Implementação em Arduino de uma Solução de Controle Supervisório em um Protótipo de Célula de Manufatura usando UltraDES

Hanna S. Vilela * **Lucas E. L. Costa** * **Matheus P. Loures** ** **Daniel Sarsur** ** **Lucas V. R. Alves** *** **Patrícia N. Pena** ****

* *Graduação em Engenharia de Controle e Automação, Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Minas Gerais (e-mail: hvilela@ufmg.br, lucaselc@ufmg.br).*
** *Programa de Pós-Graduação em Engenharia Elétrica, UFMG (e-mail: mploures@ufmg.br, danielsc@ufmg.br).*
*** *Colégio Técnico, UFMG (e-mail: lucasvra@ufmg.br).*
**** *Departamento de Engenharia Eletrônica, UFMG (e-mail: ppena@ufmg.br).*

Abstract: This paper presents a case study of an implementation of a supervisory control solution in Arduino, using UltraDES. Supervisory Control Theory is applied to carry out the logical control of the system, providing security guarantees and correct operation. Local modular control is used and the arduino coding is simplified with a tool integrated with the UltraDES library. Details of the prototype are also presented.

Resumo: O presente artigo apresenta um estudo de caso de uma implementação de solução de controle supervisório em arduino, usando o UltraDES. A Teoria de Controle Supervisório é aplicada para realizar o controle lógico do sistema, visando dar garantias de segurança e o correto funcionamento. O controle modular local é utilizado e a geração do código de programação é simplificada com uma ferramenta integrada com a biblioteca UltraDES. Detalhes do protótipo também são apresentados.

Keywords: Discrete event systems, Supervisory control theory, Embedded Systems, Engineering education

Palavras-chaves: Sistemas de eventos discretos, Teoria de controle supervisório, Sistemas Embarcados, Ensino de Engenharia

1. INTRODUÇÃO

A evolução da tecnologia tem impulsionado avanços significativos na área de sistemas embarcados, permitindo soluções cada vez mais sofisticadas em diversas aplicações. Embora a abordagem tradicional de programação seja comumente utilizada dentro desse contexto, algumas dificuldades podem ser identificadas no que diz respeito à escalabilidade ou garantias do correto funcionamento do sistema.

Este artigo apresenta um estudo de caso detalhado sobre a implementação de uma solução a partir da Teoria de Controle Supervisório (TCS) (Ramadge e Wonham, 1989) para realizar o controle lógico de uma célula de manufatura, garantindo segurança e funcionamento correto. A abordagem utilizando a TCS permite uma modelagem mais intuitiva e modularizada tanto dos equipamentos quanto das restrições às quais eles estão sujeitos. Por esse motivo, também, torna-se mais fácil a manutenção do modelo em caso de necessidade de reconfiguração da planta.

Utiliza-se como estudo de caso uma adaptação da célula de manufatura proposta por (De Queiroz e Cury, 2002), cujos detalhes estão apresentados na Seção 3. Um Arduino Nano é utilizado para controle do sistema, que foi construído utilizando motores de passo, LEDs, peças fabricadas por meio de manufatura aditiva e outros componentes.

Um conjunto de autômatos é obtido para representar o comportamento dos equipamentos e especificações, e a solução do problema de controle é obtida pela aplicação da TCS. Posteriormente, utiliza-se a ferramenta de geração automática de código de programação para placas de desenvolvimento (Loures, 2024), implementada de forma integrada à biblioteca de modelagem e análise de sistemas à eventos discretos UltraDES (Alves e Pena, 2023). A partir do código gerado, para cada estado do controlador, são atribuídos os sinais lógicos que atuam sobre os equipamentos da planta. Este trabalho visa, portanto, validar a ferramenta desenvolvida, pela geração de código, implementação e teste no protótipo.

É importante destacar a relevância do uso de plantas didáticas, que são sistemas ou ambientes simulados que reproduzem aspectos reais de uma indústria ou processo, no ensino de engenharia como uma estratégia educacional eficaz para complementar o aprendizado teórico em sala de aula (Freitas, 2023). Esses recursos proporcionam aos alunos uma experiência prática e imersiva, sendo um complemento valioso para a formação de estudantes de engenharia, principalmente em um cenário de rápido avanço tecnológico, devendo, sempre que possível, ser integrados à grade curricular (Cavalcante et al., 2020).

