Aluno: Rodrigo Lopes Catto
Orientador: Fábio de Miranda

Mapeamento, localização e navegação em áreas externas com robô autônomo

São Paulo
2019

Índice

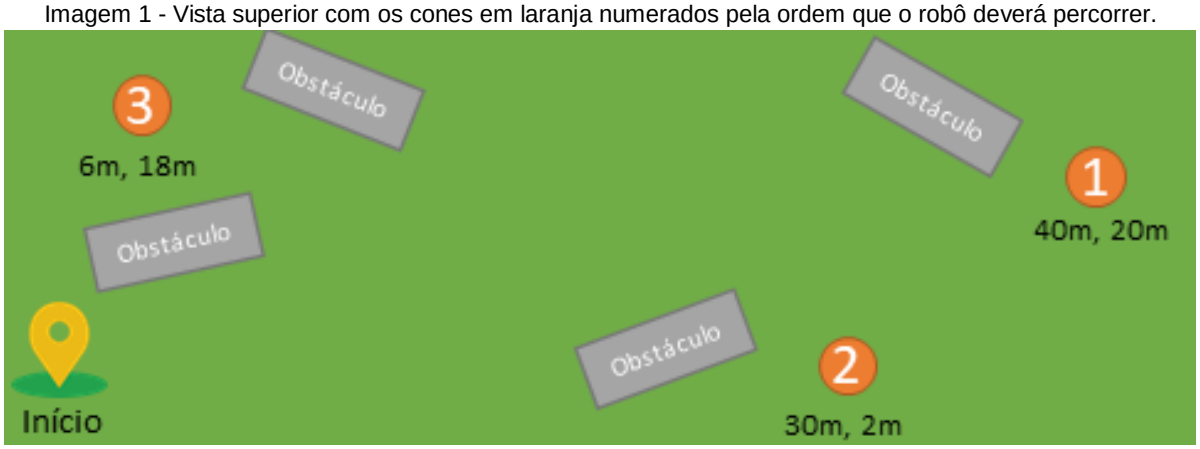
Introdução

A navegação autônoma apenas com dados providos por sensores locais e coletados à medida em que um robô se desloca em um ambiente ainda não mapeado é um dos maiores desafios da robótica móvel autônoma. Podemos notar o uso de robôs móveis cada vez mais perto do nosso dia a dia, o que demonstra que esta área é promissora num futuro próximo. Um exemplo disso, é o uso de robôs para aplicações domésticas como aspiradores de pó e cortadores de grama, aplicações de segurança patrimonial e também missões de resgate em regiões de acidentes pouco acessíveis, aplicações militares na forma de aviões autônomos de monitoramento aéreo, aplicações industriais como robôs de logística e, por fim, aplicações urbanas como transporte de pessoas por meio de veículos totalmente independentes.

Objetivo

Um dos principais objetivos deste projeto é a criação de um robô totalmente autônomo que seja capaz de planejar rotas com base nos dados providos por seus sensores. Uma forma de validar o projeto será a inscrição na competição Winter Challenge da faculdade Mauá em setembro de 2019 na categoria denominada Trekking. Nesta competição o robô terá que identificar três cones e planejar uma rota para passar e encostar em todos os cones em uma ordem pré definida no menor tempo possível, em um ambiente externo com obstáculos. A seguir podemos ver uma imagem representando as posições dos cones e a ordem que o robô deverá seguir:

Imagem 1 - Vista superior com os cones em laranja numerados pela ordem que o robô deverá percorrer.

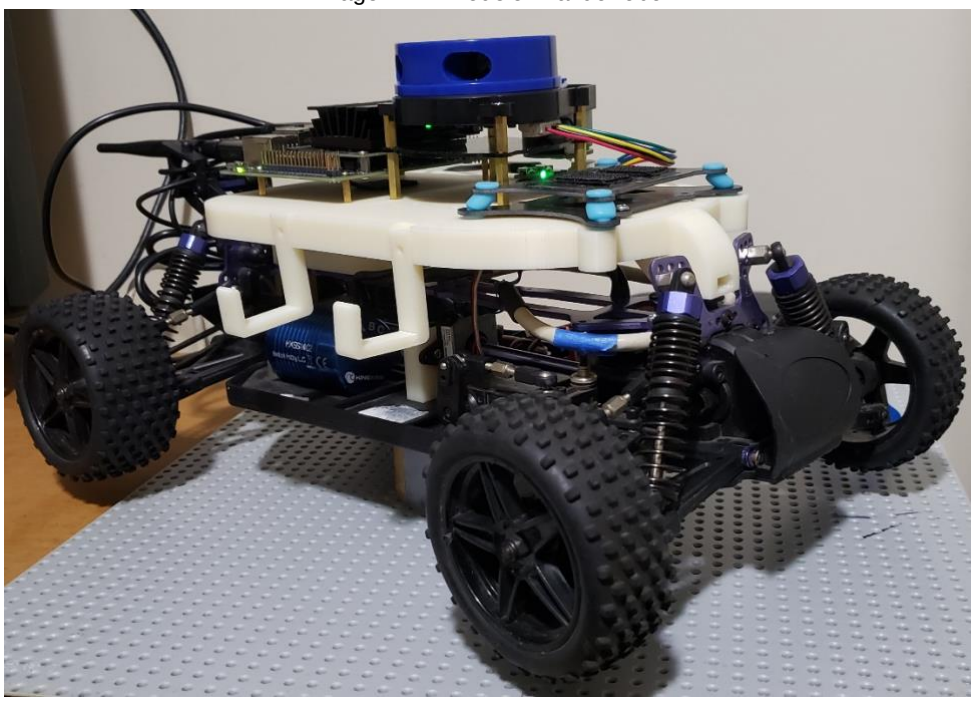


Fonte: Autor (2019)

Esse projeto irá envolver desde a implementação do hardware (atuadores, carro off-road, sensores), até o desenvolvimento do software utilizando o Robot Operating System (ROS, sistema operacional de robôs). Com tudo isso o projeto poderá também ser utilizado como objeto de pesquisa nas diversas disciplinas de robótica, pois este pode ser considerado a união entre o mundo da computação com o da

mecatrônica. E ainda pode trazer visibilidade acadêmica por meio da competição. Segue abaixo uma imagem do que seria um modelo 3D estimado do robô.

