ESTUDO SOBRE O PROCESSO *CONVEYOR TRACKING* E SUA IMPLEMENTAÇÃO ATRAVÉS DE RECURSOS NATIVOS E NÃO PROPRIETÁRIO

STUDY ON THE CONVEYOR TRACKING PROCESS AND ITS IMPLEMENTATION THROUGH NATIVE AND NON-OWNERS RESOURCES

Yago Kauan Rozão Cypriano[1]
João Hermes Clerici[2]
yago.kauan@mail.usf.edu.br

[1]Aluno do Curso de Engenharia Elétrica, Universidade São Francisco; Campus Itatiba
[2]Professor Orientador, Curso de Engenharia Elétrica, Universidade São Francisco; Campus Itatiba.

RESUMO. O objetivo deste documento é apresentar o funcionamento do processo *conveyor tracking* utilizado em robôs industriais através de pacotes adicionais, e em geral opcionais, oferecidos pelos fabricantes destes robôs. Em seguida, é abordado uma metodologia para implementação deste mesmo processo através da utilização dos recursos nativos de um robô, permitindo a execução de aplicações de rastreamento de transportadoras sem a necessidade de aquisição de um *software* proprietário.

Palavras-chave: robôs, *conveyor tracking*, transportadores industriais, seguimento de esteira.

ABSTRACT. *The purpose of this document is to present the operation of the conveyor tracking process used in industrial robots through additional packages, and generally optional, offered by the manufacturers of these robots. Then, a methodology for implementing this same process through the use of native resources of a robot is discussed, allowing the execution of conveyor tracking applications without the need to purchase proprietary software.*

INTRODUÇÃO

Quando o assunto é robô industrial, imediatamente se associa a interação destes com os produtos fabricados nas linhas de produção ou até mesmo com os processos de fabricação. Portanto, é muito comum nas indústrias, que robôs e esteiras transportadoras se relacionem de forma a operar em sincronismo, isto é, o robô manipulando ou inspecionando peças sem a parada do meio que as transportam, seja uma esteira, corrente, carrossel ou até mesmo mesas giratórias.

Inicialmente, até os anos 1970, interações entre robôs e transportadores industriais eram feitas através da parada parcial ou total da linha para o robô executar suas tarefas. Com a necessidade de atingir grandes volumes de produção e evitar essas paradas de linha, e obter sincronismo com correias transportadoras com precisão e velocidade, se fez necessário uma solução de controle com complexidade bastante elevada para o rastreamento de correias transportadoras, comumente chamado de *Conveyor Tracking*.

O processo *Conveyor Tracking* foi desenvolvido pelos fabricantes de robôs e manipuladores, de forma que fosse possível integrar esse controle de rastreamento de correias de forma intuitiva e fácil, através de módulos dos próprios fabricantes, normalmente vendido como um pacote opcional.

Entretanto, para algumas aplicações, nas quais não é necessário uma solução completa com os processos *Conveyor Tracking* disponibilizados como pacotes opcionais dos fabricantes de robô, pode ser possível (sem parada da linha) realizar o rastreamento de transportadores com o robô através de programação convencional dentro da unidade de comando do robô, juntamente com informações externas dos transportadores, como velocidade, função de movimento (linear ou rotativo), posição do produto ou peça a ser manipulada, etc.

Neste projeto, foram estudados os fundamentos do processo *Conveyor Tracking,* e com base nesses estudos, implementou-se processos de sincronização das operações de robôs com o movimento de transportadores industriais de forma simplificada e específica para cada necessidade, tornando o sistema mais fácil, customizado e sem custos adicionais em relação aos módulos opcionais fornecidos pelos fabricantes.

REFERENCIAL TEÓRICO

"*Conveyor tracking* requer que um robô realize uma determinada tarefa enquanto a peça de trabalho está sendo movida continuamente por um transportador" (WILHELM, 1988, p. 283).

Em outras palavras, é necessário que haja um sincronismo entre o robô e o mecanismo responsável por transportar o objeto de trabalho para a realização da tarefa desejada de forma contínua.

Segundo Wilhelm (1988), robôs que realizam esse processo de rastreamento de transportador exigem um *hardware e software* diferenciados, se comparado com robôs comuns que realizam operações em estações estáticas. (WILHELM, 1988).

Com *hardware* diferenciado, entende-se que a unidade de controle do robô deverá possuir entradas para ligação de *encoders* ou portas de comunicação que receba informações (velocidade, posicionamento, sentido de movimento, etc) do mecanismo que se deseja sincronizar. Já para *software*, implica em um desenvolvimento computacional de um programa ou algoritmo que realizará a interpretação dessas informações obtidas via *hardware* e processá-las de forma a resultar no trabalho desejado de sincronização.

O ramo automobilístico é o setor que mais se destaca no uso de *conveyor tracking,* por se tratar de grandes linhas de montagem (tanto em comprimento como em tamanho do produto). Processos de soldagem, pintura, manuseio de materiais e montagem são algumas das aplicações que utilizam robôs para execução de tarefas, dispensando parcialmente a mão de obra humana, que está sujeita a erros e imprecisões no processo.

Para o desenvolvimento de uma aplicação como essa, PARK e LEE (1992, p. 379) propuseram em seu artigo sobre análise de movimento de robôs e planejamento para *conveyor tracking*, as seguintes características:

1) O sistema do transportador se movimenta a uma velocidade constante, e a peça é estática em relação ao transportador.
2) A orientação da garra robótica é alinhada com a peça inicialmente e mantida fixa durante o seguimento da peça pelo robô.
3) O robô rastreia a peça ao longo de um caminho em linha reta sob o transportador.

A primeira característica proposta por PARK e LEE serve para descrever de forma matemática a equação que traz a posição vetorial da peça no sistema em função do vetor de velocidade da esteira, como mostrado na equação (1):

$$x_p(t) = v_b t - v_b t_o + x_p(t_o), \quad t >= t_o \tag{1}$$

$$x_p(t) = v_b, \quad t >= t_o \tag{2}$$

onde v_b denota o vetor de velocidade da esteira, e $x_p(t)$ denota o vetor de posição da peça em relação ao momento inicial no instante t.

A segunda característica sugere que inicialmente a garra do robô não está alinhada ou sincronizada com a peça, e que o deverá ser feito para só então ser possível que o trabalho requerido pelo robô, de forma contínua, seja realizado.

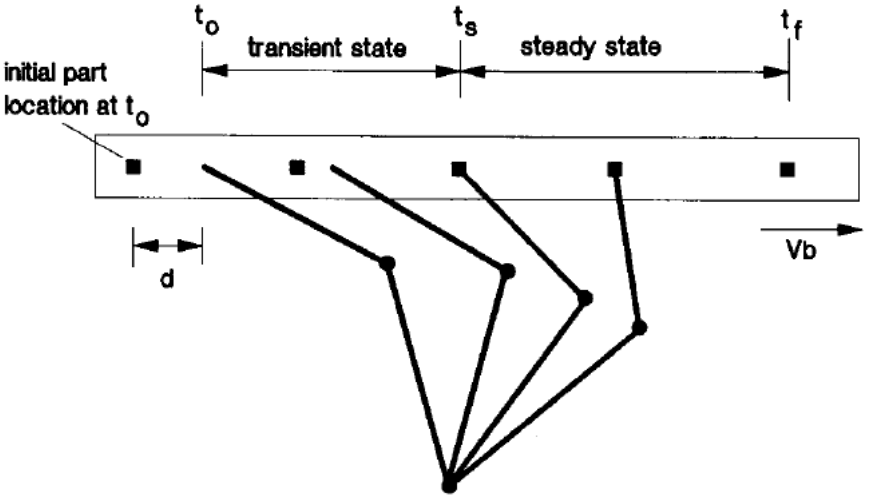
Portanto, a resultante da equação (2) é utilizada, juntamente com a informação de posição da garra, para realizar esse alinhamento, dada pela equação (3):

$$x_h(t) = x_p(t_o + d) \tag{3}$$

em que $x_h(t)$ denota a posição vetorial da garra do robô em relação ao momento inicial no instante t, e d denota o vetor de desvio entre a posição inicial da peça e da garra robótica.

