UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

HIGOR PARIZI STRAZZI
LETICIA IARA DE SOUZA
LUCAS RADUY GOMES DE CAMARGO
SÉRGIO HENRIQUE ZANFORLIM FILHO

IDENTIFICAÇÃO E CONTROLE REMOTO DE UMA PLANTA DIDÁTICA DE VAZÃO E NÍVEL

CURITIBA

2022

HIGOR PARIZI STRAZZI
LETICIA IARA DE SOUZA
LUCAS RADUY GOMES DE CAMARGO
SÉRGIO HENRIQUE ZANFORLIM FILHO

IDENTIFICAÇÃO E CONTROLE REMOTO DE UMA PLANTA DIDÁTICA DE VAZÃO E NÍVEL

IDENTIFICATION AND REMOTE CONTROL OF A LEVEL AND FLOW TEACHING PLANT

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR)

Orientador: Prof. Dr. Thiago Alberto Rigo Passarin

Co-orientador: Prof. Dr. Alexandre José Tuoto Silveira Mello

CURITIBA

2022

HIGOR PARIZI STRAZZI
LETICIA IARA DE SOUZA
LUCAS RADUY GOMES DE CAMARGO
SÉRGIO HENRIQUE ZANFORLIM FILHO

IDENTIFICAÇÃO E CONTROLE REMOTO DE UMA PLANTA DIDÁTICA DE VAZÃO E NÍVEL

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Controle e Automação do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná (UTFPR)

Data de aprovação: 08 / 06 / 2022

Prof. Dr. Thiago Alberto Rigo Passarin
Universidade Tecnológica Federal do Paraná

Prof. Dr. Alexandre José Tuoto Silveira Mello
Universidade Tecnológica Federal do Paraná

Prof. Dr. Guilherme Luiz Moritz
Universidade Tecnológica Federal do Paraná

CURITIBA

2022

RESUMO

Este trabalho apresenta o desenvolvimento e implementação de um sistema de interfaceamento e controle para a planta didática de vazão e nível presente no laboratório CB-009 na Universidade Tecnológica Federal do Paraná. O trabalho possui três contribuições principais: a identificação do sistema, a disponibilização da planta para controle remoto via protocolo UDP e a implementação de um controle automático e manual com supervisão através do supervisório Elipse E3 e protocolo de rede industrial Modbus TCP. Tais tarefas são realizadas através do microcomputador Raspberry Pi 3. O desenvolvimento e implementação deste trabalho permite maior versatilidade da planta através da disponibilização de uma interface Ethernet para acesso remoto.

Palavras-chave: Identificação de sistema, Controle Multivariável, Redes Industriais

ABSTRACT

This work presents the development and implementation of an interface and control system for the didactic flow and level plant at the CB-009 laboratory in Universidade Tecnológica Federal do Paraná. This paper has three main contributions: the identification of the system, the availability of the plant for remote control via UDP protocol and, the implementation of an automatic and manual control with supervision through the Elipse E3 supervisory and Modbus TCP industrial network protocol. Such tasks are performed through the Raspberry Pi 3 microcomputer. The development and implementation of this paper allows major versatility of the plant through the availability of an Ethernet interface for remote access.

Keywords: Systems Identification, Multivariable Control, Industrial Networks

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ARX	Autorregressivo com entradas externas
CLP	Controlador Lógico Programável
CSMA/CD	Carrier-Sense Multiple Access With Collision Detection
ERR	Error reduction rate
ISO	International Organization for Standardization
LAN	Local Area Network
MAC	Media Access Control
MIMO	Multiple Input Multiple Output
MISO	Multiple-input single-output
MMQ	Método dos Mínimos Quadrados
MMQO	Método de mínimos quadrados ortogonais
MPC	Model predictive control
MSE	Mean Squared Error
NRMSE	Normalized Root Mean Square Error
OSI	Open System Interconnection
PID	Proporcional Integrativo Derivativo
RGA	Relative Gain Array
SCADA	Supervisory Control and Data Acquisition
SISO	Single-input single-output

SUMÁRIO

1 INTRODUÇÃO

1.1 TEMA

A teoria de controle é, no presente, altamente utilizada em diversas áreas e abordagens. Entretanto, para que essa teoria se tornasse multidisciplinar e uma vasta abordagem de pesquisa, foi necessária a passagem por períodos históricos caóticos, como a Primeira e a Segunda guerras mundiais. Dessa forma, por volta de 1960 percebeu-se que os métodos de controle utilizados não representavam, de forma acurada, a realidade. Devido a isso, passou-se a dar uma nova e melhor atenção para essa área (CARA; IRIONDO, 2003).

Como parte da teoria de controle, o método de identificação de sistemas foi desenvolvido e tornou-se um conjunto de métodos que podem ser aplicados em diversos processos reais em que as entradas e saídas podem ser medidas. Portanto, suas aplicações incluem processos industriais e sistemas de controle (BILLINGS, 2013).

A Identificação pode ser dividida em dois objetivos: o primeiro é feito por meio de aproximação, a qual, através de um conjunto de valores, busca-se um modelo aproximado que resulta em bons valores de previsão e baixo erro quadrático médio. Nestas aplicações a forma do modelo não é o mais importante. Essa abordagem pode ser aplicada em sistemas como, por exemplo, previsão climática, ações da bolsa de valores e classificação de padrões. Já o segundo objetivo está ligado com o desenvolvimento de modelos que reproduzam as características dinâmicas do sistema com o modelo mais simples possível, e que seja possível relacionar os componentes do modelo aos comportamentos do sistema em estudo (BILLINGS, 2013).

A identificação do sistema é a definição de um modelo matemático que permite a elaboração e aplicação de estratégias de controle. Dentro dos sistemas de controle, têm-se os sistemas multivariáveis, os quais baseiam-se em um modelo dinâmico do processo, sendo a precisão do modelo uma questão fundamental. Em muitos problemas os modelos não estão prontamente disponíveis, dificultando assim o processo de controle. Por este motivo, há um grande interesse na identificação do sistema ou processo, ou seja, o desenvolvimento de modelos dinâmicos empíricos a partir de de dados entrada-saída. Normalmente, o processo de

identificação do sistema é a etapa que mais demanda tempo na implementação industrial de estratégias avançadas de controle (KOZÁK, 2014).

Para a implementação industrial do controle pode-se utilizar a aplicação de uma rede industrial de comunicação. Redes industriais são protocolos de comunicação amplamente utilizados no ambiente industrial para controlar e supervisionar processos, através de um tráfego rápido e preciso de sinais entre os sensores, atuadores, controladores e demais componentes do chão de fábrica até o nível de gerenciamento de informação.

Seu surgimento se deu com o advento dos instrumentos digitais, onde surgiu-se a necessidade de algo que pudesse interligá-los, daí nasceu a ideia da criação e padronização de uma rede que pudesse ligar todos os equipamentos e disponibilizar todos os sinais do processo em um único meio físico. O desenvolvimento de um padrão internacional demandou muitos anos e encontros de grupos como a *International Society of Automation* (ISA), a *International Electrotechinical Comission* (IEC) e os comitês de padronização do PROFIBUS (norma alemã) e de padronização do FIP(norma francesa), que se uniram e fundaram o Comitê Internacional IEC/ISA SP50 *Fieldbus*.

Nos anos 2000, o comitê criou o *fieldbus* padrão IEC, denominado IEC 61158, composto por oito protocolos de comunicação distintos, que ainda não abrangiam todas as aplicações industriais. Desta forma, mais tarde, criou-se a IEC 61784, que corrigiu as especificações da IEC 61158 e definiu os chamados *"profiles"*, onde foram incluídos vários protocolos que utilizam o meio físico da Ethernet, bem como os protocolos IP, TCP e UDP.

1.1.1 DELIMITAÇÃO DO TEMA

O objetivo de controlar um sistema é fazer com que sua saída responda conforme o desejado quando se aplica a ele uma certa entrada, para isso é necessário controlar essa entrada. Portanto, para obter um bom controle de um sistema, é de suma importância que este sistema possua um modelo bem identificado e um interfaceamento com a planta disponível para a aplicação futura de variadas formas de controle.

Os métodos de identificação de sistemas serão aplicados a uma planta didática de vazão e nível localizada no laboratório CB-009 da Universidade Tecnológica Federal do Paraná campus Curitiba, afim de se obter o modelo matemático do sistema.

Outra abordagem será a disponibilização de uma estrutura para o futuro desenvolvimento de técnicas de controle, possibilitando assim aplicações em aulas de diferentes disciplinas da universidade, tais como Sistemas de Controle, Identificação de Sistemas e

Controladores Lógicos Programáveis.

1.2 PROBLEMAS E PREMISSAS

A teoria de controle é uma ciência baseada em modelos e o desempenho de um controlador é determinada pela acurácia do modelo em representar o sistema real (BENNETT, 2017). Sendo assim, em casos onde não é utilizado o correto método de identificação, o modelo identificado pode não atender a esses requisitos, muitas vezes devido a ruídos presentes nos valores medidos. Para este trabalho os possíveis problemas serão: desenvolver uma estrutura que utilize um microcomputador para captura e envio de dados, criar uma interface com a planta e conseguir encontrar um modelo adequado a partir da identificação do sistema.

Um microcomputador será utilizado para ler e controlar os sensores e atuadores da planta, adquirindo os dados necessários para o processo de identificação e posteriormente se estabelecerá a comunicação com os agentes externos à planta via protocolo de redes *Ethernet*. Este sistema de redes será utilizado para envio e recebimento de dados relevantes à operação da planta. Um dos problemas a serem resolvidos é viabilizar uma taxa de transmissão que seja compatível com a dinâmica do sistema, pois este é um fator determinante na possibilidade de realizar o controle remoto pelos futuros usuários da planta didática, uma vez que tem relação direta com a taxa de amostragem.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Desenvolver um sistema embarcado que realize as tarefas de geração e aquisição de sinais para identificação do sistema e disponibilização de acesso via rede aos sensores e atuadores da planta didática de vazão e nível.

1.3.2 OBJETIVOS ESPECÍFICOS

- Elaborar uma revisão com bases nas referências bibliográficas, sobre instrumentação industrial, identificação de sistemas e métodos de controle aplicáveis à planta didática;
- Desenvolver um sistema embarcado para que a planta didática seja controlada por outros dispositivos através de um protocolo de redes industriais, tal como gerar e ler sinais que serão utilizados na identificação.

- Definir o método de identificação do sistema da planta;
- Realizar a identificação do sistema;
- Estabelecer comunicação via protocolo de rede;
- Implementar um controlador PID;
- Elaborar uma interface de supervisão;

1.4 JUSTIFICATIVA

Atualmente a planta possui apenas dois controladores do tipo Proporcional Integrativo Derivativo (PID) descentralizado, implementados nos Controladores *Multi-Loop* CD600 *Plus* da empresa SMAR. Estes controladores permitem apenas a sintonia das três constantes de um controle PID e não permitem o uso de diferentes controladores.

Sabendo-se que a planta de vazão e nível é utilizada didaticamente em várias disciplinas da área de controle e automação e que o acesso dos alunos aos sensores e atuadores é limitado, pois seu controlador atual é dedicado, avistou-se a necessidade de um estudo e implementação de um novo sistema. Nesse sistema de rede o controlador passará a ser externo, abrindo um leque de possibilidades de atuação e aquisição de dados de sensores e atuadores, possibilitando assim a implementação de novas estratégias de controle.

Esse estudo e projeto tem como justificativa a implementação real dos conhecimentos adquiridos ao longo do curso, assim como o enriquecimento didático dos alunos que terão contato com a planta, visto que após a conclusão do trabalho a planta ficará disponível para que o usuário crie seu próprio sistema de identificação e/ou de controle, além de possibilitar o acesso à planta via rede.

1.5 PROCEDIMENTOS METODOLÓGICOS

O método de pesquisa e ação será utilizado para o desenvolvimento do trabalho. Esse método possui natureza empírica e é fortemente baseado em “pesquisa na ação”, ou seja, a pesquisa é participativa e simultânea à ação (COUGHLAN; COGHLAN, 2002). O discorrer do projeto será fundamentado nas três bases da pesquisa e ação. Tais bases consistem em, primeiramente, um estudo preliminar do contexto ao qual a pesquisa será aplicada, assim como suas justificativas. No segundo passo, tem-se a coleta de dados, o planejamento da ação, a implementação e avaliação, analisando assim se os resultados da implementação foram

satisfatórios ou será necessário uma nova aquisição de dados e ação. Por fim, o terceiro alicerce da pesquisa e ação consiste na verificação e monitoramento dos passos anteriores (DRESCH et al., 2015).

Vistos os fundamentos do método da pesquisa e ação, pode-se agora adequá-los ao tema deste trabalho.

1.6 ESTRUTURA DO TRABALHO

Esse trabalho está estruturado conforme a seguir. No Capítulo 1 é apresentada uma breve introdução ao tema. No Capítulo 2 apresenta-se a fundamentação e aprofundamento teórico em identificação de sistemas, sistemas de controle, redes industriais e protocolos, instrumentação, hardware e software utilizados. Já no Capítulo 3 são descritos todos os procedimentos práticos realizados. Em seguida, o Capítulo 4 mostra os resultados obtidos assim como as dificuldades e problemas encontrados durante o desenvolvimento desse trabalho. Por fim, o Capítulo 5 apresenta as conclusões obtidas e a discussão dos resultados, assim como as sugestões para trabalhos futuros envolvendo o tema.

2 REFERENCIAL TEÓRICO

2.1 IDENTIFICAÇÃO DE SISTEMAS

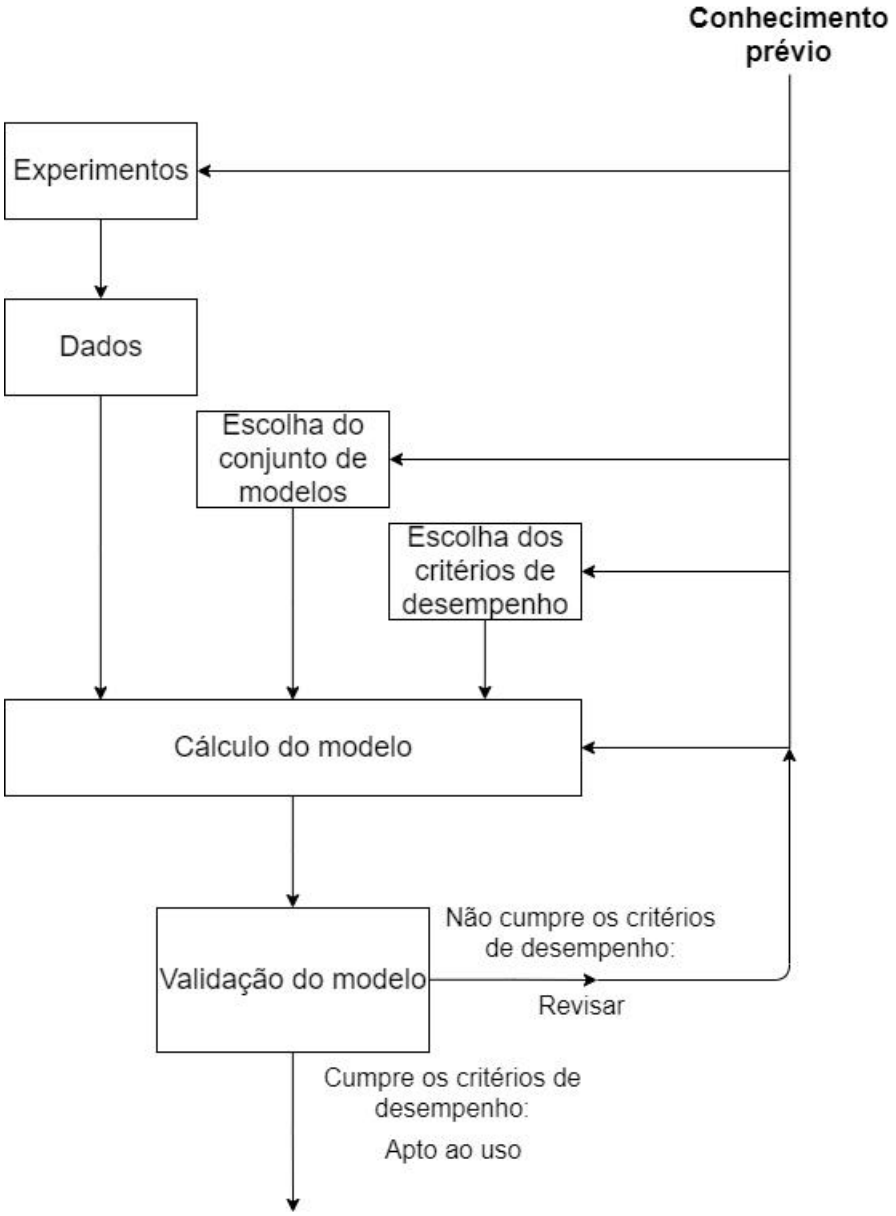
2.1.1 O PROCESSO DE IDENTIFICAÇÃO

O processo de identificação de um sistema baseia-se em três principais passos: coleta de dados, seleção de conjunto de modelos adequados e escolha do modelo. A coleta de dados é realizada durante um processo de identificação especificamente projetado. Os dados de entrada e saída são registrados durante esse processo e fica a critério do usuário quais sinais e em que momento medi-los, além de também decidir quais sinais de entrada irá utilizar de forma a tornar esses dados o mais informativos possíveis.

Após a etapa de coleta de dados tem-se a etapa mais importante e difícil do processo, o usuário precisará selecionar um conjunto de modelos adequados para o sistema específico. Esse conjunto de modelos pode ser obtido de três formas, a partir de uma modelagem minuciosa, que pode levar em consideração modelos lineares que não possuem referência à formação física do sistema, chamado de conjunto de modelos de caixa preta. Já a modelagem com referência a interpretação física do sistema pode ser chamada de caixa cinza, nesse caso, os parâmetros do conjunto são ajustáveis. E por fim, a modelagem caixa branca que também é regida pela interpretação física do sistema, porém não se faz necessário a realização de experimentos, tornando apenas necessário algumas medições de verificação e validação.

Terminada a etapa de seleção de conjuntos de modelos adequados ao sistema, é necessário escolher o melhor modelo, ou seja, aquele que apresenta o melhor desempenho ao reproduzir os dados medidos. Este é o método de identificação (LJUNG, 1999). É possível que o primeiro modelo escolhido não apresente o melhor desempenho, sendo assim, todos os passos devem ser revisados e refeitos, se necessário. A Figura 1 ilustra de forma resumida, mas objetiva, o processo de identificação de um sistema como um todo.

Figura 1: Etapas de identificação de um sistema



Fonte: adaptado de Ljung (1999).

2.1.2 MODELOS DE SISTEMAS

No processo de desenvolvimento de um sistema de controle é necessário conhecimento prévio da planta a ser controlada, este conhecimento pode ser expresso por um modelo que descreve as características por trás da dinâmica desta planta. Este modelo auxilia na escolha da estratégia de controle que melhor atende aos requisitos de desempenho definidos. Além do auxílio ele se torna parte necessária em certas abordagens, sendo integrado no controlador.

Uma forma de obtenção deste modelo é a modelagem caixa branca, onde um sistema

é dividido em subsistemas e estes possuem suas propriedades bem estabelecidas a partir de leis físicas e trabalhos empíricos já documentados. Em uma modelagem deste tipo não é necessário realizar experimentos com o sistema e também não é necessário o uso dos dados de entrada e saída, sendo utilizado apenas algumas medições básicas de atributos do sistema (LJUNG, 1999). Essa modelagem é mais abrangente e portanto, proporciona um entendimento maior do sistema como um todo, porém, por ser necessário um conhecimento maior das leis físicas e químicas que regem o sistema, o processo de modelagem não é tão fácil e rápido.

