Organização de uma rede *machine to machine* (M2M) para controle distribuído de um robô humanoide

Ana Cláudia Banderchuk*, Diesson Stefano Allebrandt †, Daniel Lohmann‡ e Renan Augusto Starke§
Departamento Acadêmico de Eletrônica, IFSC
Florianópolis – Brasil
Email: *ana.banderchuk@gmail.com, †diesson.floripa@gmail.com, ‡daniel.lohmann@ifsc.edu.br §renan.starke@ifsc.edu.br

***Resumo*—Um problema enfrentado no desenvolvimento de robôs móveis consiste na quantidade de sensores e atuadores que utilizam uma grande massa de condutores que pode ser reduzida através de redes de comunicação com ou sem fio. Assim, propõe-se a organização da estrutura de tópicos do protocolo de comunicação MQTT juntamente com o projeto de *hardware* e *software* para o controle da cinemática de um robô humanoide. Avaliou-se ainda os tempos de respostas do microcontrolador ESP8266 responsável pela implementação dos protocolos de rede em diferentes intervalos entre mensagens em um sistema operacional de tempo real (FreeRTOS).**

***Palavras-chave*—robô humanoide, redes para sistemas embarcadas, sistemas operacionais embarcados, Internet das coisas.**

I. Introdução

Um dos grandes desafios de sistemas automatizados e robotizados é a integração dos vários subsistemas de controle, atuação e sensoriamento. Estes sistemas devem comunicar-se gerando um modelo de computação distribuída, que pode ser de tempo real, necessitando de um sistema de rede adequado e de baixa latência. Atualmente, utiliza-se redes cabeadas com protocolos de comunicação já consolidados como CAN e *Ethernet*. Já no caso de robôs humanoides, o uso de uma rede cabeada pode ser um problema considerando a massa total das partes móveis. Há também a dificuldade da infraestrutura cabeada entre as juntas e ligamentos.

A problemática deste artigo está relacionada com a integração dos subsistemas de controle, atuação e sensoriamento utilizando-se de uma rede (sem fio, com fio e mista), aplicando-a a um robô humanoide. Os objetivos estão relacionados com a implementação da comunicação entre os atuadores e sensores responsáveis pela movimentação dos braços, mãos, pescoço e cabeça aliados ao sistema de visão computacional.

O projeto que está relacionado com este artigo está dividido em três eixos de desenvolvimento: estudo dos requisitos selecionando o tipo de rede adequado (banda, frequência, protocolos e requisitos de tempo real), seleção do sistema operacional adequado para cada tipo de nó, implementação ou adequação dos sensores e atuadores já existentes no robô e, finalmente, integração com o sistema de controle e avaliação do sistema adotando um modelo com computação distribuída.

No caso deste artigo, apresentar-se-á algumas contribuições iniciais relacionadas a:

- desenvolvimento de uma arquitetura orientada a serviço aplicada a um robô humanoide utilizando um modelo *publish/subscribe*;
- apresentação de testes de tempo de resposta aplicados ao sistema em um microcontrolador com rede sem fio WiFi (IEEE 802.11) e sistema operacional preemptivo (FreeRTOS).

Este artigo é divido entre as seções Introdução, Robô humanoide, Redes e a Internet das coisas, Estrutura e planejamento dos tópicos MQTT, Infraestrutura de implementação e experimentação, Testes de tempo de repostas e Conclusões e trabalhos futuros. A seção Robô humanoide apresenta uma breve descrição da aplicação "robô" e os projetos associados enquanto a seção Redes e a Internet das coisas descreve brevemente alguns dos protocolos utilizados e arquiteturas associadas a redes *Machine2Machine*. A seção Estrutura e planejamento dos tópicos MQTT apresenta a organização dos tópicos para o controle do robô e a Infraestrutura de implementação e experimentação descreve como foram implementados a arquitetura de controle do robô e os testes realizados. Por fim descreve-se os testes de tempo de resposta realizados e apresenta-se depois as Conclusões e possíveis trabalhos futuros.

II. Robô humanoide

Melhorar habilidades cognitivas de máquinas possui amplo interesse na comunidade científica e industrial principalmente na robótica, tanto na móvel quanto para os modelos humanoides [1]. O uso de sistemas autônomos como robôs móveis e sistemas articulados para manipulação de objetos são duas novas fronteiras do desenvolvimento tecnológico.

