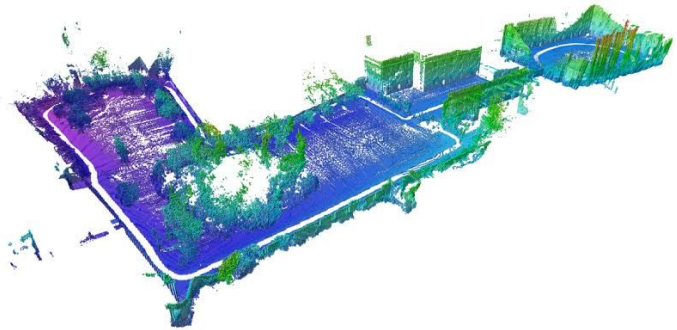
SEL0630 - Aplicações de Microprocessadores II

Augusto Ribeiro Castro - 9771421
Bruno Arantes de Achilles Mello - 9866490

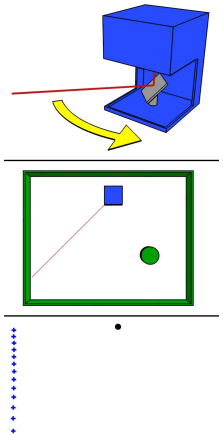
Proposta

- ➔ Mapeamento de tridimensional de ambientes utilizando um LIDAR bidimensional
- ➔ Utilizar um servomotor para girar o LIDAR
- ➔ Baseado em: https://www.youtube.com/watch?v=8ezyhTAEyHs&feature=youtu.be
- ➔ Embarcar o sistema em um robô controlado remotamente por um computador
- ➔ Criar o mapa do ambiente requer conhecer a posição atual do robô!
  - ◆ Como localizar o robô e mapear o ambiente ao mesmo tempo?
  - ◆ SLAM: simultaneous localization and mapping



LIDAR

- Tecnologia óptica de detecção remota que obtém a distância de objetos dentro de um intervalo de varredura a partir de propriedades lidas da luz refletida (diferença de fase)
- Modelo utilizado: Hokuyo URG-04LX-UG01
  - Distância de detecção: de 20 mm a 5,6 m
  - Latência: 100 ms por scan (240°)
  - Percorre um intervalo de 240° com 0,36° de resolução



SLAM

- ➔ Os dados do LIDAR são distâncias e ângulos referentes ao sensor
- ➔ Para o mapeamento, os pontos precisam ser colocados no espaço e para localizar o robô no espaço a fim de produzir novos pontos no mapa, é preciso conhecer o mapa
- ➔ **A localização depende do mapeamento e o mapeamento depende da localização (SLAM)**
- ➔ Problema genérico e muito comum: drones, carros autônomos e robôs domésticos
- ➔ Solução comum: utilizar um dispositivo que fornece a localização relativa à posição inicial do sistema

SLAM

- Possíveis soluções:
  - GPS para conhecer o posicionamento: dispositivos acessíveis financeiramente possuem erros gigantes
  - Uso de encoder óptico nos motores para estimar o deslocamento: necessita de modelo de corpo rígido da aplicação e é muito sensível a erros, como acúmulo de imprecisões e erros de derrapagem
  - Uso de uma câmera observando um ponto fixo e de coordenadas conhecidas e utilizar os parâmetros de calibração da câmera para obter a posição relativa: não usamos câmera e o uso de pontos fixos conhecidos limita a aplicação
- Abordagem escolhida: **sensor óptico de mouse**
  - Mouse comum possui resolução de 300 pontos por polegada => precisão de 0,0085 cm
  - Barato e comum, fácil acesso das leituras

SLAM

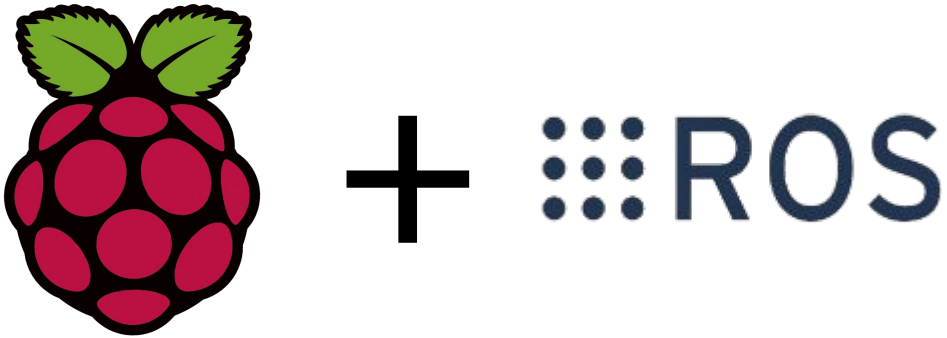
- Implicações da solução adotada:
  - Mouse não detecta giro em torno do próprio eixo: o dispositivo precisa ser colocado fora do ponto de giro do sistema
  - Como o sensor é uma espécie de câmera, para algumas cores de fundo pode não funcionar
  - Terrenos não planos dificultam o uso da abordagem
- A partir do instante inicial, o sensor óptico de mouse fornece deslocamentos em X e em Y do sistema, que são acumulados ao longo do tempo para resolver a localização. O ponto a ser inserido no mapa é a leitura do LIDAR somado da localização atual do robô

