Constrained Application Protocol (CoAP) no Arduino UNO R3: Uma Análise Prática

Cleber B. da Porciúncula[1], Sílvio Beskow[1], Daniel Stefani Marcon[1], Jéferson Campos Nobre[1]

[1]Universidade do Vale do Rio dos Sinos (Unisinos)
Porto Alegre – RS – Brasil

{cleber.bp,segbeskow}@gmail.com, {danielstefani,jcnobre}@unisinos.br

Abstract. *Constrained Application Protocol (CoAP) is an initiative to exchange information in restricted devices, and its use is appropriate for Internet of Things (IoT). Currently, a set of initiatives is being developed for the creation of a library, using several languages (including C, C ++, Python and Java) to evaluate the characteristics and problems of the protocol in real environments. These libraries are being tested on Linux operating systems or on specific embedded systems, which have more robust computing resources than restricted devices. This work implements a CoAP library written in the C language and its installation in an Arduino Uno R3, which has the characteristics of a restricted device.*

Resumo. *O protocolo CoAP (Constrained Application Protocol) é uma iniciativa de padronização para troca de informações em dispositivos restritos, sendo o seu uso apropriado para o ambiente de Internet das Coisas (IoT). Atualmente, um conjunto de iniciativas está em desenvolvimento para a criação de uma biblioteca, utilizando várias linguagens (incluindo C, C++, Python e Java) com o objetivo de avaliar as características e problemas do protocolo em ambientes reais. Estas bibliotecas estão sendo testadas em sistemas operacionais Linux ou em sistemas embarcados específicos, que possuem recursos computacionais mais robustos que dispositivos restritos. Este trabalho implementa uma biblioteca CoAP escrita na linguagem C e sua instalação em um Arduino UNO R3, que possui as características de um dispostivo restrito.*

1. Introdução

Dispositivos restritos computacionalmente, normalmente em termos de processador, memória e consumo de energia, possuem aplicações em diversos tipos de organizações. Tais dispositivos são frequentemente usados como sensores, atuadores ou objetos inteligentes [Bormann et al. 2014]. Em termos de cenários, dispositivos restritos podem ser utilizados, por exemplo, em ambientes industriais, comerciais e diretamento no meio ambiente. Tais dispositivos podem formar redes de comunicação trocando informações entre si e com outras redes, como a Internet.

Redes formadas por dispositivos restritos computacionalmente são costumeiramente conhecidas como Internet das Coisas (*Internet of Things* - IoT). Tais redes podem também apresentar restrições, como, por exemplo, canais de comunicação inconstantes e com taxa de transferência muito limitada [Bormann et al. 2014]. Da mesma forma que em redes tradicionais, a comunicação de tais dispositivos possui requisitos de segurança

para garantir Confidencialidade, Integridade e Disponibilidade (*Confidentiality, Integrity, and Availability* - CIA).

Mecanismos de sergurança em dispositivos restritos precisam seguir padronizações, a fim de permitir utilizações interoperáveis e abertas. Neste contexto, diversos orgãos de padronização produzem documentos relacionados a dispositivos restritos. Dentre tais orgãos, o *The Internet Engineering Task Force* (IETF) é um grupo definido para desenvolver e aprimorar especificações, padrões e tecnologias associadas à Internet. Através de Grupos de Trabalho (*Working Groups*- WG), novos padrões e tecnologias são consideradas, analisadas e, por consenso, a proposta poderá se tornar um novo padrão de tecnologia oferecido para a Internet [Hoffman 2013].

Diversos WGs do IETF produzem padrões relacionados com dispositivos restritos. Neste contexto, o WG *Authentication and Authorization for Constrained Environments* (ACE) propõe o uso de mecanismos que suportam a autenticação e autorização em ambientes formados por dispositivos restritos [Kaduk et al. 2014]. Além de garantir a autenticação e autorização, os mecanismos propostos pelo ACE WG pretendem não afetar o tempo de resposta dos dispositivos, a fim de manter a Qualidade de Experiência (*Quality of Experience* - QoE) dos usuários em um nivel adequado.

O ACE WG desenvolve um conjunto de padronizações para mecanismos adequados a dispositivos restritos. Dentre tais mecanismos, podem ser ressaltados: *Constrained Application Protocol* (CoAP) [Shelby et al. 2014]), *Datagram Transport Layer Security* (DTLS) [Rescorla and Modadugu 2012] e OAuth 2.0 [D. Hardt 2012]. No entanto, não existem trabalhos que descrevem utilizações práticas dos mecanismos descritos no ACE WG.

O presente trabalho descreve os mecanismos de segurança desenvolvidos pelo ACE WG através de um estudo de caso de cunho prático. Tal estudo de caso envolveu o desenvolvimento e adaptação de protocolos, além da realização de experimentos para verificar as propriedades da utilização do ACE em um ambiente restrito. O dispositivo restrito empregado no presente trabalho foi o Arduino UNO R3[1].

O presente artigo está organizado da seguinte forma. Na Seção 2, é apresentada a fundamentação teórica. Na Seção 3, o ACE WG é descrito em maiores detalhes. Na Seção 8, perpectivas futuras para a autenticação e autorização de dispositivos restritos são ressaltadas. Finalmente, conclusões e trabalhos futuros são discutidos na Seção 9.