Imagem 2 – Modelo final do robô



Fonte: Autor (2019)

Metodologia

i. Escolha da plataforma

A plataforma de desenvolvimento escolhida para esta aplicação foi o ROS (Robotic Operational System), que na verdade é uma coleção de frameworks. Ele possui vários pacotes que facilitam a criação e o trabalho no desenvolvimento de robôs. Atualmente o ROS aceita Python e C++ como linguagens de programação.
Dados projetos anteriores e o nível de experiência do autor com a linguagem de programação Python, esta foi a escolha como a linguagem na implementação com o pequeno robô móvel. Além disso, o Python é uma linguagem que vem ganhando espaço e tem uma comunidade muito forte, principalmente na área de visão computacional que será abordada mais para frente neste trabalho.

ii. Hardware

O hardware utilizado é um minicarro off-road modelo SunFire da Exceedrc. O carrinho inicialmente contava com tração 4x4, um motor de corrente contínua para propulsão que era conectado a um controlador de velocidade SP03018. Porém, devido a necessidade de um motor mais eficiente e potente, foi realizada uma troca por um motor brushless Kinexsis 1/10 4-Pole 4000Kv e um ESC (Eletronic Speed Controller) que consegue controlar o motor tanto para frente quanto para trás com 70A de corrente nominal e pode ser alimentado de 2S (8,4V) até 3S (12,6V).
Também tem um servo motor de 10kg/cm que controla a direção das duas rodas da frente por meio de um sistema denominado Bell-crank steering. E para alimentar tudo isso foi utilizado uma bateria Lipo (que é a química que oferece uma das maiores capacidades energéticas por volume, atualmente) de 2200mAh, 7,4V e 70C de descarga, fornecendo em sua capacidade máxima, cerca de 154A de corrente contínua. Segue abaixo a imagem do conjunto.

Imagem 3 – Chassis e conjunto do motor

Fonte: Amazon [2] & NitroRCX [3]

Outro hardware importante para o desenvolvimento do projeto, foi a placa on-board. Ela será responsável por todo processamento dos sensores e principalmente da navegação autônoma. Inicialmente foi utilizada uma Raspberry Pi 2, que logo foi substituída por uma Raspberry Pi 3 B por questões de limitação de processamento. E finalmente quando a RPi 3 não dava mais conta, foi necessária a utilização de uma Nvidia Jetson Nano. Abaixo segue uma tabela comparando as três placas.

Tabela 1 – Comparativo entre as placas de desenvolvimento

Placa	Processador	GPU
Raspberry Pi 2	Cortex-A7 @ 900MHz Quad Core	VideoCore IV
Raspberry Pi 3	Cortex-A53 @ 1.2 GHz Quad Core	VideoCore IV
Nvidia Jetson Nano	ARM A57 @ 1.43 GHz Quad-core	128-core Maxwell

Fonte: Autor (2019)

A Jetson Nano foi carregada com a configuração padrão recomendada pela Nvidia, que contém o Ubuntu 18.04 e o ROS na versão Melodic.

Para fazer as medições da odometria do robô, foi utilizado um encoder no centro do eixo que transmite potência mecânica da parte traseira para a dianteira. O encoder é um CUI AMT-113Q-V, que possui diversas vantagens, na qual a principal é o fato de seu modo de medição dos pulsos ser capacitivo, diferentemente da maior parte dos

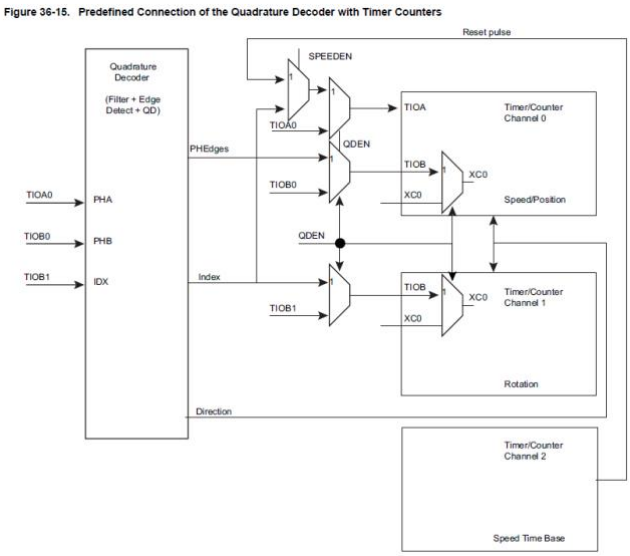
encoders no mercado que são óticos. Pelo fato de ser capacitivo o sensor é a prova de poeira e outras sujeiras que o carro estará sujeito no dia da competição. Fora isso o encoder é programável e possui em sua resolução máxima 4048 pulsos por volta, mais do que o suficiente para medir a velocidade do eixo.

Imagem 4 – Encoder CUI AMT113Q-V

Fonte: CUI [4]

Para realizar as medições da velocidade do encoder, controlar o servo motor em conjunto com o ESC e comunicar via serial com a Jetson Nano, foi utilizado um arduino DUE. Este possui um clock de 84MHz, entradas/saídas digitais e analógicas e o mais importante, um decoder de quadratura que possui velocidade suficiente para realizar a leitura do encoder e retornar o valor de velocidade.

Imagem 5 – Decoder de quadratura do Arduino DUE



Fonte: Atmel [5]

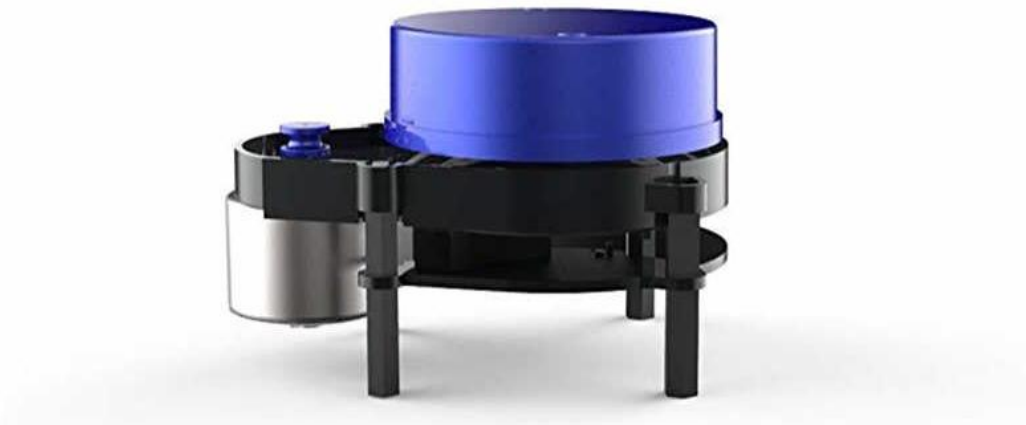
O arduino DUE possui nível lógico de 3.3V, o que é um problema uma vez que o encoder funciona apenas com 5V. Assim, foi necessária a utilização de um conversor de nível lógico para converter a entrada de 5V para 3.3V e não queimar nenhuma porta do arduino. Tudo isso foi montado sobre um protoshield para deixar o conjunto mais compacto e robusto, eliminando a utilização de jumpers que podem se tornar grandes fontes de ruído e prejudicar as medições.