No contexto deste artigo, aplica-se a estrutura da rede em um robô humanoide InMOOV [2] em tamanho real, impresso em uma impressora 3D. Há um servo motor para cada movimento, como por exemplo, um para o movimento horizontal e outro para o vertical da cabeça. Cada braço contém cerca de dez servomotores. São necessários também sensores de realimentação da posição de cada movimento e proteção de sobrecarga e sobrecorrente para cada motor. Percebe-se assim, pela quantidade de sensores e atuadores, a necessidade de criação de um sistema para a implementação de uma rede com controle distribuído.

Na Figura 1 é apresentado um diagrama do robô, demonstrando o posicionamento dos servomotores e os respectivos movimentos.

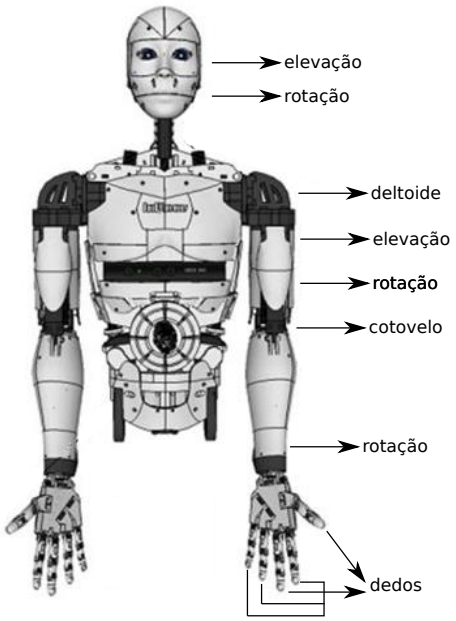

Figura 1. Mapeamento dos motores do robô humanoide InMOOV.

O robô humanoide é objeto de estudo para diversos subsistemas, entre eles destaca-se:

- visão computacional: identificação de objetos (*Deep Learning*) e estimação de distâncias;
- sensor táctil: percepção de toque em objetos;
- elaboração das matrizes de movimentação (cinemática robótica);
- **Rede e sistema de controle distribuído**.

Em termos de trabalhos correlatos, pode-se citar algumas publicações envolvendo robô humanoide em aplicações relacionadas a interpretação/ensino de linguagens de sinais e aprimoramento da programação de movimentos complexos. Em [3] mapeia-se os movimentos complexos do esqueleto humano utilizando-se de sensores *Kinnect* aplicando-os em matrizes cinemáticas do robô InMOOV. Em [4] é elaborada uma análise cinemática inicial para que um robô seja utilizado em aplicações de tradução entre linguagens falada e de sinais. Já em [5], as estruturas mecânicas do robô são reforçadas para aumento de confiabilidade. Até o conhecimento dos autores, os motores do robô humanoide em questão são controlados por microcontroladores de pequeno porte (8-bits) conectados por interface USB em um computador convencional. Não há uma arquitetura de rede de controle definida, o que propomos neste trabalho.

III. Redes e a Internet das coisas

Redes entre dispositivos são cada vez mais utilizadas em ambientes industriais [6], comerciais e residenciais para o suporte de aplicações de monitoramento. O uso de sistemas inteligentes comunicando-se através de uma rede está relacionado com a filosofia *Internet of Things* (IoT), que tem chamado atenção significativa à comunidade científica [7] e industrial. *IoT* é considerada como parte da Internet que é formada por bilhões de "coisas" comunicando-se inteligentemente.

No caso de robôs, há vários sensores que, aliados com a grande quantidade de atuadores responsáveis pela movimentação do robô (juntas, rodas, etc), formam um grande conglomerado de nós. Nós passivos podem ser modificados para serem inteligentes, formando uma rede de sensores e um sistema de controle distribuído. Quando utiliza-se um sistema de comunicação sem fio, pode-se facilmente reduzir a interconexão cabeada e consequentemente a massa do robô.

Considerando um sistema adaptativo, onde um novo nó pode integrar a rede para aprimorar ou criar uma nova funcionalidade, é interessante utilizar o conceito de uma Arquitetura Orientada a Serviço (SoA – *Service-oriented Architecture*) [7]. SoA assegura a interoperabilidade entre os sistemas heterogêneos consistindo em quatro funcionalidades principais:

- Camada de sensoriamento (sensores e atuadores): objetos de hardware que detectam e/ou mensuram grandezas físicas ou químicas no ambiente.
- Camada de rede: infraestrutura de comunicação com ou sem fio para implementar a comunicação entre os nós.
- Camada de serviço: cria e gerencia os serviços requisitados pelos usuários ou aplicações.
- Camada de interface: consiste de métodos de interação entre usuários ou aplicações.

