

Samuel Oliveira Milagre

Márcio Ribeiro de Oliveira Filho

Álan Crístoffer e Sousa

# MANUAL DA PLATAFORMA PARA CONTROLE DE PROCESSOS

LACHESIS 1.3.0

MOIRAI 1.3.0



Divinópolis

2019



## Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Instalação	2
1.1.1	Python	2
1.1.2	Moirai	2
1.1.3	Lachesis	3
<b>2</b>	<b>Conexão</b>	<b>5</b>
<b>3</b>	<b>Configuração de Hardware</b>	<b>9</b>
<b>4</b>	<b>Resposta do Sistema</b>	<b>13</b>
4.1	Degrau	14
4.2	Impulso	16
4.3	Pulso largo	17
4.4	Escada	18
4.5	Forma livre	19
<b>5</b>	<b>Controle do Sistema</b>	<b>21</b>
<b>6</b>	<b>Gráficos</b>	<b>27</b>
<b>7</b>	<b>Simulação do Modelo</b>	<b>31</b>
<b>8</b>	<b>Controle Livre</b>	<b>33</b>
<b>9</b>	<b>PID</b>	<b>35</b>
	<b>Bibliografia</b>	<b>37</b>



# Capítulo 1

## Introdução

A plataforma de controle foi desenvolvida para auxiliar no ensino de controle e análise de sistemas. Ela foi desenvolvida de forma que o usuário possa utilizar a linguagem *Python* para escrever seus controladores.

Os dois principais objetivos da plataforma é rodar em um *hardware* com baixo poder processamento, como o *Raspberry Pi*, e exibir gráficos em tempo real. Para isso ela foi dividida em dois aplicativos, visando distribuir o processamento. O primeiro, denominado **moirai**, deve ser instalado na máquina que faz interface com o *hardware* e funciona como um servidor. O segundo, **Lachesis**, funciona como cliente e é a interface gráfica que será utilizada pelo usuário para gerenciar os teste e realizar configurações.

A divisão em dois aplicativos faz com que o processamento de dados e exibição de gráficos ocorra no computador do usuário, enquanto o processamento do controlador ocorre na máquina próxima ao hardware. Os aplicativos se comunicam por rede, logo o usuário pode utilizar o **Lachesis** em sua máquina pessoal enquanto o **moirai** roda em um *Raspberry Pi*, por exemplo.

Em um exemplo de aplicação podemos ter o usuário utilizando seu computador pessoal, que se comunica com um *Raspberry Pi* que controla um processo através de um CLP — computador lógico programável. Também é possível ter ambos aplicativos na mesma máquina controlando uma planta através de *Arduinos*.

## 1.1 Instalação

### 1.1.1 Python

O **moirai** é escrito em *Python 3* e requer que o mesmo esteja instalado na máquina. Há várias maneiras de se obter o interpretador *Python*. Ele pode ser baixado diretamente do site <https://www.python.org>, instalado pelo gerenciador de pacotes do seu sistema operacional (*apt*, *dnf*, *brew*, etc), instalado por gerenciadores de versão (*pyenv*) ou por uma suíte (*Anaconda*).

Este último é necessário na plataforma *Windows* já que não há binários para esta plataforma no repositório *PyPi* para várias bibliotecas utilizadas para controle e a compilação é extremamente difícil devido a falta de suporte das bibliotecas usadas pela suíte *SciPy*.

Seguem as maneiras mais fáceis de se instalar o *Python* no *Windows*, *Ubuntu* e *macOS*:

- **Windows** Baixe e instale o *bundle Anaconda*.
- **Ubuntu** `sudo apt install python3-scipy python3-matplotlib`
- **macOS** Instale o *Homebrew*: <https://brew.sh>  
`brew install python3`  
`pip3 install numpy matplotlib`

### 1.1.2 Moirai

Após instalado o *Python*, deve-se instalar o **moirai** utilizando o gerenciador de pacotes *pip*. Para isso basta executar em um terminal:

```
pip3 install moirai
```

O aplicativo depende de um banco de dados (*MongoDB* ou *MySQL*) e, caso deseje-se utilizar o CLP da *Siemens*, da biblioteca *Snap7*. A instalação desses foi inserida como opção no aplicativo, e pode-se executá-la através do comando:

```
moirai --install --sudo
```

Os aplicativos serão baixados e instalados conforme necessário. Caso a instalação falhe, uma mensagem de erro será exibida mostrando qual *software* não pôde ser instalado e pedindo que a instalação manual seja realizada.

Uma vez que as dependências tenham sido instaladas, deve-se configurar uma senha para o aplicativo, que será necessária para realizar a conexão pelo **Lachesis**. Para isso execute

```
moirai --set-password=1234
```

substituindo 1234 pela senha desejada.

Para executar o aplicativo, simplesmente digite `moirai` em um terminal. A mensagem “Connected webapi to Hardware” indica que o aplicativo está pronto para processar conexões. Para sair do aplicativo, clique na janela do terminal e precione CTRL+C.

### 1.1.3 Lachesis

O **Lachesis** é distribuído na forma de instalador e aplicativo. Para *Windows* e *macOS*, vá para a página <https://github.com/acristoffers/Lachesis/releases> e baixe a versão mais atual (*.exe* para *Windows*, *.dmg* para *macOS*). Para Linux, execute o comando `sudo snap install lachesis` em um terminal.



# Capítulo 2

## Conexão

Os softwares **Lachesis** e **moirai** se comunicam através de um protocolo de rede, mais especificamente o *HTTP (HyperText Transfer Protocol)*. Isso possibilita tê-los instalados em diferentes computares conectados em rede. A configuração para que ambos se comuniquem é simples, sendo necessário informar apenas o *IP* e a porta onde o **moirai** está escutando, esperando por conexões.

Para determinar o *IP* de uma máquina pode-se usar o comando **ifconfig** no *Linux* ou **ipconfig** no *Windows*. Caso o computador esteja ligado a mais de uma rede, como *WiFi* e *Ethernet* (rede cabeada), deve-se prestar atenção a qual endereço pertence a qual rede física. Os endereços especiais *localhost* e *127.0.0.1* apontam para a própria máquina, assim, quando usado no **Lachesis**, ele tentará encontrar o **moirai** na mesma máquina em que ele se encontra instalado.

É necessário especificar a porta de conexão, mas essa será sempre *5000*, a não ser que algum *proxy* ou redirecionamento de porta seja empregado. Para especificar a porta basta adicionar *:5000* ao final do *IP*. Por exemplo: *localhost:5000* e *192.168.11.1:5000*.

Pode-se ver na Figura 1 o módulo de conexão do **Lachesis**. Para se conectar a um servidor **moirai** basta digitar o endereço de *IP* com a porta e a senha que foi configurada.

Por padrão o endereço *localhost:5000* já vem preenchido. Caso outras conexões já tenham sido realizadas os endereços serão listados em *Conexões anteriores*. Basta clicar em um endereço para que ele seja preenchido no campo correto.

Ao clicar em Conectar algum erro pode ocorrer. Falha de conexão normalmente é causado por endereço incorreto ou erro na rede. Verifique que o endereço está certo e que seu computador está de fato na mesma rede que o computador contendo o **moirai**. Certifique-

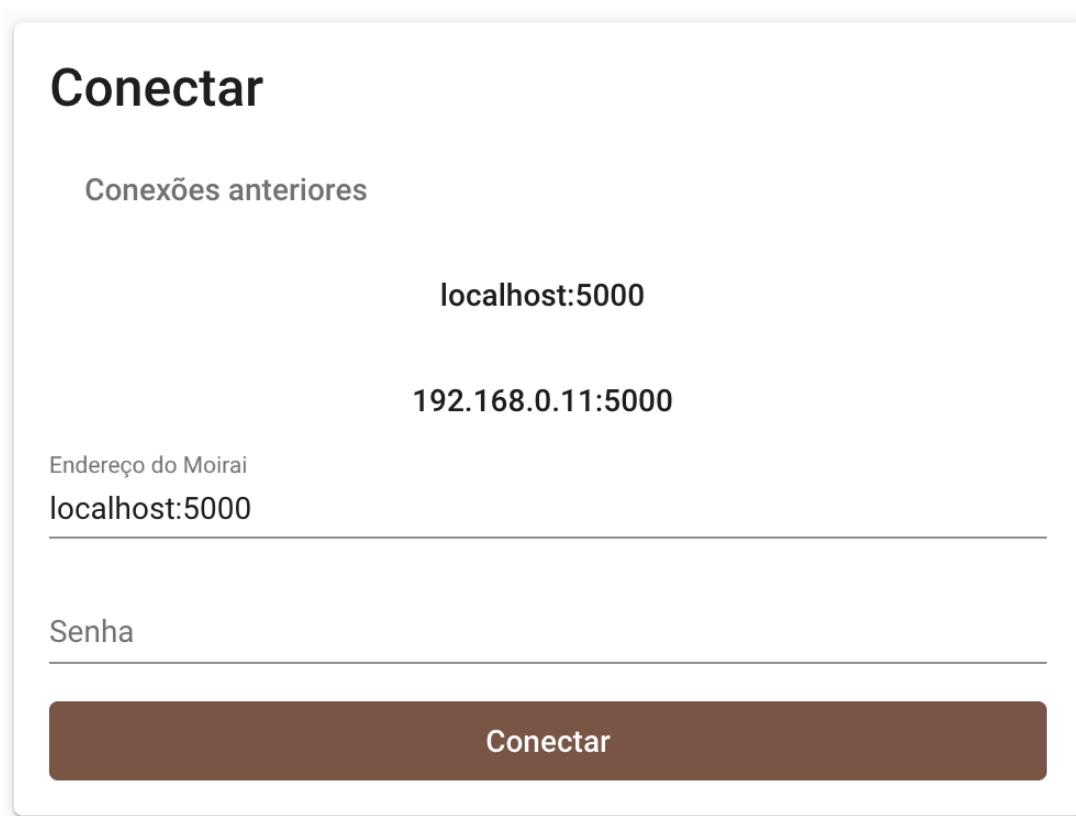


Figura 1 – Módulo de conexão

se também que o **moirai** está em execução. Caso o erro se refira a versão do **moirai**, esse deve ser atualizado. Para isso basta executar o comando `pip3 install --upgrade moirai` em um terminal na máquina onde o **moirai** se encontra instalado. Não se esqueça de reiniciar o aplicativo após a atualização.

