Sistema FIEB | SENAI CIMATEC

PELO FUTURO DA INOVAÇÃO

CENTRO UNIVERSITÁRIO SENAI CIMATEC

**Relatório Final**

# Desafios para o laboratorio robotica e sistemas autonomos

Apresentada por: Tiago Barretto Sant'Anna

Orientado por: Prof. Marco Reis, M.Eng.

Dezembro de 2021

Tiago Barretto Sant'Anna

# Desafios para o laboratorio robotica e sistemas autonomos

Salvador
Centro Universitário SENAI CIMATEC
2020

# Resumo

O trabalho consiste na explanação das percepções acerca dos desafios passados. Assim conta como foi desenvolvido e as táticas utilizadas para a conclusão dessa atividade. Tem como objetivo a absorção de conhecimentos com o intuito de realizar projetos de robótica. Para isso foram realizados estudos de uma gama de conhecimentos para poder realizar as atividades, sendo entre eles programação, simulação e *ROS*.

**Palavras-chave**: robótica, estudo, programação, simulação.

# Abstract

The work consists of explaining perceptions about past challenges. So it tells how it was developed and the tactics used to complete this activity. It aims to absorb knowledge in order to carry out robotics projects. For this, studies were carried out on a range of knowledge to be able to carry out the activities, including programming, simulation and *ROS*

**Keywords**: robotics, study, programming, simulation.

# Sumário

# Lista de Figuras

# Lista de Tabelas

Capítulo Um

# Introdução

Este artigo consiste na exposição dos desafios realizados para o Laboratório de Robótica e Sistemas Autônomos. Assim, esses desafios foram realizados ao longo de dois meses. Esses desafios tem como principal função realizar a capacitação para pode atuar dentro do laboratório

## *1.1 Objetivos*

Os objetivos são realizar desafios de robótica, programação  e simulação com a função de adquirir conhecimentos para poder atuar dentro do laboratório.

### *1.1.1 Objetivos Específicos*

Os objetivos específicos deste projeto são:

utilizá-loAprender ROS e como utiliza-lo; Desenvolver habilidades de programação em Python e C++; Obter conhecimento de simulação e programação de robôs usando o Webots; Criar um package no ROS para controlar o turtlesim; Obter conhecimentos a cerca de navegação usando o Husky

## *1.2 Justificativa*

Esse trabalho tem como função de capacitar o estudante acerca das ferramentas necessarias para trabalhar com robótica, sendo aprendendo a utilizar o framework ROS com a versão noetic, ou habilitando seu conhecimento em programação.

## *1.3 Organização do documento*

Este documento apresenta 5 capítulos e está estruturado da seguinte forma:

- **Capítulo ?? - 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 relatório final está estruturado;
- **Capítulo ?? - Fundamentação Teórica**: Explicita toda a base teorica com o qual o trabalho foi produzido, tratando das bases para a sua construção;
- **Capítulo ?? - Materiais e Métodos**: Evidencia por qual processo foi executado o objetivo desse artigo até a obtenção dos resultados;
- **Capítulo ?? - Resultados**: Mostra o que foi obtido com a realização desse trabalho;
- **Capítulo ?? - Conclusão**: Apresenta as conclusóes, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.

Capítulo Dois

# Conceito do projeto

Para a realização de cada desafio foi necessário captar conhecimento de diversas fontes para poder construí-lo. Contudo fica evidente a necessidade de trazer esses materiais que inspiraram a produção de tal material.

## *2.1 Desafio Workbooks Python*

O desafio de Workbooks utilizando a linguagem python foi composto de 16 desafios de programação utilizando a linguagem python. Partindo desse contexto, tudo se iniciou com o estudo de bibliotecas e python, para poder realizar as tarefas da forma mais eficiente possível.

### *2.1.1 Median of Three Values*

Consiste em escrever uma função que receba três valores e retorne a mediana entre eles, ou seja, o valor do meio entre todos.

### *2.1.2 The Twelve Days of Christmas*

A tarefa, basicamente, é escrever uma função na qual se digitado o número de um verso da canção The Twelve Days of Christmas, o programa escreva esse verso especifico. Portanto o programa deve possuir uma função que realiza tal feito, e além disso, uma função principal, que escreve os versos de 1 até 12. Para realização foi necessario uma analise logica das relações entre os versos e os paragrafos

### *2.1.3 Center a String in the Terminal*

O código produzido deve ser receber uma variável do tipo *string* qualquer e o número de um terminal, e deve devolver essa *string* centralizada no terminal. Para realizar isso foi

feita uma função e uma parte principal do código que receba os *inputs* e emita na tela a saída desejada.