Na Seção 2, conceitos preliminares são apresentados para melhor compreensão deste trabalho. A Seção 3 descreve o funcionamento e a construção da célula de manufatura. A Seção 4 apresenta a modelagem das plantas e especificações, geração

da estrutura de controle e a implementação do código. Ao final, na Seção 5, são apresentadas as conclusões do artigo.

2. PRELIMINARES

Nesta seção são abordados conceitos fundamentais de Sistemas a Eventos Discretos, Teoria do Controle Supervisório e implementação.

2.1 Sistemas a Eventos Discretos

Sistemas a Eventos Discretos (SED) são definidos como sistemas dinâmicos, compostos por um conjunto de estados e suas respectivas transições e a evolução dos estados internos deste sistema é determinada pela ocorrência de eventos. Estes eventos, que se manifestam em intervalos irregulares ou indeterminados, são os principais responsáveis pela progressão dos estados do sistema (Cassandras e Lafortune, 2008).

Nesta abordagem, os eventos são definidos como quaisquer ocorrências que se manifestam no sistema e que têm o potencial de indicar o início ou o término de uma tarefa específica, ou algum evento interno do sistema. O conjunto de eventos (ou alfabeto) é denotado por Σ. Exemplos cotidianos de eventos incluem ações como ativar ou desativar um motor elétrico, iniciar ou concluir a operação de uma máquina, e até mesmo o ato de um usuário pressionar um botão.

As sequências finitas de símbolos são chamadas de cadeias, sendo ε a cadeia com nenhum símbolo e o conjunto de todas as cadeias associadas a um alfabeto é denotado por Σ^*. Sejam as cadeias $s,t,u \in \Sigma^*$. A cadeia t é prefixo de s se existe u tal que $s = tu$. Qualquer subconjunto de Σ^* é chamado de linguagem. Linguagens podem ser finitas ou infinitas e uma forma fechada de representar linguagens é por meio de autômatos.

Definição 1. Um autômato finito determinístico é definido por uma quíntupla $G = (Q, \Sigma, \delta, q_0, Q_m)$. Em que Q é um conjunto finito de estados, Σ é o alfabeto, δ ($\delta : Q \times \Sigma \to Q$) é a função de transição de estados, $q_0 \in Q$ é o estado inicial e $Q_m \subseteq Q$ é o conjunto de estados marcados de G.

A função de transição δ pode ser estendida para cadeias de eventos com a função $\hat{\delta} : Q \times \Sigma^* \to Q$, que mapeia para qual estado de Q o sistema transita a partir do estado q com a ocorrência de uma sequência $s \in \Sigma^*$. O autômato tem a ele associadas duas linguagens: a linguagem gerada composta de todas as cadeias fisicamente possíveis, $\mathscr{L}(G) = \{s \in \Sigma^* | \hat{\delta}(q_0, s) \in Q\}$, e a linguagem marcada composta de todas as cadeias de $\mathscr{L}(G)$ que terminam em estados marcados, $\mathscr{L}_m(G) = \{s \in \mathscr{L}(G) | \hat{\delta}(q_0, s) \in Q_m\}$.

Em relação às linguagens, existem algumas operações que podem ser feitas sobre as mesmas. Umas delas, que será importante para os tópicos abordados em sequência, é o prefixo-fechamento. O prefixo-fechamento de uma linguagem A é denotado por $\bar{A}$ e consiste do conjunto de prefixos das cadeias da linguagem.

Prefixo-fechamento: $A = \{0, 11\}$: $\overline{A} = \{\varepsilon, 0, 1, 11\}$.

2.2 Teoria de Controle Supervisório

Por meio da Teoria de Controle Supervisório, é possível obter um supervisor a partir das plantas e especificações. Este supervisor modela o comportamento em malha fechada do sistema controlado e atua desabilitando os eventos que comprometem o funcionamento correto da planta, garantindo a lógica de controle ótima, minimamente restritiva e não-bloqueante.

Para essa abordagem, os eventos da planta são subdivididos em controláveis (Σ_c) e não-controláveis (Σ_u). A partir dessa divisão, é possível definir a propriedade da controlabilidade, em que os eventos não-controláveis não podem ser desabilitados pelo supervisor caso eles estejam habilitados na planta. Uma linguagem desejada $K = \mathscr{L}_m(G \parallel E)$ é dita controlável em relação à planta G se $\bar{K}\Sigma_u \cap \mathscr{L}(G) \subseteq \bar{K}$. Quando K não é controlável, sabe-se que existe a máxima sublinguagem controlável em relação à planta que está contida em K, o $S = Sup\mathscr{C}(K, G)$. O supervisor S pode ainda ser reduzido (Su e Wonham, 2004), de tal forma que toda a informação redundante da planta é retirada da estrutura do supervisor, tornando necessária a implementação da planta em conjunto com o supervisor reduzido para implementar o comportamento em malha fechada.

