HOLOS
ISSN 1807 - 1600

UM SISTEMA DE COMUNICAÇÃO VIA SOCKET EM UMA REDE WI-FI PARA CONTROLE DE UM ROBÔ DE INSPEÇÃO

S. R. RIBEIRO, A. S. LIMA[*], L. FALETTI[**] e C. FUSCHILO
Centro Federal de Educação Tecnológica do Rio de Janeiro (CEFET/RJ)
Coord. Automação Industrial, Dep. Eng. Mecânica, Dep. Eng. Eletrônica
alexandre.silva.lima@cefet-rj.br[*]

Submetido 06/03/2017 - Aceito 13/04/2017
DOI: 10.15628/holos.2017.5737

RESUMO

Este trabalho apresenta uma aplicação desenvolvida para controlar remotamente um robô de inspeção. Para construir o projeto eletromecânico foi utilizado um *kit Lego NXT Mindstorm*. A aplicação foi desenvolvida em JAVA e conta com uma interface gráfica que permite operar e controlar o robô remotamente através de rotinas que rodam em um ambiente cliente / servidor. A comunicação entre estação controladora (servidor) e estação robótica é feita utilizando uma rede *wi-fi*, que funciona sobre o protocolo TCP (*Transmission Control Protocol)* e UDP (*User Datagram Protocol*), juntamente com o protocolo IP (*Internet Protocol*) configurados em uma porta de comunicação. Uma segunda camada de aplicação foi desenvolvida com um conjunto de instruções (*sockets*) responsáveis pela comunicação e pelos movimentos controlados do robô, os quais são transmitidos da estação servidora para estação cliente. Para modelar o sistema, os eventos, os controles e os estados do componente robótico foram utilizados alguns diagramas da UML 2.4. O componente robótico, por sua vez, foi construído utilizando um chassi de madeira, um núcleo controlador *NXT* e três servo motores *Lego NXT Mindstorm*, além de uma estação computacional para enviar e receber os sinais *wi-fi*. Para permitir à operação e o controle a distância, foi necessário instalar uma *webcam* em um dos servo motores com rotações para direita e para a esquerda. A finalidade da *webcam* é capturar as imagens do ambiente através da estação cliente e enviá-las ao servidor em tempo real, por meio do protocolo RTP (*Real-time Transfer Protocol*). Um conjunto de *sockets* foi utilizado para enviar os comandos de operação da estação servidora ao componente robótico. Desta forma, é possível a manipulação do robô pelo controlador e o monitoramento do ambiente a ser inspecionado.

PALAVRAS-CHAVE: Lego NXT Mindstorm, Robô de inspeção, Protocolos de Comunicação, Ambiente de Controle.

A COMMUNICATION SYSTEM VIA SOCKET IN A WI-FI NETWORK TO CONTROL A INSPECTION ROBOT

ABSTRACT

This paper presents an application developed to control remotely an inspection robot. To build the electromechanical design we used a kit Lego Mindstorm NXT. The application was developed in Java and has a graphical interface that allows operating and controlling the robot remotely through routines that run in a client/server environment. Communication between controller station (server) and robotics station is made using a Wi-Fi network, which runs over TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), along with the IP (Internet Protocol) configured in a communication port. A second application layer was developed with a set of instructions (sockets) responsible for communicating and for controlled robot movements, which are transmitted from the host server to the client station. To model the system, controls events and states of robotic components were used some diagrams of the UML 2.4. The robotic component was built using a wooden frame, one NXT controller core and three servo motors Lego Mindstorm NXT, and a computer station to send and receive *wi-fi* signals. To permit the operation and control from a distance, it became necessary to install a Webcam on one of the servo motors with rotations to the right and left. The purpose of the Webcam is to capture images of the environment through the client station and send them to the server in real time, through the RTP (Real-time Transfer Protocol). A set of sockets has been used to send operating commands from the host server to the robotic component. Thus, the handling robot controller is possible and the monitoring of the environment to be inspected.

KEYWORDS: Lego NXT Mindstorms, Robot of inspection, Communications protocols, Control environment.

HOLOS
ISSN 1807 - 1600

1 APRESENTAÇÃO

Os processos industriais têm sofrido uma forte influência da automática nos últimos anos. As organizações encontraram na robótica e consequentemente na automação uma saída funcional para atender diversas atividades laborais. Estas atividades envolvem desde as necessidades mercadológicas, a obediência aos processos e padrões, a inspeção de produtos e a necessidade própria de aumentar consideravelmente a sua produção. Além disso, existem diversas atividades, sobretudo em supervisões de ambientes industriais, em que o trabalhador está sujeito a riscos potenciais contra sua integridade física.

Alguns trabalhos relacionados à Engenharia e Segurança do Trabalho, como: (MT-Br, 1978/2013), (Guimarães & Mauro, 2004), (Folkard & Monk, 1979) destacam alguns exemplos de riscos ocupacionais associados diretamente com a atividade a ser executada, como: (1) atividades submarinas; (2) atividades em ambientes explosivos; (3) atividades em ambientes confinados; e, (4) tarefas de verificação de ambientes desconhecidos ou inexplorados.

Para cada atividade a ser desenvolvida nestes ambientes existe algum tipo de risco associado ao homem ou não existe a possibilidade de realizar a tarefa. Assim, a substituição do trabalho humano pelo trabalho de uma máquina, neste caso especial, por um robô que pode ser controlado remotamente é uma saída viável e muito funcional.

Os robôs, segundo a *Robotics Industries Association (RIA)*, são dispositivos formados por componentes eletromecânicos ou biomecânicos que são capazes de realizar tarefas de maneira autônoma, multifuncional, pré-programada ou controlada remotamente, projetado para manusear materiais, peças, ferramentas ou dispositivos especiais, através de movimentos pré-programados para a realização de um conjunto de tarefas (Hedel & Hounsell, 2004).

