Avaliação do Sistema Computacional de um Robô Explorador

Anderson Luiz Fernandes Perez, Fernando Emilio Puntel, Giann Carlos Spiler Nandi, Joildo Schueroff Elder Dominghini Tramontin

Abstract—Due to the complexity of the hardware and software on mobile robotics projects is essential that the computer system that composes a robot is robust and reliable. Commonly robots perform different tasks and thus have a complex hardware. due the complexity of the hardware it is necessary to use an operating system. It is also common for guiding robots in the environment through images, so the design of an artificial vision system is essential for the robot can achieve your goal. This paper describes an evaluation of the computer vision system and a comparison between the embedded operating systems MQX Lite and FreeRTOS for a mobile robot developed for indoor and outdoor environments. The robot consists of a modular hardware where each party is responsible for a stage of control.

Index Terms—Mobile robots, embedded operating system, artificial computer vision, MQX Lite, FreeRTOS.

I. Introdução

HÁ alguns anos, a robótica era associada somente aos braços manipuladores, empregados na construção de algum bem de consumo, tais como os robôs soldadores utilizados nas montadoras de veículos. Esta visão do ”robô industrial” vem sendo modificada ano após ano, pois é cada vez mais comum interagirmos com robôs em nosso dia a dia, por exemplo, o robô aspirador de pó Roomba ou o robô cão Aibo da Sony.

O uso de robôs móveis pode ser uma alternativa quando o tipo de atividade a ser executada possa colocar em risco a vida humana [1]. Desta forma, robôs vêm sendo utilizados para substituir os seres humanos em tarefas consideradas perigosas, repetitivas ou até mesmo estressantes [2].

Devido a esta nova era da robótica onde robôs são construídos para as mais diversificadas tarefas, desde a exploração espacial [3], acompanhamento de pessoas com necessidades especiais [4] e também auxilio na área médica [5], o projeto e a construção de tais robôs torna-se mais complexo.

Um robô móvel tem como objetivo explorar um ambiente seja ele indoor ou outdoor a procura de objeto ou objetos específicos. Este tipo de robô deve ser dotado de um sistema de visão artificial que seja robusto o suficiente para que o robô consiga identificar o objeto ao qual procura. Por exemplo, robôs para auxiliar pessoas com alguma deficiência [6], para auxiliar um motorista na condução de um veículo [7] ou até mesmo para detectar e desarmar minas terrestres [8].

Dependendo do problema ao qual o robô irá atuar, o projeto do software também se torna complexo, pois é possível que o robô tenha que executar algumas funcionalidades de maneira concorrente. Desta forma, o uso de um sistema operacional embarcado é importante, sobretudo porque o programador se preocupará somente com os aspectos funcionais do programa, ficando a cargo do sistema operacional o interfaceamento com o hardware do robô.

Neste trabalho será descrito a avaliação do sistema de visão artificial para um robô explorador de ambientes indoor e outdoor. Além do sistema de visão, os sistemas operacionais embarcados MQX Lite e FreeRTOS também são avaliados em termos da eficiência no gerenciamento e na execução das tarefas no robô. Fazendo com o que robô navegue em um ambiente aberto ou fechado desviando de obstáculos.

Este artigo está organizado como segue: na Seção II é descrito em detalhes o robô explorador, para uma melhor compreensão optou-se por descrever separadamente as arquiteturas de hardware e de software; a Seção III descreve o sistema de visão artificial para o robô explorador; a Seção IV descreve os sistemas operacionais embarcados MQX Lite e FreeRTOS; a Seção V descreve os resultados da avaliação do sistema de processamento de imagens do robô explorador; na Seção VI são descritos os resultados obtidos a partir dos experimentos com os sistemas operacionais embarcados MQX Lite e FreeRTOS; na Seção VII são feitas as considerações sobre o trabalho apresentado, bem como uma descrição sucinta de alguns trabalhos futuros.

Novembro 11, 2013

II. Descrição do Robô Explorador

O robô explorador é projetado para atuar em ambiente indoor, ou seja, em ambientes conhecidos, como uma casa, um edifício, um corredor etc.

O robô é dotado de uma câmera do tipo webcam, responsável pela captura das imagens do ambiente e alguns sensores de ultrassom, que detectam a proximidade do robô a algum objeto presente no ambiente, evitando assim o choque com obstáculos.

A base mecânica do robô é um carrinho de controle remoto modelo caminhonete, onde estão sendo adaptados todos os circuitos eletrônicos responsáveis pelo controle do robô explorador. A Figura 1 ilustra uma foto do modelo da caminhonete de controle remoto utilizada como base para o robô explorador.

