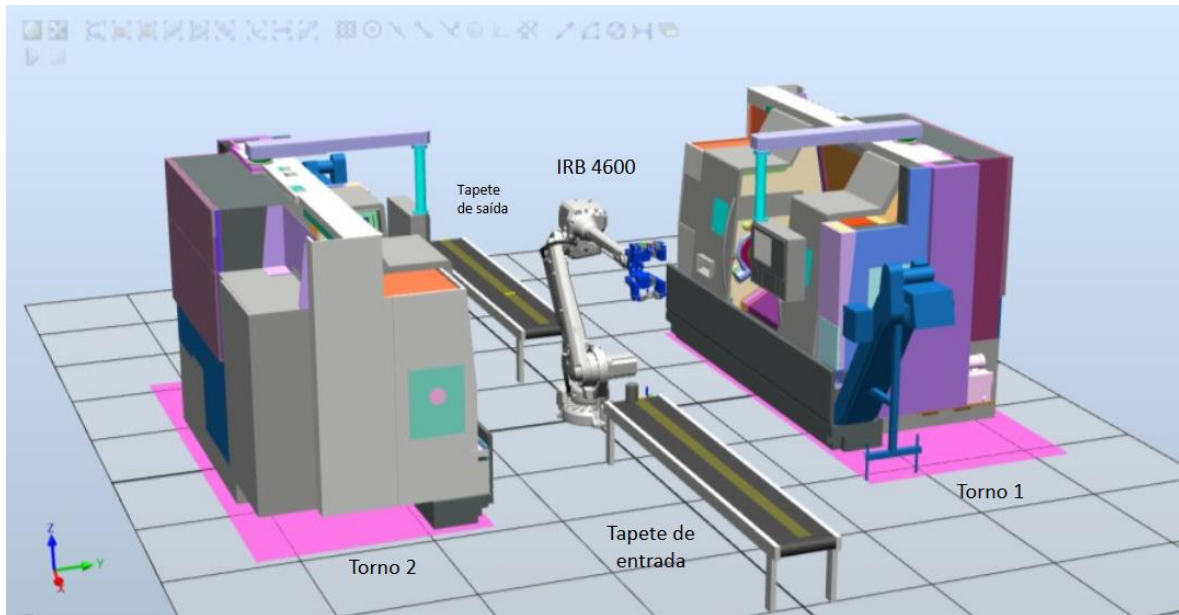


# Fundamentos de Robótica Industrial

## Trabalho 1



Trabalho realizado por:

André Grilo nº 46440

Valter Francisco nº 39383

Professor Francisco Campos

## Contents

1.	Nota Introdutória .....	3
2.	<i>Targets</i> .....	4
3.	<i>Paths</i> .....	5
4.	<i>Rapid e Simulation</i> .....	7

## 1. Nota Introdutória

Neste trabalho utilizamos o software RobotStudio da ABB para simular uma estação de trabalho com o objectivo de processar peças em tornos.

