Uso de um braço robótico como elemento de aprendizagem no ensino de Sistemas Operacionais

Rafael Henrique de Andrade, rafaelhandrade@outlook.com, FCT/UNESP
Maurício Araújo Dias, madias@fct.unesp.br, FCT/UNESP
Raphael Garcia, raphael@fct.unesp.br, FCT/UNESP
Paula Tavares Pinto, paula@ibilce.unesp.br, IBILCE/UNESP
Andreia Cristiane Silva Wiezzel, andreia@fct.unesp.br, FCT/UNESP

Resumo: A robótica pode ser aplicada ao ensino de variados temas em diferentes áreas do conhecimento, despertando muito interesse entre os estudantes. Considerando essa afirmação, neste artigo propõe-se uma maneira de melhorar qualitativamente a aprendizagem de estudantes em relação aos conceitos da disciplina Sistemas Operacionais. Para tanto, desenvolveu-se um *software* que propicia o uso de um braço robótico. Esse *software* se baseia em conceitos sobre Sistemas Operacionais para realizar toda a movimentação do braço e apresenta dados em interface de simples compreensão. Os resultados obtidos na aplicação do *software* e do braço robótico durante aulas de Sistemas Operacionais mostram que os estudantes tiveram maior facilidade em assimilar o conteúdo ensinado.

Palavras-chave: robótica, sistemas operacionais, ensino, programação.

Using a robotic arm as learning element in Operating Systems classes

Abstract: Robotics can be used to teach different topics in distinct areas of knowledge which has aroused much interest in students. Considering that, this paper presents a practice used to improve the learning of students in relation to the study of concepts in the discipline Operating Systems. In order to do so, we developed a software that provides the use of a robotic arm. This software is based on concepts of Operating Systems to make the arm movement and it also presents the data in a simple display interface. The results, achieved with the application of the software and robotic arm as part of teaching and learning in Operating Systems classes, show that the students assimilated the teaching content in an easier way.

Keywords: robotics, operating systems, teaching, programming.

1. Introdução

A robótica é uma área de estudo amplamente difundida, dentro e fora do meio acadêmico. Pesquisa como as de Lima (2007), Weinberg e Yu (2003), Fiorini (2005), Chiesa *et al.* (2013), Sasahara e Cruz (2007) e Jung (2013), reforçam que a robótica pode ser aplicada em conjunto com outras formas de ensino para aumentar a assimilação de conceitos pelos estudantes. Lima (2007) afirmou que a robótica constitui, ainda, veículo ideal para motivar estudantes, independentemente de estarem no ensino fundamental, médio ou superior. Tal ideia é reforçada por Weinberg e Yu

(2003) que citam o poder atrativo da robótica, tendo em conta o aumento significativo de participantes em uma feira que ocorre em Portugal.

Alguns poderiam citar que trazer robótica para dentro da sala de aula pode ser inviável, devido aos altos custos dos recursos robóticos disponíveis. Isso até poderia ser um problema há duas décadas. Todavia, várias indústrias desenvolvem brinquedos que podem ser utilizados para desenvolvimento da robótica, conforme mostrado por Fiorini (2005), Weinberg e Yu (2003), Sasahara e Cruz (2007) e Jung (2013). Sasahara e Cruz (2007) citam, inclusive, a possibilidade de serem fabricados braços robóticos com sucata, tornando o acesso a tal recurso ainda mais barato.

Algumas escolas, em outros países, começaram a criar clubes de robótica que interagiam com os conteúdos ensinados em algumas disciplinas. Isso colocava em prática todo o conhecimento trabalhado nas aulas teóricas. Lima (2007) e Chiesa *et al.* (2013) registram que em alguns casos, cursos de robótica foram incorporados aos currículos e passaram a fazer parte do quadro de disciplinas oferecidas. Os trabalhos evidenciaram o fato de que todos os estudantes que tiveram contato com a robótica conseguiram assimilar mais facilmente os conteúdos trabalhados, além de se interessarem mais por áreas que inicialmente eram vistas como entediantes, por serem muito teóricas.