### *2.1.4 Capitalize It*

Foi necessário realizar uma função que receba um texto e o coloque de acordo com a escrita correta e o devolva mostrando ele no terminal. Assim sendo, deverá colocar a primeira letra como maiúscula e toda letra após ".", "!"ou "?".

### *2.1.5 Does a String Represent an Integer?*

Para realizar a atividade foi necessário realizar um código que verifique que a *string* digitada seja um número inteiro, levando em consideração o sinal positivo ou negativo digitado na frente do restante do texto. Dessa forma, deverá ser feito uma função que diga se o texto digitado é um inteiro ou não e um programa que apresente isso ao usuário.

### *2.1.6 Is a Number Prime?*

A tarefa consiste em identificar se um número é primo e dizer se é ou não. Dessa maneira, deve-se realizar uma função que faça isso e escrever um texto na tela de acordo com o resultado.

### *2.1.7 Random Password*

O código realizado tem como objetivo gerar uma senha de 7 a 11 caracteres. Para isso, deverá escolher aleatoriamente entre os valores 33 e 126 da tabela *ASCII*. Por fim, deve mostrar na tela exclusivamente a senha gerada.

### *2.1.8 Check a Password*

Consiste em verificar se uma senha é segura ou não. Portanto, para a senhas ser considerada segura deve ter no minimo 8 caracteres, no minimo uma letra minuscula, uma letra maíuscula e um numero, Para, dessa maneira retornar verdadeiro ou falso.

### *2.1.9 Arbitrary Base Conversions*

Deve-se escrever um programa que converta de uma base numérica para outra, podendo ser qualquer uma entre 2 e 16. Dessa maneira, o programa tem que ser dividido em diversas funções, dentre elas, uma que converta de uma base arbitrária para uma base 10, e da base 10 para uma base arbitrária e um programa principal que receba a base numerica e exiba a conversão. Para a execução foi estudado bases numericas e as suas formas de conversão.

### *2.1.10 Reduce a Fraction to Lowest Terms*

Consiste em fazer um programa que receba uma fração qualquer e reduzir até ela se tornar uma fração irredutível. Visto isso foi criado uma função que realize essa tarefa. Para isso foi necessário fazer um estudo das relações matemáticas.

### *2.1.11 Reduce Measures*

Tem como objetivo fazer um programa que faça a redução de medidas de copos, colheres de sopa e colheres de chá. Assim, deve pegar unidades menores de medida e distribuí-las na escala maior assim que possível. Para isso foi realizado um estudo matemático para poder aplicar na função com esse propósito.

### *2.1.12 Ceasar Cipher*

Consiste basicamente em fazer uma função que receba um texto e o codifique em ceasar cipher ou o decodifique. Partindo disso, esse tipo de codificação consiste basicamente em deslocar qualquer letra três posições no alfabeto. Assim o programa deve receber um texto a ser transformado e um valor que caso positivo, codifique, e caso negativo, decodifique. Portanto foi necessário fazer um estudo sobre manipulação de *strings* em python.

### *2.1.13 Perfect Numbers*

O programa a ser desenvolvido tem como principal tarefa descobrir se um número inteiro positivo se caracteriza como número perfeito e escrever todos os números perfeitos de 1 até 10000. Assim sendo, para realizar essa tarefa foi necessário fazer um estudo sobre números perfeitos e descobrir qual número é perfeito para assim realizar o código.

### *2.1.14 Recursive Palindrome*

Tem como objetivo criar um código utilizando a recursividade que descobre se a função é um palíndromo ou não. Dessa forma foi feito um estudo acerca de recursividade para poder desenvolver o código.

## *2.2 Desafio C++*

O desafio de C++ foi composto por três desafios de programação que utilizam a linguagem de programação C++. Portanto, para a primeira tarefa foi feito um estudo matemático para a entender como funciona o triângulo de pascal. Em seguida para a segunda tarefa foi feito um estudo combinatório sobre permutações e depois um estudo sobre bibliotecas com a função de realizar essas combinações de palavras de forma mais eficaz. Por fim, para a terceira foi feito um estudo acerca de bibliotecas que trabalhassem com arquivos para assim ler o código que estava escrito. Dessa maneira, foram baseados na construção dos desafios.

