Sistema Robótico com Controlo de Força Para Operações de Furação

Diego Gabriel Gomes Rosa

Dissertação do MIEM

Orientador: Prof. Dr. Paulo Augusto Ferreira de Abreu

Co-orientador: Prof. Dr. António Manuel Ferreira Mendes Lopes

Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Mecânica

Julho de 2013

Dê-me uma alavanca e um ponto de apoio, e moverei o mundo.

Arquimedes.

Resumo

A utilização de sistemas robóticos para operações de furação tem-se mostrado útil, em especial quando é preciso realizar movimentos repetidamente e em superfícies que não se ajustam ao uso das tradicionais máquinas-ferramenta. Exemplos disso são as aplicações de furação em fuselagem de aviões e o uso de robôs em cirurgias. Sabe-se também que tais operações podem ser aperfeiçoadas com a utilização de um sistema controlado em força. Entretanto, é conhecido que isso não é habitualmente realizado, ainda que existam algoritmos de controlo de força já disponíveis nos controladores de muitos robôs industriais presentes no mercado.

Portanto, com o intuito de conhecer melhor essa tecnologia e suas funcionalidades, serão realizados neste trabalho alguns ensaios a partir de diferentes estratégias de controlo e condições de furação. A partir disso, será feita uma análise dos resultados obtidos, onde serão conferidas as vantagens e as limitações de executar-se furos com robôs através de cada uma das estratégias utilizadas.

Por fim, será mostrado que uma das estratégias de controlo de força usada é útil quando se precisa evitar danos na peça maquinada por conta do surgimento de valores muito altos de força durante a furação. Porém, quando se necessita apenas de uma grande precisão na profundidade do furo, o controlo em posição continua a ser a melhor opção. E, caso seja preciso realizar controlo de força com maior precisão de posicionamento, este trabalho deixará também algumas propostas de como isso pode ser feito.

Abstract

The use of robotic systems in drilling operations have been showing its advantages, especially when it is necessary to perform repeatable movements and movements in surfaces that are not adapted well to the traditional machine tools. In addition, there are many examples about its use, as the drilling in aircraft fuselages or the robots use in surgeries. It is also known that the application of a force control system can improve this type of operation. However, it is identified that it is rarely done, even though there are force control algorithms embedded in many industrial robot controllers presents in the market.

Therefore, with a view to know better this technology and its functionalities, will be made in this project some trials with different control strategies and drilling conditions and then a detailed examination of the obtained results. After, will be possible to analyze the advantages and the limitations in drilling operations using this kind of control strategies.

In the end, it will be shown that one of the used control strategies is useful when it's necessary to avoid damages in the workpiece because of high force values executed during the drilling. However, when it's necessary only a good accuracy in the drill's depth, the position control continue being the best option. And, in case of use of force control added to position accuracy, this work will propose some ways about how to do it.

Agradecimentos

Primeiramente agradeço aos professores Paulo Abreu e António Mendes Lopes pelo acompanhamento, dedicação e paciência concedidos durante a realização do trabalho, o que foi essencial para sua realização.

Agradeço também ao professor Francisco Freitas, responsável pela unidade curricular Dissertação de automação, pelo esforço em auxiliar-nos e acompanhar-nos sempre, garantindo assim a evolução e conclusão do projeto.

Ainda, meus agradecimentos ao professor António Torres Marques, diretor do Departamento de Engenharia Mecânica, que colaborou com o fornecimento de materiais.

Aos professores Andersan dos Santos Paula, Diomar César Lobão e José Flávio Feiteira, da Universidade Federal Fluminense (Brasil), que colaboraram diretamente neste período de estudos e em que estive a realizar a dissertação.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), minha gratidão pela oportunidade e pelo apoio financeiro para concluir este período de estudos na Universidade do Porto.

À Universidade do Porto, pelo acolhimento e pela constante colaboração em todo o período em que estive em Portugal

E, obviamente, meus agradecimentos a Deus, à minha família e a todos os amigos, com os quais pude contar em todos os momentos de minha vida e, em especial, durante a realização deste trabalho. Sem eles certamente não conseguiria chegar ao fim.

Índice

Lista de figuras

Lista de gráficos

Lista de tabelas

Capítulo 1

Introdução e Objetivos

A robótica é uma área da automação que tem obtido grande destaque nas últimas décadas. E, mesmo em tempos de crise, o mercado dos robôs industriais manteve sempre um considerável nível de vendas, conforme mostrado na figura 1.1. Tudo isso advém de um simples facto: a robótica é solução viável e simples para inúmeras aplicações. Pintura industrial, soldadura, paletização, corte, montagem ou inspeção são alguns exemplos de aplicações de um robô industrial.

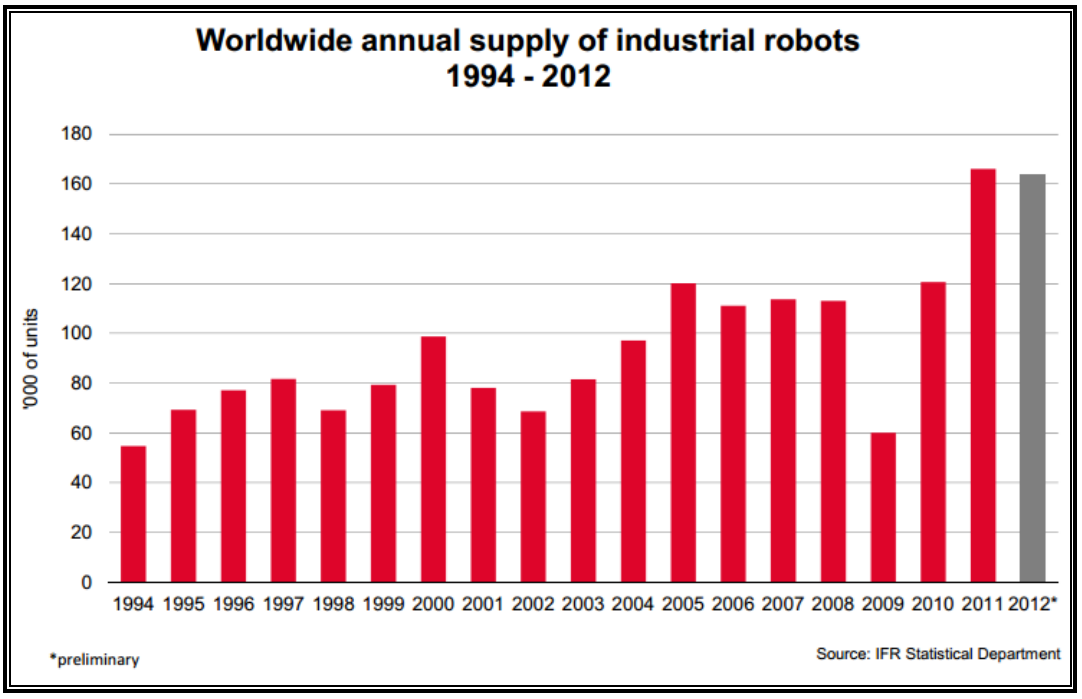

Figura 1.1 - Número de robôs vendidos nos últimos anos

Todavia, altos gastos na aquisição e na manutenção de um robô são fatores que ponderam qualquer decisão. Além disso, robôs são equipamentos limitados para determinados fins, como transportes de cargas muito elevadas. Portanto, eles só são realmente utilizados onde são necessários e fiáveis. São também uma alternativa quando se precisa fazer trabalhos que exigiriam muito do ser humano (utilização de força, alta repetibilidade e excesso de tempo dedicado a uma única atividade, exposição à riscos, etc.) ou trabalhos que simplesmente se tornam mais fáceis e/ou econômicos com sua presença.

Outro ponto de interessante análise refere-se à expansão da tecnologia nos mais diversos tipos de indústria. Primeiramente pela versatilidade de um sistema robótico, visto que um único robô poder ser programado de diversas formas, para diferentes aplicações e em indústrias distintas. Além disso, outras qualidades são atraentes, como a elevada repetibilidade de movimentos, a fácil disposição no espaço, a facilidade de realizar programações básicas e a interação do controlador com outros equipamentos.

Logo, todos estes fatores combinados contribuem para o sucesso da robótica nos dias atuais. E, com uma maior utilização dos robôs, é também esperado que surjam novos desafios quanto às aplicações aos engenheiros e investigadores relacionados a esta área. Um desses desafios será explorado nesta dissertação, que é justamente acerca do uso de robôs na furação.

1.1 - Operações de Furação

A furação é uma operação essencial atualmente, além de ser um dos processos de maquinagem mais utilizados na indústria. Para se compreender melhor essa questão, basta observar o ambiente ao nosso redor. Frigoríficos, aviões, carros, computadores ou simples óculos (vide figura 1.2), quase tudo possui furos em sua construção.

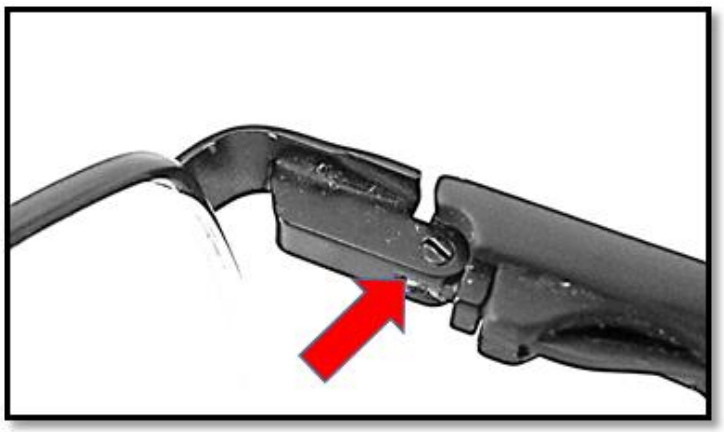

Figura 1.2 - Furo na articulação da armação de uns óculos

Porém, apesar de muito utilizada atualmente, o avanço desta tecnologia remonta a séculos atrás. Por exemplo, os primeiros registros sobre furação de metais foram apontados a partir dos anos 1800 [1]. E neste intervalo de tempo muitos avanços foram realizados, tanto no âmbito teórico quanto no experimental. Afinal, já há muito tempo sabe-se que um simples furo é de importância tal que, se mal realizado, pode comprometer permanentemente a qualidade final da peça maquinada. E ainda mais, uma peça dessas, dependendo de seu uso, pode originar riscos à integridade física das pessoas que a utilizam.

Dessa forma, fica evidente que é preciso sempre investir em processos e tecnologias que visem melhorar a qualidade do furo e também diminuir custos de processo. Um exemplo relacionado a isso é dado por Stone & Krishnamurthy [2], que mostra que até poucos anos atrás, na indústria aeronáutica (cujo exemplo de aplicação pode ser visto na figura 1.3), 60% das rejeições de peças ocorriam justamente por erros nos processos de furação. E mais, como este processo é geralmente realizado no final da produção, não se perde apenas as peças, mas todo o trabalho que nelas já foi executado anteriormente. Ou seja, estas são perdas de elevado valor financeiro. Contudo, estes números não transmitem somente a informação sobre prejuízos, mas principalmente a necessidade de aprimoramento e inovação nos processos industriais.

Figura 1.3 - Furação em fuselagem de aeronave

Outro exemplo que imprime a necessidade de desenvolvimento do processo de furação está ligado à delaminação, que é a principal causa de falhas estruturais em compósitos [3]. E como afirma Durão et al [4], uma furação mal realizada neste tipo de material pode gerar esse problema e, potencialmente, ocasionar a perda prematura das estruturas quanto estas estiverem em serviço. Ainda de acordo com o autor, a delaminação nem sempre é visível na superfície, o que torna o processo muito mais complexo. Assim, novamente o processo de furação envolve a segurança e a integridade física humana, bem como o aspecto financeiro.

Portanto, soluções são necessárias para os mais diversos problemas ligados à furação. Porém, quando se pensa em uma solução, deve-se avaliar os impactos positivos e negativos que esta possui em cada caso. Por exemplo, muitas vezes o tempo de realização de uma furação pode ser mais importante que a precisão dimensional de um furo. Em outros casos, é preciso dar prioridade para o formato da peça que se deseja furar. Isso significa que algumas decisões devem ser tomadas antes de uma escolha ser concluída. Neste trabalho, a solução inclui a utilização de um robô, que oferece inicialmente grandes vantagens e algumas limitações, como veremos a seguir.

1.2 - Furação com Robôs

O uso de robôs em processos de furação é bem recente e ainda gera muita desconfiança. E certamente há motivos para isso. Como comenta Robert Isaksson [5], a baixa rigidez presente nos manipuladores robóticos pode permitir que haja imprecisão no posicionamento durante a furação e consequente propagação de danos no interior do furo.

Portanto, comparando a rigidez de um robô com a de máquinas ferramentas tradicionais, seria óbvio chegar à conclusão que robôs são impróprios para furação. Entretanto, como cita Gürsel Alici em seu artigo [6], este é um problema que pode ser contornado através do uso de determinadas estratégias de controlo de força, algumas das quais serão abordadas mais adiante neste trabalho.

E, ainda quanto às características do robô para a furação, é preciso dizer que este apresenta muitas vantagens interessantes. A primeira dessas está ligada à sua alta flexibilidade. Um robô de seis eixos, por exemplo, está em posição de superioridade quando comparado com soluções tradicionais.

Na figura 1.4 observa-se um exemplo da flexibilidade de um robô quando comparado com uma máquina linear, como a da figura 1.5, em operações de maquinagem.

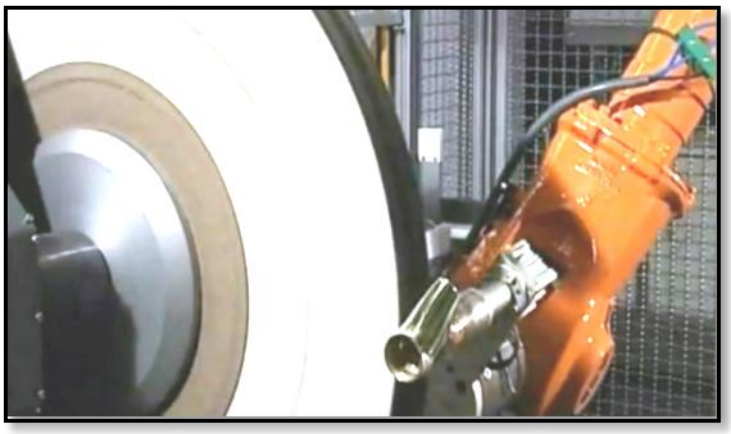

Figura 1.4 - Robô para polimento de material sanitário

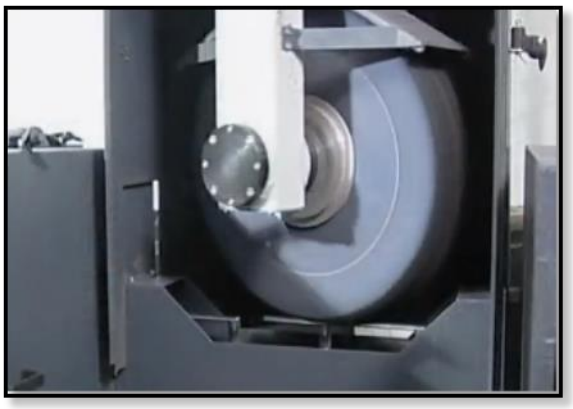

Figura 1.5 - Politriz linear automática

Portanto, se em um primeiro instante o pensamento seria desistir da utilização de um robô, após uma avaliação mais consistente chega-se à conclusão de que, na verdade, é preciso investir em alguma solução para os problemas levantados anteriormente.

E mais, além da questão das vantagens e desvantagens existentes no robô ao compará-lo com outros tipos de máquinas específicas para maquinagem, há operações em que o robô deixa de ser opção e torna-se necessidade. Um exemplo interessante é a furação de fuselagem de aviões. Como nos mostra a revista *Manufacturing Engineers* na edição de Março de 2013 [7], a flexibilidade inerente aos manipuladores robóticos apresenta um grande potencial de aplicação. Isso ocorre pelo facto de uma máquina-ferramenta usualmente não ser capaz de mover-se de forma a realizar furos em diferentes posições e orientações, de forma ágil e com alguma garantia de qualidade. Esta é uma particularidade dos robôs.

Ainda há outra promissora aplicação para robôs, que é a utilização em cirurgias, mais especificamente, cirurgias ortopédicas [8]. Obviamente, neste caso, o robô não é pré-programado para realizar operações automaticamente, mas para auxiliar o cirurgião na perfuração de ossos. E mais, há aqui um novo detalhe, que é o facto de ossos serem materiais

heterogêneos. Assim, a diferença de densidade entre camadas de perfuração podem ser consideradas um desafio extra a enfrentar, bem como o problema da rigidez descrito anteriormente.

E, finalmente, quanto à solução do problema da rigidez, existem propostas de soluções. Uma primeira, mais simples, poderia passar por restringir o uso de robôs à furação apenas de objetos macios e em que a precisão não fosse tão crítica. Mas, com o desenvolvimento de robôs que são capazes de alcançar pontos com grande precisão e de exercer forças cada vez maiores, uma segunda proposta passa pela a utilização do controlo de força.

A opção de controlar o robô em força não só para a furação, mas qualquer operação que envolva maquinagem, tem levado os fabricantes a desenvolverem funcionalidades específicas para tal uso. Um exemplo é o *RobotWare Machine Force Control*, integrado no controlador IRC 5 (Industrial Robot Controller), do fabricante ABB. O controlador, que pode ser visto na figura1.6, possui a capacidade de controlar o robô em força através de duas diferentes estratégias de controlo. Obviamente, outras soluções têm surgido e surgirão nos próximos anos.

Figura 1.6 - Controlador IRC 5, ABB

1.3 - O Controlo de Força

O controlo de força é uma possível solução a ser utilizada no comando de um robô. Esta solução consiste basicamente em fornecer, através de uma estratégia de controlo específica, a capacidade de o robô “sentir” as forças externas que o mesmo exerce sobre alguma superfície.

Porém, é óbvio que um algoritmo de controlo não é capaz de alterar as limitações mecânicas existentes em um robô. Ou seja, quando controlado em força, a rigidez do robô permanece a mesma. Porém, como nos afirma novamente Isaksson [5], se mantemos uma força constante sobre a superfície maquinada, isso evita, por exemplo, o risco de deslizamento (ainda que de pequenas dimensões) sobre a peça. Isso já seria suficiente para garantir uma melhor qualidade ao produto.

Além disso, ao furar-se com controlo de força, o manipulador robótico é capaz de reconhecer e alterar as forças (ou binários) aplicadas durante todo o movimento, a fim de evitar danos não só na superfície, mas também no interior da peça.

Outra vantagem da aplicação do controlo de força é que se pode, através dela, aumentar a vida útil das ferramentas ou de outros componentes usados na furação. Afinal, como se trata de uma operação mecânica, o desgaste sempre existe e sempre pode ser diminuído, se utilizadas as forças corretas ou aconselháveis pelos fabricantes durante o processo.

Portanto, se na ausência de controlo de força a resposta do robô ao encontrar um obstáculo (que pode ser um defeito interno ao material) poderia aumentar significativamente a força de contacto resultante do facto de o robô tentar manter a velocidade especificada durante a programação, com o controlo de força a situação é bem diferente. É evidente que, para uma superfície simples ou um material homogêneo e não muito duro, o resultado da furação controlada em trajetória e velocidade pode ser satisfatório. Porém, na realidade, os materiais e superfícies encontrados não serão sempre adequados para o uso de um robô. Dessa forma, para que se possa ter um sistema mais flexível, a solução citada torna-se atraente.

E quanto à implementação de um sistema robótico dotado de controlo de força, isso é realizado basicamente através da inserção de uma célula de carga no braço do robô ou sob a peça a ser maquinada. Além disso, é necessária a utilização de um algoritmo de controlo específico. Ou seja, pouco muda na prática. Todavia, os resultados podem ser potencialmente bem diferentes, conforme tentaremos mostrar neste trabalho.

1.4 - Objetivos do Trabalho

Conforme foi visto até aqui, as operações de furação que utilizam um robô controlado em força apresentam um considerável benefício para diversas aplicações industriais. Entretanto, para tal feito, seria necessário desenvolver um algoritmo de controlo de força para um controlador. Mas há uma segunda opção: utilizar algum algoritmo já presente em controladores existentes no mercado.

De facto, se há uma aplicação específica em mente, o desenvolvimento de um algoritmo particular para essa operação pode ser proveitoso. Mas isso exige tempo e custos. Por outro lado, a segunda solução seria interessante. Inicialmente, por não ser necessário desenvolver

um novo algoritmo de controlo, pois a solução já está disponível no controlador. E, quanto ao facto de ser uma solução simplificada para operações diversas, o que seria uma limitação, é na verdade uma possível vantagem, pelo facto de ter uma ampla gama de utilizações.

Portanto, tendo em vista a utilização da segunda solução apresentada, é possível estabelecer os objetivos do trabalho. Estes podem ser resumidos em:

- ✓ Estudo de um sistema de furação baseado em um robô industrial cujo controlador seja provido de algoritmos específicos para controlo de força;
- ✓ Implementação de uma célula robótica para furação, utilizando diferentes estratégias de controlo;
- ✓ Posterior avaliação do sistema através de ensaios com diversos materiais, tipos de furo, velocidades de furação e análise de forças exercidas.

Proposta de Trabalho

Para alcançar tais objetivos, é preciso fornecer uma proposta de trabalho. Esta consiste primeiramente em um estudo detalhado das soluções hoje existentes para os problemas citados. Depois, é necessário definir que recursos são utilizados, a começar pelo robô utilizado no trabalho, o ABB IRB 2400/16, presente no laboratório de Robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto. Também é necessário recorrer ao *software* RobotStudio para a programação *offline* do robô. O programa é fornecido pela mesma empresa, ABB. Outros acessórios, como *spindle* e célula de carga, são imprescindíveis para a execução do trabalho.

Com os recursos disponíveis, é possível iniciar a implementação da célula robótica (virtual), que consiste em uma reprodução, em *software*, da célula robótica real. Este processo é também executado no *software* RobotStudio e gera um resultado como o que observa-se na figura 1.7.

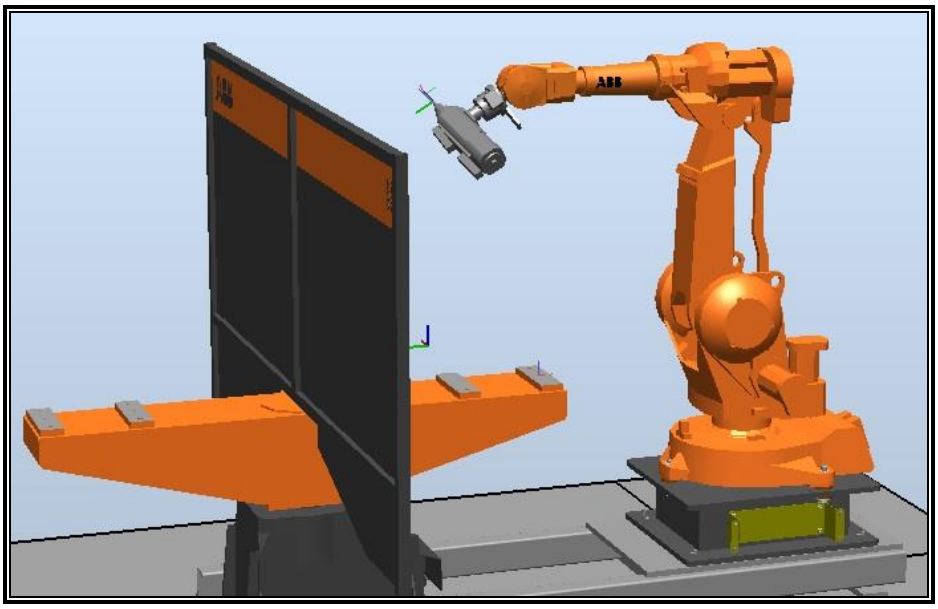

Figura 1.7 - Célula robótica criada no software ABB RobotStudio

Com a célula robótica pronta, é possível iniciar a programação do robô, utilizando o *software* RobotStudio novamente. O primeiro programa a ser produzido é algo mais simples, com o robô sendo controlado apenas em trajetória e velocidade. Isso por duas razões. Primeiramente porque assim é possível ter-se uma noção geral do processo. Além disso, a célula controlada apenas em posição e velocidade se torna necessária para futuras comparações com o robô controlado em força.

Após esta etapa, será realizada a programação do robô recorrendo a controlo de força, a partir de duas estratégias de controlo disponíveis no programa do controlador. Este programa, chamado RobotWare, é fornecido pela empresa ABB. E as estratégias de controlo de força utilizadas serão o controlo de força *Speed Change* e *Force Pressure*.