2. Fundamentação Teórica

2.1. Dispositivos Restritos

Um dispositivo restrito computacionalmente possui restrições de energia, de armazenamento, de capacidade de transmissão e de processamento. Eles são projetados, em sua maior parte, sem uma interface de comunicação com o usuário, pois são destinados a realizar uma interação sem a utilização da intervenção humana [Seitz et al. 2016].

Um conjunto de dispositivos restritos pode formar uma rede, e esta rede pode apresentar características restritas, como um canal não confiável, com perdas, limitação

[1] Arduino UNO R - https://store.arduino.cc/usa/arduino-uno-rev3

de largura de banda e uma topologia altamente dinâmica [Bormann et al. 2014]. Em ambientes restritos, os requisitos de autenticação e autorização representam desafios para a utilização de protocolos. Os dispositivos restritos podem ser classificados conforme a Tabela 1.

Nome	Data Size	Code Size	Característica
Class 0, C0	«10KiB	«100KiB	restrição severa, usados como sensores
Class 1, C1	∼10KiB	∼100KiB	restrição alta, protocolos específicos (CoAP)
Class 2, C2	∼50KiB	∼250KiB	restrição baixa, pode-se utilizar mais de um protocolo

Tabela 1. Classificação não definitiva de Dispositivos Restritos adaptado da RFC7228 - Fonte: RFC7228 [Bormann et al. 2014]

Os dispositivos ***Class 0*** são basicamente sensores, pois seus recursos de processamento e memória são extremamente limitados, não possuindo capacidade de comunicação com a Internet de forma segura. É necessário a utilização de dispositivos com maiores recursos computacionais como *proxies*, *gateways* ou servidores, que permitirão que os dispositivos ***Class 0*** possam realizar a sua comunicação com o meio externo. O gerenciamento destes dispositivos não é realizado de maneira tradicional; eles são pré-configurados com um conjunto de informações básicas (e raramente reconfigurados) e sua iteração com o meio externo é realizada através de sinais simples em resposta a uma solicitação de dados.

Os dispositivos ***Class 1*** possuem restrições de processamento e memória, o que torna a sua comunicação com a Internet bastante difícil quando necessitam utilizar uma pilha de protocolos completa como o HTTP [R. Fielding and J. Reschke 2014], TLS [Dierks and Rescorla 2008] e XML. Porém, os dispositivos ***Class 1*** podem utilizar uma pilha de protocolos específica para ambientes restritos, como o CoAP sobre UDP. Os dispositivos podem oferecer funções de suporte a segurança necessárias para a utilização em conjunto com uma grande rede, podendo integrar-se a uma rede IP, mas a utilização de seus recursos deve ser realizada de forma econômica, devido as suas restrições de memória e processamento.

Os dispositivos ***Class 2*** são menos limitados em recursos e são capazes de suportar a maioria das pilhas de protocolos utilizadas em computadores e servidores. Contudo, os dispositivos ***Class 2*** podem se beneficiar da utilização de protocolos para ambientes restritos, pois tais protocolos possuem uma maior eficiência energética, além de consumir menor largura de banda. Utilizando menos recursos, mais recursos tornam-se disponíveis para aplicações, reduzindo custos de desenvolvimento e aumentando a sua interoperabilidade.

Dispositivos com limitações de recursos (como CPU, memória, energia e comprimento de banda) podem formar uma rede de componentes restritos, utilizados como sensores ou dispositivos inteligentes. Tais dispositivos em rede, podem possuir restrições, como perda de canais, limitações de banda e possuir uma topologia altamente dinâmica (com a inclusão ou a saída de dispositivos). Os dispositivos realizam coleta de informações em diversos ambientes, como prédios, fábricas, florestas, hospitais, submersos e outros, trabalhando sob inúmeras restrições como energia, memória, consumo de banda e

habilidade de comunicação. A RFC7102 [Vasseur 2014] utiliza o termo LLN (*Low-Power and Lossy Network*) para descrever as várias restrições em uma rede de dispositivos restritos.

2.2. Arquitetura de Sistemas Embarcados

Os sistemas embarcados começaram a ser desenvolvidos em meados de 1970 pela Texas Instruments com o desenvolvimento do microcontrolador TMS 1000 [de Oliveira 2017]. O autor destaca que o objetivo foi automatizar tarefas nos dispositivos e sistemas agregando mais funcionalidades ao componente e que, devido a isso, torna-o barato e acessível. Ainda salienta que esses dispositivos receberam a nomenclatura SoC (System-on-Chip) e, devido às suas características de controle e automação, passaram a se chamar de microcontroladores. A Figura 1 apresenta um exemplo de arquitetura interna de um microcontrolador embarcado de um modulo ESP8266 da Espessif com componentes de processamento e interfaces de entrada e saída.

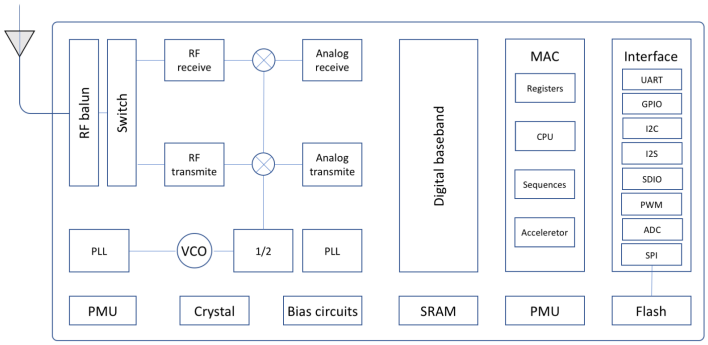

Figura 1. Arquitetura interna do microcontrolador ESP8266 - Fonte:[de Oliveira 2017].