Com base nestas considerações, este artigo propõe uma maneira de melhorar qualitativamente a aprendizagem de estudantes em relação aos conceitos da disciplina Sistemas Operacionais, pelo uso de um *software* que faz controle de um braço robótico. Tal *software* tem componentes de um Sistema Operacional: escalonador de processos, gerente de memória e *software* de entrada e saída. Desse modo a disciplina Sistemas Operacionais, que costuma ser teórica, torna-se mais prática, possibilitando maior envolvimento e compreensão por parte dos estudantes.

Para melhor compreensão da proposta apresentada por este trabalho, este artigo apresenta-se organizado em seções, como descrito a seguir. A Seção II apresenta a metodologia empregada em conformidade com a proposta, a Seção III mostra e discute os resultados alcançados por este trabalho e a Seção IV comenta as conclusões.

2. Metodologia

Esta seção apresenta os materiais e o procedimento usados na pesquisa.

2.1 Materiais

Todo o desenvolvimento do *software* foi realizado por meio do Sistema Operacional Microsoft Windows 8.1, e *hardware* Intel Core ® i7 3537U 2,0 GHz, 8GB de memória RAM e 512GB de SSD. A *Integrated Development Environment* (IDE) de programação usada foi o Qt, que é um *framework* multiplataforma para desenvolvimento em linguagem C/C++. O Qt 5.0.1 ou superior é compatível com uma série de Sistemas Operacionais, dentre eles Microsoft Windows, Mac OS X, Linux/X11, Solaris, entre outros. A versão usada no projeto foi a 5.2.0.

Também foi utilizado um braço robótico modelo OWI-535, que vem como um brinquedo desmontado num *kit*, que pode ser usado, por exemplo, para ensinar fundamentos básicos de mecânica, eletrônica e controle do braço. O braço robótico é da categoria de servo-controlados fixo, sendo comandado, em sua forma original, por um *joystick*. Possui mobilidade limitada e conta com 5 motores, ou graus de liberdade. A pinça tem um LED de iluminação e uma abertura máxima de 4,5 cm; o eixo do punho gira 120º; o cotovelo, 300º; o ombro, 180º e a base, 270º.

Para realizar a comunicação com a placa e enviar os sinais de movimentação ao braço, utilizou-se a biblioteca libusb-wind32 que funciona por meio da porta USB e possui compatibilidade com Sistemas Operacionais Microsoft Windows. A biblioteca permite que aplicações do usuário acessem dispositivos USB de modo genérico sem escrever nenhuma linha de código adicional neste processo de comunicação.

2.2 Procedimento

Esta seção faz a apresentação do *software* e do braço robótico.

2.2.1 Janelas do *software*

O *software* possui uma interface principal composta por uma tabela que mostra todos os processos existentes, o estado atual deles (pronto, rodando ou finalizado) e a quantidade de memória que cada processo utiliza, conforme mostrado pela Figura 1. Também, nessa interface, tem-se um grupo de botões responsáveis por ativar ou desativar as filas que o algoritmo de escalonamento usa. Um botão para iniciar/parar o escalonador também encontra-se disponível.

