



UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
TRABALHO DE CONCLUSÃO EM ENGENHARIA DE
CONTROLE E AUTOMAÇÃO

Estudo da Utilização do Matlab/Simulink Aplicado ao Controle de Sistemas a Eventos Discretos

Autor: Anderson S. Carvalho

Orientador: Prof. Dr. Marcelo Götz

Porto Alegre, 03 de Dezembro de 2015

Sumário

Sumário	ii
Agradecimentos	iii
Resumo	iv
Lista de Figuras	v
Lista de Tabelas	vi
Lista de Símbolos	vii
Lista de Abreviaturas e Siglas	viii
1 Introdução	1
1.1 Motivação	1
1.2 Objetivos	1
1.3 Estrutura do Trabalho	1
2 Revisão Bibliográfica	2
2.1 Sistemas a Eventos Discretos	2
2.2 Linguagens Formais e Autômatos	3
2.3 Teoria de Controle Supervisório	5
2.3.1 Controle Monolítico	6
2.3.2 Controle Modular Clássico	6
2.3.3 Controle Modular Local	6
3 Materiais e Métodos	8
3.1 Softwares:	8
3.1.1 Destool FAUDES:	8
3.1.2 Matlab/Simulink:	8
3.1.3 Arduino EC Target:	10
3.2 Hardwares:	11
3.2.1 Arduino UNO:	11
4 Formulação do Problema e Estudo de Caso	12
4.1 Modelagem da Planta	12
4.1.1 Funções:	12
4.1.2 Especificações de Operação:	15
4.2 Cálculo dos Supervisórios	15
4.3 Conversão para Simulink	17
4.4 Simulação	21
4.5 Geração de Código para Arduino	22
5 Resultados	26
6 Conclusões e Trabalhos Futuros	31
7 Referências	32
8 Apêndice	34

Agradecimentos

Agradeço, primeiramente, aos meus pais, Rogério Rocha Carvalho e Rosangela da Silveira, às minhas irmãs, Andressa e Júlia, e a todos os familiares que participaram da minha vida não somente no período de realização deste trabalho, mas também durante toda a graduação pelo apoio, carinho e suporte incondicional a mim fornecido.

Agradecimento especial a minha mãe pela educação, motivação e inspiração passadas a mim durante toda minha vida e que me propiciaram chegar até aqui.

Agradeço ao meu padrasto Claudio Otto Pacheco pelo exemplo de profissional e auxílio para me esclarecer dúvidas sobre engenharia, quando mais precisei.

Agradeço a minha namorada Anelize Soares Müller pelo amor, compreensão e motivação nos momentos mais difíceis, não deixando que eu desistisse em momento algum dos meus objetivos. Agradeço, também, a seus familiares que considero como a minha segunda família.

A todos os professores que fizeram parte desta etapa de minha vida, e que dispuseram de paciência e dedicação para transmitir conhecimento e sabedoria tornando este momento possível. Em especial a meu orientador prof. Dr. Marcelo Götz que sempre esteve disposto a me auxiliar, tendo grande influência na realização deste trabalho.

Finalmente, agradeço a todos os amigos e colegas de trabalho que se fizeram presentes proporcionando momentos de descontração e alegria durante todos esses anos.

Resumo

A Teoria de Controle Supervisório aplicada a sistemas a eventos discretos é uma área muito importante na indústria, principalmente na manufatura. Este projeto foi proposto de maneira a intensificar as experiências práticas deste tema dentro do meio acadêmico. O objetivo do trabalho é o de avaliar o uso do Matlab/Simulink como ferramenta de suporte no desenvolvimento de controle supervisório, até a sua implementação em uma plataforma embarcada de execução. Para avaliação, é realizado o modelo de uma pequena planta de manufatura, e com a ajuda de alguns toolboxes do Matlab, projetar um controle supervisório para a mesma e então realizar a sua conversão para uma linguagem popular em sistemas embarcados, no caso o C/C++. Para tanto, será utilizado o pacote Embedded Code Generator, do Simulink. Assim, espera-se atingir um controle de fácil implementação em uma plataforma, como por exemplo, o Arduino. Além disto, possibilitará um maior aprofundamento do tema, quando da aplicação desta metodologia em sala de aula.

Lista de Figuras

Figura 2.1 - Representação de SED e relação com outros sistemas.....	2
Figura 2.2 - Autômato associado ao conjunto E	4
Figura 2.3 – Função dos possíveis “caminhos” resultantes do produto	4
Figura 2.4 – Função dos possíveis “caminhos” resultantes do paralelo	5
Figura 2.5 – Representação do controle supervísório	5
Figura 3.1 – Ambiente de trabalho Destool	8
Figura 3.2 – Ambiente de trabalho Simulink	9
Figura 3.3 – Blocos com funções para Arduino	10
Figura 3.4 – Plataforma Arduino Uno.....	11
Figura 4.1 – Planta a ser controlada	12
Figura 4.2 – Autômato alimentador	14
Figura 4.3 – Autômato esteira rolante 1	14
Figura 4.4 – Autômato esteira rolante 2	14
Figura 4.5 – Autômato armazenador/despachador	14
Figura 4.6 – Autômatos das especificações 1 e 2.....	15
Figura 4.7 – Autômatos das especificações 3 e 4.....	15
Figura 4.8 – Autômatos dos supervísórios 1 e 2	16
Figura 5.1 – Sinais de comando alimentador e armazenador.....	27
Figura 5.2 – Sinais de comando esteira rolante 1	27
Figura 5.3 – Sinais de comando esteira rolante 2	28
Figura 5.4 – Sinais de comando com interpretação física.....	29
Figura 5.5 – Monitor serial Arduino	30
Figura 8.1 - Malha de controle alimentador	34
Figura 8.2 - Malha de controle esteira rolante 2.....	34
Figura 8.3 - Malha de controle armazenador.....	35

Lista de Tabelas

Tabela 4.1 – Sinais disponíveis na planta	13
Tabela 4.2 – Sinais de comando na planta	18
Tabela 5.1 – Instantes de tempo em que ocorrem os eventos.....	26

Lista de Símbolos

s - Segundos

V - Volts

t - Instante de tempo

ε - Cadeia de eventos vazia

E - Conjunto de eventos

L - Conjunto representativo de uma linguagem formal

X - Conjunto de estados finitos em um autômato

x_o - Estado inicial em um autômato

Σ - Conjunto de eventos em um autômato

X_m - Conjunto de estados marcados ou finais em um autômato

f - Função de transição em um autômato

Lista de Abreviaturas e Siglas

UFRGS - Universidade Federal do Rio Grande do Sul

SED - Sistema a Eventos Discretos

TCS - Teoria de Controle Supervisório

RAM - Random Access Memory

ROM - Read Only Memory

SDVC - Sistema Dinâmico a Variáveis Contínuas

SDVD - Sistema Dinâmico a Variáveis Discretas

CLP - Controlador Lógico Programável

LED - Light Emitting Diode

1 Introdução

O setor de manufatura é uma área importante no ramo industrial, que procura desenvolver novas tecnologias buscando sempre melhorar a qualidade dos produtos e aumentar a produção. Uma das maneiras de se atingir esses objetivos é aumentando o nível de automação das linhas de produção nas fábricas.

Uma das formas de se organizar uma linha de produção automatizada, visando o controle da mesma, é utilizando modelos de Sistemas a Eventos Discretos (SED). Assim, os sinais que envolvem o sistema geram eventos em diversos instantes de tempo. Para auxiliar no projeto de controle destes sistemas uma das propostas é a Teoria de Controle Supervisório (TCS). A TCS é uma tentativa de formalizar o projeto de controle em SED's, propondo formas de modelar uma planta e de gerar seu controle. Assim, pretende-se obter, na planta física que foi modelada, uma operação correta seguindo especificações pré-definidas.

1.1 Motivação

Além dos tópicos já levantados, uma das maiores motivações deste trabalho é estudar novos meios de utilização e desenvolvimento de práticas de modelagem, simulação e aplicação de sistemas voltados ao controle, principalmente, na área da automação industrial, agregando ainda a geração de código para sistemas embarcados. Têm-se em vista o uso de materiais e métodos amplamente difundidos em meio acadêmico, como o Matlab/Simulink, ferramenta muito utilizada neste meio, propiciando assim, a disseminação dos conhecimentos adquiridos neste trabalho em laboratórios e salas de aula.

1.2 Objetivos

Estudar e analisar métodos de utilização de software comum ao meio acadêmico para a modelagem e simulação de sistemas a eventos discretos e controle supervisório, aliando isto à geração de códigos de baixo nível, como C/C++, para aplicação dos sistemas desenvolvidos em sistemas embarcados, possibilitando o desenvolvimento e aprendizado da área, tanto no estudo teórico quanto prático.

1.3 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2 é apresentada uma revisão dos principais conceitos necessários para a compreensão da proposta de trabalho apresentada, juntamente com suas devidas referências. No Capítulo 3 são descritas as ferramentas de hardware e software utilizadas no desenvolvimento do trabalho, bem como a maneira como elas interagem entre si, de forma a compor a estrutura final do projeto. No Capítulo 4 são detalhadas a formulação do problema e o estudo de caso, o processo de modelagem das máquinas que compõe o sistema, o projeto das restrições e do controle supervisório, a conversão dos modelos para o Simulink e a geração de código para sistemas embarcados. No Capítulo 5 são apresentados os resultados obtidos durante as simulações em softwares e com o código salvo no Arduino. Por fim, no Capítulo 6 são apresentadas as conclusões obtidas com este trabalho, bem como, indicações e sugestões de trabalhos futuros que contribuam para a continuação de pesquisas na área.

2 Revisão Bibliográfica

2.1 Sistemas a Eventos Discretos

Sistema é toda entidade que recebe dados em uma ou mais entradas, processa esses dados e gera uma ou mais saídas. Como modelos mais comuns de sistemas existem os modelos estáticos e dinâmicos. Nos modelos estáticos as saídas em um instante de tempo t dependem apenas das entradas neste mesmo instante de tempo. Nos modelos dinâmicos as saídas em um instante de tempo t dependem das entradas neste mesmo instante de tempo e nos instantes anteriores, como se o sistema tivesse uma “memória”. [Götz, 2014].

Um SED é um sistema dinâmico discreto e dirigido por eventos, isto é, com espaço de estados discreto e cuja evolução de estado depende inteiramente da ocorrência assíncrona de eventos discretos [Cunha, 2003]. As características destes sistemas incluem, basicamente, a ocorrência de eventos em paralelo e a evolução do mesmo de acordo com o encadeamento destes eventos. O gráfico de evolução dinâmica difere dos sistemas a variáveis contínuas (SDVC) e a variáveis discretas (SDVD), conforme pode ser visto na Figura 2.1.

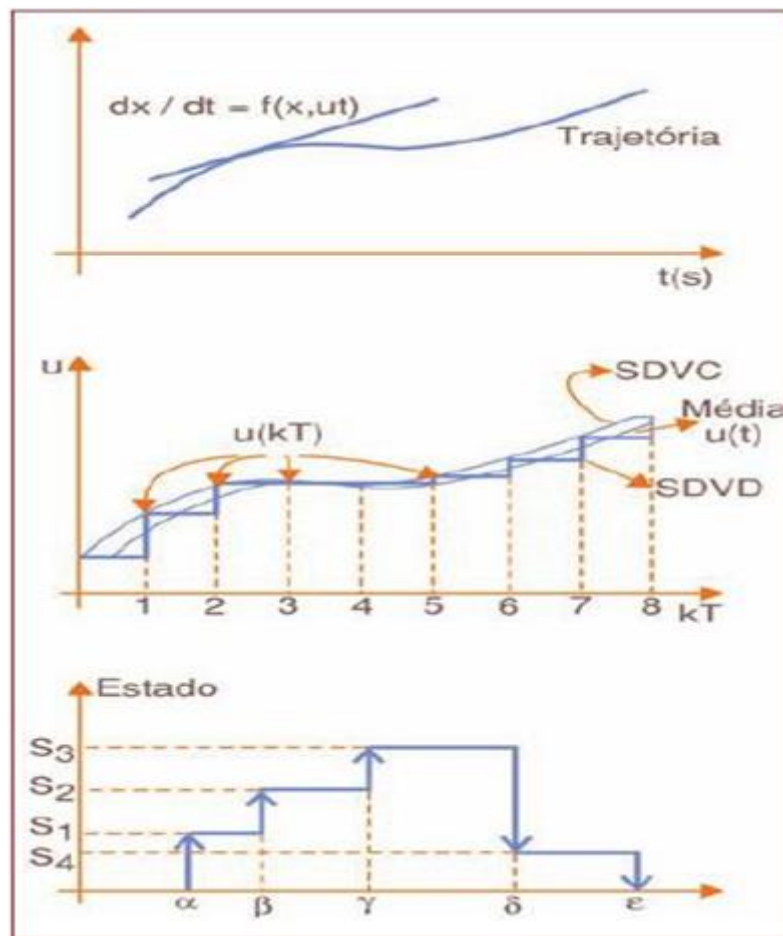


Figura 2.1 - Representação de SED e relação com outros sistemas

Fonte: MECATRÔNICA Atual (2015)

Pela Figura 2.1 pode-se perceber que, ao contrário dos SDVC e SDVD, os SED não podem ser modelados por equações diferenciais e/ou equações de diferenças. É neste momento que *Linguagens e Autômatos* se fazem necessários.

2.2 Linguagens Formais e Autômatos

As linguagens formais (ou linguagens estruturadas em frases) podem ser vistas como conjuntos. Consequentemente, muito da teoria e das principais operações da área de linguagens formais estão baseadas na, ainda mais fundamental, teoria dos conjuntos da matemática discreta [Ramos, 2008].

Cada linguagem possui um *alfabeto* associado, a partir do qual é possível estabelecer diferentes cadeias relacionando cada item do alfabeto em inúmeras sequências distintas. Cada “letra” do alfabeto é um evento e as sequências de eventos possíveis são chamadas de “palavras” [Cassandras e Lafortune, 2008].

Um exemplo de alfabeto é apresentado na Equação 1, bem como, duas linguagens com possíveis sequências de eventos para este alfabeto:

$$E = \{a, b, g\} \quad (1)$$

$$L_1 = \{\varepsilon, a, ab, abb, abg\} \quad (2)$$

$$L_2 = \{g\} \quad (3)$$

A sequência ε é chamada de sequência vazia, ou seja, não há ocorrência de nenhum evento. Há algumas operações que podem ser realizadas com as linguagens e, como ilustração, são apresentadas aqui duas das mais comuns.

Concatenação: Realiza a concatenação entre cada uma das cadeias de eventos presentes nas linguagens, na ordem em que as mesmas são apresentadas.

$$L_1L_2 = \{g, ag, abg, abbg, abgg\} \quad (4)$$

Fechamento-Kleene: Realiza a inclusão de todas as cadeias finitas possíveis, incluindo a cadeia vazia ε .

$$E^* = \{\varepsilon, a, b, g, aa, ab, ag, ba, bb, bg, ga, gb, gg, aaa, aba, \dots\} \quad (5)$$

Os autômatos seriam a representação gráfica das linguagens, indicando os estados existentes, os eventos possíveis e as transições associadas a eles e que interligam um estado a outro.

