS FINAIS_____

DISTÂNCIA ABAIXO ÍSQUION	MEMBRO REMANESC. CIRCUNF.	TENSÕES GOLO CIRCUNF.	MOLDE POSITIVO CIRCUNF.		GESSO REMOVIDO	FINAL

Figura 2.3 - Ficha de utente utilizada para reconhecimento do estado do paciente.

Através de uma consulta presencial, o técnico faz as medições do membro onde vai ocorrer a fixação. Geralmente, marcando as áreas debilitadas, as superfícies de pressão e algumas possíveis áreas de alivio diretamente na pele do paciente. Após todas as dimensões serem registadas, o técnico sobrepõe sobre o coto do utente ligaduras de resina ou ligaduras de gesso, pressionando ao máximo contra o membro, para garantir que todas as marcas registadas e que todos os relevos fiquem gravados após a secagem das ligaduras.

Depois do molde negativo estar concebido, procede-se com o enchimento do mesmo com gesso para gerar o molde positivo. Normalmente, aquando o enchimento é implementado um veio de metal no centro, para facilitar o manuseamento do molde positivo à *posteriori*.

De seguida, o molde positivo é fixado numa bancada onde se promove a conceção das correções. Existem dois métodos de correção dos moldes.

Após a correção adequada do molde positivo é fabricado o encaixe de prova ou o encaixe final – onde o paciente vai colocar o coto e fixar ao membro. Os encaixes de prova são produzidos em policarbonato, geralmente com aditivos para que o material seja o mais termicamente moldável possível. Na máquina de vácuo é realizada a fixação do molde positivo seguida da sobreposição de uma placa de policarbonato que, após fechada nas extremidades é pressionada pelo sistema de vácuo contra o molde, adotando todos os relevos do mesmo.

Por fim, é removido da máquina de vácuo e destruído o molde positivo ficando, assim, apenas o encaixe de prova. Aquando a prova, o técnico é capaz de realizar algumas alterações manualmente do encaixe (por isso a necessidade de ser termicamente moldável) consoante as indicações do utente.

Os encaixes finais são, na sua generalidade, maquinados a partir de um material compósito para garantir resistência na prótese final. Se previamente foi realizado um encaixe de prova, este serve como molde negativo e volta-se a proceder ao enchimento com gesso e sucessiva produção do encaixe final na máquina de vácuo. Caso apenas seja indicado o fabrico de um único encaixe, a partir do molde positivo é diretamente realizado o produto final. De notar que, em trabalhos minuciosos como este, é sempre preferível a realização de um encaixe de prova, para eliminar o maior número de erros possível, erros os quais, na maior parte das vezes não dizem respeito ao trabalho do técnico de saúde, mas sim do próprio movimento da pessoa com o membro artificial em relação aos membros de apoio.

2.1.3. Processo digital e industrial de produção de moldes, cosméticas e ortóteses

Em alguns casos do fabrico de próteses e ortóteses o processo de tiragem dos moldes torna-se complicado. Por exemplo, moldes para ortóteses da cabeça, para cosméticas do braço e da mão em que é extremamente difícil retirar as ligaduras de gesso (ou de resina) do paciente com os devidos contornos e relevos ou sem tornar o procedimento sujo e demorado, são utilizadas técnicas mais avançadas para recriar o molde positivo do membro. Uma delas é a utilização de um *scanner* 3D para a obtenção de uma geometria digital do molde.

São várias as formas de digitalização tridimensional, estas dependem do tipo de scanner que é necessário e do tipo de digitalização que se pretende (se é uma digitalização a longa ou curta distância ou, se

é pretendida uma maior ou menor resolução da imagem). Entre as diferentes formas de digitalização, existem três modos principais de se proceder com a obtenção de uma imagem tridimensional: digitalização por tecnologia de laser, projeção e estruturação de luz ou por trajetória ótica (EMS, 2019).

Na tecnologia a laser é utilizada triangulação trigonométrica onde feixes de luz são consecutivamente enviados até a superfície a digitalizar, sendo a reflexão do feixe captada pelos sensores do *scanner*, criando, assim, uma superfície com milhões de pontos (EMS, 2019).

Já numa tecnologia de projeção e estruturação de luz, é enviado até à superfície um padrão, normalmente constituído por contrastes de luz LED branca ou azul que, consoante as deformações do padrão causadas pelas irregularidades da superfície, formam digitalmente os relevos da mesma (EMS, 2019).

Por fim, a utilização de um scanner de trajetória ótica baseia-se na obtenção de *frames* em relação a um ponto conhecido do espaço sendo que, quando as coordenadas são lidas pelos sensores, a imagem bidimensional é colocada e aglomerada com as outras diferentes imagens retiradas de pontos conhecidos (EMS, 2019).

Após a obtenção do molde digital, através de *softwares* de CAD como, por exemplo, "Meshmixer" (utilizado em várias empresas por ser *open source*), como representado na figura 2.4, são efetuadas correções a nível da digitalização, nomeadamente, falhas de dados potencialmente existentes, eliminação de pontos em excesso e alteração do tipo e forma de malha do molde, consoante o nível de rigor desejado.

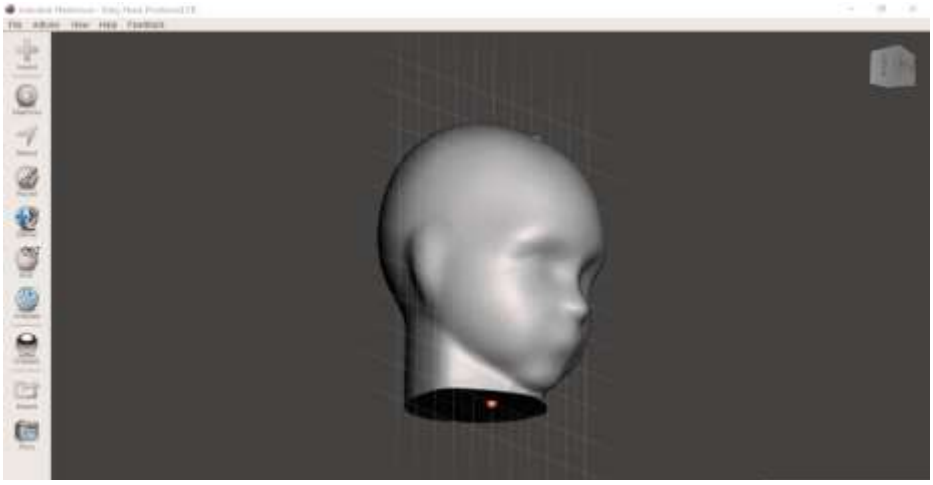


Figura 2.4 - Posicionamento da cabeça para maquinar no *software Meshmixer* (Autodesk).

Posto isto, em *softwares* de modelação CAD, dependendo do tipo de produto final, é possível transformar a malha obtida numa superfície paramétrica podendo, posteriormente, adicionar uma espessura.

Em certos casos, é necessário proceder á correção dos moldes positivos. No fabrico digital de moldes de cosméticas (por exemplo), como são produzidas em elastómero de poliuretano por impressão 3D tem de ser implementada uma estrutura de encaixe para acoplar a cosmética ás restantes partes da prótese. Já em capacetes para correção de plagiocefalia, a correção é diretamente feita nas malhas pois, como

se trata de um molde positivo produzido por desbaste de um bloco, são apenas necessárias as coordenadas da superfície.

Por fim, o molde digital é convertido em código através de *software* CAM, guardado e enviado para a máquina que produzirá uma peça final ou um molde positivo, representada na figura 2.5.

Figura 2.5 - Molde positivo de um paciente com plagiocefalia maquinado na CNC da Padrão Ortopédico.

2.1.4. Diferença entre fabrico de próteses e ortóteses convencional e digital

Como em todas as formas de produção, existem sempre diferenças na produção convencional (manual) e na produção digital ou mais automatizada. Todavia, segundo os técnicos da empresa Padrão Ortopédico, qualquer uma das duas opções não deixa de ser correta e a melhor solução depende sempre do caso que se trata. Na tabela 2.3, encontram-se algumas diferenças gerais entre estes dois tipos de fabrico.

Tabela 2.3 - Diferença entre fabrico de próteses e ortóteses convencional e digital.

Fabrico convencional de moldes	Fabrico digital de moldes
Vasta utilização de diferentes utensílios	Geralmente utilização apenas de um scanner, computador e de uma máquina CNC
O processo de tiragem do molde é demorado	O processo de tiragem do molde é realizado com o scanner e é relativamente rápido
Há grandes quantidades de desperdício de materiais para fabricar o molde	São apenas desperdiçados os materiais de excesso do bloco ou os suportes (no caso de FDM)

Tabela 2.3 - Diferença entre fabrico de próteses e ortóteses convencional e digital. (cont.)

Fabrico convencional de moldes	Fabrico digital de moldes
Processo mais evasivo para o paciente (principalmente na tiragem do primeiro molde)	Processo muito menos evasivo para o paciente (não há praticamente contacto)
Maior sensibilidade na correção das zonas de alívio e pressão	Como não é um processo físico não há tanta sensibilidade a fazer as correções devidas
Difícil de armazenar o modelo físico do molde	O modelo digital é facilmente guardado
Dificuldade em alterar as correções e formas já existentes	Qualquer modelo digital é fácil de alterar independentemente do formato existente
Processo muito demorado	Processo demorado mas, comparativamente ao convencional é consideravelmente mais rápido
Muito mais fácil de fazer correções específicas que implicam a utilização de peso e pressões para a alteração da forma dimensional	É difícil ter noção das formas dimensionadas quando é necessário aplicar uma carga

2.2. Máquinas CNC e a indústria de próteses e ortóteses

Por definição, relacionada com o tipo de máquina pretendida para este estudo, uma CNC fresadora é uma máquina de comando numérico assistido por computador capaz de remover material de um dado bloco utilizando, simultaneamente, o movimento de uma ferramenta de corte com o movimento de, pelo menos, dois eixos ao mesmo tempo (Smid, 2003).

É sabido que, as máquinas de comando numérico assistidas por computador, nomeadamente, as CNC, são utilizadas para uma grande variedade de produção. Consoante a sua complexidade e a precisão pretendida pelo operador, estas máquinas são capazes de criar uma ampla gama de peças (Smid, 2003).

No que toca à área do fabrico de próteses e ortóteses, existem duas empresas que se dedicam, exclusivamente, ao fabrico de máquinas CNC para a produção de moldes destes equipamentos em específico. Apesar de existirem mais máquinas capazes de produzir as peças desejadas, os produtos destas marcas focam-se, essencialmente, na melhor utilização das CNC para os objetivos intrínsecos na área.

Apesar dos materiais utilizados para a produção de moldes serem de baixa densidade, aumentando a facilidade de corte do mesmo, a peça exige que o rigor e a precisão sejam extremamente elevados, para que o molde reflita, com a maior semelhança, o membro do utente. Diminuindo, assim, falhas e incómodos no posterior fabrico da prótese ou ortótese ou, até mesmo a necessidade de executar um novo molde.

Uma das principais especificações destas máquinas é o tamanho do curso do eixo X tal que, a CNC permita a revolução de um mole desde um dedo, até um membro inferior inteiro na mesma máquina.

2.2.1. Máquinas "Vorum"

Como a própria empresa reivindica, todas as máquinas "Vorum" são rápidas, precisas, fáceis de operar e feitas para durar. Funcionam como parte integrante da solução "Vorum" CAD/CAM para próteses e ortóteses: basta importar o arquivo de projeto "Canfit 3D" para o molde, colocar uma peça bruta de escultura e pressionar "Iniciar". Em minutos, um molde prostático ou ortótico complexo está pronto para o seu fabrico. Dispositivos finais, como palmilhas e assentos de espuma macia, podem ser fresados diretamente nestas máquinas "Vorum" (Vorum, n.d.).

A "Vorum" produz máquinas de três a sete eixos, todas elas previamente programadas para trabalhar com o próprio *software* CAD/CAM desenvolvido pela empresa (Vorum, n.d.).

Por exemplo, na figura 2.6, está representada uma CNC de quatro eixos fabricada por esta empresa. Contemplando um eixo rotativo e uma superfície de corte plana, a máquina de 4 eixos é a solução para diversas necessidades de fabrico como assentos e palmilhas ortopédicas, além de uma ampla gama de modelos de ortóteses e próteses (Vorum, n.d.).

Figura 2.6 - CNC 4 eixos "Vorum" (Vorum, n.d.).

Oferece, ainda, versatilidade e eficiência máximas. A grande superfície plana, facilita a produção direta de uma almofada de assento ou de até 8 pares de palmilhas macias de uma vez. Permite a produção

de modelos e dispositivos com uma variedade de materiais duros e macios, incluindo poliuretano, EVA e cortiça. Sendo as dimensões desta máquina reduzidas (2 x 2,6 m^2) (Vorum, n.d.).

É de salientar que a empresa fornece todos os equipamentos necessários para a correta utilização da máquina, desde *software*s de *scan*, equipamento, *software* de CAM/CAD e a máquina em si (Vorum, n.d.).

2.2.2. Máquinas "Rodin4d"

A "Rodin4d" é também uma empresa que visa responder às necessidades de produtores e fabricantes de próteses e ortóteses. Tal como a "Vorum", disponibiliza em catálogo todos os componentes necessários para a produção de aparelhos ortopédicos, iniciando-se de igual modo em *software*s de *Scan* e CAM/CAD para, posteriormente, implementar o resultado nas suas máquinas produzindo, assim, as peças desejadas pelo operador (rodin4d, 2018).

Existem várias máquinas CNC e robôs fabricados pela empresa, para que o comprador possa adquirir de um modelo mais simples a um mais complexo, conforme a necessidade (rodin4d, 2018).

O modelo "S" da "Rodin4d" está representado na figura 2.7. Este apresenta um tamanho compacto, facilitando a integração no ambiente de trabalho. Apesar do seu tamanho, possibilita o fabrico de quase todos os aparelhos ortopédicos requisitados (rodin4d, 2018).

A área de trabalho é de 700 x 500 mm^2 tendo como principais características técnicas de maquinagem (rodin4d, 2018):

- Motor do eixo: 3000/10000 T / min;
- Fonte de alimentação: 380 V, 60 Hz, 25 A;
- Tempo médio de ciclo: 20 min;
- Nível de som <80db;
- Tamanho: largura 1600 x comprimento 1740 x altura 1880 mm;
- PC industrial integrado.

Figura 2.7 - CNC 4 eixos "Rodin4d" (rodin4d, 2018).

2.2.3. Controladores de máquinas CNC e plataformas GUI no mercado

Os controladores e as interfaces CNC vivem em codependência, ou seja, grande parte dos controladores existentes no mercado estão acoplados à respetiva interface do comerciante. À exceção de certos protótipos encontrados por programadores *freelancers*, a industria responsável por fabricar este tipo de equipamento tende a ser exclusiva. Por outras palavras, a compra de um controlador para uma CNC, inclui a compra de todos os restantes produtos, nomeadamente, motores especificas, drivers e sensores.

Por serem normalmente empresas que fabricam as próprias máquinas, estes equipamentos são ideais apenas para as máquinas que os mesmos fornecem. Assim, não é garantida uma eficiência adequada quando implementados em máquinas diferentes daquela para a qual foram concebidos. Posto isto, a maioria destas filiais, apenas fornece informações detalhadas do produto e do custo quando contactados pelo comprador, para poder validar a sua escolha e para garantir que o equipamento não sofrerá alterações indesejadas nem será alvo de "estudos" para produzir equipamentos idênticos.

Inicialmente, avaliando alguns fornecedores *online* destes equipamentos, estipulou-se um intervalo de preços de, aproximadamente, 9 000 a 19 000 euros. Este custo remete para a compra de *kits retrofit* para CNC, ou seja, um conjunto de equipamentos para renovar as máquinas CNC. É claro que, o custo do produto advém da qualidade e fiabilidade do mesmo. Para empresas que se gerem pela marca e que requisitam grandes quantias de equipamentos, é necessária uma coerência nos equipamentos para que, se precisa, seja possível a interação com o fabricante (Machine Tool Products, n.d.).

O conjunto "CNC 808D da Siemens" pré-fabricado, representado na figura 2.8, é para uma fresadora básica de 3 eixos. Os componentes do *kit* incluem: *Display*/ controlador CNC "Siemens" (PPU), painel de controlo da máquina (MCP), (3) *drivers* e servo motores "Siemens" com conjuntos de cabos, gabinetes, braço pendente, componentes de controlo, (2) fontes de alimentação, transformador de controlo, saída e relés de segurança, desconexão rotativa, disjuntores reinicializáveis. O custo é de aproximadamente treze mil euros (Machine Tool Products, n.d.).

Figura 2.8 - *Kit* CNC 808D da Siemens pré-fabricado (Machine Tool Products, n.d.).

O conjunto "CNC TKP Fagor 8055i FL MC TKP" de 3 eixos para CNC (fresadora) é um sistema servo-CNC pré-projetado. O sistema dos servos motores é projetado num gabinete elétrico. Os dois sistemas são integrados para fornecer um sistema *turnkey* (funcional e pronto para instalação). Toda a eletrónica necessária é fornecida, incluindo os cabos de alimentação do motor, cabos do codificador e servo motores. Cada sistema é pré-cablado e configurado para sua aplicação específica de máquina por técnicos da marca e está totalmente funcional na entrega. Este equipamento, representado na figura 2.9, tem um custo de, aproximadamente, dezanove mil euros (Machine Tool Products, n.d.).

Figura 2.9 - *Kit* CNC TKP Fagor 8055i FL MC TKP (Machine Tool Products, n.d.).

Entretanto, existem sempre mais alternativas de mercado. Grandes distribuidores mundiais de produtos de baixo custo, como o *Alibaba* e o *Aliexpress* fornecem alternativas substancialmente mais em conta, nunca garantindo, claro, instalação, eficiência e compatibilidade com o equipamento.

Na comunidade do *AliExpress*, existem modelos, como o representado na figura 2.10, um *kit* de *hardware* para CNC, "ddcsv3.1 4-axis", que engloba quatro motores Nema 23/57, um controlador de movimento, uma fonte de alimentação e um botão de paragem de emergência. Com um custo muito mais acessível de, aproximadamente, quatrocentos e quarenta euros.

Figura 2.10 – *Kit* para CNC ddcsv3.1 4-axis (AliExpress, 2020).

O grande problema em adquirir componentes para a construção de máquinas CNC separadamente, incide na incompatibilidade entre o *hardware* e *software*. Apesar de ser possível a compra de interfaces e controladores isoladamente, estes necessitam do *software* próprio, especificamente desenvolvido para cada um eles.

A acrescentar ao elevado custo de mercado do *hardware* em questão seria, de certo modo irrelevante, a análise de conjuntos destes dois componentes para instalar na máquina CNC em estudo. Por um lado, porque esta já provém de um *firmware* completamente funcional e simples instalado, por outro lado, porque, como já foi mencionado, haveria necessidade de alterar todos os componentes existentes na máquina, à exceção da estrutura, para garantir eficiência. De salientar também que, o intervalo monetário estipulado é remetente a máquinas de alto rendimento, dimensionadas e projetadas para suportar uma elevada gama de materiais para corte, sendo que, seria um investimento extremamente elevado para o tipo de trabalho que a CNC da empresa Padrão Ortopédico acarreta.

Anda assim, haveria a possibilidade de encontrar um *firmware open-source* que fornecesse maior facilidade em projetar uma interface para a máquina. Apesar de não ser viável, pois estes *firmwares* estão ligados a programas de controlo apenas desenvolvidos para computadores, seria, tendo em conta o custo de um equipamento de origem, uma opção a explorar. A vantagem deste tipo de *software* é que, para além de gratuitos, são ajustáveis às necessidades do utilizador, dependendo do numero de eixos, tipo de motores, binário, velocidades máximas e parâmetros de maquinagem.

2.2.4. *Firmwares* para controlo de Máquinas CNC

Um dos *software*s mais utilizados pela comunidade construtora de máquinas de comando numérico assistido por computador, é o "Marlin". Este *firmware* é compatível com máquinas que suportam o movimento de quatro eixos, geralmente três lineares e um rotativo. Pode ser encontrado em plataformas de programadores como o GitHub (Rdpowers, 2016).

Este *firmware* RepRap (com lincensa gratuita segundo a GNU, "General Public License") foi programado a partir de firmwares como o "Sprinter" e o "Grbl" por Erik van der Zalm. Permite a interrupção de cursos de maquinagem, comporta o acesso de cartões SD para transferência do código G, suporte para implementação de um ecrã LCD, menu para a interface, representada na figura 2.11, entre outros (Rdpowers, 2016).

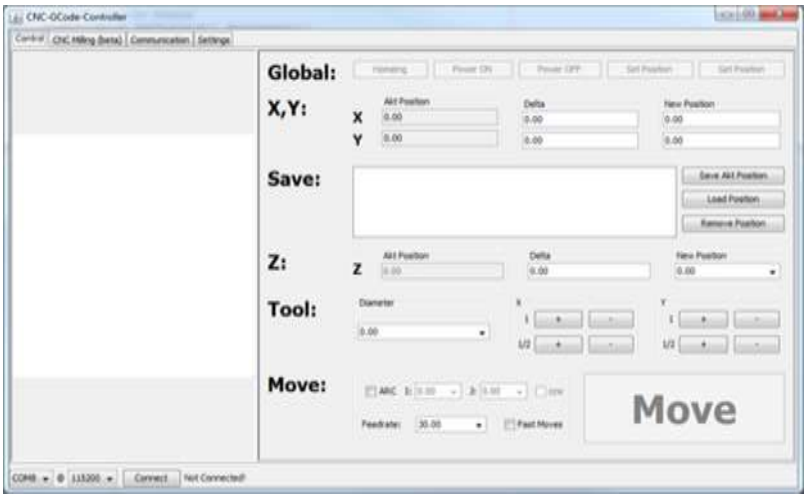


Figura 2.11 - Interface fornecida para o MArlin firmware. (Breiler, 2020a)

Por outro lado, o "Grbl" é uma alternativa de alto desempenho e baixo custo que visa controlar o movimento em máquinas CNC de fresagem. Este *firmware* está projetado para ser utilizado em qualquer tipo de controlador Arduino do mercado (Breiler, 2020a).

Este controlador é escrito em "C" altamente otimizado, utilizando todos os recursos inteligentes dos *chips* AVR para obter um tempo preciso e operação assíncrona. É capaz de manter até 30kHz de pulsos de controlo estáveis e livres de *jitter*[1] (Breiler, 2020a).

Este programa aceita código G consoante os padrões universais e foi testado com a saída de várias ferramentas CAM, não apresentando anomalias. Arcos, círculos e movimento helicoidal são totalmente suportados, bem como todos os outros comandos primários do código G. Funções macro, variáveis e a maioria dos ciclos fixos não são suportados, mas possíveis de executar em plataformas GUI. (idem)

O "Grbl" promove o controlo total antecipado da aceleração. Isso significa que o controlador preverá até 18 movimentos futuros, planeando assim, antecipadamente, as velocidades para executar uma aceleração suave e movimentos com a menor vibração possível (Breiler, 2020a).

Este *firmware* é *open-source* e encontra-se lançado através da licença "GPLv3" (Breiler, 2020a).

No projeto atual, está a ser utilizada uma expansão do Grbl para cinco eixos, "Grbl-Mega-5X" (figura 2.12), de modo a incluir os eixos rotativos, nomeadamente, o eixo A (Breiler, 2020a).

Figura 2.12 - Logótipo da extensão de 5 eixos do Grbl (Breiler, 2020a).

[1] Jitter – Variação do tempo em que os dados chegam ao recetor, quanto maior o jitter (milissegundos), maior a latência da transmissão de dados.

Esta versão do Grbl foi lançada a partir da licença inicial na plataforma online "GitHub". Neste caso, devido à maior quantidade de sistemas eletrónicos a incluir, é exclusivamente necessária a utilização de um controlador Arduino Mega (Breiler, 2020a).

Tal como os próprios fornecedores indicam, qualquer operador que necessite de um controlador para fresagem simples e agradável e que pretenda a sua implementação em qualquer tipo de controlador Arduino, capaz de acolher todas as ligações eletrónicas necessárias para o controlo dos motores, pode utilizar este *firmware*. Assim, é desnecessária a utilização de estruturas cabladas extremamente complexas, facilmente conseguidas através do controlador Arduino (Breiler, 2020a).

Também utilizadores que pretendam implementar este controlador em trabalhos e projetos mais específicos, são evidenciados pelo fornecedor como incluídos no grupo de pessoas que o "Grbl" será uma mais valia para elas (Breiler, 2020a).

Intrínsecamente, no que diz respeito ao projeto onde se encontra a ser utilizado, esta foi a opção inicial de controlador implementada pelo criador da CNC, eliminando os custos da compra de um *software* específico para controlar a máquina e de todos os componentes mecânicos e estruturais necessários para os implementar, sendo que, é facilmente controlável via uma ligação em série em qualquer computador (Breiler, 2020a).

O interpretador de código G implementa um subconjunto do padrão "LinuxCNC" e é compatível com a maioria das ferramentas CAM (Breiler, 2020a).

A lista de código G suportado pela versão de Grbl 1.1. encontra-se representada na tabela 2.4 (Breiler, 2020a).

Tabela 2.4 - Códigos G suportados (versão 1.1) (Breiler, 2020a).

G0	Psicionamento rápido
G01	Interpolação linear
G2, G3	Interpolação circular no sentido horário (G2) ou sentido anti-horário (G3)
G4	*Dwell* (tempo de permanência)
G10	Entrada de dados de Deslocamentos de Coordenadas de Trabalho
G17, G18, G19	Seleção de Plano: XY (G17); XZ (G18); YZ (G19).
G20, G21	Entrada de dados: em polegadas (G20) ou em milímetros (G21)
G28	Prosseguir para a posição pré-definida
G28.1	Definir posição pré-definida (de todos os eixos)
G38.2	Sondagem (dois eixos)

Tabela 2.4 - Códigos G e M suportados (versão 1.1) (Breiler, 2020a). (cont.)

G40	Anulação da compensação de raio da ferramenta de corte
G43, G49	Ativação da compensação de comprimento da ferramenta (G43); Anulação da compensação de comprimento da ferramenta (G49)
G53	Sistema de Coordenadas absolutas
G54, G55, G56, G57, G58, G59	Sistemas de Coordenadas de Trabalho
G61	Modos de controlo de caminho (*stop-check*)
G80	Cancelar ciclos fixos do grupo 09
G90, G91	Coordenadas absolutas (G90); Coordenadas incrementais (G91)
G91.1	Modos de distância *Arc* IJK
G92	Desvio de Coordenadas
G92.1	Limpar desvios do sistema de coordenadas
G94	Modos de avanço (unidade por minuto)
M0, M2, M30	Pausa (M0), Fim do Programa (M2), Fim do programa e retorno (M30)
M3, M4, M5	Controlo da *Spindle*
M8, M9	Controlo de refrigeração (Ligar (M8) e Desligar (M9))
M56	Controlo de anulação de movimento de estacionamento

Todo o processo de implementação baseia-se na conexão em série do controlador Arduino previamente equipado com o "Grbl", a um canal para enviar e receber informação. Para isso, é necessária a utilização de uma banda de frequência predefinida, neste caso, 115200 que contempla oito bits, sem paridade e com um bit de pausa (Breiler, 2020a).

Existe, também *open-source*, um controlador compatível com sistemas operativos como "Windows" e "Mac", específico para comunicar com o *firmware* "Grbl" incluso na máquina CNC. Dentro de muitas versões, é possível observar a constituição gráfica do controlador, "Grbl Controller", na figura 2.13 (Breiler, 2020a).

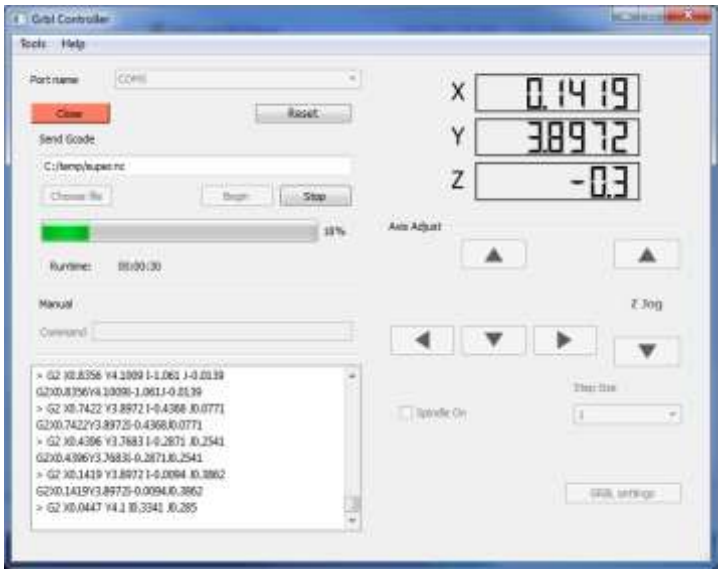


Figura 2.13 - Plataforma Gráfica de Comunicação com o *firmware*, "Grbl Controller" (Breiler, 2020a).

Após a ligação em série, se esta for bem-sucedida, o Grbl envia uma mensagem para o canal como a exemplificada na figura 2.14.

```
Grbl 0.9i ['$' for help]
```

Figura 2.14 - Resposta do *firmware* quando ativo.

Para a comunicação com o *firmware* através de um canal é necessário que a linguagem utilizada seja a requerida pelo controlador. Os programadores do Grbl utilizam caracteres de linguagem ASCII[2] como meio de comunicação, sendo que, quando o operador envia qualquer um dos códigos compatíveis, o "Grbl" responde com funções ou listas de funções predefinidas, tais como as descritas na tabela 2.5, para cada um dos comandos (Breiler, 2020a).

O "Grbl" é uma ferramenta extremamente equipada, própria para utilizar em máquinas de comando numérico assistidas por computador, pelo que, os parâmetros estão predefinidos com uma enorme variedade de operações que, consoante um valor opcional, dado pelo próprio controlador, é possível, com uma simples troca do valor de parâmetros, ajustar o programa às necessidades do utilizador, esta lista de parâmetros encontra-se descrita no apêndice I. (Breiler, 2020a).

[2] ASCII - "*American Standard Code for Information Interchange*", traduzido para código padrão americano para troca de informação, corresponde a uma linguagem computacional que retorna funções como caracteres.

Tabela 2.5 - Principais parâmetros de comunicação com o "Grbl" *Firmware* (Breiler, 2020a)

"$$"	Ver definições do Grbl;
"$#"	Ver os parâmetros (#);
"$G"	Ver estado da análise;
"$I"	Ver informação de Versão;
"$N"	Ver blocos iniciais;
"$x=valor"	Alterar o valor das definições do Grbl;
"$Nx=linha"	Alterar blocos iniciais;
"$C"	Alterar modo de leitura do código G;
"$X"	Desbloquear o modo de alarme;
"$H" ;	Executar o *homing*[3]
"~"	Iniciar ciclo de código;
"!"	Interromper o ciclo de código;
"?"	Verificar o estado atual do controlador;
"ctrl-x"	Executar reset ao *firmware*.

[3] Homing – regresso ao ponto inicial definido na máquina como zero-máquina.

3. Análise detalhada da CNC fabricada na empresa Padrão Ortopédico

A máquina de comando numérico assistido por computador encontrada na empresa Padrão Ortopédico foi inicialmente construída para revolucionar a criação de moldes de capacetes para pacientes com plagiocefalia. A CNC foi concebida pelo Engenheiro Joel Gomes que, de acordo com o desejado pela empresa, implementou um mecanismo com dois eixos principais X e Z, representando estes um plano de deslocamento vertical e horizontal, respetivamente, e um eixo auxiliar em torno do eixo Z, o eixo rotativo A que, faz a peça girar em torno de si própria verticalmente.

Esta máquina funciona através da remoção de material (fresadora), onde um bloco de matéria é alocado ao prato giratório movido sobre o eixo A que, ao longo do seu percurso sofre remoção de material pela fresa (ferramenta de corte) implementada paralela ao eixo Z (plano da ferramenta de corte). Por sua vez, a fresa exerce maior ou menor profundidade ou movimento longitudinal consoante as coordenadas dadas pelo código G, ou por comando manual, ao sistema mecânico do eixo Z, ou X, respetivamente.

A máquina tem como dimensões máximas uma altura de dois metros e comprimento de um metro e quarenta centímetros. O seu esqueleto foi dimensionado a partir de perfis de alumínio e placas de acrílico como representado na figura 3.1. Esta máquina CNC foi nomeada, pelo criador, “Nada”.

Figura 3.1 - CNC "NADA" Padrão Ortopédico.

Apesar de existir contacto com o engenheiro que desenvolveu a máquina, este já não se encontrava na empresa aquando a pesquisa e estudo da mesma. Posto isto, todo o estudo e conhecimento ao longo

deste capítulo é fruto do trabalho realizado na empresa, sem qualquer guia ou informações prévias sobre a CNC.

3.1. Constituição e funcionamento da CNC da Padrão Ortopédico

A máquina de comando numérico "Nada", foi construída de raiz, em função das necessidades pretendidas para a criação de moldes positivos para fabrico de próteses e ortóteses.

Como já foi referido, apesar de, em primeira instância, o objeto ter sido a execução de moldes de pacientes com plagiocefalia, esta máquina tem um sistema de eixos simples, mas, geralmente utilizado pelas outras CNC do mercado para a área de trabalho e saúde em que se engloba.

Os principais componentes mecânicos que englobam os sistemas de transmissão de movimento para que, seja possível uma idealização do atual funcionamento da máquina e todos os outros componentes estarão especificados no apêndice II.

O esquema representado na figura 3.2 enquadra, simplificadamente, os quatro movimentos que são exercidos aquando uma maquinagem na CNC.

O motor, representado a azul, é responsável por mover a ferramenta num plano horizontal, através do fuso de esferas ao qual os componentes estão mecanicamente ligados. A vermelho, encontra-se os motores responsáveis pelo movimento do eixo X, que promovem o movimento vertical da ferramenta, deslocando todo o sistema através das guias verticais, consoante o movimento transmitido desses motores para o sistema de polias e correia.

O motor do eixo rotativo A está representado a laranja e é responsável por transmitir um movimento rotativo sobre o eixo X através de um sistema de polias. Por fim, a *Spindle* (representada a verde) promove a rotação direta da ferramenta.

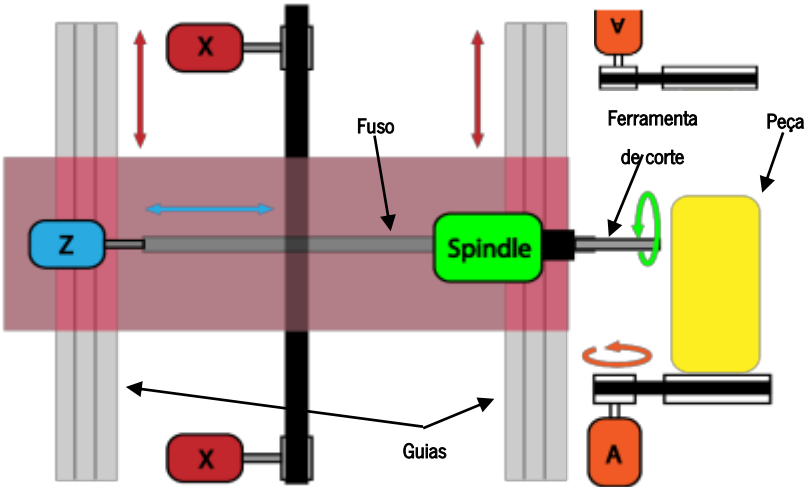


Figura 3.2 - Esquema do funcionamento das deslocações de eixos na CNC NADA.

Numa análise mais aprofundada à máquina, é possível identificar os diferentes grupos de sistemas, nomeadamente, os componentes mecânicos, os sistemas de motores e *Drivers* (e respetivas fontes de transformação), o controlador (posprocessador) e a interface Homem-Máquina.

Na figura 3.3, está representado um esquema em função da passagem de informação, energia e da ligação de todos os componentes do mecanismo.

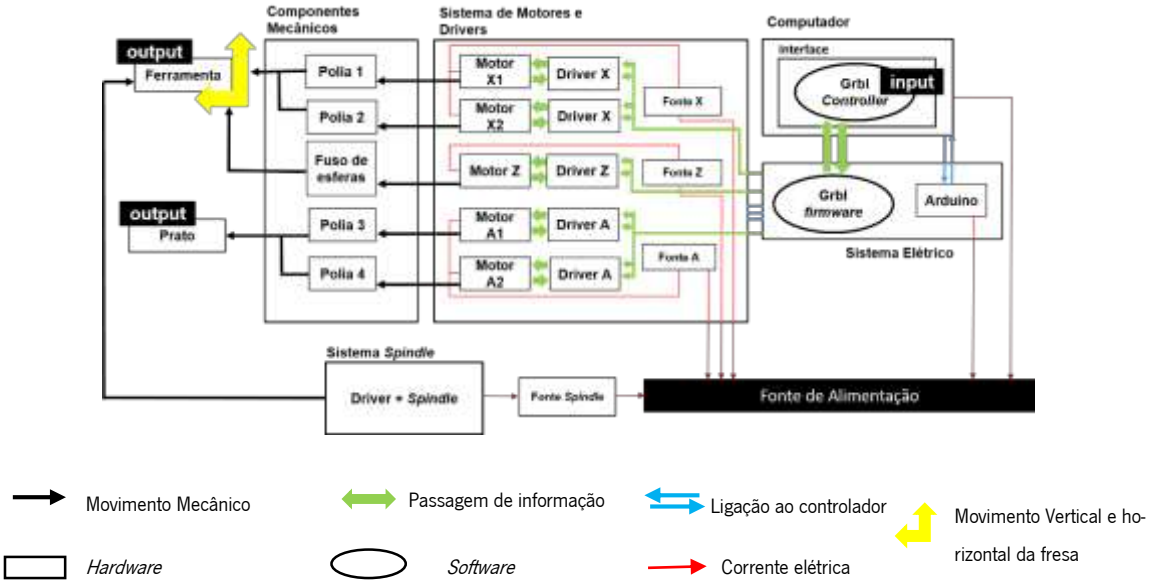


Figura 3.3 - Esquematização da passagem de informação, corrente e movimento da CNC NADA.

Em termos de circulação das funções, o *input* de informação é sempre gerado na interface homem-máquina (MMI) que, neste caso, encontra-se externa a todos os outros sistemas do aparelho, dado que é utilizado um computador com o *software* "Grbl" para fazer a gestão de parâmetros, envio de códigos e tarefas, entre outros. O *input* é enviado para o "Grbl" *firmware* encontrado no controlador Arduino da CNC.

O *firmware* guarda e trata a informação recebida no *buffer* do controlador Arduino (PLC). Após o tratamento da informação, ou seja, a tradução da linguagem, esta é enviada para os drivers dos motores que transformam essa informação em pulsos elétricos que são sucessivamente enviados para os motores e daí, transformados em movimentos mecânicos, *output*, da ferramenta (movimento vertical ou horizontal identificado com setas amarelas na figura 3.3) ou no movimento rotativo do prato onde é ficado o bloco. A rotação da fresa é um *output* externo do sistema sendo que, a *spindle* não se encontra ligada ao controlador Arduino, tendo uma interface manual separada de ajuste de valores.

O sistema MMI (apêndice III) da máquina comporta então um computador externo que é ligado ao controlador Arduino da máquina todas as vezes que é necessário o manuseamento da mesma e que, aquando toda a operação, tem de continuar ligado. Na figura 3.4, está representado o sistema elétrico existente na CNC da P.O. e a respetiva conexão do PLC (apêndice III) e do MMI.

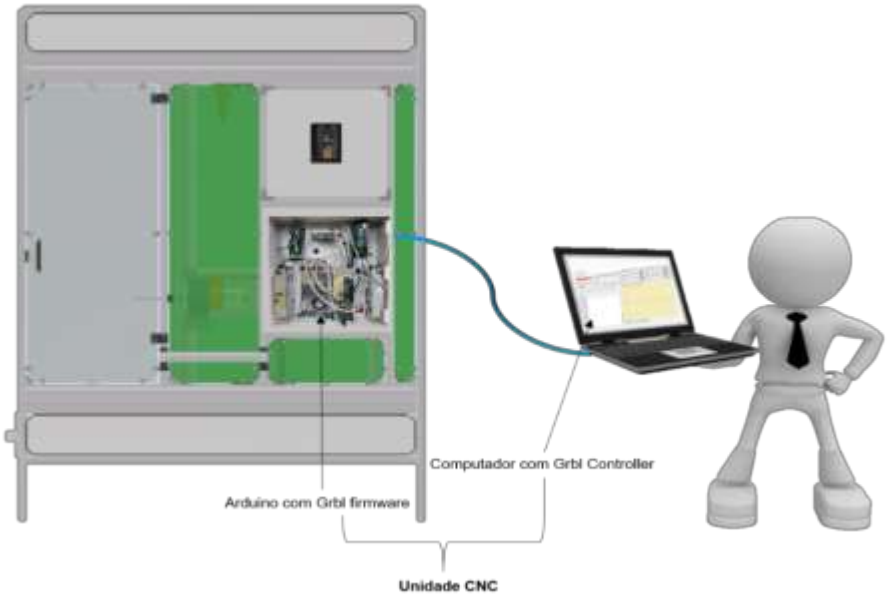


Figura 3.4 - Sistema elétrico da Máquina CNC "Nada" e a conexão ao "Grbl" *firmware*.

A unidade CNC da máquina é então constituída por estes dois conjuntos de *hardware* e *software.* A ligação é sempre realizada através da entrada USB 2.0 do controlador Arduino para a USB do computador, identificando a porta do canal em série e ligando o Grbl Controller no computador. Após realizada a conexão, o operador tem total controlo sobre a máquina, nomeadamente para promover a alterar os 136 parâmetros existentes no "Grbl" *firmware*, mover os eixos manualmente, promover e parar operações. Caso o computador fique sem bateria aquando uma operação a máquina continua a funcionar a não ser que seja manualmente cortada a energia. Porém, é impossível voltar a ligar o canal anterior, ou seja, operador deixa de ter controlo sobre a operação que está a ser executada. Isto acontece por serem sistemas separados, ou seja, o *firmware* não é ativado só depois do Grbl estar ligado, mas sim, porque existe energia na máquina.

A ligação para a passagem de informação entre os drivers e o controlador Arduino no circuito é executada de forma cablada. Como é necessário ligar cada um dos componentes ao pórtico de corrente (5 V) e à Terra (GND), está implementada no sistema uma *protoboard* para permitir todas as ligações. Ainda, cada pino do motor é ligado aos *drivers* que, por sua vez, estão ligados à fonte de alimentação do. Os pinos dos motores para direção, limite e pulso, estão ligados diretamente ao controlador Arduino (através da *protoboard*) e encontram-se definidos na tabela 3.1.

Tabela 3.1 - Ligação dos pinos do "Grbl" no controlador Arduino Mega 2560 (pinout).

Função		Pino
Limite	Eixo Z	53
	Eixo X	51
	Eixo A	50
Direção	Eixo Z	37
	Eixo X	35
	Eixo A	34
Pulso de Step	Eixo Z	22
	Eixo X	24
	Eixo A	25
Reset		0
Stop (pausa)		1
Start (continuar)		2
Porta de Segurança		3

Os pinos correspondentes ao "reset", "stop", "start" e "porta de segurança" estão ligados a todas os *drivers* de cada motor, para que, atuem igualmente em todos quando utilizados.

Esta informação depende do tipo de controlador Arduino utilizado, sendo que, a CNC utiliza um controlador Arduino Mega 2560 para ser possível a ligação dos pinos referentes ao eixo A.

Na figura 3.5 está representado o esquema do circuito elétrico da CNC.

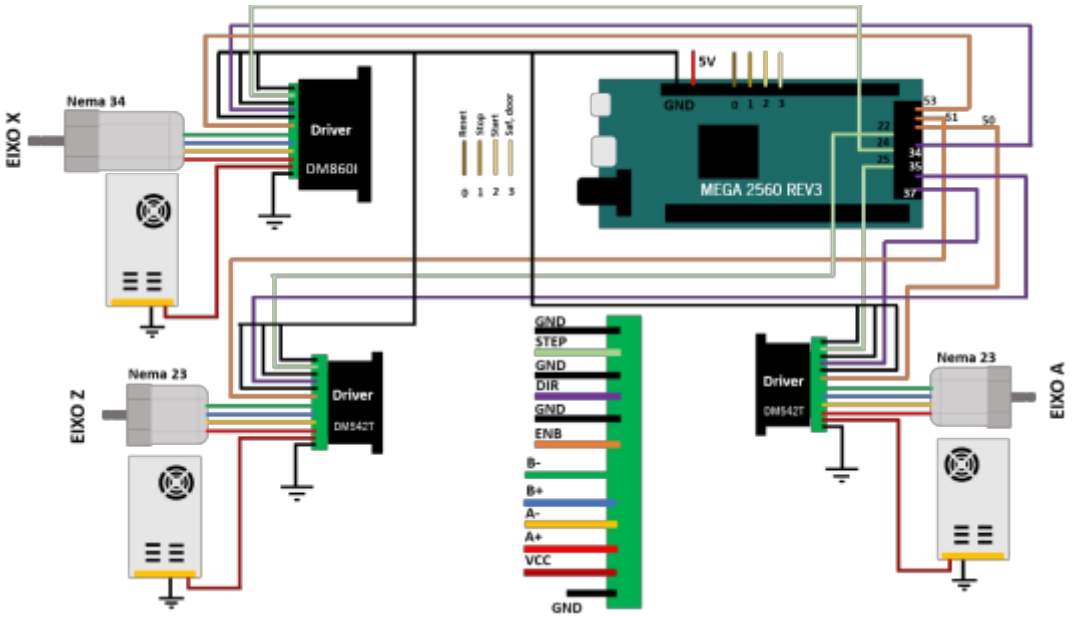


Figura 3.5 - Circuito elétrico da CNC.

3.2. Estrutura da CNC da Padrão Ortopédico

Existem centenas de tipos de próteses e ortóteses no mercado, não evidenciando que cada projeto é um projeto e está estritamente condicionado pelo paciente que posteriormente irá utilizar a peça. Quer isto dizer que, os tamanhos das próteses e ortóteses e, por consequência direta, os seus moldes, podem ter dimensões completamente distintas umas das outras.

Posto isto, é de notar que a área de trabalho da máquina, representada na figura 3.6, tem de satisfazer todas as dimensões necessárias para conseguir fazer o maior número de dimensões de moldes diferentes.

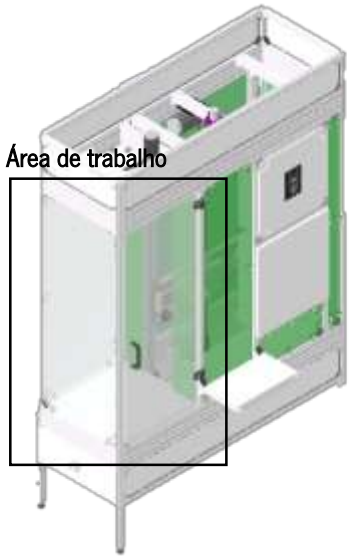


Figura 3.6 - Área de trabalho da Máquina CNC "Nada" (Silva, 2021).

Como a maioria dos moldes são gerados por revolução e, como essa revolução está incutida na peça a maquinar verticalmente, o percurso vertical da máquina é o mais elevado.

Já o percurso da ferramenta horizontalmente é reduzido, tendo por base que, como é necessária a criação do bloco de matéria consoante o tipo de peça que se pretende maquinar, este será dimensionado para não sucumbir a remoções de material desnecessárias, resultando em longos percursos de profundidade que destabilizam as operações, perda de tempo de maquinagem desnecessário e desperdício material do bloco. Posto isto, é possível identificar que, o eixo com maior movimento em todas as maquinagens é o eixo A (eixo de rotação). A este eixo, é possível fixar um bloco com dimensões até 400 mm de diâmetro.

Uma descrição mais detalhada dos componentes está representada no apêndice II.

A área de trabalho da máquina está compreendida na zona em forma de paralelepípedo identificada na figura. Corresponde a 900 mm de altura por 400 x 400 mm^2 de base. No compartimento restrito à maquinagem da peça, representado mais discriminadamente na figura 3.7, a *spindle* com potência de 2200 W é programada para girar a uma dada velocidade, dependente do material a cortar que, promove a rotação da ferramenta de corte diretamente ligada. À *spindle*, está diretamente fixada um porta-ferramentas onde é colocada a ferramenta de corte.

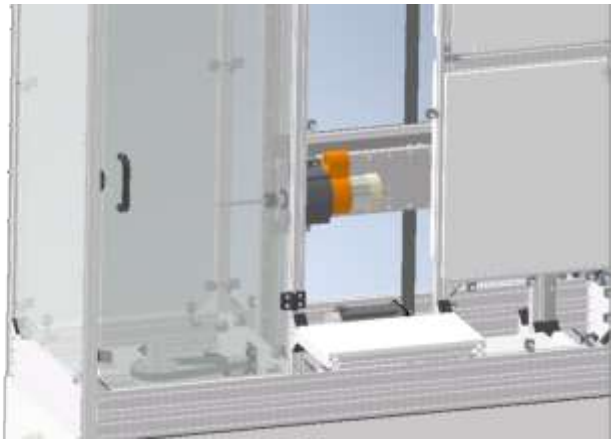


Figura 3.7 - Modelação 3D do sistema de transmissão de movimento do eixo de rotação A (Silva, 2021).

O bloco a maquinar encontra-se afixado a um prato que, por sua vez está acoplado ao eixo de rotação A, representado na figura 3.8. Este prato fica posicionado num veio paralelo ao veio do motor que promove a rotação da peça. A transmissão do movimento realiza-se através de um sistema duas polias e correia para diminuição da transmissão de potência.

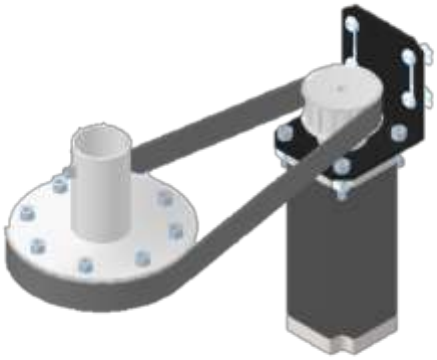


Figura 3.8 - Mecanismo de Transmissão de movimento do eixo A (Silva, 2021).

O veio que suporta o prato do bloco é afixado à polia, de 34 dentes, do sistema. O movimento é concebido a partir da transmissão através da correia de passo de 10 mm que, por sua vez, está ligada ao pinhão de 14 dentes (ambas as polias com passo de 10 mm) fixado ao veio do motor NEMA 23 (ver especificações no apêndice II). Estes motores, motores de 200 steps com um ângulo de passo de 1,8º, têm como

unidade de momento 3 Nm (Newton metro) e funcionam a uma tensão de 4,2 V com corrente de alimentação de 3,5 A.

Um movimento semelhante é realizado no topo da área de trabalho, representado na figura 3.9, com vista a que, no caso de blocos de material maiores, seja possível uma dupla amarração da peça, evitando oscilações da mesma.

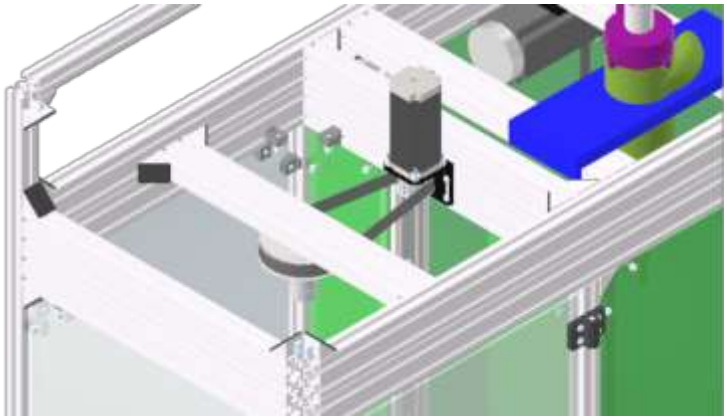


Figura 3.9 - Motor superior do movimento de rotação A (Modelação 3D) (Silva, 2021).

Para ambos os sistemas, a relação de transmissão, i, corresponde a, aproximadamente, 2,429 (rácio entres os 34 e os 14 dentes).

O eixo de movimento X, que promove o movimento vertical da máquina, representado na figura 3.10, é constituído por dois motores NEMA 34 que, tal como os NEMA 23, possuem 200 steps a passo com um ângulo de 1,8. Estes motores, com as especificações evidenciadas no apêndice II, funcionam a uma tensão de 5 V e corrente de 5 A. Com duas polias inseridas nos veios de cada motor e ligadas por uma correia. As polias, ambas com 14 dentes, possuem um passo de 10 mm (tal como a correia).

Neste caso, os dois motores utilizados são para auxiliar o movimento, ou seja, como todo o sistema demonstrado na figura tem de ser movimentado aquando a spindle, a utilização de dois motores por

Figura 3.10 - Modelação 3D do sistema de movimento do eixo de translação X (vertical) (Silva, 2021).

parte do fabricante ajuda a que a força seja distribuída por ambos os veios, evitando erros de translação causados por vibrações ou excesso de força sofrida pelo eixo que leva os motores a saltar passos.
Por fim, o eixo do movimento Z é conseguido através de um fuso de esferas de dois milímetros de passo e curso de 500 mm, acoplado a um motor NEMA 23 (idêntico aos do eixo A – ver apêndice II), que impulsiona a translação horizontal da ferramenta, representado na figura 3.11.

Figura 3.11 - Eixo de movimento Z (Modelação 3D) (Silva, 2021).

3.3. Estudo e avaliação do fabrico de moldes positivos na CNC da empresa Padrão Ortopédico

Resumidamente, neste capítulo vão ser destacados os passos do processo de maquinagem de um molde na CNC existente na empresa Padrão Ortopédico antes de se realizarem alterações na máquina.
Neste caso, a peça final a ser gerada é o membro inferior de um utente (do joelho ao tornozelo).
Como todos os processos de maquinagem, é necessário começar por criar o ficheiro CAD do objeto final. Nesta empresa é utilizado um *Scan* 3D ("Einscan Pro +") que, através da projeção de feixes de luz, calcula a distância entre pontos pela deformação do padrão projetado na perna do utente. Ao longo do procedimento começa-se a gerar, no *software* exclusivo do dispositivo, uma superfície (oca) com o relevo da perna do utente, como se pode observar na figura 3.12. No fim, o ficheiro é guardado como ".stl"[4].

[4] ".stl"- formato de ficheiro relativo à linguagem de estereolitografia, criação de modelos tridimensionais computorizados.

O segundo passo remete para o tratamento do ficheiro, por exemplo, no caso dos capacetes para pacientes de plagiocefalia posicional, é necessária a modificação do molde da cabeça como, aumentar as regiões de sensibilidade ou alinhamento do crânio, para isto, é utilizado o *software* "Solidworks".

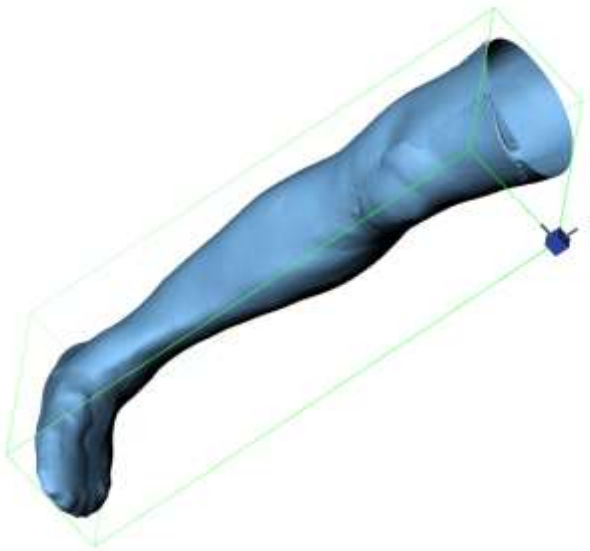


Figura 3.12 - Geração da superficie de relevo da perna.

É também necessário transformar a "casca" gerada por *scanner* num objeto sólido, bem como alinhar as extremidades da peça (aquando a utilização de *scanner* os limites da perna ficam geralmente pouco delineados, cobertos de altos e baixos – ver figura 3.12). Para a concessão destas etapas, é usado o *software "*Meshmixer*"*, representado na figura 3.13.

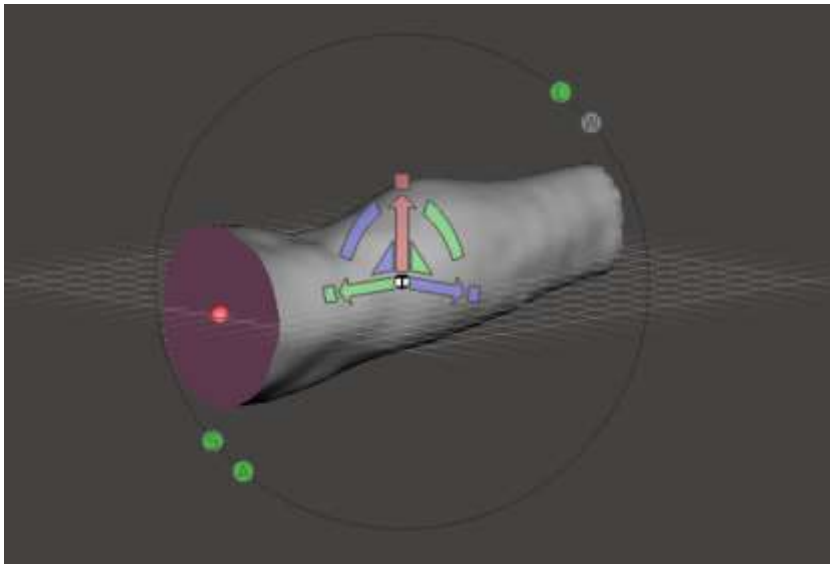


Figura 3.13 - Posicionamento do molde da perna no *software Meshmixer*.

O alinhamento da peça é executado no *software* "Meshmixer" onde, como observado na figura 3.13 é utilizado a extremidade maior do membro para definir a base do futuro bloco de maquinagem. O centro

dessa base está identificado com a esfera a vermelho na figura. Já o eixo a verde, terá de ser alterado para que corresponda ao eixo X (visto que a peça tem de ser maquinada verticalmente) e o eixo a vermelho, corresponde ao eixo do Z.

A máquina está dimensionada para que o bloco seja maquinado com uma amarração no eixo A que permite a rotação do mesmo.

Sendo peças por revolução, a altura da perna é definida pelo eixo X (vertical) e os diferentes diâmetros (relevo) pelo eixo Z (horizontal), eixo da ferramenta de corte, como representado na figura 3.14. Posto isto, o maior comprimento da perna tem de ser paralelo ao eixo X e perpendicular ao eixo Z.

Uma das etapas que também é indispensável, é o alinhamento da peça com o plano de corte que corresponderá ao"x0" ou à base do bloco. No caso da base se encontrar distanciada do ponto de referência, o *software* de geração do CAM assumirá essa distância como espaço a maquinar. Após finalizar todos os alinhamentos, é importante exportar o ficheiro como ".step"[5] para que este seja compatível com programa de CAM.

O programa que permite a geração do CAM é denominado por "Deskproto", neste caso, a versão 6.1. Para este tipo de máquina é apenas possível utilizar esta versão ou versões mais recentes, permitindo a inibição do eixo Y e inserção de um eixo rotativo A.

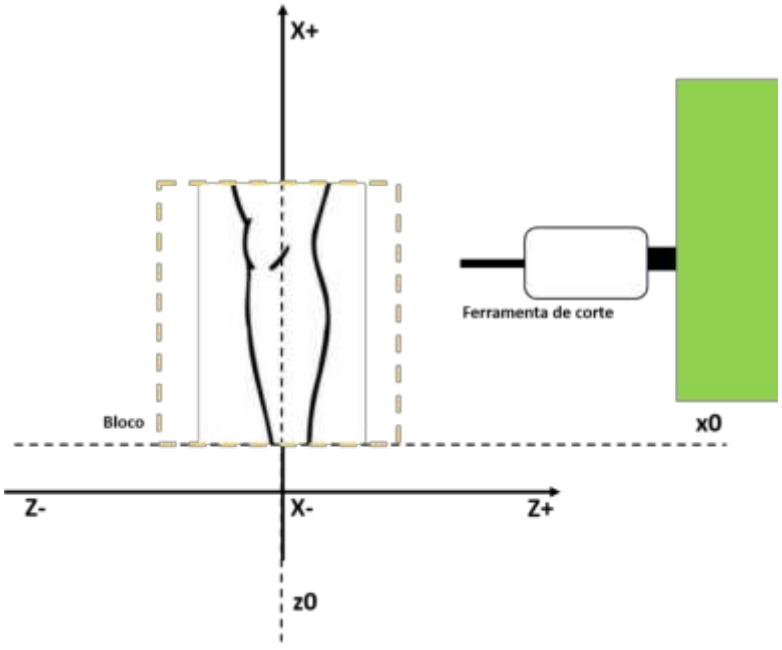


Figura 3.14 - Alinhamento dos eixos para a maquinagem em função das características da máquina.

Primeiramente, é fulcral definir a máquina onde vai ser maquinada a peça. Na figura 3.15, estão expressos os parâmetros utilizados para a definição do equipamento.

[5] ".step" – formato de ficheiro com linguagem padrão para o intercâmbio de dados de produtos industriais.

As características principais são o pós-processador ("Grbl"), a área de trabalho (X – 900 e Z - 400) que, neste caso, não é utilizada toda a região de trabalho que a CNC comporta, o diâmetro do porta-ferramentas, 20 mm, e a velocidade mínima e máxima de maquinagem (*feed rate*[6] de 0 a 700 mm/min). As características relativas à *spindle* também são, por norma, muito importantes (definem a velocidade corte) mas, neste caso, como a spindle está separada eletricamente da máquina, ou seja, é controlada manualmente, não será necessária a alteração destes parâmetros.

No que toca a erros de tempo o valor estipulado encontra-se no "1", ou seja, o tempo de maquinagem estimado pela máquina é igual ao tempo real de maquinagem.

O segundo passo a implementar no uso deste programa é a definição da ferramenta de corte. É possível observar na figura 3.16, os parâmetros remetentes à ferramenta utilizada na CNC. A ferramenta nomeada de "Carbide", tem uma extremidade esférica ("ball"), 150 mm de comprimento total, 100 mm de comprimento de corte e 10 mm de diâmetro de corte.

