Implementação do protocolo CoAP para serviços de monitoramento em redes de sensores sem fio

Rafael de Lucena Valle[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

rafaeldelucena@inf.ufsc.br

Abstract. *Currently the Internet has limited capability of monitoring and acting on the real world. Both network layer and application layer interoperability are needed between WSNs and networked objects in the Internet to solve this problem.*
Up to now, diverse structures of sensor nodes and the non-standard protocols used were the biggest obstacles in front of integrating wireless sensor networks into the Web.
Standard and widely used protocols are needed, RESTful Web services is a very good option for that matter. But for constrained nodes requires a lighter protocol than HTTP, especially due to the verbose nature of HTTP and XML specifications.
This work presents an RESTful Web service infrastructure for wireless sensor networks using Constrained Application Protocol and the integration with the Internet via General Packet Radio Service.

Resumo. *Redes de sensores e atuadores são utilizadas para a captura, processamento de informação e atuação sobre um ambiente, tornando-as importantes em aplicações de controle, telemetria e rastreamento.*
Estas redes são compostas por nós sensores que trabalham em conjunto a fim de obter dados de um ambiente. Possuem processadores, transmissores e receptores de dados simplificados, restrições de memória e energia, geralmente sem alimentação constante de energia. Contudo possuem um custo baixo de equipamentos, tornando interessante a implantação destes sistemas.
O protocolo HTTP, um protocolo de aplicação muito utilizado na atualidade, foi desenvolvido para computadores de propósito geral, onde essas restrições não existem. Um protocolo leve como CoAP pode tornar viável o desenvolvimento de aplicações web em redes de sensores sem fio.
Este trabalho propõe uma infraestrutura de comunicação entre redes de sensores sem fio e a Internet, utilizando protocolos leves entre os nós sensores e um gateway GPRS para aproveitar a cobertura da tecnologia GPRS.
Com a implementação do CoAP é esperado uma redução de consumo de energia e memória, em relação a outros protocolos de aplicação existentes.

1. Introdução

Avanços recentes nas tecnologias de sistemas eletrônicos, semicondutores, sensores, microcontroladores e rádios tornaram possível o desenvolvimento de redes de sensores de baixo custo e baixo consumo de energia uma realidade.

Nós que participam destas redes geralmente são compostos por computadores e rádios simplificados, que possuem restrições de memória, processamento, energia e capacidade de comunicação, mas um custo relativamente baixo de equipamentos [Dargie and Poellabauer 2010].

Os requisitos para uma rede de sensores distribuída são: reconfiguração com estação base, controle autonômo de operação e gerência de energia, auto-monitoramento, eficiência energética para longo tempo de operação e apta a incorporar diversos sensores [Bult et al. 1996].

O objetivo geral desse trabalho é descrever e implementar webservices em uma rede sensores sem fio que farão a aquisição dos dados do ambiente e disponibilizarão as informações captadas na Internet.

Este trabalho realizará o desenvolvimento de aplicações web e software de sistema para fazer a ponte entre a rede de sensores e a Internet. O Sistema operacional utilizado será o EPOS, que possui uma implementação de pilha UDP/IP. A aplicação integradora GPRS/802.15.4 irá executar na plataforma EposMoteII utilizando uma extensão GPRS.

A comunicação entre os nós da rede será feita através do protocolo de aplicação CoAP, um protocolo específico para redes de sensores sem fio. Será utilizado um porte de uma implementação livre do protocolo CoAP.

Sendo assim, os objetivos específicos são:

1. Portar o protocolo CoAP para o EPOS;
2. Desenvolver a aplicação gateway GPRS/802.15.4.
3. Desenvolver uma WEP API para coleta da informação dos nós sensores.
4. Avaliar a solução desenvolvida.

2. Justificativa e Motivação

O maior consumo de energia de um nó sensor é no envio e recebimento de dados, quando mantem seu transmissor ligado. Além disso quem recebe a mensagem precisa montá-la e tratar as partes corrompidas, podendo gerar retransmissões. Inclusive o desafio dos algoritmos de comunicação nesta área é manter os rádios ligados o mínimo de tempo possível sem comprometer a conectividade do nó [Jennifer Yick 2008].