Esse processo é realizado no início do movimento, para que exista espaço mínimo disponível para a realização das operações desejadas pelo robô. Essa área foi denominada como estado transiente (*transient state)*, também conhecida como zona de sincronização, e é possível observar na figura 1:

Figura 1 - Formulação de um problema para *conveyor tracking*



Fonte - PARK e LEE: An Approach to Robot Motion Analysis and Planning for Conveyor Tracking

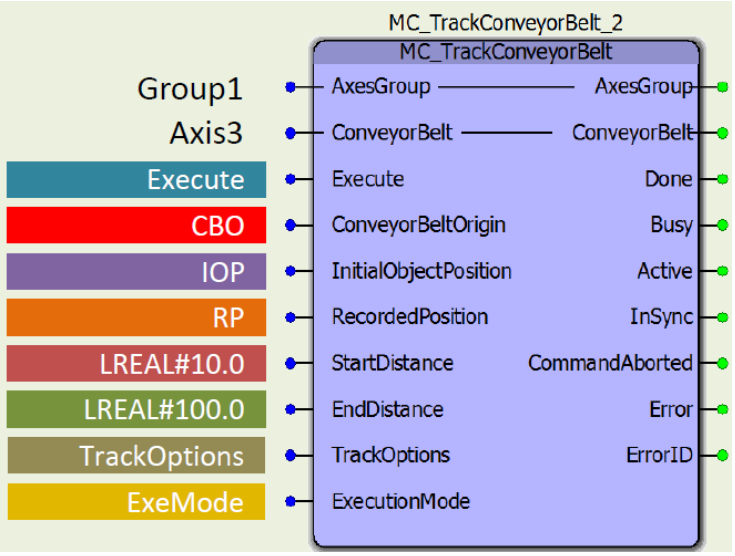
Por fim, a terceira característica nada mais é do que a permanência do sincronismo alcançado pelo sistema até o final do transportador ou até a conclusão da operação realizada pelo robô.

Um problema dessas proposições, mais especificamente, da primeira, onde assume-se que a velocidade do transportador é constante, é justamente em aplicações em que a velocidade do transportador não é constante, seja devido a particularidades do processo ou variações externas que influenciam no movimento do transportador.

Como citado anteriormente, as aplicações que utilizam *conveyor tracking* vem crescendo e evoluindo cada vez mais. Com isso, os próprios fabricantes de robôs desenvolveram seus próprios módulos, que são integrados na unidade de controle em alguns modelos de robôs, possibilitando a programação e parametrização do robô para realizar o rastreamento de transportadores, levando em consideração, a posição do transportador através de *encoders*, e não da velocidade.

Os robôs da *Yaskawa, por* exemplo, fornecem como módulos adicionais para aplicações de *conveyor tracking,* o chamado *MC_TrackConveyorBelt*, que agrupam os cálculos necessários em um bloco lógico com parâmetros de entrada e saída, como visto na figura 2.

Figura 2 - Bloco de controle *conveyor track* da Yaskawa

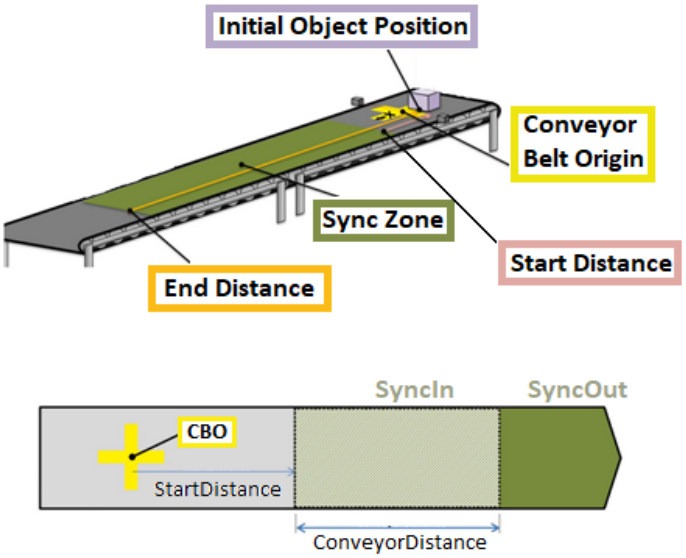


Fonte - Referência rápida MC_TrackConveyorBelt (TCP_QRg_v6)

Nessa solução proposta pela *Yaskawa*, é possível observar que os dois primeiros parâmetros de entrada recebem a posição do eixo do transportador, através de um *encoder*. Também é possível observar a similaridade com a segunda característica proposta por PARK e LEE, na qual é necessário realizar o alinhamento da garra robótica com a peça na região inicial do transportador (estado transiente), descrito aqui como *InitialObjectPosition* e *StartDistance*, que são as informações necessárias para realizar esse sincronismo.

Na figura 3, é representado essas zonas de sincronização:

Figura 3 – Zonas de Sincronização em *Conveyor Tracking*

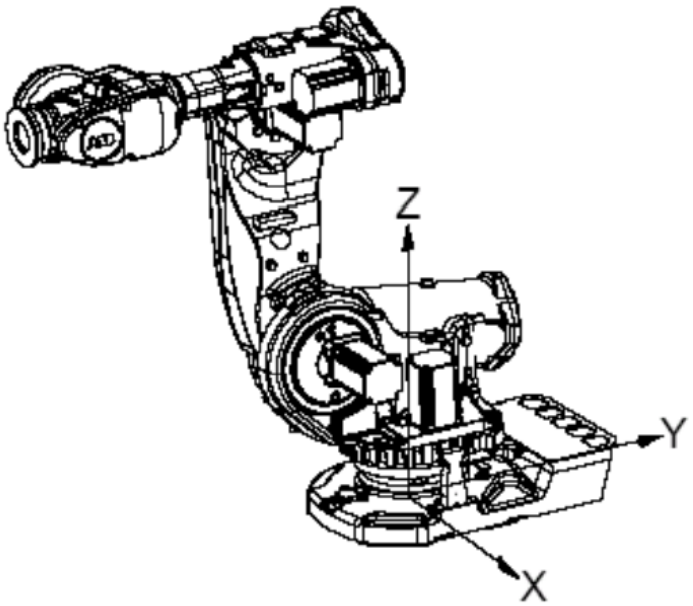


Fonte – *Webinar: Intro to Conveyor Tracking*

Com base nessas informações, para desenvolver um meio alternativo de rastrear uma peça em movimento (sem necessidade de adquirir os módulos adicionais dos fabricantes de robôs), cujo é a proposta deste trabalho, alguns parâmetros não poderão ser descartados, conhecendo sua importância para o processo de *conveyor tracking.*

Um desses parâmetros, é a identificação dos sistemas de coordenadas do robô que deverá ser manipulado em sincronia com a movimentação da peça no transportador. O *Tool Center Point*, chamado normalmente de *tool0* pelos fabricantes de robôs, é o ponto que se moverá ao longo do sistema de coordenadas base do robô, como mostra a figura 4. Esse ponto se localiza no centro da flange do robô.