Quando aplicada em sistemas reais, a Teoria de Controle Supervisório encontra barreiras, devido à complexidade dos sistemas e da inerente complexidade computacional das operações com autômatos. A síntese dos supervisores monolíticos, que modelam o comportamento em malha fechada do sistema em uma única estrutura, podem resultar em autômatos extremamente grandes. Visando mitigar o problema da complexidade computacional, algumas extensões à TCS foram propostas. Neste trabalho, optou-se por utilizar o controle modular local (De Queiroz e Cury, 2000), que propõe a construção de um supervisor local (S_j) para cada especificação (E_j) com suas respectivas plantas locais (G_j), com $j = 1 \ldots n$.

Caso não haja conflito entre os supervisores modulares locais, suas ações são equivalentes à ação do monolítico. A verificação dessa propriedade pode ser feita verificando se a igualdade $\overline{S_1} \| \cdots \| \overline{S_n} = \overline{S_1 \| \cdots \| S_n}$ é verdadeira, onde n é o número total de supervisores.

2.3 Geração do código de programação

Para implementar de forma simplificada um SED em um sistema embarcado, foi desenvolvida uma ferramenta integrada com o UltraDES que automatiza a conversão da solução de controle supervisório para Arduino (Loures, 2024). Para adaptar a lógica de um SED ao Arduino representam-se estados, eventos, transições e autômatos de forma completa, considerando as limitações de memória e processamento do dispositivo. Cada estado é representado por um número inteiro de 16 bits, permitindo até 65.535 estados, com uma função associada que executa tarefas específicas relacionadas a esse estado. Os eventos são representados por uma estrutura que encapsula um *array* de *uint8_t* (*unsigned int* de oito bits), onde cada bit indica se o evento ocorreu ou não, e as transições de estado são determinadas com base nos eventos ocorridos. O mesmo é utilizado, uma vez que serão feitas operações lógicas bit a bit, onde cada posição no vetor representará um evento, e de acordo o número de eventos, será determinado o tamanho do vetor de *uint8_t*.

Existe, nessa implementação, a classe "*Automaton*", capaz de representar um autômato finito genérico que oferece flexibilidade ao permitir a configuração do autômato com um número variável de estados, eventos habilitados e funções de transição e execução. Além disso, o programa utiliza operações de máscara de bits, como *AND*, *OR*, *XOR* e *NOT*, para manipular os eventos de forma eficiente, garantindo o funcionamento correto da lógica do sistema de automação e ocupando menos espaço na memória.

A ferramenta utilizada é responsável pela geração do *template* do projeto, juntamente com as classes `Automaton.h` e `Event.h`, prontas para serem carregadas no Arduino. Resta ao usuário apenas a tarefa de alterar o arquivo `UserFunctions.cpp`, no qual deve ser implementada a configuração dos pinos do Arduino, definir a ação relacionada a cada estado da planta (ou seja, definir os valores de saída para o sistema físico em cada estado) e, por fim, implementar como os eventos controláveis e não controláveis são recebidos. Por padrão, há uma implementação de exemplo das ações de estado e das ações de recebimento dos eventos, que tem como finalidade ilustrar onde devem ser feitas as alterações. No entanto, por questões de generalidade, o usuário final deve modificar esses arquivos de acordo com as necessidades do seu projeto.

3. CÉLULA DE TERMOPROCESSAMENTO (CTP)

A planta utilizada (Figura 1) foi adaptada de (De Queiroz e Cury, 2002). Ela consiste de uma mesa circular de quatro estágios (G_0), uma esteira de entrada (G_1), uma unidade de tratamento térmico (G_2), uma unidade de teste (G_3) e um atuador separador (G_4) que desloca a peça da mesa para o destino final. O funcionamento da planta, apresentada no diagrama da Figura 1, para produção de uma peça consiste em sete passos:

(1) a esteira avança depositando uma peça na posição P_1;
(2) a mesa gira 72°;
(3) a peça vai para tratamento térmico na posição P_2;
(4) a mesa gira 72°;
(5) a peça é levada para a posição P_3 onde é submetida a um teste que vai classificar se deve ser depositada na caixa na posição O_0 ou na posição O_1;
(6) a mesa gira 72°;
(7) o atuador separador coloca a peça na caixa correspondente.