Para máquinas existentes no mercado, este processo é muito mais simples pois o programa contempla uma biblioteca de máquinas e ferramentas predefinidas. Todavia, este passo é executado apenas uma vez enquanto é utilizado o mesmo *software*.

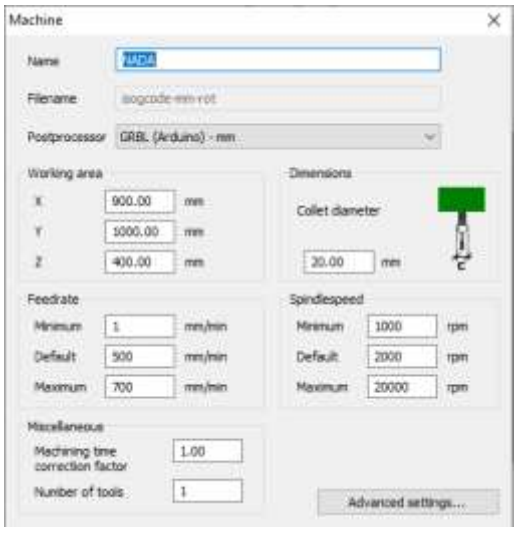


Figura 3.15 - Caracterização da CNC no programa *DeskProto*.

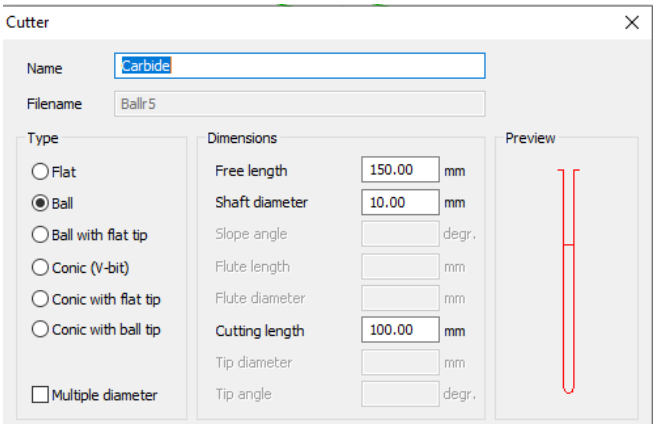


Figura 3.16 - Caracterização da ferramenta de corte no programa DeskProto.

Após a definição da CNC e da ferramenta de corte, passa-se então para as especificações da maquinagem em si.

[6] *Feed rate* é a velocidade de alimentação dos eixos da máquina, normalmente definida a partir do eixo da spindle dado que, este sofre mais momentos do que os restantes.

É inserido o ficheiro ".step" no programa que cria automaticamente uma "part". Ao editar, é possível escolher um nome para o projeto de CAM, a máquina a utilizar e, neste caso muito importante, se é para utilizar um eixo de rotação A ("XZA paths"), como exemplificado na figura 3.17. Ao colocar a utilização do eixo A, os parâmetros remetentes ao eixo Y são automaticamente desativados.

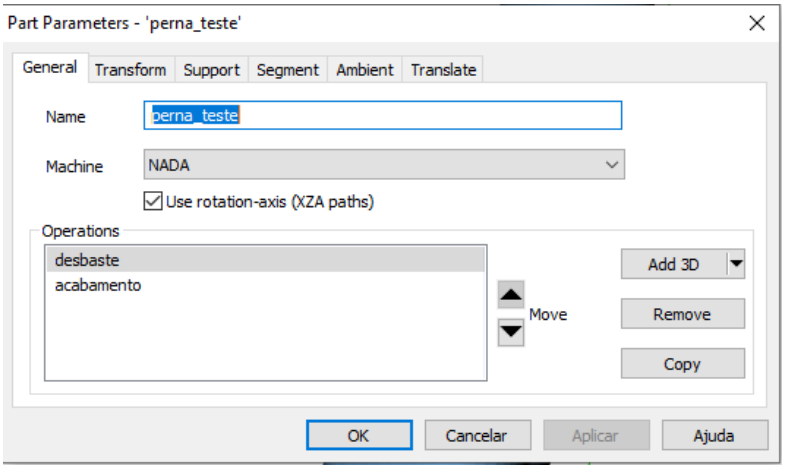


Figura 3.17 – Parâmetros gerais da peça no *software DeskProto*.

Outro aspeto importante é o dimensionamento do bloco a maquinar para que a ferramenta saiba onde começar e acabar o percurso, processo representado na figura 3.18. A altura do bloco definida é sempre a altura da peça a maquinar (402 mm – altura da perna), caso, na prática o bloco tenha uma altura superior, é apenas necessário marcar o zero peça mais acima (com a devida diferença de alturas) do zero do bloco real.

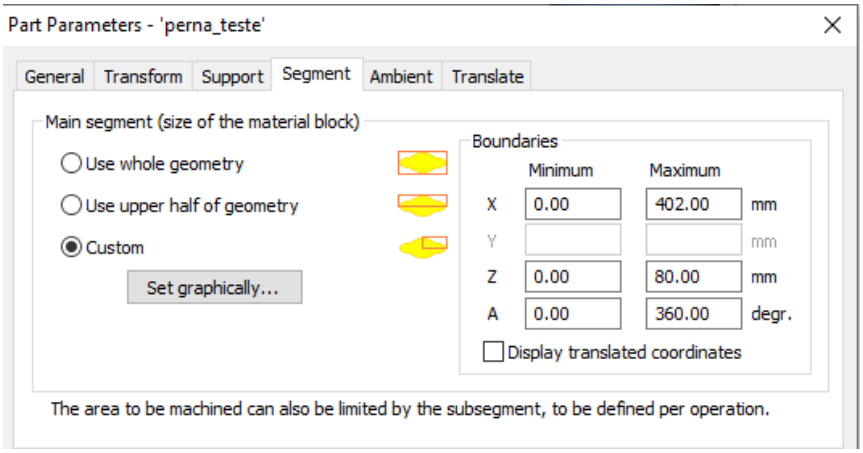


Figura 3.18 - Definição do bloco de maquinagem inicial no *software* DeskProto.

O movimento horizontal máximo, em Z, é metade do diâmetro máximo do bloco (sendo que a ferramenta nunca pode ultrapassar o zero que se encontra como centro da peça), nomeadamente, 80 mm. Por fim, o bloco é definido como cilíndrico com o eixo A a gerar trajetórias de 360º.

É ainda anulado o movimento de translação do eixo X relativo à peça, obtendo no final uma área de corte representada na figura 3.19 pela linha verde.

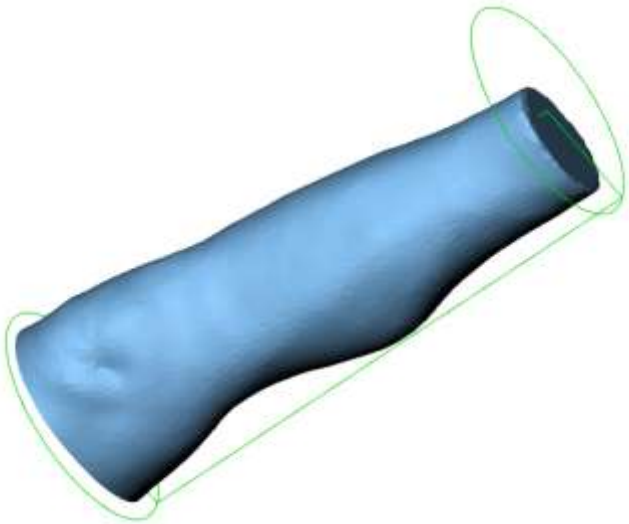


Figura 3.19 - Dimensionamento da área do bloco de corte.

Inicia-se a este ponto a definição de operações. São geralmente utilizadas duas operações, uma de desbaste e outra de acabamento da peça.

No que toca à operação de desbaste, este é realizado com a ferramenta editada "Carbide", uma distância entre passagens de 6 mm e um passo de 0.667 mm. A distância de passagens deve ser sempre inferior ao diâmetro da ferramenta, para evitar que, quando a profundidade de corte não atinge o diâmetro máximo da ferramenta, surjam zonas onde o material não é removido totalmente. A velocidade definida para o movimento dos eixos é de 700 mm/min, exemplificado na figura 3.20, promovendo um movimento paralelo da ferramenta e em torno do eixo de rotação A (como observado na figura 3.21).

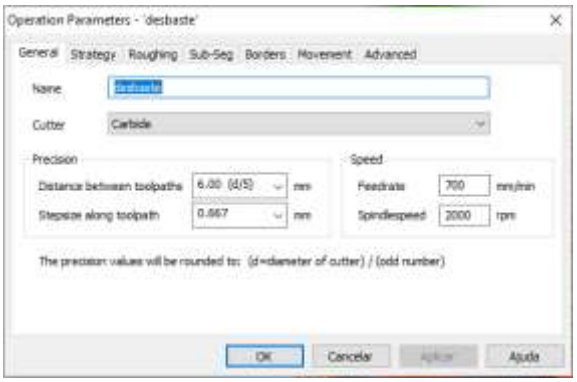


Figura 3.20 - Definições gerais da operação de desbaste.

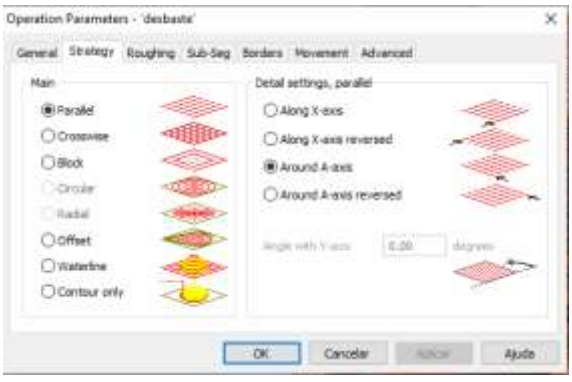


Figura 3.21 - Definições de estratégia do Desbaste.

Em cada camada de remoção do material a ferramenta atinge uma profundidade de 5 mm, deixando, no final desta operação (operação total de desbaste), uma camada de 10 mm para executar o acabamento, estes parâmetros estão enunciados na figura 3.22.

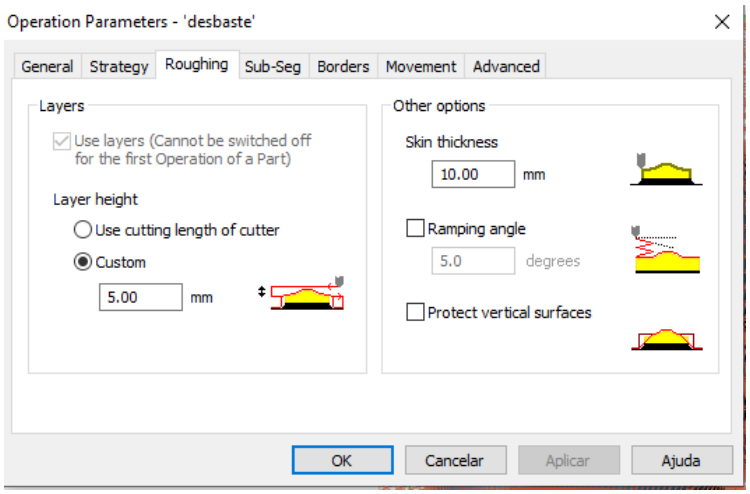


Figura 3.22 - Parâmetros de remoção de material.

Para finalizar a inserção dos parâmetros desta etapa, é adicionado o tipo de caminho que a ferramenta irá percorrer, nomeadamente, "*Meander*", ou seja, a ferramenta promove um caminho curvo horizontal de 0º a 360º, sobe 6 mm no bloco e retoma o caminho agora de 360º para 0º. Quando a ferramenta tem de se afastar da peça para esta girar em torno de A é dado uma distância de 5 mm entre o bloco e ferramenta, como estipulado na figura 3.23.

Ao chegar ao topo do bloco, a ferramenta desce até à posição x0 e retoma todo o processo até finalizar o desbaste por completo.

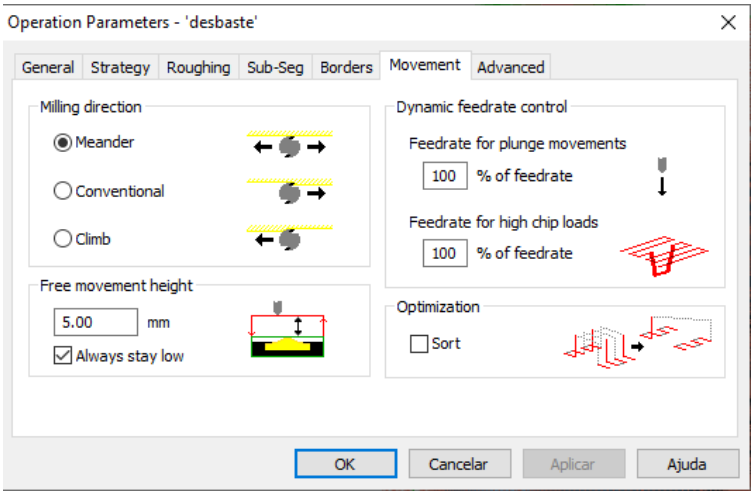


Figura 3.23 - Parâmetros de movimento da ferramenta da operação de desbaste.

Passando agora para a operação de acabamento, é de salientar que os únicos parâmetros de operação alterados de uma operação para a outra, estão representados nas figuras 3.24 e 3.25 e dizem respeito

à remoção de material, agora, paralela ao eixo X, ou seja, o material é removido verticalmente, e à camada que havia sido deixada propositadamente para esta operação que passa de 10 mm para 0 mm.

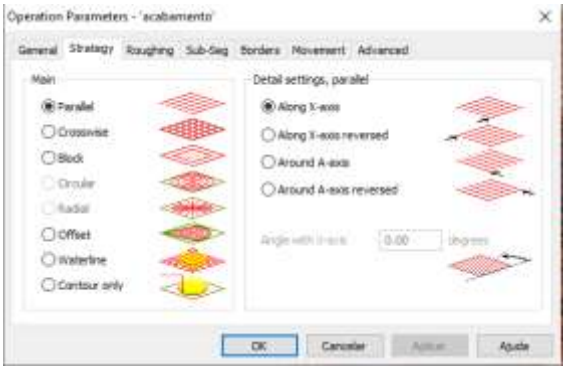


Figura 3.24 - Parâmetros de estratégia da operação de acabamento.

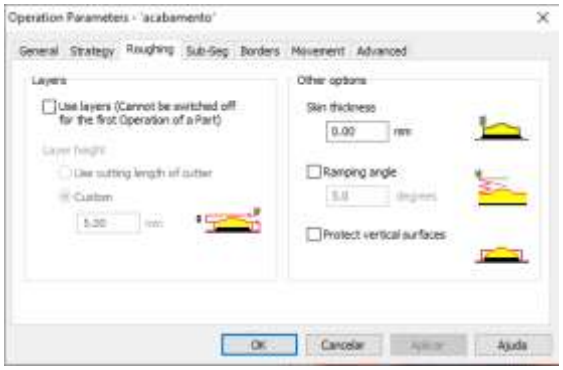


Figura 3.25 - Parâmetros de remoção de material da operação de acabamento.

Após as duas operações geradas, é possível visualizar o caminho da ferramenta em cada operação.

Na figura 3.26 está representado o desbaste do bloco e na figura 3.27 o tempo estimado para a concessão do mesmo. De notar que, este tempo pode não ser o executado pela máquina (dado que o fator de correção pode não ser igual a um). É ainda importante salientar que, na zona da base da peça, onde se encontra o joelho do membro, está programada apenas uma passagem de desbaste, devido à diferença mínima entre o diâmetro da peça e do bloco nessa área.

Como existe uma grande redução de diâmetro na parte superior da peça, gera-se uma remoção de material significativa o que, torna o processo mais moroso.

Figura 3.26 - Caminhos percorridos pela ferramenta na operação de desbaste.

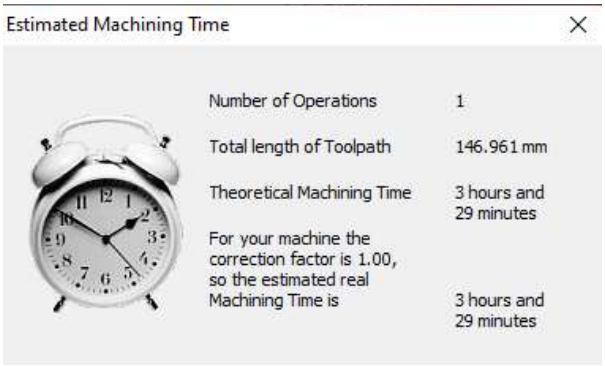


Figura 3.27 - Tempo estimado pelo "Deskproto" para execução do desbaste.

Como é possível observar, o desbaste do material é realizado em torno do bloco à medida que este gira no eixo A e a ferramenta vai aumentando o valor de X.

No acabamento a ferramenta exerce o movimento contrário, tal como descrito na figura 3.28, removendo material ao longo do eixo X do bloco até que a peça dê uma volta de 360º. Na figura 3.29 está o tempo que o programa estimou para o acabamento da peça.

Figura 3.28 - Percurso da ferramenta de corte para a operação de acabamento.

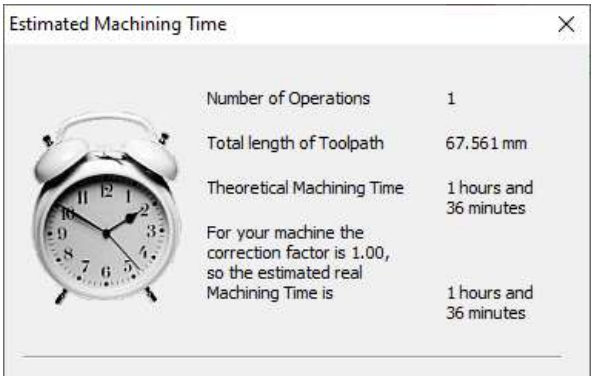


Figura 3.29 - Tempo estimado para execução da operação de acabamento.

Somando os tempos das duas operações, o tempo total de maquinagem estimado é de cinco horas e seis minutos. É um tempo considerável para um processo na verdade tão simples, porém, o nível de pormenor necessário na perna para posteriormente o técnico utilizar para fabrico de uma ortótese inibe a possibilidade de aumentar o passo do percurso e a velocidade da máquina (sendo que estas podem gerar vibrações aquando o movimento e penetração da peça).

Finalmente, é exportado o código G em formato ".nc" que é equivalente ao formato ".txt", exemplificado na figura 3.30, que, estará pronto para enviar para a máquina. Observa-se a diferença entre operações, sendo que, inicialmente é dado o comando à máquina de mover compulsivamente o eixo A 0.667 mm. Já no acabamento essa distância é movida no eixo X.

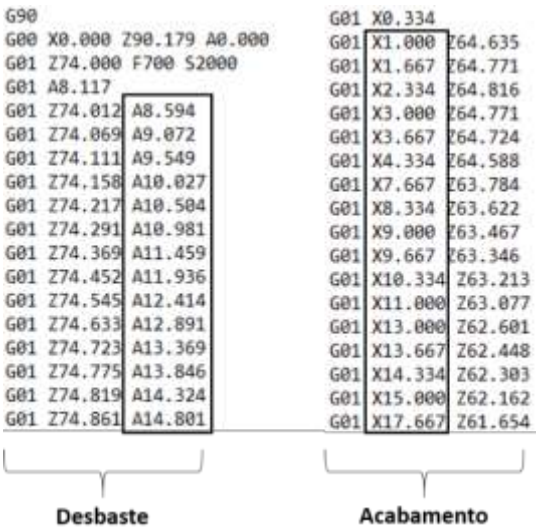


Figura 3.30 - Ficheiro de Código G da maquinagem da perna.

Seguidamente, é necessária a produção do bloco para a maquinagem. Nesta etapa, o bloco é fabricado pelo técnico da empresa num molde de PLA já existente. O molde é lentamente preparado devido à quantidade de parafusos que são necessários fixar. Também a diferença de dimensões, possível de observar na figura 3.31, provocadas pelo fabrico de blocos anteriores (devido à pressão dos gases aquando a reação), é uma dificuldade a ter em conta. Esta pressão também provoca diminuição da espessura das paredes do molde, aumentando o volume da peça.

Figura 3.31 - Diferença nas dimensões dos moldes devido às pressões da reação química.

Nos furos laterais do molde são colocados oito parafusos M6 com a respetiva porca, numa tentativa de suportar a pressão exercida.

Os moldes, com 20 cm de diâmetro interno e 22,5 cm de altura, são ainda fixados à base por oito parafuso, também M6.

A base do molde é também o prato, observado na figura 3.32, onde o bloco será posteriormente fixado enquanto é maquinado. É constituído por um veio de 3 cm de diâmetro externo que servirá de fixador ao eixo A, um prato com 24 cm de diâmetro externo e uma parte superior prismática que ficará no interior do bloco de forma a conceder-lhe estabilidade aquando a maquinagem.

Figura 3.32 - Prato de maquinagem (base do molde).

Após a todas as fixações do molde, é necessário o isolamento com fita adesiva de qualquer folga que exista (principalmente neste caso que os moldes demonstravam dimensões diferentes) e a inserção de um saco plástico no seu interior (que funciona como lubrificante de baixo custo para a posterior remoção dos moldes laterais).

Este objeto é fruto da reação química de dois produtos, nomeadamente, poliuretano de baixa densidade (200) e um endurecedor, representados na figura 3.33.

Figura 3.33 - Componentes para a formação do material do bloco.

Idealmente esta reação deveria ser promovida com um molde completamente fechado, porém, devido à diferença de alturas de cada parte do molde, não foi possível, pelo que, a densidade do material final é ligeiramente inferior à pretendida. É importante que seja medida a quantidade de produto a utilizar, convenientemente, a mesma quantidade de cada um.

Para a realização deste bloco foi utilizado um quilograma de cada reagente, sendo, como observado na figura 3.34, uma porção excessiva de utilização.

Figura 3.34 - Formação do bloco de poliuretano de baixa densidade.

Depois do técnico promover uma mistura homogénea destes reagentes, estes são vertidos para o molde, ocorrendo a reação.

Após a solidificação, são removidos os moldes laterais, cortado o excesso de material e aparafusados quatro parafusos na base com comprimento suficiente para entrarem no bloco (de modo a promover uma maior fixação).

Foram utilizados quatro parafusos para madeira (devido ao roscado ser maior o que, promove a amarração de mais material evitando que a vibração faça com que o parafuso seja removido gradualmente do bloco) com quatro centímetros de comprimento total.

Por fim, o bloco de 20 x 22,5 cm^2, está pronto (representado na figura 3.35). tendo a base uma altura de dez milímetros e os parafusos quarenta milímetros, seria de esperar que as maquinagens apenas ocorressem acima desse valor, ou seja, três centímetros acima da base do bloco, isto é, dependendo do quando a ferramenta desbastará ao diâmetro total da peça.

Figura 3.35 - Bloco fabricado na empresa de 20 x 22,5 cm.

É de notar que, neste caso específico, não foi utilizado um bloco realizado na empresa como em todas as outras maquinagens que já haviam sido realizadas, dado que a dimensão deste molde está preparada para maquinar "cabeças de bebé" e não uma altura de 400 mm de uma "perna". Assim, foi utilizado um bloco encomendado, exatamente com as medidas referidas anteriormente no processo de CAM. De igual forma, o bloco foi aparafusado ao prato rotativo com os mesmos parafusos de madeira M6, perfurando a base num diâmetro de três centímetros (ocupando 1,5 cm de distância de maquinagem do eixo Z).

Finalmente, é necessária a preparação da máquina. A primeira coisa a fazer é ligar a ficha da máquina à tomada, de seguida, é necessária a ativação da passagem de corrente para o sistema motor e controlador Arduino e, por fim, desativar o travão dos motores. Os botões responsáveis por estas funções estão representados na figura 3.36.

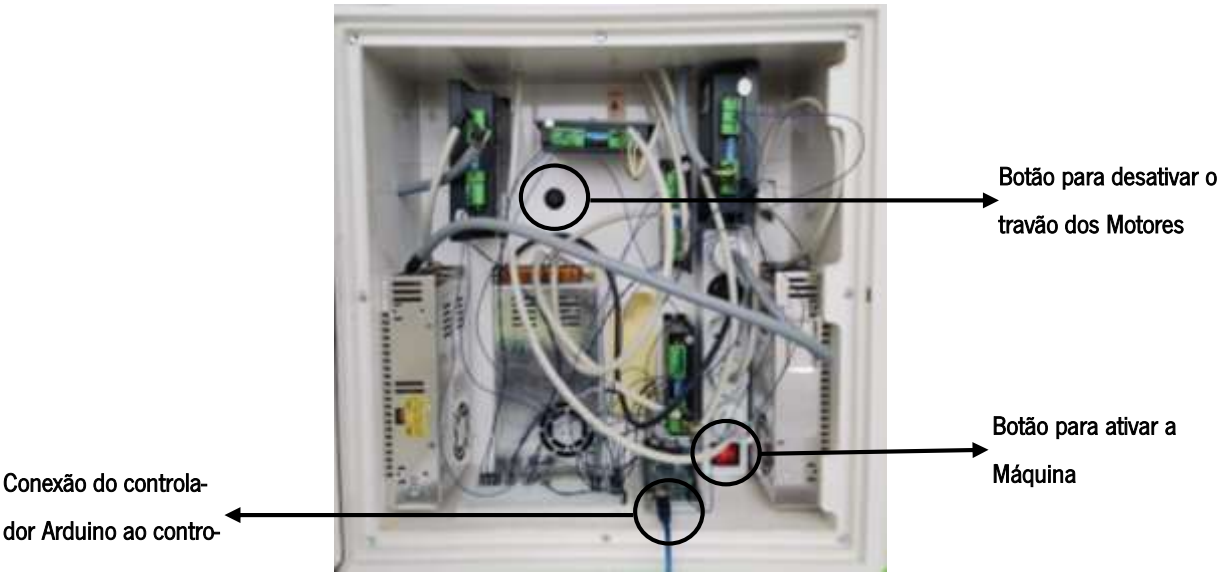


Figura 3.36. Botões responsáveis por ativação da máquina.

Após ligar a máquina é necessária a ligação do controlador à mesma através de um cabo USB 2.0 – USB. Neste caso, a ligação é para com um computador portátil, visto que o único controlador capaz, "Grbl Controller", só pode ser utilizado através deste tipo de dispositivo. A ligação fica então como representada na figura 3.36. Na figura 3.37 é possível observar o computador ligado à máquina.

Figura 3.37- Ligação do controlador (Grbl Controller) à maquina para ativação do canal de comunicação.

O “Grbl Controller”, *software* utilizado para complementar o pós-processador “Grbl” (incluso na máquina), tem como funcionalidade fornecer um canal de comunicação com a mesma. Para isso, é necessário compreender os conceitos gerais de código G (sendo que na empresa existe apenas um funcionário com tal conhecimento). Na figura 3.38 pode-se observar o menu geral deste *software*.

Antes de qualquer operação, a porta onde está inserida o controlador Arduino tem de ser ativada e o “Grbl” abrirá automaticamente o canal da máquina.

Figura 3.38 - Grbl Controller.

A próxima etapa tem de ser executada ainda sem o bloco inserido na máquina, isto é, é necessário definir o zero do eixo Z manualmente antes de colocar o bloco, isto porque, este é no centro do eixo A, como mencionado na figura 3.13. Para isso, comandos como “G91 Z-50 f500”[7] (coordenadas relativas) têm de ser enviados através da linha de comandos, observando quando a ponta da ferramenta encontra o centro do eixo A. Este processo seria realizado de forma manual, mais propriamente, “a olho”.

Para “zerar” o eixo X, é necessário agora colocar o bloco na máquina, já em posição de maquinagem. Tendo em conta a distância dos parafusos colocados na base do bloco, é definida uma altura qualquer como zero do X, desde que fique altura suficiente para a maquinagem de toda a altura da peça. Tanto o zero de Z como o zero de X, são definidos por “G92 Z0” e “G92 X0”, respetivamente. Após essa parametrização é possível enviar comandos em coordenadas absolutas (G90).

Para finalizar o processo, é ligada a *spindle*, como representado na figura 3.39.

A velocidade da *Spindle* não é definida diretamente, sendo que a gama de frequência que tem de ser imposta é de 85 Hz a 90 Hz, gerando a velocidade correta para a maquinagem deste tipo de material.

[7] Comando em linguagem G para movimentar a ferramenta em coordenadas relativas cinquenta milímetros na direção do eixo Z negativamente.

Depois de ligar a *Spindle* basta carregar o ficheiro com o código G para o Grbl e clicar em "Begin".

Figura 3.39 - Controlador da Spindle.

Aquando a maquinagem é preciso ter em atenção a necessidade de aspirar a máquina, como já foi mencionado, apesar dos componentes serem resistentes a pós, o material desbastado é tão fino que se penetra entre as correias e as polias, fazendo com que o movimento encrave.

Apesar do tempo estimado pelo programa "Deskproto" ter sido cinco horas e seis minutos, a máquina demorou cerca de seis horas a maquinar a peça, resultando no molde positivo de uma perna representado na figura 3.40.

Figura 3.40 - Molde positivo da perna do utente maquinado na CNC da Padrão Ortopédico.

3.4. Limitações da máquina CNC da empresa Padrão Ortopédico

Apesar da máquina de comando numérico encontrada na empresa ser eficiente na realização do seu propósito inicial, ou seja, na concessão de moldes positivos para capacetes de pacientes com plagiocefalia, este mecanismo, construído com um orçamento de apenas cinco mil euros, apresentava ainda algumas falhas no que toca a ergonomia e automatização do seu sistema.

A principal mudança requerida pela empresa seria a implementação de um terceiro eixo, de movimento linear (eixo do y), para tornar possível o fabrico de palminhas ortopédicas diretamente da máquina e a otimização do sistema de controlo, adquirindo a máquina uma interface gráfica amigável ao operador da mesma.

Também o facto de ser necessário, de todas as vezes que se pretendesse maquinar qualquer peça, disponibilizar um computador para manter ligado à máquina sendo assim possível a comunicação homem-máquina da mesma, era um inconveniente ao operador.

Outros problemas estariam ainda ligados à performance da CNC, nomeadamente, a necessidade de realizar sempre *homing* manual da mesma, bem como o zero peça.

Com o orçamento inicial reduzido, não foi também implementado, em primeira instância, um sistema de aspiração no mecanismo, resultando em acumulação de resíduos por toda a CNC.

3.4.1. Implementação do eixo do Y

Como já foi referido, a máquina em estudo possui dois eixos principais, um em torno de X (que promove o movimento vertical) e outro em torno de Z (que promove o movimento horizontal segundo o eixo da ferramenta de corte), como representado na figura 3.41.

A hipótese de implementar um eixo Y (eixo representado a amarelo) na CNC foi estudada e conferenciada com o orientador da empresa. Chegando à breve conclusão que, para além desta implementação acarretar custos elevados adicionais ao orçamento da máquina, poderia ser uma tarefa arriscada, dado que, idealmente, este movimento teria de ser realizado por todo o sistema mecânico (figura 3.41) já existente.

Acrescentando ao facto da CNC estar projetada para corte de um tipo de material diferente do material de maquinagem de palmilhas e, o facto de não ser eficiente utilizar uma máquina com uma grande área de trabalho para a concessão de peças pequenas (visto que não poderiam ser feitas mais do que uma peça de cada vez pois, as palmilhas são diferentes consoante o utente). Chegou-se à única hipótese viável e rentável para a empresa: adquirir uma CNC cartesiana de baixo custo para produzir apenas as palmilhas.

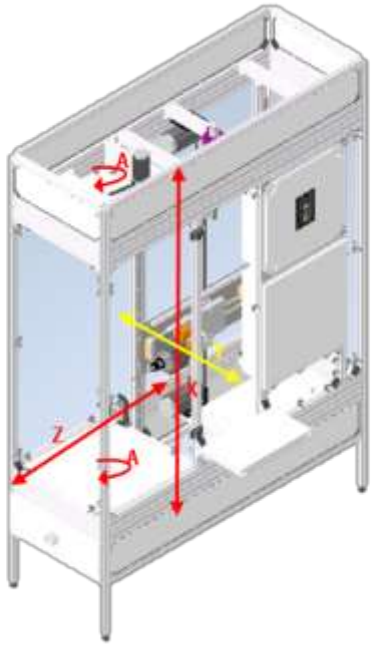


Figura 3.41 - Esquematização do eixo Y, em falta na CNC.

3.4.2. Implementação de interface homem-máquina

Este é o tópico abordado para esta dissertação, não só por ser um dos principais objetivos do estudo da CNC de comando numérico assistido por computador, mas também, pelo facto de que não seria uma CNC se não possuísse uma interface homem-máquina, capaz de controlar todas as varáveis intrínsecas ao sistema mecânico e eletrónico da máquina. Tal como está referido anteriormente, a interface é um dos três principais componentes de uma máquina de comando numérico assistido por computador, servindo como controlador da máquina.

No subcapítulo anterior e na figura 3.42, está exemplificado o que é necessário para comunicar com o a CNC, a ligação do computador à máquina tem vários riscos e falhas, dos quais: a necessidade de dispensar o aparelho por todo o tempo de maquinagem, possibilidade de dano devido à proximidade com a máquina, acumulação do pó do desbaste no aparelho, descarga de energia para o aparelho caso aconteça, não é comodo para o operador se instalar e operar a máquina.

Figura 3.42 - Ligação do computador à Máquina CNC.

O próprio *software* utilizado, "Grbl Controller*"*, é simples para quem não carece de conhecimento neste tipo de máquinas e linguagem, mas, para os técnicos da empresa, sem qualquer formação em máquinas industriais e CAM, torna-se complexo e até perigoso ter de enviar os comandos manuais através de código G, sendo que, aumentar uma unidade a uma dimensão pode resultar na perfuração do bloco ou até mesmo das paredes da CNC.

3.4.3. Automatização de operações de *homing* e zero peça

Estas especificações estão completamente dependentes do *hardware* e *software* da máquina já existente. É um problema relativamente simples que implica acrescentar sensores de posição e batentes na estrutura da CNC. A programação destes processos está completamente dependente do programa utilizado para controlo dos eixos.

Apesar de ser um entrave relativamente simples de resolver, é crucial ao bom funcionamento da máquina, evitando processos repetitivos desnecessários e aumentando a segurança do operador, bem como da própria máquina.

Como já foi referido, o zero do eixo Z e o zero do eixo X é posicionado manualmente, resultando em erros na peça e abrindo portas a um problema ainda maior, caso seja necessária a paragem e encerramento da máquina, estes terão de ser definidos novamente. Posto isto, se não existir um ponto de referência, a peça vai sofrer alterações na sua dimensão.

3.4.4. Implementação de sistema de aspiração

A CNC "Nada" foi concebida com equipamentos resistentes a resíduos sólidos, nomeadamente, ao pó libertado com o corte da espuma para o fabrico de moldes (situação representada na figura 3.43).

Porém, o excesso de resíduo no momento da maquinagem promove erros de corte bem como a danificação cíclica persistente dos componentes da máquina, para não falar da rutura total do movimento de eixos assim que o pó penetra entre a correia e a polia. Apesar de até os motores apresentarem durabilidade para este tipo de desgaste, a área de trabalho da máquina onde se encontra a peça e a área onde se encontram os mecanismos de transmissão de movimento estão interligadas, sem meio de separação.

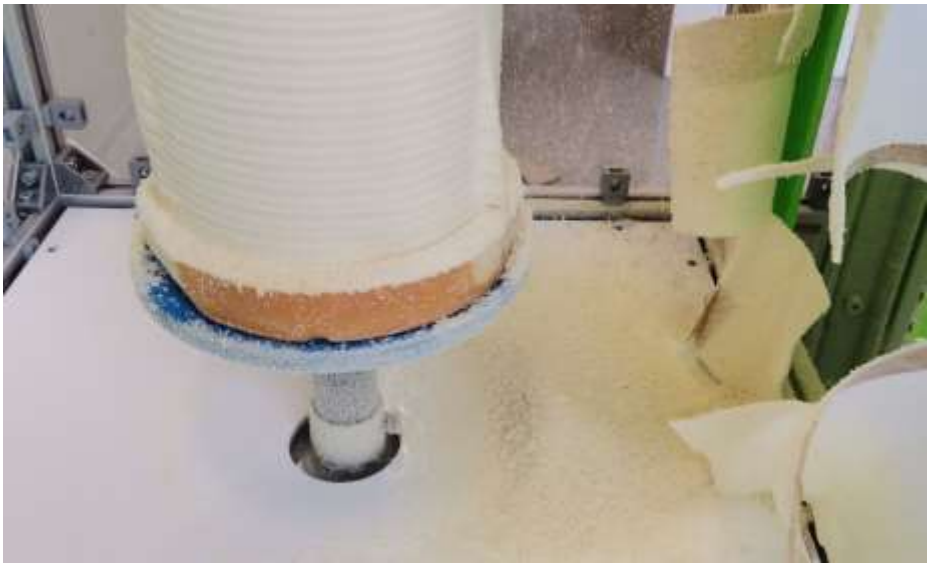


Figura 3.43 - Deposição do pó do desbaste na CNC e nos seus componentes.

Seria, assim, necessária a projeção de um sistema de aspiração automático para a máquina. Este tema foi tratado, paralelamente, na dissertação de mestrado do aluno Ivo Lopes Silva.

Outro problema associado ao pó é a falta de separação das partes da máquina. Estas encontravam-se separadas por folhas de papel fixadas umas às outras, porém, apesar de provavelmente ter resultado aquando as primeiras maquinagens, o papel foi ganhando forma até se encontrar completamente aberto, como pode ser observado na figura 3.44.

Figura 3.44 - Divisão da área de trabalho e da área mecânica da CNC.

3.4.5. Instabilidade da Máquina

Apesar do material não exercer vibrações elevadas aquando a maquinagem, existem sempre vibrações, seja do movimento dos eixos ou da *Spindle*. A máquina está suportada por quatro vigas de alumínio, representadas na figura 3.45.

Figura 3.45 - Vigas de alumínio que suportam a CNC.

Após algumas maquinagens, constatou-se que, quando são executadas penetrações de 10 mm, por exemplo, como a força que a máquina promove é superior, existe uma vibração considerável da máquina que, em peças com uma altura superior, pode gerar dimensões não desejáveis.

3.4.6. Modificação dos Moldes e Prato Rotativo

Outra limitação encontrada, relativa á máquina, porém não associada ao sistema mecânico da mesma, foram as deformações dos moldes onde são produzidos os blocos de poliuretano de baixa densidade. Este material exerce pressão sobre as paredes internas do molde e do prato de fabrico que, fabricados através da impressão 3D de PLA, forçam o enchimento das placas de plástico a moverem-se, alterando as dimensões das mesmas, como representado na figura 3.46.

Figura 3.46 - Deformações causadas nos moldes de PLA pela pressão do poliuretano de baixa densidade em reação.

4. Interface gráfica para automatização da máquina CNC

Esta interface foi desenvolvida com o intuito de conceder ao utilizador a possibilidade de utilizar a máquina de uma forma mais ergonómica, não sendo necessário o uso de um computador externo à CNC. Uma das principais vantagens da criação desta aplicação foi a facilidade de utilização.

4.1. Objetivos da criação da interface gráfica

Um dos requisitos para a criação da interface homem-máquina assentava na dificuldade de encontrar um programa intuitivo para que qualquer utilizador da máquina conseguisse facilmente aprender a manobrá-la. Para não falar do facto de que, a maioria das interfaces são pagas e as restantes são desenvolvidas para sistemas operativos específicos como "Windows" e "Mac OS", impossibilitando a utilização de um equipamento mais pequeno e económico para a sua implementação no próprio ambiente da máquina.

Uma das razões principais para criar um programa de raiz foi o facto de poucos programas comportarem a utilização de mais do que três eixos de movimento.

Posto isto, as hipóteses seriam: implementar uma interface já existente, *open-source*, adquirindo um computador que ficaria apenas ligado à Máquina de comando numérico, continuar a utilizar uma aplicação, igualmente *open-source*, no computador do utilizador, tendo este de ficar conectado à máquina durante todo o processo de maquinagem, não podendo assim ser utilizado para outro dever. Adquirir um programa pago para permitir a sua implementação num dispositivo de menores dimensões podendo permanecer afixado à máquina ou, por fim, realizar uma interface simples, mas coerente que permita a utilização de equipamentos de menores dimensões para ser exclusivamente parte do processo de manuseamento da CNC.

De salientar que, a máquina, como já havia sido programada através do *firmware* "Grbl", seria um requisito principal que, qualquer que fosse a aplicação escolhida integrasse a mesma forma de comunicação, estando fora de questão a troca de *firmware* do controlador Arduino que levaria a custos adicionais não suportados pelo cliente.

Também a necessidade de alterações estruturais da máquina no caso de se permanecer com a utilização de um computador portátil, mesmo este sendo exclusivo à máquina, alterações estas consequentes do tamanho do equipamento seriam um defeito posto pelo cliente. Não esquecendo que a utilização de

qualquer tipo de computador convencional requeria o uso de aparelhos indispensáveis a este como rato e teclado, aumentando ainda mais a área necessária de trabalho.

Ficou concluído que, encontrar uma interface pré-existente no mercado seria uma hipótese inviável pois, como relatado, estas interfaces estão diretamente ligadas ao controlador que, por sua vez, está especificamente concebido para equipamentos do seu fabricante. Também a utilização contínua de um computador de trabalho seria ineficiente, sendo que um dos requisitos da empresa seria deixar de disponibilizar um computador portátil sempre que fosse necessária a interação com a CNC. Posto isto, a utilização de um *firmware* e *software* diferente ao já utilizado seria uma ideia descartável pelo facto destes programas apenas funcionarem em ambientes como "Windows" e "Mac OS", apenas encontrados em computadores fixos ou portáteis, para além de estarem, maioritariamente, disponíveis para manipulação de máquinas com três eixos de movimento. Em suma, a alternativa mais viável seria então a concessão de uma interface específica para a máquina CNC.

Esta interface foi criada, principalmente, para possibilitar a qualquer tipo de utilizador o manuseamento da CNC fabricada de raiz na empresa Padrão Ortopédico. Na realidade, esta interface é apenas uma "conversa" com o "Grbl" *firmware* existente na máquina. O funcionamento é trivial, resumindo-se este em diferentes caracteres, ou conjunto de caracteres, enviados para a aplicação do controlador Arduino que, ao ler o pacote de dados recebidos processa essa mesma informação executando o comando que lhe é atribuído.

O próprio Grbl já provém de uma ampla biblioteca de funções e parâmetros que torna o processo de comunicação muito mais auxiliado.

Como revisto nos problemas associados à máquina preexistente, o equipamento não possuía uma comunicação direta com o operador, tendo de se utilizar meios externos para aceder à máquina.

4.2. Seleção do controlador a implementar na CNC

O *firmware* "Grbl" incluso na CNC é suportado pelo *hardware* de um Controlador Arduino Mega 2560 R3 que promove a comunicação com os drivers dos motores da máquina. Como, em parte, o controlador é escrito em "C++" (linguagem de programação sensível ao Controlador Arduino), seria de esperar que, para uma melhor eficiência de comunicação, a interface também deveria ser programada segundo o mesmo critério. Deu-se, assim, início ao estudo da hipótese da implementação de um LCD tátil ao controlador Arduino encontrado na CNC, como o representado na figura 4.1.

Figura 4.1 - TFT *touch* 3.2" - controlador arduino mega *shield* (BotnRoll, 2020b).

O *shield* Controlador Arduino 3.2 "TFT LCD é um *display* TFT multicor compatível com Controlador Arduino Mega com ecrã tátil e compartimento para cartão SD. O driver TFT é baseado em SSD1289[8] com dados de 8 bits e controlo de 4 bits para a interface (BotnRoll, 2020b).

O primeiro entrave remete para a impossibilidade de utilização de *shield* no controlador Arduino com o *firmware*. Todos os pinos do dispositivo necessários para a conexão do *display touch* são utilizados para enviar e receber informação dos drivers do motor. Por outro lado, os controladores Arduinos apenas conseguem suportar um programa por aparelho, ou seja, implementar um *software* para a interface não seria possível, de todo, no mesmo controlador Arduino.

Para ser possível a visualização dos dados recebidos pelo "Grbl" e o envio de informação do utilizador para o *firmware*, existe uma ligação em série entre uma interface e o controlador Arduino. As ligações em série são

executadas através de pinos de receção e envio, Rx e Tx (figura 4.2), existentes no dispositivo ou, através de um cabo de dados conectado a um computador. Neste último caso, os dados enviados e recebidos podem ser visualizados no painel de série no controlador Arduino IDE.

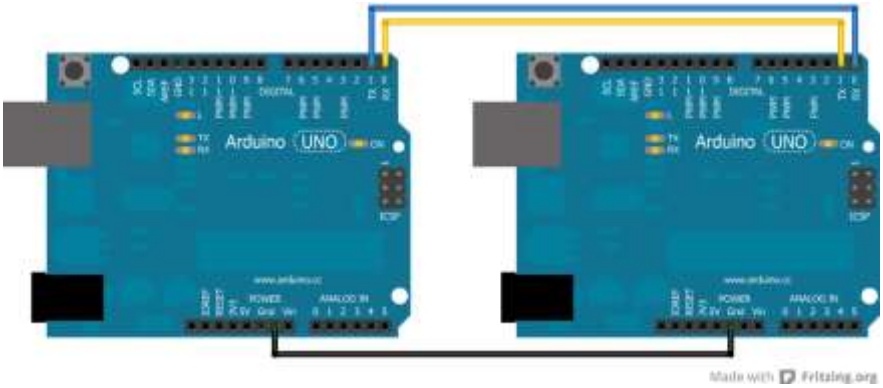


Figura 4.2 - Comunicação em série entre dois controlador Arduinos (Case, 2003).

[88] Referência para um ecrã com driver e RAM incluídos num só sistema.

Os controladores Arduinos promovem uma ligação "*master-slave*", ou seja, um controlador Arduino mestre envia ordens para o controlador Arduino escravo e este responde, isto se os programas intrínsecos a estes forem correspondentes.

Neste caso, como representado na figura 4.3, o projeto insidia em programar uma interface tátil com botões capazes de enviar códigos específicos da linguagem do "Grbl" para o controlador Arduino, por sua vez, este seriam guardados no buffer para serem enviados por comunicação em série para o controlador Arduino com o *firmware* que, após a ordem, processaria a informação dada e enviaria uma resposta.

Por fim, o problema em utilizar o controlador Arduino como fonte de comunicação deve-se à necessidade de mais do que dois pinos de série, ou seja, o controlador Arduino *Master* teria de ter dois pinos para comunicar com o ecrã e dois pinos para comunicar com o *slave*. Posto isto, estudou-se a possibilidade de utilizar outra forma de comunicação direta entre um programa de interface e o "Grbl".

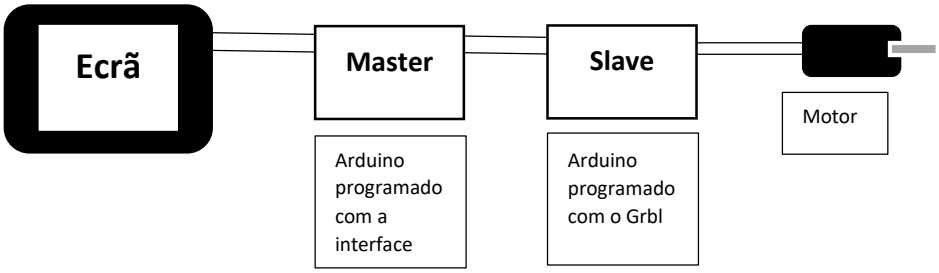


Figura 4.3 - Esquema de comunicação entre a interface, controlador Arduino, Grbl e máquina.

O Raspberry Pi é um computador de baixo custo e dimensões. Capaz de se conectar a um computador ou televisão, teclado e rato. Como os sistemas operativos para este dispositivo são desenvolvidos em "Python", é possível correr programas em "Scratch" e "Python", garantido a compatibilidade. Através dele, é viável realizar qualquer tarefa que um computador fixo ou portátil faria (RASPBERRY PI FOUNDATION, n.d.).

Existem vários modelos deste produto no mercado sendo que, o seu preço varia dependendo das características, tais como, memória RAM, dimensões, capacidade de conexão, número de entradas de conexão de *hardware* externo (RASPBERRY PI FOUNDATION, n.d.).

O Raspberry Pi utilizado para a implementação do *software* da interface foi o modelo Raspberry Pi 4 "Model" B (Pi4B). Este é o último modelo lançado pela empresa que é possível adquirir com um ecrã tátil compatível com *hardware*. Este aparelho comporta mais RAM do que os anteriores e o CPU, GPU e o sistema operativo foram melhorados em comparação com os dos modelos anteriores (Raspberry Pi (Trading) Ltd., 2019a).

Este dispositivo, representado na figura 4.4, comporta um sistema Quad core 64-bit ARM-Cortex A72[9] (que corre a 1,5 GHz). Neste caso, com 4 Gigabytes de LPDDR4[10] RAM e suporta duas ligações HDMI (para a conexão com dois ecrãs). Compreende ainda uma entrada para cartão SD e quatro entradas USB (Raspberry Pi (Trading) Ltd., 2019a).

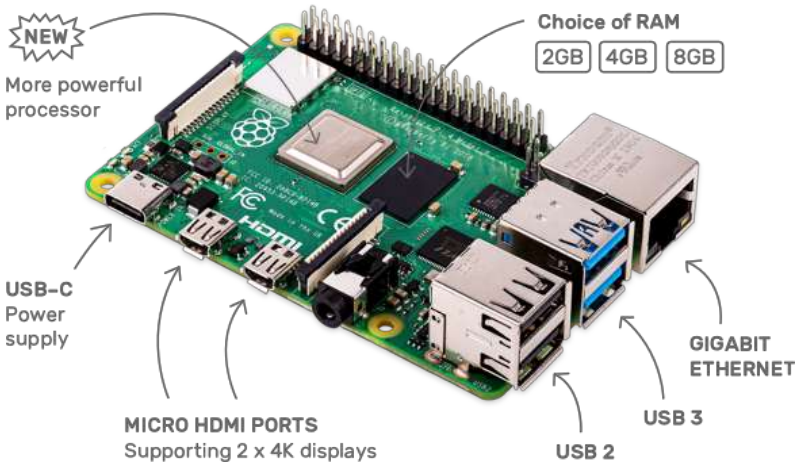


Figura 4.4 - Raspberry Pi 4 "Model" B (Pi4B) (Raspberry Pi (Trading) Ltd., 2019a).

O ecrã tátil compatível com o Pi4B (figura 4.5) é um *display* LCD onde, através da conexão pelo terminal DSI[11] é possível a troca de informação entre os dois dispositivos.

Figura 4.5 - Raspberry Pi Touch *Display* (Raspberry Pi (Trading) Ltd., 2019b).

É importante saber que, o Raspberry tem a capacidade, quando suportado por *software*, de estabelecer ligação entre o LCD e um ecrã (HDMI) em simultâneo, sendo assim, muito mais fácil para um programador desenvolver o código no seu computador habitual e testá-los simultaneamente no ecrã do Raspberry.

[9] Sistema de arquitetura de sistemas em *pipelines* que promove a maior rapidez de execução de tarefas, na medida que, podem ser executadas várias funções simultaneamente e não, esperar que uma tarefa termine para começar a outra.
[10] Memória com 32 bits e dois canais de 16 bits cada um para dividir os dados de passagem e diminuir a potência necessária por unidade de tempo.
[11] Ligação em série de um ecrã ao respetivo *shield*.

A instalação destes equipamentos é relativamente simples, é apenas necessária a conexão o cabo de dados e os fios elétricos de energia (preto e vermelho), como é demonstrado na figura 4.6.

Este ecrã tem uma resolução de 800 x 480 pixéis, segundo o sistema RGB (coloração de 24-bit). É um ecrã *multi-touch*[12] de 10 pontos. As suas dimensões, externa e de visor são, respetivamente, 192.96 × 110.76 mm^2 e 154.08 × 85.92 mm^2 (Raspberry Pi (Trading) Ltd., 2019b).

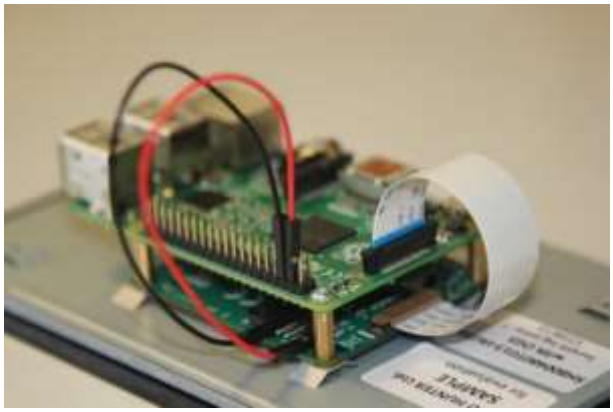


Figura 4.6 - ligação do Raspberry Pi4B ao ecrã tátil (Raspberry Pi (Trading) Ltd., 2019c).

O custo deste conjunto de equipamentos é, aproximadamente, cento e cinquenta euros, dependendo do fornecedor.

Na tabela 4.1, estão representadas algumas especificações do controlador implementado com controlador Arduino e do controlador através do Raspberry Pi.

Tabela 4.1 - Especificações gerais dos dispositivos: Controlador Arduino 2560 R3 e Raspberry Pi 4 model B

(dados agregados de "Controlador Arduino Mega 2560 R3" retirados de (BotnRoll, 2020a); dados agregados de "Raspberry Pi 4" retirados de (Raspberry Pi (Trading) Ltd., 2019a))

Dispositivo	Controlador Arduino Mega 2560 R3	Raspberry Pi 4
Linguagem de programação	C++	Python, C, C++
Tipo de fonte de alimentação	Bateria, Transformador	Transformador (tipo C)
Diferença de Potencial	3.3 V ou 5 V	5 V
Corrente necessária	Corrente por pino de 40 mA	~2500 mA
Ecrã compatível	TFT *touch* (até 3.5'')	Raspberry Pi touch *display* (até 10 '')
Modo de conexão com o ecrã	Aquisição de *Shield*	Vem implementado com *Shield*
Modo de conexão com controlador Arduino da máquina	Pinos Rx e Tx	USB 2.0 – USB B

[12] Permite clicar no ecrã em vários pontos ao mesmo tempo.

Tabela 4.1 - Especificações gerais dos dispositivos: Controlador Arduino 2560 R3 e Raspberry Pi 4 model B (cont.)

Dispositivo	Controlador Arduino Mega 2560 R3	Raspberry Pi 4
***Hardware* para leitura de dispositivos externos**	Aquisição de equipamentos externos para leitura de cartões SD, USB 2.0, USB 3.0, C, ...	Portas USB: 2 USB 3.0; 2 USB 2.0
Preço do dispositivo	37, 90 €	41,95 € (2 Gb)
Preço do ecrã	21,50 € (3.5 ‘’)	82,00 € (7 ’’)
Preço total	50, 40 €	123,95 €

Olhando a um lado empreendedor, é notável a diferença de preços existente para cada uma das aplicações (preços correspondentes à loja de revenda eletrónica “Bot n Roll”), porém, se for somada a quantidade de equipamentos externos que teriam de ser adicionados se fosse utilizado o Controlador Arduino Mega, como leitor de cartões SD (sensivelmente cinco euros), um *display* de maiores dimensões (com preço aproximadamente de cento e vinte cinco euros para um ecrã de dez polegadas) que, por sua vez, requereria um novo *shield* para implementar no controlador Arduino, sendo possível a ligação de todos estes equipamentos externos, o custo deste sistema seria muito menos vantajoso, tanto a nível económico como de complexidade de projeto.

O facto do Raspberry, ao contrário do controlador Arduino, conter 4 entradas para ligação de dados, facilita a conexão do dispositivo a qualquer tipo de equipamento externo, seja o controlador Arduino incluso na máquina ou um dispositivo para passagem de código G.

No Mega, o LCD tem de ser implementado através de um *shield* (à parte para ecrã maiores) que ocupa os pinos de receção do controlador Arduino, sendo necessário utilizar outros pinos de receção para ligar o dispositivo ao controlador Arduino da máquina.

Por último, em termos de linguagem, o Raspberry permite, através do seu *software* (*open-source*), executar programas escritos em várias línguas de código e simultaneamente controlar o *firmware* da máquina. Já o controlador Arduino apenas consegue executar programas escritos em C++ sendo que, cada dispositivo só pode incluir um só código.

Em suma, será utilizado para o projeto o sistema físico que engloba o Raspberry Pi 4 modelo B (2 Gb) e o Raspberry Pi *Touch Display.*

4.2.1. Programação do Raspberry Pi 4 modelo B e do ecrã tátil

A vantagem deste equipamento é que o seu sistema operativo é gratuito e fornecido diretamente pela empresa a partir da hiperligação "www.raspberrypi.org/downloads/". O sistema operativo "Raspbian" foi desenvolvido exclusivamente para estes computadores e deriva de uma versão do Linux. Para uma instalação mais simples, o fornecedor oferece uma aplicação que contém todos os ficheiros necessários para a passagem do programa para o dispositivo. Neste ponto, é necessário fazer download do "NOOBS", representado na figura 4.7.

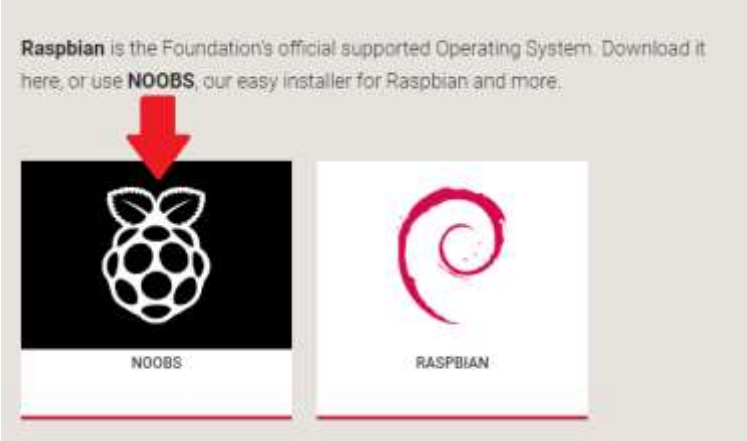


Figura 4.7 - Excerto do site onde é possível descarregar o NOOBS para proceder à instalação do RASPBIAN (Raspberry Pi (Trading) Ltd., 2020).

Após o download do ficheiro ".zip" do NOOBS, é necessário extrair todos os ficheiros nele contidos. Estes ficheiros serão passados para um cartão de memória que comportará o sistema operativo. De notar que, este cartão de memória tem de estar formatado[13]. Após transferir os ficheiros para o cartão SD, insere-se o cartão no Raspberry Pi4, já devidamente ligado ao ecrã. Na figura 4.8, está a representação do que deverá aparecer.

Figura 4.8 - Instalação do Sistema operativo presente no cartão SD no Pi4B (Raspberry Pi (Trading) Ltd., 2020).

[13] "File Allocation Table", ou seja, tabela de alocação de artigos onde ficheiros de imagem ou vídeo ficam num lugar fico e marcado, numa "tabela" para que, o dispositivo consiga aceder diretamente aos mesmos.

Ao clicar no botão "*Install*", o equipamento vai proceder à instalação automaticamente até aparecer um aviso que dita se a instalação foi bem-sucedida.

O ambiente do sistema operativo é, de origem, idêntico ao representado na figura 4.9.

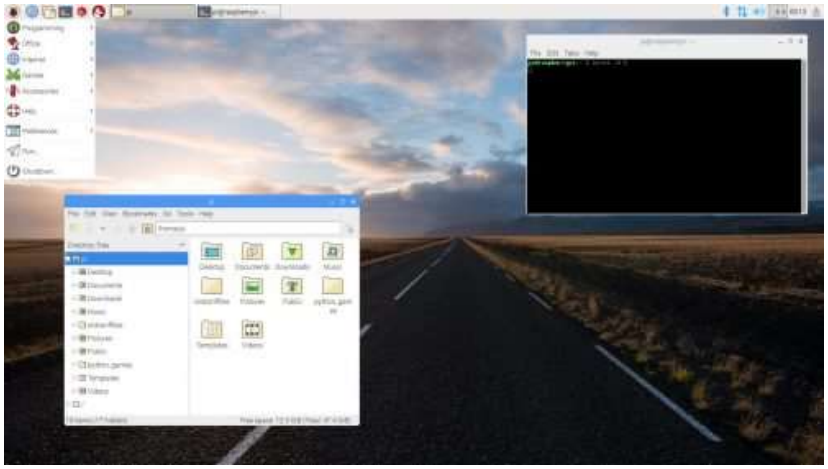


Figura 4.9 - Ambiente de Trabalho do sistema operativo Raspbian (Raspberry Pi (Trading) Ltd., 2020).

Como é possível observar trata-se, de facto, de uma pequena versão de um computador habitual de trabalho. Com uma estrutura idêntica aos ambientes *Linux*, é possível armazenar conteúdos em pastas, editar as preferências de utilizador e, como o *software* é programado em "Python", é possível executar programas em "python" diretamente da linha de comandos (isto se a versão do "python" estiver atualizada).

4.3. *Design* concetual e arquitetura da interface

O "Grbl" já provém de uma vasta biblioteca de funções e parâmetros que torna o processo de comunicação muito mais auxiliado (Breiler, 2020a).

Para a projeção da interface, o seu design teve de ir de encontro às especificações do *firmware* utilizado, ou seja, dependendo do que o programa permite ou não permite fazer.

A principal função da interface é permitir ao utilizador o envio do ficheiro com o código G para o controlador na máquina, todas as outras funções seriam adjacentes, ou seja, funções de preparação da máquina para a maquinagem e funções de informação enquanto a maquinagem da peça.

O *design* do *display* foi inspirado na disposição de informação da interface para computador do "Grbl Controller", representada na figura 4.10. Este controlador apresenta apenas um menu que permite a manipulação da máquina segundo as opções:

- Identificação da porta onde está ligado o controlador Arduino da máquina;
- Opção para fechar a porta;

- *Reset* do “Grbl”;
- Escolha do ficheiro e representação visual do diretório desse ficheiro;
- Botão para começar a enviar o conteúdo do ficheiro para o “Grbl”;
- Botão de paragem de envio;
- Barra de Progresso do envio do ficheiro para o “Grbl”;
- Tempo de trabalho da máquina (crescente);
- Painel para comunicação do “Grbl”;
- Janela para introdução de comandos a enviar ao “Grbl” (enviados através da tecla “Enter” de um teclado);
- Monitores de coordenadas (X, Y, Z) do movimento dos eixos;
- Painel de comando manual para três eixos;
- Botão para escolher o movimento dos eixos manualmente;
- Botão de ativação da *Spindle*;
- Botão para aceder às definições do “Grbl”.