Figura 4 - Coordenadas de um robô articulado



Fonte - IRC5 Programação Básica

A complexidade do cálculo de informação de coordenada realizada pela unidade de controle do robô dos fabricantes, como mostrado no bloco computacional da *Yaskawa* para que seja possível realizar o seguimento da peça, deverá ser transferido ao *PLC* da linha, que fará através da leitura de *encoder* presente no transportador e informações previamente definidas para obter a relação do movimento do motor do transportador com a ferramenta (*tool0)* do robô que fará o seguimento.

Em um artigo recente da *Control Automation,* o autor Muhammad explica o funcionamento de um *encoder,* e como esse dispositivo é necessário para desenvolver a aplicação desejada neste trabalho, bem como já é utilizado nas soluções propostas por outros pesquisadores e fabricantes de robôs.

"O *encoder* fornece saída toda vez que o conjunto se move em uma direção circular. A saída é na forma de sinais elétricos. Dependendo do tipo de *encoder*, a saída pode ser uma série de trens de pulso ou um valor fixo que pode ser usado para qualquer tipo de tarefa." (Muhammad, 2020).

O tipo de *encoder* necessário para a aplicação é do tipo incremental, devido a simplicidade que oferece e por ser suficientemente satisfatório para o projeto, uma vez que o processo não depende da posição absoluta do transportador para seguimento da peça, apenas da posição a partir de um determinado instante (acionamento de sensores de detecção de peças).

METODOLOGIA

Por trás de qualquer elemento transportador de peças, seja uma esteira, carrossel, mesa giratória ou qualquer outro, existe um motor que fará o transporte ocorrer. Portanto, é possível identificar a posição de qualquer elemento presente no transportador, desde que seja possível realizar a leitura desse motor, seja através de um tacômetro, para ler a velocidade de rotação do motor, ou um *encoder*. Obtendo essa informação, atrelada a alguns cálculos que se fazem necessários em casos de acoplamentos, redutores e sistemas de transmissão, que afetem na coordenada final da peça em relação ao motor, um valor que identifica a posição da peça é obtido.

Utilizando essa informação, que varia constantemente de acordo com o movimento do transportador, é possível comandar um robô para rastrear a peça.

O método adotado para realização deste trabalho consiste nos seguintes procedimentos:

1. Ambiente de simulação
   - 1.1. Demonstração dos *softwares* de simulação utilizados
   - 1.2. Criação dos cenários e organização dos periféricos
2. Programação CLP
   - *2.1. Software e Hardware*
   - 2.2. Leitura de *encoder*
   - 2.3. Realização de cálculos para conversão de pulsos do *encoder* em posição
   - 2.4. Interface IHM
3. Programação Robô
   - 3.1. Elaboração do programa principal do robô de coleta de peça
   - 3.2. Elaboração de programa em *background* para atualização das coordenadas de base do robô durante movimentação do programa principal
4. Comunicação Industrial
   - 4.1. Estabelecimento de comunicação entre robô e PLC para leitura de posição do objeto de trabalho

Ambiente de Simulação

Com a evolução da tecnologia e ferramentas de desenvolvimento de projetos, atualmente é possível contar com *softwares* de simulação que auxiliam nos testes e desenvolvimento de aplicações antes de serem fabricados, dessa forma, prevendo materiais necessários para o projeto, redução de custos e maior assertividade e precisão nos resultados finais.

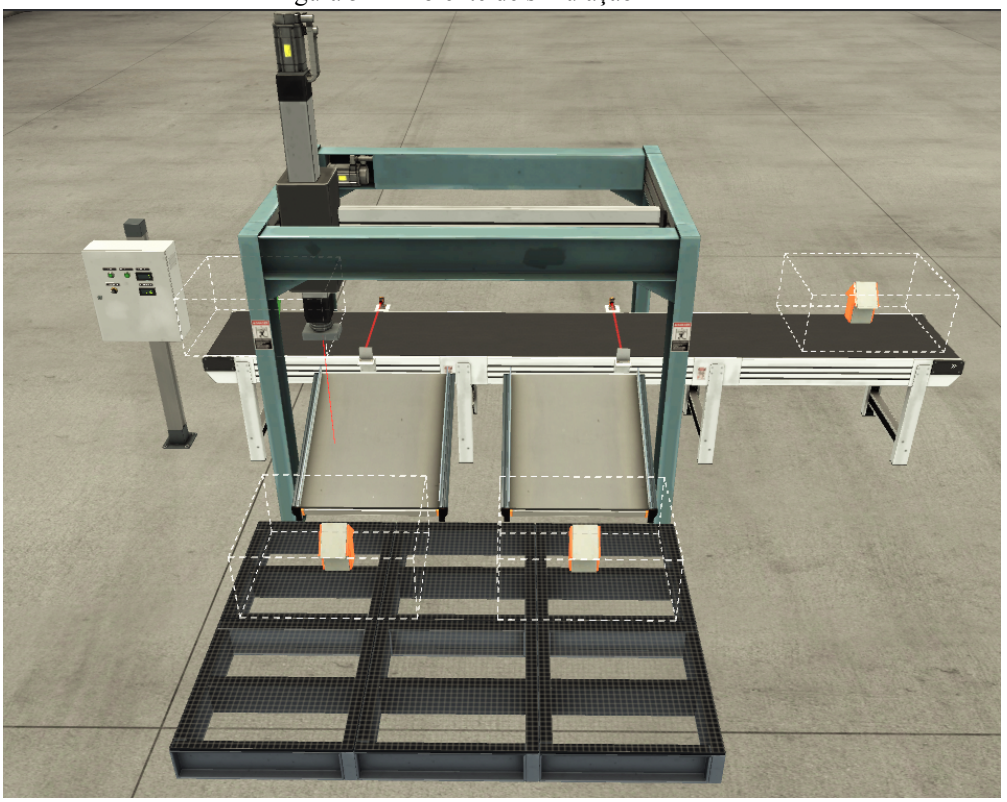
Para iniciar o desenvolvimento o qual esse trabalho visa entregar, o *software* de simulação *Factory IO* foi utilizado, juntamente com o *software* do *PLC* escolhido (*TIA Portal* da Siemens). Dessa forma, foi possível desenvolver o programa que realizou toda a análise, processamento e controle das informações dos periféricos envolvidos e testar seu funcionamento dentro do ambiente de simulação do *Factory IO*, que tem capacidade de enviar e receber os sinais físicos e comandar esses periféricos.

Inicialmente, os equipamentos necessários para o desenvolvimento do projeto são um PLC (Controlador Lógico Programável), responsável por toda a complexidade e processamento de informações, uma esteira com lona para o transporte de peças, um *encoder* para a leitura de posição do motor da esteira, com saída física, dois sensores retroreflexivos

para detecção de passagem da peça no início e fim da esteira, e um robô articulado ou *gantry* (robô de coordenadas cartesianas).

O cenário criado dentro do ambiente de simulação é apresentado na figura 5:

Figura 5 - Ambiente de simulação



Fonte - Autor Próprio

Ao detectar a passagem de uma peça pelo sensor de entrada, o robô irá iniciar o movimento para sincronizar a posição da garra com a peça. Após sincronizado, o robô fará a coleta da peça através de vácuo e irá paletizar em uma determinada região.