Os mecanismos de confiabilidade na transmissão de dados, técnicas para se manter uma conexão do TCP e rearranjos que são feitos para garantir a ordem das mensagens recebidas não são adequados para um dispositivos com suprimento limitado de energia, como uma bateria ou uma placa fotovoltáica. Estas técnicas fazem que os transmissores fiquem ligados por mais tempo, para manter a conexão ou até mesmo para reenvio de mensagens.

Por sua vez o protocolo do UDP, não mantém conexão, dados são recebidos fora de ordem e o envio é feito de uma mensagem por vez. Isto implica também na redução do tamanho do cabeçalho do pacote.

A falta de padronização dos protocolos afeta o desenvolvimento de uma rede pública ubíqua de uma cidade inteligente, por exemplo. Grande parte das soluções utiliza protocolos proprietários que se comunicacam apenas com os produtos de um mesmo fabricante.

Um protocolo leve como CoAP pode tornar viável a criação de aplicações web em redes de sensores sem fio por um baixo custo. Com a utilização do CoAP é esperado uma redução de consumo de energia e memória, em relação a outros protocolos de aplicação existentes.

Estas características demostram uma alternativa interessante para estes equipamentos limitados. Testes feitos em implementações de sistemas operacionais similares ao EPOS, como Contiki e TinyOS, utilizando o protocolo CoAP demonstram redução no consumo de energia e memória em relação ao HTTP [Kuladinithi et al. 2011].

Em lugares aonde não existe o acesso a rede cabeada ou sem fio, como lugares afastados, na área rural, por exemplo a distribuição da informação para Internet será feita através de um gateway. Atualmente o padrão GPRS oferece a maior cobertura dentre as tecnologias de transmissão de telefonia no Brasil, atingindo cerca de 5477 municípios [Anatel 2014].

3. CoAP

O CoAP é um protocolo de aplicação com intuito de ser utilizado em dispositivos eletrônicos simplificados, permitindo comunicação interativa com a Internet e outros dispositivos. A especificação do protocolo é a RFC 7252. Um dos principais objetivos do CoAP é ser uma alternativa protocolo web genérico para redes com dispositivos com restrição de energia e memória [Zach Shelby 2013].

A comunicação entre os pontos no CoAP é assíncrona usando o UDP. A confiabilidade é um parâmetro opcional e funciona através de um mecanismo de retransmissão exponencial. Possui 4 tipos de mensagem: Confirmável, Não-Confirmável, Confirmação (ACK) e Reset. A figura 1 mostra o formato do pacote.

Uma mensagem CoAP deve caber num único pacote IP, para que seja transmitida numa camada de enlace limitada.

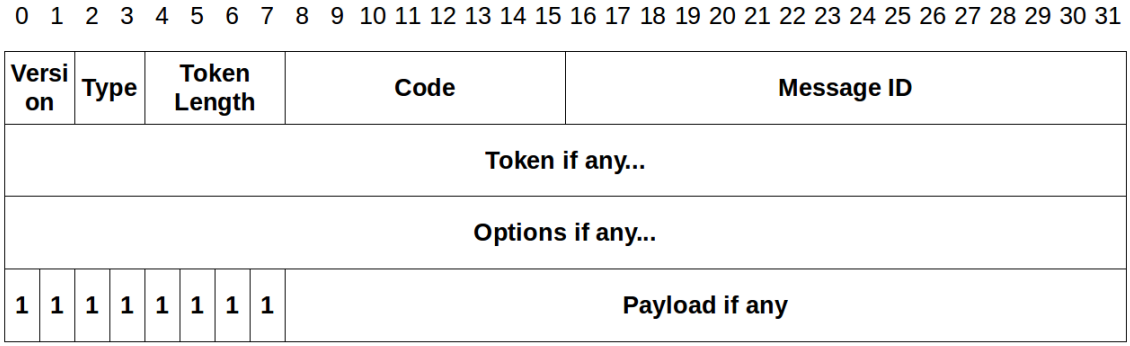


Figure 1. Formato do pacote CoAP em bits [Zach Shelby 2013].

