# Laboratório 2 - Sistema Logístico Autônomo de Transporte

Universidade de Brasília
Departamento de Ciência da Computação
CIC0248 - Sistemas de Tempo Real - 2022/2
22 de fevereiro de 2023

Davi Salomão Soares Corrêa
18/0118820
davi.salomao@aluno.unb.br

Francisco Henrique da Silva Costa
18/0120174
francisco.henrique@aluno.unb.br

Matheus Teixeira de Sousa
18/0107101
teixeira.sousa@aluno.unb.br

## 1. Objetivos

Este laboratório é desenvolvido no contexto da disciplina de Sistemas de Tempo Real da Universidade de Brasília e tem por objetivo implementar, em ambiente simulado por meio do *software* CoppeliaSim [3] com o pyRTOS [8], o sistema logístico autônomo de transporte previamente elaborado no laboratório anterior [4] e verificar o cumprimento dos requisitos temporais propostos.

## 2. Introdução

A logística é uma área da administração responsável pela gestão, armazenamento e distribuição de recursos para uma determinada atividade [5]. Dentro do contexto da engenharia, essa área pode envolver situações como o planejamento de compra e de distribuição de material, definição de escalas de manutenção de máquinas, planejamento da produção. É com base nessa ideia que o projeto proposto busca adaptar um sistema logístico autônomo de transporte como empregado na montadora da Porsche [2].

Durante o processo de montagem do Porsche 911, um veículo autônomo faz o transporte dos motores produzidos na oficina de montagem para a linha de montagem principal, em outro ponto da fábrica. Para isso, o veículo conta com um sistema de orientação baseado em ímãs colocados abaixo do piso de concreto, de modo que o sistema consegue se orientar e encontrar o caminho correto. Além disso, como o ambienta da fábrica é altamente colaborativo, cada veículo deve ter uma série de mecanismos de segurança para evitar colisões.

No entanto, antes de efetivamente tratar dos detalhes do projeto implementado, as próximas seções da introdução são dedicadas à revisão de conceitos importantes da teoria aplicada e à descrição das correções realizadas na proposta inicial. Depois, a seção 3 apresenta os detalhes da implementação realizada no CoppeliaSim para simulação do sistema. Então, a seção 4 apresenta os resultados obtidos durante os testes do sistema modelado. Por fim, a seção 5 discute os resultados obtidos e apresenta observações finais sobre o trabalho.

### 2.1. Tempo de resposta

O tempo de resposta (*response time*) de uma tarefa, definido pelo intervalo de tempo entre a chegada da tarefa e sua conclusão, deve ser menor ou igual ao seu *deadline* relativo [6]. Esse valor inclui, além do tempo de execução da tarefa, eventuais intervalos ocasionados por tratadores de interrupção ou outras tarefas de maior prioridade. Como esse valor pode variar, o objetivo durante a análise do tempo de resposta é determinar o tempo de resposta no pior caso (WCRT - *worst-case response time*).

Para sistemas complexos, a análise estatística do tempo de resposta pode ser adotada para tentar caracterizar o comportamento de uma tarefa. Para medir o tempo de resposta, é preciso executar o sistema com um certo cenário de entrada de dados. Para estimar o tempo no pior caso, o conjunto de cenários deve ser abrangente o suficiente para ser representativo para o sistema em análise. O testes para medição podem ser divididos em quatro categorias: teste de tarefa, teste comportamental, teste intertarefas e teste de sistema [6].

Uma estatística importante, extraída das medições realizadas, é o tempo de resposta máximo observado, também conhecido como HWM (*high water mark*). Em muitos casos, o HWM pode ser acrescido de uma margem de segurança para ser utilizado como um estimador do WCRT

para fins de verificação do cumprimento dos requisitos temporais em uma perspectiva otimista [6]. O valor da margem deve estar relacionado com a variância do tempo de execução, uma vez que este parâmetro mensura o grau de incerteza das medições. Quanto maior a variância, maior a incerteza e maior deve ser a margem utilizada.

