

Arranjo Cliente Servidor para um Robô Industrial com Controladora Aberta

Thalles Oliveira Campagnani * Renato de Sousa Dâmaso *

* Depto. de Eng. Mecatrônica do CEFET-MG, Unidade de Divinópolis
(thallescampagnani@gmail.com; renatosd@cefetmg.br)

Abstract: This article describes the implementation of a software server, called OpenSever, in an industrial robotic system to facilitate the implementation of external control loops. The robotic system used is composed of an open controller, an external industrial computer and an industrial robotic manipulator. The open controller allows receiving movement commands from software compiled on the industrial computer, making use of the *eORL* library. But this system is limited to the computational capacity of the industrial computer, its architecture *x86*, its operating system based on *Linux*, the programming language C/C++, which the library *eORL* is compatible, and connection via cables. In order to overcome the aforementioned limitations, this software server was implemented in the industrial computer to enable the sending of movement commands by a software client on an external device. Such device can be of any architecture, operating system and programmed in any preferred language, as long as it has the ability to communicate with the server via the TCP/IP protocol through the network cable or wireless connection, and can do so use of simple open source operating system libraries. Once implemented, tests were carried out with some architectures, operating systems and programming languages. Some of the collected results are presented at the end.

Resumo: O presente artigo descreve a implementação de um software servidor, chamado OpenSever, num sistema robótico industrial para facilitar a implementação de malhas de controle externas. O sistema robótico utilizado é composto por uma controladora aberta, um computador industrial externo e um manipulador robótico industrial. A controladora aberta permite o recebimento de comandos de movimentação vindos de softwares compilados no computador industrial fazendo uso da biblioteca *eORL*. Mas este sistema é limitado à capacidade computacional do computador industrial, a sua arquitetura *x86*, ao seu sistema operacional baseado em *Linux*, à linguagem de programação C/C++, que a biblioteca *eORL* é compatível, e à conexão via cabos. A fim de superar as limitações citadas, foi implementado esse software servidor no computador industrial para possibilitar o envio de comandos de movimentação por um software cliente em um dispositivo externo. Tal dispositivo pode ser de qualquer arquitetura, sistema operacional e programado em qualquer linguagem de preferência, desde que tenha a capacidade de se comunicar com o servidor pelo protocolo TCP/IP através do cabo de rede ou conexão sem fio. Além disso, podendo fazer uso de bibliotecas simples de código aberto do sistema operacional. Depois de implementado, foram feitos testes com algumas arquiteturas, sistemas operacionais e linguagens de programação. Ao final, são apresentados alguns dos resultados coletados.

Keywords: C5G Open, Industrial Robotic Manipulator, Client server arrangement

Palavras-chaves: C5G Open, Manipulador Robótico Industrial, Arranjo cliente servidor

1. INTRODUÇÃO

Os robôs industriais tradicionais possuem uma arquitetura de controle fechada fornecida pelo fabricante. Esse tipo de solução segue o paradigma a qual deve ser priorizada a robustez e confiabilidade do funcionamento e da execução das tarefas, além de atender a certos requisitos normativos e legais que regem as indústrias. No entanto, é inegável que essa forma dificulta a realização de novas implementações e atividades de pesquisa com os mesmos.

Algumas tentativas feitas pela academia foram relatadas na literatura no sentido de abrir a estrutura de controle de alguns robôs industriais (Roberti et al., 2010). Re-

centemente, alguns fabricantes de robôs perceberam essa demanda por parte das instituições de ensino e pesquisa e de forma semelhante ao que vem acontecendo com os softwares de código aberto do *ROS* (Sistema Operacional Robótico, em tradução livre), oferecem a opção de abertura do sistema de controle de robôs (Kubus and Johansson, 2010), isto é, uma controladora aberta.