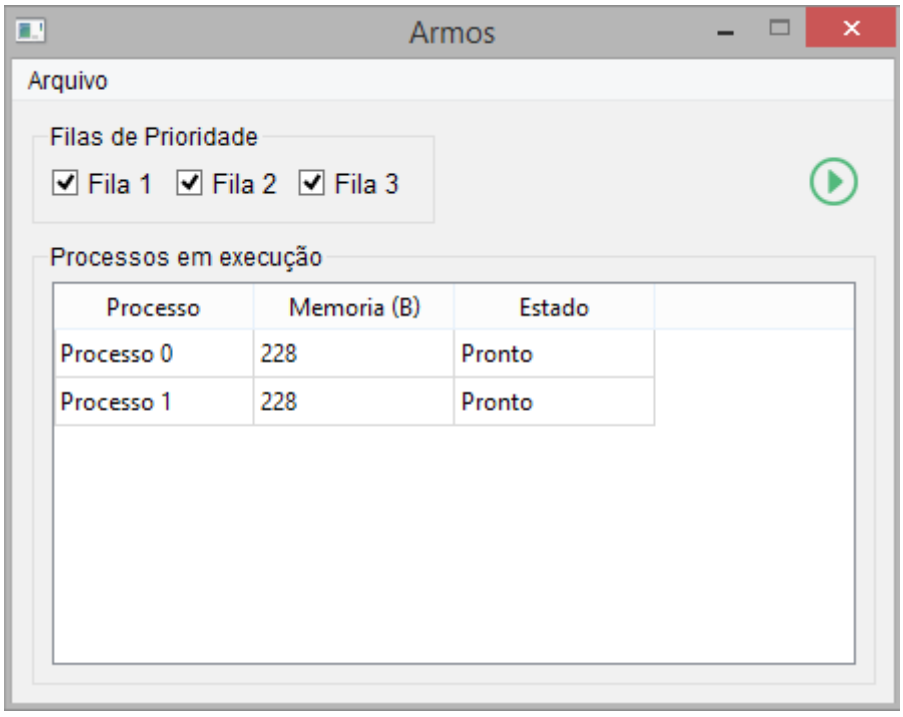

Figura 1 – Janela que representa a interface principal do *software*.

2.2.2 Controlador do braço robótico

O Controlador do braço robótico é responsável por toda a comunicação e execução dos movimentos desejados. Várias estruturas de controle são necessárias para operar o braço. Quando um controlador é criado, são definidos todos os tempos máximos pelos quais cada motor do braço pode funcionar e variáveis para armazenar o tempo pelo qual cada motor do robô já se moveu.

O controlador do braço deve tomar cuidado no momento de realizar os movimentos. Isso porque o braço utilizado é muito simples e frágil e não possui por si mesmo travas que impeçam que os motores sejam forçados. Para contornar tal problema foi desenvolvido um sistema de tempos. Cada motor do braço possui um tempo máximo de movimentação para ambas direções: cima, baixo; direita, esquerda; abrir e fechar. Em uma matriz, armazenam-se os tempos máximos de cada motor para cada direção. Se o tempo desejado para a movimentação do motor for superior aos limites, o movimento não pode ser executado. A ideia do algoritmo é muito simples: ao realizar o movimento para um dos lados, o tempo é subtraído de tal direção e adicionado à oposta.

Além disso, o controlador ainda proporciona a realização dos movimentos. Todos os métodos são muito parecidos em sua essência. Eles verificam se o LED do braço está ou não aceso, definindo assim qual cadeia de bits mandar para o braço. Caso seja verificada uma falha da comunicação com o braço, uma tentativa de reestabelecer a comunicação é realizada. As diferentes chamadas desses métodos, de forma combinada, proporcionam uma grande gama de movimentos, alguns bem elaborados. No final de toda ação, a cadeia de bits limpar é chamada para parar o movimento do braço. A Tabela 1 mostra as respectivas cadeias de bits e os motores que acionam.

Tabela 1- Motores do braço e suas respectivas cadeias de bits

Motor	**Ação**	**LED Acesso**	**LED Apagado**
Garra (Motor 1)	Abrir	0B0000001000100000	0B0000000000100000
	Fechar	0B0000001000010000	0B0000000000010000
Pulso (Motor 2)	Subir	0B0000001000001000	0B0000000000001000
	Descer	0B0000001000000100	0B0000000000000100
Cotovelo (Motor 3)	Subir	0B0000001000000001	0B0000000000000001
	Descer	0B0000001000000010	0B0000000000000010
Ombro (Motor 4)	Subir	0B0001001000000000	0B0001000000000000
	Descer	0B0010001000000000	0B0010000000000000
Base (Motor 5)	Esquerda	0B0000011000000000	0B0000010000000000
	Direita	0B0000101000000000	0B0000100000000000

2.2.3 *Quantum*

O *quantum* é uma medida de tempo em que os processos podem ter suas tarefas executadas no processador. Todos os sistemas operacionais modernos utilizam esse princípio. Neste sistema, a classe Quantum é um contador de tempo, permitindo que os

processos sejam executados. Definiu-se que o quantum seria de 15 segundos. Na prática esse valor é menor, mas foi adotado para facilitar a compreensão por aqueles que utilizam o sistema. Quando o tempo acaba, um sinal é emitido e tratado pelo escalonador.

