

UNIVERSIDADE DE SÃO PAULO  
ESCOLA DE ENGENHARIA DE SÃO CARLOS

SINDÉLIO HENRIQUE LIMA

Internet das “coisas”: controlando atuadores através de páginas na Internet

São Carlos  
Novembto de 2017



# SINDÉLIO HENRIQUE LIMA

Internet das “coisas”: controlando atuadores através de páginas na Internet

Monografia apresentada ao Curso de Engenharia de Computação, da Escola de Engenharia de São Carlos e Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro de Computação.

Orientador: Prof. Dr. Maximiliam Luppe

São Carlos  
Novembro de 2017



AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,  
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS  
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

L732i Lima, Sindélio Henrique  
Internet das "coisas": controlando atuadores  
através de páginas na Internet / Sindélio Henrique  
Lima; orientador Maximiliam Luppe. São Carlos, 2017.

Monografia (Graduação em Engenharia de Computação)  
-- Escola de Engenharia de São Carlos e Instituto de  
Ciências Matemáticas e de Computação da Universidade de  
São Paulo, 2017.

1. Internet das coisas.
2. Segurança.
3. Atuadores.
4. Conectividade.
5. Controle.
- I. Título.



## FOLHA DE APROVAÇÃO

**Nome:** Sindélio Henrique Lima

**Título:** "Internet das "coisas": controlando atuadores através de páginas na Internet"

**Trabalho de Conclusão de Curso defendido em 28/11/2017.**

**Comissão Julgadora:**

Prof. Dr. Maximiliam Luppe  
(Orientador) - SEL/EESC/USP

**Resultado:**

Aprovado

Prof. Dr. Fernando Santos Osório  
SSC/ICMC/USP

Aprovado

Profa. Associada Kalinka Regina Lucas Jaquie  
Castelo Branco  
SSC/ICMC/USP

Aprovado

**Coordenador do Curso Interunidades Engenharia de Computação:**

*Prof. Dr. Maximiliam Luppe*



*Dedico esta obra aos meus pais,  
pois mesmo sem planos me  
trouxeram a este mundo e fizeram o  
melhor para minha criação.*



## AGRADECIMENTOS

Ao Prof. Dr. Maximiliam Luppe, que me apoiou em projetos de monitoria, ensino e finalmente no projeto de conclusão de curso. Neste último com sua orientação e concessão de uma placa Intel Edison, na qual o projeto foi desenvolvido.

A Juliana Karoline de Sousa por emprestar um kit de sensores e atuadores para o desenvolvimento do projeto.

Ao Josias Blos, por contribuir com ideias e discussões acerca do projeto.

Ao Prof. Dr. Adalberto Panobianco Bergamasco, por suas humildade e caridade inspiradoras.

A todos os Professores que contribuíram com minha formação, de forma direta ou indireta.

À Silvana e Jussarae aso demais secretários do curso de Engenharia de Computação e Engenharia Elétrica, por sua ajuda prestada ao longo do curso.

A USP por me acolher como morador, estudante e cidadão.



*“The Internet of Things has the potential to  
change the world, just as the Internet did.  
Maybe even more so”*

Kevin Ashton [1]



## RESUMO

LIMA, S. H. **Internet das “coisas”: controlando atuadores através de páginas na Internet.** 2017. 72p. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos e Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2017.

Este projeto consiste no controle de um atuador por meio de uma página na Internet. O atuador escolhido para testes foi uma buzina, devido a sua disposição e facilidade de uso. Um sistema foi proposto e desenvolvido com base em requisitos de funcionalidade e segurança levantados. A segurança do sistema criado foi um princípio de desenvolvimento considerado ao longo de todo o projeto. A plataforma de desenvolvimento do projeto consistiu em uma placa Intel Edison, como *hardware* de prototipação, e na Intel XDK, como *software* no qual foram escritos a maioria dos programas do sistema criado. A linguagem de programação principal foi o Node.js, mas também foram usadas HTML, CSS, Javascript e bash. No sistema criado, um cliente se comunica via Internet com um servidor, que controla localmente o atuador de acordo com os comandos dados pelo cliente. O servidor é executado na placa Intel Edison, num ambiente Linux. Neste cenário, basta ao cliente possuir conectividade com a Internet e credenciais autorizadas para poder acessar e controlar o atuador. O cliente usado para testes foi o *notebook* do Autor. Usou-se criptografia na comunicação via Internet entre o cliente e o servidor. O sistema criado atingiu as expectativas de funcionalidade e segurança, porém há margem para passos futuros. Conclui-se que controlar atuadores pela Internet é perigoso devido às fragilidades de segurança. As maiores fragilidades de segurança estão nas interfaces *web*, e grande atenção deve ser dada a estas. Possibilidades de próximos passos: subir o servidor para uma nuvem computacional, controlar múltiplos atuadores simultaneamente, controlar atuadores mais complexos.

