

Estudo e aplicação de técnicas de *simulação hardware in the loop* para uso no ensino de controle e automação industrial

Igor André Houri e Costa

Escola de Engenharia - Colegiado de Engenharia de Controle e
Automação, UFMG



Projeto Final de Curso

Orientador: Prof. Anísio R. Braga, UFMG

Belo Horizonte-MG, fevereiro de 2022

Estudo e aplicação de técnicas de *simulação hardware in the loop* para uso no ensino de controle e automação industrial

Igor André Houra e Costa

Escola de Engenharia - Colegiado de Engenharia de Controle e Automação, UFMG

Belo Horizonte-MG, fevereiro de 2022

Resumo

A simulação *Hardware in the loop* (HIL) combina a operação de componentes reais e componentes simulados de um mesmo sistema para possibilitar o projeto e o teste de diferentes tipos de peças e projetos. Suas principais vantagens consistem na redução do custo de desenvolvimento, maior segurança e repetibilidade dos experimentos. Trata-se, então, de uma técnica adequada para aulas práticas em cursos de ensino de engenharia ou técnicos, visto que processos industriais complexos têm sua instalação em ambientes limitadas por questões de custo e/ou espaço físico. Daí surge a proposta do presente trabalho: estudar, aplicar e desenvolver técnicas de simulação HIL voltadas para o ensino de controle e automação industrial. Este projeto é parte integrante do projeto do MICAsh, "o Módulo de Instrumentação, Controle e Automação - sistemas híbridos", um dispositivo pedagógico que terá dentre suas muitas funcionalidades, o modo de experimentos *Hardware in the loop*. No desenvolvimento do trabalho, foram desenvolvidos modelos de simulação física de sistemas utilizando o Simscape, do Matlab, e métodos e protocolos de comunicação serial com o ESP32s; principal parte da integração entre o mundo real (MICAsh e periféricos) e o mundo simulado (Simscape). Foram avaliados se os modelos simulados correspondem ao comportamento esperado dos sistemas representados e se há perda expressiva de informações na comunicação com o ESP32s. Os resultados encontrados demonstram que o Simscape e os métodos criados para a comunicação podem ser utilizados em experimentos HIL e podem ser integrados ao MICAsh para o uso em aulas práticas.

Palavras-chave: *Hardware in the loop*, ensino de controle e automação, MICAsh, Simscape, ESP32-S.

Sumário

1	Introdução	1
1.1	Simulação <i>Hardware in the loop</i>	1
1.1.1	A simulação <i>Hardware in the loop</i> na educação	3
1.1.2	Módulo de Instrumentação, Controle e Automação - sistemas híbridos	3
1.2	Escopo do trabalho	6
1.3	Objetivos do projeto	6
1.4	Comentários finais	6
2	Descrição das ferramentas utilizadas e suas aplicações	8
2.1	Características e uso dos recursos utilizados	8
2.1.1	MATLAB e Simscape	8
2.1.2	Raspberry PI 3B	9
2.1.3	ESP 32-S	9
2.2	Características dos sistemas-exemplo modelados	11
2.2.1	Tanques cilíndricos desacoplados	11
2.2.2	Filtro passa-baixas	13
2.2.3	Termopar	15
2.3	Comentários finais	16
3	Descrição da metodologia e métodos	17
3.1	Funções e blocos utilizados na comunicação	17
3.1.1	Funções e blocos utilizados para o envio da entrada do processo	18
3.1.2	Funções e blocos utilizados para o envio da saída do processo	22
3.2	Diagramas de simulação dos processos-exemplo	27

3.2.1	Tanques cilíndricos desacoplados	27
3.2.2	Filtro passa-baixas	28
3.2.3	Termopar	29
3.3	Procedimentos necessários para utilização dos métodos criados	29
3.3.1	Montagem de hardware	30
3.3.2	Configuração ESP32-S	31
3.3.3	Configuração Simulink/Simcape	31
3.4	Comentários finais	33
4	Análise dos resultados obtidos	34
4.1	Tanques cilíndricos desacoplados	34
4.2	Filtro passa-baixas	37
4.3	Termopar	39
4.4	Análise geral dos valores de erro	41
4.5	Observações Gerais	41
4.6	Comentários finais	42
5	Conclusões e propostas de trabalhos futuros	43
5.1	Propostas de trabalhos futuros	43
5.2	Disponibilidade dos métodos produzidos	44
	Referências	44

Listas de Figuras

1.1	Comparativo das configurações de um experimento que não utiliza o método HIL e outro que utiliza o método HIL. Figura retirada de (17)	2
1.2	Projeto do painel do MICAsh	4
1.3	Projeto da tela do Módulo Computador de Processos (MCP) do MICAsh	4
1.4	Diagrama em blocos da arquitetura do MCP-MICAsh, em que EA inclui os circuitos de condicionamento das Entrada Analógicas, SA das saídas analógicas, ED das Entradas Digitais e SD das Saídas Digitais.	5
2.1	Esquema de tanques desacoplados	12
2.2	Circuito do filtro passa-baixas duplo	13
2.3	Esquema de um termopar	15
3.1	Ícone personalizado do subsistema criado para ler e decodificar a mensagem enviada pelo ESP32-S pelo canal serial do Raspberry Pi.	20
3.2	Diagrama do subsistema criado para ler e decodificar a mensagem enviada pelo ESP32-S pelo canal serial do Raspberry Pi.	20
3.3	Diagrama da função MAP no ambiente Simulink do Matlab.	21
3.4	Ícone personalizado do subsistema criado para codificar e enviar a saída da simulação pelo canal serial do Raspberry Pi.	22
3.5	Diagrama do subsistema criado para codificar e enviar a saída da simulação pelo canal serial do Raspberry Pi.	22
3.6	Diagrama do subsistema criado para codificar e enviar duas saídas da simulação pelo canal serial do Raspberry Pi.	25
3.7	Modelo de simulação física do processo de tanque simples construído no Simscape	28
3.8	Modelo de simulação física do Filtro passa-baixas construído no Simscape . .	28
3.9	Modelo de simulação física do termopar construído no Simscape	29
3.10	Esquema da montagem utilizada nos experimentos HIL	30

3.11	Configuração do ritmo de simulação. O número 1 na caixa de texto diminui o tempo de simulação na escala 1:1, fazendo com que um segundo da simulação seja o mais próximo possível de um segundo real.	32
3.12	Localização da opção de modo " <i>Connected IO</i> "	32
4.1	Resultados do primeiro experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S	35
4.2	Resultados do segundo experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S	36
4.3	Resultados do terceiro experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S	36
4.4	Resultados dos experimentos com o filtro passa-baixas, incluindo comparação das saídas registradas no Simscape e no ESP32-S	38
4.5	Resultados dos experimentos com o termopar, incluindo comparação das saídas registradas no Simscape e no ESP32-S	40

Agradecimentos

Agradeço aos meus pais, André e Adriana, e a meu irmão Victor, pelo apoio, carinho e inspiração em todos os momentos.

Ao Anísio, pela oportunidade, pelas orientações e pelo encorajamento durante a execução deste projeto.

Aos amigos da MEJA, que me permitiram crescer e amadurecer lado a lado. O apoio de vocês foi e é essencial.

Aos amigos da Escola de Engenharia, especialmente Henrique e Bruno. Dividir os desafios da graduação com vocês fez da subida mais alegre e fácil.

A todos os professores que fizeram parte da minha formação humana e profissional, dentro e fora da UFMG. Vocês têm forças e grandezas admiráveis.

Capítulo 1

Introdução

Este trabalho propõe o estudo da *simulação hardware in the loop* e seu uso no ensino de controle e automação industrial, avaliando sua aplicação em aulas laboratoriais desses assuntos. Terá como produto final quatro (4) experimentos *hardware in the loop* com integração com o Módulo de Instrumentação, Controle e Automação – sistemas híbridos (MICAsh) (3) que serão disponibilizados para o uso em aulas.

1.1 Simulação *Hardware in the loop*

A técnica de simulação *Hardware in the loop* (HIL) propõe a operação de componentes reais junto com componentes simulados em tempo real , beneficiando-se do conceito de que sistemas de engenharia são compostos de subsistemas menores e mais simples (17). Em outras palavras, consiste em integrar simulações de subsistemas, implementadas em código ou softwares especializados, a versões físicas dos demais subsistemas para possibilitar o projeto e o teste de diferentes tipos de processos.

Um simulador HIL é entendido como "uma configuração que prototipa partes de um dado sistema em hardware e que, virtualmente, simula o resto do sistema em questão"(7). Em (7) enfatiza-se que um bom simulador HIL deve manter fluxo de informações bidirecional entre os subsistemas físicos e virtuais envolvidos no experimento.

A substituição de partes físicas de um sistema por partes virtualmente simuladas é ilustrada nas figuras 1.1(a) e 1.1(b) a seguir (9):

A 1.1(a) representa a montagem de um experimento de controle clássico. Neste, há a planta do processo físico estudado, o Computador lógico programável - CLP (*Programable logical computer - PLC*) e o computador utilizado para programação e *download* do código do PLC. Já na figura 1.1(b), observa-se que a planta foi substituída pelo conjunto formado por outro computador e uma placa de entrada-saída para captura de dados. Assim, o computador PC1 simula em tempo real o modelo da planta e a placa de captura faz a interface bidirecional; lendo os resultados da simulação e enviando-os para PLC e lendo as saídas do PLC e colocando-as nas entradas do processo simulado.

É importante ressaltar que não é necessário substituir a planta inteira para caracterizar a simulação *Hardware in the loop*. A tabela 1.1 divide a planta em três partes (atuador, processo e sensor) e mostra cinco possíveis combinações de formatos (real ou simulado) para

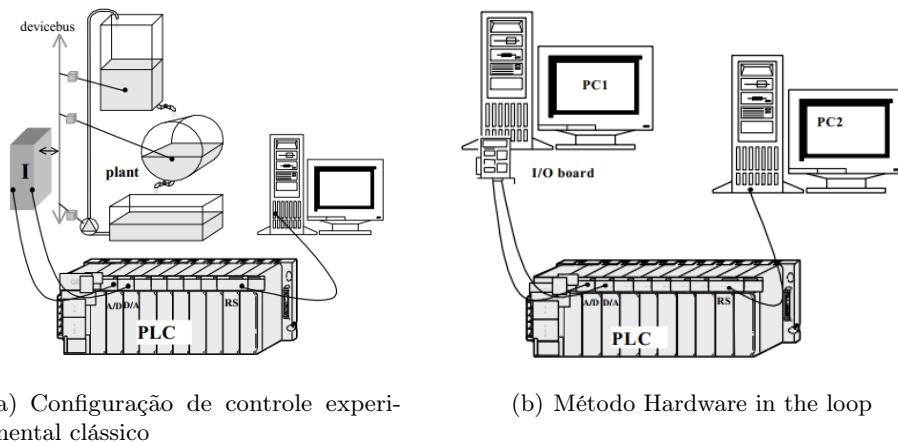


Figura 1.1: Comparativo das configurações de um experimento que não utiliza o método HIL e outro que utiliza o método HIL. Figura retirada de (17)

experimentos HIL (17). Misturas e alternâncias dos casos mostrados são frequentes para testar diferentes aspectos dos componentes envolvidos.

Casos	Formato no experimento		
	Atuadores	Processo	Sensores
1	Real	Simulado	Simulado
2	Real	Simulado	Real
3	Real	Real	Real
4	Simulado	Simulado	Simulado
5	Simulado	Simulado	Real

Tabela 1.1: Componentes reais e simulados da planta para simulação *Hardware in the loop*

As principais vantagens da simulação HIL são listadas abaixo; (17) e(7):

- Redução do custo e tempo de desenvolvimento de produtos;
- Fidelidade aos sistemas reais;
- Repetibilidade dos experimentos;
- Realização em ambiente controlado;
- Segurança para testes relacionados à falhas de componentes ou condições extremas de funcionamento;
- Utilidade para treinamento de pessoal em operação em sistemas arriscados (aeronaves, por exemplo).
- Modificação das dinâmicas dominantes dos processos simulados, permitindo aceleração ou redução do tempo de simulação.

Devido à sua versatilidade, a simulação HIL se tornou indispensável para as indústrias automotiva, aeroespacial, marinha e de defesa por oferecer importante contribuição para o desenvolvimento de sistemas de segurança veicular, sistemas de distribuição de energia, veículos não tripulados, etc. (7). Além disso, as técnicas de simulação HIL se tornaram ferramentas de grande valia na educação e formação de engenheiros e técnicos, como descrito na seção a seguir.

1.1.1 A simulação *Hardware in the loop* na educação

As aulas de laboratório no ensino de engenharia remontam desde os primórdios da educação formal para a profissão, no século XIX (8); incorporando, assim, a praxis como proposta por São Tomás de Aquino. Práticas em laboratório são colocadas nos currículos dos cursos de engenharia e técnicos com o objetivo comum de "relacionar teoria e prática" ou "trazer o 'mundo real' para a sala de aula"(9).

Conclui-se, então, que os avanços das tecnologias de simulação têm sido grandes aliadas às disciplinas práticas. Como processos industriais complexos podem ser muito caros ou ter suas implantações em ambientes educacionais limitadas por questões de espaço físico, a simulação ganhou peso nos currículos das aulas. Desta forma, a técnica de simulação *Hardware in the loop* também é útil de ser aplicada em aulas e cumprir com o objetivo de relacionar, sinergicamente, a teoria com a execução prática dos conceitos teóricos.

Daí surge a proposta deste trabalho: estudar, aplicar e desenvolver técnicas de simulação HIL aplicando ferramentas de hardware de baixo custo e software de simulação personalizado para uso no ensino de controle e automação industrial.

1.1.2 Módulo de Instrumentação, Controle e Automação - sistemas híbridos

O Módulo de Instrumentação, Controle e Automação - sistemas híbridos (MICAsh) é um dispositivo pedagógico desenvolvido para o ensino de controle e automação industrial. O MICAsh conta com um Computador Lógico Programável (CLP), botoeiras de acionamento, sirene, servo-motor, lâmpadas e conectores a serem utilizados em diferentes aulas.

O MICAsh possui um Módulo Computador de Processo (MCP) de arquitetura de baixo custo baseada em um computador numa pastilha (Raspberry PI 3B) com módulo microcontrolador ESP32S que interfacea com hardwares externos via conversores analógico-digital-analógico. O MCP-MICAsh incorpora também circuitos de condicionamento de sinais para adequar e converter sinais de corrente de 4 a 20mA para os níveis de voltagem do ESP32S. A figura 1.3 ilustra a tela do MCP enquanto a figura 1.4 ilustra a arquitetura do MCP-MICAsh.

O Módulo Computador de Processo terá o modo chamado de "simulador de processo", dedicado à simulação *hardware in the loop* de processos. Este trabalho é parte do projeto do MCP, visto que estuda as técnicas de simulação HIL e a possibilidade de realizá-las integrando o Simscape™, da Mathworks, ao Raspberry Pi 3B e ESP 32-S, que se comunicarão com o CLP do MICAsh.