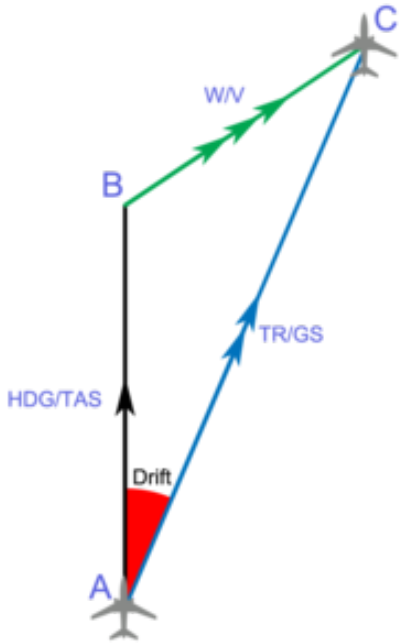
## *2.3 Turtlesim setpoint position*

Consiste em utilizar o software *Turtlesim* para através do ROS escolher uma posição e fazê-lo se deslocar até ela. A movimentação programada na *Turtlesim* se baseia em uma teoria principal, que é o *Dead Reckoning*. Dessa forma, o *Dead Reckoning* consiste em calcular a distância que falta do robô para determinado objetivo e incorporar esses valores a sua posição e velocidade, como pode ser exemplificado na figura **??**. No caso da *Turtlesim* foi calculado sua distância usando o teorema de pitágoras e o ângulo pelo qual a tartaruga precisou virar pela trigonometria.

Na navegação do *Turtlesim*, o teorema de pitagoras é utilizado para calcular a distância entre a mesma e o objetivo definido durante cada instante. Para isso foi utilizado a fórmula $d = \sqrt{(x_\mathrm{f} - x_\mathrm{i})^2 + (y_\mathrm{f} - y_\mathrm{i})^2}$ . Nos quais $\mathrm{x_f}$ e $\mathrm{y_f}$ são as distâncias finais e $\mathrm{x_i}$ e $\mathrm{y_i}$ são as distâncias da tartaruga em determinado instante.

Assim, para calcular quanto que o *Turtlesim* precisa girar faz o uso da trigonometria, para calcular o ângulo do qual a *Turtlesim* está deslocada em relação ao objetivo. Com isso, se faz o uso da fórmula de arctg na qual $\theta = \frac{y_\mathrm{f} - y_\mathrm{i}}{x_\mathrm{f} - x_\mathrm{i}}$ . Por fim, pegamos esse ângulo e subtraímos do ângulo atual da *Turtlesim* para descobrir o quanto a tartaruga precisa rotacionar.

Figura 2.1: Dead reckoning




Fonte: Wikipedia.

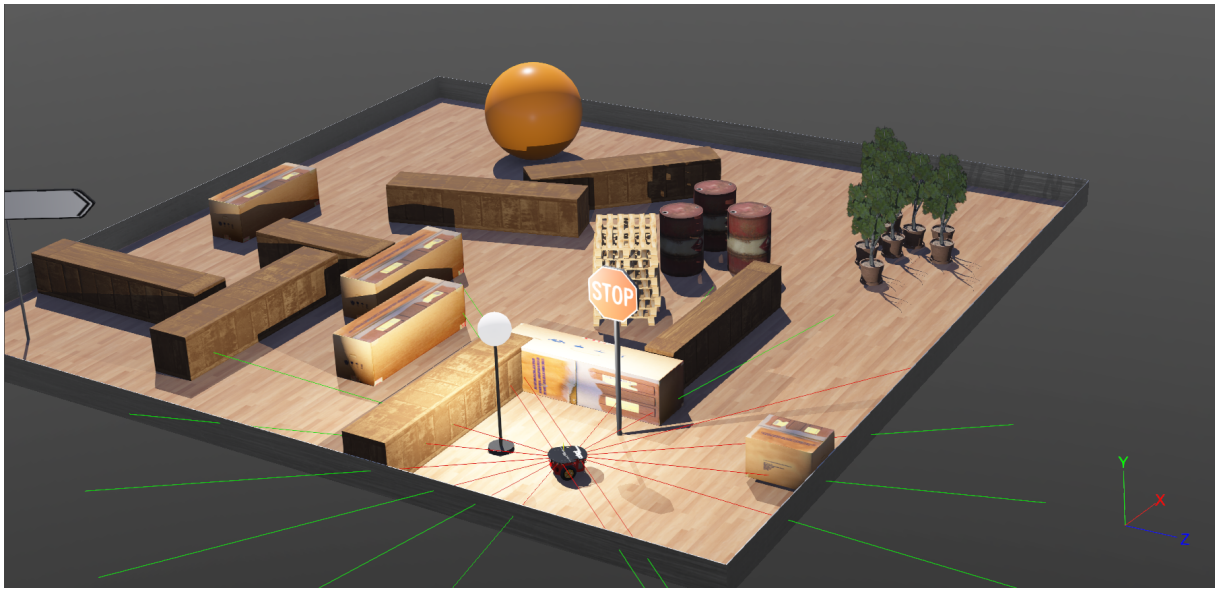
Para definir a velocidade da tartaruga multiplicamos a distância por uma constante e para descobrir o quanto ela precisa rotacionar multiplicamos a angulação por outra constante. Assim esses dados são publicados no topic cmd_vel para alterar a velocidade da turtle até que ela chegue no objetivo especificado.