Uma opção para atender essa demanda foi desenvolvida pelo fabricante de robôs industriais *Comau*, adicionando uma arquitetura complementar à controladora de seus robôs, a *Comau C5G*. Essa extensão, denominada *Comau C5G Open Controller*, é feita através da conexão da controladora a um computador industrial externo, denominado

LPC (*Linux Personal Computer*), como é mostrado na Figura 1. (Antonelli et al., 2010).

A Figura 2 permite a visualização da solução fornecida pelo fabricante, sendo que, a conexão *Ethernet* TCP-IP precisa ser intermediada por um modem, roteador ou *switch* não fornecido nessa extensão.

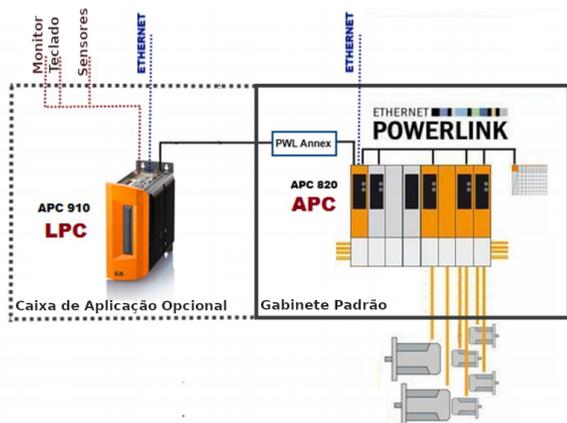


Figura 1. Visão geral sobre o hardware do sistema C5G-Open

Adaptado de (Ferrara, 2013)

Assim, é possibilitado à controladora do robô industrial enviar dados sensoriais e receber comandos de movimentação de um software a ser programado pelo usuário, a taxas de até 1 pacote a cada 0,4ms (1 pacote contém todas as leituras de sensores do robô). Isso possibilita a criação de softwares que implementem malhas de controle adicionais como: controle de força, visão computacional, entre outros. (Ferrara, 2013)

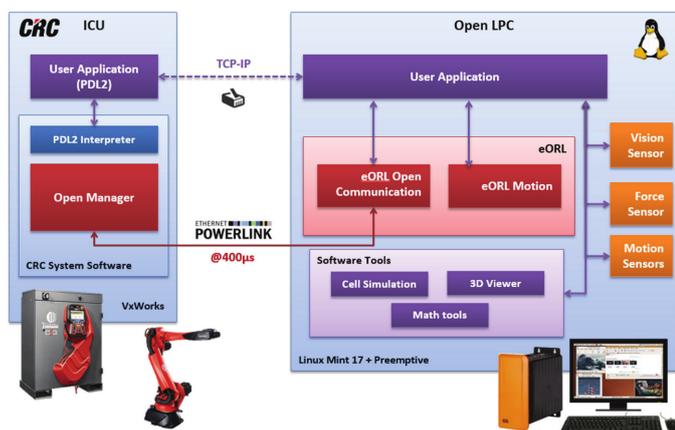


Figura 2. Visão geral do software do sistema C5G-Open
Fonte: (Comau_Robotics, 2019)

Entretanto, existem algumas limitações, por exemplo: o software (*User Application* na Figura 2) que implementa essa malha de controle adicional precisa ser compilado e executado nesse computador industrial externo, rodando o sistema operacional *Linux Mint 17* compilado para arquitetura x86, conectado à controladora por um cabo de rede especial chamado *Ethernet PowerLink* (visto na Figura 2). Ele precisa ser programado em C/C++ e utilizar a

biblioteca proprietária de código fechado desenvolvida pela *Comau* chamada *eORL* (Biblioteca Robótica Realística Melhorada, em tradução livre) para controlar a comunicação.

Além disso, uma utilização inadequada por um usuário que deixe a conexão com o *LPC* habilitada do lado da controladora e que não tenha a adequada resposta do lado do *LPC*, por exemplo, poderá produzir um erro de comunicação que acabará sendo armazenado em seus registros e impedirá a liberação dos *drivers* da controladora pelo seu módulo de gerenciamento de segurança, chamado C5G-SDM (Módulo de Distribuição e Segurança, em português) (Comau_Robotics, 2010).