2.3. Arduino UNO R3

O Arduino é uma plataforma *Open Source* de baixo custo destinada ao desenvolvimento de sistemas embarcados [de Oliveira 2017], ficando claro que ele não é um microcontrolador. É uma placa integrada que possui um microcontrolador, geralmente da fabricante Atmel e que possui interfaces de entrada e saída, além de comunicação com computadores através de uma porta USB (*Universal Serial Bus*). Conjuntamente a isso, o Arduino pode ser alimentado de duas formas: uma pela porta USB ligada ao computador, que não é muito recomendada pela variação de alimentação que pode existir de um modelo para outro de computador, e pelo conector de energia através de fonte de alimentação propria ou por um sistema de bateria.

Um importante destaque se dá para a IDE (*Integrated Development Enviroment*), que utiliza a linguagem de programação C e está disponível gratuitamente na Internet. As bibliotecas do Arduino são desenvolvidas em C++. Dessa forma, seus codigos podem ser reaproveitados, tornando o uso mais prático. A Figura 2 apresenta uma prototipagem com uma placa clone Arduíno Uno R3 de uma fabricante nacional, junto com um módulo *WiFi ESP8266*. Os módulos são explicados no tópico seguinte.

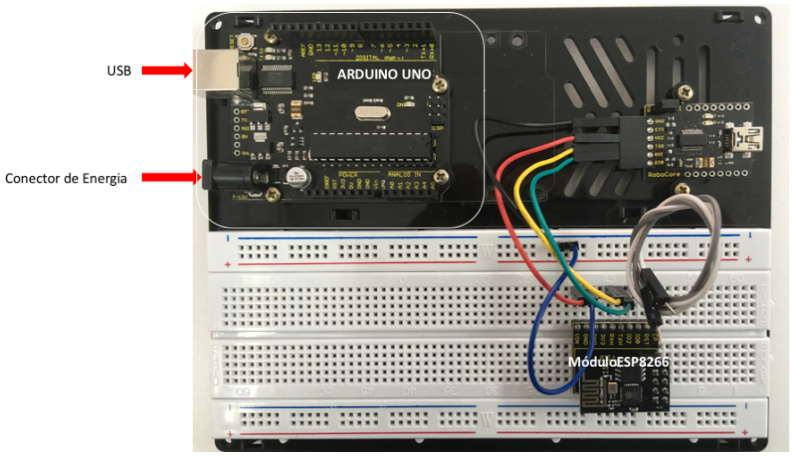

Figura 2. Prototipagem placa clone Arduíno Uno R3 com um módulo *WiFi ESP8266*

2.3.1. Módulos do Arduino UNO R3

As placas Arduino possuem uma vasta gama de módulos que permitem usá-lo para uma série de experimentos através da prototipagem. Os módulos também são conhecidos como *SHIELDS*. Como exemplos, destacam-se os **sensores** (temperatura, luminosidade e outros), bem como **atuadores** e os **módulos de comunicação** (*Ethernet, WiFi e Bluetooth*).

2.3.2. Outros Tipos de Arduino

O Arduino utilizado no desenvolvimento desse trabalho é um Clone Arduino UNO R3. Salienta-se que Arduino é uma marca italiana registrada. Sua plataforma de desenvolvimento de sistemas embarcados é *Open Source*, como mencionado no item 2.3. Por essa razão, o esquema do seu circuito é aberto, o que permite que outros fabricantes produzam seus próprios "Arduinos". Desse modo, eles são denominados de clones ou compatíveis. Durante a pesquisa para aquisição desse hardware, poderá o comprador se deparar com esses termos. Alguns clones poderão possuir melhorias com relação ao registrado, por isso, é aconselhável focar no projeto para depois decidir.

As placas diferem entre si nas suas capacidades computacionais, como RAM, ROM, processador e sistema operacional Linux embutido. As placas tendem a seguir as nomenclaturas da marca registrada. Assim tem-se: placa UNO R3, MEGA, LEONARDO R3, NANO, YÚN e outras. Não faz parte do foco desse trabalho especificar cada placa, apenas identificar e reforçar a placa utilizada no seu desenvolvimento.

2.4. Constrained Application Protocol(CoAP)

O CoAP, definido na RFC7252 [Shelby et al. 2014], é um protocolo de transferência de informações projetado para o ambiente Web, para utilização em ambientes com atributos restritos. Esses ambientes estão sujeitos a perdas, em decorrência da baixa potência de transmissão e processamento. O protocolo suporta dispositivos com pouca memória RAM/ROM, altas taxas de erro e é usado em redes pessoais e de baixa taxa de transferência. Seu projeto é amplamente utilizado em redes M2M (*machine-to-machine*), com

modelo semelhante ao sistema cliente-servidor do HTTP [Wei et al. 2018]. Por fim, ainda trabalha com baixo consumo de energia e alto grau de automação.

O CoAP trabalha integrado com o protocolo HTTP [R. Fielding and J. Reschke 2014]. Ele possuí suporte a *multicast*, e seu desempenho eficiente produz baixa sobrecarga e de fácil utilização em ambientes restritos.