### 2.2. Correções e modificações

A partir das orientações recebidas durante a apresentação do laboratório 1 [4], a dinâmica de movimentação do veículo foi repensada e a tarefa **Seguir rota**, antes com um *deadline* de 10 minutos, foi dividida em outras duas tarefas (**Calcula velocidade** e **Ajusta velocidade**) com *deadlines* menores. Além disso, para focar no desenvolvimento das soluções de tempo real, um modelo de robô móvel disponível na biblioteca do CoppeliaSim foi utilizado como veículo de transporte.

Inicialmente, o veículo deveria aumentar a velocidade após longos períodos parados em uma tentativa de compensar o tempo perdido e não perder o *deadline*. No entanto, esse comportamento foi removido em função da dificuldade de implementar essa característica sem afetar o movimento do veículo. Por fim, a tarefa **Alarme** foi removida e transformada em uma notificação via terminal para simplificar a implementação no CoppeliaSim.

## 3. Metodologia

### 3.1. Definição das tarefas

Com as modificações realizadas, as tarefas a serem implementadas estão descritas na tabela 1. A tarefa **Ajusta mapa**, com prioridade 2, define o caminho a ser seguido pelo veículo. Dessa forma, essa tarefa deve ser realizada antes das outras tarefas relacionadas ao movimento do robô. Depois, na tarefa **Ajusta velocidade**, temos o menor *deadline*, uma vez que o ajuste da velocidade pode ocorrer com o veículo em movimento por meio de interrupções para parar o robô em caso de colisão.

| Tarefas | Prior.[1] | Deadline | Período |
|---|---|---|---|
| Ajusta mapa | 2 | 1s | esporádica |
| Ajusta velocidade | 3 | 0,05s | esporádica |
| Calcula velocidade | 5 | 0,3s | aperiódica |
| Controla garra | 1 | 10s | esporádica |
| Lê sensores | 4 | 0,2s | periódica |

1 - Prior. é uma abreviação de prioridade.

Tabela 1: Tarefas e requisitos temporais

A tarefa de menor prioridade, **Calcula velocidade**, implementa o código para o cálculo da velocidade que é executado sempre que possível. Essa tarefa foi implementada dessa forma para minimizar o efeito subamortecido da movimentação do veículo em função do atraso de comunicação entre o programa e o CoppeliaSim.

Em seguida, temos a tarefa **Controla garra**, com a maior prioridade. Essa situação se justifica pelo fato de não poder ter nenhuma outra tarefa enquanto a garra de movimenta, tanto por questões lógicas (o robô pode pegar uma peça se movimentando) quanto por segurança. Por fim, a tarefa **Lê sensores**, única tarefa periódica, habilita a execução das tarefas relacionadas à movimentação do robô e age como uma medida de segurança contra obstruções.

### 3.2. Implementação no CoppeliaSim

Como interface de robótica, o simulador CoppeliaSim em conjunto com a biblioteca pyRTOS foi adotado para testar e validar algoritmos em tempo real. Na simulação, o robô *KUKA youBot* foi escolhido por ser um modelo com mobilidade flexível e apresentar um braço manipulador adequado para ações movimentação de materiais.


Figura 1: *KUKA youBot* [1]

#### 3.2.1 Braço manipulador

Inicialmente, foram obtidos os dados do fabricante [1] do robô *KUKA youBot* para calcular a alcançabilidade da atuação do braço manipulador de cinco graus de liberdade. No CoppeliaSim, o princípio de cinemática inversa foi aplicado, no qual o centro da garra foi definido como referencial para determinação do movimento das juntas no braço robótico.

Nas situações de movimentação de peça, o ponto alvo da garra foi referenciado em um bloco e, por meio das funções de agrupamento cinemático, braço realiza as operações de translação e rotação de suas juntas adequadas. Na programação, uma função de espera é exigida na transição entre o controle de erro do veículo para parar próximo ao alvo e, ao mesmo tempo, realizar movimentação do braço.