Os autores Anderson Luiz Fernandes Perez (email: anderson.perez@ufsc.br), Fernando Emilio Puntel (email: fernandopuntel@gmail.com), Giann Carlos Spileri Nandi (email: giannnandi@gmail.com), Joildo Schueroff (email:joildoschueroff@gmail.com) e Elder Dominghini Tramontin (elder10010@hotmail.com), são vinculados ao Laboratório de Automação e Robótica Móvel da Universidade Federal de Santa Catarina - Campus Araranguá

Fig. 1 – Modelo da caminhonete utilizada no robô explorador

A caminhonete de controle remoto é constituída por dois motores de corrente contínua, um para controlar a direção das rodas dianteiras (direita e esquerda) e outro para controlar as rodas traseiras (seguir em frente ou recuar, dar a ré).

A Figura 2 ilustra o modelo cinemático do robô explorador onde é possível observar os deslocamentos, tanto do robô como do servo motor de posicionamento da câmera, nos eixos x e y, bem como os ângulos de movimentação.

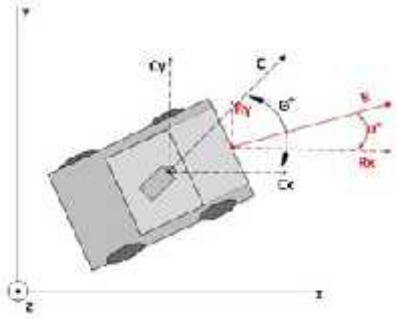

Fig. 2 – Modelo cinemático do robô explorador

A navegação do robô é baseada nas informações visuais recebidas pela câmera e também pelas informações provenientes dos sensores de ultrassom. O objetivo da navegação visual é permitir que o robô siga um determinado objeto, desta forma, conforme o objeto se movimente no ambiente, o robô também se movimentará, mantendo o objeto sempre no alvo, ou seja, no foco da câmera.

O controle de robôs móveis baseado em informações visuais é muito utilizado em robótica móvel. O trabalho de [9] apresenta um estudo comparativo de várias metodologias de navegação visual em um robô aéreo quadrimotor. A indústria automobilística também tem interesse nesta área para dotar os veículos automotores de maior segurança, permitindo que o motorista tenha mais recursos na tomada de decisões, sobretudo aquelas onde há um maior risco de acidente.

Com o aumento da complexidade nos automóveis, faz-se necessário o uso de um sistema operacional, principalmente para comunicação de dados pela rede CAN (Controller Area Network). [10] propõem troca de dados em tempo real utilizando um sistema operacional embarcado em um automóvel.

As Seções II-A e II-B descrevem, respectivamente, o projeto do hardware e o projeto do software do robô explorador.

A. Arquitetura do Hardware do Robô Explorador

DEVIDO a complexidade do projeto do robô explorador, seu hardware é composto de dois módulos, sendo cada um deles responsável por uma etapa do controle do robô. O primeiro módulo, onde acontece o processamento da imagem, é chamado de módulo Deliberativo. O segundo, chamado de módulo Reativo, é responsável pela leitura dos sensores de ultrassom e o controle dos atuadores e efetuadores dos robôs. A Figura 3 ilustra a arquitetura de hardware do robô explorador onde é possível observar os módulos Deliberativo e Reativo

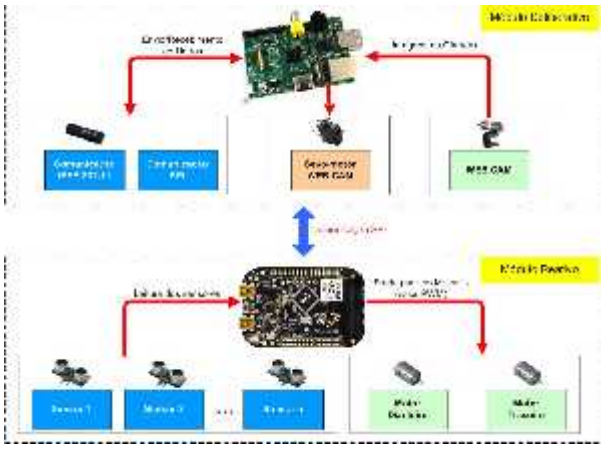

Fig. 3 – Arquitetura de hardware do robô explorador

O módulo Deliberativo é composto por uma placa do tipo Raspberry Pi [11], responsável pelo processamento das informações captadas do ambiente pela câmera, pelo controle do servo motor onde a câmera está fixada e outras tarefas responsáveis pelo controle de alto nível do robô. A figura 4 ilustra uma foto de uma placa Raspberry Pi modelo B.

