UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PLATAFORMA DE DESENVOLVIMENTO DE SISTEMAS DE CONTROLE: ESTUDO DE CASO APLICADO EM UM AGV

LUAN DIAS RODRIGUES

ORIENTADOR: **PROF. DR. ORIDES MORANDIN JUNIOR**

São Carlos - SP

Março/2026

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PLATAFORMA DE DESENVOLVIMENTO DE SISTEMAS DE CONTROLE: ESTUDO DE CASO APLICADO EM UM AGV

LUAN DIAS RODRIGUES

Projeto de Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.

Orientador: **Prof. Dr. Orides Morandin Junior**

São Carlos - SP

Março/2026

Folha de Aprovação

Defesa de dissertação de mestrado do(a) candidato(a) Luan Dias Rodrigues, realizada em 04/03/2026

Comissão Julgadora

Prof(a) Dr(a) Orides Morandin Júnior (UFSCar)

Prof(a) Dr(a) Valdinei Luis Belini (UFSCar)

Prof(a) Dr(a) Zilda de Castro Silveira (USP)

O relatório de defesa assinado pelos membros da comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação

Dedico este trabalho aos meus pais, Sebastião Rodrigues da Silva e Maria Estela Dias da Silva, que, mesmo enfrentando desde a infância a necessidade constante de trabalhar, jamais desistiram dos estudos. Ao formarem sua família, fizeram da educação a principal prioridade para os filhos. Também dedico minha tia Vilmacis Leão da Silva, que por meio da simplicidade de seu sorriso firmava um conforto e ensinamento aos nossos corações.

AGRADECIMENTO

Gostaria de expressar minha profunda gratidão, primeiramente, aos meus pais, Sebastião Rodrigues da Silva e Maria Estela Dias da Silva, que desde o momento em que realizei minha inscrição, não mediram esforços para me apoiar e auxiliar nesta nova jornada. Agradeço também ao meu orientador, Professor Doutor Orides Morandin Junior, por ser mais do que um guia acadêmico. Sua orientação foi essencial não apenas para o desenvolvimento deste trabalho, mas também para meu crescimento pessoal. Em diversos momentos, me ajudou a alcançar meus objetivos e me incentivou a seguir meu sonho de me tornar professor. Meu reconhecimento vai também para Wallace Reis, que, mesmo à distância, foi um grande orientador em momentos de necessidade. Sua dedicação e apoio foram fundamentais para o desenvolvimento deste projeto. Por fim, agradeço ao meu companheiro de laboratório, o mestrando Vinicíus Martins, que teve papel essencial na correção e desenvolvimento dos testes do AD01, sempre disposto a colaborar e compartilhar seus conhecimentos.

RESUMO

A robótica móvel tem ganhado destaque na indústria, especialmente com o uso de AGVs (Automated Guided Vehicles), que contribuem para maior eficiência e segurança nos processos produtivos. Para que o AGV siga corretamente uma linha, utiliza-se um controlador, sendo o PID (Proporcional, Integral e Derivativo) um muito utilizado por sua simplicidade e eficácia. Neste trabalho, foi desenvolvida uma plataforma de desenvolvimento voltada para sistemas de controle, aplicada ao projeto de um AGV seguidor de linha. A plataforma permite a implementação e testes de funcionalidades, além de garantir a continuidade do projeto por meio de um padrão documental composto por sete documentos e uma arquitetura de software bem definida. Ensaios práticos com o AGV utilizando o controlador PID demonstraram o funcionamento do sistema e validaram a proposta da plataforma como ferramenta para manter o legado e facilitar futuras evoluções.

Palavras-chave: Controlador PID, Robótica Móvel, UML, Arquitetura de Software, Documentação Técnica.

ABSTRACT

Mobile robotics has gained prominence in the industry, especially through the use of AGVs (Automated Guided Vehicles), which contribute to greater efficiency and safety in production processes. To ensure accurate line-following behavior, AGVs rely on controllers, with the PID (Proportional, Integral, and Derivative) being the most commonly used due to its simplicity and effectiveness. This work presents the development of a control systems platform applied to a line-following AGV project. The platform enables the implementation and testing of functionalities, while also ensuring project continuity through a standardized documentation model consisting of seven technical documents and a well-defined software architecture. Practical experiments with the AGV using the PID controller demonstrated proper system operation and validated the platform as an effective tool for preserving project legacy and supporting future enhancements.

Keywords: PID Controller, Mobile Robotics, UML, Software Architecture, Technical Documentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGV	Veículos Auto Guiados (do inglês: Automated Guided Vehicles)
RGB	Vermelho, Verde Azul (do inglês: Red Greem Blue)
PID	Proporcional Integral Derivativo
SOA	Arquitetura Orientada a Serviços (do inglês: Service Oriented Architecture)
AD01	AGV Diferencial 01
FPS	Frames Por Segundo
SO	Sistema Operacional
ARM	Máquina RISC Avançada (do inglês: Advanced RISC Machine)
MAE	Valor Máximo do Erro Absoluto (do inglês: Maximum Absolute Error)
MSE	Erro Quadrático Médio (do inglês: Mean Squared Error)
RMSE	Média Quadrática do Erro (do inglês: Root Mean Squared Error)
ISE	Integral do Erro Quadrático (do inglês: Integral of Squared Error)
ITSE	Integral do Erro Quadrático Ponderado pelo Tempo (do inglês: Integral Time Square Error)
UML	Linguagem Modelada Unificada (do inglês: Unified Modeling Language)

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

A robótica móvel é um ramo que possui uma certa notoriedade para as indústrias. SABATTINI et al. (2013) destaca que a utilização de veículos autoguiados (Automated Guieded Vehicles (AGVs)) nas indústrias resulta em uma melhor eficiência e segurança nos sistemas de produção, quando consideramos a presença de rotas programadas, sensores de segurança e a redução de falha humana. Segundo RIBEIRO (2022) a implementação de AGVs em ambiente fabril pode ser vantajosa para reduzir o custo de mão de obra, ter uma flexibilidade para alterar o percurso, maior controle das atividades e custo energético menores.

Os AGVs podem ser classificados perante sua carga, podendo ser veículos de reboque, veículos de carga unitária, porta-paletes e empilhadeira, por exemplo. Em relação ao sistema de navegação os AGVs se dividem em orientação a laser, orientação de seguimento de linha, orientação de ponto magnético e orientação sobre código de barras (Lynch et al., 2018).

OLIVEIRA, REIS e MORANDIN JUNIOR (2019) apresentam que atualmente os sensores para controle de movimento mais populares e acessíveis para um robô AGV são os sensores magnéticos. Mas, em contrapartida, para se utilizar esse sistema é necessário

instalações pré-planejadas de suas rotas, o tornando pouco flexível. Contudo, como solução do problema os autores propõem o uso de uma câmera USB com sensor RGB (red-green-blue) como sensor do AGV, validando seu uso em um controlador Proporcional Integral Derivativo (PID).

Os controladores PID possuem um bom desempenho em uma vasta gama de condições de operação, além de possuir uma facilidade de utilização, pois, para implementar um controlador PID é necessário determinar somente três parâmetros, o ganho proporcional, designado Kp, ganho integral ou integrativo Ki e o ganho derivativo Kd. O processo que determina os valores dos ganhos é denominado sintonia PID (Dorf; Bishop, 2018).

Visto a importância dos AGVs, a proposição de uma plataforma para desenvolvimento de sistemas de controle para AGVs se torna viável, para assim, por meio de uma arquitetura definida e um correto padrão de documentação, possibilitar a análise de quais técnicas e controladores apresentam uma maior eficiência e resultam em um comportamento mais desejável do robô móvel.

Uma plataforma de sistemas de controle integra simulação, prototipagem e testes de algoritmos de controle, possibilitando o rápido desenvolvimento e verificação de sistemas, permitindo a realização dos testes em tempo real (Vekić et al., 2012). A plataforma de desenvolvimento de sistemas de controle também permite a interação entre hardware e software, facilitando o desenvolvimento de funções e algoritmos de sistemas de controle (Hagengren; Sandberg, 1986).

Um ponto importante a ser destacado ao desenvolver tal plataforma é a necessidade de documentá-la. Pois até mesmo um sistema simples deve ser modelado antes de implementado. E como projetos são dinâmicos, ou seja, estão sempre em mudanças e atualizações, documentá-los se torna essencial e, quando realizado de forma correta, detalhada, precisa e atualizada, resulta em uma maior facilidade do projeto ser corrigido ou modificado, sem gerar erros antigos (Guedes, 2018).

A plataforma proposta neste trabalho foi desenvolvida no laboratório TEAR, sendo elaborada desde 2018. Em razão da pandemia e da dificuldade de uma padronização na documentação, o projeto não pôde ser finalizado adequadamente, sendo

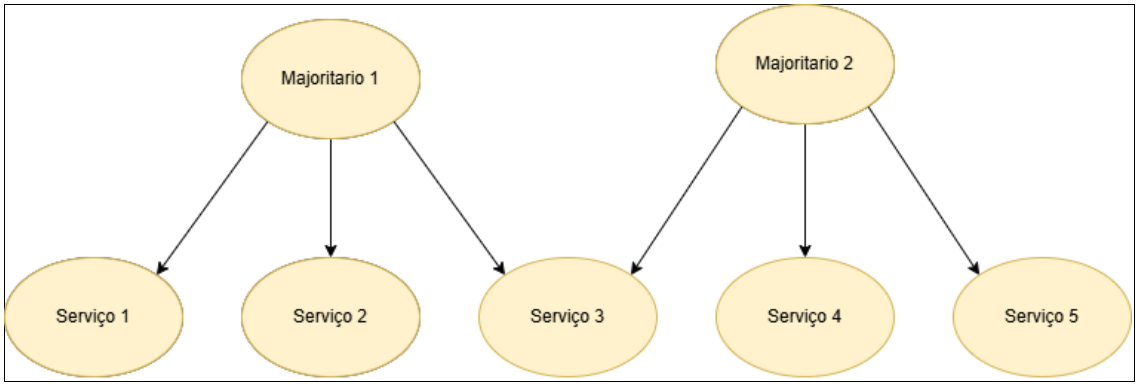
necessário recomeçá-lo com uma documentação que permite a facilidade de interpretação, manutenção e suporte para novas atualizações.

A arquitetura SOA (Service Oriented Architecture) é uma opção adequada para o desenvolvimento da plataforma de forma a atender os requisitos de projeto elencados inicialmente, sejam eles, reuso de código, aplicação em diferentes plataformas de computação, adaptabilidade a novos sensores e novas funções. SOA é ideal para sistemas reconfiguráveis, podendo oferecer serviços de forma estática ou dinâmica, auxiliando na mudança do sistema (Yen et al., 2008).

O paradigma SOA, em sua essência, tem o objetivo de criar sistemas modularizados ao invés de criar uma grande aplicação em um único bloco. A arquitetura propõe que o sistema seja dividido em várias funções, disponibilizadas em forma de serviços. Isso resulta em um baixo acoplamento, reduzindo resíduos de códigos e redundâncias de funcionalidades (Silva, 2006).

A arquitetura SOA, exemplificada na Figura 1.1, facilita a modularidade e manutenção do código, pois, ao ser necessário alterar algum serviço, apenas um bloco ou arquivo de código precisa ser modificado ou desenvolvido. Por exemplo, caso o Serviço 2 seja um controlador PID, para adicionar um controlador Fuzzy basta escrever a lógica Fuzzy no Serviço 2.

1.1 – Arquitetura SOA



1.2 Motivação

O uso de uma plataforma de desenvolvimento de sistemas de controle garante realizar testes adequados para o estudo de caso específico, o AGV, auxiliando na definição dos melhores parâmetros e técnicas para o produto final. No decorrer do desenvolvimento dois fatores são de extrema importância: a escolha de uma arquitetura adequada e a documentação.

Como apresentado, a arquitetura SOA objetiva criar sistemas modularizados, o que torna muito vantajoso no presente contexto, pois isso auxilia na manutenção e em novas versões do projeto. Em sistemas de controle, temos diversos possíveis controladores, então acoplar o controlador em um único bloco acelera o desenvolvimento do projeto, não sendo necessário modificar o resto do projeto.

A arquitetura SOA auxilia em uma maior organização do código, salientando que a mesma reduz redundância de funcionalidades e resíduos de códigos, o que se torna útil para sistemas de controle, além da facilidade de realizar testes, podendo direcionar testes unitários para cada serviço isolado.

Para uma plataforma de desenvolvimento é muito importante que todos que irão ler a documentação compreendam-na claramente. Então é de extrema relevância que o desenvolvimento dos documentos seja realizado de forma correta, abrangendo vários níveis de conhecimentos, possibilitando que até mesmo pessoas que não sejam programadores entendam os objetivos e o funcionamento dos serviços propostos.

Quando a documentação é feita em diferentes níveis, torna-se muito mais fácil encontrar falhas e novas opções para possíveis atualizações, além auxiliar a compreender a limitação do método utilizado, visto que não há dependência exclusiva das linhas de código para se entender o método.

O uso de uma arquitetura adequada e uma documentação robusta além de favorecer o desenvolvimento do projeto e a manutenção do código, também auxilia a manter um legado no projeto, visto que, novas versões poderão ser realizadas a partir da documentação anterior do projeto.

O legado se torna um fator muito importante para o laboratório TEAR, pois o projeto vem sendo desenvolvido desde 2018 e ainda não foi implementado. Por isso a criação de uma plataforma de desenvolvimento documentada se torna importante, para garantirmos que além do funcionamento correto do AGV ele ofereça suporte para que o desenvolvimento continue.

1.3 Justificativa

Um código modular apresenta uma melhor compreensão e auxilia a manutenção, além de facilitar a implementação de novos métodos e melhorar a qualidade geral do código, sendo muito favorável a implementação em sistemas de grande escala (Teymourian; Izadkhah; Isazadeh, 2022).

Pela modularização aprimorando a compreensão e a capacidade de manutenção do código, o projeto apresenta uma integração mais fácil dos serviços, resultando em uma maior produtividade e eficiência no desenvolvimento do sistema, sendo muito útil em sistemas mais complexos (Barros; Ouyang; Wei, 2020).

O sistema modular facilita muito o desenvolvimento. A ideia é dividir o programa em blocos pequenos e bem definidos. Com essa organização, a manutenção se torna mais simples e os erros diminuem. Outra vantagem é que o código pode ser reutilizado e, como os detalhes ficam encapsulados, fica mais fácil entender como tudo funciona (Brown, 1978).

O uso do SOA oferece uma flexibilidade e escalabilidade do sistema, permitindo uma comunicação perfeita entre serviços e usuários, suportando o encapsulamento, reutilização e composição de componentes (Mendoza-Pitti et al., 2021).

A arquitetura SOA permite obter um sistema configurável, podendo selecionar e compor serviços de forma dinâmica assim atendendo constantes mudanças, aprimorando a adaptabilidade e a eficiência (Yen et al., 2008).

As características do SOA permite oferecer uma alta flexibilidade, assim auxiliando que as soluções continuem respondendo de forma eficaz a mudanças, sendo essas

mudanças a justificativa do SOA oferecer uma boa flexibilidade no projeto (Will; Koppen; Saake, 2014).

Em sistemas embarcados, SOA facilita no quesito de componentes flexíveis e autônomos, permitindo funcionalidades plug-and-play, autoconfiguração e diagnósticos, além de suportar comunicação em tempo real, possibilitando uma comunicação e interação eficaz entre dispositivos no ambiente industrial (Cucinotta et al., 2009).

Para um AGV, adotar uma arquitetura modular baseada em SOA é uma escolha estratégica, pois combina a flexibilidade e a comunicação em tempo real necessárias ao sistema. Essa abordagem é especialmente válida, já que o controle de tempo é um requisito crítico e a necessidade de alterar ou implementar novos módulos é frequente; assim, a SOA organiza o projeto em serviços independentes, facilitando a manutenção, a reutilização e a evolução do AGV sem comprometer seu desempenho.

1.4 Objetivos

O objetivo geral do presente trabalho é a criação de uma plataforma de desenvolvimento para sistemas de controle, aplicado a um AGV denominado AD01 (AGV Diferencial 01). Para que o objetivo final seja atingido, os seguintes objetivos específicos devem ser completados:

- Utilizar uma arquitetura modular.
- Determinar os documentos necessários para o desenvolvimento.
- Utilizar o controlador PID no sistema de controle.
- Realizar a sintonia PID no AD01.
- Validar a plataforma de desenvolvimento de sistemas de controle.

1.5 Delimitações do trabalho

- Não será abordado métodos de sintonia do controlador.

- O AGV não constará com uma modelagem matemática.
- O controlador será limitado ao PID.
- Não será apresentado os testes de validação de software e hardware.

1.6 Método de pesquisa e desenvolvimento

Seguindo a classificação quanto à natureza da pesquisa proposta por Silva e Menezes (2001), esta pesquisa possui natureza aplicada, pois visa viabilizar uma plataforma de sistemas de controle em um AGV, gerando conhecimentos voltados para uma aplicação prática. Ao analisar o comportamento do AGV, a pesquisa também apresenta características descritivas.

Quanto à classificação de Silva e Menezes (2001) referente aos procedimentos técnicos, trata-se de uma pesquisa experimental, justificada pela realização de testes no AGV com manipulação de variáveis. Além disso, configura-se como um estudo de caso, uma vez que a plataforma é aplicada a um AGV específico, em um contexto realista.

Em relação à abordagem do problema, de acordo com Silva e Menezes (2001), a pesquisa é majoritariamente quantitativa, ao analisar dados como erro e desempenho do AGV. Contudo, também apresenta características qualitativas, ao validar a modularidade da plataforma desenvolvida.

1.7 Organização do trabalho

O primeiro capítulo apresenta uma introdução sobre o tema, iniciando com a contextualização em sequência mostrando a motivação e justificativa que demonstram a importância do tema abordado, os objetivos gerais e específicos do trabalho são expostos consequentemente junto com as delimitações que a presente pesquisa obteve, finalizando o primeiro capítulo é abordado os métodos de pesquisa e desenvolvimento. O capítulo 2 desenvolve todo conhecimento teórico necessário para o compreendimento do trabalho, e que auxilie na validação dos resultados, e, em sequência, serão apresentados trabalhos

que estão relacionados à pesquisa. A proposta está descrita no capítulo 3, de acordo com os objetivos gerais e específicos da pesquisa. Para o capítulo 4 está destinada a descrição dos métodos de testes e validação dos resultados. Os resultados do trabalho e discussões estão descritos no capítulo 5, apresentando as considerações finais e propostas de trabalhos futuros.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

2.1 Considerações iniciais

O presente capítulo apresenta uma base teórica para um entendimento básico sobre o tema, além de apontar as bases que o justificam e apontam sua importância por meio de trabalhos publicados pela comunidade científica. Para isso, o capítulo é dividido em quatro partes. A primeira sendo a atual, a apresentação. A segunda parte a fundamentação teórica apresentando a base teórica para o entendimento da pesquisa. A terceira, os trabalhos correlatos, em que são apresentadas pesquisas semelhantes, assim justificando um interesse da comunidade científica sobre o assunto. E por fim as considerações finais do capítulo.

2.2 Fundamentação teórica

2.2.1 Sistemas embarcados

Um sistema embarcado é considerado como qualquer computador que seja uma parte de um sistema maior e depende do seu próprio microprocessador, projetado para realizar funções específicas desse sistema (Wolf, 2002).

Os sistemas embarcados são uma integração de componentes mecânicos e eletrônicos, denominados hardwares, com as funções orientadas por informações, denominada software, desempenhando um papel importante na mecatrônica, utilizando sistemas mecânicos, atuadores e processamento digital de informações para sistemas de controle (Bertozzi; Chen; Zingaretti, 2018).

Em sistemas de controle, os sistemas embarcados são cruciais, permitindo o processamento de dados em tempo real, integração de sensores e o controle do atuador, garantindo a operação adequada, além de facilitar o monitoramento e a reconfiguração, auxiliando na confiabilidade e desempenho do sistema em ambiente dinâmico (Khalgui et al., 2011).

2.2.2 Sistema Operacional

Um Sistema Operacional (SO) é definido como um software que gerencia os recursos de hardware, auxilia na execução de aplicativos, além de possibilitar acesso a recursos como entrada e saída de informação e sistemas de arquivos, sendo responsável como a interface entre hardware e software (Geer, 2009).

Para sistemas embarcados existem SOs específicos, sendo SOs especializados e projetados para gerenciar recursos de hardware e software do sistema embarcado, permitindo a operação eficiente do disposto, e geralmente com base em plataformas de código aberto como Linux (Ortiz, 2001).

Os SOs embarcados executam normalmente uma versão reduzida do Linux, oferecem baixos custos computacionais, flexibilidade e transferência eficiente de dados

em tempo real, tornando uma vantagem quando comparado aos SOs comuns (Alkar; Karaca, 2009).

2.2.3 Controlador PID

Segundo Dorf e Bishop (2020), o controlador PID é uma forma de controle muito utilizada em processos industriais, sendo um controlador de três termos, o proporcional representado por KP, integral representado por KI, e derivativo representado por KD, com sua função transferência:

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s \quad (1)$$

O KP (Proporcional) afeta a resposta do erro atual. Ao aumentar essa constante obtém-se uma melhora na rapidez da resposta do sistema, mas em sua contrapartida valores muitos elevados podem causar instabilidade e oscilações.

O KI (Integral) lida com o erro acumulado ao longo do tempo. Essa constante é responsável por eliminar o erro estacionário, mas seu mau uso ocasiona instabilidades devido ao acúmulo rápido de erro.

Enfim, o KD (Derivativo) responde a variação do erro. O uso dela permite reduzir oscilações e melhorar a estabilidade do sistema, mas o aumento significativo dela torna o controlador sensível a ruídos.

Dorf e Bishop (2020) apresentam uma tabela demonstrando o efeito do aumento dos ganhos do PID em uma resposta ao degrau e explica que o método para determinar estas constantes é denominado sintonia PID.

Tabela 2.1 Efeito do Aumento dos Ganhos do PID (DORF e BISHOP, 2020)

Ganho do PID	Máxima Ultrapassagem Percentual	Tempo de Acomodação	Erro em Regime Estacionário
Aumentando K_P	Aumenta	Impacto mínimo	Diminui
Aumentando K_I	Aumenta	Aumenta	Erro em regime estacionário nulo
Aumentando K_D	Diminui	Diminui	Nenhum impacto

A função transferência do controlador PID apresentada pela Equação 1 é utilizada em sistemas contínuos. Para a implementação digital do controlador PID é necessário aplicar a transformada z. Segundo Dorf e Bishop (2020), os controladores precisam estar em um período fixo chamado período de amostragem, representado pela letra T, realizando um ciclo de leitura e ação do controlador.

Aplicando a transformada Z, Dorf e Bishop (2020) apresentam a função transferência no domínio z do controlador PID como:

$$G_c(z) = K_P + \frac{K_I T_Z}{z-1} + K_D \frac{z - T_Z}{T_Z} \quad (2)$$

Dorf e Bishop (2020) apresentam que por meio da função transferência é retirado o algoritmo completo da equação, assim sendo possível implementá-la em microcontroladores ou microprocessadores, como processadores da arquitetura ARM.

2.2.4 Arquitetura ARM

A arquitetura ARM (Advanced RISC Machine), uma família de arquitetura de processadores que usam o conjunto de instruções reduzido (RISC), apresenta uma boa eficiência e baixo consumo de energia e é amplamente utilizado em sistemas embarcados (Jiang et al., 2024).

A arquitetura ARM oferece um alto desempenho, baixo consumo de energia, tamanho pequeno da matriz e paralelismo eficiente por meio de técnicas como tempo de execução variável, à tornando ideal para sistemas embarcados (GOODACRE; SLOSS, 2005).

2.3 Trabalhos correlatos

2.3.1 Trabalhos do TEAR

Em uma revisão sistemática Reis e Morandin Junior (2021) pesquisa os principais sensores utilizados em AGV, diversos sensores são encontrados e dentre eles estão os já

apresentados Lidar, Sensor Magnético e Câmera, mas outros sensores aparecem com uma frequência menor como o Sensor Óptico, Bluetooth e sensor de distância a laser. Por mais que vários sensores sejam apresentados nos 31 artigos dispostos, o que mais se destaca é o uso de câmeras sendo a forma de sensor para AGV em 15 artigos.

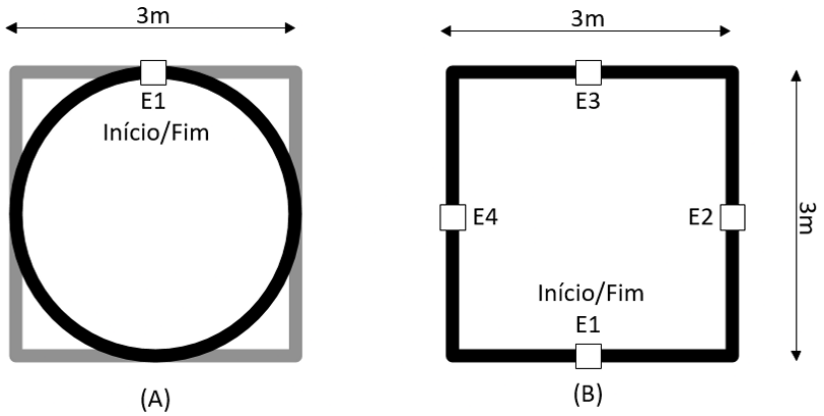
Outro ponto a ser destacado da pesquisa é a tecnologia utilizada, sendo a mais utilizada nas câmeras a faixa de cores. Outra tecnologia identificada na revisão sistemática, mas apenas quatro vezes, foi o QR Code.

A respeito de sistemas de controle de posição de um AGV, Reis, Couto e Junior (2023) apresentam por revisão sistemática o acionamento diferencial com 62%, de um total de 92 artigos de AGVs porém, 49 desses trabalhos obtiveram apenas resultados simulados. A revisão apresentou também um interesse na interação de técnicas de controle inteligente com outras estratégias de controle dispostas em um terço dos artigos.

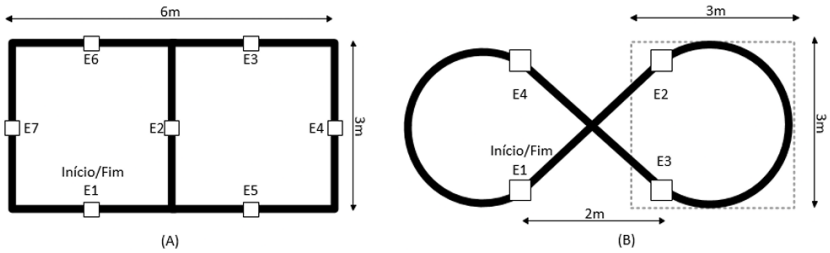
Reis, Oliveira e Morandin Junior (2019) e Oliveira, Reis e Morandin Junior (2019) em ambos artigos propõem o uso de uma câmera RGB como o sensor para calcular o erro do AGV. O primeiro artigo obteve como resultado que o uso da câmera RGB como sensor traz uma resposta competitiva em termos de tempo e resolução na medição. Como resultado do segundo artigo a diferentes valores da resolução da câmera obtêm um impacto significativo na malha de controle. Mas vale salientar que o sistema se aproximou da estabilidade marginal devido ao fato de se utilizar um Arduino Mega para o sistema de controle do AGV e um computador portátil para o processamento de imagem.

Oliveira (2018) utiliza o Unity3D para realizar uma representação dinâmica de um AGV, tendo sua proposta defendida pelo auxílio em sintonização, podendo ser realizada em ambiente simulado e empregada no ambiente real. Para os testes e validação as simulações foram realizadas em quatro pistas, as duas primeiras circulares e quadrada, e as duas últimas em formato de oito, um digital e um infinito. As pistas estão apresentadas na Figura 2.1 e Figura 2.2. Uma das principais vantagens de se utilizar a simulação é a facilidade de criar vários ambientes diferentes.

2.1 Pistas Circular e Quadrada (OLIVEIRA, 2018)



2.2 Pistas em 8 digital e infinito (OLIVEIRA, 2018)



Para realizar os testes da simulação, dos controladores foram utilizados, o PID e o Fuzzy, sendo testados em todas as pistas e também simulando a presença de diferentes cargas sobre o AGV. Uma das principais contribuições do trabalho foi a relevância do uso de simuladores na sintonia podendo se tornar em um auxiliador na sintonia real do AGV. Vale salientar que a pesquisa não obteve uma comparação com o veículo real.

Reis (2023), para aumentar a exatidão de na posição do AGV, utiliza um controlador Fuzzy-adaptativo em cascata utilizando múltiplos sensores, para que o sistema de controle se adapte em relação às diferentes condições de carga. O trabalho utiliza uma câmera como sensor de posição do AGV e considera o sistema como multivariável. Os testes foram realizados em um AGV físico denominado AD02, uma versão do AD01 em escala de proporções menores, sendo controlado por uma Raspberry Pi 3B.

Para a comparação dos resultados foi utilizado um controle PID e os indicadores de desempenho valor máximo do erro absoluto (MAE - Maximum Absolute Error), erro

quadrático médio (MSE - Mean squared Error), média quadrática do erro (RMSE - Root Mean Squared Error), integral do erro quadrático (ISE – Integral of Squared Error), integral do erro quadrático ponderado pelo tempo (ITSE – Integral Time Square Error), além dos índices IAE e ITAE.

2.3.2 Uso da arquitetura SOA

A arquitetura SOA é usada em projetos como Ke et al. (2023), em que apresenta um trabalho que propõe o uso da arquitetura para uma coleta inteligente de dados para uma integração eficiente com aplicações reais, melhorando a qualidade, confiabilidade e segurança dos sistemas, sendo um auxiliador otimizando a coleta e análise de dados com sistemas que usam inteligência artificial.

Kyösti e Lindström (2022) analisam a possibilidade do uso do SOA com suporte para microsserviços em desenvolvimento e operação de soluções de automação industrial. Como benefícios apresentados na proposta os autores apresentam a redução do tempo e esforço na engenharia, melhoria na flexibilidade e escalabilidade, aumento na segurança cibernética, facilidade em integrar novos módulos além de evitar uma dependência de fornecedores.

Kyösti e Lindström (2022) abordam que o uso do SOA é promissor em soluções de automação, mas ainda assim exige um investimento em capacitações técnicas para que os benefícios sejam realmente alcançados, apresentando também uma alternativa para indústria 4.0.

Chengmeng e Zhen (2010) apresentam de forma conceitual a proposta de uma construção de sistema de gestão da informação empresarial com uma arquitetura baseada em SOA, a proposta do trabalho objetiva uma melhor eficiência, flexibilidade e integração dos processos empresariais, as principais aplicações citadas são para instituições educacionais, empresas de produção e planejamento e indústrias como petróleo e manufatura.

Chengmeng e Zhen (2010) apresentam que os principais problemas encontrados para o desenvolvimento de sistemas de gestão de informação são, sistemas legados com

baixa interoperabilidade, redundância de dados e dificuldades de manutenção e altos custos operacionais e de atualização. Utilizando a arquitetura SOA é possível sanar os problemas apresentados.

Outro exemplo de sistemas utilizando SOA é proposto por Ying-pei e Ting-ting (2011) defendendo o uso da arquitetura como uma alternativa para superar desafios de fragmentação e isolamento de dados. O trabalho apresenta a proposta do sistema com o intuito de melhorar a eficiência administrativa das universidades.

Ying-pei e Ting-ting (2011) propõem o modelo de integração baseado em SOA para conectar sistemas administrativos independentes e promover compartilhamento de dados entre departamentos, visto que um dos grandes problemas nos sistemas apresentados são sistema desenvolvidos de forma isolada, não apresentando uma padronização entre os departamentos e a redundância e inconsistência de dados.

Bhadoria et al. (2022) utilizam um recurso matemático que auxilia a distribuição dos recursos disponíveis no sistema, a arquitetura apresentada é o SOA com cinco camadas, sendo elas, operacional, componentes de serviço, serviços, processos de negócio e consumidor. O foco do trabalho é alocar recursos críticos dinamicamente em ambientes IoT.

O trabalho de Bhadoria et al. (2022) apresenta aplicações da arquitetura SOA em sistemas distribuídos inteligentes, como os apresentados em IoT, com múltiplos dispositivos e a necessidade constante de uma comunicação eficiente.

2.3.3 Uso da arquitetura SOA em sistemas embarcados

Em sistemas embarcados, um exemplo muito interessante é proposto por Karademir e Cetin (2007), o qual aborda como a Arquitetura SOA pode ser utilizada afim de melhorar a interoperabilidade entre sistemas de satélite. O artigo argumenta que como a arquitetura é modular ela pode ser eficaz em sistemas de tempo real.

A proposta de Karademir e Cetin (2007) é a utilização do SOA para facilitar a comunicação entre diferentes sistemas de satélites, onde em vários casos apresentam protocolos diferentes. Como benefício do uso da arquitetura, os autores apresentaram a

reutilização de componentes de software, facilidade de manutenção e escalabilidade, melhoria na segurança e confiabilidade dos sistemas espaciais.

Em sistemas embarcados, outro exemplo do uso do SOA foi proposto por Rodrigues et al. (2011) em que se discute o uso da arquitetura em sistemas embarcados, o foco foi para um sistema de aviação não tripulada (UAV). O trabalho explora o objetivo de modularizar e integrar componentes de sistemas embarcados complexos e melhorar a reutilização, manutenção e abstração em sistemas que necessitam de sistemas rigorosos de confiabilidade e tempo real.

Rodrigues et al. (2011) apresentam uma abordagem promissora para sistemas embarcados, mas com ressalvas, recomendando o uso do SOA seletivo em seções não críticas em sistemas complexos, apresentando uma forte recomendação por permitir a comunicação entre subsistemas.

Püllen et al. (2024) apresentam uma proposta do uso de uma arquitetura ramificada do SOA, denominada ASOA (*Automotive Service-Oriented Achitecture*), uma arquitetura adaptada do SOA para o contexto automotivo. O trabalho apresenta a aplicação para veículos de passeio, os conceitos podem ser explorados e aplicados para AGV e VARs (*Vehicular Autonomous Resources*).

O uso da arquitetura proposta por Püllen et al. (2024) possui o objetivo principal de apresentar uma alta confiabilidade e comportamento em tempo real, além de adaptabilidade e manutenção eficiente, junto com um desacoplamento funcional.

Rumez et al. (2020) apresentam o uso do SOA no setor automotivo como uma ferramenta para auxiliar o sistema de comunicação dentro do veículo. Focando em apresentar como os módulos independentes se comunicam entre si com o uso da arquitetura.

No laboratório TEAR, as pesquisas desenvolvidas utilizam, em sua maioria, a arquitetura SOA no AGV, tornando um padrão no desenvolvimento de software do laboratório. O uso da arquitetura se inicia pela proposta de Souza (2021) que propõe de forma teórica o uso da arquitetura para o desenvolvimento do software do AGV. O trabalho de Souza serve como grande norteador para a construção de todo software do AD01.

2.4 Considerações finais

O uso do controlador PID apresenta um bom desempenho em uma vasta gama de aplicações. Mas vale salientar que por ser um sistema de embarcado a técnica a ser abordada é por controle discreto. Para sua aplicação, além da sintonia dos ganhos Kp, Ki e Kd é necessário ajustar o também o período de amostragem. Mas ainda assim o PID é controlador de fácil implementação, por isso sendo uma escolha viável como ponto de partida para uma plataforma de desenvolvimento de sistemas de controle.

O uso de câmera como sensor do AGV vem sendo o método mais utilizado nos trabalhos acadêmicos, o que pode ser justificado por sua versatilidade, fácil implementação e precisão suficiente para as aplicações. A construção do AGV apresenta uma base acadêmica como referência, não sendo necessária a construção do zero, sendo já defendido o uso da arquitetura SOA em AGVs e o uso de simulações para entender melhor o comportamento do robô.

A arquitetura SOA pode ser usada em sistemas embarcados, sendo uma arquitetura modular, de fácil implementação e manutenção se torna uma ótima alternativa no desenvolvimento de uma plataforma.

Com a base teórica apresentada e os trabalhos teóricos, é possível justificar a viabilidade no desenvolvimento de uma plataforma de desenvolvimento de sistemas de controle, aplicando o estudo de caso em um AGV que possui como sensor uma câmera RGB.

Para que a plataforma obtenha seu legado, o produto final deve ser um documento que apresente toda a construção do software, possibilitando para o novo implementador o entendimento geral da arquitetura.

Capítulo 3

PROPOSTA DO TRABALHO

3.1 Considerações iniciais

O presente capítulo apresenta a proposta do trabalho de acordo com os objetivos apresentados na introdução. Para que a proposta seja delineada, o capítulo foi dividido em quatro partes. A primeira é a apresentação do capítulo. A segunda parte é uma breve explicação sobre a proposta do trabalho, assim demonstrando sua construção, de acordo com os objetivos específicos. A terceira parte descreve o desenvolvimento da proposta, apresentando as características do AD01, junto com a descrição de como esse desenvolvimento será documentado para garantir um legado e facilidade de atualizações futuras. E para finalizar, um breve resumo dos principais pontos descritos compõem a quarta parte.

3.2 Proposta

A proposta do presente trabalho é a criação de uma plataforma de sistemas de controle de um AGV seguidor de linha, orientado por uma câmera. O controlador embarcado será o PID, por seu bom desempenho a uma vasta gama de condições de

operações e sua simplicidade funcional, sendo necessário ajustar 4 parâmetros KP, KI, KDe o período de amostragem na tentativa de encontrar uma resposta desejável no sistema de controle.

O desenvolvimento do software da plataforma se baseia na arquitetura SOA, por ser uma arquitetura modular, simples e válida para sistemas embarcados, reduzindo os resíduos e o uso de códigos duplicados, permitindo, assim, a reutilização de código. Para que o legado de desenvolvimento permaneça, foi desenvolvido um padrão de sete documentos para o desenvolvimento de cada serviço: documento de demanda de produção, documento de requisitos, diagrama UML (Linguagem Modelada Unificada, do inglês *Unified Modeling Language*) de casos de uso, diagrama UML de atividades, diagrama UML de sequência, pseudocódigo e diagrama UML de classes.

A plataforma deve oferecer a modularidade, permitindo uma facilidade em alterar as técnicas de um serviço, facilitando a manutenção e realização de testes da plataforma. O software final deve permitir ao usuário determinar parâmetros para configuração da malha de controle e um sistema de intertravamento de segurança.

A plataforma utilizará um controlador PID para o sistema de controle do AGV. Como resposta da ação do controlador, o AGV possuirá dois motores que variam sua velocidade em relação ao valor da ação do controlador, tornando um AGV com tração diferencial de duas rodas. Para determinar o erro do sistema será utilizado uma câmera RGB, onde por meio do serviço de cálculo de erro será determinado o erro em ângulo e distância do AGV em relação a uma linha guia.

Uma plataforma de desenvolvimento deve apresentar uma facilidade em realizar os testes, para isso será construído um sistema de aquisição de dados, assim então o usuário pode enviar os parâmetros no momento da inicialização do AGV. Um sistema de intertravamento de segurança também será implementado, para que quando necessário o movimento do AGV seja interrompido no momento em que alguma das botoeiras de segurança será acionada.

Todo o sistema deve utilizar a arquitetura SOA para que ele seja modular, assim então quando necessário alterar algum serviço, os outros não necessitam ser alterados.

O produto da plataforma não se resulta somente do AGV físico com um código modular sendo executado. Como resultado da plataforma um Relatório Técnico do Software deve ser construído e apresentado, pois a partir desse relatório é possível que o futuro implementador tenha um suporte adequado.

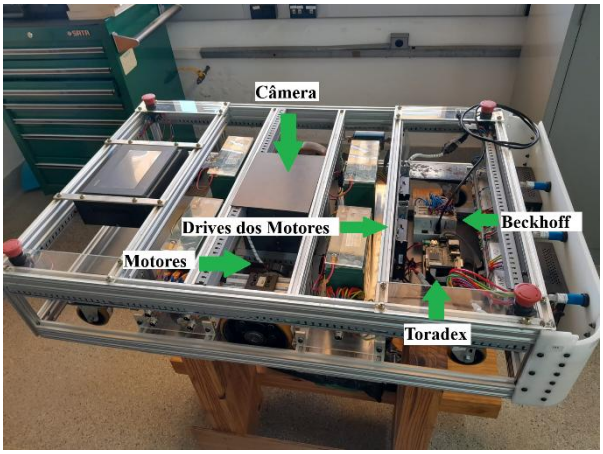
O relatório técnico de software deve conter todos os documentos necessários para o entendimento do software completo, onde para uma nova implementação ou manutenção, o responsável use o relatório como ponto de partida.

O legado da plataforma deve-se resumir no relatório, pois em seu desenvolvimento deve ser considerado que o leitor somente através dele consiga compreender o funcionamento de cada linha de código, e também através do relatório poderá ser identificado possíveis melhorias ou escolhas de novas técnicas.

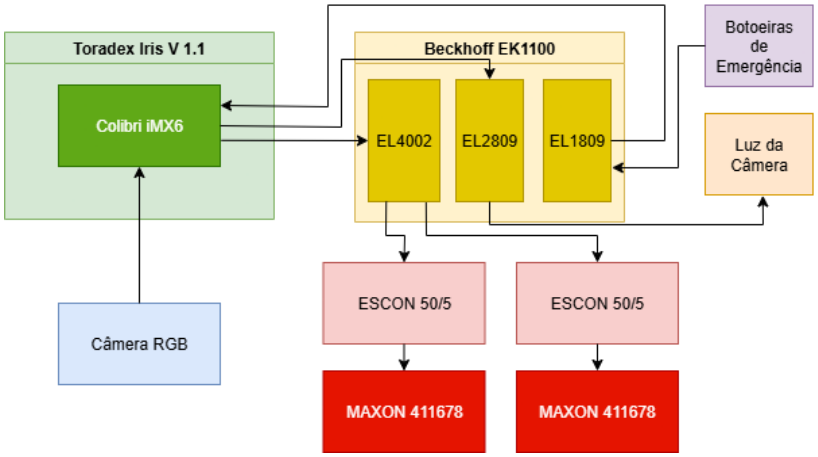
3.3 Desenvolvimento

O AGV utilizado denominado AD01, como apresentado na Figura 3.1. Sendo um robô seguidor de linha orientado por uma câmera RGB posicionada no centro geométrico do veículo, controlado uma Toradex Colibri iMX6 DualLite 512MB na placa Iris V 1.1 que se comunica por protocolo Ethercat com uma remota Beckhoff EK 1100. A remota é responsável pelo controle das entradas e saídas utilizando os módulos EL4002 para saídas analógicas, EL2809 para saídas digitais e EL1809 para entradas digitais. Para o movimento do AGV são utilizados dois motores da Maxon 411678 dispostos no eixo transversal central do veículo, no mesmo eixo da câmera, controlados por drivers ESCON 50/5. A Figura 3.2 apresenta as relações dos componentes por diagrama de blocos.

