

FUTURO DO TRABALHO, TRABALHO DO FUTURO

ROBÓTICA GUIADA POR VISÃO COMPUTACIONAL INDUSTRIAL

Apostila do aluno

João Augusto Oliveira Luiz – Engenheiro II

Autor da apostila

Larissa Jessica Alves – Analista de Suporte Pedagógico III

Revisão da apostila

Fit Instituto de Tecnologia

Sorocaba, outubro de 2021

Autor



Engenheiro de controle e automação, com mais de 6 anos de experiência em automação industrial. Especialista em projetos de máquinas especiais com sistemas de controle dedicado, robótica industrial, redes de comunicação, visão computacional e integração com sistemas supervisórios de alto nível. Instrutor no programa Fit Tech Academy e professor dos cursos de robótica e supervisório na FACENS.

APRESENTAÇÃO

A presente apostila é um instrumento teórico que complementa o curso de capacitação de Robótica guiada por visão computacional, executado pelo Fit-Instituto de Tecnologia. Nela, veremos os fundamentos da robótica industrial, conceitos de movimentação do robô no espaço, como acionar entradas e saídas digitais e realizar a programação de um projeto de pega e depósito de peças (*Pick and Place*). Nesta apostila também será tratado o reconhecimento de imagens por câmera conectada ao robô, e a tomada de decisão conforme o objeto detectado.

A apostila está dividida em 9 seções, iniciando com os conceitos básicos de um robô industrial, onde será abordada as partes de um robô industrial e condições de segurança para manuseio. Em seguida, será abordado o sistema de coordenadas de um robô industrial e os pontos de operação desse, seguida de uma introdução à lógica de comandos na interface RC+. A sessão seguinte será composta dos comandos de entrada e saída digitais e memórias, seguidos dos conceitos de movimentação, velocidade, orientação do braço robótico e potências. Posteriormente, será tratado o reconhecimento de imagens por câmera conectada ao robô, para que esse realize correções de posicionamento de acordo com as coordenadas identificadas pelo sistema de visão. Por fim, trataremos da separação de peças e tomada de decisão por meio de visão computacional, e a integração do sistema com softwares externos.

Este material é baseado em artigos científicos, periódicos, revistas científicas e livros científicos. É extremamente recomendável ao aluno que, ao final da leitura de cada seção, realize os exercícios propostos e acesse os materiais indicados nas referências bibliográficas, para aprofundar a leitura desse material e complementar o que foi lido aqui.

Desejo a você, prezado aluno, que tenha um excelente curso!!

Boa Leitura !!

Indicação de ícones



Saiba mais: *oferece novas informações que enriquecem o assunto ou “curiosidades” e notícias recentes relacionadas ao tema estudado.*



Exemplos: *descreve exemplos sobre o assunto que está sendo abordado.*



Atenção: *indica pontos de maior relevância no texto.*



Avisos: *oferece avisos referente ao assunto..*

Sumário

1	Conceitos básicos de um robô industrial.....	6
1.1.	Definição	6
1.2.	Partes de um robô industrial	7
1.3.	Condições de segurança para manuseio de robôs	8
1.4.	Sistema de coordenadas de um robô Industrial	9
1.5	Relembrando o que é uma base tridimensional.	7
2	Introdução ao software RC+	9
2.1.	Criando um projeto.....	9
2.2.	Código fonte do projeto	10
2.3.	Janela de comandos	11
2.4.	Tela de configurações do robô.....	12
2.5.	Criando pontos para movimentação	13
2.6.	Movimentando manualmente para um ponto específico	15
2.7.	Compilando e salvando os pontos	16
2.8.	Exercício 1: criando um código de movimentação	16
2.9.	Janela de execução	18
2.10.	Janela de help.....	20
3	Conceitos de movimentação e potência do robô industrial	22
3.1.	Movimento Ponto-a-ponto (PTP).....	22
3.2.	Movimento de juntas	23
3.3.	Caminho contínuo (CP).....	23
3.4.	Orientação do braço robótico	24
3.5.	Potência do robô	26
3.6.	Velocidade e aceleração do robô.....	26

3.7.	Criando um programa de movimentação	27
3.7.1	Definição dos pontos	28
3.8	Exercício 2: Programação de movimentação usando Go e Move	30
4	Comandos de entrada e saídas digitais em robôs industriais	33
4.1.	Mapeando e nomeando as entradas e saídas	33
4.2.	Monitoramento das entradas e saídas	34
4.3.	Acessando as entradas e saídas no programa	6
5	Loops e funções no programa	8
5.1	Loops em SPEL+	8
5.2	Exercício 3: Utilização dos comandos On/Off e Sw	9
5.3	Funções em SPEL+	9
6	Orientação da ferramenta do robô e sistema de coordenadas relativo	12
7	Reconhecimento de imagem e correção de posicionamento.....	15
7.1	Fixação e posicionamento da câmera	17
7.2	Configuração no software EPSON RC+ 7.0.....	17
7.2.1	Criação de uma sequência de inspeção	18
7.2.2	Criação de uma calibração do robô	21
7.2.3	Criando a rotina de calibração	27
7.2.4	Executando a rotina de calibração	29
8	Separação de peças e tomada de decisão por imagem	33
8.1.	Comando Vrun	33
8.2.	Comando VGet	33
8.3.	Aplicação do programa criado.....	34
8.4	Exercício 4 – Programação usando sistema de visão	35

9	Integração com softwares terceiros	36
	Conclusão	38
	Referências	39

2.	CONTROLE DE REVISÃO DO DOCUMENTO / <i>DOCUMENT REVISION</i>	
CONTROL	39	

1 Conceitos básicos de um robô industrial

1.1. Definição

Um robô industrial é uma máquina manipuladora, com vários graus de liberdade, controlada automaticamente, reprogramável, multifuncional, que pode ter a base fixa ou móvel, para utilização em aplicações de automação industrial.

Por grau de liberdade, temos como definição o número de movimentos individuais das articulações. Esse parâmetro também identifica a versatilidade do robô.

Um conceito importante para tratarmos também é o de elos e juntas. Um manipulador é uma combinação de elementos rígidos (corpos e elos), conectados entre si através de articulações (Juntas).

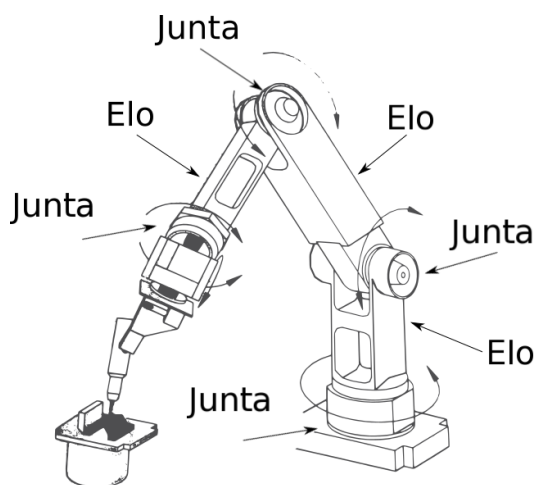


Figura 1 - Juntas e elos de um robô industrial Fonte: imagem adaptada de PAZOS, Fernando – Automação de Sistemas & Robótica.

Tratando-se de Juntas, podemos separá-las em 2 tipos principais: **Prismática e Rotação (revolução).**

- **Juntas Prismáticas (translação):** são responsáveis pelos movimentos de translação relativa entre dois elos.
- **Juntas de Rotação (revolução):** são responsáveis pelo movimento de rotação relativa entre dois elos.

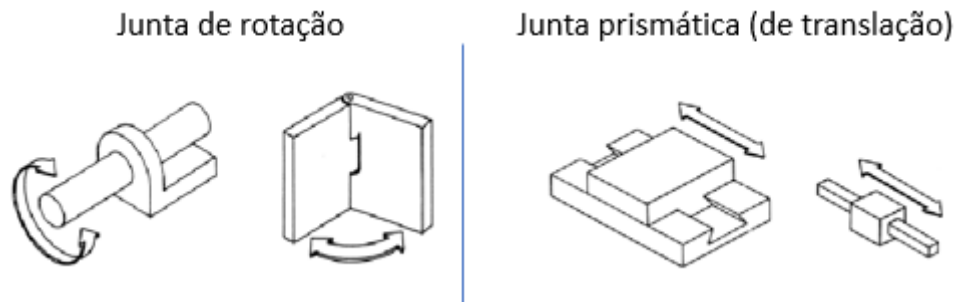


Figura 2 - Funcionamento das juntas de rotação e prismática, Fonte Adaptado da Aula de Robótica_2017 do prof. Mário Luiz Tronco.

Em geral, prefere-se as juntas de Revolução às Prismáticas, pelo motivo das primeiras serem mais compactas e confiáveis.

1.2. *Partes de um robô industrial*

Um robô industrial é composto em sua maioria por 3 componentes principais, **Controlador, Estrutura Mecânica ou Manipulador e Efetuador**:

- **Controlador:** responsável por armazenar o código fonte do robô, controlar os drivers e realizar as conexões com o mundo externo.
- **Estrutura Mecânica ou Manipulador:** parte mecânica, composta pelos motores, engrenagens e reduções. É a parte que move durante o processo de operação do robô.
- **Efetuator:** comumente chamado de “garra”, é a parte que realiza a interação com o objeto ou produto manipulado pelo robô. Pode ser

composto de dispositivos para movimentação, como garras e ventosas, ou ferramentas, como parafusadeiras, ferramentas de pintura e solda.

1.3. Condições de segurança para manuseio de robôs

Para operar um robô industrial, é necessário garantir condições de segurança. Para isso, temos que garantir alguns parâmetros, que são:

- **O circuito Safeguard deverá estar fechado:** cortina de luz, scanner de área, entre outros, que devem estar livres de invasão.
- **O circuito de emergência deverá estar fechado:** botão de emergência, porta de segurança, entre outros, que devem estar livres.

Através do visor do controlador, é possível observar o *status* do circuito de segurança:




	From power-on to boot	While running	
LED	All blink	LED for current operation mode (TEST, TEACH, AUTO, PROGRAM) turns ON.	
7 segment	All lights out		READY (Normal)
			Emergency Stop
			Safeguard
		Four digits	Error

Figura 3 - Tabela indicativa dos estados de operação do robô. Fonte: Manual de operação EPSON RC+ 7.0.

Quando houver uma variação no sinal de segurança do controlador, esse entrará em modo de parada de emergência. Todos os motores param

imediatamente, e o código fonte entra em um estado de emergência, parando sua execução.

Um erro  pode ser observado no visor do controlador.




	From power-on to boot	While running	
LED	All blink	LED for current operation mode (TEST, TEACH, AUTO, PROGRAM) turns ON.	
7 segment	All lights out		READY (Normal)
		 (highlighted with a red border)	Emergency Stop
			Safeguard
		Four digits	Error

Figura 4 - Destaque do indicador de parada de emergência no display do controlador.
Fonte: Manual de operação EPSON RC+ 7.0

1.4. Sistema de coordenadas de um robô Industrial

Um robô industrial opera por meio de coordenadas pré-definidas, que podem ser gravadas em sua memória durante o processo de programação. Essa programação pode ser feita por meio de *software* proprietário, fornecido pelo fabricante junto com a compra. Para exemplo desse curso, vamos analisar o robô articulado de 6 eixos (Figuras 5 e 6).

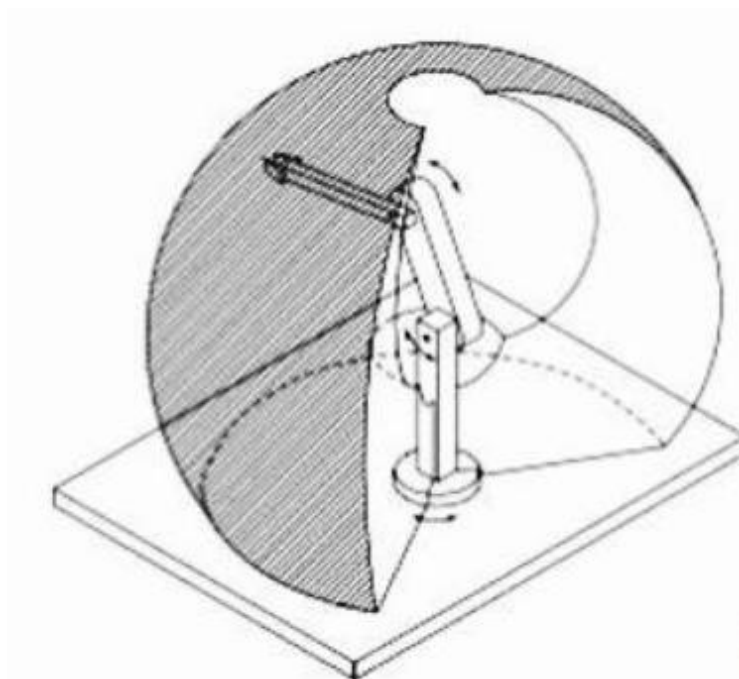


Figura 5 - Envelope de trabalho de um robô de 6 eixos. Fonte: adaptado de *Manipulators-and-their-work-envelopes-a-Cartesian-manipulator*.

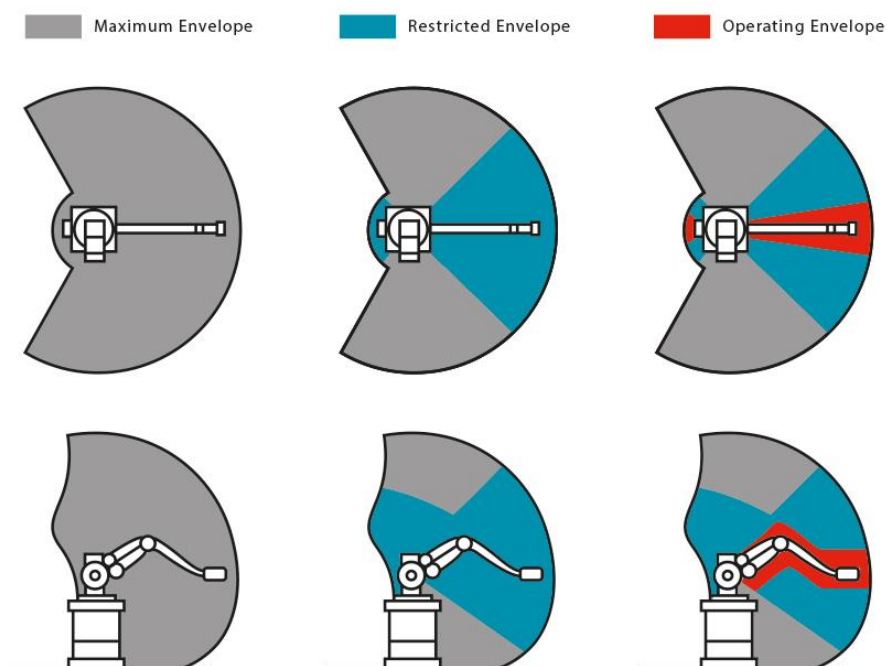


Figura 6 - Envelope de operação de um robô de 6 eixos. Fonte: Distrelec.

Para alcançar todos os pontos do espaço demonstrado nas figuras acima, temos que observar as características construtiva do robô.

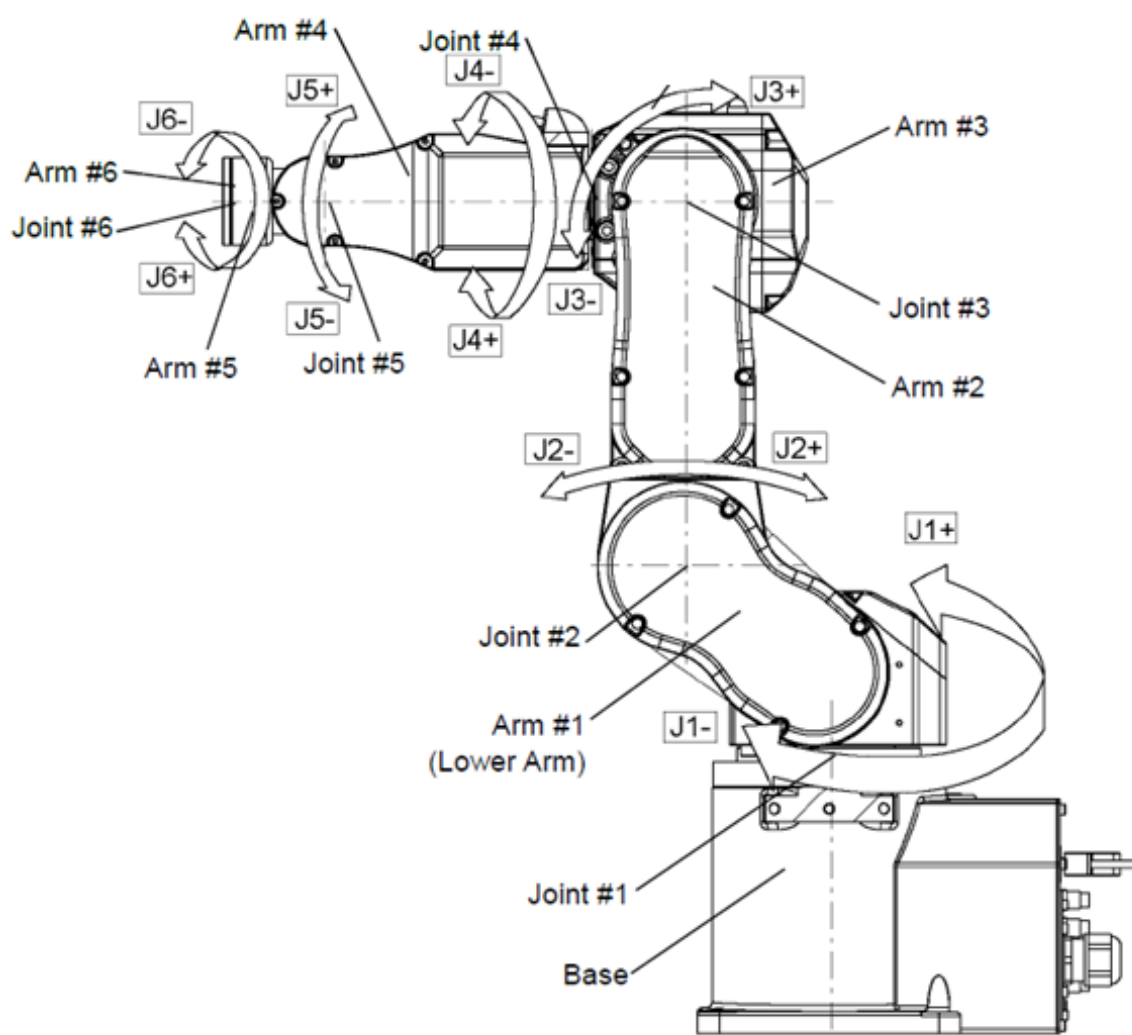


Figura 7 - Juntas, eixos e braços de um robô de 6 eixos. Fonte: Manual de operação EPSON RC+ 7.0.

Legenda:

Junta #1: Todo o robô rotaciona.

Junta #2: O braço inferior balança.

Junta #3: O braço superior balança.

Junta #4: O punho rotaciona

Junta #5: O punho balança.

Junta #6: A mão rotaciona.

Um robô industrial articulado usualmente é montado de duas formas principais, em **mesas e bases**, ou em **teto**. Em cada uma dessas configurações, temos que nos atentar ao sistema de coordenadas que se modifica.

