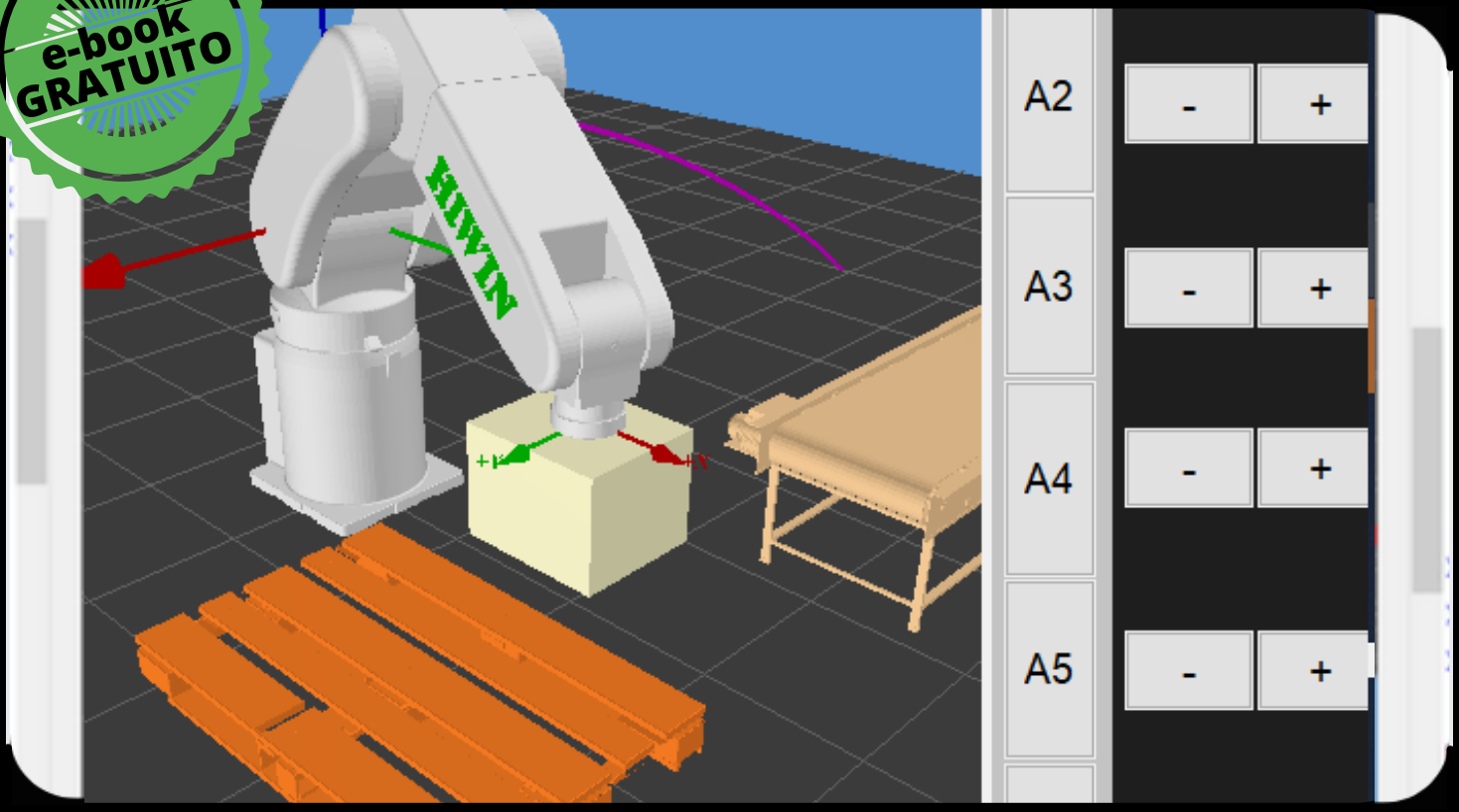
ROBÓTICA INDUSTRIAL

Conhecendo o Simulador HRSS

JOSÉ SOARES BATISTA LOPES
JÚLIO CESAR DE ALMEIDA FREITAS

e-book GRATUITO



VÍDEOS NO YOUTUBE PARA FACILITAR A APRENDIZAGEM.

2021

JOSÉ SOARES BATISTA LOPES, DR.
PROFESSOR DO CURSO DE MECATRÔNICA DO INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE - CAMPUS PARNAMIRIM

JÚLIO CESAR DE ALMEIDA FREITAS, MSC.
PROFESSOR DO CURSO DE PÓS-GRADUAÇÃO DE INDÚSTRIA 4.0 E NA GRADUAÇÃO EM TECNOLOGIA EM MECATRÔNICA INDUSTRIAL NO SENAI - ARMANDO DE ARRUDA PEREIRA

ROBÓTICA INDUSTRIAL

CONHECENDO O SIMULADOR HRSS

Prof. Dr. José Soares Batista Lopes
IFRN - Campus Parnamirim
Rua Antônia de Lima Paiva, 155 - Bairro Nova Esperança, Parnamirim - CEP: 59143-455
jose.soares@ifrn.edu.br / mecatronicanapratica

Prof. Msc. Júlio Cesar de Almeida Freitas
SENAI - Armando de Arruda Pereira
Rua Santo André, 680 - Boa Vista - São Caetano do Sul/SP
CEP 09572-000
julio.freitas@sp.senai.br / jucaf1100@gmail.com

2021

DIAGRAMAÇÃO
José Soares Batista Lopes
CAPA
José Soares Batista Lopes
ILUSTRAÇÕES
canva.com
ÍCONES DA CAPA
canva.com

FICHA CATALOGRÁFICA
Catalogação da publicação na fonte elaborada pelo Bibliotecário
Cícero Filho Tavares – CRB: 15/511

Lopes, José Soares Batista.
L864r Robótica Industrial: conhecendo o simulador HRSS /José Soares Batista Lopes. – Parnamirim, 2021.
80 p. : il. Collor ;

ISBN: 978-65-00-32480-8

1.Robótica Industrial. 2. Software. 3. Simulador - HRSS. I Lopes, José Soares Batista. II. Freitas, Júlio Cesar de Almeida. III. Título. Batista. II. Titulo.

CDU 621.865.8

Edição eletrônica: E-books
Linha Editorial: Tecnologias (Ciências aplicadas)
Disponível para download em:
http://memoria.ifrn.edu.br

Lista de Figuras

Pag.

Lista de Figuras

Pag.

Lista de Figuras

Pag.

Agradecimentos

Agradeço à Deus por tudo, a minha família pela paciência e compreensão no desenvolvimento desta obra, a todos os colegas e amigos que de alguma forma contribuíram e ao meu parceiro o Prof. Júlio pelos conhecimentos compartilhados.

Desejo aos leitores sucesso na leitura e na carreira profissional como Robotistas. Espero que essa obra contribua no aprendizado como estudantes e/ou profissionais.

Prof. Dr. José Soares Batista Lopes

Em primeiro lugar agradeço a Deus por esta oportunidade de compartilhar o que de graça foi-me dado, em segundo lugar entender que os caminhos que traçamos são respostas as nossas preces. Agradecer também ao fabricante de robô HIWIN por disponibilizar está valiosa ferramenta de forma livre, para o estudo e o entendimento. E para concluir não poderia deixar de exaltar o trabalho ímpar do amigo e professor Soares, pela iniciativa e determinação que conduziu este trabalho com a simplicidade e leveza sem pretensão de edificar um marco, mas simplesmente organizar e distribuir a certeza de uma orientação direcionadas aqueles que buscam entender e desenvolver está senda do conhecimento.

Prof. Msc. Júlio Cesar de Almeida Freitas

Apresentação

Este material tem o objetivo de orientar as práticas iniciais da disciplina Introdução a Robótica Industrial. O Software HRSS (Hiwin Robotic System Software) da HIWIN auxilia o aluno na programação off-line de rotinas de programação de movimentos, ou seja, sem a necessidade física de um manipulador industrial para a prática, auxiliando o processo de aprendizagem da disciplina, diminuindo assim as dificuldades do ensino remoto.

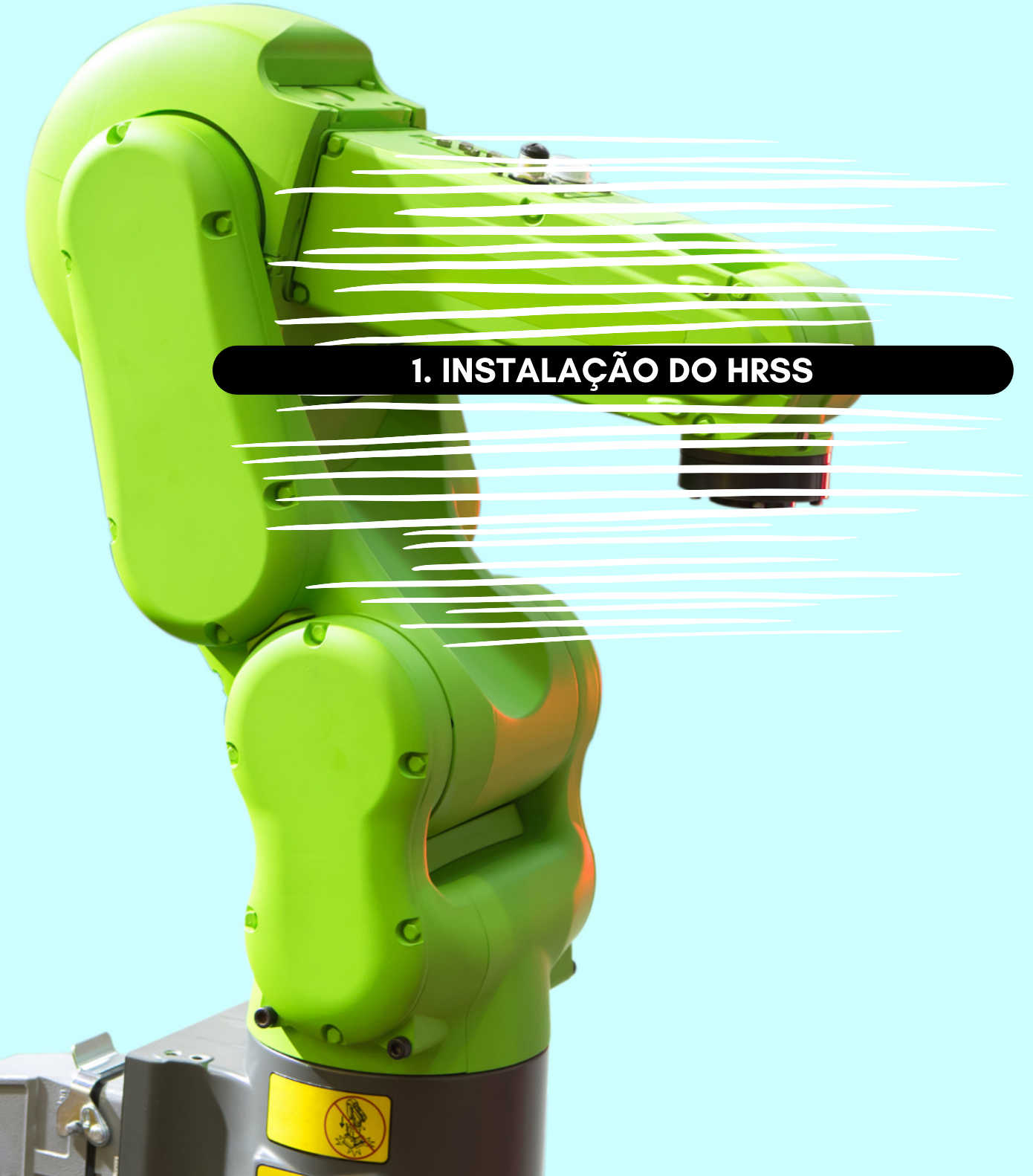
Este e-book está dividido em 10 capítulos. O capítulo 1 inicia com a instalação do simulador HRSS, em seguida, o capítulo 2 apresenta o ambiente de trabalho, os capítulos 3 e 4 ensinam como trabalhar com os movimentos básicos, o capítulo 5 mostra como adicionar a garra no robô, o capítulo 6 apresenta as entradas e saídas digitais, depois no capitulo 7 temos a realização de uma prática, o capítulo 8 aborda a interação do programa Putty com o HRSS usando a rede ethernet, o capítulo 9 aborda o desenvolvimento de um programa de paletização e por fim no capítulo 10, as principais instruções utilizadas.

Todos os capítulos apresentam recursos (vídeos gravados) associados ao conteúdo, que podem ser acessados pelo QRcode e/ou por links no próprio e-book. Os vídeos estão listados publicamente e armazenados no Canal do YouTube "Mecatrônica na Prática", clique na figura abaixo para conhecer o Playlist.

Desejamos que este e-book sirva como ponto de partida para os alunos e interessados na área da Robótica Industrial, no desenvolvimento das suas carreiras como técnicos, tecnólogos e/ou engenheiros.

Os Autores.

Sumário



1. INSTALAÇÃO DO HRSS

1. INSTALAÇÃO DO HRSS

Objetivos:

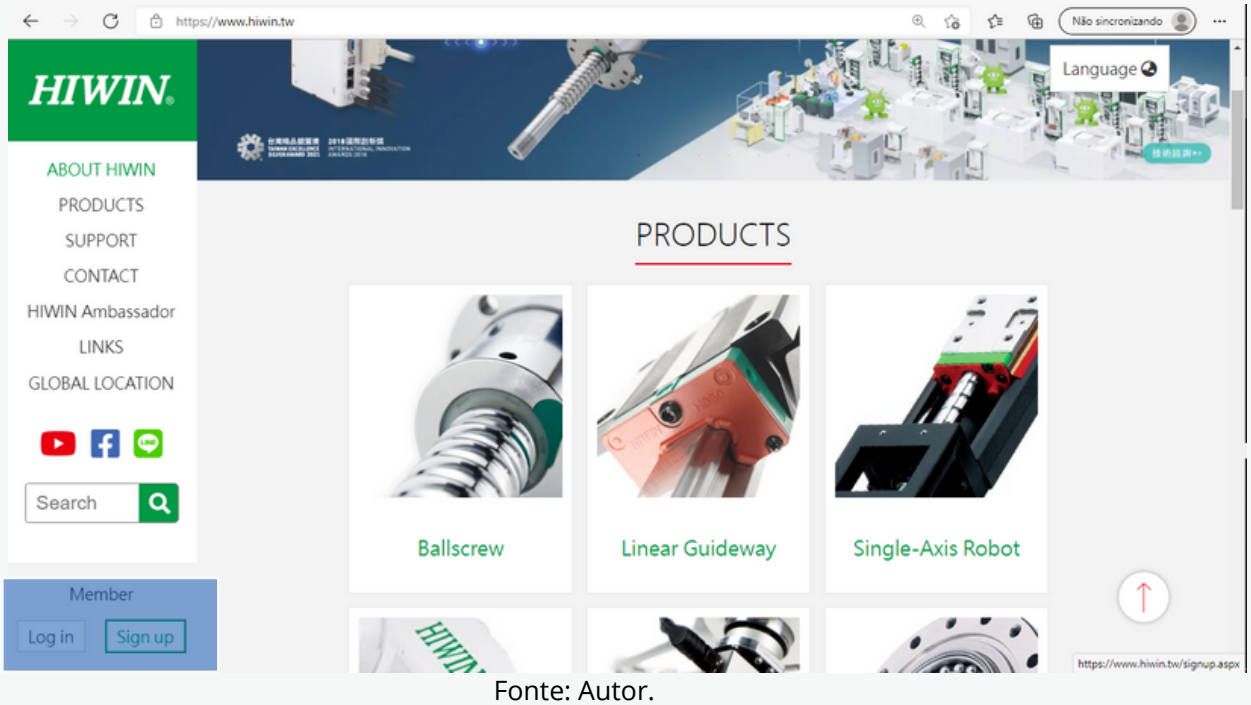
1. Etapas para o download;
2. Instalação do HRSS

Etapas para o download e instalação do HRSS.

1. Etapa 1 é necessário realizar o cadastro no site abaixo na área azul, figura 1.1.

https://www.hiwin.tw

Figura 1.1 - Home page da HYWIN.



Fonte: Autor.

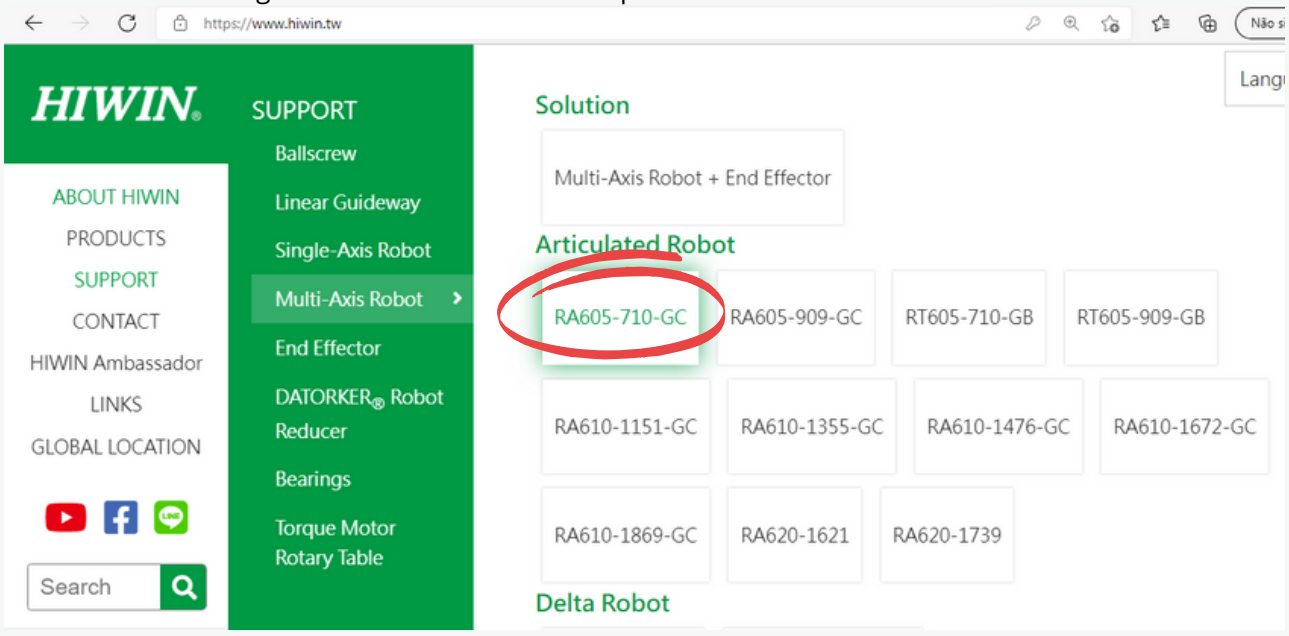
Em seguida realizar o Login, figura 1.2.

Figura 1.2 - Download do HIWIN.

Fonte: Autor.

Realizado o download. De um clique no link da figura 1.3.

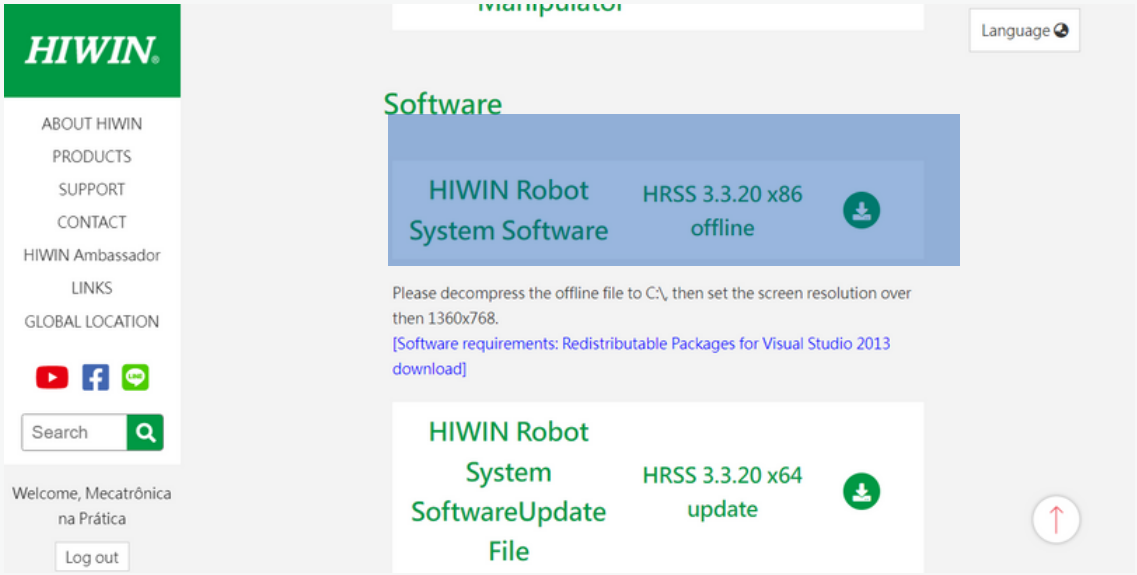
Figura 1.3 - Acessar o caminho para o download do simulador.



Fonte: Autor.

Em seguida faça o download do programa HRSS 3.3.20 x86, figura 1.4.

Figura 1.4 - Download do Software - HRSS 3.3.20 x86.



Fonte: Autor.

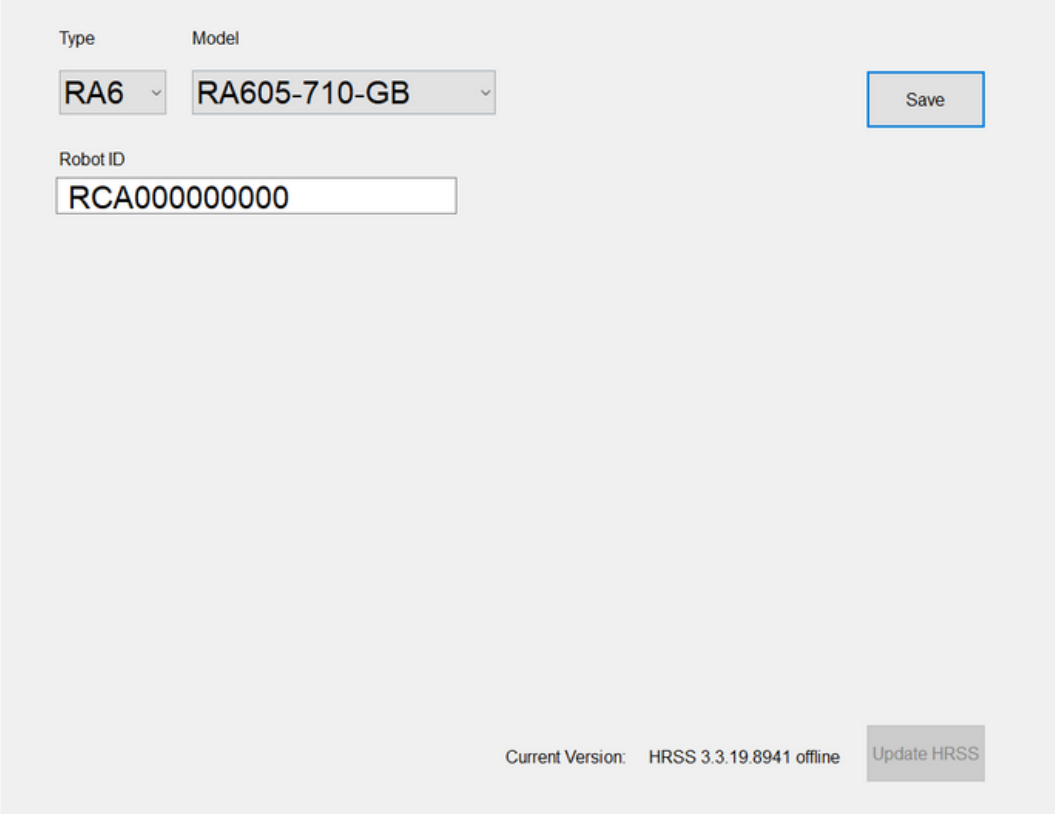
Descompacte o arquivo em um diretório e pronto. Já podemos utilizar o programa HRSS, figura 1.5.