Por fim, nem todos os programas necessários para o desenvolvimento de aplicações podem ser instalados no *LPC*, como é o caso do *ROS*, devido ao fato que os drivers do Sistema Operacional Ubuntu (ou derivados dele, como Linux Mint) para funcionar (Bisson, 2014). Atualmente é executado no *LPC* a versão 17 do Linux Mint, que é baseado na versão LTS (suporte de longa duração, em tradução livre) de 2014 do Ubuntu. Então, todos os programas e versões de bibliotecas do sistema operacional são versões ultrapassadas e, geralmente, não oferece suporte à instalação das novas versões dos mesmos, como é o caso do *ROS 2*, que é a mais nova versão do *ROS*.

Visando oferecer mais liberdade e flexibilidade para o usuário, no que diz respeito à programação da malha de controle adicional e ao dispositivo que irá executá-la, foi desenvolvido um software chamado OpenServer. Este, tem também o objetivo de trazer mais segurança para a controladora do robô do laboratório do CEFET-MG em Divinópolis durante a execução de malhas de controle experimentais. Ele utiliza a biblioteca *eORL* para controlar o robô e oferece uma *API* (Interface de Programação de Aplicativo, em tradução livre) baseada em rede TCP/IP, o qual permite que um programa cliente em outro dispositivo controle o robô através dele. Este programa cliente pode ser escrito em qualquer linguagem de programação, compilado em qualquer arquitetura e executado em um sistema operacional de preferência. O OpenServer permite que o programa cliente leia os sensores do robô e envie os ângulos de referência para a controladora, possibilitando que o robô seja controlado até mesmo por aplicativos de *smartphone* que utilizem a *API*.

Como mencionado acima, acredita-se que essa abordagem possa reduzir riscos ao equipamento por erro ou imperícia ao ser programada uma malha de controle usando recursos do sistema *Open* fornecidos pelo fabricante. Pois, o OpenServer é programado para continuar se comunicando com a controladora, mesmo que a conexão com o programa cliente seja perdida. Além disso, ela é capaz de oferecer uma alternativa de *API* mais simples de ser aprendida e utilizada.

2. O OPENSERVER

Para desenvolver o OpenServer foi aproveitada a estrutura padrão do sistema fornecida pela empresa fabricante, como detalhado na Figura 3, onde: (1) é a conexão elétrica entre Controladora e Manipulador Robótico; (2) representa a

conexão *PowerLink* entre *LPC* e controladora; (3) é a conexão entre roteador e controladora; e (4) conexão entre roteador e o *LPC*.

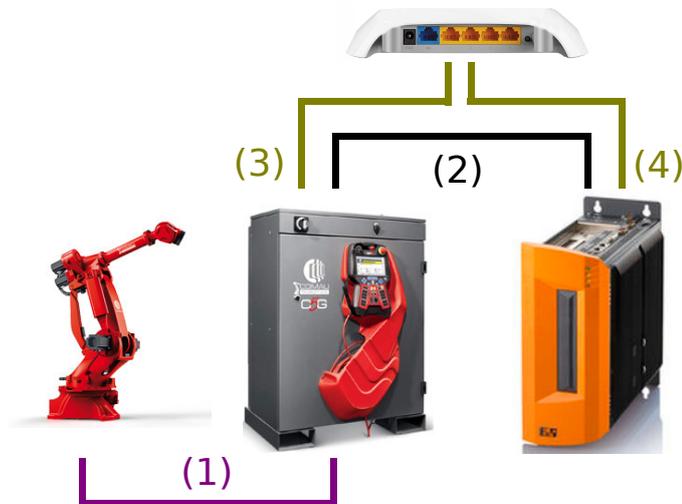


Figura 3. Estrutura padrão do sistema C5G Open

O OpenServer é um software do tipo servidor que permite o desenvolvimento de programas clientes que recebam por ele os dados sensoriais vindos da controladora do robô e envie através dele os comandos de movimentação para a mesma. No entanto, essa comunicação passará a ser feita usando uma rede *Ethernet* convencional, através do uso do protocolo TCP/IP. Isso permite que o programa cliente possa ser programado na linguagem de programação de preferência do usuário, rodando em uma arquitetura e sistema operacional disponíveis.

