

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

RÔMULO CÉSAR CARDOSO FILHO

**MODELAGEM E IMPLEMENTAÇÃO DE CONTROLE DE NÍVEL EM SISTEMAS
DE TANQUES**

PONTA GROSSA

2022

RÔMULO CÉSAR CARDOSO FILHO

**MODELAGEM E IMPLEMENTAÇÃO DE CONTROLE DE NÍVEL EM SISTEMAS
DE TANQUES**

MODELING AND IMPLEMENTATION OF LEVEL CONTROL IN TANK SYSTEMS

Trabalho de Conclusão de Curso
como requisito parcial à obtenção do
título de Bacharel em Engenharia
Química, do Departamento
Acadêmico de Engenharia Química da
Universidade Tecnológica Federal do
Paraná, campus Ponta Grossa.
Orientador: Profº. Dr. Everton Moraes
Matos

**PONTA GROSSA
2022**



[4.0 Internacional](https://creativecommons.org/licenses/by/4.0/)

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

RÔMULO CÉSAR CARDOSO FILHO

**MODELAGEM E IMPLEMENTAÇÃO DE CONTROLE DE NÍVEL EM SISTEMAS
DE TANQUES**

Trabalho de Conclusão de Curso de Graduação
apresentado como requisito para obtenção do título
de Bacharel em Engenharia Química da
Universidade Tecnológica Federal do Paraná
(UTFPR).

Data de aprovação: 22/junho/2022

Everton Moraes Matos (Orientador)
Doutorado
Universidade Tecnológica Federal do Paraná

César Augusto Canciam
Doutorado
Universidade Tecnológica Federal do Paraná

Giane Gonçalves Lenzi
Doutorado
Universidade Tecnológica Federal do Paraná

**PONTA GROSSA
2022**

Para Helena, o exemplo de fé, amor e sabedoria que tenho o privilégio de chamar de avó.

AGRADECIMENTOS

Gostaria de agradecer a todos aqueles que me auxiliaram nesta jornada, tanto com apoio intelectual quanto emocional durante todos os momentos em que precisei de ajuda.

Agradeço a toda minha família, em especial a minha avó Helena e meu tio Jorge, que me inspiraram a trilhar esta jornada, sendo as pessoas quem eu tento me espelhar todos os dias de minha vida.

Aos meus queridos amigos Gabriel Teixeira Santos e Rodrigo Shoiti Simões, por todo o companheirismo e inspiração ao adentrar no mundo de controle de processos. Agradeço a todos que me auxiliaram no desenvolvimento deste projeto, em especial a Felipe Nishihara pela paciência e conselhos fornecidos.

E por fim, à Universidade Tecnológica Federal do Paraná e todas as pessoas de sua comunidade acadêmica, que me acolherem durante tantos anos e me forneceram tantos momentos felizes, dos quais carrego com carinho em minhas lembranças. Em especial, agradeço ao professor Dr. Everton Moraes Matos, por toda dedicação prestada em seu ensino e por todo o apoio e orientação neste trabalho.

RESUMO

Este trabalho teve como objetivo o estudo de técnicas de modelagem e implementação de um controlador PID de um sistema de tanques de água. A planta foi composta por dois tanques, uma bomba hidráulica e um sensor ultrassônico. Microcontroladores Arduino foram utilizados para estabelecer a comunicação Planta/Scilab-Xcos a fim de monitorar o sinal referente ao nível, bem como transmitir as ações de controle a serem executadas. Após construção do sistema e implementação do controlador PID, os ensaios demonstraram uma resposta rápida do controlador aos distúrbios aplicados, mantendo o sistema em estabilidade. Apesar da metodologia implementada ter se mostrado satisfatória, novos testes, com diferentes tipos de sensores e técnicas de sintonia do controlador podem ser realizados a fim de buscar melhores resultados no controle do processo.

Palavras-chave: automatização; controle de nível; Arduino; Scilab; Xcos; PID.

ABSTRACT

The purpose of this paper was the study of modeling methods and implementation of a PID controller in a water-level tank system. The plant was composed by two tanks, a hydraulic pump, and an ultrasonic sensor. Arduino microcontrollers were used to establish the communication Plant/Scilab-Xcos with the objective of monitoring the water level signal, as well as transmitting control actions to be executed. After building the system and implementing the PID controller, the tests demonstrated a quick response of the controller to applied disturbances, keeping the system in stability. Even though the methodology as shown good results, new tests, with different kinds of sensors and controlling design techniques should be performed in the pursue of better results in the process control.

Keywords: automatization; level control; Arduino; Scilab; Xcos; PID.

LISTA DE ILUSTRAÇÕES

GRÁFICOS

Gráfico 1 - Comportamento de um controlador proporcional de acordo com o ganho	19
Gráfico 2 - Controle PI para variados valores de T_i com K_c constante igual a 1	20
Gráfico 3 - Método de Curva de Reação por de Ziegler-Nichols	23
Gráfico 4 - Esquema de oscilação para sintonia em malha fechada	24
Gráfico 5 - Ensaio em malha aberta no sistema	38
Gráfico 6 - Relação Sinal PWM versus Vazão	40
Gráfico 7 - Simulação do sistema em malha fechada	41
Gráfico 8 - Oscilação sustentada	42
Gráfico 9 - Ensaio 1 do controle PID Arduino-Xcos com desvio no setpoint	45
Gráfico 10 - Ensaio 2 do controle PID Arduino-Xcos com desvio no setpoint ..	46
Gráfico 11 - Ensaio 3 do controle PID Arduino-Xcos com desvio no setpoint ..	46
Gráfico 12 - Ensaio 4 do controle PID Arduino-Xcos com desvio no setpoint ..	47
Gráfico 13 - Ensaio 1 do controle PID Arduino-Xcos com desvio na entrada ...	48
Gráfico 14 - Ensaio 1 do controle PID Arduino-Xcos com desvio na entrada ...	49
Gráfico 15 - Ensaio 3 do controle PID Arduino-Xcos com desvio na entrada ..	50
Gráfico 16 - Ensaio 4 do controle PID Arduino-Xcos com desvio na entrada ...	50

FOTOGRAFIAS

Fotografia 1 - Sistema de controle de nível completo	37
Fotografia 2 - Vista superior do sistema	37
Fotografia 3 - Detalhe dos Arduinos, Circuito RC e Driver PWM	37

FIGURAS

Figura 1 - Esquema de um sistema de controle em malha aberta	15
Figura 2 - Controle em malha fechada	15
Figura 3 - Representação de um diagrama de blocos	16
Figura 4 - Sistema de armazenagem de massa de primeira ordem com atraso	18
Figura 5 - Placa Arduino BlackBoard Uno R3	26
Figura 6 - Esquema do protótipo	27

Figura 7 - Bomba RS385	28
Figura 8 - Módulo Driver PWM 5V	28
Figura 9 - Sensor HC-SR04.....	29
Figura 10 - Esquema de funcionamento do sensor ultrassônico	30
Figura 11 - Interface gráfica Scilab-Xcos	31
Figura 12 - Detalhe do gerenciador de módulos Atoms com biblioteca para Arduino.....	32
Figura 13 - Exemplo de aquisição de temperatura ambiente diretamente pelo Xcos.....	32
Figura 14 - Monitor Serial da IDE Arduino com valores do nível e sinal PWM enviado ao circuito RC.....	34
Figura 15 - Diagrama de blocos para adquirir o sinal de tensão do sensor ultrassônico	35
Figura 16 - Sinal de tensão do circuito RC recebido pelo Xcos.....	35
Figura 17 - Ligações eletrônicas dos componentes do projeto.....	36
Figura 18 - Diagrama de blocos do sistema com PID implementado	43
Figura 19 - Diagrama de blocos ajustado com desvios na entrada.....	48

LISTA DE TABELAS

Tabela 1 - Resultados do teste de vazão da bomba versus sinal PWM	39
Tabela 2 - Parâmetros do PID sintonizado utilizando o método de Ziegler-Nichols em malha fechada	42

SUMÁRIO

1 INTRODUÇÃO	12
1.1 Objetivos	13
1.1.1 Objetivo Geral	13
1.1.2 Objetivos Específicos	13
2 FUNDAMENTAÇÃO TEÓRICA	14
2.1 Sistemas de Controle de Processos	14
2.2 Modelos de sistemas dinâmicos	16
2.3 Diagramas de blocos	16
2.4 Sistemas de primeira ordem	17
2.4.1 Modelagem de primeira ordem de um sistema de tanques de armazenamento	17
2.5 Tipos de controladores	18
2.5.1 Controlador Proporcional – P	18
2.5.2 Controlador Proporcional Integral – PI	20
2.5.3 Controlador Proporcional Diferencial Integral – PID	21
2.6 Sintonia de controladores	21
2.6.1 Método de sintonia de Ziegler-Nichols	22
<u>2.6.1.1 Método da curva de reação</u>	<u>22</u>
<u>2.6.1.2 Método de sintonia em malha fechada</u>	<u>23</u>
2.7 Implementação de controle automático de processos pela interação Arduino – Scilab	24
3 MATERIAIS E MÉTODOS	26
3.1.1 Microcontrolador Arduino Uno	26
3.1.2 Protótipo de tanque	27
3.1.3 Bomba hidráulica.....	28
3.1.4 Driver PWM	28
3.1.5 Sensor ultrassônico	29
3.1.6 Simulação em Scilab-Xcos	30
3.2 Métodos	31
3.2.1 Comunicando o Arduino e componentes ao Scilab-Xcos.....	31
<u>3.2.1.1 Adquirindo dados da distância do sensor ultrassônico pelo Xcos</u>	<u>32</u>
<u>3.2.1.1.1 Configurando a primeira placa Arduino</u>	<u>33</u>

3.2.1.1.2 Circuito RC e configuração da segunda placa Arduino	34
3.2.1.1.3 Acionando a bomba pelo Xcos	36
3.2.2 Modelagem matemática do sistema	38
<u>3.2.2.1 Vazão da bomba em diferentes potências</u>	<u>39</u>
<u>3.2.2.2 Função de transferência do tanque</u>	<u>40</u>
3.2.3 Sintonia do controlador.....	41
3.2.4 Montagem do diagrama de blocos do sistema	43
4 RESULTADOS.....	45
4.1 Ensaio com variação no setpoint	45
4.2 Ensaio com variação na entrada do sistema	47
5 CONCLUSÃO	52
REFERÊNCIAS.....	53
APÊNDICE A – CÓDIGO SENSOR ULTRASSÔNICO	56

1 INTRODUÇÃO

Com o crescente avanço da ciência e da engenharia, o controle automático de processos se faz cada vez mais necessário em nossa sociedade, onde além de ser indispensável em sistemas de navegação modernos, tornou-se presente em diversos processos industriais da atualidade. A magnitude de sua relevância pode ser vista em todo processo que necessite de controle de variáveis como temperatura, fluxo, umidade, pressão e viscosidade, variáveis estas que definem o fluxo de operação fabril e a dinâmica dos processos como um todo (OGATA, 2010).

Ainda segundo Ogata (2010), o entendimento da teoria de controle automático de processos e sua aplicação na prática geram grandes benefícios, como diminuição de custos, melhoria na qualidade e aumento de produção, se torna interessante conhecermos esse campo a fundo.

Um dos métodos de controle mais implementados em processos industriais é o controlador PID. Isso se deve ao fato dele ser capaz desempenhar funções importantes para o desempenho e controle do processo: ele pode eliminar o erro estacionário, comum em controladores proporcionais, pela ação integrativa e prever o comportamento do sistema pela sua ação derivativa. O controle PID é frequentemente combinado com programação, seletores, e funções de blocos a fim de criar sistemas de controle automáticos complexos, sendo ele na escala hierárquica das estratégias de controle, o mais baixo. Assim, o controlador PID pode ser considerado o “feijão com arroz” da engenharia de controle de processos, sendo uma arma fundamental no arsenal de todo engenheiro (ÄSTROM; HÄGGLUND, 1995).

Comumente se faz necessário a utilização de softwares e hardwares específicos para a implementação de um controle automático de processos, a fim de simular o processo por meio de modelagem matemática do sistema e de sintonização dos parâmetros do controlador utilizado. Muitos destes softwares, no entanto, não são de livre acesso, dificultando assim o acesso ao aprendizado e prática do controle de processos (BARRETO; ESCOBAR, 2014).

Uma boa abordagem para o aprendizado do controle de processos é a realização de projetos onde se projeta e constrói as próprias unidades, otimizando e buscando soluções práticas e técnicas para os problemas a serem trabalhados no projeto (FELDER, 2012).

Diante disso, o seguinte trabalho se tratou do projeto, construção e controle PID do nível de um sistema de tanques, utilizando a interface gratuita e de fácil acesso do software Scilab e sua ferramenta de modelagem e simulação integrada Xcos.

Como hardware, foi utilizado a plataforma Arduino, um microcontrolador de código aberto de baixo custo usado em projetos eletrônicos e aquisição de dados, controle e monitoramento.

1.1 Objetivos

Os objetivos desse trabalho estão divididos em gerais e específicos, os quais estão citados abaixo.

1.1.1 Objetivo Geral

Construir e implementar um sistema de controle de nível automático PID em tanques.

1.1.2 Objetivos Específicos

Este trabalho possui como objetivos específicos:

- Realizar a modelagem matemática do sistema de controle de nível a ser trabalhado;
- Integrar a ferramenta gratuita Xcos presente no *software* Scilab com a plataforma de código aberto Arduino;
- Utilizar de sensores eletrônicos a fim de estabelecer uma conexão entre o sistema físico e virtual.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Controle de Processos

O sistema de processos de uma planta química engloba um arranjo de diferentes operações unitárias, como trocadores de calor, colunas de destilação, evaporadores, sistemas de tanques, entre outros, que interagem entre si de maneira sistemática e racional. O objetivo no processo químico é, em suma, uma otimização da integração entre esses processos, a fim de obter os parâmetros de operação desejados, de maneira segura, eficiente e econômica (STEPHANOPOULOS, 1984).

O objetivo do controle de processos é regular as variações de fatores fundamentais para as operações a serem realizadas, sendo esses desvios, oscilações em temperatura, pressão, nível, vazão, entre outros limites operacionais que venham a ter relevância no processo. Variações são recorrentes no processo industrial, então medidas de controle devem ser tomadas a fim de atingir os requisitos de produção. (SMITH; CORRIPIO, 1997).

Antes de continuar, é interessante definir alguns conceitos básicos de controle de processos. Utilizando as definições de Ogata (2010), define-se:

Processo, como sendo uma progressão natural da operação, por meio de modificações sistemáticas e graduais a fim de se obter um resultado esperado.

Sistemas são a combinação de peças atuantes que interagem entre si a fim de atingir um objetivo.

Variável controlada de processo: é o aspecto a ser regulado em um valor determinado.

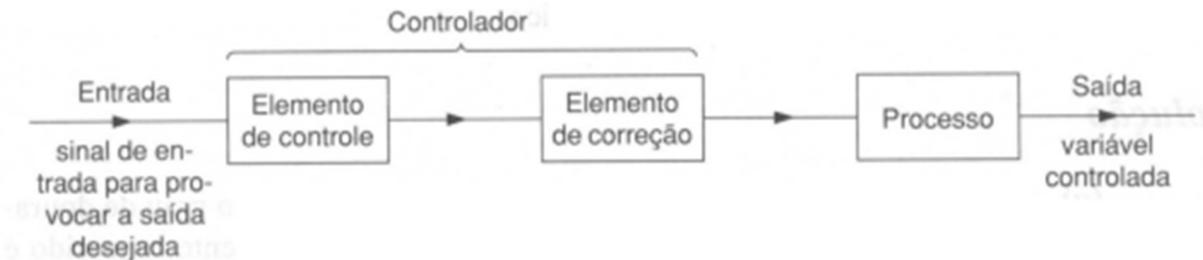
Ponto fixo, também conhecido como *setpoint*, é um valor especificado para a variável do processo.