Controlo de Força *Speed Change*:

O *RobotWare Force Control Speed Change* é um produto da família de programas desenvolvidos para controladores da empresa ABB. É recomendado quando a precisão é importante e o resultado final deve possuir dimensões bem específicas, de acordo com o manual do fornecedor [9].

Quanto ao seu uso, o *FC Speed Change* baseia-se em realizar uma redução de velocidade de maquinagem quando um determinado valor de força é atingido, a fim de que as forças sobre a peça diminuam e sua qualidade não seja comprometida.

Um exemplo de utilização é dado na figura 1.8. Assim que o robô encontra um obstáculo, a velocidade da trajetória é reduzida para um patamar mínimo. Ao ultrapassar o obstáculo, a velocidade é reestabelecida para a que foi programada.

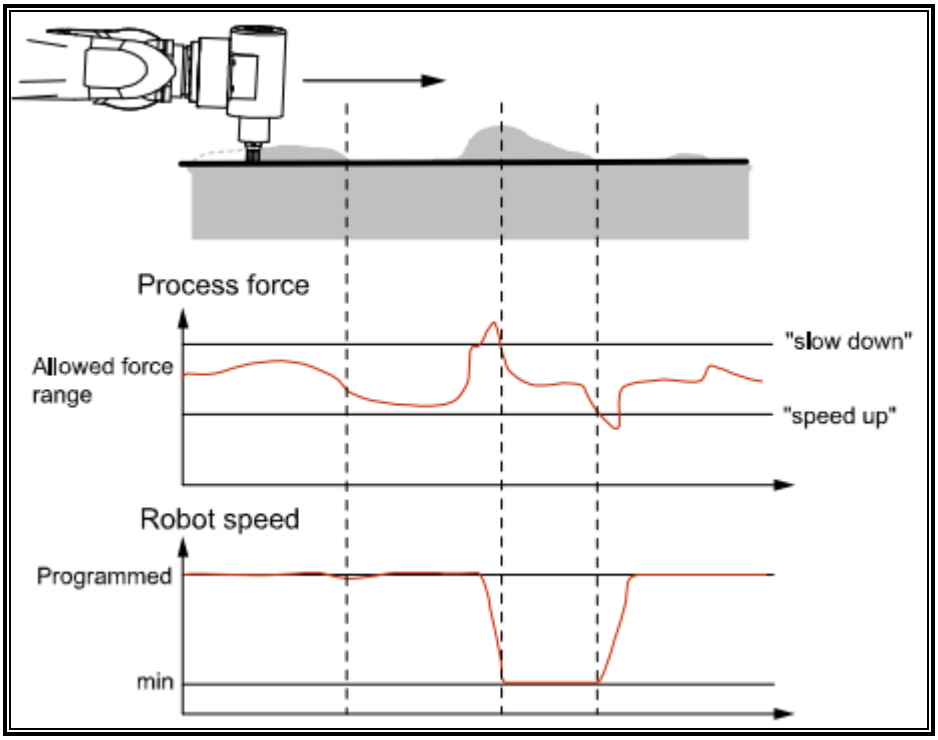

Figura 1.8 - Utilização do FC Speed Change

Uma informação relevante é que o programa permite escolher não apenas dois níveis de velocidade, mas quantos forem necessários. No entanto, a ABB recomenda que o uso de mais de quatro patamares de velocidade deve ser utilizado com reserva, visto que o tempo de resposta do sistema diminuirá.

Controlo de Força *Force Pressure*:

Da mesma forma que o *Force Control Speed Change*, o *Force Control Pressure* é uma solução da empresa ABB para problemas de maquinagem e tem como objetivo tornar um robô sensível à forças de contacto. Dessa forma, o sistema robótico tende a manter uma força constante de contacto sobre uma superfície, mesmo que as posições sejam desconhecidas [9].

O *FC Pressure* baseia-se em uma programação de trajetória bem próximo à superfície que se deseja maquinar. Quando a ferramenta do robô se aproxima da superfície, o controlo de força é ativado e o robô inicia uma procura à superfície. Ao encontrá-la, mantém uma força de contacto até o final da trajetória programada. Ao fim, o robô deixa a superfície e o controlo de força é desativado. Um modelo básico dessa programação é dado na figura 1.9.

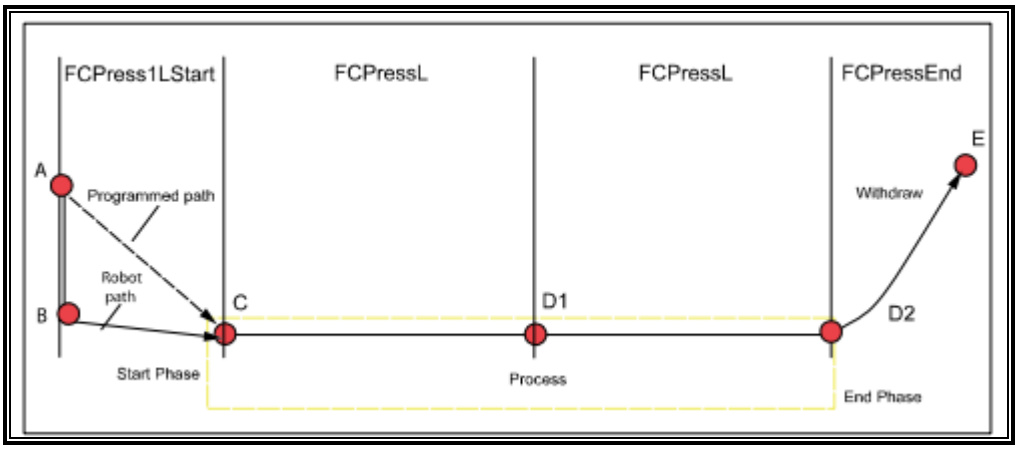

Figura 1.9 - Utilização do FC Pressure

Entretanto, é necessário advertir que essa programação contida na figura 1.9 é apenas uma dentre diversas possibilidades existentes. Contudo, isto é explorado com mais detalhes no Capítulo 2 deste relatório.

E, para complementar a proposta de trabalho, após a programação são realizados testes para verificar vantagens e desvantagens de se utilizar tais estratégias de controlo. Isto é apresentado no Capítulo 3.

Por fim, no Capítulo 4 são apresentadas as conclusões do trabalho e fornecidas algumas sugestões de trabalhos futuros.

1.5 - Recursos

Nesta seção são detalhados os recursos empregues para a realização do trabalho. Estes são divididos em dois tópicos. No primeiro são apresentados as aplicações relacionadas com uma programação do robô e com a aquisição de dados para análise posterior. Na segunda parte são descritos os componentes físicos utilizados durante os ensaios.

1.5.1 – *Software*

i. ABB RobotStudio:

O RobotStudio, cujo ecrã inicial é visto na figura 1.10, é um *software* produzido pelo fabricante ABB e cuja finalidade é a programação *offline* de seus robôs.

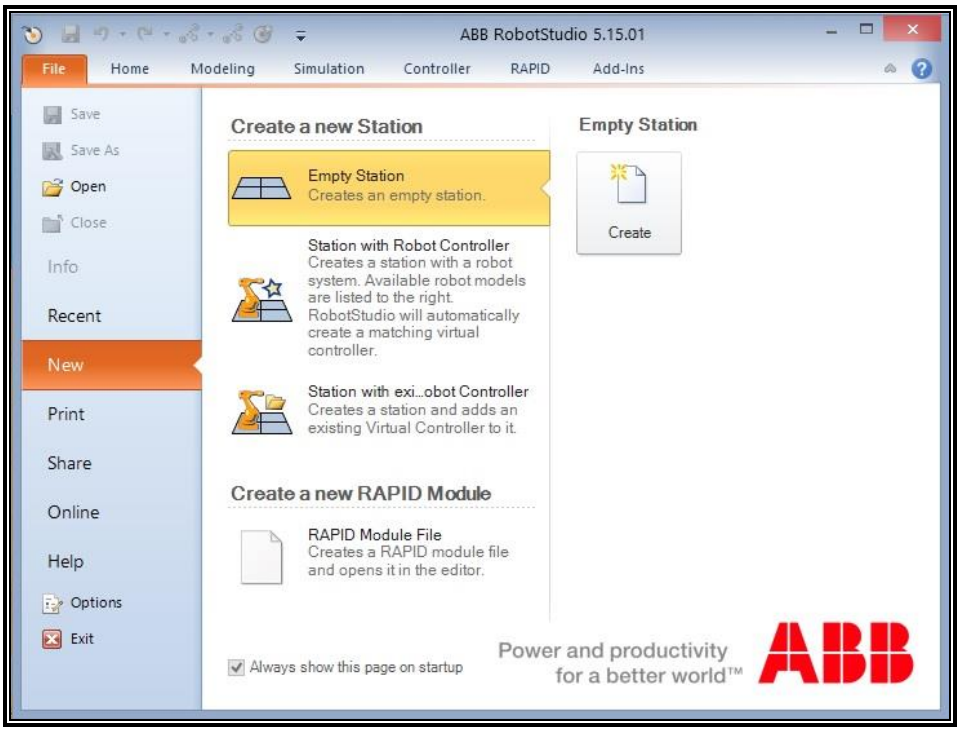

Figura 1.10 - Vista do Ecrã Inicial do RobotStudio

Partindo de uma estação de trabalho vazia ou a partir de um sistema robótico já existente, é possível criar um ambiente virtual integrado não só pelo robô, como também por diversos equipamentos que podem trabalhar junto a ele [10] (figura 1.11).

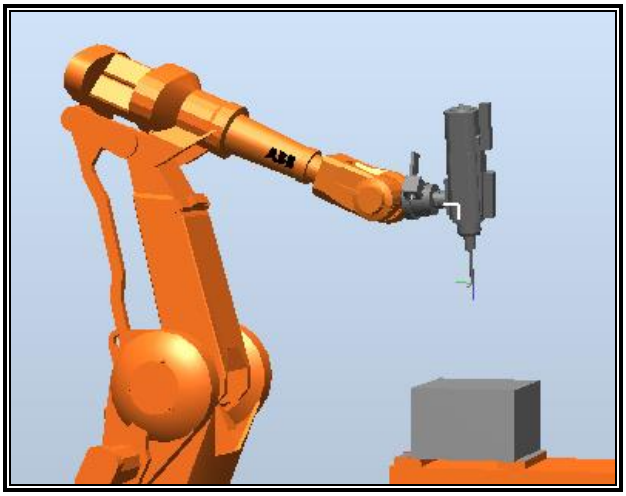

Figura 1.11 - Sistema Robótico no RobotStudio

Tal *software* permite também a verificação de instruções fornecidas ao robô, a simulação destes movimentos e a geração de um código em linguagem RAPID (figura 1.12), que pode ser descarregado diretamente para o controlador do robô.

```
TRAP Write_data_to_array
    cont:=cont+1;
    time{cont}:=ClkRead(clock10);
    ForceVector:=FCGetForce(\WObj:=WOBJFURO);
    tcp_pos{cont}:=CPos(\Tool:=Spindle_tool\WObj:=WOBJFURO);
    xforce{cont}:=ForceVector.xforce;
    yforce{cont}:=ForceVector.yforce;
    zforce{cont}:=ForceVector.zforce;
    xtorque{cont}:=ForceVector.xtorque;
    ytorque{cont}:=ForceVector.ytorque;
    ztorque{cont}:=ForceVector.ztorque;

ENDTRAP
```

Figura 1.12 - Exemplo de um código RAPID

A linguagem RAPID é a linguagem de programação utilizada exclusivamente em robôs da empresa ABB.

Este programa oferece ainda outras vantagens interessantes. Uma delas é devido à posse de uma interface que possibilita ao programador a opção de escrever os códigos diretamente na linguagem do robô ou utilizar a navegação por botões e barras para depois sincronizar o robô com um controlador virtual. Após a sincronização um código RAPID é automaticamente gerado, sem exigir um grande esforço da parte do programador.

Também é interessante o facto de poder-se modelar algumas peças de formato simples dentro do próprio programa. No caso de peças mais complexas, há a opção de importar arquivos de programas de desenho CAD, como o *SolidWorks*. No caso, basta que o desenho esteja salvo no formato “.step”.

Outra vantagem presente no RobotStudio é o chamado Virtual FlexPendant, que nada mais é que uma versão virtual da consola real do robô. Ou seja, é uma reprodução da programação *online* contida no *software* de programação *offline* e que permite ainda a realização de *jogging* com o robô, a geração de gráficos relativos ao movimento ou mesmo a programação do robô diretamente pelo *RobotWare Machining Force Control*, como podemos ver na figura 1.13, o que será útil na realização do trabalho.

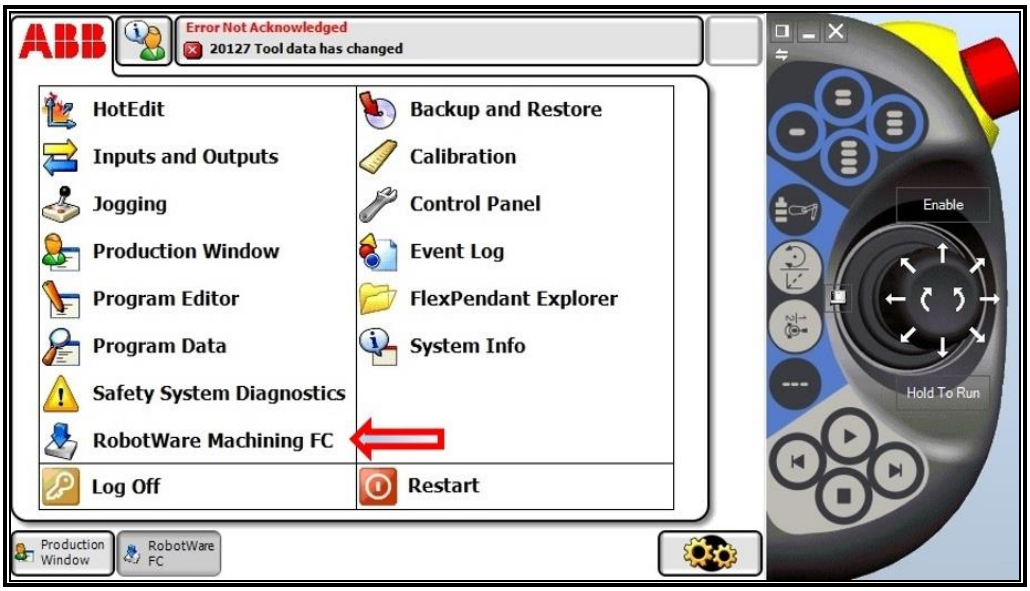

Figura 1.13 - Virtual FlexPendant

Por fim, uma última funcionalidade que será amplamente utilizada no trabalho é referente à aquisição de dados. Através da ativação de algumas funções internas é possível ao controlador obter informações do robô, como posição, velocidade, força ou tempo de execução do programa, e gravá-los em ficheiro ".txt" logo em seguida, para posterior análise dos sinais.

Entretanto, como qualquer *software*, possui limitações. Uma dessas é relativa justamente à frequência de aquisição de dados. Esta é restrita e recolhe dados somente a cada décimo de segundo, o que é pouco para um sistema robótico. Outra pequena limitação, mas não tão prejudicial, é relativa ao uso quando o robô é controlado em força (*FC Pressure*). O programa não possui um emulador interno, ou seja, sempre que se tenta realizar uma simulação, o programa retorna um erro por não encontrar uma força.

Outro ponto que limita o *software* a ser utilizado em computadores mais antigos (muitas vezes presentes nas indústrias) é relativo aos requisitos mínimos do sistema. É preciso utilizar o Microsoft Windows XP SP3 ou mais recente, CPU de 2 GHz ou mais veloz, memória de 3 GB (para sistemas de 32 bit) ou 8 GB (sistemas de 64 bit), um espaço disponível de pelo menos 5 GB em disco e placa gráfica compatível com o DirectX 9 ou OpenGL.

Entretanto, mesmo com essas limitações, o RobotStudio é um programa com muitas funcionalidades e facilidades. Por isso será utilizado no trabalho, apesar da existência de outros programas genéricos que poderiam cumprir parte do que o RobotStudio realiza. As versões utilizadas no trabalho foram o RobotStudio 5.15.00.01, RobotStudio 5.15.01 e RobotStudio 5.15.02, sendo que foram alterados à medida que a empresa divulgava uma nova atualização do produto.

ii. ABB Test Signal Viewer:

Outra solução da ABB que visa colaborar com sistemas controlados em força é o ABB Test Signal Viewer (figura 1.14). Este é um programa que está ligado em tempo real à aquisição de dados do transdutor de força, ou seja, permite a leitura e a aquisição de sinais de força e binário. E, comparando a aquisição de dados deste programa com a do RobotStudio, o ABB Test Signal Viewer apresenta três grandes vantagens.

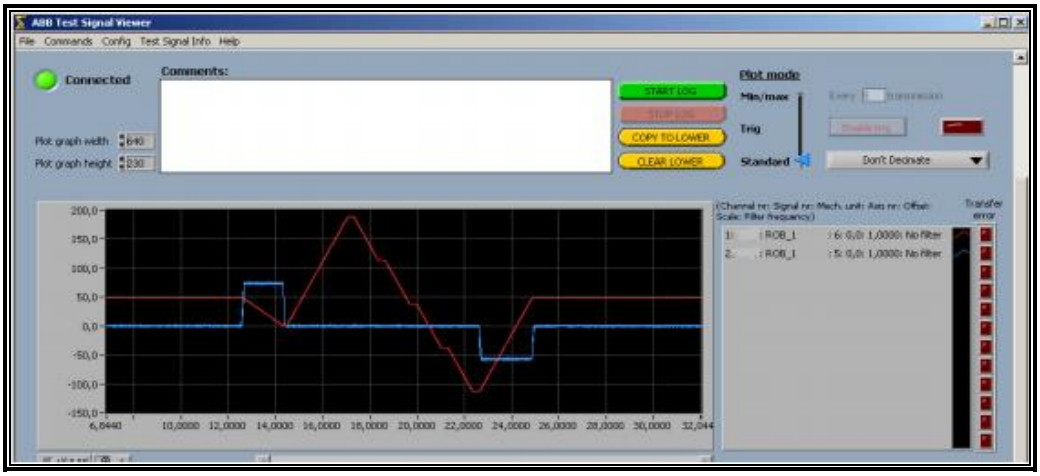

Figura 1.14 - ABB Test Signal Viewer

A primeira vantagem reside no facto de o programa ter uma *interface*, apesar de simples, muito amigável com o usuário. Assim, tanto programação quanto leitura e armazenamento de dados se realizam sem grandes dificuldades. E mais, há duas formas de se gravar os dados gerados pelo programa, o que o torna muito flexível. Uma dessas formas é em arquivos ".txt", o que permite que os dados sejam lidos e analisados posteriormente por outros programas (a figura 1.15 mostra um gráfico gerado em Excel com dados obtidos no ABB Test Signal Viewer). A segunda forma é em arquivo ".dat", que somente pode ser aberto pelo próprio ABB Test Signal Viewer mas permite que os gráficos originalmente gerados sejam novamente observados.

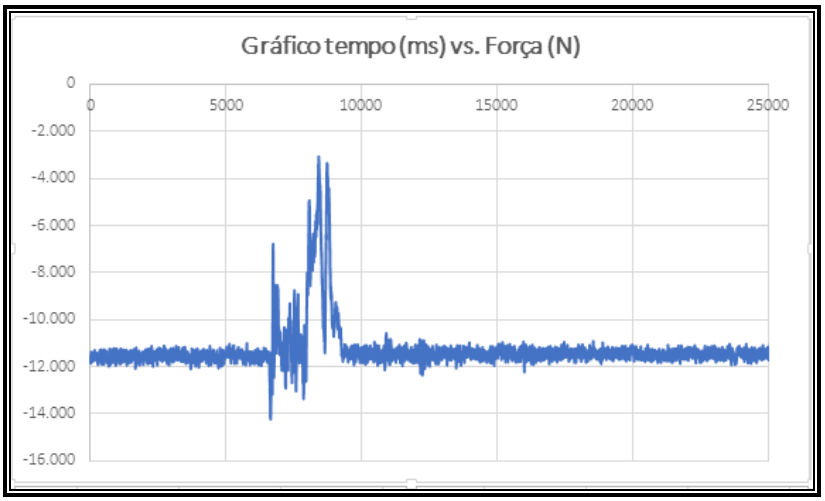

Figura 1.15 - Gráfico gerado no Excel com dados do ABB Test Signal Viewer

Além disso, a aquisição de dados é realizada com maior frequência, sendo possível configurar o programa para adquirir dados até o intervalo de 0,504 milissegundos [11]. Este facto constitui a segunda vantagem deste programa sobre o ABB RobotStudio. Portanto, ao analisar sinais de força, que normalmente apresentam certa dispersão nos valores (ruído), há uma melhor visibilidade devido à maior frequência de leituras.

Por fim, a terceira vantagem está na possibilidade de utilização de filtros *Low Pass*.

Entretanto, mesmo com tantas vantagens, não será possível utilizar o programa em plenitude de recursos. O programa permite apenas a leitura dos sinais do sensor de força expressos no referencial de base do sensor, que por razões de configuração mecânica, não é coincidente com o referencial de aplicação de forças ao realizar a furação. Obviamente, seria possível determinar uma matriz de transformação para alterar os valores salvos pelo programa para o referencial adequado. Porém, é possível programar em linguagem RAPID a leitura das forças de contacto no referencial adequado, pelo que será esta a opção utilizada para aquisição das forças de contacto.

iii. MATLAB:

O MATLAB (*Matrix Laboratory*) é um programa da empresa MathWorks que possui como objetivo principal a manipulação de variáveis numéricas, sejam elas contínuas ou discretas. Neste trabalho, será utilizado para manipulação e visualização dos dados adquiridos pelo controlador durante o movimento do robô. Será também útil posteriormente na geração de gráficos que ajudem na compreensão dos ensaios. Sua *interface* gráfica é mostrada na figura 1.16. Nela é possível observar basicamente três páginas. Ao fundo, a página principal do programa. À direita apresenta-se o algoritmo executado e à esquerda está um gráfico gerado.

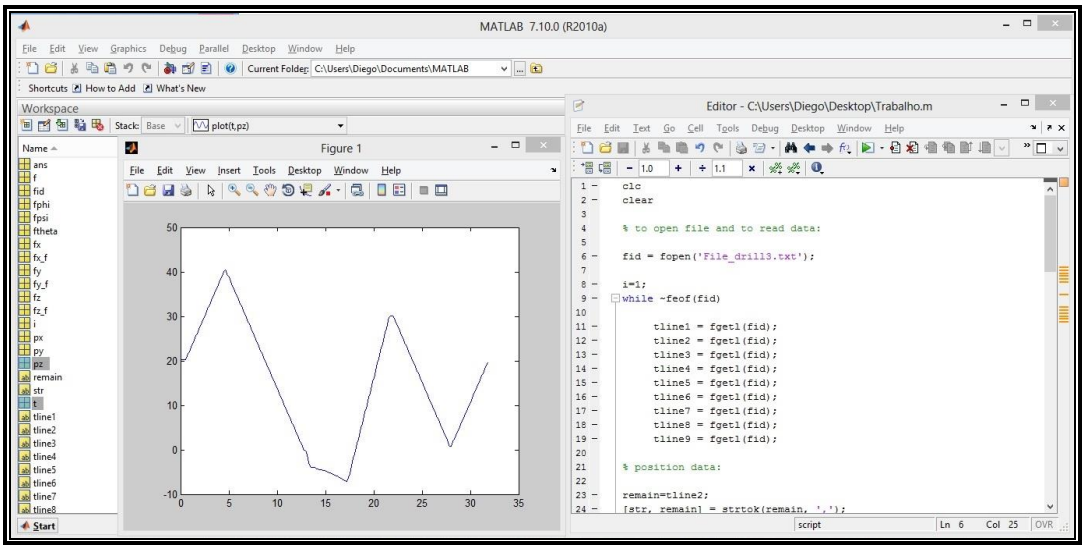

Figura 1.16 - Interface Gráfica do MATLAB

Um grande proveito em utilizá-lo nesta dissertação refere-se à sua facilidade de programação para leitura dos dados, visto que o controlador do robô gera um arquivo ".txt" com um formato que exige alteração da posição dos dados (em colunas) para geração de gráficos. Um exemplo de parte do ficheiro original gerado no controlador é mostrado na figura 1.17.