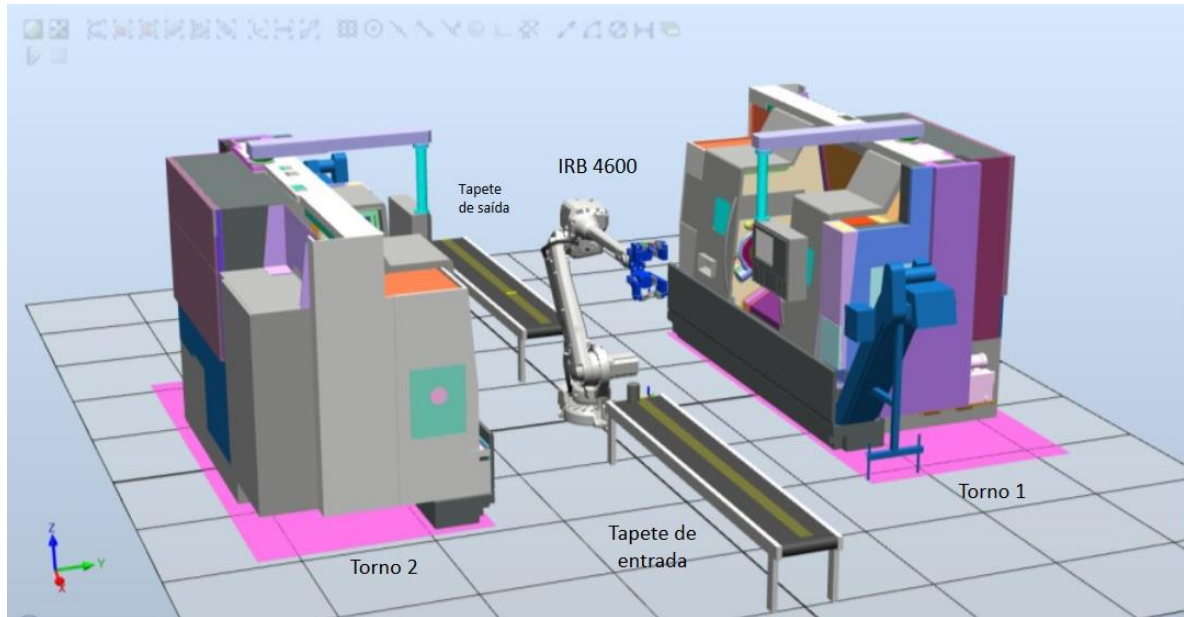


Imagem 1 - Estação de Trabalho

A imagem a cima mostra a estrutura da nossa estação, composta por dois tapetes rolantes, um de entrada de peças e um de saída, dois tornos que podem ser usados independentemente ou simultaneamente e um braço robótico com duas garras para movimentação de peças.

Inicialmente vamos processar 10 peças utilizando apenas um torno e uma garra.

Depois vamos processar 10 peças utilizando um torno mas as duas garras, em que uma leva a peça por processar e troca pela peça já processada.

Por último vamos processar 10 peças utilizando os dois tornos e só uma das garras.

O objectivo é comparar quanto os tempos que cada um destes processos demora e tirar conclusões quanto ao mais vantajoso.

Para isso temos que começar por fazer os *targets*, os *paths* entre targets e programar o processo em *rapid*.

## 2. Targets

Para criar um *target* levamos o robô até ao ponto desejado e utilizamos a funcionalidade *teach target*. Como exemplo temos a figura a baixo que mostra o nosso primeiro *target* em que o robô pega na peça que vem do tapete de entrada.