Para isso, o OpenServer foi programado dentro das limitações da plataforma, ou seja, na linguagem C++ usando a biblioteca *eORL* do fabricante do robô, usando a rede *PowerLink* para fazer a comunicação com a controladora do robô e sendo compilado para o sistema operacional baseado em Linux com arquitetura x86. Além disso, ele usa bibliotecas padrão em C++ disponíveis para o sistema operacional *Linux Mint* para realizar a comunicação via rede TCP/IP. A estrutura desenvolvida para execução do OpenServer passa então a ser conforme mostra a Figura 4, onde: (5) representa a conexão entre o roteador e o dispositivo externo, a qual, pode ser feita através de cabeamento ou conexão sem fio; e (6) é a conexão entre roteador e rede institucional.

A rede institucional do CEFET-MG oferece acesso à internet. No entanto, as portas do roteador são bloqueadas para acesso externo. Dessa forma, é criada uma rede interna segura onde os softwares são executados. Por esta razão, não foram implementadas soluções de criptografia de dados, bem como de autenticação. Isso ocorreu também pelo motivo de ser um ambiente acadêmico, controlado e sem tráfego de dados sensíveis. A estrutura real do sistema pode ser visto na Figura 5.

Estão disponíveis dois computadores para a execução de programas cliente, que podem ser vistos na Figura 6, conectados à rede interna.

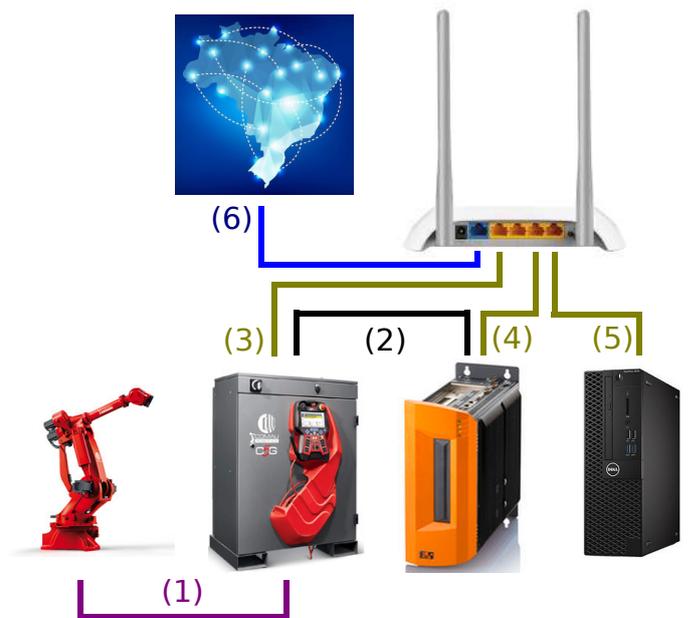


Figura 4. Estrutura desenvolvida para execução do OpenServer



Figura 5. Estrutura do laboratório onde será executado o OpenServer

Para utilizar o sistema *Open* é crucial a instalação da biblioteca *eORL* no *LPC* na mesma versão da *eORL* da controladora (Comau_Robotics, 2019). No caso, essa biblioteca já estava instalada no *LPC*, sendo necessária apenas a atualização dela para a última versão estável. Além disso, quando se desejar usar o sistema *Open*, é necessário habilitar o modo *Open* nos eixos desejados dentro do programa *Crcopen*, conforme indicado na Figura 7.

