ROBOTECA: USANDO ROBÔS LEGO MINDSTORM EM SALA DE AULA

Leliane Nunes de Barros – leliane@usp.br
Universidade de São Paulo, Instituto de Matemática e Estatística.
Rua do Matão, 1010.
05508-090 – Cidade Universitária - SP
Valquiria Fenelon Pereira – vfenelon@usp.br
Universidade de São Paulo, Escola Politécnica Mecatrônica.
Av. Prof. Mello Morais, 2231
05508-030 – Cidade Universitária - SP

Resumo: *O uso crescente de robôs em aplicações industriais e domésticas, bem como a facilidade de aquisição de robôs educacionais e de pesquisa de baixo custo, tem despertado o interesse de professores e alunos para o uso de robôs nas escolas, tanto no ensino fundamental e médio, como em cursos de graduação. Nesse contexto, o problema de construção e programação de robôs envolve uma variedade de desafios para os estudantes, com muitas oportunidades de aprendizado. Em particular, um projeto de robôs possui vários aspectos do mundo real, envolvendo: (i) a observação do mundo, através da leitura de sensores e (ii) a interação com o ambiente, através de um programa de controle de motores. Assim, robôs podem ser usados para o ensino de programação de computadores, bem como para o ensino dos conceitos básicos de programação de robôs móveis. Este artigo relata a experiência de um curso de introdução à programação de robôs móveis, usando os kits educacionais Lego Mindstorm, para alunos de graduação em Ciência da Computação. As aulas foram baseadas em métodos de solução de problemas com a proposta de um desafio final extraído da Competição Brasileira de Robótica (CBR).*

Palavras-chave: *Robôs na Educação, Robôs Lego Mindstorm, Aprendizado de Programação.*

1. INTRODUÇÃO

O Bloco Programável LEGO® foi desenvolvido nos laboratórios do MIT (*Massachusetts Institute of Technology*) e lançado em 1996. Ele foi idealizado para ser usado como uma ferramenta para o aprendizado construtivista (Papert, et al., 1991) (Ginling, et al., 1995). O kit educacional[1] LEGO® *Mindstorms* é uma plataforma muito utilizada em escolas pelas seguintes razões: (1) por sua robustez; (2) facilidade de manipulação (sem a necessidade do uso de ferramentas); (3) possibilidade de reuso e configuração de acordo com o objeto de estudo e; (4) por seu baixo custo (Gómez-de-Gabriel, et al., 2011; Leitão, et al., 2005; Klassner, 2002).

A robótica na educação tem sido explorada em vários níveis: no ensino fundamental (Silva, et al., 2008); ensino médio (Ginling, et al., 1995; Osborne, et al., 2010; Barak, 2009), e no ensino superior (Gómez-de-Gabriel, et al., 2011) (Leitão, et al., 2005) (Trevisan, et al., 2006). Em todos eles o objetivo comum é motivar o aluno a buscar o conhecimento multidisciplinar para o desenvolvimento de robôs através de três etapas básicas: criação e especificação de um projeto para a engenharia do robô; sua construção física (por exemplo,

[1] http://education.lego.com/ (último acesso em 17 de junho de 2013).

contendo um conjunto de engrenagens, rodas, motores, atuadores e sensores) e sua programação (isto é, a elaboração de um programa de computador usando comandos gráficos simples, ou algumas centenas de linhas de comandos de uma linguagem de programação padrão, como C ou Java). Esse processo de construção permite a integração do conteúdo aprendido em disciplinas teóricas com aplicações práticas (Kloc, et al., 2009).

Em geral, o uso de robôs na educação pode empregar as seguintes abordagens pedagógicas: (1) *problem-based learning* (Riek, 2013), em que o aprendizado ocorre através da resolução de problemas com robôs, propostos pelo professor, sem uma aula instrucional tradicional prévia; (2) *active-learning* (Riek, 2013), em que conceitos novos são aprendidos e imediatamente aplicados ao robô; (3) *competition-based learning* (Gómez-de-Gabriel, et al., 2011), é feita uma competição entre robôs desenvolvidos por equipes de alunos de forma a motivar a construção de robôs com o melhor desempenho possível.

No ensino superior, os kits robóticos tem sido usados para introduzir procedimentos e equipamentos típicos da área de automação da engenharia (elétrica, mecânica ou mecatrônica) (Carvalho, et al., 2008; Gómez-de-Gabriel, et al., 2011; Fagin, 2001; Pereira, et al., 2012; Cruz, et al., 2008). A construção de robôs que se comunicam para realizar tarefas específicas, bem como o controle da navegação e dos manipuladores dos robôs envolvem, naturalmente, conceitos de diferentes disciplinas, a saber: geometria plana e espacial; cinemática e dinâmica; arquiteturas de robôs inteligentes; algoritmos e tratabilidade; programação de computadores; etc. Assim, a robótica também pode ser explorada em cursos de matemática, física, teoria da informação, ciência da computação, dentre outros.

