Trabalho Final de ATR (2025/2)

1. INTRODUÇÃO E CONTEXTO

O setor de mineração tem passado por uma transformação significativa com a adoção de tecnologias da Indústria 4.0, sendo os veículos autônomos uma das tecnologias promissoras dessa evolução. Nesse contexto, a adoção de veículos autônomos se mostra como uma solução estratégica para aumentar a eficiência, reduzir acidentes e melhorar o controle sobre os processos de extração e transporte. A implementação bem-sucedida desses sistemas exige uma base sólida em sistemas de tempo real, onde a corretude funcional deve ser garantida dentro de restrições temporais estritas.

O desenvolvimento de veículos autônomos apresenta desafios consideráveis, sendo necessário integrar diferentes fontes de informação, lidar com falhas em tempo real, realizar o planejamento de rotas e ainda garantir sincronização entre múltiplos caminhões operando em paralelo. Além disso, os sistemas precisam oferecer modos distintos de operação (manual e automático), interfaces seguras para interação com operadores locais e comunicação eficiente com o sistema central de gestão da mina.

Nesse trabalho, você e seu grupo farão o projeto e implementação uma versão simplificada do sistema de controle de um caminhão autônomo. O trabalho possibilitará o desenvolvimento e implementação de conceitos de programação concorrente em um caso realista de automação industrial em tempo real.

2. OBJETIVOS

Aplicar os conhecimentos da disciplina Automação em Tempo Real na implementação de um sistema de controle e monitoramento de um veículo autônomo de mineração, conforme a arquitetura proposta, contemplando tanto o controle embarcado do caminhão quanto a integração com um sistema central de gestão da mina.

Objetivos específicos:

- Implementar tarefas concorrentes para navegação, sensores, falhas e interface.
- Utilizar mecanismos de sincronização (mutex, semáforos, variáveis de condição).
- Implementar modos de operação (manual local e remoto automático).
- Desenvolver comunicação entre processos via IPC e pub/sub (MQTT).
- Construir interfaces gráficas simplificadas para operação local e gestão central.

3. ORGANIZAÇÃO DO DOCUMENTO

O Capítulo 4 apresenta as especificações gerais do sistema, que deve ser desenvolvido em duas etapas. O Capítulos 5 detalha as entregas da Etapa 1 e da Etapa 2. O Capítulo 6 apresenta instruções gerais para o desenvolvimento.

4. ESPECIFICAÇÕES GERAIS DO SISTEMA

O seu grupo foi contratado para desenvolver a aplicação embarcada no sistema de controle de um caminhão autônomo numa mina. O desenvolvimento será simplificado, contendo apenas algumas partes críticas. O sistema completo é ilustrado na Figura 1, em que cada tarefa é identificada com um retângulo, e a interação entre elas é representada por setas.

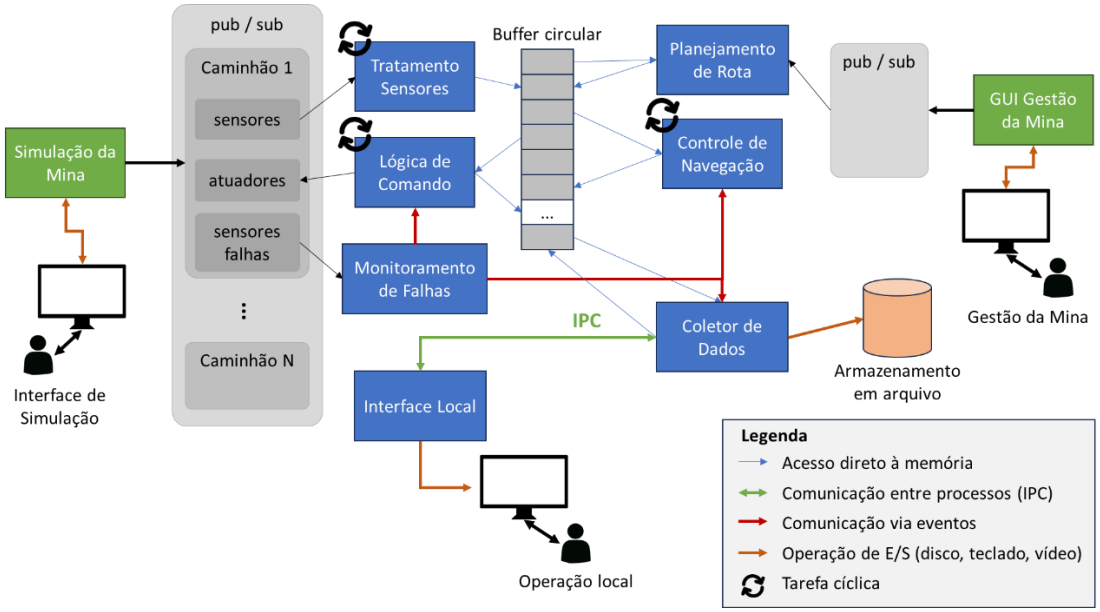

Figura 1 - Sistema simplificado de controle de um caminhão autônomo.

