VIII Encontro de Iniciação Científica e Tecnológica
VIII EnICT
ISSN: 2526-6772
IFSP – Câmpus Araraquara
19 e 20 de outubro de 2023

INTEGRAÇÃO DE INTERFACES PARA ROBÔ MANIPULADOR DIDÁTICO

LUCAS QUADRELLI DA SILVA[1], RICARDO SOARES RUBIN[2]

[1] Graduando em engenharia mecânica, Instituto Federal de São Paulo, Câmpus Araraquara, lucas.quadrelli@aluno.ifsp.edu.br
[2] Prof. Ensino básico, técnico e superior, Instituto Federal de São Paulo, Câmpus Araraquara, ricardo.rubin@ifsp.edu.br

Área de conhecimento: Robotização – 3.05.05.04-6

RESUMO: Este estudo em andamento busca criar uma interface de integração em arquitetura aberta para um robô manipulador didático previamente desenvolvido com Arduino, que visa auxiliar no aprendizado e ensino de robótica nas escolas de nível técnico e superior. O foco principal está no estudo e desenvolvimento de uma interface amigável, educativa e de fácil entendimento capaz de monitorar e controlar o braço robótico por meio de cálculos matemáticos inclusos no framework de robótica ROS (Robot Operating System) e realizar simulações de movimentação a fim de elevar a qualidade das aulas de robótica sem a necessidade de grandes investimentos para aquisição de maquinário industrial e treinamento.

PALAVRAS-CHAVE: robô manipulador didático; interface de integração, ROS, arquitetura aberta.

INTRODUÇÃO

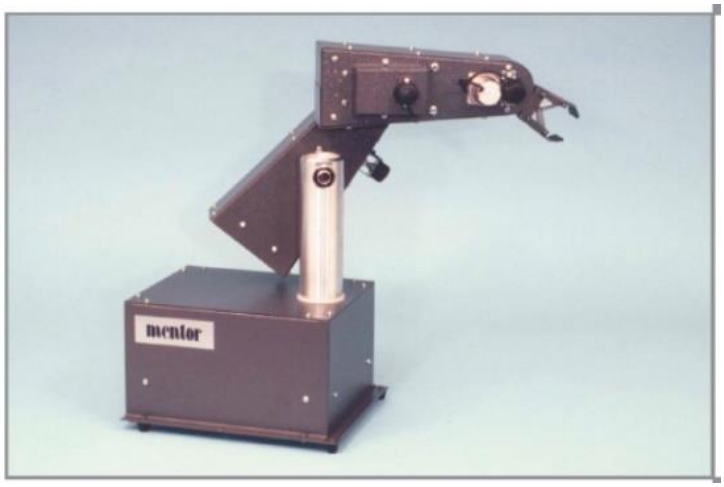
De acordo com o relatório da International Federation of Robotics (2022), no ano de 2021 houve um recorde de mais de 517 mil novos robôs industriais instalados no mundo. Com isso, a quantidade de robôs no globo já ultrapassa a marca de 3,5 milhões de unidades.

Embora existam diversas opções didáticas como por exemplo LEGO, Pitsco e VEX Robotics, os conceitos que podem ser trabalhados com eles permanecem distantes do chão de fábrica e das aplicações reais de robôs industriais. Por outro lado, o uso de robôs industriais em sala de aula, embora permitam o aprendizado de tópicos relacionados ao uso dos mesmos, ficam limitados para o ensino dos principais conceitos de robótica, uma vez que utilizam arquitetura proprietária.

O estudo da cinemática e dinâmica, ajuste dos dispositivos de controle, planejamento de trajetória, entre outros, ficam limitados a exposição do professor e a simuladores quando disponíveis. O ensino e a pesquisa em robótica requerem sistemas que sejam reconfiguráveis e reprogramáveis baseados em arquitetura aberta. (MUÑOZ, 2015).

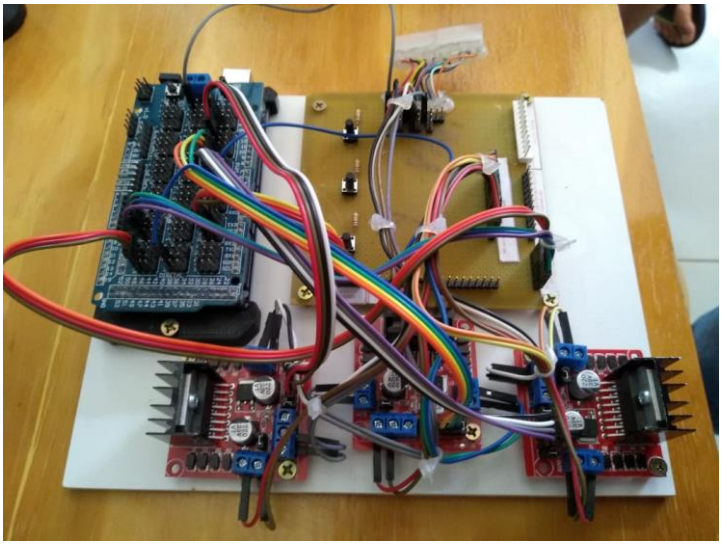
Um projeto anterior utilizou um manipulador robótico didático existente no campus e com a placa controladora danificada (figura 1) como base de partida para desenvolver um sistema que atendesse as premissas previamente elencadas. Tendo os atuadores e sensores já montados no braço robótico, foi desenvolvido o hardware controlador, baseado em Arduino, conforme figura 2, melhorando a viabilidade de custo da pesquisa.