Um robô móvel é um dispositivo mecânico e eletrônico, contendo sensores e motores, alguns deles usados para sua locomoção, que pode funcionar de modo: tele-guiado, programado e autônomo. Através dos três elementos básicos de um robô: sensoriamento, programação e ação, a robótica móvel oferece uma excelente oportunidade de aprendizado para alunos do ensino superior. Alguns exemplos de contexto em que a robótica móvel pode ser empregada em cursos de ciência da computação são:

- alunos iniciantes podem realizar projetos práticos de programação na disciplina de Introdução à Programação de Computadores – primeira disciplina de programação. Por exemplo, a linguagem de programação dos robôs Lego® Mindstorm pode ser a mesma adotada por essas disciplinas (existem compiladores para o NXT para as linguagens C e Java, dentre outras). Uma das vantagens de se usar robôs numa disciplina introdutória de programação é que através da visualização do robô executando uma tarefa num ambiente real, o aluno poderá observar o funcionamento de seus primeiros programas e fazer uma associação direta entre a lógica do programa e o comportamento do robô. Assim, a depuração de *bugs* e erros de lógica de programação, pode se tornar bem mais fácil para um estudante principiante de computação;
- alunos que já possuem conhecimento de programação podem experimentar situações bastante diferentes daquelas tipicamente encontradas em disciplinas tradicionais de computação (introdutórias ou avançadas) em que as entradas dos sistemas desenvolvidos são, em geral, na forma de arquivos de software e o resultado da execução de programas só pode ser observado na tela do computador. Na programação de robôs, as entradas são medidas de sensores e as saídas são comandos de controle dos motores que podem ser observados no comportamento do robô, ao invés da tela do computador. Além disso, a programação de robôs exige lidar com situações do mundo real (como por exemplo, alteração de luz do ambiente, decaimento de baterias, ruídos, tratamento de tempo real, etc.);
- alunos que já possuem conhecimento avançado de programação podem aprender noções da disciplina de Inteligência Artificial, como agentes inteligentes, autônomos e reativos, bem como noções de representação de conhecimento e multi-agentes. Os alunos podem

ainda explorar conceitos da evolução da robótica inteligente, como arquiteturas reativas, hierárquicas e híbridas (com planejamento e aprendizagem de máquina) bem como a navegação e localização com mapa completo ou parcial do ambiente.

Alunos que possuem em seus currículos disciplinas de robótica têm cada vez mais participado de competições organizadas em seus municípios e estados. A *Competição Brasileira de Robótica* (CBR)[2] foi criada para incentivar a construção do raciocínio científico e a produção de novos conhecimentos na área da robótica através de diversos desafios tais como, futebol de robô, busca e resgate, programação de humanoides, entre outros. Na modalidade da CBR para estudantes de graduação, a *IEEE Standard Educational Kits* (SEK)[3], os alunos devem desenvolver um projeto empregando somente kits educacionais padronizados (VEX[4] ou LEGO®), em geral, envolvendo a cooperação entre dois robôs.

Nesse artigo, relatamos uma experiência realizada no curso de Ciência de Computação do Instituto de Matemática e Estatísticas da USP, com a disciplina chamada de *Introdução à Programação de Robôs Móveis*, que oferece aos alunos da graduação a oportunidade de estudarem conceitos básicos da robótica móvel utilizando os kits educacionais Lego *Mindstorms*. A disciplina é dividida em três módulos. No **Módulo I** são apresentados os componentes do kit robótico, as primeiras instruções de seu manuseio e operação. No **Módulo II** são apresentados os conceitos básicos de controle de *malha-aberta* e *malha-fechada*, bem como as ideias básicas do controle *proporcional-integral-derivativo* (controle PID) da teoria de controle da engenharia, propondo desafios práticos, como por exemplo, "*movimentar o robô mantendo-o a uma distância X da parede"*. No **Módulo III**, todos os conceitos anteriormente aprendidos são usados para a solução de um desafio de maior dificuldade, tanto na construção física do robô como em sua programação. Esse desafio é selecionado a partir da modalidade SEK da Competição Brasileira de Robótica. Essa disciplina faz parte do projeto ROBOTECA do Instituto de Matemática e Estatística da USP, um dos projetos selecionados do programa INOVALAB da Pró-Reitoria de graduação da USP para inovação de material didático e ensino.

Esse artigo está organizado da seguinte forma, na Seção 2, fazemos uma breve descrição do kit educacional Lego *Mindstorm* e do ambiente de programação selecionado (*Brickx Command Center*). Na Seção 3 são descritos os problemas propostos nos três módulos da disciplina. Na Seção 4 fazemos as considerações finais e conclusões.

2. O KIT EDUCACIONAL LEGO MINDSTORM NXT E NXC

A diferença do kit educacional LEGO (série 9797) dos kits vendidos em lojas de brinquedos é que ele vem com uma caixa para armazenamento e organização das peças e possui alguns sensores adicionais. Ele contém 431 peças incluindo o bloco NXT (Figura 1(a)), três servo-motores, sensor ultrassônico (ou sonar), sensor de som, sensor de luz, dois sensores de toque, bateria recarregável, cabos para conexão e um guia com instrução de construção de um primeiro robô (LEGO Mindstorms Education, 2006). O bloco LEGO® NXT contém um micro controlador ARM de 32 bit, display de LCD, quatro portas de entradas para sensores e três portas de saída para atuadores. A comunicação com um computador para monitorar e ou baixar programas no bloco NXT pode ser feita via cabo USB ou via Bluetooth. Também é possível a comunicação entre blocos via Bluetooth.

[2] http://www.cbrobotica.org/ - último acesso em 17 de junho de 2013
[3] http://www.cbrobotica.org/sek.htm - último acesso em 17 de junho de 2013
[4] http://www.vexrobotics.com/ último acesso em 17 de junho de 2013

O kit educacional LEGO já vem com um software de programação gráfica, NXT-G. Porém, com uma alteração em seu *firmaware*, o bloco NXT também pode ser programado usando linguagens de programação Java e C, dentre outras. NXC (*Not eXactly C)* é uma linguagem criada (por John Hansen) especialmente para os robôs Lego e possui comandos que facilitam a execução de tarefas paralelas e concorrentes, permitindo o desenvolvimento de programas de controle sofisticados (Benedettelli, 2007). Os programas NXC são escritos e compilados no ambiente de desenvolvimento *Bricx Command Center* (BrickxCC).