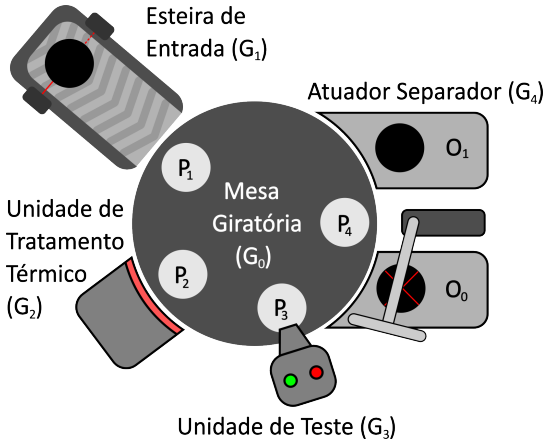


Figura 1. Diagrama da Célula de Termoprocessamento.

A fase de operação da esteira conta com um sensor de entrada de peça. Caso alguma peça seja detectada, ocorre o acionamento da esteira, que realiza o depósito da peça na mesa giratória. No tratamento térmico, a peça é inicialmente submetida ao processo de aquecimento e, posteriormente, passa pelo resfriamento. Na estação de teste, o sensor magnético avalia a peça em magnética ou não-magnética. Por fim, o atuador separa as peças de acordo com o resultado da medição do sensor magnético no estágio anterior. Caso a peça seja magnética, o atuador se desloca visando empurrar a peça para a caixa correspondente à posição O_0, caso contrário, a peça é levada para a caixa, identificada pela posição O_1. Em relação à planta original de (De Queiroz e Cury, 2002), a interação entre o teste e o atuador é diferente, sendo necessário, após verificar que a peça é magnética, deslocar o atuador antes de girar a mesa de forma a posicionar o atuador para colocar a peça na caixa correta.

Tabela 1. Eventos considerados para desenvolvimento dos modelos.

	Eventos		
	Início	**Etapas Intermediárias**	**Finalização**
Mesa Giratória	a_0		b_0
Esteira	a_1		b_1
Tratamento Térmico	a_2		b_2
Teste	a_3	b_{mag}, b_{naomag}	b3
Atuador Separador	a_e	b_e, a_d	b_d
Σ_c**:** $\{a_i\}$, $i \in \{0,1,2,3,4,e,d\}$		Σ_u**:** $\{b_i\}$, $i \in \{0,1,2,3,4,e,d\}$	

3.1 Implementação do Protótipo

A célula de manufatura foi inicialmente desenvolvida, no contexto de uma disciplina de pós-graduação oferecida na UFMG, por um doutorando, Prof. Alan Mendes Marotta (CEFET-Divinópolis), que gentilmente doou o protótipo para o LACSED.

Este protótipo está instalado dentro de um caixa de madeira de 36 cm por 26 cm e possui uma mesa circular de acrílico, uma esteira de entrada e mais três estágios representando equipamentos de tratamento térmico, teste e um atuador que retira as peças da mesa conforme sua classificação: magnetizadas ou não (Figura 2). As estruturas plásticas foram criadas com o uso de manufatura aditiva, LEDs foram usados para sinalizar o funcionamento dos estágios, um sensor magnético do tipo *reed-switch* foi usado para detectar a magnetização da peça, motores de passo para girar a mesa e a esteira, e um servo motor foi utilizado para mover o atuador separador. Não há sensores de presença nos estágios da mesa.

A célula conta com um Arduino Nano, responsável por realizar o controle de todos os periféricos. Fez-se necessário o uso de *drivers* para motores de passo devido à baixa corrente de saída dos pinos desse microcontrolador. Uma placa perfurada gerencia a distribuição de energia para todos os dispositivos além de possuir um capacitor para estabilização da alimentação e um diodo para prevenir sobrecarga na porta USB destinada à programação e monitoramento da planta.

O tratamento térmico realizado pela planta é simulado por meio de LEDs vermelhos e amarelos que se acendem para representar o aquecimento da peça ao passo que os LEDs azuis presentes na lateral representam o resfriamento da peça. LEDs verde e vermelho foram empregados também na etapa de teste para que indicar o resultado do teste de magnetismo, sendo o LED verde para peça não-magnética e o LED vermelho para peça magnética.