Figura 1.5 - Executável - HRSS 3.3.20 x86.

Fonte: Autor.

Ao abrir pela primeira vez a janela da figura 1.6 aparecerá com opções de escolher o tipo e o modelo de robô a ser trabalhado no projeto.

Figura 1.6 - Configuração do tipo e modelo no HRSS 3.3.20 x86.



Fonte: HRSS 3.3.20.9452 32 bit.

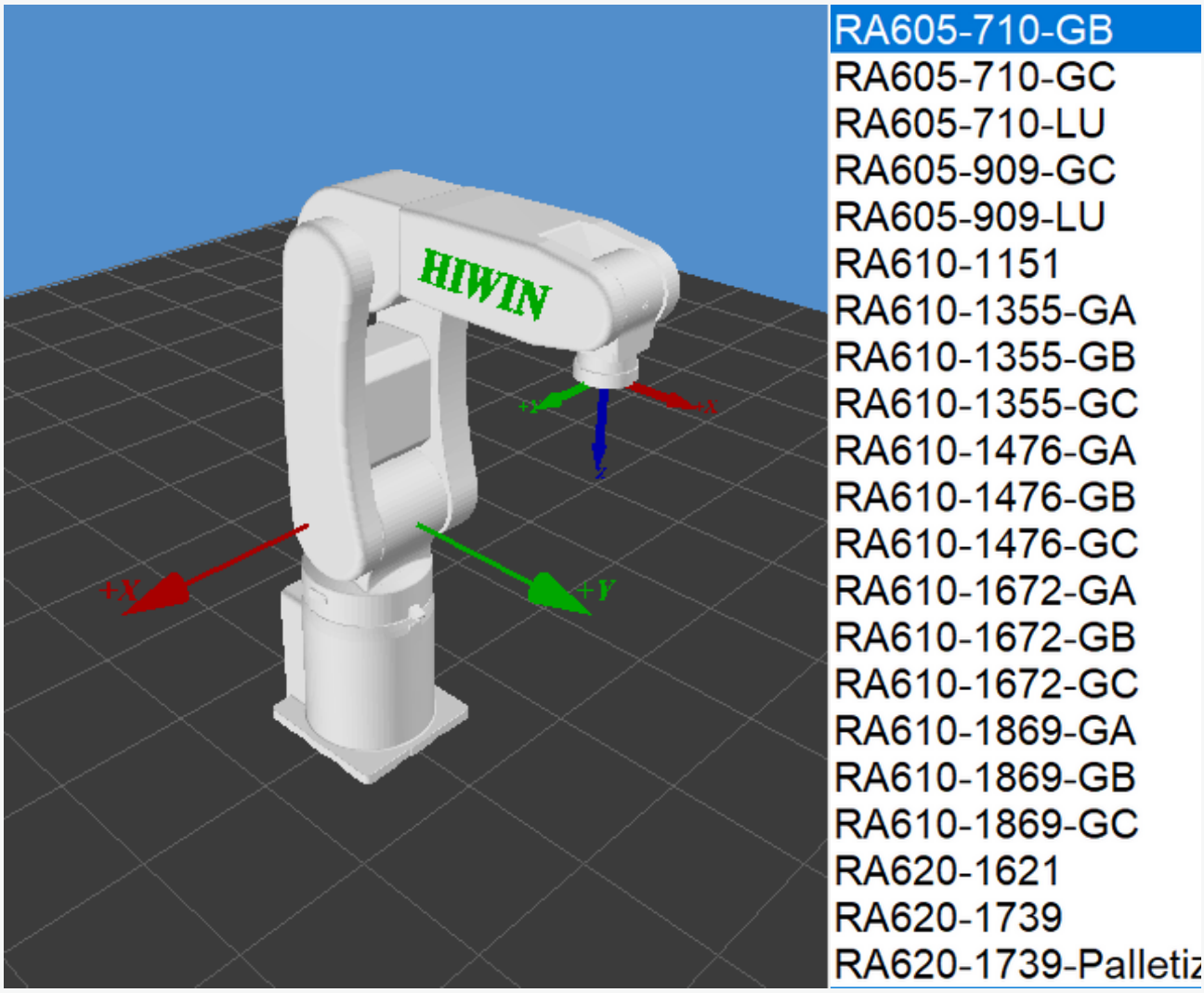
O HRSS apresenta 4 tipos de robôs industriais:

RA - Robôs Articulados (RA6 ou RT);
RD -Robôs Delta (RD4);
RS - Robôs Scara (RS4);
RWS - Robôs Wafer

Vamos verificar a seguir.

1. RA - Robôs Articulados (RA6 ou RT);

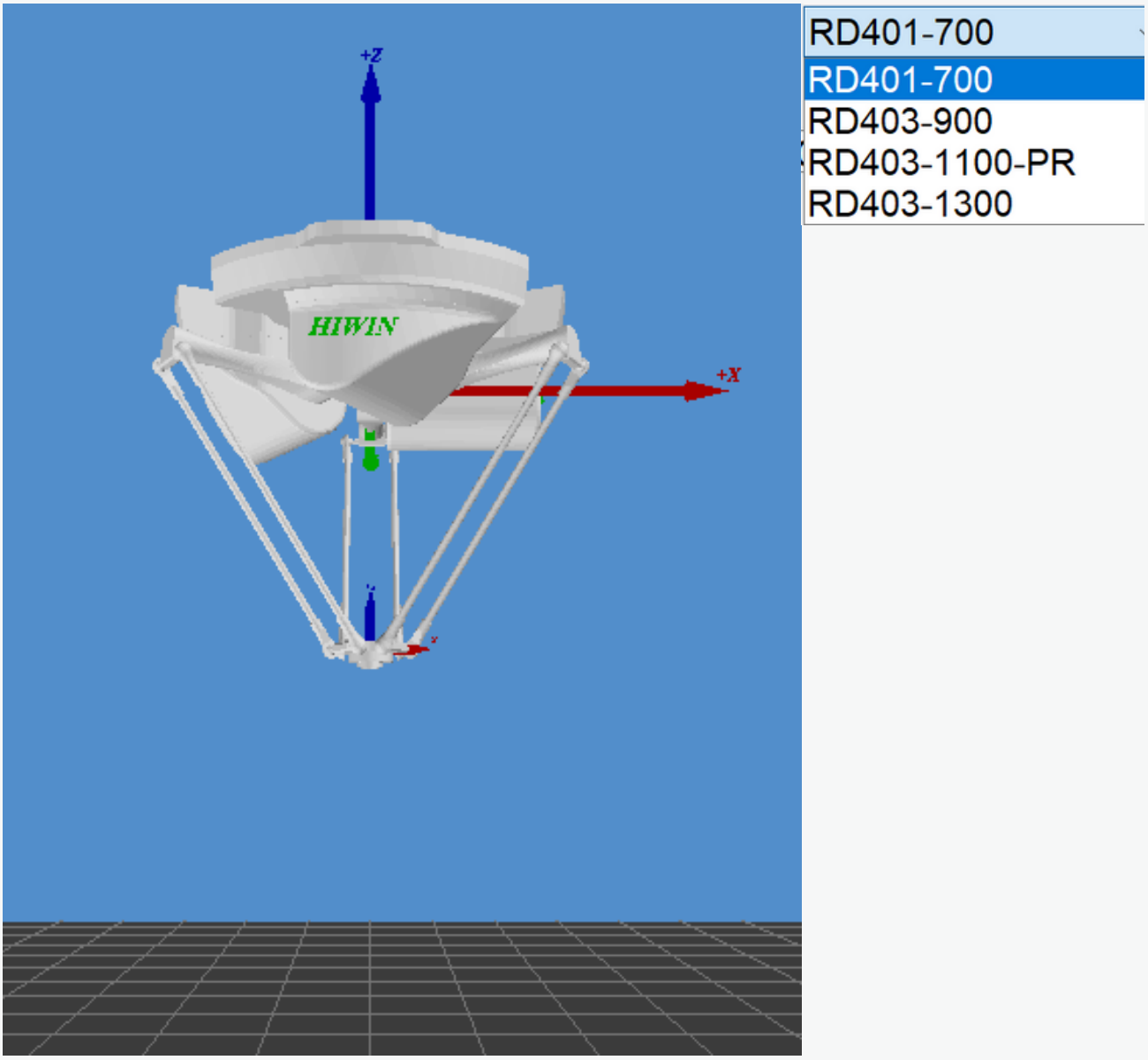
Figura 1.7 - RA605-710-GB no HRSS 3.3.20 x86.



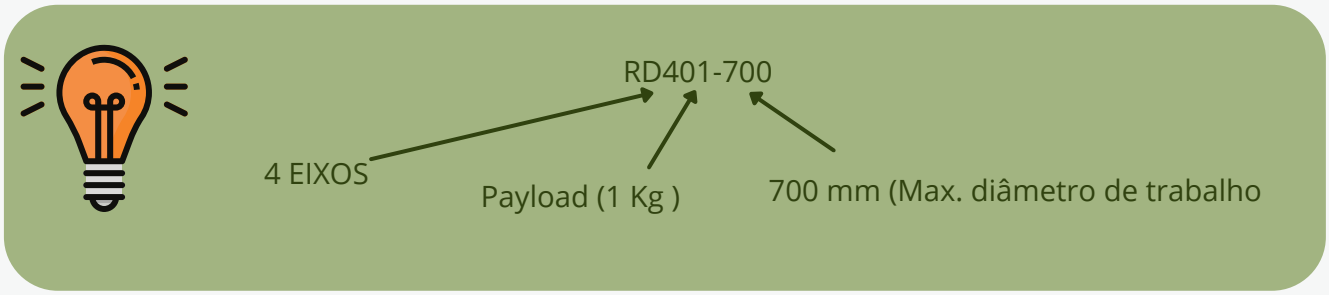
Fonte: HRSS 3.3.20.9452 32 bit.

2. RD -Robôs Delta (RD4);

Figura 1.8 - RD401-700 no HRSS 3.3.20 x86.

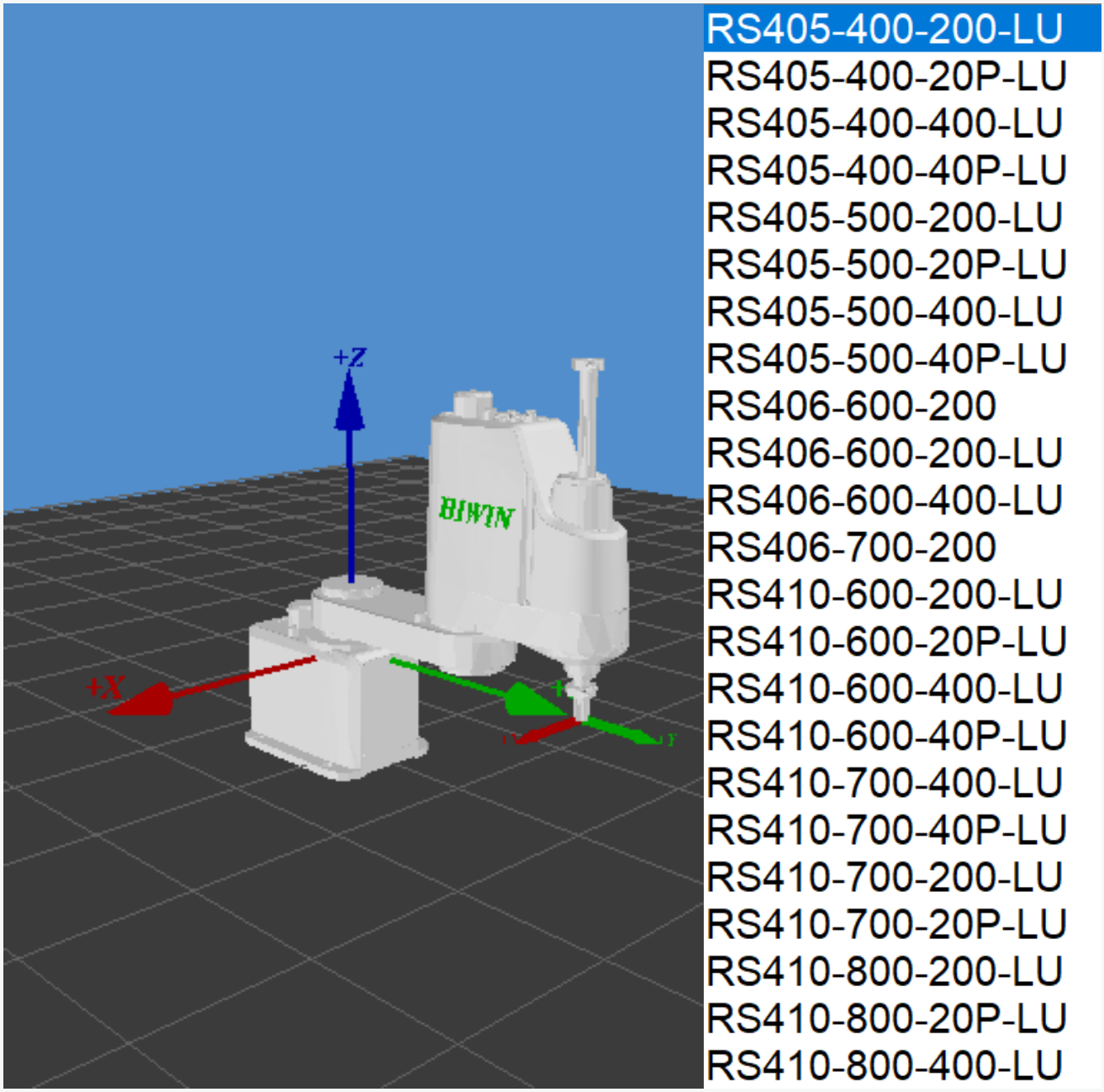


Fonte: HRSS 3.3.20.9452 32 bit.

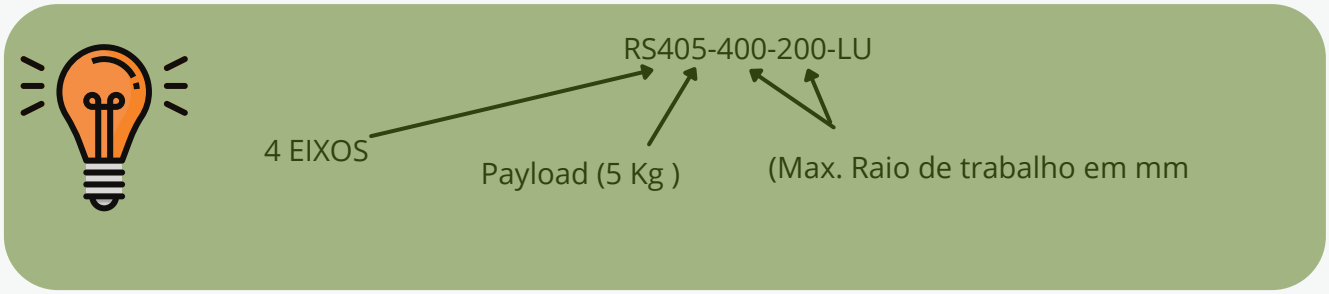


3. RS - Robôs Scara (RS4);

Figura 1.9 - RS405-400-200-LU no HRSS 3.3.20 x86.



Fonte: HRSS 3.3.20.9452 32 bit.



Para visualizar a interface gráfica sem cortes o usuário deve ajustar a sua resolução para 1360x768. O link abaixo tem um vídeo realizando todas as etapas anteriores, figura 1.10 ou acesse o vídeo clicando na mão www.

Figura 1.10 - Link para assistir o vídeo com as etapas de instalação.

https://www.youtube.com/watch?v=ap9MwXt3tDQ&t=3s

Fonte: Autor.

2. CONHECENDO O AMBIENTE HRSS

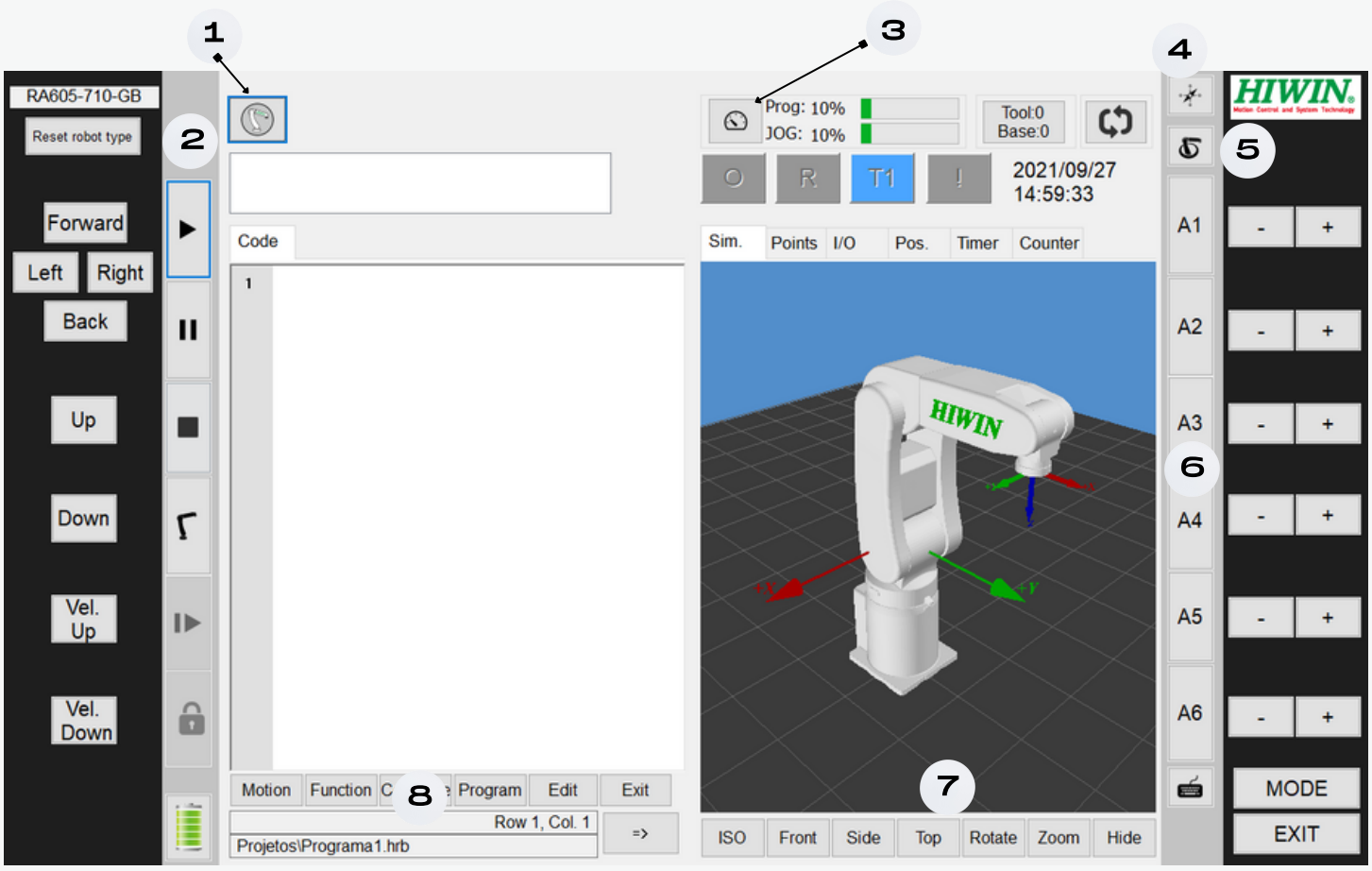
2. CONHECENDO O AMBIENTE HRSS

Objetivos:

1. Conhecer o ambiente HRSS (menu, botões de configuração, botões de movimento do robô, botões de visualização do robô)
2. Modos de operação AUT, EXT, T1 e T2

Ao abrir o programa HRSS aparece a janela da figura 2.1.

Figura 2.1 - Ambiente do programa HRSS.



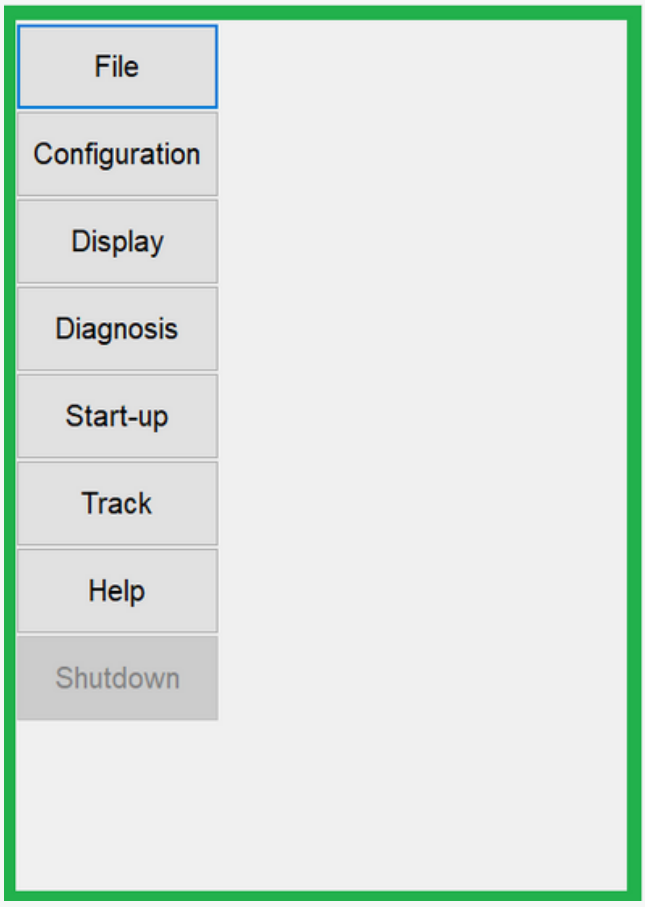
Fonte: HRSS 3.3.20.9452 32 bit.

Na figura 2.1 identificamos as seguintes partes do programa HRSS:

1. Botão Menu com as opções novo, abrir, salvar e sair do programa;
2. Iniciar Simulação, Parar, Pausar e posição Home;
3. Controle de velocidade do programa e dos movimentos;
4. Posição do Flexpendant em relação ao Manipulador;
5. Sistema de coordenadas;
6. Juntas de acordo com o sistema de coordenadas;
7. Visualização do Robô manipulador;
8. Gerenciamento de pasta e de arquivos.

A figura 2.2 mostra as opções do menu.

Figura 2.2 - Opções do menu do HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

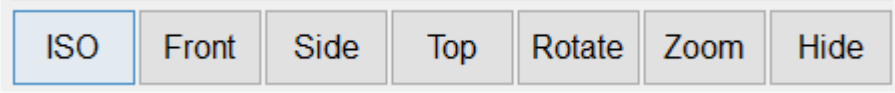
A figura 2.3 mostra as opções de controle da simulação do Robô.

Figura 2.3 - Opções de controle da simulação do HRSS.

