UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Análise da Implementação de um Sistema Embarcado de Identificação de Parâmetros para Ajuste de Controlador PID

Autor: Ezequiel Kneip Maraschin

Orientador: Marcelo Götz

Porto Alegre, julho de 17

Sumário

Agradecimentos

Gostaria de agradecer a todos que contribuíram para que este trabalho se tornasse realidade, à UFRGS pela formação que me foi dada, aos verdadeiros professores que possuem o desejo de engrandecer o intelecto de seus alunos, ao invisível e imensurável que permeia nossa experiência na busca pelo conhecimento e, por fim, à minha família que me manteve de pé em meio às diversas turbulências.

Resumo

A prototipagem de novos sistemas é uma necessidade constante da indústria, visando aprimorar seus processos, reduzir custos e aumentar seu mercado. O desenvolvimento de sistemas utilizando a abordagem de projeto baseado em modelos é uma ferramenta que capacita o projetista a verificar e validar sua modelagem dos sistemas, avançando categoricamente até o produto final. Neste trabalho, foi utilizada esta metodologia na implementação de um ensaio de identificação de sistemas, dentro do escopo de uma tese de doutorado, utilizando um sistema embarcado para obter os parâmetros necessários para o ajuste automático de um controlador PID. Para a validação da metodologia foram utilizadas duas plantas resistivo-capacitivas e duas funções de ordem fracionária, sendo estas últimas as quais compõem o ensaio de identificação utilizado. Os modelos foram desenvolvidos no Simulink/Matlab e foram avaliadas suas implementações em um Arduino Due.

Abstract

The prototyping of new systems is a constant need of the industry, aiming to improve processes, reduce costs and increase its market. System development using model-based design approach is a tool that enables the designer to verify and validate their modeling of the systems, advancing categorically to the final product. In this work, this methodology was used in the implementation of a system identification test, in the scope of a doctoral thesis, using an embedded system to obtain the necessary parameters for automatic tuning of a PID controller. For the validation of the methodology, two capacitive resistive plants and two fractional order functions were used, the latters being part of the identification test used. The models were developed in Simulink/ Matlab and their implementations were evaluated in an Arduino Due.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

s – Segundos

V – Volts

m – Mili

μ – Micro

w – Frequência angular [rad/s]

k – quilo

Hz – Hertz

Kp – Ganho proporcional

Ti – Constante de tempo da ação integral

Td – Constante de tempo da ação derivativa

Lista de Abreviaturas e Siglas

FOI – Fractional Order Integrator

PID – Proportional Integral Derivative

MiL – Model in the Loop

SiL – Software in the Loop

PiL – Processor in the Loop

HiL – Hardware in the Loop

EFO – Extended Forced Oscillation Method

FPP – Fractional Power Pole

TLC – Target Language Compiler

DAC – Digital to Analogic Converter

AD – Conversor Analógico Digital

LSB – Least significant bit

RC – Resistivo Capacitiva

ZPK – Zero-Pole-Gain (Zero-Polo-Ganho)

1 Introdução

Sistemas embarcados são aqueles onde o sistema computacional é dedicado para uma única aplicação. Neste contexto, diversos dispositivos eletrônicos são desenvolvidos, com o intuito de oferecer menor tamanho, custo e necessidade de recursos computacionais.

Um sistema operacional é muito complexo e muitas das suas funcionalidades são desnecessárias em diversas aplicações, além de ser custoso em termos de processamento. Por isso, para estas aplicações, são utilizadas ferramentas de projeto e simulação que permitem o desenvolvimento de sistemas com menor custo computacional em processadores de menor porte que serão mais baratos e funcionais.

O Simulink é uma ferramenta poderosa de simulação e projeto de sistemas de controle, entre outras finalidades. Com ela, é possível validar diferentes abordagens de controle em um sistema. Além disso, possui ferramentas de geração de código C/C++ para sistemas embarcados, o Simulink Coder e o Embedded Coder (MATLAB & Simulink, 2015), as quais possuem bibliotecas para integração com um compilador para o código que poderá ser carregado diretamente em um processador de menor porte.

O Arduino é uma plataforma de desenvolvimento que integra diversas funcionalidades, entre entradas e saídas analógicas/digitais, comunicação facilitada com o computador (Arduino, 2017a). E o mais importante: é livre de direitos autorais – exceto o nome – permitindo assim um baixo custo. Portanto, esta plataforma globalizou-se permitindo o desenvolvimento de milhares de projetos de automação e controle de equipamentos dedicados, os já mencionados sistemas embarcados. Apesar de ser uma plataforma voltada para aplicações amadoras, tem sido utilizada como plataforma de prototipação rápida para sistemas relativamente pequenos.

Portanto, este trabalho propõe fortalecer o campo de estudo de projeto e desenvolvimento de sistemas de controle embarcados, utilizando as ferramentas de geração de código para poder concentrar o poder intelectual no desenvolvimento de um modelo coerente, o qual deverá ser validado e avaliado com a capacidade das ferramentas e dispositivos oferecidos para este fim.

Mais especificamente, os objetivos principais deste trabalho são avaliar o custo computacional do sistema de identificação de parâmetros e posterior implementação destes blocos gerados pelo software de projeto e simulação de sistemas de controle – Simulink – em um sistema embarcado. Para isso, será utilizada a ferramenta Simulink Coder e um Arduino Due (Arduino, 2017b), o qual possui um processador ARM Cortex M3.

O sistema de identificação de parâmetros utilizado, em desenvolvimento no contexto de uma pesquisa de doutorado, é uma extensão do método de oscilação do relé, este último amplamente difundido na teoria de controle moderno (Ogata, 2003). Ele consiste em forçar uma oscilação sustentada em uma planta cuja resposta em frequência não necessariamente cruza o eixo real negativo do plano complexo. Isto é conseguido com a adição de um bloco que insere uma diferença de fase conhecida no sistema. É importante que este bloco possua uma fase constante em uma ampla faixa de frequência para garantir que a diferença de fase inserida é conhecida na frequência da oscilação sustentada e, portanto, seja possível identificar o sistema com o resultado do ensaio. Com a identificação

dos parâmetros do sistema na frequência de oscilação sustentada, é possível sintonizar um controlador Proporcional Integral Derivativo (PID) com uma margem de fase conhecida.

O trabalho utiliza a metodologia de desenvolvimento baseado em modelos (do inglês, *Model Based Design*). Serão utilizados os conceitos de *Model in the Loop* (MiL), *Software in the Loop* (SiL), *Processor-In-the-Loop* (PiL) e *Hardware in the loop* (HiL) para avaliar desempenho e corretude do processamento no sistema embarcado para um bloco integrador de ordem fracionária (*Fractional Order Integrator* – FOI), o qual é a função de maior custo computacional que compõe o sistema de identificação, além de blocos matemáticos adicionais dos modelos. A ferramenta *Embedded Coder* com o auxílio do pacote *Simulink Support Package For Arduino Hardware* faz a interface entre o ambiente de simulação no computador e o processador embarcado, além da geração do código embarcado.

Este trabalho possui a seguinte estrutura: o Capítulo 2 apresenta uma revisão dos principais conceitos utilizados e necessários para a compreensão da proposta de trabalho apresentada, juntamente com suas referências. No Capítulo 3 são descritas as ferramentas de hardware e software utilizadas, bem como a maneira como elas interagem entre si, de forma a compor a estrutura do projeto. No Capítulo 4 são detalhadas a formulação do problema e o estudo de caso, o processo de modelagem do ensaio de identificação e a aplicabilidade das metodologias de projeto. No Capítulo 5 são apresentados os resultados obtidos durante as diferentes etapas da implementação. Por fim, no Capítulo 6 são apresentadas as conclusões obtidas, bem como, indicações e sugestões de trabalhos futuros que contribuam para a continuação de pesquisas na área.

2 Revisão Bibliográfica

2.1 Metodologias de Projeto

O desenvolvimento de sistemas de controle é um processo complexo e bastante peculiar da área de atuação onde o sistema se encontra, podendo ser realizado com foco nos processos contínuos ou discretos. Independentemente do objetivo do sistema, a simulação é uma etapa importante do desenvolvimento, onde pode-se identificar as principais dificuldades de implementação, requisitos de *hardware* e *software* mínimos e também verificação e validação do modelo matemático proposto pelo projetista.

Nestas diretrizes, a simulação pode ser realizada com diferentes abordagens. Em sistemas micro controlados, está se tornando comum o uso de ferramentas de simulação que incluam a capacidade de verificar e validar a funcionalidade do sistema que se deseja implementar no dispositivo a ser utilizado. Alguns destes conceitos são o *Model in the Loop (MiL)*, *Software in the Loop (SIL),* o *Processor in the Loop (PIL)*, e o *Hardware in the Loop (HIL)*, explicados mais detalhadamente a seguir. Estes conceitos são a base do desenvolvimento baseado em modelos (do inglês *Model Based Design*) (Bringmann, 2008).

Uma simulação MiL é definida por Levine (2010) como modelos matemáticos/lógicos de sistemas executando e trocando dados em uma plataforma virtualizada. Atualmente o computador é a principal plataforma onde essa metodologia é executada. Esta simulação é comum em sistemas de controle e pode ser vista da seguinte forma.

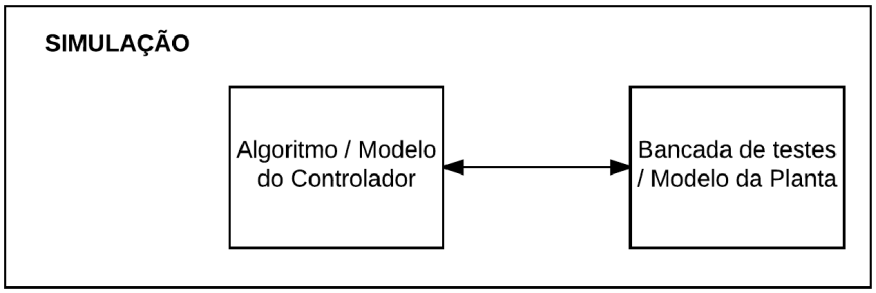


Figura 1: Simulação MiL

Fonte: Autor

Esta forma de simulação é a mais difundida, por ser mais prática e geralmente mais barata que os outros métodos. A dificuldade se encontra em acrescentar as tolerâncias devidas aos modelos utilizados, de forma a aproximar-se ao máximo da realidade da aplicação.

2.1.1 Software in the Loop (SiL)

SiL é uma técnica de desenvolvimento de sistemas embarcados que viabiliza a aferição do código de programa já na linguagem do dispositivo final, comumente chamado de *Target*. Geralmente é utilizado a linguagem de programação C e C++ (Bringmann, 2008). Desta forma, é utilizado um bloco representando o sistema embarcado na malha de simulação e o código é então validado com os resultados obtidos. O Simulink Coder é o responsável por gerar o código que irá rodar no aplicativo em paralelo à simulação do Simulink, emulando o programa que será carregado no sistema embarcado

posteriormente. Cabe salientar que nesta etapa ainda não se utiliza o hardware alvo no processo de simulação.

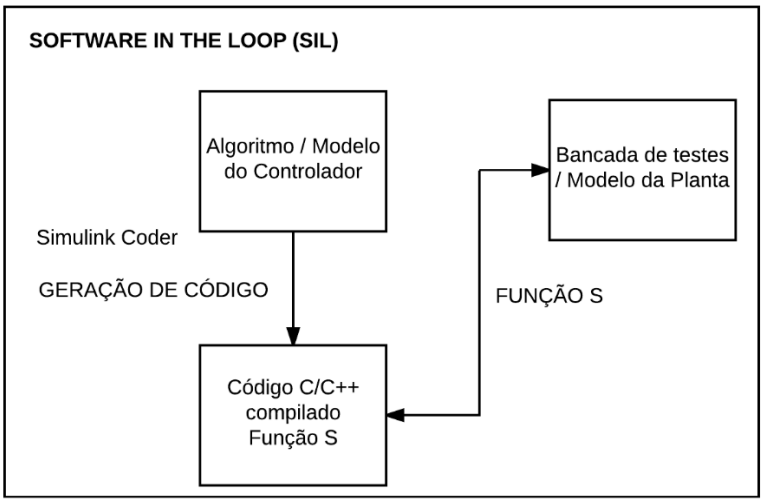


Figura 2: Simulação com geração de código

Fonte: Autor

A Função S mencionada na Figura 2 é uma transcrição em linguagem de computador de um bloco ou script do Simulink escrito em C/C++. Esta função é especialmente útil quando utilizada de acordo com o compilador do *target* desejado, chamado de TLC (*Target Language Compiler*). Assim, é otimizada para reduzir os custos computacionais de acordo com o processador definido.

2.1.2 Processor in the Loop (PiL)

PiL, por sua vez, é uma etapa acima da SiL, uma vez que o código em C/C++ irá rodar diretamente no *Target*. Com a interface de comunicação estabelecida, a simulação acontece concorrentemente ao computador. Os resultados obtidos com esta simulação permitem avaliar o custo de memória e processamento necessários para garantir a precisão numérica e desempenho mínimo desejados ao sistema embarcado. O Embedded Coder é a biblioteca do Simulink responsável por gerar o código do programa e das interfaces de comunicação e monitoramento do processo no microcontrolador. Este método é mais pesado computacionalmente que o SiL, pois são adicionadas rotinas no código para fazer a aferição do desempenho e custo computacional de cada função gerada a partir dos blocos de simulação originais. Além disso, todo o código é carregado e executado no *Target*. A seguir, a Figura 3 apresenta um diagrama que ilustra esta metodologia.

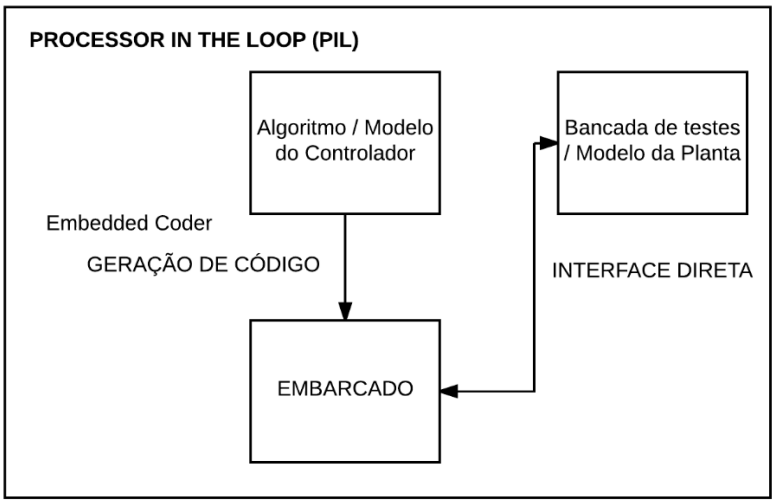


Figura 3: Simulação com o microcontrolador concorrente ao sistema

Fonte: Autor

Uma observação interessante da metodologia PIL é que ela garante o processamento de cada função independente do tempo de amostragem definido para o modelo principal, para garantir que os resultados da simulação estão de acordo com o código gerado. Ou seja, o objetivo principal é avaliar a corretude do modelo sendo executado na plataforma alvo. Em contrapartida, é possível registrar os tempos de processamento de cada função do código C/C++ e assim remodelar o sistema, ou invalidar a escolha da arquitetura alvo, caso o a execução do modelo na arquitetura alvo esteja sobrecarregando a mesma. Uma vantagem clara desta etapa é que esta identificação ocorre ainda na etapa de prototipagem, diminuindo os riscos e gastos em aquisição de material que pode não comportar o modelo especificado para o projeto.

2.1.3 Hardware in the Loop (HiL)

A última etapa de implementação antes da fase final de testes – onde todo o processo já acontece fora do computador – é chamada de *Hardware in the loop* (HIL), a qual propõe-se a integrar também entradas e saídas analógicas ou digitais ao processamento no sistema embarcado, impondo uma situação de tempo real. Nesta metodologia, sensores e atuadores fazem parte da malha de simulação e trocam sinais com o microcontrolador, então é possível identificar peculiaridades de hardware não existentes nas etapas anteriores.