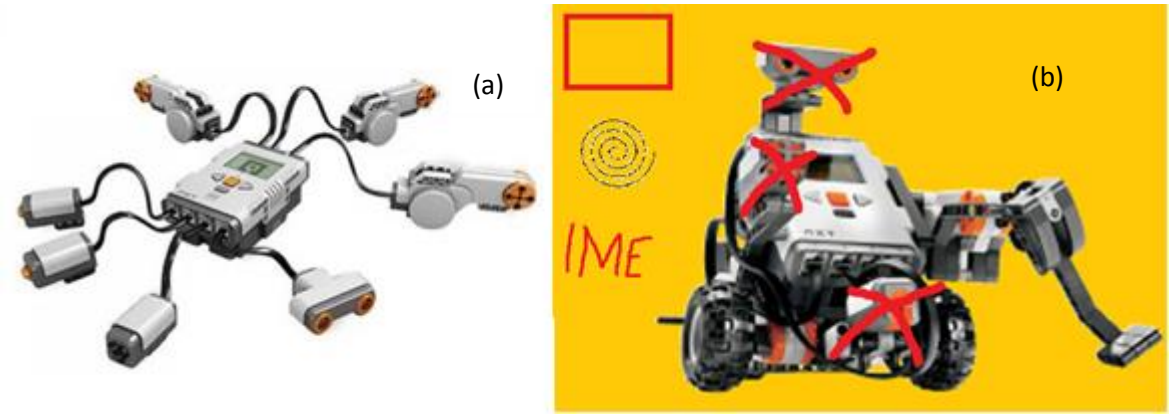


Figura 1 : (a) Bloco LEGO Mindstorms NXT com sensores e atuadores; (b) Robô_Base usado nas três tarefas de controle de malha aberta: *andar retângulo, andar em espiral* e *escrever a palavra IME*.

3. *INTRODUÇÃO À PROGRAMAÇÃO DE ROBÔS MÓVEIS*

A seguir, vamos descrever o conteúdo de cada módulo da disciplina Introdução à Programação de Robôs Móveis, incluindo a construção do robô utilizado e os problemas propostos.

3.1. Módulo I - Apresentação dos componentes do kit Lego Mindstorm.

Do ponto de vista da inteligência artificial um robô é definido como "*uma entidade que percebe o ambiente e atua sobre ele, usando um programa que escolhe uma ação para cada possível sequencia de percepções".* Chamamos esse programa de *"**programa do agente**"* (ou programa do robô).

Após uma rápida apresentação dos componentes do kit e do ambiente de programação BrickxCC, os alunos construíram o primeiro robô, descrito passo-a-passo no guia de construção que acompanha o kit (LEGO Mindstorms Education, 2006), que chamamos de Robô-Base contendo: duas rodas ligadas a um mesmo eixo, cada roda controlada por um motor distinto (isto é, *direção diferencial*); um terceiro motor para controlar um braço (lançador de bolas); um sensor de toque, um sonar e um sensor de luz. A Tabela 1 mostra o conjunto de programas extraídos do guia de programação NXT (Benedettelli, 2007) dado aos alunos durante o Módulo I, para teste de cada um dos elementos do Robô-Base.

O primeiro programa (Programa 1 da Tabela 1), após ser compilado, carregado no robô e executado, serviu para ilustrar o uso dos motores das rodas imprimindo ao robô o seguinte comportamento*: virar para direita, virar para a esquerda, andar para frente, andar para trás e parar*. Após a observação desse comportamento e o programa dado, foram propostos os seguintes problemas para os alunos resolverem:

> ***Problema 1:*** *Modificar o Programa 1 fazendo com que o robô percorra um quadrado de aproximadamente 50 cm de lado, parando ao alcançar a posição inicial (Programa 2).*

Problema 2: *Modificar o Programa 2 fazendo com que o robô ande numa trajetória espiral.*

Tabela 1: Programas para introdução do NXC e uso de sensores (Benedettelli, 2007).

Programa 1: Motores	**task main() {**
	OnFwd(OUT_B,80); OnFwd(OUT_C,0);
	Wait(3000);
	OnFwd(OUT_B,0); OnFwd(OUT_C,80);
	Wait(3000);
	OnFwd(OUT_BC,50); Wait(3000);
	OnRev(OUT_BC,50); Wait(3000);
	Off(OUT_BC); }
		Sensor de Toque:.	task main(){
			SetSensor(IN_1,SENSOR_TOUCH);
			OnFwd(OUT_AC, 75);
			while (true) {
			if (SENSOR_1 == 1)
			{
			OnRev(OUT_AC, 75); Wait(300);
			OnFwd(OUT_A, 75); Wait(300);
			OnFwd(OUT_AC, 75);} } }

Sensor de Luz:	**#define THRESHOLD 40**
	task main(){
	SetSensorLight(IN_3);
	OnFwd(OUT_AC, 75);
	while (true) {
	if (Sensor(IN_2) > THRESHOLD) {
	OnRev(OUT_C, 75); Wait(100);
	until(Sensor(IN_2) <= THRESHOLD);
	OnFwd(OUT_AC, 75);} } }
		Sensor de Som:	**#define THRESHOLD 40**
			#define MIC SENSOR_2
			task main(){
			SetSensorSound(IN_2);
			while(true){
			until(MIC < THRESHOLD);
			OnFwd(OUT_AC, 75); Wait(300);
			until(MIC > THRESHOLD);
			Off(OUT_AC); Wait(300); }}

