UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO
CURSO DE ENGENHARIA MECATRÔNICA E ENGENHARIA DA COMPUTAÇÃO

RELATÓRIO

1ª Projeto de Sistemas Robóticos Autônomos - Meta 01

ATYSON JAIME DE SOUSA MARTINS: Nº 20190153956
JOSE LINDENBERG DE ANDRADE: Nº 20200150293
JÚLIA COSTA CORRÊA DE OLIVEIRA: Nº 20200149087
SAMUEL CAVALCANTI: Nº 20200149318

Natal-RN

2021

ATYSON JAIME DE SOUSA MARTINS: Nº 20190153956
JOSE LINDENBERG DE ANDRADE: Nº 20200150293
JÚLIA COSTA CORRÊA DE OLIVEIRA: Nº 20200149087
SAMUEL CAVALCANTI: Nº 20200149318

1ª PROJETO DE SISTEMAS ROBÓTICOS AUTÔNOMO - META 01

Relatório presentado à disciplina de Sistemas Robóticos Autônomos, correspondente à avaliação parcial da 1º unidade do semestre 2021.2 do curso de Engenharia Mecatrônica e Engenharia da Computação da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Pablo Javier.**

Professor: Pablo Javier Alsina.

Natal-RN
2021

RESUMO

Esse relatório tem como objetivo mostrar o procedimento da obtenção da primeira meta para a simulação de um robô móvel com acionamento diferencial e o desenvolvimento de um sistema de controle cinemático que permita ao mesmo executar movimentos especificados em espaço livre de obstáculos.

Palavras-chave: Robô móvel; Coppeliasim; Pioneer 3-DX.

Lista de Figuras

1 Robô Pioneer 3-DX com 16 sensores utilizado na simulação do V-rep. 7
2 Script lua controlador do Pionner 8
3 Habilitando o remoteAPI 8
4 comunicação entre a aplicação em python com o simulador 9
5 função principal da nossa aplicação python 9
6 Recebimento de posições e ângulos. 10
7 Gráfico da orientação do robô. 10
8 Gráfico da velocidade do robô. 11
9 Gráfico da posição do robô. 11

Sumário

1 INTRODUÇÃO **7**

2 RELATO **7**

3 RESULTADOS **10**

4 CONCLUSÃO **12**

1 INTRODUÇÃO

Este trabalho é um relato de uma simulação de um robô móvel com acionamento diferencial, onde futuramente irá ser feito um sistema de controle cinemático que permita ao mesmo executar movimentos especificados em espaço livre de obstáculos. Para simular o robô foi utilizado o simulador Coppeliasim (Versão: 4.2), conhecido anteriormente como V-Rep, o robô de acionamento diferencial escolhido foi o Pioneer 3-DX.

2 RELATO

No simulador Coppeliasim, o robô Pioneer 3-DX possui 16 sensores de proximidade, no qual cada sensor é capaz de detectar objetos a uma distância de 1 metro. Ele possui também um script em Lua (linguagem de programação) onde nele está escrito o seu controlador. No Coppeliasim existe um tipo de junta chamada *Revolute joints* e é com essa junta que as rodas do Robô Pioneer 3-DX simulado se movem na cena. Na Figura abaixo é possível ver o robô simulado.

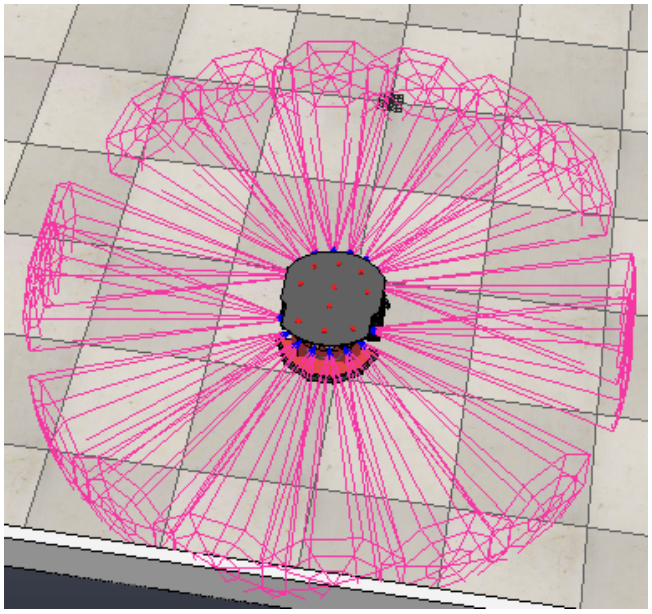

Figura 1: Robô Pioneer 3-DX com 16 sensores utilizado na simulação do V-rep.

