INOVA+
Cadernos de Graduação da Faculdade da Indústria

ESTUDO EXPLORATÓRIO SOBRE APLICAÇÃO DE AMR: DESENVOLVER E SIMULAR SISTEMA DE NAVEGAÇÃO PARA ESTERILIZAÇÃO DE AMBIENTES

Bacharel em Engenharia de Software
Período: 6

Orientador
Professor Me. Mauricio Antonio Ferste

Autores
Bruna Pereira das Neves

Clístenes Grizafis Bento

Diego Murilo Sousa da Luz

Leonardo Pestilo dos Santos

Luiz Henrique Pereira Isbaes

Ozeias Mateus Santos Thomaz

RESUMO

O presente trabalho teve como foco desenvolver sistema de navegação para robôs móveis destinados a esterilização de ambientes. O objetivo geral deste trabalho é desenvolver e simular sistema de navegação para robô móvel que se movimente em um ambiente fechado por tempo pré-estipulado pelo usuário, tendo como objetivos específicos identificar na literatura quais os conceitos sobre robótica móvel, programação, aplicação Web, prototipagem e desenvolvimento de software, verificar quais requisitos e ferramentas necessárias para o desenvolvimento do sistema, modelar sistema proposto de acordo com os requisitos, confeccionar protótipo de interface de usuário de acordo com os requisitos, desenvolver sistema de acordo com o modelo e avaliar e validar resultados obtidos. As metodologias adotadas foram pesquisa de campo, bibliográfica, documental e de internet com temas relevantes ao assunto. Para a fundamentação teórica com o objetivo de facilitar o entendimento do leitor sobre o assunto, encontram-se esclarecimentos sobre engenharia de requisitos, UML, prototipagem e programação web. No início do desenvolvimento encontra-se a parte de levantamento de requisitos para o sistema. A segunda parte do desenvolvimento foi a modelagem do sistema de acordo com os requisitos levantados, seguido da etapa de prototipagem de telas e definição do ambiente para simulação do sistema de navegação proposto. Na quinta parte iniciou-se o desenvolvimento do sistema onde a equipe paralelamente desenvolveu o front-end, o middleware e o back-end de maneira paralela, seguido das etapas de avaliações e testes, discussão dos resultados obtidos e considerações finais. Na última parte encontram-se as referências e os anexos do trabalho.

Palavras-chave: 1 - Desenvolvimento de *Software*. 2 - Engenharia de *Software*. 3 - Robótica móvel.

1. INTRODUÇÃO

A robótica móvel é a área de pesquisa que aborda assuntos relacionados ao controle de veículos autônomos e semiautônomos, tendo como diferença entre as outras áreas de pesquisa da robótica a ênfase em robôs que se locomovem de maneira automatizada, autônoma, controlada por um piloto ou suas combinações (DUDEK; JENKIN, 2002).

Dentre as categorias de robôs móveis existe o Robô Móvel Autônomo (AMR, do inglês Autonomous Mobile Robots) que, de acordo com a empresa de robôs industriais Mobile Industrial Robots (2020) são robôs que se locomovem por meio de mapas que seu *software* constrói no ambiente por onde passa ou através de desenhos pré-carregados, o que reduz o custo e o tempo para sua aplicação nos mais diversos ambientes.

Atualmente o Brasil está passando por um período de pandemia onde de acordo com o Ministério da Saúde (2021) contabiliza mais de vinte milhões de casos confirmados de Covid-19 (vírus SARS-CoV-2). Mazur et al (2020) ressaltam a importância do isolamento social, higiene pessoal e uso de equipamentos de proteção individual como medidas para conter a disseminação da doença e propõem o uso de dispositivos de desinfecção de ambientes usando luz ultravioleta tipo C (UV-C) como premissa de preservação da vida.

Silva et al (2021) apontam que o uso de radiação ultravioleta tipo C vem sendo bastante utilizado por conta da pandemia de Covid-19, tendo hospitais como principais ambientes de utilização e alertam para os cuidados com a dose de radiação emitidas pela luz UV-C, o tempo de exposição e a distância entre a fonte e o objeto ou superfície, pois o uso pode ser nocivo para a saúde se não bem administrado.

Com a finalidade de proporcionar segurança na utilização de luz UV-C ou outros dispositivos, o presente trabalho, em parceria com a empresa Selettra, visa desenvolver um sistema de navegação para robôs móveis (AMR) equipados com os dispositivos de esterilização para ambientes, dispensando a presença humana nos ambientes.

1.2 IDENTIFICAÇÃO DA UNIDADE CONCEDENTE DO TRABALHO INTEGRADOR

Razão Social: Selettra Automação e Robótica LTDA;

Nome de Fantasia: Selettra;

Ramo: Fabricação de máquinas e equipamentos para uso industrial específico, peças e acessórios. Fabricação de aparelhos e equipamentos para distribuição e controle de energia elétrica. Instalação de máquinas e equipamentos industriais. Instalação e manutenção elétrica. Comércio atacadista de material elétrico. Comércio varejista de material elétrico. Serviços de engenharia;

CNPJ: 07.781.920/0001-33;

Localização: Rua Santa Efigenia, 31 – Roseira, na cidade de São José dos Pinhais - PR, CEP: 83070-200.

1.3 CONTEXTOS DA SITUAÇÃO NA EMPRESA

A Selettra Automação e Robótica LTDA é uma empresa brasileira presente no mercado desde 2002 e já desenvolveu trabalhos nas áreas de mecânica, elétrica e eletrônica. Atua no mercado de automação elétrica, CLPs (Controladores de Lógica Programavel) e IHMs (Interface Homem Máquina). Especialista em automação Industrial e uso de AGVs (veículo guiado automatizado).

1.4 OBJETIVOS

Os objetivos do trabalho estão separados em geral e específicos.

1.4.1 Objetivo Geral

Desenvolver e simular sistema de navegação para robô móvel que se movimente em um ambiente fechado por tempo pré-estipulado pelo usuário.

1.4.2 Objetivos Específicos

a) identificar na literatura quais os conceitos sobre robótica móvel, programação, aplicação *Web*, prototipagem e desenvolvimento de *software*;
b) verificar quais requisitos e ferramentas necessárias para o desenvolvimento do sistema;
c) modelar sistema proposto de acordo com os requisitos;
d) confeccionar protótipo de interface de usuário de acordo com os requisitos;
e) desenvolver sistema de acordo com o modelo;
f) avaliar e validar resultados obtidos.

1.5 METODOLOGIA

A metodologia utilizada no presente trabalho foi pesquisa exploratória, que segundo Marconi e Lakatos (2017) consiste em análise de dados levantados através de pesquisas

bibliográficas, estudo de caso e pesquisa de campo. O que para Gil (2017) tem o propósito de aumentar a familiaridade com o problema, objetivando a deixá-lo mais explícito ou facilitar a criação de hipóteses.

Com a finalidade de identificar os principais conceitos desenvolvidos neste trabalho, utilizou-se pesquisa bibliográfica, definido por Gil (2017) e Marconi e Lakatos (2017) como sendo uma pesquisa realizada a partir de materiais já publicados, com base em textos de artigos científicos, livros, ensaios críticos, entre outros.

Após as pesquisas foram realizadas etapas de coleta de dados através de entrevista informal com o responsável da empresa, definida por Gil (2017) como uma simples conversa objetivando coleta de dados, observação participativa e não participativa, que para Marconi e Lakatos (2017) é um processo ao qual o pesquisador entra em contato com o grupo ou a realidade estudada, onde no caso de participativa interagindo com ela e não participativa sem interagir com ela. Feito o levantamento e análise dos dados foi desenvolvido um protótipo, o que para Ambrose e Harris (2011) oferece a oportunidade de testar uma ideia com o objetivo de verificar se a mesma obtém êxito na prática. Sendo que esta é parte essencial para o processo de desenvolvimento de um produto, permitindo avaliação de sua funcionalidade para que seja possível fazer a produção definitiva (MACHADO, 2013).

Com o protótipo pronto foi desenvolvido o sistema utilizando os conceitos pesquisados seguidos de etapa de testes para avaliação e validação do sistema, que de acordo com Pressman e Maxim (2016) servem para verificar se o produto já pode ser construído ou precisa ser feito alguma correção.

Na última etapa do trabalho, foram realizadas as discussões sobre os resultados obtidos com as pesquisas, levantamento de requisitos, modelagem, prototipagem e desenvolvimento do sistema.

2. FUNDAMENTAÇÃO TEÓRICA

Com a finalidade de proporcionar melhor compreensão sobre os assuntos abordados, o presente capítulo está dividido em 5 partes, sendo elas engenhariam de requisitos, linguagem de modelagem unificada (UML, do inglês *Unified Modeling Language*), prototipagem e programação *web*.

2.1 ENGENHARIA DE REQUISITOS

Para Pressman e Maxim (2016) o amplo espectro de tarefas e técnicas que levam a uma compreensão dos requisitos é chamado engenharia de requisitos. Na concepção do processo de

software a engenharia de requisitos é um ato de engenharia de *software* que começa durante a atividade de comunicação e se estende a modelagem. Ela deve ser ajustada as utilidades do processo, do projeto, do produto e das pessoas.

Segundo Sommerville (2011) os requisitos da engenharia de requisitos são as descrições do que o sistema deve fazer, do seu funcionamento até as restrições além dos serviços oferecem. Esses requisitos retratam as necessidades dos usuários para um sistema.

Os requisitos estão divididos em dois tipos, requisitos funcionais e não funcionais. Sommerville (2011) relata que os requisitos funcionais de um sistema apresentam o que ele deve fazer. Quando apresentado como requisitos de usuário, os requisitos funcionais são descritos de forma abstrata, para serem entendidos pelos usuários do sistema. No entanto, requisitos de sistema funcionais mais característicos descrevem em detalhes as funções, suas entradas e saídas do sistema.

Já os requisitos não funcionais para Machado (2011) não estão conectados diretamente com as funções do sistema e sim se preocupam com padrões de qualidade como confiabilidade, desempenho, robustez, segurança, usabilidade, portabilidade, legibilidade, qualidade, manutenibilidade, entre outros. São considerados dados significativos, pois definem se o sistema será eficiente para a tarefa que for realizar.

Na tabela 1 pode-se comparar as principais diferenças entre os requisitos funcionais e não funcionais.

TABELA 1 – Requisitos funcionais e não funcionais

Requisitos funcionais	**Requisitos não funcionais**
Descrição: podem ser objetivo de usuário: especificados ao nível de uma única tarefa de responsabilidade de um usuário. Ou com objetivo agregador: especificados sob o nível de várias tarefas com vários usuários.	**Descrição:** são divididos em organização (locais de operação, *hardware*), Implementação (plataforma, linguagem de programação), Qualidade (facilidade de uso, eficiência, manutenção) e ambientais (segurança, privacidade, sigilo, interoperabilidade).
Exemplos: cadastrar produtos, consultar produtos, realizar compras, emitir nota fiscal.	**Exemplos:** Deve funcionar no sistema *windows*, deve ter um servidor dedicado, deve ser intuitivo, deve ser compatível com sistema *android*, tem que ser feito em linguagem *java.*

FONTE: Bento et al (2021, p. 211)

Pressman e Maxim (2016) ressaltam que além do processo de levantamento de requisitos é necessário avaliar e validar os requisitos antes de começar o desenvolvimento do sistema. Para isso é necessário desenvolver etapas de testes a serem feitas pela equipe e alinhadas junto ao cliente.

O levantamento de requisito permite ao desenvolvedor modelar como o sistema irá funcionar, desde a sua interação com o usuário até a estrutura em que será desenvolvido pelos programadores, dentre as ferramentas de modelagem de sistema destaca-se a *Unified Modeling Language* (UML).

2.2 UML

De acordo com Fowler (2014) a UML (*Unified Modeling Language*) é um grupo de representação gráfica para facilitar o entendimento e funcionamento de um *software*, grande parte desses gráficos são diagramas que utilizam o estilo orientado a objeto, os principais são: Diagrama de Caso de uso, Diagrama de classes, Diagrama de objetos, Diagrama de colaboração, Diagrama de sequência, Diagrama de atividades, Diagrama de estados, Diagrama de componentes, Diagrama de depuração, Diagrama de pacotes e outros, todos os diagramas tem suas funções e seus conceitos, a UML é controlada pela OMG (*Object Management Group*) criado por um grupo de empresas para estabelecer padrão para o entendimento global destas representações.

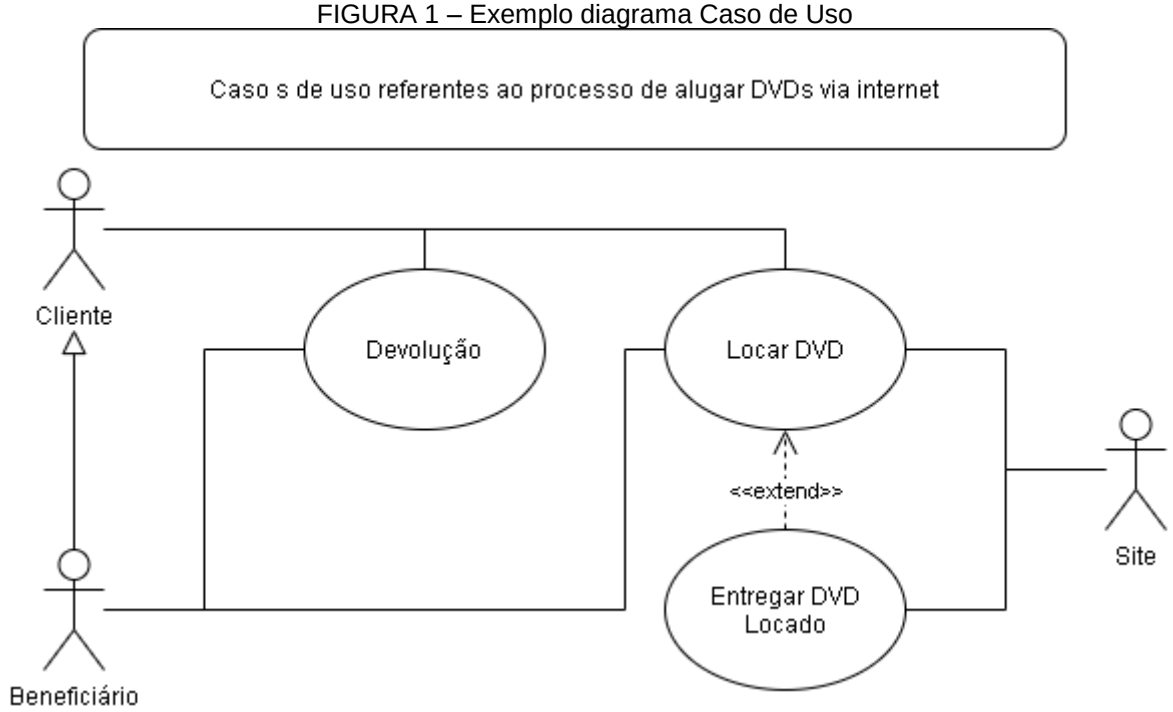
Para Martins (2010) a UML foi criada com o propósito de representar e modelar *softwares* e é utilizada em diversas áreas, atualmente a UML não está relacionada apenas com *softwares* mas com diversos outros tipos de sistema, como: workflow, estrutura e comportamento de dispositivos eletrônicos e mecânicos, *design* de *hardware*, além de diversos outros processos. A UML é utilizada por diversas empresas a maioria delas utilizam as etapas do RUP (*Rational Unified Process*), essas etapas apresentam quando e como são feitos procedimentos para realizar o trabalho completo e atingir o objetivo, ou seja, criar um *software* ou atualizar um já existente.

2.3.1 Diagrama de Caso de Uso

Medeiros (2010) classifica o caso de uso como a parte mais importante do desenvolvimento de *software*, desde o seu início até a sua conclusão. Sendo uma ferramenta de consulta, acerto, reuniões, discussão, alterações em requisitos e alterações em desenho, sendo construído por diversas ações que serão realizadas no sistema e dispõe de atores que são entidades externas que as executam.

O caso de uso possui descrição detalhada de cada ação no sistema seguido de um diagrama para representar todas essas ações, conforme ilustra a figura 1 que descreve as ações realizadas pelo cliente, beneficiário e site em relação ao processo de locação de uma locadora online.

