UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCOS ALEXANDRE ZORNITTA FERREIRA

DESENVOLVIMENTO DE MÉTODO DE UM BENCHMARKING PARA ANÁLISE COMPARATIVA DE DESEMPENHO DE CONTROLADOR INDUSTRIAL COMERCIAL E RASPBERRY UTILIZANDO PLATAFORMA CODESYS

CHAPECÓ
2021

MARCOS ALEXANDRE ZORNITTA FERREIRA

DESENVOLVIMENTO DE MÉTODO DE UM BENCHMARKING PARA ANÁLISE COMPARATIVA DE DESEMPENHO DE CONTROLADOR INDUSTRIAL COMERCIAL E RASPBERRY UTILIZANDO PLATAFORMA CODESYS

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Me. Adriano Sanick Padilha

CHAPECÓ
2021

Ferreira, Marcos Alexandre Zornitta

Desenvolvimento de método de um benchmarking para análise comparativa de desempenho de controlador industrial comercial e Raspberry utilizando plataforma CodeSys / Marcos Alexandre Zornitta Ferreira. – 2021.

48 f.: il.

Orientador: Prof. Me. Adriano Sanick Padilha.

Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2021.

1. CONTROLADORES LÓGICOS PROGRAMÁVEIS. 2. CONTROLE. 3. AUTOMAÇÃO. 4. CODESYS. 5. RASPBERRY. I. Padilha, Prof. Me. Adriano Sanick, orientador. II. Universidade Federal da Fronteira Sul.

E-mail: marcoszornitta@gmail.com

MARCOS ALEXANDRE ZORNITTA FERREIRA

DESENVOLVIMENTO DE MÉTODO DE UM BENCHMARKING PARA ANÁLISE COMPARATIVA DE DESEMPENHO DE CONTROLADOR INDUSTRIAL COMERCIAL E RASPBERRY UTILIZANDO PLATAFORMA CODESYS

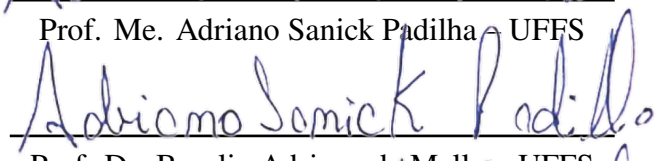
Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Me. Adriano Sanick Padilha

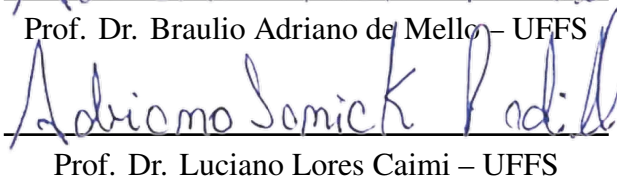
Aprovado em: 20/10/2021.

BANCA AVALIADORA

Prof. Me. Adriano Sanick Padilha – UFFS



Prof. Dr. Braulio Adriano de Mello – UFFS



Prof. Dr. Luciano Lores Caimi – UFFS

AGRADECIMENTOS

Agradeço primeiramente à Deus por me conceder saúde, força e coragem para chegar ao final deste trabalho. Aos meus pais e familiares por todo o apoio e paciência durante esta jornada.

Agradeço ao meu orientador Me. Adriano Sanick Padilha por todas as revisões, conversas e orientações que tivemos durante o período de trabalho, não somente referentes ao trabalho, mas também para a vida. Também aos membros das bancas que muito contribuíram para a execução e aprimoramento deste trabalho: Dr. Braulio Adriano de Mello e Dr. Luciano Lores Caimi.

RESUMO

Controladores lógicos programáveis (CLPs) são sistemas embarcados baseados em microprocessadores que formam hoje a principal base da automação industrial. Entretanto, a lenta evolução em seus métodos de programação e a sua falta de flexibilidade tem desacelerado significativamente o crescimento do desenvolvimento industrial, enquanto os custos e a complexidade dos problemas a serem resolvidos tendem a aumentar cada vez mais. Assim, este trabalho tem como objetivo realizar a análise comparativa de desempenho entre um CLP com hardware de baixo custo com suporte a linguagem de alto nível e um CLP tradicional de médio porte adotando metodologias de *benchmarking*, medindo o tempo de processamento de diferentes processos, para viabilizar e encorajar o uso dos hardwares de baixo custo uma vez que o desempenho do CLP de baixo custo seja igual ou superior ao do CLP tradicional. Na análise chega-se ao resultado que o CLP de baixo custo baseado no Raspberry Pi 4 alcança uma performance de processamento maior que a do CLP tradicional, porém, não foi possível chegar a um resultado comprovando que o CLP de baixo custo consiga desempenhar todas as funções do CLP tradicional em qualquer tipo de aplicação, devido a outros fatores que não foram estudados neste trabalho como interfaces de rede, protocolos de comunicação e velocidade de memória.

Palavras-chave: CONTROLADORES LÓGICOS PROGRAMÁVEIS. CONTROLE. AUTOMAÇÃO. CODESYS. RASPBERRY.

ABSTRACT

Programmable Logic Controllers (PLCs) are embedded systems based on microprocessors that currently are considered the main basis of industrial automation. However, its lack of flexibility and slow evolution in programming methods have been impacting on industrial evolution, while costs and complexity of issues to be solved tend to increase. This work aims to perform a comparative performance analysis between low-cost hardware PLC with high-level language and medium-sized traditional PLC. By means of benchmarking methodologies and multiple measurements of time processing, this paper intent to compare the performance of the PLCs, aspiring to facilitate and encourage the use of low-cost hardware. The results show that the low-cost PLC based on the Raspberry Pi 4 achieves a higher processing performance than the traditional PLC. However, due to other factors not tested in this work - such as network interfaces, communication protocols and memory speed - it was not possible to reach a conclusive result covering all functions and all application of the traditional PLC.

Keywords: PROGRAMMABLE LOGICAL CONTROLLERS. CONTROL. AUTOMATION. CODESYS. RASPBERRY.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLP Controlador Lógico Programável

E/S Entradas e Saídas

GPIO General-Purpose Input/Output

I2C Inter-Integrated Circuit

RTC Real Time Clock

SPI Serial Peripheral Interface

SSH Secure Shell Protocol

ST Structured Text

UCP Unidade Central de Processamento

SUMÁRIO

1 INTRODUÇÃO

No cenário atual, é extremamente raro um sistema de controle industrial que não faça uso de pelo menos um CLP (Controlador Lógico Programável).

O dispositivo que inicialmente foi criado para substituir os antigos painéis de reles configuráveis, nada práticos e extremamente limitados, dominou rapidamente o setor por reduzir os custos de implementação/manutenção e espaço físico de painéis e também elevar a automação a um novo nível, fornecendo ao usuário final (programadores, operadores, gerentes, etc.) um vasto leque de aplicações customizáveis e facilmente configuráveis conforme a necessidade de produção.

> Toda planta industrial necessita de algum tipo de controlador para garantir uma operação segura e economicamente viável. Desde o nível mais simples, em que pode ser utilizado para controlar o motor elétrico de um ventilador para regular a temperatura de uma sala, até um grau de complexidade elevado, controlando a planta de um reator nuclear para produção de energia elétrica. FRANCHI e CAMARGO (2008)

Entretanto, se analisarmos o estado da arte das tecnologias e compararmos as dos CLPs, podemos notar uma grande lacuna no que se trata em capacidade de software. Os CLPs estão a cada dia com hardwares mais modulares, robustos e complexos, porém a forma de programação se manteve quase inalterável durante todo esse período e em contra partida os custos de hardware e licenciamento de software aumentam significativamente, inviabilizando a automatização de vários setores de pequenas industrias. A linguagem LADDER, que se apresenta como a principal linguagem de programação dos CLPs, se destacou por conseguir modelar de forma fácil e agradável as lógicas de reles que antigamente eram realizadas fisicamente de equipamento por equipamento. Por outro lado, esta linguagem suporta algumas aplicações de funcionalidade mais avançadas, as quais os hardwares atuais poderiam suportar, como por exemplo, a utilização de tecnologias de inteligência artificial para o controle preditivo de um equipamento ou processo.

> O diagrama LADDER é apenas uma representação lógica, trabalhando somente com símbolos, não considerando a tensão envolvida nas barras de alimentação nem a intensidade da corrente pelo circuito. Os contatos e outros dispositivos, no diagrama, estão em cada momento abertos ou fechados, e as bobinas, por consequência, ficam desenergizadas ou energizadas. MORAES e CASTRUCCI (2001)

Sendo assim, de fato é uma linguagem que não leva em consideração absorver outras necessidades além das condições para o acionamento de equipamentos no chão de fábrica, utilizando sinais de sensores, contatos elétricos e redes de comunicação.

Outro fator amplamente discutido, são os limites de software que os CLPs tradicionais

possuem devido seus paradigmas de programação extremamente rígidos, os quais acabam estagnando o público-alvo de desenvolvimento limitando-se a praticamente engenheiros, técnicos de automação e eletricistas além de dificultar a implementação de funcionalidades para processos devido a falta de suporte para linguagens de alto nível (se limitando nas linguagens padronizadas pela norma IEC-61131-3) mesmo o hardware do controlador suportando tal implementação.

Não é possível discutir de forma satisfatória o avanço dos sistemas de controle e automação e o conceito de industria 4.0 levando em consideração a forma de construção de softwares para CLPs atuais e os pilares que modelam essa nova etapa da indústria. Sendo assim, este trabalho é um estudo para o desenvolvimento de um processo de *benchmarking* para a realização da análise comparativa através da métrica tempo de ciclo de scan para avaliação das performances de processamento entre um CLP baseado em um Raspberry Pi 4 e um controlador tradicional de médio porte já consolidado no mercado, de forma a encorajar novos estudos e a utilização de outros conceitos de hardwares de baixo custo com suporte a linguagem de alto nível em sistemas de automação.

1.1 OBJETIVO GERAL

Desenvolver e aplicar métodos de *benchmarking* para realizar a análise comparativa de performance de processamento de um Raspberry Pi 4 desempenhando a função de CLP com um segundo CLP de médio porte presente no mercado, utilizando mais de uma linguagem de programação normatizada pela IEC-61131-3.

1.2 OBJETIVOS ESPECÍFICOS

- Compreender o funcionamento das partes de um CLP;
- Compreender como o Codesys cria o ambiente para a utilização do Raspberry como um CLP;
- Definir as métricas de avaliação do processo de *benchmarking*;
- Criar as rotinas padrões de software nos CLPs para a execução do processo de *benchmarking* utilizando linguagens de programação padronizadas pela IEC-61131-3;
- Criar e configurar o ambiente de *benchmarking*;
- Analisar e comparar resultados obtidos do *benchmarking* entre o Raspberry e o CLP de médio porte proposto.

1.3 JUSTIFICATIVA

Este trabalho se justifica com base no atual cenário da tecnologia industrial, em que demonstrando que hardwares de baixo custo e com suporte a linguagens de alto nível (como o Raspberry Pi 4) podem atender as demandas básicas de um CLP tradicional, é possível agregar mais tecnologia as industrias por preços mais acessíveis. Assim desenvolvendo principalmente as pequenas e médias indústrias, ampliando o público-alvo dos CLPs para pessoas de fora da área de automação e elétrica, acelerando o desenvolvimento do estado da arte da automação industrial se aproximando cada vez mais do conceito de indústria 4.0.

Neste sentido, a proposta deste trabalho é desenvolver um método de *benchmarking* para avaliar a performance de um hardware de baixo custo (Raspberry Pi 4) e comparar com um CLP de médio porte, utilizando métricas e conceitos presentes nos *benchmarks* específicos para automação industrial.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão revisado elementos fundamentais para o desenvolvimento do trabalho, como: conceitos, elementos e formas de programação dos CLPs, o funcionamento da plataforma Codesys necessária para implementar as funcionalidades de um CLP no Raspberry Pi 4 e as métricas para a avaliação de desempenho dos CLPs.

2.1 CONTROLADORES LÓGICOS PROGRAMÁVEIS

Nas palavras de DEY e SEN (2020), os controladores lógicos programáveis são microprocessadores dedicados ao controle de sistemas industriais que continuamente monitoram os estados de suas entradas físicas e tomam decisões baseadas no programa de usuário para controlar dispositivos de saída.

Projetados e desenvolvidos pela General Motors (GM) em 1968, a ideia principal dos CLPs era de substituir os antigos painéis a relê (Figura 1) e que possuísse as seguintes características: sobreviver ao ambiente industrial, fácil programação, fornecer manutenção por engenheiros e técnicos e ser reutilizável. Também podem ser pensados como computadores industriais que possuem uma arquitetura desenvolvida especialmente para interfacear a ação do usuário através de E/S para instrumentos de chão de fábrica. Possuem basicamente de dois módulos: unidade central de processamento (UCP) e interface de E/S discretas.