3.1 AGV AD01



3.2 Diagrama dos Componentes do AD01

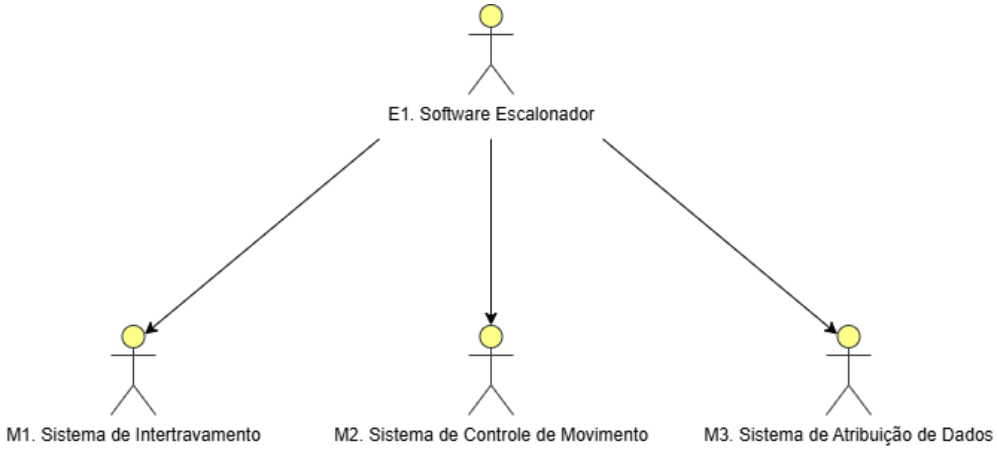


Para o desenvolvimento de software, a organização ocorreu por meio de um escalonador que possui a função de chamar cada majoritário, denominado por sistema, em uma sequência, período de tempo ou momento já determinado. Cada majoritário possuí um objetivo no software geral.

O Sistema de Intertravamento, denominado Majoritário 1 (M1), é responsável pela segurança do AGV, interrompendo a conexão com os motores, assim, parando o movimento do AGV. O M2, Sistema de Controle de Movimento é o responsável pelo movimento do AGV, realizando os cálculos de erros, ação do compensador, realimentação e determinado a velocidade de cada motor do AGV. O M3, Sistema de Atribuição de Dados é responsável pela inicialização de todos os dados necessários para o correto

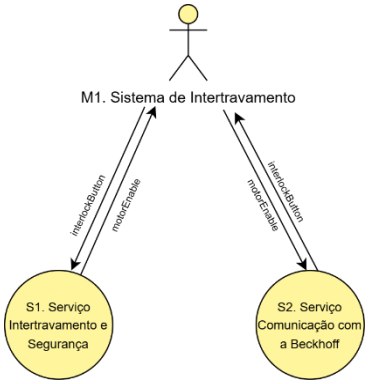
funcionamento do AGV, desde de variáveis de softwares como condições iniciais de hardware, para que o AD01 inicialize com segurança. Com os três majoritários comunicados com o escalonador, obtém-se o topo da arquitetura, apresentado na Figura 3.3.

3.3 Arquitetura Escalonador e Majoritários



O sistema de intertravamento funciona com dois serviços, o S1, Serviço de Intertravamento e Segurança, responsável pela tomada de decisão de desativar ou não os motores. E o S2, Serviço Comunicação com a Beckhoff, responsável por ativar ou desativar os motores pelo drive ESCON através da comunicação ethercat. A arquitetura do sistema de intertravamento e seus serviços estão sendo apresentados na Figura 3.4.

3.4 Arquitetura Sistema de Intertravamento

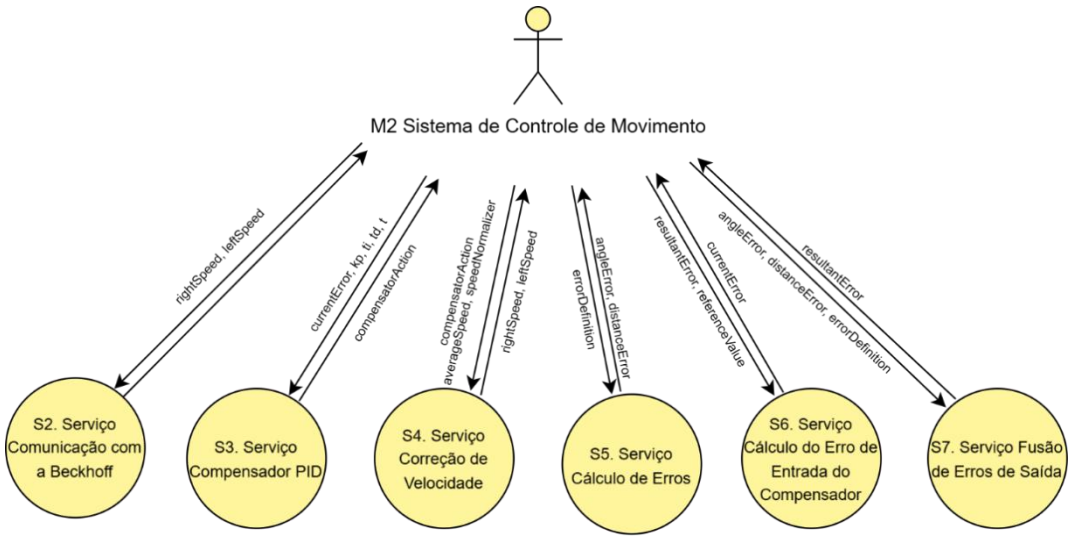


O M2, Sistema de Controle de movimento, é a malha de controle do sistema, possui 6 serviços, S2, Serviço Comunicação com a Beckhoff, sendo o mesmo serviço utilizado no M1, uma vantagem da arquitetura SOA, com isso o mesmo serviço será utilizado, não sendo necessário criá-lo novamente. Para o M2, o S2 será responsável por enviar o valor analógico das velocidades dos motores para o driver ESCON através da comunicação Ethercat.

O Serviço Compensador PID, S3, realiza cálculo da ação do compensador. S4, o Serviço Correção de Velocidade, através dos valores determinados pelo S3 e de informações definidas pelo usuário, determinará os valores das velocidades dos motores, mantendo uma velocidade média. S5, o Serviço Cálculo de Erros, através da câmera, captura a imagem, aplica os filtros na imagem e, em sequência, realiza os cálculos de erro em distância, ângulo ou ambos, dependendo da escolha do usuário.

S6, Serviço Cálculo de Erro de Entrada do Compensador, é responsável por realizar o cálculo da realimentação da malha de controle, subtraindo o erro medido da referência definida pelo usuário. No S7, Serviço de Fusão de Erros, o usuário pode escolher 3 opções de erro a ser considerado pelo controlador, o erro em distância, em ângulo ou a fusão de ambos os erros. Além disso, o S7 é responsável por, a depender da escolha do usuário, retornar o valor do erro na escala padrão, de -100 a 100. A arquitetura do sistema de controle de movimento e seus serviços está representada na Figura 3.5.

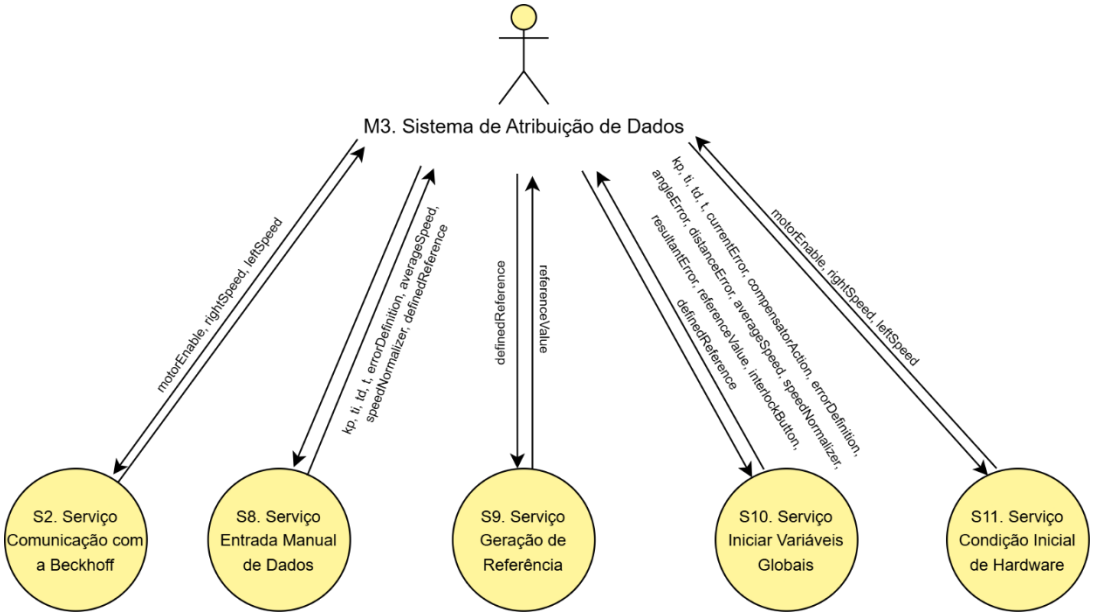
3.5 Arquitetura Sistema de Movimento



O M3, Sistema de Atribuição de Dados, é responsável pelas condições iniciais de hardware e software do AGV, possui 5 serviços. S2, mesmo serviço usado no M1 e M2, sendo responsável por desativar os motores quando o AGV é inicializado, como uma medida de segurança.

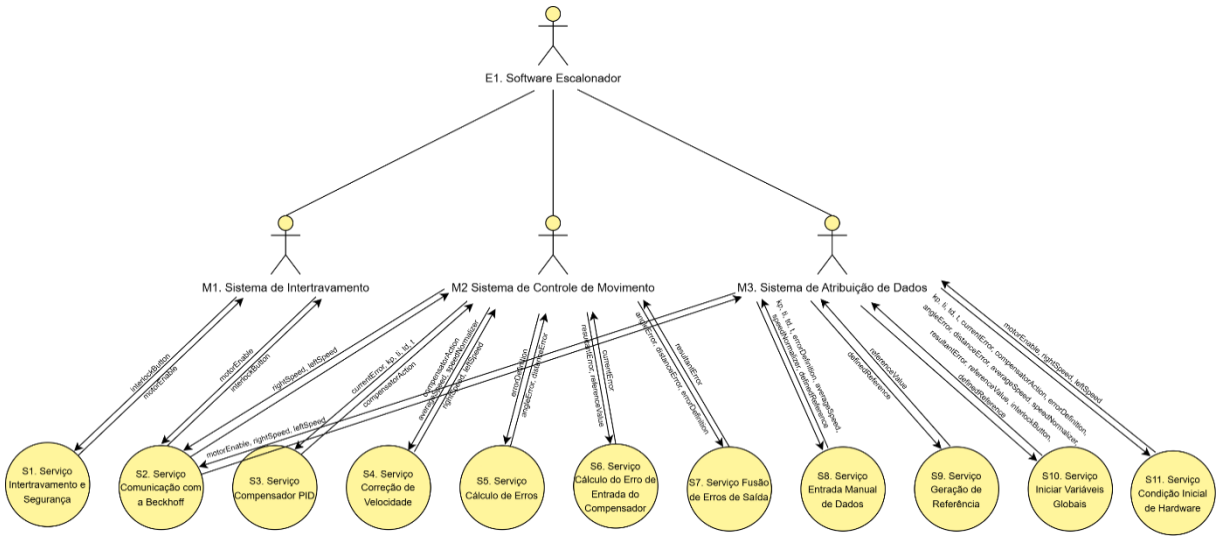
No Serviço Entrada Manual de Dados, S8, o usuário determina alguns parâmetros para configuração da malha de controle. S9, o Serviço Geração de Referência, cria uma entrada em degrau com o valor de referência determinado pelo usuário. S10, o Serviço Iniciar Variáveis Globais, é responsável por inicializar todas as variáveis globais no início da execução do software. S11, o Serviço Condição Inicial de Hardware, permite a inicialização do AD01 com a segurança que os drivers dos motores estarão desativados. A arquitetura do sistema de aquisição de dados e seus serviços estão apresentados na Figura 3.6.

3.6 Arquitetura Sistema de Aquisição de Dados



Com os 3 Sistemas majoritários sendo executados pelo software escalonador, temos o diagrama geral do software. Com os três sistemas e seus serviços apresentados, obtém-se a arquitetura geral do AD01 disposto na Figura 3.7.

3.7 Arquitetura Software AD01



Para que o software seja modificado, é necessária uma clareza de todos documentos pertencentes a modificaçao realizada, sendo importante uma documentação como suporte, para guia de entendimento do sistema. Com isso, um padrão de desenvolvimento de software foi elaborado para a plataforma de desenvolvimento, possibilitando que não somente o engenheiro de software inicial da plataforma realize melhorias no AD01.

O primeiro documento, Documento de Requisitos apresentado no Apêndice A, aborda o módulo em alto nível, apresentando de forma simples, para que a comunicação sobre o serviço não fique limitada somente a quem saiba programar, mas sim para todos que possuem o conhecimento da finalidade do serviço.

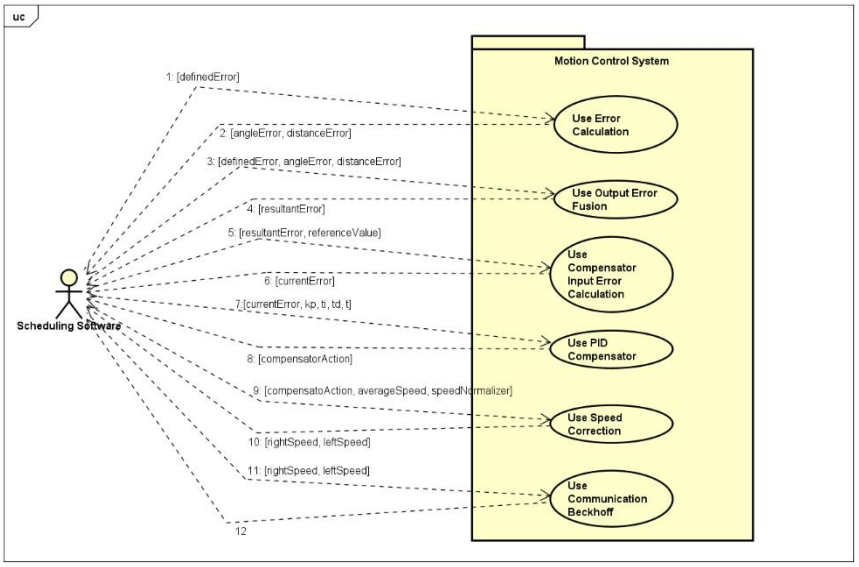
O Documento de Requisitos pode contar ou não com uma introdução, para que o leitor compreenda o problema relacionado ao serviço, e apresenta uma descrição geral com os tópicos; Funções do Produto, Característica do Usuário, Restrições Gerais, Suposições e Dependências. Em seguida devem ser apresentados os Requisitos Específicos, divididos em requisitos funcionais e requisitos não funcionais. A importância dos requisitos para esse arquivo é principalmente para quem realiza os testes do mesmo, pois por esse tópico o desenvolvedor saberá o que deve ser testado.

O segundo documento, Documento de Demanda de Produção apresentado no Apêndice B, é o documento responsável por discutir qual técnica será abordada para o serviço, podendo conter imagens, equações ou uma descrição da sequência de como o

serviço deve ser realizado. É importante que no tópico descrição de métodos seja apresentado o mínimo de conceito computacional e abordar em como o método funciona.

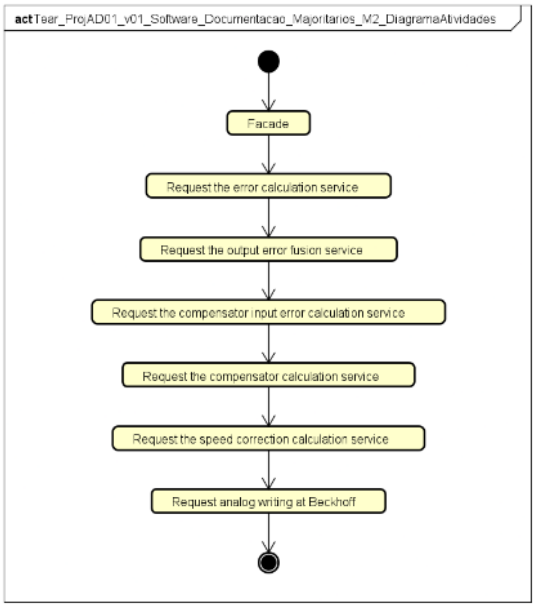
O terceiro documento, diagrama de casos de uso, é um diagrama que demonstra como o serviço irá interagir com o usuário. Para o atual projeto o padrão deste diagrama foi alterado, sendo adicionado não somente os casos de uso, mas também como ele interage, então assim apresentando a ordem que o serviço executa em cada caso de uso, e a variável utilizada em cada momento da execução. Um exemplo de diagrama de casos de uso, está apresentado na Figura 3.8, demonstrando os casos de uso do sistema de controle de movimento.

3.8 Diagrama de Casos de Uso do Sistema de Controle de Movimento



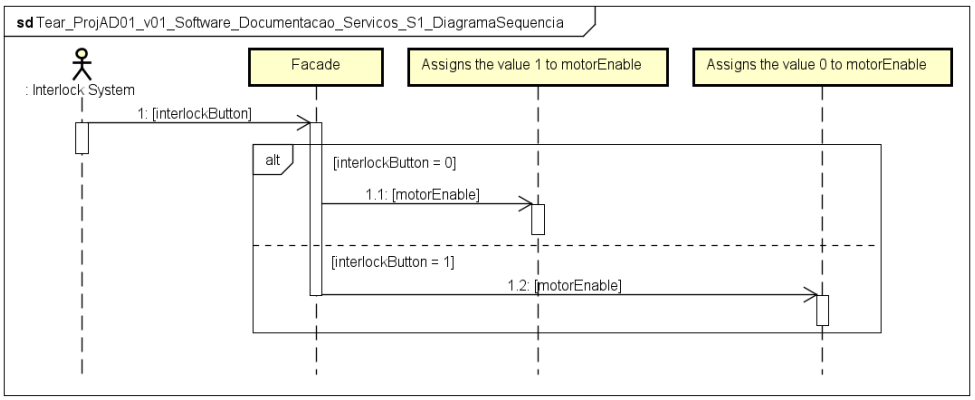
O quarto documento, diagrama de atividades, é um fluxograma que auxilia na apresentação do serviço tanto para as pessoas envolvidas no desenvolvimento quanto para pessoas que não tem o conhecimento de programação, demonstrando como a atividade será realizada. Assim, é um diagrama que deve ser apresentado com clareza e concisão. Um exemplo de diagrama de atividades está apresentado na Figura 3.9, exemplificando as sequências de atividades do sistema de controle de movimento.

3.9 Diagrama de Atividades do Sistema de Controle de Movimento



O quinto documento, diagrama de sequência, para o atual projeto é um diagrama complementar ao diagrama de atividades, composto pela sequência de ações que o algoritmo deverá realizar e quais variáveis serão utilizadas. Esse diagrama se torna um pouco mais próximo das linguagens de programação, podendo apresentar palavras reservadas como *if, while, for, loop*. Um exemplo do diagrama de sequência está apresentado na figura 3.10, exemplificando a sequência de execução do serviço de intertravamento.

3.10 Diagrama de Sequência do Serviço de Intertravamento



O sexto documento, pseudocódigo, é um documento muito próximo do código final do serviço, apresentado em uma linguagem simples e natural. Essa descrição possibilita que o algoritmo seja compreendido por qualquer pessoa, sem a necessidade de conhecimento de linguagens de programação específicas. Um exemplo de pseudocódigo está apresentado na Figura 3.11, exemplificando o pseudocódigo do serviço de intertravamento.

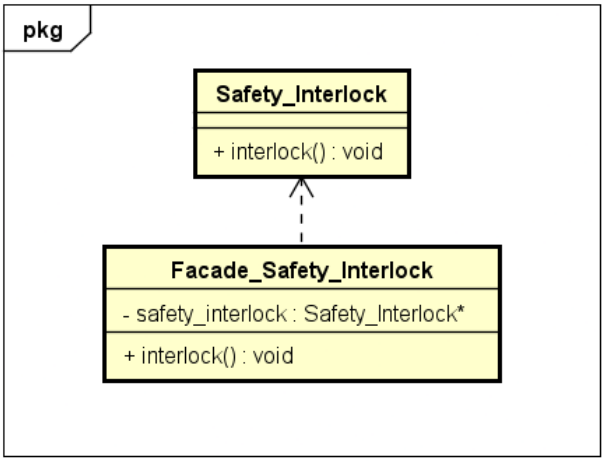
3.11 Pseudocódigo do Serviço de Intertravamento

```
include: globals.h
         Initial_Hardware_Condition.h
define: N/A
variables:  N/A

method interlock()
        if interlockButton = 1
                // Assigns the value 1 to motorEnable
                motorEnable ← 1
        else
                // Assigns the value 0 to motorEnable
                motorEnable ← 0
        end method
```

O sétimo documento, diagrama de classes, é um diagrama destinado à quem possui afinidade com linguagem de programação, sendo muito importante para o desenvolvimento do projeto, pois no diagrama de classes são apresentados os atributos, métodos e operações presente nas classes implementadas. Um exemplo de diagrama de classes está apresentado na Figura 3.12, exemplificando a classe do serviço de intertravamento.

3.12 Diagrama de Classes do Serviço de Intertravamento

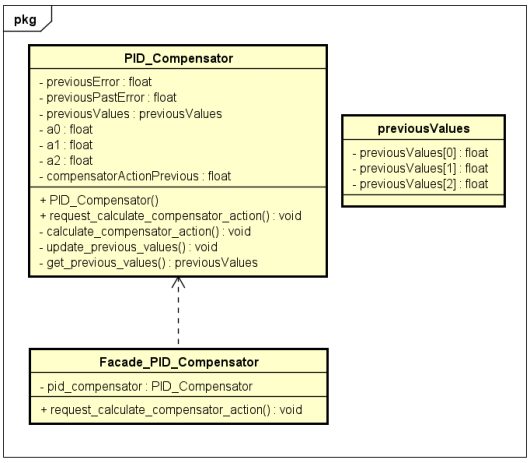


Para facilitar a comunicação, cada módulo usa uma "fachada", uma interface simples que mostra apenas o essencial e esconde os detalhes internos. Isso torna o código mais claro e desacoplado, facilitando a manutenção e os testes, pois mudanças internas não afetam o uso externo. Além disso, aumenta a segurança, já que o acesso ao módulo é restrito apenas a essa interface.

Para entender melhor a finalidade da fachada é possível observar o diagrama de classes do serviço S3, o compensador PID. O serviço possui 5 métodos, sendo o primeiro o método construtor, para inicializar em zero os valores de erro, depois o método que solicita o cálculo do compensador, o terceiro método é o método que realiza o cálculo do compensador, mais dois métodos auxiliares para utilizar e armazenar os valores de erros anteriores. Além da classe possuir 7 atributos para realizar os cálculos.

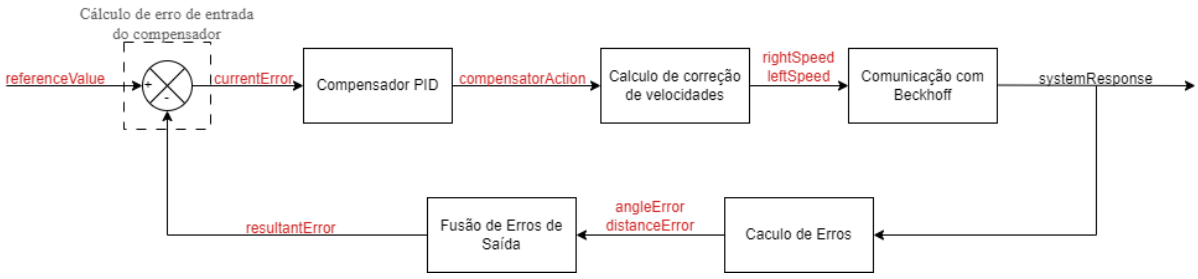
Para calcular a ação do compensador, basta enviar o valor do erro e receber o resultado. O usuário não precisa conhecer os detalhes internos da classe, interagindo apenas com o essencial. A fachada funcionou como uma interface simples de comunicação com a classe, apresentando para o usuário o único método que realmente importa para solicitar o cálculo. Para entender a fachada é possível ver sua aplicação na Figura 3.13, onde a fachada possui somente o método que o usuário irá utilizar.

3.13 Diagrama de Classes do Serviço Compensador PID



A malha de controle do sistema é executada pelo sistema de controle de movimento, o M2, quando montado o diagrama de casos de uso do M2 é possível descrever o diagrama de blocos da malha de controle como presente na Figura 3.14. Cada bloco retangular, incluindo o quadrado pontilhado, corresponde a um serviço utilizado pelo M2, as variáveis que são enviadas para os serviços são variáveis públicas do sistema, sendo representadas pela cor vermelha na imagem.

3.14 Diagrama de Blocos da Malha de Controle



O sistema operacional instalado na Toradex é o Toradex Embedded Linux Reference Multimedia Image, versão 6.8.1, justificado por apresentar um suporte a placa colibri, uma documentação muito detalhada no site da Toradex, e proporcionar drivers já instalado no SO para testar e configurar a câmera. Facilitando o desenvolvimento, testes e definição de parâmetros para o serviço de cálculo de erro.

O driver V4L2 (Vídeo Linux) já instalado no SO possui suporte para webcam USB, assim possibilitando uma escolha com segurança da taxa de FPS, resolução da imagem e formato da imagem capturada.

Para comunicação o SO com um computador externo possui duas formas de acesso ao terminal, ambas documentadas no suporte pelo site da Toradex. A primeira forma pode ser por console serial, sendo a recomendada para o carregador de inicialização e o kernel do Linux. A segunda opção de comunicação é por SSH (Secure Shell), podendo ser habilitada no momento da instalação do SO.

Para uma maior organização, a compilação do sistema implementado será realizada pelo CMakeLists, cada módulo contará com um arquivo de texto com as configurações e particularidades de cada serviço na sua própria pasta. Na pasta raiz do software, um arquivo de texto principal será responsável por unir todos os módulos, adicionar as bibliotecas necessárias e realizar a compilação cruzada.

Na Figura 3.15, é apresentado um exemplo de CMakeLists do serviço S5, cálculo de erro, além de conter os elementos básicos, o exemplo contém o uso da biblioteca OpenCV para capturar e filtrar a imagem e realizar os cálculos de erro e distância.

Primeiramente, deve-se em formato de comentário apresentar o diretório em que o arquivo está em relação a pasta raiz do projeto, pois todos arquivos CMakeLists terão o mesmo nome.

Como o serviço utiliza a biblioteca OpenCV, e será realizado a compilação cruzada, será necessário realizar a compilação da biblioteca de forma estática para a arquitetura ARM. Assim não sendo necessário a realização da instalação da biblioteca diretamente no SO. Esse padrão de utilizar biblioteca estática será utilizado em todas as bibliotecas e módulos do projeto, facilitando a realização de testes e atualizações.

Com a biblioteca instalada corretamente para a arquitetura desejada, deve-se adicionar o local de instalação da biblioteca e realizar as chamadas necessárias para seu uso. Cada módulo do projeto será considerado pelo CMakeLists principal como uma biblioteca, por isso determinamos um nome para o módulo, com a inicial E para escalonador, M para majoritário e S para serviço, acompanhado de um número. Na Figura

3.15, o serviço de cálculo de erro será denominado S5 e será considerado como uma biblioteca pública. Esse padrão de CMakeLists seguirá para todos módulos do projeto.

3.15 CMakeLists do Serviço de Cálculo de Erro

```
# Servicos/S5/CMakeLists.txt

set(OpenCV_DIR "/home/luan/Downloads/opencv-4.8.0/build")
find_package(OpenCV REQUIRED)

set(SOURCES Error_Calculation.cpp
            Facade_Error_Calculation.cpp)
add_library(S5 ${SOURCES})

target_include_directories(S5 PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
target_include_directories(S5 PUBLIC ${CMAKE_SOURCE_DIR}/Servicos/S10)
target_include_directories(S5 PUBLIC ${OpenCV_INCLUDE_DIRS})

target_link_libraries(S5 PRIVATE S10)
target_link_libraries(S5 PRIVATE ${OpenCV_LIBS})
```

O arquivo textual compilação principal tem a função somente de importar os módulos e bibliotecas do projeto, importando o caminho de cada módulo e biblioteca.

Para a realização da compilação cruzada foi utilizado o compilador presente no instalador de pacotes do Ubuntu. Com isso o compilador utilizado foi o gnueabihf, para C e C++. Com o uso do Cmake não é necessário mudanças nos CMakeLists para realizar a compilação cruzada, somente o uso de um arquivo adicional denominado toolchain.cmake.

O toolchain.cmake apresentado na Figura 3.16 é responsável por apresentar o diretório do compilador ARM, os diretórios das bibliotecas que podem ser usadas e configurações simples de compilação, como determinar que o arquivo final seja completamente estático, ou seja, não necessitando de alguma biblioteca adicional instalada no SO.

As bibliotecas foram extraídas do próprio SO da toradex, sendo copiadas as pastas usr e lib do sistema e adicionada no computador onde ocorrerá a compilação cruzada em uma pasta denominada sysroot.