Fig. 4 – Raspberry Pi modelo B

O Raspberry Pi modelo B pode ser considerado um computador de baixo custo. É dotado de um processador da família ARM de 32 bits, com clock de 700 MHz, 512 MB de memória RAM, interfaces RCA, HDMI, Ethernet e uma interface para cartão do tipo SD, onde é possível instalar o

sistema operacional. O Raspberry Pi também conta com portas de entrada e saída, GPIO (General Purpose Input/Output).

O módulo Reativo é composto por uma placa do tipo Freedom Board da Freescale [12], ilustrada na Figura 5, onde é executado o sistema operacional MQX. Esta placa é responsável por todo o processamento e controle de baixo nível do robô, tais como: leitura de sensores e envio de informações aos atuadores e efetuadores.

Fig. 5 – Freedom Board KL25Z

A divisão do sistema de controle do robô em dois módulos foi necessária devido ao volume de processamento para a execução das várias tarefas de controle. É importante salientar, conforme consta na Figura 3, que os módulos Deliberativo e Reativo irão se comunicar via protocolo SPI (Serial Peripheral Interface), um protocolo fio a fio implementado na interface de entrada e saída de ambas as placas (Raspberry Pi e Freedom Board).

B. Arquitetura do Sofware do Robô Explorador

CONFORME descrito na Seção 2.1 o hardware do robô é formado por dois módulos, sendo o primeiro composto por uma placa do tipo Raspberry Pi modelo B e o segundo por uma placa do tipo Freedom Board da Freescale. Cada módulo é responsável por uma etapa do controle do robô, desta forma, o sistema de controle também é modularizado, conforme ilustra a Figura 6.

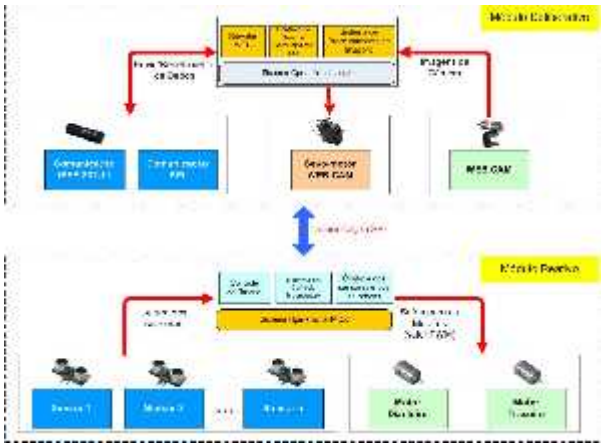

Fig. 6 – Arquitetura de software do robô explorador

O módulo Deliberativo possui um sistema operacional Linux, a distribuição Weezy configurada para o Raspberry Pi. Neste módulo são executadas três tarefas, sendo a principal o sistema de processamento de imagens, onde são processadas todas as imagens provenientes da webcam; a segunda tarefa é o controle do servo e da comunicação SPI que é responsável por reposicionar o servo de acordo com as coordenadas fornecidas pelo sistema de processamento de imagens e também enviar e receber dados via protocolo SPI para/do módulo Reativo; a terceira tarefa é um servidor WEB que fica ativo para que um usuário possa enviar comandos remotamente para o robô via rede 802.11 (rede sem fio).

O módulo Reativo é gerido pelo sistema operacional MQX Lite e FreeRTOS onde também são executadas três tarefas, sendo a primeira o sistema de controle principal deste nível, um sistema neurofuzzy (Petru Rusu, Petriu et al. 2013) que é baseado em uma rede neural do tipo Perceptron de Múltiplas Camadas com o ajuste de sua saída baseado em um sistema Fuzzy, esta tarefa é responsável por receber as informações provenientes do protocolo SPI com o módulo Deliberativo, bem como informações das tarefas controle da bateria e controle dos sensores e dos atuadores.

A tarefa controle de bateria é responsável por monitorar a carga da bateria do robô, dependendo do nível de carga de bateria algumas funcionalidades podem ser desativas. Já a tarefa controle dos sensores e dos atuadores é responsável por ler os dados provenientes dos sensores de ultrassom e os repassar ao controle neurofuzzy, este por sua vez, irá determinar uma saída para os atuadores que é basicamente é composta por direção e velocidade.

III. Sistema de Visão Artificial para o Robô Explorador

PARA [13] sistemas de visão computacional, é um sub-campo de inteligência artificial, que visa o processamento de imagens fixas e sequ^ncia de vídeos, e seus exemplos podem incluir reconhecimento, reconstrução de dados dos mais diversos possíveis, entre outros.