SoA trata um sistema com um conjunto bem definido de objetos simples ou subsistemas que podem ser reutilizados e mantidos individualmente.

Em termos de protocolos utilizados em SoA, pode-se citar:

- CAN (*Controller Area Network*): protocolo de comunicação amplamente utilizado em aplicações industriais e automotivas.
- IEEE 802.11 (WLAN), IEEE 802.15.4(ZigBee), IEEE 802.15.1(Bluetooth): comunicação sem fio amplamente utilizada em computares e acessórios sem fio.
- IETF *Low power Wireless Personal Area Networks* (6LoWPAN): implementação do protocolo IPv6 sobre redes de pequeno porte.
- MQTT (*Message Queue Telemetry Transport*): protocolo leve de mensagens sobre a camada TCP/IP utilizando o conceito de *publish/subscribe*.

Deve-se também considerar uma plataforma de *software* para a implementação destes sistemas embarcados que promova suporte ao hardware e também suporte dos protocolos de comunicação. Dentro destes, pode-se considerar pequenos sistemas operacionais de tempo real como por exemplo o TinyOS [8], Contiki [9] e FreeRTOS [10]. Algumas plataformas específicas para desenvolvimento de aplicações IoT já possuem a implementação de alguns protocolos de comunicação por *hardware* ou incorporadas ao seu ambiente de desenvolvimento, como o ESP8266 [11], que integra uma solução de baixo consumo energético com WiFi integrado.

Realizando uma avaliação dos protocolos existentes, optou-

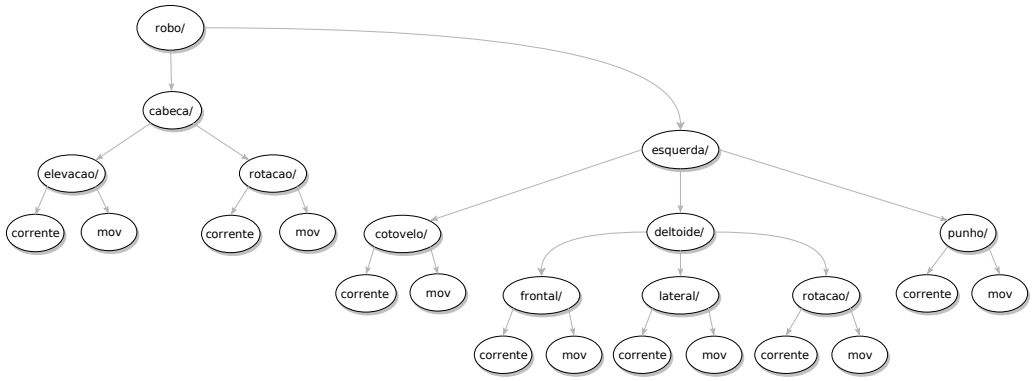

Figura 2. Planejamento dos tópicos MQTT para o controle do robô para as sub-árvores cabeça e braço esquerdo. Braço direito apresenta sub-árvore simétrica à da esquerda. Controle dos dedos está omitido, mas pertence a sub-árvore punho.

se pela utilização do MQTT [12]. O protocolo utiliza-se do modelo de Publicação/Subscrição – *Publish/Subscrie* – para a comunicação, por meio de um *Broker*. O *Broker* é o responsável por receber, enfileirar e enviar as mensagens dos clientes que publicam em determinado tópico x para os clientes subscritos neste tópico x. O cliente *publisher* é o responsável por se conectar ao *broker* e publicar as mensagens para um determinado tópico de interesse. E o cliente *subscriber* é responsável por se conectar ao *broker* e se "inscrever" em determinado tópico, para assim, receber todas as mensagens de interesse deste tópico publicadas por outros clientes.

Em termos de clientes MQTT para microcontroladores, desta-case o projeto *Eclipse Paho* [13] que apresenta implementação de clientes em várias linguagens de programação como Java, Python, C e C++. Há implementações adequadas para sistemas operacionais de tempo real como o FreeRTOS executados por microcontroladores de baixo poder computacional.