Segundo Lima (2005) existem diversos tipos de robôs para os mais variados tipos de aplicações. Os tipos mais comuns são: Os Robôs de Inspeção, os ROV's **(***Remotely Operated Underwater Vehicle*) e AUV's (*Autonomous Underwater Vehicles*), que são utilizados na robótica submarina, os AGV's (*Automated Guided Vehicle*), são veículos com caminhos e tarefas automatizadas, e os LGV´s (*Laser Guided Vehicle*), são robôs que se deslocam sob-rodas pelo ambiente de trabalho (Lima, 2005).

Neste trabalho especificamente, o modelo de robô desenvolvido possui as características de um robô de inspeção com rodas, operado remotamente através de uma conexão *Wi-Fi* e uma interface computacional residente em uma estação servidora e uma estação cliente.

Para a construção do robô foi utilizado os seguintes componentes: (1) um chassi de madeira como plataforma; (2) um notebook com ponto de acesso *Wi-Fi* embarcado no robô como estação cliente; (3) um *desktop* como estação servidora; (4) uma webcam para captura de imagens do ambiente; e, (5) componentes do kit *LEGO NXT Mindstorm*, sendo: 3 (três) servo motores e 1 (um) controlador NXT.

Toda a parte computacional foi desenvolvida a partir de recursos de programação disponibilizados pela linguagem JAVA.

1.1 Trabalhos Correlatos

Existe uma grande variedade de ferramentas computacionais construídas e aplicadas que auxiliam a operação de componentes robóticos.

Nesta pesquisa, serão destacados os trabalhos que possuem uma correlação direta com o trabalho desenvolvido, como: (Elnagar & Lulu, 2004) que construiu uma ferramenta visual para aprendizagem apoiada por computador; (Lima, 2005) com o trabalho sobre concatenação dos

HOLOS ISSN 1807 - 1600

movimentos do manipulador e da câmera de um ROV; (Will, 2004) com uma aplicação para projetos e implementação de controles robóticos para aplicações industriais; (Ribeiro, Lima & Faletti, 2012) que desenvolveu uma interface computacional para manipulação de um braço robótico controlado remotamente; (Fung *et al*., 2011) que implementou um simulador baseado em lógica *fuzzy* para controle de suporte robótico para inspeção de vestuário; (Sakagami, Ishimaru & Kawamura, 2013) que apresentou um sistema de inspeção robótica subaquática usando contato mecânico; (Okamoto *et al*., 2011) com a proposta de um robô autônomo de inspeção de ambiente de armazenamento de gás; e (Vasile & Buiu, 2011) com um sistema computacional para aplicações de robótica colaborativa, aplicada em implementações de otimização para enxame de partículas.

2 FRAMEWORK CONCEITUAL

O robô funciona através de tracionamento, sendo este executado por dois servo motores. O controlador NXT, envia os comandos para os dois servo motores, que direcionam e controlam os movimentos do robô. Um terceiro servo motor foi equipado com uma *webcam* para permitir os movimentos de posicionamento e direcionamento da câmera no ambiente de inspeção. O objetivo é capturar e enviar as imagens ao servidor para que o robô possa ser operado pelo usuário. Um emissor / receptor *Wi-Fi* foi usado para estabelecer a conexão em ambiente cliente-servidor através de uma rede *Wi-Fi* e servindo como estação cliente. O controle remoto do robô é realizado através de uma interface gráfica simples, de fácil interação e manipulação, que possui uma tela de captura e exibição dos vídeos e botões direcionais para controlar os movimentos do robô.

2.1 . Modelagem Conceitual

A representação conceitual do sistema e o modelo arquitetural foram elaborados a partir das representações dispostas na UML, proposta por (Booch, Rumbaugh & Jacobson, 1998) e (Booch, Rumbaugh & Jacobson, 2006).

A UML foi escolhida para representar os padrões e os processos de construção e execução do software devido ao dinamismo e o poder de representação de seus diagramas (Coad & Yordon, 2002), e também pela flexibilidade e pela fácil adaptabilidade ao modelo do software proposto.

Ao todo foram desenvolvidos três diagramas: (1) diagrama de casos de uso, (2) diagrama de classes; (3) diagrama de atividades. Estes três diagramas são suficientes para a representação do modelo conceitual e da arquitetura do sistema para controle da unidade robótica.

2.1.1 Diagrama de Caso de uso

O caso de uso é um classificador que descreve a funcionalidade proposta para um sistema, que tem o objetivo de auxiliar a comunicação entre os *stakeholders*. Neste trabalho, especificamente, usamos o diagrama de casos de usos para representar as conexões no ambiente cliente-servidor e a conexão com o controlador NXT. A comunicação estende também ao usuário do sistema (representado por um *stickman*), através das telas de controle e visualização das imagens geradas pela *webcam*. A Figura 1 apresenta o diagrama de caso de uso usado na representação do modelo conceitual do sistema.