Todo o projeto do protótipo, arquivos para impressão 3D, lista de componentes, circuitos, código implementado e *firmware* carregado podem ser encontrados no Github [1] do Laboratório de Análise e Controle de Sistema a Eventos Discretos da UFMG.

4. DESENVOLVIMENTO

O objetivo do trabalho é apresentar um protótipo de um sistema a eventos discretos, desenvolver uma solução usando a teoria do

[1] https://github.com/lacsed

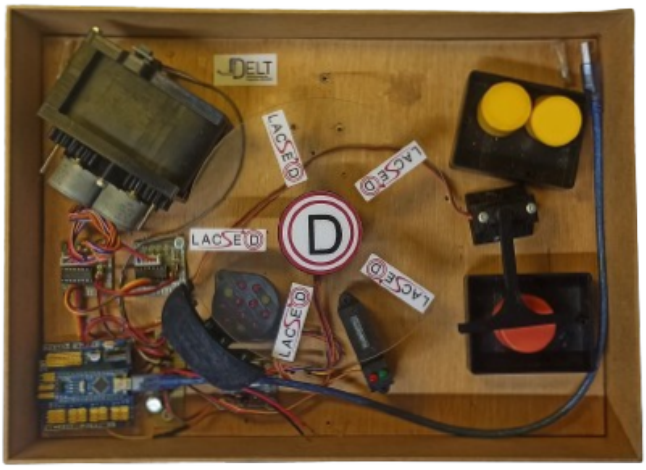


Figura 2. Protótipo desenvolvido.

controle modular local, fazer a conversão automática da solução em código para Arduino usando o UltraDES, complementar a implementação e testar.

A primeira etapa foi modelar o sistema, suas especificações e obter os supervisores não-conflitantes pela síntese modular local. Tanto a modelagem quanto as especificações foram adaptadas de (Cury, 2001) para tratar as diferenças propostas no protótipo implementado. As plantas estão apresentadas na Seção 4.1, as especificações na Seção 4.2 e algumas informações sobre os supervisores são apresentadas na Seção 4.3. Posteriormente, na Seção 4.4, apresentam-se informações sobre a implementação do código gerado automaticamente em Arduino.

4.1 Modelagem da Planta

Cada dispositivo do sistema é modelado por um autômato G_i, sendo que a mesa giratória (G_0), a esteira (G_1) e o tratamento térmico (G_2) possuem modelos de 2 estados, conforme a Figura 3, onde $i = 0, 1, 2$. Os eventos correspondentes à operação de ligar estes equipamentos (a_i) são controláveis e os eventos que desligam os equipamentos são não-controláveis (b_i), conforme Tabela 1.

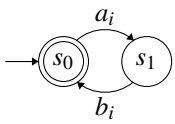


Figura 3. Autômatos G_i: Mesa G_0 ($i = 0$), Esteira G_1 ($i = 1$) e Tratamento térmico G_2 ($i = 2$).

Para modelar a unidade de teste (G_3), considera-se que a operação é iniciada com o evento controlável a_3. Em seguida, de acordo com resultado da detecção do sensor magnético, a peça é classificada como peça magnética (b_{mag}) ou peça não magnética (b_{naomag}). Após a conclusão da medição, a operação é encerrada pelo evento não-controlável b_3, conforme Figura 4.

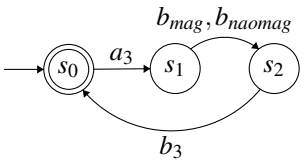


Figura 4. Autômato G_3: Unidade de Teste.

O modelo representado na Figura 5 descreve o funcionamento do atuador, que tem como posição inicial a posição O_0. Partindo da posição inicial, o atuador pode realizar o deslocamento do braço para a posição O_1, com o evento a_e. O evento b_e indica que o atuador chegou na sua posição final O_1, onde agora só pode se deslocar de volta para a posição O_0, com o evento a_d. Ao finalizar seu movimento, evento b_d, o atuador retorna à sua posição inicial.

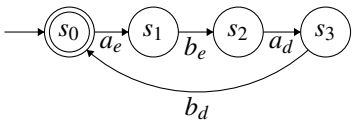
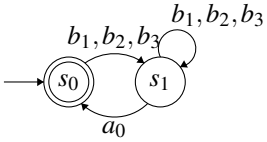


Figura 5. Autômatos G_4: Atuador Separador.

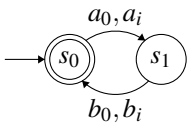
4.2 Modelagem das Especificações

As especificações foram desenvolvidas com objetivo de garantir a operação correta da planta. As especificações são descritas a seguir.