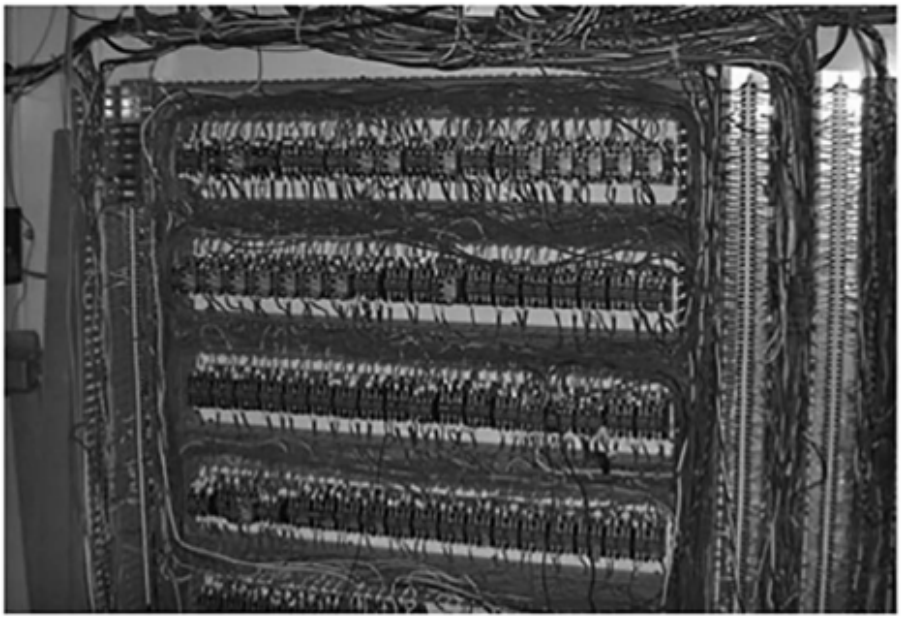


Figura 1 – Exemplo de painel a relê
Fonte: DEY e SEN (2020)

2.1.1 Unidade central de processamento (UCP)

Este é o elemento mais importante de um CLP pois é o responsável por gerenciar todas as atividades do controlador. A UCP é formada basicamente por três componentes: processador, memória e fonte de alimentação conforme, apresentada na Figura 2.

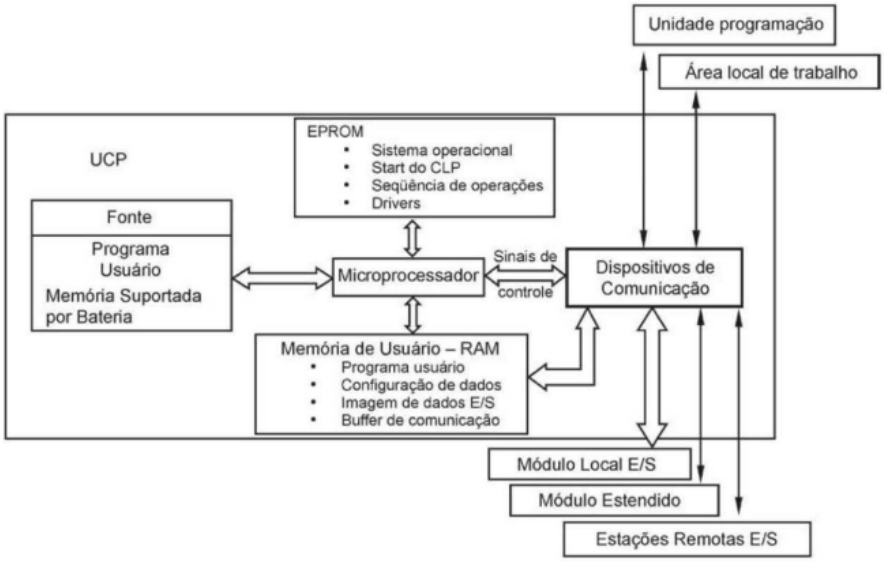


Figura 2 – Diagrama de blocos UCP
Fonte: MORAES e CASTRUCCI (2001)

2.1.2 Processador

> O processador é basicamente um microprocessador que interpreta os sinais de entrada e gera ações de controle, conforme o programa de usuário gravado em sua memória. Ele também pode se comunicar com interfaces gráficas e outros dispositivos externos (como um computador utilizado para descarregar o programa de usuário). O tamanho da palavra do processador e a velocidade de *clock* (relógio) do microprocessador indicam a velocidade de processamento do CLP. DEY e SEN (2020)

Responsáveis pelas operações matemáticas, manipulação de dados e diagnósticos de rotinas, os processadores (ou microprocessadores) realizam suas funções interpretando e executando uma série de programas de sistema que são gravados no microprocessador e possuem prioridade a programas de usuário (descrito na sessão 2.1.4). Estes programas de sistemas são encarregados de monitorar periféricos, dispositivos de campo (módulo de entrada), gerenciar memória, comunicar com interfaces de operação e realizar chamadas de sistemas para fornecer ao programa de usuário o que é necessário para a execução do mesmo.

Segundo DEY e SEN (2020), processadores geralmente conseguem manipular palavras de tamanho 4, 8, 16, 32, ou 64 bits, a qual indica o tamanho da palavra do processador. De um CLP de pequeno para grande porte, a velocidade de clock (relógio) do sistema pode variar de 1 MHz até 1GHz.

2.1.3 Ciclos de scan

A função mais básica que um CLP pode fornecer é a leitura de todos os dispositivos de entrada em campo, executar o programa do usuário para tomar decisões de acordo com os dados coletados e retornar para dispositivos de saída a ação desejada. Para isso, o processador executa a Figura 3, seguindo uma ordem sequencial de eventos chamado de scan.

Figura 3 – Representação de um ciclo de scan contínuo
Fonte: DEY e SEN (2020)

Os ciclos de scans podem ser divididos em contínuos ou periódicos, onde os contínuos são novamente chamados após serem concluídos e os periódicos possuem um intervalo entre suas chamadas. Caso um processador possua mais de um scan contínuo a execução dos mesmos se dará de forma intercalada (conforme requisição do scan ao processador).

> O tempo necessário para completar um ciclo de scan é chamado de tempo de ciclo de scan e indica a rapidez de reação do controlador às mudanças nas entradas. Se o CLP reagir a um sinal que muda de estado duas vezes durante um tempo de scan, é possível que ele não detecte essa mudança; por exemplo, se a UCP levar 8 ms para executar o ciclo de scan um programa e um contato na entrada for aberto e fechado a cada 4 ms, o programa pode não responder à mudança de estado do contato. Ela detectará uma mudança se esta ocorrer durante a atualização do arquivo tabela de imagem da entrada, mas não responderá a todas as mudanças. PETRUZELLA (2014)

PETRUZELLA (2014) também adiciona que o tempo de ciclo de scan real é calculado cada vez que a instrução END é executada e armazenado na memória do CLP, representando o final do ciclo de scan. O dado do tempo de ciclo de scan pode ser monitorado via programação do CLP (através de programas monitores) e inclui o máximo, o mínimo e o último tempos de ciclo de scan. A Figura 4 demonstra o forma de obtenção completa do tempo de ciclo de scan.

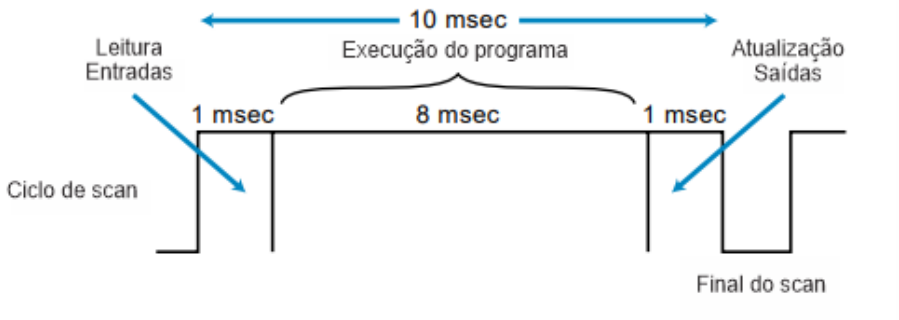


Figura 4 – Representação do tempo de ciclo de scan
Fonte: BRYAN e BRYAN (1997)

Algumas chamadas de sistema necessitam de processamento contínuo além do que um ciclo de scan consegue fornecer. Imagine que em um determinado processo de dosagem de líquidos o controlador precise manter uma bomba ligada por 30 segundos e o ciclo de scan é periódico com um intervalo de 150 milissegundos entre os scans, ou seja, o temporizador seria incrementado apenas a cada 150 milissegundos (chamada do scan) fazendo com que o controle não seja preciso. No caso dessas funções especiais, ao serem chamadas o sistema operacional se encarrega de criar um novo ciclo contínuo exclusivo para a contagem de tempo utilizando o módulo de relógio em tempo real (em inglês, Real Time Clock - RTC) presente no controlador e retornando para quem a chamou apenas os estados de execução (tempo decorrido, finalizado, rodando, etc.).

2.1.4 Memória de sistema

Umas das características mais importantes de um CLP é a possibilidade fornecida ao usuário de rapidamente alterar o programa que está sendo executado.

> A memória pode ser situada em duas categorias: volátil, que perderá suas informações armazenadas se a energia total faltar ou for desligada, pode ser alterada facilmente e é adequada à maioria das aplicações quando há uma bateria para fornecer energia para a cópia de segurança (backup); e não volátil, que tem a capacidade de reter a informação quando a energia é desligada acidentalmente ou intencionalmente, permitindo que o CLP mantenha sua programação. Como o seu nome sugere, os controladores lógicos programáveis possuem memória programável, o que permite ao usuário editar e modificar programas de controle. PETRUZELLA (2014)

Conforme representado por MORAES e CASTRUCCI (2001), um CLP possui 4 tipos de memória:

- Memória EPROM: É um tipo de memória considerada parte do próprio CLP, os quais são responsáveis pelo *start-up* do controlador, armazenamento de dados e gerenciamento das sequências de operações. Esse tipo de memória não é acessível ao usuário do CLP;
- Memória do usuário: É uma área de memória reservada para o armazenamento de instruções fornecidas pelo programa de usuário. A UCP processa esse programa e atualiza a memória de dados internos e a de imagem E/S. A memória possui dois estados:
  - RUN: em operação, com scan cíclico;
  - PROG: parado, quando se carrega o programa aplicativo no CLP.
- Memória de dados: Armazena dados referentes ao processamento do programa do usuário, isto é, uma tabela de valores manipuláveis;
- Memória-Imagem das E/S: Memória que reproduz o estado das E/S presentes no CLP.

2.1.5 Sistema discreto de E/S

O sistema discreto de E/S fornece uma interface entre as conexões dos componentes no campo e a UCP. A interface de entrada permite que a informação do estado relativa ao processo seja comunicada à UCP e, portanto, permite que esta comunique os sinais da operação pela interface de saída para os dispositivos do processo sob seu controle. PETRUZELLA (2014), conforme representados na Figura 5.

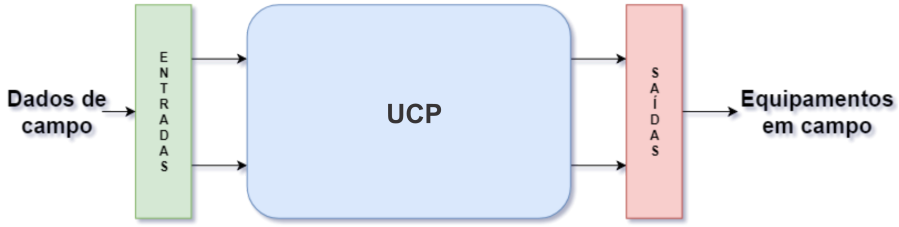


Figura 5 – Diagrama do barramento de E/S
Fonte: Adaptado de BRYAN e BRYAN (1997)

Consideremos o conceito de sistemas discretos como todo aquele em que o seu estado só muda durante os instantes $\{t_0, t_1, t_2, ...\}$ e que estes instantes são o tempo de scan reservado para atualização de E/S, então durante o tempo reservado para o programa de usuário as variáveis de campo se mantém inalteráveis MORAES e CASTRUCCI (2001).

Através de diferentes interfaces, circuitos e dispositivos de campo, o controlador é capaz de armazenar grandezas físicas (como proximidade, posição, movimentação, nível, temperatura, pressão, etc.) associadas a máquina ou processo que se deseja automatizar. Resumidamente, E/S são o conjunto de sensores, contatos elétricos e atuadores que coletam os dados e controlam o meio físico.

2.1.6 Classificação dos controladores

Segundo DEY e SEN (2020), podemos classificar a classificação dos CLPs podem ser divididas em três categorias:

- Classificação estrutural

    1. Integrado - Os CLPs que possuem integrado no mesmo hardware a fonte, unidade central de processamento (UCP) e as interfaces de E/S.
    2. Modular - Os CLPs que possuem suas partes em módulos, normalmente acoplados a uma base responsável por interliga-los através de um barramento.

