Protocolo Websocket em Hardware Restrito para Redes de Sensores sem Fio

Article · October 2019

CITATIONS
0

READS
22

1 author:

Elany Marinho Branches Farias
Federal University of Minas Gerais
5 PUBLICATIONS **1** CITATION

SEE PROFILE

Protocolo Websocket em Hardware Restrito para Redes de Sensores sem Fio

Elany Marinho Branches Farias*. Flávio H.Vasconcelos.**

**Escola de Engenharia,,Universidade Federal de Minas Gerais;(e-mail: elany7@gmail.com).*
***Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais (e-mail: fvasc58@gmail.com)*

Abstract: This paper reports on the evaluation performed with the Websocket protocol for use in the application layer of a Wireless Sensor Network, contrasting it with the HTTP protocol. The sensor node is composed of the restricted ESP8266 NodeMCU hardware fitted together with an MPU6050 inertial sensor module composed of gyroscope and accelerometer. The protocols were tested and analyzed quantitatively in the same scenarios and network conditions. The results obtained presented the Websocket protocol as a suitable choice for this type of application.

Resumo: Este artigo faz um relato da avaliação realizada com o protocolo Websocket para uso na camada de aplicação de uma Rede de Sensores Sem Fio, contrastando-o com o protocolo HTTP. O nó sensor é composto pelo hardware restrito ESP8266 NodeMCU equipado juntamente com um módulo sensor inercial MPU6050 composto por giroscópio e aceleromêtro. Os protocolos foram testados e analisados de forma quantitativa nos mesmos cenários e condições de rede. Os resultados obtidos apresentaram o protocolo Websocket como uma escolha adequada para esse tipo de aplicação.

Keywords: Wireless sensor networks; application layer; websockets; hardware; protocol.

Palavras-chaves: Redes de sensores sem fio; camada de aplicação; websockets; hardware; protocolo.

1. INTRODUÇÃO

Os primeiros trabalhos relatando o uso de Redes de Sensores sem Fio (RSSF) ou *Wireless Sensor Networks (WSN)*, em inglês, foram publicados no início da década de 1980 e foram decorrentes da utilização dos dispositivos denominados nós sensores que unem tecnologias de comunicação sem fio, processamento, memória e sensoriamento. E como aconteceu com outras tecnologias, as RSSF também tiveram como origem a área militar datando por volta de 1978 pela DARPA (Defense Advanced Research Projects Agency).

No entanto, o aumento significativo no interesse pela comunidade de pesquisa e pelas indústrias por estas redes foi observado a partir de 2001, principalmente devido aos avanços tecnológicos voltados para projetos de sensores, tecnologias da informação e redes sem fio (Katsikeas, 2016). Dentre as vantagens que as RSSF trouxeram, está a redução de riscos e custos em implementação de sistemas que utilizavam cabeamento, além da expansão de suas áreas de domínio.

Os nós sensores são dispositivos autônomos e aptos a interagirem em seu ambiente através de sensores e/ou atuadores e, assim, controlar parâmetros do local monitorado (Karl e Willing, 2005). Cada nó sensor realiza a coleta dos dados, sendo capaz de processa-los localmente e disseminar as informações para um elemento conhecido como *Sink Node* ou Estação Base (BS – *Base Station*). A capacidade de cada nó sensor pode variar do simples que monitora apenas um parâmetro para um nó sensor que combina vários tipos de fenômenos. Além disso, a tecnologia de comunicação também pode variar, de infravermelho ou frequências de rádio, por exemplo (Dargie e Poellabauer, 2010).

Com o crescimento dessas redes e as amplas possibilidades de uso que apresentam, é necessário entender a importância da flexibilidade de suas aplicações não apenas em cenários críticos, mas também em cenários não críticos (Pereira, 2016). As RSSF possuem uma notável diversidade de categorias de aplicações como ambientes de monitoramento ambiental, de atividades humanas, em transporte, na agricultura, na saúde, dentre outros.

De acordo com Pereira *et al* (2011), uma RSSF pode ser utilizada para obter e disseminar informações sobre fenômenos físicos e biológicos, fornecendo uma interface entre o mundo físico e o mundo virtual. Para alcançar tal objetivo é necessário que a arquitetura da RSSF esteja de acordo com as necessidades da aplicação desejada. A arquitetura básica para este tipo de rede está dividida em três blocos, sendo: (1) Infraestrutura, (2) Protocolo de rede e (3) Protocolo de Aplicação.