Imagem 6 – Conjunto Aruino DUE e conversor de nível lógico

Fonte: Arduino [6] & SparkFun [7]

Outro sensor de grande importância para realizar o mapeamento e localização (tópicos que serão abordados nas próximas páginas) é o LIDAR ("Light Detection And Ranging"). Este sensor emite e recebe pulsos de laser, assim é calculado o tempo que leva para emitir e receber o laser e por meio da velocidade da luz é possível determinar a distância percorrida. Porém ao se montar um laser em uma plataforma giratória, transforma-se uma medida de apenas uma dimensão em uma medida com duas dimensões. O LIDAR utilizado é um YDLIDAR X4, ele possui rotação variando de 6Hz até 12Hz, com cerca de 5.000 pontos por segundo. Neste projeto o sensor foi configurado para rodar com 7Hz, assim, fornecendo cerca de $\frac{5.000\ pontos/s}{7\ rotações\ /s} \cong 714\ \frac{pontos}{rotação}$, o que dá aproximadamente um ponto a cada 0.5 graus de resolução. Essa velocidade foi escolhida, pois é um ponto de equilíbrio entre resolução e velocidade de atualização.

Imagem 7 – LIDAR YDLIDAR X4



Fonte: YDLIDAR [8]

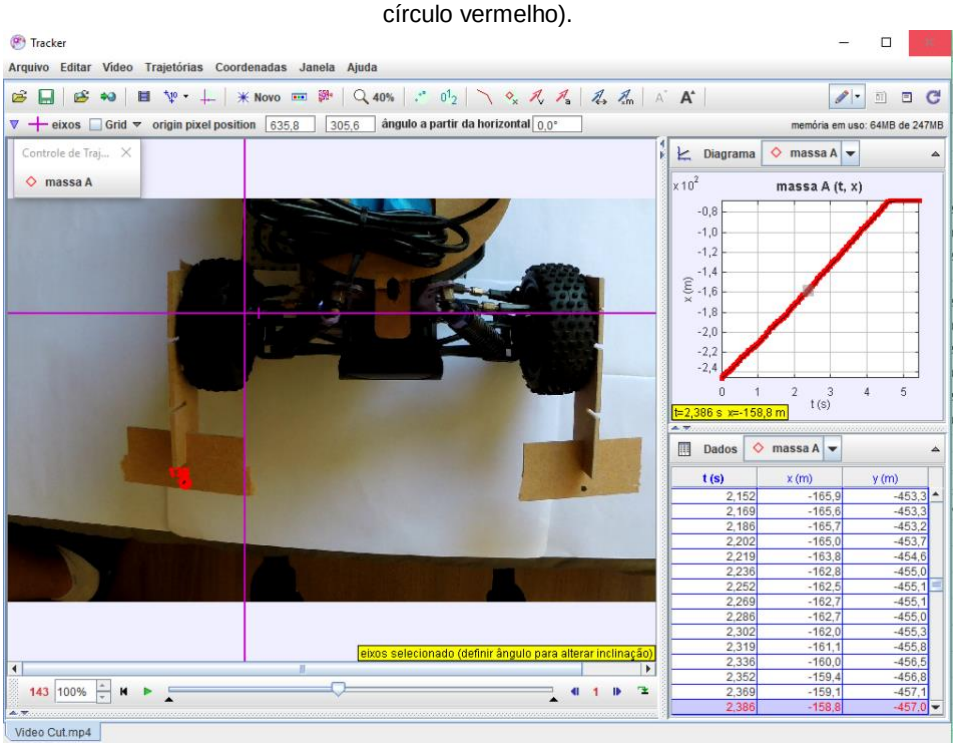
iii. Odometria

A odometria pode ser segmentado em duas etapas, o modelo de direção que está relacionado com o servo motor e o modelo da propulsão que é relacionado ao motor brushless. Ambos os modelos juntos, é possível estimar a posição do carrinho dado um intervalo de tempo e comandos de controle.

Modelo de Direção

A mecânica utilizada para direcionar o robô é conhecida como direção sino-manivela, e seu comportamento em relação a variação do ângulo é o objeto de estudo dessa secção. Para análise foi feita uma estação de teste com duas madeiras conectadas paralelamente as rodas da frente do carrinho e uma câmera por cima para filmar o movimento das rodas conforme a variação do ângulo do servo motor. Os comandos enviados ao servo foram mandados pelo Raspberry Pi. O servo rotacionava de 135 graus de um em um grau até 195 graus em intervalos de 100 milissegundos. Além disso é preciso saber qual o ângulo da roda em cada intervalo de tempo, assim foi utilizado o software livre *Tracker Video Analysis,* que é um dos melhores programas grátis para rastreamento e medições em vídeo. Em cada uma das madeiras anexadas as rodas, tinha um ponto preto e o software conseguiu realizar o rastreamento do ponto preto ao longo do vídeo e retornar uma tabela com a posição nas coordenadas X, Y e tempo. Abaixo segue a imagem do procedimento.

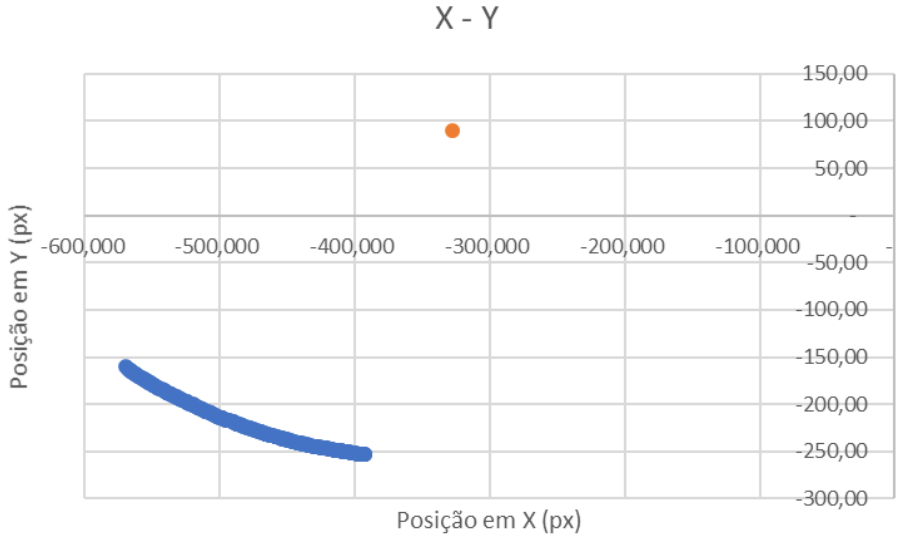
Imagem 8 – Vídeo da movimentação das rodas com o rastreamento do ponto preto na madeira (indicado por um círculo vermelho).