Uma vez autenticado o módulo de conexão se altera, bem como a barra lateral, como pode ser visto na Figura 2. A barra lateral exibe os módulos existentes na plataforma, enquanto o módulo de conexão mostra as opções de desconectar, alterar senha e salvar/restaurar.

Ao clicar em desconectar o aplicativo volta ao estado inicial, não exibindo os módulos na lateral e pedindo pelo *IP* e senha.

Para alterar a senha do **moirai** basta digitar a nova senha duas vezes, nas duas caixas próprias, e clicar em Aplicar.

As opções Salvar e Restaurar fazem *backup* e restauração de todo o banco de dados do **moirai**. Deve-se atentar ao fato que a restauração apaga o banco atual antes de importar os novos dados, resultando em perda de dados que não tenham sido salvos em *backup*.

Como o *backup* salva o banco de dados inteiro, os testes realizados também são salvos.

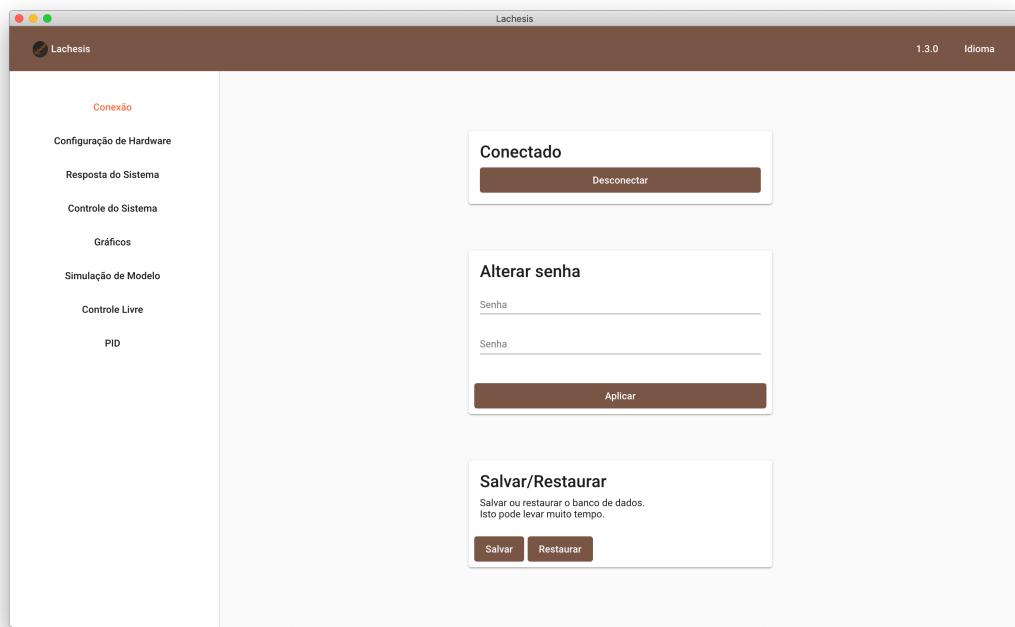


Figura 2 – Módulo de conexão após autenticação

Essa opção pode então ser utilizada para remover dados do banco, acelerando o aplicativo, sem perder dados. Também pode ser utilizada para migrar o **moirai** de uma máquina pra outra.



# Capítulo 3

## Configuração de Hardware

A plataforma foi desenvolvida para se comunicar com qualquer *hardware*. A comunicação se dá através de *drivers*, escritos em *Python*. Isso é possível através do projeto AHIO.

Alguns *drivers* já estão disponíveis por padrão, mas pode-se escrever novos *drivers* personalizados. Para isso copie o arquivo do *driver* do *Arduino* (*Github*) e modifique para acessar seu *hardware*. Caso necessário, leia os comentários em *abstract\_driver.py*

Salve o arquivo em algum diretório em seu computador e inicie o **moirai** utilizando o seguinte comando:

```
AHIO_PATH="/caminho/para/diretorio" moirai
```

A variável de ambiente **AHIO\_PATH** funciona como a variável PATH dos sistemas *UNIX*.

Os *drivers* padrão são: *snap7*, *Arduino*, *GenericTCPIO*, *Raspberry*, *SISO Model* e *Dummy*. Nesse módulo, representado na Figura 3, pode-se configurar o *driver* desejado. Primeiramente deve-se selecionar o *driver* na lista. Duas seções serão abertas ao selecionar um *driver*: *Configuração* e *Portas*.

Na seção *Configuração* deve-se inserir a configuração específica do *driver*. As opções são os parâmetros do método *setup* quando se utiliza a biblioteca *AHIO* diretamente. Normalmente serão parâmetros de localização do *hardware*, como porta do *Arduino* ou endereço do CLP.

A seção *Portas* (Figura 5) permite a configuração das entradas e saídas do sistema. Ao clicar em *Adicionar* uma nova entrada em branco é inserida. Ela contém as seguintes colunas: *Nome da porta*, *Alias da porta*, *Tipo de porta*, *Valor desligado*. O botão  remove a entrada.

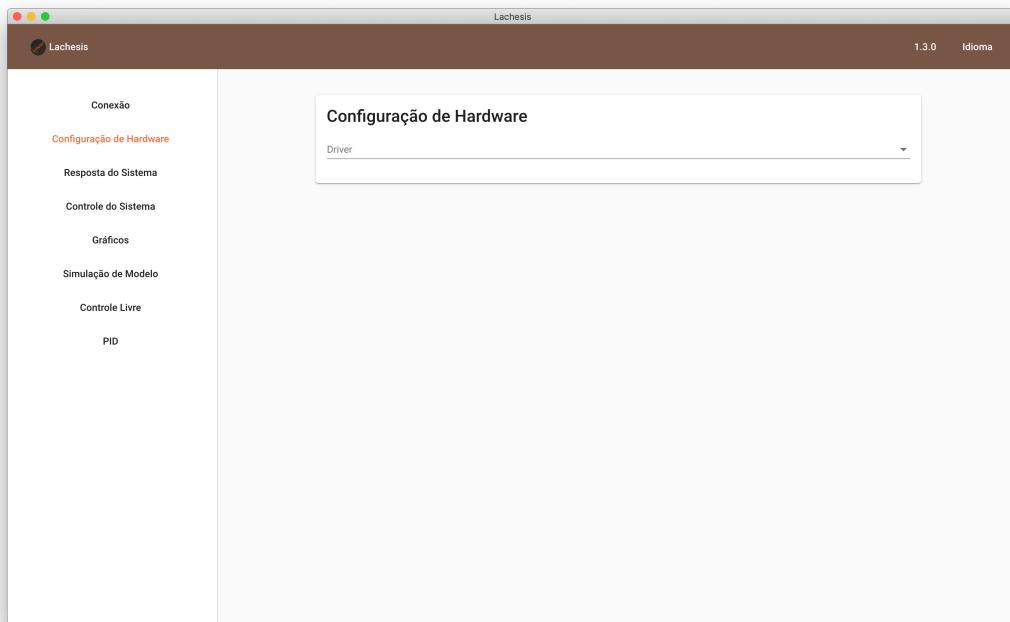


Figura 3 – Módulo Hardware sem *driver* selecionado

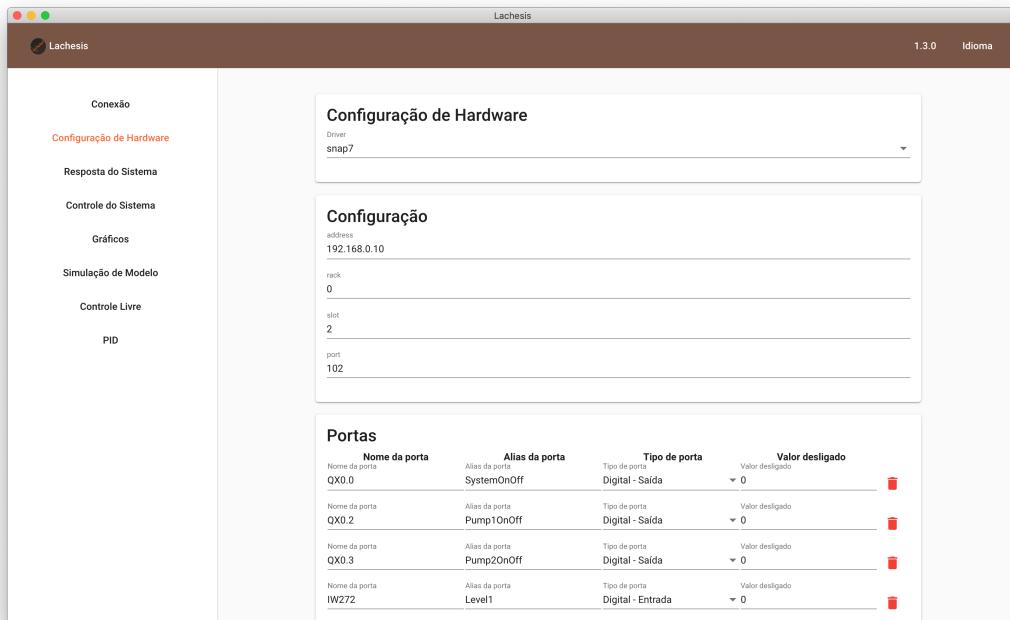


Figura 4 – Módulo Hardware com *driver* selecionado

O *Nome da porta* pode ser uma caixa de seleção (*dropbox*) ou uma caixa de texto, a depender do *driver*. No caso do *Arduino*, por exemplo, será uma *dropbox* com todas as portas do dispositivo. No caso de um CLP (*driver Snap7*) será uma caixa de texto onde o usuário pode digitar o nome da *TAG*, como “QX0.0” ou “IW292”. Esse campo referencia

Portas			
Nome da porta	Alias da porta	Tipo de porta	Valor desligado
Nome da porta	▼ Alias da porta	Tipo de porta	▼ Valor desligado
<b>Adicionar</b>			

Figura 5 – Seção Portas

a porta física no hardware.