A pilha de protocolos utilizada em um projeto de uma RSSF depende do objetivo de cada aplicação, em face das restrições e padrões necessários, e podem ser mapeadas em um modelo de comunicação como TCP/IP ou modelo *Open System Interconnection* (OSI). Tendo por base o modelo adotado, dentre as várias camadas que o software de rede se organiza, a camada de aplicação se destaca, pois, é a mais

próxima do usuário final. No que refere a RSSF, esta camada atua no processamento da informação, formatação e armazenamento dos dados.

Este artigo trata da avaliação e comparação através da implementação de dois protocolos para a camada de aplicação em uma RSSF. Os testes visaram um cenário em que a RSSF trabalha em um ambiente *indoor* (uma rede de exemplo) na qual o consumo de energia não é uma restrição, uma vez que a troca por outra fonte de energia seria de fácil acesso. Além disso, é utilizado o protocolo IEEE 802.11 na camada física para transmissão dos dados. Esse tipo de cenário pode ser utilizado em aplicações que necessitam de uma aquisição de dados como, por exemplo, voltado para área médica ou industrial.

No segundo e terceiro tópico revisa-se sobre a arquitetura e protocolos para camada de aplicação voltados para RSSF, respectivamente. No quarto tópico é descrito a configuração projetada para os experimentos e seus resultados. E por fim, no quinto tópico a conclusão.

2. TRABALHOS RELACIONADOS

Na literatura encontram-se trabalhos como em Kosanovic e Kosanovic (2017) em que os autores realizaram uma análise comparativa com as variações do protocolo HTTP, *polling*, *long polling* e *streaming* com Websocket. Os resultados apresentados pelos autores comprovaram a eficiência do protocolo Websocket com o menor *overhead* na troca de pacotes, reduzindo o *delay* no fornecimento de informações e redução do consumo de energia, o que ocasionou na melhora significativa do tempo de vida das baterias de cada nó sensor.

Ma e Sun (2013) realizaram um estudo do monitoramento em tempo real com uma RSSF com Websocket para edifícios inteligentes remotos. Esses autores analisaram comparativamente os métodos de abordagem HTTP *polling*, Socket in ActiveX, FlashSocket e Websocket. Os resultados obtidos confirmaram o melhor desempenho do Websocket sobre os protocolos citados através da menor média de tempo de latência apresentada perante os testes realizados.

Em um outro trabalho, Mijovic, Shehu e Buratti (2016) avaliaram três protocolos de camada de aplicação via experimentação com cenários voltados para Internet of Things (IoT). Os protocolos escolhidos foram escolhidos visando a fácil implementação em um dispositivo acessível buscando uma comparação justa e realística. Foram considerados como parâmetros avaliativos a eficiência do protocolo e a média RTT (*Round Trip Time*). Os cenários propostos foram uma LAN e uma rede IoT. Os autores chegaram à conclusão que o RTT médio é maior para a rede IoT e que dentre os protocolos avaliados (CoAP, MQTT QoS 0 e QoS 1 e Websocket), o protocolo CoAP alcançou a melhor eficiência e MQTT QoS 1 o pior resultado. 3.

3. PROTOCOLO WEBSOCKET

Documentada na RFC 6455 em 2011 pela *Engineering Task Force* (ETF), o protocolo *Websocket* foi projetado para substituir tecnologias de comunicação bidirecional que utilizam HTTP. Dessa forma, permitindo que esse mecanismo uma vez executado em aplicações web com o servidor utilize o protocolo TCP na camada de transporte. Além disso, o mesmo foi projetado para ser utilizado em navegadores web, mas também permitindo o uso com diferentes propósitos.

O protocolo Websocket permite o envio de mensagens que pode conter texto com codificação UTF-8 ou dados binários. O menor overhead detectado é devido ao cabeçalho projetado para ser curto para a redução do consumo de largura de banda.

A API (*Application Programming Interface*) disponibilizada define as especificações para aplicações web. É baseada em evento e de fácil implementação. Os quatro eventos que compõem a APi, são descritos na Tabela I (Srinivasan *et al*, 2013):

Tabela I. Eventos API Websocket

Evento	Descrição
onopen	Chamado quando a conexão Websocket é estabelecida.
onmessage	Chamado quando uma mensagem é recebida.
onclose	Chamado quando a conexão Websocket é fechada devido ao recebimento de handshake ou falha.
onerror	Chamado quando ocorre erro devido a memória buffer cheia.