Fonte: Autor (2018)

Ao exportar e analisar os dados no Excel, foi possível criar o gráfico das posições X e Y com uma estimativa calculada com base nos pontos da posição do centro da circunferência criada. A seguir temos o gráfico.

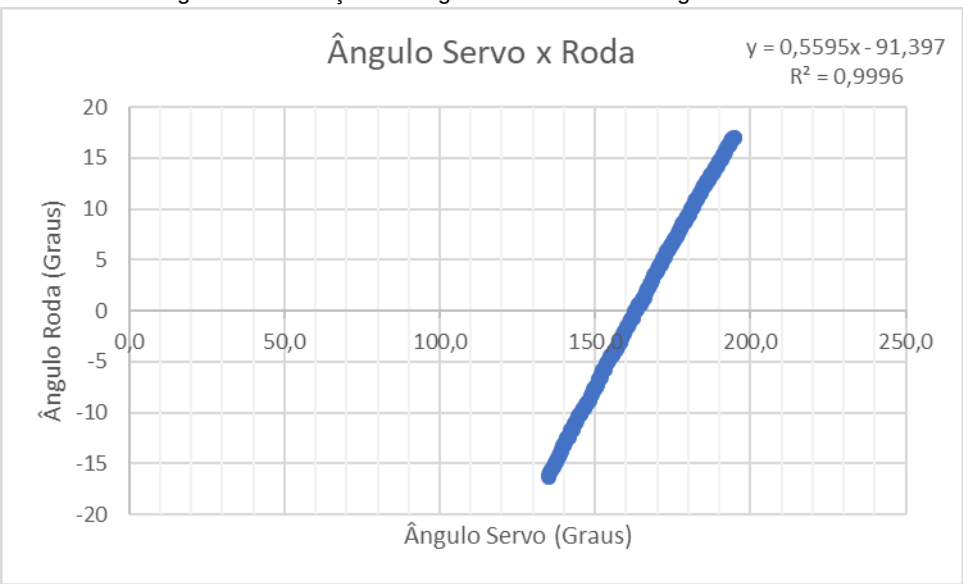
Imagem 9 – Posição em X pela posição em Y com centro estimado no círculo laranja.



Fonte: Autor (2018)

Sabendo o centro estimado do círculo é possível descobrir o ângulo de cada ponto e consequentemente é possível descobrir o ângulo da roda com comando ATAN do Excel. Ao juntar os dados de ângulo da roda com o do servo motor, foi possível descobrir uma relação entre eles e criar uma linha de tendência linear como podemos ver abaixo.

Imagem 10 – Relação do ângulo da roda com o ângulo do servo.



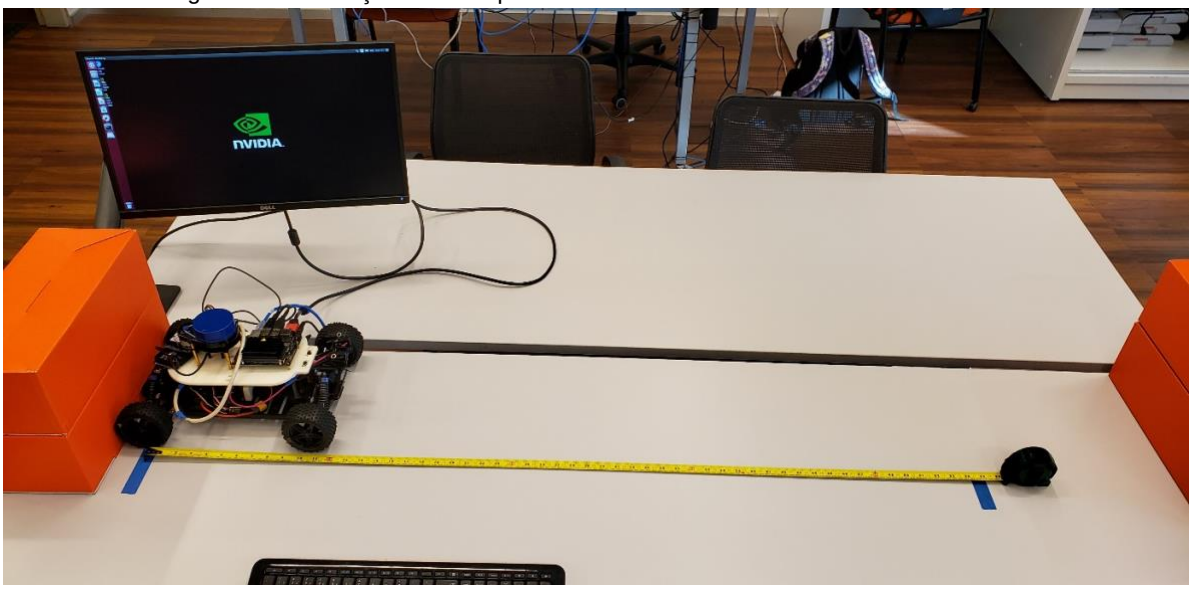
Fonte: Autor (2018)

Assim, foi encontrada a equação, em que dado o ângulo do servo, é possível encontrar o ângulo da roda, ou vice e versa. Equação é representada a seguir:
$\boldsymbol{y = 0,5595 * x - 91,397}$, onde y é o ângulo da roda e x é o ângulo do servo.