O *Alias da porta* é o nome dado para essa porta física no aplicativo. Esse nome serve como uma variável e será exibido em formulários e acessado por código. Escolha um nome que identifique bem o que está ligado à porta. Por exemplo, num sistema de tanques temos o alias “Pump1” associado à porta “QW272”, pois essa porta irá comandar a bomba número 1 do sistema.

O *Tipo de porta* varia de acordo com a escolha da porta física. Pode ser uma combinação de digital ou analógica, entrada ou saída. Também pode ser do tipo saída *PWM*. O *driver* pode limitar as opções disponíveis de acordo com a porta selecionada.

*Valor desligado* apenas faz sentido para atuadores. É o valor que será escrito na porta em determinadas situações. Normalmente é utilizado para desligar o sistema em caso de falhas ou após alguns testes. Se a interface oferece a opção de definir valores após o teste, essa opção não será usada. Caso contrário, sim.

Após inserir uma porta, duas novas seções se tornam disponíveis: *Calibração* e *Intervolvimento*.

Na seção *Calibração* (Figura 6) pode-se definir funções de calibração para as portas. No campo *Porta* deve-se selecionar uma das portas registradas anteriormente. Note que já são listados os *alias*. No campo *Alias* deve-se definir um novo nome para a porta. Ambas estarão disponíveis nos formulários e código posteriormente. O campo *Calibração estática* permite a definição de uma fórmula matemática que ligue as duas portas.

Como as portas serão relacionadas depende se ela é uma entrada ou saída. No caso de entrada, a nova porta *Alias* recebe o valor da expressão quando se substitui o valor de x pelo valor da porta selecionada. No caso de saída, o contrário. Tomando os valores preenchidos como exemplo, temos:

$$Level1CM = 0.003502821818313 * Level1 - 22.161089959823745$$

Calibração			
Porta	Alias	Calibração Estática	
Porta Level1	Alias ▼ Level1CM	Calibração Estática 0.00325145413 * x - 20.4711036	
Porta Pump1	Alias ▼ Pump1PC	Calibração Estática x * 205.7 + 7200	
<b>Adicionar</b>			

Figura 6 – Seção Calibração

e

$$Pump1 = Pump1PC * 205.7 + 7200.$$

Na seção *Intertravamento* (Figura 7) é possível configurar expressões que, quando verdadeiras, irão definir o valor de outra variável, além de parar o teste. Exemplo disso é o intertravamento em um sistema de tanques comunicantes, onde, caso o nível de água ultrapasse 70 centímetros, o teste é parado e a variável *SystemOnOff* assume valor zero, desligando o sistema.

Intertravamento				
Entrada	Expressão	Saída	Valor	
Entrada Level1CM	▼ x >= 70	Saída SystemOnOff	▼ 0	
Entrada Level2CM	▼ x >= 70	Saída SystemOnOff	▼ 0	
<b>Adicionar</b>				

Figura 7 – Seção Intertravamento

No final há opções de exportar/importar configurações de *hardware* para/de um arquivo, possibilitando o *backup* apenas das configurações do *driver* ou o compartilhamento de configurações entre pessoas. O botão *Reiniciar* reinicia o formulário, sem salvar as alterações. Para salvar uma configuração nova é necessário clicar na opção *Aplicar*. Nenhuma configuração será salva se esse botão não for pressionado.

# Capítulo 4

## Resposta do Sistema

Nesse módulo são feitas as configurações de teste em malha aberta. O nome do módulo remete ao fato que ele permite obter a resposta do sistema à um sinal de entrada. Na Figura 8 pode-se ver a lista de testes configurados, bem como as opções de gerenciar tais testes.

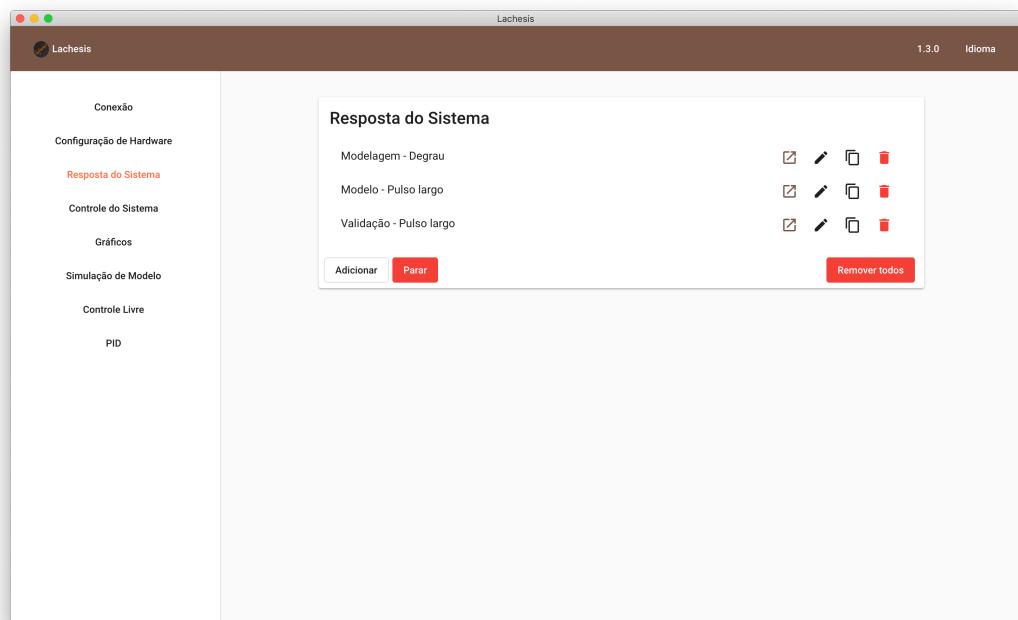


Figura 8 – Módulo Resposta do Sistema

O botão *Clonar* ( ) cria uma cópia do teste. Essa opção é útil para criar vários testes que possuem apenas pequenas diferenças entre si. O botão *Executar* ( ) inicia o teste e navega para o módulo *Gráficos*. O botão *Parar* irá interromper **qualquer** teste que esteja em execução, não importa o tipo.

Ao adicionar ou editar um teste é possível criar um sinal facilmente, apenas preenchendo um formulário. Várias formas de onda estão disponíveis para preenchimento rápido. Cada opção será detalhada nas seções a seguir. Algumas configurações são comuns e serão tratadas na Seção 4.1.

## 4.1 Degrau

Todo teste tem um nome. O nome do teste, juntamente com sua data de execução, serão exibidos na lista de gráficos, onde pode-se ver os gráficos e exportar os dados (Ver Capítulo 6). Portanto, escolha um nome que permita identificar o motivo de tal teste ter sido realizado e/ou os parâmetros que foram usados em sua execução. Em um ambiente onde mais de uma pessoa utiliza o mesmo sistema, também é boa prática colocar seu nome no teste.

Na aba *Degrau*, mostrada na Figura 9, é possível definir um sinal desse tipo. Para isso deve-se informar o valor inicial  $V_0$ , o acréscimo  $\Delta V$ , que será somado ao valor inicial após  $T_0$  segundos, e o tempo  $T_1$  que o sinal  $V_0 + \Delta V$  ficará aplicado.

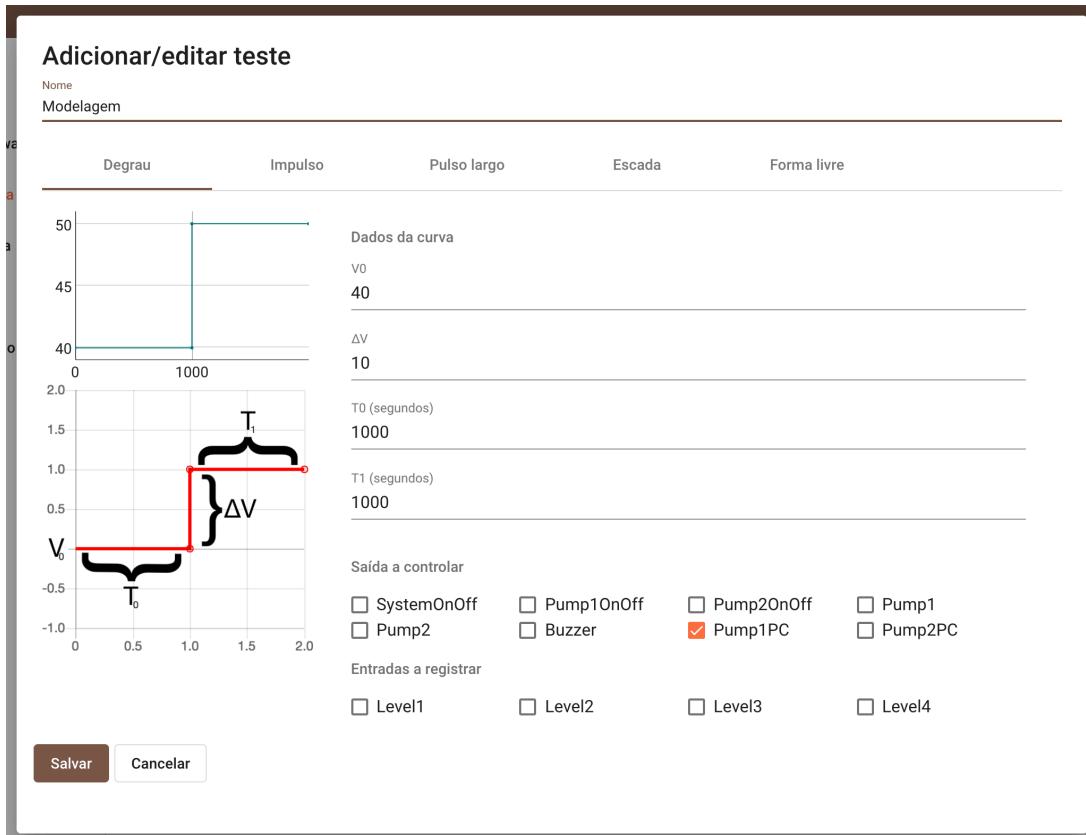


Figura 9 – Sinal do tipo degrau

A seção *Configuração de entrada/saída* permite escolher em quais atuadores o sinal descrito será aplicado. É possível escolher mais de um atuador, mas todos receberam o mesmo sinal. Também deve-se escolher quais entradas serão registradas, como mostrado na Figura 10.