FIGURA 1 – Exemplo diagrama Caso de Uso



FONTE: adaptado de Medeiros (2010, p. 47)

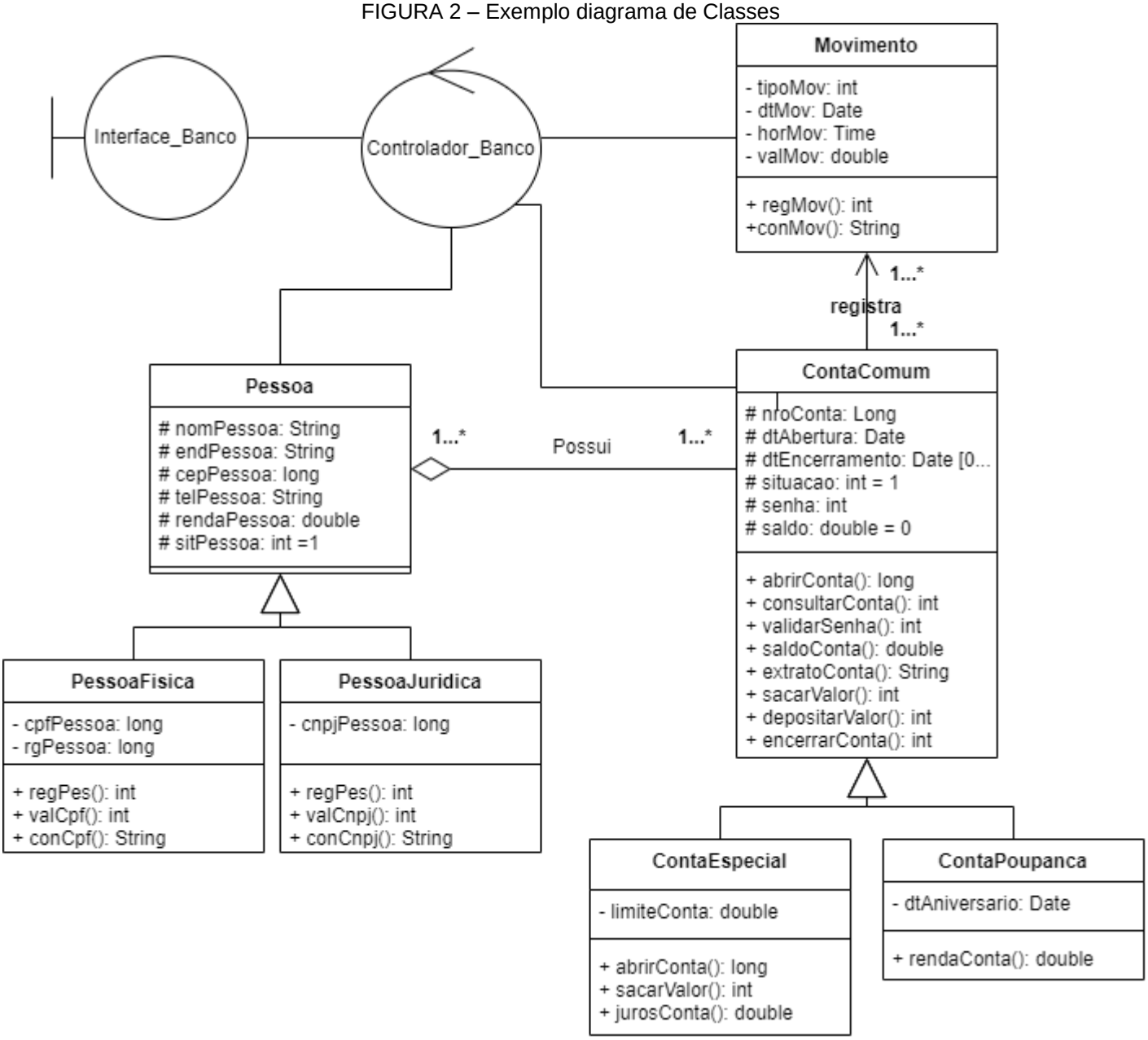
Guedes (2011) acredita que o diagrama de caso de uso, através de uma linguagem simples, permite a compreensão dos comportamentos externos ao sistema por qualquer pessoa, buscando apresentar a perspectiva do usuário.

2.2.2 Diagrama de Classes

Para Franco (2012), ao examinar um sistema é necessário que o analista de sistema desenvolva abstrações, regras e conceitos que relatam as funcionalidades do programa. O diagrama de classes permite esse levantamento, que através da abstração define-se os recursos, suas interações e informações para execução.

O diagrama de classes demonstra as diferentes classes que compõem o sistema e como elas se interagem. É composto fundamentalmente por classes, interfaces e relacionamentos. As classes contam com nome, atributo e métodos. Cada classe refere-se a objetos do mesmo modelo, onde cada definição de classe recebe um nome exclusivo. O nome da classe é necessário para definir o tipo de objeto que é demonstrado, tendo que ser um substantivo (FRANCO, 2012).

FIGURA 2 – Exemplo diagrama de Classes



FONTE: adaptado de Guedes (2011, p. 32)

A figura 2 ilustra o exemplo de um diagrama de classes aplicado em um sistema bancário.

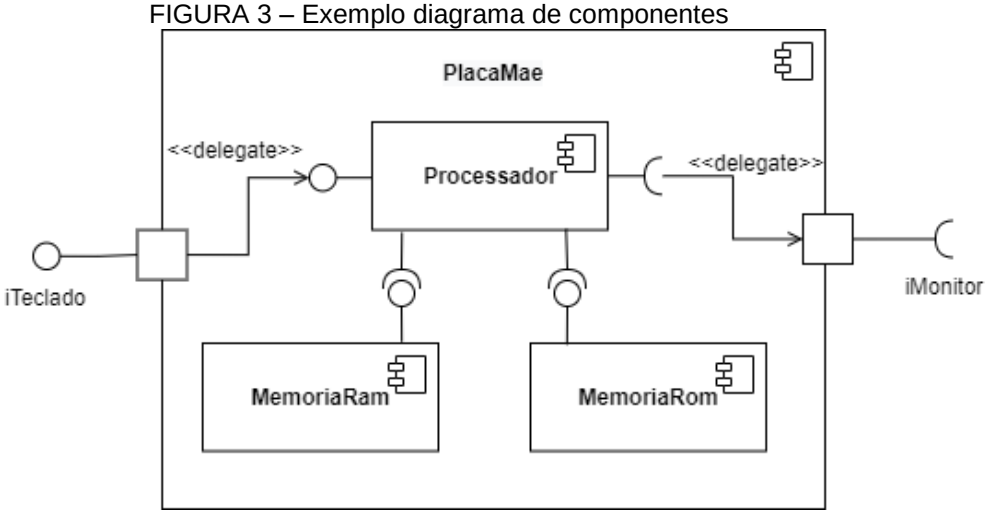
Guedes (2011) relata que o diagrama de classes é um dos mais importantes da UML, serve de suporte para a maioria dos demais diagramas. Como o próprio nome informa, define a estrutura das classes de um sistema, determinando os métodos e atributos que cada classe contém, além de determinar como as classes se associam e trocam informações.

2.2.3 Diagrama de Componentes

De acordo com Medeiros (2010) um componente é qualquer parte de um sistema necessária para a consecução de um *software*, ou seja, você pode representar qualquer parte do

seu sistema em um diagrama de componentes. No UML 2 os componentes do diagrama são representados por um retângulo com um símbolo formado por outros três retângulos em seu interior, conforme mostra a figura 3. Para representar a comunicação entre componentes utiliza-se o conceito de interface que é uma linha ligando um componente a outro sendo interrompida por um círculo e uma linha côncava, o círculo representa o componente que envia a informação e a linha concava o que recebe, a figura 3 ilustra o diagrama de componentes de uma placa mãe de computador.

FIGURA 3 – Exemplo diagrama de componentes



FONTE: adaptado Guedes (2011, p. 282)

Para Guedes (2011) o diagrama de componentes é utilizado para documentar a estrutura de elementos físicos de um sistema permitindo a melhor compreensão, além de facilitar a reutilização do código.

A modelagem irá auxiliar na criação de um protótipo do projeto, para que possa ser apresentado pelo cliente, facilitando o processo de prototipagem.

2.3 ARQUITETURA DE *SOFTWARE*

De acordo com Pressman e Maxim (2016) a arquitetura de um *software* é uma representação que nos permite analisar se um projeto será efetivo no atendimento dos requisitos, realizar mudanças em um estágio em que o projeto é relativamente fácil e reduzir riscos associados a construção de um *software*.

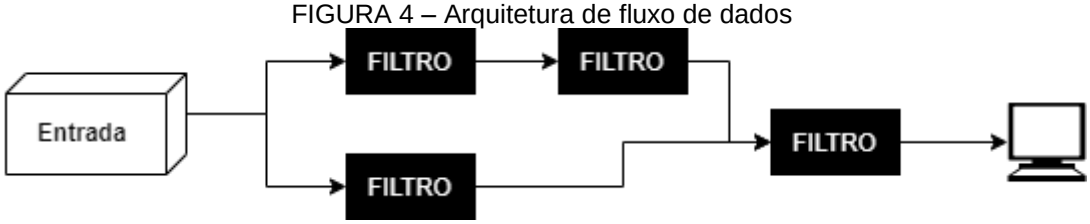
Bass, Clemensts e Kazman (2012) definem arquitetura de *software* como sendo uma estrutura (ou estruturas) do sistema que abrange os componentes de *software*, as propriedades extremamente visíveis e a relação entre componentes.

Dentre os tipos de arquiteturas de *software* existentes serão apresentadas a arquitetura de fluxo de dados, em camadas e orientada a objetos.

2.3.1 Arquitetura de Fluxo de Dados

É uma arquitetura aplicada quando dados de entradas são modificados por intermédio de uma série de componentes computacionais, conhecidos como filtros, ligados por flechas conhecidas como tubos, resultando na saída de um dado modificado de maneira específica (PRESSMAN; MAXIM, 2016). A figura 4 mostra um exemplo de arquitetura de fluxo de dados.

FIGURA 4 – Arquitetura de fluxo de dados



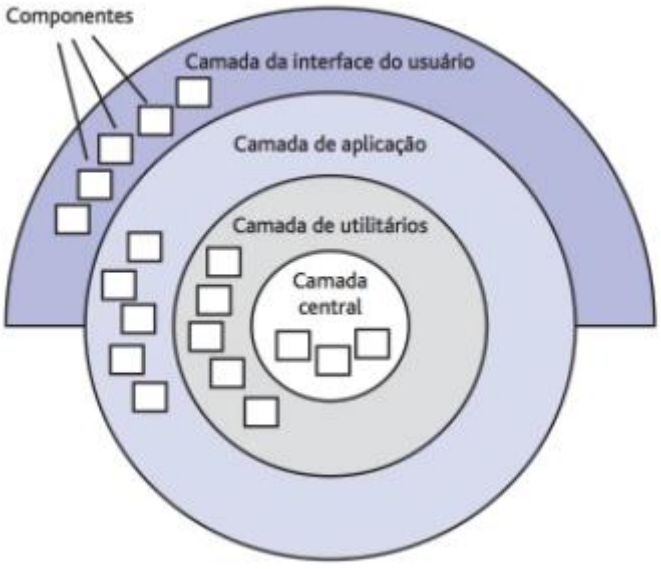
FONTE: Os autores (2021)

Zenker et al (2019) corrobora com o autor supracitado e acrescenta que na arquitetura de fluxo de dados, geralmente, um componente se comunica com outro de maneira interdependente e restrita.

2.3.2 Arquitetura em Camadas

Para Pessman e Maxim (2016) a arquitetura em camadas é estruturada em várias camadas diferentes iniciando da em uma linguagem próxima a linguagem humana e progressivamente se tornam mais próximas a linguagem de máquina, a figura 5 mostra a estrutura da arquitetura em camadas.

FIGURA 5 – Arquitetura em camadas



FONTE: Pressman e Maxim (2016)

De acordo com Zenker et al (2019) a arquitetura em camadas consiste em níveis que vão desde a interface (camada mais externa) até a camada central (mais próxima a linguagem de máquina.

2.3.3 Arquitetura Orientada a Objetos

De acordo com Zenker et al (2019) a arquitetura orientada a objetos é baseada em classe e objetos com certo grau de encapsulamento. Na arquitetura orientada a objetos os componentes dos sistemas manipulam dados encapsulados através de operações, onde a coordenação e comunicação entre componentes ocorrem através de mensagens (PRESSMAN; MAXIM, 2016).

2.4 PROTOTIPAGEM

De acordo com Morais et al. (2018) protótipos são versões de baixo custo de um produto, que podem ser testados e compartilhados pelas pessoas envolvidas com o projeto, é classificado como a fase experimental de um projeto e seu objetivo é encontrar a melhor solução para um produto.

Existem as prototipagens de baixo custo também conhecidas como prototipagem rápida, que ao contrário da prototipagem tradicional, está cada vez mais barata, com baixo tempo de produção e confiável. Este tipo de prototipagem está relacionado ao conjunto de tecnologias para criar simulações de objetos virtuais baseados nos físicos, fornecendo informações gráficas de maneira em que o cliente possa avaliar o produto antes da criação de um protótipo físico, possibilitando alterações não planejadas inicialmente (MACHADO, 2013).

Santos (2006) ressalta que o processo de prototipação é eficiente pelo seu custo-benefício devido a economia com retrabalhos. Os protótipos podem ser definidos como: baixa fidelidade, média fidelidade e alta fidelidade, onde cada nível de fidelidade diz respeito do quanto o protótipo está próximo do projeto final.

TABELA 2 – Níveis de protótipos

Protótipos de baixa fidelidade	Tem o propósito de testar funcionalidades mais simples e gerais do projeto, em outras palavras serve para testar de forma simplificada a interação do usuário com o sistema, sem a necessidade de se preocupar com interface. Os protótipos de baixa

	fidelidade, também chamados de rascunhos ou sketches.
Protótipos de média fidelidade	Envolvem mais as características de interface juntamente das funcionalidades básicas do sistema, podem não ter funcionalidades tão abrangentes, porém a interação do usuário com o sistema é melhor desenvolvida através de funções básicas de *design*, botões clicáveis e entre outras funcionalidades mais próximas do modelo final.
Protótipos de alta fidelidade	Possuem tanto características de interface quanto funcionalidades de interação ao usuário bem próximas ao desenvolvimento final, muitas vezes podem até serem utilizados como testes funcionais do sistema, o objetivo dos protótipos de alta fidelidade é verificar a usuablidade do projeto para que possa ser validado aos usuários.

FONTE: adaptado Santos (2006, p. 261)

Depois de pronto o protótipo é apresentado para a parte interessada a qual avalia o resultado, sugere correções e valida o desenvolvimento.

2.5 PROGRAMAÇÃO *WEB*

De acordo com Mariano e Melo-Minardi (2017) a aplicação *web* é separada em dois polos de ação: o *back-end* que são os códigos que fazem o sistema funcionar corretamente com todas suas funções, executado na máquina servidor e o *front-end* em que se desenvolve a parte visual da aplicação, funcionará diretamente na máquina do usuário. O desenvolvimento *web* é utilizado por todas as organizações que possuem um site ou um sistema na internet, para a criação de uma aplicação *web* é possível utilizar diversas linguagens, porém as mais utilizadas são: HTML, CSS, PHP, JavaScript e SQL.

Para Purewal (2010) a criação de uma aplicação *web* é uma tarefa complicada e deve ser organizada. A organização é feita através de pastas padrão que tem como objetivo organizar o projeto e torná-lo funcional.

Na aplicação *web* toda parte visual é feita por uma linguagem de marcação, já a parte lógica é realizada por uma linguagem de programação e suas informações ficam salvas em um banco de dados.

2.5.1 Linguagem de Marcação

Para Eis e Ferreira (2012) a linguagem de marcação é usada para publicações de conteúdo de textos, imagens, vídeos, áudios, entre outros conteúdos visuais normalmente voltado a aplicações *web*.

A linguagem de marcação é formada por *tags* (do inglês etiquetas) de marcação onde cada *tag* descreve um conteúdo dentro da aplicação *web* podendo estar em diferentes documentos. *Tags* são as palavras-chave ocultas dentro da página web que descrevem como é formatado e exibido o conteúdo dentro da aplicação *web*. O *Hyper Text Markup Language* (HTML) é a linguagem de marcação padrão para criação de páginas *web* e de escrever toda a sua estrutura (w3schools.com, 2020).

2.5.2 Linguagem de Programação

Para Ascenio e Campos (2012) programação é o ato de desenvolver algum programa para um determinado processamento de dados, para que o computador compreenda o que precisa ser executado é necessário que essa programação seja escrita em uma linguagem própria, chamada de linguagem de programação e o texto escrito nessa linguagem é chamado de código-fonte.