O desenvolvimento deste software torna mais segura a utilização do sistema *Open* em um ambiente de ensino e pesquisa, pois, o único software que precisará ser executado no *LPC* passa a ser o *OpenServer* em sua versão estável. Já no dispositivo externo serão executados os softwares experimentais, em desenvolvimento, os quais podem travar durante a execução. Quando isso acontece, o *OpenServer* continua sendo executado no *LPC* e comunicando com a controladora do robô, enviando a última referência armazenada. Dessa forma, o risco de se interromper bruscamente a comunicação com a controladora é evitado.



Figura 6. Estrutura do laboratório onde será executado o OpenServer

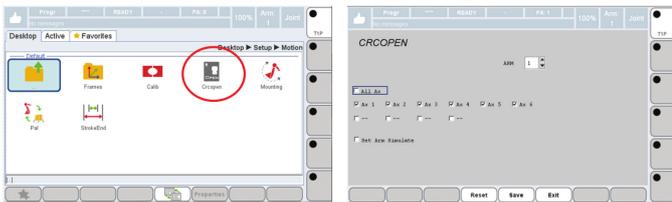


Figura 7. Procedimento para habilitar o modo Open dos eixos do robô (Comau_Robotics, 2019)

Além disso, o *OpenServer* expõe apenas um subconjunto das funcionalidades e configurações da *eORL*, simplificando a utilização e reduzindo a quantidade de erros possíveis.

2.1 Funcionamento

Foi implementado um modo *DEBUG*, a ser ativado antes do processo de compilação que imprime no terminal qual função foi executada, o comando vindo do programa cliente e a resposta enviada, conforme pode ser visto nas Figuras 8, 9 e 10. Quando o modo *DEBUG* não está ativado apenas a biblioteca *eORL* imprime textos no terminal, por exemplo, a tela de boas vindas, mensagens de erro, entre outros.

Quando o software é executado, ele tenta se conectar com a controladora usando a biblioteca *eORL* através da rede *Powerlink*. Quando a conexão é bem sucedida o software entra em modo *Listen*, ou seja, fica esperando um cliente se conectar ao *socket* através da rede TCP/IP via *Ethernet*, conforme pode ser visto na Figura 8.

Quando um cliente se conecta ao servidor é realizada uma troca de pacotes para verificar a integridade da conexão. Caso essa verificação seja bem sucedida o servidor fica esperando o recebimento de instruções do cliente, conforme pode ser visto na Figura 9. O servidor só atua mediante os comandos do cliente.

O primeiro caractere recebido pelo servidor informa qual procedimento ele deve seguir. No caso, a instrução 'p' é atualmente a instrução padrão. Ela está configurada para os valores de referência das juntas do robô (truncado em 5 casas decimais, mas, sem uso da vírgula para representar as casas decimais) virem concatenados na forma de