Os campos do pacote CoAP são: a versão do CoAP, implementações devem utilizar este campo com o valor 1. O tipo: campo para definir o tipo da mensagem: Confirmável (0), Não-Confirmável (1) , de Confirmação (2) ou Reset (3).

O tamanho do Token: utilizado para controle de requisições e repostas. O tamanho do Token pode variar entre 0 e 8 bytes. Tamanhos entre 9 a 15 são reservados e não devem

ser usados. É um campo sempre gerado pelo cliente CoAP.

O Código: separados em 3-bit mais significativos para classes e 5-bits menos significativos para detalhe. As classes podem indicar uma requisição (0), uma resposta de sucesso (2), e uma resposta de erro do cliente (4), ou uma resposta de erro do servidor (5), as outras classes são reservadas. Em um caso especial o código 0.00 indica uma mensagem vazia.

O ID da mensagem: usada para deduplicação de mensagens e confirmação ou reset de mensagens. É gerado por quem envia a mensagem, no caso de uma mensagem confirmável ou reset, a resposta deve possuir o ID da mensagem enviada. A implemetação da geração dos IDs está aberta, depende da aplicação que o CoAP será usado, porém é recomendado que o valor inicial seja randômico.

A transmissão de mensagems é controlada basicamente pelos parâmetros: ACK TIMEOUT, ACK RANDOM FACTOR, MAX RETRANSMIT, NSTART, Leisure e PROBING RATE. Estes parâmetros são respectivamente: o tempo que uma mensagem confirmável aguarda o ACK; fator de randomicidade para gerar os ACK TIMEOUTs subsequentes; contador para o número máximo de tentativas de retransmissão; número limite de interações simultâneas mantidas por um servidor.

A Leisure é o tempo que o servidor aguarda para responder uma requisição multicast, é calculada: $Leisure = S * G/R$. Aonde S é o tamanho estimado da reposta, G é uma estimativa do tamanho do grupo e R é a taxa de transmissão. PROBING RATE: é a taxa média para transmissão de dados. Estes parâmetros definem a temporização do sistema.

Uma falha na transmissão ocorre quando atingir o número máximo de tentavivas ou receber uma mensagem de RESET. Quando receber um ACK a transmissão da mensagem confirmável é completa. O servidor irá ignorar mensagens que chegam por multicast quando não puder responder nada de útil.

Na situação aonde possuir uma informação suficientemente nova pode responder na própria mensagem de confirmação (ACK). Essa técnica é chamada de "Piggy-backed" um mecanismo de transmissão para mensagens confirmadas.

Uma requisição é inicializada ao preencher o campo código no cabeçalho do CoAP e gerar um token. Para finalizar o fluxo é necessário que a resposta chegue e o token seja o mesmo. A especificação também prevê fluxo de requisição com confirmação, e resposta separada com confirmação.

A descoberta de recursos é feita quando um servidor recebe uma requisição GET para o recurso /.well-know/core. O servidor CoAP deve responder no formato CORE link Format[Shelby 2012]. A descoberta de serviços no protocolo CoAP é feita através de socket Multicast. Os recursos são identificados por uma URI, e os métodos são implementados de forma similar ao HTTP.

4. EPOS

O EPOS é um sistema operacional multithread com suporte a preempção, desenvolvido em C++ que faz uso intenso de programação orientada a aspectos utilizando templates.

Foi projetado utilizando ADESD, Application Driven Embedded System Design,

um método para projeto de sistemas embarcados orientados à aplicação. Esta metodologia guia o desenvolvimento paralelo de hardware e software além de manter portabilidade. O EPOS possui porte para as seguintes arquiteturas: MIPS, IA32, PowerPC, H8, Sparc, AVR e ARM. [EPOS 1999]

A portabilidade é atingida utilizando entidades chamadas de Mediadores de Hardware que fornecem interfaces simples para acesso as funções específicas de arquitetura. Estas interfaces são utilizadas por entidades abstratas como alarmes e threads periódicas. Abaixo uma a figura 2 ilustrando a visão geral do EPOS.

