INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

***CAMPUS* SÃO PAULO**

DANILO OLIVEIRA MARTINS

DESENVOLVIMENTO DE FERRAMENTA PARA CONFIGURAÇÃO DE SISTEMAS DE CONTROLE EM DISPOSITIVOS EMBARCADOS (IoTControl)

Dissertação apresentada ao Programa de Pós-Graduação Stricto Sensu do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo como parte dos requisitos para obtenção do título de Mestre em Automação e Controle de Processos.

Orientador:
Prof. Dr. Alexandre Brincalepe Campo
Coorientador:
Prof. Dr. Almir Fernandes

São Paulo

2017

Catalogação na fonte
Biblioteca Francisco Montojos - IFSP Campus São Paulo
Dados fornecidos pelo(a) autor(a)

M379d Martins, Danilo Oliveira
Desenvolvimento de ferramenta para configuração de sistemas de controle em dispositivos embarcados (iotcontrol) / Danilo Oliveira Martins. São Paulo: [s.n.], 2017.
124 f. il.

Orientador: Alexandre Brincalepe Campo
Co-orientador: Almir Fernandes

Dissertação (Mestrado Profissional em Automação e Controle de Processos) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2017.

1. Controle. 2. Internet das Coisas. 3. Controle Embarcado. 4. Ferramenta Gráfica de Controle. I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo II. Título.

CDD 629.8

MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO PAULO
DIRETORIA GERAL DO *CAMPUS* SÃO PAULO
Coordenadoria de Registros Escolares de Pós-Graduação

ATA DE EXAME DE DEFESA DE DISSERTAÇÃO

Nome do Programa: **Mestrado Profissional em Automação e Controle de Processos**

Nome do(a) Aluno(a): Danilo Oliveira Martins

Nome do(a) Orientador(a): Prof. Dr. Alexandre Brincalepe Campo

Nome do(a) Coorientador(a): Prof. Almir Fernandes

Título do Trabalho: "Desenvolvimento de ferramenta para configuração de sistemas de controle em dispositivos embarcados (IoTControl)"

Abaixo o resultado de cada participante da Banca Examinadora

Nome completo dos Participantes Titulares da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Alexandre Brincalepe Campo – Orientador	IFSP – SPO	APROVADO
Prof. Dr. Tarcisio Fernandes Leão – Membro Interno	IFSP – SPO	APROVADO
Prof. Dr. Kechi Hirama - Membro Externo	USP	Aprovado
Nome completo dos Participantes Suplentes da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Eduardo Alves da Costa – Membro Interno	IFSP – SPO	
Prof. Dr. Edson Caoru Kitani – Membro Externo	FATEC	

Considerando-o: [X] APROVADO
[ ] NÃO APROVADO

Assinaturas

São Paulo, 27 de setembro de 2017

Presidente da Banca

Membro Interno

Membro Externo

Observações:

Dedico esta dissertação aos meus pais
Lenira e João, e a minha esposa Carol
Pela paciência, apoio, incentivo e
Compreensão

AGRADECIMENTOS

A Deus.

Aos professores do Mestrado em Automação e Controle de Processos do IFSP. Pela dedicação e responsabilidade aplicada no ensino de alto valor que viabilizou esta oportunidade especial na minha jornada em busca do conhecimento.

Ao professor Alexandre Brincalepe Campo pelo bom exemplo e contribuição como orientador e ao professor Almir Fernandes pelos ensinamentos e compartilhamento de informações essenciais para o desenvolvimento deste projeto.

À minha esposa Ana Carolina pela compreensão e paciência durante esta jornada de meus estudos.

Ao Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), que proporcionou a execução deste trabalho.

Muito obrigado.

“One machine can do the work of fifty ordinary men.
No machine can do the work of one extraordinary man.”

Elbert Hubbard

RESUMO

Este trabalho descreve o desenvolvimento de uma ferramenta para sistemas de controle e supervisão de dados em dispositivos embarcados. Esta ferramenta é capaz de configurar parâmetros de controle dinamicamente, dispensando interromper o processo, possibilita acesso e operação remota. O foco deste trabalho consiste no desenvolvimento do programa aplicativo computacional de controle e a interface de rede que permite a interação homem máquina, sendo que com esta interface é possível realizar o controle e supervisão do processo, possibilitando ao usuário supervisionar e selecionar algoritmos. O programa computacional aplicativo está embarcado no dispositivo físico conectado ao processo a ser controlado. Este aplicativo apresenta, em sua tela inicial, o modelo de controlador selecionável pelo usuário, permite a mudança em tempo de execução da estrutura do controlador, assim como dos parâmetros que o definem. Os modelos sugeridos pelo aplicativo devem ser selecionados pelo usuário e configurados para o objeto de controle desejado. Desta forma caberá ao usuário estabelecer os parâmetros iniciais, bem como qual modelo melhor se aplica ao objeto de controle. A configuração e definição dos parâmetros de entrada além de possíveis distúrbios a serem simulados no sistema devem ser definidos a partir da seleção inicial do modelo. O programa permite o teste de diversas estruturas de controle através do uso de uma interface gráfica intuitiva e permite ainda a coleta de dados do sistema que está sendo testado para que sejam aplicadas técnicas de ajuste de parâmetros de controle, sintonizando o controlador. O sistema proposto foi verificado em uma planta de controle experimental, composta de um motor elétrico com encoder acoplado ao dispositivo computacional por uma interface eletrônica, sendo que suas funcionalidades foram testadas e avaliadas através de métodos gráficos. Para interação com o sistema foi utilizado um aparelho de telefone celular com acesso a internet, utilizado como interface visual.

Palavras-chave: Controle, ferramenta gráfica de controle, controle embarcado, internet das coisas.

ABSTRACT

This paper describes the development of a tool for control and supervision systems on embedded devices. This tool is capable of configure control parameters without stopping the process allowing remote access and operation. The main subject of the work is the development of application software and network program with human machine interface. Thru this interface is possible to perform the control and supervision of the process, permit the user to monitor and select algorithms. The application software, embedded into the physical device that connect to the control plant, uses the network program to present into initial screen the selectable controller model. The tool allows, in real time, the modification on the controller structure as well as the parameters that define it. The user must configure the controller, in accordance to the desired control object. Establish the initial parameters, as well as which model best applies to the control object are in charge of the user. The configuration and definition of the input variables and possible disturbances simulated in the system must be set from the initial selection of the model. The program allows the test of control structures from an intuitive graphical interface. It also allows the acquisition of system data and the controller parameters, tuning techniques may apply. The proposed system evaluated in an experimental control plant, formed by an electric motor with encoder connected to the computational device by an electronic interface, have the features test and evaluation made by applying graphical methods. To allow interaction with the system a smartphone connects in the plant thru a webpage.

Key-Words: Control, Graphical tool, embedded controller, Internet of Things (IoT).

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

Prover uma tecnologia que permita, através de uma ferramenta, configurar e supervisionar remotamente uma planta de controle, oferecendo conectividade entre dispositivos e que atenda a demanda tecnológica de IoT (*Internet of Things* – Internet das Coisas) vem a ser relevante segundo Hermann Kopetz (KOPETZ, *Internet of things In Real-time systems.* Springer US, 2011 p.308) “Ao longo dos últimos 50 anos, a Internet cresceu exponencialmente a partir de uma pequena rede de pesquisa, que incluía apenas alguns nós, a uma rede mundial disseminada com mais de um bilhão de usuários. A miniaturização e a redução de custos de dispositivos eletrônicos possibilitam expandir a Internet para uma nova dimensão: objetos inteligentes”. É possível prover esta tecnologia aplicada a controladores com o desenvolvimento de uma ferramenta capaz de configurar parâmetros de controle, possibilitando acesso e operação remota, sem interferir no processo, permitindo a conectividade entre dispositivos através de uma rede de dados e tornando possível observar o comportamento do sistema em diversas configurações. Esta plataforma pode contribuir como ferramenta para o desenvolvimento de novos projetos.

A implementação dos programas computacionais que permitem o funcionamento deste sistema bem como o desenvolvimento da interface de sinais elétricos atuadores na planta de controle e sua interface gráfica (homem máquina) são escopo deste trabalho e compõem a plataforma. Os algoritmos de controle empregados em malha fechada são PID, PI e FUZZY, definidos em pesquisa documental realizada nas literaturas básicas da área de Controle e Automação de Processos. Segundo Katsuhiko Ogata, (OGATA, K. *Engenharia de Controle Moderno*. Terceira edição. 1998 p.544) “A utilidade dos controles PID reside na sua aplicabilidade geral à maioria dos sistemas de controle”.

1.1. OBJETIVOS

O objetivo deste trabalho é desenvolver uma plataforma que demonstre um sistema de controle implementado com o emprego de tecnologias de programas computacionais baseadas em uma arquitetura computacional em rede. Através de um sistema prático com aplicação de algoritmos de controle e supervisão, proporcionar o

acesso remoto de supervisão e interação oferecendo uma alternativa tecnológica para sistemas de controle no contexto de internet das coisas (IoT).

- Desenvolver um programa computacional capaz de atuar em uma planta de controle através de uma interface eletrônica;
- Projetar uma interface eletrônica para demonstrar o comportamento do sistema processando algoritmos de controle em paralelo e possibilitando ao usuário configurar parâmetros e alterar o controlador desejado.
- Integrar o programa computacional ao subsistema de armazenamento e telecomunicação de dados;
- Analisar o comportamento operacional do sistema com processamento paralelo de algoritmos de controle de acesso e supervisão remotos.

1.2. JUSTIFICATIVA

Esta plataforma proporcionará uma tecnologia para supervisionar e controlar processos de forma remota, sem interferência no processo. Isto permite uma operação e gerenciamento automatizado, utilizando como base o conceito de IoT (internet das coisas). A expectativa de crescimento para soluções baseadas em IoT é apresentada na Figura 1:

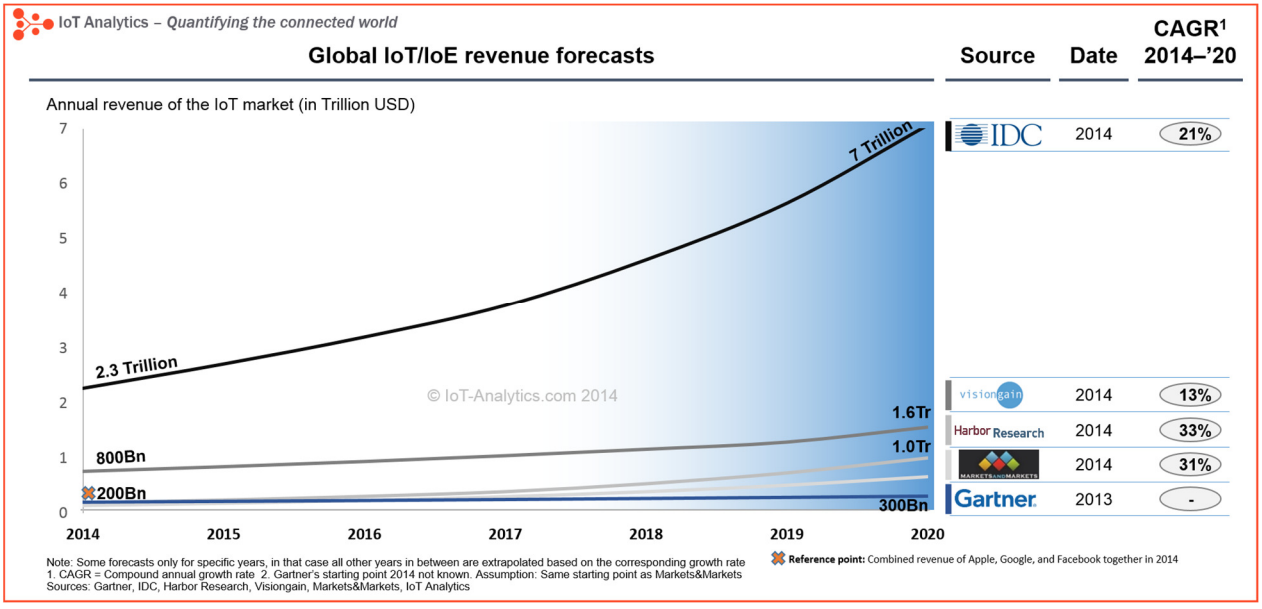


Figura 1 – Previsão Mercado IoT

Fonte: <http://theinternetofthings.report/Resources/Whitepapers/546e0460-8f65-4523-9eb5-9bf51b99f330_IoT%20market%20analysis%20Sizing%20the%20opportunity.pdf> Acesso em janeiro, 2016.

Com isso, este trabalho vem a ser relevante, atual e adequado para estudos na área de Engenharia de Automação e Controle. A utilização de um banco de dados embarcado ao sistema de controle proporciona a capacidade de armazenamento de eventos e registros, expansível de acordo com o tamanho do dispositivo de memória disponível, podendo chegar a armazenar anos consecutivos de dados. Uma aplicação que necessite armazenar dados por um prolongado período se beneficiará do uso desta tecnologia, a capacidade de armazenamento local, mesmo que de forma temporária, poderá trazer vantagens a muitas aplicações. Segundo Stan Schneider, (SCHNEIDER, 2016) a Internet revolucionou a forma como as pessoas se comunicam e trabalham. Marcou o início de uma nova era da informação livre para todos, mas a próxima onda da Internet não é sobre pessoas. Trata-se de dispositivos inteligentes, conectados. Para interagir com sucesso com o mundo real, estes dispositivos devem trabalhar em conjunto com velocidades e capacidades muito além do que as pessoas utilizam. A Internet das Coisas (Internet das coisas) vai mudar o mundo, talvez mais profundamente do que a Internet de hoje. Há aproximadamente um bilhão de sites na Internet, isso produziu resultados expressivos conforme Figura 2:

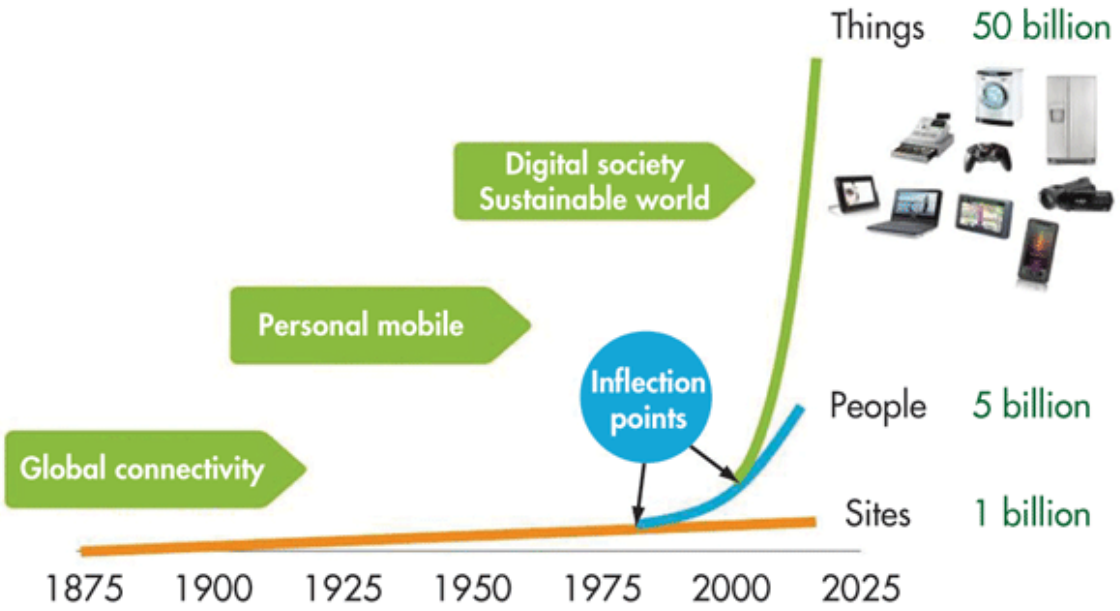


Figura 2 – Dispositivos conectados

Fonte: <http://www.electronicdesign.com/iot/understanding-protocols-behind-internet-things>
Acesso em janeiro, 2016.

Neste novo contexto que se torna realidade a justificativa deste trabalho se transforma em desafio ao prover uma tecnologia capaz de disponibilizar variáveis de uma planta de controle em um ambiente conectado.

1.3. ESTRUTURA DO TRABALHO

O Cap.2 apresenta o levantamento bibliográfico acerca dos conceitos empregados para os algoritmos de controle.

No Cap. 3 contém a teoria de funcionamento e explicação sobre os subsistemas, módulos e ferramentas utilizadas.

No Cap. 4 constam os materiais especificados.

O Cap. 5 contém listados e discutidos todos os resultados e métodos para os testes realizados de cada etapa do projeto.

O Cap. 6 apresenta as conclusões e possibilidades de trabalho futuro.

2. REVISÃO DA LITERATURA

Este trabalho foi possível por meio da pesquisa de literaturas que abordam diferentes áreas do conhecimento, principalmente Engenharia Elétrica nas subáreas: Automação Eletrônica de Processos Elétricos e Industriais, Controle de Processos Eletrônicos (Retroalimentação). Em Ciência da Computação nas subáreas: Sistemas de Computação e Metodologia e Técnicas da Computação. As literaturas referenciadas apresentam uma conexão entre estas áreas de conhecimento. De acordo com Bhagawandas Pannalal Lathi, (LATHI, B. P. *Modern Digital and Analog Communication Systems 3e Osece*. Oxford university press 1998), “Um sinal contínuo no tempo pode ser processado pelo processamento de suas amostras discretizadas no tempo. Por esta razão é importante manter uma taxa de amostragem suficientemente alta para que o sinal original possa ser reconstruído sem erro (ou com uma aceitável tolerância) através destas amostras.” Este conceito pode ser somado ao que é apresentado segundo Katsuhiko Ogata (OGATA, K. *Engenharia de Controle Moderno*. Terceira edição. 1998) “A modelagem matemática de um sistema dinâmico é definida como um conjunto de equações que representam a dinâmica do sistema com precisão ou, pelo menos, de forma bastante aceitável. Um sistema pode ser representado de muitas maneiras diferentes e, portanto, pode haver muitos modelos matemáticos, dependendo da

perspectiva que se considere.” Então é possível encontrar em (MITRA, 2006) métodos que descrevem aplicações de controle implementadas em ambientes computacionais.

A conexão entre as áreas do conhecimento, Engenharia Elétrica e Ciência da Computação está na implementação de programas computacionais dentro do ambiente de computação de dados que processará os algoritmos que executam o controle assim como na estrutura de rede que permitirá a comunicação de dados entre os dispositivos.

É possível definir que a arquitetura do sistema, foi concebida com base em conhecimentos publicados sobre IoT (internet das coisas), citado (KOPETZ, 2011) por sua contribuição na descrição de sistemas críticos implementados no contexto de IoT, com a tratativa de problemas como deadlock (impasse entre processos), sincronização de clock (ciclo de execução) e tolerância a falhas que devem ser considerados no desenvolvimento de sistemas. Em (HAHN, 2017) encontra-se exemplos de aplicações em IoT, onde fica evidente a importância dos protocolos de comunicação que estão bem descritos na referência (SCHNEIDER, 2016). Estes conhecimentos possuem como base a referência de (POSTEL, 1981) que detalha o protocolo internet.

A base de conhecimento para implementação dos programas computacionais está presente nas referências: (DATE et al, 1987) que detalha SQL, linguagem de consulta estruturada, o programa aplicativo desenvolvido em linguagem de programação interpretada descrita na referência (VAN ROSSUM 2009), para o programa computacional de rede desenvolvido está baseado na tecnologia descrita em (CASTAGNETTO et al, 1999). Estes somados aos conhecimentos em (RICHARDSON et al, 2012) que descrevem as possibilidades do dispositivo computacional adotado neste trabalho, então foi possível relacionar as referências (KO et al, 2001) onde está descrita uma arquitetura de sistema semelhante ao proposto neste trabalho, com a implementação de um servidor de dados como solução para acesso remoto. Semelhante ao proposto por (XIAOYI et al, 2016) em sua pesquisa descreve um controlador remoto de fluxo baseado em IoT.

Outros trabalhos com semelhante tema, na área de controle, foram relevantes; (ANG et al, 2005) descreve o projeto e ajuste de controladores PID, (ABDULAMEER et al, 2016) aplica o controlador PID em um motor DC descrevendo os procedimentos de sintonia dos parâmetros na planta de controle semelhante à adotada como experimento

prático deste trabalho. Estes trabalhos se basearam nos métodos propostos por (ZIEGLER e NICHOLS, 1942) de forma semelhante neste trabalho estão descritos os métodos aplicados na planta de controle adotada no experimento prático. Como referência de estudo sobre a máquina motor de corrente contínua encontra-se em (BARBI, 1985) um detalhado estudo sobre motores de indução. Este trabalho foi desenvolvido com base e inspiração nestas referências.

As referências adotadas mesclam conhecimentos atuais contento referências publicadas em 2017 com referências básicas e consagradas na área acadêmica sendo a mais antiga publicada em 1942. Esta diversidade de conhecimentos vem a contribuir com uma ligação entre as grandes áreas estudadas, tecnologias estão sendo desenvolvidas e impulsionam a evolução no ramo da computação, automação e controle. É importante o vínculo com as referências básicas porque possibilita a busca do conhecimento na fonte, todavia novas publicações conterão estes conhecimentos embutidos e poderão agregar conhecimentos que contribuem para melhoria e melhor descrição das aplicações de conceitos.

Neste trabalho as referências são atuais, contribuíram para o aperfeiçoamento da implementação e encurtamento do tempo necessário para concluir a implementação prática.

3. DESENVOLVIMENTO

Para realizar o processamento dos algoritmos de controle o programa computacional processará a digitalização do controlador, é permitido ao usuário selecionar controladores através de uma *webpage* (página de internet) bem como definir parâmetros. Determinando assim os fatores aplicados aos sinais adquiridos de entrada amostrados e digitalizados neste sistema de controle digital.

Para análise deste processamento de algoritmos de controle, é possível visualizar através de gráficos, dos sinais presentes no sistema em função do tempo, gerados pela ferramenta. Desta forma a seleção de algoritmos em tempo de execução é possível e o programa computacional executa o ciclo de controle, garantindo que seus resultados sejam processados e apresentados conforme o algoritmo selecionado. Na Figura 3 apresenta-se ilustrado um modelo simplificado:

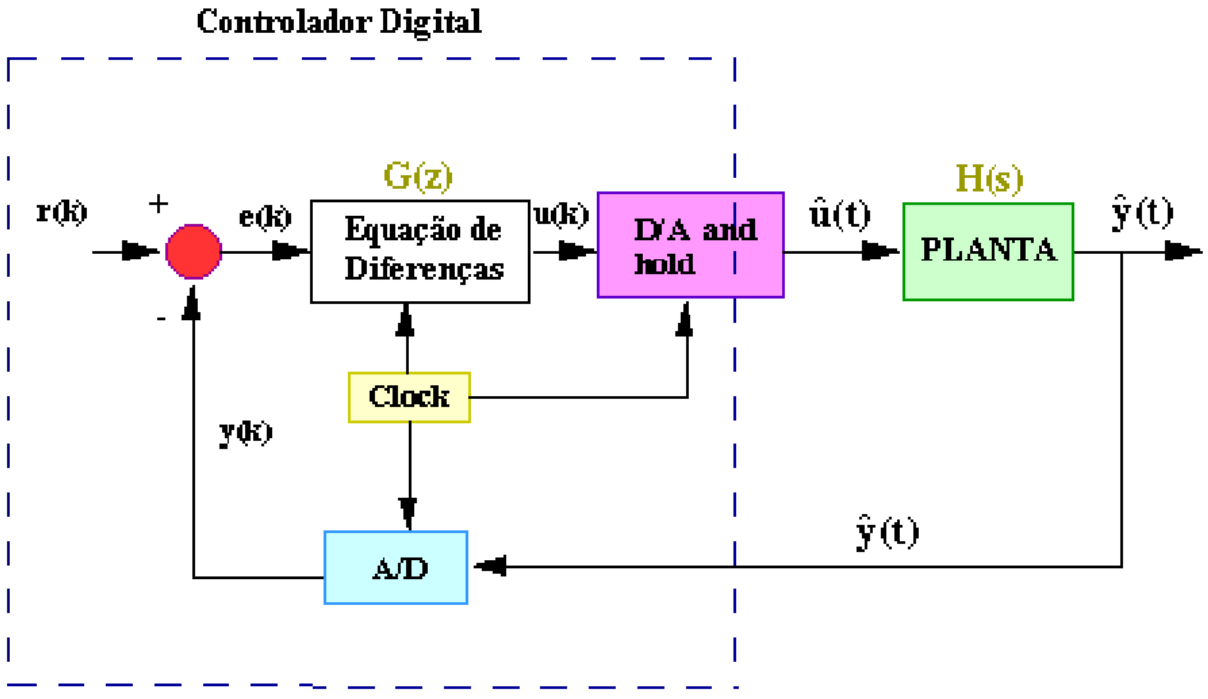


Figura 3 - Modelo Simplificado Controlador Digital

Fonte: Autor, 2016

O programa computacional realiza a interface com um subsistema de armazenamento, tratamento e telecomunicação de dados denominado de **servidor de dados**, neste contexto o subsistema é responsável por garantir a integridade dos dados que serão processados pelo aplicativo de controle bem como o armazenamento destes dados de forma remota com o tratamento e telecomunicação destes dados. O emprego desta tecnologia possibilita a dinâmica de operação remota em tempo real, proporcionando acesso sem interferir no processo executado na planta de controle.

O servidor de dados possibilita adicionalmente a escalabilidade do sistema, de forma que pode ser implementado não apenas em uma única planta de controle mas em diversas plantas, como por exemplo em uma linha de produção fabril onde diversos processos são controlados por ‘n’ controladores. Já neste contexto a supervisão remota se mostra uma eficiente e necessária ferramenta para o gerenciamento operacional. A implementação deste sistema possibilita o gerenciamento operacional de plantas de controle, a seleção de algoritmos, configuração de parâmetros e variáveis, e torna possível que isto seja feito sem interferência no processo. Para esta solução foi utilizado um “banco de dados”, capaz de armazenar e gerenciar informações. Disponibilizando interfaces de acesso e manipulação aos dados em uma arquitetura de rede. O seguinte diagrama em blocos representa, na Figura 4, as estruturas computacionais envolvidas nesta plataforma.

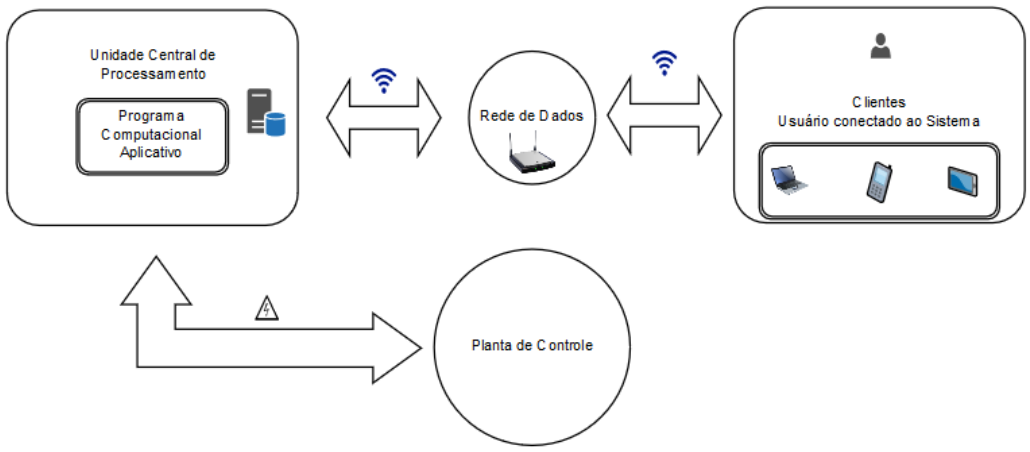


Figura 4 – Sistema

Fonte: Autor, 2017

A arquitetura do sistema consiste de dois subsistemas de processamento, ‘IHM Controlador Servidor’ e ‘Controlador Cliente’. Com suporte de comunicação de dados em rede local sem fios com o servidor de dados representado com a nomenclatura do subsistema computacional escolhido ‘c-treeACE’, conforme Figura 5:

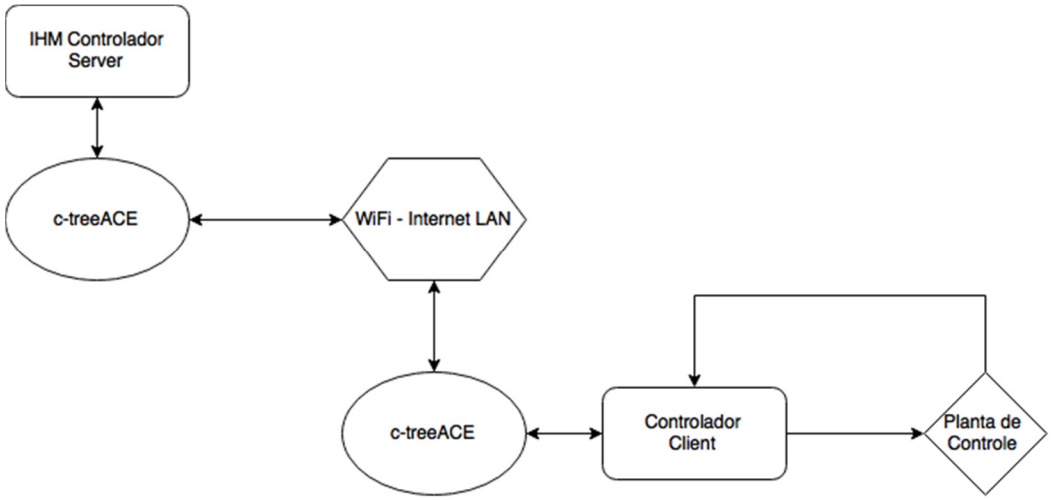


Figura 5 - Arquitetura básica do Sistema

Fonte: Autor, 2016

Para demonstrar o uso da plataforma em uma aplicação real foi utilizado como planta de controle um motor em miniatura de corrente elétrica contínua. A interface elétrica de acionamento é executada utilizando um componente eletrônico de forma a isolar o acionamento elétrico do comando eletrônico. Isto é necessário para proteger o dispositivo eletrônico computacional e permitirá o acionamento do motor e controle de velocidade em função da tensão elétrica aplicada que estará sendo controlada por um sinal PWM (*Pulse Width Modulation* – Modulação por Largura de Pulso).

3.1 Controle

Com a planta de controle experimental foi possível explorar a execução do sistema em malha aberta exercitando a operação e obtendo dados através da medição dos sinais elétricos que permitem determinar a velocidade e sentido de rotação do motor, também com estes sinais foi possível exercitar o sistema em malha fechada realizando a realimentação no sistema com o sinal de velocidade obtido. Os algoritmos sugeridos são: Malha Aberta, PI, PID e FUZZY.

3.1.1 Malha Aberta

Para malha aberta temos apenas uma conversão em que o sinal de entrada é submetido a uma multiplicação por uma fator proporcional e transformado no sinal de saída aplicado na planta do sistema.

$$f(t) = K\ entrada(t) \qquad (1)$$

3.1.2 PI

Já em malha fechada temos a aplicação de algoritmos de controle que utilizam realimentação, processando a leitura do sinal obtido no processo e assim corrigindo o sinal aplicado. Para o PI (Proporcional Integral) a função de transferência considerada é:

$$Gc(s) = Kp\ (1 + \frac{1}{Ti\ s}) \qquad (2)$$

Aplicando a transformada bi-linear:

$$s = \frac{2}{T} * \frac{1-z^{-1}}{1+z^{-1}} \qquad (3)$$

Resultará:

$$G(z) = Kp * (1 + \frac{1}{\left(\frac{2Ti}{T}\right)\frac{1-z^{-1}}{1+z^{-1}}}) \qquad (4)$$

E aplicando a transformada inversa teremos a equação de diferenças:

$$u(k) = u(k-1) + Kp\left(1 + \frac{T}{2Ki}\right)e(k) + Kp\left(-1 + \frac{T}{2Ki}\right)e(k-1) \qquad (5)$$

Onde T representa o intervalo de tempo entre amostras (ciclo de controle), Kp e Ki são respectivamente os termos Proporcional e Integral. Esta equação está implementada no programa computacional aplicativo sendo processada quando o algoritmo PI for selecionado.

