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 estabelecidos. Assim este relatório exibe 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 diversos para poder realizar estes desafios.

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

Abstract

The work consists in the explanation of the perceptions about the established challenges. So this report shows 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, several studies were carried out to be able to carry out these challenges.

Keywords: robotics, study, programming, simulation.

Sumário

Lista de Figuras

Capítulo Um

Introdução

A robótica é a area que cresce de maneira gigantesca no mercado atual (**??**). Devido a esse fator urge capacitar mão de obra no mercado. Assim foi criado o Centro de Competências de Robótica e sistemas autônomos, para permitir uma capacitação de profissionais nessa area.

Dessa forma, o Centro de Competências elabora atividades para poder treinar essa futura mão de obra. Assim surgem os desafios executados dentro deste relatório.

O seguinte relatório foi realizado para o Centro de Competências de Robótica e Sistemas Autônomos dentro do Centro Universitário Senai CIMATEC. Este tem por objetivo expor os desafios de capacitação realizados para poder atuar em atividades dentro do laboratório. Sendo realizados em um período de 2 meses

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:

- Aprender 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 necessárias para trabalhar com robótica, sendo aprendendo a utilizar o framework ROS com a versão noetic, ou habilitando seu conhecimento em programação.

A programação é a base para o desenvolvimento de robôs. Desse modo urge utilizar linguagens que facilitam essa produção. Segundo (**??**) o python tem uma função muito importante na robótica, que se deve graças a facilidade de escrita da sua linguagem. Ja o C++ tem um fator importante ligado a performance, ja que é uma linguagem de mais baixo nível.

Para poder desenvolver robôs é muito importante fazer sua simulação, testando sua programação e funcionamento. Assim, o Webots cumpre essa função sendo um simulador gratuito e opensource. Neste simulador, pode testar como o protótipo se comporta fisicamente no ambiente e se a programação está cumprindo o seu papel. Evitando que se construa um protótipo físico precocemente.

O uso de frameworks de robótica é indispensável para quem trabalha nessa area, visto que eles facilitam o trabalho de programação e simulação dos projetos desenvolvidos. Assim o foi usado o ROS noetic, sendo a versão mais recente do ROS 1, devido a sua popularidade. Como visto em (**??**).

1.3 Organização do documento

Este documento apresenta 3 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 ?? - Desenvolvimento**: Explicita com toda a base teorica como o trabalho foi desenvolvido, o processo pelo qual ele passou e o que foi obtido.
- **Capítulo ?? - Conclusão**: Apresenta as conclusões, contribuições e algumas sugestões de atividades de pesquisa a serem desenvolvidas no futuro.

Capítulo Dois

Desenvolvimento

Nesta seção todo o trabalho desenvolvido é exposto, através da fundamentação teórica, os metódos utilizados e os resusltados obtidos dentro de cada desafio.

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.

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.

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

2.1.1 Median of Three Values

De acordo com **??**) consiste em escrever uma função que receba três valores e retorne a mediana entre eles, ou seja, o valor do meio entre todos.

```
def median(a, b, c):
    if(c >= a >= b or b >= a >= c):
        return a
    elif(c >= b >= a or a >= b >= c):
        return b
    else:
        return c

print('Digite 3 valores para tirar sua mediana')
x = int(input('x: '))
y = int(input('y: '))
```

```
    z = int(input('z: '))

    print(f'Esses valores tem como mediana {median(x,y,z)}')
```

Dentro da função que faz esse calculo é realizada um diversas comparações entre os três valores para achar aquele que compõe a mediana.

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, um código principal, que escreve os versos de 1 até 12. Para realização foi necessário uma analise logica das relações entre os versos e os parágrafos.