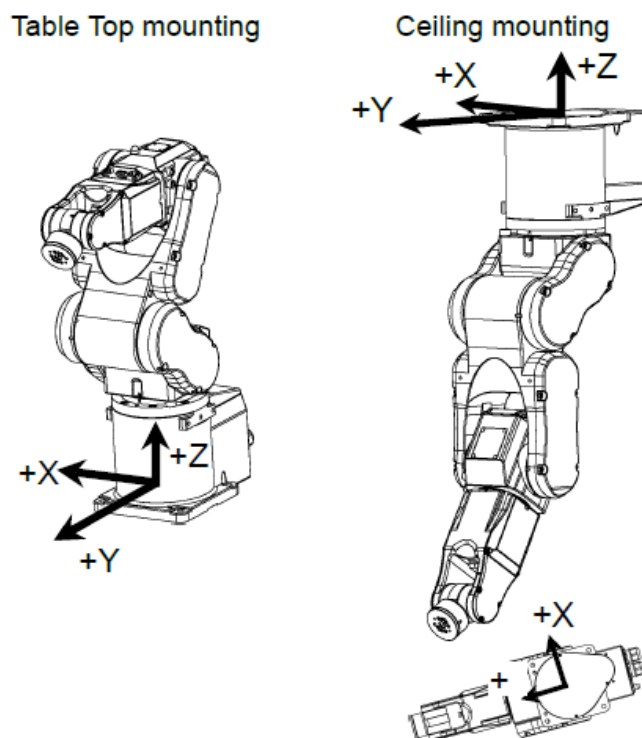


Figura 8 - Tipos de montagem de um robô de 6 eixos. Fonte: Manual de operação EPSON RC+ 7.0.

Para o robô de 6 eixos, um ponto é definido como a posição e orientação do sistema de coordenadas da ferramenta, em relação ao sistema de coordenadas retangular.

Um ponto é definido pela posição (X, Y, Z) e a orientação especificada pela orientação (U, V, W) sendo esses:

- U: Rotação no eixo X
- V: Rotação no eixo Y
- W: Rotação no eixo Z

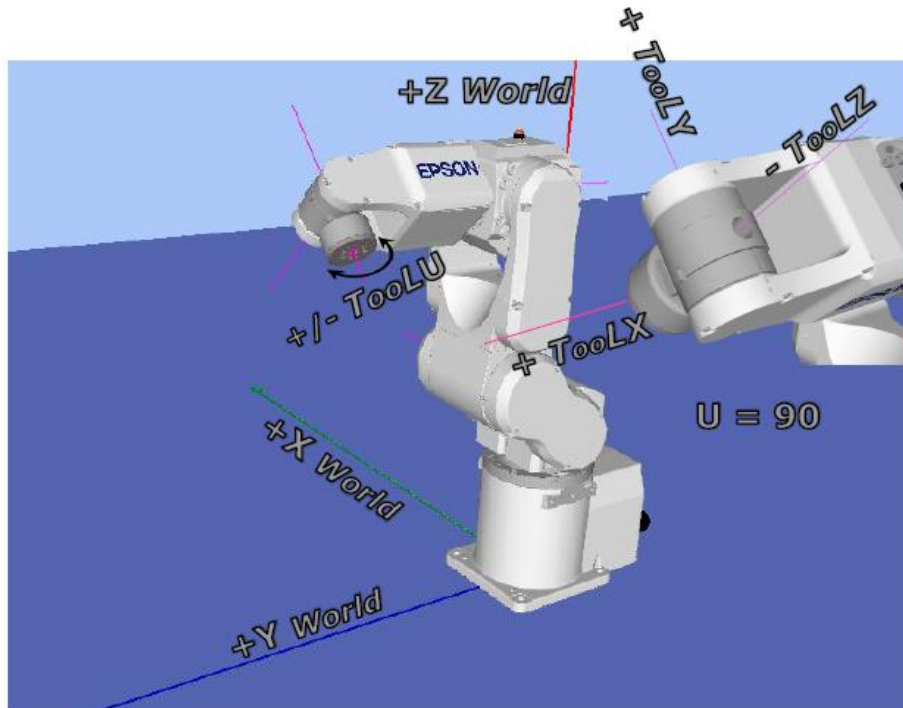


Figura 9 - Sistema de coordenadas de um robô. Fonte: Adaptado de manual de operação EPSON RC+ 7.0.

Um ponto no robô é definido pelas coordenadas X, Y, Z e U, sendo as 3 primeiras definidas em milímetros, a partir da origem localizada na base do robô, e a coordenada U definida pela posição do punho em graus.

1.5 Relembrando o que é uma base tridimensional.

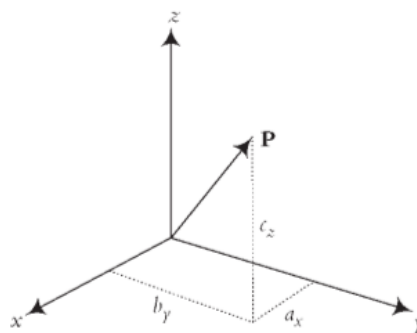


Figura 10 - Sistema cartesiano. Fonte: Adaptado de Aula de Robótica, Prof. Denis. Borg.

$$\mathbf{P} = a_x \mathbf{i} + b_y \mathbf{j} + c_z \mathbf{k}$$



Sempre verifique a localização dos novos pontos programados antes de retornar ao funcionamento normal do sistema.



Sempre esteja preparado para falhas quando realizar alterações de pontos. Se o manipulador se comportar de forma anormal, acione imediatamente o botão de emergência.

2 Introdução ao software RC+

Para este curso, usaremos como base o robô EPSON C4L, compatível com o simulador C4 Sample do *software* EPSON RC+ 7.0. Com o intuito de realizar programações no *software*, precisamos primeiro entender sobre a arquitetura e o funcionamento da aplicação, que abordaremos nas próximas subseções.

Para download do *software*, basta acessar a página da internet do fabricante, disponível em:

<https://epson.com.br/Suporte/Rob%C3%B4s/Software/sh/s650>

Selecione a opção EPSON RC+ 7.0

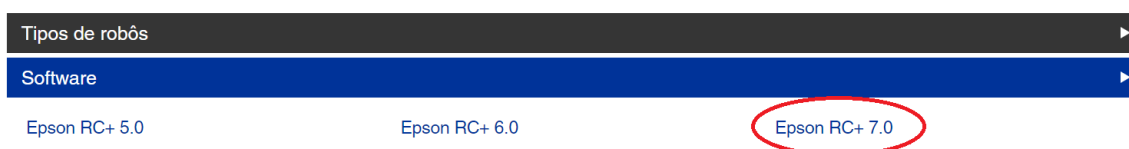


Figura 11 - Página de download da EPSON. Fonte: Autoria própria.



As Figuras a seguir da seção 2 são todas de autoria própria utilizando o *software* EPSON RC+7.0.

2.1. Criando um projeto

O primeiro passo para a criação de um projeto de automação utilizando o robô, é a criação do projeto no *software* EPSON RC+. Para isso, é necessário seguir o seguinte passo:

- Abrir o *software* Epson RC+, começar um projeto novo no menu “**Project**”-> **New** e nomear o programa como desejado.

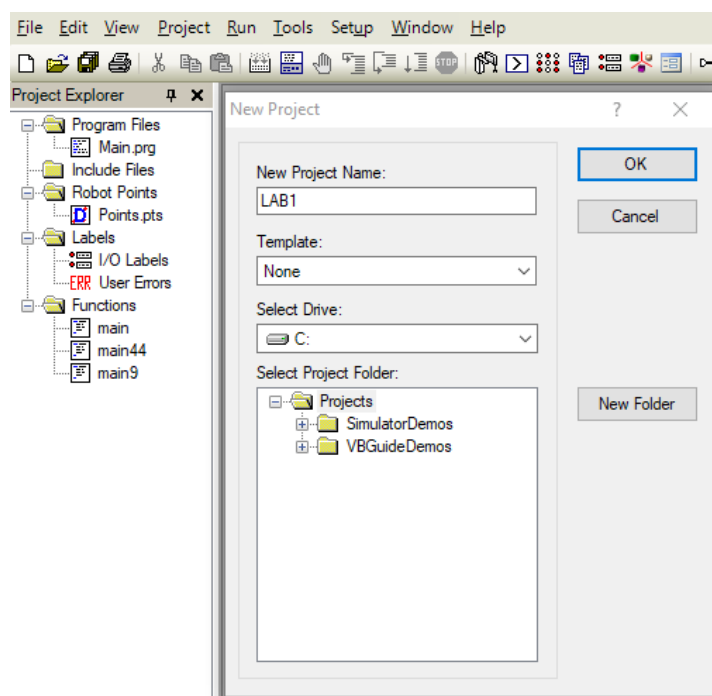


Figura 12- Tela de criação de projeto. Fonte: autoria própria.

2.2. Código fonte do projeto

O Código fonte do projeto deve ser criado na tela **Main.prg**, por meio de linguagem textual SPEL, nome dado à linguagem usada nos programas EPSON, muito parecida com a linguagem de programação VBA (Visual Basic for Applications).

A tela **Main** está localizada na árvore do projeto, ao lado esquerdo da interface gráfica.

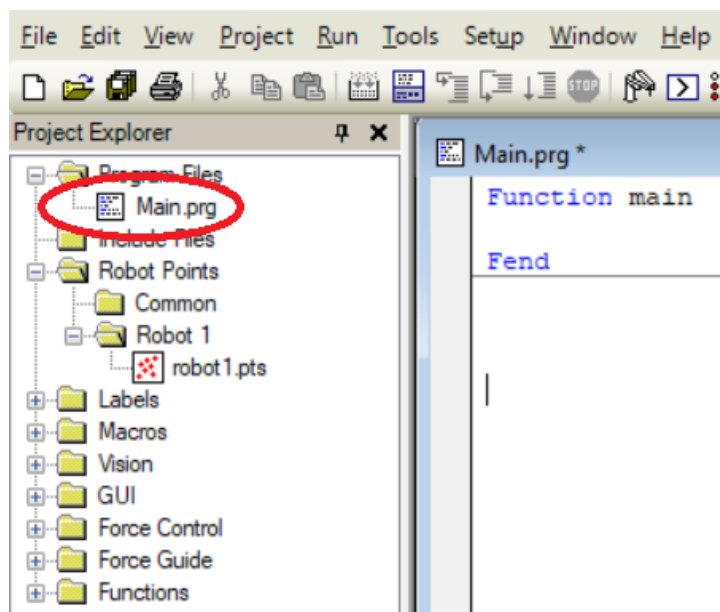


Figura 13 - Tela Main na interface do programa EPSON. Fonte: Autoria própria.

2.3. Janela de comandos

A janela de comandos simula em ASCII um terminal que se comunica diretamente com o controlador do robô. Nessa janela, é possível executar comandos SPEL diretamente da linha de comando e ver o resultado.

Para acessar essa tela, é necessário selecionar a opção **Tools -> Command Window...**, ou navegar pela barra de tarefas até o ícone.

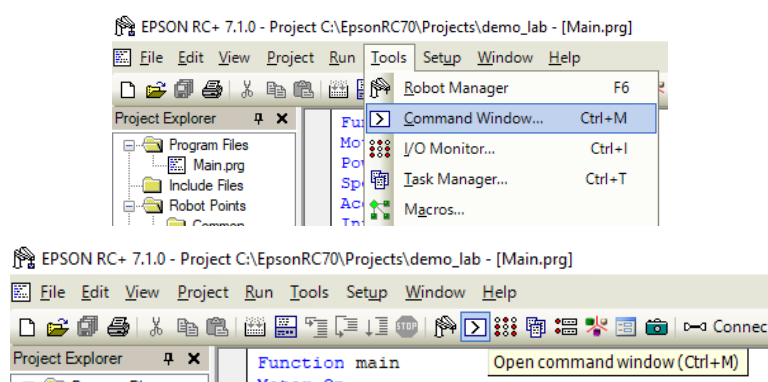


Figura 14- Tela de criação da janela de comando. Fonte: autoria própria.

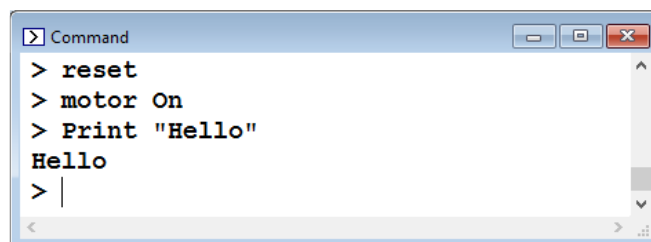


Figura 15- Tela da janela de comando. Fonte: autoria própria.

2.4. Tela de configurações do robô

Por meio da ferramenta **Robot Manager (F6)**, localizado no menu superior, podemos configurar os estados de operação do robô.

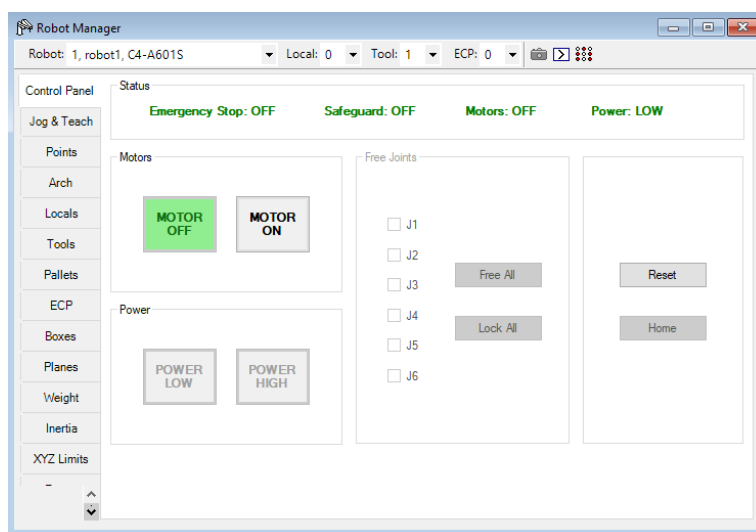


Figura 16- Robot Manager. Fonte: autoria própria.

Essa ferramenta é utilizada vastamente para a configuração do robô, controle manual e criação/edição de padrões dentro do projeto. Para realizar a

movimentação manual do robô e salvar os pontos no arquivo de configurações, podemos utilizar a tela de **Jog & Teach**.

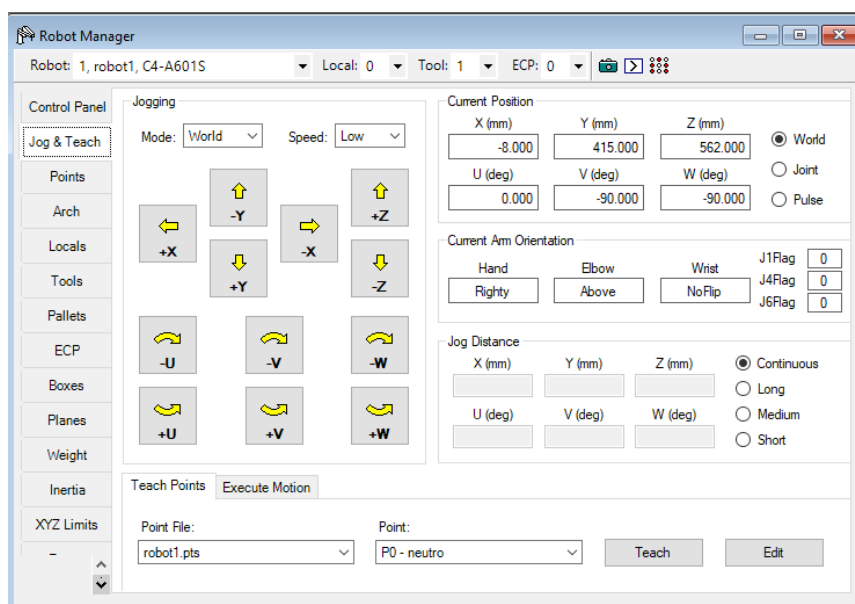


Figura 17- Movimentação manual no Robot Manager. Fonte: autoria própria.

Nesta interface, é possível comandar os motores do robô, energizando-os, e realizar movimentações para configurar a posição por meio desta tela. Uma vez energizado os motores, é possível utilizar esta tela para movimentar nos 6 eixos representados no braço robótico.

2.5. Criando pontos para movimentação

Para salvar um ponto desejado ao seu programa, utilize os botões de coordenadas no Robot Manager, e selecione uma das opções de ponto ainda não utilizadas na lista localizada na parte inferior da tela, e selecionar a opção **Teach**.

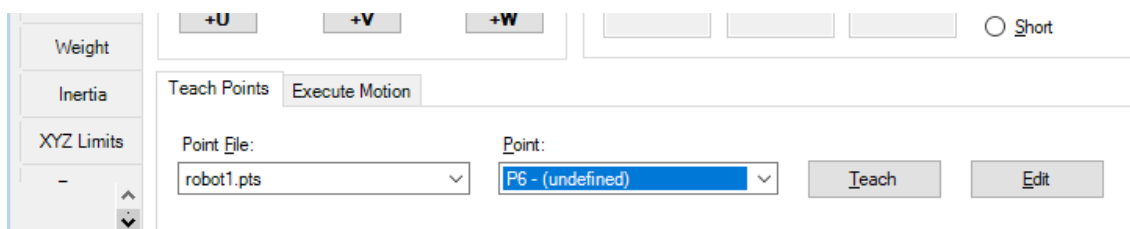


Figura 16- Área de gravação de pontos no robô. Fonte: autoria própria.

Ao selecionar o botão **Teach**, uma janela aparecerá onde é possível nomear o ponto para melhor utilização e referência ao longo do projeto.

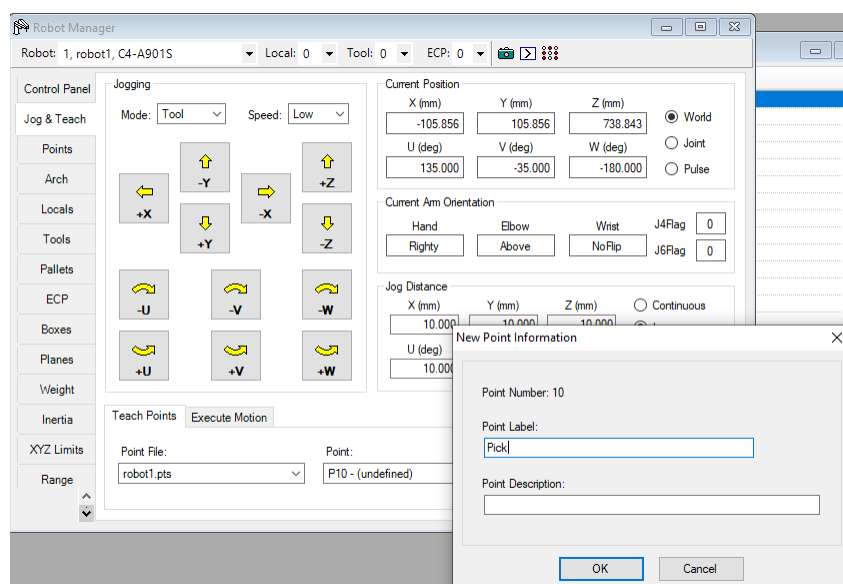
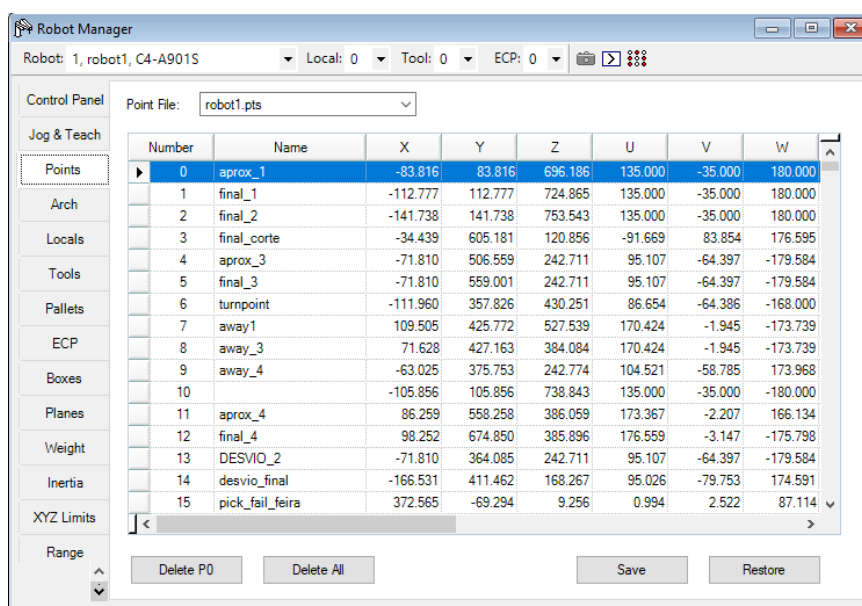


Figura 18- Nomeação do ponto criado. Fonte: autoria própria.

