Desenvolvimento e implementação de protocolo e das camadas de comunicação para robôs móveis educacionais

Development and implementation of protocol and communication layers for mobile educational robots

Antonio Valerio Netto[1]

RESUMO

O artigo descreve o desenvolvimento de um protocolo de comunicação sem fio para robôs móveis e a implementação das suas camadas de comunicação que incluem a base instalada no computador; o software embarcado no dispositivo rádio-base e o software embarcado no próprio robô móvel. Trata-se de um robô de pequeno porte fabricado no Brasil que possui software e hardware livres. Sua aplicação é para área de edutainment onde desenvolvedores podem criar atividades práticas e didáticas para serem utilizadas no aprendizado de lógica de matemática e linguagem de programação. São apresentados os testes de campo e a validação do sistema.

Palavras-chave: Robótica móvel. Edutainment. Protocolo de comunicação, Software embarcado.

1 Pesquisador em desenvolvimento tecnológico, CNPq | avnetto@hotmail.com

Desenvolvimento e implementação de protocolo e das camadas de comunicação para robôs móveis educacionais

Development and implementation of protocol and communication layers for mobile educational robots

ABSTRACT

The article describes the development of a wireless communication protocol for mobile robots and the implementation of their communication layers that include the installed base in the computer; the software embedded in the base radio device and the software embedded in the mobile robot itself. It is a small robot manufactured in Brazil that has free software and hardware. Its application is for edutainment where developers can create practical and didactic activities to be used in learning math logic and programming language. Field tests and system validation are presented.

Keywords: Mobile robotics. Edutainment. Communication protocol. Embedded software.

1 Introdução

Inicialmente, os robôs foram utilizados para a automação de processos de produção industrial. Com o desenvolvimento e a diminuição dos custos das tecnologias envolvidas, os robôs começaram também a ser utilizados para outros propósitos tais como: robôs sociais, entretenimento, educação, segurança, saúde, logística, além da realização de tarefas em ambientes subaquáticos entre outros. Segundo a Comissão Econômica das Nações Unidas, mais de 40 mil robôs industriais foram instalados nas Américas em 2016 (WORLD ROBOTICS, 2016). Desses, em torno de 1,8 mil estão no Brasil. O México é o segundo maior usuário de robôs na região, seguido pela Argentina. De acordo com a Comissão Econômica, cresce o número de robôs nas indústrias brasileiras. Na avaliação do estudo, para 2019 serão 3,5 mil robôs no Brasil. Os setores, automobilístico e de eletrônicos, são os principais usuários da tecnologia no Brasil. O estudo prêve, em termos de unidades, um plantel mundial de robôs industriais de 2.589.000 unidades para o final de 2019, representando uma taxa de crescimento anual médio de 12% entre 2016 e 2019. Em 2015, existiam 1,8 milhão de unidades.

A robótica é um dos principais campos da tecnologia, ultrapassando as fronteiras da engenharia tradicional. Entender a complexidade dos robôs e suas aplicações requer conhecimentos de engenharia elétrica e mecatrônica, computação, matemática e física (SCIAVICCO & SICILIANO, 1996). Neste contexto, a robótica é uma área de pesquisa eminentemente interdisciplinar. A idéia de máquinas suficientemente complexas para executar tarefas de precisão tipicamente humanas é mais antiga do que a tecnologia que as tornaram possíveis. Filmes de ficção científica da primeira metade do século XX já retratavam robôs como formas de vida artificial, geralmente se diferenciando dos homens apenas pela falta de sentimento e pela aparência. O termo robô foi introduzido pela primeira vez no vocabulário ocidental pelo autor teatral tcheco Karel Capek em sua peça “Os Robôs Universais de Rossum”, sendo a palavra *“robota”*, o termo tcheco para “trabalho”. Desde então, o termo tem sido aplicado a uma grande variedade de dispositivos mecânicos, como dispositivos teleoperados, veículos submarinos, veículos autônomos, etc. Virtualmente, qualquer coisa que opere com algum grau de autonomia, geralmente sobcontrole computacional, tem sido chamada de robô (KOIVO, 1989).

As primeiras aplicações com sucesso de robôs manipuladores geralmente envolviam algum tipo de transferência de material, tais como injeção em moldes ou estampagem, onde o robô meramente auxiliava uma prensa carregando a peça ou transferindo uma peça finalizada. Os primeiros robôs eram capazes de ser programados para executar uma seqüência de movimentos, tais como se mover para um ponto A, fechar a garra, se mover para um ponto B, etc. Contudo, não tinham capacidade sensorial externa. Aplicações mais complexas, tais como solda, pintura e montagem, exigiam não só movimentos mais complexos, mas também algum tipo de sensoriamento externo, como visão, tato ou sensoriamento de força, devido à maior interação do robô com o ambiente. Deve-se destacar que as aplicações dos robôs não estão limitadas às tarefas industriais, existem diversas outras aplicações de robótica em áreas onde a presença humana é indesejável ou impraticável. Dentre essas aplicações estão a exploração submarina e interplanetária, o resgate e reparo de satélites, a desativação de dispositivos explosivos, o trabalho em ambientes radioativos, entre outras (CHINZEI et al., 2000).

Dentro da área de robótica, destacam-se os Robôs Móveis Autônomos (RMA), cuja aplicação é muito diversificada, estendendo-se desde a exploração espacial até a ajuda a idosos e a deficientes, passando pelo manuseio de materiais perigosos em condições arriscadas ou operações militares, dispensando em muitas aplicações a presença do ser humano (SIEGWART et al, 2004) (MARCHI, 2001). Os RMA são aqueles que interagem com o meio ambiente onde estão inseridos por meio de mecanismos sensoriais, sendo capazes de operar autonomamente em um ambiente desconhecido e não necessariamente estruturado. Os robôs que apresentam um “comportamento inteligente” são capazes de interagir com o ambiente, tomar decisões e aprender enquanto realizam suas tarefas

(KRAMER & SCHEUTZ, 2007). No Brasil, uma das aplicações da tecnologia é para o desenvolvimento de times de robôs móveis capazes de jogar futebol com outros robôs. O futebol de robôs, que inicialmente era apenas uma oportunidade de testar e comparar novas técnicas de visão, algoritmos de tempo real entre outras, passou a ser uma importante ferramenta de integração da comunidade científica e acadêmica, envolvendo pesquisadores, professores e alunos de ensino médio técnico e universitário (COSTA & PEGORARO, 2000) (NAIR *et al.*, 2002).