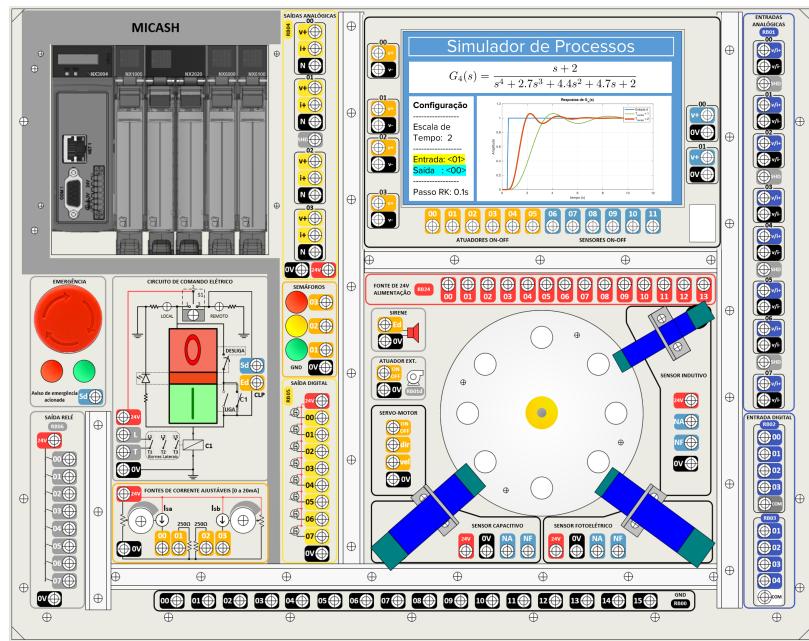


Figura 1.2: Projeto do painel do MICAsh

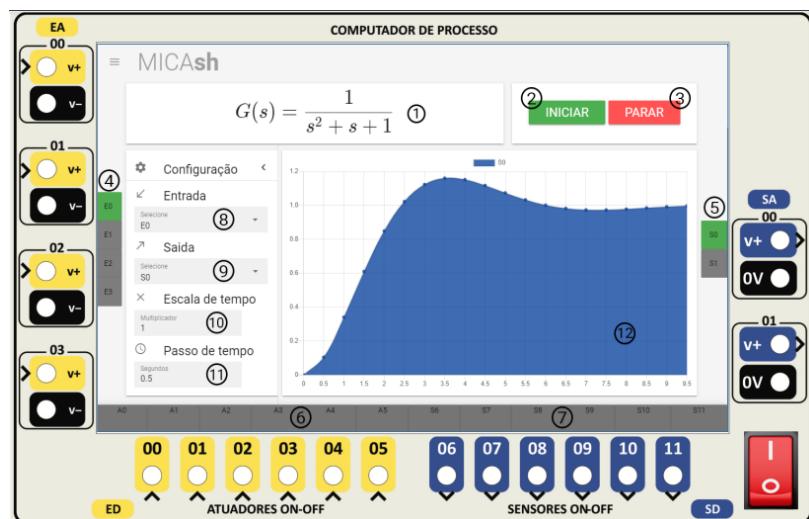


Figura 1.3: Projeto da tela do Módulo Computador de Processos (MCP) do MICAsh

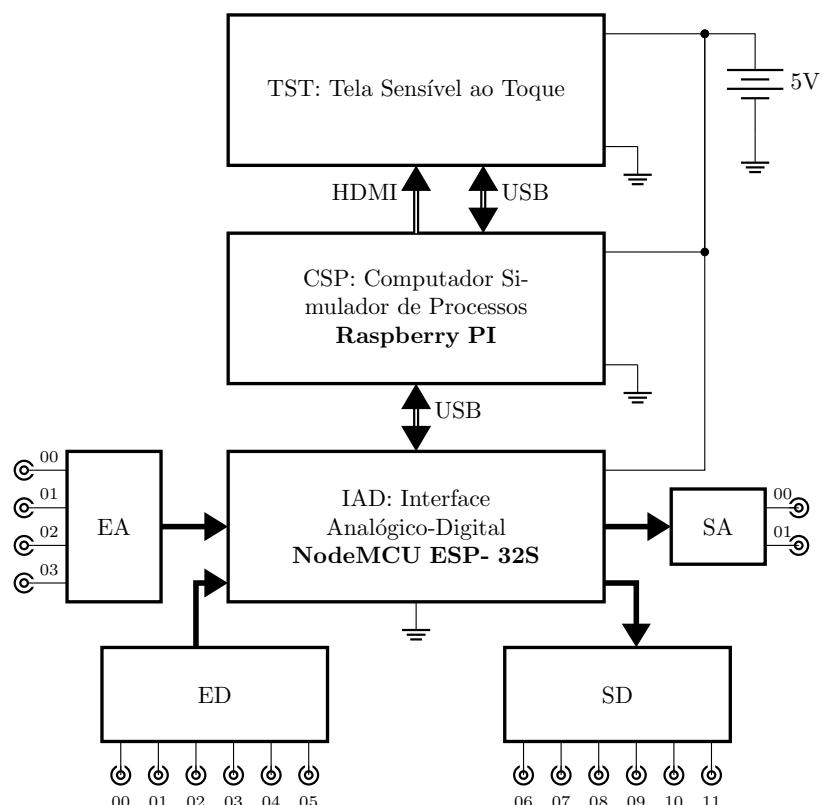


Figura 1.4: Diagrama em blocos da arquitetura do MCP-MICAsh, em que EA inclui os circuitos de condicionamento das Entradas Analógicas, SA das saídas analógicas, ED das Entradas Digitais e SD das Saídas Digitais.

1.2 Escopo do trabalho

O foco deste projeto é desenvolver, estruturar e implementar modelos dinâmicos híbridos de sistemas multifísicos na plataforma do Módulo de Instrumentação, Controle e Automação – sistemas híbridos (MICAsh) (3).

Os modelos multifísicos implementados no MCP-MICAsh são utilizados para aplicações didáticas na formação de técnicos em Eletrônica e Automação Industrial. Modelados fisicamente e simulados digitalmente, estes sistemas estarão ligados a uma interface analógico-digital para possibilitar o fluxo de informações entre a simulação e os componentes físicos em volta dela.

Destaca-se que o interesse deste trabalho está na modelagem física de plantas simples e implementação da simulação digital em tempo real destes modelos em uma plataforma de baixo custo. Estratégias de controle dos sistemas simulados não fazem parte do escopo inicial. Os programas de simulação HIL dos processos do projeto serão desenvolvidos para possibilitar o uso de aplicações externas, desenvolvidas por terceiros (e.g. professores e alunos da universidade).

1.3 Objetivos do projeto

O principal objetivo deste trabalho é o estudo, desenvolvimento e aplicação de técnicas de simulação *hardware in the loop* voltadas para o ensino de controle e automação industrial, analisando a verossimilhança das simulações propostas e suas aplicabilidades em sala de aula.

Além disso, visa explorar o Simscape™, software que será utilizado para criar os modelos das simulações físicas dos sistemas e disponível para a comunidade acadêmica da UFMG por meio das licenças do Matlab™ e ainda subutilizado como ferramenta didática na universidade, e sua serventia no desenvolvimento de experimentos *Hardware in the loop*.

Em suma, os objetivos do projeto são:

- Estudar a modelagem de sistemas multifísicos integrando sistemas elétrico, mecânico, térmico, hidráulico.
- Implementar e simular sistemas multifísicos no Simulink/Simscape
- Simular os modelos Simulink/Simscape em tempo real e avaliar a integração ao módulo MCP do MICAsh.

1.4 Comentários finais

Este capítulo definiu a *Simulação Hardware in the loop (HIL)* como a simulação que integra subsistemas físicos e reais com subsistemas simulados digitalmente e apresentou o Módulo Computador de Processos do MICAsh. Além disso, explicitou a proposta do trabalho em

estudar, aplicar e desenvolver técnicas de simulação HIL de baixo custo e voltadas para o ensino de controle e automação industrial; utilizando para tal o Simscape™ da Mathworks.

Capítulo 2

Descrição das ferramentas utilizadas e suas aplicações

A execução deste projeto e o cumprimento de seus objetivos requerem recursos que permitem a simulação física de sistemas de diferentes princípios, a execução das simulações em tempo real e o envio, condicionamento e interpretação do resultado das simulações. Juntos, esses recursos proporcionarão os experimentos *hardware in the loop* para o módulo MCP-MICAsh.

Este capítulo se divide em duas seções principais: a primeira apresenta quais os recursos e ferramentas utilizadas para o desenvolvimento do projeto, bem como e porque serão empregadas para tal. A segunda seção descreve os processos-exemplo a serem simulados para possibilitar os experimentos HIL e seus princípios físicos.

2.1 Características e uso dos recursos utilizados

2.1.1 MATLAB e Simscape

O MatlabTMé um software de alta perfomance e de cálculo numérico e é amplamente utilizado para análise de gráficos, resolução de sistemas lineares, cálculos matriciais, processamentos de sinais e a criação de gráficos. Constituído de diferentes ferramentas, o MatlabTMpermite a criação de algoritmos para reproduzir ou automatizar o trabalho realizado. (10)

Baseado no MatlabTM, existe o SimscapeTM. Voltado para a modelagem e simulação em tempo reais de sistemas físicos, o Simscape conta com diversas bibliotecas que apresentam componentes de sistemas elétricos, mecânicos, magnéticos, térmicos, pneumáticos, hidráulicos e híbridos. Ademais, possui a funcionalidade de criação de componentes personalizados e parametrização dos modelos utilizados.

Desta forma, o SimscapeTMpermite a simulação de sistemas a partir de diagramas mais próximos da versão física do sistema. Esta característica é um auxílio para aqueles que constróem e utilizam os modelos construídos no SimscapeTM, em detrimento das simulações realizadas a partir de diagramas de blocos de funções de transferência, muito utilizados na engenharia.

O simscape pode ainda ser combinado com a biblioteca de suporte ao Raspberry Pi do SimulinkTMpara possibilitar a transferência dos dados envolvidos na simulação do Matlab

para o MICAsh. Esta combinação será parte primordial deste trabalho, visto que uma das principais características da simulação HIL consiste na troca de dados entre diferentes softwares e hardwares.

Todos os recursos citados nesta seção estão inclusos na licença de uso do Matlab disponibilizada pela UFMG aos seus alunos.

2.1.2 Raspberry PI 3B

Definido pela fabricante como um "computador de placa única", o Raspberry PiTM apresenta todos os principais componentes de um computador pessoal comum. De preço acessível, é ideal para pequenos projetos e construção de protótipos e, por isso, é amplamente utilizado para ensino, pesquisa e desenvolvimento de projetos

O Raspberry Pi utilizado para o desenvolvimento deste trabalho é do modelo 3B que tem as seguintes especificações([Pi](#)):

- Processador Quad Core 1.2GHz Broadcom BCM2837 de 64 bits;
- Memória RAM de 1GB;
- Módulo wireless, Lan e Bluetooth BCM43438 integrado;
- Ethernet 100 Base;
- Módulo de entrada e saída de uso geral (General Purpose Input/Output - GPIO) de 40 pinos;
- Entrada HDMI;
- Quatro entradas USB 2.0;
- Entrada de câmera CSI: permite conectar uma câmera Raspberry PiTM;
- Entrada de display DSI: permite conectar um display sensível ao toque Raspberry PiTM;
- Slot para micro SD

Neste projeto, o Raspberry PiTM receberá os resultados das simulações executadas no Simscape/Simulink e, junto com o ESP 32-S fará a integração com o CLP do MICAsh.

2.1.3 ESP 32-S

Os NodeMCU ESP32-STMsão microcontroladores de baixo custo e consumo de energia integrados a módulos de comunicação Wi-fi e/ou bluetooth, a depender do modelo. Neste projeto, o ESP 32-S será a interface da parte simulada dos experimentos *hardware in the loop*. Com seus canais de Conversão Analógico-Digital (*Analogue to Digital Converter*, ADC) e Conversão Digital-Analógico (*Digital to Analogue Converter*, DAC), será utilizado para a conversão dos sinais vindos da simulação dos processos para a leitura do CLP e vice-versa.

O ESP32-S, então, será utilizado para a interface entre as partes reais e simuladas do experimento HIL. Para a parte real, o MICAsh, será necessário configurar os endereços de entrada e saída (GPIO) do módulo de forma que a integração e a conversão dos sinais via DAC/ADC sejam feitas corretamente. Para as simulações executadas no Simscape/MatlabTM, será necessário providenciar um canal de comunicação exato e eficiente para garantir que as entradas e saídas do processo sejam traduzidas corretamente e em tempo real para os componentes reais envolvidos no experimento. O ESP32-S possibilitará esta troca de informações via canais de comunicação serial.

Desta forma, será parte essencial na interface bidirecional entre os componentes da simulação HIL.

A tabela 2.1(3) apresenta as configurações de endereços de entrada e saída do NodeMCU32-S para o módulo Computador de Processos do MICAsh. Trata-se de uma configuração realizada fora do escopo deste trabalho, mas que deve ser respeitada para que o projeto seja integrado ao MICAsh.

Na tabela 2.1, a sigla SD significa saída digital e SA significa saída analógica. De maneira semelhante, ED e EA significam entrada digital e entrada analógica, respectivamente. Os pinos sem função explícita não são utilizados no módulo Computador de Processos do MICAsh.

Pino	IO-ESP	Função	Função	IO-ESP	Pino
1	[x][06][CLK]			5V input power	38
2	[x][07][SD0]			[CMD][D11][x]	37
3	[x][08][SD1]			[SD3][D10][x]	36
4	[15][ADC13(T3)]	SD06		[SD2][D09][x]	35
5	[02][ADC12(T2)]	SD07		ED00 [ADC14(T4)][13]	34
6	[00][ADC11(T1)]	SD0		GND	33
7	[04][ADC10(T0)]	SD09		ED01 [ADC15(T5)][12]	32
8	[16]	SD10		ED02 [ADC16(T6)][14]	31
9	[17]	SD11		ED03 [ADC17(T7)][27]	30
10	[x][05][SPI]			SA00 [DAC1][26]	29
11	[x][18][SPI]			SA01 [DAC0][25]	28
12	[x][19][SPI]			ED04 [ADC04(T8)][33]	27
13	GND			ED05 [ADC04(T9)][32]	26
14	[21]			EA00 [ADC07][35]	25
15	[x][03][RX]			EA01 [ADC06][34]	24
16	[x][01][TX]			EA02 [ADC03][39]	23
17	[22]			EA03 [ADC00][36]	22
18	[x][23][SPI]			EN[RESET]	21
19	GND			3.3V	20

Tabela 2.1: Pinagem do módulo NodeMCU32-S e as funções atribuídas aos pinos no módulo Computador de Processo do MICAsh. Tabela retirada de (3)

2.2 Características dos sistemas-exemplo modelados

Esta seção descreve os sistemas-exemplo a serem modelados e simulados para a realização deste projeto. Demonstra-se nas subseções abaixo os princípios físicos que regem os sistemas e qualquer consideração e/ou simplificação.