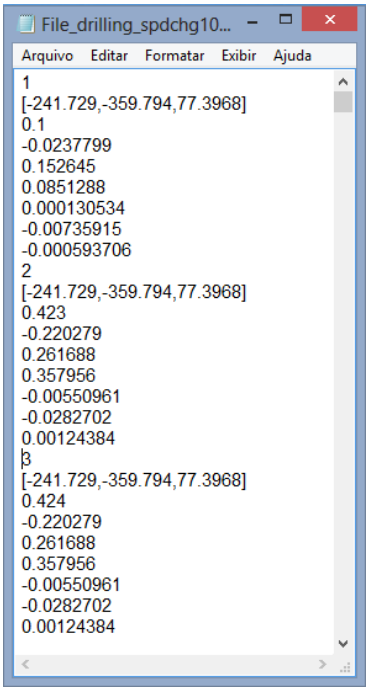

Figura 1.17 - Dados gerados durante a movimentação do robô

Obviamente, outros programas poderiam ser utilizados para a mesma finalidade neste trabalho, como o Microsoft Excel. Porém, o MATLAB apresenta maior agilidade no tratamento dos números e exibe algumas facilidades durante sua programação.

O algoritmo utilizado para leitura dos dados será melhor explorado no capítulo seguinte e encontra-se também no Anexo A desta dissertação. A versão do programa utilizada é a R2010a.

1.5.2 – Equipamentos e Materiais

i. Robô IRB 2400

O IRB 2400 é um robô da ABB para baixa carga. Considerado o robô mais popular de sua classe, conforme diz o fabricante [12], é uma boa opção para operações de soldadura a arco, corte, desbaste, polimento, manuseio de material e maquinagem em geral.

É fabricado em três modelos diferentes. O IRB 2400L é a versão que possui menor capacidade, 7 quilogramas, e maior alcance, 1,80 metros. Já as versões 2400/10 e 2400/16 (figura 1.18), ambas com alcance de 1,5 metros, possuem capacidades de manuseio de carga de 12 e 20 quilogramas, respetivamente. A terceira versão apresentada é a que será utilizada neste trabalho.

Figura 1.18 - Robô ABB IRB 2400/16

As características do ABB IRB 2400/16 que são imperativas para a realização trabalho são mostradas a seguir:

- ✓ Capacidade de carga: 20 quilogramas;
- ✓ Número de eixos do robô: 6. Isso permite uma maior diversidade de posições de furação;

- ✓ Repetibilidade do robô: 0,06 mm, o que garante uma tolerância mínima de posicionamento IT 7 (ISO) no interior dos furos[13];

Mais informações sobre o robô podem ser consultadas em seu *Data Sheet* [12].

ii. Spindle

O *spindle* XLC-070, produzido pela empresa PDS (*Precision Drive Systems*), é uma ferramenta essencial para furação quando se exige uma boa precisão nos resultados. Equipado com um motor de 2,2 kW, é capaz de realizar rotações máximas de até 24.000 RPM (ou 40.000 RPM, dependendo da frequência utilizada no controlador). O equipamento, que pode ser visto na figura 1.19, é fixado ao elemento terminal do robô pneumaticamente, através de um sistema de mudança automática de ferramenta.

Figura 1.19 - Spindle XLC-070

Além disso, é importante ressaltar que este equipamento possui algumas limitações para maquinagem. Portanto, o fabricante exibe no manual do produto [14] uma lista com os materiais em que a maquinagem não é possível, é possível se usada com precaução ou pode ser normalmente utilizada, conforme a tabela 1.1.

Titanium & Thermal Resistant	Foil	Not Suitable
Nickel Alloyed Steel	Sheets	Not Suitable
Stainless Steel (300 Series)	Sheets	Not Suitable
Stainless Steel (400 Series)	Sheets	Not Suitable
Carbon Alloyed Steel	Sheets	Not Suitable
Mild Steel	Sheets	Not Suitable
Stone, Granite & Marble	Blocks and Slabs	Not Suitable
Aluminum & Light Alloys	All	Use Caution
Polycarbonate Rigid Plastic	Sheets & Molded	Use Caution
Solid Hardwoods, Oak, Maple,	Sheets	Use Caution
Plywood & OSB	All	Normal Application
Solid Softwoods, Pine, Fir, Birch	Sheets	Normal Application
Flexible Plastic & PVC	All	Normal Application
MDF (Medium Density	Sheets	Normal Application
Particle Board	Sheets	Normal Application
Rigid Foam (Vitrified Plastic)	Blocks and Molded	Normal Application

Tabela 1.1 - Possíveis materiais para maquinagem com o spindle;

iii. Transdutor de força

O transdutor de força ATI Delta 330-30 (figura 1.20) é uma solução interessante quando necessário obter dados de força e binário simultaneamente. Com uma gama de leitura de forças que varia de ±330 N (eixos x e y) até os ±990 N (eixo z), bem como uma leitura de binários com valores de até ±30 Nm [15], este sensor de apenas 910 gramas consegue ser uma boa opção para leitura de dados de processo numa aplicação robótica.

Figura 1.20 - Sensor de Força ATI Delta 330-30

Além da ampla gama de valores de leitura, que são suficientes para a aplicação em questão, o transdutor possui uma resolução também compatível para furações. Para a captação de forças nos eixos x e y, a resolução é de 1/16 N. Para o eixo z, 1/8 N. E em relação aos binários, a resolução do sensor é de 5/1333 Nm.

Quanto à fixação do sensor ao robô, esta é feita mecanicamente, ou seja, ele está suficientemente rígido para garantir uma qualidade e precisão de leituras.

iv. Mesa posicionadora

Apesar de não ser utilizada no trabalho, a célula robótica utilizada possui uma mesa posicionadora giratória, a IRBP 500C. Esta é utilizada quando se necessita mover a peça durante a movimentação do robô. Com uma capacidade de 500 Kg e uma repetibilidade de 0,1 mm, é uma opção quando uma boa dinâmica é exigida no sistema. Ela pode ser vista na figura 1.21.

Figura 1.21 - Mesa posicionadora IRBP 500C

v. Materiais a serem furados

Após uma visão geral do que se possui em termos de *software* e hardware relativo ao robô, é necessário também determinar alguns materiais a serem ensaiados ao final da programação, para teste e validação de resultados.

Entretanto, a escolha não é algo relativamente simples. A proposta inicial seria realizar testes primeiramente em materiais macios, a fim de evitar possíveis danos ao robô. Somente depois seria realizada uma avaliação com algum material mais usual, como o alumínio. Por fim, seriam realizados testes complementares com materiais específicos, como algum tipo de compósito, algum material não homogêneo ou mesmo superfícies inclinadas.

Depois de uma avaliação mais aprimorada, que envolveu desde a simplificação dos testes (visto que um número alto de materiais envolveria um número muito elevado de testes) até a questão financeira (um material interessante a se furar seria o GLARE®, material atualmente usado na aviação, porém de custo elevado e baixa disponibilidade), definiu-se que serão utilizados os seguintes materiais:

Madeira:

O primeiro material escolhido para ser ensaiado é uma peça de madeira. A vantagem de se utilizar a madeira inicialmente deve-se ao facto de ser um material relativamente macio, se comparado à maioria dos metais, e que permite uma clara aquisição de dados. Outro material que poderia ser inicialmente utilizado é a espuma, por exemplo. Entretanto, ela exige valores de força muito baixos para ser furada, o que tornaria a análise um pouco mais complicada, visto que os valores poderiam ser facilmente confundidos com ruídos.

Portanto, foi escolhido utilizar uma viga de eucalipto. Esta possui caracteristicamente uma alta densidade e uma dureza classificada entre média e alta [16]. Além disso, possui boas propriedades mecânicas, como a resistência ao impacto, o que a torna sua utilização muito vulgar. Na figura 1.22 pode-se observar a madeira escolhida para iniciar a realização de testes no presente trabalho. Esta encontra-se já com alguns furos, resultado dos ensaios iniciais.

Figura 1.22 - Madeira utilizada nos ensaios de furação

Alumínio:

O alumínio foi o segundo material escolhido. Além de ser um material amplamente usado em aplicações mecânicas, é sempre sujeito à operações de maquinagem, como a furação. Outra característica do alumínio é que, dentre os metais, talvez seja o mais adequado para operações com robôs, visto que é relativamente macio. Portanto, para os testes, foi escolhido usar pequenas barras do metal (como mostra a figura 1.23*)*, de diâmetro 20 mm e cortadas no de comprimento 18 mm. Estas serão furadas transversalmente.

Figura 1.23 - Alumínio utilizado nos ensaios de furação

Compósito:

A escolha da utilização de um material compósito deve-se primordialmente a duas questões. Primeiro, ao facto de que o robô será sujeito não a apenas um tipo de resistência durante a furação, mas pelo menos duas diferentes, relativas aos materiais que constituem o compósito. Isso significa que é possível realizar uma análise mais minuciosa do controlo de força quando há variação de resistências mecânicas no interior de um material.

A segunda razão é devido à ampla utilização deste tipo de material atualmente, o que leva à necessidade de, por muitas vezes, preparar o material para diversos usos através de operações de furação. Porém, como já foi dito anteriormente, uma furação mal realizada pode danificar por completo a peça e impedir seu uso futuro. No caso do compósito, a situação é um pouco mais complicada, devido ao risco de surgimento de delaminação.

Portanto, tendo em vista a análise de furação com robôs controlados em força neste tipo de material, foi escolhido um compósito laminar reforçado com fibra de vidro, o qual observa-se na figura 1.24.

Figura 1.24 - Fibra de vidro a ser utilizada nos ensaios de furação

vi. Brocas

As brocas são essencialmente os equipamentos mais importantes na concretização de uma furação. Entretanto, esta não é a única forma de realizar um furo. Pode-se pensar em realizar furos através de forjamento, eletroerosão ou com uso de oxiacetileno. Mas é justamente a utilização da broca o método mais comum na indústria, devido à simplicidade, versatilidade e ao baixo custo envolvidos com este tipo de maquinagem.

Porém, para uma boa escolha das brocas, é necessário saber antes em que, basicamente, consiste uma broca. E a broca é, por definição, um elemento mecânico capaz de realizar dois movimentos simultâneos para a efetuação de um furo [17]. E, conforme verifica-se na figura 1.25, o primeiro (B) é o movimento de avanço, linear, da broca sobre a peça, na forma de esmagamento. Por isso é realizado com velocidades relativamente baixas (normalmente denotadas em m/min). O segundo (A) é o movimento principal de corte, que está ligado à rotação da broca e este é sempre um valor elevado (normalmente dado em RPM).

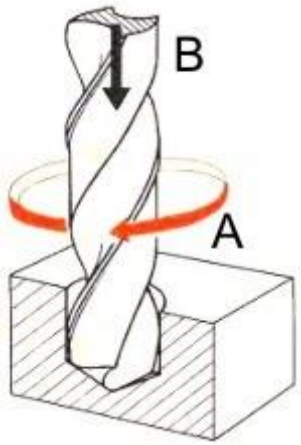

Figura 1.25 - Movimentos durante a furação

Por fim, é preciso escolher efetivamente quais serão as brocas escolhidas. A decisão foi tomada a partir de três pontos. São eles:

- ✓ Material a ser furado;
- ✓ Diâmetros de furação mais comuns;
- ✓ Diâmetros críticos para furação.

O primeiro critério é essencial para a escolha de brocas em qualquer situação, tanto para uso em robôs, através de um *spindle*, quanto para uso em outros equipamentos elétricos. Dessa forma, há basicamente dois tipos de brocas a serem utilizadas.

A primeira broca a ser escolhida é uma broca helicoidal em aço rápido (HSS-G) com encabadouro sextavado de 1/4”, da marca Bosch. As brocas helicoidais são as mais utilizadas em geral, mas essa possui alguns diferenciais, se comparadas a outros produtos existentes no mercado. Inicialmente por possuir uma afiação e ser construída de tal forma que é ideal para metais ferrosos, metais não ferrosos e materiais sintéticos duros ao mesmo tempo. Ou seja, pode ser utilizada neste trabalho tanto para furação de alumínio, quanto para furação da fibra de vidro.

Além disso, a broca Bosch HSS-G possui autocentragem, o que torna a pré-furação desnecessária quando o diâmetro do furo for igual ou inferior a 10 mm. E ainda, esta broca é capaz de fornecer uma tolerância de diâmetro H8 de acordo com a norma DIN 1412 C, ou seja, desvios máximos de apenas 22 µm no diâmetro do furo.

A segunda broca escolhida para a furação de madeira é também da marca Bosch. A broca em espiral, também em aço rápido (HSS) e com encabadouro sextavado de 1/4" possui ainda uma ponta centradora, que permite realizar furos mesmo em superfícies lisas.

As brocas escolhidas podem ser vistas na figura 1.26. À esquerda está a broca helicoidal para furação do alumínio e da fibra. À direita a broca em espiral para furação de madeira:

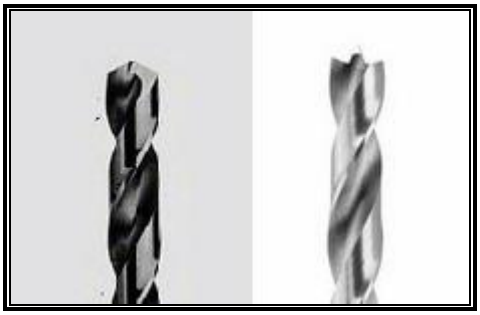

Figura 1.26 - Brocas escolhidas para os ensaios

Quanto aos diâmetros, dois diferentes foram escolhidos. Primeiramente um diâmetro de 8 mm, que é muito usual em furações na indústria. Um segundo valor escolhido é o diâmetro de 3 mm, pelo facto de ser um diâmetro crítico na furação. Como nos afirma Souza [17], furações com diâmetros de brocas menores ou iguais a 3,5 mm podem fazer com que a broca sofra flambagem e consequentemente causar imperfeições no furo.

A tabela 1.2 mostra então um resumo das brocas escolhidas, bem como o preço sugerido pelo fabricante em seu *website* em Portugal.

Quantidade	Descrição	Diâmetro (mm)	Comprimento de trabalho (mm)	Comprimento total (mm)	Preço (EUR)
01	Broca helicoidal para metal HSS-G	3	33	74	3,20
01	Broca helicoidal para metal HSS-G	8	75	117	7,35
01	Broca em espiral para madeira HSS	3	33	74	2,50
01	Broca em espiral para madeira HSS	8	75	117	4,65

Tabela 1.2 - Brocas escolhidas para o trabalho

Capítulo 2

Implementação

Neste capítulo são fornecidos, com os devidos detalhes, o procedimento de implementação e programação da célula robótica. Também são comentadas algumas dificuldades encontradas durante o processo de programação do robô, do sistema de aquisição de dados e, por fim, são expostas as soluções encontradas para alcançar os objetivos do trabalho.

E, conforme foi mencionado anteriormente, o primeiro passo para o avanço do trabalho consiste na construção de uma célula robótica virtual que represente, com o maior nível de pormenores possível, a célula robótica real, existente no laboratório. Para isso, será utilizado o *software* RobotStudio e algumas de suas funcionalidades.

A célula real pode ser vista na figura 2.1 e os detalhes de como obter a célula robótica virtual serão vistos da seção seguinte.

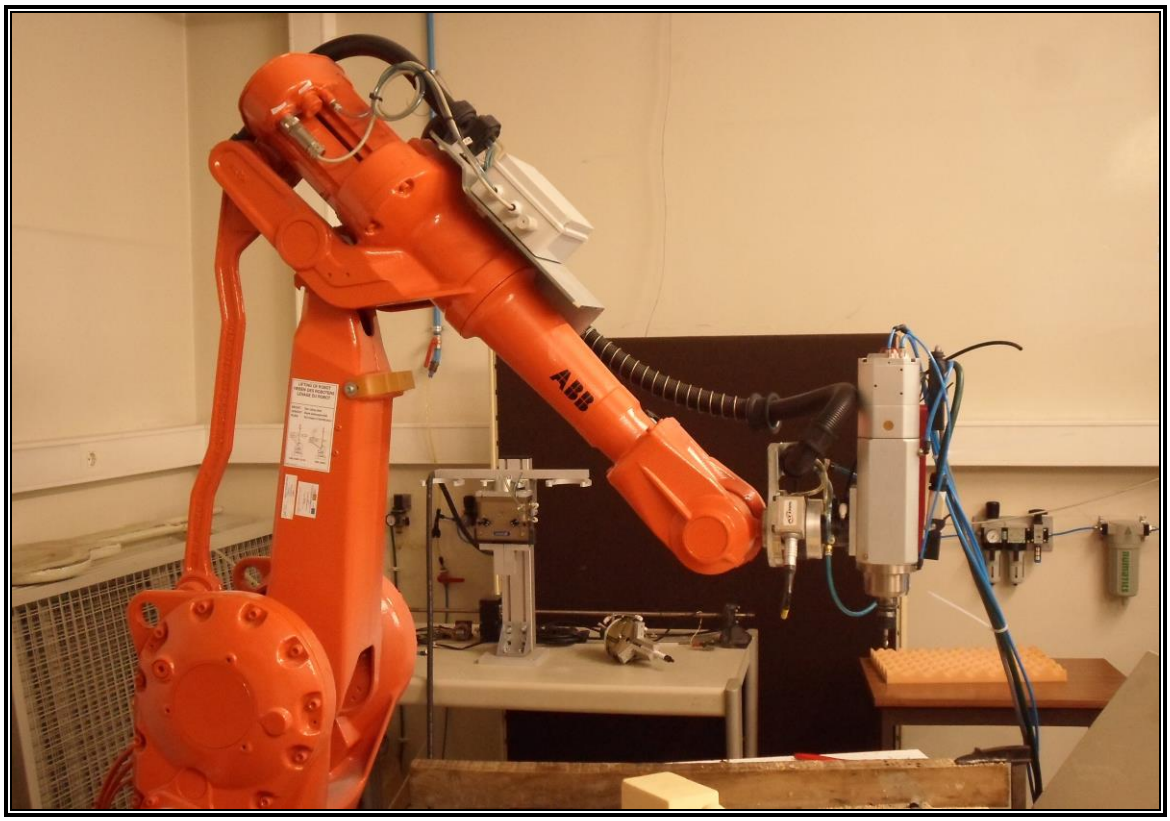

Figura 2.1 - Célula robótica presente no laboratório da Universidade

2.1 - Célula Robótica Virtual

A implementação da célula robótica virtual é essencial para que se realize uma correta programação *offline* do robô, a nível de qualidade visual e também alguma precisão dimensional. E o processo de obtenção da célula virtual inicia-se com a aquisição dos dados da célula robótica real em que se trabalhará. A forma mais vulgar de fazer-se tal procedimento é via sincronização, através do *software* RobotStudio ligado ao controlador real do robô.

Com o RobotStudio aberto e o controlador do robô ligado ao computador, é preciso selecionar a opção *Add Controller*, no menu *Controller* do programa. Em seguida, após a conexão estar completa, deve-se clicar, ainda no menu *Controller*, na opção *Go Offline*. Surgirá então no centro do ecrã uma janela, como mostrada na figura 2.2. Deve-se certificar-se que as opções "*Add system to station*" e "*Create a Transfer relation with the original system*" estejam selecionadas.

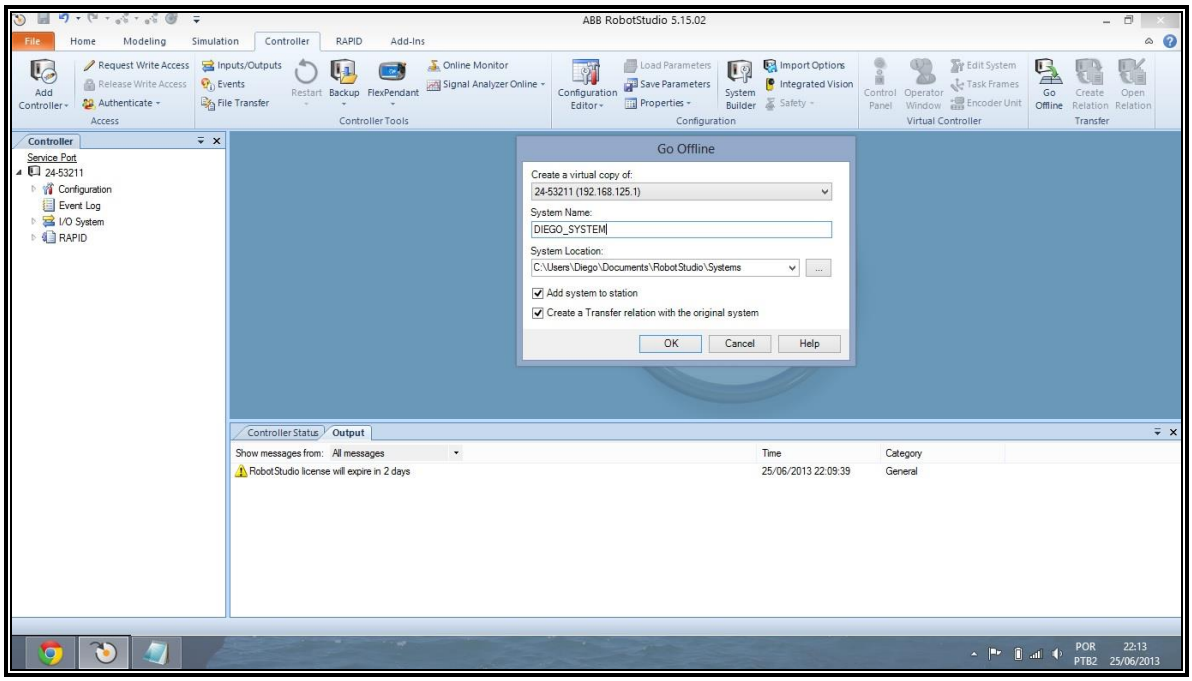

Figura 2.2 - Sincronização da célula robótica

Ainda nesta janela, deve-se inserir um novo nome para o sistema e confirmar as alterações. O programa iniciará então a aquisição dos dados do controlador relativos à célula robótica real. Um novo controlador, agora virtual, é criado. Este é uma cópia do controlador real instalado na célula robótica.

Além disso, o controlador utilizado para este trabalho é também responsável pelo controlo de uma mesa posicionadora. Por isso surgirá uma nova janela, apenas para que se confirme o modelo da mesa utilizada. No caso, basta selecionar o modelo na biblioteca do programa, que é o Irbp_c500_m2009_rev1, como indicado na figura 2.3, e confirmar a seleção.

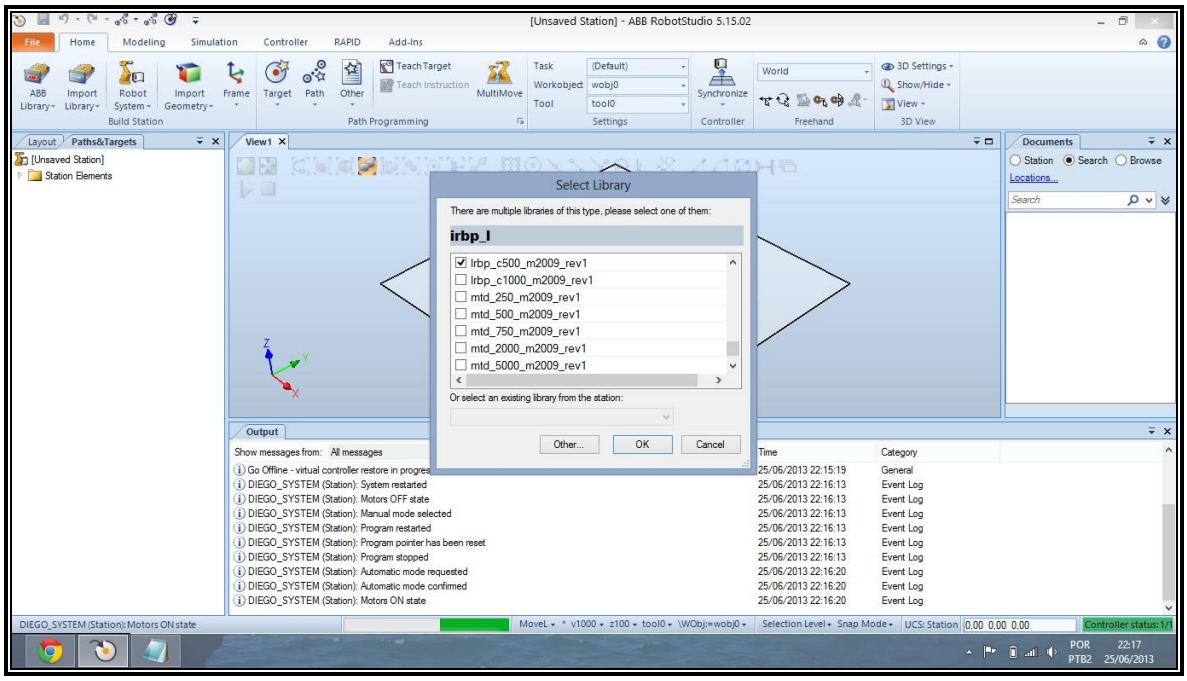

Figura 2.3 - Seleção da mesa posicionadora

Depois de executadas as tarefas anteriormente descritas, teremos uma célula no ecrã do computador, como a mostrada na figura 2.4.