Para o robô movimentar-se na cena é preciso que exista alguns comandos, podendo ser tanto aplicados diretamente no simulador, na linguagem Lua, quanto por uma aplicação externa. O código deve recuperar as informações das juntas, que representam os atuadores do robô, e atribuir a sua velocidade; isso foi observado ao verificar que o script do controlador já vem com o modelo Pioneer presente no simulador, o que pode ser visto na Figura 2 .

Figura 2: Script lua controlador do Pionner

Como o presente grupo ficou encarregado também de apresentar um seminário posteriormente sobre a criação de uma aplicação externa que se comunica com o simulador, então resolveu-se passar o código escrito em Lua para uma aplicação externa, escrita em Python. Para isso, foi seguido a própria documentação do Coppeliasim, que encontra-se no link: `https://www.coppeliarobotics.com/helpFiles/en/remoteApiClientSide.htm`, onde basicamente foi copiado os arquivos sugeridos pela documentação e inserido um novo script Lua, do tipo *non threaded*, no simulador que inicializa uma conexão via socket na *porta 19999* quando a simulação é inicializada e encerra a conexão quando a simulação é parada. O código que inicializa o socket pode ser visto na figura 3.

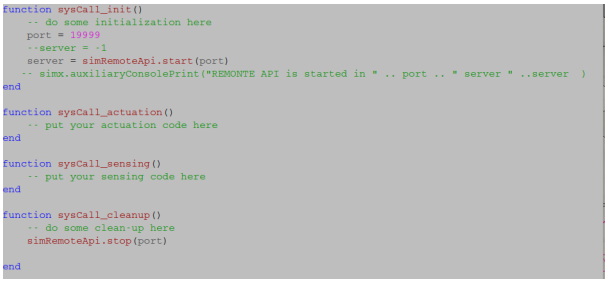

Figura 3: Habilitando o remoteAPI

Após isso, foi consultado a referência, no seguinte link: `https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm` como utilizar as mesmas funções que são executadas em Lua em Python, e, com isso, foi reimplementado o mesmo controlador escrito em Lua para Python. A Figura 4 mostra algumas das funções usadas pela aplicação implementada para extrair informações do simulador e realizar uma comunicação com ele.

