



Centro Universitário

CENTRO UNIVERSITÁRIO  
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA - IESB  
ENGENHARIA DE COMPUTAÇÃO

**ÁLVARO MATHEUS ANDRIOLI**

**PLATAFORMA PARA SÍNTESE E ANÁLISE DE CONTROLADORES DINÂMICOS**

Brasília - DF  
2017

**ÁLVARO MATHEUS ANDRIOLI**

**PLATAFORMA PARA SÍNTESE E ANÁLISE DE CONTROLADORES DINÂMICOS**

Trabalho de Conclusão de Curso  
apresentado ao curso de Engenharia de  
Computação do Instituto de Educação  
Superior de Brasília, como requisito parcial  
para obtenção do grau de Bacharel em  
Engenharia de Computação.

Orientador: Prof. Dr. Roberto de Souza  
Baptista

Brasília - DF  
2017

Andrioli, Álvaro Matheus.

Plataforma para síntese e análise de controladores dinâmicos / Álvaro Matheus Andrioli; Orientador Roberto Baptista. - Brasília, 2017;

68 p.:il. Color.; 30 cm.

Trabalho de Conclusão de Curso (Graduação - Engenharia de Computação) - Instituto de Educação Superior de Brasília, 2017.

1. Plataforma. 2. Teoria de controle. 3. Controle automático. I. Título. II. Baptista, Roberto.

**ÁLVARO MATHEUS ANDRIOLI**

**PLATAFORMA PARA SÍNTESE E ANÁLISE DE CONTROLADORES DINÂMICOS**

Trabalho de Conclusão de Curso aprovado  
pela Banca Examinadora com vistas à  
obtenção do título de Bacharel em  
Engenharia de Computação do Instituto de  
Educação Superior de Brasília.  
Brasília, DF 20 de Junho de 2017.

Banca Examinadora:

---

Prof. Dr. Roberto de Souza Baptista - orientador

---

Prof. Dr. Thiago Raposo Milhomem de Carvalho

---

Prof. Msc. Luís Gustavo Aquino de Carvalho

Brasília - DF  
2017

## RESUMO

O projeto consiste na construção de uma plataforma para teste de controle automático e suas respectivas análises sobre desempenho e robustez; O sistema de controle testado no projeto foi o controle proporcional; A plataforma possui um grau de liberdade simulando o movimento de giro de um *quadrotor*, foi construída utilizando matérias simples como madeira e perfil retangular de alumínio para baratear o custo de produção; Serão explicados no decorrer deste texto quais paradigmas e padrões de projetos foram utilizados na construção do software e firmware. As técnicas de análise de sistemas como o lugar geométrico das raízes, modelo matemático, e processo de identificação do sistema; Como resultado do projeto foi obtido um modelo matemático estimado por técnicas de identificação de sistemas e também foram feitos ensaios obtendo-se uma resposta ao degrau do sistema com uma variação do ganho até o sistema estar em completa oscilação; Os resultados obtidos com uma resposta ao impulso do sistema foram insatisfatórios, pois a rápida transição do sinal não é suficiente para tirar os rotores da inércia; A motivação para a realização deste projeto vem da importância da teoria de controle sobre nosso cenário atual de tecnologias e sistemas embarcados; No mercado brasileiro há pouco ou quase nenhuma plataforma educacional para testes de controle, assim foi decidido a construção de tal para a análise posterior dos resultados de tais técnicas de controle.

**Palavras-chave:** Teoria de Controle, Plataforma, Controle Automático.

## **ABSTRACT**

This Project consists in the construction of a platform for automatic control and its analysis regarding performance and robustness; The control system tested in the project was proportional; This platform has one degree of freedom, simulating a roll motion of quadcopter, was constructed using simple materials like wood and rectangular aluminum profile to reduce the cost of production; This text will explain which paradigms and design patterns were employed in the constructions of the software and firmware; The techniques of systems analysis, such as the roots of locus method, mathematical model, and system identification process. Techniques of identification of systems and also tests were made obtaining a step response of the system with a variation of the gain until the system is in complete oscillation state; The results obtained with a system impulse response were unsatisfactory because the rapid transition of the signal is not enough to take the rotors out of inertia; As a result of the project was obtained a mathematical model estimated by techniques of identification of systems and also tests were made obtaining a response to the step of the system with a variation of the gain until the system is in complete oscillation; The results obtained with a system impulse response were unsatisfactory because the rapid transition of the signal is not enough to take the rotors out of inertia.

**Key words:** Theory of control, Platform, Automatic Control.

## LISTA DE FIGURAS

<i>Figura 1: Plataforma quanser aero.....</i>	10
<i>Figura 2: Arduino uno.....</i>	12
<i>Figura 3: Ilustração e esquemático do potenciômetro.....</i>	13
<i>Figura 4: Ligação entre o Arduino e o ESC.....</i>	14
<i>Figura 5: ESC Red Brick 30 Amperes.....</i>	14
<i>Figura 6: Motor brushless Turnigy 1100 KV.....</i>	15
<i>Figura 7: Circuito gerador de PPM a partir do PWM.....</i>	16
<i>Figura 8: Modulação PWM vs PPM.....</i>	16
<i>Figura 9: Demodulação PPM.....</i>	17
<i>Figura 10: Amostragem por produto da função trem de impulsos.....</i>	18
<i>Figura 11: Amostragem ideal no domínio da frequência.....</i>	19
<i>Figura 12: Atitude de giro quadricóptero.....</i>	20
<i>Figura 13: Diagrama de corpo livre da plataforma.....</i>	21
<i>Figura 14: Diagrama de blocos do sistema de controle realimentado.....</i>	24
<i>Figura 15: Resposta característica de um sistema de segunda ordem subamortecido ao degrau.....</i>	24
<i>Figura 16: Exemplo de lugar de raízes.....</i>	26
<i>Figura 17: Tela principal da ferramenta de identificação de sistemas do matlab.....</i>	27
<i>Figura 18: Ciclo de vida de uma aplicação de página única (SPA).....</i>	30
<i>Figura 19: Ciclo de vida do websocket.....</i>	31
<i>Figura 20: Diagrama de comunicação entre o servidor, interface e Arduino.....</i>	33
<i>Figura 21: Esquema elétrico.....</i>	34
<i>Figura 22: Plataforma para simulação de atitude de giro de um quadricóptero.....</i>	35
<i>Figura 23: Resposta do sistema a um sinal binário aleatório.....</i>	36
<i>Figura 24: Ferramenta ident com os dados do sistema.....</i>	37
<i>Figura 25: Teste de assertividade das funções de transferência.....</i>	37
<i>Figura 26: Lugar de raízes da função de transferência estimada.....</i>	38
<i>Figura 27: Resposta ao degrado do sistema.....</i>	39
<i>Figura 28: Sistema de atualização de dados do AngularJS.....</i>	41
<i>Figura 29: Caso de uso do sistema.....</i>	42
<i>Figura 30: Diagrama de sequência do fluxo de configuração do controle.....</i>	43
<i>Figura 31: Diagrama de sequência do fluxo de inicialização do controle.....</i>	44
<i>Figura 32: Diagrama de sequência do fluxo de controle automático.....</i>	44
<i>Figura 33: Diagrama de sequência do fluxo de finalização do controle.....</i>	45
<i>Figura 34: Tela de visualização de sensores.....</i>	45
<i>Figura 35: Tela de visualização dos sensores com a visualização da câmera ativa.</i>	46
<i>Figura 36: Tela de configuração de controle proporcional.....</i>	47
<i>Figura 37: Tela de configuração de controle personalizado.....</i>	47
<i>Figura 38: Fluxo principal do programa.....</i>	49
<i>Figura 39: Exemplo de código para controle de um ESC utilizando a biblioteca Servo.h.....</i>	50
<i>Figura 40: Formato da string enviada para o servidor.....</i>	51
<i>Figura 41: Fluxograma da interrupção.....</i>	51
<i>Figura 42: Diagrama de execução de evento.....</i>	54
<i>Figura 43: Diagrama de classes do sistema de funções de controle.....</i>	55
<i>Figura 44: Resposta do sistema com ganho de 1,01.....</i>	57
<i>Figura 45: Resposta ao degrau com ganho 1.....</i>	58
<i>Figura 46: Resposta ao degrau com ganho 1,2.....</i>	58
<i>Figura 47: Resposta ao degrau com ganho de 1,4.....</i>	59

<i>Figura 48: Resposta ao degrau com ganho 1,6.....</i>	59
<i>Figura 49: Resposta ao degrau com ganho de 1,8.....</i>	60

## LISTA DE ABREVIATURAS E SIGLAS

AC	<i>Alternate Current</i> (Corrente Alternada).
BIBO	<i>Bounded-Input, Bounded-Output</i> (Entrada Limitada, Saída Limitada).
CSS	<i>Cascading Style Sheets</i> (Folha de Estilo em Cascata).
DC	<i>Direct Current</i> (Corrente Contínua).
DCL	Diagrama de Corpo Livre.
DOM	<i>Document Object Model</i> (Modelo de Objeto de Documento).
ESC	<i>Electronic Speed Control</i> (Controle de Velocidade Eletrônico).
HTML	<i>HyperText Markup Language</i> (Linguagem de Marcação de Hipertexto).
HTTP	<i>Hypertext Transfer Protocol</i> (Protocolo de Transferência de Hipertexto).
IDE	<i>Integrated Development Environment</i> (Ambiente de Desenvolvimento Integrado).
MVVM	Padrão de projeto Model-View-ViewModel.
PID	Controlador proporcional, integral, diferenciador.
POSIX	<i>Portable Operating System Interface</i> (Interface Portável Entre Sistemas Operacionais).
PPM	<i>Pulse Position Modulation</i> (Modulação por Posição de Pulso).
PWM	<i>Pulse Width Modulation</i> (Modulação por Largura de Pulso).
SPA	<i>Single Page Application</i> (Aplicação de Única Página).
SVG	<i>Scalable Vector Graphics</i> (Gráficos Vetoriais Escaláveis).
TCP	<i>Transmission Control Protocol</i> (Protocolo de Controle de Transmissão).
UDP	<i>User Datagram Protocol</i> (Protocolo de Datagramas de Utilizador).

## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>10</b>
<b>2 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>12</b>
<b>2.1 Arduino.....</b>	<b>12</b>
<b>2.2 Potenciômetro.....</b>	<b>13</b>
<b>2.3 ESC (<i>Electronic Speed Control</i>).....</b>	<b>13</b>
<b>2.4 Motor <i>Brushless</i>.....</b>	<b>14</b>
<b>2.5 PPM (<i>Pulse Position Modulation</i>).....</b>	<b>15</b>
2.5.1 Processo de modulação.....	16
2.5.2 Processo de demodulação.....	16
<b>2.6 Teoria da amostragem.....</b>	<b>17</b>
2.6.1 Amostragem ideal.....	17
2.6.1.1 <i>Função trem de impulsos</i> .....	18
<b>2.7 Modelagem matemática.....</b>	<b>19</b>
2.7.1 Diagrama de corpo livre.....	20
<b>2.8 Sistemas de controle realimentado.....</b>	<b>24</b>
2.8.1 Especificações no domínio do tempo.....	24
2.8.2 Lugar geométrico das raízes.....	25
2.8.3 Critério de estabilidade BIBO( <i>bounded input, bounded output</i> ).....	26
<b>2.9 Teoria e ferramentas para identificação de sistemas.....</b>	<b>26</b>
2.9.1 Ferramenta <i>ident matlab</i> .....	27
<b>2.10 Python.....</b>	<b>28</b>
2.10.1 PySerial.....	28
2.10.2 Flask.....	28
<b>2.11 Electron.....</b>	<b>29</b>
2.11.1 HTML.....	29
2.11.2 CSS.....	29
2.11.2.1 <i>Bootstrap</i> .....	29
2.11.3 JavaScript.....	30
2.11.3.1 <i>Angular JS</i> .....	30
2.11.3.2 <i>CanvasJs</i> .....	30
<b>2.12 Web Sockets.....</b>	<b>31</b>
2.12.1 <i>SocketIO</i> .....	32
<b>3 DESENVOLVIMENTO.....</b>	<b>33</b>
<b>3.1 Esquema elétrico.....</b>	<b>33</b>
<b>3.2 Construção mecânica.....</b>	<b>34</b>
<b>3.3 Identificação do sistema.....</b>	<b>35</b>
3.3.2 Usando a ferramenta <i>ident</i> .....	36
3.3.3 Lugar de raízes da função de transferência estimada.....	38
<b>3.4 Interface com o usuário.....</b>	<b>39</b>
3.4.1 Desenvolvimento de aplicativos híbridos.....	39
3.4.2 Electron.....	40
3.4.3 Frameworks utilizados para construção da tela.....	40
3.4.3.1 <i>AngularJs</i> .....	40
3.4.3.2 <i>Bootstrap</i> .....	41
3.4.3.3 <i>CanvasJs</i> .....	41
3.4.4 Caso de uso.....	42
3.4.5 Diagrama de sequência das funcionalidades.....	43

3.4.6 Telas do sistema.....	45
<b>3.5 Firmware.....</b>	<b>47</b>
3.5.1 Comunicação com o servidor.....	47
3.5.2 Programa principal.....	48
3.5.2.1 Geração de sinal PPM para controle dos motores.....	49
3.5.3 Interrupção de tempo.....	50
3.5.3.1 Timers do Arduino Uno.....	50
3.5.3.2 Fluxo de execução da interrupção de tempo.....	51
<b>3.6 Servidor.....</b>	<b>51</b>
3.6.1 Python.....	52
3.6.1.1 Flask SocketIO.....	52
3.6.2 Padrões de projetos.....	53
3.6.2.1 Singleton.....	53
3.6.2.2 Sistema baseado em eventos.....	53
3.6.2.3 Funções de controle.....	54
3.6.3 Compartilhamento da porta serial.....	55
<b>4 RESULTADOS.....</b>	<b>57</b>
<b>5 CONCLUSÕES.....</b>	<b>60</b>
<b>5.1 Próximos trabalhos.....</b>	<b>61</b>

## 1 INTRODUÇÃO

A teoria de controle automático tem desenvolvido um papel muito importante no nosso cotidiano, tornando processos de produção mais rápidos e precisos, assim barateando produtos e os fornecendo com maior qualidade.

O controle automático visa fazer o controle de uma ou mais variáveis, chamadas de variáveis controladas, através de uma referência, como, por exemplo, a velocidade angular de um motor elétrico de corrente contínua (DC) dado um sinal de tensão elétrica de referência.

A implementação de uma plataforma para teste de controle, onde esta é acessível e modular, é de grande importância visto o défice do mercado brasileiro destes tipos de produtos.

A planta a ser controlada no processo é uma plataforma com um grau de liberdade sendo está uma simulação da atitude de giro de um quadricóptero. O projeto mecânico foi inspirado pela plataforma de controle *aero* da quanser mostrado na figura 1



*Figura 1: Plataforma quanser aero.*

A planta foi escolhida por fornecer um grau razoável de dificuldade em sua modelagem e para o estímulo no estudo das técnicas de controle.

O projeto possui como objetivo primário a construção de uma plataforma para testes de controle automáticos, o desenvolvimento de uma interface modular que permita a implementação de mais de uma função de controle e visualização em tempo real dos sensores da planta, além da comunicação entre a plataforma e a interface do sistema. O

sistema de interface foi desenvolvido utilizando modelos de desenvolvimento híbrido, ou seja aplicativos nativos de plataformas com uma camada de tecnologia web.

Após a construção da plataforma será realizado teste com algumas funções de controle para a avaliação de performance e robustez.

A estrutura do trabalho consiste em um capítulo de fundamentação teórica sendo ele o capítulo 2, seguido do capítulo 3 sobre o desenvolvimento do projeto.

O capítulo 4 trata dos resultados alcançados no desenvolvimento do projeto e o capítulo 5 é constituído da conclusão geral do projeto e também de algumas possibilidades de trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os referenciais teóricos da construção da plataforma de teste de controle e a análise de algumas funções de controle automático. Do item 2.1 ao item 2.4, o referencial se trata dos componentes físicos utilizados na construção e os itens subsequentes são referidos ao sistema de software e comunicação entre os componentes.

### 2.1 Arduino

Arduino é uma plataforma de prototipagem de hardware livre que utiliza um microcontrolador Atmel AVR e uma IDE de desenvolvimento em uma linguagem própria (ARDUINO, *What is Arduino?*).

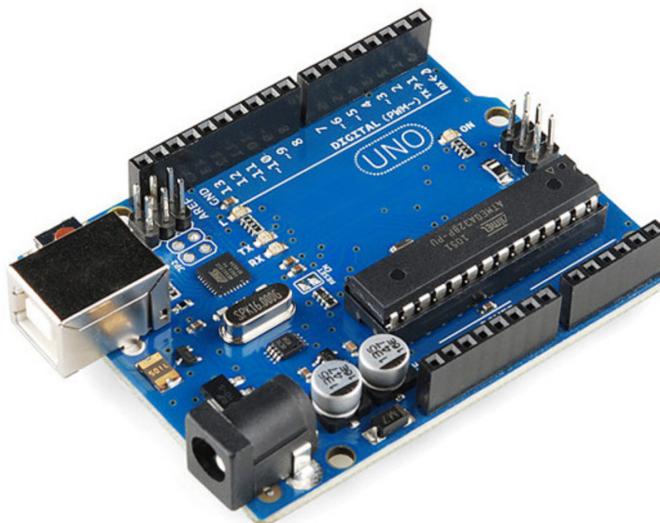


Figura 2: Arduino uno.