3.16 Toolchain Para Compilação Cruzada

```
# toolchain.cmake

# Defina o nome do sistema e do processador
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR arm)

# Defina o compilador C e C++
set(CMAKE_C_COMPILER /usr/bin/arm-linux-gnueabihf-gcc)
set(CMAKE_CXX_COMPILER /usr/bin/arm-linux-gnueabihf-g++)

set(CMAKE_C_FLAGS "-static")
set(CMAKE_CXX_FLAGS "-static")

# Defina o caminho do sysroot (caminho absoluto, no seu caso
~/sysroot)
set(CMAKE_FIND_ROOT_PATH /home/luan/sysroot)

# Apenas para garantir que o CMake encontre as bibliotecas dentro do
sysroot
set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

A escolha do SO foi feita de acordo com o que oferece um melhor suporte da Toradex, facilidade do uso e armazenamento do instalador, e suporte às funcionalidades necessárias para todo o sistema que o AD01 necessita. Com isso o uso de um SO feito pela própria fornecedora da placa se torna o caminho mais viável a ser seguido.

Sendo possível baixar a imagem pelo site da Toradex, o sistema operacional Torizon OS 6.8.1+build.25 habilitando a versão *Reference Multimedia Image*, uma imagem que já possui instalado o suporte para o driver GStreamer e V4L2, sendo muito importante para configurar a webcam utilizada como sensor de erro do AD01. Contemplando as necessidades para utilizar o SO no AGV, a comunicação SSH é habilitada no momento da instalação, facilitando a comunicação entre o usuário e a Toradex.

Como uma plataforma de desenvolvimento é importante também realizar testes unitários e de integração dos componentes de hardwares da plataforma, pois não

somente os serviços de códigos podem ser alterados, mas os componentes também. Um exemplo, a câmera pode ser alterada por outra com uma resolução diferente, ou que apresente uma taxa de FPS maior.

Por isso é necessário implementar também um plano de testes. O plano de testes é importante não somente para a implementação ou alteração de outros módulos, mas é um auxiliar muito importante para a manutenção eletromecânica da plataforma, visto que realizar os testes unitários e depois o de integração de todos os componentes da plataforma, é possível identificar a falha mecânica ou elétrica do AD01.

Para descrição dos testes, será apresentado um exemplo de sequência de testes realizados para a comunicação Ethercat e seus módulos de leitura e escrita digital, e escrita analógica. Os testes da comunicação Ethercat não são os únicos necessários, como já apresentado o exemplo do teste de câmera, mas reflete a importância de implementar um protocolo de testes para o desenvolvimento de uma plataforma de sistemas de controle.

Destacando a importância dos testes, foram realizados os unitários de acionamento da lâmpada, sendo um teste simples de escrita digital, que por sua vez, além de testar a parte eletrônica e escrita digital, é um teste simples para a comunicação Ethercat.

O segundo teste foi o teste unitário das botoeiras de emergência. Com esse teste, é possível avaliar a leitura digital. A importância do teste de emergência vai além da leitura digital, visto que o uso das botoeiras é uma medida de segurança presente no AD01.

O terceiro teste é o de acionamento dos motores. Para que o acionamento dos motores ocorra, é necessário além de uma escrita digital para ativar os drivers, uma escrita analógica, que determina a velocidade de cada motor. O quarto teste é uma sequência do terceiro, pois relata o teste de variação de velocidades dos motores, indo além da escrita analógica, mas considerando em enviar os sinais analógicos em proporções corretas.

Como apresentado há uma grande relevância do teste das botoeiras de emergência, pois elas serão utilizadas como medida de segurança. Um exemplo de testes

de integração está em desativar os drivers dos motores após o acionamento das botoeiras de emergência.

3.4 Considerações finais

A utilização da documentação auxilia na manutenção e legado do projeto. Com todos os documentos feitos de forma correta, consistente e acessível, a avaliação do módulo pode ser realizada por diversas pessoas, não sendo limitada apenas a quem entende de programação, assim facilitando a análise e discussão de novas técnicas para o módulo.

O SO escolhido oferece o suporte necessário para que o desenvolvimento do projeto ocorra de forma fluida, com um site apresentando uma documentação completa e uma equipe que auxilia a solucionar problemas pelo e-mail de suporte.

A compilação apresenta os requisitos de modularidade, fácil manutenção e facilidade na realização dos testes, cumprindo o objetivo da arquitetura escolhida para o projeto.

A realização de um protocolo de testes é de extrema importância, pois através dos testes é possível analisar o uso de novos módulos e realizar a manutenção eletromecânica do AD01, sendo um grande contribuinte para a construção de uma plataforma de desenvolvimento.

O centro do documento é garantir que todos que leiam o relatório de software consigam compreender o funcionamento do AD01, assim possibilitando uma melhor discussão e avaliação para novas melhorias, sendo esse relatório o produto responsável para manter o legado da plataforma de desenvolvimento.

Capítulo 4

VALIDAÇÃO

4.1 Considerações iniciais

O presente capítulo objetiva descrever como o trabalho será validado, realizando ensaios e índices suficientes para que os objetivos da dissertação sejam atendidos e comprovados. O capítulo está dividido em cinco partes, a primeira sendo a atual a descrição do conteúdo. A segunda parte aborda como a proposta será validada. Para isso foram escolhidos quatro índices que podem descrever de forma simples e inicial o comportamento do AD01. A terceira parte descreve como os testes foram realizados, abordando as configurações de parâmetros indicadas como padrão para os ensaios. A quarta parte descreve os testes realizados e os resultados obtidos em forma de gráficos e tabela para comparação. Para finalizar o capítulo uma conclusão dos principais pontos encontrados.

4.2 Forma de validação

Para validar o controlador empregado, avaliar a plataforma de desenvolvimento de sistemas de controle e entender suas características, foi realizada a sintonia de um controlador PID, assim entendendo a necessidade de cada ganho do controlador de

acordo com o comportamento do AGV. A sintonia aplicada na plataforma foi a sintonia manual.

Como método quantitativo de comparação entre as sintonias e melhor entendimento do comportamento do AGV foram escolhidos cinco índices na avaliação das plantas. O primeiro índice utilizado é o IAE (Integral do Erro Absoluto, do inglês *Integral of Absolute Error*). Apresentado na Equação 3.

$$\text{IAE} = \int_0^T |e|\, dt \quad (3)$$

O termo e corresponde ao erro, 0 é limite inicial de tempo e T o limite final tempo.

O segundo índice foi o ITAE (Integral do Tempo Multiplicado pelo Erro Absoluto, do inglês *Integral of Time Multiplied by Absolute Error*), utilizado para penalizar o erro ao longo do tempo. Apresentado na Equação 4.

$$\text{ITAE} = \int_0^T t \cdot |e|\, dt \quad (4)$$

O termo t corresponde ao valor do tempo no momento da interação.

O terceiro índice utilizado, EMA (Erro Máximo Absoluto), analisou o maior erro presente no controlador. Uma métrica simples, mas apresentando o pior caso de erro presente no controlador, tornando uma métrica auxiliar.

O quarto e último índice utilizado é a variância do erro. Uma métrica utilizada para avaliar o quanto os erros variam em relação à média dos erros, ou seja, utilizado para visualizar a variabilidade e dispersão do erro. Apresentado na Equação 5.

$$\text{s}^2 = \frac{\sum_{i=1}^{n}(e_i - \bar{e})^2}{n-1} \quad (5)$$

O termo n corresponde ao número de amostras, ei é o erro atual, e o e é a média de todos os erros.

Os quatro índices utilizados apresentam o intuito de analisar o erro da trajetória do AD01, portanto quanto mais próximo de zero melhor, assim sendo realizado as sintonias com o intuito de reduzir o máximo possível os quatro índices.

O erro a ser calculado, para os testes, é a fusão do erro em distância e ângulo. Para que a fusão ocorra a seguinte sequência deve ser aplicada. Primeiro calcula o erro em ângulo e normaliza de -100 a 100 conforme a Equação 6. Onde representa o ângulo do AGV em relação ao centro da linha guia.

$$e_{\hat{a}ngulo\ normalizado} = 100 \cdot sen(\theta) \quad (6)$$

Em sequência se calcula o erro em distância normalizado. Para que a normalização ocorra utiliza-se a Equação 7. O termo d Representa a distância em pixel entre o centro da imagem e o centro do AGV.

$$e_{dist\hat{a}ncia\ normalizado} = 100 \cdot \frac{d}{n\acute{u}mero\ de\ colunas\ /\ 2} \quad (7)$$

Com os erros em distância e ângulo normalizados, é realizado a média entre eles como a Equação 8, assim obtendo a fusão dos erros, em que o valor esteja normalizado de -100 a 100.

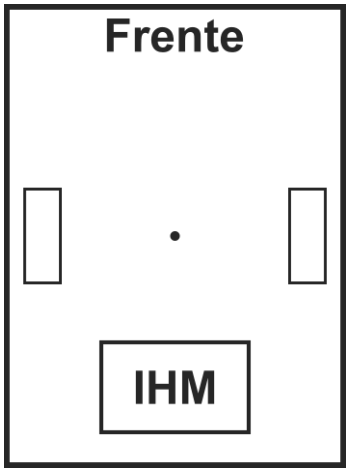
$$e_{normalizado} = 100 \cdot \frac{e_{\hat{a}ngulo\ normalizado} + e_{dist\hat{a}ncia\ normalizado}}{2} \quad (8)$$

4.3 Testes

Os testes foram realizados no TEAR, então as limitações de tamanhos de retas e curvas se dão pelo espaço de realização dos ensaios. As linhas guias foram montadas no chão do laboratório com fita isolante. Para a alimentação do AGV foi utilizado uma fonte externa ligada ao AGV com um cabo de tamanho suficiente para realizar todos os testes possíveis no ambiente.

O AD01 foi projetado para movimentar em ambos os sentidos, por isso, foi determinado um padrão para definir a frente e a parte de trás do AGV. Como escolha de padronização dos testes foi determinado que a IHM está mais próxima da parte de trás do AD01, e a parte da frente está mais próxima da Toradex, módulos da Beckhoff e dos Drivers dos motores, como apresentado na Figura 4.1.

4.1 Orientação do AD01



As especificações da câmera são de 30 FPS em uma resolução de 160 x 120 pixels, sendo essa a resolução mínima da câmera. As escolhas dos parâmetros atuais se justificam pela necessidade de uma baixo processamento de dados no momento da execução da malha de controle.

Para o período da malha de controle foi determinado 150 ms, justificados principalmente pelo tempo gasto para a realização do serviço de cálculo de erro. A velocidade média do AD01 foi determinada em 10% da velocidade máxima dos motores. A variação de velocidade das rodas foi de 50%.

Quando utilizado curvas, o raio da curva foi de 3 metros. Em razão ser muito superior ao raio mínimo que o AD01 pode fazer com as configurações apresentadas anteriormente, não alterando a velocidade média. A Equação 9 define o raio mínimo.

$$R = \frac{l}{2} \cdot \frac{V_d + V_e}{V_d - V_e} = \frac{0{,}75}{2} \cdot \frac{15 + 5}{15 - 5} = \frac{0{,}75}{2} \cdot \frac{20}{10} = 0{,}75m \quad (9)$$

O termo, l é distância entre as rodas, Vd foi considerado o valor máximo possível em porcentagem de velocidade em uma das rodas com as configurações apresentadas, Ve foi considerado o valor mínimo possível em porcentagem de velocidade em uma das

rodas. Assim apresentando a situação de curva com o menor raio possível para a atual situação.

A porcentagem de velocidade apresentada é determinada de acordo com a velocidade máxima das rodas, sendo definida primeiramente calculando-se a velocidade do motor após a redução, para o cálculo em rpm utiliza-se a Equação 10.

$$V_r = \frac{V_n}{redução} = \frac{3740}{15} = \frac{748}{3} = 249,\bar{3}\ rpm \quad (10)$$

A velocidade nominal do motor é de 3740 rpm com um redutor de 15:1, assim como resultado temos uma velocidade após a redução de 249,3 rpm. Para transformar essa velocidade em rotações por segundos utilizando a Equação 11.

$$V_{r\,rps} = \frac{V_r}{60} = \frac{249,\bar{3}}{60} = 4,1\bar{5}\ rps \quad (11)$$

Agora se calcula primeiramente a circunferência da roda pela Equação 13.

$$c = \pi \cdot d = \pi \cdot 0,15 \approx 0,4712\ m \quad (13)$$

O valor 0,15m representa o diâmetro da roda do AGV. Para encontrar a velocidade máxima do AD01, basta multiplicar a velocidade em rps pela circunferência da roda, assim obtendo a velocidade em metros por segundo com a Equação 14.

$$V_{m/s} = V_{r\,rps} \cdot c = 4,1\bar{5} \cdot 0,4712 \approx 1,95\ m/s \quad (14)$$

Com os cálculos anteriores é possível concluir que a velocidade máxima que o AD01 poderá atingir é de aproximadamente 1,95 m/s.

A pista em linha reta possui um comprimento de 4 metros, onde o AGV interrompia o movimento quando acionado as botoeiras de emergência. Para que as botoeiras fossem acionadas dois critérios foram utilizados, quando o AGV chegar ao fim da linha, ou quando o AGV não conseguir capturar a imagem da linha pela câmera.

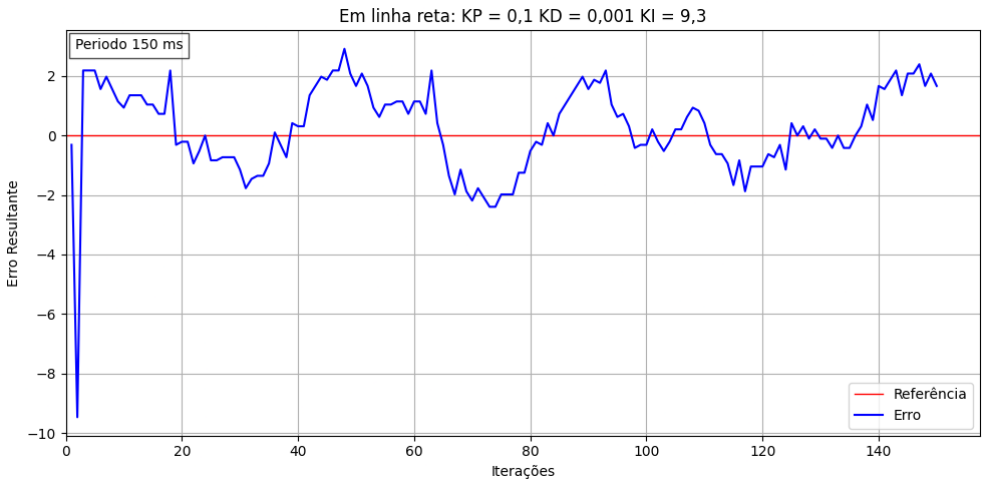
Na pista com curva, o AGV inicia na entrada da curva e realiza uma curva de 90°, com um raio de três metros. Os critérios de parada são os mesmos de quando o AGV está em linha reta.

Para que os ensaios não apresentem alterações devido a ações externas, sempre o AGV inicia no mesmo ponto de partida, com todas as rodas alinhadas facilitando o movimento para frente em linha reta. Todos os ensaios foram realizados três vezes para avaliar se o resultado apresentado não foi um comportamento atípico

4.4 Resultados atingidos

O primeiro teste foi realizado com a linha guia reta e o uso do erro em distância. Como resultado do comportamento do AGV obtivemos o seguinte comportamento do AD01.

4.2 Resposta do Sistema em Linha Reta: Sintonia PID KP=0,1 KI=9,3 KD=0,001

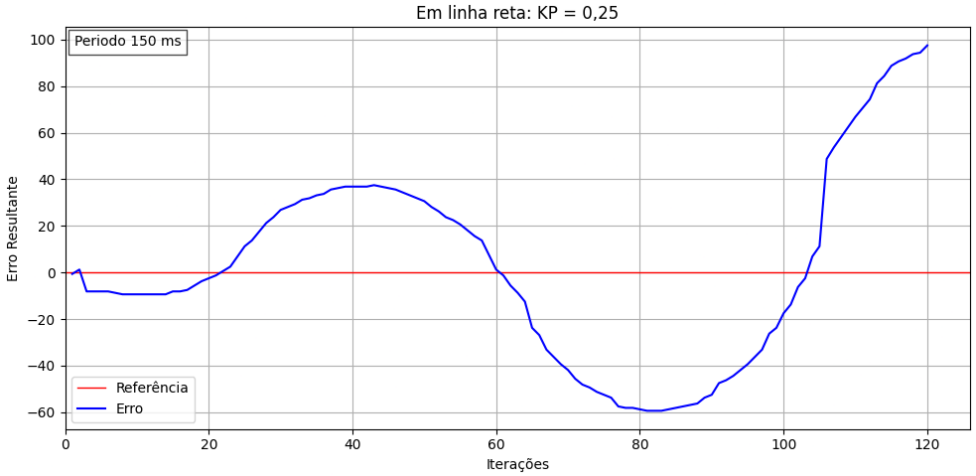


Para chegar no resultado apresentado, um comportamento foi observado. O valor do ganho proporcional que estabiliza o sistema é muito baixo. Ao realizar os primeiros testes para a sintonia manual, foram encontrados os seguintes resultados com Kp. Primeiramente Kp = 0,25, depois Kp = 0,1 e em sequência Kp = 0,001.

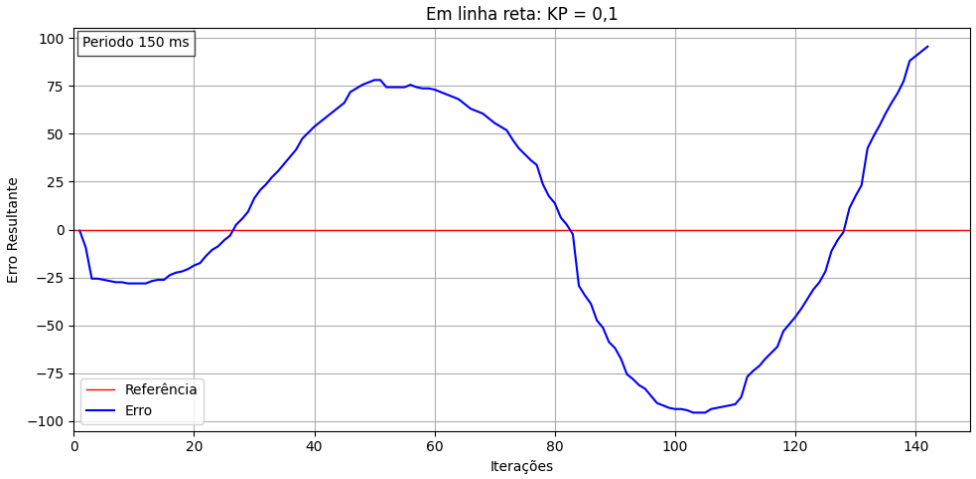
Apresentados nas Figuras 4.3 4.4 e 4.5 foi observado que, mesmo reduzindo muito o ganho proporcional, não foi possível encontrar uma tendência de estabilidade, visto que em todas as situações apresentadas o uso somente do ganho proporcional não foi o suficiente para que o AD01 encontre a estabilidade em linha reta. Para que seja

encontrada uma sintonia para a planta apresentada foi necessário o uso do ganho Kd, não sendo possível encontrá-la somente com um controlador proporcional.

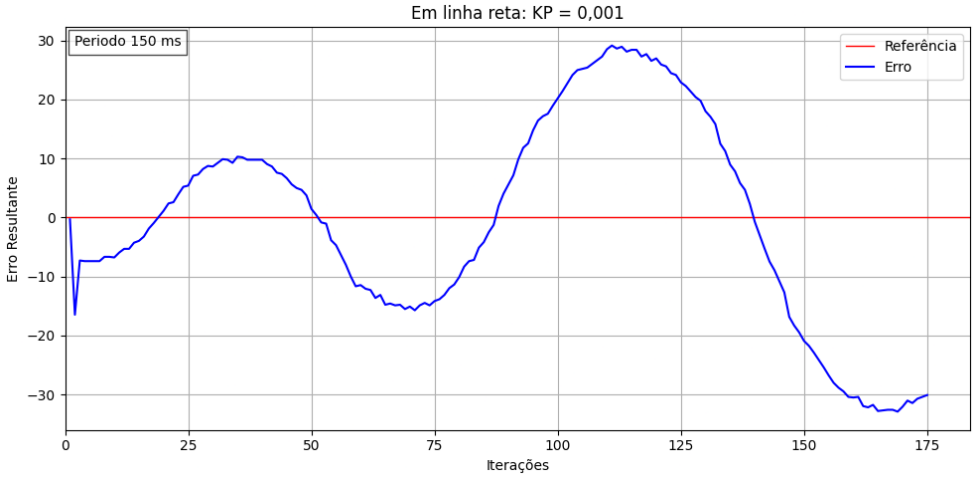
4.3 Resposta do Sistema em Linha Reta: Sintonia Proporcional KP=0,25



4.4 Resposta do Sistema em Linha Reta: Sintonia Proporcional KP=0,1



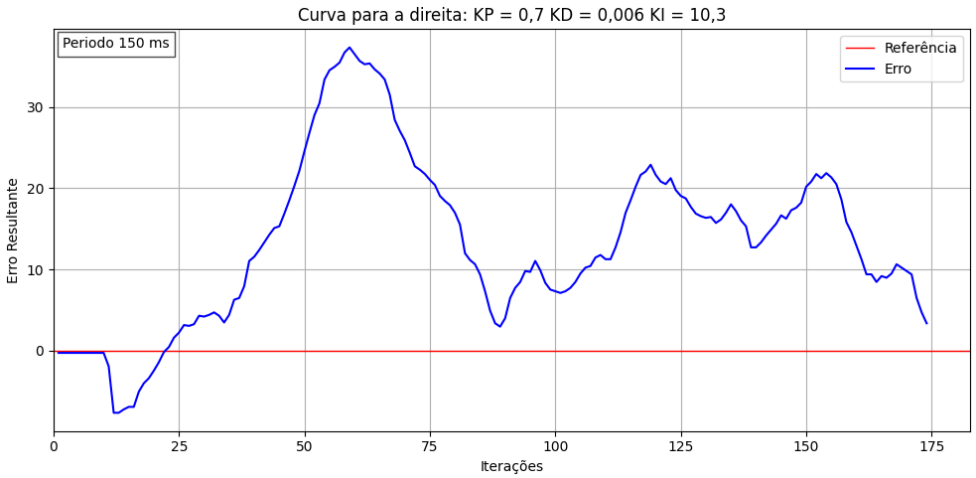
4.5 Resposta do Sistema em Linha Reta: Sintonia Proporcional KP=0,001



Vale ressaltar a diferença de escala apresentada nas Figuras 4.3, 4.4 e 4.5, onde a variação da sintonia da Figura 4.2 não ultrapassa o erro absoluto de 3% após o erro máximo do início do movimento. Já as sintonias das Figura 4.3, 4.4 e 4,5 utilizando somente o controlador proporcional apresentam uma tendência de crescimento e os erros já nas mesmas iterações se aproximam de 100%.

O segundo teste realizado foi com o AD01 em curva para a direita. Como esperado para que o AD01 pudesse realizar a curva, seria necessário o aumento do Kp, pois com o ganho proporcional muito baixo, o AGV poderia não ter uma resposta desejada nas curvas, chegando em uma sintonia estável com Kp = 0,7 Kd = 0,006 e Ki = 10,3.

4.6 Resposta do Sistema em Curva Para Direita: Sintonia PID KP=0,7 KI=10,3 KD=0,006



A sintonia do AD01 com a curva para a direita apresenta uma tendência em ficar com o erro positivo, o que demonstra no AGV uma tendência de realizar a curva pelo lado esquerdo da curva. Vale salientar que a curva apresenta um raio de 3m e tem o comprimento de um quarto da circunferência.

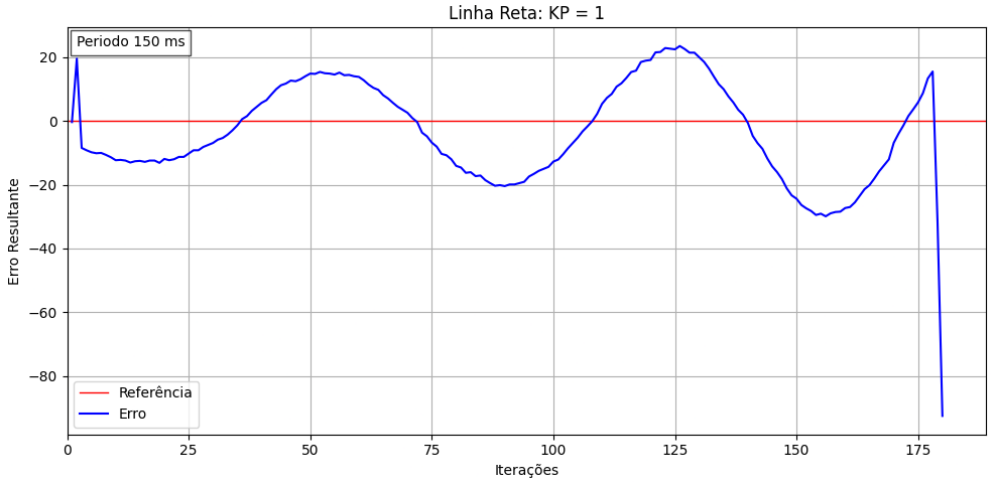
Para finalizar a sintonia em curva do AD01 foi realizado o teste com a curva para a esquerda. O exemplo de sintonia estável permaneceu semelhante a sintonia para a planta do AD01 realizando a curva para direita, mas uma leve alteração no ganho derivativo. A sintonia é de Kp = 0,7 Kd = 0,008 e Ki = 10,3.

4.7 Resposta do Sistema em Curva Para Esquerda: Sintonia PID KP=0,7 KI=10,3 KD=0,008

Apresentado os testes com estabilidade é importante mencionar ensaios que resultaram em uma instabilidade do sistema, pois esses resultados também são de extrema importância para entender o funcionamento do AD01.

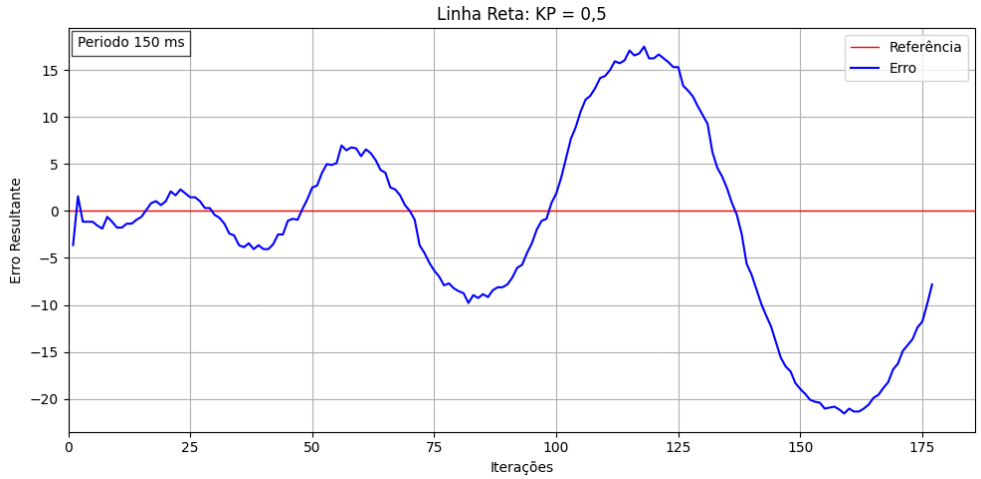
Para início dos testes do controlador, um ponto de partida é realizar o ensaio com um controlador proporcional de ganho unitário. Na realização desse teste foi possível, conforme apresentado na Figura 4.8 observar uma tendência de crescimento do erro resultante. Através do primeiro ensaio, é possível concluir que o ganho proporcional deve ser reduzido, no intuito de encontrar uma estabilidade.

4.8 Resposta do Sistema em Linha Reta: Sintonia Proporcional KP=1



Apresentado o resultado da sintonia do controlador proporcional de ganho unitário, foi realizado o ensaio com o controlador proporcional com o ganho de 0,5. Como apresentado pela Figura 4.9, ainda assim é possível ver uma tendência de crescimento do erro, mas em tempo maior que do controlador proporcional de ganho unitário. Essa análise foi o ponto de partida para encontrar a sintonia presente na Figura 4.2, a partir dela é possível concluir que o ganho proporcional deve ser diminuído.

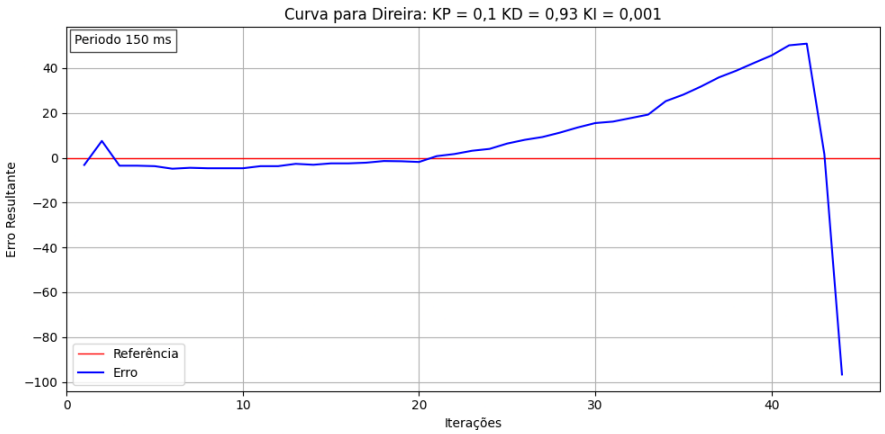
4.9 Resposta do Sistema em Linha Reta: Sintonia Proporcional KP=0,5



Visto que a tendência é reduzir o ganho proporcional em linha reta, o ponto de partida para curvas não é o mesmo, pois como avaliado na Figura 4.10, pois quando o ganho proporcional é muito baixo, o controlador não tem proporção o suficiente para agir,

assim resultando na saída da linha no momento da curva. Observando a Figura 4.10 é possível ver que o sistema se torna instável em curvas com o ganho proporcional muito baixo, o que demanda em aumentar o Kp a ponto de que ele se torne estável em curvas.

4.10 Resposta do Sistema em Curva Para Direita: Sintonia Proporcional KP=0,1

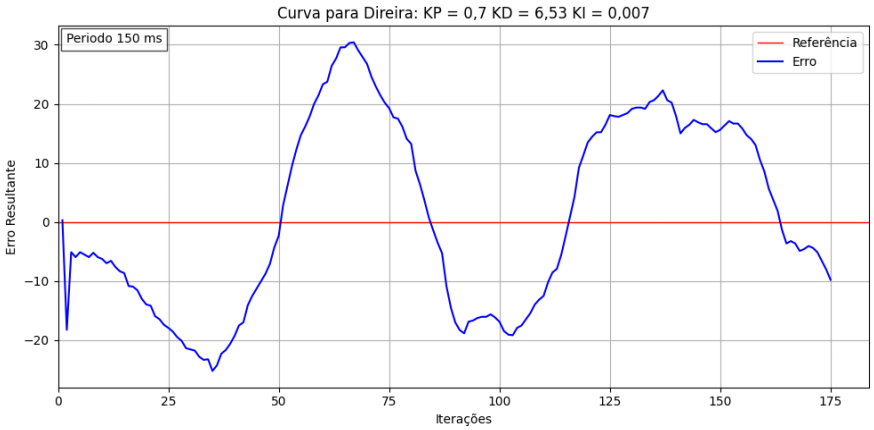


O ganho proporcional foi aumentado até 0,5 ainda assim se mantendo instável, como apresentado na Figura 4.11. Quando adicionado o ganho proporcional em 0,7 o sistema se tornou estável e resultou nas Figuras 4.6 e 4.7 já apresentadas nos resultados de sistemas estáveis.

4.11 Resposta do Sistema em Curva Para Direita: Sintonia Proporcional KP=0,5 KI=0,005 KD=4,6

Com o ganho proporcional em 0,7 apresentado na Figura 4.12, foi encontrado a estabilidade em curva, sendo necessário apenas alterar os ganhos integrais e derivativos para encontrar uma melhor resposta no sistema, como apresentado nas Figuras 4.6 e 4.7.

4.12 Resposta do Sistema em Curva Para Direita: Sintonia Proporcional KP=0,5 KI=0,007 KD=6,53



Uma sequência de testes base para o AD01 foi realiada. Vale salientar que outros testes ainda são importantes. Dentre os testes importantes deve-se destacar teste com o AGV em curva para a direita seguido de uma reta, AGV em curva para a esquerda seguido de uma reta, AGV em reta seguido de uma curva para a direita, AGV em reta seguido de uma curva para a esquerda, e todos esses testes apresentados e citados com o AD01 em movimento reverso. Além disso, todos os testes citados devem ser executados com diferentes cargas.

A realização de todos esses testes são fundamentais para entender o comportamento do AD01. Como o objetivo é validar uma proposta de plataforma de desenvolvimento de sistemas de controle aplicada em um AGV, e considerando as limitações de tempo e espaço, os testes apresentados foram suficientes para validar a plataforma. Mas ainda é necessário realizar uma sintonia com o intuito de encontrar um melhor comportamento para o AD01 em todas as situações apresentadas.

Como comparação dos comportamentos das diferentes situações apresentadas, o uso de uma tabela foi gerado, sendo apresentado na Tabela 4.1. Os resultados da Tabela 4.1 são somente dos ensaios que apresentaram a estabilidade e considerados o melhor comportamento do AD01.

Tabela 4.1 Resultados dos Ensaios

Ensaio	IAE	ITAE	EMA	s^2
Linha Reta	171	11.659	9	2,2
Curva Para Direita	2.134	181.975	37,4	119,6
Curva Para Esquerda	1.813	133.708	23,8	85

A não simetria perfeita do AGV pode ser observada na a tabela acima e as diferentes sintonias necessárias para um melhor resultado para curvas com o mesmo raio, mas com sentidos diferentes. Quando considerado o movimento no sentido frontal do veículo em curvas, ele apresenta uma leve tendência a ir para a esquerda. Tal comportamento não é identificado quando observado o AD01 se movimentando em linha reta.

4.5 Considerações finais

Inicialmente deve-se considerar a justificativa do primeiro ensaio ser feito somente com a distância, ao invés do padrão de testes, que é o uso da fusão de distância e ângulo. O primeiro ensaio foi realizado com a distância primeiramente para que possamos realizar uma sintonia mais simples e já confirmada que possível, pois a fusão de ângulo e distância não foi testada em outros AGVs.

Mesmo com os presentes testes validando a plataforma de desenvolvimento de sistemas de controle, outros ensaios em diferentes plantas são necessários para determinar uma melhor sintonia do AD01 e descrever melhor o comportamento do mesmo.

A incapacidade de encontrar uma sintonia somente com o ganho proporcional pode ser um reflexo do período utilizado na malha de controle, e por mais que o período foi determinado em razão do tempo de processamento de imagem, técnicas para reduzir esse tempo é importante. Além de ser importante realizar testes em relação ao ganho de fase do sistema, o que justificaria encontrar a estabilidade do AGV somente após o uso do ganho derivado.

Uma leve tendência do AD01 foi encontrada com os presentes ensaios, com novos testes em diferentes situações, pode-se encontrar outras tendências, assim auxiliando para melhor entendimento e descrição geral do comportamento do AGV, ajudando futuramente em uma sintonia mais adequada para o máximo de situações possíveis.

Capítulo 5

CONCLUSÃO

5.1 Síntese do contexto da proposta e dos objetivos

Uma plataforma de desenvolvimento de sistema de controle permite o desenvolvimento rápido de sistemas embarcados e um suporte adequado para a realização de testes. Muitos projetos quando desenvolvidos apresentam uma carência no suporte para projetos futuros, consequentemente necessitando reescrever linhas de códigos redundantes para realizar uma simples atualização, quando não necessário escrever todo o código novamente.

Quando a plataforma de desenvolvimento é feita de forma correta, é importante garantir a facilidade em manutenções e atualizações, assim auxiliando em testes e adaptações, e garantido que o projeto tenha um suporte adequado para manter um legado, possibilitando novas pesquisas e implementações no mesmo.

Com isso a criação da plataforma de desenvolvimento de sistemas de controle se tornou uma alternativa viável e necessária para o projeto, assim então, garantindo para os próximos responsáveis um suporte necessário em novas implementações, atualizações e manutenções.

O estudo de caso se baseou em uma demanda existente atualmente, sendo um AGV seguidor de linha, projetado para transporte de carga. A montagem do AD01 permite uma velocidade próxima de 2 m/s, e um transporte de cargas superiores a 100kg. O uso de uma câmera como sensor apresenta uma precisão aceitável, maior adaptabilidade e suporte para atualizações. A comunicação realizada entre a Toradex e os módulos IO da Beckhoff por ethercat não apresentou problemas, e o serviço criado na plataforma é disponibilizado com facilidade de uso para novas implementações.

A arquitetura SOA foi uma grande facilitadora em auxiliar na construção de um suporte para novas implementações, justificada por apresentar um desenvolvimento modular, não necessitando alterar outros serviços para criar novos serviços e sistemas.

A garantia de um suporte e facilidade para novos implementadores do AD01 está também na documentação criada, pois com os sete documentos apresentados escritos de forma correta, é possível que o implementador entenda como cada serviço funciona em diferentes níveis de dificuldades e proximidade com a linguagem.

Para sistemas de controle, é muito importante ter um bom controle de tempo e consumir o mínimo possível de consumo da CPU e RAM do hardware, o que apresenta a necessidade do desenvolvimento em uma linguagem de baixo nível, então sendo viável a implementação em C++. O C e C++ denominados linguagens nativas são amplamente utilizadas em sistemas embarcados, resultando em um grande suporte da comunidade, auxiliando muitos desenvolvedores a resolver problemas.

O uso de CMake se tornou indispensável no atual projeto, visto que, cada serviço foi organizado em diferentes pastas e em diferentes níveis, além de ser crucial para a compilação do OpenCV e da ethercat. Com uma documentação robusta e uma boa padronização, a implementação do CMake se torna uma tarefa simples, além do mesmo apresentar uma facilidade e suporte no momento da compilação cruzada.

A escolha do SO, além de oferecer uma facilidade no uso da placa, apresenta um suporte ativo pelo e-mail da Toradex. A Toradex também oferece uma documentação detalhada sobre o SO escolhido, além de estar disponível para download da imagem e o instalador, assim podendo armazenar as informações em um ambiente local, auxiliando na padronização, pois assim usamos sempre a mesma versão em todas as instalações.

A compilação cruzada como já apresentado se torna simples com o uso do Cmake, sendo necessário apenas importar do SO da Toradex a pastas sysroot, baixar o compilador para arquitetura ARM no Linux, e adicionar o arquivo toolchain.

Como o projeto apresenta uma plataforma de desenvolvimento para sistemas de controle, é ideal que seja implementado o controlador. O uso do PID se tornou viável, apresentando uma estabilidade em situações de linha reta e curvas para ambos os lados, assim possibilitando o auxílio para novas implementações para sistemas de controle na plataforma.

Apresentados a discussões acima, o objetivo geral e objetivos específicos do trabalho foram concluídos com êxito. Validando um plataforma de desenvolvimento de sistemas de controle, em um estudo de caso aplicado em um AGV. A plataforma oferece um suporte para que novos implementadores possam realizar alterações, inserir novos módulos e realizar devidas manutenções e testes. Consequentemente eliminando uma grande dor presente no TEAR e em outros laboratórios de pesquisa, a dificuldade de manter o legado de um projeto.

Como produto final, segue o Apêndice C, o relatório técnico de software. Sendo um documento completo, oferecendo o entendimento de cada serviço presente no AD01, e o documentado suporte para novas implementações do projeto. Através desse documento o legado do projeto será mantido e o será possível realizar as manutenções e melhorias no código.

5.2 Contribuições e delimitações

O presente trabalho auxiliou em um sistema de desenvolvimento que garante o legado das pesquisas. Obedecendo o padrão de documentação e construção do código, novas pesquisas podem ser implementadas com facilidade.

Para que o AD01 se torne um produto final, novas implementações devem ser realizadas, mas serão feitas com um suporte adequado, possibilitando o desenvolvimento futuramente de um produto seguro e ideal para o mercado.

A documentação de código auxiliou a concluir os objetivos do trabalho, mas também apresentou uma contribuição importante na forma de desenvolvimento. Visto que para que no momento do desenvolvimento vários erros possam ocorrer, mas podem ser solucionados olhando para níveis de implementação mais simples, então resultando com a presente documentação um suporte para entendimento do problema.

Quando feita de forma correta e sequencial, a documentação também auxilia á encontrar e corrigir erros antes da implementação do código, visto que todos os métodos devem ser descritos primeiramente em texto e em sequência em diagramas, possibilitando um entendimento pela leitura e por imagens.

Como limitação, o presente trabalho apresentou um foco no desenvolvimento de códigos, não apresentando um suporte para a parte mecânica do AD01, sendo muito importante também garantir que o projeto apresente o suporte para implementação e manutenção de módulos físicos também.

Para garantir que o AD01 se torne um produto final, é extremamente importante a realização de testes de hardware e de software, garantindo assim que o AGV apresente o desenvolvimento necessário e entenda suas limitações.

5.3 Trabalhos futuros

Dadas as limitações do projeto e conclusões, os trabalhos futuros são pesquisas para melhorar cada vez mais o desempenho e entendimento do AD01. Com isso, a realização de mais testes para a sintonia do AGV se torna um ponto importante para novas implementações.

Os testes e sintonia realizados no presente trabalho apenas justificaram que a plataforma de desenvolvimento de sistemas de controle funciona. Mas foram limitados por razões de espaço e tempo. Então é de extrema importância a realização de testes com o AGV para frente e pra trás, em linhas retas, curvas, sequências de linha reta e curvas,

curva e linha reta, linha reta curva e linha reta, todas para esquerda e direita, pista em 8 e outros testes em ambientes maiores.

Assim então, com todos os testes apresentados, encontrar uma sintonia estável para todos os casos de testes ou a maioria dos casos de testes se torna viável, além de ser crucial para entender o funcionamento do AD01 e suas limitações.

Todos os ensaios citados também podem ser adaptados para diferentes cargas, velocidades e variações de velocidade da roda, resultando no melhor desempenho do controlador em diferentes situações.

Como a plataforma de ensaio permite a modularidade, é interessante a implementação de novos controladores além do PID tradicional. Visto que como o AD01 é desenvolvido para transporte de carga, e diferentes cargas apresentam diferentes massas, o uso de controladores adaptativos podem ser uma implementação viável para o produto final.

A implementação de um controlador fuzzy pode ser um caminho, pois ele apresenta um melhor desempenho em sistemas não lineares, caso do AGV, além de ser um controlador adaptativo, resultando em um controlador que possa solucionar os problemas de duas limitações presentes no PID tradicional.

Ao desenvolver uma plataforma de desenvolvimento, a documentação dos testes é crucial, pois as realizações dos testes devem estar de acordo com as limitações e funcionalidades presentes na própria documentação de software. Então criar uma documentação de teste de software é importante.

Com o avanço da indústria e da análise de dados, a conectividade entre máquinas via sistemas unificados ou protocolos padrão tornou-se essencial. Portanto, incorporar um módulo de comunicação sem fio é estratégico para garantir que o AGV esteja preparado para as demandas do mercado atual.