O sistema de visão artificial do robô explorador é baseado em uma webcam que faz a captura das imagens e está fixada a um servo motor que faz a varredura em busca do objeto padronizado inicialmente durante a configuração do sistema. O Raspberry Pi, é responsável pelo processamento das imagens recebidas pela webcam, controlando o posicionamento do servo motor conforme o deslocamento do objeto no ambiente.

Para a construção do software de visão foi utilizado o ROS (Robot Operating System) um framework de construção de sistemas robóticos, que pode ser usado nas mais diversas aplicações de software [14], bem como a biblioteca OpenCV (Open Source Computer Vision Library) [15].

Com a informação da visão, um robô pode ter a percepção do ambiente e executar tarefa complexas. No entanto, para robôs móveis, é difícil de adquirir imagens claras e adequadas nas circunstâncias dinâmicas de um ambiente, com um sistema de visão de rastreamento é uma das melhores soluções para amenizar essa dificuldade na localização de objetos [16].

O processamento de imagens baseia-se em encontrar as coordenadas x e y em relação ao centro do frame. Logo após essas operações sobre a imagem ela é descartada pelo

sistema de visão, observado que o sistema não encontra os padrões desejados, a imagem é descartada imediatamente. Vale ressaltar que que fatores externos podem comprometer a qualidade do sistema de visão, tal como a luminosidade ou algum defeito no hardware de visão

IV. Sistemas Operacionais MQX Lite e FreeRTOS

NAS subseções subsequentes serão descritos sucintamente as principais características do sistema operacional MQX Lite (Subseção 3.1) e do sistema operacional FreeRTOS (Subseção 3.2).

É importante salientar que esses sistemas foram escolhidos por sua simplicidade e facilidade de programação e também porque são compatíveis com o microcontrolador Kinetis L Cortex M0+ presente na placa Freedom Board da Freescale [12], placa esta presente no módulo reativo do sistema de controle do robô explorador, apresentado na seção anterior.

A. Sistema Operacional MQX Lite

O MQX Lite é um Sistema Operacional de Tempo Real (SOTR) totalmente integrado com a plataforma da Freescale a Freedom Board. Mesmo sendo um SOTR de pequeno porte, oferece suporte as principais funções de um SO embarcado, tais como semáforos, gerenciamento de memória e gerenciamento de threads. O kernel do MQX Lite é baseado no SOTR MQX, a Figura 7 ilustra a diferença entre os componentes suportados por ambos os sistemas [17].

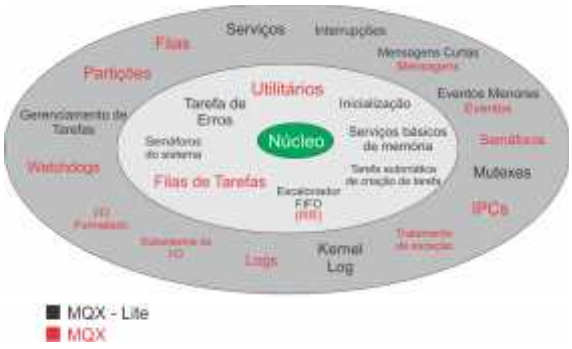

Fig. 7 – Comparação entre os sistemas MQX e MQX Lite. Extraído e Adaptado de: [18]

O escalonador de processos do MQX Lite é baseado no algoritmo FIFO (First in, First out). Possui vários componentes que são divididos em principais e opcionais. Os componentes principais representam as funções do MQX Lite que serão incluídas na imagem executável após a compilação do projeto que se está desenvolvendo, isto é, apenas os componentes inclusos na placa. Porém um aplicativo pode estender suas funcionalidades, configurando os componentes do núcleo e adicionando outros componentes [18].

Para alocação de memória o MQX Lite fornece alocação de memória leve, processo de alocação muito parecido com o padrão das funções da biblioteca padrão da linguagem de programação C malloc() e free(). A diferença é que o componente de alocação de memória fornece um mecanismo seguro tanto para alocação e como para a liberação de memória para tarefas concorrentes [18].

B. Sistema Operacional FreeRTOS

O FreeRTOS é um SOTR de código aberto. Apresenta um bom desempenho e possui um núcleo pequeno, com um escalonador de processos baseado no algoritmo de prioridades, também suporta semáforos binários e filas de mensagens [19].

Atualmente o FreeRTOS possui versões para 23 (vinte e três) arquiteturas que variam de 8 a 32 bits. Suas principais características são a portabilidade, escalabilidade e simplicidade, além de suportar um número elevado de tarefas [20].