Sensor Ultrassônico:	#**define NEAR 15 //cm** **task main(){** **SetSensorLowspeed(IN_4);** while(true){ OnFwd(OUT_AC,50); **while(SensorUS(IN_4)>NEAR);** Off(OUT_AC); OnRev(OUT_C,100); Wait(800); }}	Programas com subrotinas e funções:	**sub turn_around(int pwr) {** OnRev(OUT_C, pwr); Wait(900); OnFwd(OUT_AC, pwr);**}** **inline void turn_time(int pwr, int turntime){** OnRev(OUT_C, pwr); Wait(turntime); OnFwd(OUT_AC, pwr);**}** task main(){… **turn_around(75);** **turn_time(75, 2000);**…}

Programa NXC com multi-tarefas	task music() {
	while (true) {
	PlayTone(TONE_A4, MS_500);
	Wait(MS_600);}}
	
	task movement(){
	while (true) {
	OnFwd(OUT_A, Random(100));
	Wait(Random(SEC_1));}}
	
	task controller() {
	Wait(SEC_10); stop music;
	Wait(SEC_5); StopAllTasks(); }
	
	task main() {
	Precedes(music, movement, controller);}
		Programa NXC com concorrência (usando o sensor de toque):	**mutex moveMutex;**
			
			task move_square(){
			while (true){
			Acquire(moveMutex);
			OnFwd(OUT_AC, 75); Wait(1000);
			OnRev(OUT_C, 75); Wait(500);
			Release(moveMutex);} }
			
			task check_sensors(){
			while (true){
			if (SENSOR_1 == 1){
			Acquire(moveMutex);
			OnRev(OUT_AC, 75); Wait(500);
			OnFwd(OUT_A, 75); Wait(500);
			Release(moveMutex);} } }
			
			task main() {
			Precedes(move_square, check_sensors);
			SetSensorTouch(IN_1);}

Através da solução dos Problemas 1 e 2, os alunos puderam aprender a usar os comandos básicos de controle dos motores, numa direção diferencial, definindo os tempos (comando *Wait*) e movimentação (comandos *OnFwd* e *OnRev*) de cada motor (saídas B e C do bloco NXT). Note que para percorrer a distância aproximada de 50 cm, os alunos adotaram uma estratégia de tentativa e erro. Isso ficou mais evidente com a proposta do Problema 3.

Problema 3 (Robô escritor): *Modifique o Robô-Base e construa um programa que faça com que o robô "segure" uma caneta e escreva a palavra "IME" em uma folha de papel, utilizando o terceiro motor para levantar e abaixar a caneta.*

Enquanto os alunos realizaram facilmente a tarefa de programar para uma trajetória quadrada ou espiral, o Problema 3 serviu para mostrar que inicializar e finalizar trajetórias com precisão (para formar a palavra IME) é uma tarefa difícil (isto é, movimentar o robô fazendo com que a ponta da caneta, localizada na lateral do robô, percorra uma dada trajetória). Os três primeiros problemas propostos foram exemplos de problemas de *malha-aberta*, pois não fizeram uso da percepção do ambiente. Isto é, o Robô-Base foi usado sem a leitura dos sensores, como mostra a Figura 1(b).

Em seguida, os alunos passaram a usar os sensores para resolver problemas de controle do tipo *malha-fechada* na construção de robôs que usam suas percepções para observarem o resultado de suas ações e atuarem ainda melhor (e não dependerem tanto do seu programador para definir com precisão os tempos e potências dos motores). Para isso, foram apresentados, de forma breve, cada um dos sensores disponíveis no kit, em termos de como eles coletam dados do ambiente e o tipo de informação que eles fornecem ao programa. Os comandos de leitura de cada sensor foram introduzidos através dos programas sobre sensores da Tabela 1, dados aos alunos para serem compilados e executados.

Para depurar os programas, foi mostrado aos alunos como escrever na tela do bloco os valores lidos pelos sensores. Isso serviu, por exemplo, na descoberta de quais são os valores que definem um intervalo de cor desejado para o sensor de cor e qual é a máxima e mínima distancia que o sensor de ultrassom lê. A informação sensorial foi necessária na solução dos dois problemas descritos a seguir.

Problema 4 (Robô que evita obstáculos) *Modifique e programe o seu Robô-Base para desviar de obstáculos estáticos de forma eficiente, utilizando os sensores de toque e ultrassônico. Faça o robô percorrer a arena, procurando explorar o máximo da área, durante o tempo de 2 minutos.*

Problema 5 *Coloque o seu robô, junto com os robôs das outras equipes da classe para evitar obstáculos estáticos e dinâmicos. Seu robô deve evitar choques, percorrendo a arena com o máximo de varredura (Figura 2).*

A linguagem NXC tem uma estrutura que permite explorar os conceitos de execução de várias tarefas (multitarefas) paralelas e concorrentes (com o uso de semáforos). O conceito multitarefa pode ser entendido como ações realizadas simultaneamente, como por exemplo, *mover para frente enquanto faz a leitura de um sensor.* A ideia de semáforo é evitar, por exemplo, o uso simultâneo de um mesmo recurso, o que pode gerar comportamentos indesejados no robô. A programação com semáforos (uma variável especialmente criada pelo programador e usada de maneira especial (vide manual do NXC (Benedettelli, 2007)) funciona como um indicador de que um recurso, por exemplo, o motor, está sendo utilizado por outra tarefa e que será liberado no final. A solução do Problema 4 e do Problema 5 envolve esse conceito: *para explorar a arena enquanto evita obstáculos, os motores de controle das rodas devem ser vistos como um recurso a ser compartilhado*.