- Classificação funcional

    1. Baixo desempenho - Utilizado apenas para realizar operações lógicas, operações sequenciais e algumas operações analógicas de sistemas de controle simples.
    2. Médio desempenho - Suporta todas as operações de baixo desempenho e outras, como: operações aritméticas, conversão de números, transferência de dados, E/S remotas, controle de interrupção e sub-rotinas. Utilizado para o controle de sistemas complexos.
    3. Alto desempenho - Suportas todas as operações de baixo e médio desempenho e outras, como: operações com matrizes, suporte a redes de comunicação, etc. Utilizado para o controle de processos de plantas industriais.

- Classificação por E/S

    1. Pequeno porte - Possui até 256 pontos de E/S, normalmente compostos por um único módulo com capacidade de memória de até 4000 palavras;
    2. Médio porte - Possui de 256 até 2048 pontos de E/S, com capacidade de memória de até 8000 palavras;
    3. Grande porte - Possui construção modular com UCP principal e auxiliares. Módulos E/S digitais e analógicas, módulos especializados, módulos para redes locais. Permitem a utilização de até 4096 pontos. A memória pode ser otimizada para o tamanho requerido pelo usuário.

2.1.7 IEC-61131-3

Conforme o crescimento da popularidade dos CLPs várias empresas começaram a desenvolver seu próprio controlador e utilizar a sua forma de programação, gerando um grande desconforto para os engenheiros e técnicos que utilizavam estes equipamentos. Publicado em 1992, o IEC-61131-3 é o conjunto de manobras e comandos de baixa tensão como uma estrutura mecânica dotada de uma combinação de equipamentos de manobra, controle, medição, sinalização, proteção, regulação etc. completamente montados, com todas as interconexões internas elétricas MORAES e CASTRUCCI (2001). Uma derivação deste conceito foi utilizado em 1993 dando origem ao padrão universal de linguagem para controle de processos, denominado IEC-61131-3.

O princípio básico de padronização é que um usuário possa ser capaz de criar um algorítimo para um controlador de qualquer marca e importar este mesmo algorítimo em um controlador de outra marca com o mínimo de alterações necessárias.

Segundo IQBAL, KHAN e KHAN (2013), as principais modificações que a norma IEC-61131-3 trouxe foram:

- Declaração de variáveis agora se torna estruturada assim como a declaração de variáveis em linguagens de mais alto nível;
- Declaração de tipos de dados definidos pelo usuário agora é permitido com a nova padronização;
- Variáveis globais e locais agora possuem seu próprio escopo de declaração.

Este trabalho tem como foco a programação dos controladores, em que utilizando a norma IEC-61131-3 como base de programação se realizará a comparação de desempenho entre os controladores selecionados.

2.1.8 Linguagem de programação Ladder

A linguagem ladder é a mais comum entre CLPs tradicionais. Consiste basicamente em representações simbólicas de contatos de relé (variáveis de condição) e bobinas relé (variáveis de controle) formando equações booleanas. Exemplifica MORAES e CASTRUCCI (2001) descrevendo que o diagrama ladder parte de duas linhas verticais, também chamadas de barras de alimentação (representando a linha da esquerda 24V e a linha da direita 0V de tensão). Cada representação de causalidade é feita por uma linha horizontal. Esta linha, por sua vez, é formada por pelo menos um elemento controlado (bobina relé) e um conjunto de condições para o controle desse elemento (contatos de relé).

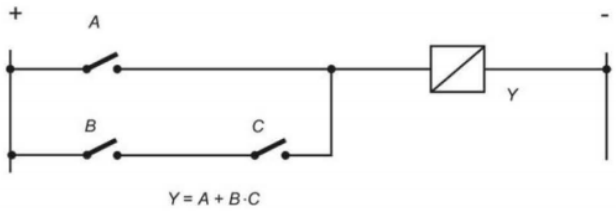


Figura 6 – Exemplo de diagrama Ladder
Fonte: MORAES e CASTRUCCI (2001)

Na Figura 6, observamos a representação da equação booleana $Y = A + B.C$ em que dois contatos abertos em série representam uma condição lógica booleana AND (.) e duas linhas em paralelo representam uma condição lógica booleana OR (+). O ladder também fornece ao usuário diversas outras funções de aritmética, temporizadores, contadores, etc. em que se passa como parâmetros de entrada os valores a serem computados e o resultado armazenado em variáveis previamente alocadas pelo usuário na memória de aplicação.

2.1.9 Linguagem de programação texto estruturado

A linguagem de texto estruturado ou (ST do inglês Structured Text) consiste em instruções delimitadas por ponto e vírgula que controlam o fluxo do programa (laços de repetições, condicionai, etc), alteram os valores de variáveis e saídas digitais/analógicas e chamam outros programas.

Na Figura 7 temos a representação da mesma lógica antes implementada em Ladder na Figura 6, agora implementada em texto estruturado.

Fora das linguagens de programação de CLPs, ST é comparável a PASCAL e C, portanto, muitos programadores que trabalharam com essas linguagens, acham mais fácil se adaptar à sintaxe do ST, MAKARCHEVA (2019).

Exemplo programação em texto estruturado

Y = A + B . C

```
IF ("A" = True) OR ("B" = True AND "C" = True) THEN
    "Y" := 1;
ELSE
    "Y" := 0;
END_IF;
```

Figura 7 – Exemplo de texto estruturado
Fonte: Adaptação da Figura 6, de MORAES e CASTRUCCI (2001)

2.2 CODESYS

A plataforma Codesys desenvolvida pela empresa alemã 3S-Smart se baseia na norma IEC-1131-3 para disponibilizar ao usuário, dentro de uma única plataforma, o ambiente de programação para diversos controladores presentes atualmente no mercado permitindo assim o uso de todas as linguagens de programação (gráficas ou textuais) apresentadas anteriormente.

O software é disponibilizado de forma gratuita para download no site oficial da ferramenta, porém alguns módulos especiais necessitam a aquisição de licenças pagas.

Segundo NUNES (2018), O software Codesys basicamente pode ser dividido em 3 camadas, sendo elas:

- Camada de Desenvolvimento: Na qual se encontra o ambiente de desenvolvimento do programa;
- Camada de Comunicação: Serve de interface de comunicação entre a camada de desenvolvimento e a cama do dispositivo, sendo responsável pela troca de variáveis entre camadas;
- Camada do Dispositivo: Para que um sistema ou dispositivo possa ser programado com o Codesys, é necessário que tenha o Runtime System do Codesys implementado e adaptado para o mesmo hardware.

2.3 RASPBERRY PI

> Raspberry Pi é um computador baseado em Linux de pequeno porte e baixo custo que pode ser conectado a um monitor, teclado e mouse. Além de ser capaz de executar todas as funções padrões de um computador *desktop* com possibilidades adicionais de interação com o mundo exterior. MAKARCHEVA (2019)

Dentro de um valor acessível, o projeto necessita de um sistema embarcado confiável, sem comprometer a execução do programa do usuário, que possua relógio em tempo real (RTC) embarcado, barramento para conexões E/S e capacidade para expansão caso necessário. Considerando tais requisitos, optou-se por utilizar o sistema embarcado Raspberry Pi 4 (Figura 8), que com algumas de suas respectivas especificações atende o solicitado:

- Processador:Cortex-A72 (ARM v8) 64-bit @1.5GHz
- Memória: 2GB SDRAM
- Barramento E/S: Conector GPIO de 40 pinos
- Expansão: Módulo de expansão através do conector GPIO

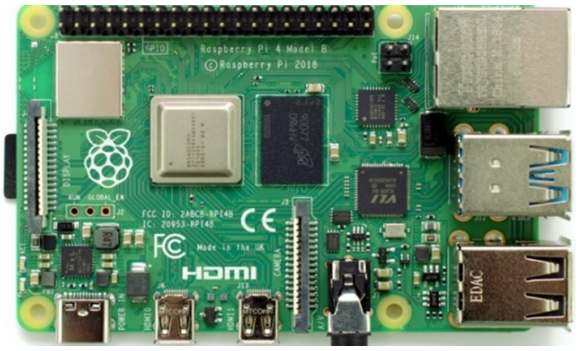


Figura 8 – Visão superior do Raspberry Pi 4
Disponível em: <https://www.tecmundo.com.br/produto/143125-raspberry-pi-4-passa-testes-impressiona-benchmark-desempenho.htm>

Porém, o sistema embarcado escolhido não tem como objetivo suportar ambientes agressivos (como o industrial) e é facilmente prejudicado por condições extremas de temperatura, umidade e principalmente ruídos elétricos que comumente são encontrados no ambiente industrial, por conta de cabos de alta tensão, inversores de frequência, etc. Por isso, deve-se incorporar ao controlador uma placa de blindagem, melhorando a eliminação de ruídos eletromagnéticos e ambientes variáveis de ambientes agressivos.

2.3.1 Conector GPIO

Também conhecido como porta GPIO (general-purpose input/output), será o responsável pela comunicação do controlador proposto com os dispositivos de campo. Presente na parte superior da placa, esse conector possui 40 pinos distribuídos com as funções demonstradas na Figura 9.

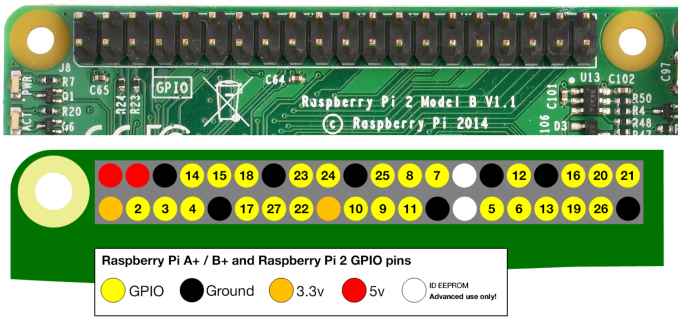


Figura 9 – Definições de pinagem conector GPIO
Disponível em: <https://www.raspberrypi.org/documentation/usage/gpio/>

Sendo assim, temos 26 pinos que podem ser configurados como entrada ou saída digital de 3.3V definidos pelo usuário.

2.3.2 Instalações e parametrizações do Raspberry

Com a utilização do Codesys Runtime, o Raspberry deixa de ser utilizado como um microcomputador convencional e torna-se um controlador "escravo"de outro computador, o qual se conecta a ele através de protocolo SSH, utilizando o software de engenharia do Codesys. Para isso, deve-se realizar uma série de passos de instalações e configurações em ambos os lados, controlador (Raspberry) e a estação de engenharia (computador com software Codesys).

Primeiramente deve-se instalar o sistema operacional no Raspberry, para isso um instalador nomeado NOOBS (New Out Of the Box Software), pode ser baixado diretamente do site oficial do Raspberry Pi e extraído em um cartão SD vazio. Uma vez plugado ao controlador, é possível selecionar o sistema operacional que se deseja instalar. No caso deste trabalho, o sistema operacional selecionado foi o Raspberry Pi OS, um sistema operacional baseado em Linux que possui suporte oficial para o Raspberry Pi.

Uma vez instalado o sistema operacional, uma série de configurações devem ser realizadas sobre o mesmo para a instalação e bom funcionamento do ambiente Codesys:

- Como o Raspberry não possui um módulo de relógio em tempo real, ele sincroniza suas configurações de data/hora através de conexão internet. Sendo assim, deve-se configurar a região correta para correção de fuso-horário;
- Deve-se habilitar as interfaces comunicação I2C e SSH, conforme demonstrado na Figura 10;

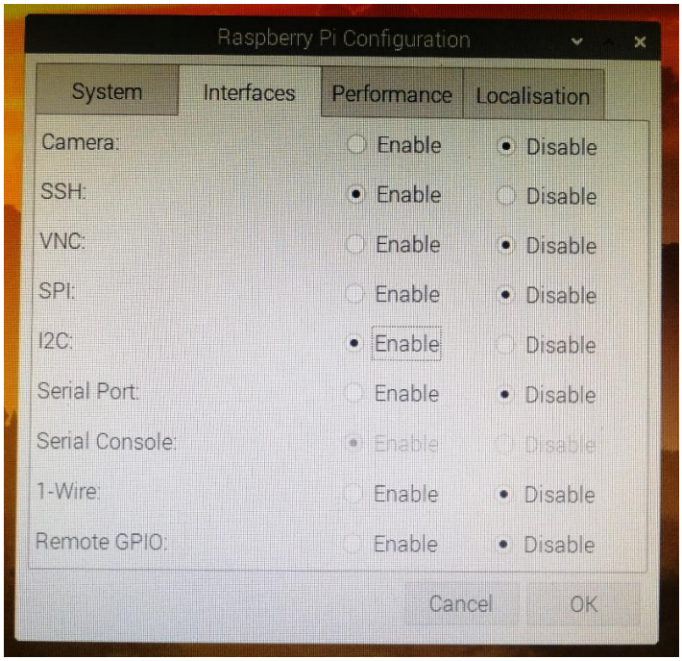


Figura 10 – Configurações de interface Raspberry
Fonte: MAKARCHEVA (2019)

- Deve-se configurar um endereço IP para o Raspberry.