Com relação à robótica aplicada à área educacional, trata-se de uma ferramenta que fornece aos alunos, a oportunidade de vivenciar experiências semelhantes às que terão na vida real, dando a estes a chance de solucionar problemas difíceis mais do que observar formas de solução (VALENTE, 1993). A robótica é uma ferramenta interdisciplinar, visto que a construção de um novo mecanismo, ou a solução de um novo problema, freqüentemente extrapola a sala de aula. Na tentativa natural de buscar uma solução, o aluno questiona professores de outras disciplinas que podem ajudá-lo a encontrar o caminho mais indicado para a solução do seu problema (BECKER, 2001).

A camada de comunicação *wireless* descrita neste artigo faz parte da plataforma robótica móvel desenvolvida para área educacional. Esta plataforma tem como objetivo permitir a criação de atividades didáticas baseada em *edutainment*. Esse termo foi assinalado por Richard Oliver para designar a reunião das indústrias do entretenimento e da educação com uma visão das possibilidades de futuro que Oliver projetava para meados deste século. Em linhas gerais uma aliança entre as poderosas empresas da informação, de games e entretenimentos, e as instituições de ensino (EGENFELDT-NIELSEN, 2011). Existem no mercado diversos sistemas robóticos que provêm pacotes de *software* e robôs móveis com aplicações focadas na educação como Kepera (KTEAM, 2017), Pioneers e Seekur (OMRON, 2017) entre outras. Contudo, estas plataformas são focadas na utilização individual dos robôs para o cumprimento de tarefas específicas. Na plataforma robótica desenvolvida, os robôs podem se comunicar por meio de um rádio-modem, tornando possível a formação de um sistema de controle distribuído e descentralizado. Esse sistema pode ser usado para formar um grupo de robôs que se comportam como um "enxame de abelhas", no qual a inteligência do conjunto é emergente e maior que a soma da inteligência das partes (NONDADA *et al.*, 2002). Tal sistema inteligente possui alta flexibilidade e complexidade, suficientes para resolver tarefas das mais variadas. Este é um conceito da área de Inteligência Artificial, mais especificamente no campo de agentes autônomos e inteligentes (FONG *et al.*, 2002).

A relevância dessa plataforma robótica desenvolvida consistiu na combinação de tecnologias para o desenvolvimento de um *kit* aplicado tanto para o ensino médio técnico e graduação, quanto para pesquisas, permitindo trabalhar com o conceito de times de futebol de robôs ou até mesmo "enxame" de robôs. Sistemas com este grau de flexibilidade e abrangência não é comum no país. Com a plataforma também é possível utilizar a Internet para realizar atividades de tele-operação, isto é, controlar e programar o robô remotamente pela WEB possibilitando uma gama de subprojetos a serem desenvolvidos futuramente na área de telerobótica. Como por exemplo, uma ferramenta de ensino a distância em escolas que não dispõem de recursos para constituírem um laboratório especializado em robótica educacional. Essas escolas poderão adquirir apenas o *software* e utilizar, por meio da Internet, os robôs disponíveis em outro local físico, como por exemplo, centros de robótica aplicada.

2 Plataforma robótica móvel

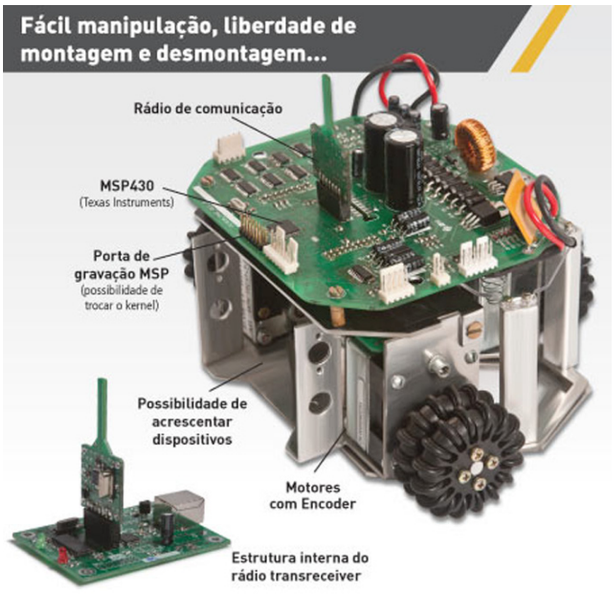
O objetivo geral do projeto foi o desenvolvimento do *hardware* do robô móvel e da estação rádio-base. Este projeto teve como meta introduzir no mercado brasileiro uma plataforma robótica com a aplicação de *software livre*. No caso específico, dessa etapa do trabalho, o foco foi na construção de um protocolo de comunicação entre o robô e a estação rádio-base que pudesse, futuramente,

ser modificado pelos usuários da plataforma conforme a necessidade da aplicação a ser dada ao robô móvel.

A plataforma robótica desenvolvida consiste em um sistema composto por um computador ou notebook com *software* instalado para controle dos robôs (pode ser controlado mais de uma unidade), por uma estação rádio-base (hardware conectado ao computador que provê comunicação sem fio), unidades de robôs móveis além de um conjunto de baterias recarregáveis e um carregador.

Na Figura 1 é apresentada uma visão do *hardware* da solução final do robô. No caso esse robô é formado por um módulo de alimentação com baterias recarregáveis e o módulo de movimentação composto por três motores independentes com três *encoders*, possibilitando movimentação para frente, para trás, para os lados, rotação no próprio eixo ou movimentos compostos com os descritos anteriormente. Também existem outros quatro módulos: módulo de rádio para comunicação sem fio com o microcomputador; módulo de processamento que controla todo o robô; módulo de visão que consiste em uma micro-câmera que transmite as imagens ao microcomputador; e por fim, o modulo de sensoriamento que é composto por quatro sensores que detectam a presença de obstáculos e informam qual a distância do robô a um objeto próximo.

Figura 1 – Imagem do robô móvel e do radio transreceiver



Fonte: site da Xbot (2016)

No caso da estação rádio-base, ela consiste de um circuito transceiver de rádio e uma porta USB (*Universal Serial Bus*) para ser conectada ao computador. Cada bateria consiste de um pack especial que contém várias pilhas conectadas de 1.5V. Foi necessário desenvolver uma bateria própria para que possibilitasse suprir a energia do robô conforme sua necessidade (pulsos de consumo alto quando o robô inicia o movimento saindo da inércia e mudando de direção bruscamente). Um carregador também foi desenvolvido especialmente para este tipo de bateria e possui um monitoramento da temperatura durante a carga rápida (Figura 2).