**Adicionar/editar teste**

Entradas a registrar

- Level1
- FlowMeter1
- Level2CM
- Level4CM
- Level2
- FlowMeter2
- Level2PC
- Level4PC
- Level3
- Level1CM
- Level3CM
- FlowMeter1CA
- Level4
- Level1PC
- Level3PC
- FlowMeter2CA

Tempo de amostragem

Tempo de amostragem  
1

Saídas fixas

Saída	Valor	Remover
SystemOnOff	1	
Pump1OnOff	1	

Adicionar

Saídas após os testes

Saída	Valor	Remover
SystemOnOff	0	

Adicionar

**Salvar** **Cancelar**

Figura 10 – Configuração comum

*Tempo de amostragem* é o tempo entre cada leitura dos sensores. Os atuadores também serão atualizados apenas depois desse tempo desde a última atualização.

*Saídas fixas* são portas que terão seus valores escritos antes do teste iniciar. Seu principal objetivo é permitir definir valores de portas que atuam como chave geral do sistema. Da mesma forma *Saídas após o teste* serão definidas após o teste terminar, ou caso algum erro ocorra durante a execução, como, por exemplo, o acionamento de um intertravamento.

## 4.2 Impulso

Nessa aba pode-se definir um sinal do tipo impulso discreto. Deve-se definir as durações e o valor inicial, mas não a variação da amplitude, como no caso do sinal degrau. Essa amplitude será calculada de forma a fazer com que o gráfico do sinal tenha área 1 na região sob  $\Delta T$ .

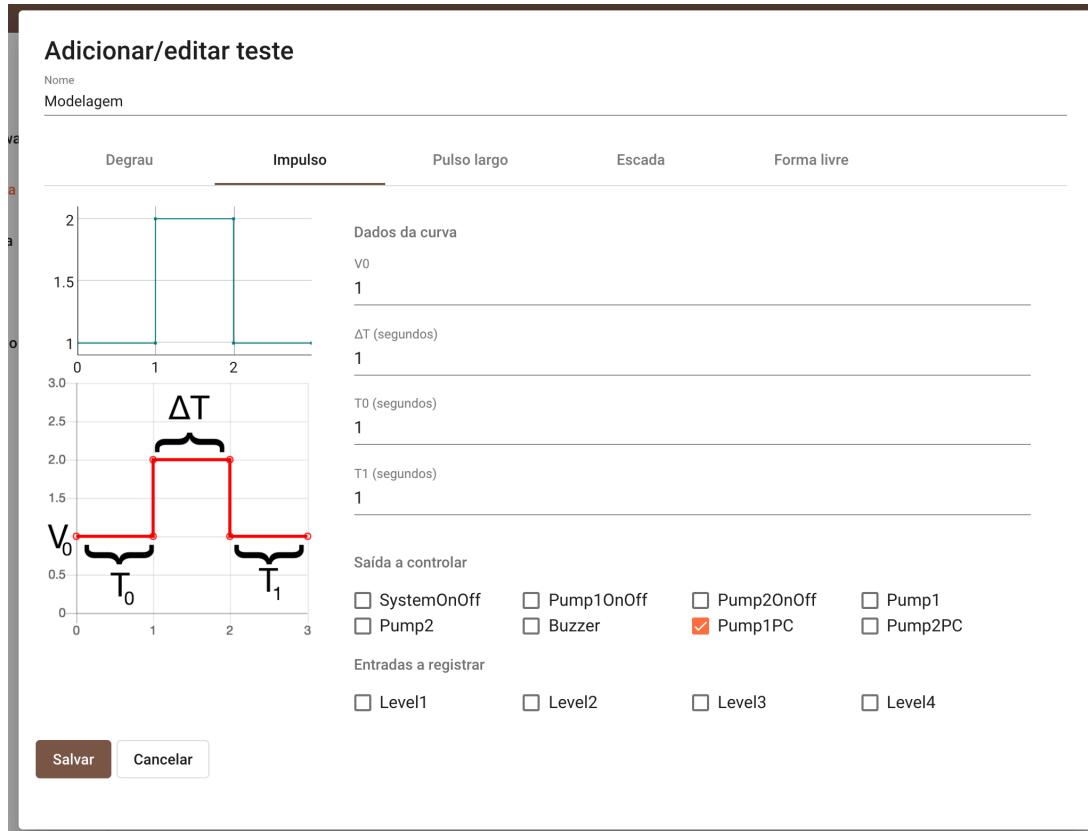


Figura 11 – Impulso

## 4.3 Pulso largo

Parecido com o sinal do tipo degrau, porém deve-se definir um segundo acréscimo e duração, fazendo com que esse sinal seja a composição de dois sinais do tipo degrau.

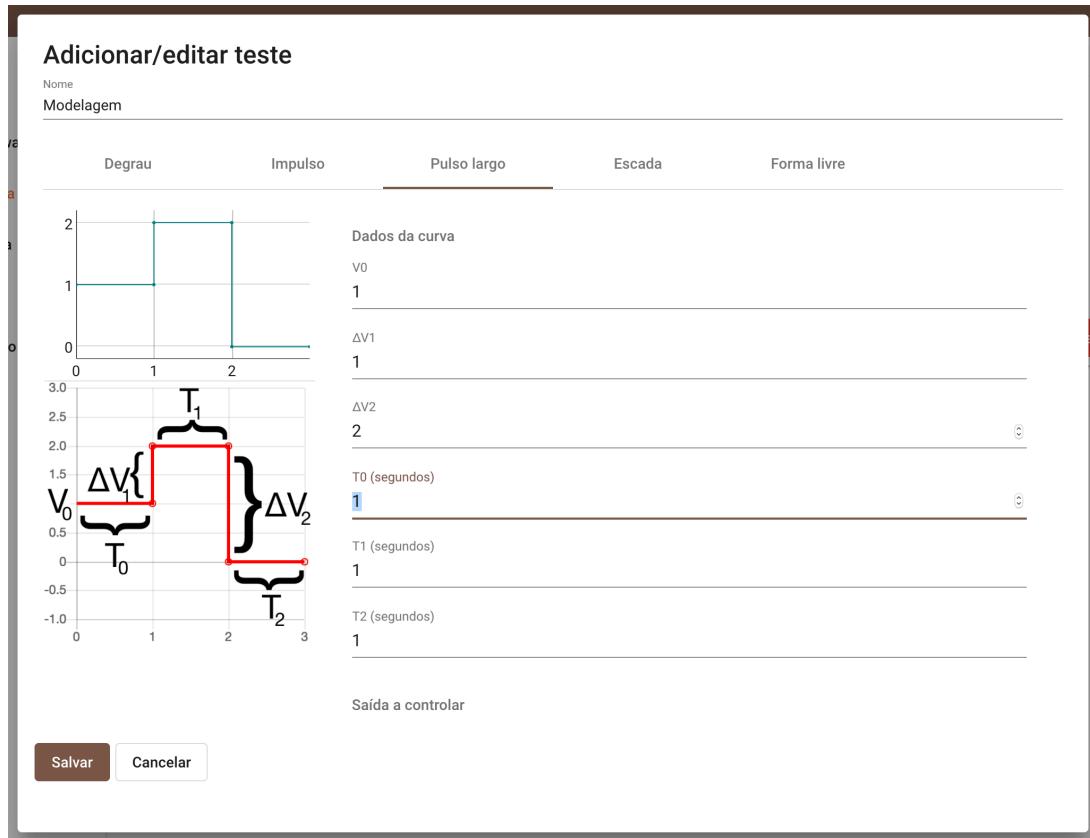


Figura 12 – Pulso largo

## 4.4 Escada

Permite a configuração simples de um sinal do tipo escada. Basta informar os valores inicial e final, o número de degraus e a duração de cada degrau.

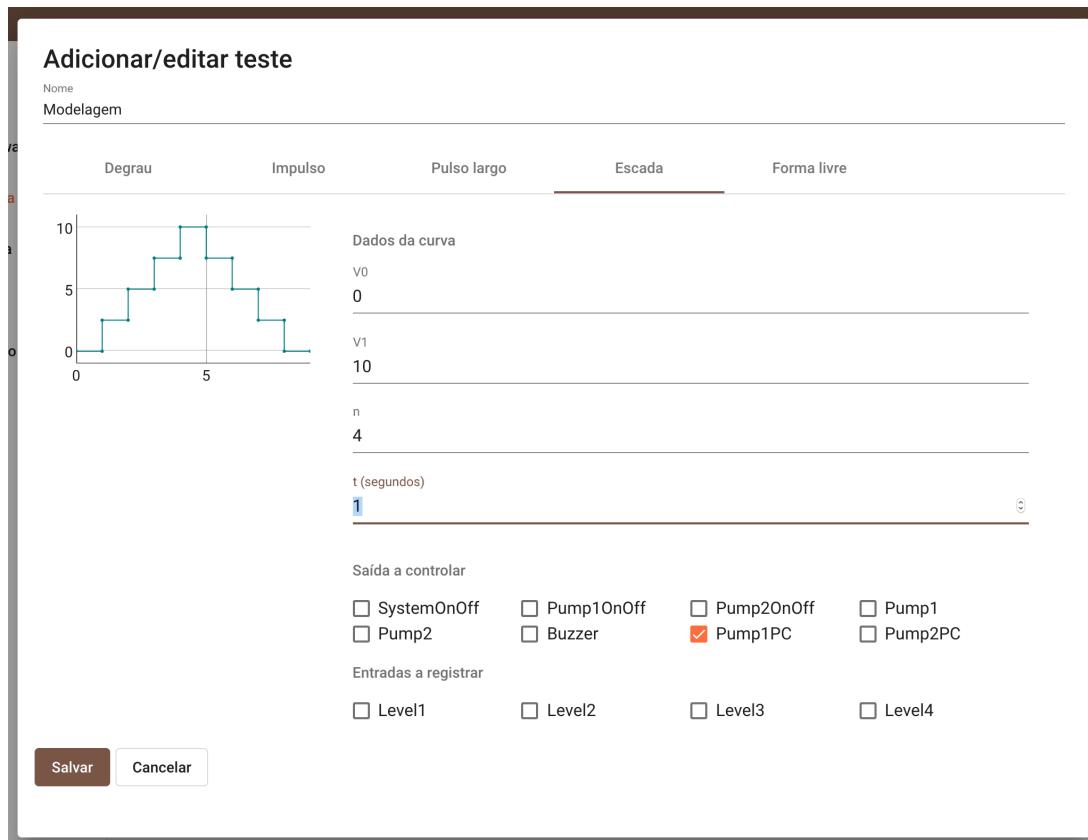


Figura 13 – Escada

## 4.5 Forma livre

Nessa aba o sinal pode ser construído ponto a ponto. Também é possível importar dados de outros aplicativos nos formatos *JSON* e *CSV*. No caso do *JSON* os dados devem ser no formato `[{"x": 0, "y": 0}, {"x": 1, "y": 1}]`. Caso seja usado *CSV*, o tempo deve vir na primeira coluna e a amplitude na segunda.