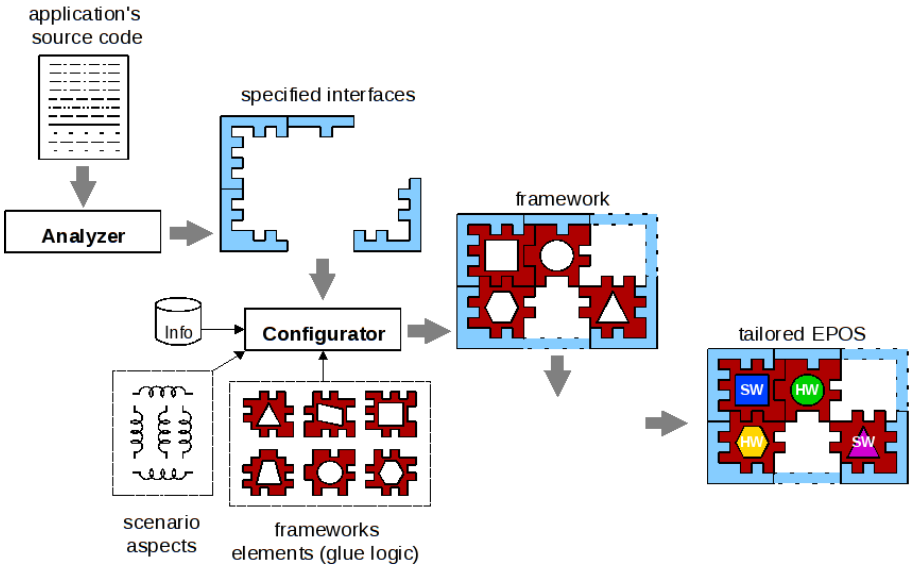


Figure 2. Overview do EPOS.

O EPOS também possui uma interface de software/hardware que abstrai sensores de forma uniforme, definindo classes de dispositivos baseados numa finalidade. Possui abstrações para entidades temporais como relógio, alarme e cronômetro, biblioteca com estruturas de dados e sequenciadores. Permitindo o uso de ferramentas para geração automatizada de abstrações de sistemas [Antonio Augusto Frohlich 1999].

A infraestrutura de comunicacação do EPOS para redes de sensores sem-fio é implementada pelo protocolo C-MAC, Configurable MAC, que provê suporte a comunicacação de baixo nível (MAC - Medium Access Control).

Possui uma pilha de protocolos basedo na arquitetura OSI e outra abordagem Cross-layer. A implementação IPv4 utiliza um controle de fluxo aonde os nós sinalizam a disponibilidade de buffer para mensagem única em tempos ajustando o tempo entre a troca de mensagens [Frohlich et al. 2013].

5. Desenvolvimento

Os requisitos funcionais da solução coletar informação do ambiente através de sensores e transmití-las através da Internet e fácil integração com a Internet mesmo em locais sem rede WIFI.

As principais funções deste gateway são receber os dados da rede de sensores e encaminhá-las para um servidor remoto que armazenará essas informações e exibirá de forma conveniente para o usuário final.

As funções da da aplicação do gateway são:

1. Configuração, envio e recebimento de SMS.
2. Configuração de contexto PDP, Configuração GPRS.
3. Configuração UDP/IP no enlace GPRS.
4. Envio e recebimento de requisições CoAP.

As funções da aplicação cliente serão:

1. Listar recursos CoAP web de uma RSFF.
2. Enviar Requisições e receber respostas CoAP da RSFF.

Os requisitos não funcionais são:

Armazenamento: deve ser suficientemente pequeno para ser utilizado em microcontroladores.

Energia: consumir pouca energia para longa durabilidade.

Valor: utilizar uma infraestrutura de hardware simples para realizar as tarefas.