Figura 2 – Imagem do rádio transreceiver jutamente com o carregador e as baterias do robô móvel

Fonte: site da Xbot (2016)

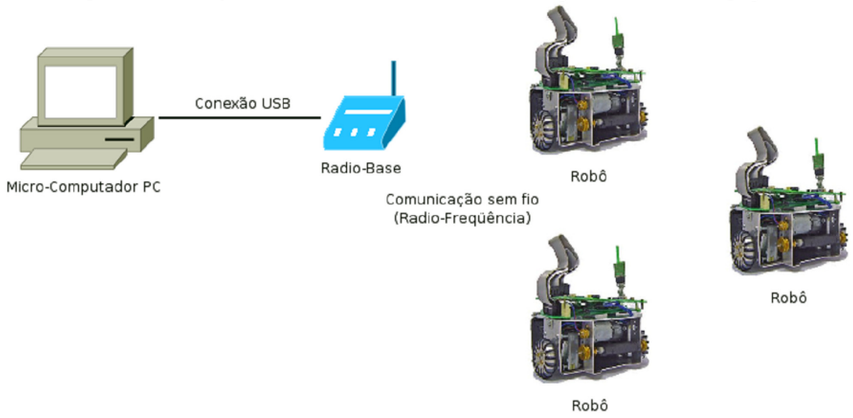
No caso do *software*, o conceito da plataforma universal abrange dois tipos de usuário final: usuário especializado que quer elaborar seus próprios algoritmos de controle e o usuário menos especializado que quer movimentar o robô de maneira fácil ou está interessado em outras técnicas que não a de programar diretamente. O *softwar*e que é executado no computador está disponível em duas camadas distintas. A Camada 1 consiste em um *software* completo, onde o usuário final poderá: determinar trajetos para o robô andar; fazer gravações de vídeo ou fotos enquanto o robô anda; determinar programas com laços, condições de sensores; gravar estes programas em disco; e trabalhar com a programação em blocos onde o usuário tem uma interface gráfica no estilo "*drag-and-drop*" para montar os seus programas.

No caso da Camada 2, a mesma, consiste em um *software* em forma de biblioteca e compilador, onde o usuário final poderá: escrever um programa em linguagem C++; compilar o programa e executá-lo; elaborar sua própria técnica de controle e inclusive comandando vários robôs. Como complemento, o *software* possui um conversor da Camada 1 para a Camada 2, onde um usuário iniciante poderá programar em blocos e solicitar uma conversão de blocos para programa em linguagem C++.

Cada robô é dotado de um *transceiver* de rádio-freqüência operando a 160 Kbps. Para permitir a comunicação entre os robôs, e entre, o computador e os robôs com segurança, foi necessário o uso de códigos de tratamento de erros (ruído no rádio), que permitiram a detecção e/ou a correção de erros. Para abordar este problema, foram combinados os algoritmos de Hamming, CRC (*Cyclic Redundancy Check*) e de paridade (WEBER, 2000). Na Figura 3 é apresentada uma visão geral do uso dessa plataforma. Baseada na análise da necessidade da aplicação foi determinado que o computador PC sempre funcione na comunicação como um equipamento "mestre" (master) e os robôs como "escravos" (*slav*e). Portanto, os robôs apenas respondem às solicitações do computador. Os robôs não precisam enviar um dado ao computador PC sem que o mesmo solicite antes. A seguir as principais características dessa comunicação:

- Capacidade de enviar dezenas de comandos por milissegundo (taxa máxima de 115200 baud rate);
- Formato simples do pacote (visando usar poucos recursos de processamento no robô);
- Integridade (o sistema confere se os dados que foram recebidos são os mesmos que foram enviados);
- Confiabilidade (o software reenvia os dados em caso de perda na transmissão ou na resposta).

Figura 3 – Componentes do sistema de comunicação e suas interligações



Fonte: elaborada pelo próprio autor

Uma vez que os robôs e o computador PC usam a mesma banda e podem transmitir simultaneamente, o uso de um protocolo de comunicação é indispensável. Como o meio de transmissão é compartilhado (todos transmitem e recebem na mesma banda de freqüência), um robô pode, por exemplo, começar a transmitir sinais para o computador PC ou para outro robô ao mesmo tempo em que um dos robôs está realizando uma comunicação com o computador PC. Caso isto aconteça, os sinais são misturados, resultando na destruição da comunicação de ambas as partes, ou seja, nenhuma das entidades envolvidas será capaz de conversar. A utilização de protocolos como ALOHA ou CSMA (*Carrier Sense Multiple Access*, ou acesso múltiplo com detecção de portadora) e suas variações (STALLINGS, 2000), permitiu que vários dispositivos (robôs e computador PC) acessem o meio de transmissão sem conflitos. Eles minimizam, com variadas taxas de eficiência, a ocorrência de colisões, permitindo que seja realizada uma comunicação robusta.

3 Metodologia

A metodologia de trabalho iniciou-se com um estudo do projeto inicial do sistema de comunicação. Em seguida foram levantadas as necessidades e as características desejadas do projeto. Posteriormente, foi realizada a proposição dos caminhos que seriam seguidos; e realizada a deliberação das decisões sobre as soluções tecnológicas a serem integradas. Por fim, foi realizada a implementação, e posteriormente, a validação da camada de comunicação *wireless* proposta.

O primeiro trabalho realizado foi no desenvolvimento de um novo *software* interno do rádio-base (*software* embarcado). Verificou-se que o atual estado de funcionamento do rádio-base não era adequado. O código fonte original tinha diversos defeitos devido ao uso excessivo de memória ou falha em processamentos de códigos "em paralelo" com interrupções do sistema (áreas críticas – *race conditions*) (TANENBAUM, 2003). Foi realizada uma remodelagem e novamente implementada em linguagem C/C++ para o microcontrolador MSP430 da Texas Instruments. Para a fase de validação do rádio-base foi montado um procedimento de testes que envolviam a utilização de um computador PC com um *software* de testes para enviar pacotes ao rádio-base; e um robô móvel com um *software* "*dummy*" que responde aos pacotes. Outro teste realizado usou um rádio-base com um *software* interno que responde aleatoriamente a alguns pacotes, ocasionando uma transmissão simultânea com o robô e a perda do pacote por interferência (para simular interferências).