2.2.1 Tanques cilíndricos desacoplados

O primeiro processo cujo modelo será desenvolvido no Simscape consiste em dois tanques cilíndricos; cada um com um canal para entrada de água e outro para a saída. A vazão de entrada do tanque 1 é determinada pela rotação (velocidade angular) de uma bomba d'água. A vazão de saída deste mesmo tanque obedece à lei de Bernoulli e é dependente da altura da coluna de água presente no tanque (15).

O segundo tanque opera de maneira semelhante, com o diferencial de que seu canal de entrada é conectado ao canal de saída do tanque 1. Assim, a vazão de saída do tanque 1

q_{out1} é igual à vazão de entrada do tanque 2 q_{in2} :

$$q_{out1} = q_{in2} \quad (2.1)$$

A figura 2.1 ilustra a configuração dos tanques. Eles são ditos desacoplados por que o tanque 2 não interfere no tanque 1, de forma que a vazão de saída do tanque 1 só dependa da altura de coluna d'água presente no respectivo tanque (1).

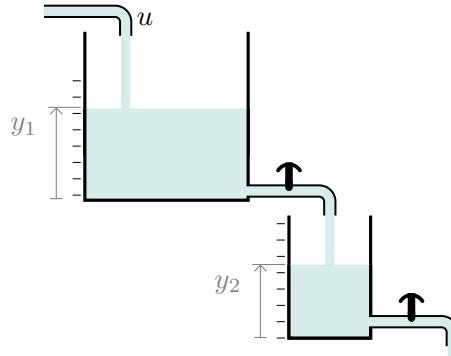


Figura 2.1: Esquema de tanques desacoplados

Para a modelagem física do processo, existem as seguintes grandezas:

- y_1 e y_2 representam as alturas das colunas d'água dos tanques 1 e 2, respectivamente;
- a_1 e a_2 representam as áreas de seção transversal dos canais de saída dos tanques 1 e 2, respectivamente;
- A_1 e A_2 representam as áreas de seção transversal dos cilindros dos tanques 1 e 2, respectivamente;
- q_{in1} e q_{in2} representam as vazões de entrada dos tanques 1 e 2, respectivamente e
- q_{out1} e q_{out2} representam as vazões de saída dos tanques 1 e 2, respectivamente.

Considerando que haverá balanceamento de massa, a variação infinitesimal de y_1 no tempo é proporcional a vazão resultante (15):

$$A_1 \frac{dy_1}{dt} = -q_{out1} + q_{in1} \quad (2.2)$$

A vazão de saída q_{out1} , pela lei de Bernoulli, é proporcional ao nível y_1 de água:

$$q_{out1} = a_1 \sqrt{2gy_1} \quad (2.3)$$

onde g é a aceleração da gravidade.

As variações do nível e a vazão de saída variam de forma análoga para o tanque 2:

$$A_2 \frac{dy_2}{dt} = -q_{out2} + q_{in2} \quad (2.4)$$

$$q_{out2} = a_2 \sqrt{2gy_2} \quad (2.5)$$

Já que $q_{out1} = q_{in2}$ (equação 2.1), pode-se substituir 2.3 e 2.5 em 2.4:

$$A_2 \frac{dy_2}{dt} = -a_2 \sqrt{2gy_2} + a_1 \sqrt{2gy_1} \quad (2.6)$$

O volume de água dentro dos tanques responde à equação:

$$V_i = y_i A_i \quad (2.7)$$

2.2.2 Filtro passa-baixas

Filtros passa-baixas são componentes importantes na eletrônica, visto que vários dos sinal capturados por transdutores ou sensores podem ter componentes oscilatórias de alta frequência, dificultando a leitura e processamento do sinal para seu uso final.

Um filtro passa-baixas pode ser implementado com resistores e capacitores, tornando-os circuitos simples. Devido à sua simplicidade e sua larga utilização, o filtro RC de dupla seção é o segundo processo a ser simulado no Simscape™.

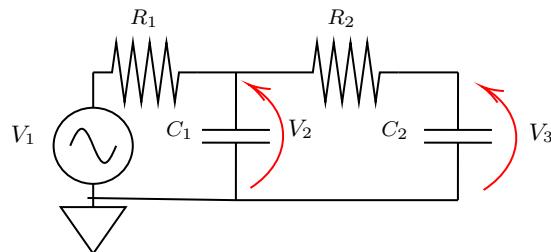


Figura 2.2: Circuito do filtro passa-baixas duplo

A figura 2.2 ilustra o circuito do filtro duplo, apresentando suas duas malhas. A segunda malha, ou seção, formada por R_2 e C_2 está acoplada a primeira, R_1 e C_1 , o que produz um efeito de carga. Assim, a tensão de saída V_3 depende da tensão de saída V_1 (4).

Aplicando-se a Lei de Kirchhoff para o nó entre o resistor R_1 , o resistor R_2 e o capacitor C_1 , obtém-se:

$$I_1 + I_2 + I_3 = 0 \quad (2.8)$$

A partir da Lei de Ohm, obtém-se as seguintes equações para as correntes I_1 , I_2 e I_3 :

$$\begin{cases} V_1 = \frac{V_2 - V_1}{R_1} \\ V_2 = s \cdot C_2 \cdot V_3 \\ V_3 = s \cdot C_1 \cdot V_2 \end{cases} \quad (2.9)$$

Substituindo as equações acima na equação 2.8 e remanejando os fatores, encontra-se V_1 em função de V_2 e V_3 :

$$\frac{V_2 - V_1}{R_1} + sC_2V_3 + sC_1V_2 = 0 \quad (2.10)$$

$$sR_1C_2V_3 + (sR_1C_1 + 1)V_2 = V_1 \quad (2.11)$$

A segunda malha é um filtro RC passa-baixa de seção simples, logo:

$$\frac{V_2 - V_3}{R_2} = sC_2V_3 \quad (2.12)$$

Manipulando a equação 2.12:

$$V_2 = (sR_2C_2 + 1)V_3 \quad (2.13)$$

Para obter a relação de V_3 em função de V_1 , substitui-se a equação 2.13 em 2.11:

$$sR_1C_2V_3 + (sR_1C_1 + 1)(sR_2C_2 + 1)V_3 = V_1 \quad (2.14)$$

$$\frac{V_3}{V_1} = \frac{1}{sR_1C_2V_3 + (sR_1C_1 + 1)(sR_2C_2 + 1)} \quad (2.15)$$

Fixando as seguintes constantes de tempo, obtém-se a função de transferência da equação 2.17:

$$\begin{cases} \tau_1 = R_1C_1 \\ \tau_2 = R_2C_2 \\ \tau_{12} = R_1C_2 \end{cases} \quad (2.16)$$

$$\frac{V_3}{V_1} = \frac{1}{\tau_1\tau_2 s^2 + (\tau_1 + \tau_2 + \tau_{12})s + 1} \quad (2.17)$$

Assumindo que $R_2 = 10R_1$ e $C_2 = 0.1C_1$ de forma que $\tau = \tau_1 = \tau_2 = R_1C_1$ e $\tau_{12} = 0.1\tau$, pode-se minimizar o efeito de carga da segunda malha, reescreve-se a função de transferência 2.17 (4):

$$\frac{V_3}{V_1} \approx \frac{1}{\tau^2 s^2 + 2\tau s + 1} \approx \frac{1}{(\tau s + 1)^2} \quad (2.18)$$

A nova função de transferência da equação 2.18 representa um filtro passa-baixas de segunda ordem com frequência de corte única, $\omega_c = \frac{1}{\tau}$. Desta forma, o filtro deve se comportar como um sistema de segunda ordem superamortecido para variações em degrau.

2.2.3 Termopar

O termopar é um transdutor de temperatura em tensão que consiste em dois fios de metais diferentes unidos em uma das pontas, como mostrado na figura 2.3. Se a temperatura T_1 é diferente da temperatura T_2 , as pontas livres (também chamadas de pontas de referência) apresentarão diferença de tensão entre si. Assim, ao posicionar a junção em um ambiente com temperatura desconhecida e as pontas de referência em local com temperatura ambiente ou conhecida, pode-se medir T_2 (6) (18).

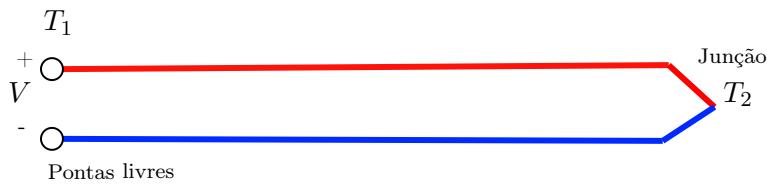


Figura 2.3: Esquema de um termopar

A equação 2.19 mostra a relação entre as temperaturas e tensão gerada entre as pontas livres. Nela, ddp representa a diferença de potencial elétrico, S_1 e S_2 são os coeficientes de Seebeck dos dois materiais dos quais o termopar é composto e T_1 e T_2 são as diferentes temperaturas.

$$ddp = \int_{T_1}^{T_2} (S_1 - S_2).dT \quad (2.19)$$

A partir da equação 2.19, pode-se concluir que a diferença de potencial é nula se os dois materiais ou as duas temperaturas são iguais. Ressalta-se também que a temperatura do ambiente onde se encontram as pontas livres deve ser conhecida para que seja possível determinar a temperatura da junção.

Enquanto é possível produzir vários termopares diferentes com as distintas combinações entre materiais metálicos, existem alguns pares conhecidos por exibirem propriedades que os fazem mais ou menos lineares para diferentes faixas de temperatura e, consequentemente, melhor aplicáveis para certas situações (6). Estes pares recebem nomes para melhor identificação e mais informações sobre eles podem ser encontrados na tabela abaixo:

Tipo de termopar	Faixa de utilização (°C)
B	0 a 1820
E	-270 a 1000
J	-210 a 1200
K	-270 a 1372
N	-270 a 1300
R	-50 a 1768.1
S	-50 a 1768.1
T	-270 a 400

Tabela 2.2: Tipos comerciais de termopares e suas faixas de temperatura de utilização (14)

Para estes pares comerciais de termopares, pode-se determinar a temperatura da junção com as equações 2.20 e 2.21 abaixo. A equação 2.20 pode ser utilizada para os termopares do tipo B, E, J, N, R, S, T para todas as faixas de utilização e para o termopar do tipo K nos casos em que a temperatura da junção é menor que 0 grau Celsius(0°C) (14).

$$ddp = \sum_{i=0}^n c_i \cdot T^i \quad (2.20)$$

A equação 2.21 é utilizada para os termopares do tipo K nos casos em que a temperatura da junção é maior ou igual a 0 grau Celsius(0°C).

$$ddp = \sum_{i=0}^i c_i \cdot T^i + a_0 e^{a_1(T-a_2)^2} \quad (2.21)$$

Para as duas equações acima, ddp é a diferença de potencial elétrico em mV , T é a temperatura da junção e c_i é o i -ésimo elemento da tabela de coeficientes do termopar. Esta tabela pode ser encontrada no portal do *National Institute of Standards and Technology (NIST)*, o Instituto Nacional de Padrões e Tecnologia do departamento de comércios dos E.U.A. (14).

No caso da equação 2.21, a_0 , a_1 e a_2 também são valores que podem ser verificados no portal citado acima.

2.3 Comentários finais

Este capítulo descreveu as ferramentas e os recursos utilizados para a execução deste projeto, definindo suas principais partes e suas utilizações: Simscape para simulação física de diferentes processos e Raspberry Pi e ESP32-S para integrar o Simscape com o MICAsh-MCP. Além disso, apresentou os processos tomados como exemplo para a avaliação e experimentação prática da possibilidade da incorporação de experimentos *Hardware in the loop* no MICAsh-MCP.

Capítulo 3

Descrição da metodologia e métodos

Este capítulo apresenta a lógica e o funcionamento dos métodos (funções, códigos-fonte e diagramas de simulação) criados para possibilitar os experimento HIL no MICAsh, além de explicar como configurá-los e utilizá-los.

É dividido em três seções: a primeira apresenta os recursos desenvolvidos para a comunicação de dados entre Simscape e ESP32-S. A segunda seção apresenta os diagramas construídos para as simulações dos processos-exemplo apresentados no capítulo anterior. Por fim, a terceira seção apresenta as configurações necessárias para a utilização destes recursos em experimentos similares em projetos e aulas.

3.1 Funções e blocos utilizados na comunicação

Para a comunicação serial com o conjunto Raspberry pi/Simscape, foi necessário estabelecer um protocolo de comunicação entre ele e ESP32-S que pudesse transmitir todos os valores do conjunto real de números para possibilitar o uso do MICAsh-MCP em diversos experimentos *Hardware in the loop* com as simulações de diferentes tipos de processos. Junto a este protocolo, funções foram criadas nos códigos do ESP32 e do Simscape para converter as mensagens do protocolo de envio para o formato decimal.

No código-fonte executado pelo ESP32-S, duas funções foram criadas para o envio e recepção das mensagens seriais, além de duas outras funções auxiliares.

No simscape, foram criados dois blocos de função responsáveis pela comunicação serial. Um dedicado para a leitura da entrada do processo, o sinal enviado pelo CLP para o ESP32-S e deste para o SimscapeTM, e outro responsável pelo envio da saída do processo; calculada pelo próprio SimscapeTM.

As duas subseções abaixo descrevem as funções e blocos utilizados para o envio da entrada do processo ao Simscape e aqueles utilizados para o envio da saída da simulação ao ESP32, respectivamente.

3.1.1 Funções e blocos utilizados para o envio da entrada do processo

A entrada do processo deve ser calculada pelo MICAsh, transmitida ao ESP32-S e, depois, transmitida ao conjunto Raspberry/Simscape. A principal função do código-fonte do ESP32-S para a leitura da entrada no valor analógico é chamada de "leituraEntrada". Em primeiro lugar, esta função lê a entrada do canal analógico do microcontrolador e filtra seu valor, utilizando para isso a equação de um filtro de 1^a ordem sem atraso. Este filtro previne o falseamento do sinal (ou *aliasing*) causado pelo ruído de medição com componentes de frequências maiores que a frequência de Nyquist ω_n (5)

O valor de saída do filtro, então, deve ser convertido para um formato que possibilita a transmissão para o conjunto Raspberry Pi/Simscape via canal serial. Esta codificação ocorre em duas etapas. A primeira consiste em utilizar a função auxiliar "Map" para transformar o valor da ação de controle da escala do processo para a escala definida para o envio. A equação 3.1 mostra a função executada, retirada de (2) :

$$Y_{atual} = \frac{(X_{atual} - X_{min}) \cdot (Y_{max} - Y_{min})}{X_{max} - X_{min}} + Y_{min} \quad (3.1)$$

onde X_{atual} equivale ao valor atual da ação de controle na escala de entrada da função, X_{min} e X_{max} equivalem, respectivamente, aos valores mínimo e máximo possíveis para a escala de entrada. De maneira análoga, Y_{atual} equivale ao valor presente da ação de controle, mas desta vez na escala de saída da função. Y_{min} e Y_{max} equivalem aos valores mínimos e máximos para a escala de saída. Assim, se $X_{atual} = X_{min}$, $Y_{atual} = Y_{min}$. De forma similar, se $X_{atual} = X_{max}$, Y_{atual} será igual a $= Y_{max}$. Para todo valor de X_{atual} entre X_{min} e X_{max} , Y_{atual} terá valor proporcional entre Y_{min} e Y_{max} , respeitando a equação 3.1.

