Federação das Indústrias do Estado da Bahia

CENTRO UNIVERSITÁRIO SENAI CIMATEC

Engenharia Elétrica

Projeto Theoprax de Conclusão de Curso

Desenvolvimento do robô de inspeção.

Apresentada por: Carlos Alberto Pereira
Cleber Couto Filho
Davi Costa
Ícaro Nascimento

Orientador: Prof. Marco Reis, M.Eng.

Setembro de 2018

Carlos Alberto Pereira
Cleber Couto Filho
Davi Costa
Ícaro Nascimento

Desenvolvimento do robô de inspeção.

Projeto Theoprax de Conclusão de Curso apresentada ao , Curso de Engenharia Elétrica do Centro Universitário SENAI CIMATEC, como requisito parcial para a obtenção do título de **Bacharel em Engenharia**.

Área de conhecimento: Interdisciplinar

Orientador: Prof. Marco Reis, M.Eng.

Salvador
Centro Universitário SENAI CIMATEC
2016

Resumo

Escreva aqui o resumo da dissertação, incluindo os contextos geral e específico, dentro dos quais a pesquisa foi realizada, o objetivo da pesquisa, assunção filosófica, os métodos de pesquisa usados e as possíveis contribuições que o que é proposto pode trazer à sociedade.

Palavras-chave: Palavra-chave 1, Palavra-chave 2, Palavra-chave 3, Palavra-chave 4, Palavra-chave 5

Abstract

Escreva aqui, em inglês, o resumo da dissertação, incluindo os contextos geral e específico, dentro dos quais a pesquisa foi realizada, o objetivo da pesquisa, assunção filosófica, os métodos de pesquisa usados e as possíveis contribuições que o que é proposto pode trazer à sociedade.

Keywords: Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5

Sumário

1 Introdução **1**
1.1 Objetivos . . . 3
1.1.1 Objetivos Específicos . . . 3
1.2 Justificativa . . . 4
1.3 Requisitos do cliente . . . 4
1.4 Organização do Projeto Theoprax de Conclusão de Curso . . . 4

2 Conceito do Sistema **6**
2.1 Estudo do estado da arte . . . 6
2.2 Descrição do sistema . . . 6
2.2.1 Especificação técnica . . . 6
2.2.2 Arquitetura geral do sistema . . . 6
2.2.2.1 Arquitetura do sistema de movimentação . . . 7
2.2.3 Arquitetura de software . . . 8
2.3 Desdobramento da função qualidade . . . 8
2.3.1 Requisitos técnicos . . . 8
2.3.1.1 Funcionamento do robô . . . 8
2.3.1.2 Códigos da programação disponíveis em repositório online 9
2.3.1.3 Documentação técnica de final de projeto . . . 10
2.3.1.4 Protótipo do robô . . . 10

3 Materiais e Métodos **11**
3.1 Especificação dos componentes . . . 11
3.1.1 Estrutura analítica do protótipo . . . 11
3.1.2 Lista de componentes . . . 11
3.2 Diagramas mecânicos . . . 11
3.3 Modelo esquemático de alimentação e comunicação . . . 11
3.3.1 Diagramas elétricos . . . 11
3.3.2 Esquemas eletrônicos . . . 12
3.4 Especificação das funcionalidades . . . 12
3.4.1 Fluxo das informações . . . 12
3.4.2 Motion Planning . . . 12
3.4.2.1 Definição da funcionalidade . . . 12
3.4.2.2 Dependências . . . 12
3.4.2.3 Premissas Necessárias . . . 13
3.4.2.4 Descrição da Funcionalidade . . . 13
3.4.2.5 Saídas . . . 14
3.4.3 Actuation . . . 15
3.4.3.1 Definição da funcionalidade . . . 15
3.4.3.2 Dependências . . . 15
3.4.3.3 Premissas Necessárias . . . 15
3.4.3.4 Descrição da Funcionalidade . . . 15
3.4.3.5 Saídas . . . 16
3.4.4 Power Management . . . 17
3.4.4.1 Definição da funcionalidade . . . 17

3.4.4.2 Dependências . . . 17
3.4.4.3 Premissas Necessárias . . . 17
3.4.4.4 Descrição da Funcionalidade . . . 17
3.4.4.5 Saídas . . . 19
3.4.5 System Integrity Check . . . 19
3.4.5.1 Definição da funcionalidade . . . 19
3.4.5.2 Dependências . . . 20
3.4.5.3 Premissas Necessárias . . . 20
3.4.5.4 Descrição da Funcionalidade . . . 20
3.4.5.5 Saídas . . . 21
3.5 Interface do Usuário . . . 22
3.6 Simulação do sistema . . . 22

4 Resultados **23**
4.1 Testes unitários . . . 23
4.1.1 Configuração dos servomotores . . . 23
4.1.1.1 Utilização do software Mixcell . . . 23
4.1.1.2 Modos Mestre/Escravo . . . 24
4.1.1.3 ID, Baud rate e Modos de operação . . . 24
4.1.2 Controle dos servomotores utilizando a biblioteca dynamixel driver 24
4.1.2.1 Teste dos controladores de posição . . . 25
4.1.2.2 Teste dos Controladores de velocidade . . . 25
4.1.2.3 Teste Controladorcontrolador de trajetoria . . . 25
4.1.2.4 Compatibilidade com diferentes protocolos de comunicação 25
4.1.3 Controle dos servomotores utilizando a biblioteca dynamixel workbench . . . 26
4.1.3.1 Controladores de posição . . . 26
4.1.3.2 Controladores de velocidade . . . 26
4.1.3.3 Controladores de trajetória . . . 26
4.1.3.4 Comunicação em rede . . . 27
4.1.4 Compatibilidade do Elir Robot com o MoveIt! . . . 27
4.1.4.1 Teste de Estado de Colisão . . . 27
4.1.4.2 Teste de Definição da corrente-cinemática e end-effector . 27
4.1.5 Robô de testes Davictory . . . 28
4.1.5.1 Teste com diferentes plugins de cinematica . . . 28
4.1.6 Cálculo da cinemática inversa . . . 29
4.1.7 Power Management . . . 29
4.1.7.1 Gravação e validação de Firmware na placa de Power Management . . . 29
4.1.8 Acesso remoto a nuc . . . 30
4.2 Testes integrados . . . 30
4.3 Avaliação da prontidão tecnológica . . . 30
4.4 Trabalhos futuros . . . 30

5 Conclusão **31**
5.1 Considerações finais . . . 31

A QFD **32**

B Diagramas mecânicos **33**

C Diagramas eletro-eletrônicos **34**

D Wireframes **35**

E Logbook **36**

Referências **37**

Lista de Tabelas

Lista de Figuras

1.1 Inspeção de linhas de transmissão feita por aeronaves tripuladas. 2
1.2 Interação humana durante a inspeção de linhas de transmissão. 2
1.3 Realização de inspeção em linhas de transmissão através da observação humana. 3

2.1 Arquitetura Geral do sistema de movimentação 7

3.1 Fluxograma de funcionamento da funcionalidade de Motion Planning . . . 14
3.2 Fluxograma da funcionalidade Actuation 16
3.3 Fluxograma de funcionamento da funcionalidade de Power Management . . 18
3.4 Fluxograma da rotina para checagem do sistema 21

Lista de Siglas

THEOPRAX
WWW ..... World Wide Web

Lista de Simbolos

∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble
∂ .... Bla bla bla
$\prod$ .... ble ble ble