HOLOS ISSN 1807 - 1600

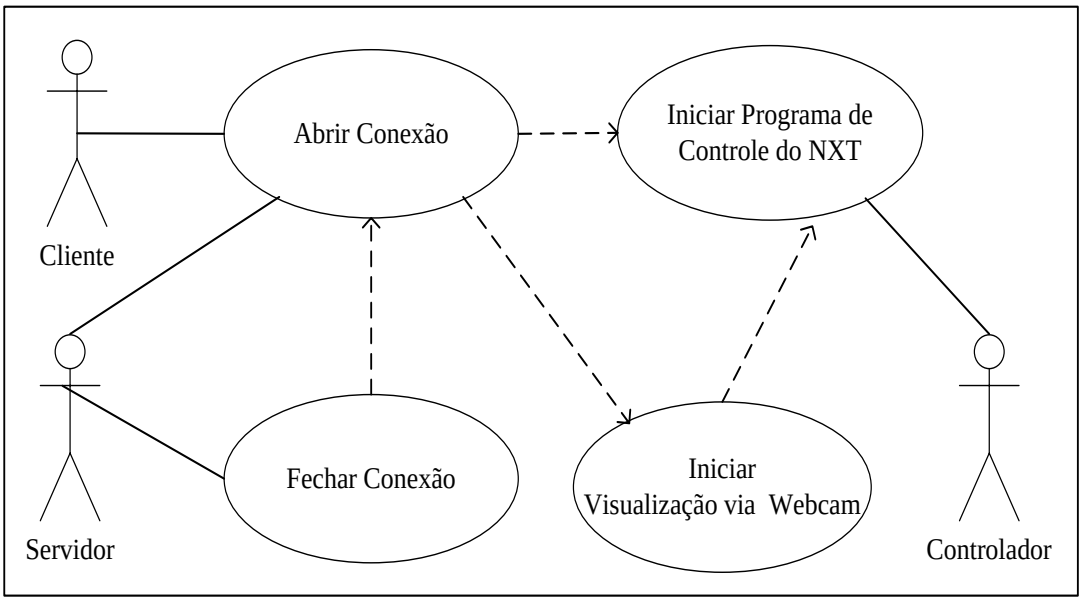

Figura 1. Diagrama de caso de uso

Os *stickman* representam o usuário do sistema e os usuários externos e/ou independentes, mas que possui uma relação direta com as funções do Software. Nesta proposta os *stickman* cliente e servidor são funções independentes do software, ou seja, que rodam fora da aplicação, mas que possui uma interface de comunicação com as funções do software e com o controlador NXT. As funções do software são representadas pelas elipses, as quais caracterizam conceitualmente cada módulo. Os fluxos indicam a relação e a conexão entre as funções. Os itens abaixo especificam cada elipse do diagrama de caso de uso.

- **Abrir Conexão:** função que representa os dois ambientes (cliente / servidor) e que estabelece uma conexão via *socket set* entre a estação cliente e a estação servidora.
- **Iniciar *NXT Control Program*:** Função que estabelece a conexão do **s**ervidor com o NXT através de uma interface USB (*Universal Serial Bus*). O cliente envia um comando ao servidor via *mouse* ou teclado, e este o envia ao NXT para executar a ação correspondente, que pode ser: (1) andar para frente; (2) retroceder; (3) ir para a esquerda; e (4) ir para a direita.
- **Fechar Conexão:** fecha a conexão quando o servidor é encerrado, finalizando também a conexão cliente.
- **Iniciar Visualização via Webcam:** Esta função tem como objetivo representar o processo de captura (envio e recebimento) das imagens do ambiente através de uma conexão *via socket*.

2.1.2 Diagrama de Classe

O diagrama de classes é um mapeamento do mundo real representado por um conjunto de classes/objetos, interfaces, colaborações e relacionamentos. O diagrama de classes é importante para a visualização, a especificação, e a representação de modelos estruturais do sistema (Booch *et al*., 2006). Após a análise e especificação dos requisitos de software, foi construído o diagrama de classe mostrado na Figura 2. O diagrama em questão consiste nas seguintes classes / objetos: Classe NXT, Classe GUI (*Graphic Unit Interface*), Classe Server e Classe Cliente.