Em simulação, com tempo de simulação do CoppeliaSim, essa realização de movimento se aproximou da realidade. Entretanto, ao ligar o modo *Real Time*, a função de espera se torna bastante custosa para realização do movimento do braço. Dessa forma, esse movimento foi abstraído na simulação de tempo real para apenas a atuação da garra, ou seja, abrir e fechar.

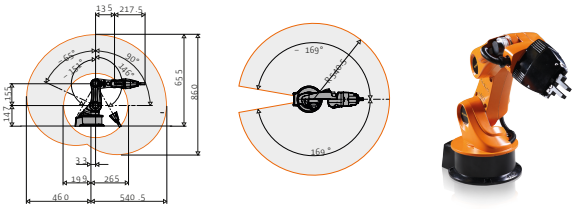

Figura 2: Braço manipulador [1]

### 3.2.2 Plataforma

A plataforma do *KUKA youBot* apresenta mobilidade omnidirecional, sendo ideal para o circuito criado e inspirado no ambiente fabril. Além disso, seu modelo é holonômico e possui três graus de liberdade coincidentes com número de suas velocidades atuáveis. Sob essa perspectiva, o robô apresenta o número mínimo de variáveis independentes que representam completamente o sistema de forma que o espaço de configurações é estabilizável.

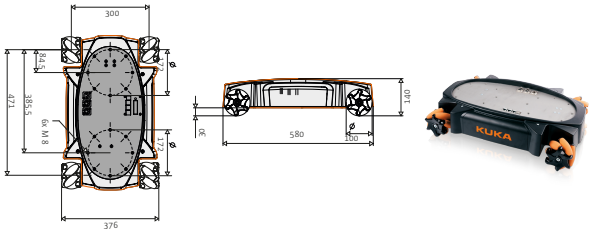

Figura 3: Plataforma do *KUKA youBot* [1]

## 3.3. Dinâmica do veículo

No modelo de movimento do veículo, as coordenadas entre a velocidade linear e a velocidade angular com relação ao centro de massa do veículo foram adotadas. Inicialmente, obtivemos o modelo cinemático direto (equação 3.3), em que o valor da velocidade angular (entrada) lida por um sensor, por exemplo, corresponde a uma velocidade linear (saída).

Contudo, no CoppeliaSim, assumimos a cinemática inversa, na qual um valor alvo de velocidade linear é utilizado para determinar a velocidade angular necessária 2. Esse modo de operação é interessante visto que é fundamentado em uma abordagem de controle de robótica móvel com observação a partir da saída, ao invés de apenas previsões cinemáticas.

Nesse contexto, convertemos as posições alvo no processo de movimentação logística em velocidade lineares para que, depois, fossem estimadas as velocidades angulares necessárias nas juntas da plataforma móvel. Abordagem adotada possibilita melhor controle da robótica móvel para compensar a incerteza criada na integração dos parâmetros de posição e orientação.

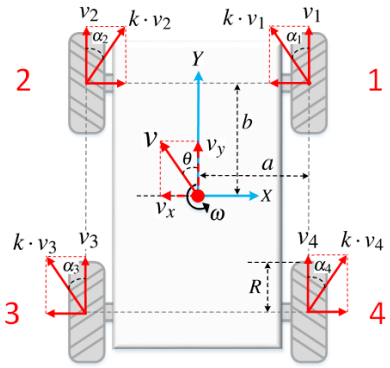


Figura 4: Corpo rígido do veículo [7]

$$\begin{cases} v_1 = v_y - v_x + a\omega + b\omega \\ v_2 = v_y + v_x - a\omega - b\omega \\ v_3 = v_y - v_x - a\omega - b\omega \\ v_4 = v_y + v_x + a\omega + b\omega \end{cases} \tag{1}$$

$$\implies v_i = \omega_i R$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -1 & 1 & (a+b) \\ 1 & 1 & -(a+b) \\ -1 & 1 & -(a+b) \\ 1 & 1 & (a+b) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{2}$$