Esta interface, inclui ainda um botão de ferramentas no canto superior direito que fornece ao operador uma série de opções do envio de informação e dos eixos da máquina.

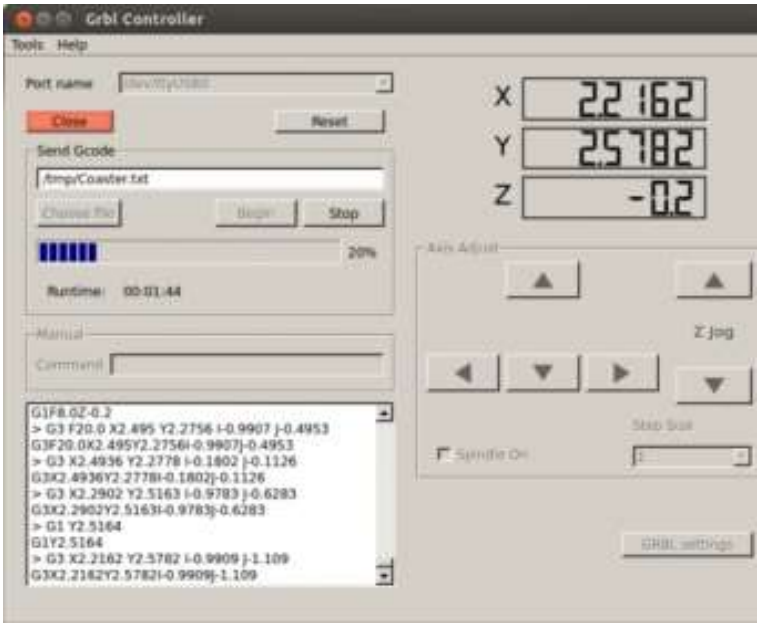


Figura 4.10 – “Grbl Controller” interface.

Como a nova interface foi concebida para facilitar o operador da sua utilização, ou seja, torná-la num ambiente amigável de comunicação entre o homem e a máquina, pretendia-se uma disposição de elementos mais acessível a utilizadores sem formação neste campo, por outras palavras, mais intuitiva na sua utilização.

Apesar desta interface apenas usufruir de um menu, como é utilizada em computadores de bancada ou portáteis, o operador teria sempre acesso a um teclado e rato, ou seja, seria possível, desta forma, escrever diretamente todos os comandos necessários de alteração da máquina sem que estes se apresentassem no menu. Por um lado, a estrutura torna o programa muito mais comprimido. Todavia, na nova interface pretendia-se que certas operações importantes estivessem ao alcance de um só botão, não havendo necessidade do utilizador "inexperiente" proceder à leitura dos códigos necessários para executar essa mesma função e enviá-los para o controlador, facilitando a interação com a máquina.

Posto isto, seria apreciável a inserção de todas estas funções na interface implementada. Ainda, algumas funções apenas possíveis a partir da digitalização de código, se exequível, deveriam ser substituídas por botões ou outro tipo de objeto que permitisse ao utilizador uma interação facilitada, sem necessidade de pesquisar, decorar ou escrever estes mesmos códigos.

É importante perceber que, realizar funções básicas como desbloquear o sistema, definir o modo de movimento ou selecionar os milímetros a mover o eixo a partir de um botão visível para o operador, em vez de utilizar um código escrito diretamente na máquina, pode não ser o processo mais simples e prático para fazer o pretendido. Todavia, para técnicos sem experiência nos comandos e no código universal, apesar de ter de selecionar mais botões, é muito mais fácil e intuitivo do que escrever linhas de código em si, evitando erros de escala (como, por exemplo, pôr uma casa decimal a mais numa medida) ou erros de movimento (tal como escrever o eixo errado a utilizar). Assim, ao clicar no botão, o operador vai ser obrigado a pensar no que quer fazer e efetivamente ver o que quer fazer.

Dividindo os tipos de funções, como representado na figura 4.11, haveriam então quatro grupos principais, sendo estes o "Envio de código e Maquinagem", "Manipulação Manual dos eixos", "Alteração das definições e parâmetros da Máquina" e "Comunicação Homem-Máquina". Esta repartição facilita o agrupamento de objetos aquando a organização visual da interface.

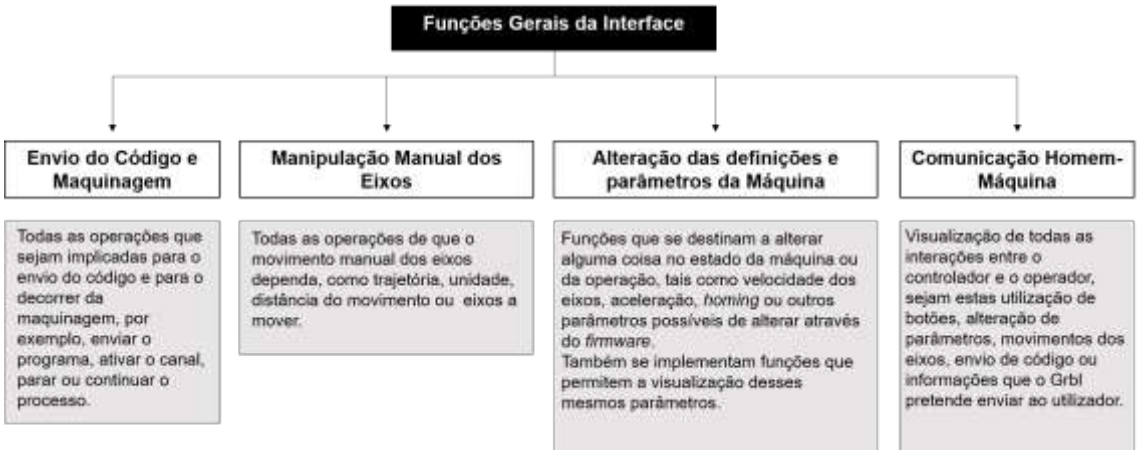


Figura 4.11 - Funções gerais a implementar na interface.

4.3.1. Funções relacionadas ao envio do código e maquinagem: ativação de comunicação, envio de ficheiros e visualização do estado de operação e da máquina

Como já foi referido, o principal objetivo da interface é o envio do código G para a máquina. Para isso, como representado no esquema da figura 4.12, são necessárias duas tarefas principais: ligar a interface à máquina (ao controlador) e escolher o ficheiro a enviar. Seriam então, à partida, necessários três botões para cada uma destas funções.

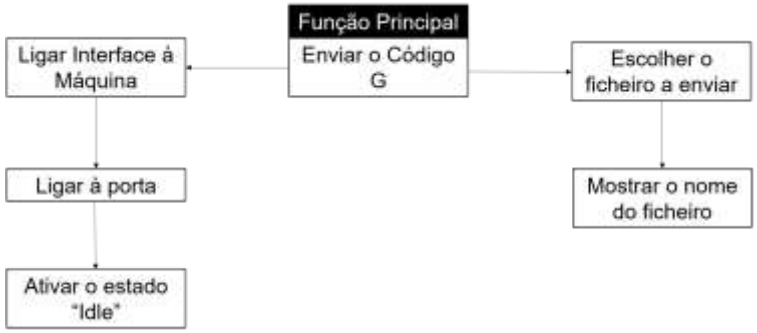


Figura 4.12 - Esquematização das funções necessárias para enviar o ficheiro de código para o controlador.

Olhando para a função "Ligar Interface à máquina", é possível observar que esta depende de duas subfunções. Na versão "Grbl Controller", existe um botão para "Ligar à porta" pretendida pelo utilizador. Porém, na interface projetada, este botão foi descartado, sendo que, o dispositivo está diretamente ligado à máquina, sempre pelo mesmo canal, possibilitando a programação de apenas uma porta que abrirá sempre que se der início ao programa.

Por outro lado, ao abrir o "Grbl Controller", o programa dá automaticamente uma ordem de "Idle" à CNC. No caso da nova interface, este botão tem de ser ativado, por modo de segurança (assim, mesmo que, por engano, o utilizador promova o movimento de algum eixo antes do pretendido, este não se moverá, mantendo a máquina e o utilizador em segurança).

Ter-se-ia, assim, apenas um botão para ativar o estado "Idle" da máquina sendo que, a interface está eletricamente ligada ao sistema da máquina (em série) e a porta também de encontra conectada.

Para a função "Escolher ficheiro", seria então necessário um botão que permitisse ao utilizador procurar o ficheiro pretendido e selecioná-lo. Por via de verificação da escolha, é plausível a perceção visual, pela parte do operador, do nome (ou diretório) do ficheiro escolhido. Assim, para a execução desta função, seriam então implementados um botão para escolher o ficheiro e uma janela onde apareceria o nome do ficheiro escolhido.

Passando agora a funções necessárias aquando a concessão da peça, ou seja, após o envio do programa. Neste campo, existem dois tipos de operação, operações informativas e operações físicas. De notar que

existem muitas funções que seriam necessárias à boa prática da maquinagem, mas, que não são permitidas pelo controlador executar enquanto a máquina se encontra a maquinar, por forma de segurança do utilizador e pelo próprio *firmware* não as conseguir identificar.

No esquema da figura 4.13. Estão as funções necessárias (e permitidas pelo controlador) para o utilizador monitorizar a máquina enquanto esta se encontra no processo de maquinagem.

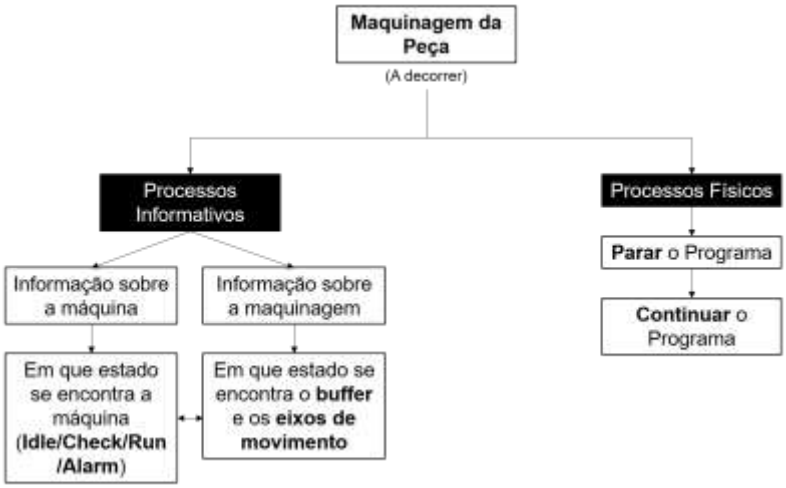


Figura 4.13 - Processos Informativos e Físicos aquando a maquinagem de uma peça.

Estas funções no "Grbl Controller" eram executadas apenas através de comandos enviados pela barra de inserção de comandos, excetuando o processo de "parar" e "continuar" que se encontram descriminados em botões.

A nível de Processos informativos é vantajoso, ao utilizador, saber em que estado se encontra a máquina. Por exemplo, muitas vezes acontece que, o operador não desbloqueia o canal ("Idle") antes de enviar o código, o que faz com que o "Grbl" envie um erro. Em termos de informação de maquinagem, é importante saber em que estado está o buffer do controlador Arduino que está a processar o código enviado, assim, o utilizador pode saber em que parte do código se encontra. Por último, mas não menos importante, saber onde os eixos de movimento se encontram nos planos de corte. Na nova interface, é necessária a introdução de um botão que peça esta informação ao "Grbl" de modo a que este a retorne visualmente.

Os processos físicos permitidos pelo "Grbl", enquanto a máquina se encontra a maquinar, remetem para a paragem do movimento e a respetiva retoma dele. Assim, a implementação de um botão para parar o processo e outro para continuar seria indispensável.

É possível então definir, nesta primeira fase, quatro níveis de funções hierárquicas para implementar na interface, como representado na figura 4.14.

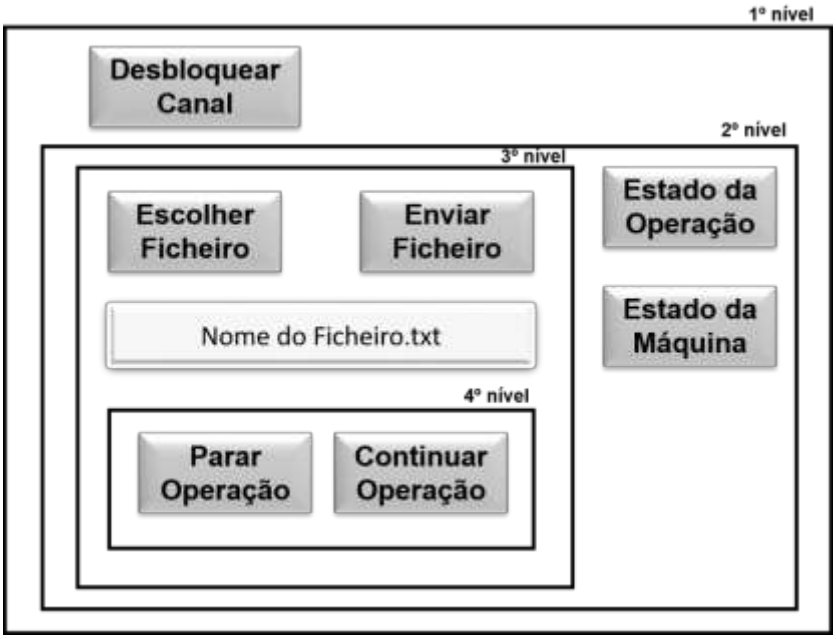


Figura 4.14 - ilustração de botões e entradas de texto para cada função requerida.

Num primeiro nível encontra-se o botão de "Idle", que seria implementado como "Desbloquear Canal", para uma maior facilidade no que toca ao operador para compreender a função deste botão. No segundo nível encontram-se os botões para fornecer informação sobre o estado da operação e o estado da máquina que, não dependeriam da maquina se encontrar já em processo de maquinagem podendo, assim, o utilizador obter estas informações antes de começar o processo. O terceiro nível remete para o envio do ficheiro: um botão para escolher, outro para enviar e uma caixa de texto para enunciar o nome ou diretório do ficheiro escolhido. No quarto e último nível, estão as operações de parar e recomeçar a maquinagem, sendo estes botões só necessários quando a maquinagem está em curso.

4.3.2. Funções relacionadas à manipulação manual dos eixos de movimento: eixo a movimentar, sentido e distância de movimento

Assim como o "Grbl Controller", quase todas as interfaces para CNC provêm de uma opção manual de movimento dos eixos. Esta operação é importante para corrigir a posição dos eixos antes do processo de maquinagem, para testar o movimento de cada eixo ou para realizar *homing* manual (Breiler, 2020a).
Na máquina CNC em questão, são utilizados três eixos de movimento, X, Z e A. Para cada eixo, haveria a necessidade de introduzir um botão para ativar o seu movimento.
Porém, assumindo que este movimento é incremental, ou seja, da posição onde se encontra o eixo, é adicionado um valor, sendo que, pode ser positivo ou negativo, consoante o sentido para que se pretende movimentar, há necessidade de saber qual o valor de distância que se pretende percorrer (e o seu respetivo sentido). Para um aumento da intuição quando em interação com o operador, definiu-se que,

tal como no "Grbl Controller", se iria implementar dois botões para movimentar cada eixo. Um botão para promover o movimento no sentido positivo e outro no sentido negativo, como representado na figura 4.15.

Figura 4.15 - Objetos necessários a incluir na interface para o controlo manual dos eixos.

Seria ainda necessário algum tipo de botão para escolher a distância a percorrer (seja esta positiva ou negativa).

É vantajoso que cada botão, intuitivamente, leve o operador a assumir o sentido que o movimento irá tomar. Por exemplo, os botões para o eixo Z devem ser colocados horizontalmente, visto que o eixo do Z promove o movimento horizontal da ferramenta. Os botões do eixo X deverão estar dispostos na vertical (tal como o movimento que efetuam). Para a organização do sentido, é necessário definir a face da máquina que estará voltava para o operador e o parâmetro do controlador que define o sentido. Se a face voltada para o operador tiver a ferramenta do lado direito da peça, então, o movimento positivo do Z e do A será para a esquerda e o de X será para cima, assentando nos sentidos impostos na figura 4.15.

O objeto que dará um número à distância a percorrer terá de ser dado em mm ou graus, sendo que, o eixo A é de rotação, logo, a transmissão de movimento é efetuada em graus (ao contrário de Z e X, em milímetros).

4.3.3. Funções relacionadas com alteração de definições e dos parâmetros da máquina

Como referido anteriormente, o "Grbl" padece de um leque de parâmetros que podem ser alterados (que influenciam a leitura e processamento do controlador quando enviadas certas ordens) e códigos que permitem visualizar informação sobre o estado da máquina e das operações e mandar diretamente operações específicas. Todos os parâmetros alteráveis são importantes para o utilizador, dependendo do que este pretende maquinar e como.

Dentro das definições do "Grbl", existem três funções para visualizar três diferentes tipos de parâmetros, nomeadamente, a lista de definições do sistema, os blocos iniciais da máquina e os códigos G54 a G59, estes comandos estão organizados conforme o esquema da figura 4.16.

Figura 4.16 - Tipos de comandos associados aos parâmetros alteráveis do Grbl *firmware*.

É relevante para o utilizador saber os valores em que se encontram as definições todas do sistema (parâmetros do "Grbl"), de modo a que, se não forem os que pretender, alterar para os valores favoráveis à máquina e à maquinagem. Os blocos iniciais da máquina mostram os blocos que se encontram no buffer do controlador Arduino para processar, assim, o utilizador sabe, ao inserir algum código para maquinar se a máquina não processará nenhum código primeiro. Por fim, os códigos G54 a G59 são parâmetros que definem a localização do espaço de trabalho através de coordenadas, podendo ser alteradas consoante a máquina que se utiliza pelo que, fornecem ao operador a noção de espaço que este tem para maquinar.

Em suma, na interface deverão ser implementados três botões, uma para cada lista de parâmetros acima descrita (Breiler, 2020a).

No que toca a alterar parâmetros, o *firmware* contém 136 parâmetros alteráveis, alguns com valores numéricos tabelados (que na linguagem de *software* traduzem o valor numa função específica – ver anexo A) e outros onde, o utilizador, pode solicitar valores (como, por exemplo, aceleração dos movimentos ou binário máximo). Como se prevê uma interface tátil, sem acesso a teclado ou rato, a alteração destes valores teria de ser toda executada no ecrã, opção que, no "Grbl Controller" é apenas adquirida com o envio de comandos específicos para o controlador (Breiler, 2020a).

Sabendo que existem 136 variáveis e que todas as variáveis são numéricas, é necessário então proceder à execução de um objeto que dê para escolher uma das variáveis, um ou mais botões para selecionar o valor que se pretende dar à variável e um botão para enviar a alteração para o controlador processar.

Por fim, para facilitar ao utilizador o processamento de certas funções através do envio de comandos, operações gerais como "Resets" deveriam também dispor de um botão. O "Grbl" fornece três tipos de "Reset", um *reset* parcial, um *reset* dos parâmetros G54-G59 e um *reset* de EEPROM[14] (Breiler, 2020a). Como a interface estará sempre conectada à mesma máquina, é possível também definir um valor correto de *homing* do eixo, o que permite programar um botão para fazer o *homing* automático da máquina. Assim, como representado na figura 4.17, os botões necessários relacionados com alteração de definições e dos Parâmetros da Máquina são, no mínimo, dez objetos e uma caixa de texto.

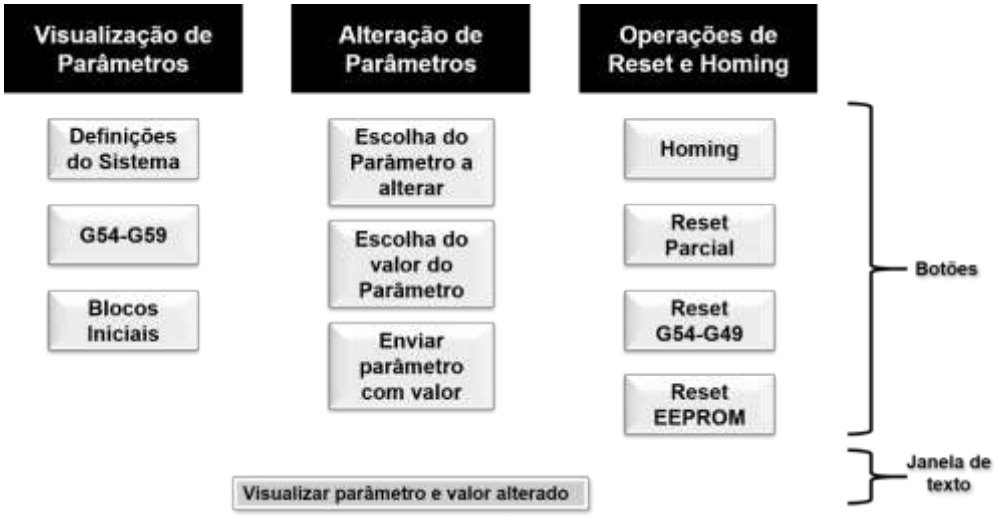


Figura 4.17 - Objetos necessários na parte remetente para as definições e parâmetros do Grbl.

4.3.4. Funções relacionadas com a comunicação homem-máquina

Todos os objetos previamente descritos e projetados, promovem envio de informação, a qual, espera sempre uma resposta pela parte do operador. Ainda, caso exista algum erro ou aviso detetado pelo controlador, é necessário que este consiga aceder a essa informação. Assim, uma das partes mais importantes da interface, é saber o que está a ser enviado para o "Grbl", se está a ser recebido e a resposta do *firmware*. Para isso, é necessário algum tipo de ecrã ou janela, como a existente no "Grbl Controller" que digita toda a informação enviada e recebida como se fosse uma "conversa" entre o homem e a máquina, como representado na figura 4.18.

Figura 4.18 - Janela de comunicação para visualizar o envio de informação e respetiva resposta.

[14] Alocação de memória de um dispositivo direcionada apenas para guardar dados para o dispositivo ler.

Num momento mais tardio de conceção, é necessário introduzir uma quinta parte no que toca a funções gerais do programa, nomeadamente, um módulo presente em quase todos os programas, o menu "Ajuda". Este menu terá incidência na interpretação do que cada objeto da plataforma faz quando utilizado, nos parâmetros (e respetiva definição de cada um) que podem ser alterados através do controlador e nos erros e alarmes que o "Grbl" pode enviar ao utilizador. Assim, o operador tem acesso rápido às variáveis necessárias para compreender a interface e a comunicação com o "Grbl" *firmware*.

4.4. Disposição dos objetos na interface

Tal como já foram distribuídos os diversos tipos de funções, é importante dividi-los também visualmente. Deste modo, o operador detém mais facilidade em encontrar o que necessita. Ao reduzir a quantidade de informação que o utilizador vê de cada vez, o programa torna-se menos confuso e mais simples, se organizado de forma correta.

Inicialmente, aquando a implementação das funções principais, tais como, a janela de visualização da informação e o envio do ficheiro, foi idealizada a primeira organização da interface, podendo esta ser observada na figura 4.19.

Esta interface provinha de apenas quatro menus principais que dividiam o programa em: visualização da comunicação e envio dos ficheiros de gcode, ajuste de parâmetros da máquina, definições gerais da CNC e ligação da interface à máquina.

Figura 4.19 - Primeira Organização da interface gráfica.

À medida que se foram programando todas as funções necessárias, percebeu-se que não seria a melhor forma de organizar esta interface, na medida que, por exemplo, a janela de comunicação era só visualizada no primeiro painel, tendo que, o operador andar a movimentar-se de painel em painel para alterar e verificar os parâmetros alterados, por exemplo.

Também o facto de existir uma parte exclusiva à ligação do dispositivo à interface não seria necessário pois, a partir do momento em que o ecrã do raspberry estivesse ligado a primeira vez à CNC, poderia ser aberto o *port* automaticamente, nunca tendo de ser alterado.

Por fim, o facto de identificar todas as funções partindo apenas de imagens, é simples para quem conhece vivamente a interface, porém, para qualquer outra pessoa, poderia levar a erros ou enganos. Para não falar do facto que, havia sido utilizada uma biblioteca de ícones *open-source* que, não está diretamente ligada ao cerne da CNC em questão.

A segunda interface, desta vez programada com mais funções de raiz, está representada na figura 4.20.

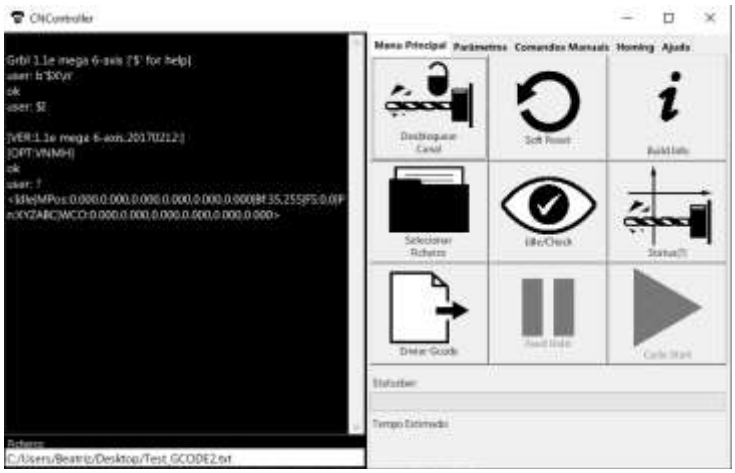


Figura 4.20 - Secção inicial da segunda interface concebida, "Menu Principal".

Esta já continha cinco secções onde, era possível observar a secção e a janela de comunicação simultaneamente.

Nesta interface já foram utilizados ícones desenhados e concebidos no *software* "Adobe CC Illustrator 2020". Para além do "Menu Principal", foi ainda inserido um menu para modificação de parâmetros, "Parâmetros", um menu para movimentar manualmente a máquina, "Controlos Manuais", um menu relativo às funções de *homing*, "Homing" e, por fim, um menu de ajuda, "Ajuda". Na figura 4.21, está representado o menu de comandos manuais que ainda não existia na primeira interface a ser concebida. Com esta secção já seria possível a movimentação manual dos eixos (positivo ou negativo). Todavia, ainda seria necessário definir em que critério se pretendia movimentá-los (coordenadas absolutas[15] ou relativas[16]) e, caso fosse um movimento absoluto, ter-se-ia de realizar previamente a definição dos zeros da peça, ainda não programados nesta interface. Por sua vez, com a definição dos zeros inserida nesta secção, torna-se irrelevante ter uma secção apenas para o *homing.*

[15] Coordenadas que, assumindo um zero como ponto de referência, são movidas sempre a partir dessa mesma referência.
[16] Coordenadas que se movem um determinado valor a partir do ponto onde a máquina se encontra.

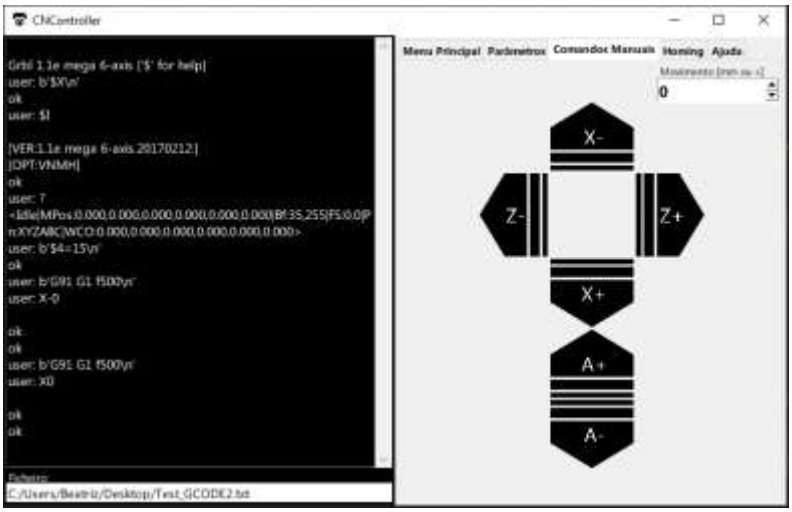


Figura 4.21 - "Comandos Manuais", segunda secção da segunda interface gráfica concebida.

Posto isto, deveriam ser encontrados os objetos que remetem para tudo o que é necessário para o envio de ficheiros pois, como a Interface está diretamente ligada ao "Grbl", o utilizador não tem necessidade de movimentar manualmente eixos e alterar parâmetros de todas as vezes que necessita de fazer uma maquinagem. De seguida, como segundo tipo de função mais utilizada, deveria aparecer o comando manual de eixos, muitas vezes utilizado antes de iniciar o processo de maquinagem. Em terceiro lugar, todos os objetos que executam alteração de parâmetros e funções especificas do "Grbl". Por último, o menu de ajuda do operador, para momentos em que este necessite de informação sobre o *software*.

Como, todas as partes do programa anteriormente faladas, promovem o envio de informação para o controlador e esperam a respetiva resposta, a janela de comunicação entre o "Grbl" e o utilizador deve ser possível de visualizar em qualquer das partes do programa onde este se encontrar.

Através desta lógica, surge então a repartição da interface em menus, como se fossem separadores, onde o utilizador clica e pode mudar o menu em que se encontra. Os quatro menus descritos foram "Menu Principal", onde se encontram os objetos relativos à ativação da máquina, envio de ficheiros e operações aquando a maquinagem, "Comando Manual", onde estão dispostos os comandos que permitem movimentar os eixos manualmente, "Definições", para todos os objetos de alteração de parâmetros, visualização e execução de funções específicas e "Ajuda" que promove auxílio ao operador. Todos estes menus encontram-se discriminados, assim como as funções de cada componente do menu, no apêndice V.

Apesar desta, terceira e última interface, já se encontrar praticamente toda programada. Ao longo do projeto e examinação, são sempre encontradas mais funções e necessidades na sua utilização que, posteriormente são implementadas.

O “Menu Principal” que se observa na figura 4.22, comporta todas as tarefas que operador tem de fazer no início e aquando a operação de maquinagem, na primeira linha encontram-se os botões que promovem o desbloqueio do canal e a informação do estado do mesmo. Na segunda, as operações para escolher e enviar o ficheiro para a CNC. Na terceira, um botão para fazer um *soft reset* do *firmware* e dois botões para parar e continuar a maquinagem após iniciada.

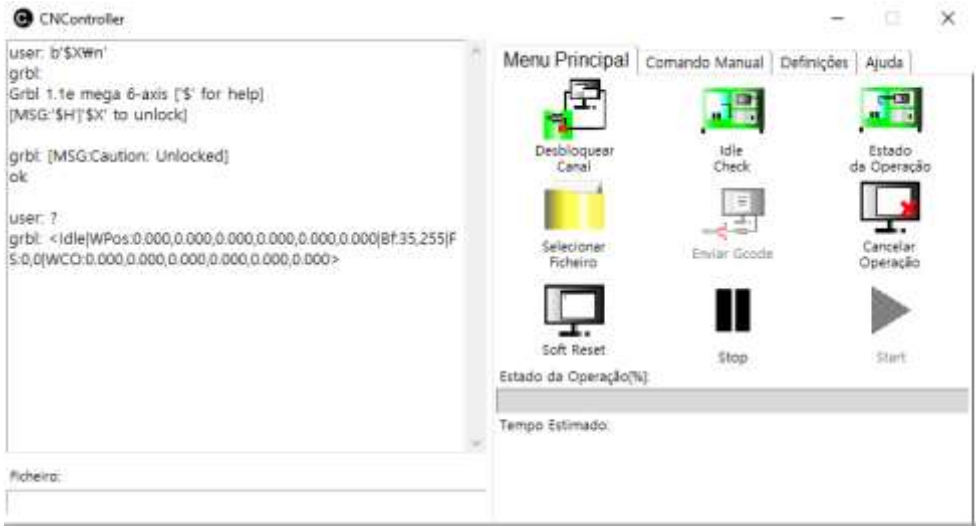


Figura 4.22 - Menu Principal da interface gráfica "CNController".

Neste caso, foi clicado o botão de desbloquear o canal e, de seguida, o botão de estado da operação. No painel de comunicação pode observar-se a mensagem enviada e a resposta recebida.

A secção seguinte, “Comando Manual”, observada na figura 4.23, comporta seis botões para movimentar os três eixos da máquina positiva e negativamente segundo o padrão definido. É possível escolher o tipo de coordenadas que se pretende para o movimento, escolher os zeros peça e ainda fazer o zero máquina do eixo do Z.

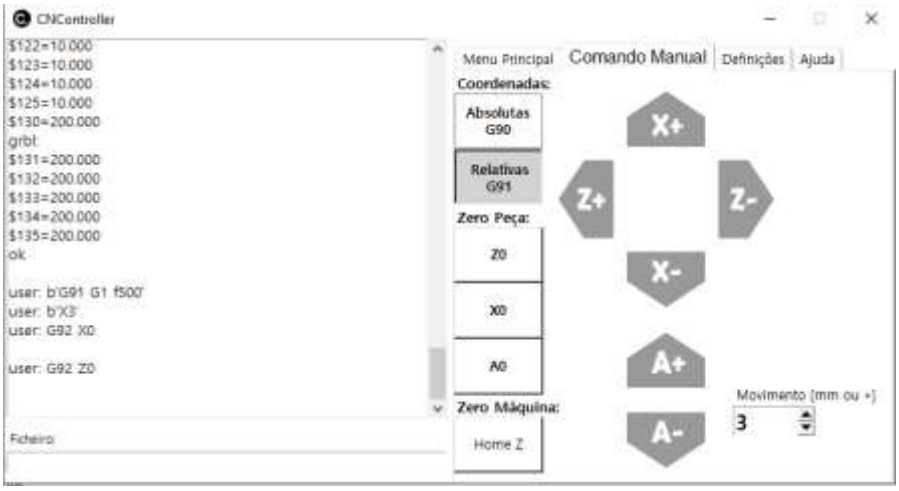


Figura 4.23 - Comando Manual da interface gráfica "CNController".

Como é possível observar na figura 4.23, o utilizador premiu o botão para coordenadas relativas e movimentou três milímetros do eixo positivo X. de seguida, realizou o X0 e o Z0 da peça no ponto onde se encontrava.

Na figura 4.24, está exposta a terceira secção da interface, "Definições". Aqui, é possível alterar os parâmetros relativos ao *firmware*, através da seleção do parâmetro a alterar e do valor que se pretende para o mesmo.

O parâmetro para ativar o *homing*, como é utilizado mais frequentemente, foi recriado num botão para ser mais fácil de alterar.

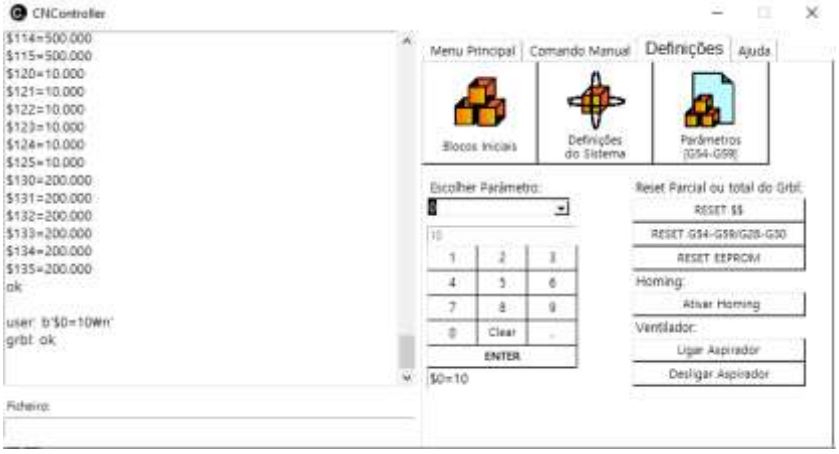


Figura 4.24 - Definições da interface gráfica "CNController".

O utilizador pediu ao "Grbl" para enviar os parâmetros do sistema, como representado na figura 4.24 no painel e, seguidamente, alterou o parâmetro "$0" para o valor 10, ao qual o *firmware* respondeu "ok". Os parâmetros dos "Grbl" ficam guardados sempre na base de dados a não ser que se realize um *reset* do EEPROM.

Por fim, a última parte do programa é um menu "Ajuda", representado na figura 4.25, que possui três botões para ajudar o operador no que toca a erros e alarmes do *firmware*, definição do que os botões da interface fazem e definição de cada parâmetro do "Grbl".

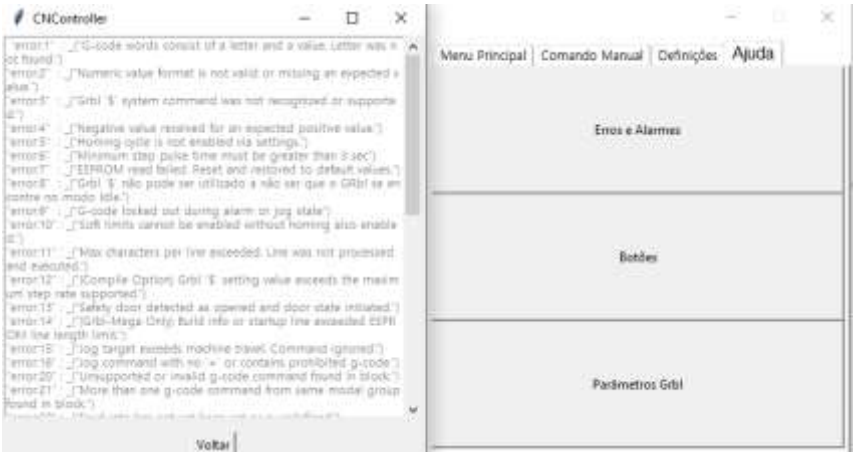


Figura 4.25 - Ajuda da interface gráfica "CNController".

Neste caso, o utilizador clicou no botão "Erros e Alarmes" e apareceu uma janela no lugar do painel de comunicação onde estão descritos todos os erros e alarmes que o *firmware* pode enviar e o respetivo significado.

No anexo E encontram-se, detalhadamente descritos, todos os botões e objetos da interface.

4.5. Programação da interface para a CNC da empresa Padrão Ortopédico

A interface desenvolvida para a máquina CNC foi inteiramente escrita em "python", uma linguagem de programação de objetos, nomeadamente, "Python 3" (Rodas de Paz, n.d.).

Para uma melhor abordagem e implementação das bibliotecas necessárias, foi também utilizado um "Python" IDE, isto é, um programa de edição de código predefinido, "Pycharm", e um ambiente virtual gerado pelo compilador "Anaconda 3".

Como base do programa foi utilizada a biblioteca *"Tkinter"*, uma biblioteca que fornece objetos de contexto visual.

O código correspondente a todo o programa encontra-se descriminado no apêndice VI.

Foi adotada uma estrutura "MVC" para a organização do código: "Model-View-Control". Esta estrutura, simplificada na figura 4.26, é geralmente usada em programas para facilitar ao programador a relação entre diferentes tipos de variáveis (Pop & Altar, 2014).

O "Model" (modelo) engloba todas as variáveis numéricas ou literárias do programa. Esta parte do programa tem como função registar e guardar os itens pretendidos pelo utilizador (Pop & Altar, 2014).

O "View" (Visualizador) contempla todos os objetos visíveis do programa, ou seja, todo o código que originará algo que o utilizador vê através de um ecrã (Pop & Altar, 2014).

Por fim, o "Control" (controlador) direciona as variáveis entre o "Model" e o "View", de modo a que estas percorram um caminho sem se perderem (Pop & Altar, 2014).

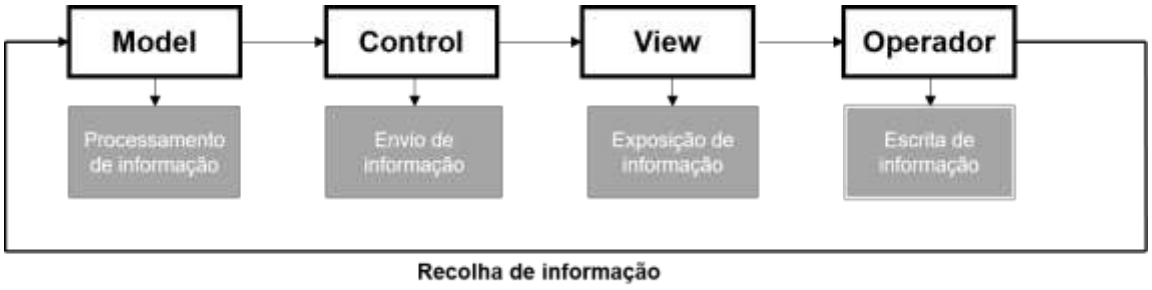


Figura 4.26 - Fluxo de informação no modelo MVC.

Resumidamente, na GUI criada, existem então quatro classes, três MVC e um main que executa o programa. Estas classes correspondem cada uma a um *script* de "Python3". As variáveis e funções utilizadas estão descritas na imagem 4.27, esquema UML simulado no IDE "Pycharm Professional 2020".

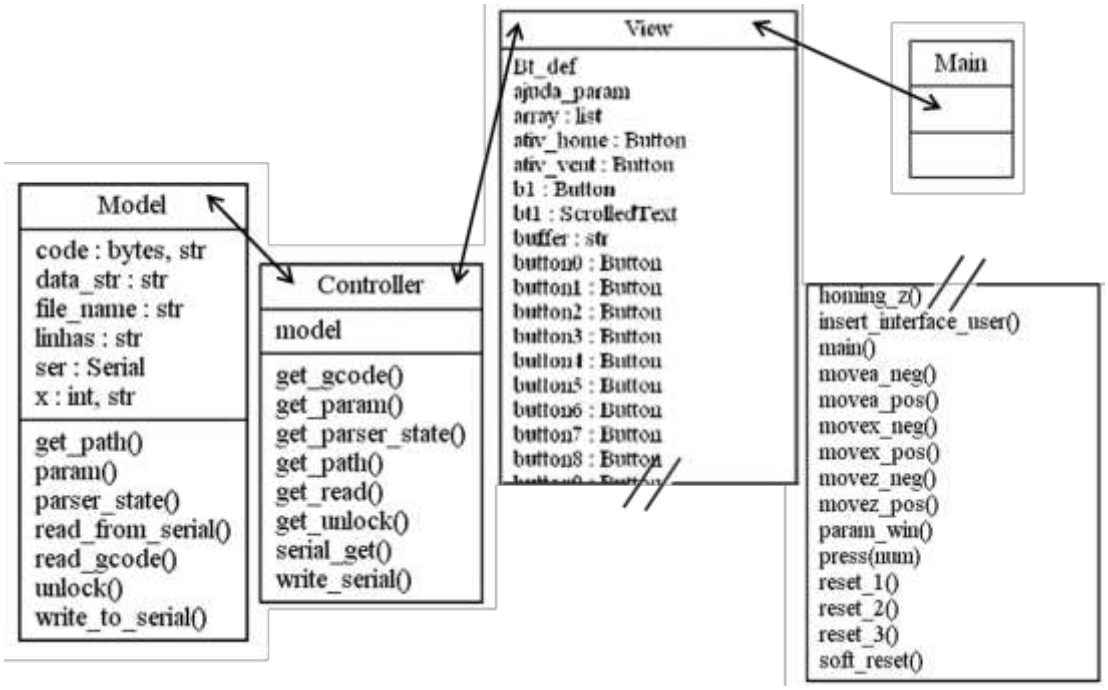


Figura 4.27 - Variáveis e funções da estrutura MVC dos *scripts* de "python".

4.5.1. Leitura e processamento de variáveis: model

A informação ou os dados, tal como evidenciado no esquema da figura 4.26, exercem um movimento cíclico no programa. O "Model" processa os dados que entram no sistema atribuindo valores às variáveis que, por sua vez, são enviadas para o "Control" que as encaminha para o "View". No "View", as variáveis são transformadas em objetos visíveis pelo operador para que, deste modo, as possa alterar ou validar. Se o operador inserir novos dados no programa, estes serão reencaminhados de novo para a o "View" e percorrerão o percurso anterior mais uma vez. (Pop & Altar, 2014)

O "Model" está dividido em variáveis e funções de processamento de variáveis. Na figura 4.28, encontram-se as dependências de cada uma delas.

É nesta classe que está programado o canal em série que promove a ligação da interface com o "Grbl" *firmware* da máquina e a possibilidade de escolha do ficheiro de código G que se pretende maquinar.

Resumidamente, são criadas variáveis com valores nulos na definição de início da classe e utilizadas as funções para dar valores a essas variáveis para, posteriormente, serem enviadas para o "control" e serem processadas.

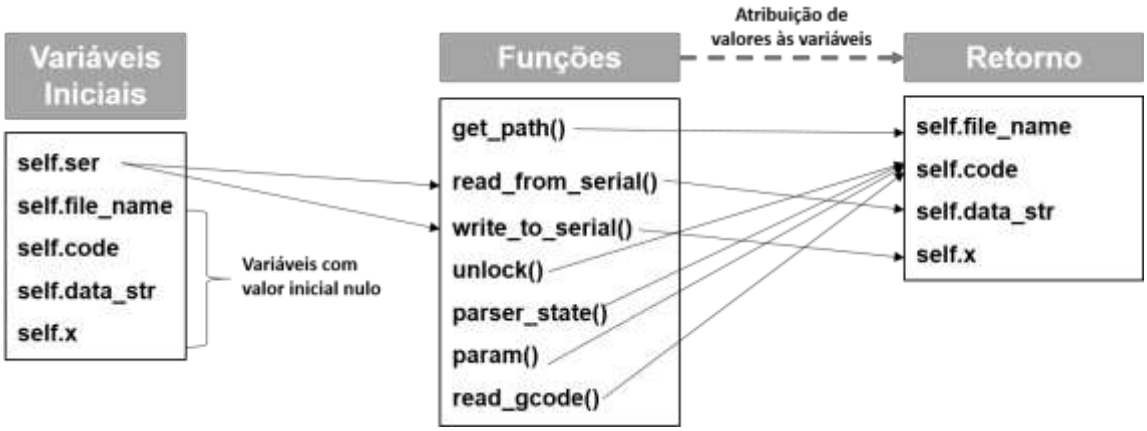


Figura 4.28 - Fluxo de informação das variáveis na classe "Model".

Como todos os módulos "Class[17]", é necessário enunciar as variáveis inicias que, iniciarão instantaneamente com o correr do programa, segundo os valores que lhes foram atribuídos. Neste *script*, foram utilizadas três bibliotecas gerais do "python3", representadas no excerto de código da figura 4:29.

```
"from Tkinter import filedialog
import serial
import time
"
```

Figura 4.29 - Bibliotecas utilizadas na Classe "Model".

A "filedialog", uma biblioteca secundária da "Tkinter", promove certas operações como abrir um diretório do sistema operativo (consoante as especificações), selecionar um ficheiro, ler o ficheiro, alterar o ficheiro, guardar as alterações submetidas a um ficheiro, entre outras (Rodas de Paz, n.d.).

A biblioteca "serial" está desenvolvida para programar todo o tipo de funções que remetem para a comunicação em série de dois dispositivos, nomeadamente, enviar e receber dados (Liechti, 2018).

Por fim, a biblioteca "time" comporta uma gama de funções gerais temporais para controlar operações ou variáveis. Esta biblioteca é normalmente utilizada no âmbito de funções de outras bibliotecas que, necessitam de coordenação temporal (Rodas de Paz, n.d.).

Como representado na figura 4.30, é definida uma primeira variável "self.ser" que, tem como função ligar o canal serial entre o dispositivo onde o programa se encontra e um outro dispositivo. A expressão "serial.Serial" remete para a chamada da função "Serial" (de ligação do canal) na biblioteca importada "serial". Uma das variáveis utilizadas desta função, o "port" (porta), é definida pela porta "COM8", canal onde se encontra ligado o cabo de dados referente ao controlador Arduino que contém o "Grbl". Este nome varia consoante a porta do *hardware* onde se encontra conectado e o dispositivo com que se

[17] Construtor de objetos, ou seja, agrupa objetos (e a informação deles) que, podem ser utilizados por toda a classe.

pretende estabelecer a ligação. No sistema operativo “Windows”, é possível visualizar o nome das portas no “Gestor de Dispositivos”, representado na figura 4.31 (Liechti, 2018).

```
class “Model”:
    def __init__(self):
        self.ser = serial.Serial('COM8', 115200, timeout=None, xon-
xoff=True, stopbits=1, bytesize=8)
```

Figura 4.30 - Variável inicial "self.ser" que abre o canal em série com o controlador da máquina.

Para que a conexão seja possível, os dispositivos têm de se encontrar na mesma banda, neste caso, como o “Grbl” utiliza a banda 115200, a segunda variável da ligação em série, “baudrate” é dada pelo numerário “115200”. Ainda segundo as especificações do “Grbl”, é sabido que número de bits de dados é oito com um bit de paragem, por isso, também na ligação em série é estipulado “stopbits=1, bytesize=8”. A variável “timeout=None” faz com que a ligação em série seja imediata e, por último, “xonxoff=True” que permite o controlo de movimento de dados na comunicação (Liechti, 2018).

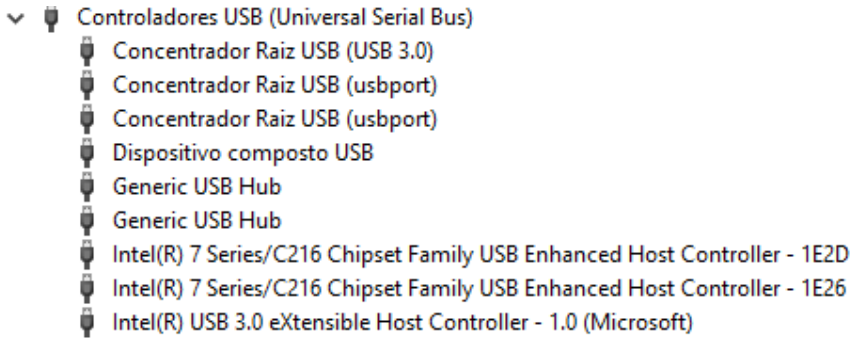


Figura 4.31 - Gestor de dispositivos do Windows.

As variáveis iniciais “self.file_name = ''”, “self.code = ''”, “self.data_str = ''” e “self.x = ''” são utilizadas nas funções dentro da class “Model” tendo, assim, de ser evidenciadas logo de início, para não ocorrerem erros de processamento de informação. Deste modo, é possível interligar estas variáveis exclusivamente à *class* “Model”, tornando-se variáveis locais. A utilização do caracter aspas (“), designa um valor nulo, ou seja, consoante a variável seja numérica ou *string*.

A função “def get_path(self)”, que retorna a variável “self.file_name”,é utilizada para guardar o diretório do ficheiro escolhido e, mais tarde, abrir, processar e enviar o ficheiro para o “Grbl”. A função que define a variável, nomeada por “filedialog.askopenfilename” é uma função da biblioteca “filedialog”, “askopenfilename”, responsável por fazer *pop-up* de uma janela do sistema operativo para o utilizador procurar o ficheiro que pretende (Rodas de Paz, n.d.).

A opção “initialdir="/"”, representada na figura 4.32, é responsável por especificar a localização do disco onde se pretende procurar o ficheiro. Neste caso, “/”, faz com que a janela aberta seja o disco em geral, podendo o operador pesquisar em qualquer compartimento do disco. Pode-se alterar este diretório para qualquer pasta específica na memória do computador (Rodas de Paz, n.d.).

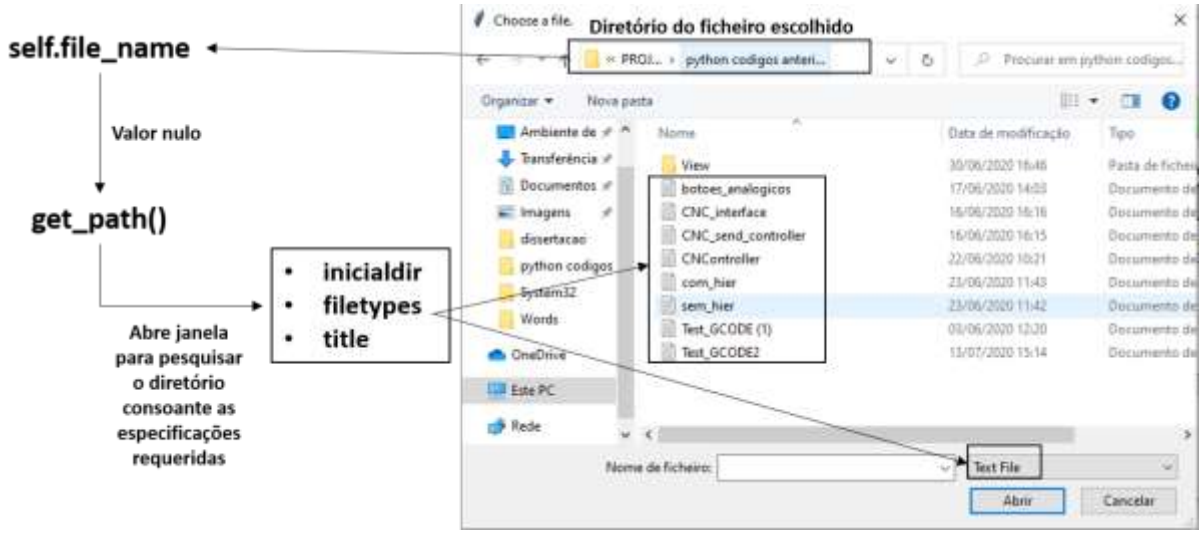


Figura 4.32 - Atribuição de um valor à variavel "self.file_name".

Como os ficheiros requeridos, ficheiros de código G, são ficheiros de texto (“.txt”), não é necessário encontrar outro tipo de ficheiro na pesquisa. Assim, com a variável “filetypes=(("Text File", "*.txt")”, o tipo de ficheiro pode ser definido como apenas “Text.File” (ficheiro de texto). A variável “title="Choose a file."” Atribui um nome à janela de pesquisa, nomeadamente, “Choose a file” (Rodas de Paz, n.d.).

Ainda na classe “Model”, apesar do canal de comunicação em série já se encontrar ativo, o programa espera que se especifique a diferença entre enviar e receber dados do computador para o “Grbl”. A função “def read_from_serial(self)” é responsável por receber todos os dados que o “Grbl” envia pelo canal, retornando-os através da variável “self.data_str”. Esta função engloba um ciclo “While” que, enquanto o programa o considerar verdadeiro (“True”), está continuamente a ser processado. Neste caso, a condição do qual a veracidade deste depende é a expressão “if self.ser.inWaiting() > 0”, ou seja, se a variável que reconhece o canal, “self.ser”, identifica bits de dados no buffer do emissor superiores a zero, “self.ser.inWaiting > 0”. A variável “self.data_str” tomará o valor dos bits presentes no buffer. Estes bits são lidos através da função da biblioteca “serial”, “self.ser.read”. como os bits apresentam caracteres “ascii” codificados, estes terão de ser descodificados, “decode('ascii')”. Para uma questão de garantir que o código está a funcionar, é pedido ainda que seja expressa a informação referida no *prompt*, através da função “print(self.data_str)” (Rodas de Paz, n.d.).

A função “time.sleep(0.01)”, coordena o intervalo de tempo que cada ciclo espera para recomeçar após terminar, neste caso, 0,01 segundos.

Por outro lado, a função “def write_to_serial(self)” é responsável pelo envio de informação do utlizador para o “Grbl”. Os dados são colocados na variável “self.x” que, toma o valor segundo a função “self.ser.write(self.code)”: a função “write” da biblioteca “serial” envia, pelo canal, “self.ser”, o que o utilizador pretende, nomeadamente, o conteúdo da variável “self.code” (Liechti, 2018).

As funções “def unlock(self)”, “def parser_state(self)” e “def param(self)” retornam valores específicos para funções estipuladas pelo “Grbl”, expressas na figura 4.33. A função “unlock” define a variável “self.code”, que, como dito, é responsável por enviar dados para o *firmware*, para o valor “$X”, promovendo o desbloqueio do canal. Como o “Grbl” compreende caracteres em “ascii”, é então necessária a utilização dos mesmos, para isso há a complementação da *string* para “b'$X\n'” (“b” define o início de linha e “\n” o critério de paragem). A definição “parser_state” altera o valor de “self.code” para “$G” (também com a respetiva codificação) que, promove o pedido dos parâmetros de estado do bloco inicial ao “Grbl”. Por fim, a função “param” altera o valor de “self.code” para “$$”, solicitando os parâmetros gerais do programa (Breiler, 2020a).

```
"def unlock(self):
    self.code = b'$X\n'
    print(self.code)
    return self.code

def parser_state(self):
    self.code = b'$G\n'
    print(self.code)
    return self.code

def param(self):
    self.code = b'$$\n'
    print(self.code)
    return self.code"
```

Figura 4.33 - Variáveis "unlock", "parser_state" e "param" da classe “Model”.

Por último, a função “def read_gcode(self)”, representada na figura 4.34, provavelmente a função mais importante da interface. Esta função abre o ficheiro do código G selecionado, “open(self.file_name, ‘r’)”, onde o caráter “r” o invoca com permissão para ler. É criado um “flushInput” do canal para que não seja ocupada a memória real do controlador Arduino, assim, mal os dados são lidos pelo controlador Arduino, são apagados do seu *buffer*. Parte desta função foi adaptada de um programa[18] de comunicação em série publicado na plataforma “Github” pelo utilizador “Chamnit”.

O ficheiro é lido linha a linha e transforma cada linha na variável “l_block”. É indicado para esperar pela indicação de “ok” do “Grbl” para alterar o valor desta variável e enviar a linha seguinte.

[18] Disponível em : https://github.com/gnea/grbl/blob/master/doc/script/stream.py (janeiro, 2021.

Esta função não só envia o código como verifica, constantemente, se o estado do *buffer* permite enviar mais uma linha.

Caso ocorra algum tipo de erro na leitura, este vai ser lido pelo código e apresentado no painel de comunicação.

```
“def read_gcode(self):
    self.response=False
    with open(self.file_name, 'r') as f:

            # Wait for grbl to initialize and flush startup text in serial input
        time.sleep(5)
        self.serial.flushInput()
        self.reading = 'Streaming Gcode...'
[…]
if self.verbose: print("SND>" + str(self.l_count) + ": \"" + l_block + "\"")
self.resulta = l_block + '\n'
self.talk = self.serial.write(self.resulta.encode())  # Send g-code block to
grbl
self.grbl = '{}'.format(self.resulta)
self.interface.insert(END, self.grbl)
self.interface.yview(END)
while 1:
    self.rbl_out = self.serial.readline().strip().decode()  # Wait for grbl res-
ponse with carriage return
    if self.grbl_out.find('ok') >= 0:
        if self.verbose:
            print("  REC<" + str(self.l_count) + ": \"" + self.grbl_out + "\"")
        break
[…]”
```

Figura 4.34 - Função "read_gcode" da classe “model”.

4.5.2. Envio das variáveis processadas para a interface: “control”

Como havia sido especificado, o “Control” (Controlador) é como um canal entre o “Model” e o “View”. Neste caso, como o programa está escrito em “Python”, uma linguagem de objetos, não haveria necessidade de ter esta classe a intermediar a informação. Mas, com vista a cumprir o modelo seguido, MVC, o “Control” foi inserido como meio de retornar as variáveis desenvolvidas no “Model” para o “View”. Para isso, é apenas importada a classe “Model”, do ficheiro “model”, como biblioteca (“from model import Model”).

Esta classe, ao contrário das restantes, comporta apenas uma variável inicial, pois, apenas se pretende que estas funções retornem valores quando chamadas pelo utilizador.

A variável inicial “self.model” define a classe “Model” para que, possam ser utilizados os valores previamente retornados no “Model” na classe “Controller”.

De notar que, para utilizar as variáveis do “Model” é necessária a utilização do caminho gerado “self.model”, ou seja:

- "self.model,ser"
- "self.model.get_path"
- "self.model.read_from_serial"
- "self.model.write_to_serial"
- "self.model.unlock"
- "self.model.parser_state"
- "self.model.param"
- "self.model.read_gcode"

Estas variáveis são retornadas, respetivamente, pelas funções da classe "Controller": "serial_get", "get_path", "get_read", "write_serial", "get_unlock", "get_parser_state", "get_param" e "get_gcode", podendo, assim, ser utilizadas na classe "View" após importar o "Control", como representado na figura 4.35.

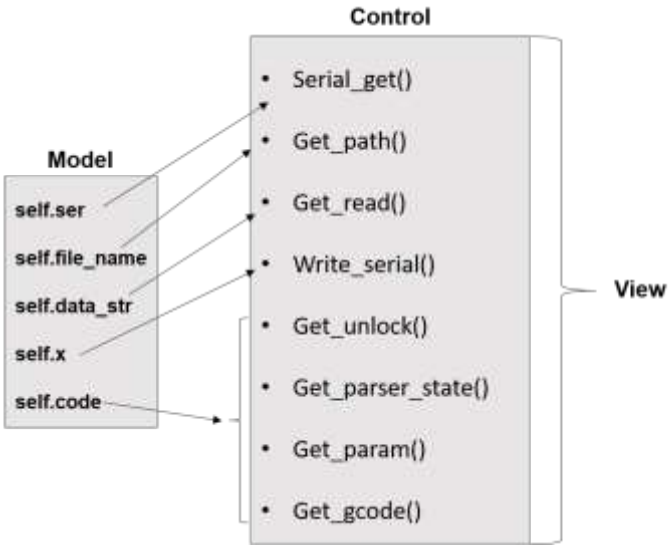


Figura 4.35 - Dependência das funções do "Control" nas variáveis do "Model".

4.5.3. Visualização dos objetos e variáveis pelo utilizador: view

O *script* "View" engloba tudo o que é possível ao utilizador visualizar na interface, desde a janela do programa, os botões, as barras, as imagens, as caixas de texto ou até as próprias letras.

Como já foi referido, a biblioteca em que assenta este programa, denominada de *"*Tkinter*"*, promove a utilização de funções e códigos específicos para criar diferentes tipos de objetos visíveis, cada um com as suas diferentes funções (Rodas de Paz, n.d.).

De modo geral, qualquer código é iniciado por enunciar as bibliotecas utilizadas, não só pela organização do trabalho, mas, também pelo facto de que a hierarquia da escrita é de extrema importância. Ao utilizar uma biblioteca, é necessário que esta esteja importada antes da primeira vez que se pretenda utilizar.