Distúrbios, são perturbações que tendem a afetar o valor de saída do sistema, o deslocando do ponto fixo.

Sistema de controle em malha aberta: um sistema de controle no qual a saída não tem efeito na ação de controle, ou seja, a entrada terá sempre uma condição de operação imutável. Um exemplo é o controle de luzes em um semáforo por tempo. A variável a ser controlada, a passagem de carros, não interfere no funcionamento do semáforo, portanto o processo deve ser operado configurado por calibração prévia.

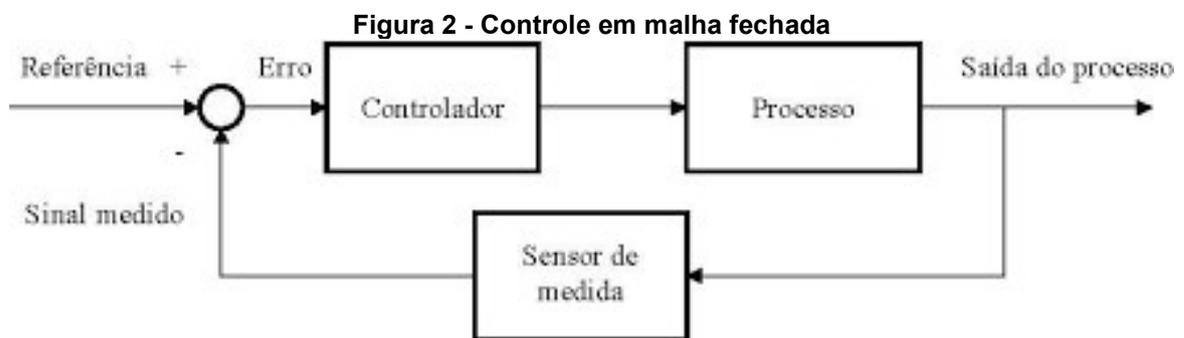
Na Figura 1, pode-se observar um diagrama de blocos representando uma malha aberta.

Figura 1 - Esquema de um sistema de controle em malha aberta



Fonte: Adaptado de Ogata (2010)

Sistema de controle em malha-fechada: um sistema no qual o sinal de saída altera o controle do processo, ou seja, o processo é realimentado. A diferença entre o sinal de saída e a entrada é utilizado a modo de atingir o valor desejado para o processo, reduzindo o erro atuante. A Figura 2 é mostrado um diagrama de controle de um sistema em malha fechada.



Fonte: Oliveira (1999)

As peças fundamentais para a realização do controle por malha fechada operam de acordo com o diagrama de blocos apresentado na Figura 2, estas sendo os sensores, um controlador, o comparador e um elemento final de controle ou atuador

O comparador afere o erro entre a variável medida pelos sensores utilizados no processo com o ponto fixo definido. Essa diferença é utilizada pelo controlador a fim de modular a resposta transferida para o atuador do sistema (OGATA, 2010).

2.2 Modelos de sistemas dinâmicos

O ponto de partida em uma análise de um sistema é sua modelagem. Uma vez que esse modelo é obtido, é possível utilizar-se de ferramentas analíticas ou digitais para fins de análise do sistema. Para grande parte dos sistemas dinâmicos é possível representá-los por equações diferenciais, em que a resposta desse sistema, dada certa entrada, pode ser encontrada resolvendo essas equações (OGATA, 2010).

Todavia, a maioria dos sistemas apresenta comportamento não linear, sendo necessário utilizar de aproximações lineares para aplicação da transformada de Laplace, obtendo assim as relações de entrada-saída como *função de transferência* (DORF E BISHOP, 2009).

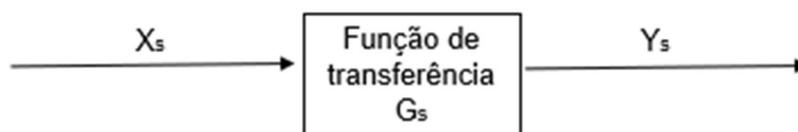
Admitindo condições iniciais nulas, a Equação 1 corresponde a função de transferência G_s , definida pela transformada de Laplace da saída do sistema Y_s , sobre a transformada da entrada X_s .

$$G_s = \frac{\mathcal{L} \text{ saída}}{\mathcal{L} \text{ entrada}} = \frac{Y_s}{X_s} \quad (1)$$

2.3 Diagramas de blocos

Geralmente um sistema de controle apresenta diversos componentes. Com o objetivo de explicitar as funções de cada um, é comum ser utilizado um diagrama específico chamado “diagrama de blocos”, como apresentado na Figura 3. Em um diagrama de blocos, os componentes do sistema e suas variáveis são conectadas entre si através de “blocos funcionais”, que são símbolos para operações matemáticas sobre o sinal de entrada em direção ao bloco que produz a saída do sinal (OGATA, 2010).

Figura 3 - Representação de um diagrama de blocos



Fonte: Autoria Própria (2022)

Com a construção de diagramas de blocos, o sistema pode ser visualizado de maneira direta e simplificada, onde se torna mais fácil fazer um fluxograma do

processo como um todo ao conectar os blocos dos componentes de acordo com a fluxo do sinal (OGATA, 2010).

2.4 Sistemas de primeira ordem

Segundo Stephanopoulos (1984), um sistema de primeira ordem é aquele cuja saída $y(t)$ pode ser modelada por uma EDO de primeira ordem. Esse sistema linear, ou linearizado pode ser descrito por

$$a_1 \frac{dy}{dt} + a_0 y = b f(t) \quad (2)$$

em que $f(t)$ é a entrada do processo. Se $a_0 \neq 0$, temos:

$$\frac{a_0}{a_1} \frac{dy}{dt} + y = \frac{b}{a_0} f(t) \quad (3)$$

Definindo $\frac{a_0}{a_1} = \tau$ e $\frac{b}{a_0} = K$, chegamos na equação que define o sistema de primeira ordem:

$$\tau \frac{dy}{dt} + y = K f(t) \quad (4)$$

em que τ é definido como sendo a constante de tempo do processo e K como o ganho estático do processo.

Tomando as condições iniciais $y(0) = 0$ e $f(0) = 0$ e aplicando a transformada de Laplace na equação 4, chega-se à função de transferência de sistema de primeira ordem

$$G_s = \frac{y_s}{f_s} = \frac{K}{\tau s + 1} \quad (5)$$

Processos de primeira ordem são definidos pela sua capacidade de armazenar energia, massa ou momento e pela resistência associada ao fluxo de massa ou energia ou pelo momento ao atingir a capacidade. Logo um sistema de tanques, que tem a capacidade de armazenar líquidos ou gases pode ser modelado como um sistema de primeira ordem (STEPHANOPOULOS, 1984).

2.4.1 Modelagem de primeira ordem de um sistema de tanques de armazenamento

O balanço de massa para um sistema de tanques com capacidade de armazenar massa é dado por

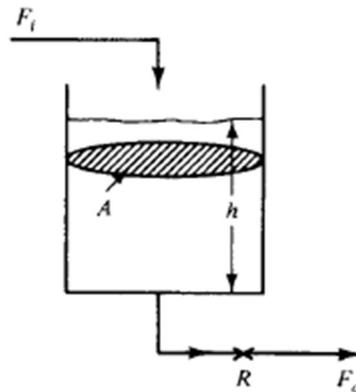
$$A \frac{dh}{dt} = F_i - F_o = F_i - \frac{h}{R} \quad (6)$$

tal que a saída F_o é linearmente relacionada à pressão do nível de água h , por meio da resistência fornecida pelo tubo ou válvula de saída. A Equação 6 pode ser rearranjada como

$$AR \frac{dh}{dt} + h = RF_i \quad (7)$$

em que A é a área da secção transversal do tanque. Um esquema representando um sistema de armazenagem de massa pode ser visto na Figura 4.

Figura 4 - Sistema de armazenagem de massa de primeira ordem com atraso



Fonte: Adaptado de Stephanopoulos (1984)

Da Equação 7, a equação de transferência será dada por

$$G_s = \frac{h_s}{F_{is}} = \frac{K}{\tau s + 1} = \frac{R}{ARs + 1} \quad (8)$$

em que a constante de tempo do processo é igual a AR e o ganho do processo equivale à resistência R .

2.5 Tipos de controladores

Modelos diferentes de controladores existem e devem ser selecionados de acordo com cada tipo de processo a ser controlado. Entre os controladores mais básicos existem o P, PI e PID.

2.5.1 Controlador Proporcional – P

O controlador proporcional é o tipo mais simples de todos. Este atua de maneira proporcional ao erro, de acordo com a Equação 9:

$$c(t) = K_c e(t) + C_s \quad (9)$$

Em que:

- $c(t)$ é a variável a ser manipulada;
- K_c é o ganho proporcional do controlador;
- $e(t)$ é o erro associado ao sinal;
- C_s é o valor da saída do controlador, por exemplo $c(t) = C_s$ quando o erro é igual a zero.

O controlador proporcional é descrito pelo seu ganho proporcional K_c , em que ao passo que este aumenta, maior será a sensibilidade do controlador aos erros associados ao sinal $e(t)$ (STEPHANOPOULOS, 1984).

Definindo o desvio do sinal como

$$c'(t) = c(t) - C_s \quad (10)$$

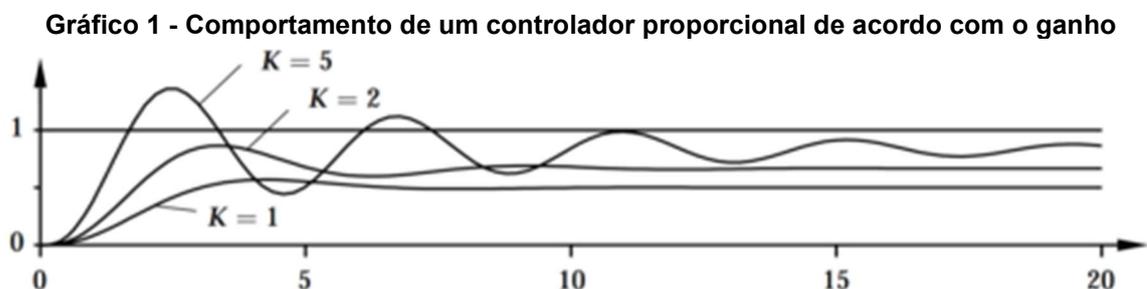
e tomando

$$c'(t) = K_c e(t) \quad (11)$$

é obtido, após aplicar a transformada de Laplace a função de transferência para o controlador proporcional

$$G_c(s) = K_c \quad (12)$$

A ação do controlador proporcional infere, quando utilizado isoladamente, um desvio em relação ao *setpoint*, se estabilizando em um novo ponto de equilíbrio que apresenta erro, assim como pode ser visto no Gráfico 1. Esse regime é conhecido como *off-set* (ÄSTROM; HÄGGLUND, 1995).



Fonte: Ästrom e Hägglund (1995)

Percebe-se que ao aumentar o valor do ganho proporcional, menor será o *off-set*, no entanto, ocorrerá maior presença oscilação no sistema (ÄSTROM; HÄGGLUND, 1995).

Por apresentar esse *off-set* característico, o controlador P não é frequentemente utilizado, uma vez que há, geralmente, a necessidade de controlar o processo em seu *setpoint* (SMITH; CORRIPIO, 1997). No entanto, como o projeto desenvolvido nesse trabalho é um controle de nível, em que o empecilho de um *off-set* seria a secagem ou transbordamento do tanque, o controlador P poderia ser suficiente.

2.5.2 Controlador Proporcional Integral – PI

O sinal atuante do controlador PI é correlacionado com o erro pela equação

$$c(t) = K_c * e(t) + \frac{K_c}{\tau_I} \int_0^t e(t) dt + C_s \quad (13)$$

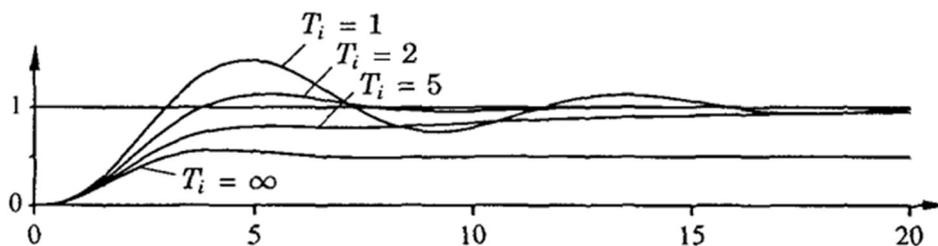
em que τ_I é o tempo integral, ou de repetição, dado geralmente em minutos. Esse fator de repetição é um parâmetro ajustado e é comumente chamado de *minutos por repetição*. Alguns fabricantes não calibram seus controladores por esse termo, mas sim pelo seu termo recíproco $1/\tau_I$, o qual é chamado de *repetições por minuto* ou *taxa de atualização* (STEPHANOPOULOS, 1984).

Da Equação 13, pode-se aplicar a transformada de Laplace, chegando à função de transferência:

$$G_c(s) = K_c \left(1 + \frac{1}{\tau_I s} \right) = K_c + \frac{K_i}{s} \quad (14)$$

A ação integral causa a saída do controlador $c(t)$ a variar conquanto o erro persistir no processo. Logo, esse tipo de controlador pode eliminar até erros pequenos. O gráfico 2 mostra um exemplo de controle PI.

Gráfico 2 - Controle PI para variados valores de T_i com K_c constante igual a 1



Fonte: Åstrom e Hägglund (1995)

Para um controle PI, a resposta é mais rápida para valores menores de T_i apresentando maiores oscilações no processo. Ao elevar o T_i a um valor infinito, o processo se assemelha a um controle puramente proporcional. (ÄSTROM; HÄGGLUND, 1995).

2.5.3 Controlador Proporcional Diferencial Integral – PID

Além de apresentar em seu funcionamento a ação proporcional e integral, o controlador PID possui a ação derivativa. Esta é capaz de prever o comportamento do sistema, aplicando uma velocidade de correção relativa à velocidade do desvio, ou seja, quanto maior à discrepância entre o valor medido da variável controlada e do *setpoint*, mais rápida a sua atuação, quanto menor, mais lenta a atuação será (GONÇALVES, 2003).

A resposta desse controlador é dada pela Equação 15:

$$c(t) = K_c e(t) + \frac{K_c}{T_i} \int_0^t e(t) dt + K_c \tau_d \frac{de}{dt} + C_s \quad (15)$$

em que τ_d é a constante derivativa dada em minutos. Da Equação 15 pode-se obter a sua função de transferência,

$$G_c(s) = K_c \left(1 + \frac{1}{T_i s} + \tau_d s \right) \quad (16)$$

Em um sistema em que há erro constante a ação derivativa não atua, já que $de/dt = 0$. Portanto a ação derivativa por si só é incapaz de eliminar o *off-set*. Em um sistema com respostas com muito ruído e pouco erro ele pode utilizar grande capacidade operacional, uma vez que irá computar extensas derivadas mesmo senso desnecessário. Por esse motivo, para a maioria dos processos, o controle PI já se mostra suficiente (STEPHANOPOULOS, 1984).

2.6 Sintonia de controladores

Para o funcionamento estável do sistema, os diversos parâmetros do controlador devem ser ajustados. Ao designar parâmetros coerentes com o processo, pode-se esperar boa estabilidade do sistema, mesmo perante ação de falhas e distúrbios. Esse procedimento para encontrar esses valores é chamado sintonização (ÄSTROM; HÄGGLUND, 1995).