O objetivo da identificação de sistemas é criar modelos matemáticos que descrevem o sistema e que representam suas características dinâmicas de forma simples. Para a obtenção do modelo que atenda a estes requisitos, experimentos são realizados no sistema e os dados de entrada e saída são coletados e armazenados. Posteriormente estes dados são analisados em busca de um modelo matemático que possua a melhor representação, sendo umas das escolhas possíveis e mais utilizadas para critério de desempenho, o cálculo do erro quadrático médio, do inglês *Mean Squared Error* (MSE). Essa abordagem resulta em um modelo que não é diretamente mapeável através das leis físicas que regem o sistema, seus parâmetros são utilizados como forma de ajustes para melhor representar os conjuntos de dados, chamamos este um modelo caixa preta (AGUIRRE, 2015). Uma vantagem desta modelagem é que ela apenas requer os dados de entrada e saída para obtenção do sistema, mas como desvantagem, não há nenhum conhecimento prévio do sistema.

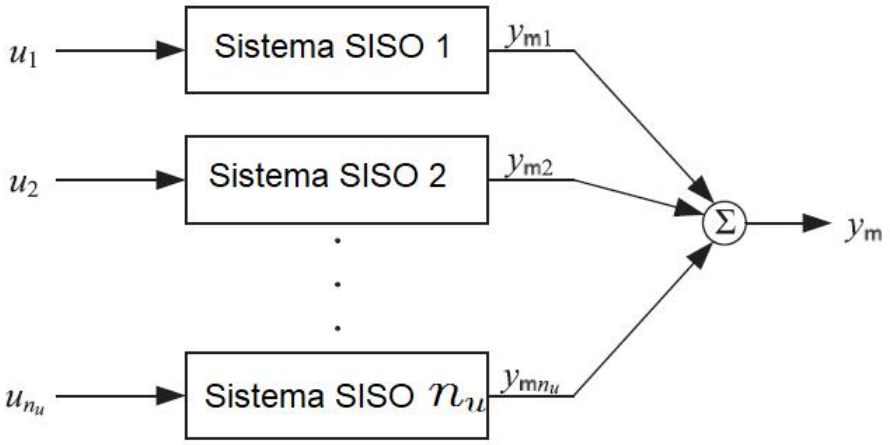
Em muitos casos é utilizado uma combinação dos dois métodos descritos acima, a qual ganha o nome de modelo caixa cinza. Nesse modelo é utilizada uma combinação do conhecimento prévio das dinâmicas do sistema para determinar a estrutura e alguns parâmetros do modelo, e são usados os dados de entrada e saída para a determinação dos parâmetros restantes e de dinâmicas não modeladas previamente. Por já possuir um conhecimento prévio do sistema, essa modelagem pode fornecer um modelo mais preciso porém, é necessário que o conjunto de dados seja representativo (CAMERON; HANGOS, 2001).

2.1.3 IDENTIFICAÇÃO DE UM SISTEMA MIMO

O processo de identificação de sistemas *Multiple Input Multiple Output* (MIMO) se torna mais simples quando este sistema é considerado como uma combinação linear de vários subsistemas *Multiple Input Single Output* (MISO) (KON et al., 2013). Dessa forma é necessário um subsistema MISO para cada variável de saída y. Esses subsistemas MISO podem ser novamente decompostos em vários sistemas *Single Input Single Output* (SISO), representado na Figura 2, assim as entradas dos subsistema MISO podem ser compostas por variáveis de

entrada *u* e variáveis de saída *y* de outros subsistemas.

Figura 2: Representação de um sistema MISO linear para identificação de modelos



Fonte: adaptado de Kon et al. (2013).

2.1.4 MÉTODOS DE IDENTIFICAÇÃO DE SISTEMAS

Os dois métodos mais conhecidos e que poderão ser utilizados no desenvolver desse trabalho são o Método dos Mínimos Quadrados (MMQ) e o Métodos dos Mínimos Quadrados Ortogonais(MMQO).

O MMQ é um método de otimização matemática que tem por finalidade gerar o que se chama em estatística de Regressão Linear. O método consiste em tentar reduzir ao máximo o resíduo ou erro da regressão, que é a diferença entre a soma dos quadrados das diferenças entre o valor estimado e os dados observados.

A utilização deste estimador tem como premissas que as variáveis sejam lineares entre si, ou seja, o modelo deve possuir parâmetros lineares e também que o erro ou distúrbio seja distribuído aleatoriamente com distribuição normal.

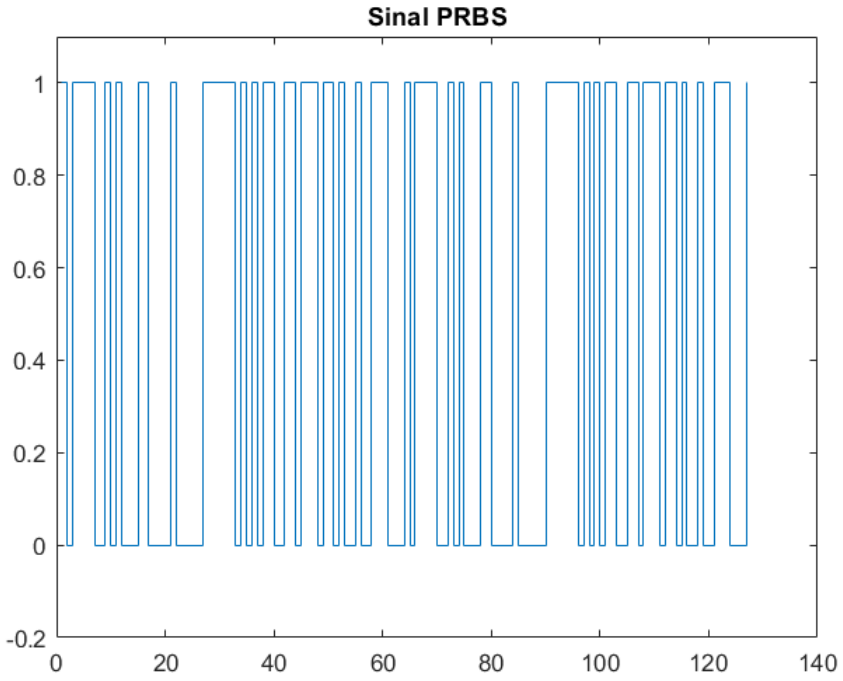
Já o MMQO tem como objetivo inicial servir de estimador de parâmetros para modelos não lineares. Sua ideia principal consiste em introduzir um modelo auxiliar com termos ortogonais sobre um conjunto de dados, portanto o coeficiente de modelos individuais pode ser estimado independentemente de outros termos desse modelo. A aplicação desse algoritmo permite observar a contribuição de cada termo na variação da saída do sistema.

Além dos métodos de identificação de sistemas, existem alguns sinais que são utilizados para auxiliar no processo de identificação. Um sinal de entrada amplamente utilizado é o *Pseudo-random binary sequences* (PRBS), justamente pela relativa simplicidade

de implementação e uma função de correlação semelhante ao ruído branco. O PRBS é um sinal periódico e determinístico que pode ser projetado para excitar a entrada, em uma frequência relevante para o sistema, em um único ciclo de dados. Além de fornecer ao usuário um meio para descartar dados corrompidos e reter dados mais importantes para a estimativa e validação do modelo. É gerado a a partir da equação de diferenças abaixo: (LJUNG, 1999)

$$u(t) = rem(A(q)u(t).2) = rem(a_n + u(t-1) + ... + a_n u(t-n).2). \quad (1)$$

Figura 3: Sinal PRBS



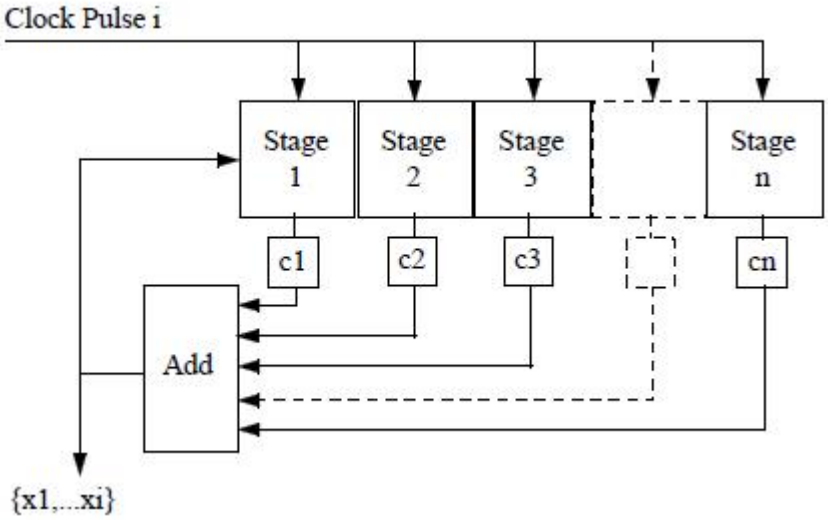
Fonte: Autoria própria.

Entretanto, o sinal PRBS nem sempre é adequado para problemas não lineares, uma vez que ele consiste em apenas dois níveis, conforme mostrado na Figura 3. Além de que os dados obtidos podem não fornecer informações suficientes para a identificação do comportamento linear. Nesses casos torna-se mais interessante a utilização do sinal *Pseudo-random multi level sequences* (PRMLS) (BRAUN et al., 1999), que permite ao usuário evidenciar o comportamento do sistema não linear enquanto manipula o conteúdo harmônico do sinal para permitir a estimativa da dinâmica linear na presença de não linearidades.

Semelhante ao PRBS, o sinal PRMLS é um sinal determinístico, periódico e possui sua função de correlação semelhante a do ruído branco, esta correlação é visualizada dentro de um período do sinal. Este sinal é gerado a partir de um registrador de deslocamento e adição

de módulo, com base na teoria de campos finitos. Um campo finito, chamado de Campo de Galois, definido por *q* elementos, orienta a criação de uma sequência pseudo-aleatória *x1, x2, ... xi*, a partir de um registrador de deslocamento. Os coeficientes das saídas desse registrador de deslocamento são derivados de um polinômio *f(x)* em *GF(q)*. A Figura 4 ilustra a configuração do registrador de deslocamento para a criação do sinal PRMLS.

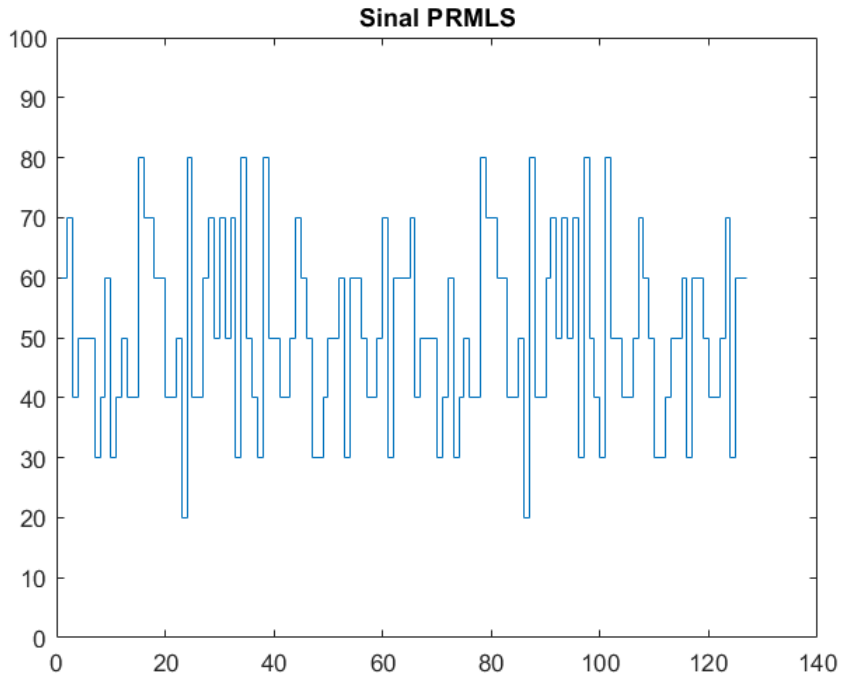
Figura 4: Algorítimo de geração do sinal PRMLS



Fonte:Braun et al. (1999)

Após a criação da sequência, ela é então mapeada para os níveis relativos da entrada com base no campo finito escolhido, o múltiplo de harmônicos a serem suprimidos e o número de termos do possível modelo. Depois do sinal mapeado, dimensiona-se-o com uma amplitude aceitável e cada elemento na sequência mapeada é introduzido na planta, que deseja identificar, após um tempo de comutação predeterminado *T* ter ocorrido. O usuário necessita ainda, escolher criteriosamente os parâmetros do sistema ou projeto para a geração do sinal. Na Figura 5 é representado um sinal PRMLS.

Figura 5: Sinal PRMLS



Fonte: Autoria própria.

2.1.5 MSE

A sigla MSE vem do inglês *Mean Squared Error*, que significa Erro Quadrático Médio, com esse artifício é possível medir o ajuste do modelo ao sistema real. Um MSE igual a zero informa que ambos os dados são coincidentes, ou seja que não há erro entre modelo e sistema, portanto um bom modelo, idealmente, tem valores de MSE próximos a zero. A equação para o cálculo do MSE é dada a seguir.

$$MSE = \frac{\sum(y_i - Y_i)^2}{n} \tag{2}$$

Em que y_i é o valor real observado, Y_i o valor do modelo e n a quantidade de pontos observados. (JAMES et al., 2013)

2.1.6 NRMSE

O NRMSE vem do inglês *Normalized Root Mean Square Error*, que significa Erro de raiz quadrada média normalizada, está variação do MSE é obtida ao normalizar a raiz quadrada do MSE, essa variação do MSE permite a comparação do erro entre modelos com diferentes

escalas (ZAWBAA et al., 2016), e pode ser calculado a partir da seguinte equação,

$$NRMSE = \frac{\sqrt{MSE}}{y_{max} - y_{min}}.100 \tag{3}$$

em que y_{max} e y_{min} são o maior e menor erro, respectivamente.

2.1.7 SISTEMAS DE CONTROLE

Os sistemas de controle tornaram-se peças fundamentais no controle de processos industriais devido ao avanço das mudanças econômicas, da ampla concorrência e do desenvolvimento de novas tecnologias e regulamentos mais severos referente a qualidade e segurança. O controle de alto desempenho e os processos de medição juntamente com a modelagem e a otimização de processos, tornaram-se partes essenciais de plantas industriais, pois desenvolvem processos mais flexíveis e mais complexos que são imprescindíveis para o sucesso de processos fabris.

Basicamente, um sistema de controle atua controlando as saídas do sistema de uma maneira pré-estabelecida, através de entradas previamente definidas. Os controles são comumente classificados entre malha aberta ou malha fechada. Um sistema de controle de malha aberta é aquele em que a saída não é medida e nem realimentada em relação a entrada, ou seja, o sinal de saída não exerce nenhum controle sobre o sistema. Normalmente, tomam-se ações baseadas em tempo e não há uma verificação da ação de saída. Já em sistema de controle de malha fechada ou de realimentação, o sinal do erro (a diferença entre o sinal de entrada e o de realimentação) realimenta o controlador. Essa configuração diminui o erro do sistema e faz com que o sistema seja menos sensível a distúrbios porém, com tendência a instabilidade (OGATA, 2014).

Para a modelagem de sistemas de controle faz-se necessário o uso de ferramentas matemáticas tais como funções de transferência e modelagem em espaço de estados. A função de transferência é capaz de determinar o comportamento dinâmico do processo após alterações em sua variável de entrada e possui um papel importantíssimo no projeto e análise de sistemas de controle. Este modelo de função é capaz de expressar a equação diferencial que relaciona duas variáveis do processo, uma variável dependente ou de entrada e uma variável independente ou de saída (SEBORG et al., 2010). A representação algébrica da função de transferência é dada

pela equação (4),

$$G(s) = \frac{\mathscr{L}[saída]}{\mathscr{L}[entrada]} = \frac{Y(s)}{X(s)} \tag{4}$$

em que $\mathscr{L}$ denota a Transformada de Laplace.

Já a modelagem em espaço de estados, diferentemente da função de transferência, envolve três tipos de variáveis: variáveis de entrada, saída e de estado. As variáveis de estado são o menor conjunto de variáveis capaz de determinar o estado de um sistema dinâmico. A quantidade de variáveis dinâmicas presentes nesse sistema é igual ao número de integradores presente nesses mesmo sistema. Sendo assim, a modelagem em espaço estados pode ser descrita matematicamente pela equação (5), onde A é a matriz de estado, B a matriz de entrada, C a matriz de saída e D a matriz de transmissão direta para um sistema invariante no tempo.

$$\begin{aligned} x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{5}$$

Função de transferência e espaço de estados são essenciais para a modelagem de sistemas SISO, MIMO e MISO. SISO é um sistema escalar. Já o MIMO é um sistema vetorial, se uma alteração é realizada em uma das variáveis de entrada, isto afeta diretamente todas as saídas, ou seja, há uma interação entre saídas e entradas. Sua função de transferência básica é dada por

$$y(s) = G(s)u(s) \tag{6}$$

onde u é um vetor de entradas e G é a matriz da função de transferência.

2.1.8 O CASO SISO

Em sistemas SISO o controlador PID é muito utilizado em processos industriais e está presente na maioria dos dispositivos industriais de controle como os controladores lógicos programáveis (CLP). A utilidade desse controlador consiste na aplicação devida da soma de três ações de controle, o proporcional, integral e derivativo. O controle proporcional é proporcional ao erro do sistema. Já o controle integral é proporcional à integral do erro e se relaciona com valores passados do mesmo. Por fim, a ação derivativa baseia-se na derivada temporal do erro, que fornece uma estimativa de tendência do sinal para valores no futuro próximo (VISIOLI, 2006).