Na figura 4.36 estão representadas as bibliotecas utilizadas para a escrita do *script* "View". A biblioteca "Tkinter" possui diversas extensões interessantes para o programador, que permitem a aglomeração de objetos do mesmo tipo em secções, nomeadamente a biblioteca "ttk" e "scrolledtext". A partir da biblioteca "ttk" é possível agrupar diversos formatos de botões e janelas, conseguindo assim editar todos os objetos simultaneamente, evitando linhas de código desnecessárias e tempo de trabalho. No caso da biblioteca "scrolledtext", há vantagem na definição dos códigos apenas para o grupo específico desse objeto, neste caso, uma janela de texto já programada com barra lateral, onde apenas é necessário o ajuste de parâmetros como, por exemplo, tamanho, letra ou cor (GeeksforGeeks, 2020).

```
"from Tkinter import *
from Tkinter import ttk
import threading
import time
import Tkinter.scrolledtext as tkst
from Tkinter.ttk import Progressbar
from controller import Controller
from error_list import lista_erros
"
```

Figura 4.36 - *Scripts* e bibliotecas importadas da class "View".

Para facilitar a escrita do código, algumas das bibliotecas foram importadas com siglas, tal como "tkst", diminuído assim o tamanho das funções e código em que estas necessitem de ser enunciadas. Pelo mesmo motivo, realizou-se também a importação de objetos específicos da biblioteca no caso, por exemplo, do *widget* "Progressbar" (van Hattem, 2020).

As últimas duas linhas apresentadas não se referem a bibliotecas intrínsecas ao "Python", mas sim, aos dois *scripts* que reencaminham informação para ser apresentada visualmente, sendo estes o "controller" e a "error_list".

Cada parte do programa está dividida em classes, neste caso, para a divisão da classe "View", é necessária a resposta às perguntas:

- O que aparece, visualmente, logo após o início do programa?
- O que acontece quando o utilizador interage com o programa?
- Como é que acontece?

Como se trata de uma interface homem-máquina, é de esperar que, na ideia primordial, exista uma janela com botões que, ao pressionar os botões seja exercida uma função. Posto isto, é possível então responder as perguntas acima referidas. Primeiro é necessário aparecer uma janela com botões, de

seguida é necessário que o utilizador possa pressionar os botões e, por fim, estes terão de fazer algo acontecer.

Começando pelo início: gerar uma janela e gerar botões. Tendo em conta que tudo o que é visualizado é código, é necessário que, tudo o que se vê quando o programa é corrido, seja programado exatamente para aparecer em simultâneo. A maneira mais simples de fazer tal coisa é recorrendo a uma variável inicial "def __init__" que, automaticamente, quando a class é chamada, executa todos os comandos que nela se encontram definidos.

O código, apenas nesta definição inicial, comporta 61 *widgets* que transformam o programa no que o utilizador vê quando o abre, isto é, as quatro janelas observáveis na figura 4.37.

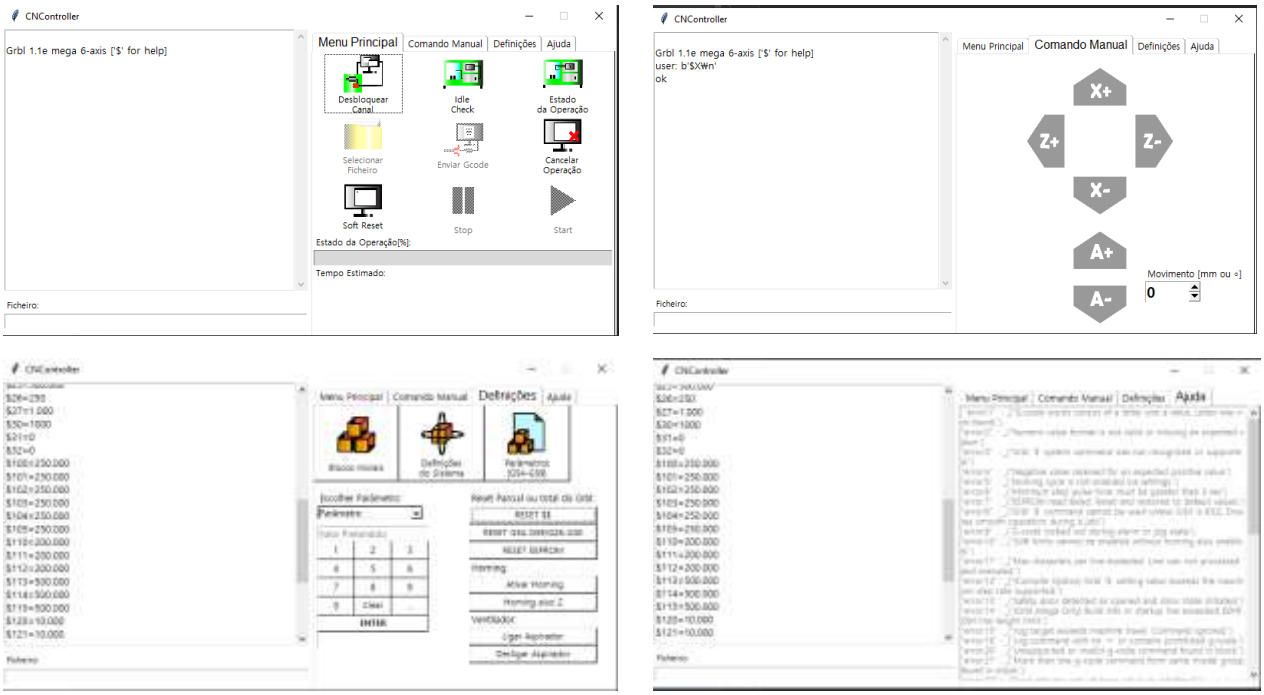


Figura 4.37 - Menus Visualizados pelo Operador da Interface.

Caminhando por partes, como seria de esperar, para implementar o que quer se seja numa janela de visualização, primeiro, é necessário criar a janela.

Na figura 4.38, encontram-se as primeiras quatro e fundamentais linhas de código para a interface. "self.root = Tk()" promove a criação de uma janela através da biblioteca *"Tkinter"*, sendo esta considerada como *master* de qualquer outra subjacente. É também nomeada através da função ".title" e dimensionada a partir da função "*.geometry*". Como esta aplicação é para ser utilizada num RaspberryPi 4, com ecrã tatil de 800 pixéis de comprimento por 400 pixéis de largura, a janela foi programada para a impossibilidade de ser redimensionada aquando a utilização ("self.root.resizable(width=False, height=False)") utilizando-se as variáveis de tamanho *width* (comprimento) e *height* (altura) como variáveis binárias onde a sua variação é dada como "Falsa" (Rodas de Paz, n.d.).

```
“self.root = Tk()
self.root.title("CNController")
self.root.geometry("800x400")
self.root.resizable(width=False, height=False)
self.root.config(bg='white')”
```

Figura 4.38 - Programação da janela principal "root".

Ainda sobre a janela principal, é programada uma cor para a mesma, neste caso, branco.

Após a introdução de uma janela, será então possível implementar objetos na janela (relembrando sempre que a hierarquia de acontecimentos é de extrema importância). Caso um objeto seja implementado previamente ao *widget master* desse objeto, o compilador não vai reconhecer e, portanto, a aplicação não será iniciada.

Para uma melhor compreensão da hierarquia de objetos, na figura 4.39, apresenta-se um diagrama com a sucessão de acontecimentos na direção de mais elevada hierarquia para mais baixa.

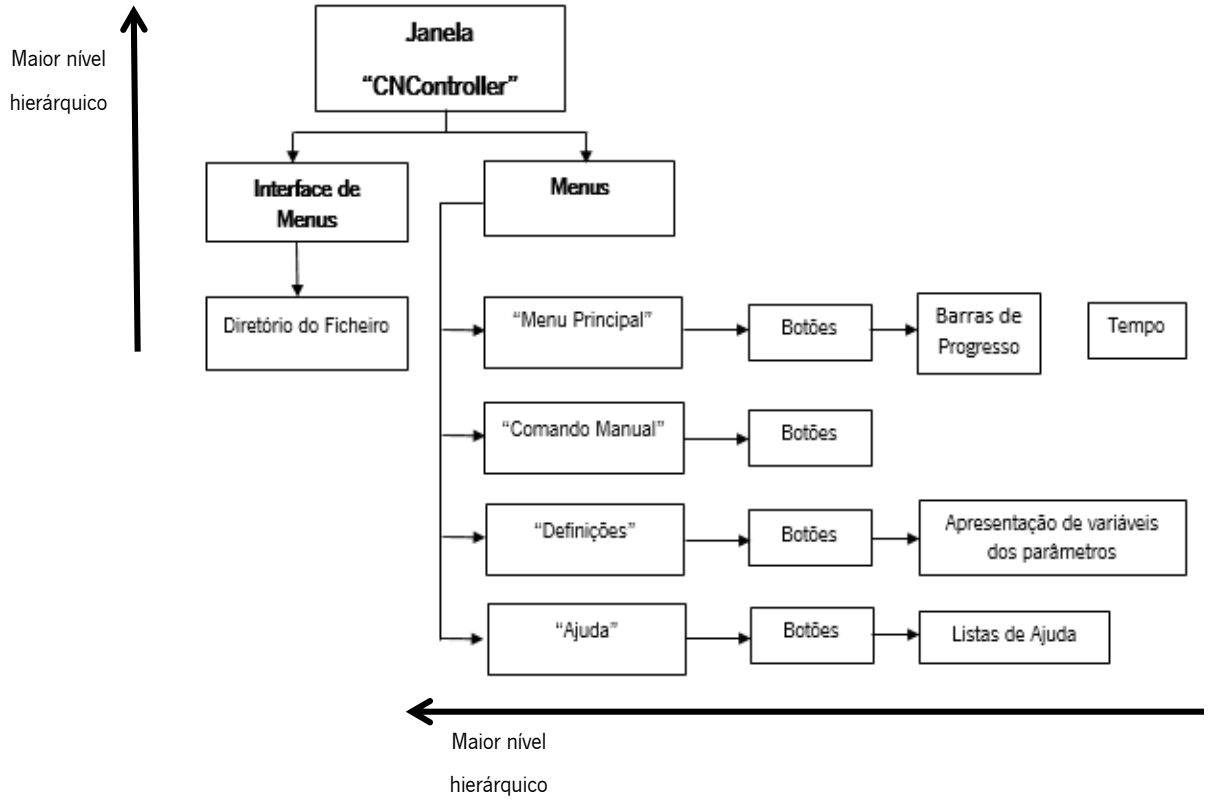


Figura 4.39 - Esquematização dos níveis de informação requeridos para a interface.

Posto isto, a janela principal vai-se encontrar dividida em duas janelas secundárias, uma que dará entrada à interface de comunicação e ao diretório do ficheiro de código a maquinar (do lado esquerdo da interface) e uma outra janela, no lado direito, que se dividirá nos quatro menus descritos. A divisão foi realizada desta forma para que, em qualquer menu que o operador se encontre, consiga visualizar o envio e

receção de informação que está a ser submetido entre o controlador e o "Grbl" implementado na máquina.

A variável "self.interface", na figura 4.40, representa a criação da janela de texto do lado esquerdo do programa e a sua respetiva barra lateral de controlo. Este tipo *widget* é implementado a partir da biblioteca "tkst" e o objeto em si denomina-se por "ScrolledText" (GeeksforGeeks, 2020).

As variáveis incidentes são a janela onde se encontra ("self.label"), a cor de fundo("*bg*"), a cor da letra("*fg*)" e a fonte("*font*").

Qualquer que seja o objeto tem de ser obrigatoriamente posicionado na janela, caso contrário, este não irá aparecer. Nesta situação utilizou-se a técnica "place" que permite posicionar o *widget* de texto através das coordenas x e y em pixéis da janela, sendo possível dimensionar a altura e o comprimento do mesmo.

Como o objeto já está configurado com o barra de *scroll* é necessário dizer ao compilador onde será apresentada a barra, neste caso, será ao longo do eixo Y, ou seja, verticalmente.

Seguidamente, encontra-se definida a caixa de texto "self.nome" que promove o aparecimento da frase "Ficheiro:" antes da caixa de texto onde aparecerá o diretório. É possível escrever no *widget label* incluindo uma especificação denominada por "texto=' ' ".

O último objeto inserido na *label* "self.label" é então a caixa de texto responsável por apresentar o diretório do ficheiro escolhido pelo operador, "self.entry_file_path".

Figura 4.40 - Identificação dos objetos da janela de comunicação correspondentes às variáveis programadas.

A expansão da biblioteca *"Tkinter"*, "tkk", fornece certos módulos ao utilizador, nomeadamente, um grupo de *widgets* intitulado de "Notebook". Este núcleo permite a elaboração de uma estrutura com submenus, representada na figura 4.41, automaticamente gerada após o programador incluir a quantidade de *labels* que pretende na *label* principal com os respetivos nomes para cada *label*. Em cada *frame*

criada é possível agrupar qualquer objeto, sendo que, de uma *frame* para a outra, a informação que vai ser visualizada será diferente (Foundation, 2021).

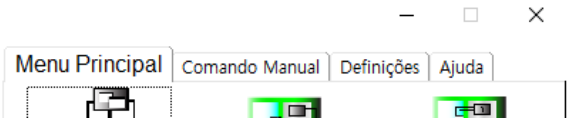


Figura 4.41 - Visualização do formato ttk.Notebook da biblioteca *"Tkinter"*.

Inicialmente, foi criada uma *label* para agrupar a parte esquerda da interface, já mencionada, englobando esta a janela de comunicação com o controlador Arduino e o respetivo *path* (diretório) do ficheiro a enviar. Na "self.label", é posicionado então o grupo de *frames* geradas pelo "ttk.Notebook". O agrupamento de objetos através de diferentes *labels* é de extrema importância para facilitar o posicionamento dos *widgets*, dado que na mesma *label* não é possível utilizar todos os métodos de posição em simultâneo, sendo estes preferenciais dependendo do tipo de objeto e estrutura pretendidos. Assim, é preferível a utilização de *labels* de menor tamanho dentro de outras *labels* (Foundation, 2021).

Como se pode ver na figura 4.42, é criado um "tabControl", ou seja, uma variável "Notebook", com origem na "label2". Dá-se, de seguida, a implementação de quatro *frames* para cada *tab*, "label1", "label3", "label1_1" e "label5": "Menu Principal", "Comando Manual", Definições e "Ajuda" (respetivamente) (Foundation, 2021).

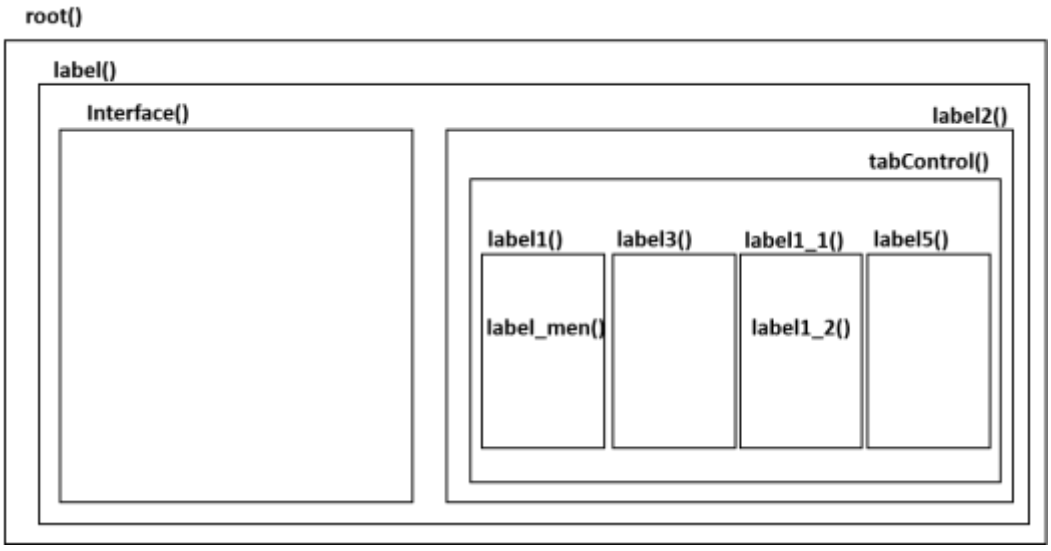


Figura 4.42 - Dependência em camada das diferentes *labels* da interface.

Existem ainda as labels "label_men", responsável por agrupar os botões do "Menu Principal" e a *label* "label1_2" que mais tarde englobará parte dos botões da *tab* "Comando Manual".

No final da implementação de todas as *tabs* desejadas, é necessário dizer ao complidor que os objetos existem e como existem, neste caso, é utilizada a estrutura "pack" com duas características previamente requeridas: "expand=1" para a *tab* onde o utilizador se encontra aumentar aquando a utilização e "fill=

both" para que todo o comprimento do espaço onde se encontram os nomes das *frames* seja preenchido automaticamente (Foundation, 2021).

Como foi esquematizado, o "Menu Principal" engloba três grupos de *widgets* principais: botões, barras de progresso e caixas de texto (para frases e para o tempo).

Nesta *frame*, os botões estão todos posicionados numa *label* submissa à "label1", "label_men". Isto porque, pretendia-se uma estrutura matricial, facilmente obtida através da implementação de posição "grid". Porém, se tudo fosse implementado deste modo, restringiria o alinhamento de outros objetos que não contemplassem o mesmo tamanho que os botões ou vice-versa.

Foi então idealizada uma matriz 3x3 para agrupar os botões principais do programa, podendo o utilizador, apenas através do "Menu Principal" programar a CNC para maquinar.

Na primeira linha de botões, são implementados os objetos "unlock_grbl", "gcode_mode" e "status" (representados no código da figura 4.43).

```
"# linha 1

        self.unlock_grbl = Button(self.label_men, text='Desbloquear\nCanal', command=self.controller_unlock,
                                  image=self.funlock, compound=TOP, width=100, height=75, font=("Malgun Gothic",
8),
                                  bg='white', borderwidth=0)
        self.unlock_grbl.grid(column=0, row=0)

        self.gcode_mode = Button(self.label_men, text='Idle\nCheck', command=self.get_gcode_mode,
                                 image=self.fidle, borderwidth=0, compound=TOP, width=100, height=75,
                                 font=("Malgun Gothic", 8), bg='white')
        self.gcode_mode.grid(column=1, row=0)
        self.status = Button(self.label_men, text='Estado\nda Operação', command=self.get_status,
                             image=self.fstatus, borderwidth=0, compound=TOP, width=100, height=75,
                             font=("Malgun Gothic", 8), bg='white')
        self.status.grid(column=2, row=0)
          "
```

Figura 4.43 - Programação dos botões da primeira linha do "Menu Principal": "Desbloquear Canal", "Idle/Check" e "Estado da Operação".

Cada um destes botões possui, como especificações, a *label master*, o texto a apresentar, o comando que executam ao clicar, uma imagem seguida de "compound" (para ambos, imagem e texto, aparecerem), uma largura e uma altura, a fonte do texto, a cor de fundo e a espessura do rebordo.

O aspeto final está representado na figura 4.44.

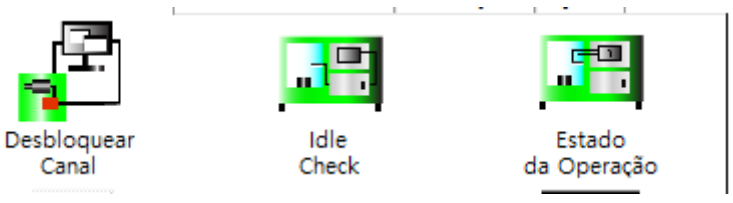


Figura 4.44 - Apresentação dos botões da primeira linha do Menu Principal.

A implementação de imagens nos botões é simples, apenas é realizada a transformação de uma imagem "png" para um objeto do código. Esta operação também é programável, sendo necessária a nomeação de cada imagem e especificação do diretório onde esta se encontra através do código "PhotoImage", como explícito na figura 4.45, da biblioteca "Tkinter", que promove essa transformação (Murmu, 2020).

```
"# ficheiros de imagem para os botoes tab1
self.fstartup = PhotoImage(file="fstartup.png")\
    .subsample(8, 8)
[…]"
```

Figura 4.45 - Criação de objetos de imagem para o "Menu principal".

No final, como o tamanho das imagens utilizadas foi de 500px por 500px, estas tiveram de ser redimensionadas de forma a caberem no botão, utilizando-se a ferramenta de redimensionamento de imagem "subsample(8,8)" que, neste caso, divide a largura e a altura da imagem oito vezes (Murmu, 2020).

A linha dois e a linha 3 de botões foram executadas exatamente da mesma forma, contemplando apenas especificações diferentes, como as imagens, o texto e os comandos a executar.

É de notar que todos os botões contemplam a mesma largura e altura, bem como o mesmo tamanho para cada imagem, o mesmo tipo de fonte (Malgun Gothic, tamanho 8) e fundo branco, isto torna a interface mais coerente e agradável para o utilizador. As imagens foram desenhadas em concordância com a máquina e demonstram pequenos símbolos relativos às operações que cada botão executa para captar a atenção do operador e tornar o processo mais intuitivo.

Foi ainda preparado um widget "Progressbar" para estimar a percentagem do documento que contém o código G que já foi enviado para o controlador. Para a formatação desta variável, "self.progresso", é necessária a estipulação da *label* onde se encontra, da orientação da mesma, "orient = HORIZONTAL", do comprimento da barra, do estilo (dos estilos existentes na biblioteca "Progressbar") e do modo, "determinate". Esta última função tem a ver com a leitura de números, isto é, se a variável é apenas temporal ou se é determinado o seu valor segundo uma função que envia o valor a colocar (de 0 a 100%). O código deste *widget* está representado na figura 4.46.

```
163     # STATUS AND TIME
164     self.statusbar = Label(self.label_men, text='Estado da Operação[%]:', bg='white', fg='black',
165                            font=("Malgun Gothic", 8))
166     self.statusbar.grid(column=0, row=3, columnspan=2, sticky='W')
167     self.estimated_time = Label(self.label_men, text='Tempo Estimado:', font=("Malgun Gothic", 8), bg='white',
168                                 fg='black')
169     self.estimated_time.grid(column=0, row=5, columnspan=2, sticky='W')
170     # PROGRESS BAR WIDGET
171     self.progress = Progressbar(self.label_men, orient=HORIZONTAL, style='black.Horizontal.TProgressbar',
172                                 length=390, mode='determinate')
173     self.progress.grid(column=0, row=4, columnspan=3)
```

Figura 4.46 – Programação da variável "self.progress".

Passando agora aos *widgets* inseridos no segundo *tab* ("self.label3 = ttk.Frame(self.tabControl)/ self.tabControl.add(self.label3, text='Comando Manual')"), "Comando Manual", estes objetos proporcionam ao operador da máquina a possibilidade de manusear os motores de cada eixo de movimento. É constituída por seis botões, uma caixa de texto e um *widget* "SpinBox" (Foundation, 2021).

Representados na figura 4.47, os botões para os comandos manuais estão inseridos na "label3_1" para ser possível a utilização da geometria "grid" (método matricial de organização). Cada botão tem as mesmas especificações: *label master*, imagem, comando, espessura do rebordo, estado e cor de fundo.

```
"# COMANDOS MANUAIS CNC
self.label3_1 = Label(self.label3, bg='white')
self.label3_1.place(x=80, y=5)
self.move_a_pos = Button(self.label3_1, image=self.photoa_mais, command=self.movea_pos,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_a_pos.grid(row=4, column=1)
[...]"
```

Figura 4.47 - Proramação dos botões dos comandos manuais da CNC.

- "self.movea_pos" para o botão "self.move_a_pos", que dá origem, visualmente, ao objeto descrito como "A+";
- "self.movea_neg" para o botão "self.move_a_neg", que dá origem, visualmente, ao objeto descrito como "A-";
- "self.movez_pos" para o botão "self.move_z_pos", que dá origem, visualmente, ao objeto descrito como "Z+";
- "self.movez_neg" para o botão "self.move_z_neg", que dá origem, visualmente, ao objeto descrito como "Z-";
- "self.movex_pos" para o botão "self.move_x_pos", que dá origem, visualmente, ao objeto descrito como "X+";
- "self.movex_neg" para o botão "self.move_x_neg", que dá origem, visualmente, ao objeto descrito como "X-".

Na especificação "image=", foram utilizadas as imagens geradas, tal como as previamente falas, pela biblioteca *"Tkinter"*, com a função "PhotoImage" (Murmu, 2020). As imagens foram criadas especificamente para o programa no *software* "Adobe CC Illustrator 2020".

Como os botões são implementados na definição "__init__", por definição, eles estariam ativos inicialmente. Porém, não é possível, por prevenção de erros e danos, a utilização de qualquer comando de código G antes do "Grbl" estar desbloqueado, por isso, o estado dos botões foi alterado para "DISABLED" ("state=DISABLED").

O fundo de cada botão é branco para não sobressair comparativamente ao fundo da *label* ("bg='white'") e, para aumentar a estética de apresentação, o rebordo ("borderwidth") é igualado a zero, para dar a sensação que apenas existe a seta da imagem e não um quadrilátero a envolvê-la (figura 4.48).

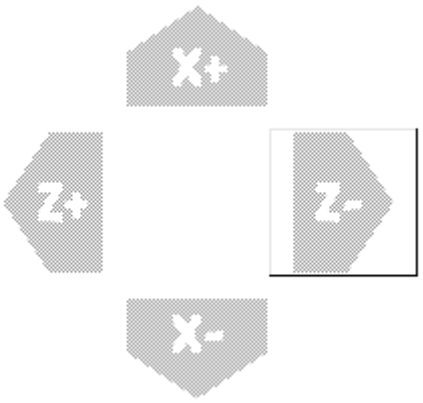


Figura 4.48 - Diferença entre valor zero e um de "borderwidth" dos botões.

Neste caso, da forma como os botões se encontram programados, existe apenas uma variável em jogo, nomeadamente, o valor inteiro, numérico que, é possível movimentar nos eixos. Na figura 4.49, está representado o processo de transmissão da informação, desde que o operador escolhe um valor para o movimento e clica no botão do eixo e sentido que pretende movimentar.

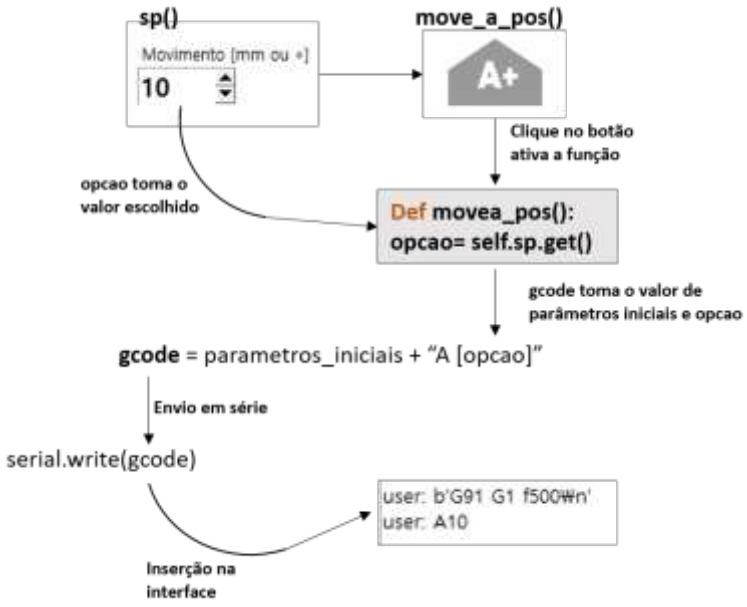


Figura 4.49 - Esquematização do desenvolvimento de informação após a escolha de um movimento de eixo.

A variável "self.sp", através da biblioteca *"Tkinter"*, origina uma "Spinbox", isto é, uma caixa de valores, neste caso de 0 a 100, onde é possível, a partir de duas setas, aumentar ou diminuir o valor pretendido com incrementos de uma unidade (Murmu, 2020).

Para referenciar a "Spinbox", foi criada uma *label*, "self.palavra_escolha" com o texto "Movimento [mm ou °]" dado que, dois eixos executam o movimento em milímetros (X e Z) e o eixo rotativo promove um movimento em graus (eixo A) (Murmu, 2020).

A "Spinbox" engloba variáveis como a *master label* ("self.label3"), o valor mínimo e máximo possível ("from_=0, to=100"), a largura do botão (neste caso 5) e a fonte.

Como esta se encontra numa posição não favorável à utilização de uma geometria matricial, foi utilizada a estruturação "place", onde se especificam os pixéis onde o canto superior esquerdo do *widget* se encontrará, dentro da *label master*.

Para a programação gráfica do submenu "Definições" organizaram-se duas *labels* principais, uma intrínseca à outra para, mais uma vez, ser possível utilizar diferentes parâmetros geométricos de organização. A "self.label1_1" é uma *frame* do *widget* "Notebook", responsável pela janela que o utilizador vê ao clicar na *tab* "Definições". Para a implementação de uma gama de botões específica dentro da *label* principal, foi criada uma *label*, "self.label1_2", posicionada pelo canto superior esquerdo da mesma, no ponto (200, 108) (Murmu, 2020).

Na janela "Definições" coexistem três botões principais que permitem visualizar, na janela de comunicação, os diferentes tipos de parâmetros predefinidos no "Grbl", um painel para alterar os parâmetros, um painel com as opções de "reset" do "Grbl", botões para utilizar o *homing* e o ventilador.

Os botões dos parâmetros foram programados tal como todos os outros botões de imagem, fazendo "compound" da imagem pretendida e do texto de cada um.

O objeto "self.startup_blocks", responsável por pedir ao controlador Arduino os blocos iniciais da máquina está construído para ter o texto "Blocos iniciais". Já o objeto "self.param_grbl", com o texto "Definições do Sistema" permite ao utiliador observar, na janela de comunicação, todos os parâmetros predefinidos para o diferente tipo de operações e funções da CNC como, por exemplo, velocidade e aceleração dos movimeno dos eixos, *homing*, unidade das medidas, posicionamento inicial dos eixos, etc.. A este botão estão interligadas duas funções, uma que fará aparecer todas as definições possíveis da máquina e outra, que mostra o "parser state" da CNC.

O *widget* "self.cardinal_param" remete para os parâmetros do código G entre o G54 ao G59 onde, se não ocorrer nenhum erro, o comando ligado ao botão fará aparecer no ecrã os valores sobre estes dados. Estes três botões estão definidos de igual modo, utilizando uma estrutura ".grid" para a colocação dos objetos sendo, assim, colocados apenas numa linha "row=0" em três colunas diferentes (0, 1 e 2). As imagens utilizadas são todas ficheiros ".png" para a possível orientação em pixéis, reproduzidas em Adobe CC illustrator como todas as imagens do *software*. Após a utilização da conversão para

"PhotoImage", estas imagens foram reduzidas à escala 1 para 8 em ambos os lados, como forma de ajustar às dimensões da interface.

Por fim, após a definição das imagens, do texto, da posição e da fonte da letra de cada botão, está exemplificado, na figura 4.50, o resultado visual final.

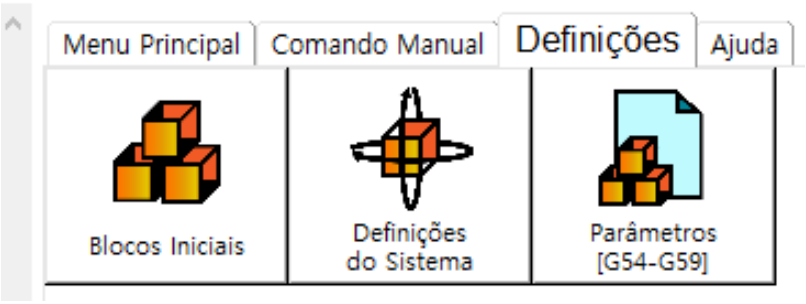


Figura 4.50 - Visualização dos botões de parâmetros programados.

Ainda neste separador de informação, estão organizados mais dois grupos de funções. Como já havia sido dito, uma parte remete para a modificação dos parâmetros predefinidos e outra que, engloba todas as opções de "Reset" da CNC, as opções de *Homing* e de Ventilação da máquina.

Como seria necessária uma estrutura matricial para a organização do teclado referente aos números responsaveis pela introdução do valor necessário para a alteração dos parâmetros da CNC e para os botões para o reset parcial ou total, homing e ventilação, foram então inseridas, dentro da *label* principal, duas labels secundárias para acoplar os restantes *widgets*.

A *label* "self.label1_1" contempla, após os botões de imagem , uma *label* apenas para enunciar um texto, "Escolher Parâmetro", para que, o utilizador saiba que, a partir desse ponto, estão as opções de escolha para modificar os parâmetros.

Seguidamente, é utilizado um *widget* da bibilioteca "ttk", "Combobox", para mostrar o *array* onde estão inseridos todos os parâmetros possiveis de alterar no "Grbl", de 0 a 136. Este *array*, representado na figura 4.51, é definido nos parâmetros iniciais do ficheiro "View" de modo que, mal ocorra o início do programa, os valores da "Combobox" sejam aumaticamente modificados para o valor do array.A variável "self.array" é dada como uma "list", comando que promove a criação de um intervalo de valores, neste caso de números inteiros. É utilizado o comando "range" como forma de automaticamente gerar um intervalo de zero ao número pretendido de inteiros, nomeadamente, 136 (Rodas de Paz, n.d.)(Foundation, 2021).

```
"self.array = list(range(136))"
```

Figura 4.51 - *Array* para a formação da lista de parâmetros que se podem escolher para alterar.

Como é possível observar no esquema da figura 4.52, a "ttk.Combobox" está também inserida na "self.label1_1", como texto inicial tem a palavra "Parâmetro". O mecanismo deste botão é extremamente simples: "values=self.array" define que os valores apresentados quando o operador clica na seta do botão correspondem aos valores do *array* (0 a 136), "textvariable=self.escolhido" faz com que o valor, escolhido pelo operador, seja guardado na variável "self.escolhido" para que, mais tarde, possa ser utilizado para modificar o parâmetro (Foundation, 2021).

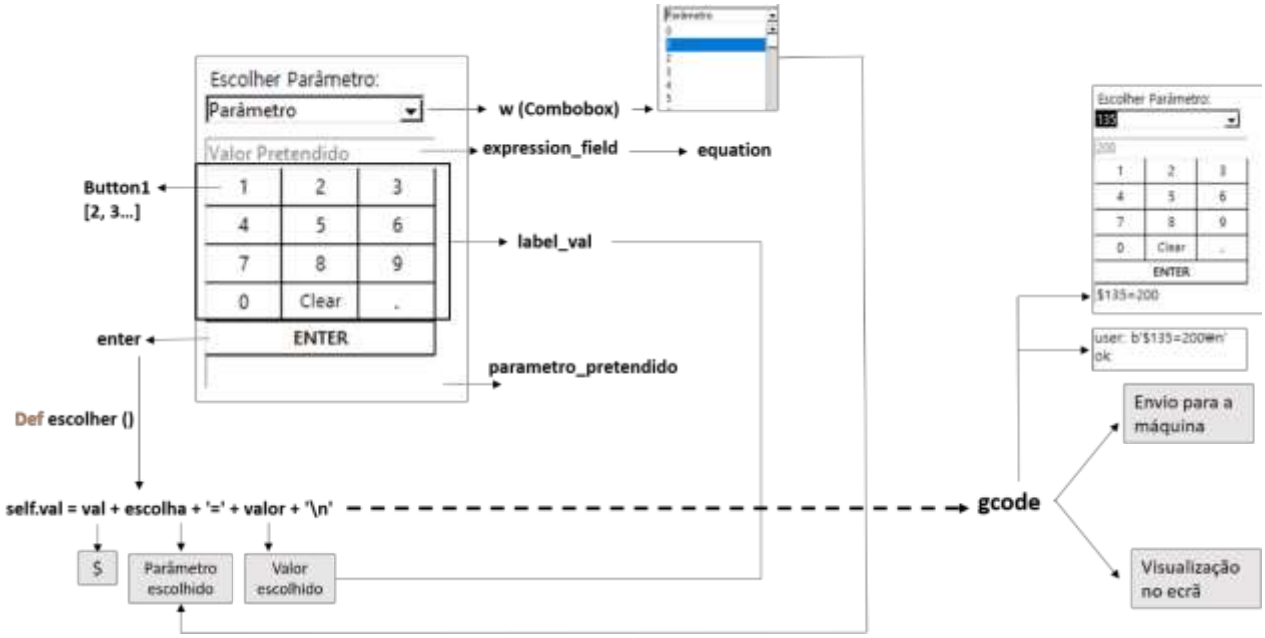


Figura 4.52 - Identificação das variáveis e objetos correspondentes e interligação dos mesmos para alterar parâmetros na interface.

Para que a primeira impressão do operador da máquina seja intuitiva, foi definido o texto "Parâmetro" como opção inicial da "Combobox" de modo que, qualquer pessoa que utilize a interface saiba que é naquela lista que se seleciona o parâmetro a alterar (Foundation, 2021).

Na *label* "self.*label*_val", posicionada no interior da *label* principal, estão inseridos todos os *widgets* responsáveis pela submissão do valor para o qual se pretende alterar o parâmetro previamente escolhido. Primeiramente, é disposto um *widget* "Entry" ("self.expression_field") que, promove a inserção de qualquer valor no espaço gerado. Neste caso, o valor inserido é dado pela variável "self.equation" ("textvariable=self.equation"). Este objeto está posicionado com uma estrutura ".grid", ou seja, orientada por colunas e linhas. Como se trata do primeiro objeto, encotra-se na coluna 0 e na linha 0, com um "columnspan=3", ou seja, este objeto pode ocupar três colunas sem modificar o tamanho dos outros *widgets*. Caso não fosse utilizada a função "columnspan", todos os objetos encontrados na primeira coluna teriam a mesma dimensão da caixa de texto ("width=21", ou seja, 21 pixéis), ficando, visualmente , como está representado na figura 4.53 (Rodas de Paz, n.d.).

Inicialmente o valor da variável “self.equation” é dado como “Valor Pretendido” (“self.equation.set('Valor Pretendido')”), para que, mais uma vez, o utilizador saiba o local exato onde irá aparecer o valor que selecionou.

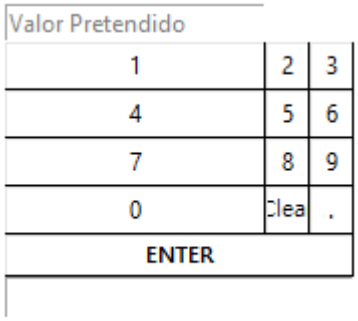


Figura 4.53 - Posicionamento das Colunas sem a utilização de "Columnspan".

Na figura 4.54, é possível observar como foram programados os botões do teclado em formato de “calculadora”. Todos dispostos de uma forma matricial 4x3, com inclusão dos botões correspondentes de 0 a 9, “Clear”, “.” e “Enter”. Os botões identificados como números foram programados de forma a representarem caracteres do teclado normal de um computador quando clicados, segundo a função “command=lambda: self.press(2)”, por exemplo, caso o operador carregue no botão com o número “2” que, fará a admissão do numero inteiro “2” para a variável “self.equation”.

Os botões “Clear” e ”.” foram programados para o caso do utilizador de enganar ou pretender adicionar números decimais aos valores. Estes pertencem à última linha da matriz, “row= 5”. Enquanto o botão “self.ponto” é igualmente utilizada apenas a função “self.press” para executar o comando, o botão “Clear” tem como comando a função “self.clear” que, visualmente apaga o valor inserido na caixa de texto “self.expression_field”. Este botão é necessário caso utilizador queira alterar o valor inserido ou mudar outro parâmetro para outro valor.

O botão “Enter”, “self.enter” ocupa inteiramente a linha inferior da “calculadora”. Com o comando “self.escolher”, este botão promove o envio do código pelo canal de ligação em série até ao controlador Arduino, para a mudança do parâmetro pretendido para o valor inserido. Neste botão não só é utilizada a função “columnspan” para que, visualmente, o botão possa ocupar três colunas da matriz sem alterar o comprimento dos outros *widgets* mas, também é utilizada a função “sticky="nsew"” (norte-sul-este-oeste) isto porque, deste modo, os botões ocupam o tamanho por completo dos blocos de linhas e colunas e não apenas o necessário para aglomerar os caracteres inseridos como texto.

Finamente, ainda na *label* “self.*label*_val” foi programada uma “Entry” que, permite a visualização da escolha por completo, ou seja, o parâmetro escolhido e o valor dado a esse parâmetro.

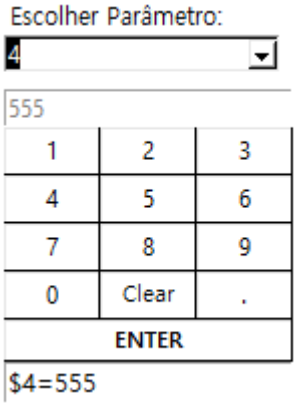


Figura 4.54 - Escolha e seleção do parâmetro.

No excerto de código da figura 4.55, identifica-se uma variável responsável pela inserção do número do parâmetro e valor na *Entry* "self.parametro_pretendido" que, localizada na sétima e última coluna da matriz, mostra, como dito, o valor total escolhido pelo operador para ser enviado ao "Grbl".

```
"self.variavel = StringVar()
self.parametro_pretendido = Entry(self.label_val, textvari-
able=self.variavel, fg='black',
                                  font=("Malgun Gothic", 10))
self.parametro_pretendido.grid(column=0, row=7, columnspan=3,
sticky='nsew')"
```

Figura 4.55 - Caixa de texto utilizada para demostrar o parâmetro a alterar e o valor correspondente.

No lado direito da janela "Definições", por baixo dos botões de imagem, encontra-se a *label* secundária "self.label1_2". Este painel foi posicionado, pelo ponto superior esquerdo da *label*, no ponto (200, 108) com dimensões de 220 px por 300 px.

Os botões e as *labels* de texto encontradas neste separador são todas estruturadas pela função ".grid", organização por blocos matriciais, onde, todos os *widgets* se encontram na mesma coluna.

Inicialmente é apresentado o texto "text='Reset Parcial ou total do "Grbl":'" programado na *label* "self.entry_reset" localizada na primeira linha e primeira e única coluna: "self.entry_reset.grid(row=0, column=0, sticky="nsew")".

Seguido do titulo, encontram-se três botões referentes à execução de três diferentes tipos de *resets* do "Grbl". Estes objetos, nomeados por "self.restore1", "self.restore2" e "self.restore3" estão, respetivamente, colocados na segunda, terceira e quarta linha da *label* secundária.

Cada um tem o texto referente à ação que executa, nomeadamente "RESET $$", "RESET G54-G59/G28-G30" e "RESET EEPROM" bem como um comando derivado dessa mesma função.

Por fim, foram programados dois botões e duas *labels* de texto, uma para anunciar os botões de "Homing" e outra para enunciar as opções do "Ventilador".

Para a programação do menu “Ajuda”, inserido no “self.tabControl” como “self.label5 = ttk.Frame(self.tabControl)” através do objeto *Frame* da biblioteca *“Tkinter ”*, foram importados três ficheiros de texto diferentes, com vista a inseri-los em janelas distintas, consoante a necessidade do operador. Como se observa no excerto de código da figura 4.56, dos *scripts* “error_list”, “botoes_def” e “parametros”, foram importados, respetivamente, os ficheiros de texto “lista_erros”, “Bt_def” e “param”.

```
“from error_list import lista_erros
from botoes_def import Bt_def
from parametros import param”
```

Figura 4.56 - Texto importado para submeter informação no menu "Ajuda".

Para ser possível utilizar os ficheiros no ambiente da classe “View”, estes têm de ser enunciados como variáveis iniciais. Assim sendo, os textos tomaram o valor das variáveis “self.texto_ajuda”, “self.Bt_def” e “self.ajuda_param” representados na figura 4.57.

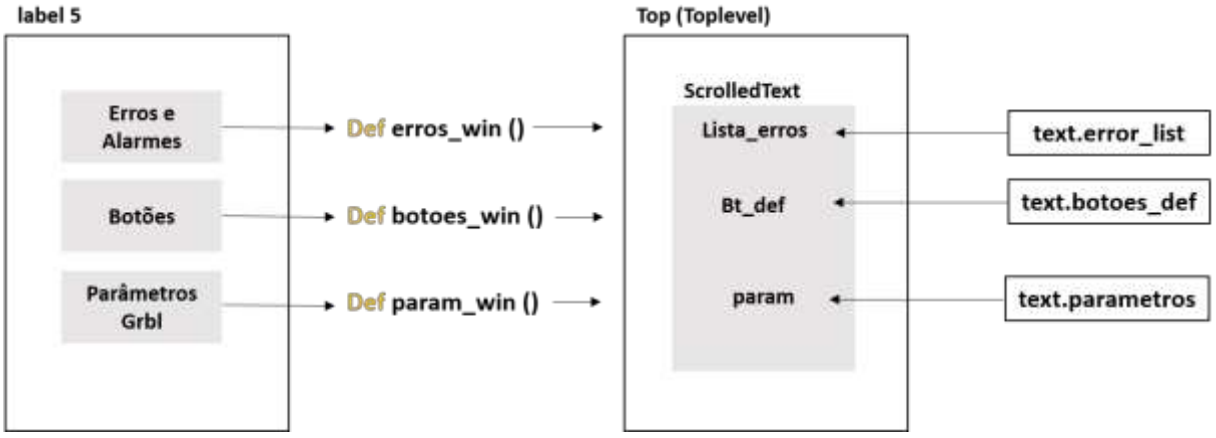


Figura 4.57 - Fluxo de informação e variáveis no menu "Ajuda".

Foram criados três botões para ser possível o acesso a esta informação. O botão “erros_button”, é definido a partir do objeto “Button” e contempla o texto “Erros e Alarmes”, de modo a que o operador saiba o que o botão executa. Tem como comando a função “self.erros_win”, responsável por fazer pop-up de uma janela com a definição dos erros e alarmes do “Grbl”.

Nesta Definição, é inicialmente inserida uma variável “Toplevel”, variável esta responsável por criar uma hierarquia de janelas (*frames* ou *labels*) dentro da plataforma GUI. A variável “self.top” define a “self.label5” como a *label* mestre do menu “Ajuda”: “self.top = Toplevel(self.label5)” (Rodas de Paz, n.d.).

Para dimensionar a janela que fará *pop-up*, de modo a ser coerente com a interface, foi definido um valor de “400x400” pixéis. De seguida, é criado um objeto *label* inserido na janela mestre “self.top” que, por sua vez, provém de um objeto de texto “tkst.ScrolledText” (Foundation, 2021).

A variável inicial “self.texto_ajuda” é inserida no objeto de texto, “self.erros1” e, por fim, é criado o botão “self.b1”, com o texto “voltar” que, permite ao utilizador fechar a página de informação.

Os botões “botões_button” e “param_grbl_button” foram programados de igual modo com a exceção do texto inserido para cada um que, tomou o valor das duas variáveis de texto restantes e, por uma questão visual, foram inseridos, através da função de posição “.place”, com um incremento de 120 pixéis na variável “y” de cada um, tornando possível a disposição representada na figura 4.58.

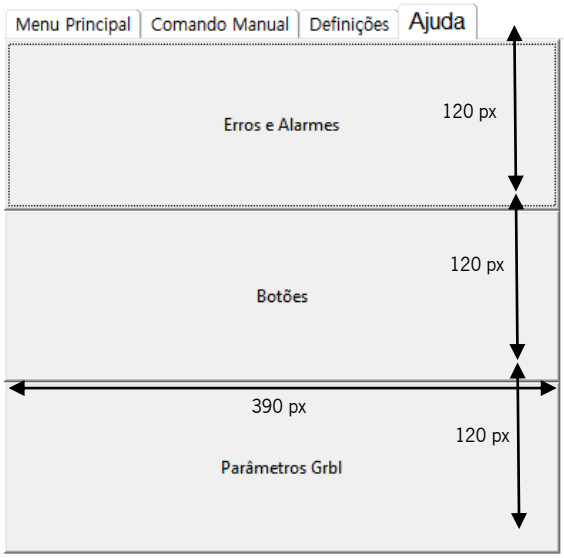


Figura 4.58 - Disposição linear dos botões do Menu Ajuda.

4.5.4. Compilação da interface e início de ciclo: main

Para finalizar a concessão do código, é realizada uma class exclusivamente para promover o correr do programa. Esta classe, “Main”, admite apenas a classe “View” como função a processar, como expresso no excerto de código da figura 4.59.

```
“#!/usr/bin/python3
from view import *

if __name__ == "__main__":
    “View”()”
```

Figura 4.59 – “Main” *script*.

A linha “#!/usr/bin/python3” garante que a versão do “Python” instalada que está a ser utilizada para correr o programa é o “Python3”.

Inicialmente é importado tudo (“*”) do *script* “view” para ser possível processar a classe “View” (com acesso a todas as bibliotecas e ficheiros importados anteriormente).

É utilizada a função “if __name__ == "__main__"” como forma de correr o programa. O “__name__” é uma variável interna que avalia o nome do módulo atual. No entanto, se um módulo estiver a ser executado diretamente (a partir da linha de comando), então “__name__” é definido como a *string*

“__main__”. Ou seja, ao executar este código, as variáveis tomarão valores iguais e, por isso, a função ““View”()” será executada. Abrindo, assim, a interface pretendida (Fooz, 2009).

4.6. Implementação da interface na máquina CNC e validação de resultados

Antes de implementar a Interface com o Controlador na CNC, foi necessário fazer alguns ajustes no equipamento, nomeadamente, a programação dos parâmetros da CNC, o *homing* e a produção de um prato rotativo e molde de blocos.

4.6.1. Programação dos parâmetros step/mm e aceleração dos eixos

Aquando a programação da interface para a máquina, foram encontrados erros nos parâmetros do “Grbl” relativos à mesma, tais como, erros nos “steps/mm” executados pela CNC e nas suas acelerações o que, resultava em movimentos não concordantes com os pedidos pelo utilizador.

Genericamente, a equação para calcular os steps/mm é o rácio entre os *steps* por revolução multiplicados pelos *microsteps* e os milímetros por revolução.

Assim, sabendo que os steps por revolução dos motores são 200 steps (1,8°) é apenas necessário encontrar os *microsteps* utilizados pelo fabricante e calcular os milímetros por revolução para cada caso. A fórmula geral para o cálculo dos steps/mm está representada na equação 1.

$$\frac{step}{mm} = \frac{(step/revolução) \times microsteps}{mm\ por\ revolução}$$

Equação 1 - Calculo dos steps/mm (Breiler, 2020a).

No eixo Z, a transmissão de movimento é executada a partir de um sistema de um motor Nema 23 (200 steps) que faz movimentar o eixo ao longo de duas guias através de um fuso de esferas com rosca trapezoidal de passo 2 mm. (igus, 2019)

Existindo *microsteps* aplicados correspondentes a dois, ver tabela de *microsteps* do apêndice II, a expressão teórica do valor step/mm do eixo Z está descrita na equação 2.

$$\frac{step}{mm} do\ eixo\ z = \frac{200x2}{2}\ step/mm$$

Equação 2 - Cálculo dos step/mm do eixo Z.

Assim, o valor do parâmetro “$102” do “Grbl” deverá ser alterado para 200 steps/mm, devendo ser este posteriormente acertado manualmente.

De seguida, para o cálculo deste parâmetro no eixo X (vertical), parâmetro “$100”, é necessário recorrer à contagem de dentes da polia que intersetam a correia, dado que o sistema de transmissão é resultado

de duas polias de dimensões iguais ligadas cada uma a dois motores Nema34 (200 step). As polias têm 8 dentes interlaçados na correia com um passo de 8 mm.

Posto isto, na equação 3, encontra-se a fórmula que remete para os mm/step do eixo X.

$$\frac{step}{mm} do\ eixo\ X = \frac{200\ \times 16}{8\ \times 8}\ step/mm$$

Equação 3 - Cálculo dos steps/mm do eixo x.

Com 16 *microsteps* ativos, ver tabela de *microsteps* do apêndice II, 8 dentes em contacto com a correia com 8 mm de passo, o valor de steps por milímetro do eixo X é então 44,444 steps/mm.

Por fim, para o eixo A (eixo de rotação), assistido por um motor Nema 23 (200 steps), o valor foi ajustado manualmente, dado a falta de informação sobre o sistema de transmissão que engloba duas polias e a correia. Teoricamente, o valor de steps é definido pela equação 4.

$$\frac{step}{mm} do\ eixo\ A = \frac{200\ \times 8\ \times 34}{14\ \times\ \pi 110}\ steps/mm$$

Equação 4 - Cálculo dos Steps/mm do eixo de rotação A (Aasvik, 2015).

Assim, com *microsteps* de 8 vezes o valor dos *steps* do motor (200), ver tabela de *microsteps* do apêndice II, 14 dentes da polia do motor e 34 da polia passiva e um diâmetro de 110 mm da polia passiva, tem-se um valor de steps/mm igual a, aproximadamente 11,244.

Apesar destes valores serem teoricamente calculados, após alguns testes na máquina, os parâmetros foram ajustados para valores aproximados dos teóricos.

Os parâmetros ajustados no "Grbl" , $100, $102 e $103, estão representados na figura 4.60, correspondentes aos steps/mm de X, Z e A, respetivamente (Breiler, 2020a).

```
$100=44.000
$101=250.000
$102=195.000
$103=10.520
```

Figura 4.60 - Steps/mm dos eixos X. Z e A.

Foram várias vezes retirados valores, a partir de pontos de referência a distâncias de 5, 10 e 20 mm para o eixo X e Z, de modo a garantir que estes se encontravam a andar, efetivamente, a medida requerida. O eixo A foi acertado a partir de rotações de 90, 180 e 360 graus com o auxílio de um transferidor.

Por fim, para testar a veracidade dos movimentos aquando a maquinagem, foi maquinado um joelho de um utente como teste de prova, representado na figura 4.61. Foram tiradas algumas medidas ao joelho para comparar com o resultado final.

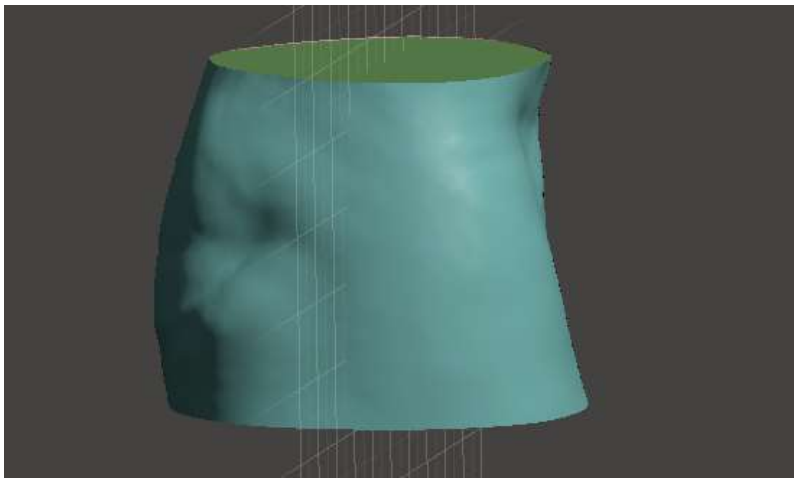


Figura 4.61 - Joelho para maquinar como prova de teste dos parâmetros (*Meshmixer*).

Foi utilizado um bloco de poliuretano de baixa densidade, através de um molde já existente, para realizar a maquinagem. Este bloco, como previamente se encontrava projetado apenas para dar origem a moldes positivos de cabeças de pacientes com plagiocefalia, tinha um diâmetro muito superior ao maior diâmetro do joelho, resultando num tempo de maquinagem muito superior ao esperado. Na figura 4.62, é possível observar os parâmetros utilizados, através do *software* "Deskproto", para maquinar a peça.

Para o desbaste optou-se por uma distância de passagem de seis milímetros para garantir que era desbastado todo o material. A distância deste parâmetro para o acabamento foi cinco milímetros, metade do diâmetro da ferramenta, para garantir um melhor acabamento à peça.

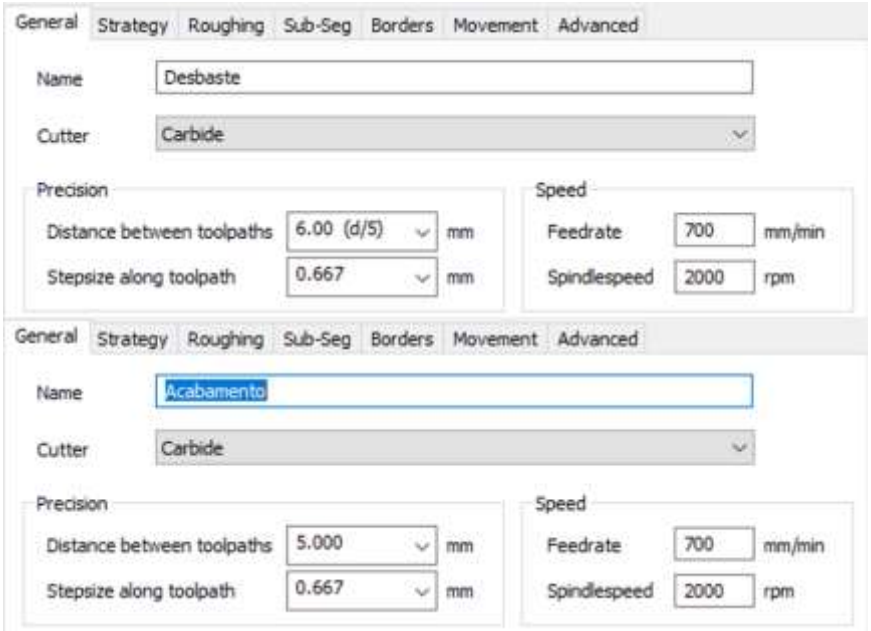


Figura 4.62 - Parâmetros utilizados no "Deskproto" para maquinagem do joelho.

Na figura 4.63, encontra-se o cálculo do tempo estimado pelo programa (uma hora e dezasseis minutos) e as dimensões do bloco de trabalho (identificado pela linha verde na simulação CAM dos processos de desbaste e acabamento).

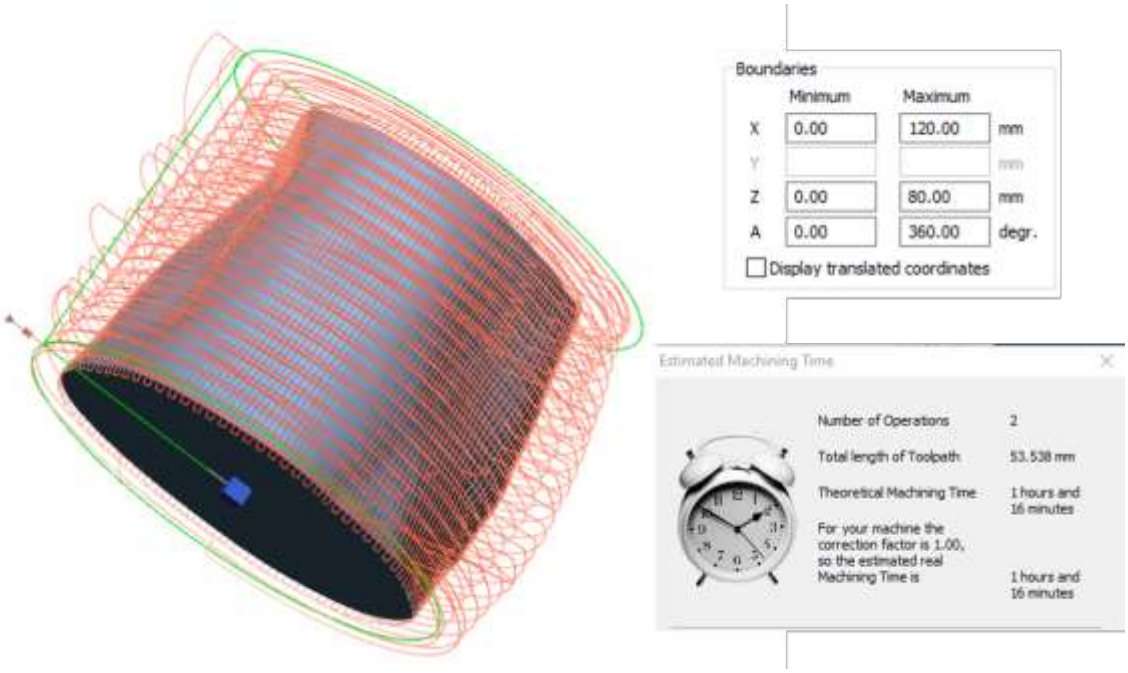


Figura 4.63 - Simulação do percurso da ferramenta de corte, tempo estimado de trabalho e dimensões do bloco inicial.

O joelho encontrava-se posicionado com a parte superior (parte da coxa) mais próxima do prato, para que existisse um maior suporte, evitando assim vibrações, e para que não houvesse interferência com os parafusos que agarram o bloco ao prato aquando a penetração da ferrament. Observando a figura 4.64, é possível observar o joelho maquinado.

Figura 4.64 - Molde positivo do joelho após a maquinagem.

Para verificar se as dimensões da peça final correspondiam às dimensões do modelo CAD foi utilizada uma escala para retirar várias medidas da peça, nomeadamente das bases do joelho e da sua altura.

No *"Meshmixer"*, as dimensões principais, a azul 122,5 mm, a vermelho 112,61 mm e a verde 142,2 mm, encontram-se representadas na figura 4.65.

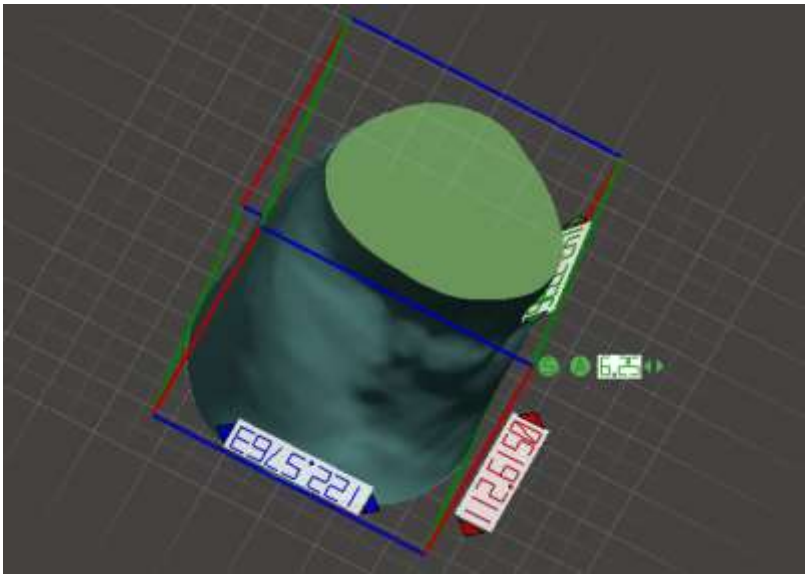


Figura 4.65 - Dimensões em X, Y e Z do modelo do joelho CAD.