Outra dificuldade que os alunos devem ter que enfrentar nessa disciplina é a navegação orientada, questão crucial da robótica móvel, que pode ser feita seguindo marcações no chão, como uma linha a ser seguida (Figura 2, direita) ou uma arena com linhas perpendiculares definindo uma grade ou matriz (Figura 2, esquerda). Nesse último caso, o robô pode manter certa noção de localização, se movimentando de célula em célula da matriz.

Problema 6 (Robô seguidor de linha) *Modifique e programe o seu robô para seguir a linha de um circuito oval, em qualquer sentido, completando o circuito no menor tempo possível.*

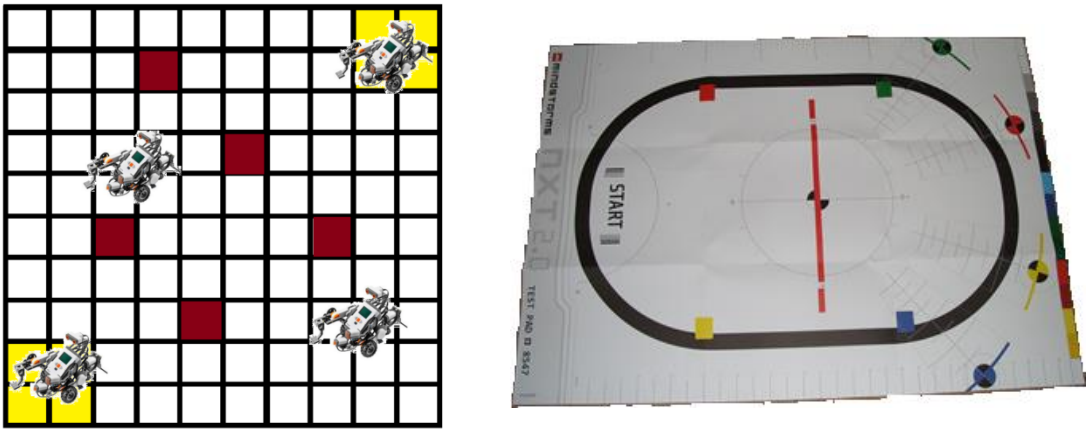


Figura 2: Arena utilizada no Problema 4 e no Problema 5 (*robô que evita obstáculos e outros robôs móveis*) e pista oval utilizada no Problema 6 (*robô seguidor de linha*).

Problema 7 (Segue parede numa sala fechada) *Seguir parede a uma distância X e sempre que perceber um obstáculo a frente, girar 90º e continuar seguindo parede.*

Para resolver o Problema 6, os alunos tiveram que decidir qual sensor usar para seguir a linha e como programar o robô para retornar ao circuito cada vez que ocorresse uma percepção indicando que o robô saiu da linha. Para resolver o Problema 7, os alunos também tiveram que decidir quais sensores utilizar para manter o robô na distancia correta da parede. Como eles utilizaram as paredes da arena (simulando uma sala fechada), o percorrimento da trajetória quadrada da arena, exigiu o uso de mais um sensor para detectar fim de parede e curva de 90 graus. Aqui também houve a necessidade do uso de semáforos: os motores devem ser compartilhados para: (i) manter a distância da parede e (ii) fazer a curva de 90 graus.
Ao final desse módulo, os alunos aprenderam: a programar uma direção diferencial, a calibrar os valores dos sensores; escolher qual sensor pode ser mais apropriado para cada tarefa; qual a melhor posição do sensor no robô; quais fatores podem influenciar na leitura dos sensores; e como tudo isso tudo afeta o comportamento do robô. Em robótica é comum que a leitura do sensor tenha algum ruído e o desafio para esses alunos está em descobrir porque o programa, aparentemente perfeito, não funciona, ou seja, o robô apresenta um comportamento muito diferente do esperado.

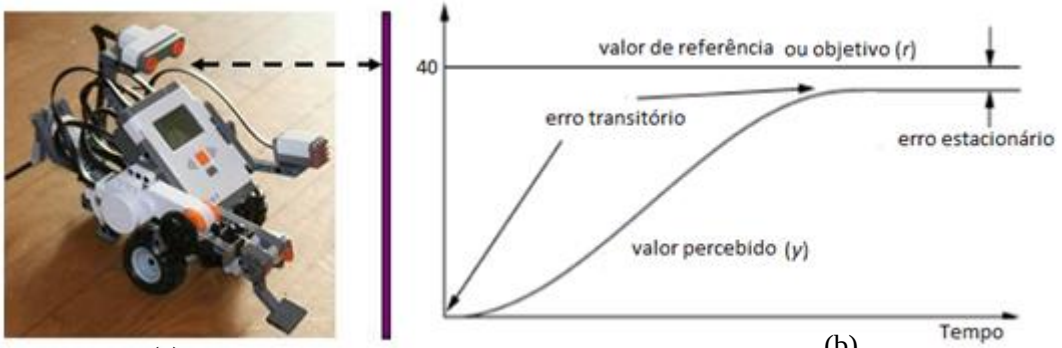
3.2 Módulo II – Conceito de controle PID.