O Arduino tem se popularizado pela sua facilidade de uso, pois em sua placa já consta com um conversor USB RS232, circuito de alimentação e estabilização de entradas etc. Também é possível a adição de funcionalidades à placa ao encaixe de um módulo externo como o de *ethernet* assim dando a funcionalidade de comunicação TCP/IP não nativo ao Arduino.

O desenvolvimento do software do Arduino é feito em uma IDE própria com funções de alta ordem assim facilitando configurações e também tarefas mais árduas como a geração de múltiplos sinais PWM.

## 2.2 Potenciômetro

Como diz (MARKUS OTÁVIO, 2011), o potenciômetro é um componente eletrônico que fornece uma resistência ajustável através de uma angulação sobre uma haste. Este componente é semelhante a um resistor comum com o diferencial de ser ajustável.

O potenciômetro possui três terminais e um contato deslizante ligado a haste, assim girando a haste se modifica a distância entre um terminal e o contato deslizante alterando a resistência, pois a resistência é proporcional a distância.

Este componente tem papel fundamental no projeto sendo ele responsável pela medição do valor de referência e do ângulo de giro do sistema.

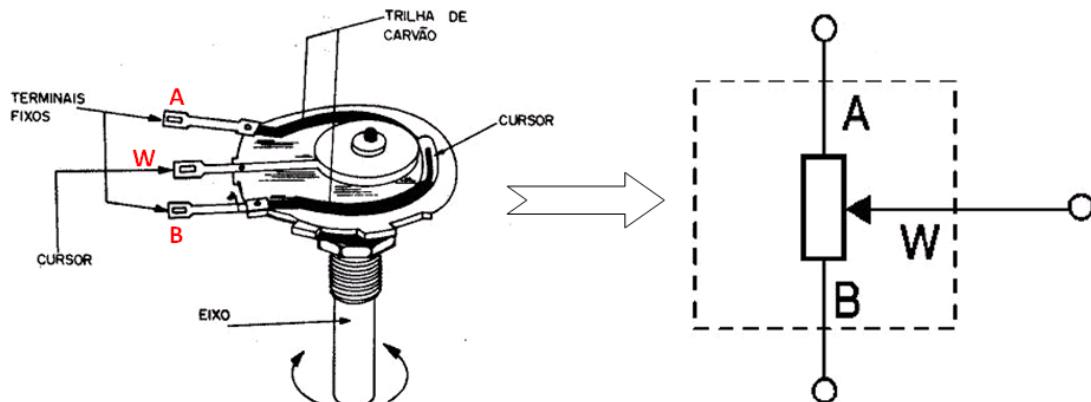


Figura 3: Ilustração e esquemático do potenciômetro.

## 2.3 ESC (*Electronic Speed Control*)

ESC (*Electronic Speed Control*) ou controle eletrônico de velocidade é um circuito eletrônico responsável pelo controle de velocidade do motor.

A controladora responsável por receber e transmitir os sinais de atuação não possui potência o suficiente para o devido acionamento de um motor, assim a controladora é ligada ao ESC e, assim, passando a informação de velocidade, o ESC é ligado a uma fonte externa que possui a capacidade de alimentar os motores. O ESC

recebe o sinal da controladora modulado em posição de pulso PPM e atuando sobre o motor.

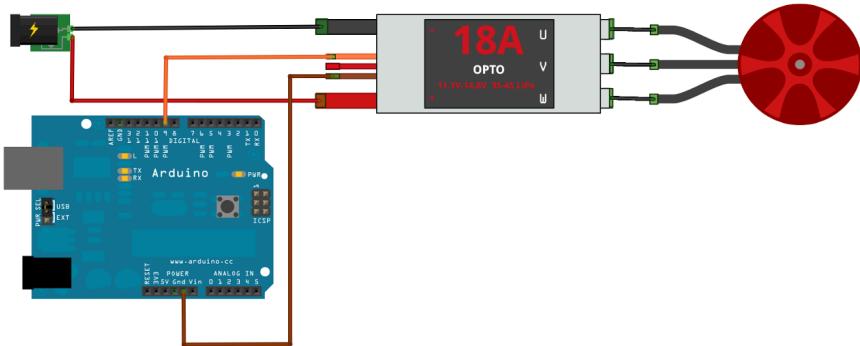


Figura 4: Ligação entre o Arduino e o ESC.

A figura 4 ilustra a ligação do ESC ao Arduino e ao motor utilizando uma fonte externa

O ESC utilizado na construção da plataforma foi o modelo da Red Brick 30A.

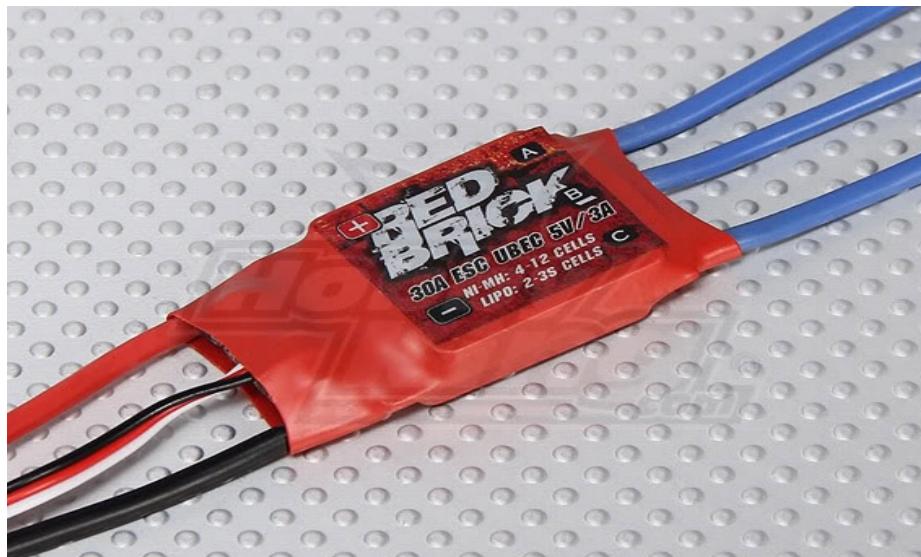


Figura 5: ESC Red Brick 30A.

## 2.4 Motor *Brushless*

Motores elétricos são equipamentos que convertem energia elétrica em energia mecânica. Existem alguns tipos de motores de corrente contínua (DC), aqui será citado dois tipos que são os de escova (*Brushed*) e os sem escovas (*Brushless*).

Motores de escova são mais simples permitem um grande controle de velocidade, posição, valor de construção menos elevado e etc.

Motores sem escovas são considerados a evolução de motores de corrente contínua. Seu funcionamento é bastante parecido com o de escova e se difere apenas que se utiliza um sensor de efeito *Hall* para identificar os polos dos ímãs e fazer a comutação da corrente nas bobinas.

Foram utilizados no projeto dois motores *brushless turnigy* 1100 KV. A tensão de funcionamento destes motores está entre 7,4 e 15 volts.



*Figura 6: Motor brushless Turnigy 1100 KV.*

## **2.5 PPM (Pulse Position Modulation)**

A modulação da posição de pulsos (PPM - pulse position modulation) consiste em posicionar um pulso retangular de amplitude e duração fixas dentro do intervalo de amostragem, de forma que a posição relativa seja proporcional ao sinal analógico.

A vantagem desta modulação reside no fato de que o formato de impulso é sempre o mesmo, facilitando a regeneração do sinal, além de consumir menor potência em comparação com a modulação PWM, pois a parte ativa do pulso sempre será de mesmo comprimento.

### 2.5.1 Processo de modulação

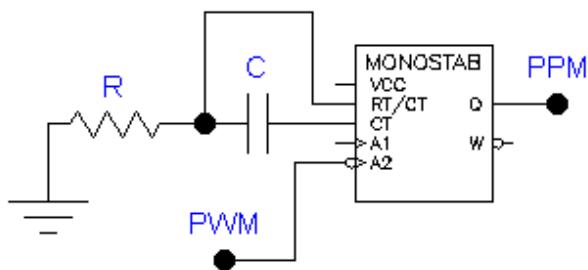


Figura 7: Circuito gerador de PPM a partir do PWM.

O processo de modulação em posição de pulso se dá gerando um sinal modulado por largura de pulso (PWM) e posteriormente utilizar um detector de bordas de subida ou descida e mantendo um pulso de duração constante. A seguir é ilustrado a diferença entre as modulações PWM e PPM.

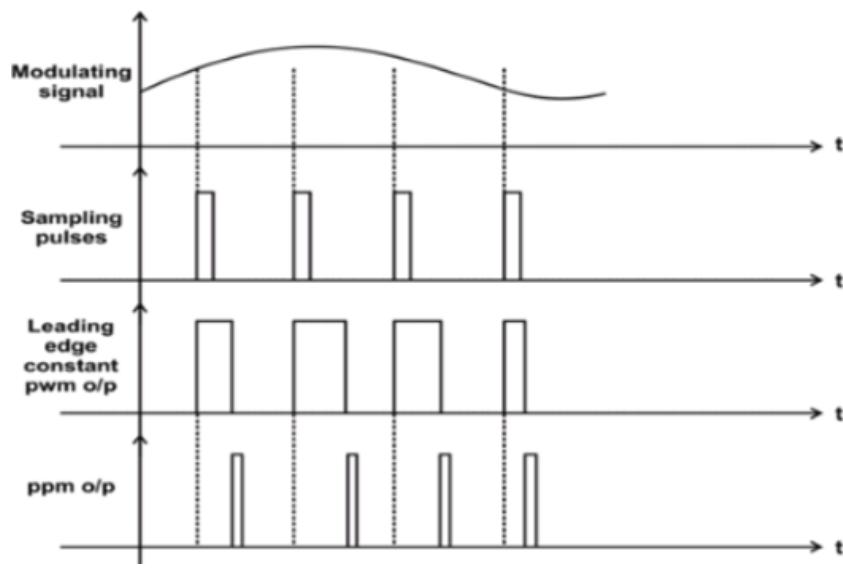


Figura 8: Modulação PWM vs PPM.

### 2.5.2 Processo de demodulação

O processo de demodulação do sinal PPM consistem em gerar um sinal PWM a partir do PPM. O sinal PPM é conectado a uma porta lógica juntamente com um oscilador local sincronizado, assim gerando o sinal PWM, e após esta conversão o sinal resultante será filtrado por um passa baixa e assim recuperando o sinal original.

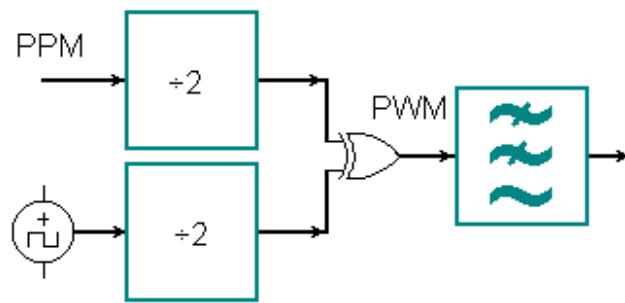


Figura 9: Demodulação PPM.

## 2.6 Teoria da amostragem

Em nosso mundo a maior parte dos sinais encontrados na natureza são sinais contínuos, ou seja, possuem uma precisão infinita, sendo necessário uma quantidade de memória infinita para representá-los de maneira exata.

Ao se trabalhar com sinais e sistemas contínuos em computadores digitais há a necessidade de converter um sinal contínuo em um sinal discreto, assim fazendo a amostragem do sinal (OGATA, 1995).

Um sinal variante no tempo pode ser amostrado em um intervalo de tempo  $T$ , formando assim uma sequência de valores discretos.

Existem várias maneiras de se obter a sequência discreta de um sinal contínuo, neste texto será abordada a técnica de amostragem ideal por produto.

### 2.6.1 Amostragem ideal

A amostragem ideal consiste em utilizar da função Delta de Dirac ou impulso, descrito na equação 1.

$$\int_{-\infty}^{\infty} \delta(t) dt = 1 \quad (1)$$

A função Delta de Dirac possui uma propriedade muito importante para a amostragem de sinais que se chama a propriedade da amostragem. Ao integrar o produto da função impulso com outra função  $f(t)$  qualquer no instante de tempo  $a$  terá como resultado aproximadamente a própria função  $f(t)$  no instante  $a$ , como descrito na equação 2.

$$\int_{-\infty}^{\infty} \delta(t - a)f(t)dt = f(a) \quad (2)$$

### 2.6.1.1 Função trem de impulsos

A função trem de impulsos consiste na somatória de infinitos impulsos em períodos de tempo  $T$ . O produto da função trem de impulsos com uma função  $f(t)$  qualquer obtém-se como resultado o sinal  $f[n]$  amostrado como mostrado na figura 10.

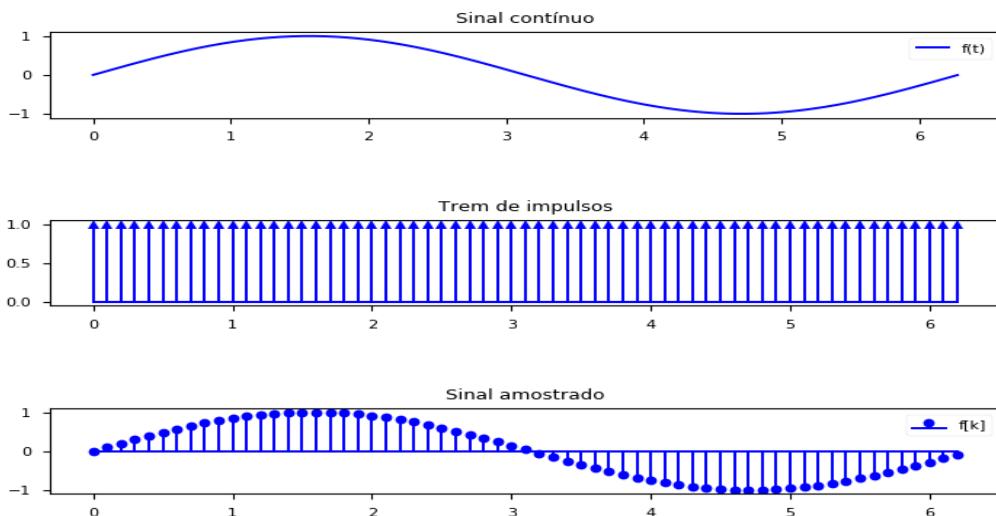


Figura 10: Amostragem por produto da função trem de impulsos.

O sinal amostrado é representado pela símbolo \* e está descrito pela equação

$$f[k] = f(kT)^* = \sum_{k=-\infty}^{\infty} \delta(t - kT)f(kT) \quad (3)$$

No processo de amostragem descrito na equação 3 replica-se o espectro em frequência do sinal original em períodos de frequência igual a  $f_s = 1/T_s$  onde  $T_s$  é o período de amostragem. O processo genérico de amostragem está ilustrado na figura 11.

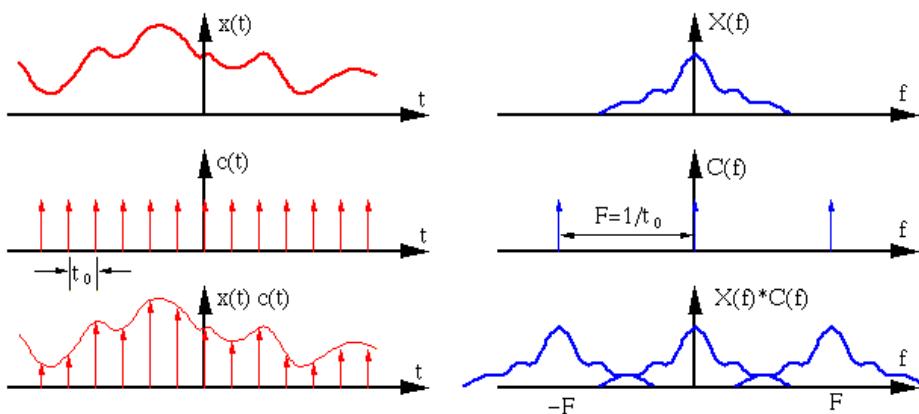


Figura 11: Amostragem ideal no domínio da frequência.

Onde

$x(t)$  e  $X(f)$  Sinal original e seu espectro de frequência respectivamente.

$c(t)$  e  $C(f)$  Trem de impulsos e seu espectro em frequência respectivamente.

$x(t) * c(t)$  e  $X(f) * C(f)$  Sinal amostrado e seu espectro de frequência respectivamente.

Ao se replicar o espectro do sinal no domínio da frequência há a possibilidade de se sobrepor as altas frequências como mostra a figura 11. Para isso ser evitado sempre se deve amostrar o sinal a uma taxa pelo menos duas vezes maior a que a frequência máxima presente no sinal, isto para sinais limitados em largura de banda (DINIZ; SILVA; NETTO, 2004).

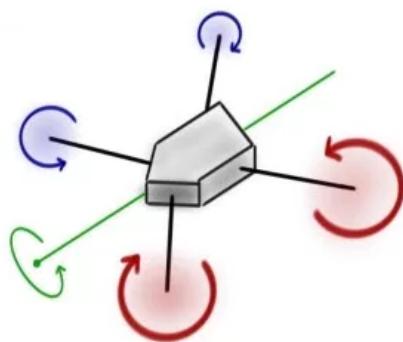
Para sinais com o espectro de frequência infinito utiliza-se um filtro passa baixa com frequência de corte igual a metade da frequência de amostragem, assim garantindo que a sobreposição ou efeito de *aliasing* não ocorra. O critério de utilização da frequência de amostragem pelo menos duas vezes maior que a maior frequência presente no sinal é chamado de critério de Nyquist . O filtro passa baixa aplicado a este contexto é denominado *anti-aliasing*.