Na medida equivalente à reta verde (142,2 mm), após a maquinagem, observou-se uma diminuição de 3 mm, porém, como representado na figura 4.66 (canto superior direito) a vermelho, a linha verde estipula a dimensão de um ponto na extremidade da face superior do joelho ao ponto mais externo da peça, neste caso, a parte da rótula e não a outra extremidade da face. Posto isto, a diferença de 142 mm para 139 mm pode dever-se a essa mesma distância. De notar que, esta medida foi retirada várias vezes devido à instabilidade do material que pode induzir em erro (apêndice IV).

A medida mencionada a azul (122 mm) está ligeiramente sobredimensionada, demonstrando um valor de 124 mm, como apresentado na figura 4.66. A instabilidade desta medida pode ter ocorrido por má medição (devido à visão) ou porque, aquando a maquinagem, houve necessidade de retirar a peça da máquina e voltar a marcar o zero do eixo Z. Mais uma vez, no apêndice IV podem-se verificar todas as medições realizadas.

Retirar a peça para marcar novamente o zero dos eixos pode afetar os diâmetros da peça pois, como é um procedimento realizado manualmente o erro humano (paralaxe) é um fator de erro muito comum na metrologia, sendo que um milímetro desviado do zero anterior pode resultar em dois milímetros de desvio no final da peça.

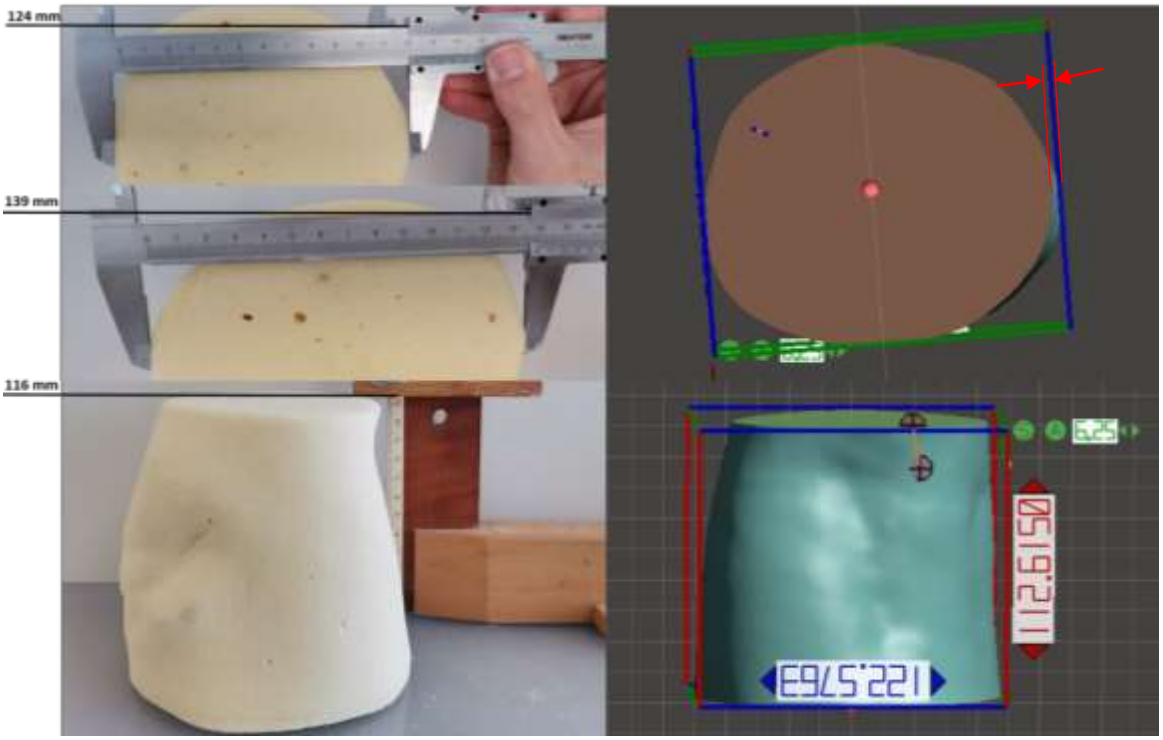


Figura 4.66 - Medidas obtidas na peça maquinada em comparação com as medidas do modelo CAD.

A altura da peça, a vermelho, apenas consequente do movimento do eixo X, tem a maior discrepância de todas, sendo a do modelo CAD 112,6 mm e a final de 116 mm. Porém, a peça foi cortada com um serrote manual, ficando marcada uma superfície visível, na figura 4.67, que provavelmente é a origem do excesso de milímetros.

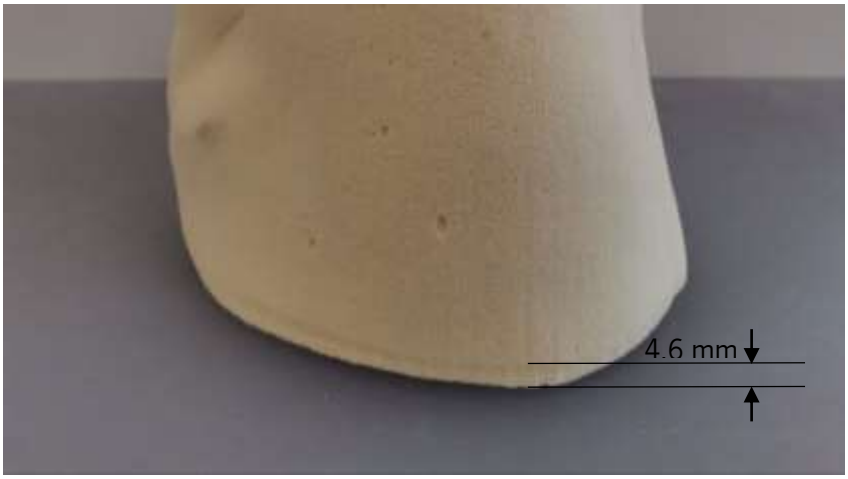


Figura 4.67 - Excesso de material na peça final cortada.

4.6.2. *Homing* do eixo de translação z

A realização do *homing* da máquina, foi implementada apenas no eixo de translação Z (negativa), com a inserção de um fim de curso e um batente.

O batente escolhido foi adquirido da revendedora "BotnRoll", *microswitch* 5A/250V (ref. CEL10007), que opera com um atuador de alavanca plana a temperaturas entre -25ºC e 85ºC. (OMRON, 2013)

No controlador Arduino, o pino que corresponde à inserção de um *switch* para o movimento negativo do eixo do Z (NO) é o pino 12. Para fechar o circuito, é necessário ligar à terra (COM), no pino GND. (Breiler, 2020b)

O *switch* está colocado numa placa de alumínio perpendicular ao plano de translação da ferramenta, como representado na figura 3.68.

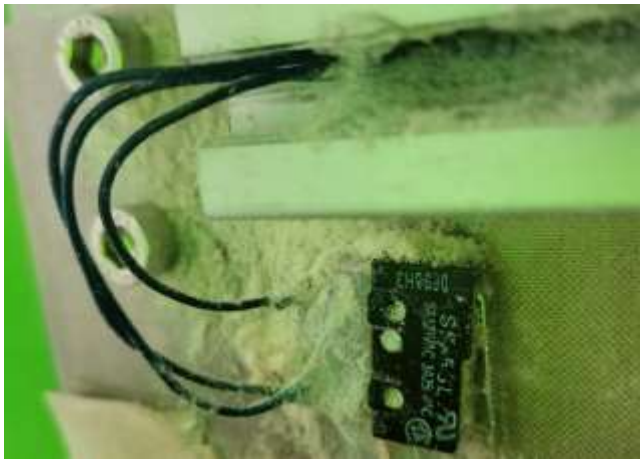


Figura 4.68 - Limit Switch SS implementado na CNC.

Por outro lado, foi implementada uma peça no perfil paralelo ao movimento do Z com um furo onde é inserido um parafuso (figura 3.69) que, mais tarde, na função de *homing*, ativará o fim de curso.

O parafuso, ao bater na alavanca plana, fecha o circuito entre o sensor e a alavanca, fazendo, assim, parar o eixo Z.

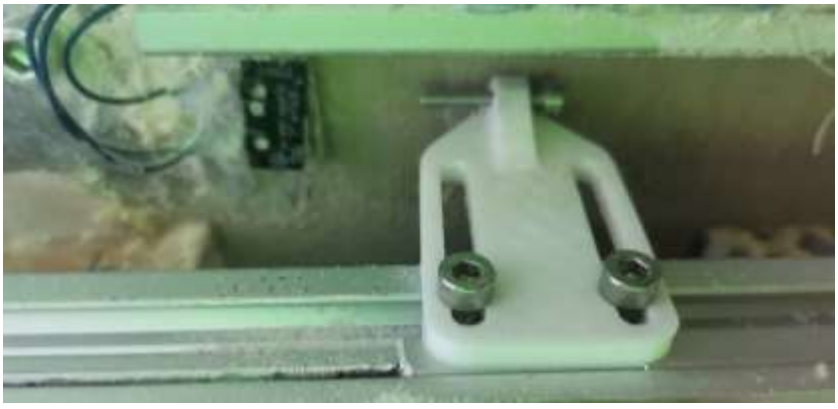


Figura 4.69 - Fim de curso e batente (parafuso).

Para facilitar o *homing* do eixo Z, que convém ser no centro da rotação do eixo A, ou seja, do bloco a maquinar. O batente ficou a uma distância de 230 mm do limite do perfil, de modo a que, a ponta da ferramenta, ultrapassasse o centro da área de trabalho por 15 mm. Como são apenas geradas superfí-

cies de revolução, não é necessária a passagem da ferramenta para coordenadas negativas em Z, todavia, para peças completamente desbastadas no topo, é necessária que esta atinga o ponto zero (calibrado como observado na figura 3.70 e figura 3.71) sem qualquer problema.

De modo ao processo funcionar, o parâmetro $22 do "Grbl" tem de ser "=1" e o $23 tem de corresponder a uma sequência em que o movimento seja para "Z-", neste caso, foi colocado "$23=4", ver tabela de "mask" do *homing* no apêndice I.

É ainda possível alterar a velocidade de *homing* no parâmetro $25. Esta ficou definida como 400 mm/min. O *homing* do eixo é simplesmente ativo com o comando "$Hz".

Figura 4.70 - Alinhamento do eixo do Z para marcar o zero peça.

Figura 4.71 - Alinhamento do eixo do Z (vista de cima) para marcar o zero peça.

Na interface foram criados dois botões para ativar a definição de *homing* e realizar o *homing* do eixo, sendo que, após o clique, a ferramenta move-se até à posição. Os botões estão representados na figura 4.72.

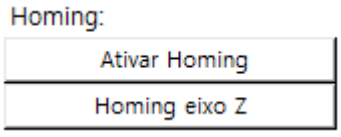


Figura 4.72 - Botões para ativar o homing ($22=1) e para fazer homing ao eixo do Z ($Hz), respetivamente.

Mais tarde, o objetivo é ativar todos os fins de curso possíveis do eixo, podendo, assim, colocar e retirar o bloco sem reprogramar os zeros da peça (podendo causar alterações no produto final).

4.6.3. Projeção e produção de um prato e moldes para geração de blocos de poliuretano

Como anteriormente falado, os blocos maquinados na CNC são de poliuretano de baixa densidade, uma "espuma" que, após arrefecida, solidifica e permite a moldagem de plástico por cima da mesma. O problema deste material é que é altamente reativo e, aquando a sua formação, cria pressões elevadas

nas paredes do recipiente que a molda em forma de cilindro. Como os moldes existentes se encontravam deformados devido a estas pressões, houve necessidade de criar uns novos. Porém, existia uma carência paralela de coexistir um tipo de bloco mais pequeno, para evitar o desbaste excessivo quando toca a maquinar peças com menores diâmetros e, por outro lado, moldes com uma altura mais elevada, para ser possível a maquinagem de moldes positivos de membros de maiores dimensões.

O molde existente tratava-se de duas partes de um cilindro que eram aparafusadas uma à outra e, ainda, ao prato que, posteriormente, se colocava na máquina. Após a solidificação do químico, eram retirados os moldes laterais, ficando apenas o bloco no prato, como representado na figura 4.73.

Figura 4.73 - Bloco de Poliuretano de baixa densidade no prato da CNC.

As dimensões dos blocos fabricados com este molde são, aproximadamente, 200 mm de diâmetro de base por 225 mm de altura. Este diâmetro, é necessário manter para a maquinagem de moldes positivos de cabeças, porém, no caso de se pretender realizar, por exemplo, o molde de uma perna, em que a parte superior da perna também necessita deste diâmetro, a altura deveria ser aumentada, de forma a reduzir a quantidade de vezes que se terá de dividir o membro para o maquinar na totalidade. Assim, mantendo o princípio de funcionamento deste molde, foi projetado um molde cilíndrico com 200 mm de diâmetro e com 300 mm de altura, representados na figura 4.74. Inicialmente era pretendida uma altura máxima de 400 mm, porém, não seria possível imprimir uma altura tão elevada nas impressoras 3D da empresa.

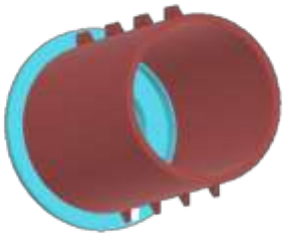


Figura 4.74 - Modelação CAD dos moldes cilíndricos com dimensões 200 x 300 mm.

Para a realização de blocos de menores dimensões, como pés ou joelhos dimensionou-se um molde com diâmetro de 120 mm de base e com 300 mm de altura, representados na figura 4.75.

Figura 4.75 - Modelação CAD dos moldes cilíndricos com dimensões 120 x 300 mm.

Ambos os moldes contemplam oito furos para os fixar um ao outro, quatro de cada um dos lados, com um diâmetro de furo de 4 mm.

Para a fixação de ambos os moldes à base, seria também desejável a projeção de um novo prato, que permitisse o acoplamento de qualquer um dos tamanhos definidos. Na figura 4.76, está representado o prato dimensionado. Este suporte contém duas saliências cilíndricas com dois centímetros de altura e cinco milímetros de espessura em cada diâmetro de molde, para evitar que escoe poliuretano quando este se encontra ainda líquido.

Figura 4.76 - Prato dimensionado para a fixação dos moldes e do bloco.

Tanto os furos que se encontram no exterior da saliência de maior diâmetro (dez furos) como da de menor (8 furos), são furos de 5 mm tais como os dimensionados na parte inferior dos moldes laterais, para que seja possível a fixação das peças.

O cone (de diâmetro máximo de 50 mm e mínimo de 30 mm), permite a melhor amarração do bloco ao prato, auxiliado por quatro parafusos M6 na sua extremidade. O veio utilizado para fixar o prato à máquina entra a uma profundidade de cinco centímetros do cone.

Na figura 4.77 é possível observar a vista em corte do conjunto de todos os elementos do sistema e do bloco formado pelos moldes.

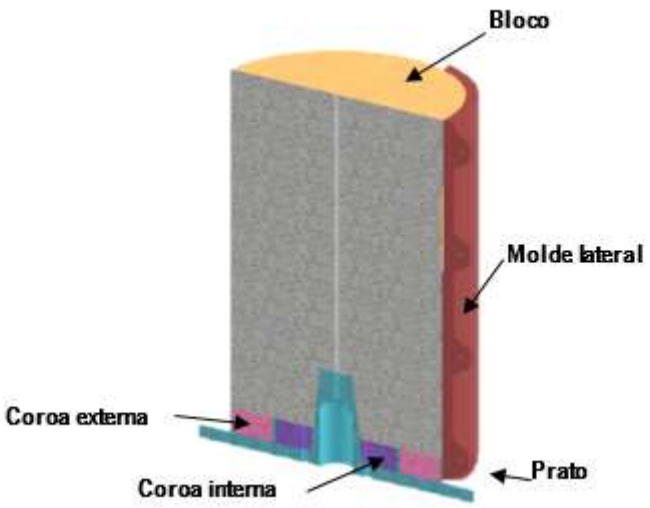


Figura 4.77 - Esquema de imagem CAD do sistema para a criação dos blocos de poliuretano de baixa densidade

Para segurança das paredes com dois centímetros de altura, foram ainda dimensionadas duas "coroas" para ocupar todo o fundo do prato, aumentando a estabilidade do processo. As coroas permitem que o bloco possua uma base estável, representada na figura 4.78 e não adira às saliências de baixa espessura, evitando que as danifique.

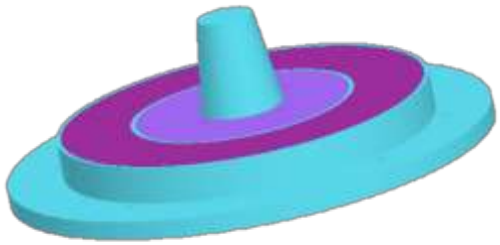


Figura 4.78 - Prato para colocação do bloco com as coroas inseridas.

Todas estas peças foram impressas em PLA com um *infill* de 20% a uma temperatura de 197 °C e uma velocidade de 3000 mm/min. O preenchimento das peças foi aumentado para evitar as deformações ocorridas nos moldes anteriores. A velocidade de maquinagem foi consideravelmente elevada para evitar que o tempo de fabrico de cada peça excedesse as trinta horas.

Este molde ainda não foi completamente testado dado a impossibilidade de imprimir as coroas da figura 4.78. Porém, moldes teste impressos pelos mesmos parâmetros foram testados e não deformaram.

4.6.4. Implementação da Interface na CNC

Com todos os componentes terminados, procedeu-se à introdução da interface na CNC. Primeiramente, foi impresso em PLA um suporte para o ecrã e Raspberry, retirado do *site*, *open-source*, "Thingiverse" dimensionado pelo utilizador "mkellsy" (Mkellsy, 2020). Para a fixação da capa e do suporte ao conjunto (RaspberryPi e ecrã tátil), utilizaram-se 16 parafusos M2 de dois centrimos de altura. O resultado pode ser observado na figura 4.79.

Figura 4.79 - Raspberry Pi4 modelo b e ecrã de 7" com suporte.

A capa e o suporte foram escolhidos pela facilidade de adaptação de novos suportes. No verso da capa, como representado na figura 4.80, os suportes são aparafusados por dois parafusos (em cada um).

Figura 4.80 - Parte de trás da capa do conjunto.

Estes componentes facilmente podem ser virados para suportar o ecrã como base numa superfície plana, como podem ser trocados por outro tipo de suporte, como, por exemplo, um sistema de guias para retirar o ecrã da parede da máquina quando necessário.

O facto da própria capa incluir um sistema de respiração para as alhetas do dispositivo dissiparem o calor e do furo para implementação de uma ventoinha foram também fatores que levaram à escolha desta capa dado que, o Raspberry, ficará várias horas ligado a acompanhar a maquinagem na CNC.

Após a implementação da capa e do suporte é hora de incluir a interface na máquina. Para evitar a furação da placa de acrílico da CNC, a mesa já existente foi posicionada ao nível do técnico que utilizará a máquina. Por fim, foi ligado o dispositivo no quadro elétrico da máquina e colocado na mesa, como observado na figura 4.81.

Figura 4.81 - Máquina "Nada" com interface implementada.

A interface liga automaticamente quando é ligado o botão da máquina (ver apêndice VII, manual de utilizador da máquina). Na figura, é possível observar os cabos de ligação ainda por fora do quadro elétrico. Iste deve-se ao facto de o quadro ainda não ter sido furado para colocação de fios e cabos, na medida que, tal operação só seria favorável, quando os cabos do projeto paralelo de implementação de um sistema de aspiração, fosse concluído para, desse modo, constatar o melhor local de passagem de cabos. Na figura 4.82, está representada a interface ativa já ligada ao controlador Arduino da máquina.

Figura 4.82 - Interface ativa na CNC.

4.6.5. Teste e validação da Interface na CNC da Padrão Ortopédico.

Como já havia sido relatado, a interface já contempla a ligação à porta do controlador Arduino especificada no código. Sendo assim, é apenas necessário ligar a máquina e testar a passagem de informação. O ecrã do Raspberry estava a ser partilhado com o computador (via VNC[19]) para ser possível a tiragem de capturas de ecrã do dispositivo, mas todas as operações foram realizadas no Raspberry.

O primeiro passo é verificar se o "Grbl" está a aceder ao canal. Na figura 4.83 encontra-se o sucedido após clicar no botão de "Desbloquear Canal" da Interface.

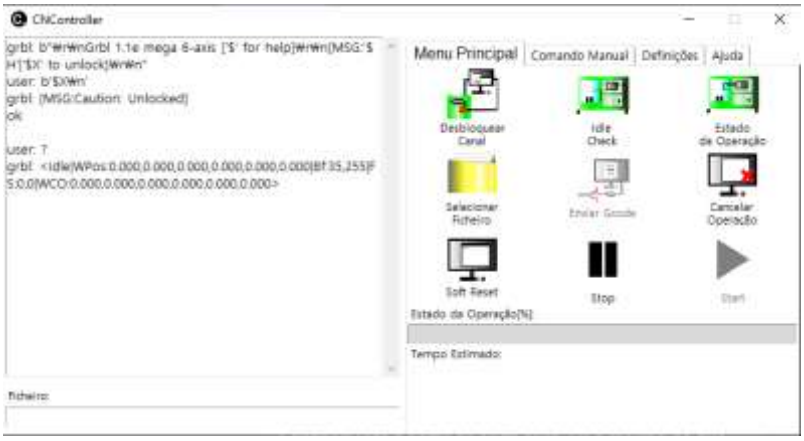


Figura 4.83 - Ecrã da Interface - operação de desbloquear canal e verificar estado de operação.

Como é possível observar, o "Grbl" enviou uma mensagem a dizer "ok" e "Caution: Unlocked", ou seja, "Cuidado: Desbloqueado". Assim, pode comprovar-se que o canal está a funcionar corretamente na CNC. É também constatável que o botão "Estado de operação" está em funcionamento sendo que foi clicado e o "Grbl" procedeu ao envio da informação presente no ecrã, na figura 4.83.

Seguindo os passos normais de preparação da máquina para efetuar uma maquinagem (ver anexo VII), procedeu-se ao movimento dos eixos X, Z e A, em coordenadas relativas, para posicionar a ferramenta e o prato, como representado na figura 4.84. Foi efetuado um movimento de 2 mm em cada eixo.

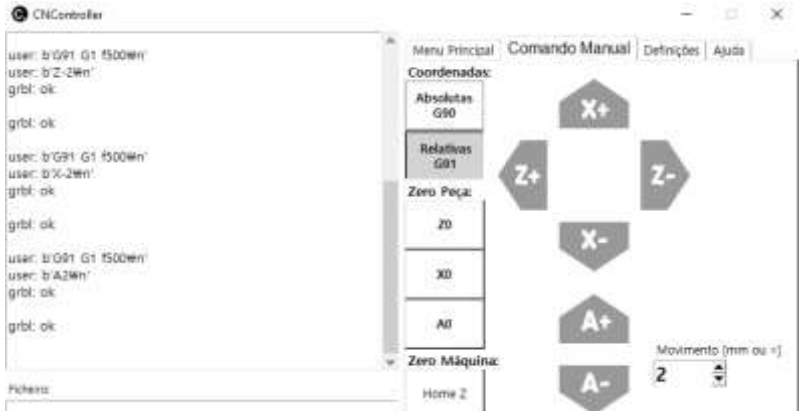


Figura 4.84 - Ecrã da Interface - Movimento em coordenadas relativas dos três eixos de trabalho.

[19] VNC é um *software* que permite, via *Wi fi*, partilhar o ecrã do Raspberry com um computador.

De notar que todos os movimentos foram enviados com as coordenadas selecionadas e que o "Grbl" respondeu com "ok", movendo-se dois milímetros em cada eixo. Foi, de seguida, clicado no botão de "estado de operação" o qual respondeu com as novas coordenadas de posição da ferramenta (2,2,2). Foram marcados os zeros peça (botões "Z0", "X0" e "A0") com o resultado representado na figura 4.85.

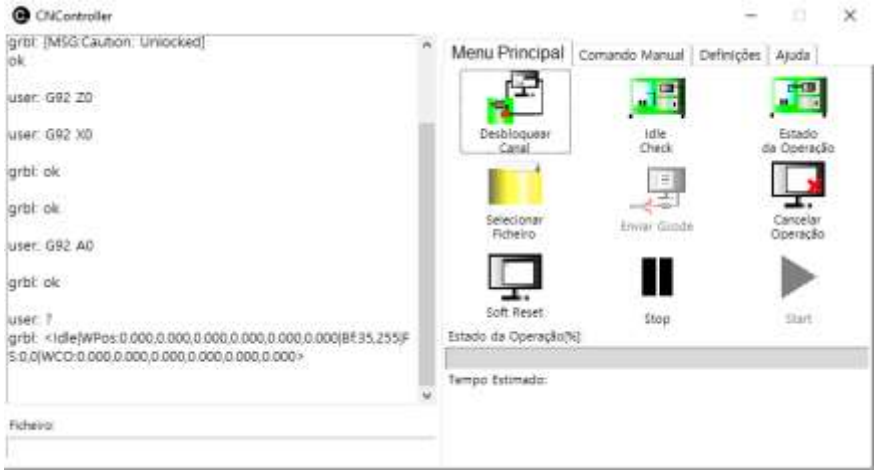


Figura 4.85 - Ecrã da interface - marcação de zeros peça e estado de operação.

Como demonstrado, após a marcação dos zeros, no estado de operação as coordenadas do sistema voltaram a ser representadas como (0,0,0).

Para testar o envio de coordenadas absolutas (e na medida de verificar os zeros), foram movidos os eixos cinco milímetros em coordenadas absolutas, como representado na figura 4.86.

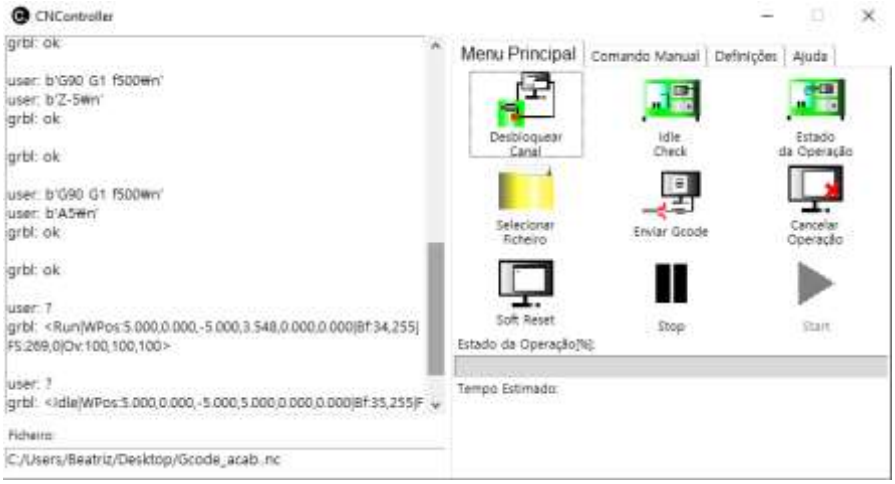


Figura 4.86 - Ecrã da Interface - Movimento dos eixos em coordenadas absolutas e estado de operação.

Foram então provados os movimentos em coordenadas absolutas ("G90" presente e recebido com "ok" pelo *firmware*). No eixo Z foram movidos 5 mm no sentido negativo e nos X e A no sentido positivo. Pelo estado de operação é notável que os eixos se moveram para a posição pretendida apesar de, no primeiro estado de operação o eixo A ainda não estava na posição quando foi pedida a informação à máquina. Já no segundo estado de operação, as coordenadas encontravam-se em (5,-5,5).

Por fim, foi enviado um ficheiro de código G para a CNC (figura 4.87). O código foi também enviado com sucesso e a CNC procedeu ao movimento dos eixos.

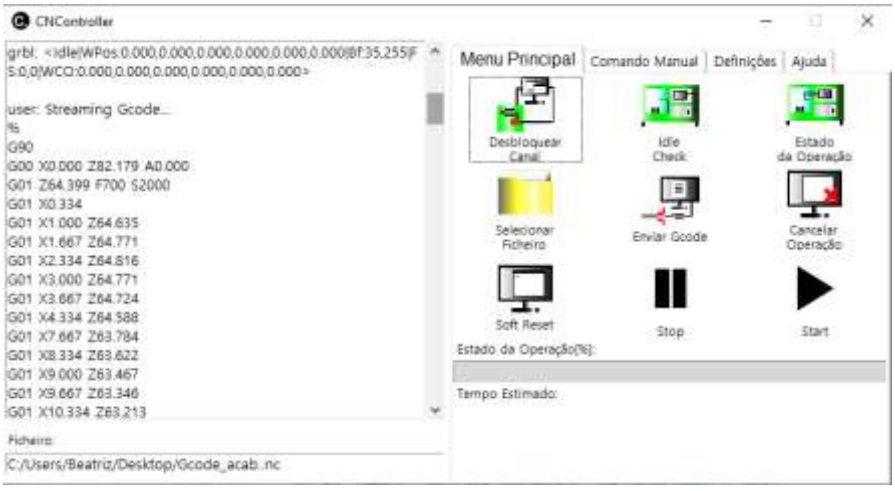


Figura 4.87 - Ecrã da interface - envio do código G para a CNC.

Como é possível observar, o código estava a ser enviado e, apesar de não ser possível ver o estado de operação no ecrã, para evitar conflito de informação (visto que a função pede ao "Grbl" um "Estado de operação" por segundo), é possível observá-lo na linha de comandos do Raspberry, como expresso na figura 4.88.

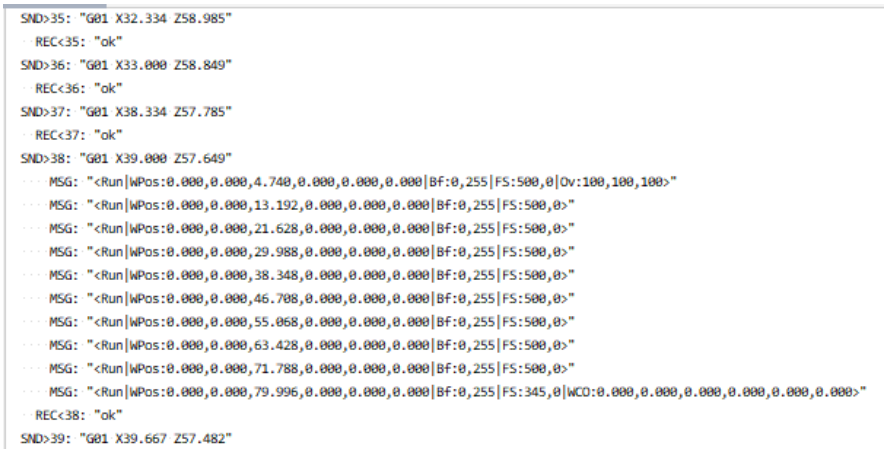


Figura 4.88 - Linha de Comandos Raspberry.

No estado, a máquina está no modo "Run" e as coordenadas estão a ser alteradas à medida que os eixos se movem na CNC. É também mencionado o estado do *buffer* do controlador Arduino ("Bf"). A interface está a esperar pelo "ok" do "Grbl" para enviar a linha de código seguinte sendo que, já só existem 0,255 *bits* de espaço no *buffer*.

Finalmente, foram testados os botões de “Stop” e “Start” aos quais, a CNC parou imediatamente o movimento dos eixos e o programa deixou de enviar ao código. (quando clicado o botão “Stop”). Ao clicar em “Start”, a CNC retomou o movimento e o “CNController” retomou o envio de código, como representado na figura 4.89.

```
 REC<21: "ok"

G-code streaming finished!
 Time elapsed:  0.10934925079345703

WARNING: Wait until Grbl completes buffered g-code blocks before exiting.
 Press <Enter> to exit and disable Grbl.<Run|WPos:0.000,0.000,0.000,4.776,0.000,0.000|Bf:15,255|FS:500,0|Ov:100,100,100>

<Run|WPos:0.000,0.000,0.000,13.232,0.000,0.000|Bf:15,255|FS:500,0>

<Run|WPos:0.000,0.000,0.000,21.684,0.000,0.000|Bf:15,255|FS:500,0>

<Run|WPos:0.000,0.000,0.000,30.136,0.000,0.000|Bf:15,255|FS:500,0>

<Run|WPos:0.000,0.000,0.000,38.588,0.000,0.000|Bf:15,255|FS:500,0>

<Run|WPos:0.000,0.000,0.000,44.868,0.000,0.000|Bf:15,255|FS:51,0>

<Run|WPos:4.396,0.000,0.000,45.000,0.000,0.000|Bf:16,255|FS:500,0>
```

Figura 4.89 - Linha de comandos: Final da maquinagem e espera da paragem dos eixos.

No final do envio de código, a CNC continua a enviar o estado de operação até os eixos efetuarem todos os comandos, como representado na figura 4.90.

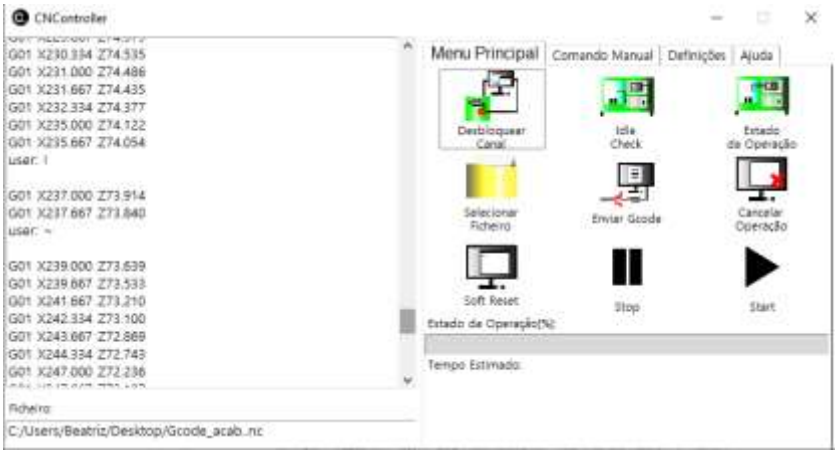


Figura 4.90 - Ecrã da Interface - Utilização dos botões de "Start" e "Stop".

Posto isto, a principal funcionalidade do ecrã está terminada, sendo que, é possível executar uma maquinagem, tanto pela parte de preparação da máquina como pelo envio de código G. De notar que, este processo foi realizado sem a utilização de envio manual de comandos para a máquina, aliás, a interface não possui sequer uma entrada para tal função.

A alteração de parâmetros não foi realizada, porém, já havia sido testada e os resultados encontram-se no apêndice V, “Composição gráfica da interface”.

5. Conclusões e Trabalhos futuros

Neste capítulo serão retiradas as conclusões sobre o estudo e o trabalho realizado sobre a máquina CNC da empresa Padrão Ortopédico e a projeção da interface homem máquina concebida. Serão ainda constatados alguns tópicos que deverão ser melhorados futuramente, com a utilização do dispositivo.

5.1. Conclusões

Com o término deste projeto, é possível dar os objetivos inicialmente propostos como compridos. No fim, foi inserida uma interface gráfica na CNC da empresa Padrão Ortopédico, capaz de controlar a máquina sem o auxílio de objetos e dispositivos externos. De notar que, apesar dos objetivos principais cumpridos, existe ainda um grande caminho pela frente no que toca a automatização da máquina "Nada".

A nível do estudo dos componentes da máquina, catalogou-se quase todos os seus componentes com as respetivas fontes de aquisição. Todavia, por falta de informação registada, alguns dos componentes foram apenas dimensionados e catalogados através da medição manual das suas dimensões. É de notar que existem componentes da máquina que se tornam secundários no seu funcionamento, por exemplo, os motores do eixo X e do eixo A poderiam ser reduzidos apenas a um por cada eixo.

Olhando para o processo de maquinagem da CNC, teria sido muito mais fácil e menos moroso se, o criador e única pessoa que sabia manusear a máquina, tivesse estado presente para realizar uma prova exemplo, ou mesmo para tirar dúvidas que surgiram aquanto os testes que se realizaram para descobrir quais os parâmetros importantes e como se operava no dispositivo. Contudo, o estudo da maquinagem na CNC foi completado e, desta vez, foi descrito todo o processo necessário para a maquinagem, todas as peças e parâmetros importantes bem como, redigido um "Manual de Utilizador" para qualquer operador saber o que vai encontrar na máquina e como a manusear.

A um nível mais fulcral do projeto, na inserção e programação da interface gráfica homem-máquina, é possível, felizmente, dizer que o programa ficou concluído, na medida que, dois dos técnicos da empresa responsáveis pela utilização da máquina, ambos sem qualquer formação em máquinas industriais, já se encontram a operar a CNC a partir da interface, "CNController", criada. Pelo que, garante que o principal objetivo do programa, criar uma interface gráfica intuitiva para o utilizador, foi cumprido.

No que toca à escolha da parte física do projeto, ou seja, o Raspberry Pi 4, modelo b e o respetivo ecrã tátil de sete polegadas, foi, sem dúvida, uma melhor escolha do que a utilização de um controlador Arduino. A única observação a ser feita é o tamanho do ecrã. Talvez, um ecrã de maiores dimensões

fosse benéfico para que o utilizador visualizasse melhor os dados da interface e para clicar nos botões do programa. Esta questão ainda foi pensada várias vezes, porém, inicialmente não havia um orçamento aceitável para um ecrã de maiores dimensões, sendo que o custo aumenta a uma taxa elevada. Este controlador, devido às restrições encontradas na máquina CNC, foi a melhor opção possível, podendo o dispositivo substituir totalmente utilização de um computador na manipulação da máquina, afirmando a autonomia da mesma.

Como é sabido, foram alterados alguns parâmetros cruciais do "Grbl" *firmware* (controlador da máquina). Quando se efetuou o primeiro teste com o intuíto de realizar um molde de um membro inferior de um paciente, observou-se que os *steps* por milímetro do aperelho não estavam a fazer corresponder as distâncias da máquina às distâncias pedidas no ficheiro de código G. Como foi representado, esses parâmetros foram alterados ("$100", "$102" e "$103" do "Grbl"), seguindo-se várias medições das peças obtivas para comprovar que o problema estaria resolvido.

Infelizmente, não foi possível apresentar, para este trabalho, as peças do molde imprimidas na totalidade, devido à ocupação das impressoras 3D da empresa. Por essa razão, não foi permitido o teste das paredes do molde aquando a formação do bloco de poliuretano, para garantir que não sucumbiam à pressão. Ainda assim, o *infill* das peças foi aumentado em 10 % e, peças teste anteriores, exatamente com as mesmas especificações, mas de uma altura inferior, foram utilizadas na manufatura de moldes e não sofreram quaisquer alterações.

A implementação do *homing* do eixo Z foi muito favorável para o funcionamento da máquina, retirando muitos passos no processo de maquinagem (agora apenas dois passos para marcar o zero peça do eixo Z). Ainda assim, seriam necessários mais estudos para ser possível a total automatização do processo de todos os zeros da peça.

Apesar de ser possível alterar ficheiros de código diretamente na interface, seria benéfico o utilizador conseguir abrir o ficheiro e guardá-lo com as alterações realizadas. De notar que, esta interface está construída de forma possuir sinergia com o programa de geração de código G que a empresa possui.

Após todo o trabalho realizado, foi preparado um artigo relativamente à conceção da interface para a máquina CNC, "Study, Design and Development of a Man Machine Interface for a CNC", submetido para o jornal científico "Periodica Polytechnica Mechanical Engineering". O artigo encontra-se no apêndice VIII.

5.2. Trabalhos futuros

Tal como qualquer programa para inserir num dispositivo, existem sempre alterações a realizar ao longo do tempo. À medida que a máquina for utilizada e esmiuçada pelo operador, vão sempre ocorrer necessidades diferentes pois, quanto maior o conhecimento, maior a quantidade de parâmetros necessários de alterar na máquina. Por isso, um dos próximos passos a tomar é a inserção de novos botões e funcionalidades na máquina, aquando a necessidade por parte dos técnicos da mesma.

Pequenas funções, não extremamente necessárias, mas agradáveis para a utilização do GUI, seria colocar o tempo de maquinagem no ecrã, o que, ainda não acontece porque não foi possível determinar uma ligação entre o tempo de maquinagem real e o estipulado no *software* "Deskproto".

A nível de funções que à partida já deveriam estar implementadas, nomeadamente, a conexão dos fins de curso no eixo X (positivo e negativo) e no eizo Z (negativo), deveria ser o próximo passo neste projeto. Após a sua implementação, será possível fazer o homing total da máquina mal a CNC é ligada o que, para além de dar ao operador um ponto de partida, permite calcular diretamente o zero peça no eixo do Z e do eixo do X, eliminando praticamente todos os passos da preparação da máquina. No que toca a simplicidade, esta tarefa iria facilitar de tal modo a maquinagem que, apenas seria necessário ligar a máquina, fazer o *homing*, ligar a *spindle* e enviar o ficheiro, operações estas que, demoram um máximo de cinco minutos.

Na medida de melhorar o funcionamento da máquina, ao descartar um dos motores "desnecessários" na mesma, seria favorável e lucrativo a troca do motor (ou seja, venda) e a compra de um sistema de deteção de posição.

Seria ainda desejável um ecrã de maiores dimensões, para facilitar a utilização dos botões (aumentar o tamanho dos botões).

Em termos de atualização da interface, o botão "Reset G54-59" encontrado no separador "Definições" deveria ser alterado para "Zero Peça" sendo que ele promove a definição para as coordenadas de trabalho predefinidas.

Por fim, apesar de não ser necessária, seria agradável ter sempre a opção de enviar comandos diretamente através de uma linha de comandos, já que, o Raspberry Pi4 permite a ligação direta de um teclado e de um rato ou até mesmo a instalação de um teclado no próprio ecrã. Por isso, a implementação de uma nova secção com uma entrada para comandos.

Referências bibliográficas

Aasvik, M. (2015). *Tutorial: Calibrating Stepper Motor Machines with Belts and Pulleys*. https://www.norwegiancreations.com/2015/07/tutorial-calibrating-stepper-motor-machines-with-belts-and-pulleys/

ACIERA Iniciação ao comando numérico das máquinas ferramenta. (n.d.).

Aguiar, F. B. (n.d.). *PLAGIOCEFALIA – O QUE É?*

AliExpress. (2020). *Nema23/57 integrado híbrido servo motor cnc kit ddcsv3.1 4-axis sistema de controle de movimento dc fonte de alimentação parada de emergência mpg*.

BotnRoll. (2020a). *ARDUINO MEGA 2560 R3*. https://www.botnroll.com/pt/arduino-controladores/53-arduino-mega-2560-8058333490083.html

BotnRoll. (2020b). *TFT TOUCH 3.2'' - ARDUINO MEGA SHIELD (DESCONTINUADO)*. https://www.botnroll.com/pt/lcds-e-displays/585-tft-touch-28-itdb02-28.html

Breiler, J. (2020a). *Grbl Settings*. *1*. https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration

Breiler, J. (2020b). *Set up the Homing Cycle*. https://github.com/gnea/grbl/wiki/Set-up-the-Homing-Cycle

Case, J. (2003). *Arduino to Arduino Serial Communication*. https://robotic-controls.com/learn/arduino/arduino-arduino-serial-communication

Chang, P. P. Q. (2020). *An Approach to Plagiocephaly in Infants and the Role of Helmet Therapy*. https://specialty.mims.com/topic/an-approach-to-plagiocephaly-in-infants-and-the-role-of-helmet-therapy-?topic-grouper=cme

EMS. (2019). *Types of 3D Scanners and 3D Scanning Technologies*. https://www.ems-usa.com/3d-knowledge-center/3d-scanning-knowledge-center/3d-scanning-technical-papers/

Fooz, M. (2009). *What does if __name__ == "__main__": do?* https://stackoverflow.com/questions/419163/what-does-if-name-main-do

Foundation, P. S. (2021). *tkinter.ttk — Tk themed widgets*.

GeeksforGeeks. (2020). *Python Tkinter – ScrolledText Widget*. https://www.geeksforgeeks.org/python-tkinter-scrolledtext-widget/

Healy, A., Farmer, S., Pandyan, A., & Chockalingam, N. (2018). A systematic review of randomised controlled trials assessing effectiveness of prosthetic and orthotic interventions. In *PLoS ONE* (Vol. 13, Issue 3). https://doi.org/10.1371/journal.pone.0192094

igus. (2019). *Fuso trapezoidal drylin®, rosca direita, aço C15*.

Jin, Y. A., Plott, J., Chen, R., Wensman, J., & Shih, A. (2015). Additive manufacturing of custom orthoses and prostheses - A review. *Procedia CIRP*, *36*, 199–204. https://doi.org/10.1016/j.procir.2015.02.125

Liechti, C. (2018). *pySerial Documentation*.

Lusardi, M. (1994). Orthotics and Prosthetics in Rehabilitation. In *Age and Ageing* (Vol. 23, Issue SUPPL. 3). https://doi.org/10.1093/ageing/23.suppl_3.S28

Machine Tool Products. (n.d.). *Milling Machine CNC Retrofit Kits*. https://www.machinetoolproducts.com/category/milling-machine-cnc-retrofit-kits/

Mkellsy. (2020). *Raspberry Pi 4B Touchscreen Case by mkellsy September 30, 2020*. thingiverse.com/thing:4610955?fbclid=IwAR1dCdHe9-eS68yZxk-lCdTkSxFRLRP1fOzXTUgNzrHWTV_oX3cYaoQxS-g

Murmu, N. (2020). *Basics For Displaying Image In Tkinter Python*. https://www.c-sharpcorner.com/blogs/basics-for-displaying-image-in-tkinter-python

Neves, I. (2015). *A Plagiocefalia Posicional*. https://osteopraxis.wordpress.com/2015/07/02/a-plagiocefalia-posicional/

OMRON. (2013). *Subminiature Basic Switch Offers High Reliability and Security*. 1–8.

Pop, D., & Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*, *69*, 1172–1179. https://doi.org/10.1016/j.proeng.2014.03.106

Quintero-Quiroz, C. (2017). *Materials for lower limb prosthetic and orthotic interfaces and sockets: Evolution and associated skin problems*. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-00112019000100117#c1

Raspberry Pi (Trading) Ltd. (2019a). *Raspberry Pi 4 Model B Datasheet*.

Raspberry Pi (Trading) Ltd. (2019b). *Raspberry Pi Touch Display*. https://www.raspberrypi.org/products/raspberry-pi-touch-display/

Raspberry Pi (Trading) Ltd. (2019c). *Raspberry Pi Touch Display*. https://www.raspberrypi.org/documentation/hardware/display/

Raspberry Pi (Trading) Ltd. (2020). *Raspberry Pi OS*. https://www.raspberrypi.org/software/

RASPBERRY PI FOUNDATION. (n.d.). *What is a Raspberry Pi?* https://www.raspberrypi.org/help/what-is-a-raspberry-pi/

Rdpowers. (2016). *CNC-Mini-Mill/Firmware/"Marlin"-"Marlin"_v1/*. https://github.com/CCHS-Melbourne/CNC-Mini-Mill/tree/*master*/Firmware/"Marlin"-"Marlin"_v1

Rodas de Paz, A. (n.d.). *Tkinter GUI Application Development Cookbook_ A practical solution to your GUI*

development problems with Python and Tkinter. Packt Publishing.

rodin4d. (2018). *Model S*. http://rodin4d.com/en/Products/manufacturing/Milling-Machines

Silva, I. R. (2021). *Modelação da Máquina P.O.*

Smid, P. (2003). *CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming*. https://books.google.pt/books?hl=pt-PT&lr=&id=JNnQ8r5merMC&oi=fnd&pg=PA1&dq=CNC+definitions&ots=P-MJOU6QAQ&sig=2-t6YYKrvhq9K0_EBsiVKg4uba4&redir_esc=y#v=onepage&q&f=false

Suh, S.-H., Kang, S.-K., Chung, D.-H., & Stroud, I. (2008). *Theory and Design of CNC Systems - (Springer series in advanced manufacturing)*. https://doi.org/10.1007/978-1-84800-336-1

Tavares, J. M. R. S., & Relvas, C. (2012). *Introdução ao Controlo Numérico Computorizado – I Conceitos Gerais Bibliografia*. 1–14.

Thomas, P. C. (2021). *Understanding CNC Machining*. https://www.thomasnet.com/articles/custom-manufacturing-fabricating/understanding-cnc-machining/

van Hattem, R. (2020). *Welcome to Progress Bar's documentation!* https://progressbar-2.readthedocs.io/en/latest/

"Vorum". (n.d.). *"Vorum" Prosthetics and Orthotics Carvers*. https://"Vorum".com/op-carvers/

YAMADA, T. (1982). Manufacturing Automation. In *Journal of the Japan Society of Precision Engineering* (Vol. 48, Issue 7). https://doi.org/10.2493/jjspe1933.48.945

Zenie, J. R. (2016). *Prosthetic Options for Persons with Upper-Extremity Amputation*.

Apêndice I. Parâmetros e definições do Grbl

O “Grbl” é um *firmware* para comunicação em série, numa banda de 115200, 8 bits, sem paridade e com um bit de paragem (Breiler, 2020a).

Os comandos “$” servem para alterar as definições e parâmetros do *firmware* e para pedir ao “Grbl” informações sobre os mesmos (Breiler, 2020a).

Na fig. 1, encontram-se descriminados os parâmetros principais do “Grbl” *firmware* (Breiler, 2020a).

“$$”	Ver definições do Grbl;
“$#”	Ver os parâmetros (#);
“$G”	Ver estado da análise;
“$I”	Ver informação de Versão;
“$N”	Ver blocos iniciais;
“$x=valor”	Alterar o valor das definições do Grbl;
“$Nx=linha”	Alterar blocos iniciais;
“$C”	Alterar modo de leitura do código G;
“$X”	Desbloquear o modo de alarme;
“$H” ;	Executar o homing
“~”	Iniciar ciclo de código;
“!”	Interromper o ciclo de código;
“?”	Verificar o estado atual do controlador;
“ctrl-x”	Executar reset ao firmware.

Apêndice - Figura 1 - Parâmetros gerais do Grbl firmware.

O parâmetro “$C” serve para alterar o modo de leitura do “Grbl ntre “IDLE” e “CHECK”. Todos os restantes parâmetros têm de ser utilizados no modo Idle, à exceção de “!”, “~” e “?” que são permitidos aquando a maquinagem, ou seja, no modo “Run” (Breiler, 2020a).

Existem 136 parâmetros de valores para definições, sendo que, parte desses parâmetros são booleanos (0 ou 1) e outros têm definida uma máscara para efetuar a sua função com modos ou características diferentes (Breiler, 2020a).

Nas tab. 1, 2 e 3, encontram-se descriminados os parâmetros do “Grbl” à exceção dos valores para definição dos três eixos secundários possíveis (Breiler, 2020a).

Apêndice - Tabela 1 - Definições, valor padrão, unidade e número dos parâmetros do Grbl (Breiler, 2020a).

Parâmetro $x = val	Valor padrão	Unidade	Definição
$0	10	µs	Pulso de step (microssegundo)
$1	25	ms	Delay de leitura de step

Apêndice - Tabela 2 - Definições, valor padrão, unidade e número dos parâmetros do Grbl (Breiler, 2020a) (cont.)

Parâmetro $x = val	Valor padrão	Unidade	Definição
$2	0	mask	Inversão da porta dos steps
$3	6	mask	Inversão da direção
$4	0	bool	Ativar inversão de steps
$5	0	bool	Inversão do pino de limite de eixo
$6	0	bool	Inversão de pino de procura do limite
$10	3	mask	Reportar situação da máquina
$11	0.020	mm	Desvio de junção
$12	0.002	mm	Tolerância de arco
$13	0	bool	Reportar em inches
$20	0	bool	Limites da maquinagem
$21	0	bool	Limites da máquina
$22	0	bool	Ativar homing
$23	1	mask	Inversão do sentido de homing
$24	50	mm/min.	Feed rate do homing
$25	635	mm/min.	Feed rate de procurar limite
$26	250	ms	Tempo e fechar o circuito de fim de curso
$27	1	mm	Recuo após fim de curso

Apêndice - Tabela 3 - Definições, valor padrão, unidade e número dos parâmetros do Grbl (Breiler, 2020a). (cont.)

Parâmetro $x = val	Valor padrão	Unidade	Definição
$100	800	Step/mm	Steps por milímetro em x
$101	800	Step/mm	Steps por milímetro em y
$102	800	Step/mm	Steps por milímetro em z
$110	635	mm/min	Distância máxima por unidade de tempo em x
$111	635	mm/min	Distância máxima por unidade de tempo em y
$112	635	mm/min	Distância máxima por unidade de tempo em z
$120	50	mm/s^2	Aceleração em x
$121	50	mm/s^2	Aceleração em y
$122	50	mm/s^2	Aceleração em z
$130	225	mm	Máximo deslocamento de X
$131	125	mm	Máximo deslocamento de Y
$132	170	mm	Máximo deslocamento de Z

As "Mask", sombreadas a amarelo, são parâmetros que têm um intervalo de valores que efetuam a função de formas diferentes (Breiler, 2020a).

Para "$2", estão representados os valores da máscara para cada situação na tab. 4. A coluna X, Y e Z, refere-se à inversão de direção do pulso dos motores, neste caso, "N" se não estão invertidos e "Y" para estarem invertidos (Breiler, 2020a).

Apêndice - Tabela 4 - "Mask" para os valores de inversão de step e direção do movimento (Breiler, 2020a).

Valor	"Mask"	X	Y	Z
0	00000000	N	N	N
1	00000001	Y	N	N
2	00000010	N	Y	N
3	00000011	Y	Y	N
4	00000100	N	N	Y
5	00000101	Y	N	Y
6	00000110	N	Y	Y
7	00000111	Y	Y	Y

No caso do parâmetro "$3", para alterar a direção de movimento de motores, a tabela é idêntica à representada na figura acima (Breiler, 2020a).

Na "Mask" para "$10", é selecionado o valor que retorna as características pedidas pela função "?". O operador pode escolher se quer ver todos os parâmetros ou apenas um, ou o conjunto de dois, por exemplo (Breiler, 2020a).

Os principais valores definidos estão representados na tab. 5 (Breiler, 2020a).

Apêndice - Tabela 5 - "Mask" para o relatório de estado do Grbl (Breiler, 2020a).

Report Type	**Value**
Machine Position	1
Work Position	2
Planner Buffer	4
RX Buffer	8
Limit Pins	16

Na coluna da direita está representado o tipo de relatório que se vai receber e à esquerda o valor a dar ao parâmetro (Breiler, 2020a).

Por fim, a última "Mask", "$23" remete para a inversão da direção do *homing*. Na tab. 6, encontram-se os valores a atribuir ao parâmetro e o que cada um fará (Breiler, 2020a).

Apêndice - Tabela 6 - "Mask" para os valores de direção de homing (Breiler, 2020b).

Homing direction	Value
X+ Y+ Z+	0
X- Y+ Z+	1
X+ Y- Z+	2
X- Y- Z+	3
X+ Y+ Z-	4
X- Y+ Z-	5
X+ Y- Z-	6
X- Y- Z-	7

No que toca aos parâmetros para eixos secundários, A, B e C (como representados pelo Grbl), são sempre os valores "$" seguintes, por ordem descrita, dos valores para o eixo "Z". Por exemplo, no que toca a steps/mm, o eixo z comporta o valor "$102", o eixo A "$103", B "$104" e C "$105".

Apêndice II. Componentes da Máquina CNC “Nada”

Componentes da CNC e especificações dos mesmos representadas na tab.7, 8 e 9.

Apêndice - Tabela 7 - Tabela de Componentes da CNC e características (23HS45-3504S - Nema23.Pdf, n.d.; 34HS59-5004S - Nema34.Pdf, n.d.; AliExpress, n.d.; BotnRoll, 2020a; Stepper Online, 2017; worten, 2020).

Nome	Referência	Quant.	Corrente (output)	Ddp	Dimensões (máx.) e Especificações
Spindle	Spindle CNC 2200 W	1	10 A	220 V	80x80x281 mm^3
Motor eixo A	Nema 23 Bipolar 3Nm 23HS45-3504S	2	3.5A	4,2 V	57x57x114 mm^3
Motor eixo X	Nema 34 Bipolar 13 Nm 34HS59-5004S	2	5 A	5 V	25x37x150 mm^3
Motor eixo Z	Nema 23 Bipolar 3Nm 23HS45-3504S	1	3.5A	4,2 V	57x57x114 mm^3
Controlador Arduino	MEGA 2560 REV3 A000067	1	20-50 mA	5 V	53,3x51,52 mm^2
Cabo para ligação	USB 2.0 TipoA/TipoB MITSAI 6896944	1	-	-	3 m
Fontes de alimentação	S-350-60	3	5,9 A	60 V DC	215x115x50 mm^3
Driver eixo A	Digital Stepper Drive DM542T	2	1,0 A – 4,2 A	20-50 V DC	25,5x75,5x118 mm^3
Driver eixo Z	Digital Stepper Drive DM542T	1	1,0 A – 4,2 A	20-50 V DC	25,5x75,5x118 mm^3
Driver eixo X	Digital Stepper Drive DM860I	2	2,4 A – 7,2 A	20-80 V DC	42,5x97x115 mm^3
Fonte de tensão da Spindle	Inversor tensão de entrada 110-220 V YL620 A	1	10 A	110-220 V AC	Não especificado
Colett	ER25	1	-	-	26x26x34 mm^3 Ø = 10 mm
Bomba de água	80 W HY - 3500	1	0,36 A	220 V	Não específicado

Apêndice - Tabela 8 - Tabela de componentes da CNC e características (igus, 2019).

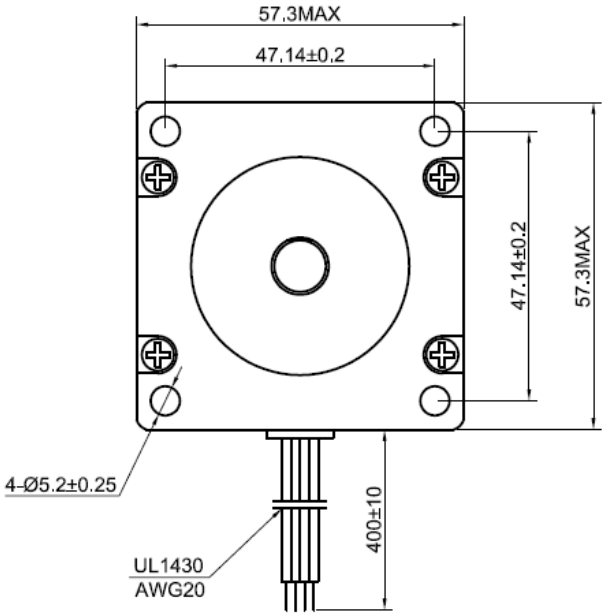
Nome	Referência	Quant.	Dimensões	Dentes	Passo	Carga (máx.)
Polia eixo X	HJZ-Drive 0634223857570	2	-	14	8 mm	-
Polia eixo A	Não especificado	1	-	34	10 mm	-
Pinhão eixo A	HJZ-Drive 0634223857570	1	-	14	10 mm	-
Correia eixo A	Não especificado	1	-	-	10 mm	-
Correia eixo X	Não especificado	1	-	-	10 mm	-
Fuso de esferas	DLE-SA-0006	1	Carruagem = 100 mm Curso = 500 mm	-	2 mm	Axial=750 N Radial=2000 N

Apêndice - Tabela 9 - Especificações da ferramenta de corte.

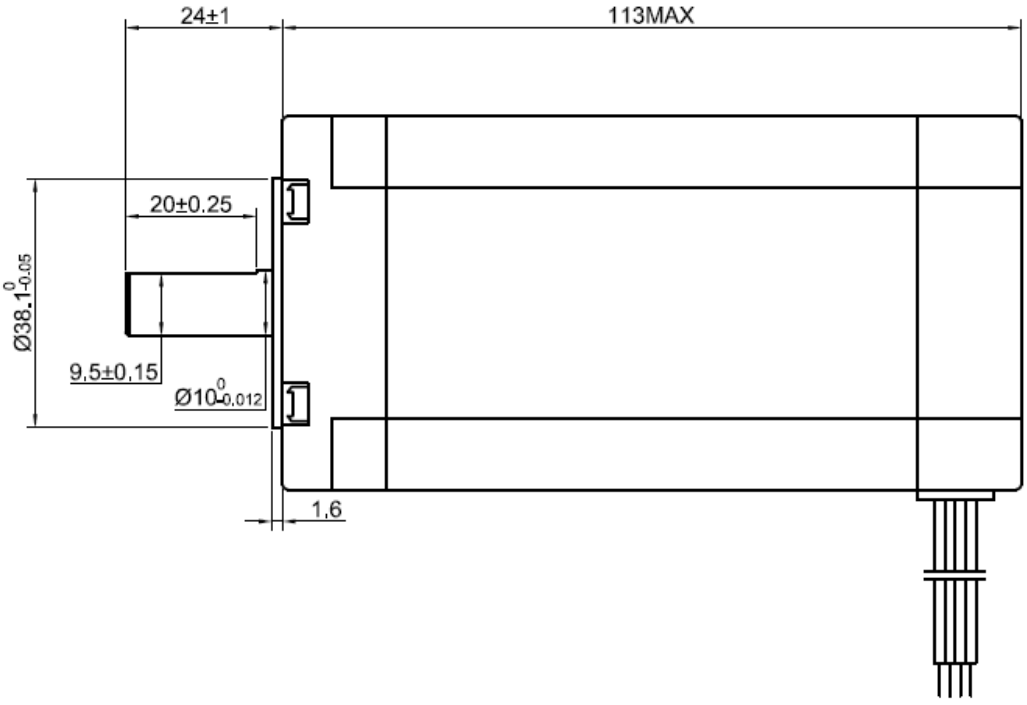
Componente	Referência	Formato	Curso total	Comprimento de corte	Flautas	Diâmetro
Ferramenta de Corte	Não encontrada	Ponta esférica	200 mm	100 mm	4	10 mm

- **Motor Nema 23**

As dimensões do motor estão descritas nas fig. 2 e 3, em milímetros.



Apêndice - Figura 2 - Desenho técnico do motor nema 23 (mm).



Apêndice - Figura 3 - Desenho técnico do motor nema 23 (mm).