A saída do controlador é dada pela equação

$$U(s) = (K_p + \frac{K_i}{s} + K_d s)E(s) \tag{7}$$

O controlador PID descrito pela equação (7) descreve o controle em tempo contínuo, entretanto para a aplicação em equipamentos de controle digital é necessário a realização da discretização dela (NISE; SILVA, 2002). A análise do sistema discretizado se dá pela transformada Z, definida por:

$$F(z) = \sum_{k=0}^{\infty} f(kT)z^{-k} \tag{8}$$

em que $z = e^{Ts}$, assim, aplicando o método trapezoidal de discretização na equação (7), obtém-se:

$$\frac{U(z)}{E(z)} = K_p + \frac{1}{K_i}\frac{T}{2}\frac{z+1}{z-1} + K_d\frac{z-1}{Tz} \tag{9}$$

Essa equação apresenta o controle PID discreto no domínio Z em sua forma mais simples, entretanto, muitas vezes em sistemas reais os ruídos presentes na aquisição traz a necessidade da adição de filtros passa-baixa para a parcela derivativa. A partir da equação (9) é possível aplicar a transformada Z inversa para obter a equação na forma de tempo discreto.

$$U(z) = Kp.E(k) + K_i\frac{T}{2}[E(k) + E(k-1)] + U(k-1) + \frac{K_d}{T}[E(k) - (k-1)] \tag{10}$$

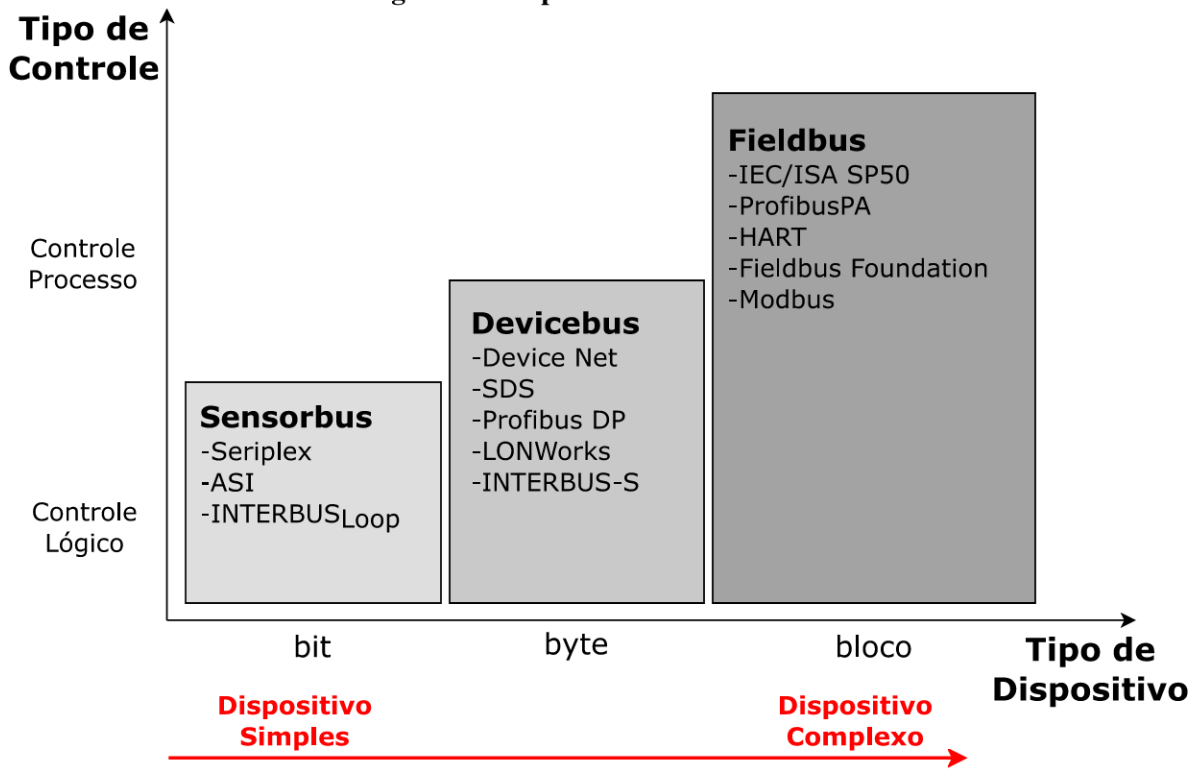
Dessa maneira é possível a implementação digital do controlador PID.

2.1.9 REDES INDUSTRIAIS

As redes industriais surgiram com o intuito de substituir o sistema tradicional de conexões ponto a ponto, na qual cada ponto de entrada e saída é conectado diretamente ao CLP. O sistema tradicional possui um alto custo de implementação, devido a quantidade de entradas e saídas necessárias no CLP e quantidade de cabos, além da dificuldade no diagnóstico de falhas. As redes industriais permitem diversas soluções para estes problemas do sistema tradicional, sendo que cada protocolo de rede possui uma série de características que devem ser comparadas no momento da seleção da rede a ser utilizado em um projeto. Os diferentes tipos de protocolos de redes industriais podem ser divididos em três grandes grupos, em relação à

quantidade de informação transmitida e o tipo de controle a ser realizado, estes são o *Fieldbus*, *Devicebus* e *Sensorbus* (LUGLI; SANTOS, 2010). A figura 6 expõe os três grupos conforme a relação de quantidade de informação e a categoria de controle em que cada protocolo se encaixa. Protocolos que trabalham a nível de bit são utilizados para controle lógico em aplicações que necessitam de velocidade, já os protocolos que transportam informações maiores são utilizados para controle de processo, onde a quantidade de informações transportadas é maior e a velocidade pode ser menor.

Figura 6: Grupos de redes industriais



Fonte: Lugli e Santos (2010).

O Modelo OSI, *Open System Interconnection*, é composto por sete camadas e é usado para descrever a maneira que deve ocorrer a interação entre protocolos e aplicações em dispositivos presentes em uma rede. As camadas do modelo OSI são organizadas como uma pilha, sendo a camada um a base, e cada camada realiza serviços para a camada diretamente superior. Segundo Briscoe (2000) as sete camadas do modelo OSI são:

- A primeira camada é Física, que determina as características elétricas e os meios de transmissão além das características físicas da rede.
- A Segunda camada é conhecida como Enlace e define as estratégias de acesso para o compartilhamento do meio físico.

- A terceira camada, a de Rede, fornece um meio de comunicação em sistemas abertos. Pode estabelecer, manter ou encerrar as conexões de redes estabelecidas.
- A quarta camada é conhecida como camada de transporte, tem a função de garantir a integridade e confiabilidade dos dados da camada cinco.
- A quinta camada, de sessão, estabelece uma comunicação entre dois *hosts* permitindo a troca de dados entre si.
- A sexta camada, de apresentação, se responsabiliza por realizar a tradução de dados da camada de aplicação para pacotes a serem utilizados pelas camadas inferiores e vice-versa, criptografar e descriptografar, e conversões de protocolos.
- A sétima e última camada, conhecida como camada de aplicação, onde encontra-se o usuário final e garante a interação homem-máquina.

Em uma comunicação que possui as sete camadas do modelo OSI, a informação é gerada na sétima camada pelo usuário no dispositivo transmissor e essa informação percorre as camadas inferiores. Cada camada acrescenta informações no início e fim desse pacote. Após a transmissão do pacote de informações no meio físico para o dispositivo receptor, essa informação passa pelo processo reverso, no qual o pacote escala as camadas partindo da primeira até a sétima, então os dados acrescentados pelas camadas são removidos e o segundo usuário recebe apenas a informação criada pelo primeiro usuário.

O padrão de comunicação Ethernet, IEEE 802.3, é a tecnologia de rede local (LAN), do inglês *Local Area Network*, mais utilizada atualmente, este padrão define as duas primeiras camadas do modelo OSI. Os meios físicos adotados são os cabos de par trançado e fibra óptica. E a segunda camada do modelo OSI no padrão IEEE 802.3 é governada pelo algoritmo de detecção de colisão *Carrier-Sense Multiple Access With Collision Detection* (CSMA/CD) e o endereço MAC, do inglês *Media Access Control*, é um identificador único de cada equipamento e identificá o transmissor e receptor de cada pacote de dados (SOMMER et al., 2010).

2.1.9.1 PROTOCOLO MODBUS

O protocolo Modbus foi desenvolvido pela Modicon em 1979 para a troca de dados entre equipamentos industriais em aplicações cliente-servidor e se tornou um protocolo aberto gerido pela Modbus Organization, uma organização independente e sem fins lucrativos. O fato de suas especificações serem publicas, tornou o Modbus o protocolo de comunicação

padrão da industria para comunicação entre dispositivos de diversas marcas (MODBUS ORGANIZATION, 2020).

O protocolo Modbus se encaixa na sétima camada do modelo OSI, a camada de aplicação, ou seja, ele não especifica um meio físico, dessa forma ele é compatível com qualquer protocolo existente para as camadas inferiores do modelo OSI, assim possibilitando o uso em diversos meios físicos, como EIA/TIA-232, EIA/TIA-485 e Ethernet via TCP/IP (KUPHALDT, 2008).

Este protocolo consiste em um conjunto de funções que realizam ações de leitura e escrita de dados em equipamentos industriais, e segue um esquema de arbitragem mestre-escravo, ou cliente-servidor no caso do Modbus TCP/IP. O dispositivo mestre, ou cliente, envia uma mensagem para equipamento escravo, ou servidor, com um código Modbus e as informações necessárias para que o equipamento escravo realize a ação. Os códigos de função Modbus atribuem ações de leitura e escrita, individual ou múltipla de variáveis discretas ou analógicas. Essas variáveis são distribuídos em quatro grupos no protocolo Modbus, essas variáveis podem ser discretas com tamanho de 1 *bit* ou analógicas, com 16 *bits* de tamanho. Na Tabela 1 é representado os tipos de variáveis descritos pelo protocolo Modbus, as faixas de endereços de cada grupo, suas ações e os códigos que podem ser utilizados em cada grupo de variável.

Tabela 1: Funções e endereços Modbus

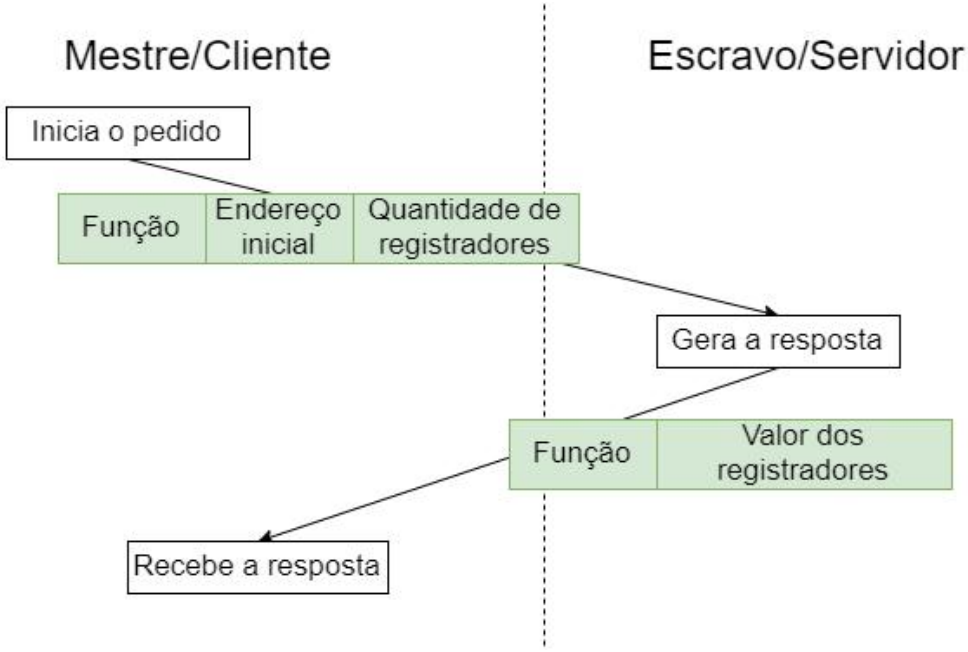
Código Modbus	**Faixa de endereços**	**Tipo**	**Ação**
01, 05, 15	00001 até 09999	*Coils*, saída discreta	Leitura/Escrita
02	10001 até 19999	*Discrete input*, entrada discreta	Apenas leitura
04	30001 até 39999	*Input register*, entrada analógica	Apenas leitura
03, 06, 16	40001 até 49999	*Holding register*, variável analógica	Leitura/Escrita

Fonte: adaptado de Kuphaldt (2008).

Uma representação básica do funcionamento da comunicação Modbus TCP/IP sem a ocorrência de erros é exibida na Figura 7, nesse exemplo é representado um pedido de leitura de alguns registradores enviado pelo dispositivo mestre ao escravo e sua resposta. Em um pedido de leitura é enviado o código da função de leitura, o endereço do primeiro registrador a ser lido e a quantidade de registradores a serem lidos. Em um pedido de escrita é adicionado os valores a serem escritos em um pacote semelhante ao de leitura de dados.

Porém o protocolo Modbus TCP/IP não é considerado um protocolo de Ethernet Industrial, pois ele utiliza o padrão *Ethernet* e está acima das camadas dos protocolos TCP, IP e MAC, o qual utiliza o algoritmo CSMA/CD e deixa de ser um protocolo de rede de tempo-real.

Figura 7: Exemplo de comunicação Modbus TCP/IP sem a ocorrência de erros



Fonte: Autoria própria.

Dessa forma o protocolo Modbus TCP/IP não suporta comunicação determinística, garantia de pontualidade das informações e troca eficiente de pequenos pacotes de dados, portanto o protocolo Modbus TCP/IP é indicado para monitoramento de processos e casos onde tempos de entrega em torno de 100 ms sejam toleráveis (KASBERGER, 2011).

2.1.9.2 PROTOCOLO UDP

O protocolo de datagrama do usuário, conhecido como UDP (do inglês *User Datagram Protocol*), se encaixa na quarta camada do modelo OSI, a camada de transporte. Esse protocolo possui um método de comunicação na qual a transferência da informação ocorre entre o remetente e o destinatário em que os dados são transmitidos sem a garantia de que o destinatário esteja pronto para receber, assim não há a confirmação de recebimento. Uma característica positiva do UDP é que sua utilização é muito eficaz na necessidade de transmissão de dados de maneira rápida. Em contrapartida, não há garantia que esses dados cheguem da maneira desejada (pode haver perda de informações e duplicação de pacote), em outras palavras, o UDP não possui grande confiabilidade.

O Protocolo UDP é uma alternativa ao protocolo TCP, ele promove acesso direto ao serviço básico de IP, cedendo acesso direto ao serviço de entrega de datagramas, tendo como porta de origem e de destino 16 bits. A principal diferença entre esse protocolo para o TCP é que o TCP é orientado a conexão, exigindo regras e mecanismos para que essa conexão seja

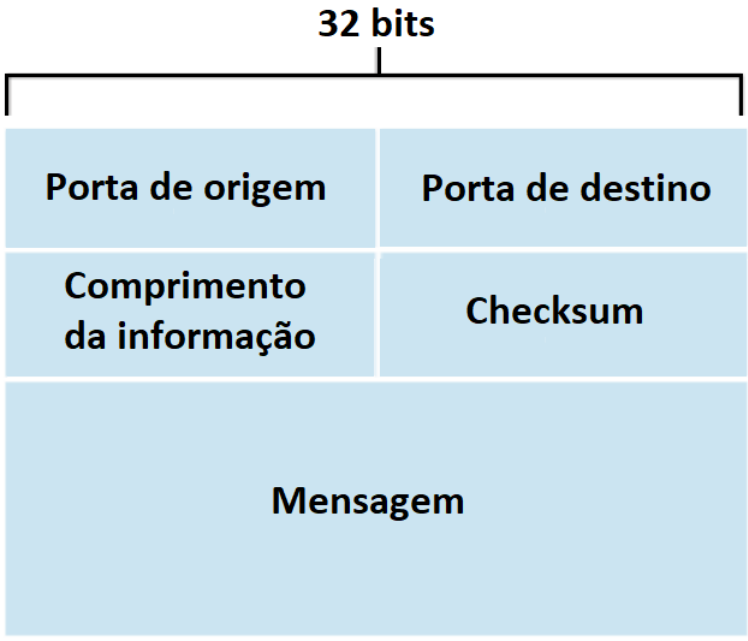
estabelecida, facilitando monitorar o fluxo de informações, detectar e corrigir erros presentes. O UDP por sua vez transmite dados de menor sensibilidade, em que a perda de parte dessa informação não seja altamente prejudicial para o serviço. (THAMMADI, 2011).

A estrutura do UDP é bem simples em comparação aos outros procolos como o TCP e é composto por quatro partes:

- Porta de origem.
- Porta de destino.
- Comprimento da informação.
- *Checksum*

Em que a porta de origem e o *checksum* não são campos obrigatórios, a estrutura do UDP é composta conforme a Figura 8.

Figura 8: Estrutura UDP



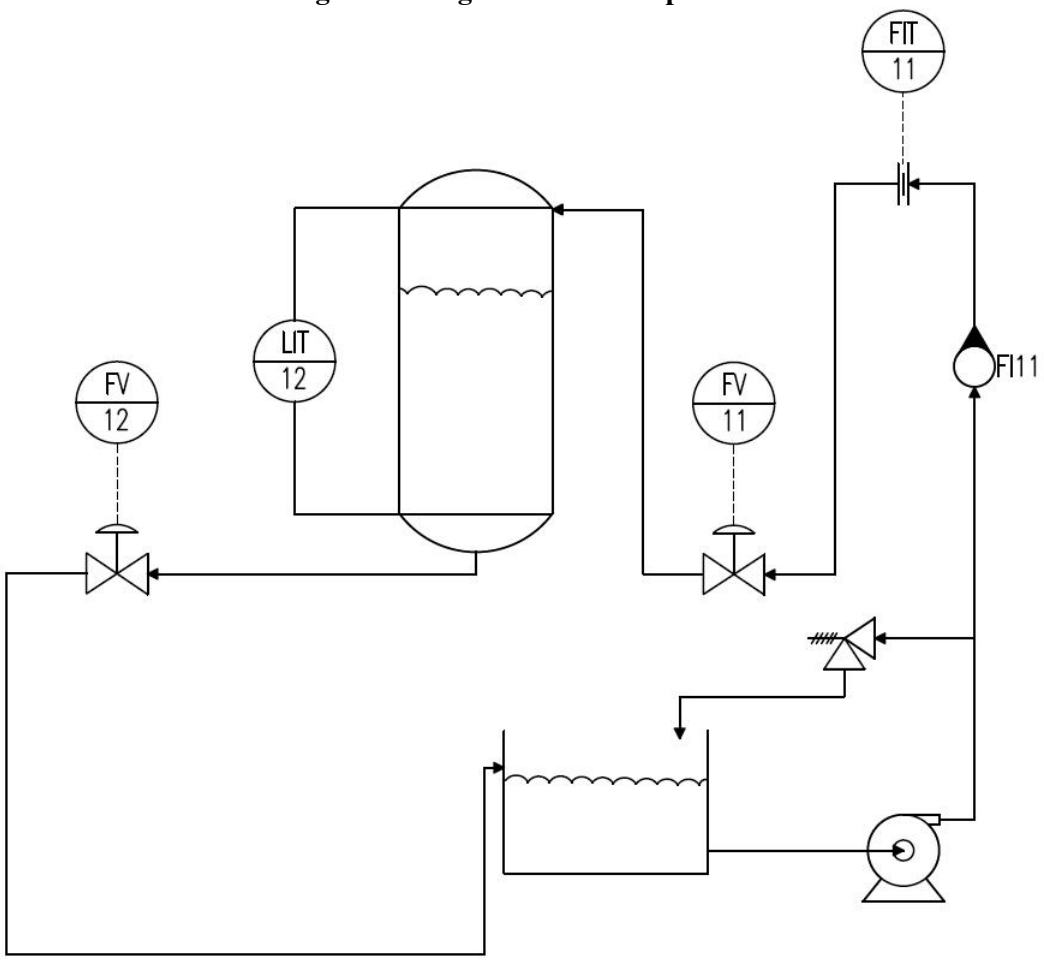
Fonte: adaptado de Kurose e Ross (2010)

2.2 INSTRUMENTAÇÃO DA PLANTA

2.2.1 PLANTA DIDÁTICA

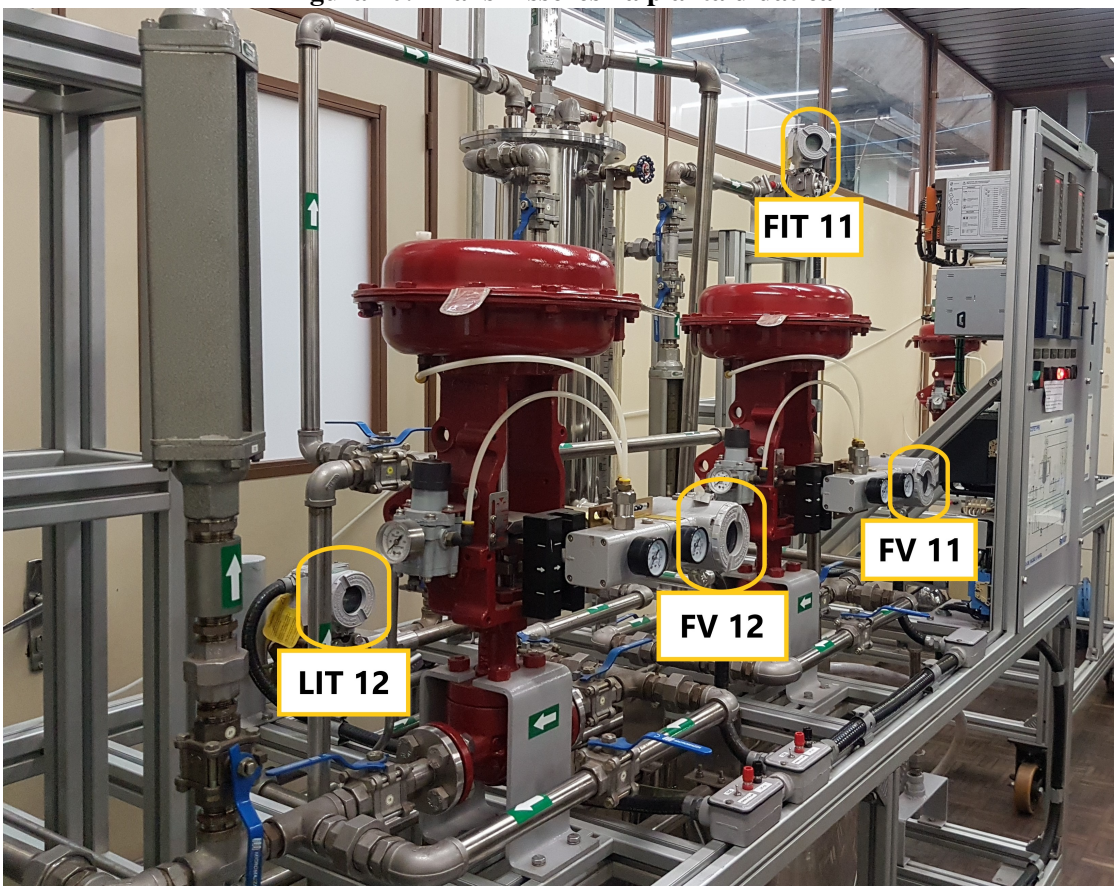
A planta didática a ser utilizada neste trabalho é uma variação das plantas didáticas *PD3 Series* da *Smar Technology Company* e está presente no laboratório CB-009 da UTFPR - Curitiba. Toda a instrumentação da planta didática é composta por equipamentos utilizados no meio industrial, dentre os presentes no conjunto, alguns serão cruciais para a aplicação do controle multivariável na planta. Visto que estes equipamento são de suma importância, eles serão descritos em mais detalhes a seguir.