Capítulo Um

Introdução

No Brasil, a eletricidade é gerada por hidrelétricas, termoelétricas, parques eólicos e usinas nucleares. Na maioria dos casos, devido a condições geográficas e de segurança, a energia gerada nem sempre é utilizada ou consumida no local de sua geração. Portanto, há a necessidade do uso de linhas de transmissão para transportar energia gerada na fonte geradora para a carga do consumidor (**??**). O mercado consumidor brasileiro é composto de cerca de 47 milhões de unidades. Em termos de linhas de transmissão de energia, são cerca de 98.648,3 km, que devem estar operando 24 horas por dia, 7 dias por semana, 365 dias por ano e em perfeito estado de manutenção, para garantir eletricidade para os consumidores (**??**)

No Brasil, há uma quantidade considerável de linhas de transmissão de alta tensão que já ultrapassaram a vida útil as quais foram destinadas. Com o envelhecimento dos cabos, a inspeção para manutenção preventiva é um fator de extrema relevância para garantir o perfeito funcionamento dos sistemas elétricos. De um modo geral, as inspeções nas linhas de transmissão de alta tensão são realizadas regularmente de forma visual, a fim de identificar a necessidade da realização de manutenções preventivas. As inspeções buscam verificar a integridade física dos componentes das linhas, em termos de fissuras, corrosão e eventuais danos que venham a prejudicar o fornecimento de energia elétrica. Essas inspeções envolvem a análise da integridade estrutural das torres, da condição dos isoladores, das conexões das linhas de transmissão, dentre outros, a fim de se verificar a existência de eventuais pontos de ruptura.

Um dos métodos empregados para detecção de pontos quentes nos cabos é o imageamento térmico, que é capaz de identificar uma elevação de temperatura nos cabos, o que é um indício de possíveis pontos de ruptura. A inspeção através de câmera térmica é uma importante ferramenta no campo das inspeções para manutenções preventivas. Outros pontos a serem inspecionados envolvem as condições do local onde as torres são instaladas, pois a vegetação e eventuais construções devem ser mantidas a uma distância mínima segura, tal que não ocorra nenhum contato entre quaisquer estruturas e as torres ou cabos de transmissão, evitando assim interferências no funcionamento da linha.

Além disso, é essencial a garantia de dispor-se de um terreno em condições de trânsito de veículos para o transporte do pessoal de manutenção, transporte de ferramentas, dentre outros fatores. Durante vários anos, a inspeção de linhas de transmissão de alta tensão tem sido feita regularmente através de aeronaves tripuladas. As aeronaves executam vôos

em baixa altitude e muito próximos das linhas de transmissão conforme mostrado nas Figuras 1.1 e 1.2.

Figura 1.1: Inspeção de linhas de transmissão feita por aeronaves tripuladas.

Figura 1.2: Interação humana durante a inspeção de linhas de transmissão.

Em alguns casos, devido às características geográficas da região, condições climáticas e outros fatores que venham a dificultar o sobrevôo, há uma grande exposição dos tripulantes a riscos associados à tarefa. Além dos perigos aos quais os tripulantes são expostos, a inspeção feita com aeronaves tem um custo bastante elevado. Outra forma alternativa de inspeção é o uso de veículos terrestres, porém essa forma é muito limitada, pois boa parte das linhas de transmissão está localizada em áreas de difícil acesso terrestre, muitas vezes restritas pelas características geográficas da região. Além disso, o ângulo de visão é, muitas vezes, desfavorável para a realização da inspeção.

Outra maneira de inspecionar as linhas de transmissão é através de eletricistas que literalmente caminham sobre os cabos de linhas de transmissão de alta tensão (Figura

Figura 1.3: Realização de inspeção em linhas de transmissão através da observação humana.

1.3), realizando inspeção visual e termográfica. Esse tipo de inspeção é lenta e não é viável, tendo em vista que o país possui milhares de quilômetros de linhas de transmissão.

Neste contexto vários robôs de inspeção de linhas de transmissão foram desenvolvidos, porém poucos deles consistiram em projetos de engenharia que sejam aplicáveis no mundo real, além disso a maioria eram robôs tele-operados, ou seja robôs controlados por seres humanos. Um dos pontos diferenciais deste projeto de tese é a proposição de um desenvolvimento de uma navegação autônoma utilizando técnicas de aprendizagem de máquinas até então não utilizadas em robôs de inspeção de linhas de transmissão de alta tensão.

1.1 Objetivos

Desenvolver um protótipo de um robô de inspeção de linhas de transmissão

1.1.1 Objetivos Específicos

Desenvolver 4 funcionalidades para o robô, sendo elas:

- Actuation

  Funcionalidade responsável por atuar os motores que movimentam o robô.

- Motion Planning

Tem como objetivo planejar a movimentação das partes do robô

- Power Management

  Responsável pelo gerenciamento e distribuição de energia para os componentes elétricos e eletrônicos do robô.

- System Integrity Check

  Checagem da integridade do sistema antes do robô entrar em funcionamento.

  content...

1.2 Justificativa

O pesquisador/estudante deve apresentar os aspectos mais relevantes da pesquisa ressaltando os impactos (e.g. científico, tecnológico, econômico, social e ambiental) que a pesquisa causará. Deve-se ter cuidado com a ingenuidade no momento em que os argumentos forem apresentados.

1.3 Requisitos do cliente

Foi especificado pelo cliente que o robô ELIR realize tais funções:

- Transpor obstáculos e cadeia de isoladores;
- Deslocar-se através do consumo de baterias;
- Deslocamento/movimento realizada por servomotores;
- Realizar as funções de forma autônoma.

1.4 Organização do Projeto Theoprax de Conclusão de Curso

Este documento apresenta x capítulos e está estruturado da seguinte forma:

- **Capítulo 1 - Introdução**: Contextualiza o âmbito, no qual a pesquisa proposta está inserida. Apresenta, portanto, a definição do problema, objetivos e justificativas da pesquisa e como este projeto theoprax de conclusão de curso está estruturado;

- **Capítulo 2 - Conceito do Sistema**: Conceitua o sistema por meio de diagramas que representam as arquiteturas do robô em diferentes níveis de abstração, abordando o estudo do estado da arte, desdobramento dos requisitos de qualidade e funcionamento do projeto. ;
- **Capítulo 3 - Materiais e Métodos**: Mostra os materiais e métodos que foram utilizados durante o projeto, contendo a especificação de componentes, sendo eles softwares e dispositivos, assim como os esquemas elétricos e eletrônicos - e a descrição das funcionalidades;
- **Capítulo 4 - Resultados**: Exibe os resultados obtidos durante a confecção do projeto, apresentando os testes unitários e integrados, assim como as datas e quem o efetuou;
- **Capítulo 5 - Conclusão**: Apresenta as conclusóes, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.

Capítulo Dois

Conceito do Sistema

Quanto maior for a rapidez de transformação de uma sociedade, mais temporárias são as necessidades individuais. Essas flutuaçõess tornam ainda mais acelerado o senso de turbilh da sociedade.

(Alvin Toffler)

Quanto maior for a rapidez de transformação de uma sociedade, mais temporárias são as necessidades individuais. Essas flutuações tornam ainda mais acelerado o senso de turbilhão da sociedade.