Todos os pontos salvos durante o projeto, são salvos em um arquivo texto nomeado **robot1.pts**. Esse arquivo pode ser acionado na tela do Robot Manager, ou navegando pelas pastas do seu sistema operacional.



Number	Name	X	Y	Z	U	V	W
0	aprox_1	-83.816	83.816	696.186	135.000	-35.000	180.000
1	final_1	-112.777	112.777	724.865	135.000	-35.000	180.000
2	final_2	-141.738	141.738	753.543	135.000	-35.000	180.000
3	final_corte	-34.439	605.181	120.856	-91.669	83.854	176.595
4	aprox_3	-71.810	506.559	242.711	95.107	-64.397	-179.584
5	final_3	-71.810	559.001	242.711	95.107	-64.397	-179.584
6	turnpoint	-111.960	357.826	430.251	86.654	-64.386	-168.000
7	away_1	109.505	425.772	527.539	170.424	-1.945	-173.739
8	away_3	71.628	427.163	384.084	170.424	-1.945	-173.739
9	away_4	-63.025	375.753	242.774	104.521	-58.785	173.968
10		-105.856	105.856	738.843	135.000	-35.000	-180.000
11	aprox_4	86.259	558.258	386.059	173.367	-2.207	166.134
12	final_4	98.252	674.850	385.896	176.559	-3.147	-175.798
13	DESVIO_2	-71.810	364.085	242.711	95.107	-64.397	-179.584
14	desvio_final	-166.531	411.462	168.267	95.026	-79.753	174.591
15	pick_fail_feira	372.565	-69.294	9.256	0.994	2.522	87.114

Figura 19- Lista de pontos cadastrados no projeto. Fonte: autoria própria.

2.6. Movimentando manualmente para um ponto específico

Na aba de **Jog & Teach**, é possível movimentar o robô de forma semi automática até um ponto cadastrado. Para isto, basta alternar para a aba “**Execute Motion**” e selecionar o tipo de movimento e o ponto desejados.

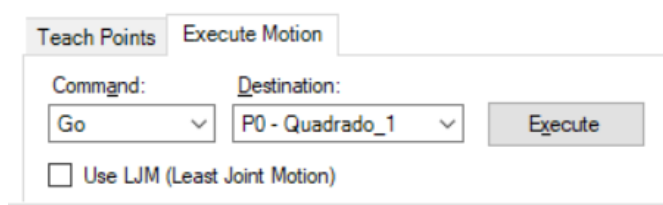


Figura 20- Movimentação manual no Robot Manager. Fonte: autoria própria.

2.7. *Compilando e salvando os pontos*

Uma vez verificado que a movimentação está de acordo com o esperado, devemos compilar as alterações pressionando a opção **Build Project (Ctrl+B)**.

Assim pode-se iniciar a programação do código fonte.

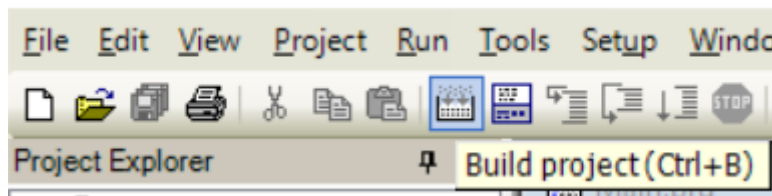


Figura 21- Botão Build Project. Fonte: autoria própria.

2.8. *Exercício 1: criando um código de movimentação*

Abaixo deve-se seguir os passos para a criação de um código de movimentação.

- **Na janela de comando:**
Digite -> Pulse 0,0,0,0,0,0
- **Na janela do Robot Manager:**

Selecione Jog Mode – Joint

J1 -25°

J3 +15°

J2 -66°

J5 -35°

Salve como P0

Selecione Jog Mode – World

Selecione Jog Distance para Long

X -200mm

Salve como P1

Selecione Jog Mode – World

Selecione Jog Distance para Long


X -200mm

Salve como P2

- Na tela Main.prg digite o seguinte código:

```
Function main
Motor On
Go P0
Go P1
Go P2
Fend
```

Figura 22- Trecho de código do exercício 1. Fonte: autoria própria.

Execute o programa pressionando **F5** ou clicando no botão  então clique no **botão Start**.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto do simulador após a execução de cada uma das linhas do código

Código fonte da função MAIN

2.9. Janela de execução

Na barra de tarefas, selecione a opção **Run -> Run Window (F5)**, ou localize e selecione o botão de **“Open run window”** na barra de tarefas.

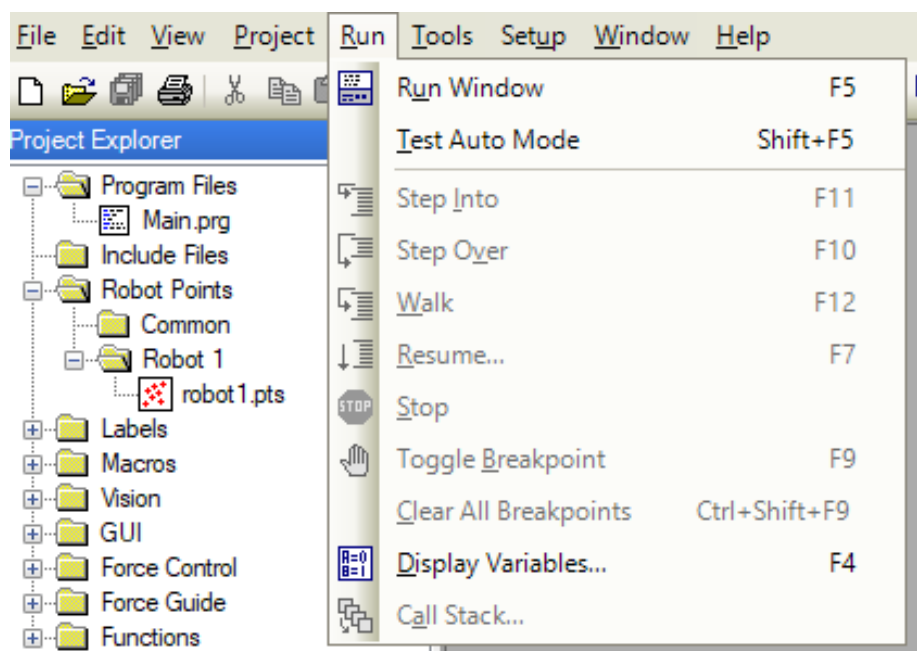


Figura 23-Localização do botão Run window. Fonte: autoria própria.

A tela de execução será apresentada na interface:

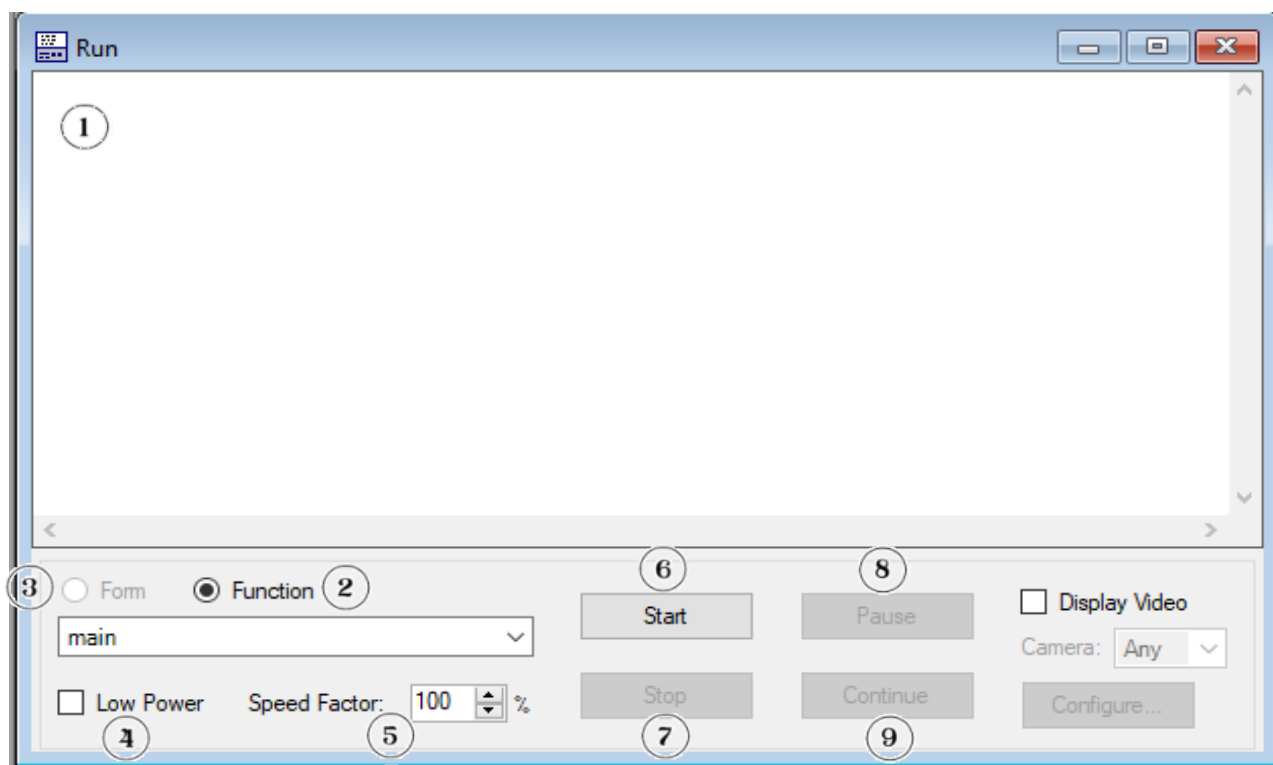


Figura 24-Janela Run Window. Fonte: autoria própria.

Observe os diferentes campos da Janela Run Window (Figura 24):

- | | |
|--|--|
| <p>1 Área de texto: mostra as saídas do programa, textos e <i>status</i> de erro.</p> | <p>4 Low Power: quando essa caixa está selecionada, o controlador ignora o comando Power HIGH, fazendo com que o código execute em Low Power.</p> |
| <p>2 Function: selecione a função para iniciar.</p> | |
| <p>7 Form: selecione um form para iniciar. Utilizado se você criou uma GUI customizada.</p> | <p>5 Speed Factor: especifica a velocidade de movimentação do motor relativa ao valor máximo do programa. Se o</p> |

programa executa a 90 e o speed factor for 50%, o robô se moverá a 40.

⑥ **Start:** inicia a função selecionada.

⑦ **Stop:** para todas as tarefas. Se o robô estiver executando um movimento,

esse irá desacelerar até parar. **Ctrl+C:** Mesma função que o botão Stop.

⑧ **Pause:** pausa a tarefa. Se o robô estiver executando um movimento, esse irá desacelerar até parar.

⑨ **Continue:** continua tarefas pausadas.

2.10. Janela de help

Para acessar a janela de ajuda do *software*, no menu de tarefas, clique em **Help -> How Do I (Ctrl+F1)**.

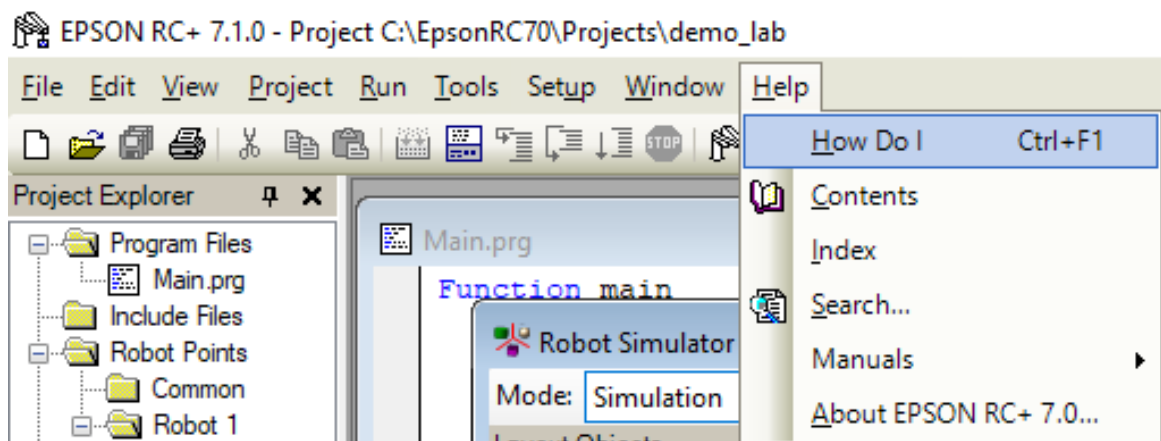


Figura 25-Área de acesso do Help. Fonte: autoria própria.

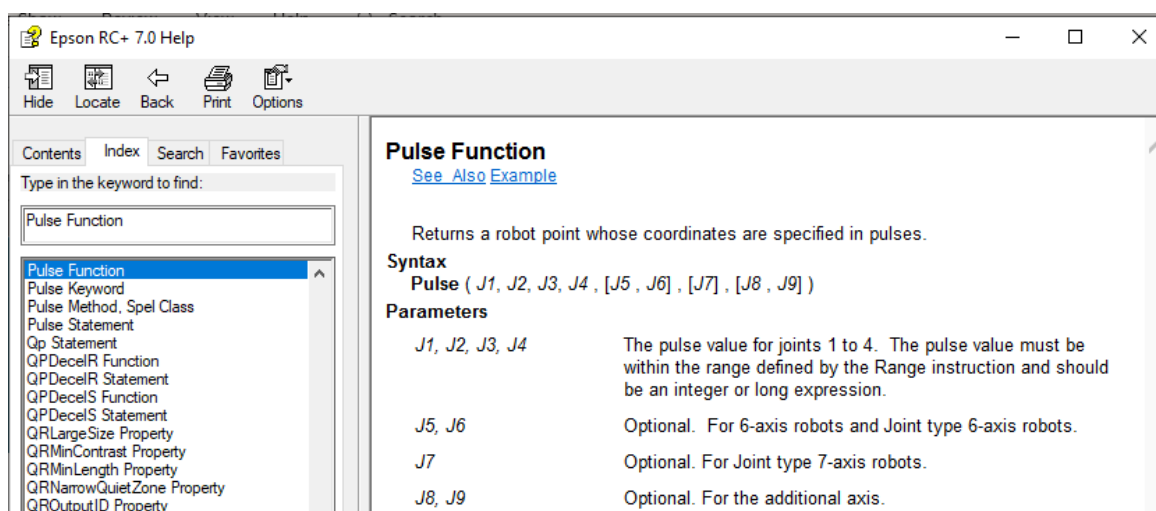


Figura 26 -Janela de Help do EPSON. Fonte: autoria própria.

3 Conceitos de movimentação e potência do robô industrial

Para controlarmos um robô industrial, precisamos primeiramente entender os tipos de movimentos que esse dispositivo está sujeito. Para isso, podemos realizar algumas perguntas relacionadas a operação do nosso manipulador. Algumas delas são:

- **Qual operação meu robô irá fazer?**
 - Pick/Place, paletização, soldagem, aplicação de cola?
- **Estou preocupado(a) com o caminho que ele fará até o destino?**
 - Deve fazer um movimento totalmente retilíneo? Podem haver variações?
- **Existem obstáculos no caminho do meu robô?**
 - Peças, suportes, grades, colunas?

A seguir apresentaremos os tipos de movimentos que podem ser aplicados ao robô.

3.1. *Movimento Ponto-a-ponto (PTP)*

O movimento ponto a ponto move o robô de uma posição inicial diretamente para o ponto especificado. Esse movimento pode não ser uma linha reta. Na linguagem SPEL, podemos usar os seguintes comandos para realizar essa operação na programação:

Go

Executa um movimento simultâneo de 4 eixos com desaceleração e para no ponto especificado

Jump ou Jump3

Faz um movimento de “portal” até a posição. Primeiro movimenta em Z para cima, e em seguida viaja horizontalmente, finalizando com movimento em Z para baixo.

Pass

Move o robô para uma posição próxima, mas não exatamente ao ponto especificado

3.2. Movimento de juntas

O movimento de juntas move individualmente cada uma das juntas do robô. No caso do robô de 6 eixos, é possível ter o controle individual de cada um dos 6 motores. Para isso, podemos usar os seguintes comandos durante a programação:

- **Pulse**

Executa o movimento simultâneo dos 6 eixos usando o valor especificado de pulsos.

- **JTRAN**

O comando JTRAN pode ser usado para movimentar uma junta do robô para uma posição especificada em graus ou milímetros, dependendo do tipo de junta.

- **PTRAN**

Use a junta para mover uma junta específica para um número específico de pulsos da posição corrente

3.3. Caminho contínuo (CP)

O Caminho contínuo (CP) quando possuímos restrições de movimento, quanto ao caminho que o robô percorrerá entre o ponto inicial e final, usamos

comandos do tipo CP. Esse comando garante que durante todo o processo de movimentação entre o ponto de origem e destino, o robô possui total controle e precisão da sua posição corrente. Para essa aplicação, podemos usar os comandos abaixo:

- **CP**

Comando de movimento que une a desaceleração e aceleração do robô formando um movimento constante entre os pontos

- **MOVE**

Executa um movimento linear, onde todos os eixos movem interpolando garantindo uma linha reta.

- **Arc**

Tipo de movimento curvo, onde um arco é definido por 3 pontos e o braço se move em um movimento circular entre esses pontos, parando no terceiro.

- **Curve**

Cria dados para moverem o manipulador através de uma curva definida por uma série de pontos em um arquivo.

3.4. Orientação do braço robótico

Um manipulador de 6 eixos pode acessar um ponto no espaço de 6 formas diferentes, dado sua grande flexibilidade e graus de liberdade. A imagem abaixo (Figura 25) nos mostra um exemplo de acesso de diferentes formas.