## 2.7 Modelagem matemática

Um sistema é uma combinação de componentes que trabalham juntos para atingir um objetivo comum. Na teoria de controle a modelagem matemática de um sistema é de suma importância para prever seu comportamento (OGATA, 2010).

A dinâmica de vários sistemas é descrita em termos de equações diferenciais que são obtidas a partir de leis da física, como as leis de Newton para a mecânica e as leis de Kirchhoff para sistemas elétricos (OGATA, 2010).

O sistema a ser modelado é o movimento de giro de um quadricóptero exemplificado na ilustração 12. O protótipo construído possui dois rotores presos a uma haste onde geram empuxo que movimentam a haste em torno do seu ponto de fixação.



*Figura 12: Atitude de giro quadricóptero.*

### 2.7.1 Diagrama de corpo livre

O diagrama de corpo livre(DCL) é uma representação gráfica para a visualização das forças, movimentos aplicados a um corpo e assim podendo observar a força resultante obtida. Este diagrama retrata um corpo conectado com todas as forças e momentos bem como reações que agem sobre este, assim facilitando a resolução de problemas complexos (HALLIDAY; et al, 2016).

O diagrama de corpo livre apresentado na imagem 13 é referente à plataforma construída para a simulação do movimento de giro do quadricóptero.

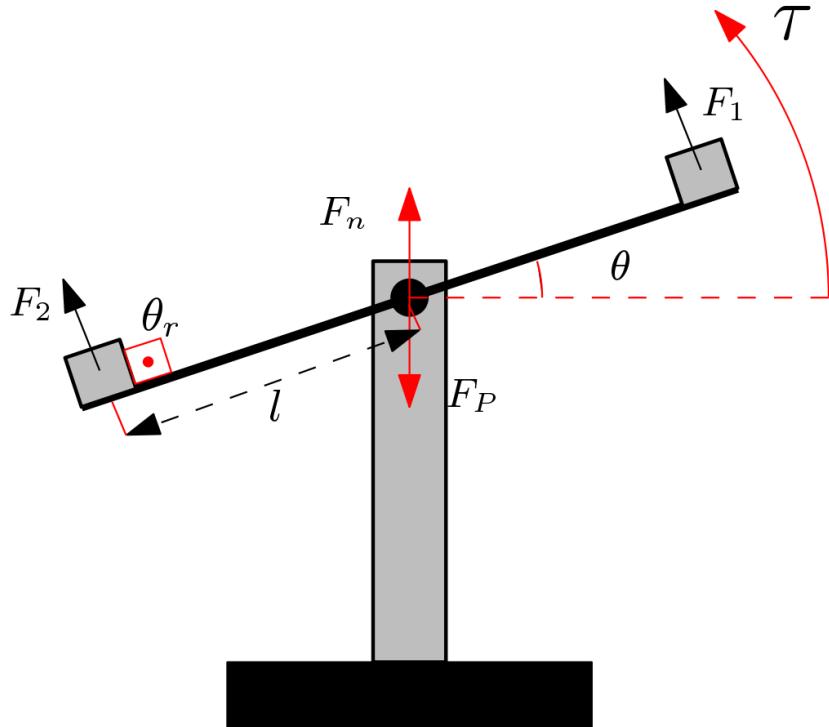


Figura 13: Diagrama de corpo livre da plataforma.

A partir do diagrama pode-se estimar a equação diferencial que rege o sistema. Por se tratar de um sistema rotacional mecânico utilizam-se as equações de Newton, citada abaixo, para o movimento rotacional.

$$\sum \tau = I\alpha \quad (4)$$

Sendo

$\tau$  O torque aplicado ao objeto.

$I$  O momento de inercia resultante do objeto.

$\alpha$  Aceleração do objeto.

A força peso  $F_p$  é aplicado exatamente sobre o ponto de apoio do eixo de giro assim obtendo-se uma força igual e contrário chamada força normal  $F_n$ . O efeito resultante da força normal com a força peso é igual a zero  $F_p - F_n = 0$ , pois o sistema mecânico de giro é simétrico.

A força gerada pelo rotor depende de algumas características complexas de se modelar, por este motivo foi escolhido usar um modelo linearizado.

Basicamente a hélice em rotação gera uma diferença de pressão da parte superior com a parte inferior e assim surgindo o empuxo  $F_1$  e  $F_2$ . A diferença de pressão gerada é proporcional ao quadrado da velocidade angular sendo mostrada na equação 5.

$$P = \omega^2 k_p \quad (5)$$

Onde

$k_p$  Constante de proporcionalidade de pressão por velocidade angular.

A velocidade angular do rotor de um motor de corrente contínua é diretamente proporcional ao a tensão aplicada ao motor, sendo representado pela equação 6.

$$\omega = \frac{E}{k_e} \quad (6)$$

Onde

$E$  Tensão sobre o motor.

$k_e$  Constante de proporcionalidade de tensão por velocidade angular.

E por fim a última relação necessária para a construção da equação diferencial do sistema descrita na próxima equação, onde se relaciona a pressão com a força.

$$F = AP \quad (7)$$

Sendo

$A$  Área.

Então a força de empuxo gerada por um rotor depende da área  $A$  da hélice, da constante de proporcionalidade de pressão  $k_p$ , da constante de proporcionalidade da tensão  $k_e$  e da tensão  $E$  aplicada ao rotor. Assim a equação dos empuxos do sistema é:

$$F = A \left( \frac{E}{k_e} \right)^2 k_p \quad (8)$$

ou

$$F = \frac{Ak_p E^2}{k_e^2} \quad (9)$$

Utilizando a equação de Newton a descrição do sistema será:

$$\tau = I * \alpha$$

$$F_1 l \sin(\theta_r) - F_2 l \sin(\theta_r) = I\alpha$$

$$(F_1 - F_2) \sin(90)l = I\alpha$$

$$\frac{Ak_p E_1^2}{k_e} - \frac{Ak_p E_2^2}{k_e} l = I\alpha$$

$$\frac{Ak_p}{k_e^2} l(E_1^2 - E_2^2) = I\alpha \quad (10)$$

Sendo

$l$  A distância do centro do rotor até o ponto de apoio da haste.

$\theta_r$  O ângulo entre o rotor e a haste.

Considerando a equação 10 e substituindo  $E_1^2 - E_2^2$  por  $x$  e  $\theta$  por  $y$  obtém-se a equação 11.

$$\frac{Ak_p}{k_e^2} lx = I \frac{d^2y}{dt^2} \quad (11)$$

Aplicando a transformada de Laplace e isolando os valores, será obtido a função de transferência do sistema.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{Ak_p l}{k_e^2 I s^2} \quad (12)$$

Como visto na equação 12 o sistema é modelado por uma função de transferência de segunda ordem.

## 2.8 Sistemas de controle realimentado

Um sistema de controle realimentado é um sistema que estabeleça a relação de comparação da entrada de referência com a saída utilizando a diferença como meio de controle (OGATA, 2010).

O sinal de erro atuante, que é a diferença entre a entrada e a saída do sistema alimenta a função de controle de modo a minimizar o erro e acertar a saída do sistema ao valor desejado. A figura a 14 mostra o diagrama de blocos de um sistema de controle com realimentação.

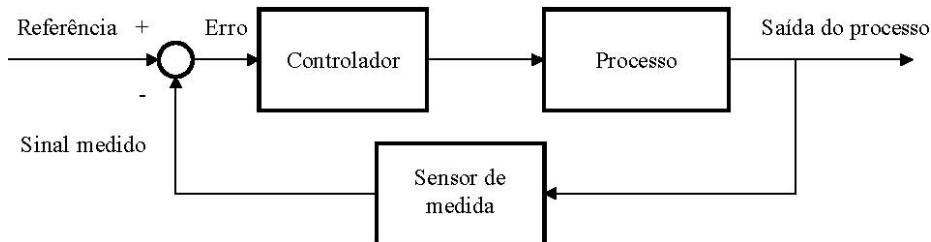


Figura 14: Diagrama de blocos do sistema de controle realimentado.

### 2.8.1 Especificações no domínio do tempo

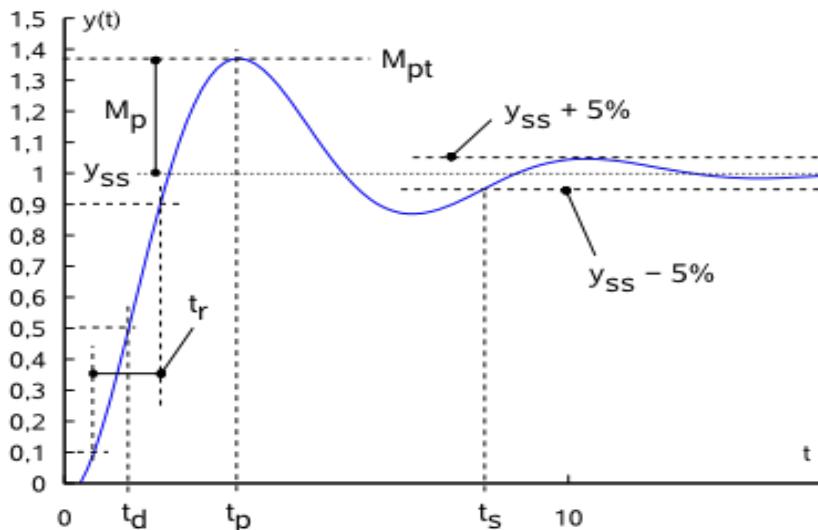


Figura 15: Resposta característica de um sistema de segunda ordem subamortecido ao degrau.

Ao se aplicar à entrada do sistema de segunda ordem um sinal do tipo degrau unitário obtém-se uma resposta característica. Esta resposta possui um conjunto de parâmetros de desempenho, tais como tempo de subida  $t_r$ , tempo de estabilização  $t_s$ , percentual de sobre-impulso  $m_p$  e etc.

Uma das métricas mais relevantes para a análise é o tempo de subida, que representa o intervalo de tempo para o sistema sair de 10% a 90% do valor final da resposta.

Tempo de estabilização, que representa o espaço de tempo necessário para o sistema atingir a resposta em estado permanente.

Percentual de sobre-impulso  $m_p$ , que é o valor máximo alcançado pela saída do sistema antes de se estabilizar.

E por último o erro de estado permanente, que é a diferença entre o valor alcançado pelo sistema e o desejado mostrado no gráfico como  $y_{ss}$ .

### 2.8.2 Lugar geométrico das raízes

Considerando o sistema representando pela figura 14, onde a função de controle é  $K$ , o processo  $G(s)$  e com a função de transferência do sensor igual a 1 obtém-se a seguinte função de transferência em malha fechada.

$$\frac{Y(s)}{X(s)} = H(s) = \frac{KG(s)}{1 + KG(s)} \quad (13)$$

Ao se igualar o denominador da equação 13 a zero se obtém a equação conhecida como equação característica, onde é responsável pelo comportamento do sistema. As raízes do numerador da equação característica são conhecidos como zeros do sistema e as raízes do denominador são chamados de polos do sistema.

Ao se tirar o módulo da equação característica se observa que o módulo do produto da função de transferência com o ganho deve obedecer a relação mostrada na equação 14, assim é possível ajustar o ganho para que os polos e zeros do sistema se movimentem no plano complexo para estabilizar e melhorar as características da resposta ao sistema.

$$|KG(s)| = |-1| \quad (14)$$

O gráfico ilustrado na figura 16, mostra o comportamento dos polos e zeros da equação característica ao se alterar o ganho  $k$  do sistema. O caminho feito pelos polos e zeros ao alterar o ganho do sistema é chamado de lugar geométrico das raízes.

Na ilustração 16 os polos e zeros mostrados em vermelho possuem  $k$  igual a zero e os em azul  $k$  maior que zero.

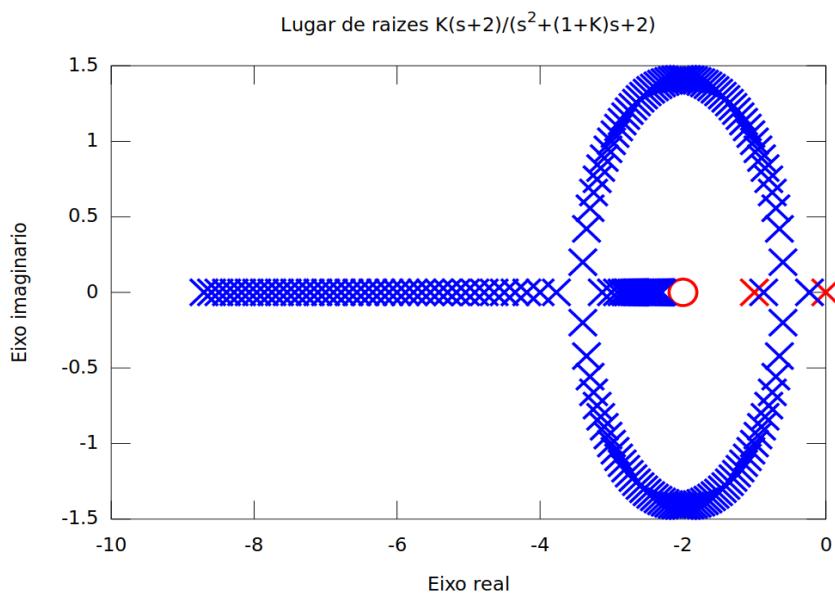


Figura 16: Exemplo de lugar de raízes.

### 2.8.3 Critério de estabilidade BIBO(*bounded input, bounded output*)

Para o sistema possuir a estabilidade BIBO ao se aplicar uma entrada limitada em amplitude no sistema a saída obrigatoriamente será também, limitada em amplitude (DINIZ; SILVA; NETTO, 2004).

Para qualquer sistema possuir a estabilidade BIBO todos os polos da equação característica devem estar ao lado esquerdo do plano complexo.

## 2.9 Teoria e ferramentas para identificação de sistemas

A identificação de sistemas é um conjunto de ferramentas matemáticas e algorítmicas que permite construir modelos de sistemas dinâmicos a partir de dados medidos (LJUNG, 2012).

Muitas das vezes descrever um modelo de um sistema dinâmico de maneira direta é difícil ou até inviável por sua complexidade inerte a processos reais.

Uma alternativa a este modelo seria encontrar as variáveis que afetam significativamente sistema (entradas) e medir todas as variáveis que descrevem o sistema (saídas) de interesse e, então, identificar a relação entre as entradas e saídas, sem entrar em muitos detalhes.

Para se obter uma estimativa do modelo é necessário aplicar à planta um sinal controlado de média zero onde este não irá tirar a planta do estado de equilíbrio e medir sua saída, assim pode-se utilizar algum método de obtenção de parâmetros através de um modelo pré-determinado.

### 2.9.1 Ferramenta *ident matlab*

A caixa de ferramenta *ident* do *matlab* serve para a identificação de sistemas onde se têm as entradas e as saídas medidas, assim obtendo um modelo matemático de maneira simples.

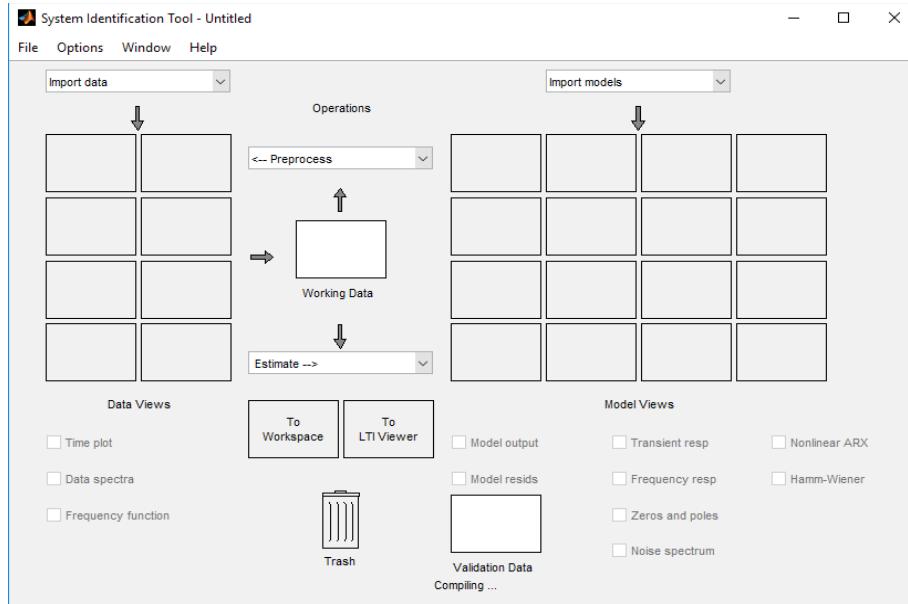


Figura 17: Tela principal da ferramenta de identificação de sistemas do *matlab*.

A tela principal da ferramenta de identificação de sistemas do *matlab* está ilustrada na figura 17. Nesta tela é possível carregar entradas e saídas de medições e estimar modelos que permitam vincular tais entradas e saídas assim determinando o sistema.

Ao lado esquerdo da tela estão as entradas e saídas vinculadas e ao lado direito está os modelos preditivos estimados pelo sistema.

Há várias maneiras de se importar os dados para a ferramenta de identificação, tais como importação de dados no domínio da frequência ou do tempo, representação em espaços de estados etc.

## 2.10 Python

Python é uma linguagem de programação criada por *Guido van Rossum* em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade, em outras palavras, Python é uma linguagem que foi criada para produzir códigos de bons padrões, fácil de manter de maneira e de maneira rápida (*PYTHON SOFTWARE FOUNDATION. Python 3.6.3 documentation*).