Figura 9: Diagrama P&ID da planta.



Fonte: Autoria Própria.

Figura 10: Transmissores na planta didática



Fonte: Autoria Própria.

Na planta didática são realizadas medições das grandezas de vazão e nível e controle do posicionamento de válvulas através dos transmissores da fabricante *Smar Technology Company*. Para aquisição dos dados de vazão e nível foi utilizado os transmissores indicadores de vazão e nível FIT 11 (*Flow indicator trasmiter*) e LIT 12 (*Level indicator trasmiter*) respectivamente, modelo LD400, mostrados nas Figuras 9 e 10. Já para o controle dessas variáveis a planta conta com os posicionadores de válvulas de vazão FV 11 (*Flow valve*) e FV 12, modelo FY301, também destacados nas figuras.

Vazão e nível são grandezas frequentemente utilizadas em diversas industrias e podem ser aferidas com diferentes tipos de sensores. A planta é equipada com transmissores de pressão diferencial, tanto na medição de vazão como na medição de nível, que são compostos por um elemento transdutor, o qual transforma um sinal de pressão, no seu diafragma, em um sinal elétrico.

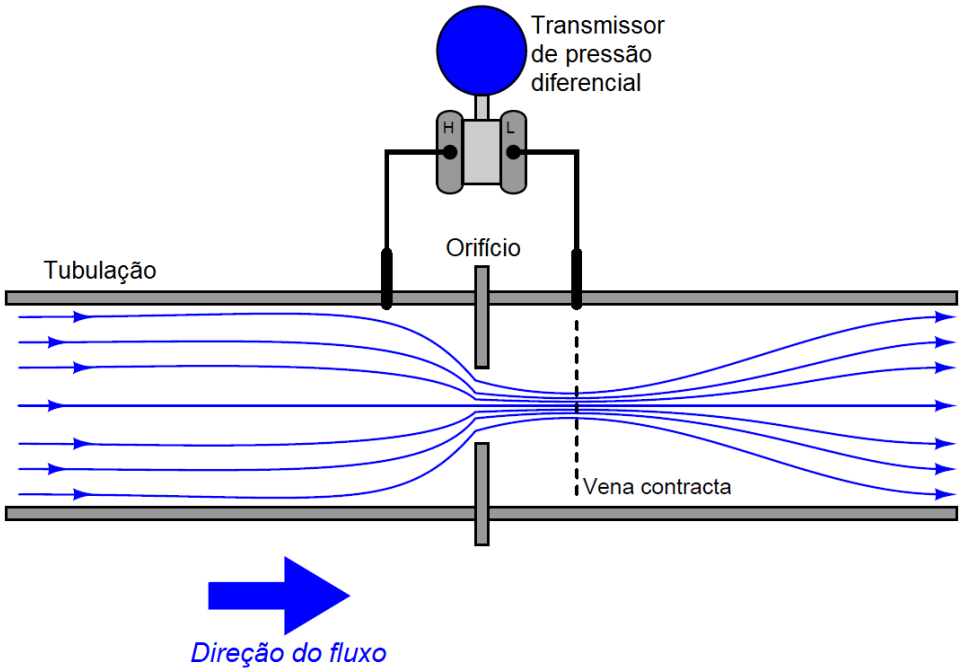
Esse sinal corresponde à diferença de pressão entre as duas câmaras de pressão do transdutor. Após o sinal de pressão ser convertido em um sinal elétrico, ele é transmitido como

um sinal analógico de tensão, de corrente ou como um sinal digital em uma rede industrial, porém este sinal é usualmente transmitido como sinal de corrente entre 4 a 20mA;

2.2.1.1 MEDIÇÃO DE VAZÃO

O uso de transmissores de pressão diferencial na industria é bastante comum, esse princípio utiliza uma placa fina de metal com um orifício, de dimensão e posição conhecida, que é posicionada dentro da tubulação no ponto onde será realizado a medição. Um transmissor de pressão diferencial é posicionado de forma a medir a queda de pressão na placa de orifício. Quando um fluido transita nessa tubulação ele sofre uma queda de pressão ao passar pela placa de orifício, essa queda de pressão é relacionada com a vazão e a densidade desse fluido (KUPHALDT, 2008).

Figura 11: Transmissor de pressão diferencial e placa de orifício em uma tubulação



Fonte: adaptado de Kuphaldt (2008).

Na equação (11) Q representa a vazão, k uma constante relacionada a correções e área da tubulação, Δp a pressão diferencial e por fim ρ é a densidade do fluido.

$$Q = k\sqrt{\frac{\Delta p}{\rho}} \tag{11}$$

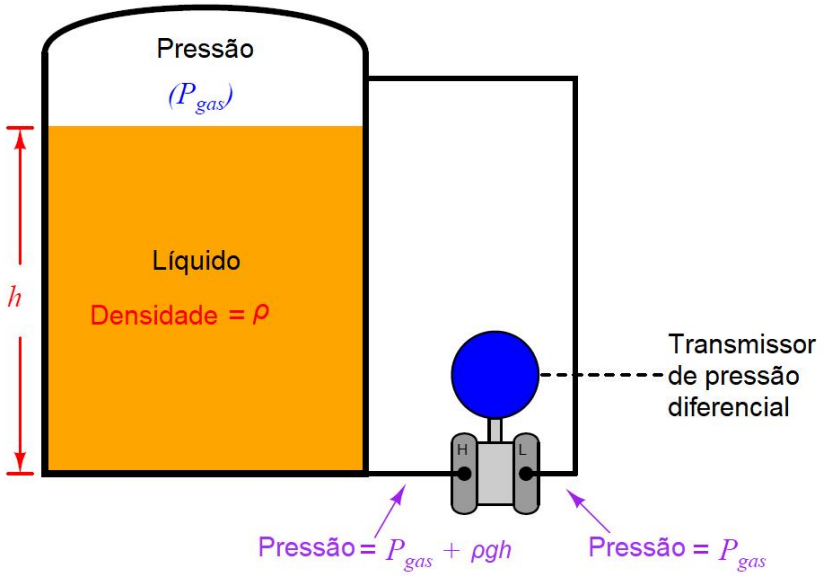
A equação (11) é uma versão reduzida da Equação de Bernoulli, que representa a

vazão em termos de diferença de pressão na placa de orifício e a densidade do fluido. Esta técnica de medição de vazão oferece alguns benefícios como: confiabilidade a longo prazo; operação simples; não possui partes móveis; baixo custo. (BOYES, 2009).

2.2.1.2 MEDIÇÃO DE NÍVEL

Transmissores de pressão diferencial também são utilizados para a medição indireta do nível. Este equipamento possibilita inferir nível de líquidos em tanques com pressão positiva além da coluna de líquido a ser medida. Para realizar a inferência do nível de líquido presente no tanque o transmissor é posicionado na mesma altura de nível zero do tanque, a câmara H de alta pressão realiza a medição de pressão no fundo do tanque, e a câmara L de baixa pressão realiza a medição no topo do tanque.

Figura 12: Medição indireta de nível com transmissor de pressão diferencial



Fonte: adaptado de Kuphaldt (2008).

Nesta configuração o sinal de pressão presente nas câmaras do transmissor pode ser expresso pela diferença entre a pressão no fundo do tanque e a pressão no topo do tanque, conforme apresentado na equação (12).

$$\Delta P = \underbrace{(P_{gas} + \rho g h)}_{\text{Câmara H}} - \underbrace{P_{gas}}_{\text{Câmara L}} = \rho g h, \tag{12}$$

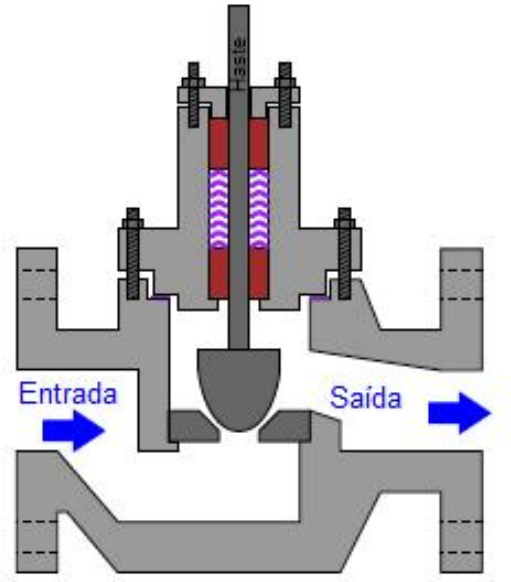
Portanto o sinal de saída do transmissor é proporcional a densidade do líquido e altura da coluna de líquido, dessa forma a pressão exercida pelo gás no tanque se torna irrelevante e o sinal transmitido representa apenas o nível do tanque (KUPHALDT, 2008).

2.2.1.3 VÁLVULAS DE CONTROLE

As válvulas de controle são instrumentos amplamente utilizados em sistemas de controle industrial. É possível classificá-las em dois grandes grupos: válvulas de controle discretas (Liga/Desliga) e as válvulas de controle variáveis, que são responsáveis por controlar a vazão do fluído de forma gradual (KUPHALDT, 2008).

A planta utilizada neste trabalho possui duas válvulas de controle variável do tipo globo, uma responsável pelo controle da vazão de entrada no tanque de nível, e a outra responsável pelo controle da vazão de saída do mesmo.

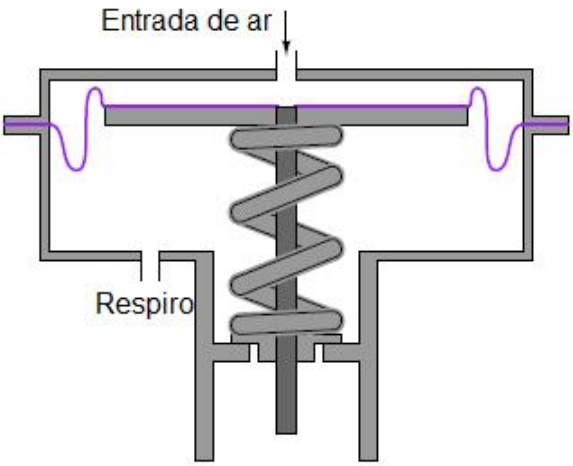
Figura 13: Válvula de controle tipo globo



Fonte: adaptado de Kuphaldt (2008).

Essas possuem atuadores pneumáticos que aplicam pressão em um diafragma, ilustrado na Figura 14, na cabeça da válvula (parte superior do instrumento) e este transfere movimento para a haste de comando esboçado na Figura 13.

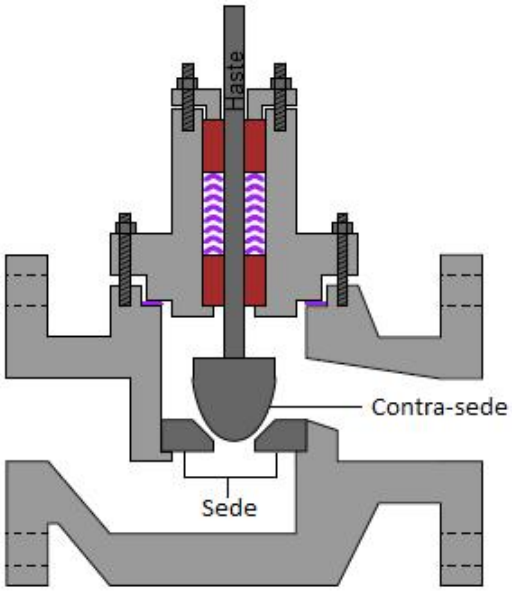
Figura 14: Atuador pneumático com diafragma



Fonte: adaptado de Kuphaldt (2008).

As válvulas globo atuam no fluxo através de duas partes principais de sua construção, uma parte móvel (contra-sede) e uma parte fixa (sede), a distância entre as partes, mostrada na Figura 15, permite o controle do fluxo no processo (KUPHALDT, 2008).

Figura 15: Sede e Contra-sede



Fonte: adaptado de Kuphaldt (2008).

2.2.1.4 VÁLVULAS DE ALÍVIO DE PRESSÃO

As válvulas de alívio de pressão são exemplos de válvulas de segurança, que atuam como à prova de falhas, muito utilizadas em indústrias químicas. Esse tipo de válvula funciona

como dispositivo de proteção, a fim de controlar ou limitar a pressão do sistema uma vez que, a sobrepressão pode danificar um equipamento ou processo. O alívio da pressão excedente ou seu controle é realizado quando a válvula se abre, expulsando o fluído para fora do sistema através de um caminho auxiliar, reduzindo assim a pressão do sistema. A válvula de alívio de pressão foi desenvolvida para operar em qualquer momento, mesmo quando houver queda de energia elétrica e seus controladores estiverem inoperantes, pois sua única fonte de energia é obtida do fluido que passa através do processo.

Existem diferentes tipos de válvulas de alívio de pressão, uma delas é a válvula com molas, desenvolvida para as necessidades de um sistema simples que necessite de uma proteção confiável e aplicável principalmente em serviços com água. Ela consiste em um bico de entrada, um disco mantido contra o bico para fazer com que o fluxo não ocorra durante a operação normal do sistema, a mola para manter o disco fechado e um corpo para conter os elementos operacionais. A mola atua como regulador da pressão em que a válvula abrirá, ou seja, sua carga é ajustada de acordo com essa pressão.

2.2.2 HARDWARE E SOFTWARE

2.2.2.1 RASPBERRY PI 3

Para a realização da coleta de dados e atuação como interface de rede Modbus TCP e UDP, optou-se pela utilização do microcomputador Raspberry Pi 3 B, esse conta com um microprocessador Broadcom BCM2837 com quatro núcleos e frequência de clock de 1,2 GHz e 64 bits, além de contar com conectividade Ethernet e WiFi. O Raspberry também possui quarenta pinos de GPIO, do inglês General Purpose Input and Output, dentre esses pinos, quatro deles são saídas PWM por hardware, além de ofertar esses sinais, este mini computador possui diversas portas de comunicação. Dentre as opções disponíveis no mercado, concluiu-se que o Raspberry Pi 3 seria o controlador mais viável a se utilizar por conta da sua capacidade operacional e seu custo compatível com o orçamento disponível para o projeto. A linguagem de programação utilizada é a linguagem Python, que possui muitos manuais de fácil acesso na Internet além de contar com muitas bibliotecas para o Raspberry Pi, dessa maneira facilitando a realização deste trabalho e a replicabilidade do mesmo (UPTON; HALFACREE, 2012).

2.2.2.2 CONVERSORES

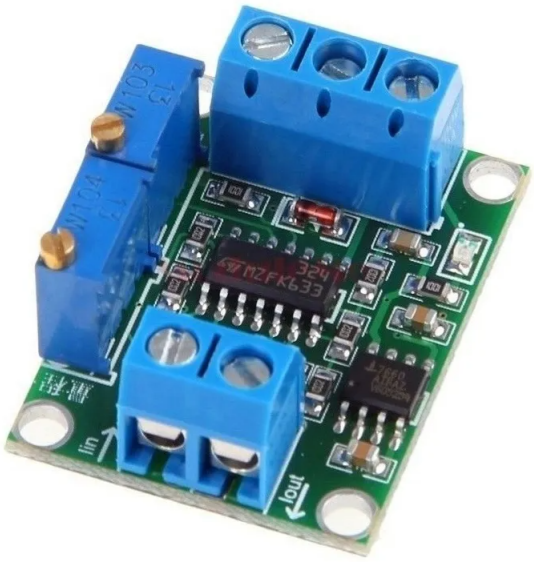
Os transmissores e posicionadores presentes na planta didática operam com sinal de corrente de 4 a 20 mA, ou seja, para enviar informações de posição para as válvulas de controle

e para receber os valores de vazão e nível do sistema é necessário ler e enviar dados via sinal de corrente de 4 a 20 mA (NOVA SMAR S/A, 2018a), (NOVA SMAR S/A, 2018b). Dessa maneira serão necessários dois conversores genéricos voltados à prática de prototipagem acoplados ao sistema de hardware.

O conversor 2 será utilizado para transformar o sinal PWM (*Pulse width modulation*) do Raspberry Pi de 0 a 3,3 V em sinal de corrente para os posicionadores (Figura 17), o conversor 1 transforma o sinal de corrente dos sensores em sinal de tensão de 0 a 3,3 V (Figura 16). Para realizar a leitura do sinal analógico de tensão será utilizado o circuito integrado ADS1115 (Figura 18), este possui uma resolução de 16 bits em suas quatro entradas analógicas e envia o sinal convertido por meio de uma rede I2C. Este circuito integrado é encontrado em lojas de eletrônica e na Internet, integrado em módulos para o uso em prototipagem. Na internet é comum encontrar várias opções de conversores com essas características e que, a princípio, qualquer um que opere nos mesmos níveis de tensão, corrente e impedância pode ser utilizado.