## *2.4 Desafio Webots*

O Webots é uma plataforma opensource usada para simular robôs. Desse modo, o desafio consiste em utilizar essa plataforma para simular um robô chamado piooner3x, corrigindo o código já existente e alterando ele para caso ele encontre uma luminária pare de se locomover.

A partir disso foram realizados os tutoriais dessa plataforma para ter conhecimento de como utilizá-la e como simular os robôs. Em seguida foi colocado um sensor de luz no piooner3x para que o mesmo tenha uma forma de detectar a luminosidade do ambiente e seu código foi alterado para que se a leitura do sensor ultrapasse determinado valor ele pare.

Figura 2.2: Webots




Fonte: Autoria propria.

## 2.5 Desafio Husky

Husky é um veículo UGV no qual atraves do ROS pode ser simulado em conjunto com o gazebosim e o rviz, sendo ambos simuladores no qual o primeiro é voltado para o ambiente ao redor do husky e o segundo é como esse ugv percebe o mundo. O desafio consiste em utilizar os simuladores para testar diferentes formas de navegação com o husky.

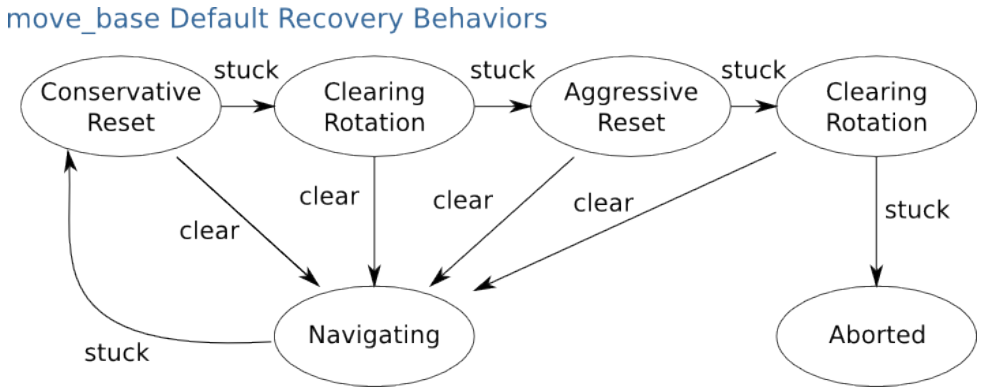
O primeiro desafio é simular o husky com o package move base. Consiste em dar uma localização no mundo e ele irá tentar atingir esse objetivo. Caso o ugv identifique algum obstáculo ele irá desviar, ou caso fique preso irá entrar em um processo chamado conservative reset se parar de ficar preso voltará a navegação, caso não entrará em clearing rotation, se mesmo assim continuar preso iniciaram um aggressive reset e continuando preso vai por fim fazer uma clearing rotation e se mesmo assim continuar preso vai abortar a ação. Como mostra na imagem **??**.

O segundo desafio é o amcl demo, o qual é a junção do move base com o amcl. Assim o amcl é um sistema probabilístico de localização do robô o qual através de sensores de laser fazem o tracking da posição do robô dentro de um mapa.