Python tem uma biblioteca padrão imensa, que contém classes, métodos e funções para auxílio em diversas tarefas, desde acesso a bancos de dados a interfaces gráficas com o usuário. Python possui muitas ferramentas para lidar com dados científicos.

O suporte à linguagem Python é nativa em praticamente todos os sistemas operacionais baseados em *Linux*, assim sendo uma ótima opção para ser multiplataforma. Como uma das principais características da linguagem é uma grande comunidade ativa que produz bibliotecas e principalmente, à linguagem ser de código aberto.

### 2.10.1 PySerial

PySerial é um módulo Python que fornece acesso a comunicação serial. O módulo é multiplataforma tendo suporte aos principais sistemas operacionais do mercado: Windows, OSX, Linux, BSD e qualquer sistema baseado no POSIX (*Welcome to pySerial's*).

### 2.10.2 Flask

Flask é um pequeno framework web escrito em Python, tem a flexibilidade da linguagem de programação Python e provê um modelo simples para desenvolvimento web. Uma vez importando no Python, Flask pode ser usado para economizar tempo construindo aplicações web (*ARMIN RONACHER; et al. Welcome to Flask*).

Flask é chamado de microframework por sua estrutura de núcleo sucinta fornecendo apenas as funcionalidades principais sendo possível incrementar suas funcionalidades com módulos separados.

## 2.11 Electron

Electron é um framework de código aberto desenvolvido pela equipe do GitHub para a construção de aplicativos de desktop multiplataforma com tecnologias web: HTML, CSS e JavaScript. O Electron utiliza o visualizador web Chromium e a tecnologia NodeJS para sua execução (GITHUB. *Electron Documentation*).

### 2.11.1 HTML

*Hypertext markup language* (HTML) ou linguagem de marcação de hipertexto é a linguagem padrão para construção da estrutura das páginas web. Ela fornece estruturas como cabeçalhos, parágrafos, títulos, tabelas, listas e etc (W3C. *HTML5*).

O HTML utiliza notação de tags para a marcação do conteúdo do site. Tags são nomes de estruturas rodeados por colchetes.

### 2.11.2 CSS

Cascading Style Sheets (CSS) é uma "folha de estilo" composta por camadas e utilizada para definir a aparência em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação como HTML.

O CSS define como serão exibidos os elementos contidos no código de uma página da internet e sua maior vantagem é efetuar a separação entre o formato e o conteúdo de um documento assim não tendo a necessidade de modificar a estrutura do documento para se alterar a forma como ele é apresentado (MOZILLA; et al. *CSS basics*).

#### 2.11.2.1 Bootstrap

Bootstrap é um framework CSS e HTML desenvolvido pela equipe do twitter que fornece vários sistemas e estruturas padrões para a construção de páginas web. A

principal vantagem em utilizar o bootstrap é a responsividade adquirida para o sistema (MARK; JACON; et all. *Bootstrap Introduction*).

O conceito de responsividade se da pela adaptação da visualização do sistemas em diferentes navegadores e também em diferentes tamanhos de telas, desde telas de dispositivos móveis até televisores, sem a perca de funcionalidade e qualidade de visualização do conteúdo.

### 2.11.3 JavaScript

JavaScript é uma linguagem de programação dinâmica, interpretada e com o característica de programação funcional e paradigma de orientação a objetos utilizada principalmente nas construção de dinamismo nas páginas web.

Atualmente o interpretador da linguagem, v8 engine, trouxe grandes avanços e um dos mais significativos foi a possibilidade de execução fora de um browser e assim foi criado o NodeJS.

#### 2.11.3.1 Angular JS

Angular JS é um *framework* de desenvolvimento de *front-end* com o objetivo de conceder dinamismo as páginas estáticas criadas com o HTML e CSS. Ele foi criado e é mantido pela equipe da Google (GOOGLE. *AngularJS API Docs*).

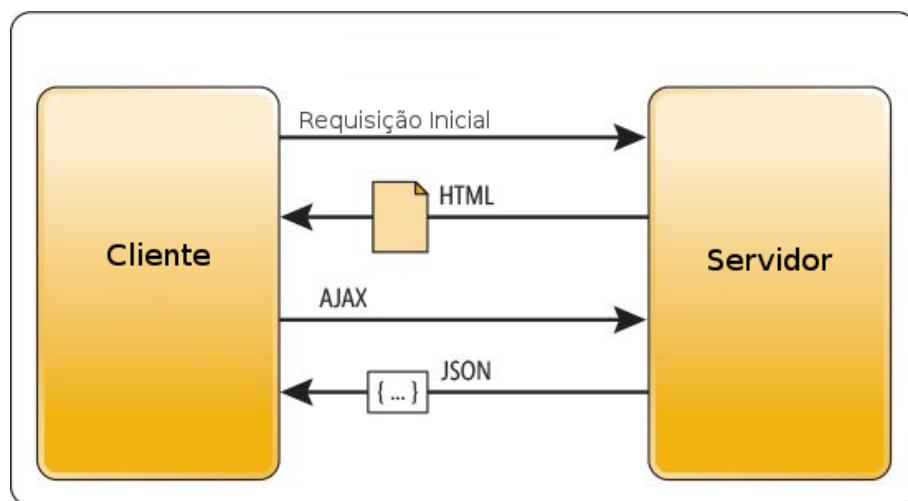


Figura 18: Ciclo de vida de uma aplicação de página única (SPA).

Angular foi desenvolvido para single page application (SPA) ou seja, todo o sistema é constituído de uma única página sendo carregado cada parte separadamente por demanda, assim deixando o sistema mais performático, o ciclo de vida de um sistema em SPA é mostrado na figura 18.

#### 2.11.3.2 CanvasJs

CanvasJS é uma API para geração de gráficos em navegadores que utiliza a tecnologia de tela de pintura (canvas) inclusa no HTML5 melhorando em até dez vezes a performance e sendo portável para vários tipos de plataforma diferente das outras API's gráficas que utilizam imagens geradas por *Scalable Vector Graphics* (SVG) ou na plataforma de multimídia *Flash* (FENOPIX. *Introduction to CanvasJS JavaScript Charts*).

### 2.12 Web Sockets

Um Socket é a abstração das primeiras camadas dos protocolos TCP ou UDP para reduzir a complexidade de comunicação entre processos pela rede de dados. O socket transmite dados em formato binário, em seguida é mostrado o ciclo de vida de um WebSocket.

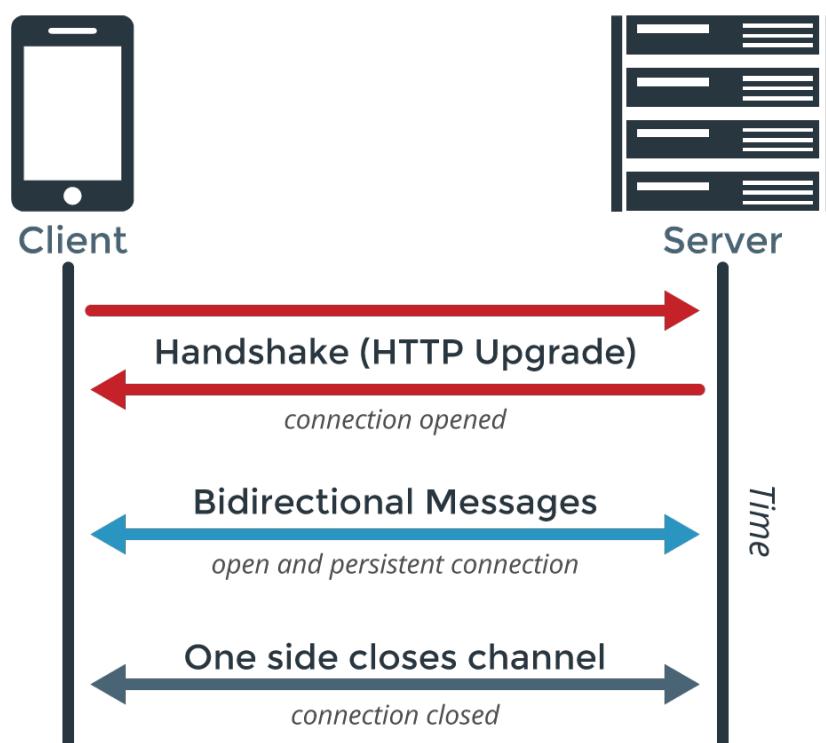


Figura 19: Ciclo de vida do websocket.

O protocolo WebSocket, por sua vez, possui a finalidade muito mais específica de ser implementado em navegadores web para comunicação persistente e bidirecional entre o código JavaScript com o servidor da aplicação web, comunicando-se sempre sobre o protocolo TCP.

O formato do protocolo é em texto e o *handshake* que inicia a comunicação é muito parecido com o HTTP, assim servidores web podem facilmente servir HTTP e WebSockets na mesma porta.

### 2.12.1 SocketIO

SocketIO é uma implementação de um protocolo de comunicação em tempo real entre cliente e servidor baseada no protocolo webSocket. Por ser uma implementação própria de comunicação o servidor e o cliente necessita desta mesma implementação.

A implementação para o cliente é feita em linguagem JavaScript. Já para o servidor a implementação foi feita sobre o servidor web Flask e se chama Flask SocketIO ([SOCKET.IO. Welcome to Flask-SocketIO's documentation](#)).

### 3 DESENVOLVIMENTO

O desenvolvimento do projeto consiste na construção de uma plataforma para testes de controle automáticos, interface com o usuário e também com testes de controle proporcional.

O sistema de software consiste em três partes, ao *firmware* da plataforma que irá realizar a leitura dos sensores, receber os sinais de controle e atuar sobre o sistema eletromecânico, o servidor que irá implementar as funções de controle e fazer o intermédio de comunicação com a interface do usuário; e também a própria interface com o usuário que foi desenvolvida com a tecnologia *webview* para facilitar o desenvolvimento de aplicativos multiplataformas. Daí nasce a necessidade de se utilizar mais de uma tecnologia para a comunicação entre as partes do software.

Um dos principais desafios foi o desenvolvimento da comunicação entre as partes do sistema, interface, servidor e a plataforma. Como requisito a comunicação deve ser em tempo real para possibilidade de se aplicar as técnicas de controle.

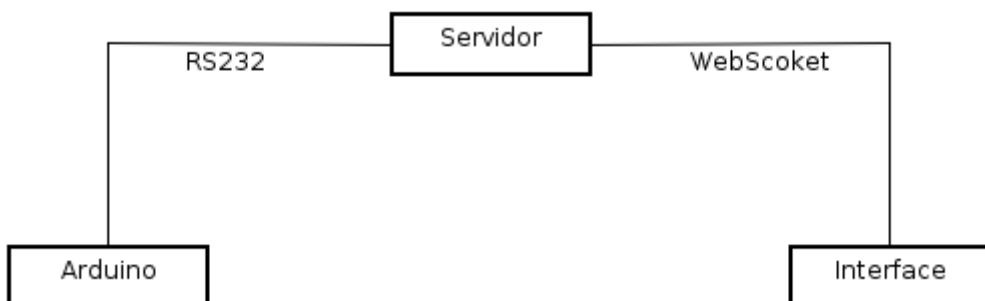


Figura 20: Diagrama de comunicação entre o servidor, interface e Arduino.

O servidor se comunica com a interface através da tecnologia de *websocket*, esta tecnologia foi escolhida por ser uma comunicação *full-duplex*, onde o cliente e servidor podem se comunicar em alta performance, além da conexão permanecer ativa durante todo o processo.

A comunicação feita entre o servidor e a plataforma, implementada com o Arduino, é realizada com a comunicação serial padrão a RS232 através da porta USB (*Universal Serial Bus*).

#### 3.1 Esquema elétrico

O sistema elétrico é composto por uma fonte de 12 volts externa para a alimentação dos motores e a alimentação do microcontrolador pela porta USB padrão do computador.

Para a alimentação e controle de velocidade dos motores foi utilizado um ESC que faz o intermédio entre o microcontrolador e a fonte de alimentação externa.

Os sensores são ligados diretamente ao Arduino. A imagem ilustra as ligações elétricas feitas para a alimentação e sensoriamento do sistema.

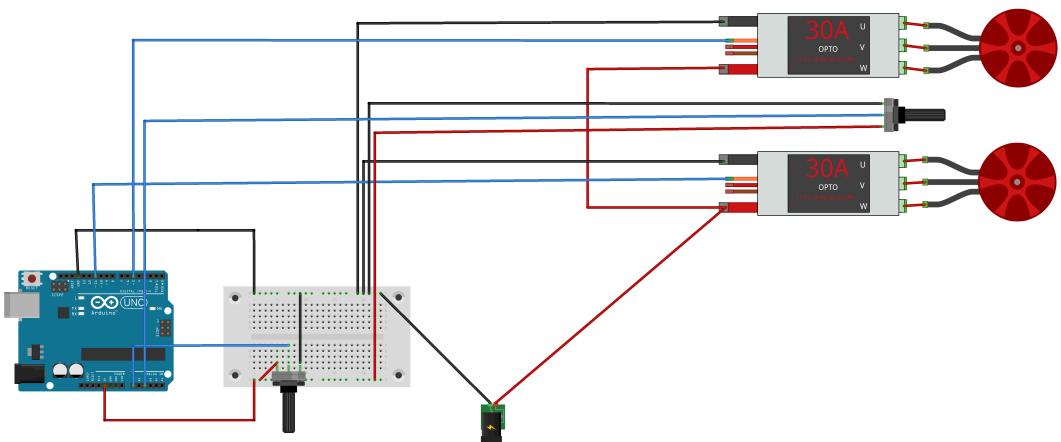


Figura 21: Esquema elétrico.

Como pode ser visto na imagem anterior o esquema elétrico é extremamente simples, suprindo apenas a necessidade de energia para o sistema de atuação, os motores e sensoriamento dos potenciômetros.

### 3.2 Construção mecânica

A plataforma usada para a simulação de atitude de giro de um quadricóptero foi construída com uma base de madeira medindo 30x30cm, uma haste central presa na base, que serve de apoio para uma outra haste que irá executar o movimento de giro.

A segunda haste foi presa à primeira haste utilizando um parafuso com folga para que permita a movimentação de forma livre.

Um potenciômetro foi preso de forma fixa no parafuso e ligado a segunda haste, servindo como sensor de giro para a plataforma.

A plataforma final está ilustrada na figura 22.



*Figura 22: Plataforma para simulação de atitude de giro de um quadricóptero.*

### 3.3 Identificação do sistema

Como mostrado na equação 14, a função de transferência está em termos de constantes advindas das propriedades físicas inerentes ao sistema como a constante de proporcionalidade entre velocidade angular das hélices e a diferença de pressão gerada  $k_p$ . Todas estas constantes são difíceis de determinar seu valor com exatidão assim foi proposto utilizar-se de técnicas de identificação de sistema para estimar a função de transferência do sistema.

#### 3.3.1 Modelo estimados

De acordo com a função de transferência descrita na equação 14, também utilizando a equação 13 para sistemas em malha fechada com um ganho unitário e obtemos a forma geral da equação de transferência de segunda ordem.

Foi Utilizado um sinal aleatório binário que varia entre dois valores com média aproximadamente zero para evitar que a plataforma saísse do seu estado de equilíbrio. E também medido o sinal de saída do sistema para poder estimar a função de transferência do sistema.

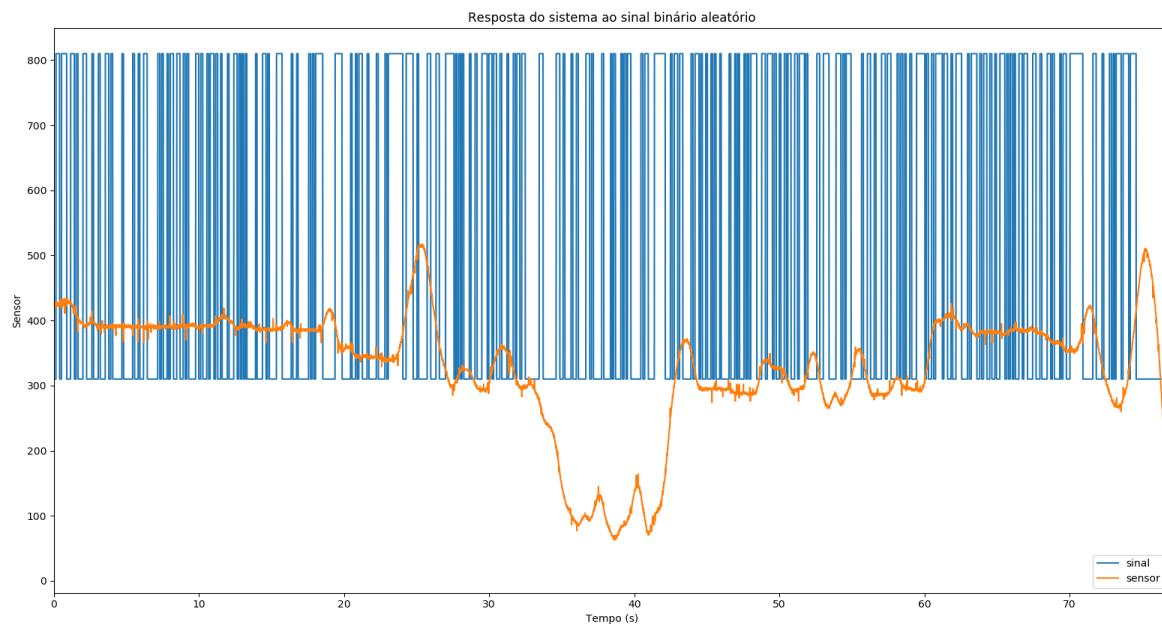


Figura 23: Resposta do sistema a um sinal binário aleatório.