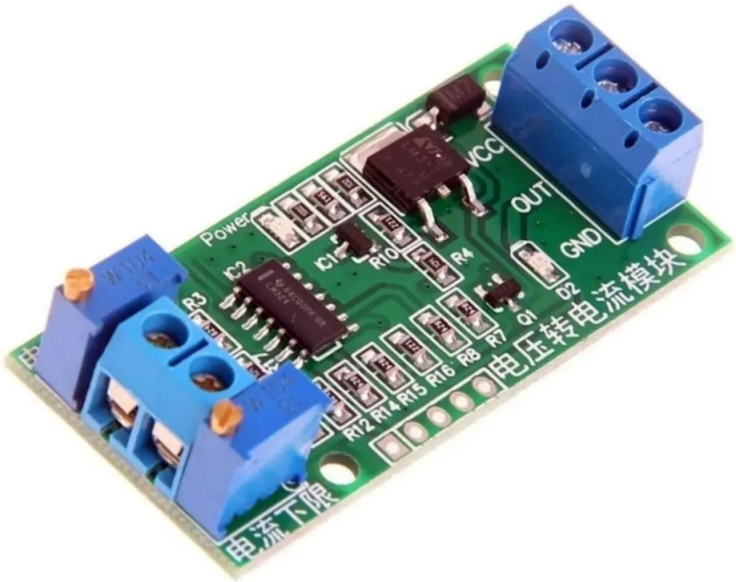
O circuito de ligação do Raspberry Pi e conversores encontra-se no anexo A e os componentes citados nas figuras abaixo.

Figura 16: Conversor 1



Fonte: Mercadolivre (2020)

Figura 17: Conversor 2



Fonte: Mercadolivre (2020)

Figura 18: AD1115

Fonte: Casa da robótica (2022)

2.2.2.3 SISTEMA DE CONTROLE SUPERVISÓRIO E AQUISIÇÃO DE DADOS

Os sistemas SCADA realizam a interligação entre diferentes sistemas distribuídos em uma planta ou entre varias industrias. Este sistema possui as funções de coletar os dados dos sensores e atuadores do sistema, enviar essas informações para uma área central, disponibiliza-los para que operadores possam monitorar a planta e controlar os parâmetros de operação do processo, além de gerar análises e relatórios. (BAILEY; WRIGHT, 2003).

Para realizar a supervisão do processo de controle sobre a planta optou-se em utilizar o software do Elipse E3, que é uma plataforma IHM/SCADA para aplicações de monitoramento e comando em tempo real desenvolvido para atender aos requisitos necessário, desde pequenos projetos até complexos centros de operação. O E3 Studio é um dos componentes principais

do Elipse e será utilizado para a criação de telas, configuração da comunicação e edição de *scripts*. Outro componente fundamental é o E3 *Server*, servidor de aplicações responsável por sincronizar as bases de dados e gerenciar os principais processos. Por último o E3 *Viewer*, utilizado como interface de operação com o usuário, tornando possível visualizar e operar a aplicação de qualquer computador, via internet ou intranet (ELIPSE SOFTWARE, 2020).

3 PROCEDIMENTOS PRÁTICOS

Neste capítulo serão apresentados os procedimentos utilizados para adaptação da planta didática de vazão e nível em uma planta acessível via rede para disciplinas relacionadas à instrumentação industrial, identificação de sistemas e controle multivariável. Para tal, foram necessárias adaptações físicas, inclusão de uma plataforma de interface com novo hardware, aplicação de softwares para monitoramento e aquisição de dados e disponibilização dos dados da planta via protocolo UDP.

Os materiais e dispositivos utilizados estão descritos na Tabela 2.

Tabela 2: Materiais e dispositivos

Quantidade	Materiais	Descrição
01	RaspBerry Pi 3	Microcomputador
01	ADS1115	Conversor A/D - I2C
02	Conversor genérico de Tensão para Corrente	0 a 3.3 V para 4 a 20 mA
02	Conversor genérico de Corrente para Tensão	4 a 20 mA para 0 a 3.3 V
04	Borne	Borne tipo banana fêmea
02	Chave	Chaves HH alavanca 2 posições
01	Cabo	Cabo PP 2x0.5 - 2 metros
08	Cabo Banana - Banana	Cabo + Borne banana macho
01	Fonte de alimentação	Fonte 24 V, 2 A

3.1 ADAPTAÇÃO DA PLANTA

A planta, em sua configuração original, não disponibiliza acesso aos sinais das válvulas e sensores. Dessa forma, fez-se necessário a modificação do esquema elétrico, com a utilização de chaves que permitem a comutação entre o circuito original e o novo sistema proposto e a adição de bornes tipo banana para a coleta e transmissão de dados. As figuras 19 e 20 a seguir, mostram as adaptações mencionadas.

Figura 19: Acesso ao transmissor de vazão (FIT-11)

Fonte: Autoria própria

Figura 20: Acesso à valvula de vazão (FV-11)

Fonte: Autoria própria

A Figura 19 representa a adaptação realizada para disponibilizar um ponto de coleta de dados do transmissor de vazão. Para isso utilizou-se uma chave HH, de duas posições, que seleciona em qual dos sistemas a planta será utilizada, sistema original ou novo sistema. Na primeira posição (chave virada para a esquerda) a planta trabalha com seu circuito original, já na segunda posição (chave virada para a direita) a planta trabalhará no novo sistema proposto.

Os mesmos componentes foram utilizados na Figura 20, entretanto a função dessa adaptação é o envio de dados para a válvula. Da mesma maneira, a chave HH, de duas posições, realiza a comutação entre sistema original da planta e sistema novo. Em ambas as adaptações foi mantido o mesmo padrão de descrição de função dos outros pontos presentes na planta, em que a posição ”NORMAL”da chave é a operação com o circuito original da planta, e a segunda posição ”CALIBR.”é o modo em que o instrumento selecionado fica desligado do circuito original, essa função também é utilizada para realizar a calibração desses instrumentos.

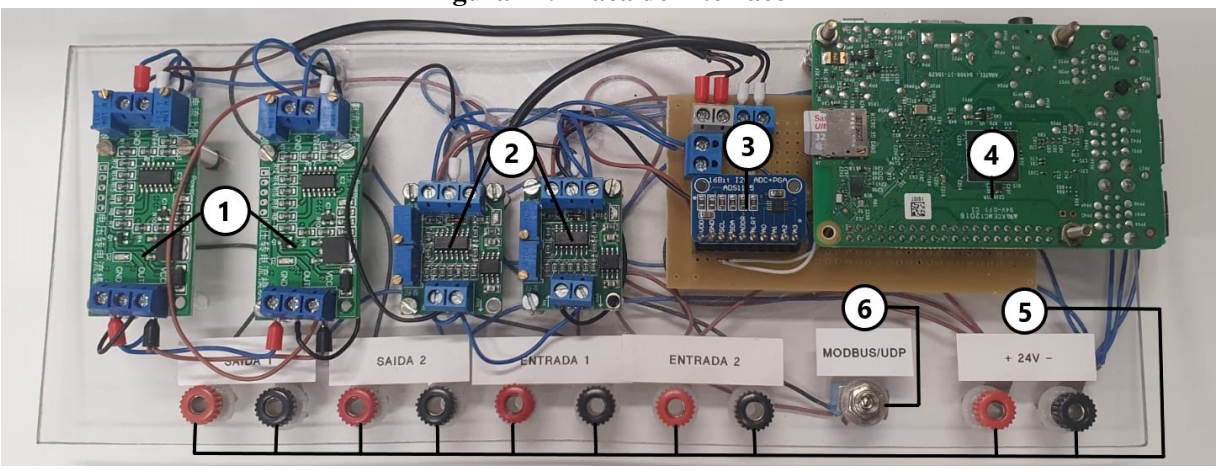
3.2 INTERFACE COM A PLANTA

3.2.1 PLACA DE INTERFACE E MONTAGEM DO CIRCUITO

As adaptações realizadas na planta disponibilizaram o acesso aos pontos de envio e coleta de dados. A partir disso fez-se necessário a montagem de uma placa de interface que se conecta a esses pontos, ou seja, uma placa que ao ser conectada a planta consiga enviar comandos para as válvulas assim como receber os dados dos transmissores, além de permitir a conexão rede para que outros dispositivos acessem a planta.

A montagem do circuito foi realizada em uma placa de policarbonato baseado no esquemático elétrico apresentado no Anexo A, conforme mostrado na Figura 21.

Figura 21: Placa de interface



Fonte: Autoria Própria.

A Figura 21 mostra o resultado final da montagem da placa de interface é composta por cinco componentes diferentes numerados na própria figura, são eles:

1. Par de conversores de tensão para corrente. Esses são responsáveis por comandar as válvulas e sua função no circuito é converter os comandos enviados pelo microcomputador RaspBerry Pi, que variam de 0 a 3.3 V para as válvulas que trabalham de 4 a 20 mA. Esses conversores estão representados no Anexo A como ”Conversor 1”.

2. Par de conversores de corrente para tensão. Responsáveis por receber os valores dos transmissores. Sua função no circuito é converter os sinais dos transmissores, que variam de 4 a 20 mA para 0 a 3.3 V, que posteriormente é convertido para um sinal digital através do conversor A/D (3), dessa maneira tornando possível a leitura através do microcomputador RaspBerry Pi. Esses conversores estão representados no Anexo A como ”Conversor 2”.

3. Conversor A/D responsável por transformar os sinais analógicos de tensão dos conversores 2 (Anexo A) para sinais digitais, podendo assim serem lidos pelo RaspBerry Pi.

4. Microcomputador RaspBerry Pi 3. Responsável por gerenciar todo o sistema.

5. Bornes tipo banana para conexão física com a planta.

6. Chave seletora entre modos de operação.

3.2.2 COMISSIONAMENTO

Após a conclusão da montagem física da placa de interface (Figura 21) utilizou-se o auxílio de três softwares, o MATLAB, o sistema operacional do Raspberry e o supervisório SCADA, possibilitando o envio de comandos, a leitura e análise dos dados fornecidos pelo sistema.

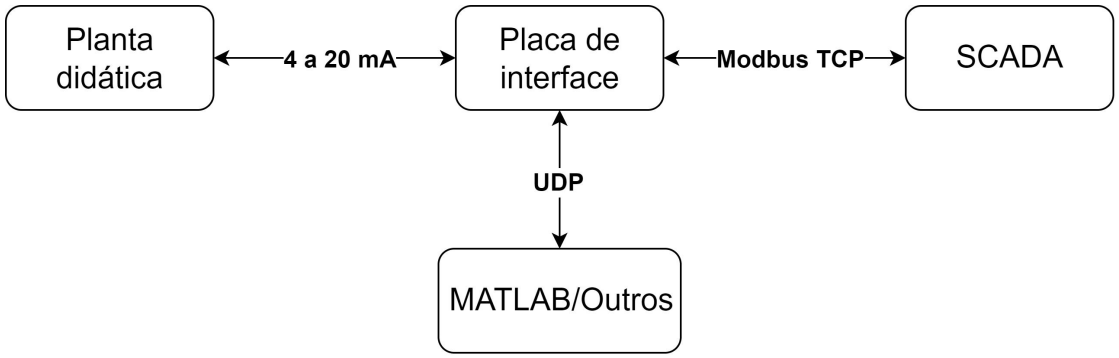
Na primeira etapa, bibliotecas disponíveis na linguagem Python foram utilizadas para auxiliar na interação do software com os dispositivos presentes na placa e com os equipamentos industriais presentes na planta.

Em seguida, utilizou-se um sinal PRMLS para obter os dados de entrada e saída e dar início à etapa de identificação de sistema. Optou-se pela ferramenta MATLAB para gerar este de sinal e posteriormente transferi-lo para o Raspberry através de um arquivo CSV, possibilitando a inserção das informações de entrada para as válvulas. Assim, foi possível coletar os dados de

saída dos transmissores, dados esses que foram utilizados para a identificação e análise gráfica com o auxílio do MATLAB.

Por fim, após a identificação do sistema o acesso aos equipamentos da planta foi disponibilizado através da implementação do protocolo de rede industriais UDP no microcomputador RaspBerry Pi.

Figura 22: Diagrama das interfaces



Fonte: Autoria própria.

3.2.3 COMUNICAÇÃO VIA REDE

3.2.3.1 IMPLEMENTAÇÃO

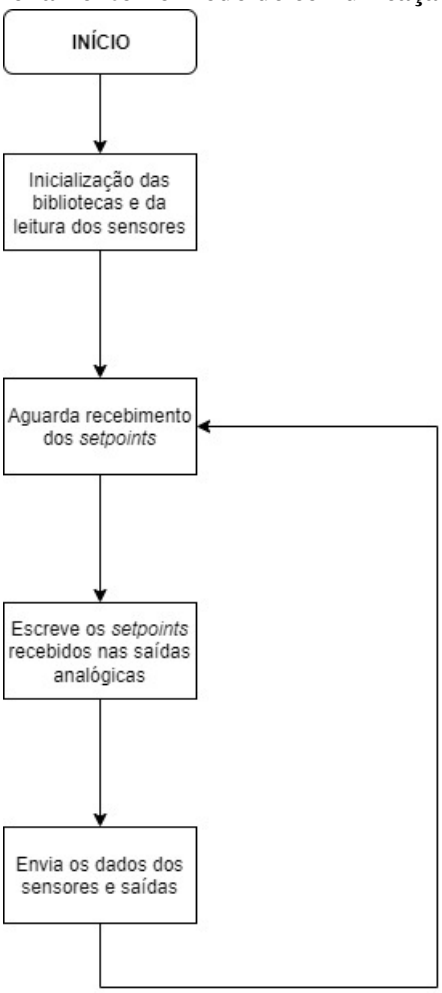
No microcomputador foram implementados dois protocolos de redes para realizar o controle da planta didática. É possível realizar a comunicação utilizando o protocolo Modbus TCP, em que está disponível um modo automático e um modo manual, e o protocolo UDP, em que também está disponível o controle manual. A seleção dos protocolos de comunicação é realizada por meio de uma chave presente na placa de interface.

Durante a operação utilizando o protocolo Modbus é possível alternar entre os modos automático e manual por meio de um *holding register*. Em modo manual os valores de *setpoint* enviados a placa de interface são enviados diretamente as válvulas, sem a atuação de um controlador, ou seja, é realizado o controle em malha aberta. Já o controle do sistema em modo automático, em que as malhas de vazão e nível são controladas por um controlador PID diagonal, os *setpoints* e as constantes dos controladores PID são alterados por meio de *holding registers*, nesse modo também é possível monitorar a atuação das válvulas e os valores lidos pelos sensores. No modo automático o sistema de controle é do tipo malha fechada.

Esses dois modos foram implementados no Raspberry Pi utilizando a linguagem de programação Python e foi utilizado as bibliotecas *pyModbusTCP* para criar um servidor Modbus TCP no microcomputador (LEFEBVRE, 2022), e a biblioteca *simple-pid* para realizar o controle PID das malhas (LUNDBERG, 2022). Para realizar a leitura dos sensores utilizou-se uma *thread* separada, com o intuito de obter um tempo de aquisição menor.

Quando o microcomputador opera em modo manual com o protocolo UDP, é utilizado a biblioteca *socket* para criar um servidor UDP e também é realizada a aquisição dos dados dos sensores por meio de uma *thread*, o funcionamento desse modo da interface é descrito em forma de fluxograma na Figura 23.

Figura 23: Funcionamento no modo de comunicação utilizando UDP

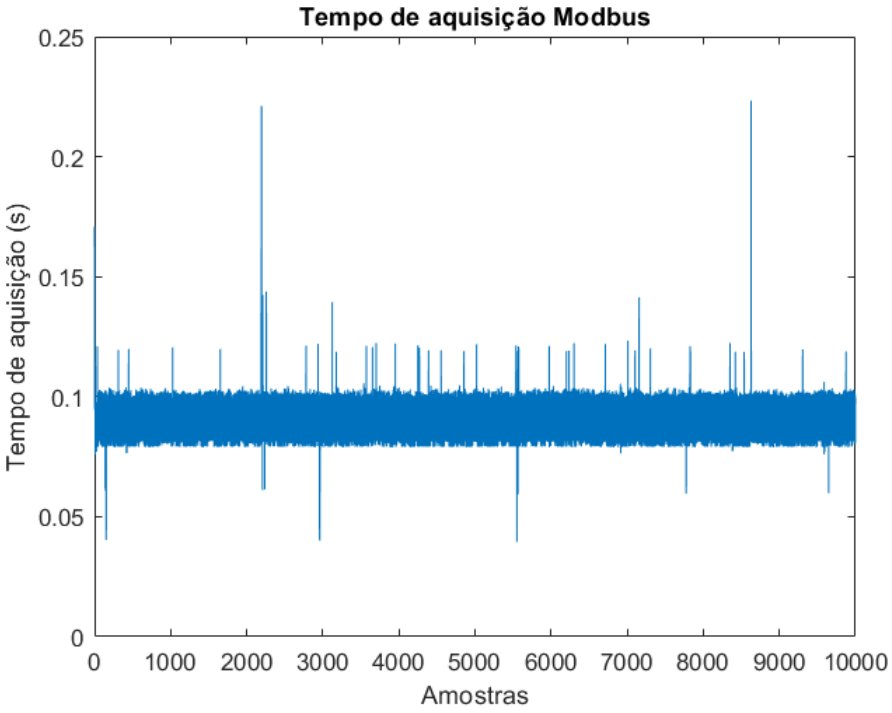


Fonte: Autoria própria.

O uso do protocolo UDP para controle manual da planta foi implementado para possibilitar o controle remoto, uma vez que o protocolo Modbus TCP possui tempo de aquisição maior e mais instável, diferente do protocolo UDP que possui tempo de aquisição menor. As

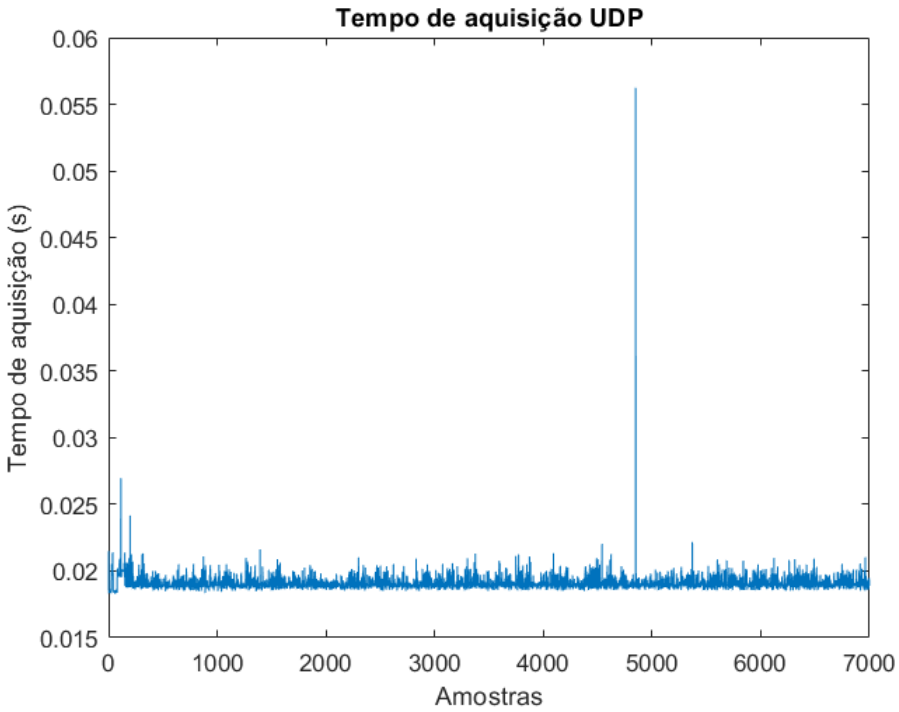
figuras a seguir mostram a plotagem dos dados de tempo coletados em cada um dos protocolos aplicados no Raspberry Pi, assim como o funcionamento utilizando o protocolo UDP.