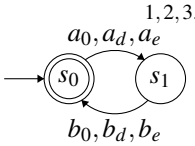
Especificações para giro da mesa (E_a, E_{b1}, E_{b2}, E_{b3}, E_{b4}): Com a especificação E_a, pretende-se impedir que a mesa gire se não houver finalização de operação em alguma etapa. Com as especificações E_{b1}, E_{b2}, E_{b3}, E_{b4} pretende-se condicionar o giro da mesa à finalização de todas as atividades iniciadas.



(a) Especificação E_a.



(b) Especificação E_{bi}, $i = 1, 2, 3$.



(c) Especificação E_{b4}

Figura 6. Especificações para correto funcionamento da mesa.

Especificação de interação entre etapas (E_{c1}, E_{c2}, E_{c3}, E_{c4}): As especificações das Figuras 7, 8 e 9 visam tratar os possíveis problemas relacionados ao tráfego das peças, como sobreposição, retrabalho e acionamentos indevidos entre os equipamentos G_1, G_2 e G_3. E_{cj}, $j = 1, 2$, relacionam a movimentação de peças entre os estágios 1 e 2 e entre 2 e 3. Por exemplo, levando-se em conta os estágios $j = 2$ e $j + 1 = 3$, essa especificação impede que a peça seja submetida ao tratamento térmico duas vezes sem que a mesa gire ($b_2 a_0$) e força que o teste (a_3) só ocorra após o giro da mesa. Caso ocorra um segundo b_2, deve-se ter um segundo a_3 e outro giro da mesa (a_0).

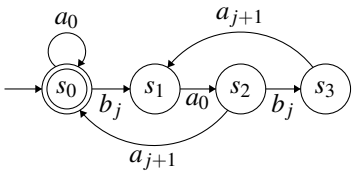


Figura 7. Especificação $E_{cj}, j = 1, 2$

E_{c3} relaciona a movimentação de peças entre os estágios 3 e 4. Essa especificação garante que o atuador separador se

posicione antes ($b_3a_ea_0$) ou depois ($b_3a_0a_e$) da mesa ter girado, dependendo do resultado do teste e especificação E_{c4}.

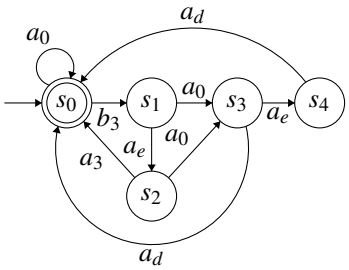


Figura 8. Especificação E_{c3}.

E_{c4} tem o objetivo de assegurar a movimentação correta do atuador, de acordo com a medição obtida na unidade de teste, ou seja, que o atuador se posicione para executar o movimento correto, depositando a peça na caixa da posição O_0 ou O_1. Assim, com o resultado da medição do teste (b_{mag} ou b_{naomag}), caso haja a indicação de que a peça é magnética, o braço se desloca em direção à posição O_1, aguardando que a mesa gire e o atuador seja redirecionado novamente para a posição O_0, retirando a peça da mesa. Caso haja a indicação de que a peça seja não-magnética, o atuador permanece em repouso e, após o giro da mesa, ele se desloca, alocando a peça em O_1. O atuador retorna, então, ao seu estado inicial, posicionando-se sobre O_0.

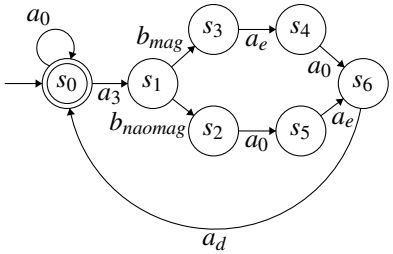


Figura 9. Especificação E_{c4}.

A partir dos modelos das plantas e especificações, é possível identificar quem são as plantas locais que têm relação com cada especificação. Isto é possível a partir da obervação de todas as especificações e análise de seus eventos, para que seja feita a busca por eventos comuns nos autômatos que representam o modelo das plantas.