Vários métodos existem e são viáveis para implementar a sintonia de controladores, um não sendo necessariamente melhor do que outro, devendo se analisar cada processo individualmente (SMITH; CORRIPIO, 1997).

2.6.1 Método de sintonia de Ziegler-Nichols

O estudo pioneiro para o procedimento de sintonia de controladores foi desenvolvido por Ziegler-Nichols, que publicaram em 1942 duas metodologias principais, uma sendo realizada em sistemas de controle de malha fechada e o outro com controle em malha aberta (OGATA, 2010).

2.6.1.1 Método da curva de reação

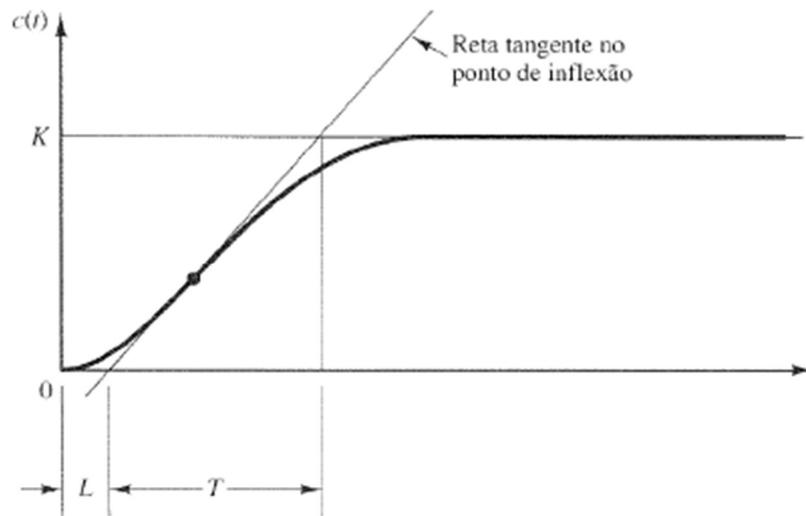
No método de malha aberta, também conhecido por *curva de reação*, se aplica um distúrbio em formato de degrau a partir do estado estacionário do sistema a fim de aproximá-lo de um sistema de primeira ordem com *tempo morto* (OGATA, 2010).

De acordo com Smith e Corripio (1997), para utilizar esse método o sistema deve apresentar um comportamento do tipo “*curva S*”. A curva S é definida por duas constantes, o tempo de resposta do sistema, ou *tempo morto* “L” e pela constante de tempo “T”, em que sua função de transferência é dada por:

$$G(s) = \frac{K e^{-Ls}}{T_s + 1} \quad (17)$$

Para conhecer os parâmetros do sistema de primeira ordem com tempo morto, traça-se um tangente no ponto de inflexão do sistema a fim de se obter o tempo para estabilização do novo sistema, “T”, determinando-se na intersecção com a linha $c(t) = K$ (OGATA, 2010). O método da curva de reação por Ziegler-Nichols pode ser visto no Gráfico 3.

Gráfico 3 - Método de Curva de Reação por de Ziegler-Nichols



Fonte: Ogata (2010)

Com os valores de T e L obtidos, pelo método de Ziegler-Nichols é recomendado a sintonização dos parâmetros do controlador PID de acordo com as equações 18, 19 e 20 (OGATA, 2010).

$$K = 1,2 \frac{T}{L} \quad (18)$$

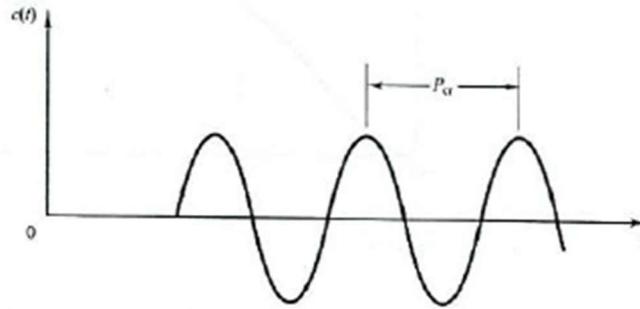
$$T_I = 2L \quad (19)$$

$$T_D = \frac{L}{2} \quad (20)$$

2.6.1.2 Método de sintonia em malha fechada

Aplicado em malha fechada, neste método define-se o T_i do controlador tendendo ao infinito e o T_d igual a zero, no intuito de fazer o sistema se comportar de maneira puramente proporcional. Após isso, o valor do termo proporcional K é constantemente aumentado até um ponto crítico K_{cr} , em que o sistema começa a apresentar oscilações constantes. A partir da frequência dessas oscilações, se obtêm o parâmetro P_{cr} , ou período crítico. Esses valores então são aplicados para determinar o valor das constantes do controlador (OGATA, 2010). O Gráfico 4 ilustra o comportamento de um sistema em malha fechada com oscilação constante.

Gráfico 4 - Esquema de oscilação para sintonia em malha fechada



Fonte: Ogata (2010)

De maneira análoga ao método em malha aberta e com os parâmetros P_{cr} e K_{cr} obtidos, pode-se obter os valores de sintonização dos controladores por meio das equações 20, 21 e 22 (OGATA, 2010).

$$K = 0,6K_{cr} \quad (21)$$

$$T_I = 0,5P_{cr} \quad (22)$$

$$T_d = 0,125P_{cr} \quad (23)$$

Este método apresenta uma resposta rápida do controlador em relação ao distúrbio, no entanto por ser considerada agressiva nesse quesito, tendo a presença de altas oscilações e *overshoots* em relação à mudança do *setpoint*, é interessante considerar a utilização de sintonias mais conservadoras (SEBORG *et al*, 2011).

Uma vez obtidos os valores das constantes do controlador, esses parâmetros podem ser utilizados na Equação 16 a fim de descrever o comportamento do controlador PID do sistema.

2.7 Implementação de controle automático de processos pela interação Arduino – Scilab

Durante o último século, o uso de computadores como auxiliar na indústria cresceu exponencialmente. Atualmente muitos sistemas, outrora manuais, realizam funções controladas automaticamente por algoritmos. A tendência é que nos próximos anos cada vez mais o trabalho braçal perca espaço para sistemas digitais automáticos (DELOITTE, 2015).

Com os avanços da eletrônica, cada vez se tornam mais disponíveis módulos prontos para uso em microcontroladores que necessitam apenas de um *software* para serem utilizados. O uso desses microcontroladores eletrônicos é uma opção

interessante na criação de sistemas de controle versáteis e de baixo custo, pois possuem capacidade de medições com precisão adequada e com grande variedade de aplicações. Entre elas, pode-se citar o uso dos sensores de pressão, temperatura, posição, dentre outros (CASTRO, 2020).

Ainda segundo Castro (2020), das plataformas de microcontroladores, o Arduino se destaca como sendo uma ótima opção, haja vista a facilidade de programação, baixo custo de aquisição e vasta gama de informações disponíveis online.

O Arduino trata-se de uma plataforma de código livre que permite direta interação de seu hardware e software com outros dispositivos com funções definidas, estendendo suas aplicações por meio de módulos. Tais módulos podem ampliar a utilização do Arduino a outras plataformas, aumentando ainda mais sua versatilidade (MCROBERTS, 2011). Dessa maneira, a plataforma pode atuar como uma interface física de entrada e saída de sinais, tendo seu processamento feito por softwares de computação numérica (CAMARGO *et al.*, 2015).

O Scilab é um software de uso gratuito e código aberto muito utilizado para fins acadêmicos ou industriais. O programa apresenta centenas de funções matemáticas e um alto nível de programação, possibilitando também seu uso para controle de processos (SCILAB, 2022a).

O software apresenta em sua biblioteca uma ferramenta para estudo e controle de sistemas dinâmicos chamada Xcos. Por meio dessa ferramenta, é possível simular um sistema por meio de diagramas de blocos presentes em sua paleta de funções, que tem a habilidade de adquirir e controlar dados obtidos por meio de comunicação direta com alguma porta serial. É possível fazer essa comunicação direta com o Arduino, ao compilar o módulo *toolbox Arduino* na placa do *hardware* e fazer a conexão via cabo USB (SCILAB, 2022b).

3 MATERIAIS E MÉTODOS

Neste capítulo, estão apresentadas as informações à respeito dos componentes utilizados no projeto e a metodologia aplicada na construção e controle do sistema de nível de tanques.

3.1 Materiais

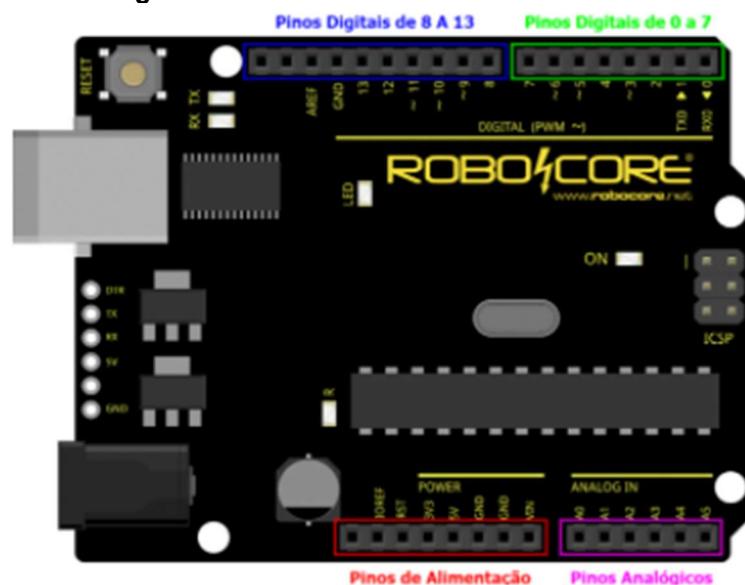
A seção a seguir se destina a explicar a respeito dos equipamentos utilizados no desenvolvimento deste projeto e do *software* de controle a ser utilizado.

3.1.1 Microcontrolador Arduino Uno

Muitos modelos de microcontroladores Arduino existem no mercado, cada um com características distintas para se adequar a vários níveis de projetos.

A placa escolhida para esse trabalho foi a BlackBoard Uno R3 da Robocore, mostrada pela Figura 5, que atende aos requisitos para o sistema em estudo. A placa é um modelo mais robusto comparado à placa Uno clássica e que permite acesso direto ao microcontrolador ATmega328, no qual caso haja algum defeito, a substituição do microcontrolador pode ser feita, sem necessariamente ter que descartar a placa, fornecendo um ótimo custo-benefício (ROBOCORE, 2022).

Figura 5 - Placa Arduino BlackBoard Uno R3



Fonte: Robocore (2022)

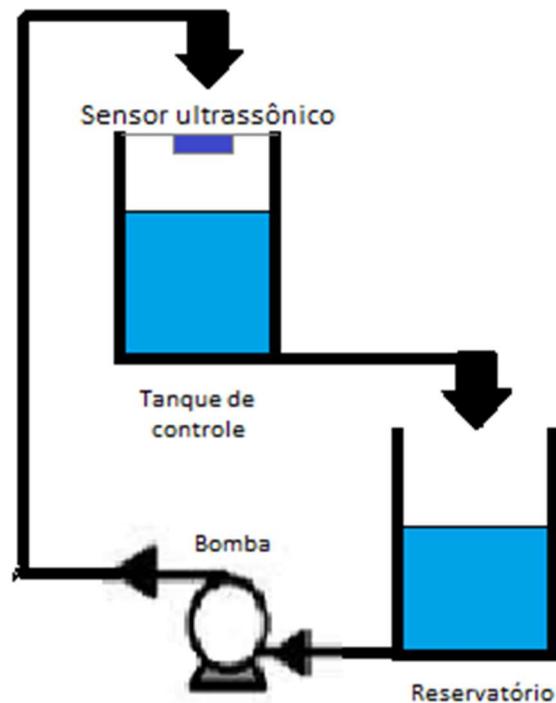
O Arduino UNO possui 14 pinos de entradas-saídas digitais, dos quais 6 deles podem ser usados como saídas PWM. Possui também 6 entradas de sinal analógico, uma entrada USB e um alimentador de bateria 5V (ARDUINO, 2022a).

3.1.2 Protótipo de tanque

Na construção do protótipo de controle de nível PID, foram utilizados dois tanques em acrílico transparente, um sendo o tanque de controle e outro servindo de reservatório de água. Os tanques possuem altura de 25 cm e área transversal de 175 cm².

A três centímetros do fundo do tanque foi instalado um tubo de plástico de 5 mm de diâmetro, a fim de se obter a saída do processo por gravidade. O tanque de controle, como mostrado pela Figura 6, é alimentado por uma bomba centrífuga e a leitura da altura do nível do tanque é medida pelo sensor ultrassônico posto no topo do tanque.

Figura 6 - Esquema do protótipo



Fonte: Autoria Própria (2022)

3.1.3 Bomba hidráulica

Para o abastecimento de água no tanque, foi escolhido a bomba de água modelo RS385. Esta bomba foi desenvolvida especialmente para projetos de prototipagem, operando com uma vazão máxima 2 litros por minuto.

A bomba RS 385, apresentada na Figura 7, possui um peso e tamanho reduzidos, podendo ser utilizada em ampla variedade de projetos. Ela opera com voltagem DC de 9 a 15 Volts e corrente máxima de 2 Amperes, podendo elevar água até 3 metros de altura (ARDUINOMEGA, 2022).

Figura 7 - Bomba RS385

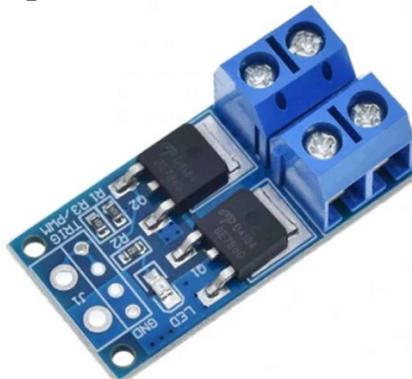


Fonte : Arduinomega (2022)

3.1.4 Driver PWM

Com o intuito de controlar a potência do motor da bomba, que opera por corrente direta, um módulo de pulso PWM é uma opção barata e funcional. Na Figura 8 é apresentado o modelo do Driver PWM utilizado no projeto.

Figura 8 - Módulo Driver PWM 5V



Fonte: Eletrônica (2022)

O driver é alimentado por uma fonte DC 12 Volts e recebe o sinal PWM vindo do controlador, convertendo em sinal de tensão analógica. Essa tensão fornece potência à bomba de acordo com o controle realizado pelo *software*.

3.1.5 Sensor ultrassônico

Para a aferição da altura do tanque, foi escolhido para este trabalho o sensor ultrassônico de distância HC-SR04, mostrado na Figura, sendo capaz de medir distâncias entre dois centímetros e quatro metros com precisão de até três milímetros.

Figura 9 - Sensor HC-SR04

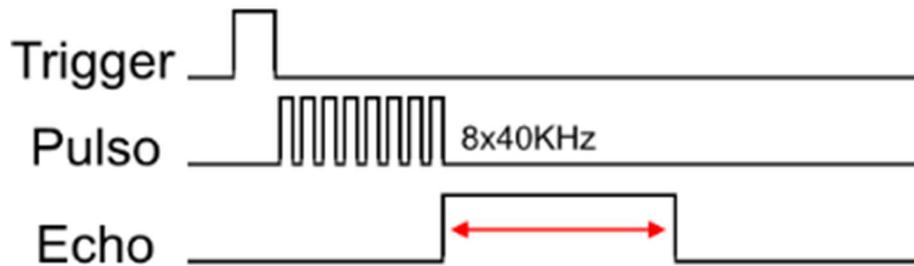


Fonte: Robocore (2022)

O sensor HC-SR04 baseia seu funcionamento na propriedade física da reflexão das ondas sonoras. Uma vez que as ondas sonoras apresentam velocidade constante e conhecida, é possível determinar a distância entre o sensor e o objeto pelo tempo de envio e recebimento do sinal.