Primeiramente foi necessário a analise das letras da musica completa, encontrada em (**??**). Com esta analise foi possível perceber que padrões se repetiam na musica, como certas estrofes e também que algumas surgiam de acordo com o paragrafo da musica. Após esta etapa buscou-se as relações matemáticas entre os padrões encontrados. Assim desenvolvendo a função.

```
    def music(verse):
        if(verse > 102): return ''
        pos = 3
        count = 0
        n = verse
        days = ['first', 'second', 'third', 'fourth', 'fifth', '
    sixth', 'seventh', 'eighth', 'ninth', 'tenth', 'eleventh','
    twelfth']
        vs3 = ['A partridge in a pear tree.','And partridge in a
     pear tree.']
        vs2 = ['Twelve fiddlers fiddling,','Eleven ladies
    dancing,','Ten pipers piping,','Nine drummers drumming,','
    Eight maids a-milking,','Seven swans a-swimming,','Six geese
    a-laying,','Five gold rings,','Four colly birds,','Three
    French hens,','Two turtle doves']

        vs1 = ['Two turtle doves','Three French hens,','Four
    colly birds,','Five gold rings,','Six geese a-laying,','Seven
     swans a-swimming,','Eight maids a-milking,','Nine drummers
```

```
drumming,','Ten pipers piping,','Eleven ladies dancing,','
Twelve fiddlers fiddling,']

        while(n >= pos):
                n = n - pos
                pos += 1
        estrofe = pos - 3

        if(n == 1):
            return 'The '+ days[estrofe] + ' day of Christmas,'
        elif(n == 2):
            return 'My true love sent to me'
        elif(n == 0 and estrofe == 1):
            return vs3[0]
        elif(n == 0 and estrofe != 1):
            return vs3[1]
        else:
            x = vs2[11-estrofe + n-3]
            return x


    for i in range(1, 13, 1):
        print(music(i))
```

A função desenvolvida recebe o numero do verso desejado, que pode ser até 102, e retorna esse verso. Dentro dela é realizado um calculo para descobrir ou formar a frase que vai ser enviada. Desse modo, o código principal, é feito de uma forma que escreva na tela os doze primeiros parágrafos da canção.

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.

```
    from typing import Text

    def center_string (text, spaces):
        c =''
```

```
        a = len(text)
        b = (spaces/2) - (a/2)
        for i in range(0, int(b), 1):
            c += ' '
        c += text
        return c


t = str(input('texto: '))
s = int(input('espa os: '))

print(center_string(t,s))
```

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 "?".

```
def capitalize(text):
    text = text[0].upper() + text[1:]
    tam = len(text)
    for i in range(1, tam-2, 1):
        if(text[i] == 'i' and text[i-1] == ' ' and text[i+1]
== ' '):
            text = text[:i] + 'I' + text[i+1:]
        elif(text[i] == '.' or text[i] == '!' or text[i] ==
'?'):
            if(text[i+1] == ' '):
                text = text[:i+2] + text[i+2].upper() + text
[i+3:]
            else:
                text = text[:i] + text[i+1].upper() + text[i
+2:]
    return text

t = str(input('Escreva o texto a ser capitalizado: '))
print(capitalize(t))
```

Dentro da função que realiza essas tarefas ela percorre a string ate identificar as situações especificas e substituir as letras por letras maiúsculas. Assim se parte essa string de depois as junta novamente para poder fazer essa troca.

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.

```
    def isInteger(txt):
        txt = txt.strip()
        if(len(txt) < 1 ): return False
        elif(txt[0] == "+" and txt[1:].isnumeric()): return True
        elif(txt[0] == "-" and txt[1:].isnumeric()): return True
        elif(txt.isnumeric()): return True
        else: return False

    s = str(input('Write a string to discover if is a integer: '
    ))

    if(isInteger(s) == 1): print('Is integer')
    else: print('Is not integer')
```

A função percorre toda a string em busca de caracteres númericos, caso todos sejam númericos retorna verdadeiro e caso não retorna falso. Assim o código principal recebe a string e exibe o resultado.

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.

```
    def isPrime(n):
        count=0
        if(n == 1): return True
        for i in range(2, n, 1):
```

```
            if(n % i == 0): count += 1
        if(count > 1): return False
        else: return True

    num = int(input('Digite o numero que voce deseja descobrir
se     primo: '))

    if(isPrime(num)): print('    primo')
    else: print('N o     primo')
```

De acordo com **??**) "Os números primos são aqueles que apresentam apenas dois divisores: um e o próprio número". Então nesta função se conta todos os divisores de um número escolhido, para descobrir se é primo ou não.