Manzano (2014) corrobora com os autores supracitados e acredita que para escrever programas é necessário utilizar no mínimo duas ferramentas: Um editor de textos para o código-fonte e outro para transformá-lo em código de máquina, ou seja, um código que o computador compreenda. Essas e outras ferramentas podem ser encontradas em um mesmo pacote que recebe o nome de Ambiente Integrado de Desenvolvimento (IDE, do inglês *Integrated Development Environment*). Existem vários tipos de IDEs disponíveis, para os diversos tipos de linguagens existentes.

2.5.3 Banco de Dados

Segundo Korth, Silberchatz e Sudarshan (2012), o banco de dados é uma coleção de dados que representam informações sobre um domínio específico. Barboza e Freitas (2018) acrescentam que um banco de dados bem construído é formado por um aglomerado de informações, que juntas geram algum sentido ao contexto em que estão inseridas e possuem um sistema de gerenciamento de banco de dados (SGBD), que são um conjunto de programas que permitem definir, manipular e construir bases de dados para diversas finalidades.

De acordo com Machado (2020) para consulta e manipulação dentro do ambiente de banco de dados existe a linguagem de banco de dados, na qual a adotada atualmente como padrão é a *Structured Query Language* (SQL, linguagem estruturada de pesquisa) devido sua forma de realizar manipulação e consulta de dados, baseada em um modelo relacional.

Barboza e Freitas (2018) corroboram com o autor supracitado informando que a linguagem SQL age de maneira intermediária, como uma ponte entre o que precisa ser feito e o banco de dados efetivamente, permitindo a realização de inserções, remoção, edição ou qualquer outra atividade com os dados que já estão no banco de dados ou irão para ele.

3. LEVANTAMENTO DE REQUISITOS

O presente capítulo trata do levantamento de requisitos, onde foram coletados e analisados os requisitos de negócio, funcionais e não funcionais para o sistema a ser criado.

Foram coletados inicialmente por entrevista informal através de uma reunião entre as partes interessadas com a finalidade de entender a necessidade do cliente/usuário, feito isso foi realizado uma análise de cenário simulando como funcionaria o sistema.

Após ter realizado a coleta de dados foi utilizado uma técnica chamada 5W2H que conforme Lisbôa e Godoy (2012) é uma técnica que permite identificar as rotinas e dados mais importantes para um projeto ou unidade de produção, sendo formado por sete perguntas conforme ilustra a tabela 3.

TABELA 1 – Descrição 5W2H

5W	*What* (O que?)	Qual a atividade? Qual é o assunto? O que deve ser medido? Quais os resultados dessa atividade? Quais atividades são dependentes delas? Quais atividades são necessárias para o início da tarefa? Quais insumos necessários?
	Who (Quem?)	Quem conduz a operação? Qual a equipe responsável? Quem executará determinada

		atividade? Quem depende da execução da atividade? A atividade depende quem para ser iniciada?
	Why (Por quê?)	Por que a operação é necessária? Ela pode ser omitida? Por que A, B e C foram escolhidos para executar a atividade?
	Where (Onde?)	Onde a operação será conduzida? Em que lugar? Onde a atividade será executada? Onde serão feitas as reuniões presenciais da equipe?
	When (Quando?)	Quando será feito? Quando será o início da atividade? Quando será o término? Quando serão as reuniões presenciais?
2H	*How* (como?)	Como conduzir a operação? De que maneira? Como a atividade será executada? Como acompanhar o desenvolvimento dessa atividade? Como A, B e C vão interagir para executar esta atividade?
	How much (quanto?)	Quanto custa realizar a mudança? Quanto custa a operação atual? Qual é a relação custo/benefício? Quanto tempo está previsto para atividade?

Fonte: Adaptado de Lisbôa e Godoy (2012, p. 37)

Com a técnica 5W2H obteve-se os seguintes resultados:

a) ***What*:** desenvolver um sistema de navegação para robô móvel já existente para o robô circular em ambientes fechados;

b) ***Who*:** sistema desenvolvido para o próprio robô, destinado a pessoa que pretende operá-lo;

c) ***Why*:** garantir a segurança de pessoas no processo de esterilização de ambientes;

d) ***Where***: será usado pelo robô em ambientes vazios a serem esterilizados;

e) ***When*:** será usado no horário que for mais conveniente ao usuário, sendo programado para executar a navegação por prazo pré-estipulado em minutos;

f) ***How*:** através de um robô móvel e um dispositivo com acesso a rede do robô e aplicação *Web*;

g) ***How Much*:** estima-se um valor de R$170.000,00, pela venda do robô já com sistema de navegação e R$ 40.000 apenas pelo sistema de navegação.

Com o auxílio da ferramenta 5W2H foram desenvolvidos os requisitos de negócio, funcionais e não funcionais do sistema.

Requisitos de negócio:

a) esterilizar ambientes vazios garantindo a segurança das pessoas através do uso de robótica móvel;
b) oferecer aos clientes robô de fácil utilização sem a necessidade treinamento, com o uso de apenas um manual de instruções;
c) dispensar o uso de infraestrutura para a aplicação do sistema, bastando apenas o robô para funcionar.

Requisitos funcionais:

a) o sistema deve ter tela principal: o sistema deve permitir que o usuário se comunique através de uma tela principal onde pode definir parâmetros e iniciar a esterilização;
b) o sistema deve ter tela de controle manual: o sistema deve permitir que o usuário possa mover o robô manualmente através de um controle remoto em uma tela de controle manual;
c) o sistema deve ter as opções iniciar, pausar e cancelar: o sistema deve permitir que o usuário através da tela principal possa iniciar o processo de esterilização do ambiente, pausar o processo se necessário retornando do ponto em que parou ou cancelar o processo caso deseje;
d) o sistema deve fazer esterilização de ambiente por prazo programado: o sistema deve através da tela principal permitir que o usuário ajuste o tempo em minutos que deseja que o robô se movimente pelo ambiente fazendo esterilização;
e) o robô deve voltar ao ponto de início após terminar o prazo programado: Após encerrar o tempo de execução o robô deve voltar para o ponto em que começou a esterilização;
f) o robô deve informar seu estado: O robô deve informar através de suas telas se está disponível, ocupado, em modo manual, em modo automático e quanto tempo falta para terminar sua tarefa;
g) o robô deve informar a porcentagem de bateria: O robô deve informar através de suas telas qual a porcentagem de bateria e exibir uma.

Requisitos não funcionais:

a) o sistema deve funcionar em um robô já existente: O sistema dever ser compatível com um robô existente e que possuí outros sistemas funcionando;
b) o sistema de navegação deve ser feito através de uma ferramenta chamada ROS: O robô já existente funciona utilizando uma ferramenta chamada ROS, o novo sistema deve ser construído utilizando essa ferramenta;

c) **a Interface de interação com o usuário dever ser feita através de aplicação *Web*:** A interação do sistema deve ser feita utilizando aplicação *Web*;
d) **a interface de interação do sistema deve funcionar em *desktop*:** O usuário do sistema deve poder acessar ao sistema de um computador;
e) **o sistema deve funcionar em rede local:** O sistema deve funcionar sem a necessidade de internet, através de um ip fixo em uma rede local;
f) **o sistema deve funcionar no sistema operacional Ubuntu:** O computador disponível no robô tem como sistema operacional Ubuntu 20.04;
g) **a comunicação entre o *front-end* e o *back-end* deve ser feita através de uma ferramenta chamada Node-red:** o sistema já utilizado no robô utiliza o node-red para fazer a comunicação entre a ferramenta ROS, um CLP da Siemens e a aplicação *web*
h) **as telas do sistema devem ser intuitivas:** Deve
i) **o sistema deve permitir mudanças futuras:** O sistema deverá ser modificável e expansível para futuras atualizações.

Com os requisitos levantados foi possível realizar a modelagem do sistema para apresentar ao cliente como o sistema funcionará e aos desenvolvedores mostrando como o sistema será desenvolvido.

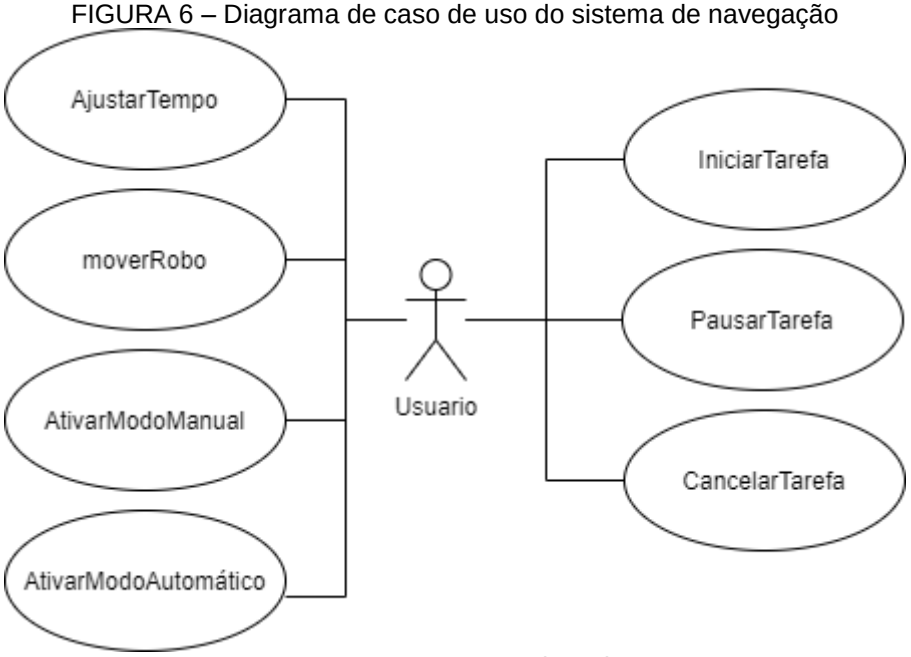
4. MODELAGEM

A modelagem dos requisitos levantados foi através de diagramas da UML, o primeiro deles foi o diagrama de caso de uso, cujo objetivo é apresentar para as partes interessadas as funcionalidades do sistema. Diagrama acompanha uma tabela com a descrição detalhada de cada caso de uso junto com os seus ciclos.

4.1 CASO DE USO

Os casos de uso criados para o sistema estão divididos em diagrama que permite a identificar visualmente cada funcionalidade do sistema e linguagem estruturada que permite a compreensão de cada fluxo e critérios dos casos.

FIGURA 6 – Diagrama de caso de uso do sistema de navegação



FONTE: Os autores (2021)

Para cada caso de uso foi construído uma tabela contendo o nome do caso de uso os atores que representam os usuários do sistema, descrição detalhada da funcionalidade, as condições que a precedem, o que ocorre após usar essa funcionalidade, o fluxo básico que descreve a interação entre o sistema e os atores, fluxos alternativos que descrevem interações que podem ocorrer além do fluxo básico e os fluxos de exceções que ocorrem quando há algum erro na interação.

A tabela 4 descreve o caso de uso de ajustar o tempo da tarefa.

TABELA 4 – Caso de uso: ajustar tempo

Caso de uso – ajustarTempo	
Nome	CU01
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento do ajuste de tempo (em minutos) em que o robô AMR irá executar sua tarefa.
Pré-condições	O sistema dever ter sido iniciado, os atores devem estar acessando a tela principal do sistema e o robô deve estar no estado de disponível.
Pós-condições	É configurado o tempo em minutos que o robô AMR irá executar sua tarefa a partir do momento em que ela for iniciada.

Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe campo de formulário para preenchimento com valor padrão preenchido.
2. Digita o novo tempo em minutos.	
	3. exibe o novo tempo no campo de formulário.

FONTE: Os autores (2021)

A tabela 5 descreve o caso de uso Iniciar Tarefa.

TABELA 5 – Caso de uso: iniciar tarefa

Caso de uso – iniciarTarefa	
Nome	CU02
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento da ação de iniciar tarefa que o robô AMR irá executar.
Pré-condições	O sistema dever ter sido iniciado, os atores devem estar acessando a tela principal do sistema e o robô deve estar no estado de disponível, em modo automático e o tempo de tarefa pré-estipulado.
Pós-condições	O robô AMR inicia sua tarefa de esterilização do ambiente.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe botão com o símbolo “*Play”*
2. Clica no botão.	
	3. Verifica o estado, o modo do robô e o tempo para execução da tarefa. Se o estado for disponível ou “em pausa”, o modo for automático e o tempo for maior que zero: inicia tarefa do robô pelo prazo pré-estipulado, caso contrário executa o fluxo de exceção.

Fluxo de exceção	
Ações dos atores	**Ações do sistema**
	1. aguarda os parâmetros estarem de acordo para a execução do fluxo básico.

FONTE: Os autores (2021)

A tabela 6 descreve o caso de uso Pausar Tarefa.

TABELA 6 – Caso de uso: pausar tarefa

Caso de uso – pausarTarefa	
Nome	CU03
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento da ação de pausar tarefa que o robô AMR está executando.
Pré-condições	O CU02 deve ter sido executado e o robô deve estar no modo automático.
Pós-condições	O robô AMR para a execução da sua tarefa e aguarda comando para retomar de onde parou.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe botão com o símbolo “*pause”*
2. Clica no botão.	
	3. Verifica o estado, o modo do robô e o tempo para execução da tarefa. Se o estado for ocupado, o modo for automático e o tempo for maior que zero: para o robô e o contador de tempo e altera o estado para “em pausa”, caso contrário executa o fluxo de exceção.
Fluxo de exceção	
Ações dos atores	**Ações do sistema**
	1. aguarda os parâmetros estarem de acordo para a execução do fluxo básico.

FONTE: Os autores (2021)

A tabela 7 descreve o caso de uso Cancelar Tarefa.

TABELA 7 – Caso de uso: cancelar tarefa

Caso de uso – cancelarTarefa	
Nome	CU04
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento da ação de cancelar tarefa que o robô AMR está executando.
Pré-condições	O CU02 de ter sido executado e o robô deve estar no modo automático.
Pós-condições	O robô AMR para a execução da sua tarefa e muda o estado para disponível.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe botão com o símbolo “*cancel”*
2. Clica no botão.	
	3. Verifica o estado, o modo do robô e o tempo para execução da tarefa. Se o estado for ocupado, o modo for automático e o tempo for maior que zero: para o robô, reseta o contador de tempo e altera o estado para disponível, caso contrário executa o fluxo de exceção.
Fluxo de exceção	
Ações dos atores	**Ações do sistema**
	1. aguarda os parâmetros estarem de acordo para a execução do fluxo básico.

FONTE: Os autores (2021)

A tabela 8 descreve o caso de uso Ativar Modo Manual.

TABELA 8 – Caso de uso: ativar modo manual

Caso de uso – ativaModoManual	
Nome	CU05
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade

	descrever o funcionamento de mudança de modo do robô AMR para manual.
Pré-condições	O sistema dever ter sido iniciado, os atores devem estar acessando alguma tela do sistema e o robô deve estar em modo automático.
Pós-condições	O robô AMR entra em modo manual e pode ser controlado remotamente.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe *checkbox* marcado, com legenda “modo automático”.
2. desmarca o *checkbox*.	
	3. verifica o estado do robô, se for disponível muda o robô para modo manual, caso contrário executa o fluxo de exceção.
Fluxo de exceção	
Ações dos atores	**Ações do sistema**
	1. cancela a tarefa que o robô está executando, reseta o contador de tempo, muda o estado para disponível e deixa o robô em modo manual.

FONTE: Os autores (2021)

A tabela 9 descreve o caso de uso Ativar Modo Automático.

TABELA 9 – Caso de uso: Ativar modo automático

Caso de uso – ativarModoAutomatico	
Nome	CU06
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento de mudança de modo do robô AMR para automático.
Pré-condições	O sistema dever ter sido iniciado, os atores devem estar acessando alguma tela do sistema e o robô deve estar em modo manual.

Pós-condições	O robô AMR entra em modo automático e pode iniciar tarefas.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. Exibe *checkbox* desmarcado, com legenda “modo automático”.
2. marca o *checkbox*.	
	3. muda o modo do robô para automático.

FONTE: Os autores (2021)

A tabela 10 descreve o caso de uso Mover Robô

TABELA 10 – Caso de uso: mover robô

Caso de uso – moverRobo	
Nome	CU07
Atores	Usuário do sistema
Descrição	Esse caso de uso tem como finalidade descrever o funcionamento do controle remoto do robô.
Pré-condições	O sistema dever ter sido iniciado, os atores devem estar acessando a tela de controle manual do sistema e o robô deve estar no modo manual.
Pós-condições	O robô AMR movimenta-se de acordo com a direção que é selecionado.
Fluxo básico	
Ações dos atores	**Ações do sistema**
	1. exibe controlador direcional para movimentação do robô AMR.
2. seleciona uma direção.	
	3. verifica modo do robô, se ele estiver no modo manual movimenta o robô para direção selecionada, caso contrário executa o fluxo de exceção.
Fluxo de exceção	
Ações dos atores	**Ações do sistema**

	1. aguarda os parâmetros estarem de acordo para a execução do fluxo básico.