Fonte: HRSS 3.3.20.9452 32 bit.

A figura abaixo ajuda na visualização do robô em relação a área de trabalho.

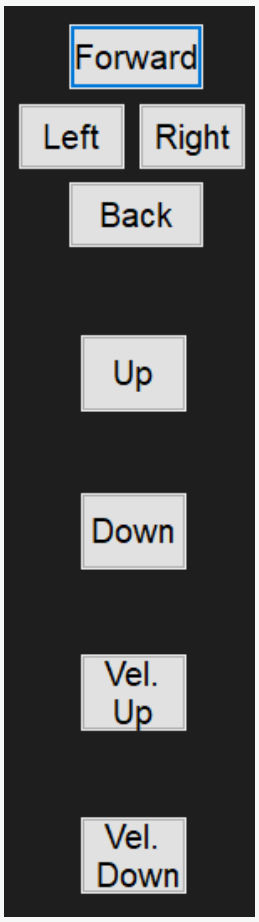
Figura 2.4 - Opções de visualização do robô no HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

A figura 2.5 mostra os botões de movimento do robô, experimente. Mas, antes aumente a velocidade de movimentação com os botão **Vel. Up**.

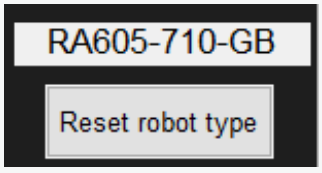
Figura 2.5 - Opções de movimento com botões no HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

O botão **Reset robot type** da figura 2.6 permite a troca de modelos de robôs do fabricante. Para isso, pressione o botão e confirme. Em seguida, saia do programa e abra-o novamente.

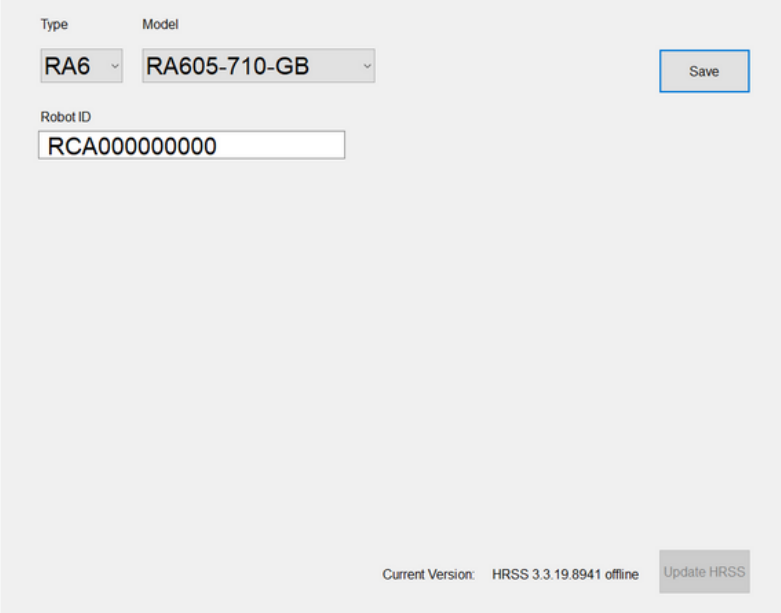
Figura 2.6 - Botão de reset no HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

A janela da figura 2.7 aparecerá, então, escolha o modelo desejada e pressione o botão **save**.

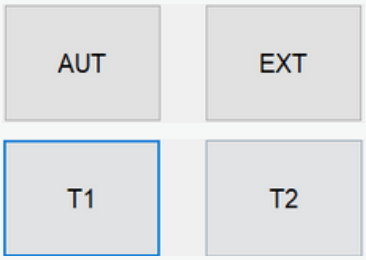
Figura 2.7 - Tipos de robôs no HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

Venho destacar os modos de operação: AUT, EXT, T1 e T2 quando clica no botão **MODO** ao lado do botão **EXIT,** figura 2.1. Encontramos as opções da figura 2.8.

Figura 2.8 - Modos de operação no HRSS.



Fonte: HRSS 3.3.20.9452 32 bit.

- **AUT** : Modo automático (Utilizado para testar os programas, sem poder alterar)
- **EXT**: Modo automático externo(Usado para comunicar externo com CLP, por exemplo)
- **T1**: Modo expert (Usado para testar os programas, alterar os programas e as instruções com velocidade de operação máxima 250 mm/s)
- **T2**: Modo Usuário(Usado para testar os programas com velocidade de operação menor que 250 mm/s)

A próximo capítulo veremos as instruções de movimentos:

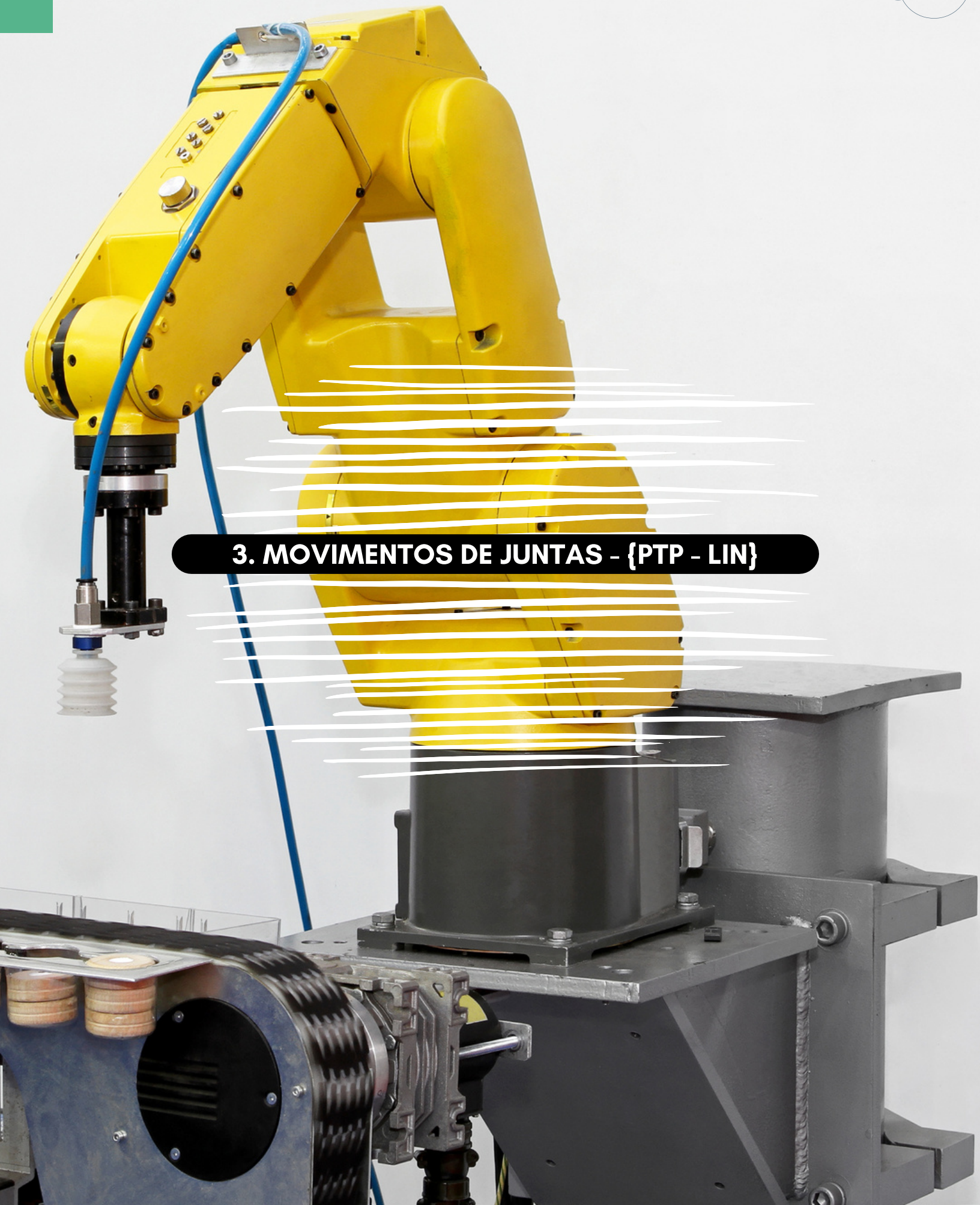
1. Point-to-point : PTP
2. Linear : LIN
3. Circular : CIRC

O link abaixo tem um vídeo demonstrando a manipulação do robô no ambiente e o ambiente de trabalho. Acesse com o link do QRcode da figura 2.9 ou clique na mão www.

Figura 2.9 - QRCode com o vídeo explicando o ambiente do HRSS.

https://www.youtube.com/watch?v=EFYBtxB7mP0&list=PLqTzJCvlOflhu7-yOFBIn0GJZezKplQgX&index=2

Fonte: Autor.



3. MOVIMENTOS DE JUNTAS - {PTP - LIN}

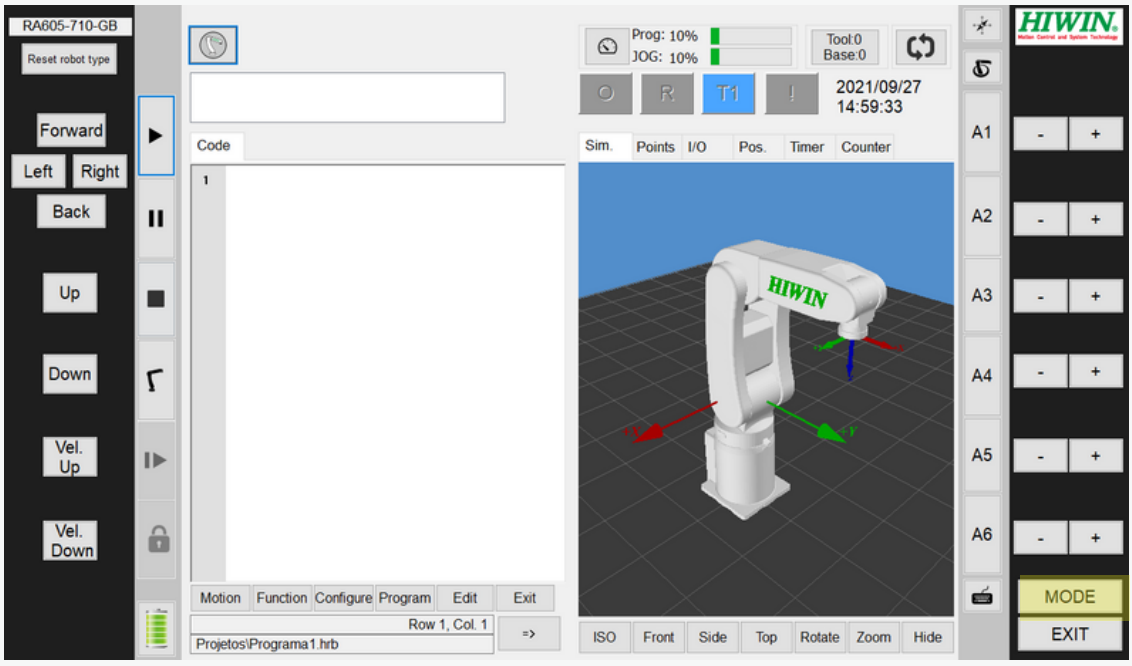
3. MOVIMENTOS DE JUNTAS COM AS INSTRUÇÕES:PTP E LIN

Objetivos:

1. Criar pontos para uma trajetória (utilizando a variável E6POS e Teach Pendant - TP)
2. Realizar movimentos
   a. Instrução PTP
   b. Instrução LIN

Podemos movimentar o robô manipulador. Uma vez escolhido modelo do robô manipulador: RA605-710-GB no ambiente de trabalho, figura 3.1.

Figura 3.1 - Movimentos com o Robô RA605-710-GB.

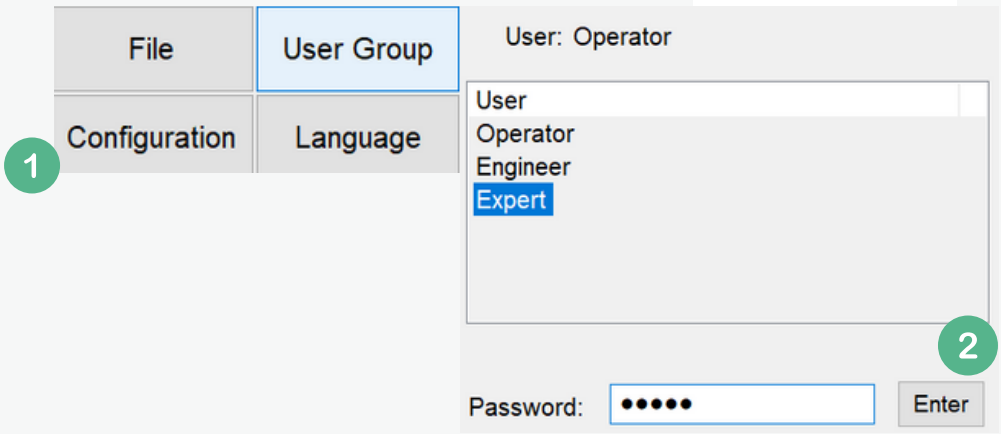


Fonte: HRSS 3.3.20.9452 32 bit.

Antes de iniciar a programação deve-se verificar o **modo** esta ativo em **T1**, figura 3.1. Para isso, clique no botão MODE e escolha a opção T1, caso não apareça a opção, clique novamente no botão MODE.

Depois verificar, se o usuário **Expert** esta ativo. No *Menu - Configuration - User Group*, em seguida clica em *Expert*. A senha é **hiwin**, figura 3.2.

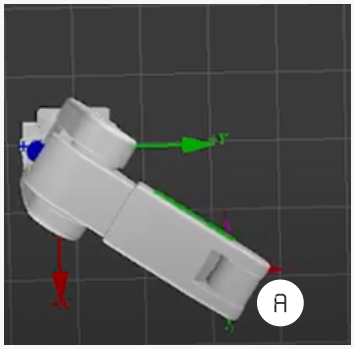
Figura 3.2 -Menu User Group no modo Expert e senha hiwin.



Fonte: HRSS 3.3.20.9452 32 bit.

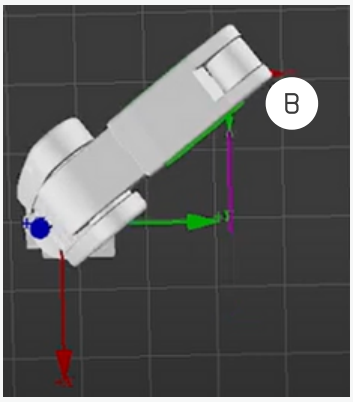
Após realizar os movimentos do ponto A para o ponto B considerando o eixo X (em vermelho), figura 3.3 (A) e (B). Ao lado as coordenadas referentes aos pontos, figuras 3.3 (C) e (D).

Figura 3.3 - (A) e (B) com os movimentos e (C) e (D) com as coordenadas.



A1	-33.007	degree
A2	-12.195	degree
A3	13.498	degree
A4	0.000	degree
A5	-91.303	degree
A6	-33.007	degree
X	239.050	mm
Y	368.000	mm
Z	293.500	mm
A	180.000	degree
B	0.000	degree
C	90.000	degree

C



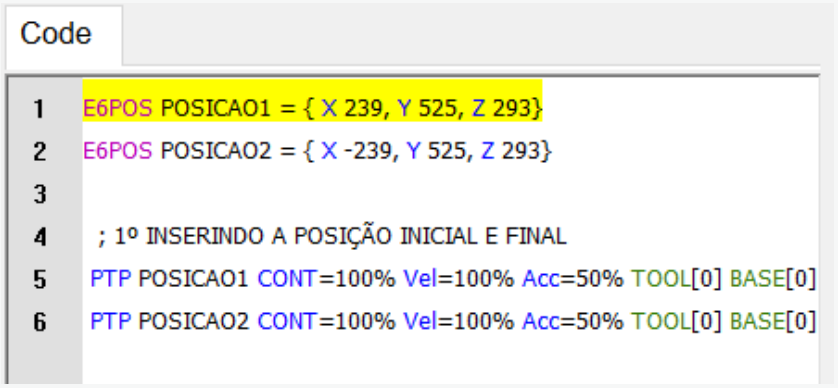
A1	37.973	degree
A2	-17.281	degree
A3	19.891	degree
A4	0.000	degree
A5	-92.610	degree
A6	37.973	degree
X	-287.238	mm
Y	368.000	mm
Z	293.500	mm
A	180.000	degree
B	0.000	degree
C	90.000	degree

D

Fonte: HRSS 3.3.20.9452 32 bit.

(1) Vamos digitar o código envolvendo a variável E6POS e a instrução PTP, figura 3.4.

Figura 3.4 -Código com o Robô RA605-710-GB.




```
E6POS POSICAO1 = { X 239, Y 525, Z 293}
E6POS POSICAO2 = { X -239, Y 525, Z 293}

 ; 1º INSERINDO A POSIÇÃO INICIAL E FINAL
PTP POSICAO1 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
PTP POSICAO2 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
```

Fonte: HRSS 3.3.20.9452 32 bit.

OBS: O movimento com a instrução PTP é considerado um movimento aéreo. Pois, não nos preocupamos com desvios durante a trajetória. Realiza um movimento de um arco.

(2) Vamos digitar as instruções abaixo e testar o movimento:

```
; 2º DESLOCANDO A COORDENADA DESEJADA
    PTP {X 200}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
```

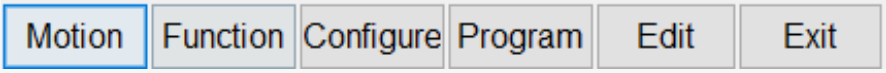
A INSTRUÇÃO **E6POS** CRIA POSIÇÃO PASSANDO 3 PARÂMETROS { X, Y , Z}, HÁ TAMBÉM A POSSIBILIDADE DE PASSAR OS ÂNGULOS.

(3) Vamos digitar as instruções abaixo e testar o movimento:

```
; 3º DESLOCANDO A JUNTA
    PTP {A1 45}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
    PTP {A1 -45}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
```

A partir desse menu podemos criar os pontos na botão: Motion, figura 3.5.

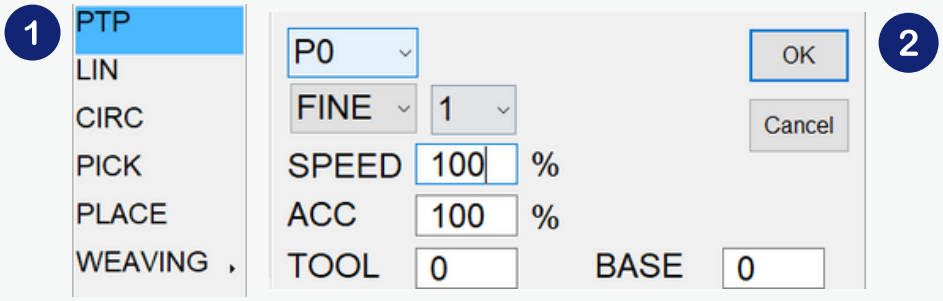
Figura 3.5 - Botões com funções, edição e configurações do Robô.



Fonte: HRSS 3.3.20.9452 32 bit.

Em seguida criar os pontos na botão: Motion, figura 3.5 para as duas coordenadas A e B. Lembrando de realizar o movimento para cada coordenada e depois realizar os procedimentos (1) e (2) da figura 3.6.

Figura 3.6 - Escolher (1) em seguida colocar os parâmetros da opção (2).



Fonte: HRSS 3.3.20.9452 32 bit.

Dessa forma iremos obter o código abaixo utilizando o módulo TP (Teach Pendant).

```
; UTILIZANDO O MODULO TP
    PTP P0 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
    PTP P1 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
```

O código abaixo trabalha com os diferentes tipos de implementação com as instruções PTP e LIN, figura 3.7.

Figura 3.7 - Instruções do Robô RA605-710-GB.

```
DECL E6POINT P0={A1 -33.002, A2 -12.178, A3 13.392, A4 0.000, A5 -91.214, A6
-33.002, X 239.000, Y 368.000, Z 293.000, A 180.000, B 0.000, C 90.000, E1 0.000,
E2 0.000, E3 0.000}
;NULL; Tool=0; Base=0; E1Mode=NULL; E2Mode=NULL; E3Mode=NULL
DECL E6POINT P1={A1 33.002, A2 -12.178, A3 13.392, A4 0.000, A5 -91.214, A6
33.002, X -239.000, Y 368.000, Z 293.000, A 180.000, B 0.000, C 90.000, E1 0.000,
E2 0.000, E3 0.000}
;NULL; Tool=0; Base=0; E1Mode=NULL; E2Mode=NULL; E3Mode=NULL
; P1 (239 368 293) X Y Z
;P2 (-239 368 293) X Y Z
E6POS POSICAO1 = { X 239, Y 525, Z 293}
E6POS POSICAO2 = { X -239, Y 525, Z 293}

WHILE 1==1
    ; 1º INSERINDO A POSIÇÃO INICIAL E FINAL
        PTP POSICAO1 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
        PTP POSICAO2 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]

    ; 2º DESLOCANDO A COORDENADA DESEJADA
    ;       PTP {X 200}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]

    ; 3º DESLOCANDO A JUNTA
    ;        PTP {A1 45}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]
    ;        PTP {A1 -45}  CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0]

    ; UTILIZANDO O MODULO TP
    ;        PTP P0 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
    ;       PTP P1 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]

    ; 1º INSERINDO A POSIÇÃO INICIAL E FINAL
    LIN POSICAO1 FINE=1 Vel=1000mm/s Acc=100% TOOL[0] BASE[0]
    LIN POSICAO2 FINE=1 Vel=1000mm/s Acc=100% TOOL[0] BASE[0]
     ; 2º DESLOCANDO A COORDENADA DESEJADA
   ; LIN {X 150} FINE=1 Vel=1000mm/s Acc=100% TOOL[0] BASE[0]

    ; 3º DESLOCANDO A JUNTA
    ; LIN {A1 60} FINE=1 Vel=1000mm/s Acc=100% TOOL[0] BASE[0]
ENDWHILE
```

Fonte: O autor.

O link abaixo tem um vídeo demonstrando as instruções PTP e LIN com o robô no ambiente com o QRcode da figura 3.8 ou clique na mão www.

Figura 3.8 - QRCode com o vídeo explicando as instruções PTP e LIN.

https://www.youtube.com/watch?v=oBPlp0HrMHA&list=PLqTzJCvIOflhu7-yOFBIn0GJZezKplQgX&index=3

Fonte: Autor.