Para o desenvolvimento desse rádio-base foi estudado os *datasheets* e também os *application notes* dos fabricantes dos componentes eletrônicos envolvidos. A metodologia para resolução do *software* do computador PC foi realizada da seguinte maneira:

- Reuniões com a equipe de desenvolvimento do aplicativo destinado ao usuário para determinação da API (*Application Programming Interface*) do mesmo;
- Pesquisa sobre o funcionamento de *threads* em *software* com comunicação USB e acesso de várias *threads* para mesma porta USB;
- Codificação da classe de acesso à comunicação do robô;
- Validação da comunicação utilizando ambiente real, com três robôs simultâneos e um robô simulando ruídos ou falhas na resposta;
- Integração com o *software* da equipe do aplicativo do usuário final.

Foi desenvolvido um protocolo de comunicação para atender as necessidades do projeto. O protocolo foi baseado em pacotes, e cada pacote tem 30 octetos (tamanho fixo), sendo:

- 5 octetos: endereço do dispositivo "fonte";
- 5 octetos: endereço do dispositivo "destino";
- 1 octeto: canal de rádio utilizado;
- 1 octeto: tamanho dos dados úteis;
- 2 octetos: comando;
- 0 a 14 octetos: argumentos do comando;
- 1 octeto: digito verificador de integridade ("*checksum*");
- 1 octeto: "*package*-ID";
- Demais octetos: preenchidos com zero (não utilizados).

O campo "*checksum*" possibilita que o receptor do pacote verifique se os dados recebidos conferem com os que foram transmitidos. Tanto os pacotes de solicitação, quanto os de resposta seguem o mesmo padrão acima descrito. Uma resposta quando comparada a uma solicitação, tem os campos "destino" e "fonte" trocados entre si, e o "comando" e "*package*-ID" permanecem o mesmo. O campo "*package*-ID" é um identificador sequencial do pacote, e tem a função de auxílio no caso de comunicação com falhas. Este campo foi adicionado posteriormente na comunicação para resolver o caso de retransmissão. Estes casos de retransmissão são abordados nos exemplos a seguir (Quadro 1):

Quadro 1 – Exemplo de transmissão de comando entre o computador PC e o robô

Passo	Comando	Número do package
1	PC envia para robô: ande 10 cm para frente	package-ID = 1
2	Robô envia para PC: comando recebido com sucesso	package-ID = 1
3	PC envia para robô: ande 20 cm para trás	package-ID = 2
4	Robô envia para PC: comando recebido com sucesso	package-ID = 2
5	PC envia para robô: ande 5 cm para direita	package-ID = 3
6	O pacote do PC sofre uma interferência e não chega ao Robô	
7	Após um tempo, o PC percebe que a resposta não foi recebida e repete: PC envia para robô: ande 5 cm para a direita	package-ID = 3
8	Robô envia para PC: comando recebido com sucesso	package-ID = 3
9	PC envia para robô: ande 10 cm para frente	package-ID = 4
10	Robô recebe o pacote e responde: comando recebido com sucesso com package-ID = 4, porém a resposta sofre uma interferência e não chega ao PC	
11	Após um tempo, o PC percebe que a resposta não foi recebida e repete: PC envia para robô: ande 10 cm para frente	package-ID = 4

12	O robô percebe que o pacote é uma retransmissão, então como ele já executou o movimento, apenas responde novamente, sem executar de novo	
13	O PC recebe a resposta: comando recebido com sucesso	package-ID = 4
14	PC envia para robô: ande 5 cm para a esquerda	package-ID = 5
15	O pacote é corrompido e chega com erro de "checksum" no robô: o robô não executa o comando, nem responde; apenas descarta o pacote recebido	
16	Após um tempo, o PC percebe que a resposta não foi recebida e repete: PC envia para Robô: ande 5 cm para a esquerda	package-ID = 5
17	Robô envia para PC: comando recebido com sucesso	package-ID = 5
18	PC envia para robô: ande 5 cm para a direita	package-ID = 6
19	O robô recebe o pacote, executa o comando e responde: comando recebido com sucesso	package-ID = 6
20	O PC recebe a resposta corrompida (com erro no "checksum"), então o PC repete o pedido: ande 5 cm para a direita	package-ID = 6
21	O robô percebe que o pacote é uma retransmissão, então como ele já executou o movimento, apenas responde novamente, sem executar de novo: robô envia para PC: comando recebido com sucesso	package-ID = 6

Fonte: elaborada pelo próprio autor

É possível notar no exemplo acima que foram encontrados quatro problemas possíveis na comunicação e que eles foram tratados corretamente.

- Problema 1: pacote não chegou - na transmissão do comando do PC para o robô – tratado nos passos 5 a 8;
- Problema 2: pacote não chegou - na transmissão da resposta do robô para o PC – tratado nos passos 9 a 13;
- Problema 3: pacote corrompeu – na transmissão do comando do PC para o robô – tratado nos passos 14 a 17;
- Problema 4: pacote corrompeu – na transmissão da resposta do robô para o PC – tratado nos passos 18 a 21;
- Comunicação normal: exemplos nos passos 1, 2 e também em 3, 4.

A retransmissão ocorre por *timeout*, ou seja, se após um determinado tempo, o computador PC não obteve uma resposta bem sucedida, ele retransmite o pacote. Após algumas falhas sucessivas, o PC indica o estado de "falha de comunicação". O tempo de *timeout* e o número de falhas consecutivas são configuráveis no *driver* do PC, uma ideia da ordem de grandeza destes valores é de 50 ms para o *timeout* e de três falhas sucessivas.

No protocolo, o campo destinado ao "comando" indica o que será realizado pelo robô. A seção de "argumentos" no protocolo é dependente do comando. Os comandos que foram determinados para o robô são divididos em três grupos: de controle ou diagnóstico, de instrumentação e de movimentação. No quadro 3 são apresentados os comandos de controle ou diagnóstico. No caso do comando de instrumentação, existe somente um chamado "getSensors" que retorna o estado dos sensores de proximidade do robô. No quadro 3 são apresentados os comandos de movimentação.