### 3.3.2 Usando a ferramenta ident

Foram carregados os dados medidos no domínio do tempo e suas respectivas entradas para a ferramenta *ident* do *matlab* assim dando o processo de identificação.

Como mostrado na equação 17 o sistema possui ao menos dois polos por este motivo foi testado modelos com nenhum, um e dois zeros em sua equação característica, como mostra na imagem 24.

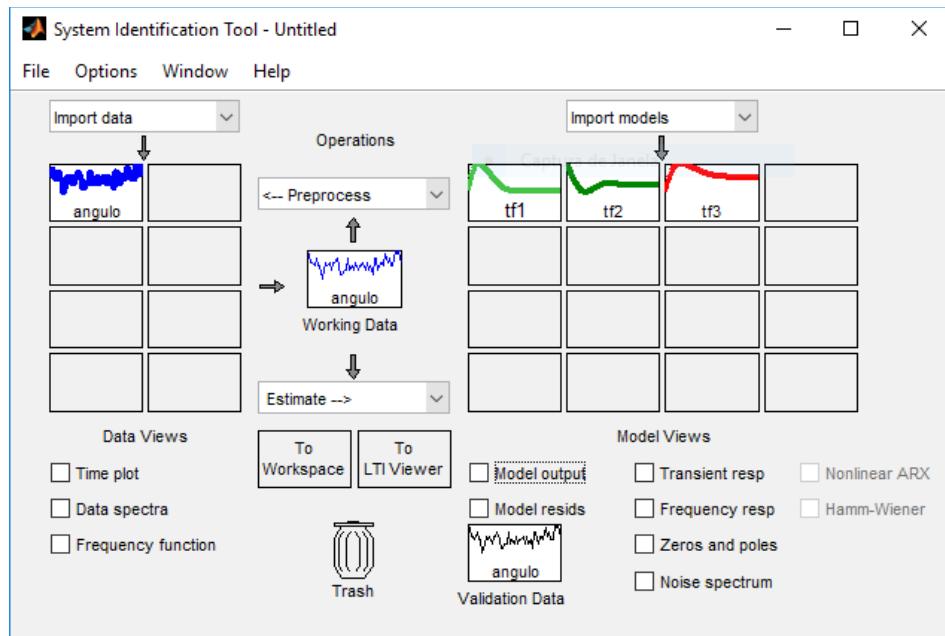


Figura 24: Ferramenta ident com os dados do sistema.

As funções de transferência tf1, tf2 e tf3 possuem nenhum zero, um zero e dois zeros respectivamente, e todas as funções possuem dois polos. Assim foi feita a identificação do sistema.

O método utilizado para a determinação dos parâmetros da função de transferência do sistema foi o de mínimos quadrados.

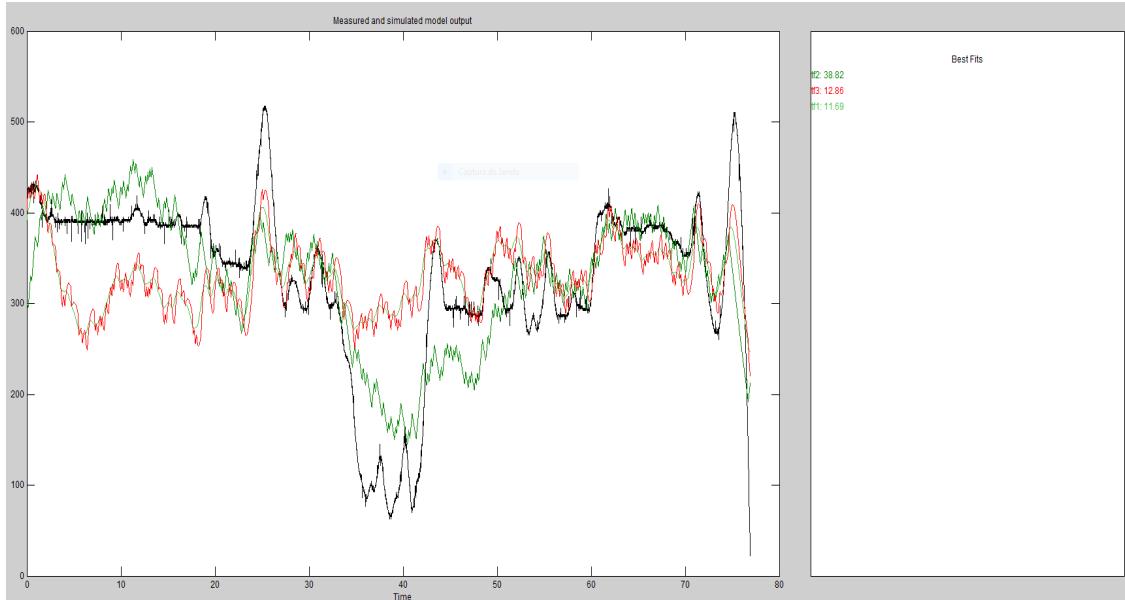


Figura 25: Teste de assertividade das funções de transferência.

Como pode ser visto na imagem 25 todas as funções de transferência tiveram um baixo índice de assertividade, mesmo assim foi escolhida a função de transferência tf2 para a estimativa de ganho para o melhor desempenho do sistema, pois a função de transferência tf2 obteve a melhor assertividade dentre as funções testadas.

A função de transferência tf2 possui dois polos e apenas um zero, provavelmente este zero advém de uma linearização das constantes de proporcionalidade da igualdade 12.

A função de transferência tf2 é mostrado na equação abaixo.

$$tf2(s) = \frac{0,3509s + 0,007287}{s^2 + 0,09338s + 0,0136} \quad (18)$$

### 3.3.3 Lugar de raízes da função de transferência estimada

Analizando o lugar de raízes do sistema é possível observar que este é estável e possui uma resposta com sobre-impulso, pois o sistema possuí todos os polos na região de estabilidade e estas primeiras observações se comprovam ao analisar a resposta ao degrau do sistema.

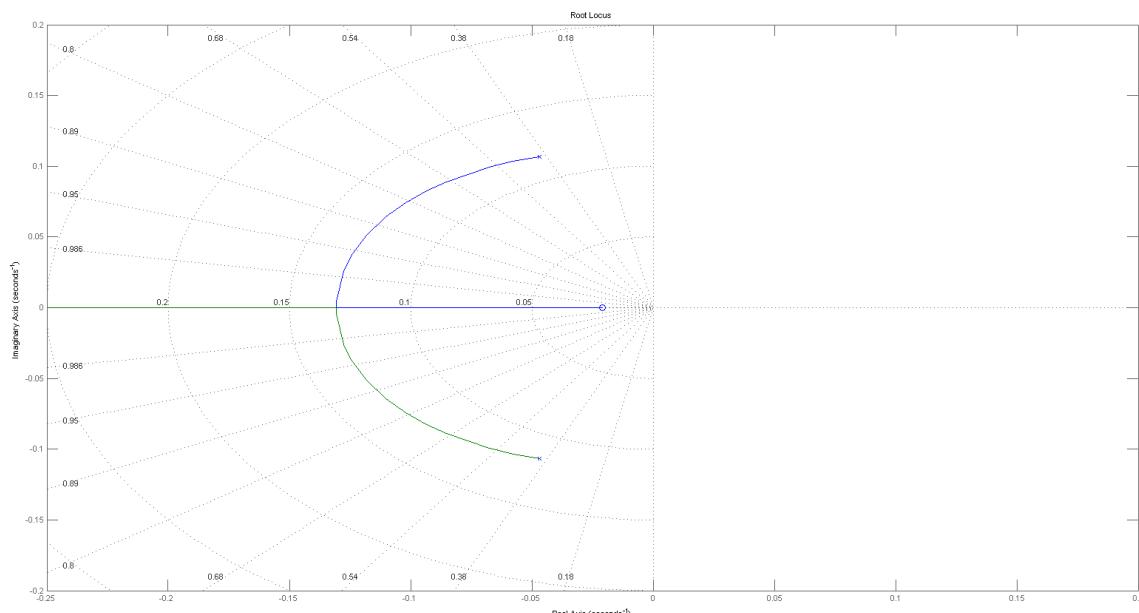


Figura 26: Lugar de raízes da função de transferência estimada.

A resposta ao degrau do sistema releva que o erro em estado permanente do sistema é de aproximadamente 50% e o tempo de estabilização do sistema esta entre 80 e 90 segundos.

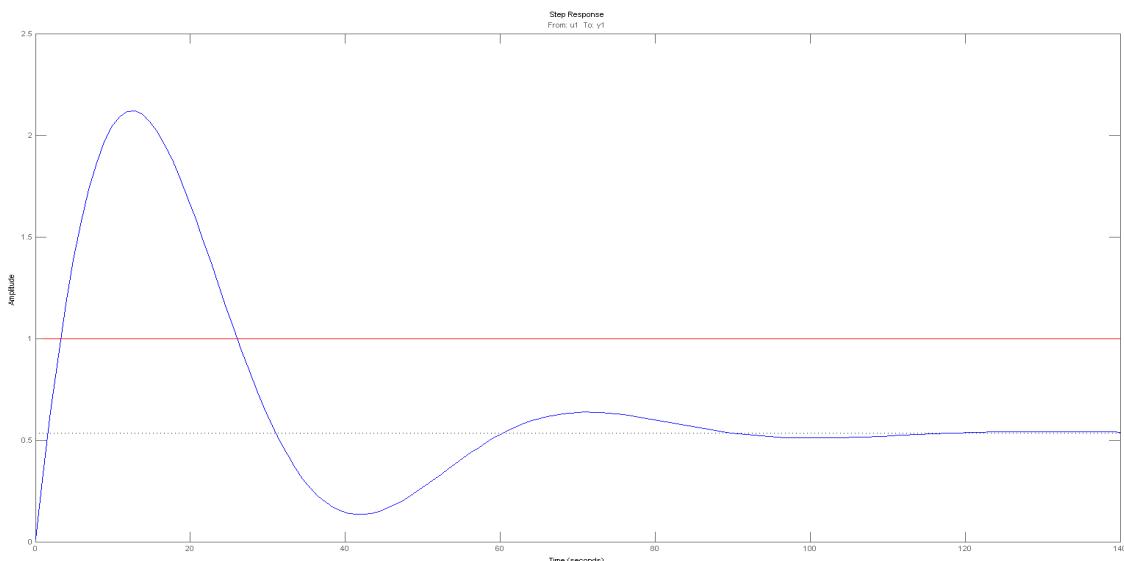


Figura 27: Resposta ao degrau do sistema.

### 3.4 Interface com o usuário

Interface homem-máquina ou interface de usuário é um equipamento ou ferramenta de interação e *feedback* entre o sistema e o ser humano onde sua função desempenhada é o melhor controle e facilidade de uso.

A comunicação entre máquina e usuário utilizasse de software como botões, caixas de diálogo, mensagens, alertas e etc. E hardware como LED's, monitores e e tec.

O sistema de interface de usuário foi elaborado utilizando o framework electron, que auxilia o desenvolvimento de aplicativos híbridos, assim aumentando a gama de recursos disponíveis e também a facilidade de portabilidade para outras plataformas.

#### 3.4.1 Desenvolvimento de aplicativos híbridos

Aplicativos híbridos são uma união de tecnologias nativas de cada plataforma com tecnologias web. A tecnologia nativa é como um envoltório que encapsula o sistema web.

Algumas das vantagens do desenvolvimento de aplicativos híbridos são: a facilidade de desenvolvimento, maior variedade de recursos a serem utilizado,

portabilidade entre diversas plataformas, mas há também alguns pontos fracos tais como baixa performance e etc.

### 3.4.2 Electron

Electron é um framework de desenvolvimento de aplicativos híbridos para desktop desenvolvido e mantido pelo github.

Em seu desenvolvimento foi utilizados tecnologias provenientes da web, seu backend é desenvolvido em NodeJs e o frontend utiliza o webview, renderização de páginas web, chromium, assim podendo HTML, CSS, javaScript e etc para se construir uma interface em um aplicativo nativo da plataforma.

### 3.4.3 Frameworks utilizados para construção da tela

Um conjunto de frameworks foram utilizados para a construção da tela, cada um com sua função específica, fornecer um desenvolvimento em camadas, responsividade, interatividade e etc.

Em seguida será citado os frameworks mais importantes utilizados no desenvolvimento.

#### 3.4.3.1 AngularJs

AngularJs é um framework de desenvolvimento de páginas web dinâmicas desenvolvido e mantido pela Google.

De acordo com os desenvolvedores do angular ele surgiu com a necessidade de se tornar o HTML das páginas web mais dinâmico e também evitar o uso excessivo da árvore DOM. Ele fornece um desenvolvimento em camadas MVVM (*model-view-view-model*).

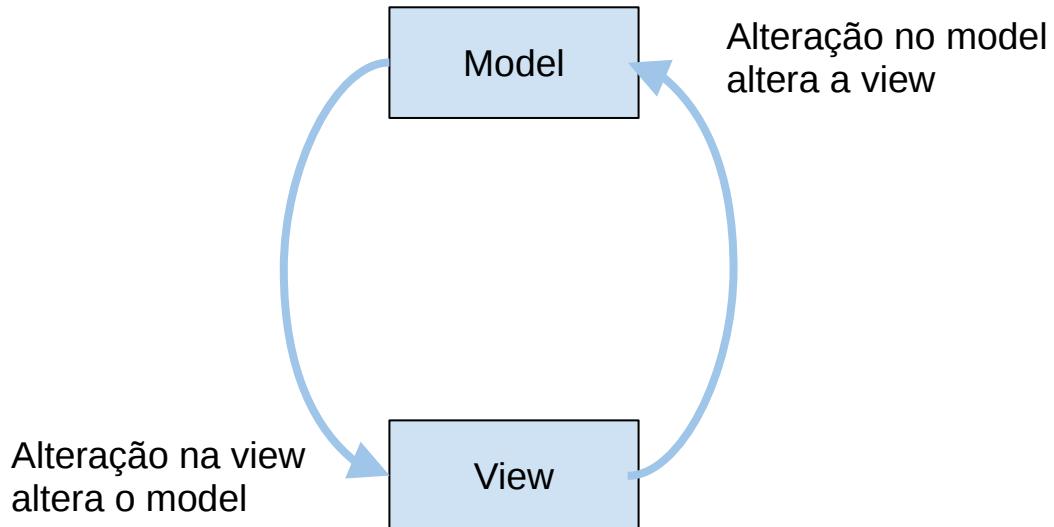


Figura 28: Sistema de atualização de dados do AngularJS.

MVVM fornece bem uma visão de como é seu funcionamento interno. Qualquer modificação na visualização dos dados gera automaticamente uma correspondência no modelo de dados e vice-versa, este comportamento é chamado *two way data binding*.

Angular é ideal para desenvolvimento de aplicações *single page*, ou seja toda a sua aplicação será em apenas uma página utilizando rotas virtuais para trazer apenas o conteúdo a ser alterado na tela.

No projeto o Angular tem a função de criar os controles da tela, as funcionalidades e também as rotas, em resumo dar o dinamismo ao sistema.

#### 3.4.3.2 Bootstrap

Bootstrap é outro framework utilizado no projeto, assim como o Angular ele fornece um padrão de desenvolvimento, mas agora aplicado a estrutura da página HTML e CSS.

Bootstrap foi desenvolvido pelo *Twitter* para facilitar o desenvolvimento de sua plataforma, o principal objetivo em usar o bootstrap é a responsividade, capacidade de se adequar a vários tipos e tamanhos de tela, na interface do sistema.

Seu uso no projeto se dá pela facilidade em organização do conteúdo a ser apresentado ao usuário e principalmente pela sua responsividade.

#### 3.4.3.3 CanvasJs

CanvasJs é um framework para geração de gráficos baseado em javascript, este foi utilizado para desenvolver a visualização em tempo real das leituras realizadas pelo firmware.

#### 3.4.4 Caso de uso

O sistema atualmente possui quatro casos de uso que são eles: *configurar função de controle*, *iniciar fluxo de controle*, *visualização dos gráficos dos sensores* e por fim o caso de uso *finalizar processo de controle*.

O diagrama de caso de uso da figura 29 evidencia uma visão geral do sistema, assim apresentando os autores responsáveis pela operação do sistema, descrevendo as principais funcionalidades do sistema e a interação dessas funcionalidades com o ator.

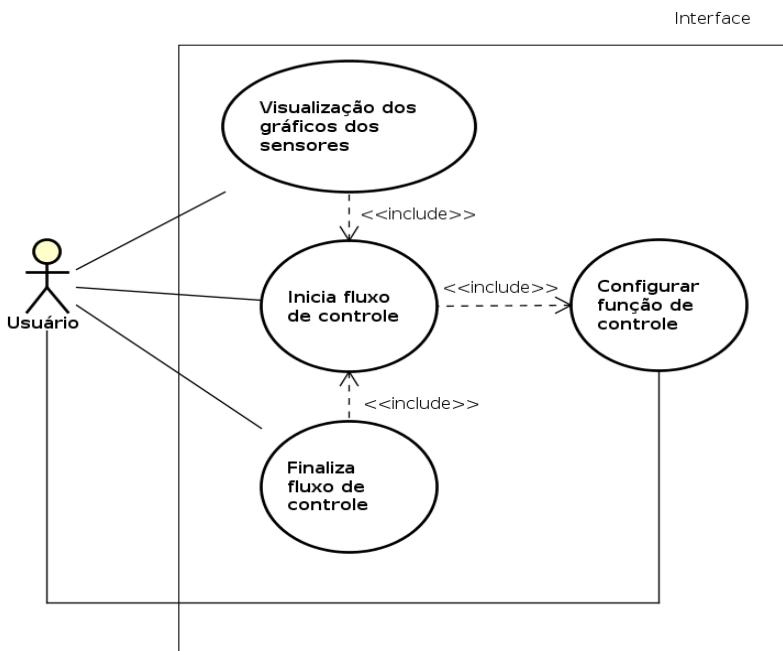


Figura 29: Caso de uso do sistema.