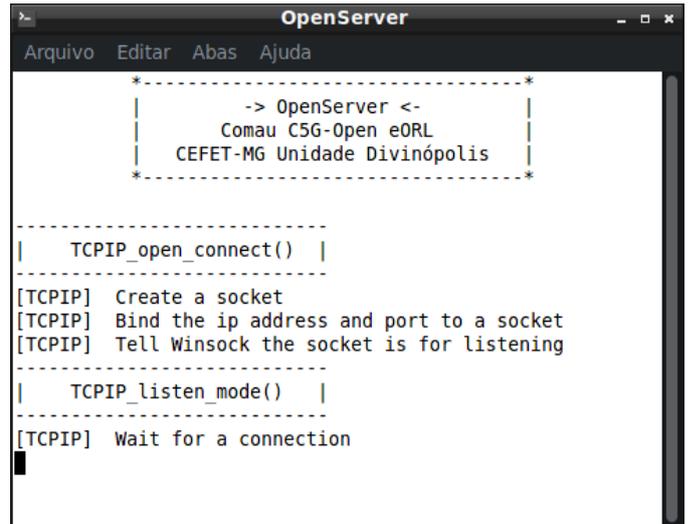


Figura 8. OpenServer esperando a conexão de um *software* cliente

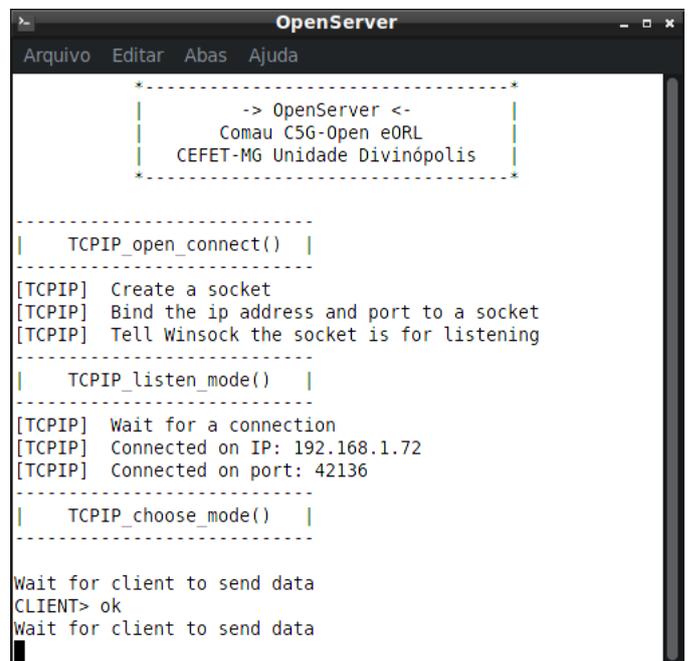


Figura 9. OpenServer esperando a instrução do *software* cliente conectado

strings logo após tal instrução, como pode ser visto na Figura 10. Ao receber essa instrução, imediatamente o servidor atualiza as variáveis de referência das juntas do robô na biblioteca *eORL*, as quais são enviadas para a controladora. Em seguida, também na forma concatenada sem uso de vírgula, o servidor envia a resposta ao programa cliente com a leitura mais recente dos sensores das juntas do robô armazenada na memória do servidor, as quais, foram recebidas da controladora aberta.

Esta forma de transmitir os dados (*strings* concatenadas) foi desenvolvida com a intenção de ser uma construção simples de ser implementada em outras linguagens de programação.

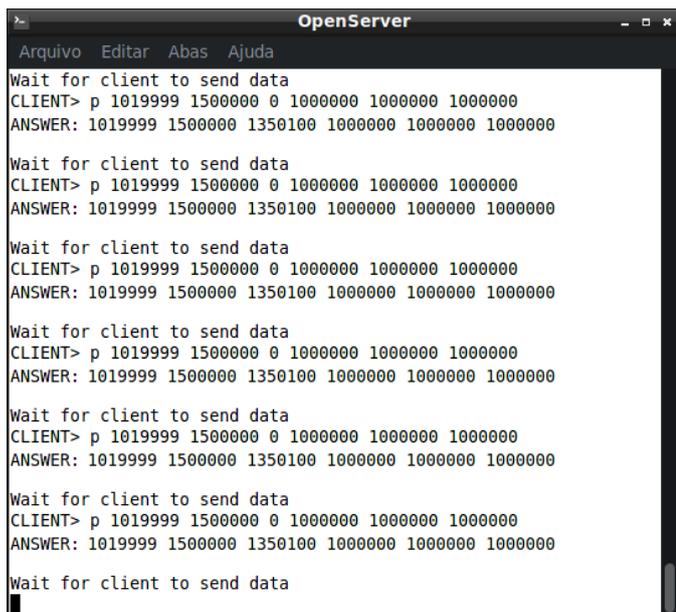


Figura 10. OpenServer executando a instrução do *software* cliente e respondendo com a leitura dos sensores