# 4. Resultados

Após implementação no CoppeliaSim, simulamos e medimos o tempo de resposta das tarefas propostas por meio da função *time* da biblioteca do Python. Para cada conjunto de medições, criamos o histograma para analisar a frequência e estimamos o WCRT como a soma do HWM e do desvio padrão, ou seja,

$$\text{WCRT}_{\text{est}} = \text{HWM} + \sigma. \tag{3}$$

## 4.1. Ajusta mapa

Na tarefa **Ajusta mapa**, foram realizadas 100 medições ao longo de 11 voltas, cujo resultado pode ser visto na figura 5. A partir dos valores medidos, extraímos o HWM

de 0,0926 segundos e o desvio padrão de 0,0177 segundos. Dessa forma, o WCRT estimado para essa tarefa é de **0,1104 segundos**.

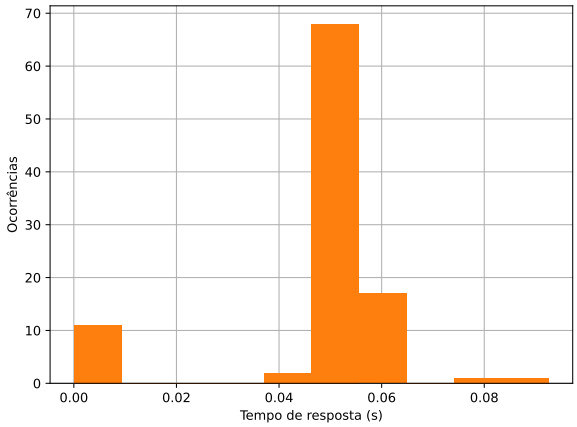


Figura 5: Histograma do tempo de resposta para tarefa Ajusta mapa

## 4.2. Ajusta velocidade

Depois, na tarefa **Ajusta velocidade**, foram realizadas 3113 medições ao longo de 1 volta, cujo resultado pode ser visto na figura 6. A partir dos valores medidos, extraímos o HWM de 0,0369 segundos e o desvio padrão de 0,0007 segundos. Dessa forma, o WCRT estimado para essa tarefa é de **0,0375 segundos**.

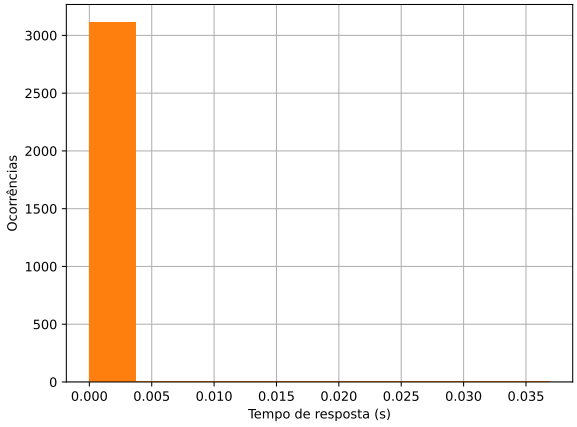


Figura 6: Histograma do tempo de resposta para tarefa Ajusta velocidade

## 4.3. Calcula velocidade

Então, na tarefa **Calcula velocidade**, foram realizadas 3186 medições ao longo de 1 volta, cujo resultado pode ser visto na figura 7. A partir dos valores medidos, extraímos o HWM de 0,1107 segundos e o desvio padrão de 0,0109 segundos. Dessa forma, o WCRT estimado para essa tarefa é de **0,1216 segundos**.