Essa é a última metodologia onde a interface principal de simulação acontece ainda no sistema computadorizado, no caso, o modo *External* do Simulink, juntamente com as bibliotecas de geração de código utilizadas pelo Embedded Coder, fazem o carregamento do modelo do sistema, incluindo a capacidade de monitoramento e registro das variáveis desejadas do microcontrolador para o ambiente de simulação. Na Figura 4 está ilustrado um diagrama que representa a interação entre os sistemas nesta metodologia. Em algumas plataformas de simulação, o sistema sob controle ainda é simulado. Porém, para permitir sua interação em tempo real com o controlador (sendo executado na plataforma alvo), necessita-se de um sistema especial.

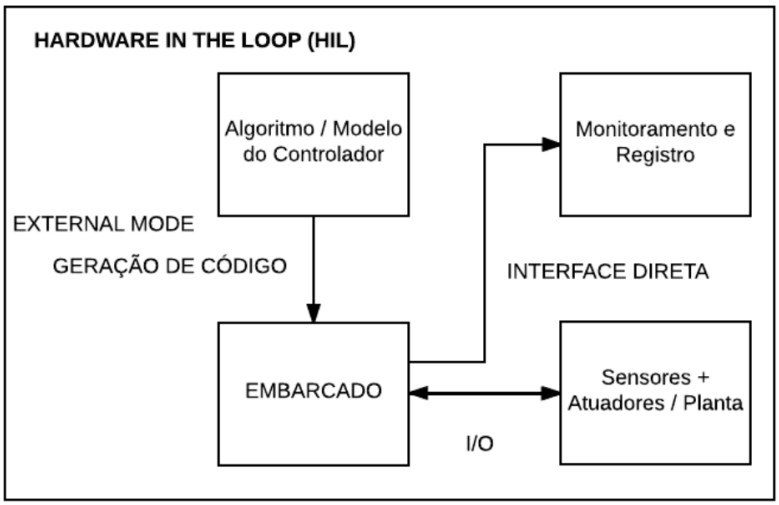


Figura 4: Diagrama exemplo da metodologia *HIL*

Fonte: Autor

2.2 Ensaio de identificação

Foi utilizado neste trabalho uma modificação do método clássico do relé, a qual foi denominada *Extended Forced Oscillation Method* (EFO) (Bazanella et. al, 2016), este último sendo tema de pesquisa atual de doutorado, e baseia-se no método clássico pois também utiliza um ponto da resposta em frequência da planta para realizar o ajuste do controlador PID.

O método clássico do relé consiste basicamente em identificar o ponto de frequência máximo da planta – frequência ressonante – onde a resposta em frequência da função cruza o eixo real negativo no diagrama de *Nyquist* (Ogata, 2003). Assim, são identificados um ganho e uma frequência deste ponto. Então é projetado um controlador por alocação de polos com o ganho identificado anteriormente, igualando o mesmo à uma estrutura de controle PID demonstrada por Ogata (2003) para obter os ganhos integral, diferencial e proporcional. Plantas que não possuam o ponto de frequência máximo não são identificáveis através deste método, o que limita sua aplicabilidade. Na Figura 5 a seguir, está ilustrado o ensaio clássico do relé em um modelo do Simulink.

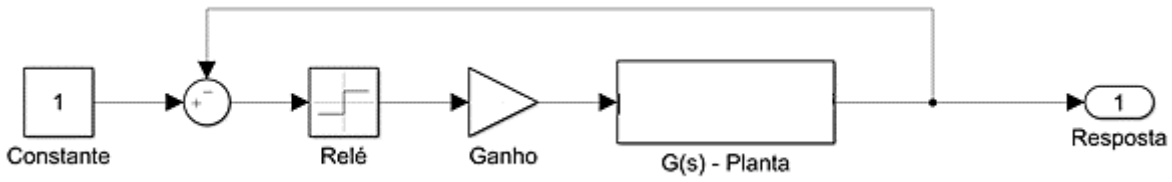


Figura 5: Modelo utilizando o ensaio clássico do relé

Fonte: Autor

Expandindo o ensaio clássico do relé para o estendido, este último propõe-se a identificar uma classe de plantas mais abrangente ao utilizar qualquer ponto da resposta em frequência da planta para ajustar o controlador e garantir uma margem de fase

específica. Isto é obtido de forma semelhante ao método clássico, onde se garante um desempenho mínimo conhecido a partir de um ponto da resposta em frequência da planta identificado pelo ensaio. A diferença é que, neste caso, inserimos uma função de transferência que acrescenta a fase necessária na resposta em frequência do sistema para que o mesmo cruze o eixo real negativo e o faça oscilar. Assim, sabendo-se a fase da função inserida no modelo na frequência de oscilação, pode-se identificar a fase do sistema e garantir a margem de fase necessária projetando o controlador conforme segue. Na Figura 6, apresenta-se um modelo do ensaio estendido em Simulink.

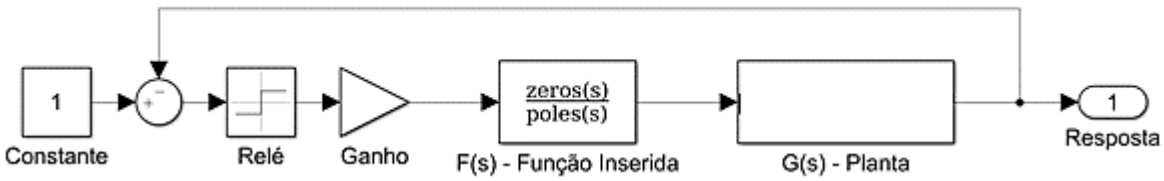


Figura 6: Modelo utilizando o ensaio estendido do relé

Fonte: Autor

Bazanella et. al (2016) afirma que se a resposta em frequência de uma planta não cruza o eixo real negativo, a margem de ganho é infinita, desde que o controlador não contribua com um atraso de fase muito grande. Portanto, para esta classe de plantas, obter uma margem de fase segura é o único objetivo do projeto do controlador. Dessa forma, a identificação do sistema ocorre em um ponto da resposta em frequência onde essa margem de fase é garantida e então se projeta o controlador com as seguintes premissas: a contribuição de fase do controlador naquela frequência é pequena; e a magnitude da função de transferência em malha fechada do sistema nesta mesma frequência é unitária. Se conseguirmos desenvolver este controlador, teremos, sendo θ a margem de fase desejada:

$$C(jw_\theta)G(jw_\theta) = 1\angle\theta \tag{1}$$

Identificando o sistema na frequência w_θ, obtém-se a amplitude e a fase da variável $G(jw_\theta)$ e, assim, o controlador se resume a:

$$C(jw_\theta) = \frac{1\angle\theta}{G(jw_\theta)} = \frac{1}{M_\theta}\angle 0 \tag{2}$$

Onde M_θ é o módulo da resposta em frequência da planta no ponto de fase θ, o ponto que foi identificado com a margem de fase desejada.

É necessário especificar um valor para a margem de fase desejada. Wolovich (1993) sugere utilizar 45° para conciliar robustez com desempenho transitório para diferentes situações práticas. Bazanella (2016) realizou diversos testes com plantas similares à deste trabalho utilizando uma margem de fase igual a 60° e atestando que este também é um valor válido, principalmente para situações onde não se sabe a priori se o sistema possui uma frequência máxima e qual sua margem de fase para ela.

Definindo a utilização de uma margem de fase igual a 60°, a oscilação sustentada do sistema em malha fechada consiste em identificar o ponto da resposta em frequência do

sistema onde a fase é igual a 120°, pois assim é possível ajustar o controlador com as premissas anteriores e garantir a margem de fase especificada.

Para ajustar o controlador PID que satisfaça a Equação (2), é necessário que o atraso de fase fornecido pelo bloco Proporcional Integral (PI) seja de mesma grandeza que o avanço de fase fornecido pelo bloco Proporcional Derivativo (PD) naquela frequência. Da mesma forma, a magnitude acrescida por um deverá ser diminuída pela do outro. Fórmulas clássicas como Ziegler-Nichols utilizadas pela literatura sugerem uma contribuição de -10° para um controlador PI nas frequências máximas de operação da planta (Ogata, 2003).

Sendo a amplitude da resposta em frequência da planta no ponto de margem de fase igual a 120° definido por M_{120}, o projeto do controlador é dividido pelo bloco PI e PD conforme segue.

$$C_{PI}(s) = K_p \left(1 + \frac{1}{T_i s}\right) = \frac{1}{M_{120}} \angle(-10°) = K_p - j\frac{K_p}{w_{120} T_i} \tag{3}$$

Igualando as partes reais e imaginárias obtém-se:

$$K_{PI} = \frac{\cos(10°)}{M_{120}} \;\; ; \;\; T_i = \frac{1}{w_{120}\tan(10°)} = \frac{T_{120}}{2\,\pi\tan(10°)} \tag{4}$$

Para o bloco PD, tem-se para a fase:

$$\angle 1 + \mathrm{j}w_{120}T_d = \arctan(w_{120}T_d) = 10° \rightarrow T_d = \frac{\tan(10°)}{w_{120}} = \frac{\tan(10°)}{2\pi} T_{120} \tag{5}$$

E para o ganho fica:

$$K_{PD} = \sqrt{1 + (w_{120}T_d)^2} = \sqrt{1 + tan^2(10°)} = \frac{1}{\cos(10°)} \tag{6}$$

Desta forma, para se garantir o exposto em (2), é necessário que o ganho do PI seja reduzido pelo ganho do PD, resultando por fim em:

$$K_{PID} = \frac{cos^2(10°)}{M_{120}} \tag{7}$$

Assim, utilizando as equações (4), (5) e (7) é possível ajustar o controlador PID a partir do conhecimento do ponto da resposta em frequência da planta onde a contribuição de fase é igual a -120°. Para identificar este ponto, a função F(s) inserida no método é o integrador de ordem fracionária (FOI) com uma contribuição de fase de -60° aproximadamente constante em uma grande faixa de frequências. Assim, a oscilação, que ocorre no ponto de -180° de fase, estará na frequência onde o FOI deve possuir -60° e a planta -120° de fase. Bazanella (2016) expõe que se obtém a amplitude da resposta em frequência da planta com a seguinte fórmula.

$$|G(jw_1)| = \frac{\pi A}{4\mathrm{d}|\mathrm{F}(\mathrm{j}w_1)|} \; ; \angle G(jw_1) = -180° - \angle F(jw_1) \tag{8}$$

Sendo,

A = Amplitude do sinal oscilatório sustentado

d = Ganho do relé
w_1= Frequência da oscilação

Determina-se assim o período da oscilação e sua amplitude, também garantindo que a contribuição de fase do FOI naquela frequência possui a contribuição de fase desejada, é possível realizar o ajuste do controlador PID com as formulações acima explicitadas. Na Tabela 1 a seguir, estão listadas as fórmulas resumidas para a obtenção dos parâmetros do PID, considerando as premissas do controlador assumidas anteriormente.

Tabela 1: Fórmula para os parâmetros do PID

Fonte: Bazanella (2016)

K_P	$T_i\ [s]$	$T_d\ [s]$
$\frac{0{,}97}{M_{120}}$	$0{,}90T_{120}$	$0{,}028T_{120}$

2.3 Integrador de Ordem Fracionária (FOI)

Um integrador de ordem fracionária é uma função matemática irracional com a seguinte representação.

$$I_m(s) = \frac{1}{s^m} \tag{9}$$

Onde s é um número complexo e m é um número fracionário. Desta forma, temos uma função de transferência irracional e a resposta em frequência da mesma não pode ser obtida através de métodos convencionais. Existem diversas formas de representar este tipo de função na literatura, dentre elas a representação Polo de Potência Fracionária (*FPP – Fractional Power Pole*), utilizada por Charef (1992), que demonstra que estes sistemas podem ser representados por uma rede em cascata de circuitos RC. A aproximação utilizada considera um produtório de polos e zeros que representam as principais singularidades observadas na resposta em frequência de um polo fracionário. Cada conjunto de ganho, zero e polo é um circuito RC que, combinados, resultam na resposta em frequência aproximada da função.

São definidos parâmetros de tolerância e distância entre os conjuntos de singularidades utilizados, além das frequências mínimas e máximas onde deseja-se obter uma contribuição de fase constante. A construção da função de transferência se inicia com a ideia de manter o ganho de $-20m$ dB/década na resposta em frequência em uma banda de passagem específica. Dessa forma, a contribuição de fase do sistema não se altera nesta banda. Para isso, são concatenadas diversas singularidades (par zero polo) que juntas formam um zig-zag de respostas alternadas de 0 dB/década a -20 dB/década em torno da curva de $-20m$ dB/década ideal na faixa de frequência desejada. As tolerâncias definem as distâncias entre as singularidades e o número delas necessário para compor a resposta estipulada em projeto. Quanto menores as tolerâncias, maior será o número de singularidades necessários para compor a função de transferência final. No apêndice está exposto o cálculo da função de transferência em um *script* do MatLab. A seguir, estão explicadas as principais equações que fundamentam a construção do modelo.

Tendo então a representação do polo fracionário conforme segue.

$$H(s) = \frac{1}{\left(1+\frac{s}{p_T}\right)^m} \approx \frac{\prod_{i=0}^{N-1}\left(1+\frac{s}{z_i}\right)}{\prod_{i=0}^{N}\left(1+\frac{s}{pz_i}\right)} \tag{10}$$

Onde p_T é constante de tempo e m é a fração da potência entre 0 e 1. Como dito anteriormente, uma curva com $-20m$ dB/década pode ser aproximada por um número de linhas de zig-zag alternadas entre 0 dB/década e -20 dB/década, representados pelos produtórios de polos e zeros da Equação 10. Como o comportamento em frequência que desejamos na curva é em uma banda finita, uma aproximação de grandeza N é suficiente.

Assumindo então uma tolerância y (decibel) para o erro entre as linhas zig-zag e a linha desejada. Tem-se para os polos e zeros conforme segue.

$$\begin{gathered}
p_0 = p_t 10^{\left\lceil \frac{y}{20m} \right\rceil} \\
z_0 = p_0 10^{\left\lceil \frac{y}{10(1-m)} \right\rceil} \\
p_1 = z_0 10^{\left\lceil \frac{y}{10m} \right\rceil} \\
z_1 = p_1 10^{\left\lceil \frac{y}{10(1-m)} \right\rceil} \\
\vdots \\
z_{N-1} = p_{N-1} 10^{\left\lceil \frac{y}{10(1-m)} \right\rceil} \\
p_N = z_{N-1} 10^{\left\lceil \frac{y}{10(1-m)} \right\rceil}
\end{gathered} \tag{11}$$

Onde p_t é a frequência de canto e é determinada no ponto de -3 dB da função de transferência original, p_0 é a primeira singularidade e é determinada pelo erro y e p_N é a última singularidade determinada por N.

Como o erro y é constante entre as singularidades, Charef (1992) utiliza uma progressão geométrica entre as singularidades e resume o cálculo da função de transferência aproximada a partir da Equação 10.

O número de singularidades N é determinado pela frequência máxima desejada na banda finita da resposta em frequência da função, em conjunto com a frequência inicial e o passo entre cada singularidade determinado pela tolerância y. O cálculo preciso do número de singularidades encontra-se no apêndice do mesmo artigo (Charef, 1992).

3 Materiais e Métodos

Neste trabalho realizou-se uma combinação de software e hardware de forma a validar e analisar a usabilidade de um procedimento teórico – com a utilização de um bloco de integração fracionária – da simulação até o sistema físico real. Para isso, estão listados a seguir as plataformas utilizadas.