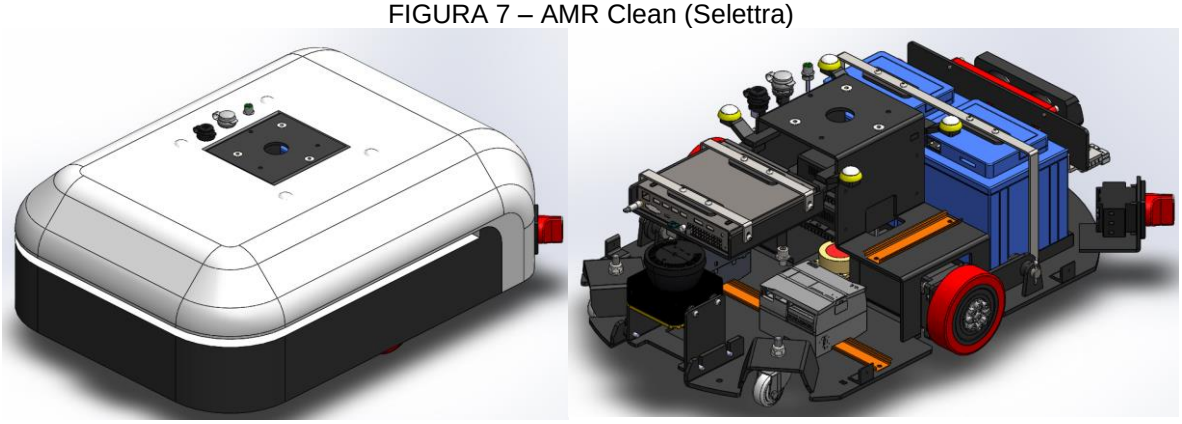
FONTE: Os autores (2021)

Após a modelagem dos casos de uso iniciou-se a modelagem do sistema para o desenvolvimento, onde o primeiro passo foi entender a estrutura física do robô e como cada componente se comunica.

4.2 DESCRIÇÃO DO ROBÔ MÓVEL

O AMR oferecido pela empresa Selettra tem seu modelo defino pelo nome CLEAN, é um robô móvel capaz de rotacionar em seu próprio eixo, equipado com dois motores um para cada roda, *encoders* para leitura de deslocamento para as rodas, sensor escâner do tipo lidar com ângulo de varredura de 275º e alcance de 40 metros, sensor de rotação, acelerômetro, um controlador lógico programável (CLP), um mini computador, além de outros componentes disponíveis em seu projeto, a figura 7 ilustra o modelo Clean da Selettra.

FIGURA 7 – AMR Clean (Selettra)



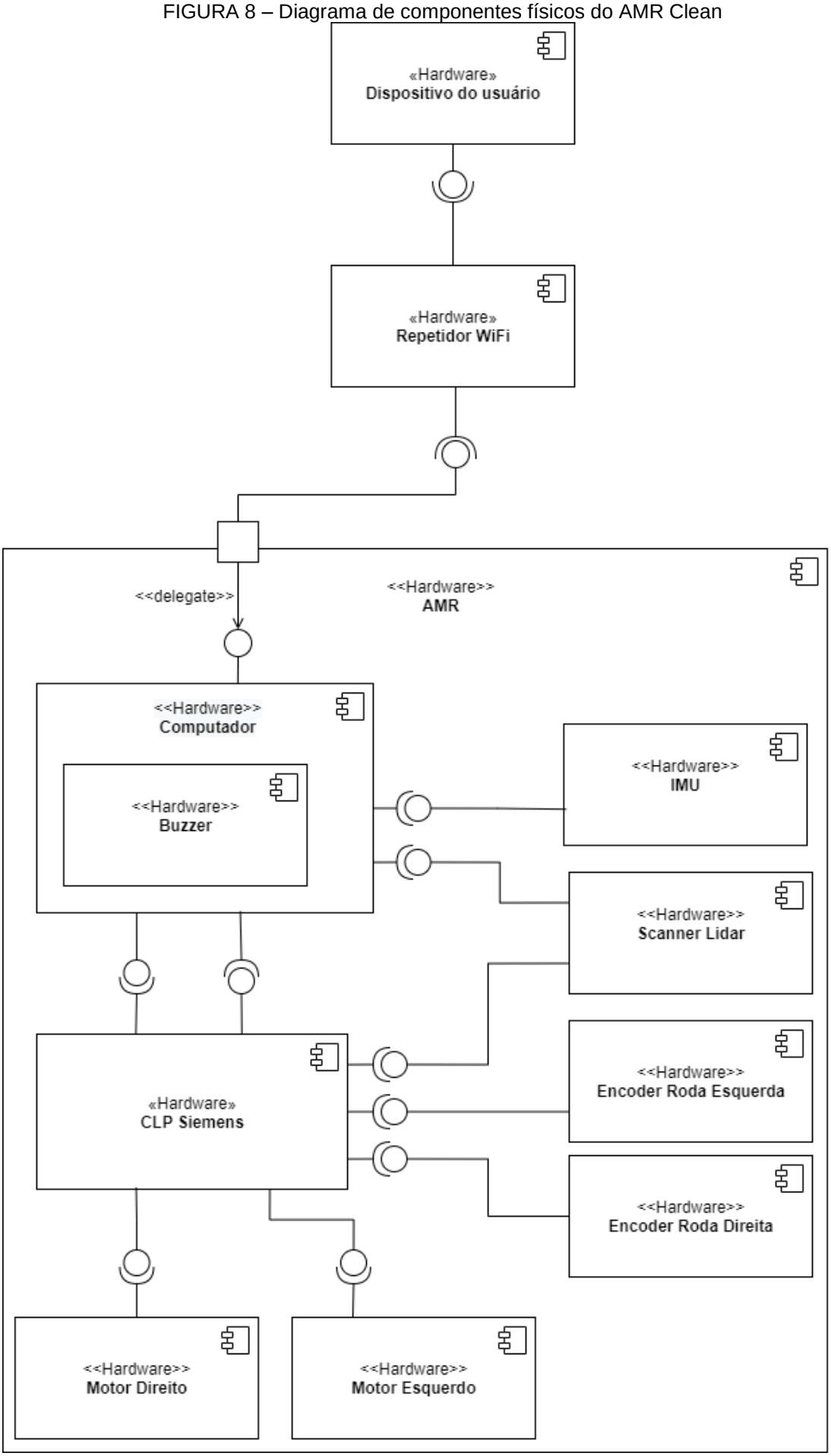
FONTE: Selettra (2021)

A equipe de desenvolvimento do presente projeto foi pessoalmente na empresa para conhecer o AMR clean e assim desenvolver o diagrama de componentes do modelo físico do robô.

4.2.1 Diagrama de componentes do modelo físico do robô

O diagrama de componentes foi elaborado utilizando os principais componentes físico que serão utilizados pelo sistema com a finalidade de proporcionar melhor compreensão de como o AMR funciona e auxiliar na modelagem da arquitetura de *software* que será utilizada. A figura 8 ilustra o diagrama de componentes criado.

FIGURA 8 – Diagrama de componentes físicos do AMR Clean



FONTE: Os autores (2021)

Com o diagrama é possível compreender o funcionamento da comunicação entre os componentes do robô, onde através de um dispositivo, podendo ser um smartphone, tablet ou computador, o usuário envia e recebe informações do AMR por intermédio de uma rede local WiFi. O computador do AMR é o responsável pelo processamento dos dados necessários para realizar a movimentação do robô, recebendo dados de rotação e acelerômetro de um sensor chamado IMU, dados de localização de objetos através de um escâner laser lidar e dados de deslocamento de cada roda obtidos pelo CLP e enviando dados de movimentação para o CLP que por sua vez aciona os motores de acordo com os dados recebidos. O computador possui também um buzzer que é uma buzina para emissão de sinais sonoros de acordo com a necessidade do cliente.

O escâner lidar também envia informações para o CLP para ajustes de alerta e segurança de acordo com a distância de algum objeto em relação ao robô, onde dependendo das configurações o robô é capaz de reduzir a velocidade em estado de alerta e parar em estado de segurança, essas configurações funcionam de acordo com o alcance do ângulo de varredura e distância do escâner.

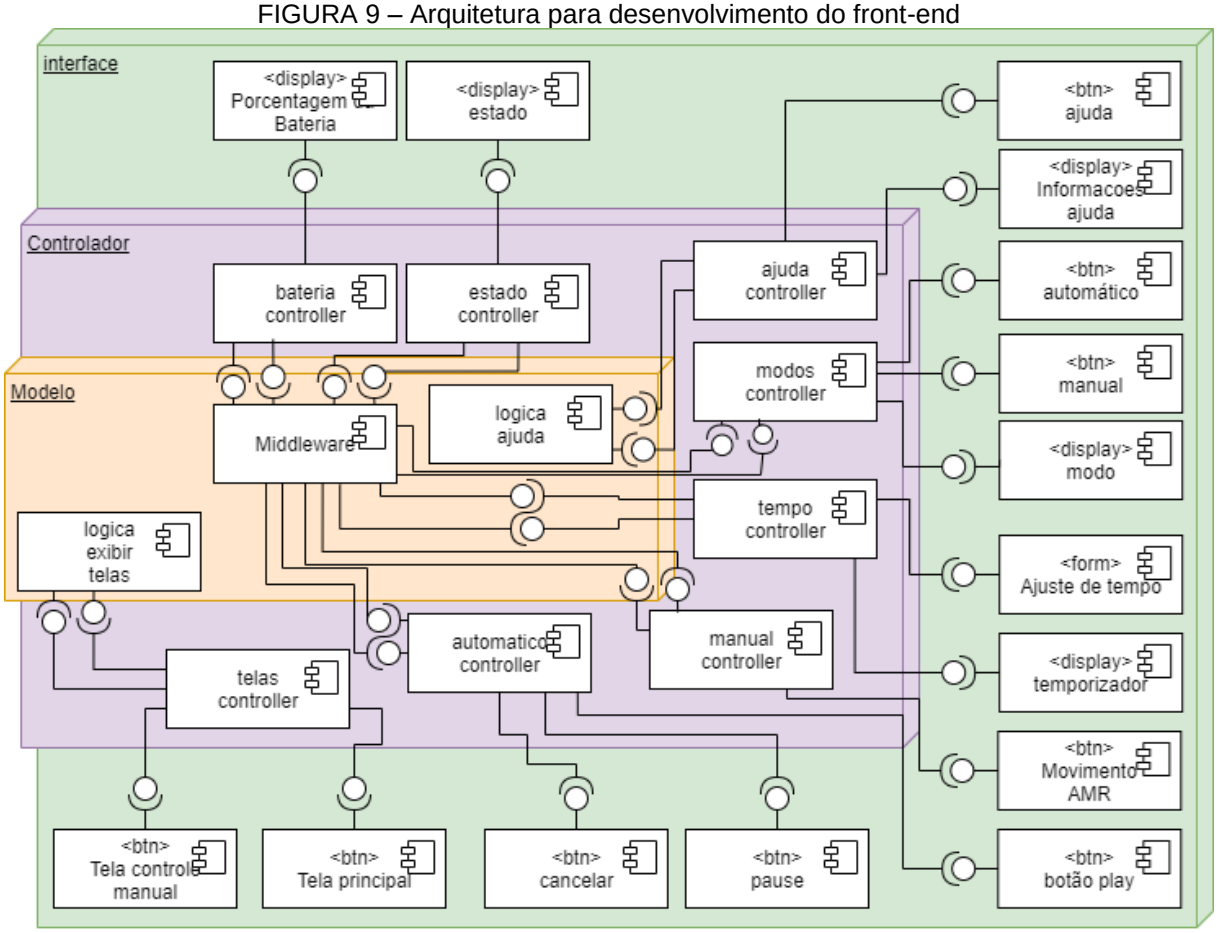
Atualmente com a ajuda de seus sensores o AMR Clean já possui um sistema de controle manual, sendo controlado de maneira remota por um dispositivo usando rede WiFi, possui também um sistema de localização, sendo capaz de gravar mapa do ambiente em que está inserido e identificar sua localização dentro do mapa, o robô também possui um sistema de movimentação, onde ao receber informações de velocidade linear e angular movimenta-se de acordo com os dados recebidos.

Após estudar a características do AMR e os sistemas já embarcados nele, percebeu-se que para realizar o desenvolvimento de um sistema de navegação em um ambiente vazio e fechado é necessário dividi-lo em três partes *front-end* responsável pela interação entre o usuário e o sistema seguindo o modelo criado nos casos de uso, *back-end* responsável pelo processamento das informações e realizações das atividade, fazendo a integração entre o sistema de localização e movimentação já presentes no AMR e o middleware que será responsável pela comunicação entre o *front-end, back-end* e outros eventuais componentes necessário para o sistema.

4.3 MODELAGEM *FRONT-END*

Para a modelagem do *front-end* do sistema escolheu-se a arquitetura de camadas e os diagramas da UML de componentes e implantação, onde o resultado é ilustrado na figura 9.

FIGURA 9 – Arquitetura para desenvolvimento do front-end



FONTE: Os autores (2021)

A arquitetura do front-end está dividida em três níveis:

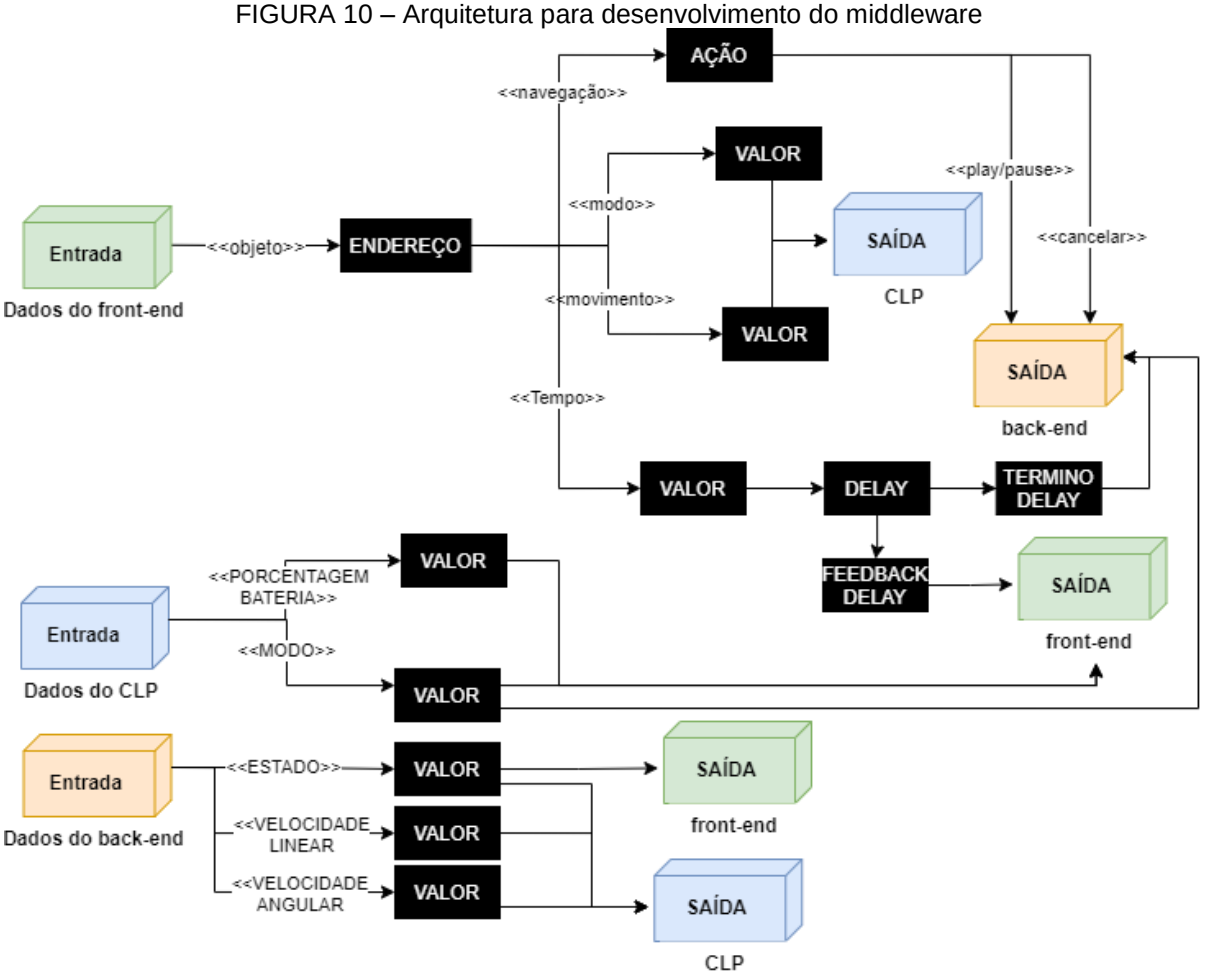
1. **Interface:** Descreve os componentes que devem aparecer na interface gráfica do sistema, o qual o usuário irá interagir, a legenda <btn> representa que é um botão, <form> entrada de formulário e <display> informação exibida na tela;
2. **Controlador:** Descreve os componentes que fazem a comunicação entre o modelo e a interface, realizando alguma chamada de função ou enviando e recebendo mensagens;
3. **Modelo:** Descreve os componentes responsáveis a parte lógica do front-end.