Esse será desenvolvido em duas etapas complementares:

- **Etapa 1**: definição da arquitetura completa do sistema e implementação de todas as tarefas em azul na Figura 1. Definição e implementação do buffer circular (e.g. 200 posições) e suas interações com as demais tarefas. As interações definidas pelas setas em vermelho (eventos) laranja (I/O) e em verde (IPC) não são implementadas nessa etapa. O servidor publisher e subscriber e as interfaces de Gestão da Mina e Simulação da Mina também não são implementadas nessa etapa.

- **Etapa 2**: implementação das interfaces de **Gestão da Mina** e **Simulação da Mina**. Essa implementação não precisa ser necessariamente em C++, você pode usar uma linguagem que simplifique o processo (e.g. Python com pygame). Implementação do servidor e clientes MQTT (*publisher* e *subscriber*) para comunicação com os sistemas. Testes de funcionalidade e escalabilidade do sistema, medindo o tempo gasto por cada tarefa em situações críticas.

A seguir serão apresentados com mais detalhes cada módulo do sistema. Esta é uma especificação básica de funcionamento do sistema. Existem várias nuances e detalhes que serão definidos por você e seu grupo para garantir a funcionalidade do sistema.

A Tabela 1 mostra os sensores e atuadores disponíveis no caminhão.

Tabela 1 - Sensores e atuadores do caminhão autônomo.

	Nome da variável	Tipo	Descrição
I	`i_posicao_x`	int	Posição do veículo no eixo x, com relação a um referencial absoluto em solo obtido pelos sensores de posição (e.g. GNSS).
I	`i_posicao_y`	int	Posição do veículo no eixo y, com relação a um referencial absoluto em solo obtido pelos sensores de posição (e.g. GNSS).
I	`i_angulo_x`	int	Direção angular da frente do veículo com relação à direção leste (leste = ângulo zero), obtido pelos sensores inerciais.
I	`i_temperatura`	int	Temperatura do motor (varia entre -100 e +200). Essa temperatura possui um nível de alerta se $T > 95\ ^\circ\mathrm{C}$ e gera defeito se $T > 120\ ^\circ\mathrm{C}$.
I	`i_falha_eletrica`	bool	Se for true, indica presença de falha no sistema elétrico do veículo.
I	`i_falha_hidraulica`	bool	Se for true, indica presença de falha no sistema hidráulico do veículo
O	`o_aceleracao`	int	Determina a aceleração do veículo em percentual (-100 a 100%)
O	`o_direcao`	int	Determina a angulação do veículo em graus (-180 a 180). Ao acelerar o veículo avança na direção do ângulo.

A leitura dos sensores de posicionamento é realizada pela tarefa **Tratamento Sensores**, que faz um processamento de filtragem de ruídos usando um filtro de média móvel de ordem M (média das últimas M amostras medidas). Essa tarefa alimentará um buffer circular para leitura das demais tarefas.

Os valores dos atuadores (saída) serão determinados pela tarefa de **Lógica de Comando**, que, a partir da leitura dos sensores (tratados) no buffer determinará o estado do veículo (manual, automático, funcionando, com defeito) e processará os comandos. A Tabela 2 mostra os possíveis estados e sua codificação na variável do programa.

Tabela 2 - Estados e comandos para o caminhão. Os estados são gerados pela tarefa "Lógica de Comando", a partir da leitura do buffer (dados tratados). Os comandos são realizados pelo operador na tarefa de "Interface Local".

	Nome da variável	Tipo	Descrição
E	`e_defeito`	bool	Estado que identifica a presença de defeito ou defeito não reconhecido pelo operador (1:defeito, 0: sem defeito)
E	`e_automatico`	bool	Estado que identifica o modo de operação do veículo (0: manual, 1: automático)
C	`c_automatico`	bool	Comando para passar o caminhão para o modo automático (true). O reset desse comando
C	`c_man`	bool	Comando para passar o caminhão para o modo manual (true).
C	`c_rearme`	bool	Comando para rearmar algum defeito que tenha ocorrido no caminhão.
C	`c_acelera`	-	Comando para acelerar o caminhão.
C	`c_direita`	-	Comando para alterar o ângulo (subtração) do caminhão.
C	`c_esquerda`	-	Comando para alterar o ângulo (soma) do caminhão.

Atenção: essa organização de variáveis é sugestiva, se quiser pode organizar essas informações de forma diferente no seu código, por exemplo, juntando vários valores booleanos numa variável inteira.

O sistema ainda conta com uma tarefa de **Monitoramento de Falhas**, responsável por ler as informações dos sensores de falhas (`i_temperatura`, `i_falha_eletrica`, `i_falha_hidraulica`, conforme Tabela 1) e disparar eventos para as tarefas de **Lógica de Comando**, **Controle de Navegação** e **Coletor de Dados**.