Periféricos utilizados

A aplicação inicial consiste em uma esteira de 6 metros com os sensores de detecção de peças e um robô de coordenadas cartesianas que ficará posicionado acima da esteira. Um PLC Siemens da série S7-1200 será utilizado para realizar a leitura do *encoder* acoplado ao eixo de uma esteira.

Como descrito anteriormente, para que o robô possa realizar o rastreamento de uma peça de trabalho em movimento sob uma esteira, é necessário conhecer a posição durante todo o período em que se deseja manter o robô em sincronia com o produto. Uma forma de obter essa informação é com a utilização de *encoders.* Estes dispositivos são capazes de transformar um valor de posição em sinal elétrico digital.

A escolha do *encoder* para esse projeto possui resolução suficiente para medir o deslocamento da peça na esteira a cada 1 milímetro. Para isso, foi necessário calcular o perímetro do eixo da esteira, obtendo o valor de deslocamento dessa esteira a cada revolução do eixo.

O modelo de *encoder* incremental utilizado nesse projeto é apresentado na figura 6:

Figura 6 - *Encoder* Incremental de 600 pulsos

Fonte - Mercado Livre - https://produto.mercadolivre.com.br/MLB-1974868779-encoder-incremental-pnp-600-pulsos-5-a-26-volts-_JM?quantity=1

Para a realização de leitura do posicionamento da esteira, foi utilizado as fases disponibilizadas pelo *encoder* nas entradas rápidas do PLC, sendo que no caso de um S7-1200 1211C DC/DC/DC, são 3 contadores rápidos, nas entradas I0.0/I0.1, I0.2/I0.3 e I0.4/I0.5, conforme tabela 1 (quadrante verde):

Tabela 1 - Entradas para contadores rápidos (HSC)

	Inputs (Counting / Frequency)																		Outputs (Axis of motion)							
	CPU														Signal boards								all CPUs with DC outputs			
												CPU 1214C			4DI 200kHz				4DO 200kHz							
						CPU 1212C											2DI / 2DO (200 kHz)									
				CPU 1211C																						
I/Os	I0.0	I0.1	I0.2	I0.3	I0.4	I0.5	I0.6	I0.7	I1.0	I1.1	I1.2	I1.3	I1.4	I1.5	I4.3	I4.2	I4.1	I4.0	Q4.0	Q4.1	Q4.2	Q4.3	Q0.0	Q0.1	Q0.2	Q0.3
HSC_1	1	2		[R]											[R]		2	1	CLK	DIR			CLK	DIR		
HSC_2		[R]	1	2											2	1	[R]				CLK	DIR			CLK	DIR
HSC_3					1	2		[R]																		
HSC_4						[R]	1	2																		
HSC_5									1	2	[R]				[R]		2	1								
HSC_6												1	2	[R]	2	1	[R]									
																			PTO1		PTO2		PTO1		PTO2	

Rules:
1. For every HSC, only one in/out area can be choosen (CPU inputs, SB inputs, SB outputs, or CPU DC outputs).
2. Every input can be used with only one HSC.

	Max. frequency [kHz]	CPU DI/DO	Signal boards 2DI/2DO	Signal boards 200 kHz 2DI/2DO	Signal boards 200 kHz 4DI	Signal boards 200 kHz 4DO
HSC_1	SP	100/100	30/20	200/100	200	100
	MP	80/-	20/-	160/-	160	
HSC_2	SP	100/100			200	100
	MP	80/-			160	
HSC_3	SP	100/-				
	MP	80/-				
HSC_4	SP	30/-				
	MP	20/-				
HSC_5	SP	30/-	30/-	200/-	200	
	MP	20/-	20/-	160/-	160	
HSC_6	SP	30/-			200	
	MP	20/-			160	

Legend	Operating phase: Single phase	Two phase	AB Quadrature
1	CLK	CLK UP	CLK A
2	[DIR]	CLK DN	CLK B
[R]	optional external reset input (only for "counting")		
CLK	clock input		
DIR	direction input (for "axis of motion")		
[DIR]	optional external direction input (for "single phase")		
CLK UP	clock up input (for "two phase")		
CLK DN	clock down input (for "two phase")		
CLK A	clock A input (for "AB quadrature")		
CLK B	clock B input (for "AB quadrature")		
SP	Single phase		
MP	Multi-phase (Two phase / AB Quadrature)		
	possible with all CPUs		
	possible with CPU 1212C / CPU 1214C		
	possible only with CPU 1214C		
	possible with SBs 1223 / SB 1221 DC 200kHz 4xDI		
	possible with SBs 1223 / SB 1222 DC 200kHz 4xDO		
	only possible with SB 1221 DC 200kHz 4xDI		
	only possible with SB 1222 DC 200kHz 4xDO		

Fonte - SIMATIC S7-1200 HSCs p. 10

O controle da esteira foi realizado de forma independente, através do próprio controlador da esteira, onde é possível variar a velocidade do movimento, o que não deve

influenciar na leitura de posicionamento pelo *encoder* desde que essa velocidade não ultrapasse à frequência máxima dos sinais do *encoder*, que é calculado pelo produto da sua resolução por quantidade de revolução por segundo do eixo.

Após realizar o acoplamento do *encoder* ao eixo do motor da esteira, estabelecer as ligações entre os periféricos e configurar os HSC (*High Speed Counter*) do PLC, foi necessário desenvolver em sua programação, blocos de cálculos que irão devolver na saída, um valor de coordenada, que refere-se a posição da peça na esteira, cujo o robô deverá perseguir. Esse cálculo é realizado após o acionamento do sinal de entrada de peça, localizado no início da esteira.

Programação CLP

Um dos motivos para incluir o desenvolvimento da programação em CLP é justamente transferir a complexidade de cálculos realizados pelos métodos proprietários de robôs para uma unidade de controle (CLP) cuja presença e disponibilidade costuma ser comum em linhas produtivas, além de possuírem maiores flexibilidades de programação.