Utilizou-se para os procedimentos deste trabalho um Arduino Due, como plataforma alvo, e um computador portátil de 64 *bits* utilizando o sistema operacional Windows 10 Pro, como plataforma de simulação de modelos, com a seguinte estrutura de *software* e bibliotecas:

- MATLAB R2017a 9.2
    - *Support for MinGW-w64 compiler 4.9.2 (versão 17.1.0)*: Compilador necessário para a geração do código de baixo nível que será carregado no microcontrolador do embarcado (arquitetura alvo).
- Simulink 8.9
    - *Simulink Coder*: Responsável por gerar código C/C++ nativo a partir dos modelos do Simulink.
    - *Embedded Coder*: Responsável por gerar código C/C++ nativo a um microcontrolador específico, neste trabalho foi utilizado o ARM Cortex M3.
    - *Simulink Support Package for Arduino Hardware (versão 17.1.0)*: Responsável por realizar a interface de comunicação e instrumentação entre o Arduino e o PC, criando as bibliotecas necessárias para a tradução das funções matemáticas e lógicas equivalentes aos dois sistemas, também possibilitando a monitoração do tempo de processamento das mesmas com comunicação *online* durante a simulação.
- Arduino IDE 1.6.4
    - *Arduino Due driver*: Responsável por garantir a correta identificação do dispositivo através da porta serial do computador, incluindo a organização das chamadas e respostas a esta porta tendo em vista o sistema micro controlado conectado.

O computador portátil possui uma boa arquitetura (*Quad-Core* e +4GB de RAM). O Arduino Due é composto por um microcontrolador AT91SAM3X8E de 32 bits. Este Arduino é o de maior capacidade de processamento do mercado atual, e ainda é considerado de baixo custo, sendo este de pouco mais de cem reais, viabilizando sua utilização em larga escala para prototipagem e validação de novos sistemas. Mais detalhes desta plataforma podem ser encontrados no site do fabricante (Arduino, 2017).

O método utilizado para verificar a viabilidade da implementação do ensaio de identificação no sistema embarcado foi seguir a metodologia de desenvolvimento baseado

em modelos, onde os algoritmos e funções matemáticas são verificados e validados em diferentes etapas, cada uma avançando cada vez mais perto da prototipagem e da execução em tempo real do sistema.

Foram utilizados os modos de simulação descritos no Capítulo 2, sendo eles a simulação MiL, que visa validar a modelagem inicial do sistema; a simulação em SiL, que visa validar a corretude do algoritmo em C/C++ gerado pelo software; a simulação em PIL, que visa validar a corretude do algoritmo rodando diretamente no sistema embarcado em concorrência com a simulação no computador, incluindo a capacidade de monitorar os tempos das funções executadas pelo microcontrolador; por fim a simulação HIL, que possibilita a troca de sinal com sensores e atuadores durante a execução da simulação, possibilitando a inspeção e registro da sensibilidade dos conversores AD e DA utilizados, inclusive a capacidade de resposta do embarcado de forma mais completa. A última etapa da implementação seria a prototipagem do circuito, visando a utilização somente do sistema embarcado executando em tempo real sem intervenção do computador. Esta última etapa não foi desenvolvida neste trabalho, mas é uma sugestão de trabalhos futuros mencionada no Capítulo 6.

Para validar os valores identificados pelo ensaio com sinal oscilatório, foi realizado o projeto do controlador PID a partir do demonstrado no Capítulo 2 e comparado o desempenho dos sistemas na resposta a uma referência do tipo degrau em malha aberta e malha fechada. Este processo foi realizado no MATLAB e no Simulink.

4 Formulação do Problema e Estudo de Caso

Como visto anteriormente, plantas que não possuem o ponto de frequência ressonante não podem ser identificadas a partir do método clássico do relé, pois não geram a oscilação sustentada na sua saída. Para complementar o método, de forma a agregar estas classes de sistemas, fez-se a adição do bloco de integração fracionária no ensaio. Este bloco, contudo, possui uma representação matemática complexa e de difícil implementação prática. Essa difícil implementação prática é um dos problemas a ser analisado neste trabalho.

Realizou-se o processo de implementação em um sistema embarcado partindo-se desde a simulação pura em modelos matemáticos rodando no computador até a simulação em tempo real com troca de sinais entre a planta e o microcontrolador.

Neste trabalho, a própria saída analógica disponível no Arduino Due – um *DAC*, conversor analógico digital – foi utilizada como atuador do sistema. O sensor utilizado para medir a tensão de saída foi a entrada analógica do mesmo Arduino, assim simplificando o sistema para concentrar-se na viabilidade da aplicação com as metodologias explicadas no Capítulo 2.

Utilizando duas plantas RC, uma de segunda e outra de primeira ordem, realizou-se a identificação do período e da amplitude da oscilação sustentada de cada sistema com um bloco FOI para cada (-60° e -120° de fase, respectivamente) e posteriormente calculou-se os ganhos para o ajuste do PID. Para isso, validou-se a modelagem do sistema de acordo com as metodologias de projeto já discutidas. Após a validação, analisou-se os resultados e comparou-se o desempenho das plantas com os controladores projetados.

4.1 Modelos

4.1.1 Plantas

As duas plantas RC mencionadas anteriormente foram escolhidas visando ganho próximo do unitário e frequência de corte perto de 2 Hz. Os valores comerciais dos resistores e capacitores foram

$$R = 33\ k\Omega$$
$$C = 2.2\ \mu F$$

Na Figura 7 a seguir, está ilustrado o modelo da planta RC de segunda ordem.

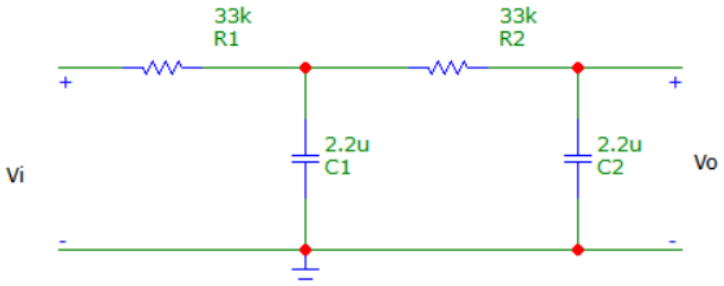


Figura 7: Planta RC de segunda ordem.

Fonte: Autor

O modelo da função de transferência é obtido a partir das Leis de Kirchoff, onde se relaciona a tensão de saída pela tensão de entrada. O equacionamento foi obtido a partir do exposto por Nilsson (2008).

A planta de primeira ordem corresponde ao primeiro estágio da planta anterior, e está ilustrada na Figura 8 a seguir.

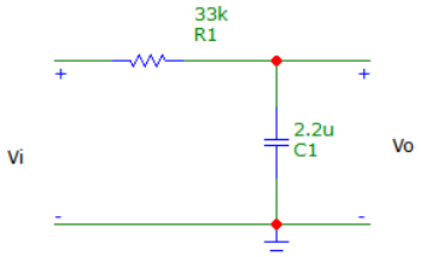


Figura 8: Planta RC de primeira ordem.

Fonte: Autor

Sua função de transferência é calculada da seguinte forma:

$$G(s) = \frac{V_o}{V_i} = \frac{1}{1 + RCs} = \frac{\frac{1}{RC}}{\frac{1}{RC} + s} = \frac{\frac{1}{0{,}0726}}{\frac{1}{0{,}0726} + s} = \frac{13{,}744}{s + 13{,}744} \quad (12)$$

Já o modelo de segunda ordem, então, é dado pela Equação (13) a seguir.

$$G2(s) = \frac{1}{s^2 R_1 R_2 C_1 C_2 + s(R_1 C_1 + R_1 C_2 + R_2 C_2) + 1} = \frac{189{,}72}{s^2 + 41{,}32s + 189{,}72} \quad (13)$$

Pode-se notar que o ganho DC e a frequência de corte de ambos os sistemas são praticamente iguais. Na Figura 9 a seguir, o diagrama de Bode dos dois circuitos.

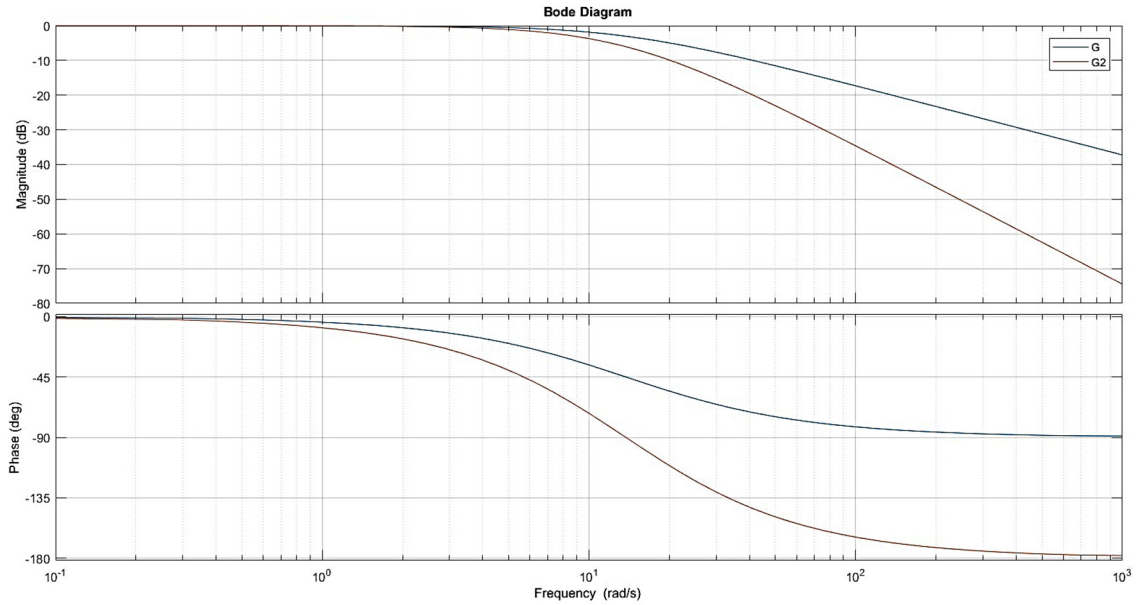


Figura 9: Diagrama de Bode das plantas RCs.

Fonte: Autor

4.1.2 FOI

O modelo do FOI utilizado foi obtido baseado na implementação desenvolvida por Charef (1992) e, para isso, foi utilizado o *script* em MatLab mencionado na Seção 2.3, com a definição das tolerâncias e banda de passagem necessárias para os cálculos das singularidades visando a obtenção de um modelo racional da função. Na Tabela 2, estão demonstrados os polos e zeros da função de transferência já discretizada com até 9 dígitos significativos. Em uma representação do tipo ZPK (zero-pole-gain), basta utilizar os valores da tabela com o ganho K = 0.0001091 para montar a função de transferência.

Tabela 2: Zeros e Polos do FOI de 60°

Fonte: Autor

Zeros	Polos
0,999995395	0,99999799
0,999984033	0,99999303
0,999944638	0,999975833
0,999808052	0,999916207
0,999334602	0,999709489
0,997694712	0,998993051
0,992029385	0,99651287
0,972632056	0,987960627
0,908204579	0,958866012
0,714089796	0,864261538
0,267140355	0,596854785
-0,334526141	0,066451145

Isto significa que o sistema resultante é de décima segunda ordem. O diagrama de bode, por sua vez, é ilustrado pela Figura 2 a seguir.

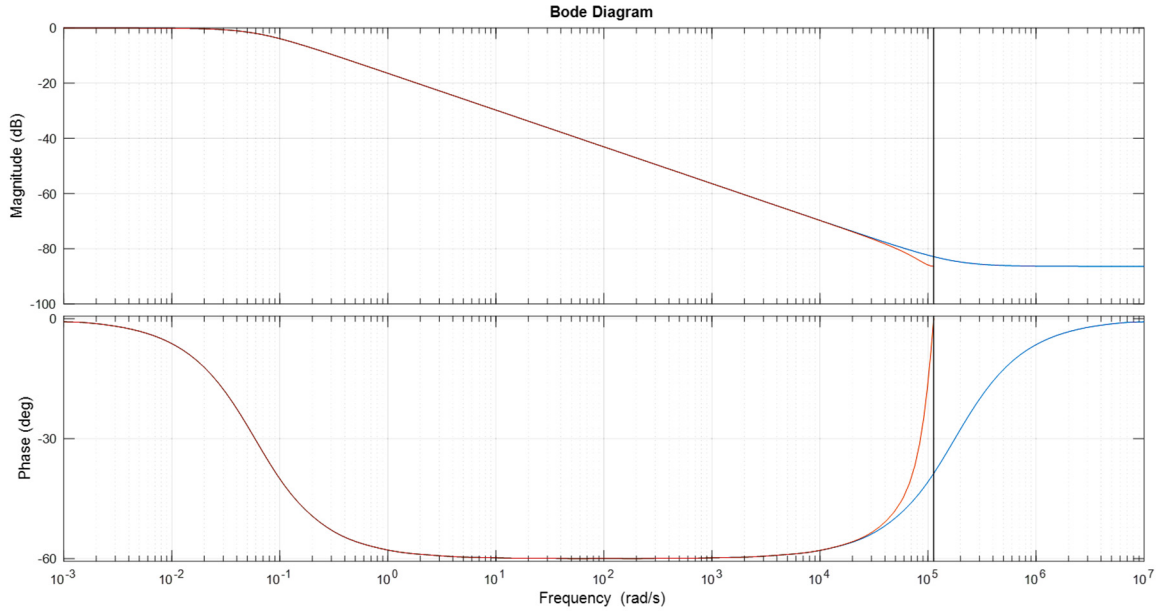


Figura 10: Diagrama de Bode do FOI de 60º de fase

Fonte: Autor

A curva em vermelho é a função discretizada pelo comando *c2d* do MatLab, a qual foi utilizada para a implementação no micro controlador. A curva azul foi a obtida a partir do *script* utilizado. Pode-se observar que a construção do FOI foi bem-sucedida, mantendo-se a fase constante em ampla faixa de frequência, mais especificamente de 1 rad/s até 10000 rad/s, definidos em projeto.

O FOI de -120° de contribuição de fase foi construído com a concatenação de dois blocos FOI de -60°, resultando assim em uma função de transferência de vigésima quarta ordem. O modelo possui o seguinte diagrama de Bode, ilustrado na Figura 11.

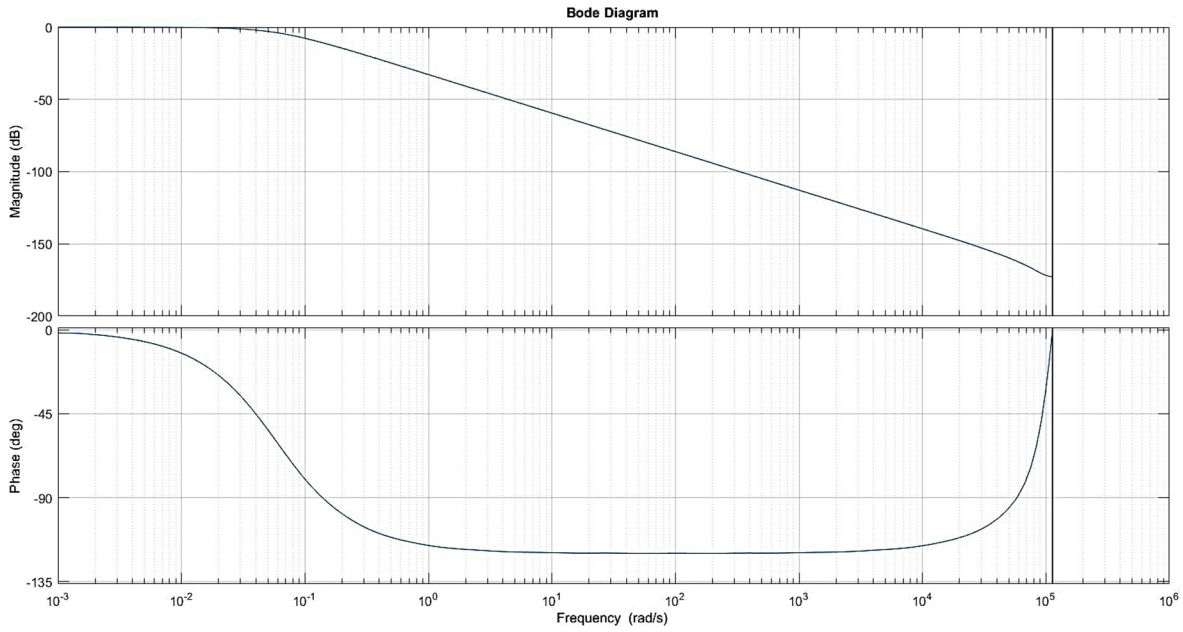


Figura 11: Diagrama de Bode do FOI de -120° de fase

Fonte: Autor

4.1.3 Modelo Completo

O modelo final do ensaio consiste então na realimentação da planta com o relé e o FOI, incluindo a instrumentação necessária para aferir a amplitude e o período da oscilação sustentada na saída do sistema. Para isso, foi desenvolvido um modelo que verifica os máximos e mínimos locais da resposta em tempo de execução, pois desta forma quando o sistema atingir a oscilação sustentada e sua amplitude e período não oscilarem mais, os registros de amplitude e período identificados não sofrerão mais alteração e poderão então serem utilizados para identificar a resposta em frequência da planta.