Interoperabilidade: A padronização na comunicação visa facilitar a interconexão dos sistemas de diversas plataformas.

Durante o desenvolvimento foram realizados diversos estudos para escolher módulo GPRS adequado à tarefa e o trabalho necessário para acoplar o protoloco. Testes de validação dos sistemas de software e validação do módulo GPRS foram realizados.

A aplicação responsável pelo roteamento de mensagens para Internet utiliza a tecnologia GPRS, provida por um módulo GSM/GPRS da Quectel o M95.

A aplicação é composta pelos nós webservers CoAP, um nó cliente CoAP que fará o roteamento para Internet utilizando um módulo GPRS. Os webservers informam a temperatura, através de respostas a requisições CoAP. A figura 3 ilustra a interconexão entre os nodos da rede.

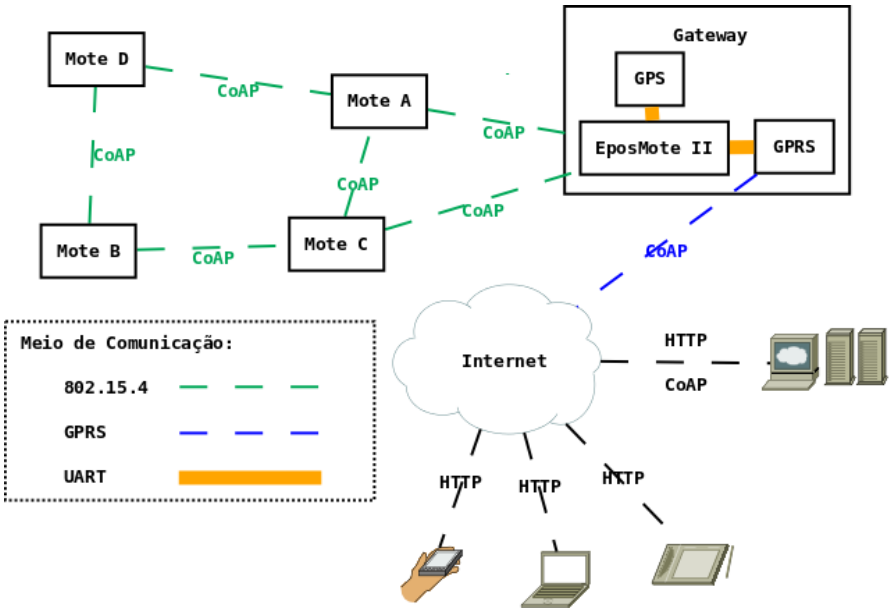


Figure 3. Visão geral sobre comunicação do sistema.

A biblioteca utilizada para montar o pacote CoAP foi a biblioteca [Mills 2014] algumas correções e testes para facilitar a verificação da execução correta dos algoritmos internos durante o desenvolvimento da aplicação. As alterações resultaram em contribuição para o referido projeto, aceita pelo mantenedor.

Para o funcionamento desta biblioteca no EPOS, e para utilizar uma MTU limitada a 128 bytes é utilizado um buffer com um valor máximo e armazenado os dados do pacote no buffer. Foi necessário alterar os tipos das variáveis para se adquerem ao EPOS.

A modelagem da solução visa uma implementação modular, para facilitar a verificação do correto funcionamento e também para ser de fácil adaptaçã a diversas plataformas. O sistema proposto é composto pela aplicação gateway, por um servidor coap externo e um servidor http externo, este apenas para facilitar o acesso de dispositivos sem suporte ao protocolo CoAP.

O código desenvolvido do servidor CoAP externo e o código de sistema da plataforma alvo estão disponíveis em [de Lucena Valle 2014].

6. Resultados

Atualmente a aplicação não possui uma implementação segura do procotolo CoAP, que utiliza o DTLS. Para resolver este problema é necessário que o pacote seja criptografado.