No Módulo I, os conceitos básicos de controle de *malha-aberta* e *malha-fechada* foram explorados na solução dos Problemas 1-7. Em especial, os Problemas 4-7 fizeram com que os alunos sentissem a grande dificuldade de usar informações sensoriais para o controle do comportamento do robô (por exemplo, manter o robô a uma distância da parede ou seguir uma linha). Com isso, a robótica móvel mostra a necessidade de se aprender técnicas conhecidas de teoria de controle, como o controle *proporcional-integral-derivativo* (PID), bastante utilizado em soluções de automação (Nise, 2012). O controle PID pode ser usado para calcular quais os valores de potência devem ser enviados aos motores para que o robô móvel atinja seu objetivo com o menor erro, rapidez e sem oscilações. Considere o problema a seguir:

Problema 8 *– Faça o robô se movimentar perpendicularmente a uma parede e parar a 40 cm de distancia da parede (*Figura 3*(a)). Utilize o robô básico do manual.*

A curva da Figura 3(b) foi mostrada aos alunos para indicar a variação dos valores

percebidos pelo robô da distância do robô até a parede, medida através do sensor ultrassônico e introduzir os termos básicos da teoria de controle. O objetivo do controle é fazer com que o *valor percebido* se aproxime do *valor de referência* r = 40 cm. O *erro transitório* é a faixa de valores entre o inicio e a estabilização do sistema. O *erro estacionário* corresponde a um erro residual que persiste após o sistema estabilizar (Nise, 2012).



(a) (b)

Figura 3 (a) Uso do sensor ultrassônico para parar o Robô-Base a uma distância fixa da parede; (b) Ilustração das respostas do sistema de controle (modificado de (Nise, 2012)).

Tabela 2 Exemplo de códigos em NXC utilizados para resolver o Problema 8.

Controle bang-bang	`while(true) {`
	`  y = SensorUS(IN_4);`
	`  if( r < y){`
	`  OnFwd(OUT_AC,u);}`
	`  if( r > y){`
	`  OnRev(OUT_AC,u);}`
	`  if(r == y){`
	`  Coast(OUT_AC);}}}`
		Controle proporcional	`while(true) {`
			`  y = SensorUS(IN_4);`
			`  e = (y - r);`
			`  u = kp * e;`
			`  OnFwd(OUT_AC,u);`
			`  }`

Controle integrativo	`while(true) {…`
	`  // soma dos erros`
	`  E = E + e;`
	`  u = ki * E;`
	`  OnFwd(OUT_AC,u);`
	`  }`
		Controle PID	`while(true) { …`
			`  e = (y - r); // proporcional`
			`  E = E + e; // integral`
			`  vve = e - eant; //derivativo`
			`  u = (kp * e)+(ki * E)+(kd * vve);`
			`  eant = e;  OnFwd(OUT_AC,u);  }`

No controle PID, trabalhamos com valores de erro, ou seja, com a diferença entre um valor de referencia, ou objetivo (por exemplo, parar a 40 cm de distância da parede), e o valor de fato percebido pelo sensor (por exemplo, o sonar). Na aula foram apresentados sete tipos de controladores de malha fechada para resolver o Problema 8, a saber: *bang-bang*, proporcional, integral, proporcional-integral, derivativo, proporcional-derivativo e o proporcional-integral-derivativo. A Tabela 2 resume quatro tipos de controle.

Chamaremos de *u* o valor da potência aplicada pelo controlador, *r* o valor de referência ou objetivo, *y* o valor lido pelo sensor. O erro *e* é dado por (*y* - *r*), ou seja, o quanto falta para atingir a distancia desejada.

$$u = \begin{cases} u_{max}\ if\ \ e < 0 \\ -u_{max}\ if\ \ e > 0 \\ 0 \quad\ if\ e = 0 \end{cases} \tag{1}$$

$$u(t) = k_i \sum e(t) \tag{2}$$

$$vve = e_t - e_{t-1} \tag{3}$$

O controle ***bang-bang*** é um controle simples (Tabela 2), dado pela Equação (1), apesar de muito simples, esse tipo de controlador oscila na saída porque tem a mesma reação

(mesmo valor de u), para qualquer que seja o erro, positivo ou negativo, seja ele pequeno ou grande.

O ***controle proporcional*** usa, obviamente, um valor de controle *u* proporcional ao erro *e*, ou seja, $u = k_p e$. Assim, quanto menor o erro, menor será o valor de *u*, convergindo para um *erro estacionário* (erros muito pequenos continuam a receber uma resposta, ainda que pequena). Para um melhor ajuste da proporcionalidade entre o erro *e* a resposta do controle, usamos um fator de ganho $\boldsymbol{k_p}$. Se o fator de ganho for alto pode gerar oscilações e *overshoot*, se for baixo a convergência para o objetivo é mais lenta. O programa na Tabela 2 ilustra a implementação do controle proporcional na linguagem NXC.

Em seguida, mostramos aos alunos que o ***controle integrativo*** pode eliminar o erro de convergência estacionária de um controle puramente proporcional, computando uma saída *u* do controlador *proporcional à soma dos erros medidos ao longo de um intervalo de tempo* (sequencia de percepções). O controle integral também considera um fator $\boldsymbol{k_i}$ (como constante de ganho) e é dado pela Equação (2), sendo t o instante de tempo em que é feita cada medida do sensor (determinado pelo tempo utilizado por uma iteração do programa do robô). O programa na Tabela 2 ilustra a implementação do controle integral na linguagem NXC.

Como se sabe, isoladamente o controle integrativo não é uma técnica de controle que deve ser utilizada. Ela deve ser usada com a parcela proporcional, porém quando a parcela integrativa é muito atuante, isso pode gerar instabilidade no sistema. Assim, introduzimos o ***controle derivativo***, que melhora ainda mais o controle proporcional-integral pois adiciona uma parcela proporcional à velocidade de variação do erro (*vve)* Equação (3).

O objetivo da parcela derivativa é atuar, tanto no início do controle como no tratamento de possíveis distúrbios. Também aqui, a intensidade de ação dessa parcela é proporcional ao valor de um fator de ganho k_d.