Após estas configurações, o Raspberry está pronto e acessível para futuras instalações que serão realizadas através do computador utilizado como estação de engenharia.

2.4 MÉTRICAS DE AVALIAÇÃO

Para avaliar o desempenho dos controladores, um processo de aplicação de métricas denominado *benchmark* deve ser executado nos controladores que se desejam comparar.

> No mundo da computação de controle, *benchmark* é o processo de executar um programa em ladder, texto estruturado (ST), ou diagrama de blocos em um controlador industrial para graduá-lo de acordo com sua performance referente a outros similares para uma aplicação específica. IQBAL, KHAN e KHAN (2013).

Normalmente, processo de *benchmark* para programas consideram o tempo de execução de um código de mil linhas, o que não retrata de fato o tempo real de execução do mesmo para uma aplicação específica ou um hardware específico. Assim, esta forma de avaliação não se faz válida para controladores.

Para metodologia realizar processos de *benchmark*s para controladores, segundo IQBAL, KHAN e KHAN (2013), deve suprir as seguintes metas:

1. Análise de uma plataforma em particular e sua aplicação;

2. Performance com graduação baseada nos objetivos e variedades dos controladores industriais.

Porém, existem alguns obstáculos para se atingir tais metas em controladores industriais, como por exemplo:

- Os fabricantes podem ajustar os parâmetros para impulsionar o desempenho do controlador em quesitos específicos para o *benchmark*;
- Em controladores não é possível executar vários programas ao mesmo tempo;
- É comprovado empiricamente que a maioria do desempenho dos controladores degradou drasticamente em uma utilização de mais de setenta por cento da sua capacidade de processamento e é recomendado não utilizar mais de trinta por cento se o usuário estiver considerando um pequeno projeto de expansão ou modificação em um futuro próximo e não utilizar mais de cinquenta a sessenta por cento para uma resposta eficiente.

Considerando as metas a serem obtidas e as dificuldades encontradas, IQBAL, KHAN e KHAN (2013) define que a métrica de avaliação para CLPs deve ser a medição do tempo de ciclo de scan, tanto do programa de usuário, quanto o processamento das E/S. Porém, para se obter a medição correta desta métrica, é necessário estabelecer alguns pré-requisitos:

- Desenvolvimento de rotinas padrões;
- Seleção de ambientes de execução;

2.4.1 Desenvolvimento de rotinas padrão

Deve-se desenvolver rotinas padrões para os testes de *benchmark* sem que as mesmas sejam tendenciosas com o controlador a ser medido, pois um controlador pode funcionar perfeitamente sob uma condição específica. Um bom exemplo disso, são os tipos de dados (booleano, inteiro, real e etc.), pois dependendo do controlador o tempo de processamento para cada tipo de dado pode diferir.

Assim, o processo de *benchmark* não pode ser baseado utilizando apenas uma rotina de programa, mas sim várias rotinas utilizando diferentes métodos e tipos de dados a serem comparados individualmente.

2.4.2 Seleção de ambientes de execução

Para a análise do Raspberry e outro CLP de médio porte já presente no mercado, deve-se definir os ambientes de execução dos programas a serem analisados.

Como o objetivo deste trabalho é comparar o desempenho do Raspberry utilizando como comparativa um CLP de médio porte já consolidado no mercado, optou-se por utilizar o CLP

1212C DC/DC/DC (Figura 11) da familia de controladores S7-1200 do fabricante Siemens Energy & Automation, Inc (uma das marcas que dita o segmento de CLPs) por ser um CLP amplamente utilizado para diversas aplicações de pequeno e médio porte.

Figura 11 – Controlador Siemens S7-1200 1212C DC/DC/DC
Fonte: Site oficial Siemens

Memória Integrada	100 kbytes
Processamento de booleanos	0.08 µs / instrução
Processamento de inteiros	1.7 µs / instrução
Processamento de reais	2.3 µs / instrução
Capacidade de entradas digitais	8 entradas (expansível até 142)
Capacidade de saídas digitais	6 entradas (expansível até 138)

Tabela 1 – Especificações do fabricante CLP Siemens 1212C DC/DC/DC
Fonte: Site oficial Siemens

2.4.3 Medição do tempo de ciclo de scan

A medição do tempo de ciclo de scan em programas sequencias podem ser facilmente mensurados, uma vez que a duração do scan pode ser obtida registrando o tempo de início e fim da da operação. Porém, a precisão desta medição pode ser afetada uma vez que os controladores fragmentam seu processamento para executar outras operações para atualização das tabelas de E/S, aquisição de dados por rede, etc. Por isso, é necessário medir o tempo de ciclo de scan sem que nenhum programa ou função esteja sendo chamada naquele momento e descontar ao tempo total do tempo de scan com o programa sendo executado, assim conseguindo uma medição mais confiável do tempo real de execução do programa.

2.4.4 Medição de processamento de E/S

Um fator também fundamental para se avaliar o desempenho de um controlador é o tempo que o mesmo leva para processar sinais de campo (digitais ou analógicos). Com isso, deve-se criar um processo de *benchmark* para análise do processamento de operações entre sinais digitais e analógicos.

2.4.4.1 Processamento de sinais digitais

Segundo IQBAL, KHAN e KHAN (2013), os sinais de campo digitais são selecionados em sequencia de forma incremental. Para analisar o processamento desses sinais começa-se com uma operação AND entre 2 a 50 entradas executando a mesma 10000 vezes variando seus valores e medindo o tempo total da execução de cada ciclo para mensurar o tempo mínimo e máximo obtidos. Aumentando o intervalo de entradas utilizadas na operação AND adquirimos a curva do tempo de processamento por entradas digitais presentes em cada operação.

A Figura 12 exemplifica este método aplicado pelo autor IQBAL, KHAN e KHAN (2013) para comparação do tempo de processamento entre dois controladores de marcas diferentes em que o mesmo ao rodar as 10000 vezes marcou o tempo mínimo e máximo de execução de um ciclo de entradas digitais.

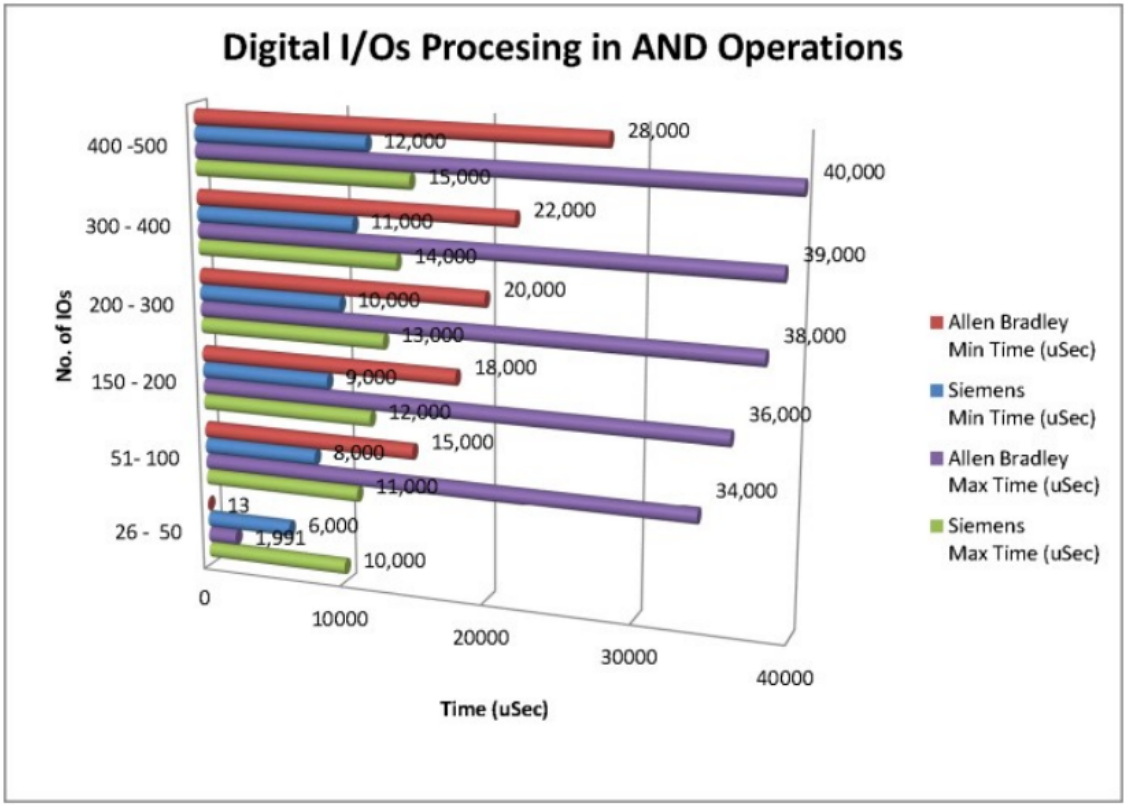


Figura 12 – Comparação de processamento de entradas digitais entre dois controladores
Fonte: IQBAL, KHAN e KHAN (2013)

2.4.4.2 Processamento de sinais analógicos

O processo para obtenção do tempo máximo e mínimo de processamento de entradas analógicas se assemelha ao processo dos sinais digitais, porém neste caso, deve-se utilizar operações de soma ou subtração entre os valores sempre os modificando a cada ciclo de scan de monitoramento.

O gráfico da Figura 13 exemplifica este método aplicado pelo autor IQBAL, KHAN e KHAN (2013) para comparação do tempo de processamento entre dois controladores de marcas diferentes, em que o mesmo ao rodar as 10000 vezes, marcou o tempo mínimo e máximo de execução de um ciclo de processamento de entradas analógicas.

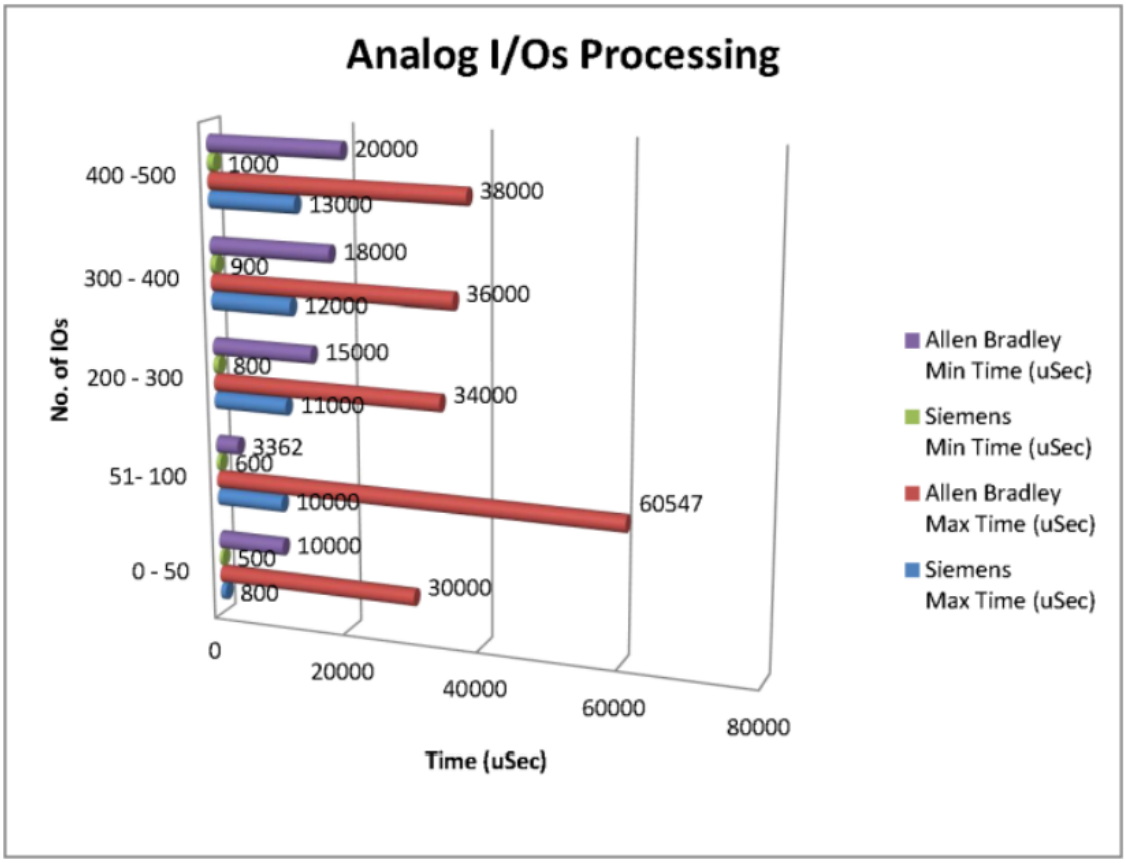


Figura 13 – Comparação empírica de processamento de entradas analógicas entre dois controladores

Fonte: IQBAL, KHAN e KHAN (2013)

3 TRABALHOS RELACIONADOS

Os trabalhos relacionados selecionados inspiraram a proposta deste trabalho, contribuindo com a construção da metodologia a ser aplicada.