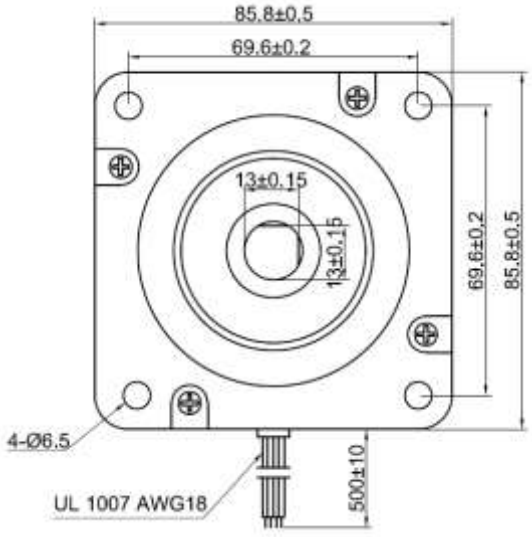
As especificações do produto pelo vendedor estão presentes na tab. 10.

Apêndice - Tabela 10 - Especificações do motor Nema 23.

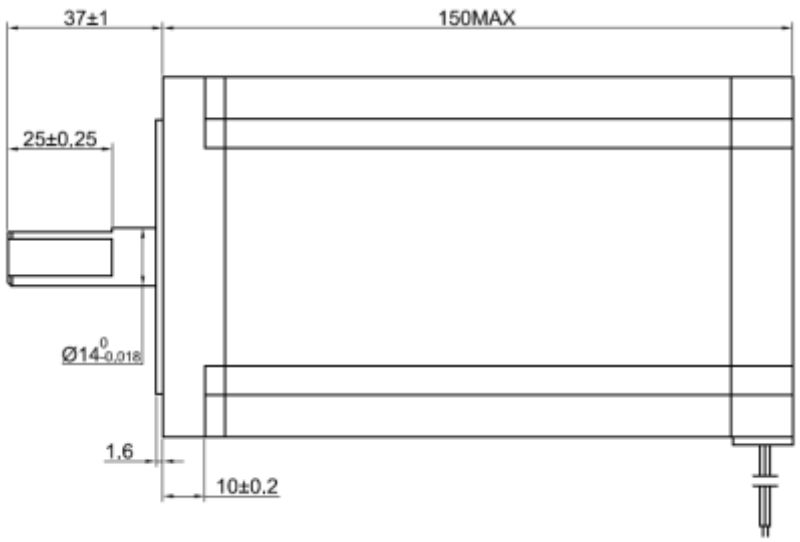
SPECIFICATION / CONNECTION	BIPOLAR
AMPS/PHASE	3.50
RESISTANCE/PHASE(Ohms)@25°C	1.10±10%
INDUCTANCE/PHASE(mH)@1KHz	4.00±20%
HOLDING TORQUE(Nm)[lb-in]	3.00[26.55]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	——
WEIGHT(Kg)[lb]	——
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

- **Motor Nema 34**

As dimensões do motor estão descritas nas fig. 4 e 5, em milímetros.



Apêndice - Figura 4 - Desenho técnico do motor nema 34 (mm).



Apêndice - Figura 5 - Desenho técnico do motor nema 34 (mm).

As especificações do produto pelo vendedor estão presentes na tab.11.

Apêndice - Tabela 11 - Especificações do motor Nema 34.

SPECIFICATION / CONNECTION	BIPOLAR
VOTAGE(VOC)	5.00
AMPS/PHASE	5.00
RESISTANCE/PHASE(Ohms)@25°C	1.00±10%
INDUCTANCE/PHASE(mH)@1KHz	11.00±20%
HOLDING TORQUE(Nm)[lb-in]	13.00[115.06]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	3600.00
WEIGHT(Kg)[lb]	5.00[11.02]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL;FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm（UNDER NORMAL TEMPERATURE AND HUMIDITY）	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

- **Drivers para motores do eixo A e Z**

Na tab. 12 encontram-se representadas as ligações possíveis para os diferentes microsteps.

Apêndice - Tabela 12 - Microsteps dos drivers dos motores do eixo A e Z.

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

- **Drivers para motores do eixo X**

Na tab.13 encontram-se representadas as ligações possíveis para os diferentes microsteps.

Apêndice - Tabela 13 - Microsteps dos drivers dos motores do eixo X.

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	ON	ON	ON	ON
4	800	OFF	ON	ON	ON
8	1600	ON	OFF	ON	ON
16	3200	OFF	OFF	ON	ON
32	6400	ON	ON	OFF	ON
64	12800	OFF	ON	OFF	ON
128	25600	ON	OFF	OFF	ON
256	51200	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
200	40000	OFF	OFF	OFF	OFF

Apêndice III. Definição de CNC e características

A maquinagem segundo o conceito CNC é utilizado no quotidiano da Industria, principalmente na área da manufatura (Thomas, 2021).

O termo "CNC" significa "Controlo Numérico Computorizado" e Máquina CNC designa um sistema mecânico que incide em remover material de uma dada peça (peça de trabalho ou bruta) no qual as suas ferramentas são controladas a partir de um computador (Thomas, 2021).

A maquinagem por CNC é comummente utilizada numa vasta gama de materiais, nomeadamente, metais, plásticos, madeira, vidro, espuma e compostos (Thomas, 2021).

A grande diferença entre CNC e máquina CNC é que, enquanto CNC apenas define o controlo numérico assistido por computador, a máquina CNC é uma máquina programável que é capaz de realizar as operações de maquinagem de forma autónoma. Todavia, é natural os autores e técnicos dirigirem-se a este tipo de máquinas apenas pelo termo "CNC" (Thomas, 2021).

Os processos de manufatura por remoção de material são frequentemente apresentados em contraste de processos de adição (como impressão 3D) ou processos de deformação, como injeção de moldes.

Apesar da maquinagem por CNC permitir produção de alta precisão e exatidão de peças simples a um bom custo de mercado, com o aumento da complexidade das peças a fabricar, a relação custo-benefício é limitada (Thomas, 2021).

Tipos De Máquinas CNC

A Maquinagem por CNC é um processo de fabrico adequado para uma ampla variedade de indústrias, incluindo industria automóvel, aeroespacial, construção e agricultura. Capaz de produzir uma variedade de produtos, como estruturas de automóveis, equipamentos cirúrgicos, motores de avião e ferramentas manuais e de jardim. O processo abrange várias operações diferentes de maquinagem controladas por computador, nomeadamente, processos mecânicos, químicos, elétricos e térmicos - que removem o material necessário da peça de trabalho para produzir uma peça ou produto personalizado. Os processos mecânicos mais comuns nas máquinas CNC são a furação, fresagem e torneamento. Como a máquina em estudo é uma máquina CNC fresadora, este tópico vai ser mais especificado em termos de componentes da máquina (Thomas, 2021).

A fresagem emprega ferramentas rotativas de corte multiponto para moldar a peça de trabalho. As ferramentas de fresagem são orientadas horizontal ou verticalmente e incluem fresas de topo, fresas helicoidais e fresas de chanfro (Thomas, 2021).

O processo de fresagem CNC também utiliza máquinas de fresagem habilitadas por CNC, conhecidas como fresadoras, que podem ser orientadas horizontal ou verticalmente. As fresas básicas são capazes de movimentos de três eixos, porém, modelos mais avançados permitem a utilização de eixos adicionais

(secundários). Os tipos de fresas disponíveis incluem fresagem manual, fresagem plana e fresagem universal (Thomas, 2021).

Sistema De Eixos

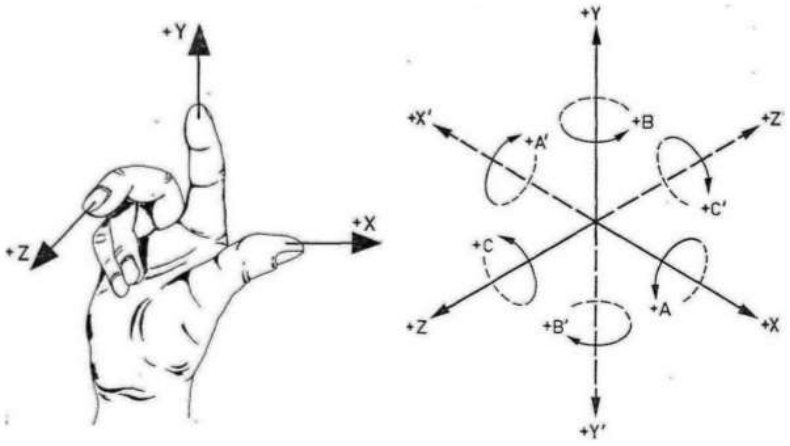
Para uma melhor compreensão da necessidade de controlo destes mecanismos, é importante perceber o que de facto se está a controlar. As máquinas CNC incluem um sistema de eixos mecânicos que se deslocam linearmente ou promovem a rotação dos componentes de forma a posicionar a peça ou a ferramenta na posição pretendida para remoção de material.

Os eixos lineares principais (X, Y e Z) das máquinas CNC identificam as direções dos deslocamentos principais das partes móveis da máquina. São, por exemplo, a mesa porta peças e o cabeçote móvel. (identificar figuras) (Tavares & Relvas, 2012)

As ferramentas da máquina são programadas pelo sistema cartesiano de coordenadas (observado na fig. 6). O eixo Z é sempre alinhado na direção da *Spindle*, ou seja, paralelo à árvore. O eixo X é normalmente paralelo à maior dimensão da mesa da máquina e, por fim, o eixo Y é paralelo à menor dimensão da mesa da máquina ferramenta.

Os parâmetros A, B e C são os eixos complementares da máquina. Designam os movimentos angulares em torno dos eixos X, Y e Z, respetivamente. (YAMADA, 1982)

Estes eixos são normalmente reconhecidos em mesas giratórias de posicionamento de peças ou cabeçotes orientáveis.



Apêndice - Figura 6 - Orientação dos eixos principais e secundários(ACIERA Iniciação Ao Comando Numérico Das Máquinas Ferramenta, n.d.)

Dependendo da complexidade da máquina, esta pode estar dotada de mais três eixos, U, V e W (eixos complementares de deslocamento) que são, respetivamente, paralelos a X, Y, Z.(Tavares & Relvas, 2012).

Processos de Maquinagem CNC

Evoluindo do processo de maquinagem de controlo numérico (CN) que, utilizava fita perfurada, a maquinagem CNC é um processo de manufatura que utiliza controlos computadorizados para operar e manipular máquinas e ferramentas de corte para moldar matéria. Embora o processo de maquinagem CNC ofereça vários recursos e operações, os princípios fundamentais do processo permanecem praticamente os mesmos em todos eles. O processo básico de maquinagem CNC inclui as seguintes etapas (Thomas, 2021):

- Projeção de um modelo CAD;
- Conversão do modelo CAD para CNC;
- Preparação da máquina CNC;
- Execução da maquinagem.

Design de modelo CAD

O processo de maquinagem CNC começa com a criação de um desenho CAD vetorial 2D ou 3D de peças sólidas internamente ou por uma empresa de serviços de desenho CAD / CAM. O *software* de desenho auxiliado por computador (CAD) permite que designers e fabricantes produzam um modelo ou renderização de suas peças e produtos junto com as especificações técnicas necessárias, como dimensões e geometrias, para a produção da peça ou produto (Thomas, 2021).

Os projetos de peças maquinadas através da CNC são restringidos pelas capacidades da máquina CNC e das ferramentas. Por exemplo, a maioria das máquinas-ferramentas CNC são cilíndricas, portanto, as geometrias de peças possíveis por meio do processo de maquinagem CNC são limitadas, pois as ferramentas criam seções de canto curvas. Além disso, as propriedades do material a ser removido, o projeto da ferramenta e as capacidades de fixação da máquina restringem ainda mais as possibilidades do projeto, como espessuras mínimas da peça, tamanho máximo da peça e inclusão e complexidade de cavidades e recursos internos (Thomas, 2021).

Uma vez que o projeto CAD é concluído, o operador exporta-o para um formato de arquivo compatível com a CNC, como STEP ou IGES.

Conversão de CAD

O projeto CAD formatado é executado por meio de um programa, geralmente um *software* de manufatura auxiliada por computador (CAM), para extrair a geometria da peça e gerar o código de programação digital que controlará a máquina CNC e manipulará o ferramental para produzir a peça projetada sob medida (Thomas, 2021).

As máquinas CNC usam várias linguagens de programação, incluindo código G e código M. A mais conhecida das linguagens de programação CNC, código geral ou geométrico, conhecido como código G, controla quando, onde e como as ferramentas da máquina se movem - por exemplo, quando ligar ou desligar, a velocidade de deslocamento para uma localização específica e quais os caminhos a seguir através da peça de trabalho. O código de função diverso, conhecido como código M, controla as funções auxiliares da máquina, como automatizar a remoção e substituição da tampa da máquina no início e no final da produção, respetivamente (Thomas, 2021).

Depois do programa CNC ser gerado, o operador faz *upload* deste para a máquina CNC.

Configuração da Máquina

Antes do operador executar o programa CNC, ele deve preparar a máquina CNC para operação. Essas preparações incluem fixar a peça de trabalho diretamente na máquina, em fusos de máquinas ou em tornos de máquina ou dispositivos de fixação semelhantes, e anexar as ferramentas necessárias, como brocas e fresas de topo, aos componentes adequados da máquina (Thomas, 2021).

Assim que a máquina estiver totalmente configurada, o operador pode executar o programa CNC (Thomas, 2021).

Execução das Operações de Maquinagem

O Programa CNC age como instruções para a máquina, submetendo comandos ditando as ações das ferramentas e movimentos no computador de controlo da CNC que, por sua vez, manipula as ferramentas (Thomas, 2021).

Ao iniciar o programa, a CNC é comandada para começar o processo de maquinagem. No decorrer das operações, o controlador guia a máquina de forma a executar as operações de maquinagem necessárias para a concessão do produto projetado (Thomas, 2021).

Constituintes da Máquina CNC

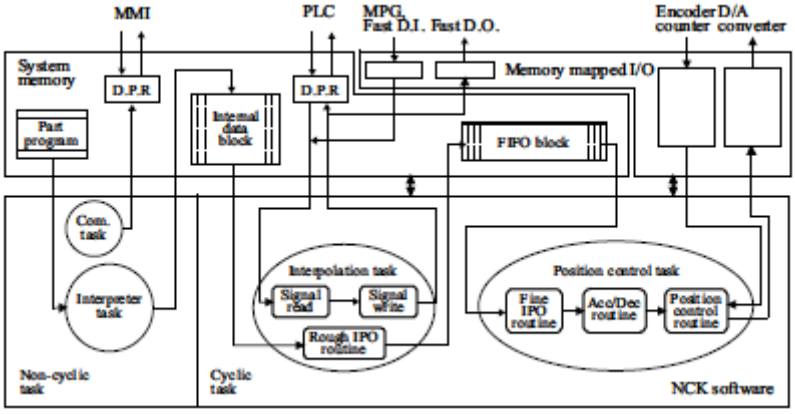
O sistema CNC é composto por três unidades principais: a unidade de comando numérico (CN) que promove a interface homem-máquina responsável pelo controlo de posição, a unidade dos motores e a unidade dos drivers. A CNC é geralmente tratada apenas como a unidade de Comando Numérico que, pode ser dividida em três grandes grupos de componentes, sendo estes o MMI, o PLC e o NCK (Suh et al., 2008).

A unidade MMI (*"Man Machine Interface"*, traduzido por Interface Homem-Máquina) oferece

a interface entre o CN e o utilizador, executa o comando de operação da máquina, exibe o estado da máquina e oferece funções para edição do programa das peças e comunicação.

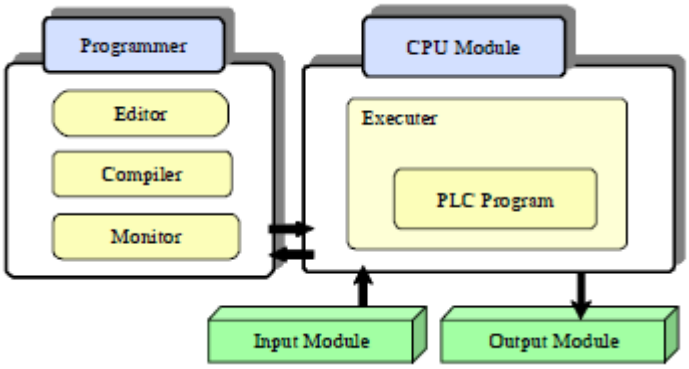
A unidade NCK, esquematizada na fig. 7, ("*Numerical "Control" Kernel*", traduzido para Controlo Numérico Kernel), é o núcleo do sistema da CNC. Este interpreta o programa e executa interpolação, controlo de posição e

compensação de erros com base no programa de peças interpretado. Por fim, controla também o sistema motor e promove a maquinagem da peça bruta (Suh et al., 2008).



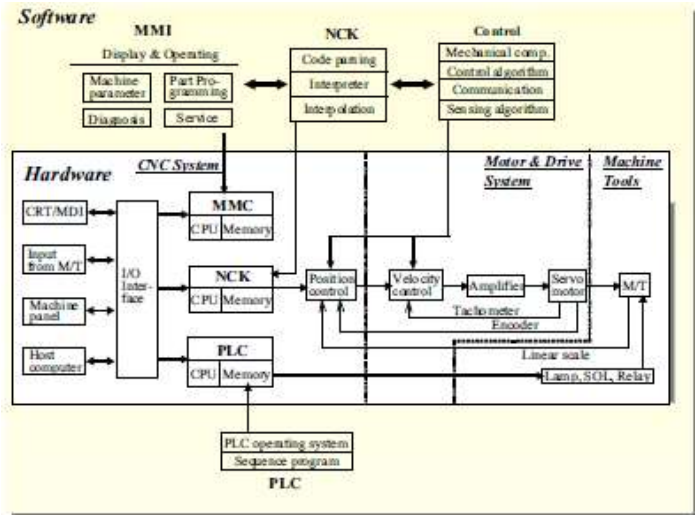
Apêndice - Figura 7 - Blocos funcionais NCK (Suh et al., 2008) .

O PLC ("*Programmable Logic "Control"*", traduzido para Controlo Lógico Programável), representado na fig. 8, controla sequencialmente a mudança da ferramenta, velocidade do fuso, mudança da peça de trabalho, e processamento de sinal de entrada/saída. Desempenha o papel de controlador do comportamento da máquina com execução do controlo motor (Suh et al., 2008).



Apêndice - Figura 8 - Arquitetura e funções do sistema PLC (Suh et al., 2008).

Na fig. 9, está representado, esquematicamente, o sistema de uma máquina CNC e as suas devidas interações. Cada um dos grupos de componentes contempla uma parte de *software* e outra de *hardware* (Suh et al., 2008).



Apêndice - Figura 9 - Sistema e interações da máquina CNC (Suh et al., 2008).

Do ponto de vista do *hardware*, as máquinas-ferramentas CNC consistem em CNC, sistema de acionamento de motor e ferramentas. A saída do controlo de posição é enviada para o sistema de acionamento do motor, o sistema de acionamento do motor opera um motor pelo controlo de velocidade e torque e, finalmente, faz com que a parte móvel se mova por meio do dispositivo de transmissão de energia. No sistema CNC, os módulos do processador que processam as funções da unidade MMI, unidade NCK, e a unidade PLC consiste no processador principal, uma ROM do sistema e uma RAM que armazena as aplicações do utilizador, partes do programa e os programas do PLC, respetivamente (Suh et al., 2008). O módulo de processamento está conectado a uma interface equipada com um meio de interação com o operador, tanto físico como visual, entradas externas e barramento de sistema. Portanto, a arquitetura de um sistema CNC é semelhante ao de um computador comum. O sistema CNC também possui um sistema analógico/digital, dispositivo de entrada/saída para comunicação direta com máquinas externas e uma comunicação por interface para ligar um dispositivo de acionamento de motor externo com uma entrada/saída do módulo. Posto isto, é possível implementar parâmetros do sistema de acionamento no sistema de comando numérico, monitorizar o estado do sistema global e ajustar especificações como redução de ruido e vibrações (Suh et al., 2008).

Ao expandir o conceito de comunicação digital, o mecanismo de comunicação destas máquinas foi aplicado a dispositivos de entrada e saída. Ou seja, a conexão entre um sistema CNC e uma variedade de sensores e dispositivos mecânicos é possível através de apenas uma linha de comunicação. Para este mecanismo de comunicação, existem normas e protocolos de rede como Profi-Bus, CAN Bus e

InterBus-S (Suh et al., 2008).

Do ponto de vista do *software*, o sistema CNC pode ser mostrado como na figura 24287(acima).

O sistema CNC consiste em funções MMI que suportam a operação e o programa de utilizador, edição e exibição do estado da máquina, funções NCK que executam interpretação, interpolação e controlo e, por fim, funções do PLC que executam programas lógicos sequenciais (Suh et al., 2008).

Apêndice IV. Medições dos Moldes de teste para verificação dos Steps/mm.

- **Medições das peças teste maquinadas** (tab. 14)

As dimensões virtuais da perna encontram-se na página seguinte descritas como dimensões 1 e 2.

Apêndice - Tabela 14 - Medições das peças maquinadas para testar os parâmetros.

Peça		Medida [cm]		
Joelho	Base Diâmetro Menor	12,39	12,36	12,39
	Base Diâmetro Maior	13,90	13,91	13,91
	Altura	11,54	11,50	11,53
Pé	Altura	15,4	15,5	15,5
	Comprimento	7,90	7,94	7,90
	Largura Maior	9,88	9,90	9,93
Perna	Altura	39,0	38,8	38,9
	Base Diâmetro Menor	12,53	12,48	12,52
	Base Diâmetro Maior	13,50	13,52	13,50

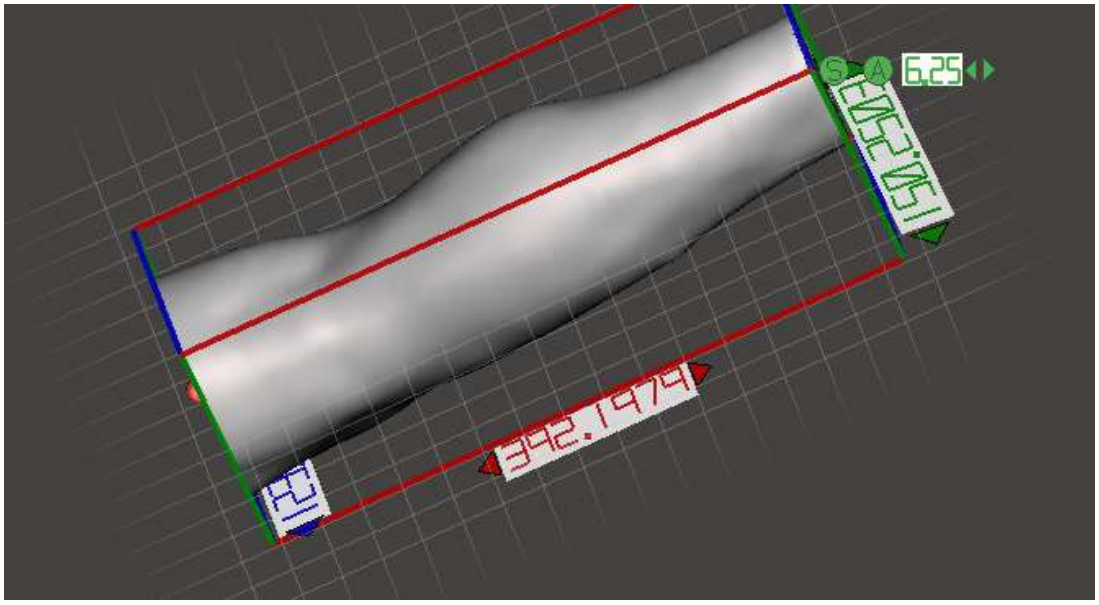
As dimensões sombreadas a azul foram retiradas com uma fita métrica de 2 m com escala mínima igual a 1 mm, ou seja, o erro absoluto será de ±0,5 mm.

As medidas sombreadas a laranja foram retiradas com um paquímetro erro absoluto de ± 0,02 mm.

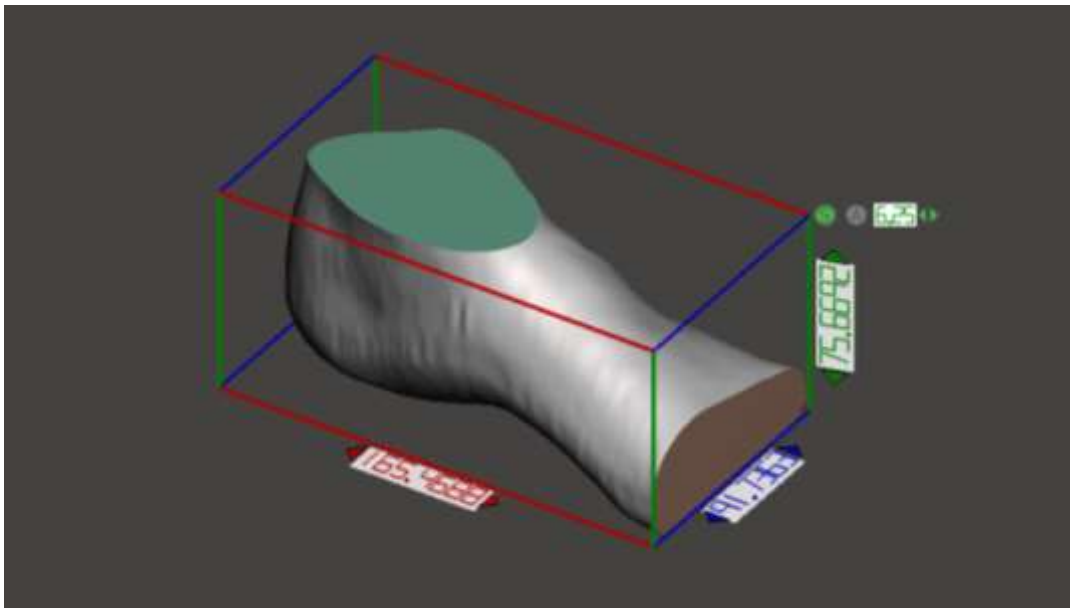
Sendo assim, o valor estimado é representado na tab.15.

Apêndice - Tabela 15 - Média de valores e erro absoluto.

Peça		Medida [mm]
Joelho	Base Diâmetro Menor	123,8 ± 0,02
	Base Diâmetro Maior	139.0 ± 0,02
	Altura	115,2 ± 0,02
Pé	Altura	154 ±0,5
	Comprimento	79,1 ±0,02
	Largura Maior	99,0 ±0,02
Perna	Altura	389 ±0,5
	Base Diâmetro Menor	125,1 ±0,02
	Base Diâmetro Maior	135,1 ±0,02



Apêndice - Figura 10 - Dimensões da perna: diâmetro maior - 150.25 mm, diâmetro menor - 123 mm, altura - 392,20 mm.

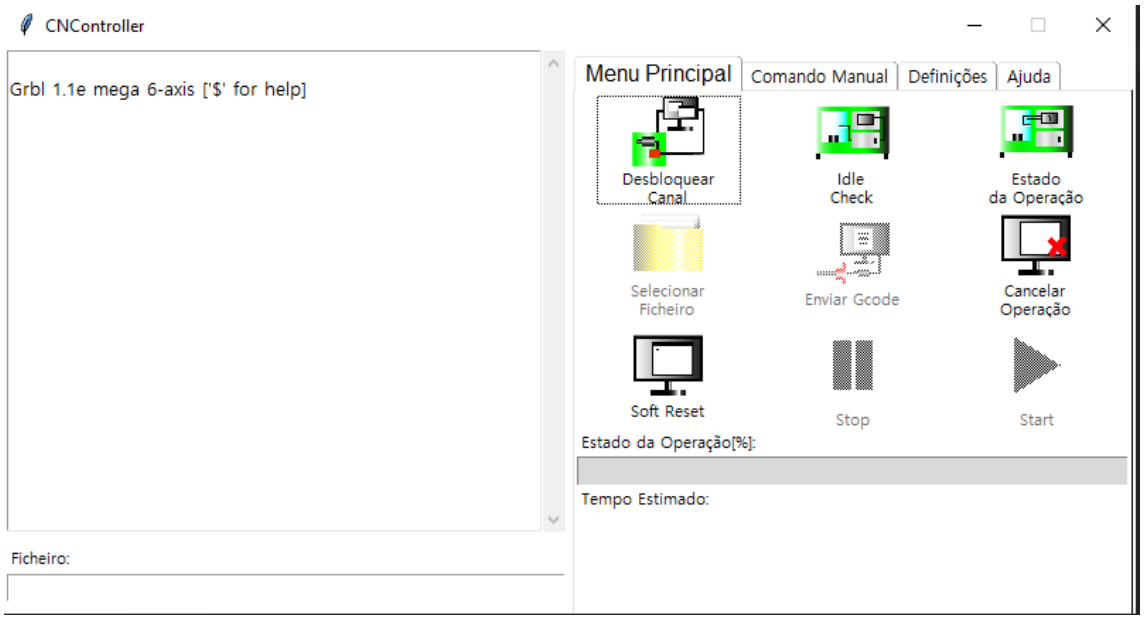


Apêndice - Figura 11 - Dimensões do pé: comprimento - 165.47 mm, largura - 91.74 mm, altura - 75.67 mm

Apêndice V. Composição gráfica da interface

A aplicação usufrui de quatro menus, deveras intuitivos, com diferentes funções em cada um. É possível ser utilizada em qualquer sistema operativo, o que permite o manuseamento num ecrã tátil, sem a necessidade de utilização de um rato ou teclado para funções exclusivamente ligadas à máquina de comando numérico em questão. Apesar de ser formatada para a CNC da Padrão Ortopédico, é possível ser editada para qualquer máquina que suporte o GRBL *firmware.*

Na fig. 12, encontra-se representado o menu principal da aplicação.



Apêndice - Figura 12 - Menu Principal da Interface desenvolvida.

Este menu é extremamente simples, porém, goza de todas as ações necessárias para por uma peça a ser maquinada.

Começando pelo lado esquerdo do programa, é possível ver o *widget* de comunicação com a máquina. Esta janela de texto é apresentada em qualquer dos quatro submenus, com o intuito do utilizador saber sempre qual a informação que está a ser enviada, distinguida pelo início de segmento frásico "user:", e qual a informação que está a ser recebida pela CNC.

Ainda do lado esquerdo da aplicação, existe uma caixa de texto que será preenchida após o operador escolher o ficheiro de código G a ser enviado.

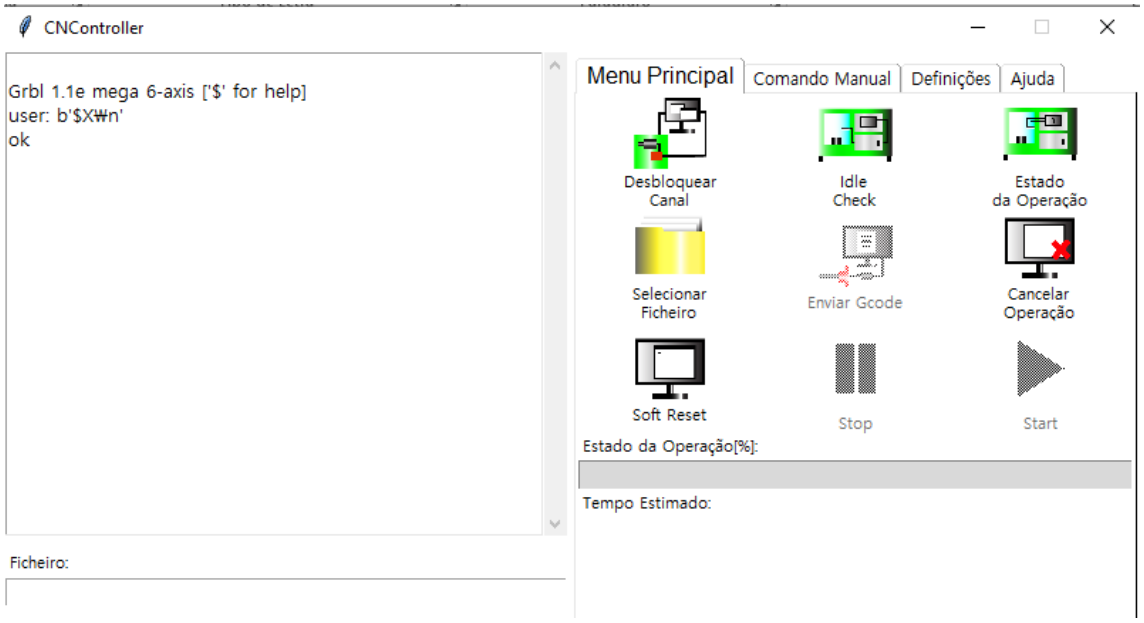
Posteriormente, no lado direito da interface são identificados os quatro submenus, nomeadamente, o menu principal, comando manual, definições e ajuda. O utilizador pode saber em qual dos menus se encontra a partir do tamanho (maior) de letra dessa mesma secção.

Neste caso, como indica na fig. 12, está em grande plano o "Menu Principal" que, como havia sido dito, permite ao operador da máquina a maquinagem de uma peça.

Dividindo os ícones de imagem em uma matriz 3x3 em que, cada posição remete a um botão diferente tem-se que:

1x1 – Desbloquear o Canal: este botão é crucial para a utilização da máquina. Quando o *software* inicia a ligação em série da interface-máquina é ativada, porém, o caminho de comunicação está interrompido por segurança, ou seja, não será lida qualquer que seja a mensagem enviada para o *firmware* se esta remeter para a movimentação dos motores do sistema.

Para evitar qualquer engano do operador, como pode ser visualizado na fig. 13, a interface foi programada para impossibilitar o utilizador de escolher o ficheiro e enviar sem antes clicar no botão de desbloquear o canal. Quando este é clicado, é enviado para O GRBL o comando "$X".



Apêndice - Figura 13 - Execução do botão responsável pelo comando de desbloquear o canal.

Após o envio do comando "$X", responsável por desbloquear a receção de comandos ao GRBL, este responde com "ok", como está descrito na figura 17. Caso ocorra algum erro inesperado, o próprio *firmware* indica um erro que aparecerá na janela de comunicação que, mais tarde, poderá ser consultado no menu "ajuda".

Também o botão de "Selecionar Ficheiro" fica ativo, podendo desta forma, ser utilizado.

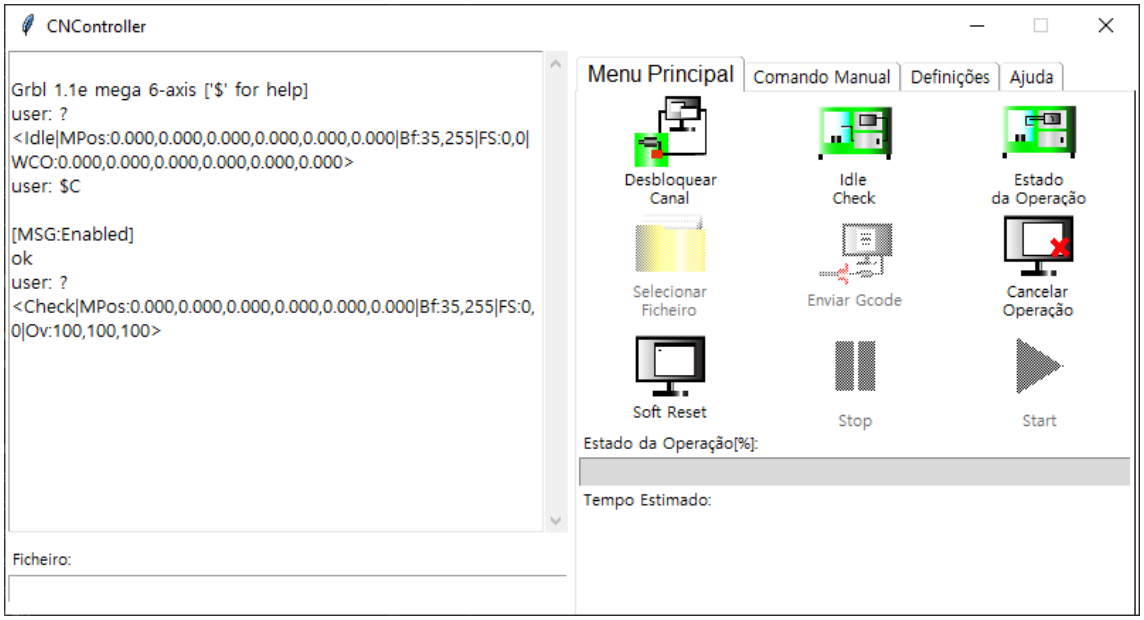
Este comando é de extrema importância pois, qualquer informação guardada no *buffer* do controlador Arduino não será transmitida até que o utilizador esteja preparado para a efetiva maquinagem da peça.

1x2 – "IDLE/Check": este botão permite selecionar o estado da máquina, por outras palavras, consiste em escolher se a maquina está a ouvir a informação enviada ou mão. O estado "IDLE" é praticamente idêntico à função "Desbloquear" anteriormente falada, porém, em casos que este estado seja alterado

com a máquina em funcionamento apenas este botão funcionará se o utilizador pretender a sua alteração. Por exemplo, aquando o início do processo de maquinar uma peça ou alterar parâmetros de maquinagem, o estado da máquina terá sempre de ser "IDLE" que por sua vez passará a "*Run*" durante a maquinagem. Em situações como "*reset*" da máquina, esta muda o seu estado para "*Check*" que, mais uma vez, pode apenas ser alterado pelo botão referido (fig. 14). O estado "*Check*" é um meio de proteção do *firmware* em si, que visa com que não sejam mudados parâmetros aquando a maquinagem de algo, ou também, que não se utilizem os comandos manuais simultaneamente com operações automáticas em curso.

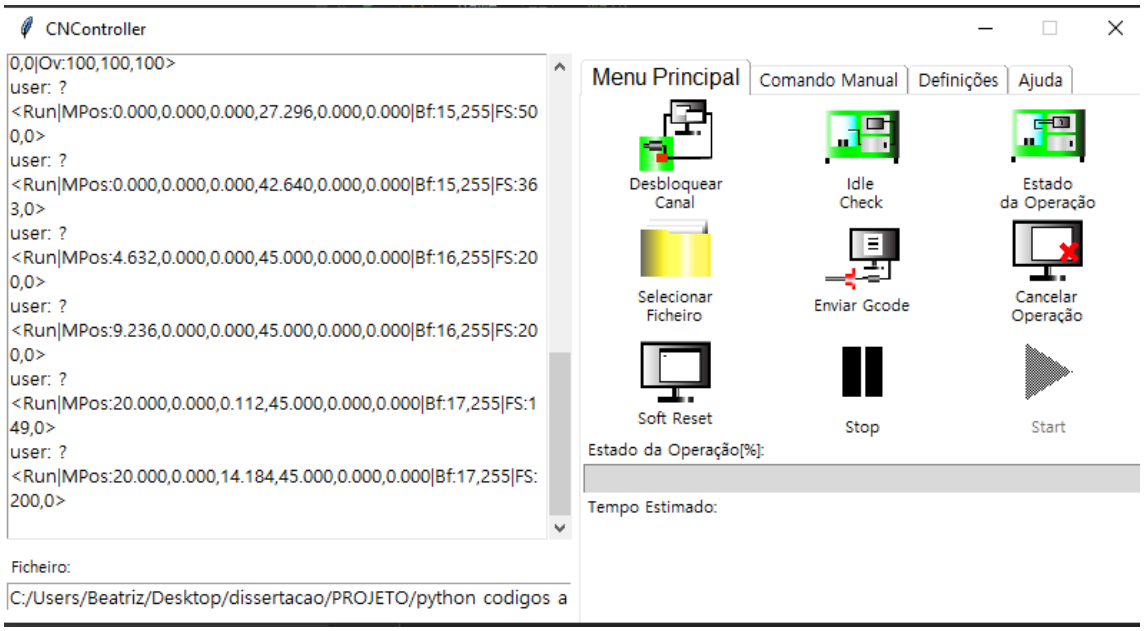
Existe ainda um outro estado, que sugere apenas o início da ligação em série designado por "*Alarm*"

Este botão consiste apenas no envio do código "$C" para o GRBL, representado na figura z ao qual, mas uma vez, na ausência de erro, o GRBL responde com "ok".



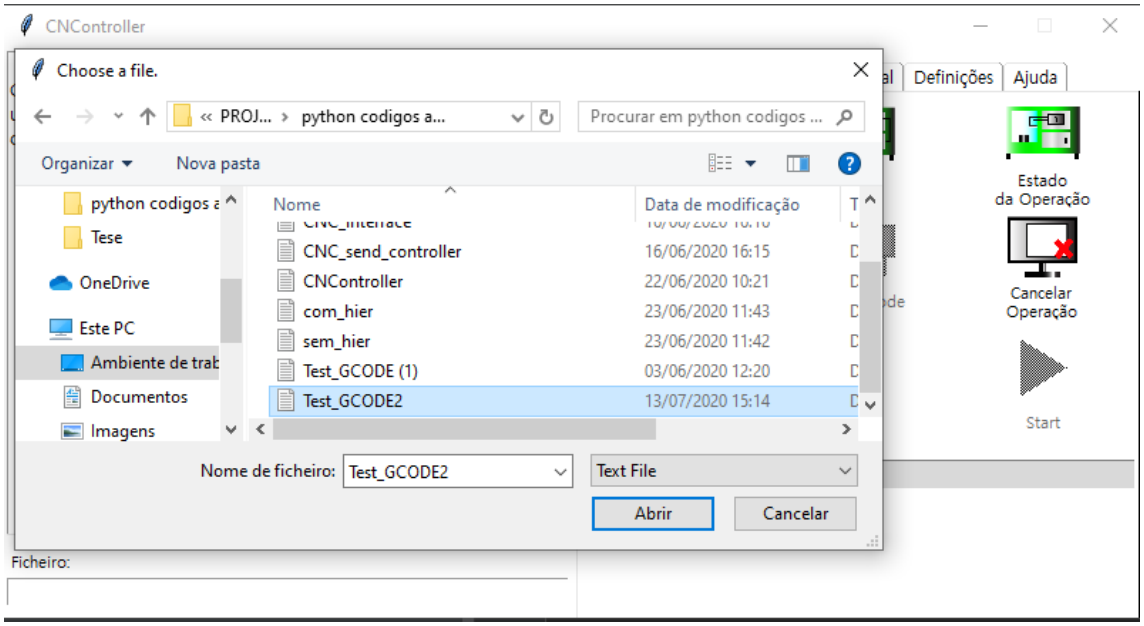
Apêndice - Figura 14 - Comunicação utilizador-máquina: Botão IDLE/Check.

1x3 – Estado da Operação: este botão é relevante para a orientar o utilizador do que se está a passar a qualquer momento. Sem restrições de utilização, podendo assim, ser clicado a qualquer instante de tempo, o "Estado de Operação" envia o código "?" para o *firmware* que, responde com as informações predefinidas que podem ser: o estado da máquina, o espaço do buffer, as coordenadas no plano XOY ou a velocidade da operação. Estes parâmetros podem ser alterados, podendo mostrar todas as informações ou parte delas ao operador. A informação aparecerá como ilustrada na fig. 15, nomeadamente, no que toda a estado, posição do eixo X e Y, espaço do Buffer e velocidade.



Apêndice - Figura 15 - Comunicação utilizador-máquina: Botão Estado da Operação.

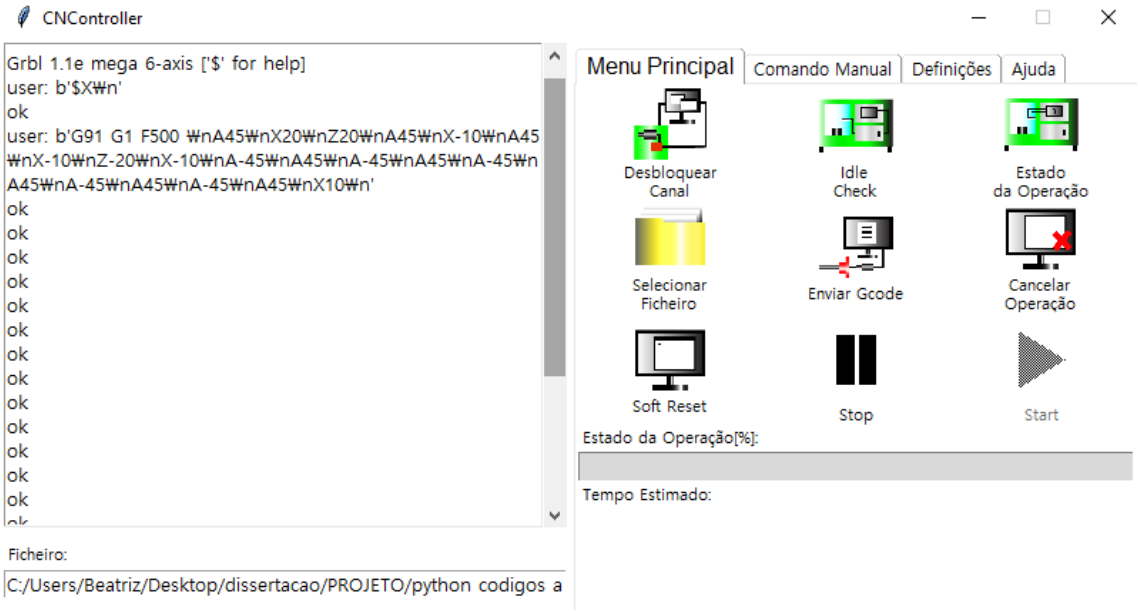
2x1 – Selecionar Ficheiro: Este ícone permite selecionar o ficheiro que detém o código G para o processo de maquinagem. Pode-se selecionar o ficheiro de qualquer parte do dispositivo, independentemente do sistema operativo do mesmo, tendo sido programado para tal, como é visível na fig. 16.



Apêndice - Figura 16 - Comunicação utilizador-máquina: Botão Selecionar Ficheiro.

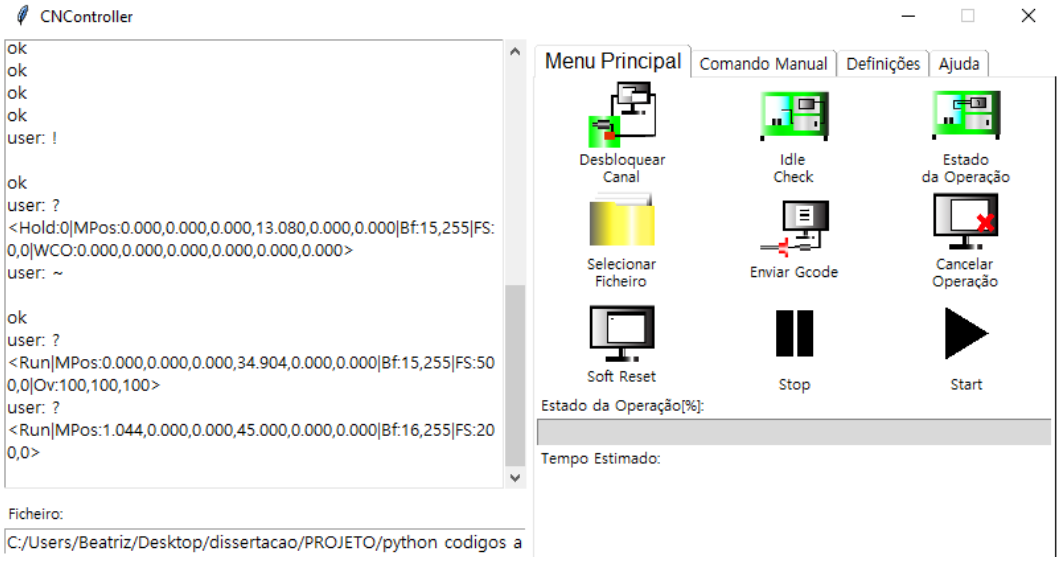
Uma janela de pesquisa fará *pop-up* a partir da qual será possível a seleção do ficheiro do código G. Após a seleção, o diretório do ficheiro será visualizado na caixa de texto no canto inferior esquerdo do programa e, o botão "Enviar Gcode" aparecerá desbloqueado.

2x2 – “Enviar Gcode”: envia o código para o controlador Arduino para o *firmware* ler e interpretar de forma a maquinar a peça, o GRBL enviará “ok” por cada linha de código lida pelo tradutor, visível na fig. 17, e, no caso da ocorrência de erro, por defeito, irá também enviar o numero do erro para ser verificado.



Apêndice - Figura 17 - Comunicação utilizador-máquina: Botão Enviar Gcode.

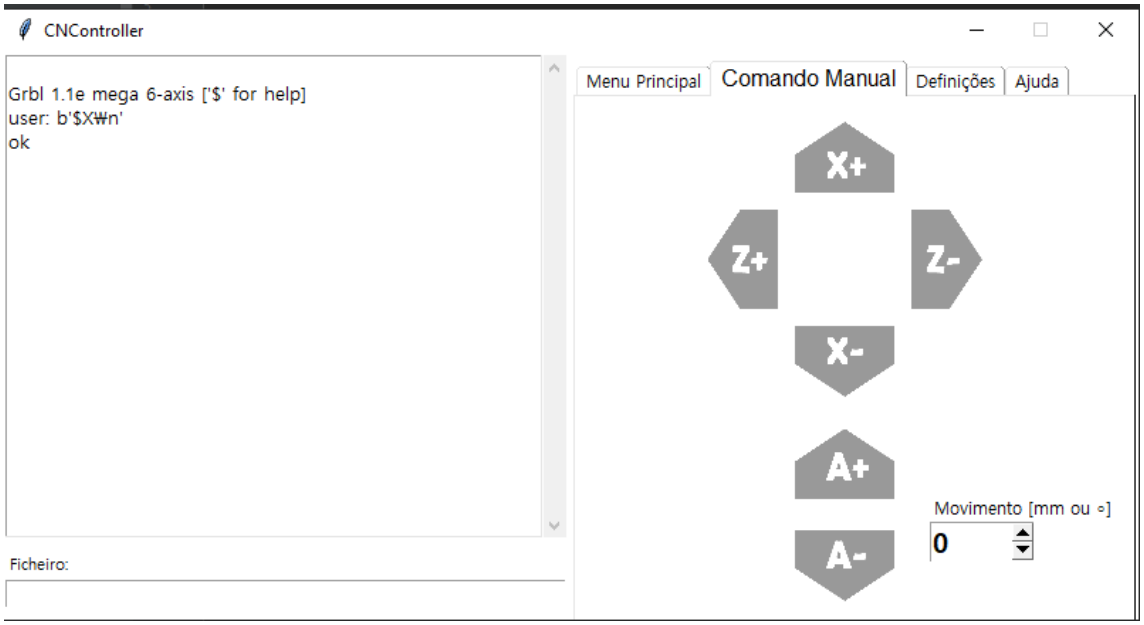
3x2 e 3x3 – “Stop” e “Start”: Estes botões são única e exclusivamente utilizados aquando um processo de maquinagem. Permitem o utilizador parar e recomeçar o processo quando estes necessitarem de o fazer. O botão “Stop” envia a mensagem “!” para o *firmware* que faz com que o estado de operação mude para “*Hold*”, como mostra a fig. 18, que inibe o movimento dos motores, ou seja, provoca uma quebra de corrente. Ao clicar no botão “Start”, a maquinagem pode ser continuada pelo envio do caracter “~” que exerce uma ligação da corrente novamente.



Apêndice - Figura 18 - Comunicação utilizador-máquina: Botões Start/Stop.

Ainda no menu principal, mas, depois dos botões descritos, é possível visualizar uma barra remetente ao processo da operação e um local para visualizar o tempo estimado.

No que toca à barra do "Estado de operação" esta identifica, percentualmente, a quantidade de código G lida e interpretada que se encontra no buffer do controlador Arduino. Não sendo esta responsável pelo processo de maquinagem em si. O segundo submenu é utilizado para operar manualmente a máquina, na fig. 19 é possível ver seis botões em forma de seta que permitem o movimento dos eixos X, Z e A, positiva ou negativamente, consoante o valor pretendido de deslocamento.

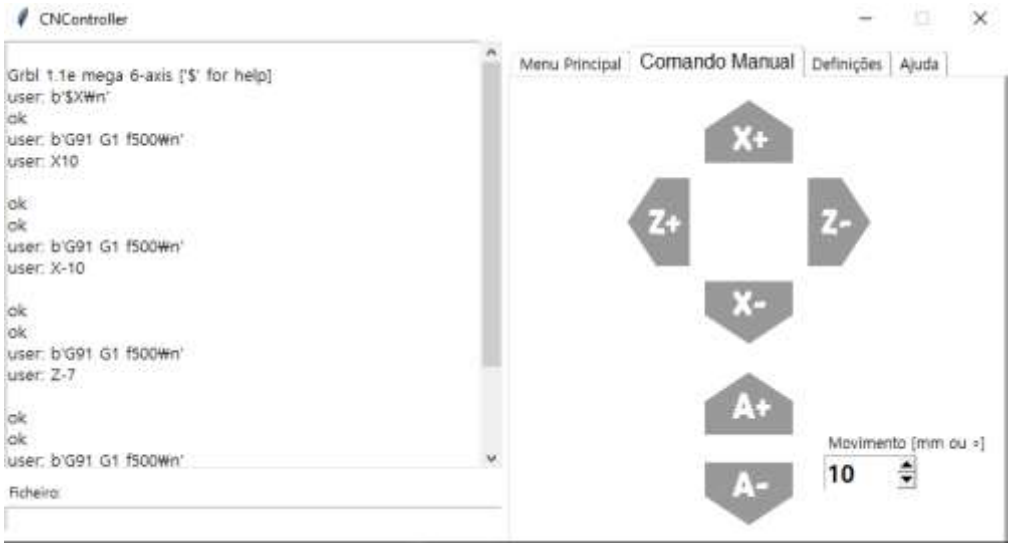


Apêndice - Figura 19 - Menu Comando Manual.

O valor do deslocamento pretendido pode ser definido no botão do canto inferior direito através das setas representadas. Este valor está inserido num vetor de números inteiros de zero a cem, sendo que a direção é decidida consoante o botão que é posteriormente clicado, dano o GRBL uma resposta como a da figura O. Numa abordagem mais concetual da interface, presumindo que a parte frontal da CNC onde esta será implementada é a face que engloba as portas de comunicação. Estes botões estão visualmente sincronizados com o movimento efetivo dos eixos da máquina, mais uma vez, para facilitar ao utilizador uma melhor perceção do movimento.

Salienta-se que, como forma de segurança. Não é possível utilizar estes comandos aquando um processo de maquinagem automático ou com a máquina no estado "*Check*" ou "*Alarm*". Evitando, assim, falhas no percurso da ferramenta, erros nas peças, movimento excessivo da ferramenta (para fora da área de trabalho), interceção de informação no buffer do controlador Arduino, visto que, mesmo enviando a informação, esta não é imediatamente recebida, ficando em "lista de espera" no *buffer*.

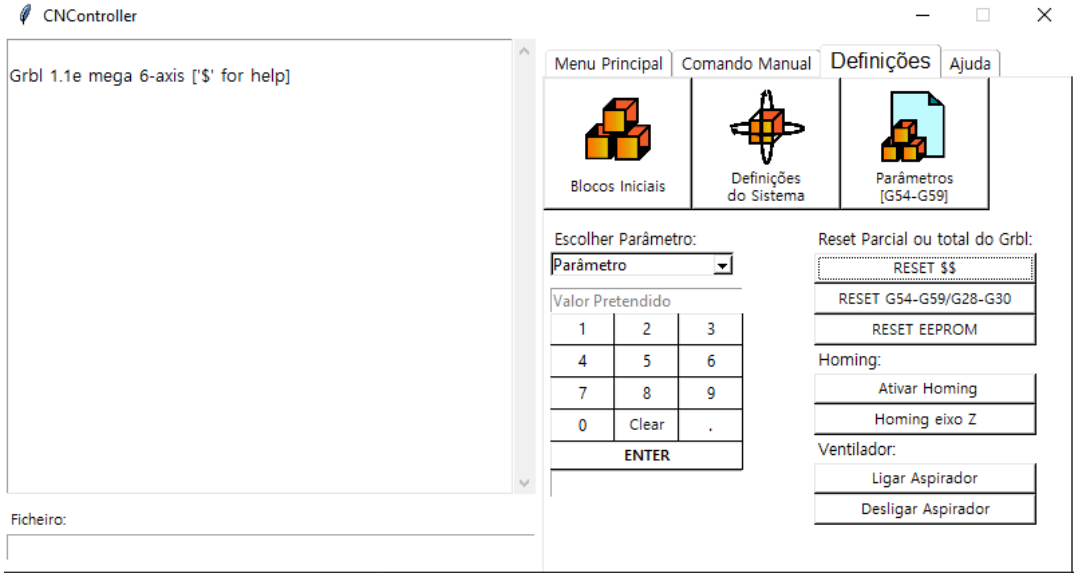
Tendo em atenção que ao clicar em qualquer comando o operador envia os comandos principais seguidos do eixo que pretende deslocar (“X”, “Z” ou “A”) e do número colocado na janela de seleção do valor de deslocamento (fig.20).



Apêndice - Figura 20 - Menu Comando Manual: interação e utilização.

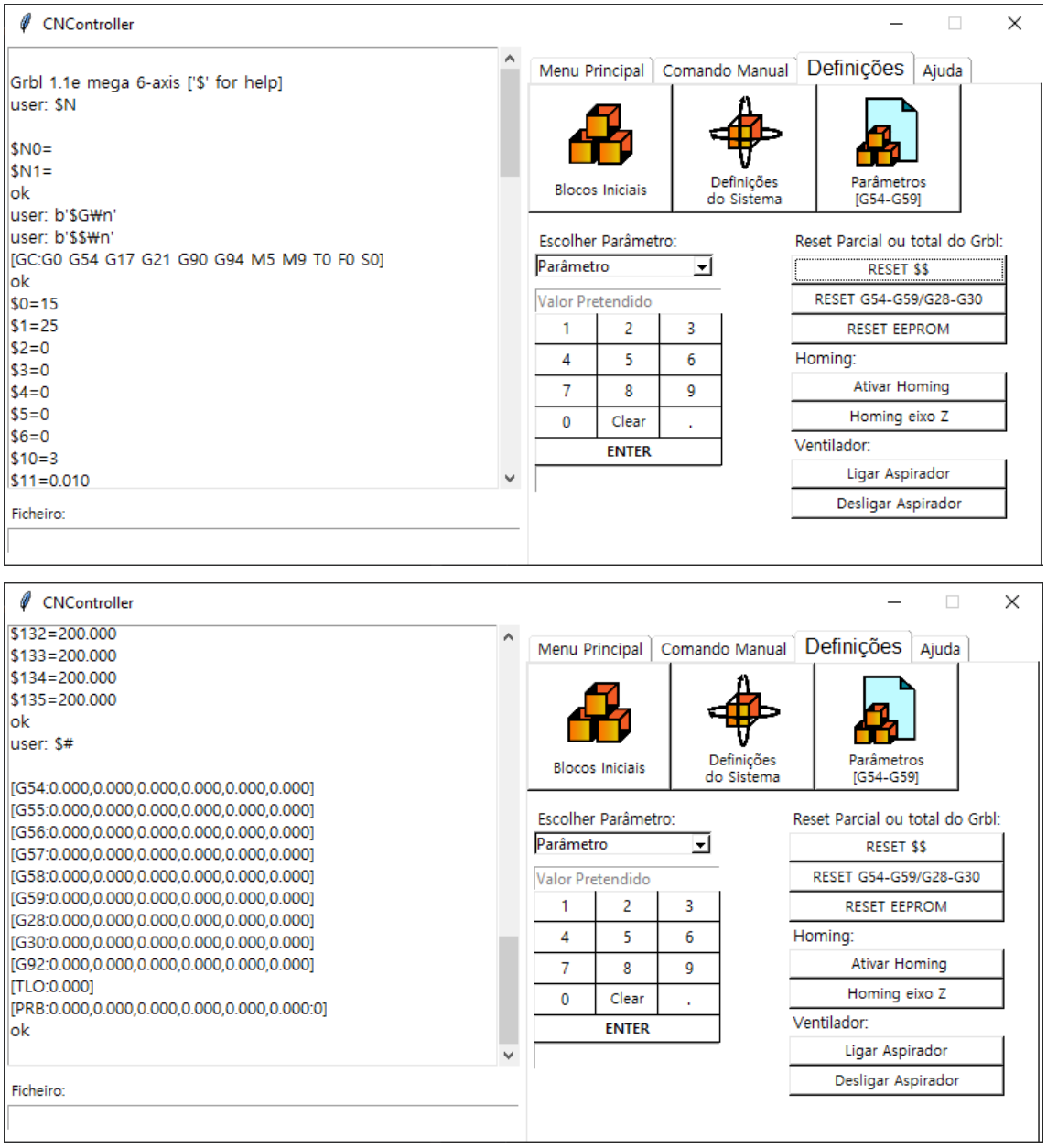
Passando agora ao terceiro submenu, intitulado de “Definições”. Esta secção é dedicada a todas as operações standard to GRBL, modificação de parâmetros de trabalho, visualização dos mesmos, processo de *homing* da máquina, *reset* do programa, entre outros.

Para facilitar a introdução da interface num ecrã tátil, sem a necessidade de teclado ou rato, cada função é definida por um botão, inibindo a necessidade do operador de escrever, o menu está exemplificado na figura 21.



Apêndice - Figura 21 - Menu Definições.

Os três botões iniciais são responsáveis pela visualização dos diferentes parâmetros já definidos do GRBL *firmware.* Como se pode ver na fig. 22, o utlizador ao clicar nos botões "Blocos Iniciais", "Definições do Sistema" ou Parâmetros [G54-G59], envia para o controlador Arduino os comandos "$N", "$G/$$" ou "$#", respetivamente.



Apêndice - Figura 22 - Visualização dos Parâmetros fornecidos pelo Grbl.

O GRBL responde, na ausência de erros, com a lista dos parâmetros pretendida.

Estes parâmetros podem ser alterados a partir da seleção do parâmetro a alterar, da inscrição do valor pretendido para o parâmetro e do clicar na tecla "ENTER". A este processo o *firmware* responderá com "ok". Em caso de estar algum processo de maquinagem em curso, o programa indicará um erro, para evitar uma alteração de parâmetros indesejada.

Por fim, no lado direito do submenu, encontram-se diversas funções automaticamente preparadas para a máquina.

Na secção "Reset Parcial ou total do Grbl", existem três botões que retornarão parâmetros de raiz do controlador Arduino.