Finalmente, o controle PID resulta da união da três técnicas: o controle básico com a parcela proporcional (P); com diminuição do erro estacionário pela parcela integrativa (I) e com a redução de oscilação pela parcela derivativa (D). O programa na Tabela 2 mostra o código NXC que implementa um controle PID completo. Finalmente, com ajustes adequados dos fatores de ganho (k_p, k_i *e* k_d) os alunos resolveram o Problema 8 com controle PID.

Como último desafio desse módulo, para sedimentar os conceitos do controle PID, foi pedido aos alunos para que eles resolvessem o Problema 9.

> ***Problema 9 (Robô seguidor de parede com controle PID)*** *- Considere o mesmo desafio do Problema 4, porém aplicando o controle PID.*

Para resolver o Problema 9 os alunos precisaram adaptar os programas da Tabela 2 para fazer o controle da direção diferencial (isto é, u deve ser decomposto em potências para os dois motores). Com esse desafio, os alunos puderam comparar o controle PID com a solução que eles implementaram para o Problema 7. Todos os alunos tiveram ainda a oportunidade de sentir a dificuldade que um engenheiro de controle enfrenta para ajustar os três parâmetros de ganho (k_p, k_i *e* k_d) de acordo com as condições do ambiente e características do robô.

3.2. Modulo III – Desafio Livre.

Nesse módulo do curso, foi proposto um desafio livre, que envolve a construção de um robô específico e sua programação. O desafio adotado foi extraído da modalidade SEK da Competição Brasileira de Robótica de 2012 e 2013.

> ***O jogo THBall.***[5] *Esse jogo é realizado em uma arena dividida por uma zona morta de 10 cm (vala formada por duas paredes de 10 cm (Figura 4)). Dois times adversários, com dois robôs cada, ficam em lados opostos da arena, com 8 bolas laranjas e 4*

[5] http://www.cbrobotica.org/regras/SEK2013_v1_0_portugues.pdf (último acesso em 17 de junho de 2013)

bolas azuis, inicialmente dispostas em cada lado da arena, como mostrado na Figura 4. Cada partida tem duração de 5 minutos. O objetivo do jogo é manter em seu campo o maior número de bolas azuis e o menor número de bolas laranja. As bolas laranja devem ser lançadas para o campo adversário, passando por cima da zona morta.

Antes de resolver o problema completo foram propostos dois novos problemas para os alunos resolverem, baseadas no desafio. A ideia é a de dividir a dificuldade do problema original em tarefas menores, possíveis de serem resolvidas pelos alunos com base nas soluções dos Problemas 1-8.

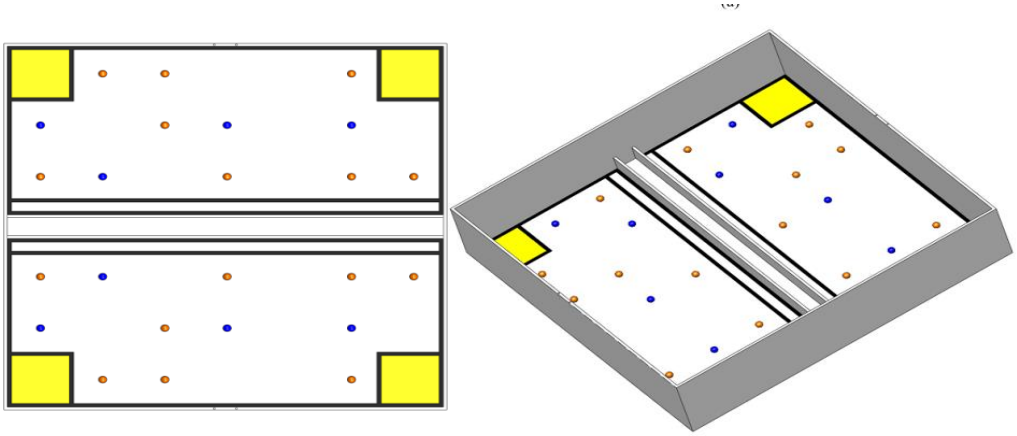


Figura 4. Vista superior e vista 3D da arena THBall da competição SEK 2013.

Problema 9 *– Programe o Robô-Base para percorrer a arena para chutar bolas azuis para a direita e bolas vermelhas para a esquerda. Considere que as bolas estão dispostas como no THBall.*

Problema 10 *– Construir e programar um dispositivo, fixo ou móvel, capaz de lançar uma bola para o lado oposto da arena.*

A ideia foi fazer com que os alunos, através da experiência adquirida com os Problemas 9 e 10, estivessem aptos para resolver o THBall. A solução do Problema 10 foi apresentada por 12 equipes de 2 alunos. Assim foi possível escolher *o melhor atirar de bolas* entre as 12 propostas, para ser copiado (clonado) pelas demais equipes. Finalmente, foi feita uma competição entre 5 equipes de 4 a 5 participantes. Durante as semanas que antecederam a competição, as 5 equipes trabalharam horas seguidas nas tardes de aula, sem interrupções até mesmo para um intervalo, o que demonstrou a motivação dos alunos. Um resultado interessante foi observar que todas as equipes participaram com um nível de desempenho semelhante e com um espírito bastante competitivo.

4. CONSIDERAÇÕES FINAIS

Ao observar um comportamento indesejado do robô, um aluno considerar as seguintes situações: a existência de um erro no programa do robô, devido a erro na leitura dos sensores ou no valor esperado de leitura, a bateria do robô pode estar fraca, ou ainda, a construção física do robô está incorreta. Todas essas situações são novas para alunos da computação, incluindo aqueles com conhecimento avançado de programação de computadores. Além disso, o aluno deve, muitas vezes, ter que trabalhar com recursos escassos, isto é, com um número reduzido de sensores e motores. Assim, pode-se dizer que os kits robóticos permitem aulas na forma de verdadeiros *laboratórios de engenharia e de programação*. Na ciência da computação e cursos afins (isto é, engenharia de computação, automação e mecatrônica), esse tipo de disciplina tem o potencial de trabalhar como complemento de outras matérias de programação avançada, desenvolvendo projetos que solidifiquem os conceitos, bem como a criatividade, o trabalho em grupo com limites de prazos e limite de recursos.