O modelo ficou dividido em três seções, a primeira é responsável por realizar a oscilação, com o relé e o FOI, a segunda verifica os máximos e mínimos do ensaio e reinicia estes valores a cada ciclo para atentar-se aos máximos e mínimos locais e, por fim, a terceira parte é responsável por aferir o tempo entre cada máximo e mínimo local, determinando assim o período da oscilação. As figuras 12, 13 e 14 a seguir ilustram o modelo do sistema de identificação de parâmetro proposto.

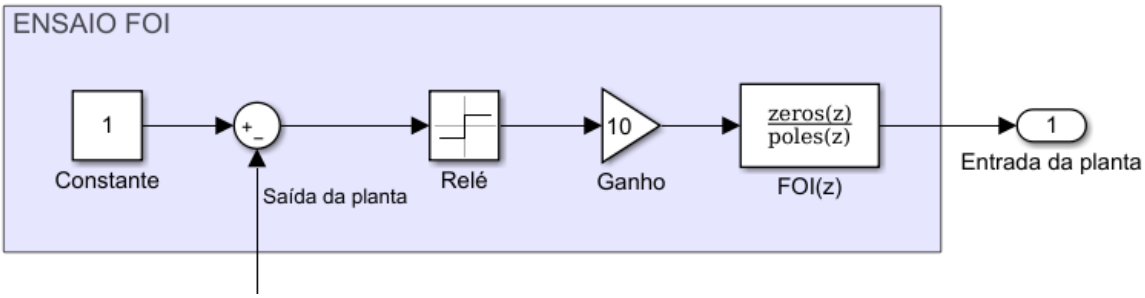


Figura 12: Ensaio oscilatório com o FOI.

Fonte: Autor

O valor constante unitário da Figura 12 é a referência de entrada do sistema, portanto é em torno deste valor que a oscilação ocorrerá. O ganho utilizado visa aumentar a amplitude da oscilação. Este valor de ganho é utilizado para obter a amplitude da resposta em frequência da planta na frequência da oscilação sustentada, conforme visto no Capítulo 2.

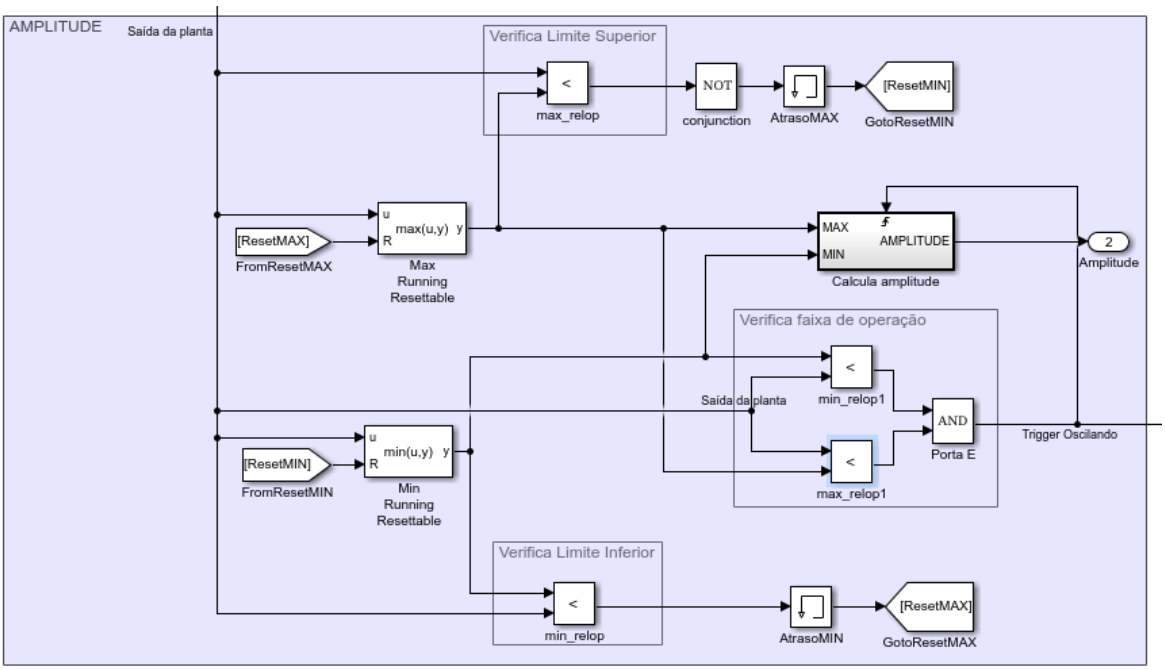


Figura 13: Identificação da amplitude.

Fonte: Autor

Para a identificação da amplitude, o modelo exposto na Figura 13 consiste de três comparadores e de dois registradores, um de valor máximo e outro de valor mínimo. O modelo recebe o sinal de saída da planta, onde a oscilação ocorre, como entrada.

Este sinal é então monitorado para registrar o seu valor máximo e mínimo e então comparado com cada registrador. Se o sinal está abaixo do valor registrado como de valor máximo, é realizado o *reset* do valor mínimo. Se o sinal está acima do valor mínimo, é realizado o *reset* do valor máximo. Desta forma, é garantido que a cada máximo e mínimo locais um novo limite inferior e superior fosse definido, possibilitando assim o registro

dinâmico da amplitude do sinal, o qual tende-se a estabilizar ao longo do ensaio. Somente quando o sinal estiver oscilando dentro de um limite máximo e mínimo identificado, ou seja, estiver em uma oscilação estável, haverá um disparo para a função que calcula a amplitude do sinal, utilizando os limites já identificados. O cálculo da amplitude é realizado pela subtração do valor máximo pelo valor mínimo para obter a amplitude pico-a-pico e então dividir por dois para obter a amplitude de pico do sinal.

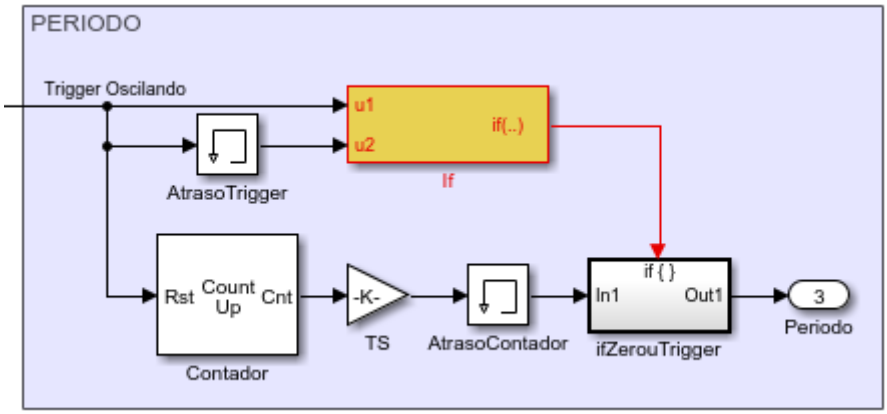


Figura 14: Identificação do período.

Fonte: Autor

Conforme ilustrado na Figura 14, para a obtenção do período da oscilação sustentada foi utilizado o sinal de disparo do modelo da amplitude em um contador, o qual será reinicializado cada vez que o disparo ocorrer, ou seja, cada vez que o sistema completar um ciclo de oscilação dentro dos limites estabilizados. O contador incrementa sua contagem a cada amostra do sinal até sofrer sua reinicialização. Então, para converter o valor para segundos, é necessário multiplicar o sinal proveniente do contador pelo tempo de amostragem. Para realizar o registro apenas do último sinal antes da reinicialização – o período da oscilação de fato – foi criado uma memória do disparo e do sinal do contador, para quando o disparo fosse reinicializado o último valor do contador fosse enviado pela função *IF*.

Pode-se observar que alguns condicionais foram utilizados visando aprimorar o desempenho do sistema. Por exemplo, o sistema que registra os valores de amplitude e do período somente é executado quando o ensaio está dentro de uma faixa de oscilação determinada em tempo de execução pelo algoritmo de identificação dos máximos e mínimos locais.

O modelo principal utilizado faz referência ao modelo do sistema de identificação para poder-se alterar os tipos de simulação desejados conforme a etapa de desenvolvimento. Na Figura 15 a seguir, o modelo de identificação está contido no *Model*, o qual está realimentado com o modelo da planta de segunda ordem.

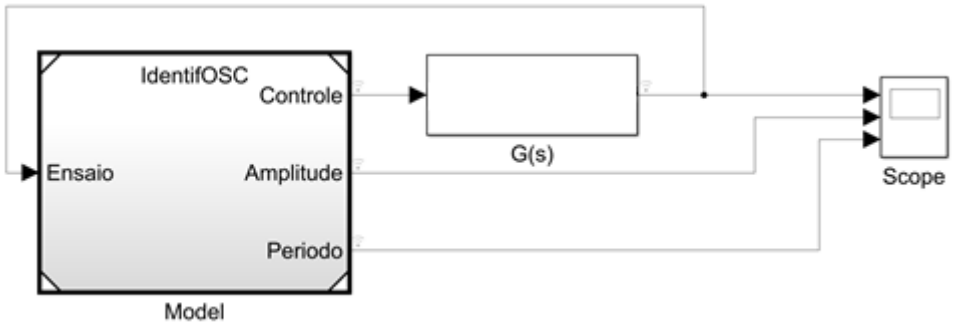


Figura 15: Modelo do ensaio completo em simulação

Fonte: Autor

O bloco G(s) é o modelo matemático da planta, que é necessário para realizar as primeiras etapas das simulações. O modelo de identificação é referenciado pelo modelo principal e pode ter seu modo de simulação alternado em suas propriedades para executar os modos SiL e PiL.

4.2 SiL

A simulação em SiL propõe-se a validar numericamente e logicamente a geração de código C/C++ dos modelos em Simulink, e ainda permite o registro dos tempos de execução dos métodos do código gerado, utilizando o processador do computador.

Para realizar esta simulação é necessário definir o tipo de simulação no modelo referenciado, conforme citado anteriormente, para Software-in-the-loop (SiL). Os blocos utilizados são então discretizados e transcritos da linguagem do Simulink para a linguagem C/C++ definida nos parâmetros do modelo. Por exemplo, na Figura 16 a seguir, é possível definir a biblioteca de substituição de código para as funções usadas pelo GCC do ARM Cortex-M3, que é o processador do Arduino Due.

Para execução do código no microcontrolador, é necessário desabilitar a caixa de seleção *continuous time*, uma vez que somente modelos discretos podem ser carregados no firmware.

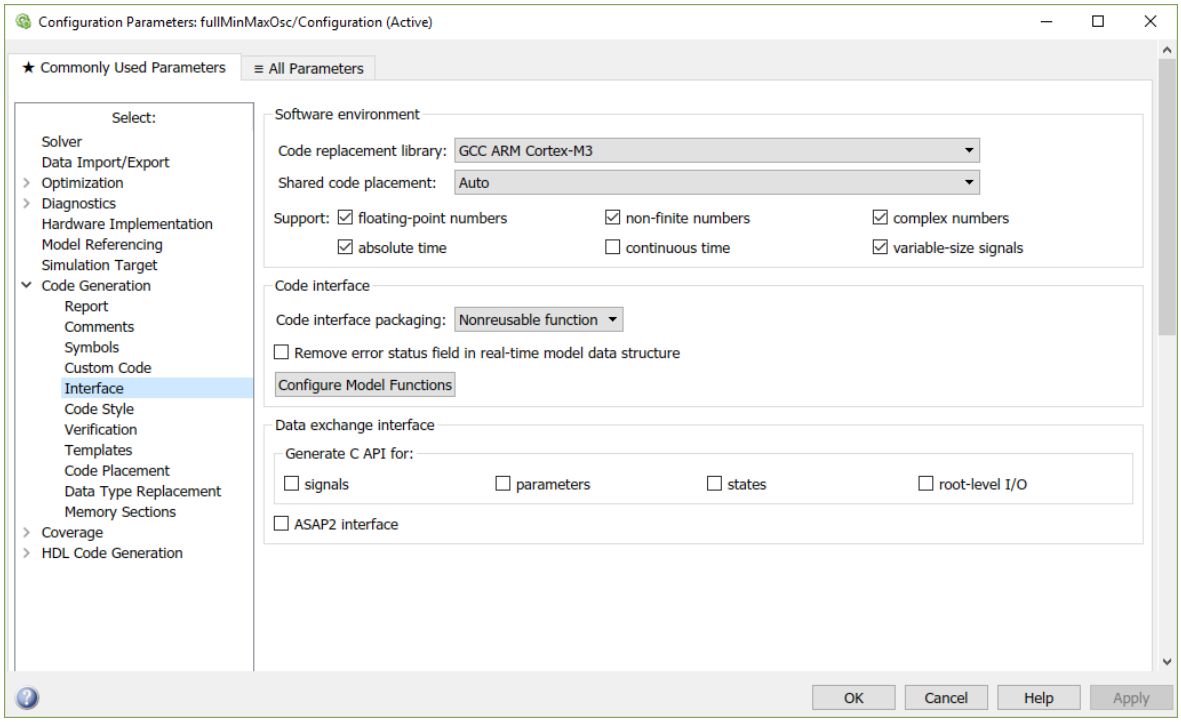


Figura 16: Parâmetros de interface para geração de código do Simulink.

Fonte: Autor

Outras opções, como o perfil do tempo de processamento do código, também foram utilizadas para dimensionar o custo de processamento dos blocos, principalmente do bloco integrador de ordem fracionária. Para isso, na seção *Verification* da geração de código é possível habilitar o registro deste perfil, conforme ilustrado na Figura 17 a seguir.

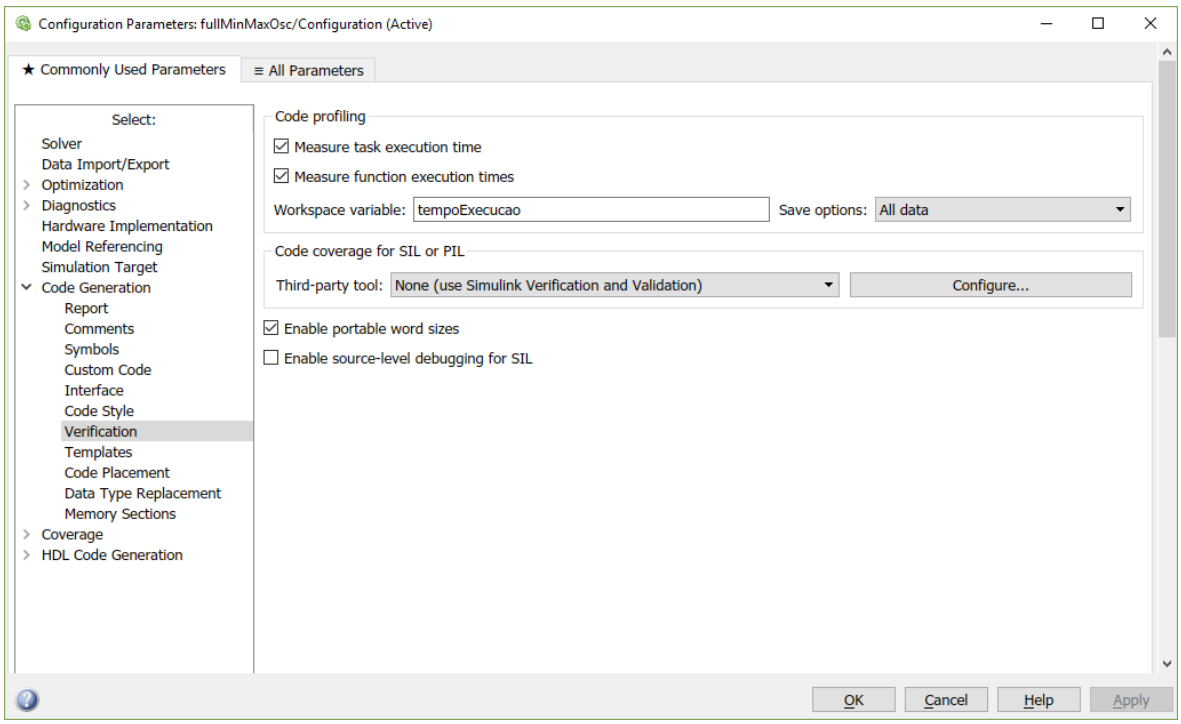


Figura 17: Habilitando o registro do tempo de execução
Fonte: Autor

As opções para registro de tempo de execução são simples e dependem exclusivamente do tipo de simulação que será executado, pois as opções só geram resultados quando definido o modo de simulação para SiL ou PiL. Além disso, é possível também fazer uma análise do código com ferramentas de terceiros, porém neste trabalho foi utilizado somente o Simulink como software de desenvolvimento, verificação e validação.