O PI está representado na Figura 6:

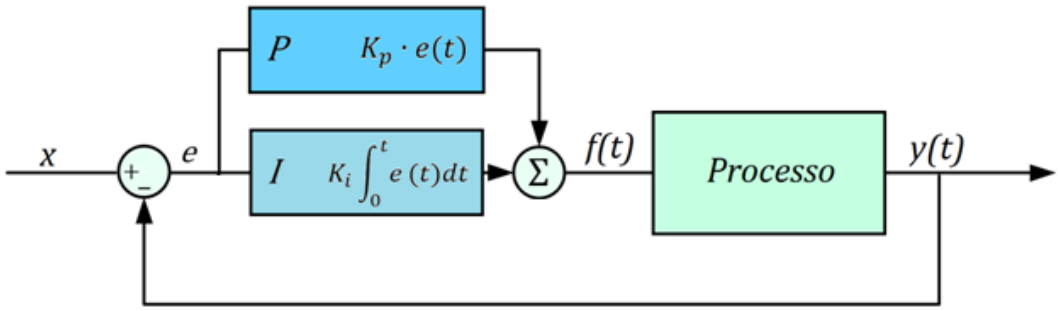


Figura 6 - PI

Fonte: Autor, 2017

3.1.3 PID

É possível considerar como mais popular o controlador PID (Proporcional Integral Derivativo). A seguinte equação define o controlador PID:

$$f(t) = Kp\, e(t) + Ki \int_0^t e(t)\, dt + Kd \frac{de(t)}{dt} \qquad (6)$$

Onde Kp, Ki e Kd são respectivamente os termos Proporcional, Integral e Derivativo. É possível representar este controlador com o seguinte diagrama de blocos, Figura 7:

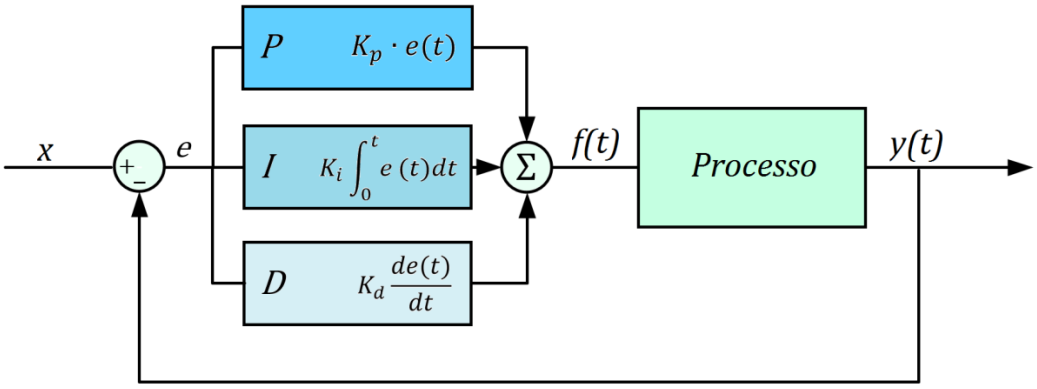


Figura 7 - PID

Fonte: Autor, 2017

E a função de transferência:

$$\frac{U(z)}{E(z)} = -\frac{Kp\left(T+2Ti+Tz^{-1}-2Tiz^{-1}\right)}{2Ti(z^{-1}-1)} \qquad (7)$$

A seguinte equação de diferenças representa o PID:

$$u(k) = u(k-1) + K1\, e(k) + K2\, e(k-1) + K3\, e(k-2) \quad (8)$$

Onde:

- K1 = Kp + Ki + Kd
- K2 = -Kp - 2 Kd
- K3 = Kd

Kp, Ki e Kd são respectivamente os termos Proporcional, Integral e Derivativo.

Neste contexto o experimento prático considera a velocidade obtida através da contagem de pulsos elétricos obtidos pela eletrônica embarcada junto a planta de controle como o sinal realimentado “y(t)”, a velocidade selecionada pelo usuário como a entrada “x”, o erro “e” que é a diferença entre a velocidade desejada e a velocidade obtida, os fatores configurados K, Ki e Kd respectivamente P, I e D.

O programa computacional aplicativo que está embarcado junto a planta de controle é responsável por processar este algoritmo, as configurações dos parâmetros são obtidas através do programa de rede que utiliza a tecnologia do banco de dados para realizar o gerenciamento e disponibilidade dos dados configurados para o programa computacional aplicativo.

3.1.4 FUZZY

É possível considerar FUZZY como o controlador mais moderno, a implementação considera as seguintes regras.

Se erro < fuzN3 então:

- pertinencia = 1.0
- valorsaida = defuzN3

Se (erro >= fuzN3) e (erro < fuzN2) então:

- pertinencia = erro/(fuzN3 + fuzN2)
- valorsaida = (pertinencia*defuzN3) + (pertinencia*defuzN2)

Se (erro >= fuzN2) e (erro < fuzN1) então:

- pertinencia = erro/(fuzN2 + fuzN1)
- valorsaida = (pertinencia*defuzN2) + (pertinencia*defuzN1)

Se (erro >= fuzN1) e (erro < fuzZ) então:

- pertinencia = erro/(fuzN1 + fuzZ)
- valorsaida = (pertinencia*defuzN1) + (pertinencia*defuzZ)

Se erro == fuzN1 então:

- pertinencia = 1.0
- valorsaida = defuzZ

Se (erro > fuzZ) e (erro <= fuzP1) então:

- pertinencia = erro/(fuzZ + fuzP1)
- valorsaida = (pertinencia*defuzZ) + (pertinencia*defuzP1)

Se (erro >= fuzP1) e (erro <= fuzP2) então:

- pertinencia = erro/(fuzP1 + fuzP2)
- valorsaida = (pertinencia*defuzP1) + (pertinencia*defuzP2)

Se (erro >= fuzP2) e (erro <= fuzP3) então:

- pertinencia = erro/(fuzP2 + fuzP3)
- valorsaida = (pertinencia*defuzP2) + (pertinencia*defuzP3)

Se erro > fuzP3 então:

- pertinencia = 1.0
- valorsaida = defuzP3

Onde os parâmetros fuzN3, fuzN2, fuzN1, fuzZ, fuzP1, fuzP2, fuzP3, defuzN3, defuzN2, defuzN1, defuzZ, defuzP1, defuzP2, defuzP3 são configuráveis e armazenados no banco de dados. E valorsaida representa a variação somada a saída do sistema.

A Figura 8 representa graficamente o modelo aplicado:

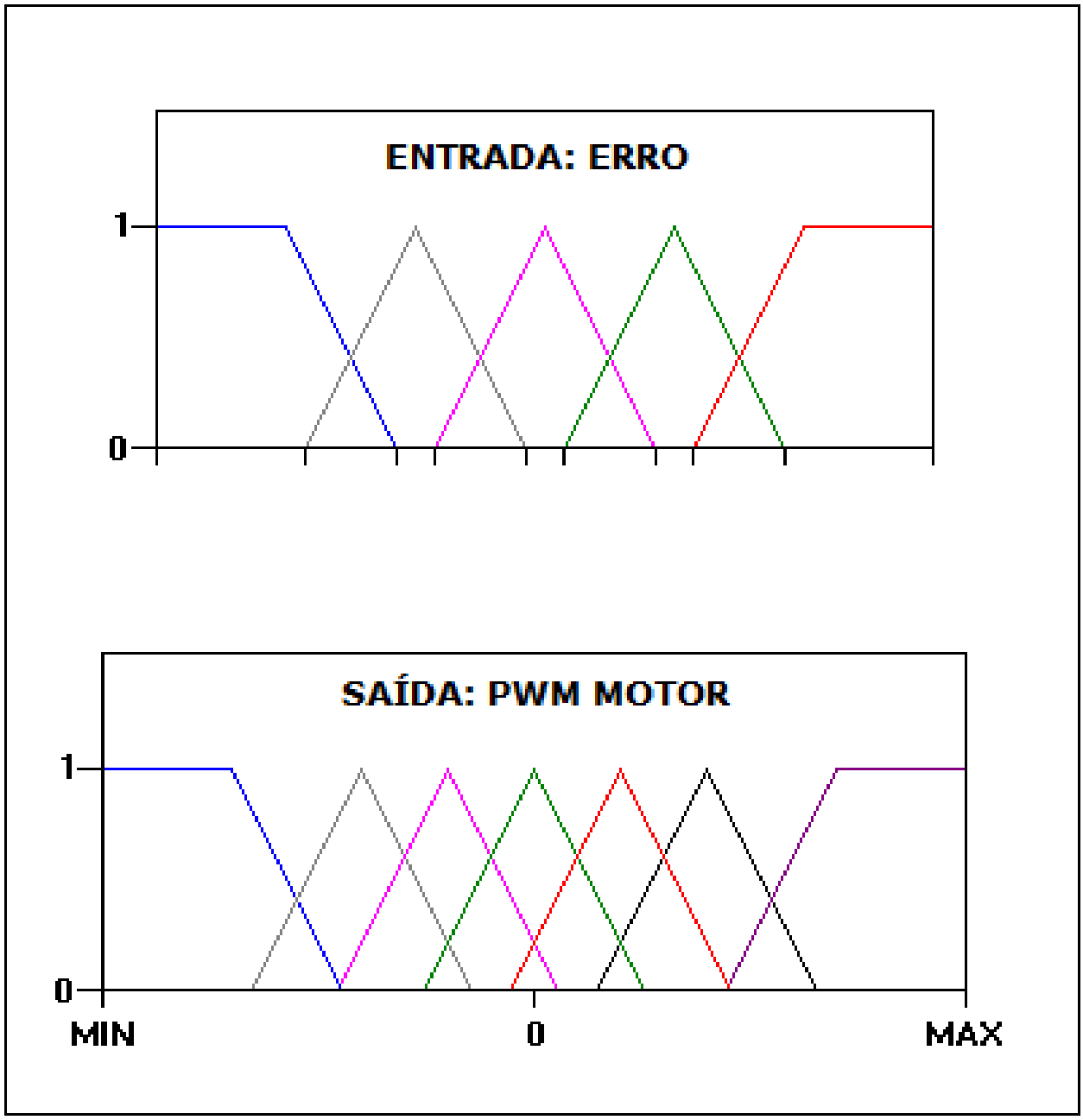


Figura 8 – FUZZY

Fonte: Autor, 2017

3.2 Programa Computacional

O programa computacional foi desenvolvido com base no sistema operacional Linux Debian em sua compilação distribuída para o subsistema computacional Raspberry Pi denominado Raspbian, a partir de plataforma de desenvolvimento Python que opera como interpretador de instruções, suas características foram descritas neste documento.

Para o módulo “IHM Controlador Servidor”, a fim de permitir a operação em diversos ambientes computacionais, é a realizada a troca de dados utilizando o banco de dados c-treeACE em sua interface Python API. O módulo “Controlador Cliente” foi desenvolvido em linguagem de programação PHP em operação embarcada ao servidor e com acesso deste módulo através de um serviço computacional de rede executado pelo componente “Apache” que disponibilizará acesso e execução de conteúdo através de páginas de rede de dados (internet).

Os subsistemas que compõe o programa computacional são apresentados na Figura 9:

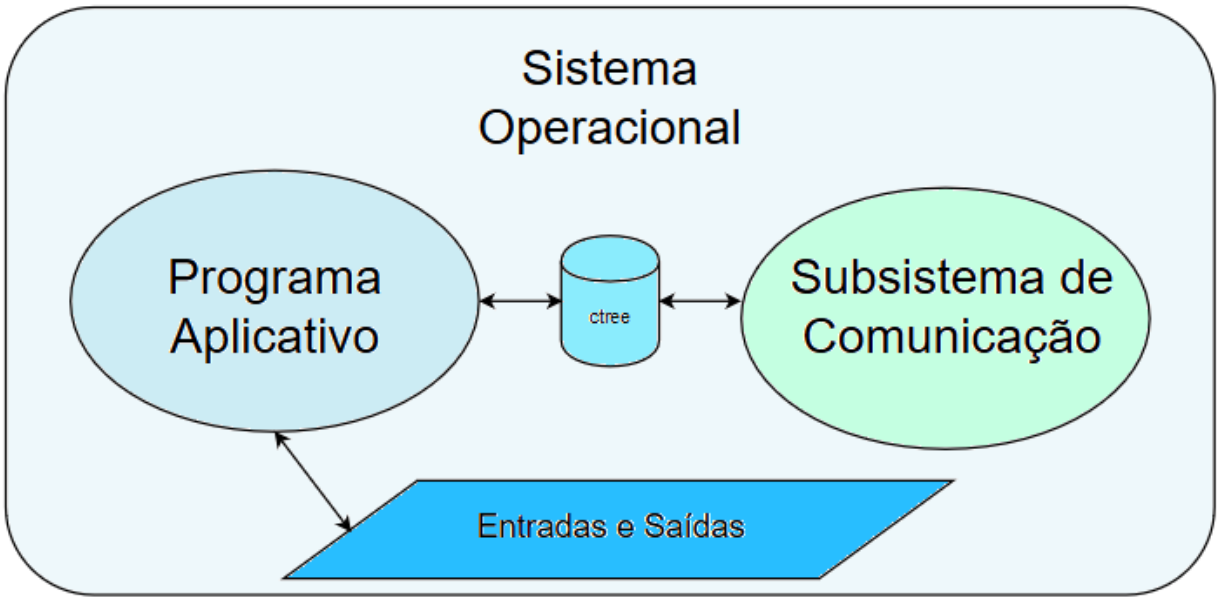


Figura 9 – Subsistemas do Programa Computacional

Fonte: Autor, 2016

3.2.1 Sistema Operacional

Para realizar o gerenciamento operacional do processador foi utilizado o sistema operacional Raspbian, o qual proporciona a base para o programa computacional aplicativo. Este sistema é responsável pelo gerenciamento e controle do clock de processamento, leitura e escrita de registros, endereçamento de memória e mapeamento de I/O's. Trata-se de uma distribuição Linux, compilada para processadores ARM em um pacote específico para o Raspberry Pi, denominado de Raspbian-Jessie, datado 9 de fevereiro de 2016. A Figura 10 representa a arquitetura:

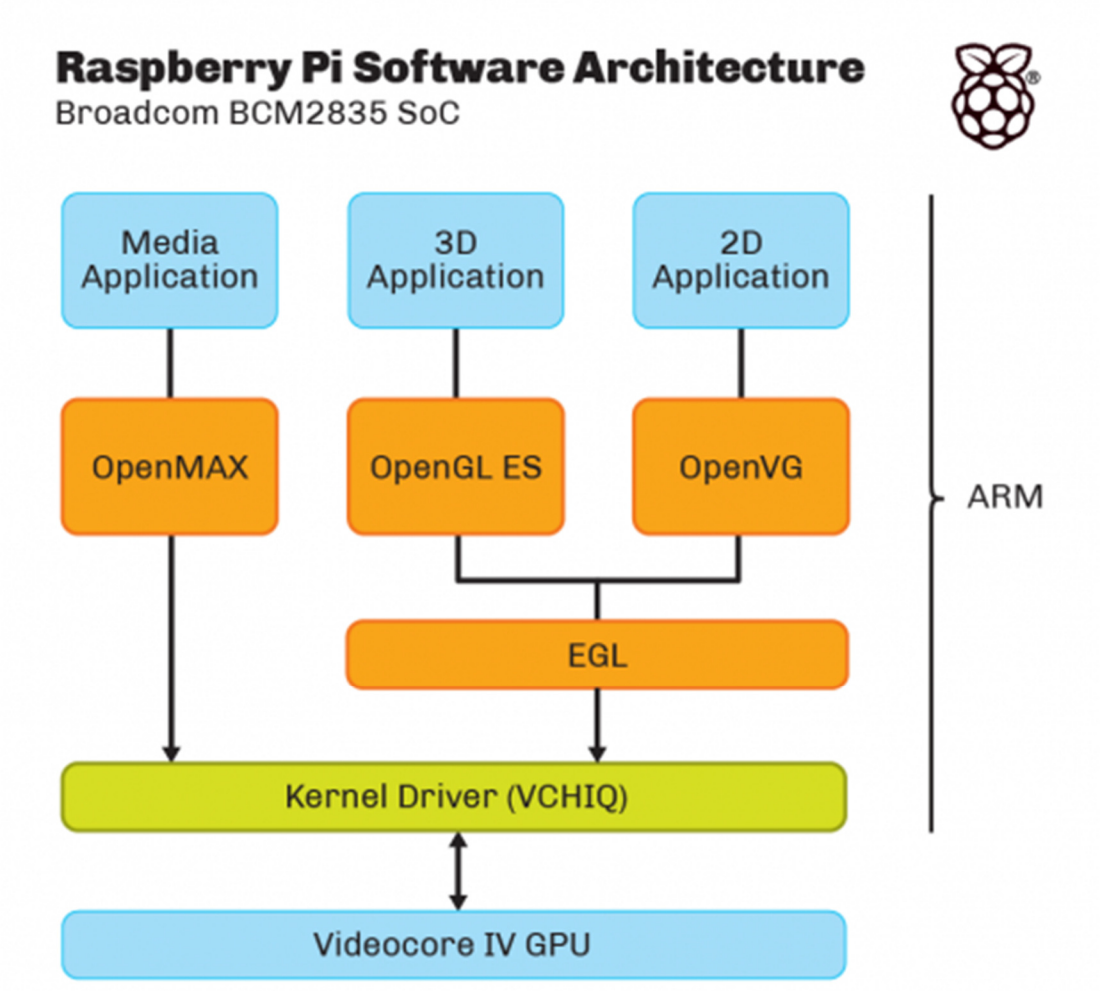


Figura 10 – Arquitetura do Sistema Operacional

Fonte: Raspberry Pi Foundation <http://www.raspberrypi.org/wp-content/uploads/2012/10/Architecture-and-Source.png>, Acesso em 2016

3.2.2 Programa Computacional Aplicativo

O programa computacional aplicativo atende aos requisitos funcionais de aplicação, realiza a leitura dos sinais de entrada no sistema, processa o cálculo do algoritmo de controle selecionado e fornece a saída aos atuadores do sistema. Desenvolvido com a linguagem de programação Python o que possibilita acesso às interfaces de baixo nível disponíveis no sistema operacional com acesso aos periféricos e as entradas e saídas de uso geral.

3.2.2.1 Python e SQL

O aplicativo desenvolvido na linguagem de programação Python foi escrito com base na referência (VAN ROSSUM 2009), foi adotada a interface com o banco de dados direta com comandos em SQL, a referência (DATE et al, 1987) detalha a linguagem de consulta estruturada. É possível resumir o funcionamento do programa aplicativo escrito em Python como um roteiro de instruções interpretadas que executam consulta de dados através de SQL, processamento do algoritmo selecionado de controle e escrita nos periféricos de entrada e saída bem como escrita no banco de dados.

3.2.2.2 Gerenciamento de Entradas e Saídas

Este módulo se encarrega de tratar as conversões necessárias entre os valores lidos pelo sistema em unidades físicas, de forma a possibilitar a interpretação dos dados bem como a conversão de valores para atuação no sistema.

Estes pinos são uma interface física entre o dispositivo computacional e os dispositivos externos. No nível mais simples, pode-se pensar neles como interruptores que permitem ler sinais (entrada) ou escrever sinais (saída). Dezessete dos 26 pinos são pinos GPIO (*General Purpose Input Output*) que significa Entradas e Saídas de Uso Geral. Os demais são pinos de alimentação ou terra. Possibilita programar os pinos para interagir com o mundo real. Entradas podem ser introduzidas a partir de um sensor ou um sinal de outro computador ou dispositivo, por exemplo. A saída também pode ligar um LED para enviar um sinal ou dados para outro dispositivo. Se o dispositivo computacional estiver em uma rede, é possível controlar os dispositivos que estão conectados a ele de qualquer lugar e esses dispositivos podem enviar dados de volta.

Este gerenciamento oferece conectividade e controle de dispositivos físicos através da Internet.

3.2.3 Programa Computacional de Rede

O subsistema de comunicação é composto de módulos do sistema operacional e os programas computacionais de rede desenvolvidos, sendo as páginas desenvolvidas em HTML e os roteiro interpretado em linguagem PHP.

3.2.3.1 HTML e PHP

O programa computacional de rede que é composto de páginas HTML e roteiros PHP forma junto com o gerenciamento de comunicação do sistema operacional o subsistema de comunicação. Estas tecnologias foram adotadas por permitirem a operação por qualquer dispositivo capaz de conectar a rede através de um navegador de internet, permitindo uma operação através de um computador ou um aparelho de telefonia celular. A referência (CASTAGNETTO et al, 1999) descreve em detalhes os comandos e sintaxes destas linguagens utilizadas.

3.2.4 Banco de Dados

Neste trabalho foi aplicado o banco de dados c-treeACE da FairCom, que possui diversas interfaces, tanto orientadas a registros (ISAM, método de acesso sequencial indexado) quanto relacionais (“SQL”). Sendo uma solução que exige poucos recursos computacionais. O banco permite ainda gerenciamento e controle do tráfego de dados na rede. Este componente estará configurado com acesso e persistência em memória não volátil, e com isso, acesso ao histórico de dados durante toda a operação, e até com registros gravados em operações anteriormente executadas. É utilizada a interface de acesso ao banco de dados denominada ODBC API para o módulo controlador cliente (Programa Computacional de Rede), implementado em linguagem PHP. Para o programa computacional aplicativo foi utilizada a interface Python API.

O c-treeACE foi adotado para este desenvolvimento por facilitar o desenvolvimento do programa computacional, objeto deste trabalho, provendo interfaces de fácil utilização e suporte multi plataforma. A Figura 11 ilustra as interfaces utilizadas:

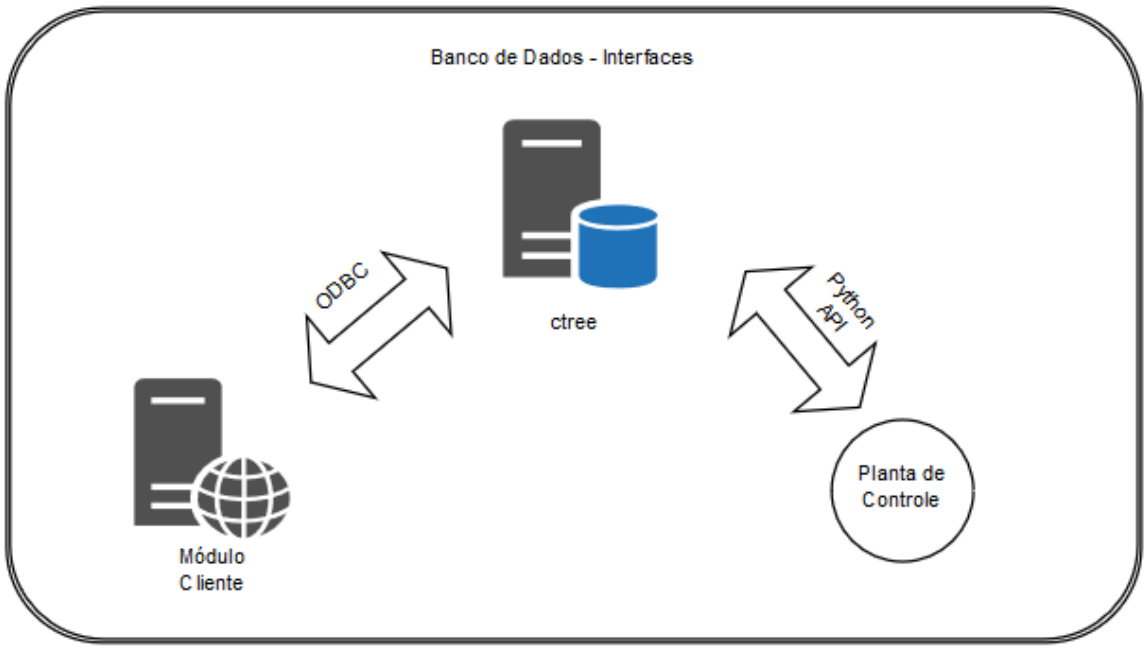


Figura 11 – Banco de dados

Fonte: Autor, 2017

3.2.4.1 FairCom c-treeACE

O servidor c-treeACE suporta gerenciamento de banco de dados de alto nível, foi adotado para este projeto por atender ao requisito de operar no sistema operacional Raspbian e possuir interface com programas desenvolvidos em linguagens Python e PHP.

Está descrito neste documento, em “Anexo I”, as funções e características desta ferramenta. Adicionalemente é possível obter outras informação na página de internet do fabricante, na seção documentos em: <http://www.faircom.com/developers/documentation>. Acesso em 2016.

3.2.4.2 SQL

SQL, sigla em língua estrangeira *Structed Query Language* e traduzida como Linguagem de Consulta Estruturada é uma linguagem para acesso e pesquisa para banco de dados relacional.

O padrão para a linguagem foi realizado pela American National Standards Institute (ANSI) em 1986 e ISO em 1987. O SQL foi revisto em 1992 e a esta versão foi dado o nome de SQL-92. Foi revisto novamente em 1999 e 2003 para se tornar SQL:1999 (SQL3) e SQL:2003, respectivamente. O SQL:1999 usa expressões regulares de emparelhamento, queries recursivas e gatilhos (triggers). Também foi feita uma adição controversa de tipos não-escalados e algumas características de orientação a objeto. O SQL:2003 introduz características relacionadas ao XML, sequências padronizadas e colunas com valores de auto-generalização (inclusive colunas-identidade). Possibilita que código de idioma procedural possa ser embutido e interaja com o banco de dados. Incluindo uma programação em linguagem Java na base de dados e armazenados na forma de *Stored Procedures* ou Procedimento Armazenado. Neste trabalho foi empregado o uso desta linguagem para armazenar os dados obtidos no sistema e ajustar os valores de calibração. Está descrito em “Anexo II” os subsistemas que compõe o SQL.

3.2.4.3 Base de dados

Foi organizado nesta plataforma a base de dados composta de estruturas representadas em SQL.

3.2.4.3.1 Configura

A tabela “configura” contém a estrutura de configuração para o algoritmo de conversão dos pulsos elétricos, obtidos pela interface eletrônica, em velocidade. Adicionalmente contém a configuração do intervalo de tempo para armazenar dados do sistema e a quantidade de pontos a serem criados na geração de gráficos.

```
create table "admin"."configura" ("pulsosvelcalc" integer,
      "tempovelcalc" float (8),
      "tempociclo" float (8),
      "grafpts" integer );
```

3.2.4.3.2 Controle

A tabela “controle” contém a estrutura de parâmetros para o algoritmo de controle. Adicionalmente contém a configuração da entrada para o sistema de controle.

```
create table "admin"."controle" (
      "ajuste" integer,
      "valork" float (8),
      "valorki" float (8),
      "valorkd" float (8) );
```

3.2.4.3.3 Dados

A tabela “dados” contém a estrutura de dados armazenados do sistema. Adicionalmente contém a criação de índices que otimizam a pesquisa nesta estrutura de dados.

```
create table "admin"."dados" (
      "ajuste" integer,
      "velocidade" integer,
      "erro" integer,
      "kp" float (8),
      "ki" float (8),
      "kd" float (8),
      "kmotor" float (8),
      "tempo" timestamp
);
create index "admin"."indadados" on "admin"."dados" ("ajuste");
create index "admin"."indtdados" on "admin"."dados" ("tempo");
create index "admin"."indvdados" on "admin"."dados" ("velocidade");
```

3.2.5 Subsistema de Comunicação

Este subsistema realiza a troca de dados entre o módulo server e o módulo client, garantindo que o sistema entenda a troca de informações de forma direta, abstraindo a transmissão e recepção de dados através do protocolo de comunicação TCP/IP. Foi adicionado neste subsistema os componentes de serviço computacional de rede disponibilizando acesso e execução de conteúdo através de páginas de rede de dados (internet). A disposição destes componentes é representada pela Figura 12:

Figura 12 – Tela Subsistema de Comunicação

Fonte: Autor, 2017

Foi utilizado o protocolo de comunicação TCP/IP, descrito em “Anexo III – TCP/IP”, devido aos benefícios de padronização, ser um protocolo robusto, escalável, multiplataforma, prover interconectividade e por ser o protocolo de acesso a internet.

3.3 Unidade Central de Processamento

A missão de realizar o ciclo de controle deve ser executada em tempo real, o que determina atender ao requisito de resposta esperada pelo sistema mecânico. A Figura 13 representa o controlador e a interação com os dados, operações de processamento:

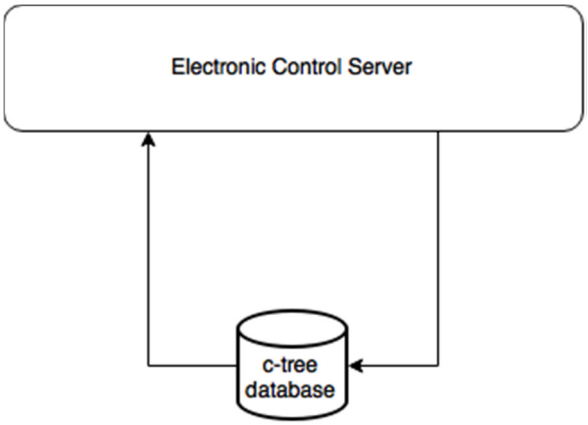


Figura 13 – Processamento de dados

Fonte: Autor, 2016

A atuação do sistema na planta é realizada por um controlador escravo conforme Figura 14:

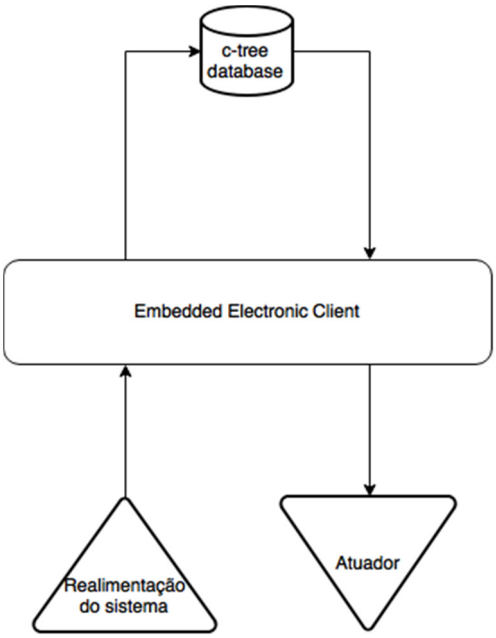


Figura 14 – Processamento de atuação

Fonte: Autor, 2016

3.4 Estrutura de dados

Foi estabelecido como padrão utilizar três tabelas de dados para compor as informações entre o centro de controle e a unidade controladora. Estes elementos são conforme Figura 15:

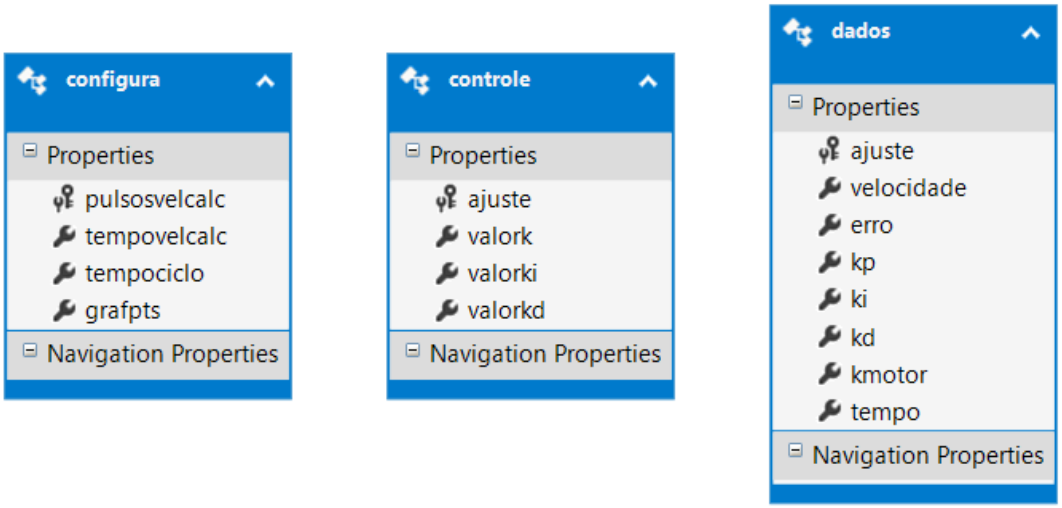


Figura 15 – Estrutura de dados

Fonte: Autor, 2016

3.4.1 Configura

A tabela “configura” contém a configuração da quantidade de pulsos a ser contadas para executar o cálculo da velocidade “pulsosvelcalc”:

```
"pulsosvelcalc" integer,
```

Em “tempovelcalc” está a configuração do intervalo máximo para o cálculo de velocidade uma vez que em baixas velocidades ou em estado de repouso não haveriam pulsos obtidos:

```
"tempovelcalc" float (8),
```

No elemento “tempociclo” está a configuração do intervalo de tempo para armazenamento dos dados obtidos no sistema:

```
"tempociclo" float (8),
```

Em “grafpts” está a quantidade de pontos a serem gerados na criação do gráfico.

```
"grafpts" integer );
```

3.4.2 Controle

A tabela “controle” contém a configuração da entrada para o sistema de controle no elemento “ajuste”:

```
"ajuste" integer,
```

No elemento “valork” está a configuração do parâmetro de controle para o ganho do sistema:

```
"valork" float (8),
```

No elemento “valorki” está a configuração do parâmetro de controle para o integrador do sistema:

```
"valorki" float (8),
```

No elemento “valorkd” está a configuração do parâmetro de controle para o derivativo do sistema:

```
"valorkd" float (8) );
```

3.4.3 Dados

A tabela “dados” contém a estrutura de dados armazenados do sistema. Está armazenado em “ajuste” o valor selecionado para a entrada do sistema.

```
"ajuste" integer,
```

Está armazenado em “velocidade” o valor calculado e que determina a velocidade do sistema.

```
"velocidade" integer,
```

Em “erro” está o sinal de entrada subtraído do sinal realimentado no sistema.

```
"erro" integer,
```

No elemento “kp” está o valor da configuração do parâmetro de controle para o ganho do sistema:

```
"kp" float (8),
```

No elemento “ki” está o valor da configuração do parâmetro de controle para o integrador do sistema:

```
"ki" float (8),
```

No elemento “kd” está o valor da configuração do parâmetro de controle para o derivativo do sistema:

```
"kd" float (8),
```

No elemento “kmotor” está o valor da configuração do parâmetro que determina a conversão que transforma a velocidade desejada em uma porcentagem que equivalerá ao valor imposto a largura de pulso da saída do sistema:

```
"kmotor" float (8),
```

No elemento “tempo” está o registro da data e hora em que o armazenamento foi executado:

```
"tempo" timestamp
```

3.5 Interface com usuário

É possível ao usuário realizar a configuração bem como a seleção de controladores sugeridos, os parâmetros iniciais devem ser informados pelo usuário e na ausência de dados é carregado pelo sistema uma configuração padrão. Esta interface é constituída de um conjunto de informações visuais que foram denominados como telas, é possível então dividir as funcionalidades do aplicativo em telas sendo as macro divisões: Controle, Configura, Gráfico e Abortar. A Figura 16 representa a tela inicial:

Figura 16 – Tela Inicial

Fonte: Autor, 2017

3.5.1 Configuração

A primeira etapa a ser considerada para utilização da plataforma é a configuração, nesta fase o usuário deverá considerar o seu ambiente, a planta de controle e a capacidade computacional do sistema.