Em seu trabalho, NUNES (2018) explorou a viabilidade da utilização do Raspberry, com o objetivo de redução de custos para a implementação de um sistema de medição e cálculos de grandezas elétricas. Utilizando a plataforma Codesys para a configuração e programação do Raspberry, juntamente com um Arduino Uno para a obtenção das grandezas em campo. O autor comprovou através da implementação e demonstração de valores que a aplicação de sistemas utilizando Raspberry podem e devem ser explorados como forma de baratear os custos de sistemas de automação de pequeno e médio porte.

Seguindo o objetivo de diminuir os custos de hardware, MAKARCHEVA (2019) traz em seu trabalho um estudo de caso, substituindo um CLP comercial existente em uma aplicação por um Raspberry utilizando a plataforma Codesys, com o intuito de realizar aquisição de dados de processos e máquinas existentes para posteriormente disponibilizar dados qualitativos sobre os mesmos em uma interface WEB. Comparando o Raspberry e outros controladores comerciais, a autora através de alguns parâmetros como: capacidade de processamento, quantidade de memória e custo, conclui que a utilização do Raspberry com a plataforma Codesys trará um melhor custo-benefício para a aplicação em questão. Conclui-se pela autora, que com algumas modificações de hardware e tratativas de software específicas (principalmente tratativas de erros de comunicação e perda de alimentação do controlador) o Raspberry atende a aplicação sugerida, porém alerta-se para o uso do mesmo em aplicações que requerem alta precisão (controles PID, controles de posicionamento, etc) pela falta de alguns módulos de forma nativa (como por exemplo o RTC), categorizando-o para utilização em aplicações restritas a coleta e tratamento de dados em pequenas aplicações.

Trazendo em seu trabalho métricas que comprovam a viabilidade do Raspberry Pi 3 em comparação com outros dois hardwares propostos (Arduino Mega e BeagleBone Black), ROMERO (2018) desenvolve o controle de uma planta didática para controle de fluxo e nível de tanques utilizando a plataforma Codesys e hardware Raspberry Pi 3, implementando malhas de controle PID para ajuste automático do sistema. O autor conclui com sua pesquisa, que o hardware selecionado juntamente com a plataforma Codesys atendeu os processos que necessitam de poucos recursos de campo (sensoriamento e ativação de componentes), sendo uma escolha de baixo custo para estes.

Os autores IQBAL, KHAN e KHAN (2013) trazem em seu trabalho formas de medir e graduar o desempenho de CLPs que seguem a norma de padronização IEC-61131-3. Demonstrando métricas para medição de tempo de ciclo de scan, atualização das tabelas de E/S e desempenho de contadores. Os autores afirmam que com os dados obtidos, através das métricas aplicadas, pode-se comparar o desempenho de diferentes controladores (que sigam a normativa IEC-61131-3) de forma a graduar indicadores e realizar o processo de *benchmark* entre

os dispositivos selecionados. Porém, os métodos mostrados pelos autores foram aplicados em emuladores de CLPs de duas marcas presentes no mercado, se diferenciando do que é proposto neste trabalho, onde as métricas foram aplicadas em CLPs reais, sendo um deles de baixo custo e não tradicional (dispositivo que já tem como proposta ser um CLP).

4 DESENVOLVIMENTO

4.1 MÉTRICAS ADOTADAS

A métrica de avaliação e os procedimentos de configuração (pré-requisitos) adotados para o desenvolvimento deste trabalho foram embasados na capacidade de hardware de cada CLP, sendo necessário adapta-los de forma que independentemente do hardware ou software, para que a análise fosse imparcial. Assim, a métrica selecionada foi o tempo de ciclo de scan e os procedimentos de configuração selecionados foram:

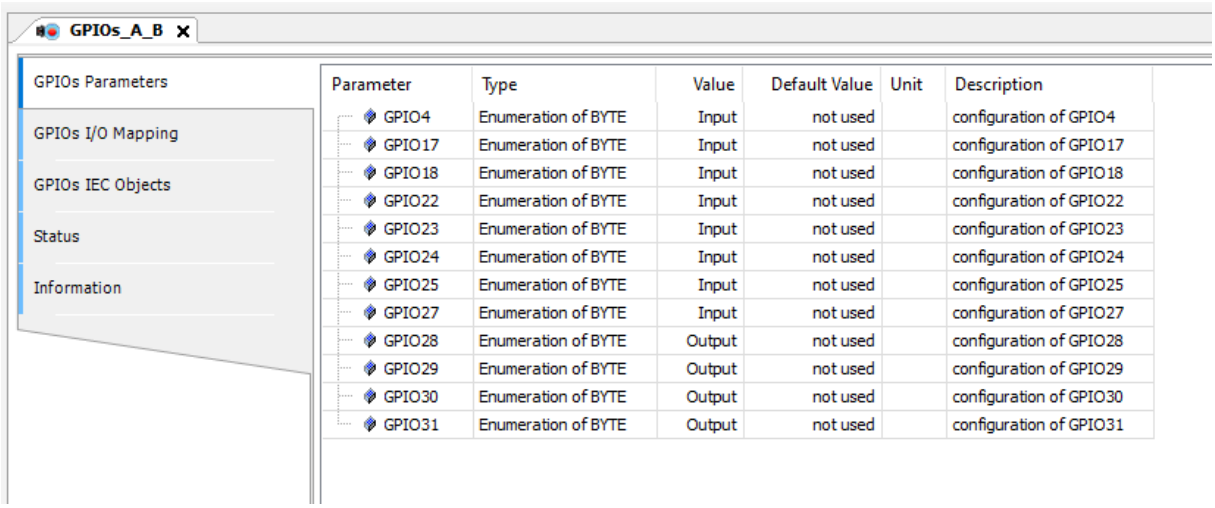
- Preparação dos ambientes de execução;
- Rotinas padrão para processo de *benchmarking*;

4.2 PREPARAÇÃO DOS AMBIENTES DE EXECUÇÃO

Conforme descrito por IQBAL, KHAN e KHAN (2013), a seleção dos ambientes de execução em que se deseja realizar a análise comparativa deve ser feita de forma que ambos não sejam consideravelmente divergentes em capacidade de hardware, como: arquitetura do processador, quantidade de memória, quantidade de E/S, etc.

Entretanto, controladores tradicionais com especificações de arquitetura de processador e quantidade de memória que sejam próximos ao do Raspberry Pi 4, dificilmente são encontradas no mercado. Assim, para tentar nivelar os ambientes de execução foi utilizado no Raspberry um sistema operacional 32 bits (CLPs mais utilizados atualmente possui arquitetura 32 bits) e a principal característica para a seleção do CLP comercial foi a quantidade de E/S. Neste trabalho o CLP Raspberry foi configurado no modo "*Multicore*", em que se utiliza mais de um núcleo de processador para processamento paralelo de tarefas caso necessário.

O Raspberry Pi 4 pode utilizar 12 pinos GPIO (das 40 disponíveis) para serem configuradas como entradas ou saídas digitais, as demais ficam reservadas para outras funcionalidades como SPI, I2C e alimentação. A Figura 14 demonstra as configurações dos pinos GPIO utilizados para o processo de *benchmarking* realizado nestre trabalho.



Parameter	Type	Value	Default Value	Unit	Description
GPIO4	Enumeration of BYTE	Input	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	Input	not used		configuration of GPIO17
GPIO18	Enumeration of BYTE	Input	not used		configuration of GPIO18
GPIO22	Enumeration of BYTE	Input	not used		configuration of GPIO22
GPIO23	Enumeration of BYTE	Input	not used		configuration of GPIO23
GPIO24	Enumeration of BYTE	Input	not used		configuration of GPIO24
GPIO25	Enumeration of BYTE	Input	not used		configuration of GPIO25
GPIO27	Enumeration of BYTE	Input	not used		configuration of GPIO27
GPIO28	Enumeration of BYTE	Output	not used		configuration of GPIO28
GPIO29	Enumeration of BYTE	Output	not used		configuration of GPIO29
GPIO30	Enumeration of BYTE	Output	not used		configuration of GPIO30
GPIO31	Enumeration of BYTE	Output	not used		configuration of GPIO31

Figura 14 – Definições dos pinos GPIO Raspberry na plataforma Codesys
Fonte: O autor

Assim, o controlador tradicional selecionado foi o Siemens 1212C DC/DC/DC por suas especificações de hardware serem próximas ao do Raspberry conforme descrito na Tabela 1.

O Raspberry e o 1212C possuem preparação e configurações específicas para a execução do processo de *benchmarking* em especial o ambiente do Raspberry, em que bibliotecas adicionais e interfaces de comunicação devem ser instaladas, conforme demonstrado no anexo A.

4.3 ROTINAS PADRÃO PARA PROCESSO DE *BENCHMARKING*

As escolhas de rotinas padrão para consumo de processamento (gerando os tempos de ciclo de scan) dos controladores seguiram três critérios:

- Ciclo de scan sem programa de usuário;
- Tempo de processamento das E/S;
- Operações em memória.

4.3.1 Tempo de processamento sem programa de usuário

O objetivo de se registrar o tempo de ciclo de scan dos CLPs sem que nenhum programa de usuário esteja sendo executado é definir o tempo de scan que cada controlador consome para executar suas funcionalidades de sistema operacional, como: escalonamento de processos, atualização das tabelas de E/S, execução do programa monitor, etc.

Este tempo pode ser utilizado como base para definir o tempo de ciclo mínimo de um CLP e também deve ser descontado dos demais testes para se obter apenas o tempo de ciclo do programa de usuário que está sendo executado.

4.3.2 Tempo de processamento entre operações das E/S

Nesse processo são realizadas operações entre as entradas digitais do controlador para verificar o tempo de scan para a atualização das tabelas de E/S. A operação definida foi a XOR, uma vez que utilizando esta operação é necessitar verificar todas as entradas a fim de definir se a quantidade de entradas com níveis lógicos alto é ímpar, para então definir o valor da saída. O que poderia não ocorrer em uma lógica utilizando operações OR por exemplo, em que caso apenas um valor possua nível lógico alto, o valor da saída já é conhecido. Possibilitando assim alguma forma de simplificação pelo compilador de um dos CLP utilizados. Assim, são utilizadas operações XOR em série, conforme demonstrado na Figura 15. Note que na os símbolos "=1"não representam o estado lógico da operações, mas sim a representação simbólica da operação XOR.

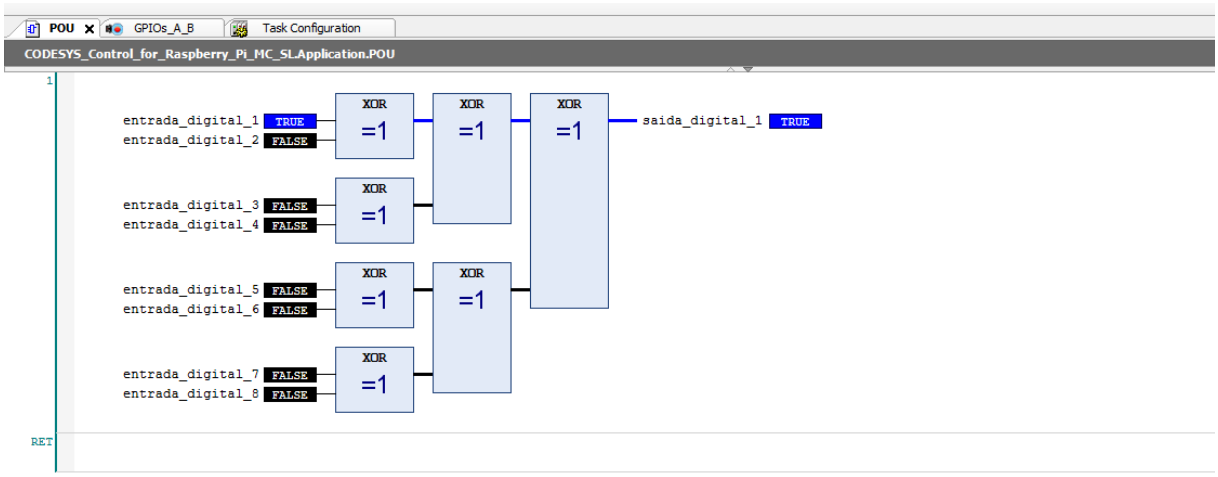


Figura 15 – Lógica de operações XOR entre as entradas do CLP Raspberry na plataforma Codesys

Fonte: O autor

A mesma ação poderia não ocorrer caso fosse utilizada uma operação lógica OR ou AND por exemplo, em que o compilador poderia de alguma forma simplificar a operação de modo em que qualquer uma das entradas estivesse em nível lógico alto ou baixo, dependendo da operação utilizada, o valor de saída já seria conhecido.