4.3 PiL

Para o PiL, são necessárias bibliotecas de integração com o microcontrolador, responsáveis por realizar a comunicação – neste trabalho foi utilizada comunicação serial via USB – e registro do desempenho da execução do código no embarcado. A proposta principal do PIL é avaliar a precisão numérica e condições de processamento para realizar as tarefas necessárias, levando em consideração os requisitos do projeto e do modelo a ser implementado. Desta forma, os resultados desejados baseiam-se em estimar o tempo de processamento de cada bloco, o custo em memória da sua alocação, e se os resultados estão iguais aos obtidos nas etapas anteriores, indicando que o modelo está robusto o suficiente para rodar no embarcado. Estes valores são adquiridos através da execução do código gerado na arquitetura alvo.

Da mesma forma que os modos anteriores, para habilitar a simulação em PiL deve-se definir no modelo referenciado o tipo desejado. A Figura 18 a seguir ilustra como aparenta um modelo referenciado definido para simular em PiL.

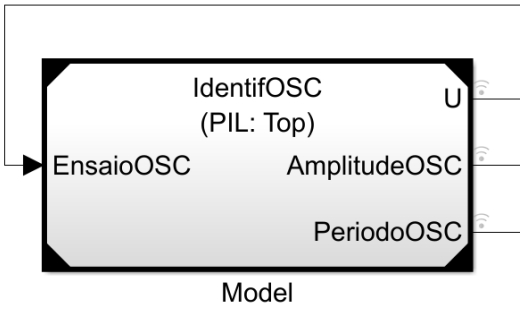


Figura 18: Modelo definido para simulação em PIL

Fonte: Autor

A plataforma utilizada possui recursos para análise da implementação em diversos sistemas embarcados do mercado, desenvolvendo drivers específicos para cada placa. No caso do Arduino Due, já existe uma configuração pré-definida de tradução de variáveis, levando em consideração seu tamanho em bits, seu nome, entre outros. Na Figura 19, é possível verificar as diferentes variáveis utilizadas pelo código e seus tamanhos relativos ao embarcado utilizado.

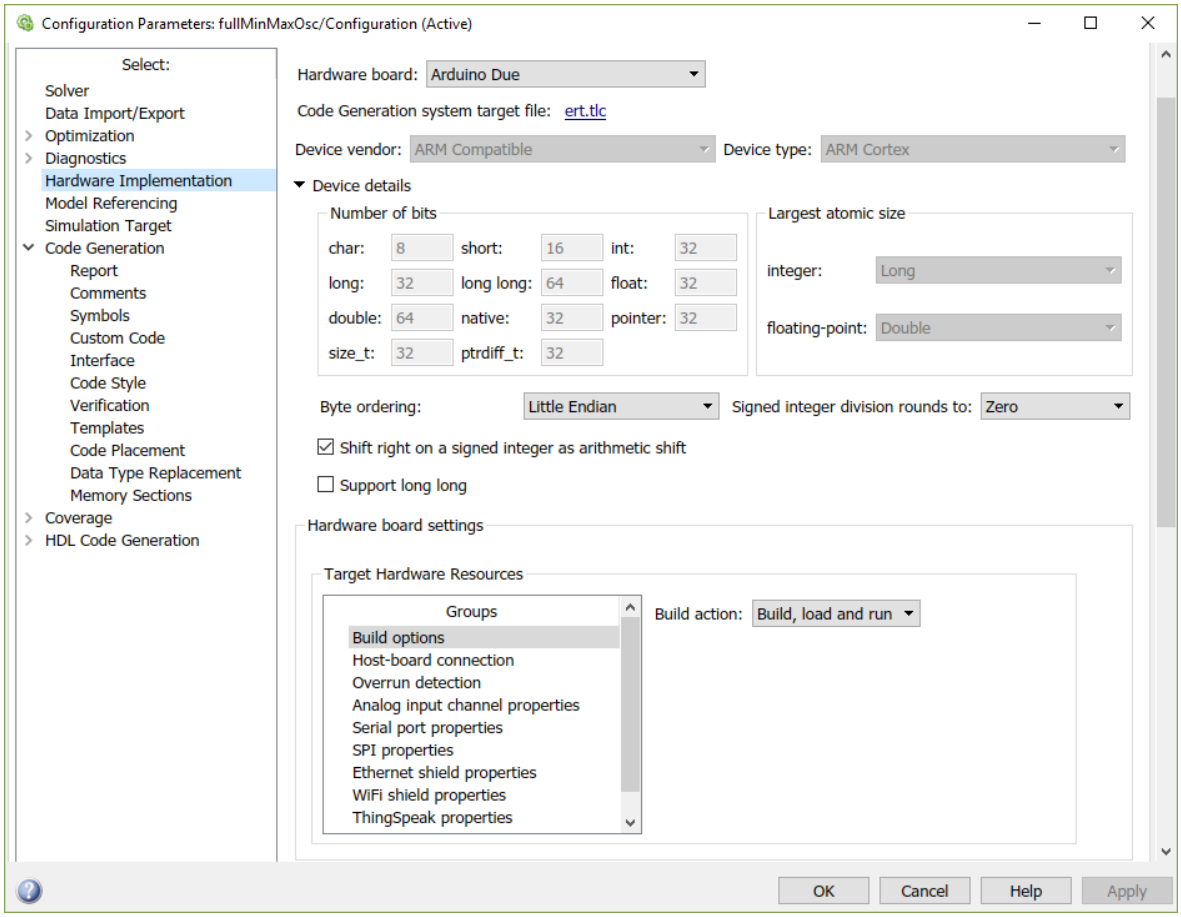


Figura 19: Configurações de implementação no Hardware embarcado.

Fonte: Autor

4.4 HiL

Uma ferramenta muito útil da seção ilustrada na Figura 19 e disponível na metodologia *HiL* é a chamada *Overrun Detection*, a qual habilita um pino de saída digital do microcontrolador para sinal lógico alto quando a pilha de comandos do microcontrolador não consegue voltar ao início do laço no tempo determinado, indicando que não é possível garantir a corretude do processamento dentro do tempo desejado. Assim, pode-se conectar um LED à esta porta e monitorar visualmente se o modelo simulado em HiL está processando o algoritmo de acordo. Esta ferramenta não está disponível na simulação em PiL pois, conforme explicado anteriormente, este modo de simulação se concentra em garantir a corretude do algoritmo e em avaliar o custo computacional do mesmo no embarcado, independentemente do tempo de ciclo estipulado.

O modelo utilizado para este método teve uma adequação de grandeza do sinal tendo em vista os níveis de tensão disponibilizados pelo hardware do Arduino.

O bloco responsável pela aquisição do sinal de tensão na entrada analógica do Arduino foi conectado ao nó de tensão de saída das plantas RC. Esta entrada só permite a leitura de tensões contínuas de 0 a 3,3 Volts. Uma vez realizada a aquisição, um sinal com o tipo de dado inteiro de 10 bits – 0 a 1024 LSB (Least significant bit, que significa a unidade do bit menos significativo) – é disponibilizado para o microcontrolador.

Da mesma forma que a aquisição de sinal, a atuação realizada pelo conversor analógico digital possui um nível de tensão de saída que varia de 0,56 a 2,76 Volts, implicando em uma faixa de 2,2 Volts de excursão de sinal, convertidos a partir de um sinal inteiro de 12 bits, ou seja, de 0 a 4096 LSB, disponibilizado pelo microcontrolador.

Na Figura 20 a seguir, uma ilustração da interligação do Arduino Due com a planta RC de segunda ordem.

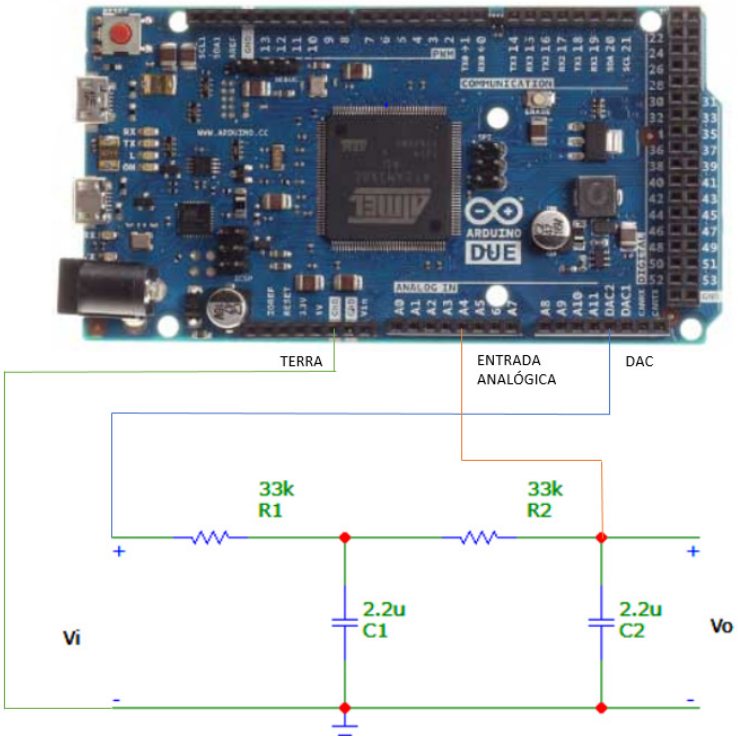


Figura 20: Interligação do Arduino com a planta RC de segunda ordem.

Fonte: Autor

Para habilitar o modo de simulação em HiL no Simulink, deve-se alterar a forma de simulação no modelo interno – que será embarcado – e utilizar os blocos de comunicação com as portas do Arduino disponibilizados pelo Support Package neste novo modelo. Além disso, definir o tipo de simulação para External. Assim, a interface com o computador tem apenas o objetivo de registrar os sinais obtidos e calculados no micro controlador, para serem utilizados no cálculo dos ganhos do controlador posteriormente.

Outra alteração importante no modelo foi que a faixa de operação do mesmo ficou restrita ao número de bits do conversor analógico digital do Arduino, que é 10 bits, correspondendo a um número lógico de 0 a 1024. Dessa forma, uma nova referência constante para o ensaio de oscilação foi definida para 300 LSB, os quais significam pouco menos de 1 Volt na planta física. Além disso, a saída analógica DAC do Arduino possui 0,56 Volts quando em nível lógico zero, portanto foi necessário realizar um offset na entrada do modelo para nivelarmos as faixas de trabalho. Na Figura 21 a seguir, está exemplificado a realização desse ajuste.

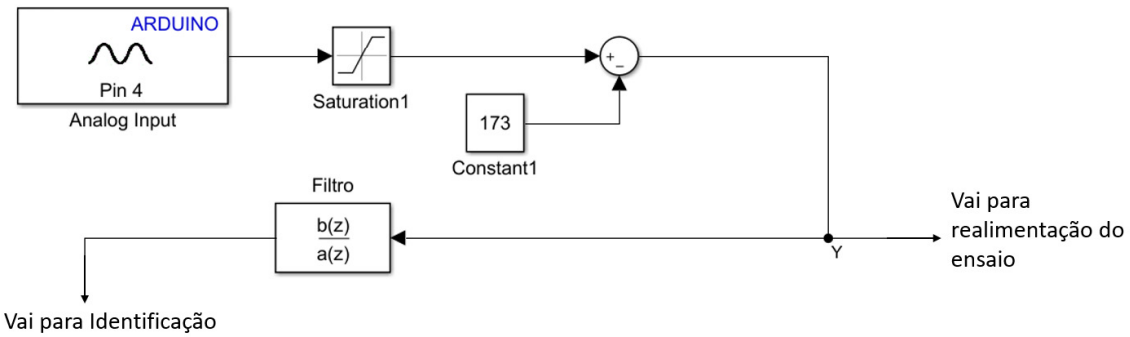


Figura 21: Adequação do sinal da entrada analógica do Arduino.

Fonte: Autor

O valor de 173 LSB foi obtido fazendo-se a divisão dos 1024 bits de resolução da entrada analógica do Arduino pela sua amplitude de aquisição de 3,3 Volts, e então multiplicando pelos 0,56 Volts do DAC para obter o novo zero lógico do modelo, conforme a Equação 14.

$$\frac{1024}{3{,}3} * 0{,}56 = 173\ LSB \tag{14}$$

Semelhante adequação foi realizada na saída do modelo – entrada da planta RC – uma vez que o DAC do Arduino Due possui 12 bits, ou seja, de 0 a 4096 LSBs lógicos, foi necessário escalonar o sinal de saída do ensaio em quatro vezes para adequar-se à grandeza de trabalho do mesmo. Assim, ambos os lados do modelo se referem às mesmas medidas de tensão na planta física. Na Figura 22 a seguir, está ilustrado a saída do modelo, apontando finalmente para o DAC do Arduino.

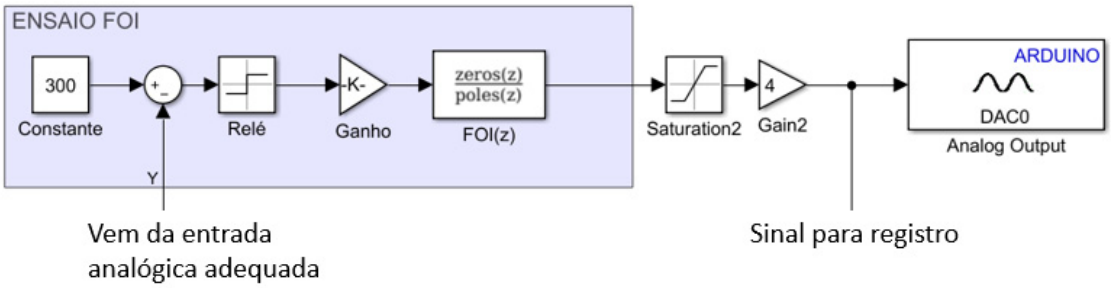


Figura 22: Saída do modelo - Entrada da planta RC.

Fonte: Autor

Os blocos de saturação utilizados são para garantir a faixa de operação dos mesmos e para indicar, nos resultados, caso estejam trabalhando acima ou abaixo dos limites, o que implicaria na inviabilidade de utilizá-los posteriormente.

4.5 Ajuste do Controlador PID

O ajuste do PID é obtido com o cálculo dos parâmetros Kp, Ti e Td, conforme explicitado no Capítulo 2, equações (4), (5) e (7).

Para a validação do modelo de identificação aqui utilizado, foi desenvolvido um script no Matlab que utiliza os valores de amplitude e período obtidos no ensaio, juntamente com o modelo do FOI para calcular os valores dos ganhos do controlador.

O algoritmo utiliza o período da oscilação para encontrar a frequência e assim a resposta em frequência do FOI na mesma, utilizando-se assim de seu ganho e fase para identificar a amplitude da planta e, por fim, definir os ganhos. O *script* completo pode ser encontrado no Apêndice deste trabalho.