SLAM

- ➔ Como adicionar a terceira dimensão no mapa criado se os dados do LIDAR empregado são bidimensionais?
- ➔ Servomotor preso a um suporte move o LIDAR para cima e para baixo.
- ➔ Aplicar o ângulo do servomotor em cada instante aos dados recebidos do LIDAR para o mapeamento tridimensional

Como embarcar a tecnologia?

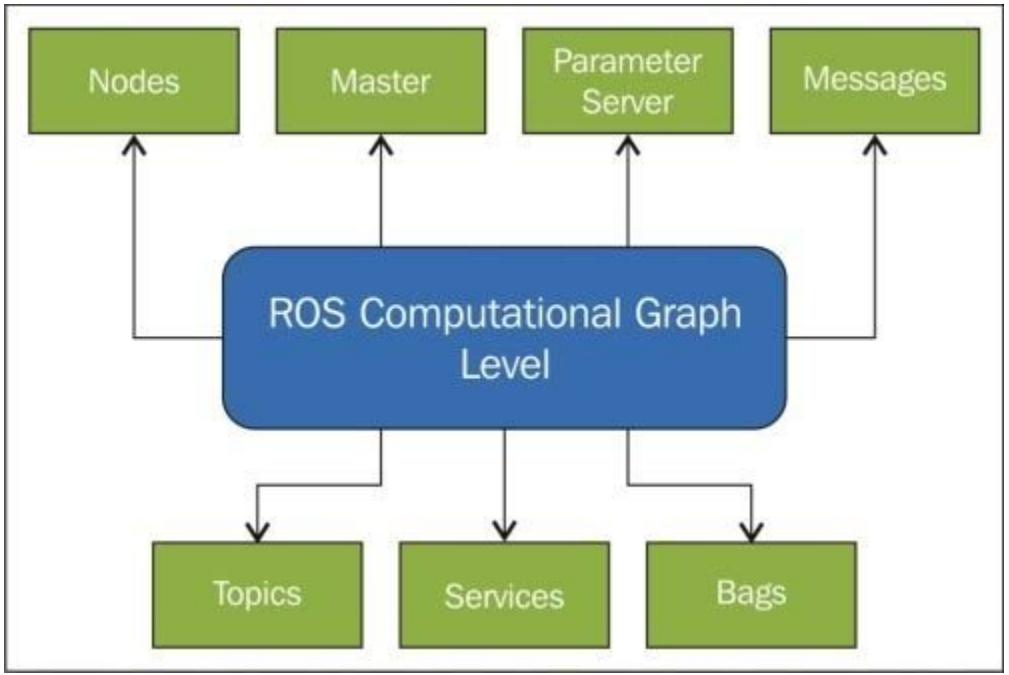
➔ Raspberry Pi 3 Model B com ROS Kinetic



ROS

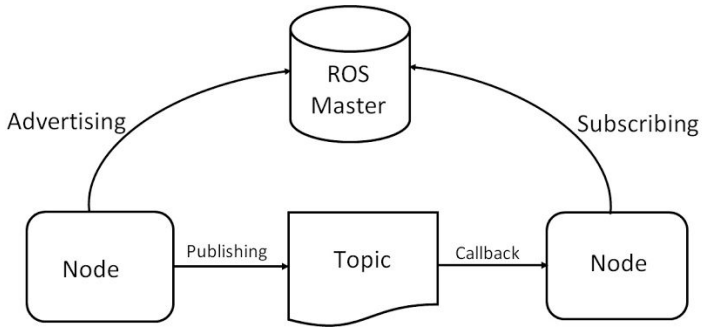
- ➔ Robot Operating System: conjunto de frameworks que fornecem serviços padrões de um sistema operacional, como abstração de hardware, controle de dispositivos de baixo nível, passagem de mensagem entre processos e gerenciamento de pacotes
- ➔ Muito utilizado para o desenvolvimento de robótica e para sistemas distribuídos
- ➔ Desenvolvido na Universidade de Stanford a partir de 2007
- ➔ Versão Kinetic Kame lançada em 2013 e suportada até 2021 é compatível com o Raspbian

ROS


Nodes
Master
Parameter Server
Messages
ROS Computational Graph Level
Topics
Services
Bags