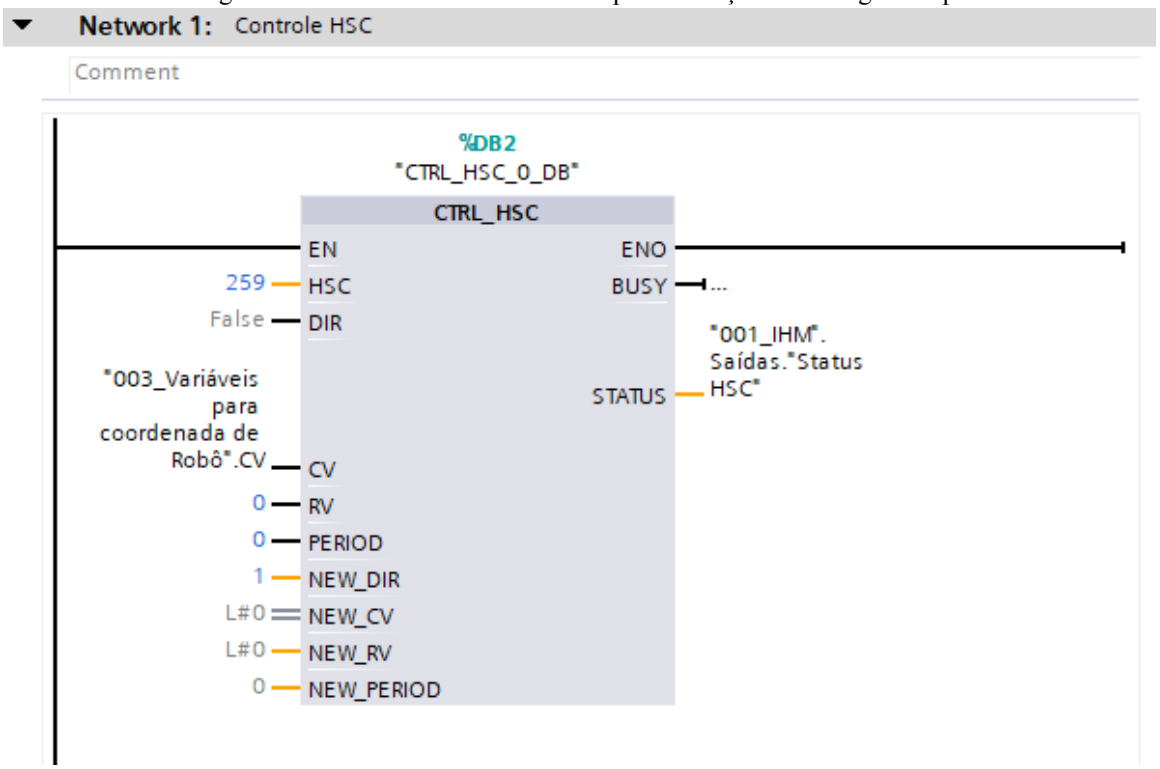
O *software* descrito abaixo foi desenvolvido e descarregado em um CLP S7-1200 1211C AC/DC/Rly. A utilização de um hardware real (CLP e *Encoder*) em conjunto com um *software* de simulação (*Factory IO*) se fez necessária devido às limitações do *software* em questão para processar as informações do *encoder* com precisão.

A estrutura de programação do CLP consiste nos seguintes passos:

1. Realizar a leitura de pulsos do *encoder* (após detecção da peça), registrando-o em uma variável do tipo inteiro.
2. Calcular o perímetro da esteira
3. Calcular a relação entre 1 pulso do *encoder* e o deslocamento da esteira
4. Multiplicar a resolução encontrada com a quantidade pulsos registrados desde a detecção da peça

O bloco de programação utilizado para realizar a leitura dos sinais do *encoder* é apresentado na figura 7:

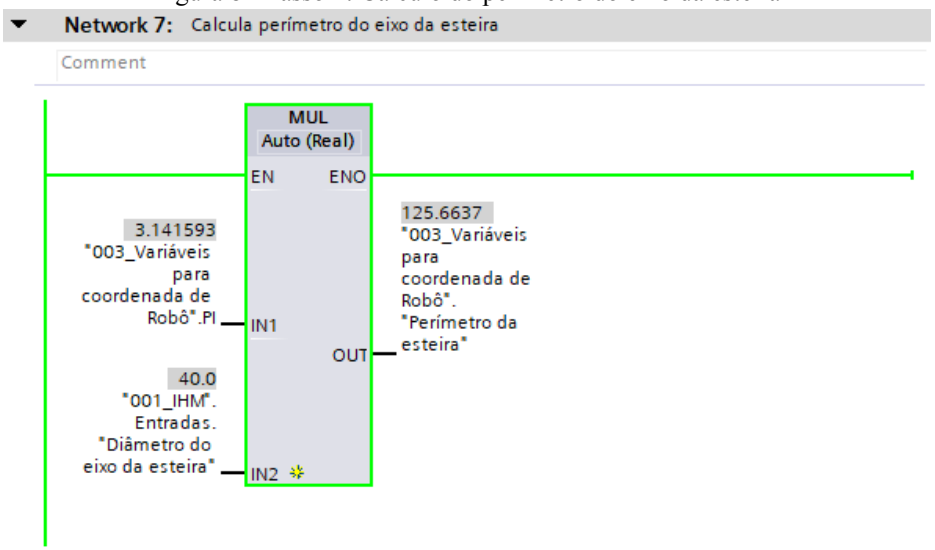
Figura 7 - Passo 1: Leitura do *encoder* para obtenção da contagem de pulsos



Fonte - Autor Próprio

Considerando uma esteira com eixo de 40 milímetros de diâmetro, o deslocamento linear dessa esteira a cada volta é de aproximadamente 125,66 milímetros ($2\pi r$). Portanto, a resolução do *encoder* necessária para medir 1 milímetro é de no mínimo 126 pulsos por revolução. Esse cálculo do perímetro da esteira é mostrado na figura 8:

Figura 8 - Passo 2: Cálculo do perímetro do eixo da esteira



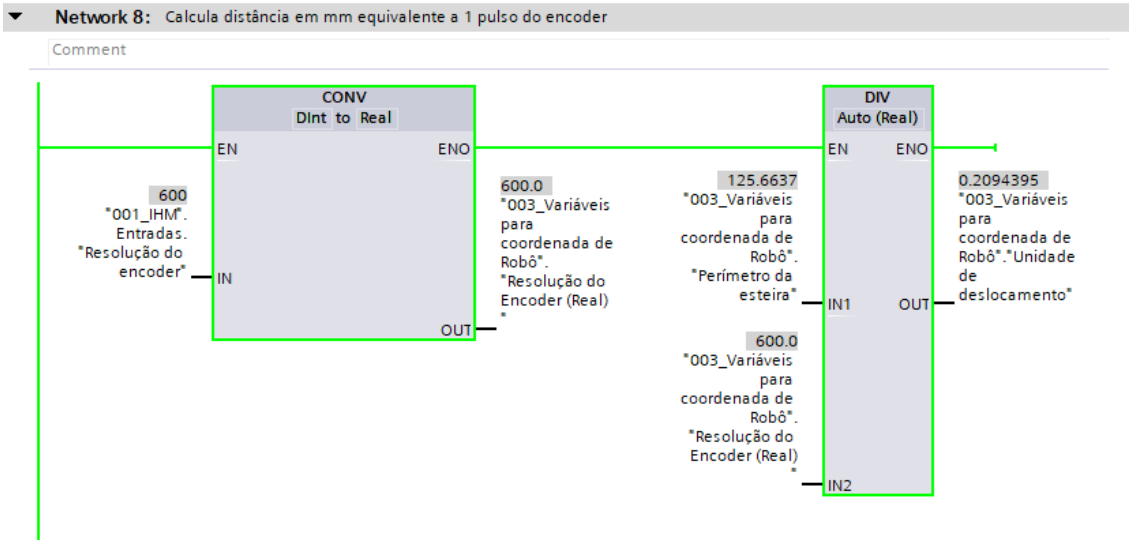
Fonte - Autor Próprio

Considerando então que as informações atuais das variáveis para o projeto sejam:

- Deslocamento da esteira por volta: 125,6637 mm
- Resolução do *encoder*: 600 pulsos

teria-se que cada pulso do *encoder* correspondesse a 0,2094 mm de deslocamento da peça na esteira. Portanto, para cada pulso contado pelo PLC, 0,2094 mm é acrescentado a uma variável do tipo REAL que corresponde a coordenada a ser enviada ao robô. O cálculo que relaciona o perímetro da esteira e a resolução do *encoder* é mostrado na figura 9:

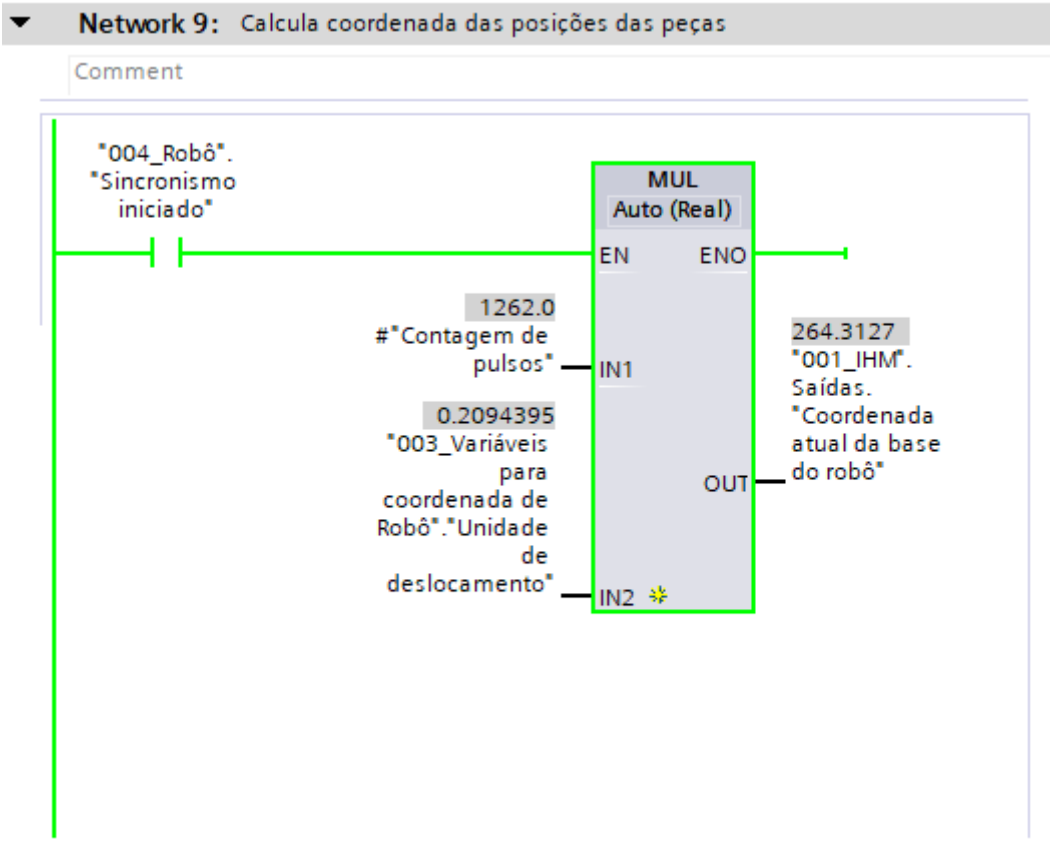
Figura 9 - Passo 3: Cálculo da relação entre *encoder* e deslocamento da esteira



Fonte - Autor Próprio

Com as informações de relação entre pulsos do *encoder* e deslocamento da esteira, basta realizar a multiplicação entre esses valores para obter a posição em mm do objeto de trabalho, que deverá ser informado ao robô para o rastreamento.

Figura 10 - Passo 4: Cálculo da relação entre *encoder* e deslocamento da esteira



Fonte - Autor Próprio

Programação Robô

O modelo de robô disponível para utilização pelo laboratório da Universidade São Francisco - USF é o ABB IRB 2400, mostrado na figura 11:

Figura 11 - Robô ABB 6 Eixos

Fonte - IRB 2400 *Industrial Robot* - http://georgesidra.edublogs.org/

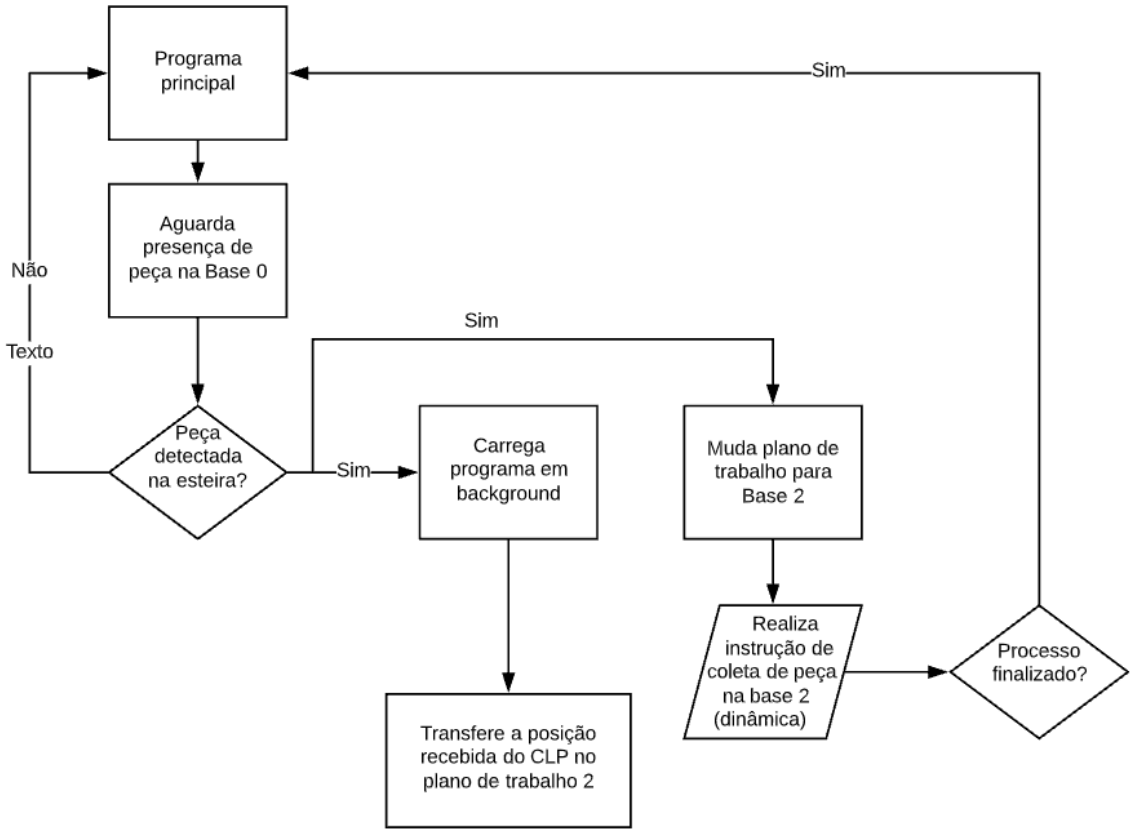
A data de manufatura desse robô é de 09/07/1999, o que implica em um modelo que conta apenas com interfaces de comunicação serial RS232 e RS485. A princípio, foi definido que seria utilizado o protocolo serial RS232 para estabelecer a comunicação entre o PLC Siemens S7-1200 e o robô. Entretanto, em testes posteriores, foi detectado uma incompatibilidade da versão do robô com a instrução de abertura de comunicação serial, onde só seria possível através de funções avançadas, não disponíveis no robô em questão. Portanto, foi considerado no desenvolvimento da metodologia, a interface Ethernet/IP.

Figura 12 - Plaqueta de identificação do controlador do robô

Fonte - Autor Próprio

A programação do robô é realizada de forma cíclica em que deverá ir para uma coordenada onde um dos eixos (o mesmo em relação ao sentido de movimento da esteira) será dinâmica, ou seja, atualizada constantemente pelo PLC, através de uma rede de comunicação serial ou *profinet*. A estrutura do programa do robô foi desenvolvida como no fluxograma apresentado na figura 13:

Figura 13 - Fluxograma de programação do robô



Fonte - Autor Próprio

Para isso, foi utilizado a função multitarefa, que permite a execução de vários programas em paralelo. A multitarefa é executada colocando os programas a serem executados em paralelo. A execução da operação multitarefa é iniciada ativando-a a partir do painel de operação ou por um sinal de entrada dedicado ou executando uma instrução relacionada à operação multitarefa.