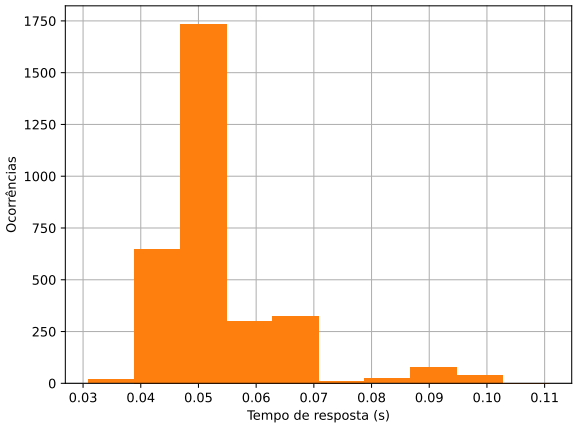


Figura 7: Histograma do tempo de resposta para tarefa Calcula velocidade

## 4.4. Controla garra

Para a tarefa **Controla garra**, foram realizadas 24 medições ao longo de 11 voltas, cujo resultado pode ser visto na figura 8. A partir dos valores medidos, extraímos o HWM de 1,858 segundos e o desvio padrão de 0,6654 segundos. Dessa forma, o WCRT estimado para essa tarefa é de **2,523 segundos**.

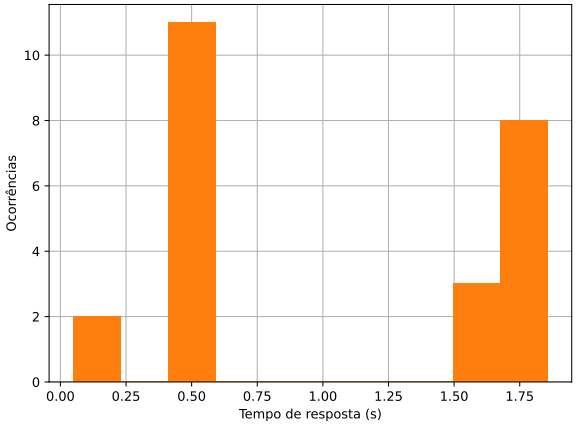


Figura 8: Histograma do tempo de resposta para tarefa Controla garra

## 4.5. Lê sensores

Por fim, na tarefa **Lê sensores**, foram realizadas 3117 medições ao longo de 1 volta, cujo resultado pode ser visto na figura 9. A partir dos valores medidos, extraímos o

HWM de 0,1117 segundos e o desvio padrão de 0,0107 segundos. Dessa forma, o WCRT estimado para essa tarefa é de **0,1224 segundos**.

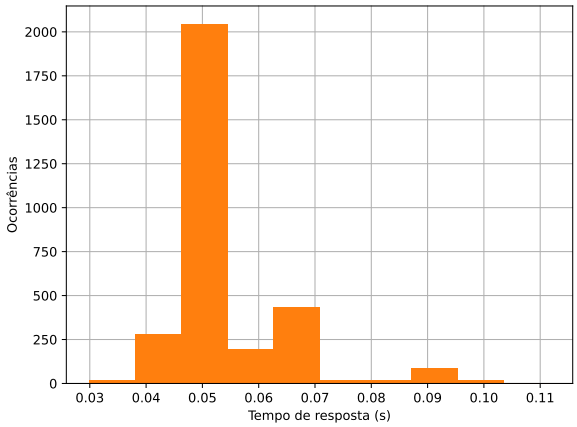


Figura 9: Histograma do tempo de resposta para tarefa Lê sensores

### 4.6. Resumo dos tempos de resposta

De fato, a partir da tabela 2 (resumo dos valores estimados), é possível verificar que todos os WCRT estimados são menores que o *deadline* relativo das respectivas tarefas.

| Tarefas | Deadline | HWM | WCRT |
|---|---|---|---|
| Ajusta mapa | 1s | 0,09s | 0,11s |
| Ajusta velocidade | 0,05s | 0,037s | 0,038s |
| Calcula velocidade | 0,3s | 0,11s | 0,12s |
| Controla garra | 10s | 1,86s | 2,52s |
| Lê sensores | 0,2s | 0,11s | 0,12s |