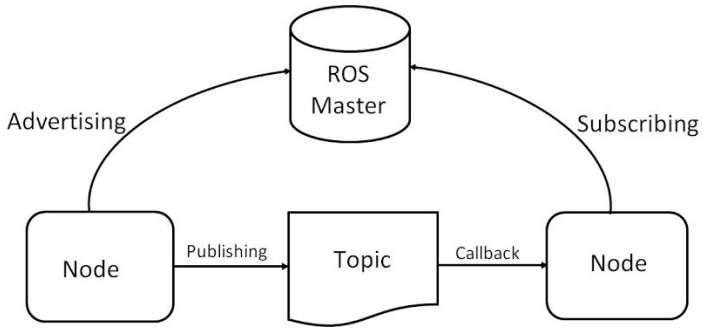
ROS Master

- ➔ Processo principal que registra os nós
- ➔ Configura a comunicação entre os nós em um tópico
- ➔ Controla parâmetros de atualização dos servidores



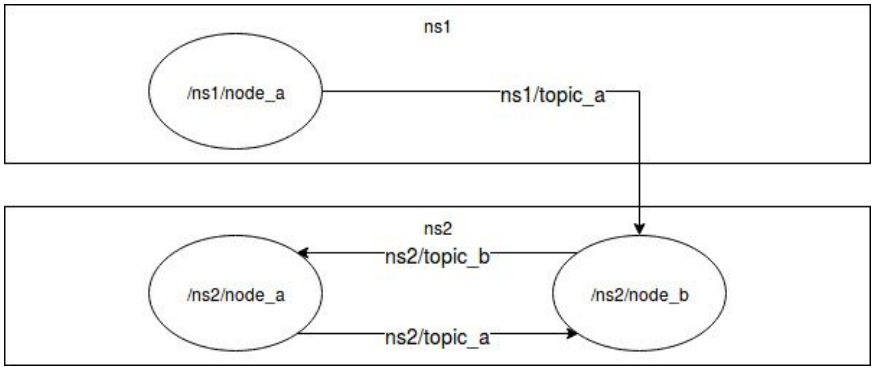
Nodes

- ➔ Os nós representam os processos em execução do ROS
- ➔ Os nós tomam ações como receber/enviar informações a outros nós
- ➔ Os nós podem receber ou enviar requisições de ações de outros nós



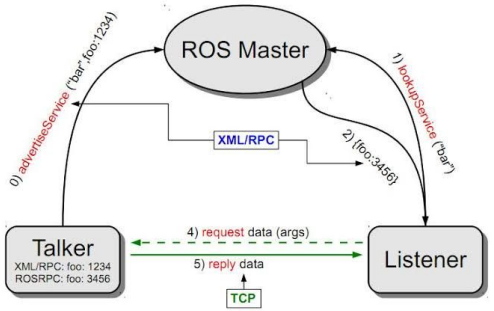
Topics

- ➔ Os tópicos são os barramentos nos quais cada nó envia ou recebe mensagens
- ➔ Quando um nó envia mensagem a um tópico, ele **publica**
- ➔ Para um nó receber mensagens de um tópico, ele **subscreve** àquele tópico
- ➔ Os nós podem publicar e subscrever a tópicos



Services

- ➔ Os nós podem prover ou requisitar serviços de um outro nó
- ➔ Usados para paradigmas de requisição e resposta (mais comuns e apropriadas a sistemas distribuídos que a estratégia de publicar/subscrever)
- ➔ As saídas de um serviço são sempre uma única saída para uma entrada



Parameter server

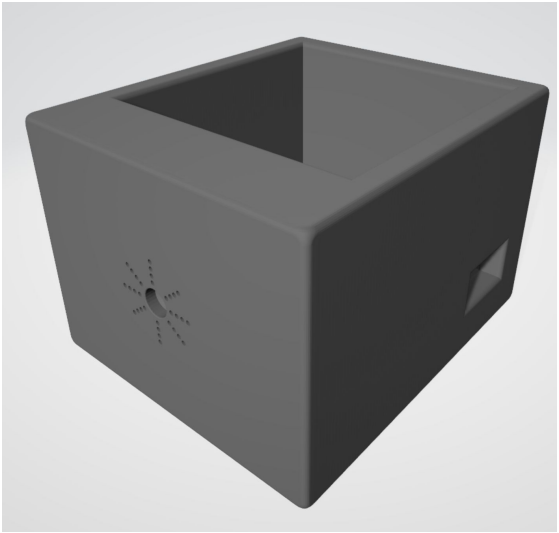
➔ Base de dados compartilhada entre nós para possibilitar acesso comum a informações que mudam pouco ou que são pouco acessadas