Desta forma, um autômato determinístico de estados finitos é a quintupla:

$$G = (X, \Sigma, f, x_0, X_m) \quad (6)$$

Onde:

- X é o conjunto finito de estados do autômato;
- Σ é o conjunto de símbolos (eventos) que definem o alfabeto;

- $f : X \times \Sigma \rightarrow X$ é a função de transição, possivelmente parcial, ou seja, não há necessidade da função ser definida para todo elemento de Σ em cada estado de X ;
- x_0 é o estado inicial do autômato;
- X_m é o conjunto de estados marcados ou finais, $X_m \subseteq X$.

Um autômato pode ser representado na forma de um grafo dirigido, onde os nós representam os estados e os arcos são as transições capazes de ocorrer devido a um evento [Cury, 2001]. Um exemplo de autômato associado ao conjunto de eventos da Equação 1 é apresentado na Figura 2.2.

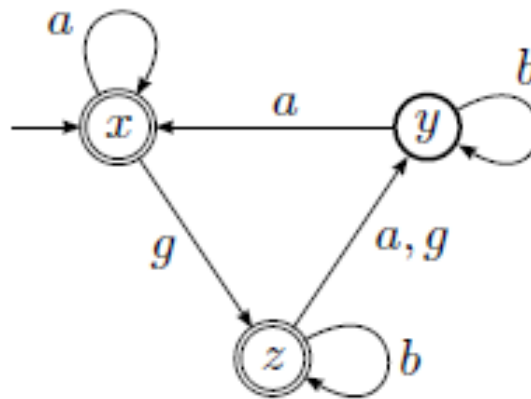


Figura 2.2 - Autômato associado ao conjunto E

Fonte: Götz (2014)

Os nós com dupla circunferência são os estados marcados, e o arco sem associação a um evento indica o estado inicial do autômato.

Entre as operações realizadas com autômatos destacam-se duas, o *produto* e o *paralelo* entre autômatos. Estas operações permitem que modelos sejam combinados para geração de um modelo maior, que represente o funcionamento de todos os autômatos envolvidos na operação. Essas operações são brevemente descritas abaixo, e uma descrição mais detalhada de cada operação é apresentada em [Götz, 2014].

Produto:

$$G_1 \times G_2 = A_c(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1 \times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \quad (7)$$

Onde, A_c representa apenas a linguagem acessível, ou seja, aqueles estados resultantes da operação produto, que não podem ser acessados, são descartados, e ainda:

$$f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)) & \text{se } e \in \Gamma(x_1) \cap \Gamma(x_2) \\ \text{indefinido} & \text{outros casos} \end{cases}$$

Figura 2.3 – Função dos possíveis “caminhos” resultantes do produto

Fonte: Götz (2014)

Paralelo:

$$G_1 || G_2 = A_c(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1||2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \quad (8)$$

$$f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)) & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, e), x_2) & \text{se } e \in \Gamma_1(x_1) - E_2 \\ (x_1, f_2(x_2, e)) & \text{se } e \in \Gamma_2(x_2) - E_1 \\ \text{indefinido} & \text{outros casos} \end{cases}$$

Figura 2.4 – Função dos possíveis “caminhos” resultantes do paralelo

Fonte: Götz (2014)

2.3 Teoria de Controle Supervisório

A teoria de controle supervisório é uma abordagem introduzida por Ramadge e Wonham (1989) que se destina ao controle lógico de SED's, baseado na teoria de linguagens formais e autômatos. Além disso, permite tratar problemas básicos da teoria de controle como controlabilidade e observabilidade.

Os supervisórios funcionam basicamente da seguinte forma: os sinais gerados pela planta são monitorados seguindo as teorias de SED's e então o supervisório “responde” à planta determinando quais sinais podem ou não ocorrer, de acordo com o estado em que a mesma se encontra. Essa determinação segue o conceito de eventos controláveis e observáveis.

- *Eventos controláveis*: São eventos cuja ocorrência pode ser habilitada ou desabilitada pelo supervisório. Geralmente, estão associados a atuadores do sistema, como o evento de acionar um motor elétrico.
- *Eventos observáveis*: Estes eventos são apenas observados (monitorados) pelo supervisório, ou seja, sua ocorrência não pode ser controlada. Estão, geralmente, associados a sinais de sensores.

Na Figura 2.3 temos a demonstração da ideia por trás da teoria de controle supervisório, onde S é o supervisório e G uma planta qualquer sendo controlada pelo mesmo. A planta “informa” ao supervisório o conjunto de eventos (s) que estão ocorrendo no momento e o supervisório condiciona a planta a atuar da forma desejada, habilitando ou desabilitando eventos controláveis.

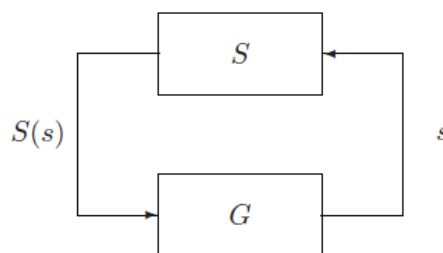


Figura 2.5 – Representação do controle supervisório

Fonte: Cassandras e Lafortune (2008)

Para a síntese do controle supervisorio, a metodologia envolve:

- Modelagem dos componentes da planta por autômatos;
- Descrição dos requisitos e especificações comportamentais desejadas da planta também por autômatos;
- Cálculo do autômato supervisorio para controle da planta, com a utilização de algoritmos específicos.

A operação que realiza o cálculo do autômato supervisorio é chamada de SupC. Após a obtenção do autômato supervisorio, este deve ser implementado dependendo da tecnologia utilizada, como por exemplo, CLP ou alguma plataforma microcontrolada.

Existem três tipos de controle supervisorio difundidos atualmente, descritos na sequência. Maiores detalhes sobre como é feito o controle em cada método e os algoritmos para cálculos destes são apresentados em [Götz, 2014].

2.3.1 Controle Monolítico

O controle monolítico baseia-se no cálculo de um único supervisorio que atuará sobre um modelo único para toda uma planta. Realizam-se as operações de paralelo sobre todos os autômatos dos componentes de um sistema, gerando-se assim um modelo único que representa todos os estados possíveis do sistema.

A mesma operação é realizada sobre todas as especificações da planta, para gerar um autômato de especificações global. Assim, é aplicado o algoritmo *SupC* para calcular o supervisorio capaz de controlar a planta.

Este método de síntese de controle, geralmente, produz grandes supervisórios, em número de estados e transições, que aumenta significativamente a cada novo componente inserido na planta.

2.3.2 Controle Modular Clássico

Neste método segue-se com um único modelo para o sistema, porém, agora são calculados autômatos supervisórios independentes, responsáveis por atender cada especificação da planta.

Dessa forma, são gerados autômatos menores, pois agora o supervisorio está dividido em vários. No entanto, como ainda utiliza-se um único modelo para o sistema real, os tamanhos dos autômatos de cada supervisorio são relativamente grandes.

2.3.3 Controle Modular Local

O controle modular local surge como boa opção, em relação aos métodos anteriores, pois, diminui consideravelmente o tamanho dos autômatos dos supervisórios, bem como, facilita a aplicação de alterações no modelo da planta como, inclusão/exclusão de eventos, adição de novos componentes, entre outros.

Diferentemente dos outros dois métodos, a planta é dividida em módulos locais associados às especificações que os envolvam. Assim, uma especificação que define como

duas máquinas interagem entre si, terá um supervisor calculado sobre um modelo de planta que represente apenas estas duas máquinas, pois o mesmo não exerce controle sobre os outros componentes.

Este método divide a malha de controle em pequenos módulos, gerando autômatos menores e mais fáceis de manipular, facilitando assim a utilização do controle em SED's em sistemas embarcados, como é um dos objetivos deste trabalho. Por esse motivo, este foi o método escolhido para o controle supervisorio no presente trabalho. Porém, como será visto mais adiante, este método pode gerar alguns problemas quando mais de um supervisorio exercer controle sobre um mesmo evento, sendo necessária uma verificação adicional sobre estes eventos.

3 Materiais e Métodos

3.1 Softwares:

3.1.1 Destool FAUDES:

A modelagem dos autômatos e cálculo dos supervisórios é feita utilizando o software FAUDES Destool, desenvolvido pelo grupo de Controle Automático da Universidade de Erlangen-Nuremberg. O software é baseado na análise de sistemas a eventos discretos e realiza operações sobre as linguagens dos autômatos para a geração dos supervisórios.

Na Figura 3.1 é apresentado o ambiente de trabalho deste software.

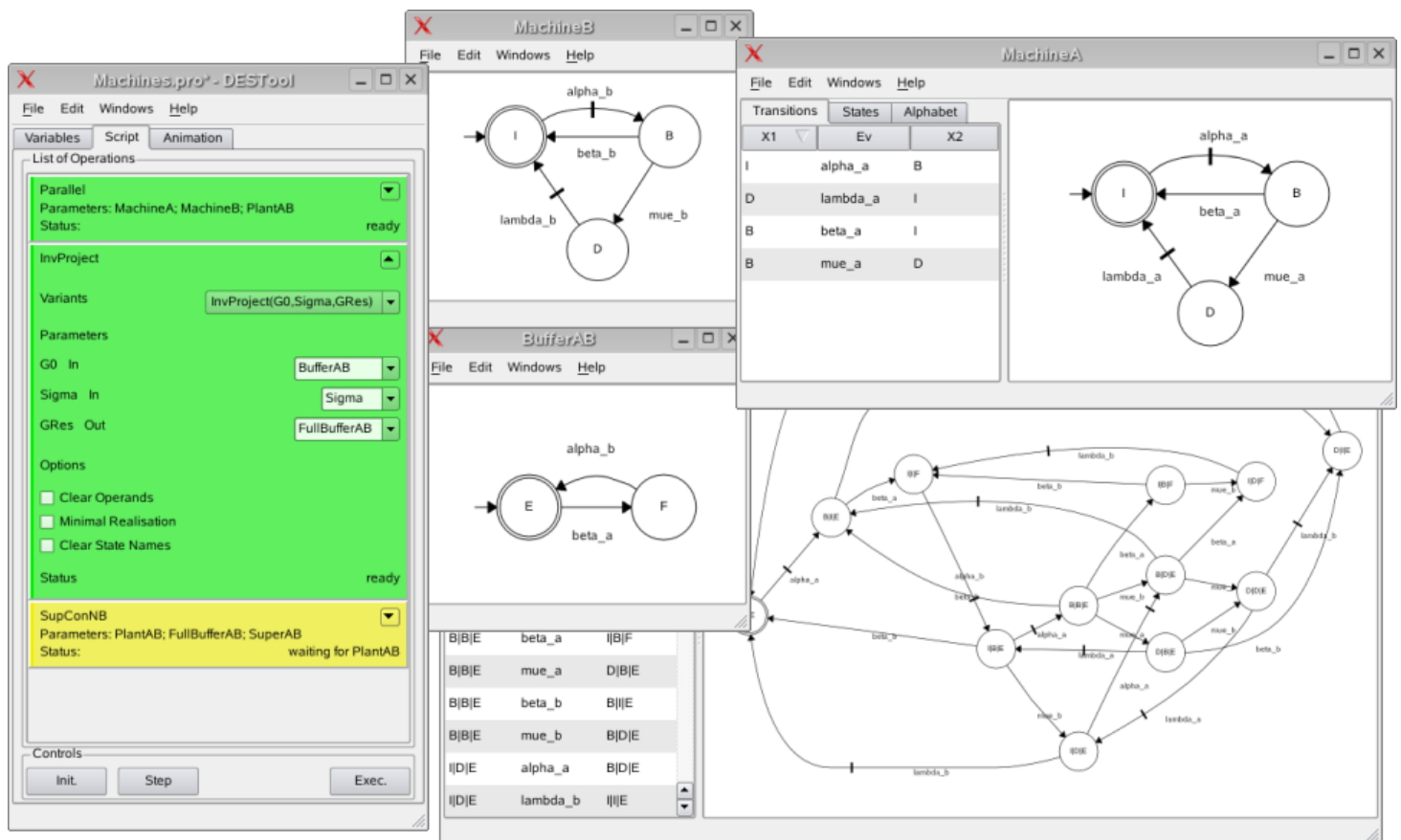


Figura 3.1 – Ambiente de trabalho Destool

Fonte: Friedrich-Alexander Universitaet Erlangen-Nuernberg (2015)

3.1.2 Matlab/Simulink:

O Matlab (MATrix LABORatory) é um software interativo, que integra a capacidade de fazer cálculos, visualização gráfica e programação em um ambiente fácil de usar, em que problemas e soluções são expressos em uma linguagem matemática familiar. Os usos típicos para o Matlab incluem:

- Cálculos matemáticos;
- Desenvolvimento de algoritmos;
- Modelagem, simulação e confecção de protótipos;

- Análise, exploração e visualização de dados;
- Gráficos científicos e da engenharia;
- Desenvolvimento de aplicações, incluindo a elaboração de interfaces gráficas com o usuário.

O Matlab possui diversos módulos, chamados de pacotes, que agrupam funções para diferentes aplicações. Uma das ferramentas mais conhecidas e que foi amplamente utilizada neste trabalho é o Simulink.

O Simulink é uma ferramenta para modelar, simular, e analisar sistemas dinâmicos, baseando-se em modelos. Suporta sistemas lineares e não lineares modelados em tempo contínuo, tempo discreto ou com uma mistura dos dois. Os sistemas também podem ter partes diferentes que são amostradas ou atualizadas com taxas diferentes de tempo.

A Figura 3.2 apresenta um pouco do ambiente de trabalho do Simulink. É neste ambiente que foram realizadas a implementação dos supervisórios, suas interações e a geração do código para sistemas embarcados. A versão do Matlab utilizada neste trabalho foi a 2015a.