Estão descritos, neste capítulo, as telas e os campos de configuração disponíveis.

3.5.1.1 Tela Configura

Para acessar esta tela a partir da tela inicial é necessário clicar na opção “Configura”, conforme Figura 17:

Figura 17 – Tela Inicial, opção Configura

Fonte: Autor, 2017

Esta tela contém as configurações da plataforma onde é possível configurar a quantidade de pulsos obtidos para executar o cálculo da velocidade, o intervalo de tempo máximo para executar o cálculo de velocidade, o intervalo de tempo para armazenamento dos dados do sistema e a quantidade de pontos processados na exibição gráfica, conforme Figura 18:

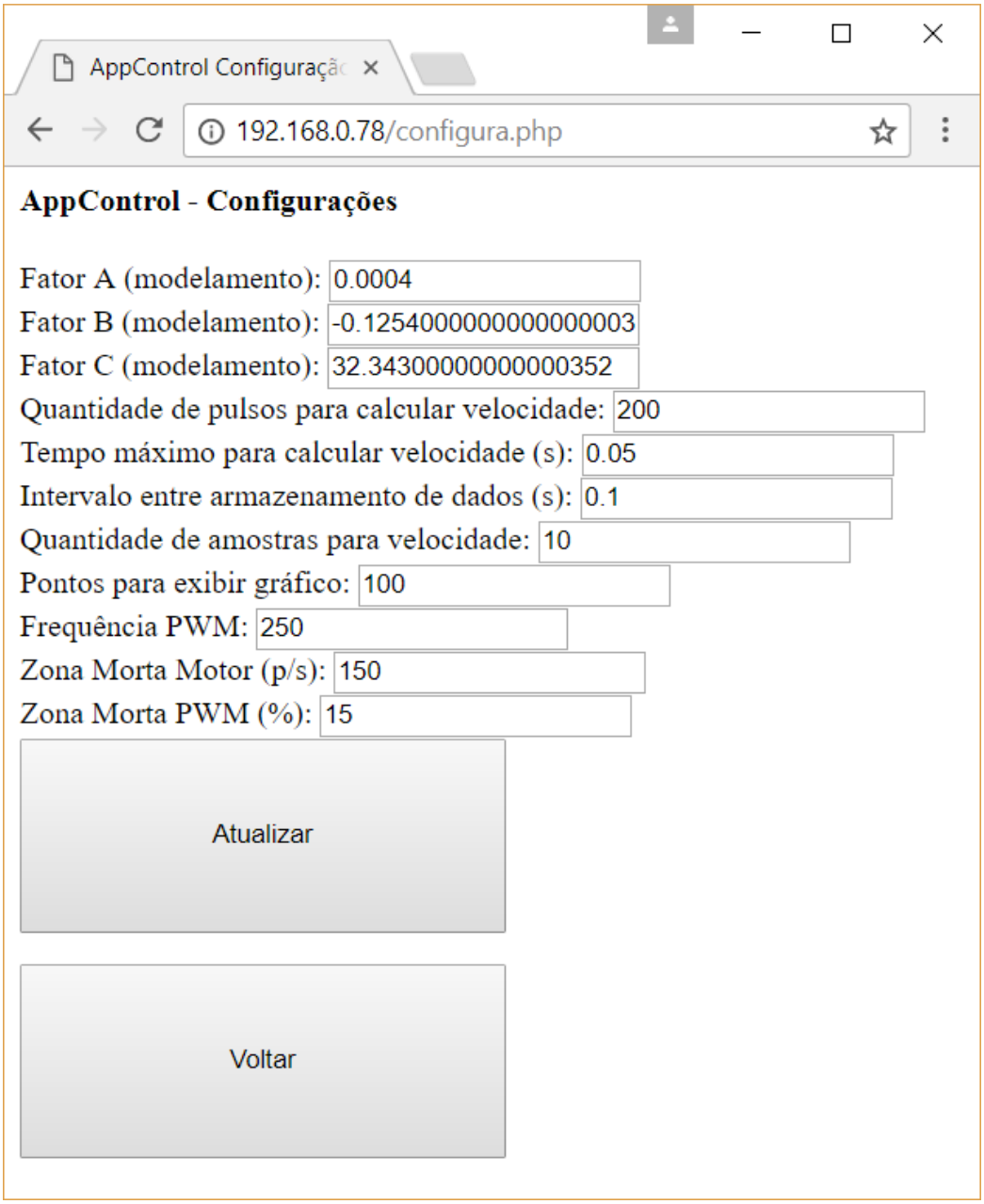


Figura 18 – Tela Configuração de Plataforma

Fonte: Autor, 2017

Os elementos de configuração referente a função que relaciona a velocidade lida em pulsos por segundo com a tensão elétrica aplicada na planta através do PWM, são representados por Fator A, Fator B e Fator C, correspondendo aos elementos a, b e c de uma equação de segundo grau:

$$y = ax^2 + bx + c \qquad (9)$$

A frequência empregada no sinal de saída PWM é configurável, e pode ser alterada pelo usuário, adicionalmente os parâmetros Zona Morta Motor e Zona Morta PWM possibilitam configurar o valor mínimo de sinal PWM aplicada a planta em função da Zona Morta encontrada no sistema. No experimento prático a zona morta refere-se a faixa de atuação em que a máquina motor de corrente contínua permanece estática mesmo com a aplicação de um determinado nível de tensão elétrica.

3.5.2 Controle

Para acessar esta tela a partir da tela inicial é necessário clicar na opção “Controle”, nesta etapa considera-se que a ferramenta estará corretamente configurada, isto é, de acordo com a planta que se deseja controlar. Conforme Figura 19:

Figura 19 – Tela Configuração de Controle

Fonte: Autor, 2017

Esta tela abrirá a possibilidade de selecionar o algoritmo que deseja utilizar, podendo ser em malha aberta ou fechada. Sendo as opções de seleção conforme Figura 20:

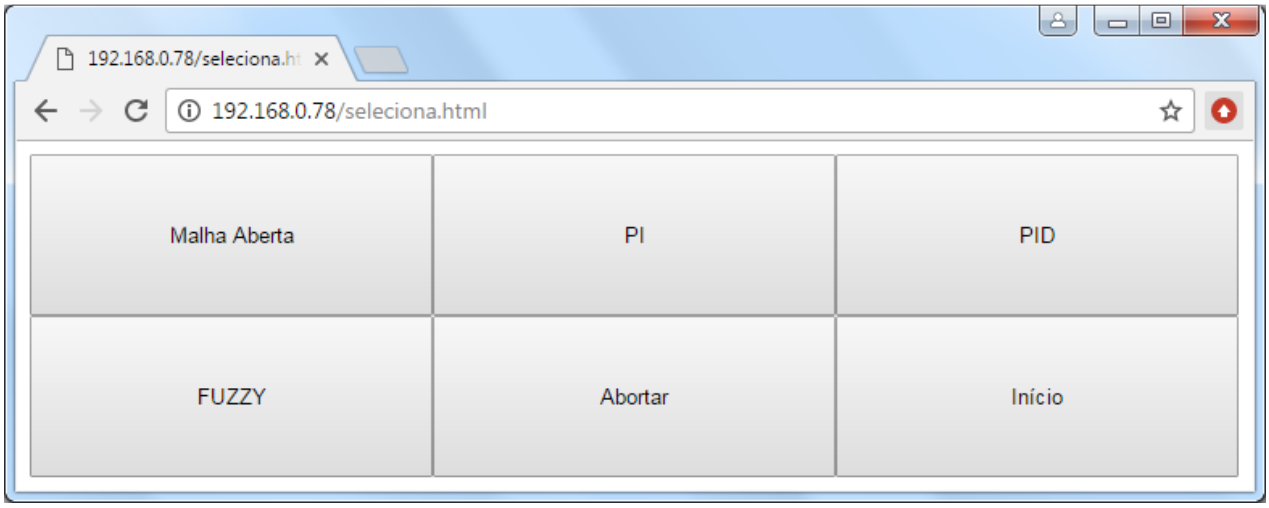


Figura 20 – Tela Seleção de Controle

Fonte: Autor, 2017

Em malha fechada temos as opções de 'PI', 'PID' e 'FUZZY', com a opção 'Malha Aberta' é possível exercitar a planta a fim de obter o comportamento do sistema de forma empírica, aplicando sinais e verificando a resultante obtida na planta. Com a finalidade de facilitar a operação da ferramenta está disponível a opção de 'Abortar' a execução de um algoritmo. Em 'Início' é possível ao usuário retornar a tela inicial caso queira realizar alterar alguma configuração.

Os programas computacionais de controle implementados possuem um mecanismo de proteção que têm a finalidade de evitar que um processo em execução seja interrompido por uma nova requisição involuntária, isto garante que apenas um algoritmo estará em execução e que caso o usuário opte por alterar esta seleção deverá executar o comando abortar através do botão 'Abortar'.

Este mecanismo executa este controle através da escrita em arquivos temporários que são apagados ao término da execução de um programa computacional.

3.5.2.1 Tela PI

A opção PI, Figura 21, contém as configurações do controlador onde é possível configurar a velocidade desejada em pulsos por segundo, o fator de ganho proporcional e o fator de ganho integrador:

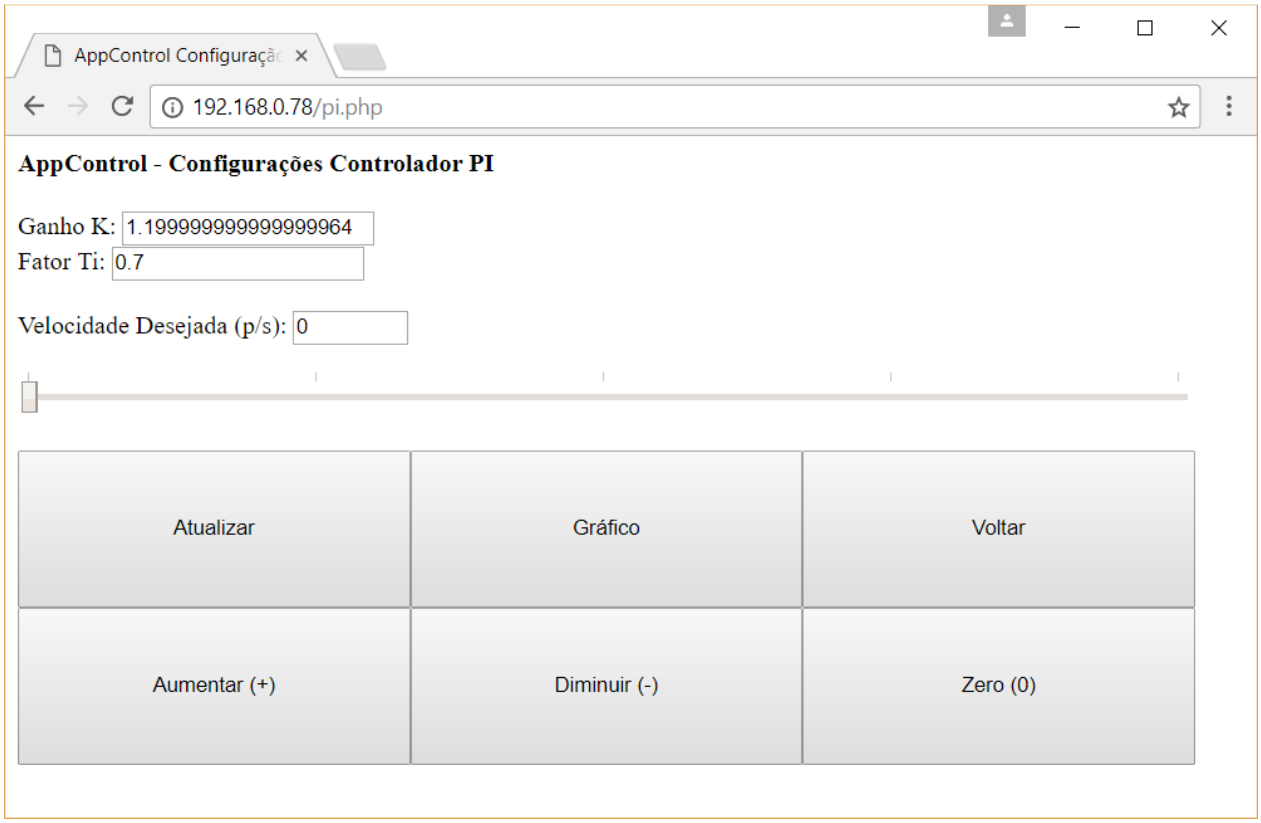


Figura 21 – Tela Seleção de Controle

Fonte: Autor, 2017

Também é possível através dos botões ‘Aumentar (+)’, ‘Diminuir (-)’ e ‘Zero (0)’ selecionar a velocidade desejada em passos, sendo o incremento de 10 pulsos/s. Para conveniência o botão ‘Gráfico’ está disponível na mesma tela permitindo ao usuário visualizar em tempo de execução a resposta do sistema através de um gráfico das velocidades obtidas e desejadas em função do tempo. O botão ‘Atualizar’ realiza a gravação dos dados selecionados no sistema e o botão ‘Voltar’ é a opção para retornar a tela anterior.

3.5.2.2 Tela PID

A opção PID, Figura 22, contém as configurações do controlador onde é possível configurar a velocidade desejada em pulsos por segundo, o fator de ganho proporcional, o fator de ganho integrador e o fator de ganho da componente derivativa, assim como na opção PI temos a mesma interface gráfica a fim de facilitar a operação para o usuário:

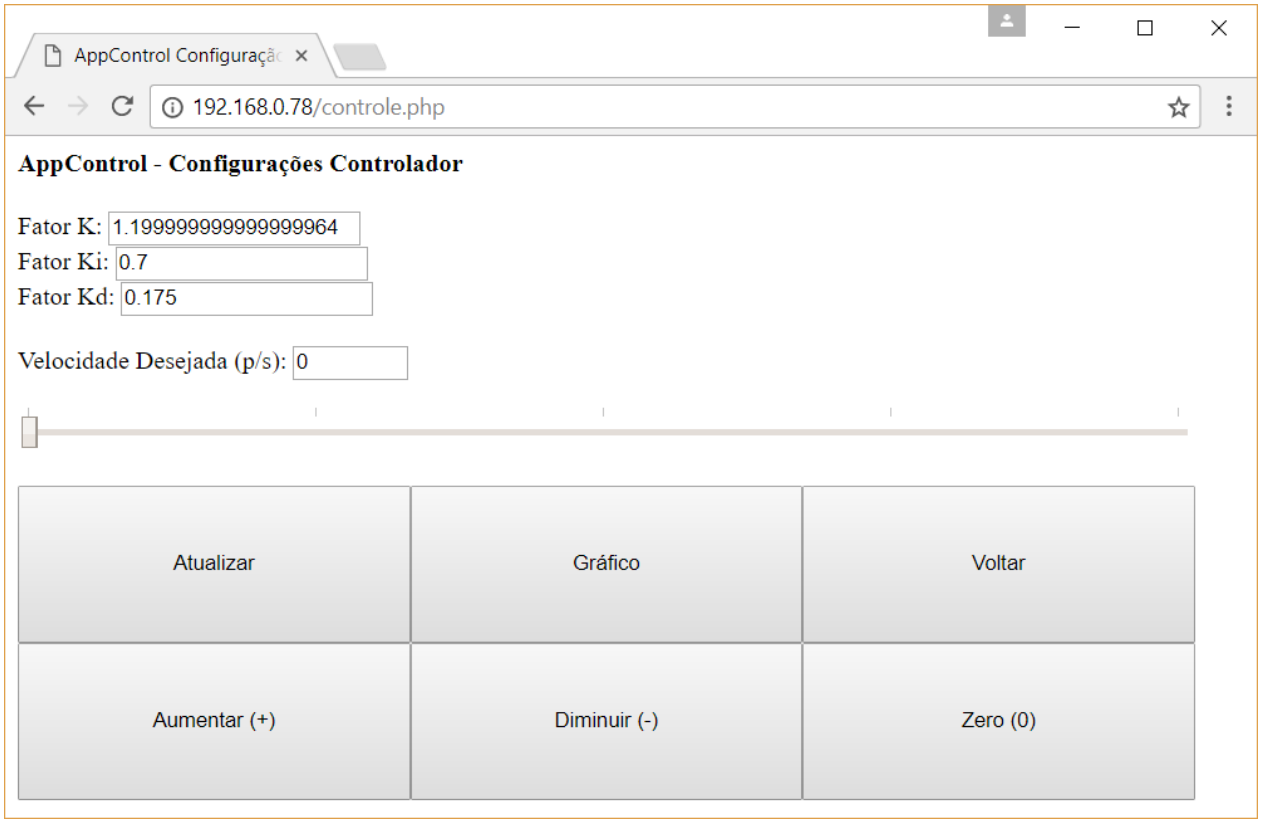


Figura 22 – Tela Configurações de Controle

Fonte: Autor, 2017

3.5.2.3 Tela Malha Aberta

A opção ‘Malha Aberta’, Figura 23, contém as configurações de velocidade desejada em pulsos por segundo, o fator de ganho, assim como nas opções anteriores temos a mesma interface gráfica a fim de facilitar a operação para o usuário:

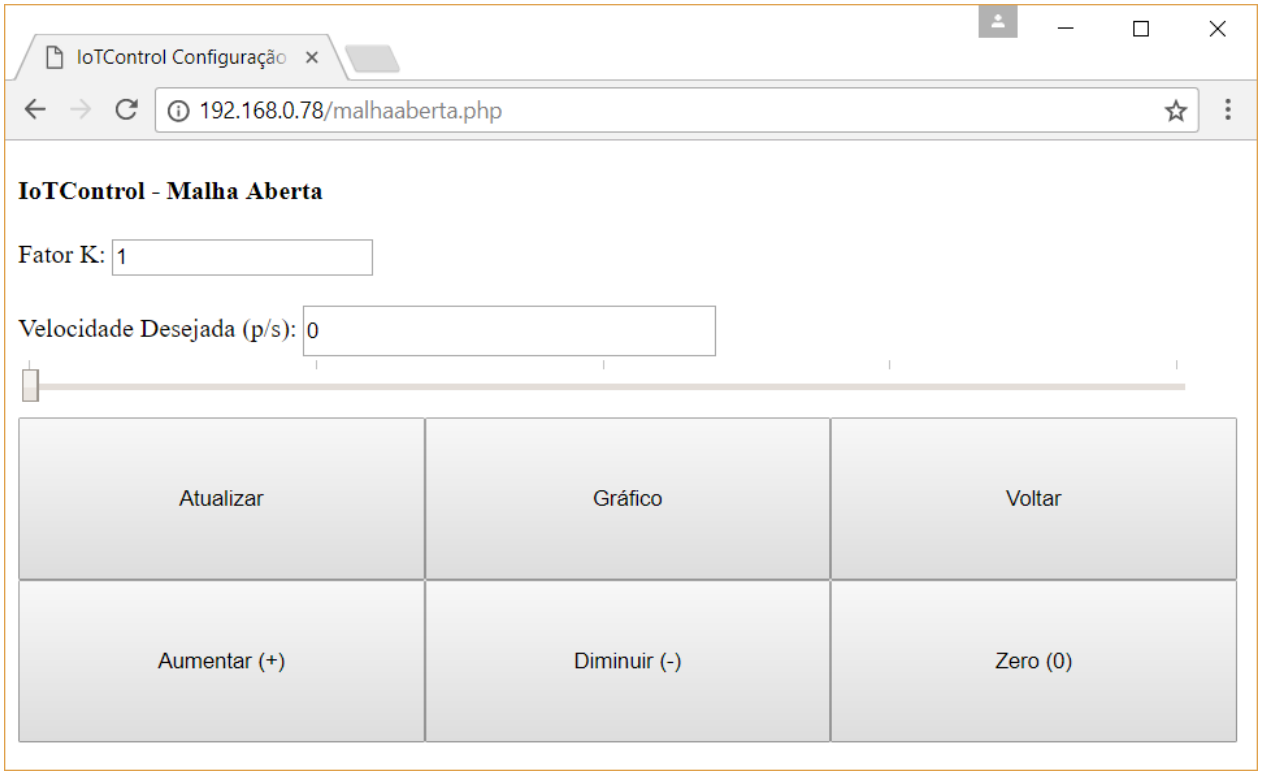


Figura 23 – Tela Malha Aberta

Fonte: Autor, 2017

3.5.2.4 Tela FUZZY

A opção ‘FUZZY’ contém as configurações de velocidade desejada em pulsos por segundo, os parâmetros fuzN3, fuzN2, fuzN1, fuzZ, fuzP1, fuzP2, fuzP3, defuzN3, defuzN2, defuzN1, defuzZ, defuzP1, defuzP2, defuzP3, assim como nas opções anteriores temos a mesma interface gráfica a fim de facilitar a operação para o usuário.

3.5.3 Gráfico

Esta tela, Figura 24, é a opção que permite ao usuário estudar o comportamento do sistema, nesta é gerado um gráfico das velocidades desejada e obtida em função do tempo, onde é possível observar sobressinal, tempo de amortecimento entre outros:

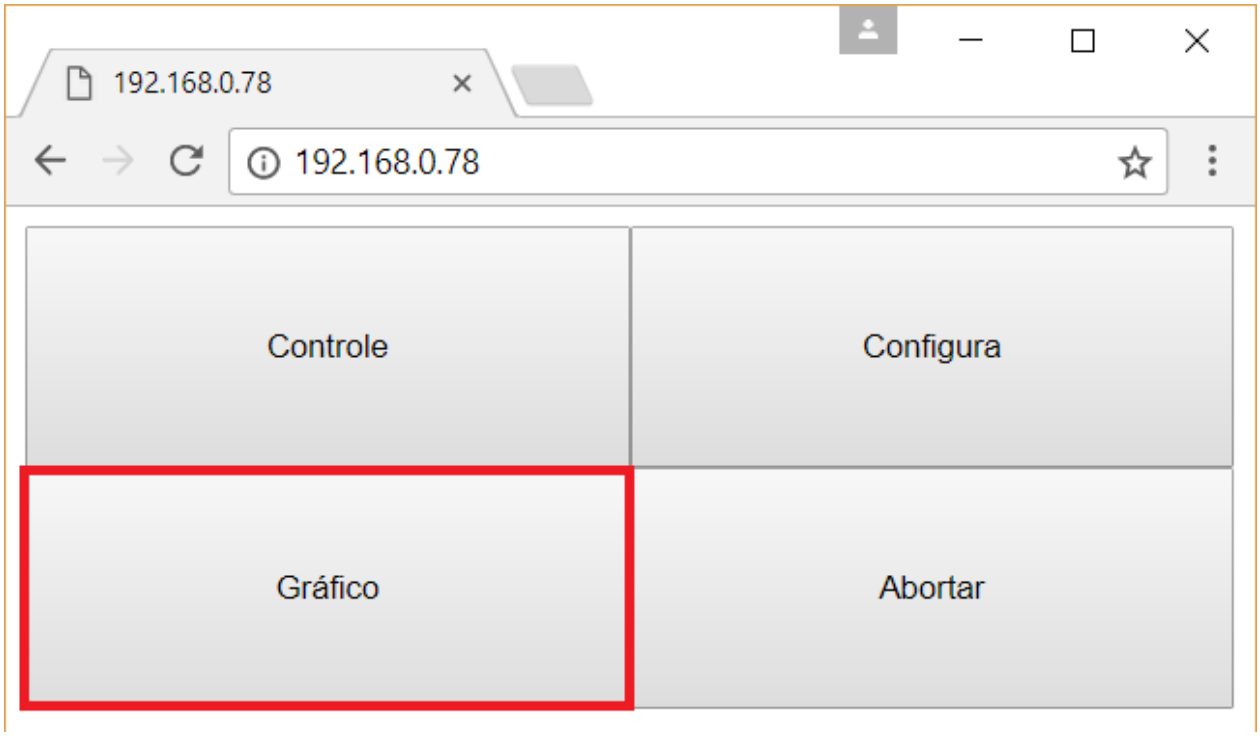


Figura 24 – Comando Gráfico

Fonte: Autor, 2017

Selecionada a opção ‘Gráfico’ é possível operar a ferramenta através dos botões ‘Aumentar (+)’, ‘Diminuir (-)’ e ‘Zero (0)’ alterando a velocidade desejada em passos, sendo o incremento de 10 pulsos/s.

Os métodos gráficos permitem ao usuário obter as informações do sistema necessárias para calcular, através dos métodos 1 e 2 de Ziegler-Nichols, os fatores Kp, Ki e Kd de um controlador PID.

3.5.3.1 Tela Gráfico

Então com a ferramenta gráfica disponível na tela gráfico, Figura 25, é possível obter informações do comportamento do sistema:

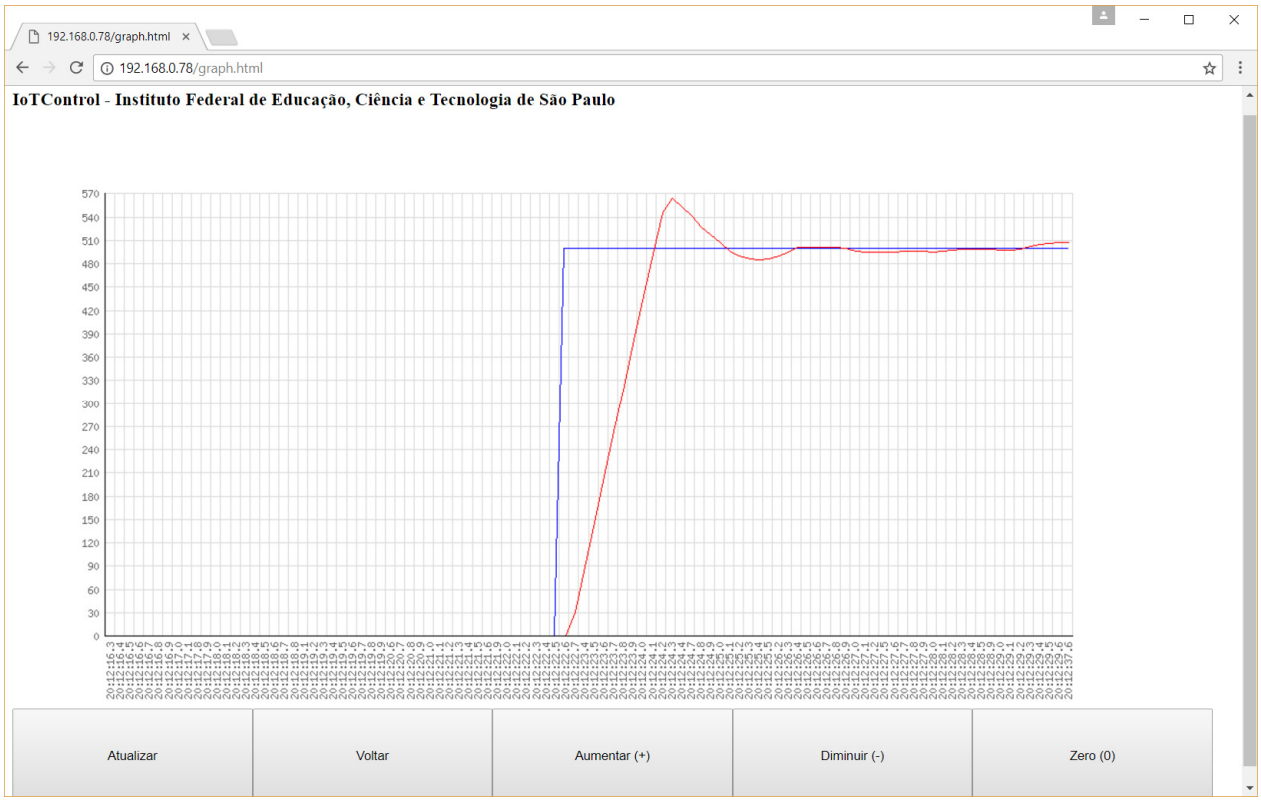


Figura 25 – Tela Gráfico

Fonte: Autor, 2017

Neste gráfico é possível notar o comportamento do sistema com sobressinal e estabilização quando o controlador PID foi ajustado com os fatores Kp, Ki e Kd obtidos através do método de Ziegler-Nichols.

Outro exemplo de tela gráfica, Figura 26, é apresentado abaixo:

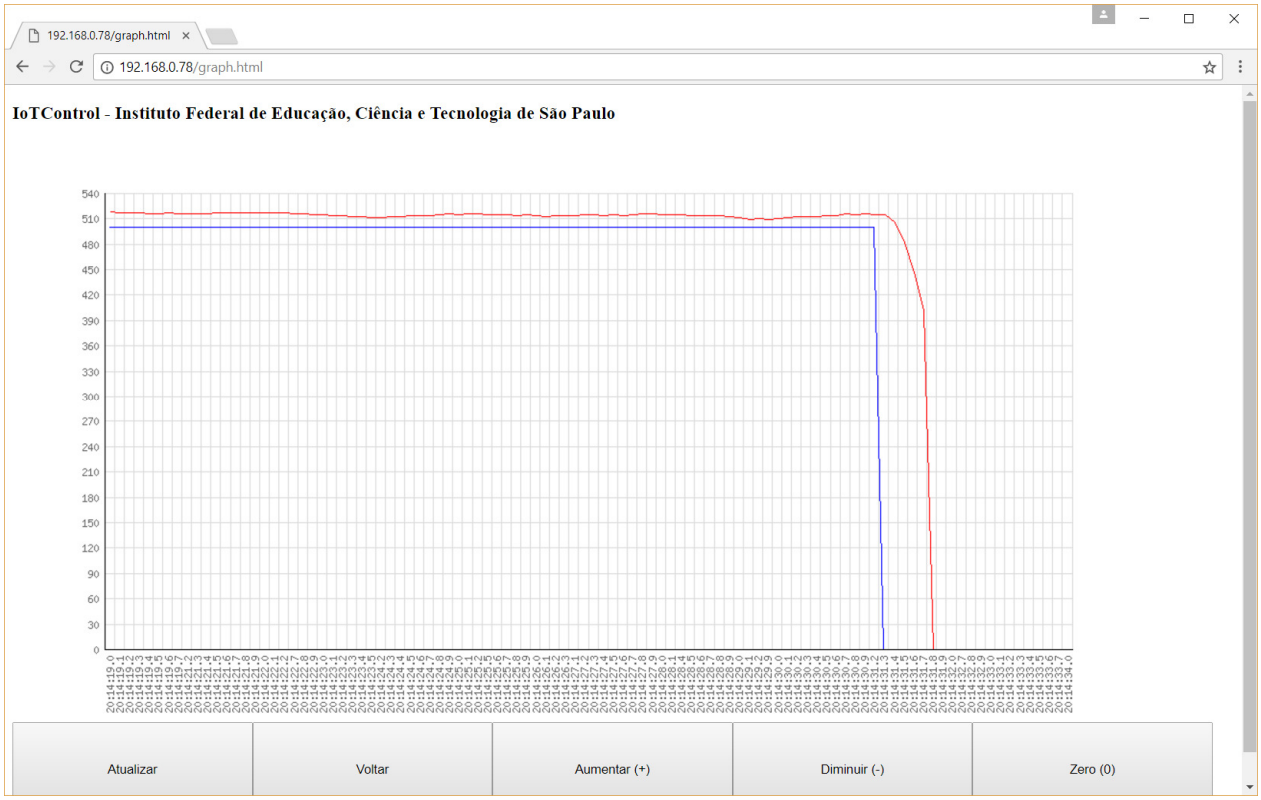


Figura 26 – Tela Gráfica de Parada

Fonte: Autor, 2017

É possível notar o atraso na reação do sistema, onde o sinal comandado realiza a borda de descida sendo o primeiro a atingir o valor nulo e em seguida é acompanhado pela descida no valor de velocidade obtida no sistema.