Características do protocolo CoAP:

- satisfaz os requisitos para protocolos Web M2M;
- possui protocolo UDP [Postel 1980];
- suporte nativo a multicast;
- suporte opcional a unicast;
- permite troca de mensagens assíncrona;
- possuí cabeçalho simples e de baixa complexidade;
- permite HTTP *stateless*;
- possibilita que *proxies* acessem recursos CoAP via HTTP e;
- dispõe de conexão segura com o uso de DTLS.

No CoAP, a troca de mensagens é similar ao HTTP. Em interações M2M o protocolo deve estar ativo em ambos os lados, cliente/servidor. A requisição para troca de mensagem é semelhante a uma requisição HTTP, enviada pelo cliente requisitando uma ação (*Method Code*) ou recurso (uma URI) para o servidor. O servidor envia uma resposta (*Response Code*) que pode incluir a representação de um recurso.

O CoAP utiliza troca de mensagens assíncronas por meio do UDP. Essa troca de mensagem é realizada logicamente através de uma camada de transporte que utiliza confiabilidade opcional. São definidos 4 tipos de mensagens: *Confirmable, Non-confirmable, Acknowledgement, Reset*.

O protocolo CoAP é entendido através de uma abordagem de duas camadas: a que interage com o protocolo UDP e de requisição/resposta com a camada de aplicação, conforme definido na RFC 7252 [Shelby et al. 2014].

Tanto ação (*Method Code*) como resposta (*Response Code*) podem transportar requisições e respostas. Na interação dos quatro tipos de mensagens, requisições podem transportar mensagens de confirmação e não-confirmação, bem como mensagens de resposta podem transportar mensagens de conexão.

A troca de mensagens no CoAP utiliza pacotes compactos, elas se localizam na seção de dados do datagrama UDP. O CoAP é um protocolo leve que pode ser transmitido por outros protocolos: SMS, TCP, SCTP. Suas mensagens possuem um formato simples, conforme mostra a Figura 2.

2.5. Segurança da Informação

O protocolo CoAP não possui mecanismo de segurança próprio. Para garantir os requisitos de Autenticação e Autorização, ele pode ser utilizado em conjunto com o protocolo DTLS [Rescorla and Modadugu 2012] e IPSec [Bormann 2012] (opcional). A Segurança da Informação (SI) está alicerçada na tríade confidencialidade, integridade e disponibilidade (CID). Ela se aprofunda e se enraíza em todas as áreas que tenham a informação como ativo a ser protegido. A SI não está restrita somente a CID. No artigo intitulado *An*

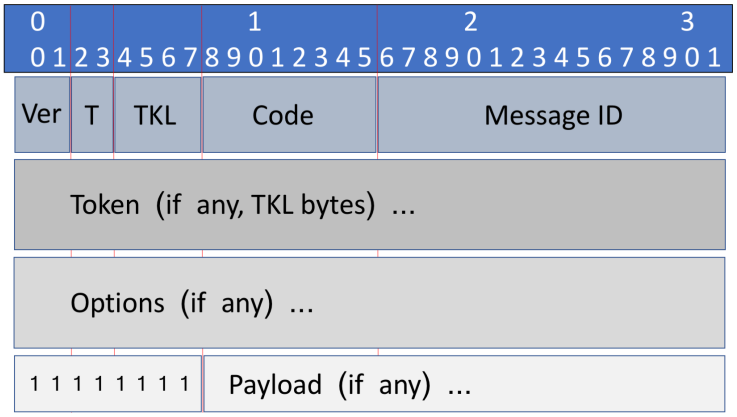

Figura 3. Formato da Mensagem CoAP - Fonte: RFC7252 [Shelby et al. 2014].

Ontology for Network Security Attacks [Simmonds et al. 2004], existe dois mnemônicos que são usados para resumir serviços que devem ser providos por ela: a) o clássico CID e b) o triplo A – *Authentication, Authorization e Accounting* (Autenticação, Autorização e Conta). Uma pessoa não pode usar os serviços de uma conta de usuário até ter sido autorizada e previamente autenticada.

No WG *ACE*, apenas dois requisitos de segurança estão sendo abordados: a Autenticação e a Autorização. Esses requisitos são comentados a seguir.

1. **Autenticação** é um mecanismo que garante que uma entidade (pessoa, organização, aplicação) é quem afirma ser: a) **Autenticação de entidade par** - confirma a identidade de uma entidade. Sua finalidade é evitar que uma entidade ilegítima se disfarce ou repita dados de forma não autorizada; b) **Autenticação da origem de dados** - confirma a legitimidade da origem dos dados. Porém, não garante que esses dados não tenham sido manipulados antes.
2. **Autorização** é um mecanismo que, após a entidade ser autenticada, seus privilégios com relação aos recursos (Ler, Gravar, Modificar ou Controle total) dentro do sistema são definidos em uma base centralizada através da autorização.

A Tabela 2 mostra uma visão geral dos ambientes e protocolos para autenticação e autorização.