O sensor ultrassônico utilizado apresenta quatro pinos, sendo eles um pino de alimentação de 5 volts *VCC*, *TRIG*, *GND* e *ECHO*. Inicialmente, um pulso é enviado para o pino *TRIG* que em seguida enviado oito pulsos de 40 kHz pelo emissor do sensor, para que em seguida a leitura seja feita pelo pino *ECHO*. Quando a onda é recebida, o tempo de ida e volta é determinado pelo algoritmo compilado na placa Arduino (ROBOCORE, 2022). O esquema de funcionamento do sensor ultrassônico pode ser visto na Figura 10.

Figura 10 - Esquema de funcionamento do sensor ultrassônico



Fonte: Adaptado de Thomsen (2021)

A conversão em distância pode ser realizada utilizando o tempo captado do sinal Echo e a velocidade do som pela Equação 24.

$$Distância = \frac{v_{som} * t_{resposta}}{2} \quad (24)$$

em que v_{som} representa a velocidade do som (340 m/s) e $t_{resposta}$ representa o tempo do sinal coletado pelo pino Echo.

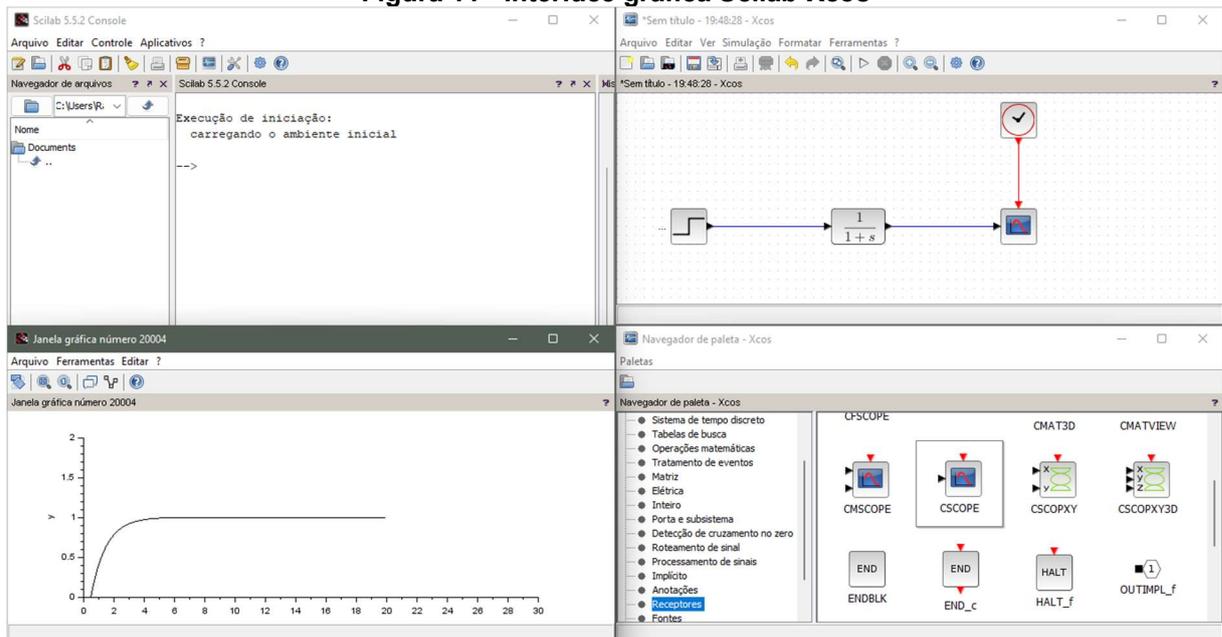
3.1.6 Simulação em Scilab-Xcos

Existem programas comerciais avançados para o desenho e implementação de sistemas de controle no disponíveis no mercado, no entanto, suas licenças de uso são, por vezes, custosas demais (SANJUAN, 2015).

Uma opção é utilizar o Scilab em conjunto com o Arduino, empregando um *software* gratuito e de código livre juntamente com um hardware popular e de baixo valor aquisitivo que sejam compatíveis com o projeto.

O Scilab possui diversas ferramentas para cálculos numéricos de maneira livre e aberta, criando um ambiente inclusivo para a engenharia e aplicações científicas. Uma dessas ferramentas é o Xcos (*Connected Object Simulator*), desenvolvida para modelagem e simulação de sistemas dinâmicos. Este fornece uma interface gráfica onde é possível, por meio de criação de um diagrama de blocos, editar modelos e aplicar funções pré-definidas simplesmente conectando os blocos entre si (SCILAB, 2022c). A Figura 11 apresenta algumas janelas da interface gráfica do Scilab-Xcos.

Figura 11 - Interface gráfica Scilab-Xcos



Fonte: Autoria Própria (2022)

3.2 Métodos

Nesta seção encontram-se as informações relevantes a programação e montagem dos componentes eletrônicos do projeto, bem como sobre a configuração e comunicação do sensor e do *hardware* com o *software* de controle.

Também estão presentes o método de modelagem do sistema utilizado para a definição dos parâmetros do controlador e uma breve explicação sobre a operação dos diagramas de blocos feitos no Xcos.

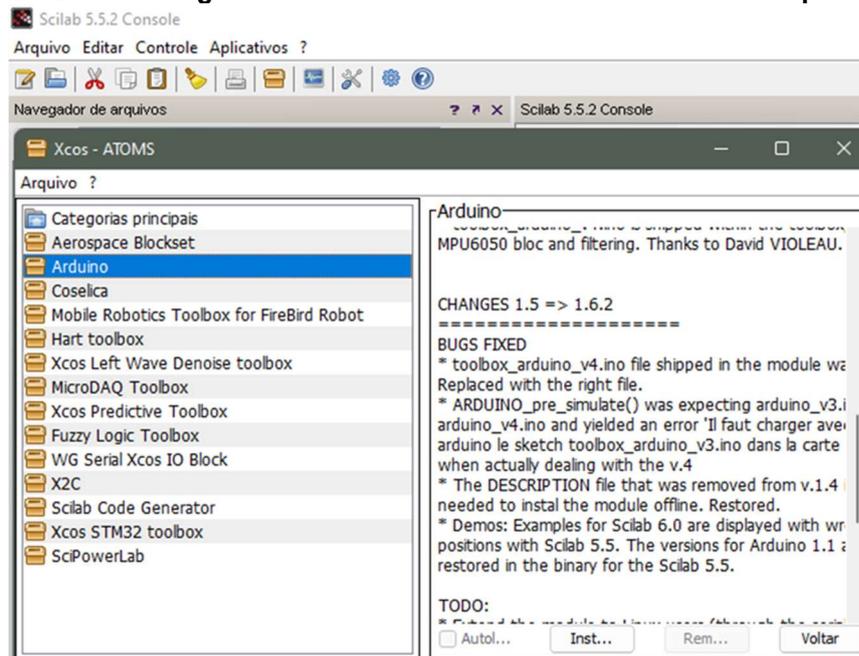
3.2.1 Comunicando o Arduino e componentes ao Scilab-Xcos

O primeiro passo para o desenvolvimento do projeto é estabelecer a comunicação entre a plataforma Arduino e o sensor ultrassônico com o Scilab.

Para isso, é necessária a utilização da *toolbox Arduino*, que é disponibilizada para download pelo site da biblioteca de módulos *Atoms* do *software* (ATOMS, 2022).

Uma vez baixado o módulo, este deve ser compilado na placa Arduino Uno a ser conectada ao *software* e então, instalar a ferramenta de comunicação serial Arduino pelo própria Scilab, disponível em *gerenciador de módulos - ATOMS – Xcos – Arduino*. Em detalhe na Figura 12, pode-se ver a janela do gerenciador *ATOMS* com a biblioteca Arduino a ser instalada.

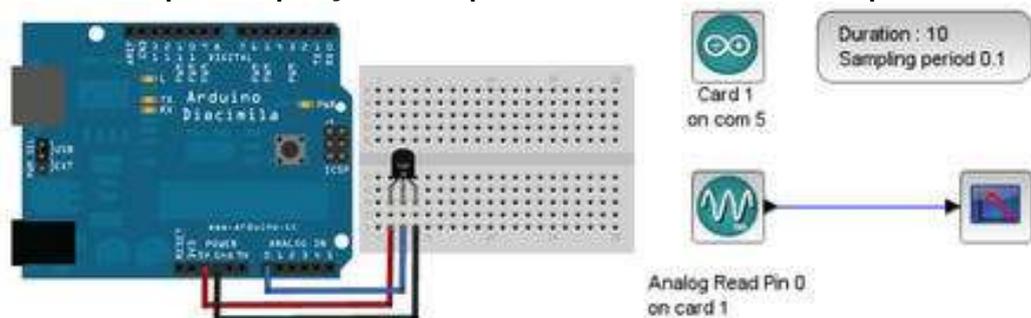
Figura 12 - Detalhe do gerenciador de módulos Atoms com biblioteca para Arduino



Fonte: Autoria Própria (2022)

A partir de então os blocos de comunicação do Arduino estarão inclusos na paleta de blocos do Xcos e a interação entre o *hardware* e o *software* pode ser realizada diretamente pelo Xcos, como mostrada pela Figura 13 (SCILAB, 2022c).

Figura 13 - Exemplo de aquisição de temperatura ambiente diretamente pelo Xcos



Fonte: Scilab (2022b)

3.2.1.1 Adquirindo dados da distância do sensor ultrassônico pelo Xcos

Como comentado na seção 3.1.5, o sensor ultrassônico opera com a captação de dois sinais. Um do pino Trigger, que envia o pulso ultrassônico e outro do pino Echo que recebe o som refletido. Com o tempo entre o envio do pulso e seu retorno, é possível calcular a distância.

Para converter o tempo em distância é necessário de um algoritmo compilado na própria IDE do Arduino, ou que o *software* utilizado possua um módulo próprio para o sensor, como encontrado no *Simulink*, uma extensão do aplicativo de controle pago *MatLab*, semelhante ao Scilab-Xcos (MATHWORKS, 2022).

Infelizmente, o Xcos ainda não possui um módulo de compatibilidade próprio para o sensor ultrassônico. Além disso, o programa apresentou problemas ao captar os sinais digitais do sensor ultrassônico, o que dificultou a aquisição dos dados diretamente pela interface do *software*.

A solução encontrada para este problema foi utilizar de uma segunda placa Arduino Uno no projeto. Inicialmente a distância do sensor ao objeto é obtida por meio de algoritmo compilado na primeira placa e em seguida, tem seu sinal digital transformado em analógico por meio de um circuito RC. A partir deste ponto, o sinal transformado pode ser enviado à segunda placa Arduino, que é configurada com a *toolbox Arduino*, podendo assim ter os dados acessados pelo Xcos.

3.2.1.1.1 Configurando a primeira placa Arduino

Primeiramente, o circuito conectando a placa Arduino ao sensor ultrassônico foi montado utilizando os pinos 5 e 6 para a entrada Echo e Trigger respectivamente e ligando o Arduino à porta USB “COM 3” do computador.

Em seguida, um algoritmo foi compilado pela IDE do próprio Arduino, disponibilizada para download no site da plataforma. Por conveniência, existem a disposição bibliotecas criadas pela empresa Arduino e por sua comunidade, que facilitam a programação e deixam o código do algoritmo mais simples e elegante (ARDUINO, 2022b).

Para a programação do algoritmo foi utilizada a biblioteca *ultrasonic.h*, que com poucas linhas de código é possível obter o resultado da distância do objeto ao sensor. Em seguida o algoritmo converte a distância em nível por meio da Equação 25:

$$\text{nível} = \text{distância} - \text{altura do tanque} \quad (25)$$

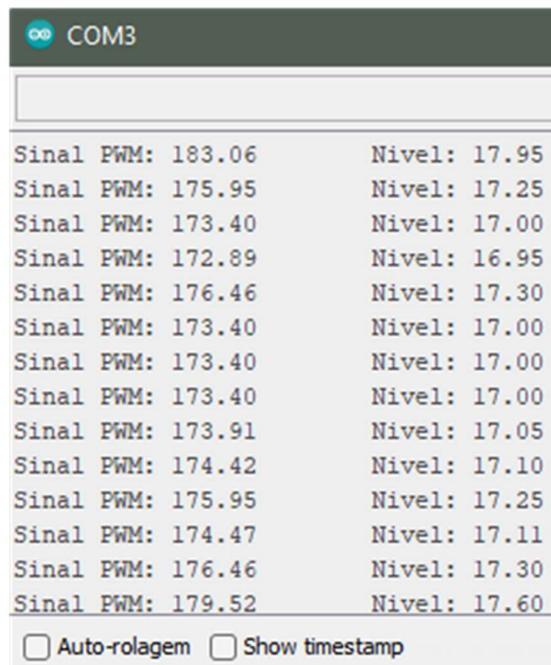
e o envia como sinal digital para o circuito RC através do pino 9 da placa Arduino.

O sinal digital enviado é limitado à altura máxima do tanque e proporcional ao nível, ou seja, o valor 0 corresponde ao tanque vazio e o valor de 255 corresponde ao tanque cheio.

Assim como no trabalho de Silva *et. al* (2018), durante os testes iniciais com as leituras do sensor ultrassônico foi percebido uma variação considerável nas leituras de nível mesmo com o tanque e o sensor estando estáveis, causada pelo ruído no sensor. Então no intuito de tentar reduzir essas variações na leitura, foi escolhido utilizar o sensor ultrassônico de maneira que a cada dois segundos, dez valores de nível são medidos e sua média calculada, para somente então ter o valor de nível enviado.

O algoritmo completo utilizado na primeira placa Arduino pode ser visto no Apêndice A. Na Figura 14, os valores de nível medidos pelo sensor ultrassônico posicionado no topo do tanque são mostrados pela IDE do Arduino, juntamente com o sinal digital enviado ao circuito RC.

Figura 14 - Monitor Serial da IDE Arduino com valores do nível e sinal PWM enviado ao circuito RC



Sinal PWM	Nivel
183.06	17.95
175.95	17.25
173.40	17.00
172.89	16.95
176.46	17.30
173.40	17.00
173.40	17.00
173.40	17.00
173.91	17.05
174.42	17.10
175.95	17.25
174.47	17.11
176.46	17.30
179.52	17.60

Auto-rolagem Show timestamp

Fonte: Autoria Própria (2022)

3.2.1.1.2 Circuito RC e configuração da segunda placa Arduino

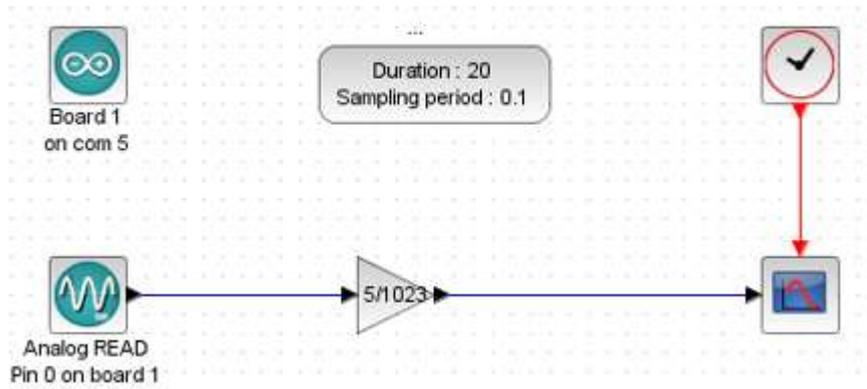
O sinal PWM saindo do pino 9 da primeira placa Arduino é conectado a entrada do circuito RC, que o transforma em sinal de tensão. O circuito é constituído de um resistor de 10 k Ω ligado em série à um capacitor de 10 μ F.