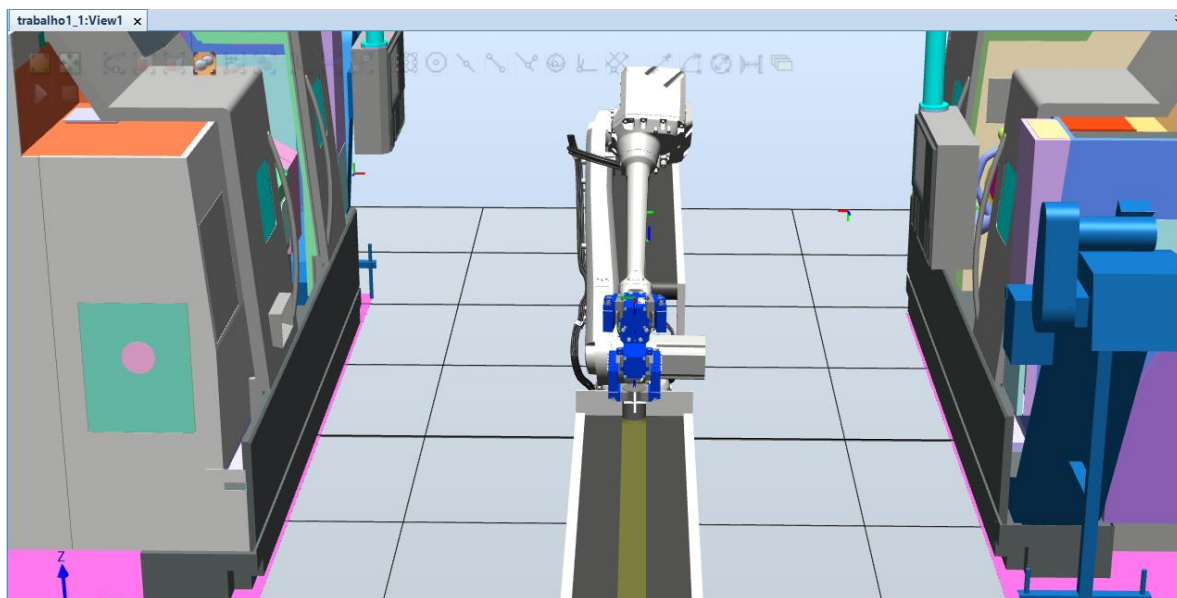


Imagem 2 - Criação do primeiro target

Para criar mais *targets* temos várias opções, utilizadas dependendo da necessidade, a primeira foi descrita anteriormente, a outras passam por copiar *targets* já criados e aplicar movimentos de translação, rotação ou ambos.

O número de *targets* a criar são tantos quantos o que acharmos necessários retirando os obrigatórios como é o caso dos *targets* que envolvem pegar e largar as peças nos tapetes bem como os de colocar e retirar as peças dos tornos. É importante criar *targets* em pontos intermédios para posterior criação de *paths* de forma a que o robô não faça uma trajetória que possa entrar em colisão com qualquer parte da estação.

Para criarmos targets usando a segunda garra duplicámos o target, no caso de retirar a peça do torno por exemplo, trocámos a ferramenta a usar, aplicámos uma rotação de 180° e fizemos pequenas translações para ajustar a garra de forma a ter a certeza que iria pegar na peça sem entrar em colisão com o torno.

### 3. Paths

Um *path* basicamente é um deslocamento do robô entre dois ou mais *targets* definidos anteriormente.

Para criar um *path* utilizamos a funcionalidade *path* e arrastamos os *targets* pretendidos para o *path* criado.

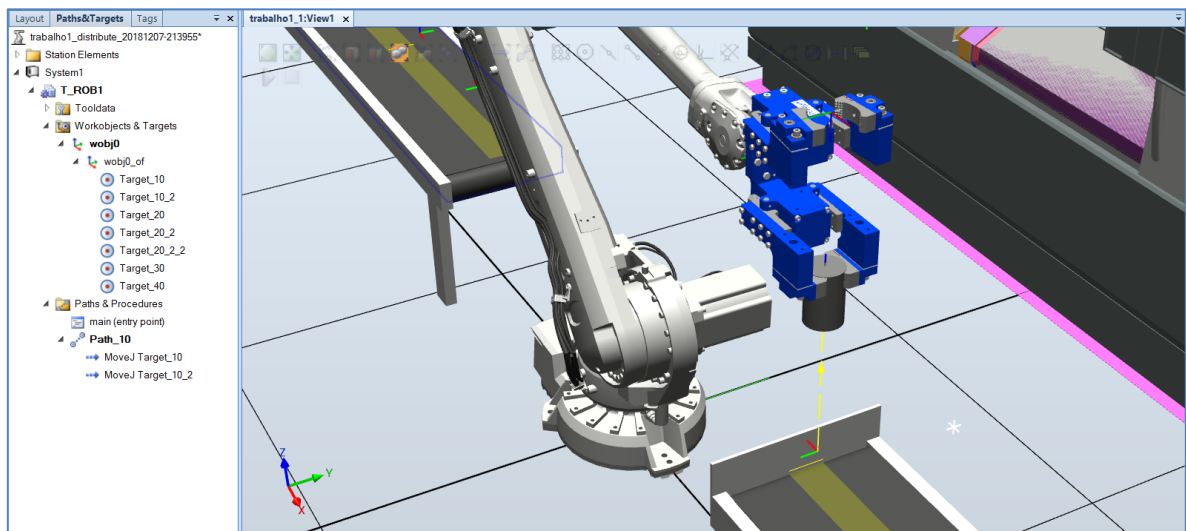


Imagem 3 - Exemplo de *path*

Na imagem acima temos o exemplo de um *path*, representado pela linha amarela a tracejada. Este representa o retirar a peça do tapete e levar até uma posição intermédia que também pode ser usada como posição de repouso.