O comando `csvwrite("data.csv", transpose([x; y]))` pode ser usado para transformar vetores de valores em *CSV* no *MATLAB*. Após executar esse comando, um arquivo como o nome *data.csv* será criado. Basta abri-lo com um editor de texto e copiar o conteúdo para a caixa e clicar em *Importar*.

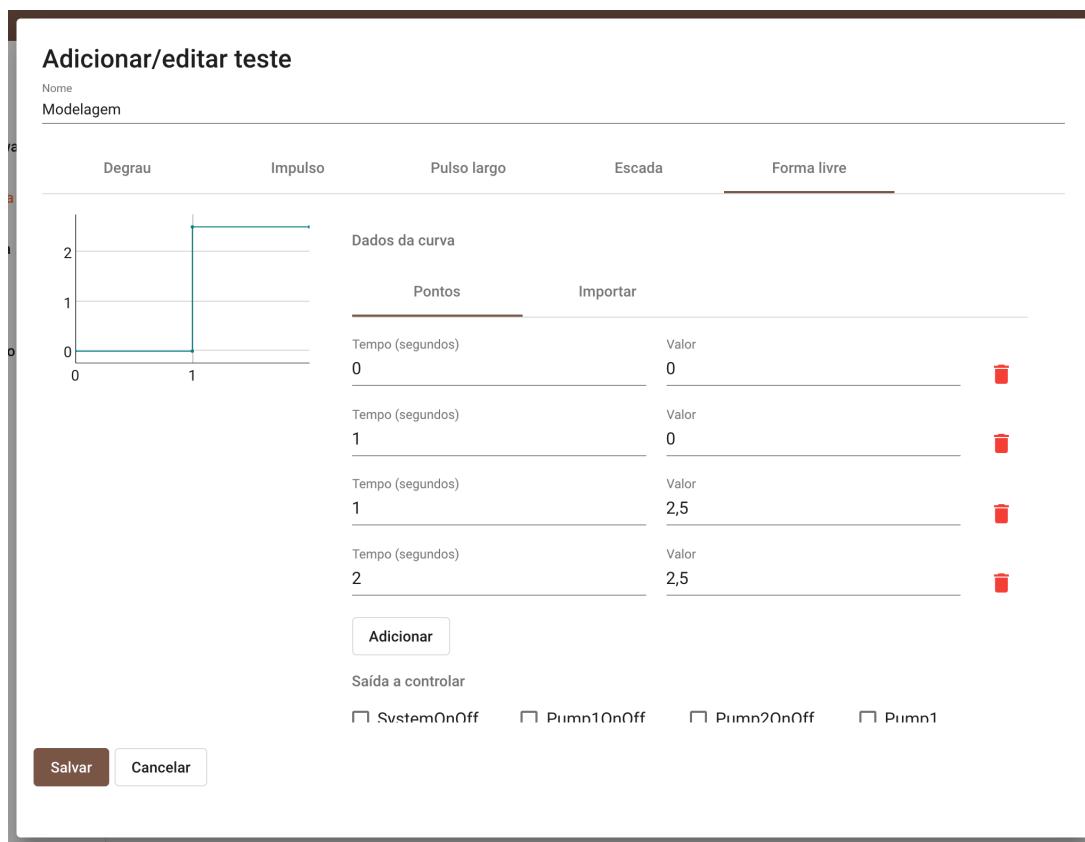


Figura 14 – Forma livre



# Capítulo 5

## Controle do Sistema

Esse módulo permite que o usuário escreva testes utilizando a linguagem de programação *Python*. Dessa forma o usuário tem total liberdade para controlar as saídas do sistema, e, por dispor de uma linguagem de programação completa, possibilita o desenvolvimento de qualquer tipo de controlador. Embora ideal para controle em malha fechada, também é útil para criar sinais complexos para a malha aberta, além de permitir a aplicação de sinais diferentes em portas diferentes.

A tela inicial do módulo, representada na Figura 15, lista os controladores salvos. As opções são as mesmas presentes no módulo *Resposta do Sistema* (Capítulo 4).

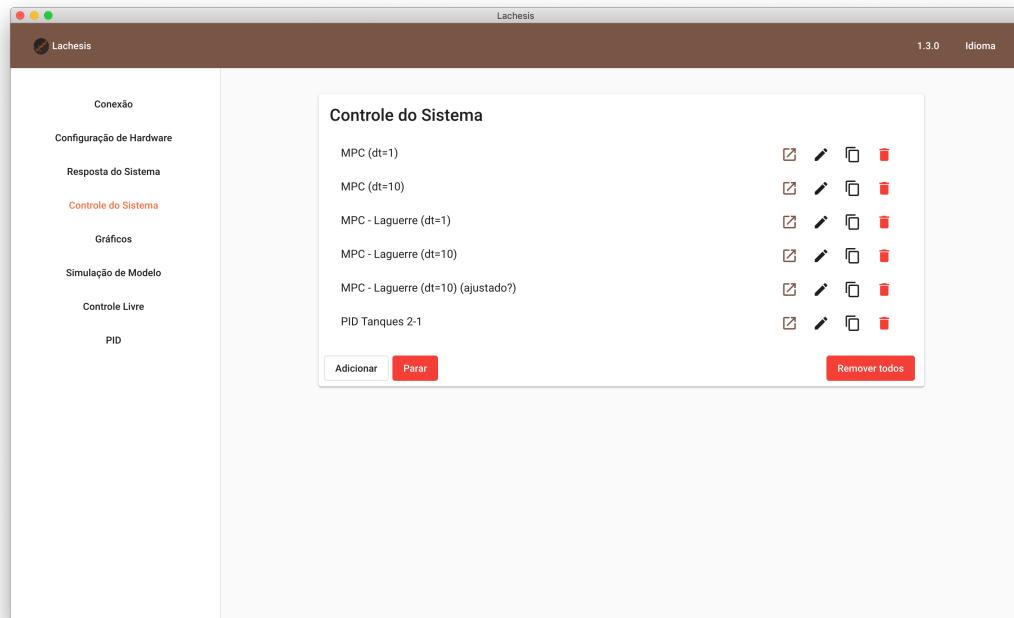


Figura 15 – Módulo Controle do Sistema

Ao clicar em *Adicionar* ou em *Editar* (  ), uma nova interface (Figura 16) se abre permitindo a edição do controlador. No topo temos os campos *Nome* — assim como em *Resposta do Sistema* e sendo pertinente as mesmas recomendações — *Tempo de amostragem* e *Tempo total de execução*, que é o tempo máximo que o teste será executado. O mesmo pode ser interrompido precocemente por motivos de erro ou intertravamento.

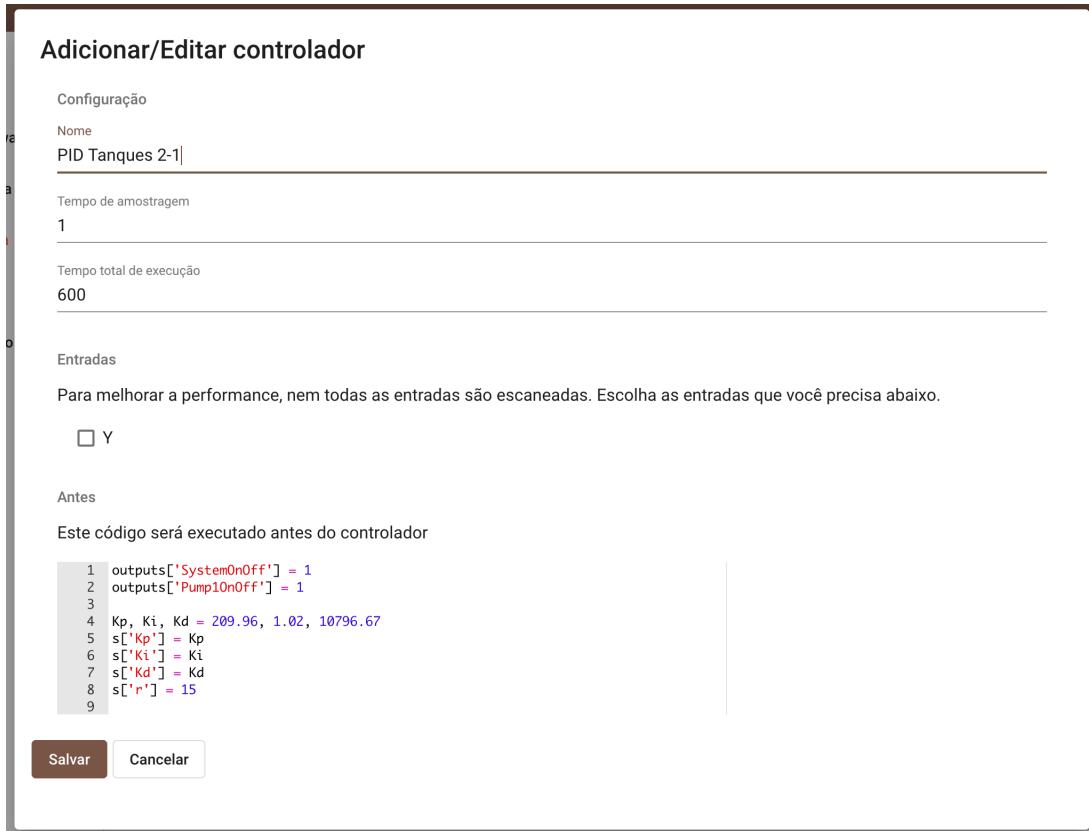


Figura 16 – Edição de controlador

Na seção *Entradas* o usuário deve escolher as portas de entrada que serão lidas. Apenas as portas selecionadas serão lidas e estarão disponíveis no código. Isso evita ler informações desnecessárias e agiliza a execução do código.