2.2.4 Gerente de memória

O gerente de memória é responsável por fazer o controle de uso de memória dos processos. Sempre que se deseja alocar uma posição de memória, o gerente fica responsável por verificar as restrições impostas pelo algoritmo de gerência de memória. Neste trabalho, foram usadas partições fixas com tamanho único, devido à simplicidade e facilidade para implementação. Essa técnica consiste na divisão igualitária da memória em um número de partições pré-definidas, sendo assim chamado tamanho único. Essa técnica é dita "partições fixas", pois suas posições não se alteram durante a execução do sistema.

Sempre que uma requisição para alocar memória para determinado processo for feita, o gerente de memória verifica se o montante solicitado excede o valor máximo reservado para aquele processo. Se isso acontecer o processo então é escalonado e não poderá ser mais executado, pois não existe memória suficiente para que ele continue a sua execução. Sendo possível atender ao pedido, a memória é alocada e a quantidade total utilizada pelo processo é atualizada.

2.2.5 Classe principal

A classe principal sustenta todo o *software*, sendo responsável por criar toda a base para o sistema. Nela, todos os métodos para modificação da interface e instanciação dos objetos são definidos. De uma maneira geral, ela coordena a interação do usuário com o sistema, permitindo que todos os objetos necessários para o escalonamento e a gerência de memória sejam criados.

A interação do estudante em ativar ou desativar as filas de prioridades impactam diretamente em como o escalonamento acontece. Conforme a necessidade do escalonador, métodos verificam qual fila de prioridade o estudante deseja manter ativa e qual a fila de prioridade máxima ativada.

Os valores iniciais, ou padrões, são definidos para melhor experimentar o sistema, permitindo ao estudante interagir com o *software* e visualizar o que está acontecendo. Um método específico fica responsável por instanciar as filas de processos, cuja tabela de processos e seus atributos são apresentados ao estudante. O *software* espera pela ação do estudante para iniciar todo o processo de escalonamento e gerência de memória. Um método específico, em conjunto com alguns outros auxiliares, dá início ao escalonamento.

2.2.6 Escalonador

O escalonador de processos é a parte mais importante do *software*. Sem ele, somente um processo seria executado (monoprogramação). Todo processo executado pelo escalonador é representado em uma classe homônima. Uma vez que o escalonador é inicializado, ele sempre ficará verificando se existe algum novo processo adicionado ao sistema e que precisa ser executado. Isso garante continuidade ao sistema.

O papel fundamental do escalonador de processos é garantir uma execução completa e justa de todos os processos, maximizando o tempo em que o processador realiza tarefas, diminuindo o tempo de resposta e o tempo de espera, segundo Silberschatz (2001). Escolheu-se o algoritmo de múltiplas filas com prioridades para implementar o escalonador, porque a ideia é permitir prioridade no sistema.

Conforme o escalonador é chamado para executar os processos, uma série de passos e verificações são realizadas. Primeiramente, o processo atual precisa ser suspenso, mudando seu estado de rodando para pronto. A verificação das filas de prioridade é realizada em seguida. Se existe alguma fila habilitada, o escalonador verifica se ainda existem processos com prioridade mais alta a serem executados. Se não existir e o processo que acabou de ser executado foi finalizado, então a próxima fila de prioridade mais alta é selecionada. Caso ainda existam processos para serem executados, uma nova verificação é realizada para saber se o processo que acabou de ser executado foi finalizado. Se positivo, o processo então é eliminado da fila de processos e um novo processo daquela fila é selecionado. Do contrário, o processo é recolocado no final da fila a que pertence e um novo processo é escolhido para ser executado. No final do processo, uma variável atribuída como verdadeira permite que o processo entre em execução.