HOLOS ISSN 1807 - 1600

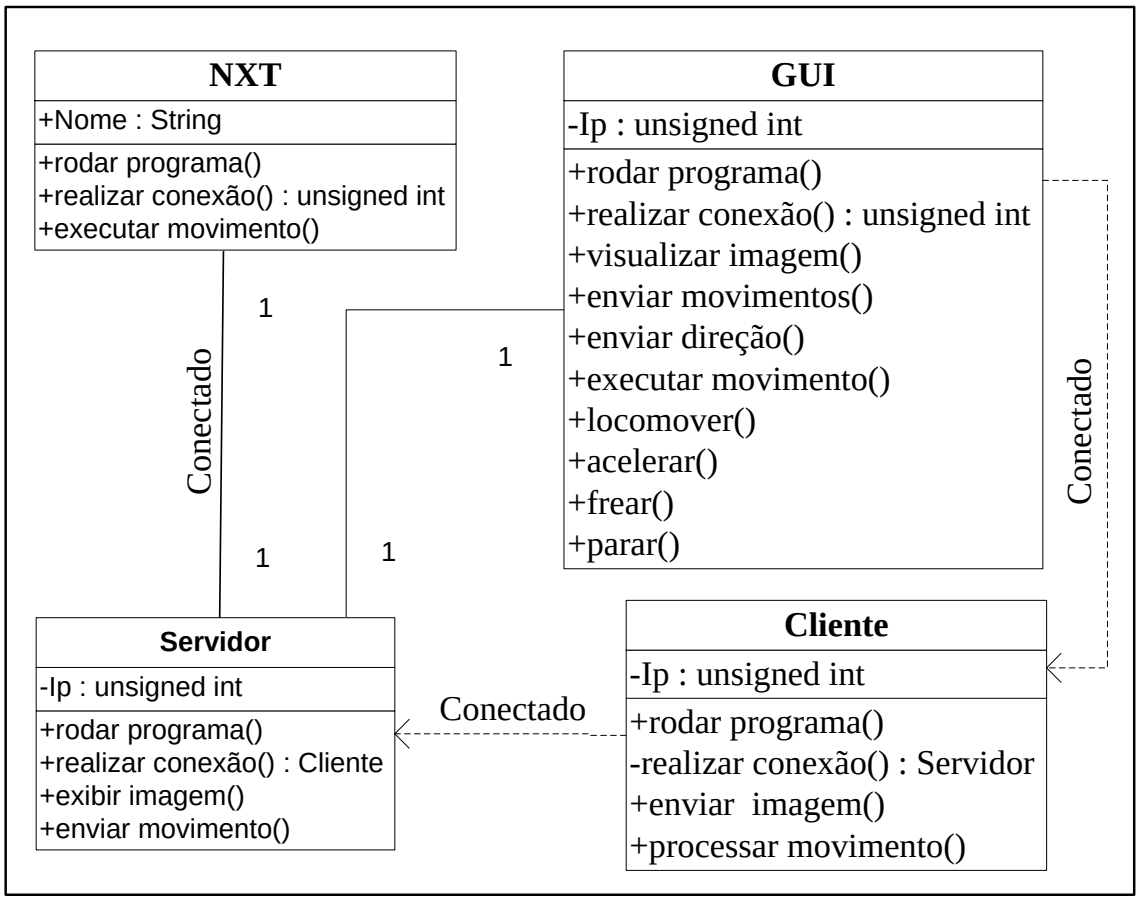

Figura 2. Diagrama de classes

2.1.3 Diagrama de Atividades

Para modelar o sequenciamento dos eventos e consequentemente os estados em tempo de execução, foi construído o diagrama de atividades da UML. O diagrama de atividade representa os fluxos conduzidos pelo processamento das tarefas (Larman, 2007). Em outras palavras, é essencialmente um gráfico de fluxo que representa a sequência exata de passos que serão executadas. O diagrama de atividade apresentado nesta pesquisa apresenta os fluxos de operação do sistema para manipulação e controle do robo de inspeção partindo de um estado inical que é a realização da conexão com o servidor. Os passos seguintes referem-se às informações necessárias para controlar o robô, usado para representar os passos de execução do robô de inspeção a partir dos eventos disparados pela ferramenta computacional em conformidade com os fluxos e as elipses estabelecidas no diagrama de caso de uso. A Figura 3 (a), (b), (c) e (d) a seguir, apresenta o sequenciamento das tarefas do ambiente computacional desenvolvido para o controle e operação do robô de inspeção.