(Alvin Toffler)

2.1 Estudo do estado da arte

flkjasdlkfjasdlkfjs

2.2 Descrição do sistema

lasdjflsadjf

2.2.1 Especificação técnica

lakjfldksjfdslakjf

2.2.2 Arquitetura geral do sistema

lkasjdflksdajflk;

2.2.2.1 Arquitetura do sistema de movimentação

De forma a garantir uma movimentação efetiva do robô é necessária a integração de diversos ferramentas físicas e de software, como a estrutura de movimentação adequada, sistema ordenamento de missão, controle de potência,demandando assim um *framework* e um sistema operacional.

A inspeção de linha foi denominada missão, para cade vez que o robô começar a realizar a inspeção, será considerado o início de uma nova missão.

Para garantir a execução correta da missão e ultrapassagem dos obstáculos de forma efetiva, se dividiu o sistema em 4 principais subsistemas, sendo elas : *Motion Planning* , *Actuation*,*System Integrity Check* e *Power Management*. A arquitetura geral do sistema de movimentação está mostrada na figura 2.1, ilustrando os subsistemas e suas funcionalidades. A estrutura física do robô foi projetada para que sejam realizados movimentos

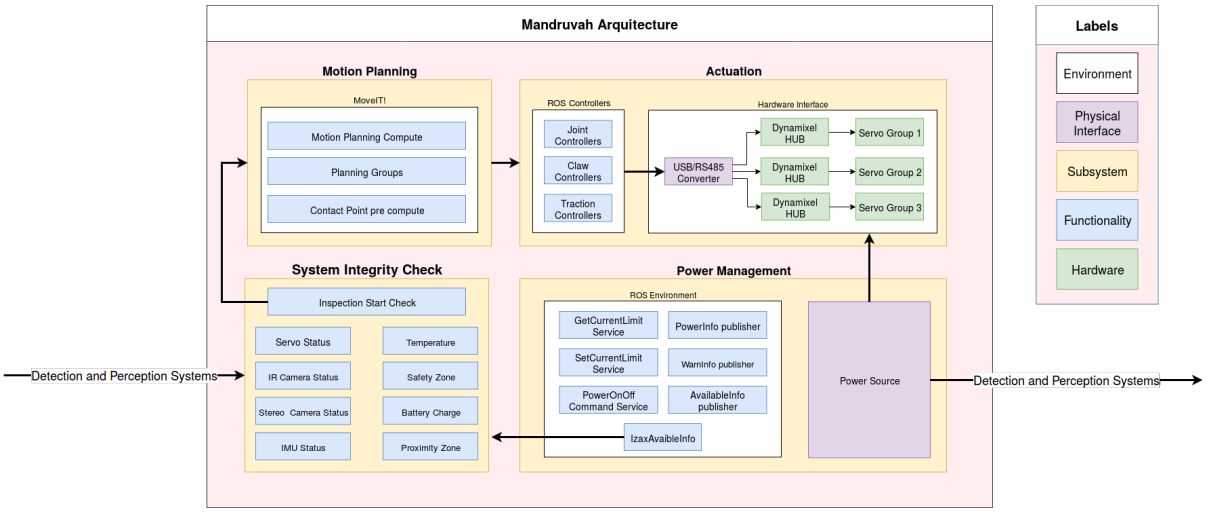

Figura 2.1: Arquitetura Geral do sistema de movimentação

Fonte: Própria

de translação e transposição de obstáculos presentes na linha de transmissão,consistindo de unidades de tração para a translação na linha e juntas nos braços e garras para a realização da ultrapassagem de obstáculos. O controle da estrutura física do robô está relacionado com a *Actuation*. A transposição dos obstáculos é um grande desafio para essa aplicação, visto que será necessário a aplicação da cinemática inversa no robô. A cinemática inversa consiste num conjunto de equações que definem o movimento do robô para a movimentação de um ponto à outro, tal modelo é extraído à partir da estrutura do robô. A funcionalidade responsável por calcular esse modelo e encontrar como será feita a movimentação foi denominada *Motion Planning*. Para garantir a execução correta da missão e preservar a integridade do robô foi estipulada uma funcionalidade que checa os dispositivos antes de cada missão, denominada *System Integrity Check*. E com a finali-

dade de realizar o controle da potência no robô foi será utilizado o projeto de uma placa específica para esse papel, assim todos os aspectos relacionados à alimentação do robô, assim como consumo e monitoramento estão atrelados ao *Power Management*.

O *framework* ROS possibilita a integração de todas essas funcionalidades, sua estrutura baseada em nós facilita a identificação dos problemas e possibilita a modularização do código. Fornecendo também diversas ferramentas como o *MoveIt!*, que será utilizada para o *Motion Planning*, assim como drivers de compatibilização para os servo motores adotados no projeto.

2.2.3 Arquitetura de software

2.3 Desdobramento da função qualidade

asdfsdafsf

2.3.1 Requisitos técnicos

Foi determinado pelo cliente os seguintes requisitos técnicos.

- Desempenho de deslocamento: Percorrer 15km por dia
- Velocidade de deslocamento: Velocidade média sem obstáculos será de 0.5 m/s
- Ultrapassagem de obstáculos: Volume máximo dos obstáculos 410x330x150mm
- Autonomia de Potência: 2 horas de autonomia
- Sistema Operacional: Linux
- Backend: C++ e Python
- Framework: ROS Kinetic Kame

2.3.1.1 Funcionamento do robô

O funcionamento do robô será comprovado por um teste realizado dentro da instituição, em um modelo reduzido de linha de transmissão. Sem condições ambientais adversas, realizando a parada baseada no sinal da câmera de detecção de obstáculos e

com o teste operado manualmente. Serão desenvolvidas rotinas de software para: início da missão; simular detecção de obstáculo e parada emergencial. As especificações para os testes são:

- Condutor: LINNET e diâmetro: 18,3mm;
- Obstáculos: Grampo de suspensão e amortecedor de vibração;
- O robô será colocado manualmente na linha;
- A operação se iniciará à uma distância de 1 metro do obstáculo;
- A parada será realizada com base no sinal do sistema de detecção, a uma distância de 50cm do obstáculo;
- O comando para início da missão será feito por meio do terminal do Linux , por meio de acesso remoto;
- A estrutura para o teste será fornecida pela a empresa;

As etapas para realização do teste são:

- O robô será manualmente posicionado na linha, à uma distância de 1 metro do obstáculo;
- O comando para iniciar a inspeção será enviado via acesso remoto, por meio do terminal Linux;
- Após receber o comando, o robô iniciará um movimento na linha em direção ao obstáculo;
- Ao receber o sinal de obstáculo detectado, o robô irá parar e esperar o pós-processamento do sistema de detecção;
- Após o processamento, irá fazer a ultrapassagem referente ao tipo de obstáculo, e continuar se deslocando na linha;

2.3.1.2 Códigos da programação disponíveis em repositório online

Os códigos produzidos serão disponibilizados na plataforma GitHub, onde os pacotes produzidos durante o projeto estão organizados em 4 repositórios referentes às funcionalidades do robô.

2.3.1.3 Documentação técnica de final de projeto

A documentação foi definida como um relatório denominado Conceptual and Design Report , Databooks com as informações e logbooks com os testes.

2.3.1.4 Protótipo do robô

O cliente disponibilizou as partes mecânicas do robô, sendo entregue pela equipe o protótipo do robô montado. É incluída na montagem a disposição dos cabos e unidade de processamento do robô.