O kernel do FreeRTOS é totalmente escrito na linguagem C, composto por quatro arquivos, sendo assim legível e de fácil manutenção [21]. A Figura 8 ilustra a estrutura do kernel do sistema operacional FreeRTOS.

Fig. 8 – Estrutura do kernel do sistema FreeRTOS Extraído e adaptado de: [22]

Um SOTR aloca memória para cada tarefa, fila, mutex e semáforo que é com as função malloc() e depois libera a memória com a função free(), porém nem sempre estão disponíveis em sistemas embarcados, além de ocuparem um espaço considerável no programa. Para contornar este problema o FreeRTOS matém sua API (Application Programming Interface) de alocação de memória em sua camada portátil, que está fora dos arquivos de origem que implementam as funcionalidades do núcleo RTOS, permitindo uma implementação específica para o sistema de tempo real que está sendo construído [23].

V. Avaliação do Sistema de Visão do Robô Explorador

A A avaliação do sistema de visão artificial foi realizada com o objetivo de o robô localizar um quadrado de cor amarela em um cubo de Rubik. Durante o experimento um quadrado de cor amarela foi colocado em posições diferentes a fim de testar a efetividade do sistema de visão e de localização do alvo pela câmera.

Uma imagem digital basicamente é uma matriz de pixels (picture elements), cada qual contendo valores de intensidade das 3 (três) cores básicas: azul, verde e vermelho. Para realizar

o reconhecimento de padrões, primeiramente é realizada a binarização da imagem. A binarização consiste em gerar uma nova matriz, de mesmo tamanho, e analisar pixel a pixel resultando em ’1’ caso os valores de RGB estejam dentro de um limite específico e ’0’ caso estejam fora deste intervalo.

A Figura 9 e a Figura 10 ilustram, respectivamente, a posição do quadrado de cor amarela em dois experimentos que foram realizados. Nas figuras percebe-se uma borda verde no quadrado amarelo indicando que o sistema de visão detectou o alvo e o está perseguindo, ou seja, caso o objeto se movimente o sistema de visão fará com que a câmera se movimente também objetivando manter o foco no objeto que se está perseguindo.

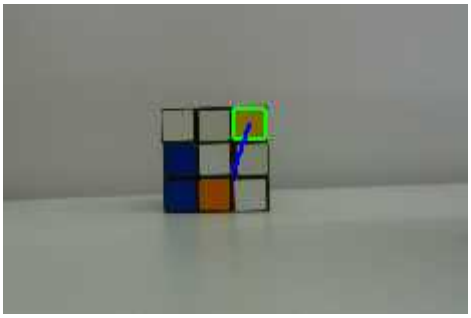

Fig. 9 – Primeiro exemplo de detecção do objeto

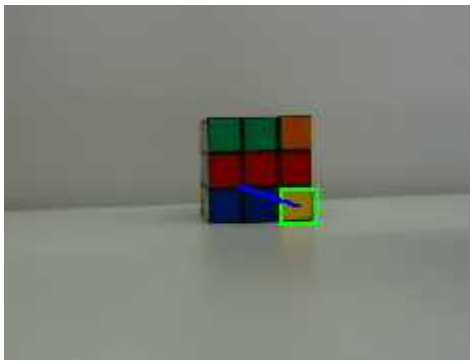

Fig. 10 – Segundo exemplo de detecção do objeto

O sistema de visão artificial é responsável por processar as imagens capturadas pela câmera para calcular o ângulo e a posição que o objeto rastreado se encontra do robô. Essas informações são necessárias para controlar o servo motor da câmera e também são enviadas via SPI para o módulo reativo que aciona os motores do robô a fim de posicioná-lo de frente ao objeto rastreado.

É importante salientar que o nível deliberativo recebe do nível reativo informações sobre o nível de carga da bateria do robô. Essa informação da bateria pode ser acessada por um usuário externo via internet, pois o nível deliberativo possui um servidor web que pode ser acessado para que, eventualmente, alguém controle o robô remotamente ou para visualizar o nível de bateria do mesmo.

Durante o processo de extração de características o módulo visão artificial extraí imagens que são passadas para um servidor web a cada 5 (cinco) segundos independentemente das características das imagens passadas.

Fig. 11 – Interface Controle do Sistema Remoto

A 11 ilustra a interface do controle remoto do robô explorador. Com o controle remoto o usuário poderá obter dados sobre velocidade, ângulo do motor de passo dianteiro, último comando executado e a quantidade de carga da bateria.