O funcionamento básico consiste em estabelecer uma conexão entre clientes e servidores inicialmente com cabeçalho HTTP utilizando um único socket TCP. Após conectado, é removido o cabeçalho HTTP e trocado para o cabeçalho *Websocket*, permitindo a simultaneidade na troca de mensagens em ambas as direções, de modo a oferecer uma conexão persistente e dedicada. Isso significa que o servidor também tem permissão de enviar mensagens para o cliente a qualquer momento (Figura 1). A confiabilidade é garantida utilizando mecanismos próprios.

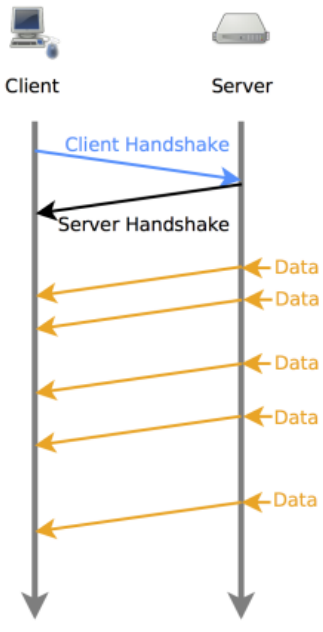

Figura 1. Protocolo Websocket (Karadogan, 2013)

O mecanismo de atualização do cabeçalho é apresentado na Figura 2, na qual aponta o estabelecimento do primeiro *handshake* como a requisição para o servidor. A atualização indica que deverá ser mudado para protocolo *Websocket* na linha 3.

```
1 GET /chat HTTP/1.1
2 Host: server.example.com
3 Upgrade: websocket
4 Connection: Upgrade
5 Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
6 Origin: http://example.com
7 Sec-WebSocket-Protocol: chat, superchat
8 Sec-WebSocket-Version: 13
```

Fig 2. Requisição para o servidor (Karadogan, 2013)

```
1 HTTP/1.1 101 Switching Protocols
2 Upgrade: websocket
3 Connection: Upgrade
4 Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
5 Sec-WebSocket-Protocol: chat
```

Fig 3. Resposta do servidor (Karadogan, 2013)

O segundo *handshake* (Figura 3) apresenta a resposta do servidor com o código 101 indicando a devida atualização do campo *Upgrade* para *Websocket*. O campo *Sec-** indica a confirmação do servidor perante a requisição. Uma vez estabelecida a conexão, o *Websocket* torna-se um canal *full duplex* entre o cliente e o servidor (Karadogan, 2013).

4. CONFIGURAÇÃO DE TESTE

A RSSF utilizada como base para a realização dos testes tem como características ser uma aplicação para ambiente *indoor*, especificamente em um laboratório. Trata-se de uma rede desenvolvida com o propósito de analisar a marcha humana utilizando sensores inerciais juntamente com o hardware ESP8266 NodeMCU.

A estação de trabalho é composta por: (a) um notebook Samsung Intel Core i3 7th geração, 4GB memória RAM com sistema operacional Windows 10 e (b) nós sensores compostos pelo módulo MPU 6050 e ESP8266 NodeMCU.

É importante ressaltar que o consumo de energia não foi um parâmetro avaliado neste trabalho devido aos requisitos da aplicação, e às condições específicas de rede, uma vez que, neste caso, a troca de baterias da fonte alimentação não constitui um problema.

A aplicação é fundamentada no modelo de arquitetura TCP/IP empregando quatro camadas lógicas segundo Tanenbaum (2011), sendo estas as camadas de: aplicação, transporte, rede e física.

Os nós sensores se conectam a uma rede sem fio local utilizando como protocolo de camada física a estrutura IEEE 802.11. Para o endereçamento utiliza-se *Internet Protocol* (IP) na camada de rede e o *Transmission Control Protocol* (TCP) na camada de transporte. Já na camada de aplicação foram empregados dois protocolos para os testes: *HyperText Transfer Protocol* (HTTP) e *Websocket*.

Os nós sensores são formados por um hardware restrito SoC (*System on Chip*) microcontrolador NodeMCU com módulo ESP8266 que fornece a função de conectividade sem fio embutida (IEEE 802.11 b/g/n) de baixo custo, acesso aos 16 GPIOs (*General Purpose Input/Output*), comunicação I2C (*Inter-Intergrated Circuit*), UART (*Universal Asynchrounous Receiver/Transmiter*), SPI (*Serial Peripheral Interface*) e um canal analógico. A alimentação fica por conta de 3,3v e até 256mA.