Com o módulo sem fio, novas sintonias podem ser enviadas para o AD01 enquanto o mesmo está sendo utilizado, não sendo necessário rotinas de enviar o AD01 para uma manutenção, comunicar por conexão serial à um computador e definir novos parâmetros de sintonia.

Além de facilitar o uso de tecnologia de sistemas *ciberfisico*, onde computadores centrais possam monitorar o AD01 e realizar simulações em tempo real, assim então, por meio das simulações e dados monitorados buscar melhores sintonias, e encontrar comportamentos atípicos do AGV, encaminhando-o para manutenções preditivas e corretivas.

O monitoramento dos erros do AGV foi realizado por meio dos logs salvos internamente no AD01, o que pode prejudicar a frequência do sistema de controle. A implementação de uma medição externa de AGVs é uma demanda importante para ser implementada, visto que, será reduzido o uso de rotinas internas presentes no AD01, facilitando o monitoramento e coleta de dados.

O sistema de controle do AGV não se estabiliza somente com o controlador Kp pode ser um sinal de que a frequência do sistema está muito baixa, a ponto de o controlador não conseguir atuar no sistema. Um estudo analisando a margem de fase do sistema é extremamente importante, visto que a frequência utilizada é a máxima possível com o atual sistema de tratamento de imagem. Melhorar o sistema de cálculo de erro pode ser uma alternativa para aumentar a frequência, pois ele é o serviço que mais consome tempo no sistema de controle de movimento. Para que o sistema possa ser melhorado, dois caminhos podem ser seguidos. Primeiro utilizar hardwares com capacidade de processamento melhores, segundo, reduzir o custo de processamento através de outros filtros ou técnicas de processamento de imagem.

REFERÊNCIAS

ALKAR, Ali Ziya; KARACA, Mehmet Atif. An Internet-Based Interactive Embedded Data-Acquisition System for Real-Time Applications. **IEEE Transactions on Instrumentation and Measurement**, v. 58, n. 3, p. 522–529, 2009.

BARROS, Alistair; OUYANG, Chun; WEI, Fuguo. Static Analysis for Improved Modularity of Procedural Web Application Programming Interfaces. **IEEE Access**, v. 8, p. 128182–128199, 2020.

BERTOZZI, Massimo; CHEN, Bo; ZINGARETTI, Primo. **Introduction to the Special Issue on Applications of Mechatronic and Embedded Systems (MESA) in ITS**. **IEEE Transactions on Intelligent Transportation Systems**Institute of Electrical and Electronics Engineers Inc., , 1 fev. 2018.

BHADORIA, Robin Singh *et al.* Cone Model in Resource Provisioning for Service-Oriented Architecture System: An Effective Network Management to the Internet of Things. **IEEE Access**, v. 10, p. 61385–61397, 2022.

BROWN, W. Microsystems Modular Programming in PL/M. **Computer**, v. 11, n. 3, p. 40–46, 1978.

CHENGMENG, Xue; ZHEN, Zeng. The construction of SOA-based enterprise information management system. *In*: 2010.

CUCINOTTA, Tommaso *et al.* A real-time service-oriented architecture for industrial automation. **IEEE Transactions on Industrial Informatics**, v. 5, n. 3, p. 267–277, ago. 2009.

DORF, Richard C.; BISHOP, Robert H. **Sistemas de Controle Modernos**. Tradução: Rubens Junqueira Magalhães Afonso. *[S.l.: S.n.]*.

DOS REIS, Wallace Pereira Neves; MORANDIN JUNIOR, Orides. Sensors applied to automated guided vehicle position control: a systematic literature review. **International Journal of Advanced Manufacturing Technology**, v. 113, n. 1–2, p. 21–34, 1 mar. 2021.

GEER, David. The OS Faces a Brave New World. **Computer**, v. 42, n. 10, p. 15–17, 2009.

GOODACRE, J.; SLOSS, A. N. Parallelism_and_the_ARM_instruction_set_architecture. *[S.d.]*.

GUEDES, G. T. A. **UML 2 Uma Abordagem Pratica**. 3. ed. São Paulo: Novatec, 2018.

HAGENGREN, Bo; SANDBERG, Ulf. A Control Center Laboratory with a Realistic Power System Model. **IEEE Transactions on Power Systems**, v. 1, n. 3, p. 292–297, 1986.

JIANG, Muhui *et al.* Examiner-Pro: Testing Arm Emulators Across Different Privileges. **IEEE Transactions on Software Engineering**, v. 50, n. 11, p. 2786–2806, 2024.

KARADEMIR, Funda; CETIN, Semih. Use of Service-Oriented Architecture in Satellite Interoperability. *In*: 2007.

KE, Po-Chuan *et al.* A Service-Oriented Framework for An Integrated Study of Intelligent Data Collection and Application. *In*: 2023.

KHALGUI, Mohamed *et al.* Reconfigurable Multiagent Embedded Control Systems: From Modeling to Implementation. **IEEE Transactions on Computers**, v. 60, n. 4, p. 538–551, 2011.

KYÖSTI, Petter; LINDSTRÖM, John. SOA-Based Platform Use in Development and Operation of Automation Solutions: Challenges, Opportunities, and Supporting Pillars towards Emerging Trends. **Applied Sciences**, v. 12, n. 3, 2022.

LYNCH, Liam *et al.* Automated Ground Vehicle (AGV) and Sensor Technologies- A Review. *In*: Limerick, Ireland: IEEE, 4 dez. 2018. Disponível em: <https://ieeexplore-ieee-org.ez31.periodicos.capes.gov.br/document/8603640>. Acesso em: 23 mar. 2025

MENDOZA-PITTI, Luis *et al.* Towards a Service-Oriented Architecture for the Energy Efficiency of Buildings: A Systematic Review. **IEEE Access**, v. 9, p. 26119–26137, 2021.

OLIVEIRA, Diogo de; REIS, Wallace dos; MORANDIN JUNIOR, Orides. A Qualitative Analysis of a USB Camera for AGV Control. **Sensors**, v. 19, n. 19, 1 out. 2019.

ORTIZ, S. Embedded OSs gain the inside track. **Computer**, v. 34, n. 11, p. 14–16, 2001.

PÜLLEN, Dominik *et al.* A Security Process for the Automotive Service-Oriented Software Architecture. **IEEE Transactions on Vehicular Technology**, v. 73, n. 4, p. 5036–5053, 2024.

REIS, Wallace Pereira Neves dos; COUTO, Giselle Elias; JUNIOR, Orides Morandin. **Automated guided vehicles position control: a systematic literature review**. **Journal of Intelligent Manufacturing**Springer, , 1 abr. 2023.

RIBEIRO, M. C. **Análise e melhoria dos processos de uma empresa industrial gráfica e estudo da viabilidade de implementação de um sistema de AGVs**. Dissertação (Mestrado Em Engenharia Industrial)—Braga, Portugal: Universidade do Minho, out. 2022.

RODRIGUES D. ET AL. Application of SOA in Safety-Critical Embedded Systems. *In*: LEE GEUK AND HOWARD, Daniel and Ślezak Dominik (org.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

SABATTINI, Lorenzo *et al.* Technological roadmap to boost the introduction of AGVs in industrial applications. *In*: Cluj-Napoca, Romania: IEEE, 5 jul. 2013. Disponível em: <https://ieeexplore.ieee.org/document/6646109>. Acesso em: 21 mar. 2025

SILVA, Edna Lúcia da; MENEZES, Muskat Estera. **Metodologia da Pesquisa e Elaboração de Dissertação** . 3. ed. Florianópolis : *[S.n.]*.

SILVA, Felipe Madeira da. **SOA - Arquitetura Orientada a Serviços**. Monografia—São Paulo, SP: Universidade de São Paulo, jul. 2006.

TEYMOURIAN, Navid; IZADKHAH, Habib; ISAZADEH, Ayaz. A Fast Clustering Algorithm for Modularization of Large-Scale Software Systems. **IEEE Transactions on Software Engineering**, v. 48, n. 4, p. 1451–1462, 2022.

VEKIĆ, Marko S. *et al.* Ultralow Latency HIL Platform for Rapid Development of Complex Power Electronics Systems. **IEEE Transactions on Power Electronics**, v. 27, n. 11, p. 4436–4444, 2012.

WILL, Liane; KOPPEN, Veit; SAAKE, Gunter. Flexibility in SOA operations: The need for a central service component. *In*: Institute of Electrical and Electronics Engineers Inc., 2 dez. 2014.

WOLF, W. What is embedded computing? **Computer**, v. 35, n. 1, p. 136–137, 2002.

YEN, I. Ling *et al.* QoS-Reconfigurable Web Services and Compositions for High-Assurance Systems. **Computer**, v. 41, n. 8, p. 48–55, 2008.

YING-PEI, W. U.; TING-TING, S. H. U. Research on Information System Integration in Colleges Based on SOA. **Procedia Engineering**, v. 24, p. 345–349, jul. 2011.

SOUZA, Luís Fernando Ferreira de. Arquitetura de Software Orientada a Serviços para AGV. 2021. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) – Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2021.

Orientador: Prof. Dr. Orides Morandin Jr.

Apêndice A*

DOCUMENTO DE REQUISITOS

Documento de Requisitos

Serviço Cálculo de Erros (S5)

Introdução

O serviço deve calcular a distância e o ângulo do AGV em relação a linha guia conforme ilustrado nas imagens a seguir.

Figura 1: AGV na posição ideal

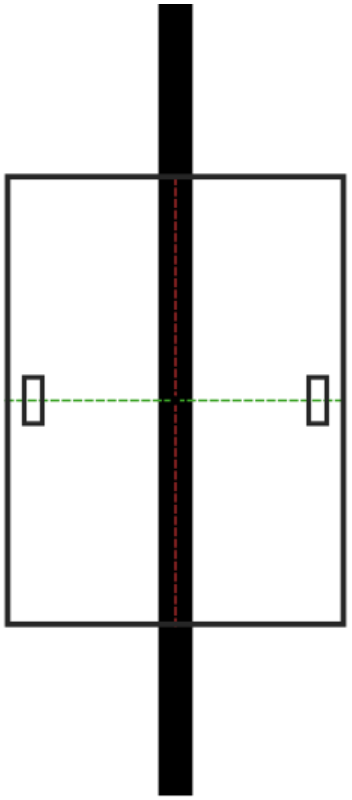
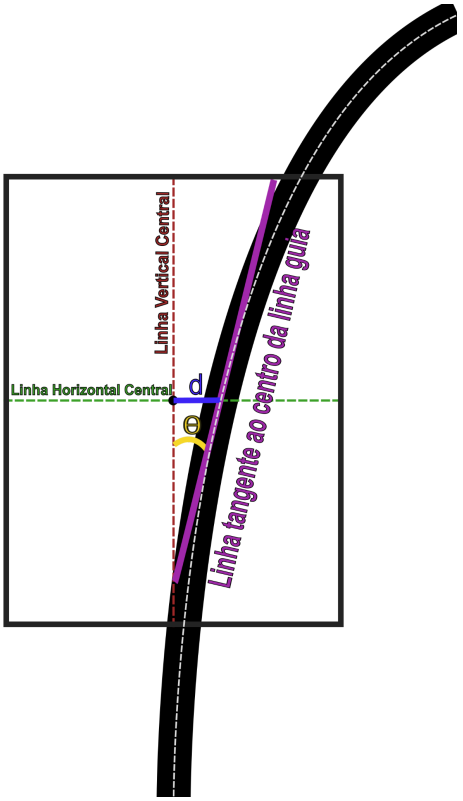


Figura 2: AGV erros e linhas de referência



A primeira imagem representa uma posição ideal do AGV em relação a linha guia, com os erros de ângulo e distância iguais a zero. A segunda imagem ilustra o erro em ângulo **θ** e distância **d**.

Cálculo da distância

A distância (**distanceError**) representada na imagem pela linha azul e a letra **d**, é a diferença entre o centro do AGV e o ponto central da linha guia, medido no cruzamento com a linha horizontal central.

Cálculo do ângulo

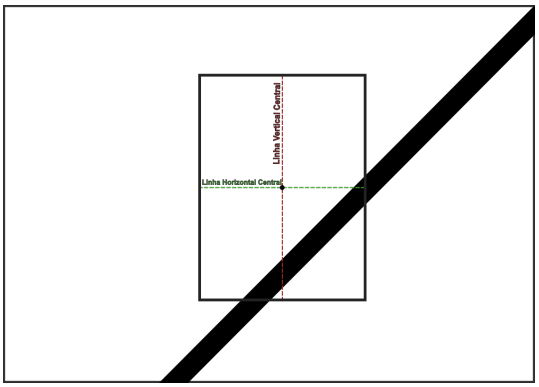
O ângulo (**angleError**) representado na imagem pela linha amarela e a letra **θ**, é formado entre a linha vertical central do AGV (vermelha) e a tangente da linha guia (roxo), no ponto de cruzamento da linha guia e a linha horizontal central do AGV (verde).

Erros

O serviço deve considerar o erro em distância positiva quando estiver à direita do centro da imagem e negativo quando estiver à esquerda. O ângulo positivo é considerado quando estiver no sentido horário da linha central vertical e negativo quando estiver no sentido anti horário.

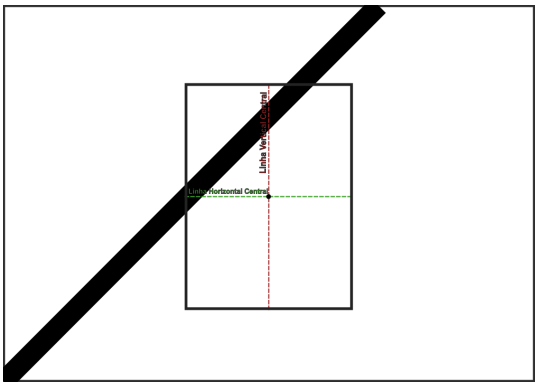
A seguir está apresentado o comportamento desejado do ângulo e distância de acordo com a imagem. Outras combinações de valores podem ocorrer, as imagens são apenas para demonstrar que os valores podem ser positivos, negativos ou zero.

Figura 3: Erros em distância e ângulo positivos



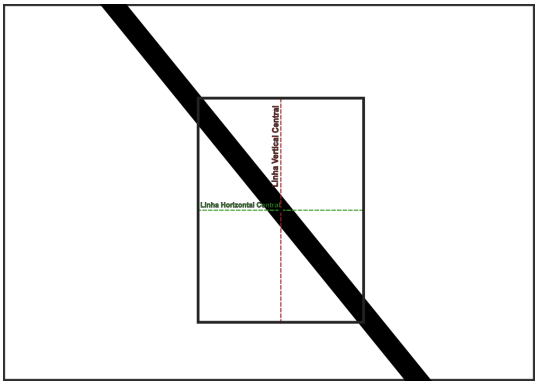
Distância e ângulo com valores positivos.

Figura 4: Erro em distância negativo e ângulo positivo



Distância com valor negativo e ângulo com valor positivo.

Figura 5: Erro em distância zero e ângulo negativo



Distância com valor zero e ângulo com valores positivos.

Descrição Geral

Funções do Produto

Cálculo do erro em distância e ângulo do AGV em relação a linha guia.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

A linha guia deve ser uma cor que se destaca em relação ao piso.

Suposições e Dependências

- O sistema presume que os parâmetros, contraste, fps e foco estejam corretamente calibrados no momento da operação.
- O serviço depende que as variáveis globais **errorDefinition**, **angleError** e **distanceError** já estejam criadas.

Requisitos Específicos

Requisitos Funcionais

RF1 - Calcular o ângulo e a distância da linha guia em relação ao centro da imagem processada.

RF2 - Os erros de ângulo e distância devem ser normalizados no intervalo de -100 a +100, e sua margem de erro é de 1mm para distância e 1° para o ângulo.

RF3 - O serviço deve modificar as variáveis globais **angleError** e **distanceError**.

RF4 - O serviço deve ler a variável global **errorDefinition**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender imagens distorcidas.
  - RNF2 - Atender imagens rasuradas.
  - RNF3 - Atender imagens desfocadas
  - RNF4 - Atender imagens com linha de referência com uma cor diferente do preto.
- **Manutenibilidade**
  - RNF5 - A manutenção como incremento e função pode-se realizar a validação de ruídos gerados por diferentes fontes de luz externa.

Apêndice B*

DOCUMENTO DE DEMANDA DE PRODUÇÃO

Documento de Demanda de Produção

Serviço Cálculo de Erros (S5)

Descrição de métodos

Filtragem da imagem

Após a imagem ser capturada, será necessário trabalhar a imagem até transformá-la em binário. Usa-se três etapas:

1. Converter a imagem para **escala de cinza**;
2. Aplicar o filtro **gaussiano**;
3. Aplicar o método de **binarização**. A binarização utilizará um limiar fixo para separar os elementos da linha de referência do fundo.

Para as aplicações dos filtros a biblioteca utilizada será o OpenCV.

Escolha do erro

Deve-se escolher o erro de ângulo ou distância ou ambos.

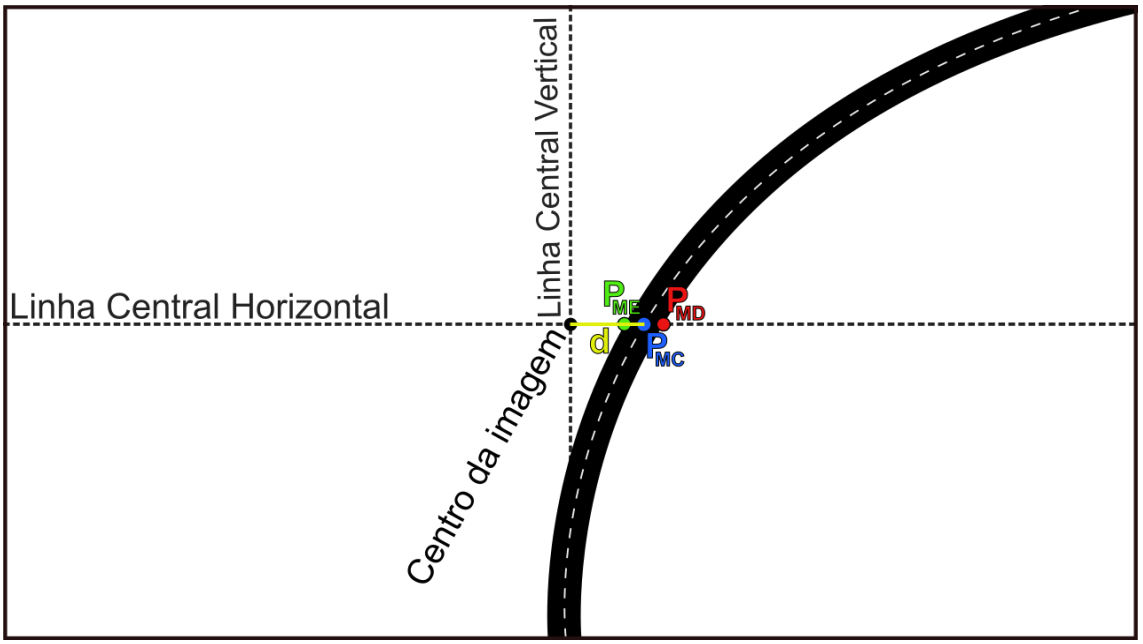
Utilizar a variável global **errorDefinition**. Quando a variável for igual a 1 será realizado somente o cálculo de distância. Quando a variável for igual a 2 será realizado somente o cálculo de ângulo. Quando a variável for diferente dos valores citados, será calculado distância e ângulo.

Cálculo da Distância

O cálculo da distância (**d**) é realizado com base nas coordenadas de dois pontos e uma linha de referência na imagem:

1. A linha de referência está localizada a 50% da altura da imagem e é representada pela linha cinza denominada de Linha Central Horizontal.
2. O primeiro ponto preto encontrado nessa Linha Central Horizontal está marcado na imagem pela cor verde denominado $\mathbf{P_{ME}}$.
3. O último ponto preto encontrado na mesma Linha Central Horizontal está marcado na imagem pela cor vermelha denominado $\mathbf{P_{MD}}$.

Figura 1 Linhas e pontos de referência para cálculo de distância



O objetivo de encontrar os dois pontos é determinar o ponto central entre eles $\mathbf{P_{MC}}$, representado pelo ponto azul na imagem. E enfim encontrar a distância entre o ponto central e o centro da imagem.

Usando as seguintes equações encontra-se a distância da linha guia (**d**).

$$x_{MC} = \frac{x_{ME} + x_{MD}}{2} \quad \text{[equação 1]}$$

$$Ponto\ central\ da\ imagem = \frac{número\ de\ colunas}{2} \quad \text{[equação 2]}$$

$$d_{pixels} = x_{MC} - Ponto\ central\ da\ imagem \quad \text{[equação 3]}$$

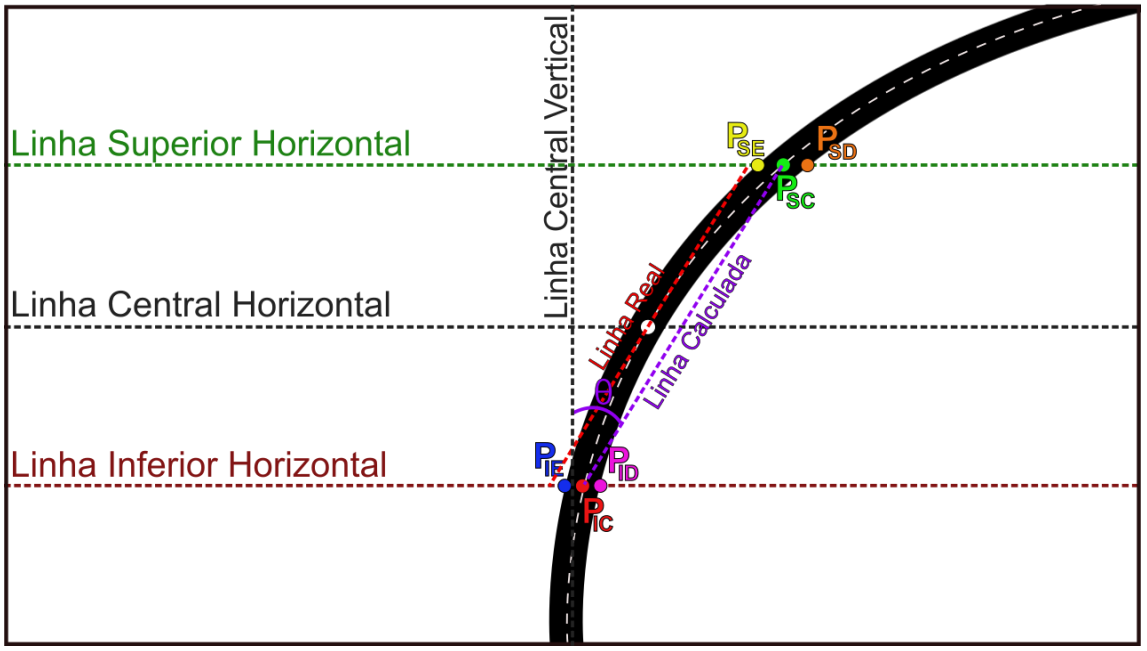
$$d = \frac{d_{pixels}}{número\ de\ colunas/2} \cdot 100 \quad \text{[equação 4]}$$

Cálculo do Ângulo

O cálculo do ângulo (**θ**) é realizado com base nas coordenadas de quatro pontos e duas linhas de referência na imagem, esses cálculos resultam em uma linha paralela a Linha Real que desejamos calcular o ângulo, portanto ao calcular o ângulo dessa nova linha temos o valor do ângulo da Linha Real. Os quatros pontos são:

1. O primeiro ponto preto encontrado na linha localizada a 25% da altura da imagem, que está representada pela linha verde. Esse ponto está marcado na imagem pela cor amarela denominado $\mathbf{P_{SE}}$.
2. O último ponto preto encontrado na linha localizada a 25% da altura da imagem, que está representada pela linha verde. Esse ponto está marcado na imagem pela cor laranja $\mathbf{P_{SD}}$.
3. O primeiro ponto preto encontrado na linha localizada a 75% da altura da imagem, que está representada pela linha vermelha. Esse ponto está marcado na imagem pela cor azul $\mathbf{P_{IE}}$.
4. O último ponto preto encontrado na linha localizada a 75% da altura da imagem, que está representada pela linha vermelha. Esse ponto está marcado na imagem pela cor roxo $\mathbf{P_{ID}}$.

Figura 2 Linhas e pontos de referência para cálculo de ângulo



O objetivo de encontrar os quatro pontos é para determinar dois pontos:

- Ponto representado pela cor verde $\mathbf{P_{SC}}$, sendo o meio entre os pontos $\mathbf{P_{SE}}$ e $\mathbf{P_{SD}}$.
- Ponto representado pela cor vermelha $\mathbf{P_{IC}}$, sendo o meio entre os pontos $\mathbf{P_{IE}}$ e $\mathbf{P_{ID}}$.

Usando as seguintes equações encontra-se o ângulo da linha guia.

$$x_{SC} = (x_{SD} + x_{SE}) / 2 \quad \text{[equação 5]}$$

$$y_{SC} = y_{SD} = y_{SE} \quad \text{[equação 6]}$$

$$x_{IC} = (x_{ID} + x_{IE}) / 2 \quad \text{[equação 7]}$$

$$y_{IC} = y_{ID} = y_{IE} \quad \text{[equação 8]}$$

$$\Delta_x = x_{IC} - x_{SC} \quad \text{[equação 9]}$$

$$\Delta_y = y_{IC} - y_{SC} \quad \text{[equação 10]}$$

$$\theta_{rad} = - arctan(\frac{\Delta_y}{\Delta_x}) \quad \text{[equação 11]}$$

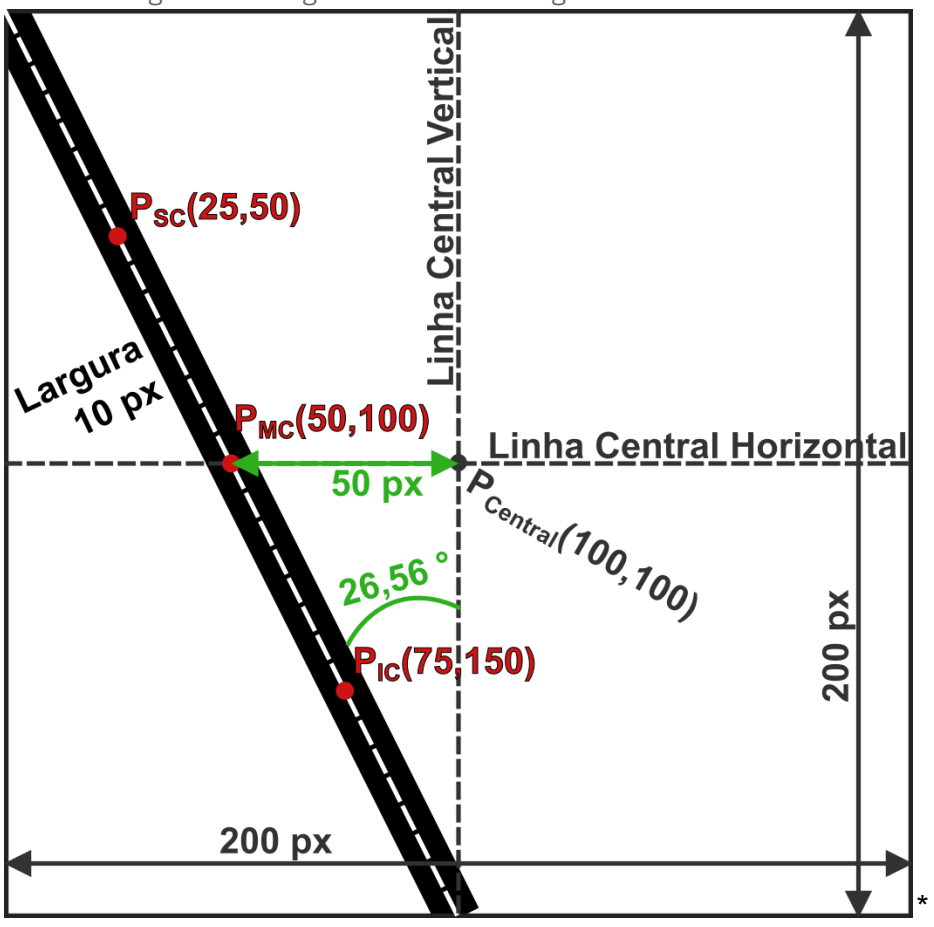
$$\theta = sen(\theta_{rad} - \pi/2) \cdot 100 \quad \text{[equação 12]}$$

A função trigonométrica **arctan** está com o sinal negativo para que o sentido horário seja positivo e o sentido anti-horário seja negativo.

Na equação 12 ocorre a subtração por $\pi/2$ para mudar a orientação, pois a referência do openCV é a linha central vertical, e a referência para o serviço é a linha central horizontal.

Exemplo de uso e serviço

1. Com a seguinte linha guia descrita na imagem.



Respeitando toda a descrição deve ser criado a linha guia sem a necessidade da linha branca tracejada no centro, das linhas de referência e das descrições em texto.

2. Os resultados retornados devem ser:
    - **angleError** = -44,72
    - **distanceError** = - 50

Referências

TCC Rodrigo Santos.

Tear_DrWallacePNR_RelatorioTecnico_Relatorio_Tecnico_20220805

Histórico de alterações

14/11/24:

- Reestruturação completa do documento para englobar uso geral do serviço, e não apenas para AGVs.
- Adição de métodos de obtenção e inserção de valores dos ganhos KP e KD, em falta desde a última alteração do documento.

23/11/24:

- Alteração completa nos métodos, removendo os métodos computacionais e descrevendo o que é método e como funciona PID que será utilizado pelo serviço.

25/11/24:

- Escrita mais direta da documentação.
- Adição da fórmula para explicar o algoritmo.

26/11/24

- Adição do cálculo dos termos.
- Adição das variáveis globais t, ti e td.

11/12/24

- Alteração das imagens
- Novo exemplo com valores não triviais.

16/01/25

- Adição da linha paralela na imagem do cálculo do ângulo.

01/09/25

- Alteração dos filtros e da sequência, de mediana, escala de cinza e binarização, para escala de cinza, gaussiana e binarização.

Apêndice C*

RELATÓRIO TÉCNICO DE SOFTWARE

Relatório Técnico de Software

Introdução

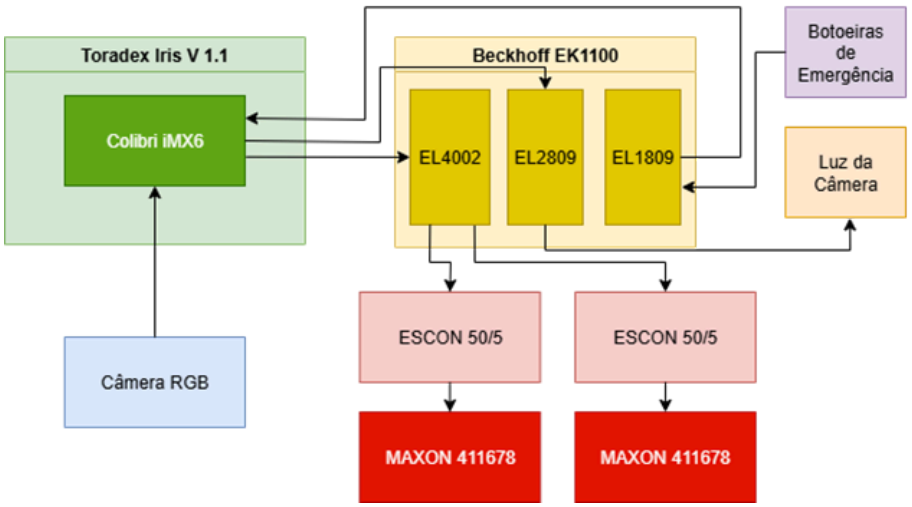
O presente relatório se destina a apresentar toda documentação para a construção do software da plataforma de desenvolvimento de sistemas de controle, aplicado no AD01. O AD01 é um robô de tração diferencial em duas rodas.

Como sensor de erro, o AD01 utiliza uma câmera RGB posicionada no centro geométrico do veículo. O controle do AD01 é feito através de um computador de sistemas embarcados Toradex Colibri iMX6 DualLite 512MB na placa Iris V 1.1 que se comunica por protocolo de comunicação Ethercat com uma remota Beckhoff EK 1100. A remota é responsável pelo controle das entradas e saídas utilizando os módulos EL4002 para saídas analógicas, EL2809 para saídas digitais e EL1809 para entradas digitais. Para o movimento do AGV são utilizados dois motores da Maxon 411678 dispostos no eixo transversal central do veículo, no mesmo eixo da câmera, controlados por drivers ESCON 50/5..

Diagrama de Componentes

Para um melhor entendimento do AD01, a seguir está presente um diagrama de blocos com todos os componentes utilizados para o desenvolvimento do AD01 até o momento, e suas relações com os demais componentes.

Figura 1: Diagrama de Componentes do AD01



Arquitetura de Software

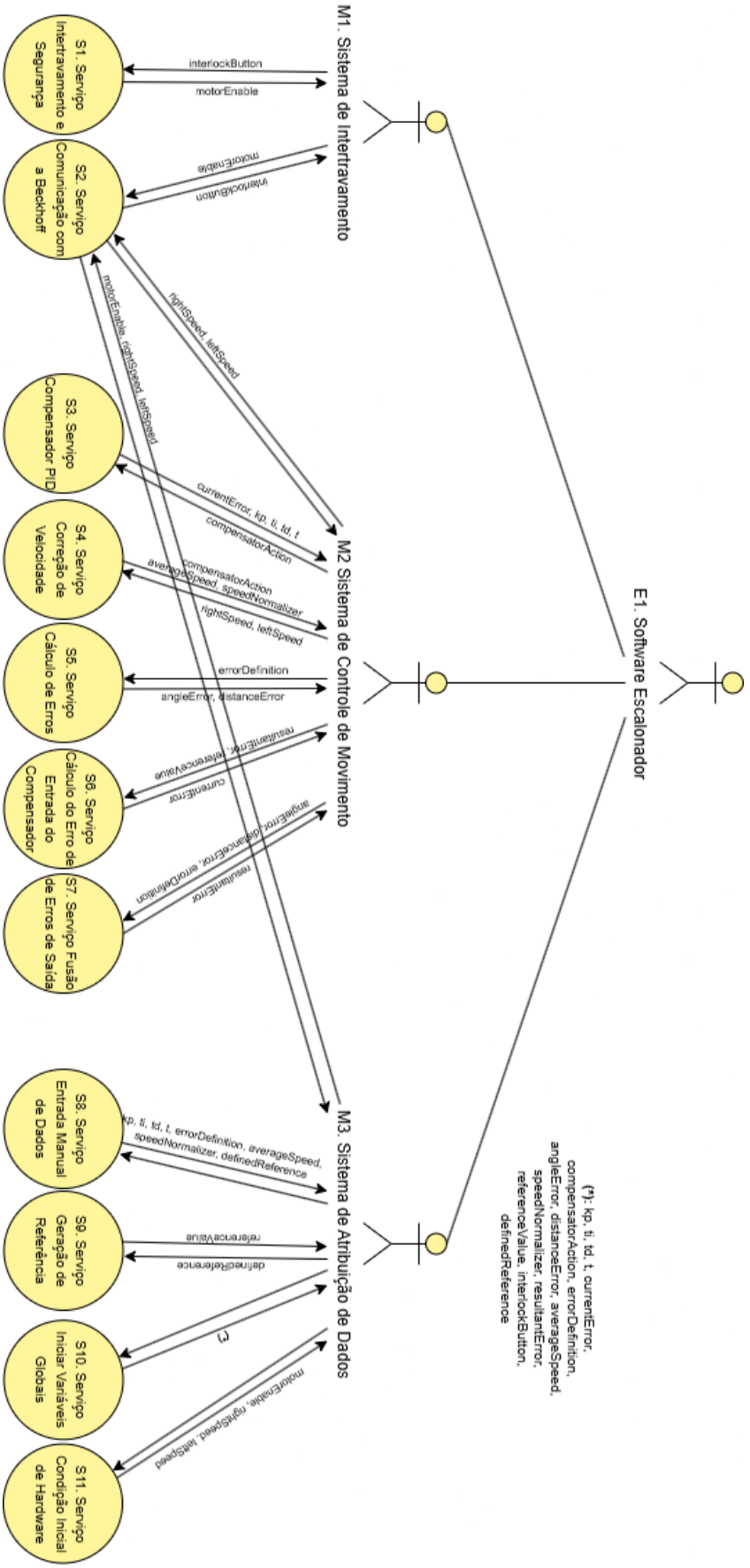
Para o desenvolvimento do software, a arquitetura escolhida foi baseada na arquitetura Orientada a Serviços (SOA), sendo basicamente uma arquitetura que organiza o software em serviços independentes e reutilizáveis, que se comunicam por interfaces padronizadas. Essa abordagem promove flexibilidade, interoperabilidade e agilidade na integração de sistemas.

Com 11 serviços numerados de S1 à S11, 3 majoritários numerados de M1 à M3 e um Escavador E1.

Cada serviço deve ser responsável por executar uma função específica, quando solicitada pelo majoritário, que por sua vez executa a sequência de chamadas de cada serviço. Os majoritários são

requisitados pelo escalonador, que é responsável em controlar o tempo e chamada de cada majoritário. A arquitetura de software pode ser representada pela Figura 2.

Figura 2: Arquitetura Geral de Software do AD01



Documento de Requisitos

Software Escalonador

Descrição Geral

Funções do Produto

Realizar a ordem de execução dos majoritários, em um determinado período de tempo

Características do Usuário

Este software é consumido pelo Sistema Operacional do AGV.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O software depende da correta execução do Sistema de Intertravamento, Sistema de Controle de Movimento e Sistema de Atribuição de Dados.
- O software depende da variável global **t**.

Requisitos Específicos

Requisitos Funcionais

RF1 - Solicitar o Sistema de Atribuição de Dados, somente no início da execução.

RF2 - Solicitar o Sistema de Controle de Movimento.

RF3 - Solicitar o Sistema de Intertravamento.

RF4 - Realizar o loop de execução entre Sistema de Controle de Movimento e Sistema de Intertravamento em um período de tempo fixo e determinado.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - Não identificado.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Software Escalonador

Descrição de métodos

Sequência de ações

Para que o sistema funcione a sequência de execução deve ser realizada:

1. Solicitar o Sistema de Atribuição de Dados somente uma vez, no início da execução do software
2. Realizar um loop em um determinado período de tempo na seguinte ordem:
    1. Sistema de Controle de Movimento.
    2. Sistema de Intertravamento.

Exemplo de uso e serviço

Quando acionado o software escalonador a seguinte sequência deve ser executada:

1. Solicitar o Sistema de Atribuição de Dados somente uma vez, no início da execução do software
2. Em loop:
    1. Sistema de Controle de Movimento.
    2. Sistema de Intertravamento.

Referências

Histórico de alterações

-

Diagramas

Software Escalonador

Diagrama de Casos de Uso

Figura E1.1: Diagrama de Casos de Uso

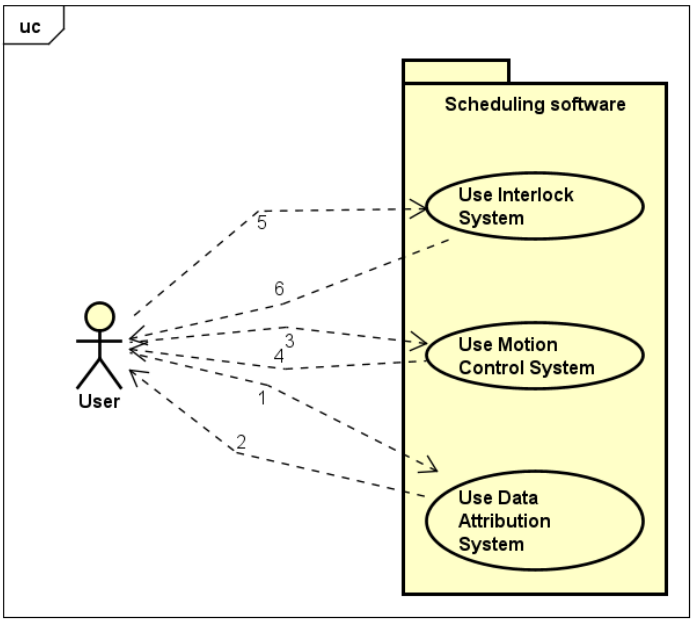


Diagrama de Atividades

Figura E1.2: Diagrama de Atividades

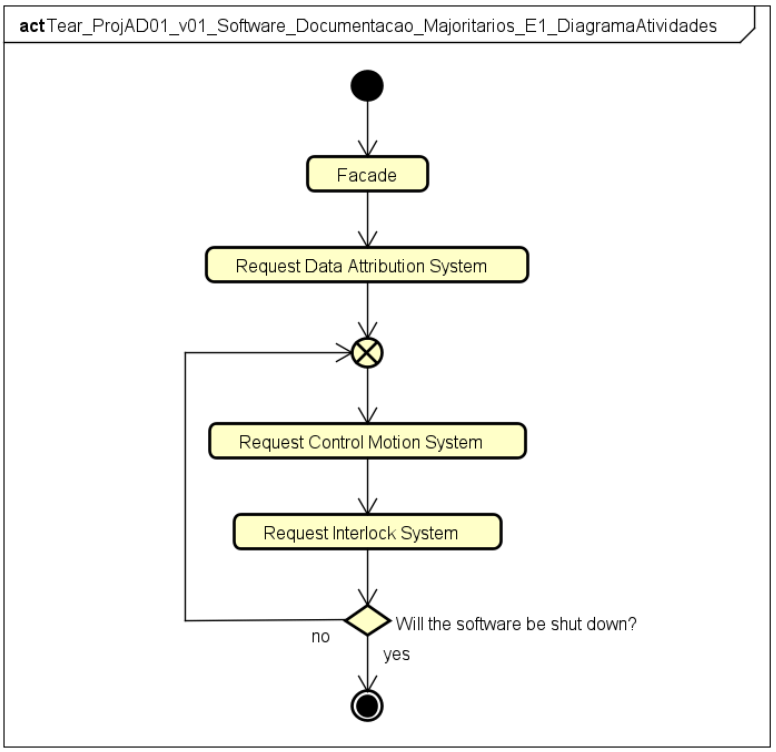


Diagrama de Sequência

Figura E1.3: Diagrama de Atividades

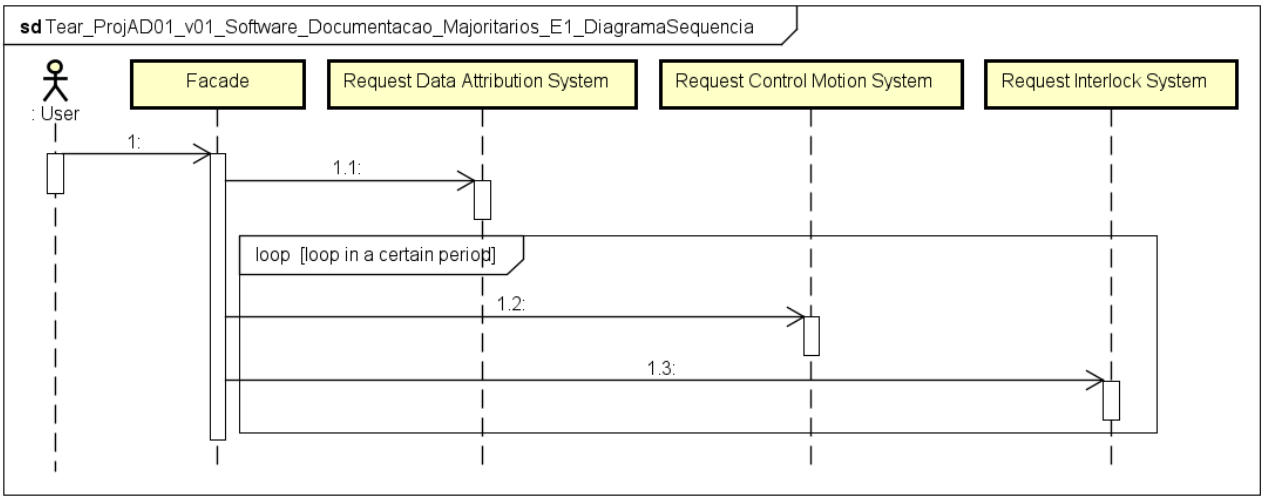
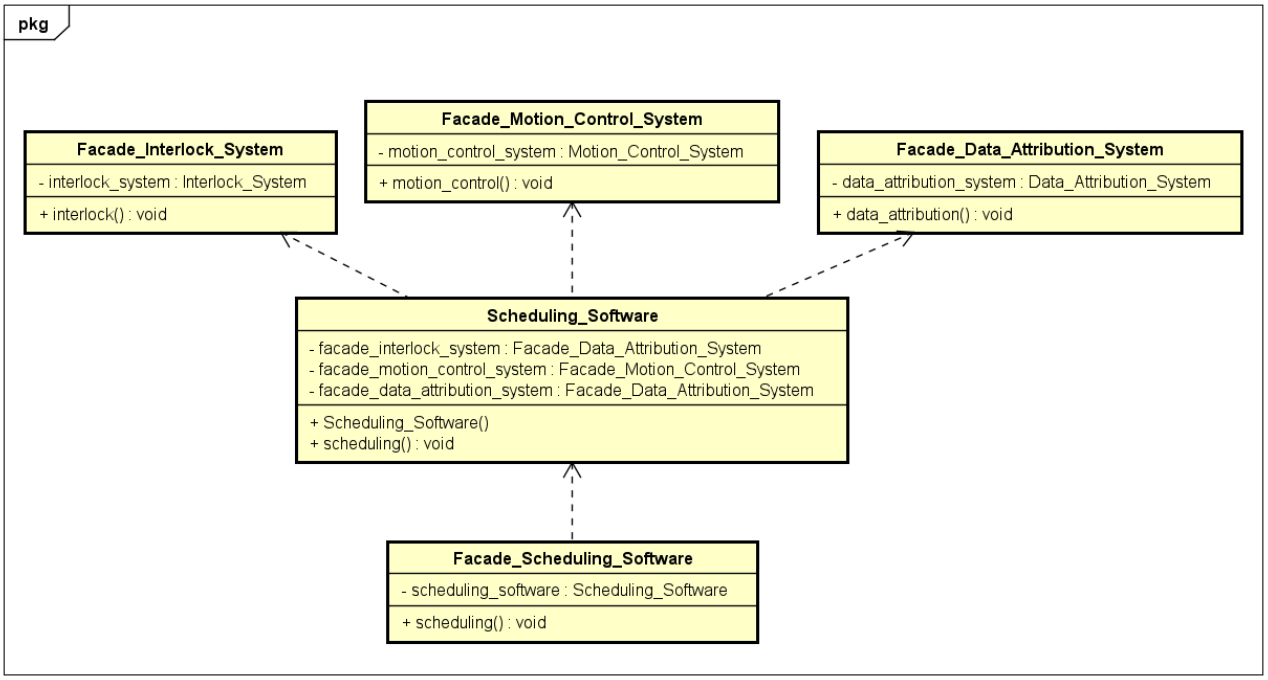


Diagrama de Classes

Figura E1.4: Diagrama de Classes



Pseudocódigo

Software Escalonador

Scheduling_Software

include: Facade_Interlock_System.h
Facade_Motion_Control_System.h
Facade_Data_Attribution_System.h
Facade_Start_Global_Variables.h
define: N/A
variables: facade_interlock_system // Facade_Interlock_System instantiation from the Facade_Interlock_System class
facade_motion_control_system // Facade_motion_control_system instantiation from the Facademotion_control_system class
facade_data_attribution_system // Facade_data_attribution_system instantiation from the Facade_data_attribution_system class

function Scheduling_Software() // Constructor
// Request Data Attribution System
facade_data_attribution_system.data_attribution()
end function

function scheduling()
while true
// Request Control Motion System
facade_motion_control_system.motion_control()

// Request Interlock System
facade_interlock_system.interlock()

delay in **t** ms

end function

Facade_Scheduling_Software

include: NA
define: NA
variable: scheduling_software // Scheduling_Software instantiation from the Scheduling_Software

function scheduling()
scheduling_software.scheduling()
end function

Documento de Requisitos

Sistema de Intertravamento (M1)

Introdução

O Sistema de Intertravamento deve interromper o movimento do AGV AD01, quando alguma botoeira externa for acionada. Este sistema é o responsável pela segurança do AGV.

Descrição Geral

Funções do Produto

Interromper o movimento do AGV.

Características do Usuário

Este sistema é consumido pelo Software Escalonador..

Restrições Gerais

Não identificado.

Suposições e Dependências

- O sistema depende da correta execução do Serviço de Intertravamento e Segurança, e Serviço de Comunicação com a Beckhoff.
- O serviço depende de botoeiras de segurança instaladas no AGV.

Requisitos Específicos

Requisitos Funcionais

RF1 - Solicitar a leitura digital das botoeiras de segurança.

RF2 - Interromper a execução dos demais serviços.

RF3 - Solicitar a escrita digital de habilitar ou desabilitar os drivers dos motores.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Apresentar um tempo de resposta inferior a 100ms.
  - RNF2 - O serviço deve ser operacional em toda execução do software.
- **Manutenibilidade**
  - RNF3 - A manutenção como incremento e função pode-se realizar uma saída de sinalização da interrupção.

Documento de Demanda de Produção

Sistema de Intertravamento (M1)

Descrição de métodos

Solicitação do Serviço de Comunicação com Beckhoff (S2)

O M1 deve solicitar a escrita e leitura do sinal das botoeiras de emergência por meio do Serviço de Comunicação com Beckhoff, seja para habilitar ou desabilitar os drivers dos motores.

Solicitação do Serviço de Intertravamento de Segurança (S1)

O M1 deve solicitar o serviço de Intertravamento de Segurança (S1) caso a leitura das botoeiras externas apresente um resultado positivo, ou seja, caso um botão seja acionado pelo usuário físico no AGV AD01.

Exemplo de uso e serviço

1. O S2 inicia o giro dos motores do AGV;

2. Quando acionado a botoeira de emergência, os drivers dos motores são desativados pelo sinal indicado pelo S1.

Referências

Não identificado.

Histórico de alterações

28/01/25:

- Criação do documento.

04/03/25:

- Revisão de métodos para bater com o diagrama de casos de uso.

Diagramas

Sistema De Intertravamento (M1)

Diagrama de Casos de Uso

Figura M1.1: Diagrama de Casos de Uso

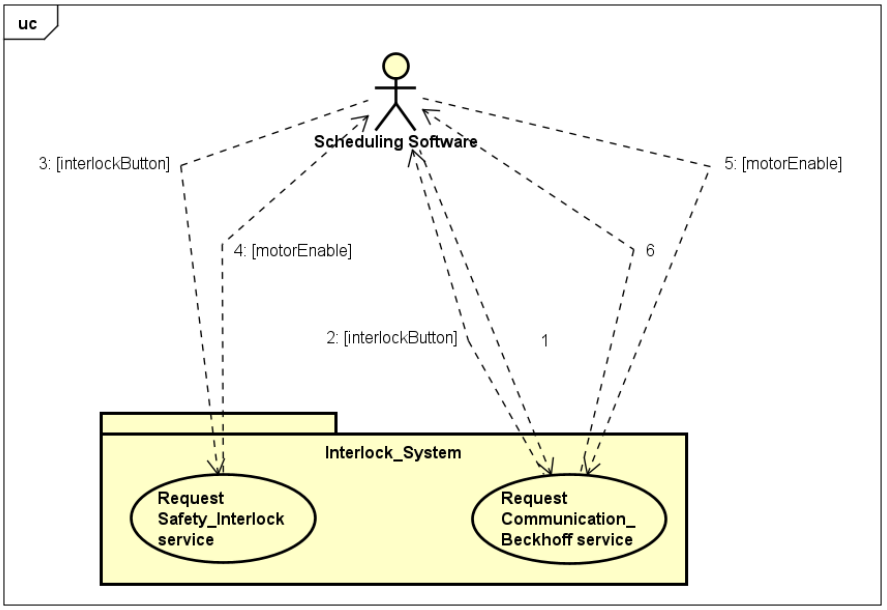


Diagrama de Atividades

Figura M1.2: Diagrama de Atividades

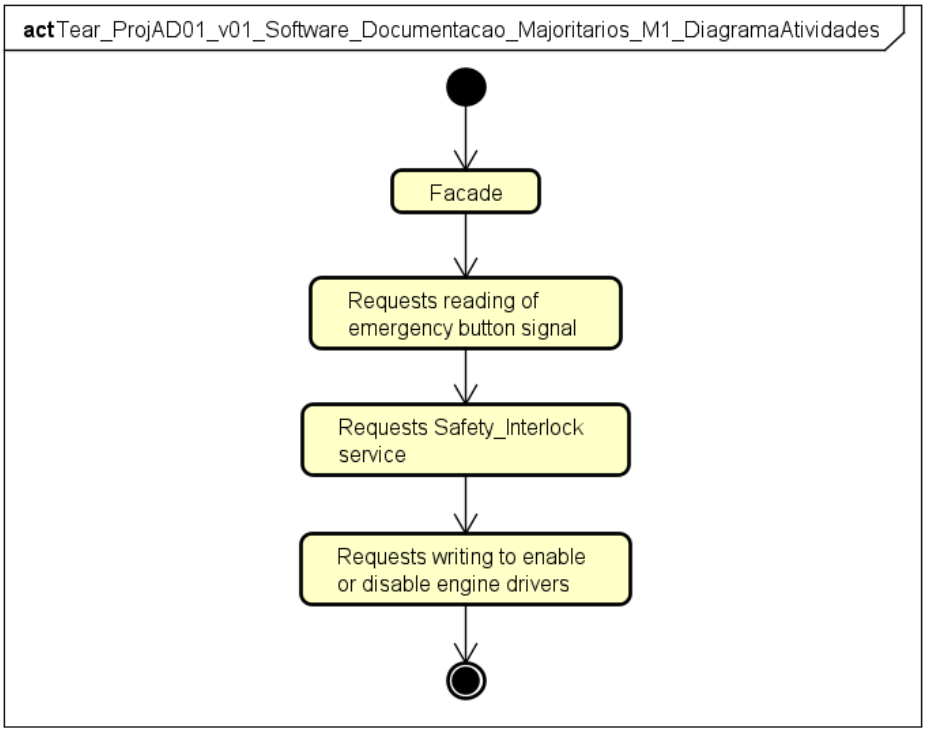


Diagrama de Sequência

Figura M1.3: Diagrama de Sequência

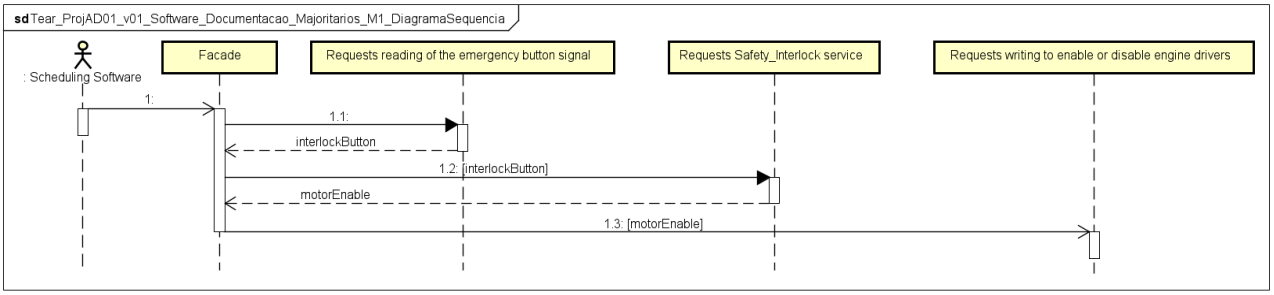
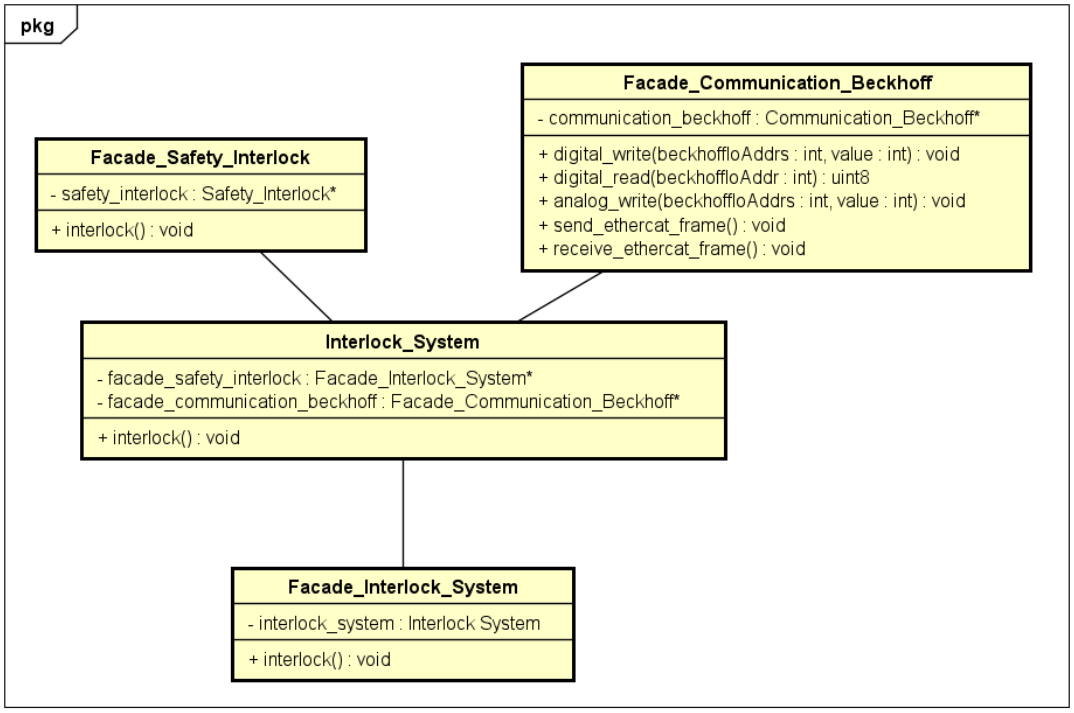


Diagrama de Classes

Figura M1.4: Diagrama de Classe



Pseudocódigo

Sistema de Intertravamento (M1)

Interlock_System

include: globals.h
Beckhoff_IO_Addrs.h
define: N/A
variables: facade_safety_interlock // Facade_Safety_Interlock instantiation from the Facade_Safety_Interlock class
facade_communication_beckhoff // Facade_Communication_Beckhoff instantiation from the Facade_Communication_Beckhoff class

function interlock()
// Requests reading of the emergency button signal
facade_communication_beckhoff.receive_ethercat_frame()
interlockButton ←
facade_communication_beckhoff.digital_read(IN_EMERGENCY_BUTTON)

// Requests interlocking and safety service
facade_safety_interlock.interlock()

// Requests writing to enable or disable engine drivers
facade_communication_beckhoff.digital_write(OUT_ENABLE_M1, motorEnable)
facade_communication_beckhoff.digital_write(OUT_ENABLE_M2, motorEnable)
facade_communication_beckhoff.send_ethercat_frame()
end function

Facade_Interlock_System

include: NA
define: NA
variable: interlock_system // Interlock_System instantiation from the Interlock_System class

function interlock()
interlock_system.interlock()
end function

Documento de Requisitos

Sistema de Controle de Movimento (M2)

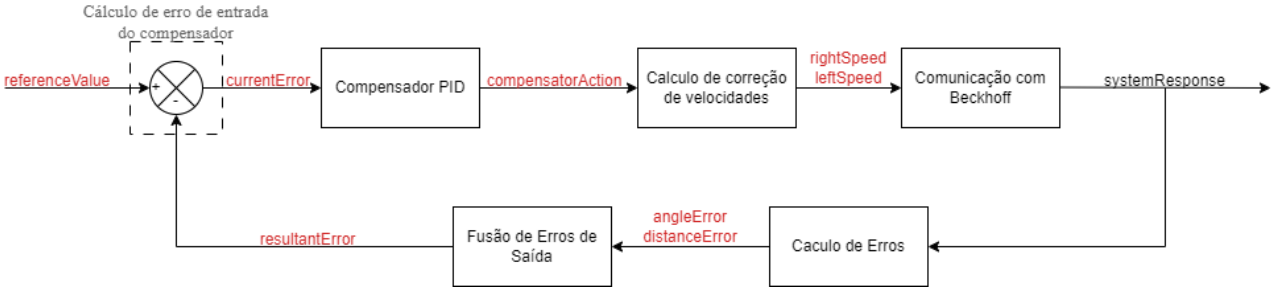
Introdução

O sistema é responsável pela malha de controle do software. Sendo esse sistema a junção dos seguintes serviços:

- Comunicação com a Beckhoff
- Compensador PID
- Correção de velocidade
- Cálculo de erros
- Cálculo de erro de entrada do compensador
- Fusão de erros de saída

A Figura 1 representa cada serviço na malha de controle. Em blocos e pontilhados são os serviços utilizados , e em vermelho são as variáveis de saída e entrada de cada serviço.

Figura M2.1: Malha de controle



Nota-se que na Figura 1 não estão dispostas todas as variáveis necessárias para cada serviço, mas sim as variáveis que representam o fluxo de execução do diagrama de blocos.

Descrição Geral

Funções do Produto

Realizar a ordem de execução da malha de controle.

Características do Usuário

Este sistema é consumido pelo Software Escalonador.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O sistema depende da correta execução do Serviço de Comunicação com a Beckhoff, Serviço compensador PID, Serviço correção de velocidade, Serviço Cálculo de erros, Serviço cálculo de erro de entrada do compensador, Serviço fusão de erros de saída.

Requisitos Específicos

Requisitos Funcionais

RF1 - Solicitar o cálculo de erro.

RF2 - Solicitar a fusão de erros de saída.

RF3 - Solicitar o cálculo de erro de entrada do compensador.

RF4 - Solicitar o cálculo do compensador.

RF5 - Solicitar o cálculo de correção de velocidades.

RF6 - Solicitar a escrita analógica na Beckhoff.

Requisitos Não Funcionais

- **Desempenho**
  - RNF1 - Apresentar um tempo de execução inferior a 30ms.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF2 - Realizar a sequência de execução sem interrupções.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Sistema de Controle de Movimento (M2)

Descrição de métodos

Sequência de ações

Para que o sistema funcione a sequência de execução deve ser realizada:

1. Solicitar o serviço cálculo de erro.
2. Solicitar o serviço fusão de erros de saída.
3. Solicitar o serviço cálculo de erro de entrada do compensador.
4. Solicitar o serviço cálculo do compensador.
5. Solicitar o serviço cálculo de correção de velocidades.
6. Solicitar a escrita analógica na Beckhoff.

Exemplo de uso e serviço

1. Quando acionado o majoritários a seguinte sequência deve ser executada:
    1. Solicitar o serviço cálculo de erro.
    2. Solicitar o serviço fusão de erros de saída.
    3. Solicitar o serviço cálculo de erro de entrada do compensador.
    4. Solicitar o serviço cálculo do compensador.
    5. Solicitar o serviço cálculo de correção de velocidades.
    6. Solicitar a escrita analógica na Beckhoff.

Referências

Histórico de alterações

-

Sistema de Controle de Movimento (M2)

Diagrama de Casos de Uso

Figura M2.2: Diagrama de Casos de Uso

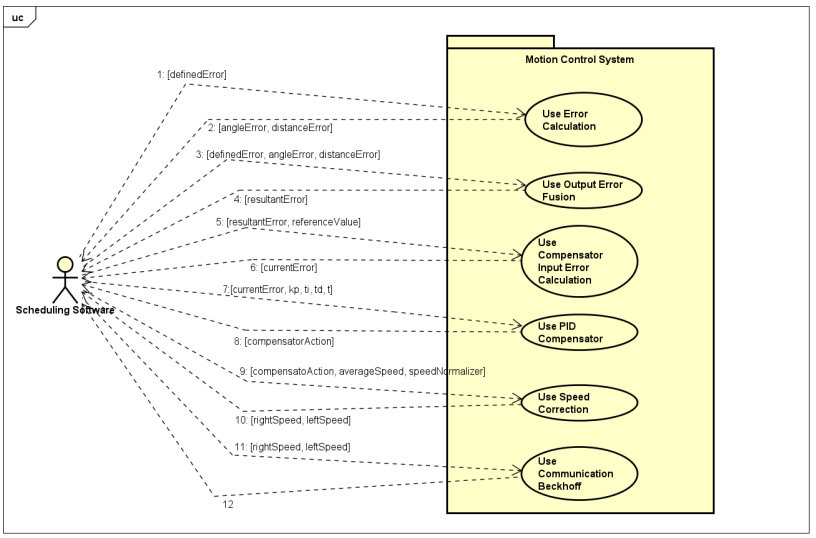


Diagrama de Atividades

Figura M2.3: Diagrama de Atividades

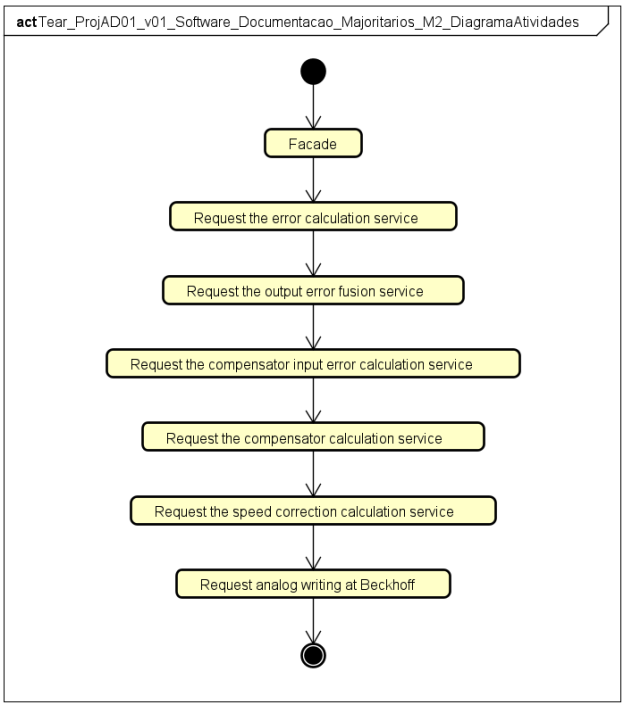


Diagrama de Sequência

Figura M2.3: Diagrama de Sequência

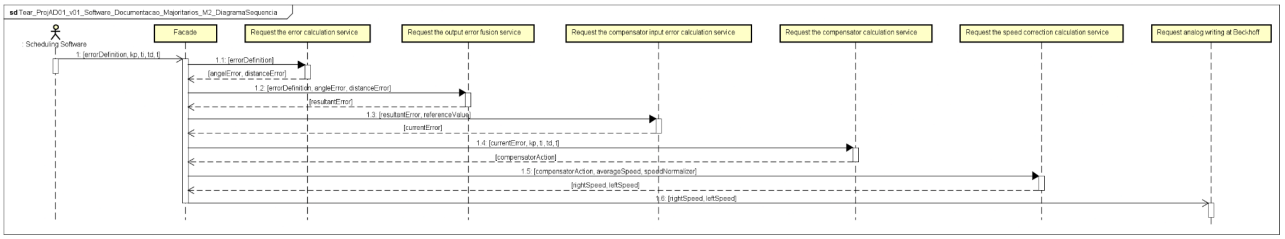
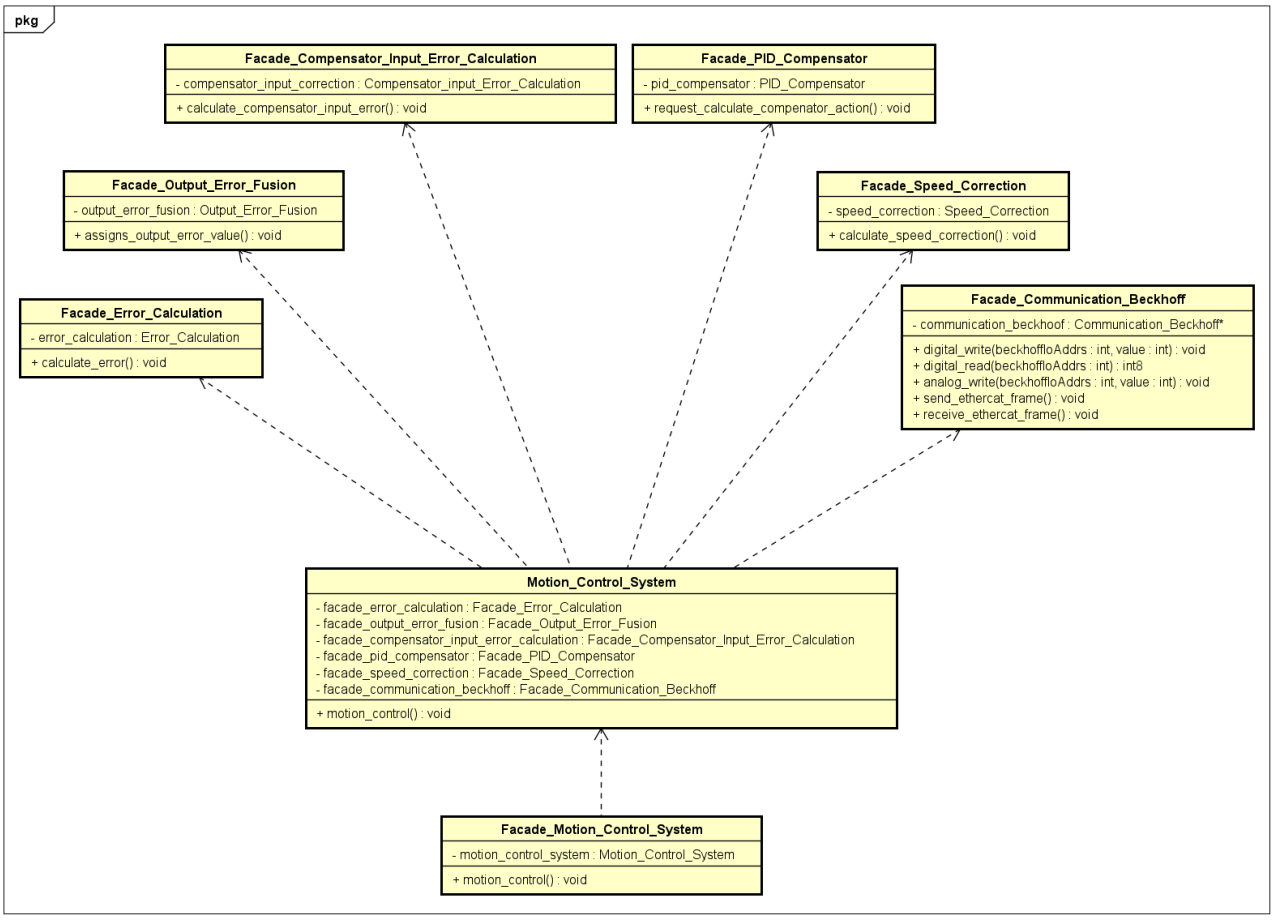


Diagrama de Classe

Figura M2.4: Diagrama de Classe



Pseudocódigo

Sistema de Controle de Movimento (M2)

Motion_Contol_System

include: Facade_Error_Calculation.h
Facade_Output_Error_Fusion.h
Facade_Compensator_Input_Error_Calculation.h
Facade_PID_Compensator.h
Facade_Speed_Correction.h
Facade_Communication_Beckhoff.h

define: N/A

variables: facade_error_calculation // Facade_Error_Calculation instantiation from the Facade_Error_Calculation class
facade_output_error_fusion // Facade_Output_Error_Fusion instantiation from the Facade_Output_Error_Fusion class
facade_compensator_input_error_calculation // Facade_Compensator_Input_Error_Calculation instantiation from the Facade_Compensator_Input_Error_Calculation class
facade_pid_compensator // Facade_PID_Compensator instantiation from the Facade_PID_Compensator class
facade_speed_correction // Facade_Speed_Correction instantiation from the Facade_Speed_Correction class
facade_communication_beckhoff // Facade_Communication_Beckhoff instantiation from the Facade_Communication_Beckhoff class

function motion_control()