Figura 24: Tempo de aquisição utilizando o protocolo Modbus TCP



Fonte: Autoria própria.

Figura 25: Tempo de aquisição utilizando o protocolo UDP



Fonte: Autoria própria.

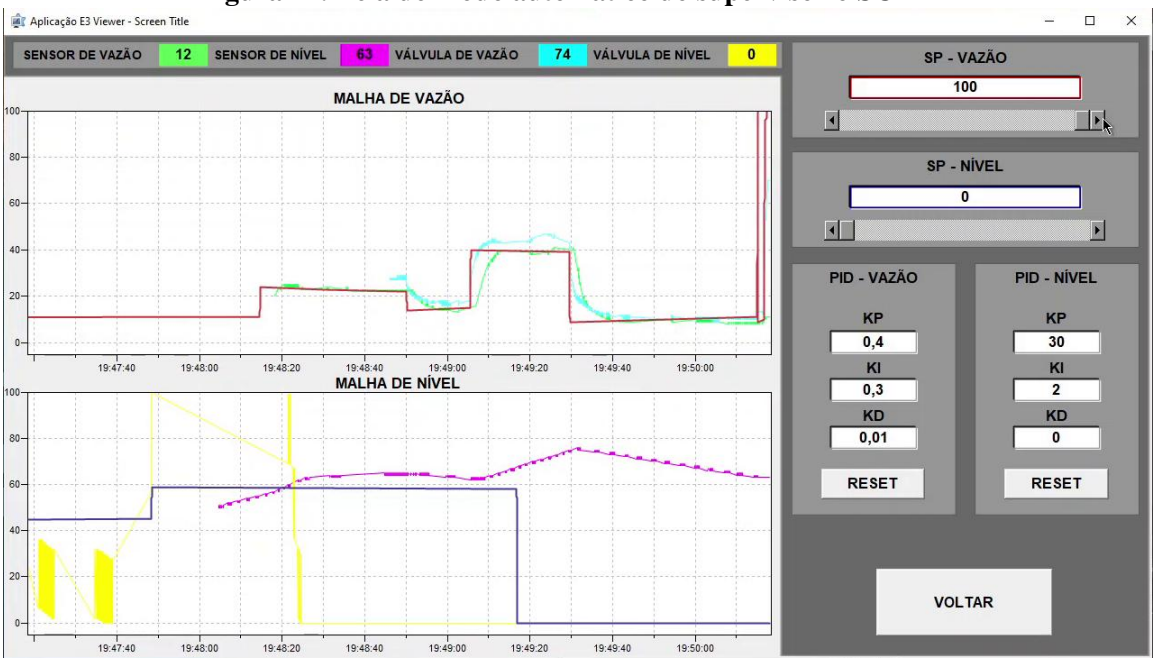
3.2.4 SCADA

As figuras a seguir mostram as telas elaboradas para o sistema supervisório com o o auxílio do software Elipse E3. A Figura 26 representa a tela inicial que permite o usuário selecionar entre dois modos de operação do sistema: manual ou automático. Após a seleção do modo de operação, o usuário é direcionado para a tela do modo automático, Figura 27, ou para a tela do modo manual, Figura 28.

Figura 26: Tela Inicial do supervisório SCADA

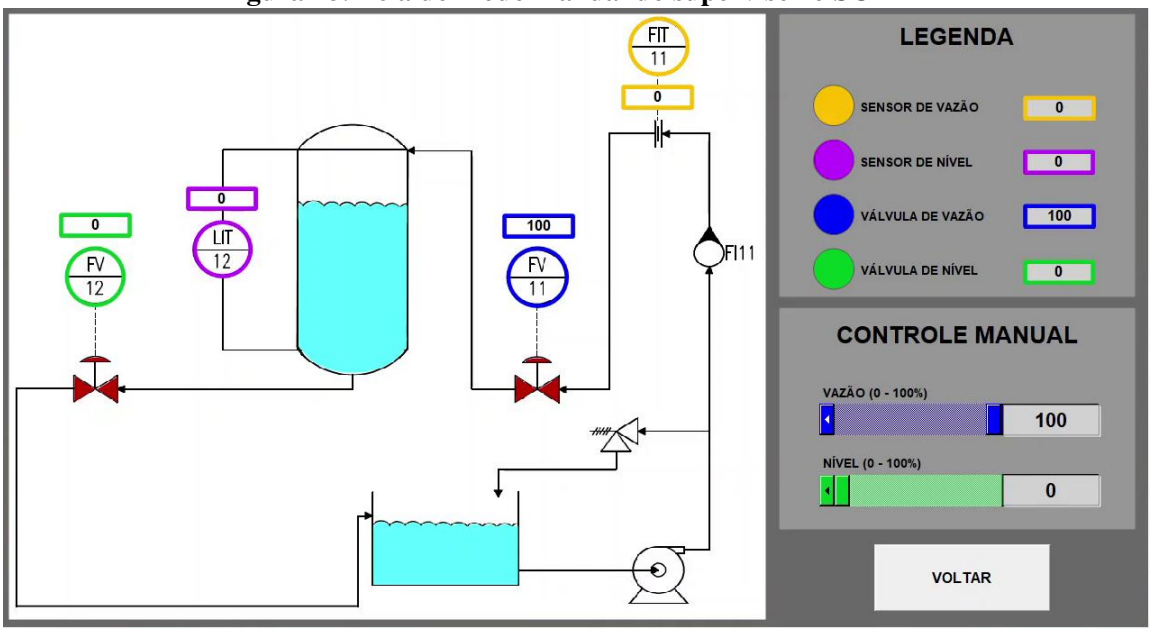
Fonte: Autoria própria.

Figura 27: Tela do modo automático do supervisório SCADA



Fonte: Autoria própria.

Figura 28: Tela do modo manual do supervisório SCADA



Fonte: Autoria própria.

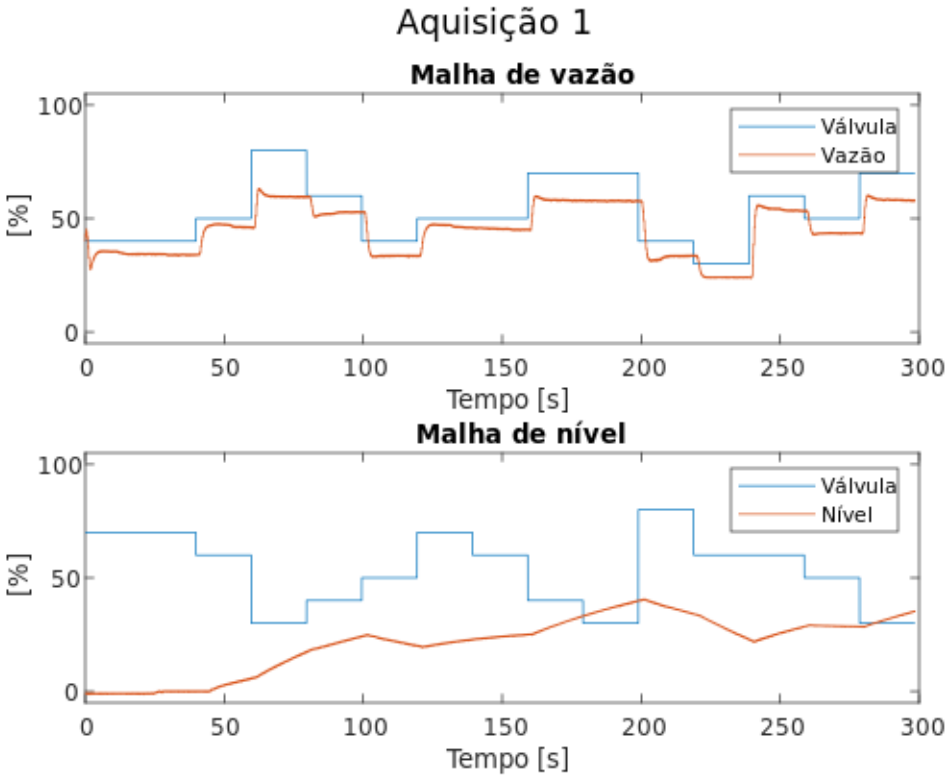
4 RESULTADOS OBTIDOS

Visto que esse trabalho possui dois grandes objetivos, identificação da planta didática e comunicação via rede, além da disponibilização da interface para aplicação de diferentes tipos de controle, pode-se concluir cada um dos objetivos no seu discorrer e nessa seção apresenta-se os resultados obtidos em cada etapa.

4.1 IDENTIFICAÇÃO DE SISTEMAS

A identificação do sistema da planta didática foi realizada com o auxílio do software MATLAB, através da implementação de um algoritmo escrito na linguagem própria do software. Para o desenvolvimento dos experimentos utilizou-se o microcomputador Raspberry Pi para atuar nas válvulas com um sinal PRMLS, também gerado via MATLAB, que foi utilizado como sinal de *setpoint* e para coletar os dados medidos pelos sensores de vazão e nível. Foram realizados diversos experimentos com duração de 298,5 segundos, e uma frequência de amostragem de 50Hz. Os dados referentes a um desses experimentos é representado na Figura 29.

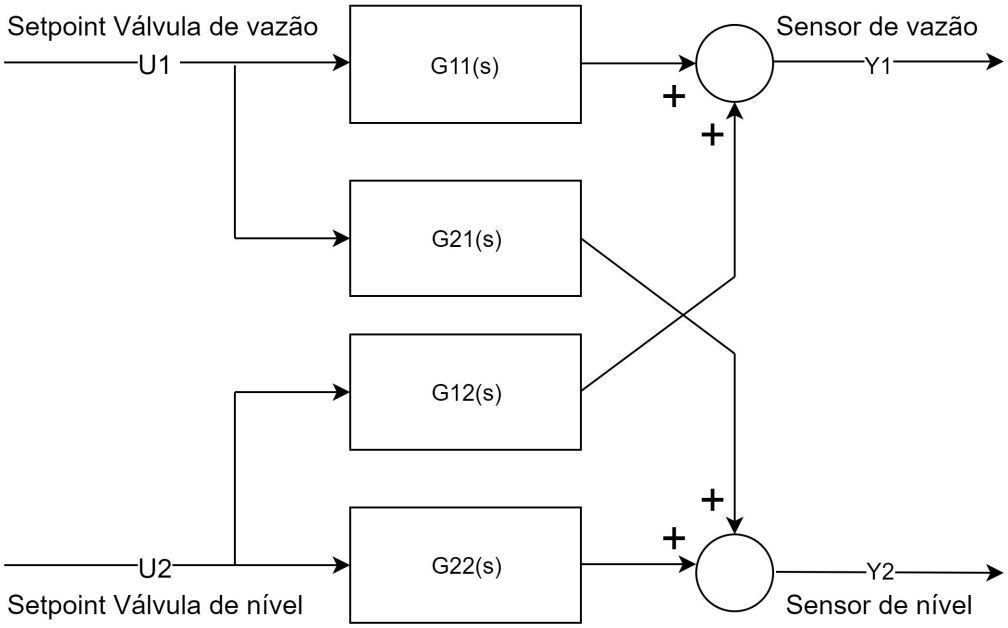
Figura 29: Dados colhidos da planta didática



Fonte: Autoria própria.

Após a coleta de dados, realizou-se as etapas de cálculo do modelo e validação, para isso foi utilizada a *toolbox* do MATLAB chamada *System Identification Toolbox* (LJUNG, 2020). Essa ferramenta auxilia na obtenção de modelos, ao aplicar os métodos e algoritmos para identificar um modelo conforme os parâmetros selecionados. Foi escolhido utilizar a função *tfest*, que estima uma função de transferência $H(s)$ para os dados de entrada e saída coletados no experimento, esse tipo de modelo foi escolhido por conta da familiaridade do grupo e o extenso uso desse tipo de modelo nas disciplinas de controle deste curso de graduação. Para o calculo do modelo foi necessário definir o número de pólos e zeros da função de transferência a ser estimada. A função *tfest* utiliza uma combinação de métodos de identificação para encontrar o modelo com menor erro entre os dados estimados e os dados de validação (LJUNG, 2020).

Figura 30: Diagrama de entradas e saídas



Fonte: Autoria própria.

Para o cálculo do modelo, função de transferência da planta foi separado em dois sistemas MISO, conforme a Figura 30, e testados várias configurações de polos e zeros a fim de determinar o melhor modelo, esses dados estão dispostos na Tabela 3 e Tabela 4

Tabela 3: MSE dos modelos testados para *Y1*

Z/P	1	2	3	4	5	6	7	8	9	10
0	15.62	12.17	11.23	11.07	10.99	16.06	1073.99	96.39	61.95	105.06
1		11.44	11.16	12.04	126.79	70.27	47.79	10.83	429.54	59.34
2			11.01	11.00	11.00	11.10	9.20	25.92	126.87	101.36
3				12.40	10.95	10.31	8.57	10.09	16.77	111.73
4					11.02	14.08	8.85	73.36	25.66	9.86
5						6.87	9.91	33.72	7.33	358.34
6							11.61	270.21	17.63	1075.14
7								8.89	8.32	14.41
8									10.21	7279.94
9										10.49

Tabela 4: MSE dos modelos testados para *Y2*

Z/P	1	2	3	4	5	6	7	8	9	10
0	25.70	9.26	29.28	68.88	2.86	2.68	3.09	31.72	1306.15	3047917.18
1		13.86	4.16	33.12	4.32	2.29	2.50	7.50	7464.19	313.85
2			8.55	5.02	3.52	2.49	2.13	13.18	11.05	363.47
3				1.03	2.90	6.13	2.87	48.04	261.42	260.06
4					8.95	2.13	51.61	6.99	55.68	379.53
5						2.48	2.04	45.10	13576.42	180.36
6							7.83	27.15	16.26	1703.50
7								6.36	22.06	9.17
8									108.81	69.42
9										26.23

Utilizando o valor MSE como principal critério de decisão, foi escolhido o modelo para o subsistema Y1, ver Tabela 3, com seis polos e cinco zeros que possui MSE igual a 6,87, da mesma maneira para o subsistema Y2, ver Tabela 4, com quatro polos e três zeros que possui MSE igual a 1,03. Por fim, a função de transferência completa da planta didática foi montada conforme (13).

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{13}$$

$$G_{11} = \frac{0{,}6889s^5 - 0{,}1099s^4 - 3{,}44s^3 + 12{,}89s^2 - 0{,}0403s + 0{,}02666}{s^6 + 6{,}822s^5 + 18{,}11s^4 + 26{,}22s^3 + 15{,}97s^2 + 0{,}05131s + 0{,}03119}$$

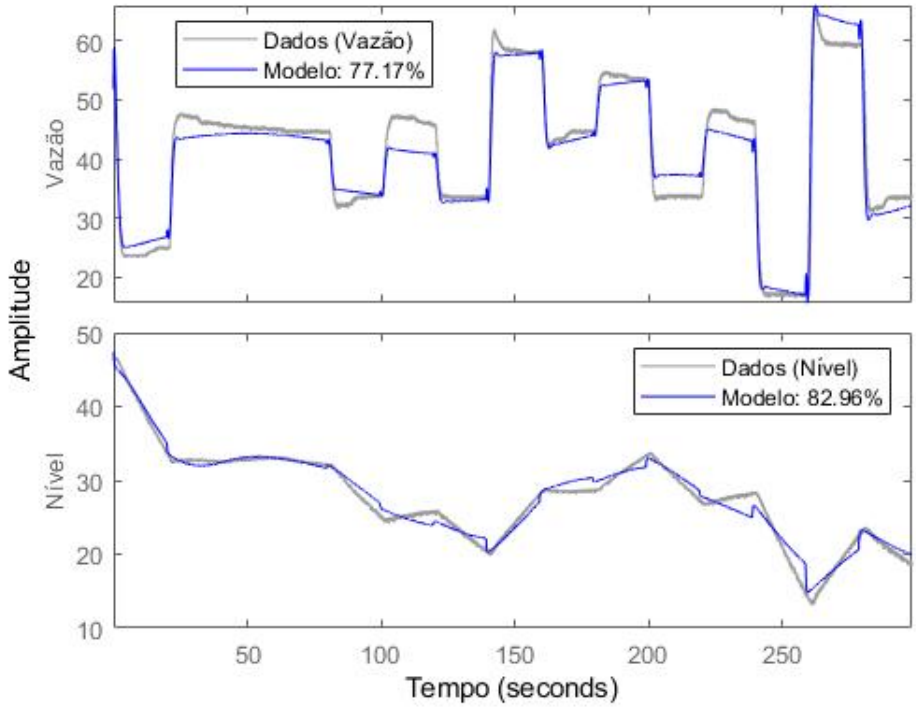
$$G_{12} = 0$$

$$G_{21} = \frac{-0{,}3008s^3 + 0{,}04263s^2 - 0{,}001004s + 4{,}724e{-}05}{s^4 + 4{,}401s^3 + 0{,}09015s^2 + 0{,}005354s + 4{,}877e{-}05}$$

$$G_{22} = \frac{0{,}0006106s^3 - 0{,}0005784s^2 + 1{,}406e{-}05s - 2{,}593e{-}07}{s^4 + 0{,}06537s^3 + 0{,}004083s^2 + 5{,}93e{-}05s + 1{,}048e{-}06}$$

Utilizando outro conjunto de dados da planta foi realizada a validação do modelo, comparando os dados do sistema real e a simulação utilizando o modelo estimado. Essa comparação foi realizada utilizando a função *compare(data,sys)* e está representada na Figura 31, em que é possível visualizar que o modelo identificado, indicado na cor azul, tem o comportamento muito semelhante ao sistema real, indicado na cor cinza. Além da comparação visual, essa ferramenta também calcula a qualidade do ajuste, do inglês *Goodness of fit*, que mostra o quanto o sistema identificado representa o sistema real.

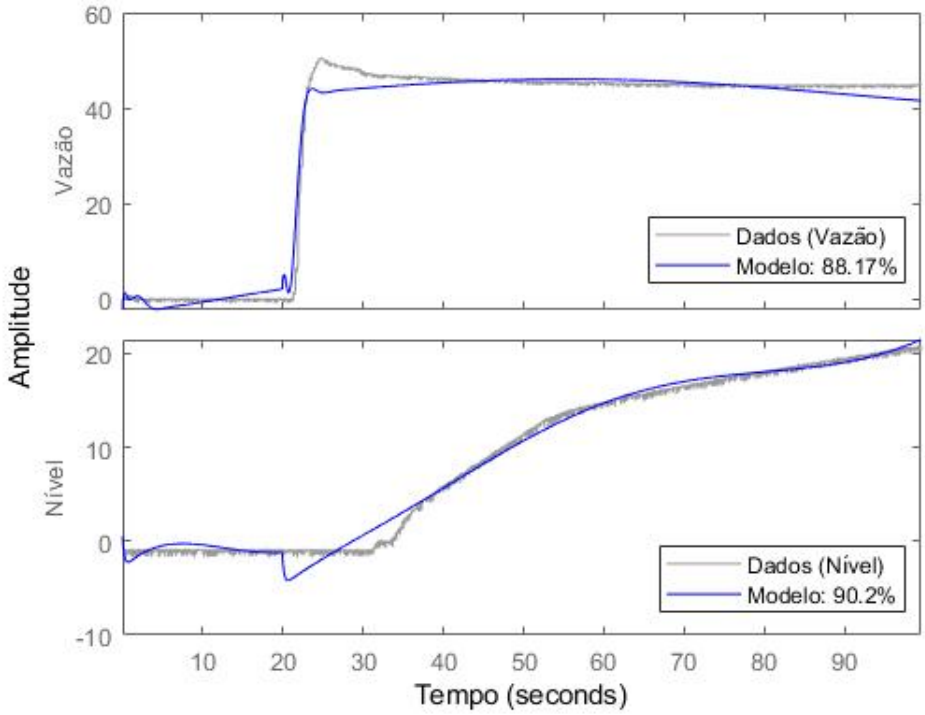
Figura 31: Modelo identificado



Fonte: Autoria própria.