No campo *Antes*, visto na Figura 17, podemos inserir o código que será executado, uma vez, antes do controlador iniciar. Esse campo é ideal para calcular ganhos estáticos (como de observadores ou *MPC*) e para salvar informações constantes e valores iniciais, como  $K_p$  ou  $x_0$ .

Algumas variáveis estão disponíveis nesse campo que o usuário pode fazer uso:

- **inputs** um dicionário com o valor lido na entrada logo antes da execução desse campo. Na Figura 16 podemos ver que a entrada *S9* foi selecionada, logo podemos

**Adicionar/Editar controlador**

Y

**Antes**

Este código será executado antes do controlador

```

1 outputs['SystemOnOff'] = 1
2 outputs['Pump1OnOff'] = 1
3
4 Kp, Ki, Kd = 209.96, 1.02, 10796.67
5 s['Kp'] = Kp
6 s['Ki'] = Ki
7 s['Kd'] = Kd
8 s['r'] = 15
9
10 s['ei'] = 0
11 s['ed'] = 0
12

```

**Controlador**

```

1 def qinv(f, b):
2     if b == 1:
3         return (f + 336.814503666668) / 22.6242879333334
4     else:
5         return (f + 290.475253333334) / 17.18420266666667
6

```

**Salvar** **Cancelar**

Figura 17 – Campo *Antes*

ler seu valor acessando `inputs['S9']`.

- **outputs** um dicionário inicialmente vazio. Ao fim da execução desse bloco, todas as saídas que forem adicionadas a esse dicionário serão aplicadas no *hardware*. Por exemplo, para ligar a bomba 1 de um sistema, configurada em *Configurações de Hardware* como *Pump1PC*, com 40% de potência, podemos utilizar o código `outputs['Pump1PC'] = 40`.
- **s** variável de estado. Essa variável é um dicionário e estará disponível na próxima seção. Ela deve ser usada para persistir dados durante a execução do teste. Por exemplo, o estado atual de um modelo em espaço de estados pode ser salvo nessa variável e alterado posteriormente durante a execução do controlador. É como se fosse uma variável global. Exemplo de uso: `s['Kp'] = 1`.
- **dt** o tempo de amostragem.
- **np** a biblioteca numpy já é importada e está disponível sob o nome *np*, não sendo necessário reimportá-la.

- **math** a biblioteca math já é importada, não sendo necessário reimportá-la.

No campo *Controlador* (Figura 18) deve-se inserir o código que será executado a cada *tempo de amostragem* segundos. As mesmas variáveis disponíveis no campo *Antes* estão disponíveis nesse campo, além de:

- **t** tempo de execução em segundos.
- **log** todas as entradas e saídas nas variáveis **inputs** e **outputs** são salvas e podem ser vistas em gráficos e exportadas para arquivos. A variável **log** permite fazer a mesma coisa com variáveis que não são entrada ou saída. Por exemplo, pode-se gerar uma entrada para o erro na saída adicionando o erro a esse dicionário, assumindo que o valor da saída anterior esteja salvo como *y\_anterior*: `log['erro'] = y - s['y_anterior']`. Na seção *Gráficos* (Capítulo 6) a variável *erro* estará disponível nas opções de variáveis de gráficos e de exportação de dados.

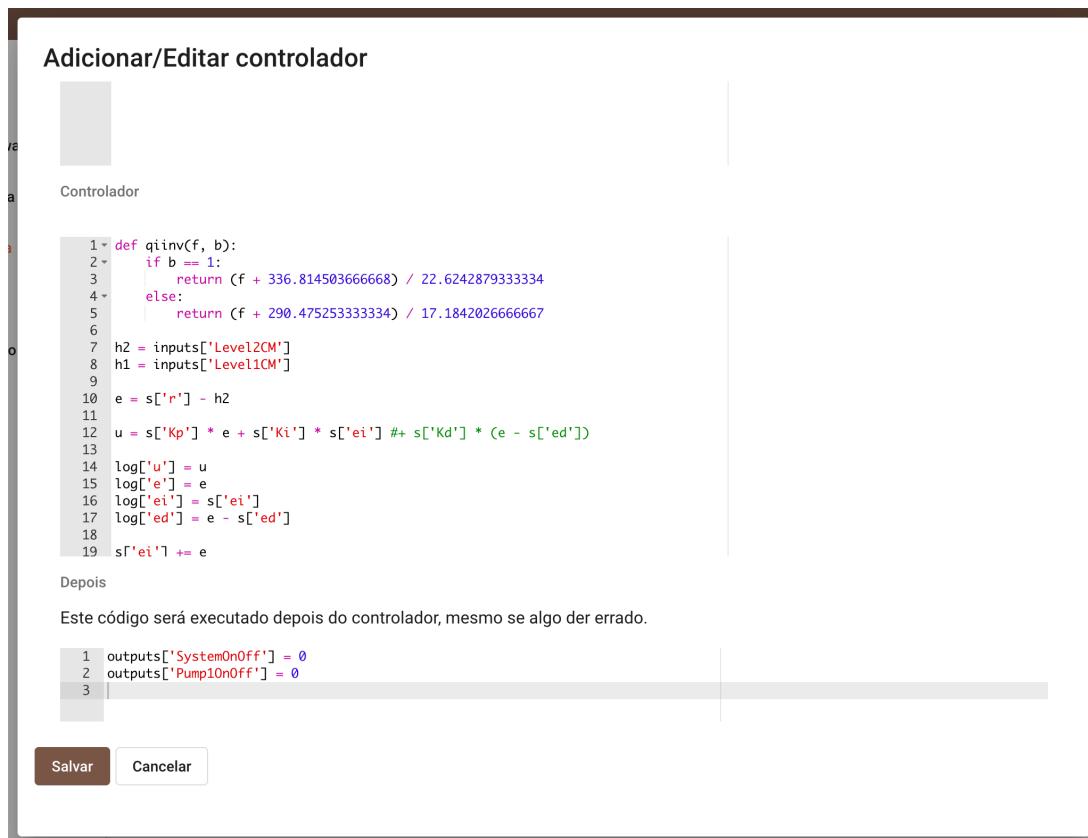


Figura 18 – Campo *Controlador*

No campo *Depois* (Figura 19) pode-se inserir código para desligar o sistema. Esse campo será executado em caso de erro ou caso o teste execute até o final. As variáveis *inputs*, *outputs*, *math* e *np* estão disponível nesse campo.

```

12 u = s['Kp'] * e + s['Ki'] * s['ei'] #+ s['Kd'] * (e - s['ed'])
13
14 log['u'] = u
15 log['e'] = e
16 log['ei'] = s['ei']
17 log['ed'] = e - s['ed']
18
19 s['ei'] += e
20 s['ed'] = e
21
22 outputs['Pump1PC'] = int(sorted([qinv(u, 2), 0, 100])[1])
23

```

Depois

Este código será executado depois do controlador, mesmo se algo der errado.

```

1 outputs['SystemOnOff'] = 0
2 outputs['Pump1OnOff'] = 0
3

```

**Salvar** **Cancelar**

Figura 19 – Campo *Depois*

O pseudo-código abaixo representa o funcionamento do programa, como se os campos fossem funções. Obviamente a implementação real é bem mais complexa. Deve-se atentar ao fato que o controlador será executado sempre num tempo múltiplo de  $dt$ . O cálculo do tempo de espera leva em conta o tempo que foi necessário para executar o código, e sempre aguardará para iniciar a próxima execução em um múltiplo. Se o tempo de execução do código for menor que  $dt$ , então o código sempre executará em  $k*dt$  segundos. Se o código levar mais de  $dt$  segundos para executar, a próxima execução será agendada para o próximo  $k*dt$ .

---

```
1 import math
2 import numpy as np
3
4 s = dict()
5 output = dict()
6 inputs = moirai.read_inputs()
7 dt = moirai.test_dt()
8
9 s, outputs = user.antes(inputs, output, s, dt, math, np)
10
11 moirai.write_outputs(outputs)
12
13 while True:
14     t = moirai.sleep(dt)
15     inputs = moirai.read_inputs()
16     log = dict()
17     outputs = dict()
18
19     s, outputs, log = user.controlador(inputs, s, dt, math, np, t, log)
20
21     moirai.write_outputs(outputs)
22     moirai.save_variables(inputs, outputs, log)
23
24 output = dict()
25 outputs = user.depois(outputs)
26 moirai.write_outputs(outputs)
```

---

# Capítulo 6

## Gráficos

Esse módulo exibe os gráficos dos testes realizados e do teste em execução. Nele é possível visualizar os sinais em tempo real durante a execução do teste. Nos testes finalizados é possível exportar os dados. Na Figura 20 podemos ver a listagem de gráficos com as opções pertinentes. O botão *Parar* funciona como nos módulos *Resposta do Sistema* e *Controle do Sistema*.