O autor do sistema: qualquer pessoa que possua conhecimento para operá-lo. O mesmo tem acesso a todas as funcionalidades desenvolvidas.

Ao acessar o sistema o ator deverá inicialmente fazer uma configuração de função de controle para assim o sistema iniciar o fluxo de controle. Dado o fluxo de controle iniciado o ator poderá visualizar os gráficos do sensores e finalizar o fluxo de controle quando desejado.

Para a execução do caso de uso *iniciar fluxo de controle* é necessário uma função de *feedback* para atuação na planta. A função de *feedback* é configurada no caso de uso

configuração de função de controle e por este motivo é necessário a sua execução antes da inicialização fluxo de controle.

A comunicação existente entre os demais casos de uso é necessária para viabilizar a ação do autor entre as funcionalidades.

### 3.4.5 Diagrama de sequência das funcionalidades

O diagrama de sequência ilustrado na figura 30 representa a sequência do processo de configuração da função de controle que irá atuar na planta. Este processo é iniciado capturando qual função de controle será utilizada e seus respectivos parâmetros de configuração, será enviado um comando de configuração da função de controle após a interação entre o usuário e o sistema de interface. Estes dados serão encaminhados para o servidor, para a implementação e validação, e após o processamento será dado uma resposta para a interface onde o usuário conseguirá ter um *feedback* da configuração.

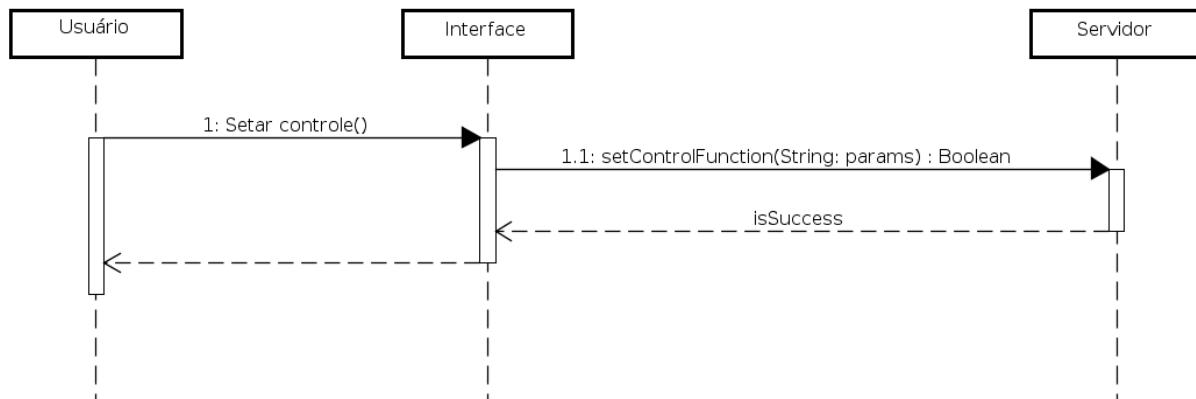


Figura 30: Diagrama de sequência do fluxo de configuração do controle.

A sequência de inicialização do processo de controle automático é representada na figura 31. O sistema irá bloquear o usuário da função de inicialização do controle automático até que a plataforma esteja conectada e uma função de controle configurada, este bloqueio se dá através da desativação da função que inicia este fluxo; Ao iniciar o processo será encaminhado uma mensagem ao servidor que irá abrir a comunicação com o firmware, permitindo o controle da planta, e em seguida será enviado uma mensagem para a interface validando a inicialização do processo e uma mensagem será mostrada na interface ao usuário.

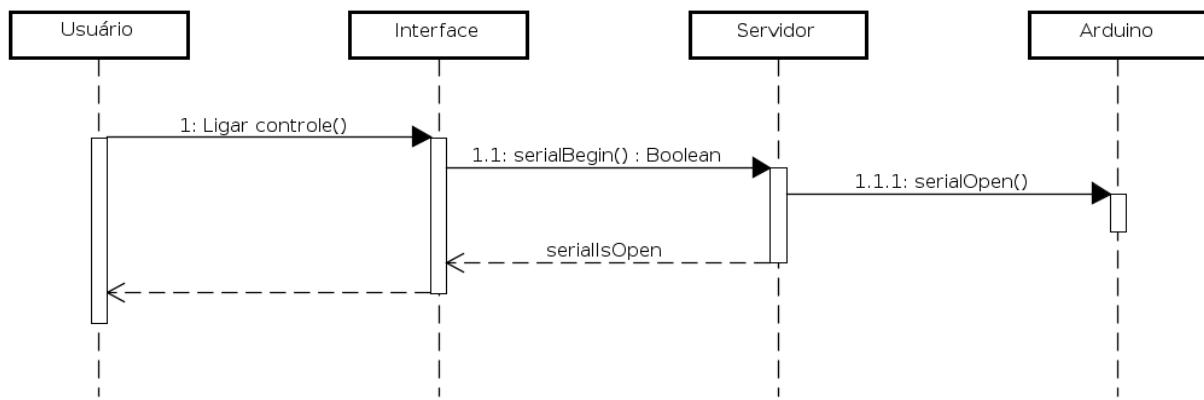


Figura 31: Diagrama de sequência do fluxo de inicialização do controle.

O processo de controle automático exposto na figura 32 é responsável pelo controle automático da planta. O firmware irá realizar as leituras dos sensores e transmitir para o servidor que processará os dados recebidos, estes dados serão retransmitidos para a interface onde o usuário do sistema poderá acompanhar sua evolução.

Em sequência o servidor irá processar os dados gerando os sinais de controle, baseados na função de controle configurada anteriormente, e os transmitir para o firmware atuar na plataforma. Este fluxo é iterativo sendo repetido a cada 20 milissegundos até a solicitação de finalização por parte do usuário.

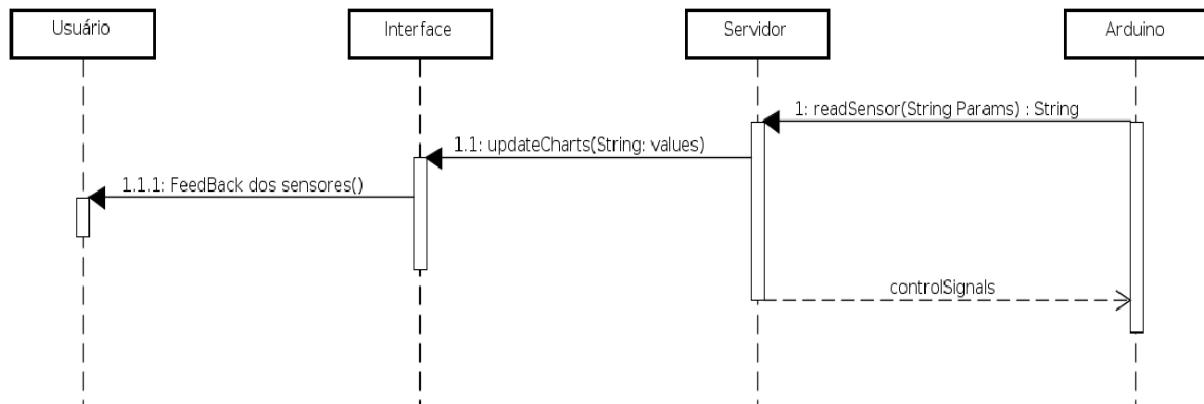


Figura 32: Diagrama de sequência do fluxo de controle automático.

A sequência de finalização do processo de controle automático é retratada na figura 33. O sistema irá bloquear o usuário da função de finalização até que o sistema esteja executando o processo de controle automático; A interface irá enviar um comando para servidor informando-o da finalização, o servidor por sua vez irá se comunicar com a plataforma para que está coloque o sistema no estado inicial e assim desligando os motores, após a comunicação do servidor com a plataforma, ele irá enviar uma mensagem para a interface dando o feedback sobre o processo.

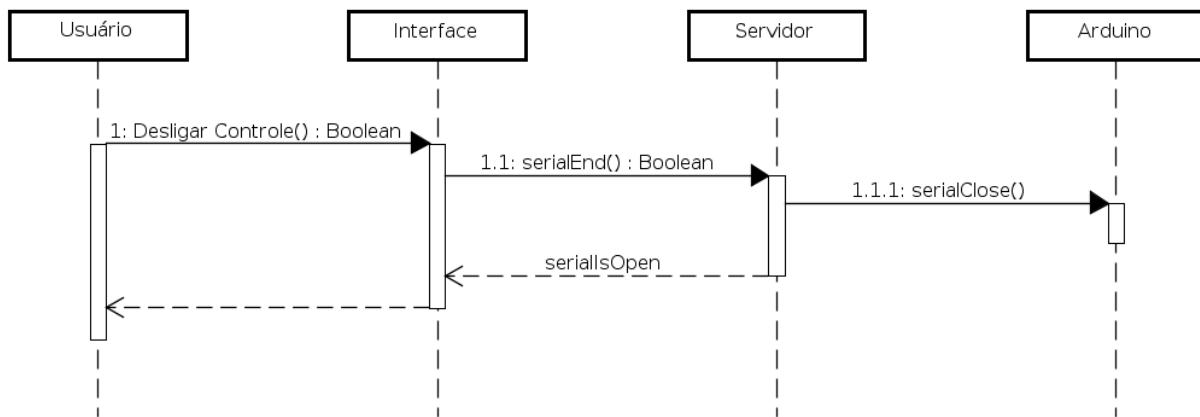
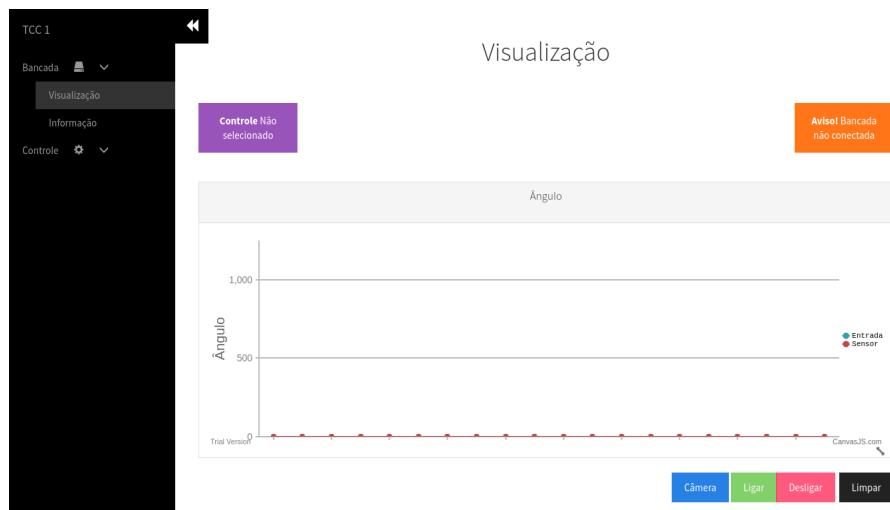


Figura 33: Diagrama de sequência do fluxo de finalização do controle.

### 3.4.6 Telas do sistema

O sistema consiste em dois principais tipos de tela, que são elas: tela de visualização dos sensores ilustrada nas figura 34 e 35, e as telas de configuração de controle mostradas nas figuras 36 e 37.

Na lateral esquerda de todas as telas será mostrado o menu de navegação retrátil que dá acesso a todas as telas do sistema e a na lateral direita é mostrado o conteúdo de cada tela.



*Figura 34: Tela de visualização de sensores.*

A tela de visualização dos sensores tem as funcionalidades de iniciar e finalizar o fluxo de controle e obviamente monitorar a leitura dos sensores.

Os cantos superiores possuem notificações sobre o sistema; No canto superior esquerdo se encontra a notificação sobre a função controle automática selecionado e ao

canto direito a notificação de conexão da plataforma; Logo abaixo das notificações estão os gráficos dos sensores de arfagem e guinada, as respectivas imagens ilustradas a lateral de cada gráfico e por fim os botões que iniciam e finalizam o fluxo de controle.

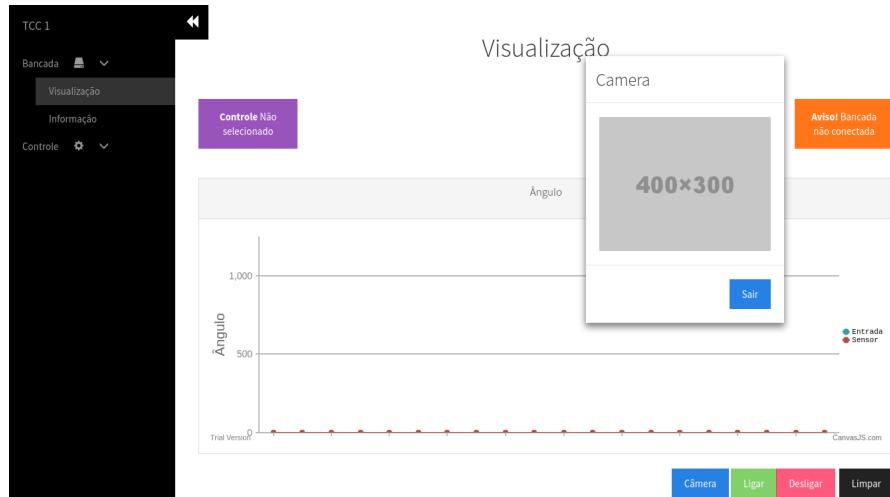


Figura 35: Tela de visualização dos sensores com a visualização da câmera ativa.

Foi adicionado um recurso de visualização remota por uma *webcam* para futura implementação de controle automático remoto.

Na figura 36 é mostrado a tela de configuração do controle proporcional onde se escolhe apenas o ganho para cada grau de liberdade do sistema, ou seja arfagem e guinada. Ao final da tela está presente um botão que enviará as configurações do controle para o servidor.

A tela de controle proporcional consiste em um campo de texto onde pode-se escrever uma função na linguagem de programação python que será interpretada e executada no fluxo de controle automático. Esta tela está representada na figura 37.

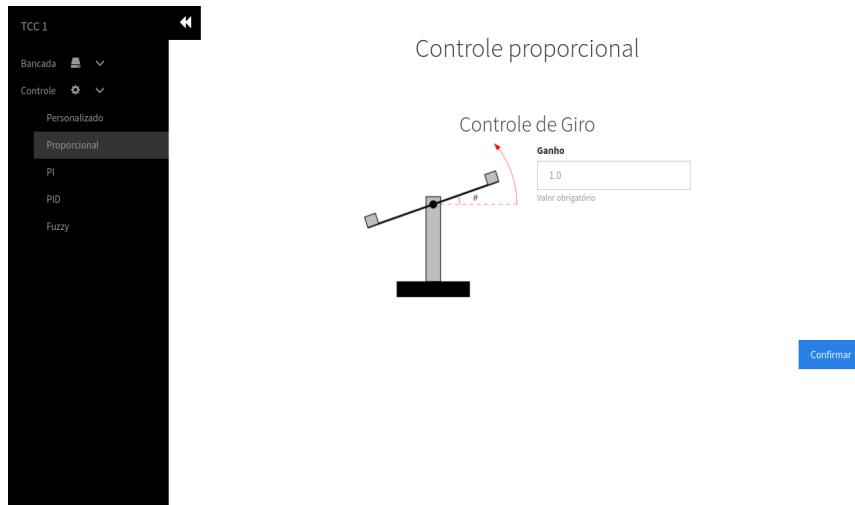


Figura 36: Tela de configuração de controle proporcional.

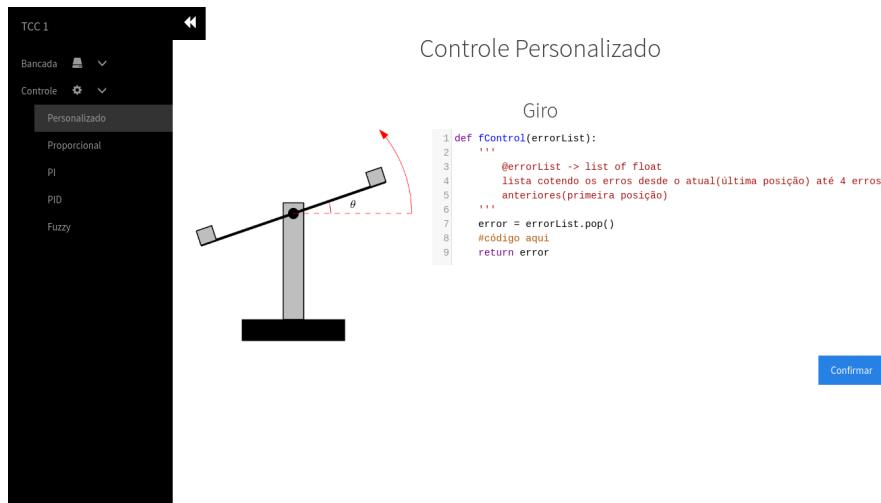


Figura 37: Tela de configuração de controle personalizado.