IV. Estrutura e planejamento dos tópicos do MQTT

Com base no sistema de tópicos do protocolo MQTT e o robô humanoide em questão, pode-se planejar a estrutura de tópicos da rede MQTT conforme a Figura 2. O primeiro tópico, definido como raiz, foi "robo/" que está relacionado como endereçamento inicial. Ele indexa informações vindas dos demais dispositivos conectados à rede. Relacionados ao tópico "robo/", encontram-se três subtópicos "esquerda/", "direita/" e "cabeca/", referentes ao braço esquerdo, braço direito e cabeça do robô respectivamente. O tópico "cabeca/" possui os subtópicos "elevacao/" e "rotacao/". Já o tópico "esquerda/" tem relação com o braço esquerdo possuindo os seguintes subtópicos: "cotovelo/", "deltoide/" e "punho/". O braço direito e os dedos das mãos estão omitidos na Figura 2 para melhorar a visualização do diagrama.

Os tópicos "cotovelo/", "elevacao/" e "rotacao/", por exemplo, possuem como subtópicos "corrente/" e "mov/", sendo "mov/" referente aos clientes responsáveis pelo controle de elevação e rotação dos seus respectivos membros. Esse tópico recebe as publicações vindas do controlador principal, para a comunicação com os seus periféricos. Como por exemplo, caso solicitado que a cabeça seja levantada em 45º, essa informação será enviada ao tópico "robo/cabeca/elevacao/mov/", pois apenas o servomotor referente ao deslocamento vertical da cabeça deve ser acionado. Já o tópico "corrente/" é utilizado para a publicação da corrente dos servomotores utilizados para a movimentação do robô bem como estado e erros relacionados às proteções de sobrecarga ou sobrecorrente de determinado motor.

Além da organização e padronização dos movimentos do robô, a estrutura de tópicos apresentada pela Figura 2 foi criada para orientar a aplicação "robô" a uma Arquitetura Orientada a Serviço (SOA). Conforme os nós responsáveis pelo movimento de cada membro vão inserindo-se na rede, a aplicação de controle é informada e um novo "serviço" é configurado. Por exemplo, após a inicialização do cliente responsável pela movimentação do deltoide, a aplicação de controle pode agora realizar movimentos deste membro adicionando um novo "serviço" ao robô. Procedimentos semelhantes são realizados para cada serviço como as respectivas movimentações, visão computacional e futuros sensores tácteis.

V. Infraestrutura de implementação e experimentação

A infraestrutura de implementação e experimentação deste trabalho é dividida entre projetos de *hardware* e *software*. A interação entre vários componentes é ilustrada pela Figura 3.

Considerando o *hardware*, pode-se destacar:

- Robô humanoide InMOOV. Projeto para impressão 3D disponível em [2].
- Servomotores analógicos (movimentação e controle de posição por modulação por largura de pulso – PWM).
- Cartões de circuito impresso desenvolvidos pelo autores:
  - Controle dos dedos das mãos: microcontrolador ESP8266 com circuitos de leitura e proteção de corrente para cinco servomotores.
  - Controle dos demais movimentos: microcontrolador ESP8266 implementando apenas os protocolos de

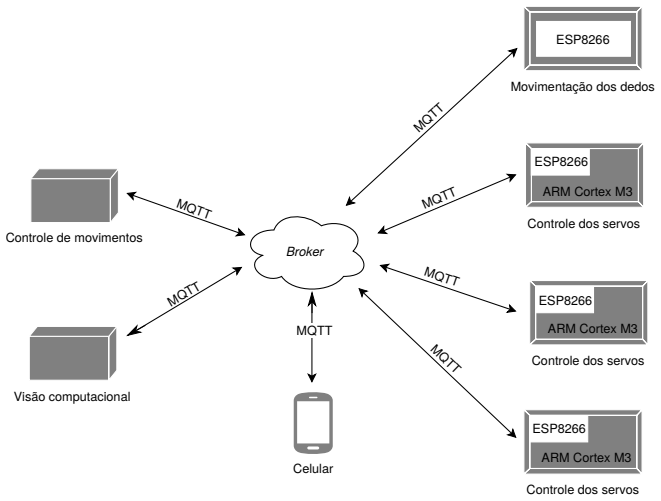

Figura 3. Interação entre os componentes para o controle do robô.

rede WiFi/MQTT (*Network Processor*) e microcontrolador STM32F103 (ARM Cortex M3) para controle de servomotores e proteções de corrente.

Os cartões responsáveis pela movimentação dos dedos das mãos possuem limitação quanto ao espaço, os servomotores são de baixa potência e as restrições de tempo real do *firmware* de controle são mais brandas comparadas aos outros membros do robô. Desta forma, optou-se por utilizar apenas o microcontrolador ESP8266 com WiFi integrado para estas movimentações e recepção dos comandos por protocolo MQTT.