4.3.3 Tempo de processamento de operações em memória

Nesse processo são realizadas operações de cópia e transferência de valores em endereços de memórias alocados em vetores de tamanhos pré definidos pelo usuário com o intuito de calcular o tempo de ciclo para as operações envolvidas na execução das mesmas.

Para este trabalho, o algoritmo utilizado é semelhante a uma funcionalidade de "*array shift*", em que existem dois vetores de mesmo tamanho n e mesmo tipo de dado e em uma estrutura de *loop* a posição x do primeiro vetor é copiado e transferido para a posição $x + 1$ do segundo vetor até $n - 1$, conforme demonstrado na Figura 16.

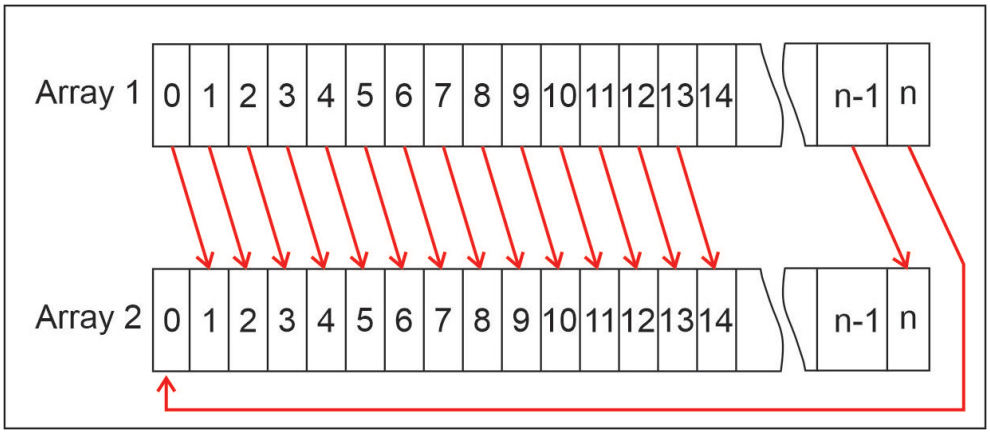


Figura 16 – Representação gráfica do algorítimo "*Array Shift*"
Fonte: O autor

Dois pontos que devem ser levados em consideração são: tamanho dos vetores e o tipo de dado a ser utilizado. Ambos afetarão diretamente a quantidade de memória a ser alocada e também no tempo do ciclo de scan. Considerando que o controlador 1212C possui uma quantidade de baixa de memória, o tipo de dado selecionado foi o UINT com tamanho de 2 bytes em dois testes, um com dois vetores de 1.000 posições e outro com dois vetores de 10.000 posições cada. É importante ressaltar que, devido a quantidade de memória do controlador 1212C, ao utilizar um tipo de dado maior (como DINT, de 32 bits), torna-se menor a capacidade de posições dos respectivos vetores e assim descaracterizando a finalidade dos testes com *array shift*. Com isso, em um segundo teste, foi utilizado dois vetores de tamanho 1.000 com uma estrutura de dados do tipo STRING(20) (uma vetor de tamanho 20 caracteres), gerando um formato de matriz que cada posição do vetor existe um outro vetor do tipo CHAR com 20 posições de tamanho 1 byte cada, influenciando no tempo de ciclo de scan dos CLPs. Neste caso, devido a necessidade da utilização de laços de repetição para realizar as cópias de cada posição do primeiro vetor, foi utilizado a linguagem de programação texto estruturado, conforme demonstrado na Figura 17.

```
Array_Shift
4   END_VAR
5   VAR
6       Array1: ARRAY[0..ArraySize] OF UINT;
7       Array2: ARRAY[0..ArraySize] OF UINT;
8       i: UINT;
9   END_VAR

1   FOR i := 1 TO ArraySize DO
2       Array2[i] := Array1[i - 1];
3   END_FOR
4   array2[0] := array1[ArraySize];
```

Figura 17 – Lógica array shift implementada em texto estruturado
Fonte: O autor

4.4 MEDIÇÃO DOS TEMPOS DE CICLO DE SCAN

4.4.1 Utilização do programa monitor das plataformas para coleta de tempos de scan

Cada CLP possui um chamado "programa monitor", com objetivo de recolher informações de execução sobre o mesmo, como: quantidade de memória utilizada, logs de falhas e ciclos de scan. A plataforma de programação de cada CLP acessa essas dados e traz ao usuário para serem utilizados como diagnósticos do sistema, conforme é demonstrado o programa monitor da plataforma TIA Portal da Siemens na Figura 18 e o programa monitor da plataforma Codesys na Figura 19.

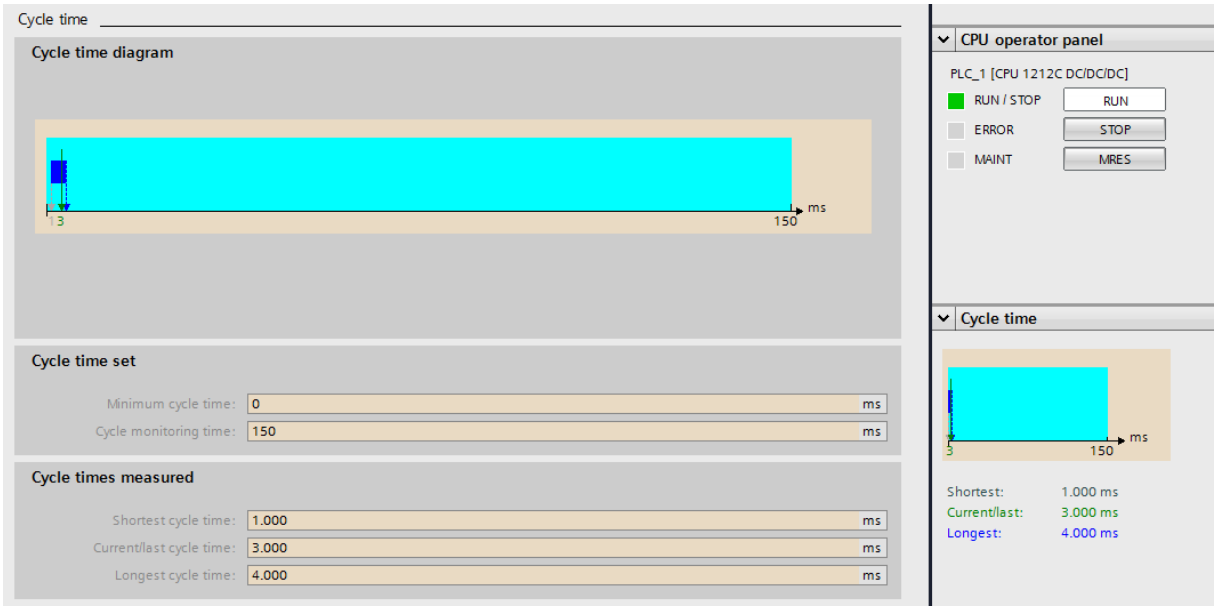


Figura 18 – Informações de tempo de ciclo de scan do programa monitor na plataforma TIA Portal da Siemens
Fonte: O autor

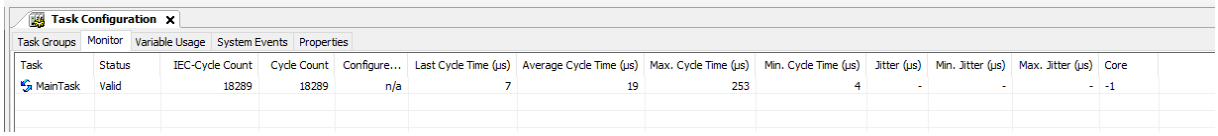


Figura 19 – Informações de tempo de ciclo de scan do programa monitor na plataforma Codesys
Fonte: O autor

Com esses valores obtidos também é possível realizar a coleta de dados para o processo de *benchmarking* através dos valores de ciclo de scan, que são eles:

- Menor tempo de ciclo de scan;
- Maior tempo de ciclo de scan;
- Último tempo de ciclo de scan.

4.4.2 Utilização de osciloscópio para coleta de tempos de ciclo de scan

Analisando as medições dos ciclos de scan através do programa monitor do CLPs, notou-se uma diferença na forma de com o tempo de scan era coletado. O programa monitor do CLP 1212C apresentava o tempo de scan completo do controlador, com execução do programa de usuário, execução de funções internas, etc. Enquanto o programa monitor do CLP Raspberry, apresentava apenas o tempo de scan da execução da rotina do programa do usuário.

Então, optou-se por realizar uma segunda forma de medição do ciclo de scan utilizando um osciloscópio como ferramenta externa para monitorar uma saída digital dos CLPs, alternando seu estado lógico ao final de cada ciclo de scan. Assim, foi possível estabelecer o mesmo parâmetro de medição para os tempos de scan sem que necessite utilizar ferramentas do próprio fabricante, o que poderia também não ser coerente com a realidade (devido alguma modificação de parâmetros no programa monitor pelo fabricante).

Para isso, ambos os controladores foram instalados em uma bancada, conforme demonstrado na Figura 20, criando um ambiente controlado, para a coleta dos tempos de scan das rotinas padrão selecionadas.

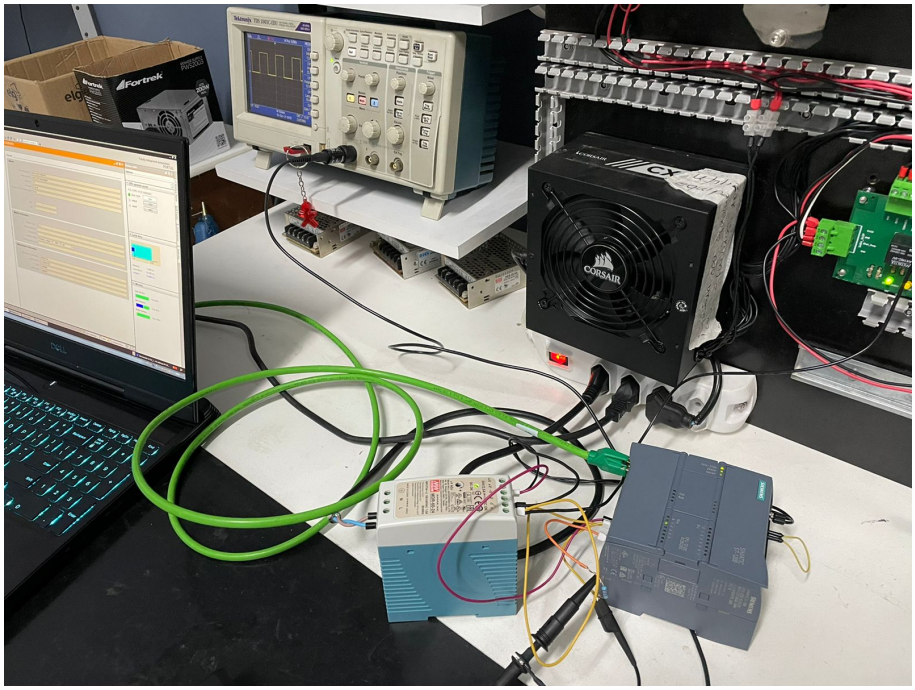


Figura 20 – Bancada para coleta de tempos de scan
Fonte: O autor

Assim, os *benchmarks* foram realizados novamente medindo a largura do pulso presente na saída digital selecionada nos CLPs através do osciloscópio.

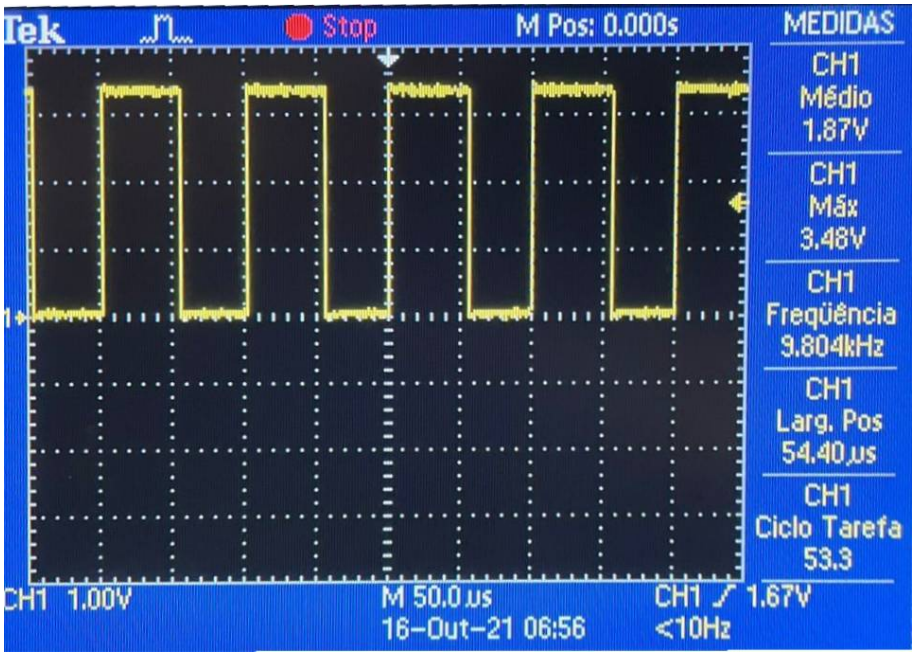


Figura 21 – Exemplo de medição de ciclo de scan no osciloscópio
Fonte: O autor

Na Figura 21, observamos o campo "Larg. Pos", que nos apresenta o intervalo de tempo entre a borda de subida e descida de um pulso, o que representa o tempo de scan entre uma execução e outra do programa de usuário em um ciclo contínuo.