3. Autenticação e Autorização para Ambientes Restritos (ACE)

Ambientes com recursos restritos fornecem um desafio maior para a execução de aplicações e protocolos. Eles precisam processar informações com garantia de qualidade na entrega de dados. O ACE é um *framework* de um WG do IETF baseado nas interações do protocolo OAuth 2.0, usando *tokens* e terminais que não interagem com o ambiente. O objetivo é fornecer o uso padronizado dos requisitos de segurança, autenticação e autorização através dos protocolos OAuth 2.0 e CoAP em ambientes os quais não é possível utilizar o HTTP. A estrutura garante que os dispositivos com restrições de recursos, que conectados entre si compõem as redes de componentes restritos, possam ser autenticados e autorizados, respectivamente, para acessar determinados recursos do sistema.

A especificação do *framework* ACE define requisitos de autorização para o ambiente IoT. O Framework ACE é formado por quatro blocos básicos: OAuth 2.0, CoAP,

Tabela 2. Comparação dos protocolos de comunicação e de segurança entre a Internet e a IoT e as camadas OSI clássicas - Fonte: Editada e traduzida pelo autor do original [Cirani et al. 2013]

Camadas	Ambiente Restrito OSI	Internet OSI	Ambiente Restrito Protocolo SEG	Internet Protocolo SEG
Aplicação	CoAP Requests/Responses Messages	HTTP	CoAPs Requests/Responses Messages	HTTPs
Transporte	UDP	TCP	DTLS	TLS
Rede	IPv6/6LowPAN	IP	IP/Ipsec/HIP	IP/Ipsec/HIP
Link	MAC	MAC	MAC	MAC
Física	PHY	PHY	PHY	PHY

CBOR e COSE. No OAuth 2.0, a comunicação entre o *token*, o cliente e o *authentication service* (AS) é realizada via uma URI. O *framework* ACE **recomenda** o uso do CoAP e uma alternativa para o uso da URI: o cliente utiliza uma mensagem CBOR enviando o *payload* via requisição *POST*. O OAuth 2.0 utiliza mensagem JSON [T. Bray 2014] no *payload* como resposta para o cliente; o ACE **exige** o uso do CBOR e, dependendo do conteúdo, utiliza algum tipo de criptografia.

O *framework* é composto por perfis e extensões: o ***Credential Provisioning*** assume a falta de uma infraestrutura comum de distribuição de chaves. São providas credenciais de autenticação mútua pela AS; ***Proof-of-Possession*** por padrão, implementa a prova de posse, vinculando uma chave ao *token*, que comprova sua posse por um dispositivo; ***ACE Profiles*** fazem limitação dos códigos ou protocolos que um *Client* (C) ou *Resource Server* (RS) suporta. Um *Authentication Server* (AS) gerencia e compara a compatibilidade das escolhas feitas por um determinado cliente durante a sua interação com RS. O AS especifica por parâmetros no *token* de resposta como as autenticações mútuas são feitas entre C e o RS; ***OAuth 2.0*** *requer que os dados trocados com o AS sejam criptografados e protegidos por integridade.* Além disso, é requerido que o AS e o ponto final que se comuniquem com ele (cliente ou RS) realizem a autenticação mútua. Recomenda-se o uso do CoAP como alternativa aos parâmetros URI: o remetente (cliente ou RS) codifica os parâmetros de sua solicitação como um mapa CBOR e envia esse mapa como a carga útil do pedido POST [Kaduk et al. 2014].

A dinâmica do *framework* ACE possui os modos de operação a seguir: **Descobrindo servidores de Autenticação**, para determinar o AS encarregado de um recurso hospedado no RS, C pode enviar uma mensagem inicial de Solicitação de Recursos Não Autorizados para RS. RS então nega o pedido e envia o endereço do AS de volta para C; **Concessão de Autorização**, para solicitar um token de acesso, o cliente obtém autorização do proprietário do recurso ou usa suas credenciais de clientes como concessão. A autorização é expressa sob a forma de uma concessão de autorização; **Token de Introspecção (Opcional)**, essa é uma função opcional e fornecida pelo AS. O token é usado pelo RS ou cliente para consultar o AS para metadados sobre um determinado token, como, por exemplo, validade ou escopo. Adaptando para ambientes restritos, usa-se o CBOR; **Token de Acesso**, esse *framework* recomenda o uso do CBOR web token (CWT)

como especificado em [ID.ietf-ace-cbor-web-token]. Com o objetivo de facilitar o processamento offline do acesso dos tokens, requisita-se o "cnf"de [I-D.ietf-ace-cwt-proof-of-possession] e especifica a requisição "scope"para os tokens da Web JSON e CBOR, isso permite acesso aos tokens [Kaduk et al. 2014].

A utilização do CoAP em conjunto com o DTLS propõe uma alternativa viável para a solução de autenticação e autorização neste tipo de dispositivo restrito. O CoAP pode ser utilizado de quatro maneiras:

1. **NoSec**: sem segurança, DTLS desabilitado, uso de IPSec recomendável;
2. **PreSharedKey**: DTLS habilitado, utiliza chaves compartilhadas simétricas, conforme RFC4279 [P. Eronen and H. Tschofenig 2005];
3. **RawPublicKey**: DTLS habilitado, utiliza chaves assimétricas sem um certificado digital, conforme RFC7250 [P. Wouters et al. 2014];
4. **Certificate**: DTLS habilitado utilizando certificado digital X.509 conforme RFC5280 [Cooper et al. 2008].

O CoAP é um protocolo de serviço em que as mensagens são encapsuladas sendo transmitidas através do *payload* do UDP. O CoAP utiliza a porta 5683 como porta padrão. Como o HTTP, o CoAP utiliza o *framework* OAuth 2.0 para realizar o processo de autenticação, fazendo o uso de *tokens* para conceder o privilégio necessário ao recurso solicitado.