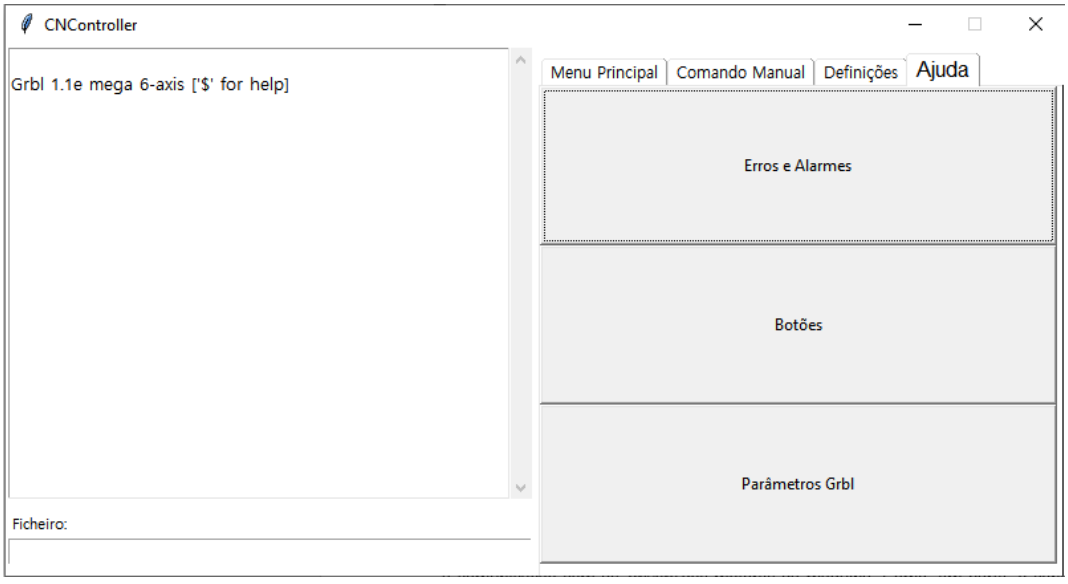
O "RESET $$" fará com que os valores das definições do sistema retomem os valores de fábrica. Já o "Reset G54-G59/G28-G30" faz com que os parâmetros indicados no próprio nome recuem para os iniciais. Por fim, o último botão, "RESET EEPROM", promove, a todos os parâmetros existentes, os valores de origem.

Por fim encontram-se ainda quatro botões, dois referentes ao *homing* da máquina e outros dois, referentes à ativação do ventilador.

Em primeira instância é necessário salientar, mais uma vez, que este programa foi exclusivamente formado para a máquina CNC relatada, pelo que, haveria só necessidade de fazer *homing* apenas do eixo "Z", isto porque, trabalhando a máquina sempre com peças de diferentes alturas, no zero peça, o valor de X, varia sempre consoante a peça a maquinar.

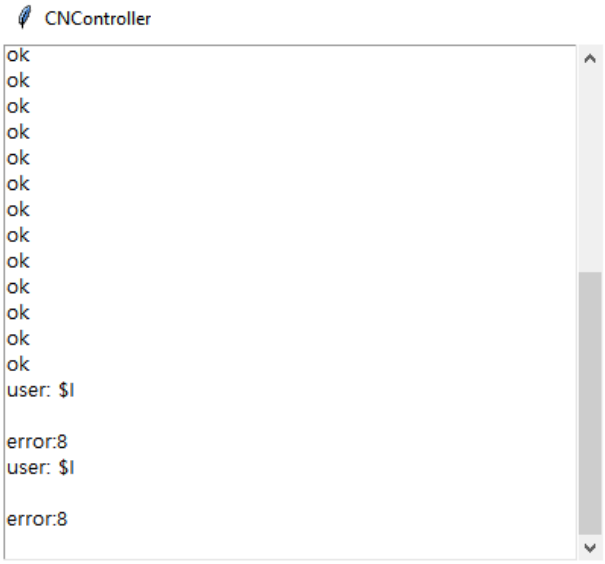
Os botões apresentados para o ventilador, foram implementados para uma posterior introdução de um sistema de aspiração para remover as limalhas de material da máquina.

O quarto e último menu apresentado, menu "Ajuda", está destinado apenas a colheita de informação teórica, pela parte do operador, sobre a interface e o Grbl *firmware.* Como é possível observar na fig. 23, este submenu comporta apenas três botões.



Apêndice - Figura 23 - Menu "Ajuda" do "CNController".

O primeiro botão, "Erros e Alarmes", lista os erros e alarmes que o Grbl pode enviar aquando a sua utilização, por exemplo, como demonstrado na fig. 24, "error 8".

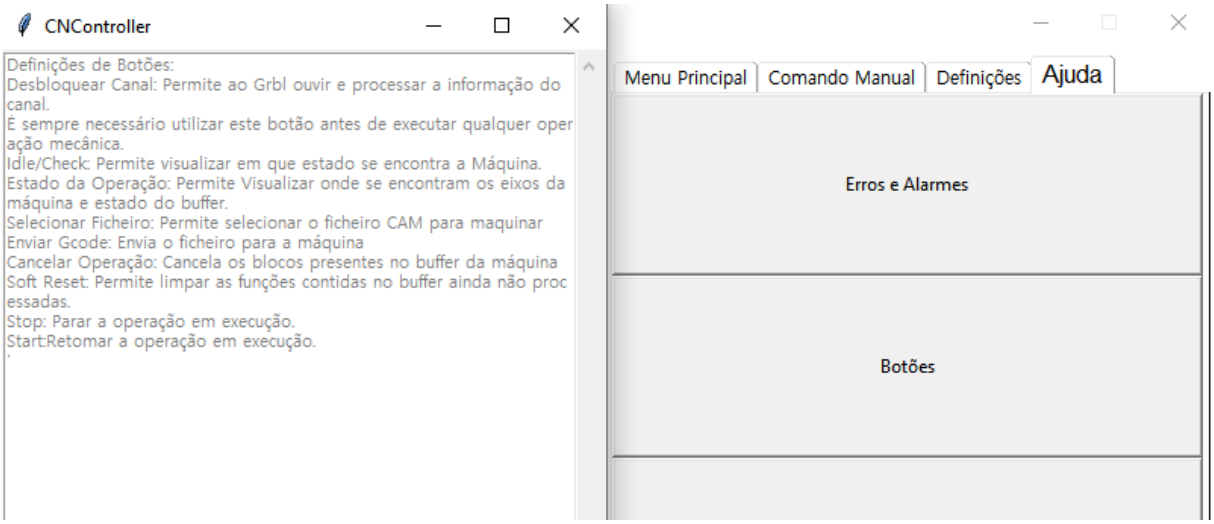


Apêndice - Figura 24 - Mensagem intercetada do Grbl: "Error:8".

A partir deste botão, o utilizador pode aceder a um texto onde encontra a definição do erro, neste caso, demonstrado na fig. 25, o "error: 8" remete para a utilização de linguagem por parte do operador não acessível enquanto a máquina está no modo "run".

Apêndice - Figura 25 - Definição de "error 8" no menu ajuda, botão "Erros e Alarmes".

O segundo botão, "Botões", esclarifica o operador do que cada botão da interface promove (com uma breve definição), como expresso na fig. 26.



Apêndice - Figura 26 - Janela pop-up para ajuda das definições dos botões.

Por fim, o botão "Parâmetros Grbl", permite ao operador visualizar todos os parâmetros possíveis de alterar no *firmware*.

Apêndice VI. Código para programação da interface

Script "model.py"

```
from tkinter import filedialog
import time
import serial
import re
import sys
import argparse
import threading



class Model:
    def __init__(self):
        self.ser = serial.Serial('COM8', 115200, writeTimeout=0, xonxoff=True, stopbits=1, bytesize=8)
        self.file_name = ''
        self.code = ''
        self.data_str = ''
        self.x = ''
        self.linhas = ''
        print(self.ser.get_settings())

        self.RX_BUFFER_SIZE = 128
        self.BAUD_RATE = 115200
        self.ENABLE_STATUS_REPORTS = True
        self.REPORT_INTERVAL = 1.0  # seconds
        self.talk='puta'
        self.is_run = True  # Controls query timer

        # Define command line argument interface
        self.parser = argparse.ArgumentParser(
            description='Stream g-code file to grbl. (pySerial and argparse libraries required)')
        # parser.add_argument('gcode_file', type=argparse.FileType('r'),
        #           help='g-code filename to be streamed')
        # parser.add_argument('device_file',
        #                   help='serial device path')
        self.parser.add_argument('-q', '--quiet', action='store_true', default=False,
                            help='suppress output text')
        self.parser.add_argument('-s', '--settings', action='store_true', default=False,
                            help='settings write mode')
        self.parser.add_argument('-c', '--check', action='store_true', default=False,
                            help='stream in check mode')
        self.args = self.parser.parse_args()
        self.verbose = True
```

```
    if self.args.quiet:
        self.verbose = False
    self.settings_mode = False
    if self.args.settings:
        self.settings_mode = True
    self.check_mode = False
    if self.args.check:
        self.check_mode = True

# Periodic timer to query for status reports
# TODO: Need to track down why this doesn't restart consistently before a release.
def send_status_query(self):
    self.ser.write('?'.encode())

def periodic_timer(self):
    while self.is_run:
        self.send_status_query()
        time.sleep(self.REPORT_INTERVAL)

def get_path(self):
    self.file_name = filedialog.askopenfilename(initialdir="/",
                                        filetypes=(("All Files", '.*'),), title="Choose a file.")
    return self.file_name

def read_from_serial(self):
    while True:
        self.ser.flushOutput()
        if self.ser.inWaiting() > 0:
            self.data_str = self.ser.read(self.ser.inWaiting()).decode('ascii')
            print(self.data_str)
        time.sleep(0.02)
        return self.data_str

def write_to_serial(self):
    self.x = self.ser.write(self.code)
    return self.x

def unlock(self):
    self.code = b'$X\n'
    print(self.code)
    return self.code

def parser_state(self):
```

```
    self.code = b'$G\n'
    print(self.code)
    return self.code

def param(self):
    self.code = b'$$\n'
    print(self.code)
    return self.code

def read_gcode(self):
    with open(self.file_name, 'r') as f:

        # Wait for grbl to initialize and flush startup text in serial input
        time.sleep(5)
        self.ser.flushInput()

        if self.check_mode:
            print("Enabling Grbl Check-Mode: SND: [$C]"),

            self.ser.write("$C".encode())
            while 1:
                self.grbl_out = self.ser.readline().strip().decode()  # Wait for grbl response with carriage return
                if self.grbl_out.find('error') >= 0:
                    print("REC:", self.grbl_out)

                    print("  Failed to set Grbl check-mode. Aborting...")
                    quit()
                elif self.grbl_out.find('ok') >= 0:
                    if self.verbose:
                        print('REC:', self.grbl_out)
                    break

        self.start_time = time.time()

        # Start status report periodic timer
        if self.ENABLE_STATUS_REPORTS:
            self.timerThread = threading.Thread(target=self.periodic_timer)
            self.timerThread.daemon = True
            self.timerThread.start()

        # Stream g-code to grbl
        self.l_count = 0
        self.error_count = 0
```

```
if self.settings_mode:
    print("SETTINGS MODE: Streaming", self.args.gcode_file.name, " to ", self.args.device_file)
    for line in f:
        self.l_count += 1  # Iterate line counter
        # l_block = re.sub('\s|\(.*?\)','',line).upper() # Strip comments/spaces/new line and capitalize
        l_block = line.strip()  # Strip all EOL characters for consistency
        if self.verbose: print("SND>" + str(self.l_count) + ": \"" + l_block + "\"")
        self.resulta = l_block + '\n'
        self.talk =self.ser.write(self.resulta.encode())  # Send g-code block to grbl
        while 1:
            self.rbl_out = self.ser.readline().strip().decode()  # Wait for grbl response with carriage return
            if self.grbl_out.find('ok') >= 0:
                if self.verbose:
                    print("  REC<" + str(self.l_count) + ": \"" + self.grbl_out + "\"")
                break
            elif self.grbl_out.find('error') >= 0:
                if self.verbose:
                    print("  REC<" + str(self.l_count) + ": \"" + self.grbl_out + "\"")
                self.error_count += 1
                break
            else:
                print("    MSG: \"" + self.grbl_out + "\"")
else:
    self.g_count = 0
    self.c_line = []
    for line in f:
        self.l_count += 1  # Iterate line counter
        l_block = re.sub('\s|\(.*?\)', '', line).upper()  # Strip comments/spaces/new line and capitalize
        # l_block = line.strip()
        self.c_line.append(len(l_block) + 1)  # Track number of characters in grbl serial read buffer
        self.grbl_out = ''
        while sum(self.c_line) >= self.RX_BUFFER_SIZE - 1 | self.ser.inWaiting():
            out_temp = self.ser.readline().strip().decode()  # Wait for grbl response
            if out_temp.find('ok') < 0 and out_temp.find('error') < 0:
                print("    MSG: \"" + out_temp + "\"")  # Debug response
            else:
                if out_temp.find('error') >= 0: self.error_count += 1
                self.g_count += 1  # Iterate g-code counter
                if self.verbose: print("  REC<" + str(self.g_count) + ": \"" + out_temp + "\"")
                del self.c_line[0]  # Delete the block character count corresponding to the last 'ok'
        resulta = l_block + '\n'
        self.talk = self.ser.write(resulta.encode())  # Send g-code block to grbl
        if self.verbose: print("SND>" + str(self.l_count) + ": \"" + l_block + "\"")
```

```
            # Wait until all responses have been received.
            while self.l_count > self.g_count:
                self.out_temp = self.ser.readline().strip().decode()  # Wait for grbl response
                if self.out_temp.find('ok') < 0 and self.out_temp.find('error') < 0:
                    print("    MSG: \"" + self.out_temp + "\"")  # Debug response
                else:
                    if self.out_temp.find('error') >= 0: self.error_count += 1
                    self.g_count += 1  # Iterate g-code counter
                    del self.c_line[0]  # Delete the block character count corresponding to the last 'ok'
                    if self.verbose: print("  REC<" + str(self.g_count) + ": \"" + out_temp + "\"")

        # Wait for user input after streaming is completed
        print("\nG-code streaming finished!")
        self.end_time = time.time()
        is_run = False
        print(" Time elapsed: ",self.end_time - self.start_time, "\n")
        if self.check_mode:
            if self.error_count > 0:
                print("CHECK FAILED:", self.error_count, "errors found! See output for details.\n")
            else:
                print("CHECK PASSED: No errors found in g-code program.\n")
        else:
            print("WARNING: Wait until Grbl completes buffered g-code blocks before exiting.")
            input("  Press <Enter> to exit and disable Grbl.")

        # Close file and serial port
        f.close()
        self.ser.close()
        time.sleep(2)
    return self.talk

def __str__(self):
    return self.talk

#def da_por_favor(self):
 #   self.thread = threading.Thread(target=self.read_gcode, daemon=True)
  #  self.thread.start()
   # if self.thread.is_alive():
    #    print('is alive')
    #return
```

Script "control.py"

```
from MVC_pack.model import Model

class Controller:

    def __init__(self):
        self.model = Model()

    def talking(self):
        return self.model.talk

    def serial_get(self):
        return self.model.ser

    def get_path(self):
        return self.model.get_path()

    def get_read(self):
        return self.model.read_from_serial()

    def write_serial(self):
        return self.model.write_to_serial()

    def get_unlock(self):
        return self.model.unlock()

    def get_parser_state(self):
        return self.model.parser_state()

    def get_param(self):
        return self.model.param()

    def get_gcode(self):
        # return self.model.gcode()
        return self.model.read_gcode()
```

Script "view.py"

```
from tkinter import *
from tkinter import ttk
import time
import tkinter.scrolledtext as tkst
from tkinter.ttk import Progressbar
from MVC_pack.controller import Controller
from text.error_list import lista_erros
from text.botoes_def import Bt_def
from text.parametros import param
import re
import argparse
import threading
from threading import Event

class View:
    def __init__(self):
        self.root = Tk()
        self.root.title("CNController")
        self.root.iconbitmap('C:/Users/Beatriz/Desktop/dissertacao/PROJETO/versoes_codigo/ultimo/Tese/MVC_pack/images/icon_entrada.ico')
        self.root.geometry("800x400")
        self.root.resizable(width=False, height=False)
        self.root.config(bg='white')
        self.control = Controller()
        self.serial = self.control.serial_get()
        self.reading = StringVar()
        self.grbl = ''
        # definir o que enviar ao grbl
        self.parametros_iniciais = ''
        self.array = list(range(136))
        self.buffer = ''
        self.data_str=''
        print(self.array)
        self.texto_ajuda = lista_erros
        self.Bt_def = Bt_def
        self.ajuda_param = param
        self.gcode = ""
        self.file_name = ""
```

```
##############################################################################################################
        self.RX_BUFFER_SIZE = 128
        self.BAUD_RATE = 115200
        self.ENABLE_STATUS_REPORTS = True
        self.REPORT_INTERVAL = 1.0  # seconds
        self.talk='puta'
        self.is_run = True  # Controls query timer

        # Define command line argument interface
        self.parser = argparse.ArgumentParser(
            description='Stream g-code file to grbl. (pySerial and argparse libraries required)')
        # parser.add_argument('gcode_file', type=argparse.FileType('r'),
        #           help='g-code filename to be streamed')
        # parser.add_argument('device_file',
        #                   help='serial device path')
        self.parser.add_argument('-q', '--quiet', action='store_true', default=False,
                            help='suppress output text')
        self.parser.add_argument('-s', '--settings', action='store_true', default=False,
                            help='settings write mode')
        self.parser.add_argument('-c', '--check', action='store_true', default=False,
                            help='stream in check mode')
        self.args = self.parser.parse_args()
        self.verbose = True
        if self.args.quiet:
            self.verbose = False
        self.settings_mode = False
        if self.args.settings:
            self.settings_mode = True
        self.check_mode = False
        if self.args.check:
            self.check_mode = True
        self.response = True
##############################################################################################################
##

        # ficheiros de imagem para os botoes tab1
        self.fstartup = PhotoImage(file="images/fstartup.png")\
            .subsample(8, 8)
        self.fparam = PhotoImage(file="images/fparam.png")\
            .subsample(8, 8)
        self.fparam_card = PhotoImage(file="images/fparamcard.png")\
            .subsample(8, 8)
```

```
self.funlock = PhotoImage(file="images/funlock.png")\
    .subsample(8, 8)
self.freset = PhotoImage(file="images/freset.png")\
    .subsample(8, 8)
# ajustar isto depois
self.fbuild = PhotoImage(file="images/fcancel.png")\
    .subsample(8, 8)
self.ffile = PhotoImage(file="images/ffile.png")\
    .subsample(8, 8)
self.fidle = PhotoImage(file="images/fidle.png")\
    .subsample(8, 8)
self.fstatus = PhotoImage(file="images/fstatus.png")\
    .subsample(8, 8)
self.fsend = PhotoImage(file="images/fsend.png")\
    .subsample(8, 8)
self.fhold = PhotoImage(file="images/fhold.png")\
    .subsample(8, 8)
self.fcycle = PhotoImage(file="images/fcycle.png")\
    .subsample(8, 8)

# ficheiros de imagem para os botoes manuais
self.photox_mais = PhotoImage(file="images/x_mais.png")\
    .subsample(6, 6)
self.photox_menos = PhotoImage(file="images/x_menos.png")\
    .subsample(6, 6)
self.photoz_mais = PhotoImage(file="images/z_mais.png")\
    .subsample(6, 6)
self.photoz_menos = PhotoImage(file="images/z_menos.png")\
    .subsample(6, 6)
self.photoa_mais = PhotoImage(file="images/a_mais.png")\
    .subsample(6, 6)
self.photoa_menos = PhotoImage(file="images/a_menos.png")\
    .subsample(6, 6)

# LABELS
s = ttk.Style()

s.theme_create("yummy", parent="alt", settings={
                "TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},
                "TNotebook.Tab": {
                    "configure": {"padding": [5, 1], "background": 'white', "font": ("Malgun Gothic", 9)},
                    "map":       {"background": [("selected", 'white')],
                                  "font": [("selected", 'bold')],
```

```
                                ''expand'': [(''selected'', [1, 1, 1, 0])]})})

        s.theme_use(''yummy'')
        s.configure('.', foreground='black', background='white')
        self.label = Label(self.root, bg='white')
        self.label.place(x=0, y=0, width=400, height=480)
        self.label2 = Label(self.root, bg='white')
        self.label2.place(x=400, y=0, width=400, height=480)
        self.tabControl = ttk.Notebook(self.label2)
        self.label1 = ttk.Frame(self.tabControl)  # Create a tab
        self.tabControl.add(self.label1, text='Menu Principal')
        self.label_men = Label(self.label1, width=400, height=480, bg='white')
        self.label_men.place(x=0, y=0)
        self.label3 = ttk.Frame(self.tabControl)
        self.tabControl.add(self.label3, text='Comando Manual')
        self.label5 = ttk.Frame(self.tabControl)
        self.label1_1 = ttk.Frame(self.tabControl)
        self.tabControl.add(self.label1_1, text='Definições')
        self.label1_2 = Label(self.label1_1, width=220, height=300, bg='white')
        self.label1_2.place(x=200, y=108)
        self.tabControl.add(self.label5, text='Ajuda')
        self.tabControl.pack(expand=1, fill=''both'')

        # interface de comunicação e diretório
        # self.nome1 = Label(self.label, text='Interface:', font=(''Malgun Gothic'', 10), bg='black')
        # self.nome1.grid(column=0, row=0)
        self.interface = tkst.ScrolledText(self.label, bg=''white'', fg=''black'', font=(''Malgun Gothic'', 10))
        self.interface.place(height=340, width=395, x=0, y=0)
        self.interface.yview(END)
        self.nome = Label(self.label, text='Ficheiro:', font=(''Malgun Gothic'', 8), bg='white', fg='black')
        self.nome.place(x=0, y=350)
        self.entry_file_path = Text(self.label, font=(''Malgun Gothic'', 10))
        self.entry_file_path.place(height=20, width=395, x=0, y=370)
# Label 1
        # BUTTONS

# linha 1

        self.unlock_grbl = Button(self.label_men, text='Desbloquear\nCanal', command=self.controller_unlock,
                                  image=self.funlock, compound=TOP, width=100, height=75, font=(''Malgun Gothic'', 8),
                                  bg='white', borderwidth=0)
        self.unlock_grbl.grid(column=0, row=0)
```

```
        self.gcode_mode = Button(self.label_men, text='Idle\nCheck', command=self.get_gcode_mode,
                                 image=self.fidle, borderwidth=0, compound=TOP, width=100, height=75,
                                 font=("Malgun Gothic", 8), bg='white')
        self.gcode_mode.grid(column=1, row=0)
        self.status = Button(self.label_men, text='Estado\nda Operação', command=self.get_status,
                             image=self.fstatus, borderwidth=0, compound=TOP, width=100, height=75,
                             font=("Malgun Gothic", 8), bg='white')
        self.status.grid(column=2, row=0)
    # linha 2
        self.search_file_button = Button(self.label_men, text='Selecionar\nFicheiro',
                                         command=self.controller_getfilename,
                                         image=self.ffile, borderwidth=0, compound=TOP, width=100, height=75,
                                         font=("Malgun Gothic", 8),
                                         state=DISABLED, bg='white')
        self.search_file_button.grid(column=0, row=1)
        self.send_code = Button(self.label_men, text='Enviar Gcode', command= lambda: (self.da_por_favor()), state=DISABLED,
                                image=self.fsend, borderwidth=0, compound=TOP, width=100, height=75,
                                font=("Malgun Gothic", 8), bg='white')
        self.send_code.grid(column=1, row=1)
        self.cancel = Button(self.label_men, text='Cancelar\nOperação', command=self.get_buil_info,
                             image=self.fbuild, borderwidth=0, compound=TOP, width=100, height=75,
                             font=("Malgun Gothic", 8), bg='white')
        self.cancel.grid(column=2, row=1)

    # linha 3
        self.soft_reset = Button(self.label_men, text='Soft Reset', command=self.soft_reset,
                                 image=self.freset, borderwidth=0, compound=TOP, width=100, height=75,
                                 font=("Malgun Gothic", 8), bg='white')
        self.soft_reset.grid(column=0, row=2)
        self.feed_hold = Button(self.label_men, text='Stop', command=self.get_feedhold,
                                image=self.fhold, borderwidth=0, compound=TOP, width=100, height=75,
                                font=("Malgun Gothic", 8), bg='white')
        #state=DISABLED
        self.feed_hold.grid(column=1, row=2)
        self.cyclestart = Button(self.label_men, text='Start', command=self.cycle_start,
                                 image=self.fcycle, borderwidth=0, compound=TOP, width=100, height=75,
                                 font=("Malgun Gothic", 8),
                                 state=DISABLED, bg='white')
        self.cyclestart.grid(column=2, row=2)

        # STATUS AND TIME
        self.statusbar = Label(self.label_men, text='Estado da Operação[%]:', bg='white', fg='black',
                               font=("Malgun Gothic", 8))
```

```
        self.statusbar.grid(column=0, row=3, columnspan=2, sticky='W')
        self.estimated_time = Label(self.label_men, text='Tempo Estimado:', font=("Malgun Gothic", 8), bg='white',
                                    fg='black')
        self.estimated_time.grid(column=0, row=5, columnspan=2, sticky='W')
        # PROGRESS BAR WIDGET
        self.progress = Progressbar(self.label_men, orient=HORIZONTAL, style='black.Horizontal.TProgressbar',
                                    length=390, mode='determinate')
        self.progress.grid(column=0, row=4, columnspan=3)


# Label1_1
        # BUTOES DOS PARAMETROS

        # linha 1
        self.startup_blocks = Button(self.label1_1, text='Blocos Iniciais', image=self.fstartup,
                                     compound=TOP, command=self.get_startup_blocks, width=103,
                                     font=("Malgun Gothic", 8), bg='white')
        self.startup_blocks.grid(column=0, row=0, sticky='nsew')

        self.param_grbl = Button(self.label1_1, text='Definições\ndo Sistema', image=self.fparam, compound=TOP,
                                 width=103, command=lambda: ( self.controller_param()),
                                 font=("Malgun Gothic", 8), bg='white')
        self.param_grbl.grid(column=1, row=0, sticky='nsew')
        self.cardinal_param = Button(self.label1_1, text='Parâmetros\n[G54-G59]', image=self.fparam_card, compound=TOP,
                                     width=103,
                                     command=self.get_cardinal,
                                     font=("Malgun Gothic", 8), bg='white')
        self.cardinal_param.grid(column=2, row=0, sticky='nsew')

        # MUDAR PARAMETROS

        self.label_val = Label(self.label1_1, borderwidth=0, bg='white')
        self.label_val.place(x=5, y=157)
        self.val = '$'
        self.nome = Label(self.label1_1, text='Escolher Parâmetro:', font=("Malgun Gothic", 9), bg='white')
        self.nome.place(x=5, y=110)
        self.escolhido = StringVar()
        self.w = ttk.Combobox(self.label1_1, width=20, text='Parâmetro', textvariable=self.escolhido, values=self.array)
        self.w.place(x=5, y=130)
        self.w.set('Parâmetro')
# self.numero_para_parametro = Label(self.label1_1, text='Valor Alterado:', font=("Malgun Gothic", 8), bg='white')
# self.numero_para_parametro.place(x=5, y=340)
        self.variavel = StringVar()
```

```
self.parametro_pretendido = Entry(self.label_val, textvariable=self.variavel, fg='black',
                                  font=(''Malgun Gothic'', 10))
self.parametro_pretendido.grid(column=0, row=7, columnspan=3, sticky='nsew')

# self.alterar_param = Button(self.label1_1, text= 'Alterar', command=self.escolher)
# self.alterar_param.place(x=125, y=200)
# BOTÕES DOS NUMEROS PARA ALTERAR OS PA^RAMETROS

self.expression = ""
# StringVar() is the variable class
# we create an instance of this class
self.equation = StringVar()
# create the text entry box for
# showing the expression .
self.expression_field = Entry(self.label_val, textvariable=self.equation, fg='grey', width=21)
self.expression_field.grid(column=0, row=0, columnspan=3, sticky='nsew')
self.equation.set('Valor Pretendido')
# create a Buttons and place at a particular
# location inside the root window .
# when user press the button, the command or
# function affiliated to that button is executed .
self.button1 = Button(self.label_val, text=' 1 ', fg='black',
                      command=lambda: self.press(1), height=1, width=2, bg='white', borderwidth=1)
self.button1.grid(row=2, column=0, sticky=''nsew'')

self.button2 = Button(self.label_val, text=' 2 ', fg='black',
                      command=lambda: self.press(2), height=1, width=2, bg='white', borderwidth=1)
self.button2.grid(row=2, column=1, sticky=''nsew'')

self.button3 = Button(self.label_val, text=' 3 ', fg='black',
                      command=lambda: self.press(3), height=1, width=2, bg='white', borderwidth=1)
self.button3.grid(row=2, column=2, sticky=''nsew'')

self.button4 = Button(self.label_val, text=' 4 ', fg='black',
                      command=lambda: self.press(4), height=1, width=2, bg='white', borderwidth=1)
self.button4.grid(row=3, column=0, sticky=''nsew'')

self.button5 = Button(self.label_val, text=' 5 ', fg='black',
                      command=lambda: self.press(5), height=1, width=2, bg='white', borderwidth=1)
self.button5.grid(row=3, column=1, sticky=''nsew'')

self.button6 = Button(self.label_val, text=' 6 ', fg='black',
                      command=lambda: self.press(6), height=1, width=2, bg='white', borderwidth=1)
```

```
self.button6.grid(row=3, column=2, sticky="nsew")

self.button7 = Button(self.label_val, text=' 7 ', fg='black',
                      command=lambda: self.press(7), height=1, width=2, bg='white', borderwidth=1)
self.button7.grid(row=4, column=0, sticky="nsew")

self.button8 = Button(self.label_val, text=' 8 ', fg='black',
                      command=lambda: self.press(8), height=1, width=2, bg='white', borderwidth=1)
self.button8.grid(row=4, column=1, sticky="nsew")

self.button9 = Button(self.label_val, text=' 9 ', fg='black',
                      command=lambda: self.press(9), height=1, width=2, bg='white', borderwidth=1)
self.button9.grid(row=4, column=2, sticky="nsew")

self.button0 = Button(self.label_val, text=' 0 ', fg='black',
                      command=lambda: self.press(0), height=1, width=2, bg='white', borderwidth=1)
self.button0.grid(row=5, column=0, sticky="nsew")

self.clear = Button(self.label_val, text='Clear', font=("Malgun Gothic", 8), fg='black',
                    command=self.clear, height=1, width=2, bg='white', borderwidth=1)
self.clear.grid(row=5, column='1', sticky="nsew")

self.ponto = Button(self.label_val, text=' . ', fg='black',
                    command=lambda: self.press('.'), height=1, width=2, bg='white', borderwidth=1)
self.ponto.grid(row=5, column=2, sticky="nsew")

self.enter = Button(self.label_val, text='ENTER', command=self.escolher, fg='black',
                    font=("Malgun Gothic", 8, 'bold'), height=1, width=2, bg='white', borderwidth=1)
self.enter.grid(column=0, row=6, sticky="nsew", columnspan=3)

# label1_2
# RESTORE GRBL SETTINGS
self.entry_reset = Label(self.label1_2, text='Reset Parcial ou total do Grbl:', fg='black',
                         font=("Malgun Gothic", 9), bg='white')
self.entry_reset.grid(row=0, column=0, sticky="nsew")
self.restore1 = Button(self.label1_2, text=' RESET $$ ', fg='black', font=("Malgun Gothic", 8),
                       command=self.reset_1, bg='white')
self.restore1.grid(row=1, column=0, sticky="nsew")

self.restore2 = Button(self.label1_2, text=' RESET G54-G59/G28-G30 ', fg='black',
                       font=("Malgun Gothic", 8), command=self.reset_2, bg='white')
self.restore2.grid(row=2, column=0, sticky="nsew")
```

```
self.restore3 = Button(self.label1_2, text=' RESET EEPROM ', fg='black', font=("Malgun Gothic", 8),
                       command=self.reset_3, bg='white')
self.restore3.grid(row=3, column=0, sticky="nsew")

# HOMING
self.homing = Label(self.label1_2, text='Homing:', font=("Malgun Gothic", 9), bg='white')
self.homing.grid(row=4, column=0, sticky="w")
self.ativ_home = Button(self.label1_2, text=' Ativar Homing', fg='black', font=("Malgun Gothic", 8),
                        command=self.get_homing, bg='white')
self.ativ_home.grid(row=5, column=0, sticky="nsew")

# VENTILADOR

self.vent = Label(self.label1_2, text='Ventilador:', font=("Malgun Gothic", 9), bg='white')
self.vent.grid(row=7, column=0, sticky="w")
self.ativ_vent = Button(self.label1_2, text='Ligar Aspirador', fg='black', font=("Malgun Gothic", 8),
                        bg='white')
self.ativ_vent.grid(row=8, column=0, sticky="nsew")
self.desativ_vent = Button(self.label1_2, text='Desligar Aspirador', fg='black', font=("Malgun Gothic", 8),
                           bg='white')
self.desativ_vent.grid(row=9, column=0, sticky="nsew")
# label 2
# COMANDOS MANUAIS CNC
self.label3_1 = Label(self.label3, bg='white')
self.label3_1.place(x=80, y=5)
self.move_a_pos = Button(self.label3_1, image=self.photoa_mais, command=self.movea_pos,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_a_pos.grid(row=4, column=1)
self.move_a_neg = Button(self.label3_1,  image=self.photoa_menos, command=self.movea_neg,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_a_neg.grid(row=7, column=1)

self.move_z_pos = Button(self.label3_1, image=self.photoz_mais, command=self.movez_neg,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_z_pos.grid(row=2, column=0)
self.move_z_neg = Button(self.label3_1, image=self.photoz_menos, command=self.movez_pos,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_z_neg.grid(row=2, column=2)

self.move_x_pos = Button(self.label3_1, image=self.photox_mais, command=self.movex_pos,
                         borderwidth=0, state=DISABLED, bg='white')
self.move_x_pos.grid(row=1, column=1)
self.move_x_neg = Button(self.label3_1, image=self.photox_menos, command=self.movex_neg,
```

```
                        borderwidth=0, state=DISABLED, bg='white')
        self.move_x_neg.grid(row=3, column=1)
        # OPCAO PARA COMANDOS MANUAIS

        # ESCOLHER O MOVIMENTO:
        self.palavra_escolha = Label(self.label3, text='Movimento [mm ou ◦]', font=("Malgun Gothic", 9),
                                     bg='white')
        self.palavra_escolha.place(x=250, y=280)
        self.sp = Spinbox(self.label3, from_=0, to=100, width=5, font=("Malgun Gothic", 14, 'bold'))
        self.sp.place(x=250, y=300)
        self.opcao = ''
#ZERAR
        self.zerar = Label(self.label3, text='Zero Peça:', font=("Malgun Gothic", 10, 'bold'), bg='white')
        self.zerar.place(x=0, y=120)
        self.z_zero = Button(self.label3, text='Z0', font=("Malgun Gothic", 9, 'bold'),
                             bg='white', command=self.zerar_z)
        self.z_zero.place(height=50, width=80,x=0, y=140)

        self.x_zero = Button(self.label3, text='X0', font=("Malgun Gothic", 9, 'bold'),
                             bg='white', command=self.zerar_x)
        self.x_zero.place(height=50, width=80, x=0, y=190)
        self.a_zero = Button(self.label3, text='A0', font=("Malgun Gothic", 9, 'bold'),
                             bg='white', command=self.zerar_x)
        self.a_zero.place(height=50, width=80, x=0, y=240)
#HOMING DO Z
        self.homingZ = Label(self.label3, text='Zero Máquina:', font=("Malgun Gothic", 10, 'bold'), bg='white')
        self.homingZ.place(x=0, y=290)
        self.home_z = Button(self.label3, text='Home Z', fg='black', font=("Malgun Gothic", 9),
                             command=self.homing_z, bg='white')
        self.home_z.place(height=50, width=80,x=0, y=310)

#COORDENADAS ABSOLUTAS E RELATIVAS
        self.coord = Label(self.label3, text='Coordenadas:', font=("Malgun Gothic", 10, 'bold'), bg='white')
        self.coord.place(x=0, y=0)
        self.g90 = Button(self.label3, text='Absolutas\nG90', font=("Malgun Gothic", 9, 'bold'),
                             bg='white', command=self.absoluto)
        self.g90.place(height=50, width=80,x=0, y=20)
        self.g91 = Button(self.label3, text='Relativas\nG91', font=("Malgun Gothic", 9,'bold'),
                             bg='white', command=self.relativo)
        self.g91.place(height=50, width=80,x=0, y=70)

# LABEL 5
        erros_button = Button(self.label5, text="Erros e Alarmes", command=self.erros_win)
```

```
        erros_button.place(height=120, width=390, x=0, y=0)
        botoes_button = Button(self.label5, text="Botões", command=self.botoes_win)
        botoes_button.place(height=120, width=390, x=0, y=120)
        param_grbl_button = Button(self.label5, text="Parâmetros Grbl", command=self.param_win)
        param_grbl_button.place(height=120, width=390, x=0, y=240)

        self.read_grbl = self.serial.read(self.serial.inWaiting())
        self.insert_grbl()

        self.root.mainloop()



    def write_on(self):
        self.serial.write(self.gcode)

    def thread_gcode(self):
        thread = threading.Thread(target=self.write_on, daemon=True)
        thread.start()
        if thread.is_alive():
            print('is alive')
        return

# mover para posicao 30
    def move_30(self):
        self.gcode = b'G90 Z30\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

#texto menu ajuda
    #ERROS e ALARMES

    def erros_win(self):
        self.top = Toplevel(self.label5)
        self.top.geometry("400x400")
        self.erros = Label(self.top)
        self.erros1 = tkst.ScrolledText(self.erros, bg="white", fg="grey", font=("Malgun Gothic", 8))
        self.erros1.place(height=360, width=395, x=0, y=0)
        self.erros1.yview(END)
        self.erros1.insert(1.0, self.texto_ajuda)
        self.b1 = Button(self.top, text="Voltar", command=lambda win=self.top: win.destroy())
```

```
    self.erros.place(height=360, width=395, x=0, y=0)
    self.b1.pack(side="bottom")
    return

#BOTOES
def botoes_win(self):
    self.top = Toplevel(self.label5)
    self.top.geometry("400x400")
    self.erros = Label(self.top)
    self.bt1 = tkst.ScrolledText(self.erros, bg="white", fg="grey", font=("Malgun Gothic", 8))
    self.bt1.place(height=360, width=395, x=0, y=0)
    self.bt1.yview(END)
    self.bt1.insert(1.0, self.Bt_def)
    self.b1 = Button(self.top, text="Voltar", command=lambda win=self.top: win.destroy())
    self.erros.place(height=360, width=395, x=0, y=0)
    self.b1.pack(side="bottom")
    return
#GRBL PARAM
def param_win(self):
    self.top = Toplevel(self.label5)
    self.top.geometry("400x400")
    self.erros = Label(self.top)
    self.bt1 = tkst.ScrolledText(self.erros, bg="white", fg="grey", font=("Malgun Gothic", 8))
    self.bt1.place(height=360, width=395, x=0, y=0)
    self.bt1.yview(END)
    self.bt1.insert(1.0, self.ajuda_param)
    self.b1 = Button(self.top, text="Voltar", command=lambda win=self.top: win.destroy())
    self.erros.place(height=360, width=395, x=0, y=0)
    self.b1.pack(side="bottom")
    return

# marcar zero de Z e X
def zerar_z(self):
    self.gcode = b'G92 Z0\n'
    self.thread_gcode()
    self.reading = self.gcode.decode()
    self.insert_interface_user()
    self.main_read()
    return

def zerar_x(self):
    self.gcode = b'G92 X0\n'
    self.thread_gcode()
```

```
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return


# ativar o homing
    def get_homing(self):
        self.gcode = b'$22=1\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# fazer homing do eixo Z
    def homing_z(self):
        self.gcode = b'$hz\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# ler o status do grbl
    def get_status(self):
        self.gcode = '?'.encode()
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# Ler os parâmetros G
    def get_cardinal(self):
        self.gcode = b'$#\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# ler a build info
```

```
    def get_buil_info(self):
        self.gcode = b'$I\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# ler os startuo blocks
    def get_startup_blocks(self):
        self.gcode = b'$N\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# IDLE/Check
    def get_gcode_mode(self):
        self.gcode = b'$C\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# recomeçar um ciclo de gcode
    def cycle_start(self):
        self.gcode = b'~\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        return

# parar um ciclo de gcode
    def get_feedhold(self):
        self.gcode = b'!\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.cyclestart.config(state='normal')
        #self.main_read()
        return
```

```
# fazer soft reset(IDLE e unlock e parãmtros)
    def soft_reset(self):
        self.gcode = b'(CTRL-X)\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# fazer Reset dos parãmetros
    def reset_1(self):
        self.gcode = b'$RST=$\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# fazer reset dos parâmetros e dos blocos iniciais
    def reset_2(self):
        self.gcode = b'$RST=#\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# fazer Reset de tudo no grbl
    def reset_3(self):
        self.gcode = b'$RST=*\n'
        self.thread_gcode()
        self.reading = self.gcode.decode()
        self.insert_interface_user()
        self.main_read()
        return

# Inserir o texto na interface que o utilizador enviou
    def insert_interface_user(self):
        user = '{} {}\n'.format('user:', self.reading)
        self.interface.insert(END, user)
        self.interface.yview(END)
        return
```

```
# Procurar, escolher e inserir o ficheiro gcode
    def controller_getfilename(self):
        self.file_name = self.control.get_path()
        # guardar o path para posteriormente usar para enviar o ficheiro por serial
        path_gcode = self.file_name
        print(path_gcode)
        # por o texto no widget path
        self.entry_file_path.delete('1.0', END)
        self.entry_file_path.config(state="normal")
        self.entry_file_path.insert(END, path_gcode)
        self.entry_file_path.config(state=DISABLED)
        self.send_code.config(state='normal')
        return

# ler a resposta do Grbl
    def controller_getread(self):
        while True:
            if  self.serial.inWaiting()>0:
                self.data_str = self.control.get_read()
                self.read_grbl = '{}'.format(self.data_str)
                if len(self.data_str) > 0:
                    self.insert_grbl()
            time.sleep(2)

    def ask_for_response(self):
        while self.controller_getread():
            if self.data_str.find('ok')>=0:
                self.read_grbl=self.control.get_read()
                self.insert_grbl()


 #fazer thread da leitura do grbl
    def main_read(self):
        thread = threading.Thread(target=self.ask_for_response, daemon=True)
        thread.start()
        if thread.is_alive():
            print('is alive')
        return

# dar unlock do modo ALARM do grbl
    def controller_unlock(self):
        unlock = self.control.get_unlock()
```

```
            self.control.write_serial()
            self.reading = unlock
            self.insert_interface_user()
            self.interface.yview(END)
            self.move_a_neg.config(state='normal')
            self.move_a_pos.config(state='normal')
            self.move_x_neg.config(state='normal')
            self.move_x_pos.config(state='normal')
            self.move_z_neg.config(state='normal')
            self.move_z_pos.config(state='normal')
            self.search_file_button.config(state='normal')
            self.main_read()
            return


        def insert_grbl(self):
            if len(self.read_grbl)>0:
                self.grbl = '{} {}\n'.format('grbl:', self.read_grbl)
                self.interface.insert(END, self.grbl)
                self.interface.yview(END)
            return


    # ver parser state
        def controller_parser(self):
            parser_state = self.control.get_parser_state()
            self.control.write_serial()
            self.reading = parser_state
            self.insert_interface_user()
            self.interface.yview(END)
            self.main_read()
            return

    # ver parametros do grbl
        def controller_param(self):
            param = self.control.get_param()
            self.control.write_serial()
            self.reading = param
            self.insert_interface_user()
            self.interface.yview(END)
            self.main_read()
            return
```

```
##################################### ENVIAR GCODE ################################################
#enviar o gcode para o grbl
    #def controller_send_gcode(self):
     #   user = 'Enviando o ficheiro...\n'
      # self.interface.insert(END, user)
       # self.interface.yview(END)
        #while self.control.get_gcode:
         #   if self.serial.inWaiting() >  0:
          #      self.control.get_gcode()
           #     self.grbl = self.control.talking()
            #    self.interface.insert(END, self.grbl)
             #   self.interface.yview(END)
        #time.sleep(1)
        #user = 'Ficheiro enviado!\n'
        #self.interface.insert(END, user)
        #self.interface.yview(END)
        #time.sleep(2)
    # Periodic timer to query for status reports
    def send_status_query(self):
        self.serial.write('?'.encode())

    def periodic_timer(self):
        while self.is_run:
            self.send_status_query()
            time.sleep(self.REPORT_INTERVAL)

    def read_gcode(self):
        self.response=False
        with open(self.file_name, 'r') as f:

            # Wait for grbl to initialize and flush startup text in serial input
            time.sleep(5)
            self.serial.flushInput()
            self.reading = 'Streaming Gcode...'
            self.insert_interface_user()

            if self.check_mode:
                print("Enabling Grbl Check-Mode: SND: [$C]"),

                self.serial.write("$C".encode())
                while 1:
                    self.grbl_out = self.serial.readline().strip().decode()  # Wait for grbl response with carriage return
                    if self.grbl_out.find('error') >= 0:
```

```
            print("REC:", self.grbl_out)

            print("  Failed to set Grbl check-mode. Aborting...")
            quit()
        elif self.grbl_out.find('ok') >= 0:
            if self.verbose:
                print('REC:', self.grbl_out)
            break

self.start_time = time.time()

# Start status report periodic timer
if self.ENABLE_STATUS_REPORTS:
    self.timerThread = threading.Thread(target=self.periodic_timer)
    self.timerThread.daemon = True
    self.timerThread.start()

    # Stream g-code to grbl
self.l_count = 0
self.error_count = 0
if self.settings_mode:
    print("SETTINGS MODE: Streaming", self.args.gcode_file.name, " to ", self.args.device_file)
    for line in f:
        self.l_count += 1  # Iterate line counter
            # l_block = re.sub('\s|\(.*?\)','',line).upper() # Strip comments/spaces/new line and capitalize
        l_block = line.strip()  # Strip all EOL characters for consistency
        if self.verbose: print("SND>" + str(self.l_count) + ": \"" + l_block + "\"")
        self.resulta = l_block + '\n'
        self.talk = self.serial.write(self.resulta.encode())  # Send g-code block to grbl
        self.grbl = '{}'.format(self.resulta)
        self.interface.insert(END, self.grbl)
        self.interface.yview(END)
        while 1:
            self.rbl_out = self.serial.readline().strip().decode()  # Wait for grbl response with carriage return
            if self.grbl_out.find('ok') >= 0:
                if self.verbose:
                    print("  REC<" + str(self.l_count) + ": \"" + self.grbl_out + "\"")
                break
            elif self.grbl_out.find('error') >= 0:
                if self.verbose:
                    print("  REC<" + str(self.l_count) + ": \"" + self.grbl_out + "\"")
                self.error_count += 1
                break
```

```
            else:
                print("    MSG: \"" + self.grbl_out + "\"")
    else:
        self.g_count = 0
        self.c_line = []
        for line in f:
            self.l_count += 1  # Iterate line counter
            l_block = re.sub('\s|\(.*?\)', '',
                             line).upper()  # Strip comments/spaces/new line and capitalize
            # l_block = line.strip()
            self.c_line.append(len(l_block) + 1)  # Track number of characters in grbl serial read buffer
            self.grbl_out = ''
            while sum(self.c_line) >= self.RX_BUFFER_SIZE - 1 | self.serial.inWaiting():
                out_temp = self.serial.readline().strip().decode()  # Wait for grbl response
                if out_temp.find('ok') < 0 and out_temp.find('error') < 0:
                    print("    MSG: \"" + out_temp + "\"")  # Debug response
                else:
                    if out_temp.find('error') >= 0: self.error_count += 1
                    self.g_count += 1  # Iterate g-code counter
                    if self.verbose: print("  REC<" + str(self.g_count) + ": \"" + out_temp + "\"")
                    del self.c_line[0]  # Delete the block character count corresponding to the last 'ok'
            resulta = l_block + '\n'
            self.talk = self.serial.write(resulta.encode())  # Send g-code block to grbl
            self.grbl = '{}'.format(resulta)
            self.interface.insert(END, self.grbl)
            self.interface.yview(END)
            if self.verbose: print("SND>" + str(self.l_count) + ": \"" + l_block + "\"")
            # Wait until all responses have been received.
        while self.l_count > self.g_count:
            self.out_temp = self.serial.readline().strip().decode()  # Wait for grbl response
            if self.out_temp.find('ok') < 0 and self.out_temp.find('error') < 0:
                print("    MSG: \"" + self.out_temp + "\"")  # Debug response
            else:
                if self.out_temp.find('error') >= 0: self.error_count += 1
                self.g_count += 1  # Iterate g-code counter
                del self.c_line[0]  # Delete the block character count corresponding to the last 'ok'
                if self.verbose: print("  REC<" + str(self.g_count) + ": \"" + out_temp + "\"")

        # Wait for user input after streaming is completed
    print("\nG-code streaming finished!")
    self.end_time = time.time()
    is_run = False
    print(" Time elapsed: ", self.end_time - self.start_time, "\n")
```

```
            if self.check_mode:
                if self.error_count > 0:
                    print("CHECK FAILED:", self.error_count, "errors found! See output for details.\n")
                else:
                    print("CHECK PASSED: No errors found in g-code program.\n")
            else:
                print("WARNING: Wait until Grbl completes buffered g-code blocks before exiting.")
                input("  Press <Enter> to exit and disable Grbl.")

            # Close file and serial port
            f.close()
            self.serial.close()
            time.sleep(2)

        # fazer thread da leitura do grbl
    def da_por_favor(self):
        thread = threading.Thread(target=self.read_gcode, daemon=True)
        thread.start()
        if thread.is_alive():
            print('is alive')
        return
########################################################################################################################
# BOTOES MANUAIS DEF

    ################## PARAMETROS G90 G91 ###################################
    def relativo(self):
        self.parametros_iniciais = "G91 G1 f500".encode()
        print(self.parametros_iniciais)
        self.g91.configure(relief='sunken', bg='light grey')
        self.g90.configure(relief='groove', bg='white')
        return self.parametros_iniciais

    def absoluto(self):
        self.parametros_iniciais = "G90 G1 f500".encode()
        print(self.parametros_iniciais)
        self.g90.configure(relief=SUNKEN, bg='light grey')
        self.g91.configure(relief='groove', bg='white')
        return self.parametros_iniciais

    # mover eixo do x
    def movex_neg(self):
        self.opcao = self.sp.get()
```

```
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "X-{}".format(self.opcao)
        print(self.gcode)
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return


    def movex_pos(self):
        self.opcao = self.sp.get()
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "X{}".format(self.opcao)
        print(self.gcode)
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return

    # mover eixo do z
    def movez_neg(self):
        self.opcao = self.sp.get()
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "Z-{}".format(self.opcao)
        print(self.gcode)
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
```

```
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return

    def movez_pos(self):
        self.opcao = self.sp.get()
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "Z{}".format(self.opcao)
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return

    # mover eixo do a
    def movea_neg(self):
        self.opcao = self.sp.get()
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "A-{}".format(self.opcao)
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return

    def movea_pos(self):
        opcao = self.sp.get()
        self.gcode = self.parametros_iniciais
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.gcode = "A{}".format(opcao)
```

```
        self.gcode=str.encode(self.gcode)
        self.thread_gcode()
        self.reading = self.gcode
        self.insert_interface_user()
        self.interface.yview(END)
        self.main_read()
        return

# ALTERAR PARAMETROS
        # Function to update expression
        # in the text entry box

    def press(self, num):
        # concatenation of string
        self.expression = self.expression + str(num)
        # update the expression by using set method
        self.equation.set(self.expression)

    def escolher(self):
        val = '$'
        valor = self.equation.get()
        escolha = self.escolhido.get()
        self.val = val + escolha + '=' + valor + '\n'
        print(self.val)
        code = self.val.encode()
        self.reading = code
        self.insert_interface_user()
        self.serial.write(code)
        self.variavel.set(self.val)
        return

    def clear(self):
        self.expression = ""
        self.equation.set("")
```

Apêndice VII. Manual de Instruções da Máquina

Manual de utilizador
CNC “Nada”
Padrão Ortopédico

por

Francisca Vigo

Janeiro, 2021

Índice

Índice de Figuras

Índice de Tabelas

1. Componentes da Máquina

A CNC, nomeada de “Nada”, foi construída na empresa “Padrão Ortopédico” pelo Eng. Joel Gomes e automatizada pelos estagiários Ivo Lopes e Francisca Vigo.

Todas as especificações encontradas neste manual são para a maquinagem de **blocos cilíndricos de poliuretano de baixa densidade.**

A máquina possui uma parte mecânica e eletrónica. As dimensões da máquina estão representadas na figura 1.

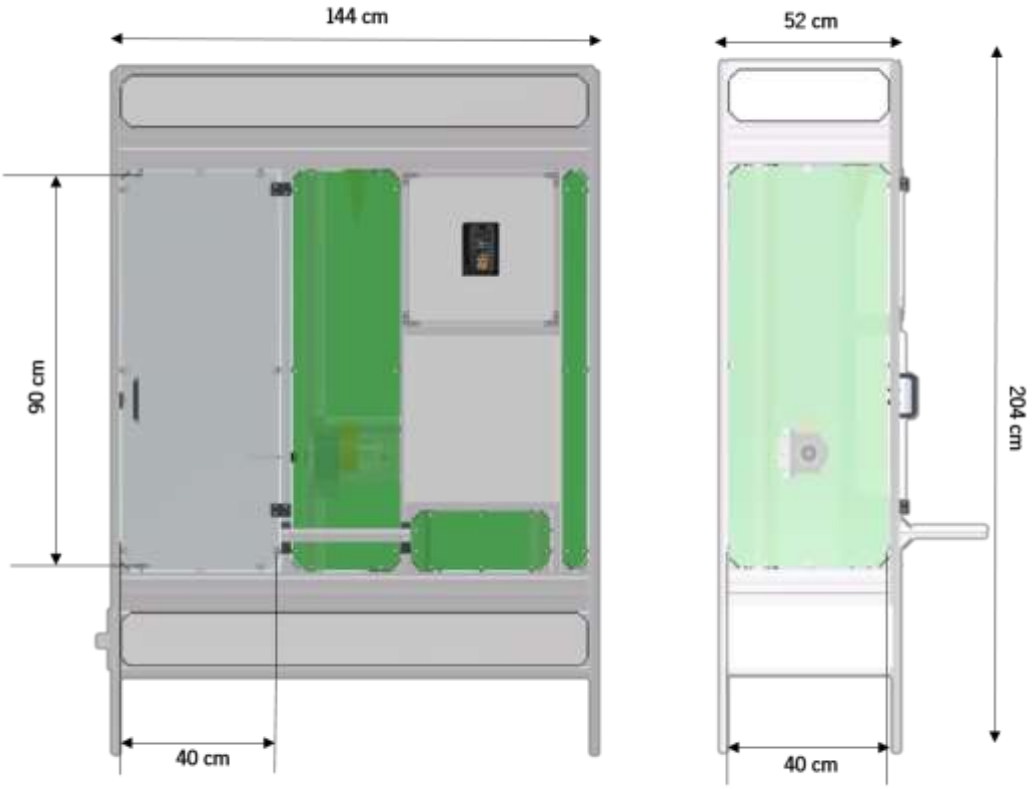


Figura 1 - Dimensões da máquina "Nada".

O **espaço de trabalho** da CNC compreende uma altura de **90 cm**, com uma base de **40 x 40 cm**. Para colocar o bloco na máquina é necessário aceder à porta (esquerda na figura) e, já com o bloco ficado ao prato, colocar o veio do prato no suporte encontrado na área.

A ferramenta deverá estar fora da área do bloco.

É desejável que o molde a maquinar não exceda o diâmetro de 40 cm nem a altura de 90 cm.

Os componentes mecânicos e a respetiva designação estão representadas nas figuras [2-4].

Os componentes da interface gráfica estão representados nas figuras [5-10].

1.1. Componentes Mecânicos

Para executar uma maquinagem é necessário que o operador tenho acesso à interface da máquina, ao controlador da spindle, ao quadro elétrico e à porta da área de trabalho, identificados na figura 2.

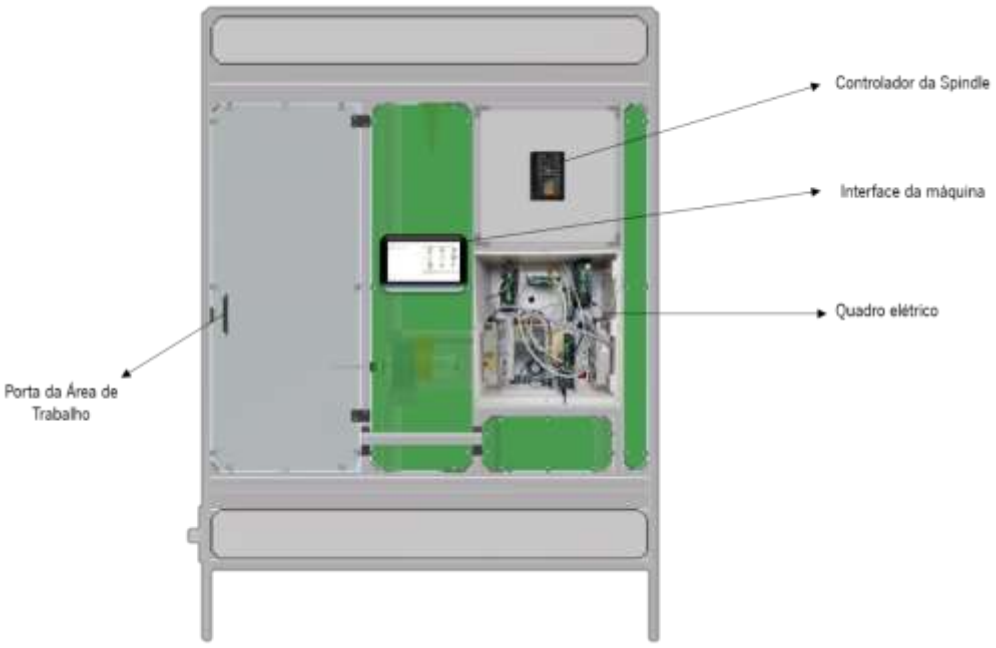


Figura 2 - Identificação de componentes da máquina.

A CNC comporta **três eixos** de movimento: **X(vertical), Z (horizontal) e A (rotacional em X).** os componentes principais desses eixos estão enunciados na tabela 1.

Sistema de transmissão		Especificações
Eixo A	Motor (2)	Nema 23
	Polia (2)	14 dentes e 34 dentes
	Correia	Passo de 10 mm
Eixo X	Motor (2)	Nema 34
	Polia (2)	14 dentes
	Correia	Passo de 10 mm
Eixo Z	Motor	Nema 23
	Fuso de esferas	Passo de 2 mm; Curso de 500 mm
Cabeçote Móvel		Largura = 400 mm; Altura = 200 mm
Porta ferramentas		Diâmetro de 30 mm
Ferramenta de corte (fresa)		Diâmetro de 10 mm; 4 hélices; ponta redonda

Tabela 1 - Componentes mecânicos da CNC.

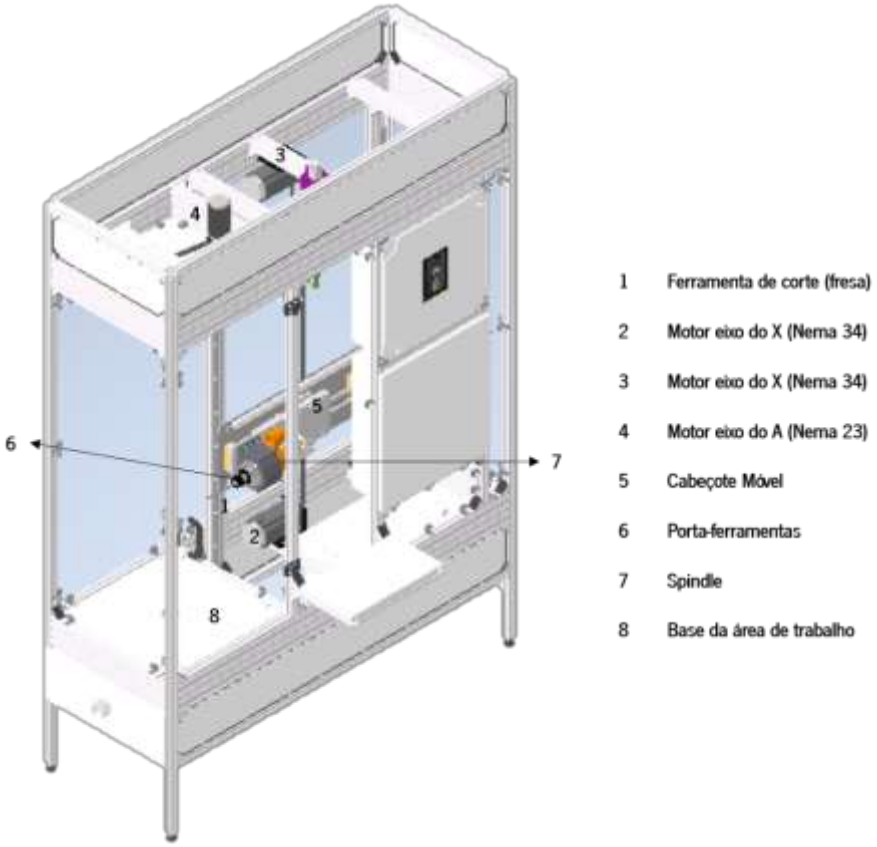


Figura 3 - Identificação de componentes da máquina.