Ao conectar o circuito a entrada analógica A0 do segundo Arduino, podemos acessar pelo Xcos o sinal PWM convertido em sinal de tensão. A segunda placa

Arduino está configurada com o algoritmo da *toolbox_Arduino_v3*, baixado pelo site do gerenciador de módulos *Atoms* do Scilab. Nesse momento, consegue-se obter pelo Xcos o sinal do sensor ultrassônico na forma de tensão.

A segunda placa Arduino foi conectada à porta USB “COM 5”. Na Figura 15 pode ser visto um esquema montado no Xcos com os blocos para Arduino configurados para receber o sinal de tensão do sensor ultrassônico.

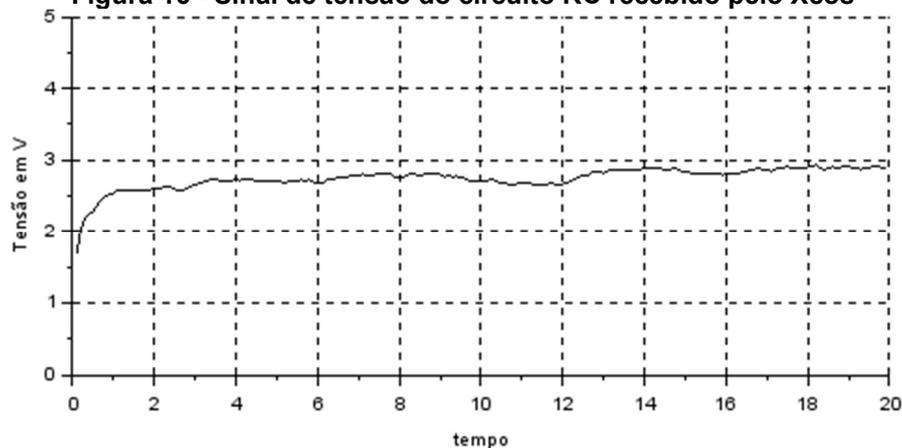
Figura 15 - Diagrama de blocos para adquirir o sinal de tensão do sensor ultrassônico



Fonte: Autoria Própria (2022)

O sinal analógico opera em uma faixa de 0 a 1023, sendo 0 é o valor para tensão mínima e 1023 para a tensão máxima. Como o circuito está alimentado pela placa Arduino, que trabalha com uma tensão de 5V, utiliza-se um bloco de ganho com a relação $5V/1023$ para obter a tensão recebida pelo pino A0, tendo esse sinal mostrado pela Figura 16. Esta tensão é proporcional ao nível do tanque medido pelo sensor ultrassônico, logo como isso pode-se facilmente convertê-la posteriormente para centímetros.

Figura 16 - Sinal de tensão do circuito RC recebido pelo Xcos

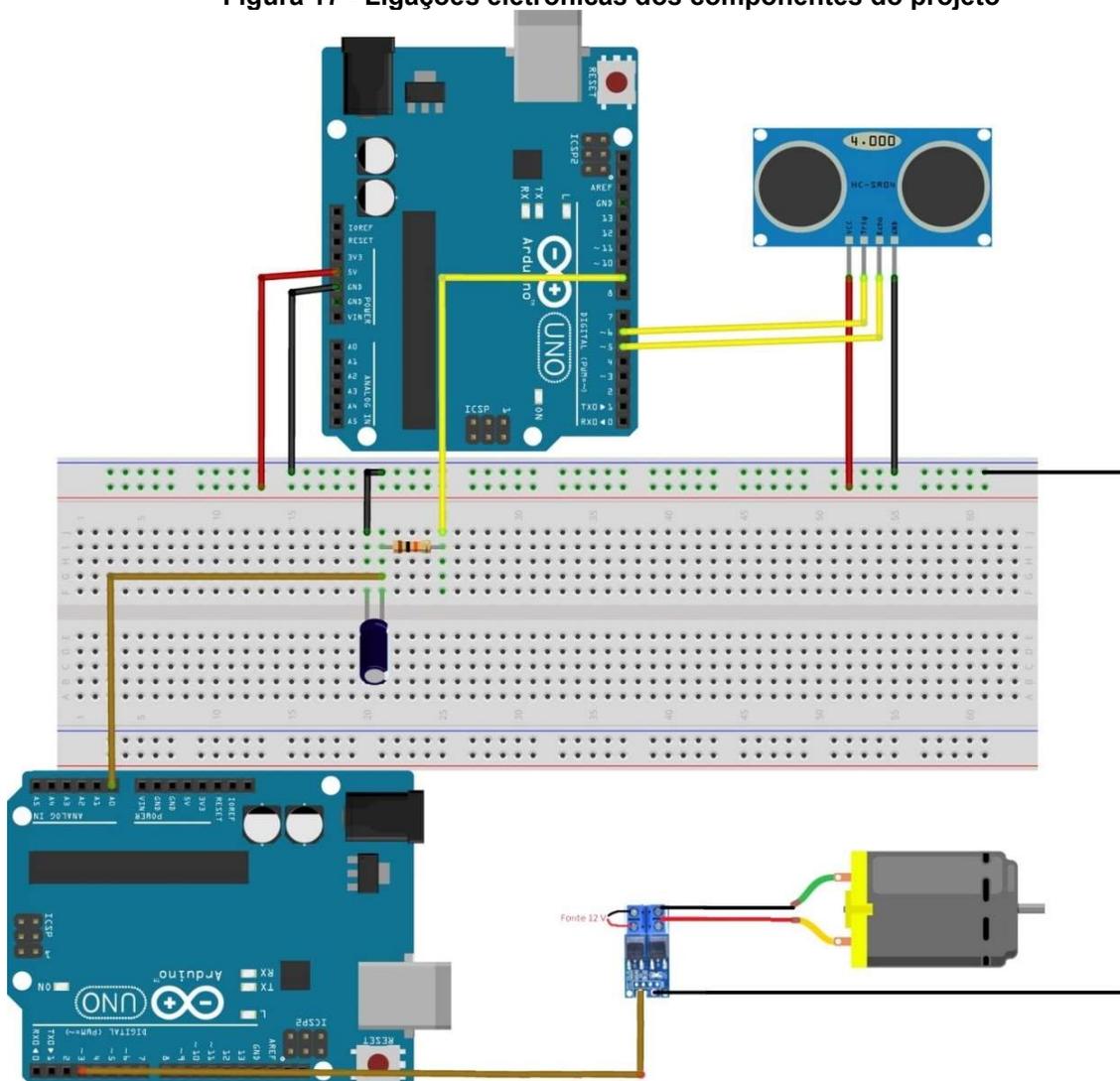


Fonte: Autoria Própria (2022)

3.2.1.1.3 Acionando a bomba pelo Xcos

A variável controlada do sistema é a vazão de entrada da bomba. Para controlar a bomba pelo Xcos, configura-se o pino 3 da segunda placa Arduino como uma saída utilizando o bloco *Analog WRITE*. Esse pino enviará o sinal PWM (de 0 a 255) para o Driver que controlará o motor da bomba. O driver é alimentado por uma fonte externa de 12V e modula a tensão enviada a bomba de acordo com o sinal PWM recebido. Na Figura 17 é mostrado um esquema completo das conexões feitas entre os componentes desse projeto.

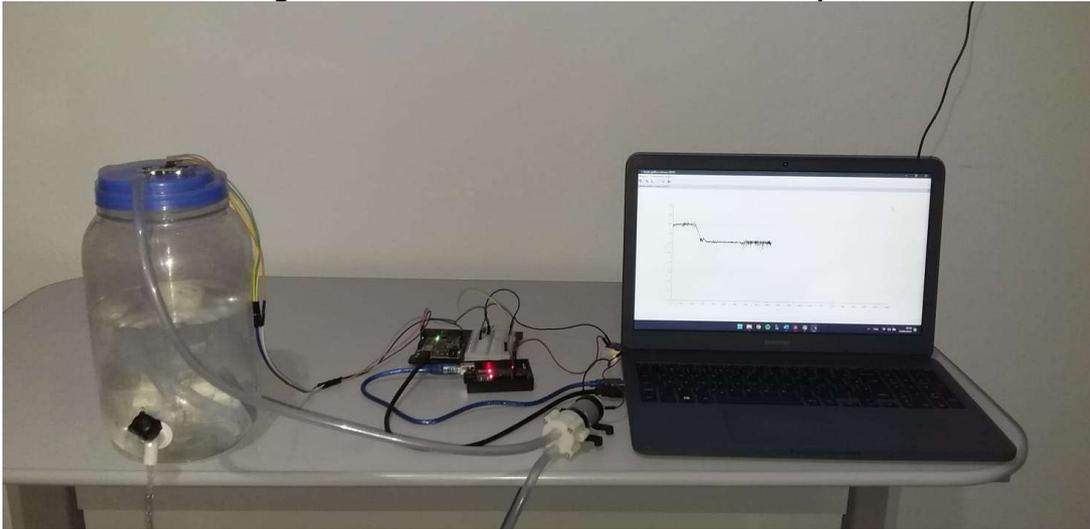
Figura 17 - Ligações eletrônicas dos componentes do projeto



Fonte: Autoria Própria (2022)

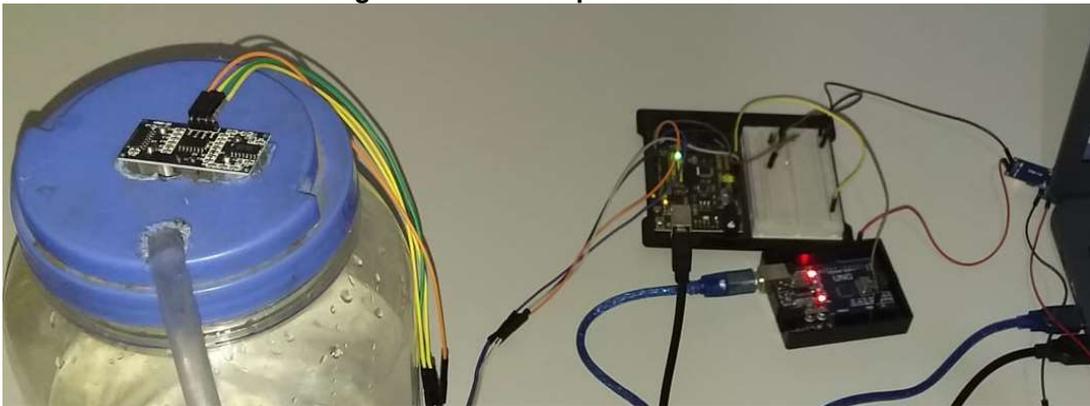
Nas fotografias 1, 2 e 3, é apresentado o sistema em operação mostrado acima.

Fotografia 1 - Sistema de controle de nível completo



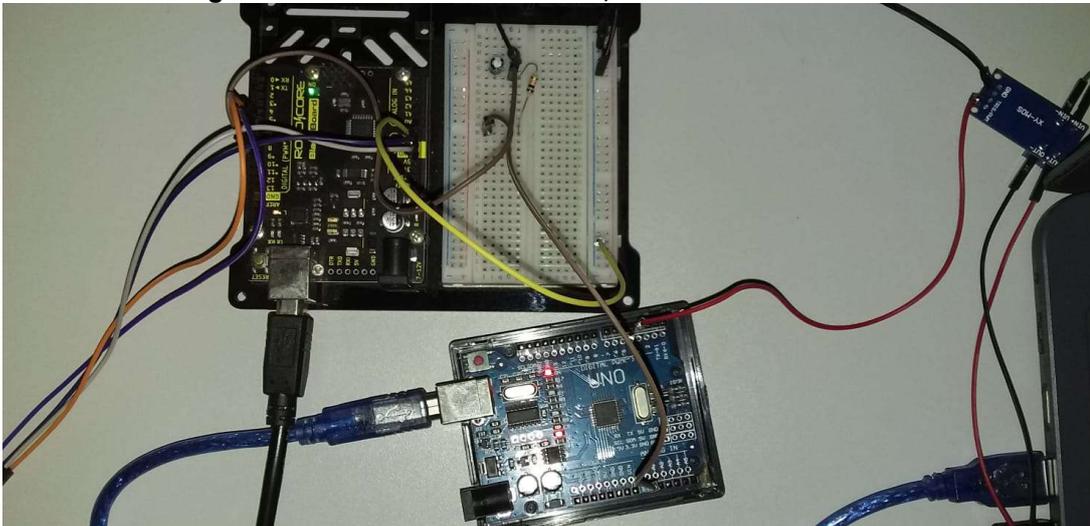
Fonte: Autoria Própria (2022)

Fotografia 2 - Vista superior do sistema



Fonte: Autoria Própria (2022)

Fotografia 3 - Detalhe dos Arduinos, Circuito RC e Driver PWM



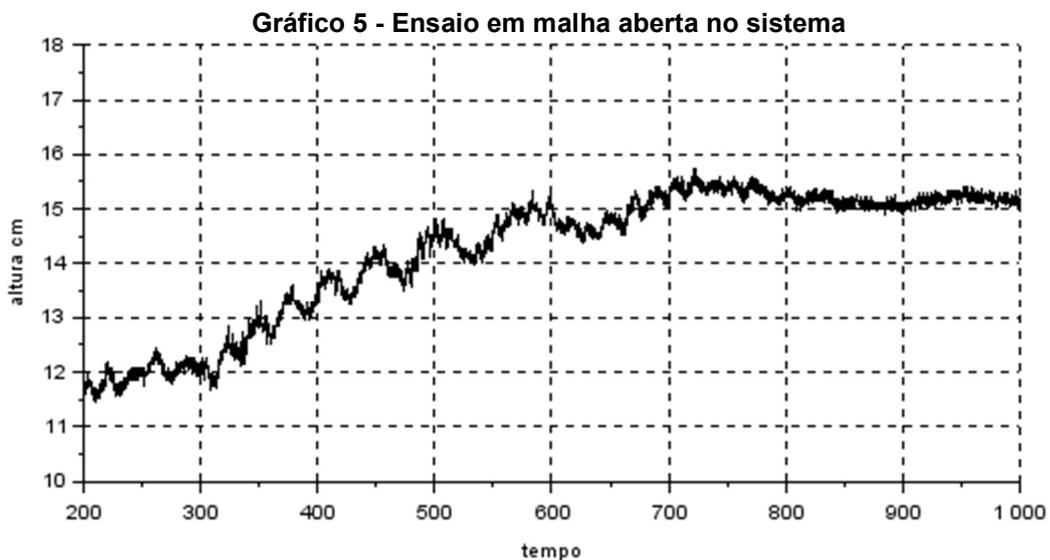
Fonte: Autoria Própria (2022)

3.2.2 Modelagem matemática do sistema

Com a comunicação entre o Arduino e o Xcos estabelecida, o próximo passo é conhecer o comportamento do sistema e configurar as constantes do controlador PID.

O sistema de tanques de armazenamento é um exemplo clássico de sistema de comportamento de primeira ordem. Existem vários métodos de modelagem do sistema, em que a maioria se baseia em deixar o sistema operando em malha aberta e acrescentar um distúrbio na entrada para se obter a “curva S” e a partir de então, aproximar o sistema a um de primeira ordem com tempo morto, podendo assim ter os parâmetros para a configuração do controlador PID definidos.