Os *paths* podem ser movimentações lineares (*MoveL*) ou movimentações de *jog joint* (*MoveJ*), o primeiro faz a trajectória linear o segundo procura a forma mais “confortável” de movimento do robô aproximando-se o máximo possível a uma trajectória linear.

O aspeto final da nossa estação de trabalho, com todos os *paths* e *targets* está apresentado na figura abaixo.

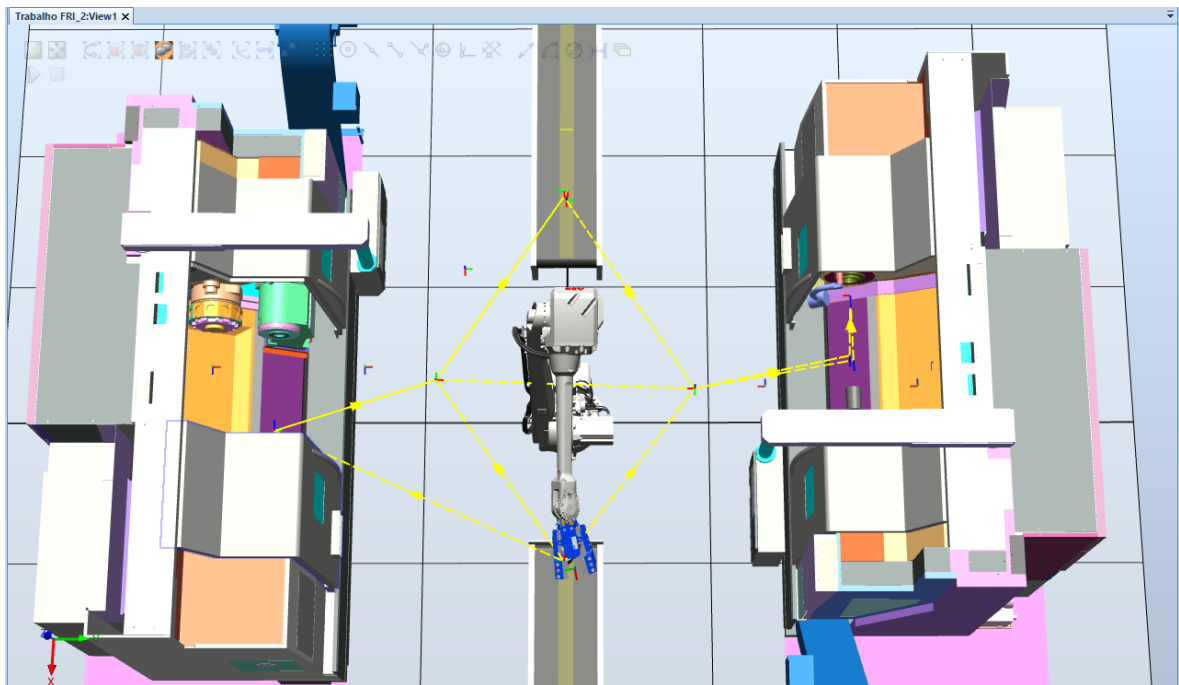


Imagem 4 - Estação de trabalho

## 4. *Rapid e Simulation*

Depois de todos os *targets* e *paths* criados iniciamos a programação em *rapid*.

O primeiro passo é sincronizar o *rapid* com a estação de trabalho utilizando a funcionalidade *synchronize* para que as informações sobre as nossas constantes, que são os *target* e os *paths*, sejam automaticamente transpostas para o *rapid*, estas informações passam pelas coordenadas, posição da garra, posição do robô, tipo de movimento, etc.