```
// Request the error calculation service
facade_error_calculation.calculate_error()

// Request the output error fusion service
facade_output_error_fusion.assigns_output_error_value()

// Request the compensator input error calculation service
facade_compensator_input_error_calculation.calculate_compensator_input_error()

// Request the compensator calculation service
facade_pid_compensator.request_calculate_compensator_action()

// Request the speed correction calculation service
facade_speed_correction.calculate_speed_correction
```

```
        // Request analog writing at Beckhoff
        facade_communication_beckhoff.receive_ethercat_frame()
        facade_communication_beckhoff.analog_write(ANALOG_SPEED_M1,right
        Speed)
        facade_communication_beckhoff.analog_write(ANALOG_SPEED_M2,leftS
        peed)
        facade_communication_beckhoff.send_ethercat_frame()
end function
```

Facade_Motion_Control_System

include: NA
define: NA
variable: motion_control_system// Motion_Control_System instantiation from the Motion_Control_System class

```
function motion_control()
        motion_control_system.motion_control()
end function
```

Documento de Requisitos

Sistema de Atribuição de Dados (M3)

Introdução

O sistema é uma união dos serviços:

- Entrada Manual de Dados
- Geração de Referência
- Iniciar Variáveis Globais
- Condição Inicial de Hardware

Sendo o primeiro sistema a ser executado, o sistema é responsável por inicializar todas as variáveis globais e componentes de hardware necessários para o funcionamento dos demais sistemas. Este sistema garante que todas as dependências críticas estejam configuradas e prontas para uso antes do início da operação dos outros sistemas.

O sistema também realiza a comunicação com a beckhoff para garantir que na inicialização do software o AGV esteja parado.

Descrição Geral

Funções do Produto

Inicializar todas as condições iniciais do software e hardware.

Características do Usuário

Este sistema é consumido pelo Software Escalonador..

Restrições Gerais

Não identificado.

Suposições e Dependências

- O sistema depende da correta execução do Serviço Entrada Manual de Dados, Serviço Geração de Referência, Serviço Iniciar Variáveis Globais, Serviço Condição Inicial de Hardware e Serviço Comunicação com a Beckhoff.
- O sistema depende de um monitor.
- O sistema depende de um teclado.

Requisitos Específicos

Requisitos Funcionais

RF1 - Solicitar a condição inicial de hardware.

RF2 - Solicitar escrita da condição inicial de hardware.

RF3 - Solicitar inicialização das variáveis globais.

RF4 - Solicitar a entrada manual de dados.

RF5 - Solicitar a geração de referência.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.

- **Escalabilidade**
    - Não identificado.
- **Confiabilidade**
    - Não identificado.
- **Manutenibilidade**
    - Não identificado.

Documento de Demanda de Produção

Sistema de Atribuição de Dados (M3)

Descrição de métodos

Sequência de ações

Para que o sistema funcione a sequência de execução deve ser realizada:

1. Solicitar o serviço Condição Inicial de Hardware
2. Solicitar a escrita digital para desabilitar os drivers dos motores
3. Solicitar a escrita analogica para deixar em zero a velocidade dos motores
4. Solicitar o Serviço Inicialização das variáveis globais
5. Solicitar o Serviço Entrada Manual de Dados
6. Solicitar o Serviço Geração de Referência

Exemplo de uso e serviço

1. Quando executado o sistema e os seguintes valores determinados para o serviço de entrada manual de dados:
   - **kp** = 1;
   - **ti** = 1;
   - **td** = 1;
   - **t** = 1;
   - **errorDefinition =** 0;
   - **averageSpeed** = 10;
   - **speedNormalizer** = 5;
   - **definedReference** = 0;
2. Os drivers dos motores devem estar desativados.
3. As saídas analógicas dos motores devem estar em zero.
4. Os seguintes valores devem estar armazenados nas variáveis:
   - **kp** = 1;
   - **ti** = 1;
   - **td** = 1;
   - **t** = 1;
   - **errorDefinition =** 0;
   - **averageSpeed** = 10;
   - **speedNormalizer** = 5;
   - **definedReference** = 0;
   - **referenceValue** = 0;
   - **interlockButton** = 0;
   - **currentError** = 0;
   - **compensatorAction** = 0;
   - **angleError** = 0;
   - **distanceError** = 0;
   - **resultantError** = 0;
   - **rightSpeed** = 0
   - **leftSpeed** = 0;
   - **motorEnable** = 0;

Referências

Histórico de alterações

Sistema de Atribuição de Dados (M3)

Diagrama de Casos de Uso

Figura M3.1: Diagrama de Classe

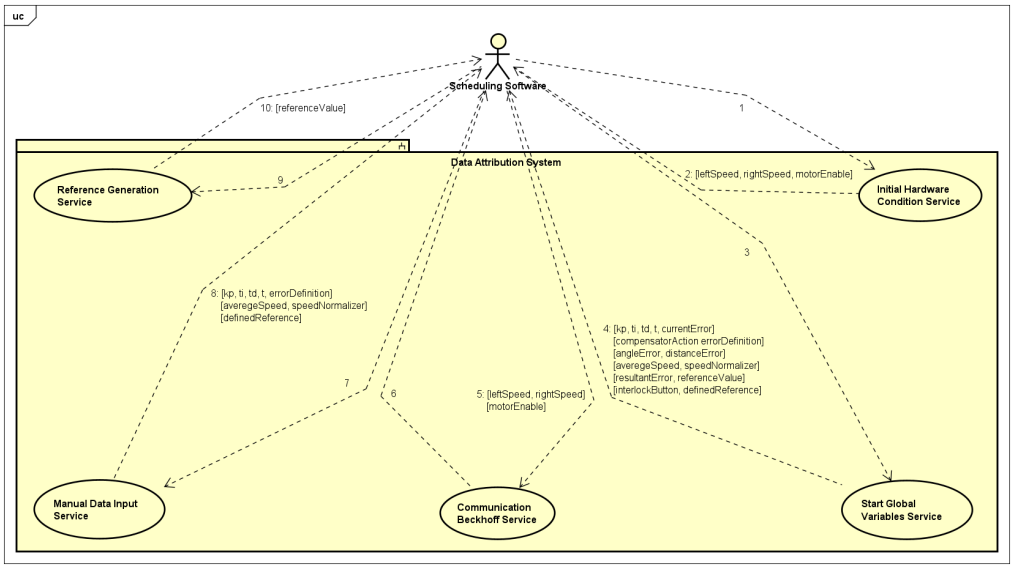


Diagrama de Atividades

Figura M3.2: Diagrama de Atividades

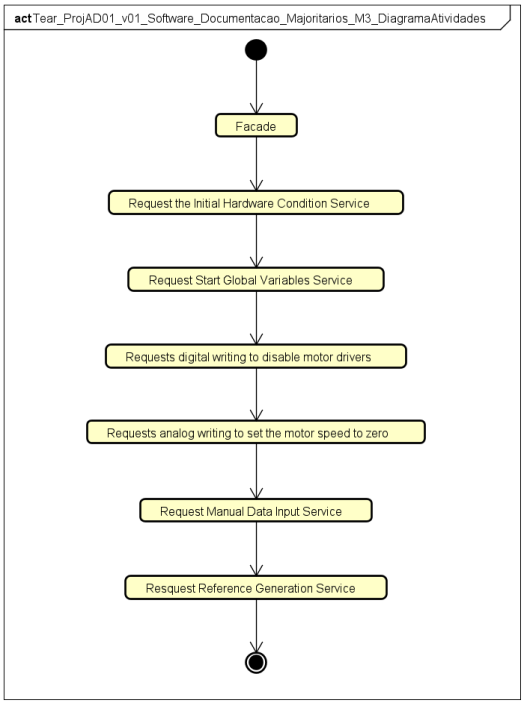


Diagrama de Sequência

Figura M3.3: Diagrama de Sequência

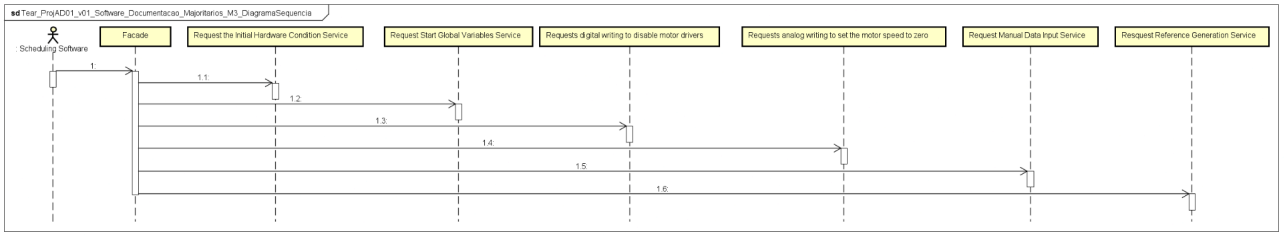
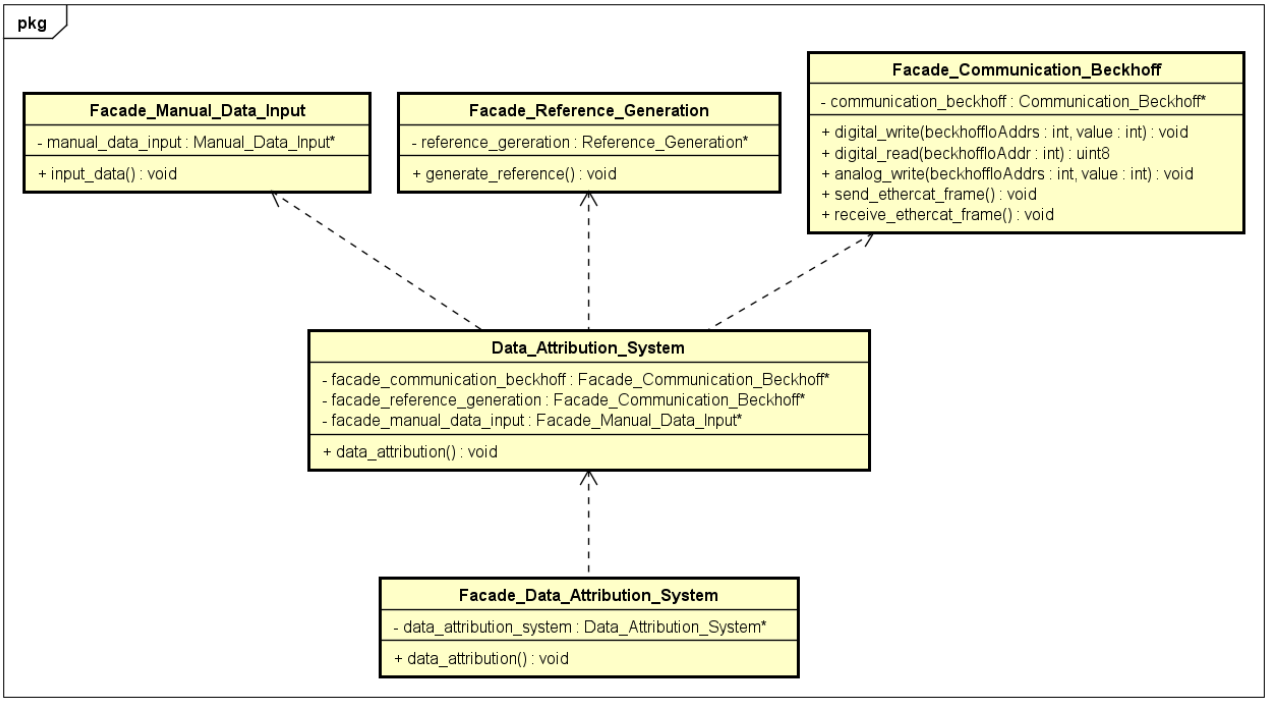


Diagrama de Classe

Figura M3.4: Diagrama de Classe



Pseudocódigo

Sistema de Atribuição de Dados (M3)

Data_Attribution_System

include: Beckhoff_IO_Addrs.h

```
// Request the Initial Hardware Condition Service
Facade_Initial_Hardware_Condition.h
// Request Start Global Variables Service
Facade_Start_Global_Variables.h
Facade_Communication_Beckhoff.h
Facade_Manual_Data_Input.h
Facade_Reference_Generation.h
```

define: N/A

variables: facade_communication_beckhoff // Facade_Communication_Beckhoff instantiation from the Facade_Communication_Beckhoff class

facade_manual_data_input // Facade_Manual_Data_Input instantiation from the Facade_Manual_Data_Input class

facade_reference_generation // Facade_Reference_Generation instantiation from the Facade_Reference_Generation class

function data_attribution()

```
// Requests digital writing to disable motor drivers
facade_communication_beckhoff.receive_ethercat_frame()
facade_communication_beckhoff.digital_write(OUT_ENABLE_M1,
motorEnable)
facade_communication_beckhoff.digital_write(OUT_ENABLE_M2,
motorEnable)

// Requests analog writing to set the motor speed to zero
facade_communication_beckhoff.analog_write(ANALOG_SPEED_M1,right
Speed)
facade_communication_beckhoff.analog_write(ANALOG_SPEED_M2,leftS
peed)
facade_communication_beckhoff.send_ethercat_frame()

// Request Manual Data Input Service
facade_manual_data_input.input_data()