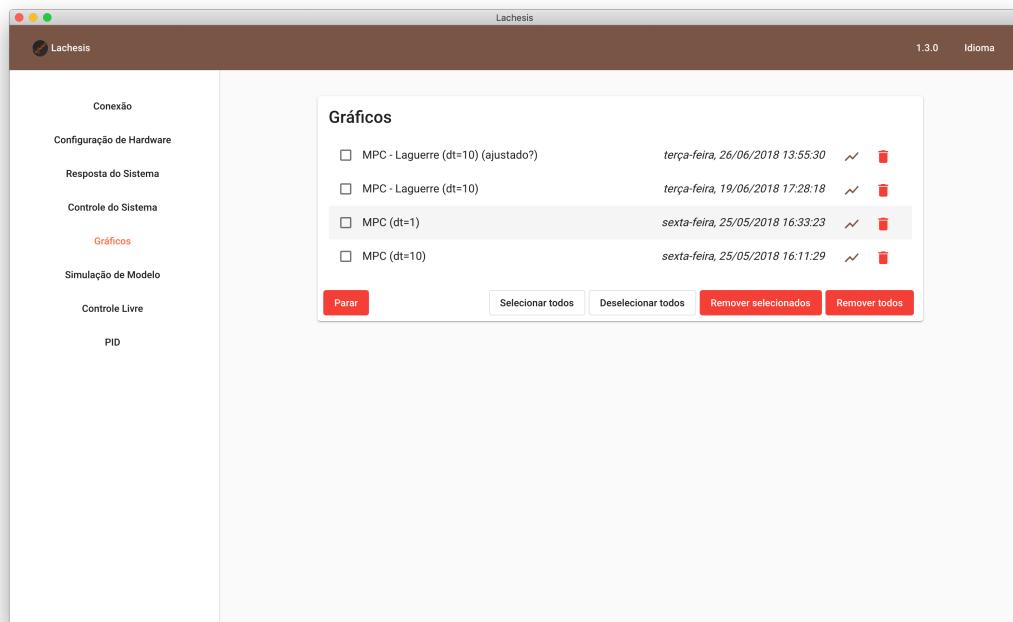


Figura 20 – Módulo Gráficos

Ao clicar em *Exibir* (↗), a seção *Gráficos* será exibida. A seção *Exportar* também aparecerá, mas apenas caso o teste não esteja em execução. A seção *Exportar* é retrátil, se expandindo ao clicar no título ou botão no lado direito. Na Figura 21 pode-se ver a

---

seção expandida. No topo pode-se selecionar o formato dos dados exportados: JSON, CSV ou MAT.

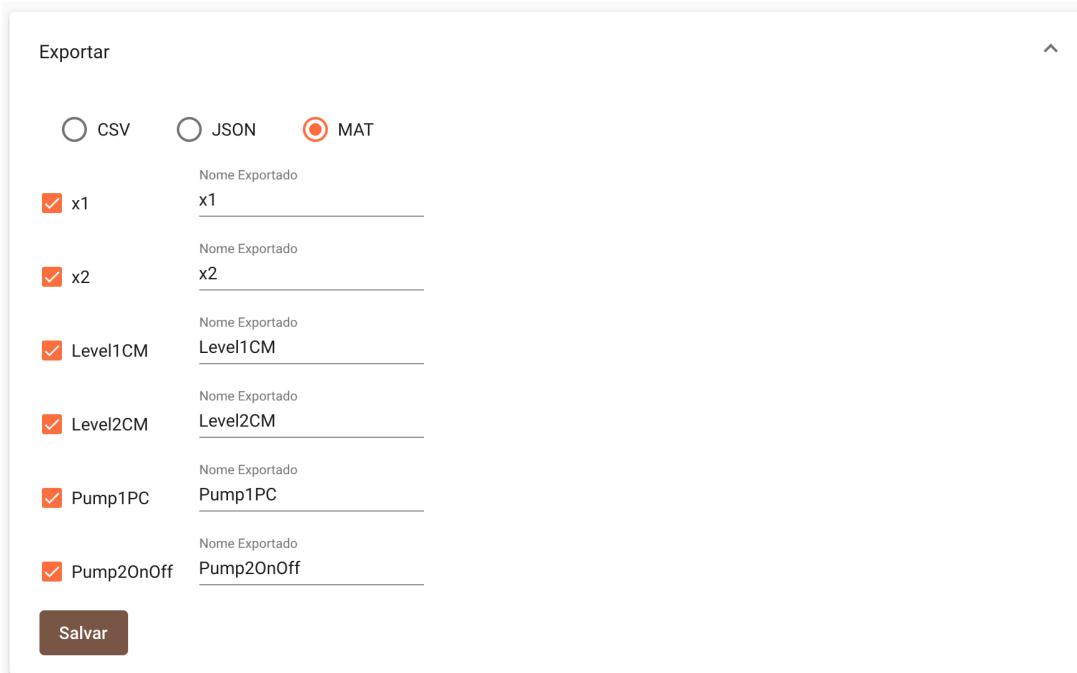


Figura 21 – Exportar dados

Logo em seguida têm-se uma lista de variáveis com *checkboxes* e caixas de texto. As variáveis com o *checkbox* selecionado serão exportadas, e o nome definido na caixa de texto será utilizado no arquivo final, ao invés do nome da variável, permitindo assim que as variáveis sejam renomeadas durante a exportação. Ao clicar em *Salvar* um arquivo no formato especificado é gerado e uma janela de salvar arquivo é exibida, permitindo escolher o local onde o arquivo será salvo.

A seção *Gráficos*, mostradas na Figura 22, permite adicionar gráficos contendo as curvas de uma ou mais variáveis. As variáveis disponíveis serão as entradas, saídas e aquelas definidas no dicionário *log*.

É possível adicionar e remover gráficos sem restrição. Para cada gráfico pode-se selecionar quais variáveis serão exibidas nele. Algumas funcionalidades de conveniência estão disponíveis, como a opção de adicionar um gráfico para cada variável e de selecionar ou deselecionar todas as variáveis em um único gráfico.

Ao passar o *mouse* sobre o gráfico, a legenda é exibida mostrando o nome de cada curva na mesma cor da linha utilizada no gráfico, além do valor no ponto atual. O valor eixo *x*, que sempre representa o tempo em segundos, é exibido antes de todas as variáveis.

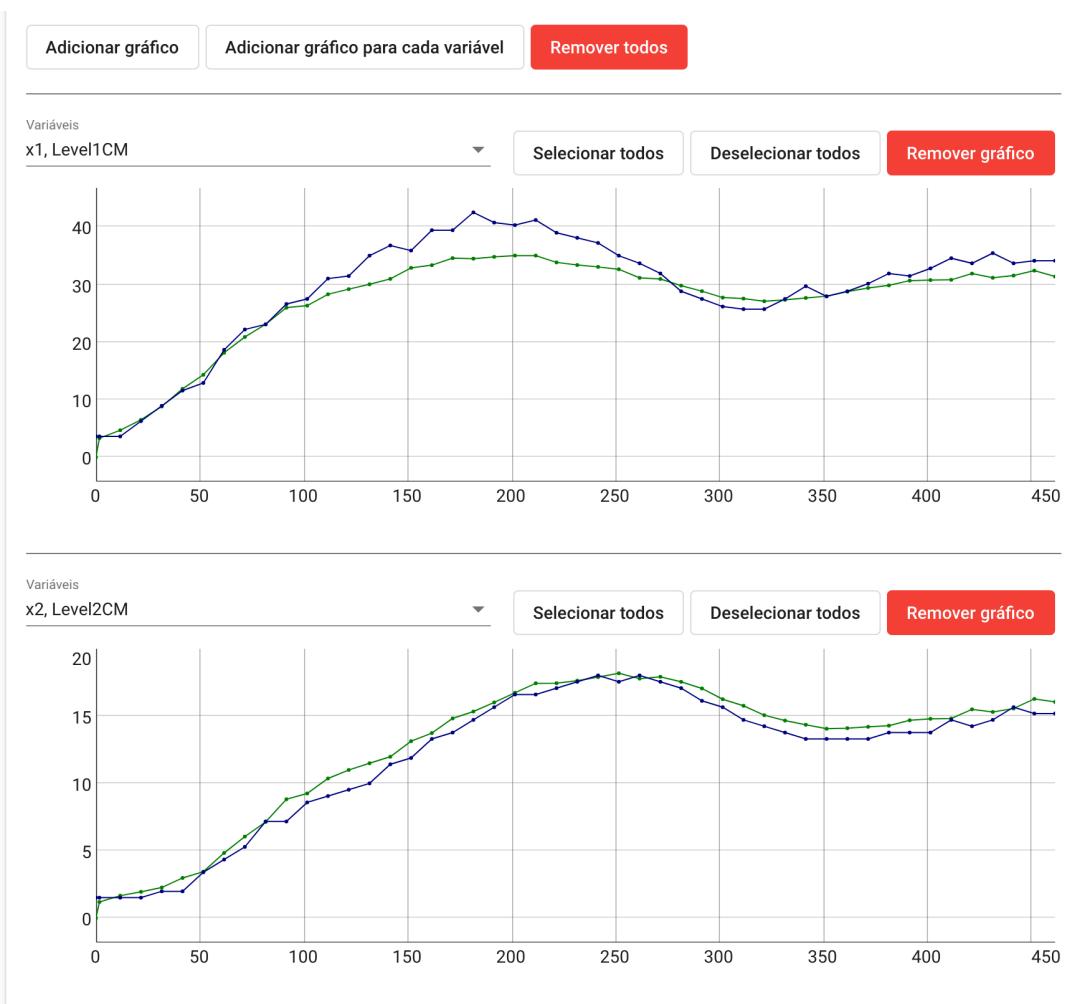


Figura 22 – Gráficos

Modelo

domingo, 24/02/2019 18:30:20

Figura 23 – Teste em execução

Quando um teste está em execução, ele é marcado com o símbolo , como pode ser visto na Figura 23. O botão *Remover* () é substituído pelo botão *Parar* () . Os gráficos de um teste em execução são atualizados a cada segundo. Essa atualização requer considerável processamento pelo aplicativo **moirai**, sendo recomendado não manter o módulo *Gráficos* aberto durante a execução de um teste em um computador com baixo poder de processamento. A lista de gráficos disponíveis também é atualizada a cada segundo, fazendo com que um teste em execução não seja listado imediatamente, mas após alguns segundos.



# Capítulo 7

## Simulação do Modelo

Esse módulo permite a simulação de um modelo. Na Figura 24 pode-se ver as opções de configuração da simulação. A simulação suporta modelos contínuos e discretos, em espaço de estados e função de transferência.