A arquitetura do front-end possibilita a criação de protótipo de telas e o desenvolvimento do sistema responsável pela interação entre o usuário e o AMR Clean.

4.4 MODELAGEM *MIDDLEWARE*

Para o desenvolvimento do *middleware* do sistema foi escolhido a arquitetura de de fluxo de dados, onde o recebe-se dados provenientes do *front-end*, esses dados então recebem modificações e são enviados para o *back-end*, onde do mesmo ocorre com dados recebidos do *back-end* e enviados para o front-end, além da entrada e saída de dados do CLP. A figura 10 ilustra a arquitetura desenvolvida para o *middleware*.

FIGURA 10 – Arquitetura para desenvolvimento do middleware



FONTE: Os autores (2021)

Conforme mostra a figura 10, no *middleware* os dados recebidos do *front-end* chegarão no formato de objeto, onde será verificado o destino de cada atributo que compõe esse objeto da seguinte forma:

a) navegação: é enviado para o *back-end* uma ação podendo ser elas “iniciar tarefa, parar tarefa ou cancelar tarefa;

b) modo: é enviado para o CLP ajuste de modo manual ou automático;

c) movimento: é enviado para o CLP números inteiros de 0 até 16, onde para cada número o AMR realizará um movimento diferente no modo manual (observação: para a simulação esses números serão enviados para o *back-end*);

d) tempo: é enviado um valor inteiro positivo que representa minutos de execução, esse valor passa por um filtro chamado *delay* que aguarda em minutos o valor recebido enviando *feedback* regressivo em segundos para o *front-end* e após o passado o tempo de *delay* enviar uma ação para o *back-end* indicando que o tempo acabou.

São recebidos dois dados através do CLP onde um deles é a porcentagem de bateria cujo valor recebido em número real é convertido em um objeto e enviado ao *front-end* e o modo

(se está em manual ou automático) cujo valor é do tipo booleano convertido em objeto e enviado para o *front-end* e o *back-end*;

O último fluxo de dados do middleware são sobre os dados recebidos vindos do *back-end* da seguinte maneira:

a) estado: envia um número inteiro de 0 até 2 referente aos estados de “disponível”, “ocupado” e “ocupado e parado”, esse estado é convertido em objeto e enviado para o *front-end* para informar ao usuário e ao CLP para alterar a cor do led do AMR Clean de acordo com o estado;

b) velocidade linear: envia para o CLP um número real informando a velocidade linear que o AMR Clean deve seguir (número positivo move o robô para frente e negativo para trás);

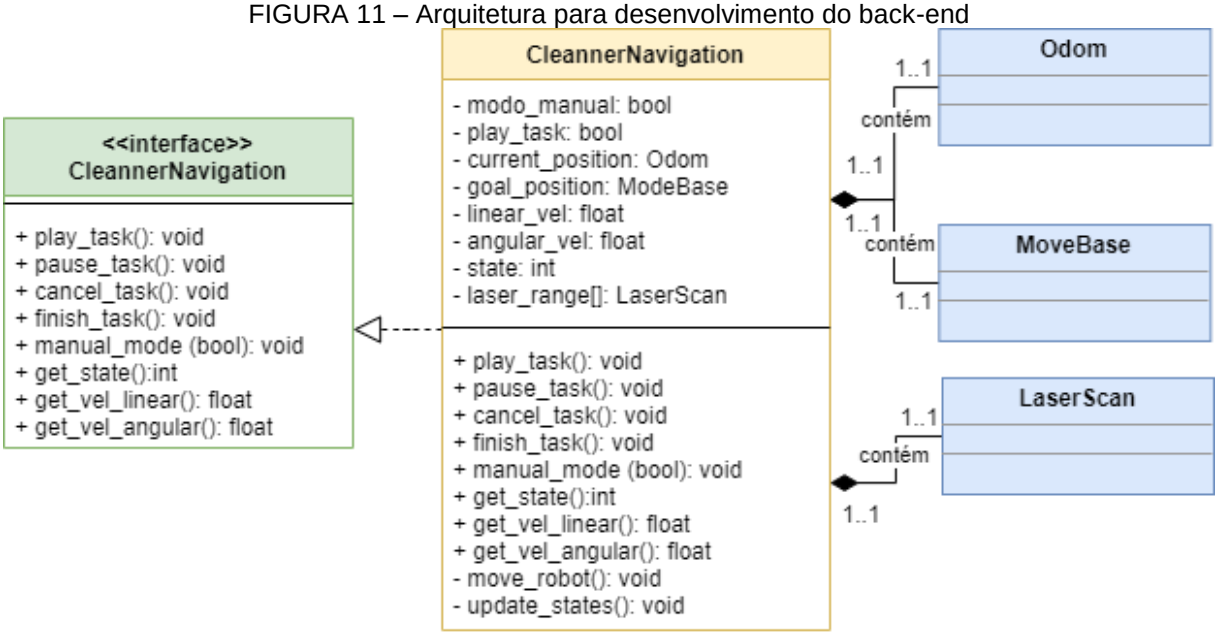
c) velocidade angular: envia para o CLP um número real informando a velocidade a velocidade angular que o AMR Clean deve seguir (número positivo rotaciona o robô no sentido anti-horário e negativo no sentido horário).

A arquitetura do *Middleware* permite que a equipe de desenvolvimento compreenda como irá funcionar o fluxo de comunicação entre o *front-end*, *back-end* e CLP.

4.5 MODELAGEM *BACK-END*

A arquitetura escolhida para modelar o funcionamento do sistema no *back-end* foi a arquitetura de orientada a objetos utilizando o diagrama de classe da UML, e tem como objetivo auxiliar os desenvolvedores a entender o funcionamento dessa parte do sistema, a figura 11 ilustra o modelo do *back-end* construído.

FIGURA 11 – Arquitetura para desenvolvimento do back-end



FONTE: Os autores (2021)

Conforme mostra o modelo o desenvolvimento do *back-end* envolve uma interface e quatro classe, sendo três delas já existentes (desenvolvidas pela Selettra) e uma que será responsável pelo sistema de navegação. Segue abaixo a funcionalidade das três classes existentes:

a) **Odom:** Classe responsável em fornecer coodernadas de localização do AMR Clean (posições x, y e ângulo *theta*);
b) **MoveBase:** Classe responsável por mover o robô para uma posição solicitada através de um método informando posições x, y e *theta* da localização desejada;
c) **LaserScan:** Classe responsável em fornecer informações capturadas pelo escâner Lidar, para cada ângulo de varredura informa através de um array a distância coletada pelo feixe de luz.

A classe “CleannerNavigation” possui oito atributos que são usados para realizar a navegação do robô, segue abaixo a descrição de cada método da classe:

a) **play_task():** ao ser chamado o método verifica se o robô está no modo manual, se a variável “play_task” é verdadeira, se ambos forem falsos altera o valor da variável “play_task” para verdadeiro, “pause_task” para falso, ajusta odometria para as coordenadas (x=0, y=0, theta=0), caso o modo manual seja falso e a variável “play_task” for verdadeiro apenas altera o valor de “pause_task” para falso;
b) **pause_task():** ao ser chamado o método verifica se o robô está no modo manual e se a variável “play_task” for verdadeira, se não estiver no modo manual e a variável “play_task” for verdadeira altera a variável “pause_task” para verdadeiro;
c) **cancel_task():** ao ser chamado o método verifica se o robô está no modo manual e se a variável “play_task” for verdadeira, se não estiver no modo manual e a variável “play_task” for verdadeira altera a variável “pause_task” para verdadeiro e a variável “play_task” para falso;
d) **finish_task():** ao ser chamado o método verifica se o robô está no modo manual e se a variável “play_task” for verdadeira, se não estiver no modo manual e a variável “play_task” for verdadeira envia o AMR Clean para a coordenada (x=0, y=0, theta=0), aguarda o robô atingir seu objetivo, altera a variável “pause_task” para verdadeiro e a variável “play_task” para falso;
e) **manual_mode(bool):** ao ser chamado o método altera o valor da variável “modo_manual” para o valor do recebido como parâmetro;
f) **get_state():** ao ser chamado o método retorna o valor da variável “state”;
g) **get_vel_linear():** ao ser chamado o método retorna o valor da variável “linear_vel”;
h) **get_vel_angular():** ao ser chamado o método retorna o valor da variável “angular_vel”;

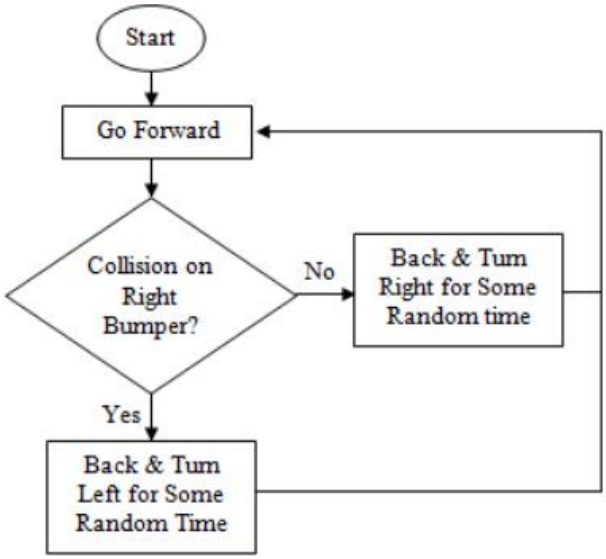
i) **move_robot():** ao ser chamado o método inicia a movimentação do robô verificando os atributos modo_manual, play_task, pause_task, current_position e laser_range e atualiza os valores das variáveis goal_position, linear_vel e angular_vel;

j) **update_states():** ao ser chamado o método atualiza o atributo state de acordo com os atributos modo_manual, play_task, pause_task.

A classe de interface serve para fazer a comunicação com o *middleware* do sistema, coletando e enviando informações.

Para o desenvolvimento do método “move_robot()” será utilizado um algoritmo de navegação usado em AMRs de limpeza, esse algoritmo é conhecido como *Random Walk* (caminhada aleatória) que de acordo com Hasan, Al-Nahid e Reza (2014) é um algoritmo baseado na ideia de movimento aleatório sem seguir nenhum roteiro, onde o robô avança até que algum obstáculo seja detectado e então para sua movimentação, em seguida ele compara as leituras do escâner Lidar decidindo para qual lado girar, e então é gerado um número aleatório definindo o quanto girar, o diagrama da figura 12 ilustra o funcionamento do algoritmo.

FIGURA 12 – Diagrama do algoritmo Random Walk



FONTE: Hasan, Al-Nahid e Reza (2014, p. 4)

Para evitar que o robô fique preso nos cantos existe uma condição chamada “escape de canto” (do inglês, *Corner Scape*) em que o tempo de giro é calculado conforme mostra a equação 1.

EQUAÇÃO 1 – Equação Corner Scape

$$Tempo\ de\ giro = [\frac{R\theta}{720 \cdot rN} + Random\left(0\ até\ \frac{R\theta}{720 \cdot rN}\right) segundos \quad (1)$$

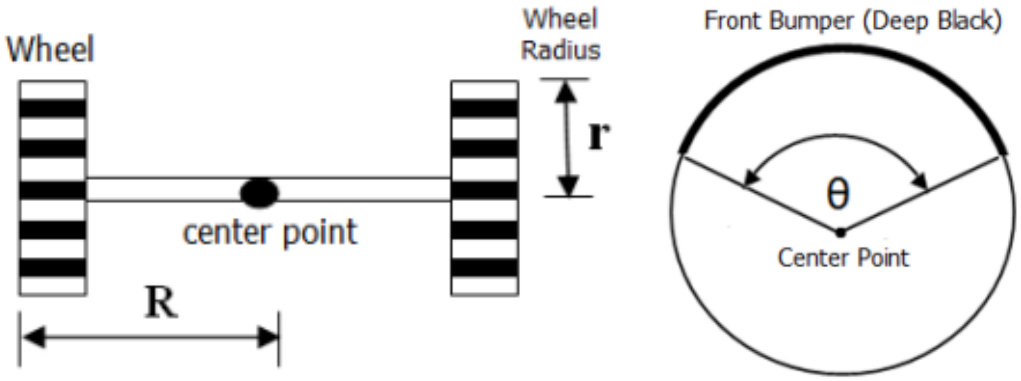
FONTE: Adaptado de Hasan, Al-Nahid e Reza (2014, p. 4)

Onde:

a) r é o raio da roda;
b) R é a distância entre o centro do robô e a roda;
c) N é o número de revoluções da roda por segundo;
d) θ é o ângulo do centro do robô em relação a sua rotação vista de cima.

A figura 13 permite melhor compreensão das variáveis presentes na equação 1. Onde ao lado esquerdo tem-se a referência das juntas do eixo de direção da roda e ao lado direito o ângulo do para-choque dianteiro em relação ao centro.

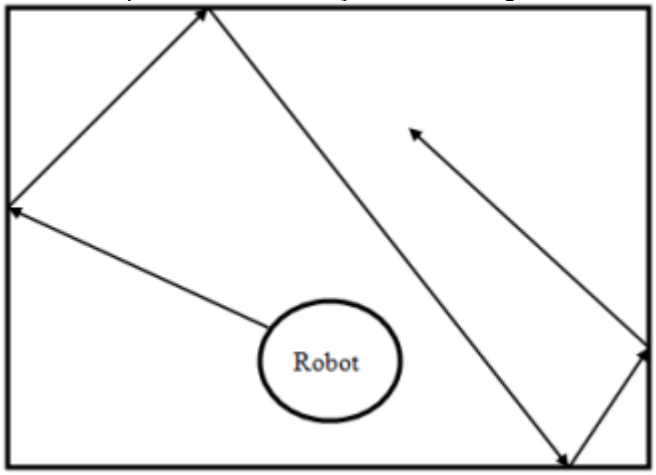
FIGURA 13 – Ilustração das variáveis presentes na equação 1



FONTE: Adaptado de Hasan, Al-Nahid e Reza (2014, p. 4)

A figura 14 apresenta demonstração de como funciona a movimentação do robô ao utilizar o algoritmo de caminhada aleatória.

FIGURA 14 – Exemplo de movimentação usando algoritmo Random Walk



FONTE: Hasan, Al-Nahid e Reza (2014, p. 4)

A arquitetura do *back-end* permitirá que a equipe de desenvolvimento crie essa parte do sistema de maneira modular, podendo ser utilizado em outros sistemas.

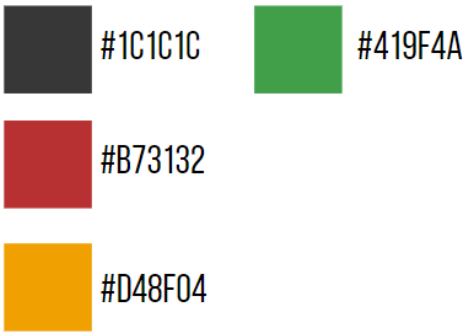
Com a modelagem do sistema realizada iniciou-se a etapa de prototipagem de tela.

5. PROTOTIPAGEM DE TELA

Para desenvolvimento da prototipagem de tela utilizou-se uma ferramenta gratuita chamada Figma[1] que funciona diretamente no navegador de internet de maneira online. A utilização do Figma proporcionou a elaboração do projeto de maneira colaborativa em equipe, otimizando o processo de prototipação.

No início do processo de prototipagem desenvolveu-se qual estilo seria utilizado nas telas do sistema e qual seria o padrão de cores, ícones, fonte e botões, a figura 15 ilustra o padrão de cores escolhido pela equipe.