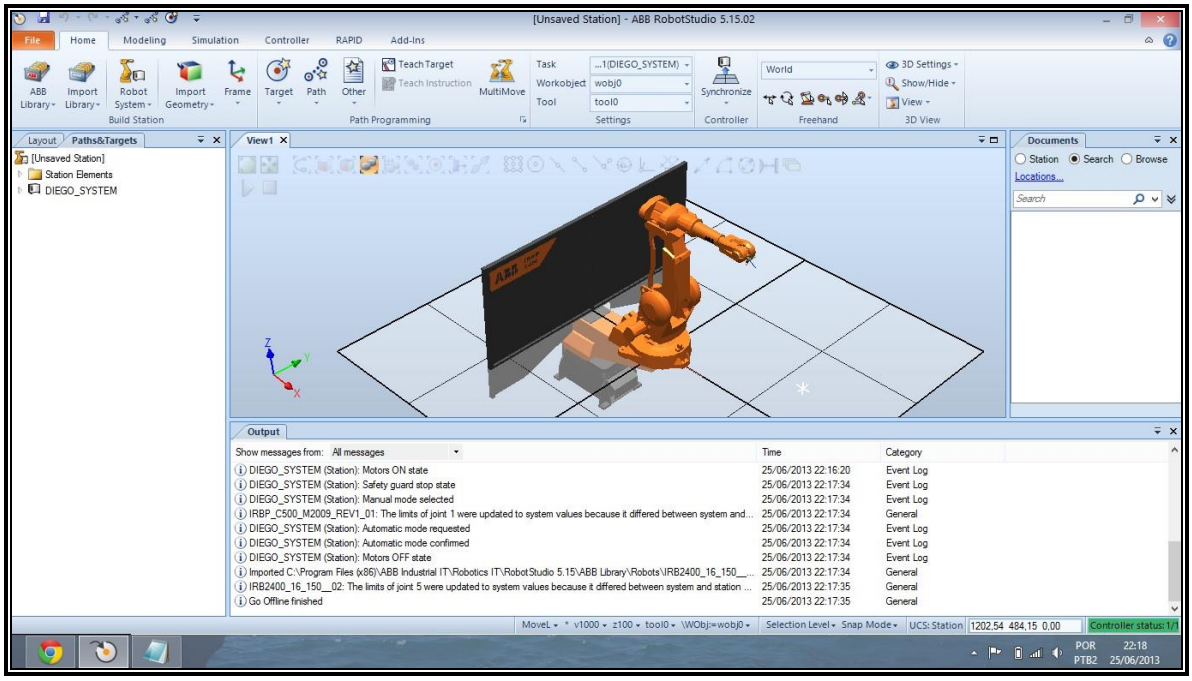

Figura 2.4 - Célula virtual criada após sincronização

Para completar o modelo primário da célula virtual, basta alterar a posição da mesa posicionadora no referencial mundo (que é o referencial principal do robô). Para isso é preciso clicar com o botão direito em seu nome (IRBP_C500_M2009_REV_01) na barra *Layout*, que está no menu *Home*. Isso pode ser observado na figura 2.5. Em seguida, selecionar "*Set Position*" e inserir os novos valores de posição e orientação, a fim de que a nova célula robótica coincida com a original, como se observa na figura 2.6.

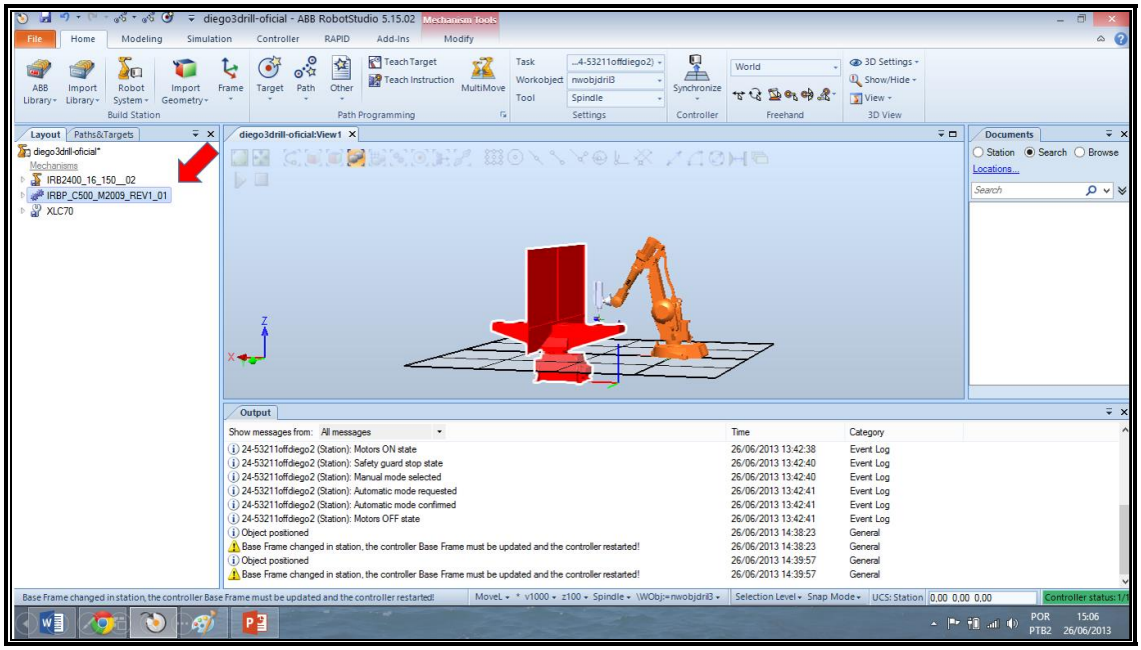

Figura 2.5 - Alterar posição da mesa posicionadora

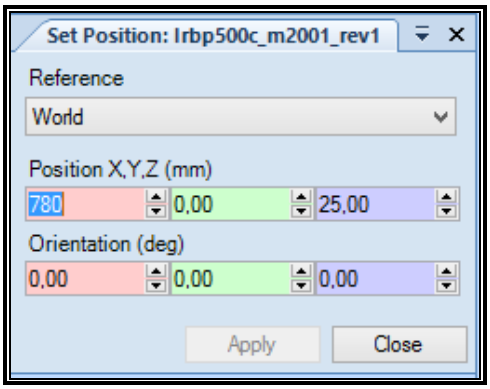

Figura 2.6 - Alterar posição da mesa posicionadora

Porém, esta é apenas a primeira tarefa a ser executada. E sabe-se que a mesa posicionadora, apesar de não ser utilizada no trabalho, está presente na célula real. Assim ela é transferida automaticamente para a célula virtual. Mas há ainda outros componentes, que

devem ser inseridos para tornar a célula virtual semelhante à real. São eles: a base do robô, o pedestal do robô, o *spindle*, o transdutor de força, os acoplamentos existentes entre elemento terminal do robô, *spindle* e transdutor e ainda um modelo de peça a ser furado. Os procedimentos para acrescentar cada um desses elementos ao robô são descritos a seguir.

Inserção da base do robô

O robô presente no laboratório da Faculdade de Engenharia da Universidade do Porto possui uma base que foi adquirida em conjunto com a célula robótica. Entretanto, apesar de o programa RobotStudio possuir uma extensa gama de produtos em sua biblioteca, tal base não se encontra originalmente presente.

Contudo, seu desenho encontra-se disponível. E devido ao facto de ser possível exportar um arquivo gerado em SolidWorks (formato .step) para o RobotStudio, seu uso na célula é simples. O processo de importação é realizado no menu *Home*, na opção *Import Library*. Basta então selecionar *Browse for Library* e encontrar o local em que o ficheiro encontra-se, no disco do computador. Isso pode ser visto na figura 2.7.

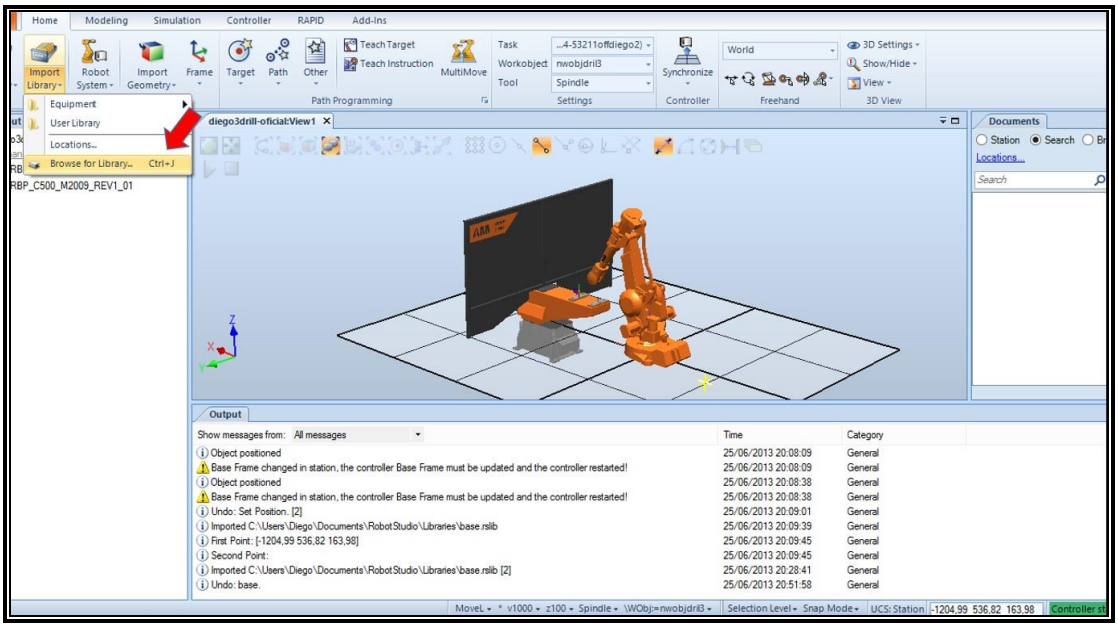

Figura 2.7 - Importar Imagem do Computador

Logo após a base aparecer no ecrã, deve-se definir uma nova posição para a mesma. Isso ocorre de forma semelhante ao que foi realizado para a mesa posicionadora. Para isso, basta clicar com o botão direito sobre a base, que agora deve encontrar-se na barra *Layout,* e depois clicar em *Set Position*. Uma janela será aberta, semelhante à da figura 2.8, onde se deve colocar os novos valores de posição, a fim de que a base fique na posição correta, sob o robô.

Figura 2.8 - Definir posição da base no referencial mundo

Entretanto, o controlador do robô não pode identificar que o mesmo encontra-se sobre uma base e sobre um pedestal. Assim, após a mudança de posição parece haver uma colisão entre a base e o robô no programa. Por isso será preciso mudar a posição do robô no *eixo z* mais adiante. Isso será feito após a inserção do pedestal na célula virtual, visto que o pedestal também influi na altura do robô em relação ao solo.

Inserção do pedestal do robô

Da mesma forma que ocorreu com a base, inserir o pedestal no programa não exige grande esforço. Além disso, visto que o pedestal já está presente na biblioteca no programa, o procedimento será outro. Basta aceder, no menu *Home,* à opção *Import Library* e, em seguida, procurar por *Equipament* e *Robot Pedestal 1400 H240*, conforme pode-se observar na figura 2.9.

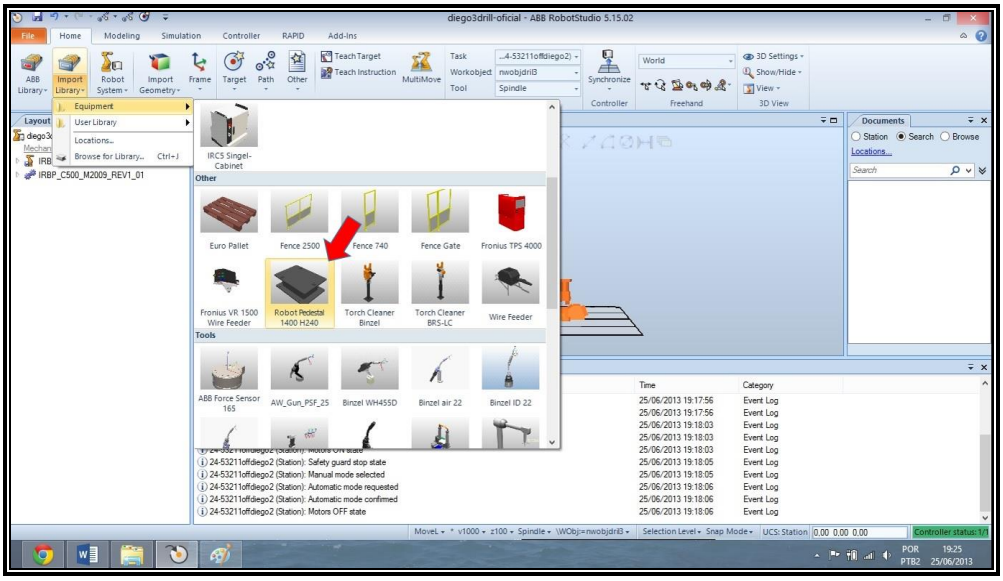

Figura 2.9 - Como importar o pedestal na biblioteca do RobotStudio

Ao fim, o pedestal aparecerá no ecrã e, coincidentemente, na mesma posição do robô e da base. Logo, este é o momento de alterar a posição do conjunto que tínhamos anteriormente (robô + mesa) e também do pedestal, que vai sobre a base. A alteração dar-se-á localmente para cada objeto e somente no eixo z. Para executar tal processo basta, ainda no menu *Home* e na barra *Layout,* clicar com o botão direito do rato sobre o objeto que se deseja mover e selecionar o *Set Position*. Após isso, aparece uma nova aba para que se insira os valores desejados, conforme visto anteriormente (figura 2.8*)*.

No caso do pedestal, este será movido de apenas 162 mm, correspondente à altura da base sobre o qual está. No caso do robô, este será movido da altura do conjunto base + pedestal, que vale 387 mm. O resultado após as alterações é o que se vê na figura 2.10.

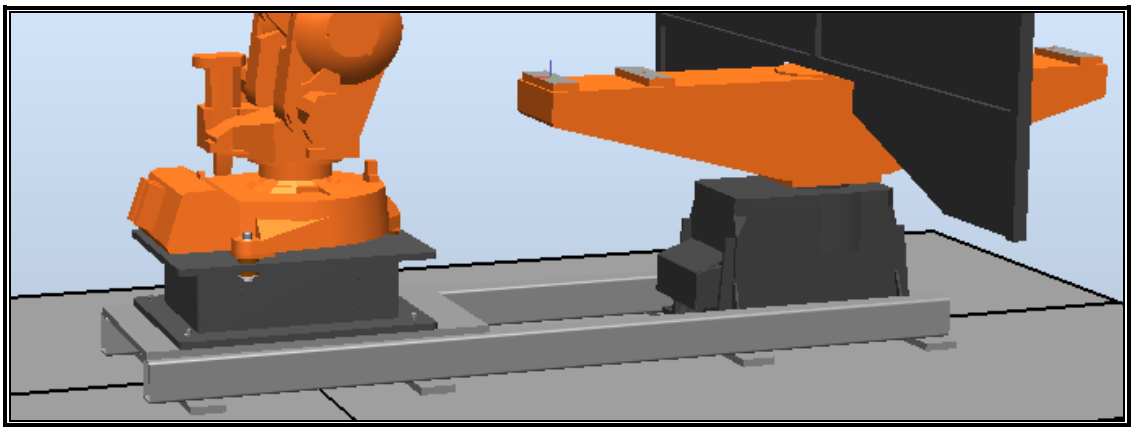

Figura 2.10 - Disposição na base da célula robótica

Note-se que este procedimento aqui descrito pode variar, de acordo com a ordem com que os procedimentos são realizados.

Inserção dos componentes no elemento terminal do robô

Quanto aos componentes que se encontram nas proximidades do elemento terminal do robô, estes podem ser inseridos no RobotStudio como uma peça apenas, visto que todos possuem contacto físico entre si. São eles: *spindle*, transdutor de força e acoplamentos. E, assim como feito com a base do robô, utilizou-se aqui também um ficheiro já disponível. O conjunto, que pode ser visto na figura 2.11, foi exportado para o RobotStudio.

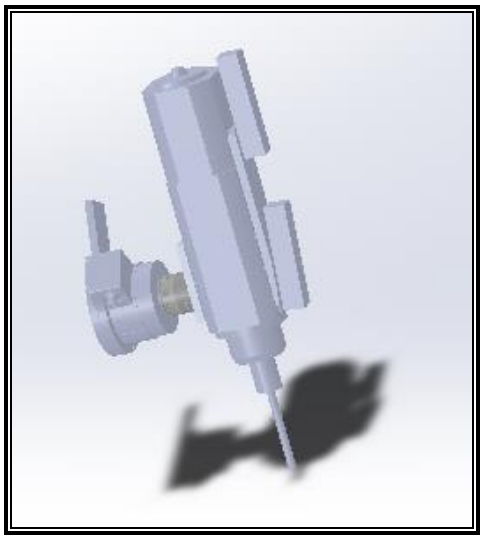

Figura 2.11 - Spindle e demais elementos no programa SolidWorks

Observe-se que é necessário que o ficheiro esteja salvo em formato "*.step*". Porém, quando importada, a imagem é direcionada para a posição zero do referencial mundo, enquanto é preciso que conjunto fique unido ao elemento terminal do robô. Para isso basta procurar, na aba *Layout* (menu *Home)*, pelo nome do ficheiro importado anteriormente. Depois basta clicar com o botão direito do rato e selecionar a opção "A*ttach to*". Em seguida aparece a opção de elementos da célula virtual em que é possível conectar o conjunto. Escolhe-se, portanto, ligar o *spindle* e os demais elementos ao robô IRB2400_16_150_102. Observe-se ainda que o programa perguntará se a posição do conjunto deve ser atualizada. Neste momento é preciso dizer sim, pois o conjunto irá diretamente para o elemento terminal do robô, conforme a figura 2.12.

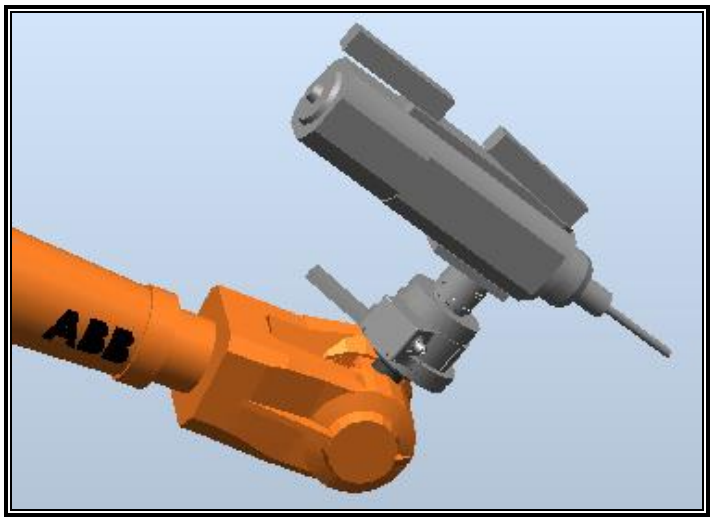

Figura 2.12 – Spindle ligado ao braço do robô

Inserção de uma peça a ser furada

Além dos componentes necessários para o funcionamento da célula robótica, é possível no RobotStudio gerar-se uma o desenho de uma peça para representação da furação.

E não é preciso sequer recorrer ao auxílio de um programa específico, como AutoCAD, SolidWorks ou CATIA. Ao entrar no menu *Modeling* do RobotStudio podemos observar a opção *Solid* a aparecer na barra do programa. Ao clicar na seta sob esta opção, o RobotStudio apresenta uma série de geometrias usuais, para que criemos uma peça qualquer (vide figura 2.13). No caso deste trabalho, para fins estéticos, uma peça em forma de caixa ou uma simples placa é suficiente.

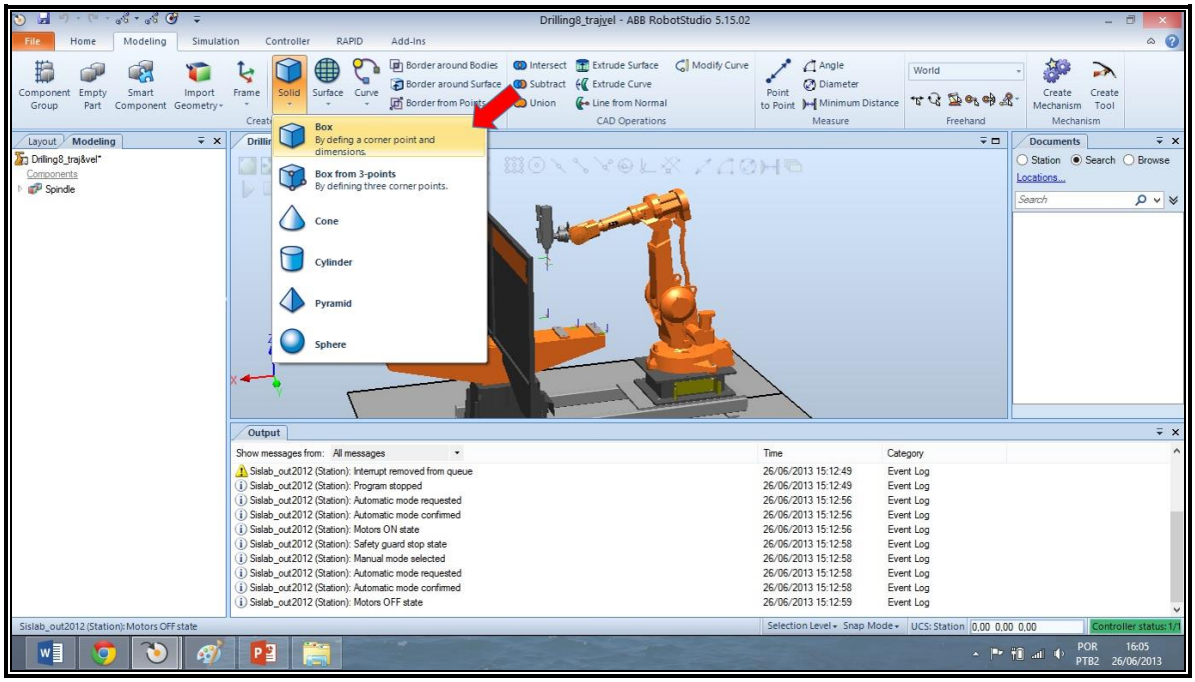

Figura 2.13 - Inserir formas básicas no RobotStudio

Para a criação da caixa ou da placa seleciona-se a opção "*Box*". Logo após é aberta uma aba, conforme mostra a figura 2.14. Nela é possível definir o tamanho do objeto, sua posição e sua orientação em um referencial.

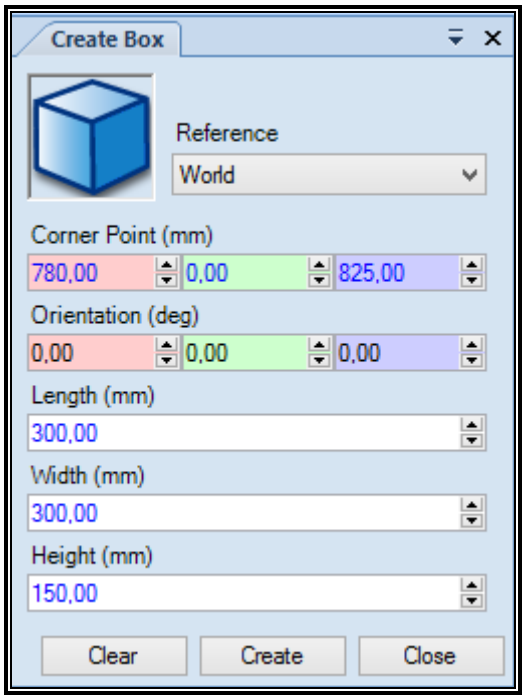

Figura 2.14 - Criação de uma peça para ser furada no RobotStudio

Ao fim dessa etapa, temos uma célula robótica completa, conforme nos mostra a figura 2.15.