HOLOS
ISSN 1807 - 1600

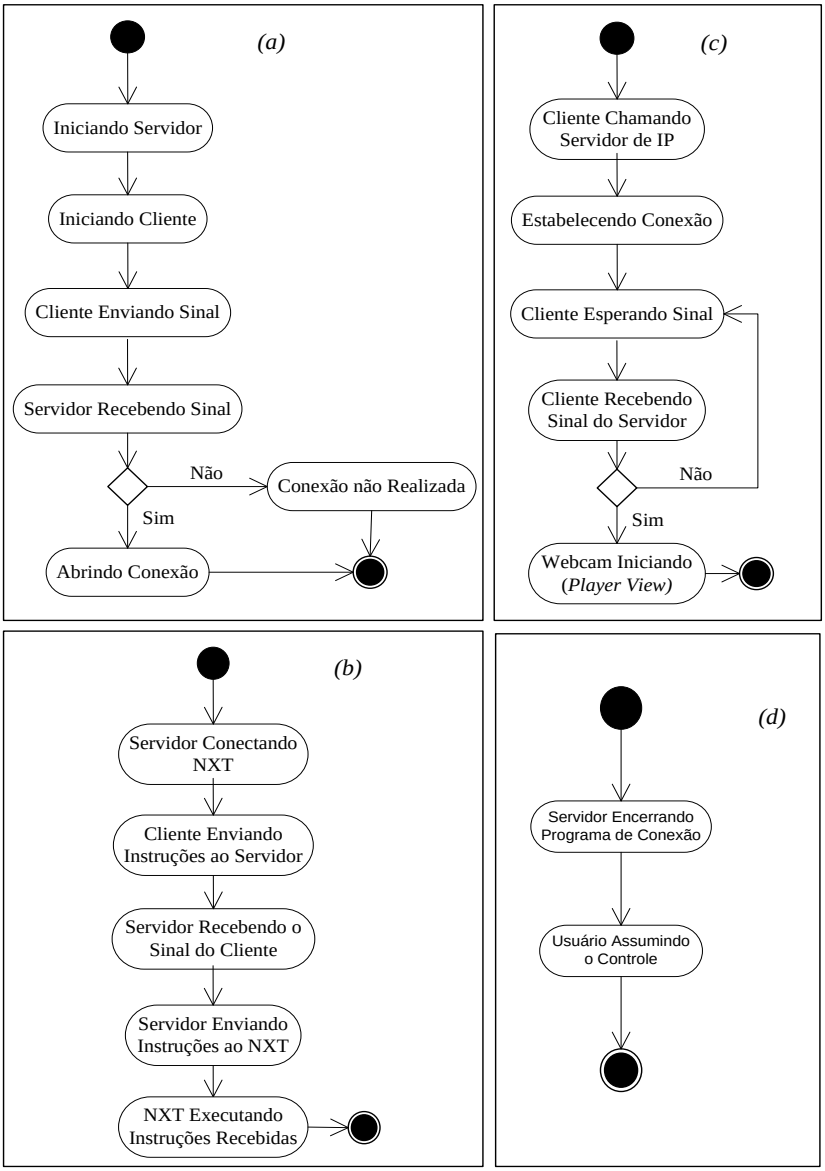

Figura 3. Diagramas de atividades

3 FRAMEWORK DE COMUNICAÇÃO, EXECUÇÃO E CONTROLE

A comunicação via *socket* é essencial para este tipo de projeto, já que esta é responsável pelo fluxo de informações nos três níveis: cliente, servidor e aplicação. A linguagem JAVA oferece dois protocolos de comunicação que podem ser usados para a programação do ambiente cliente servidor: (1) o modo orientado a conexão, que roda sobre o protocolo TCP (*Transmission Control Protocol*) (Thomas, Patel, Hudson & Ball, 1997); e (2) o modo orientado a datagrama, que roda sobre o protocolo UDP (*User Datagram Protocol*) (Hopson, Ingran & Stephen, 1997). Ambos funcionam sobre o protocolo IP (*Internet Protocol*). O modo orientado a datagrama implica que toda vez que um datagrama for enviado é necessário que o endereço do *socket* emissor e do receptor estejam contidos na "mensagem" (Silva, 2016), não sendo necessário que o *socket* origem conecte ao *socket* de destino para realizar a operação de troca de mensagens. Já o modo orientado a conexão, diferentemente do datagrama, necessita que a conexão seja estabilizada entre os *sockets*. Enquanto um dos *sockets* faz a solicitação para conexão com o servidor, o outro envia a permissão para a conexão com o cliente. Como a aplicação necessita obrigatoriamente de

HOLOS ISSN 1807 - 1600

uma conexão estabelecida para que o robô seja controlado, foi utilizado o protocolo TCP na implementação deste projeto para garantir que todo conteúdo enviado por um *socket* seja recebido pela outra extremidade exatamente na mesma ordem que foi transmitida, garantindo desta forma a integridade do fluxo de comunicação. O esquema destes dois modos de comunicação é apresentado na Figura 4 (a) e (b), representam o funcionamento do modo de comunicação por UDP e o modo de comunicação por TCP.

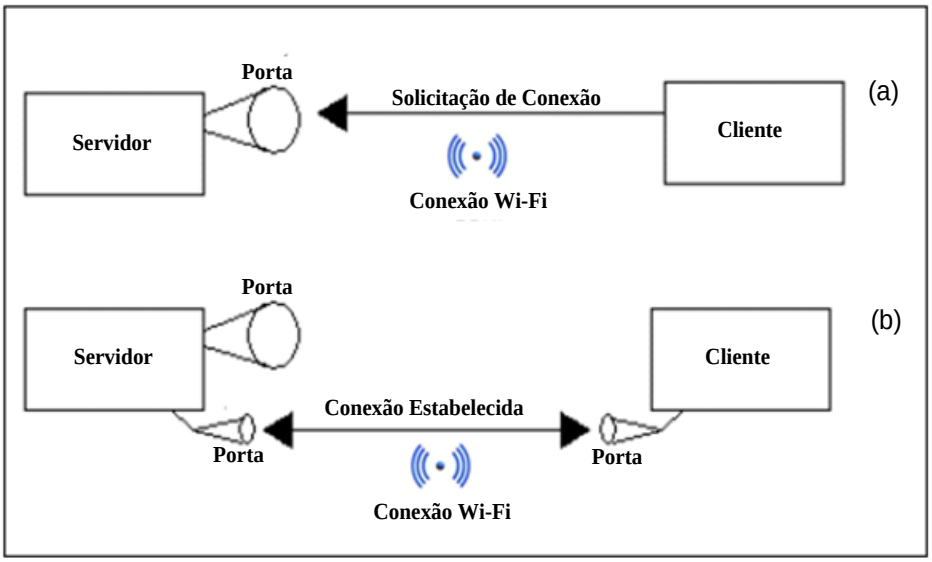

Figura 4. Comunicação via socket UDP *(a)* e TCP *(b)*. (Adaptado de (Hopson *et al*., 1997))

O entendimento do funcionamento da comunicação via *socket* permite estabelecer os critérios de comunicação usados na construção do ambiente de controle do robô de inspeção, que podem ser sumarizados por uma sequência de passos que representa o funcionamento tanto da estação cliente, quanto do servidor.

- **Cliente**

Passo 1: Cria à conexão de *socket* cliente;
Passo 2: Adquirem fluxos de leitura e escrita para o *socket*;
Passo 3: Utilizam os fluxos de acordo com o protocolo do servidor;
Passo 4: Encerra os fluxos de comunicação;
Passo 5: Encerra o *socket*.

- **Servidor:**

Passo 1: Cria a conexão de *socket* servidor e iniciam a escuta;
Passo 2: Chama o método *accept* para obter novas conexões;
Passo 3: Cria os fluxos de entrada e saída para o *socket* retornado;
Passo 4: Conduz à comunicação com base no protocolo determinado;
Passo 5: Encerra os fluxos e o *socket* do cliente;
Passo 6: Volta ao passo dois ou continuam até o passo sete;
Passo 7: Encerra o *socket* servidor.

HOLOS ISSN 1807 - 1600

Os passos descritos acima podem ser representados graficamente por um diagrama de fluxos, conforme ilustrado na Figura 5, onde o processo de comunicação via *socket* é estabelecido com o envio dos comandos para o controle do NXT.