Testes foram realizados em malha aberta na tentativa de obter esses resultados, no entanto, devido ao sensor ainda apresentar medidas instáveis, os gráficos não se mostraram adequados para determinarmos os parâmetros com precisão como mostrado no Gráfico 5.



Fonte: Autoria Própria (2022)

Para esse ensaio, após o sistema entrar em estabilidade na altura de 12 centímetros com a bomba operando em 180 PWM, um distúrbio na vazão de entrada foi inserido no ponto de 300 segundos, elevando o sinal da bomba para 210 PWM.

As oscilações do sensor impossibilitam determinar ao certo o tempo de resposta do sistema e o novo ponto de estabilidade. Por este motivo, para este trabalho foi escolhido modelar matematicamente o sistema e realizar uma simulação

pelo Xcos com a função de transferência encontrada em malha fechada, para então as constantes do controlador serem determinadas.

Pela Equação 8 pode-se facilmente achar a função que representa a planta com os parâmetros de área e resistência. A área do tanque já é conhecida, como comentado na seção 3.1.2, restando apenas achar a resistência.

Ogata (2010) faz uma equiparação desse sistema à modelagem de um circuito RC, cuja função de sistema é dada por

$$G_s = \frac{R}{RCs + 1} \quad (26)$$

sendo a capacitância C é relativa à área transversal do tanque, pois uma vez maior a área, maior a capacidade do tanque em armazenar massa. Logo, a resistência pode ser equiparada a lei de Ohm em um fluxo laminar, em que

$$R = \frac{H}{Q} \quad (27)$$

já que a corrente é diretamente proporcional à diferença de potencial, tendo H como a altura em estado estacionário do sistema e Q a vazão de líquido.

3.2.2.1 Vazão da bomba em diferentes potências

Para se conhecer a resistência é preciso conhecer o perfil vazão da bomba. Com esse objetivo, realizou-se testes com a bomba conectada ao pino 3 do segundo Arduino. Os testes foram feitos com um volume fixo de 0,6 litros em diferentes valores de PWM configurados, tendo os resultados mostrados pela Tabela 1.

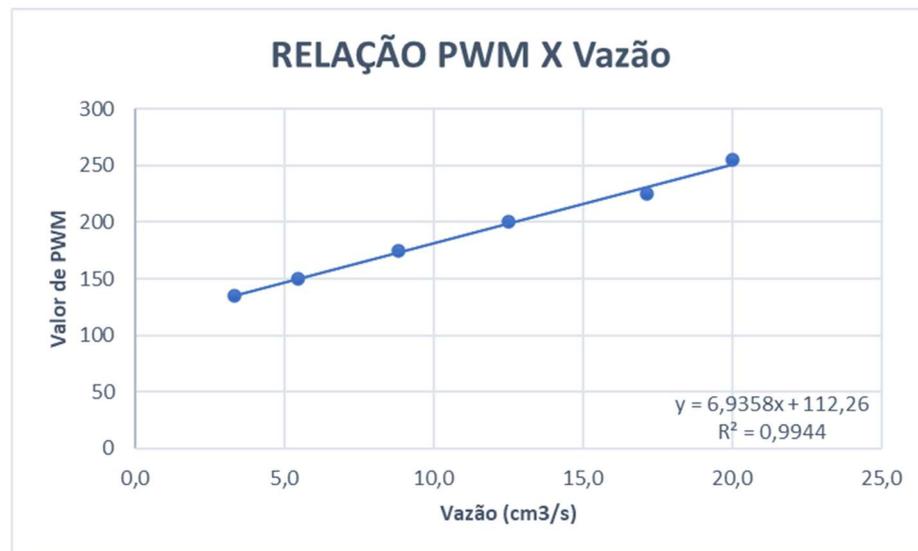
Tabela 1 - Resultados do teste de vazão da bomba versus sinal PWM

Tempo (s)	Vazão (cm³/s)	Valor PWM
30	20	255
35	17,1	225
48	12,5	200
68	8,8	175
110	5,5	150
180	3,3	135

Fonte: Autoria Própria (2022)

Durante os testes, percebeu-se que com um valor PWM abaixo de 130 não há potência suficiente para a água ser bombeada. Verificou-se também que a bomba em potência máxima atinge apenas a vazão de 1,2 litros por minuto, menos que o indicado pelo fabricante, indicando possível perda de carga no processo de bombeamento do reservatório até o tanque de controle. O Gráfico 6 mostra a relação dos resultados obtidos pelos testes da Tabela 1.

Gráfico 6 - Relação Sinal PWM versus Vazão



Fonte: Autoria Própria (2022).

Ao fim das análises, consegue-se perceber um comportamento linear da variação da vazão pela potência da bomba ($R^2 = 0,9944$), com obtendo a relação $PWM = 6,9358 * Q + 112,26$ que pode ser usada para converter o valor da variável controlada, calculada em cm^3/s , para o valor em PWM a ser enviado para a bomba.

3.2.2.2 Função de transferência do tanque

A partir do ponto em que se conhece o perfil de vazão da bomba, basta encontrar a altura em estado estacionário do sistema em determinada vazão de entrada.

Após certo tempo, com o sinal da bomba configurado em 200 PWM, ou seja, operando em vazão de $12,5 cm^3/s$, o sistema estabilizou-se em uma altura de 13,5 centímetros. Finalmente, pode-se encontrar o valor da resistência do tanque substituindo os valores na Equação 27.

$$R = \frac{13,5}{12,5} = 1,08 \quad (28)$$

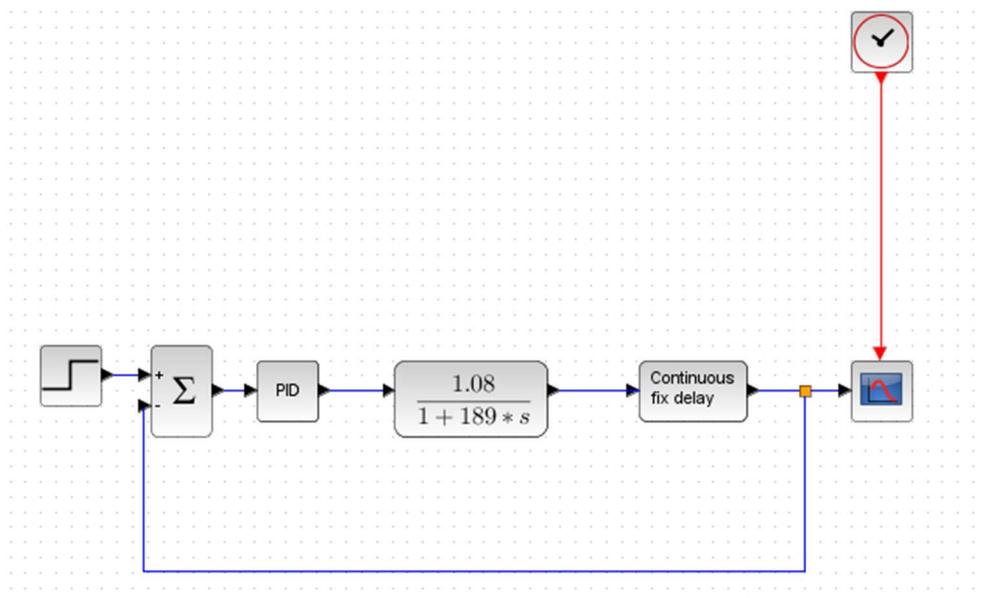
Conhecendo a área do tanque e aplicando na Equação 8, a função de transferência do sistema é dada por:

$$G(s) = \frac{1,08}{189s + 1} \quad (29)$$

3.2.3 Sintonia do controlador

Para realizar a sintonia do controlador utilizou-se uma simulação em malha fechada seguindo o método de Ziegler-Nichols, configurando um diagrama de blocos pelo Xcos, utilizando a função de transferência do sistema encontrada na seção 3.2.2.2. O Gráfico 7 mostra a simulação em malha fechada do sistema pelo Xcos.

Gráfico 7 - Simulação do sistema em malha fechada



Fonte: Autoria Própria (2022)

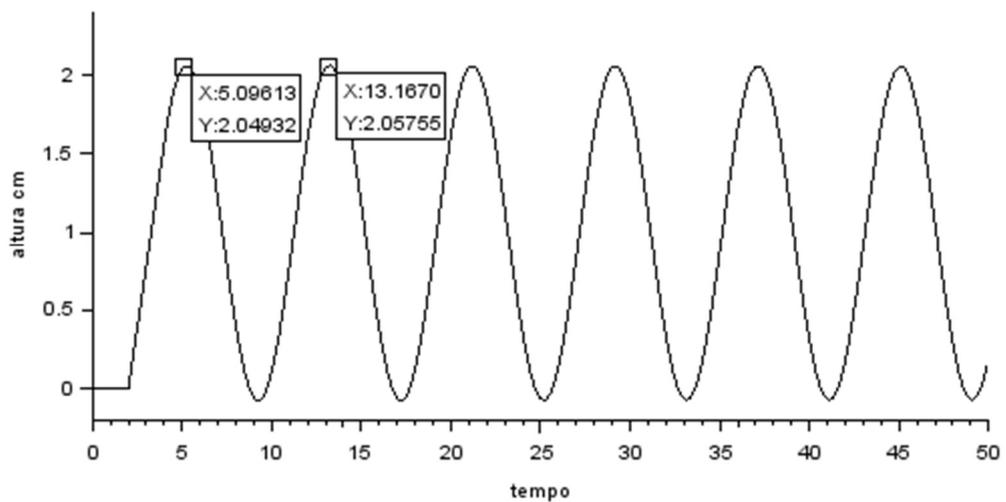
O primeiro bloco da esquerda para a direita do diagrama representa o desvio em degrau, onde no tempo 0 foi aplicado um distúrbio no valor de 1 centímetro. O bloco *Summation*, conectado em seguida, atua como comparador, aferindo erro ao sistema que em seguida é tratado pelo controlador PID.

O bloco *Continuos fix delay* aplica um atraso contínuo no sistema a fim de aproximá-lo de um sistema de primeira ordem com tempo morto (RIBEIRO; SANTOS,

2017). Para esta simulação, foi aplicado um atraso de 2 segundos, aproximando o tempo de resposta do sistema à leitura do sensor ultrassônico.

Uma vez aplicado um distúrbio degrau na entrada do processo, o controlador PID foi posto em modo de operação puramente proporcional e o ganho elevado gradativamente até atingir uma oscilação persistente. A oscilação sustentada foi encontrada em um K_{cr} de valor 138, com um P_{cr} de oito segundos, como mostrado pelo Gráfico 8.

Gráfico 8 - Oscilação sustentada



Fonte: Autoria Própria (2022)

Com K_{cr} e P_{cr} obtidos, calculou-se os valores do ganho, tempo integral e derivativo utilizando as equações 21, 22 e 23. Os parâmetros encontrados apresentam-se na Tabela 2.

Tabela 2 - Parâmetros do PID sintonizado utilizando o método de Ziegler-Nichols em malha fechada

$$K = 82,8$$

$$\tau_i = 4$$

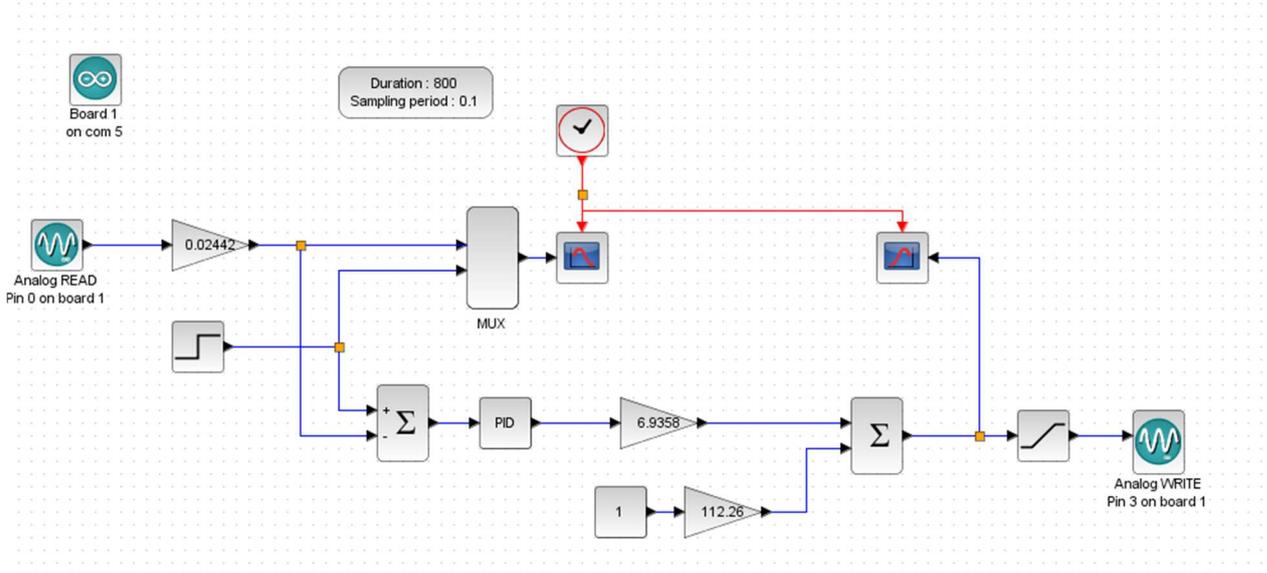
$$\tau_d = 1$$

Fonte: Autoria Própria (2022)

3.2.4 Montagem do diagrama de blocos do sistema

Ao passo em que as constantes do controlador PID foram definidas, resta configurar o diagrama de blocos do sistema *software* do Scilab-Xcos, como mostrado na Figura 18, e dar início aos testes do sistema com o controlador PID implementado.

Figura 18 - Diagrama de blocos do sistema com PID implementado



Fonte: Autoria Própria (2022)

O *setpoint* do sistema é determinado, no caso da Figura 18, pelo bloco *Step Function*, aplicando um distúrbio em degrau em um tempo pré-definido. O *setpoint* também pode ser configurado como uma constante ou com outro tipo de bloco do tipo fonte, como exemplo os blocos *Ramp* e *Pulse SC* que aplicam distúrbios tipo rampa e impulso no sistema, respectivamente.

O bloco *Analog READ*, configurado para o pino A0 do Arduino, recebe o sinal de tensão vindo do sensor, que em seguida é convertido em centímetros pelo bloco de ganho por meio da Equação 30

$$\text{nível}(cm) = \frac{\text{altura do tanque}}{\text{senal máximo analógico}} = \frac{25}{1023} \quad (30)$$

e comparado ao *setpoint* pelo bloco *Summation*, fornecendo assim o erro do processo. Esse erro é convertido em vazão pelo controlador PID configurado com os parâmetros da Tabela 2.

A partir do ponto em que o valor de vazão é calculado, este ser transformado em sinal PWM para controlar a bomba. Essa conversão é feita pelos ganhos aplicados

de acordo com a equação da reta obtida pelo Gráfico 6. Após esta conversão, o bloco *saturation* é utilizado na função de delimitar o sinal mínimo enviado à bomba em 0 e o máximo em 255.

Finalmente, o valor de PWM é enviado à bomba através do bloco *Analog WRITE*, que está conectada ao pino 3 do Arduino. Os valores do nível medido em tempo real e do *setpoint* são unidos utilizando o bloco *Mux* e mostrados em tempo real em um gráfico por meio do bloco *CScope*.