Quadro 2 – Comandos de controle ou diagnóstico

Comando	Função
“ping”	Comando para teste de comunicação
“getStatus”	Retorna o valor de algumas variáveis internas do robô, dentre elas, se a fila de comandos está vazia, se a fila está cheia, se existe um comando de movimentação ainda não concluído (em progresso)
“clearQueue”	Comando que limpa a fila de comandos do robô
“cancelMovement”	Comando que cancela o movimento atual
“cameraPower”	liga ou desliga a câmera do robô. Este comando é exclusivo aos robôs que estão equipados com câmera

Fonte: elaborada pelo próprio autor

Quadro 3 – Comandos de movimentação

Comando	Função
“move”	Comando de movimentação que recebe a velocidade, a distância e o ângulo que o robô deve efetuar. Este comando é processado em filas. Se forem enviados vários comandos “move”, eles ficam armazenados e cada um é executado assim que o anterior é completado. Com este comando pode-se fazer vários tipos de movimentos: movimentos lineares de eixo único (ex: para frente, para trás ou para os lados); movimentos lineares de eixo duplo (ex: andar para frente no sentido 30º para a direita); e rotação no eixo central (ex: girar 60 graus em torno do próprio eixo)
“moveIndep”	Comando genérico de movimentação, recebe a velocidade e a distância de cada motor do robô, de forma independente. Este comando é processado em filas (análogo ao comando “move”). Com este comando, além dos movimentos disponíveis no comando “move”, se pode fazer outros movimentos, como: movimentos combinando translação com rotação; e curva aberta (ex: curva para frente, de 90 graus e raio 50cm)
“moveNow”	Comando de movimentação similar ao “move”, porém ao receber um comando “moveNow”, o robô cancela o movimento atual, cancela os movimentos armazenados na fila e inicia imediatamente o movimento solicitado
“moveNowIndep”	Comando que combina as características dos comandos “moveIndep” e “moveNow”
“setTweezer”	Comando que movimenta a pinça do robô – pode-se solicitar para subir ou descer a pinça. O comando é executado imediatamente, não depende da fila de comandos. Este comando é exclusivo aos robôs que estão equipados com pinças
“kick”	Comando para controlar a velocidade e o sentido do dispositivo de drible e controlar o dispositivo de chute. Este comando é exclusivo aos robôs que estão equipados com esses itens

Fonte: elaborada pelo próprio autor

Existem duas necessidades distintas de comunicação: “apenas um robô” e “vários robôs”. Como o primeiro caso envolve diversas simplificações no *driver*, optou-se por desenvolver dois *drivers*, um para cada caso. Os *drivers* foram desenvolvidos como um objeto em linguagem C++, sendo implementados em fontes”.cpp” e “.h”, permitindo a compilação e uso direto no aplicativo final. Os dois *drivers* desenvolvidos são: *Driver* simplificado e *Driver* com *threads*. O *driver* simplificado é um *driver* com API simples que possibilita o envio de um comando e a recepção da resposta do mesmo. O programa principal permanece bloqueado, até que se tenha uma resposta (ou um *timeout*). No caso do *driver* com *threads* é um *driver* com API que inclui uma fila de comandos. Quando o programa principal solicita o envio de um pacote, o *driver* copia o mesmo para uma fila e o programa principal continua sua execução imediatamente. Em uma *thread* separada, o comando é enviado e retransmitido se necessário. Ambos os *drivers* possuem duas camadas de acesso:

- Camada *low-level*: chama-se o método/função de transmissão com os 30 octetos que compõem o pacote e o *driver* cuida de transmití-los, exemplo: Resposta = com->send*Package*(pacoteDados);

- Camada *high-level*: chama-se um método/função de transmissão para cada comando desejado, exemplo:
- com->robotMove(velocidade, direção);
- com->robotTurn(velocidade, ângulo);
- com->robotCameraOn( );
- com->robotCameraOff( );

Os *drivers* possuem uma camada de acesso à porta serial/USB com diretivas de compilação (#ifdef) para ser compatível com a compilação tanto para G++/Linux (GNU, 2016) quanto para Borland C++/Windows. Foi realizada uma análise do funcionamento do rádio-base original, e posteriormente, desenvolvido um novo código para o *software* embarcado.

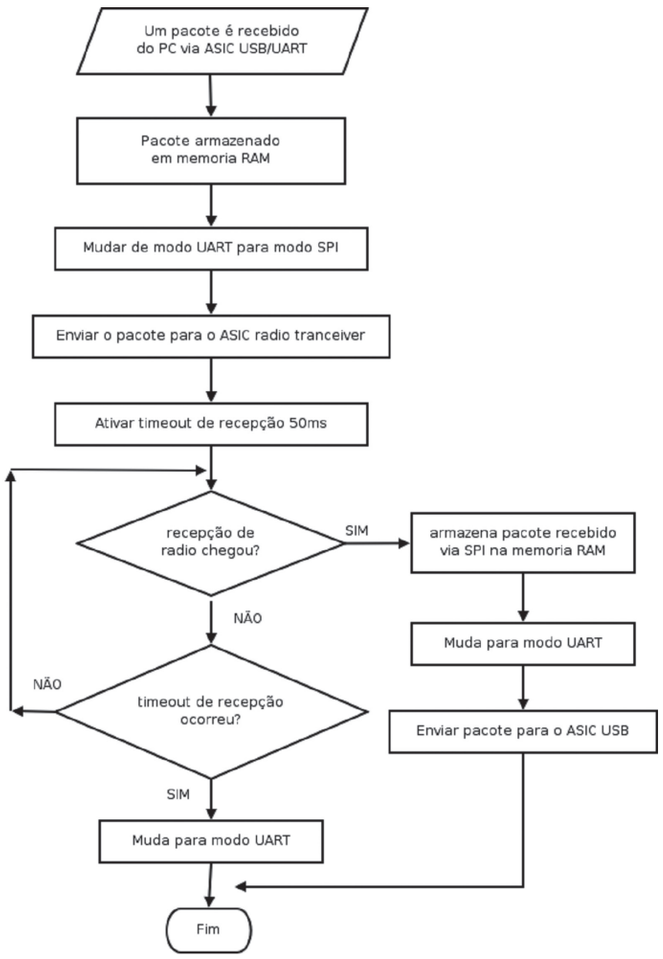
O *hardware* do rádio-base possui como principais componentes: o microcontrolador da família MSP430 da Texas Instruments (TEXAS, 2016); o circuito integrado ASIC (*Application Specific Integrated Circuit*) – conversor USB para UART da fabricante FTDI-Chip. Este circuito integrado é um conversor USB para UART (*Universal Asyncronous Receiver/ Transmitter*). Ele implementa um *slave* USB, simulando uma porta serial para o computador PC, e implementa uma UART, desse modo à transmissão fica transparente, como se estivesse utilizando uma porta serial comum no PC (FTDI, 2016).