5 RESULTADOS

Os resultados obtidos através do monitoramento do tempo de ciclo de scan das rotinas padrão executadas nos CLPs, foram organizados de forma a confrontar a média dos tempo de ciclo de scan obtidos entre o mesmo caso de teste nos dois CLPs selecionados e separados pela medição através do programa monitor e com o osciloscópio, conforme podemos observar nas Figuras 22 e 23.

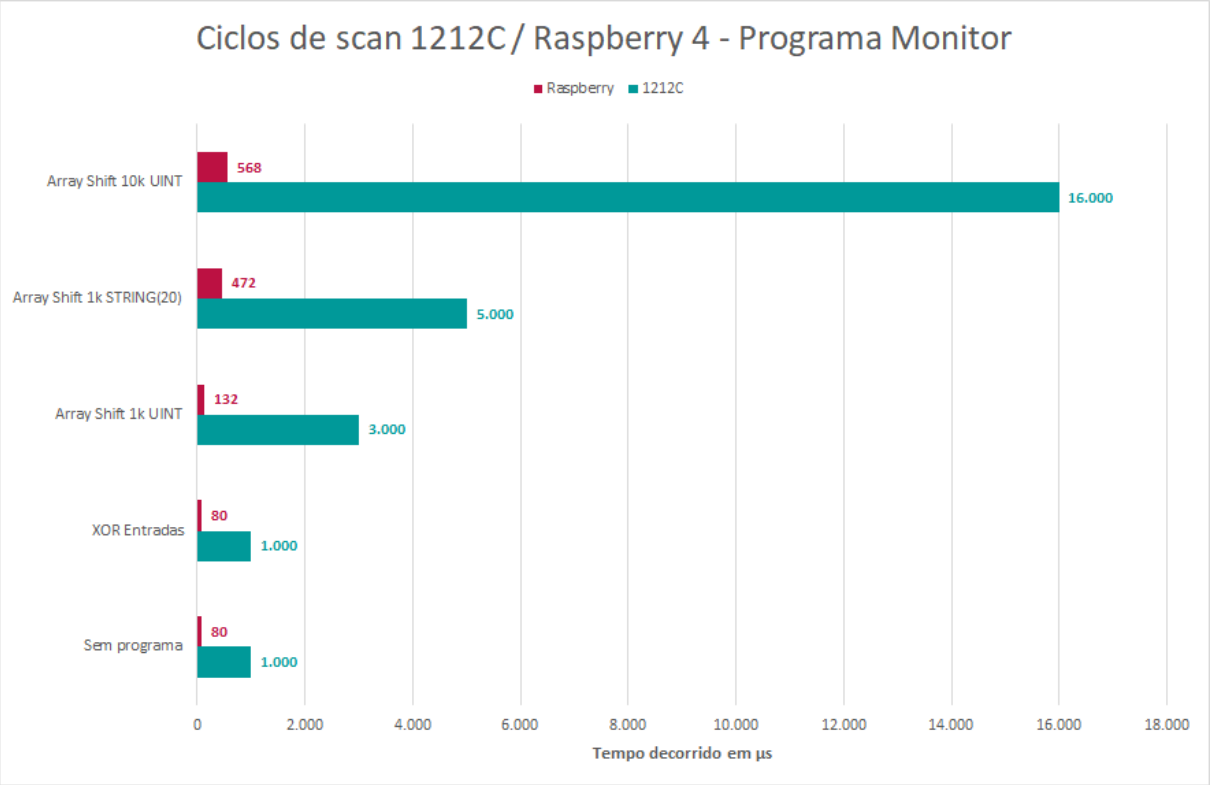


Figura 22 – Gráfico de tempos de ciclo de scan obtidos através do programa monitor
Fonte: O autor

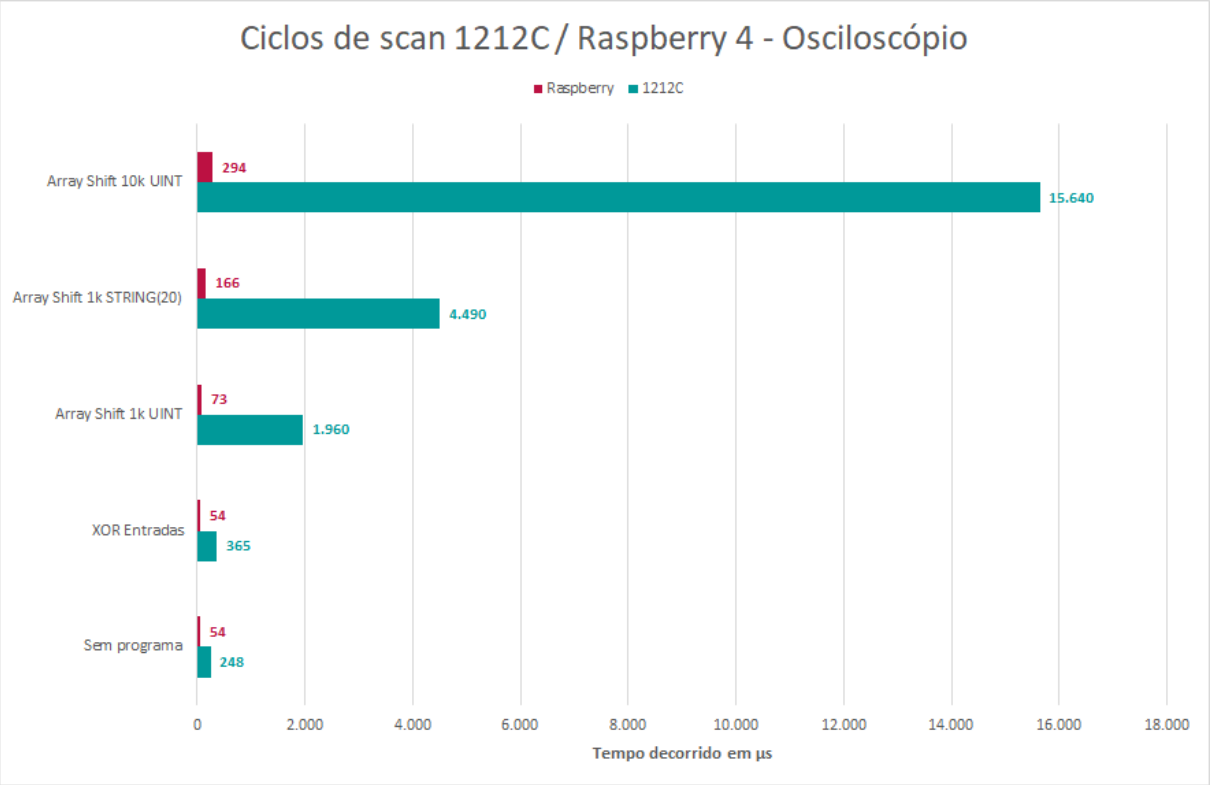


Figura 23 – Gráfico de tempos de ciclo de scan obtidos através do osciloscópio
Fonte: O autor

Os dois gráficos foram apresentados como forma de comparação entre a diferença das medidas com o programa monitor e utilizando o osciloscópio como ferramenta externa de medição. Para a discussão deste trabalho, serão considerados os resultados obtidos através da utilização do osciloscópio.

Conforme representado no gráfico da Figura 23, não há diferença significativa de desempenho entre os CLPs não executarem nenhum programa de usuário e executar somente operações entre as entradas digitais, pelo fato de os CLPs selecionados possuírem uma baixa quantidade de entradas digitais. Por exemplo, se verificarmos a Tabela 1 o CLP 1212C consome 0,08 µs para processar uma operação booleana, sendo um total de 7 operações XOR entre 8 variáveis booleanas o tempo total de processamento seria 0,56 µs, o que é um tempo irrelevante aos parâmetros adotados neste trabalho, mas que poderia ter influência sobre validações para a analise de sistemas em tempo real. O mesmo ocorre com o CLP Raspberry.

Assim, como a atualização da memória-imagem das E/S já é contemplada pelo processamento consumido na execução das funções de sistema CLP, a diferença do tempo de ciclo de scan sem a execução de programa de usuário e da execução de operações booleanas com as entradas digitais se tornam irrelevantes.

Um fator a ser analisado é o tipo de dado utilizado na operação de *array shift*. Podemos ver na Tabela 2 o comportamento dos CLPs com a alteração dos tamanhos dos vetores utilizados com o mesmo tipo de dado e o comportamento dos CLPs alterando o tipo do dado de UINT para STRING(20) mantendo os mesmos tamanhos dos vetores.

	Sem Programa	**XOR Entradas**	**UINT 1k**	**UINT 10k**	**STRING(20) 1K**
1212C	248 µs	365 µs	1960 µs	15640 µs	4490 µs
Raspberry	54 µs	54 µs	73 µs	294 µs	166 µs
Razão	4,59 x	6,76 x	26,85 x	53,20 x	27,05 x

Tabela 2 – Tabela razão do tempo de scan entre os tipos testes realizados
Fonte: O autor

Uma vez que as operações são realizadas em 32 bits (delimitado pelo processador ou pelo sistema operacional), um vetor de tipo UINT que possui 16 bits cada pode ser processado sem muito esforço. Ao modificar o tipo de dado para STRING(20), temos então um novo vetor de 20 posições do tipo de dado CHAR para cada posição do vetor STRING. Considerando que para cada CHAR é necessário alocar 8 bits, no total serão 160 bits para cada posição do vetor STRING, ou seja, 5 vezes o suportado pelo processador/sistema operacional e assim o forçando a particionar as operações em novas operações menores e consequentemente aumentando o consumo de processamento.

6 CONCLUSÃO

Alcançou-se com o presente trabalho, os propostos objetivos de criar um método de *benchmarking* e realizar a análise de desempenho de processamento entre os CLPs 1212C e Raspberry de forma satisfatória. A possibilidade de utilização do Raspberry como um CLP industrial somado a plataforma Codesys, se mostrou capaz de executar as funcionalidades básicas de um CLP com desempenho de processamento médio 23,69 vezes maior, se analisarmos a média das razões obtidas entre os tempos de ciclos de scan de cada *benchmarking*, se comparado ao CLP 1212C do fabricante Siemens. Somando a isso, o fato de o Raspberry rodar um sistema operacional com suporte a linguagem de alto nível, como Python e C++, permite ao usuário a implementação de programas que podem ser executados paralelamente ao processamento das suas funções de CLP através do serviço Codesys Runtime, gerando a quebra de alguns paradigmas presentes nos CLPs tradicionais e assim expandindo a sua capacidade de atender aplicações mais complexas (como a utilização de inteligências artificiais) e em contrapartida reduzindo os custos de implementação.

Contudo, para certificar a efetividade do mesmo em aplicações reais, alguns outros fatores que não foram analisados neste trabalho, como: robustez, interfaces de rede, protocolos de comunicação e escalabilidade, devem ser avaliados em novas análises.

Com isso, o estudo realizado neste trabalho serve como incentivo para novos estudos que gerem o avanço dos CLPs tradicionais e a quebra dos fortes e por vezes obsoletos paradigmas que os sistemas de controle atuais utilizam, assim evoluindo a tecnologia empregada nas indústrias e integrando a automação industrial a outros diversos setores da tecnologia, que atualmente, acabam se conflitando.

REFERÊNCIAS

BRYAN, L.A., e E.A. BRYAN. 1997. *Programmable controllers: theory and implementation.* Industrial Text: Georgia, USA.

CHIAVENATO, I. 1999. *Introdução à teoria geral da administração.* Makron Books: São Paulo, Brasil.

DA SILVA, Marcelo Eurípedes. 2007. "CONTROLADORES LÓGICO PROGRAMÁVEIS - LADDER", http://groupmaxi.com.br/parker/produtos-omrom-plc.pdf.

DEY, Chanchal, e Sunit Kumae SEN. 2020. *Indutrial Automation Technologies.* Taylor Francis Group, LLC.

FRANCHI, Claiton Moro, e Valter Luís Arlindo de CAMARGO. 2008. *Controladores Lógicos Programáveis: Sistemas Discretos.* Érica: São Paulo, Brasil.

IQBAL, Shafqat, Saad Ahmad KHAN e Zubair Ahmad KHAN. 2013. "BENCHMARKING INDUSTRIAL PLC PAC AN APPROACH TO COST EFFECTIVE INDUSTRIAL AUTOMATION", http://eecs.ucf.edu/~skhan/Publications/Download/SIqbal-2013-ICOSST.pdf.