A ETSI em conjunto com a IPSO desenvolveram um conjunto de testes para validar o comportamento entre diversas implementações CoAP. Este teste foi aplicado em 24, 25 de março de 2012 em Paris, conhecido como primeiro evento IOT CoAP plugtest. Para validar a interoperabilidade os teste são: especificação básica do CoAP, transferência em bloco, observação de recursos CoAP, formato CORE link. Os testes são executados entre diferentes dispositivos e implementações CoAP.

As tabelas abaixo 1, 2, 3 mostram os resultados dos testes, os símbolos "✗" e "✓" para a solução desenvolvida neste trabalho.

CENÁRIO	RESULTADO
Testes para especificação básica CoAP	
Tratar GET, confirmável.	✓
Tratar POST, confirmável.	✓
Tratar PUT, confirmável.	✓
Tratar DELETE, confirmável.	✓
Tratar GET, sem confirmação.	✓
Tratar POST, sem confirmação.	✓
Tratar PUT, sem confirmação.	✓
Tratar DELETE, sem confirmação.	✓
Tratar GET com resposta separada.	✓
Tratar requisição com Token.	✓
Tratar requisição sem Token.	✓
Tratar requisição opções URI-Path.	✓
Tratar requisição opções URI-Query.	✓
Interoperablidade em contexto de perda (CON mode, piggybacked response)	✓
Interoperablidade em contexto de perda (CON mode, delayed response)	✓
Tratar GET com resposta separada, sem confirmação.	✓

Table 1. Resultados IOT Plugtest.

CENÁRIO	RESULTADO
Testes para validar o formato de dados CORE link Format	
Descoberta de recursos well-known.	✗
Utilização de consulta para filtrar resultados.	✗
Testes para validar a transferência de blocos	
Transferência de blocos grandes utilizando GET (negociação antecipada).	✗
Transferência de blocos grandes utilizando GET (negociação atrasada).	✗
Transferência de blocos grandes utilizando o PUT.	✗
Transferência de blocos grandes utilizando o POST.	✗

Table 2. Resultados IOT Plugtest.

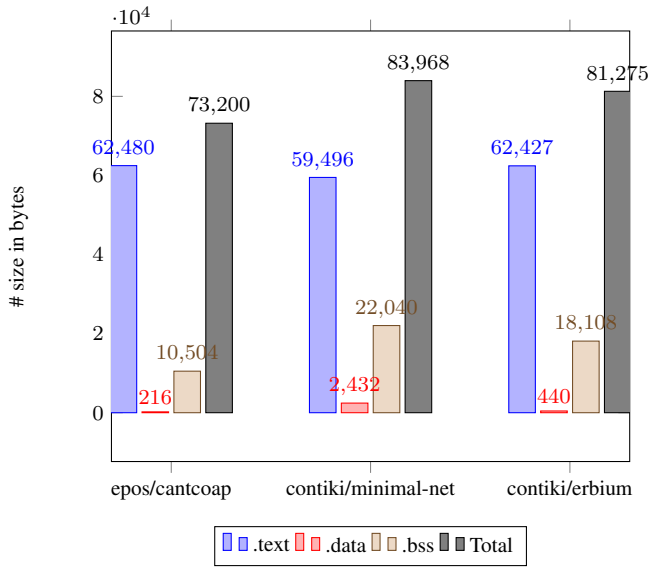
CENÁRIO	RESULTADO
Testes para observação de recursos	
Tratar observação de recursos.	✗
Parar a observação de recursos.	✗
Detecção de deregistro do cliente (Max-Age).	✗
Detecção de deregistro do servidor (client OFF).	✗
Detecção de deregistro do servidor (RESET explícito).	✗

Table 3. Resultados IOT Plugtest.

A fim de mensurar algumas informações importantes no contexto da aplicação foi utilizada a versão 4.4.5 para o GCC x86 e o GCC 4.3.2 (Sourcery G++ Lite 2008q3-66) para ARM no Contiki encontrada [Alvira 2014]. A aplicação no EPOS utiliza a versão 4.4 do compilador GCC disponível em [EPOS 1999].