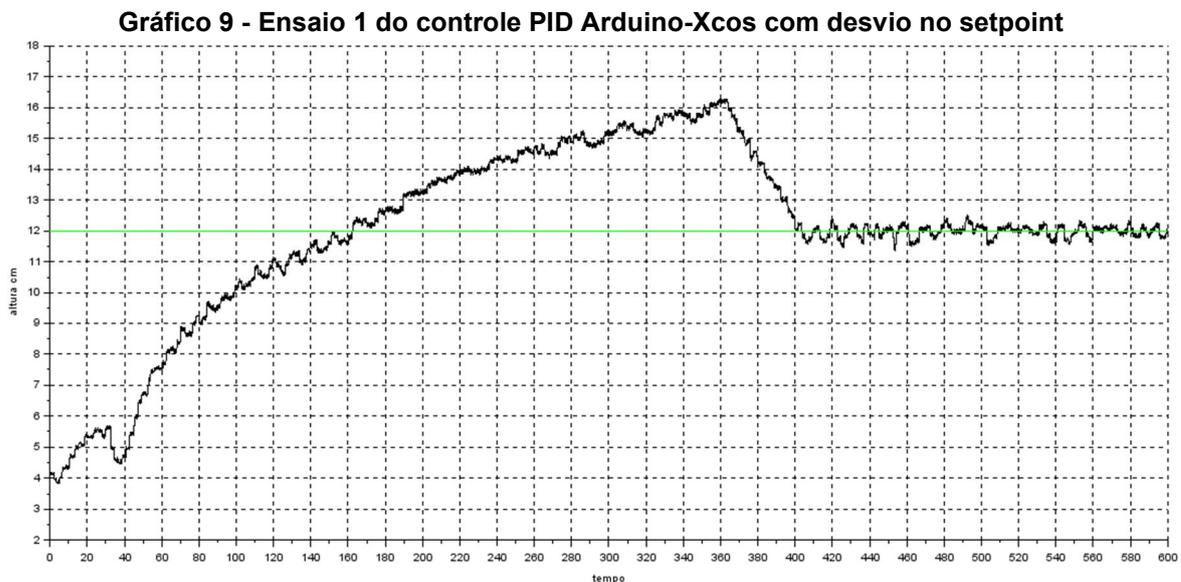
4 RESULTADOS

Neste capítulo serão apresentados os gráficos da resposta do processo frente aos diferentes distúrbios aplicados, bem como discussões à respeito do comportamento do sistema.

4.1 Ensaios com variação no setpoint

Para os ensaios iniciais realizados foram aplicados distúrbios em degrau no setpoint do sistema e analisada a sua resposta.

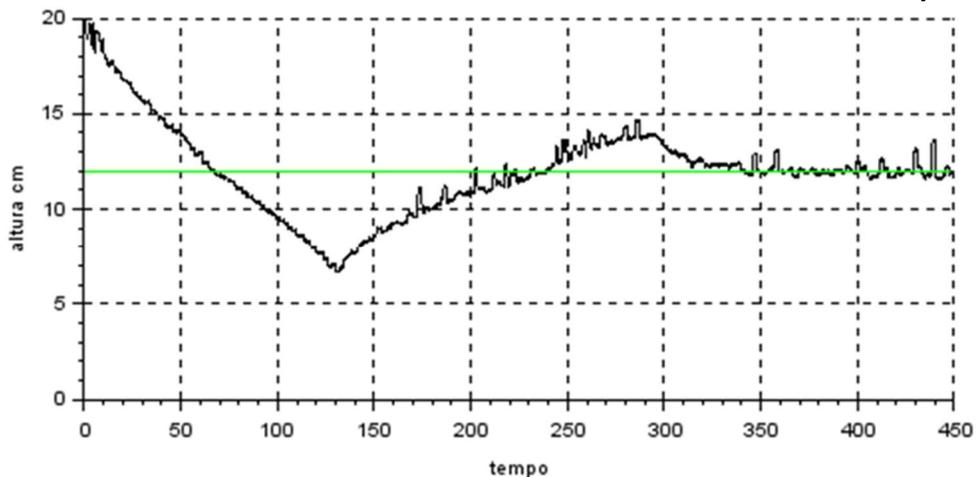
No primeiro ensaio realizado, o *setpoint* inicial foi definido em 12 centímetros de altura e partindo o sistema do nível do volume morto, (altura do tubo de saída, cerca de 4 centímetros). Nota-se no gráfico 9, um a formação de um *overshoot* de 50%, com o sistema se estabilizando após 400 segundos.



Fonte: Autoria Própria (2022)

No segundo ensaio, mostrado pelo Gráfico 2, o nível do tanque foi ajustado a uma altura de 20 centímetros, com o setpoint do sistema definido em 12 centímetros.

Gráfico 10 - Ensaio 2 do controle PID Arduino-Xcos com desvio no setpoint



Fonte: Autoria Própria (2022)

Percebe-se um *overshoot* mais agressivo, de cerca de 62,5% para o ensaio decrescente, ainda com formação posterior de *undershoot*. O sistema entra em estabilidade após 320 segundos.

No ensaio seguinte, visto no Gráfico 11, com o sistema em estado estacionário na altura de 12 centímetros, aplicou-se um distúrbio em degrau no ponto de 200 segundos, movendo o *setpoint* para 16 centímetros.

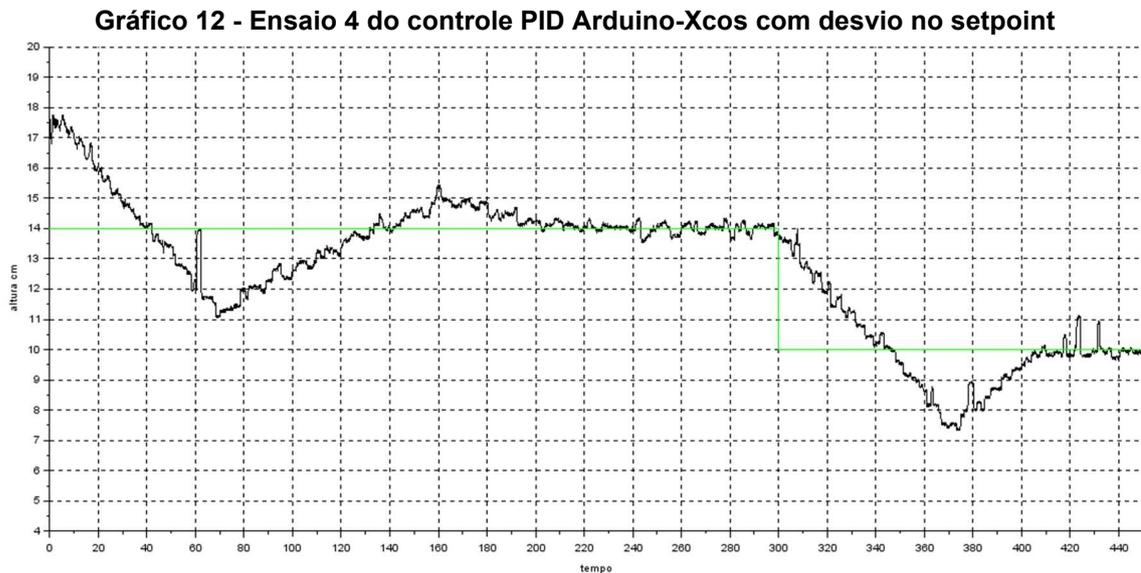
Gráfico 11 - Ensaio 3 do controle PID Arduino-Xcos com desvio no setpoint



Fonte: Autoria Própria (2022)

No terceiro ensaio o sistema se estabilizou após 260 segundos, apresentando um *overshoot* de 50% em relação ao distúrbio aplicado, mostrando um comportamento semelhante ao ensaio 1.

A seguir, o nível do tanque foi ajustado a uma altura de 18 centímetros e inicialmente definindo o *setpoint* para 14 centímetros. Em seguida, no tempo de 300 segundos, o *setpoint* foi baixado novamente para 10 centímetros. O resultado do ensaio é mostrado a seguir pelo Gráfico 12.



Fonte: Autoria Própria (2022)

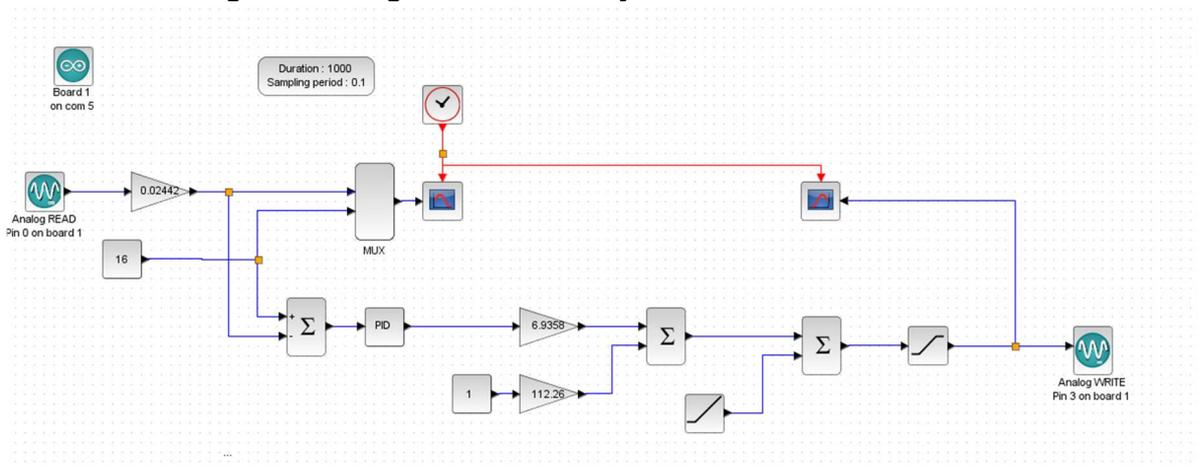
Após o fim do quarto ensaio, é visto que ocorre a presença de um *overshoot* de 75 %, com presença de *undershoot* de 25% para o primeiro distúrbio, atingindo a estabilidade após 200 segundos. Aplicado o segundo distúrbio, o sistema estabiliza-se após 110 segundos, com *overshoot* de 62,5%.

Ao fim dos testes iniciais com variação no *setpoint*, percebeu-se uma rápida resposta do controlador PID aos distúrbios. Os gráficos, no entanto, apontaram presença de *overshoots* consideravelmente grandes com algumas oscilações, o que caracteriza a configuração agressiva utilizada no controlador PID.

4.2 Ensaio com variação na entrada do sistema

Foram ainda aplicados distúrbios na entrada do sistema, configurando desvios do tipo degrau e rampa na entrada, de acordo com o diagrama de blocos ajustado da Figura 19.

Figura 19 - Diagrama de blocos ajustado com desvios na entrada

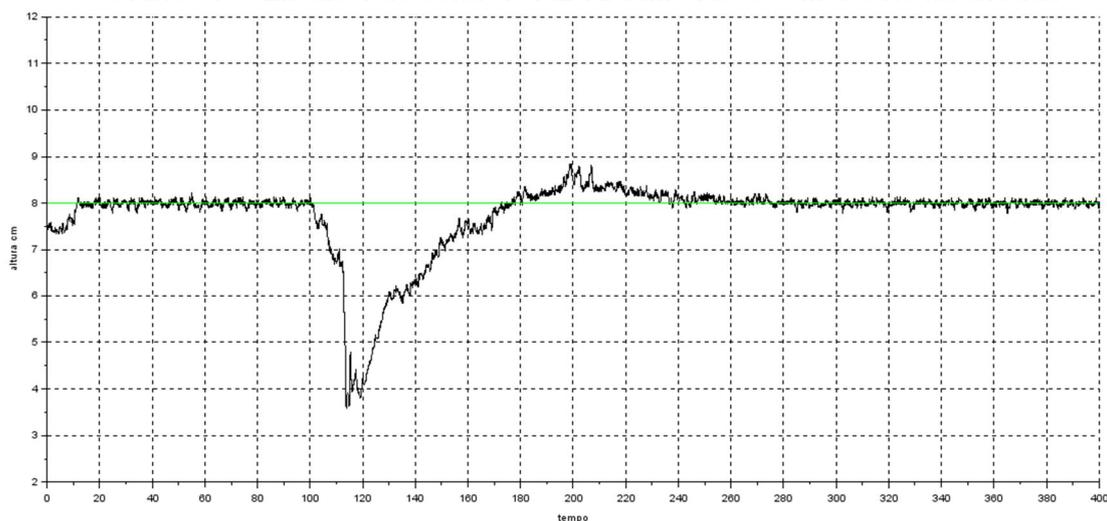


Fonte: Autoria Própria (2022)

Durante os ensaios desta seção, o *setpoint* foi fixado em um valor constante e após a saída do controlador PID, um bloco somatório foi inserido juntamente com um bloco de fonte, fornecendo um o desvio no sinal PWM da entrada do sistema.

No primeiro ensaio, após o sistema estar no estado estacionário a uma altura de 8 centímetros, foi utilizando um desvio em forma de degrau, onde o sinal da bomba recebeu um valor de -255, desligando-a completamente no tempo de 100 segundos. O resultado deste ensaio pode ser visto pelo Gráfico 13.

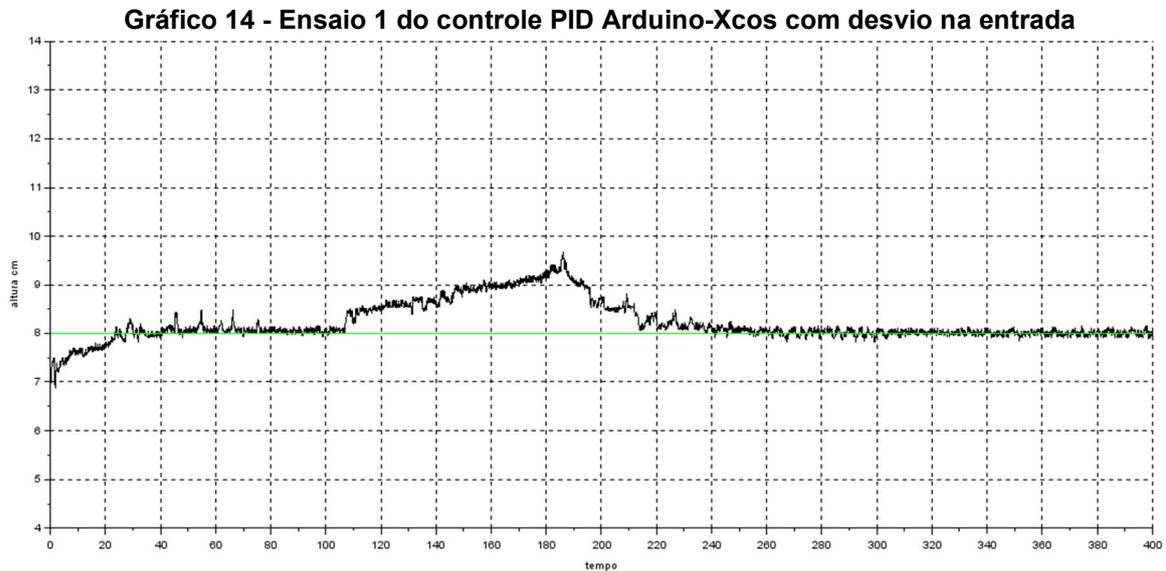
Gráfico 13 - Ensaio 1 do controle PID Arduino-Xcos com desvio na entrada



Fonte: Autoria Própria (2022)

Nota-se que após a que repentina no nível, o controlador consegue reajustar o nível e voltar o sistema ao *setpoint* após 140 segundos da aplicação do distúrbio.

O próximo ensaio, mostrado no Gráfico 14, utiliza as mesmas configurações do anterior, mudando somente o desvio em forma de degrau para 255, colocando a bomba em total potência no ponto de 100 segundos.