O DTLS é utilizado para a comunicação segura dos dados entre os dispositivos. Preservando as características do UDP, esse protocolo garante o requisito de autorização fomentando a segurança na troca de mensagens.

A Figura 4 ilustra o funcionamento do *framework* ACE em conjunto com os protocolos DTLS e CoAP; a utilização destes protocolos é conhecida como CoAPs, utilizando a porta 5684 para a comunicação. A figura mostra que o cliente solicita a autorização para o servidor de autorização. Após as chaves são comparadas e a autorização ao recurso e concedida. Neste momento, com a informação de autorização, o cliente acessa o servidor do recurso. Então o *handshake* é estabelecido e o cliente recebe o recurso, respectivamente.

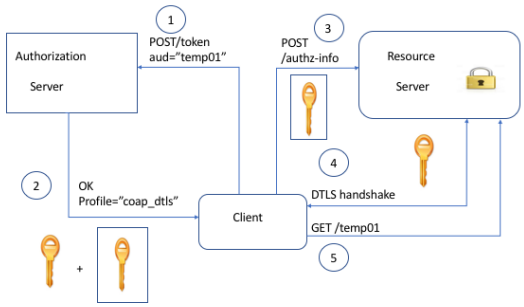

Figura 4. ACE, OAuth 2.0, DTLS - Fonte: [Beltran and Gómez-Skarmeta 2016]

4. Projeto Arduino

O Projeto Arduino é uma plataforma eletrônica *open source* de hardware e software simples e fácil de utilizar. As placas Arduino são projetadas para ler entradas (um sensor de luz, um botão, uma mensagem, por exemplo) e transformar esta informação em uma saída (ativando um motor, acendendo um LED, fazendo uma publicação na Web).

O Arduino nasceu no Ivrea Interaction Design Institute como uma ferramenta fácil para prototipagem rápida, destinada a estudantes sem experiência em eletrônica e programação. Assim que atingiu uma comunidade mais ampla, o painel Arduino começou a mudar para se adaptar às novas necessidades e desafios, diferenciando sua oferta de placas simples de 8 bits para produtos para aplicativos IoT, *wearable*, impressão em 3D e ambientes incorporados. Todas as placas Arduino são completamente *open source*, capacitando os usuários para construí-las de forma independente e eventualmente adaptá-las às suas necessidades particulares. O software é, também, de código aberto e está crescendo através das contribuições de usuários em todo o mundo, conforme a página oficial da plataforma [Arduino 2018].

5. Biblioteca microcoap

Uma proposta de implementação do protocolo CoAP na linguagem C é a biblioteca microcoap [Jaffey 2013], criada no MIT para utilização na plataforma Arduino Mega. A biblioteca não implementa todos os requisitos definidos na RFC 7252 [Shelby et al. 2014], apenas os itens básicos do protocolo(GET/PUT/POST), com o objetivo de realizar testes de instalação e utilização do protocolo em um dispositivo com restrições de recursos.

A biblioteca é composta por um código diminuto e simples, implementando um servidor CoAP; para testes de utilização sugere a utilização do pacote libcoap [Bergmann 2015] no modo cliente. Neste experimento, optou-se por utilizar a biblioteca CoAPthon [G.Tanganelli et al. 2015], pois ocorreram problemas na compilação da libcoap no sistema operacional OSX Yosemite 10.10.3, utilizado no computador.

6. Instalando a Biblioteca microap no Arduino UNO R3

O protocolo CoAP foi criado para utilização em dispositivos com limitações de recursos, conhecidos como dispositivos restritos. A plataforma Arduino torna-se uma boa opção para a realização de testes deste protocolo, pois possui arquiteturas que se encaixam na classe 1(C1) da classificação de dispositivos restritos. Com base nesta classificação, foi definida a seguinte arquitetura para o desenvolvimento do experimento:

- **Arduino UNO R3**: chip ATmega328P, 32KB memória flash, 1KB EEPROM.
- **Ethernet Shield**: fornece endereçamento IP através dos protocolos TCP e UDP utilizando a biblioteca Ethernet Library e SD Library.
- **Cabo USB**: utilizado para conectar o Arduino a um computador. Fornece energia e conexão de dados.
- **Cabo Rede Ethernet**: permite que o Arduino receba um endereço IP válido.
- **IDE Arduino**: aplicação utilizada para configuração e desenvolvimento na plataforma Arduino.
- **Notebook**: dispositivo no qual é instalado a IDE Arduino e através do cabo USB, recebe a conexão física com a placa.

- **WireShark**: *sniffer* de rede, utilizado para analisar os pacotes e demonstrar a utilização do protocolo CoAP.
- **Roteador Operadora**: roteador comercial da operadora de Internet.