Gmapping demo é o terceiro desafio, ele é a junção do move base com o gmapping. Esse package prove um SLAM (Simultaneous localization and mapping), baseado em sensores a laser. Com o gmapping é criado um mapa 2D do ambiente.

Por último foi realizado o frontier exploration demo, é composto pelo move base, gmapping,

Figura 2.3: Move Base




Fonte: ROS Wiki.

e o frontier exploration. Dessa forma, o frontier em conjunto com esses packages para realizar a exploração de ambientes

Capítulo Três

# Desenvolvimento do projeto

Durante esta seção sera descrito o processo de construção dos desafios, incluindo os estudos necessários para cada um, o processo de criação e especificidades. Será apresentado estas características para cada um desafio.

## *3.1 Desafio Workbooks Python*

Para realização deste desafio foi realizada a pesquisa sobre bibliotecas em python para cada tarefa própria e em seguida criação de scripts e testes. Para dessa forma concluir o desafio.

## *3.2 Desafio C++*

Tem como início o estudo da linguagem C++ através da Code Academy com o seu curso. Depois, para cada tarefa foi realizada uma pesquisa própria, sendo sobre matemática teoria ou bibliotecas dessa linguagem. Assim, foi realizado o desafio.

## *3.3 Turtlesim setpoint position*

O desafio iniciou- se com um estudo sobre o ROS, mais especificamente sobre publisher e subscriber. Após isso, foi feita uma pesquisa acerca das técnicas de navegação chegando no dead reckoninig. Em seguida, iniciou-se a fase de estudo da programação e testes do package até chegar no produto final.

## *3.4 Desafio Webots*

Na primeira etapa foram realizados os tutoriais do Webots, logo em diante foi feito um estudo da programação nesta plataforma. Depois disso foram realizados seguidos testes e alterações até que o robô conseguisse se locomover corretamente para após isso implementar o sensor de luz.

## *3.5 Desafio Husky*

O desafio foi realizado com o uso do ROS Noetic. Para ser realizado foi feito *git clone* de um repositório no github do Husky em um workspace. Com isso, foram realizados os desafios porém alguns erros surgiram, mas foram solucionados através de pesquisa e ajuda de colegas.

Capítulo Quatro

# Resultados

Depois de finalizados os desafios foram coletados os seus resultados e disponibilizados no github.

## *4.1 Desafio Workbooks Python*

Com a conclusão do desafio, os codigos foram testados e tiveram sucesso no seu funcionamento. Assim,

## *4.2 Desafio C++*

Os códigos foram compilados e executados obtendo sucesso no seu funcionamento. O primeiro

## *4.3 Turtlesim setpoint position*

Depois de finalizado o package tinha seu funcionamento como exigia o regulamento. Portanto as coordenadas eram digitadas e a tartaruga realizava um deslocamento ate alcançar uma distância de pelo menos 0.1 do objetivo e assim encerrava seu programa.

## *4.4 Desafio Webots*

Após o explicitado na metodologia o robô piooner3x conseguiu seguir o percurso sem grandes problemas. Desse modo e parar quando encontrava uma fonte luminosa, sendo no caso uma luminária em um canto do mapa dentro de XX segundos

## *4.5 Desafio Husky*

As diferentes navegações foram corretamente executadas e colocadas em prática dentro dos simuladores gazebosim e rviz. Com isso foi capaz perceber como as diferentes técnicas de navegação aplicadas e combinadas afetam o deslocamento do ugv

Capítulo Cinco

# Conclusão

Durante todos os desafios foram exigidos o aprendizado e domínio de novas ferramentas. Os desafios foram compostos de uma série de atividades envolvendo programação, simulação, navegação e domínio do *ROS*.

## *5.1 Considerações finais*

O trabalho fomentou um aprendizado específico de maneira prática e objetiva, voltado para a criação de habilidades, o desenvolvimento do pensamento lógico para resolução de problemas e absorção das ferramentas utilizadas. Dessa forma foi de grande importância a realização dessas atividades pois deu uma base sólida de conhecimentos que vão ser futuramente utilizados.

Apêndice A

# Diagramas mecânicos

Apêndice B

# Diagramas eletro-eletrônicos

*Desafios para o laboratorio robotica e sistemas autonomos*

Tiago Barretto Sant'Anna

Salvador, Dezembro de 2021.