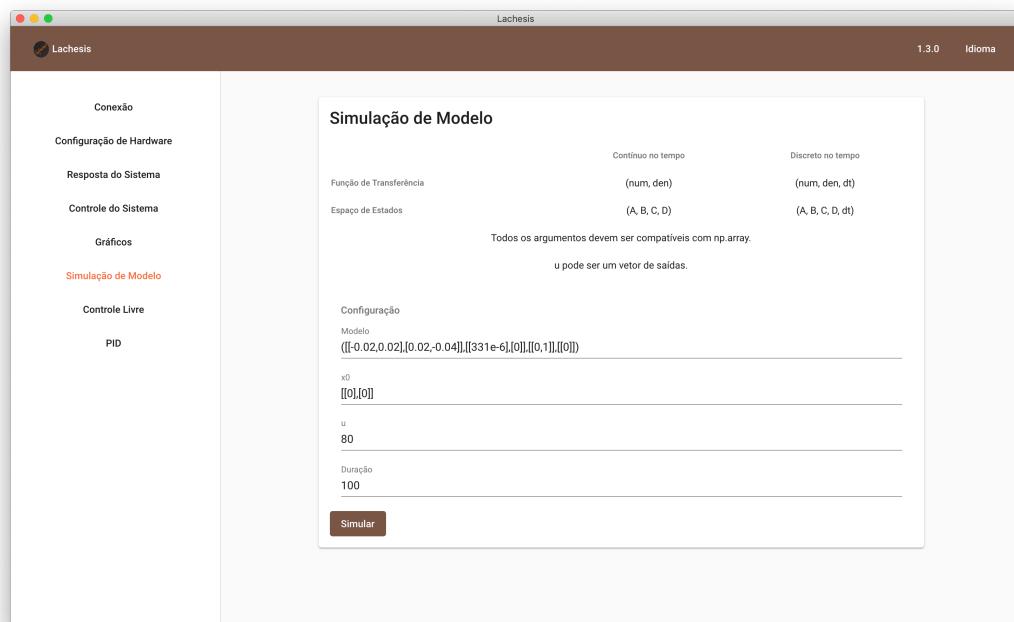


Figura 24 – Módulo Simulação de Modelo

Caso o modelo não esteja em espaço de estados discreto, ele será convertido para tal. No caso de modelos contínuos, o tempo de amostragem é calculado como:

```
dt = max(0.1, float('%.1f' % (max(abs(np.real(vals))) / 5)))
```

em que *vals* são os polos do sistema.

---

Caso o sinal de controle  $U$  seja informado como um número, ele será aplicado de forma constante pela duração da simulação. Caso seja um vetor, cada item será aplicado por uma amostragem, e o valor do campo *Duração* será ignorado.

Assim, considerando um sistema discreto com  $\delta t = 5$  segundos, podemos definir um sinal de controle em rampa de 0 a 100 e retornando a zero como:

```
[*range(0,100,5),*range(100,0,-5)].
```

Tal sinal de controle aplica uma variação de 5 a cada amostragem, e tem duração de `len([*range(0,100,5),*range(100,0,-5)]) * dt`, no caso, 200 segundos.

Os campos *Modelo*, *x0* e *U* aceitam código *Python* como entrada, como pode ser visto exemplo anterior. Como a simulação é sempre feita em espaço de estados discreto, é sempre necessário informar um *x0* adequado.

O resultado da simulação é salva como um teste, e pode ser vista também no módulo *Gráficos*, onde é possível exportar os dados.

# Capítulo 8

## Controle Livre

### Atenção!

A execução deste módulo se basea na existência de conexão entre o **Lachesis** e o **moirai**. Para parar a execução você deve navegar para um outro módulo. Os dados são atualizados a cada segundo mas apenas são aplicados quando o **moirai** os processar, o que pode levar mais de um tempo de amostragem, **inclusive para parar o teste!**

**NÃO DEIXE ESTE MÓDULO EXECUTANDO SEM ESTAR POR PERTO E ATENTO!!!**

O módulo *Controle Livre* (Figura 25) permite controlar a planta livremente. Nele é possível alterar os valores das saídas manualmente enquanto o teste está em execução.

Selecione um tempo de amostragem. É recomendado um tempo de amostragem de 1 segundo ou mais. Escolha as entradas que deseja exibir. Uma vez que o teste foi iniciado, não é possível alterar as entradas. Adicione saídas e seus valores. Essas podem ser modificadas enquanto o teste está em execução. Clique em *Executar*.

Com o teste em execução, os gráficos serão exibidos em uma seção ao final da página. Todos os dados são salvos e estarão disponíveis no módulo *Gráficos*. Basta alterar o valor da saída e clicar fora do campo, fazendo com que ele perca o foco, para que o valor seja aplicado ao sistema real.

Pode haver uma demora de mais de um tempo de amostragem entre alterar o valor e ele ser aplicado. Para parar o teste, basta navegar para outro módulo. É necessário, no mínimo, um tempo de amostragem para que seja detectado a perda de conexão e o teste seja finalizado.

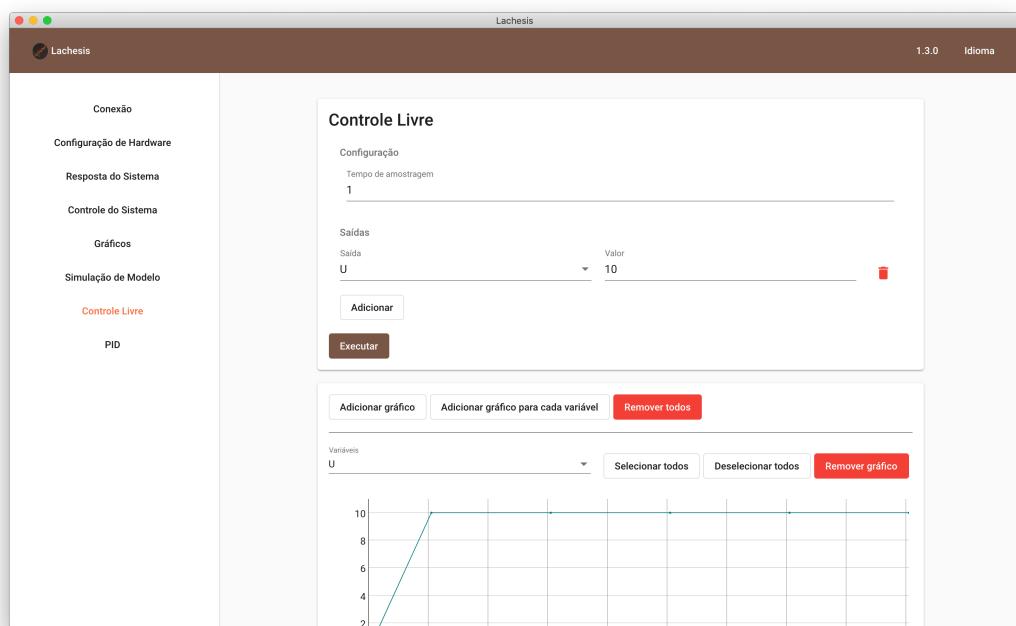


Figura 25 – Módulo Controle Livre

# Capítulo 9

## PID

### Atenção!

A execução deste módulo se basea na existência de conexão entre o **Lachesis** e o **moirai**. Para parar a execução você deve navegar para um outro módulo. Os dados são atualizados a cada segundo mas apenas são aplicados quando o **moirai** os processar, o que pode levar mais de um tempo de amostragem, **inclusive para parar o teste!**

**NÃO DEIXE ESTE MÓDULO EXECUTANDO SEM ESTAR POR PERTO E ATENTO!!!**

Esse módulo, exibido na Figura 26 permite executar um controlador PID, e a alterar os valores durante a execução. *Entrada* e *Saída* são do ponto de vista do *software*, ou seja, *Entrada* é um sensor e *Saída* um atuador. *Min* e *Max* definem os valores mínimo e máximo de saturação do atuador, respectivamente.

Assim como em *Controle Livre*, saídas fixas podem ser adicionadas e modificadas durante o teste. Os atrasos nas atualizações de valores e detecção de perda de conexão também ocorrem da mesma forma. Os dados também são salvos e ficam disponíveis no módulo *Gráficos*.

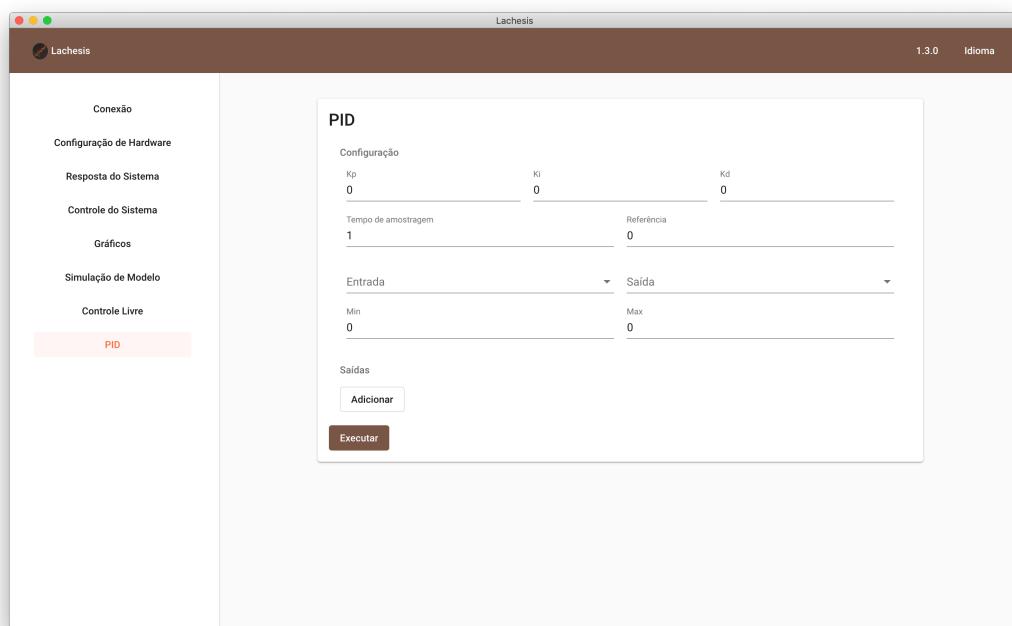


Figura 26 – Módulo PID

# Bibliografia

- [1] Á. C. Sousa, *Lachesis*, 2017. endereço: <https://github.com/acristoffers/lachesis> (acesso em 23/05/2018).
- [2] ——, *Moirai*, 2017. endereço: <https://github.com/acristoffers/moirai> (acesso em 23/05/2018).
- [3] ——, *AHIO*, 2016. endereço: <https://github.com/acristoffers/ahio> (acesso em 23/05/2018).