REFERÊNCIAS BIBLIOGRÁFICAS

Barak, Moshe and Zadok, Yair. 2009. *Robotics projects and learning concepts in science, technology and problem solving.* s.l. : Springer Netherlands, 2009. pp. 209-307. Vol. 19.

Benedettelli, Daniele. 2007. *Programming LEGO NXT Robots using NXC.* 2007.

Carvalho, José A. D. e Garcia, Marcus V. R. 2008. *Metodologias para o ensino de robótica industrial: uma abordagem prática baseada em aprendizagem vivencial.* Juiz de Fora : s.n., 2008.

Cruz, Marcia Elema Jochims Kniphoff da, Haetinger, Werner e Horn, Fabiano. 2008. *Desenvolvimento e Comercialização de Kit de Robótica Educativa, Através de Parceria Universidade-Empresa.* Rio de Janeira : s.n., 2008.

Fagin, .B, Merkle, L., e Eggers, T. 2001. *Teaching basic computer science concepts with robotics.* 2001.

Ginling, J., et al. 1995. *LEGOsheets: a rule-based programming, simulation and manipulation environment for the LEGO Programmable Brick.* Washington, DC, USA : IEEE Computer Society, 1995.

Gómez-de-Gabriel, Jesús M., et al. 2011. *Using LEGO NXT Mobile Robots With LabVIEW for Undergraduate Courses on Mechatronics.* 2011. pp. 41-47. Vol. 54.

Klassner, Frank. 2002. A case Study of LEGO Mindstorms Suitability for Artificial Intelligence and Robotic Courses at the College Level. 2002.

Kloc, Antonio Eduardo, Koscianski, André e Pilatti, Luiz Alberto. 2009. *Robótica: uma ferramenta pedagógica no campo da computação.* 2009. pp. 1394-1403.

LEGO Mindstorms Education. 2006. *NXT User Guide.* s.l. : www.mindstormseducation.com, 2006.

Leitão, Paulo, Gonçalves, José e Barbosa, José. 2005. *Learning mobile robotics using lego mindstorms.* Marbella, Spain : s.n., 2005.

Masetto, Marcos Tarciso. 2003. *Competência Pedagógica do Professor Universitário.* São Paulo : Summus, 2003.

Nise, Norma S. 2012. *Engenharia de Sistemas de Controle.* 6ª. s.l. : LTC, 2012.

Osborne, Brook R., Thomas, Antony J. e Forbes, Jeffrey R.N. 2010. Teaching with robots: a service-learning approach to mentor training. *Proceedings of the 41st ACM technical symposium on Computer science education.* 2010, Vol. SIGCSE '10, pp. 172--176.

Papert, Seymour e Harel, Idit. 1991. Situation Contructionism . *Constructionism.* s.l. : Ablex Publishing Corporation, 1991.

Pereira, Valquiria Fenelon e Santos, Paulo Eduardo. 2012. *INICIAÇÃO A ENGENHARIA DE AUTOMAÇÃO E CONTROLE COM KIT EDUCACIONAL.* Belém : s.n., 2012.

Riek, Laurel D. 2013. *Embodied Computation: An Active-Learning Approach to Mobile Robotics Education.* 2013. pp. 67-72. Vol. 56.

Siegwart, Roland e Nourbakhsh, Illah. 2004. *Introduction to Autonomous Mobile Robots.* M : Massachusetts Institute of Technology, 2004.

Silva, A. F., et al. 2008. *Diagnostic Robotic Agent in the RoboEduc Environment for Educational Robotics.* 2008. pp. 131-136.

Tool for Experimenting With Concepts of Mobile Robotics as Applied to Children's Education. **Jojoa, Edith Milena Jiménez, Bravo, Eduardo Caicedo e Cortés, Eval Bladimir Bacca . 2010.** 1, s.l. : IEEE Transactions on Education, 2010, Vol. 53.

Trevisan, Felipe W., Barros, Leliane N. de e Silva, Flávio S. Corréa da . 2006. *Designing Logic-based Robots.* 2006. pp. 11-22. Vol. 10.

USP - Pró-Reitoria de Graduação. Pró-Inovação no Ensino Prático de Graduação. *http://www.prg.usp.br/?p=1132.* [Online] [Citado em: 20 de junho de 2013.] http://www.prg.usp.br/wp-content/uploads/editalinovalab120920121.pdf.

ROBOTECA: USING LEGO MINDSTORM ROBOTS IN CLASSROOM

***Abstract**: The increasing use of robots in industrial and domestic applications, as well as the access of low cost educational and research robots, has aroused the interest of teachers and students to the use of robots in schools, both in fundamental and medium, as in undergraduate courses. In this context, the problem of building and programming robots involves a variety of challenges for students with many learning opportunities. In particular, the challenges with robots require the treatment of situations in the real world, involving sensors and actuators that can be controlled by computer programs. Robots can be used to teach introductory programming to beginners. For more advanced students, that is, they already have knowledge of programming the robots can be used to teach the basic concepts of mobile robotics and to develop skills in group situations problems in environments not fully controlled (as are software projects with inputs and outputs controlled). This article reports the experience of creating an introduction to programming mobile robots course using the LEGO Mindstorm kits education for undergraduate students in computer science.*

***Keywords**: Learning, Mobile Robotics, Competition.*