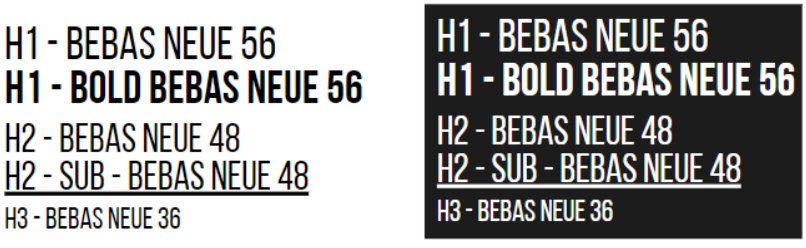
FIGURA 15 – Cores do protótipo de tela do sistema



FONTE: Os autores (2021)

As cores foram escolhidas de acordo com a logomarca da empresa Selettra, que possui tonalidades de vermelho, cinza, preto e branco. Além disso por se tratar de um robô móvel buscou-se as cores de sinalizadores de trânsito (verde, amarelo e vermelho) para representar quando o robô está disponível, ocupado, parado ou em movimento. Após escolhido as cores, foi definido qual seria a fonte utilizada e quais seriam seus tamanhos. A figura 16 mostra a tipografia escolhida para o protótipo.

FIGURA 16 – Tipografia do protótipo de tela do sistema



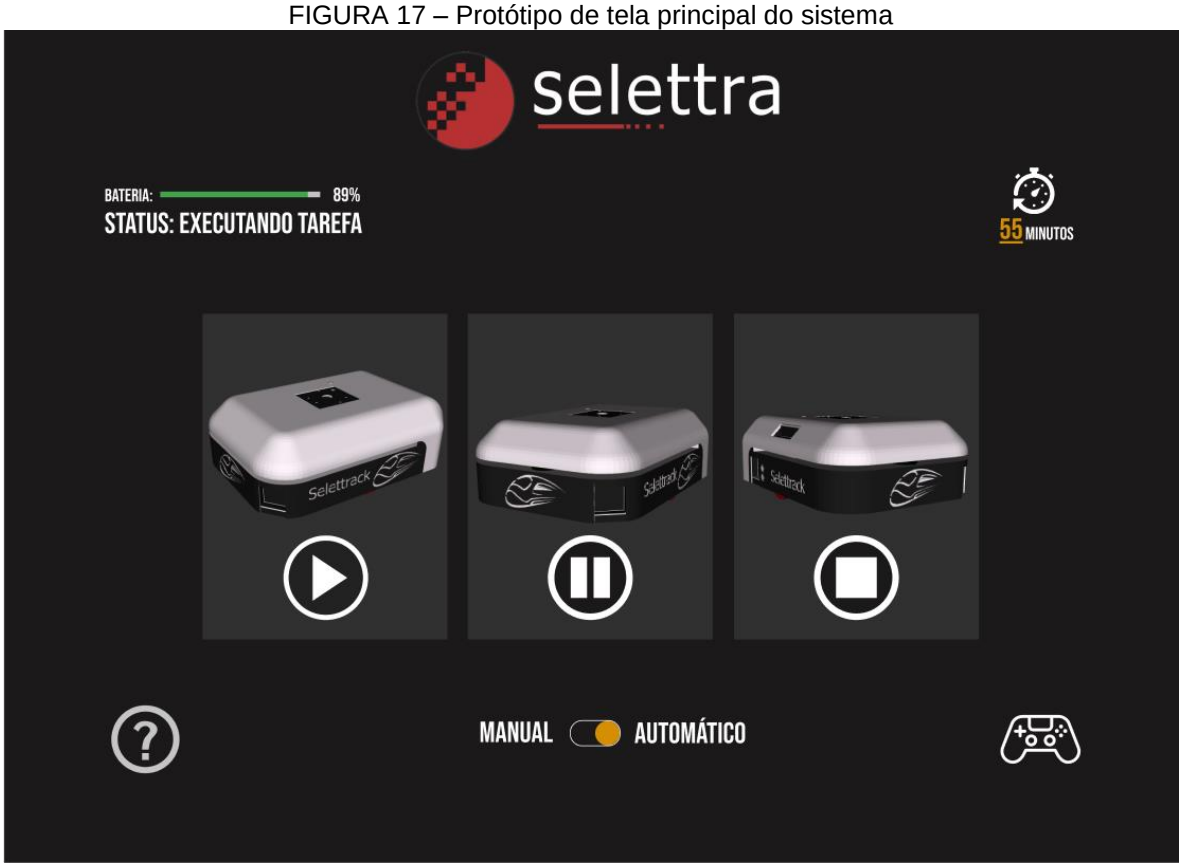
FONTE: Os autores (2021)

[1] https://www.figma.com

Escolheu-se a fonte Bebas Neue para ser a fonte principal do sistema, primeiramente por questões estéticas e por ser uma fonte disponibilizada pela google, que possibilita fácil acesso, uso e importação da fonte. O tamanho da fonte foi definido através de experiência de uso entre os membros da equipe.

Após definir o guia de estilo, foi criado a tela principal do sistema, conforme mostra a figura 17.

FIGURA 17 – Protótipo de tela principal do sistema



FONTE: Os autores (2021)

Buscou-se fazer a tela principal de maneira simples e intuitiva contemplando os componentes presentes na modelagem, permitindo que o usuário seja capar de mudar e visualizar o modo do robô (manual/automático) e o tempo de execução de tarefa, visualizar o estado do robô (disponível, ocupado, ocupado e em pausa), a porcentagem de bateria e ir para tela de controle manual. Além disso colocou-se um ícone de ajuda que ao clicar ou passar com o mouse por cima exibe descrição de cada funcionalidade da tela, a figura 18 mostra como é exibido as informações de ajuda.

FIGURA 18 – Protótipo de exibição de ajuda na tela principal do sistema

FONTE: Os autores (2021)

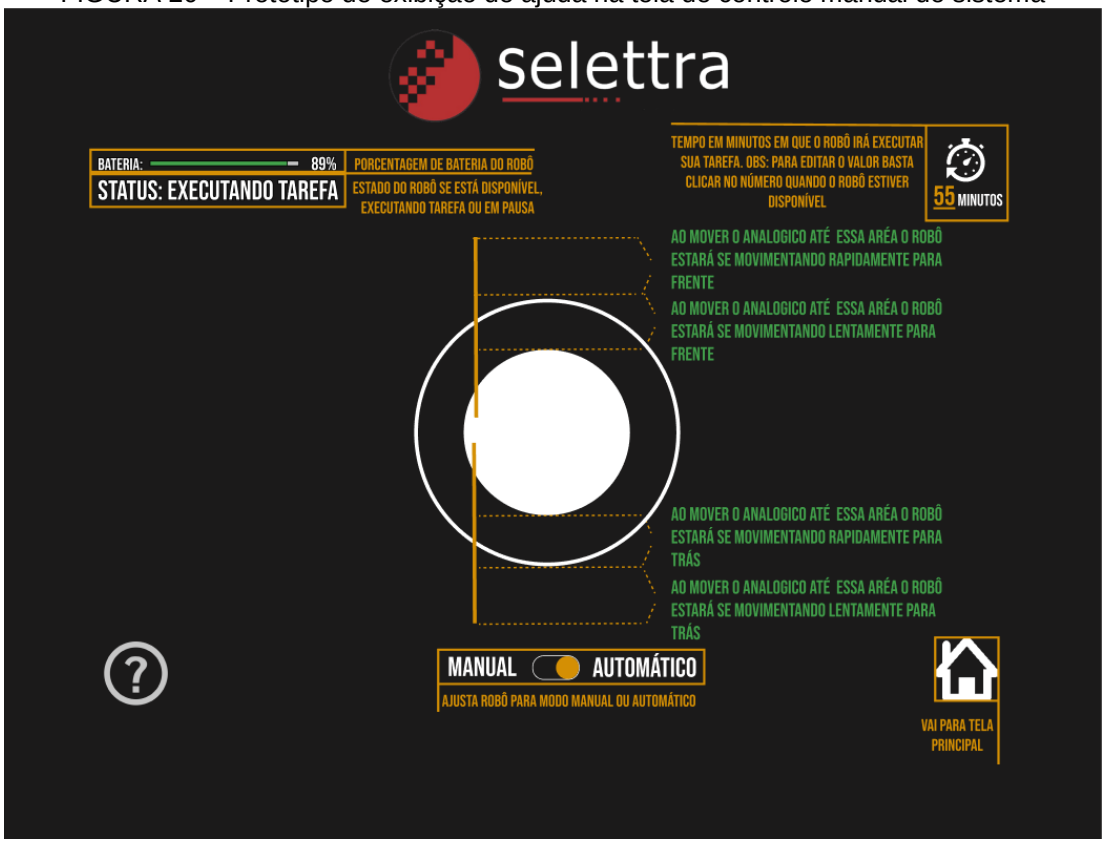
Após a tela principal foi desenvolvido a tela de controle manual, conforme ilustra a figura 19.

FIGURA 19 – Protótipo de tela para modo manual do sistema

FONTE: Os autores (2021)

Na tela de modo manual são retirados os comandos de execução de tarefas e adicionado um controlador direcional circular, onde o usuário pode controlar o robô clicando (ou pressionando o dedo) ao centro do controlador e arrastando para a posição em que deseja a movimentação do robô. Na tela de modo manual também há o ícone de ajuda que descreve a funcionalidade dos componentes conforme ilustra a figura 20.

FIGURA 20 – Protótipo de exibição de ajuda na tela de controle manual do sistema



FONTE: Os autores (2021)

Após a o desenvolvimento do protótipo de telas do sistema foi agendado reunião com um representante da empresa Selettra para validação das telas criadas e possíveis alterações. Com as telas avaliadas e validadas pela empresa iniciou-se o processo de desenvolvimento através da escolha do ambiente de simulação a ser utilizado pela equipe para demonstrar o funcionamento do sistema.

6. AMBIENTE DE SIMULAÇÃO

Para realização da navegação do AMR Clean em um ambiente de simulação escolheu-se a uma ferramenta chamada gazebo[2], que de acordo com site do *software* é um simulador capaz

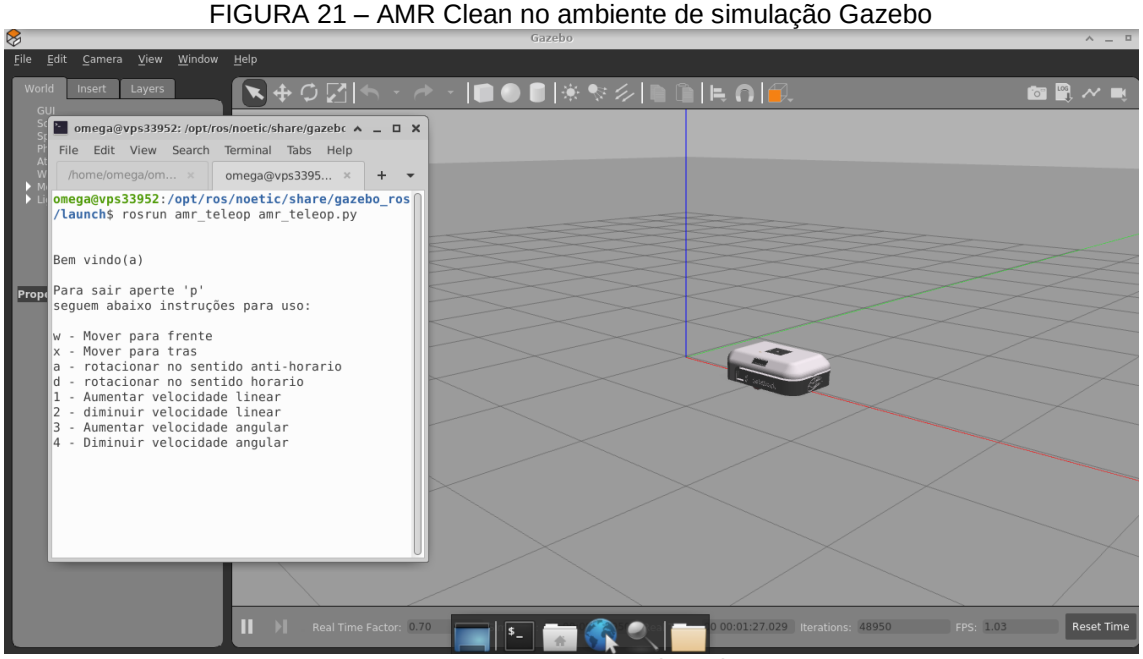
[2] http://gazebosim.org

de simular com precisão e eficiência populações de robôs em ambientes internos e externos (OPEN SOURCE ROBOTICS FOUNDATION, 2014).

A escolha desta ferramenta deve-se ao fato de ser compatível com o sistema já existente e pela empresa Selettra ser capaz de fornecer uma versão virtual do AMR Clean compatível com ela.

A figura 21 mostra a versão virtual do AMR Clean em um mapa vazio no ambiente de simulação Gazebo.

FIGURA 21 – AMR Clean no ambiente de simulação Gazebo



FONTE: Os autores (2021)

O ambiente de simulação Gazebo permite a construção de mapas para testes com robôs, que podem ser exportados e importados pelos usuários da ferramenta.

Para fins de testes a equipe escolheu um mapa disponibilizado em um repositório público pela empresa aws-robotics[3] chamado small house world[4].

[3] https://github.com/aws-robotics

[4] https://github.com/aws-robotics/aws-robomaker-small-house-world

FIGURA 22 – Mapa Gazebo aws-robomaker-small-house-world

FONTE: Os autores (2021)

Com o ambiente de simulação definido iniciou-se a etapa de desenvolvimento do sistema de acordo com a modelagem realizada e o protótipo de telas.

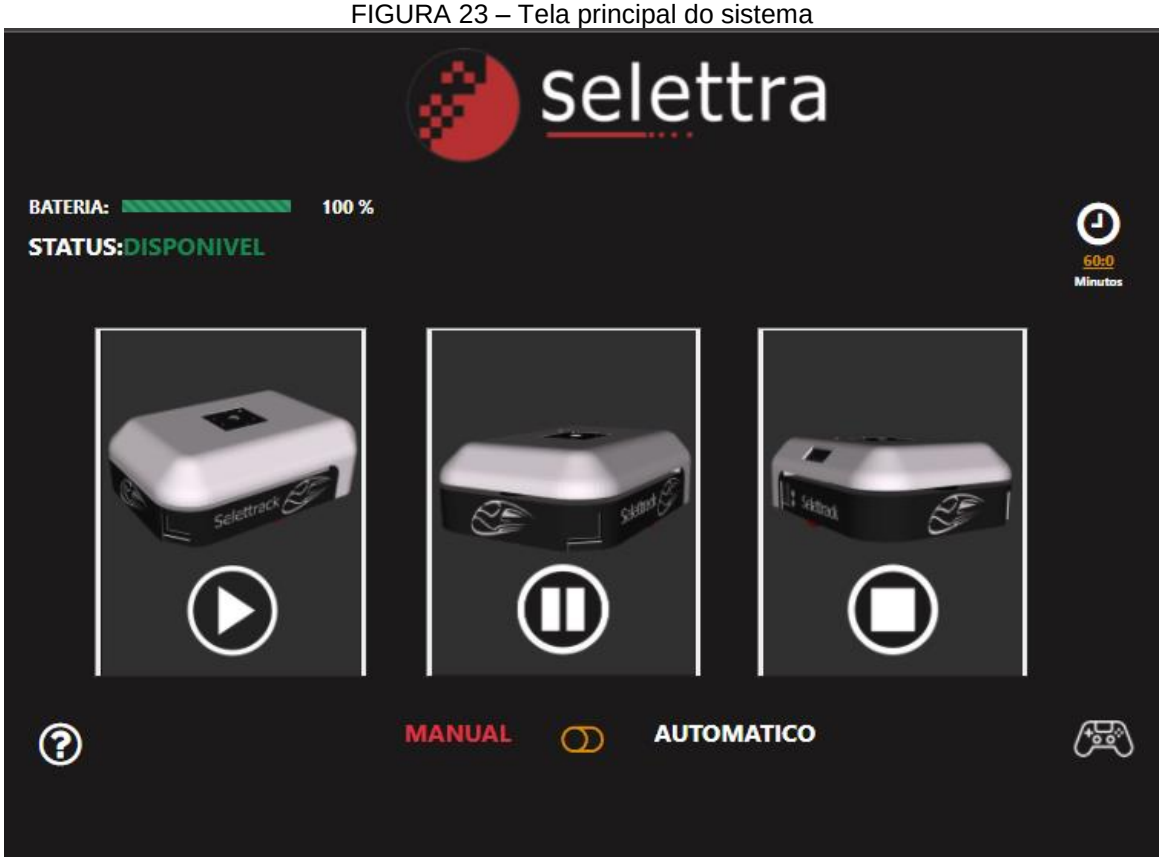
7. DESENVOLVIMENTO *FRONT-END*

A interface de usuário foi desenvolvida no formato de aplicação *web* tendo como servidor para aplicação uma ferramenta chamada node-red[5]. Para o desenvolvimento utilizou-se dois frameworks de desenvolvimento *web*, o primeiro chamado vue.js, que de acordo com seu site oficial é uma estrutura progressiva para criação de interface de usuários (VUE.JS, 2021) e o segundo chama-se Bootstrap que é um framework popular de estilo para desenvolvimento de aplicações *web* (BOOTSTRAP, 2021).