O programa principal inicializa com uma instrução que define um plano de trabalho inicial (conhecido como *Base* 0 ou *WorkCoordinate 1*), onde as coordenadas do TCP (*Tool Center Point*) da ferramenta estão na origem.

Ao detectar a presença de peça na esteira, é carregado o programa de *background*, inserido em outro slot, para ser executado em paralelo com o programa principal. Após o carregamento concluído, é trocado para outro plano de trabalho (*Base 2*), onde as coordenadas do TCP da ferramenta estão constantemente sendo atualizadas em paralelo no programa de *background*.

No programa *background*, executando em *multitask*, o plano de trabalho 2 (*Base 2*) recebe inicialmente, e apenas uma única vez, as mesmas coordenadas do plano de trabalho 1 (*Base 0*, TCP na origem). Após isso, o eixo que possui o mesmo sentido da esteira de ponto é

atualizado em x milímetros a cada *scan*, correspondente a informação de posição da peça obtida pelo *encoder* e enviada pelo PLC, e logo em seguida, essa coordenada é transferida para o plano de trabalho 2, que corresponde a *Base 2*.

Quando uma peça é detectada na esteira, o PLC re-transmite esse sinal para o robô, que está lendo, de forma cíclica, o programa principal, e pode executar o programa que tem a instrução de movimento linear. Nesse movimento, as coordenadas são eixos de valor constante e eixos de valor dinâmico, sendo enviado pelo PLC de acordo com os cálculos descritos anteriormente. Essa instrução é executada em *loop*, com os valores de coordenadas constantemente atualizadas, até a finalização do processo.

Ao finalizar o processo de rastreamento da peça e ter concluído o trabalho desejado em sincronia com o transportador, o programa do robô sai da instrução em *loop* e executa um programa para descarte da peça, e retorna para home, dessa forma, estando pronto para repetir um novo ciclo.

Os valores de coordenadas são atualizadas do PLC para o robô à uma unidade de tempo máxima previamente estabelecida, para que seja possível que a capacidade mecânica do robô execute o movimento.

Considerando que o transportador se desloca 10 milímetros a cada 1 segundo, com resolução de 0,5 mm, 20 operações de incremento nesse período (1 segundo) seriam executadas, de 0,5 mm cada, sendo que cada uma dessas operações teria sido realizada em um intervalo de 50 ms.

Se o tempo de *scan* do robô for de 4 ms, significa que o mesmo irá executar a instrução de movimento linear na mesma coordenada (ou seja, ficará parado), 12 vezes, até que receba uma coordenada diferente e realize algum movimento.

Em uma situação mais crítica (maior velocidade da esteira), se o transportador deslocar 10 milímetros a cada 10 milissegundos, considerando a mesma resolução de 0,5 mm, as 20 operações de incremento seriam executadas agora em um intervalo de 0,5 ms.

Nessa condição, a cada *scan* de 4 ms do robô, seria movimentado 4 mm no eixo em que acompanha o transportador, o que reflete uma velocidade de 4 m/s, mecanicamente impossível de atingir.

Portanto, nos cálculos efetuados pelo PLC, foi considerado uma unidade de tempo máxima para ser atualizado os valores de coordenadas ao robô (*baud rate*), pois dessa forma, mesmo que a velocidade do transportador esteja muito alta, não será ultrapassado o limite de velocidade suportado pelo robô.

Comunicação Industrial

EtherNet/IP é uma rede de comunicação industrial com especificações abertas. As especificações são gerenciadas pela ODVA (*Open DeviceNet Vendor Association, Inc.*) O protocolo industrial foi combinado com a Ethernet e padronizado como Ethernet / IP (Protocolo Industrial).

A comunicação é realizada combinando os protocolos conhecidos Protocolo Industrial Comum (CIP), TCP / IP e Ethernet. Isso permite que a Ethernet seja usada em conjunto com a rede.

Antes de iniciar a comunicação EtherNet / IP, um dos dispositivos deve abrir uma comunicação chamada de "conexão" com o outro dispositivo. O lado que abre a conexão é chamado de "scanner" e o lado aberto é chamado de "adaptador". (No caso, o CLP Siemens é o scanner e o robô é um adaptador).

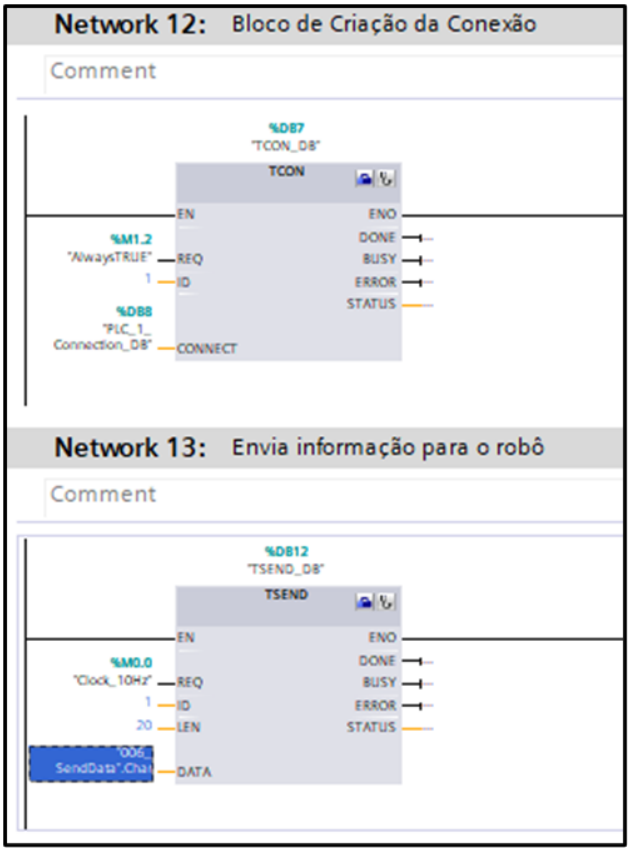
EtherNet / IP inclui comunicação cíclica (mensagens implícitas) que envia e recebe dados periodicamente e comunicação de mensagem (mensagem explícita) que envia e recebe comandos e respostas em um momento aleatório.

Com a comunicação cíclica, RPI (*Request Packet Interval*) pode ser definido de acordo com a prioridade dos dados trocados, permitindo que toda a carga de comunicação seja ajustada ao trocar dados.

Com a comunicação da mensagem, os comandos e respostas necessários são trocados no tempo necessário. A comunicação de mensagens é usada para aplicativos que não exigem a pontualidade da comunicação cíclica, como na hora de ler ou escrever as configurações do adaptador.