3.5.4 Abortar

A opção ‘Abortar’, Figura 27, dispara o encerramento do programa computacional aplicativo de controle, preparando o ambiente computacional para uma nova requisição.

Figura 27 – Abortar

Fonte: Autor, 2017

4. MATERIAIS

Para compor o sistema são necessários os seguintes componentes:

4.1 Dispositivos (Hardware)

Os componentes utilizados para construir este sistema são:

• Microcomputador de dados Marca Dell Modelo Inspiron 15R. Utilizado no desenvolvimento dos programas computacionais aplicativos, gerando códigos fontes e a documentação deste projeto.
• Raspberry Pi (dispositivo eletrônico computacional dotado de interface para sinais elétricos de entrada e saída de dados). É o dispositivo escolhido para executar o programa aplicativo, está embarcado junto a planta de controle.
• Roteador Wirelles (dispositivo capaz de realizar a comunicação de dados em rede). Este dispositivo realiza a comunicação de dados entre o dispositivo computacional embarcado e os usuários do sistema, permitindo uma conexão sem fios.
• Encoder (Sensor de rotação). Este sensor permite calcular a velocidade desenvolvida pelo motor elétrico e o sentido de rotação através de pulsos emitidos com o movimento do eixo do motor.
• Mini Motor elétrico (motor de escala miniatura de acionamento elétrico).
• Driver L293D (componente eletrônico). Este componente permite que sinais elétricos sejam convertidos em um sinal elétrico isolado que atua diretamente no motor, protegendo o circuito eletrônico do dispositivo computacional, chaveando para o motor elétrico a fonte de energia comandada.
• Fonte de alimentação (USB +5V). Componente responsável por suprir a energia elétrica para o dispositivo eletrônico computacional e para o Driver do motor elétrico.

Quadro 1 – Hardware

4.2 Programas Computacionais (Software)

Os programas computacionais do sistema são:

• Sistema operacional Windows. Utilizado como base no desenvolvimento, é o sistema instalado na plataforma de desenvolvimento dos códigos fontes.
• Sistema operacional Raspbian. É o sistema embarcado no dispositivo computacional acoplado a planta, é uma variação do sistema operacional Debian que é nativamente Linux.
• Programa computacional de controle IoTControl. Programa aplicativo desenvolvido em linguagem de programação Python, implementa os algoritmos de controle e acessa o banco de dados (manipulação e leitura).
• Programa computacional de rede IoTControl. Programa de comunicação desenvolvido em linguagens HTML e PHP, é capaz de prover as telas para os usuários e faz interface com o banco de dados.
• Banco de dados (c-treeACE). Programa que gerencia os dados, é um servidor de dados capaz de persistir os dados no sistema em memória não volátil, permite o acesso aos dados através de interfaces de programa, permitindo o acesso do programa aplicativo e do programa de rede.
• unixODBC. É um programa computacional que permite acesso através de uma interface ODBC ao banco de dados, é utilizado entre o interpretador PHP e o banco de dados ctree.
• Apache É o componente computacional que realiza a interface entre o navegador de internet e o programa computacional de rede embarcado.

Quadro 2 – Software

4.3 Elementos da Planta de Controle

Os elementos que compõem a planta de controle são o Motor, o Sensor e a Carga.

4.3.1 Motor

Dispositivo capaz de transformar energia elétrica em energia mecânica. Para este projeto adota o motor DC: 6V 210 RPM, conforme Figura 28:

Figura 28 – Motor

Fonte: Distribuidor, 2016

O motor é uma máquina composta por uma parte fixa (o estator) e outra móvel (o rotor). Parte estática da máquina é montada em volta do rotor, de forma que o mesmo possa girar internamente. Constituído de material ferromagnético, envolto em um enrolamento de baixa potência chamado de enrolamento de campo que tem a função apenas de produzir um campo magnético fixo para interagir com o campo da armadura.

Rotor é a parte móvel, montada sobre o eixo da máquina, construído de um material ferromagnético envolto em um enrolamento chamado de enrolamento de armadura e o anel comutador. Este enrolamento suporta uma alta corrente em comparação ao enrolamento de campo e é o circuito responsável por transportar a energia proveniente da fonte de energia.

Coletor, anel comutador realiza a inversão adequada do sentido das correntes que circulam no enrolamento de armadura, constituído de um anel de material condutor,

segmentado por um material isolante conecta cada uma das bobinas do enrolamento de armadura e as escovas no momento adequado. O anel é montado junto ao eixo da máquina e gira junto com a mesma. O movimento de rotação do eixo produz a comutação entre os circuitos dos enrolamentos.

Este conjunto é capaz de transformar energia elétrica de intensidade de corrente contínua em energia mecânica. A energia elétrica é fornecida aos condutores do enrolamento da armadura pela aplicação de uma tensão elétrica em seus terminais pelo anel comutador(coletor), fazendo com que se circule uma corrente elétrica nesse enrolamento que produz um campo magnético no enrolamento da armadura.

Como o corpo do estator é constituído de materiais ferromagnéticos, ao aplicar tensão nos terminais do enrolamento de campo da máquina temos uma intensificação do campos magnéticos no mesmo e, portanto, a produção de polos magnéticos espalhados por toda a extensão do estator.

Pela atuação do anel comutador que tem como função alternar o sentido de circulação da corrente no enrolamento da armadura, quando aplica-se uma tensão no comutador, com a máquina parada, a tensão é transferida ao enrolamento da armadura fazendo com que se circule uma corrente pelo mesmo o que produz um campo magnético e outros pares de polos no enrolamento da armadura.

A orientação desse campo, ou seja, a posição do polo norte e sul permanece fixa, simultaneamente temos uma tensão elétrica aplicada no enrolamento de campo no estator, assim, ao termos a interação entre os campos magnéticos da armadura no rotor e do campo no estator, os mesmos tentarão se alinhar, ou seja, o polo norte de um dos campos tentará se aproximar do polo sul do outro.

Como o eixo da máquina pode girar, caso os campos da armadura e do estator não estejam alinhados, surgirá um binário de forças que produzirá um torque no eixo, fazendo o mesmo girar. Ao girar, o eixo gira o anel comutador que é montado sobre o eixo, e ao girar o anel comutador muda o sentido de aplicação da tensão, o que faz com que a corrente circule no sentido contrário, mudando o sentido do campo magnético produzido. Assim, ao girar o anel comutador muda a posição dos polos magnéticos do campo da armadura e como o campo produzido pelo enrolamento de campo no estator

fica fixo, temos novamente a produção do binário de forças que mantém a mudança dos polos e consequentemente o movimento do eixo da máquina.

De um motor são exigidos, dinâmica, controle de rotação, torque constante e precisão de posição ou posicionamento. As características mais desejadas nos motores são o torque constante em faixa de rotação (até 210 rpm), faixa de controle da rotação e variação (até 1:210).

Dados Técnicos:

- Tensão: 6V DC
- Velocidade: 210 RPM
- Diâmetro do eixo: 4mm
- Comprimento do eixo: 12mm
- Máxima eficiência: 2.0kg.cm/170rpm/2.0W/0.60A
- Máximo torque: 5.2kg.cm/110rpm/3.1W/1.10A
- Redução: 1:34 (1:34.02)
- Comprimento total motor e eixo: 73mm
- Diâmetro do motor: 25mm
- Pinos: 1 - M1 Motor (-), 6 - M1 motor (+)

4.3.2 Encoder

Dispositivo composto de dois sensores de efeito Hall, dispostos em 90º, possibilita a emissão de pulsos através da alteração do potencial elétrico provocado pelo movimento de imãs acoplados ao eixo do motor, a disposição mecânica destes dois sensores possibilita identificar o sentido de movimento. Este Encoder gera 341 pulsos por volta, conforme Figura 29:

Figura 29 – Encoder

Fonte: Distribuidor, 2017

Dados Técnicos:

- Tensão: 3,3V DC
- Redução: 1:34 (1:34.02)
- Resolução do Encoder: 11 x Redução 34.02 = 341.2PPR

Conexões:

- 2 - GND Encoder
- 3 - Encoder A phase
- 4 - Encoder B phase
- 5 - 3.3V Encoder

4.3.3 Carga

Acoplada ao eixo do motor a fim de avaliar o comportamento do sistema simulando uma aplicação real, teremos uma roda com diâmetro de 62mm, a Figura 30 ilustra o modelo aplicado:

Figura 30 – Roda

Fonte: Distribuidor, 2017

Como resultado a 210 rpm, ou 3.5 voltas por segundo, a velocidade calculada será conforme equação:

$$Vmax = \pi d * \frac{voltas}{s} \quad (10)$$

Assim, $Vmax = 3{,}1415 * 0{,}062 * 3{,}5 = 0{,}6850 \frac{m}{s}$.

Resultando em 41 metros por minuto ou 2,4km/h.

5. MÉTODOS E RESULTADOS

A plataforma é composta da planta, dispositivo computacional e interface humano máquina, conforme Figura 31:

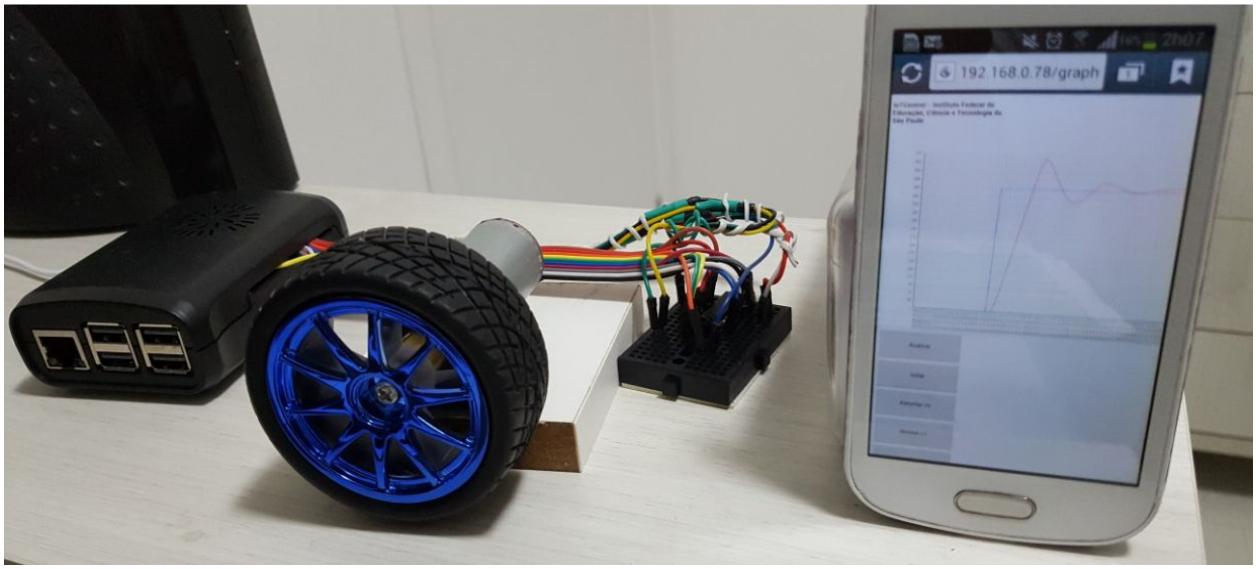


Figura 31 – Plataforma

Fonte: Autor, 2017

A metodologia aplicada ao trabalho é composta das fases de pesquisa, experimentos, desenvolvimento, teste. Durante a pesquisa foi possível encontrar tecnologias que possibilitariam alcançar os objetivos propostos, algumas das tecnologias não foram utilizadas devido aos primeiros experimentos comprovarem inadequação ou dificuldades de implementação. Neste capítulo são apresentados os métodos utilizados e a base de conhecimento com os resultados obtidos.

5.1 Pesquisa Científica e Pesquisa Aplicada

Análise de artigos e trabalhos relacionados com o contexto e ou similaridade parcial com o trabalho desenvolvido, entre os trabalhos analisados foi possível extrair as referências necessárias que servem de base para este trabalho, consideradas as áreas de conhecimento em Engenharia Elétrica para pesquisa em controle e na área de Ciência da Computação para a pesquisa em programas computacionais.

A pesquisa aplicada é a etapa de busca por tecnologias aplicáveis ao trabalho, esta fase foi realizada com interação entre os experimentos práticos e aplicação de tecnologias disponíveis. Esta etapa permitiu a evolução do trabalho, resultando na aplicação de métodos e tecnologias computacionais, para solucionar a comunicação dos dados entre os dispositivos.

5.2 Base Teórica

Selecionadas as referências deste trabalho foi consolidada a base teórica de conhecimento que sustenta o plano de desenvolvimento desta plataforma. É possível descrever que a base deste trabalho é demonstrar que algoritmos de controle podem ser executados em um ambiente computacional de rede baseado em internet. Os pilares de conhecimento são as literaturas de controle onde encontram-se as referências para implementar os algoritmos e as literaturas de computação que disponibilizam a base de conhecimento para implementar os programas computacionais que permitem o funcionamento desta plataforma em rede. Os métodos empregados no desenvolvimento foram obtidos com base nas literaturas referenciadas.

5.2.1 Planta de Controle

Para demonstrar o funcionamento do sistema foi aplicado em uma planta de controle dotada de um motor elétrico com aplicação de carga, de forma a proporcionar a análise do sistema na prática. Foi possível fechar a malha de controle utilizando o sensor contador de pulsos. Com este sinal através de sua derivada no tempo resultará na velocidade do motor. Utilizados como dados de entrada no sistema a seleção de controle de velocidade e o sinal de velocidade realimentado.

Os algoritmos foram implementados conforme exposto no capítulo 3.1 utilizando como base as publicações referenciadas, (LATHI, B. P. 1998), (OGATA, 1998) e (MITRA, 2006).

5.2.1.1 Malha Aberta

Possibilitando a operação do sistema mesmo com ausência ou falha do sensor de rotação foi implementado o controle em malha aberta, Figura 32, onde a atuação é calculada através da conversão direta da entrada do sistema, multiplicada por um fator, em saída. Representa-se o sistema com o seguinte diagrama de blocos:

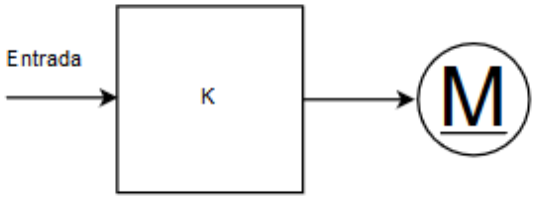


Figura 32 – Malha Aberta

Fonte: Autor, 2017

Onde K representa um fator proporcional à entrada e que é aplicado à planta.

5.2.1.2 Malha Fechada

Operação do sistema com realimentação através do sensor de rotação, onde a atuação é calculada através de algoritmos pré-selecionados e calibrados de acordo com as definições inseridas pelo usuário. Representa-se o sistema em malha fechada com o diagrama de blocos na Figura 33:

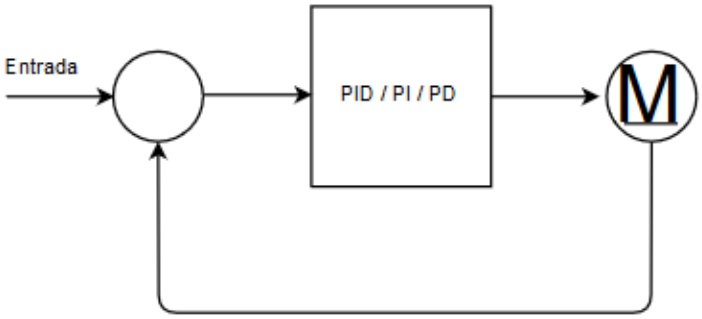


Figura 33 – Malha Fechada

Fonte: Autor, 2017

Onde PID e PI representam os algoritmos que podem ser selecionados pelo usuário.

5.2.1.3 Entradas

Como entradas do sistema teremos a seleção de velocidade selecionada através do programa computacional aplicativo e o sinal do sensor de rotação (encoder). Este faz parte da entrada do sistema somente quando em malha fechada e é composto por dois sinais elétricos, Encoder A e B respectivamente conectados ao dispositivo computacional nas entradas de pino número 16 e 20. A entrada de velocidade selecionada pelo usuário ou velocidade desejada é computada pelo sistema através de um parâmetro armazenado no banco de dados, este valor pode ser alterado pelo usuário através da interface gráfica, foi considerada a unidade pulsos por segundo (p/s) que também é adotada para o sinal realimentado (velocidade obtida).

A velocidade obtida é calculada utilizando as entradas 16 e 20, correspondem aos sensores: Encoder A e B. O algoritmo executa a contagem de cada transição lógica em borda de subida. Isto é possível porque o dispositivo computacional alimenta o encoder com um potencial elétrico de +3,3V, o encoder altera o valor de suas saídas A e B conforme o movimento do eixo. No dispositivo computacional está configurado para executar uma interrupção a cada detecção de borda de subida nas entradas 16 e 20, assim quando ocorrer uma borda de subida no sensor A e o sinal lógico da entrada do sensor B for positivo significa que o motor está com movimento no sentido horário.

5.2.1.4 Saídas

As saídas do sistema são: o sinal elétrico modulado por largura de pulso, pino 23 do dispositivo computacional, aplicado à eletrônica acoplada que permitirá a energização do atuador (motor) e os sinais digitais que definem o sentido de rotação do motor, pinos 24 e 25 do dispositivo computacional. Foi definida a saída do sistema como o conjunto entre o acionamento do programa computacional aplicativo aliado ao circuito elétrico que transforma o sinal digital recebido em tensão elétrica.

Este circuito elétrico possui interface com os seguintes sinais provenientes do programa computacional aplicativo:

Pino 1	Pino 2	Pino 7	Resultado
Positivo	Negativo	Positivo	Giro horário
Positivo	Positivo	Negativo	Giro anti-horario
Positivo	Negativo	Negativo	Parado
Positivo	Positivo	Positivo	Parado
Negativo	N/A	N/A	Parado

Quadro 3 – Conexão elétrica e sentido de giro

Fonte: Autor, 2017

A seleção de sentido é realizada pelo sinal de velocidade desejada, entrada do sistema, uma vez positivo este sinal representa um sentido de giro horário, caso este sinal for negativo o sentido de giro é anti-horário. Estes sinais elétricos digitais são aplicados no circuito elétrico representado pelo diagrama de blocos da Figura 34:

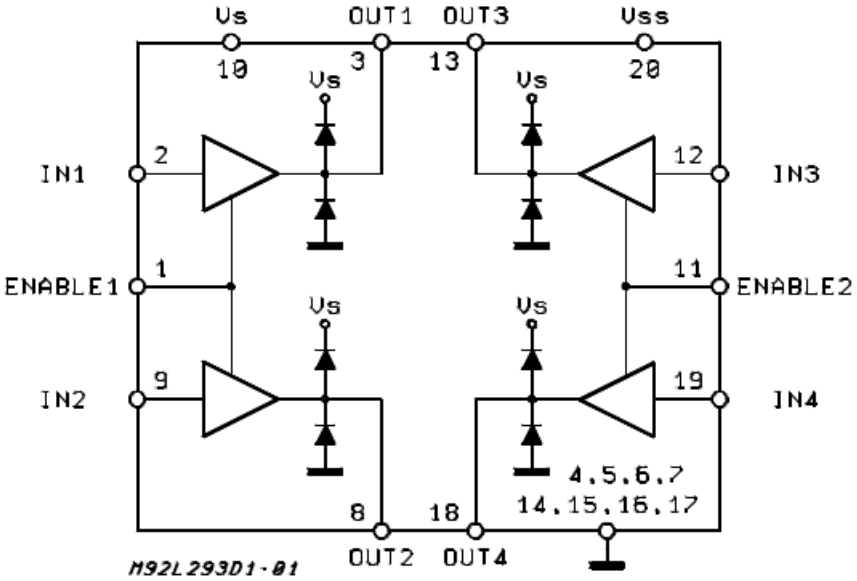


Figura 34 – Circuito acionamento motor

Fonte: ST, L293D Datasheet.

O componente eletrônico utilizado possui nome comercial de mercado L293D, com a seguinte estrutura representada pela Figura 35:

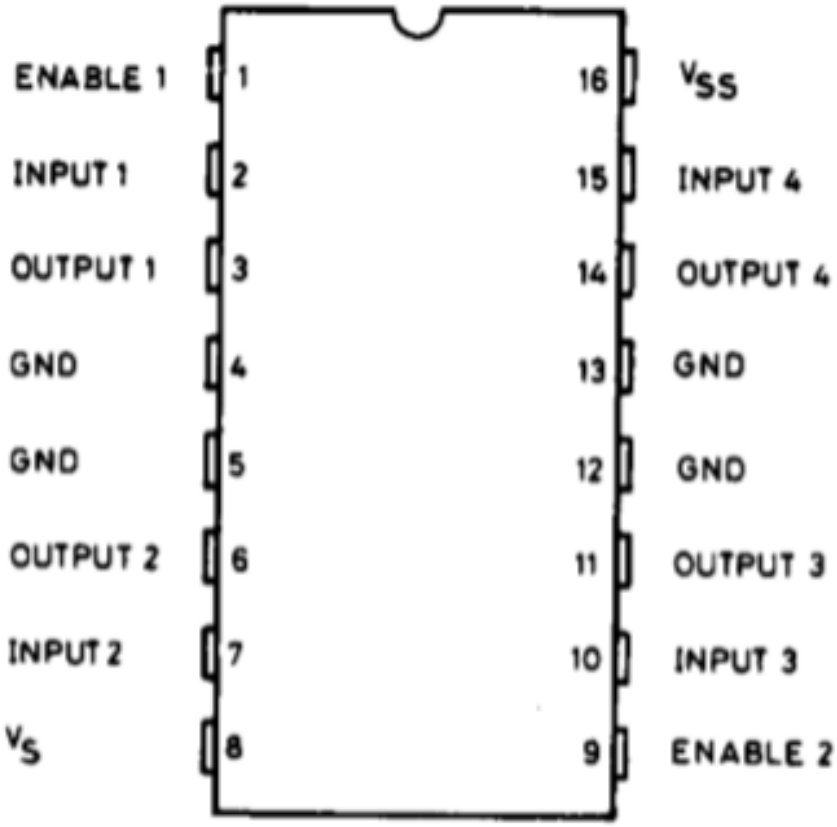


Figura 35 – Componente eletrônico

Fonte: ST, L293D DataSheet.

O circuito aplicado neste desenvolvimento está presente no “Apêndice B – Circuito Elétrico”, as conexões elétricas estão descritas no “Apêndice C – Tabela de Conexões Elétricas”.

5.3 Experimento Prático

Etapa de avaliação experimental de tecnologias e métodos, realização de experimentos aplicando controladores e tecnologias. Está descrito nas seções seguintes os métodos utilizados após a construção da plataforma, com os programas computacionais implementados. Figura 36, circuito elétrico aplicado:

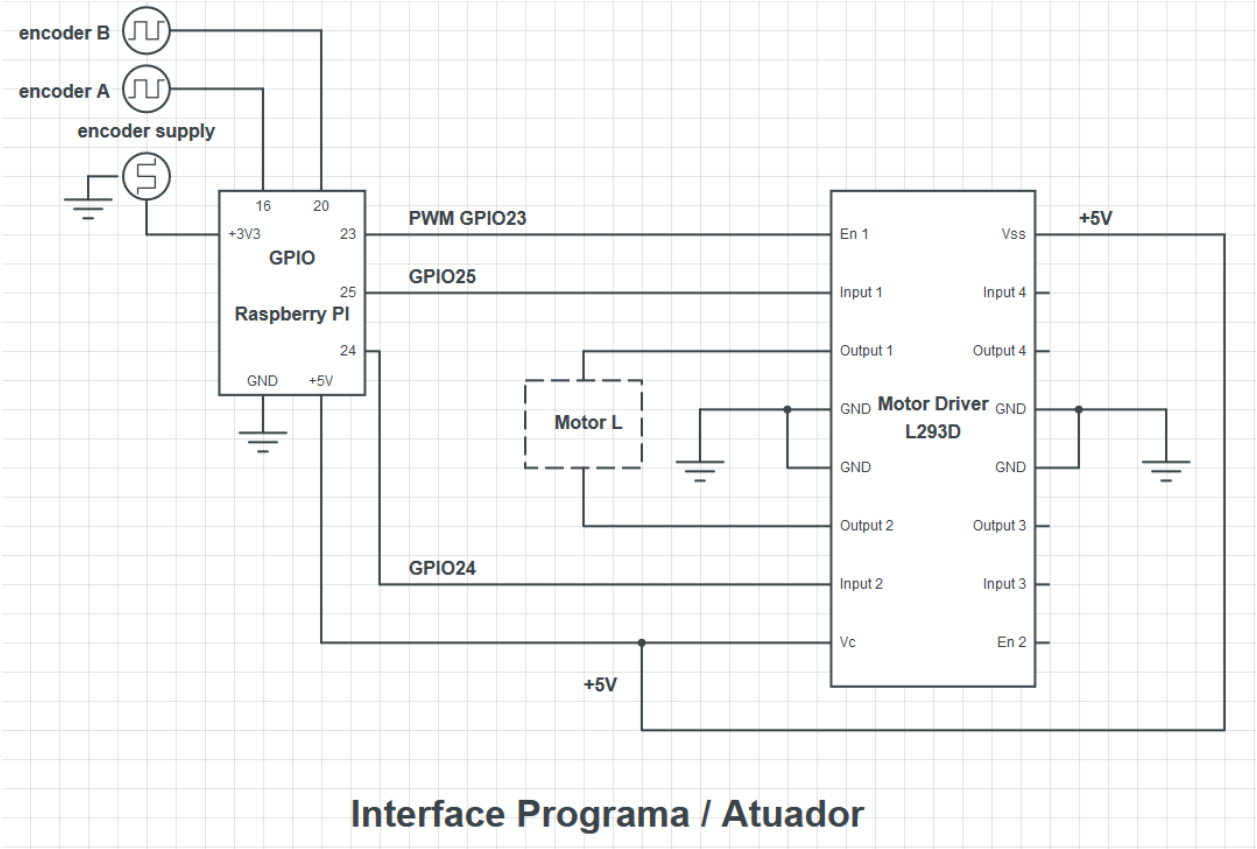


Figura 36 – Circuito Elétrico

Fonte: Autor, 2016

No circuito temos o sinal PWM GPIO23 que é a modulação de controle que habilita o componente L293D a liberar a tensão elétrica ao motor. Os sinais GPIO25 e GPIO24 determinam o sentido de movimento, invertendo a polaridade do sinal elétrico de corrente contínua que é aplicado ao motor. O componente é alimentado pela fonte de +5V. O movimento do eixo do motor é então transformado em pulsos elétricos pelos sensores de rotação, respectivamente (encoder A e B) conectados em GPIO16 e GPIO20. Estes sensores são alimentados pela fonte de +3,3V.

5.3.1 Ensaio em Malha Aberta

Com a finalidade de determinar a relação do sinal aplicado à planta e o sinal de resposta obtido, foi realizado o teste em malha aberta verificando o sinal medido pela plataforma através da ferramenta gráfica. Foi simulado o sinal de saída com as seguintes porcentagens de largura de pulso 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 e 100.

Foram obtidos os seguintes resultados:

PWM (%)	Pulsos/s
0	0
10	0
20	150
30	290
40	390
50	455
60	510
70	545
80	580
90	600
100	630

Quadro 4 – Ensaio Malha Aberta

Neste ensaio foi possível identificar que o motor possui uma zona morta que corresponde ao valor de 20% do sinal PWM, ou seja, valores inferiores não produzem movimento no motor.

Os dados obtidos pela ferramenta foram comparados com os sinais de encoder A e B, mensurados com um osciloscópio. Disponível no “Apêndice D – Sinais Encoder no Osciloscópio” deste documento.

Com os resultados obtidos foi possível traçar a seguinte curva que referencia o sinal de velocidade obtido em pulsos/s e o sinal aplicado em PWM %, exposto na Figura 37.

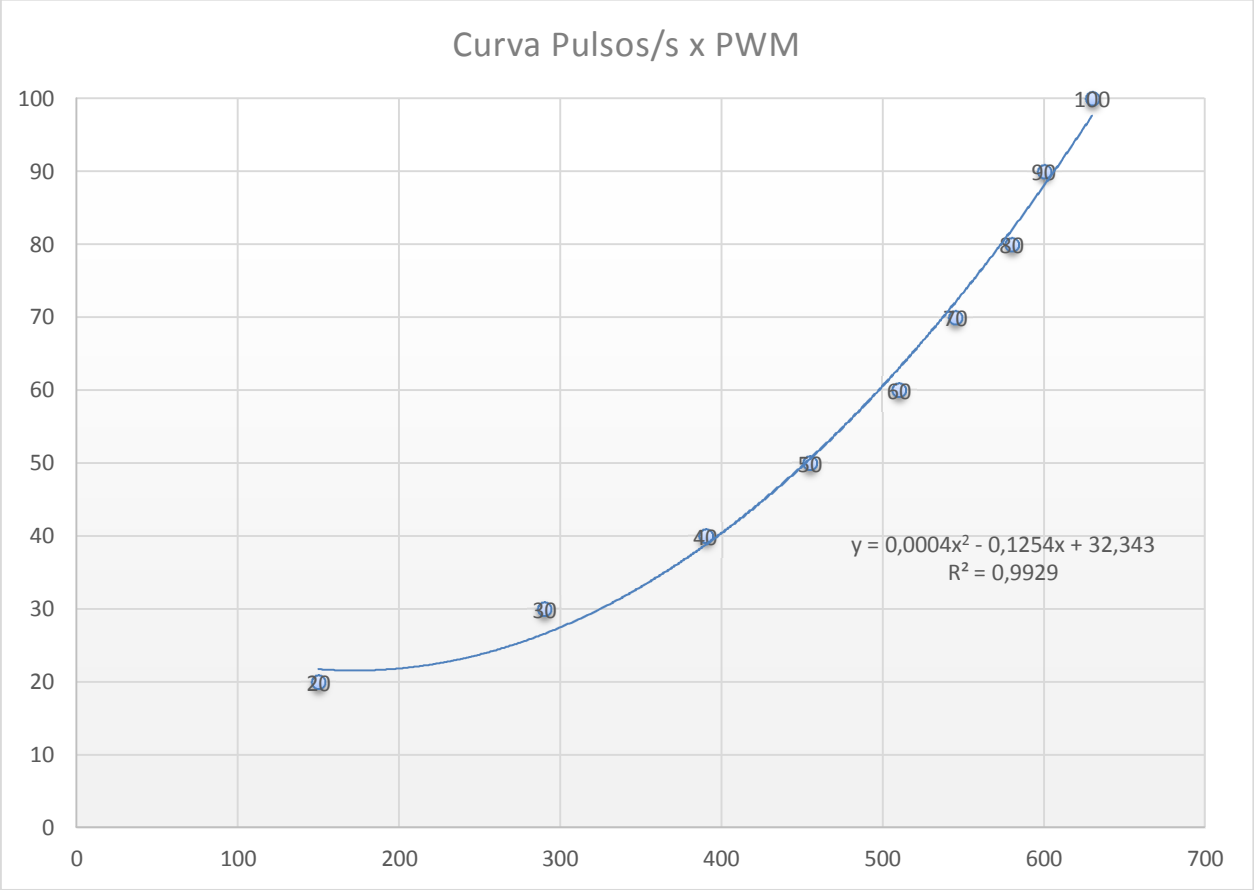


Figura 37 – Curva motor.