### 3.5 Firmware

O firmware é um software de controle e monitoramento, mais precisamente utilizado em sistemas embarcados. A implementação do firmware na plataforma Arduino foi feita em sua IDE (*Integrated Development Environment*) de desenvolvimento padrão.

A função do firmware no projeto é realizar a leitura dos sensores instalados na plataforma e os enviar para o servidor processar, também é de sua função receber os comandos do servidor e executá-los.

#### 3.5.1 Comunicação com o servidor

O firmware utiliza o protocolo RS232 para efetuar a comunicação com o servidor, assim sendo, a máquina onde é executado o servidor deve estar diretamente ligada ao Arduino com um cabo USB padrão.

A troca de informação se dá através do envio e recebimento de mensagens padrão contendo comandos a serem executados e seus respectivos parâmetros.

Os comandos possíveis de recebimento são dois: o de atuação sobre a planta do sistema, onde são enviados os sinais de controle, e o sinal de parada, que deverá interromper a leitura dos sensores e colocar o sistema no estado inicial, assim evitando manter a plataforma em uma posição fora de equilíbrio.

O comando de parada é representado pela letra 'e', o sistema ao receber este comando irá parar os dois motores, deixando assim a plataforma no estado inicial, pronto para receber outro teste de controle.

O comando para atuar sobre a planta consiste no sinal de controle podendo ser positivo ou negativo, assim atuando sobre o rotor da esquerda ou da direita. O programa principal é responsável por identificar e executar cada comando.

### 3.5.2 Programa principal

A figura 38 ilustra o fluxograma do programa principal do Arduino. Primeiramente o firmware irá configurar o ambiente, ou seja, os pinos de entrada e saída para a leitura e controle do sistema eletromecânico e também a interrupção de tempo para execução periódica da leitura dos sensores. Após a fase inicial de configuração o sistema entrará em laço esperando receber informações do servidor. A leitura de serial é feita e um *parsing*, interpretação e validação do que foi lido, é executado para obter o comando e seus parâmetros para posteriormente os efetuar.

Os comandos utilizados para a comunicação são dois: atuar sobre a planta e finalizar o processo, colocando a planta no seu estado inicial. O sistema receberá uma mensagem de texto do servidor com o valor a ser aplicado aos rotores. A magnitude do número indica a magnitude do sinal a ser aplicado e o sinal do valor lido representa o direcionamento do sinal. Sinal positivo irá atuar no rotor esquerdo e o negativo no rotor direito.

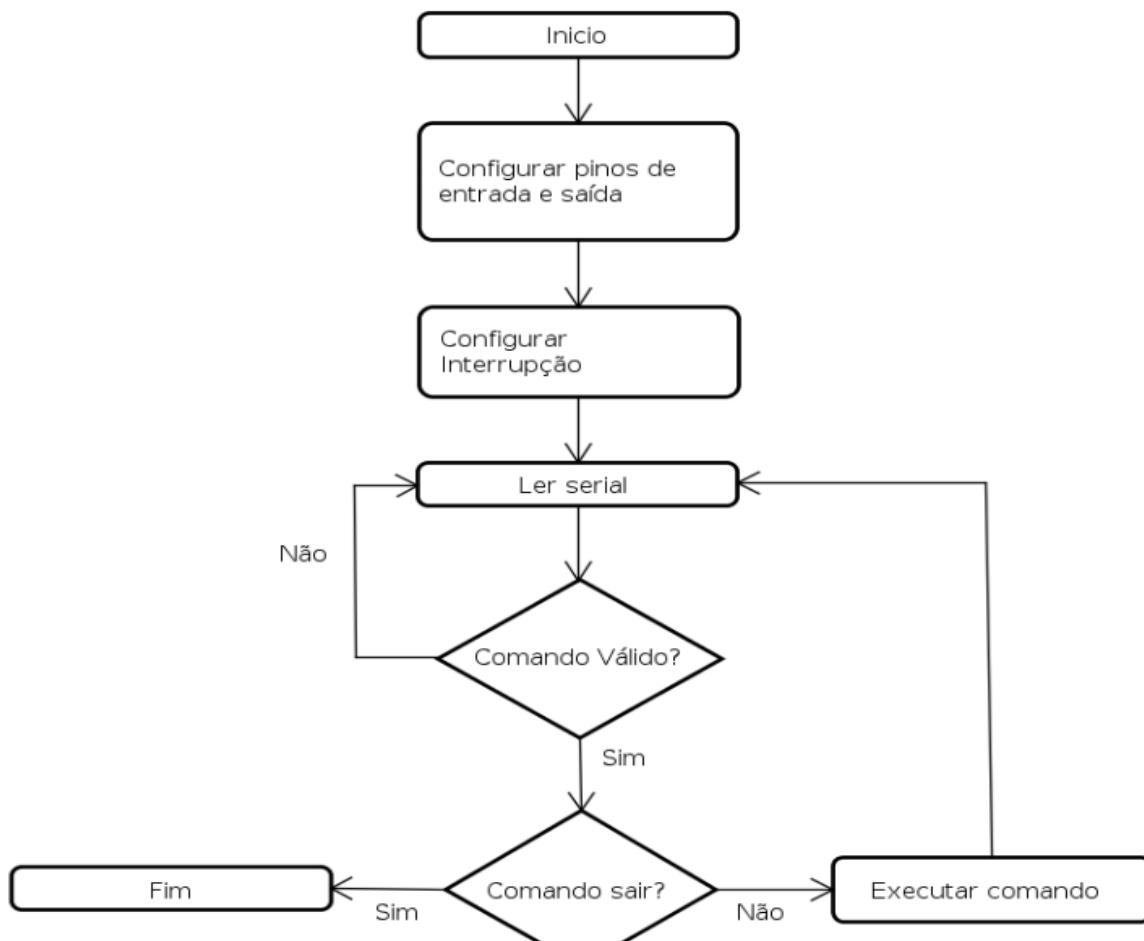


Figura 38: Fluxo principal do programa.

### 3.5.2.1 Geração de sinal PPM para controle dos motores

Como visto no esquemático do sistema elétrico a controladora não pode ser ligada diretamente nos motores, pois a controladora não consegue fornecer corrente a suficiente, além de trabalharem em faixas de tensão diferentes (olhar seção 2.4).

Para efetuar o controle é utilizado um componente chamado ESC, este componente recebe um sinal de controle do tipo PPM e atua sobre o motor.

O Arduino possui um gerador de sinal PWM nativo, mas não um de PPM assim foi necessário utilizar a biblioteca servo para implementar este tipo de modulação.

Cada ESC é ligado à placa e configurado como sendo um servomotor. Como a biblioteca foi criada para controlar servomotores e não apenas gerar um sinal modulado em posição é necessário fazer um ajuste de parâmetro, pois o sinal de controle varia de 0 a 1023. Já a função de controle do servo recebe um parâmetro que varia de 0 a 180 que representa o ângulo que o servo fará.

Como citado anteriormente na fundamentação teórica o motor possui uma banda de tensão de funcionamento que varia de 7 a 14 volts, então o parâmetro de atuação nos motores foram mapeados para os valores de 40 a 60 graus na função de controle do servo para aumentar a velocidade de resposta e diminuir a potência total assim melhorando o controle do sistema.

Segue um exemplo de código utilizando a biblioteca servo para atuar sobre um ESC.

```
#include <Servo.h>

Servo ESC; //Cria um objeto do Servo
            //para o controle de um ESC

#define ESC_PIN 10 //Pino onde o ESC está conectado

void setup() {
    ESC.attach(ESC_PIN); //Configurando ESC para o pino definido
    ESC.write(0);        //Colocando zero na saída para que o
                        //ESC regule sua escala interna
    delay(1);           //Espera para o ESC se configurar
    ESC.write(90);      //Na escala de 40 a 140, 90 colocará
                        //o motor na velocidade média
}

void loop() {
```

Figura 39: Exemplo de código para controle de um ESC utilizando a biblioteca Servo.h.

### 3.5.3 Interrupção de tempo

Na execução da leitura dos sensores é utilizada uma interrupção de *timer* para garantir a periodicidade da amostragem. A amostragem é feita a uma frequência de

$f=50\text{Hz}$  assim sendo o período de amostragem em  $T=\frac{1}{f}=0,02$  segundos ou 20 milisegundos.

#### 3.5.3.1 Timers do Arduino Uno

O Arduino Uno utilizado no projeto possui três *timers* internos para controle de funções de temporização: o *delay*, *mili* e *micro*, e também para a geração de sinais: PWM (*Pulse Width Modulation*) e PPM (*Pulse Position Modulation*). Alguns destes *timers* estão disponíveis para que possam ser utilizados para algumas ocasião.

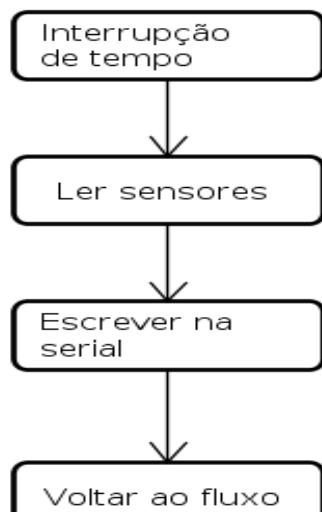
A interrupção para aferir os sensores se dá utilizando o *timer2* do Arduino uno que possui uma precisão de oito *bits*. O *timer2* foi escolhido, pois é utilizado somente na função *tone*. Como esta função não é utilizada no projeto então o *timer2* está livre para uso na aplicação.

### 3.5.3.2 Fluxo de execução da interrupção de tempo

A função de interrupção possui um fluxo bem simples; amostrar o sinal do sensor da planta: o sensor de *setpoint* e o o sensor de leitura de giro. Após a obtenção dos valores, será montada uma *string* contendo as leituras separada por uma vírgula. O formato final a ser enviado para o servidor está mostrado na seguinte imagem

“*setpoint,valor\_do\_sensor*”.

*Figura 40: Formato da string enviada para o servidor.*



*Figura 41: Fluxograma da interrupção.*

## 3.6 Servidor

Um servidor web é um software centralizado ou distribuído que fornece alguns tipos de serviços em rede a pedido de clientes.

O servidor para o projeto é necessário para a comunicação entre o sistema de interface e o hardware já que a interface é um sistema desenvolvido com tecnologia híbrida, ou seja uma página web renderizada em um *webview*.

Com a adição do servidor adiciona-se complexidade no desenvolvimento do projeto, mas abre espaços para outras funcionalidades como o controle automático remoto, onde o usuário do software pode operar a plataforma pelo sistema de internet ou intranet.

### 3.6.1 Python

Python é uma linguagem de programação de alto nível, de fácil aprendizado e muito poderosa por ter uma comunidade grande e ativa, além de python ser uma linguagem livre suportada nativamente pela maioria dos sistemas POSIX, também sendo uma das melhores ferramentas para a manipulação de números, assim se tornando uma ótima escolha para o desenvolvimento do projeto.

#### 3.6.1.1 *Flask SocketIO*

Flask é um micro-framework de desenvolvimento de servidores web desenvolvido na linguagem de programação python. A utilização do micro-framework flask se dá na implementação da comunicação entre o servidor e a interface gráfica através da tecnologia de *websockets*.

Flask-SocketIO é um módulo específico que adiciona a implementação de *websockets* em softwares desenvolvidos com o Flask de fácil utilização. Fornecendo um objeto para a implementação de chamadas a eventos do *websocket*. As principais funções são: função *on* que recebe dois parâmetros o nome do evento e a função a ser executada ao receber eventos nomeados e não nomeados.

As funções de ativação de eventos são duas a *emit* e o *send*, onde a primeira faz o “lançamento” de um evento com nome e algum parâmetro a ser transmitido e a função *send* faz o mesmo que a função anterior com o diferencial de não precisar de um nome de evento assim ativando todos os ouvintes (*listeners*) cadastrados.

A partir da demanda de eventos do *websocket* todo o software foi desenvolvido com a arquitetura orientada a eventos, onde há os ouvintes (*listeners*) que esperam o acontecimento de um evento, os observadores (*observers*) que vigiam algum evento,

como por exemplo um clique de mouse, quando este ocorre o observador avisa os ouvintes do acontecimento e assim os que estavam a espera do evento serão executado.

### 3.6.2 Padrões de projetos

Alguns padrões de projetos foram necessários ao desenvolvimento do servidor. A demanda por um sistema de evento necessitou de alguns padrões tais como o *observer* e *singleton*. Além desta demanda há outra de extrema importância, que trata os dados recebidos através da interface de usuário e os transformam em funções para serem executadas no controle automático.

#### 3.6.2.1 Singleton

O padrão de projeto *singleton* garante que em uma classe específica sempre será instanciado o mesmo objeto, assim garantindo o compartilhamento de configurações, funções e dados através do sistema.

O sistema de eventos necessita de um *singleton* que receberá as funções e parâmetros destas para a sua correta execução, assim necessitando do compartilhamento de uma mesma instância.

#### 3.6.2.2 Sistema baseado em eventos

Sistemas baseados em eventos são bastante práticos para se trabalhar com o paralelismo de informação sendo que cada evento pode disparar vários ouvintes (*listeners*) de uma só vez.

A demanda pela implementação do sistema de ventos veio a partir do funcionamento e modelo de comunicação através do *websocket*. A integração do servidor com a interface de usuário foi bastante simplificada por este tipo de arquitetura.

A figura 42 representa o fluxo do sistema de eventos. Primeiramente é adicionado a uma estrutura de dados o nome do evento e sua respectiva função, sendo que um evento pode conter mais de uma função. Após a configuração inicial o sistema ficará em uma repetição lendo todos os possíveis eventos; se algum destes for acionado o sistema irá executar todas as funções vinculadas; o sistema só sairá do laço para ser finalizado.

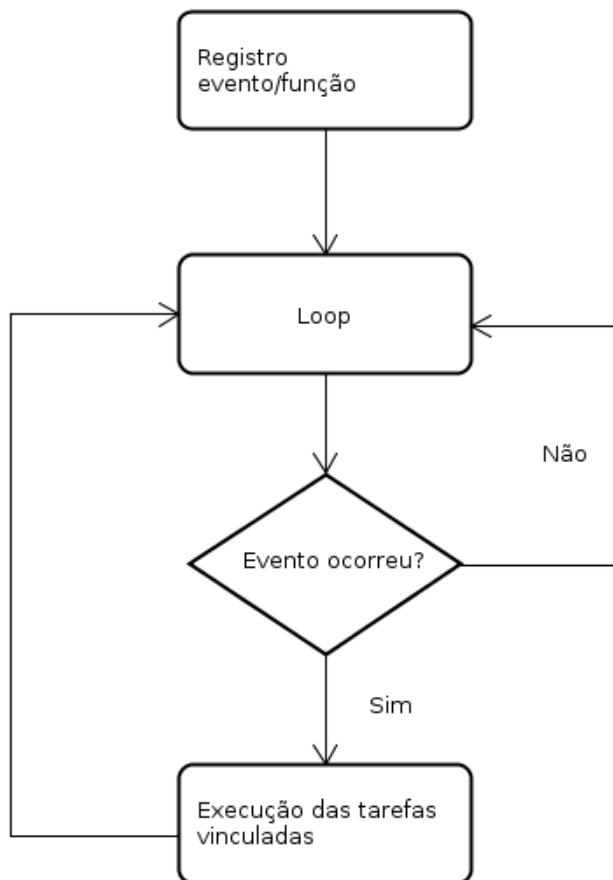


Figura 42: Diagrama de execução de evento.

### 3.6.2.3 Funções de controle

Como objetivo de testar algumas funções de controle o sistema deve ser implementado com baixo acoplamento entre as classes para facilitar a troca das funções de controle.

A figura 43 ilustra o diagrama de classes do sistema de funções de controle.

A classe *HandleController* é responsável por fazer o intermédio entre as funções do servidor e as classes de que implementam cada função de controle.

Cada função de controle implementa a interface *ControllerInterface* garantindo o polimorfismo das classes de controle. As classes de controle possuem a responsabilidade de calcular o sinal de controle dados as configurações anteriormente feitas, configuração esta que é recebida da interface de usuário e passada para a instância correta da função de controle por intermédio do *HandleController*.

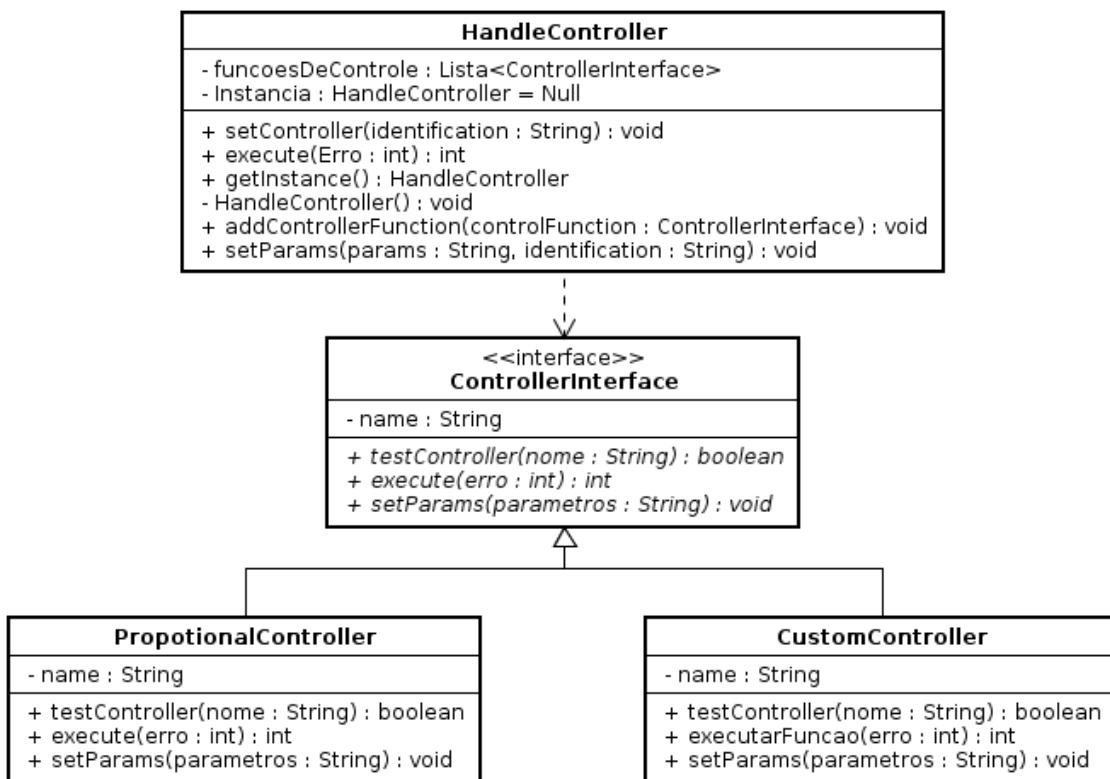


Figura 43: Diagrama de classes do sistema de funções de controle.