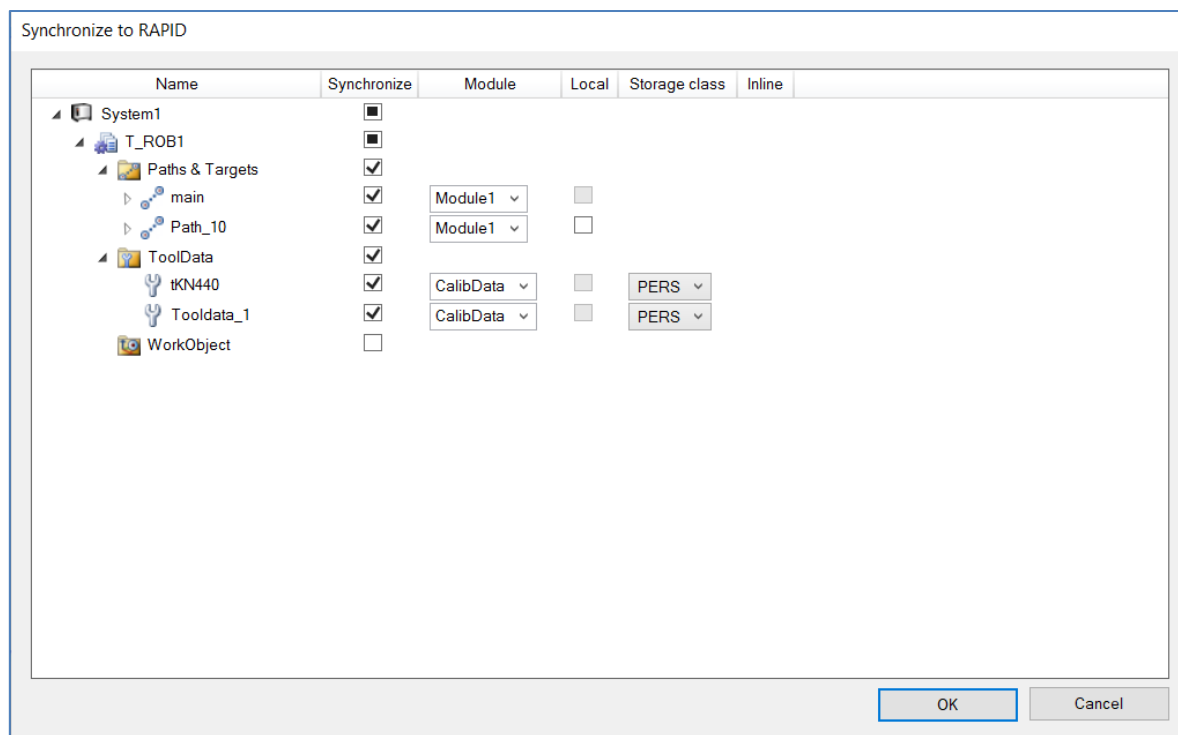
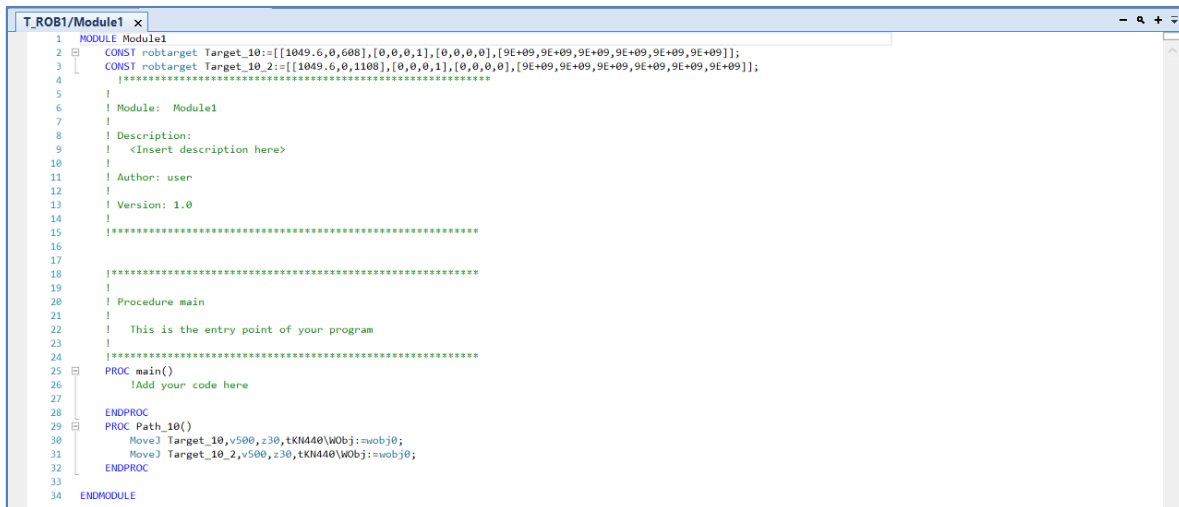