2.2.7 Abstração dos processos

Várias características são iguais a todos os processos. Assim não seria viável que todas as classes de processos implementassem os mesmos trechos de códigos. Portanto, deixaram-se tais códigos em uma classe separada que é herdada por todas as outras classes de processos. Tal maneira de trabalhar permite a adição fácil de processos ao sistema, uma vez que a estrutura igual e necessária a todos os processos não precisa ser reimplementada. Assim, pode-se focar no desenvolvimento de novas ações que são realizadas pelo braço robótico nos processos de movimento.

No momento em que um processo é criado, precisa-se atribuir uma série de parâmetros de configuração para ele. Todas as atribuições ocorrem no método construtor da classe. Nele, são definidos o nome do processo, seu estado inicial, o escalonador, variáveis para controle da atomicidade, variáveis para *quantum* do processo, variáveis de controle da execução (que determinam se o processo está finalizado, rodando ou suspenso), o controlador do braço robótico, variáveis para gerência da memória utilizada para cada processo e as conexões necessárias entre classes para que toda comunicação ocorra.

Assim como outras partes do *software*, essa classe é uma *thread*, permitindo execução simultânea com a parte da *interface* do *software*. Um dos meios mais importantes que tal classe provê às classes de processo de movimento é o controle delas. Métodos implementados permitem a inicialização, pausa, finalização ou reinicialização do processo. Utilizando duas classes que o *framework* Qt possui, a *QMutex* e a *QWaitCondition*, pode-se impor condições que devem ser respeitadas. Tais condições trabalham como um semáforo, permitindo ou não que os processos sejam executados no ambiente Windows. Quando o *quantum* do processo expira, o escalonador usa tais métodos para pausar o processo que está sendo executado e escalonar outro em seu lugar, inicializando ou reinicializando a tarefa.

Conforme um objeto tem toda sua execução realizada, ele precisa comunicar ao escalonador de processos que não necessita estar mais no processador. Ao usar o método *terminado*, o escalonador irá imediatamente selecionar outro processo para ser executado. Toda ação exercida entre escalonador e processo é necessária para evitar desperdício de tempo, tornando o escalonador mais otimizado.

2.2.8 Processos de movimentação

São dois processos principais de movimentação criados com o *software*. Tais processos são responsáveis por implementar os códigos dos movimentos que o braço robótico executa. Nas devidas classes do *software*, o primeiro processo é responsável por movimentar o braço para cima e para baixo. Esses movimentos possuem um tempo igual a 3,5 segundos cada um. O segundo processo é responsável por movimentar o braço para a esquerda e para a direita, em seu próprio eixo. Novamente, cada um desses movimentos é executado por 3,5 segundos.

Algumas ações dos processos de movimentação do braço não podem ser executadas separadamente, pois isso acarretaria em uma perda do controle do *software* e nenhum movimento então seria executado pelo braço robótico. Esse problema foi contornado adotando a atomicidade, que consiste em trechos de códigos que devem ser executados do início ao fim, sem que haja qualquer interrupção. Quando tais códigos são identificados como atômicos, isso significa para o escalonador de processos que o escalonamento naquele momento não poderá ocorrer. Ao sair de um trecho de atomicidade, o processo automaticamente avisa o escalonador de processos que sua tarefa indivisível foi finalizada e este pode prosseguir com o escalonamento.

3 Resultados e discussões

Foram sujeitos da pesquisa 146 estudantes do curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista (Unesp), campus de Presidente Prudente-SP, entre os anos de 2012 a 2015.

Os estudantes foram divididos em dois grupos: para o primeiro, composto por 82 sujeitos, o ensino de conceitos sobre sistemas operacionais se deu apenas de maneira

teórica. Este grupo estava dividido em duas turmas, respectivamente com 49 e 33 estudantes, correspondendo aos dois primeiros anos do experimento.

Para o segundo grupo, composto por 64 estudantes, o ensino de conceitos sobre sistemas operacionais se deu com o auxílio do *software* apresentado neste artigo e do braço robótico. Este grupo foi dividido em duas turmas, respectivamente com 29 e 35 estudantes, relativo aos dois últimos anos do experimento. O estudo se deu no âmbito da disciplina Sistemas Operacionais I, oferecida no segundo semestre do segundo ano do curso.