MAKARCHEVA, Anna. 2019. "IMPLEMENTATION OF A PLC ODE ON RASPBERRY PI IN CODESYS ENVIRONMENT", https://www.theseus.fi/bitstream/handle/10024/302987/Makarcheva_Anna.pdf?sequence=2&isAllowed=y.

MORAES, Cicero Couto, e Plinio de Lauro CASTRUCCI. 2001. *Engenharia de Automação Industrial.* LTC: São Paulo, Brasil.

NUNES, Alisson Fernandes. 2018. "DESENVOLVIMENTO DE UM SISTEMA SUPERVISÓRIO DE BAIXO CUSTO PARA SISTEMAS ELÉTRICOS", https://www.monografias.ufop.br/bitstream/35400000/1626/6/MONOGRAFIA_DesenvolvimentoSistemaSupervis%C3%B3rio.pdf.

PETRUZELLA, Frank D. 2014. *Controladores Lógicos Programáveis.* AMGH: Porto Alegre, Brasil.

ROMERO, Juan David Castañeda. 2018. "DESARROLLO DE UNA PLANTAFORMA HARDWARE CONFIGURABLE PARA LA IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL DE PROCESO EN UNA PLANTA DIDÁCTICA", https://red.uao.edu.co/bitstream/handle/10614/10986/T08539.pdf?sequence=5&isAllowed=y.

ANEXO A

Este anexo demonstra a instalação dos pacotes adicionais do software Codesys que devem ser instalados para a utilização do Raspberry Pi 4 como um CLP.

Para o funcionamento do software Codesys no Raspberry Pi 4, é necessário realizar a instalação de dois pacotes adicionais:

- *CODESYS Edge Gateway for Linux*;
- *CODESYS Control for Raspberry PI*.

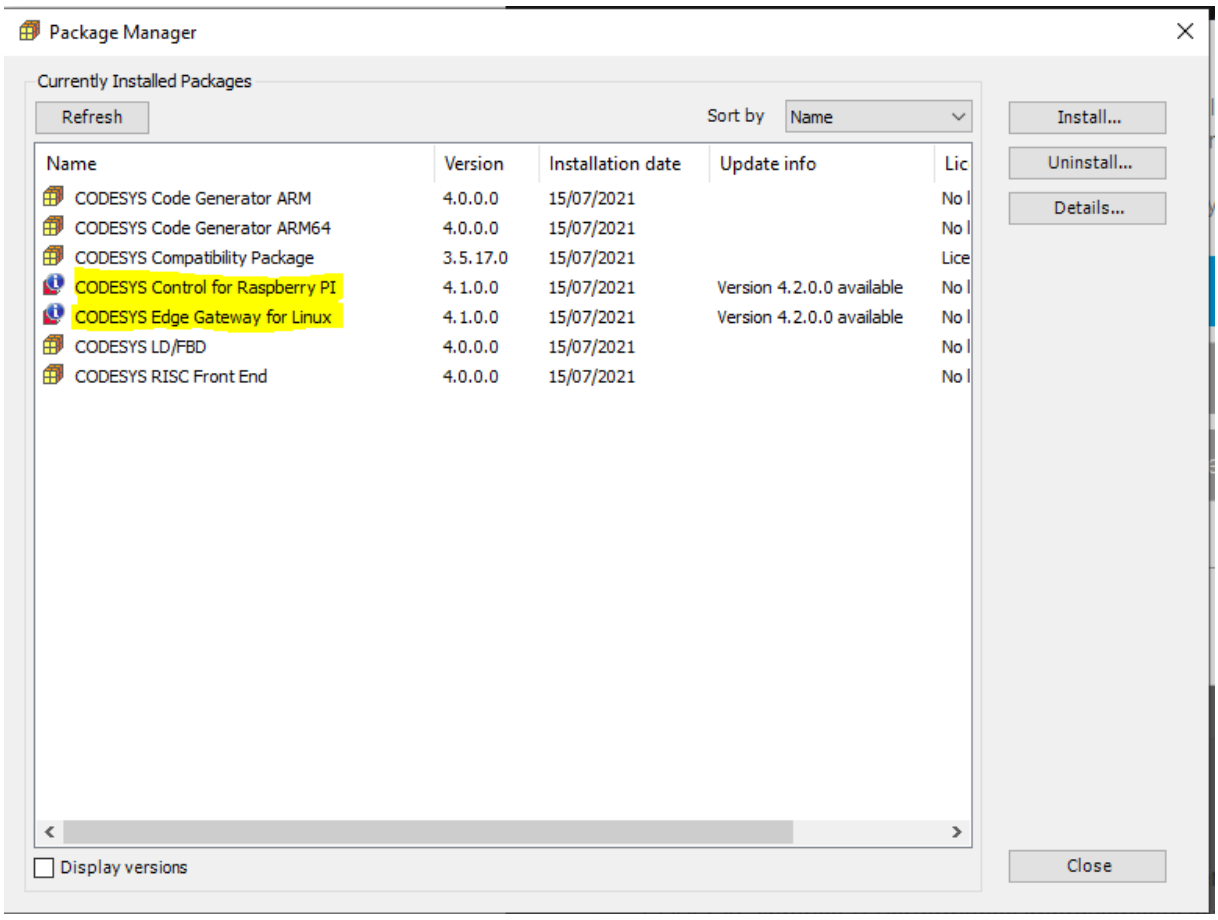


Figura 24 – Lista de pacotes adicionais instalados no Codesys
Fonte: Do autor

CODESYS Edge Gateway for Linux

Este pacote adicional serve como um driver de comunicação que possibilita o servidor do Codesys se conectar através de interfaces de rede com outros dispositivos com sistema operacional baseado em Linux. No caso deste trabalho, este pacote serve para realizar a conexão SSH entre o Codesys e o Raspberry Pi, possibilitando a instalação do outro pacote adicional *CODESYS Control for Raspberry PI* através de comandos SSH.

Após instalado, é adicionado automaticamente ao Codesys uma nova configuração de gateway na porta 1217 da interface de rede padrão do computador. Nenhuma configuração adicional é necessária.

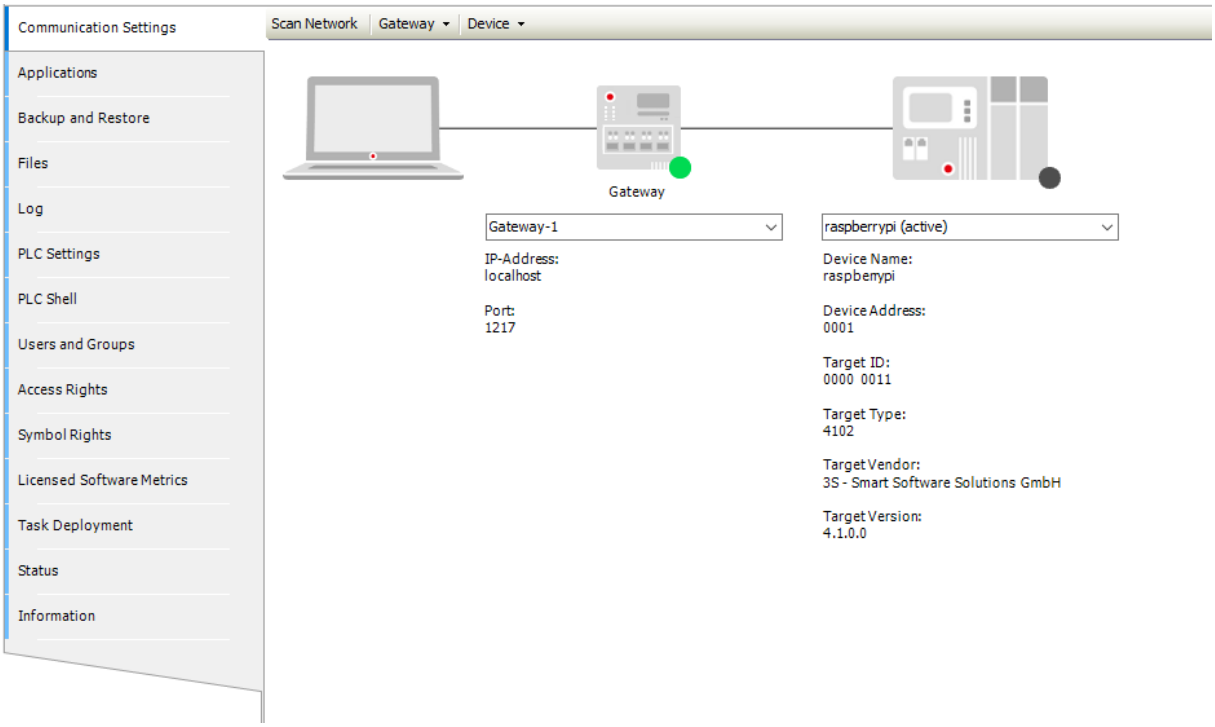


Figura 25 – Representação do gateway configurado pelo pacote *CODESYS Edge Gateway for Linux* no Codesys

Fonte: Do autor

CODESYS Control for Raspberry PI

Após instalado o gateway, já é possível acessar o Raspberry remotamente (com as configurações demonstradas na seção 2.3.2 já realizadas no Raspberry) e instalar o software de Runtime do pacote *CODESYS Control for Raspberry PI* no mesmo. Para isso, se utiliza a ferramenta *Update Raspberry Pi*, adicionada com a instalação do pacote adicional. Na ferramenta, deve-se preencher o nome de usuário e a senha configurados no Raspberry para que o acesso SSH seja autorizado.

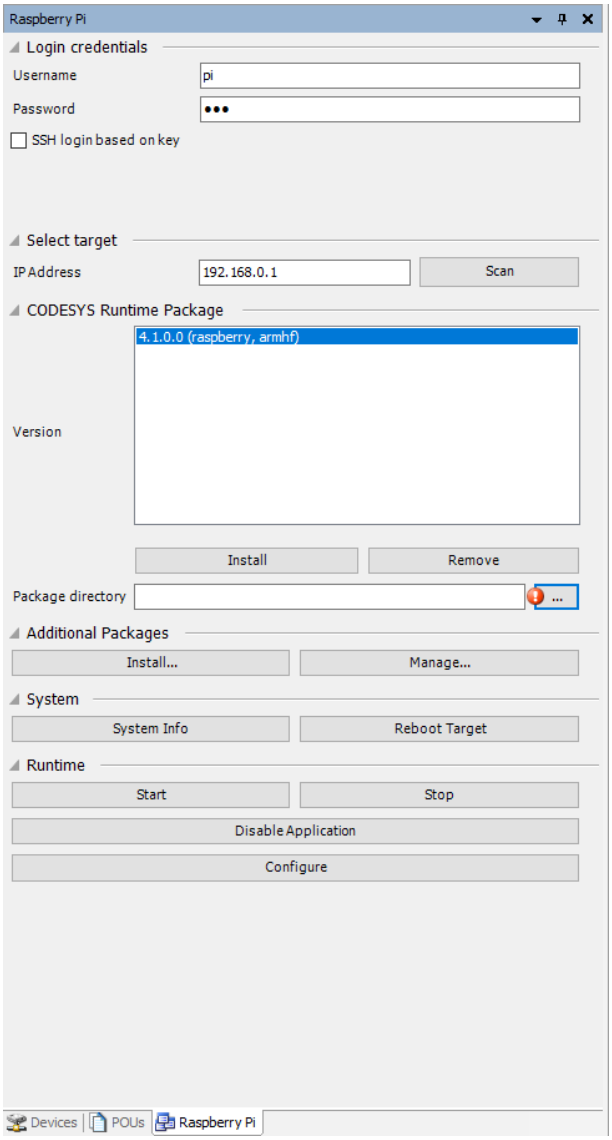


Figura 26 – Ferramenta *Update Raspberry Pi* no Codesys
Fonte: Do autor

Com tudo configurado, é possível agora instalar o módulo *CODESYS Runtime for Raspberry PI* presente no pacote adicional *CODESYS Control for Raspberry PI*, serviço esse responsável por transformar o Raspberry em um CLP podendo ser programado e monitorado por rede utilizando um computador com o Codesys (e os pacotes adicionais mencionados anteriormente), além de executar outros comandos de forma remota, como:

- Instalar pacotes adicionais no Raspberry;
- Requisitar informações do sistema operacional que esta sendo executado;
- Reiniciar o dispositivo;
- Iniciar ou parar a aplicação *runtime*;
- Desabilitar a aplicação;

- Configurar o modo da aplicação *runtime* para *Standard* ou *Multicore*.

Os dois modos de aplicação consistem na quantidade de núcleos de processadores que seram alocados para a execução do serviço *CODESYS Runtime for Raspberry PI*, enquanto o modo *Standard* limita o serviço a apenas um núcleo do processador, o modo *Multicore* irá utilizar quantos núcleos forem necessários dependendo do processamento demandado pela aplicação do usuário que será descarregado e executado no CLP.