// Request Reference Generation Service
facade_reference_generation.generate_reference()
```

end function

Facade_Data_Attribution_System

include: NA
define: NA
variable: data_attribution_system // Data_Atrribution_System instantiation from the Data_Atrribution_System class

function data_attribution()
    data_attribution_system.data_atrribution()
end function

Documento de Requisitos

Serviço intertravamento de segurança (S1)

Introdução

O serviço deve desativar os drivers dos motores quando a botoeira externa for acionada.

Descrição Geral

Funções do Produto

Realizar o intertravamento do AGV.

Características do Usuário

Este serviço é consumido pelo Sistema de Intertravamento.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que a variável global **interlockButton** já esteja criada e com valor atribuído.
- O serviço presume que a variável **motorEnable** já esteja criada.

Requisitos Específicos

Requisitos Funcionais

RF1 - Desativar os drivers dos motores.

RF2 - O serviço deve modificar a variável global **motorEnable**.

RF3 - O serviço deve ler a variável global **interlockButton**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - Não identificado.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Serviço intertravamento de segurança (S1)

Descrição de métodos

Desativar o driver

Utilizar a variável global **interlockButton**.

- Quando a variável **interlockButton** for igual a 1, os drivers deverão ser desativados, então a variável **motorEnable** recebe o valor 1.

- Quando a variável **interlockButton** for igual a 0 os drivers deverão ser ativados, então a variável **motorEnable** recebe o valor 0.

Exemplo de uso e serviço

1. Com os valores de **interlockButton** = 0;
2. O resultado retornado deve ser:
   - **motorEnable** = 0;

Referências

Histórico de alterações

05/01/25

- Criação do documento

Diagramas

Serviço intertravamento de segurança (S1)

Diagrama de Casos de Uso

Figura S1.1: Diagrama de Classe

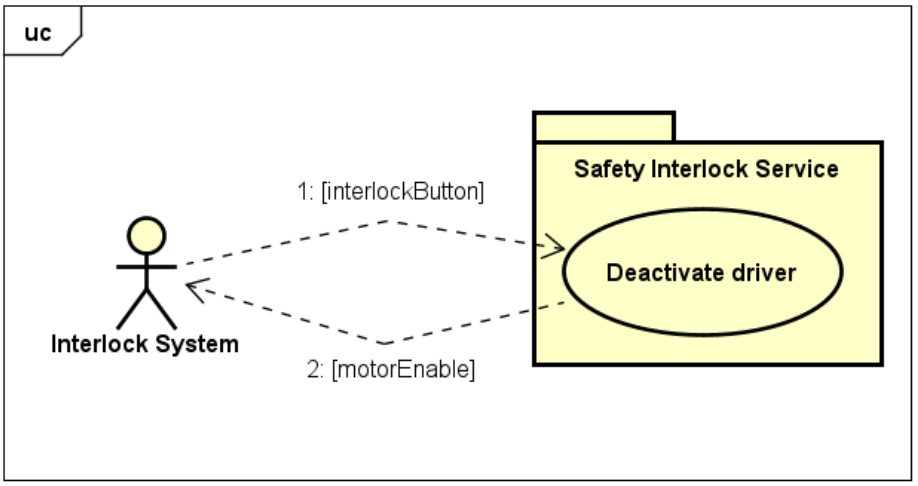


Diagrama de Atividades

Figura S1.2: Diagrama de Atividades

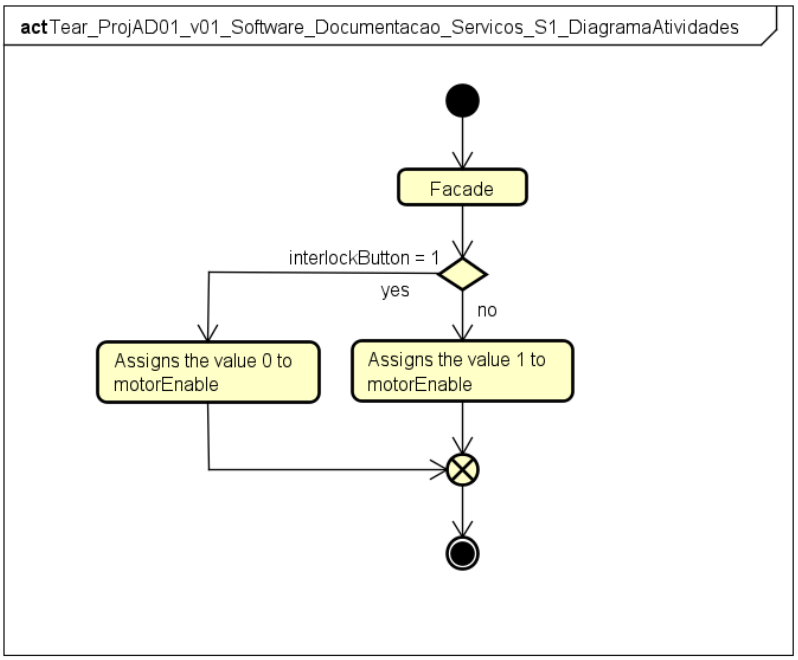


Diagrama de Sequência

Figura S1.3: Diagrama de Sequência

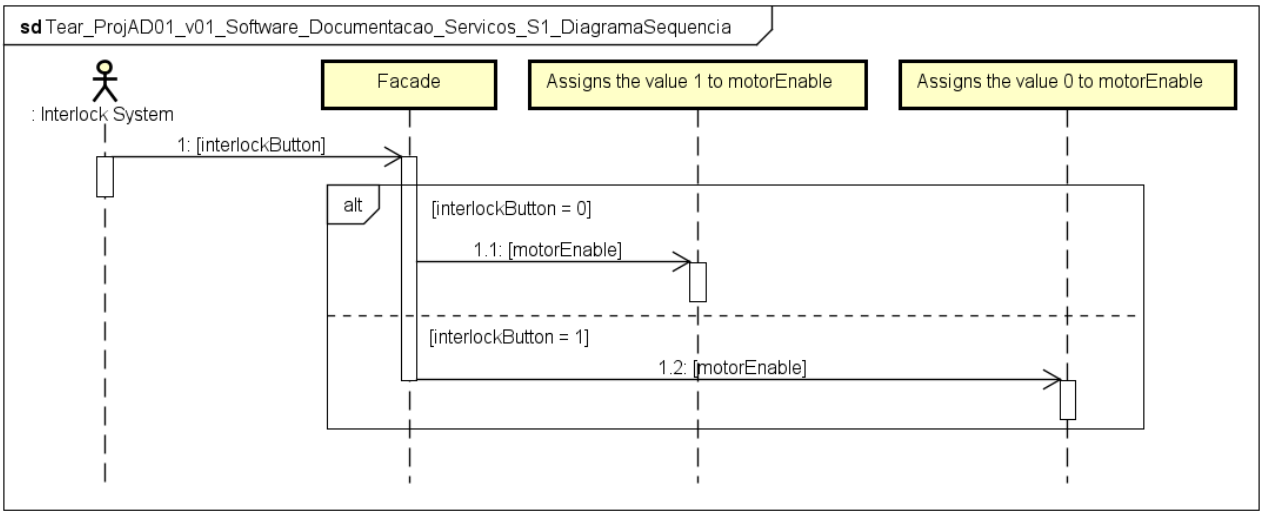
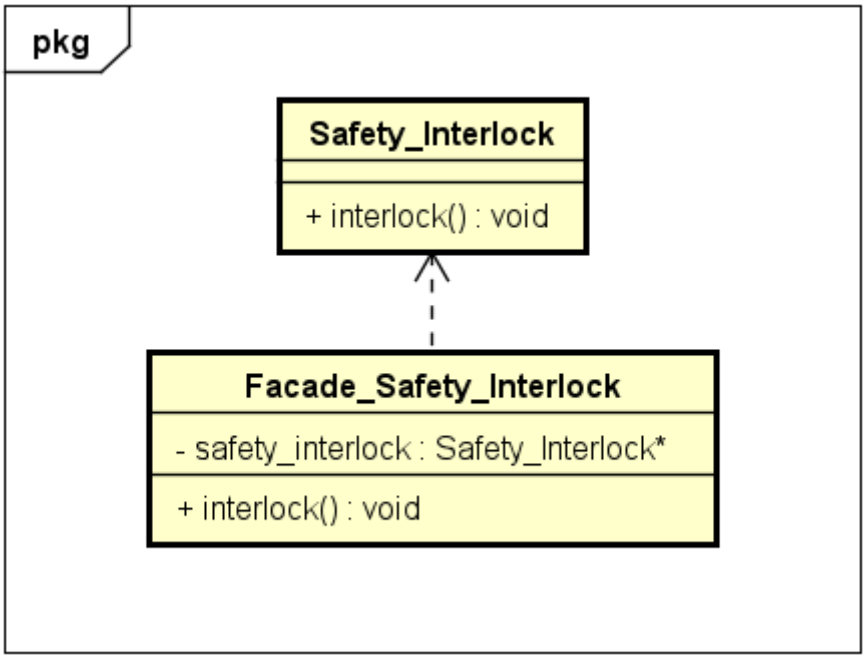


Diagrama de Classe

Figura S1.2: Diagrama de Classe



Pseudocódigo

Serviço intertravamento de segurança (S1)

Safety_Interlock

include: globals.h
Initial_Hardware_Condition.h
define: N/A
variables: N/A

method interlock()
if interlockButton = 1
// Assigns the value 1 to motorEnable
motorEnable ← 1
else
// Assigns the value 0 to motorEnable
motorEnable ← 0
end method

Facade_Safety_Interlock

include: NA
define: NA
variable: safety_interlock // Safety_Interlock instantiation from the Safety_Interlock class

method interlock()
safety_interlock.interlock();
end method

Documento de Requisitos

Serviço comunicação com a Beckhoff (S2)

Introdução

O serviço fornece uma comunicação da CPU e as I/Os da beckhoff.

Descrição Geral

Funções do Produto

Fornecer uma interface de comunicação EtherCAT, utilizando a biblioteca SOEM .

Características do Usuário

Este serviço é consumido pelo Sistema de Intertravamento, Sistema de Controle de Movimento e Sistema de Atribuição de Dados.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende da biblioteca SOEM (Simple Open EtherCAT Master Library).
- O serviço depende de versões do CMake igual ou superior a 3.9.
- O serviço depende do kernel do linux igual ou superior a 2.6.
- O serviço depende de um módulo Beckhoff que utilize a comunicação EtherCAT.

Requisitos Específicos

Requisitos Funcionais

RF1 - Leitura Digital da Beckhoff.

RF2 - Escrita Digital da Beckhoff.

RF3 - Escrita Analógica da Beckhoff.

RF4 - Inicialização da comunicação.

Requisitos Não Funcionais

- **Desempenho**
  - Frequência de 1kHz.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - Frequência constante.
- **Manutenibilidade**
  - Aplicar Real-Time.

Documento de Demanda de Produção

Serviço comunicação com a Beckhoff (S2)

Descrição de métodos

Implementação

O serviço Communication_Beckhoff já está implementado, por isso a documentação será somente em relação à fachada, entradas e saídas do serviço, não apresentando a lógica interna do serviço.

Comunicação

Será utilizado a biblioteca SOEM.

Endereço das Entradas e Saídas

Para que o sistema saiba os endereços de entradas e saídas será utilizado um arquivo denominado **Beckhoff_IO_Addrs.h** contendo o endereço de cada módulo utilizado no AGV AD01.

Exemplo de uso e serviço

- Ao solicitar uma saída digital para a saída da câmera da lâmpada.
- A lâmpada da câmera deve ascender.

Referências

Documentação SOEM.

Histórico de alterações

Documento de Demanda de Produção

Serviço comunicação com a Beckhoff (S2)

Diagrama de Casos de Uso

Figura S2.1: Diagrama de Classe

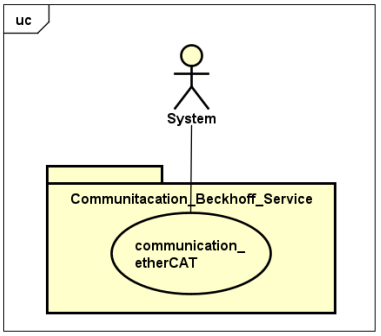


Diagrama de Atividades

Figura S2.2: Diagrama de Atividades

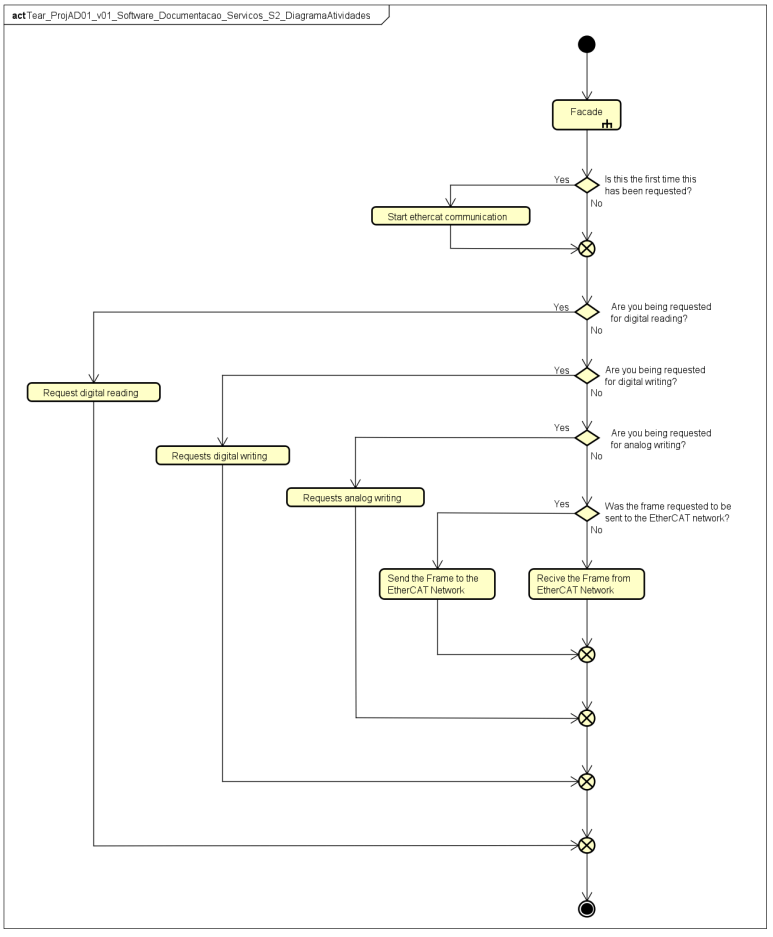


Diagrama de Sequência

Figura S2.3: Diagrama de Sequência

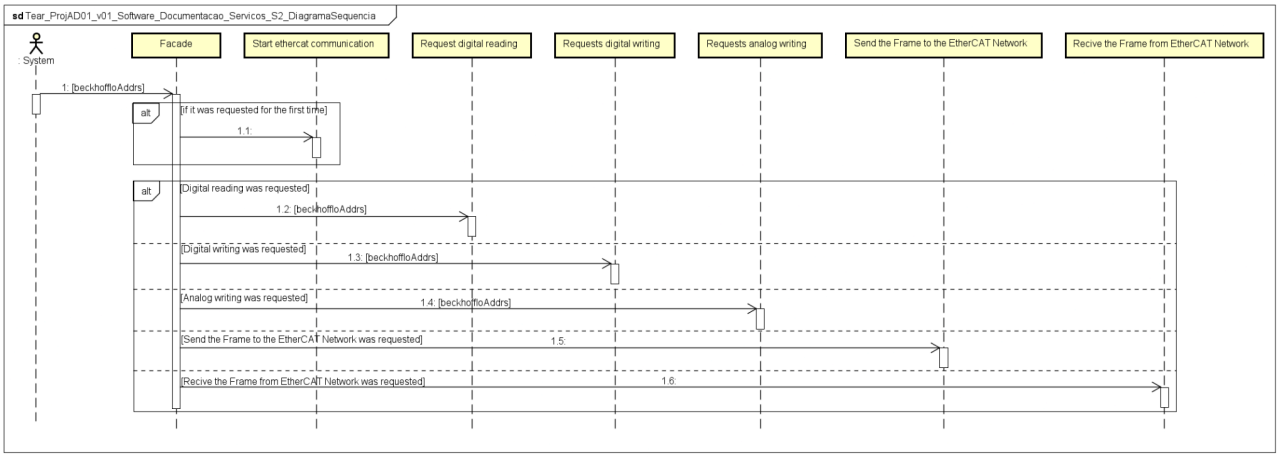
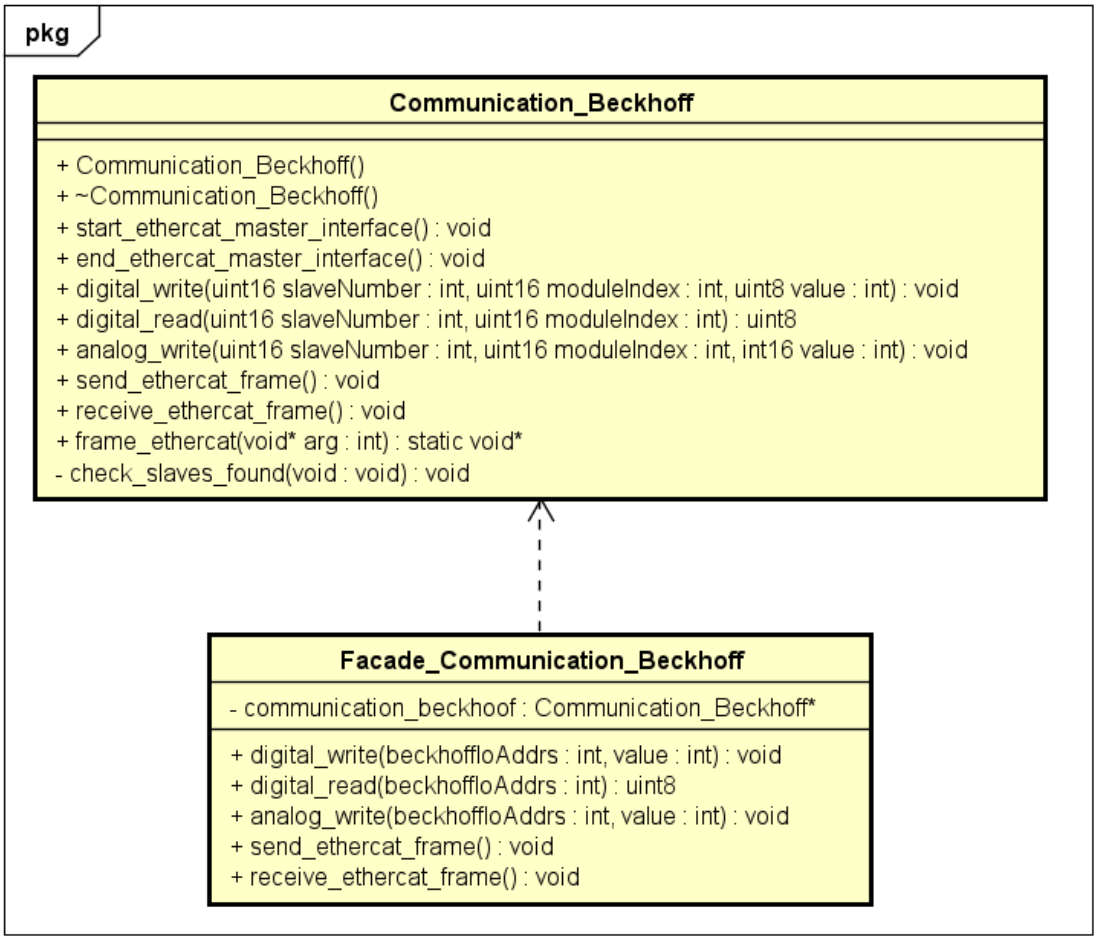


Diagrama de Classe

Figura S2.4: Diagrama de Classe



Pseudocódigo

Serviço comunicação com a Beckhoff (S2)

Facade_Communication_Beckhoff

include: NA
define: NA
variable: communication_beckhoff // Communication_Beckhoff instantiation from the Communication_Beckhoff class

method digital_write(beckhoffIoAddrs, value)
    communication_beckhoff.digital_write(EL2809, beckhoffIoAddrs, value)
end method

method digital_read(beckhoffIoAddrs)
    communication_beckhoff.digital_read(EL1809, beckhoffIoAddrs)
end method

method analog_write(beckhoffIoAddrs, value)
    communication_beckhoff.analog_write(EL4002, beckhoffIoAddrs, value)
end method

Documento de Requisitos

Serviço de Compensador PID (S3)

Descrição Geral

Funções do Produto

A função desse produto é realizar o cálculo do compensador PID.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende das seguintes variáveis globais já criadas com seus valores atribuídos: **kp**, **ti**, **td**, **t**, **currentError**.
- O serviço depende que a variável global **compensatorAction** já esteja criada.
- A variável global **kp** não pode ser igual a zero.

Requisitos Específicos

Requisitos Funcionais

RF1 - Utilizar as variáveis globais (**kp**, **ti**, **td**, **t**, **currentError**, **compensatorAction**).

RF2 - Calcular a ação de controle usando a fórmula PID com os parâmetros fornecidos globalmente. A fórmula do PID deve ser implementada conforme a equação 1.

$$PID = Kp \cdot E + Ki \cdot \int E + Kd \cdot \frac{dE}{dt} \quad \text{[equação 1]}$$

Onde:

E é o erro atual.

$\int E$ é a soma dos erros passados (termo integral).

dE/dt é a taxa de variação do erro (termo derivativo).

RF3 - O valor calculado pela ação de controle (**compensatorAction**) deve ser saturado com os limites de -100 e +100.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender sistemas de primeira ordem.
  - RNF2 - Atender sistemas subamortecidos e superamortecidos.
  - RNF3 - Atender sistema crítico.

- **Manutenibilidade**
    - RNF4 - A manutenção como incremento de função pode-se realizar a validação em entrada em rampas.

Documento de Demanda de Produção

Serviço Compensador PID (S3)

Descrição de métodos

Cálculo dos valores de ganho ki e kd

Usando o período (**t**), termo integral (**ti**), termo derivativo (**td**) e ganho proporcional (**kp**) não nulo, são calculados os termos **ki** e **kd**, respectivamente, conforme as equações 1 e 2.

$$ki = kp * (t / ti)$$ [equação 1]

$$kd = kp * (td / t)$$ [equação 2]

Ao calcular **ki** e **kd**, é possível calcular os coeficientes, apresentados na seção seguinte.

Cálculo dos valores de coeficientes

Os coeficientes **a0**, **a1** e **a2** são calculados baseados nos ganhos proporcionais (**kp**), integrais (**ki**) e derivativos (**kd**) conforme as equações 4, 5 e 6.

$$a0 = kp + ki + kd$$ [equação 3]

$$a1 = -(kp + 2 * kd)$$ [equação 4]

$$a2 = kd$$ [equação 5]

Após todos os cálculos serem feitos, o cálculo do valor de ação de compensador (**compensatorAction**) é feito, por meio dos cálculos apresentados na seção seguinte.

Cálculo da ação de compensador

A saída da função (**compensatorAction**) é definida pela seguinte equação:

compensatorAction = compensatorActionPrevious + a0 * currentError + a1 * previousError + a2 * previousPastError [equação 6]

Onde:

- **compensatorActionPrevious** é a ação anterior do compensador;
- **currentError** é o erro atual;
- **previousError** é o erro anterior;
- **previousPastError** é o erro passado anterior.

Para chegar à equação apresentada, na primeira execução, todos os valores de erro e a ação do controle começam em zero.

Armazenamento de valores de erro

No final de cada cálculo de valor de compensador, o serviço armazena os valores de **compensatorActionPrevious**, **previousError** e **previousPastError** para uso nas próximas execuções em que o compensador PID for chamado com um novo **currentError**.

Exemplo de uso e serviço

1. Para valores **kp** = 1, **ti** = 0,1, **td** = 0,01, **t** = 0,1 sendo a primeira execução, portanto, os erros anteriores são iguais a zero, com uma entrada no **currentError** = 5;
2. Ao calcular os valores de **a0**, **a1** e **a2**, **compensatorAction** deve ser igual a 10,5.
3. Em seguida, o serviço faz as seguintes atribuições:
    a. **compensatorActionPrevious** = **compensatorAction**;
    b. **previousPastError** = **previousError**;
    c. **previousError** = **currentError**.

Referências

Tear_DrWallacePNR_RelatorioTecnico_Relatorio_Tecnico_20220805

Histórico de alterações

14/11/24:

- Reestruturação completa do documento para englobar uso geral do serviço, e não apenas para AGVs.
- Adição de métodos de obtenção e inserção de valores dos ganhos KP e KD, em falta desde a última alteração do documento.

23/11/24:

- Alteração completa nos métodos, removendo os métodos computacionais e descrevendo o que é método e como funciona PID que será utilizado pelo serviço.

25/11/24:

- Escrita mais direta da documentação.
- Adição da fórmula para explicar o algoritmo.

26/11/24

- Adição do cálculo dos termos.
- Adição das variáveis globais t, ti e td.

30/01/25:

- Divisão dos métodos para melhor definição no diagrama de casos de uso.

27/02/25:

- Reorganização dos métodos para compreensão do serviço de maneira sequencial.

Diagramas

Serviço Compensador PID (S3)

Diagrama de Casos de Uso

Figura S3.1: Diagrama de Casos de Uso

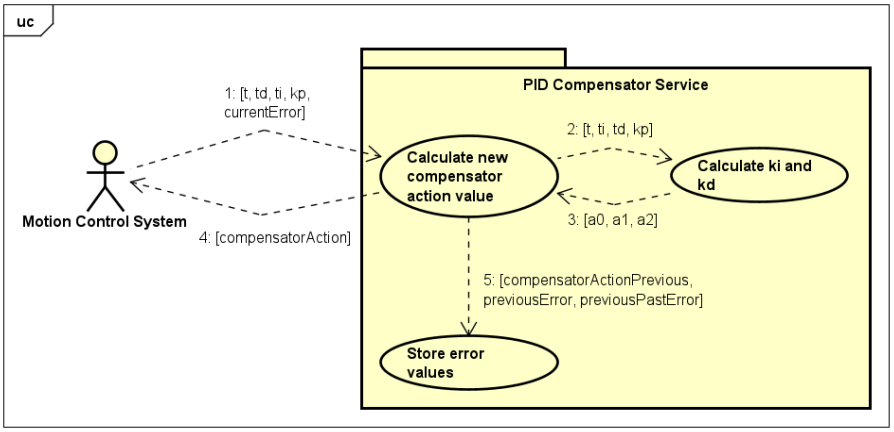


Diagrama de Atividades

Figura S3.2: Diagrama de Atividades

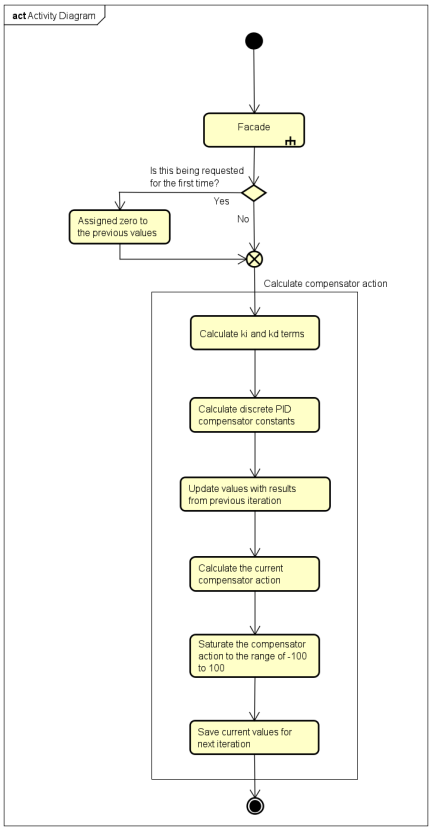


Diagrama de Sequência

Figura S3.3: Diagrama de Sequência

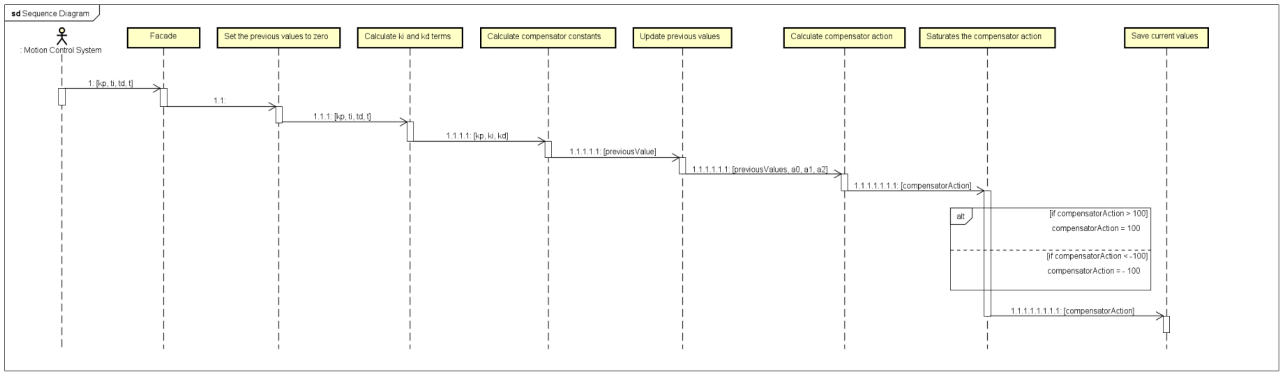
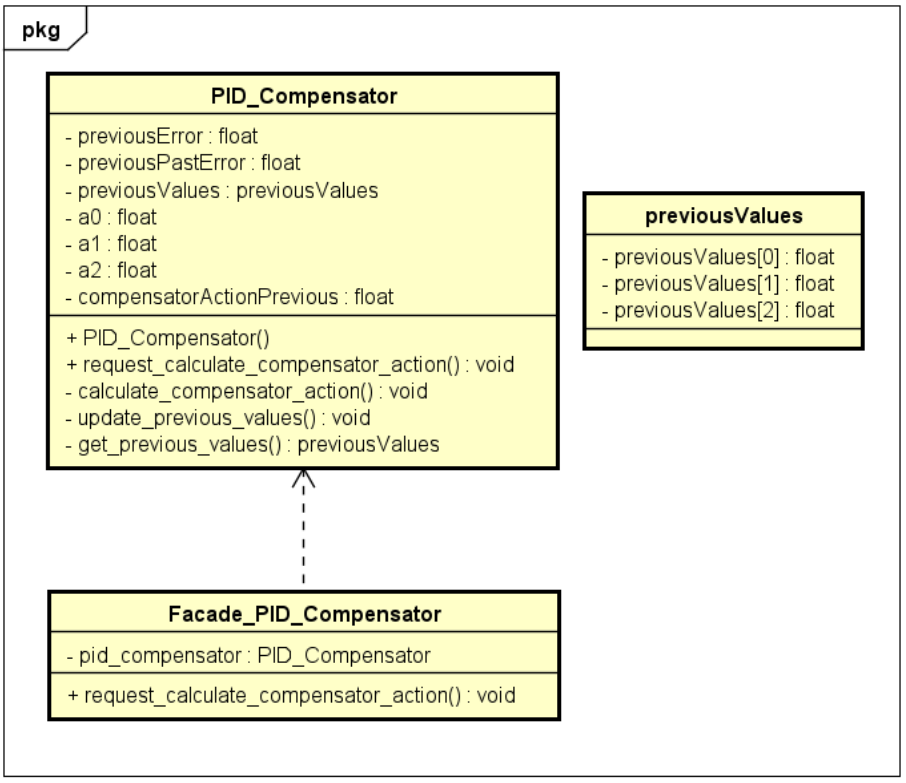


Diagrama de Classe

Figura S3.4: Diagrama de Classe



Pseudocódigo

Serviço Compensador PID (S3)

Pid_Compensator_Service

include: iostream
globals.h
define: NA
variables: previousError // float, error at previous instant
previousPastError // float, error measured two instants ago
previousValues[] // float, vector with error values and control action from past iterations
ki, kd // float, PID gains
a0, a1, a2 // float, Discrete PID controller constants
compensatorAction // float, represents, the control action for movement correction
compensatorActionPrevious // float, value of the control action at the previous instant

method PID_Compensator()
// Assigned zero to the previous values
vectorPreviousValues[0] ← 0
vectorPreviousValues[1] ← 0
vectorPreviousValues[2] ← 0
end method

method request_calculate_compensator_action()
calculate_compensator_action()
end method

// Calculate compensator action
method calculate_compensator_action()

// Calculate ki and kd terms
ki ← kp * (t / ti)
kd ← kp * (td / t)

// Calculate discrete PID compensator constants
a0 ← kp + ki + kd
a1 ← -(kp + 2*kd)
a2 ← kd

// Update values with results from previous iteration
previousValues[] ← getPreviousValues()

```
        previousError ← previousValues[0]
        previousPastError ← previousValues[1]
        compensatorActionPrevious ← previousValues[2]

        // Calculate the current compensator action
        compensatorAction ← compensatorActionPrevious + a0 x currentError +
        a1 x previousError + a2 x previousPastError

        // Saturate the compensator action to the range of -100 to 100
        if compensatorAction > 100
                compensatorAction ← +100
        else compensatorAction < -100
                compensatorAction ← -100
        // Save current values for next iteration
        update_previous_values()
        end method

method update_previous_values()
        vectorPreviousValues[0] ← currentError
        vectorPreviousValues[1] ← previousError
        vectorPreviousValues[2] ← compensatorAction
end method

method get_previous_values()
        return vectorPreviousValues
end method
```

Facade_Pid_Compensator_Service

include: NA
define: NA
variable: pid_compensator // Pid_Compensator instantiation from the Pid_Compensator class