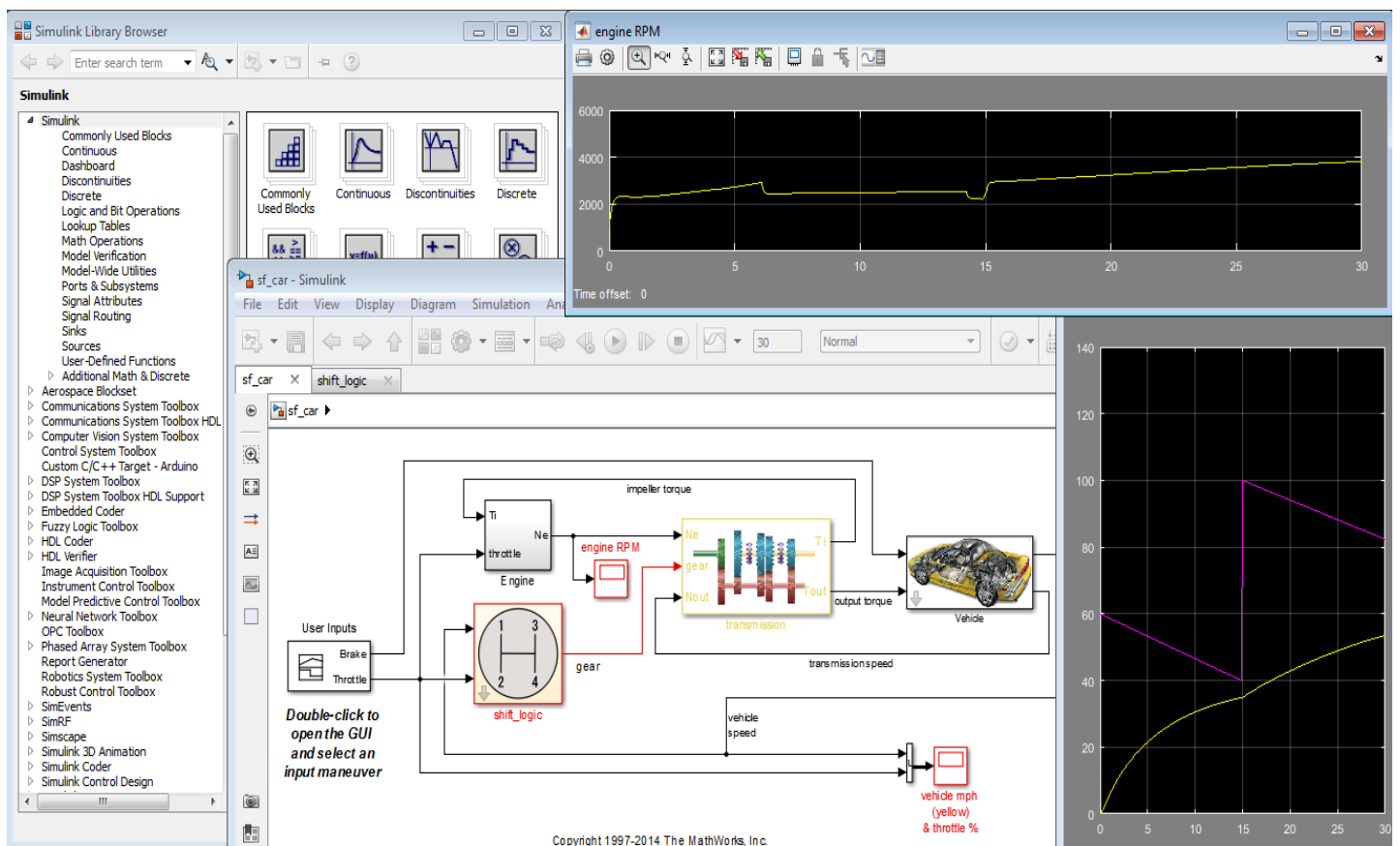


Figura 3.2 – Ambiente de trabalho Simulink

Fonte: Autor

3.1.3 Arduino EC Target:

Uma ferramenta extremamente útil do Simulink é o Embedded Coder (Codificador Embarcado, em tradução livre), capaz de gerar códigos para sistemas embarcados que representem os modelos e funções desenvolvidas, por meio de blocos, no ambiente do Simulink.

Para a geração de códigos que possam ser utilizados no sistema embarcado utilizado neste trabalho (Arduino), a MathWorks, desenvolvedora do software Matlab, disponibiliza um arquivo de extensão que pode ser adicionado ao Simulink, o Embedded Coder Target for Arduino.

Essa extensão possibilita, além da geração de códigos em C/C++ adaptados para as plataformas Arduino, a utilização de alguns blocos que representam portas físicas e funções internas do sistema embarcado e a compilação e download dos códigos diretamente para as plataformas. Na Figura 3.3 são apresentados alguns blocos disponíveis para Arduino.

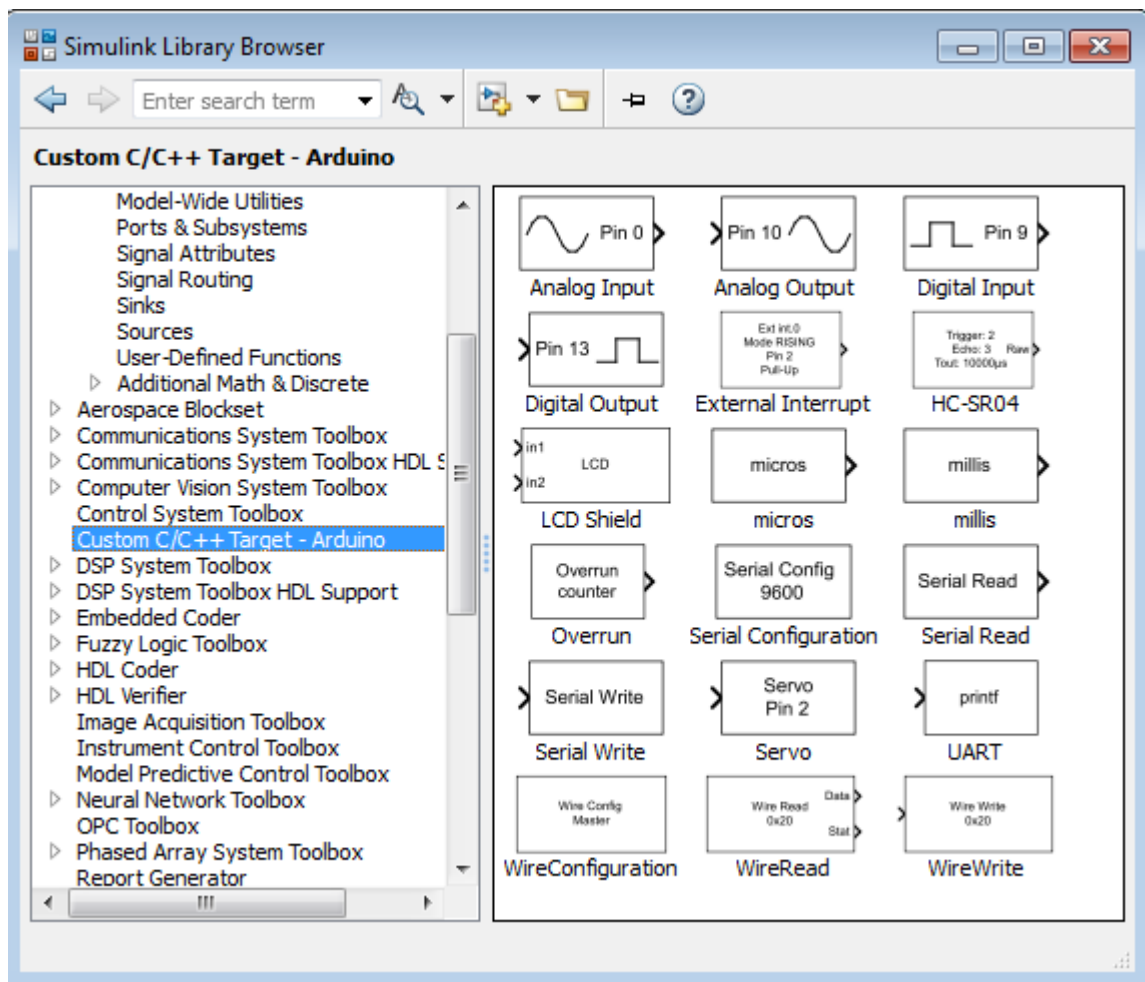


Figura 3.3 – Blocos com funções para Arduino

Fonte: Autor

3.2 Hardwares:

3.2.1 Arduino UNO:

O Arduino é uma plataforma de prototipagem de código aberto baseado em hardware e software de fácil utilização. As placas Arduino são capazes de ler entradas como a intensidade de luz em um sensor, o acionamento de um botão e/ou uma mensagem de Twitter, e transformá-las em saídas como a ativação de um motor, ligar um LED e/ou publicar algo online.

A plataforma utilizada neste trabalho é o Arduino UNO, uma das plataformas mais difundidas em centros acadêmicos, de baixíssimo custo. Ela possui um microcontrolador ATmega 328p, programado através de linguagem C/C++.

As principais características desta plataforma são [Arduino, 2015]:

- 13 portas digitais, podendo configurar-se como entrada ou saída;
- 5 portas analógicas, também configuráveis;
- Saídas de referência de tensão de 3,3 V e 5 V.

Na Figura 3.4 é mostrada a plataforma utilizada neste trabalho.



Figura 3.4 – Plataforma Arduino Uno

Fonte: Arduino (2015)

4 Formulação do Problema e Estudo de Caso

O trabalho consiste no estudo da modelagem e controle de uma mini planta industrial, baseando-se na análise de sistemas a eventos discretos e a teoria de controle supervisão, utilizando a ferramenta Simulink para simulações de desempenho e bibliotecas auxiliares para a geração de código em C/C++, permitindo a aplicação dos supervisórios em sistemas embarcados, posteriormente.

Inicialmente foi definida uma pequena planta com quatro “máquinas”, um alimentador (*stack feeder*), duas esteiras rolantes (*conveyor belt*) e um armazenador/despachador (*exit slide*). Esta planta é ilustrada pela Figura 4.1. As peças são colocadas manualmente no sistema pelo Alimentador, e as esteiras transportam as peças até o Armazenador/Despachador.

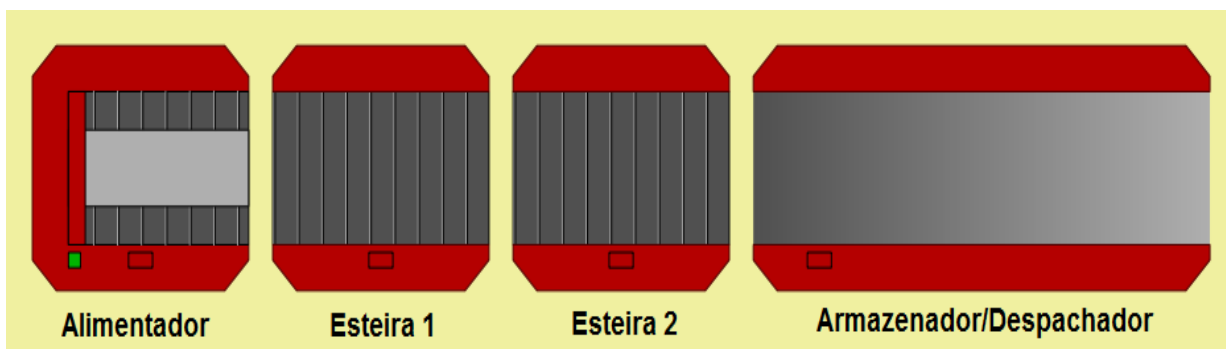


Figura 4.1 – Planta a ser controlada

Fonte: Autor

4.1 Modelagem da Planta

4.1.1 Funções:

Para a modelagem da planta é preciso estabelecer o mecanismo de funcionamento de cada máquina e dos sinais provenientes de cada uma que serão considerados, como sensores de presença de peças, acionamento de esteiras e acionamento de descarga de peças.

As funções de cada máquina na planta são:

Alimentador: Recebe, manualmente, as peças que passarão pela planta e conduz as mesmas até a máquina seguinte;

Esteira Rolante: Conduz peças de um ponto a outro da planta;

Armazenador/Despachador: Recebe peças no final da planta, podendo armazenar até 4 peças simultaneamente, e despacha as mesmas ao final do processo.

Na Tabela 1 estão declarados todos os sinais definidos neste trabalho para a correta operação da planta da Figura 4.1. Com base nestes sinais, foram definidos os modos de funcionamento de cada máquina, que serão utilizados para a formação dos autômatos que representarão as mesmas. Os modos de operação são descritos a seguir.

Alimentador: Quando o sinal de chegada de peça é enviado pelo sensor, liga-se a esteira para movimentar a peça. Ocorre então o sinal de saída de peça. Após este movimento, recebe-se o sinal de que a esteira retornou a posição inicial, então, desliga-se a esteira.

Esteira Rolante 1: A esteira é ligada inicialmente, os sinais de chegada e saída de peça ocorrem, nesta ordem. Por fim, a esteira é desligada.

Esteira Rolante 2: Segue a mesma lógica de funcionamento da esteira rolante 1, alterando-se apenas a nomenclatura dos sinais recebidos.

Armazenador/Despachador: Ao receber-se o sinal de chegada de peça, o sistema de recolhimento é ativado. Após recolherem-se as peças presentes, recebe-se o sinal de saída de peças.

Tabela 4.1 – Sinais disponíveis na planta

Fonte: Autor

Sinal	Objeto	Descrição
sf_fdhome	Alimentador	Sinal de posição inicial
sf_wpar	Alimentador	Sinal de chegada de peça
sf_wplv	Alimentador	Sinal de saída de peça
sf_fdon	Alimentador	Sinal para ligar esteira
sf_fdoff	Alimentador	Sinal para desligar esteira
cb_wpar	Esteira rolante	Sinal de chegada de peça
cb_wplv	Esteira rolante	Sinal de saída de peça
cb_bm	Esteira rolante	Sinal para ligar esteira
cb_boff	Esteira rolante	Sinal para desligar esteira
xs_wpar	Armazenador/Despachador	Sinal de chegada de peça
xs_wplv	Armazenador/Despachador	Sinal de saída de peça
xs_getwp	Armazenador/Despachador	Sinal para retirar peças

Os autômatos que representam cada máquina da planta são mostrados nas figuras 4.2, 4.3, 4.4 e 4.5. As transições com traços ao centro representam os eventos controláveis do sistema, enquanto as transições simples representam aqueles eventos apenas observáveis.

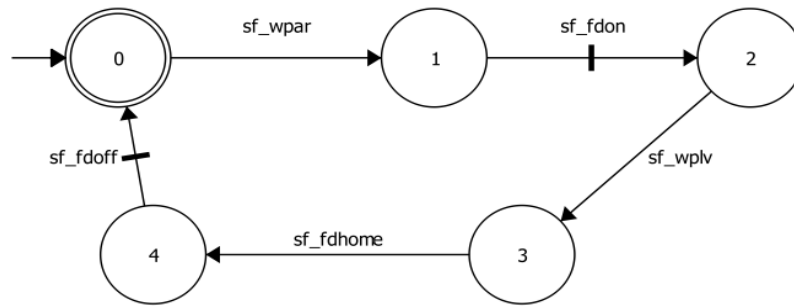


Figura 4.2 – Autômato alimentador

Fonte: Autor

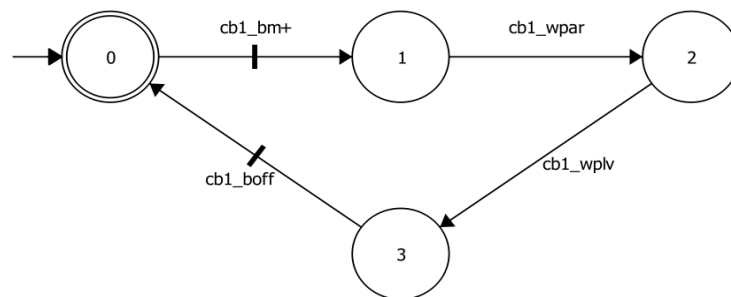


Figura 4.3 – Autômato esteira rolante 1

Fonte: Autor

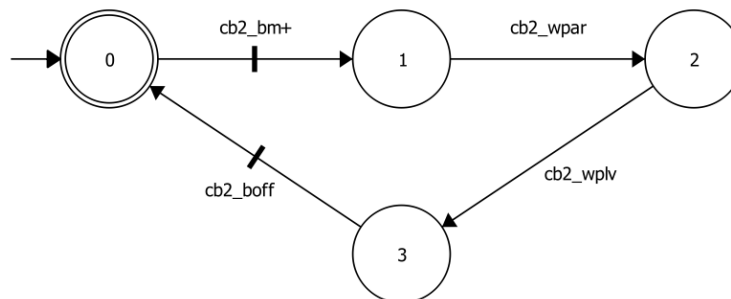


Figura 4.4 – Autômato esteira rolante 2

Fonte: Autor

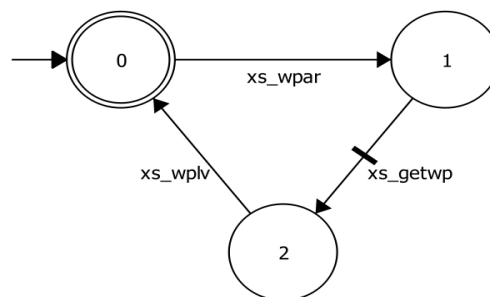


Figura 4.5 – Autômato armazenador/despachador

Fonte: Autor

4.1.2 Especificações de Operação:

Para iniciar o processo de geração dos supervisórios que controlarão a operação da planta, é preciso definir algumas especificações. Estas especificações detalham como as máquinas interagem entre si, definindo uma sequência de operações para um correto funcionamento da planta.