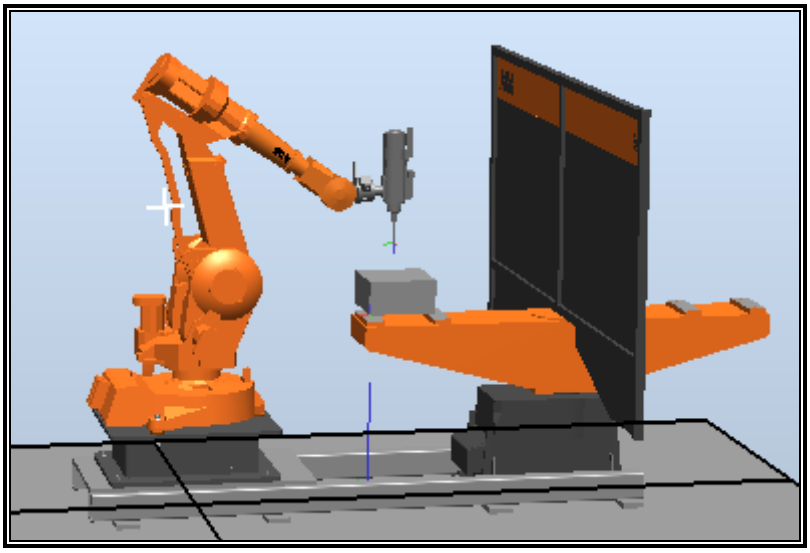

Figura 2.15 - Célula robótica virtual

Além disso, uma observação importante quando se pretende realizar uma modificação no programa usando a estação de trabalho, e não diretamente no código RAPID, deve ser sempre sincronizar a estação modificada com o controlador virtual. Da mesma forma, quando se deseja

realizar uma simulação na estação de trabalho após uma modificação no código RAPID, é preciso realizar uma sincronização, mas agora do controlador virtual para a estação.

Por fim, antes de iniciar a programação do robô, é interessante desde já definir três novos elementos, que serão peça fundamental para execução do trabalho.

Workobjects:

Um *workobject* é um referencial criado dentro do espaço de trabalho do robô. Por definição, o wobj0 (*workobject 0*) é o referencial principal do robô, também chamado de referencial *world* ou referencial mundo. Porém, o RobotStudio permite a criação de outros *workobjects*, a fim de facilitar a programação do robô. Neste trabalho, será definido um *workobject* que terá como origem o ponto em que as furações serão realizadas durante os ensaios. Isso tornará o processo mais simples e rápido, devido ao grande número de furações necessárias.

Tools:

Uma "*tool*", por tradução direta, ferramenta, é o elemento inserido no ponto terminal do robô e que tem por função executar uma tarefa concreta. Pode ser uma garra, uma ferramenta de projeção de jato de tinha ou, no caso presente, o *spindle*. Porém, quando se refere a *tool*, refere-se também a *Tool Center Point* (*TCP*). Este é a origem de um referencial para a *tool* em questão. Obviamente, no caso da furação, o *TCP* será definido na extremidade da broca.

Targets:

Targets são pontos programados que permitem definir o percurso a percorrer pela ferramenta do robô. No caso da furação, o número de *targets* necessários é reduzido, visto que compreende apenas um movimento de avanço e de recuo. Porém, como será visto mais adiante, é preciso definir alguns *targets* extras, com a finalidade de tornar a programação do robô um pouco mais simples.

2.2 – Programação

A partir do momento em que a célula robótica virtual está finalizada, é possível iniciar a programação do robô. Entretanto, é preciso ter sempre em mente o objetivo do trabalho, que é realizar furações a partir de três diferentes estratégias de controlo. A primeira consiste apenas em controlo de posição e velocidade. As duas demais envolvem controlo de força. Portanto, ao manter-se isso em mente, toda a programação, incluindo a criação de *workobjects*, *tools* e *targets*, será feita de forma a facilitar o processo como um todo.

A programação de um robô pode ser definida fundamentalmente como o processo específico dos movimentos do robô e de sua interação com outros equipamentos. Entretanto, não é somente isso que será realizado aqui. A aquisição dos dados do processo (posições,

velocidades, forças e binários aplicados, tempo de execução das tarefas) é também essencial. Em outras palavras, é preciso programar o robô não somente em trajetória, velocidade e força, mas também a fim de que o controlador realize leituras e salve dados convertidos para um formato que possa ser lido posteriormente em algum programa existente.

Além disso, no caso dos robôs da ABB, a linguagem oficial de programação é a chamada RAPID. Quanto a sua composição básica, a linguagem utiliza uma rotina principal (*main*), algumas sub-rotinas (*procedures*, *traps* ou *functions*) e precisa também de uma declaração antecedente de dados do programa (*programm data*) [18].

2.2.1 – Procedimentos iniciais

O primeiro procedimento para dar início à programação é a definição de *workobjects*. Uma das formas de se criar um *workobject*, ligado ao furo ou à peça a ser furada (no caso de realizar múltiplos furos), é a partir da definição de um *frame* (sistemas de coordenadas). E a criação de um *frame* é feita a partir de três pontos.

No menu *Home*, deve-se selecionar a opção *Frame* e, em seguida, *Frame from Three Points*. É necessário também estar certo de que estejam ativadas as funções representadas pelas setas 1 e 2 na figura 2.16.

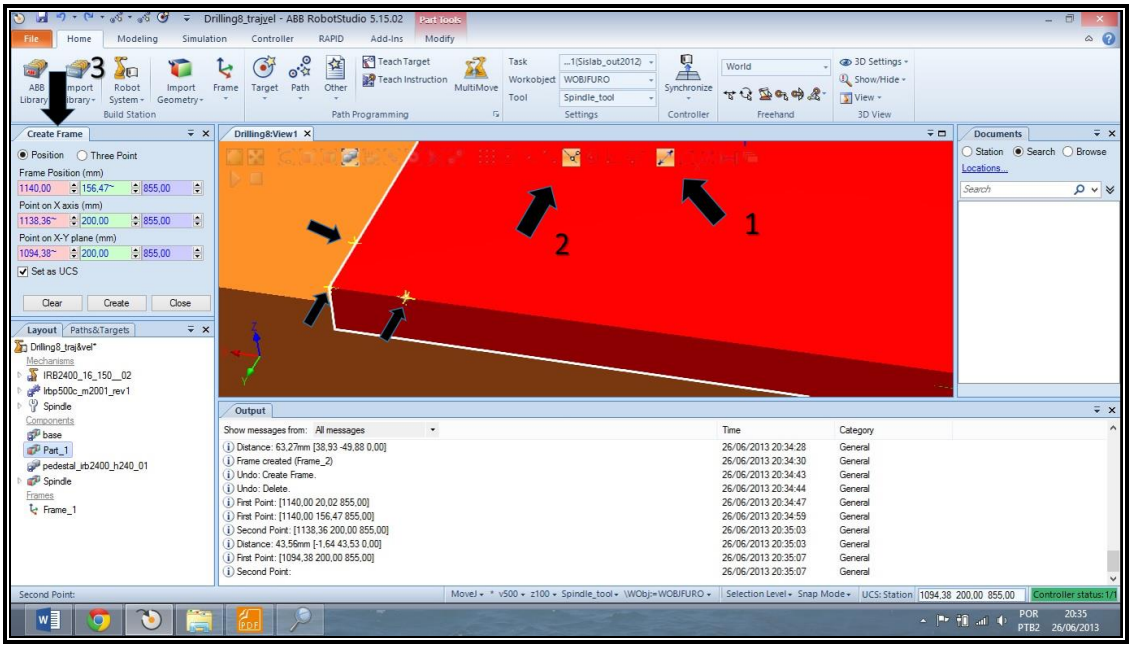

Figura 2.16 - Criação de Frame por três pontos

Imediatamente uma nova aba se abrirá, representada pela seta 3 na figura 2.16. Nela é necessário inserir três diferentes pontos de um plano xy, que definirão os eixos coordenados

do novo *frame*. Uma opção é a seleção de três pontos quaisquer na superfície da peça. Neste caso, os pontos foram selecionados na borda da peça e no sentido anti-horário (para que sejam positivos se comparados ao referencial mundo). Estes pontos são representados pelas setas menores.

Após isso, basta clicar em *Create*, na aba aberta na lateral esquerda do ecrã. O *frame* está criado e já aparece na aba *Layout*, no menu *Home*. E mais, com o *frame* criado na extremidade da peça e sabendo-se o tamanho da mesma, é possível movê-lo para qualquer outro ponto em sua superfície, caso seja preciso.

Criação do *workobject* a partir do *frame*:

O RobotStudio possui uma função que permite criar um *workobject* diretamente de um *frame*. O procedimento é simples. Basta clicar com o botão direito do mouse sobre o *frame* criado e selecionar a opção "*Convert Frame to Workobject*". A figura 2.17 mostra o resultado, um *workobject* criado na extremidade da placa.

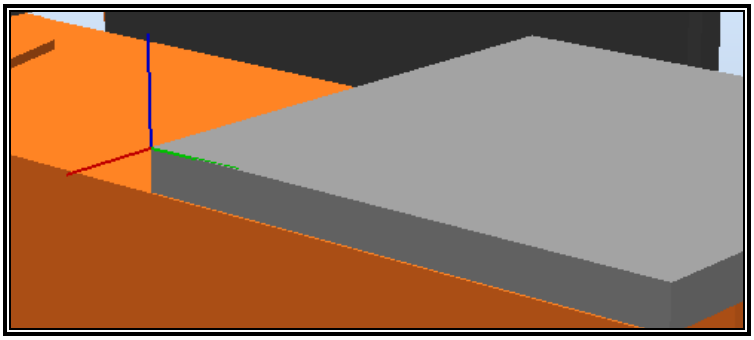

Figura 2.17 – Workobject criado na extremidade da placa

E para identificar o *workobject* é preciso, no menu *Home*, aceder à aba *Paths&Targets*. O *workobject* criado já está na estação. E, como declarado anteriormente, para obtê-lo em código RAPID é preciso somente realizar uma sincronização. Uma possibilidade para se fazer isso é entrar no menu *RAPID* e selecionar a opção *Synchronize to VC* (*Virtual Controller*).

Por fim, se necessário verificar o código RAPID do *workobject* após a sincronização, assim como sua posição e orientação no referencial mundo, é preciso apenas dar um clique duplo no módulo do programa (Module1), como é possível ver na figura 2.18.

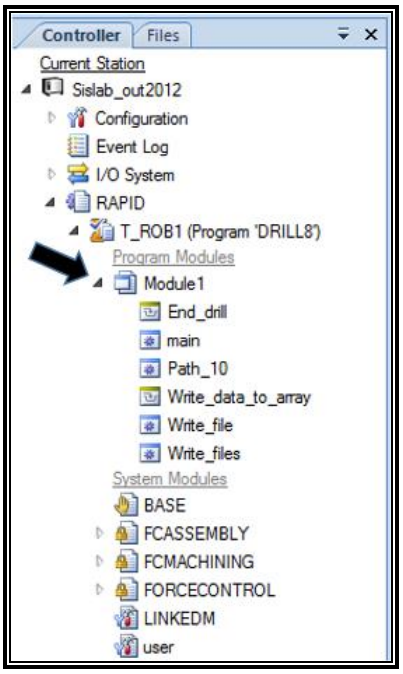

Figura 2.18 - Módulo com programa RAPID gerado

O *workobject* criado é representado na linguagem RAPID, conforme pode ser visto nos anexos B em diante, por uma variável persistente (PERS). *Workobjects* normalmente assumem esse tipo de variável no RobotStudio, visto que as PERS assumem sempre o último valor adquirido de posição [19].

Criação de uma *tooldata* (relacionada ao *spindle*):

Para a criação de um referencial de ferramenta o procedimento é muito parecido com o anterior. É possível, portanto, criar um *frame* na extremidade da broca montada no *spindle*. Deve-se também ter atenção para que os eixos coordenados fiquem alinhados com o *spindle*. A figura 2.19 mostra o resultado final.

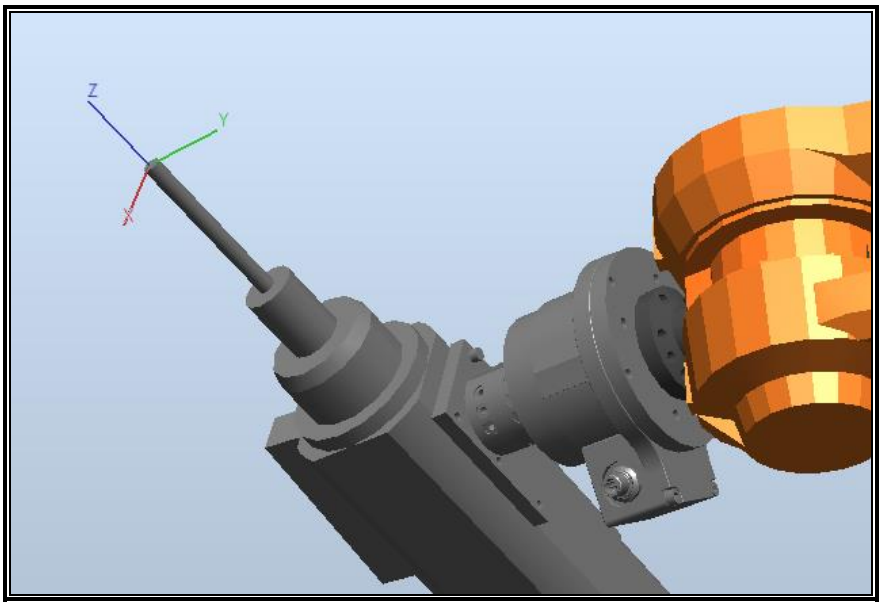

Figura 2.19 - Criação de uma tooldata vinculada ao spindle

Importante notar que os valores aqui configurados são apenas para simulação. Ao executar o programa no controlador real, a posição da *tooldata* no referencial mundo deverá ser reprogramada, da mesma forma que o *workobject*. Entretanto, as orientações poderão permanecer.

Agora, com *workobjects* e *tools* criadas, é possível avançar para o próximo passo, que consiste em determinar alguns pontos pelo qual o robô deve passar: os *targets*.

Criação de *targets*:

A definição de *targets* para furações onde se efetua apenas um furo é relativamente simples. Em tese, seriam necessários apenas três pontos a programar: um ponto mais distante do furo, um segundo ponto mais próximo e um terceiro no interior da peça. Os movimentos seriam lineares e, no retorno, passariam pelos mesmos pontos. De facto, para uma operação em que não se usa controlo de força isso seria suficiente. Porém, como veremos mais à frente, o *FC Pressure* exige alguns pontos mais. Portanto, para manter um padrão na análise das três estratégias de controlo, desde já os *targets* são definidos visando o caso mais complexo. Os pontos são então configurados de acordo com a figura 2.20.

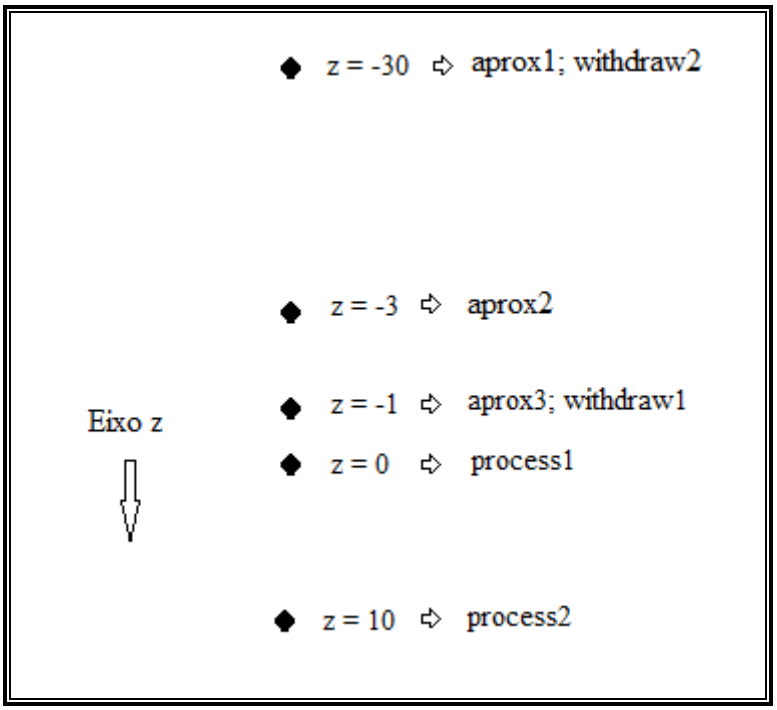

Figura 2.20 - Posicionamento dos Targets

Definidos os pontos que se deseja programar, o procedimento agora é, no menu *Home*, clicar em *Target* e selecionar *Create Target*. Entretanto, antes de executar isso, o ideal é procurar novamente pelo *workobject* criado, clicar com o botão direito do rato e selecionar *Set as UCS*. Isso indica que, a partir de agora, o *workobject* é um referencial local que podemos usar para definir os *targets* que serão criados. De facto, ao retornar à aba *Create Target*, é possível cria-los em relação ao *UCS*. Vide figura 2.21.

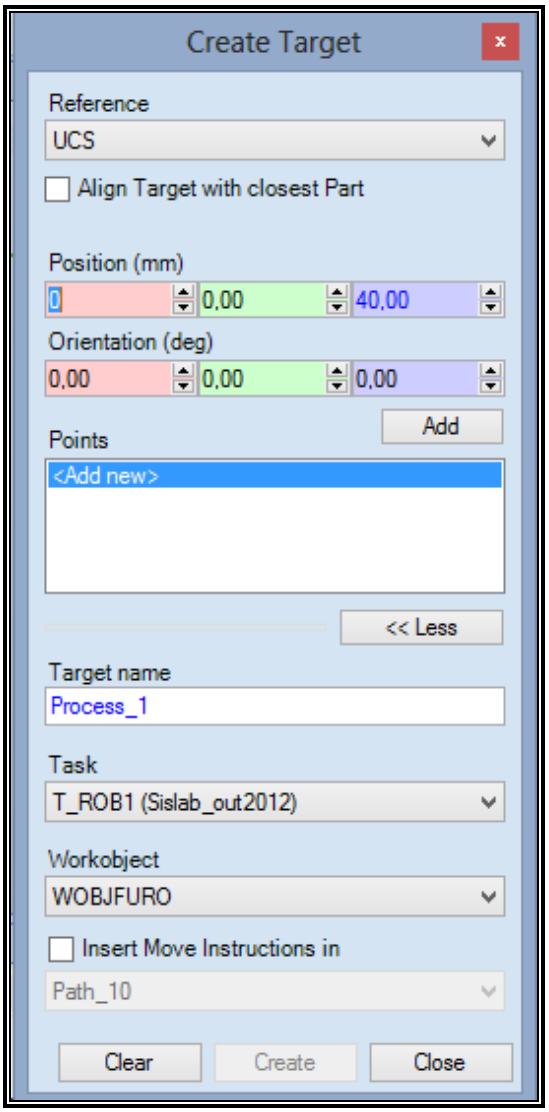

Figura 2.21 - Criação dos Targets

Após inserir os valores e o nome do *target*, deve-se clicar em *Add* e, em seguida, *Create*. Entretanto, deve-se atentar para uma potencial dificuldade que é normalmente encontrada. Ao definir os pontos para a *tool* alcançar no *workobject*, por vezes a orientação destes não são coincidentes. Em outras palavras, o RobotStudio (e o controlador do robô) sempre tentarão correlacionar os eixos do *workobject* e da *tooldata*, de forma que estes coincidam. Caso contrário, são necessárias algumas rotações nos eixos, a fim de que isso ocorra. Uma forma de verificar isso é clicar com o botão direito do *target* criado e selecionar "*View Robot at Target*". Se o robô não vai para o *target* criado, uma mudança na configuração dos eixos é necessária.

Além disso, se fosse necessário realizar múltiplos furos em uma mesma peça, o procedimento duraria um pouco mais de tempo. Neste trabalho, a diferença de um *target* para os demais é apenas na localização da origem do referencial sobre o eixo z. Em outros casos,

seria também necessária uma mudança nos eixos x e y, de acordo com as posições reais que se desejam.

E, quanto à criação dos *targets*, o mesmo processo deve ser repetido para os demais. Ao fim, ter-se-á preparada a célula para o início da programação de movimentos e para as primeiras simulações em *software*. Os *targets* criados apareceram na estação de trabalho e, após uma sincronização, estarão disponíveis em código RAPID. Um exemplo de *target* em linguagem de programação é o seguinte:

```
CONST robtarget Aprox3:=[[0,0,1],[0,0,1,0],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
```

Observe-se que o *target* criado é uma constante vinculada às posições do robô (*robtarget*). O primeiro termo, mais à frente do nome, indica a posição programada. Depois há uma informação relativa a orientações, dada em quatérnios. Em seguida a configuração do robô no programa, de acordo com os eixos do próprio robô. E, por fim, uma configuração relativa à juntas externas. Quando estas não são utilizadas, como é o caso, seu valor é sempre igual a 9E9 [19].

2.2.2 – Programação sem controlo de força

A programação do robô sem controlo de força, ou simplesmente programação em trajetória e velocidade, é caracterizada basicamente pela adição de *paths* ao que foi feito até o momento. No caso, se já existem *targets* definidos, a criação de um *path* consiste em duas ações.

A primeira é, no menu *Home*, clicar em *Path* e selecionar a opção *Empty Path*. Um *path* é então automaticamente criado na aba *Paths&Targets*. A segunda ação consiste em selecionar os *targets* que participarão do movimento e arrastá-los para dentro do *path* criado. Na figura 2.22 é possível observar os sete pontos selecionados a serem arrastados para o *path* criado, denominado drill.

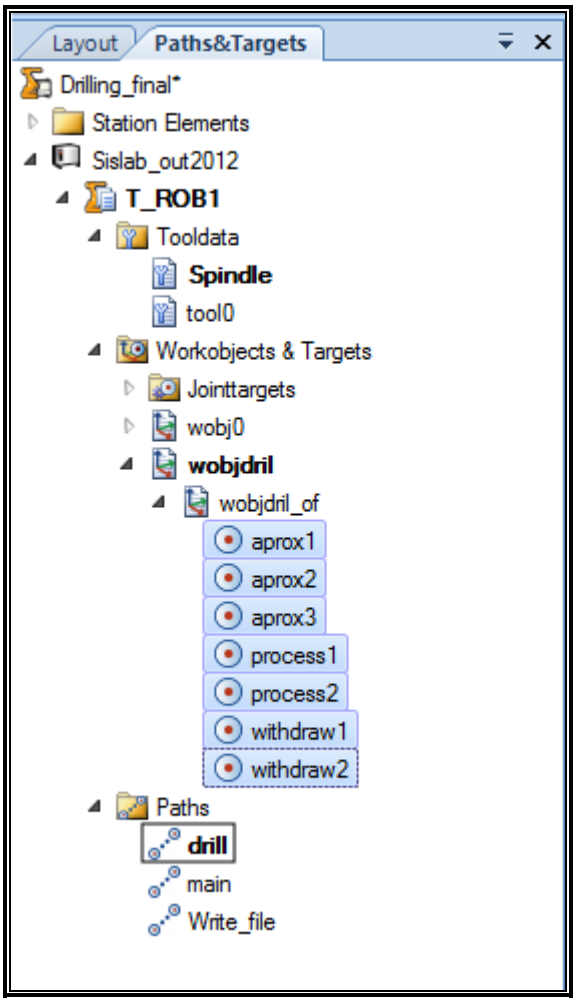

Figura 2.22 - Criação de um path

Assim a programação do robô já está quase terminada. Entretanto, os movimentos de furação devem ser lineares e com velocidades e precisões bem definidas. Para alterar esses parâmetros, é necessário clicar com o botão direito do rato sobre cada movimento dentro do *path* e selecionar a opção *Modify Instruction*. Uma nova aba é aberta, onde se pode modificar o tipo de movimento, a velocidade e a zona de aproximação. Estes não são necessariamente iguais para todos os movimentos programados neste trabalho. Além disso, a zona *fine* garante uma precisão superior do robô, mas compreende também um ponto em que o robô atinge a velocidade zero. Dessa forma, esse comando é necessário no *target* Process2, no interior do furo, e desnecessário fora. Os valores determinados serão exibidos logo a seguir, na figura 2.23.

Notar que, a cada modificação nos movimentos é necessário realizar uma reconfiguração do robô. O procedimento consiste em clicar com o botão direito do rato sobre o *path* que se deseja configurar e selecionar a opção *Configurations*. Em seguida, seleciona-se a opção *Auto Configuration*. Por fim, após uma nova sincronização com o *Virtual Controler,* um novo código é gerado e os *paths* nele aparecem.

Portanto, após esses procedimentos, o programa (que encontra-se no Anexo B) está pronto para ser exportado para o controlador. A figura 2.24 mostra a sequência de movimentos em linguagem RAPID, extraída do programa RobotStudio.

```
PROC drill()

    MoveJ aprox1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;
    MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;
    MoveL process1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL process2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;
    MoveJ aprox1,v10,z1,Spindle\WObj:=wobjdril;

ENDPROC
```

Figura 2.23 - Movimentação sem Controlo de Força

Os pontos e as instruções aqui descritos são os mesmos que foram anteriormente definidos. Quanto a elas, a instrução MoveJ (Move Joint) indica um movimento realizado no espaço das juntas do robô. Já a instrução MoveL (Move Linear) indica que o *tool center point* será transladado linearmente de um ponto para o outro. A instrução v10 indica um movimento com velocidade de 10 mm/s. Os valores de zona z1 e fine indicam a precisão de localização do robô no movimento (mais informações sobre zonas podem ser encontradas em [19]), sendo que a zona fine exige que o robô reduza sua velocidade a zero ao fim do movimento. Ainda, Spindle indica a *tooldata* utilizada e wobjdril indica o *workobject* de referência do movimento.