4. MOVIMENTOS DE JUNTAS {CIRC - SPL}

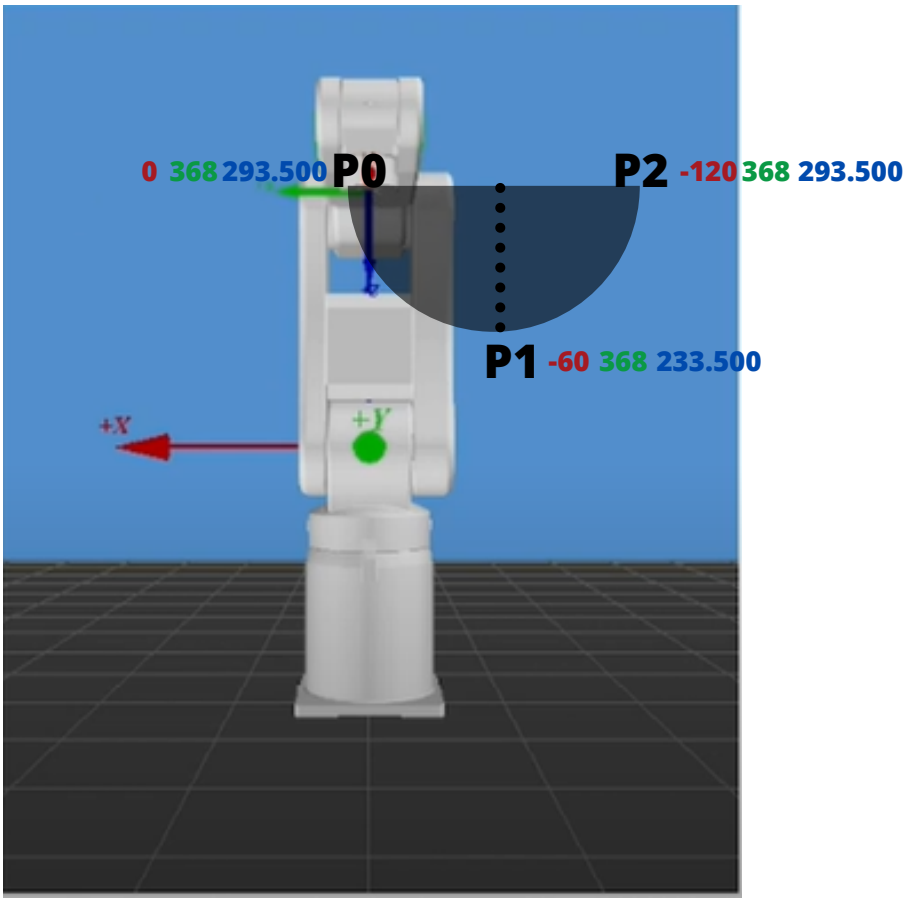
4. MOVIMENTOS DE JUNTAS COM AS INSTRUÇÕES:CIRC E SPL

Objetivos:

1. Realizar movimento circular - CIRC
2. Realizar trajetória com spline - SPL

Podemos movimentar o robô manipulador para os pontos P0, P1 e P2 de acordo com a figura 4.1.

Figura 4.1 - Criação de um semicírculo com as instruções CIRC.



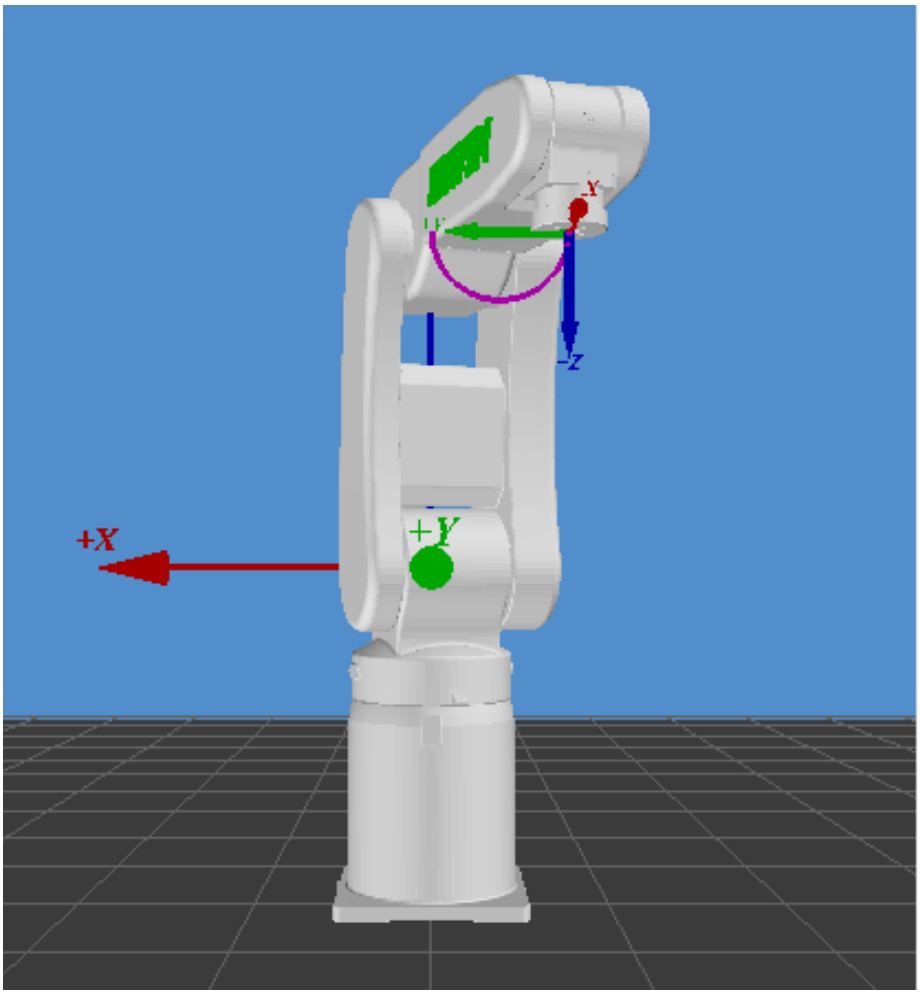
Fonte: HRSS 3.3.20.9452 32 bit.

Podemos movimentar o robô manipulador com as instruções E6POS e CIRC, código abaixo.

```
E6POS P0 = { X 0, Y 368, Z 293.500}
E6POS P1 = { X -60, Y 368, Z 233.500}
E6POS P2 = { X -120, Y 368, Z 293.500}

CIRC P1 P2 CONT=100% Vel= 2000mm/s Acc=50% TOOL[0]
```

O resultado esperado, figura 4.2.

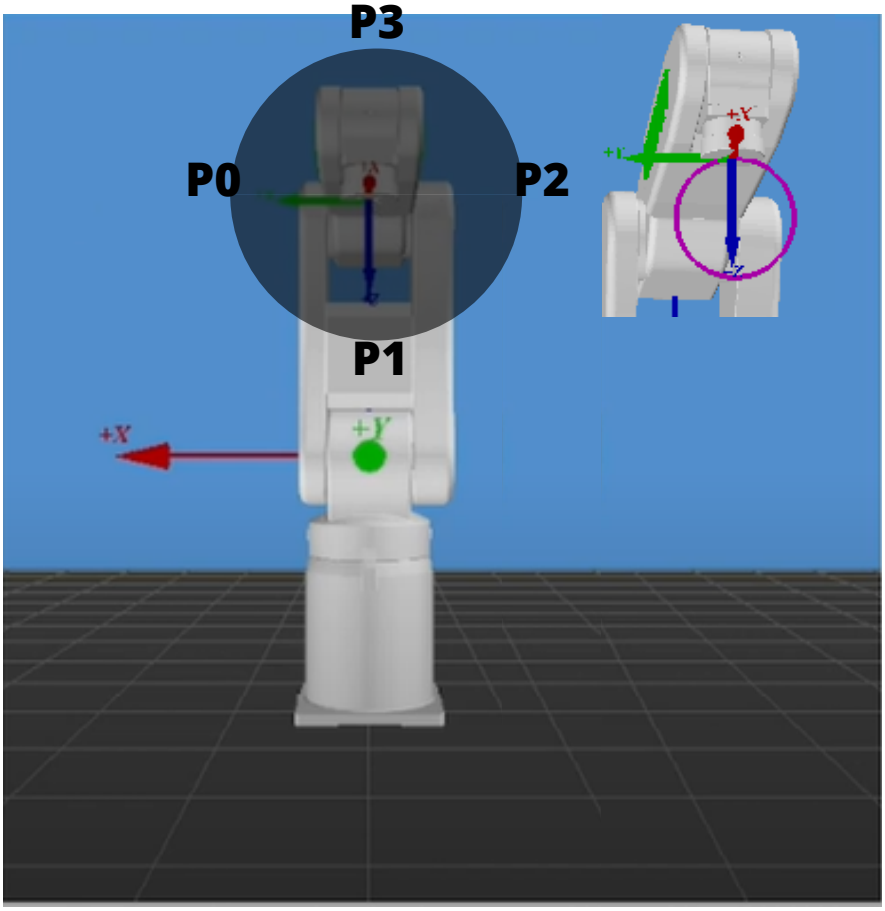
Figura 4.2 - Movimento semicírculo.



Fonte: HRSS 3.3.20.9452 32 bit.

Podemos construir o movimento do círculo completo com o robô manipulador com as instruções E6POS e CIRC, figura 4.3.

Figura 4.3 - Movimento completo do círculo.



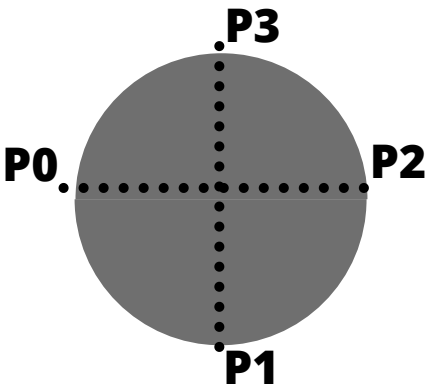
Fonte: HRSS 3.3.20.9452 32 bit.

O código a seguir realiza o movimento circular com o acréscimo do ponto auxiliar P3.

```
E6POS P0 = { X 0,    Y 368, Z 293.500}
E6POS P1 = { X-60,  Y 368, Z 233.500}
E6POS P2 = { X-120, Y 368, Z 293.500}
E6POS P3 = { X-60, Y 368, Z 353.500}

WHILE 1==1
 CIRC P1 P2 CONT=100% Vel = 2000mm/s Acc=50% TOOL[0] BASE[0]
 CIRC P3 P0 CONT=100% Vel = 2000mm/s Acc=50% TOOL[0] BASE[0]
ENDWHILE
```

Vamos agora adicionar os pontos P0 [ORIGEM] , P1[X - 60] , P2[X - 120] [Z - 60] e P3[X- 60] [Z + 60], um por um. Lembre-se não vamos alterar a coordenada Y (verde), figura 4.4.

Figura 4.4 - Coordenadas do círculo.

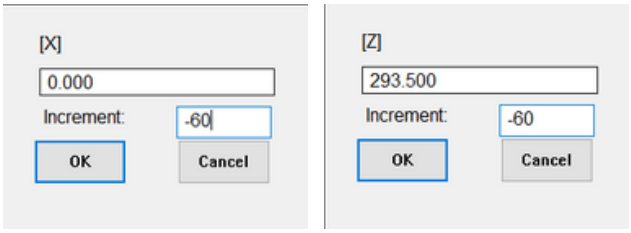


Fonte: Autor.

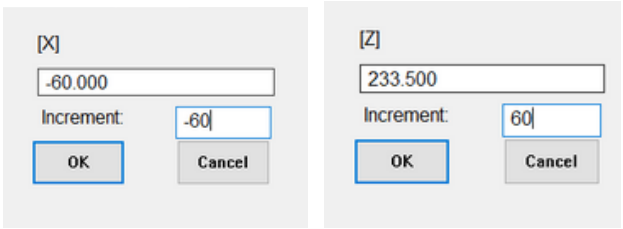
A segunda forma de implementar é acrescentando os pontos.

NO.	Name	Comment
0	P0	Home
1	P1	
2	P2	
3	P3	

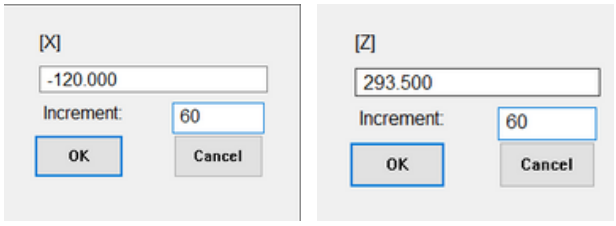
P1



P2



P3



O código do movimento a partir dos pontos adicionados com o laço infinito utilizando a instrução While.

```
;2 METODO
WHILE 1==1
 LIN P0 CONT Vel=2000mm/S Acc=50% TOOL[0] BASE[0]
 CIRC P1 P2 CONT=100% Vel= 2000mm/S Acc=50% TOOL[0] BASE[0]
 CIRC P3 P0 CONT=100% Vel= 2000mm/S Acc=50% TOOL[0] BASE[0]
ENDWHILE
```

Temos a possibilidade de criar o círculo com a instrução CIRC_REL, código abaixo.

```
;3 METODO
    CIRC_REL {X-100, Z 100}{X-100, Z-100}
```

Por último, a instrução SPLINE.

```
;4 METODO
SPLINE
  SPL P0
  SPL P1
  SPL P2
ENDSPLINE
```

Podemos movimentar o robô manipulador com as instruções CIRC e SPL, figura 4.5.

Figura 4.5 - Código com as instruções CIRC e SPL.

```
;E6POS P0 = { X 0,    Y 368, Z 293.500}
;E6POS P1 = { X-60,  Y 368, Z 233.500}
;E6POS P2 = { X-120, Y 368, Z 293.500}
;E6POS P3 = { X-60, Y 368, Z 353.500}

;WHILE 1==1
; CIRC P1 P2 CONT=100% Vel = 2000mm/s Acc=50% TOOL[0] BASE[0]
; CIRC P3 P0 CONT=100% Vel = 2000mm/s Acc=50% TOOL[0] BASE[0]
;ENDWHILE

;2 METODO
;WHILE 1==1
; LIN P0 CONT Vel=2000mm/S Acc=50% TOOL[0] BASE[0]
; CIRC P1 P2 CONT=100% Vel= 2000mm/S Acc=50% TOOL[0] BASE[0]
; CIRC P3 P0 CONT=100% Vel= 2000mm/S Acc=50% TOOL[0] BASE[0]
;ENDWHILE

;3 METODO
;      CIRC_REL {X-100, Z 100}{X-100, Z-100}

;4 METODO
SPLINE
  SPL P0
  SPL P1
  SPL P2
ENDSPLINE
```

Fonte: Autor.

A figura abaixo tem um vídeo demonstrando as instruções CIRC e SPL ou clique na mão www.

Figura 4.6 - QRcode com o vídeo com as instruções CIRC e SPL.

https://www.youtube.com/watch?v=JLdN3niRiPM&list=PLqTzJCvIOfIhu7-yOFBIn0GJZezKplQgX&index=4

Fonte: Autor.

5. ADICIONANDO GARRA NO ROBÔ RA605-05-710-GB

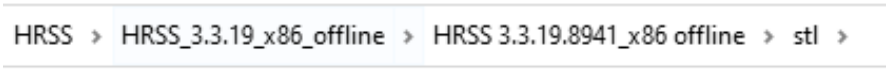
5. ADICIONANDO GARRA NO ROBÔ RA605-05-710-GB

Objetivos:

- Adicionar a garra no robô

Vamos adicionar uma ferramenta ao robô manipulador. Vocês podem criar a ferramenta[1] ou adicionar uma já existente. Neste capitulo vamos adicionar uma já existente como exemplo. Para isso, acessem o link ***https://grabcad.com/library*** para o download do modelo **pneumetic gripper** no formato ***stl***. Para realizar o download deve-se preencher um cadastro. Para adicionar o arquivo devemos inserir na pasta stl de acordo com o caminho abaixo, figura 5.1.

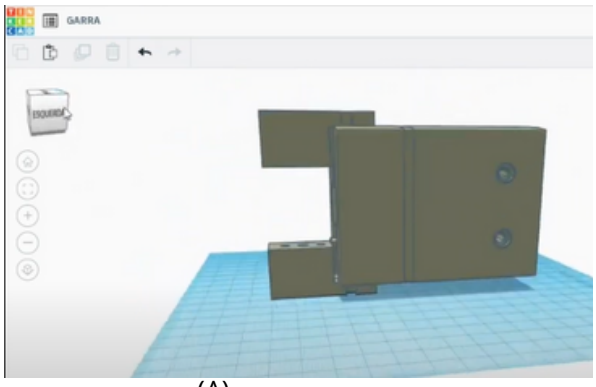
Figura 5.1 - Caminho da pasta stl.



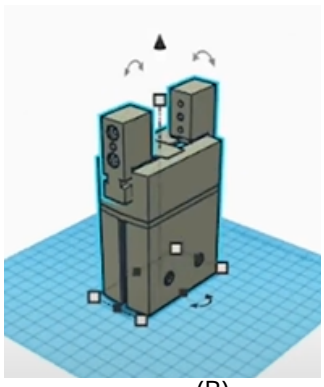
Fonte: Autor.

Observamos no tinkercard que a garra esta deitada, figura 5.2 (A) e para inserir no robô devemos rotacionar a garra, figura 5.2 (B). Caso, não corrija a garra será adicionada no robô de forma errada.

Figura 5.2- (A) Garra original e (B) Garra rotacionada no tinkercard.



(A)



(B)

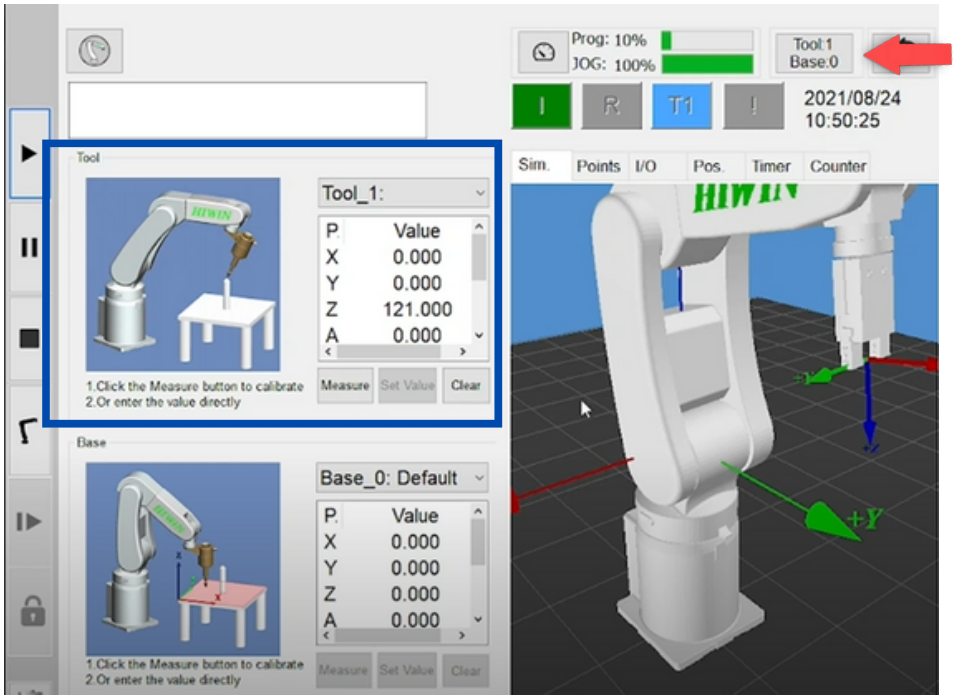
Fonte: Autor.

Após corrigir a posição da garra devemos exportar o arquivo e adicionar na pasta stl do programa. Com a instrução **ADDTOOL** *nome_da_garra* conseguimos adicionar a garra no manipulador.

[1] *A ferramenta pode ser modelada em qualquer software de CAD, desde que possa ser exportado com a extensão .stl. Devem ser criadas dois arquivos, o primeiro com a imagem da ferramenta sem a peça de trabalho (workpece) e o segundo com a ferramenta e a peça de trabalho.*

Nesse ponto devemos calibrar a ferramenta, ou seja, adicionar a referência da ferramenta paro o robô, o TCP (Tool Center Point ou ponto central da ferramenta). Clique na seta para aparecer a janela abaixo. Altere o Tool_0 para Tool_1 e na coordenada Z adicione 121 mm, figura 5.3.

Figura 5.3- Configuração Tool_1.



Fonte: HRSS 3.3.20.9452 32 bit.

O link abaixo tem um vídeo demonstrando como adicionar e corrigir a posição da garra com o robô no ambiente com o QRcode da figura 5.4 ou clique na mão www.

Figura 5.4 - QRCode mostrando como adicionar a garra ao robô.

https://www.youtube.com/watch?v=P5q1bguNACk&list=PLqTzJCvIOfIhu7-yOFBIn0GJZezKplQgX&index=5

Fonte: Autor.

6. TRABALHANDO COM AS ENTRADAS E SAÍDAS DIGITAIS

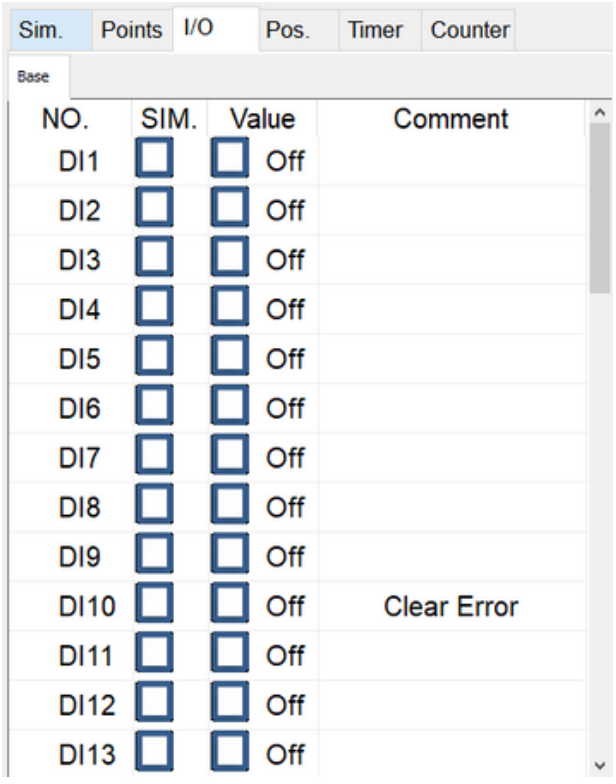
6. TRABALHANDO COM AS ENTRADAS E SAÍDAS DIGITAIS

Objetivos:

1. Entradas e saídas digitais [$] com o robô;
2. Instrução WAIT FOR;
3. Instrução SET_OVERRIDE_SPEED;

Vamos trabalhar com as entradas e saídas digitais. Clique na aba I/O e habilite a caixa da coluna SIM para poder ativar a simulação digital, em seguida, você pode alterar o estado de Off (desligado) para On (ligado), figura 6.1.

Figura 6.1 - Ativar os módulos digital do Robô.



Fonte: HRSS 3.3.20.9452 32 bit.

Adicione 4 pontos no ambiente de trabalho e simule com o código abaixo.

```
;$DI[n] - Digital Input
;$DO[n] - Digital Output
;$RI[n] - Robot Input
;$RO[n] - Robot Output
;VO[n] - Valvule Output
SET_OVERRIDE_SPEED 100 ; Esta instrução aumenta a velocidade
                       ; de execução para  100%
$DO[1] = FALSE

WHILE 1==1
  IF $DI[1] == TRUE THEN
    PTP P0 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
    PTP P1 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
  ENDIF
  IF $DI[2] == TRUE THEN
    PTP P2 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
    PTP P3 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
  ENDIF
ENDWHILE
```

Tente agora utilizando a instrução WAIT FOR com o código abaixo.