Tabela 2: Tarefas, deadline e WCRT estimado

## 5. Discussão e conclusões

Dos resultados para o tempo de resposta da tarefa **Ajusta mapa** (figura 5), é possível observar um padrão no qual um ponto do percurso mais rápido que os outros. Esses 11 valores medidos correspondem aos momentos nos quais o veículo está na última posição do trajeto. Enquanto isso, as medidas com maior tempo de resposta estão em pontos nos quais há comunicação para controle da garra do robô.

Em função da simplicidade da tarefa **Ajusta velocidade** (apenas ajuste da velocidade das rodas), o tempo de resposta mais curto (figura 6) é justificável. No entanto, nas figuras das tarefas **Calcula velocidade** e **Lê sensores**, é possível observar uma variedade maior no tempo de resposta. e Esse comportamento pode ser justificado por dois fatores: menores prioridades, especialmente no caso da primeira, e forte dependência dessas tarefas com as implementações realizadas no CoppeliaSim.

Para a tarefa **Controla garra**, é possível observar um tempo de resposta bem abaixo do *deadline* especificado (cerca de 4 vezes). Esse resultado pode ser explicado pela simplificação na implementação da tarefa no simulador. Além disso, as duas regiões observadas na figure 8 podem ser relacionadas com as ações de pegar (mais lenta) e soltar (mais rápida) um objeto em função do ajuste de força durante coleta da peça.

O comportamento observado na tabela 2 sugere que, de uma perspectiva otimista, o sistema deve ser capaz de cumprir os requisitos temporais para a maioria dos casos. No entanto, como o WCRT real não pode ser encontrado, uma vez que o WCET *worst-case execution time* ou tempo de execução no pior caso) também não pode, não é possível afirmar que o sistema nunca perderá o *deadline*, ainda que as probabilidades envolvidas sejam baixas.

Por fim, ao longo do trabalho, implementamos um modelo de sistema logístico autônomo de transporte em ambiente simulador via CoppeliaSim para atender requisitos temporais por meio de um microkernel utilizando o pyRTOS. Os resultados obtidos verificam o cumprimento dos *deadlines* definidos e os testes realizados durante a apresentação sugerem um bom comportamento do sistema para observância das prioridades.

## Referências


[1] KUKA youBot Research & Application Development in Mobile Robotics. Disponível em: https://www.kuka.com/. Online; Acesso em 16 de fev. de 2023.

[2] Carros Cascavel. Mega fábricas porsche natgeo dublado port. Disponível em: https://www.youtube.com/watch?v=J38S7zZuOkA. Online; Acesso em 18 de jan. de 2023.

[3] CoppeliaSim. Robot simulator coppeliasim: create, compose, simulate, any robot. Disponível em: https://www.coppeliarobotics.com/. Online; Acesso em 19 de jan. de 2023.

[4] D. S. S. Corrêa, F. H. da Silva Costa, e M. T. de Sousa. Laboratório 1 - sistema logístico autônomo de transporte. 2023. Disciplina Sistemas de Tempo Real, UnB 2022/2.

[5] J. Crestani. Logística: o que é, por que é importante e quais os principais tipos. Disponível em: https://www.mutuus.net/blog/logistica-o-que-e/, 2021. Online; Acesso em 18 de jan. de 2023.

[6] R. S. de Oliveira. *Fundamentos dos Sistemas de Tempo Real*. Edição do Autor, 1ª edição, 2018.

[7] F. J. R. González e M. V. Villanueva. *Simulación del robot KUKA YouBot en el entorno de CoppeliaSim*. PhD thesis, Escuela Técnica Superior de Ingeniería, 2021.

[8] B. Williams. pyRTOS. Disponível em: https://github.com/Rybec/pyRTOS. Online; Acesso em 16 de fev. de 2023.