Capítulo Três

Materiais e Métodos

asdfasdfsdf

3.1 Especificação dos componentes

asjdflkdjsaf

3.1.1 Estrutura analítica do protótipo

asdkjfsdalkjf

3.1.2 Lista de componentes

asfkjdsahfkjs

3.2 Diagramas mecânicos

asdfsdaf

3.3 Modelo esquemático de alimentação e comunicação

asdfadsfsdfs

3.3.1 Diagramas elétricos

asdfsdaf

3.3.2 Esquemas eletrônicos

asdfsdaf

3.4 Especificação das funcionalidades

asdfadsfsdfs

3.4.1 Fluxo das informações

asdfsaf

3.4.2 Motion Planning

3.4.2.1 Definição da funcionalidade

A funcionalidade de *Motion Planning* é responsável por realizar o planejamento da trajetória do Robô, utilizando o software *MoveIt!* que realiza o cálculo da cinemática inversa para encontrar a melhor forma de ultrapassar os obstáculos.

3.4.2.2 Dependências

O software moveit pode utilizar o modelo matemático da cinemática inversa do robô ou um arquivo do tipo URDF. O nome URDF é uma sigla para *Unified Robot Description Format*, esse arquivo é uma especificação em `XML` utilizada para descrever robôs. Modelos em URDF apresentam uma simplicidade na descrição do robô, e para o caso do Robô *Elir*, utilizar o modelo URDF possibilitará uma aproximação fiel ao modelo real do robô, assim para o cálculo da cinemática inversa será utilizado o seu modelo URDF e não o seu modelo matemático.

3.4.2.3 Premissas Necessárias

Para o correto funcionamento dessa funcionalidade as seguintes premissas são necessárias:

- A configuração dos limites de giro das juntas do robô estarão compatíveis com os comandos enviados
- O modelo URDF do robô estará adequado com o modelo físico
- O pacote gerado pelo *MoveIt! Setup Assistant* estará configurado adequadamente

3.4.2.4 Descrição da Funcionalidade

A movimentação do robô na linha acontecerá por movimentos de translação e transposição de obstáculos. A translação na linha será feita por controladores de torque nas rodas do robô, enquanto a transposição do obstáculos utilizará o moveit. Por meio da ferramenta *MoveIt! Setup Assistant*, se utiliza o modelo do robô para criar um pacote do ROS com os principais arquivos pelo moveit. A configuração correta do moveit possibilita que se utilizem as funções da sua biblioteca para o cálculo da trajetória, levando em consideração também obstáculos no caminho.

O moveit fornece uma *user interface* que recebe o end-effector,a nomenclatura atribuída ao node feito em python que recebe o *end-effector* é `moveit_commander`. O *node* responsável por fazer a integração da user interface com os parâmetros recebidos pelo *ROS Parameter Server* com o *end-effector* para fazer os cálculos é denominado `move_group`. O *node* `move_group` também pode receber parâmetros como leituras dos sensores do robô e nuvens de pontos.

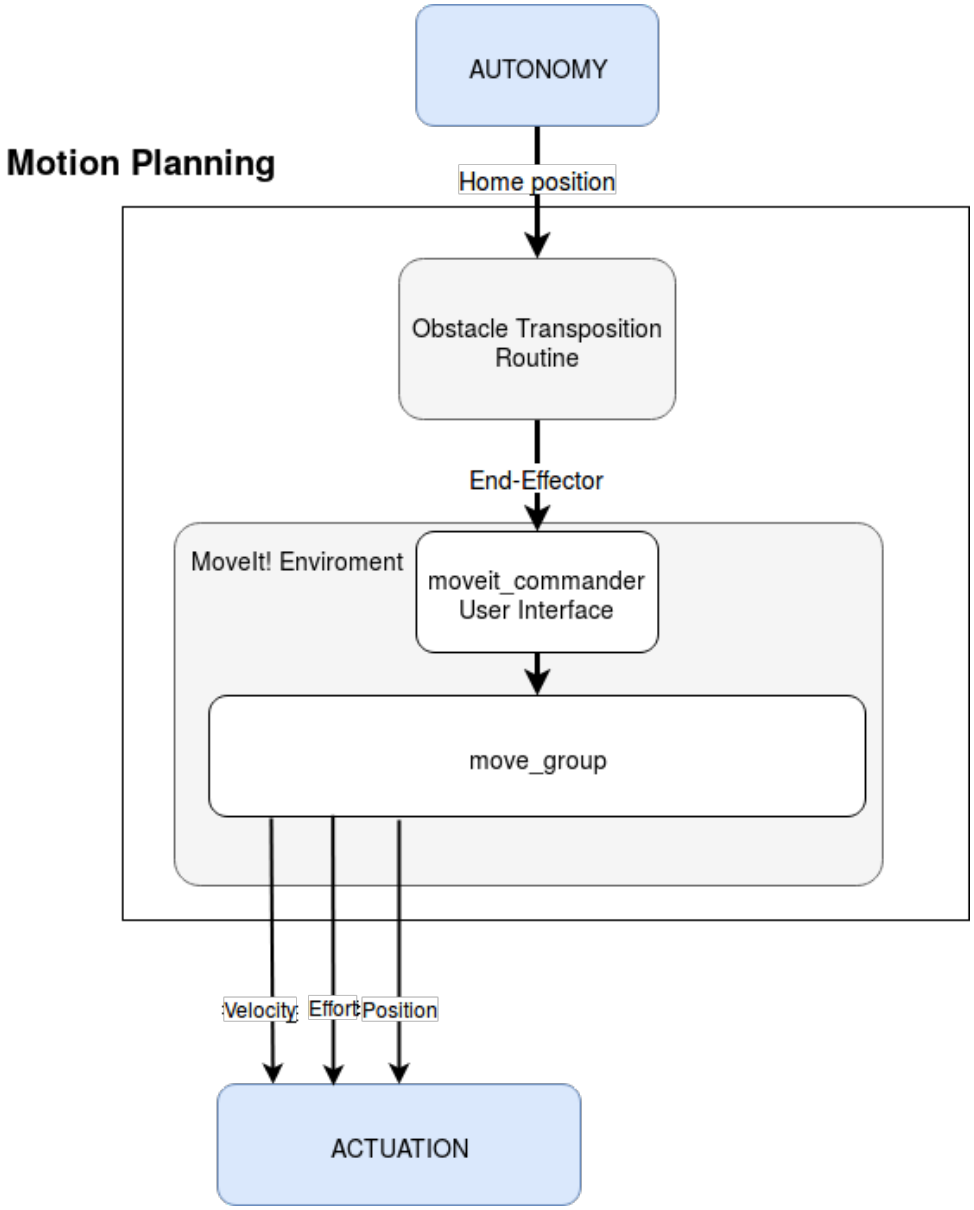

Figura 3.1: Fluxograma de funcionamento da funcionalidade de Motion Planning

Fonte: Própria

3.4.2.5 Saídas

Por meio da compatibilização do *MoveIt!* com o *ROS*, a saída dessa funcionalidade são os comandos de velocidade, esforço e posição para cada junta do robô.

3.4.3 Actuation

3.4.3.1 Definição da funcionalidade

A funcionalidade de Actuation tem como objetivo mover a estrutura física do robô, possibilitando o controle dos movimentos das juntas, garras e unidades de tração.