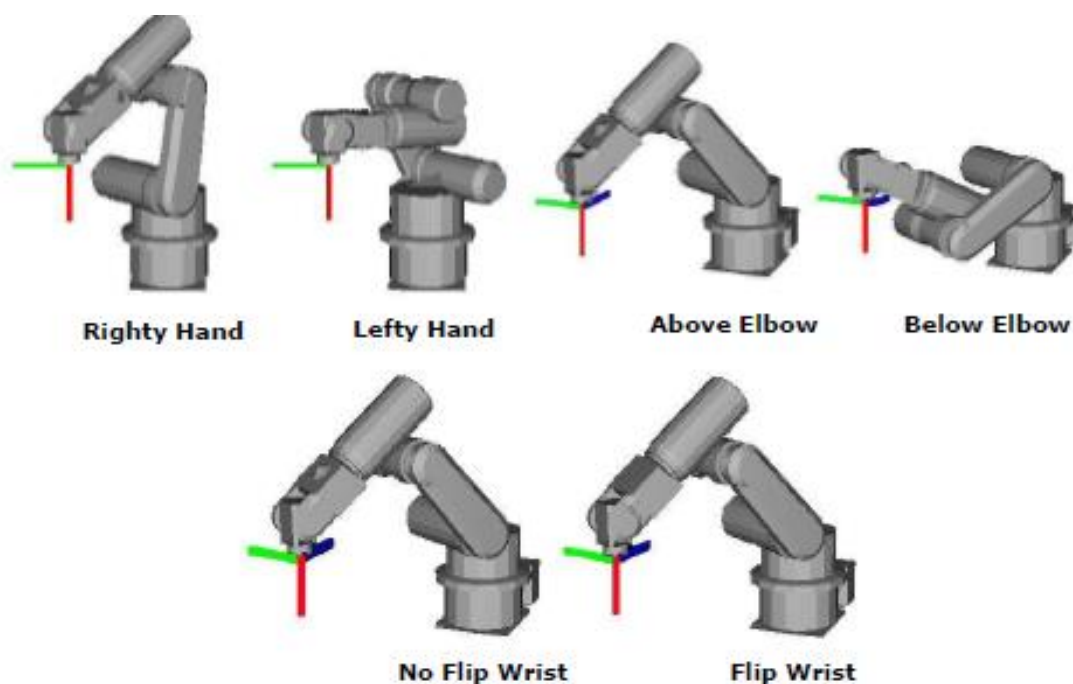


Figura 27 -Posições possíveis para acesso ao ponto de um robô 6 eixos. Fonte: Manual de operação EPSON RC+ 7.0.

Na aba **Points** do **Robot Manager**, podemos ver que quando criamos um ponto, os valores de Hand, Elbow e Wrist são salvos sem que tenhamos que interferir.

Number	Label	Hand	Elbow	Wrist	J1Flag	J4Flag	J6Flag
0	neutro	Righty	Above	Flip	0	0	0
1	ponto_prova_1	Righty	Above	Flip	0	0	0
2	ponto_prova_2	Righty	Above	Flip	0	0	0
3	ponto_prova_3	Righty	Above	Flip	0	0	0
4	ponto_prova_4	Righty	Above	Flip	0	0	0
5							

Figura 28 -Aba Points, exibindo os parâmetros de orientação do braço robótico. Fonte: autoria própria.

3.5. *Potência do robô*

Quando trabalhamos com um robô industrial, podemos selecionar dois modos de operação quanto à potência dos motores.

- **Power Low** (padrão)

Potência do motor é limitada a 30%, aceleração e desaceleração limitadas a 10%, velocidade limitada a 5%.

- **Power High**

Potência máxima em todos os parâmetros.

3.6. *Velocidade e aceleração do robô*

A velocidade e aceleração são parâmetros muito importantes durante o projeto, pois podem determinar o tempo de ciclo de uma operação, assim como ser crucial em processos onde é necessário um manuseio mais cauteloso do objeto.

O robô EPSON possui os seguintes parâmetros de velocidade:

- **Speed**

Especifica ou mostra a velocidade do braço para uso nos comandos PTP. Valor entre 1-100%.

Ex.: **Speed** 20.

- **ACCEL**

Especifica ou mostra a aceleração e desaceleração para o uso de instruções PTP. Valores entre 1-100%

Ex.: **ACCEL** 50,50.

- **SpeedS**

Especifica ou mostra a velocidade do braço para comandos CP como Move, Arc. Valores especificados de 1-2000 mm/seg.

Ex.: **SpeedS** 1200 [mm/seg].

- **AccelS**

Especifica ou mostra a aceleração e desaceleração para o uso de instruções CP. Valores entre 1-25000 mm/seg²

Ex.: **AccelS** 20000 [mm/seg²].

3.7. Criando um programa de movimentação

Para ajudar no entendimento da aplicação, pode-se abrir o simulador, e selecionar a opção “Render TCP Path” nas configurações. Isso fará com que uma linha amarela se forme na posição da ferramenta do nosso efetuador.

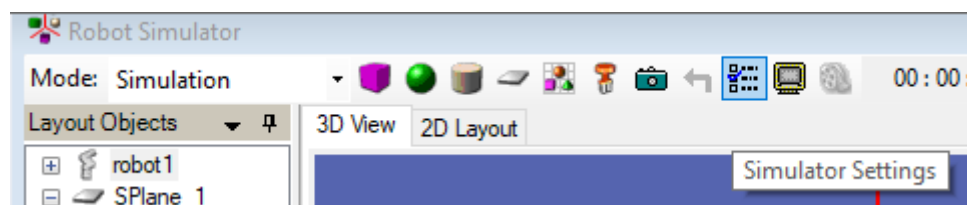


Figura 29 – Botão de configurações do simulador. Fonte: autoria própria.

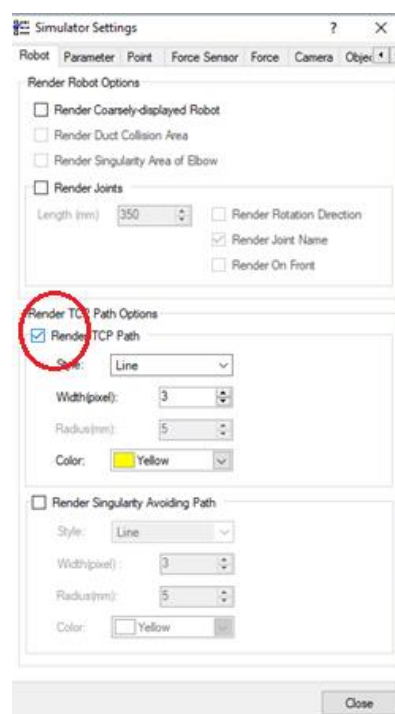


Figura 30 -Opção Render TCP Path no simulador. Fonte: autoria própria.

Durante as simulações, pode ser necessário apagar as linhas para deixar mais limpa a interface e, perceber a diferença entre os movimentos. Para isso usa-se o botão **“Clear TCP Path”**.



Figura 31 – Botão Clear TCP Path. Fonte: autoria própria.

3.7.1 Definição dos pontos

Para a validação desse conceito, pode-se usar 5 pontos, conforme o mostrado na figura abaixo.

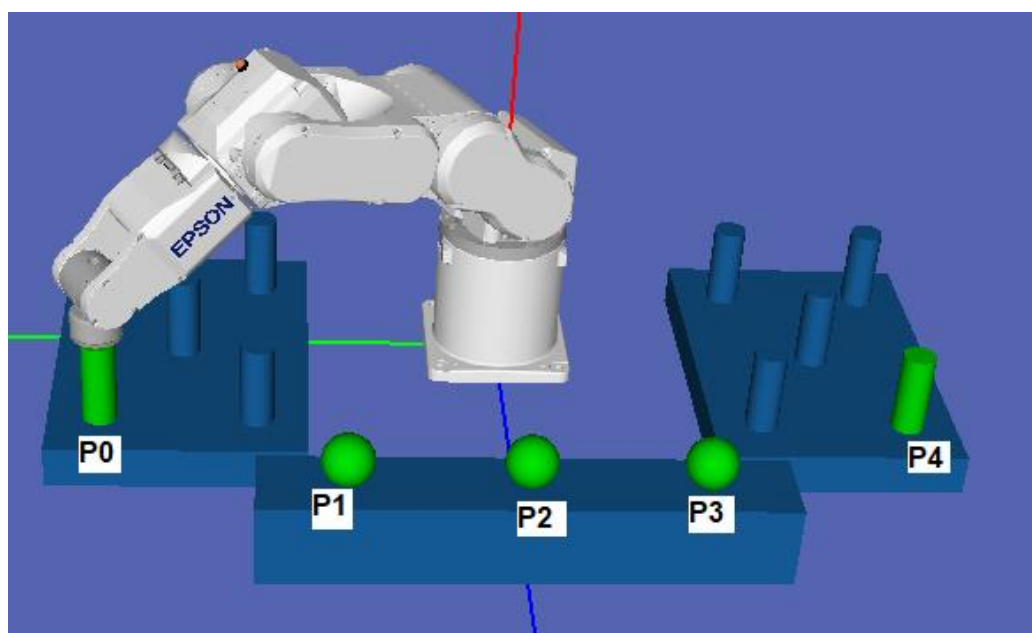


Figura 32 -Pontos que serão utilizados no laboratório. Fonte: autoria própria.

Movimente o robô até uma posição próxima ao topo dos objetos, sem encostar, e salve cada uma delas. Lembre-se de dar um nome adequado para cada um dos pontos.

Robot Manager															
Robot: 1, robot1, C4-A601S Local: 0 Tool: 1 ECP: 0															
Control Panel	Point File: robot1.pts														
Jog & Teach															
	Number	Label	X	Y	Z	U	V	W	Local	Hand	Elbow	Wrist	J1Flag	J4Flag	J6Flag
Points	0	Pino_esq	474.936	275.777	163.112	90.000	0.000	-180.000	0	Righty	Above	NoFlip	0	0	0
Arch	1	bola_esq	192.874	441.201	163.111	90.000	0.000	-180.000	0	Righty	Above	NoFlip	0	0	0
	2	bola_meio	-8.379	441.201	163.112	90.000	0.000	-180.000	0	Righty	Above	NoFlip	0	0	0
Locals	3	bola_dir	-206.804	441.201	163.112	90.000	0.000	-180.000	0	Righty	Above	NoFlip	0	0	0
Tools	4	pino_dir	-479.304	268.114	163.112	90.000	0.000	-180.000	0	Righty	Above	NoFlip	0	0	0
	5														
Pallets	6														
	7														
ECP	8														
	9														
Boxes															

Figura 33 -Tabela points usada no modelo do exercício. Fonte: autoria própria.

3.8 Exercício 2: Programação de movimentação usando Go e Move

Para esse exercício, pode-se usar o template de código abaixo:

```
Function main
```

```
Motor On 'vamos ligar os motores
```

```
Power High 'vamos mudar a potência e velocidade dos motores
```

```
Speed 40 'Vamos mudar a velocidade e aceleração para os comandos PTP
```

```
Accel 40, 40
```

```
SpeedS 1500 'Vamos mudar a velocidade e aceleração para os comandos CP
```

```
AccelS 1500, 1500
```

```
Do
```

```
'seu código aqui
```

```
Loop
```

```
Fend
```

Para o exercício, programe os movimentos para realizar cada uma das figuras, e execute o seu programa usando o “Run Window (F5)” e pressionando Start.

Limpe os traços do TCP da tela após cada execução!

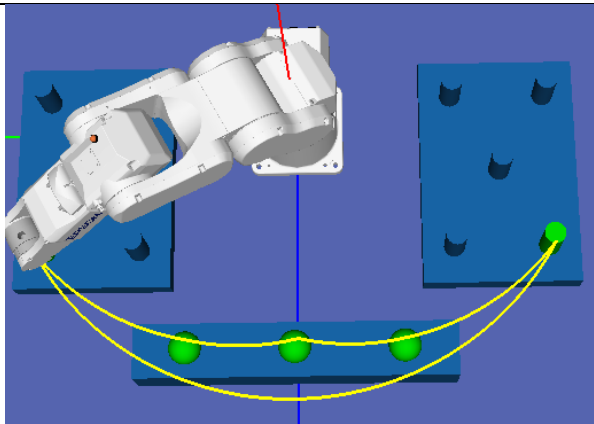
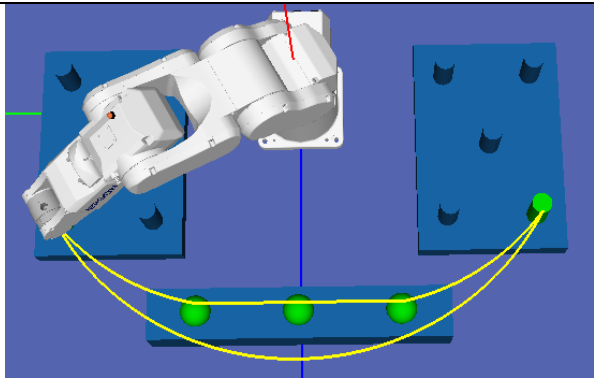
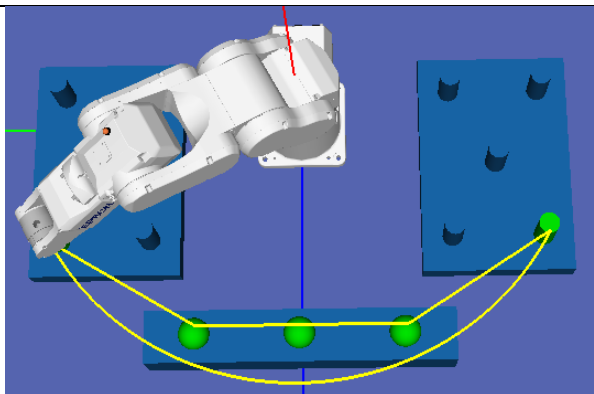
Item	Figura	Código utilizado	Figura do simulador formada pelo código
1			
2			
3			

Figura 34 -Movimentações usando diferentes tipos de comando. Fonte: autoria própria.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, podendo inserir na plataforma quantos arquivos

quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto de cada um dos movimentos realizados no seu simulador

Código utilizado em cada um dos movimentos

4 Comandos de entrada e saídas digitais em robôs industriais

Os robôs industriais possuem uma controladora, responsável por realizar o processamento e controle dos motores, encoders, e garantir o correto funcionamento do programa gravado na sua memória. Estas controladoras possuem conexão com o mundo externo por meio de conectores de entradas e saídas digitais, ou por meio de conectores especiais, voltados para protocolos de comunicação específicos, como CanOpen, Profinet, Profibus, etc (Figura 33).



Figura 35 -Controladora RC700-A da EPSON com conector de entradas e saídas. Fonte: manual de operações EPSON RC+ 7.0

4.1. Mapeando e nomeando as entradas e saídas

No *software* EPSON RC+ 7.0, podemos nomear as entradas e saídas digitais para facilitar o uso pelo sistema, e tornar a programação mais intuitiva. Para isso, acessamos a área do programa chamada “**I/O Label Editor**”.

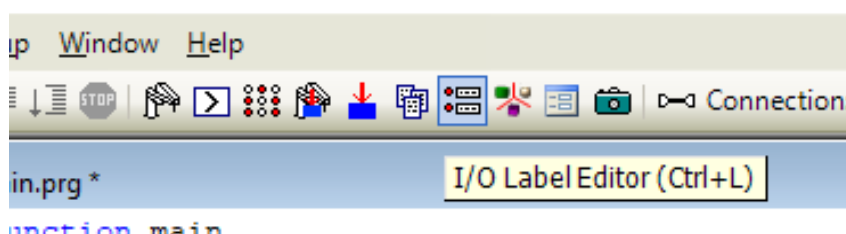


Figura 36 -Botão para acesso a área de edição de nomes. Fonte: autoria própria.

Nesta tela, são listadas todas as entradas e saídas digitais físicas e virtuais que poderão ser acessadas pelo *software* na programação. Nela, podemos identificar os bits e bytes com nomes únicos e específicos, que poderão facilitar no momento do desenvolvimento.

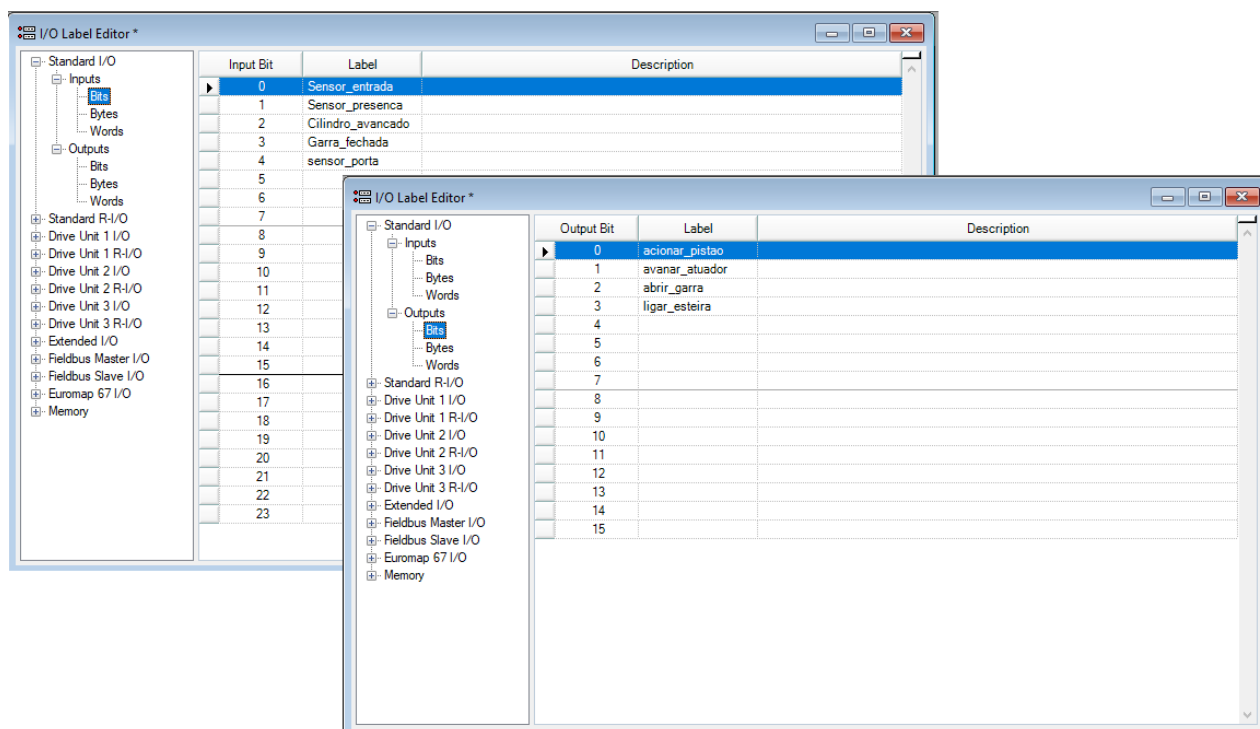


Figura 37 -Tela “I/O Label editor” com valores nos bits de entrada e saída. Fonte: autoria própria.

4.2. Monitoramento das entradas e saídas

Após criar nomes intuitivos para as entradas e saídas, é possível acessar a área **“I/O Monitor”** para realizar o monitoramento dessas portas.

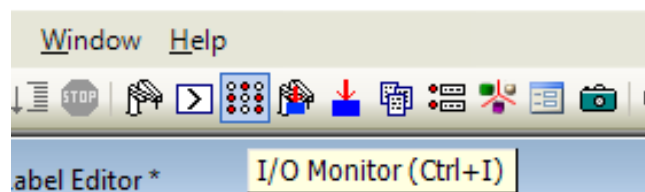


Figura 38 -Botão de acesso a área de monitoramento das entradas e saídas. Fonte: autoria própria.

Na Figura 39, são listadas todas as entradas e saídas, já com as identificações atribuídas anteriormente na janela “**I/O Label Editor**”. Ao lado direito da Figura é possível acionar as saídas digitais realizando um duplo clique com o mouse na bola localizada ao lado do nome escolhido.