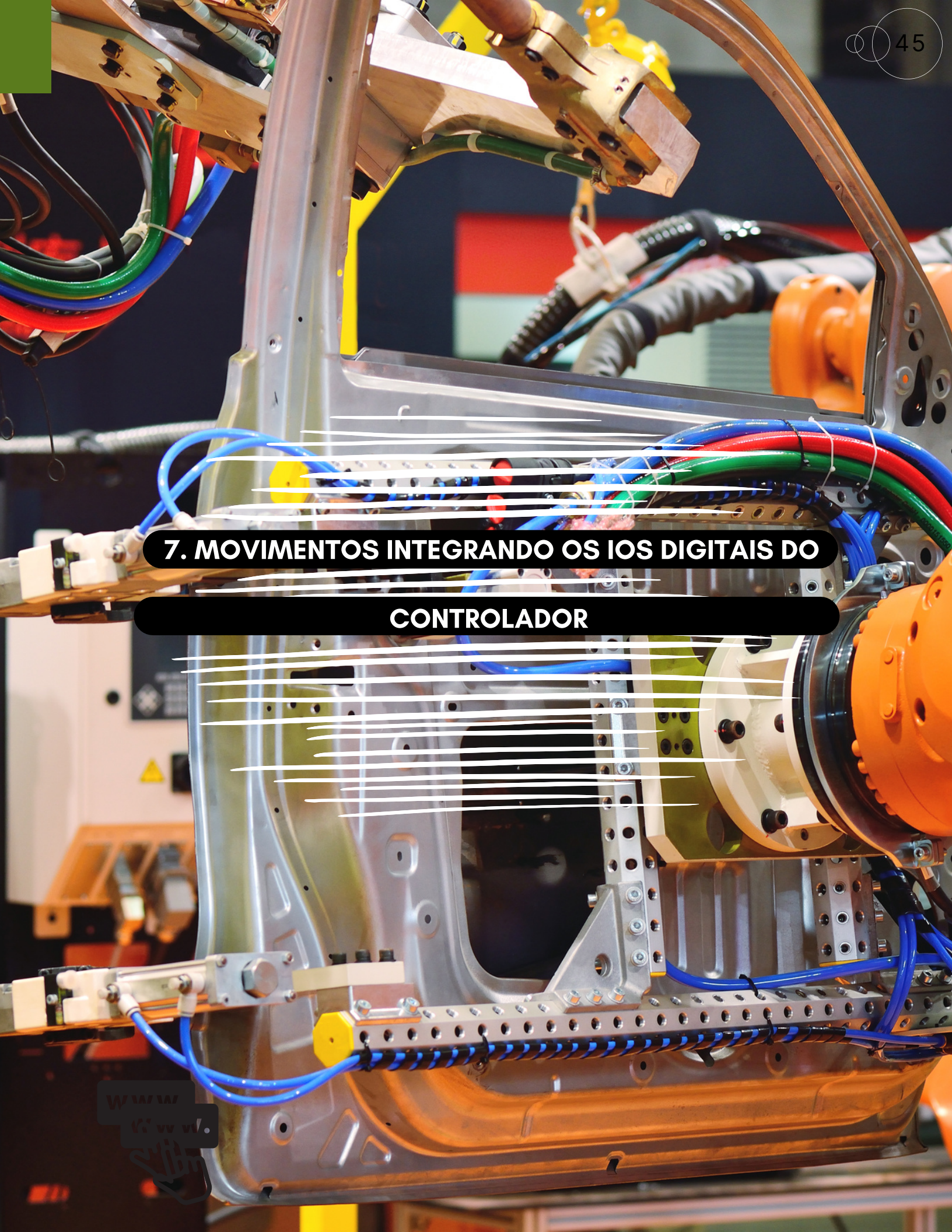
```
SET_OVERRIDE_SPEED 100

WAIT FOR $DI[1] == TRUE
PTP P0 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
PTP P1 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]

WAIT FOR $DI[2] == TRUE
PTP P2 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
PTP P3 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
```

O link abaixo tem um vídeo demonstrando como adicionar entradas digitais coordenada com os movimentos com o QRcode da figura 6.2 ou clique na mão www.

Figura 6.2 - QRCode com o link demonstrando entradas e saídas digitais.

https://www.youtube.com/watch?v=dT_7g-Bb6jU&list=PLqTzJCvIOfIhu7-yOFBIn0GJZezKplQgX&index=6

Fonte: Autor.



7. MOVIMENTOS INTEGRANDO OS IOS DIGITAIS DO CONTROLADOR

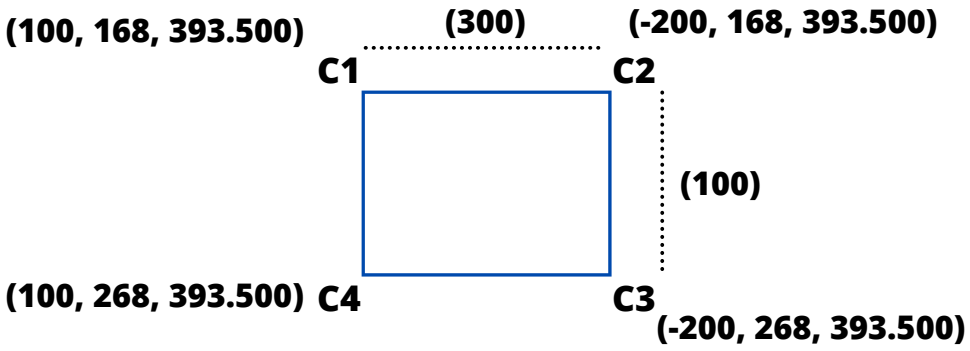
7. MOVIMENTOS INTEGRANDO OS IOS DIGITAIS DO CONTROLADOR

Objetivos:

- Programar movimentos [Retângulo e Círculo] integrando com as entradas digitais

Primeiro vamos construir o movimento de um retângulo com as coordenadas (C1,C2, C3 E C4) na figura 7.1.

Figura 7.1 - Criação do movimento 1.

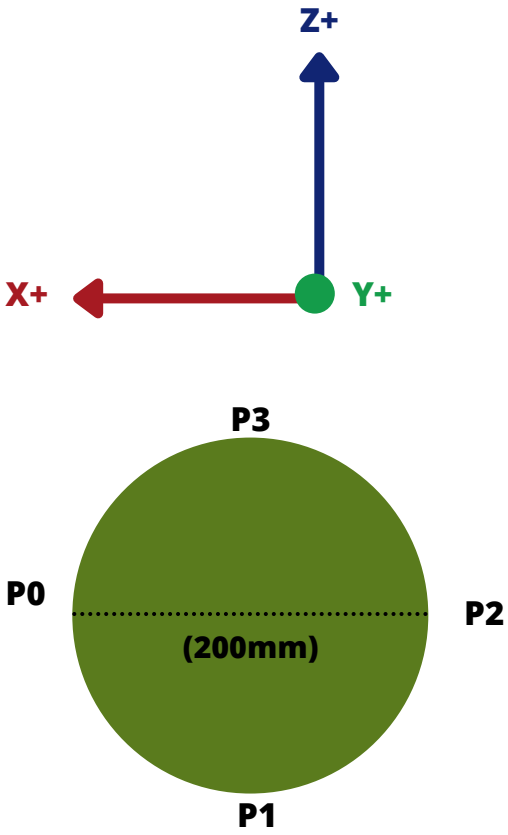



```
SET_OVERRIDE_SPEED 100
E6POS C1={ X  100, Y 168, Z 393.500}
E6POS C2={ X -200, Y 168, Z 393.500}
E6POS C3={ X -200, Y 368, Z 393.500}
E6POS C4={ X  100, Y 368, Z 393.500}

LIN C1 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
LIN C2 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
LIN C3 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
LIN C4 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
```

Fonte: Autor.

Segundo passo: vamos criar um movimento de circulo com raio de 100mm, figura 7.2.

Figura 7.2 - Sistema de coordenada e o circulo.



Fonte: Autor.

Depois de criado os pontos P0, P1, P2 e P3 podemos testar o código abaixo.

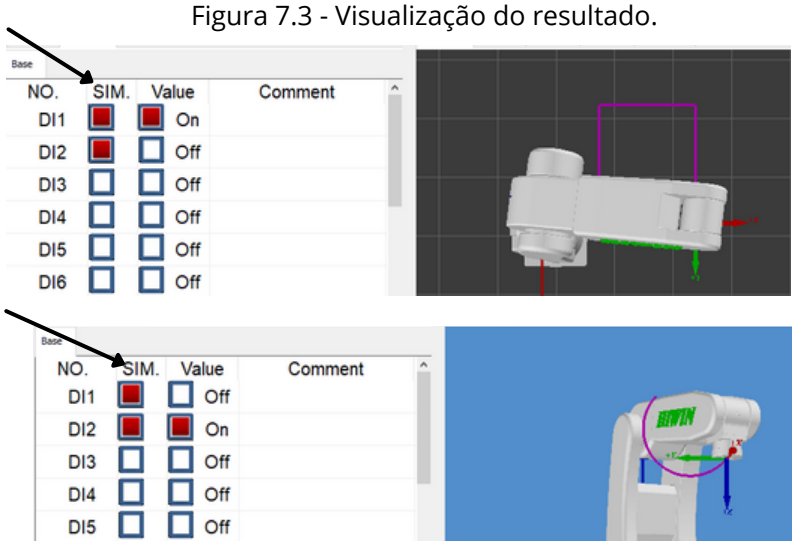
```
CIRC P1 P2 CONT Vel=100mm/s Acc=100% TOOL[0] BASE[0]
CIRC P3 P0 CONT Vel=100mm/s Acc=100% TOOL[0] BASE[0]
```

Adicionando o acionamento pelas entradas digitais de acordo com o código abaixo.

```
SET_OVERRIDE_SPEED 100
E6POS C1={ X  100, Y 168, Z 393.500}
E6POS C2={ X -200, Y 168, Z 393.500}
E6POS C3={ X -200, Y 368, Z 393.500}
E6POS C4={ X  100, Y 368, Z 393.500}

WHILE 1==1
   IF $DI[1] == TRUE THEN
      LIN C1 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
      LIN C2 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
      LIN C3 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
      LIN C4 FINE=1 Vel=1000mm/s Acc=50% TOOL[0] BASE[0]
   ENDIF
   IF $DI[2] == TRUE THEN
      CIRC P1 P2 CONT Vel=100mm/s Acc=100% TOOL[0] BASE[0]
      CIRC P3 P0 CONT Vel=100mm/s Acc=100% TOOL[0] BASE[0]
   ENDIF
ENDWHILE
```

Obteremos o resultado da figura 7.3.

Figura 7.3 - Visualização do resultado.



Fonte: HRSS 3.3.20.9452 32 bit.

Observem na figura 7.3 umas setas na coluna SIM. Estas caixas devem estar habilitadas para podermos simular as entradas e saídas digitais. Caso, não estejam não será possível simular as variáveis digitais, por exemplo.

O link abaixo tem o vídeo demonstrando como realizar a programação passo a passo os movimentos, a partir do QRcode da figura 7.4 ou clique na mão www.

Figura 7.4 - Programação passo a passo.

https://www.youtube.com/watch?v=9xG_4TP5hkw&list=PLqTzJCvIOfIhu7-yOFBIn0GJZezKplQgX&index=7

Fonte: Autor.

8. ACESSANDO REMOTAMENTE O ROBÔ PELA REDE ETHERNET

8. ACESSANDO REMOTAMENTE O ROBÔ PELA REDE ETHERNET

Objetivos:

- Utilizando o programa Putty para controlar os movimentos do Robô utilizando a rede Ethernet.

Vamos seguir o algoritmo simplificado da figura 8.1.

Figura 8.1 - Criação do movimento 1.

```
Inicializa parâmetros de comunicação
Laço:
        Se instrução == verdadeira
                Realiza movimento
                Realiza movimento
        Se instrução == verdadeira
                Realiza movimento
                Realiza movimento
fim_Laço
fecha comunicação
```

Fonte: Autor.

Realize o download do programa Putty para essa prática:

https://www.putty.org/

O código abaixo realiza os movimentos de acordo com as instruções enviadas pelo programa Putty.

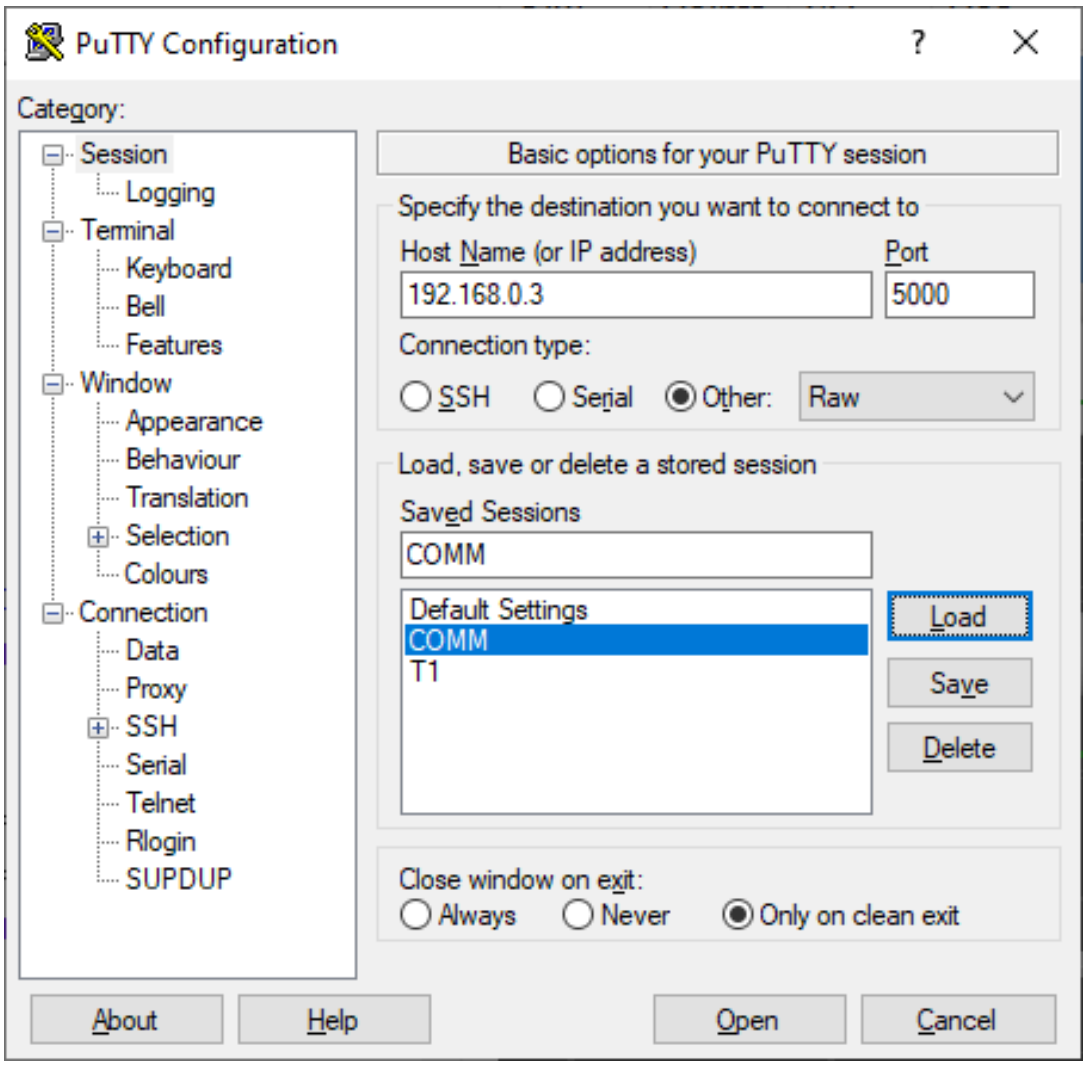
Figura 8.2 - Código para comunicação ethernet.

```
SET_OVERRIDE_SPEED 100
STRING COMANDO, RETORNO
INT HANDLE
; ETH - ETHERNET , SER - SERIAL
COPEN(ETH, HANDLE)
CCLEAR (HANDLE)

LOOP
  CREAD( HANDLE, COMANDO)
  $DO[1] = STRCMP( COMANDO, "INICIA")
  $DO[2] = STRCMP( COMANDO, "MOVIMENTO CIRCULAR")

  IF $DO[1] == TRUE THEN
    PTP P0 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
    PTP P1 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
    RETORNO = "PRONTO MOVIMENTO PTP EXECUTADO"
    CWRITE( HANDLE, RETORNO)
    $DO[1] = FALSE
  ENDIF

  IF $DO[2] == TRUE THEN
    PTP P2 FINE=1 Vel=10% Acc=100% TOOL[0] BASE[0]
    CIRC P3 P4 FINE=1 Vel=100mm/s Acc=100% TOOL[0] BASE[0]
    RETORNO = "PRONTO MOVIMENTO CIRCULAR EXECUTADO"
    CWRITE( HANDLE, RETORNO)
    $DO[2] = FALSE
  ENDIF

  CCLEAR( HANDLE)
ENDLOOP
```

Fonte: Autor.

A seguir a configuração do programa Putty e o seu funcionamento com o HRSS. A figura 8.3 mostra os parâmetros de configuração necessários para a comunicação com o HRSS.

Figura 8.3 - Configuração Putty.

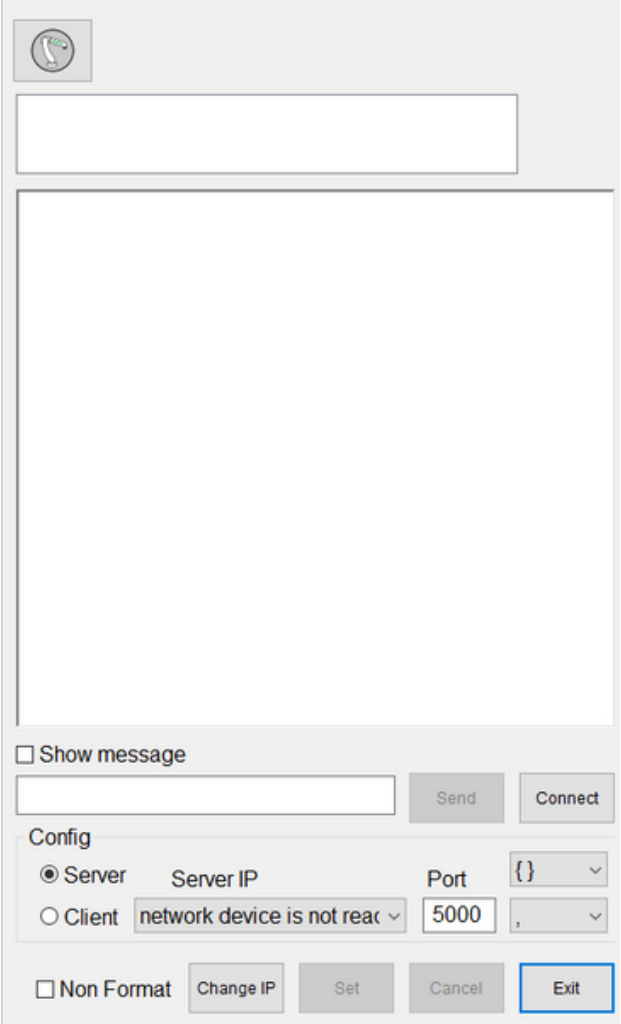


Fonte: Autor.

Pressione o botão Open e digite as instruções esperadas no HRSS para poder realizar os movimentos.

No HRSS possui alguns endereços padrões de comunicação Ethernet. Acesse o **Menu**, depois **Start-up** e clique em **Network Config**. A janela da figura 8.4 aparecerá.

Figura 8.4 - janela de Configuração ethernet.



Fonte: HRSS 3.3.20.9452 32 bit.

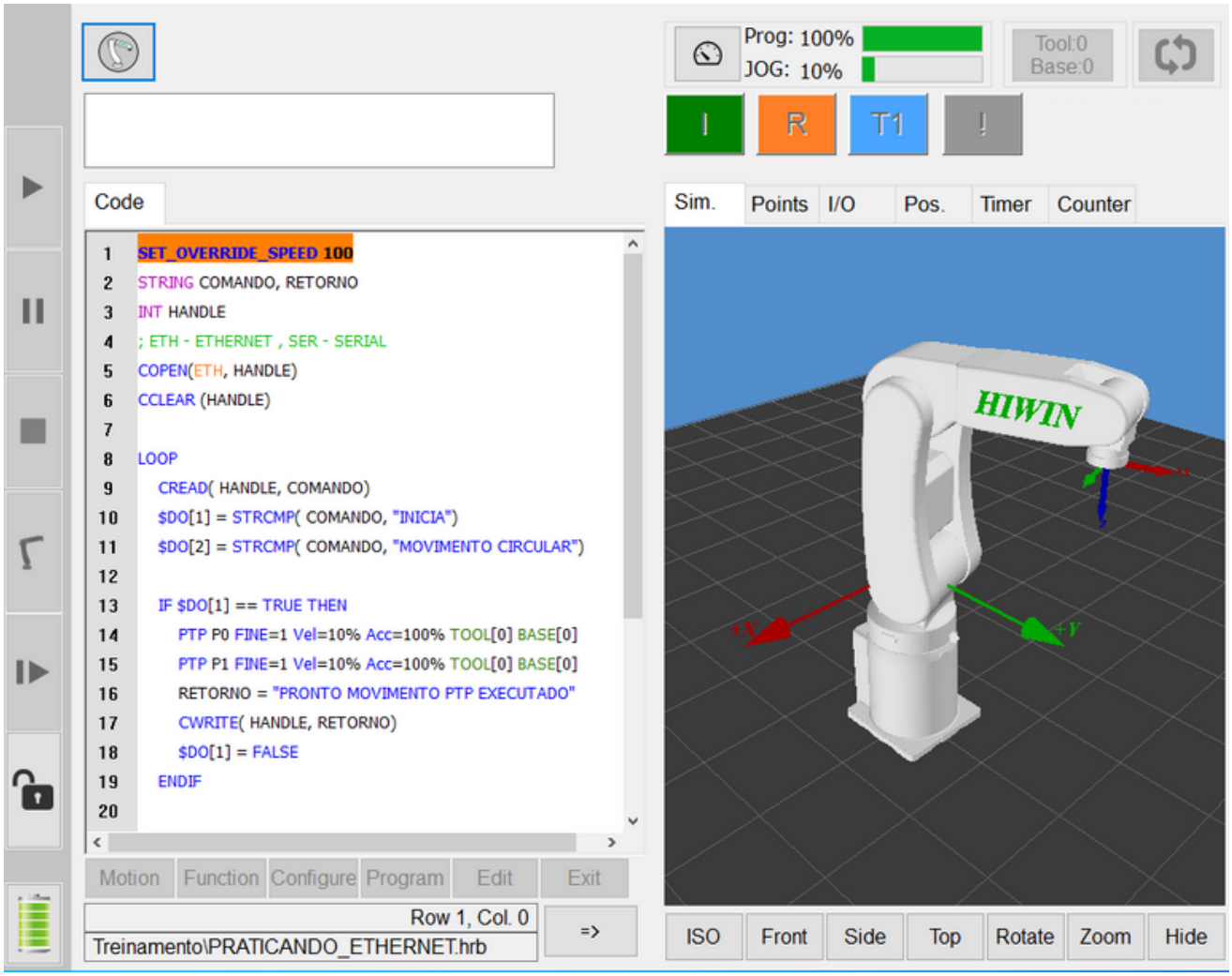
Quando pressiona-se o botão Connect é possível verificar os IP disponíveis no HRSS, figura 8.5.

Figura 8.5 - Lista de IP do HRSS.

Fonte: HRSS 3.3.20.9452 32 bit.

A sequência a seguir, mostra o funcionamento em 3 etapas. A figura 8.6 inicializa o programa HRSS e aguarda a comunicação com o Putty.