As flags de compilação utilizadas foram as padrões de cada sistema de construção dos projetos para a arquitetura x86 e para para ARM7 do MC1224V utilizado no EPOS-MoteII e no Econotag, plataforma presente no Contiki e utilizado para comparativos.

No gráfico abaixo observa-se que a solução do EPOS, no total de memória utilizada, cerca de 12,82% menor que a implementação do Contiki Minimal Net e 9,93% menor que a implementação utilizando o Erbium.



7. Conclusão

O objetivo principal foi atigindo, o desenvolvimento de um gateway simplificado para redes de sensores que repassa as mensagens recebidas para um servidor CoAP na Internet. É possui acessar o sistema utilizando um cliente HTTP, facilitando a disponibilização de serviços de sensoriamente e atuação.

As principal contribuições deste trabalho são a avaliação de diferentes implementações em diversas plataformas e a implementação de uma infraestrutura para o desenvolvimento de aplicações que utilizem redes de sensores sem fio.

O protocolo CoAP é adequado a redes com um elevado número de nós baseando-se na arquitetura da Web, focando na utilização dos princíos REST para o desenvolvimento de novas aplicações [Fielding 2000].

O tamanho do código respeitou o espaço de armazenamento restrito, comum desse tipo de aplicação e foi possível criar uma aplicação que possa ser utilizado em uma vasta gama de dispositivos.

Uma aplicação que poderia reduzir ainda mais o custo para integrar as redes a Internet, é a implementação de um gateway que utilize Software Defined Transceiver, assim é possível integrar uma gama enorme de dispositivos e utilizar apenas um módulo transceiver para enviar os dados da RSSF para Internet.

Outro trabalho futuro, seria um gerador de código que utiliza como entrada uma linguagem de especificação dos possíveis recursos e como saída gera um código ANSI C mínimo de um servidor web utiliza CoAP e seus respectivos recursos. Este gerador deve ser genérico suficiente para ser fácil a adaptação de diferentes pilhas CoAP/UDP/IP, arquiteturas e tipos de sensores.

Outra idéia interessante é um modelo para apresentação dos dados coletados dos recursos disponíveis na rede.

References

Alvira, M. (2014). Library code for the freescale mc13224v arm7 soc with 802.15.4 radio.

Anatel (2014).

Antonio Augusto Frohlich, W. S.-P. (1999). Epos: an object-oriented operating system.

Bult, K., Burstein, A., Chang, D., Dong, M., Fielding, M., Kruglick, E., Ho, J., Lin, F., Lin, T.-H., Kaiser, W., Marcy, H., Mukai, R., Nelson, P., Newburg, F. L., Pister, K., Pottie, G., Sanchez, H., Sohrabi, K., Stafsudd, O., Tan, K. B., Yung, G., Xue, S., and Yao, J. (1996). Low power systems for wireless microsensors. In *Low Power Electronics and Design, 1996., International Symposium on*, pages 17–21.

Dargie, W. and Poellabauer, C. (2010). *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons.

de Lucena Valle, R. (2014). Coap gateway source code.

EPOS (1999). Epos project.

Fielding, R. T. (2000). *Representational State Transfer (REST)*. PhD thesis.

Frohlich, A., Okazaki, A. M., Steiner, R. V., Oliveira, P., and Martina, J. E. (2013). A cross-layer approach to trustfulness in the internet of things. In *9th Workshop on Software Technologies for Embedded and Ubiquitous Systems (SEUS), Paderborn, Germany*, pages 884–889.

Jennifer Yick, Biswanath Mukherjee, D. G. (2008). Wireless sensor network survey.

Kuladinithi, K., Bergmann, O., Pötsch, T., Becker, M., and Görg, C. (2011). Implementation of coap and its application in transport logistics. *Proc. IP+ SN, Chicago, IL, USA*.

Mills, A. (2014). Cantcoap documentation.

Shelby, Z. (2012). Constrained restful environments (core) link format. RFC 5280 (Proposed Standard).

Zach Shelby, Klaus Hartke, C. B. (2013). Constrained application protocol (coap). Internet-Drafts.