Modelo de Propulsão

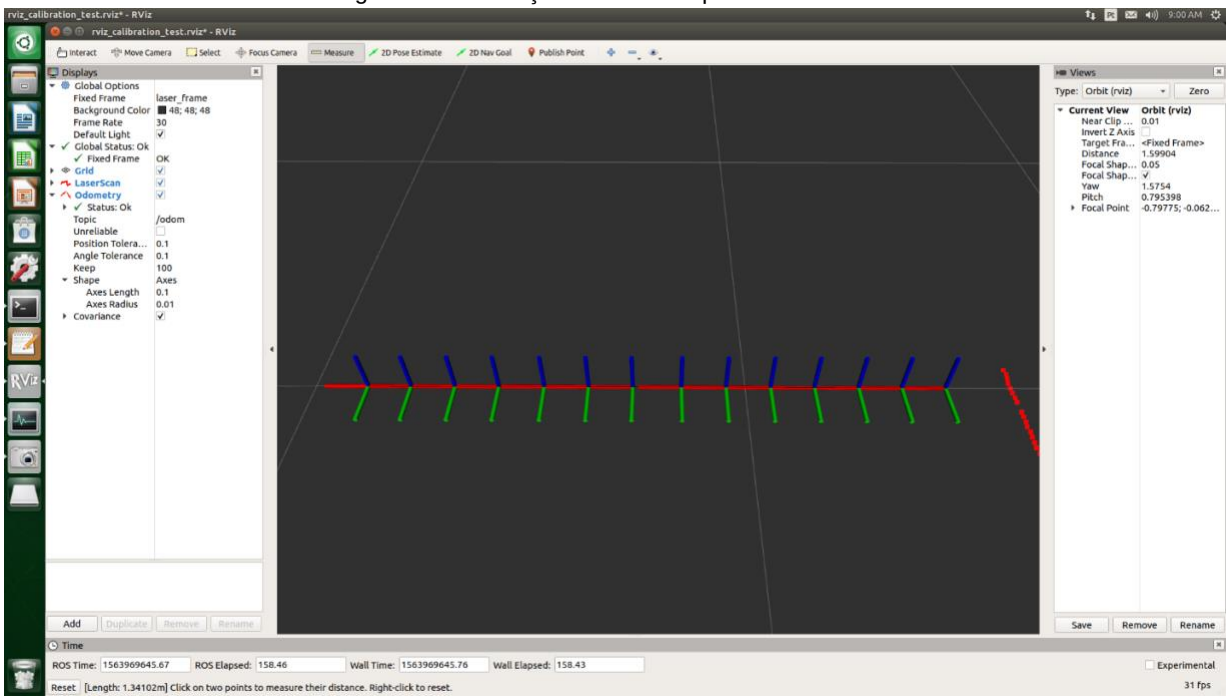
Outra parte fundamental para a construção da odometria, é a relação entre a quantidade de pulsos e a distância linear percorrida pelo carrinho, ou então, a velocidade integrada no tempo e a distância percorrida. O arduino due via comunicação serial passa os valores da velocidade em rpm do encoder. Porém ainda é necessário calibrar estes valores para que a velocidade no eixo em rpm seja convertida para a velocidade nas rodas em metros por segundo. O método utilizado para descobrir esta relação será mostrado a seguir. Primeiramente o carrinho foi colocado sobre uma mesa que continha duas caixas cuja altura era detectada pelo LIDAR. Uma caixa se encontrava na frente do carrinho e a outra na parte de trás. Pela medição do LIDAR e somando os pontos que estava em 0 grau (frente) e o de 360 graus (trás), obteve-se exatos 1,81 metros menos a largura do robô, obtendo 1,35 metros. Assim o robô foi posicionado encostando em uma das caixas e empurrado manualmente até encostar na outra caixa. Como o código da odometria já estava calculando os valores, visualizou-se no software de visualização de sensores do ROS denominado RVIZ, a transformada cartesiana se movendo conforme o movimento do carrinho. Após completar o percurso em linha reta de uma caixa até a outra, foi medido pelo RVIZ qual foi a distância percorrida por este. Caso o valor estivesse maior, então seria necessário incluir um fator que diminuísse o valor da velocidade lida pelo encoder e caso contrário seria necessário multiplicar por um valor maior. Vários testes foram feitos até chegar em um valor condizente com a realidade. Segue abaixo uma foto do ambiente de validação.

Imagem 11 – Validação do teste para obter os valores corretos de velocidade linear.



Fonte: Autor (2019)

Imagem 12 – Validação dos valores por meio do RVIZ.



Fonte: Autor (2019)

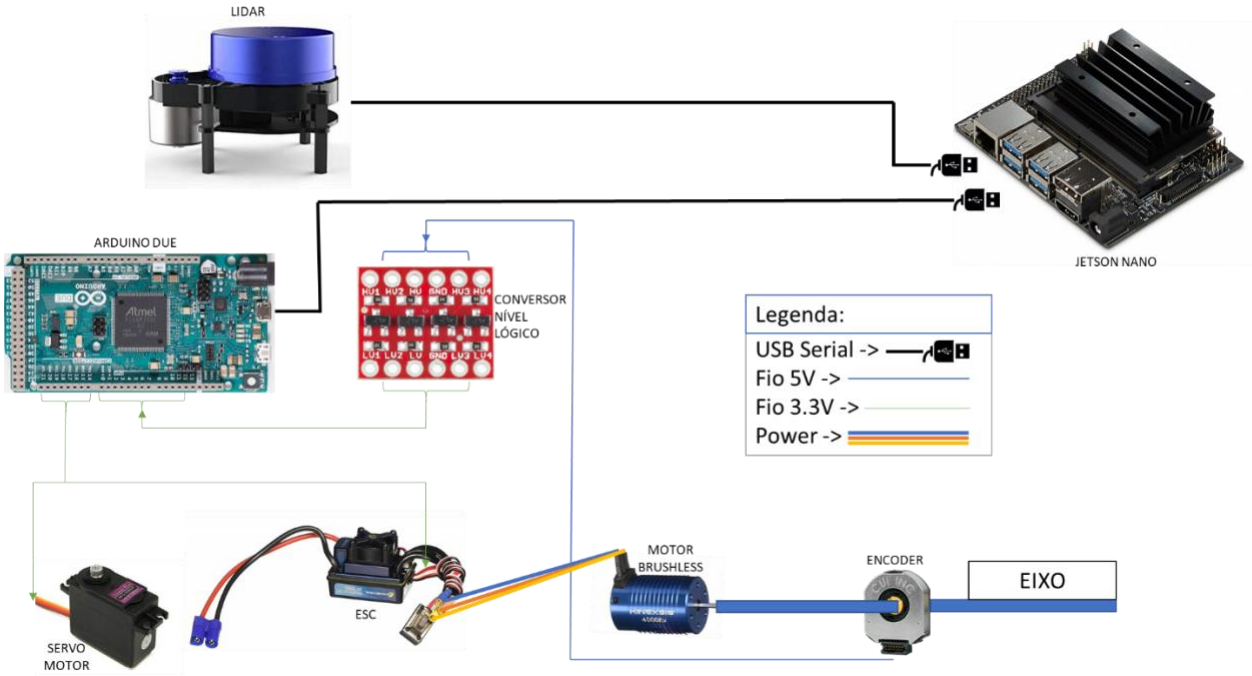
Na imagem acima é possível ver no canto inferior esquerdo o valor da distância medido no RVIZ que é cerca de 1,34 metros, ou seja, muito próximo do valor esperado. No final, obteve-se a seguinte relação:

$\boldsymbol{y = 0,000398 * x}$, onde y é a velocidade linear em metros por segundo da roda e x é o valor da velocidade em rotações por minuto no eixo da transmissão do carrinho.