Neste caso, a escala de entrada equivale ao conjunto de valores possíveis da entrada do processo enviada pelo PLC, que pode variar de acordo com o sistema simulado. A escala de saída é o conjunto de valores definidos para o protocolo de envio, sendo $Y_{min} = 0$ e $Y_{max} = 4095$.

As vantagens de utilizar esta função para mapear os números a serem enviados pelo ESP32-S está na universalidade de uso, pois funciona com quaisquer pares de escala; até mesmo aquelas que contém números negativos ou com valores absolutos elevados. Assim, é garantida a compatibilidade do módulo de experimentos HIL do MICAsh com diversos processos diferentes e não apenas os descritos no presente texto. A biblioteca de funções matemáticas para o Arduino tem sua própria função "Map" nativa, mas esta só funciona para números inteiros (2). Para garantir o funcionamento também com números racionais, a função "Map" utilizada foi declarada como uma função nova e que utiliza números do tipo *float*.

Depois da conversão de escala do número a ser enviado, a função "leituraEntrada" separa o número em quatro números diferentes. Um que representa o milhar, outro que representa a centena, o terceiro para a centena e o último para a unidade. Estes números são enviados individualmente pelo canal serial no fim da execução de "leituraEntrada".

Abaixo, verifica-se os códigos-fonte das funções "leituraEntrada" e "Map":

```

1 void leituraEntrada(){ // Le a saida do plc , a entrada do processo
    simulado .
2     yk = analogRead(AN_Pot1); // Le a entrada ADC
3     yf = yf + alfa_f*(yk-yf); // filtro digital 1a. ordem sem atraso de
    amostragem
4
5     if (mCtr % m ==0){ // amostra a vari vel filtrada decimada por m.
6         ym = yf;
7     }
8
9     // Codifica o para o formato necessario para envio ao simulink
10    In_processo = ((float)ym * in_maxIN)/4095.0;
11    In_processo = Map(In_processo , in_minIN , in_maxIN,in_minOUT,in_maxOUT)
    ;
12
13    int i = In_processo/1000;
14    outcomingMilhar = (uint8_t)i;
15    In_processo = In_processo - (float)i*1000;
16    i = In_processo/100;
17    outcomingCentena = (uint8_t)i;
18    In_processo = In_processo - (float)i*100;
19    i = In_processo/10;
20    outcomingDezena = (uint8_t)i;
21    In_processo = In_processo - (float)i*10;
22    outcomingUnidade = (uint8_t)In_processo;
23
24    // Envio das mensagens ao simulink
25    MySerial.write(outcomingMilhar);
26    MySerial.write(outcomingCentena);
27    MySerial.write(outcomingDezena);
28    MySerial.write(outcomingUnidade);
29 }
```

Programa 3.1: Função LeituraEntrada

```

1 float Map ( int x, int in_min , int in_max , int out_min , int out_max){ // 
    mapeia um numero de um range(escala) a outro .
2     return ((float)x - in_min)*(out_max - out_min)/(in_max - in_min) +
    out_min;
3 }
```

Programa 3.2: Função Map

No SimscapeTM, criou-se um bloco personalizado responsável por receber a mensagem serial enviada pelo ESP32-S, decodificá-la e convertê-la para a escala necessária para a entrada do processo simulado. Este bloco representa um diagrama construído especialmente para este trabalho e que posteriormente foi generalizado e transformado em um bloco do Simulink para facilitar seu uso com simulações diferentes. As figuras 3.1 e 3.2 mostram a máscara personalizada para o bloco, para facilitar sua identificação, e o diagrama executado por ele, respectivamente.

Como pode ser visto na figura 3.2, o diagrama do bloco de recepção de mensagens serial enviadas pelo ESP32-S pode ser dividido em duas partes. A primeira parte é o bloco *Serial*

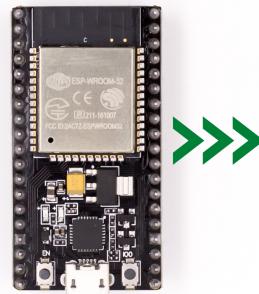


Figura 3.1: Ícone personalizado do subsistema criado para ler e decodificar a mensagem enviada pelo ESP32-S pelo canal serial do Raspberry Pi.

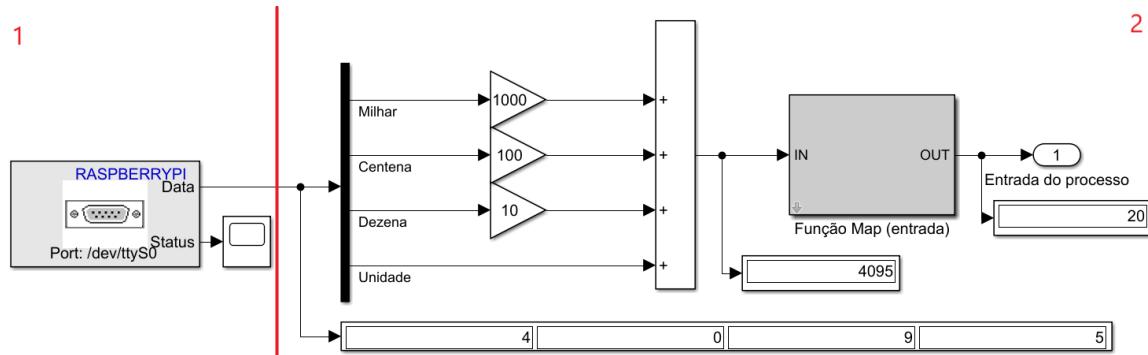


Figura 3.2: Diagrama do subsistema criado para ler e decodificar a mensagem enviada pelo ESP32-S pelo canal serial do Raspberry Pi.

Read da biblioteca de suporte ao Raspberry Pi do Simulink™. É este bloco que lê as mensagens que chegam no canal serial do Raspberry Pi. Ele recebe como principais parâmetros o modelo do Raspberry Pi utilizado, o endereço do canal serial utilizado, a taxa de transmissão (ou *baudrate*) em bits por segundo, o tamanho e o formato dos dados a serem recebidos. Este bloco conta com duas saídas: a primeira disponibiliza os dados lidos da mensagem recebida, a segunda mostra a situação (*status*) dos dados lidos. O valor da saída de *status* varia de acordo com as condições da tabela 3.1, retirada de (13):

Valor da saída Status	Descrição
0	Sucesso na transmissão
1	RX do canal serial ocupado
2	TX do canal serial ocupado
4	Erro de paridade
8	Erro de janela
16	Erro no barramento
32	Dado não disponível

Tabela 3.1: Possíveis valores da saída *Status* do bloco *Serial Read* e suas descrições.

A segunda parte do diagrama consiste na decodificação da mensagem. Como o ESP32-S envia quatro números ao Simscape, é necessário, ler estes números e colocá-los nas respectivas ordem de grandeza. Assim, o bloco *Serial Read* lê 4 sinais diferentes e é utilizado um demultiplexador para separá-los. O primeiro dos sinais representa a casa dos milhares do número enviado, o segundo representa a centena, o terceiro a dezena e, por fim, o quarto e último representa a unidade deste número. Os valores dos ganhos desta seção não devem ser alterados.

Depois, o número é composto com o somador e passa pelo bloco que é equivalente a função "Map", descrita anteriormente e representada pela equação 3.1, para que a ação de controle possa retornar à escala requerida pelo processo antes de ser aplicado à sua entrada. O diagrama da função "Map" foi construído utilizando blocos de funções matemáticas do Simulink e pode ser visto na figura 3.3:

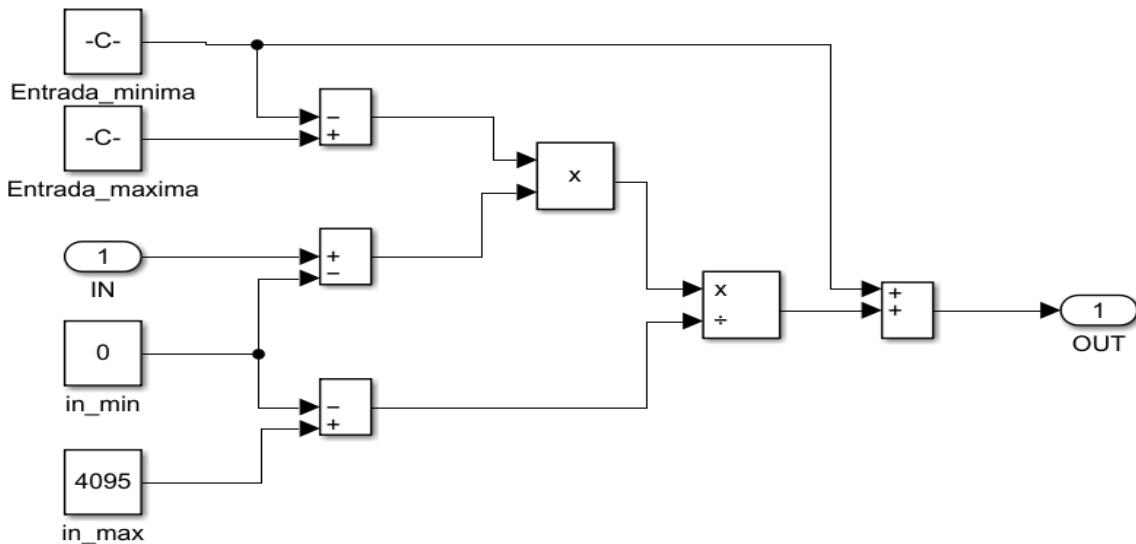


Figura 3.3: Diagrama da função MAP no ambiente Simulink do Matlab.

Existem três displays disponíveis no bloco que podem ser utilizados para acompanhar a recepção e processamento da mensagem enviada pelo ESP32-S. Vale ressaltar que o bloco de recepção de mensagens vindas do ESP32-S foi generalizado para facilitar seu uso. Desta forma, foi configurado para que os principais parâmetros possam ser editados direto do ícone da figura 3.1, sem que seja necessário alterar individualmente no diagrama. Assim, basta clicar duas vezes no bloco para abrir a janela de configuração.

3.1.2 Funções e blocos utilizados para o envio da saída do processo

Uma vez que os processos tem sua saída calculada, é necessário codificar e enviar este valor para o ESP32-S e, posteriormente, ao MICAsh. Para isso, utiliza-se o bloco de envio de mensagens serial das figuras 3.4 e 3.5.

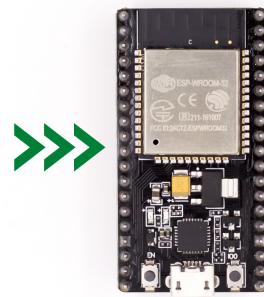


Figura 3.4: Ícone personalizado do subsistema criado para codificar e enviar a saída da simulação pelo canal serial do Raspberry Pi.

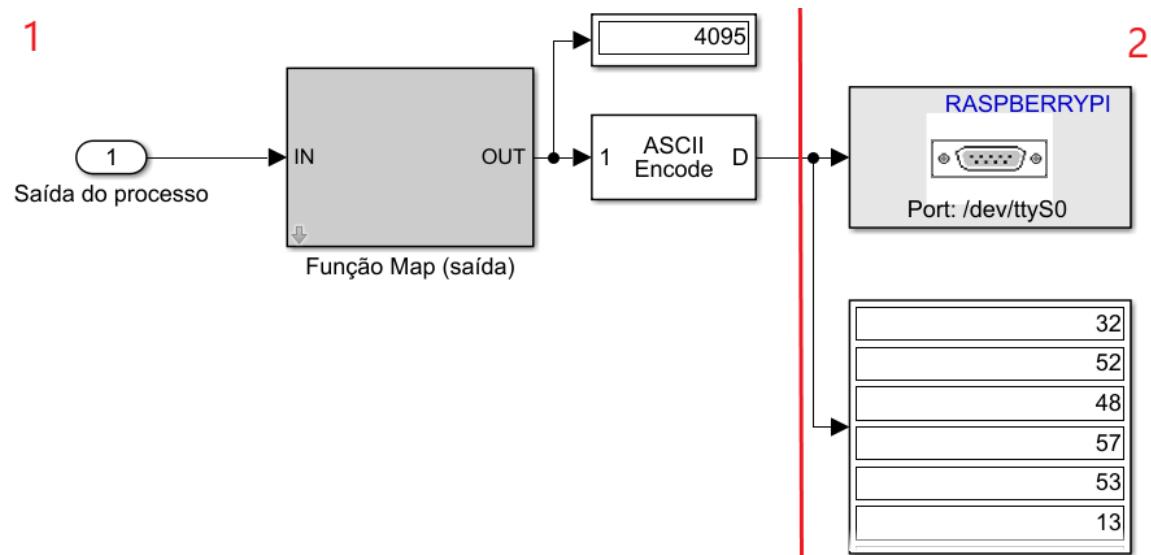


Figura 3.5: Diagrama do subsistema criado para codificar e enviar a saída da simulação pelo canal serial do Raspberry Pi.

Caractere	Código ASCII
Espaço	32 (marca o início da sequência)
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
quebra de linha	13 (marca o fim da sequência)

Tabela 3.2: Código ASCII dos caracteres possíveis na sequência de saída do bloco *ASCII encode* do bloco de envio de mensagens para o ESP32-S

O diagrama da figura 3.5 está dividido em duas partes. A primeira é responsável pela codificação da saída do processo para que possa depois ser enviada. Assim, a saída é transformada pela função Map, deixando-a na escala necessária para o envio e, em seguida, é codificada em ASCII. O bloco *ASCII Encode* do Simulink converte cada dígito do número em uma sequência de códigos ASCII, mais um código que sinaliza o início e outro que sinaliza o fim do número. A observação da figura 3.5, que ilustra a codificação do número 4095 em ASCII em uma sequência de seis dígitos, e da tabela 3.2 permitem enxergar os códigos de início da mensagem, conteúdo, e código de fim.