E o circuito integrado ASIC – transceiver de rádio da fabricante Nordic Radio Inc. Este circuito integrado implementa uma solução completa de comunicação para pequenas distâncias à rádio. Ele utiliza 127 canais de radiofrequência na faixa de 2.4GHz e baixas potências. Seu alcance é adequado ao uso do robô, e em nossos testes, funciona normalmente em um raio de 10 metros (NORDIC, 2016). O transceiver da Nordic possui um protocolo próprio de comunicação, onde os pacotes tem tamanho fixo e são transmitidos com um "checksum" gerado pelo próprio ASIC, garantindo a integridade dos dados. Em caso de perda ou troca em alguma parte do pacote, o receptor rejeita o pacote completamente. O receptor apenas avisa o microcontrolador da presença de dados se os mesmos forem válidos. Em relação ao EMS (do inglês, Suscetibilidade Eletro Magnética) algumas fontes de ruído influenciam o alcance e a taxa de erros de transmissão. Em testes práticos, este *transceiver* não funciona bem se posicionado bem próximo a um monitor de computador (CRT). Mantendo o rádio-base a mais de 60 cm do monitor, esta interferência torna-se não significativa.

A comunicação relacionada ao conversor USB é realizada por meio de uma interface UART. A comunicação com o *transceiver* de rádio é realizada por meio de uma interface SPI (*Serial Peripheral Interface*). O microcontrolador do rádio-base possui apenas um módulo de comunicação USART (*Universal Synchronous Asynchronous Receiver Transmitter*), portanto o circuito multiplexa este módulo entre o modo UART e o modo SPI com os dois ASIC. Para evitar conflitos, foram utilizados os pinos RTS (*Request To Send*) e CTS (*Clear To Send*) do mesmo, ou seja, o computador PC é avisado para aguardar enquanto a USART do microcontrolador está sendo utilizada pela SPI (DICT, 2016). CTS é o sinal utilizado para controle de fluxo em comunicações seriais. O par de sinais RTS/CTS indica ao transmissor o momento em que o receptor está pronto para receber dados. Isso evita que o transmissor envie dados enquanto o receptor não estiver disponível (SCHILLING & BELOVE, 1981).

Um fluxograma simplificado do *software* embarcado do rádio-base é apresentado na Figura 4.

Figura 4 – Fluxograma simplificado do software do rádio-base



Fonte: elaborada pelo próprio autor

Esse fluxograma é apenas didático, pois não leva em consideração determinados aspectos, como: *watchdog*, chaveamento de *clock*, interrupções, *timers*, etc. Inicialmente, já existiam alguns comandos implementados para o robô móvel. Contudo após uma análise, foi identificado que alguns comandos e funções que eram necessários não tinham ainda sido codificados. Diante disso, foi providenciado a implementação dos mesmos. São eles: Comando de movimentação independente para as três rodas (para execução de curvas abertas) e imediata (aborta comandos de movimentação em execução ou na fila e executa a movimentação solicitada imediatamente); Comandos para solicitação de status da fila de execuções; Comando para verificação se um movimento solicitado anteriormente já foi completado; e técnica de retransmissão. Foram também implementados alguns comandos para que o rádio-base seja configurado pelo computador PC, sem que sejam retransmitidos dados para os robôs. Eles são:

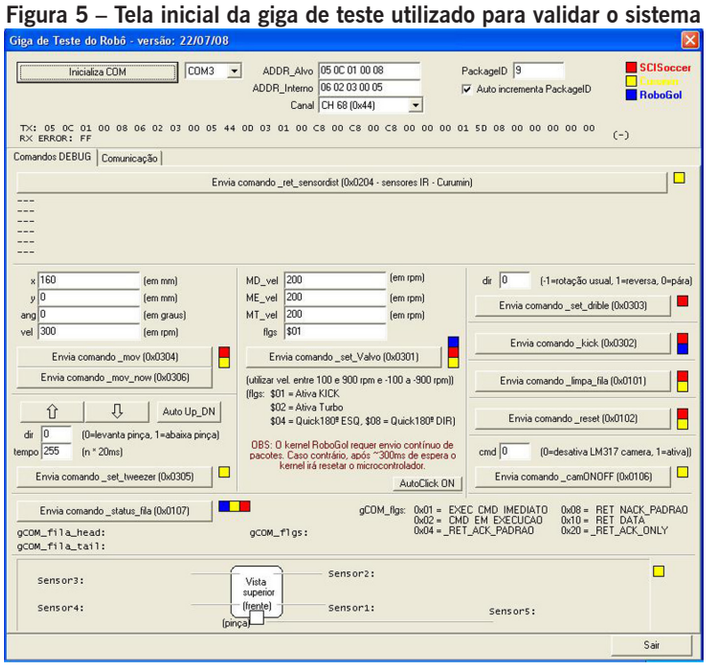
- "ping" – diagnóstico da confiabilidade da comunicação entre rádio-base e PC;
- "reset" – PC solicita que o rádio-base efetue um *reset* imediatamente;
- "handshake" – configura o rádio-base para utilizar um dos 3 modos disponíveis de controle de fluxo para a porta USB;
- "watchdog" – configura o tempo ou desativa o *watchdog* do rádio-base;
- "id" – retorna a identificação do *hardware* do rádio-base e de sua versão interna de *software*;
- "addr" – obtém ou troca o endereço interno do rádio-base.

4 Resultados

Para a realização da fase de validação, foi desenvolvida uma giga para permitir um leque de testes práticos que pudesse abranger todas as possibilidades de comunicação do sistema. Esta giga é um *software desktop* baseado no protocolo desenvolvido para comunicação wireless. Ele é capaz de testar o *software* embarcado presente em cada robô e também a biblioteca utilizada para comunicação com o computador PC. Inclusive este aplicativo está disponibilizado para o usuário em geral com o objetivo de auxiliar a detectar problemas nos robôs tanto da parte de hardware quanto do *software* embarcado e o protocolo de comunicação. Lembrando que a plataforma possui *software* e *hardware* livres, isto é, o usuário tem acesso tanto ao código fonte quanto aos esquemas elétricos para permitir que ele adicione novos módulos e funcionalidades. Em linhas gerais é possível verificar se há problemas na comunicação, enviar comandos para os motores e realizar a leitura dos sensores em tempo real.