Linux embarcado

- ROS é desenvolvido preferencialmente para sistemas baseados em Unix
- Versão Kinetic Kame possui suporte ao Raspbian
- Raspberry Pi 3 possibilita o uso do ROS e a comunicação via rede sem fio com o computador
- Biblioteca nativa WiringPi possibilita acesso a diversos pinos de GPIO utilizando linguagem C
- A Raspberry combina poder computacional para dar suporte ao ROS com facilidade de programação de baixo nível para controlar o servomotor e possíveis periféricos necessários

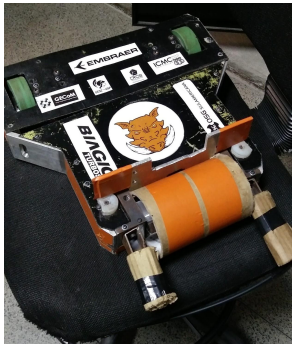
Conhecimentos extras

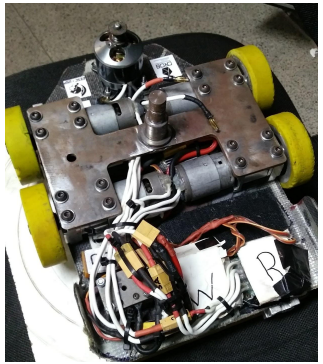
- ➔ Modelagem de um suporte para acoplar o LIDAR ao servomotor
- ➔ Uso de impressão 3D para fabricação da peça
  - ◆ Duas impressoras no prédio da engenharia de computação
  - ◆ Sala de impressoras da EESC no CRob



Aplicação do sistema

- ➔ Como forma de validação, o sistema será testado em um dos robôs de combate do Warthog Robotics
- ➔ Atualmente as categorias de combate de robôs são radiocontroladas, mas há um desejo de automatização num futuro próximo





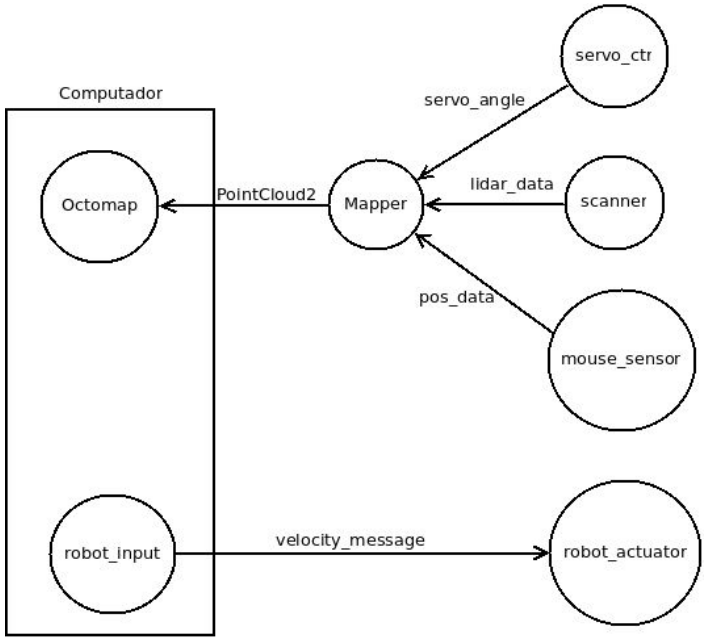
Controle da plataforma

- ➔ Para acionar os motores e controlar o robô, utilizar a Raspberry para enviar comandos utilizando a modulação de posição de pulso para a placa Scorpion XL que controla um par de dois motores
- ➔ A locomoção requer a adição de mais um nó na arquitetura
- ➔ Comandos enviados a partir do teclado do computador ao invés de via rádio: abordagem natural via ROS

Materiais usados no projeto

- ➔ Raspberry Pi 3 model B
- ➔ LIDAR Hokuyo
- ➔ Servomotor
- ➔ Mouse óptico
- ➔ Placa de acionamento de motor Scorpion XL
- ➔ Robô de plataforma

Arquitetura completa



Tarefas realizadas

- ➔ Modelagem da arquitetura e das componentes necessárias
- ➔ Nós mapper, scanner e servo_ctr
- ➔ Recebimento de distância a partir do sensor de mouse
- ➔ Servo_ctr: acionamento do motor usando GPIO da Raspberry e a biblioteca WiringPi
- ➔ Mapper: recebe mensagens, falta sincronizar por tempo e gerar a mensagem de PointCloud2
- ➔ Modelagem do suporte a ser impresso

Tarefas pendentes

- ➔ Sensor de mouse: Implementar o nó no ROS e gerar mensagens
- ➔ Servo_ctr: integrar a atuação do servo via GPIO ao nó do ROS e validar a informação de ângulo
- ➔ Mapper: Sincronizar por tempo e gerar a mensagem de PointCloud2
- ➔ Impressão do suporte projetado
- ➔ Nós robot_input e robot_actuator para envio de comandos ao robô