3.4.3.2 Dependências

Essa funcionalidade depende das funcionalidades de *Power Management* e *Motion Planning*. O *Power Management* será responsável por fazer alimentação dos motores, possibilitando controlar a corrente máxima fornecida para cada grupo. A dependência em relação à funcionalidade de *Motion Planning* está atrelada principalmente com o software *MoveIt!*, que ao receber um *end-effector*,realiza o cálculo de trajetória e envia os comandos de velocidade, esforço e posição para os controladores das juntas, garras e unidades de tração.

3.4.3.3 Premissas Necessárias

Para o correto funcionamento desse módulo, devem ser consideradas as seguintes premissas:

- Os motores devem estar configurados de acordo com o padrão de ID determinado pela equipe, fazendo parte da mesma malha de controle;
- Os controladores das juntas,garras e unidades devem estar configurados de acordo com os comandos que serão recebidos pelo MoveIt!;
- Os 3 grupos de motores estarão em malhas de alimentação de 12V individuais.

3.4.3.4 Descrição da Funcionalidade

O ROS disponibiliza uma série de drivers para compatibilização dos motores dynamixel, possibilitando a criação de controladores específicos no seu ambiente. Serão criados os controladores referentes as juntas e unidades de tração do robô.Os controladores receberão comandos de *velocity* e *position* do *MoveIt!* junto com os comandos para

movimentar o robô na linha. Após os comandos serem recebidos pelos controladores, eles serão enviados para o *hardware* do robô, de acordo do padrão de comunicação dos motores, por meio de comunicação serial.

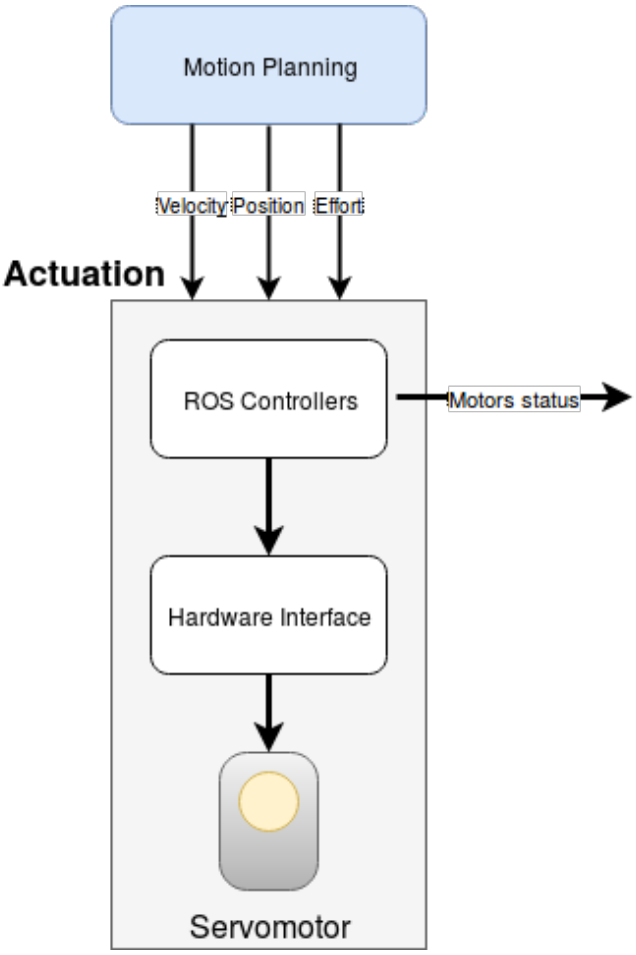

Figura 3.2: Fluxograma da funcionalidade Actuation

Fonte: Própria

3.4.3.5 Saídas

A saída desta funcionalidade é o movimento da estrutura física do robô, que estará de acordo com o planejamento de trajetória do *MoveIt!* e com as instruções para operação na linha

3.4.4 Power Management

3.4.4.1 Definição da funcionalidade

A funcionalidade de *Power Management* é responsável pelo gerenciamento de alimentação elétrica dos componentes elétricos e eletrônicos do robô, através da integração das funcionalidades de seu firmware no ambiente ROS.

3.4.4.2 Dependências

Essa funcionalidade depende da comunicação serial por meio da biblioteca *rosserial* para compatibilização e integração das funcionalidades de firmware no ambiente ROS. Operacionalização e customização do firmware embarcado no hardware de acordo com as necessidades do projeto e da alimentação fornecida pela placa multiplexadora, por meio de baterias Li-Ion NH2054 14.4 volts.

3.4.4.3 Premissas Necessárias

Para o correto funcionamento desse módulo de *Power Management*, devem ser consideradas as seguintes premissas:

- A placa multiplexadora estará conectada diretamente ao módulo de *Power Management*
- Todos os dispositivos estarão conectados nas suas respectivas entradas
- A placa deverá ser alimentada por 2 baterias de 14.4 Volts e 3 Amperes, totalizando um fornecimento de até 6 Amperes
- A placa estará conectada diretamente na NUC, por meio de uma USB

3.4.4.4 Descrição da Funcionalidade

A funcionalidade *Power Management* é responsável fornece diversos recursos em sua totalidade. O hardware utilizado (placa Zord) possui um sensor de corrente e tensão para cada porta de saída, permitindo o monitoramento individual de cada uma das portas.

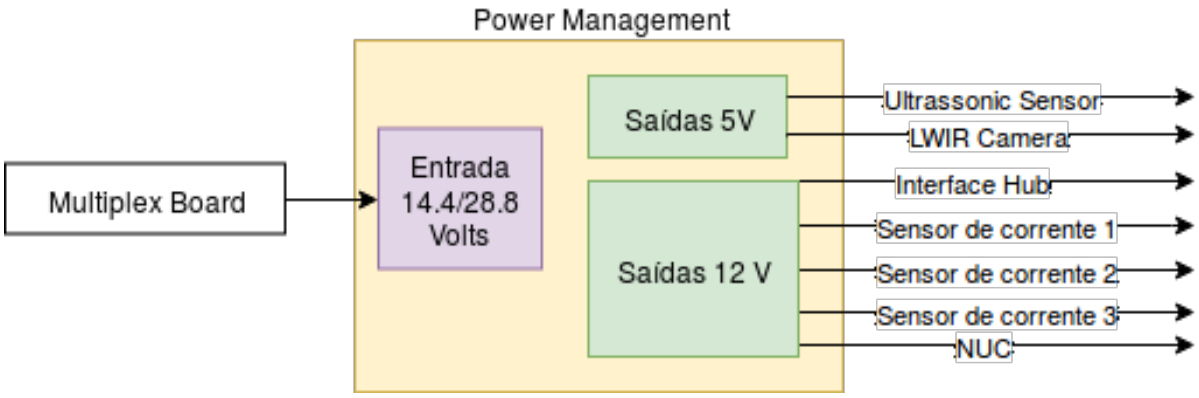

Figura 3.3: Fluxograma de funcionamento da funcionalidade de Power Management

Fonte: Própria

O microcontrolador utilizado Atmega32U4 possui um firmware embarcado onde toda a compatibilização com o ambiente ROS é realizada, o que torna essencial o uso do pacote rosserial para o seu funcionamento. O firmware é responsável pela ativação dos relés digitais em caso de surtos de corrente para proteção dos dispositivos elétricos. Os limites nos valores de corrente funcionam justamente para que o hardware interrompa a alimentação em um possível caso de surto de corrente. Todos os aspectos importantes para o funcionamento do sistema de gerenciamento de energia pode ser configurado tanto via ROS, por meio das configurações dos serviços, ou por meio do firmware, modificando os parâmetros do tempo de duração dos picos de corrente. Os principais serviços e tópicos criados pela funcionalidade Power Management no ROS são:

- *Tópicos*
  - *PowerOutput* Este tópico disponibiliza os valores de tensão e corrente de todas as portas da placa em tempo real.
  - *TakeStatus* Disponibiliza o estado de cada porta da placa, informando os eventos ocorridos e a porcentagem de corrente demandada durante a ocorrência do evento.
- *Serviços*
  - *GetCurrentLimitCommand* Este comando retorna o valor de corrente máxima de saída configurado para a porta escolhida
  - *SetCurrentLimitCommand* Este comando realiza a configuração do valor máximo de corrente de saída em uma determinada porta
  - *PowerOnOffCommand* Este comando realiza a ação de ativação ou desligamento de uma determinada porta.