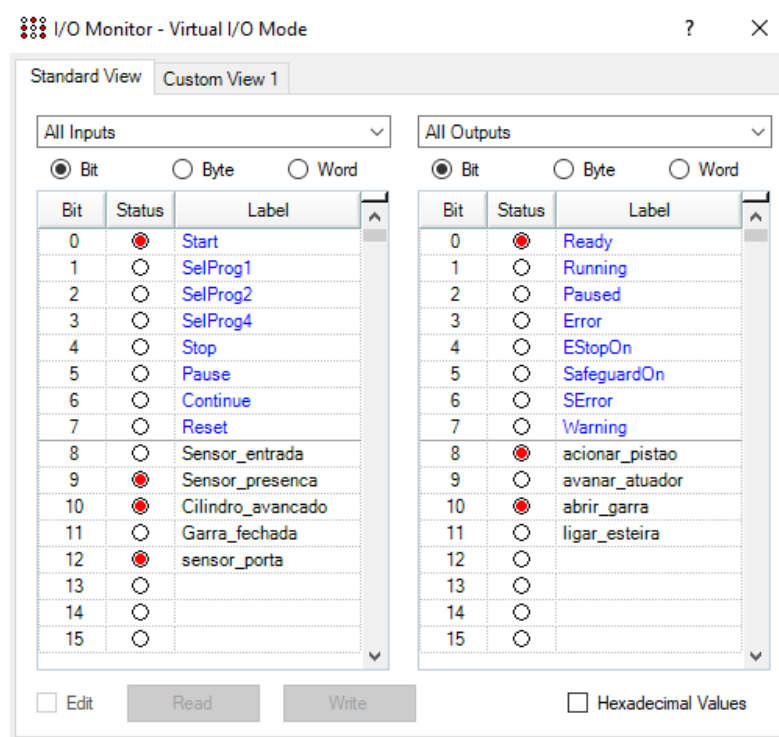


Figura 39 – Tela “I/O Monitor” com os indicadores de acionamento das entradas e saídas digitais. Fonte: autoria própria.

Também nesta mesma tela (Figura 40), ao lado esquerdo, podemos observar os sensores que estão conectados ao controlador, e seu estado (Acionado ou Desligado) de acordo com a cor da bola ao lado do nome.

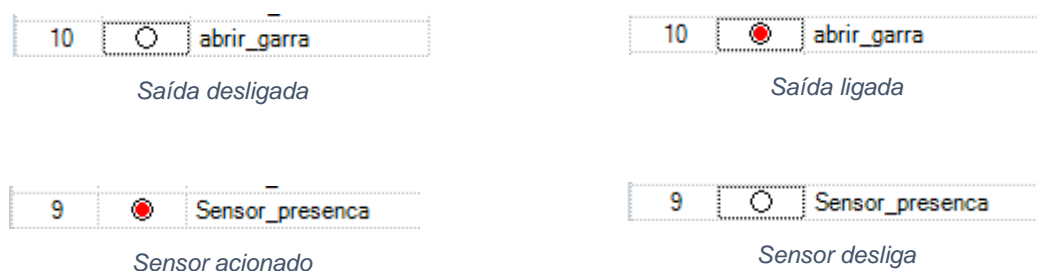


Figura 40 – Telas dos estados dos sensores que estão conectados ao controlador. Fonte: autoria própria. Figura 38 -Botão de acesso a área de monitoramento das entradas e saídas.

Fonte: autoria própria.

4.3. Acessando as entradas e saídas no programa

Agora que as entradas e saídas digitais já estão configuradas no *software*, pode-se acessar seus estados, e acioná-los por meio de comandos no programa principal. Para isso usaremos uma série de comandos que serão mostrados a seguir.

- **Sw**(bit | nome de entrada)
 - Retorna verdadeiro(1) ou falso(0) dependendo do estado do bit.
 - Caso o usuário tenha criado um nome para o bit, esse pode ser utilizado na função.

```
If Sw(garra_aberta) Then
EndIf
```

- **On/Off** (bit | Nome de saída) [tempo] [parelo] [Forçado]
 - Modifica o estado da saída para Verdadeiro (1) ou para Falso (0)
 - Tempo: opcional que especifica o tempo em que a saída ficará ativa;

- Paralelo: opcional que determina se a próxima ação será executada:
 - Imediatamente após ligar a saída – deve-se usar “0”
 - Aguardará o final da execução – deve-se usar “1” (padrão)
- Forçado
 - Liga a saída mesmo em emergência de tasks específicas.

```
On abrir_garra
Off abrir_garra
```

Além das entradas e saídas digitais, que possuem uma correspondência no mundo físico, o software EPSON RC+ permite o uso de memórias digitais, que podem ser usados para controle do processo interno do programa, ou para comunicação com softwares terceiros por meio de protocolos virtuais (TCP/IP) por exemplo.

Para acessá-las pode-se usar os comandos listados a seguir:

- **MemSw**(bit | nome de entrada)
 - Retorna verdadeiro(1) ou falso(0) dependendo do estado do bit.

```
Wait MemSw(comando_PC)
Wait MemSw(1)
```

- **MemOn/MemOff** (bit | nome de saída)
 - Liga/desliga uma variável de memória.

```
MemOn comando_PC
MemOn 1
```

5 Loops e funções no programa

5.1 Loops em SPEL+

Para um funcionamento estruturado do programa, se faz necessário a inclusão de loops e funções ao código fonte, garantindo a ordenação e modularização das tarefas executadas pelo código fonte.

Dentro da linguagem SPEL+, temos dois principais loops que poderão ser utilizados nos programa. O Loop WHILE e o loop FOR.

- **Loop WHILE**

- Executará a tarefa indefinidamente, até que uma condição seja satisfeita.
- O Loop While sempre é executado ao menos uma vez.

```
Do While sensor = True
  Go P0
  Go P1
  Go P2
Loop
```

Quando utilizamos o loop While, temos que nos atentar ao fato de se criar um loop infinito, que não possui condição de saída, e pode fazer com que o programa entre em um modo indesejado de operação.

Dependendo do processo, pode ser necessário o uso do loop infinito, mas são exceções que devem ser analisadas com muita cautela.

- **Loop FOR**

- Executará a tarefa um número pré-determinado de vezes.
- Tem-se maior controle sobre a quantidade de iterações.
- Dificilmente será feito um loop infinito.


```
Integer contador  
  
Motor On  
  
For contador = 0 To 2  
    Go P(contador)  
Next
```

Com o loop for, temos total controle da quantidade de vezes que a tarefa será executada, pois o limite de iterações deve ser passado como parâmetro na hora da chamada da função.

5.2 *Exercício 3: Utilização dos comandos On/Off e Sw*

Modifique seu programa inicial com comandos de **On**, **Off**, **SW** e **Wait** para as I/Os. Seu programa deverá ter pelo menos 1 uso de cada um desses comandos. Use o I/O Monitor para verificar o funcionamento das portas.

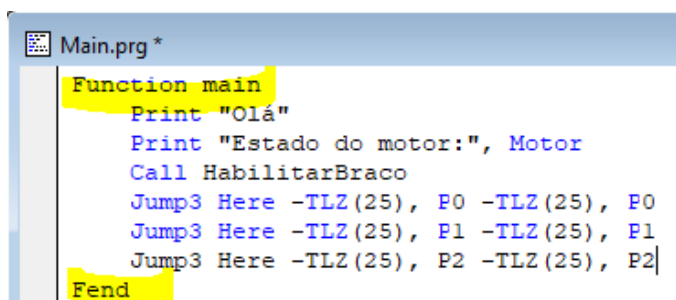
Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, podendo inserir na plataforma quantos arquivos quiser ou pode-se enviar arquivos inteiros em qualquer formato contanto que estejam compactados em .rar ou .zip, inserindo todos os arquivos de uma vez só. Além das evidências justique a Lógica que você utilizou para realizar o exercício.

5.3 *Funções em SPEL+*

Com a criação de funções, podemos criar códigos modulares e bem estruturados, separados de acordo com as suas ações no programa principal, facilitando a leitura e manutenção do sistema ao longo da sua vida.

Uma **função (Function)** indica o começo de um grupo de comandos dentro do seu código. Para indicar o fim desse grupo, é utilizado o comando **Fend**. Múltiplas funções podem existir dentro de um mesmo programa/projeto. A

função **Main** é a principal do seu programa, e será executada ao energizar o controlador.



```

Main.prg *
Function main
  Print "Olá"
  Print "Estado do motor:", Motor
  Call HabilitarBraco
  Jump3 Here -TLZ(25), P0 -TLZ(25), P0
  Jump3 Here -TLZ(25), P1 -TLZ(25), P1
  Jump3 Here -TLZ(25), P2 -TLZ(25), P2
Fend
  
```

Figura 41 –Programa padrão, com destaque à declaração da função MAIN. Fonte: autoria própria.

Além da função MAIN, pode-se criar funções customizadas, sem limite de quantidade, para melhor organizar seu código, e evitar a repetição de trechos que realizam operações idênticas ao longo do processo.

Para criar uma função usa-se o seguinte formato:

- **Function** nome [{ByRef |ByVal}] varName **As** tipo
 - **ByRef:** Se refere à variável que será tratada pela função. Nesse caso, as mudanças internas da função serão refletidas fora dela.
 - **ByVal:** Use esse parâmetro caso você não queira mudar o valor da variável fora da função (**Padrão**)

```

Function robot_near_rejeito
  If CX(Here) > 50 Then
    Go rejeito_top CP
    Go return_rejeito_P1 CP
    Go return_rejeito_P2 CP
    Go Path_Inspec_SN CP
  EndIf
Fend
  
```

Figura 42 –Função da chamada simples. Fonte: autoria própria.

```

Function Check_Divisao(A As Double, B As Double) As Boolean
    If A > 0 Then
        divisao = B / A
        Check_Divisao = True
    Else
        Check_Divisao = False
    EndIf
Fend

```

Figura 43 –Função com chamada incluindo parâmetros. Fonte: autoria própria.

No caso da função com parâmetros, podemos incluir parâmetros de entrada que serão utilizados durante a operação no interior da função, e estarão acessíveis somente às chamadas realizadas dentro delas, e também retornar valores ao programa principal como resultado do processo.

Para realizar a chamada das funções dentro do programa usamos o comando **CALL**.

A instrução **Call** faz com que o programa desvie o fluxo de execução para a função definida na chamada do comando.

Após o final da execução da função referenciada, o fluxo retorna à função principal e continua na linha seguinte.

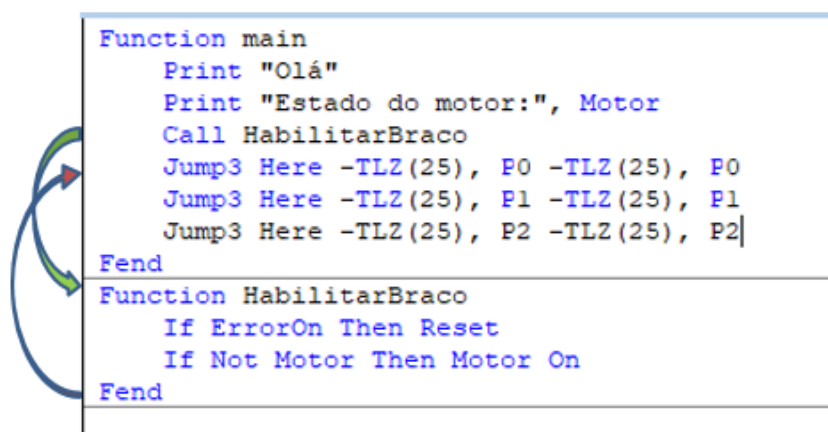


Figura 44 –Fluxo de execução da função CALL. Fonte: autoria própria.

6 Orientação da ferramenta do robô e sistema de coordenadas relativo

Quando trabalhamos com robôs industriais, é comum utilizarmos ferramentas customizadas para nossas operações, conectadas ao efetuador do robô. Essa ferramenta pode ser usada para manipulação, ou mesmo a realização de alguma tarefa no objeto em questão.

Como mostrado até o momento, todas as coordenadas salvas no software estão ligadas ao efetuador do robô sem ferramentas, ou seja, as coordenadas armazenadas na memória do dispositivo estão todas vinculadas sem considerar a presença de uma ferramenta/garra na ponta do robô.

Para criar-se um sistema de referências que condiz com a real situação do robô, com a ferramenta inclusa, se faz necessário realizar algumas configurações e ajustes no programa.

A origem da ferramenta (Tool 0) é a flange do sexto eixo. Quando todos os eixos estão na posição de 0°, o eixo de coordenadas Tool 0 está alinhado conforme a imagem

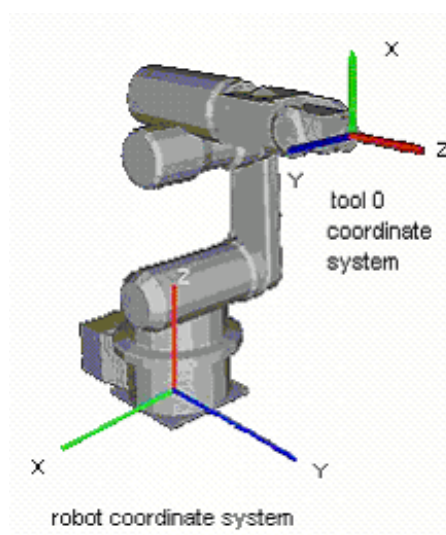


Figura 45 –Sistema de coordenadas da ferramenta de um robô de 6 eixos. Fonte: Manual de operação EPSON RC+ 7.0

A ferramenta padrão dentro do software é a tool 0. No programa EPSON RC+ 7.0, podemos usar as ferramentas Tool 1-15 para aplicar diferentes offsets no nosso processo.

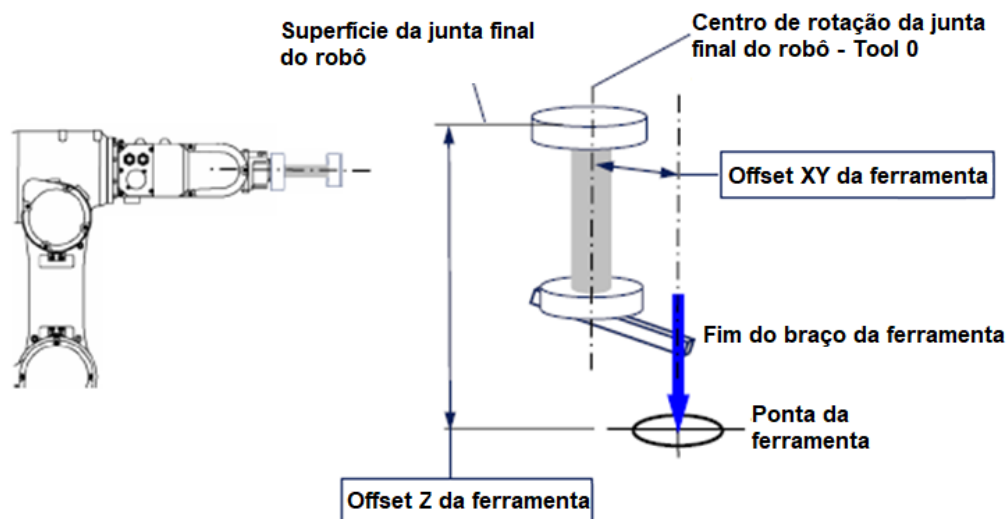


Figura 46 –Deslocamento (offset) entre o Tool 0 e a ponta da ferramenta.. Fonte: Manual de operação EPSON RC+ 7.0

Na janela Robot Manager (F6), é possível navegar até a tela de “Tools”, onde estão listadas todas as ferramentas usadas pelo robô, e seus offsets com relação ao Tool 0.

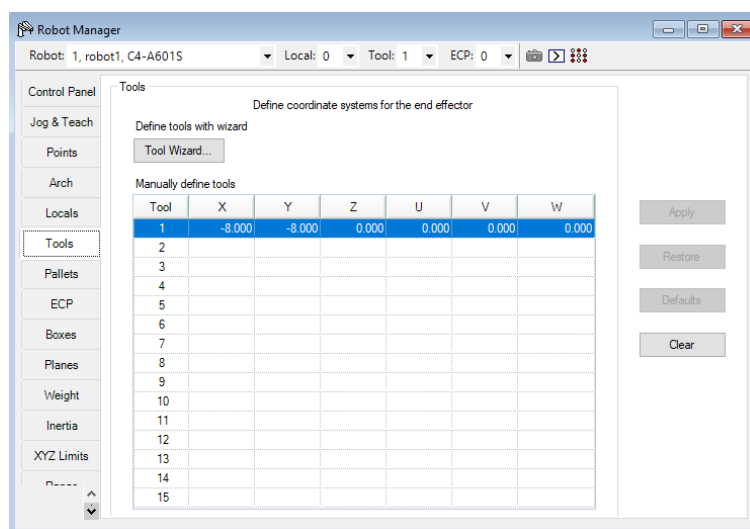


Figura 47 –Janela de configuração das coordenadas de deslocamento das ferramentas. Fonte: autoria própria.

Sabendo o deslocamento exato da ferramenta em relação do Tool 0, é possível inserir o valor manualmente na tabela. Caso esse valor seja desconhecido, ou a geometria seja muito complexa, pode-se realizar a calibração por meio do “Tool Wizard” localizado na parte superior da tela.

7 Reconhecimento de imagem e correção de posicionamento

Durante aplicações de automação em processos fabris, faz-se cada vez mais necessária a utilização de câmeras para correção do posicionamento dos robôs, visando atender à uma gama de produtos que possuem diferenças em seu dimensional, ou na posição em que esse material é recebido e apresentado à célula robótica.

Como exemplo, imagine uma fábrica que produz peças plásticas por meio de extrusão, e essas peças são então depositadas em uma esteira por gravidade, e na estação seguinte precisam ser retiradas por um robô para fazer acabamentos na peça. As peças plásticas poderão estar em qualquer orientação na esteira, desde que com a mesma face para cima, e o robô deverá corrigir seu posicionamento no momento de retirar esse produto para realizar a operação de acabamento.

Tendo cenários como esse em vista, os fabricantes de robôs desenvolveram sistemas de visão baseados em câmeras, que identificam elementos do ambiente, e realizam a correção automática do posicionamento do efetuador para garantir a correta pega. No caso do fabricante EPSON, foi criado o CV2 (Compact Vision 2).



Figura 48 – Sistema CV2 da EPSON com uma câmera. Fonte: Manual de operação CV2 EPSON.

Este *hardware* extra é conectado à controladora RC700 por meio de um cabo ethernet, e realiza a troca de informações para correto funcionamento do sistema. Com ele, não há a necessidade de processar a imagem em um computador, e a solução se torna mais compacta, reduzindo a necessidade de aquisição de equipamentos extras.