Fonte: Autor, 2017

Deste gráfico foi traçada uma linha de tendência estabelecendo uma equação polinomial de segunda ordem:

$$y = 0{,}0004x^2 - 0{,}1254x + 32{,}343 \qquad (11)$$

Esta equação representa a razão PWM de forma eficiente, temos o valor de R quadrado em 0,99.

Outra possibilidade para esta aplicação seria utilizar uma equação linear, representada pela Figura 38:

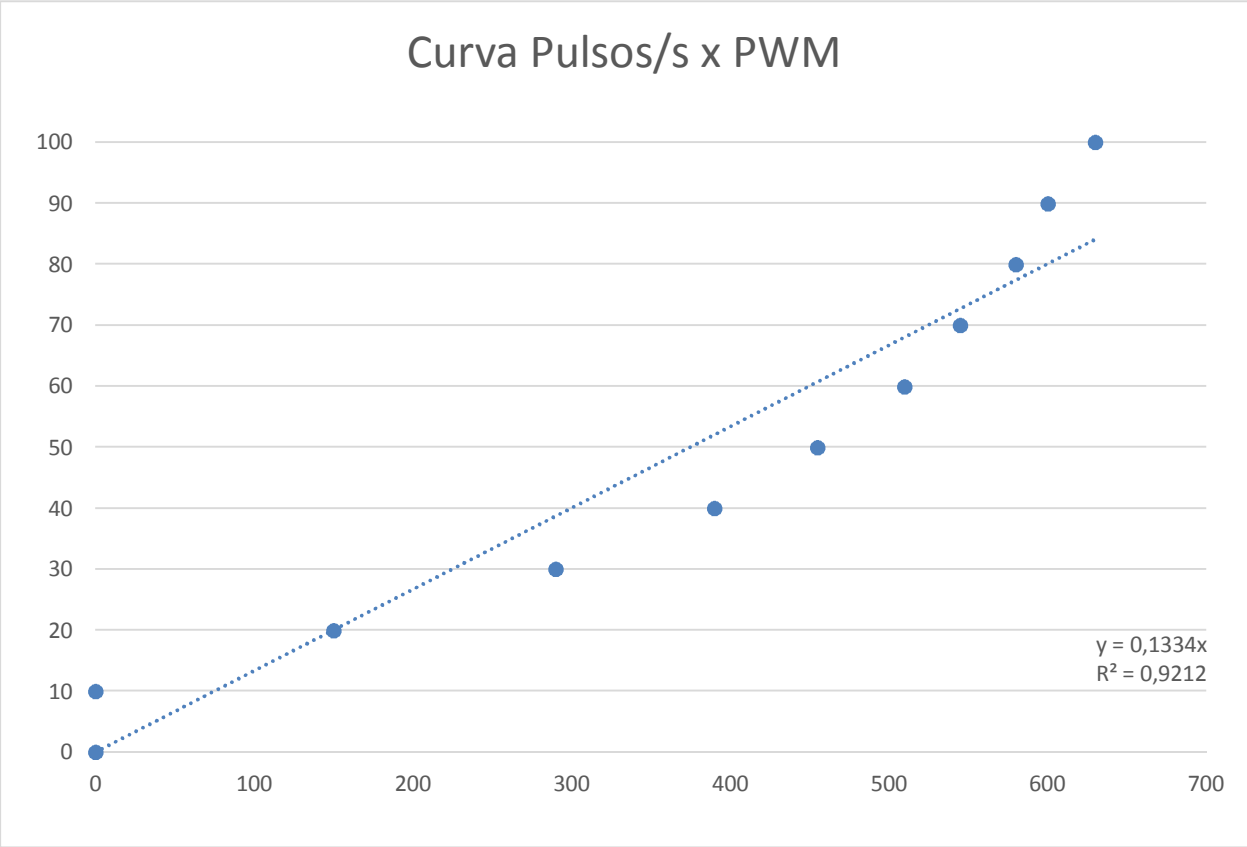


Figura 38 – Curva motor.

Fonte: Autor, 2017

Deste gráfico foi traçada uma linha de tendência estabelecendo uma equação linear:

$$y = 0{,}1334x \qquad (12)$$

Esta equação representa a razão PWM de forma eficiente, temos o valor de R quadrado em 0,92 porém a fim de minimizar erro para o modelo implementado utiliza-se a equação de segundo grau, embora não seja um requisito para este sistema de controle. Foram adotados os fatores A, B e C obtidos na Figura 37.

Sinal aplicado na saída com o valor de 100%, PWM máximo, velocidade obtida 630 pulsos/s, resultado apresentado na Figura 39:

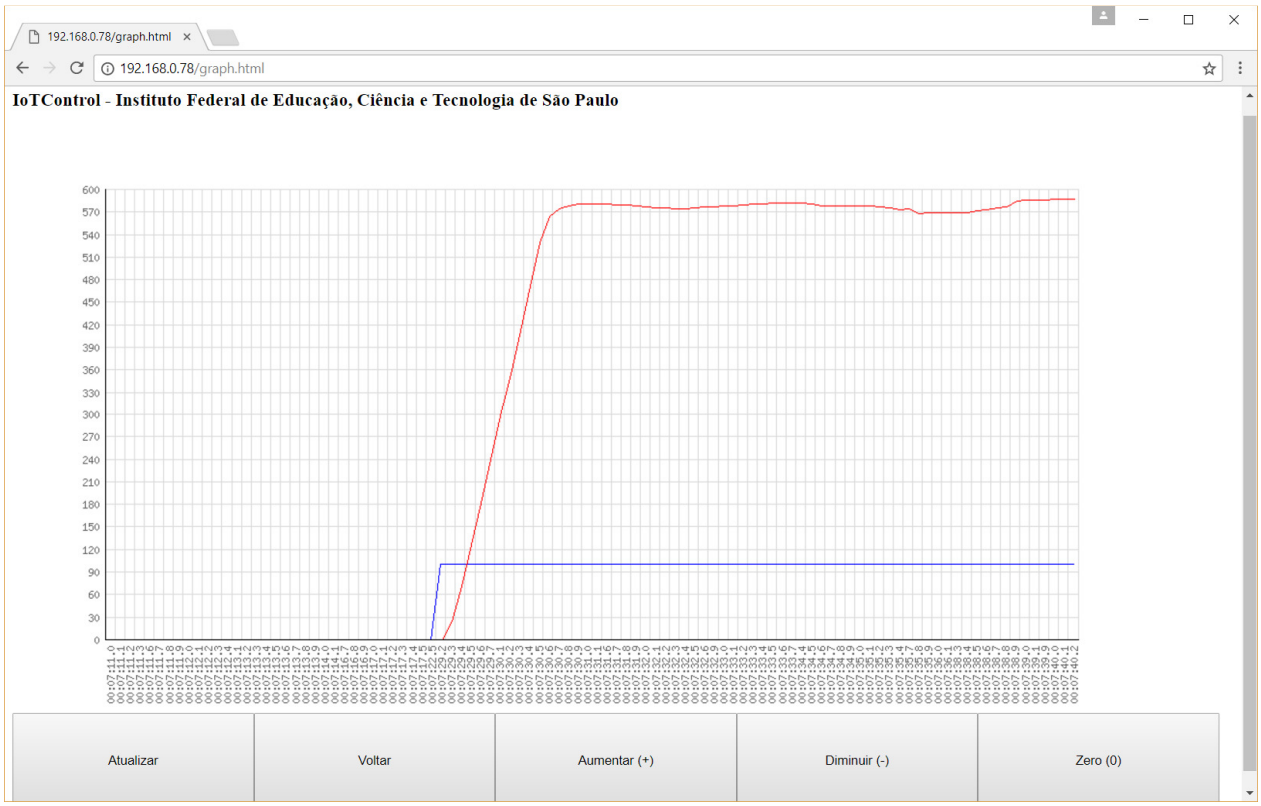


Figura 39 – Gráfico Malha Aberta largura de pulso 100%

Fonte: Autor, 2017

Este valor máximo de PWM corresponde a máxima tensão elétrica aplicada nos terminais do motor, 5 V DC. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”.

Os dados foram obtidos com o sistema em malha aberta.

Sinal aplicado na saída com o valor de 70% PWM, velocidade obtida 545 pulsos/s, resultado apresentado na Figura 40:

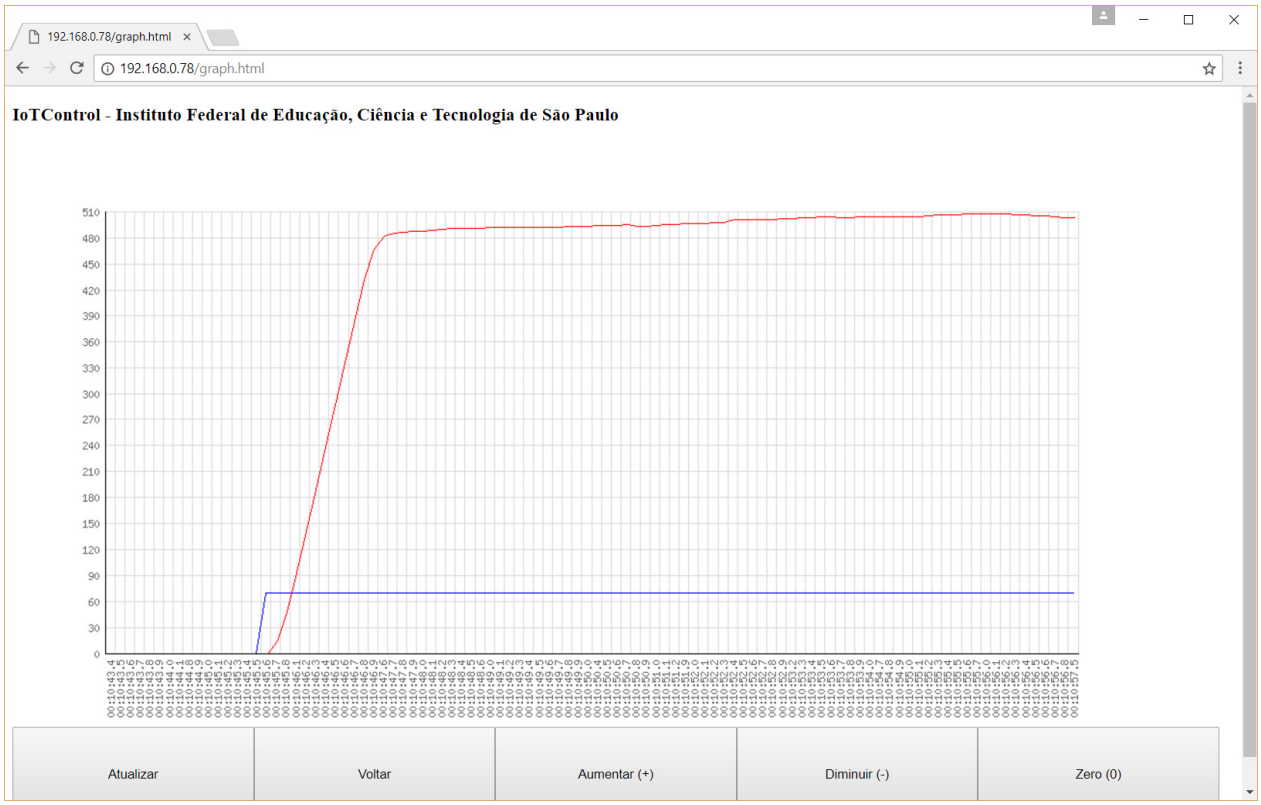


Figura 40 – Gráfico Malha Aberta largura de pulso 70%

Fonte: Autor, 2017

Com base nos dados destes gráficos é possível observar que a variação no resultado obtido foi de 14% embora a variação no valor aplicado foi de 30%. O valor de 70% PWM corresponde ao valor de 3.5V DC, tensão elétrica aplicada nos terminais do motor. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”.

Estes dados foram obtidos com o sistema em malha aberta.

Sinal aplicado na saída com o valor de 50% PWM, velocidade obtida 455 pulsos/s, resultado apresentado na Figura 41:

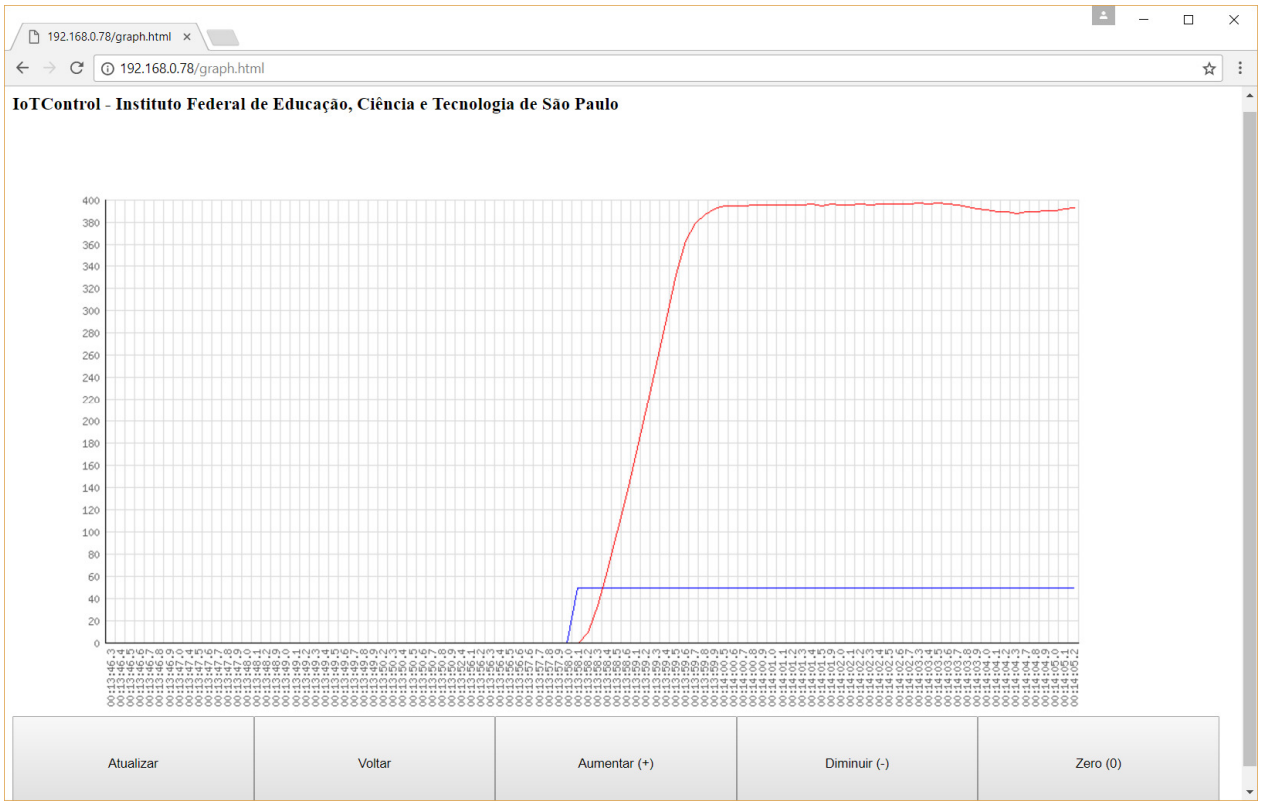


Figura 41 – Gráfico Malha Aberta largura de pulso 50%

Fonte: Autor, 2017

É possível observar que nesta faixa a oscilação no resultado é maior do que obtido anteriormente com sinais aplicados acima de 70%. Nota-se que em torno desta faixa a variação no sinal obtido deve ser considerada. O valor de 50% PWM corresponde ao valor de 2.5V DC, tensão elétrica aplicada nos terminais do motor. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”.

Os dados foram obtidos com o sistema em malha aberta.

Sinal aplicado na saída com o valor de 40% PWM, velocidade obtida 390 pulsos/s, resultado apresentado na Figura 42:

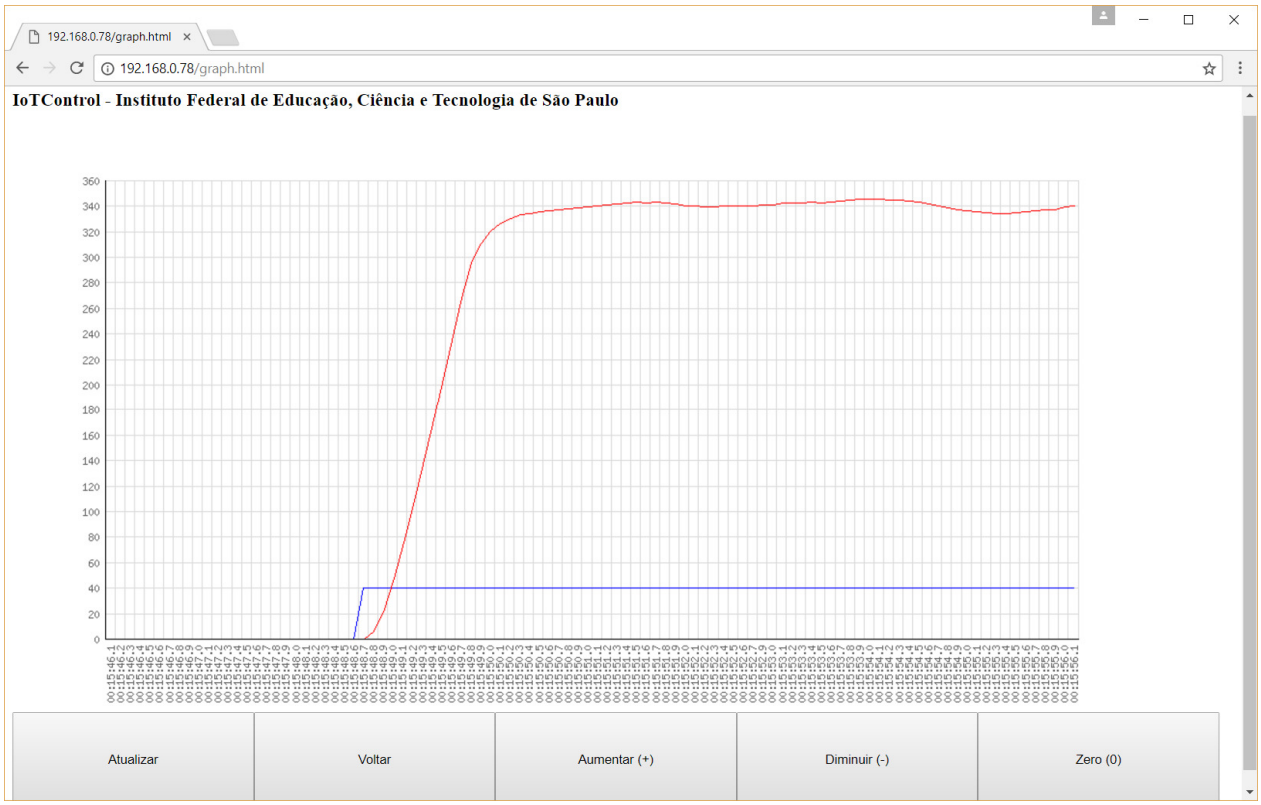


Figura 42 – Gráfico Malha Aberta largura de pulso 40%

Fonte: Autor, 2017

Com variação menor comparado ao gráfico anterior é possível considerar que a faixa a ser observada de forma especial está entre o sinal aplicado de 40% e 75%. O valor de 40% PWM corresponde ao valor de 2V DC, tensão elétrica aplicada nos terminais do motor. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”.

Estes dados foram obtidos com o sistema em malha aberta.

Sinal aplicado na saída com o valor de 20% PWM, velocidade obtida 150 pulsos/s, resultado apresentado na Figura 43:

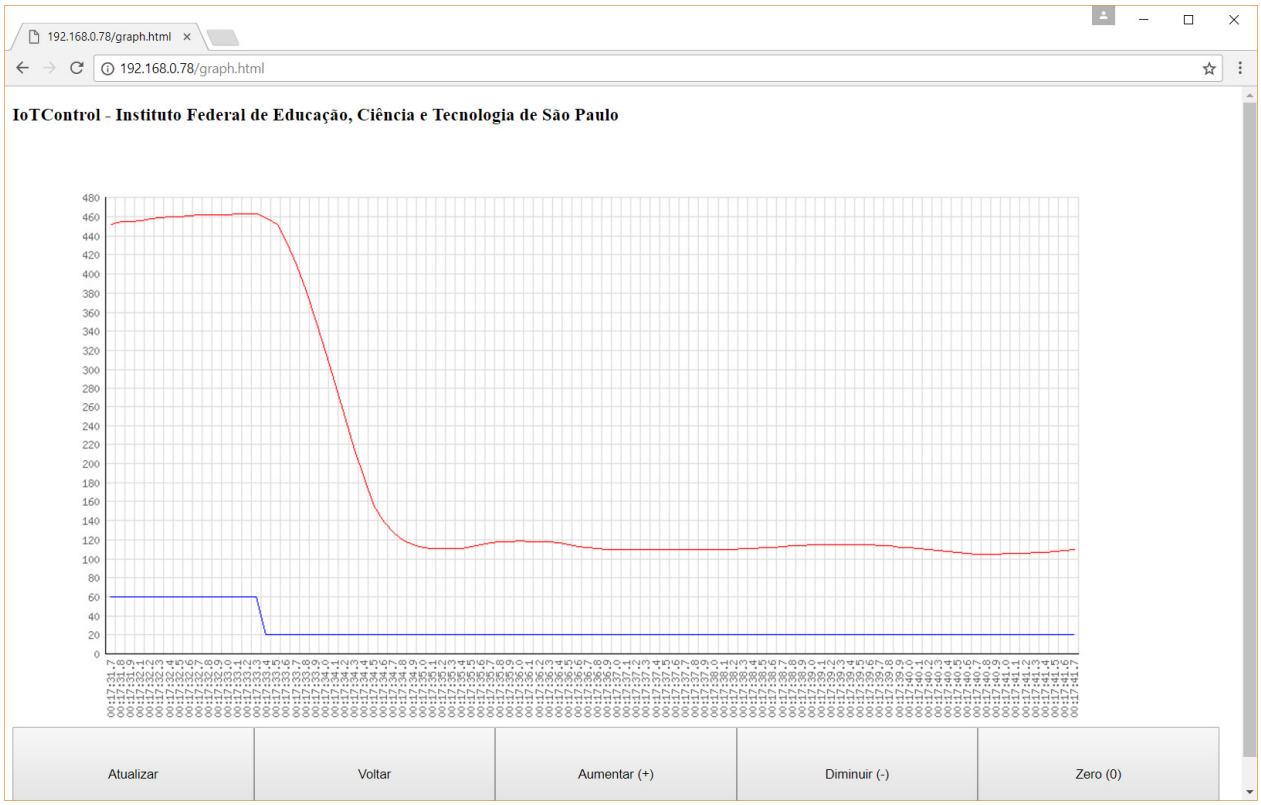


Figura 43 – Gráfico Malha Aberta largura de pulso 20%

Fonte: Autor, 2017

Comportamento estável com mínima variação no resultado obtido, sinal comandado de 60 para 20% valor de PWM. Este valor corresponde a 1 V DC, tensão elétrica aplicada nos terminais do motor. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”.

Os dados foram obtidos com o sistema em malha aberta.

Sinal aplicado na saída com o valor de 10% PWM, velocidade obtida 0 pulsos/s, resultado apresentado na Figura 44:

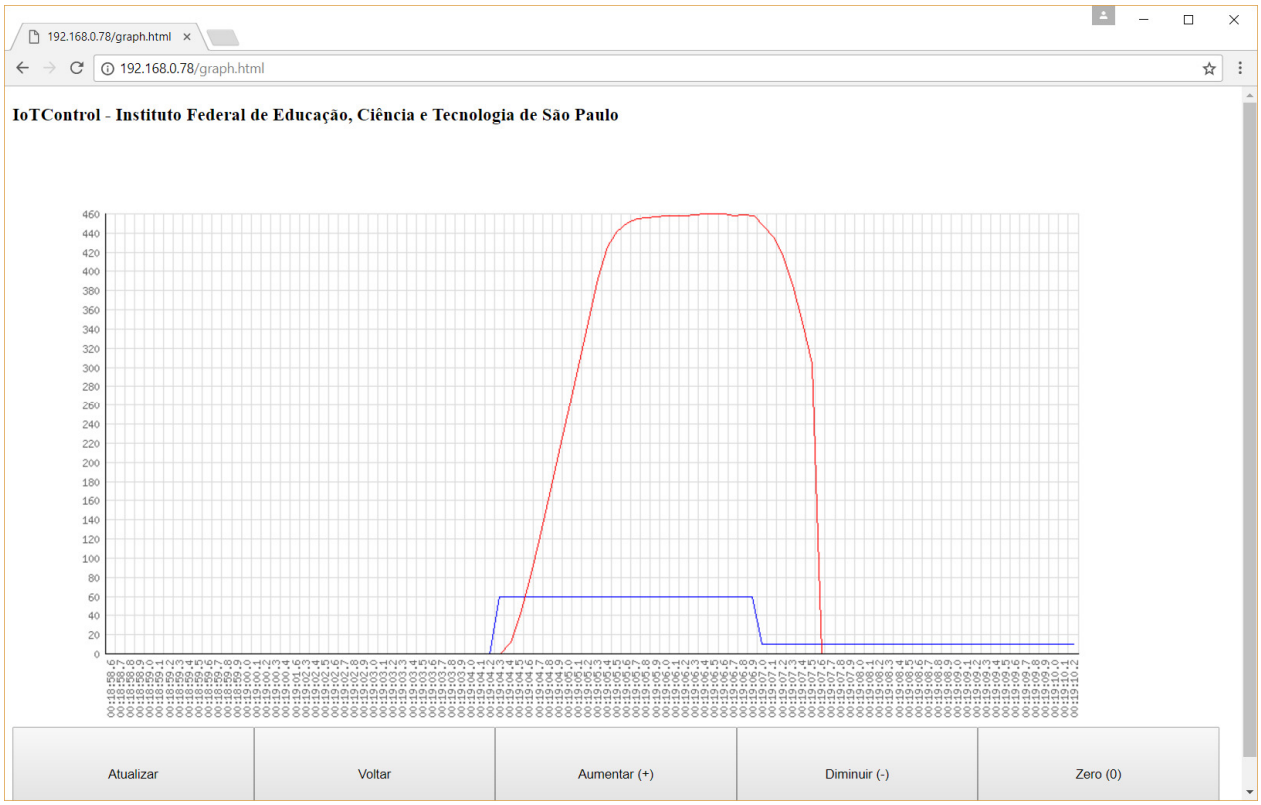


Figura 44 – Gráfico Malha Aberta largura de pulso 10%

Fonte: Autor, 2017

Neste gráfico foi exibido trecho do gráfico anterior para demonstrar que a velocidade era maior que zero antes de aplicar 10% no sinal de saída. Assim foi possível comprovar que existe uma zona morta, a ser considerado que valores menores do que 20% no sinal de saída não gerarão nenhum resultado. É denominada esta faixa de zona morta.

O valor de 10% PWM corresponde a 0.5V DC, tensão elétrica aplicada nos terminais do motor. O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”, dados obtidos com o sistema em malha aberta.

Foi utilizado o instrumento osciloscópio para verificar os sinais elétricos medidos pela ferramenta, os dados obtidos são apresentados em formato de telas no “Apêndice D – Sinais Encoder no Osciloscópio”.

5.3.2 Ziegler-Nichols Método Resposta ao Degrau (Malha Aberta)

Considere os valores obtidos em malha aberta para ser possível calcularmos os valores para malha fechada Kp, Ki e Kd, segundo o primeiro método de Ziegler-Nichols. Note o gráfico, resultado apresentado na Figura 45:

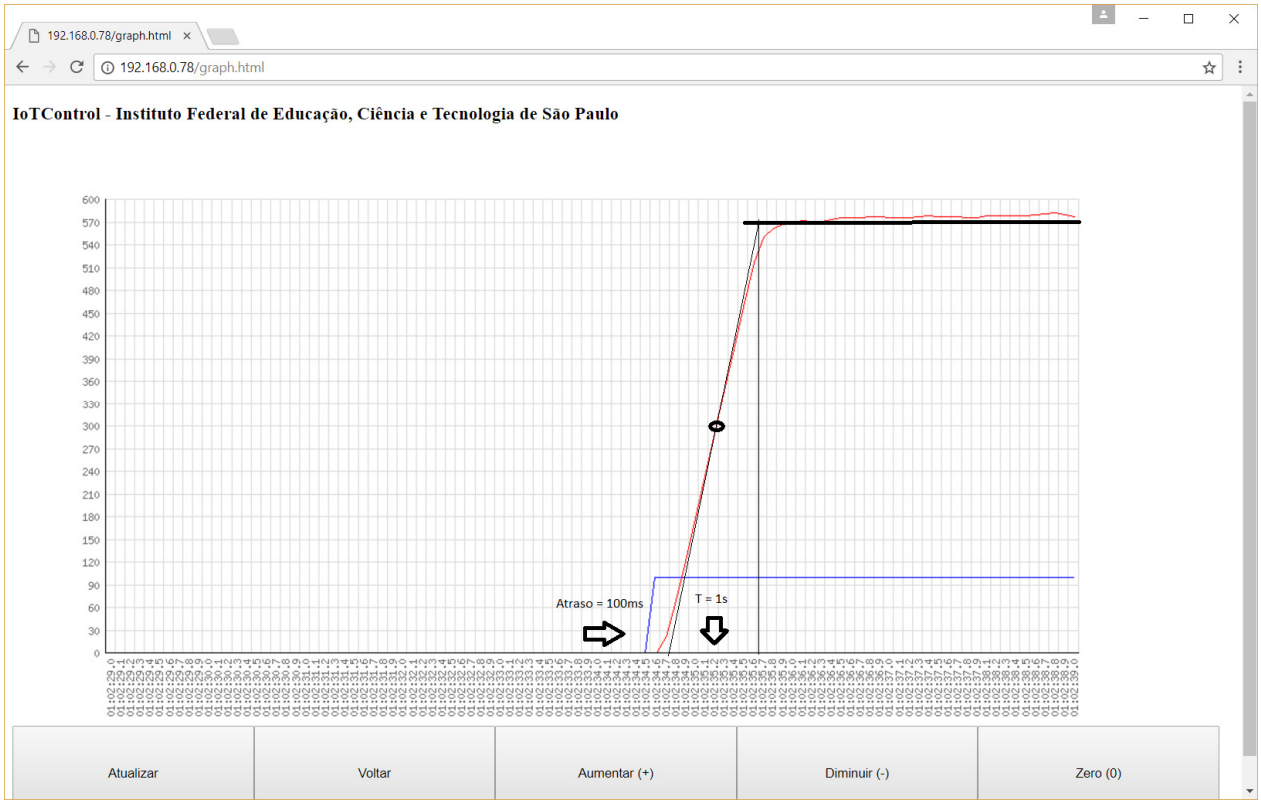


Figura 45 – Ziegler-Nichols Método Resposta ao Degrau

Fonte: Autor, 2017

O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”, dados obtidos com o sistema em malha aberta.