6.1. Componentes de Hardware e Software Utilizados no Experimento

Os componentes listados abaixo foram utilizados na montagem da arquitetura do experimento:

1. **Hardware:** (i) computador Mac Book Pro com Mac OS Yosemite 10.10.3; (ii) Arduino UNO R3; (iii) shield Ethernet; (iv) cabo USB; (v) cabo de rede Ethernet; (vi) roteador comercial de operadora de Internet
2. **Software:** (i) download da IDE Arduino 1.8.5 [2]; (ii) download da biblioteca microcoap [3]; (iii) download do sniffer WireShark [4]; (iv) download da biblioteca CoAPthon [5]
3. **Montagem Hardware:** (i) conectar Ethernet shield na placa Arduino UNO R3; (ii) conectar cabo USB no computador; (iii) conectar cabo USB na placa Arduino UNO R3; (iv) conectar cabo de rede Ethernet na placa Arduino UNO R3 e em uma entrada do roteador comercial da operadora
4. **Instalação/Configuração software:** (i) instalar IDE do Arduino; (ii) descompactar biblioteca microcoap; (iii) importar biblioteca microcoap na IDE do Arduino(*menu sketch da IDE, opção incluir biblioteca, opção adicionar biblioteca .ZIP*); (iv) compilar biblioteca microcoap(*opção verificar na IDE Arduino*); (v) fazer o upload da biblioteca para a placa Arduino UNO R3(*opção carregar na IDE Arduino*; (vi) neste momento, a biblioteca CoAP servidor espera requisições; (vii) descompactar biblioteca CoAPthon no MAC OS Yosemite; $(viii)$ compilar a biblioteca CoAPthon [6]; (ix) executar a biblioteca CoAPthon como cliente solicitando uma requisição para o Arduino no endereço IP fornecido por DHCP da operadora de Internet(*python coapclient.py -o GET -p coap://192.168.15.2/basic*); (x) executar o WireShark; (xi) realizar a captura de pacotes na rede criada(*computador, Arduino UNO R3, roteador operadora*); (xii) identificar o pacote CoAP capturado; $(xiii)$ abrir o pacote

Os ítens enumerados acima permitem que o experimento possa ser reproduzido, com variações no hardware utilizado no computador e versão de sistema operacional.

6.2. Pacote Capturado na Experiência

Para comprovação da troca de pacotes entre o Arduino UNO R3 e o computador utilizando o protocolo CoAP, foi utilizado o *sniffer* de rede WireShark para capturar o pacote de rede, como mostra a Figura 5:

7. Dificuldades Encontradas

Ao realizar uma pesquisa na Internet, encontram-se projetos de implementação do protocolo CoAP em vários tipos de bibliotecas nas mais diversas linguagens de programação.

[2] `https://www.arduino.cc/en/Main/Software`
[3] `https://github.com/1248/microcoap`
[4] `https://www.wireshark.org/download.html`
[5] `https://github.com/Tanganelli/CoAPthon`
[6] `https://github.com/Tanganelli/CoAPthon/blob/master/README.md`

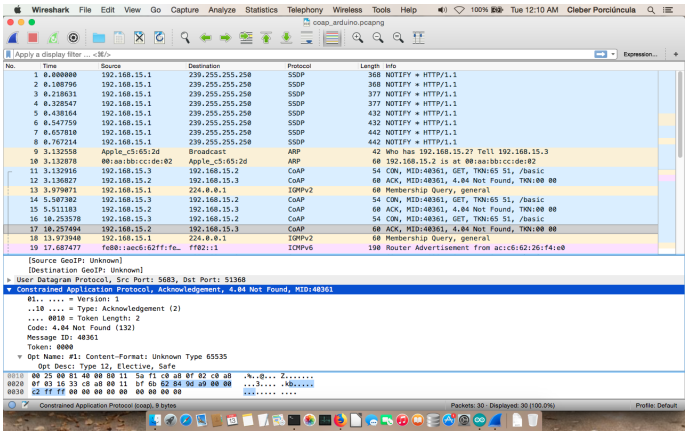

Figura 5. Pacote CoAP - Fonte: Captura realizada pelos autores no experimento

Um bom número destes projetos é destinado à execução em sistemas operacionais, em sua quase totalidade, distribuições de Linux.

Um número reduzido dos projetos é destinado à implementação de bibliotecas em sistemas embarcados, mas, ainda assim, que utilizam algum tipo de sistema operacional, seja uma distribuição Linux ou uma versão criada especificamente para tal dispositivo pela empresa que está desenvolvendo a biblioteca. Os autores deste artigo encontraram propostas de bibliotecas CoAP para a arquitetura Arduino, que são, na realidade, adaptações de bibliotecas desenvolvidas para um sistema operacional que executa em um ambiente com maiores recursos.

A biblioteca utilizada neste artigo (microcoap) foi desenvolvida para a arquitetura de um Arduino MEGA [7], que, embora seja uma placa com limitações de recursos, não se encaixa em uma definição de arquitetura restrita. Ainda não foi encontrada uma biblioteca CoAP desenvolvida para um ambiente restrito computacionalmente, seja para a arquitetura Arduino ou outro tipo de arquitetura.

7.1. Problemas Físicos e de Compilação

A comunicação com o Arduino é realizada via porta serial por meio de um cabo USB (*Universal Serial Bus*). Salienta-se a necessidade de atenção com relação a compatibilidade do mesmo com o Arduino. Não são incomuns problemas de comunicação ocasionados por esses cabos.

Ao compilar a biblioteca microcoap na IDE, foi sinalizado que o código não era compatível com o Arduino UNO, pois utilizava componentes e estruturas que, quando carregados, consumiam toda a memória *flash* da placa. Dos 32KB da placa, 0.5KB são utilizados no *bootloader*, restando 31.5KB para execução de programas. Se for considerado 80%(25.2KB) como limite para programas e os restantes 20%(6.3KB) para variáveis, um código para execução nesta plataforma precisa ser bem otimizado.