Vale ressaltar que apenas caracteres numéricos, o espaço e a quebra de linha podem aparecer no bloco de envio de mensagens para o ESP32-S via comunicação serial, visto que a codificação da saída é sempre feita com números inteiros e positivos. Caso o valor da saída seja negativo, ela ainda será lida corretamente devido a junção da função Map e do bloco *ASCII encode*.

A segunda seção do diagrama da figura 3.5 consiste no bloco *Serial Write* da biblioteca de suporte ao Raspberry Pi do Simulink™. É este bloco que envia a mensagem para o ESP32-S, enviando do canal serial do Raspberry Pi configurado. Os principais parâmetros que devem ser configurados para este bloco são: o modelo de placa Raspberry Pi utilizada, o endereço do canal serial do Raspberry Pi utilizado e a taxa de transmissão (*baudrate*) em bits por segundo.

Existem dois displays no bloco de envio de mensagem para o ESP32-S para possibilitar o acompanhamento da codificação. Os parâmetros deste subsistema também devem ser configurados ao clicar duas vezes sobre o ícone da imagem 3.1

Uma vez enviada a mensagem, cabe ao ESP32-S ler e interpretar. A função "SerialEvent" do código escrito para o ESP32-S é a principal responsável pela leitura da saída. Executada a cada iteração do loop, a função "SerialEvent" detecta se há uma nova mensagem disponível no canal Serial e a lê como um número inteiro e positivo. A depender destes números, diferentes rotinas podem ser executadas. Se o número lido for 32, o código ASCII do espaço, a variável booleana "stringRxStart" é alterada para verdadeira. De maneira análoga, se o número

recebido for 13, a quebra de linha, a variável "stringRxComplete" é alterada para verdadeira, enquanto "stringRxStart" é alterada para falsa. Assim, detecta-se quando uma nova sequência de caracteres que representam um único valor de saída começou, terminou ou está no meio de sua transmissão.

Qualquer número recebido pelo canal serial enquanto "stringRxStart" é verdadeiro e "stringRxComplete" é falsa faz parte do número que representa a saída e é convertido para o caractere com a função auxiliar "int2ascii". Depois da conversão completa da mensagem, basta utilizar a função "Map" para ajustá-la à escala de valores reais da saída do processo simulado. Depois, a saída pode ser enviada para o CLP do MICAsh.

```

1 void serialEvent() {
2     if (MySerial.available() > 0) { // Verifica se tem algo a ser lido na
      porta serial
3     // Obtem um novo byte:
4     uint8_t inChar = MySerial.read();
5
6     if (stringRxStart == true){ // Verifica se a string ja comecou
7         if (inChar == 13) { // se o caractere for um newline (\n),
      identifica que a string terminou.
8         stringRxComplete = true;
9         stringRxStart = false;
10    }
11    else {
12        incomingValue = incomingValue*10 + int2ascii(inChar); // se o
      caracter NAO for um newline (\n), le o caractere que chega e compoe a
      mensagem.
13    }
14  }
15  if (inChar == 32){ // verifica se a nova string come ou. Detecta o
      espaço e seta a flag.
16    stringRxStart = true;
17    incomingValue = 0;
18  }
19 }
20 }
```

Programa 3.3: Função Serial Event

```

1 int int2ascii (uint8_t inChar){ // funcao utilizada para decifrar os
      caracteres recebidos via porta Serial vindos do simulink.
2     switch (inChar){           // converte o codigo ASCII ao caracter.
3         case 48:
4             return 0;
5         case 49:
6             return 1;
7         case 50:
8             return 2;
9         case 51:
10            return 3;
11        case 52:
12            return 4;
```

```

13     case 53:
14         return 5;
15     case 54:
16         return 6;
17     case 55:
18         return 7;
19     case 56:
20         return 8;
21     case 57:
22         return 9;
23 }
24 }
```

Programa 3.4: Função int2ascii

O bloco do Simulink de envio de saída conta com uma variação que permite o envio de duas saídas diferentes ao ESP32-S. A figura 3.6 mostra o diagrama desta variação. O princípio de funcionamento é o mesmo que o anterior, mas duplicado. Desta forma, é necessário dois blocos de função Map e dois blocos *ASCII Encode*. A principal diferença dos blocos está na presença dos multiplexadores e nas constantes de valores iguais a 65 e 66. Os multiplexadores juntam os números em uma mensagem só e as constantes servem para rotular as saídas, pois é necessário que o ESP32-S identifique a qual saída corresponde o valor que ele lê. Desta forma, a mensagem que começa com 65 diz respeito a saída A enquanto a que se inicia com 66 se refere a saída B.

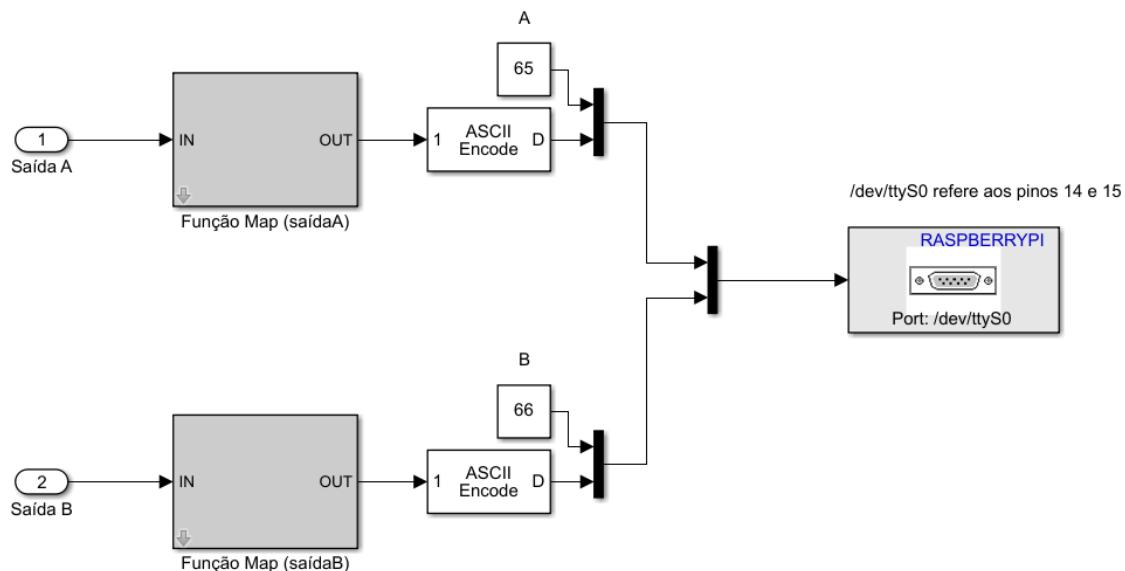


Figura 3.6: Diagrama do subsistema criado para codificar e enviar duas saídas da simulação pelo canal serial do Raspberry Pi.

Neste bloco, deve-se configurar os mesmos parâmetros da versão de valor único, além dos valores máximo e mínimo da saída B. A função "SerialEvent" do código-fonte do ESP32-S também foi alterada para comportar a transmissão de dois valores diferentes. Agora, antes de procurar pelo espaço (código 32), a função procura os códigos 65 ou 66 para identificar a qual saída (A ou B) se refere aquele novo valor.

Ressalta-se que os códigos de identificação das saídas podiam ser qualquer número diferente dos códigos da tabela 3.2, que já são utilizados para outros propósitos. Entretanto, optou-se por utilizar 65 e 66 pois representam os caracteres "A"e "B"na tabela ASCII, coincidindo assim com os nomes "Saída A"e "Saída B".

A função "SerialEvent"alterada para ler duas saídas pode ser vista abaixo:

```

1 void serialEvent() {
2     if (MySerial.available()> 0) { // Verifica se tem algo a ser lido na
      porta serial
3     // Obtem um novo byte:
4     uint8_t inChar = MySerial.read();
5
6     if(inChar == 65){ // Verifica se a saida a qual a parte da msg atual
      corresponde e a saida A e seta a flag
7         VarIndex = false;
8     }
9     if(inChar == 66){ // Verifica se a saida a qual a parte da msg atual
      corresponde e a saida B e seta a flag
10        VarIndex = true;
11    }
12    if (VarIndex == false){ // decodifica saida A
13        if (stringRxStartA == true){ // Verifica se a string ja comecou
14            if (inChar == 13) { // se o caracter for um newline (\n),
      identifica que a string terminou.
15                stringRxCompleteA = true;
16                stringRxStartA = false;
17            }
18            else {
19                incomingValueA = incomingValueA*10 + int2ascii(inChar); // se
      o caracter NAO for um newline (\n), le o caracter que chega e compoe
      a mensagem.
20            }
21        }
22        if (inChar == 32){ // verifica se a nova string comecou. Detecta o
      espaço e seta a flag.
23            stringRxStartA = true;
24            incomingValueA = 0;
25        }
26    }
27
28    if (VarIndex == true){ // decodifica saida A
29        if (stringRxStartB == true){ // Verifica se a string ja comecou
30            if (inChar == 13) { // se o caracter for um newline (\n),
      identifica que a string terminou.
31                stringRxCompleteB = true;
32                stringRxStartB = false;
33            }
34            else {
35                incomingValueB = incomingValueB*10 + int2ascii(inChar); // se
      o caracter NAO for um newline (\n), le o caractere que chega e compoe
      a mensagem.
36            }
37        }
38        if (inChar == 32){ // verifica se a nova string comecou. Detecta o
      espaço e seta a flag.

```

```

39         string RxStartB = true;
40         incomingValueB = 0;
41     }
42 }
43
44 }
45 }
```

Programa 3.5: Função Serial Event para leitura da saída A e da saída B

Apesar de ser teoricamente possível transmitir apenas uma saída com os blocos e códigos feitos para processos com duas saídas diferentes, recomenda-se utilizar os componentes específicos para saídas únicas para evitar possíveis perdas de performance.

3.2 Diagramas de simulação dos processos-exemplo

Antes de falar sobre os diagramas de simulação para cada um dos processos individualmente, é necessário ressaltar alguns pontos sobre o Simscape e alguns cuidados que se deve ter ao modelar sistemas utilizando-o.

Todo diagrama do Simscape deve conter o bloco chamado *Solver Configuration*, pois este é responsável por especificar o algoritmo utilizado para a simulação (12). Mesmo se um diagrama envolver mais de um princípio físico (e.g termopar, que envolve princípios elétricos e térmicos), uma instância do *Solver Configuration* é suficiente desde que ela esteja ligado ao circuito.

Outros dois blocos notáveis são *Simulink-PS converter* e *PS-Simulink Converter* ou, respectivamente, conversor de sinal do Simulink para sinal físico (*physical signal* ou PS) e conversor de sinal físico para sinal do Simulink. Estes conversores devem ser utilizados para a conexão de blocos do Simulink tradicional, como o bloco de constante, degrau e medição de sinais, com as entradas e saídas de um modelo de simulação física do Simscape (11).

Por fim, observa-se que modelos do Simscape só podem ser executados se o algoritmo de resolução (ou *Solver*) configurado for do tipo degrau fixo (*fixed-step*). Normalmente, o próprio software detecta a necessidade da alteração e pergunta se deve fazê-la.

3.2.1 Tanques cilíndricos desacoplados

A modelagem feita para os tanques cilíndricos desacoplados pode ser vista na figura 3.7. É um modelo que utiliza elementos das bibliotecas hidráulica e mecânica do Simscape.

A entrada enviada pelo ESP32-S é a velocidade de rotação de uma bomba que envia água de um reservatório infinito para o sistema. Quanto maior a velocidade da bomba, maior a vazão. A água passa, então, por um medidor de vazão para monitoramento da vazão de entrada do primeiro tanque. Em seguida, a água desboca no tanque 1 e, posteriormente, no tanque 2. Os níveis dos tanques são considerados saídas dos processos e enviadas para o ESP32-S.

Junto à bomba de água, em paralelo, há uma válvula de alívio da pressão hidráulica. Esta válvula permanece fechada enquanto a pressão da entrada é inferior à pressão configurada. Quando esta pressão é superada, a válvula se abre e permite a passagem do líquido; diminuindo, assim, a pressão hidráulica na tubulação e garantindo maior segurança ao sistema.

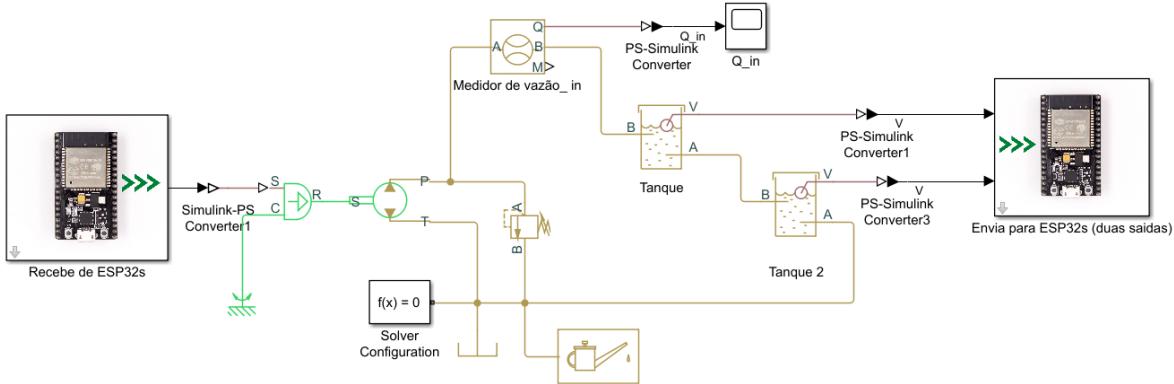


Figura 3.7: Modelo de simulação física do processo de tanque simples construído no Simscape

3.2.2 Filtro passa-baixas

O modelo para o filtro duplo foi construído com os itens da seção de elementos eletrônicos da biblioteca padrão do Simscape e pode ser visualizado na figura 3.8.

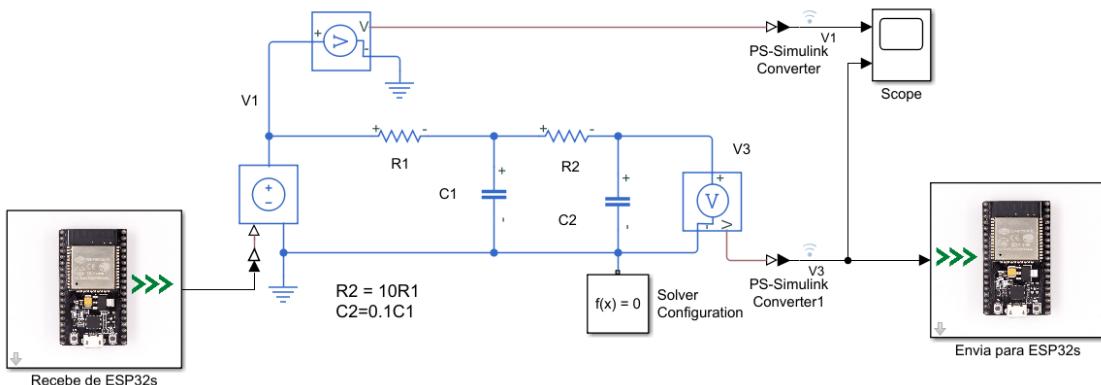


Figura 3.8: Modelo de simulação física do Filtro passa-baixas construído no Simscape

As restrições de que R_2 seja igual a dez vezes R_1 e que C_2 seja o décimo de C_1 foram respeitadas para que a equação 2.17 seja verdadeira e o circuito se comporte de fato como um filtro passa-baixas.