As telas da aplicação foram construídas buscando deixá-las igual ao protótipo de telas, a figura 23 o resultado obtido na tela principal.

[5] Informações sobre a ferramenta encontra-se no capítulo 9.

FIGURA 23 – Tela principal do sistema



FONTE: Os autores (2021)

Na tela principal o usuário consegue programar o tempo em que o robô irá realizar sua tarefa, iniciar, parar e cancelar uma tarefa existente, além de poder colocá-lo em modo manual, exibir feedback de estado do robô e opção de ajuda.

A figura 24 mostra a tela de controle manual do robô.

FIGURA 24 – Tela de controle manual do sistema

FONTE: Os autores (2021)

No controle manual o usuário pode mover o robô manualmente através do uso de um joystick na tela.

Os arquivos desenvolvidos no *front-end* encontram-se em um repositório público no GitHub[6] acessível através do endereço *web*: <https://github.com/Hobbies-Prof-Bento/random_walk_navigation/tree/main/frontend/random_walk>.

7.1 AVALIAÇÃO E TESTES *FRONT-END*

Foram feitos testes de unidade em cada parte desenvolvida e a abordagem de testes escolhida foi a *botton-up* que para Pressman e Maxim (2016) é onde cada componente criado ou programado é testado e depois trata-se o conjunto de componentes até que todo o sistema seja testado.

Para avaliar a funcionalidade e a qualidade de desenvolvimento do *front-end* e a interação entre o usuário e sistema utilizou-se a Avaliação Heurística.

O método chamado Avaliação Heurística de acordo com Barbosa e Silva (2011) trata-se de um método de avaliação criado para identificar problemas de usabilidade no processo de *design* interativo. O método orienta avaliadores a inspecionar a interface de maneira sistemática em busca de problemas que prejudiquem a usabilidade. A avaliação heurística tem como base um conjunto de diretrizes descrevendo características desejáveis da interação e da interface, conhecidas como heurísticas de Nielsen.

É recomendado que a avaliação heurística seja feita de três a cinco avaliadores, com algumas atividades a serem realizadas individualmente enquanto outras em conjunto. Na avaliação cada avaliador deve analisar se há alguma violação nas diretrizes, caso encontre alguma violação devem apontar essa violação junto com o grau de severidade (ou gravidade). O julgamento da severidade envolve três fatores: a frequência que ocorre, impacto do problema se ocorrer e a persistência do problema (BARBOSA; SILVA, 2011).

O processo de avaliação heurística iniciou-se com a avaliação individual de cada tela. Os avaliadores trouxeram os resultados de suas avaliações individuais apontando as violações e níveis de severidade encontradas nas telas. Após os apontamentos foi discutido e realizado correções na interface.

8. DESENVOLVIMENTO *BACK-END*

O *back-end* foi desenvolvido utilizando um *framework* chamado *Robot Operating System* (ROS) que de acordo com o site oficial é um sistema operacional para robôs, o qual disponibiliza bibliotecas e ferramentas para ajudar desenvolvedores de software a criar aplicações robóticas.

[6] Para informações sobre GitHub acesse: https://github.com/

A empresa Selettra disponibilizou as pastas de códigos para virtualização do robô, que junto com as pastas coletadas da AWS-robotics permitiu que a equipe desenvolvesse apenas os algoritmos de navegação.

Foi escolhido a linguagem de programação Python para o desenvolvimento dos algoritmos de navegação, onde foram criados dois arquivos um destinado ao controle do robô, desde a coleta de dados dos sensores até a sua movimentação, e outro para operacionalizar a navegação, recebendo comandos de início, parada, cancelamento e finalização de uma tarefa, fornecendo feedback do estado e solicitando movimentação do robô. Além disso foi criado um arquivo para parametrização de velocidade linear, angular, quadros do robô, distância em que o robô pode se aproximar de um objeto e os tópicos que o robô assina e publica.

Além dos arquivos fornecidos pela Selettra e AWS-robotics foram necessários adicionar os arquivos “slam-gmapping”[7] e “move-base"[8], o primeiro responsável em usar sensores do robô para criar mapa e localizar o robô dentro do mapa, e o outro para solicitar que o robô se mova para determinada posição do mapa, fornecidos pela comunidade de desenvolvimento em ROS.

Os arquivos desenvolvidos no *back-end* encontram-se em um repositório público no GitHub acessível através do endereço *web*: <https://github.com/Hobbies-Prof-Bento/random_walk_navigation/tree/main/backend/cleaner_navigation>.

8.1 AVALIAÇÃO E TESTES BACK-END

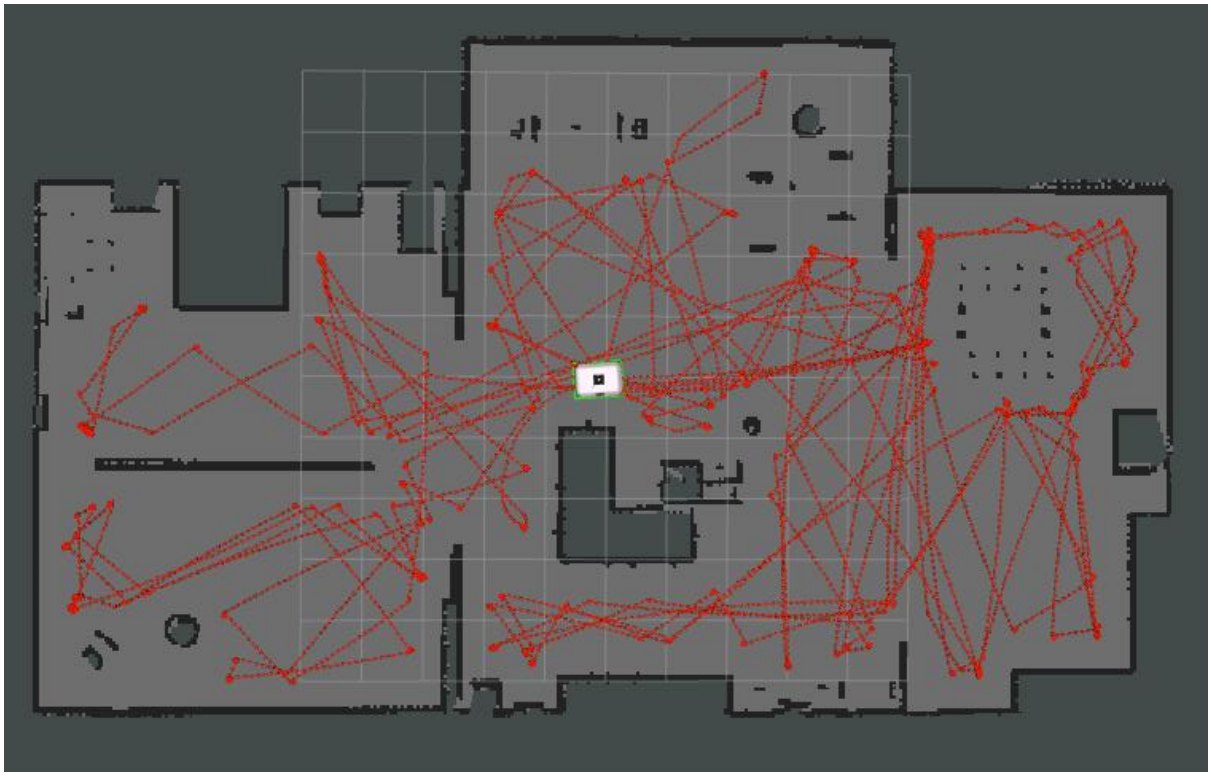
Foram feitos testes de unidade em cada parte desenvolvida e depois de pronto foi realizado um teste em todas as unidades juntas. A abordagem de testes escolhida foi a botton-up.

Na etapa final de teste do back-end utilizou-se uma ferramenta de visualização 3D para ROS chamada Rviz, essa ferramenta permite exibir graficamente a versão virtual do robô, bem como a representação de dados coletados de seus sensores.

Com a ajuda do Rviz através dos dados de odometia do foi possível criar o rastro de por onde o robô movimentou-se no simulador durante sessenta minutos a partir do comando de iniciar a navegação, a figura 25 mostra o resultado obtido.

[7] Para mais informações sobre slam-gmapping: http://docs.ros.org/en/hydro/api/gmapping/html/index.html
[8] Para mais informações sobre move-base: http://wiki.ros.org/move_base

FIGURA 25 – Rastro de movimento do robô dentro do simulador



FONTE: Os autores (2021)

Conforme mostra a figura 25 o robô percorreu por todos os cômodos do mapa atendendo assim os requisitos do sistema.

9. DESENVOLVIMENTO *MIDDLEWARE*

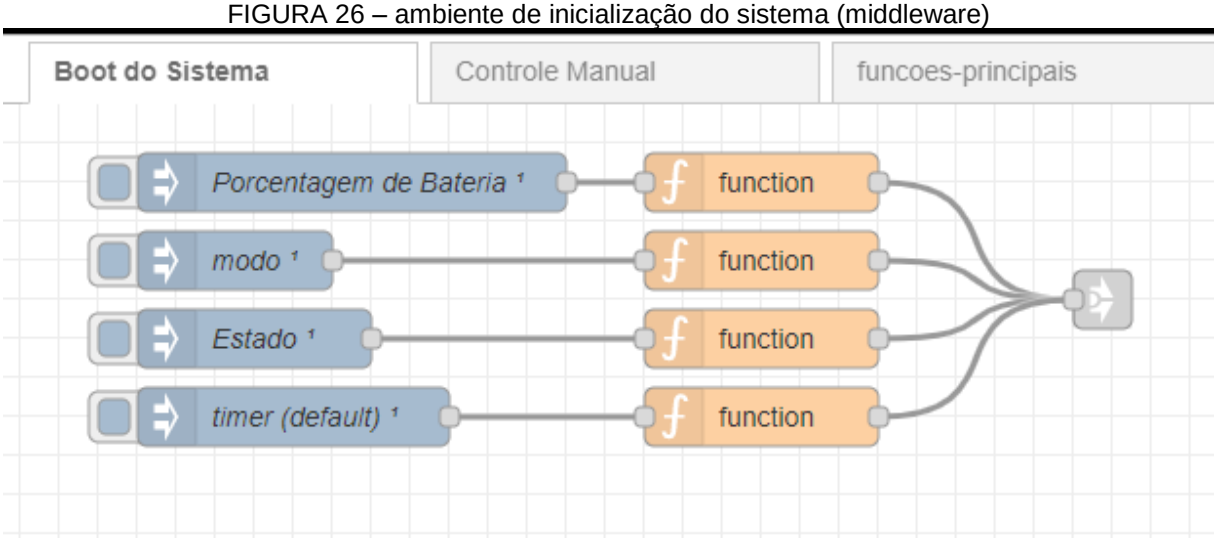
Para o desenvolvimento do *Middleware* foi utilizado uma ferramenta de programação chamada node-red[9], que fornece através do navegador de internet uma interface baseada em nós facilitando a junção de fluxos utilizados pelo sistema.

Com a finalidade de organizar o desenvolvimento do sistema a programação foi dividida em cinco ambientes, sendo eles dedicados ao início do sistema, controle manual do robô, funções que são realizadas durante o processo de execução de tarefa do robô, comunicação com o *back-end* e comunicação com o *front-end*.

A figura 26 mostra o ambiente de inicialização do sistema.

[9] Para mais informações sobre *node-red* acessar: https://nodered.org/

FIGURA 26 – ambiente de inicialização do sistema (middleware)

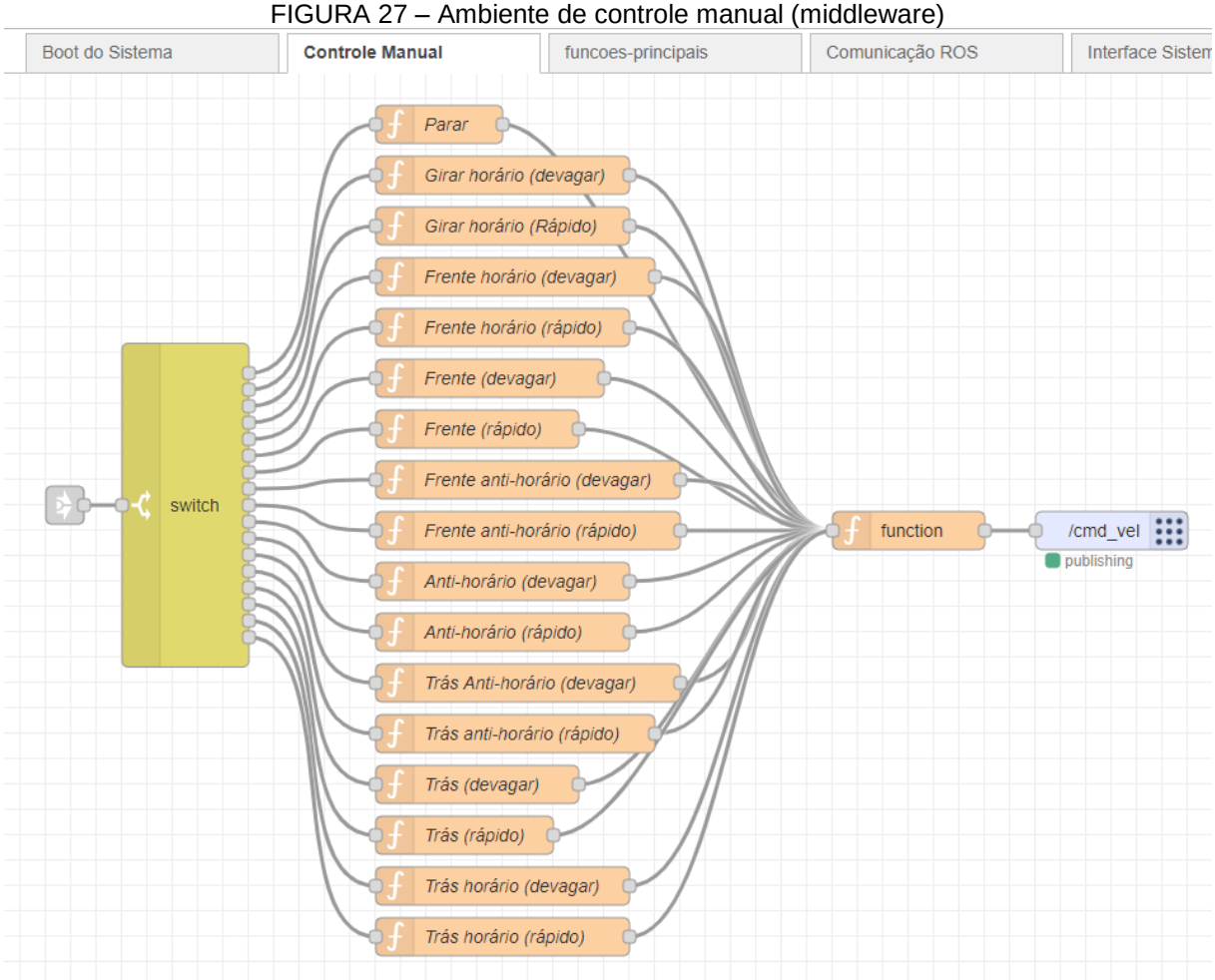


FONTE: Os autores (2021)

No ambiente de inicialização do sistema é enviado valores iniciais de porcentagem de bateria, modo, estado e tempo padrão para realização da tarefa. Depois da inicialização os valores enviados são alterados de acordo com a utilização do sistema.

A figura 27 mostra os nós criados para o controle manual.