A página da biblioteca microcoap alerta para problemas em outras placas Arduino; a solução encontrada foi alterar o código endpoints.c e diminuir uma variável do tipo *char*:

Original: static char rsp[1500] = ""; Alterado: static char rsp[500] = "";

[7] https://www.robocore.net/loja/produtos/arduino-mega-2560-r3.html

Após realizar a alteração, o código foi compilado com sucesso e dentro dos recursos da placa Arduino utilizada, conforme informado pela IDE. O *upload* da biblioteca ocorreu sem problemas.

8. Perpectivas Futuras

Neste primeiro momento, o desafio de implementar o CoAP no Arduino foi contemplado satisfatoriamente. Desse ponto em diante o foco aponta para os esforços no sentido de incorporar mecanismos de segurança nesse dispositivo.

Para tal, espera-se realizar a implementação, por meio de uma adequação como a realizada nesse trabalho, de um protocolo que garanta a segurança do ambiente, juntamente com o CoAP.

9. Conclusão e Trabalhos Futuros

A utilização do protocolo CoAP no dispositivo restrito Arduino foi possível. Uma adaptação no código da biblioteca possibilitou o seu funcionamento e testes posteriores. Todavia, sem contemplar nenhum requisito de segurança.

Como próxima etapa, a intenção é implementar segurança, através do protocolo DTLS. Para tanto, implementar primeiro o CoAP no módulo ESP8266, mostrado na Figura 2, é uma alternativa. Assim, poderão ser avaliados o comportamento do protocolo nesses dois dispositivos, definindo-se, então, o melhor para a implementação do DTLS.

Referências

[Arduino 2018] Arduino (2018). What is arduino. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>.

[Beltran and Gómez-Skarmeta 2016] Beltran, V. and Gómez-Skarmeta, A. F. (2016). An overview on delegated authorization for coap: Authentication and authorization for constrained environments (ace). *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 706–710.

[Bergmann 2015] Bergmann, O. (2015). libcoap. Disponível em: <https://libcoap.net/>.

[Bormann 2012] Bormann, C. (2012). Using CoAP with IPsec draft-bormann-core-ipsec-for-coap-00. Draft 00, CoRE Working Group.

[Bormann et al. 2014] Bormann, C., Ersue, M., and Keranen, A. (2014). Terminology for Constrained-Node Networks. RFC 7228, RFC Editor.

[Cirani et al. 2013] Cirani, S., Ferrari, G., and Veltri, L. (2013). Enforcing security mechanisms in the ip-based internet of things: An algorithmic overview. *Algorithms*, 6(2):197–226.

[Cooper et al. 2008] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor.

[D. Hardt 2012] D. Hardt, E. (2012). The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor.

[de Oliveira 2017] de Oliveira, S. (2017). *Internet das Coisas com ESP8266, Arduino e Raspberry Pi*. Novatec Editora.

[Dierks and Rescorla 2008] Dierks, T. and Rescorla, E. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor.

[G.Tanganelli et al. 2015] G.Tanganelli, Vallati, C., and E.Mingozzi (2015). Coapthon: Easy development of coap-based iot applications with python. *IEEE World Forum on Internet of Things*.

[Hoffman 2013] Hoffman, P. (2013). O Tao do IETF. RFC 00, RFC Editor.

[Jaffey 2013] Jaffey, T. (2013). microcoap. Disponível em: <`https://github.com/1248/microcoap`>.

[Kaduk et al. 2014] Kaduk, B., Schaad, J., and athleen Moriarty (2014). Authentication and Authorization for Constrained Environments. RFC 00, RFC Editor.

[P. Eronen and H. Tschofenig 2005] P. Eronen, E. and H. Tschofenig, E. (2005). Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279, RFC Editor.

[P. Wouters et al. 2014] P. Wouters, E., H. Tschofenig, E., Gilmore, J., Weiler, S., and Kivinen, T. (2014). Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7250, RFC Editor.

[Postel 1980] Postel, J. (1980). User Datagram Protocol. RFC 768, RFC Editor.

[R. Fielding and J. Reschke 2014] R. Fielding, E. and J. Reschke, E. (2014). Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230, RFC Editor.

[Rescorla and Modadugu 2012] Rescorla, E. and Modadugu, N. (2012). Datagram Transport Layer Security Version 1.2. RFC 6347, RFC Editor.

[Seitz et al. 2016] Seitz, L., Gerdes, S., Selander, G., Mani, M., and Kumar, S. (2016). Use Cases for Authentication and Authorization in Constrained Environments. RFC 7744, RFC Editor.

[Shelby et al. 2014] Shelby, Z., Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor.

[Simmonds et al. 2004] Simmonds, A., Sandilands, P., and Van Ekert, L. (2004). An ontology for network security attacks. In *Asian Applied Computing Conference*, pages 317–323. Springer.

[T. Bray 2014] T. Bray, E. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, RFC Editor.

[Vasseur 2014] Vasseur, J. (2014). Terms Used in Routing for Low-Power and Lossy Networks. RFC 7102, RFC Editor.

[Wei et al. 2018] Wei, L., Cheolwoo, J., and Jongtae, P. (2018). Iot healthcare communication system for ieee 11073 phd and ihe pcd-01 integration using coap.