Com os ganhos definidos, foi desenvolvido um modelo em Simulink para analisar o resultado do projeto do controlador. Neste modelo é realizado uma resposta ao degrau para os sistemas em malha aberta e em malha fechada. Assim, é possível comparar os desempenhos avaliando a norma L^2 do sinal de erro dos dois sistemas, também incluindo uma saturação para o sinal de controle. Na Figura 23, está ilustrado o modelo completo utilizado para a validação do ajuste do controlador.

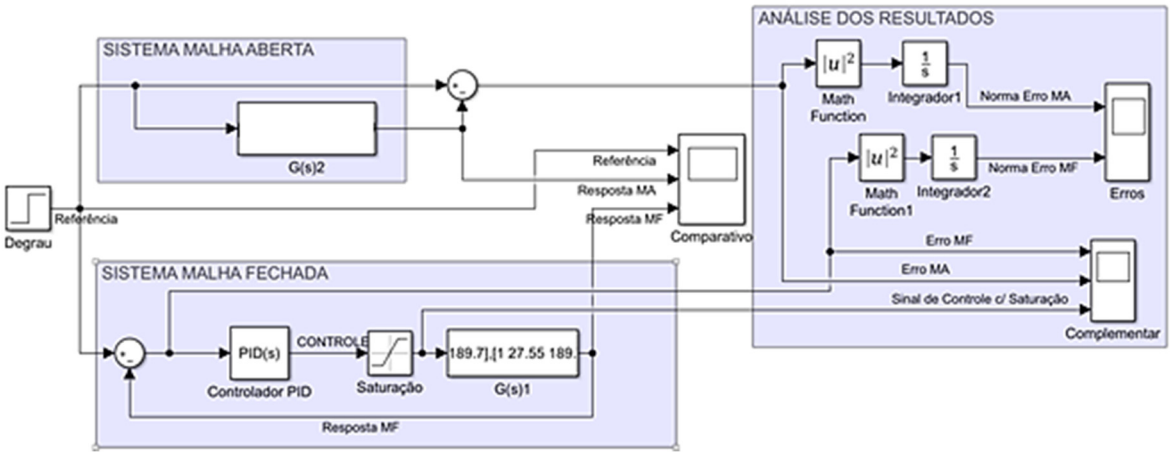


Figura 23: Modelo para validação do controlador projetado.

Fonte: Autor

O principal quesito desenvolvido no modelo para verificação do projeto do controlador é o valor da norma do sinal de erro, que possibilita uma comparação direta de desempenho entre os dois ensaios, tendo em vista que a minimização deste valor é um critério de otimização do projeto do controlador muito comum na literatura.

5 Resultados

Para a planta RC de segunda ordem, o FOI utilizado fornece 60° de contribuição de fase e a mantém aproximadamente constante de 0 a 1 kHz. Portanto, os ensaios realizados utilizaram o tempo de amostragem igual a 100 µs. Desta forma é possível garantir que o sistema seja capaz de identificar qualquer oscilação que aconteça em frequências menores e, assim, abrange-se toda faixa de frequência constante do FOI.

Os resultados obtidos com as simulações servem como base para estimar um tempo de amostragem mínimo para o ciclo de processamento do microcontrolador, além de validar que a geração de código e as manipulações matemáticas dos algoritmos envolvidos no modelo utilizado estão de acordo com o esperado. A validação da corretude do algoritmo pode ser realizada através da ferramenta *Simulation Data Inspector* do Simulink. Na Figura 24 a seguir, está ilustrado uma comparação direta entre as respostas dos ensaios obtidas pelas metodologias de simulação comum e PiL, com tolerância igual a zero na comparação numérica entre os valores, e ainda assim o programa indicou que está dentro da tolerância.

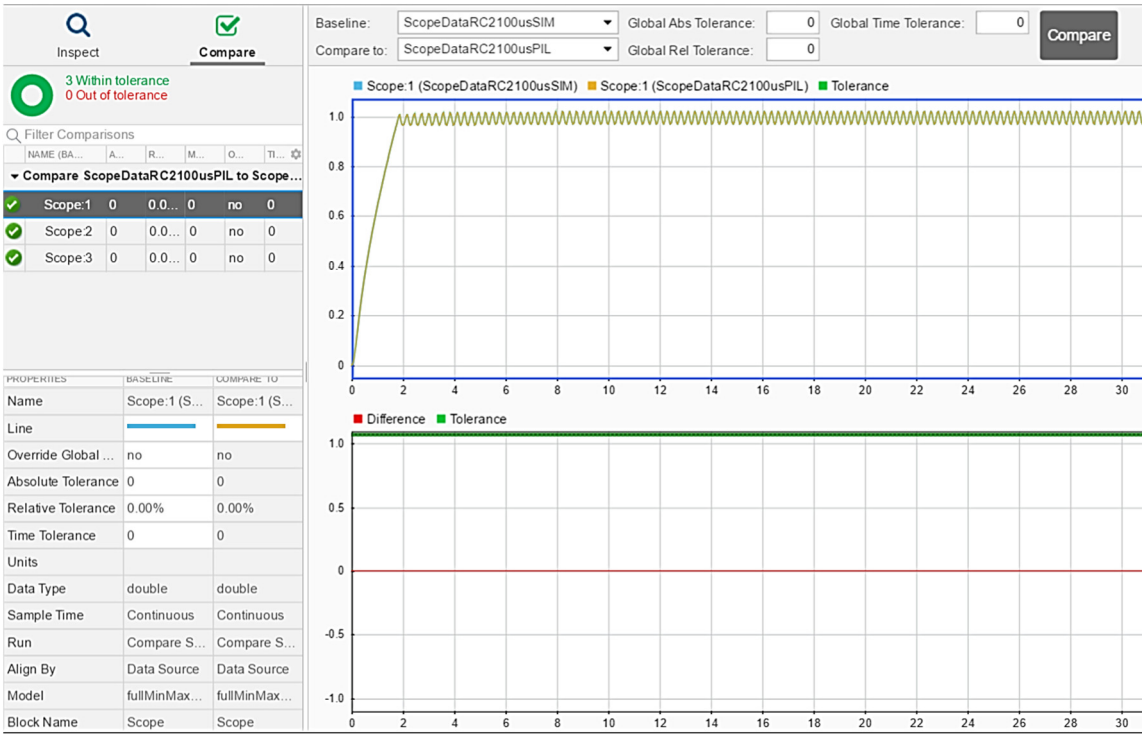


Figura 24: Comparação numérica entre a simulação MiL e a simulação em PIL.

Fonte: Autor

O primeiro gráfico da Figura 24 contempla os valores do ensaio oscilatório em si, ou seja, é a saída da planta RC ao estímulo do relé com o bloco FOI para as duas simulações. O segundo gráfico em vermelho é a diferença entre as curvas do primeiro gráfico, sendo esta nula durante toda a simulação. Os valores do eixo das ordenadas são numéricos do tipo *double* e os valores do eixo das abscissas são em segundos. O mesmo teste foi realizado para a simulação em modo SiL, a qual também se manteve dentro da tolerância zero.

Na Tabela 3, estão os resultados das análises de tempo de execução do modelo para as diferentes metodologias, onde também já foi incluído o resultado da identificação dos parâmetros de amplitude e período para cada teste.

Tabela 3: Tempos de execução para a a planta de segunda ordem.

Fonte: Autor

Método	Tempo de Simulação [s]	Tempo de Execução máxima do bloco	Tempo médio de execução	Amplitude [numérico]	Período [s]
MiL	50	NA	NA	0,0254	0,2701
SiL	50	430 µs	0,32 µs	0,0254	0,2701
PiL	50	246 µs	235 µs	0,0254	0,2701

Como os resultados numéricos foram iguais nas três simulações, na Figura 25 a seguir está ilustrada a resposta completa do ensaio apenas na metodologia PIL.

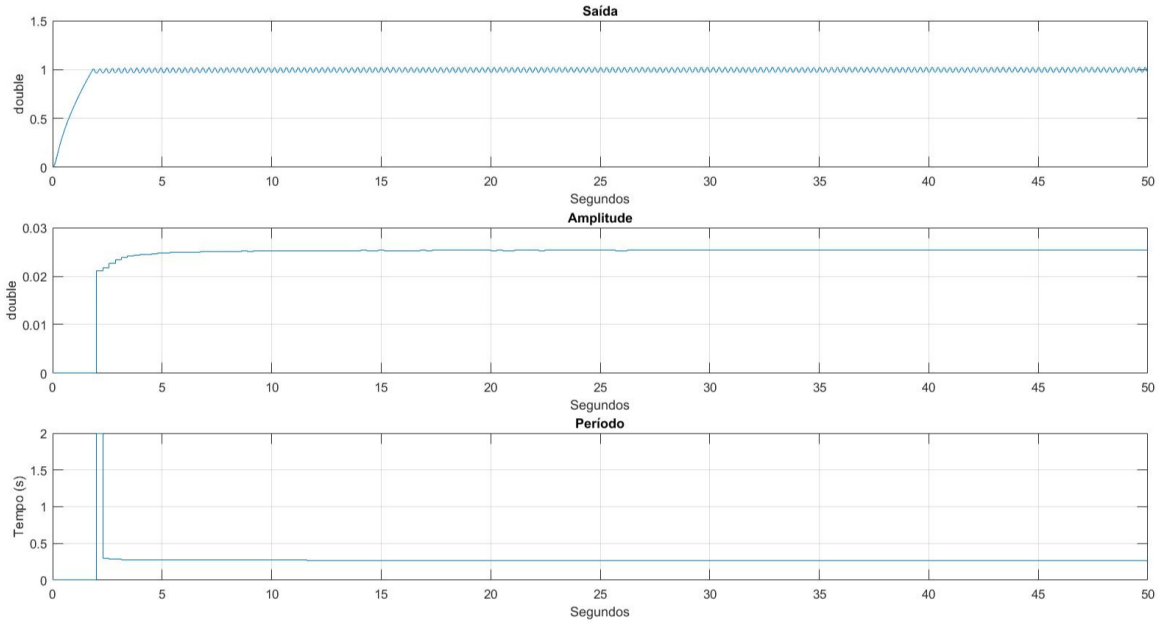


Figura 25: Ensaio completo no método PIL

Fonte: Autor

Observa-se que os valores de amplitude e período tendem a ficar constantes ao longo do ensaio. A variável foi nomeada de *double* pois nestas etapas ainda se trata de uma simulação numérica sem correspondência com a planta física. Na Figura 26, está representado em detalhe apenas o último segundo dos resultados expostos na Figura 25.

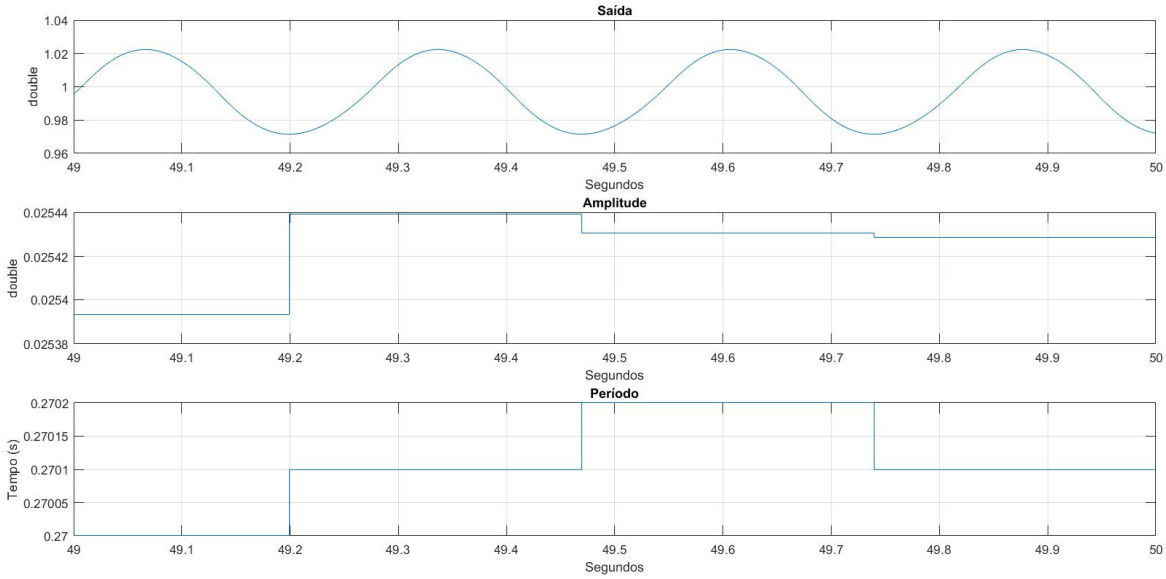


Figura 26: Detalhe do ensaio completo no método PIL.

Fonte: Autor

Com o exposto na Figura 26, é possível verificar que o ensaio foi capaz de identificar o sistema com precisão de milissegundos para o período e precisão de quatro casas decimais para a amplitude, os quais serão postos em teste na próxima etapa da implementação.

Percebe-se que a identificação do sistema ocorreu de forma exata para todas as simulações, com diferença apenas nos tempos de execução, onde a metodologia PiL teve a maior média, na faixa dos 235 microssegundos. Pode-se perceber então que para a próxima etapa de simulação será necessário incrementar o tempo de amostragem de 100 microssegundos utilizado para, ao menos, 500 microssegundos, de forma a garantir que a execução do código e a comunicação com o computador necessária ainda nesta metodologia sejam atendidas a cada ciclo de processamento.

Em uma primeira execução em laço com o Arduino e a planta RC física, denominada de HiL, o algoritmo de identificação não obteve sucesso na obtenção da amplitude e do período da oscilação. Verificando atentamente a curva de resposta da planta amostrada, foi possível notar que havia demasiado ruído presente no sinal, o que ocasionava diversos máximos e mínimos locais dentro de um único período da oscilação a qual se desejava identificar. Para contornar esta situação, foi desenvolvido no Matlab um filtro de *Butteworth* através do comando *butter*. Os parâmetros de projeto do filtro foram a frequência de amostragem do sinal – 2 kHz – e a frequência de corte desejada – 10Hz – uma vez que havia sido possível identificar visualmente o período do sinal oscilatório ruidoso. Assim, foi possível realizar um novo modelo para o sistema com o filtro incluído na entrada do sistema de identificação, fora do laço de realimentação, para limpar o sinal utilizado para aferir a amplitude e o período do mesmo. Este procedimento não interferiu nos resultados de identificação da planta pois o filtro foi projetado para prover ganho unitário nas frequências desejadas e sua contribuição de fase não afeta o algoritmo de identificação desenvolvido. Na Figura 27 a seguir, está ilustrado o resultado da execução do modelo em HiL com a utilização do filtro.

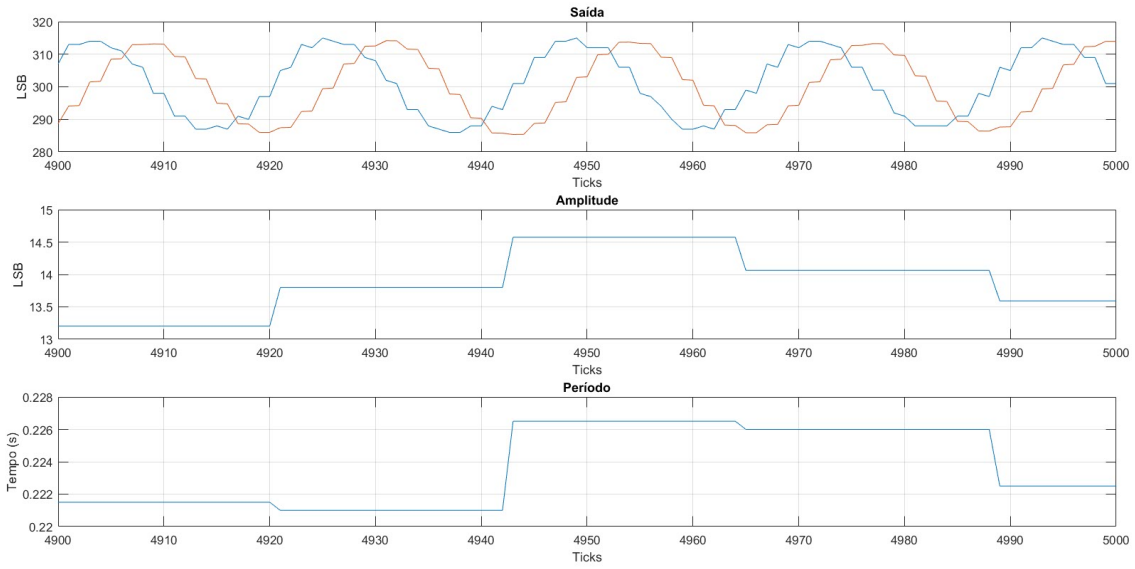


Figura 27: Resultado da execução com HIL.