Outra simulação realizada foi a resposta ao degrau, representada na Figura 32, em que apresenta um resultado semelhante ao observado na planta didática e com valores de *Goodness of fit* superiores a 85%. Para a simulação da resposta ao degrau foram utilizado dados de um experimento prático de resposta ao degrau da planta e novamente utilizou-se a função *compare*.

Figura 32: Resposta ao degrau do modelo identificado



Fonte: Autoria própria.

4.1.1 PROBLEMAS ENCONTRADOS

4.1.2 PROTOCOLO DE COMUNICAÇÃO DE REDE

Após o início dos procedimentos práticos optou-se pela adaptação de algumas práticas abordadas devido a erros e dificuldades encontradas.

Em um primeiro momento o protocolo de rede industrial escolhido para iniciar os testes de comunicação foi o protocolo Modbus TCP, entretanto encontrou-se inconsistências na taxa de amostragem dos dados. Notou-se variações inesperadas da frequência, consequentemente o controle remoto da planta poderia ser comprometido. Dessa maneira, a solução encontrada foi utilizar um protocolo mais simples, em termos de arquitetura e com maiores taxas de transmissão de dados, assim conforme descrito na seção anterior, o protocolo testado e validado foi o protocolo UDP.

4.1.3 MULTITAREFAS

Durante os procedimentos práticos pode-se notar que a maneira que o software executava o código programado impossibilitava a realização de troca de dados com a planta

e a comunicação via rede simultaneamente. Logo a utilização de tarefas *(threads)* tornou-se essencial para dar continuidade nas atividades. Esse método permitiu que o *software* executasse duas funções em paralelo, dividindo o processamento.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

Apesar dos problemas encontrados durante a fase de testes, realização de experimentos e coleta de dados, pode-se contornar essas situações e prosseguir com o objetivo inicial que se resumia em: disponibilizar um sistema embarcado, composto pelo microcomputador Raspberry Pi 3 e periféricos eletrônicos de escrita e leitura de instrumentos; realizar a identificação do sistema da planta didática, o que se deu através da *Toolbox* do software MATLAB; disponibilizar a comunicação via protocolo de redes industriais, ambos implementados em linguagem Phtyon e com auxílio de algumas bibliotecas; aplicar um método de controle e disponibilizar uma interface de supervisão, através do software Elipse E3, para monitoramento e alteração desse controlador PID embarcado.

Assim como previsto durante o aprofundamento teórico, durante os procedimentos práticos notou-se que a válvula de nível não provoca grande influência na vazão, por esse motivo a função de transferência descrita na equação (13) tem o termo G12, que relaciona a entrada u2 e a saída y1, igualado a zero, pois provavelmente possui influência desconsiderável comparada as demais, de forma que pode-se considerá-la nula sem prejudicar a análise.

O objetivo em relação a identificação foi concluído ao analisar o NRMSE dos modelos comparados ao sistema da planta. Com o auxílio do MATLAB e da *Toolbox* de identificação, obteve-se um modelo para a malha de vazão com um valor de NRMSE de $87,17\%$ e de $90,2\%$ para malha de nível, utilizando a resposta ao degrau como referência devido ao tempo de resposta do sistema real como um todo, validando assim a função transferência identificada.

O sistema embarcado provê uma interface com a planta através dos protocolos de rede, possibilitando que outros alunos possam manipular dados de entradas, coletar as informações de saída, utilizar técnicas de controle multivariável ou até aplicar métodos de identificação no sistema. Vale ressaltar que o novo sistema implementado não substitui o modelo inicial, as implementações aplicadas na planta permitem que o aluno opte por utilizar os controladores PID presentes no sistema original ou o novo sistema instalado.

5.2 SUGESTÕES PARA TRABALHOS FUTUROS

Como sugestão para futuros trabalhos pode-se se realizar uma identificação utilizando modelos não lineares, desenvolver um controlador para ser implantando no microcomputador e também desenvolver um controlador para controle remoto da planta. Fora do âmbito das disciplinas de controle e identificação, também é sugerido realizar a comunicação com a planta utilizando os CLPs presentes no laboratório CB-010 da Universidade Tecnológica Federal do Paraná, assim como a criação de funções que realizem a interface com o sistema apresentado nesse trabalho. Além das sugestões já citadas outra linha de trabalho é o aprimoramento e adição de outros protocolos de comunicação na placa de interface, experimentação com o uso de sistemas operacionais *Real Time* no Raspberry Pi.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução à Identificação de Sistemas**. [S.l.: s.n.], 2015. ISBN 978-85-423-0079-6.

BAILEY, D.; WRIGHT, E. **Practical SCADA for industry**. [S.l.]: Elsevier, 2003.

BENNETT, S. **Robust Control : Systems, Theory and Analysis.** [S.l.]: Nova Science Publishers, Inc, 2017. (Mechanical Engineering Theory and Applications).

BILLINGS, S. A. **Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains**. [S.l.]: John Wiley & Sons, 2013.

BOYES, W. **Instrumentation reference book**. 4. ed. [S.l.]: Butterworth-Heinemann, 2009.

BRAUN, M. et al. Multi-level pseudo-random signal design and"model-on-demand"estimation applied to nonlinear identification of a rtp wafer reactor. In: IEEE. **Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)**. [S.l.], 1999. v. 3, p. 1573–1577.

BRISCOE, N. Understanding the osi 7-layer model. **PC Network Advisor**, v. 120, n. 2, p. 13–15, 2000.

CAMERON, I. T.; HANGOS, K. **Process modelling and model analysis**. [S.l.]: Elsevier, 2001.

CARA, E. F.; IRIONDO, E. Z. Control theory: History, mathematical achievements and perspectives. **Boletín de la Sociedad Española de Matemática Aplicada, 26, 79-140.**, Sociedad Española de Matemática Aplicada, 2003.

CASA DA ROBóTICA. **Módulo conversor analógico - AD1115**. 2022. Disponível em: https://www.casadarobotica.com/sensores-modulos/modulos/conversores/modulo-conversor-analogico-digital-adc-ads1115-16-bit-i2c. Acesso em: 22 fev. 2022.

COUGHLAN, P.; COGHLAN, D. Action research for operations management. **International journal of operations & production management**, MCB UP Ltd, v. 22, n. 2, p. 220–240, 2002.

DRESCH, A.; LACERDA, D. P.; MIGUEL, P. A. C. Uma análise distintiva entre o estudo de caso, a pesquisa-ação e a design science research. **Revista Brasileira de Gestão de Negócios-RBGN**, Fundação Escola de Comércio Álvares Penteado, v. 17, n. 56, p. 1116–1133, 2015.

ELIPSE SOFTWARE. **Elipse E3**. 2020. Disponível em: https://www.elipse.com.br/produto/elipse-e3/. Acesso em: 06 out. 2020.

JAMES, G. et al. **An introduction to statistical learning**. [S.l.]: Springer, 2013.

KASBERGER, J. Advantages of industrial ethernet. 2011.

KON, J. et al. Practical application of model identification based on arx models with transfer functions. **Control Engineering Practice**, Elsevier, v. 21, n. 2, p. 195–203, 2013.

KOZÁK, Š. State-of-the-art in control engineering. **Journal of Electrical Systems and Information Technology**, Elsevier, v. 1, n. 1, p. 1–9, 2014.

KUPHALDT, T. R. **Lessons in industrial instrumentation**. 2.32. ed. [S.l.]: Creative Commons Attribution/PAControl. com, 2008.

KUROSE, J.; ROSS, K. **Computer Networking: A Top-down Approach**. Pearson, 2010. ISBN 9780131365483. Disponível em: https://books.google.com.br/books?id=qPAcnwEACAAJ.

LEFEBVRE, L. **pyModbusTCP 0.1.10**. 2022. Disponível em: https://pypi.org/project/pyModbusTCP/. Acesso em: 21 mar. 2022.

LJUNG, L. **System Identification: Theory for the User**. [S.l.]: Prentice Hall PTR, 1999. ISBN 0-13-881640-9.

LJUNG, L. **System identification toolbox: Reference**. [S.l.]: MathWorks, 2020.

LUGLI, A. B.; SANTOS, M. M. D. **Redes industriais para automação industrial**. [S.l.]: Saraiva Educação SA, 2010.

LUNDBERG, M. **simple-pid**. 2022. Disponível em: https://github.com/m-lundberg/simple-pid. Acesso em: 21 mar. 2022.

MERCADOLIVRE. **Conversor eletrônico**. 2020. Disponível em: https://www.mercadolivre.com.br/perfil/MC1+COMPONENTES. Acesso em: 12 fev. 2020.

MODBUS ORGANIZATION. **Modbus FAQ**. 2020. Disponível em: https://modbus.org/faq.php. Acesso em: 16 set. 2020.

NISE, N. S.; SILVA, F. R. da. **Engenharia de sistemas de controle**. [S.l.]: LTC, 2002.

NOVA SMAR S/A. **Manual de instruções e manuteção - Posicionador Inteligente de Válvulas FY301**. Sertãozinho, Brasil, 2018. Disponível em: https://www.smar.com/pdfs/manuals/.

NOVA SMAR S/A. **Manual de instruções e manuteção - Transmissor inteligente de pressão LD400**. Sertãozinho, Brasil, 2018. Disponível em: https://www.smar.com/pdfs/manuals/.

OGATA, K. **Modern Control Engineering**. [S.l.]: Pearson, 2014.

SEBORG, D. E. et al. **Process dynamics and control**. [S.l.]: John Wiley & Sons, 2010.

SOMMER, J. et al. Ethernet–a survey on its fields of application. **IEEE Communications Surveys & Tutorials**, IEEE, v. 12, n. 2, p. 263–284, 2010.

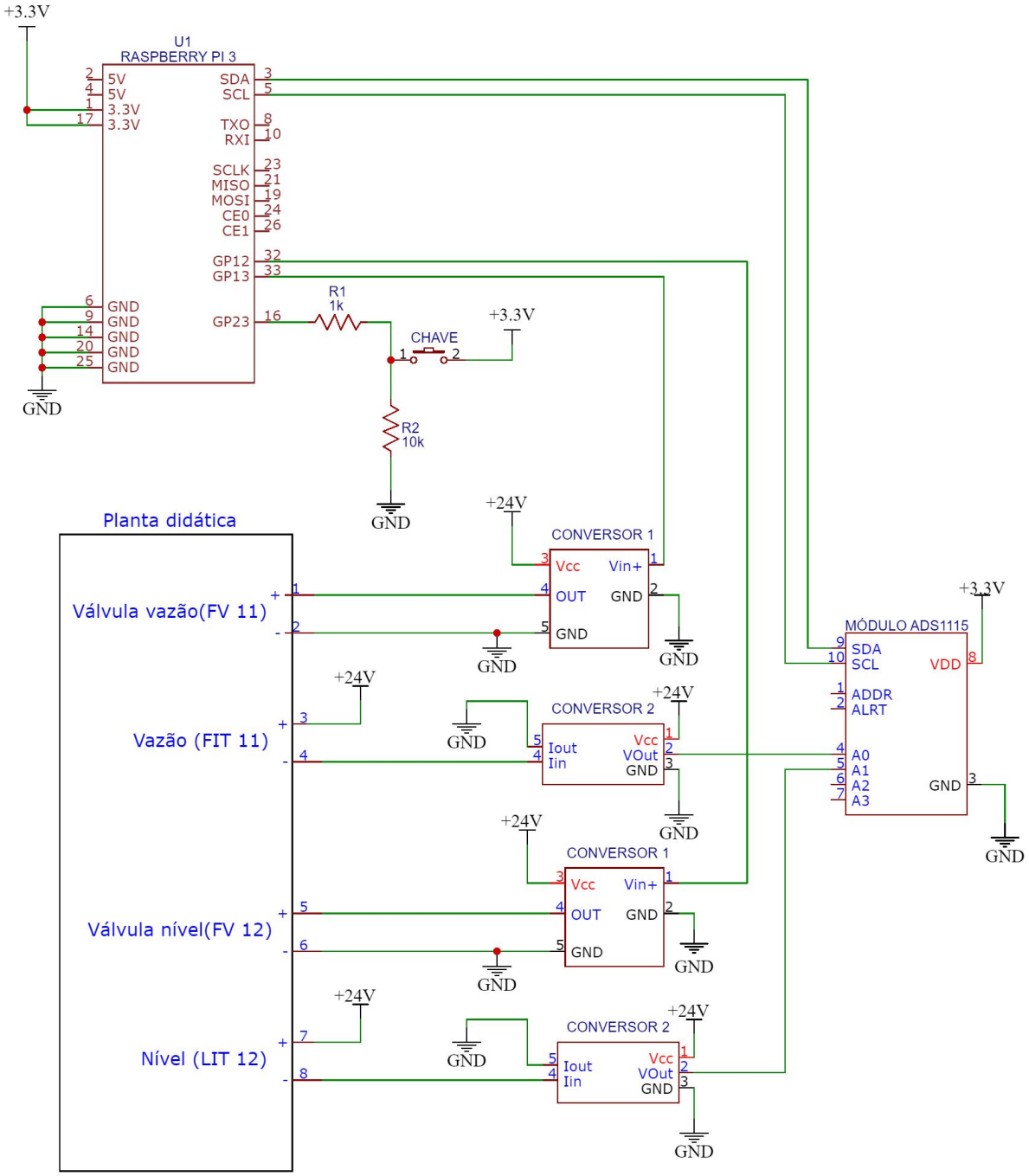
THAMMADI, A. Reliable user datagram protocol (rudp). Kansas State University, 2011.

UPTON, E.; HALFACREE, G. **Meet the Raspberry Pi**. [S.l.]: John Wiley & Sons, 2012.

VISIOLI, A. **Practical PID control**. [S.l.]: Springer Science & Business Media, 2006.

ZAWBAA, H. M. et al. Computational intelligence modeling of the macromolecules release from plga microspheres—focus on feature selection. **PLoS One**, Public Library of Science San Francisco, CA USA, v. 11, n. 6, p. e0157610, 2016.

6 ANEXO A - CIRCUITO



Fonte: Autoria Própria.

7 ANEXO B - SCRIPT DE IDENTIFICAÇÃO EM PYTHON

```
import numpy as np
import csv
import time
import Adafruit_ADS1x15
import RPi.GPIO as gpio
import threading
import signal
from queue import Queue

gpio.setwarnings(False)
gpio.setmode(gpio.BOARD)

#*** Entradas e saídas do Raspberry ***
#Saída 1: pino 33
#Saída 2: pino 32
#Entrada 1: A0 do ADC
#Entrada 2: A1 do ADC
pino_saida1=33;
pino_saida2=32;

#offset 4mA IN1 = 8304
#offset 20mA IN1 = 26544

#offset 4mA IN2 = 7872
#offset 20mA IN2 = 26224

#Inicializa as duas saídas
gpio.setup(pino_saida1,gpio.OUT)
gpio.setup(pino_saida2,gpio.OUT)

saida1=gpio.PWM(pino_saida1,300)
saida2=gpio.PWM(pino_saida2,300)
```

```
saida1.start(0)
saida2.start(0)

# Ceia uma instancia do ADCADS1115 ADC (16-bit).
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
values = [0]*4
for i in range(4):
    values[i] = adc.read_adc(i, gain=GAIN)
print('Entradas Analogicas: | {0:>6} | {1:>6} | {2:>6} | {3:>6} |'.format(*values))

#importa o sinal PRMLS
lista = np.genfromtxt("teste_3.csv",dtype="int", delimiter=",")

#Quantidade de amostras a serem realizadas
size_prmls=5
#Quantidade de amostrar por linhado PRMLS
n_repete=1000
#Quantidade total de linhas
linhas=size_prmls*n_repete
print('Quantidade de amostras:'+str(linhas))

#Cria um novo vetor para armazenar os dados coletados
lista2=np.zeros((linhas+1,5))
lista2[0,0]=time.time_ns()
s1=0
s2=0
out1=10
out2=10

exit_event = False
#Função que roda em um thread separado e realiza a
#leitura e escritas das entradas do raspberry
def worker(comm_thread):
    #obtem o timestamp do inicio da aquisição
    #usado para calcular o timestamp das aquisições
    print("Thread iniciada\n")
    while True:
        #Le os sinais do ADC
        th_in1=(adc.read_adc(0, gain=GAIN))
        th_in2=(adc.read_adc(1, gain=GAIN))
```

```
        #Envia para fora da thread os valores lidos
        comm_thread.put([time.time_ns(), th_in1, th_in2])

        #Desliga a thread
        if exit_event:
            break
#Fim da thread


#***** Programa principal *****
#Cria uma comunicação entre thread e programa principal
q1 = Queue()
#Cria um thread para realizar a leitura dos IOs
t = threading.Thread(target=worker, args=(q1, ))

#Inicia a thread
t.start()

#indice
k=1
#Realiza a aquisição
for i in range(0,size_prmls):
    print(lista[i,:])
    for j in range(0,n_repete):

        #Recebe os sinais dos sensores lidos pela thread
        fila_thread=q1.get()
        timestamp_thread = fila_thread[0]
        in1 = fila_thread[1]
        in2 = fila_thread[2]

        #Timestamp
        lista2[k,0]=timestamp_thread

        #Entradas
        lista2[k,3]=in1
        lista2[k,4]=in2

        #escreve na saida 1
        s1=lista[i,0]
        lista2[k,1]=s1
        saida1.ChangeDutyCycle(s1)
```

```
            #escreve na saida 2
            s2=lista[i,1]
            lista2[k,2]=s2
            saida2.ChangeDutyCycle(s2)

            k=k+1

        print(lista2[k-1,:])
        print(k)

print("fim")
#Desliga a thread
exit_event=True
t.join()
print("Thread desligada")

#Salva os dados em um arquivo .csv para analise posterior
np.savetxt("aq3.csv",lista2,fmt="%d",delimiter=",")
```