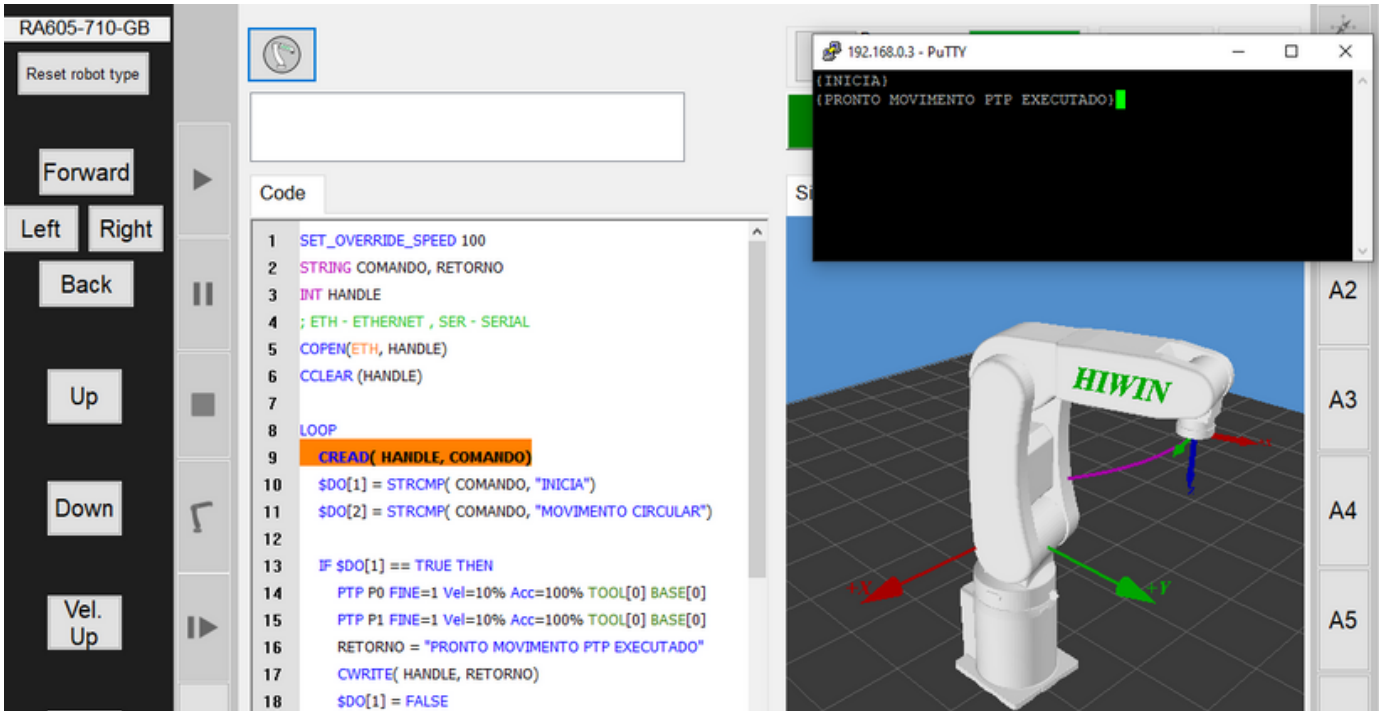
Figura 8.6 - Inicializa o programa e espera a comunicação.



Fonte: HRSS 3.3.20.9452 32 bit.

Realiza o primeiro movimento depois de receber a string **{INICIA}**, figura 8.7.

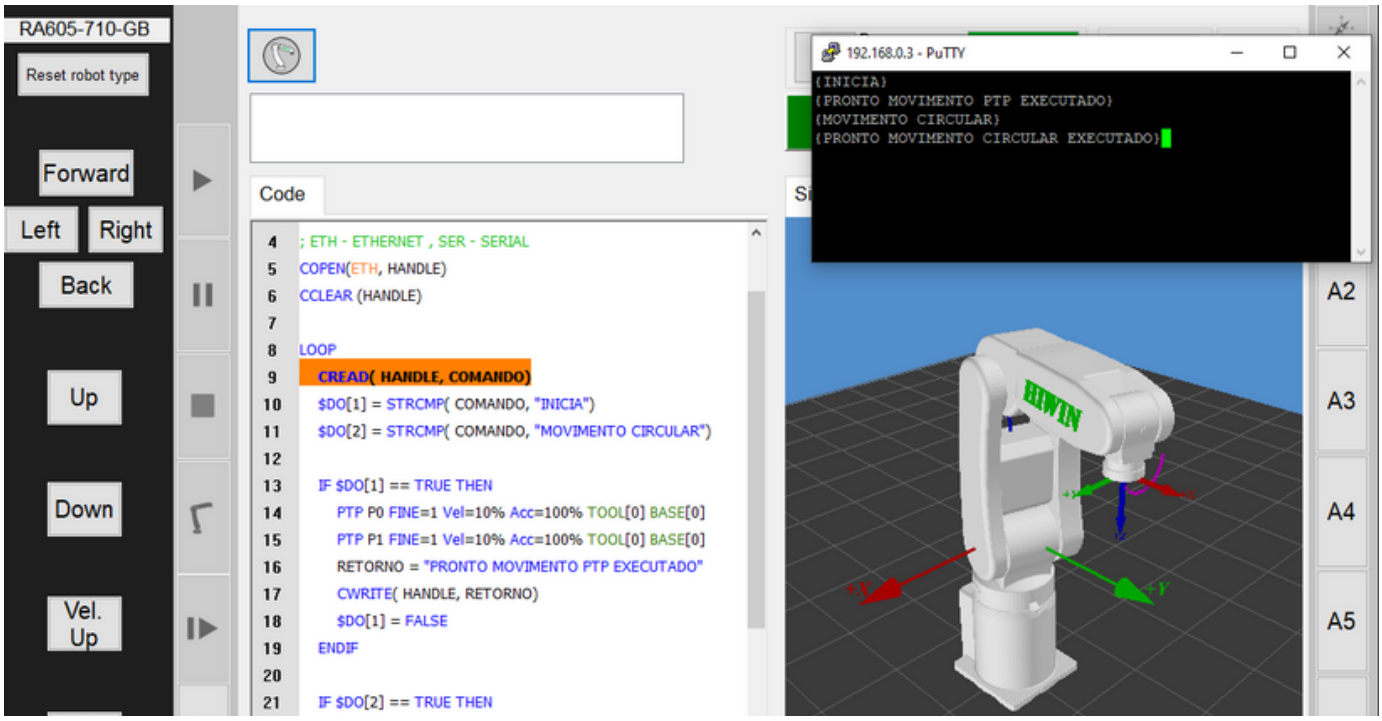
Figura 8.7 - Primeiro movimento.



Fonte: HRSS 3.3.20.9452 32 bit.

Realiza o segundo movimento depois de receber a string **{MOVIMENTO CIRCULAR}**, figura 8.8.

Figura 8.8 - Segundo movimento.



Fonte: HRSS 3.3.20.9452 32 bit.

O link abaixo apresenta o vídeo demonstrando como realizar a programação passo a passo da comunicação do Putty com o HRSS, figura 8.9 ou clique na mão www.

Figura 8.9 - Vídeo demonstrando passo a passo da comunicação.

https://www.youtube.com/watch?v=4cE__TRSIT4&list=PLqTzJCvIOfIhu7-yOFBIn0GJZezKplQgX&index=8

Fonte: Autor.

9. PROGRAMA PALLETIZAÇÃO

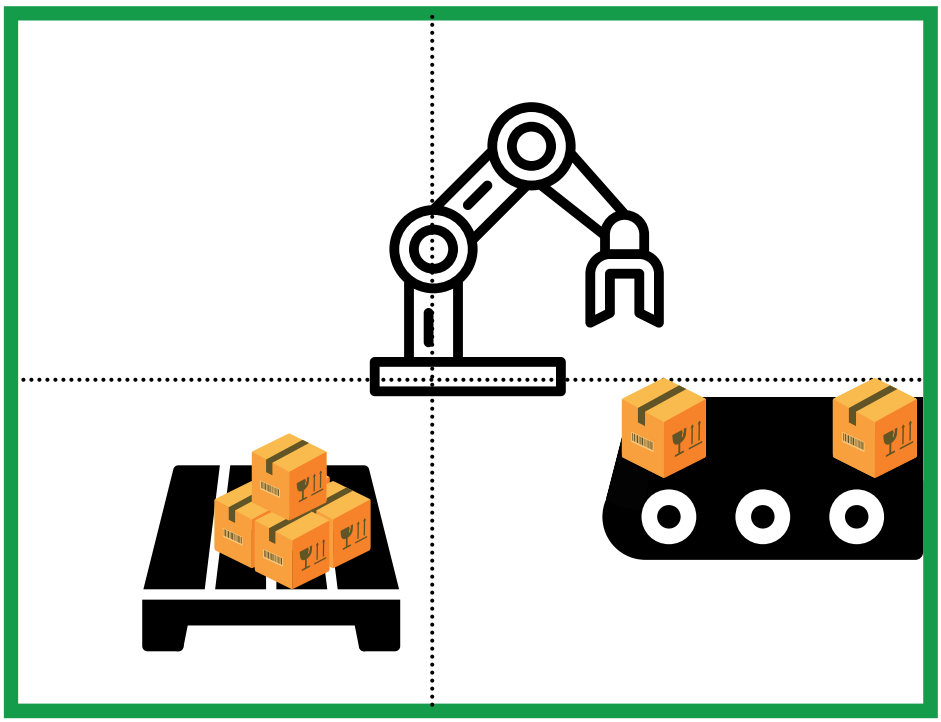
9. PROGRAMA PALLETIZAÇÃO

Objetivos:

- Inserir componentes stl;
- Implementar um programa de Palletização.

Vamos visualizar a disposição dos componentes na área de trabalho, figura 9.1.

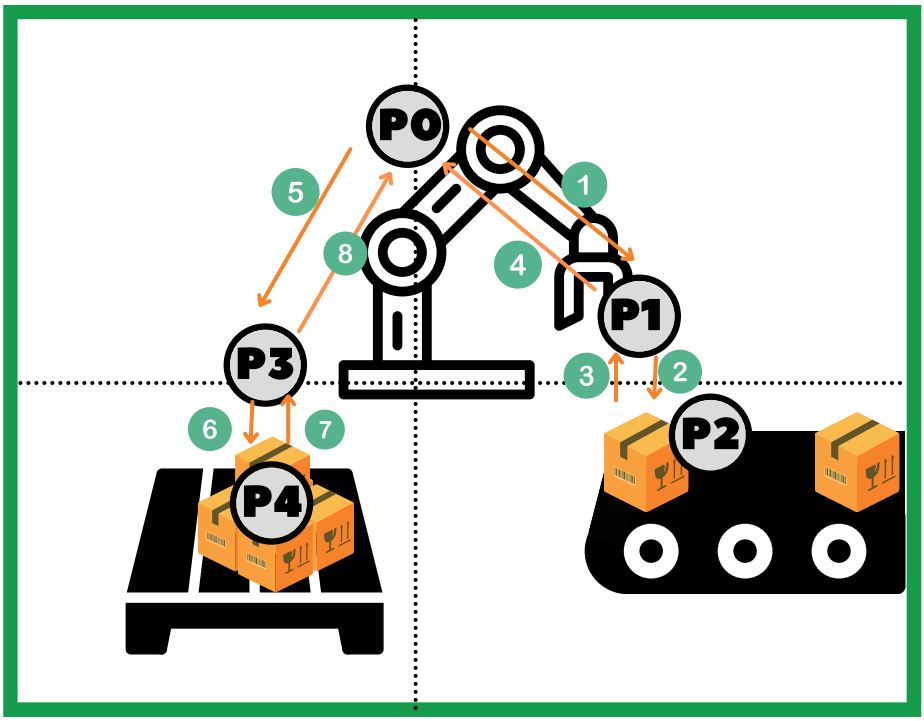
Figura 9.1 - Posição do robô na área de trabalho.



Fonte: Autor.

Vamos planejar os movimentos a partir da disposição dos componentes na área de trabalho, figura 9.2.

Figura 9.2 - Planejamento dos movimentos do robô.



Fonte: Autor.

Agora temos os percursos do manipulador com as posições:

P0 - Posição inicial do manipulador (Manipulador parado) a aguardando iniciar;
P1 - Posição intermediária para descer (Preparar pegar o objeto);
P2 - Posição para Pega o Objeto;
Sobe e Retorna a Posição P1;
Sobe e Retorna a Posição P0;
P3 - Posição intermediária para descer (Preparar deixar o objeto);
P4 - Posição para Deixar o Objeto;

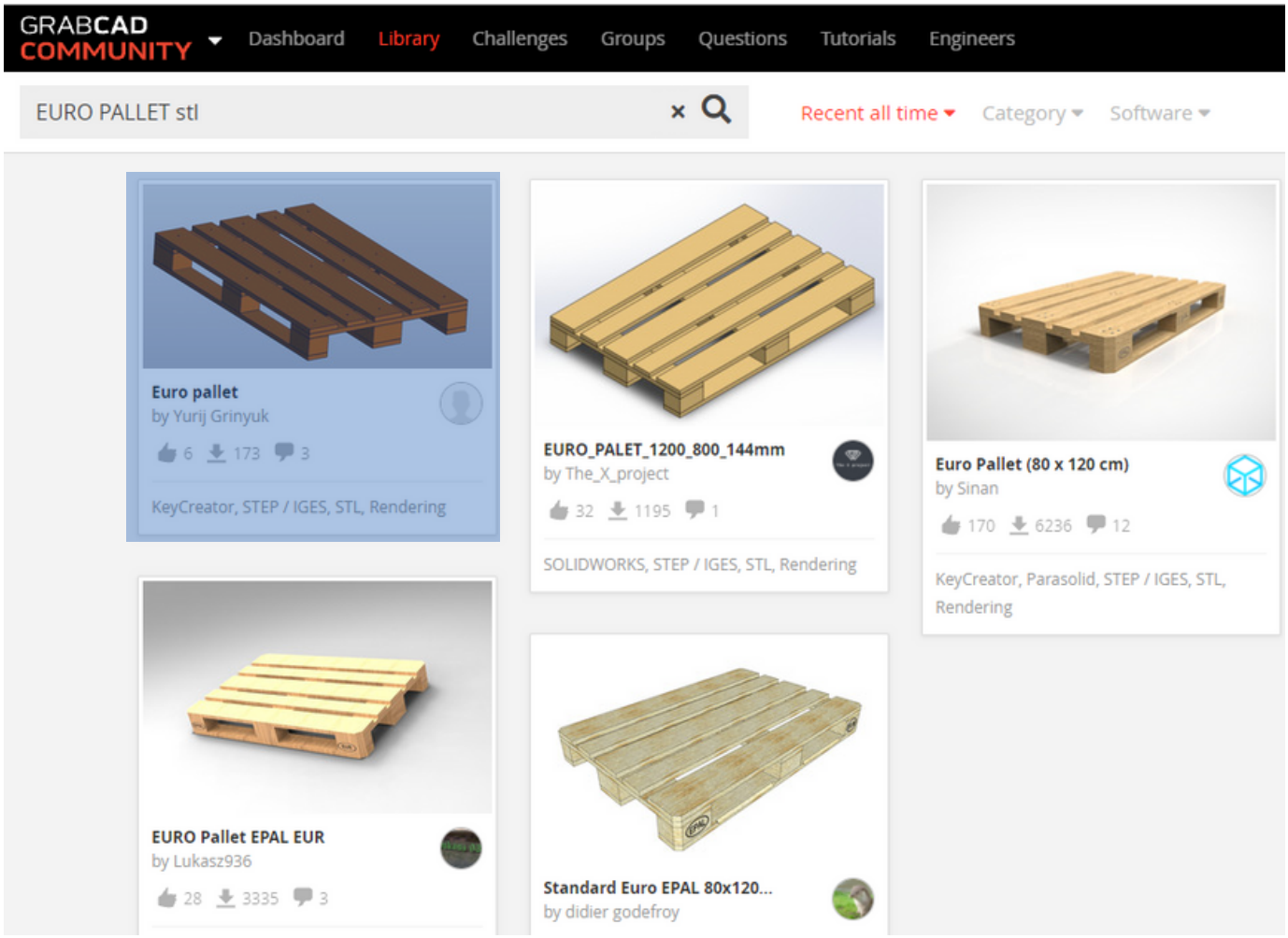
Realizado o planejamentos o próximo passo inserir os componentes na área de trabalho.

Vamos baixar alguns componentes para inserir na área de trabalho e fazer os ajustes.

1. Acesse o site grabcad.com. Caso, não tenha uma conta. Crie uma para poder realizar o download dos arquivos.

Pesquise: **EURO PALLET** (faça o download), figura 9.3.

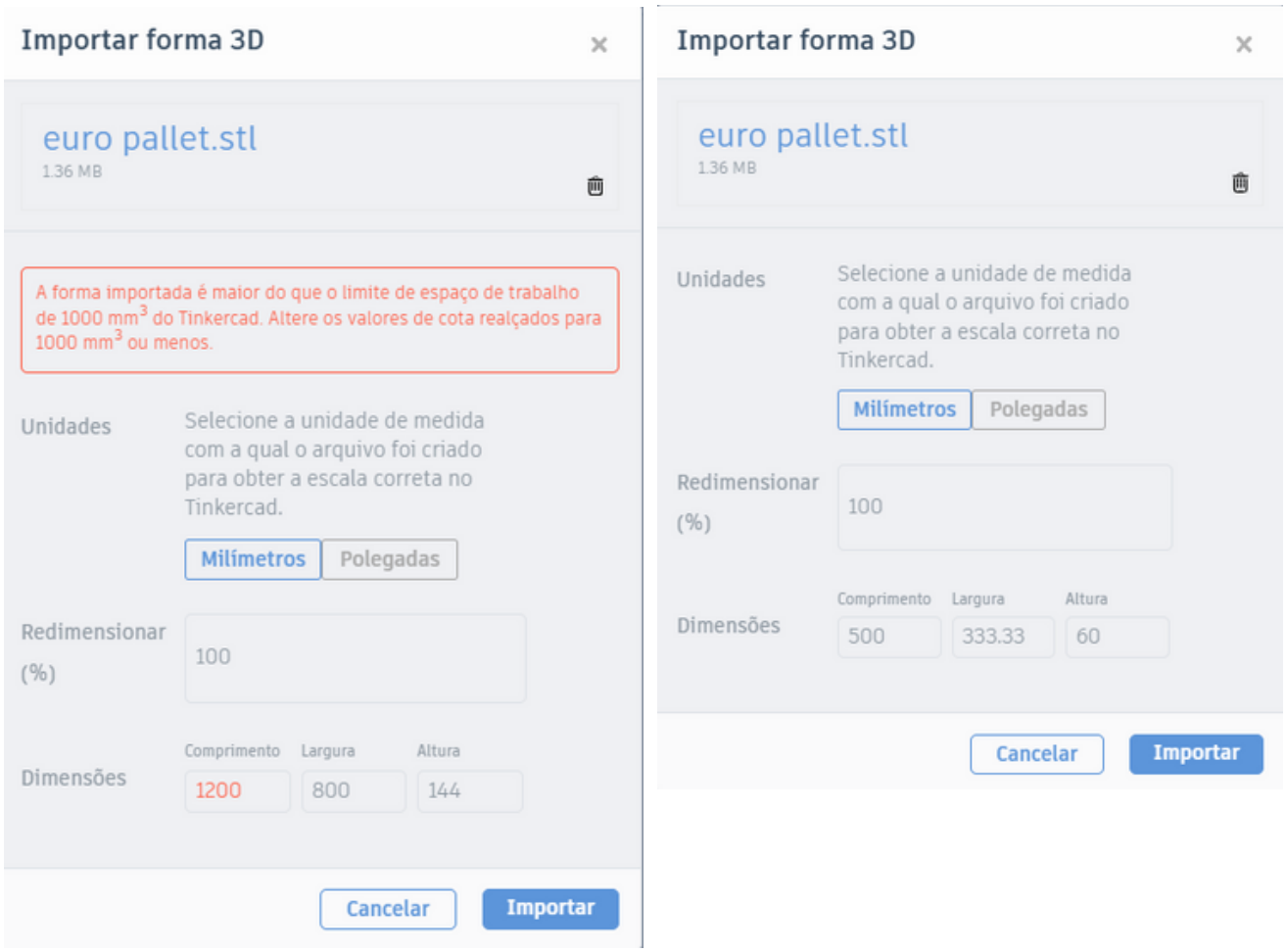
Figura 9.3 - Download do arquivo Euro_pallet.



Fonte: Autor.

Redimensione o arquivo: 1200 (comprimento), 800 (largura) e 144 (altura) para 500(comprimento), 333(largura) e 60 (altura) utilizando o Thinkercard, figura 9.4. www.

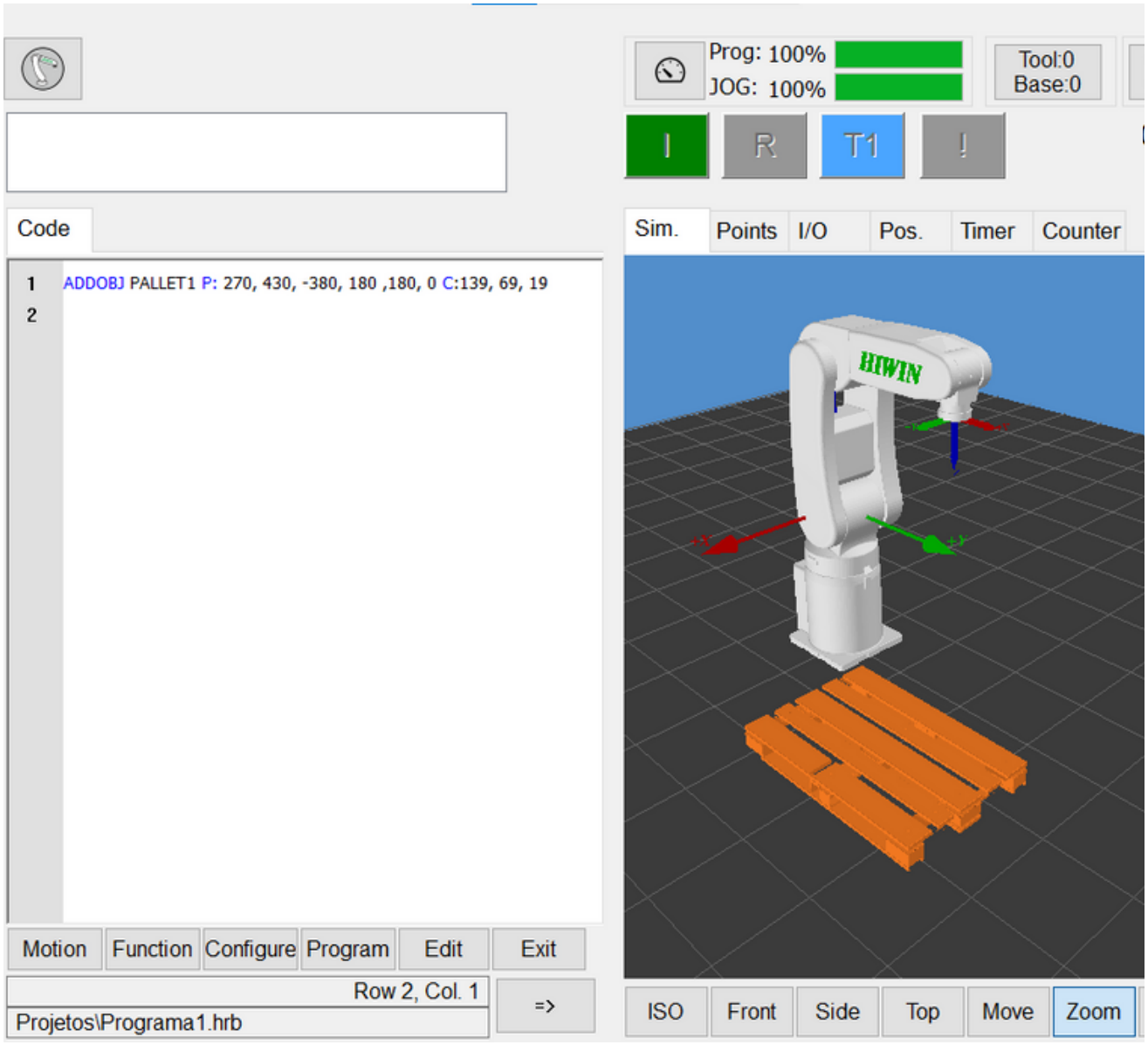
Figura 9.4 - Redimensionamento do arquivo no thinkercard.