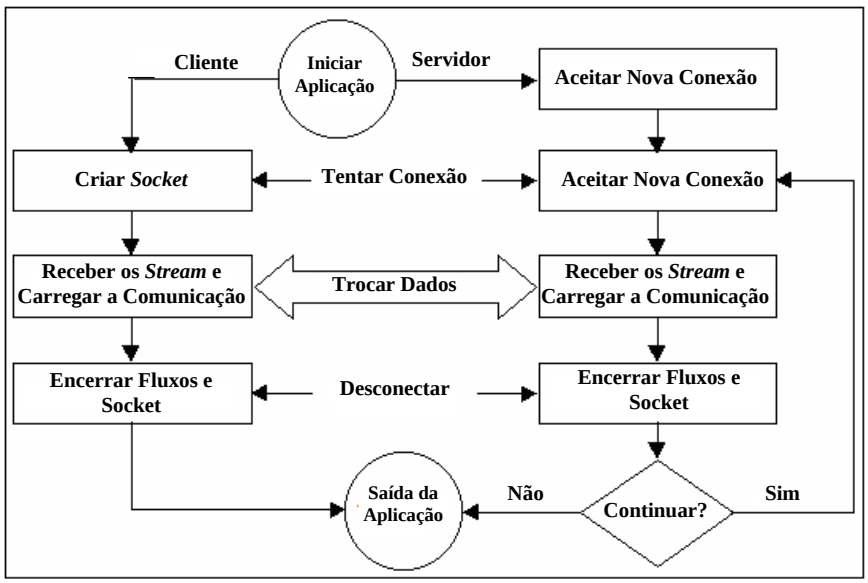

Figura 5. Diagrama de fluxo da comunicação cliente → servidor (Adaptado de (Hopson *et al*., 1997)).

3.1 Requisitos de Implantação do Framework

A Tabela 1 apresenta os requisitos necessários para o projeto de construção do *Framework* de controle do robô de inspeção. Estes requisitos são essenciais para o funcionamento correto de cada componente e estabelece todos os fluxos interno de comunicação, inclusive com o *front end* do *Framework*.

Requisitos	Justificativa	Exceção
Java Virtual Machine (JVM) instalado no cliente e no servidor.	A JVM é necessário para o correto funcionamento de toda aplicação desenvolvida em Java.	Caso não tenha o componente instalado, a aplicação não poderá ser executada.
Java Media Framework (JMF) instalado no cliente.	A JMF é necessária para o correto funcionamento de aplicações visualizadores de mídias.	Caso não tenha o componente instalado não será possível a exibição dos vídeos.
LejOS (Java for Lego Mindstorms)	O LejOS é um pacote Java para Lego NXT necessário para a conexão com o controlador Lego NXT (Falkner, 2013).	Caso não seja usada essa biblioteca, não é possível realizar a comunicação entre a estação cliente /servidor e o controlador NXT.
Java Socket (TCP)	Pacote socket set Java necessário para estabelecer a comunicação no ambiente cliente servidor	Caso não tenha a biblioteca instalada não será possível estabelecer a conexão entre o servidor e o cliente.
JAVA RTP (Real-time Transport Protocol)	O Protocolo RTP é essencial para transmissão dos sinais entre a estação cliente e a estação servidora em tempo real.	Caso não tenha este pacote não será possível capturar, enviar e receber os vídeos da Webcam.

Tabela I: Requisitos de implementação do Framework

HOLOS
ISSN 1807 - 1600

3.2 Interface de Comunicação

A interface do software permite a interação direta entre o operador e o robô. Através da imagem exibida em tempo real, capturada pela *Webcam* e através dos botões direcionais da interface de controle, o operador pode visualizar o ambiente e disparar os comandos para o robô. A interface é muito simples, porém muito funcional, sendo esta dividida em três partes: (1) visualizador de imagens: que permite ao operador visualizar o ambiente onde a máquina se encontra; (2) a função de operação dos servo motores com as rodas do robô: que possui três funções: movimentação, acelerar e desacelerar; e, (3) as funções da operação da câmera do robô: que permitem ao operador controlar a velocidade de giro da câmera e as funções de girar para esquerda e direita, além de acelerar e desacelerar o movimento. Esta função é importante porque permite ao operador observar o ambiente em volta do robô sem precisar girar o sistema todo. A tela principal do *Framework,* bem como todas suas funcionalidades é apresentada na Figura 6.