No caso dos outros movimentos do robô, utilizou-se o microcontrolador ESP8266 apenas para implementação dos protocolos de rede e um STM32F103 responsável pela controle e proteção dos servomotores. Essa estratégia foi adotada pois o ESP8266 não possui *hardware* de PWM adequado (apenas por *software*) enquanto o STM32F103 apresenta inúmeros canais PWMs por *hardware*. Além disso, a maior versatilidade do STM32F103 pode ser utilizada para padronização de outros movimentos ou implementação de um controle e proteção mais sofisticados dos motores. Na comunicação entre os dois microcontroladores utilizou-se o *hardware* de comunicação assíncrona – UART (*Universal Asynchronous Receiver-Transmitter*) – com um protocolo desenvolvido pelos autores.

No caso do *software* de controle e proteção, foram adotadas as seguintes estratégias de implementação:

- Cartão dos dedos: *firmware* utiliza o FreeRTOS (escalonamento preemptivo) com as seguintes tarefas:
  - Gerenciamento do Wifi: *wifi_task*.
  - Publicação do estado: *beat_task*.
  - Gerenciamento do MQTT: *mqtt_task*.
  - Proteção dos motores: *prot_task*.
- Cartão dos demais movimentos: *firmware* implementado em C++ para STM32F103 e *firmware* com FreeRTOS (escalonamento preemptivo) para o ESP8266 com as seguinte tarefas:
  - Gerenciamento do Wifi: *wifi_task*.
  - Publicação do estado: *beat_task*.
  - Gerenciamento do MQTT: *mqtt_task*.
  - Gerenciamento do hardware UART: *uart_task*.
  - Gerenciamento dos pacotes UART: *pkgParser_task*.

Quando algum cliente publica um mensagem em um tópico de movimentação específico, o *broker* envia a mensagem para o cliente subscrito ao tópico. A interação entre os componentes de *software* é ilustrado pelo diagrama de sequência da Figura 4. Há a interação entre duas interfaces de comunicação: a WiFi (conexão entre o *broker* e o cliente MQTT do ESP8266) e UART (conexão entre o ESP8266 e o STM32F103). A movimentação efetiva do motor é sujeita as proteções sobrecarga e sobrecorrente no motor que são implementadas pelo STM32F103 (ARM).

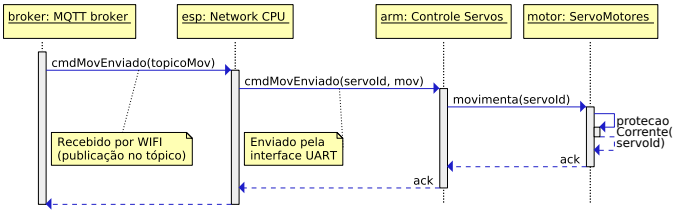

Figura 4. Diagrama de sequência: publicação de comandos para robô.

No caso do valor da corrente, o ESP8266 envia comandos UART para o STM32F103 que reporta as informações solicitadas. O valor da corrente é então publicados pelo ESP8266 no tópico específico do robô. Essa interação é ilustrada pelo diagrama de sequência da Figura 5.

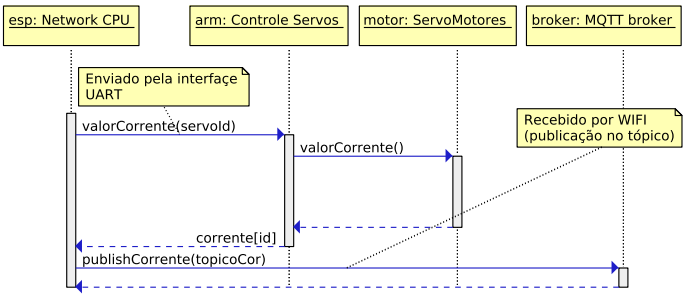

Figura 5. Diagrama de sequência: publicação do valor de corrente dos motores.

VI. Testes de tempo de resposta

Tendo definido a estrutura dos tópicos e a implementação básica do sistema de controle do robô, trabalhou-se também em uma análise de latência da aplicação em conjunto com a implementação do protocolo MQTT no microcontrolador ESP8266. Ensaios sobre o tempo de resposta da aplicação são importantes para estimar a latência entre o envio de um comando de movimento até a efetiva execução no atuador. Além disso, é necessário estimar parâmetros de sobrecarga da rede: quantidade máxima de mensagens, bem como mínimo intervalo entre elas.