Fonte: Autor.

Copiado o arquivo para a pasta **stl** podemos adicionar na área de trabalho com a instrução: **ADDOBJ**, figura 9.5.

Figura 9.5 - Instrução ADDOBJ.



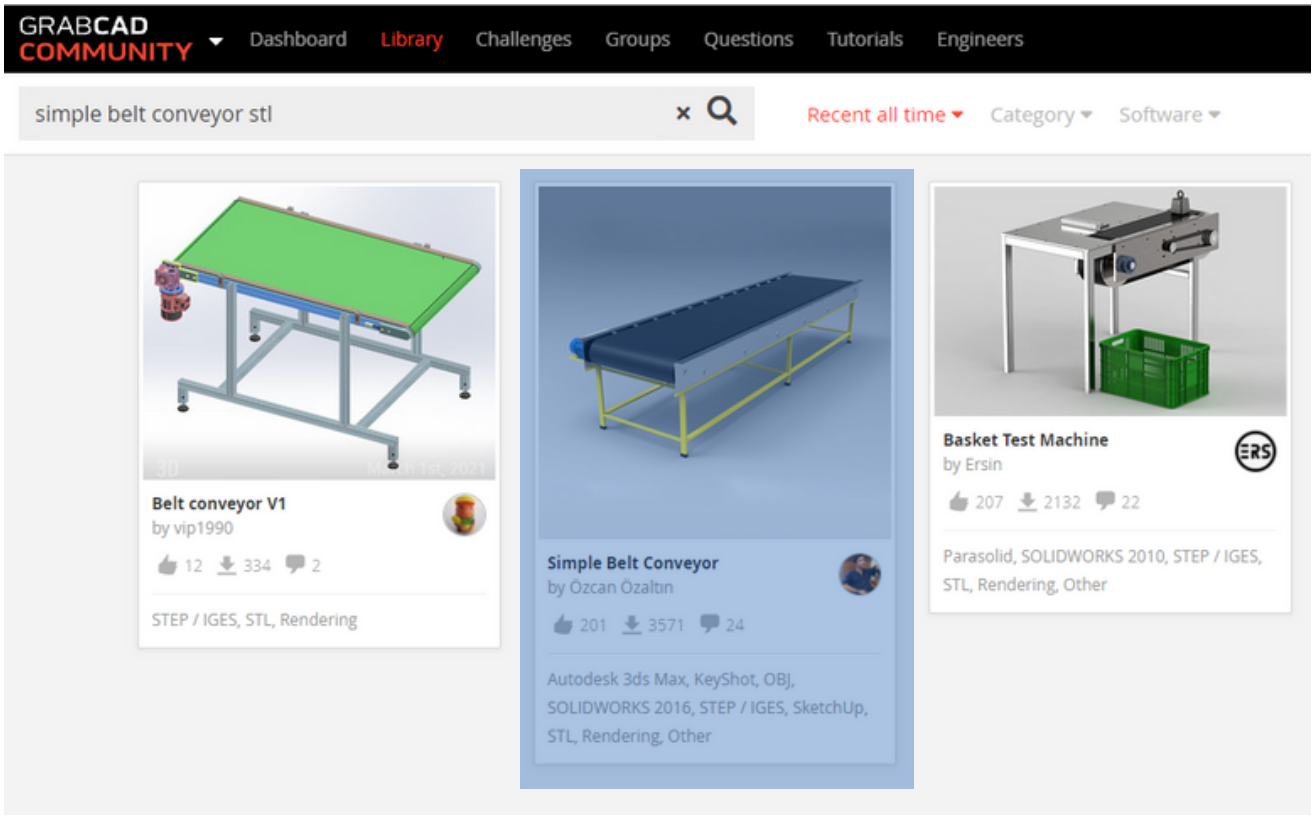
Fonte: HRSS 3.3.20.9452 32 bit.

ADDOBJ PALLET1 P: 270, 430, -380, 180 ,180, 0 C:139, 69, 19

2. Agora vamos pesquisar uma esteira.

Pesquise: **simple belt conveyor stl** (faça o download), figura 9.6.

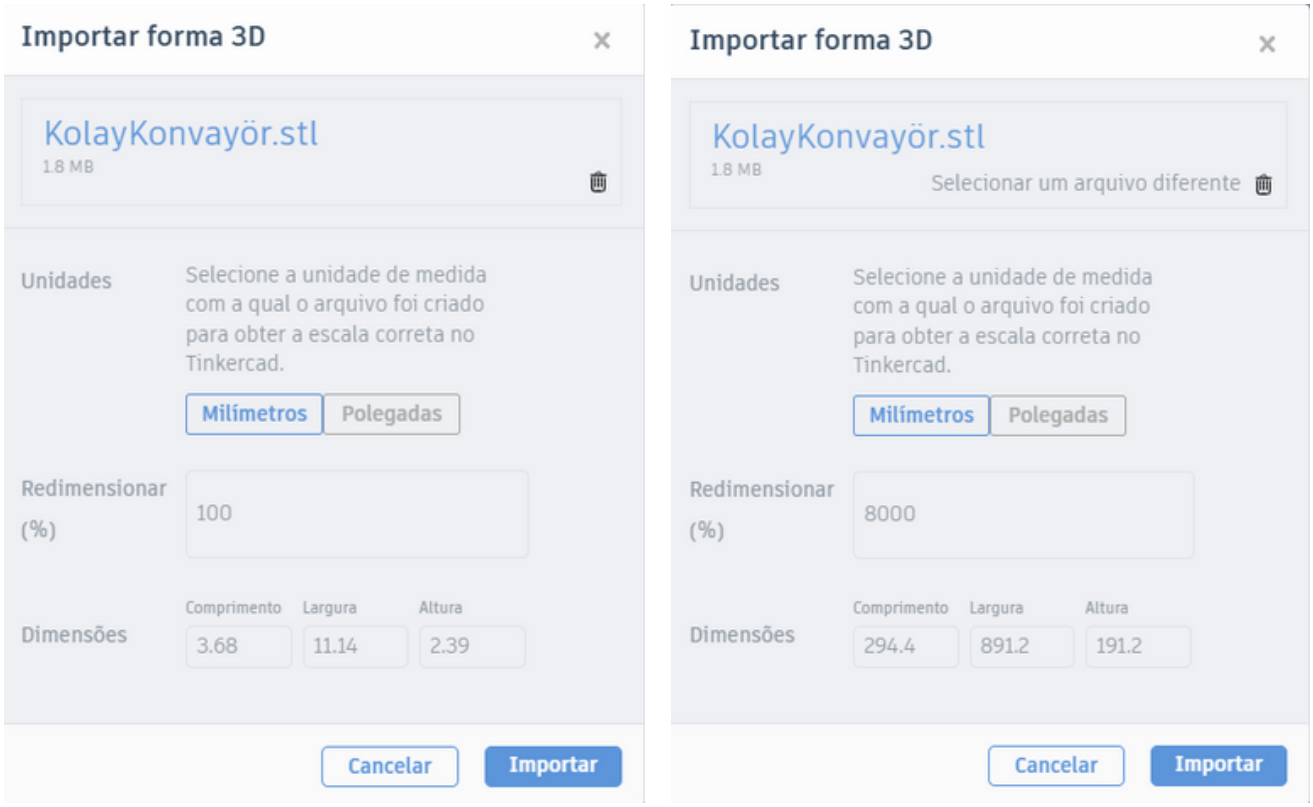
Figura 9.6 - Download do arquivo simple belt conveyor.



Fonte: Autor.

Redimensione o arquivo 3.68 (comprimento) 11.14 (largura) e 2.39 (altura) para 294.4 (comprimento) 891.2 (largura) e 191.2 (altura) aumentando o seu tamanho 8000 vezes a sua proporção com o Thinkercard, figura 9.7.

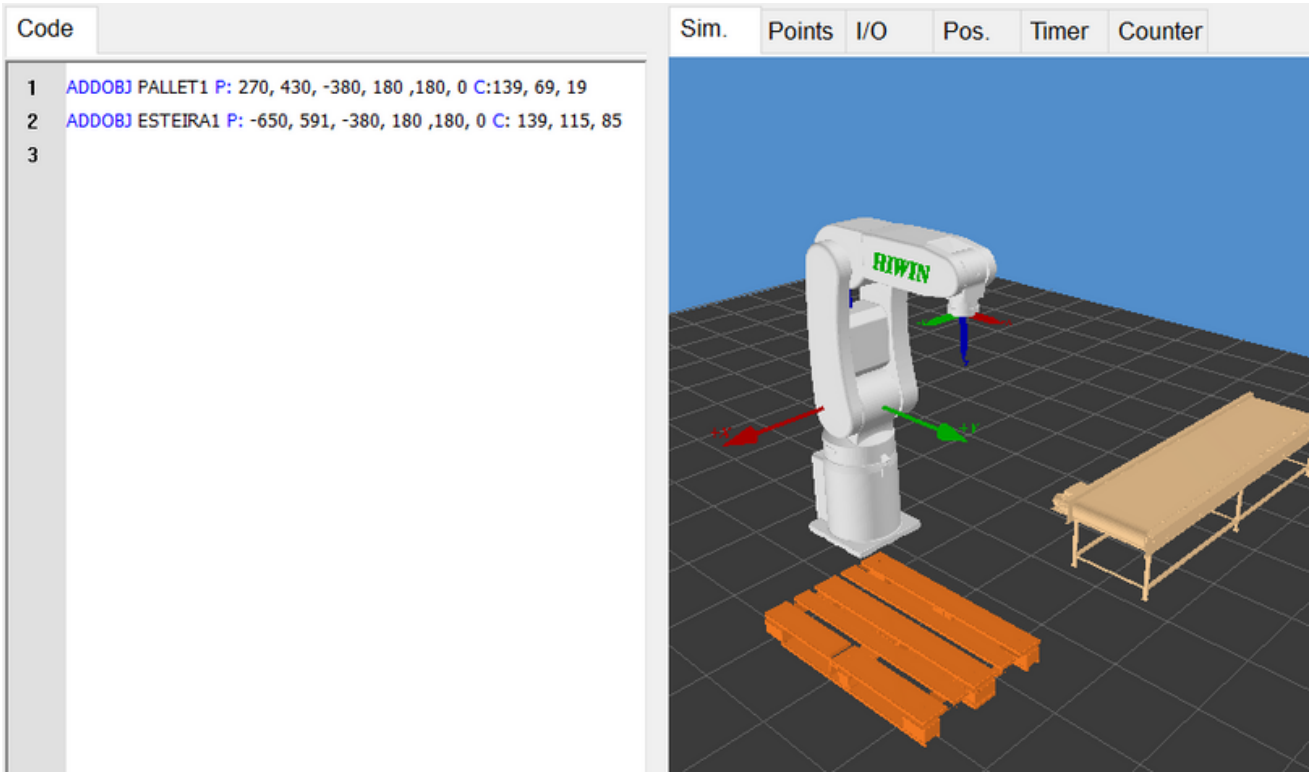
Figura 9.7 - Redimensionar a esteira no thinkercard.



Fonte: Autor.

Copiado o arquivo para a pasta stl podemos adicionar na área de trabalho com a instrução: ADDOBJ, figura 9.8.

Figura 9.8 - Instrução ADDOBJ com a esteira.



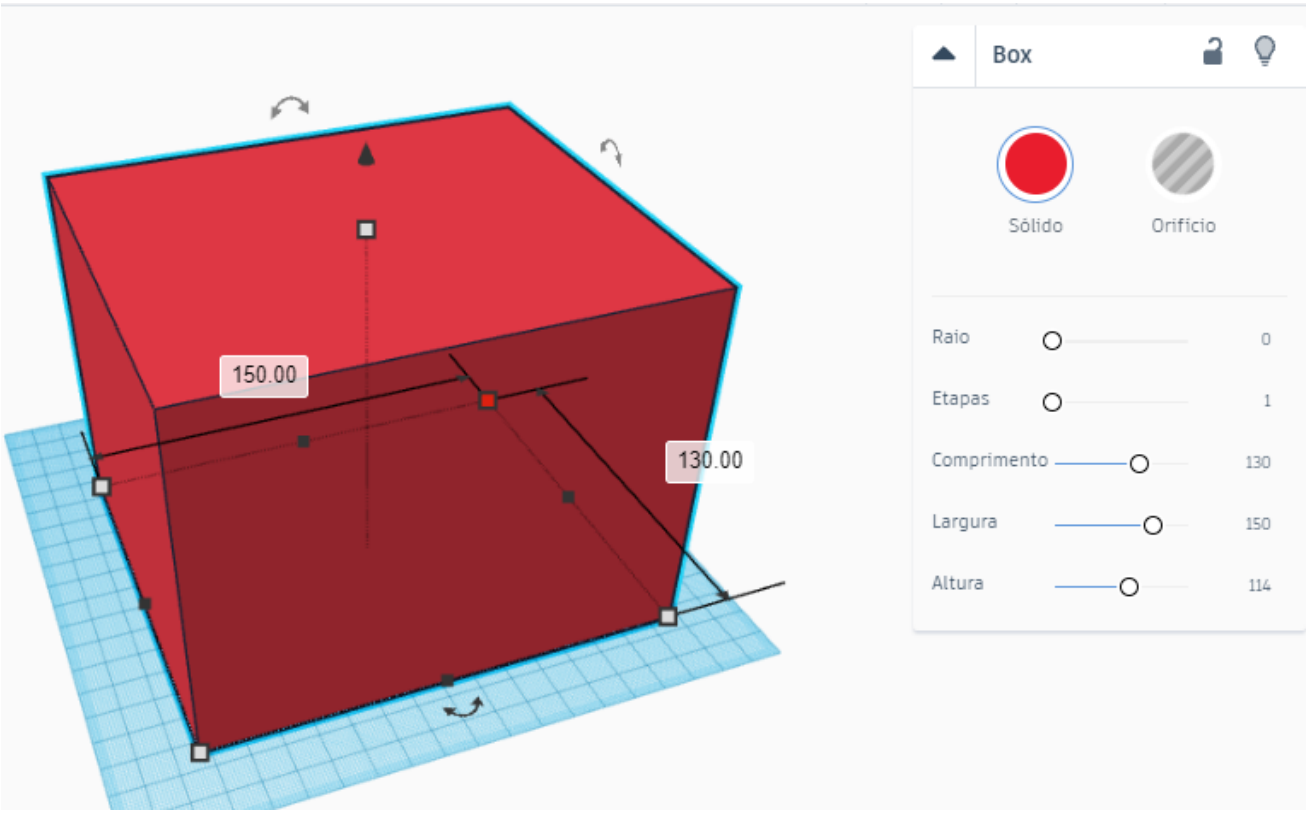
Fonte: HRSS 3.3.20.9452 32 bit.

ADDOBJ PALLET1 P: 270, 430, -380, 180 ,180, 0 C:139, 69, 19
ADDOBJ ESTEIRA1 P: -650, 591, -380, 180 ,180, 0 C: 139, 115, 85

Próximo passo, criar a caixa no *Thinkercar*. Com as seguintes dimensões:

Comprimento: **130**
Largura: **150**
Altura: **114**

Figura 9.9 - Criação da caixa para a esteira.



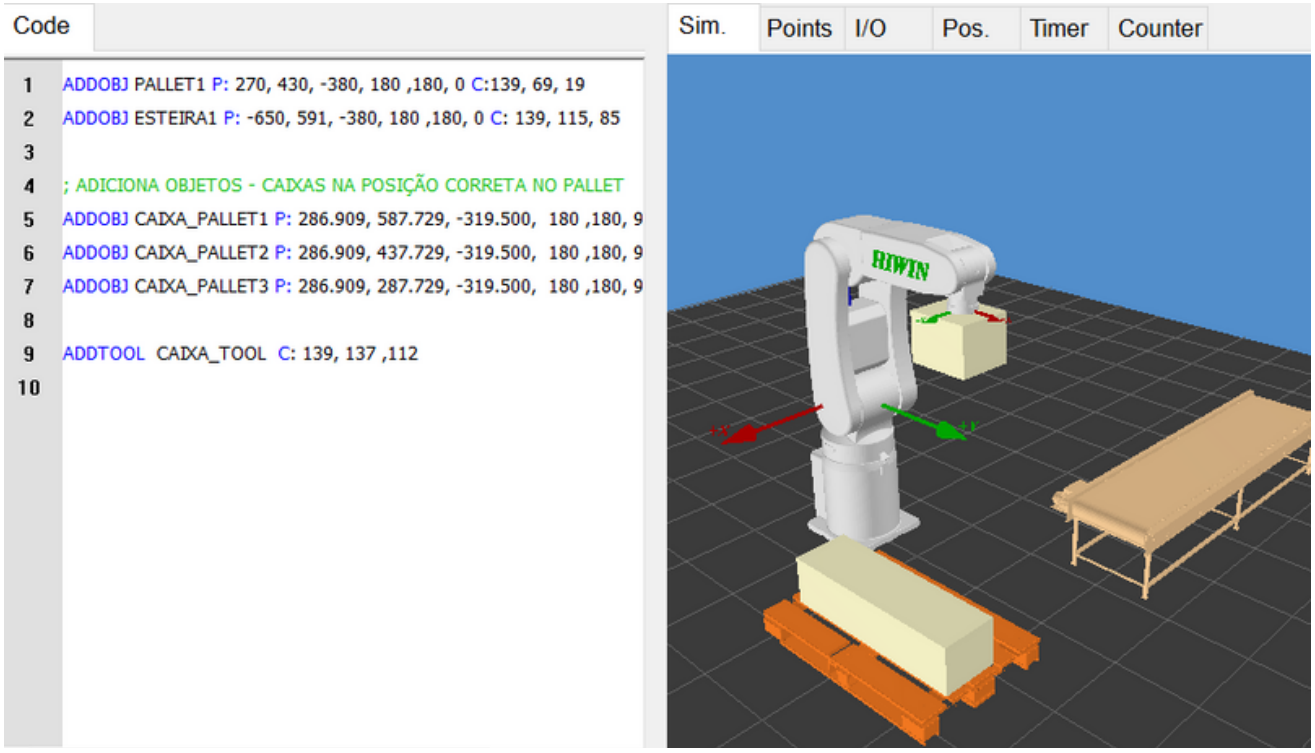
Fonte: Autor.

Em seguida vamos fazer cópias da caixa e renomeá-las. Teremos, então, os seguintes arquivos iguais com os respectivos nomes:

- caixa_esteira
- caixa_pallet1
- caixa_pallet2
- caixa_pallet3
- caixa_tool

Adicionando os objetos na área de trabalho, figura 9.10.

Figura 9.10 - Caixas na área de trabalho.



Fonte: HRSS 3.3.20.9452 32 bit.

```
ADDOBJ PALLET1 P: 270, 430, -380, 180 ,180, 0 C:139, 69, 19
ADDOBJ ESTEIRA1 P: -650, 591, -380, 180 ,180, 0 C: 139, 115, 85

; ADICIONA OBJETOS - CAIXAS NA POSIÇÃO CORRETA NO PALLET
ADDOBJ CAIXA_PALLET1 P: 286.909, 587.729, -319.500,  180 ,180, 90 C: 139, 137 ,112
ADDOBJ CAIXA_PALLET2 P: 286.909, 437.729, -319.500,  180 ,180, 90 C: 139, 137 ,112
ADDOBJ CAIXA_PALLET3 P: 286.909, 287.729, -319.500,  180 ,180, 90 C: 139, 137 ,112

ADDTOOL  CAIXA_TOOL  C: 139, 137 ,112
```

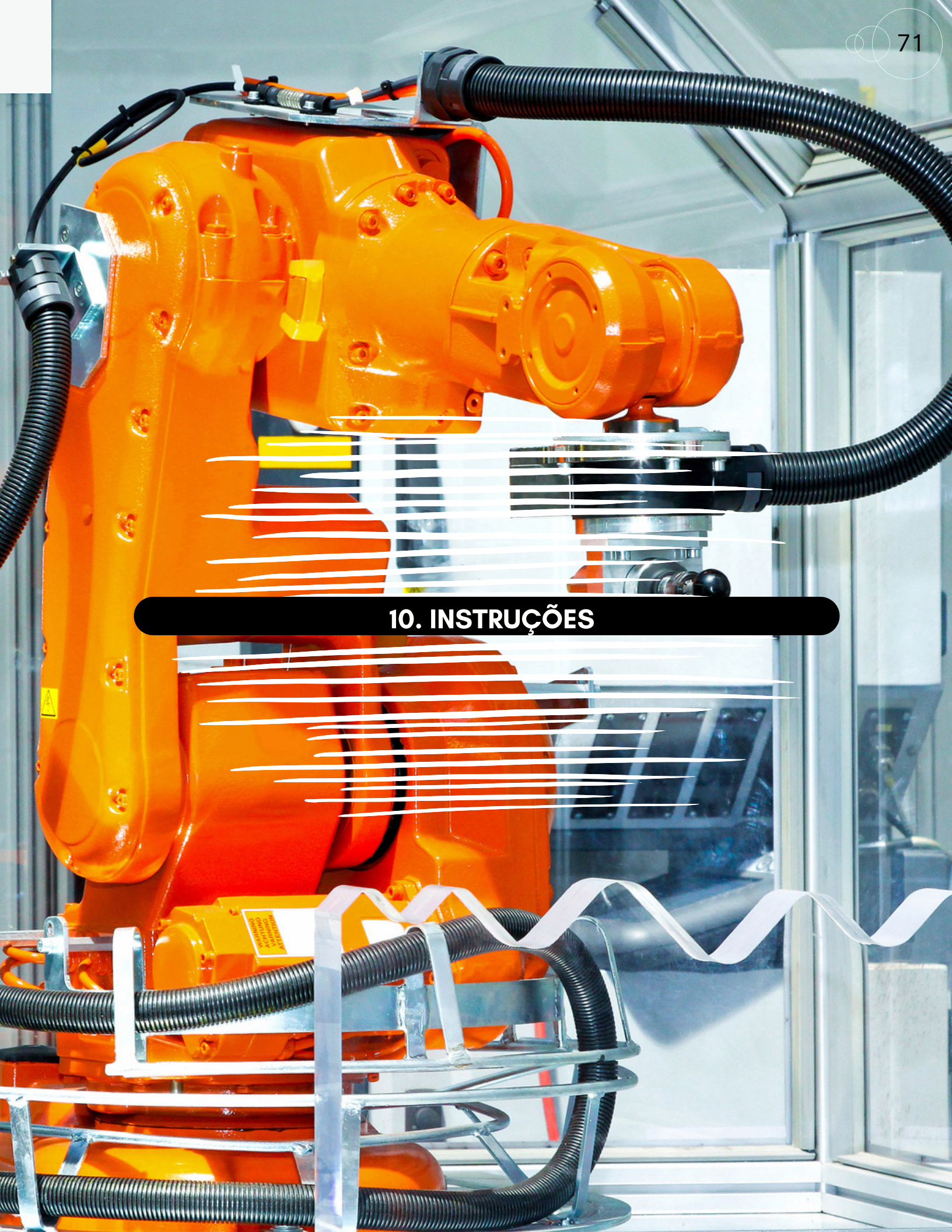
O código abaixo, implementa a sequência de planejamento desejado.

```
MOVEFLOOR 0
SET_OVERRIDE_SPEED 100

; ADDOBJ NOME P: X, Y, Z, A, B, C (SaddleBrown)
ADDOBJ PALLET500 P: 270, 430, -380, 180 ,180, 0 C:139, 69, 19

; ADICIONA OBJETOS - CAIXAS NA POSIÇÃO CORRETA NO PALLET
ADDOBJ CAIXA_PALLET1 P: 286.909, 587.729, -319.500,  180 ,180, 90 C: 139, 137 ,112
ADDOBJ CAIXA_PALLET2 P: 286.909, 437.729, -319.500,  180 ,180, 90 C: 139, 137 ,112
ADDOBJ CAIXA_PALLET3 P: 286.909, 287.729, -319.500,  180 ,180, 90 C: 139, 137 ,112

;;DESABILITA A VISUALIZACAO DAS CAIXAS NO PALLET
SHOW_OBJ CAIXA_PALLET1 FALSE
SHOW_OBJ CAIXA_PALLET2 FALSE
SHOW_OBJ CAIXA_PALLET3 FALSE

ADDOBJ ESTEIRA P: -650, 591, -380, 180 ,180, 0 C: 139, 115, 85
ADDOBJ CAIXA_ESTEIRA P: -300, 610,-192, -180, 180, 0 C: 139, 137 ,112

E6POS P_CX3 = { X 286.909, Y 287.729, Z -209.500, A 180, B 0, C 90}
E6POS P_CX2 = { X 286.909, Y 437.729, Z -209.500, A 180, B 0, C 90}
E6POS P_CX1 = { X 286.909, Y 587.729, Z -209.500, A 180, B 0, C 90}
E6POS P_CX_EST = { X -299.362, Y 600.618, Z -82.908, A 180, B 0, C 90}

;;CRIA A POSIÇÃO HOME DE INICIO DO PROGRAMA
E6POS P_HOME = { X 0, Y 368, Z 172.500, A -180, B 0, C 90}

;; 1º - MOVIMENTO ATÉ A ESTEIRA
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP P_CX_EST FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
ADDTOOL  CAIXA_TOOL  C: 139, 137 ,112
SHOW_OBJ CAIXA_ESTEIRA FALSE
```