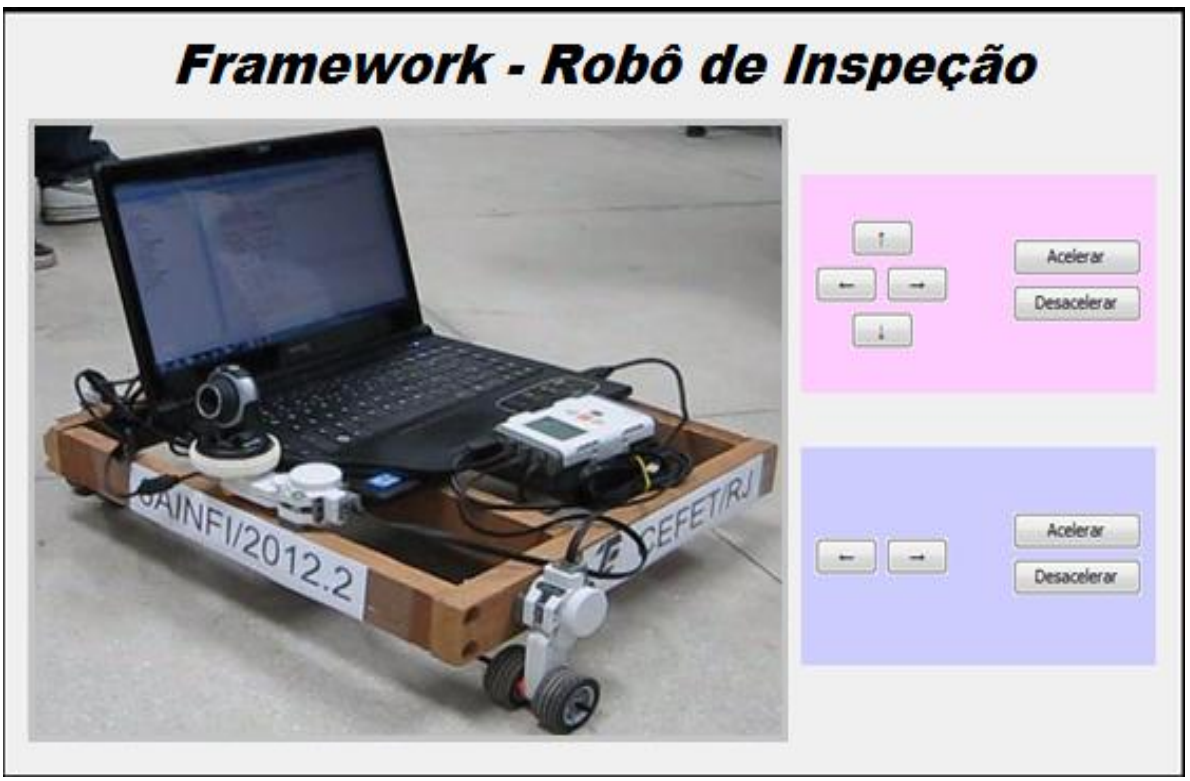

Figura 6. Interface de controle

4 AMBIENTE DE EXECUÇÃO

Como definido anteriormente, o sistema se divide em quatro partes principais: (1) Interface de controle; (2) Comunicação via *Socket* usando TCP; (3) Comunicação via RTP; e, (4) Comunicação com controlador NXT. Cada uma destas partes é responsável por uma função especifica do funcionamento do sistema, que compreende em: controle, conexão, comunicações e transmissão de um dado no sistema entre as estações. O controle é exercido pelas funções do software via interface. A conexão via *socket* é por onde passam os comandos disparados pelo o usuário. Essa conexão acontece através da porta 30 e fica permanentemente analisando a movimentação da porta enquanto existir a conexão por parte do usuário. A etapa de

HOLOS ISSN 1807 - 1600

comunicação é responsável pelo envio, recebimento e interpretação dos comandos recebidos e enviados para o NXT via USB.

Paralelamente à comunicação via *socket*, existe a comunicação via RTP. O *Real-time Transport Protocol* é o protocolo de transferência de mídia por meio de uma das portas do computador. O arquivo de mídia envolvido no sistema é o vídeo capturado pela câmera, que é enviado para uma "fonte de mídia" que foi criada para transmitir esses dados para um IP único, no servidor. É importante ressaltar também, que para utilizar a linguagem JAVA no NXT, foi necessária uma atualização de *firmware* para o LejOS, que é a versão do SO do NXT que permite a utilização direta de dados enviados via USB (*Universal Serial Bus)*. Esse SO (Sistema Operacional) permite também outras funções, como a utilização de códigos pré-compilados para a automatização do robô.

Tabela II: Tipos de Testes aplicados e resultados obtidos

Componente	Teste Aplicado	Resposta	*Status*
Conexão Cliente/Servidor	Comunicação (envio e recebimento de resposta via *socket*).	Conexão estabelecida em ≈ 0,9s (média) em uma rede Wi-Fi 10-15 dpi.	Sucesso
Webcam	Captura e visualização das imagens do ambiente.	As imagens foram capturadas e transmitidas em tempo real. Este processo depende exclusivamente da banda da rede e da distancia do robô com o ponto de acesso Wi-Fi.	Sucesso
***Framework* de Controle**	Envio de comandos para o robô (acelerar, desacelerar, esquerda, direita, ir para frente, retroceder).	Todos estímulos enviados pelo controlador foram recebidos conforme a especificação.	Sucesso
Robô de Inspeção	Recebimento dos Comandos pelo robô (acelerar, desacelerar, esquerda, direita, ir para frente, retroceder).	Os comandos recebidos foram todos executados conforme a especificação e em tempo real.	Sucesso

4.1 Trabalhos Futuros

Embora todos os testes funcionais tenham sido executados com sucesso, existem funções e componentes que ainda serão desenvolvidas e inseridas no projeto, como: sensores magnéticos, de movimento e foto sensibilidade, por exemplo. Além disso, pretende-se instalar um dispositivo de transmissão de sinal *Wi-Fi* específico para o robô com maior amplitude de sinal e programar um hardware onde será embarcado as instruções do cliente, fazendo com que o uso do notebook em sua plataforma não seja mais necessário. Pretende-se também implementar em uma próxima versão do robô um sistema para controlar o robô remotamente via *web*. Esta operação será viabilizada por meio de uma página *web* com um *applet* JAVA, onde o operador poderá de qualquer lugar colocar o robô em funcionamento e executar as tarefas de controle. Neste caso é necessário que o conteúdo esteja disponível em um serviço *www* (*World Wide Web*) e o operador esteja em um local com acesso a internet. Outra funcionalidade que também pode ser explorada no futuro com este projeto é o controle por um dispositivo móvel. Neste caso os comandos serão transmitidos para o NXT, utilizando a própria conexão Wi-Fi ou até mesmo uma comunicação via *Bluetooth*, que irá permitir realocação e operação do robô em locais mais remotos, uma vez que o sinal seria transmitido pelo dispositivo móvel e não por uma estação servidora fixa.

HOLOS ISSN 1807 - 1600

5 CONCLUSÕES

A multidisciplinaridade deste projeto exigiu durante as etapas de construção e configuração do sistema, diversos recursos disponíveis na linguagem de programação JAVA para construção do modelo conceitual e da interface gráfica do *framework*. Os componentes do LEGO NXT MINDSTORM foram necessários para a construção física do robô. Com a elaboração deste projeto foi possível visualizar a importância, a necessidade e o emprego prático deste tipo de componente robótico para a realização de tarefas de inspeção nos mais variados ambientes sem a exposição do operador. A linguagem Java com toda a sua estrutura, pacotes, classes e funções disponíveis indica que este é um caminho seguro e eficiente para construção de aplicações para a robótica e automação. Em todo o projeto de máquina foram utilizados componentes do *Lego NXT Mindstorm* e hardware disponível nas estações cliente e servidor (notebook e desktop).