O gráfico fornece Atraso, L, 100ms e Período, T = 1s, o ganho da planta é obtido com valor de velocidade sobre o valor aplicado, Kp = 6,3 utilizando a tabela teremos:

	K	Ki	Kd
P	$\frac{\frac{1}{Kp}T}{L}$	-	-
PI	$\frac{\frac{0,9}{Kp}T}{L}$	$3,33T$	-
PID	$\frac{\frac{1,2}{Kp}T}{L}$	$2T$	$0,5T$

Quadro 5 – Ziegler-Nichols Resposta Degrau

K = 1,9

Ki = 0,2

Kd = 0,05

5.3.3 Ziegler-Nichols Método Estabilidade Marginal (Malha Fechada)

Em malha fechada foram ajustados os fatores Ki e Kd em zero, anulando a atuação destes fatores, foi incrementado o fator K até que comece a ocorrer uma oscilação permanente no valor obtido. Foi anotado este valor e o período do sinal oscilante, respectivamente Kc e Tcrit. Note o gráfico, resultado apresentado na Figura 46:

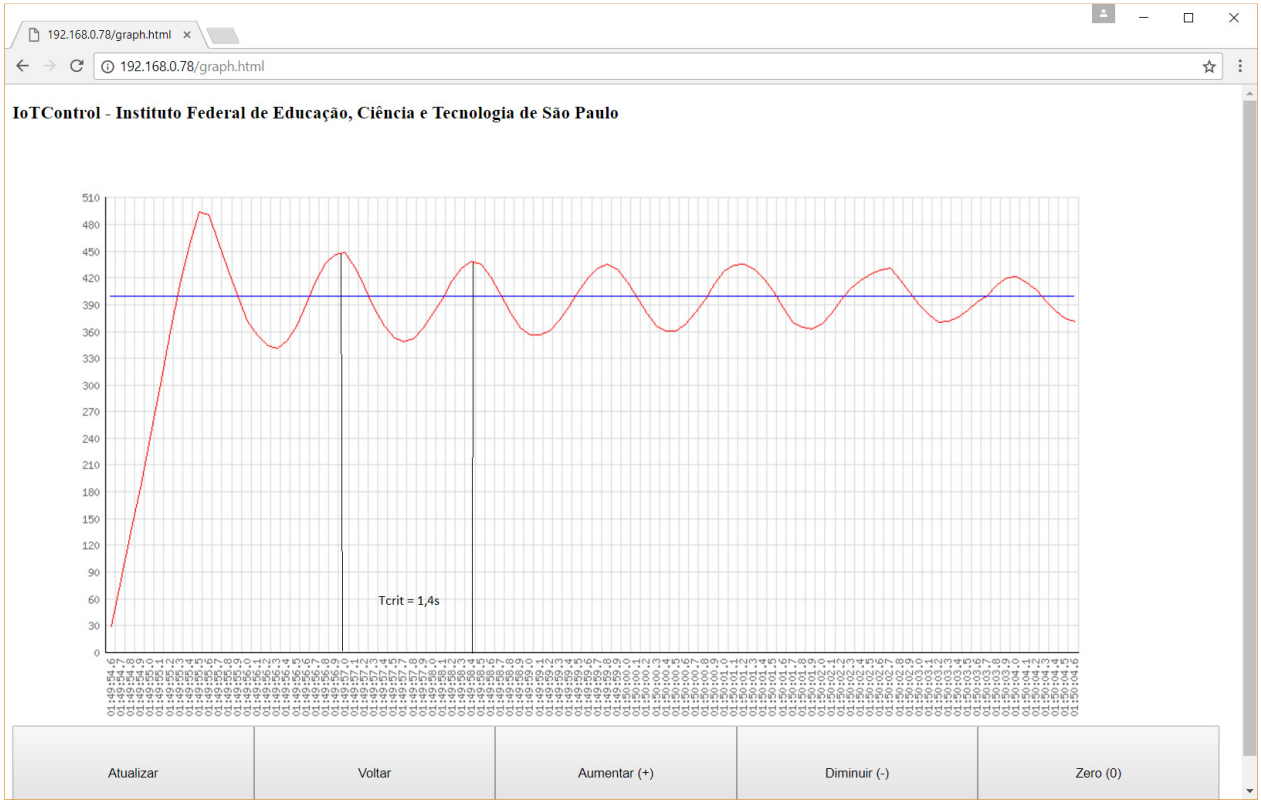


Figura 46 – Ziegler-Nichols Estabilidade Marginal

Fonte: Autor, 2017

O gráfico foi obtido utilizando a ferramenta na opção de tela “Gráfico”, dados obtidos com o sistema em malha fechada.

O gráfico fornece Período, Tcrit = 1,4s, o ganho Kcrit = 4 utilizando a tabela teremos:

	K	Ki	Kd
P	$0{,}5\ Kcrit$	-	-
PI	$0{,}45\ Kcrit$	$0{,}85\ Tcrit$	-
PID	$0{,}6\ Kcrit$	$0{,}5\ Tcrit$	$0{,}12\ Tcrit$

Quadro 6 – Ziegler-Nichols Estabilidade Marginal

K = 2,4

Ki = 0,7

Kd = 0,168

Estes fatores foram testados e comparados com os valores obtidos, aplicando o método de resposta ao degrau. Os métodos gráficos que permitiram extrair os números para calcular estes fatores foram executados com a própria ferramenta. Comprovando a utilidade e fácil operação desta plataforma.

5.3.4 Ensaio em Malha Fechada

Aplicando as entradas de velocidade de 0 a 450 pulsos/s utilizando os fatores K, Ki e Kd obtidos com o método de resposta ao degrau proposto por Ziegler-Nichols, respectivamente K = 1.9, Ki = 0.2 e Kd = 0.05 a Figura 47 ilustra a tela desta configuração:

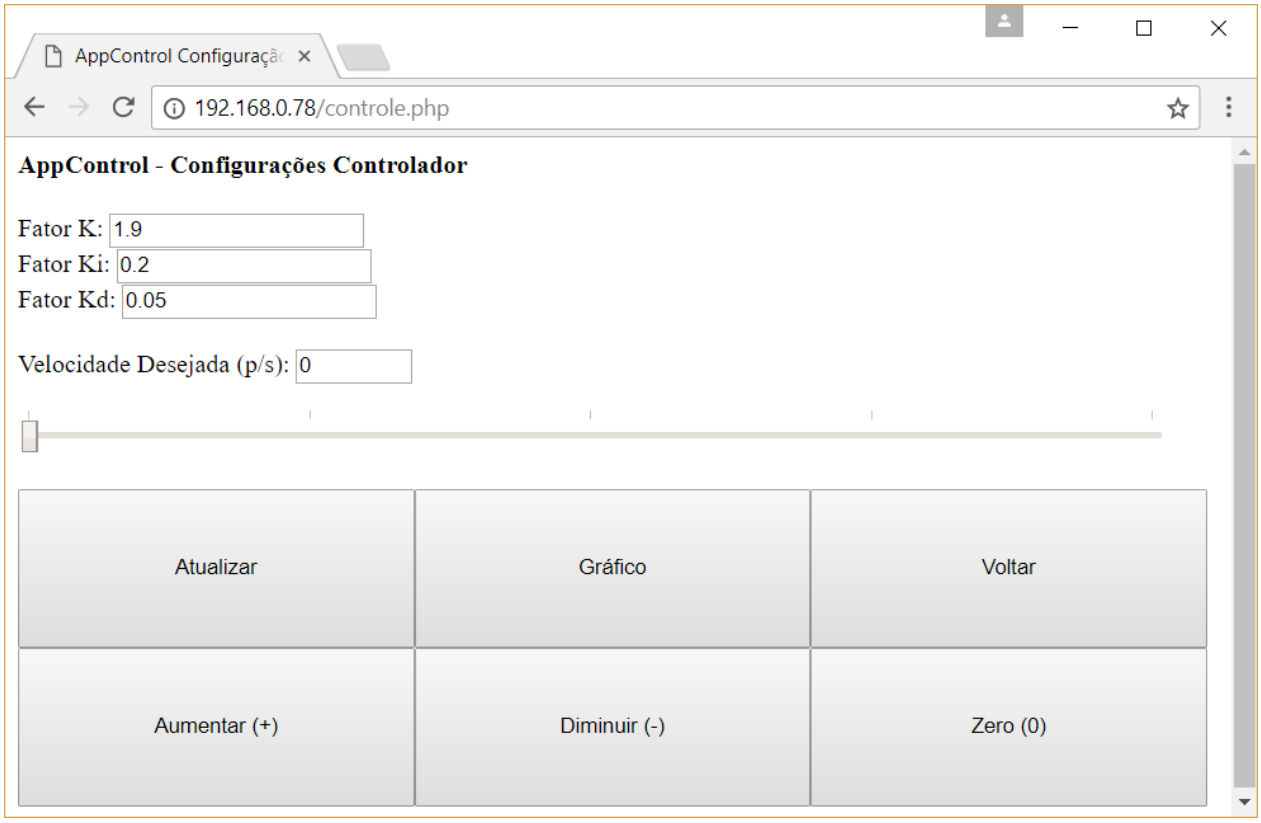


Figura 47 – Controlador PID configurado, K = 1.9, Ki = 0.2 e Kd 0.05

Fonte: Autor, 2017

O resultado é apresentado no gráfico, Figura 48:

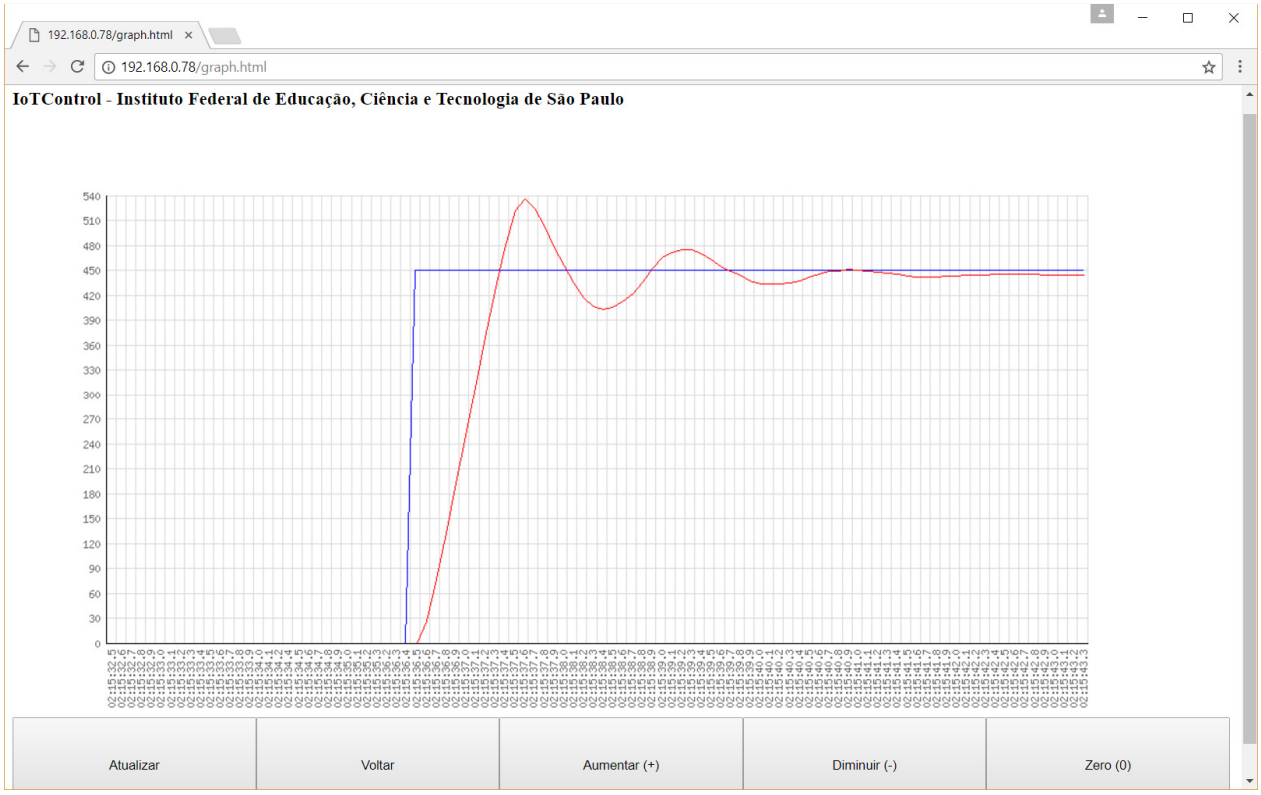


Figura 48 – Resultado PID configurado, K = 1.9, Ki = 0.2 e Kd = 0.05

Fonte: Autor, 2017

O mesmo procedimento selecionando a entrada de velocidade de 0 a 450 pulsos/s foi executado porém utilizando os fatores K, Ki e Kd obtidos com o método estabilidade marginal de Ziegler-Nichols, respectivamente K = 2.4, Ki = 0.7, e Kd = 0.168 apresentado na Figura 49:

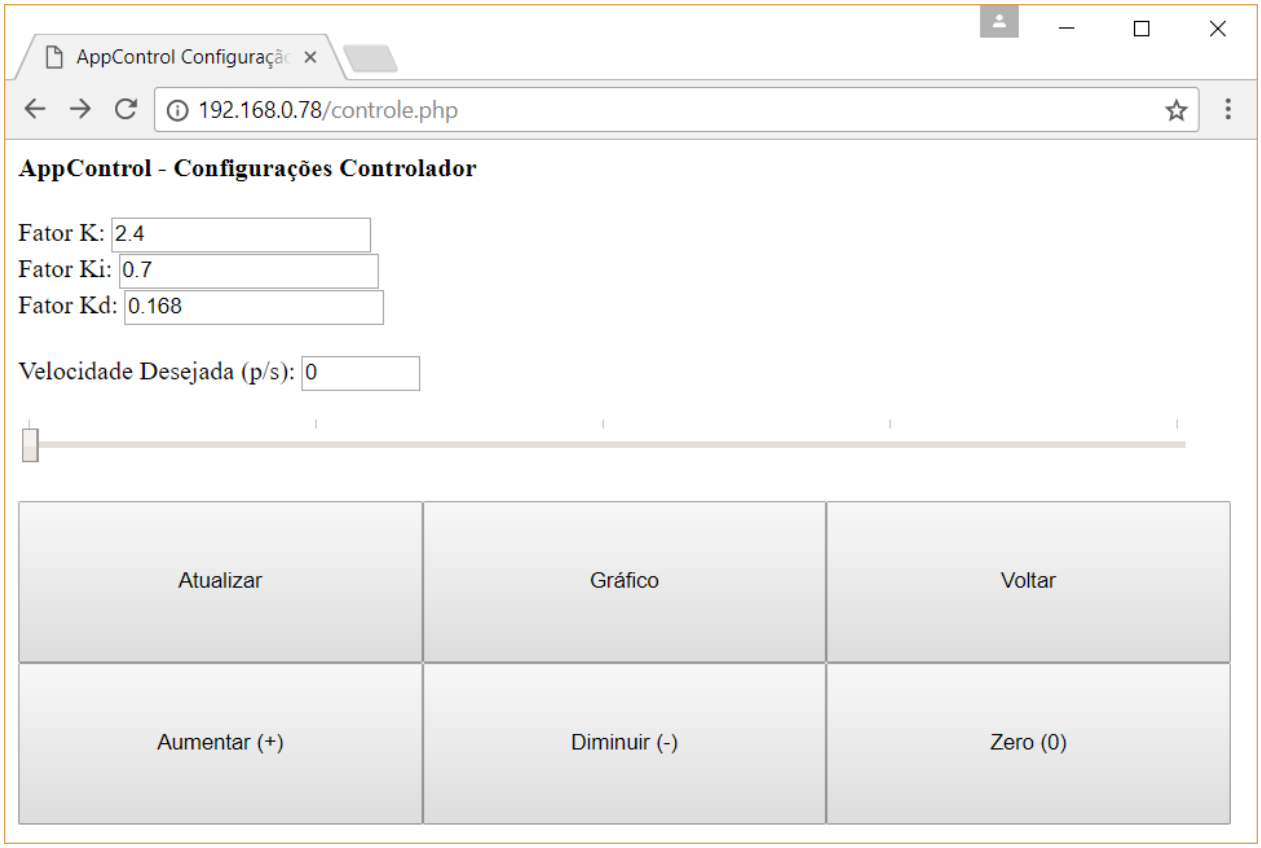


Figura 49 – Controlador PID configurado, K = 2.4, Ki = 0.7 e Kd = 0.168

Fonte: Autor, 2017

O resultado é apresentado no gráfico, Figura 50:

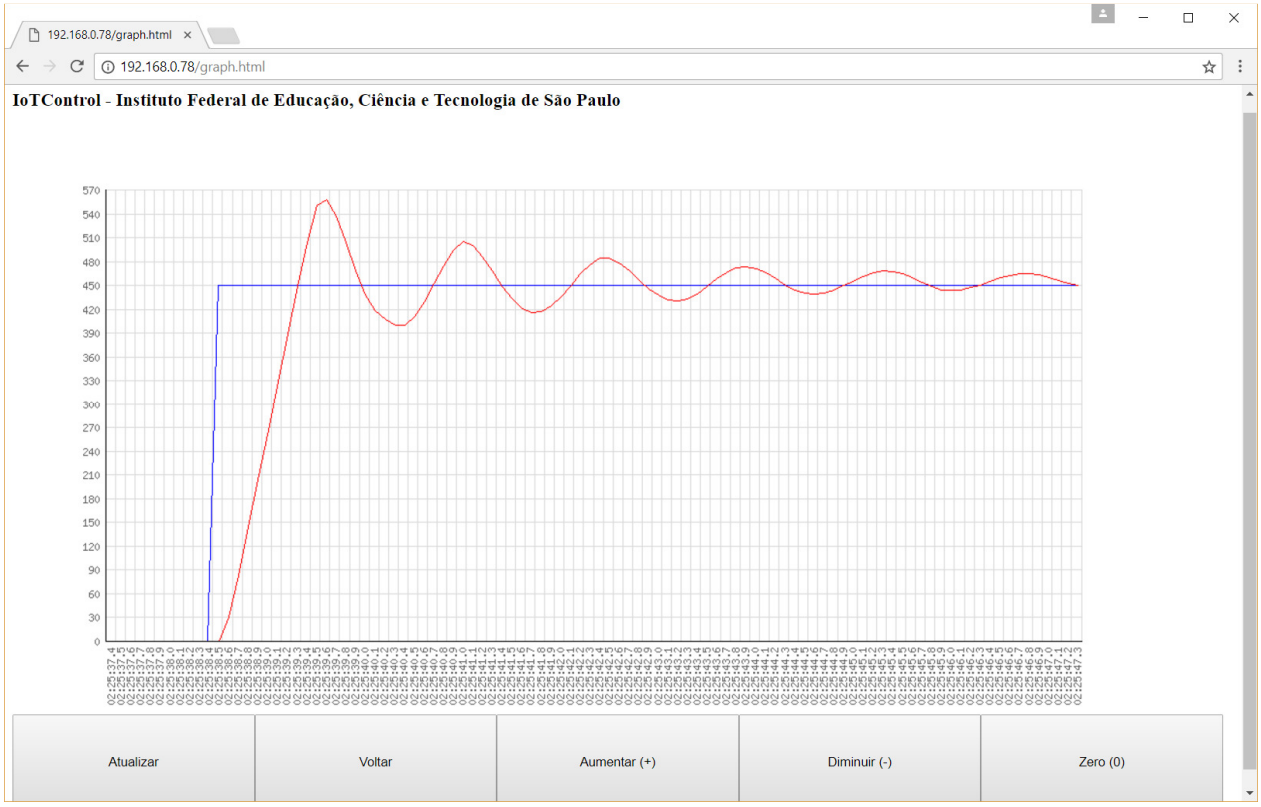


Figura 50 – Resultado PID configurado, K = 2.4, Ki = 0.7 e Kd = 0.168

Fonte: Autor, 2017

5.4 Analítica

A ferramenta possibilita, com base nos resultados gráficos obtidos, realizar a análise do controlador aplicado. Permitindo ao usuário realizar conclusões sobre o comportamento do sistema e controlador. A ferramenta possibilita a configuração de velocidades negativas, ou seja, no sentido anti-horário, o seguinte gráfico apresenta a inversão de sentido partindo de 500 para -500 pulsos/s. Os fatores utilizados foram K = 1.9, Ki = 0.2 e Kd = 0.05 resultados apresentados na Figura 51:

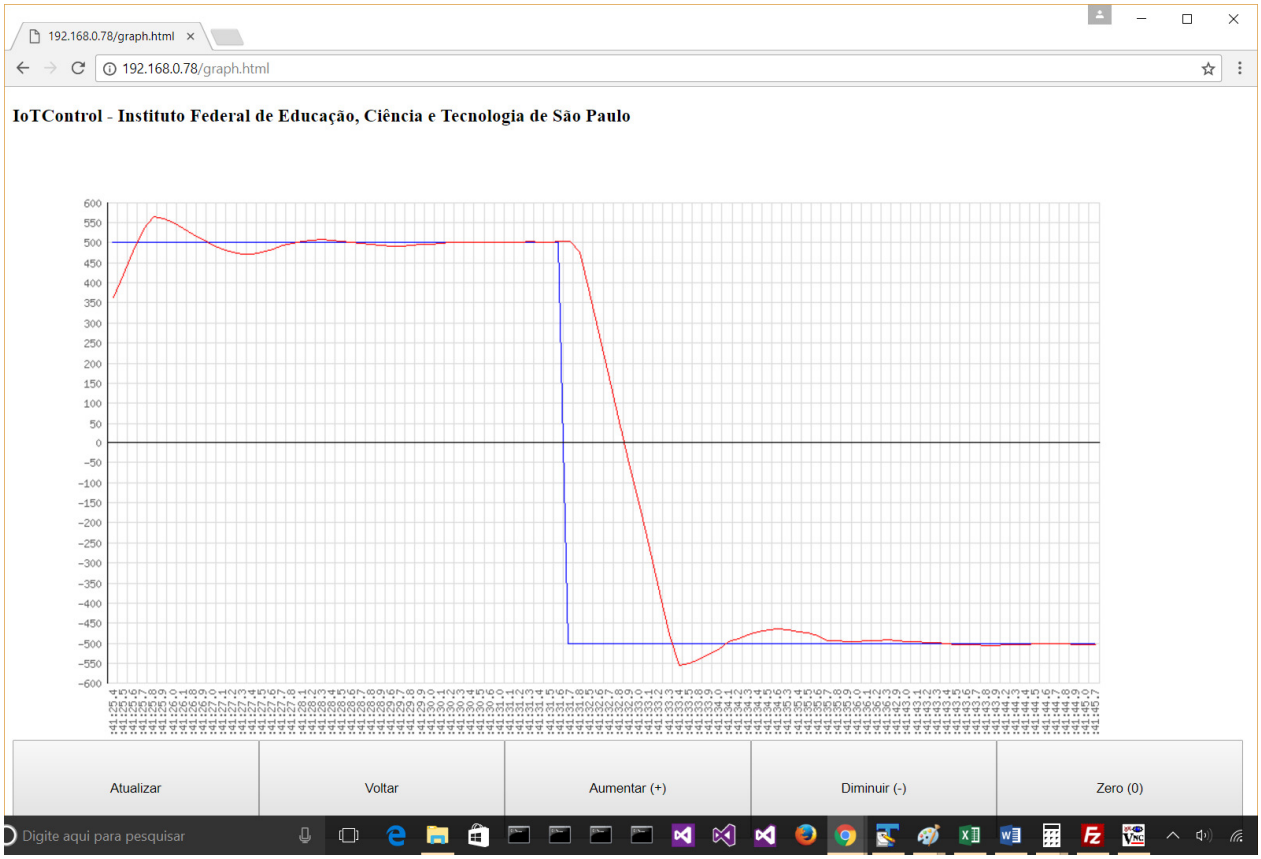


Figura 51 – Inversão de sentido

Fonte: Autor, 2017

A ferramenta apresenta uma interface gráfica intuitiva que permite ao usuário explorar as diversas funcionalidades desta plataforma.

6. CONCLUSÕES E TRABALHOS FUTUROS

Os resultados obtidos demonstram que a plataforma desenvolvida permite ao usuário operar e estudar algoritmos de controle possibilitando configurações e a visualização do comportamento do sistema em tempo de execução com a utilização de métodos gráficos, facilitando a interpretação de resultados. Esta ferramenta foi capaz de configurar parâmetros de controle dinamicamente, dispensando interromper o processo. É possível concluir que a ferramenta permite remotamente configurar e supervisionar uma planta de controle, o simples motor elétrico passou a ser um objeto inteligente, capaz de receber uma programação bem como disponibilizar dados por meio de uma rede internet, atende a demanda tecnológica de IoT (*Internet of Things* – Internet das Coisas), útil e relevante em aplicações de controle, possibilitando aos usuários e desenvolvedores o trabalho em uma rede computacional que permite um grande armazenamento de dados históricos, acesso remoto, controle de usuários conectados, monitoramento em tempo de execução, supervisão sistemática de operação. Estas vantagens enobrecem uma aplicação de controle podendo transformar um simples experimento em um dispositivo conectado possibilitando que diversos usuários possam se conectar e utilizar o experimento remotamente.

Como sugestão de trabalhos futuros o projeto poderá contar com adição de algoritmos de controle, melhoria de interface gráfica com telas ilustrativas e o incremento do número de elementos a ser controlado. O banco de dados c-treeACE pode ser explorado em sua interface ISAM permitindo uma execução com desempenho superior. Expandir a planta para uma aplicação real, como por exemplo ao invés de um simples motor elétrico de corrente contínua realizar o controle de um motor metro ferroviário e expandir a capacidade do sistema em uma plataforma computacional que permita controlar diversos motores. Nesta arquitetura seria possível controlar uma malha de trens metro ferroviários aplicando algoritmos para controlar a velocidade de diversos motores.

REFERÊNCIAS

ABDULAMEER, A. et al. **Tuning Methods of PID Controller for DC Motor Speed Control**. Indonesian Journal of Electrical Engineering and Computer Science, v. 3, n. 2, p. 343-349, 2016.

ANG, K. H.; CHONG, G.; LI, Y. **PID control system analysis, design, and technology**. IEEE transactions on control systems technology, v. 13, n. 4, p. 559-576, 2005.

ARM. **Cortex-A7 MPCore Technical Reference Manual**. <http://infocenter.arm.com/help/topic/com.arm.doc.ddi0464f/DDI0464F_cortex_a7_mpcore_r0p5_trm.pdf> Acesso em 15 de fevereiro de 2016.

BARBI, I. **Teoria fundamental do motor de indução**. Florianópolis: Editora da UFSC, 1985.

CASTAGNETTO J.; RAWAT H. et al. **Professional PHP programming**. Wrox Press Ltd, 1999.

DATE, C. J.; DARWEN, H. **A Guide to the SQL Standard**. New York: Addison-Wesley, vol. 3, 1987.

FACCIN, F. **Abordagem inovadora no projeto de controladores PID**. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

HAHN, J. **Internet of Things:** Mobile technology and location services in libraries. Library Technology Reports, cap. 1, vol. 53, 2017.

KO C. C.; CHEN B. M.; HU S. et al. **A web-based virtual laboratory on a frequency modulation experiment**. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 31, p. 295-303, 2001.

KOPETZ, H. **Emergence in the Internet of Things**. TU Wien, 2016.

KOPETZ, H. **Internet of things. In Real-time systems**. Springer US, p. 307-323, 2011.

LATHI, B. P. **Modern Digital and Analog Communication Systems 3e Osece**. Oxford university press, 1998.

LUETH, K. L. **IoT market analysis**: Sizing the opportunity. <http://theinternetofthings.report/ Resources/Whitepapers/546e0460-8f65-4523-9eb5-

9bf51b99f330_IoT%20market%20analysis%20Sizing%20the%20opportunity.pdf> Acesso em janeiro, 2016.

MITRA, S. K.; KUO, Y. **Digital signal processing: a computer-based approach**. McGraw-Hill Higher Education, vol. 2, 2006.

OGATA, K. **Engenharia de Controle Moderno**. Terceira edição. São Paulo: Prentice Hall do Brasil, 1998.

POSTEL, J. **Internet Protocol**. RFC, n. 791, 1981.

RASPBERRY-Pi. **Schematics**. <https://github.com/pborreli/documentation-3/blob/master/hardware/raspberrypi/Raspberry-Pi-Rev-2.1-Model-AB-Schematics.pdf> Acesso em 15 de fevereiro de 2016.

RICHARDSON, M.; WALLACE, S. **Getting started with raspberry PI**. O'Reilly Media, Inc., 2012.

SCHNEIDER, S. **Understanding The Protocols Behind The Internet Of Things**. <http://www.electronicdesign.com/iot/understanding-protocols-behind-internet-things> Acesso em janeiro, 2016.

ST. **An introduction to electric motors**. < http://www.st.com/content/ccc/resource/sales_and_marketing/presentation/application_presentation/group0/23/a1/94/a3/39/cf/4c/37/introduction_to_electric_motors_pres.pdf/files/introduction_to_electric_motors_pres.pdf/jcr:content/translations/en.introduction_to_electric_motors_pres.pdf> Acesso em 20 de fevereiro de 2016.

ST. **L293D Datasheet** < http://www.st.com/content/ccc/resource/technical/document/datasheet/04/ac/22/f9/20/5d/43/a1/CD00000059.pdf/files/CD00000059.pdf/jcr:content/translations/en.CD00000059.pdf> Acesso em 20 de fevereiro de 2016.

ST. **Products and solutions for Factory automation and control**. < http://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/brochure/ed/a4/6f/10/e8/da/42/0d/brfactory_web.pdf/files/brfactory_web.pdf/jcr:content/translations/en.brfactory_web.pdf> Acesso em 20 de fevereiro de 2016.

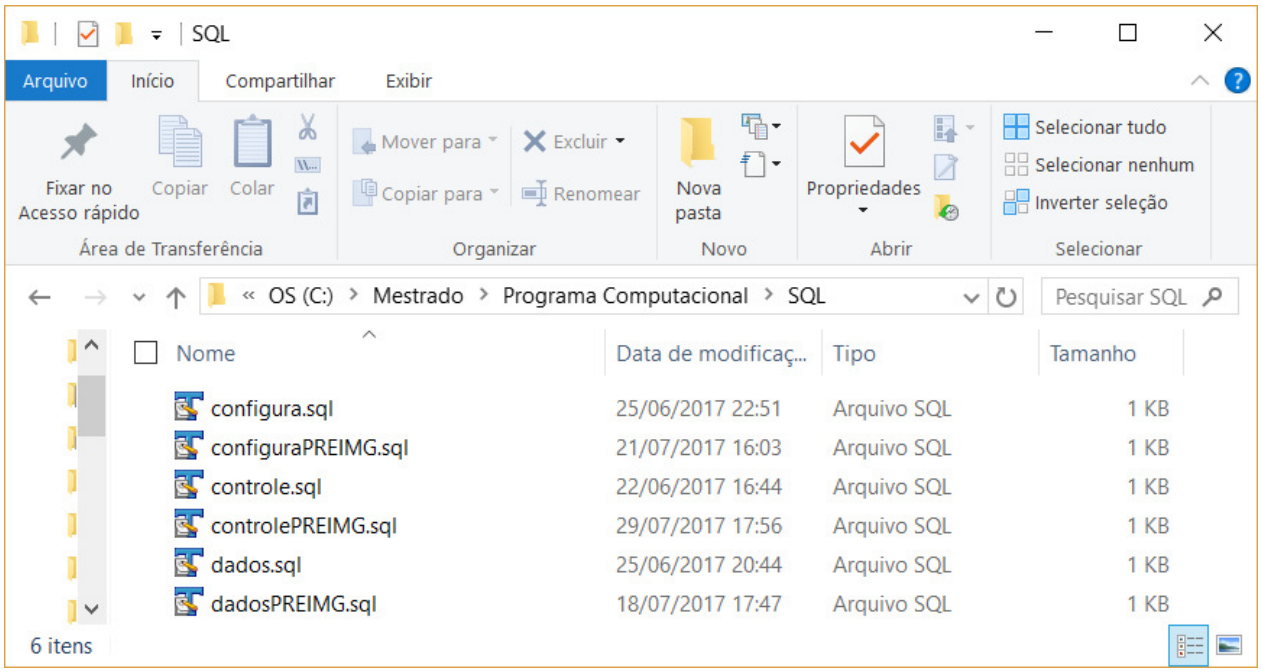
VAN ROSSUM, G. **The Python Language Reference**. Python Software Foundation, 2009.

XIAOYI, L. et al. **A research about remote flow control based on IOT**. In: Control and Decision Conference (CCDC), 2016 Chinese. IEEE, p. 4703-4706, 2016.

ZIEGLER, J.G; NICHOLS, N. B. **Optimum settings for automatic controllers**. Transactions of the ASME, p. 759–768, 1942.