Especificação 1: A esteira rolante 1 deve ligar sua esteira quando o sinal de saída de peça do alimentador for recebido;

Especificação 2: A esteira rolante 2 deve ligar sua esteira quando o sinal de saída de peça da esteira rolante 1 for recebido;

Especificação 3: A esteira rolante 1 deve desligar sua esteira quando o sinal de chegada de peça da esteira rolante 2 for recebido.

Especificação 4: A esteira rolante 2 deve desligar sua esteira quando o sinal de chegada de peça do armazenador for recebido.

As representações de cada especificação em autômato podem ser vistas nas figuras 4.6 e 4.7.

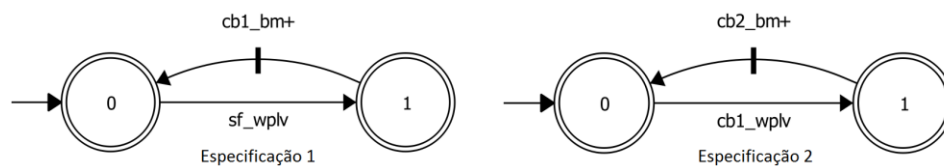


Figura 4.6 – Autômatos das especificações 1 e 2

Fonte: Autor

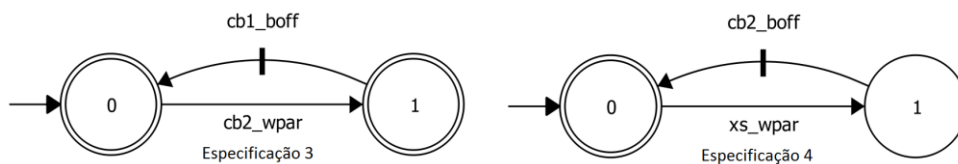


Figura 4.7 – Autômatos das especificações 3 e 4

Fonte: Autor

4.2 Cálculo dos Supervisórios

Como o controle supervisório utilizado neste trabalho segue o modelo Modular Local, é preciso gerar um supervisório para cada especificação definida para o sistema, levando em consideração apenas os modelos de máquinas que são “afetados” pela especificação utilizada. Assim, teremos 4 supervisórios para a mini planta industrial.

Para o cálculo do supervisório 1, precisamos realizar a operação de paralelo entre os autômatos do alimentador e da esteira rolante 1. Com o autômato resultante desta operação e a especificação 1 é, então, realizada a operação *SupConNB* (função disponível na ferramenta Destool que realiza o cálculo do supervisório). Esta operação gera a suprema

sublinguagem controlável, linguagem menos restritiva, associada ao módulo selecionado, no caso, o conjunto alimentador, esteira rolante 1 e especificação 1.

Para os outros supervisórios utiliza-se da mesma lógica, apenas alterando os autômatos necessários. Os supervisórios gerados desta forma apresentam tamanhos relativamente grandes em relação ao número de estados e transições, chegando facilmente às dezenas e centenas de unidades, respectivamente. Isso ocorre porque a operação *SupConNB* calcula todos os “caminhos” possíveis para os eventos associados a planta em questão.

Uma solução para este problema é utilizar a operação de redução de supervisórios (*SupReduce*). Ela baseia-se, entre outros métodos, na união de estados que apresentam uma mesma transição de chegada e formam um caminho fechado de transições entre si, por exemplo. O algoritmo para a operação de redução foi apresentado em [Su, 2003]. Realizando a operação de redução foram gerados os supervisórios reduzidos das figuras 4.8, 4.9 e 4.10.

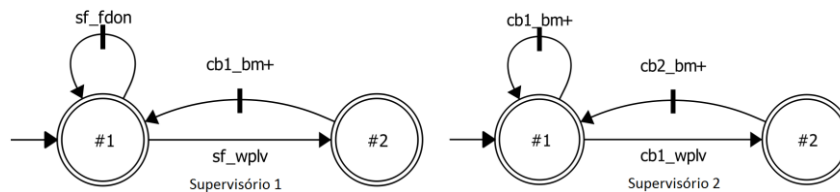


Figura 4.8 – Autômatos dos supervisórios 1 e 2

Fonte: Autor

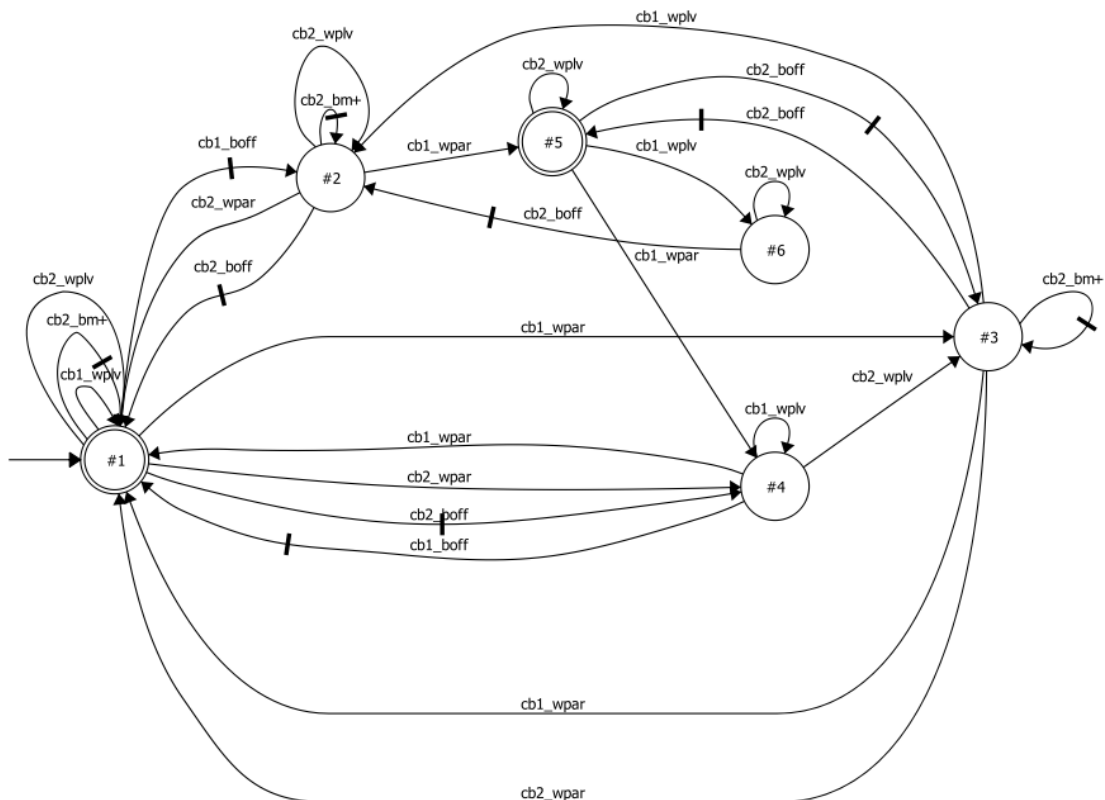


Figura 4.9 – Autômato supervisor 3

Fonte: Autor

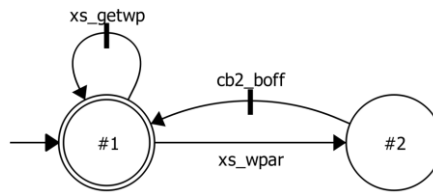


Figura 4.10 – Autômato supervisorio 4

Fonte: Autor

4.3 Conversão para Simulink

Com os supervisórios prontos, o próximo passo é utilizá-los no ambiente do Simulink e realizar as adaptações necessárias que possibilitem a geração de um código em C/C++ para posterior implementação em sistemas embarcados.

O Simulink possui um pacote chamado *Stateflow*, e com ele é possível elaborar máquinas de estados e tabelas de transições entre estados, a partir de relações de condições-consequências, [Stateflow, 2015a] e [Stateflow, 2015b]. É esta representação que foi utilizada para fazer a transição dos supervisórios calculados no Destool para o Simulink. Na Figura 4.11 temos a representação do supervisorio 1 em máquina de estados.

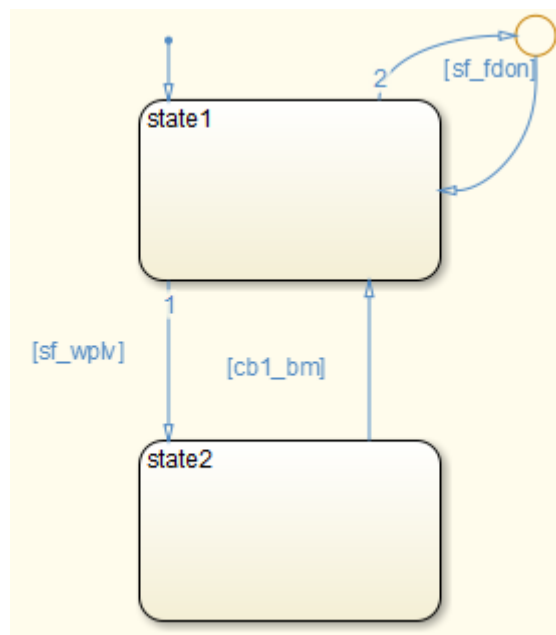


Figura 4.11 – Máquina de estados supervisorio 1

Fonte: Autor

Cada máquina de estado representa um supervisorio modular local com os mesmos estados e eventos anteriores. Neste momento já temos uma das vantagens de utilizarmos a representação dos supervisórios neste software. Isto ocorre pois, pela teoria de controle supervisorio já explanada, os supervisórios trabalham habilitando ou não os eventos de um sistema considerando que todos os eventos são gerados espontaneamente pela planta a ser controlada. No entanto, existem eventos como ligar e desligar esteiras que dependem de atuadores, na planta física, para ocorrerem, e os sinais de comando para estes atuadores podem ser modelados. Assim, associados aos eventos da Tabela 4.1 temos os sinais de

comando da Tabela 4.2 que serão acionados pelos supervisórios, quando da habilitação dos eventos a eles relacionados (máquina de estados mealy).

Desta forma, as máquinas de estados devem agora conter estes sinais de comando associados aos eventos controláveis, resultando, assim, em supervisórios como o mostrado na Figura 4.12.

Tabela 4.2 – Sinais de comando na planta

Fonte: Autor

Sinal de Comando	Evento Associado
H_cb1_bm+	cb1_bm+
H_cb1_boff	cb1_boff
H_cb2_bm+	cb2_bm+
H_cb2_boff	cb2_boff
H_sf_fdon	sf_fdon
H_sf_fdoff	sf_fdoff

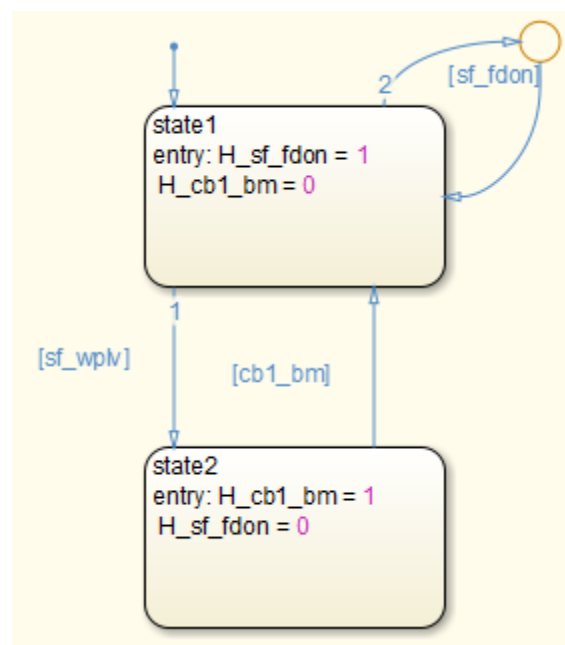


Figura 4.12 – Supervisório 1 com sinais de comando

Fonte: Autor

Porém, para que os supervisórios possam atuar de fato sobre uma planta real, utilizando-se sistemas embarcados, faz-se necessário adicionar uma interface entre os

supervisórios e os modelos de planta com o sistema físico. Esta ideia pode ser representada pela Figura 4.13.

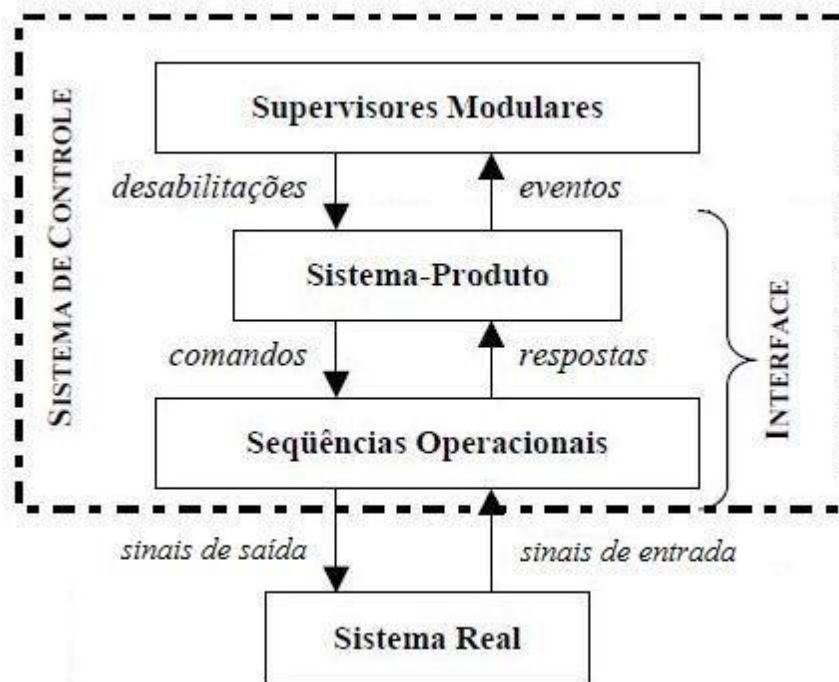


Figura 4.13 – Estrutura de controle

Fonte: Silva, Queiroz e Cury (2010)

A função desta interface é comandar o sistema real conforme os modelos abstratos das plantas e traduzir os comandos destes modelos em sinais de entrada do sistema real e as saídas do sistema real em respostas para os modelos abstratos, assim como proposto em [Queiroz e Cury, 2002]. Assim, faz-se necessário além das máquinas de estados que representam os supervisórios, máquinas representando os modelos abstratos de funcionamento de cada componente da mini planta industrial, gerando os sinais de comando para os atuadores no sistema físico.