Inicialmente foi plugado o rádio-base na entrada USB do computador PC e instalado o *driver* do rádio-base. Posteriormente, foi executado o aplicativo da giga de teste. No caso do Windows é necessário executar como administrador. A Figura 5 mostra a tela inicial do aplicativo. Nesta tela, é realizada a inicialização da comunicação com a porta COM onde está o rádio base.

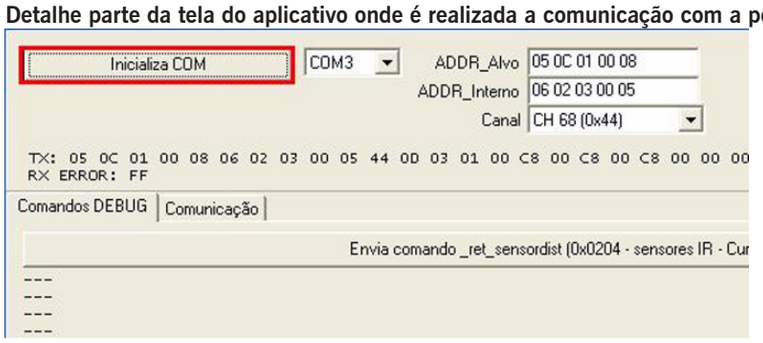
Figura 5 – Tela inicial da giga de teste utilizado para validar o sistema



Fonte: site da Xbot (2016)

Na Figura 6 é mostrada em detalhe, a parte da tela do aplicativo onde é realizada essa tarefa. Posteriormente, foi executada a tarefa de colocar os valores da identificação do rádio (RA) no campo ADDR_Interno e do robô (E.I) no campo ADDR_Alvo. Essas informações são encontradas coladas na parte de baixo do rádio base (Figura 7) e do robô móvel (Figura 8).

Figura 6 – Detalhe parte da tela do aplicativo onde é realizada a comunicação com a porta COM

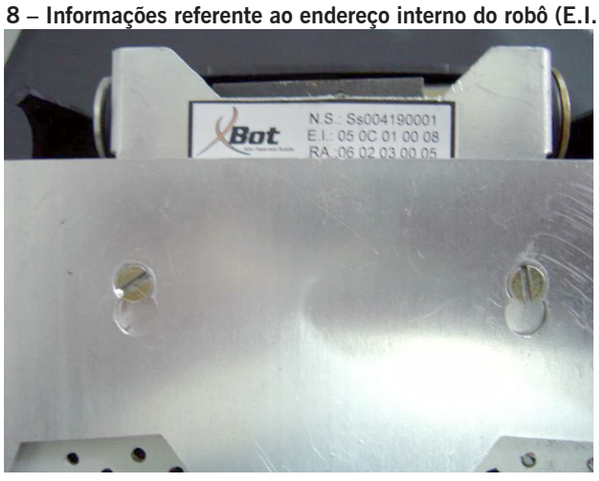


Fonte: site da Xbot (2016)

Figura 7 – Informações referente ao endereço interno do rádio base (E.I.)

Fonte: site da Xbot (2016)

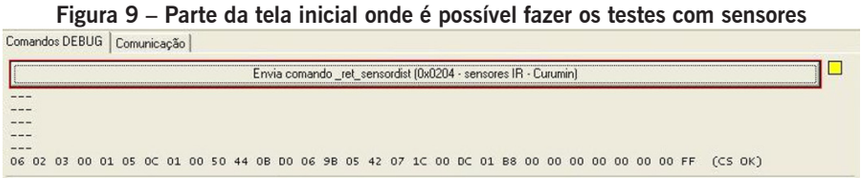
Figura 8 – Informações referente ao endereço interno do robô (E.I. e RA)



Fonte: site da Xbot (2016)

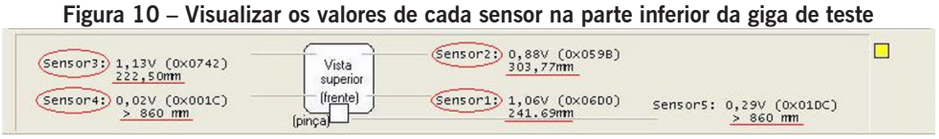
Com a conexão estabelecida foi possível realizar os testes com os sensores. Na Figura 9 é mostrada a parte da tela inicial onde é possível fazer estes testes. Basta clicar em "Envia comando _ret_sensordist" para poder ler os cinco sensores infravermelhos presentes no robô móvel. Para cada nova leitura, basta clicar nesse comando e uma nova palavra com os resultados de todos os sensores aparecem ao mesmo tempo em formato hexadecimal. Além disso, é possível visualizar os valores de cada sensor na parte inferior da giga de teste, como é mostrado na Figura 10.

Figura 9 – Parte da tela inicial onde é possível fazer os testes com sensores



Fonte: site da Xbot (2016)

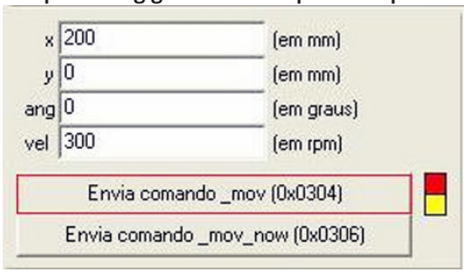
Figura 10 – Visualizar os valores de cada sensor na parte inferior da giga de teste



Fonte: site da Xbot (2016)

Para a realização dos testes de movimentação do robô foram testados vários valores de distância, como por exemplo, na tela da giga foi colocado o valor de 200 em “X” e 300 em “vel” e clicado em “Envia comando _mov” (Figura 11). Isso fará com que o robô se movimente aproximadamente 20 cm com uma velocidade aproximada de 300 rpm. O comando “Envia comando _mov_now” faz com que interrompa o comando anterior e executa o que foi por último solicitado.