A execução do projeto foi composta pelas seguintes etapas: aulas expositivas teóricas para apresentação de conceitos sobre sistemas operacionais (somente para o primeiro grupo); apresentação do *software* e do braço robótico, ambos em funcionamento, em aulas expositivas (somente para o segundo grupo); aplicação sistemática de exercícios baseados na busca por soluções para problemas reais, avaliações teóricas e aplicação de questionário de percepção do estudante. Tais atividades foram utilizadas como instrumentos de coleta de dados, que foram categorizados e comparados entre o primeiro e segundo grupo.

Para o primeiro grupo, as aulas expositivas utilizadas para ensinar Sistemas Operacionais acompanharam método tradicional de ensino, em que os estudantes geralmente são expectadores e memorizam os conceitos da disciplina, frequentemente, de forma irrefletida.

Ao segundo grupo, foram realizadas várias apresentações do *software* e do braço robótico. O *software* desenvolvido foi exibido para explicar os elementos principais da interface e como se daria a interação com ela. Em seguida, foi apresentado, então, o braço robótico utilizado em conjunto com o *software*, assim como seus movimentos básicos e graus de liberdade. Foi esclarecido, também, o que cada processo do *software* executaria. Tais ações eram tarefas básicas, porém concisas, de modo a permitir uma melhor visualização e diferenciação dos processos. Posteriormente, exibiu-se a movimentação do braço robótico, sendo gerenciado pelo *software* em questão.

Durante a movimentação do braço os estudantes verificavam os estados que os processos assumiam, estando prontos para execução ou sendo executados, assim como o escalonamento, pelas diferentes tarefas realizadas. Além disso, o *software* em funcionamento e o braço robótico em movimento permitiram aos estudantes entrar em contato com diversos outros conceitos sobre sistemas operacionais, com os quais o primeiro grupo de estudantes teve contato apenas de maneira teórica.

A aplicação sistemática de exercícios baseados na busca por soluções para problemas reais (Polya, 1961 e 1965) e (Skinner, 1966), relacionados com sistemas operacionais, assim como as demais etapas, foi realizada para ambos os grupos. Tais problemas abordavam variados conceitos sobre sistemas operacionais, por exemplo: definição de processos; diferenciação entre processos do modo usuário e modo núcleo ou *kernel*; os estados que um processo pode assumir, o que representam e sua importância; o que é escalonador de processos e seus objetivos principais; diferentes

tipos de algoritmos de escalonamento e detalhamento da implementação do projeto; definição de entrada e saída e seus tipos, dentre muitos outros conceitos.

Após essas atividades, os estudantes foram submetidos a avaliações, para verificar a aprendizagem dos conceitos. Tais avaliações consistiam de perguntas discursivas sobre os conceitos trabalhados em aula de forma teórica (primeiro grupo) ou a partir da exibição do *software* e do braço robótico (segundo grupo). Para cada questão, os estudantes precisaram interpretar os enunciados dos problemas, analisar as possíveis soluções, definir estratégias para alcançar as soluções mais adequadas e estabelecer um diagnóstico. Todo o processo foi organizado em consonância com Plants (1980) e Mayer (2002).

A partir da aplicação destas avaliações, encontraram-se, em valores aproximados, os seguintes dados: médias para as notas finais iguais a 6,16 para o primeiro grupo e a 6,64 para o segundo; taxas de reprovação iguais a 23% para o primeiro grupo e a 18% para o segundo. Nota-se um aumento de aproximadamente 8% na média das notas finais do grupo que teve contato com o *software* e com o braço robótico em relação ao grupo que não teve. Além disso, também nota-se aproximadamente 5% menos reprovações, quando o ensino envolveu o *software* e o braço robótico.

Como última atividade, um questionário de percepção do estudante, baseado na escala de Likert (1932), foi aplicado aos estudantes do segundo grupo ao final da pesquisa. É importante mencionar que houve um teste piloto antes que o questionário fosse aplicado. A seguir, apresenta-se um exemplo de questão presente no questionário.