Todavia, esta é apenas uma das estratégias utilizadas. As demais estratégias, que envolvem controlo de força, serão descritas nas seções seguintes.

2.2.3 – Programação do *FC Speed Change*

Como explicado anteriormente, o *FC Speed Change* consiste em programar o robô de tal forma que, ao encontrar um obstáculo que exerça sobre ele uma força cujo valor seja maior que tiver sido programado, o manipulador robótico diminui a velocidade de sua trajetória para um patamar inferior. Quando a força retornar a valores menores que o estabelecido, a velocidade retorna ao valor programado inicialmente. Portanto, este controlo de força está intimamente ligado ao controle de velocidade do robô (daí o nome *Speed Change*).

Por conta dessa íntima relação com o controlo de um robô em velocidade, o *Speed Change* pode ser desenvolvido a partir do programa RAPID estabelecido na seção anterior. De

facto, após a consulta ao manual [9] e leitura minuciosa sobre o funcionamento desta estratégia de controlo de força, percebeu-se que apenas duas linhas do programa são mudadas com a finalidade de ativar o controlo de força. Também é preciso atentar-se para as recomendações do fabricante ABB contidas no mesmo manual, que afirma que, sempre ao utilizar controlo de força, este deve ser ativado somente quando a ferramenta estiver a poucos milímetros da peça a ser maquinada. A partir disso, chegamos ao resultado exibido na figura 2.24.

```
PROC fcdrillspeedchange()

    MoveJ aprox1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;

    FCSpdChgAct 50\NonStopAllTime;

    MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;
    MoveL process1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL process2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw1,v10,fine,Spindle\WObj:=wobjdril;

    FCSpdChgDeact;

    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;
    MoveJ aprox1,v10,z1,Spindle\WObj:=wobjdril;

ENDPROC
```

Figura 2.24 - Movimentação com o FC Speed Change

As instruções do tipo *Move* são similares às contidas na figura 2.23. Já a instrução *FCspdChgAct*, responsável pela ativação do *FC Speed Change*, é inserida quando o robô se encontra a 1 mm da peça que será furada. O primeiro argumento em frente à instrução, o número 50, denota a força, em Newton, que será utilizada como valor limite. Já o argumento *NonStopAllTime* é ativado a fim de que o robô não pare por qualquer sobrecarga que ocorra à velocidade mínima [9]. Em seguida, quando o robô termina a furação, a instrução *FC*SpdChgDeact é ligada e o robô deixa de ser controlado em força, retornando ao controlo de posição e velocidade.

Outro ponto a ser destacado na operação que envolve controlo de força é em relação à calibração da ferramenta, que se faz sempre necessária nestes casos. Para se realizar uma calibração deve-se então utilizar a instrução *FC*Calib, que pode ser ativada no início da movimentação do robô. Aqui a instrução é para a calibração da variável tdril_LD, contida nos anexos B, C e D do presente trabalho. A variável foi obtida durante os ensaios e depois os valores obtidos foram exportados para o computador.

2.2.4 – Programação do *FC Pressure*

O *FC Pressure* é a segunda estratégia de controlo de força presente no controlador IRC5, da ABB. Seu funcionamento consiste em "obrigar" o robô a manter uma força de contacto constante sobre a peça que está a ser maquinada.

E da mesma forma que o *FC Speed Change,* o *FC Pressure* também tem por base a programação em posição e velocidade. Entretanto, um número bem maior de parâmetros deve ser configurado, que vão muito além da simples definição de uma força de contacto. E mais, além da consulta aos manuais [9], durante a programação do *FC Pressure* foi necessária a realização de alguns testes com o robô para que se pudesse entender o significado de cada parâmetro programado. Isso ocorre primeiramente porque no manual nem sempre está claro o que uma alteração dos parâmetros pode provocar. Em segundo lugar, para testar uma combinação de parâmetros que é útil na furação.

Outro auxílio utilizado posteriormente na programação do controlo de força foi o *Flex Pendant* do controlador. Através do Machining Power Pack (pacote da ABB para auxiliar na programação de operações de maquinagem), foi possível gerar um programa básico do tipo *FC Pressure*. Este foi também útil para que se chegasse a uma versão final do programa, conforme mostra a figura 2.25.

```
PROC fcdrillpress()

    MoveL aprox1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;
    MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;

    FCPress1LStart process1 *
    FCPressL process2,v10,30,fine,Spindle,\wobj:=wobjdril;
    FCPressLEnd withdraw1,v10\ForceChange:=50\ZeroContactValue:=10;

    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox1,v10,fine,Spindle\WObj:=wobjdril;

ENDPROC
```

Figura 2.25 - Movimentação com o FC Pressure

Note-se que o símbolo (*) fornecido na figura 2.25 indica, na verdade, a continuação do programa que, devido ao grande tamanho ocupado naquela linha, não permitia que a imagem fosse exportada com qualidade para este documento. A instrução completa que deveria estar presente na linha é:

FCPress1LStart process1,v10\Fz:=30,10
\ForceFrameRef:=FC_REFFRAME_WOBJ\ForceChange:=10
\DampingTune:=100\TimeOut:=1\PosSupvDist:=9e9,fine,Spindle\WObj:=wobjdril;

E em relação às instruções, as que estão relacionadas ao controlo de posição e velocidade permanecem como no primeiro programa. Já nas instruções, os parâmetros que aparecem são novos e exigem um maior detalhamento em sua explicação, como será feito.

- ✓ *FC*Press1LStart:

Função que inicia o controlo de força *FC Pressure* em uma trajetória linear. É utilizada quando o *tool center point* estiver próximo da superfície a ser maquinada e tem como característica importante o facto de buscar uma força contra o movimento em apenas uma direção. No caso, como pode-se observar no argumento da função, será utilizada uma força de 30N na direção "z". Porém, o número "10" que aparece logo em seguida indica que o robô não vai buscar inicialmente a força de 30N, mas apenas 10% de seu valor. Isso minimiza o impacto com a peça e algumas imprecisões de posicionamento. Ainda em relação a esta força, ela possui módulo e direção orientadas no *workobjetc* wobjdril. Cabe ainda observar que, quando nada se diz a respeito de forças em x e y, o sistema interpreta que essas devem ser nulas.

Já o argumento *ForceChange*, dado em N/s, está ligado ao acréscimo de força em relação ao tempo. Quanto mais baixo, mais rápida a resposta do robô. Quanto mais alto, mais lento ela torna-se.

DampingTune é a relação, em percentagem, do valor de força medido e da força resultante aplicada. Aqui o valor é 100%, que é o mesmo valor utilizado pelo robô quando este argumento é omitido. Quanto menor, maior a sensibilidade do robô às forças externas. Duas observações a respeito deste comando são:

1. Este pode assumir valores superiores a 100%;
2. O valor mínimo é de 50%.

Quanto ao TimeOut, medido em segundos, é o argumento que define o tempo em que o robô deve permanecer em busca da força de contacto. Caso o robô não encontre uma força (10% de 30N, neste exemplo), ele prossegue seu curso a partir da próxima instrução dada.

Por fim, o argumento PosSupvDist determina a distância em que o robô deve parar caso não consiga encontrar uma força de contacto. O valor padrão, caso o argumento seja omitido, é 20 mm.

- ✓ *FC*PressL:

Esta função é empregue quando o controlo de força já estiver em uso. De facto, nela somente é preciso repetir o valor de força anteriormente configurado (ou mudar o valor, caso seja necessário), assim como os valores de velocidade, zona, *tool* e *workobject*.

Importante observar aqui que esta função deve ser utilizada entre um comando do tipo *FC* Press Start e um *FC* Press End.

- ✓ *FC*PressLEnd:

É utilizado para mover o robô de um ponto em contacto com a superfície para um ponto em que não haja contacto. Ou seja, é recomendado que o segundo ponto esteja próximo da peça, mas sem contacto algum. Quando o *tool* center point do robô chega a este ponto, o controlo de força é desligado.

Quanto aos demais movimentos, são todos semelhantes ao caso em que o robô é controlado apenas em posição e velocidade.

Portanto, neste ponto do trabalho, já é possível realizar movimentos com o robô através das três estratégias de controlo que caracterizam o objetivo do trabalho. Assim, é necessário agora programar um sistema de aquisição de dados que possa colaborar com tal questão.

2.2.5 – Aquisição de dados no RobotStudio

Quanto ao sistema de aquisição de dados, duas foram as opções levantadas na introdução do relatório para serem utilizadas: o *software* RobotStudio e o ABB Test Signal Viewer. O RobotStudio será utilizado para tal fim, enquanto o ABB Test Signal Viewer será ligado ao robô apenas para monitorização das forças durante a furação. Portanto, algumas tarefas são acrescentadas aos programas gerados até o momento, sendo que esta programação é válida para as três estratégias de controlo utilizadas. E isso é feito através de uma modificação manual dos códigos RAPID gerados.

As principais modificações a fazer quanto à aquisição de dados envolve:

- ✓ Declaração das variáveis de processo:

Antes de iniciar a programação do sistema de aquisição de dados é preciso definir os dados que se desejam adquirir do robô. Basicamente são eles: tempo de execução do programa, posição da ferramenta de trabalho, velocidade linear da ferramenta e as forças e os binários exercidas pelo robô.

Todos estes dados citados são vistos pelo RobotStudio como variáveis. Porém, posição, forças e binários são variáveis do tipo vetor. Além disso, estas são adquiridas de forma ligeiramente diferentes.

A posição é uma variável vetorial adquirida diretamente pelo controlador a partir da posição instantânea do robô. Através de cálculos internos, o controlador as consegue ler e armazenar em uma variável interna denominada "*pos*" (*position*, posição). Da mesma forma o controlador realiza a leitura de tempo, que é uma variável puramente numérica.

Quanto às forças e binários, são lidas pelo controlador através do transdutor de força. Após isso, os dados são armazenados em um vetor de seis posições, denominado *FCforcevector*. Porém, o controlador possui funções específicas, também internas, que permitem a leitura das variáveis do vetor e separá-las componente a componente. Isso leva à criação de mais seis variáveis, uma para cada componente do vetor.

Outro detalhe importante quanto a isso é a determinação do tamanho do vetor. As leituras, como será visto posteriormente, serão realizadas a cada 0,1 segundo. Como o tempo de execução das furações em velocidades normais é baixo (poucos segundos), isso levaria a um vetor de, no máximo 100 ou 200 posições. Todavia, é também interessante obter dados do leitor na movimentação antes e depois do furo, para avaliação de ruídos e de trajetórias. Logo, para ter-se uma certa folga na escolha do tamanho dos vetores e ao mesmo tempo não sobrecarregar o sistema com excesso de leituras, o valor escolhido foi 2000.

Portanto, tem-se ao final desse processo as variáveis que podem ser vistas na figura 2.26.

```
VAR num time{2000};
VAR pos tcp_pos{2000};
VAR fcforcevector ForceVector;
VAR num xforce{2000};
VAR num yforce{2000};
VAR num zforce{2000};
VAR num xtorque{2000};
VAR num ytorque{2000};
VAR num ztorque{2000};
```

Figura 2.26 - Declaração das variáveis de processo

Além disso, outras variáveis serão importantes para que o robô complete a aquisição de dados, como veremos:

- ✓ Declaração das variáveis internas ao programa:

Essas "variáveis internas" são, na verdade, variáveis criadas para representar funções internas do RobotStudio. São essencialmente três: uma variável do tipo clock, que aqui será

chamada clock10, para a execução do relógio interno do controlador; uma variável do tipo intnum (interrupção, numérico) para realizar pausas do decorrer do programa para leitura das variáveis externas e que aqui será chamada interrupt; e uma variável numérica para realizar as contagens (que vão de zero a duas mil), que será chamada "cont".

Essas três variáveis (figura 2.27), em conjunto com as variáveis externas e com os *targets*, *tools* e *workobjects* definidos anteriormente, formam o conjunto de variáveis presente nos Anexos B, C e D.

```
VAR clock clock10;
VAR intnum interrupt;
VAR num cont:=0;
```

Figura 2.27 - Variáveis internas ao programa

- ✓ Programação de uma sub-rotina do tipo *trap*:

Para se realizar a leitura das variáveis durante os ensaios é necessário conectá-la a uma interrupção. E, de facto, o RobotStudio possui a capacidade de ligar uma interrupção à leitura de uma sub-rotina do tipo "*trap*". Assim, utilizando o argumento "CONNECT WITH" na rotina principal, a sub-rotina, aqui chamada "Write_data_to_array", será executada a cada interrupção feita. Essa sub-rotina pode ser vista na figura 2.28.

```
TRAP Write_data_to_array
    cont:=cont+1;
    time{cont}:=ClkRead(clock10);

    ForceVector:=FCGetForce(\WObj:=wobjdril\ContactForce);
    tcp_pos{cont}:=CPos(\Tool:=Spindle\wobj:=wobjdril);
    xforce{cont}:=ForceVector.xforce;
    yforce{cont}:=ForceVector.yforce;
    zforce{cont}:=ForceVector.zforce;
    xtorque{cont}:=ForceVector.xtorque;
    ytorque{cont}:=ForceVector.ytorque;
    ztorque{cont}:=ForceVector.ztorque;

ENDTRAP
```

Figura 2.28 - Programação da sub-rotina "trap"

Nela percebe-se a presença de um contador e de um *timer* para realizar as leituras em dado intervalo de tempo. Em seguida há a variável do tipo vetorial *ForceVector*, que armazena dados de leitura de força e binário através da função *FCGetForce* (que é uma função interna do controlador). Seus argumentos indicam que a leitura será realizada no referencial do *workobject* e que serão lidas as forças de contacto, ou seja, a ação da gravidade será desprezada. Da mesma forma, a variável vetorial *tcp_pos* armazena os dados de posição obtidos através da função *CPos*. E, por fim, observa-se também seis variáveis numéricas que armazenam, uma a uma, os valores das componentes do vetor ForceVector.

- ✓ Programação de um processo de escrita de variáveis:

Os dados lidos na rotina trap "Write_data_to_array" estão, até o momento, armazenados na memória do controlador, nas 2000 posições definidas anteriormente. Porém, é preciso registrá-las de alguma forma, a fim de que sejam lidas posteriormente.

No final da execução do programa é possível inserir um processo que efetua a gravação dos dados, em formato ".txt", no disco do controlador. Este processo, intitulado "Write_file", cria um ficheiro denominado "File_drill" na pasta "Home" do controlador. Também é necessário criar uma variável do tipo *iodev* (input/output device), para realizar a gravação. O programa gerado pode ser visto na figura 2.29.

```
PROC Write_file()

    VAR iodev file;
    open "HOME:"\File:="File_drill.txt",file;

    FOR cont FROM 1 TO 2000 DO

        write file,""\num:=cont;
        write file,""\pos:=tcp_pos{cont};
        write file,""\num:=time{cont};
        write file,""\num:=xforce{cont};
        write file,""\num:=yforce{cont};
        write file,""\num:=zforce{cont};
        write file,""\num:=xtorque{cont};
        write file,""\num:=ytorque{cont};
        write file,""\num:=ztorque{cont};

    ENDFOR

    Close file;

ENDPROC
```

Figura 2.29 - Programação do Processo de Escrita de Variáveis

- ✓ Execução da rotina principal:

Por fim, depois de definida cada sub-rotina do programa, é possível organizá-las em uma rotina principal, que será lida pelo controlador do robô durante a execução do programa. Esta rotina, denominada “main”, é organizada de acordo com a figura 2.30.

```
PROC main()

   FCCalib tdril_LD;

   ClkReset clock10;
   CONNECT interrupt WITH Write_data_to_array;
   ITimer 0.1,interrupt;
   ISleep interrupt;
   IWatch interrupt;
   ClkReset clock10;
   ClkStart clock10;

   drill;

   IDelete interrupt;
   ClkStop clock10;

   Write_file;

ENDPROC
```

Figura 2.30 - Programação da Rotina Principal

Nota-se nesta rotina principal alguns pontos importantes:

1. A calibração da ferramenta com os dados de *tdril_LD*;
2. A inicialização do clock interno antes da execução da rotina principal e das sub-rotinas que o utilizam e sua finalização ao fim dos movimentos;
3. A definição, a inicialização e a conexão da interrupção “interrupt” com a *trap* Write_data_to_array. Observação: o valor 0.1 corresponde ao intervalo de 0,1 segundos em que a interrupção é realizada;
4. A execução dos movimentos contidos no processo “drill” (ou *FC*drill, nos demais programas);
5. A escrita dos dados de processo através da sub-rotina “Write_file”.

2.2.6 – Programação em MatLab

Após terminada a programação do robô e do sistema de aquisição de dados, os dados coletados, em formato ".txt" podem ser lidos por diversos programas. O primeiro programa testado foi o Microsoft Excel. Porém, devido a dificuldades em manipular alguns dados e a problemas com lentidão na análise de informações, optou-se por recorrer ao *software* MathWorks MatLab.

Duas serão as funções deste programa neste trabalho. A primeira é relativa à manipulação dos dados gerados, pois o RobotStudio gera um ficheiro em que cada variável analisada é uma linha em texto. No caso, são nove linhas por leitura, sendo 10 o número de leituras por segundo. A primeira é a contagem de leituras do controlador do robô. A segunda linha é um vetor posição, que deve ser dividido em três componentes pelo MatLab. A terceira é a contagem de tempo. Da quarta à nona linhas estão os registros das forças e binários, respetivamente, captados pelo transdutor.

Portanto, para resolver esta primeira questão, foi preciso desenvolver um programa que faz a leitura do arquivo de texto (linha por linha), divida a segunda linha em três componentes e transforme cada conjunto de valores em uma coluna, a fim de facilitar a geração de gráficos. Além disso, o programa deve também realizar a derivação da posição, para que a velocidade calculada seja comparada com a velocidade programada. Tal programa encontra-se no Anexo A.

A segunda função do MatLab é para a geração de gráficos e visualização dos dados coletados. Isso é feito dentro do próprio programa, através das funcionalidades de visualização das tabelas geradas, dos valores máximos e mínimos encontrados e das funções plot 2-D, conforme podemos observar na figura 2.31. Nela também é possível ver as variáveis t e fz selecionadas (nesta ordem), assim como uma opção *plot* com as mesmas, na parte superior da figura. Através desse processo a visualização de gráficos torna-se mais rápida e eficaz.

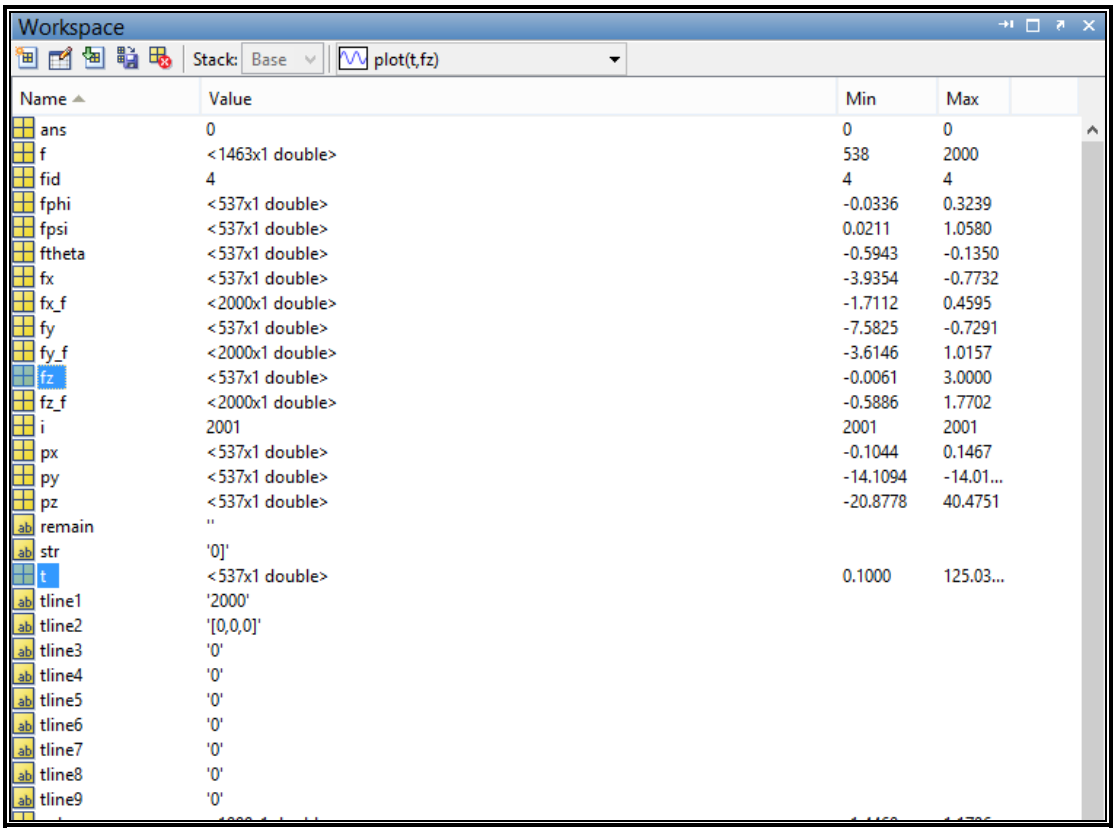

Figura 2.31 - Visualização do MatLab com os Dados Adquiridos

Portanto, com os programas RAPID já preparados e com os algoritmos para aquisição de dados criados, é possível, antes de transferir os arquivos para o controlador, realizar simulações no RobotStudio. Após isso, é possível iniciar os testes com o robô.

Capítulo 3

Ensaios e resultados

Todo o procedimento realizado até o momento tem como objetivo a avaliação das estratégias de controlo de força mencionadas no decorrer do trabalho. Todavia, é comum que problemas surjam devido aos mais diversos fatores. Portanto, para uma avaliação mais consistente, faz-se necessária a realização de ensaios que possam prever as adversidades no uso do controlo de força em furação.

Os experimentos realizados neste trabalho visam avaliar os programas RAPID obtidos, entender o significado dos parâmetros de programação e o que uma alteração destes pode provocar no processo de furação com o robô. A partir disso, pode-se estabelecer condições em que a aplicação do controlo de força é útil.

O capítulo é dividido em três partes. Na primeira parte são detalhados os procedimentos e testes realizados antes dos ensaios de furação. Na segunda parte, alguns cálculos e definições a respeito das furações a serem feitas. Somente na terceira parte os furos são efetuados. Quanto aos resultados, estes são comentados à medida que os ensaios são expostos.

Além disso, os ensaios serão realizados somente em madeira, visto que a escolha de apenas um material é suficiente para avaliação dos programas executados pelo robô e para a análise dos resultados.

3.1 – Procedimentos e Testes iniciais:

É evidente que os testes iniciam-se a partir da situação mais simples, ou seja, quando utilizada a programação em trajetória e velocidade mas sem a realização de furos. E, apesar de ser um ensaio muito simples, é essencial. Com o futuro acréscimo de forças e binários, esse primeiro teste será uma forma de comparar valores medidos.

Para tanto, programou-se uma trajetória equivalente a um furo de 10 mm de profundidade. A velocidade de trajetória utilizada foi 5 mm/s e o *spindle* esteve desligado durante toda a operação.

Como pode ser visto a seguir, os resultados foram satisfatórios para uma primeira análise e são de grande ajuda para avaliação comportamental do manipulador robótico quando este estiver submetido a forças de contacto. O gráfico 3.1 mostra simultaneamente trajetória, velocidade e forças representadas no eixo z do *workobject* (direção do furo executado).

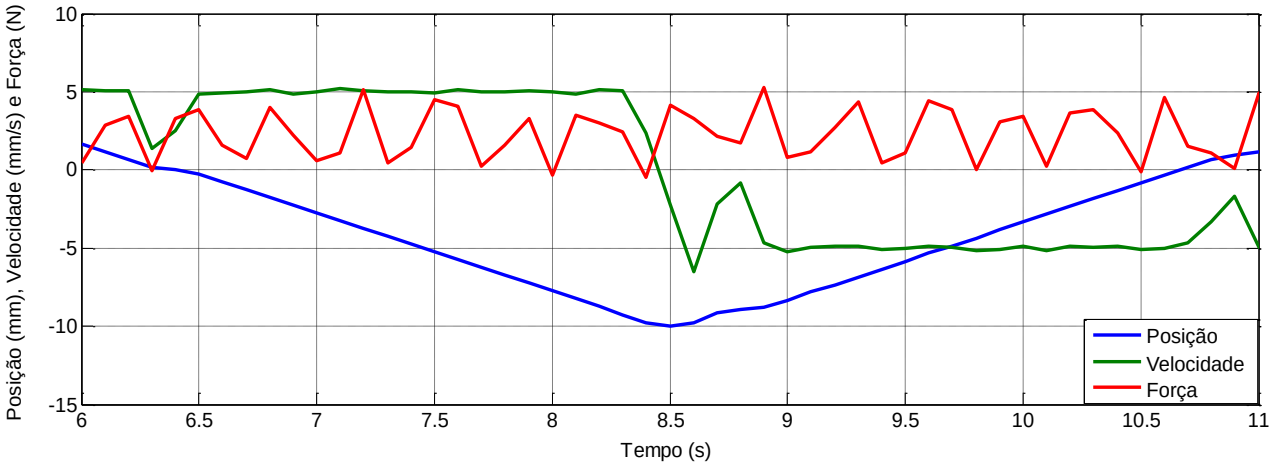

Gráfico 3.1 – Teste sem realização de furos

A partir do gráfico, pode-se observar que a trajetória segue um movimento bem linear de acordo com o tempo. Também é possível observar que esta atinge o ponto programado, equivalente a um furo de 10 mm de profundidade. Ao mesmo tempo, a velocidade é constante e sempre próxima do valor programado. Quanto ao sinal de força, há algum ruído presente, mas é possível perceber que a força aplicada permanece também constante.