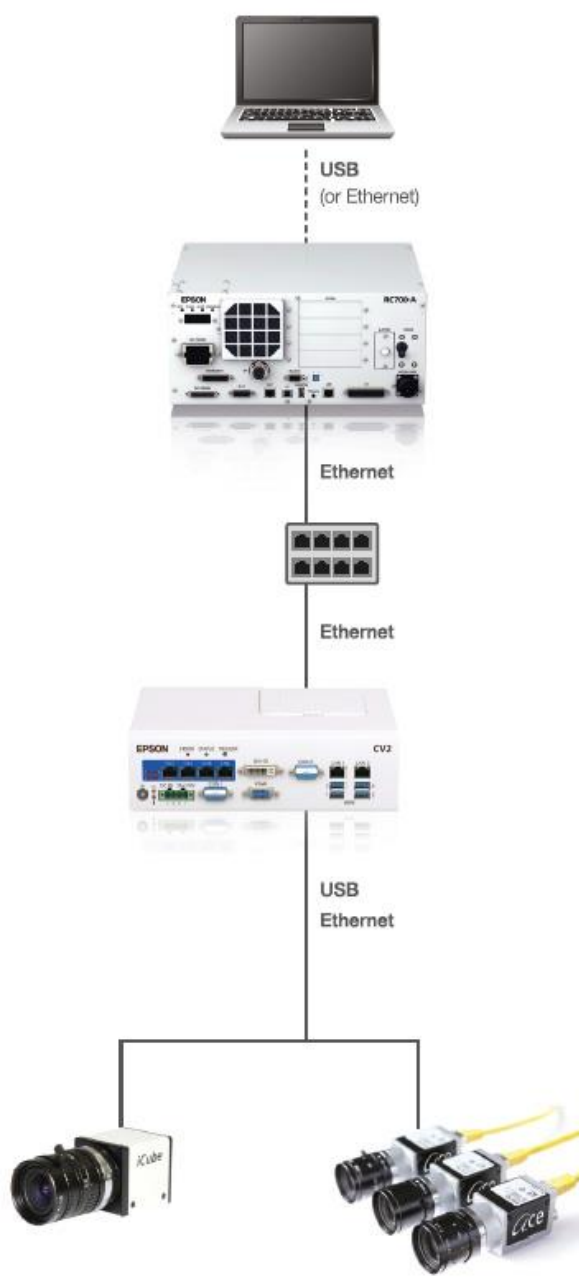


Figura 49 –Arquitetura de conexão do modelo CV2 com a controladora RC700. Fonte: Manual de operação CV2 EPSON.

7.1 Fixação e posicionamento da câmera

Para o correto funcionamento da aplicação, deve-se analisar o tipo de detecção que se deseja, e posicionar a câmera conforme a necessidade. Para este modelo, pode-se usar de 6 tipos de fixação diferentes.

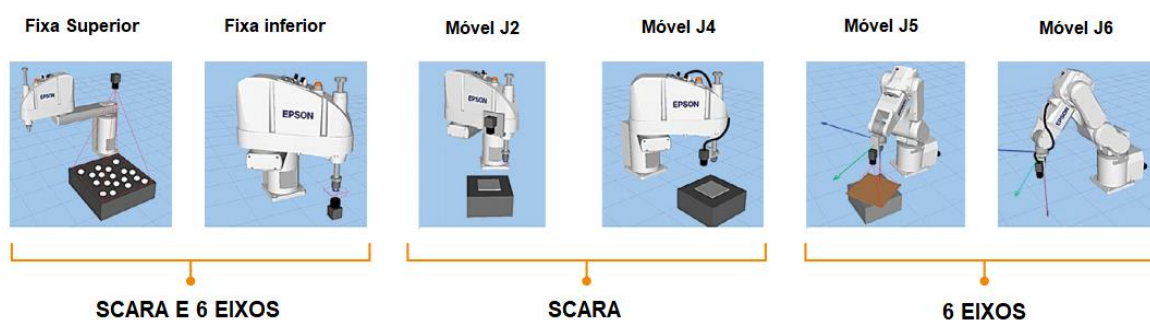


Figura 50 –Possibilidade de fixação da câmera em uso em conjunto ao robô. Fonte: Guia rápido de operação CV2 EPSON.

7.2 Configuração no software EPSON RC+ 7.0

Após a realização da montagem física, pode-se então realizar a parametrização no software EPSON.

O primeiro passo, é realizar a correção da ferramenta que será utilizada em conjunto com a câmera, para que o programa consiga realizar a correta calibração do sistema.

Para isso, deve-se utilizar do conteúdo mostrado no item 6 dessa apostila.

Após a calibração da ferramenta, deve-se então iniciar a calibração da câmera, selecionando o botão Vision (Ctrl+F9).

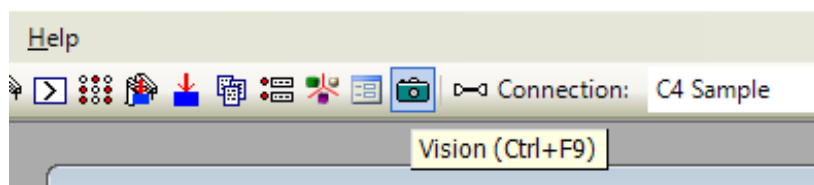


Figura 51–Botão Vision na tela do software EPSON. Fonte: autoria própria.

Isso fará com que a tela Vision Guide seja exibida na interface. Primeiramente, precisaremos inserir uma inspeção, para posteriormente realizar a calibração do robô.

7.2.1 Criação de uma sequência de inspeção

Para a criação de uma sequência de inspeção, deve-se selecionar a opção **“Sequence”** -> **“New Sequence”**:

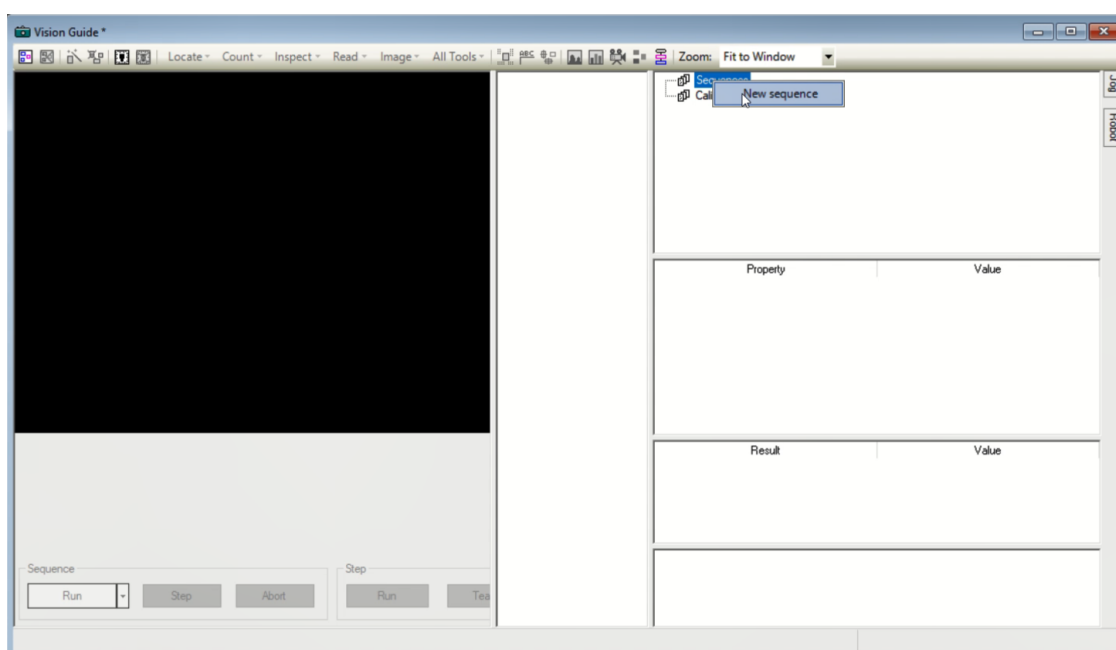


Figura 52 –Tela do Vision Guide com a adição de um novo Sequence. Fonte: autoria própria.

Assim a janela “New Vision Sequence” seja exibida. Nela, deve-se colocar o nome da análise que será realizada, e selecionar qual câmera será utilizada no processo.

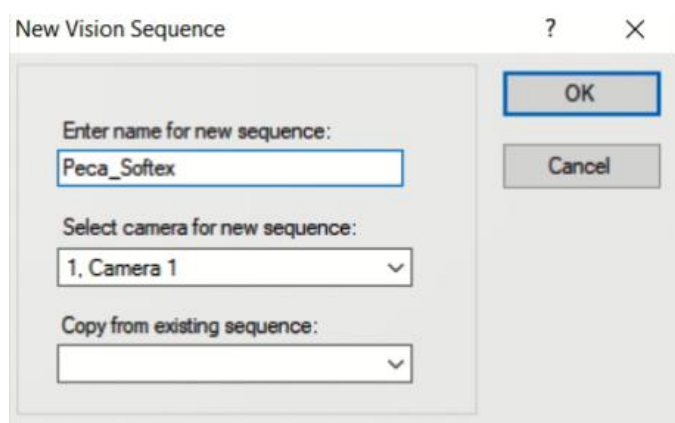


Figura 53 –Janela New Vision Sequence com nome e seleção de câmera. Fonte: autoria própria.

Ao selecionar o botão “OK”, o programa voltará para a tela do Vision Guide, sendo possível observar a imagem da câmera selecionada na interface.

Para esse exemplo, faremos a análise de peças circulares. Então pode-se navegar no menu superior até a opção “Count” -> “Geometric”.

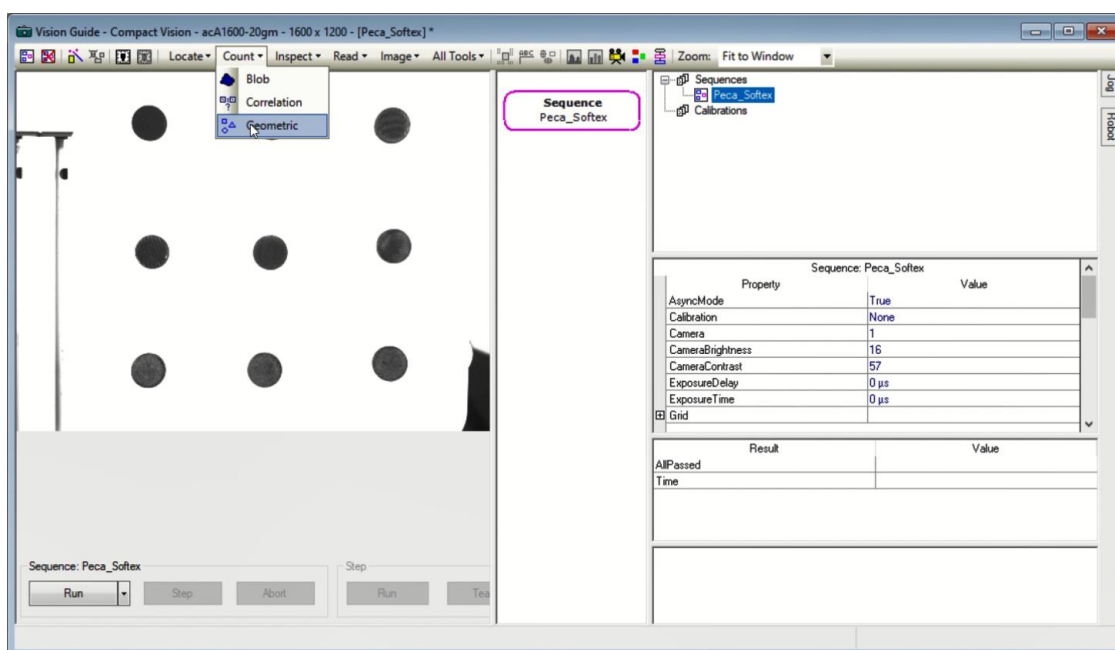


Figura 54–Tela do Vision Guide com a seleção da opção “Geometric” para detecção de formatos geométricos. Fonte: autoria própria.

Com a seleção dessa opção, uma região na cor roxa será exibida na tela. Ajuste seu formato e tamanho para se encaixar ao objeto que se deseja inspecionar.

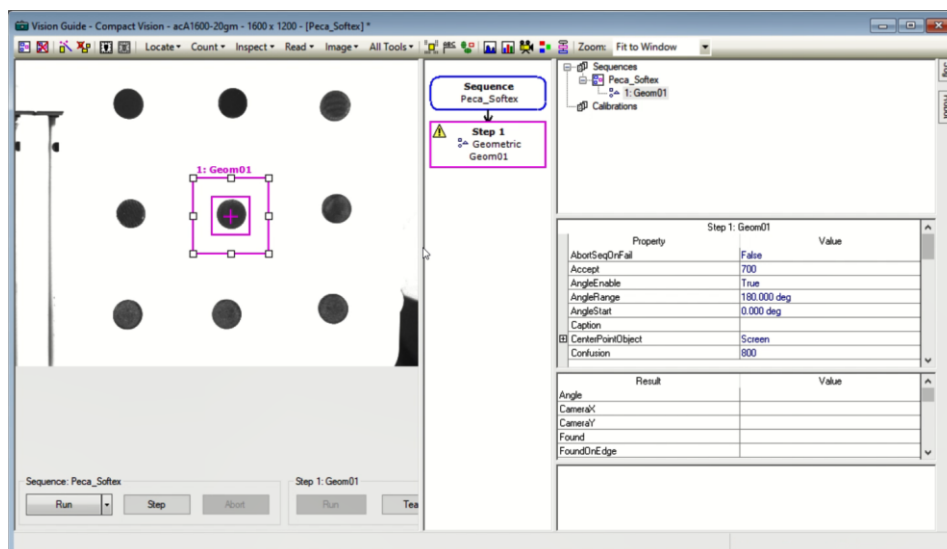


Figura 55 –Tela Vision Guide com a detecção da peça em fase de configuração. Fonte: autoria própria.

Após o posicionamento da área de interesse, ou *Region of Interest* (ROI), selecione o botão TEACH, localizado na parte inferior da tela para que o software reconheça o formato previsto.

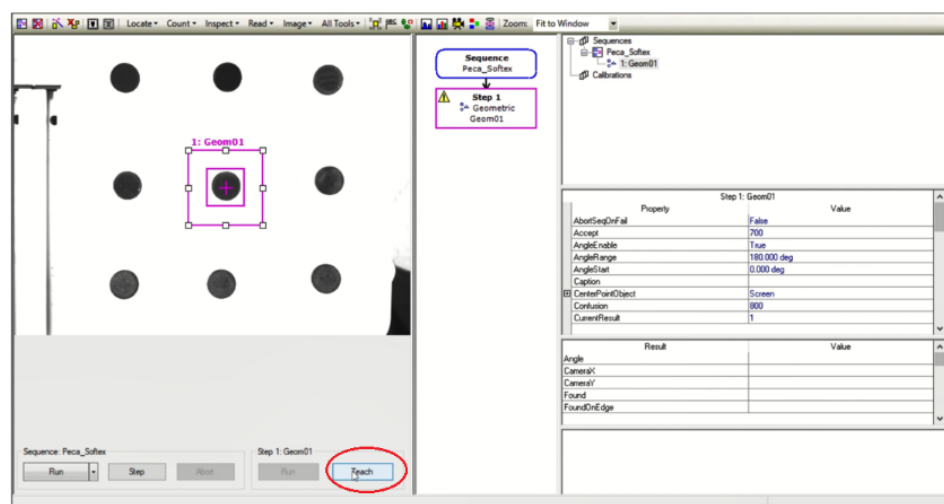


Figura 56 –Botão de Teach na interface Vision Guide. Fonte: autoria própria.

Perceba que, ao selecionar o botão “Teach”, o botão “Run” ao lado desse ficou habilitado. Selecione o botão “Run” para validar se sua inspeção foi bem sucedida. A sua ROI deverá ficar verde, e uma cruz se formará em cima da peça detectada.

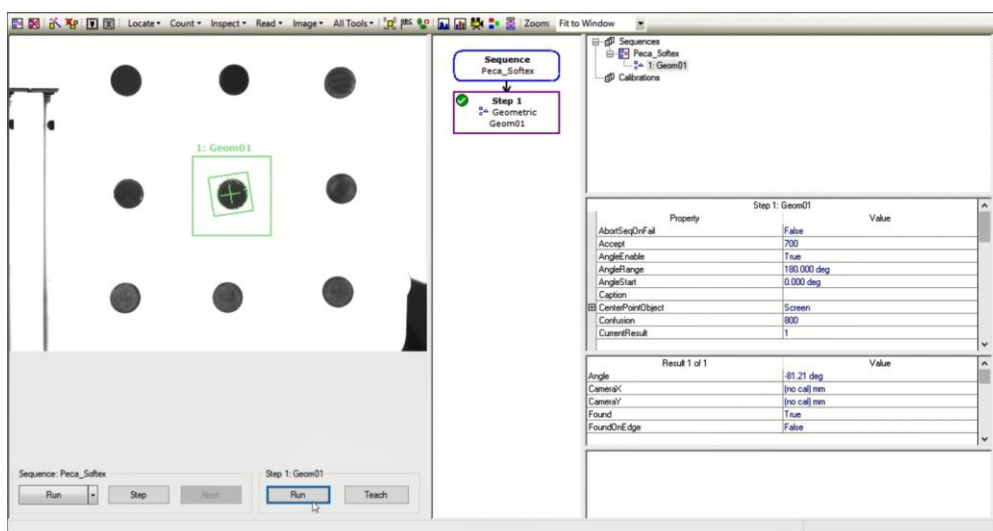


Figura 57 –Inspeção realizada com sucesso e objeto detectado. Fonte: autoria própria.

7.2.2 Criação de uma calibração do robô

Agora que temos o modelo de inspeção e sequência cadastrado, podemos criar uma calibração para o robô realizar correções automáticas de acordo com o posicionamento das peças que serão apresentadas ao sistema.

Com o botão direito do mouse, selecione a opção Calibrations -> **New Calibration**.

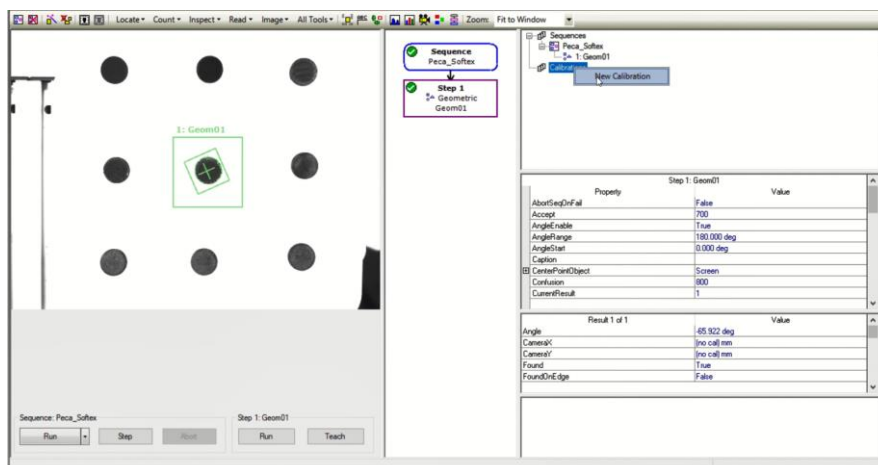


Figura 58 –Adição de uma nova calibração no Vision Guide. Fonte: autoria própria.

Na janela que é exibida (Figura 59), insira o nome da calibração, e a câmera que será utilizada, e selecione Next >.

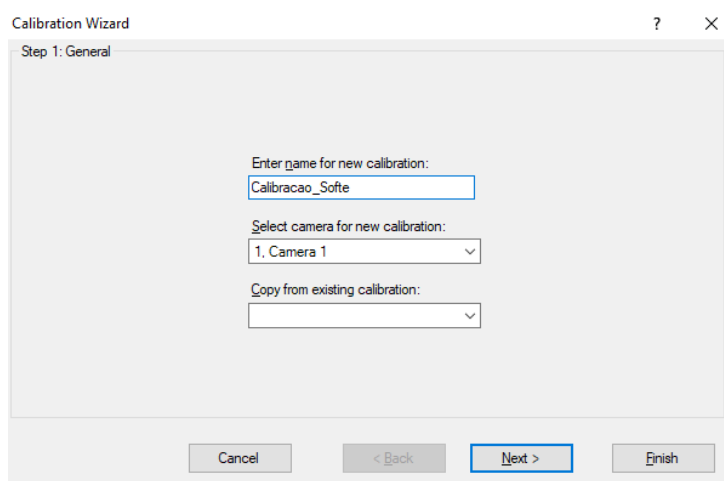


Figura 59 –Tela de criação de calibração com nome e câmera. Fonte: autoria própria.