VI. Avaliação dos Sistemas Operacionais Embarcados MQX Lite e FreeRTOS no Robô Explorador

A avaliação dos sistemas operacionais MQX Lite e FreeRTOS foram baseadas em testes realizados em bancada. Foram analisados os tempos efetivos de execução de duas tarefas do módulo reativo: a tarefas de controle dos motores e a tarefa de controle dos sensores.

Foram realizados dois experimentos onde o primeiro atestou a entrada em execução de ambas às tarefas após serem selecionadas pelo escalonador de processos e o segundo avaliou a execução das tarefas manipulando os atuadores. Em ambos os experimentos foi atribuído uma prioridade padrão para as tarefas, ou seja, 8 (oito) no MQX Lite e 0 (zero) no FreeRTOS. Os dados dos experimentos foram obtidos com o auxílio de um osciloscópio.

A. Experimento 1 - Entrada das Tarefas em Execução

Este experimento teve como objetivo validar a entrada em execução das tarefas e a efetividade do escalonador de tarefas. Na Figura 12 e na Figura 13 são ilustrados o comportamentos dos dois sistemas operacionais.

Neste experimento a tarefa sensores somente realiza a leitura dos sensores dianteiro do robô. E a tarefa motores faz a verificação do dado recebido a partir dos sensores. A atividade das tarefas é sincronizada com o uso de um semáforo, onde a tarefa sensores obtém o semáforo, fez a leitura dos sensores dianteiros e libera o semáforo para que a tarefa dos motores possa executar. Neste experimento a tarefa dos motores apenas faz o teste dos valores obtidos pelos sensores.

Ao final da execução cada tarefa faz uma chamada de sistema yield() para liberar o processador para que outra tarefa entre em execução. Para melhor visualização das tarefas foi necessário adicionar um delay de 150 ms (millisegundos) nas tarefas. Com os resultados obtidos é possível visualizar que ambos os sistemas operacionais obtiveram desempenho semelhante.

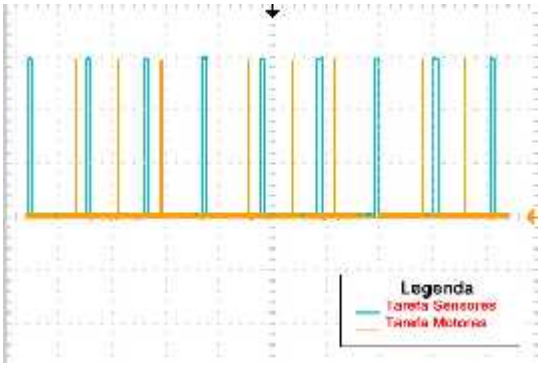

Fig. 12 – Resultado obtido com o sistema operacional MQX Lite

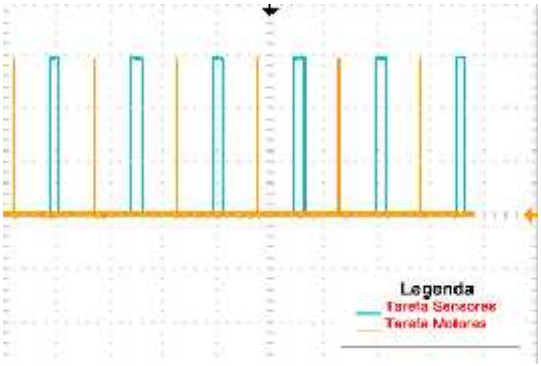

Fig. 13 – Resultado obtido com o sistema operacional FreeRTOS

B. Experimento 2 - Execução das Tarefas Manipulando os Atuadores

O objetivo deste experimento é avaliar o desempenho das tarefas na manipulação dos dados obtidos através de um sensor e conforme resultado ativar os atuadores. Na Figura 14 e na Figura 12 é possível ver o desempenho dos dois sistemas operacionais embarcados.

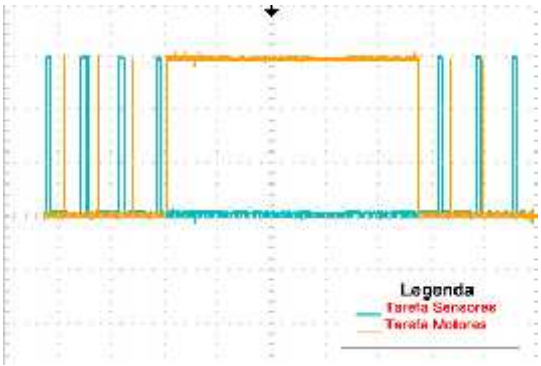

Fig. 14 – Teste MQX Lite com atuadores

Para avaliação foi utilizado os sensores dianteiros e traseiros. Neste caso quando o sensor dianteiro identificasse um objeto a menos de 45 cm, imediatamente mudava o sentido do motor traseiro, e então começava verificar o valor recebido pelo sensor traseiro, até que a distância do sensor dianteiro fosse aceitável para continuar para frente.