A tarefa de **Controle de Navegação** deve executar dois algoritmos de controle: de velocidade e de posição angular. O algoritmo deve ler os estados do buffer compartilhado para saber se o veículo está em modo manual ou automático. Se estiver em modo manual, o controle deve ser desligado. Em geral, quando em modo manual, o controlador coloca o setpoint do controlador para os valores atuais para evitar comportamento indesejado quando trocar de modo manual para automático novamente (*bumpless transfer*). Quando estiver em modo automático, o controlador deve ler o valor de setpoint estabelecido pela tarefa de **Planejamento de Rota** e executar a lógica de controle.

A tarefa de **Coletor de Dados** é responsável por obter os dados do estado do sistema, atribuir um *timestamp* e armazenar em arquivos de log no disco. Cada log de evento deve conter o *timestamp*, identificação (número) do caminhão, estado do caminhão, posição e a descrição do evento. O armazenamento no

disco deve ser feito com uma estrutura específica, detalhada na Tabela 3. Essa tarefa também é responsável por organizar os dados e estabelecer uma comunicação com a tarefa de **Interface Local**, de modo a enviar os estados atuais do caminhão e receber comandos do operador local e publicar no buffer.

A tarefa de **Interface Local** faz a interação com o operador dentro do caminhão, que deve ver na tela (e.g. terminal) os estados do sistema, principais medições e também aceitar comandos do operador, conforme a Tabela 2. Cada comando deve ser ativado por uma tecla do teclado.

A tarefa de **Simulação da Mina** produz os dados dos sensores a partir dos atuadores. Essa tarefa também deve possibilitar ao usuário, na interface de simulação, gerar defeito em algum caminhão. Os dados gerados dos sensores devem somar um ruído aleatório, de média nula, para que o algoritmo de tratamento sensores consiga tratar esse ruído. Note que na simulação você deve gerar valores de posição a partir de atuadores de aceleração e direção. Uma boa simulação implementa uma equação diferencial (ou equação de diferenças) para simular a dinâmica (inércia) de movimentação do caminhão. Para essa tarefa você pode utilizar qualquer linguagem de programação e API que facilite a implementação.

A tarefa de **Gestão da Mina** deve apresentar um mapa em tempo real da posição de todos os caminhões na mina. Essa tarefa deve conhecer a posição de todos os caminhões da mina, fornecido pelo sistema de planejamento de rotas, e prover uma interface amigável para o sistema. Nessa interface deve ser possível alterar o setpoint de posição de cada caminhão, de modo que o caminhão possa seguir um novo setpoint. Para essa tarefa você pode utilizar qualquer linguagem de programação e API que facilite a implementação.

A tarefa de **Planejamento de Rota** fará a leitura dos pontos inicial e final, definidos pelo sistema de Gestão da Mina, e definirá os setpoints de velocidade e posição angular para o sistema de Controle de Navegação.

5. ENTREGAS DO TRABALHO

As entregas estão especificadas a seguir.

1) **ETAPA 1 (03/11/2025)**: definição da arquitetura
   a) Figura da arquitetura detalhada do sistema, especificando linguagens de programação, ferramentas de sincronização, APIs e demais detalhes que serão implementados.
   b) Breve documento com uma apresentação parcial do sistema.
2) **ETAPA 2 (26/11/2025)**: sistema completo
   a) Todos os arquivos do sistema.
   b) Breve documentação, técnica, com descrição geral do sistema e explicando partes críticas do código.
   c) Vídeo de apresentação do sistema funcionando (~10 min).
   d) Apresentação do código diretamente para o professor.

6. INSTRUÇÕES GERAIS

1. O trabalho pode ser feito por grupos de até 3 pessoas. Na apresentação do código, será perguntado o papel de cada integrante no trabalho.

2. O trabalho deve ser submetido via Moodle, dentro do prazo de entrega, como um arquivo ZIP contendo todo o conteúdo.
3. Cuidado com referências absolutas a pastas, use sempre referências relativas de onde o programa principal está rodando.
4. Cuide da documentação. A sua documentação, em formato PDF, deve ter instruções de como compilar e como rodar o código.
5. Faça com que todo o código execute a partir de um único arquivo, executando todos os processos necessários.
6. Use ferramentas e conceitos de programação e desenvolvimento de sistemas para fazer uma aplicação com o máximo de profissionalismo possível, com boa organização.
7. Teste cada módulo do seu programa e cada interação com outros módulos. Faça vários testes cuidadosos no código.
8. Para que o código possa rodar no computador do professor sem contratempos, sugiro que o ambiente de programação seja realizado em um container (e.g. docker). Nesse caso, inclua o Docker File para que o professor possa gerar o ambiente de desenvolvimento.
9. Ponto extra: o seu trabalho pode receber até 5 pontos extras, a critério do professor, caso implemente melhorias, acrescentando outros módulos no programa (ex. sistema anticolisão, interface gráfica diferenciada, etc.).

Item	Pontuação
Entrega da Etapa 1 e da Etapa 2 no prazo	25
Entrega apenas da Etapa 2	15
Entrega apenas da Etapa 1	0