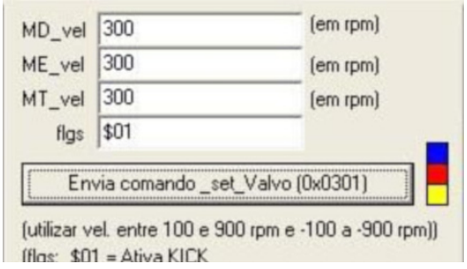
Figura 11 – Parte da tela do aplicativo giga de teste responsável por testar os movimentos do robô



Fonte: site da Xbot (2016)

Para testar especificamente os três motores do robô de forma independente foi possível utilizar outra funcionalidade presente na giga de teste. A Figura 12 mostra onde foi colocado o valor 300 no campo “MD_vel”, que corresponde a velocidade do motor direito em rpm, posteriormente, o valor 300 no campo “ME_vel”, que corresponde a velocidade do motor esquerdo em rpm e, por fim, colocado o valor 300 no campo “MT_vel”, que corresponde a velocidade do motor traseiro em rpm. Para executar, bastou clicar no botão “Envia comando _set_Valvo”. Isso fará com que o robô mantenha ligados os três motores e é preciso clicar “Envia comando_limpa_fila” da giga para que os motores sejam desligados.

Figura 12 – Tela inicial da giga de teste utilizado para validar o sistema



Fonte: site da Xbot (2016)

5 Considerações finais

O projeto foi desenvolvido adequadamente com destaque para o software-driver de comunicação entre o robô e o computador PC que foi testado e se mostrou estável sem interferência externa. Também o *software* embarcado (*firmware*) e o *hardware* do rádio-base foram implementados e testados em diversas condições de uso. Além da camada de interpretação de comandos do robô móvel devidamente implementada.

Tanto o *drive*r do computador PC quanto o rádio-base também foram testados para múltiplos robôs (três robôs simultâneos). Foram também simuladas falhas de comunicação em um ou mais robôs, e o sistema continuou funcionando nos robôs restantes, conforme desejado. Estes resultados satisfazem os objetivos inicialmente propostos com as especificações desejadas.

6 Agradecimentos

Ao apoio do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) por meio do seu programa de Desenvolvimento Tecnológico e Extensão Inovadora (DT) e da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) por meio do seu programa PIPE para pequenas empresas de base tecnológica.

7 Referências

BECKER, Fernando. **Educação e construção do conhecimento**. Porto Alegre: Artmed, 2001.

CHINZEI, K., HATA, N., JOLESZ, A., KIKINIS R., **"Surgical Assist Robot for the Active Navigation in the Intraoperative MRI: Hardware Design Issues,"** Proceedings of the IEEE/RSJ International Conference on Intelligent Robots, Vol. 1, pp. 727-732, 2000.

COSTA, A. H. R. & PEGORARO, R. **Construindo robôs autônomos para partidas de futebol: O time Guaraná.** SBA Controle & Automação, v. 11, n. 03, p. 141-149, 2000.

DICT. **Database "VERA" (Virtual Entity of Relevant Acronyms).** Disponível em: <http://www.dict.org>. Acesso em 10 de Abril de 2016.

EGENFELDT-NIELSEN, S. **Beyond edutainment: Exploring the educational potential of computer games.** Lulu, 2011.

FONG, T., I. NOURBAKHSH, AND K. DAUTENHAHN, **A survey of socially interactive robots.** Robotics and Autonomous Systems, v. 42, 2002.
FTDI. **FT232 Application notes.** Disponível em: <http://www.ftdichip.com/Products/ICs/ FT232R.htm>. Acesso em 14 de Março de 2016.

GNU. T**he Gnu C Library reference Manual.** Disponível em: <https://www.gnu.org>. Acesso em 11 de Janeiro de 2016.

KTEAM. **Khepera II.** Disponível em:<https://www.k-team.com/mobile-robotics-products/old-products/khepera-ii>. Acesso em: 18 de Novembro de 2017.

KOIVO, A. J. **Fundamentals for control of robotic manipulators.** John Wiley & Sons, Inc, 1989.

KRAMER, J. & SCHEUTZ, M. **Development environments for autonomous mobile robots:** A survey. Autonomous Robots, v. 22, n. 2, p. 101-132, 2007.

MARCHI, J. Navegação de robôs móveis autônomos: estudo implementação de abordagens. Dissertação (Mestrado em Engenharia elétrica) – Universidade Federal de Santa Catarina, Florianópolis, 2001. Disponível em:<https://repositorio.ufsc.br/handle/123456789/81441>. Acesso em 20 de Novembro de 2017.

MONDADA, F., PETTINARO, G. C., KWEE, I., GUIGNARD, A., GAMBARDELLA, L. M., FLOREANO, D., NOLFI, S., DENEUBOURG, J.-L., DORIGO, M.: SWARM-BOT: **A Swarm of Autonomous Mobile Robots with Self-Assembling Capabilities. In: Hemelrijk**, C.K. (ed.): International Workshop on Self-Organisation and Evolution of Social Behaviour. Swiss Federal Institute of Technology, Zurich Switzerland, 2002.

NAIR, R., TAMBE, M., MARSELLA, S. **Team formation for reformation in multiagent domains like RoboCup Rescue.** Proceedings of the International Symposium on RoboCup, 2002.

NORDIC **Application notes**. Disponível em: <https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF2401A> Acesso em 22 de Março de 2016.

OMRON **Research Robots.** Disponível em: <http://www.mobilerobots.com/ResearchRobots.aspx>. Acesso em 11 de Março de 2017.

SCIAVICCO, L. & SICILIANO, B. **Modeling and control of robot manipulators.** McGraw-Hill International Editions, 1996.

SCHILLING, D. & BELOVE, C. **Electronic Circuits.** Ed. Mc Graw Hill, 1981.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Autonomous mobile robots.** Massachusetts Institute of Technology, 2004.

STALLINGS, W. **Data and Computer Communications.** Prentice Hall, 2000.

TANENBAUM, A. S. **Computer Networks.** 4th edition Prentice Hall, 2003.

TEXAS INSTRUMENTS **Datasheet MSP430 Family Reference** Disponível em: <http://www.ti.com/lit/ds/symlink/msp430f5529.pdf>. Acesso em 23 de Março de 2016.
WEBER, R. F. Arquitetura de Computadores Pessoais.[S.l.]. Sagra Luzzato, 2000.

WORLD ROBOTICS. **Executive Summary World Robotics 2016 Industrial** Robots. Disponível em: <https://ifr.org/img/uploads/Executive_Summary_WR_Industrial_Robots _20161.pdf>. Acesso em 25 de Novembro de 2016.

VALENTE, J. A. **Por que o computador na educação?** In: J.A. Valente, (org.) Computadores e Conhecimento: repensando a educação. Campinas: Gráfica da UNICAMP, 1993.

XBOT. **Curumim: aprimora e acelera o aprendizado.** Disponível em: <http://www.xbot.com.br/educacional/curumim/>. Acesso em 12 de Abril de 2016.