Navigation stack

Após a calibração de todos os sensores e a certificação do funcionamento da odometria, foi necessária a conexão de todos os componentes para começar a trilhar os passos para o objetivo desta iniciação. O mapeamento, a localização e por fim a navegação, todos parte do um conjunto denominado navigation stack (Pilha de navegação). Segue abaixo a imagem mostrando todos os componentes conectados.

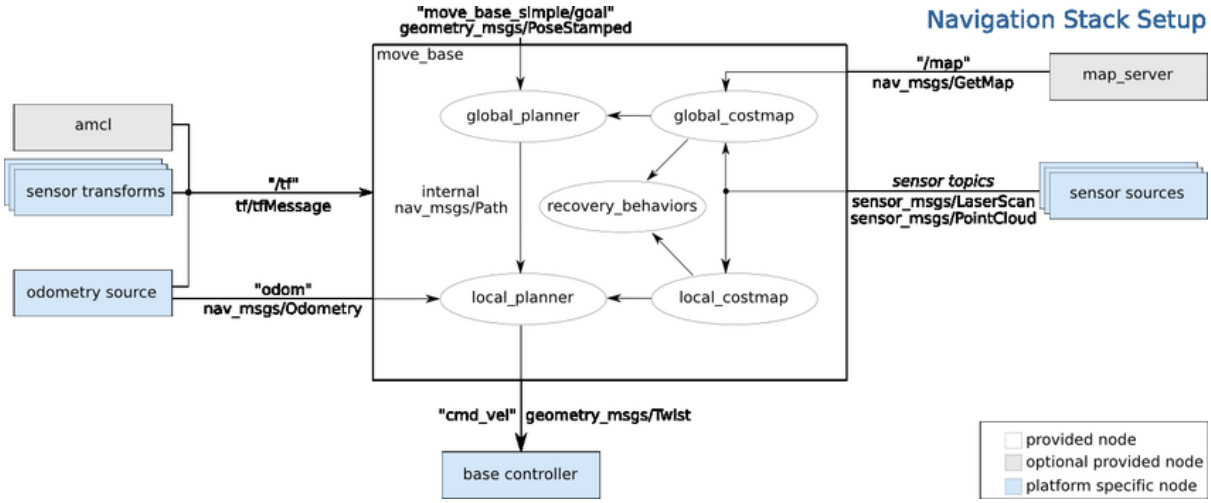
Imagem 13 – Conjunto de componentes e conexões do robô.



Fonte: Autor (2019)

Como o objetivo final do projeto envolve a navegação autônoma do robô, o ROS possui um pacote de navegação completo com várias ferramentas que podem ajudar a chegar no objetivo. Porém, para utilização destes pacotes é preciso atender certos pré-requisitos que serão avaliados a baixo. Abaixo também pode-se observar uma imagem que resume o navigation stack.

Imagem 14 – Navigation Stack do ROS.



Fonte: Research Gate [9]

Os blocos externos ao quadrado central são os “pré-requisitos” para o funcionamento do navigation stack. A explicação destes blocos se encontra abaixo:

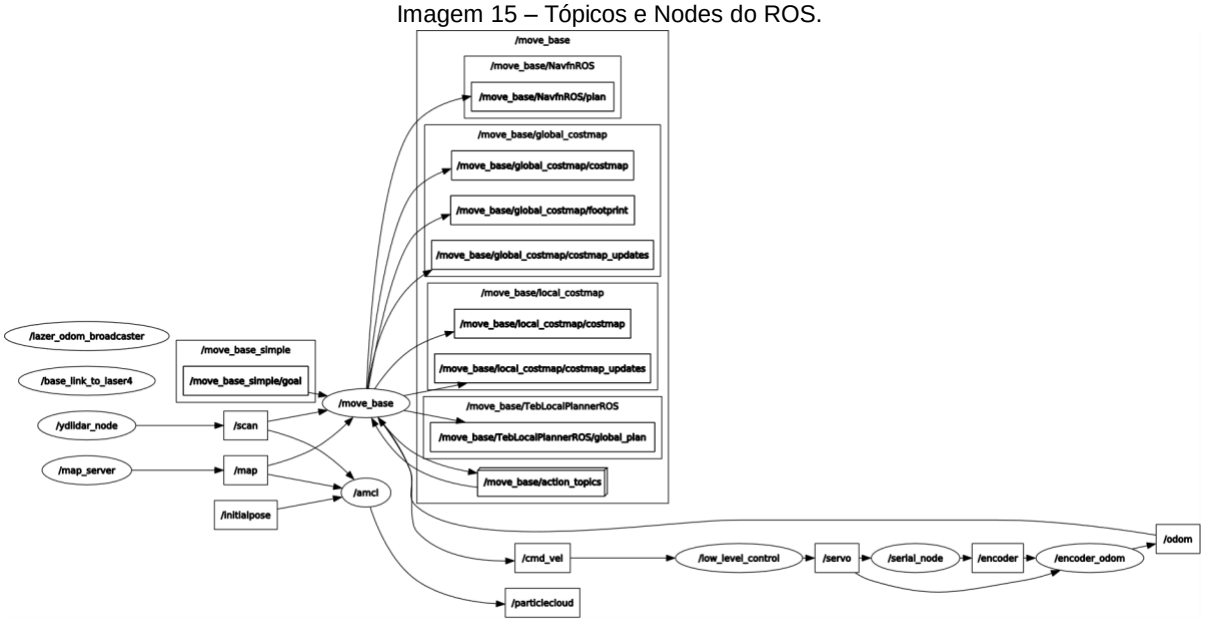
Odometry source: Odometria é publicada por meio de um código em python que faz o subscribe no valor da velocidade linear provida pela comunicação serial entre a Jetson e o Arduino e a posição angular do servo é estimada por meio dos comandos enviados no tópico “cmd_vel”.
Sensor transforms: Este bloco contém as transformadas dos sensores adotando o corpo do robô como um referencial. Como exemplo, o LIDAR fica montado no topo do carrinho, alinhado com o centro de massa, assim, sua transformada apenas contém a componente z do eixo de coordenadas.
Sensor sources: Este bloco contempla sensores como LIDAR, IMU e entre outros. No caso deste projeto, apenas o LIDAR é utilizado e, portanto, a informação fornecida por meio de seu pacote já instalado é uma nuvem de pontos.
Base controler: Este é o bloco responsável por comandar o carrinho, ele publica informações de velocidade linear e posição angular que serão convertidas devidamente para a lógica de baixo nível dos componentes de potência do carrinho e consequentemente o robô irá se movimentar.
Map server: O map server é um serviço que chama um arquivo do tipo “yaml” que possui todos os parâmetros necessários para publicar as informações referentes ao mapa.
AMCL: Adaptative Monte Carlo Localization é um algoritmo de localização. Foi utilizado a biblioteca gmapping (mapeamento e localização simultâneos) para criar um mapa. Assim, é possível realizar uma correlação entre o mapa já mapeado e os dados do LIDAR/odometria para estimar por meio da probabilidade a posição do carrinho no mapa. O interessante do algoritmo é que quanto mais você passa por lugares mapeados, maior é a certeza e menor é o desvio padrão da curva normal.