Como as taxas de transmissão de dados podem não ser necessariamente as mesmas, foi implementado um algoritmo interno que resolve esse problema de falta de sincronia. Por exemplo, a taxa de comunicação da *eORL* com a controladora varia de uma taxa de 0,4 ms a 16 ms; e a taxa de comunicação entre o programa servidor e o programa cliente não tem uma faixa definida. Assim, o desenvolvedor do programa cliente tem a liberdade de programar a taxa que for mais conveniente (dentro dos limites práticos da rede) e, inclusive, enviar os comandos de movimentação de forma não constante.

3. RESULTADOS E DISCUSSÕES

Para testar o servidor foi preciso desenvolver um *software* cliente que se comunique com ele via rede TCP/IP, o qual, permite o usuário enviar os ângulos de referência de cada junta do robô e visualizar as leituras dos sensores de posição angular (*encoder* absoluto de precisão) de cada junta em tempo real.

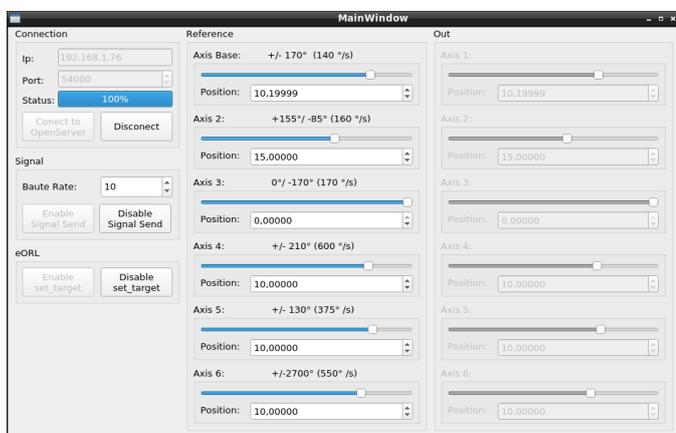


Figura 11. Software OpenClient em execução

Tal *software* foi chamado de *OpenClientExemple*, escrito em C++ usando bibliotecas padrão do sistema operacional baseado em Linux, mas, que pode ser compilado para a arquitetura *x86-64*, tendo sido executado em um computador pessoal convencional. Ele foi compilado também para *Arm64* e executado em um *Raspberry Pi 4*. Ambos testes foram realizados com sucesso, demonstrando que podem ser desenvolvidos programas clientes em outras arquiteturas.

Além disso, foi testada a comunicação com o *software Python*, usando a biblioteca padrão TCP/IP do mesmo, sendo executado no Sistema Operacional *Windows* e também testada com um dispositivo móvel rodando *Android*. A comunicação com o OpenServer foi realizada com sucesso, demonstrando que podem ser desenvolvidos *softwares* clientes em outros sistemas operacionais, inclusive para dispositivos móveis.

A taxa de comunicação de um pacote a cada 2 ms (um pacote contendo todas as leituras ou posições das juntas), que foi aferida via cabo pode ser o suficiente para a maioria das aplicações, mas, essa taxa pode ser melhorada desenvolvendo-se uma forma mais eficiente de transmitir os dados. Por exemplo, criando uma nova modalidade que, de forma síncrona, envie dados ao mesmo tempo que recebe, ao invés de usar a forma assíncrona, que aguarda o recebimento de um dado para enviar o próximo. Outra possibilidade seria criar outra modalidade em que sejam transmitidos os dados binários direto da memória RAM, ao invés de convertê-los em *strings* como foi feito nesse trabalho.

A comunicação via rede sem fio também ocorreu a uma taxa de 2 ms, com exceção de quando ocorria perda de dados devido à distância.