A placa de Gerenciamento de energia irá receber a carga das baterias pela placa multiplexadora e irá realiza o controle de alimentação dos seguintes componentes:

- Grupos de servo motores
- Grupo de sensores de corrente
- NUC
- Interface HUB
- Câmera LWIR
- Sensor ultrassônico
- Phidgets
- STM Nucleo
- Módulo GPS

3.4.4.5 Saídas

A funcionalidade irá disponibilizar a energia para o robô e as seguintes estruturas no ambiente ROS:

- Tópicos com informações de tensão e corrente nas portas
- Tópico para aviso de sobre-corrente
- Tópico para informar disponibilidade da placa
- Serviços para ler e configurar limite de corrente das portas
- Serviço para ligar ou desligar energia em uma porta

3.4.5 System Integrity Check

3.4.5.1 Definição da funcionalidade

É a funcionalidade responsável por checar a integridade do sistema antes do início da missão, verificando os subsistemas e suas variáveis.

3.4.5.2 Dependências

A funcionalidade receberá informações dos seguintes componentes

- Sensor de Temperatura
- Servomotores
- Câmera IR
- Câmera Stéreo
- IMU
- Sensor de Proximidade
- Placa de Power Management
- Sonar
- Baterias

Todas as informações serão enviadas por meio do ambiente ROS, na forma de *Services* ou *Publishers*.

3.4.5.3 Premissas Necessárias

As premissas necessárias para o funcionamento dessa funcionalidade são:

- Os subsistemas do robô irão disponibilizar o seu status no ambiente ROS por meio de tópicos ou serviços
- A checagem fará parte do planejamento de missão

3.4.5.4 Descrição da Funcionalidade

A checagem da integridade do sistema é uma funcionalidade essencial para garantir o sucesso da missão e preservar a integridade do robô. O ROS facilita essa comunicação entre os subsistemas, possibilitando que seja criada uma rotina de checagem antes de cada missão.

Será disponibilizado no sistema uma rotina para iniciar a missão. Ao receber o comando para início de missão, os sistemas serão checados sequencialmente, utilizando estrutura de *Services* e *Publishers* do ROS. Caso algum sistema apresente falha, a missão não se iniciará e o erro será mostrado no *terminal* e registrado no arquivo de `log`. Se todos os sistemas estiverem em funcionamento, se iniciará a missão. O fluxograma da funcionalidade está ilustrado na figura 3.4.

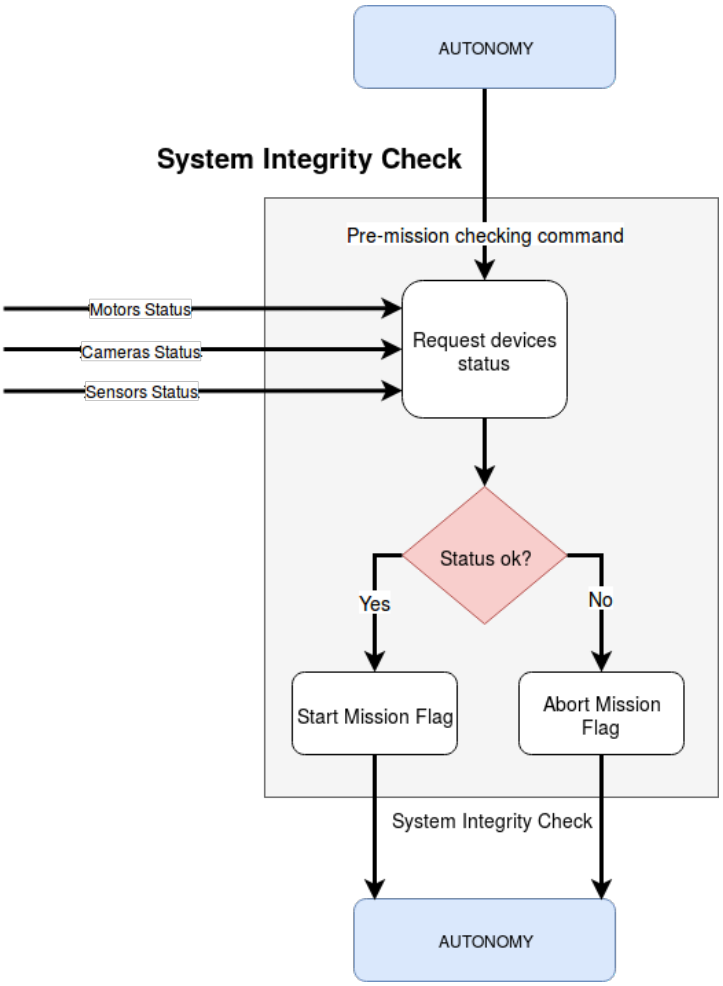

Figura 3.4: Fluxograma da rotina para checagem do sistema

Fonte: Própria

3.4.5.5 Saídas

No início da rotina de inspeção, a funcionalidade será responsável por enviar o sinal inicia a missão. Caso todos os sistemas checados estejam funcionando, a inspeção ocorrerá normalmente, se algum sistema apresentar defeitos, o defeito será mostrado no *terminal*, registrado em log e a missão será abortada.

3.5 Interface do Usuário

asdfadsfsdfs

3.6 Simulação do sistema

asdfadsfsdfs

Capítulo Quatro

Resultados

A elaboração de cronogramas, e a organização das atividades em pacotes com metas de curto médio e longo prazo contribuiu para que o andamento do projeto ocorresse de forma satisfatória. Os prazos foram mudados durante o projeto e os atrasos gerados por problemas inesperados conseguiram ser compensados com uma mudança na forma de gerir os integrantes e a inserção de tarefas em paralelo, sendo elas individuais e em subgrupos

4.1 Testes unitários

Consistem nos testes individuais das ferramentas, tem suma importância para atestar se as ferramentas pré determinadas funcionam de acordo com o esperado.

4.1.1 Configuração dos servomotores

Os primeiros testes realizados no início do desenvolvimento do robô foram a configuração dos servomotores, etapa essencial para a definição de como os motores deveriam trabalhar em conjunto. A configuração se deu utilizando diversos softwares, e após as configurações realizadas, os motores eram analisados para validar se a configuração fora realizada de maneira correta.