Fonte: Autor

A curva em azul do primeiro gráfico é o resultado do ensaio sem a passagem pelo filtro, ou seja, é a curva pura amostrada pelo Arduino e utilizada para a realimentação do sistema. A curva em laranja é o sinal filtrado, a partir do qual foi possível aferir valores de amplitude e período corretamente. Uma vez que a faixa de 3,3 Volts da amplitude da porta de entrada analógica corresponde aos 1024 valores disponíveis para 10 bits de resolução do conversor analógico digital, 1 LSB registrado no computador corresponde a aproximados 3,2 milivolts, então, na manipulação algébrica do cálculo da amplitude o valor LSB possui uma correspondência física em Volts, podendo assim ter um valor quebrado, como os 14,5 LSB (que correspondem a ∼46 mV) apresentados no gráfico.

Foi realizado um ensaio com tempo de amostragem inferior a 300 microssegundos para verificar o comportamento da execução. Neste caso o Arduino sinalizou o *overrun* no pino da porta configurada, que conforme explicitado no Capítulo 4, implica que o microcontrolador não consegue cumprir o tempo de ciclo determinado.

O próximo passo foi utilizar os valores obtidos no ensaio para ajustar o PID e verificar se o desempenho das plantas está de acordo com o esperado. Utilizando então o *Script* e o modelo descritos na seção 4.5, obteve-se os seguintes resultados, ilustrados na Figura 28. Na Tabela 4, estão os resultados obtidos para os ganhos do controlador.

Tabela 4: Ganhos do controlador PID para a planta RC de segunda ordem.

Fonte: Autor

$Kp\ [V/V]$	$Ti\ [s]$	$Td\ [s]$
21921	0,2068	0,0064

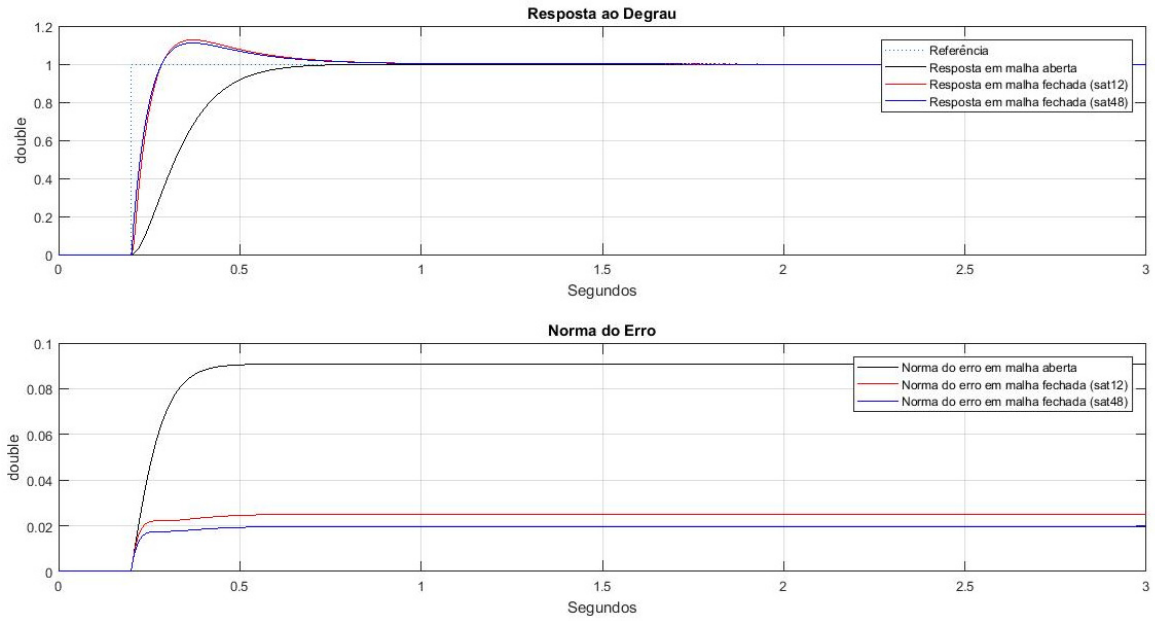


Figura 28: Resultados do ajuste do controlador.

Fonte: Autor

Foi comparado, na Figura 28, a resposta do sistema em malha aberta com malha fechada para duas possibilidades de saturação do sinal de controle, em 12 e em 48 Volts. Caso não houvesse saturação a resposta ficaria ainda superior em termos de desempenho, mas não condiria com qualquer situação prática, pois o sinal de controle teria uma grandeza muito superior às condições de trabalho da planta.

O procedimento completo realizado para a planta RC de segunda ordem foi repetido para a planta RC de primeira ordem, tendo em vista que o FOI de -120° de contribuição de fase é uma função de maior grandeza e, portanto, de maior custo computacional. Foram levantados os resultados dos ensaios nas três primeiras metodologias para a obtenção dos tempos de execução máximos do modelo. Na Tabela 5, estão demonstrados os valores obtidos, utilizando-se também um tempo de amostragem inicial de 100 microssegundos.

Tabela 5: Tempos de execução para a planta de primeira ordem.

Fonte: Autor

Método	Tempo de Simulação [s]	Tempo de Execução máxima do bloco	Tempo médio de execução	Amplitude [numérico]	Período [s]
MiL	50	NA	NA	3,968E-4	0,2716
SiL	50	366 µs	0,47 µs	3,968E-4	0,2716
PiL	50	491 µs	480 µs	3,968E-4	0,2716

Pode-se notar que os tempos de execução subiram significativamente para o modelo executado no microcontrolador se comparados com o modelo do FOI de 60°. Também é possível verificar que a amplitude da oscilação diminui, uma vez o ganho do FOI é menor, conforme ilustrado nos diagramas de bode da seção 4.1.2.

Os valores numéricos das respostas também foram comparados nas três simulações, utilizando o *Simulation Data Inspector*. Na Figura 29, está ilustrado os ensaios oscilatórios, juntamente com um detalhe dos seus últimos segundos.

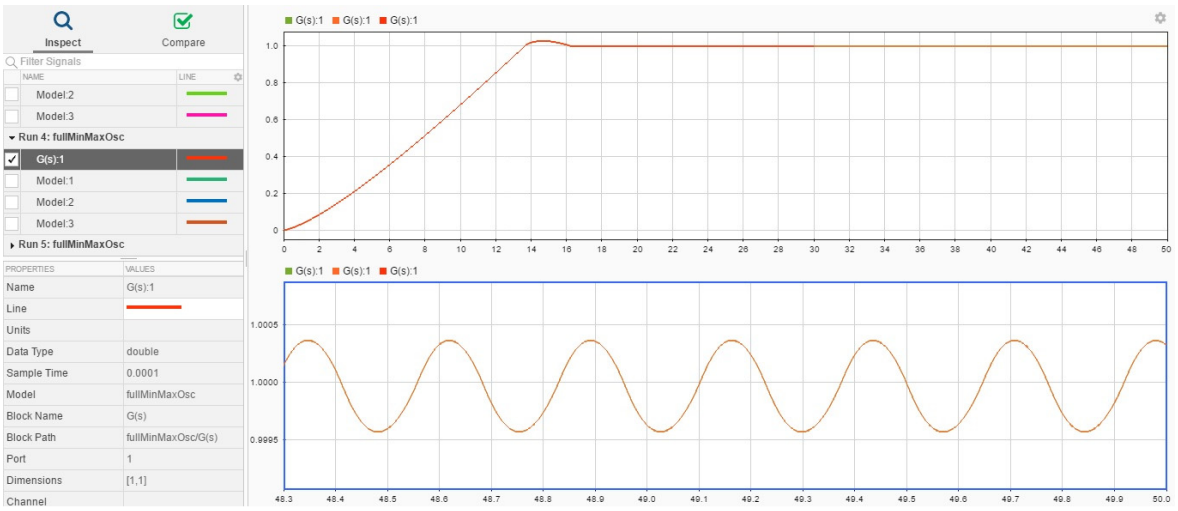


Figura 29: Detalhe do ensaio nas simulações para a planta RC de primeira ordem.

Fonte: Autor

Com a validação da corretude do algoritmo para a geração de código do FOI de -120° e suas características de tempo de execução para o Arduino, foi possível passar para a última etapa da implementação contemplada neste trabalho.

Utilizando uma estimativa semelhante ao ensaio anterior, foi definido um tempo de amostragem duas vezes superior ao tempo de execução médio obtido no ensaio PiL, ou seja, aproximadamente 1 milissegundo.

O Arduino sinalizou *overrun* e os dados obtidos mostraram-se falhados, conforme a Figura 30 a seguir.

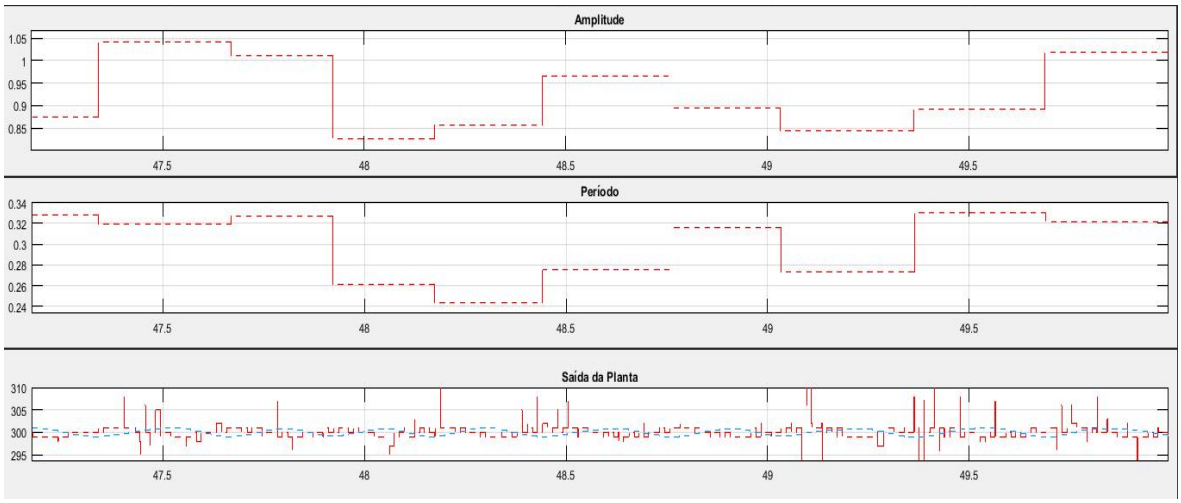


Figura 30: HiL com falha para planta de primeira ordem.

Fonte: Autor

Nota-se que o algoritmo teve uma grande excursão de valores obtidos na identificação. Além disso, a falha nos dados e o *overrun* geram incertezas nos resultados e exigiu-se um novo ensaio com tempo de ciclo de 5 milissegundos.

O novo ensaio realizado não foi bem-sucedido em termos de identificação do sistema pois o ganho do FOI ficou muito baixo para a resolução do Arduino, de forma que as oscilações na saída da planta estavam em torno de 5 mV. Portanto, foi aumentado o ganho do relé em mais cem vezes, visando aumentar a faixa de operação do ensaio para a capacidade de amostragem do Arduino. Na Figura 31 a seguir, está ilustrado os últimos segundos do primeiro ensaio, de amplitude insuficiente.

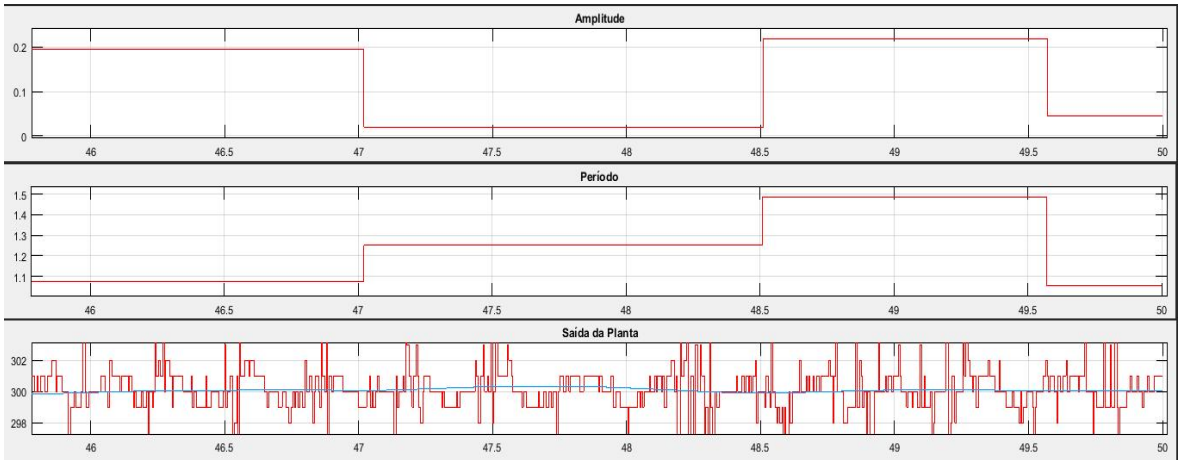


Figura 31: Ensaio HiL com falha na identificação por baixo ganho.

Fonte: Autor

Pode-se observar que a oscilação possui menos de 3 LSB de amplitude, tornando-se extremamente sensível ao ruído. Desta forma, ao aumentar-se o ganho do relé, foi possível obter valores coerentes para o ensaio, conforme ilustrado na Figura 32.

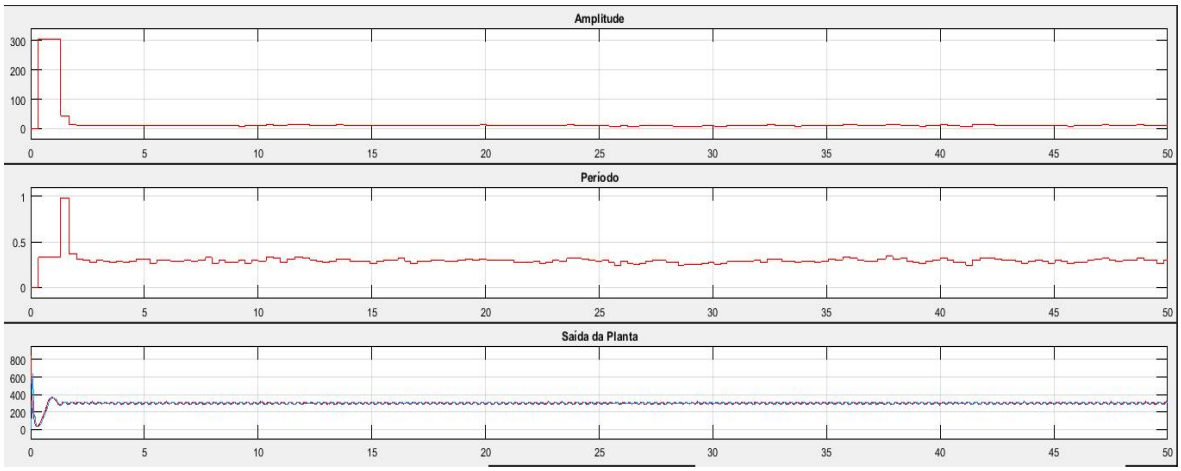


Figura 32: Ensaio com amplitude aceitável para medição pelo Arduino, utilizando a planta de primeira ordem.

Fonte: Autor

Na Figura 33, o detalhe dos resultados obtidos no ensaio.