4.3 Análise e Cálculo dos Supervisores

Tendo à disposição os modelos das plantas e das especificações, é possível calcular, a partir da Teoria de Controle Supervisório (TCS), previamente descrita na Seção 2, o supervisor deste sistema. O supervisor monolítico possui 299 estados e 682 transições, que é relativamente grande para ser implementado em placas de desenvolvimento comuns, como é o caso deste trabalho. Por isso, optou-se pela abordagem do controle modular local (também previamente discutida na Seção 2), em que se calcula um supervisor modular local para cada especificação juntamente com suas plantas locais. Para a célula de manufatura em questão, foram obtidas nove especificações, Seção 4.2, gerando nove supervisores modulares locais, não-conflitantes. Para isso, foi preciso encontrar as plantas locais de cada especificação, o que é possível de ser feito olhando para os eventos que compõem cada uma das especificações e em quais plantas os mesmos se encontram. Ou seja, toda planta que contenha, pelo menos, um evento comum com determinada especificação, será planta local da mesma. A partir disso, os supervisores encontrados foram reduzidos (Su e Wonham, 2004) e os números de estados e transições dos supervisores reduzidos obtidos são apresentados na Tabela 2. A biblioteca UltraDES (Alves e Pena, 2023) foi utilizada para síntese e redução dos supervisores.

Tabela 2. Análise dos supervisores modulares locais não-conflitantes

	Supervisores		**Supervisores reduzidos**	
	Especificação	**Plantas Locais**	**Estados**	**Transições**
S_1	E_a	G_0, G_1, G_2, G_3	3	12
S_2	E_{b1}	G_0, G_1	2	4
S_3	E_{b2}	G_0, G_2	2	4
S_4	E_{b3}	G_0, G_3	2	6
S_5	E_{b4}	G_0, G_4	2	6
S_6	E_{c1}	G_0, G_1, G_2	6	16
S_7	E_{c2}	G_0, G_2, G_3	8	19
S_8	E_{c3}	G0, G3, G4	4	17
S_9	E_{c4}	G0, G3, G4	18	84

4.4 Implementação do código

Os detalhes da conversão automática dos autômatos para código de Arduino podem ser encontrados em (Loures, 2024). Para esta versão preliminar do artigo, refere-se ao repositório[2] da ferramenta de conversão da TCS para Arduino, onde encontra-se a versão submetida do artigo em questão e a monografia de Projeto Final de Curso (Loures, 2023) que a originou.

Nessa seção, aborda-se a parte não automática da implementação.

O código gerado por meio da conversão automática é genérico e, para atender as particularidades de cada planta, é necessário complementar o código. Nessa implementação houve a necessidade de executar sequências de acionamento de motores de passo, acionamento de saídas e leitura de entradas. Apresenta-se a seguir um trecho de código adicionado ao arquivo `UserFunctions.cpp` que ilustra a complementação necessária para que os estados dos autômatos gerem saídas físicas, no caso o acionamento de um motor.

```
int motor_step = 0;
void StateActionAutomatonG1State1()
{
  step(esteira, motor_step%8);
  delayMicroseconds(1200);
  motor_step++;
  if(motor_step >= 5500)
    virtual_pins_nc[1] = true;
}
```

O trecho de código acima é responsável por executar a ação do estado S_1 do autômato G_1 que consiste em gerar a sequência de acionamento para a rotação da esteira. A fim de garantir o paralelismo de funcionamento entre as partes que compõem a planta, a função de giro é modelada de forma a executar um passo do motor por vez e então entregar o processamento de volta à implementação para que outras funções possam ser executadas entre o intervalo dos passos.

Ao se executar o estado S_1 de G_1 5500 vezes, a esteira finaliza a ação de depositar a peça na posição P_1 da mesa e, então, a variável `virtual_pins_nc[1]` passa para nível lógico verdadeiro. Essa variável é lida posteriormente como o evento não controlável de desligamento da esteira. A técnica de virtualizar os eventos de término das operações é utilizado também na unidade de tratamento térmico, porém esta baseia o desligamento da operação no tempo decorrido desde que o autômato G_2 entrou no estado S_1.

[2] https://github.com/lacsed/arduino-generator

A programação da unidade de teste consiste em uma lógica condicional que, no estado S_1 do autômato G_3, verifica o valor de um pino de entrada. Se o valor for positivo, indica que a peça está magnetizada e então o evento b_{mag} é disparado, caso contrário, o evento b_{naomag} é disparado. O trecho de código abaixo demonstra como foi implementada a lógica condicional de teste.

```
void StateActionAutomatonG3State1()
{
  if(digitalRead(13))
  {
    virtual_pins_nc[3] = true;    // bmag
    virtual_pins_c[0] = true;
    virtual_pins_c[1] = false;
  }
  else
  {
    virtual_pins_nc[4] = true;    // bnmag
    virtual_pins_c[1] = true;
    virtual_pins_c[0] = false;
  }
}
```

As variáveis `virtual_pins_nc[3]` e `virtual_pins_nc[4]` representam variáveis diretamente atribuídas às transições b_{mag} e b_{naomag} respectivamente, já as variáveis `virtual_pins_c[0]` e `virtual_pins_c[1]` são utilizadas para dar ao controlador a informação de qual tipo de peça foi detectado.