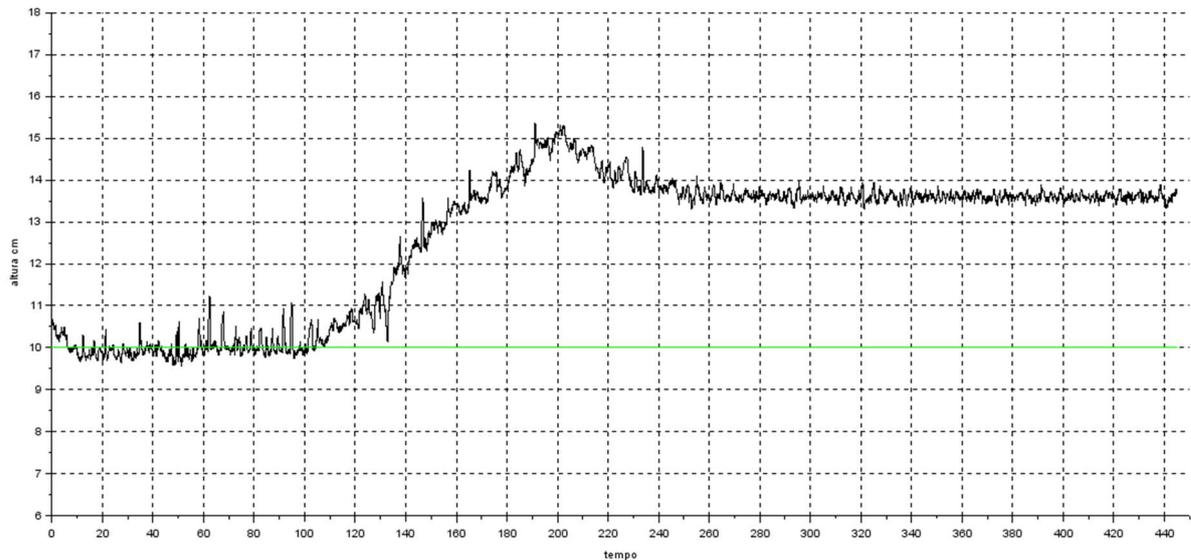
Fonte: Autoria Própria (2022)

De maneira análoga ao ensaio anterior, há um aumento súbito no nível do sistema, sendo logo regulado pelo controlador PID. O sistema retorna ao setpoint inicial após 120 segundos.

A seguir, o *setpoint* do sistema foi colocado em um valor fixo de 10 centímetros e o bloco *Step Function* substituído pelo bloco *Ramp*, a fim de promover um distúrbio de rampa unitária na entrada do sistema.

O bloco foi configurado com uma inclinação de valor 10 e aplicado no tempo de 100 segundos da simulação. Os dados obtidos deste ensaio pelo Xcos são mostrados no Gráfico 15.

Gráfico 15 - Ensaio 3 do controle PID Arduino-Xcos com desvio na entrada

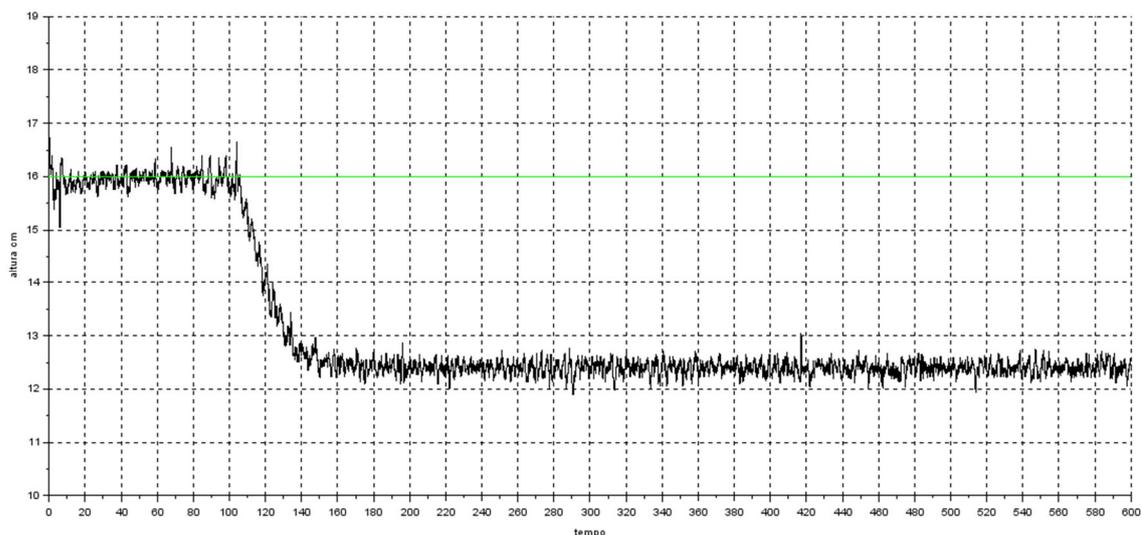


Fonte: Autoria Própria (2022)

Ao fim do ensaio, percebe-se que o sistema se estabiliza em um novo *setpoint* de 13,5 centímetros após 160 segundos de aplicado o distúrbio.

Por último, um ensaio em rampa unitária decrescente foi aplicado ao sistema com o *setpoint* definido em 16 centímetros. No estado estacionário, após 100 segundos, o desvio foi acionado com uma inclinação negativa de valor 10. O resultado do último ensaio é mostrado no Gráfico 16.

Gráfico 16 - Ensaio 4 do controle PID Arduino-Xcos com desvio na entrada



Fonte: Autoria Própria (2022)

Para o último ensaio, após 60 segundos o sistema entrada em estabilidade em um novo *setpoint* de valor de 12,5 centímetros.

Percebe-se até aqui que após aplicar um distúrbio em degrau, o controlador PID do sistema é capaz de zerar o erro causado pelo distúrbio e retornar o sistema ao *setpoint* inicial. No entanto, isso não ocorre se um desvio tipo rampa unitária é aplicado. O sistema se estabiliza, mas com um erro em regime permanente.

Segundo Ogata (2010), o mesmo erro permanente ocorreria se um distúrbio rampa fosse aplicado ao *setpoint* do sistema, uma vez que este se comporta como um de primeira ordem. Ao passo que o *setpoint* subiria seguindo a rampa unitária, o nível o acompanharia mantendo um erro finito. Em uma operação estacionária, a vazão de entrada é a mesma de saída, no entanto ocorre um erro de posição. Esse erro é proporcional a velocidade de entrada e inversamente proporcional ao ganho K do sistema.

Para evitar o transbordamento do tanque, foi optado por não realizar um ensaio com rampa no *setpoint* do sistema, uma vez que o nível iria subir acompanhando o *setpoint* até a borda onde o sensor está alocado. No entanto, ainda é possível confirmar o comportamento do sistema de primeira ordem em relação ao distúrbio de rampa unitária pelos Gráficos 15 e 16, em que após certo tempo de aplicado o distúrbio, o sistema entra em estado estacionário com um erro de posição permanente em relação ao *setpoint*.

5 CONCLUSÃO

A extensão Xcos se mostrou eficiente no estudo e simulação de sistemas dinâmicos de controle, apresentando ferramentas para a construção de malhas que tornam a aplicação dos conceitos de controle de processo mais prática e palpável.

Apesar da interação da plataforma Arduino com o Xcos ser possível por meio da *Arduino toolbox*, o *software* apresentou limitações para a captação de sinais digitais como os do sensor ultrassônico, fazendo-se necessário de um conversor de sinal digital-analógico, como o circuito RC utilizado no projeto, e da utilização de uma segunda placa Arduino Uno, dificultando a logística do sistema.

Modelando o sistema como um de primeira ordem com tempo morto e simulando seu comportamento pela interface do Xcos-Scilab, pode-se obter as constantes para a sintonia do controlador PID de maneira eficiente. Ao aproximar o modelo matemático do sistema real, é possível se obter os parâmetros necessários para o controle de maneira direta, sem necessitar de utilizar de tentativa e erro para a sintonia.

Durante a análise, nota-se uma amplitude considerável de ruído no sensor, mesmo com a tentativa de reduzi-lo ao aplicar a média calculada do sinal. Este ruído pode vir a afetar negativamente no desempenho do controlador, ocorrendo, provavelmente, devido à vibração causada pela bomba no tanque, assim como possíveis interferências captadas pelo sensor ou pelo circuito eletrônico.

Apesar da característica agressiva do método de sintonia do controlador PID utilizada, foi possível implementar um controle de nível eficiente, muito próximo ao valor de ponto fixo estabelecido. O controlador reagiu rapidamente aos distúrbios no setpoint, apesar de mostrar *overshoots* consideráveis, mesmo que ainda dentro dos limites de operação do sistema.

Para trabalhos futuros, recomenda-se a utilização de sensores diferentes para a medição do sinal de nível, bem como a utilização de técnicas mais conservadoras de sintonia dos controladores PID.

Por fim, os resultados desse estudo se mostram satisfatórios de acordo com o objetivo deste trabalho, implementar um controle PID em um sistema de tanques utilizando a plataforma Arduino em conjunto com a interface de controle Xcos do *software* Scilab.

REFERÊNCIAS

- ARDUINO, **Arduino UNO REV3**. Disponível em: <<https://store.arduino.cc>>. Acesso em 7 de abril de 2022a.
- ARDUINO. **Arduino IDE**. Disponível em: <https://www.arduino.cc/en/software>. Acesso em: 05 jun. 2022b.
- ARDUINOMEGA. **MINIBOMBA DE ÁGUA - RS-385**. 2022. Disponível em: <https://www.arduinoomega.com/mini-bomba-de-agua-rs-385-alto-fluxo>. Acesso em: 22 maio 2022.
- ÂSTRÖM, K. J.; HÄGGLUND, T. **PID Controlers: Theory, Design and Tuning**. EUA: Instrument Society of America, 1995.
- ATOMS. **Arduino Communication through Serial**. Disponível em: <https://atoms.scilab.org/toolboxes/arduino>. Acesso em: 5 jun. 2022.
- BARRETO, L.; ESCOBAR, A. **Aspectos prácticos de implementación em compensadores PID para control de posición de un automóvil**. Ingenium. v. 15, n. 29, p. 148-162, 2014.
- CAMARGO, Tiago Francisconi Borges et al. **Use of Scilab and Arduino for data acquisition environmental**. In: IEEE International Conference on Eletronic Measurement & Instruments, 12., 2015, Qindao: IEEE, 2015. p. 417 - 421.
- CASTRO, Luis Henrique Monteiro de. **O uso do Arduino e a criação de objetos educacionais em tempos e espaços desarticulados**. Revista de Ciência da Computação, Rio de Janeiro Rj, v. 2, n. 1, p. 2596-2701, jan. 2020.
- DELOITTE. **Industry 4.0: challenges and solutions for the digital transformation and use of exponential technologies**. Zurique, Suiça, 2015.
- ELETRÔNICA, Baú da. **Módulo Driver PWM**. Disponível em: <https://www.baudaeletronica.com.br/>. Acesso em: 02 jun. 2022.
- FELDER, Richard M. **Engineering Instructional Development: programs, best practices, and recommendations**. Journal Of Engineering Education. Raleigh, p. 89-122. jan. 2011.
- GONÇALVES, Marcelo Giglio. **Monitoramento e Controle de Processos**. Brasília: Senai, 2003. 100 p. 99 f
- MATHWORKS. **Ultrasonic Sensor**. Disponível em: <https://www.mathworks.com/help/supportpkg/arduino/ref/ultrasonicsensor.html>. Acesso em: 05 jun. 2022.
- MCROBERTS, M. **Arduíno Básico**. Novatec Editora. São Paulo: Novatec, 2011.

OGATA, K. **Engenharia de controle moderno**. 5.ed. São Paulo: Pearson Prentice Hall, 2010.

OLIVEIRA, A. L. LIMA, **Instrumentação – Fundamentos de Controle de Processo**. SENAI - Serviço Nacional de Aprendizagem Industrial. Espírito Santo. 1999.

PANUCI, V. C. **Análise teórica, numérica e experimental de um coletor solar de placas planas destinado ao aquecimento de ar**. Dissertação (mestrado) – Universidade Estadual de Maringá, Centro de Ciências Exatas, Departamento de Engenharia Mecânica, Programa de Pós-Graduação em Engenharia Mecânica, 2017.

RIBEIRO, A. M.; SANTOS, R. B. **Sintonia de um controlador PID em um sistema de controle de nível de tanques em série utilizando um software gratuito**. XII Congresso Brasileiro de Engenharia Química em Iniciação Científica. São Carlos, 16 jul. 2017.

ROBOCORE. **Medindo distância**. Disponível em: <https://www.robocore.net/tutoriais/kit-iniciante-v8-medindo-distancia>. Acesso em: 02 jun. 2022.

SANJUAN, J. Vásquez. (2015). **Scilab-Arduino: Diseño e implementación de sistemas de control**.

SCILAB, **What is Scilab?** .Disponível em: <<http://www.scilab.org/about>>. Acesso em 22 de maio de 2022a.

SCILAB, **Description**. Disponível em: <<http://atoms.scilab.org/toolboxes/arduino/1.6.2>>. Acesso em 22 de maio 2022b.

SCILAB. **Xcos**. Disponível em: <https://www.scilab.org/software/xcos>. Acesso em: 29 maio 2022c.

SEBORG, D. E.; MELLICHAMP, D. A.; EDGAR, T. F.; III, F. J. D. **Process dynamics and control**. 3 ed., Nova Iorque: John Wiley & Sons, 2010.

SILVA, Cássio Rodolfo Aveiro da *et al.* **Controle de nível em tanque através da mescla entre Scilab e Arduino baseado no controle PI**. In: III Simpósio Paranaense De Modelagem, Simulação E Controle De Processos, 2018, Maringá p. 92-99.

SMITH, C. A.; CORRIPIO, A. **Princípios e Prática do Controle Automático do Processo**. 3 ed., Rio de Janeiro: LTC, 2008.

STEPHANOPOULOS, George. **Chemical Process Control: An Introduction to Theory and Practice**. 1st ed. New Jersey: Prentice-Hall International Inc, 1984. 696p.

THOMSEN, Adilson. **Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino**. 2011. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>. Acesso em: 22 maio 2022.

APÊNDICE A – CÓDIGO SENSOR ULTRASSÔNICO

```
//Código implementado na primeira placa, conectada a porta USB COM 3
// com sinal PWM enviado ao circuito RC
```

```
#include <Ultrasonic.h> //Biblioteca do senso ultrassônico
Ultrasonic ultrasonic(5, 6); // definição dos pinos Echo e Trigger
float Dist, nivel, sin1,soma=0,i=0 // definição das variáveis
```

```
void setup () {
  Serial.begin(9600); //Inicializa a serial
  pinMode (9, OUTPUT); //Saída Xcos
}
```

```
void loop (){
```

```
  for(i=0;i<10;i++){ // laço for para aquisição de várias medidas de distância
    Dist = ultrasonic.read(); // recebe o valor em distância já convertido
    delay (200);
    soma += Dist;
  }
```

```
  Dist = soma/i; //variável "Dist" recebe a média dos valores do laço for
  i=0;
  soma = 0;
```

```
  nivel = 25-Dist; // Conversão do sentido dos dados da distância
```

```
  sin1 = nivel*10.2; // Conversão do Nível em PWM: nivel*255/25
  digitalWrite (9, sin1); // sinal do pino ECHO enviado para o circuito RC
  Serial.print("Sinal PWM: ");
  Serial.print(sin1);
  Serial.print("\t Nivel: ");
  Serial.println(nivel);
  Serial.println(nivel);
}
```