Para que todos os supervisórios possam funcionar de forma paralela e controlar a planta corretamente, é preciso analisar os eventos controláveis que cada supervisório está habilitando em cada instante. Assim, somente serão habilitados os eventos que estiverem sendo permitidos por todos os supervisórios, nos quais existam os eventos em questão, simultaneamente. Assim, é necessário adicionar operações de comparação ao sistema, de forma que se relaciona o número de supervisórios que estão habilitando um evento em um instante t com o número de supervisórios que possuem este evento em seu escopo. Caso os números sejam iguais, o evento é habilitado para a planta. Para a esteira rolante 1 então, temos a interface apresentada na Figura 4.14.

Neste caso, como é possível observar nos autômatos das figuras 4.8 e 4.9, o evento H_cb1_bm está presente em dois supervisórios enquanto o evento H_cb1_boff em apenas um. Assim, para que o sistema de controle gere os sinais de comando para a planta, é preciso validar a comparação que representa o número de supervisórios que possuem os eventos em questão.

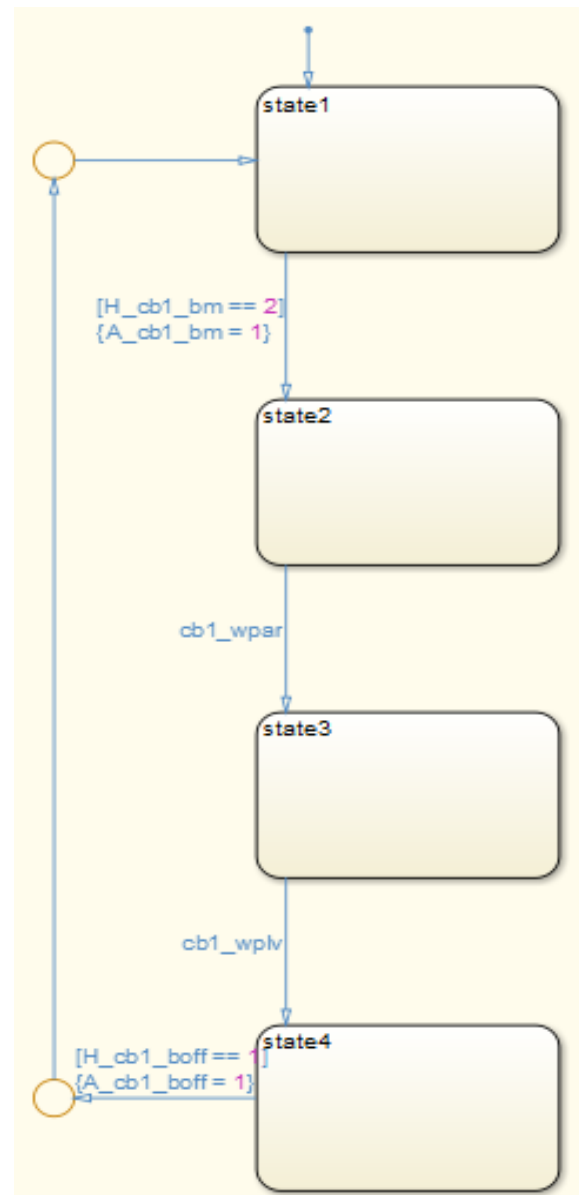


Figura 4.14 – Esteira rolante 1 com a avaliação da quantidade de habilitações dos eventos

Fonte: Autor

Agora, pode-se dizer que o modelo está completo e ao associar as entradas e saídas de cada máquina de estado, chega-se ao sistema da Figura 4.15, representando o controle sobre a esteira rolante 1. Nesta figura pode-se ver cada máquina de estados representada por um bloco do pacote *StateFlow*, chamado de *chart*. Nos blocos estão indicadas as entradas e saídas referente a cada máquina. Os eventos controláveis foram definidos como entradas do tipo dado, para que seja possível associá-los diretamente aos sinais de comando enviados à planta. Já os eventos observáveis foram definidos como variáveis do tipo evento mesmo, sendo detectados pela borda de subida (rising edge) do sinal.

A entrada de sinais do tipo evento nos blocos é dada em vetores, como pode ser visto na parte superior de cada bloco. É preciso verificar então, qual a numeração das portas de cada evento e ordená-los em um vetor na ordem correspondente. Isto é feito utilizando-se um bloco do Simulink chamado de *Mux*.

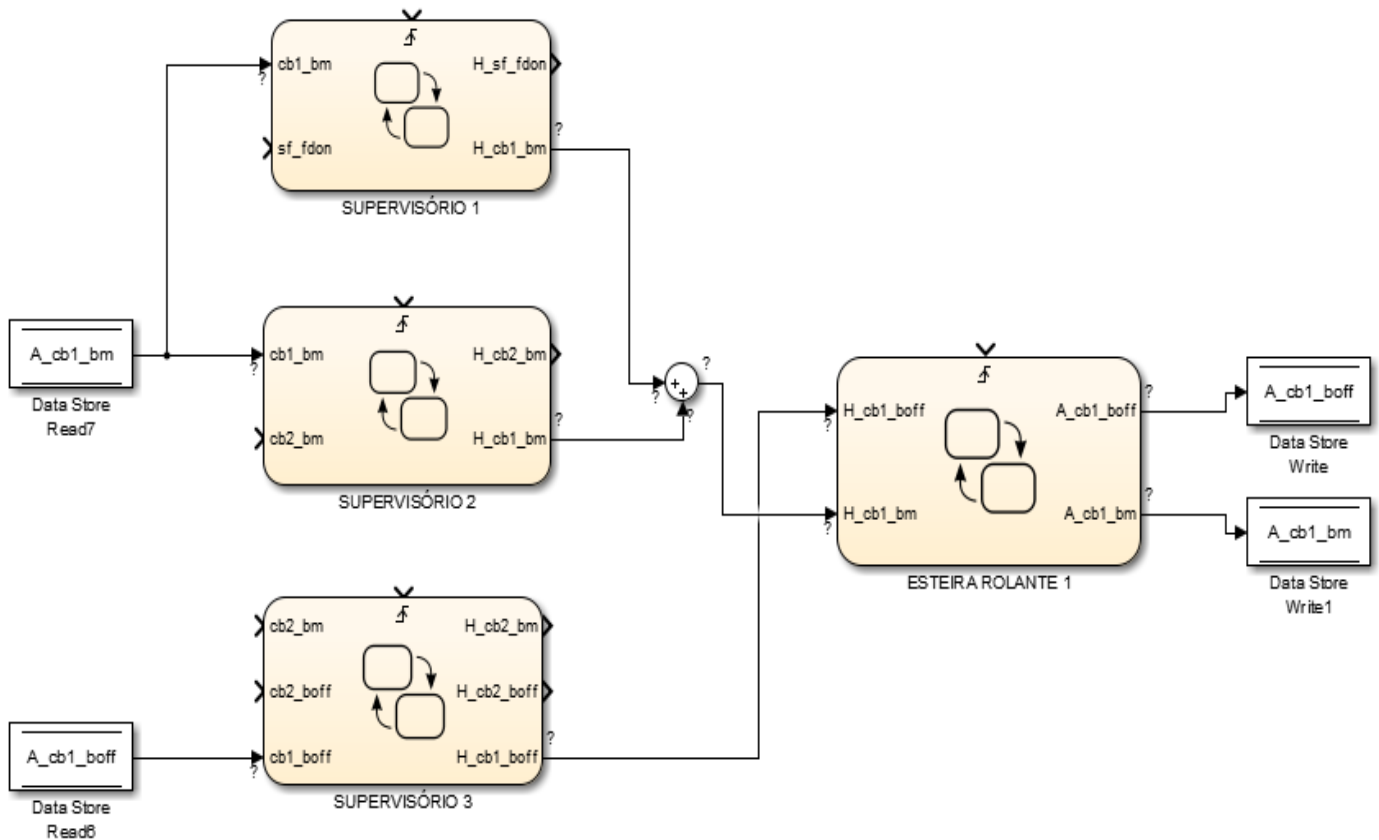


Figura 4.15 – Malha de controle esteira rolante 1

Fonte: Autor

Para a formação dos sistemas de controle dos outros componentes da planta, segue-se a mesma lógica apresentada na Figura 4.15, alterando-se apenas as entradas e saídas associadas a cada bloco.

4.4 Simulação

De forma a realizar simulações com os supervisórios desenvolvidos no ambiente Simulink, é preciso gerar os eventos observáveis através de blocos geradores de sinais, e então, levar os sinais de saída até blocos geradores de gráficos. Dessa forma, a malha está completa e pode-se acompanhar o desenvolvimento das máquinas de estado ao longo do tempo em função da ocorrência dos eventos gerados.

Além dos sinais de saída, é possível também acompanhar as mudanças de estado que ocorrem em cada supervisório, tanto em animações durante o intervalo de tempo da simulação quanto salvando todas as alterações em vetores, que podem ser analisados posteriormente no ambiente Matlab.

Com a adição dos blocos geradores de sinais e geradores de gráficos, obtém-se a malha apresentada na Figura 4.16, para a esteira rolante 1. Nesta figura pode-se observar a utilização de blocos de *step*, gerando sinais que variam de 0 para 1, ocorrência do evento, e de 0 para -1, anulação da ocorrência do evento. Assim, é possível simular os sinais pulsados que devem ser gerados pela planta real quando do funcionamento da mesma. As malhas dos outros componentes da planta são apresentadas no Apêndice deste trabalho.

Os sinais de comando que devem ir para os atuadores são conduzidos ao bloco gerador de gráficos, *scope*, para que a operação do sistema possa ser analisada posteriormente.

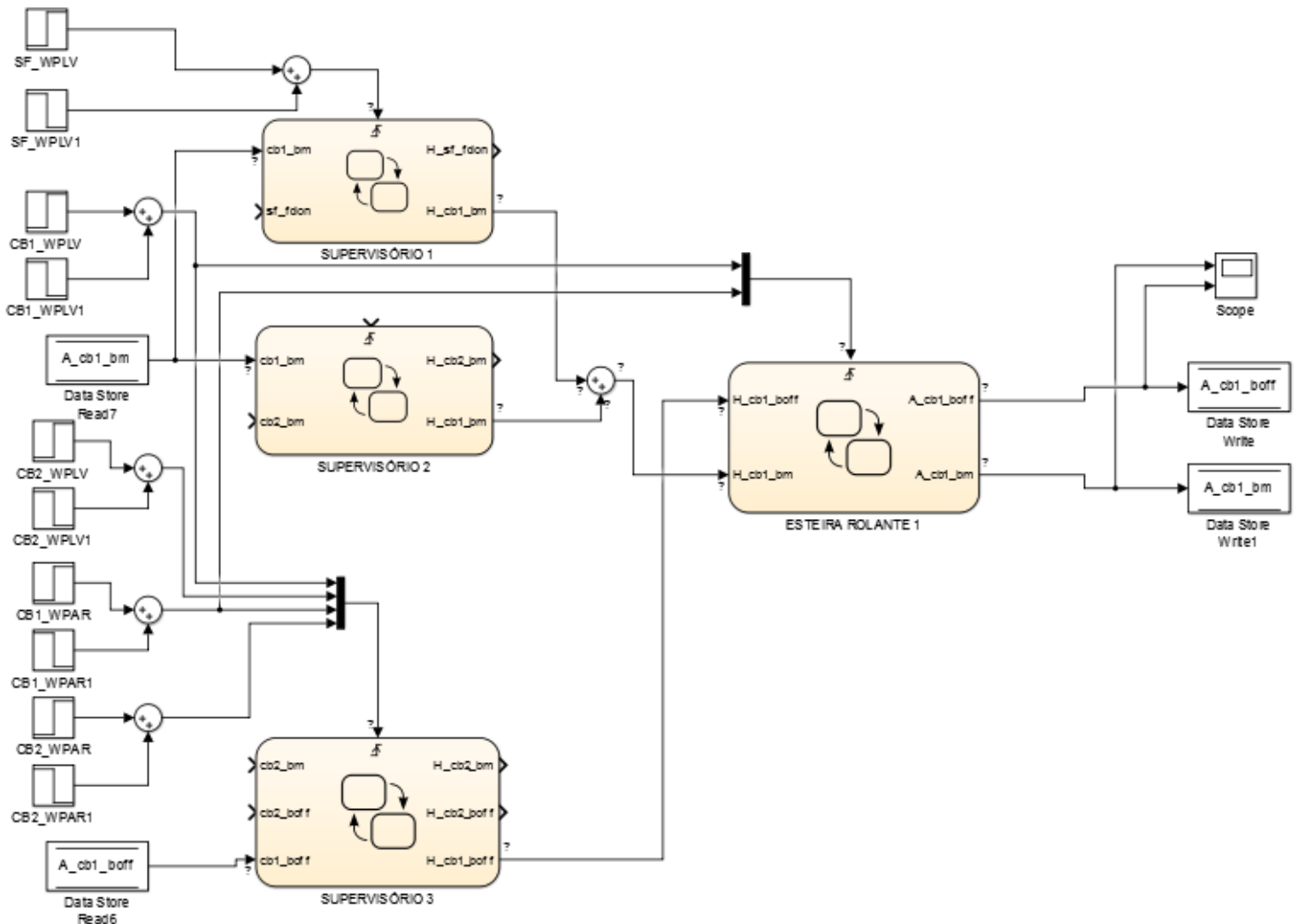


Figura 4.16 – Malha para simulação esteira rolante 1

Fonte: Autor

Durante a simulação, os supervisórios alteram seus estados de acordo com a ocorrência dos eventos associados a cada um. Essas mudanças podem ser acompanhadas durante o tempo de simulação. Outra forma de uso é utilizar, ao invés de simples eventos para comandar as transições de estados, diferentes níveis de um mesmo sinal como critério para as transições. Assim, pode-se expandir o uso desta ferramenta para autômatos híbridos também [Leal, 2005].

Além dos sinais pulsados é possível também utilizar diferentes tipos de sinais como rampa, trem de pulsos, senóides, etc. Propiciando diferentes formas de simulações, para diferentes modelos de plantas.