O atuador separador é movido por um servo motor modelo 9G com engrenagens de metal. Esse tipo de motor possui um controlador interno que garante que a posição do eixo corresponda ao sinal de referência recebido. Para acionar o servo motor sem fazer o uso de bibliotecas que geram interrupções no código e ocupam contadores do microcontrolador, o pulso que corresponde à posição foi gerado de forma manual com uma espera entre o ligar e o desligar de um pino. Ao todo são gerados 20 pulsos de um milissegundo de largura para a posição O_1 e de dois milissegundos para a posição O_0, ao término desses 20 pulsos entende-se que o braço já atingiu a posição desejada e então o evento b_3 é disparado.

5. CONCLUSÃO

A conversão automática da solução de controle supervisório modular local foi bem sucedida. A implementação do Controle Supervisório por meio da conversão automática se mostrou simples, com alto grau de abstração, uma visualização intuitiva, modular e de fácil agregação de novas regras de comportamento.

O código gerado pela ferramenta de tradução consome apenas 26% da memória de programa e 48% da memória dinâmica do microcontrolador (Arduino nano) para aplicar toda a regra de funcionamento à planta. Após as alterações no arquivo `UserFunctions.cpp`, apenas 208 linhas precisaram ser adicionadas e o consumo de memória de programa aumentou para 28% enquanto que o uso de memória dinâmica reduziu para 33%.

O sistema funciona de forma segura com qualquer número de peças, não acionando os equipamentos na ausência de peças, não girando a mesa durante a operação dos outros equipamentos, podendo realizar a operação em paralelo da esteira, tratamento térmico, teste e atuador.

O funcionamento do protótipo foi filmado e pode ser visto na página de vídeos [3] do Laboratório de Análise e Controle de Sistema a Eventos Discretos da UFMG.

AGRADECIMENTOS

Este trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, CNPq e Fapemig. Os autores gostariam de reconhecer e agradecer a doação da primeira versão do protótipo pelo Prof. Alan Mendes Marotta (CEFET-Divinópolis, MG).

REFERÊNCIAS

Alves, L.V. e Pena, P.N. (2023). Ultrades project-a multiplatform discrete event systems tool. *IFAC-PapersOnLine*, 56(2), 6081–6086.

Cassandras, C.G. e Lafortune, S. (2008). *Introduction to discrete event systems*. Springer.

Cavalcante, P.H.P., Morais, L.G.C., van der Linden, R., e da Silva Netto, A.V. (2020). Proposição de arquitetura para ensino de automação industrial: Integrando ambiente virtual 3D, ferramenta de modelagem matemática e ambiente real. In *Congresso Brasileiro de Automática-CBA*, volume 2.

Cury, JER e Queiroz, M.d.e.S.E. (2001). Síntese modular do controle supervisório em diagrama escada para uma célula de manufatura. *V Simpósio Brasileiro de Automação Inteligente. Canela (Brazil)*.

De Queiroz, M.H. e Cury, J.E. (2000). Modular supervisory control of large scale discrete event systems. In *Discrete Event Systems: Analysis and Control*, 103–110. Springer.

De Queiroz, M.H. e Cury, J.E. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Sixth International Workshop on Discrete Event Systems, 2002. Proceedings.*, 377–382. IEEE.

Freitas, G.M. e Pena, P. (2023). Modelo virtual de um sistema flexível de manufatura como instrumento no ensino de automação industrial. In *Simpósio Brasileiro de Automação Inteligente-SBAI*, 1.

Loures, M.P. e Alves, L. (2024). Geração automática de código para arduino com base na teoria de controle supervisório integrado com o ultrades. In *Submetido para XXV Congresso Brasileiro de Automática*.

Loures, M. (2023). *Geração Automática de Código para Arduino com base na Teoria de Controle Supervisório*. Projeto Final de Curso de Graduação em Engenharia de Controle e Automação.

Ramadge, P.J. e Wonham, W.M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 81–98.

Su, R. e Wonham, W.M. (2004). Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, 14, 31–53.

[3] https://www.youtube.com/watch?v=ELOjbFRPrl8