Existem dois principais *kits* de desenvolvimento – *Software Development Kit* (SDK) – para o microcontrolador ESP8266: com escalonamento preemptivo que utiliza o FreeRTOS [14] e com com escalonamento cooperativo [15] baseado no Arduino. Por questões de versatilidade e capacidade de expansão, optou-se por utilizar o FreeRTOS neste trabalho. Também avaliou-se o desempenho do Arduino mas, por questões de espaço, não foram apresentados neste artigo.

Avaliou-se o tempo de resposta em cinco cenários: mensagens enviadas a cada 10s, 5s, 1s, 500ms e 100ms. Para cada cenário, foram enviadas 200 mensagens e calculado o tempo de resposta.

Para ter um melhor controle dos testes, utilizou-se de uma rede WiFi isolada e implementou-se duas novas aplicações. Uma delas, em linguagem C++ e executada por um computador PC no sistema operacional Linux, é responsável por publicar um dado no tópico "teste/" e aguardar o recebimento de uma resposta no tópico "node/" medindo o tempo desta transação. Esta mesma máquina executa o *broker* Mosquitto. A outra aplicação é um *firmware* para o microcontrolador ESP8266 que inscreve-se no tópico "teste/" e publica imediatamente um dado no tópico de resposta "node/". A Figura 6 ilustra a interação entre a aplicação C++, o *broker* e ESP8266.

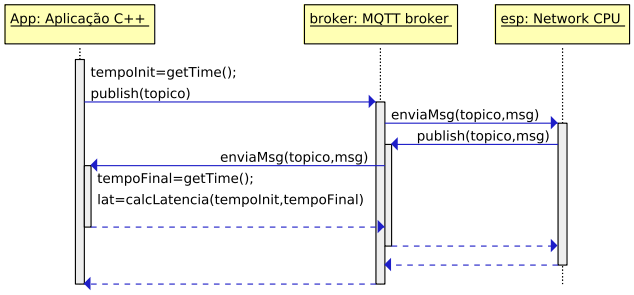

Figura 6. Sequência de interação entre os componentes para a avaliação do tempo de resposta.

Os resultados do tempo de resposta são apresentados pelas Figuras 7, 8, 9, 10 e 11 bem como suas estatísticas tempo médio ($\bar{t}$), desvio padrão (σ_t) e a porcentagem de falhas para cada cenário. Caracteriza-se uma falha quando a aplicação publica no tópico "teste/" mas não há retorno no tópico "node/".

No caso destes testes com o FreeRTOS, não foram detectadas falhas, porém certas mensagens podem ter um tempo de resposta muito mais elevado caracterizando interferências da rede e do escalonamento das tarefas do FreeRTOS. Sabe-se que a implementação com sistema operacional preemptivo exige uma complexidade adicional relacionada com o escalonamento, sincronização e proteção de dados entre as tarefas. Utilizando um intervalo de $50ms$ entre mensagens, a implementação com FreeRTOS mostrou-se incapaz de responder as 200 mensagens. Porém, os tempos mostram-se adequados para aplicação do robô pois as constantes de tempos dos servomotores são muito maiores que os tempos de respostas verificados.

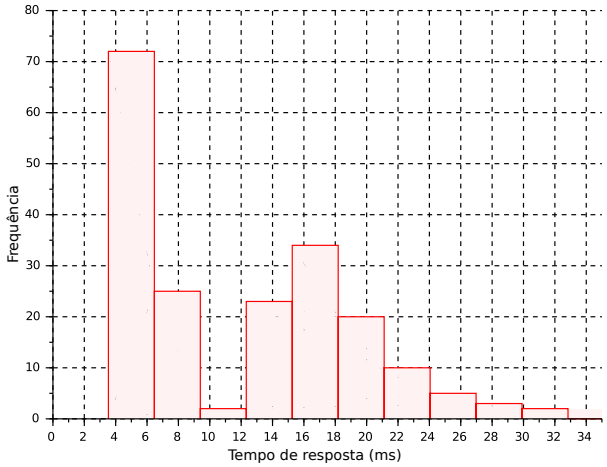

Figura 7. FreeRTOS: mensagens em 10s. Falhas = 0% $\bar{t} = 15m$ $\sigma_t = 29m$.