Em adição, o nó sensor também é composto por um módulo MPU 6050 que integra sensores inerciais de Acelerômetro e Giroscópio através do protocolo *Inter-Integrated Circuit* (I_2C) em que os dados coletados são convertidos em sinais digitais e enviados para a placa (Mota, 2017). A configuração de teste é ilustrada na figura 4.

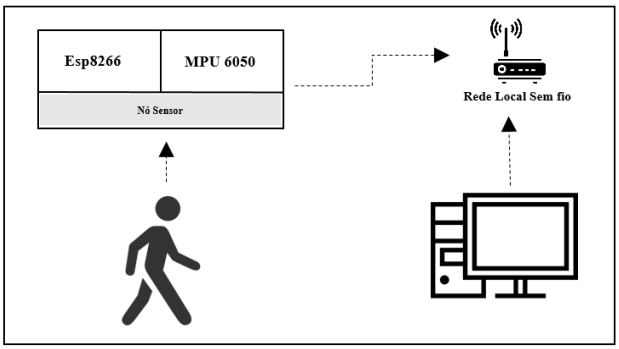

Fig 4. Configuração do experimento

Os protocolos de comunicação foram implementados no microcontrolador e a estação base serviu como interface para o recebimento dos dados coletados pelo nó sensor através dos protocolos de aplicação definidos: HTTP e *Websocket*.

5. RESULTADOS EXPERIMENTAIS

Os testes foram realizados com a conexão do nó sensor e da estação base no mesmo ponto de acesso. A estação base solicita a conexão com o nó sensor, e, uma vez confirmado, ocorre o envio dos dados coletados pelo nó para a estação. Para comparar o desempenho dos dois protocolos, os testes foram realizados de forma quantitativa.

Para o desenvolvimento deste ensaio foram considerados dois cenários de testes: o primeiro (1) realizou a coleta de 80 amostras para variados tamanhos de *payload*, sendo estes: 20 bytes, 50 bytes, 100 bytes, 200 bytes e 255 bytes. Foram calculadas as medidas estatísticas como média, mediana, mínimo, máximo e desvio padrão para cada protocolo e seu tamanho de *payload* específico. Esses valores são apresentados nas tabelas II e III. O segundo cenário (2) tratou-se da coleta dos dados em um período de tempo de três minutos, executado duas vezes para cada protocolo com um valor fixo de *payload* de 32 bytes.

As métricas consideradas foram a eficiência do protocolo como a razão entre o número de pacotes recebidos pelo número de pacotes enviados e a média dos valores referente ao tempo de resposta que compõe a diferença entre o tempo de solicitação com o tempo de resposta da aplicação.

Tabela II . Cenário 01 - Estatísticas do tempo de Resposta em milissegundos com protocolo HTTP

Payload (Bytes)	Média	Mediana	Desv.Padrão	Min.	Max
20	29.61	14	27.88	3	148
50	33.72	10.5	46.84	3	252
100	69.43	13	131,29	3	814
200	55.52	8.5	64,29	3	325
255	83.65	30	68,68	3	405

Tabela III. Cenário 01 – Estatísticas de tempo de resposta em milissegundos com protocolo Websocket

Payload (Byes)	Média	Mediana	Desv.Padrão	Mín	Máx
20	21.82	13	23.56	3	134
50	34.67	17.5	67.10	3	438
100	21.57	16	17.85	3	92
200	34.00	23.5	42.31	3	303
255	49.06	19	108,71	3	694

A figura 5 expõe a eficiência do protocolo em relação ao tamanho do *payload* em bytes. O valor máximo atingindo deve ser igual a um, uma vez que se tem um valor fixo de pacotes enviados igual a 80. Portanto, fica claro na figura 5 que o protocolo *Websocket* atingiu valores superiores que o HTTP, alcançando a total eficiência nos experimentos realizados, principalmente nos pacotes de 100, 200 e 255 bytes, enquanto que no HTTP o desempenho foi muito abaixo do esperado.