No diagrama, o bloco de recepção de mensagens do ESP32-S se conecta à fonte de tensão, que por sua vez se conecta ao circuito e a um voltímetro. O bloco de envio de mensagens ao ESP32-S se conecta a outro voltímetro, o que mede a tensão de saída do circuito.

3.2.3 Termopar

O modelo para o termopar foi construído com a biblioteca de elementos térmicos e com a de elementos eletrônicos do Simscape e pode ser visto na figura 3.9. O bloco que recebe o valor da entrada do ESP32-S se conecta a uma fonte de calor que entrega em sua saída a diferença de temperatura entre as duas entradas (A entrada S de comando, e a entrada A de referência). Colocando a referência térmica na entrada A, o valor da temperatura de saída da fonte de calor é igual ao valor presente na entrada S. O sinal recebido do ESP32-S é somado à constante 273.15 para que o ESP32-S possa enviar a temperatura em graus Celsius em vez de Kelvin.

A saída da fonte de temperatura é conectada a um termômetro para aferição da temperatura (entrada do processo) e à junção do termopar. O bloco de termopar deve ser configurado com o tipo desejado e os coeficientes disponíveis no site do *National Institute of Standards and Technology*. As pontas soltas do termopar estão conectadas a um voltímetro, que está conectado ao bloco de envio de mensagens para o ESP32-S. Como a tensão de um termopar está na ordem dos milivolts, mV, ela é multiplicada por 1000 antes de ser enviada para facilitar a codificação e a leitura do número.

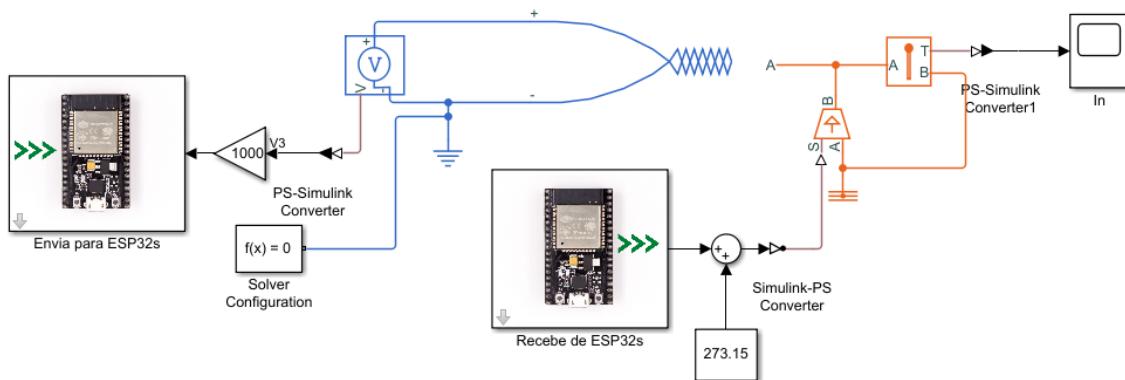


Figura 3.9: Modelo de simulação física do termopar construído no Simscape

3.3 Procedimentos necessários para utilização dos métodos criados

Esta seção apresenta os procedimentos necessários para a utilização dos métodos criados e propostos de simulação *Hardware in the loop*, incluindo a montagem física dos componentes, configurações na IDE de programação do Arduino para carregamento do código-fonte no ESP32-S e ações a serem tomadas no Simulink/Simscape do Matlab para conexão com o Raspberry Pi.

3.3.1 Montagem de hardware

Além do Raspberry Pi 3B e do ESP32-S, utilizou-se um potenciômetro na montagem física dos experimentos *Hardware in the loop* realizados. Sua função é fazer parte de um circuito divisor de tensão que se conecta à entrada analógica do ESP32-S, simulando o sinal analógico enviado pelo CLP do MICAsh. Desta forma, ao variar a resistência do potenciômetro, altera-se também a tensão que o ESP32-S enxerga como a entrada do processo que deve ser enviada ao Raspberry Pi 3B.

Para o envio e recebimento de mensagens via comunicação serial, dois dos pinos UART (*Universal asynchronous receiver-transmitter* ou Receptor-transmissor univeral e assíncrono) do Raspberry Pi foram utilizados, tais como dois dos pinos UART do ESP32-S. Além disso, utilizou-se três pinos do ESP32-S para a montagem do circuito divisor de tensão do potenciômetro: Os de alimentação (3V3 e GND) e um canal analógico-digital. A tabela 3.3 lista os pinos utilizados do ESP32-S e suas ligações enquanto a figura 3.10 ilustra as conexões.

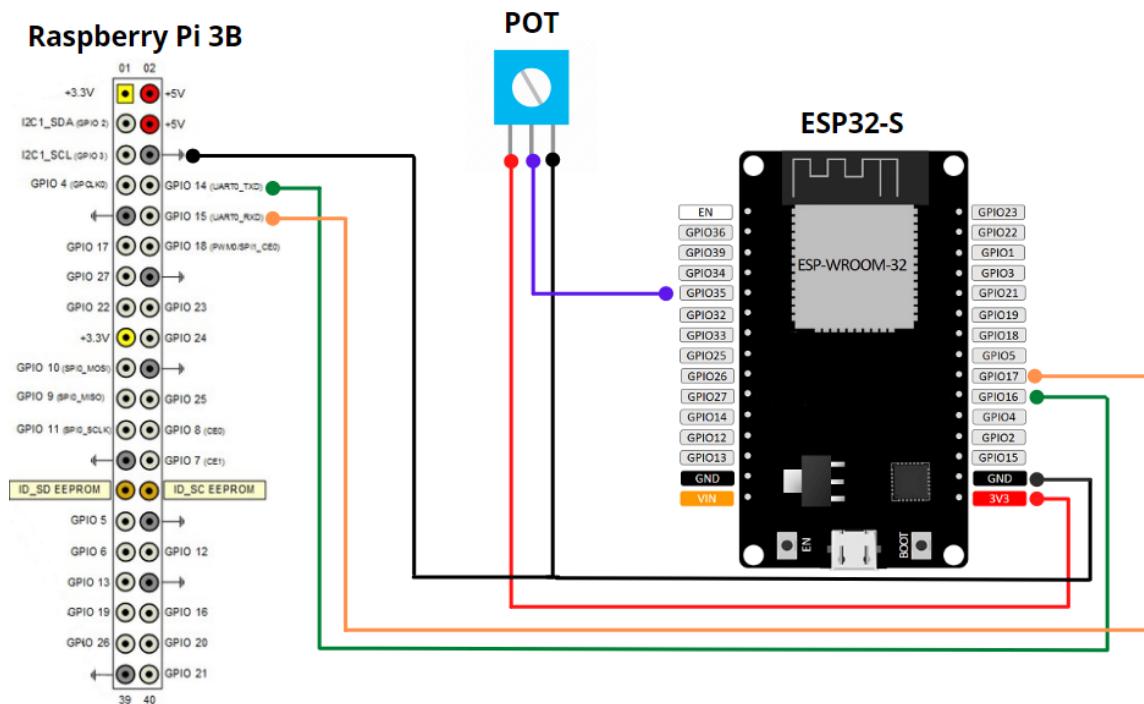


Figura 3.10: Esquema da montagem utilizada nos experimentos HIL

Tanto o Raspberry Pi quanto o ESP32-S foram energizados com cabos USB conectados ao computador que executou as simulações no Simscape, pois se desejava monitorar a execução do código-fonte do ESP32-S e os estágios das transmissões. Na aplicação cotidiana do MICAsh, não será necessário conectar o cabo USB no ESP32-S se ele puder ser energizado de outra maneira, pois a saída do sistema poderá ser lida diretamente no MICAsh. O Raspberry Pi deve continuar conectado ao computador, pois forma um conjunto com o MATLAB para executar as simulações e receber e transmitir as mensagens.

Pino do ESP32-S	Conexões
GND	GND do Raspberry Pi Uma da extremidades do potenciômetro
17 (UART2 TX)	Pino 15 (UART0 RX) do Raspberry Pi
16 (UART2 RX)	Pino 14 (UART0 TX) do Raspberry Pi
3v3	A segunda extremidade do potenciômetro
35 (ADC1 CH7)	Pino do meio do potenciômetro

Tabela 3.3: Lista de pinos do ESP32-S utilizados na montagem física dos experimentos HIL e suas conexões

3.3.2 Configuração ESP32-S

O carregamento do código-fonte do ESP32-S e o acompanhamento de sua execução pode ser feito na IDE de programação para placas Arduino, mas antes é necessário configurar o software para tal. Primeiro, é preciso clicar em "Arquivo" » "Preferências" e colar a URL a seguir no campo "URLs adicionais para Gerenciadores de Placas"

https://dl.espressif.com/dl/package_esp32_index.json

Em seguida, é necessário clicar em "Ferramentas" » "Placa" » "Gerenciador de Placas". Na tela que se abre, é possível pesquisar por "esp32" na caixa de buscas para instalar o driver de configuração do módulo. Depois da instalação, será possível selecionar o modelo de placa ESP32 sob utilização.

3.3.3 Configuração Simulink/Simcape

Para conectar o Raspberry Pi e possibilitar que ele receba os resultados das simulações, é necessário instalar a biblioteca de suporte ao Raspberry Pi do Simulink.

Para a instalação, basta abrir um modelo do Simulink e clique em "Apps" » "Get Add-ons" » "Get Hardware support packages", procurar por "Raspberry" e instalar o pacote chamado "Simulink Support Package for Raspberry Pi Hardware". Depois de finalizada a instalação, é preciso clicar "setup now" e seguir as instruções para configurar o Raspberry Pi para funcionar com o Simulink.

Depois, é necessário seguir os itens do passo a passo abaixo com o modelo do Simscape aberto:

1. Na aba "*Modeling*", clicar em "*Model Settings*". Na janela que se abrir, selecionar a opção "*Hardware Implementation*" e selecionar Raspberry Pi no campo "*Hardware Board*". As informações padrão do Raspberry Pi serão carregadas.
2. Na aba "*Simulation*", clicar na seta voltada para baixo que se localiza abaixo do botão "*Run*". Na lista de seleção, clicar em "*Simulation Pacing*" e configurar como na figura 3.11.

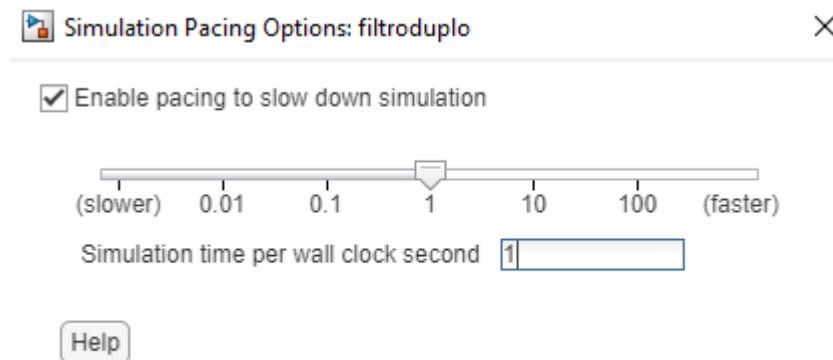


Figura 3.11: Configuração do ritmo de simulação. O número 1 na caixa de texto diminui o tempo de simulação na escala 1:1, fazendo com que um segundo da simulação seja o mais próximo possível de um segundo real.

3. Conectar os blocos de recepção de mensagem e envio de mensagem à entrada e saída do processo, respectivamente.
4. Configurar os blocos, clicando duas vezes nele para editar as informações. Nesta etapa, é necessário tomar alguns cuidados:
 - O endereço ”/dev/ttyS0” se refere aos pinos 14 e 15 do Raspberry Pi 3B, modelo utilizado nos experimentos aqui registrados, mas podem variar em outros modelos.
 - A taxa de transmissão (*baudarate*) selecionada deve ser a mesma configurada no código-fonte carregado no ESP32-S.
 - Os valores máximos e mínimos para a entrada e saída do processo devem ser os mesmos utilizados no código-fonte carregado no ESP32-S.
5. Na aba ”Hardware”, na seção ”Mode”, a opção selecionada deve ser ”Connected IO”. Se os dizeres ”Run on Board” aparecerem, basta clicar neles para habilitar a opção correta.

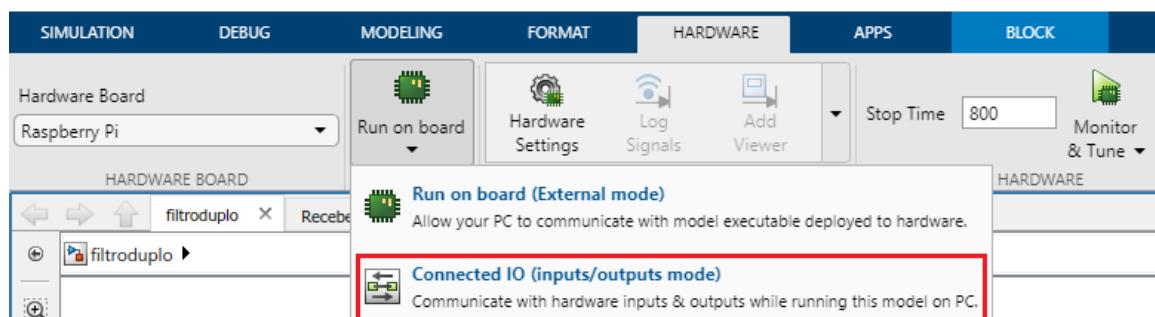


Figura 3.12: Localização da opção de modo ”Connected IO”

Para executar a simulação, basta clicar no botão ”Run with IO” na aba ”Hardware”, como ilustrado na figura 3.12. Vale ressaltar que esta opção aparece apenas se o modo selecionado for ”Connected IO” e que o Raspberry Pi deve estar conectado ao computador.

Um modelo pré-configurado foi disponibilizado para maior comodidade. Trata-se de um arquivo que já conta com os blocos de envio e recepção de mensagens serial criados e com a configuração necessária. Ele pode ser acessado no link disponível na conclusão deste relatório..

3.4 Comentários finais

Este capítulo demonstrou e explicou o funcionamento das funções, códigos-fonte e diagramas de simulação confeccionados. Além disso, apresentou os procedimentos necessários para utilizar os métodos propostos, como conexões físicas e configurações a serem realizadas; disponibilizando espécie de roteiro para replicar os experimentos HIL realizados durante o projeto.