### 3.6.3 Compartilhamento da porta serial

Para a comunicação entre o servidor e o Arduino é utilizado o protocolo serial RS232. A biblioteca em linguagem python que implementa estas funcionalidade é a PySerial.

O sistema apresenta características de tempo real, ou seja, o sistema deverá processar as respostas em um tempo limite. Para a construção da função de controle é necessária a leitura contínua dos sensores e a escrita nos atuadores da planta elétrica.

A leitura do sistema é de prioridade crítica. Para a resolução deste requisito foi elaborada uma *thread* de execução contínua cuja a funcionalidade é efetuar a leitura da serial e armazenar o resultado em um *buffer*.

O processo de escrita na serial para atuação do sistema é concomitante ao de leitura e assim foi necessário um mecanismo para realizar o compartilhamento da porta serial pelo acesso às *thread* de leitura e escrita do sistema. Este mecanismo se chama exclusão mútua (*mutex*).

Quando a *thread* de leitura da serial entra na região crítica, ou seja quando for acessar um recurso compartilhado, o *mutex* é ativo e portanto evitando que qualquer

outra *thread* acesse este recurso. Ao terminar a região de criticidade o *mutex* é liberado e assim outras *threads* que estejam aguardando pelo recurso serão liberadas e executadas.

## 4 RESULTADOS

Utilizando o software *matlab* e a função de transferência estimada do sistema foi possível ajustar o ganho para que o sistema tive o melhor desempenho possível com o controlador proporcional. O ganho estimado foi de para melhor desempenho foi 1,01.

Os resultados obtidos utilizando o ganho acima descrito não foram satisfatórios, pois o sistema e a função de transferência estimada são bastante diferentes, como mostrado no seção 3.3.

Ao se variar rápido demais o valor da entrada do sistema a plataforma tende para o outro lado e assim aumentando o erro de regime permanente, ao se variar a entrada de maneira suave o sistema consegue acompanhar, mas mesmo assim com um grande erro.

Para obter um melhor resultado sobre o sistema é necessário acrescentar um componente integrador no controle, assim reduzindo o erro estacionário.

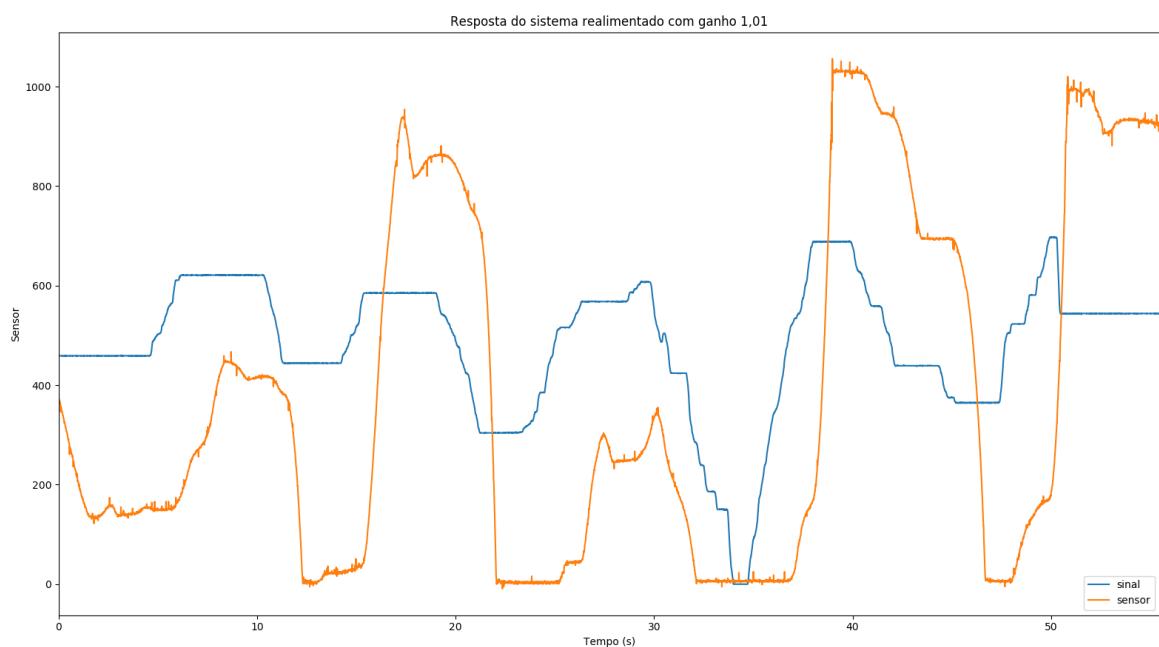


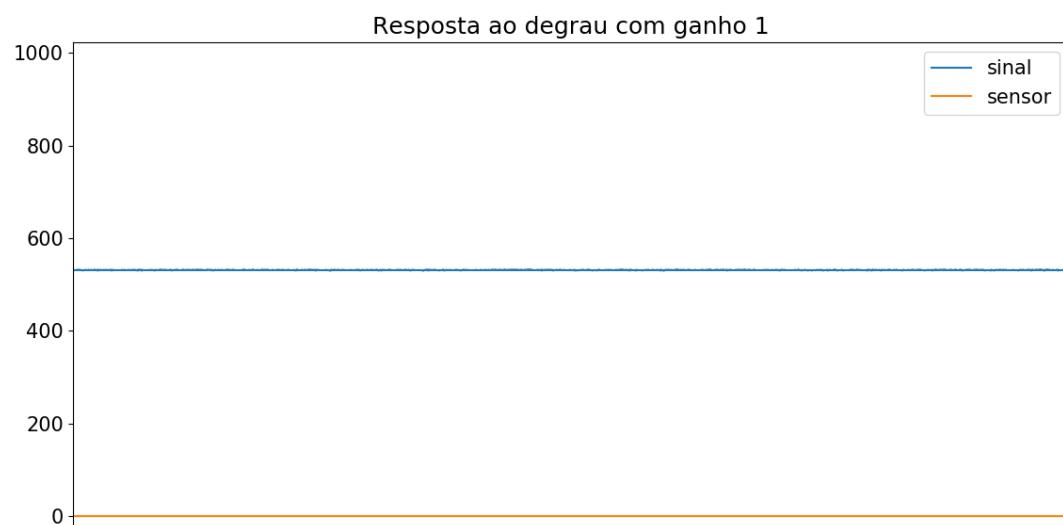
Figura 44: Resposta do sistema com ganho de 1,01.

Após a experimentação com o ganho de 1,01 do controlador proporcional foi efetuado outros testes com a variação do ganho de 1 até 1,8 com o incremento de 0,2.

Os sinais de entrada utilizados nos testes foram: o sinal degrau e o impulso, sendo que a resposta ao impulso não apresentou resultados satisfatórios, pois a aplicação de

um sinal de variação tão rapidamente não à ativação dos rotores e o sistema permanece estacionário.

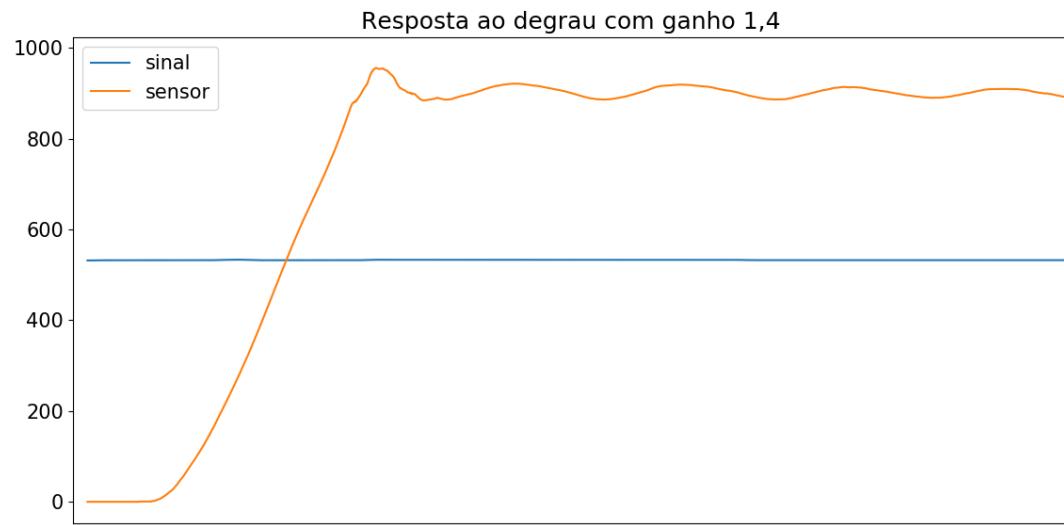
Com a análise dos resultados mostrados abaixo o sistema apresenta comportamento completamente oscilatório com o ganho maior que 1,8.



*Figura 45: Resposta ao degrau com ganho 1.*



*Figura 46: Resposta ao degrau com ganho 1,2.*



*Figura 47: Resposta ao degrau com ganho de 1,4.*



*Figura 48: Resposta ao degrau com ganho 1,6.*



*Figura 49: Resposta ao degrau com ganho de 1,8.*

Como é possível observar com os testes de ganho com o sinal de entrada degrau, obteve-se o melhor resultado com o ganho de 1,4.

## 5 CONCLUSÕES

O sistema de comunicação por mensagem desenvolvido para a integração do firmware com a interface e o servidor teve um bom desempenho.

A tecnologia de websockets utilizada para a visualização em tempo real do sistema na interface não responde bem em frequências acima de 35 Hz, assim a visualização dos dados em tempo real é comprometida.

A arquitetura escolhida de um servidor intermediando a comunicação da interface com o firmware foi de grande satisfação sendo possível a mudança de cada parte separadamente do sistema sem o comprometer por completo, também abrindo a possibilidade de implementação da operação do sistema em rede.

A plataforma construída para os testes do sistema tem boa reação, e de manufatura de baixo custo.

Os resultados gerais obtidos para o sistema foram satisfatórios onde se construiu um sistema flexível, multiplataforma, de boa performance, de baixo custo, possibilitando a implementação de uma grande gama de técnicas de controle.

## 5.1 Próximos trabalhos

- Implementação de outras técnicas de controle, como a PID e etc.
- Melhor modelagem da função de transferência para a plataforma.
- Implementação de controle para Veículos Autônomos Não Tripulados (VANT).
- Implementação de controle em rede.

## REFERÊNCIAS

ARDUINO. **What is Arduino?**. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>>. Acesso em: 10 out. 2017.

ARMIN RONACHER E CONTRIBUIDORES. **Welcome to Flask**. Disponível em: <<http://flask.pocoo.org/docs/0.12/>>. Acesso em: 10 out. 2017.

BODIN, Erik; STENHOLM, Fanny. **Modelling & Control of a 3DOF Helicopter**. 2015.

CHRIS LIECHTI. **Welcome to pySerial's**. Disponível em: <<https://pythonhosted.org/pyserial/>>. Acesso em: 10 out. 2017.

DINIZ, P. S. R.; SILVA EDUARDO ANTÔNIO BARROS DA.; NETTO, S. L. **Processamento digital de sinais: projeto e análise de sistemas**. Tradução . [s.l.] Bookman, 2004.

FENOPIX. **Introduction to CanvasJS JavaScript Charts**. Disponível em: <<https://canvasjs.com/docs/charts/intro/>>. Acesso em: 10 out. 2017.

GITHUB. **Electron Documentation**. Disponível em: <<https://electronjs.org/docs>>. Acesso em: 10 out. 2017.

GOOGLE. **AngularJS API Docs**. Disponível em: <<https://docs.angularjs.org/api>>. Acesso em: 10 out. 2017.

HALLIDAY, David; RESNICK, Robert; WALKER, Jearl. **Fundamentos de física 1: mecânica**. Tradução de Ronaldo Sérgio de Biasi. 10. ed. Rio de Janeiro: LTC, 2016. v. 1, 327 p., il., color., 28 cm. ISBN 978-85-216-3035-7.

MARK OTTO, JACOB THORNTON, E CONTRIBUIDORES DO BOOTSTRAP . **Bootstrap Introduction**. Disponível em: <<https://getbootstrap.com/docs/4.0/getting-started/introduction/>>. Acesso em: 10 out. 2017.

MARKUS OTÁVIO. **Circuitos elétricos: corrente contínua e corrente alternada, teoria e exercícios**. Tradução . [s.l.] Editora Érica, 2011.

LJUNG, L. **System identification: theory for the user**. Tradução . [s.l.] Prentice Hall PTR, 2012.

MOZILLA E CONTRIBUIDORES. **CSS basics**. Disponível em: <[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/CSS\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics)>. Acesso em: 10 out. 2017.

NEMATI, A.; KUMAR, M. **Modeling and control of a single axis tilting quadcopter**. 2014 American Control Conference, 2014.

OGATA, Katsuhiko. **Discrete-time control systems**. 2. ed. New Jersey: Prentice-Hall, 1995. 745 p., il., 24 cm. ISBN 0-13-034281-5.

OGATA, Katsuhiko. **Engenharia de controle moderno.** Tradução de Heloisa Coimbra de Souza. 5. ed. São Paulo: Pearson Prentice Hall, 2010. 809 p., il., 27 cm. ISBN 978-85-7605-810-6.

PRESSMAN, Roger S; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. Tradução de Ariovaldo Griesi. 8. ed. Porto Alegre: AMGH, 2016. il., 27 cm. ISBN 978-85-8055-533-2.

PYTHON SOFTWARE FOUNDATION. **Python 3.6.3 documentation.** Disponível em: <<https://docs.python.org/3/>>. Acesso em: 10 out. 2017.

SOCKET.IO. **Welcome to Flask-SocketIO's documentation.** Disponível em: <<https://flask-socketio.readthedocs.io/en/latest/>>. Acesso em: 10 out. 2017.

W3C. **HTML5.** Disponível em: <<https://www.w3.org/TR/html5/>>. Acesso em: 10 out. 2017.

## **APÊNDICE A – Código fonte do projeto.**

O código do projeto foi disponibilizado utilizando a ferramente GitHub, que se encontra no seguinte link <https://github.com/alvaroandrioli/tcc>.

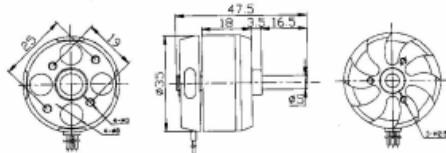
## **ANEXO A – Manual de utilização ESC.**

Link para o manual ESC utilizado no projeto  
<https://hobbyking.com/media/file/1006513628X70599X0.pdf>.

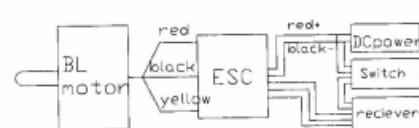
## ANEXO B – Referência motor brushless.

### D3530 Out Runner Brushless Motor Instruction

Dimension



connection



Congratulation for your purchase of high performance brushless outrunner motor series.

Our motors are designed to use perfectly with our own electronic speed controllers. But it is also possible to use them with most of other common electronic speed controllers. The motors are designed for running clockwise or counterclockwise. To change the turning direction you just simply exchange the red and yellow connection wire. If you want to use other speed controllers, please refer to your controller instruction manual.

### D3530(2808) SERIES OUTRUNNER BRUSHLESS MOTOR

Model	Voltage	KV (rpm/v)	Max Pull	Weight	Motor size	Shaft	Max Powe	ESC	Battery/Prop
D3530-8	7.4~15V	1700	1240g	74g	Φ35*30mm	Φ5.0*47.5mm	480watt	50A	LiPo2 / 9x4.7
D3530-10		1400	1180g				446watt		LiPo4 / 7x3
D3530-14		1100	1100g				313watt		LiPo2 / 10x6 LiPo4 / 7x3 LiPo2 / 12x6 LiPo4 / 8x4

#### Product description

1. Mount your motor with the includes screws tightly on your front cover of model.
2. Our motors provide high efficiency stator design.
3. Small size, lightweight and long life.

#### ATTENTION:

1. Please make sure your motor has enough cooling while running. Consider to put a hole into your model motor cover to improve the ventilation.
2. Our motors providing best performance with the recommended propeller size. If you consider to change the propeller size be aware that motor can overheat and damaged.
3. Keep motors away from moisture, dust, scrap and small items to avoide damages.  
We wish you many happy landings with our long life quality equipment.