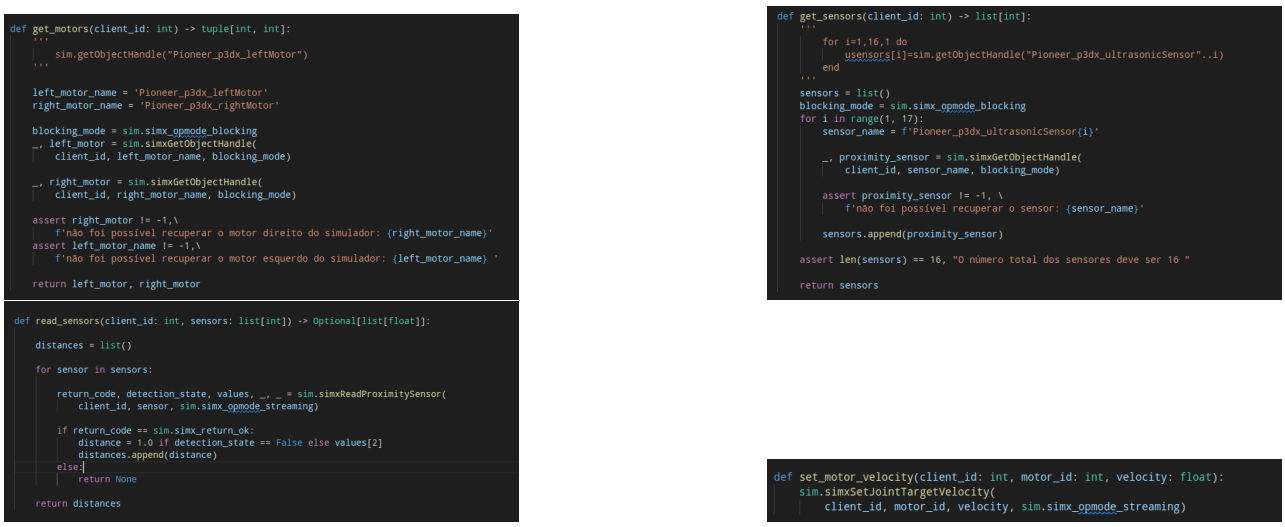

Figura 4: comunicação entre a aplicação em python com o simulador

Pode-se resumir a aplicação, implementada pelo grupo em Python, em uma simples rotina, no qual, enquanto a simulação estiver ativa é extraído dados do simulador, como valores de sensores ou posição e orientação do robô, e é enviado para uma função que representa o controlador, que retorna os valores das velocidades das rodas do robô diferencial, como pode-se observar na figura 5. O código completo pode ser encontrado no seguinte repositório do *github*: `https://github.com/samuel-cavalcanti/controle_cinematico_sistemas_autonomos_ufrn` no arquivo *simple_test.py*.

Figura 5: função principal da nossa aplicação python

3 RESULTADOS

A primeira parte do projeto, que tem como objetivo simular o robô e fazer com que ele ande com uma determinada velocidade, foi realizada com êxito. Além disso, foi possível receber as posições do robô e sua orientação, como pode ser visto na Figura abaixo.

```
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
Orientação do Robo(Theta):  94.78159462967304
position:  [-4.109951972961426, -0.652229368686676, 8.068520545959473]
euler angles:  [3.1247057914733887, 0.5268534421920776, -1.5372228622436523]
```

Figura 6: Recebimento de posições e ângulos.

Com isso, foi possível gerar os seguintes gráficos: velocidades das rodas (entradas) em função do tempo; configuração do robô (x,y,θ), (saídas), em função do tempo; gráfico das posições (x(t),y(t)) seguidas pelo robô no plano xy. Esses podem ser vistos nas imagens abaixo.

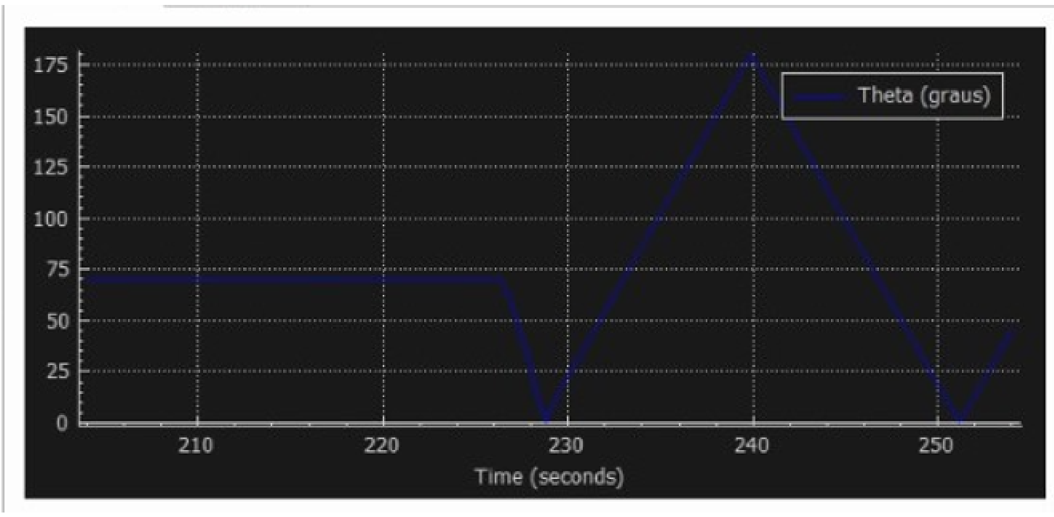

Figura 7: Gráfico da orientação do robô.