Na tela seguinte (Figura 60), selecione a posição em que a câmera está instalada na solução, para nosso exemplo, usaremos a câmera no punho do robô, ou seja, na junta de número 6. Acompanhe a imagem que é exibida para eventuais dúvidas

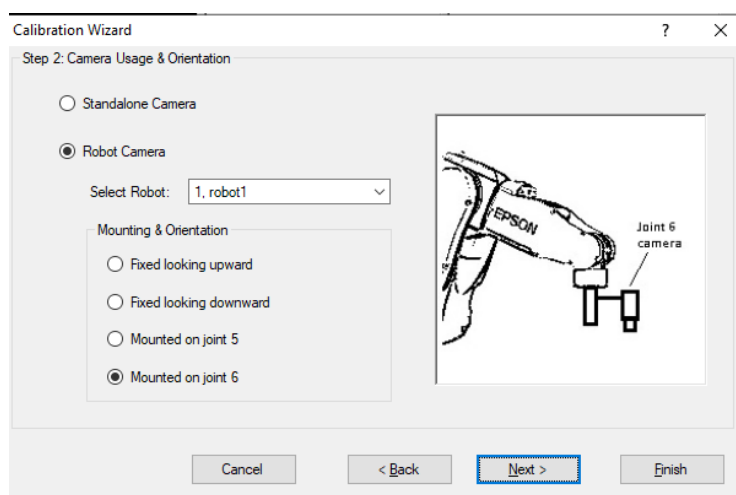


Figura 60 –Seleção da posição da câmera no robô. Fonte: autoria própria.

Na Figura 61, deveremos escolher qual a peça de referência será utilizada para calibrar o robô. Nesse caso, devemos selecionar a sequência que criamos no procedimento anterior.

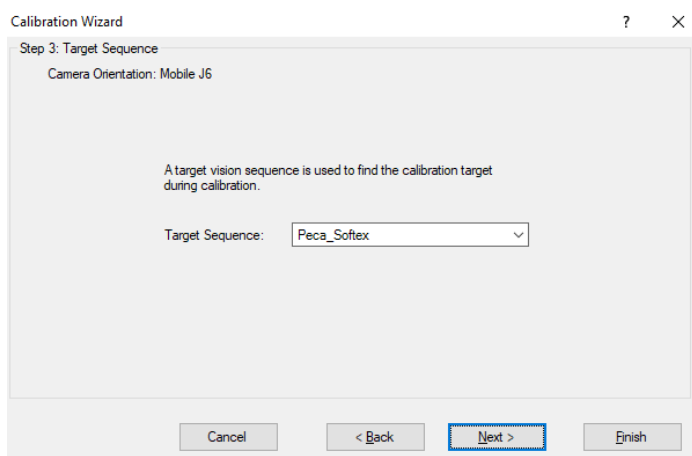


Figura 61 –Seleção da peça de referência. Fonte: autoria própria.

Na próxima tela (Figura 62), devemos escolher o sistema de coordenadas que será usado na nossa calibração, ou seja, a coordenada da ferramenta que será utilizada. No caso do exemplo, vamos manter em 0.

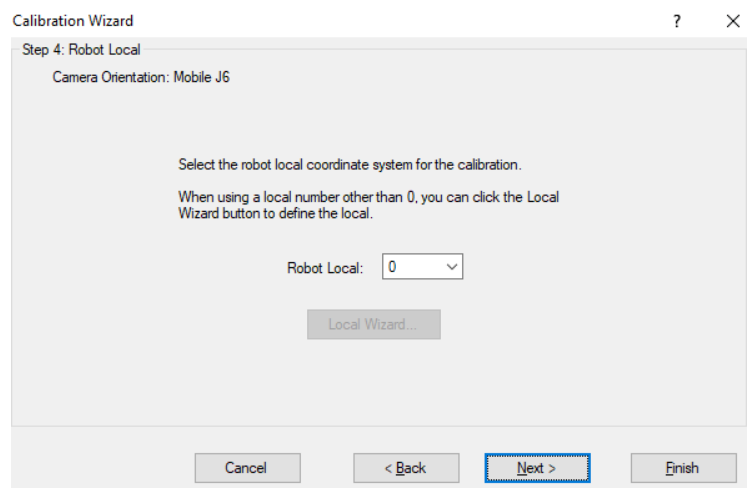


Figura 62 –seleção de ferramenta que será calibrada. Fonte: autoria própria.

No próximo passo (Figura 63), devemos escolher o tipo de referência que será utilizada durante o processo de calibração. Nesse caso, vamos utilizar o modelo “Taught Point” e desabilitar a opção “AutoReference”.

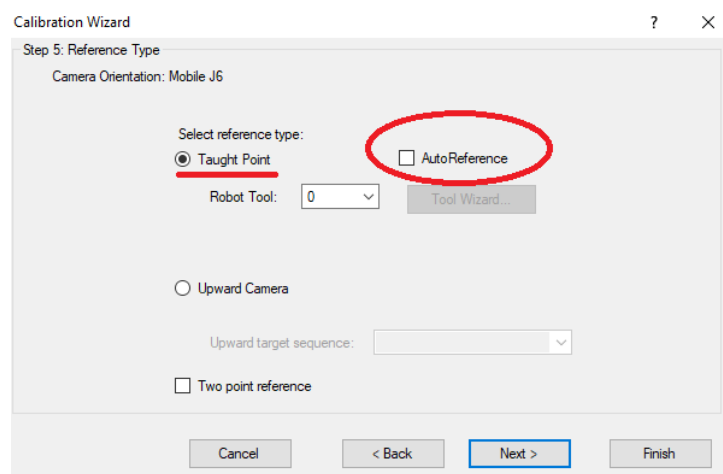


Figura 63 –Tipo de referência. Taught Point sem AutoReference. Fonte: autoria própria.

Na Figura 64, devemos selecionar o modelo de geração dos pontos de calibração da câmera. Vamos deixar essa opção selecionada no modo automático.

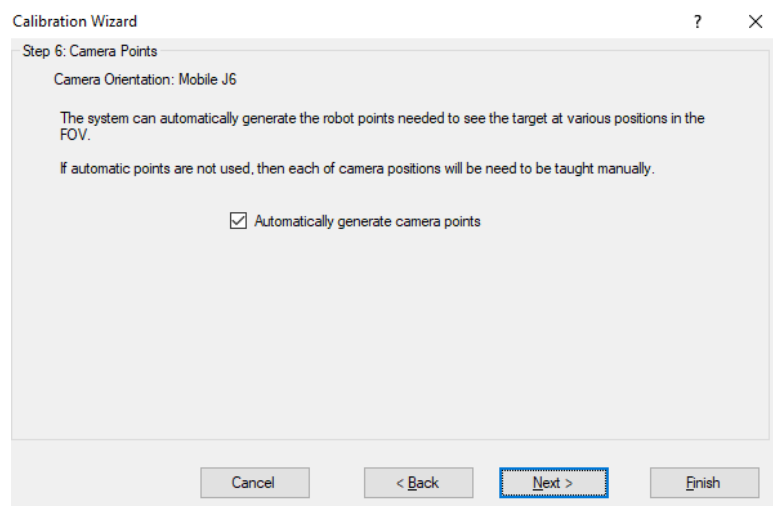


Figura 64 –Pontos da câmera com a seleção automática da geração dos pontos. Fonte: autoria própria.

No próximo passo (Figura 65), vamos desabilitar o uso de correções relacionadas à distorção da lente da câmera. Como nossa aplicação é simples, onde realizaremos a pega de peças em uma área controlada, não precisamos compensar essa distorção mas, para aplicações que requerem alta precisão do sistema, é necessário realizar a calibração da distorção da lente.

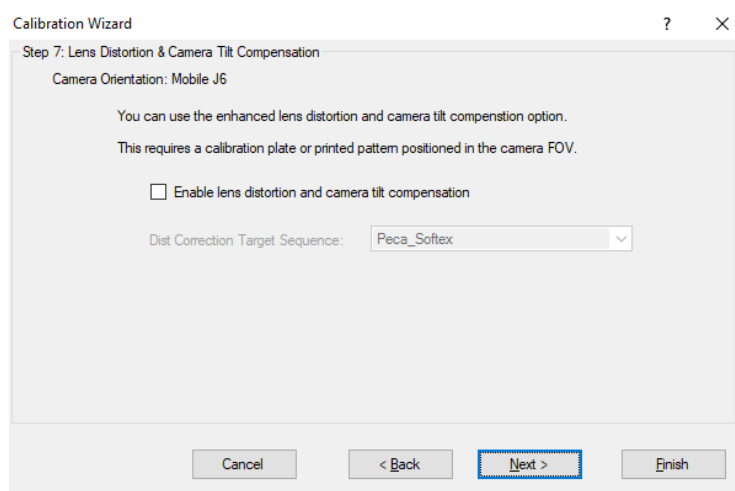


Figura 65 –Correção de distorção da lente desabilitada. Fonte: autoria própria.

No seguinte passo (Figura 66) pode-se habilitar o uso de iluminação externa, como flash ou luz controlada. No nosso caso não será necessário, mas

caso seja utilizado, deve-se especificar o atraso no acionamento da luz, e qual saída digital da controladora do robô está o comando.

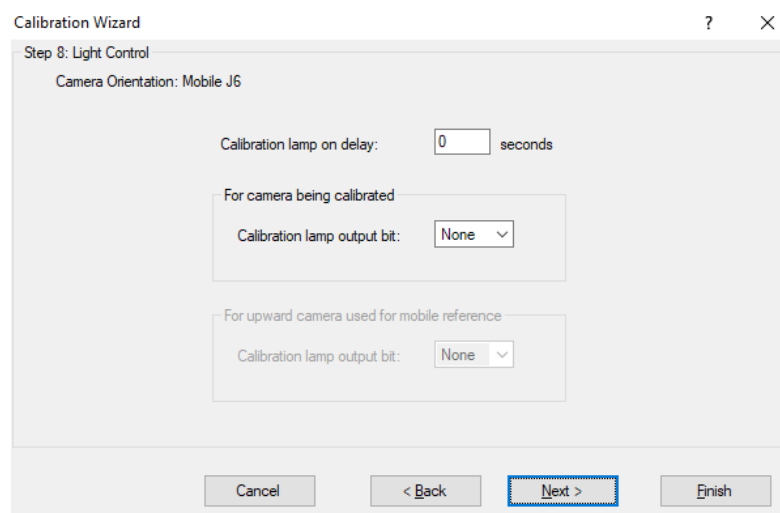


Figura 66 –Controle de iluminação externa para uso do sistema de visão. Fonte: autoria própria.

Agora devemos selecionar os parâmetros de movimentação que o robô realizará durante o processo de calibração, especificando sua velocidade, aceleração e atraso entre os movimentos que serão realizados durante o processo. Vamos deixar nos valores padrões, e avançar para a última etapa (Figura 67).

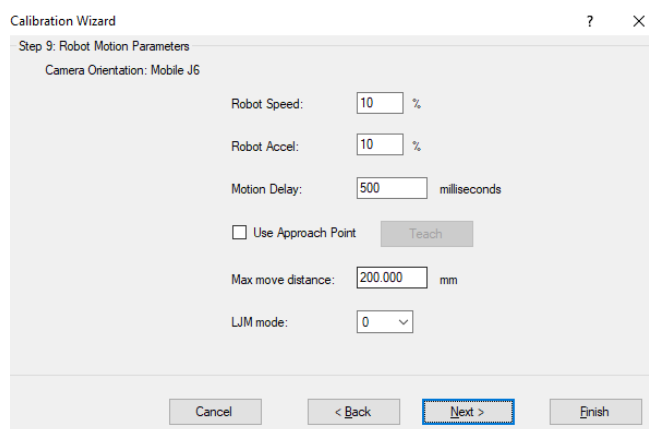


Figura 67 –Parâmetro de movimentação do robô durante o processo de calibração. Fonte: autoria própria.

Por fim, podemos verificar as informações inseridas, e selecionar o botão “Finish” (Figura 68).

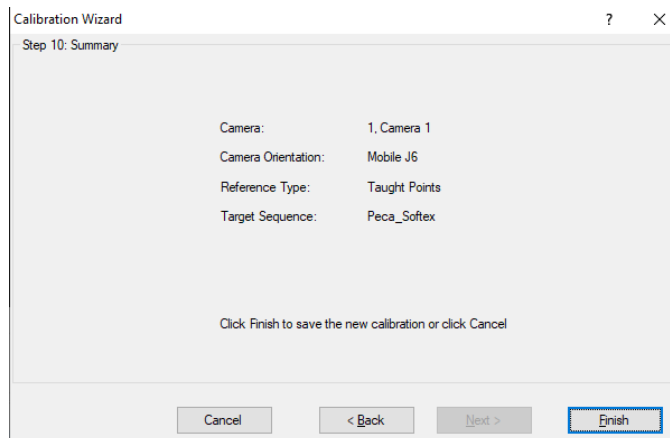


Figura 68 –Resumo da calibração criada. Fonte: autoria própria.

7.2.3 Criando a rotina de calibração

Agora que já temos uma sequencia para ser usada de base para a identificação do sistema, e uma rotina de calibração configurada, devemos iniciar o processo de calibração automática do robô.

Selecione o botão “Teach Points” localizado na parte inferior esquerda da tela do Vision Guide.

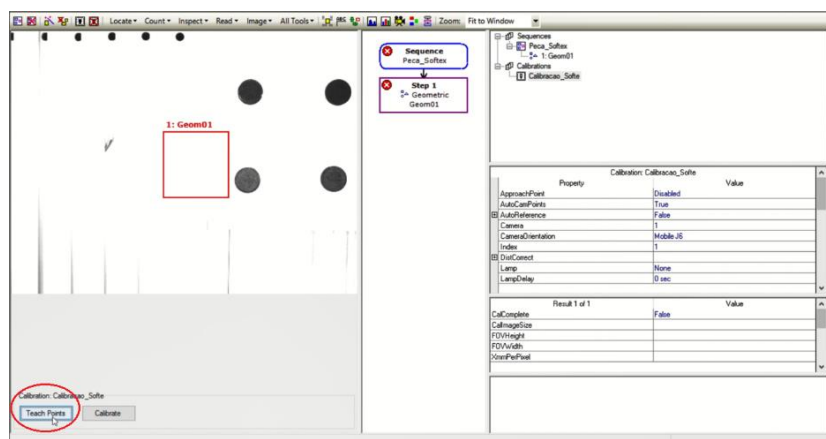


Figura 69 –Botão Teach Points na janela Vision Guide. Fonte: autoria própria.

Ao selecionar o botão, a interface mudará como mostrado na figura abaixo, e será necessário usar os controles localizados ao lado direito da tela para movimentar o robô, de modo à deixar a peça no centro do quadrado mostrado, e selecionar o botão Teach.

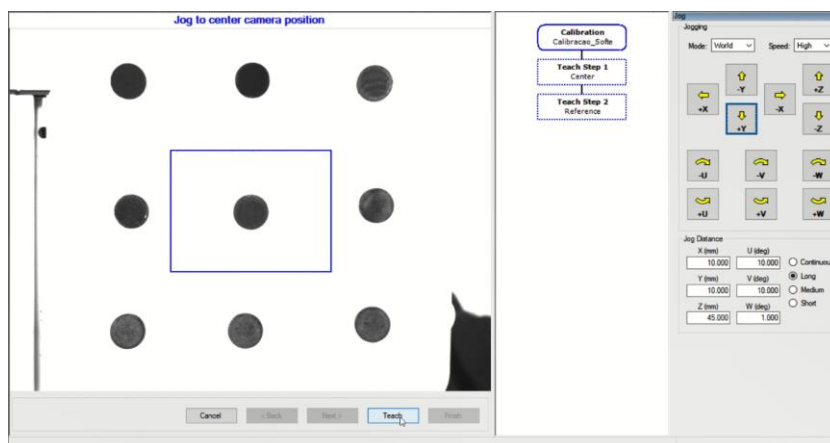


Figura 70 –Botões de movimentação do robô à direita e botão Teach localizado na parte inferior. Fonte: autoria própria.

Ao clicar no botão, a tela mudará novamente, agora mostrando uma imagem do robô, e indicando que o usuário, sem mexer com a peça inspecionada, deverá posicionar a ferramenta alinhada ao centro da peça em questão, e pressionar o botão Teach.

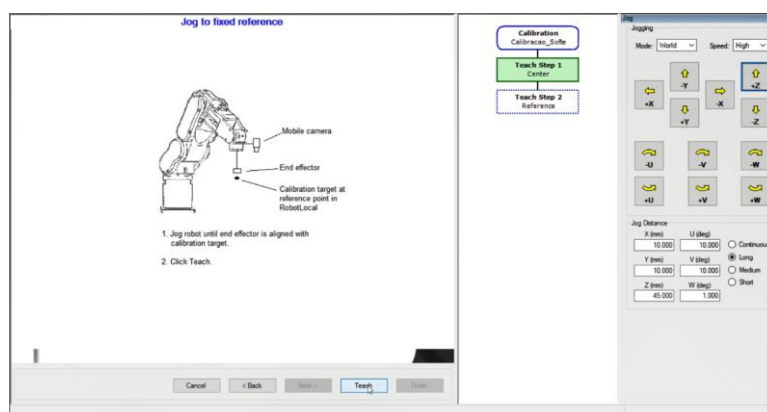


Figura 71 –Tela de calibração da ferramenta com indicação para movimentar ao centro da peça e botão Teach. Fonte: autoria própria.

Ao selecionar o botão Teach. A imagem da câmera aparecerá novamente, agora deslocada pois a ferramenta é quem está no centro da peça, e a mensagem “All calibration points are now taught” aparecerá na parte superior.

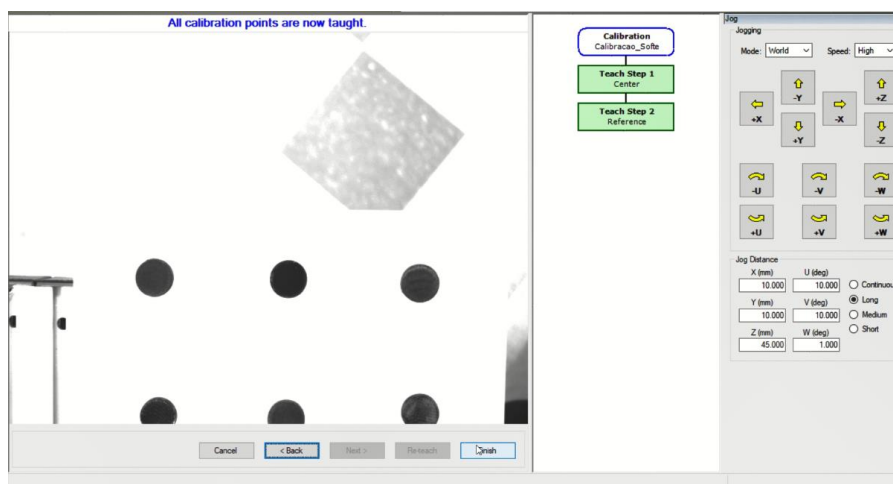


Figura 72 –Tela final de calibração da câmera e da ferramenta do robô. Fonte: autoria própria.

7.2.4 Executando a rotina de calibração

Agora com a rotina de calibração criada, podemos executá-la. Para isso, selecione o botão “Calibrate”, localizado na parte inferior esquerda da tela.