Fonte: Autor.

código (continuação)

```
;; COLOCA A CAIXA 1
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP  P_CX1 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
SHOW_OBJ CAIXA_PALLET1 TRUE
SHOW_TOOL CAIXA_TOOL FALSE
;
;;2º - MOVIMENTO ATÉ A ESTEIRA
SHOW_OBJ CAIXA_ESTEIRA TRUE
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP P_CX_EST FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
ADDTOOL  CAIXA_TOOL  C: 139, 137 ,112
SHOW_OBJ CAIXA_ESTEIRA FALSE

;; COLOCA A CAIXA 2
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP  P_CX2 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
SHOW_OBJ CAIXA_PALLET2 TRUE
SHOW_TOOL CAIXA_TOOL FALSE

;;3º - MOVIMENTO ATÉ A ESTEIRA
SHOW_OBJ CAIXA_ESTEIRA TRUE
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP P_CX_EST FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
ADDTOOL  CAIXA_TOOL  C: 139, 137 ,112
SHOW_OBJ CAIXA_ESTEIRA FALSE

;; COLOCA A CAIXA 3
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
PTP  P_CX3 FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
SHOW_OBJ CAIXA_PALLET3 TRUE
SHOW_TOOL CAIXA_TOOL FALSE

;FINALIZA
PTP P_HOME FINE=1 Vel=100% Acc=100% TOOL[0] BASE[0]
```



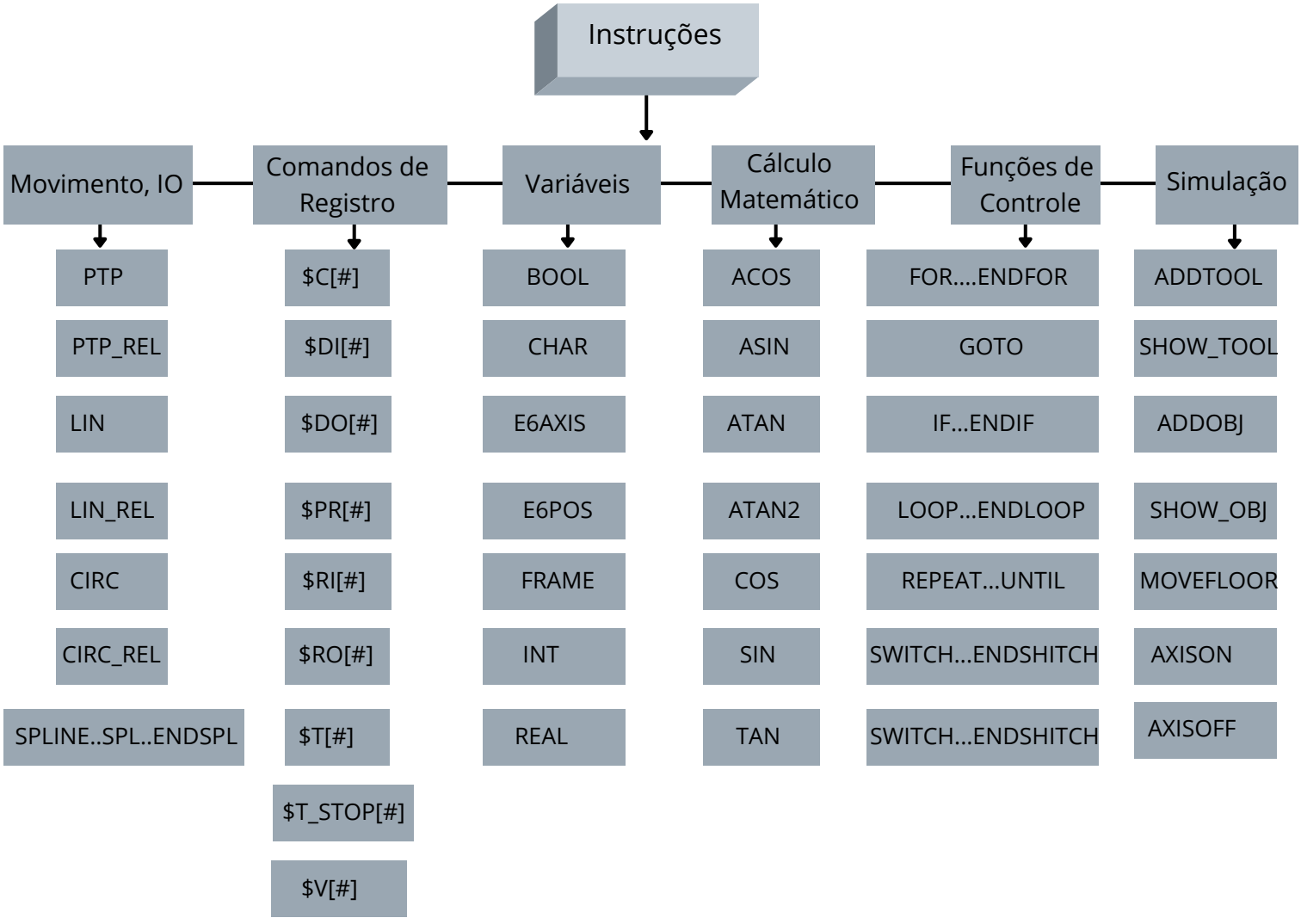
10. INSTRUÇÕES

10. INSTRUÇÕES

Objetivos:

- Resumo das principais instruções para consulta.

As instruções vão ser citadas com um comentário e um exemplo.



MOVIMENTO	Ferramenta final do robô, (TCP) se desloca para um posição registrada		
Sintaxe	INTERPOLAÇÃO <posição> <terminação de movimento> <velocidade> < aceleração> <nº da ferramenta> <coordenada de referência>		
Interpolações	**PTP**	***'Point to point'***. O TCP se desloca de forma livre de um ponto até o próximo ponto gravado. A trajetória do TCP, normalmente descreve um arco, nunca um trajeto linear.	Controle no movimento é "PTP" (*Point to point*) Vel=%
	LIN	***'Linear'***. O TCP se desloca de forma linear de um ponto até o próximo ponto gravado. A trajetória do TCP, descreve um trajeto linear, com velocidade conhecida.	Controle no movimento é "CP" (*Continuous Path*) Vel= mm/s
	CIRC	***'Circular'***. O TCP se desloca em um caminho circular. A trajetória do TCP, descreve um trajeto em arco, com velocidade conhecida. Inicia em um ponto conhecido, passa por um ponto auxiliar, até alcançar o ponto final. A partir da posição inicial do trajeto, o primeiro ponto é o auxiliar e o segundo ponto o final do trajeto.	Controle no movimento é "CP" (*Continuous Path*) Vel= mm/s
	SPLINE	***'Spline'***. O TCP se desloca em um caminho circular, com variação de raio (elipse). A trajetória do TCP, descreve um trajeto em arco, com velocidade conhecida. Inicia em um ponto conhecido, passa por um ponto auxiliar, até alcançar o ponto final.	Controle no movimento é "CP" (*Continuous Path*) Vel= mm/s
Parâmetros	posição	Posição registrada na gravação	P Nº
	Terminação de movimento	CONT e FINE, continuidade e descontinuidade de movimento. FINE o TCP irá passar e parar no ponto ensinado. CONT o TCP não passará pelo ponto a fazer um pequeno desvio, suave e depende da velocidade e aceleração. As opções que fornece % ou mm, definem um caminho fixo.	FINE = 1 ~ 3 CONT; CONT = #%; CONT= #mm
	velocidade		2000 mm/s
	aceleração		0 ~ 100%
	TOOL	Coordenada de ferramenta utilizada	0 ~ 15
	BASE	Coordenada base utilizada	0 ~ 31
Exemplos	**PTP** P1 CONT=3mm Vel=100% Acc=100% TOOL[0] BASE[0] (desvio inicia 3mm antes de chegar no ponto P1 e retoma o trajeto para o próximo ponto) **LIN** P1 CONT=100% Vel=1000mm/s Acc=1000mm/s TOOL[0] BASE[0] (desvio no ponto P1, até o próximo, o máximo possível, resulta em economia de tempo de execução) **CIRC** P1 P2 FINE=1Vel = 2000mm/s Acc=50% TOOL[0] BASE[0] (sem desvio, passa exatamente no ponto e resulta em uma pequena parada de movimento do trajeto) **SPLINE** **SPL** P1 **SPL** P2 **SPL** P3 **ENDSPLINE** CONT=100% Vel = 2000mm/s Acc=50% TOOL[0] BASE[0]		

REGISTRO – I/O	Funcionam como variáveis, podem ser utilizados como contador, temporizador. I/O atua como comandos de lógica.		
Sintaxe	CONFIGURAÇÃO -Configure-> Variable -> Counter CONFIGURAÇÃO -Configure-> Variable -> Stop Timer CONFIGURAÇÃO -Configure-> Output -> Digital ou outra interface de I/O de saída CONFIGURAÇÃO -Configure-> Variable ->		
Comandos	**#C[#]**	*REGISTRO DE CONTADOR*	
	#DI[#]	*REGISTRO DE ENTRADA DIGITAL*	
	#DO[#]	*REGISTRO DE SAÍDA DIGITAL*	
	#PR[#]	*REGISTRO DE ENTRADA ROBOT*	
	#RI[#]	*REGISTRO DE ENTRADA REMOTO*	
	#RO[#]	*REGISTRO DE SAÍDA REMOTO*	
	#T[#]	*REGISTRO DE INÍCIO DE TEMPO*	
	#T_STOP[#]	*REGISTRO DE VALOR DE SAÍDA(PARADA)*	
	#VO[#]	*REGISTRO CONTADOR*	
Parâmetros	$C[**n**]= **...**	**...** = *Valor salvo no contador "n";* *(a constante valor compreende a faixa -2147483648; +2147483648 - 32 bits)*	[**n**] =1 ~ 20
	$DI[**n**]== ...	**FALSE**=*valor 'off' ou '0'* ; **TRUE**=*valor 'on' ou '1'.*	Padrão, 16 entradas e 16 saídas.
	$T[**n**]=0 $T_STOP[**n**]=...	*Tempo armazenado na variável* *(-2147483648(ms) - +2147483648(ms) - 32 bits)* **FALSE**=*início de contagem*; **TRUE**=*Término de contagem.*	[**n**] =1 ~ 20
Exemplos	**$C** [20]=0 ; *(N° de registro da variável contador [20], valor zerado, limpo).* **$C** [20]= **$C** [20]+1 ; *(incrementando o valor de contagem, em uma unidade).*		
	$DI[1]== TRUE *;(Canal Digital 1 de entrada acionada).* **$DO[1]**== FALSE *;(Canal Digital 1 de saída desacionada).* **$RI[5]**== TRUE *;(Canal do robô 5 de entrada, acionada).* **$RO[5]**== FALSE *;(Canal do robô 5 de saída, desacionada).* **$VO[2]**== TRUE *;(Canal da válvula solenoide 2 de saída, acionada).*		
	$PR [1]** = A; (declaração de "A", é feita no início do programa. Neste caso é o tipo de variável coordenada ; E6POS A = {X 10, Y 10, Z 10, A 10, B 10, C 10}. **$PR [1] = GETPOINT; (GETPOINT adquire os valores de coordenadas e ângulos da posição atual do robô); LIN **$PR [1]** ; (Uso do registrador [posição e orientação] para executar um movimento linear); CIRC **$PR [1]** **$PR**[2] ; (O uso dos dois registradores se refere ao ponto auxiliar e ponto final).		
	$T [1]=0 ; *(N° de registro da variável de tempo [1]).* **WAIT SEC** 0 **$T_STOP** [1] = FALSE *; (início de contagem).* PTP P1 CONT=100% Vel=100% Acc=100% TOOL[0] BASE[0] *; (calcula o período que o robô móvel da posição original até o P1).* **WAIT SEC** 0 **$T_STOP** [1] = TRUE *; (término de contagem).* *Após fechar o programa o valor da variável estará armazenado no registrador.*		

<table>
<tr><td>VARIÁVEIS</td><td colspan="3">Tipos de variáveis, semelhante aos tipos de dados de dados decimal. Estas variáveis desaparecerão após o programa ser fechado.</td></tr>
<tr><td>Sintaxe</td><td colspan="3">CONFIGURAÇÃO - Configure-> Variable -> Tipo de variável
VARIÁVEIS <nome da variável> <valor da variável></td></tr>
<tr><td rowspan="8">Variáveis</td><td>BOOL</td><td>Tipo de Variável Booleana. Desaparece após o programa ser fechado.</td><td>'True' ou 'False'.</td></tr>
<tr><td>CHAR</td><td>Tipo de variável Caractere. Desaparece após o programa ser fechado.</td><td>Representa caractere específico.</td></tr>
<tr><td>E6AXIS</td><td>Tipo de variável de valor angular. Define uma posição na coordenada de junta</td><td>{A1 n, ~ A6 n}</td></tr>
<tr><td>E6POS</td><td>Tipo de variável de Coordenada. Define uma posição na coordenada cartesiana</td><td>{X n, Y n, Zn, A, B,C}</td></tr>
<tr><td>E6POINT</td><td>Tipo de variável Angular ou cartesiana. Define uma posição na coordenada cartesiana ou coordenada de junta</td><td></td></tr>
<tr><td>FRAME</td><td>Tipo de coordenada que define um Sistema de referência (BASE/TOOL)</td><td></td></tr>
<tr><td>INT</td><td>Tipo de variável inteira. Capacidade de armazenagem 32 bit.</td><td>-2147483648 +2147483648</td></tr>
<tr><td>REAL</td><td>Tipo variável de ponto real. Capacidade de armazenagem 32 bit. 6 dígitos após o ponto decimal</td><td>10^{-37} ~ 10^{-38}</td></tr>
<tr><td rowspan="6">Exemplos</td><td colspan="3">BOOL K
K = TRUE</td></tr>
<tr><td colspan="3">CHAR COLOR
COLOR = 'R'</td></tr>
<tr><td colspan="3">E6POS POINT = {X 0, Y 300, Z 200} (A variável deve ser declarada no início do programa); (Se os parâmetros angulares não são definidos, eles não serão modificados (A,B,C);
PTP POINT CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0] (POINT definido no coordenada cartesiana e o robô é deslocado para a posição armazenada em POINT, com movimento PTP);</td></tr>
<tr><td colspan="3">E6AXIS POSIÇÃO = {A1 90} (A variável deve ser declarada no início do programa); (Se os parâmetros A2, A3, A4, A5 e A6 não são definidos, eles não serão modificadas;
PTP POSIÇÃO CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0] (POSIÇÃO definido no coordenada de junta e o robô é deslocado para a posição armazenada em POSIÇÃO, com movimento PTP);</td></tr>
<tr><td colspan="3">E6POINT POINT1 = {Y 300, Z -2000, A 90} (A variável deve ser declarada no início do programa); (Neste caso a variável foi declarada como coordenada cartesiana);
PTP POINT1 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0] (POINT1 definido na coordenada cartesiana e o robô é deslocado para a posição armazenada em POINT1, com movimento PTP);
E6POINT POINT1 = {A1 90} (A variável deve ser declarada no início do programa); (Neste caso a variável foi declarada como coordenada de junta);
PTP POINT1 CONT=100% Vel=100% Acc=50% TOOL[0] BASE[0] (POINT1 definido na coordenada de junta e o robô é deslocado para a posição armazenada em POINT1, com movimento PTP);</td></tr>
<tr><td colspan="3">INT I
I = 0</td></tr>
<tr><td colspan="3">REAL R
R = 0</td></tr>
</table>

CÁLCULO MATEMÁTICO	Funções de cálculo trigonométrico		
Sintaxe	COMANDO <ângulo> <ângulo>		
Comandos	**ACOS**	Arco cosseno (X)	
	ASIN	Arco seno (X)	
	ATAN	Arco tangente (X)	
	ATAN2	Arco tangente (X, Y)	
	COS	Cosseno	
	SIN	Seno	
	TAN	Tangente	
Exemplo	REAL teste Teste = ACOS (0) Teste = ASIN (0) Teste = ATAN (0) Teste = ATAN2 (0, 1) Teste = COS (0) Teste = SIN (0) Teste = TAN (0)		

SIMULAÇÃO	Instruções do ambiente de simulação gráfica		
Sintaxe	Instruções inseridas, nas linhas de instruções do programa, através de digitação no teclado.		
Funções	**ADDTOOL**	Inserir a ferramenta de final de braço do robô.	Arquivo .stl
	SHOW_TOOL	Mostrar ou esconder a ferramenta no robô.	Arquivo .stl
	ADDOBJ	Acrescenta uma objeto no ambiente virtual	Arquivo .stl
	SHOW_OBJ	Mostrar ou esconder o objeto no ambiente virtual.	Arquivo .stl
	MOVEFLOOR	Movimentar o piso do ambiente virtual.	
	AXISON	Exibe o sistema de coordenada	
	AXISOFF	Esconde o sistema de coordenada	
Parâmetros	*ADDTOOL*	***ADDTOOL** nome do arquivo **TRUE** (mostra) / **FALSE**(esconde)* Não precisa colocar a extensão do arquivo ".stl" (O sistema de coordenada do arquivo será inserido no final do sistema de coordenada da flange do robô.)	
	SHOW_TOOL	***SHOW_TOOL** nome do arquivo.* Não precisa colocar a extensão do arquivo ".stl"	
	ADDOBJ	***ADDOBJ** nome do arquivo **P: X, Y, Z, A, B, C, C: R,G,B*** ***P:** deslocamento (mm) e orientação (°); **C:** cores (valor RGB)*	
	SHOW_OBJ	***SHOW_OBJ** nome do arquivo **TRUE** (mostra) / **FALSE**(esconde)*	
	MOVEFLOOR	***MOVEFLOOR** (distância de movimento)*	
	AXISON		
	AXISOFF		
Exemplos	*ADDTOOL*	***ADDTOOL** garra*	
	SHOW_TOOL	***SHOW_TOOL** garra TRUE / SHOW_TOOL garra FALSE*	
	ADDOBJ	***ADDOBJ** mesa P:500,200 C: 200*	
	SHOW_OBJ	***SHOW_OBJ** mesa TRUE / SHOW_OBJ mesa FALSE*	
	MOVEFLOOR	***MOVEFLOOR** (150)*	
	AXISON	***AXISON***	
	AXISOFF	***AXISOFF***	

CONTROLE	Comandos para criar programas com lógicas estruturadas e parametrizadas	
Funções	**FOR...ENDFOR**	Execução de ciclos "laços"
	GOTO	Vai para um posição definida "label"
	IF...ENDIF	Estado de condição "se..."
	LOOP...ENDLOOP	"laços"
	REPEAT...UNTIL	Repetir ciclos... "até que"
	SWITCH...ENDSWITCH	Declaração de interrupção, desvio
	WHILE...ENDWHILE	Condição lógica "enquanto"

Exemplos

FOR...ENDFOR

```
INT NUM
FOR NUM=0 TO 2 STEP1
PTP P1  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
PTP P2  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
ENDFOR
```

GOTO

```
INÍCIO:
PTP P0  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
GOTO INÍCIO
```

IF...ENDIF

```
INT NUM=1
IF NUM>1 THEN
PTP P0  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
ENDIF
```

LOOP...ENDLOOP

```
LOOP
PTP P1  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
PTP P2  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
ENDLOOP
```

REPEAT...UNTIL

```
INT NUM
REPEAT
PTP P1  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
PTP P2  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
NUM = NUM+1
UNTIL NUM > 2
```

SWITCH...ENDSWITCH

```
INT NUM
LOOP
SWITCH NUM
CASE 0
PTP P1  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
CASE 1
PTP P2  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
CASE 2
EXIT
ENDSWITCH
NUM = NUM+1
ENDLOOP
```

WHILE...ENDWHILE

```
INT NUM=2
WHILE NUM > 0
PTP P1  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
PTP P2  CONT=100%  Vel=100% Acc=50% TOOL[0] BASE[0]
NUM = NUM - 1
ENDWHILE
```

REFERÊNCIAS BIBLIOGRÁFICAS

Hiwin Technologies Corp. **Multi-Axis Robot:** Industrie 4.0 Best Partner. 2020. Disponível em: https://www.hiwin.tw/download/tech_doc/mar/Multi_Axis_Robot_DM-(E).pdf. Acesso em: 01 Out. 2021.

HIWIN Technologies Cop. **Robot System Software - HRS 3.3:** (Original Instruction) User Manual. 5. ed. 2020. Disponível em: https://www.hiwin.tw/download/tech_doc/mar/Robot_Software_Manual_3.3_(E).pdf Acesso em: 04 Out. 2021.

HIWIN Technologies Cop. **Articulated Robot - RA605-GC:** (Original Instruction) User Manual. 2020. Disponível em: https://www.hiwin.tw/download/tech_doc/mar/RA605-GC_User_Manual-(E).pdf. Acesso em: 30 fev. 2021.