Também é possível realizar uma análise das posições em x e y do *tool center point* em função do tempo. Estas são apresentadas no gráfico 3.2.

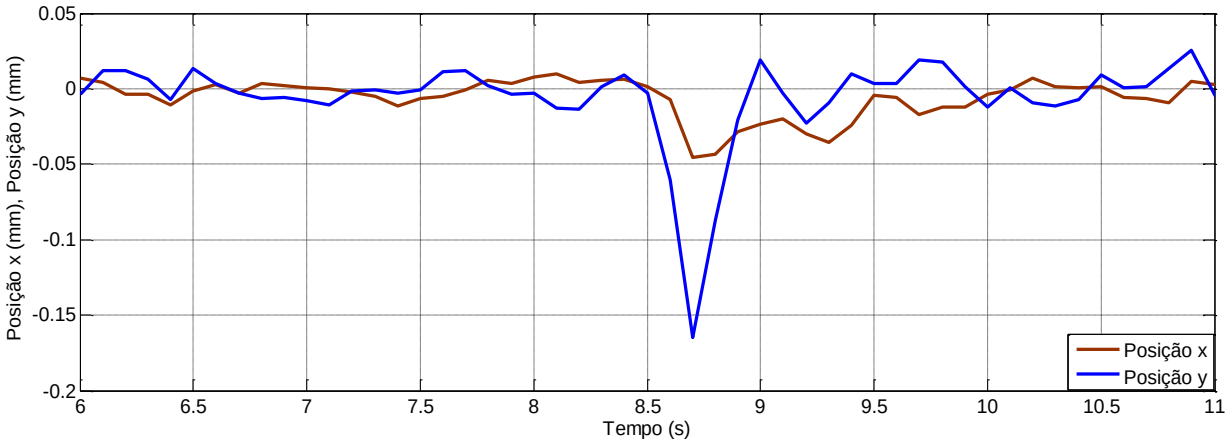

Gráfico 3.2 - Posição x e y no teste sem realização de furos

Porém, ainda que bem próximas de zero, observa-se um deslocamento de 0,15 mm no interior do furo, o que não é nada interessante para furação, especialmente de pequenos diâmetros. Por exemplo, para um furo de 1 mm de diâmetro, o valor equivale a 15% do total. Ou seja, um sistema robótico como o deste trabalho não é uma boa solução para operações de furação de pequenos diâmetros.

Além disso, antes de iniciar as furações, pensou-se em avaliar todos os programas gerados anteriormente em um material mais macio. Isso para garantir a segurança e integridade do robô. Inicialmente foi utilizada uma espuma, mas sem sucesso quando testado o sistema com controlo de força. Como a espuma possui baixa densidade, uma ínfima força é suficiente para furar o material. Dessa forma, o robô não é capaz ler valores adequados e qualquer pequena perturbação causa interferências nos testes.

Assim, uma solução encontrada para testar o controlo de força antes de furar materiais com maiores durezas foi o uso do dispositivo presente na figura 3.1.

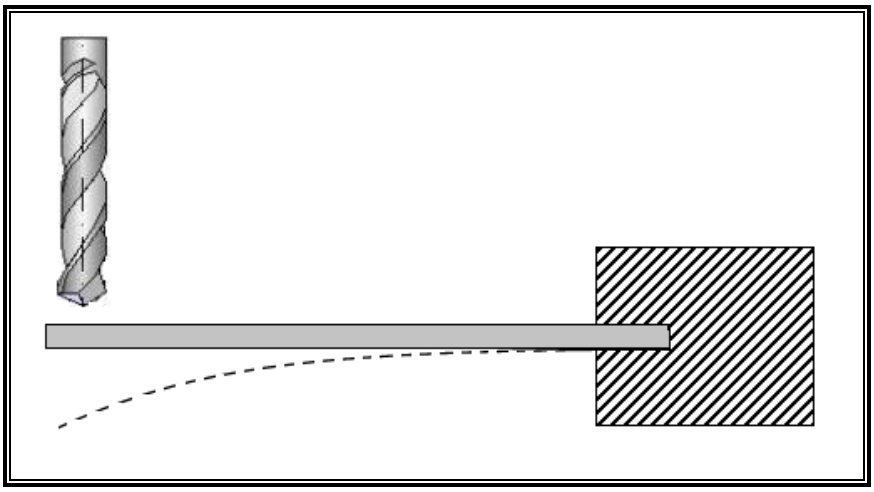

Figura 3.1 - Dispositivo para Testar o Controlo de Força

O dispositivo é composto de um material metálico flexível em formato de lâmina e uma fixação. Ele visa exercer uma força de resistência ao avanço do robô, que possui a broca em seu elemento terminal. Dessa forma, ao fazer o robô avançar perpendicularmente sobre o material – com o spindle desligado – a força tende a aumentar exponencialmente (devido ao surgimento de uma força elástica). Ao chegar ao valor de força definido no programa RAPID, ter-se-á a resposta do robô.

Os primeiros testes no dispositivo em questão foram realizados para avaliar o controlo de força *FC Speed Change*. De imediato, dois problemas foram encontrados:

1. O *FC Speed Change* realiza alterações de velocidades em níveis, de acordo com os valores de força. O padrão é haver dois valores de velocidade, 100% do configurado (se abaixo da força programada) ou 10% (se acima). Dessa forma, esta estratégia de controlo não apresenta vantagens em relação ao controlo de trajetória e velocidade. Outras configurações podem ainda ser alcançadas, mas muito limitadas, o que sugere que o *FC Speed Change* não é uma boa estratégia para furação;

2. Além disso, percebeu-se que a velocidade da trajetória é reduzida independentemente da direção da força. Ou seja, não é possível definir uma direção de controlo de força quando se utiliza o *FC Speed Change*.

Portanto, desde já foi decidido excluir o *FC Speed Change* dos demais ensaios, devido a sua inviabilidade em operações de furação. Entretanto, os testes continuam com o *FC Pressure*.

Ao utilizar-se o *FC Pressure* no dispositivo criado para os testes, observou-se que ele reage conforme o esperado. Os valores de força limite programados não foram ultrapassados e sempre que se chegava a esse valor, o robô deixava de avançar. Aproveitou-se também para

verificar a reposta do robô à forças nas outras direções que não fossem a direção de furação e esse permanecia em seu estado, sem reação.

3.2 – Definições e Cálculos:

Após os procedimentos de verificação dos programas gerados e sua utilização, é necessário fazer algumas definições e realizar alguns breves cálculos relativos à furação.

Em relação à broca, foi decidido não utilizar duas, mas apenas uma helicoidal de diâmetro 6 mm em aço rápido. Com o uso dessa broca nos ensaios, há uma diminuição do número de testes e, consequentemente, uma simplificação na análise dos resultados. Além disso, como visto, a precisão do robô torna-o impróprio para furos de diâmetros muito pequenos. Outras definições são vistas na seção seguinte.

Quanto às velocidades de corte para madeira, estas variam nas literaturas encontradas desde os 72 m/min aos 120 m/min. Ao escolher um valor intermediário de 90 m/min é possível calcular então a rotação necessária para realizar os furos.

O número de rotações por minuto, N, é dado por:

$$N = (\text{velocidade de corte} \times 1000) / (\text{diâmetro} \times \pi),$$

Onde o valor 1000 no numerador é indica uma conversão de unidades de medida.

Ou seja,

$$N = 90 \times 1000 / 6 \times 3{,}14$$

$$N = 4777 \text{ RPM}$$

Como os valores para furação são fornecidos em um intervalo e sempre são considerados apenas valores recomendados para iniciar uma furação, pode-se considerar a velocidade ideal para este caso um valor de 5000 RPM.

Já em relação ao avanço, foi estabelecido um valor de corte de 0,07mm/rev, que é aceitável para a madeira utilizada, para o diâmetro da broca e para o comprimento do furo, como pode-se conferir em [20].

Encontra-se:

$$\text{Avanço} = N \times \text{taxa de corte}$$

Assim,

$$\text{Avanço} = 5000 \times 0{,}07$$

Avanço = 350 mm/min ou 5,8 mm/s

Isso quer dizer que a velocidade de avanço do robô deve ser aproximadamente 5,8 mm/s. Mas, para simplificação dos testes, escolheu-se uma velocidade de 5mm/s. Esse valor condiz com o valor de velocidade utilizado no primeiro teste, quando o robô não realizava furo algum.

No caso do alumínio, as velocidades de corte também estão compreendidas em um largo intervalo de possibilidades. Um valor médio para furação com o uso de brocas de aço rápido é a velocidade 60 m/min. Utilizando as mesmas equações do caso da madeira, tem-se uma rotação em torno de 3000 RPM. Admitindo um valor de avanço de 0,15 mm/rev [21] e utilizando as equações anteriormente citadas, a velocidade de avanço será 450 mm/min, ou 7,5 mm/s.

O estabelecimento dos parâmetros de corte do compósito é um pouco mais complicado, visto que o material é alvo de análises primordialmente recentes. E como afirma Rubio, et al [22], por conta disso, há mesmo contradição entre autores acerca das melhores condições para furação. Portanto, admitindo valores de rotação elevados (4000 RPM e 8000 RPM) e um avanço lento (0,04 mm/rev), tem-se as velocidades de avanço aproximadamente 3 mm/s e 6 mm/s, respetivamente.

3.3 – Furações:

A última etapa deste capítulo consiste na furação através de duas estratégias de controlo. A primeira é o controlo em trajetória e velocidade, a segunda em força, através do *FC Pressure*.

Os furos realizados neste trabalho, que são todos feitos em madeira, são furos cegos. Escolheu-se este tipo de furo para que se pudesse melhor avaliar a capacidade de posicionamento dentro de uma peça durante o controlo de força. O comprimento dos furos escolhidos foi 10 mm e 18 mm, a serem combinados com a broca de 6 mm. As velocidades de furação iniciais são baseados nos cálculos da seção 3.2 deste relatório e os valores de força testados são abordados caso a caso.

Quanto à disposição física do material durante os ensaios, a madeira foi fixada na mesa posicionadora do robô através de uma interface, também de madeira, e duas morsas, de forma a garantir a ausência de movimentos e minimizar as vibrações durante a furação.

De posse disso, é possível iniciar-se os ensaios. O primeiro deles com uma profundidade programada de 10 mm, velocidade 5 mm/s, rotação de 5000 RPM e o robô controlado em trajetória e velocidade. Os valores obtidos pela aquisição de dados são mostrados no gráfico 3.3 e o valor zero do eixo vertical (posição, velocidade e força) indica a superfície da peça maquinada, a velocidade nula e a força nula, respetivamente.

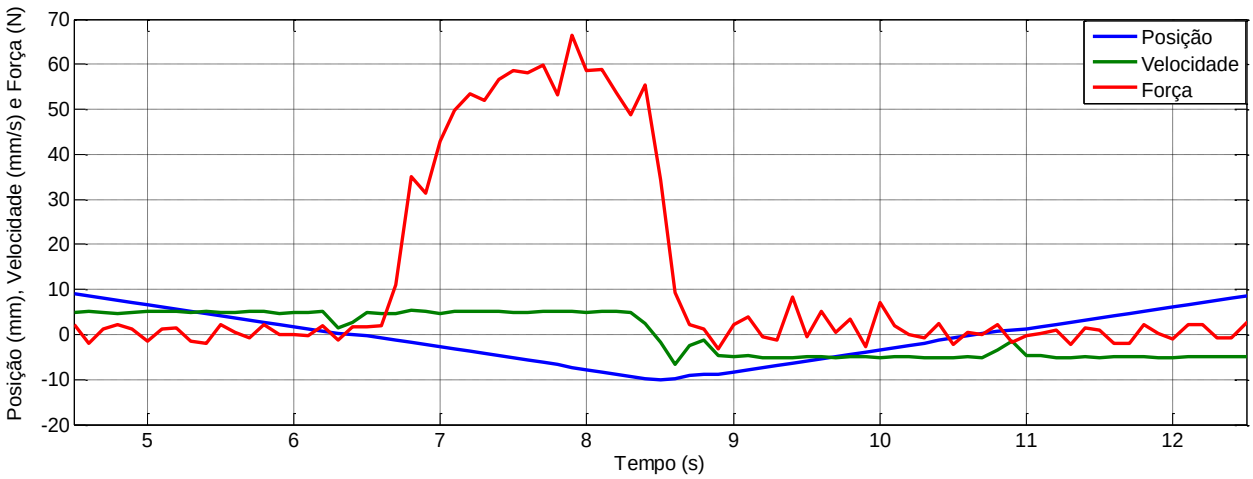

Gráfico 3.3 - Furação de madeira (10 mm) sem controlo de força

Observa-se a partir do gráfico que a trajetória programada foi alcançada sem dificuldades. O furo foi da posição zero até a -10 m e os movimentos de ida e volta demoraram aproximadamente 4 segundos, o que condiz com a velocidade programada. Assim, o robô não encontrou problemas alcançar o ponto programado e o presente gráfico é muito semelhante, em termos de posição, ao gráfico relativo ao avanço do robô sem realização de furos.

Porém, no eixo z, as forças chegam ao valor de 60 N durante a furação, o que demonstra que o manipulador robótico realiza um certo esforço para furar a peça de madeira. E a partir desses valores de força obtidos é possível estabelecer valores de força para iniciar os experimentos com furações em que o controlo de força esteja ativo.

Um dos valores testados, próximo aos 60 N encontrados no teste em que não havia controlo de força, foi o uso de uma limitação de força de 50 N durante a furação. Porém, para este valor de força, os pontos programados não foram alcançados, conforme é visto no gráfico 3.4.

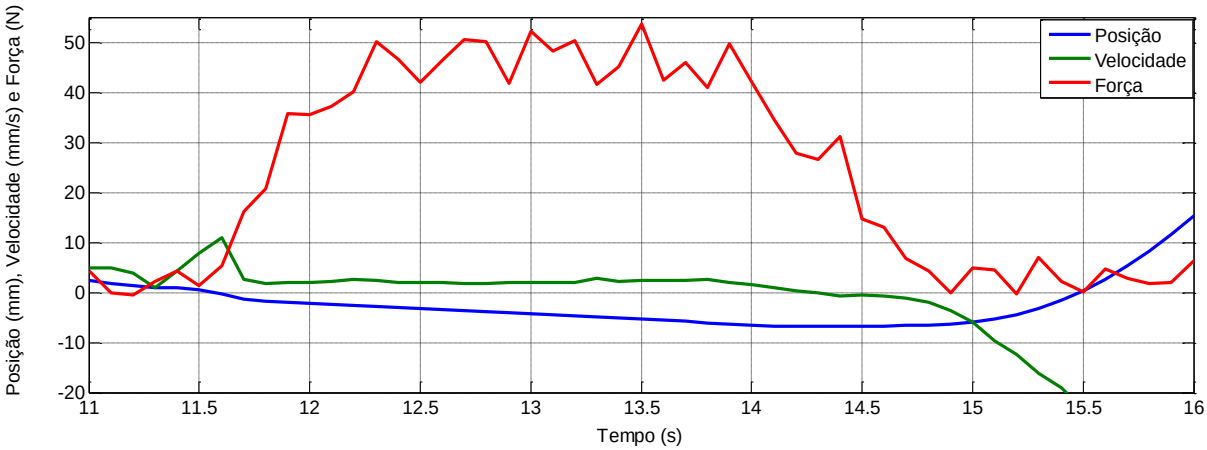

Gráfico 3.4 - Furação de madeira (10 mm) com FC Pressure de 50 N

Ou seja, uma força constante de 50 N não consegue realizar o mesmo que foi realizado quando na ausência de controlo de força. Além disso, outros valores de força próximos foram utilizados, mas em nenhum caso chegou-se aos exatos 10 mm de profundidade. O que ocorre, independente dos valores de força testados, é uma total ausência de exatidão na profundidade do furo. Porém, o número de testes realizados é pouco para que se chegue a alguma conclusão referente ao FC Pressure.

Dessa forma, com o objetivo de fazer uma melhor análise do comportamento do *FC Pressure* e descobrir o que realmente ocorre em sua aplicação, realizou-se um segundo tipo de testes com a madeira. Agora os furos são um pouco mais longos, 18 mm, e rotação permanece constante, em 5000 RPM. Para o ensaio sem controlo de força, novamente a trajetória foi alcançada e com boa precisão, conforme pode ser visto no gráfico 3.5.

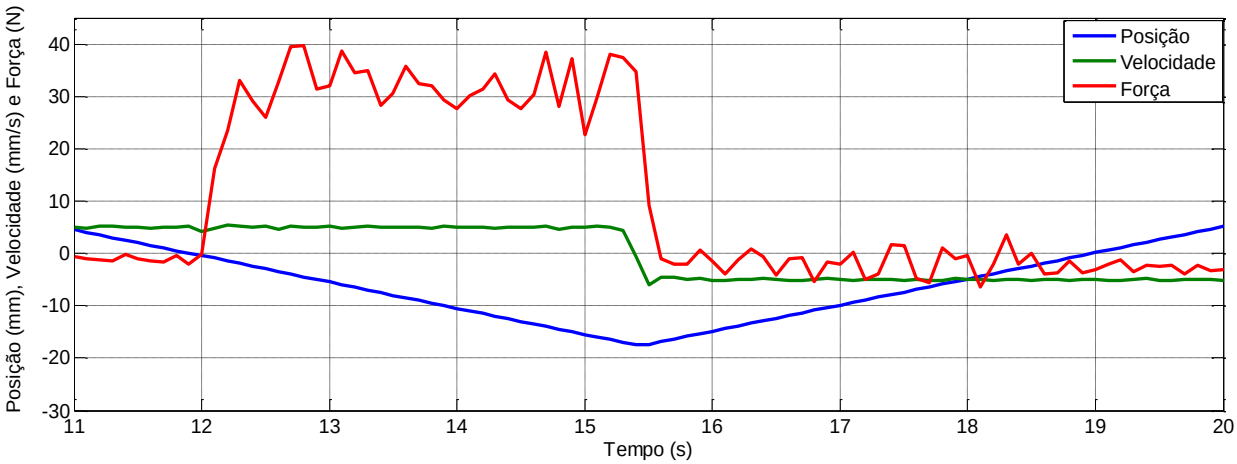

Gráfico 3.5 - Furação de madeira (18 mm) sem FC e velocidade 5 mm/s

Novamente, sem controlo de força, a posição é alcançada e a força não ultrapassa os 40 N. Porém, ao invés de repetir os ensaios de controlo de força da mesma forma que no caso anterior, busca-se agora outras soluções, a partir da manipulação de alguns dos parâmetros de furação. O primeiro desses é a velocidade da trajetória. Através de uma redução da velocidade pela metade, obtém-se o gráfico 3.6.

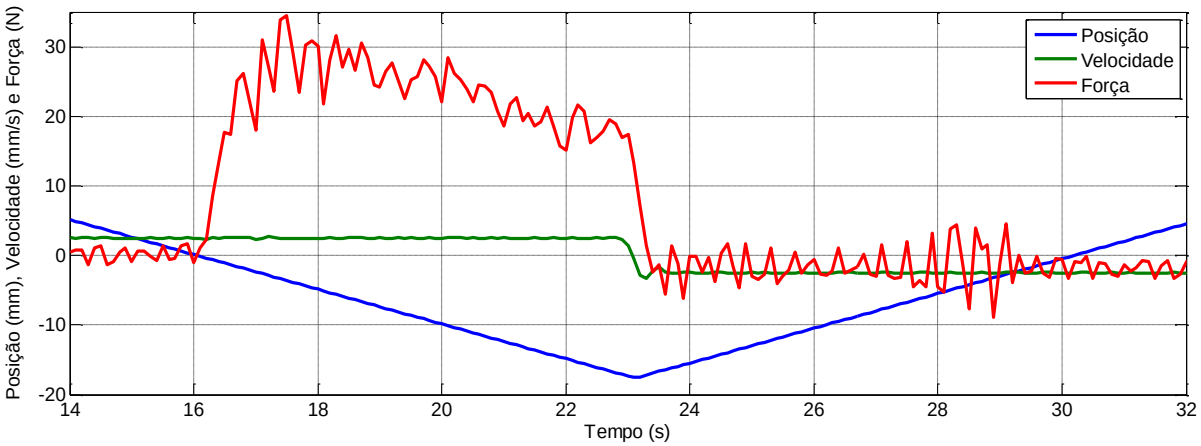

Gráfico 3.6 – Furação de madeira (18 mm) sem FC e velocidade 2,5 mm/s

Percebe-se então que, além do cumprir o posicionamento programado, há uma mudança nos valores de força exercidos pelo robô. As forças exercidas para uma velocidade menor são também menores. Portanto, é possível analisar um pouco mais essa questão através de uma nova redução de velocidades. O resultado dessa nova variação está contido no gráfico 3.7.

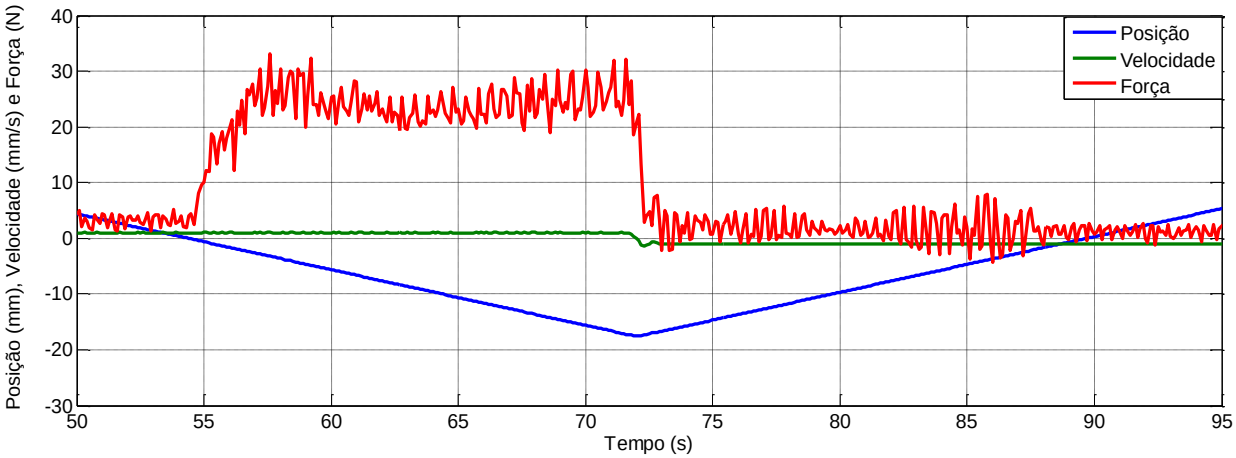

Gráfico 3.7 – Furação de madeira (18 mm) sem FC e velocidade 1 mm/s

Neste último caso, em que a velocidade é alterada, percebe-se que a variação de forças nos gráficos é muito pequena quando comparada aos anteriores. Porém, é possível confirmar que a velocidade de furação está, de alguma forma, ligada às forças exercidas durante o furo.

Assim, pode-se novamente testar o controlo de força. Mas ao invés de realizar modificações nos valores de força, serão alteradas as velocidades de furação estabelecidas no programa RAPID para o *FC Pressure*. Os resultados deste novo teste são exibidos no gráfico 3.8. As forças de contacto foram limitadas ao valor de 30 N.