4.5 Geração de Código para Arduino

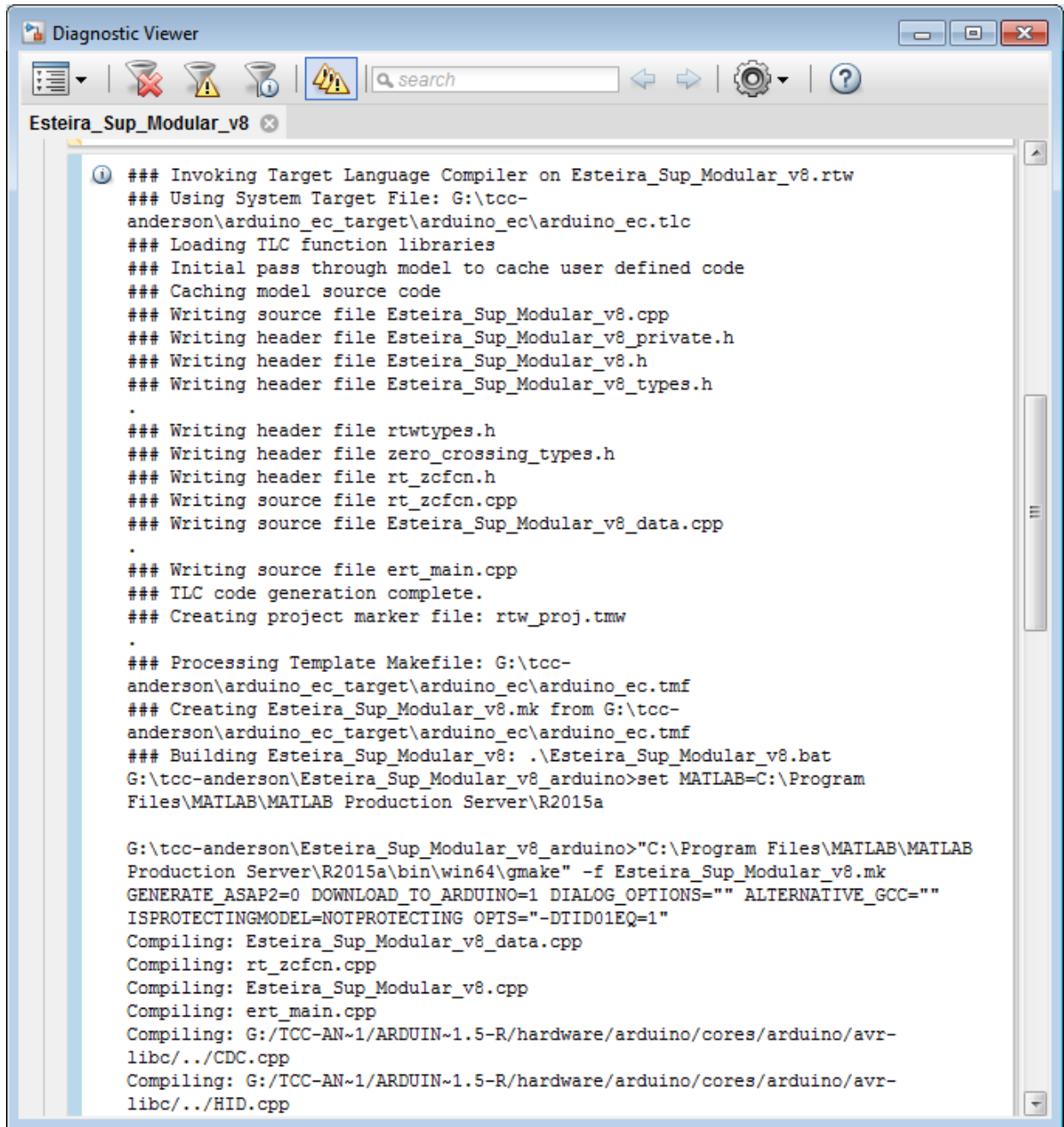
Após todas as simulações necessárias serem realizadas e verificar-se que o sistema está operando da forma desejada, pode-se então gerar o código para sistema embarcado, no caso, para o Arduino. Para isso, utilizaremos o arquivo de extensão *arduino EC target*.

Este arquivo possui um código em linguagem de Matlab que realiza as configurações iniciais no software para identificar o compilador específico para Arduino, bem como localizar a porta de comunicação em que a plataforma encontra-se conectada. Maiores detalhes sobre instalação e configuração deste arquivo são apresentadas em [*Embedded Coder Target for Arduino*, 2014].

Com estas configurações finalizadas, precisa-se apenas realizar a função de construção de sistema, *build system*, do Simulink, que irá construir todos os arquivos .h e .cpp em linguagem C/C++ necessários a utilização do modelo na plataforma Arduino. Após a montagem dos arquivos, o Simulink automaticamente realiza o download dos mesmos para o sistema embarcado.

É possível definir objetivos a serem atingidos na criação dos códigos como, eficiência em ROM, eficiência em RAM, *debugging*, etc. Dessa forma, é possível reduzir o tamanho dos arquivos gerados, liberando memória no microcontrolador e aumentando a eficiência na execução do programa.

O processo de criação dos códigos pode ser acompanhado pelo usuário através do visualizador de diagnósticos. É possível também analisar os arquivos gerados, com comentários sobre as funções e variáveis criadas para o programa, utilizando-se o relatório de geração de códigos, no próprio ambiente do Simulink. Nas figuras 4.17 e 4.18 são apresentadas amostras de como os diagnósticos e relatórios podem ser visualizados.



```

Diagnostic Viewer
Esteira_Sup_Modular_v8
### Invoking Target Language Compiler on Esteira_Sup_Modular_v8.rtw
### Using System Target File: G:\tcc-anderson\arduino_ec_target\arduino_ec\arduino_ec.tlc
### Loading TLC function libraries
### Initial pass through model to cache user defined code
### Caching model source code
### Writing source file Esteira_Sup_Modular_v8.cpp
### Writing header file Esteira_Sup_Modular_v8_private.h
### Writing header file Esteira_Sup_Modular_v8.h
### Writing header file Esteira_Sup_Modular_v8_types.h
.
### Writing header file rtwtypes.h
### Writing header file zero_crossing_types.h
### Writing header file rt_zcfcn.h
### Writing source file rt_zcfcn.cpp
### Writing source file Esteira_Sup_Modular_v8_data.cpp
.
### Writing source file ert_main.cpp
### TLC code generation complete.
### Creating project marker file: rtw_proj.tmw
.
### Processing Template Makefile: G:\tcc-anderson\arduino_ec_target\arduino_ec\arduino_ec.tmf
### Creating Esteira_Sup_Modular_v8.mk from G:\tcc-anderson\arduino_ec_target\arduino_ec\arduino_ec.tmf
### Building Esteira_Sup_Modular_v8: .\Esteira_Sup_Modular_v8.bat
G:\tcc-anderson\Esteira_Sup_Modular_v8_arduino>set MATLAB=C:\Program
Files\MATLAB\MATLAB Production Server\R2015a

G:\tcc-anderson\Esteira_Sup_Modular_v8_arduino>"C:\Program Files\MATLAB\MATLAB
Production Server\R2015a\bin\win64\gmake" -f Esteira_Sup_Modular_v8.mk
GENERATE_ASAP2=0 DOWNLOAD_TO_ARDUINO=1 DIALOG_OPTIONS="" ALTERNATIVE_GCC=""
ISPROTECTINGMODEL=NOTPROTECTING OPTS="-DTID01EQ=1"
Compiling: Esteira_Sup_Modular_v8_data.cpp
Compiling: rt_zcfcn.cpp
Compiling: Esteira_Sup_Modular_v8.cpp
Compiling: ert_main.cpp
Compiling: G:/TCC-AN~1/ARDUIN~1.5-R/hardware/arduino/cores/arduino/avr-
libc/./CDC.cpp
Compiling: G:/TCC-AN~1/ARDUIN~1.5-R/hardware/arduino/cores/arduino/avr-
libc/./HID.cpp

```

Figura 4.17 – Diagnóstico de geração de códigos

Fonte: Autor

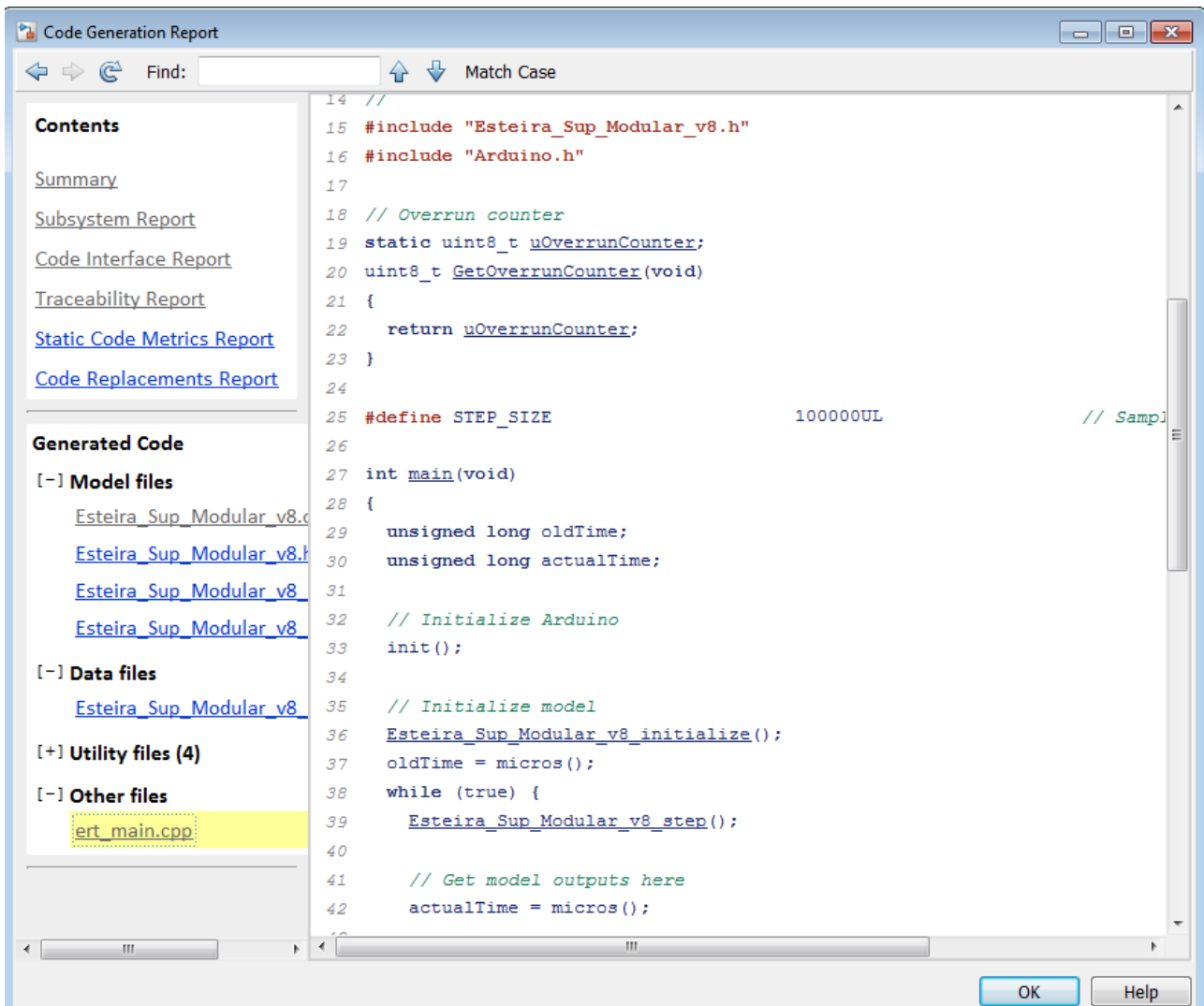


Figura 4.18 – Relatório com os códigos gerados

Fonte: Autor

5 Resultados

Ao utilizar o modo de simulação do Simulink, é possível acompanhar a ocorrência dos eventos e dos sinais de comando ao longo do tempo. Durante a simulação não há comunicação dos modelos com sistemas externos, logo, os eventos precisam ser gerados internamente. Assim, foram utilizados blocos *step*, que variam seu valor de 0 a 1, atuando em diferentes instantes de tempo, simulando assim, a sequência em que devem ocorrer os eventos da planta, quando a mesma estiver em operação.

A ordem e os instantes de tempo em que ocorrem os pré-definidos eventos são descritos na Tabela 5.1. Os eventos controláveis ocorrem de acordo com o controle dos supervisórios, dessa forma, devem atender às especificações definidas inicialmente para o sistema. As respostas dos sinais de comando, geradas pelo controle supervisório, podem ser vistas nas figuras 5.1, 5.2 e 5.3.

Tabela 5.1 – Instantes de tempo em que ocorrem os eventos

Fonte: Autor

Eventos	Tempo da Ocorrência (s)
sf_wpar	1
sf_wplv	2
sf_fdhome	3
cb1_wpar	4
cb1_wplv	5
cb2_wpar	6
cb2_wplv	7
xs_wpar	8
xs_wplv	9