Após os pré-requisitos é necessário explicar oque ocorre dentro do navigation stack:
Global planner: é utilizado para traçar um caminho do ponto de partida até o final levando em consideração o mapa inicialmente fixo.

Local planner: foi utilizado teb_local_planner como local planner, pois ele é o mais indicado no caso de robôs não holomônicos como é o caso. O local planner é responsável pela correção da rota caso um obstáculo esteja na frente do robô.
Costmap global/local: Ambos são sobreposições feitas com base em um mapa fixo no caso do global e com base na leitura do LIDAR no caso do local. Estas sobreposições informam aos algoritmos de navegação quais regiões são seguras e livre de conflitos para que o robô não tenha colisões com obstáculos ou objetos anteriormente mapeados.

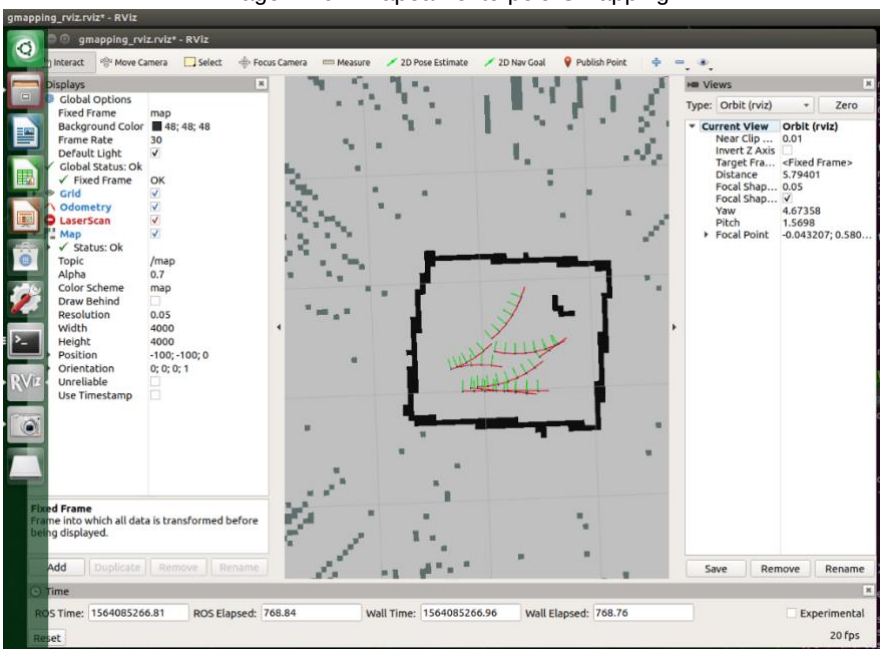
Abaixo é possível observar todos os tópicos abordados acima rodando no robô:

Imagem 15 – Tópicos e Nodes do ROS.



Fonte: Autor (2019)

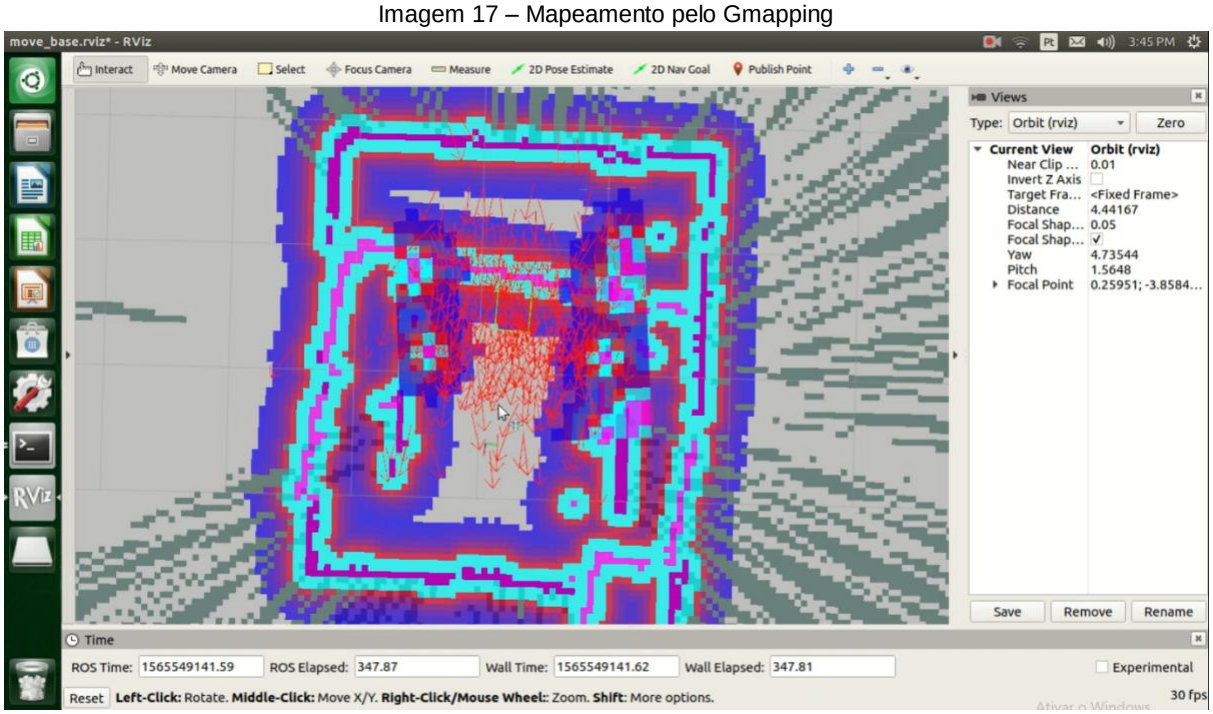
Imagem 16 – Mapeamento pelo Gmapping



Fonte: Autor (2019)