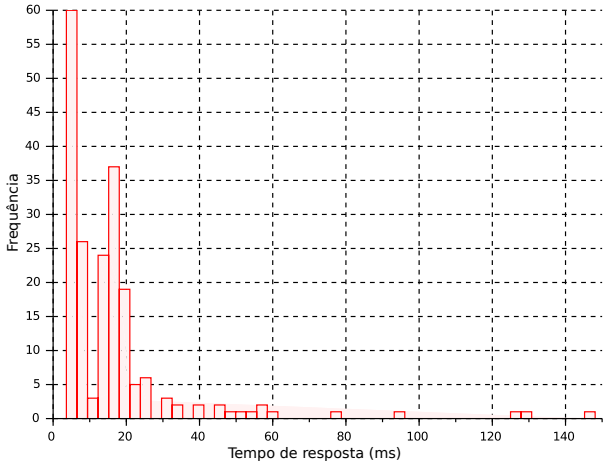

Figura 8. FreeRTOS: mensagens em 5s. Falhas = 0% $\bar{t} = 16m$ $\sigma_t = 19m$.

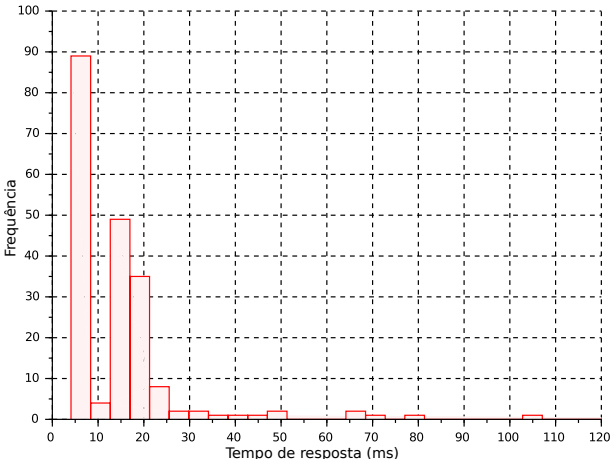

Figura 9. FreeRTOS: mensagens em 1s. Falhas = 0% $\bar{t} = 15m$ $\sigma_t = 19m$.

VII. Conclusões e Trabalhos Futuros

Este artigo trata da organização de um rede entre dispositivos de controle a atuação aplicados a um robô humanoide. A organização desta rede foi realizada utilizando o protocolo

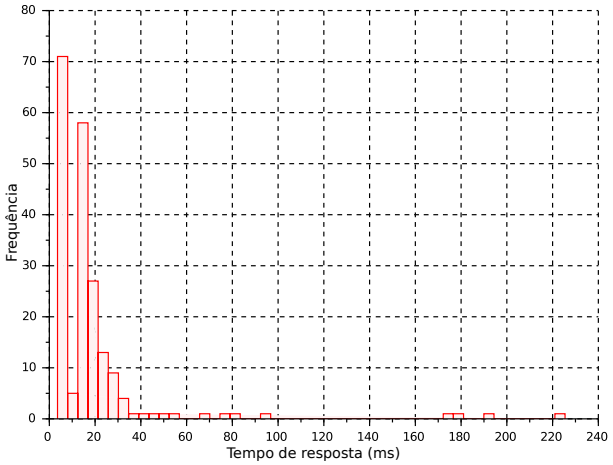

Figura 10. FreeRTOS: mensagens em 500ms. Falhas: 0% $\bar{t} = 19m$ $\sigma_t = 28m$.

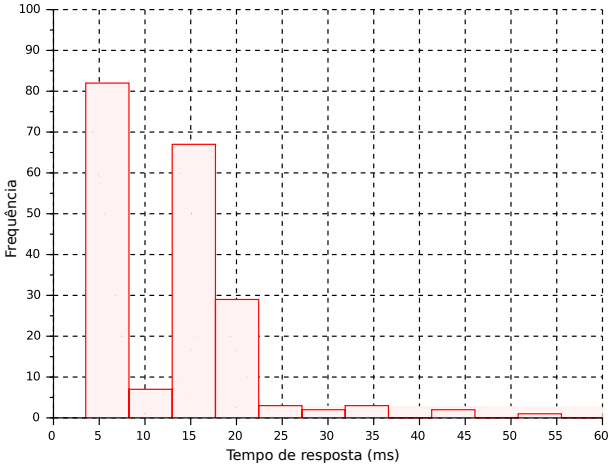

Figura 11. FreeRTOS: mensagens em 100ms. Falhas: 0% $\bar{t} = 50m$ $\sigma_t = 25m$.