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.

```
    import random
    def passGen():
        txt = ''
        lengh = random.randrange(7,10,1)
        for i in range(0, lengh, 1):
            txt += chr(random.randrange(33,126,1))
        return txt

    print(passGen())
```

Para que isso seja possível foi usado a biblioteca random para gerar números aleatórios dentro dos valores limitados da tabela *ASCII*. Tendo esses valores, os convertendo novamente para char.

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 mínimo 8 caracteres, no mínimo uma letra minuscula, uma letra

maiúscula e um numero. Para, dessa maneira retornar verdadeiro ou falso.

```
def passCheck(password):
    ucase = 0
    lcase = 0
    num = 0
    if(len(password) < 8): return False
    for i in range(0,len(password)-1,1):
        if(password[i].isupper()): ucase += 1
        elif(password[i].islower()): lcase += 1
        elif(password[i].isnumeric()): num += 1
    if(ucase >= 1 and lcase >= 1 and num >= 1): return True
    else: return False

p = str(input('Digite a senha a ser validada: '))

if(passCheck(p) == True):
    print('A senha esta boa')
else:
    print('A senha n o esta boa')
```

Portanto a função percorre o string da função verificando se ela possui as características adequadas para ser uma senha segura. Assim, a função desenvolvida retorna um valor de verdadeiro caso a senha seja segura e falso caso a senha não seja segura.

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 numérica e exiba a conversão. Para a execução foi estudado bases numéricas e as suas formas de conversão.

```
def toBase10(num, orig_base):
    base = 0
    if(num == 10): return orig_base
    elif(num >= 2 and num <= 16):
        j = len(orig_base) - 1
        for i in range(0, len(orig_base), 1):
            if(orig_base[i] == 'A' or orig_base[i] == 'a'):
                base += 10 * (num ** j)
```

```
                elif(orig_base[i] == 'B' or orig_base[i] == 'b')
:
                    base += 11 * (num ** j)
                elif(orig_base[i] == 'C' or orig_base[i] == 'c')
:
                    base += 12 * (num ** j)
                elif(orig_base[i] == 'D' or orig_base[i] == 'd')
:
                    base += 13 * (num ** j)
                elif(orig_base[i] == 'E' or orig_base[i] == 'e')
:
                    base += 14 * (num ** j)
                elif(orig_base[i] == 'F' or orig_base[i] == 'f')
:
                    base += 15 * (num ** j)
                else:
                    base += int(orig_base[i]) * (num ** j)
                j -= 1
        else:
            return 'ERROR'
        return str(base)

    def decToBaseX(num, orig_base):
        txt = ''
        txt2 = ''
        if(num == 10): return orig_base
        elif(num >= 2 and num <= 16):
            obase = int(orig_base)
            while(obase >= num):
                rest = obase % num
                obase = obase // num
                if(rest == 10):
                    txt += 'A'
                elif(rest == 11):
                    txt += 'B'
                elif(rest == 12):
                    txt += 'C'
                elif(rest == 13):
                    txt += 'D'
                elif(rest == 14):
                    txt += 'E'
```

```
                elif(rest == 15):
                    txt += 'F'
                else:
                    txt += str(rest)
            txt += str(obase)
            for j in range(len(txt)-1, -1, -1):
                txt2 += txt[j]
        return txt2

    def yToBaseX(o_base, f_base, txt):
        return decToBaseX(f_base, toBase10(o_base, txt))

    x = str(input('Digite a base:'))
    y = int(input('Digite qual o valor da base dele: '))
    z = int(input('Digite para qual base deseja converter: '))

    print(yToBaseX(y,z,x))
```

O programa consiste em três funções, uma que converte de qualquer base para a base decimal, outra em que converte a base decimal para qualquer base, e mais uma que faz a junção das duas primeiras funções. Assim para aprender como se converte bases decimais foi consultado (**??**). Assim o código permite que qualquer base seja convertida para qualquer base.

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 (**??**).

```
    def reductFrac(numerator, denominator):
        if(denominator >= numerator): x = denominator
        else: x = numerator

        for i in range(x,1,-1):
            if(numerator % i == 0 and denominator % i == 0):
                numerator = numerator/i
                denominator = denominator/i

        return [int(numerator), int(denominator)]
```

```
n = int(input('Put the numerator: '))
d = int(input('Put the denominator: '))

[x,y] = reductFrac(n,d)

print(f'{x}/{y}')
```

A função retorna um vetor com o valor do numerador e denominador que é exibido na tela pelo código.