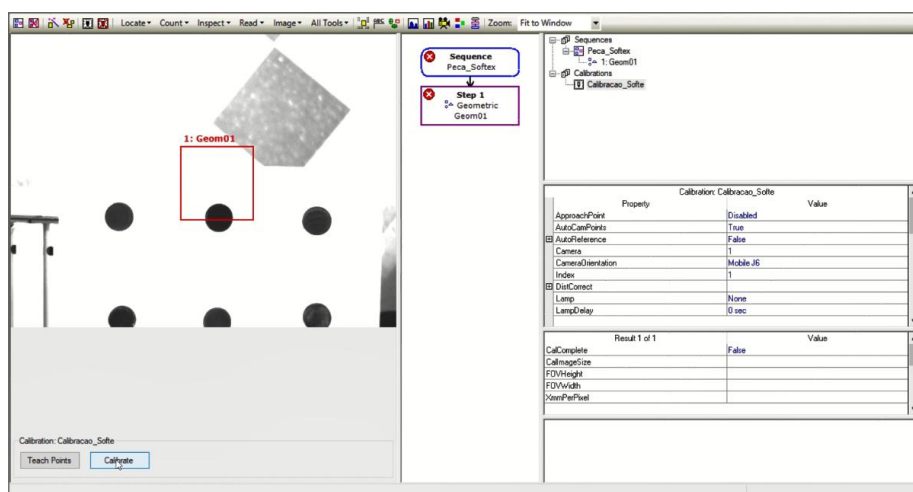


Figura 73 –Botão Calibrate usado para iniciar a rotina de calibração. Fonte: autoria própria.

Uma tela de aviso aparecerá, indicando que a calibração iniciará, e avisando sobre os possíveis riscos. Selecione a opção “yes”.

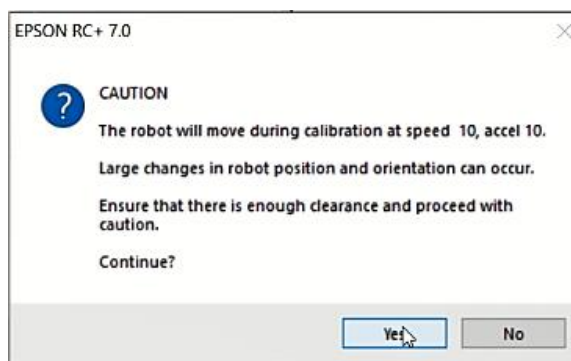


Figura 74 –Janela de aviso do robô ao iniciar a rotina de calibração. Fonte: autoria própria.

O robô começará a se movimentar, procurando a peça calibrada. Mantenha seu mouse próximo ao botão “Abort” para cancelar a operação caso identifique alguma situação de risco.

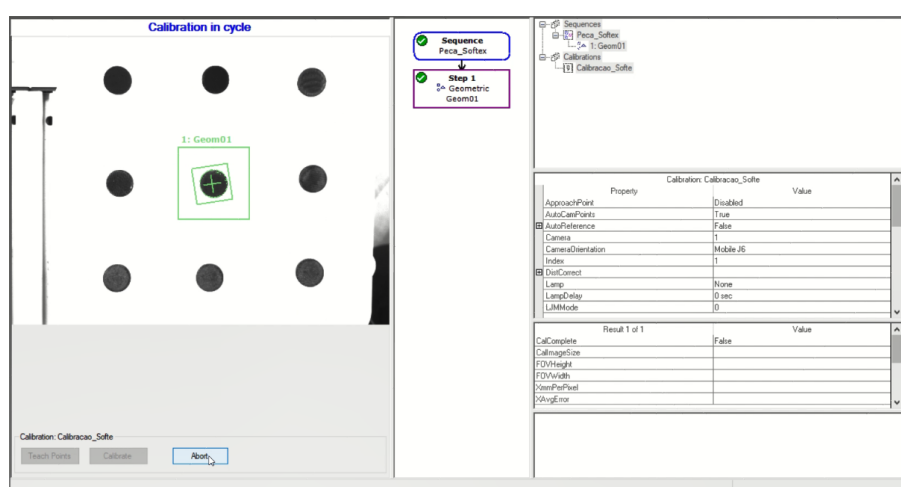


Figura 75 –Botão Abort usado para parar a calibração em caso de identificação de riscos.

Fonte: autoria própria.

O robô fará então uma série de movimentos, sempre navegando em um padrão matricial de 9 posições para identificar a peça e suas coordenadas na

câmera e no controlador. As imagens se alternarão aproximadamente como mostrado na sequência abaixo (Figura 76).

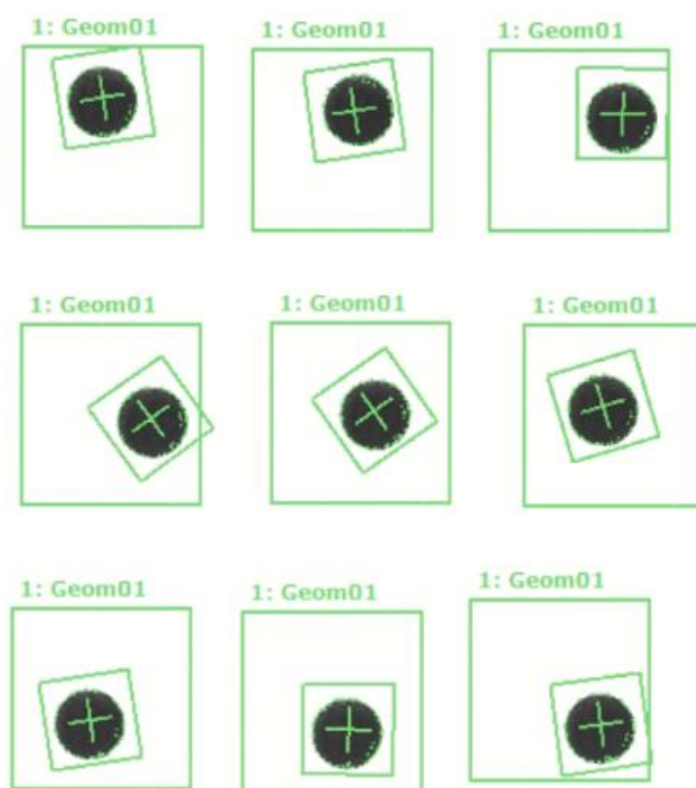


Figura 76 –Padrão de 9 imagens em formato matricial obtidas no processo de calibração.

Fonte: autoria própria.

Caso a calibração seja bem sucedida, a tela abaixo será exibida, mostrando as conversões de milímetro e pixel obtidos no processo. Isso mostra que a aplicação está apta à ser testada.

The screenshot shows a 'Calibration Complete' dialog box with two sections: 'Previous values' and 'New values'. The 'Previous values' section contains empty input fields for X mm per pixel, Y mm per pixel, Max X error, Max Y error, Avg X error, Avg Y error, X tilt, Y tilt, and FOV. The 'New values' section contains the same fields with numerical values: X mm per pixel: 0.1881, Y mm per pixel: 0.1978, Max X error: 0.9331, Max Y error: 1.4716, Avg X error: -0.0372, Avg Y error: -0.0386, X tilt: -468.17, Y tilt: -38.42, and FOV: 300.91 mm X 237.38 mm. At the bottom are 'OK' and 'Cancel' buttons.

Previous values	
X mm per pixel:	
Y mm per pixel:	
Max X error:	
Max Y error:	
Avg X error:	
Avg Y error:	
X tilt:	
Y tilt:	
FOV:	

New values	
X mm per pixel:	0.1881
Y mm per pixel:	0.1978
Max X error:	0.9331
Max Y error:	1.4716
Avg X error:	-0.0372
Avg Y error:	-0.0386
X tilt:	-468.17
Y tilt:	-38.42
FOV:	300.91 mm X 237.38 mm

Figura 77 –Tela de conclusão da calibração do sistema bem sucedida. Fonte: autoria própria.

8 Separação de peças e tomada de decisão por imagem

Com a calibração realizada na seção 7 desta apostila, podemos agora realizar a programação do *software* para realizar a separação de peças e movimentação dependendo das suas características. As sequências que foram criadas servirão para que o programa principal às execute, colete os resultados, e passe as informações para o robô. Agora vamos conhecer um pouco mais sobre o código fonte em SPEL+ e as funções de interação entre o sistema de visão Vrun e Vget.

8.1. Comando Vrun

Uma vez que uma sequência foi criada no ambiente de desenvolvimento do Vision Guide, ela pode ser testada e executada a partir do ambiente de desenvolvimento ou de um programa SPEL+.

O comando **VRun** inicia a execução das sequências de visão. Quando **VRun** é iniciado, a sequência de visão especificada começa a execução. Isso significa que uma imagem é adquirida no buffer de quadros e, em seguida, as ferramentas de visão são aplicadas a essa imagem conforme definido na sequência de visão.

Uma vez que **VRun** é executado, **VGet** é normalmente usado para obter os resultados da sequência de visão, como dados de posição da peça, status da peça boa e ruim, informações de contagem de peças ou muitos outros resultados.

8.2. Comando VGet

VGet é uma parte poderosa da estrutura do Vision Guide. Ele fornece o mecanismo principal para obter os valores de propriedade e resultado das ferramentas de visão que são executadas a partir de sequências de visão.

O uso mais comum para **VGet** é obter os valores de resultado das ferramentas de visão depois que elas foram executadas em uma sequência. Isso

permite que você use os resultados para tomar decisões, realizar cálculos, definir posições de pontos e uma série de outras coisas. Para usar VGet com resultados, você deve primeiro executar um comando **VRun** para o qual deseja obter um resultado. Por exemplo, suponha que você criou uma sequência de visão que usa um objeto Blob para descobrir quantos furos estão presentes em uma peça específica. Isso significa que você desejará obter o valor do resultado NumberFound para o objeto Blob. O seguinte programa SPEL + mostra como VGet seria usado neste caso:

```
Function main
  Integer count
  VRun HoleCnt
  VGet HoleCnt.Holes.NumberFound, count
  Print "Buracos encontrados: ", count, "buracos!"
Fend
```

8.3. Aplicação do programa criado

No caso da calibração criada na seção 7, podemos criar um programa para realizar a identificação das peças circulares, e em seguida realizar a pega dos objetos com a ferramenta do robô. Sendo assim, o código ficaria da seguinte forma:

```
Function main
  Real x_cam, y_cam, u_cam
  Boolean found

  VRun Peca_Softex
  VGet Peca_Softex.Geom01.RobotXYU, found, x_cam, y_cam, u_cam

  Go Peca_pos :X(x_cam) :Y(y_cam) :U(u_cam) +Z(20)

  Move Here -Z(20)
  On Vacuo_ON
  Go Place_peca

Fend
```

O programa mostrado acima, realiza o comando VRun para executar a rotina “Peca_Softex” que identificará as geometrias que foram cadastradas.

Após isso, com o comando VGet, pode-se recuperar o valores obtidos no processo de identificação. Nela, é especificado então o nome da sequência, o nome da ferramenta, e o atributo que se busca. No caso aqui, como estamos em busca das coordenadas X,Y,Z e U do objeto, podemos usar o comando “RobotXYU”, e especificar as variáveis de destino.

Agora que os valores estão armazenados nessas variáveis, podem-se usar os comandos clássicos de movimentação para deslocar o robô para as coordenadas específicas.

8.4 Exercício 4 – Programação usando sistema de visão

Orientação: formem grupos de até 3 pessoas para a realização do exercício abaixo.

Com base no exemplo mostrado na subseção 8.1, realize a identificação de uma peça com a câmera integrada no robô EPSON, e realize a pega e paletização dessa peça em laboratório.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, podendo inserir na plataforma quantos arquivos quiser ou pode-se enviar arquivos inteiros em qualquer formato contanto que estejam compactados em .rar ou .zip, inserindo todos os arquivos de uma vez só. **As evidências são:**

Código fonte usado

Tela Vision Guide contendo a sequência e a calibração

9 Integração com softwares terceiros

Muitas vezes, é desejável realizar a aquisição e processamento de imagem em um sistema terceiro, em Python, LabView, etc. Com isso, podemos utilizar conexões com esses softwares por meio de comunicação TCP/IP, e a criação de uma mensageria entre o robô e o sistema de inspeção externo.

No caso do exemplo abaixo, o robô abre a porta 201 em TCP/IP como servidor, e fica aguardando a conexão do cliente TCP/IP ao sistema.

O comando “Input” então fica aguardando a chegada de uma mensagem do cliente, e ao recebe-la, continua a execução e executa a chamada da função “Read_Lbl_Coord” passando o texto recebido para o processamento.

Nesse caso, a mensagem enviada pelo software terceiro era do tipo “Coord;CoordY;CoordX” e a função ParseStr consegue realizar a tratativa para separar as coordenadas em suas devidas variáveis.

```
Function Data_Receiver

    OpenNet #201 As Server
    Print "Aguardando conexao do cliente TCP"
    WaitNet #201, 5
    Print "cliente TCP conectado"

    Do
        String action$
        Input #201, action$

        Print "Comando recebido: ", action$

        Call Read_Lbl_Coord(action$)

        Print #201, "OK"
    Loop
        Print "Fechando conexão #201"
        CloseNet #201
Fend

Function Read_Flm_Coord(action$ As String)
    Integer Pos
    String toks$(0)
    Pos = -1

    Pos = InStr(action$, "Coord")
    If Pos > 0 Then
        ParseStr action$, toks$, ";"
        CoordY = Val(toks$(1))
        CoordX = Val(toks$(2))
    EndIf
Fend
```

Para que essa comunicação se faça possível, deve-se também configurar o IP e protocolo que serão utilizados na porta TCP previamente.

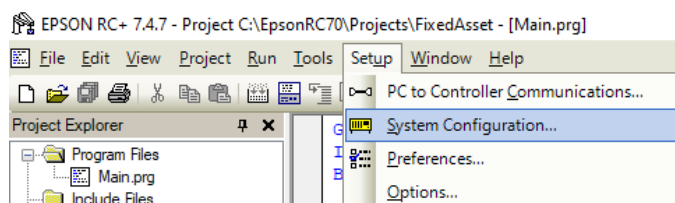


Figura 78 –Botão System configuration na interface. Fonte: autoria própria.

Ao acessar a tela System Configuration, pode-se navegar para a seção Controller -> TCP/IP e realizar a configuração da porta desejada.

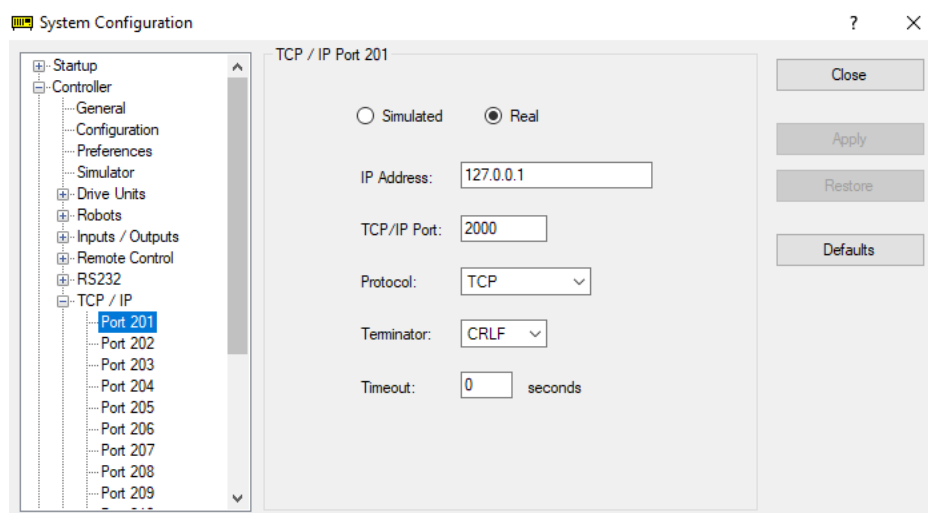


Figura 79 –Configuração da porta TCP/IP no sistema. Fonte: autoria própria.

Conclusão

A abordagem desta apostila permite ao leitor uma interação ativa com os conceitos de robótica em automações industriais, interfaces de programação robótica, e a integração de sistemas de visão ao processo de programação e criação de soluções automatizadas. A abordagem do material foi criada para estimular ao aluno autonomia, estímulo, senso crítico e contribuir para uma aprendizagem mais efetiva.

Assim, o conteúdo do curso de Robótica Industrial familiariza o leitor com o mundo tecnológico, onde a virtualização ganha, cada vez mais, espaço e agrada as empresas por reduzir custos e aumentar as margens de lucro. Obrigada por fazer parte do curso e ter realizado a leitura desta apostila. Espero que o interesse pelo assunto de robótica apresentado neste curso esteja aguçado, para que você pratique e conheça ainda mais as maravilhas da robótica e automação.

Referências

Manipulator manual C4 series Rev.17. Disponível em: <
https://epson.com.cn/robots/admin/uploads/product_catalog/files/EPSON_C4_r4.pdf > Acesso em: agosto de 2021.

EPSON RC+ 7.0 Option Vision Guide 7.0 (Ver.7.4) Properties and Results Reference Rev.7. Disponível em: <
[https://files.support.epson.com/far/docs/epson_vision_guide_70_manual-rc700_rc90\(v71r2\).pdf](https://files.support.epson.com/far/docs/epson_vision_guide_70_manual-rc700_rc90(v71r2).pdf) > Acesso em: setembro de 2021.

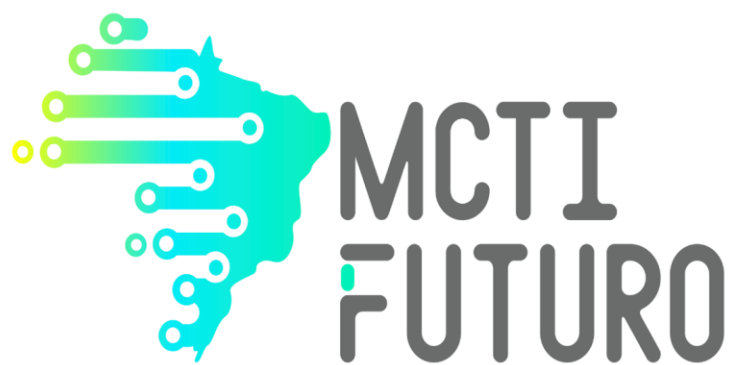
EPSON RC+ 7.0 Option Vision Guide 7.0 (Ver.7.4) Hardware & Setup Rev.7. Disponível em: <
[https://files.support.epson.com/far/docs/epson_vision_guide_hardware-cv1_cv2_pv1\(v73r3\).pdf](https://files.support.epson.com/far/docs/epson_vision_guide_hardware-cv1_cv2_pv1(v73r3).pdf) > Acesso em: setembro de 2021.

VERTULO, Rodrigo Cesar. **Robôs de Classe**. Disponível em: <
<http://labdeeletronica.com.br/robos-de-classe/>>. Acessado em: 27/10/2021.

Reverse Engineering, Redesign and Topology Optimization for Additive Manufacturing of an Industrial Robot Arm Link. Disponível em:
https://www.researchgate.net/figure/Manipulators-and-their-work-envelopes-a-Cartesian-manipulator-b-gantry-c_fig1_322210953. Acessado em: 27/10/2021

2. CONTROLE DE REVISÃO DO DOCUMENTO / DOCUMENT REVISION CONTROL

Revisão	Descrição	Razão	Autor	Data
Review	Description	Reason	Author	Date
A	-	Revisão inicial	Larissa Alves	28/10/21
B	Alteração do nome do curso para melhor adequação a ementa e inclusão de uma página de páginas.	Processo de melhoria	Larissa Alves	01/12/21



FUTURO DO TRABALHO, TRABALHO DO FUTURO

Bom curso