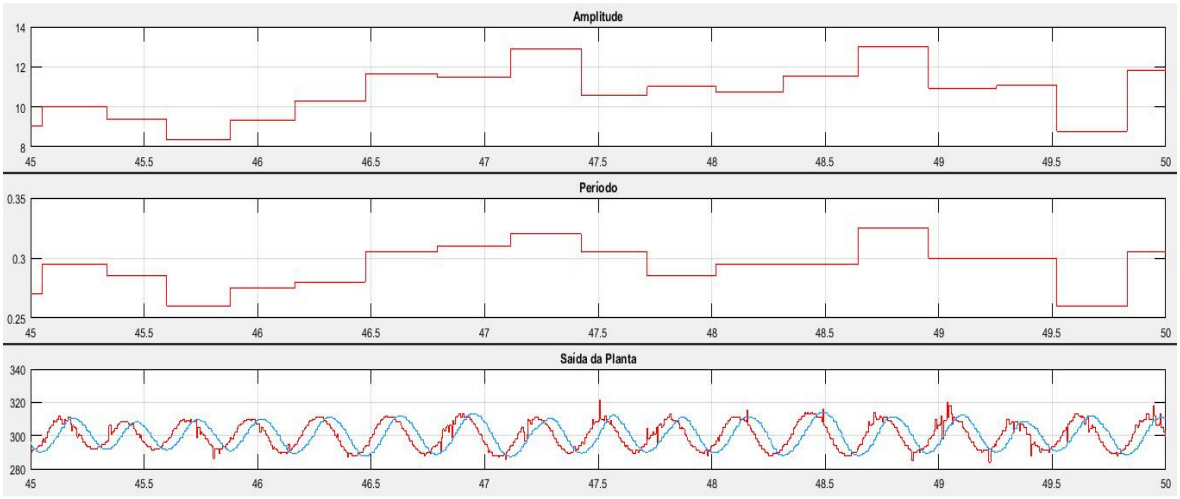


Figura 33: Detalhe do ensaio com resultados aceitáveis.

Fonte: Autor

Pode-se observar que a amplitude do sistema agora oscila em torno dos 12 LSB, que correspondem a aproximados 36 milivolts, e o período está em torno de 0,3 segundos, o que é condizente com o obtido nas etapas anteriores.

Foram obtidos então os ganhos para controlador PID com os dados do ensaio, fazendo as adaptações necessárias ao *Script* e ao modelo, e os valores obtidos foram os seguintes.

Tabela 6: Ganhos para o controlador PID para a planta de RC de primeira ordem.

Fonte: Autor

$Kp\ [V/V]$	$Ti\ [s]$	$Td\ [s]$
789370	0,2754	0,0086

Interessante ressaltar que o ganho proporcional ficou bastante alto, devido ao fato de que o ganho do relé também ter sido muito alto, em torno de centenas de milhares, para compensar o baixíssimo ganho da função de transferência do FOI e a resolução máxima do AD do Arduino para o sinal de controle disponível.

Na Figura 34, estão os resultados obtidos em simulação com o ajuste do PID, com sinais de controle saturando em 12 e 48 Volts.

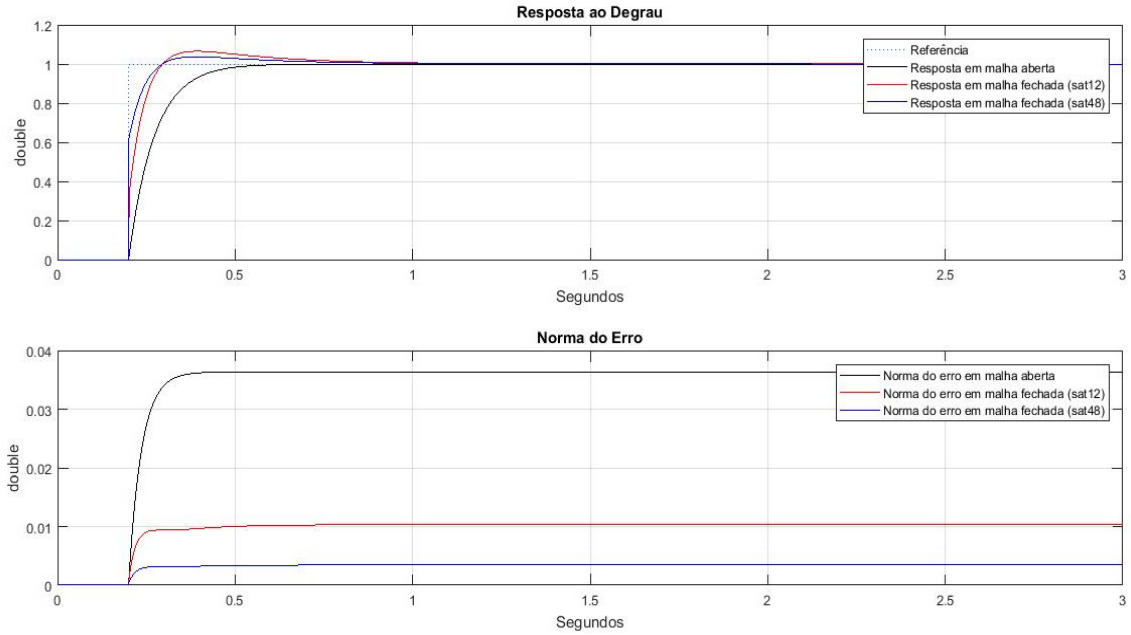


Figura 34: Resultados do ajuste do controlador.

Fonte: Autor

Os resultados do ajuste do PID com saturação no sinal de controle levaram a uma melhoria de desempenho da planta de cinco vezes a norma do sinal de erro para o ensaio em malha aberta, indicando o potencial do método para o projeto de controladores.

6 Conclusões e Trabalhos Futuros

A verificação e validação são o cerne de qualquer processo robusto o qual se está desenvolvendo baseando-se em modelos. Elas são utilizadas para avaliar a precisão dos algoritmos com geração de código e interação de software, acelerando a identificação de erros e antecipando melhorias no processo. Nesse sentido, foi satisfatório utilizar-se destas metodologias para validar a implementação do sistema de identificação de parâmetros em um Arduino.

Foi possível verificar os modelos em diferentes níveis de aplicação, desde seu modelo lógico ao seu modelo físico, identificando as principais dificuldades de cada etapa e as peculiaridades de cada implementação. Por exemplo, o modelo do FOI para a planta de primeira ordem exigiu um ganho do relé altíssimo ocasionado pela baixa limitação na resolução do conversor AD e respectiva faixa de trabalho do Arduino. Em trabalhos futuros, é possível acrescentar placas com amplificadores operacionais para garantir uma maior excursão do sinal de controle e ajuste do OFFSET do DAC, para poder trabalhar em uma faixa de tensão maior na planta e com sinais negativos, aprimorando assim a usabilidade do AD do Arduino e diminuindo sua sensibilidade ao ruído.

Com a possibilidade de identificar os tempos de execução do modelo no sistema embarcado, é possível, com uma margem de segurança, garantir que o sistema em produção terá o desempenho esperado tanto em precisão numérica quanto em corretude do algoritmo. Uma melhoria para o ensaio deste trabalho seria a adição do cálculo dos ganhos do PID e sua implementação em malha fechada, automaticamente, posterior ao ensaio de identificação no modelo embarcado, de forma a produzir um controlador automático. Com isto, seria dispensável o auxílio do computador na fase final de prototipagem do sistema. Logicamente, diversas melhorias no modelo deverão ser implementadas para realizar esta implementação, como por exemplo um teste para verificar se a contribuição de fase do FOI na frequência da oscilação forçada está dentro da sua banda de fase constante esperada, entre outros.

As metodologias vistas neste trabalho podem ser expandidas para modelos mais complexos, inclusive sistemas críticos onde a implementação física é cara e é preciso estar com modelo mais robusto possível na hora de investir nos mecanismos e peças.

7 Referências

Levine, William S. (2010) **The Control Handbook, Second Edition: Control System Applications,** CRC Press Book

Bazanella, Alexandre S., Pereira, Luís Fernando A., Parraga, Adriane (2016) **A New Method for PID Tuning Including Plants Without Ultimate Frequency** , IEEE Transactions on Control Systems Technology

Charef, A., Sun, H. H., Tsao, Y. Y., Onaral B, (1992) **Fractal System as Represented by Singularity Function**, IEEE Transactions on Automatic Control

Arduino (2017), **What is Arduino?** Disponível em: < https://www.arduino.cc/en/Guide/Introduction> - Acesso em: 01 de julho de 2017.

Arduino (2017), **Referências da placa Arduino Due** Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardDue> - Acesso em 10 de junho 2017.

W. A. Wolovich (1993) **Automatic Control Systems—Basic Analysis and Design**, New York, NY, USA: Oxford Univ. Press

Bringmann, E., Krämer, A. (2008) **Model Based Testing of Automotive Systems**, PikeTec GmbH, Germany

Ogata, K. (2003), **Engenharia de Controle Moderno** - **4ªed.**, Prentice-Hall.

Matlab & Simulink (2015) **Embedded Coder: Getting Started Guide**. Disponível em: <http://files.matlabsite.com/docs/books/matlab-docs/embedded_coder_ecoder_gs_r2015a.pdf> - Acesso em 01 de julho de 2017.

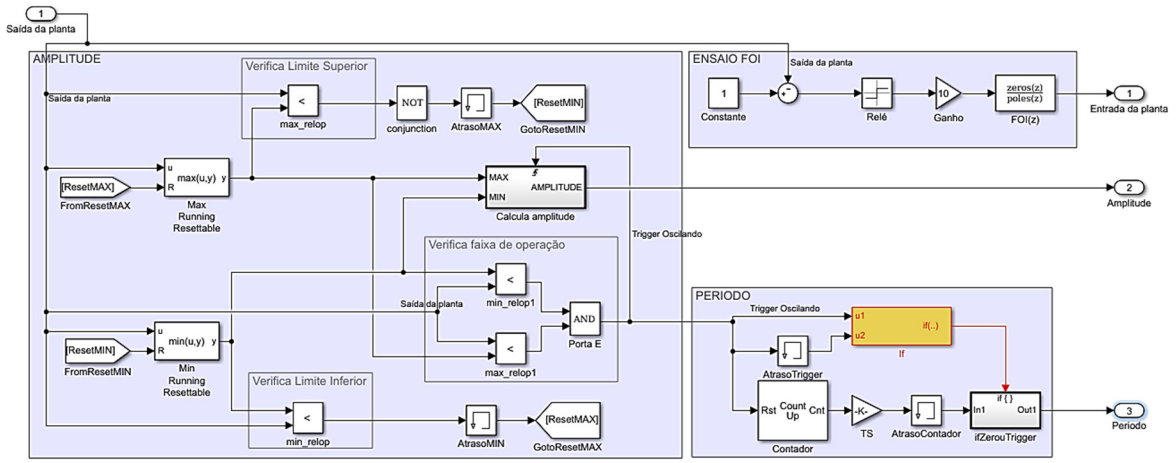
Borges, Raquel Machado (2005) **Desenvolvimento e aplicação de um sistema de diagnóstico fuzzy baseado em modelos para reatores UASB tratando esgoto sanitário.** 2005. 141 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, Es.

Carvajal, Ricardo Enrique Gutiérrez (2011) **Sobre Técnicas para Manutenção e Diagnóstico Inteligente de Dispositivos Mecatrônicos: Estudo de Caso utilizando Cálculo de Ordem Fracionária**. Tese (Mestrado) - Unicamp, Campinas.

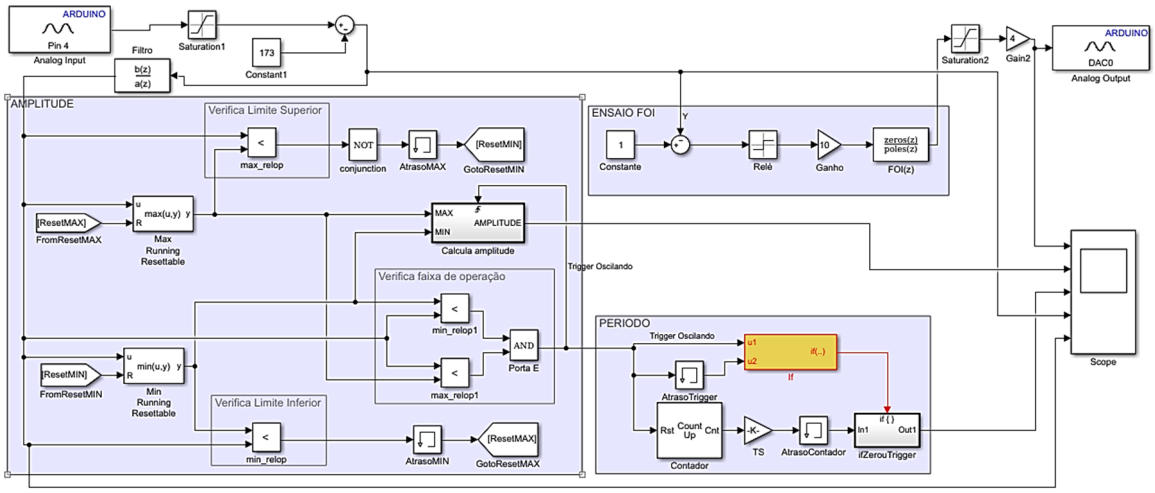
Nilsson, James W, Susan A. Riedel (2008) **Circuitos Elétricos**. Prentice Hall/Pearson, 8ª. Ed.

8 Apêndice

1. Modelo completo do sistema em Simulink que realiza o método da oscilação forçada estendido incluindo a instrumentação necessária para a identificação dos parâmetros desejados.



2. Modelo completo do sistema em Simulink para execução na metodologia HIL, onde as portas I/O podem interagir com os sensores e atuadores reais do sistema, uma das últimas etapas do desenvolvimento baseado em modelos.



3. *Script* em Matlab utilizado para obter o ajuste do PID, com base no FOI utilizado e nas variáveis de amplitude e período identificadas no ensaio.

```
%ENTRADAS
%FS é o FOI do ENSAIO
FS = Fs_Disc;
%SCOPE DATA SÃO OS DADOS DO ENSAIO
%DEFINE QUAL SCOPE BUSCAR OS DADOS
ScopeData = ScopeData500micro_filtro;

%DAQUI PRA FRENTE É AUTOMÁTICO
dataEnsaio = squeeze(ScopeData.signals(5).values);
dataAmp = squeeze(ScopeData.signals(2).values);
dataPeriod = squeeze(ScopeData.signals(3).values);
Amp = dataAmp(length(dataAmp));
Amp = 3.22*Amp/1000; % para converter de LSB lógico para Volts
Period = dataPeriod(length(dataPeriod));
omega = 6.28/Period;%2 pi * periodo, frequencia em rad/s
FSOmega = freqresp(FS,omega);
abs(FSOmega); % amplitude da resposta em frequência
ModuloF = abs(freqresp(FS,omega));
ModuloG = (3.14159*Amp)/(40*ModuloF);
FaseF = phase(FSOmega)*180/3.14159; %convertendo de rad pra graus
Kp = cos(degtorad(10))*cos(degtorad(10))/ModuloG;
Td = tan(degtorad(10))/omega;
Ti = 1/(omega*tan(degtorad(10)));
```

4. *Script* em MatLab que constrói a função de transferência do FOI de -60° de contribuição de fase

```
%% Projeto FOI
% valores de entrada
wL = 1; %freq mínima
wH = 1000; % freq máxima
m  = 60/90; %ângulo desejado
e  = 1e-2; %tolerância -3DB do início da singularidade
y  = 1.2; % tolerância entre as linhas zig-zag e a linha de -20DB

% cálculo dos parâmetros
%corner frequency
wc = wL*sqrt(10^(e/(10*m))-1)
%razao entre um zero e o polo anterior
 a = 10^(y/(10-10*m));
% razao entre um polo e o zero anterior
 b = 10^(y/(10*m));
p0 = wc*10^(y/(20*m));
z0 = a*p0;
%frequencia máxima
wmax = 100*wH;
%número de pares zero polo utilizados (singularidades)
N = round(log10(wmax/p0)/log10(a*b))+1;
F =1;
s = zpk('s');
%produtório que constrói o FOI
for k=1:N
    F = F*(1+s/(z0*(a*b)^(k-1)))/(1+s/(p0*(a*b)^(k-1)));
end
Fs = F;
```