Imagem 5 - Janela *synchronize*

Depois de sincronizar o *rapid* fica com o seguinte aspecto.



```
1 MODULE Module1
2   CONST robtarget Target_10:=[[1049.6,0,600],[0,0,0,1],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09]];
3   CONST robtarget Target_10_2:=[[1049.6,0,1100],[0,0,0,1],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09]];
4   !*****
5   !
6   ! Module: Module1
7   !
8   ! Description:
9   ! <Insert description here>
10  !
11  ! Author: user
12  !
13  ! Version: 1.0
14  !
15  !*****
16
17
18  !*****
19  !
20  ! Procedure main
21  !
22  ! This is the entry point of your program
23  !
24  !*****
25  PROC main()
26    !Add your code here
27
28  ENDPROC
29  PROC Path_10()
30    MoveJ Target_10,v500,z30,tKM440\Wobj:=wobj0;
31    MoveJ Target_10_2,v500,z30,tKM440\Wobj:=wobj0;
32  ENDPROC
33
34 ENDMODULE
```

Imagem 6 – Rapid

Como se pode ver na imagem acima os *targets* e *paths* são automaticamente inseridos na linha de comandos.

O passo seguinte do *rapid* é introduzir o sequenciamento das acções. Então é necessário reflectir sobre o que necessitamos fazer inicialmente, que passa por pegar na peça transportada pelo tapete de entrada, levá-la até ao torno, esperar que o torno transforme a peça, retirá-la do torno e transportá-la para o tapete de saída e repetir o processo para 10 peças retirando o tempo que este processo demorou.

Então começamos por introduzir o *path* que corresponde a ir buscar a peça ao tapete de entrada e colocar no final de cada linha de código “;” sempre.

Para pegar na peça é necessário fazer a garra fechar, introduzindo “*set gripper1;*” para a primeira garra ou “*set gripper2;*” para a segunda, para as abrir substituímos “*set*” por “*reset*”. Pela leitura do enunciado sabemos que as garras demoram 0.2 segundos a abrir/fechar.

Considerando a linha anterior, para termos a certeza que a garra fecha depois do robô fazer a trajectória para o *target* desejado introduzimos outra linha de código: “*WaitTime 1;*”. Esta linha faz com que o robô esteja parado durante 1 segundo (tempo de espera) após ter feito o *path* desejado. Só depois do tempo de espera é que dizemos ao robô para fechar a garra e transportá-la para o torno.

Para deixar a peça no torno fazemos o mesmo que no parágrafo anterior, utilizando “*reset gripper 1;*” para abrir a garra quando a peça está no torno, retiramos o robô do torno e iniciamos o ciclo do torno utilizando “*startLathe1;*” ou “*startLathe2;*” dependendo de qual vai ser o torno a utilizar. Introduzimos um tempo de espera que corresponde ao tempo que o torno está a processar a peça.