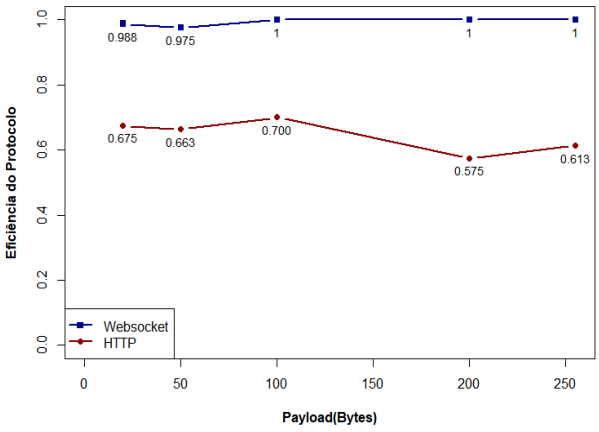

Fig 5. Eficiência do Protocolo

Nas figuras 6 e 7 são exibidos os gráficos em barras para a visualização em porcentagem de perda para cada tamanho do *payload* testado. Com estes gráficos é possível observar que o protocolo HTTP apresenta uma alta taxa de perdas de pacotes, sendo o pacote com 200 bytes o maior percentual, alcançando a faixa de 42%, ou seja, entre 80 pacotes enviados 34 foram perdidos. Essas perdas de pacotes podem ocorrer entre outros motivos, devido a arquitetura do protocolo e condições de rede que influenciam diretamente.

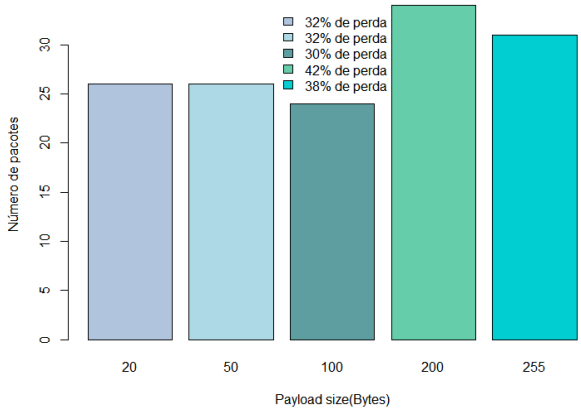

Fig 6. Representação de Perdas de Pacotes com HTTP

Por outro lado, na figura 7, tem-se os valores significativamente reduzidos de perdas de pacotes utilizando o protocolo *Websocket*, em que o mesmo chega a quatidade máxima de 2 pacotes perdidos. Deve-se salientar que, mesmo que as condições de rede possam influenciar diretamente no recebimento de pacotes, ainda sim haveria uma diferença significativa em relação ao protocolo HTTP. Para pacotes com 100, 200 e 255 bytes não houve percentual de perda detectada no teste realizado.

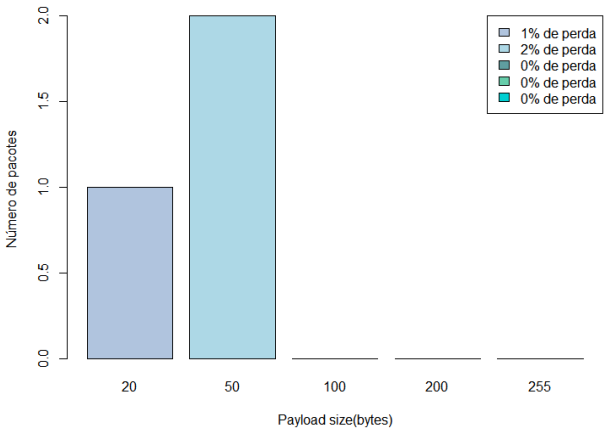

Fig 7. Representação de Perdas de Pacotes com Websocket

A figura 8 exibe os valores médios do tempo de resposta em milissegundos em relação a cada tamanho de *payload*. Segundo este gráfico é possível observar os valores em relação ao tempo que indica que o protocolo HTTP possui valores médios maiores, indicando um tempo de resposta maior. O protocolo *Websocket* tem como valor de maior média 49 milissegundos, enquanto que o HTTP ultrapassa os 90 milissegundos.

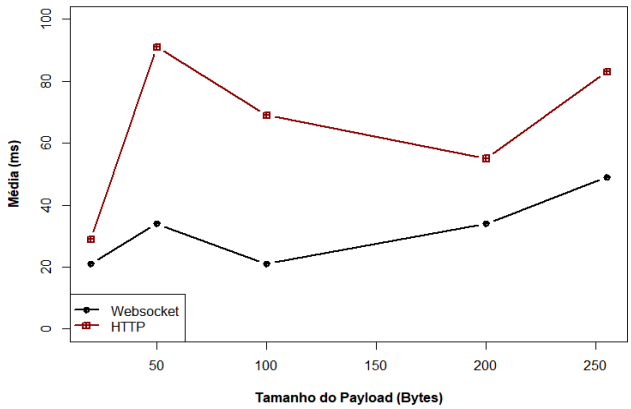

Fig 8. Representação dos valores médios de tempo por tamanho de payload por bytes

Os experimentos do segundo cenário foram realizados ao longo de um período de tempo pré-determinado (três minutos). Cada coleta foi repetida duas vezes visando uma melhor visualização.