MQTT que utiliza o modelo de publicação/subscrição com a criação de vários tópicos específicos e inter-relacionados que permitem a movimentação dos membros do robô bem como inspeção de estados de motores e sensores. Pode-se relacionar um motor ou um sensor para cada nó de rede ou agrupá-los desde que se respeite a organização dos tópicos. Conclui-se que a utilização do modelo de publicação/subscrição permite a fácil expansão dos "serviços" (movimentos, etc) e que tal organização pode ser aplicada em várias aplicações além do robô seguindo a filosofia da "Internet das coisas".

Testou-se também o desempenho do SDK com FreeRTOS para o microcontrolador ESP8266 que é amplamente utilizados na construção de pequenos sensores e atuadores. Percebe-se a escolha do SDK depende da classe da aplicação. Aplicações mais sofisticadas podem necessitar um maior controle e aproveitamento do microcontrolador e para tal deve-se utilizar um sistema operacional preemptivo possivelmente reduzindo latências e promovendo mair versatilidade da aplicação. No caso de aplicações mais simples que o desempenho é um elemento crítico, o SDK Arduino pode mostrar-se mais adequado, pois não há interferência do sistema operacional.

Como este robô é objeto de estudo para diversos subprojetos como visão computacional, sensores tácteis e cinemática robótica, pretende-se expandir a capacidade do robô com outros projetos utilizando a infraestrutura de rede proposta neste trabalho. Além disso, estudar-se-á quais elementos do FreeRTOS/ESP8266 que podem ser otimizados para melhorar ainda mais o desempenho.

Todos os fontes deste trabalho estão disponíveis no repositório https://github.com/xtarke/daeln_robot.

VIII. Agradecimentos

Os autores agradecem ao Instituto Federal de Santa Catarina pelo apoio financeiro.

Referências

[1] G. Sandini, G. Metta, and D. Vernon, *The iCub Cognitive Humanoid Robot: An Open-System Research Platform for Enactive Cognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 358–369. [Online]. Available: https://doi.org/10.1007/978-3-540-77296-5_32

[2] G. Langevin. (2018, Jul.) InMoov Project Page. [Online]. Available: http://www.inmoov.fr/project/

[3] X. Li, H. Cheng, G. Ji, and J. Chen, "Learning complex assembly skills from kinect based human robot interaction," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, dec 2017. [Online]. Available: https://doi.org/10.1109/robio.2017.8324818

[4] M. Campos-Trinidad, J. A.-D. Carpio, E. Lopez-Zapata, and R. Salazar-Arevalo, "Optimal control of a robotic system and software development for speech-to-sign language transliterating applications," in *2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*. IEEE, aug 2017. [Online]. Available: https://doi.org/10.1109/intercon.2017.8079690

[5] F. Acuna, M. Singana, F. Onate, V. Valdes, and M. Bustillos, "Humanoid interpreter for teaching basic sign language," in *2016 IEEE International Conference on Automatica (ICA-ACCA)*. IEEE, oct 2016. [Online]. Available: https://doi.org/10.1109/ica-acca.2016.7778503

[6] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18 – 23, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221384631400025X

[7] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, Apr 2015. [Online]. Available: https://doi.org/10.1007/s10796-014-9492-7

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGOPS Oper. Syst. Rev.*, vol. 34, no. 5, pp. 93–104, Nov. 2000. [Online]. Available: http://doi.acm.org/10.1145/384264.379006

[9] C. P. Team. (2018, Jul.) Contiki: The Open Source OS for the Internet of Things. [Online]. Available: http://www.contiki-os.org

[10] F. P. team. (2018, Jul.) The FreeRTOS Kernel: Market Leading, De-facto Standard and Cross Platform RTOS kernel. [Online]. Available: http://www.freertos.org/

[11] E. Systems. (2018, Jul.) ESP8266: Low-power, highly-integrated Wi-Fi solution. [Online]. Available: https://espressif.com/en/products/hardware/esp8266ex/overview

[12] (2018, Jul.) MQTT Standard and Project Page. [Online]. Available: http//mqtt.org/

[13] P. P. team. (2018, Jul.) Eclipse Paho: An open-source client implementations of MQTT. [Online]. Available: http://www.freertos.org/

[14] S. Automation. (2018, Jul.) A community developed open source FreeRTOS-based framework for ESP8266. [Online]. Available: https://github.com/SuperHouse/esp-open-rtos

[15] E. Corporation. (2018, Jul.) Espressif ESP8266 non OS SDK. [Online]. Available: https://github.com/esp8266/Arduino