Para sabermos quanto tempo demora um torno a processar uma peça utilizámos a ferramenta *Stopwatch* que se encontra na parte da simulação, começando a contar o tempo quando o torno inicia o processo e parando de contar o tempo quando a porta volta a abrir.



Station Signals		Stopwatch
		<b>Add</b>
<div> <div>⬆</div> <div>Stopwatch</div> <div>✕</div> </div>		
Name:	Stopwatch	
Start Trigger:	I/O Value	
Source:	Lathe_LB3000	
I/O Signal:	startCycle	
Value:	1	
End Trigger:	I/O Value	
Source:	Lathe_LB3000	
I/O Signal:	doorOpened	
Value:	1	
Count:	0	
Total Time:	0	
Average Time:	0	

Imagem 7 - Instruções dadas ao Stopwatch

O *Total Time* obtido foi 10.836 segundos e para garantir que o robô não entra em colisão com a porta do torno introduzimos um tempo de espera de 11 segundos e só depois dizemos ao robô para retirar a peça e levá-la até ao tapete de saída.

Tendo tudo isto em consideração escrevemos o modelo em *rapid*, que ficou com o seguinte aspeto. Este modelo foi utilizado para os dois tornos e uma garra, há que notar que não existe uma linha de comando para iniciar o trabalho do torno pois estava a acontecer um erro que parava a simulação, mas como o tempo de funcionamento do torno já estava determinado temos a certeza que não iria influenciar o resto da simulação.

```
T_ROB1/Module1 X
39  CONST robtarget Target_60_2:=[[23.087828463,853.377131572,1055.500401113],[0.708406115,0.044366825,-0.04957458,-0.702662595],[1,3,2,1],[9E+09,9E+09,9E+09,9E+09]];
40  CONST robtarget Target_100:=[[-1454,0,580],[1,0,0,0],[1,2,0,0],[9E+09,9E+09,9E+09,9E+09]];
41  CONST robtarget Target_110:=[[-1381.083094122,11.947471194,971.386774847],[0.004987196,0.998857206,-0.013899099,0.04545574],[3,-4,-1,1],[9E+09,9E+09,9E+09,9E+09]];
42  CONST robtarget TTT:=[[-214.817997579,1888.306419676,1001.028990724],[0.508650092,-0.508650112,0.491197599,-0.491197583],[0,1,1,0],[9E+09,9E+09,9E+09,9E+09]];
43
44  PROC main()
45
46  reset gripper1;
47  path_1;
48  waittime 1;
49  set gripper1;
50  path_2;
51  waittime 1;
52  reset gripper1;
53  path_3;
54  path_1;
55  waittime 1;
56  set gripper1;
57  path_2;
58  waittime 1;
59  set gripper2;
60  path_120;
61  waittime 4;
62  reset gripper1;
63  path_7;
64  waittime 6;
65  reset gripper2;
66  path_130;
67  path_40;
68  waittime 1;
69  set gripper1;
70  path_7;
71  waittime 1;
72  reset gripper1;
73  path_130;
74  path_1;
75
76  ENDPROC
77
```

Imagem 8 - Modelo *rapid*

E voltando a repetir este processo 10 vezes em simulação e retirámos que o tempo necessário para efectuar este processo foi 440 segundos, aproximadamente 7 minutos e 20 segundos.

Para respondermos à primeira pergunta do enunciado temos que utilizar as duas garras. Assim dizemos ao robô para ir buscar a primeira peça, colocá-la no torno e em vez de esperar os 11 segundos o robô vai buscar a segunda. É aqui que a segunda garra começa a ser usada pois é esta que retira a peça do torno e colóca-se a segunda peça para ser processada este processo demora cerca de 8 segundos pois demos ao robô 5 segundos para fazer a rotação do braço. Quando o robô sai do torno, para levar a primeira peça para o tapete de saída, inicia-se o processo de transformação da segunda peça. Depois de pousar a primeira peça no tapete de saída retoma-se o ciclo para 10 peças.