APÊNDICE A – ALGORITMOS DESENVOLVIDOS

As estruturas de dados que possibilitam a execução dos algoritmos fazem parte do Programa Computacional SQL:



Configura:

```
create table "admin"."configura" (
      "pulsosvelcalc" integer,
      "tempovelcalc" float (8),
      "tempociclo" float (8),
      "grafpts" integer,
      "intvelfil" integer,
      "pwmfreq" float (8),
      "motorzm" float (8),
      "pwmzm" float (8)
)STORAGE_ATTRIBUTES 'preimg;';
insert into "admin"."configura"
values('200','0.10','0.20','100','10','250.0','200.0','20.0');
```

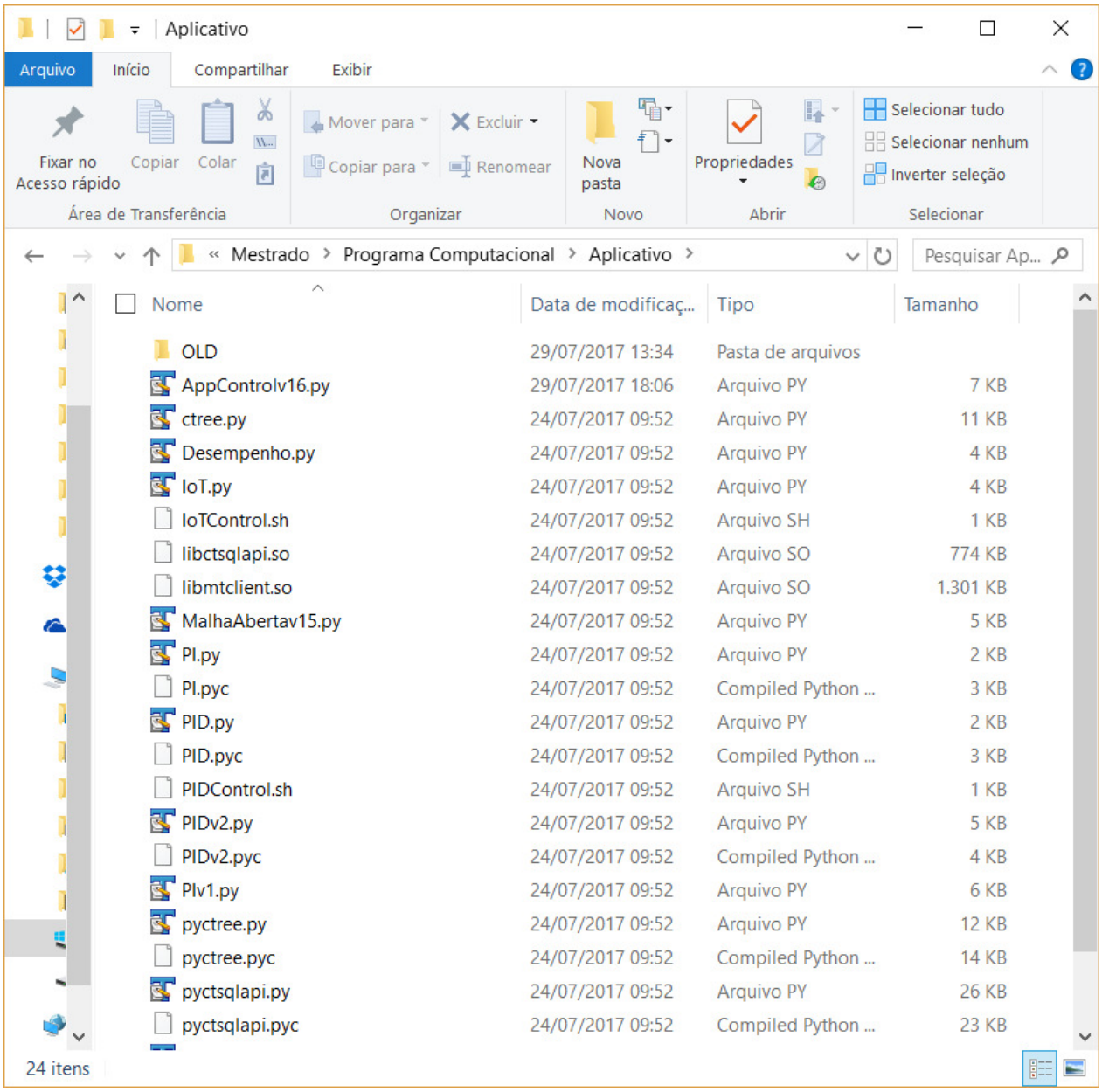
Controle:

```
create table "admin"."controle" (
      "ajuste" integer,
      "valork" float (8),
      "valorki" float (8),
      "valorkd" float (8),
      "fatora" float (8),
      "fatorb" float (8),
      "fatorc" float (8)
)STORAGE_ATTRIBUTES 'preimg;';
insert into "admin"."controle" values('0','1.00','0.50','0.05','0.0004','-
0.1254','32.343');
```

Dados:

```
create table "admin"."dados" (
      "ajuste" float (8),
      "velocidade" float (8),
      "erro" float (8),
      "kp" float (8),
      "ki" float (8),
      "kd" float (8),
      "kmotor" float (8),
      "tempo" timestamp
) STORAGE_ATTRIBUTES 'preimg;';
create index "admin"."indadados" on "admin"."dados" ("ajuste");
create index "admin"."indtdados" on "admin"."dados" ("tempo");
create index "admin"."indvdados" on "admin"."dados" ("velocidade");
```

Os algoritmos desenvolvidos fazem parte da seção; Programa Computacional Aplicativo.



```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
import threading
import os
from PID import PID
import sys
from ctypes import *
import pyctree
from datetime import datetime
from array import *

if os.path.isfile('/tmp/inic-exec'):
  print('Dupla inicia, encerrando programa.')
  sys.exit(0)

enable_pin = 23
to_L293D_IN1 = 25
to_L293D_IN2 = 24
encoderA = 16
encoderB = 20

counterA = 0.0
counterB = 0.0
contadorTempo = 0.0
velocidade = 0.0
velocidadeinst = 0.0
desejadavelocidade = 0.0

pulsosvelcalc = 10.0
tempovelcalc = 0.1
```

```
tempociclo = 0.1
velc = []#array('f', [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0])
interaveloc = 0
dir = 0

GPIO.setmode(GPIO.BCM)
GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(to_L293D_IN1, GPIO.OUT)
GPIO.setup(to_L293D_IN2, GPIO.OUT)
GPIO.setup(encoderA, GPIO.IN,  pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(encoderB, GPIO.IN,  pull_up_down=GPIO.PUD_DOWN)

p = GPIO.PWM(enable_pin, 250)

Tant = time.time()

pwm = 0.0
pid = 0.0
KVelocidade = 7.2
fatorK = 1.0
fatorKi = 0.0
fatorKd = 0.0
motorzm = 0.0
pwmzm = 0.0

fatora = 0.0004
fatorb = - 0.1254
fatorc = 32.343

def calcVelocidade():
  global Tant
  global counterA
```

```
  global counterB
  global velocidade
  global velocidadeinst
  global pulsosvelcalc
  global tempovelcalc
  global velc
  global interaveloc
  global dir

  if interaveloc == 0:
    return

  Tnow=time.time()

  interval = Tnow-Tant

  if ((interval >= tempovelcalc) | ((counterA >= pulsosvelcalc) & (counterB >=
pulsosvelcalc))):
    if counterA > 0.0:
      for i in range (interaveloc-1,0,-1):
        velc[i] = velc[i-1]
      if counterA < counterB:
        velc[0] = counterA/(interval)
      else:
        velc[0] = counterB/(interval)
      if dir >= 10:
         velc[0] = velc[0] * (-1)

      velocidadeinst = velc[0]
      velocsum = 0.0
      for i in range (0,interaveloc):
        velocsum = velocsum + velc[i]
      velocidade = velocsum / interaveloc
```

```
    else:
      for i in range (interaveloc-1,0,-1):
        velc[i] = velc[i-1]
      velc[0] = 0.0
      velocidade = velc[0]

    Tant = Tnow
    counterA = 0.0
    counterB = 0.0
    #execControle()


def printSpeed():
  print (' ')
  print ('Velocidade p/s: %s' %velocidade)
  print ('desejadavelocidade p/s: %s' %desejadavelocidade)

def saidaPWM(w1, w2):
  if w2:
    GPIO.output(to_L293D_IN1, 1)
    GPIO.output(to_L293D_IN2, 0)
    p.ChangeDutyCycle(w1)
  else:
    GPIO.output(to_L293D_IN1, 0)
    GPIO.output(to_L293D_IN2, 1)
    p.ChangeDutyCycle(w1)

def velPpwm(valpid):
  global desejadavelocidade
  global motorzm
  global pwmzm
  global KVelocidade
```

```
global fatora
global fatorb
global fatorc

if valpid < 0.0:
  valvel = valpid * (-1.0)
else:
  valvel = valpid

if valpid >= motorzm:
  calcpwm = fatora*(valvel*valvel) + fatorb*(valvel) + fatorc
else:
  calcpwm = valpid / KVelocidade

dvelocidade = desejadavelocidade

if dvelocidade < 0.0:
  dvelocidade = dvelocidade * (-1.0)
  if calcpwm < 0.0:
    calcpwm = calcpwm * (-1.0)

if dvelocidade <> 0.0:
  if calcpwm < pwmzm:
    calcpwm = pwmzm
if calcpwm < 0.0:
  calcpwm = 0.0

if calcpwm > 100.0:
  calcpwm = 100.0

return calcpwm
```

```
def execControle():
  global velocidade
  global velocidadeinst
  global desejadavelocidade
  global controle
  global pid
  global pwm

  controle.setPoint(desejadavelocidade)

  pid = controle.update(velocidade)
  pid = velocidade + pid
  pwm = velPpwm(pid)

  if desejadavelocidade >= 0.0:
    saidaPWM(pwm, 1)
  else:
    saidaPWM(pwm, 0)

def cb_EncoderA(channel):
  global  counterA
  global pulsosvelcalc
  global dir
  counterA += 1.0
  if GPIO.input(encoderB):
    if dir > 0:
      dir -= 1
  #if counterA >= pulsosvelcalc:
  #  calcVelocidade()


def cb_EncoderB(channel):
```

```
  global  counterB
  global  pulsosvelcalc
  global  dir
  counterB += 1.0
  if GPIO.input(encoderA):
    if dir < 20:
      dir += 1
  #if counterB >= pulsosvelcalc:
  #  calcVelocidade()


def main():
  global velocidade
  global desejadavelocidade
  global controle
  global pid
  global fatorK
  global fatorKi
  global fatorKd
  global pulsosvelcalc
  global tempovelcalc
  global tempociclo
  global interaveloc
  global KVelocidade
  global pwm
  global motorzm
  global pwmzm
  global fatora
  global fatorb
  global fatorc
```

```
  iniciado = open("/tmp/inic-exec", 'w+')

  GPIO.output(to_L293D_IN1, 1)
  GPIO.output(to_L293D_IN2, 0)

  p.start(0)
  p.ChangeDutyCycle(0)

  controle=PID(fatorK,fatorKi,fatorKd)

  conn = pyctree.connect(user="admin", password="ADMIN", database="ctreeSQL",
host="localhost",port="6597")

  csr = conn.cursor()

  resret = 0.0

  xstamp = datetime.now() #(2016,9,27,16,00,00)

  pwm = 0.0

  pulsosvelcalc = 10.0
  tempovelcalc = 0.1
  tempociclo = 0.1

  csr.execute("SELECT pulsosvelcalc, tempovelcalc, tempociclo, grafpts,
intvelfil, pwmfreq, motorzm, pwmzm FROM configura")
  resstr = csr.fetchone()
  pulsosvelcalc = resstr[0]
  tempovelcalc = resstr[1]
  tempociclo = resstr[2]
  grafpts = resstr[3]
  interaveloc = resstr[4]
```

```
  pwmfreq = resstr[5]
  motorzm = resstr[6]
  pwmzm = resstr[7]

  p.ChangeFrequency(pwmfreq)

  for i in range(interaveloc):
    velc.append(0.0)



  print "Obrigado por experimentar AppControl"
  print "Configuracoes aplicadas:"
  print ('pulsosvelcalc: %s' %pulsosvelcalc)
  print ('tempovelcalc: %s' %tempovelcalc)
  print ('tempociclo: %s' %tempociclo)
  print ('grafpts: %s' %grafpts)

  GPIO.add_event_detect(encoderA, GPIO.RISING, callback=cb_EncoderA)
  GPIO.add_event_detect(encoderB, GPIO.RISING, callback=cb_EncoderB)

  iniciado.close()

  while True:
    tempoprocesso = time.time()
    csr.execute("SELECT ajuste, valorK, valorKi, valorKd, fatora, fatorb,
fatorc FROM controle")
    resstr = csr.fetchone()
    resret = float(resstr[0])
    #ajusta velocidade desejada
    desejadavelocidade = resret

    # lendo valores do banco
    fatorK = resstr[1]
```

```
    fatorKi = resstr[2]
    fatorKd = resstr[3]
    fatora = resstr[4]
    fatorb = resstr[5]
    fatorc = resstr[6]

    controle.setKp(fatorK)
    controle.setKi(fatorKi)
    controle.setKd(fatorKd)

    #if desejadavelocidade != controle.getPoint:
    calcVelocidade()
    #controle.setPoint(desejadavelocidade)
    #pid = controle.update(velocidade)
    execControle()

    xstamp = datetime.now() #(2016,9,27,16,00,00)
    #print desejadavelocidade, velocidade,pwm
    csr.execute("INSERT INTO dados (ajuste, velocidade, erro, Kp, Ki, Kd,
Kmotor, tempo) VALUES(?,?,?,?,?,?,?,?)",(desejadavelocidade,
velocidade,pwm,fatorK,fatorKi,fatorKd, KVelocidade,xstamp))
    conn.commit()
    tempoprocesso = time.time() - tempoprocesso
    if tempociclo > tempoprocesso:
       tempoprocesso = tempociclo - tempoprocesso
       time.sleep(tempoprocesso)

    if os.path.isfile('/tmp/abortar-exec'):
     print('Abortar')
     os.remove('/tmp/abortar-exec')
     os.remove('/tmp/inic-exec')
     break
```

```
  GPIO.cleanup()
  csr.close()
  conn.close()
  print ('Obrigado por usar AppControl!')

if __name__ == '__main__':
    main()
```

PID:

```
            self.erro = self.vel_desejada - realimentacao_valor

            self.P_valor = self.Kp * self.erro
            self.D_valor = self.Kd * (self.erro - self.Derivador)
            self.Derivador = self.erro

            self.Integrador = self.Integrador + self.erro

            if self.Integrador > self.Integrador_max:
                  self.Integrador = self.Integrador_max
            elif self.Integrador < self.Integrador_min:
                  self.Integrador = self.Integrador_min

            self.I_valor = self.Integrador * self.Ki

            PID = self.P_valor + self.I_valor + self.D_valor

            return PID
```

Os algoritmos desenvolvidos fazem parte da seção; Programa Computacional Rede.

Index:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img {
    max-width:100%;
}
</style>
<table>
<form action="seleciona.html" method="post">
<input type="submit" style="height:100px; width:250px;" value="Controle">
</form>
<form action="configura.php" method="post">
<input type="submit" style="height:100px; width:250px;" value="Configura">
</form>
<form action="graph.html" method="post">
<input type="submit" style="height:100px; width:250px;" value="Gráfico">
</form>
<form action="geraexe.php" method="post">
<input type="submit" style="height:100px; width:250px;" value="Abortar">
</form>
</table>
<body>
</body>
</head>
</html>
```

Controle.php:

```
<?php
   print("<!DOCTYPE html>\n");
   print("<html>\n");
   print("<head>\n");
   print("\t<title>IoTControl Configuração de Controle</title>\n");
   print("<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\">\n");
   $osrunning = php_uname();
   if (strtoupper(substr(PHP_OS, 0, 3)) !== 'WIN') {
      putenv("ODBCINI=/etc/odbc.ini");
   }
   popen("/var/www/html/PID.sh", "r");
   error_reporting(E_ERROR | E_PARSE);
   print("\t<h4>IoTControl - Configurações PID</h4>\n");
   $ses = Initialize();
   $query = "SELECT ajuste, valork, valorki, valorkd FROM controle";
   $qry = odbc_exec($ses, $query);
   $result = odbc_fetch_array($qry);
   if (!$qry)
    H_Erro("odbc_exec(SELECT TOP)");

   function Initialize() {
      $test_dsn = "c-treeACE ODBC Driver";
      if(($ses = odbc_pconnect( $test_dsn,"admin","ADMIN")) == FALSE)
      {
        H_Erro("odbc_connect()");
      }else
      return ($ses);
   }
```

```
   function Done ($ses) {
      // logout
      odbc_close($ses);
   }

   function H_Erro($msg) {
      $err = odbc_errormsg();
      print("$msg - SQL ERROR: [$err] <br>\n");
      print("*** Processamento abortado *** <br>\n");
      exit();
   }
   print("</head>\n");
   print("<body>\n");
   print("</body>\n");
   print("</html>\n");
?>
<html>
    <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <table>
      <form method="post" action="controleexe.php">
        Fator K:
        <input type="text" name="valork"
value="<?php echo $result['valork'] ?>">
        <br>
        Fator Ki:
        <input type="text" name="valorki"
value="<?php echo $result['valorki'] ?>">
        <br>
        Fator Kd:
        <input type="text" name="valorkd"
```

```
value="<?php echo $result['valorkd'] ?>">
        <br><br>
        Velocidade Desejada (p/s):
        <input type="number" min="-1000" max="1000" step="1"
value="<?php echo $result['ajuste'] ?>" name="ajuste"
               for="power" oninput="power.value=ajuste.value" >
        <br>

        <input type="range" min="-1000" max="1000"  step="1"
value="<?php echo $result['ajuste'] ?>" name="power" list="powers"
               style="height:15%; width:95%; float:center;"
               for="ajuste" oninput="ajuste.value=power.value" >

        <datalist id="powers">
          <option value="-1000">
          <option value="-750">
          <option value="-500">
          <option value="-250">
          <option value="0">
          <option value="250">
          <option value="500">
          <option value="750">
          <option value="1000">
        </datalist>
        <br>
        <input type="submit" style="height:100px; width:250px;"
value="Atualizar">
      </form>
      <form action="graph.html" method="post">
      <input type="submit" style="height:100px; width:250px;" value="Gráfico">
      </form>
```

```
      <form action="index.html" method="post">
      <input type="submit" style="height:100px; width:250px;" value="Voltar">
    </form>
      <form action="Aumenta.php" method="post">
      <input type="submit" style="height:100px; width:250px;"
value="Aumentar (+)">
      </form>
      <form action="Diminui.php" method="post">
      <input type="submit" style="height:100px; width:250px;"
value="Diminuir (-)">
      </form>
      </form>
      <form action="Zero.php" method="post">
      <input type="submit" style="height:100px; width:250px;"
value="Zero (0)">
      </form>
    </table>
    </head>
    <body>
    </body>
</html>
```

controleEXE.php:

```
<?php
   print("<html>\n");
   print("<head>\n");
   print("\t<title>IoTControl Configuração de Controle</title>\n");
   print("</head>\n");
   print("<body>\n");
   $osrunning = php_uname();
   if (strtoupper(substr(PHP_OS, 0, 3)) !== 'WIN') {
```

```
      putenv("ODBCINI=/etc/odbc.ini");
   }
   error_reporting(E_ERROR | E_PARSE);
   print("\t<h4>IoTControl - Comando em processamento</h4>\n");
   Atualiza();


   function Initialize() {
      $test_dsn = "c-treeACE ODBC Driver";
      if(($ses = odbc_pconnect( $test_dsn,"admin","ADMIN")) == FALSE)
      {
        H_Erro("odbc_connect()");
      }else
      return ($ses);
   }
   function Atualiza() {
      if (filter_var($_POST['ajuste'], FILTER_VALIDATE_INT) === 0 ||
!filter_var($_POST['ajuste'], FILTER_VALIDATE_INT) === false)
      {
            $ajuste=$_POST['ajuste'];
        } else
        {
            H_Erro("Velocidade Desejada $_POST[ajuste]");
      }
      if (filter_var($_POST['valork'], FILTER_VALIDATE_FLOAT) === 0.0 ||
!filter_var($_POST['valork'], FILTER_VALIDATE_FLOAT) === false)
      {
            $valork=$_POST['valork'];
        } else
        {
            H_Erro("Fator K $_POST[valork]");
      }
```

```
      if (filter_var($_POST['valorki'], FILTER_VALIDATE_FLOAT) === 0.0 ||
!filter_var($_POST['valorki'], FILTER_VALIDATE_FLOAT) === false)
      {
            $valorki=$_POST['valorki'];
        } else
        {
            H_Erro("Fator Ki $_POST[valorki]");
      }

      if (filter_var($_POST['valorkd'], FILTER_VALIDATE_FLOAT) === 0.0 ||
!filter_var($_POST['valorkd'], FILTER_VALIDATE_FLOAT) === false)
      {
            $valorkd=$_POST['valorkd'];
        } else
        {
            H_Erro("Fator Kd $_POST[valorkd]");
      }
     $ses = Initialize();
     $query = "UPDATE controle
            SET ajuste = '$ajuste'";
     $qry = odbc_exec($ses, $query);
     if (!$qry)
         H_Erro("odbc_exec()");
     $query = "UPDATE controle
            SET valork = '$valork'";
     $qry = odbc_exec($ses, $query);
     if (!$qry)
         H_Erro("odbc_exec()");
     $query = "UPDATE controle
            SET valorki = '$valorki'";
     $qry = odbc_exec($ses, $query);
```

```
      if (!$qry)
          H_Erro("odbc_exec()");
      $query = "UPDATE controle
             SET valorkd = '$valorkd'";
      $qry = odbc_exec($ses, $query);
      if (!$qry)
          H_Erro("odbc_exec()");
      if (!odbc_commit($ses))
          H_Erro("odbc_commit()");
      print("ajuste: [$ajuste] <br>\n");
      print("valork: [$valork] <br>\n");
      print("valorki: [$valorki] <br>\n");
      print("valorkd: [$valorkd] <br>\n");
      print("*** Processamento concluído com sucesso *** <br>\n");
   }
   function H_Erro($msg) {
      $err = odbc_errormsg();
      print("$msg - ERRO, parâmetro incorreto ou sintaxe: [$err] <br>\n");
      print("*** Processamento abortado *** <br>\n");
      exit();
   }
   print("</body>\n");
   print("</html>\n");
?>
<html>
    <head>
    </head>
    <body>
    <?php
      ob_start(); // ensures anything dumped out will be caught
      $url = 'Location: '. $_SERVER['HTTP_REFERER'];
```

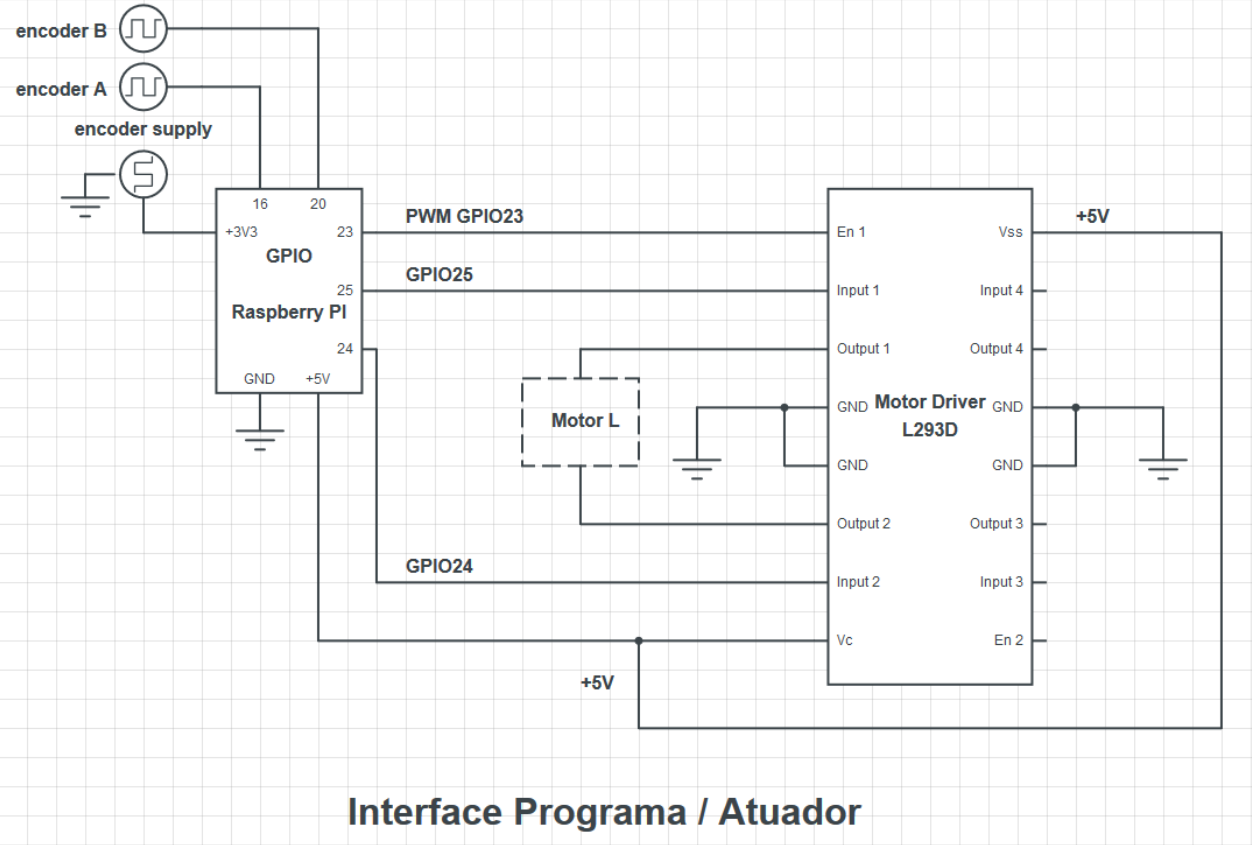
```
      while (ob_get_status())
      {
            ob_end_clean();
      }

      header( "$url" );
      ?>
    </body>
</html>
```

Graph.html:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<h3>IoTControl - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo</h3>
<img src="graph.php" />
<br>
<table>
<form>
<input type=button style="height:100px; width:250px;" value="Atualizar" onClick="window.location.reload()">
</form>
<form action="seleciona.html" method="post">
<input type="submit" style="height:100px; width:250px;" value="Voltar">
</form>
<form action="Aumenta.php" method="post">
<input type="submit" style="height:100px; width:250px;" value="Aumentar (+)">
</form>
<form action="Diminui.php" method="post">
<input type="submit" style="height:100px; width:250px;" value="Diminuir (-)">
</form>
```

```
</form>
<form action="Zero.php" method="post">
<input type="submit" style="height:100px; width:250px;" value="Zero (0)">
</form>
</table>
</head>
</html>
```

APÊNDICE B – CIRCUITO ELÉTRICO



Interface Programa / Atuador

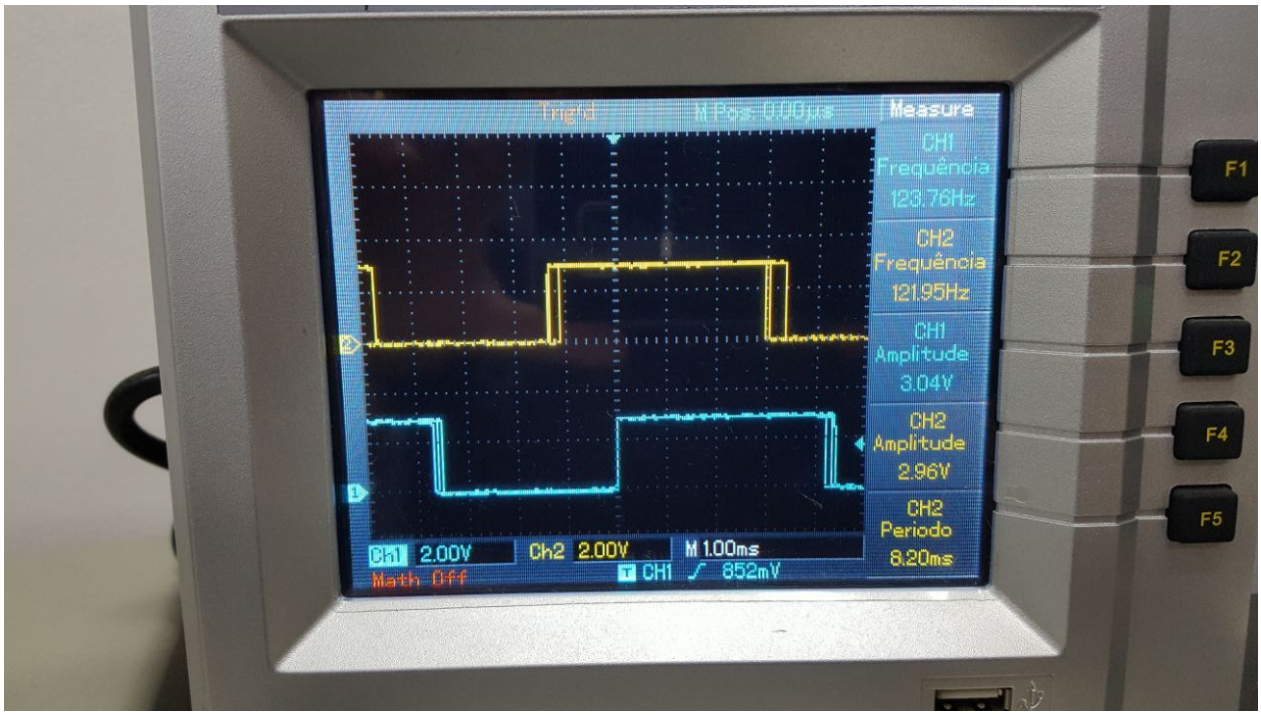
APÊNDICE C – TABELA DE CONEXÕES ELÉTRICAS

Dispositivo Computacional Raspberry Pi	Driver L293D	Motor com Encoder
Pino 1 / +3,3V		Pino 5 / Encoder Supply
Pino 2 / +5V	Pino 8 / VS e Pino 16 / VSS	
Pino 6 / GND	Pinos 4, 5, 12 e 13 / GND	Pino 2 / GND
Pino 16 / Entrada Interrupção		Pino 3 / Encoder A phase
Pino 20 / Entrada Interrupção		Pino 4 / Encoder B phase
Pino 23 / Saída PWM	Pino 1 / Enable 1	
Pino 24 / Saída Digital	Pino 7 / Input 2	
Pino 25 / Saída Digital	Pino 2 / Input 1	