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.

```
    def converter(number, unit):
        if(unit != 'cup' and unit != 'tablespoon' and unit != '
    teaspoon'): return 'Invalid unit'

        if(unit == 'tablespoon'):
            cv = number * 3
        if(unit == 'teaspoon'):
            cv = number
        elif(unit == 'cup'):
            return number + 'cup'

        cup = int(cv/48)
        cv = cv % 48
        tbsp = int(cv/3)
        cv = cv % 3
        teasp = cv

        if(cup > 1):
            c = ' cups, '
        else:
            c = ' cup, '
        if(tbsp > 1):
```

```
            t = ' tablespoons, '
        else:
            t = ' tablespoon, '
        if(teasp > 1):
            te = ' teaspoons'
        else:
            te = ' teaspoon'

        return str(cup) + c + str(tbsp) + t + str(teasp) + te

    n = int(input('Put the unit quantity: '))
    d = str(input('Put the unit name (cup/tablespoon/teaspoon):
'))

    print(converter(n,d))
```

A função primeiro identifica a unidade usada e a converte para a menor possível, o "teaspoon". Para depois fazer a distribuição de medidas. Assim, retornando o resultado.

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.

```
    def caeser_cipher(txt):
        txt2 = ''
        alphabet = ['a','b','c','d','e','f','g','h','i','j','k',
'l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','
a','b','c']
        alphaM = ['A','B','C','D','E','F','G','H','I','J','K','L
','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','A'
,'B','C']
        for i in range(0, len(txt), 1):
            for j in range(0, len(alphabet)-3,1):
                if(txt[i] == alphabet[j]):
                    txt2 += alphabet[j+3]
                elif(txt[i] == alphaM[j]):
```

```
                txt2 += alphaM[j+3]
        return txt2

    def deco_caeser_cipher(txt):
        txt2 = ''
        alphabet = ['x','y','z','a','b','c','d','e','f','g','h',
'i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','
x','y','z']
        alphaM = ['X','Y','Z','A','B','C','D','E','F','G','H','I
','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X'
,'Y','Z']
        for i in range(0, len(txt), 1):
            for j in range(0, len(alphabet)-3,1):
                if(txt[i] == alphabet[j]):
                    txt2 += alphabet[j-3]
                elif(txt[i] == alphaM[j]):
                    txt2 += alphaM[j-3]
        return txt2

    x = int(input('Se voc  deseja encriptar a mensagem digite
1, se deseja descriptar digite 0: '))
    t = str(input('Digite o texto: '))
    if(x == 1):
        print(caeser_cipher(t))
    elif(x == 0):
        print(deco_caeser_cipher(t))
    else:
        print('Digite 1 ou 0')
```

Dessa maneira foram feitas duas funções, sendo a $ceaser_cyphera funçãoquecodificaamensageme$

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.

```
    def perfect_num(num):
        cont = 0
        for i in range(1, num, 1):
            if(num % i == 0):
                cont += i
```

```
        if(cont == num):
            return True
        else:
            return False

    print('Perfect numbers:')
    for i in range(1, 10000, 1):
        if(perfect_num(i)):
            print(i)
```

Assim a função faz os cálculos matemáticos necessários para identificar e retornar um valor booleano.

2.1.13 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.*

```
    def palindrome(txt):
        txt = txt.lower()
        txt2 = ''
        for i in range(len(txt)-1, -1, -1):
            txt2 += txt[i]
        if(txt == txt2): return True
        else: return False

    p = str(input('Put a message to discorver if is a
palindrome or not: '))
    if(palindrome(p)):
        print('The message is a palindrome')
    else:
        print('The message its not a palindrome')
```

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.

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.