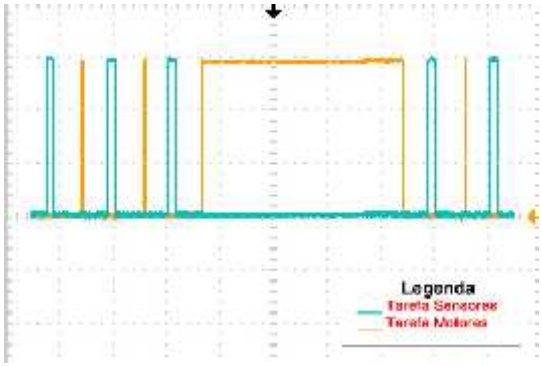

Fig. 15 – Teste FreeRTOS com atuadores

Neste teste é possível verificar que o sistema embarcado MQX Lite realizou a operação de mudar o sentido do motor traseiro de uma forma mais eficaz enquanto o FreeRTOS teve melhor desempenho na leitura dos sensores.

É possível verificar que o FreeRTOS demora 2,5 segundos para mudar o sentido do motor traseiro, enquanto o MQX Lite demorou apenas 2 segundos para realizar a mesma tarefa.

Vale ressaltar que mesmo o robô possuindo 6 sensores (3 dianteiros, 3 traseiros) neste experimento foram utilizados somente um sensor dianteiro e um traseiro.

Salientando que em ambos os experimentos é possível visualizar que em alguns momentos o processador foi utilizado para executar tarefas específicas do próprio SO.

VII. Considerações Finais

Este artigo foram apresentados os resultados da comparação de desempenho dos sistemas operacionais embarcados MQX Lite e FreeRTOS aplicados em robótica móvel desenvolvido para exploração de ambientes. Também foi descrito o sistema de visão artificial para o mesmo robô.

O robô explorador possui navegação baseada em informações visuais onde é possível marcar um determinado alvo para que o robô o siga. Com o objetivo de tornar o robô mais robusto, optou-se pela separação do hardware em dois módulos funcionais, sendo o primeiro, chamada de módulo Deliberativo, responsável por todo o processamento das imagens provenientes de uma câmera. Este módulo se comunica com o módulo Reativo, que possui um sistema operacional embarcado que é responsável pela leitura dos sensores de ultrassom e o controle dos atuadores e efetuadores. A comunicação entre ambos os módulos acontece via protocolo SPI.

O módulo deliberativo é baseado em uma placa do tipo Raspberry Pi modelo B e o módulo reativo em uma placa do tipo Freescale Freedom Board, módulo reativo. O sistema de visão artificial é executado no módulo deliberativo e é responsável por fazer com que o robô mantenha no foco da câmera um objeto pré-determinado. Os sistemas operacionais MQX Lite e FreeRTOS foram executados no módulo reativo gerenciando algumas tarefas deste mó dulo.

Os resultados obtidos demonstram que ambos os sistemas operacionais são indicados para aplicações de robótica móvel. O algoritmo desenvolvido para ambos os sistemas embarcados se mostrou bastante robusto, demonstrando um desempenho favorável nos experimentos realizados em bancada. Destaca-se o tamanho reduzido os sistemas embarcados utilizados, facilitando a experimentação e permitindo o uso de uma plataforma com pouca memória. O escalonador de tarefas de ambos os sistemas operacionais se mostraram eficientes.

Os experimentos apresentados com o sistema de visão artificial mostraram que o sistema de controle funciona bem mesmo com interferências de outros objetos. Para validar o sistema foi elaborado um experimento onde o robô necessitava encontrar um quadrado da cor amarela em um cubo de Rubik. Nos experimentos realizados o sistema de visão artificial se mostrou eficiente localizando o objeto mapeado em diferentes posições no ambiente.

Como trabalhos futuros pretende-se realizar a implementação de uma tarefa para controle de bateria, a implementação de um sistema de controle baseado em redes neurais artificiais e lógica fuzzy (sistema neuro-fuzzy) e a integração do módulo reativo com o módulo deliberativo a partir da comunicação via protocolo SPI.

VIII. Agradecimentos

Os autores, Elder Dominghini Tramontin, Fernando Emilio Puntel, Joildo Schueroff e Giann Carlos Spileri Nandi, agradecem a Universidade Federal de Santa Catarina pela bolsa de estudos.

References

[1] M. Brunner, B. Brüggemann, and D. Schulz, "Motion planning for actively reconfigurable mobile robots in search and rescue scenarios," 2012.