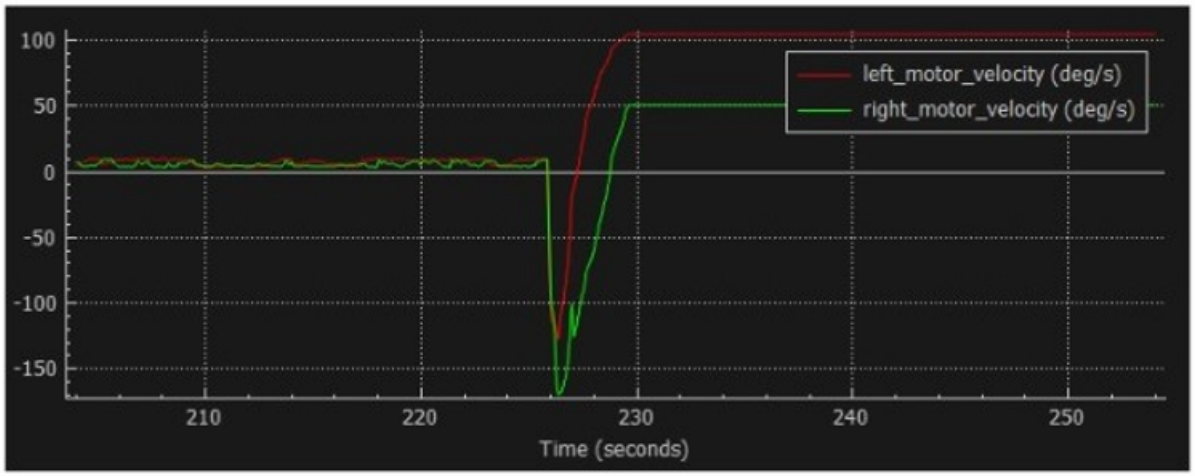

Figura 8: Gráfico da velocidade do robô.

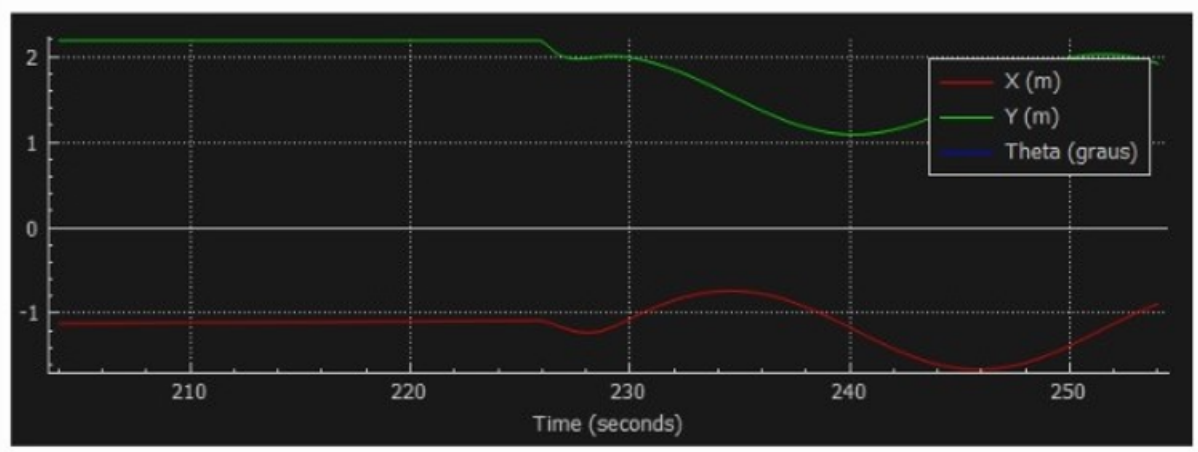

Figura 9: Gráfico da posição do robô.

4 CONCLUSÃO

A partir do desenvolvimento do trabalho proposto atingimos com exito a realização da meta 01 para o projeto 01. Como também, conseguimos aprender as funções e algorítimos que o V-Rep nos proporciona para o desenvolvimento das cinemáticas dos robôs, dessa forma, sendo possível elencar e revisar os assuntos abordados em sala de aula. Além disso, foi possível aprender como conectar o simulador com sistemas externos, sendo possível utilizar das funções do simulador em outras linguagens.

Referências

[1] API do V-RAP. Disponível em: <https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm#simxGetObjectPosition>