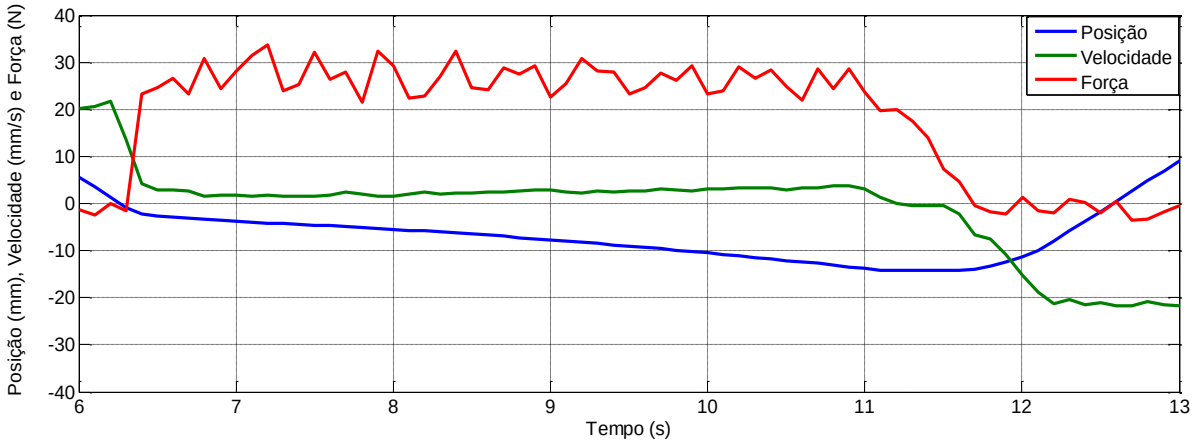

Gráfico 3.8 - Furação com controlo de força (30 N) e velocidade 5 mm/s

Um segundo ensaio foi realizado em condições semelhantes, a fim de comparação. Nele, todos os parâmetros permaneceram iguais, exceto a velocidade de avanço, 10 mm/s. O gráfico 3.9 mostra os resultados deste novo teste.

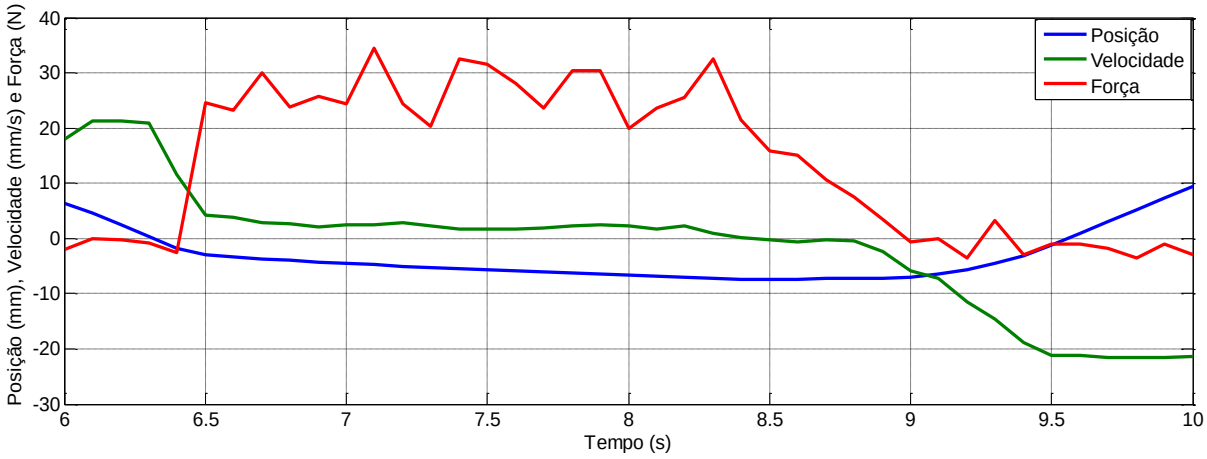

Gráfico 3.9 - Furação com controlo de força (30 N) e velocidade 10 mm/s

É possível observar, ao comparar o gráfico 3.8 e o gráfico 3.9, que o comportamento do robô com o controlo *FC Pressure* tem uma evolução distinta. Percebe-se isso com o resultado das furações. Nos gráficos, as posições alcançadas no interior do furo são completamente diferentes, mesmo que em ambos os casos a programação tenha sido um furo de 18 mm de profundidade. Porém, as forças exercidas e as velocidades efetivas durante o processo são muito parecidas. Apenas um fator foi realmente diferente entre os casos: a velocidade de avanço.

No gráfico 3.8 o tempo de furação foi o dobro que o registrado no gráfico 3.9. Entretanto, no processo de programação em momento algum é pedido para determinar o tempo de processo. Portanto, chegou-se à conclusão, após a realização de outros ensaios intermediários, que o *FC Pressure* calcula o tempo de furação de acordo com a velocidade e os pontos programados. Assim, o controlo de força permanece ativo o exato tempo que se demoraria para realizar um furo caso não houvesse controlo de força. Ou seja, como no gráfico 3.9 a velocidade calculada foi duas vezes maior que no gráfico 3.8, o tempo de furação é duas vezes menor. Isso faz com que o robô fique menor tempo controlado em força na furação e, consequentemente, um furo de menor profundidade é realizado.

O maior problema em relação a isso é o facto de não ser possível, com a utilização do *FC Pressure*, realizar furos com profundidades bem definidas. Entretanto, o controlo de força, em

si, foi realizado com sucesso, visto que manteve uma força de contacto programada praticamente constante.

Outro problema em relação ao *FC Pressure* está no facto de o fabricante não detalhar seu funcionamento no manual, mas apenas o seu uso. Isso limita a aplicação do controlo de força entre os usuários e impede, ou dificulta, o desenvolvimento de novas aplicações, como foi feito e verificado neste trabalho.

Capítulo 4

Conclusões e Trabalhos Futuros

O trabalho em questão possui como objetivo o estudo, a implementação e a análise de uma célula robótica controlada em força para furações. De facto, isto foi realizado, ainda que as soluções encontradas não sejam as melhores imagináveis. Na verdade, a partir do trabalho realizado, foi possível entender um pouco mais sobre a utilização de robôs em operações de furação e perceber que este é um campo pouco explorado e que muito deve ser aperfeiçoado.

Por exemplo, no caso do *FC Speed Change,* esta estratégia de controlo de força criada pela ABB não é uma opção interessante quando se realiza operações de furação pelo facto de retardar o processo (criando níveis muito baixos de velocidades) e pela impossibilidade de controlar-se forças em direções distintas. Ao mesmo tempo, o *FC Pressure*, que não possui tais problemas, é capaz de realizar um furo limitando e mantendo a força programada independentemente da velocidade de avanço estabelecida. Contudo, resulta que não é possível garantir a profundidade do furo a realizar.

Para garantir uma dada profundidade do furo enquanto se mantém o controlo de força *FC Pressure* poderia ser usado um esbarro mecânico. Este esbarro permitiria gerar uma força de contacto que limitaria o avanço do robô. Outra possibilidade é a escrita de uma sub-rotina no programa RAPID a fim de controlar posição em paralelo com a força. Assim, quando o *Tool Center Point* do robô, por exemplo, chegasse a uma determinada posição espacial, este poderia interromper a movimentação e finalizar o processo.

Ainda quanto à programação de robôs e sua utilização em furação, verificou-se que se trata de algo realmente possível e interessante. Obviamente há algumas limitações, tanto para o robô (que não suporta exercer forças muito elevadas) quanto para a precisão dimensional da localização e diâmetro do furo (que um robô pode não ser capaz de fornecer). Mas é uma aplicação promissora e que pode desenvolver-se muito mais. Além disso, outros detalhes, como o acoplamento ao *spindle* ser realizado de forma rígida, evitando o uso do sistema de acoplamento pneumático, pode melhorar o desempenho do robô ao efetuar a furação.

E ao tocar nestes assuntos, pode-se também estabelecer alguns trabalhos futuros, de forma a complementar o que até aqui foi estudado.

Em primeiro lugar, há uma necessidade de realizar-se mais testes, a fim de compreender melhor as limitações do robô e das estratégias de controlo estudadas. Estes testes devem envolver, ao mesmo tempo, uma maior manipulação dos parâmetros de furação, dos parâmetros de programação e também das peças furadas.

Quanto aos parâmetros de furação, pode ser útil testar furos com maiores velocidades de rotação e menores valores de avanço. Quanto à programação, seria bom uma análise mais detalhada da interferência dos inúmeros parâmetros que neste trabalho não foram explorados. E quanto às peças, pode-se buscar novas disposições geométricas ou mesmo novos materiais a serem furados, a fim de explorar ainda mais as capacidades do robô para furação.

Outros pontos mais avançados podem ser explorados futuramente, como a elaboração de algoritmos de controlo que visam realizar funções mais específicas ou simplesmente que o fabricante do robô não disponibiliza. Também no âmbito das pesquisas avançadas, uma melhoria enorme seria o desenvolvimento de sistemas mecânicos que viessem a colaborar com a rigidez do robô.

Referências

1. Stoeterau, Rodrigo Lima. *Usinagem com Ferramentas de Geometria Definida*. Texto de apoio às aulas. Escola Politécnica da Universidade de São Paulo, pp 49;

2. Stone, R. & Krishnamurthy, K. 1996. *A Neural Network Thrust Force Controller to Minimize Delamination During Drilling of Graphite-Epoxy Laminates*. International Journal of Machine Tools and Manufacture, Vol. 36, No. 9, pp. 985 - 1003;

3. Cantwell, W. J. & Morton, J. *The Impact Resistance of Composite Materials – A Review*. Composites, Vol. 22, September 1991, pp. 347 – 362;

4. Durão, Luís Miguel Pereira; Tavares, João Manuel R.S.; Marques, A. Torres; Magalhães, A. Gonçalves; Figueiredo, Miguel, 2004. *Estudo da Furação de Laminados Carbono/Epóxido com Diferentes Brocas*. Instituto Superior de Engenharia do Porto & Faculdade de Engenharia da Universidade do Porto;

5. Isaksson, Robert, 2009. *Drilling with Force Feedback*. Linköpings Universitet;

6. Alici, Gürsel, 1999. *A Systematic Approach to Develop a Force Control System for Robotic Drilling*, Industrial Robot: An International Journal, Vol. 26, No. 5, pp. 389 - 397;

7. Waurrzyniak, Patrick. *Aerospace Automation Pick Up the Pace*. Artigo para a Manufacturing Engineering Magazine, March 2013, pp. 55 - 62;

8. Aziz, Mohd Hazny & Ayub, Muhammad Azmi. *Measurement of Forces and Torques during Non-Homogeneous Material Drilling Operation*. Journal of Mechanics Engineering and Automation, July 2011, pp. 139 - 146;

9. ABB, 2007 - 2010. *Force Control for Machining*. Application Manual, Revision F;

10. Abreu, P, 2012. *Manual de Utilização - RobotStudio 5.14, Parte I – Introdução*. Faculdade de Engenharia da Universidade do Porto. Textos de apoio às aulas;

11. ABB. *ABB Test Viewer*. Manual do Programa, Version 1.3;

12. ABB, 2012. *IRB 2400 Industrial Robot*. Data Sheet, Review 7;

13. Santana, Fábio. *Metrologia*. CEFET Santa Catarina. Texto de apoio às aulas;

14. PDS, 2011. *Automatic Tool Spindle Change – Operation Manual*. Manual do Programa;

15. ATI, 2007. *Multi-Axis Force/Torque Sensor*. Catálogo do produto.

16. Informações retiradas do website *http://www.madeidura.com*, acessado pela última vez em 27 de junho de 2013;

17. Souza, André J. 2011. *Processos de Fabricação por Usinagem - Parte 2*. Universidade Federal do Rio Grande do Sul. Texto de apoio às aulas;

18. Abreu, P, 2012. *Manual de Utilização - RobotStudio 5.14, Parte II – Programação I*. Faculdade de Engenharia da Universidade do Porto;

19. ABB, 2007. *Introduction to RAPID*. Operating Manual;

20. Filho, Fernando. C. L. 2004. *Análise da Usinagem de Madeira Visando a Melhoria de Processos em Indústrias de Móveis*. Universidade Federal de Santa Catarina. Tese de doutorado.

21. Informações retiradas do website *http://www.daycounter.com/,* acessado pela última vez em 2 de julho de 2013;

22. Rúbio, J. C. Campos et al, 2007. *Furação com Alta Velocidade de Corte em Compósitos Poliméricos Reforçados com Fibras de Vidro*. Ciência e Tecnologia dos Materiais, Vol. 19, No 3/4, pp. 88 - 87.

Anexos

Anexo A:

Programa (Matlab) utilizado na análise dos dados gerados pelo robô:

```
clc
clear

% to open file and to read data:

fid = fopen('File_drilling_traj.txt');

i=1;
while ~feof(fid)

    tline1 = fgetl(fid);
    tline2 = fgetl(fid);
    tline3 = fgetl(fid);
    tline4 = fgetl(fid);
    tline5 = fgetl(fid);
    tline6 = fgetl(fid);
    tline7 = fgetl(fid);
    tline8 = fgetl(fid);
    tline9 = fgetl(fid);

% position data:

remain=tline2;
[str, remain] = strtok(remain, ',');
px(i)=str2num(str(2:end));
[str, remain] = strtok(remain, ',');
py(i)=str2num(str);
[str, remain] = strtok(remain, ',');
pz(i)=str2num(str(1:end-1));

% time data:

t(i)=str2num(tline3);

% force and torque data:

fx(i)=str2num(tline4);
fy(i)=str2num(tline5);
fz(i)=str2num(tline6);
fpsi(i)=str2num(tline7);
ftheta(i)=str2num(tline8);
fphi(i)=str2num(tline9);

i=i+1;
end

% derivation to find linear velocity in x axis:

vel_x=diff(px)/0.1;

% derivation to find linear velocity in y axis:
```

```
vel_y=diff(py)/0.1;

% derivation to find linear velocity in z axis:

vel_z=diff(pz)/0.1;

% matrix transposition:

t=t';
px=px';
py=py';
pz=pz';
fx=fx';
fy=fy';
fz=fz';
fpsi=fpsi';
ftheta=ftheta';
fphi=fphi';
vel_x=vel_x';
vel_y=vel_y';
vel_z=vel_z';

FClose(fid);

% stop reading:

f=find(t==0);
t(f)=[];
px(f)=[];
py(f)=[];
pz(f)=[];
fx(f)=[];
fy(f)=[];
fz(f)=[];
fpsi(f)=[];
ftheta(f)=[];
fphi(f)=[];
```

Anexo B:

Modelo de programa RAPID utilizado para furação sem controlo de força:

```
MODULE Module1

    PERS tooldata Spindle:=[TRUE,[[-80.2112,-200.062,177.681],[0.979924798,0,0,-
    0.199367478]],[12.1714,[31.7967,52.538,131.753],[1,0,0,0],0,0,0]];
    PERS wobjdata wobjdril:=[FALSE,TRUE,"",[[1030,30,476],[0,-0.707107,0.707107,0]],[[0,0,0],[1,0,0,0]]];

    VAR loaddata tdril_LD:=[11.9215,[27.855,49.547,130.396],[1,0,0,0],0,0,0];

    CONST robtarget aprox1:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget aprox2:=[[0,0,-3],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget aprox3:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget process1:=[[0.0,0,0],[0,0,-
0.706965,0.707249],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget process2:=[[0,0,10],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget withdraw1:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget withdraw2:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

    VAR pos tcp_pos{2000};
    VAR num xforce{2000};
    VAR num yforce{2000};
    VAR num zforce{2000};
    VAR num xtorque{2000};
    VAR num ytorque{2000};
    VAR num ztorque{2000};
    VAR num time{2000};
    VAR clock clock10;
    VAR intnum interrupt;
    VAR num cont:=0;
    VAR FCforcevector ForceVector;

    PROC main()

        FCCalib tdril_LD;

        ClkReset clock10;
        CONNECT interrupt WITH Write_data_to_array;
        ITimer 0.1,interrupt;
        ISleep interrupt;
        IWatch interrupt;
        ClkReset clock10;
        ClkStart clock10;

        drill;

        IDelete interrupt;
        ClkStop clock10;

        Write_file;

    ENDPROC

    PROC drill()

        MoveJ aprox1,v10,fine,Spindle\WObj:=wobjdril;
        MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;
        MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;
        MoveL process1,v10,fine,Spindle\WObj:=wobjdril;
```

```
    MoveL process2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;
    MoveJ aprox1,v10,z1,Spindle\WObj:=wobjdril;

  ENDPROC

  TRAP Write_data_to_array
    cont:=cont+1;
    time{cont}:=ClkRead(clock10);

    ForceVector:=FCGetForce(\WObj:=wobjdril\ContactForce);
    tcp_pos{cont}:=CPos(\Tool:=Spindle\wobj:=wobjdril);
    xforce{cont}:=ForceVector.xforce;
    yforce{cont}:=ForceVector.yforce;
    zforce{cont}:=ForceVector.zforce;
    xtorque{cont}:=ForceVector.xtorque;
    ytorque{cont}:=ForceVector.ytorque;
    ztorque{cont}:=ForceVector.ztorque;

  ENDTRAP

  PROC Write_file()

    VAR iodev file;
    open "HOME:"\File:="File_drill.txt",file;

    FOR cont FROM 1 TO 2000 DO

      write file,""\num:=cont;
      write file,""\pos:=tcp_pos{cont};
      write file,""\num:=time{cont};
      write file,""\num:=xforce{cont};
      write file,""\num:=yforce{cont};
      write file,""\num:=zforce{cont};
      write file,""\num:=xtorque{cont};
      write file,""\num:=ytorque{cont};
      write file,""\num:=ztorque{cont};

    ENDFOR

    Close file;

  ENDPROC

ENDMODULE
```

Anexo C:

Modelo de programa RAPID utilizado para furação com controlo de força *FC Speed Change*:

```
MODULE Module1

  PERS tooldata Spindle:=[TRUE,[[-80.2112,-200.062,177.681],[0.979924798,0,0,-
  0.199367478]],[12.1714,[31.7967,52.538,131.753],[1,0,0,0],0,0,0]];
  PERS wobjdata wobjdril:=[FALSE,TRUE,"",[[1030,30,476],[0,-0.707107,0.707107,0]],[[0,0,0],[1,0,0,0]]];

  VAR loaddata tdril_LD:=[11.9215,[27.855,49.547,130.396],[1,0,0,0],0,0,0];

  CONST robtarget aprox1:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  CONST robtarget aprox2:=[[0,0,-3],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  CONST robtarget aprox3:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  CONST robtarget process1:=[[0.0,0,0],[0,0,-
0.706965,0.707249],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  CONST robtarget process2:=[[0,0,10],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  CONST robtarget withdraw1:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  CONST robtarget withdraw2:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

  VAR pos tcp_pos{2000};
  VAR num xforce{2000};
  VAR num yforce{2000};
  VAR num zforce{2000};
  VAR num xtorque{2000};
  VAR num ytorque{2000};
  VAR num ztorque{2000};
  VAR num time{2000};
  VAR clock clock10;
  VAR intnum interrupt;
  VAR num cont:=0;
  VAR FCforcevector ForceVector;

  PROC main()

    FCCalib tdril_LD;

    ClkReset clock10;
    CONNECT interrupt WITH Write_data_to_array;
    ITimer 0.1,interrupt;
    ISleep interrupt;
    IWatch interrupt;
    ClkReset clock10;
    ClkStart clock10;

    FCdrill;

    IDelete interrupt;
    ClkStop clock10;

    Write_file;

  ENDPROC

  PROC FCdrill()

    MoveJ aprox1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;
```

```
    MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;

    FCSpdChgAct 50\NonStopAllTime;

    MoveL process1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL process2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;

    FCSpdChgDeact;

    MoveJ aprox1,v10,z1,Spindle\WObj:=wobjdril;

  ENDPROC

  TRAP Write_data_to_array
    cont:=cont+1;
    time{cont}:=ClkRead(clock10);

    ForceVector:=FCGetForce(\WObj:=wobjdril\ContactForce);
    tcp_pos{cont}:=CPos(\Tool:=Spindle\wobj:=wobjdril);
    xforce{cont}:=ForceVector.xforce;
    yforce{cont}:=ForceVector.yforce;
    zforce{cont}:=ForceVector.zforce;
    xtorque{cont}:=ForceVector.xtorque;
    ytorque{cont}:=ForceVector.ytorque;
    ztorque{cont}:=ForceVector.ztorque;

  ENDTRAP

  PROC Write_file()

    VAR iodev file;
    open "HOME:"\File:="File_drill.txt",file;

    FOR cont FROM 1 TO 2000 DO

      write file,""\num:=cont;
      write file,""\pos:=tcp_pos{cont};
      write file,""\num:=time{cont};
      write file,""\num:=xforce{cont};
      write file,""\num:=yforce{cont};
      write file,""\num:=zforce{cont};
      write file,""\num:=xtorque{cont};
      write file,""\num:=ytorque{cont};
      write file,""\num:=ztorque{cont};

    ENDFOR

    Close file;

  ENDPROC

ENDMODULE
```

Anexo D:

Modelo de programa RAPID utilizado para furação com controlo de força *FC Pressure*:

```
MODULE Module1

  TASK PERS tooldata Spindle:=[TRUE,[[-80.2112,-200.062,177.681],[0.979924798,0,0,-
  0.199367478]],[12.1714,[31.7967,52.538,131.753],[1,0,0,0],0,0,0]];
  PERS wobjdata wobjdril:=[FALSE,TRUE,"",[[1030,30,476],[0,-0.707107,0.707107,0]],[[0,0,0],[1,0,0,0]]];

  TASK PERS loaddata tdril_LD:=[11.9215,[27.855,49.547,130.396],[1,0,0,0],0,0,0];

  VAR robtarget aprox1:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  VAR robtarget aprox2:=[[0,0,-3],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  VAR robtarget aprox3:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  VAR robtarget process1:=[[0.0,0,0],[0,0,-0.706965,0.707249],[0,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
  VAR robtarget process2:=[[0,0,10],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  VAR robtarget withdraw1:=[[0,0,-1],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
  VAR robtarget withdraw2:=[[0,0,-30],[0,0,-0.707106781,0.707106781],[0,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

  VAR num n1ForceX:=0;
  VAR num n1ForceY:=0;
  VAR num n1ForceZ:=30;

  VAR pos tcp_pos{2000};
  VAR num xforce{2000};
  VAR num yforce{2000};
  VAR num zforce{2000};
  VAR num xtorque{2000};
  VAR num ytorque{2000};
  VAR num ztorque{2000};
  VAR num time{2000};
  VAR clock clock10;
  VAR intnum interrupt;
  VAR num cont:=0;
  VAR FCforcevector ForceVector;

  PROC main()

    FCCalib tdril_LD;

    ClkReset clock10;
    CONNECT interrupt WITH Write_data_to_array;
    ITimer 0.1,interrupt;
    ISleep interrupt;
    IWatch interrupt;
    ClkReset clock10;
    ClkStart clock10;

    FCdrill;

    IDelete interrupt;
    ClkStop clock10;

    Write_file;

  ENDPROC
```

```
PROC FCdrill()

    FCCalib tdril_LD;

    MoveL aprox1,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox2,v10,z1,Spindle\WObj:=wobjdril;
    MoveL aprox3,v10,z1,Spindle\WObj:=wobjdril;

    FCPress1LStart
    process1,v10\Fx:=n1ForceX\Fy:=n1ForceY\Fz:=n1ForceZ,100\ForceFrameRef:=FC_REFFRAME_WOBJ\ForceChang
    e:=10\DampingTune:=100\TimeOut:=1\PosSupvDist:=9e9,fine,Spindle\WObj:=wobjdril;
    FCPressL process2,v10,30,fine,Spindle,\wobj:=wobjdril;
    FCPressLEnd withdraw1,v10\ForceChange:=50\ZeroContactValue:=10;
    MoveL withdraw2,v10,fine,Spindle\WObj:=wobjdril;
    MoveL aprox1,v10,fine,Spindle\WObj:=wobjdril;

ENDPROC

TRAP Write_data_to_array
    cont:=cont+1;
    time{cont}:=ClkRead(clock10);

    ForceVector:=FCGetForce(\WObj:=wobjdril\ContactForce);
    tcp_pos{cont}:=CPos(\Tool:=Spindle\wobj:=wobjdril);
    xforce{cont}:=ForceVector.xforce;
    yforce{cont}:=ForceVector.yforce;
    zforce{cont}:=ForceVector.zforce;
    xtorque{cont}:=ForceVector.xtorque;
    ytorque{cont}:=ForceVector.ytorque;
    ztorque{cont}:=ForceVector.ztorque;

ENDTRAP

PROC Write_file()

    VAR iodev file;
    open "HOME:"\File:="File_drill.txt",file;

    FOR cont FROM 1 TO 2000 DO

        write file,""\num:=cont;
        write file,""\pos:=tcp_pos{cont};
        write file,""\num:=time{cont};
        write file,""\num:=xforce{cont};
        write file,""\num:=yforce{cont};
        write file,""\num:=zforce{cont};
        write file,""\num:=xtorque{cont};
        write file,""\num:=ytorque{cont};
        write file,""\num:=ztorque{cont};

    ENDFOR

    Close file;

ENDPROC

ENDMODULE
```