2.2.1 Desafio do Triângulo de Pascal

Para iniciar com o desenvolvimento do código foi primeiramente necessário estudar o triangulo de Pascal (??) para entende-lo. Após isso foram realizados vários testes até encontrar uma forma de desenvolver esse desafio.

```
#include<iostream>
#include<math.h>

using namespace std;

int fatorial(int n){
    int a = 1;
    if(n > 0){
        for(int i = n; i > 0; i--){
            a *= i;
        }
    }
    else{
        return 1;
    }
    return a;

}

int triangulo_de_pascal(int linha, int posicao_linha){
    int pascal;
    // verificar a validade dos numeros digitados
    if(linha == 0 && posicao_linha != 0) return 0;
    else if(linha > 0 && posicao_linha > linha+1) return 0;
    else if(linha < 0) return 0;
```

```
        // calcular o valor da determinada posicao
        if(linha == 0) return 1;
        pascal = fatorial(linha) / (fatorial(posicao_linha) *
fatorial(linha-posicao_linha));
        return pascal;
 }

 int main(){

        int a, b;

        cout << "Calculadora de triangulo de pascal\n\n";
        cout << "*considere a primeira linha como 0 e a
primeira posicao com 0 tambem*\n\n";

        cout << "Insira a linha: ";
        cin >> a;
        cout << "Insira a posicao do numero: ";
        cin >> b;

        cout << triangulo_de_pascal(a, b) << endl;

        return 0;
 }
```

O código consiste da função do triângulo de pascal, uma função para facilitar o calculo fatorial, e o main para poder executar as funções.

2.2.2 Desafio da Permutação de Cordas

Para poder realizar essa tarefa foi estudado algoritmos de Heap (–Wik44:online++IMPL–Wik44:online+). Porem para facilitar a implementação foi decidido usar a biblioteca algorithm ja que possui uma função especifica para isso.

```
#include<iostream>
#include<string>
#include<algorithm>

using namespace std;
```

```
int main(){
    string txt;
    cout << "Digite a palavra a ser permutada: ";
    cin >> txt;

    sort(txt.begin(), txt.end());
    do{
        cout << txt << endl;
    }while(next_permutation(txt.begin(), txt.end()));

    return 0;
}
```

Assim o código recebe a palavra a ser permutada e vai executando a função de next$_p$*ermutationatéquetod*

2.2.3 Programa de Autoimpressão

*Para que esse desafio seja realizado foi necessário usar uma biblioteca que permitisse manipular documentos, para isso foi usado a fstream. Assim essa biblioteca foi estudada (**??**) e seus comandos foram dominados. Para assim por em pratica o desafio.*

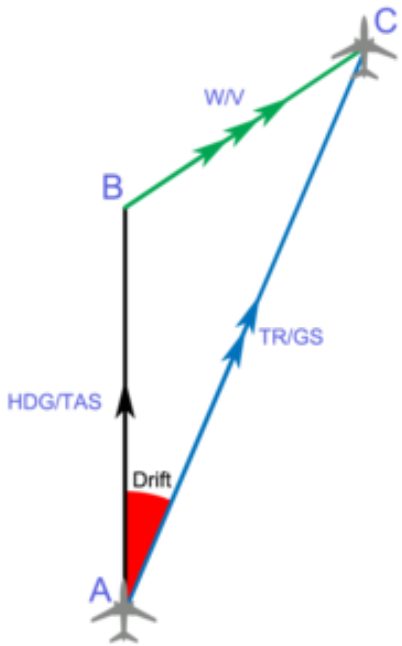
```
#include<fstream>
#include<iostream>
using namespace std;

int main(){
    std::fstream fs;
    fs.open ("autoimpress.cpp", std::fstream::in | std::
fstream::out | std::fstream::app);

    return 0;
}
```

O código é composto de um main que abre o próprio arquivo e o exibe na tela.

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.*

Figura 2.1: Dead reckoning



Fonte: Wikipedia.

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 $d = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$. *Nos quais* x_f *e* y_f *são as distâncias finais e* x_i *e* 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 $\theta = \frac{y_f - y_i}{x_f - x_i}$. *Por fim, pegamos esse ângulo e subtraímos do ângulo atual da Turtlesim para descobrir o quanto a tartaruga precisa rotacionar.*

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.

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.

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.

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.