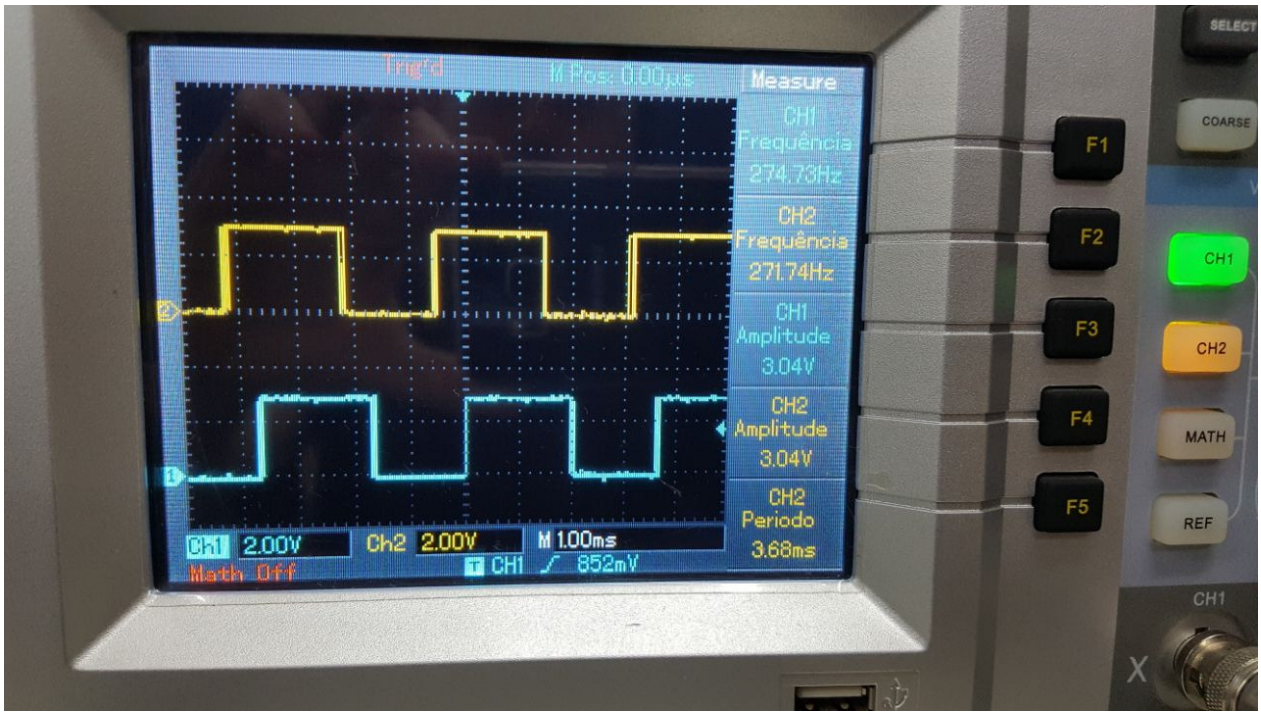
Quadro 7 – Tabela de Conexões Elétricas

APÊNDICE D – SINAIS ENCODER NO OSCILOSCÓPIO

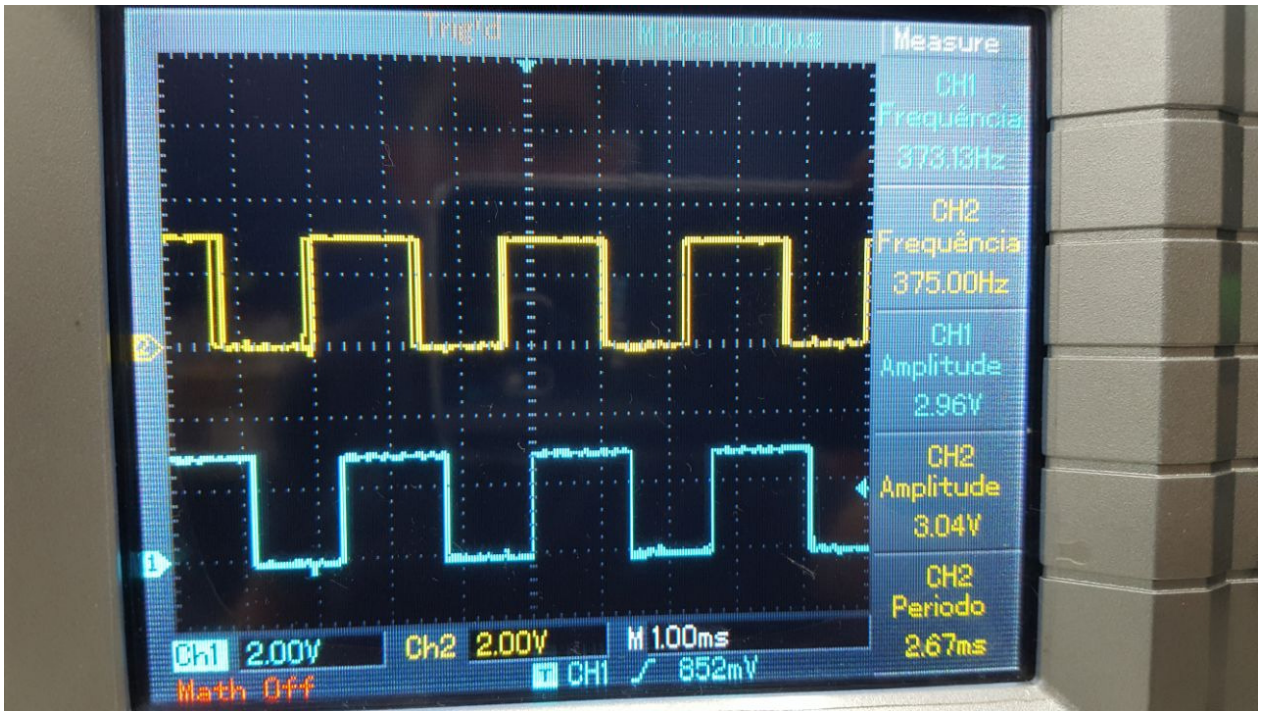
PWM 20%



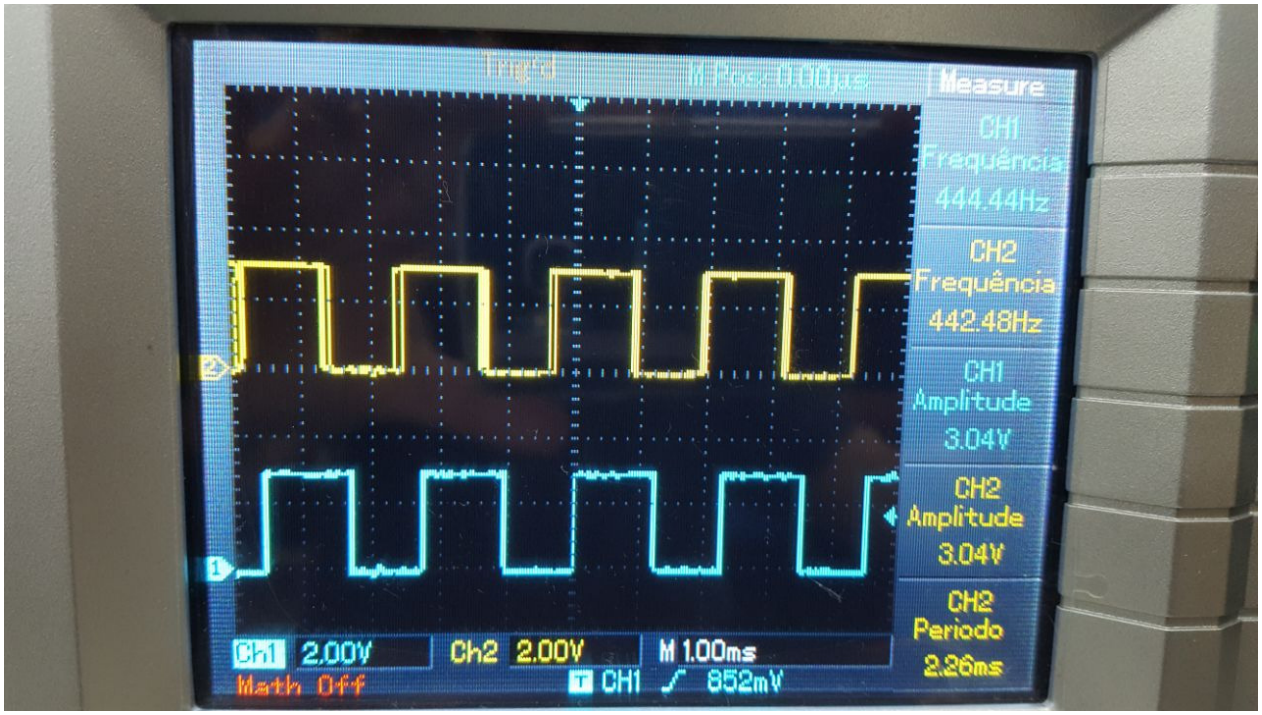
PWM 30%



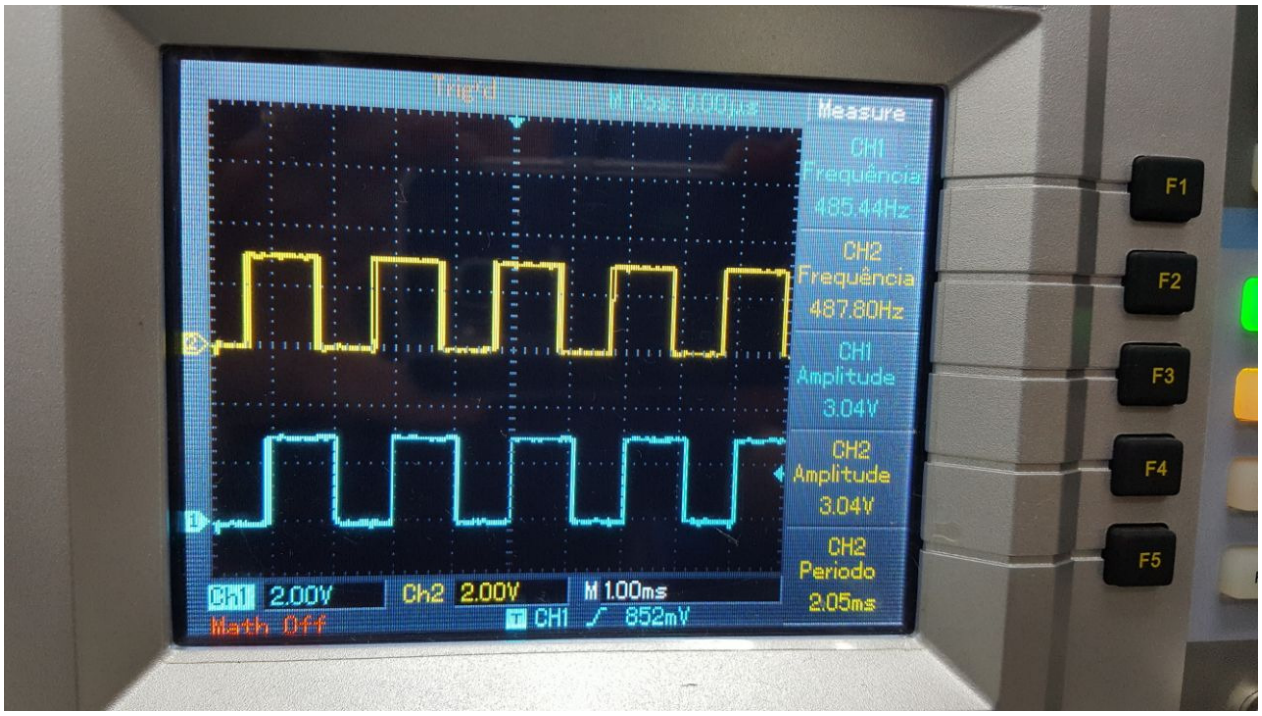
PWM 40%


Measure
CH1
Frequência
378.13Hz
CH2
Frequência
375.00Hz
CH1
Amplitude
2.96V
CH2
Amplitude
3.04V
CH2
Periodo
2.67ms
Ch1 2.00V
Ch2 2.00V
M 1.00ms
CH1 852mV
Math Off

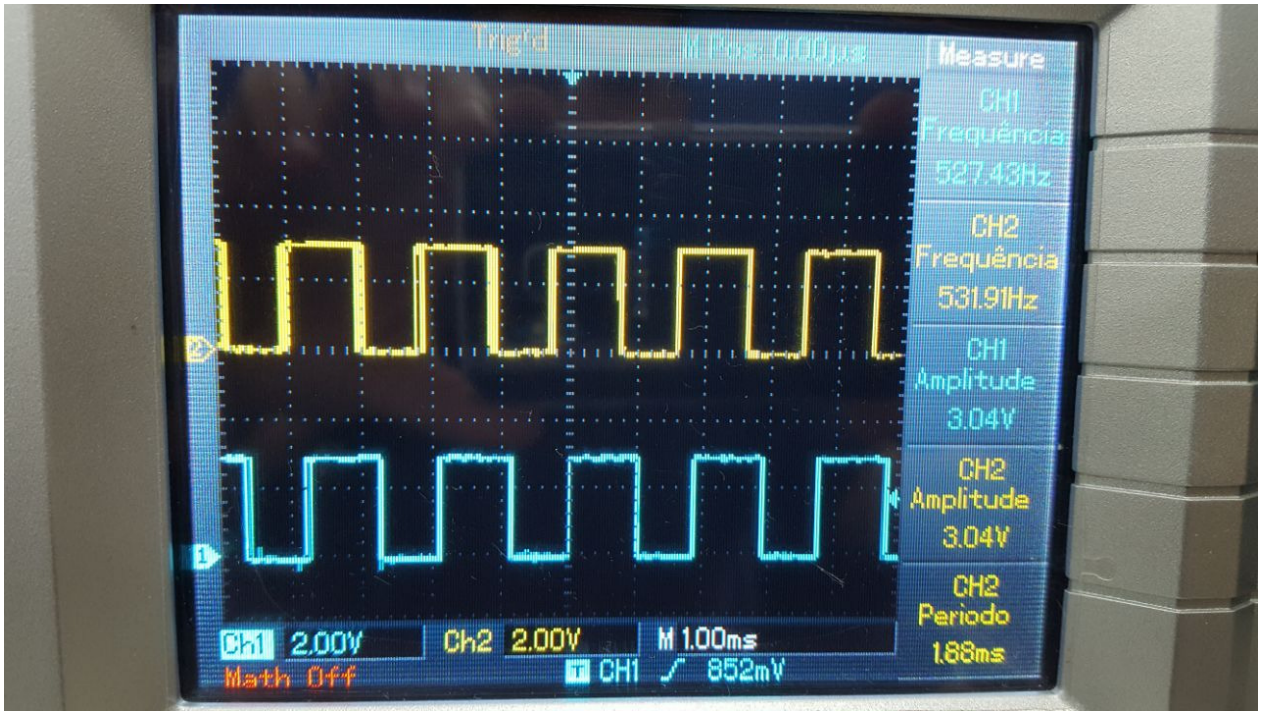
PWM 50%


Measure
CH1
Frequência
444.44Hz
CH2
Frequência
442.48Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Periodo
2.26ms
Ch1 2.00V
Ch2 2.00V
M 1.00ms
CH1 852mV
Math Off

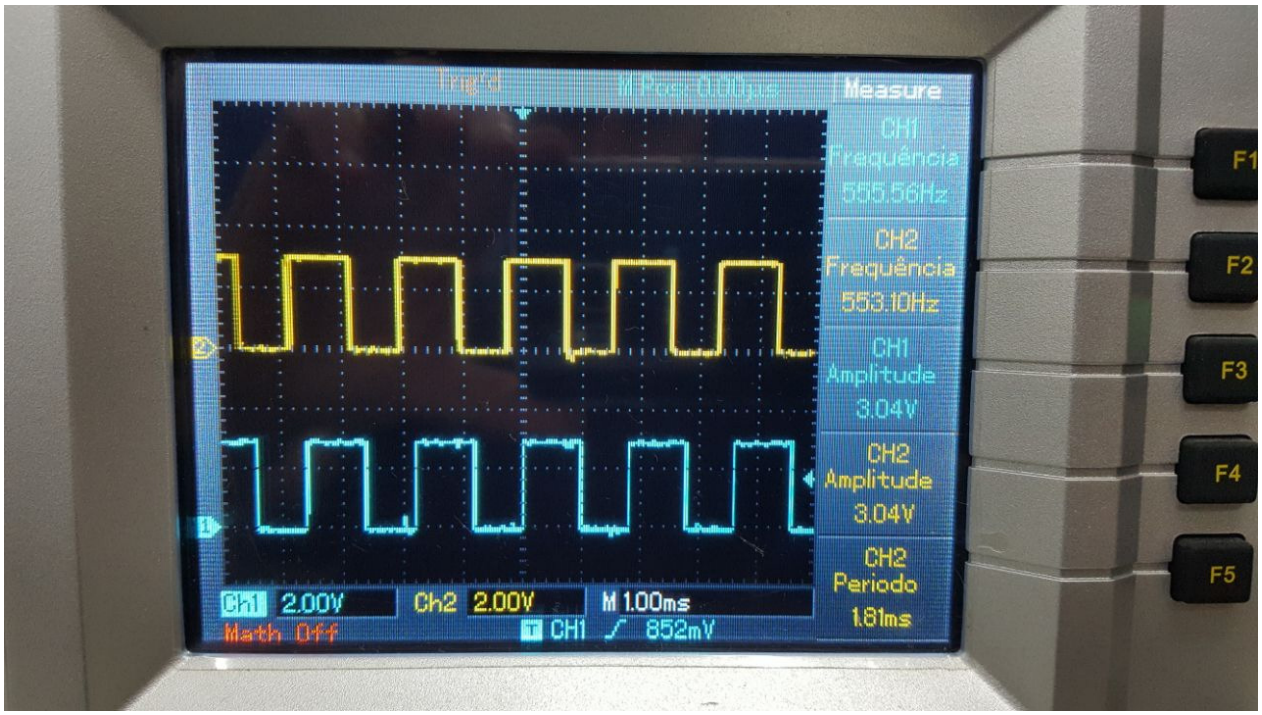
PWM 60%


Measure
CH1
Frequência
485.44Hz
CH2
Frequência
487.80Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Periodo
2.05ms
F1
F2
F3
F4
F5
Ch1 2.00V
Ch2 2.00V
M 1.00ms
CH1 852mV
Math Off

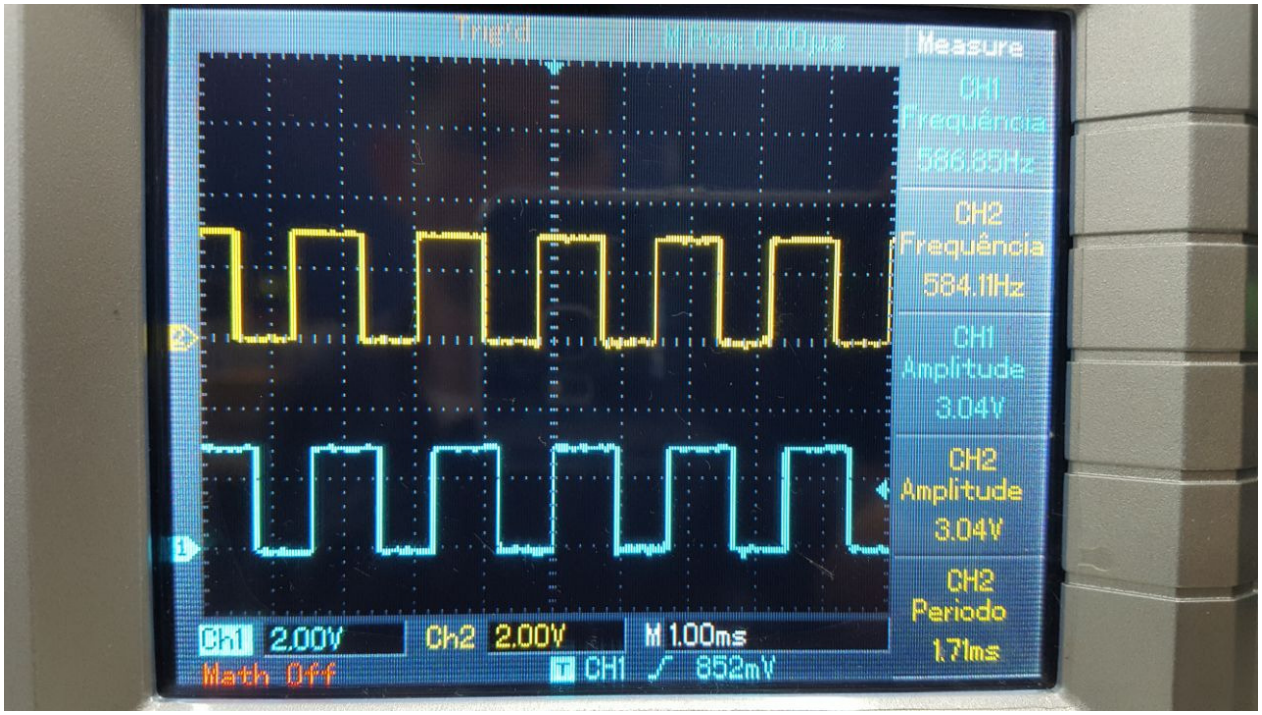
PWM 70%


Measure
CH1
Frequência
527.43Hz
CH2
Frequência
531.91Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Periodo
1.88ms
Ch1 2.00V
Ch2 2.00V
M 1.00ms
CH1 852mV
Math Off

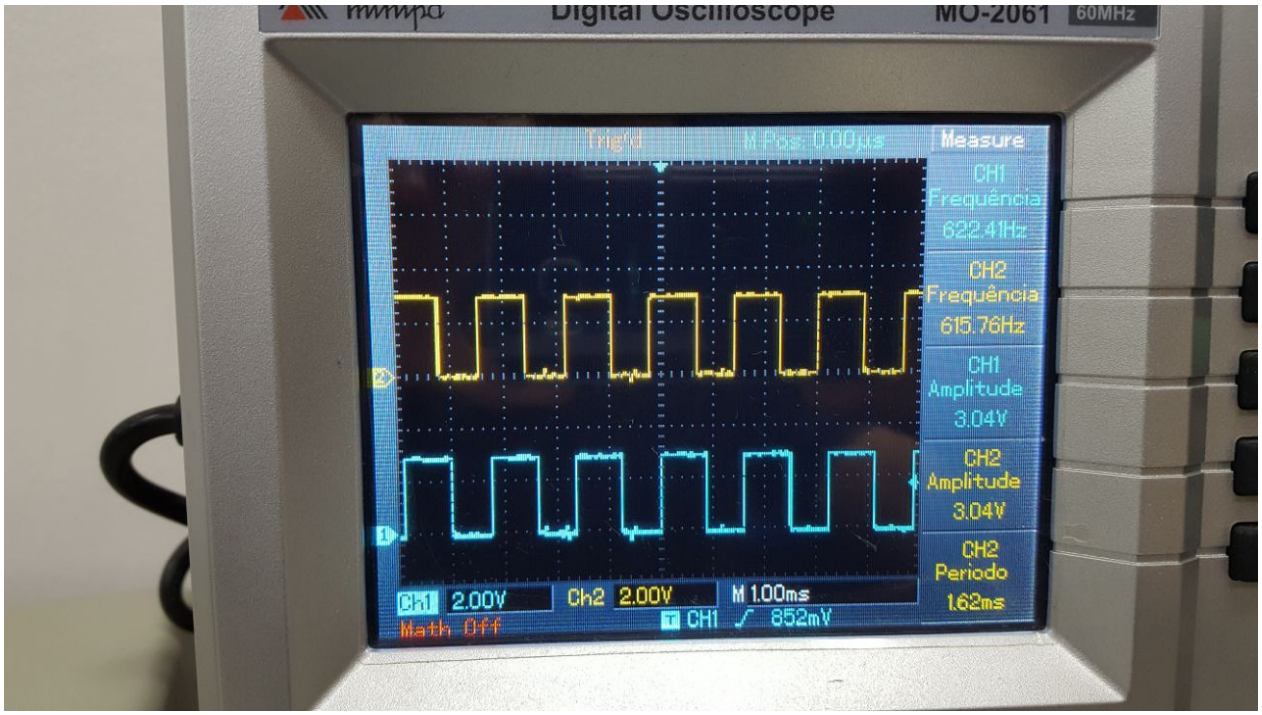
PWM 80%


Measure
CH1
Frequência
555.56Hz
CH2
Frequência
553.10Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Período
1.81ms
F1
F2
F3
F4
F5
Ch1 2.00V
Ch2 2.00V
M 1.00ms
Math Off
CH1 852mV

PWM 90%


Measure
CH1
Frequência
566.85Hz
CH2
Frequência
584.11Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Período
1.71ms
Ch1 2.00V
Ch2 2.00V
M 1.00ms
Math Off
CH1 852mV

PWM 100%


Digital Oscilloscope
MO-2061
60MHz
Measure
CH1
Frequência
622.41Hz
CH2
Frequência
615.76Hz
CH1
Amplitude
3.04V
CH2
Amplitude
3.04V
CH2
Período
1.62ms
CH1 2.00V
Ch2 2.00V
M 1.00ms
Math Off
CH1 852mV

ANEXO I – FairCom c-treeACE

O servidor c-treeACE suporta gerenciamento de banco de dados de alto nível, incluindo:

- **Computação cliente** / **servidor** - Aumenta o desempenho e fornece a capacidade de manter a integridade do banco de dados, especialmente em ambientes multiusuários. O princípio básico da computação cliente / servidor é: aplicativos, ou "clientes", interagem com o servidor, que gerencia as operações de arquivos e se comunica com os clientes.
- **Processamento de transações on-line (OLTP)** - O servidor c-treeACE pode agrupar um conjunto especificado de operações, chamado de "transação", e garantir que todas elas sejam feitas ou, se houver um problema, nenhuma será feita, por exemplo, Toda a factura é processada, ou nenhuma.
- **Controles de segurança** - O acesso ao servidor c-treeACE é controlado com IDs de usuário, senhas, permissões de arquivos e criptografia. Usuários e arquivos podem ser adicionados aos "grupos" definidos pelo administrador, por exemplo, departamento de remessa, departamento de folha de pagamento.
- **Manutenção de banco de dados e utilitários** - O servidor c-treeACE salva automaticamente as informações necessárias para uso em backups automáticos ou especificados pelo Administrador e recuperação de problemas.
- **Flexibilidade de configuração** - Noções básicas, como o protocolo de comunicação que o servidor c-treeACE usa e alocações de memória para aplicar para usuários específicos, a uma ampla gama de controles avançados.

O administrador do servidor c-treeACE tem as seguintes seis áreas de responsabilidade, cada uma das quais pode ser dividida entre várias pessoas:

Instalação

Alguém, não necessariamente o Administrador, deve carregar fisicamente o programa computacional do servidor c-treeACE no ambiente de computação. Uma vez concluído, os problemas de instalação geralmente não são mais uma preocupação, a menos que o servidor c-treeACE precise ser reinstalado, por exemplo, para instalar uma nova versão.

Operando o Servidor

Iniciando e Parando o Servidor c-treeACE: Qualquer usuário pode iniciar o Servidor c-treeACE executando o módulo executável, ctsrvr, como qualquer outro programa no ambiente. O servidor também pode ser configurado para executar como um serviço, sendo iniciado automaticamente pelo sistema operacional.

Controlando o acesso ao servidor c-treeACE

O c-treeACE permite a configuração de usuários e senhas, definindo regras de acesso aos arquivos. Estas configurações podem ser realizadas por meio de utilitários de linha de comando ou ferramentas gráficas. Mesmo com o servidor instalado no Raspberry, pode-se utilizar uma máquina na mesma rede com Windows ou Linux Desktop e administrar o banco.

Os principais utilitários de administração do c-treeACE:

- ctadmn - utilitário de administração do servidor c-treeACE, permite monitorar os usuários conectados ao servidor e suas operações, informações de consumo de recursos, parar o servidor de forma parcial (*quiesce*, para algum gerenciamento específico) ou o seu completo desligamento, além de outras tarefas.
- sa_admin – utilitário para criação de usuários e grupos.
- ctstat – utilitário para obtenção de informações estatísticas de operação do servidor, como número de bytes ou registros lidos/escritos por segundo, percentual dos dados buscados que encontram-se em cache, número de usuários conectados entre outras.

- ctdump – utilitário para solicitar ou programar o backup de dados do servidor.
- ctrdmp – utilitário para restaurar o backup de dados do servidor.
- ctstop – utilitário para interromper a operação do servidor.

Como manter a integridade do banco de dados

O c-tree Server possui algumas alternativas para manter a integridade dos dados durante sua operação. A primeira garantia é que, para arquivos criados com um modo de transação completo, mesmo em caso de queda do servidor por falta de energia ou outro problema qualquer, quando o servidor é reiniciado, os arquivos e tabelas são recuperados ao seu último estado transacional. Além desta operação automática do servidor, pode-se agendar backups periódicos de arquivos gerados pelo sistema para uso posterior na recuperação de problemas ou retornar um banco de dados para seu status em um momento anterior. Este agendamento pode ser feito por meio do arquivo de configuração do servidor ou do utilitário ctdmp descrito anteriormente. Em ambos os casos, um arquivo script simples, segundo uma sintaxe específica, é utilizado para determinar quais arquivos e de quanto em quanto tempo serão salvos.

Uma terceira alternativa para garantir a integridade dos dados é replicar os dados para um segundo servidor. Esta operação de replicação transita pela rede as alterações que ocorrem nos registros.

Configurando o servidor c-treeACE

O servidor c-treeACE é configurado por meio do arquivo ctsrvr.cfg. Neste arquivo, por meio de uma série de palavras-chave, pode-se configurar limites de alocação de memória, protocolos de comunicação, ativar um script de backup, o nome com que o servidor será conhecido na rede, sua porta, etc.

O servidor c-treeACE é iniciado por qualquer usuário autorizado a iniciar processo ctsrvr, ou automaticamente pelo sistema operacional como um serviço.

O ID (identificação) de usuário "ADMIN" (senha padrão é "ADMIN") e os membros do grupo ADMIN são os únicos usuários que podem acessar ctstop, o utilitário para parar o servidor c-treeACE, assim parar o servidor c-treeACE é sempre uma importante Responsabilidade do administrador.

ANEXO II – SQL

DML - Data Manipulation Language

Linguagem de Manipulação de Dados é utilizado para realizar consultas, alterações, inclusões e exclusões de dados presentes em registros. Estas tarefas podem ser executadas em vários registros de diversas tabelas ao mesmo tempo. Os comandos são Insert, Select, Update e Delete:

INSERT insere um registro a uma tabela existente.

UPDATE atualiza os valores de dados em uma ou mais linhas da tabela existente.

DELETE remove linhas existentes de uma tabela.

DDL - Data Definition Language

Linguagem de Definição de Dados permite criar tabelas novas e associações. Os comandos são:

CREATE: criar objeto dentro da base de dados.

DROP: apagar objeto do banco de dados.

ALTER: alterar um objeto.

DTL – Data Transaction Language

Linguagem de Transação de Dados, método que garante a persistência do dado em disco.

BEGIN WORK marca o começo de uma transação de banco de dados.

COMMIT finaliza uma transação, gerenciamento de banco de dados.

ROLLBACK descarta a transação.

DQL – Data Query Language

Linguagem de Consulta de Dados, permite uma consulta do resultado.

SELECT realizar consultas a dados pertencentes a uma tabela.

ANEXO III – TCP/IP

O TCP/IP (também chamado de pilha de protocolos TCP/IP) é um conjunto de protocolos de comunicação entre computadores em rede. Seu nome vem de dois protocolos: o TCP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão) e o IP (*Internet Protocol* - Protocolo de rede, ou ainda, protocolo de interconexão). O conjunto de protocolos pode ser visto como um modelo de camadas (Modelo OSI), onde cada camada é responsável por um grupo de tarefas, fornecendo um conjunto de serviços bem definidos para o protocolo da camada superior. As camadas mais altas, estão logicamente mais perto do usuário (chamada camada de aplicação) e lidam com dados mais abstratos, confiando em protocolos de camadas mais baixas para tarefas de menor nível de abstração.

Em 1983, a Organização de Normas Internacionais (ISO) desenvolveu um modelo de rede chamado Modelo de Referência de Interconexão de Sistemas Abertos (OSI), que definiu um quadro de comunicações de computador. O modelo de referência ISO / OSI (modelo ISO / OSI) possui sete camadas, cada uma das quais tem um nível de abstração diferente e executa uma função bem definida. O modelo ISO / OSI requer que a função de cada camada defina os protocolos de rede padronizados internacionais. As sete camadas são físico, ligação de dados, rede, transporte, sessão, apresentação e camadas de aplicação.

- Camada física transmite fisicamente sinais através de um meio de comunicação.
- Camada de enlace de dados transforma um fluxo de bits brutos (0s e 1s) da camada física em um quadro de dados sem erros para a camada de rede.
- Camada de rede controla a operação de um pacote transmitido de uma rede para outra, como por exemplo, como rotear um pacote.
- Camada de transporte divide dados da camada de sessão em pacotes menores para entrega na camada de rede e garante que os pacotes cheguem corretamente na outra extremidade.

- Camada de sessão estabelece e gerencia sessões, conversões ou diálogos entre dois computadores.
- Camada de apresentação gerencia a sintaxe e a semântica da informação transmitida entre dois computadores.
- Camada de aplicação, a camada mais alta, contém uma variedade de protocolos comumente usados, como transferência de arquivos, terminal virtual e e-mail.

O Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) desenvolveu um conjunto de padrões LAN, conhecido como IEEE Project 802, que o ISO aceitou como padrões internacionais. Os padrões IEEE LAN abordavam somente as duas camadas mais baixas, as camadas físicas e de enlace de dados, do modelo ISO / OSI.

O IEEE dividiu a camada de enlace de dados em duas subcamadas, as subcamadas de Controle de Ligação Lógico (LLC) e de Controle de Acesso Médio (MAC). A subcamada LLC, conhecida como padrão IEEE 802.2, é responsável pelas funções de link de dados que são independentes do meio subjacente. A subcamada MAC é responsável pelas funções de link de dados que dependem do meio da implementação da LAN. A implementação da LAN inclui ARCnet, Ethernet, Fast Ethernet, Token Bus, Token Ring e FDDI, que estão em conformidade com IEEE 802.2.

A Especificação de Interface de Dispositivo de Rede (NDIS) da Microsoft e a Open Data-Link Interface (ODI) da Novell são duas boas implementações das subcamadas LLC e MAC. A camada física nos padrões IEEE LAN é uma NIC física, como um adaptador Ethernet. Cada NIC tem um endereço único de 48 ou 16 bits, conhecido como hardware ou endereço MAC, para se identificar ou ser identificado para transmissão de dados nas duas camadas mais baixas.

TCP / IP é um conjunto de protocolos que o Departamento de Defesa dos EUA desenvolveu no início ARPANET em 1969. TCP / IP tem crescido muito além do projeto inicial. É o protocolo padrão na Internet e é o protocolo de rede mais utilizado atualmente.

IP implementa a função da camada de rede. Os principais protocolos em IP incluem o Protocolo de Resolução de Endereços (ARP), o Protocolo de Resolução de

Endereços Reversos (RARP), o ICMP (Internet Control Message Protocol) e o IGMP (Internet Group Management Protocol). ARP define como resolver um endereço IP de um host para um endereço de hardware; O RARP define como obter um endereço IP usando o endereço de hardware de um host; O ICMP define como se comunicar entre roteadores e hosts; E IGMP define como implementar multicast.

TCP é uma das duas implementações da camada de transporte em uma rede TCP / IP. A outra implementação é User Datagram Protocol (UDP). O TCP fornece entrega confiável e garantida de dados de um computador para outro, como um fax, e o UDP fornece apenas a entrega com o melhor esforço, semelhante ao correio normal.

Um aplicativo TCP / IP se encaixa nas três camadas superiores do modelo ISO / OSI, as camadas de sessão, apresentação e aplicativo. Os aplicativos TCP / IP mais usados incluem Telnet e ftp.