Como descrito anteriormente, o protocolo utilizado foi o *Ethernet/IP*. O bloco de programação que realiza as configurações da comunicação é mostrado na figura 14:

Figura 14 - Blocos de comunicação utilizados

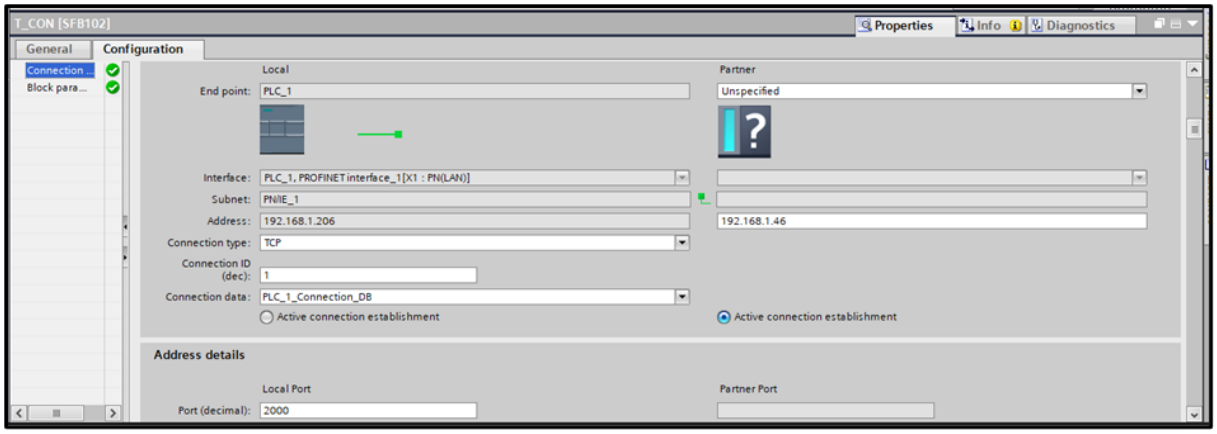


Fonte: Autor Próprio

O bloco TSEND é responsável por transmitir a informação desejada no endereço destino solicitado, no caso, a coordenada na qual o robô deve utilizar como referência no seu sistema de coordenada de base.

Essa função só funciona quando existe uma comunicação TCP associada a conexão aberta no bloco TCON. Esse bloco é utilizado para a criação de uma conexão ethernet, onde é possível configurar as informações do IP e porta do destino (adaptador), no caso o robô.

Figura 15 - Configurando o bloco TCON



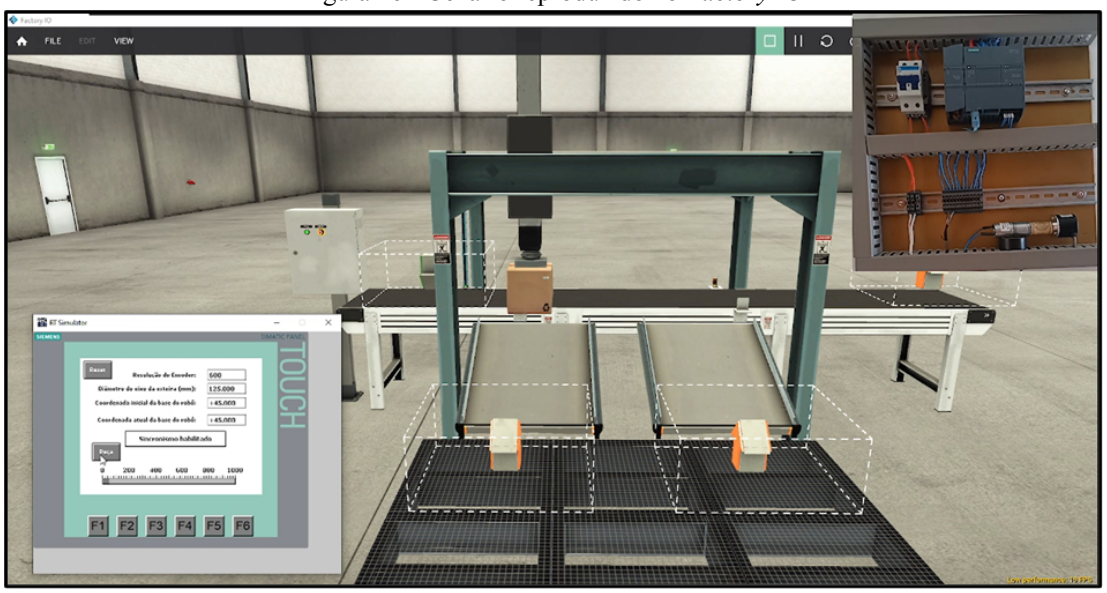
Fonte: Autor Próprio

Com a rede definida e configurada, o robô consegue atualizar sua coordenada de base em tempo real na medida que o CLP interpreta os pulsos gerados pelo *encoder*.

RESULTADOS E DISCUSSÃO

Com o TIA Portal integrado ao IO Factory, foi possível interagir com o *encoder* fisicamente enquanto o cenário era reproduzido virtualmente, conforme programação desenvolvida dentro do CLP. Abaixo é possível visualizar alguns resultados encontrados, através das figuras 16 e 17:

Figura 16 - Cenário reproduzido no *Factory IO*



Fonte: Autor Próprio

Figura 17 - Sistema cartesiano movimentando a peça conforme performance do *encoder*

Fonte: Autor Próprio

Através da plataforma *Factory IO*, foi possível verificar que o robô cartesiano correspondeu aos comandos enviados pelo controlador lógico programável. Em outras palavras, significa que a parte que corresponde ao *software* do CLP foi atendida com êxito, pois foi possível transmitir a posição atual do *encoder* para o robô dinamicamente.

CONCLUSÕES

O arranjo realizado entre CLP e robô virtual do ambiente de simulação apresenta todas as funções necessárias para o funcionamento do objetivo proposto e foi desenvolvido através dos recursos nativos e não proprietários dos robôs. Foi atingido o objetivo de reproduzir o processo *conveyor tracking* sem a necessidade de recorrer aos pacotes adicionais oferecidos pelos fabricantes de robôs. Através da metodologia apresentada, também concluiu-se que é possível adaptar o processo para diversos tipos de transportadoras, não restringindo apenas aos sistemas lineares.

REFERÊNCIAS BIBLIOGRÁFICA

CONVEYOR TRACKING GUIDE. **The conveyor tracking functionality adjusts robots paths to the motion of a conveyor.** Disponível em: https://www.universal-robots.com/articles/ur/programming/conveyor-tracking-guide/. Acesso em: 30 abr. 2021

Conveyor Tracking Short. Disponível em: http://softmc.servotronix.com/wiki/Conveyor_Tracking_Short/. Acesso em: 12 mai. 2021

Conveyor Tracking System**.** Disponível em: https://www.hiwin.tw/download/tech_doc/mar/conveyor_tracking_system-(E).pdf. Acesso em: 12 mai. 2021

Coordinates of a robot system. Disponível em: https://learnchannel-tv.com/robot/robot-coordinate-systems/. Acesso em: 01 jun. 2021

O que é *Encoder*? Para que serve? Como escolher? Como interfacear? Disponível em: https://www.hitecnologia.com.br/blog/o-que-%C3%A9-encoder-para-que-serve-como-escolher-como-interfacear/. Acesso em: 20 jun. 2021

(MUHAMMAD). A Look Into Rotary *Encoder* Types: Absolute and Incremental. Disponível em: https://control.com/technical-articles/a-look-into-rotary-encoder-types-absolute-and-incremental/. Acesso em: 05 jul. 2021

Park, T. H e Lee, B. H. **An Approach to Robot Motion Analysis and Planning for Conveyor Tracking**, IEE Transactions on Systems, Man, And Cybernetics, Vol. 22, No 2, 1992

WILHELM, W. E**. "Conveyor tracking,"** in International Encyclopedia of Robotics: Applications and Automation, R. C. Dorf, Ed. New York: Wiley, 1988