```
method compensator_action()
        pid_compensator.request_calculate_compensator_action()
end method
```

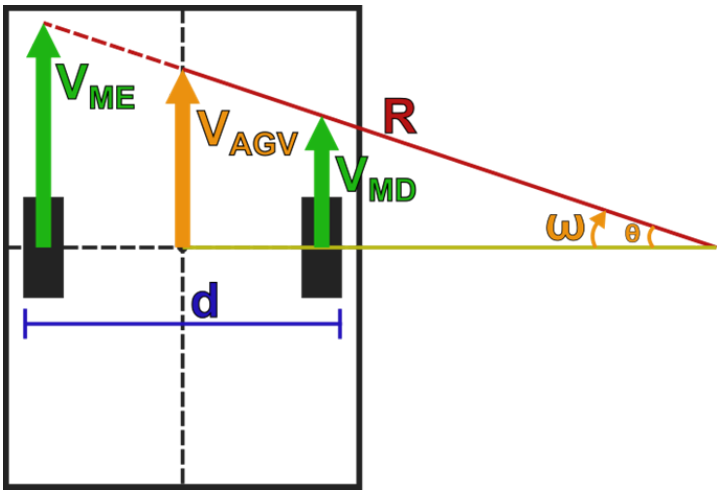
Documento de Requisitos

Serviço Correção de Velocidade (S4)

Introdução

O serviço deve calcular a velocidade dos motores do AGV com base na saída do compensador PID de forma que com a variação da velocidade de cada roda, o AGV permaneça com uma velocidade constante. O AGV possui tração diferencial de duas rodas; então, para determinar e entender seu comportamento, pode-se utilizar os seguintes métodos.

Figura S4.1: Vetores e referências associados às velocidades dos motores.



Como apresentado na Figura 1, o módulo de velocidade do AGV, representado pelo vetor de cor laranja $\mathbf{V_{AGV}}$, é a média das velocidades das rodas do AGV, representadas pelos vetores de motor esquerdo ($\mathbf{V_{ME}}$) e motor direito ($\mathbf{V_{MD}}$), representados em verde. Assim, o cálculo que descreve o módulo de velocidade do AGV pode ser descrito como:

$$V_{AGV} = \frac{V_{MD} + V_{ME}}{2}. \qquad \text{[equação 1]}$$

Quando as velocidades dos motores são diferentes, como mostra na Figura 1, o AGV faz uma curva para o lado em que a velocidade está menor. O raio dessa curva, representado na linha vermelha com a letra **R**, pode ser calculado com base na velocidade angular, representado por **ω** em laranja e o módulo da velocidade do AGV ($\mathbf{V_{AGV}}$).

$$R = \frac{V_{AGV}}{\omega}. \qquad \text{[equação 2]}$$

Para encontrar a velocidade angular, utiliza-se a as velocidades dos motores do AGV e a distância entre as duas rodas, representado pela letra **d** na linha azul.

$$\omega = \frac{V_{MD} - V_{ME}}{d}. \qquad \text{[equação 3]}$$

Finalmente, para encontrar o raio (**R**), pode-se utilizar a seguinte equação final:

$$R = \frac{V_{AGV}}{\omega} = \frac{\frac{V_{MD}+V_{ME}}{2}}{\frac{V_{MD}-V_{ME}}{d}} \quad \therefore \quad R = \frac{d}{2} \cdot \frac{V_{MD}+V_{ME}}{V_{MD}-V_{ME}}. \qquad \text{[equação 4]}$$

Outro cálculo que pode ser feito é o ângulo da curva que o AGV fez, representado por **θ** em laranja. Este ângulo é determinado de acordo com a velocidade angular e o tempo em que o AGV está realizando a curva.

$$\theta = \omega \cdot t \quad \therefore \quad \theta = \frac{V_{MD}-V_{ME}}{d} \cdot t\,. \qquad \text{[equação 5]}$$

Usando estas 5 equações apresentadas, é possível calcular o comportamento do AGV em relação às velocidades dos motores.

Descrição Geral

Funções do Produto

Cálculo da velocidade de cada motor do AGV.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que as variáveis globais **rightSpeed** e **leftSpeed** já estejam criadas.
- O serviço presume que as variáveis **compensatorAction**, **averageSpeed** e **speedNormalizer** estejam com valor atribuído.

Requisitos Específicos

Requisitos Funcionais

RF1 - Calcular as velocidades dos motores, mantendo o módulo da velocidade constante.

RF2 - As velocidades devem ser normalizadas conforme o valor apresentado na variável **speedNormalizer.**

RF3 - O serviço deve modificar as variáveis globais **rightSpeed** e **leftSpeed**.

RF4 - O serviço deve ler a variável global **compensatorAction**, **averageSpeed** e **speedNormalizer** .

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender entradas do **compensatorAction** positivas e negativas.

- **Manutenibilidade**
    - Não identificado.

Documento de Demanda de Produção

Serviço Correção de Velocidade (S4)

Descrição de métodos

Cálculo das velocidades dos motores

O cálculo da velocidade de cada motor (V_{ME} = velocidade do motor esquerdo, ou **leftSpeed**; V_{MD} = velocidade do motor direito, ou **rightSpeed**) é feito com as seguintes equações.

$$V_{ME} = V_{AGV} - V_{Corrigida} \quad \text{[equação 1]}$$

$$V_{MD} = V_{AGV} + V_{Corrigida} \quad \text{[equação 2]}$$

Onde:

- $\mathbf{V_{AGV}}$ (**averageSpeed**) é a velocidade desejada do AGV.

Cálculo da velocidade corrigida

Para encontrar o valor de $\mathbf{V_{Corrigida}}$ (**correctedSpeed**), deve ser feita uma normalização entre a saída do compensador PID, e o parâmetro de normalização **speedNormalizer**. Para isso, a seguinte equação é usada.

$$V_{Corrigida} = \frac{normalização}{100} \cdot \frac{Saída\ do\ compensador}{100} \cdot Velocidade\ Média \quad \text{[equação 3]}$$

Onde:

- **Saída do compensador** é o valor a ser normalizado (**compensatorAction**).

Exemplo de uso e serviço

1. O cálculo inicia com **compensatorAction** = -15, **averageSpeed** = 50 e **speedNormalizer** = 20.

2. Os seguintes cálculos serão realizados.

$$V_{Corrigida} = \frac{20}{100} \cdot \frac{-15}{100} \cdot 50 \ \therefore\ V_{Corrigida} = -1,5 \quad \text{[equação 4]}$$

$$V_{ME} = 50 - (-1,5) \ \therefore\ 51,5 \quad \text{[equação 5]}$$

$$V_{MD} = 50 + (-1,5) \ \therefore\ V_{MD} = 48,5 \quad \text{[equação 6]}$$

3. Os resultados retornados devem ser:
   - **leftSpeed** = 51,5;
   - **rightSpeed** = 48,5.

Referências

Histórico de alterações

17/12/24:

- Criação do documento.

30/01/25:

- Divisão dos métodos para melhor definição no diagrama de casos de uso.

01/09/25:

- Adição da velocidade média no cálculo da velocidade corrigida.

Serviço Correção de Velocidade (S4)

Diagrama de Casos de Uso

Figura S4.2: Diagrama de Casos de Uso

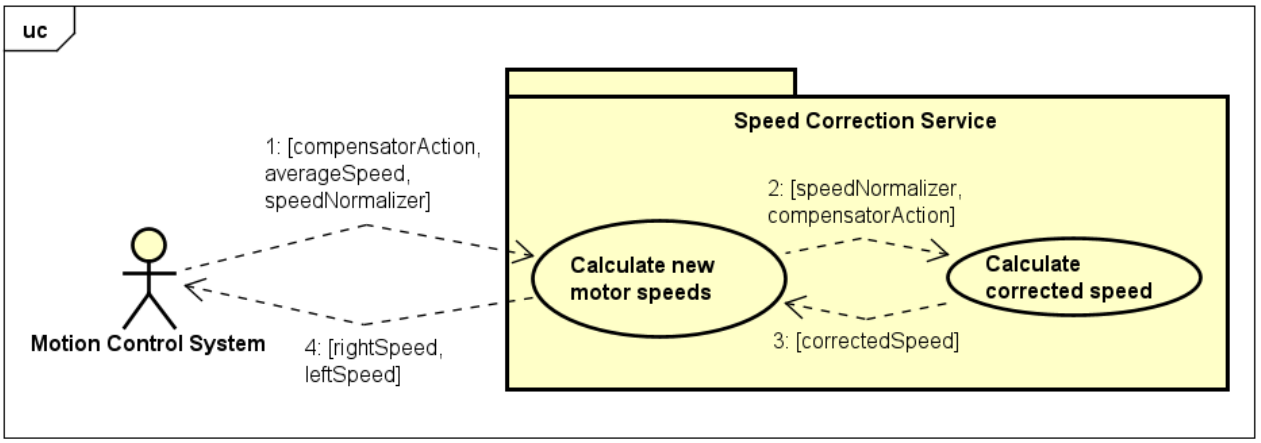


Diagrama de Atividades

Figura S4.3: Diagrama de Atividades

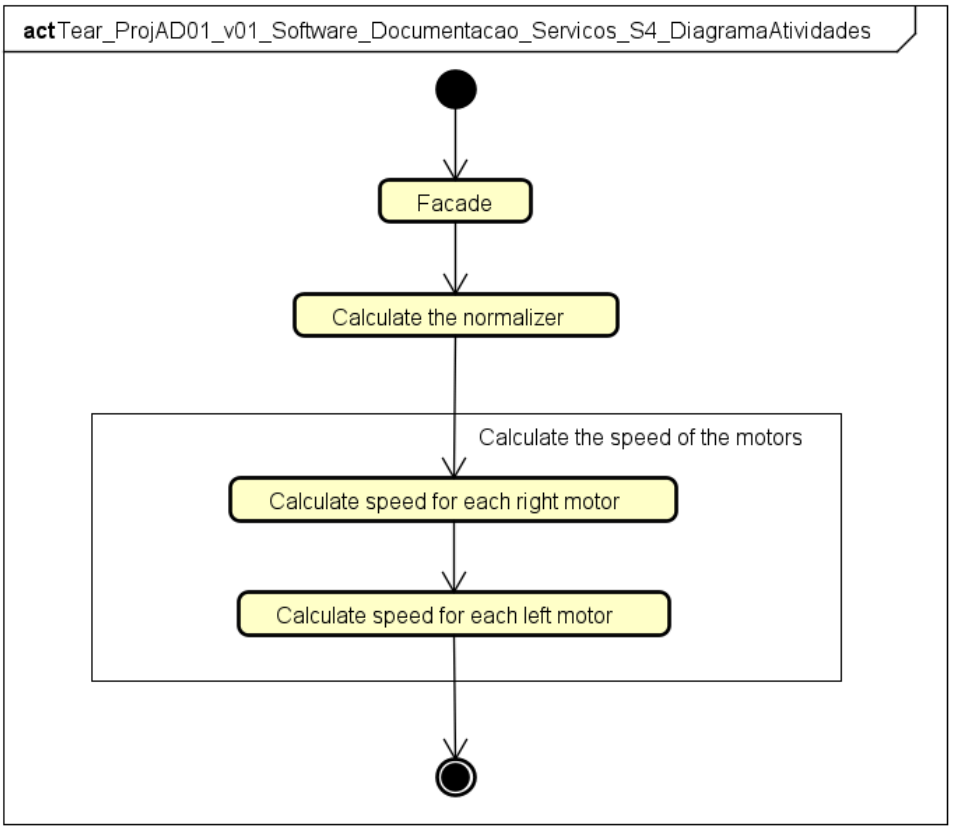


Diagrama de Sequência

Figura S4.4: Diagrama de Sequência

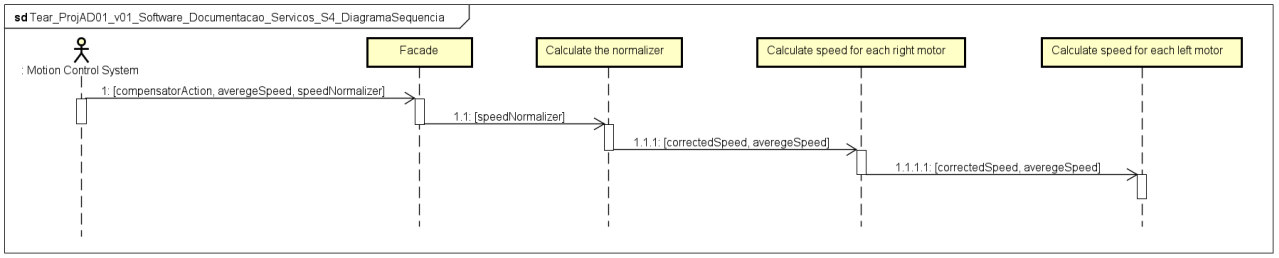
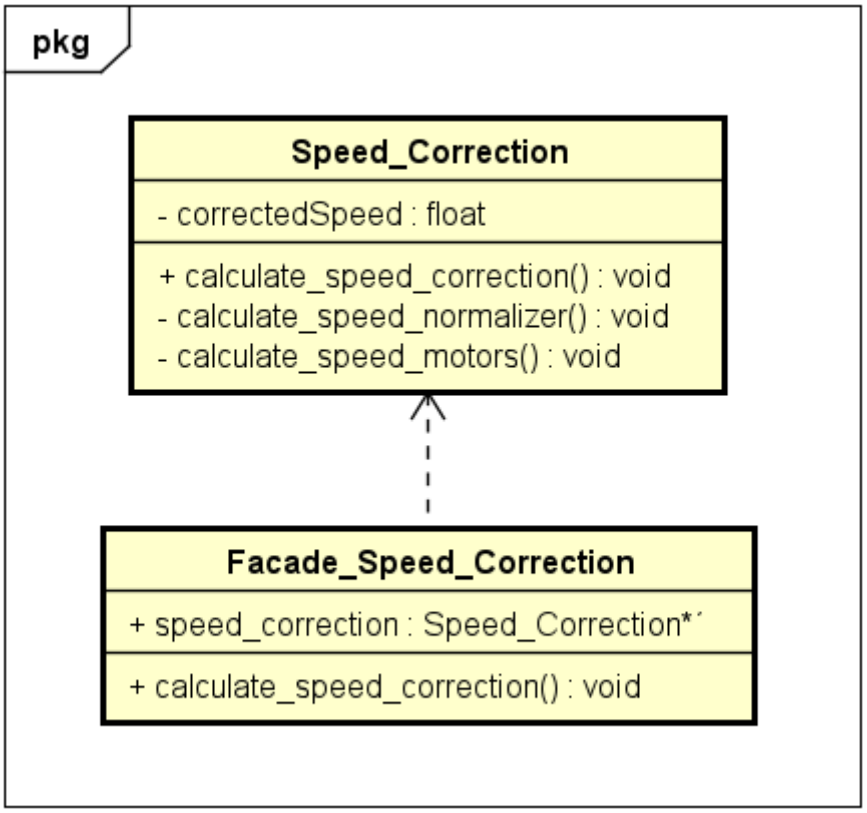


Diagrama de Classes

Figura S4.5: Diagrama de Classes



Pseudocódigo

Serviço Correção de Velocidade (S4)

Speed_Correction

include: globals.h
Initial_Hardware_Condition.h
define: N/A
variables: correctedSpeed // float, addition and subtraction value according to normalization

method calculate_speed_correction()
calculate_speed_normalizer()
calculate_speed_motors()
end method

method calculate_speed_normalizer()
// Calculate the normalizer
correctedSpeed ← (compensatorAction/100) * (speedNormalizer / 100) * averageSpeed
end method

// Calculate the speed of the motors
method calculate_speed_motors()
// Calculate speed for each right motor
rightSpeed ← averageSpeed + correctedSpeed

// Calculate speed for each left motor
leftSpeed ← averageSpeed - correctedSpeed
end method

Facade_Speed_Correction

include: NA
define: NA
variable: speed_correction // Speed_Correction instantiation from the Speed_Correction class

method calculate_speed_correction()
speed_correction.calculate_speed_correction()
end method

Documento de Requisitos

Serviço Cálculo de Erros (S5)

Introdução

O serviço deve calcular a distância e o ângulo do AGV em relação a linha guia conforme ilustrado nas imagens a seguir.

Figura S5.1: AGV na posição ideal

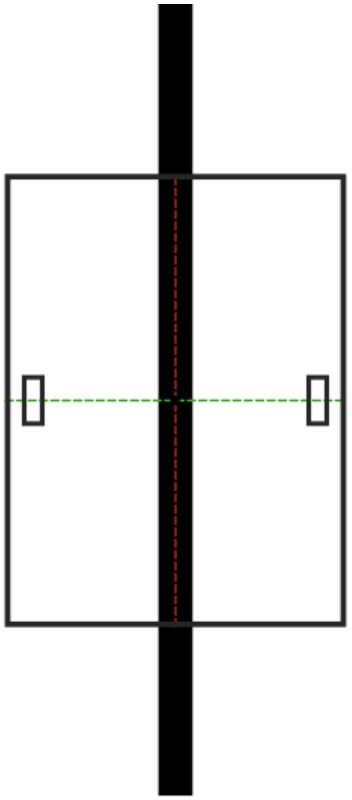
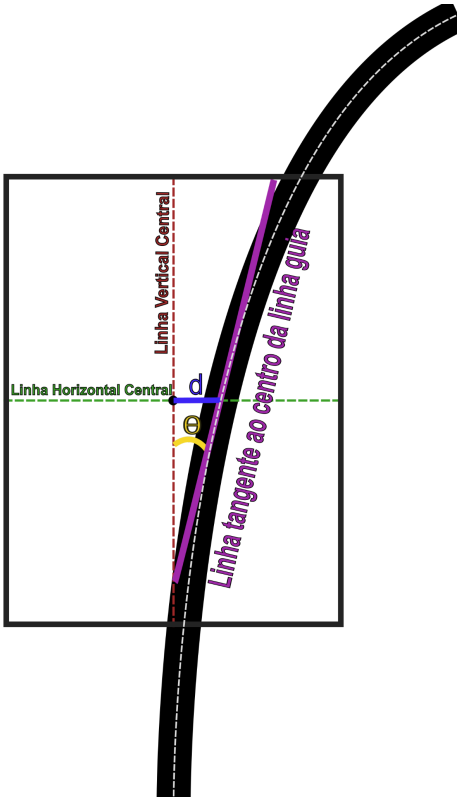


Figura S5.2: AGV erros e linhas de referência



A primeira imagem representa uma posição ideal do AGV em relação a linha guia, com os erros de ângulo e distância iguais a zero. A segunda imagem ilustra o erro em ângulo **θ** e distância **d**.

Cálculo da distância

A distância (**distanceError**) representada na imagem pela linha azul e a letra **d**, é a diferença entre o centro do AGV e o ponto central da linha guia, medido no cruzamento com a linha horizontal central.

Cálculo do ângulo

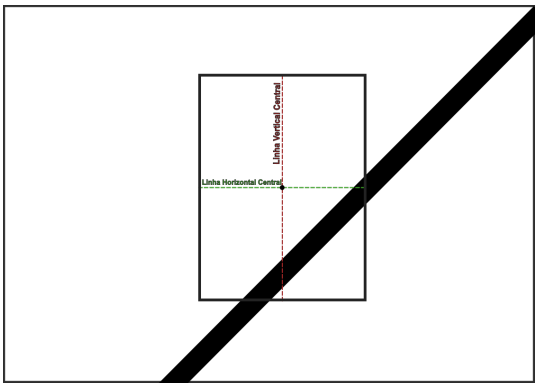
O ângulo (**angleError**) representado na imagem pela linha amarela e a letra **θ**, é formado entre a linha vertical central do AGV (vermelha) e a tangente da linha guia (roxo), no ponto de cruzamento da linha guia e a linha horizontal central do AGV (verde).

Erros

O serviço deve considerar o erro em distância positiva quando estiver à direita do centro da imagem e negativo quando estiver à esquerda. O ângulo positivo é considerado quando estiver no sentido horário da linha central vertical e negativo quando estiver no sentido anti horário.

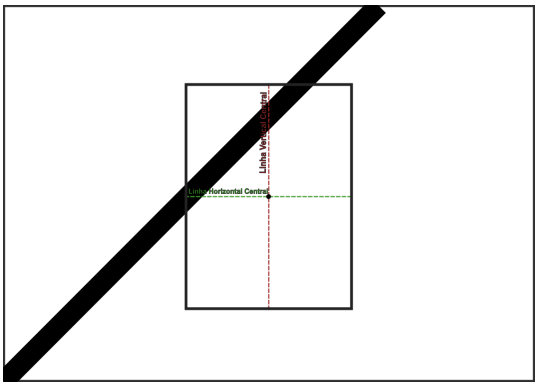
A seguir está apresentado o comportamento desejado do ângulo e distância de acordo com a imagem. Outras combinações de valores podem ocorrer, as imagens são apenas para demonstrar que os valores podem ser positivos, negativos ou zero.

Figura S5.3: Erros em distância e ângulo positivos



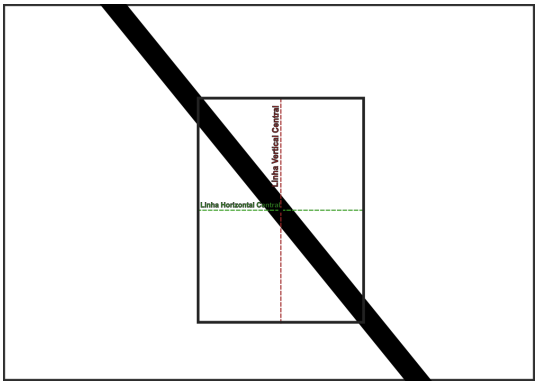
Distância e ângulo com valores positivos.

Figura S5.4: Erro em distância negativo e ângulo positivo



Distância com valor negativo e ângulo com valor positivo.

Figura S5.5: Erro em distância zero e ângulo negativo



Distância com valor zero e ângulo com valores positivos.

Descrição Geral

Funções do Produto

Cálculo do erro em distância e ângulo do AGV em relação a linha guia.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

A linha guia deve ser uma cor que se destaca em relação ao piso.

Suposições e Dependências

- O sistema presume que os parâmetros, contraste, fps e foco estejam corretamente calibrados no momento da operação.
- O serviço depende que as variáveis globais **errorDefinition**, **angleError** e **distanceError** já estejam criadas.

Requisitos Específicos

Requisitos Funcionais

RF1 - Calcular o ângulo e a distância da linha guia em relação ao centro da imagem processada.

RF2 - Os erros de ângulo e distância devem ser normalizados no intervalo de -100 a +100, e sua margem de erro é de 1mm para distância e 1° para o ângulo.

RF3 - O serviço deve modificar as variáveis globais **angleError** e **distanceError**.

RF4 - O serviço deve ler a variável global **errorDefinition**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender imagens distorcidas.
  - RNF2 - Atender imagens rasuradas.
  - RNF3 - Atender imagens desfocadas
  - RNF4 - Atender imagens com linha de referência com uma cor diferente do preto.
- **Manutenibilidade**
  - RNF5 - A manutenção como incremento e função pode-se realizar a validação de ruídos gerados por diferentes fontes de luz externa.

Documento de Demanda de Produção

Serviço Cálculo de Erros (S5)

Descrição de métodos

Filtragem da imagem

Após a imagem ser capturada, será necessário trabalhar a imagem até transformá-la em binário. Usa-se três etapas:

1. Converter a imagem para **escala de cinza**;
2. Aplicar o filtro **gaussiano**;
3. Aplicar o método de **binarização**. A binarização utilizará um limiar fixo para separar os elementos da linha de referência do fundo.

Para as aplicações dos filtros a biblioteca utilizada será o OpenCV.

Escolha do erro

Deve-se escolher o erro de ângulo ou distância ou ambos.

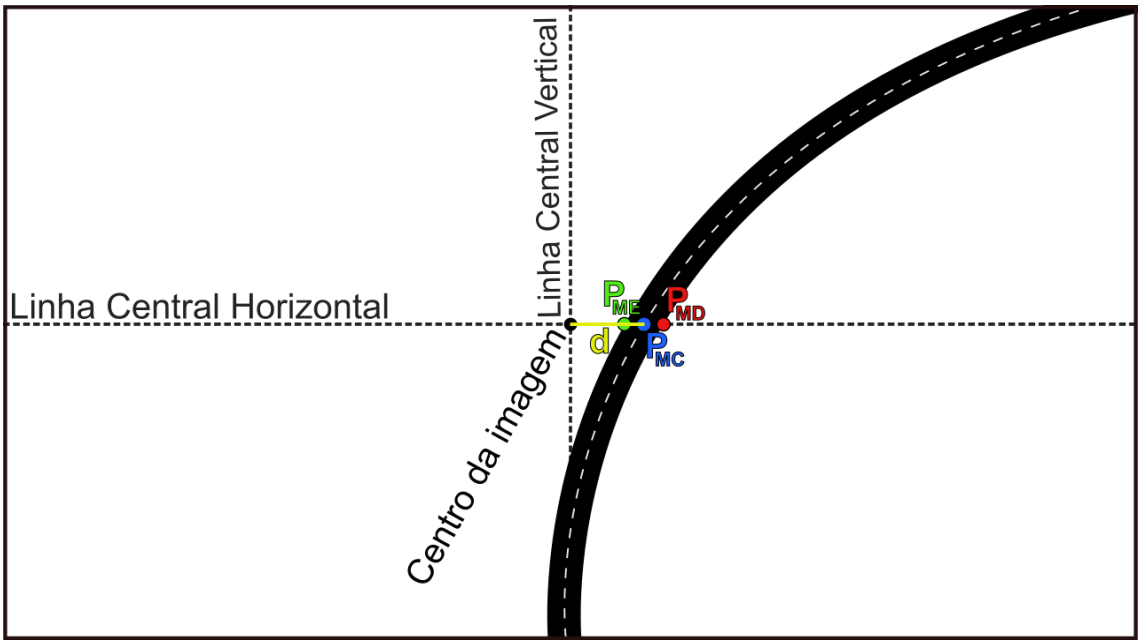
Utilizar a variável global **errorDefinition**. Quando a variável for igual a 1 será realizado somente o cálculo de distância. Quando a variável for igual a 2 será realizado somente o cálculo de ângulo. Quando a variável for diferente dos valores citados, será calculado distância e ângulo.

Cálculo da Distância

O cálculo da distância (**d**) é realizado com base nas coordenadas de dois pontos e uma linha de referência na imagem:

1. A linha de referência está localizada a 50% da altura da imagem e é representada pela linha cinza denominada de Linha Central Horizontal.
2. O primeiro ponto preto encontrado nessa Linha Central Horizontal está marcado na imagem pela cor verde denominado $\mathbf{P_{ME}}$.
3. O último ponto preto encontrado na mesma Linha Central Horizontal está marcado na imagem pela cor vermelha denominado $\mathbf{P_{MD}}$.

Figura S5.6: Linhas e pontos de referência para cálculo de distância



O objetivo de encontrar os dois pontos é determinar o ponto central entre eles $\mathbf{P_{MC}}$, representado pelo ponto azul na imagem. E enfim encontrar a distância entre o ponto central e o centro da imagem.

Usando as seguintes equações encontra-se a distância da linha guia (**d**).

$$x_{MC} = \frac{x_{ME} + x_{MD}}{2} \quad \text{[equação 1]}$$

$$Ponto\ central\ da\ imagem = \frac{número\ de\ colunas}{2} \quad \text{[equação 2]}$$

$$d_{pixels} = x_{MC} - Ponto\ central\ da\ imagem \quad \text{[equação 3]}$$

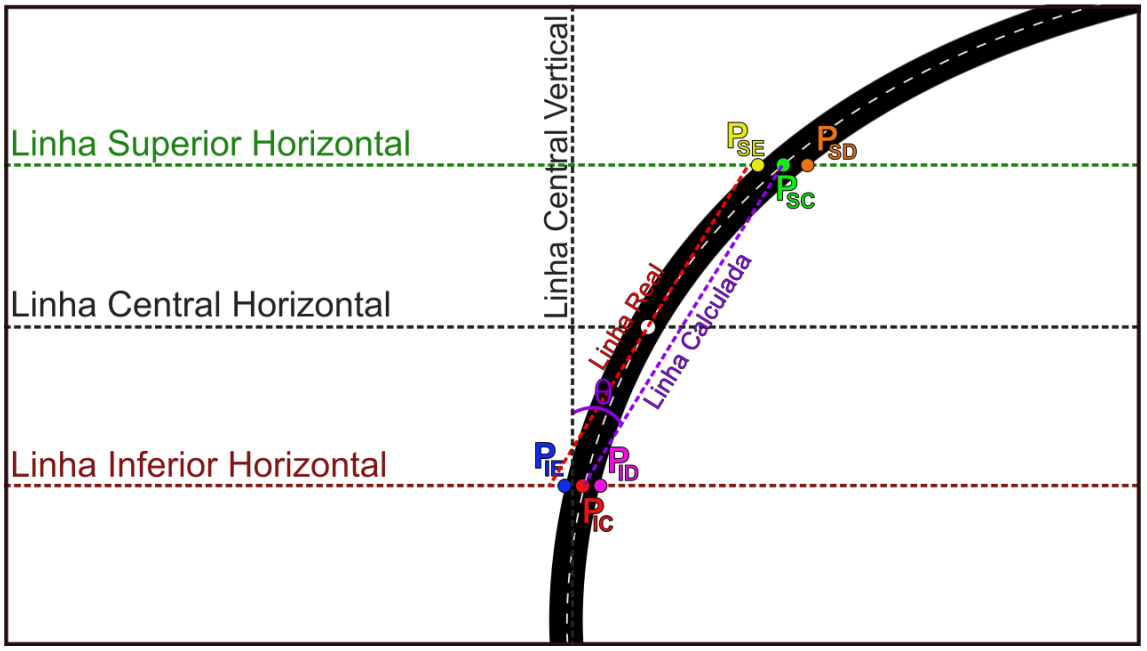
$$d = \frac{d_{pixels}}{número\ de\ colunas/2} \cdot 100 \quad \text{[equação 4]}$$

Cálculo do Ângulo

O cálculo do ângulo (**θ**) é realizado com base nas coordenadas de quatro pontos e duas linhas de referência na imagem, esses cálculos resultam em uma linha paralela a Linha Real que desejamos calcular o ângulo, portanto ao calcular o ângulo dessa nova linha temos o valor do ângulo da Linha Real. Os quatros pontos são:

1. O primeiro ponto preto encontrado na linha localizada a 25% da altura da imagem, que está representada pela linha verde. Esse ponto está marcado na imagem pela cor amarela denominado $\mathbf{P_{SE}}$.
2. O último ponto preto encontrado na linha localizada a 25% da altura da imagem, que está representada pela linha verde. Esse ponto está marcado na imagem pela cor laranja $\mathbf{P_{SD}}$.
3. O primeiro ponto preto encontrado na linha localizada a 75% da altura da imagem, que está representada pela linha vermelha. Esse ponto está marcado na imagem pela cor azul $\mathbf{P_{IE}}$.
4. O último ponto preto encontrado na linha localizada a 75% da altura da imagem, que está representada pela linha vermelha. Esse ponto está marcado na imagem pela cor roxo $\mathbf{P_{ID}}$.

Figura S5.7 Linhas e pontos de referência para cálculo de ângulo



O objetivo de encontrar os quatro pontos é para determinar dois pontos:

- Ponto representado pela cor verde $\mathbf{P_{SC}}$, sendo o meio entre os pontos $\mathbf{P_{SE}}$ e $\mathbf{P_{SD}}$.
- Ponto representado pela cor vermelha $\mathbf{P_{IC}}$, sendo o meio entre os pontos $\mathbf{P_{IE}}$ e $\mathbf{P_{ID}}$.

Usando as seguintes equações encontra-se o ângulo da linha guia.

$$x_{SC} = (x_{SD} + x_{SE}) / 2 \quad \text{[equação 5]}$$

$$y_{SC} = y_{SD} = y_{SE} \quad \text{[equação 6]}$$

$$x_{IC} = (x_{ID} + x_{IE}) / 2 \quad \text{[equação 7]}$$

$$y_{IC} = y_{ID} = y_{IE} \quad \text{[equação 8]}$$

$$\Delta_x = x_{IC} - x_{SC} \quad \text{[equação 9]}$$

$$\Delta_y = y_{IC} - y_{SC} \quad \text{[equação 10]}$$

$$\theta_{rad} = - arctan(\frac{\Delta_y}{\Delta_x}) \quad \text{[equação 11]}$$

$$\theta = sen(\theta_{rad} - \pi/2) \cdot 100 \quad \text{[equação 12]}$$

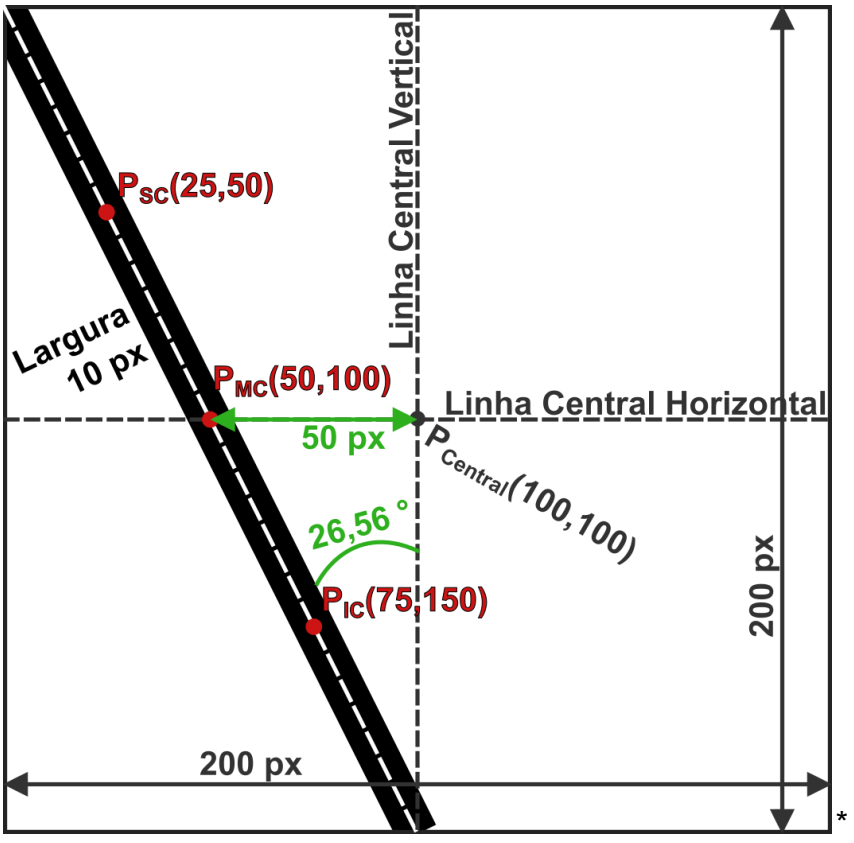
A função trigonométrica **arctan** está com o sinal negativo para que o sentido horário seja positivo e o sentido anti-horário seja negativo.

Na equação 12 ocorre a subtração por $\pi/2$ para mudar a orientação, pois a referência do openCV é a linha central vertical, e a referência para o serviço é a linha central horizontal.

Exemplo de uso e serviço

1. Com a seguinte linha guia descrita na imagem.

Figura S5.8 Guia de construção da imagem de teste



Respeitando toda a descrição deve ser criado a linha guia sem a necessidade da linha branca tracejada no centro, das linhas de referência e das descrições em texto.

2. Os resultados retornados devem ser:
    - **angleError** = -44,72
    - **distanceError** = - 50

Referências

TCC Rodrigo Santos.

Tear_DrWallacePNR_RelatorioTecnico_Relatorio_Tecnico_20220805

Histórico de alterações

14/11/24:

- Reestruturação completa do documento para englobar uso geral do serviço, e não apenas para AGVs.
- Adição de métodos de obtenção e inserção de valores dos ganhos KP e KD, em falta desde a última alteração do documento.

23/11/24:

- Alteração completa nos métodos, removendo os métodos computacionais e descrevendo o que é método e como funciona PID que será utilizado pelo serviço.

25/11/24:

- Escrita mais direta da documentação.
- Adição da fórmula para explicar o algoritmo.

26/11/24

- Adição do cálculo dos termos.
- Adição das variáveis globais t, ti e td.

11/12/24

- Alteração das imagens
- Novo exemplo com valores não triviais.

16/01/25

- Adição da linha paralela na imagem do cálculo do ângulo.

01/09/25

- Alteração dos filtros e da sequência, de mediana, escala de cinza e binarização, para escala de cinza, gaussiana e binarização.

Serviço Cálculo de Erros (S5)

Diagrama de Casos de Uso

Figura S5.9: Diagrama de Casos de Uso

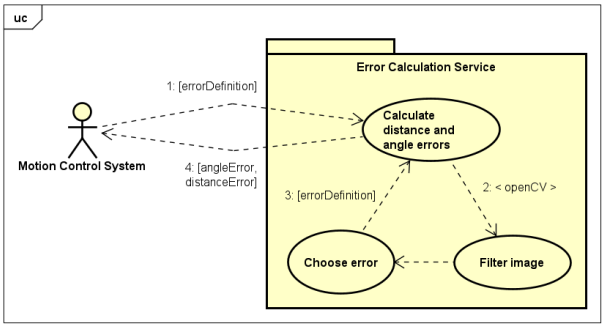


Diagrama de Atividades

Figura S5.9: Diagrama de Atividades

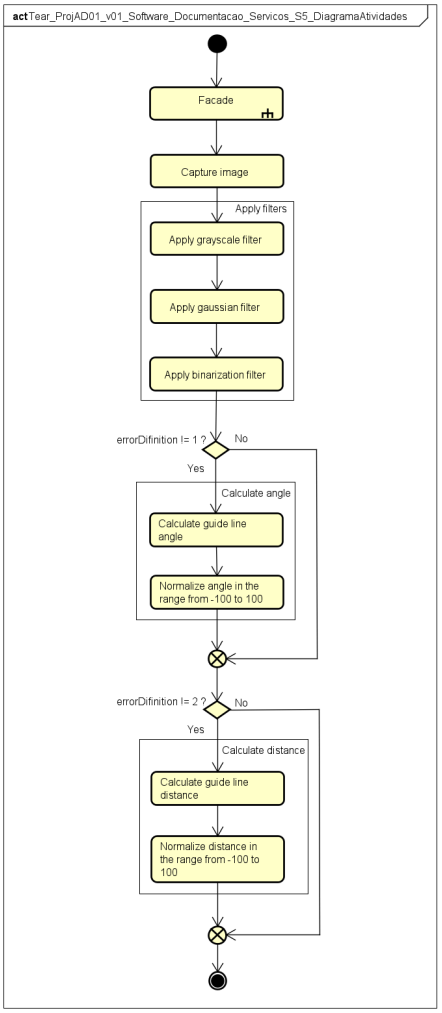


Diagrama de Sequência

Figura S5.9: Diagrama de Sequência

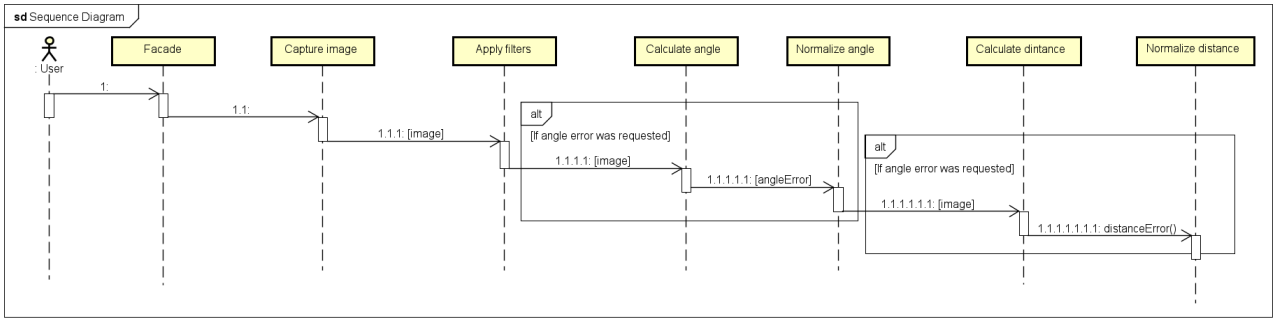
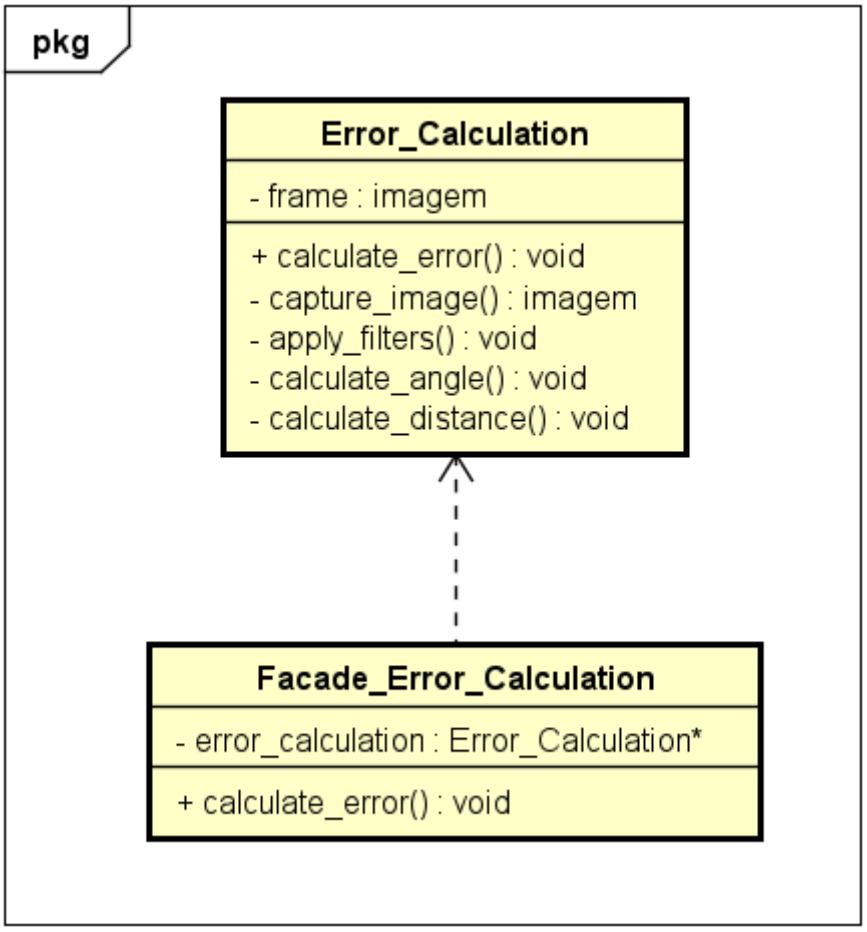


Diagrama de Classes

Figura S5.9: Diagrama de Classes



Pseudocódigo

Serviço Cálculo de Erros (S5)

Error_Calculation

```
include: opencv
         cmath
         iostream
         globals.h

define:
variables: cv::Mat filtered_image;
           cv::Mat captured_image;


method calculate_error()
        capture_image()
        apply_filters()
        if errorDefinition != 1
                calculate_angle()
        if errorDefinition != 2
                calculate_distance()
end method

// Capture image
method capture_image()
        cv::VideoCapture cap(0)
        Capture captured_image with cap
end method

// Apply filters
method apply_filters()

        filtered_image ← captured_image CLONE

        // Apply grayscale filter
        if filtered_image HAS 3 channels CONVERT filtered_image TO grayscale

        // Apply gaussian filter
        APPLY GaussianBlur ON filtered_image WITH kernel OF (11,11) AND
        standard deviation OF 0

        // Apply binary threshold
```

```
        APPLY threshold ON filtered_image WITH threshold 25, max value 255

end method

// Calculate angle
method calculate_angle()
        topRow ← floor(0.25 * number of rows)

        for each column in the topRow
                if pixel value at (row, column) is equal to 0
                        topLeftPoint ← (column, topRow)
                        break

        for each column in the topRow, starting from the last column
                if pixel value at (topRow, column) is equal to 0
                        topRightPoint ← (column, topRow)
                        break

        bottomRow ← floor(0.75 * number of rows)

        for each column in the bottomRow
                if pixel value at (bottomRow, column) is equal to 0
                        bottomLeftPoint ← (column, bottomRow)
                        break

        for each column in the bottomRow, starting from the last column
                if pixel value at (bottomRow, column) is equal to 0
                        bottomRightPoint ← (column, bottomRow)
                        break

        topCenterPoint = (topRightPoint.x - topLeftPoint.x) / 2, topRightPoint.y
                bottomCenterPoint = (bottomRightPoint.x - bottomLeftPoint.x) / 2,
                bottomRightPoint.y
        delta_x ← bottomCenterPoint.x - topCenterPoint.x
        delta_y ← bottomCenterPoint.y - topCenterPoint.y
        angle_rad ← - atan2(delta_y, delta_x)

        // Normalize angle in the range from -100 to 100
        angleError ← sin(angle_rad-(pi/2)) * 100

end method

// Calculate distance
method calculate_distance()
```

```
        row ← floor(0.5 * number of rows)

        for each column in the row
            if pixel value at (row, column) is equal to 0
                first ← (column, row)
                break

        for each column in the row, starting from the last column
            if pixel value at (row, column) is equal to 0
                last ← (column, row)
                break

        middle ← ((first.x + last.x) / 2, (first.y + last.y) / 2)
        center ← (number of columns / 2, number of rows / 2)
        distance_pixels ← (middle.x - center.x)

        // Normalize distance in the range from -100 to 100
        distanceError ← (distance_pixels/(number of columns / 2)) * 100

end method
```

Facade_Error_Calculation

include: NA
define: NA
variable: error_calculate // Error_Calculation instantiation from the Error_Calculation class

```
method calculate_error()
        error_calculation.calculate_error()
end method
```

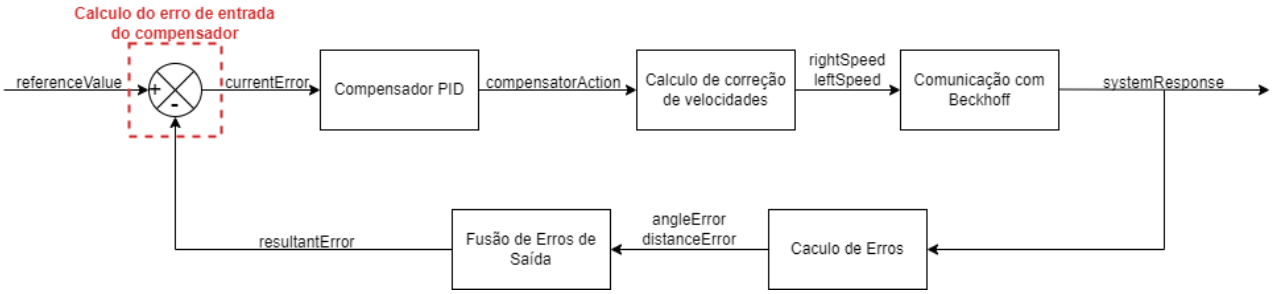
Documento de Requisitos

Serviço cálculo de erro de entrada do compensador (S6)

Introdução

O serviço deve calcular a entrada do compensador utilizando a entrada do sistema **referenceValue** e o erro **resultantError**. É possível ver o serviço destacado em vermelho na malha de controle representado na Figura 1.

Figura S6.1: Serviço cálculo de erro de entrada do compensador na malha de controle



Como apresentado na Figura 1, a saída do serviço é o erro de entrada do compensador **currentError**, sendo calculado pela seguinte equação.

$$currentError = referenceValue - resultantError \quad \text{[equação 1]}$$

Descrição Geral

Funções do Produto

Cálculo do erro de entrada do serviço compensador PID.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que a variável global **currentError** já esteja criada.
- O serviço presume que as variáveis **referenceValue** e **resultantError** estejam com valor atribuído.

Requisitos Específicos

Requisitos Funcionais

RF1 - Calcular a entrada do compensador PID.

RF2 - A variável **currentError** deve ser saturada no intervalo de -100 a 100.

RF3 - O serviço deve modificar a variável global **currentError**.

RF4 - O serviço deve ler a variáveis globais **referenceValue** e **resultantError**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender valores positivos e negativos de **referenceValue**.
  - RNF2 - Suportar uma margem de erro de 0,01.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Serviço cálculo de erro de entrada do compensador (S6)

Descrição de métodos

Subtração

Após o valores de entrada **referenceValue** e **resultantError**, a subtração é realizada de acordo com a equação 1.

$$currentError = referenceValue - resultantError \quad \text{[equação 1]}$$

Caso o valor ultrapasse 100, **currentError** recebe o valor 100. Caso seja inferior a -100, o **currentError** recebe -100, saturando a saída.

Exemplo de uso e serviço

1. Com os valores de **referenceValue** = 25 e **resultantError** = 50.
2. O resultado retornado deve ser:
    - **currentError** = -25

Referências

Histórico de alterações

-

Serviço cálculo de erro de entrada do compensador (S6)

Diagrama de Casos de Uso

Figura S6.2: Diagrama de Casos de Uso

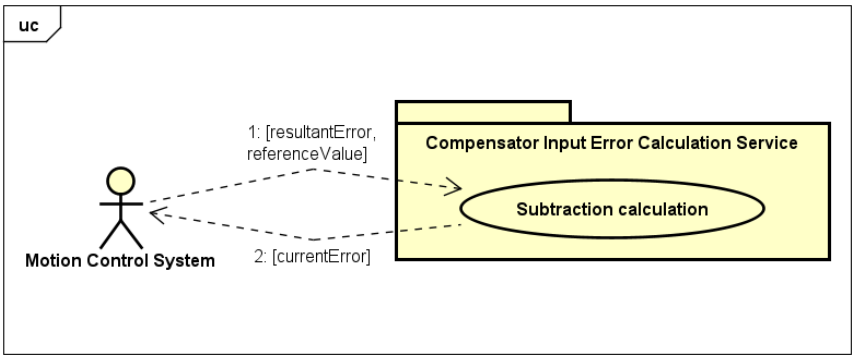


Diagrama de Atividades

Figura S6.3: Diagrama de Atividades

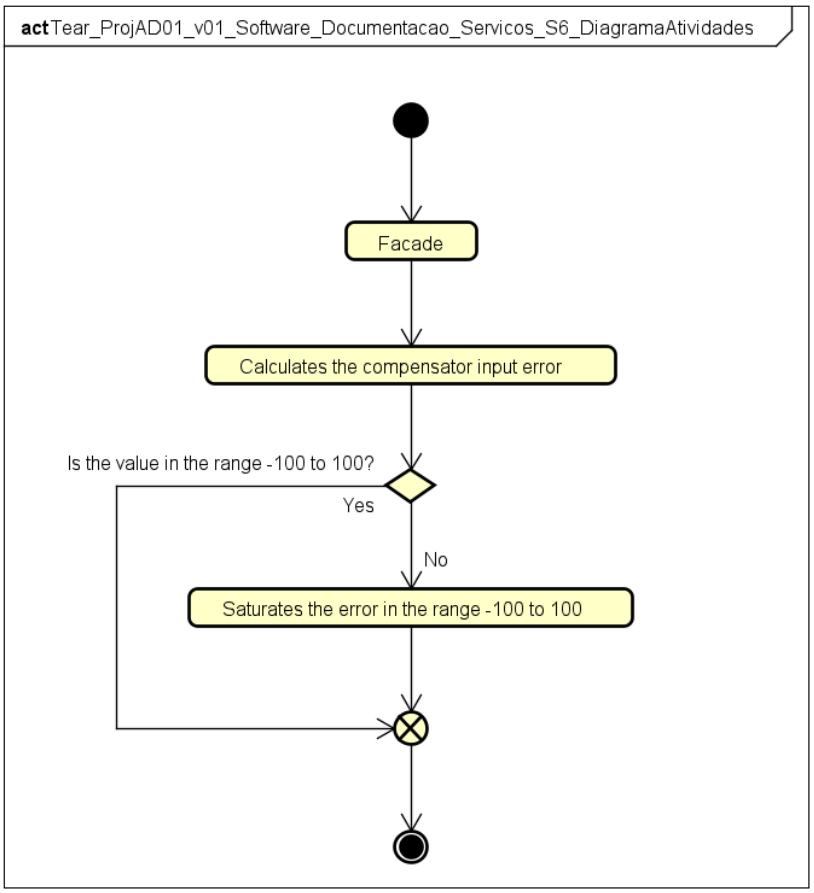


Diagrama de Sequência

Figura S6.4: Diagrama de Sequência

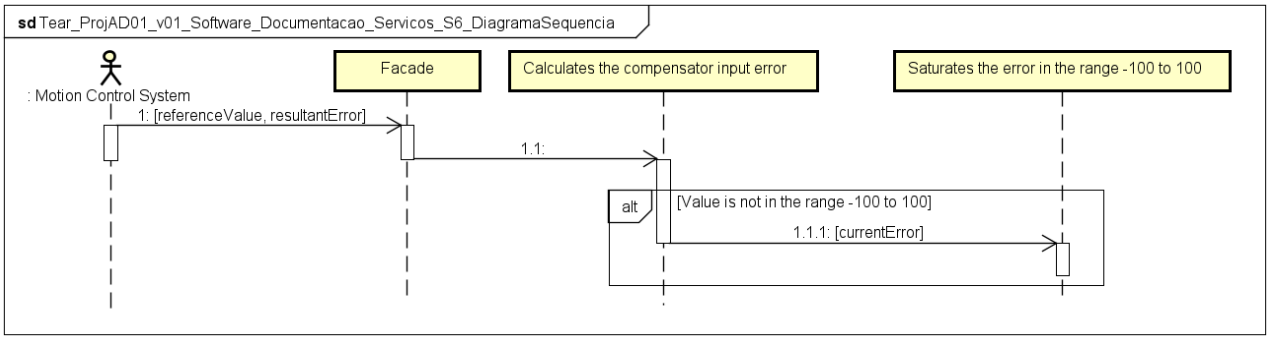
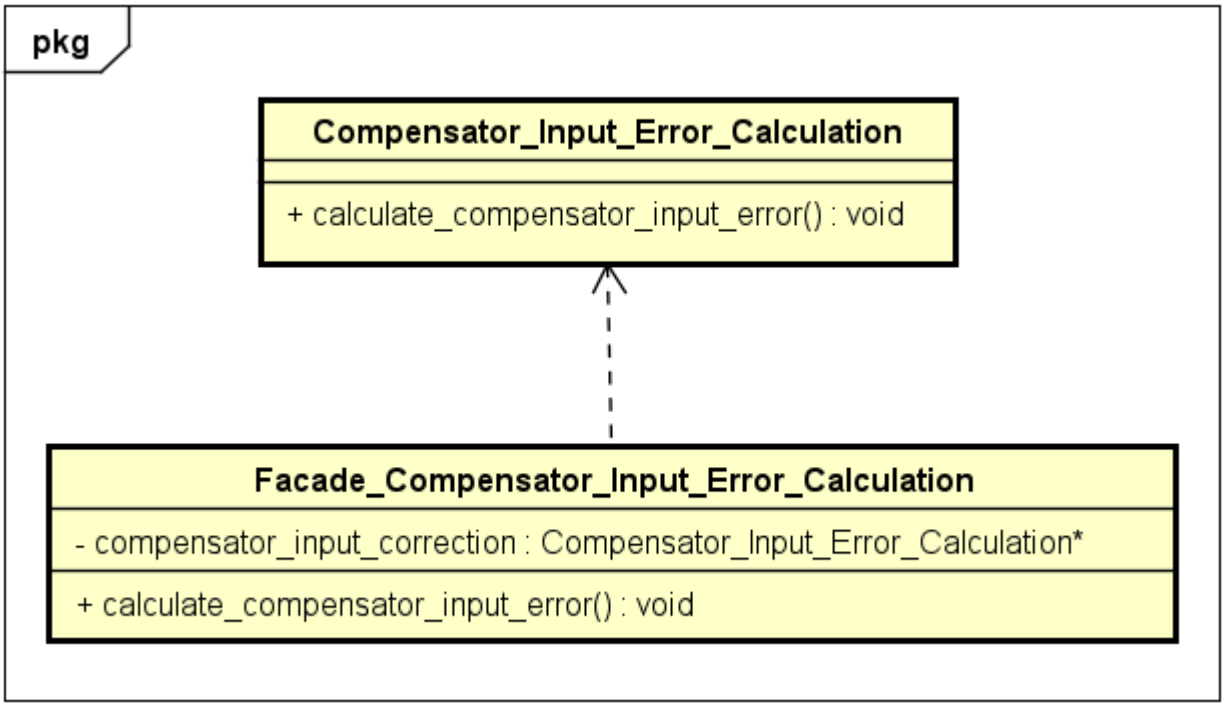


Diagrama de Classes

Figura S6.5: Diagrama de Classes



Pseudocódigo

Serviço cálculo de erro de entrada do compensador (S6)

Compensator_Input_Error_Calculation

include: globals.h
define: N/A
variables: N/A

method calculate_compensator_input_error()
    // Calculates the compensator input error
    currentError ← referenceValue - resultantError

    // Saturates the error in the range -100 to 100
    if currentError > 100
        currentError ← 100
    if currentError < -100
        currentError ← -100
    end method

Facade_Compensator_Input_Error_Calculation

include: NA
define: NA
variable: compensator_input_error_calculation // Compensator_Input_Error_Calculation instantiation from the Compensator_Input_Error_Calculation class

method calculate_compensator_input_error()
    compensator_input_error_calculation.calculate_compensator_input_error()
end method

Documento de Requisitos

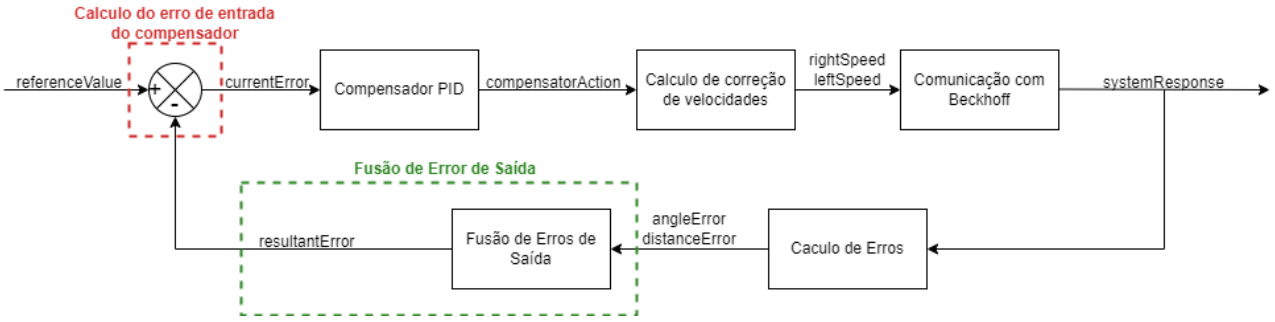
Serviço de Fusão de Erros de Saída (S7)

Introdução

O serviço deve calcular o valor do erro **resultantError**, com base na escolha do usuário de qual erro será utilizado para a malha de controle. A malha de controle poderá utilizar o erro em distância, erro em ângulo ou a fusão dos dois erros.

Como apresentado na Figura 1, **resultantError** é o erro que a malha de controle irá utilizar para o cálculo de entrada do compensador.

Figura S7.1: Serviço de cálculo de fusão de erros de saída na malha de controle



Descrição Geral

Funções do Produto

Calcular o erro de saída **resultantError**.

Características do Usuário

Este serviço é consumido pelo Sistema de Controle de Movimento.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que a variável global **resultantError** já esteja criada.
- O serviço presume que as variáveis **angleError, distanceError** e **errorDefinition** estejam com valor atribuído.

Requisitos Específicos

Requisitos Funcionais

RF1 - Calcular a saída do sistema..

RF2 - A variável **resultantError** deve ser normalizada no intervalo de -100 a 100.

RF3 - O serviço deve modificar a variável global **resultantError**.

RF4 - O serviço deve ler a variáveis globais **angleError, distanceError** e **errorDefinition**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender valores positivos e negativos de **angleError, distanceError**.
  - RNF2 - Suportar uma margem de erro de 0,01.]
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Serviço de Fusão de Erros de Saída (S7)

Descrição de métodos

Definição do erro

Deve-se escolher o erro de ângulo ou distância ou ambos.

Utilizar a variável global **errorDefinition**. Quando a variável for igual a 1 será realizado somente o cálculo de distância. Quando a variável for igual a 2 será realizado somente o cálculo de ângulo. Quando a variável for diferente dos valores citados, será calculado distância e ângulo.

Erro em distância

Caso o erro escolhido for o erro em distância **resultantError** será igual **distanceError**.

.Erro em ângulo

Caso o erro escolhido for o erro em ângulo **resultantError** será igual **angleError**.

Média de erros

Caso o erro escolhido for a fusão do erro em distância e erro em ângulo, **resultantError** será definido conforme a equação 1.

$$resultantError = \frac{disntanceError + angleError}{2}$$ [equação 1]

Exemplo de uso e serviço

1. Com os valores de **angleError** = -25, **distanceError** = 50 e **errorDefinition** = 0;
2. O resultado retornado deve ser:
   - **resultantError** = 12,5

Referências

A inserir.

Histórico de alterações

17/02/25:

- Quarto método renomeado para evitar métodos com nomes iguais.

Serviço de Fusão de Erros de Saída (S7)

Diagrama de Casos de Uso

Figura S7.2: Diagrama de Casos de Uso

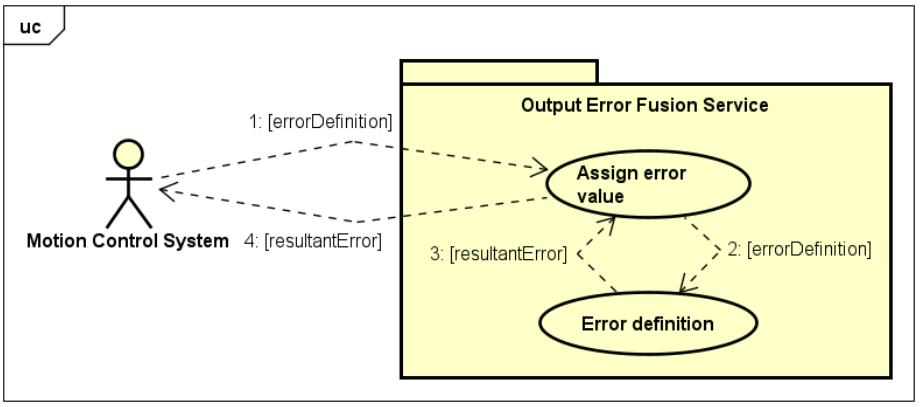


Diagrama de Atividades

Figura S7.3: Diagrama de Atividades

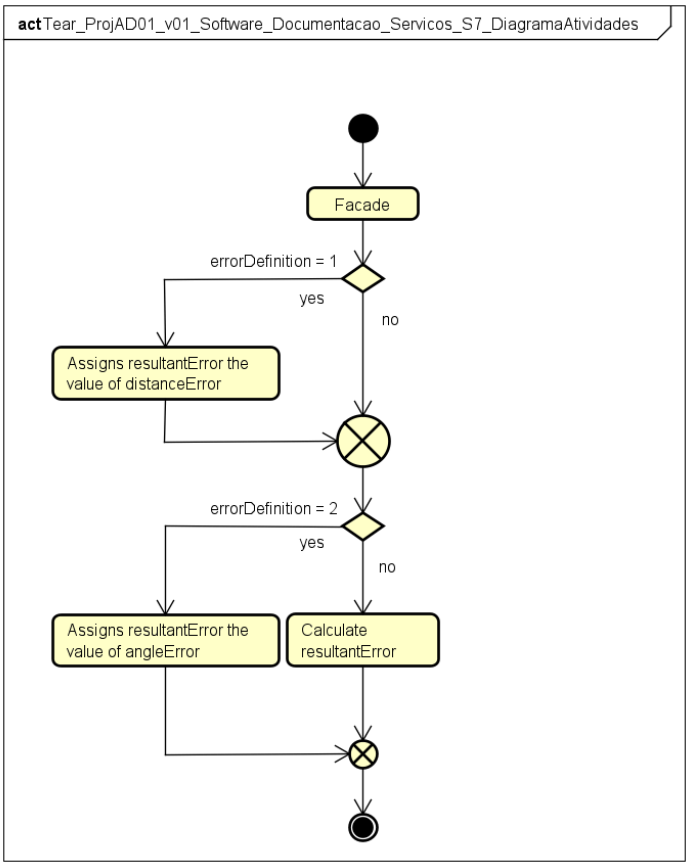


Diagrama de Sequência

Figura S7.4: Diagrama de Sequência

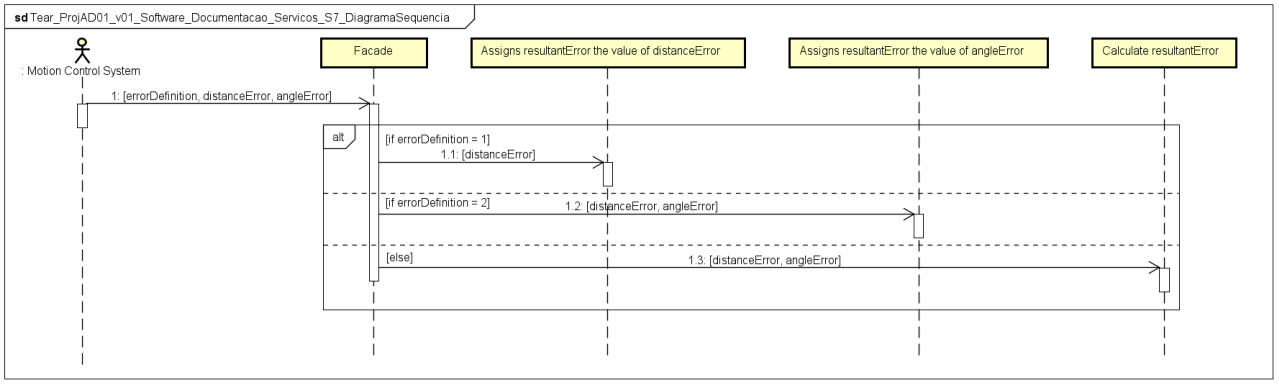
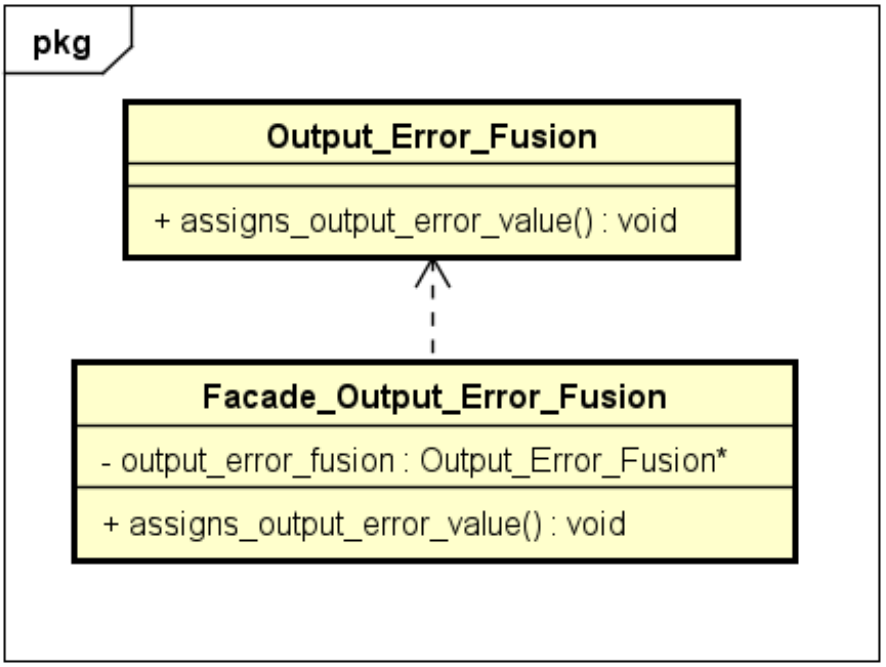


Diagrama de Classes

Figura S7.5: Diagrama de Classes



Pseudocódigo

Serviço de Fusão de Erros de Saída (S7)

Output_Error_Fusion

include: globals.h
define: N/A
variables: N/A