FIGURA 1. Manipulador robótico utilizado



Fonte: (FEEDBACK INSTRUMENTS, 2006)

FIGURA 2. Sistema controlador desenvolvido



FUNDAMENTAÇÃO TEÓRICA

O sistema desenvolvido previamente e embarcado na placa controladora realiza a leitura dos sensores, aciona os atuadores e possui um controlador PID *(Proporcional Integral Derivativo)*. Durante a análise dos sensores, a resposta do sistema apresenta ruído e tempo de resposta relativamente longo, de aproximadamente 20 segundos (PONTE, 2019). Ainda segundo o autor, esses dados foram obtidos sem um ajuste fino dos parâmetros do PID. Espera-se assim que a introdução de filtros para redução do ruído e a sintonização correta dos parâmetros do PID conduzam a um resultado melhor no tempo de resposta.

Uma vez que o sistema esteja operando corretamente, a interface de integração deverá ser implementada com capacidade de realizar operações como programação ponto-a-ponto, off-line e programação por demonstração, funcionalidades requeridas por sistemas de manufatura modernos, como interfaces de comunicação com outros dispositivos, bem como as funcionalidades requeridas para o aproveitamento deste em sala de aula e em projetos de pesquisa.

Nesse sentido, o framework para desenvolvimento de aplicações robóticas ROS (2022) tem crescido ao redor do mundo como uma alternativa de robótica aberta e tem sido utilizado com sucesso. As bibliotecas desenvolvidas por Peter Corke (2022) são outras ferramentas a serem exploradas e utilizadas no projeto. Ambas utilizam a linguagem Python.

METODOLOGIA

1. Adequação do hardware e software embarcado

Para iniciar as pesquisas, foi necessário instalar o sistema operacional Ubuntu Linux – Jammy Jellyfish (22.04) via PenDrive em um computador com acesso à internet. Após a configuração do sistema, foram baixados os arquivos do framework *(ROS HAWKSBILL)* e o simulador Turtlesim.

2. Estudo do framework ROS

Realizada a familiarização com o software ROS, foi iniciado o estudo direcionado de nível básico para entender o funcionamento das ferramentas, estruturação dos comandos, tópicos, parâmetros, serviços, nós, ações, funcionamento de interfaces *(console)* que permitem a visualização de registros *(logs)* e gravação/reprodução de dados.

3. Estudo das bibliotecas do toolbox de robótica ROS 2

Foi realizado o estudo direcionado à criação de códigos para envio e recebimento de dados via publicador e inscrito, que será utilizado para enviar informações de movimentação para o robô manipulador.

4. Estudo das manipulações de dados

Estudo direcionado de nível intermediário para manipular dados, criar ações, criar nós em um único processo e monitorar mudanças de parâmetros no terminal do Ubuntu.

RESULTADOS E DISCUSSÃO

Durante as atividades de adequação, estudo do framework e biblioteca, foi visado o aprendizado dos principais itens que possibilitariam o desenvolvimento da interface de integração. Nesse processo, foram estudados e aplicados os comandos:

- *run* – roda itens (bibliotecas, pacotes, comandos, entre outros) inclusos na biblioteca ros2;
- *turtlesim* – biblioteca do simulador turtlesim;
  - *turtle_teleop_key* – inicia a entrada de comandos via teclado para controlar o simulador;
  - *rqt* – inicia o painel de edição/conferência dos parâmetros disponíveis no nó em execução;
- *node* – busca ou exibe nó disponível em uma biblioteca ou em um comando em execução;
- *list* – lista os dados disponíveis para o comando anterior, exemplo: ros2 node list (Lista os nós disponíveis para a execução ros2;
- *service* – retorna a lista de serviços ativos;
- *find* – localiza um arquivo ou diretório dentro da biblioteca descrita;
- *show* - exibe os itens disponíveis no diretório ou em um comando ativo no momento;
- *param* – refere aos parâmetros do comando a ser escrito, exemplo: ros2 param get (busca o valor dos parâmetros disponíveis em ros2);
  - *get* – faz o recebimento de um dado;
  - *set* – faz o cadastro de um dado;
  - *dump* – exporta para um arquivo de configuração;
  - *load* – carrega um arquivo de configuração;
- *info* – retorna informações disponíveis para o nó, serviço ou biblioteca descrita previamente, exemplo: ros2 node info /turtlesim;
- *action* – refere aos parâmetros para realizar uma ação;
  - *send_goal* – envia para o simulador os parâmetros a serem alcançados para finalizar uma ação;
- *rqt_console* – inicia o painel de avisos e erros do simulador ou da operação ativa no momento;
- *launch* – roda uma aplicação/configuração a partir de um arquivo programado em linguagem python;
- *bag* – refere-se à seleção de dados para exportação;
  - *record* – grava em um arquivo os dados de entrada inseridos para movimentar o simulador;
  - *play* – reproduz os movimentos gravados em um arquivo;

- *rosdep* – manuseador de dependências utilizado para criação de pacotes;

Utilizando o simulador *turtlesim*, foi possível enviar informações para que o objeto na tela (tartaruga) se movimentasse linear e/ou radialmente, exemplificando possíveis movimentos das partes móveis do robô manipulador.

Uma das principais aplicações para explorar é a gravação e reprodução de movimentos (lineares ou radiais), que permitirá o usuário criar e enviar uma série de ações para uma ou mais máquinas.

FIGURA 4. Movimentação no turtlesim

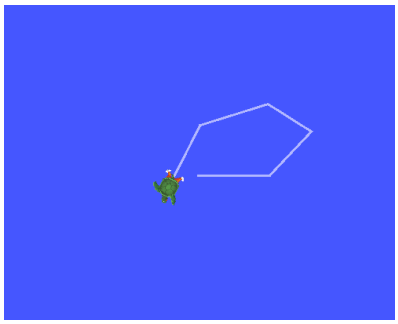
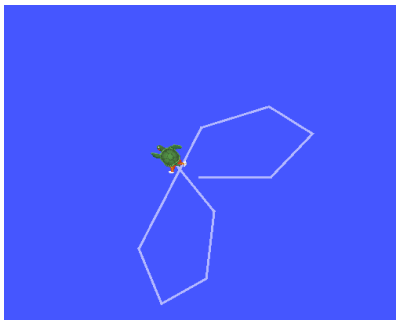


FIGURA 3. Reprodução do exemplo gravado



Até o momento, não foram realizados testes em softwares de simulação tridimensional como *Gazebo* para emular o robô manipulador citado anteriormente, porém, é notável a importância da aplicação de softwares de simulação para a aplicação correta e coerente do estudo.

CONCLUSÕES

Com este estudo, até o momento, foi possível demonstrar as etapas para o início da construção da interface de integração, visando a utilização do framework ROS, exibindo os comandos mais utilizados durante os testes de funcionalidade disponíveis na literatura. Também foi ressaltada a necessidade do conhecimento no sistema operacional Linux e seus comandos, tendo em vista que a operação do ROS é feita pelo terminal, abrindo margem para erros durante a execução dos códigos.

As próximas etapas envolvem o estudo e simulação no software *Gazebo* e o desenvolvimento da interface visual para testes práticos e de viabilidade em aulas de robótica.

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos aos orientadores por repassarem seu conhecimento durante este estudo. Também quero agradecer à instituição que disponibilizou seus equipamentos e infraestrutura, tornando possível a realização da pesquisa.

REFERÊNCIAS

CORKE, P. PeterCorke.com. Disponível em <http://petercorke.com/>. Acesso em: 03 nov. 2022.

FEEDBACK INSTRUMENTS. **Mentor Desktop Robot 35-001.** England: Feedback, 2006.

INTERNATIONAL FEDERATION OF ROBOTICS (Alemanha). **WR 2022 Industrial Robots Executive Summary.** Frankfurt: Ifr, 2022.

MUÑOZ, Francisco Mota. Development of a Multi-DOF Robotic Controller for Academic Purposes. **Journal Of Mechanics Engineering And Automation**, [s.l.], v. 5, n. 5, p.269-277, 28 maio 2015. David Publishing Company. http://dx.doi.org/10.17265/2159-5275/2015.05.001.

PONTE, C. Desenvolvimento de Interfaces Industriais para Robô Manipulador Didático. 2019 Trabalho de Conclusão de Curso, Curso de Engenharia Mecânica, IFSP, Araraquara, 2019

ROS. **Robot Operating System.** Disponível em: <http://www.ros.org/>. Acesso em: 03 nov. 2022.