O uso do braço robótico, enquanto estudo de caso, facilitou o meu aprendizado.
( ) Concordo fortemente, ( ) concordo, ( ) indeciso, ( ) discordo, ( ) discordo fortemente.
Justifique:

Tal questionário era composto por dez questões, que versavam sobre a influência do uso do braço robótico e da solução dos problemas relacionados a ele na aprendizagem de Sistemas Operacionais. Foi possível verificar que aproximadamente 99% dos participantes do segundo grupo reconhecem que o uso do braço robótico auxilia a aprendizagem para o ensino de Sistemas Operacionais.

4. Conclusões

Neste artigo, propôs-se uma maneira de melhorar a qualidade do processo de ensino e aprendizagem em relação aos conceitos da disciplina Sistemas Operacionais. Para tanto, desenvolveu-se um *software* que propicia o uso de um braço robótico como elemento de aprendizagem em sala de aula.

Os resultados mostraram que o braço facilitou a aprendizagem dos estudantes, indo ao encontro do que afirmam Lima (2007) e Weinberg e Yu (2003) em seus respectivos trabalhos. O fato da robótica, assim como a Computação, ser uma área em

constante expansão, atrai muitos estudantes, permitindo abordar diferentes conteúdos de maneira prática. Assim, a arte do "aprender fazendo" é o diferencial que motiva tal área.

Em trabalhos futuros, sugere-se uma melhor organização das *threads* dos processos, implementar outros algoritmos de escalonamento, permitir a criação de processos dinâmicos pelos estudantes com uma gama de movimentos maior, e aplicar a robótica ao ensino de outras disciplinas, como física e lógica.

Referências Bibliográficas

CHIESA, A. L.; CORBELLINI, E. M.; GONZÁLEZ, T. A.; BURMAN A. A Robotics Club Experience as a complement to Engineering Education. **IEEE Latin America Transactions**, v. 11, p. 585-590, 2013.

FIORINI, P. LEGO Kits in the Lab. **IEEE Robotics & Automation Magazine**, v. 12, p. 5, 2005.

JUNG, S. Experiences in Developing an Experimental Robotics Course Program for Undergraduate Education. **IEEE Transactions on Education**, v. 56, pp. 129-136, 2013.

LIMA, P. U. Robotics Educational Activities in Portugal: A Motivating Experience. **IEEE Robotics & Automation Magazine**, v. 14, p. 16-17, 2007.

LIKERT, R. A Technique for the Measurement of Attitudes. **Archives of Psychology**, v. 140, p. 1-55, 1932.

MAYER, R. E. A Taxonomy for Computer-based Assessment of Problem Solving. **Computers in Human Behavior**, v. 18, n.6, p.623-632, 2002.

PLANTS, H. L.; DEAN, R. K.; SEARS, J. T.; VENABLE, W. A. S. A taxonomy of problem-solving activities and its implications for teaching. In: Lubkin J. L. (Ed.). **The Teaching of Elementary Problem-solving in Engineering and Related Fields**. Washington DC: American Society for Engineering Education (ASEE), 1980 p. 21-34.

POLYA, G. "Mathematical Discovery: On Understanding, Learning and Teaching Problem Solving", Vol. I e II, New York: Wiley, 1961 e 1965.

SASAHARA, L. R.; CRUZ, S. M. S. D. Hajime – Uma nova abordagem em robótica educacional. In: **Anais do XXVII Congresso da SBC**, p. 460-461, 2007.

SILBERSCHATZ, A.; GAGNE, G.; GALVIN, P. B. **Sistemas operacionais**. Rio de Janeiro: Campus, 2001.

SKINNER, B. F. An Operant Analysis of Problem-Solving. In: B. Kleinmuntz (Ed.). **Problem solving: Research, method, and theory**. Nova York: John Wiley and Sons, 1966. p. 133-171.

WEINBERG, J. B.; YU X. Robotics in Education: Low-Cost Platforms for Teaching Integrated Systems. **IEEE Robotics & Automation Magazine**, v. 10, p. 4-6, 2003.