4. CONCLUSÕES E PERSPECTIVAS

O presente trabalho teve como foco a expansão das possibilidades de recursos da controladora aberta *C5G-Open* através da implementação de um sistema cliente servidor. Os resultados preliminares foram considerados satisfatórios, motivando a continuidade do desenvolvimento dos *softwares* e do projeto em si. A sub-rede demonstrou garantir a comunicação de forma estável e rápida. A integração dos *softwares* foi facilitada pela padronização do protocolo TCP/IP, o que ajudou expandir as possibilidades de uso do OpenServer.

A perspectiva futura é usar o OpenServer para dar início ao desenvolvimento de malhas de controle adicionais ao sistema robótico como uma malha de visão computacional. Uma câmera afixada no efetuador do robô filmará uma figura de referência, o *software* cliente processará a imagem e gerará os comandos de movimentação que serão enviados ao OpenServer, de forma a manter constante a pose (posição e orientação) do manipulador robótico em relação à pose da figura de referência. Dessa maneira, deverá ser adicionada uma câmera à estrutura desenvolvida, conforme pode ser visto na Figura 12, onde (1) representa a câmera; e (2) a figura de referência.

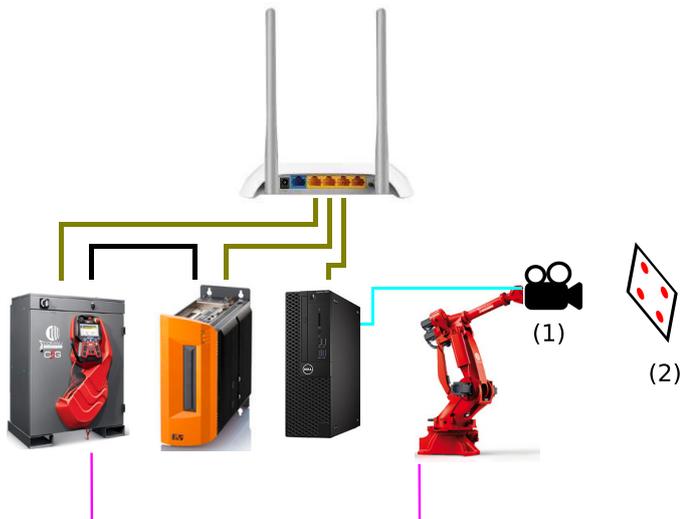


Figura 12. Estrutura proposta para utilização do Open-Server

5. AGRADECIMENTOS

Agradecemos ao CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico - pelo apoio financeiro dado a essa pesquisa através da concessão de bolsa do PIBTI - Programa Institucional de Bolsas de Desenvolvimento Tecnológico e Inovação. Agradecemos também a COMAU Robotics pelo suporte técnico e orientações dadas.

REFERÊNCIAS

- Antonelli, G., Chiaverini, S., Perna, V., and Romanelli, F. (2010). A modular and task-oriented architecture for open control system: the evolution of c5g open towards high level programming. *Workshop on the IEEE International Congress on Robotics and Automation, Lund University, Sweden*.
- Bisson, A. (2014). *Development of an interface for intuitive teleoperation of Comau manipulator robots using RGB-D sensors*. Master's thesis, Università degli Studi di Padova.
- Comau_Robotics (2010). *Control Unit C5G - Standard Version - Technical Specification*. Comau S.p.A.
- Comau_Robotics (2019). *C5G - Comau Open Controller - Instruction Handbook*. Comau S.p.A.
- Ferrara, V. (2013). *C5GOpen: The latest generation of the Industrial Robots Open Control System for University and SMEs*. Master's thesis, Politecnico di Torino.
- Kubus, D., N.K. and Johansson, S. (2010). Innovative robot control architectures for demanding (research) applications. *Workshop on the IEEE International Congress on Robotics and Automation, Lund University, Sweden*.
- Roberti, F., Soria, C., Slawinsk, E., Mut, V., and Carelli, R. (2010). Open software structure for controlling industrial robot manipulators. *In book: Robot Manipulators Trends and Development*.