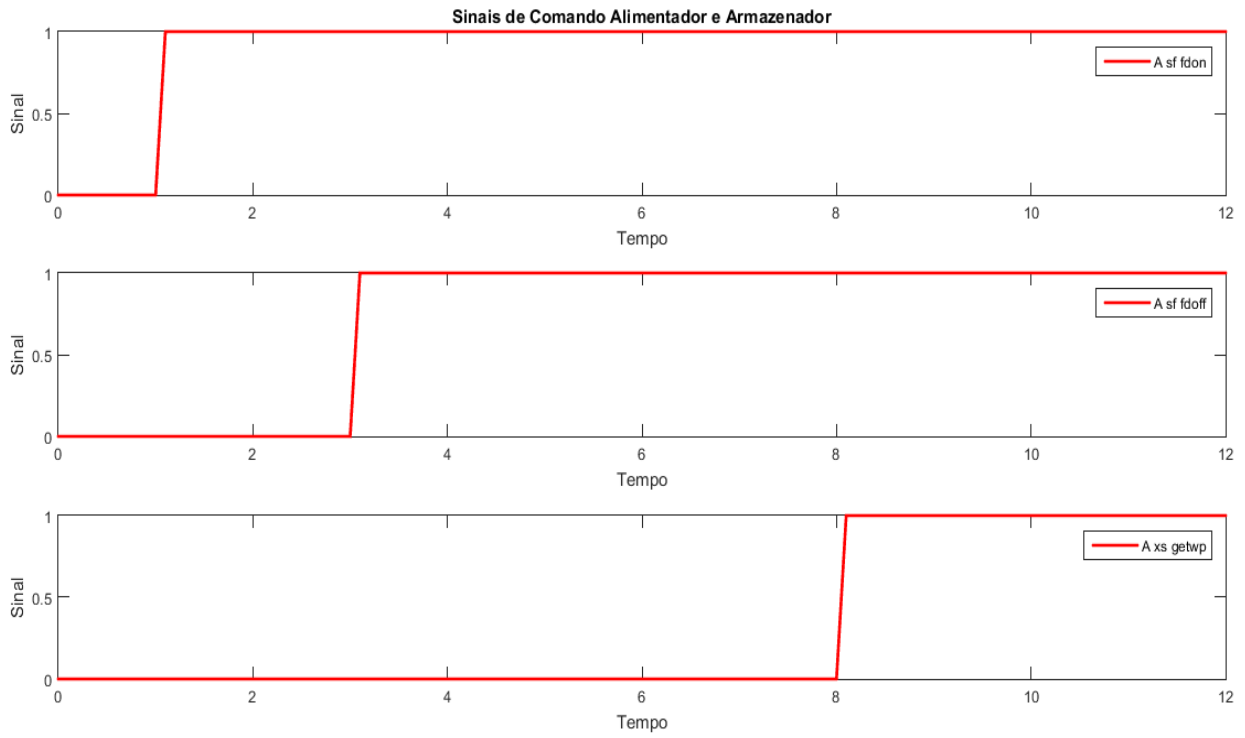


Figura 5.1 – Sinais de comando alimentador e armazenador

Fonte: Autor

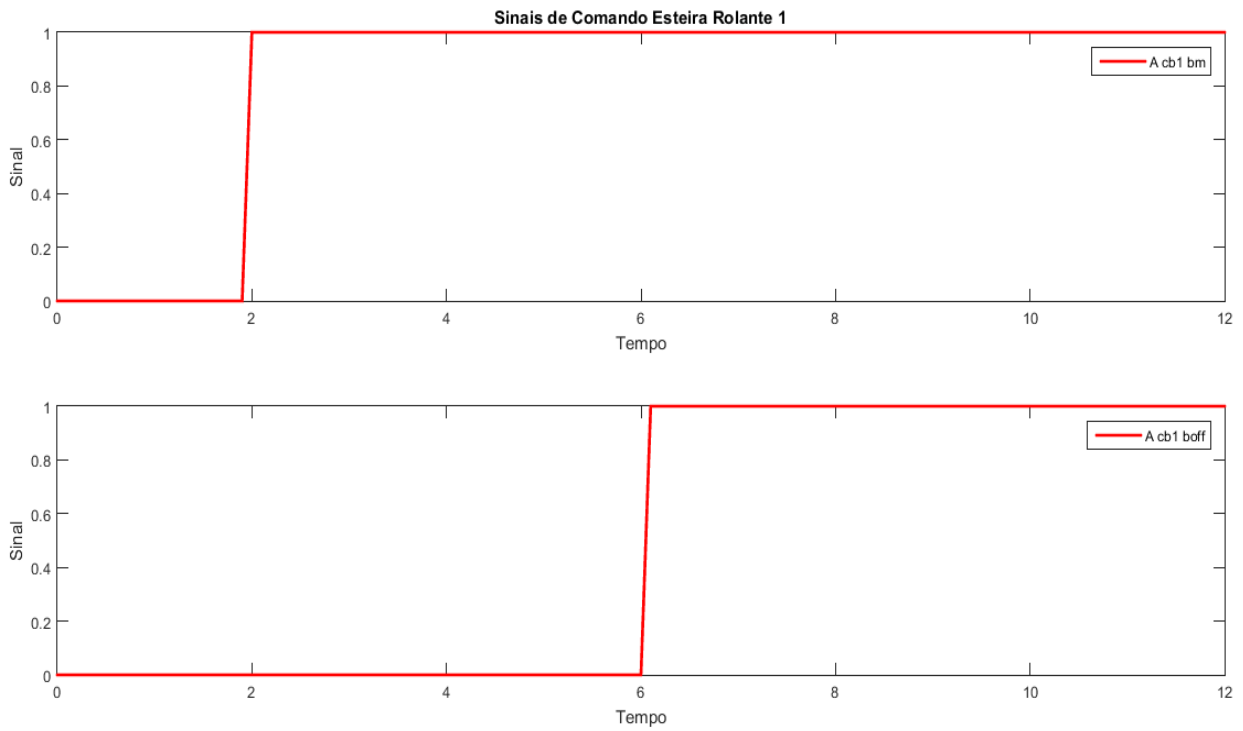


Figura 5.2 – Sinais de comando esteira rolante 1

Fonte: Autor

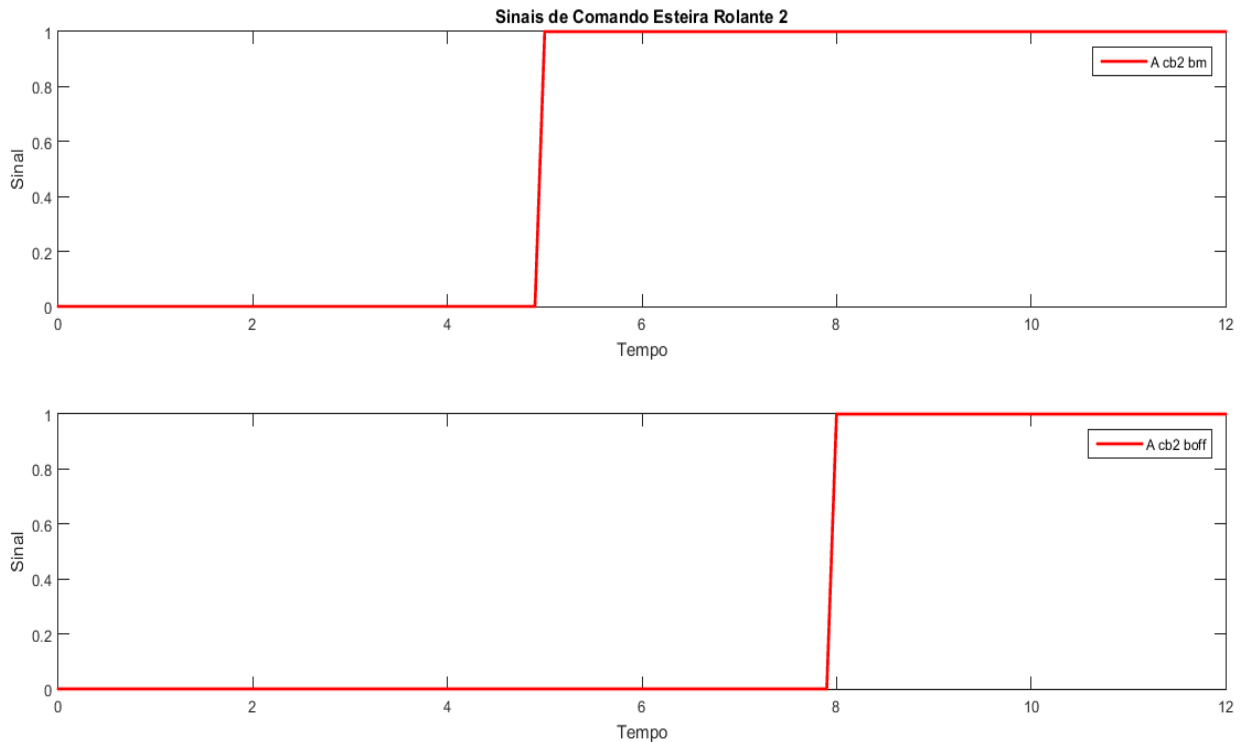


Figura 5.3 – Sinais de comando esteira rolante 2

Fonte: Autor

Nestes gráficos percebe-se que os sinais de comando ocorrem quando são habilitados pelo supervisor e mantém seu valor. Porém, a forma de acionamento destes comandos pode ser alterada, caso queira dar-se um sentido mais prático para os mesmos, algo importante quando os modelos forem aplicados em um sistema embarcado para controlar uma planta real.

Assim, foi definido para o alimentador que sua esteira é acionada quando o sinal A_sf_don apresentar valor 1 e que, para desligar a mesma, é necessário dar um pulso no sinal A_sf_doff , como um sistema de desarme. A resposta é apresentada na Figura 5.4.

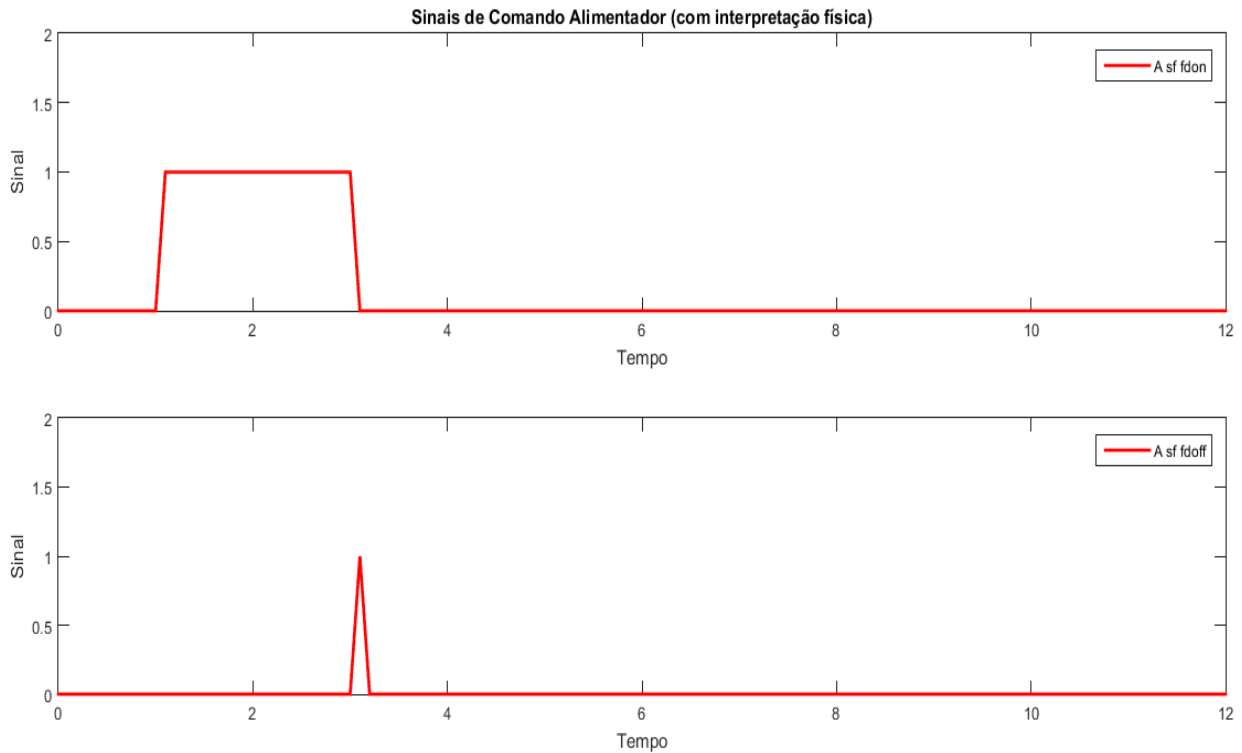


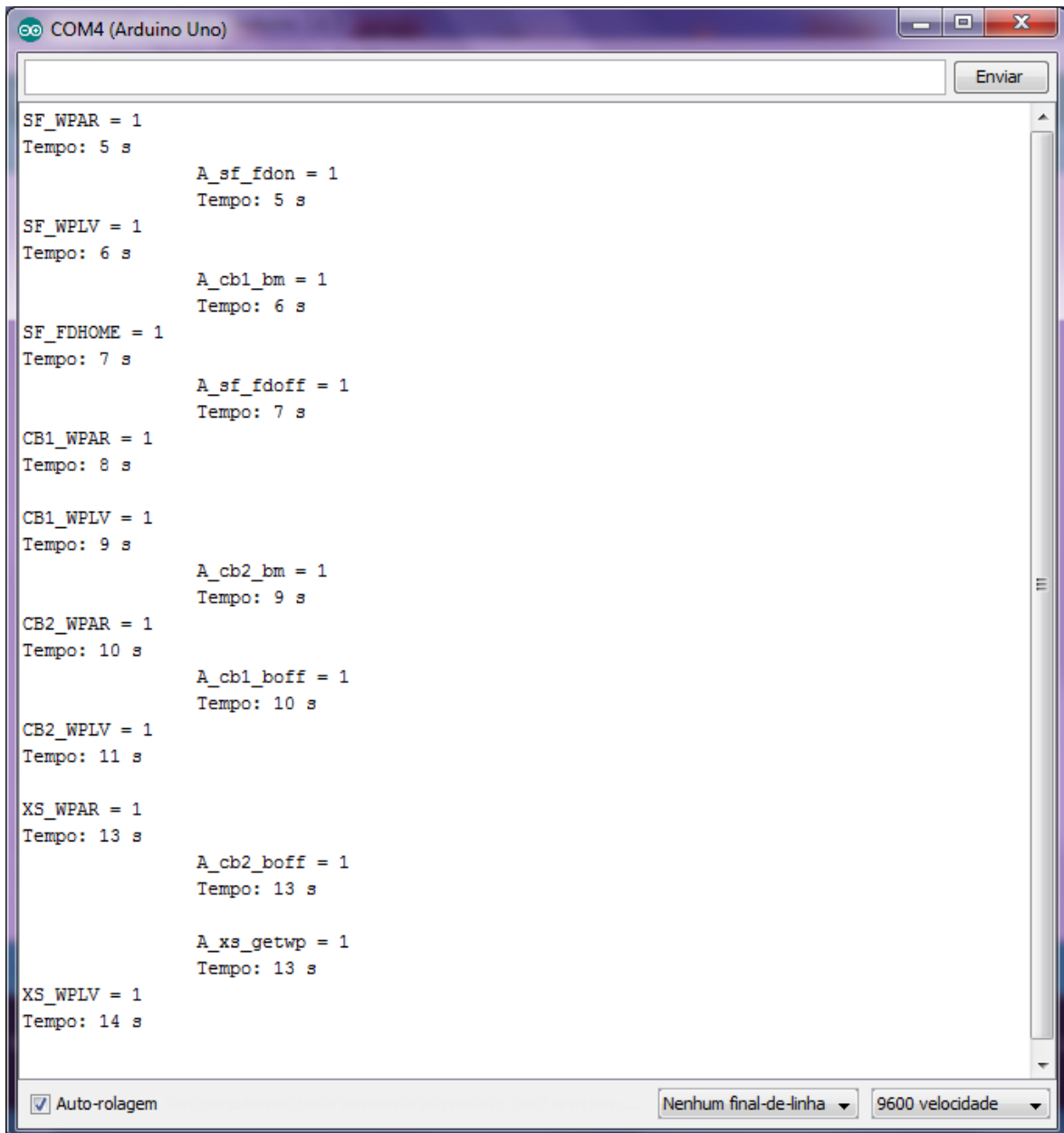
Figura 5.4 – Sinais de comando com interpretação física

Fonte: Autor

Ao fim das simulações foram gerados os códigos para o Arduino, e então os mesmos foram testados utilizando-se as portas digitais da plataforma. Cada porta de entrada representa um evento observável, quando o valor de uma porta está em “alta” (5 V) significa que o evento está ocorrendo, caso contrário, ele não ocorre.

Para representar os sinais de comando destinados aos atuadores da planta, foram utilizadas portas digitais também, porém, no modo de saída. A essas portas foram conectados LED's coloridos para auxiliar na visualização da ocorrência destes eventos.