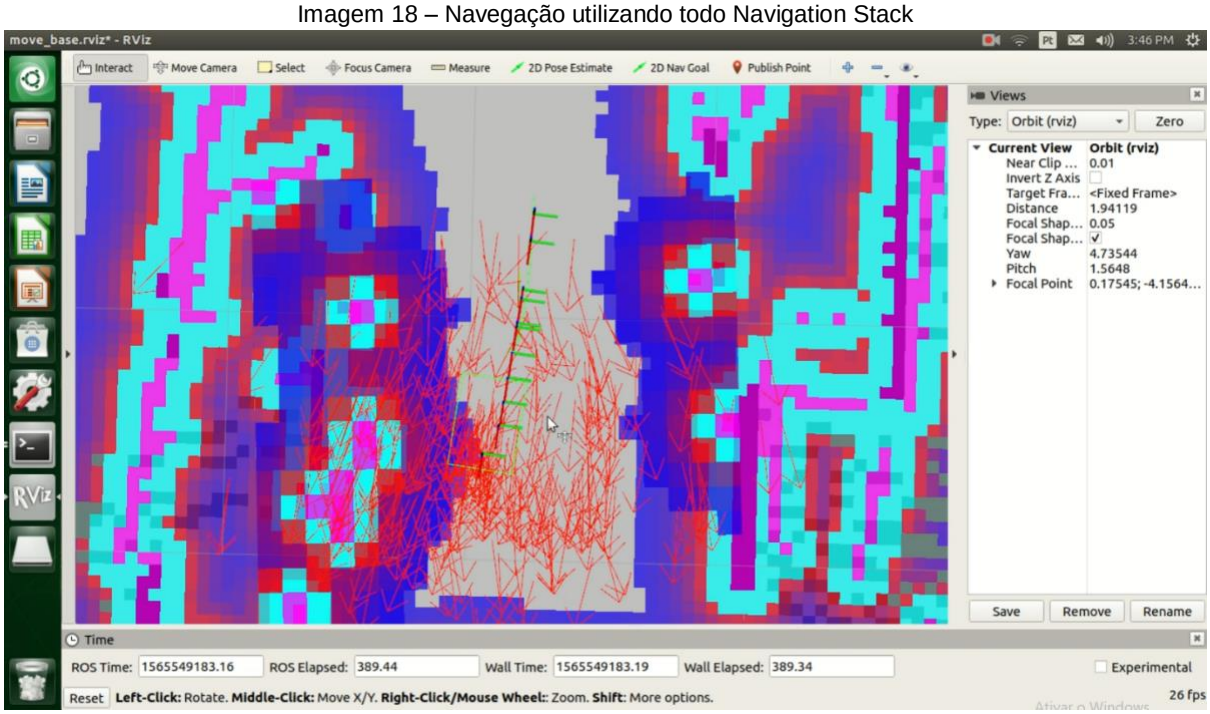
Por fim, após a implementação de todos os pacotes e ajuste de todos os parâmetros, é possível por meio do RVIZ enviar pontos de referência e o carrinho irá seguir até eles. Na figura a seguir é possível ver todos os costmaps junto com as estimativas feitas pelo AMCL. E na figura 18, podemos ver o carrinho se deslocando até um marco estabelecido pelo usuário.

Imagem 17 – Mapeamento pelo Gmapping



Fonte: Autor (2019)

Imagem 18 – Navegação utilizando todo Navigation Stack



Fonte: Autor (2019)

Conclusão

Diante de todos os fatos apresentados, pode-se concluir que a etapa de navegação autônoma foi concluída. O robô foi capaz de perceber o ambiente ao seu redor e traçar um caminho evitando obstáculos e ao mesmo tempo avaliando o caminho com menor custo possível. Ainda são necessários testes em campo para melhorar ainda mais os ajustes dos parâmetros de navegação.

Para a competição, será ideal a troca do LIDAR por uma câmera. A câmera consegue extrair características do ambiente independente da distância que eles estejam do sensor (desde que estejam no campo de visão), diferente do LIDAR que é limitado ao seu range máximo de 10 metros, que é muito menor que a distância entre os cones. A vantagem da utilização do LIDAR é a praticidade que facilita a compreensão dos programas descritos neste documento. A substituição deste não trará mais complexidade ao sistema uma vez que os princípios de mapeamento, localização e navegação podem ser facilmente adaptados.

Referências

[1] MONTEIRO, Sildomar T.; RIBEIRO, Carlos H. C. (2002). **“Obtenção de mapas cognitivos para o robô móvel magellan pro”.** Instituto Tecnológico de Aeronáutica, São José dos Campos, SP. Disponível em: Acesso em: 29 mai. 2018.

[2] **AMAZON. Kinexsis 1/10 4-Pole 4000Kv.** Disponível em: https://www.amazon.com/gp/product/B01L4C6HWC/ref=ppx_yo_dt_b_asin_title_o02_s05?ie=UTF8&psc=1. Acesso em: 9 abr. 2019.

[3] **NITRORCX. Off Road Buggy Radio Car 1/10.** Disponível em: https://www.nitrorcx.com/51c802-bahared-24ghz.html. Acesso em: 6 fev. 2019.

[4] **CUI. AMT 113Q.** Disponível em: https://www.cui.com/product/motion/rotary-encoders/incremental/modular/amt11-series. Acesso em: 7 jan. 2019.

[5] **ATMEL. SAM3X / SAM3A Series.** Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf. Acesso em: 24 jan. 2019.

[6] **ARDUINO. ARDUINO DUE.** Disponível em: https://store.arduino.cc/usa/due. Acesso em: 17 abr. 2019.

[7] SPARKFUN. **Logic Level Converter - Bi-Directional**. Disponível em: https://www.sparkfun.com/products/12009. Acesso em: 15 fev. 2019.

[8] **YDLIDAR. YDLIDAR X4 DATASHEET.** Disponível em: https://www.elecrow.com/download/X4_Lidar_Datasheet.pdf. Acesso em: 21 fev. 2019.

[9] Granosik, Grzegorz & Andrzejczak, Kacper & Kujawinski, Mateusz & Bonecki, Rafal & Chlebowicz, Lukasz & Krysztofiak, Blazej & Mirecki, Konrad & Gawryszewski, Marek. (2016). **USING ROBOT OPERATING SYSTEM FOR AUTONOMOUS CONTROL OF ROBOTS IN EUROBOT, ERC AND ROBOTOUR COMPETITIONS.** Acta Polytechnica CTU Proceedings. 6. 11. 10.14311/APP.2016.6.0011.