8 ANEXO C - SCRIPT DE IDENTIFICAÇÃO EM MATLAB

```
clear
close all
clc
%% Visualização dos dados das aquisições
% Primeira parte: Dados de entradas e saídas utilizando um sinal PRMLS nas
% válvulas, o tempo entre amostras é de 0.0199s.:

% Período de amostragem das aquisições em segundos:
Ts=0.0199;%s

% lê os arquivos em .csv e gera vetores no matlab
[data4, t_4, dados_aq1]=carregar_arquivo('aq1.csv',Ts, false, 'Aquisição 1');
[data5, t_5, dados_aq2]=carregar_arquivo('aq2.csv',Ts, false, 'Aquisição 2');
[data6, t_6, dados_aq3]=carregar_arquivo('aq3.csv',Ts, false, 'Aquisição 3');
[data7, t_7, dados_aq4]=carregar_arquivo('aq4.csv',Ts, false, 'Aquisição 4');

[data8, t_8, dados_aq5]=carregar_arquivo('step3.csv',Ts, false, 'Aquisição 5');

%% Gerando os vetores de dados
% Vetores utilizados para estimar o modelo e para testar o modelo
% Primeira saída(y1):
% Entradas: u1
%
% Segunda saída(y2):
% Entradas: u2, y1

%Dados para a modelagem e validação da primeira saída:
data_y1_est=iddata(dados_aq1(:,4), dados_aq1(:,2),  Ts);
data_y1_est.InputName=['u1'];
data_y1_est.OutputName=['y1'];

data_y1_val=iddata(dados_aq2(:,4), dados_aq2(:,2),  Ts);
```

```
data_y1_val.InputName=['u1'];
data_y1_val.OutputName=['y1'];

%Dados para a modelagem e validação da segunda saída:
data_y2_est=iddata(dados_aq1(:,5), [dados_aq1(:,2), dados_aq1(:,3)],  Ts);
data_y2_est.InputName=['u1';'u2'];
data_y2_est.OutputName=['y2'];

data_y2_val=iddata(dados_aq2(:,5), [dados_aq2(:,2), dados_aq2(:,3)],  Ts);
data_y2_val.InputName=['u1';'u2'];
data_y2_val.OutputName=['y2'];

%Dados para a validação do modelo completo:
data_y_val=iddata([dados_aq2(:,4),dados_aq2(:,5)], [dados_aq2(:,2),
dados_aq2(:,3)],  Ts);
data_y_val.InputName=['Válvula vazão';'Vávlula nível'];
data_y_val.OutputName=['Vazão';'Nível'];
data_y_val.Name='Dados';

data_y_Step=iddata([dados_aq5(:,4),dados_aq5(:,5)], [dados_aq5(:,2),
dados_aq5(:,3)],  Ts);
data_y_Step.InputName=['Válvula vazão';'Vávlula nível'];
data_y_Step.OutputName=['Vazão';'Nível'];
data_y_Step.Name='Dados';


%% Calculo do modelo
% Para a obtenção de um modelo bom é realizado o calculo de combinações de
% polos e zeros, e por fim, o modelo escolhido é o que possuir o menor MSE.
%
% Foram testados modelos com até 10 polos e 9 zeros.

maxPolos=10;
maxZeros=maxPolos-1;
modo='MSE';
np=(1:maxPolos)';
nz=(0:maxZeros)';
ind=1;

%Gera a combinação de polos e zeros
for i=1:maxPolos
    for j=0:i-1
        npnz(ind,1)=i;
```

```
        npnz(ind,2)=j;
        ind=ind+1;
    end
end
disp(length(npnz))

%Calcula os modelos
for i=1:length(npnz)
    % Calcula os modelos:
    modelo_y1_TF= tfest(data_y1_est,npnz(i,1),npnz(i,2));
    modelo_y2_TF= tfest(data_y2_est,npnz(i,1),npnz(i,2));

    % Testa os modelos:
    [y1_val, y1_NRMSE(i), y1_x0] = compare(data_y1_val,modelo_y1_TF);
    [y2_val, y2_NRMSE(i), y2_x0] = compare(data_y2_val,modelo_y2_TF);

    % Calcula o MSE:
    fit_y1(i,1)=npnz(i,1);
    fit_y1(i,2)=npnz(i,2);
    fit_y1(i,3)=goodnessOfFit(y1_val.y, data_y1_val.y,modo);

    fit_y2(i,1)=npnz(i,1);
    fit_y2(i,2)=npnz(i,2);
    fit_y2(i,3)=goodnessOfFit(y2_val.y, data_y2_val.y,modo);
    disp(i)

end

%% Melhor função de transferencia

%Encontra a melhor combinação de polos e zeros da saída 1
[M1,I1] = min(fit_y1(:,3));
bestNP_1=fit_y1(I1,1);
bestNZ_1=fit_y1(I1,2);
% bestNP_1=6;
% bestNZ_1=5;

%Encontra a melhor combinação de polos e zeros da saída 2
[M2,I2] = min(fit_y2(:,3));
bestNP_2=fit_y2(I2,1);
bestNZ_2=fit_y2(I2,2);
% bestNP_2=4;
% bestNZ_2=3;
```

```
%Exibe os melhores dados
disp(fit_y1(I1,:))
disp(fit_y2(I2,:))

%Monta as funções de transferencia
best_Y1=tfest(data_y1_est,bestNP_1,bestNZ_1);

best_Y2=tfest(data_y2_est,bestNP_2,bestNZ_2);

best_Y(1,1)=tf(best_Y1.Numerator,best_Y1.Denominator);
best_Y(1,2)=0;
best_Y(2,1)=tf(best_Y2(1,1).Numerator,best_Y2(1,1).Denominator);
best_Y(2,2)=tf(best_Y2(1,2).Numerator,best_Y2(1,2).Denominator);
best_Y.OutputName=['Vazão';'Nível'];
best_Y.InputName=['Válvula vazão';'Vávlula nível'];
best_Y.Name='Modelo';

%% Compara os resultados

figure
compare(data_y_val,best_Y,'b')
title('')
xlabel('Tempo')

figure
compare(data_y_Step,best_Y,'b')
title('')
xlabel('Tempo')



%% Funções utilizadas
%
function [data, timeStamp, dados] = carregar_arquivo(arquivo,Ts, plotar, titulo)
x_0=load(arquivo);
timestamp_ref=x_0(1,1);
timestamp_aq0=(x_0(2:end,1)-timestamp_ref).*10^-9;
time_period_aq0=(x_0(2:end,1)-x_0(1:end-1,1)).*10^-9;


in1_offset_max=26544;
in2_offset_max=26224;
```

```
in1_offset_min=8304;
in2_offset_min=7872;

out1=x_0(2:end,2);  %Válvula 1
out2=x_0(2:end,3);  %Válvula 2
out2_complementar=100-out2;
in1=(((x_0(2:end,4)-in1_offset_min)./in1_offset_max).*100); %Sensor 1
in2=(((x_0(2:end,5)-in2_offset_min)./in2_offset_max).*100); %Sensor 2

t_max=length(in1)*Ts;
t_s=(0:Ts:(t_max-Ts))';   %Vetor de tempo para o grafico

data=iddata([in1 in2],[out1 out2],Ts);
eixo=[0 300 -5 105];
if plotar
    figure
    subplot(211)
    sgtitle(titulo)
    plot(t_s',out1)
    hold on
    plot(t_s',in1)
    axis(eixo)
    hold off
    legend('Válvula','Vazão')
    ylabel('[%]')
    xlabel('Tempo [s]')
    title('Malha de vazão')

    subplot(212)
    plot(t_s',out2)
    hold on
    plot(t_s',in2)
    axis(eixo)
    hold off
    ylabel('[%]')
    xlabel('Tempo [s]')
    legend('Válvula','Nível')
    title('Malha de nível')

    saveas(gcf,strcat(titulo,'.svg'))

end
```

```
timeStamp=timestamp_aq0;

%       t     u1      u2    y1    y2
dados=[t_s, out1, out2, in1, in2];
end
```

9 ANEXO D - SCRIPT DE COMUNICAÇÃO EM PYHTON - RASPBERRY

```
# encoding: utf-8

# Importa as bibliotecas necessarias
import time
import Adafruit_ADS1x15
import RPi.GPIO as gpio
import threading
import signal
import socket
import struct
import decimal
from queue import Queue
from pyModbusTCP.server import ModbusServer, DataBank
from simple_pid import PID
import sys


# Configura os pinos de IO
gpio.setwarnings(False)
gpio.setmode(gpio.BOARD)

#*** Entradas e saídas do Raspberry ***
#Saída 1: pino 33
#Saída 2: pino 32
#Entrada 1: A0 do ADC
#Entrada 2: A1 do ADC
pino_saida1=33;
pino_saida2=32;
chave = 16

in1_offset_max=26544
in1_offset_min=8304
```

```
in2_offset_max=26224
in2_offset_min=7872

#offset 4mA IN1 = 8304
#offset 20mA IN1 = 26544

#offset 4mA IN2 = 7872
#offset 20mA IN2 = 26224

#Inicializa as duas saídas
gpio.setup(pino_saida1,gpio.OUT)
gpio.setup(pino_saida2,gpio.OUT)
gpio.setup(chave,gpio.IN)

saida1=gpio.PWM(pino_saida1,300)
saida2=gpio.PWM(pino_saida2,300)
statusChave = gpio.input(chave)

saida1.start(0)
saida2.start(0)

# Cria uma instancia do ADCADS1115 ADC (16-bit).
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
values = [0]*4
for i in range(4):
    values[i] = adc.read_adc(i, gain=GAIN)


# Inicializa as variaveis utilizadas
out1=0
out2=0
in1=0
in2=0
timestamp_thread=0

#Função que roda em um thread separado e realiza a
#leitura e escritas das entradas do raspberry
exit_event = False
def worker(comm_thread):
    #obtem o timestamp do inicio da aquisição
    #usado para calcular o timestamp das aquisições
```

```
    print("Thread iniciada\n")
    while True:
        th_in1=(((adc.read_adc(0, gain=GAIN))-in1_offset_min)/in1_offset_max)*100
        th_in2=(((adc.read_adc(1, gain=GAIN))-in2_offset_min)/in2_offset_max)*100
        comm_thread.put([time.time_ns(), th_in1, th_in2])


        #Desliga a thread
        if exit_event:
            break
#Fim da thread


def dword_to_float(word1, word2):
    float_out, = struct.unpack('>f',b''.join([word1.to_bytes(2,'big'),
    word2.to_bytes(2,'big')]))
    return float_out

#Comunicação UDP
def commUDP():
    print('UDP')
    servidorUDP = False

    #Define o IP e porta para a comunicação UDP
    localIP="10.2.0.100"
    localPort=20001

    #Define a quantidade de bytes que podem ser recebidos
    bufferSize=1024
    #Inicia a thread

    timestp=time.time_ns()
    statusChave_loc = gpio.input(16)
    print("inicio loop")
    while(statusChave_loc):
        try:
            # Define o protocolo a ser utilizado, no caso UDP
            #AF_INET = Internet
            #SOCK_DGRAM = UDP
            UDPServerSocket = socket.socket(family=socket.AF_INET,
            type=socket.SOCK_DGRAM)

            # Configura o IP e porta para a comunicacao
```

```
UDPServerSocket.bind((localIP, localPort))
print("Servidor UDP ligado\n")
UDPServerSocket.settimeout(1)
#Servidor UDP ligado
servidorUDP = True
contador=0


while(True):
    print("Aguardando receber dados para iniciar o envio\n")
    fila_thread=q1.get()
    timestamp_thread = fila_thread[0]
    print(timestamp_thread)
    #Recebe pacotes via UDP
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
    #Dados recebidos via UDP
    message = bytesAddressPair[0]
    #Endereco IP que enviou dados para o raspberry
    address = bytesAddressPair[1]
    print(message)
    clientIP  = "Client IP Address:{}".format(address)
#     print(clientIP)
#     print(address)
    #Converte os bytes recebidos em um tuple de floats
    messageFloat=struct.unpack(">ff",message)
    print(messageFloat)
    #Arruma os valores recebidos para ter apenas 3 casas decimais
    out1_UDP_f=round(decimal.Decimal(messageFloat[0]),3)
    out2_UDP_f=round(decimal.Decimal(messageFloat[1]),3)
#     address=('10.2.0.110', 53645)
    time1=time.time_ns()

    while True:
        statusChave_loc2 = gpio.input(chave)
        #Termina a comunicacao
        if statusChave_loc2 == False:
            UDPServerSocket.close()
            print('Servidor UDP desligado')
            break

       #Tratamento dos dados recebidos
        if (out1_UDP_f ≥ 100.0):
            out1_UDP_f = 100.0
```

```
if (out2_UDP_f ≥ 100.0):
    out2_UDP_f = 100.0

if (out1_UDP_f ≤ 0.0):
    out1_UDP_f = 0.0

if (out2_UDP_f ≤ 0.0):
    out2_UDP_f = 0.0


#Valores para escrever na saida recebidos via UDP
out1=out1_UDP_f
out2=out2_UDP_f

#Escreve na saida PWM os valores recebidos via UDP
saida1.ChangeDutyCycle(out1)
saida2.ChangeDutyCycle(out2)

#Recebe os sinais dos sensores lidos pela thread
fila_thread=q1.get()
timestamp_thread = fila_thread[0]
in1 = fila_thread[1]
in2 = fila_thread[2]

#Vetor de variaveis a serem enviadas
dadosEnviaUDP=[timestamp_thread, in1, in2, out1, out2]

#Converte as variaveis do tipo float para bytes
bytesToSend=bytearray(struct.pack("Qffff", dadosEnviaUDP[0],
dadosEnviaUDP[1], dadosEnviaUDP[2], dadosEnviaUDP[3],
dadosEnviaUDP[4]))

# Envia dados via UDP para o endereco que solicitou
UDPServerSocket.sendto(bytesToSend, address)

#Recebe pacotes via UDP
bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
#Dados recebidos via UDP
message = bytesAddressPair[0]
#Endereco IP que enviou dados para o raspberry
address = bytesAddressPair[1]
#Converte os bytes recebidos em um tuple de floats
```

```
                    messageFloat=struct.unpack(">ff",message)
                    #Arruma os valores recebidos para ter apenas 3 casas decimais
                    out1_UDP_f=round(decimal.Decimal(messageFloat[0]),1)
                    out2_UDP_f=round(decimal.Decimal(messageFloat[1]),1)

                    print(dadosEnviaUDP)
                    contador+=1

            except socket.timeout:
                print("UDP: Sem conexão")
                if(gpio.input(chave)):
                    continue
                else:
                    UDPServerSocket.close()
                    break
            except:
                print("UDP: Erro")
                break

    print('UDP desligado')
    UDPServerSocket.close()
    #Desliga a thread
    exit_event=True
    #q1.join()
    #t.join()


def commModbus():
    print('funcao Modbus')

    #Cria uma instancia do ModbusServer
    server=ModbusServer("10.2.0.100",port=12345,no_block=True,ipv6=False )

    try:

        pid1 = PID(1, 0.1, 0.05, setpoint=1)
        pid2 = PID(1, 0.1, 0.05, setpoint=1)

        pid1.output_limits = (0, 100)
        pid2.output_limits = (0, 100)

        pid1.sample_time = 0.01
        pid2.sample_time = 0.01
```

```
print("Iniciando servidor Modbus")
server.start()
print("Servidor Modbus iniciado")

while (gpio.input(chave) == False):

    #Recebe os dados via Modbus
    dadosRecebidos = DataBank.get_words(100,16)
    modoOperacao = DataBank.get_words(500,1)

    #Organiza os dados recebidos
    setpoint1 = int(dword_to_float(dadosRecebidos[0],dadosRecebidos[1]))
    setpoint2 = int(dword_to_float(dadosRecebidos[2],dadosRecebidos[3]))

    Kp1=dword_to_float(dadosRecebidos[4], dadosRecebidos[5])
    Ti1=dword_to_float(dadosRecebidos[6], dadosRecebidos[7])
    Td1=dword_to_float(dadosRecebidos[8], dadosRecebidos[9])

    Kp2=dword_to_float(dadosRecebidos[10], dadosRecebidos[11])
    Ti2=dword_to_float(dadosRecebidos[12], dadosRecebidos[13])
    Td2=dword_to_float(dadosRecebidos[14], dadosRecebidos[15])

    #Recebe os sinais dos sensores lidos pela thread
    fila_thread=q1.get()
    timestamp_thread = int(fila_thread[0])
    in1 = int(fila_thread[1])
    in2 = int(fila_thread[2])

    #in1=setpoint1
    #in2=setpoint2

    if modoOperacao[0]==1:
        pid1.auto_mode = True
        pid1.tunings = (Kp1,Ti1,Td1)
        pid2.tunings = (Kp2,Ti2,Td2)

        pid1.setpoint = setpoint1
        pid2.setpoint = setpoint2

        out1=pid1(in1)
        out2=pid2(in2)
        s1=[Kp2,Ti2,Td2, in2, setpoint2,out2]
```

```
            else:
                out1=setpoint1
                out2=setpoint2
            #Tratamento dos dados recebidos
            if (out1 ≥ 100.0):
                out1 = 100.0

            if (out2 ≥ 100.0):
                out2 = 100.0

            if (out1 ≤ 0.0):
                out1 = 0.0

            if (out2 ≤ 0.0):
                out2 = 0.0

            #Escreve na saida PWM os valores recebidos via UDP
            saida1.ChangeDutyCycle(out1)
            saida2.ChangeDutyCycle(out2)

            #Prepara o vetor de dados a serem enviados
            itimestamp_thread= int(timestamp_thread)
            x=itimestamp_thread.to_bytes(22, byteorder='little')

            #Envia os dados via modbus
            DataBank.set_words(200,x)
            DataBank.set_words(300,[in1])
            DataBank.set_words(301,[in2])
            DataBank.set_words(302,[out1])
            DataBank.set_words(303,[out2])

    except:

        print("Desligando servidor Modbus")
        server.stop()
        print("Servidor Modbus Desligado")

    print("Desligando servidor Modbus")
    server.stop()
    print("Servidor Modbus Desligado")

#***** Programa principal *****
udpAtivo = False
```

```
modbusAtivo = False
#Cria uma comunicação entre thread e programa principal
q1 = Queue()
#Cria um thread para realizar a leitura dos IOs
t = threading.Thread(target=worker, args=(q1, ))

t.start()

print('inicia o monitoramento da chave')

while True:
    #le o sinal da chave de seleção
    statusChave = gpio.input(16)

    if(statusChave):
        #Chave ligada: modo UDP
        if(udpAtivo == False):
            print('UDP selecionado')
            commUDP()
            udpAtivo = True
        else:
            print('UDP ja selecionado')

    else:
        #Chave desligada: modo Modbus
        if(udpAtivo):

            print('Modbus')
            commModbus()
            udpAtivo = False
        else:
            if(modbusAtivo):
                print('Modbus ativo')
            else:
                commModbus()
                print('Modbus Iniciado')
```

10 ANEXO E - SCRIPT DE COMUNICAÇÃO EM MATLAB - PC

```
clear all
close all
clc

inputArray=(load('inputSignal.csv'));
% plot(inputArray)
nAmostras=500;
indexSaida=1;
lastIndex=1;
%Cria um cliente udp e inicia a comunicação enviando os primeiros valores
u = udp('10.2.0.100',20001);
fopen(u);
fwrite(u,inputArray(1,:),'float32')

% fwrite(u,x,'float32')

i=1;
contador_Erro=0;
pause(2)
while true

    %Le os dados enviados via UDP
    A = fread(u,16,'uint8');

    %Se o vetor lido esta vazio, gera um erro
    if isempty(A)
        fprintf('Erro\n')
        contador_Erro=contador_Erro+1;
    else

        %Trata os dados lidos;
        C=uint8(A);
```

```
        if length(C)<24
            C=[C;zeros(24-length(C),1)]

        end
        dataBytes=C(9:end);

        %Timestamp o momento da aquisição
        timeStampBytes=C(1:8);
        timeStamp(i)=double(typecast(timeStampBytes,'int64'))*10^-9;
        t = datetime(timeStamp,'ConvertFrom','posixtime','TimeZone',
        'America/Sao_Paulo');

        y_rec(i,:) = typecast(dataBytes, 'single');

        fwrite(u,inputArray(indexSaida,:),'float32');

        fprintf('\nTimestamp: %s\n Setpoint: %f; Vazão: %f\n Setpoint: %f;
        Nivel: %f\n', t(i),y_rec(i,3),y_rec(i,1),y_rec(i,4),y_rec(i,2))

        inputVector(i,:)=inputArray(indexSaida,:);

        %Incrementa a quantidade de pontos lidos
        i=i+1;

        if indexSaida<length(inputArray)
            indexSaida=fix(i/nAmostras)+1;
        else
            break
        end
    end

    if contador_Erro > 10
        disp('10 erros, cancelando comunicação')
        break
    end
end
```