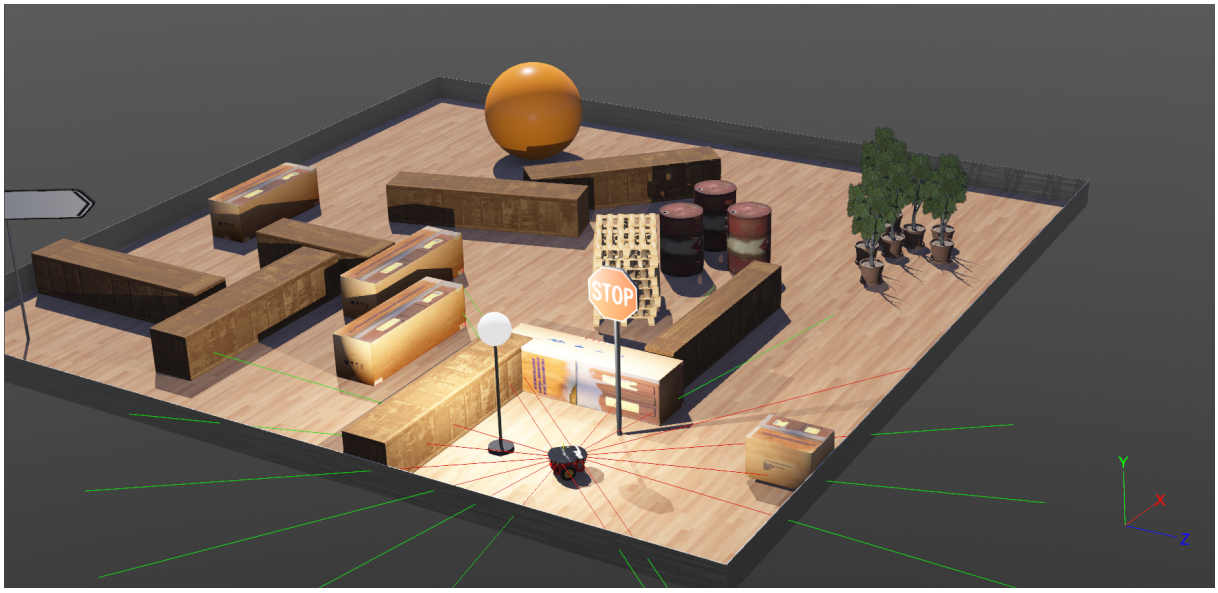
Na primeira etapa foram realizados os tutoriais do Webots, logo em diante foi feito um estudo da programação nesta plataforma. Depois disso foram realizados seguidos testes e alterações até que o robô conseguisse se locomover corretamente para após isso implementar o sensor de luz.

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

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

Figura 2.2: Webots

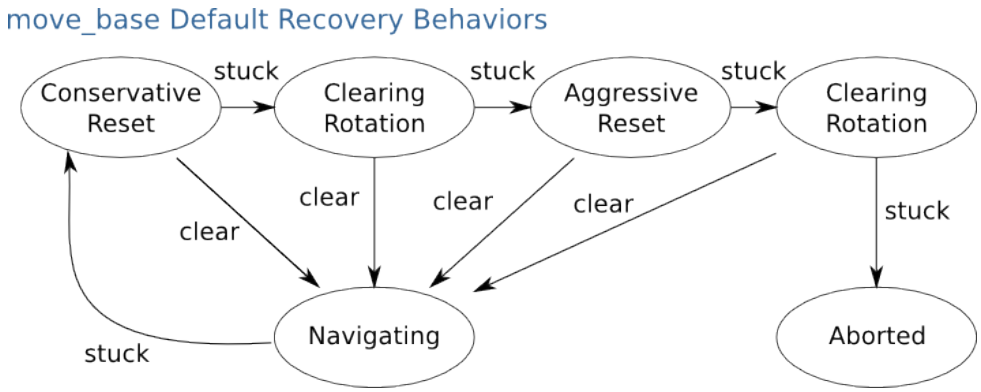


Fonte: Autoria propria.

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 **??**.

Figura 2.3: Move Base



Fonte: ROS Wiki.

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

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.

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 Três

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.

3.1 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.

Desafios para o laboratorio robotica e sistemas autonomos

Tiago Barretto Sant'Anna

Salvador, Dezembro de 2021.