FIGURA 27 – Ambiente de controle manual (middleware)

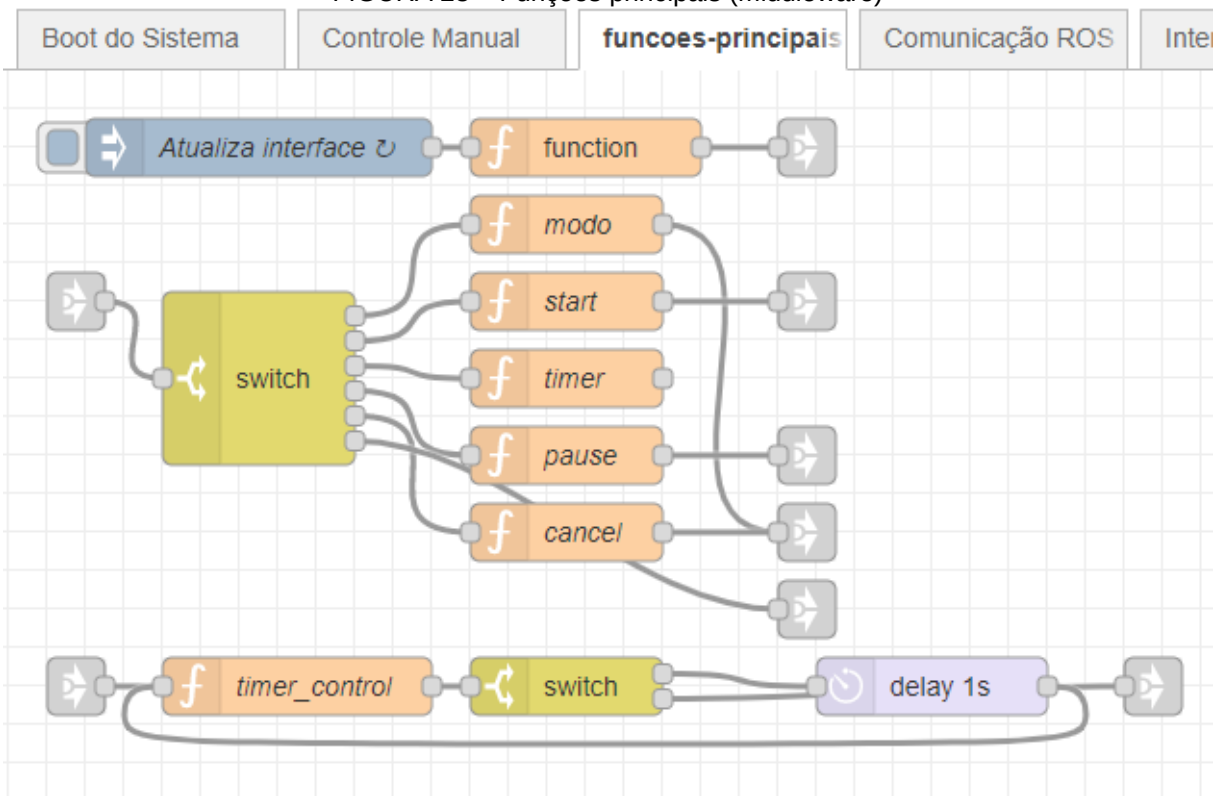


FONTE: Os autores (2021)

No ambiente de controle manual o nó chamado “switch” recebe dados da interface (*front-end*) e dependendo do valor encaminha para a função correspondente, que por sua vez envia para o *back-end*. Esse processo é feito para o simulador, para o caso do robô real o valor é enviado para o controlador lógico programável (CLP).

A figura 28 mostra as funções realizadas para a execução de tarefa do robô, conforme o modelo do capítulo 4.

FIGURA 28 – Funções principais (middleware)

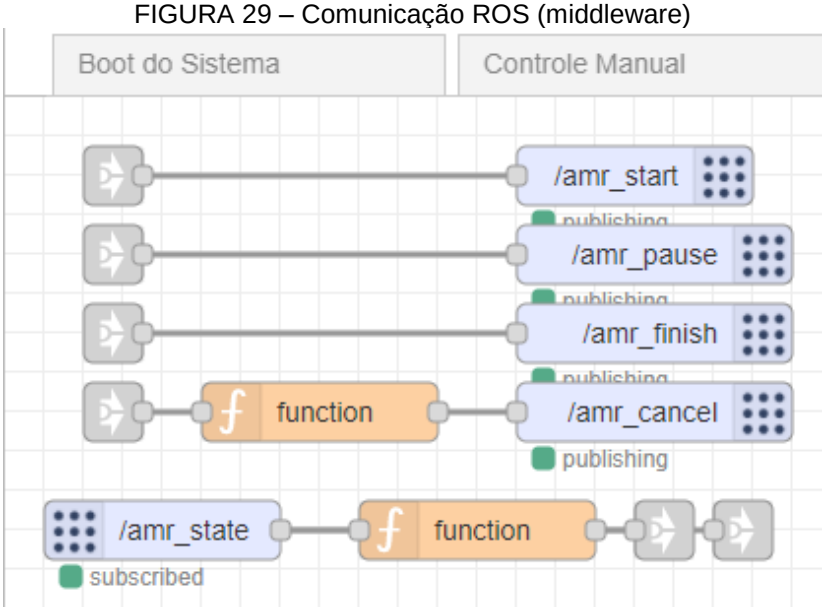


FONTE: Os autores (2021)

Nas funções principais tem uma função responsável em atualizar constantemente as informações da interface, um “switch” que filtra os dados recebidos pela interface (*front-end*), modifica e envia para seu devido destino e uma função responsável em atualizar o temporizador enviando comando para encerrar a tarefa em que o robô está executando.

A figura 29 mostra o ambiente de comunicação com o *back-end*.

FIGURA 29 – Comunicação ROS (middleware)

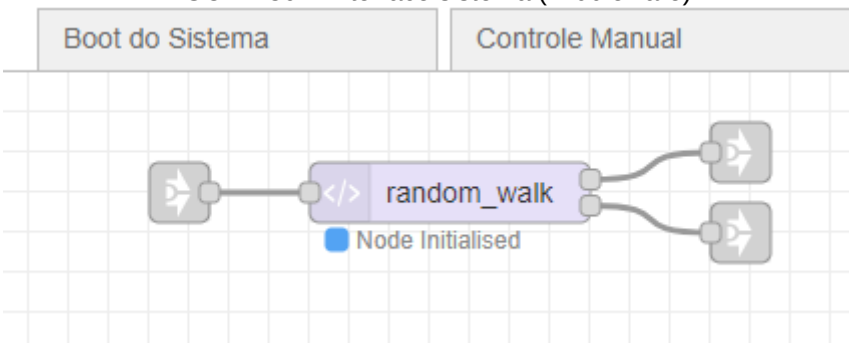


FONTE: Os autores (2021)

No ambiente de comunicação com o *back-end* os dados já modificados são enviados e os dados de estado do robô são recebidos e enviados para o front-end.

A figura 30 mostra o ambiente responsável pela comunicação com o front-end.

FIGURA 30 – Interface sistema (middleware)



FONTE: os autores (2021)

No ambiente de comunicação com o *front-end* os dados modificados são enviados para a interface do usuário e os dados recebidos são enviados para seus devidos lugares.

Os arquivos desenvolvidos no middleware encontram-se em um repositório público no GitHub acessível através do endereço *web*: < https://github.com/Hobbies-Prof-Bento/random_walk_navigation/tree/main/middleware>.

9.1 AVALIAÇÃO E TESTES MIDDLEWARE

Foram feitos testes de unidade em cada parte desenvolvida e depois de pronto foi realizado um teste em todas as unidades juntas. A abordagem de testes escolhida foi a *botton-up*.

Com os testes foi possível corrigir divergências na comunicação entre o *middleware* com o *back-end* e o middleware com o *front-end*, possibilitando assim testar o sistema como um todo.

10. RESULTADOS OBTIDOS

Após o desenvolvimento e a realização de testes de cada parte do sistema, foi realizado testes do sistema[10] como um todo, o qual proporcionou alguns resultados relevantes. A figura 31 mostra a captura de tela de teste do sistema.

FIGURA 31 – Captura de tela de teste do sistema

FONTE: Os autores (2021)

O primeiro resultado obtido ao utilizar o sistema no ambiente simulado foi a altura do escâner do robô, fazendo com que o robô entre em colisão com objetos abaixo da posição onde estão os feixes de luz. Uma solução apresentada para essa situação é adicionar um outro sensor com feixes de luz verticais e horizontais, assim é possível detectar objetos com altura próxima ao chão evitando colisão em objeto acima e abaixo da altura do robô.

Outro ponto a ser ressaltado é que o sistema foi desenvolvido e testado em um ambiente de simulação, devido a isso pode haver situações no robô real que não foram previstas no simulador, como por exemplo o tempo de atraso dos dados enviados pelos sensores e algum tipo de interferência elétrica ou magnética em seus componentes comprometendo a qualidade dos dados recebidos e enviados.

[10] Segue link com vídeo do funcionamento do sistema: https://drive.google.com/file/d/10R-trLfyYDd8Zw_K1-UH49mRosBa4Qny/view?usp=sharing

O restante dos resultados obtidos fora satisfatório, qual se corrigido as situações supracitadas é possível comercializar o sistema.

O sistema completo desenvolvido encontra-se em um repositório no GitHub[11], onde é possível que qualquer desenvolvedor com conhecimento em ROS, aplicação *web* e node-red possa aplicá-lo em seus sistemas de robótica móvel.

11. CONSIDERAÇÕES FINAIS

Com a pesquisa realizada foi possível perceber que diante ao quadro de pandemia do Covid-19 que estamos enfrentando, a robótica móvel tem um papel importante a cumprir, auxiliando na esterilização de ambientes através de dispositivos que emitem raios UV. Notou-se que com todo o processo de engenharia de software (levantamento de requisitos, modelagem, prototipagem, entre outros) pode-se desenvolver um sistema de navegação com qualidade para atingir esse objetivo.

Não houve dificuldades em redigir o presente trabalho integrador, todos os membros da equipe participaram integralmente da pesquisa de maneira em que nenhum ficou responsável por apenas um capítulo ou assunto, o que proporcionou melhor integração e praticidade nas entregas.

O desenvolvimento de um protótipo e a utilização do ambiente de simulação proporcionou interatividade entre os participantes da equipe, bem como a consciência de que um projeto idealizado mentalmente pode ser na prática diferente para pessoas distintas. Mesmo sendo de maneira virtual ajudou a alinhar e definir os detalhes e funcionalidades de um sistema junto às partes interessadas.

Para trabalhos futuros sugere-se a aplicação do sistema em um robô real com a finalidade de apresentar quais foram as alterações utilizadas para o funcionamento, além da criação de *design* responsivo para a aplicação *web* permitindo o uso em *smartphones* e *tablets*.

12. REFERÊNCIAS

ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. de. **Fundamentos da programação de computadores:** algoritmos, Pascal, C/C++ (padrão ANSI) e Java. 3 ed. São Paulo: Pearson, 2012.

AMBROSE, Gavin; HARRIS, Paul. **Design thinking**. Tradução de Mariana Belloi. rev. Porto Alegre: Bookman, 2011.

BARBOSA, Fabrício F. M.; FREITAS, Pedro H. C. **Modelagem e desenvolvimento de banco de dados**. 1 ed. Porto Alegre: Sagah, 2018.

[11] Link de acesso ao repositório:
https://github.com/Hobbies-Prof-Bento/random_walk_navigation

BARBOSA, Simone D. J.; SILVA, Bruno S. da. **Interação humano-computador**. 1ed. Rio de Janeiro: Elsevier, 2011.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. **Software architeture in Practice**. 3 ed. massachusetts:Addison-Wesley Professional, 2012.

BENTO, Clístenes G. et. al. Estudo exploratório para desenvolvimento de sistema destinado a gerenciamento de clientes e orçamentos em uma oficina mecânica localizada em São José dos Pinhais. **Inova + Caderno de Graduação da Faculdade da Indústria**. São José dos Pinhais – PR, V. 2, n. 2, p. 208-229, 2021.

BOOTSTRAP. **Introduction**. Disponível em: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>. Acesso em: 20 nov. 2021.

DUDEK, Gregory; JENKIN, Michael. ***Computational Principles of Mobile Robotics***. 2 ed. New York: *Cambridge University Press*, 2010.

FOWLER, Martin. **UML Essencial: Um breve guia para linguagem-padrão de modelagem de objetos**. 3 ed. Tradução de João Tortello. São Paulo: Bookman, 2014.

EIS, Diego; FERREIRA, Elcio. **HTML5 e CSS3 com farinha e** pimenta. 1 ed. São Paulo: Tabless, 2012.

FRANCO, Camila A. G. S. **Ferramenta Computacional para o Apoio ao Processo de Acompanhamento de Aquisição e Desenvolvimento das Competências do Aluno do Curso de Medicina**. 142 f. Dissertação (Mestrado em Tecnologia em Saúde) - Pontifícia Universidade Católica do Paraná, Curitiba, 2012.

GIL, Antonio C. **Como Elaborar Projetos de Pesquisa**. 6 ed. São Paulo: Atlas, 2017.

GUEDES, Gilleanes T. A. **UML 2:** uma abordagem prática. 3 ed. São Paulo: Novatec Editora, 2011.

HASAN, Kazi M.; AL–NAHID, Abdullah; REZA, Khondker J. Path planning algorithm development for autonomous vaccuum cleaner robots. International COnference On Informatics, Electronics & Vision, 3, 2014, Dhanka – Bangladesh. **Anais ...** *sl*: CNSER, 2014.

HEINEN, Farlei J. **sistema de controle híbrido para robôs móveis autônomos**. 130 f. Dissertação (mestrado) – Universidade do vale do Rio dos Sinos, São Leopoldo, 2002.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S. **Sistema de banco de dados**. 6 ed. Rio de Janeiro: campus, 2012.

LISBÔA, Maria G. P.; GODOY, Leoni P. Aplicação do método 5W2H do processo produtivo do produto: a joia. **Iberoamerican Jornal od Industrial Engineering**. Florianópolis – SC, V. 4, n. 7, p. 32-47, 2012.

MARCONI, Marina de A.; LAKATOS, Eva M. **Fundamentos de Metodologia Científica**. 8. ed. São Paulo: Atlas, 2017.

MACHADO, Felipe N. R. **Análise e gestão de requisitos de software**: onde nascem os sistemas. 1 ed. São Paulo: Érica, 2011.

MACHADO, Felipe N. R. **Banco de dados:** projeto e implementação. 4 ed. São Paulo: Érica, 2020.

MACHADO, Luis C. **Modelo conceitual integrando prototipagem rápida e delineamento de experimentos na concepção de novos produtos**. 95 f. Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2013.

MANZANO, José A. N. G. **Programação de computadores com C++**. 1 ed. São Paulo: Érica, 2014.

MARIANO, Diego; MELO-MINARDI, Raquel. **Introdução à Programa Web para Bioinformática: HTML, CSS, PHP e JavaScript.** 1 ed. Belo Horizonte: KDP Amazon, 2017.

MARTINS, José C. C. **Gerenciando Projetos de desenvolvimento de Software com PMI, RUP e UML.** 5 ed. Rio de Janeiro: Brasport, 2010.

MAZUR, Viviane T. et al. Desenvolvimento de dispositivos de esterilização com luz UV-C para hospital de referência no tratamento de COVID-19, In: CONGRESSO BRASILEIRO DE ENGENHARIA DE PRODUÇÃO, 10, 2020, evento on-line. **Anais...** Curitiba: APREPRO, 2020.

MEDEIROS, Ernani. **Desenvolvendo software com UML 2.0:** definitivo. São Paulo: Pearson Makron Books, 2010.

MINISTÉRIO DA SAÚDE. **Painel Coronavírus**. Disponível em <https://covid.saude.gov.br/>. Acesso em 28 set. 2021.

MORAIS, Izabelly S. de *et al*. **Introdução a big data e internet das coisas (IoT)**. Porto Alegre: SAGAH, 2018.

OPEN SOURCE ROBOTICS FOUNDATION. **Why Gazebo?** Disponível em < http://gazebosim.org>. Acesso em: 11 out. 2021.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**: uma abordagem professional. 8.ed. Porto Alegre: AMGH, 2016.

PUREWAL, Semmy. **Learning web app development:** build quickly with proven javascript techiniques. 1 ed. United States: O'Reilly, 2014.

SANTOS, Robson L. G. **Usabilidade de interfaces para sistemas de recuperação de informação na web:** estudo de caso de bibliotecas on-line de universidades federais brasileiras. 347 f. Tese (Doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

SILVA, Alan C. da. et al. **Trabalho final de curso:** Esterilização por luz ultravioleta C (UV-C). 105 f. Trabalho de conclusão de curso (graduação) – Centro Iniversitário FEI, São Bernardo do Campo, 2021.

VUE.JS. **Whats is Vue.js**. Disponível em: <https://vuejs.org/v2/guide/>. Acesso em: 20 nov. 2021.

W3SCHOOLS.COM. **HTML Introduction**. Disponível em: < https://www.w3schools.com/html/html_intro.asp>. Acesso em: 29 set. 2020.

ZENKER, Aline M *et al*. **Arquitetura de sistemas**. 1 ed. Porto Alegre: SAGAH, 2019.