Fig 9. Representação de número de pacotes recebidos por protocolo

Quanto ao gráfico de barras (Figura 9) reflete a relação do número de pacotes recebidos no tempo de três minutos. Nos dois testes, o protocolo *Websocket* recebeu uma quantidade maior de pacotes em comparação com o protocolo HTTP, validando os resultados obtidos no primeiro cenário, que demonstraram que, de forma geral, o protocolo *Websocket* apresenta vantagens sobre o protocolo HTTP.

6. CONCLUSÕES

Os tópicos apresentados neste trabalho tiveram como objetivo analisar o desempenho de dois protocolos da camada de aplicação HTTP e *Websocket* utilizados em rede de sensores sem fio em cenários não críticos com hardware restrito ESP8266 NodeMCU compondo cada nó sensor. Os testes foram analisados perante os aspectos quantitativos, como a eficiência do protocolo, média do tempo de resposta e número de pacotes recebidos.

Os resultados evidenciaram que o protocolo *Websocket* alcança uma maior eficiência em comparação com o HTTP, principalmente em relação ao tempo de resposta da aplicação diante da requisição do cliente. Ambos os protocolos trabalharam segundo o modelo TCP/IP. Os testes com o HTTP tiveram como protocolo na camada de transporte o UDP enquanto que os testes com o *Websocket* foram estabelecidos com o protocolo TCP.

Entre os dois protocolos também fica claro a redução do *overhead* na comunicação com o protocolo *Websocket* devido ao menor tamanho de cabeçalho em comparação com HTTP, além da maior taxa de mensagens alcançada pelo *Websocket*. Contudo, o *Websocket* não foi projetado para aplicações em dispositivos restritos. Porém, sua aplicabilidade em cenários utilizando RSSF tem se tornado cada vez mais alvo de estudos em virtude de suas características englobando comunicação em tempo real. Deste modo, a implementação do *Websocket* em rede de sensores sem fio mostra-se uma opção viável como protocolo para a camada de aplicação em cenários como os testados.

AGRADECIMENTOS

O presente estudo foi possível através do apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) e do suporte dos docentes e discentes da Universidade Federal de Minas Gerais (UFMG). Agradeço o empenho e atenção.

REFERÊNCIAS

Dargie, W., Poellabauer, C. (2010). “Fundamentals of Wireless Sensor Networks: Theory and Practice”. John Wiley &Sons.

Karadogan, G. M. (2013). “Evaluating *Websocket* and WebRTC in Context of a Mobile Internet of Things Gateway”. School of Informationand Communication Technology KTH Royal Institute of Technology.

Karl, H., Willing, A. (2005) “Protocols and Architectures for Wireless Sensor Networks”. John Wiley & Sons.

Katsikeas, S. (2016). “A lightweight and secure MQTT implementation for Wireless Sensor Networks”. Technical University Of Crete.

Kosanovic, M., Kosanovic, M. (2017). “Applicability of *Websocket* protocol into Wireless Sensor Networks”. 7th International Conference on Information Society and Technology – ICIST 2017.

Mijonovic, S., Shehu, E., Buratti, C. (2016). “Comparing Application Layer Protocols for the Internet of Things via Experimentation”. IEEE 2nd International Forum on Research and Technologies for Society and Industry Levaring a better tomorrow (RTSI).

Mota, F.A.O. (2017). “Aplicação de Redes de Sensores sem Fio para a Análise de Movimento Humano”. Programa de Pós-Graduação em Engenharia Elétrica – UFMG.

Mukherjee, N., Neogy, S. and Roy, S. (2016) “Building Wireless Sensor Networks: Theoretical & Practical Perspectives”. Taylor and Francis Group. New York.

Pereira, V.N.S.S. (2016). “Medição de desempenho em Redes de Sensores Sem Fios”. Tese de Doutorado. Universidade de Coimbra.

Srinivasan, L., Scharnagl, J., Schilling, K.(2013). “Analysis of Websockets as the new age Protocol for Remote Robot Tele-Operation”. 3rd IFAC Symposium on Telematics Applications.

Tanenbaum, A.S. (2011). “Redes de Computadores”. Marson.