O tempo que este processo demorou foi 350 segundos que é aproximadamente 5 minutos e 50 segundos.

Em seguida utilizámos dois tornos e uma garra. Começamos o processo como os anteriores, dizendo ao robô para colocar a primeira peça no primeiro torno. Depois da primeira peça estar a ser transformada dizemos ao robô para pegar na segunda peça e colocar no segundo torno. Não é necessário esperar que primeira peça seja transformada pois o tempo de movimento do robô é superior ao de funcionamento do torno então dizemos ao robô para levar a peça para o tapete de saída, pegar na peça seguinte e voltar a colocar no primeiro torno. Por esta altura o segundo torno já acabou de processar a segunda peça e dizemos ao robô para retirar a peça, colocar no tapete de saída e pôr outra peça a ser processada.

Repetindo este processo para 10 peças o que demorou 571 segundos que corresponde a aproximadamente 9 minutos e meio.

Por último utilizámos duas garras e dois tornos, juntando um pouco dos dois pontos anteriores, criámos os *targets* próximos do segundo torno para a segunda garra, criámos novos *paths* incluindo os novos *targets* e demos início à codificação do *rapid*. Inicialmente o robô vai buscar a primeira peça, coloca no primeiro torno, quando o robô sai para ir buscar a segunda peça, o torno começa a transformar a peça. O robô transporta a segunda peça para o segundo torno e quando sai deste inicia-se a transformação. Depois disto o robô vai buscar a terceira peça e espera 5 segundos enquanto faz a rotação do braço para proceder à troca as peças do torno 1, levando a primeira peça já transformada para o tapete de saída. Em seguida vai buscar a quarta peça para substituir pela segunda peça que já está transformada no segundo torno, faz a rotação, substitui as peças e leva a segunda para o tapete de saída, retomando este ciclo até 10 peças terem sido transformadas.

Este processo demorou 359 segundos que corresponde aproximadamente a 6 minutos.

## 5. Conclusão

Utilizar uma garra e um torno demorou 7 minutos e 20 segundos a processar 10 peças.

Utilizar duas garras e um torno demorou aproximadamente 5 minutos e 50 segundos a processar 10 peças.

Utilizar uma garra e dois tornos demorou aproximadamente 9 minutos e meio a processar 10 peças.

Utilizar duas garras e dois tornos demorou aproximadamente 6 minutos a processar 10 peças.

Utilizar apenas uma garra e dois tornos foi o processo que demorou mais tempo, depois foi uma garra e um torno, seguidamente duas garras e dois tornos e o mais rápido foi duas garras e um torno.

Ao utilizarmos duas garras em vez de uma poupámos 1 minuto e 30 segundos.

Ao utilizarmos dois tornos em vez de um demorou 1 minuto e 40 segundos a mais.

Ao utilizarmos duas garras e dois tornos em vez de duas garras e um torno demorou 10 segundos a mais a processar 10 peças.

Ao utilizarmos duas garras e dois tornos em vez de uma garra e dois tornos ganhámos 3 minutos.

Considerações adicionais no que diz respeito ao tempo que cada processo demorou: o tempo total poderia diminuir com o aumento da velocidade do robô especialmente em trajetórias lineares em que não existisse perigo de colisão, também no ajuste da posição de alguns *targets* intermédios, criados com o intuito e reduzir o perigo de colisão, para pontos que diminuíssem os movimentos que o robô tinha que fazer como também os tempos adicionais de rotação do braço para usar a segunda garra poderia ser ajustado ou talvez até poderia ter sido feito durante o movimento reduzindo o tempo que esses processos demoraram.

Provavelmente pelas razões apresentadas acima é que o processo em que utilizámos duas garras e um torno demorou menos tempo que o que utilizámos duas garras e dois tornos.

Utilizarmos dois tornos em relação a um só não compensa, pois, os tempos de deslocamento do robô são maiores, logo temos mais tempo desperdiçado.

Utilizarmos duas garras em vez de uma reflete sempre num melhoramento significativo dos tempos de processo.