4.1.1.1 Utilização do software Mixcell

O software Mixcell, desenvolvido por integrantes do grupo Mandruvah, disponível para utilização gratuita na Ros Wiki, possui a funcionalidade de realizar a configuração de parâmetros em motores Dynamixel de diversos modelos, tais como ID dos motores, modos de operação, baudrate para comunicação serial. O software foi utilizado em toda as etapas de testes dos motores, sendo fundamental para determinar o funcionamento de cada um dos motores e para o desenvolvimento e montagem do robô.

FALTA IMAGEM

4.1.1.2 Modos Mestre/Escravo

Durante os testes em modo Mestre/Escravo dos motores Dynamixel, foi observado que mesmo quando dois motores são configurados para trabalharem como mestre/escravo via software, é necessário que se haja a utilização do cabo de sincronização entre os dois motores. Foi constatado durante os testes que o cabo de sincronização transmite a informação de carga para que haja a compensação entre os motores quando trabalham em conjunto.

4.1.1.3 ID, Baud rate e Modos de operação

Durante toda a fase de testes foram observados parâmetros extremamente importantes para definição e funcionamento dos motores Dynamixel. Os IDs dos motores definem basicamente a sua identificação para as configurações de juntas, associando assim os motores as suas respectivas juntas. O baud rate é a taxa de transmissão de dados em bits por segundo entre o computador e os motores. Percebeu-se que durante os testes, para um número grande de motores trabalhando em conjunto, era necessário um valor de baudrate mais elevado para que todos os motores fossem encontrados. Os modos de operação dos motores são parâmetros que definem como os mesmos vão se comportar durante a operação, sendo eles os modos junta e roda. No primeiro modo é necessário estabelecer um limite máximo e mínimo para o giro dos motores, além da definição de qual motor será o mestre ou o escravo durante a operação (caso haja mais de um motor na junta), enquanto que no modo roda o motor irá girar livremente, mudando somente o sentido de giro.

4.1.2 Controle dos servomotores utilizando a biblioteca dynamixel driver

A biblioteca dynamixel driver foi a utilizada nesse projeto e tal, para fazer a configurações , criar os controladores, possibilitar a integração do motor no sistema ros. Essa biblioteca foi descontinuada na versão utilizada no projeto, o que fez com se mostrassem necessários testes extras. Este teste foi realizado por TALTALTALTAL, durante a fase inicial do projeto que se deu entre tal DIA até TAL DIA.

4.1.2.1 Teste dos controladores de posição

Foi realizado o teste dos motores de posição para verificar o comportamento dos mesmos. Nas juntas dos braços do robô foi utilizado o modo de controlador de posição, isto é: o motor recebe um comando de posição, que no caso é de 0 a 2PI radianos e assim é possível controlar para onde o braço irá se mover. O controlador recebe um arquivo de configuração das juntas, especificando os ids e assim pode-se controlar cada um individualmente. Foi realizado o teste em todas as juntas de posição do ELIR.

4.1.2.2 Teste dos Controladores de velocidade

Os controladores individuais de velocidade tem a função de controlar a velocidade do motor, sem controlar a posição. O motor irá rodar livremente com uma velocidade determinada até que receba um comando para parar. Este controlador foi usado nas rodas das garras que possuem a finalidade de fazer o robô andar sobre a linha. Cada roda fora testada individualmente.

4.1.2.3 Teste Controladorcontrolador de trajetoria

O controlador meta controller realiza o controle do posicionamento da junta como um todo. Neste controlador é possível controlar diversas juntas diferentes do robô por meio de uma única mensagem no ROS, contendo o nome das juntas que deverão ser controladas, as posições desejadas, tempos de execução, velocidades e esforço. Esse controlador se mostrou extremamente importante e eficiente pela sua capacidade de agrupar diversas juntas em um único comando de movimento.

4.1.2.4 Compatibilidade com diferentes protocolos de comunicação

Há uma limitação na biblioteca dynamixel driver, esta funciona somente para motores com firmware versão 1.0 devido a sua descontinuação. Foi testado para todas as versões contidas nos motores que possuíam e não houveram falhas em encontrar nenhum motor na biblioteca. Entretanto, caso houvesse necessidade de atualizar para versão v2.0, no qual possui uma taxa de baud rate maior e protocolo de comunicação melhorado, haveria a necessidade de parar com o uso a dynamixel driver e utilizar a biblioteca oficial da fabricante, a dynamixel workbench.

4.1.3 Controle dos servomotores utilizando a biblioteca dynamixel workbench

A biblioteca do ROS dynamixel workbench é outro driver para comunicação com os servomotores, recebe suporte e atualizações diretamente da ROBOTIS, sendo o driver de controle padrão do ROS para versões superiores a Kinetc. Considerou se utilizar essa ferramenta devido aos problemas de comunicação apresentados pelos motores. Ela oferece suporte para o protocolo de comunicação na versão 2.0, que suporta uma maior baudrate e muda a estrutura de comunicação.

4.1.3.1 Controladores de posição

A biblioteca oferece a opção de criar controladores de posição individuais para cada motor, sem especificar o ID. Não possibilitando uma criação controlador mestre-escravo para as juntas do robô que são atuadas por 2 motores e também a opção de selecionar quais motores da rede seriam transformados em controladores de posição, o arquivo que instancia os controladores só recebe um alcance de IDs que deve buscar na rede, transformando todos os motores nesse alcance em controladores de posição individuais.

4.1.3.2 Controladores de velocidade

Os controladores de velocidade só recebem o ID de dois motores, assim não apresentando um uso trivial, já que o robô necessita de 5 controladores de velocidade diferentes. Foi encontrado um arquivo customizado no repositório da biblioteca, mas mesmo assim não pode ser utilizado em conjunto com controladores de posição.

4.1.3.3 Controladores de trajetória

Os controladores de trajetória esperados para compatibilização com o planejamento de movimento são os padrões oferecidos pela biblioteca embutida do ROS, o `ros_control`. Porém a biblioteca não oferece um driver que instancie esses controladores de trajetória, assim não possibilitando a integração dela com o planejamento de movimento.

4.1.3.4 Comunicação em rede

Em comparação com a biblioteca antiga, a comunicação em rede dos servomotores utilizando essa biblioteca é levemente superior. Quando a comunicação apresentava problemas ,essa biblioteca se mostrou mais eficiente para encontrar os motores conectados na rede porém não possibilita o funcionamento completo, assim não se mostrando uma opção para resolução dos problemas de comunicação com os motores. Os motores com protocolo de comunicação 2.0 podem ser utilizados em conjunto com os 1.0 nessa biblioteca, porém não se tiverem o mesmo ID.

4.1.4 Compatibilidade do Elir Robot com o MoveIt!

O MoveIt! fornece uma ferramenta de configuração denominada Setup Assistant, que gera a maior parte dos arquivos necessários para a compatibilização.

IMAGEM

4.1.4.1 Teste de Estado de Colisão

Durante a configuração dos Setup Assistant , o robô apresentava um erro informando que o seu modelo estava em estado de colisão, o que significa que as partes do seu modelo estavam posicionadas erradas e estariam sempre colidindo. Foi feito um teste removendo todas as partes do robô do arquivo de configuração e checando se a indicação de erro sumia. Com as tentativas se concluiu o problema era o modelo 3D utilizado nas rodas do robô, que gerava a colisão quando se utilizava ele mais de uma vez, a resolução foi a criação de um novo modelo 3D para as rodas, após a substituição o erro sumiu.