Capítulo 4

Análise dos resultados obtidos

Este capítulo apresenta os resultados obtidos com os modelos criados para os processos-exemplo, comparando suas saídas com as saídas esperadas e identificando se os valores de saída registradas pelo ESP32-S ao receber as mensagens pelo canal serial condizem com os calculados pelo Simulink.

Diversos experimentos foram realizados com os modelos dos processos-exemplo, cada um com padrões de sinal de entrada diferentes. Para cada experimento, avaliou-se os resultados das simulações estão de acordo com os valores esperados a julgar pelas equações que regem os processos e foram registradas os sinais de saída no Simscape e no ESP32-S. Assim, pode-se observar o quanto diferente é o sinal de saída recebido e decodificado pelo ESP32-S – os valores que vão para o CLP do MICAsh – do sinal real de saída da simulação e avaliar se o atraso ou perda de resolução durante a comunicação são pequenos o suficiente para serem considerados irrelevantes.

Esta análise é feita de duas formas: visualmente, por meio de gráficos que contrapoem as duas saídas, e matematicamente; por meio do cálculo do erro quadrático médio entre os dois sinais. Quanto menor este erro, mais exato é o sinal recebido e codificado pelo ESP32-S em relação à saída calculada pelo Simscape.

As seções abaixo descrevem os experimentos realizados para cada um dos processos e reúnem os gráficos e os valores do erro quadrático médio de cada teste.

4.1 Tanques cilíndricos desacoplados

Foram realizados três experimentos com o sistema de tanques cilíndricos desacoplados, modificando-se o valor da velocidade angular da bomba d'água e verificando-se os volumes nos dois tanques.

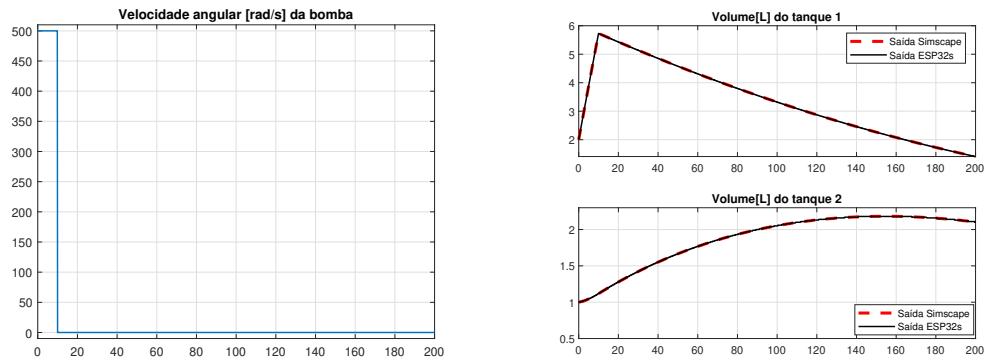
Os tanques foram configurados como iguais, com as seguintes dimensões:

1. Áreas A_1 e A_2 da seção transversal dos tanques: 10m^2
2. Áreas a_1 e a_2 da seção transversal dos canais de saída dos tanques: 3.1415 cm^2

Descreve-se a seguir os três experimentos realizados:

1. Variação em degrau na velocidade da bomba, de 500 a 0 rad/s. Volume inicial do tanque 1, V_{in1} , igual a 2 litros. Volume inicial do tanque 2, V_{in2} igual a 1 litro.
2. Variação em degrau na velocidade da bomba, de 100 a 500 rad/s. $V_{in1} = 2L$ e $V_{in1} = 1L$
3. Variação em degrau na velocidade da bomba, de 100 a 500 rad/s. $V_{in1} = 20L$ e $V_{in1} = 10L$

As figuras 4.1, 4.2 e 4.3 mostram os sinais de entrada aplicados e os gráficos dos volumes dos dois tanques em função do tempo para cada experimento.



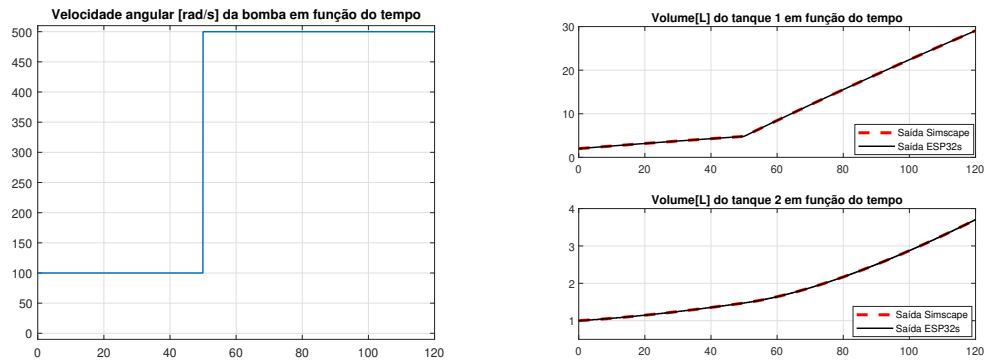
(a) Entrada do experimento 1 com o sistema de tanques acoplados (b) Saídas do experimento 1 com o sistema de tanques acoplados

Figura 4.1: Resultados do primeiro experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S

Para o primeiro experimento, 4.1 observa-se que o volume do tanque 1 diminui enquanto o volume do tanque 2 aumenta, ambos de maneira não-linear. Comparando esta tendência com as equações 2.2 a 2.7, pode-se concluir que o comportamento dos volumes dos tanques foi como o esperado. Sem vazão de entrada, o tanque 1 se esvazia à medida que a água escorre pela saída a caminho do tanque 2, com vazão proporcional à raiz quadrada da altura da coluna d'água; o que explica a não-linearidade do volume do tanque 1.

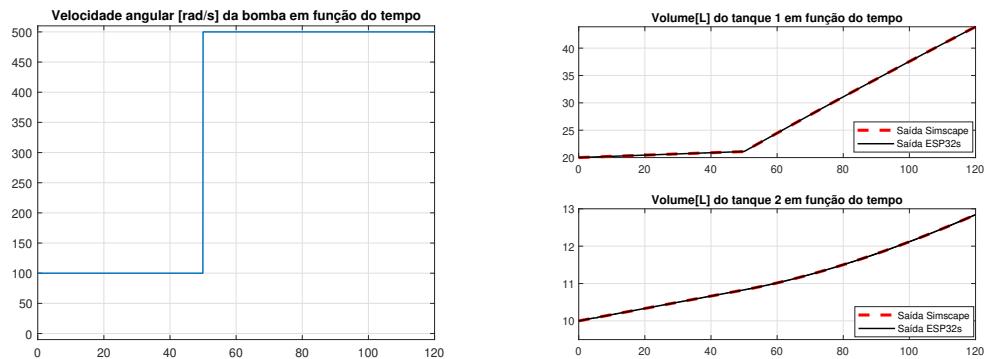
O volume do tanque 2 aumenta, pois sua vazão de entrada, que corresponde à vazão de saída do primeiro tanque, é maior que vazão de saída do tanque 2; justificado pelo fato de que, inicialmente, o volume no tanque 1 é maior do que o volume do tanque 2 e a vazão de saída é diretamente proporcional ao volume. Entretanto, à medida que o volume do tanque 1, e consequentemente a vazão de saída do tanque 1, se aproximam do volume e vazão de saída do tanque 2, a taxa de crescimento do volume do tanque 2 diminui. Depois, o volume do tanque 2 começa a diminuir, pois a vazão de entrada no tanque é pequena – ou nula – já que há pouca ou nenhuma água restante no primeiro tanque.

No segundo experimento, figuras 4.2, verifica-se que com o aumento da vazão de entrada, aumentam os volumes em ambos os tanques. Este comportamento também é explicado pelas equações 2.2 a 2.7. Quanto maior a vazão de entrada em relação à vazão de saída, maior a taxa de crescimento do volume do tanque. De forma análoga, quanto maior o volume do tanque, maior a vazão de saída deste mesmo tanque. Assim, explica-se o crescimento do volume do tanque 2, pois este recebe a vazão de saída do tanque 1.



(a) Entrada do experimento 2 com o sistema de tanques acoplados (b) Saídas do experimento 2 com o sistema de tanques acoplados

Figura 4.2: Resultados do segundo experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S



(a) Entrada do experimento 3 com o sistema de tanques acoplados (b) Saídas do experimento 3 com o sistema de tanques acoplados

Figura 4.3: Resultados do terceiro experimento com o sistema de tanques acoplados, incluindo comparação das saídas registradas no Simscape e no ESP32-S

A análise do terceiro experimento, figuras 4.3, é melhor feita se comparada com a do segundo, pois o sinal de entrada aplicado é o mesmo e a única alteração entre os dois testes é nas condições iniciais do sistema. No terceiro experimento, pode-se verificar a influência do volume do tanque – e da altura da coluna d’água – nas taxas de variação dos volumes dos tanques. No segundo experimento, percebe-se que o tanque 2 teve variação de volume menor nos primeiros 50 segundos em comparação com o terceiro; comportamento explicado pelo fato de que o tanque 1 tinha mais água – e vazão de saída maior – no início do terceiro experimento.

Entretanto, este efeito é menos perceptível depois do aumento da velocidade da bomba. A vazão de entrada maior no sistema compensa a diferença de volumes iniciais e, no fim, o tanque 2 teve variação de volume inferior no experimento 3 em comparação com o segundo. Afinal de contas, maiores volumes causam vazões de saída maiores.

A tabela 4.1 reúne os valores do Erro Quadrático Médio entre os sinais de saída do Simscape e aqueles recebidos e codificados pelo ESP32-S para o envio ao CLP do MICAsh para ambos os tanques em todos os experimentos. A análise dos valores da tabela junto com a dos gráficos anteriores permitem concluir que os valores lidos pelo ESP32-S não se distanciam dos valores reais das saídas das simulações de maneira que prejudique a operação do MICAsh no controle do processo como experimento HIL.

Experimento	Erro quadrático médio	
	Tanque 1	Tanque 2
1	0.00289 (0.289%)	0.00281 (0.281%)
2	0.00302 (0.302%)	0.00287 (0.287%)
3	0.00323 (0.323%)	0.00298 (0.298%)

Tabela 4.1: Relação de erro quadrático médio entre a saída registrada pelo Simscape e a registrada pelo ESP32-S para os experimentos realizados com o sistema de tanques desacoplados

4.2 Filtro passa-baixas

Foram realizados três experimentos para o filtro passa-baixas. Em todos, variou-se a tensão de entrada do circuito e observou-se a saída. Como $R_2 = 10R_1$ e $C_2 = 0.1C_1$, a equação 2.18 se torna verdade e o circuito deve ter o comportamento de um filtro passa baixas com frequência de corte única.

Desta forma, espera-se que para variações em degrau na entrada, a saída atinja o mesmo valor da saída em regime permanente. Para variações senoidais na entrada, a saída do filtro deve também acompanhar sua entrada, mas com desvio de fase e ligeira diminuição de sua amplitude total; a depender da frequência de corte (4).

Os experimentos realizados consistiram em aplicar nas entradas os três sinais abaixo:

1. Sequência de degraus: um de 0 a 15 V seguido de um de 15 a 0 V.
2. Senoide de amplitude $A = 10V$ e frequência $F = 0.002 \text{ rad/s}$
3. Sequência de degraus positivos crescentes.

A senoide foi produzida com o bloco próprio do Simulink. Os degraus foram aplicados mediante variação do potenciômetro e envio do valor correspondente via comunicação serialm utilizando os métodos descritos no capítulo anterior.

As figuras 4.4 apresentam os resultados dos experimentos.

Analisando os gráficos, observa-se que as saídas dos processos seguiram os padrões esperados. Além disso, verifica-se que os sinais de saída recebidos pelo ESP32-S se aproximam quase perfeitamente dos sinai de saída calculados pelo Simscape. A tabela 4.2 dispõe o erro

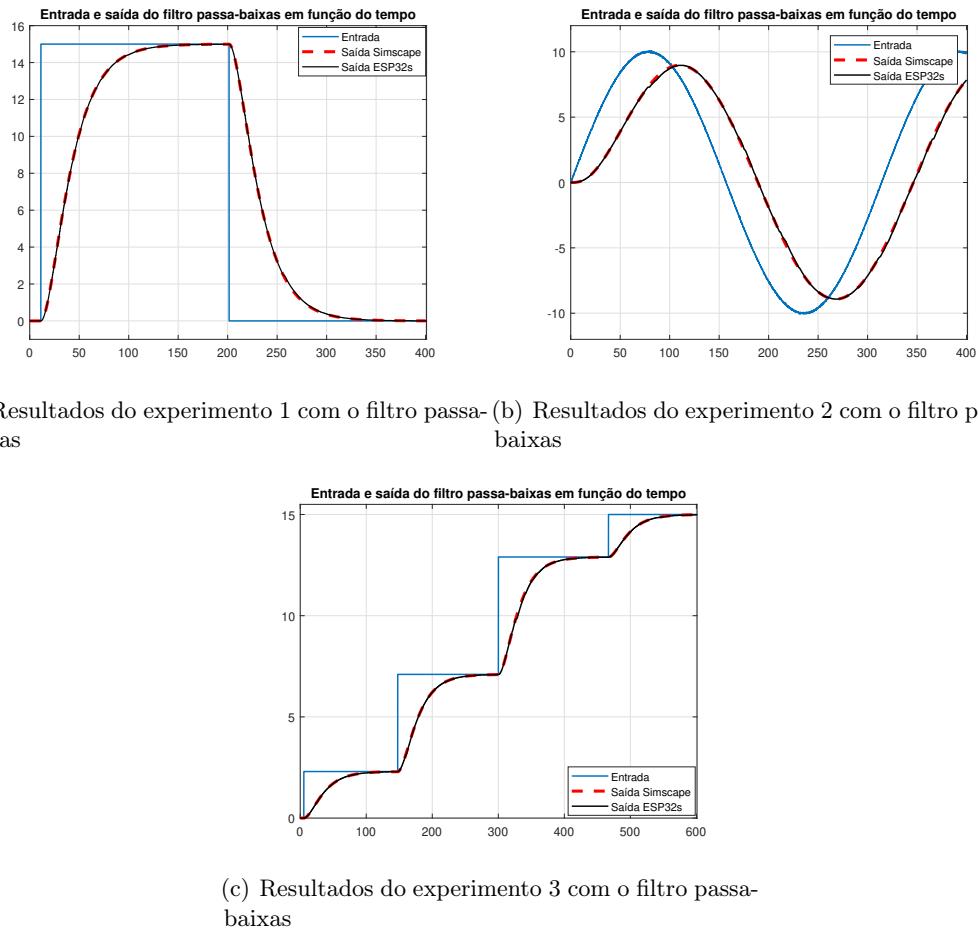


Figura 4.4: Resultados dos experimentos com o filtro passa-baixas, incluindo comparação das saídas registradas no Simscape e no ESP32-S

quadrático médio entre os dois sinais de saída traçados em cada gráfico. Analisando, percebe-se que os valores dos erros não atingem a marca de 1%, comprovando que não houve perdas significativas do sinal na transmissão para o ESP32-S.