É importante ressaltar que a contribuição deste trabalho está pautada no resultado computacional, que compreende a funcionalidade da aplicação desenvolvida para o controle do robô de inspeção e a comunicação via *socket.* E neste aspecto, o protótipo construído atendeu todas as expectativas e exigências enquanto projeto inicial. Assim, por conveniência, esta proposta não apresentou modelos matemáticos ou conceitos técnicos mais aprofundados dos servo motores ou de qualquer outro componente. Isto porque estes modelos já estão bastantes consolidados e disponíveis na literatura.

6 AGRADECIMENTOS

Especial agradecimento à Coordenação de Automação Industrial do CEFET-RJ / Campus Maria da Graça por permitir a elaboração técnica deste projeto em seus domínios. Agradecemos também a turma do 6º período de Informática Industrial (2012/2) por ter acreditado nesta proposta de projeto e por ter iniciado os trabalhos sob a orientação dos professores da Coordenação de Automação Industrial, Departamento de Engenharia Mecânica e Departamento de Engenharia Eletrônica do CEFET-RJ.

7 REFERÊNCIAS

Booch, G.; Jacobson, I.; Rumbaugh, J., (1998); The Unified Modeling Language User Guide. 1a. Edition; Ed. Addison-Wesley. Massachusetts - MA; USA.

Booch, G., Rumbaugh, J, Jacobson, I., (2006); UML: Guia do Usuário; Ed. Campus; Rio de Janeiro-RJ; Brasil.

Coad, P.; Youydon, E., (2002); Análise Orientada a Objetos e Projeto Orientado a Objetos; Editora Campos – Rio de Janeiro-RJ; Brasil.

Elnagar; A., Lulu, L., (2004); A Visual Tool for Computer Supported Learning: The Robot Motion Planning Example; From Proceeding On Artificial Intelligence and Applications; Innsbruck, Austria.

Falkner; J., (2013); Controlling Lego Mindstorms with Java; Oracle Java FAQ; 2008; published in Oracle 2013. <<https://weblogs.java.net/blog/jfalkner/archive/2008/02/controlling_leg.html>> acessado em Abril 2016.

Folkard, S.; Monk, T. H., (1979); Shiftwork and performance. *Human factors.* Santa Monica: Human Factors Soc., v.21, p.483-492.

Fung, E. H. K.; Wong K., Zhang , X. Z; Cheng, L., Yuen, C.W.M. , Wong, W. K., (2011); Fuzzy logic control of a novel robotic hanger for garment inspection: Modeling, simulation and experimental implementation; Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong - HK.

Guimarães, R. M.; Mauro M.Y.C., (2004); Riesgo de accidentes de trabajo: uma propuesta de indicadores ergonômicos de evaluación. Proceedings of the 3rd Conference on Occupational Risk Prevention; 2004. Santiago de Compostella, Espanha. Santiago de Compostella (Es): Asociación del Trabajador.

Hopson, K., Ingram, C, E, Stephen, E., (1997); *Desenvolvendo Applets com Java*. Editora Campus, pag. 335-351.

Larman, C., (2007); Utilizando UML e Padrões - Uma Introdução à Análise e ao projeto Orientado a Objetos e ao desenvolvimento Interativo; 3ª. Edição - Bookman.

Lima, A. S., (2005); Concatenação dos Movimentos do Manipulador e da Câmera de um ROV, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MT-Br; Ministério Do Trabalho (Br)., (2004); Decreto nº 3214 de 1978. Disposição sobre as condições mínimas de trabalho: Secretaria de Documentação; 1978. Disponível em www; 12:338-345; p.345, Brasília (DF) <<www.mte.gov.br>>. Acessado em Maio /2013.

Okamoto, J. Jr., Grassi, V. Jr., Amaral, P. F. S., Pinto, B. G. M., Pipa, D., Pires, D. P., Martins, M. V. M., (2011); Development of an Autonomous Robot for Gas Storage pheres Inspection; Springer Sciece Business Média.

Ribeiro, S. A, Lima, A. S.; Faletti, L. A., (2012); Uma Ferramenta Computacional para manipulação de um braço robótico; Proceedings in Mecom'12 Argentina and Mecom Magazine Argentina.

Redel, R., Hounsell, M. S., (2004); Implementação de Simuladores de Robôs com o Uso da Tecnologia de Realidade Virtual; DCC-CCT-UDESC Proceedings in IV Congresso Brasileiro de Computação – CBComp.

SAKAGAMI, N., Ishimaru, K., Kawamura, S., (2013); Development of an Underwater Robotic Inspection System using Mechanical Contact; Mizuho Shibata; Hiroyuki Onishi and Shigeo Murakami; *Dainippon Screen Mfg. Co., Ltd., Kyoto* Japan.

Silva, A. S., (2016); Comunicação de Computadores utilizando Sockets; FIPP/UNOESTE; Presidente Prudente – SP <<http://www2.unoeste.br/~chico/comunicacao_socket/>> Acessado em março de 2016.

Thomas, M. D., Patel, P. R., Hudson, A. D., Ball, D. A. Jr. (1997); Programando em Java para a Internet. Makron Books, pag. 467-489.

Vasile; C. I., Buiu, C., (2011); A software system for collaborative robotics applications and its application in particle swarm optimization implementations; Elsevier Applied Soft Computing; pg. 5498-5507; Department of Automatic Control and System Engineering, "Politehnica" University of Bucharest, Bucharest - Romania.

Will, D. J., (2004); Design and Implementation of Robotic Control for Industrial Applications; PhD Thesis of Engineering Electrical In the Faculty of Engineering; Port Elizabeth Technikon University; Port Elizaeth South Africa; SA.