4.1.4.2 Teste de Definição da corrente-cinemática e end-effector

No estudo de cinemática inversa na robótica, corrente cinemática é o nome dado para o conjunto de links e juntas que se vai calcular o movimento, onde o começo dessa corrente é a referência para o cálculo da cinemática e o final da corrente é o end-effector. No caso do Elir, é necessário se definir duas correntes cinemáticas, já que são dois braços. Com a corrente cinemática definida corretamente ,o MoveIt! consegue enviar o comando para movimentar as juntas da corrente simultâneamente. Ao definir o grupo de controle no Setup Assistant, o usuário escolhe as juntas que irão fazer parte desse grupo, e a

corrente cinemática é gerada automaticamente. Foram feitos teste considerando como parte da corrente somente as duas juntas do braço, porém ao analisar outros robôs que já haviam sido configurados, notou-se que sempre se levava em consideração a junta da base do robô para definição do grupo, onde a junta da base é a junta que prende o robô no mundo.

4.1.5 Robô de testes Davictory

Como o Setup-Assistant oferece diversas opções de configuração, optou-se por desenvolver um modelo de robô mais simples que o ELIR, de forma que se pudessem testar os conceitos, e compreender melhor a relação entre as configurações do modelo e os resultados esperados para movimento. Esse teste foi executado pelos membros Cleber Couto e Davi Oliveira, no período de: onde esse modelo foi inteiramente escrito pela equipe, tomando como base boas práticas observadas em robôs já compatibilizados código URDF. O Davictory foi construído como mostra a imagem[x]:

IMAGEM

Antes de realizar qualquer modificação na modelagem do robô ELIR primeiramente essa modificação foi feita Davictory, assim foram prevenidos quaisquer mudanças desnecessárias e otimizado o tempo devido que o robô customizado ser mais simples. Todos os testes com o planejamento de movimento foram validados também nele.

4.1.5.1 Teste com diferentes plugins de cinematica

Para configurar o pacote do MoveIT! de qualquer modelo de robô, é necessário definir qual plugin de solução da cinemática irá seguir. Como existem variados tipos de robôs com finalidades e conjuntos de movimentos diferentes essa escolha deve ser feita de forma precisa para que o robô em questão realize o movimento de forma eficiente. Durante a escolha para a cinemática do ELIR foi percebido que nenhum plugin se encaixava nas descrições devido ao número reduzido de graus de liberdades (o ELIR possui 2 graus de liberdades e os plugins necessitam de no mínimo 3). A fim de achar uma solução que atendesse ao funcionamento do ELIR, foram testados no Davictory todos os plugins de cinemática contidos no site oficial do MoveIT!. Nenhum plugin obteve sucesso em resolver a cinemática inversa.

4.1.6 Cálculo da cinemática inversa

Após a falha busca por um plugin que realizasse a cinemática inversa do ELIR, foi necessário calcular a cinemática inversa em um serviço por meio de equações. Partindo do pressuposto que o end-effector se encontra no 0 no plano cartesiano, é inserido as coordenadas para onde deve-se ir por meio das equações retorna-se os valores que cada junta deve realizar, a equação em questão é mostrada abaixo:

IMAGEM

4.1.7 Power Management

A placa de power management é responsável pelo gerenciamento de energia do robô, ela distribui de forma separada a potência para cada parte elétrica ou eletrônica, e pela sua capacidade de monitoramento, pode identificar e atuar sob algum erro na alimentação. O design e fabricação da placa foram realizados por Matheus Menezes e Branilson Luiz, tendo sua utilização adaptada para o ELIR. Diversos testes foram feitos para garantir que a mesma estivesse trabalhando de maneira correta.

4.1.7.1 Gravação e validação de Firmware na placa de Power Management

A gravação do firmware na placa de Power Management só foi realizada devido a utilização de um botão para conectar dois pinos específicos da placa (-VIN e S2) para que o microcontrolador fosse alimentado, e a conexão da placa em uma porta USB de um computador. Para que fosse possível estabelecer a comunicação entre o computador e a placa, foi necessário configurar as permissões necessárias no Ubuntu. Durante a gravação do firmware fora utilizada uma extensão do VSCode, o PlatformIO, específica para programação com microcontroladores, para realização de debug e gravação do firmware na placa. Durante o processo de gravação foi necessário manter o botão pressionado. Após a gravação do firmware na placa, o seu funcionamento fora validado através do uso dos serviços e mensagens gerados no ambiente ROS, uma vez que as informações de níveis de corrente e tensão das portas da placa eram publicados nos tópicos específicos. O teste foi coordenado por Ícaro Nascimento no dia 1 de agosto de 2018.

4.1.8 Acesso remoto a nuc

O robô deve ser acessado remotamente para que se inicie a missão, esse acesso é feito através do protocolo SSH. A comunicação no SSH funciona como servidor-cliente, onde a NUC é definida como o servidor, de forma a ser acessada remotamente. Por ser necessário o IP do servidor para realizar a conexão e durante os testes com o robô em linha não é possível consultar endereço de IP diretamente, foi criada uma rede Wi-Fi por um roteador móvel, onde o IP da NUC era fixo.

4.2 Testes integrados

asdfadsfsdfs

4.3 Avaliação da prontidão tecnológica

asdfadsfsdfs

4.4 Trabalhos futuros

asdfadsfsdfs

Capítulo Cinco

Conclusão

O resultado do projeto alcançou as expectativas. Os problemas que aconteceram conseguiram ser contornados e o tempo gasto para sua solução conseguiu se adequar ao esperado pelo cronograma das tarefas. O material produzido atende às demandas do cliente e os pacotes produzidos foram organizados buscando facilitar o uso por terceiros.

5.1 Considerações finais

A gestão do projeto do robô como um todo, fez com que o resultado produzido alcançasse as expectativas. O nível de desenvolvimento aumentou progressivamente de forma que o projeto foi conduzido, adicionando paulatinamente diversos conhecimentos específicos que não seriam vistos normalmentes durante a graduação, mas também fortalecendo conhecimentos já formados.

O que foi produzido para o projeto estará disponível para futuras consultas, o que impulsiona o desenvolvimento de projetos semelhantes. Novos estudos podem ser iniciados como trabalhos de graduação, ou pós-graduação, realizando provas de conceito e abrindo oportunidades para novas tecnologias.

O início do projeto se deu de forma lenta, por apresentar uma área do conhecimento nova para maior parte da equipe. A robótica necessita da integração de diversas áreas diferentes, e para a engenharia elétrica, o conhecimento de diversas camadas de abstração. Com a experiência e o desenvolver das atividades, os conhecimentos adquiridos possibilitaram atividades em paralelo e o aumento da versatilidade dos integrantes.

A experiência como um todo foi muito enriquecedora, adicionando conhecimentos que serão necessários no futuro e proporcionando um crescimento para todos os participantes.

Apêndice A

QFD

Apêndice B

Diagramas mecânicos

Apêndice C

Diagramas eletro-eletrônicos

Apêndice D

Wireframes

Apêndice E

Logbook

Referências Bibliográficas

BARABÁSI, A. L. *Linked: A Nova Ciência dos Networks*. São Paulo: Leopardo Editora, 2003.

NEWMAN, A.-L. B. M.; WATTS, D. J. *The Structure and Dynamics of Networks*. Princeton, NJ, USA: Princeton University Press, 2006.

WATTS, D. J. *Six Degrees: The Science of a Connected Age*. New York: W W Norton & Co., 2003.