Experimento	Erro quadrático médio
1	0.00332 (0.332%)
2	0.00510 (0.510%)
3	0.00351 (0.351%)

Tabela 4.2: Relação de erro quadrático médio entre a saída registrada pelo Simscape e a registrada pelo ESP32-S para os experimentos com o filtro passa-baixas

4.3 Termopar

Foram realizados três experimentos com o modelo do termopar da figura 3.9, que foi configurado como um termopar do tipo K. Cada experimento atingiu faixas de temperatura diferentes com o objetivo de ver diferentes valores de tensão de saída do termopar.

Os experimentos foram realizados todos mediante variação do potenciômetro conectado no ESP32-S. Esta variação correspondia a variação na temperatura na junção do termopar. Lista-se abaixo as faixas de temperatura consideradas em cada experimento.

1. 0°C a 80°C
2. 223°C a 423°C
3. -30°C a 0°C

A tabela 4.3 reúne os coeficientes das equações 2.20 e 2.21 configurados no modelo para a realização dos experimentos. Os experimentos 1 e 2 utilizaram os termos da coluna $T > 0^\circ\text{C}$ e a equação 2.21. O experimento 3 utilizou os valores da coluna $T \leq 0^\circ\text{C}$ e a equação 2.20.

Coeficiente	$T \leq 0^\circ\text{C}$	$T > 0^\circ\text{C}$
C_0	0	-0.176004136860e-1
C_1	0.394501280250e-1	0.389212049750e-1
C_2	0.236223735980e-4	0.185587700320e-4
C_3	-0.328589067840e-6	-0.994575928740e-7
C_4	-0.499048287770e-8	0.318409457190e-9
C_5	-0.67509059173e-10	-0.560728448890e-12
C_6	-0.574103274280e-12	0.560750590590e-15
C_7	-0.310888728940e-14	-0.320207200030e-18
C_8	-0.104516093650e-16	0.971511471520e-22
C_9	-0.198892668780e-19	-0.121047212750e-25
C_{10}	-0.163226974860e-22	0
a_1	0	0.118597600000
a_2	0	-0.118343200000e-3
a_3	0	0.126968600000e3

Tabela 4.3: Coeficientes do termopar do tipo K utilizados para temperatura T maior e menor que zero. Valores retirados de (14)

As figuras 4.5 mostram os gráficos das temperaturas aplicadas no termopar e as respectivas diferenças de potencial em função do tempo para os três experimentos.

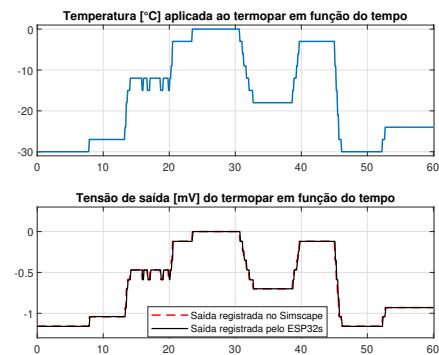
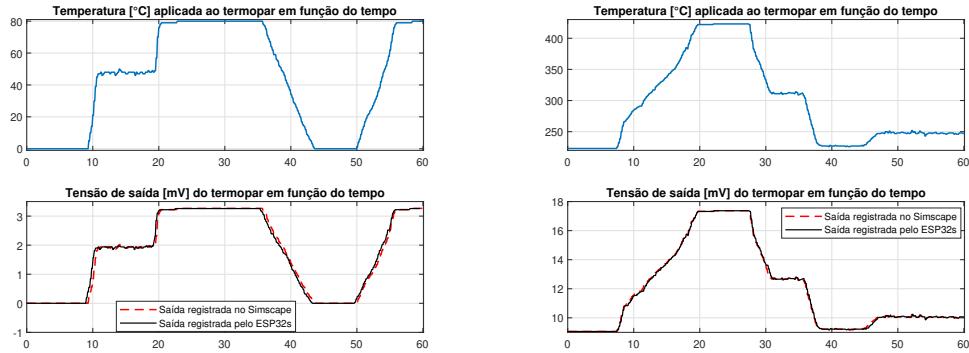


Figura 4.5: Resultados dos experimentos com o termopar, incluindo comparação das saídas registradas no Simscape e no ESP32-S

Substituindo os valores de temperatura dos gráficos e os coeficientes da tabela 4.3, encontra-se os valores de tensão mostrados nas figuras 4.5. Desta forma, as tensões apresentadas pelo modelo correspondem aos valores esperados e pode-se dizer que o modelo do termopar utilizado se aproxima satisfatoriamente de um termopar real.

Além de verificar a saída do modelo, comparou-se também os valores da saída recebida pelo ESP32-S com o valor do Simscape. A tabela 4.4 reúne o Erro Quadrático Médio para cada experimento realizado com o termopar. Analisando-a, percebe-se que os erros são valores baixos e, junto com análise dos gráficos da figura 4.5, pode-se concluir que não há perdas significativas no envio da saída para o ESP32-S, seja em resolução, seja em atraso.

Experimento	Erro quadrático médio
1	0.00435 (0.435%)
2	0.00299 (0.299%)
3	0.00254 (0.254%)

Tabela 4.4: Relação de erro quadrático médio entre a saída registrada pelo Simscape e a registrada pelo ESP32-S para os experimentos com Termopar

4.4 Análise geral dos valores de erro

O Erro Quadrático Médio é uma medida utilizada para verificar a variabilidade entre dois conjuntos de dados de tamanho iguais. Frequentemente utilizado para verificar a proximidade entre os valores preditos e reais de uma grandeza, foi utilizado neste trabalho para averiguar a proximidade entre o sinal de saída da simulação e a mesma saída depois de passar pelo canal de comunicação serial e os processos de conversão executados pelo ESP32-S. Em outras palavras, utilizou-se o Erro Quadrático Médio para verificar se a perda de informação na comunicação entre ESP32-S e o conjunto Simscape/Raspberry Pi com os métodos desenvolvidos é relevante e pode prejudicar os experimentos HIL realizados no MICAsh-MCP.

As tabelas 4.1, 4.2 e 4.4 mostram que os valores de erro foram menores que 1%, indicando que pouco se perde durante a transmissão dos valores do processo. Isto confirma que os métodos utilizados para envio e recepção das grandezas não distorcem o sinal para níveis acima de 1% do valor real e que tais métodos podem ser utilizados nos experimentos *Hardware in the loop* do módulo MCP do MICAsh.

As diferenças observadas entre os valores do Simscape e os do ESP32-S podem ser explicadas principalmente pela diferença de resolução entre os dois. Enquanto o Matlab/Simscape trabalha com até quinze casas decimais, o ESP32-S representou as grandezas com duas. Outros fatores aleatórios também podem interferir nesta diferença.

4.5 Observações Gerais

Ao longo dos testes de funcionamento dos métodos e dos experimentos de validação listados nas seções anteriores, observou-se pontos interessantes de serem relatados.

Idealmente, o computador seria utilizado apenas para projetar o modelo do sistema simulado, que seria carregado e executado diretamente no Raspberry Pi. Entretanto, o Simscape não tem suporte para a chamada "simulação externa", que consiste em exportar o diagrama de simulação como código-fonte a ser carregado em outro processador. Assim, faz-se necessário o uso do computador durante a simulação da maneira descrita neste capítulo e no anterior: em conjunto com Raspberry Pi.

Além disso, observou-se que, em alguns momentos, durante a execução das simulações, o Simscape pode exibir a mensagem "*Unable to pace at specified rate*". Isto significa que ele não consegue executar a simulação no ritmo selecionado – ou seja, em tempo real. Observou-se que o número de processos e programas em execução no computador afeta a performance da simulação desta maneira; o que torna a solução desenvolvida mais dependente de boa capacidade de processamento do que o desejado.

O *pace* pode ser alterado para taxas mais lentas e fazê-lo pode contornar o problema, mas para isso será necessário adaptar o programa executado no CLP do MICAsh para garantir que a atuação considere o ritmo desacelerado da simulação.

Por fim, ressalta-se que quanto maior a escala de valores da saída e entrada configurada, menor será a exatidão do número recebido, pois um incremento unitário na escala de envio (0

a 4095) representa um incremento maior na escala real da grandeza. Assim, é recomendado utilizar valores máximo e mínimo tão próximos quanto possível para garantir erro menor entre o valor real e o valor interpretado pela rotina que interpreta a mensagem; seja ela o bloco de receber mensagens no Simulink ou a função "SerialEvent" do ESP32-S.

4.6 Comentários finais

Este capítulo reuniu os resultados dos experimentos realizados com os modelos e métodos criados durante o projeto, avaliando a eficiência e exatidão da comunicação serial entre Simscape/Raspberry Pi e ESP32-S; demonstrando que se perde pouco no envio e recepção e que os métodos podem ser utilizados nos experimentos HIL do módulo MCP do MICAsh.

Capítulo 5

Conclusões e propostas de trabalhos futuros

A partir das análises dos resultados obtidos e da verificação de que os métodos de comunicação serial propostos e elaborados não deformaram os sinais enviados e se mostraram confiáveis, pode-se concluir que os objetivos propostos para o trabalho foram alcançados. Houve, assim, sucesso em desenvolver um método de aplicação factível para simulações *Hardware in the loop* que apresentasse custo relativamente baixo e pudesse ser utilizado para o ensino prático de controle e automação industrial.

Pode-se dizer, também, que a solução proposta apresenta as principais características de um experimento HIL: integração de simulações virtuais de parte de um sistema (modelos do Simscape) com componentes reais (ESP32s, MICAsh e eventuais periféricos) e fluxo de informações bidirecional.

Além disso, obteve-se sucesso em explorar e utilizar o Simscape, o módulo de simulações físicas do Matlab, incorporando-o na solução proposta. Isto faz com que o método de simulação HIL descrito neste relatório seja versátil, pois possibilita que diferentes sistemas sejam modelados e integrados ao MICAsh sem requerer outros materiais senão os descritos anteriormente. Trata-se de uma característica interessante para a aplicabilidade da técnica em ambientes educacionais, pois pode se adaptar a diferentes necessidades.

5.1 Propostas de trabalhos futuros

Quanto ao método de simulação HIL descrito neste texto, elenca-se as seguintes propostas de melhoria e/ou trabalhos futuros.

- Utilizar os métodos de envio e recepção de mensagens serial criados para o Simscape e o ESP32s com sistemas diferentes dos que os tomados como exemplos para testes neste relatório;
- Integrar os métodos de envio e recepção de mensagens serial criados para o Simscape e o ESP32s ao módulo central do MICAsh, que contém o CLP, e executar o controle e/ou atuação das simulações;
- Adaptar os métodos de envio de entrada para o Simscape para suportar sistemas de múltiplas entradas;

- Eliminar a necessidade do computador durante a execução da simulação se o Simscape passar a suportar a simulação externa.

5.2 Disponibilidade dos métodos produzidos

Os diagramas e códigos-fonte criados durante o projeto, incluindo o *template* que contém os blocos para Simulink de recepção e envio de mensagens serial estão disponíveis no link a seguir:

https://github.com/houriigorcosta/MICAShMCP_HIL

Referências Bibliográficas

- [1] A. R. Braga, C. P. Braga, H. C. M. (2010). Sistema de tanques acoplados e desacoplados (stad)para o estudo de controle e automação de processos: Modelagem e controle. In de Automática, S. B., editor, *XVIII Congresso Brasileiro de Automática - CBA2010*, volume Volume único, pages 3058–3064. Sociedade Brasileira de Automática.
- [2] Arduino (2019). Map function for arduino reference page. Disponível em : <https://www.arduino.cc/reference/en/language/functions/math/map>. Acessado em 11 de janeiro de 2022.
- [3] Braga, A. R. (2018). Módulo computador de processo. NT-LEIC12–2018-20 MICAsh COLTEC-UFMG.
- [4] Braga, A. R. (2019). Filtro rc de seção dupla. Technical report, COLTEC/UFMG.
- [5] Braga, A. R. (2022). Medição e compensação digital da dinâmica de um sensor. Technical report, COLTEC/UFMG.
- [6] Doebelin, E. O. (1990). *Measuremente Systems - Application and Design*. McGraq-Hill Publishing Company, fourth edition.
- [7] Fathy, H. K., Filipi, Z. S., Hagenau, J., and Stein, J. L. (2016). Review of hardware-in-the-loop simulation and its prospects in the automotive area. *Proceedings of SPIE - The international society for optical engineering*.
- [8] Feisel, L. D. and Rosa, A. J. (2005). The role of the laboratory in undergraduate engineering education. *Journal of Engineering Education*, 94(1):121–130.
- [9] Grega, W. (1999). Hardware-in-the-loop simulation and its application in control education. *29th ASEE/IEEE Frontiers in Education Conference*.
- [10] Mathworks. Matlab: Designed for the way you think and the work you do. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em 02 de Novembro de 2021.
- [11] Mathworks. Simscape documentation: Simulink-ps converter. Disponível em: <https://www.mathworks.com/help/physmod/simscape/ref/simulinkpsconverter.html>. Acesso em 02 de Novembro de 2021.
- [12] Mathworks. Simscape documentation: Solver configuration. Disponível em: <https://www.mathworks.com/help/physmod/simscape/ref/solverconfiguration.html>. Acesso em 02 de Novembro de 2021.
- [13] Mathworks. Simulink documentation: Serial read. Disponível em:<https://www.mathworks.com/help/supportpkg/raspberrypi/ref/serialread.html>. Acesso em 05 de Novembro de 2021.

- [14] NIST (2018). Nist standard reference database 60, version 2.0 (web version). Disponível em:https://srdata.nist.gov/its90/main/its90_main_page.html. Acesso em 01 de Outubro de 2021.
- [15] Nunes, J. L. (1998). Modeling and control of the quadruple-tank process. Master's thesis, Lund Institute of Technology, Department of Automatic Control. Lund Institute of Technology, box 118. S 221 00. Lund, Sweden.
- [Pi] Pi, R. Raspberry pi 3b. Disponível em: <https://www.raspberrypi.com/products/raspberry-pi-3-model-b/>. Acesso em 02 de Junho de 2021,.
- [17] R.Isermann, J.Schaffnit, and S.Sinsel (1998). Hardware-in-the-loop simulation for the design and testing of engine-control systems. *IFAC Proceedings volumes*, 31:1–10.
- [18] Scervini, M. (2009). Thermocouples: The operating principle. Disponível em: <https://www.msm.cam.ac.uk/utc/thermocouple/pages/ThermocouplesOperatingPrinciples.html>. Acesso em 01 de outubro de 2021.