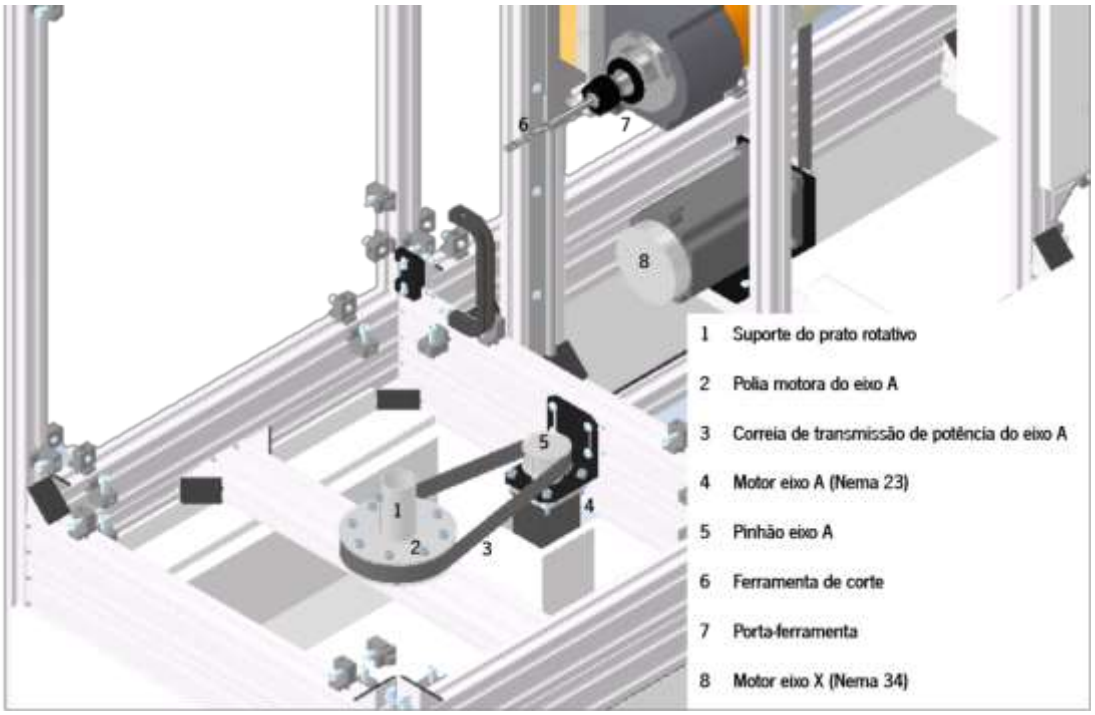


Figura 4 - Identificação de componentes da máquina.

1.2. Interface Gráfica

A interface da Máquina comporta quatro secções de trabalho, “Menu Principal”, “Comando Manual”, “Definições” e “Ajuda”.

1.2.1. **Menu principal** (Representado na figura 5)

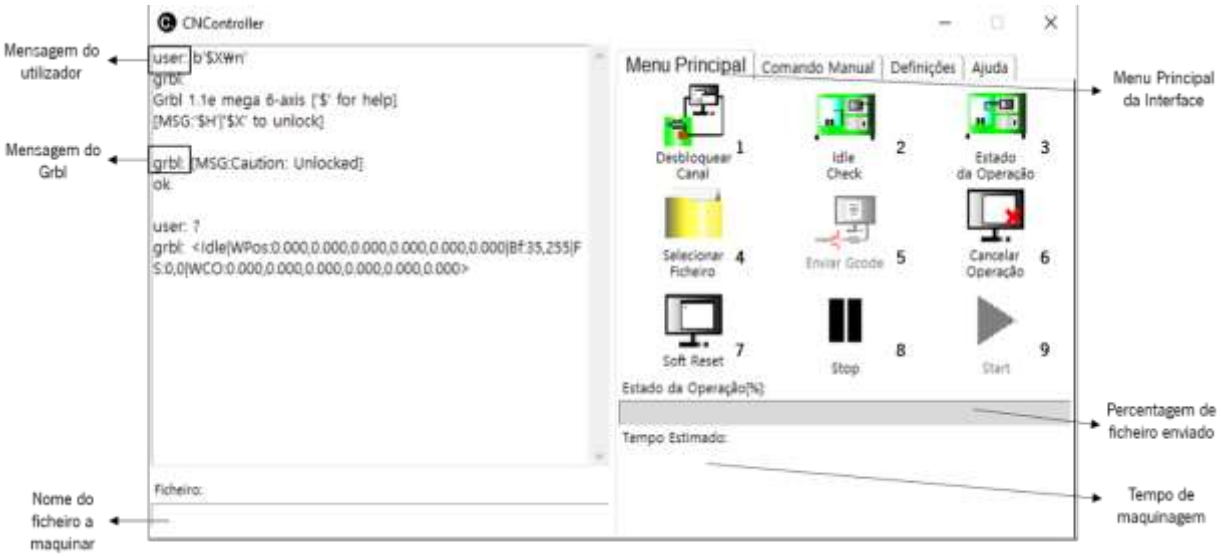


Figura 5 - Menu principal da Interface.

Tabela 2 - Designação das definições de cada botão da figura 5.

Botão	Função	Código
1	Desbloquear o canal em série	$X
2	Mudar de modo: Idle ou Check	$C
3	Estado da Máquina	?
4	Escolher ficheiro para maquinagem	-
5	Enviar ficheiro	-
6	Cancelar Operação	Ctrl-X
7	Soft Reset	Ctrl-x
8	Stop	!
9	Start (depois de paragem)	~

1.2.2. **Comando Manual** (Representado na figura 6)

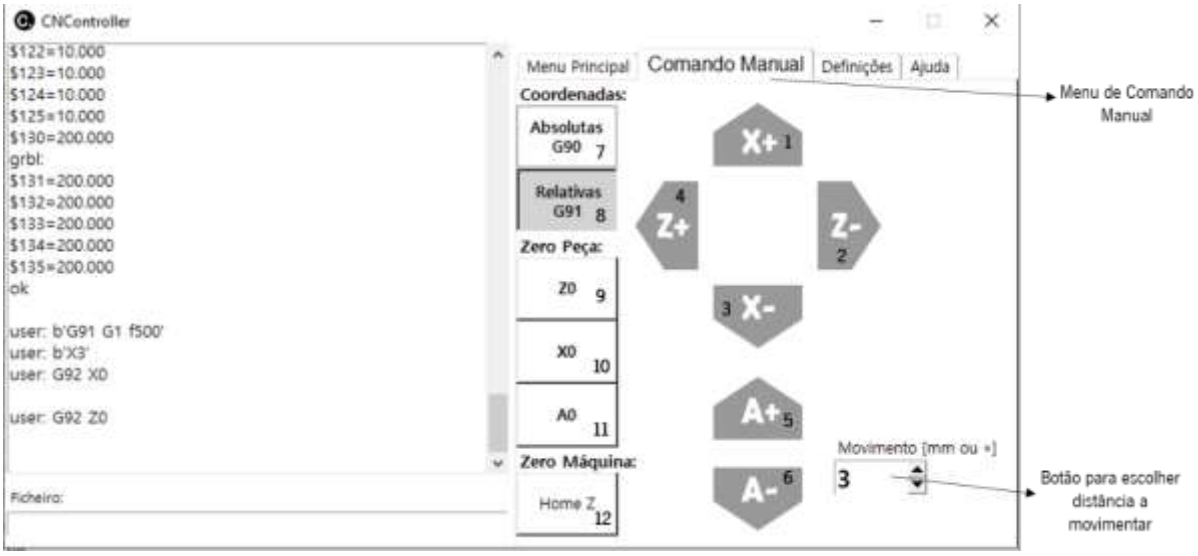


Figura 6 - Comando Manual da Interface.

Tabela 3 - Designação das definições de cada botão da figura 6.

Botão	Função	Código
1	Mover eixo X positivo (+)	Xval
2	Mover eixo Z negativo (-)	-Zval
3	Mover eixo X negativo (-)	-Xval
4	Mover eixo Z positivo (+)	Zval
5	Mover eixo A positivo (+)	Aval
6	Mover eixo A negativo (-)	-Aval
7	Pôr máquina em coordenadas absolutas	G90 G1 f500
8	Pôr máquina em coordenadas relativas	G91 G1 f500
9	Zero peça em Z	G92 Z0
10	Zero peça em X	G92 X0
11	Zero peça em A	G92 A0
12	Homing do eixo Z	Hz

Nota: val é o número escolhido pelo utilizador no botão para escolher a distância a percorrer

1.2.3. **Definições** (Representado na figura 7)

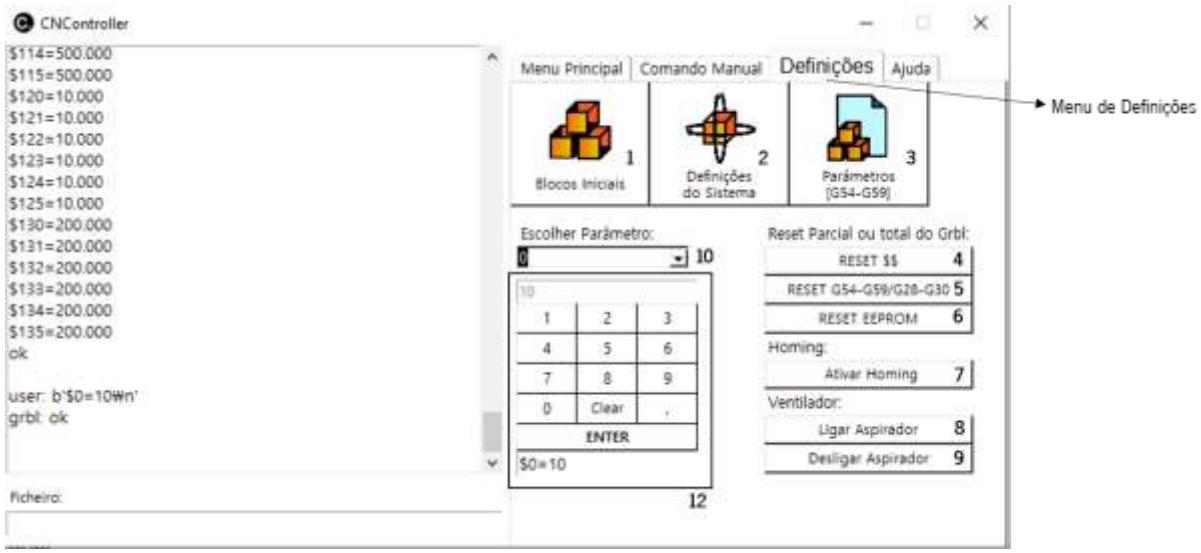


Figura 7 - Definições da Interface.

Tabela 4 - Designação de cada botão representado na figura 7.

Botão	Função	Código
1	Mostrar blocos iniciais	$G
2	Mostrar parâmetros da CNC	$$
3	Mostrar parâmetros G54-G59	$#
4	Reset do Grbl	$RST=$
5	Reset dos parâmetros G54-G59	$RST=#
6	Reset do EEPROM	$RST=*
7	Ativar opção de homing	$22=1
8	Ligar aspirador	-
9	Desligar aspirador	-
10	Escolher o parâmetro **$val***	-
11	Escolher numerário do parâmetro*	-

Especificação da seleção da variável “val”, figura 5.

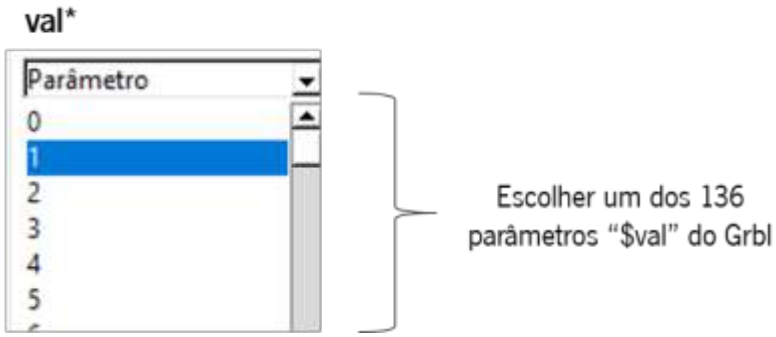


Figura 8 - Seleção do parâmetro "val".

Selecionar o parâmetro que se pretende alterar, de 0 a 136, na lista de parâmetros.

Para marcar o valor do parâmetro, figura 10, utilizam-se os botões com números de 0 a 9, o ponto para adicionar numerário decimal.

Caso se pretenda apagar o número marcado, carregar no botão “Clear”.

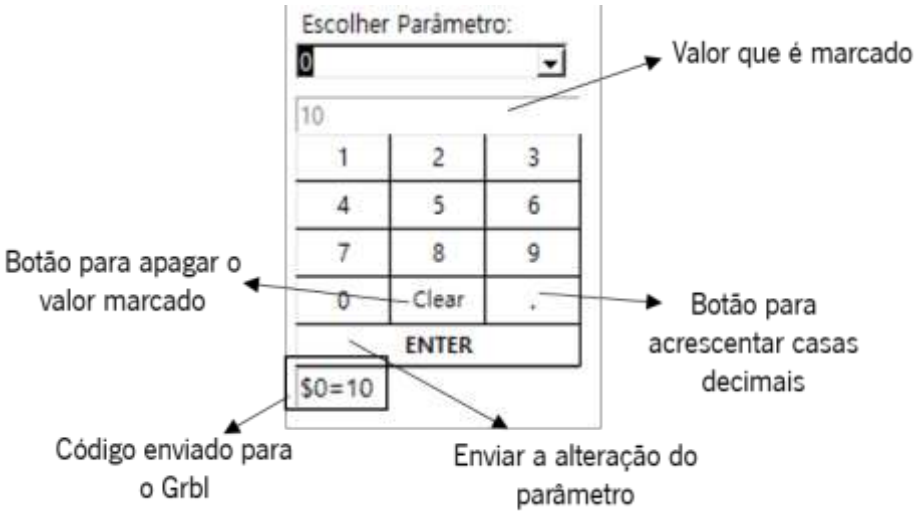


Figura 9 - Introdução do valor do parâmetro.

1.2.4. **Ajuda** (Representado na figura 8)

Figura 10 - Ajuda da interface.

2. Instruções de Maquinagem

A lista de instruções está numerada por ordem de função nas figuras deste tópico. Qualquer função específica ao operador será evidenciada seguida da figura (figuras 9 a 15).

A CNC DEVE ESTAR LIGADA À CORRENTE.

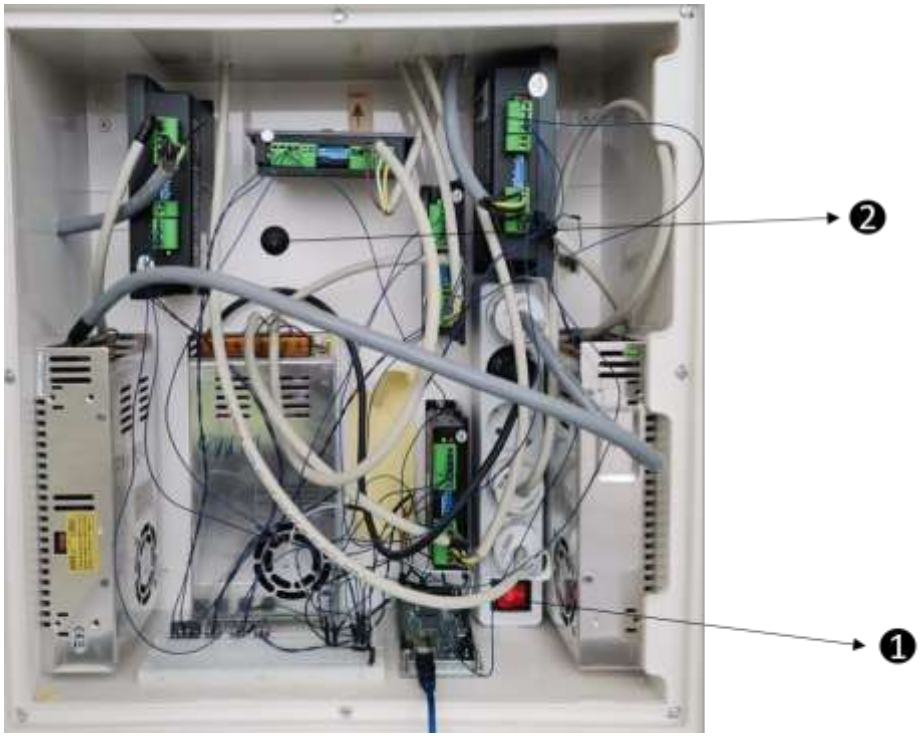


Figura 11 - Ligar a máquina e desativar os travões dos motores.

- Ligar o botão da máquina;
- Ligar o botão de bloqueio dos motores;

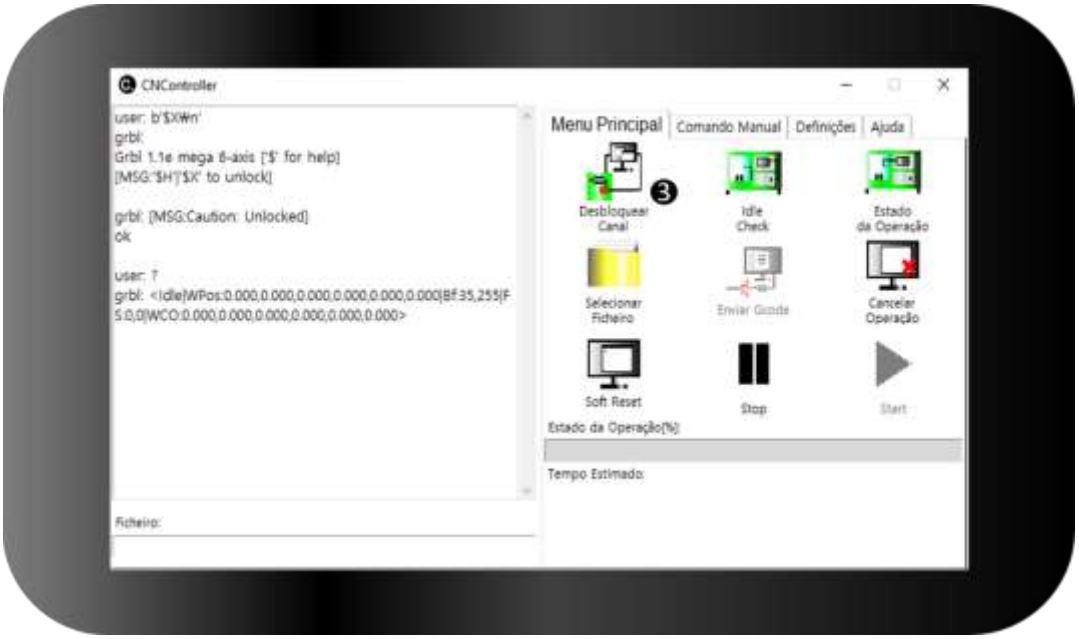


Figura 12 - Ativar o Canal da Máquina.

- Clicar no botão de “**Desbloquear Canal**”.

O Grbl vai enviar uma mensagem para o visor a referir-se como desbloqueado.

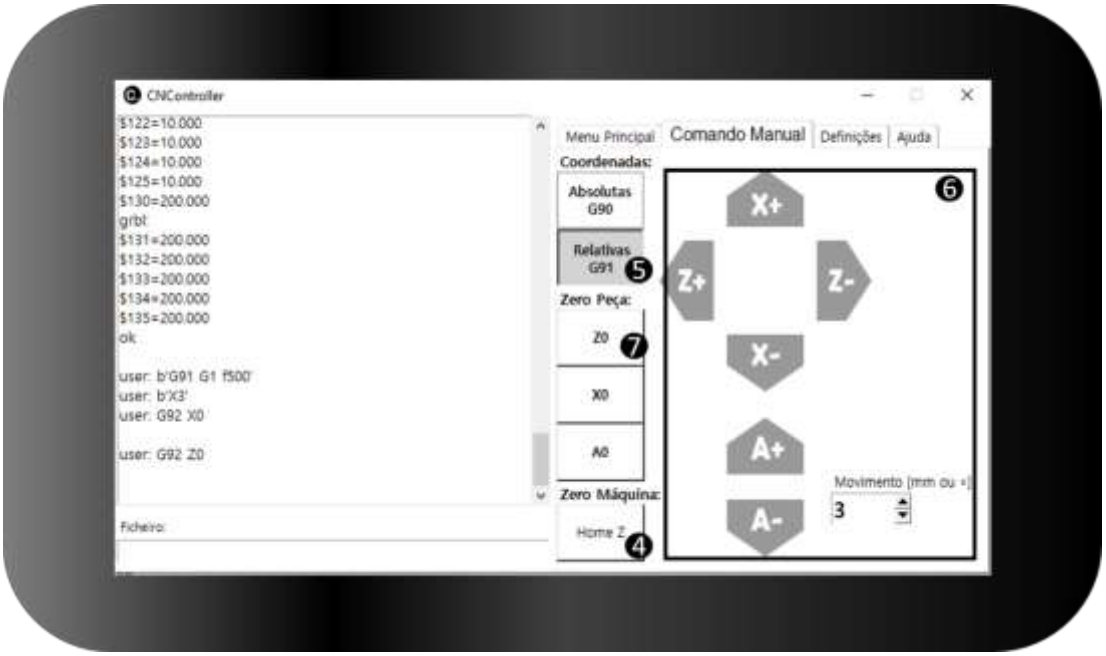


Figura 13 - Definir os zeros peça.

- Clicar no botão de homing do eixo Z, “**Home Z**”;
- Clicar no botão para coordenadas relativas, “**Relativas G91**”;
- Mover o **eixo Z 17 mm para a direita** (sentido **negativo**);
- Clicar no botão “**Z0**”;

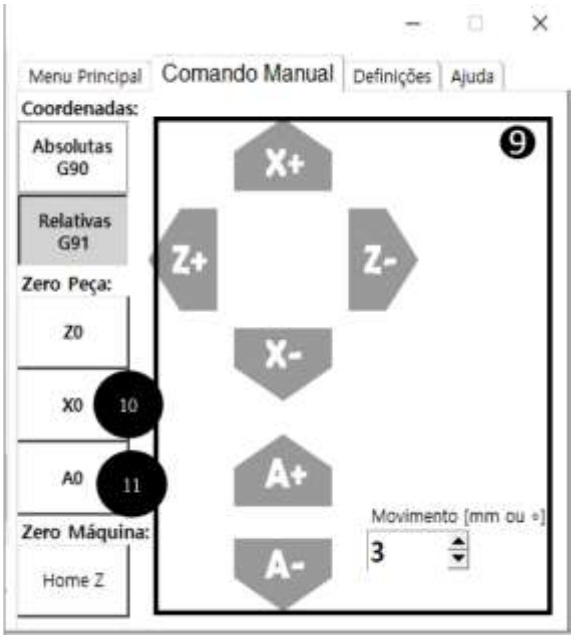


Figura 14 - Colocação do bloco na máquina e zeros do eixo X e A.

PODERÁ TER DE SE DESLOCAR O EIXO Z PARA A DIREITA PARA COLOCAR O BLOCO.

- Colocar o prato com o bloco no suporte e apertar o parafuso;
- Deslocar o eixo X para a base do bloco (ou altura que se pretende maquinar);
- Clicar no botão "**X0**";
- Clicar no botão "**A0**".

Figura 15 - Ativar a spindle.

- No controlador da spindle, utilizar o **botão radial** para aumentar a frequência para um valor entre **85 e 90 Hz.**

A FERRAMENTA DE CORTE COMEÇARÁ A GIRAR A UMA VELOCIDADE DE 2000 mm/min.

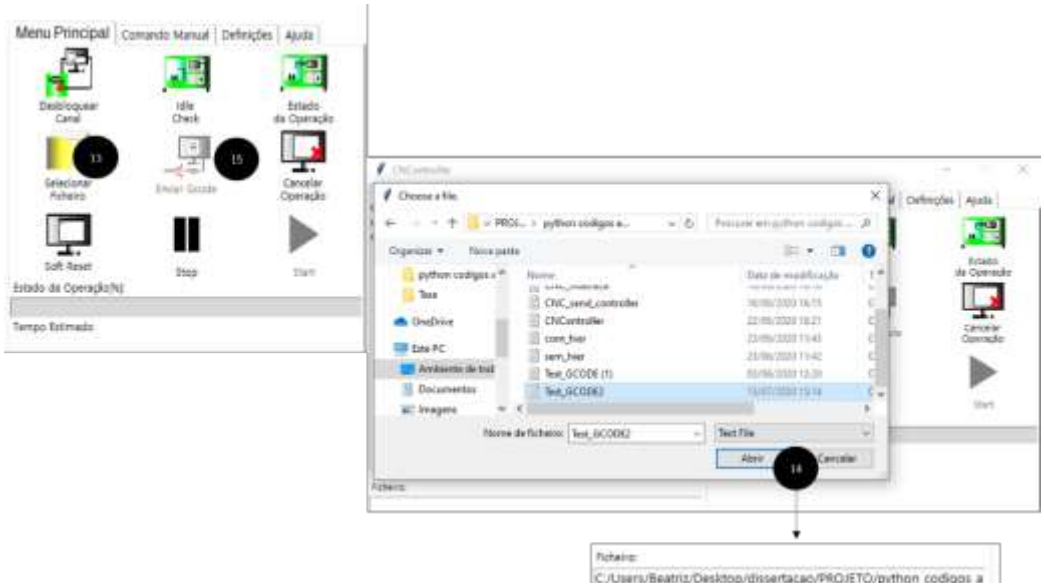


Figura 16 - Escolha do ficheiro com o código G.

- Clicar no botão "**Selecionar Ficheiro**";
- Escolher o ficheiro de código G na janela que abre.
- Clicar no botão "**Abrir**" da janela.
- Clicar no botão "**Enviar Gcode**"

A MÁQUINA COMECARÁ A MAQUINAGEM NESTE PONTO.

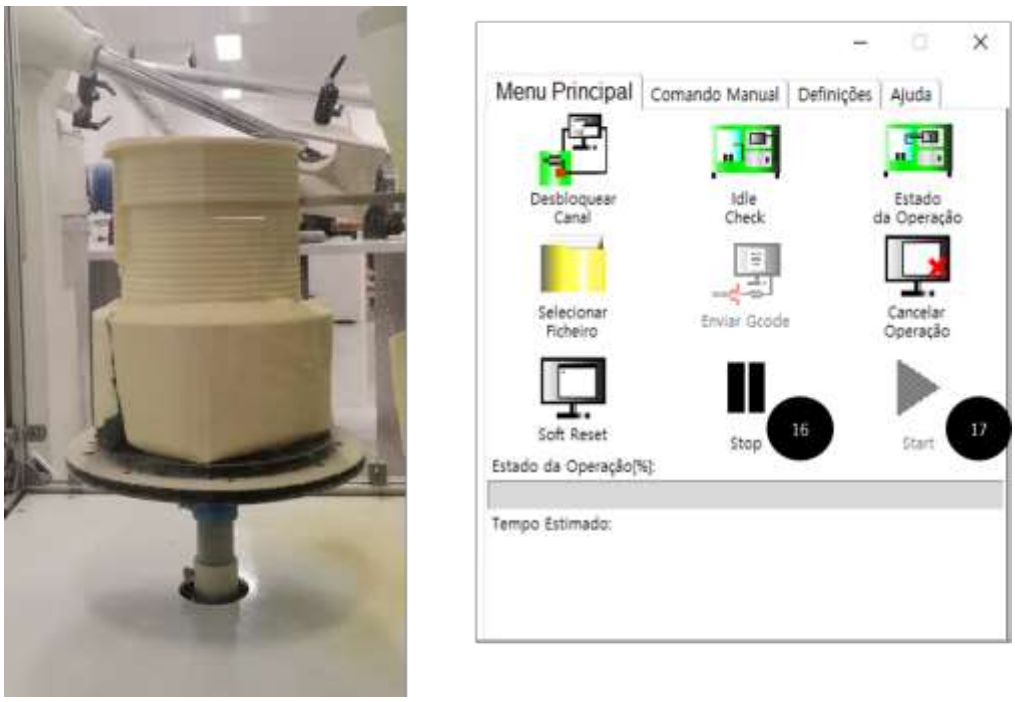


Figura 17 - Paragem e recomeço de maquinagem.

- Caso se pretenda parar a maquinagem, momentaneamente, clicar no botão "**Stop**";
- Para recomeçar a maquinagem, após paragem, clicar no botão "**Start**".

No final da maquinagem, a ferramenta desloca-se da peça e é necessário **devolver a frequência zero à spindle no botão radial do controlador.**

3. Paragem de Emergência da CNC e desligar a máquina

Caso, aquando uma operação, for necessário desligar a máquina, os passos de **EMERGÊNCIA** são:

- Clicar no botão "**Stop**" da interface;
- No **Controlador de spindle**, colocar a frequência a **zero** no **botão radial** (mover máximo da direita para a esquerda).
- Ativar o **botão de travões dos motores** no **quadro elétrico**.
- **Desligar a corrente** no botão do **quadro elétrico**.

Para desligar a máquina após uma operação, os passos são idênticos, mas sem necessidade de parar a maquinagem:

- No **Controlador de spindle**, colocar a frequência a **zero** no **botão radial** (mover máximo da direita para a esquerda).
- Ativar o **botão de travões dos motores** no **quadro elétrico**.
- **Desligar a corrente** no botão do **quadro elétrico**.

DEVE-SE DESLIGAR A TOMADA DA MÁQUINA APÓS UTILIZAÇÃO.

4. Lista de Parâmetros para alterar no Grbl

Lista de parâmetros que se podem alterar no Grbl para **modificar o funcionamento da CNC**. Os parâmetros estão representados nas tabelas 5 e 6.

Tabela 5 - Parâmetros do Grbl.

Parâmetro $x = val	Valor padrão	Unidade	Definição
$0	10	µs	Pulso de step (microssegundo)
$1	25	ms	Delay de leitura de step
Parâmetro $x = val	Valor padrão	Unidade	Definição
$2	0	mask	Inversão da porta dos steps
$3	6	mask	Inversão da direção
$4	0	bool	Ativar inversão de steps
$5	0	bool	Inversão do pino de limite de eixo
$6	0	bool	Inversão de pino de procura do limite
$10	3	mask	Reportar situação da máquina
$11	0.020	mm	Desvio de junção
$12	0.002	mm	Tolerância de arco
$13	0	bool	Reportar em inches
$20	0	bool	Limites da maquinagem
$21	0	bool	Limites da máquina
$22	0	bool	Ativar homing
$23	1	mask	Inversão do sentido de homing
$24	50	mm/min.	Feed rate do homing
$25	635	mm/min.	Feed rate de procurar limite
$26	250	ms	Tempo e fechar o circuito de fim de curso
$27	1	mm	Recuo após fim de curso
Parâmetro $x = val	Valor padrão	Unidade	Definição
$100	800	Step/mm	Steps por milímetro em x
$101	800	Step/mm	Steps por milímetro em y
$102	800	Step/mm	Steps por milímetro em z

Tabela 6 - Parâmetros do Grbl. (cont.)

Parâmetro $x = val	Valor padrão	Unidade	Definição
$110	635	mm/min	Distância máxima por unidade de tempo em x
$111	635	mm/min	Distância máxima por unidade de tempo em y
$112	635	mm/min	Distância máxima por unidade de tempo em z
$120	50	mm/s^2	Aceleração em x
$121	50	mm/s^2	Aceleração em y
$122	50	mm/s^2	Aceleração em z
$130	225	mm	Máximo deslocamento de X
$131	125	mm	Máximo deslocamento de Y
$132	170	mm	Máximo deslocamento de Z

As variáveis "**Mask**" têm os seus valores nas tabelas 7,8 e 9.

Tabela 7 - "Mask" para os valores de inversão de step e direção do movimento, $2 e $3 (Breiler, 2020a).

0	00000000	N	N	N
1	00000001	Y	N	N
2	00000010	N	Y	N
3	00000011	Y	Y	N
4	00000100	N	N	Y
5	00000101	Y	N	Y
6	00000110	N	Y	Y
7	00000111	Y	Y	Y

Tabela 8 - "Mask" para o relatório de estado do Grbl, $10 (Breiler, 2020a).

Report Type	Value
Machine Position	1
Work Position	2
Planner Buffer	4
RX Buffer	8
Limit Pins	16

Tabela 7 - "Mask" para os valores de direção de homing, $23 (Breiler, 2020b).

Homing direction	Value
X+ Y+ Z+	0
X- Y+ Z+	1
X+ Y- Z+	2
X- Y- Z+	3
X+ Y+ Z-	4
X- Y+ Z-	5
X+ Y- Z-	6
X- Y- Z-	7

Apêndice VIII. Artigo: “Study, Design and Development of a Man Machine Interface for a CNC”

Study, Design and Development of a Man Machine Interface for a CNC

Francisca Vigo[1*]

[1] Department of Mechanical Engineering, University of Minho, Campus de Azurém, Av. da Universidade, 4800-058 Guimarães, Portugal
[*] Corresponding author, e-mail: franciscavigo@gmail.com

Abstract

This work focuses on the design of a Man-Machine interface for the CNC of a company named "Padrão Ortopédico" easily handled by any technician (without any type of training in industrial machines).

The CNC machine has two main axes, Z and X, with Z being the cutting tool axis (horizontal) and X being the vertical movement axis. The low density polyurethane block is placed on a rotary plate along the secondary axis A of the CNC and the control of the machine is promoted through the "Grbl Controller", a software inserted in a portable computer connected to the machine's Arduino. For the design of the graphical interface, the previously used software served as a basis for defining the necessary functions to be implemented, such as "Sending code and machining", "Manual manipulation of the axes", "Changing the settings and parameters of the Machine" and "Communication Man-Machine". The functions were programmed in the Python3 language, using the "Tkinter" and "Pyserial" libraries as primary resource. The program was finally implemented on a Raspberry Pi4 (model b) with a seven-inch touchscreen.

Keywords

CNC, Interface, Grbl, Python, RaspberryPi, Tkinter

1 Introduction

CNC milling machines are a recurring tool in the major prosthetic and orthotic industries, and it is a constant need in this medium to produce positive molds for users' members. The machine developed at the company "Padrão Ortopédico" is equipped with two main and one secondary axis systems, with no capacity to be controlled at the root but through a portable computer with open-source software.

The lack of expertise of the company's technicians in machinery and industrial machinery is an obstacle for them when using complex programs to send code to the CNC, which could put at risk the veracity of the obtained piece, the state of the machine or the technician himself.

By basing the new interface on previous software, without the need to purchase an existing controller on the market, which claims to purchase mechanical systems compatible with it, it is possible to comply with one of the objectives for the company, the low cost design. Also, it makes it unnecessary to replace parts already purchased.

This work is not focused on the study of the machine and its controller, but on the possibility of communicating with it without the need to send codes that are not intuitively visible to an inexperienced operator.

2. Objectives

Generically, this project focuses on the design of a Human-Machine Interface for the CNC of the company Padrão Ortopédico easily handled by any technician of the company (without any type of training in industrial machines). Since, initially, there is no information about the machine itself, it is necessary to acquire physical data (mechanical and electronic) and data in the scope of the software used to translate and control the CNC's movements. After collecting this information, it's possible to characterize its limitations and the necessary specifications for the idealization of the interface.

3 State of art

The CNC system consists of three main units: the numerical control unit (CN) that promotes the Human-Machine Interface responsible for position control, the motor unit and the driver unit. The CNC is generally treated only as the Numerical Control unit, which can be divided into three large groups of components: the MMI, the PLC and the NCK [1]. It is known that computer-assisted numerical control machines, namely CNC machines, are used for a wide variety of production. Depending on their complexity and the precision sought by the operator, these machines are capable of creating a wide range of parts [2].

Grbl is a high-performance, low-cost alternative that aims to control movement on CNC milling machines. This firmware is designed to be used in any type of Arduino controller on the market [3]. The program accepts G code according to universal standards and has been tested with

the output of several CAM tools, with no anomalies. Arcs, circles and helical movement are fully supported, as are all other primary G code commands. Macro, variable functions and most canned cycles are not supported, but possible to run on GUI platforms. This firmware has an open-source interface called Grbl Controller, showed on figure 1. This software is the basis of the interface that will be created [3].

4 CNC machine

The computer-assisted numerical control machine found at the company "Padrão Ortopédico" was initially built to revolutionize the creation of helmet molds for patients with plagiocephaly. This machine works by removing material (milling machine) where a block is allocated to the rotating plate moved on axis A which, along its path, undergoes material removal by the cutter (cutting tool) implemented parallel to the Z axis (cutting tool plane). In turn, the cutter has greater or lesser depth or longitudinal movement depending on the coordinates given by the G code, or by manual command, to the mechanical system of the Z axis, or X, respectively.

4.1 CNC operation system

In terms of the flow of information (figure 1), the input is always generated on the man-machine interface (MMI), which, in this case, is external to all the other systems of the device, since a computer with the "Grbl Controller" software is used to manage parameters, send codes and tasks, among others. The input is sent to the Grbl firmware found on the CNC's Arduino controller.

The firmware stores and handles the information received in the Arduino controller (PLC) buffer.

After processing the information, i.e., the translation of the language, it's sent to the drivers of the motors that transform this information into electrical pulses that are successively sent to the motors and then transformed into mechanical movements (output) of the tool (vertical or horizontal movement identified with yellow arrows in figure 1) or in the rotational movement of the plate where the block is located. The milling rotation is an external output of the system because the spindle is not connected to the Arduino controller, having a separate manual value adjustment interface. The machine's MMI system includes an external computer that is connected to the machine's Arduino controller whenever it is necessary to handle the machine and that, during the entire operation, must remain connected.

The connection for passing information between the drivers and the Arduino controller in the circuit is carried out in a cabled way. Pins for motor's direction, limit and pulse, are directly connected to the Arduino controller (through the proto board) and are defined in table 1.

Table 1 - Pinout Grbl firmware Arduino board 2560.

Function		Pin
Limit	Z axis	53
	X axis	51
	A axis	50
Direction	Z axis	37
	X axis	35
	A axis	34
Step pulse	Z axis	22
	X axis	24
	A axis	25

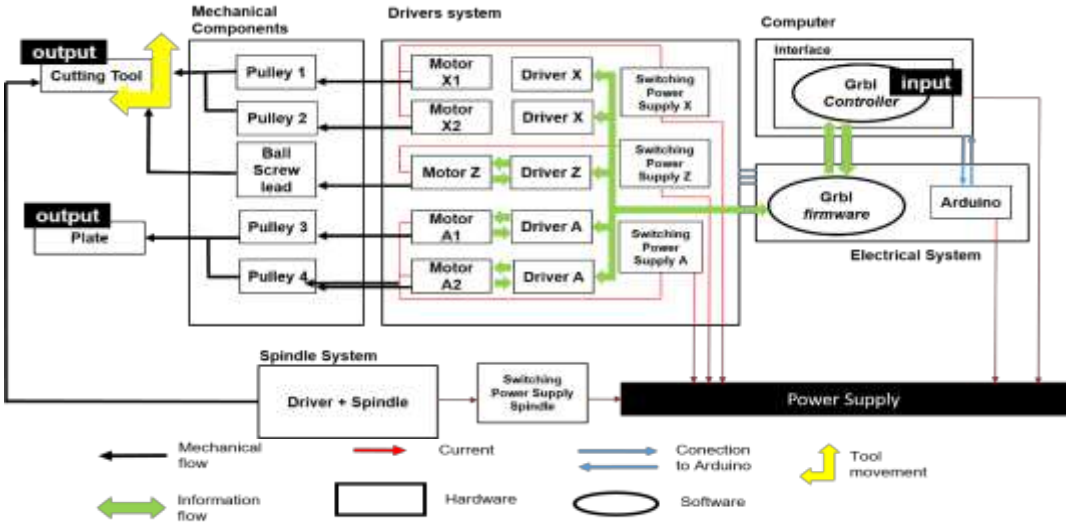


Figure 1. Information flow of the CNC machine.

5 Human-Machine Interface

This interface named “CNController” was developed in order to give the user the possibility of using the machine in a more ergonomic way, without the need for a computer external to the CNC. One of the main advantages of creating this application was the ease handling of it.

5.1 Program Functions

The design of the display was inspired by the information layout of the “Grbl Controller” computer interface. This controller has only one menu that allows the machine to be manipulated according to the options:

- Identification of the port where the machine's Arduino controller is connected;
- Option to close the door;
- Reset Grbl;
- Choice of the file and visual representation of the path file;
- Button to start sending the contents of the file to Grbl;
- Stop button;
- Progress bar for sending the file to Grbl;
- Machine working time;
- Grbl communication panel;
- Window for entering commands to send to Grbl (sent via the “Enter” key on a keyboard);
- Coordinate monitors (X, Y, Z) for the movement of the axes;
- Manual control panel for three axes;
- Button to choose the movement of the axes manually;
- Spindle activation button;
- Button to access the Grbl settings.

Dividing the types of functions, as shown in figure 2, there would then be four main groups, these being “Code sending and Machining”, “Manual manipulation of the axes”, “Alteration of the Machine's settings and parameters” and “Communication Machine man". This distribution facilitates the grouping of objects when the visual organization of the interface.

5.2 The interface architecture

As the different types of functions have already been distributed, it is important to divide them visually as well. In this way, the operator is more easily able to find what he needs. By reducing the amount of information that the user sees at a time, the program becomes less confusing and simpler, if organized correctly. Despite the interface it’s almost completely programmed, throughout the project and examination, more functions and needs are always found in its use, which are subsequently implemented.

The “Main Menu” shown in figure 3, includes all the tasks that the operator has to perform at the beginning and when machining, the buttons that promote the unlocking of the channel and the status information of the channel are found in the first line. In the second, the operations to choose and send the file to the CNC. In the third, a button to soft reset the firmware and two buttons to stop and continue machining after stopping it.

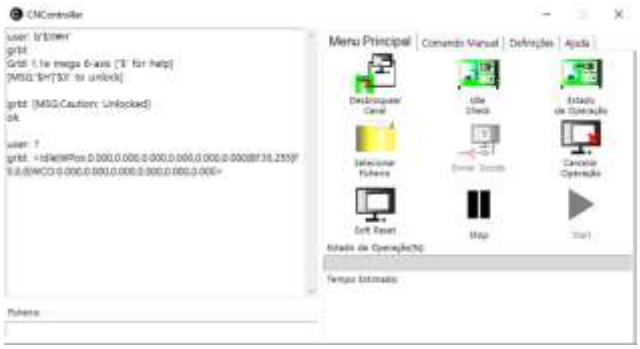


Figure 3. "Main Menu" of the GUI interface.

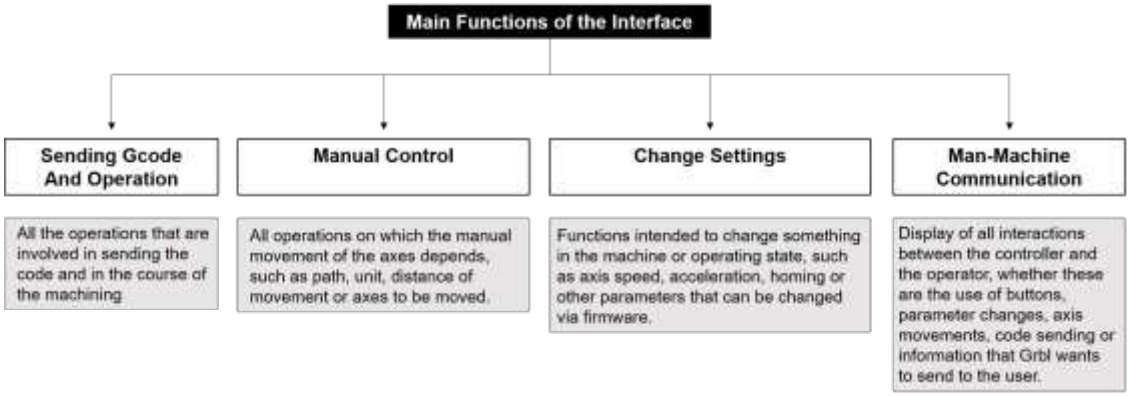


Figure 2. General functions to be implemented in the interface.

In this case, the button to unlock the channel and status button were clicked. The message sent and the response received can be seen on the communication panel.

The next section, “Manual Control”, shown in figure 4, has six buttons to move the three axes of the machine positively and negatively according to the defined pattern. It is possible to choose the type of coordinates that are intended for the movement, choose the part zeros and also set the Z axis machine zero.

As you can see in figure 4, the user pressed the button for relative coordinates and moved three millimeters of the positive X axis. Then, he performed the X0 and Z0 of the part at the point where it was.

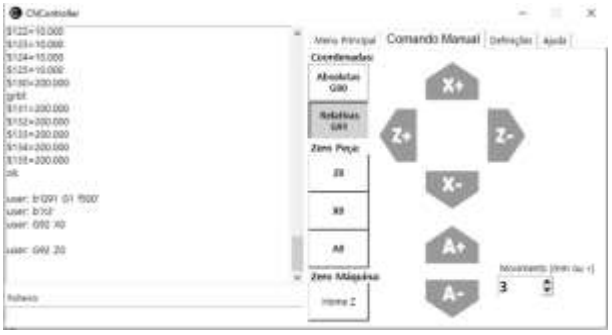


Figure 5. "Manual Control" of interface.

In figure 5, the third section of the interface “Settings” is exposed. Here, it is possible to change the parameters related to the firmware, by selecting the parameter to be changed and the desired value for it.

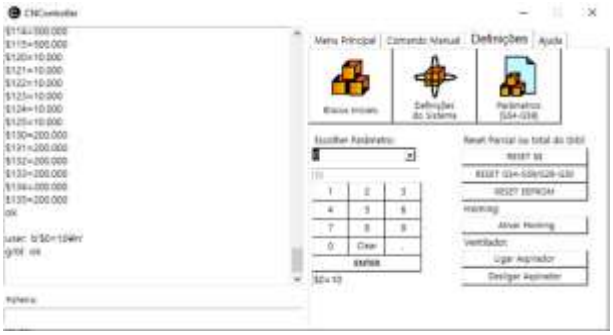


Figure 6. Interface "Settings" menu.

Finally, the last part of the program is a “Help” menu, represented in figure 6, which has three buttons to assist the operator with regard to errors and alarms in the firmware, definition of what the interface buttons do and definition of each Grbl parameter.

In this case, the user clicked on the “Errors and Alarms” button and a window appeared in the place of the communication panel, where all the errors and alarms that the firmware can send and their meaning are described.

Figure 4. Interface "Help" menu.

5.3 Programming the Interface

The interface developed for the CNC machine was entirely written in python, an open source object programming language, namely Python 3 [4].

For a better approach and implementation of the necessary libraries, a Python IDE was also used, i.e., a predefined code editing program, “PyCharm”, and a virtual environment generated by the “Anaconda 3” compiler. As the basis of the program, the library “Tkinter” was used, a library that provides objects with visual context.

An MVC structure was adopted for the organization of the code: "" Model "-" View "-" Control "". This structure, simplified in figure 7, is generally used in programs to make easier for the programmer to see the relationship between different types of variables [5].

The “Model” includes all numerical or string variables in the program. This part of the program has the function of registering and saving the items desired by the user [6].

The “View” (Viewer) includes all the visible objects of the program, i.e., all the code that will originate something that the user sees through a screen [6].

Finally, the “Control” (controller) directs the variables between the “Model” and the “View”, so that they take a path without getting lost [6].

In summary, in the GUI created, there are then four classes, three MVC scripts and a main script that executes the program. The variables and functions used are described in figure 7, simulated UML schema in the Pycharm Professional 2020 IDE [7].

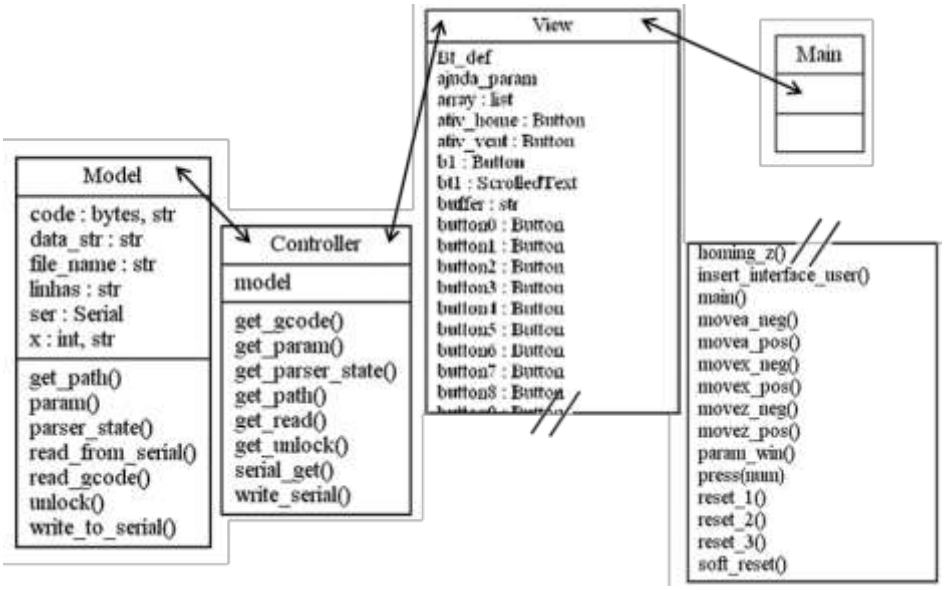


Figure 7 - Information flow in the MVC program.

5.3.1 Processing of variables – Model

The information (or data) has a cyclical movement in the program. The "Model" processes the data that enter the system by assigning values to the variables that, in turn, are sent to the "Control" that forwards them to the "View". In the “View”, the variables are transformed into objects visible by the operator so that, in this way, he can change or validate them. If the operator enters new data into the program, they will be redirected back to the “View” and will go through the previous route again. [6].

The “Model” is divided into variables and variable processing functions. Figure 8 shows the dependencies of each of them.

It’s in this class that is programmed the serial channel that promotes the interface connection with the machine's Grbl firmware and the possibility of choosing the G code file to be machined.

Briefly, variables with null values are created in the definition of the beginning of the class and the functions are used to give values to these variables to later be sent to the “control” and be processed.

Like all “Class” modules, it is necessary to state the initial variables that will start instantly with the program running, according to the values assigned to them. In this script, three general python3 libraries were used, represented in the code excerpt from figure 9 [8].

```
"from Tkinter import filedialog
import serial
import time
        "
```

Figure 9 - Model libraries.

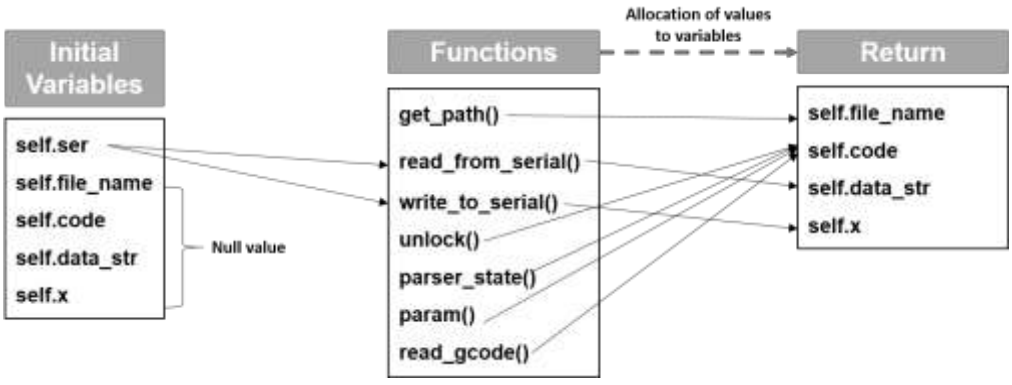


Figure 8 - Data flow in Model script.

As shown in figure 10, a first variable “self.ser” is defined, whose function is to connect the serial channel between the device where the program is located and another device. The expression “serial.Serial” refers to the call to the “Serial” function (channel connection) in the imported “serial” library. One of the variables used in this function, the “port”, is defined by the “COM8” port, the channel where the data cable for the Arduino controller containing Grbl is connected. This name varies depending on the hardware port that is connected to and the device.

```
“class Model:
    def __init__(self):
        self.ser = serial.Serial('COM8',
115200, timeout=None, xonxoff=True,
stopbits=1, bytesize=8)
”
```

Figure 10 - Definition of serial channel.

The initial variables “self.file_name = ””, “self.code = ””, “self.data_str = ””” and “self.x = ””” are used in the functions within the “Model” class, being evidenced right from the start, so that no information processing errors occur. In this way, it is possible to link these variables exclusively to the “Model” class, becoming local variables. The use of the quotation mark (“) character designates a null value, that is, depending on whether the variable is numeric or string.

The function "def get_path (self)", which returns the variable "self.file_name", is used to save the directory of the chosen file and, later, open, process and send the file to Grbl. The function that defines the variable, named “filedialog.askopenfilename”, is a function of the “filedialog” library, “askopen-filename”, responsible for popping up an operating system window for the user to search for the file he wants (Foundation, 2021). The data flow in this function, represented in figure 11, is responsible for specifying the location of the disk where the user wants to search for the file. In this case, "/", makes the open window the disk in general, and the operator can search in any compartment of the disk. The user can change this directory to any specific folder in his computer's memory [4].

Finally, the function “def read_gcode (self)”, represented in figure 12, probably the most important function of the interface. This function opens the selected G code file, “open (self.file_name,‘ r ’)”, where the character “r” invokes it with permission to read [9]. A “flushInput” of the channel is created so that the actual memory of the Arduino controller is not occupied, thus, once the data is read by the Arduino controller, it is deleted from its buffer [9].

The file is read line by line and transforms each line into the “l_block” variable. It’s indicated to wait for Grbl's “ok” response to change the value of this variable and send the next line.

This function not only sends the code but also constantly checks whether the buffer status allows one more line to be sent.

If any type of error occurs in the reading, it will be read by the code and presented on the communication panel.

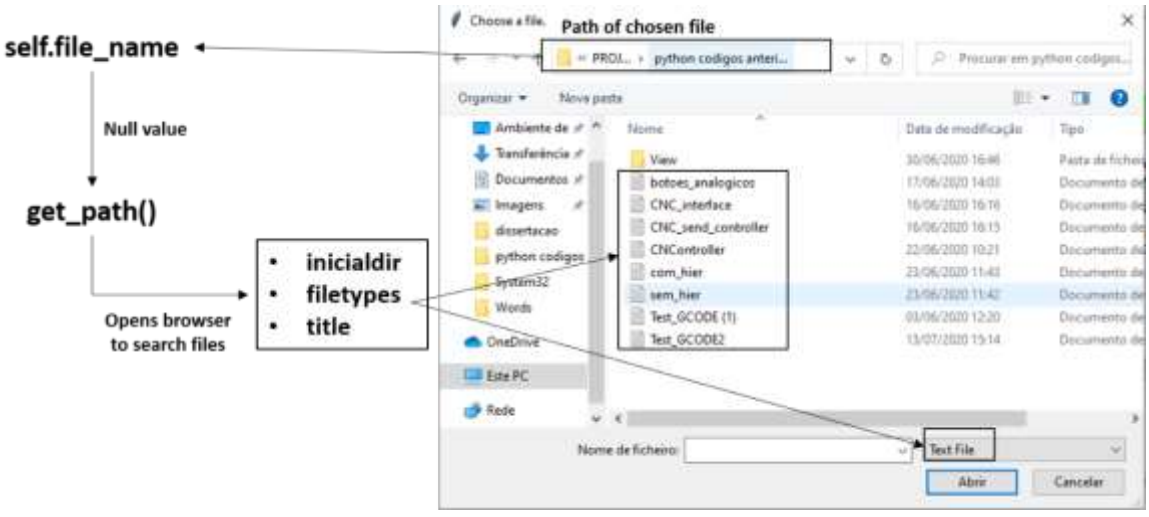


Figure 11 "Askopenfile" function to search and select file.

```
“def read_gcode(self):
    self.response=False
    with open(self.file_name, 'r') as f:
            # Wait for grbl to initialize and
flush startup text in serial input
        time.sleep(5)
        self.serial.flushInput()
        self.reading = 'Streaming Gcode...' […]
self.talk =
self.serial.write(self.resulta.encode())  #
Send g-code block to grbl[…]
while 1:
    self.rbl_out =
self.serial.readline().strip().decode()  # Wait
for grbl
[…]”
```

Figure 12 - Read the “gcode” file and send to Arduino function.

5.3.2 Sending variables – Control

As previously specified, “Control” is like a channel between “Model” and “View”. In this case, as the program is written in Python, an object language, there would be no need to have this class to intermediate the information. But, in order to match the model, MVC, the "Control" was inserted as a means of returning the variables developed in the "Model" to the "View". For this purpose, only the “Model” class, from the “model” file, as a library (“from model import “Model””) is imported.

This class, unlike the others, contains only one initial variable, since these functions are only intended to return values when called by the user.

The initial variable “self.model” defines the “Model” class so that the values previously returned in the “Model” can be used in the “Controller” class.

Note that, to use the “Model” variables, it is necessary to use the generated path “self.model”. The variables are returned, by the functions of the “Controller” class “serial_get”, “get_path”, “get_read”, “write_serial”, “get_unlock”, “get_parser_state”, “get_param” and “get_gcode”, thus being able to be used in the “View” class after importing the “Control”.

5.3.3 View variables – View

The “View” script includes everything that is possible for the user to view in the interface, from the program window, to the buttons, bars, images, text boxes or even the letters themselves.

As already mentioned, the library on which this program is based, called “Tkinter”, promotes the use of specific functions and codes to create different types of visible objects, each with two different functions [4]. Figure 13 shows the libraries used for writing the “View” script.

Figure 14 shows the first four fundamental lines of code for the interface. “Self.root = Tk ()” promotes the creation of a window through the “Tkinter” library, which is considered to be the master of any other underlying one [10].

```
“from Tkinter import *
from Tkinter import ttk
import threading
import time
import Tkinter.scrolledtext as tkst
from Tkinter.ttk import Progressbar
from controller import Controller
from error_list import lista_erros”
```

Figure 13 - View libraries.

```
“self.root = Tk()
self.root.title(""CNController"")
self.root.geometry("800x400")
self.root.resizable(width=False, height=False)
self.root.config(bg='white')”
```

Figure 14 - Creating Tkinter's window.

5.3.4 Compile Program – Main

To finalize the granting of the code, a class is held exclusively to promote the running of the program. This class, "Main", admits only the class "" View "" as a function to be processed, as expressed in the code excerpt of figure 15.

The line “#! / Usr / bin / python3” ensures that the version of python installed that is being used to run the program is Python3.

Initially everything (“*”) from the “view” script is imported to be able to process the “View” class (with access to all previously imported libraries and files).

```
“#!/usr/bin/python3
from view import *

if __name__ == "__main__":
    “View”()”
```

Figure 15 - Main script code.

The function “if __name__ ==" __main__ "” is used as a way to run the program. The "__name__" is an internal variable that evaluates the name of the current module. However, if a module is being executed directly (from the command line), then “__name__” is defined as the string

"__main__". That is, when executing this code, the variables will take equal values and, therefore, the "View" () "function will be executed. Thus opening the intended interface [12].

5.4 Testing

To test the GUI, the block's zeros were marked and a gcode file was sent through the serial channel.

The program added the code and performed a correct function, waiting for the firmware to respond to avoid overloading the Arduino's buffer. In figure 16 a description of how the information was showed in the interface is showed.

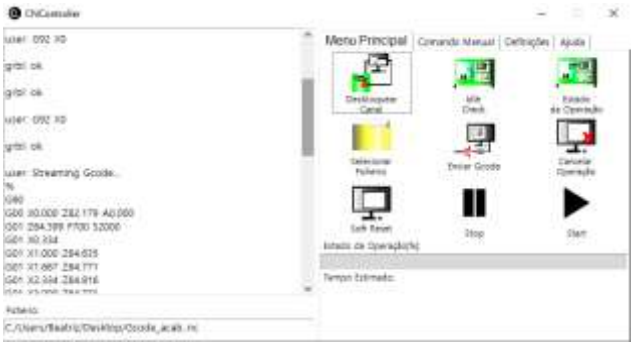


Figure 16 - Sending Gcode file via serial channel.

6 Conclusions

With the conclusion of this project, it is possible to give the objectives initially proposed as long. In the end, a graphic interface was inserted in the CNC of the company Padrão Ortopédico, capable of controlling the machine without the aid of external objects and devices. It should be noted that, despite the main objectives met, there is still a long way to go when it comes to machine automation.

6.1 Program Outcome

At a more central level of the project, in the insertion and programming of the graphical man-machine interface, it's possible, happily, to say that the program was concluded, as two of the company's technicians responsible for using the machine, both without any training on industrial machines, have already operating the CNC from the created "CNController" interface, which guarantees that the main objective of the program, creating a graphical and intuitive interface for the user, has been fulfilled.

6.2 Faced Problems

The main difficulties were the study of the functioning of the existing firmware on the machine and the interconnection of the different Python libraries so that the program would not only send and receive data, but also make it possible to visualize the same data without crashing.

7 References

[1] Suh, S.-H. et al. (2008) Theory and Design of CNC Systems - (Springer series in advanced manufacturing). doi: 10.1007/978-1-84800-336-1.

[2] Smid, P. (2003) CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming. Available at: https://books.google.pt/books?hl=pt-PT&lr=&id=JNnQ8r5merMC&oi=fnd&pg=PA1&dq=CNC+definitions&ots=P-MJOU6QAQ&sig=2-t6YYKrvhq9K0_EBsiVKg4uba4&redir_esc=y#v=onepage&q&f=false.

[3] Breiler, J. (2020) 'Grbl Settings', 1. Available at: https://github.com/gnea/grbl/wiki/Grbl-v1.1-Configuration.

[4] Rodas de Paz, A. (no date) Tkinter GUI Application Development Cookbook_ A practical solution to your GUI development problems with Python and Tkinter. Packt Publishing.

[5] Sung, K., Shirley, P., & Baer, S. (2013). The Model-View-Controller Architecture. *Essentials of Interactive Computer Graphics*, *Mvc*, 131–168. https://doi.org/10.1201/b15723-12

[6] Pop, D. and Altar, A. (2014) 'Designing an MVC Model for Rapid Web Application Development', Procedia Engineering. Elsevier B.V., 69, pp. 1172–1179. doi: 10.1016/j.proeng.2014.03.106.

[7] Wild, T. (2015). *PyCharm as a Python IDE for Generating UML Diagrams*. https://waterprogramming.wordpress.com/2015/07/29/pycharm-as-a-python-ide-for-generating-uml-diagrams/

[8] Shino, E. (2019). *Classes in Python*. https://enrijetashino.github.io/project/classes-in-python/

[9] Liechti, C. (n.d.). *pySerial API*. PySerial. https://pyserial.readthedocs.io/en/latest/pyserial_api.html

[10] Ferg, S. (2012). *Thinking in Tkinter*. 1–41.

[11] Foundation, P. S. (2021). *Tkinter Dialogs*. https://docs.python.org/3/library/dialog.html

[12] Fooz, M. (2009) What does if __name__ == "__main__": do? Available at: https://stackoverflow.com/questions/419163/what-does-if-name-main-do.