```
method assigns_output_error_value()
        if errorDefinition = 1
                // Assigns resultantError the value of distanceError
                resultantError ← distanceError
        if errorDefinition = 2
                // Assigns resultantError the value of angleError
                resultantError ← angleError
        else
                // Calculates resultantError
                resultantError ← (distanceError + angleError) / 2
end method
```

Facade_Output_Error_Fusion

include: NA
define: NA
variable: output_error_fusion // Output_Error_Fusion instantiation from the Output_Error_Fusion class

```
method assigns_output_error_value()
        output_error_fusion.assigns_output_error_value()
end method
```

Documento de Requisitos

Serviço Entrada Manual de Dados (S8)

Introdução

O serviço deve atribuir valores determinados pelo usuário, em variáveis globais. Sendo essas variáveis:

- **kp** (Ganho proporcional do compensador PID).
- **ti** (Ganho integral do compensador PID).
- **td** (Ganho derivativo do compensador PID).
- **t** (Período de execução da malha de controle).
- **errorDefinition** (Erro utilizado na malha de controle).
- **averageSpeed** (Velocidade média do AGV).
- **speedNormalizer** (Limite da variação de velocidade das rodas do AGV).
- **definedReference** (Referência da malha de controle).

Descrição Geral

Funções do Produto

Atribuir os valores que o usuário determinou para as variáveis selecionadas.

Características do Usuário

Este serviço é consumido pelo Sistema de Atribuição de Dados.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que as variáveis globais **kp, ti, td, t, errorDefinition, averageSpeed, speedNormalizer** e **definedReference** já estejam criadas.

Requisitos Específicos

Requisitos Funcionais

RF1 - Apresentar em forma de texto uma guia para que o usuário possa determinar os valores para cada variável.

RF2 - O serviço deve modificar as variáveis globais **kp, ti, td, t, errorDefinition, averageSpeed, speedNormalizer** e **definedReference**.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Limitar quando necessário o valor da variável.

- **Manutenibilidade**
    - Não identificado.

Documento de Demanda de Produção

Serviço Entrada Manual de Dados (S8)

Descrição de métodos

Entrada de dados

Para que o usuário saiba quando determinar cada valor, será apresentado os seguintes textos para cada variável:

- **kp** (Valor Kp desejado “Valor 0 não aceito”).
- **ti** (Valor Ti desejado).
- **td** (Valor Td desejado).
- **t** (Período de execução da malha de controle “Valor 0 não aceito”).
- **errorDefinition** (Digite 1 para que o erro escolhido seja em distância, 2 para que o erro escolhido seja em ângulo e qualquer outro valor para que o erro escolhido seja os dois).
- **averageSpeed** (Velocidade média do AGV de 5 a 95).
- **speedNormalizer** (Limite da variação de velocidade das rodas do AGV de 5 a 50).
- **definedReference** (Referência da malha de controle, não é permitido valores negativos).

Limitar os valores dos dados

Quando o valor definido pelo usuário não está no intervalo aceito, será apresentado novamente o texto para que o usuário determine um novo valor.

Exemplo de uso e serviço

1. Caso o usuário insira os seguintes valores para cada variável:
   - **kp** = 1;
   - **ti** = 1;
   - **td** = 1;
   - **t** = 1;
   - **errorDefinition** = 0;
   - **averageSpeed** = 10;
   - **speedNormalizer** = 5;
   - **definedReference** = 0;
2. Esses mesmos valores devem estar armazenados em suas respectivas variáveis.

Referências

Histórico de alterações

-

Serviço Entrada Manual de Dados (S8)

Diagrama de Casos de Uso

Figura S8.1: Diagrama de Casos de Uso

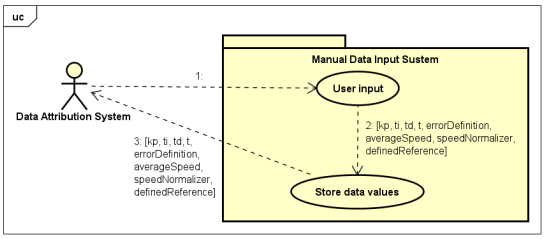


Diagrama de Atividades

Figura S8.2: Diagrama de Atividades

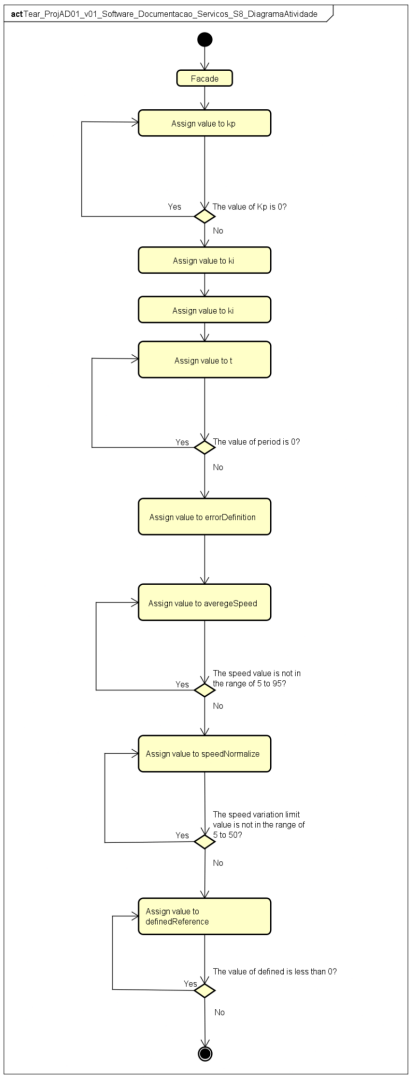


Diagrama de Sequência

Figura S8.3: Diagrama de Sequência

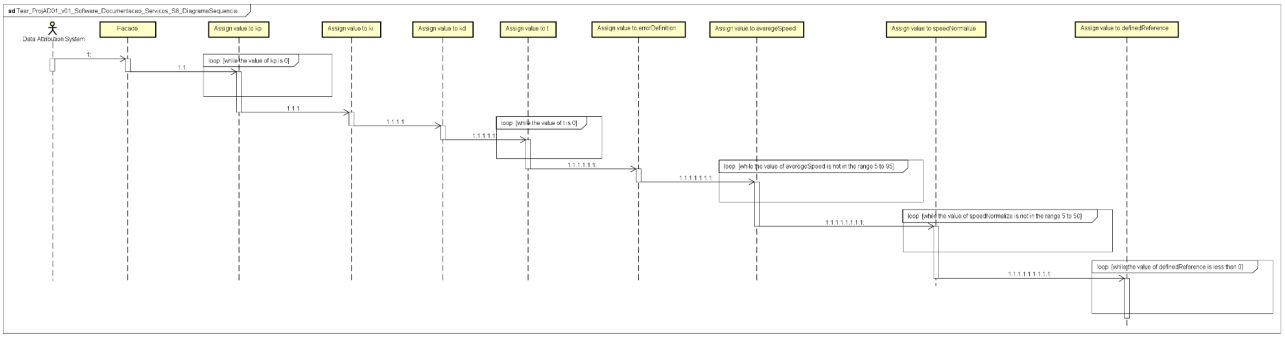
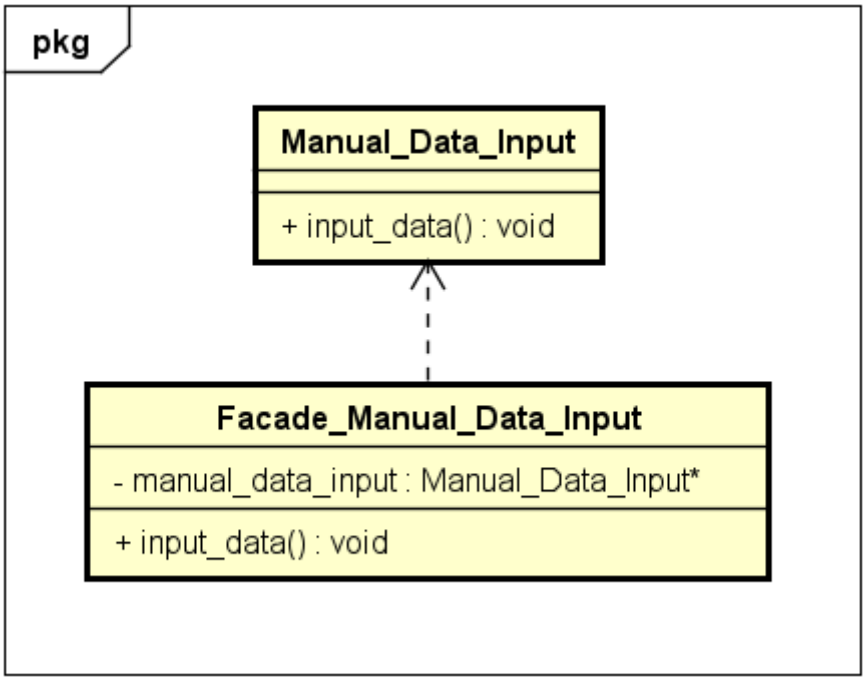


Diagrama de Classes

Figura S8.4: Diagrama de Classes



Pseudocódigo

Serviço Entrada Manual de Dados (S8)

Manual_Data_Input

include: globals.h
define: N/A
variables: N/A

method input_data()

```
// Assign value to kp
kp ← 0
while kp = 0
        print Valor Kp desejado “Valor 0 não aceito”
        read ← kp

// Assign value to ki
print Valor ti desejado
read ← ti

// Assign value to ki
print Valor td desejado
read ← td

// Assign value to t
t ← 0
while t = 0
        print Período de execução da malha de controle “Valor 0 não aceito”
        read ← t

// Assign value to errorDefinition
print Digite 1 para que o erro escolhido seja em distância, 2 para que o erro escolhido seja em ângulo e qualquer outro valor para que o erro escolhido seja os dois
read ← errorDefinition

// Assign value to averageSpeed
averageSpeed ← 0
while averageSpeed < 5 or averageSpeed > 95
        print Velocidade média do AGV de 5 a 95
        read ← averageSpeed


// Assign value to speedNormalizer
speedNormalizer ← 0
```

```
        while speedNormzalizer < 5 or speedNormzalizer > 50
            print Limite de variação de velocidade das rodas do AGV de 5 a 50
            read ← speedNormalizer

        // Assign value to definedReference
        definedReference ← -1
        while definedReference < 0
            print Referência da malha de controle, não é permitido valores
        negativos
            read ← definedReference

end method
```

Facade_Manual_Data_Input

```
include: NA
define: NA
variable: manual_data_input // Manual_ Data_Input  instantiation from the
Manual_Data_Input class

method input_data()
    manual_data_input.input_data()
end method
```

Documento de Requisitos

Serviço de Geração de Referência (S9)

Descrição Geral

Funções do Produto

O serviço deve gerar o sinal de entrada da malha de controle.

Características do Usuário

Este serviço é consumido pelo Sistema de Atribuição de Dados.

Restrições Gerais

Não identificado.

Suposições e Dependências

- O serviço depende que a variável global **referenceValue** já esteja criada.
- O serviço depende que a variável global **definedReference** já esteja com valor atribuído.

Requisitos Específicos

Requisitos Funcionais

RF1 - Utilizar a variável global **referenceValue**.

RF2 - Modificar a variável global **definedReference**.

RF3 - Gerar o sinal de entrada da malha de controle.

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - RNF1 - Atender entrada zero
  - RNF2 - Atender entrada degrau.
- **Manutenibilidade**
  - RNF3 - A manutenção como incremento de função pode-se realizar a validação em entrada em rampas e sigmoides.

Documento de Demanda de Produção

Serviço de Geração de Referência (S9)

Descrição de métodos

Definição da variável de referência

Para a atual versão as entradas são constantes os valores serão definidos pelo usuário, portanto a variável global **referenceValue** será igual a **definedReference**.

Exemplo de uso e serviço

1. Para valores **definedReference** = 1, a saída deve ser **referenceValue** = 1.

Referências

Histórico de alterações

-

Serviço de Geração de Referência (S9)

Diagrama de Casos de Uso

Figura S9.1: Diagrama de Casos de Uso

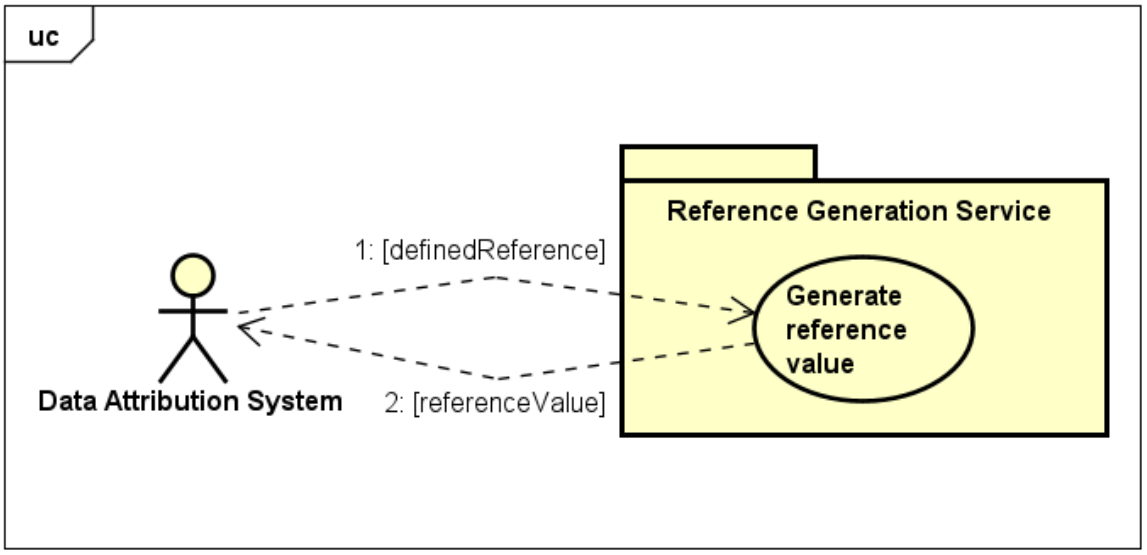


Diagrama de Atividades

Figura S9.2: Diagrama de Atividades

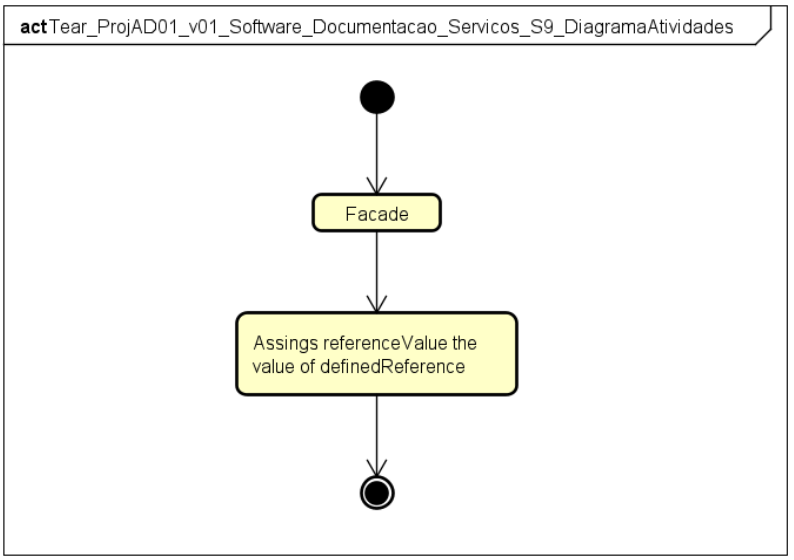


Diagrama de Sequência

Figura S9.3: Diagrama de Sequência

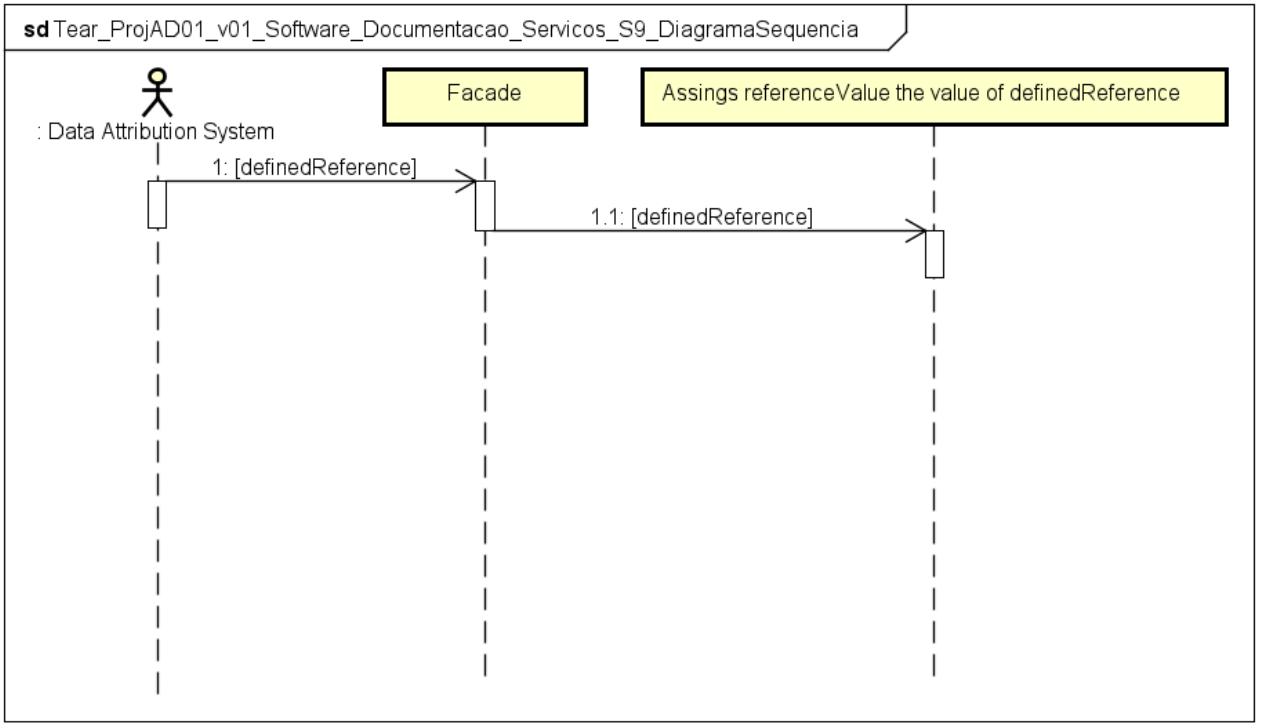
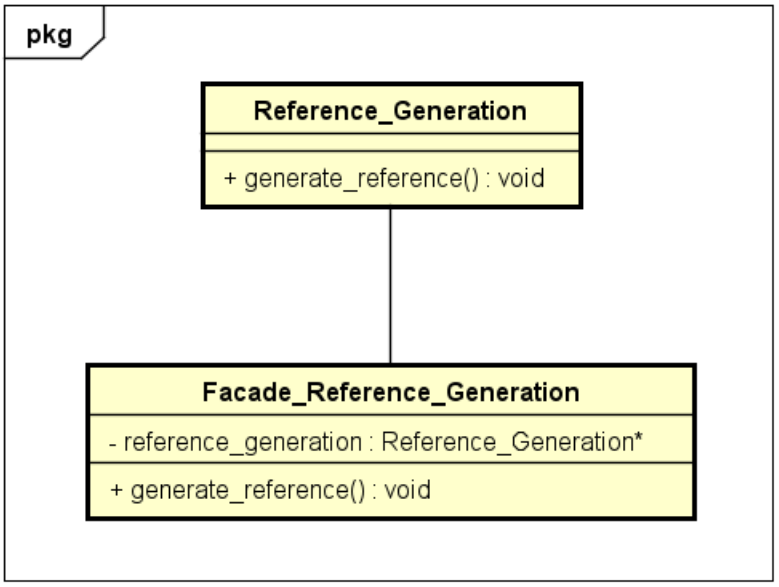


Diagrama de Classes

Figura S9.4: Diagrama de Classes



Pseudocódigo

Serviço de Geração de Referência (S9)

Reference_Generation

include: iostream
globals.h
define: NA
variables: NA

method generate_reference()
// Assings referenceValue the value of definedReference
referenceValue ← definedReference
end method

Facade_Reference_Generation

include: NA
define: NA
variable: reference_generation // Reference_Generation instantiation from the Reference_Generation class

method generate_reference()
reference_generation.generate_reference()
end method

Documento de Requisitos

Serviço Iniciar Variáveis Globais (S10)

Descrição Geral

Funções do Produto

Iniciar variáveis globais de software.

Características do Usuário

Este serviço é consumido pelo Sistema de Atribuição de Dados.

Restrições Gerais

Não identificado.

Suposições e Dependências

Não identificado.

Requisitos Específicos

Requisitos Funcionais

RF1 - Inicializar e atribuir valores às seguintes variáveis globais.

- **kp**
- **ti**
- **td**
- **t**
- **currentError**
- **compensatorAction**
- **errorDefinition**
- **angleError**
- **distanceError**
- **averageSpeed**
- **speedNormalizer**
- **resultantError**
- **referenceValue**
- **interlockButton**
- **definedReference**

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - Não identificado.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Serviço Iniciar Variáveis Globais (S10)

Descrição de métodos

Atribuição de valores

De forma sequencial os seguintes valores devem ser atribuídos às suas variáveis:

- **kp** = 1.00
- **ti** = 0.1
- **td** = 0.01
- **t** = 0.1
- **currentError** = 0.00
- **compensatorAction** = 0.00
- **errorDefinition** = 0
- **angleError** = 0
- **distanceError** = 0
- **averageSpeed** = 50
- **speedNormalizer** = 20
- **resultantError** = 0
- **referenceValue** = 0
- **interlockButton** = 0
- **definedReference** = 0

Exemplo de uso e serviço

1. Após a execução do serviço os seguintes valores devem estar armazenados em suas respectivas variáveis.
   - **kp** = 1.00
   - **ti** = 0.1
   - **td** = 0.01
   - **t** = 0.1
   - **currentError** = 0.00
   - **compensatorAction** = 0.00
   - **errorDefinition** = 0
   - **angleError** = 0
   - **distanceError** = 0
   - **averageSpeed** = 50
   - **speedNormalizer** = 20
   - **resultantError** = 0
   - **referenceValue** = 0
   - **interlockButton** = 0
   - **definedReference** = 0

Referências

Histórico de alterações

Serviço Iniciar Variáveis Globais (S10)

Diagrama de Casos de Uso

Figura S10.1: Diagrama de Casos de Uso

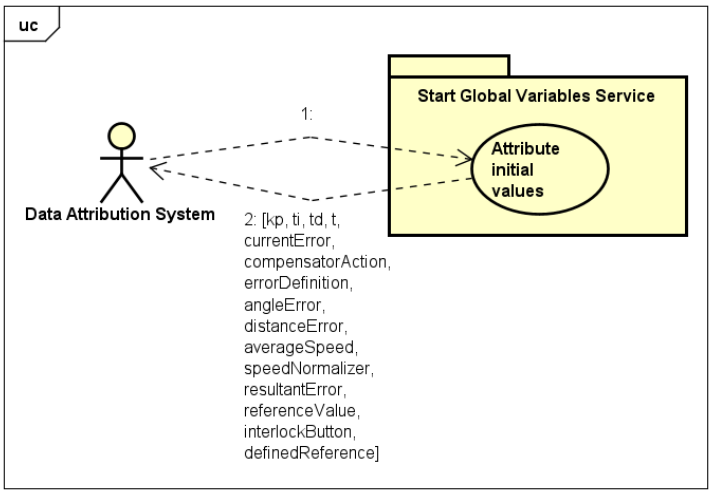


Diagrama de Atividades

Figura S10.2: Diagrama de Atividades

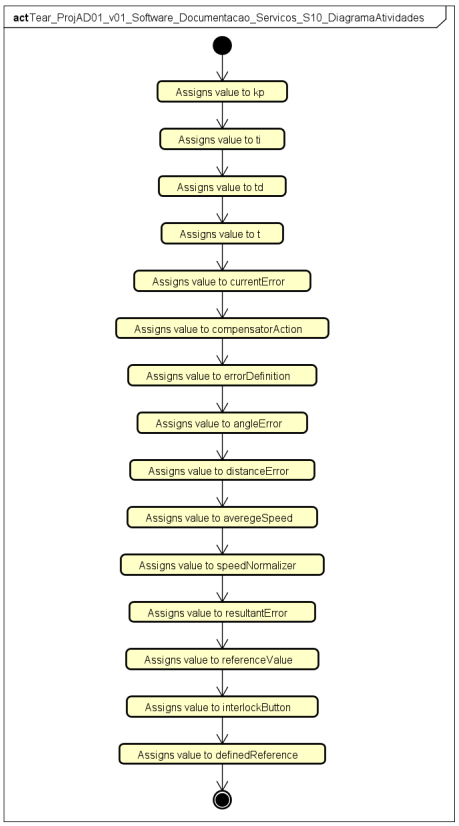
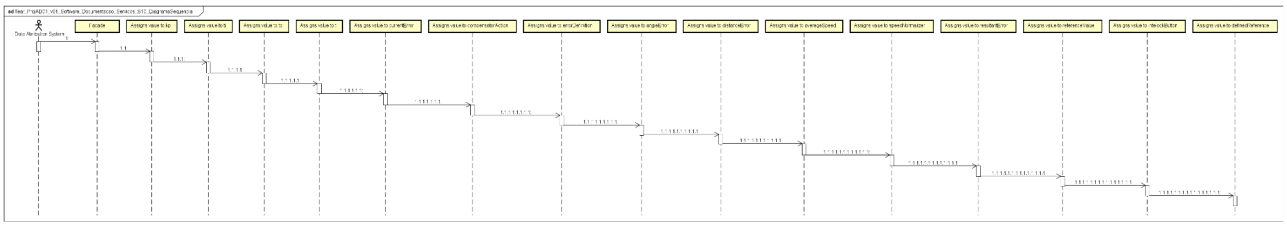


Diagrama de Sequência

Figura S10.3: Diagrama de Sequência



Pseudocódigo

Serviço Iniciar Variáveis Globais (S10)

Start_Global_Variables

include: NA
define: NA
variable:

float **kp** ← 1.00
float **ti** ← 0.1
float **td** ← 0.01
int **t** ← 0.1
float **currentError** ← 0.00
float **compensatorAction** ← 0.00
int **errorDefinition** ← 0
float **angleError** ← 0
float **distanceError** ← 0
float **averageSpeed** ← 50
float **speedNormalizer** ← 20
float **resultantError** ← 0
float **referenceValue** ← 0
bool **interlockButton** ← 0
float **definedReference** ← 0

Documento de Requisitos

Serviço de Condição Inicial de Hardware (S11)

Descrição Geral

Funções do Produto

Iniciar parâmetros de hardware do sistema.

Características do Usuário

Este serviço é consumido pelo Sistema de Atribuição de Dados.

Restrições Gerais

Não identificado.

Suposições e Dependências

Não identificado.

Requisitos Específicos

Requisitos Funcionais

RF1 - Inicializar e atribuir valores zero, às seguintes variáveis globais.

- **rightSpeed**
- **leftSpeed**
- **motorEnable**

Requisitos Não Funcionais

- **Desempenho**
  - Não identificado.
- **Escalabilidade**
  - Não identificado.
- **Confiabilidade**
  - Não identificado.
- **Manutenibilidade**
  - Não identificado.

Documento de Demanda de Produção

Serviço de Condição Inicial de Hardware (S11)

Descrição de métodos

Atribuição de valores

De forma sequencial os seguintes valores devem ser atribuídos às suas variáveis:

- **rightSpeed** = 0
- **leftSpeed** = 0
- **motorEnable** = 0

Exemplo de uso e serviço

1. Após a execução do serviço os seguintes valores devem estar armazenados em suas respectivas variáveis.
    - **rightSpeed** = 0
    - **leftSpeed** = 0
    - **motorEnable** = 0

Referências

Histórico de alterações

Serviço de Condição Inicial de Hardware (S11)

Diagrama de Casos de Uso

Figura S11.1: Diagrama de Casos de Uso

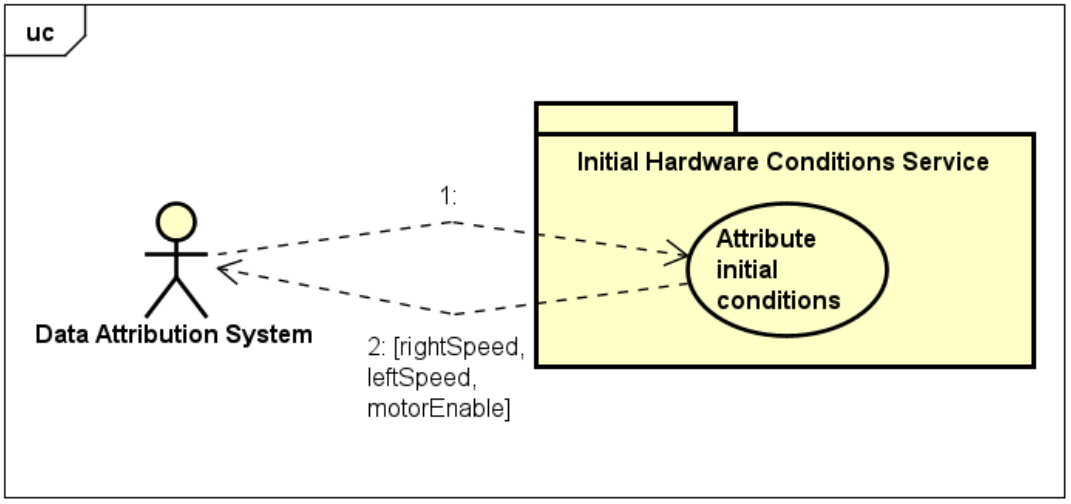


Diagrama de Atividades

Figura S11.2: Diagrama de Atividades

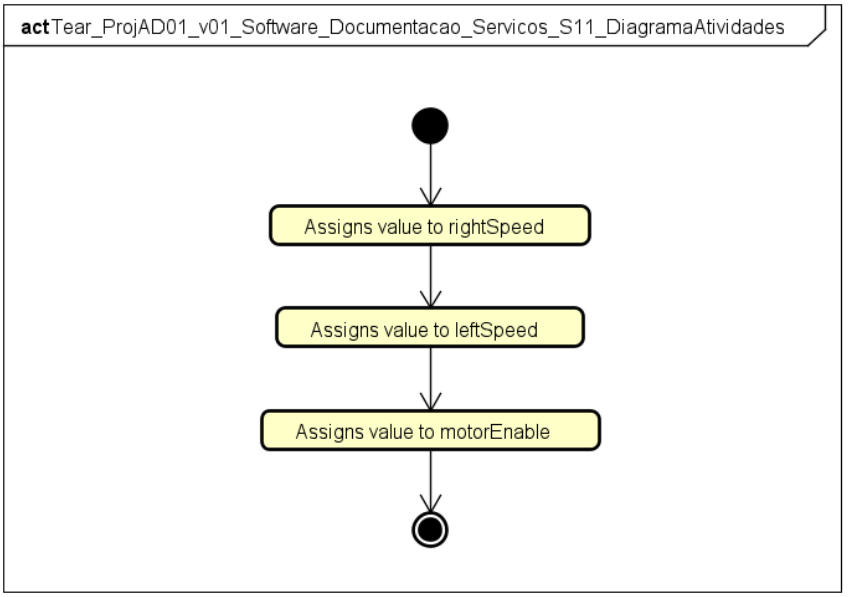
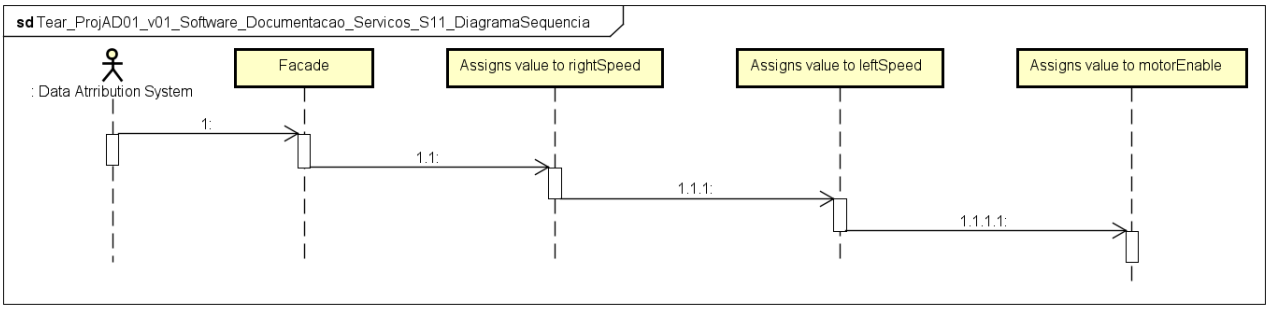


Diagrama de Sequência

Figura S11.3: Diagrama de Sequência



Pseudocódigo

Serviço de Condição Inicial de Hardware (S11)

Initial_Hardware_Condition

include: NA
define: NA
variable:

float **rightSpeed** ← 0
float **leftSpeed** ← 0
bool **motorEnable** ← 0