Durante os testes foi utilizada também a comunicação serial do Arduino, assim, era possível monitorar as ocorrências dos eventos pelo monitor serial do mesmo. As respostas apresentadas pelo sistema embarcado ficaram em conformidade com as obtidas nas simulações, o que pode ser visualizado na Figura 5.5.



```
COM4 (Arduino Uno)
Enviar

SF_WPAR = 1
Tempo: 5 s
      A_sf_fdon = 1
      Tempo: 5 s

SF_WPLV = 1
Tempo: 6 s
      A_cb1_bm = 1
      Tempo: 6 s

SF_FDHOME = 1
Tempo: 7 s
      A_sf_fdoff = 1
      Tempo: 7 s

CB1_WPAR = 1
Tempo: 8 s

CB1_WPLV = 1
Tempo: 9 s
      A_cb2_bm = 1
      Tempo: 9 s

CB2_WPAR = 1
Tempo: 10 s
      A_cb1_boff = 1
      Tempo: 10 s

CB2_WPLV = 1
Tempo: 11 s

XS_WPAR = 1
Tempo: 13 s
      A_cb2_boff = 1
      Tempo: 13 s

      A_xs_getwp = 1
      Tempo: 13 s

XS_WPLV = 1
Tempo: 14 s
```

Auto-rolagem Nenhum final-de-linha 9600 velocidade

Figura 5.5 – Monitor serial Arduino

Fonte: Autor

6 Conclusões e Trabalhos Futuros

Com a utilização do Matlab/Simulink, a simulação dos supervisórios e a geração de códigos para sistemas embarcados tornou-se algo prático, capaz de apresentar e identificar erros no projeto de forma mais rápida, algo que até então representava um tempo considerável, principalmente na identificação de quais eram os erros de fato.

Os resultados atingidos apresentam-se de forma satisfatória. É possível simular o funcionamento dos supervisórios projetados, de forma rápida e direta, gerando-se gráficos que indicam as ocorrências ou não de cada um dos possíveis eventos.

A geração do código para sistemas embarcados também se apresentou como uma operação rápida e automatizada. Os testes com a plataforma Arduino Uno corresponderam às expectativas, apresentando respostas semelhantes às simulações.

O cálculo do supervisório ainda depende do software Destool, uma vez que o Simulink não apresenta nenhum pacote que trabalhe com os conceitos da teoria de controle supervisório de forma direta. Como ainda não é possível realizar o cálculo dos supervisórios diretamente no Matlab, a transferência dos mesmos de um software para outro ainda é um trabalho manual.

Em trabalhos futuros seria interessante explorar mais a modelagem de SED's no Matlab/Simulink, utilizando autômatos híbridos [Villani, Yen-Tsang e Miyagi, 2002], por exemplo. Estudar formas de realizar as etapas de projeto e síntese de supervisório diretamente no Matlab como em [Zad, 2013], e associar estes supervisórios com autômatos, modelados em máquinas de estado, no Simulink. É possível utilizar comunicação Modbus entre o Arduino e o software FlexFact [Pereira, 2014], geralmente utilizado em conjunto com o Destool, para visualização do desempenho dos códigos gerados em uma interface mais “amigável”.

Outra abordagem possível também seria realizar o projeto dos supervisórios em ferramentas externas ao Matlab e gerar um arquivo que armazene essas informações e ao mesmo tempo possua uma linguagem compreensível pelo Matlab, de forma que o mesmo, possa abrir o arquivo diretamente, sem necessidade de conversões, [Meeusen, 2012] e [van de Mortel-Fronczak et al., 2014]. Isto aumentaria significativamente as vantagens da utilização destes ambientes, largamente utilizados no meio acadêmico e industrial, e descartaria a necessidade de transferir informações “manualmente” entre diferentes ferramentas.

7 Referências

Arduino. **Arduino Uno**. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoBoardUno>>. Acesso em: 12 set. 2015.

Cassandras, C. G. e Lafortune, S. (2008) **Introduction to discrete event systems**, Springer, 2ª edição, ISBN 978-0-387-33332-8.

Cunha, A. E. C. (2003) **Controle supervísório de SEDs: Introdução**, Apostila, Instituto Militar de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica.

Cury, J. E. R. (2001) **Teoria de controle supervísório de sistemas a eventos discretos**, V Simpósio Brasileiro de Automação Inteligente, Canela-RS.

Embedded Coder Target for Arduino. **Arduino EC Target**. Disponível em: <<http://www.mathworks.com/matlabcentral/fileexchange/30277-embedded-coder-target-for-arduino>>. Acesso em: 22 set. 2015.

FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG. **Destool**. Disponível em: <<http://www.rt.techfak.fau.de/FGdes/destool/>>. Acesso em: 12 set. 2015.

Götz, M. (2014) **Apostila de sistemas a eventos discretos, autômatos & teoria do controle supervísório**, Apostila, UFRGS, Departamento de Engenharia Elétrica.

Leal, A. B. (2005) **Controle supervísório modular de sistemas híbridos**, Tese de Doutorado em Engenharia Elétrica, UFSC.

Matlab & Simulink (2015) **Stateflow: getting started guide**. Disponível em: <<https://www.mathworks.com>>. Acesso em: 15 set. 2015.

Matlab & Simulink (2015) **Stateflow: user's guide**. Disponível em: <<https://www.mathworks.com>>. Acesso em: 15 set. 2015.

MECATRÔNICA Atual. **Supervisão de sistemas a eventos discretos – Parte 1**. Disponível em: <<http://www.mecatronicaatual.com.br/educacao/790-superviso-de-sistemas-a-eventos-discretos-parte-1?showall=&limitstart=>>>. Acesso em: 10 set. 2015.

Meeusen, K. A. (2012) **Application of supervisor synthesis to the design of cruise control**, Projeto final de Bacharelado, Eindhoven University of Technology.

Pereira, F. A. (2014) **Infraestrutura para implementação em linguagem C de controladores supervísórios para plantas de manufatura virtuais**, Trabalho de conclusão de curso em Engenharia de controle e automação, UFRGS.

Queiroz, M. H. e Cury, J. E. R. (2002) **Synthesis and implementation of local modular supervisory control for a manufacturing cell**. VI Workshop Internacional em Sistemas a Eventos Discretos, IEEE, ISBN 0-7695-1683-1

Ramos, M. V. M. (2008) **Linguagens formais e autômatos**, Apostila, Universidade Federal do Vale do São Francisco, Curso de Engenharia de Computação.

Silva, G. Y., Queiroz, M. H. e Cury, J. E. R. (2010) **Síntese e implementação de controle supervisório modular local para um sistema de AGV**. XVIII Congresso Brasileiro de Automática, CBA, Bonito-MS.

Su, R. e Wonham, W. M. (2003) **Supervisor reduction for discrete-event systems**, Artigo, Sistemas Dinâmicos a Eventos Discretos: Teoria e Aplicação, Kluwer Academic Publishers.

van de Mortel-Fronczak, J. M. et al. (2014) **Supervisor synthesis in model-based automotive systems engineering**, Artigo, ICCPS, Berlim-Alemanha, IEEE, ISBN 978-1-4799-4930-4/14.

Villani, E., Yen-Tsang, C. e Miyagi, P. E. (2002) **Simulação de sistemas híbridos usando Matlab/Simulink**. II Congresso Nacional de Engenharia Mecânica (CONEM), João Pessoa-PB

Zad, S. H. (2013) **Discrete event control kit (DECK) – User manual**, Concordia University, Departamento de Engenharia Elétrica e de Computação.

8 Apêndice

Na sequência são apresentadas as malhas de controle formadas para simulação, de cada componente da planta, no ambiente Simulink:

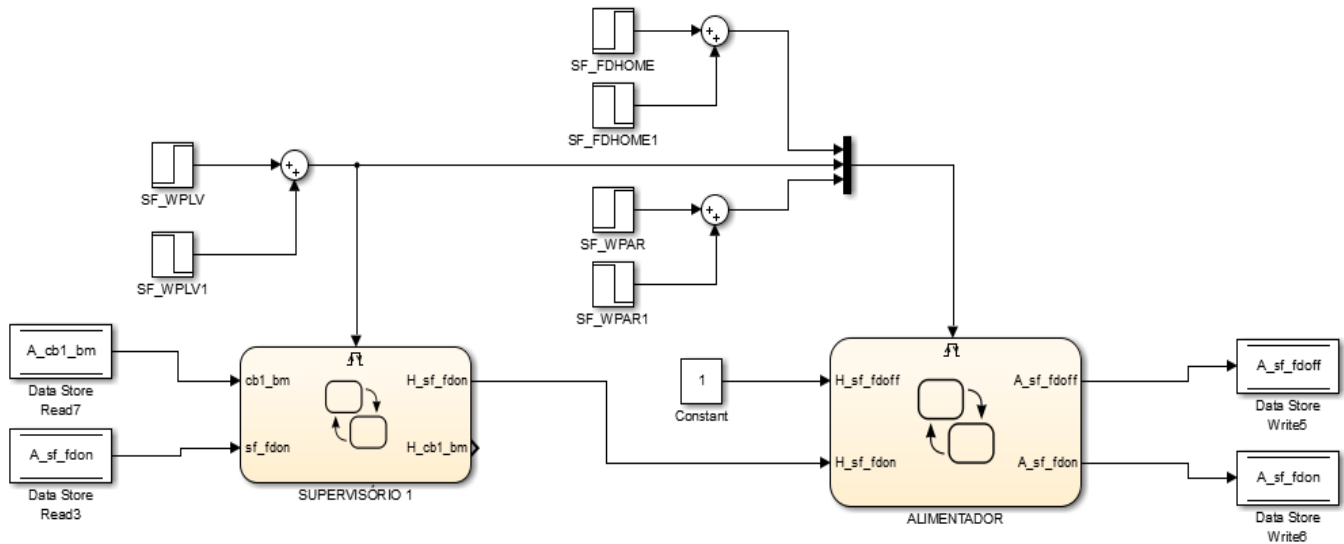


Figura 8.1 - Malha de controle alimentador

Fonte: Autor

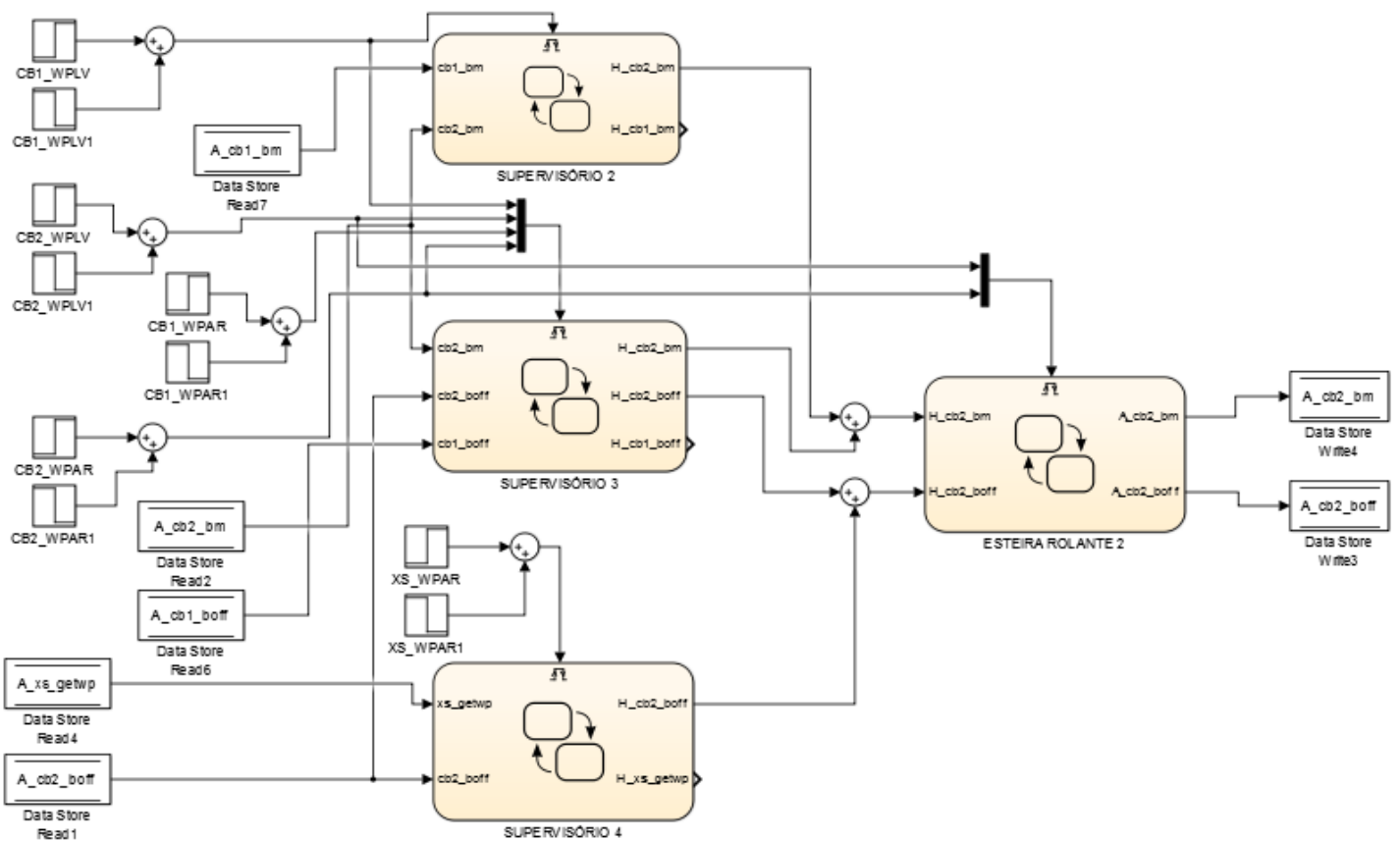


Figura 8.2 - Malha de controle esteira rolante 2

Fonte: Autor

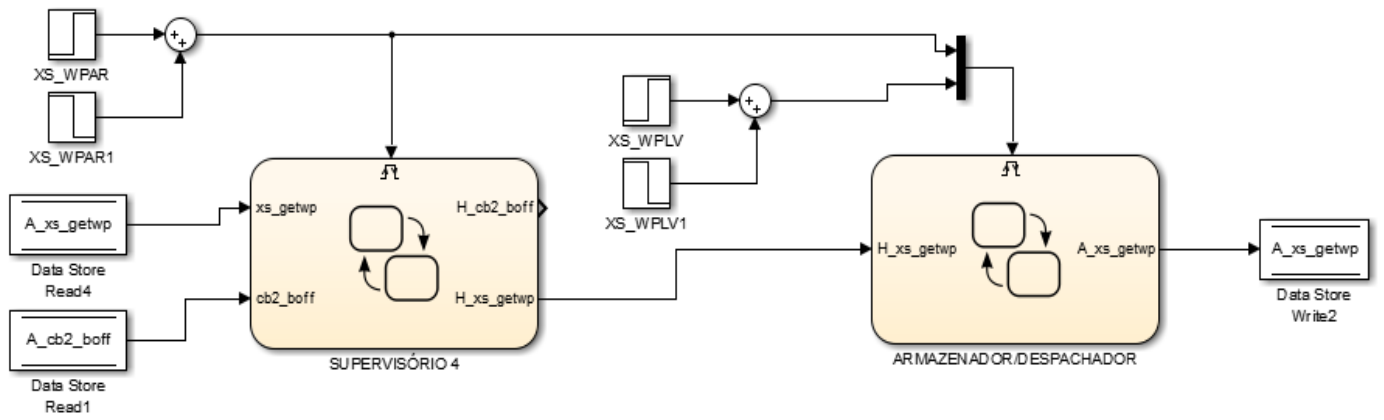


Figura 8.3 - Malha de controle armazenador

Fonte: Autor