[2] F. Kanehiro, E. Yoshida, and K. Yokoi, "Efficient reaching motion planning and execution for exploration by humanoid robots," in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012, pp. 1911–1916.

[3] J. P. Laboratory, "Mars science laboratory curiosity rover." 2013.

[4] K. Mykoniatis, A. Angelopoulou, and J. Kincaid, "Architectural design of artemis: A multi-tasking robot for people with disabilities," in Systems Conference (SysCon), 2013 IEEE International, 2013, pp. 269–273.

[5] K. Entsfellner, R. Tauber, D. Roppenecker, J. Gumprecht, G. Strauss, and T. Lueth, "Development of universal gripping adapters: Sterile coupling of medical devices and robots using robotic fingers," in Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on, 2013, pp. 1464–1469.

[6] M. Betke, J. Gips, and P. Fleming, "The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities," Neural Systems and Rehabilitation Engineering, IEEE Transactions on, vol. 10, no. 1, pp. 1–10, 2002.

[7] F. Stein, "The challenge of putting vision algorithms into a car," in Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on. IEEE, 2012, pp. 89–94.

[8] P. Gader, J. M. Keller, H. Frigui, H. Liu, and D. Wang, "Landmine detection using fuzzy sets with gpr images," in Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, vol. 1. IEEE, 1998, pp. 232–236.

[9] Y. Qi, "A performance analysis of vision-based robot localization system," 2013.

[10] C.-S. Tsai, K.-S. Tsai, and M.-T. Hsu, "An implementation of the enhanced-can bus network connection in car real-time embedded software system," in Control, Automation and Systems (ICCAS), 2012 12th International Conference on, 2012, pp. 277–283.

[11] M. Schmidt, "Raspberry pi. a quick-start guide," Progmatic Bookshelf, 2012.

[12] Freescale, FRDM-KL25Z User's Manual, Freescale, 09 2012.

[13] D. Gerónimo, J. Serrat, A. M. López, and R. Baldrich, "Traffic sign recognition for computer vision project-based learning," 2013.

[14] W. Garage, "Robot operating system (ros)," 2010.

[15] G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library. O'reilly, 2008.

[16] W. Bahn, J. Park, C.-h. Lee, T.-i. Kim, T. Lee, M. M. Shaikh, K.-s. Kim, and D.-i. Cho, "A motion-information-based vision-tracking system with a stereo camera on mobile robots," in Robotics, Automation and Mechatronics (RAM), 2011 IEEE Conference on. IEEE, 2011, pp. 252–257.

[17] Silica, Freescale MQX Software Solutions, 2012.

[18] Freescale, "Freescale mqx real-time operating system - users guide," vol. Acesso em: 06, set de 2013, 2013.

[19] FreeRTOS, The Standard Solution For Small Embedded Systems, Technical Report,, http://www.freertos.org., 2008.

[20] R. Inam, J. M. aki Turja, M. S. . odin, S. M. H. Ashjaei, and S. Afshar, "Support for hierarchical scheduling in freertos," Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, p. 10, 2011.

[21] R. Barry, "Real time application design using freertos in small embedded systems," 2003.

[22] S. Niu, J. Zhang, B. Wang, and X. Fu, "Analysis and implementation of migrating real-time embedded operating system freertos kernel based on s3c44b0 processor," in Information Science and Engineering (ISISE), 2012 International Symposium on. IEEE, 2012, pp. 430–433.

[23] R. Berry, FreeRTOS User Manual, 1st ed., FreeRTOS, 2007.

Anderson Luiz Fernandes Perez Professor dos curso de Tecnologias da Informação e Comunicação e Engenharia da Computação na Universidade Federal de Santa Catarina. Membro do Laboratório de Automação e Robótica Móvel - LARM.

Elder Dominghini Tramontin Acadêmico do curso de Tecnologias da Informação e Comunicação na Universidade Federal de Santa Catarina. Membro do Laboratório de Automação e Robótica Móvel - LARM.

Fernando Emilio Puntel Acadêmico do curso de Tecnologias da Informação e Comunicação na Universidade Federal de Santa Catarina. Membro do Laboratório de Automação e Robótica Móvel - LARM.

Gian Carlos Spiler Nandi Acadêmico de engenharia de computação na Universidade Federal de Santa Catarina. Membro do Laboratório de Automação e Robótica Móvel - LARM.

Joildo Schueroff Acadêmico do curso de Tecnologias da Informação e Comunicação na Universidade Federal de Santa Catarina. Membro do Laboratório de Automação e Robótica Móvel - LARM.