Palavras-chave: 1. Internet das coisas. 2. Segurança. 3. Atuadores. 4. Conectividade. 5. Controle.



## ABSTRACT

LIMA, S. H. **Internet of “things”: controlling actuators through web pages.** 2017. 72p. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos e Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2017.

This project consists of the control of an actuator by means of an Internet page. A buzzer was chosen as the actuator for test purposes, mainly due to its availability and ease of use. A system was proposed and developed based on raised requisites of security and functionality. The security of the created system was a design principle followed during the entire project. The project development platform consisted of an Intel Edison board, as prototyping hardware, and the Intel XDK, as software in which most of the programs of the system were wrote. The main programming language was the Node.js, but were also used HTML, CSS, Javascript and bash. In the created system, a client communicates via Internet with a server, that which controls locally the actuator according to the commands given by the client. The server is executed at the Intel Edison board, in a Linux ambient. In this scenario, The client need only have connectivity to the Internet and authorized credentials to be able to access and control the actuator. The Author's notebook was used as a client for tests. Cryptography was used in the communication via Internet between the client and the server. The created system reached the expectations of functionality and security, however there is margin for future work. The conclusion is that controlling actuators through the Internet is dangerous due to security fragilities. The biggest fragilities in security are in the web interfaces, and greater attention should be dedicated for those. Possibilities for next steps: upload the server to a computer cloud, to control multiple actuators simultaneously, to control more complex actuators.

Keywords: 1. Internet of things. 2. Security. 3. Actuators. 4. Connectivity. 5. Control.



## LISTA DE ILUSTRAÇÕES

Figura 1 – Número de dispositivos conectados à Internet ao longo dos anos.....	39
Figura 2 – Número de nós seguros e inseguros conectados à Internet ao longo dos anos..	40
Figura 3 – Placa de desenvolvimento Intel Edison.....	44
Figura 4 – Kit de sensores e atuadores Grove <i>Starter Kit Plus</i> .....	45
Figura 5 – Ambiente de projetos da Intel XDK.....	46
Figura 6 – Ambiente de desenvolvimento da Intel XDK.....	46
Figura 7 – Verificação da porta USB correta para comunicação serial com a Edison.....	49
Figura 8 – Configuração do Putty para comunicação serial com a Edison.....	50
Figura 9 – Terminal de comandos para a Edison.....	50
Figura 10 – Arquitetura do sistema.....	51
Figura 11 – BurpSuite após ataque de força bruta.....	57
Figura 12 – LOIC configurado para realizar enchente de pacotes UDP .....	58
Figura 13 – Relatório de vulnerabilidades criado pelo OpenVAS.....	59
Figura 14 – Tentativa de uso do scaneador Acunetix online.....	60
Figura 15 – Relatório de scaneamento de portas UDP gerado pelo nmap online.....	60
Figura 16 – Tentativa de scaneamento com o nikto online.....	61



## LISTA DE TABELAS

Tabela 1 – Comparaçao entre as plataformas usuais de projetos IoT.....	45
Tabela 2 – Comandos mais comuns para a Edison.....	51



## LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programmer Interface</i>
CSS	<i>Cascade Style Sheets</i>
DDoS	<i>Distributed Denial of Service</i>
EESC	Escola de Engenharia de São Carlos
FTDI	<i>Future Technology Devices International</i>
HTML	<i>Hyper Text Markup Language</i>
HTTPS	<i>Hyper Text Transfer Protocol Secure</i>
ICMC	Instituto de Ciências Matemáticas e de Computação
Intel XDK	<i>Intel Cross platform Development Kit</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol address</i>
IPv6	<i>Internet Protocol version 6</i>
LOIC	<i>Low Orbit Ion Cannon</i>
MQTT	<i>Message Queueing Telemetry Transport</i>
M2M	<i>Machine To Machine</i>
MRAA	<i>low level skeleton library for communication on GNU/Linux platforms</i>
OASIS	<i>Organization for the Advancement of Structured Information Standards</i>
OpenSSL	<i>Open Secure Sockets Layer</i>
OpenVAS	<i>Open Vulnerability Assessment Scanner</i>
OWASP	<i>Open Web Application Security Project</i>
RSA	<i>Rivest Shamir and Adelman</i>
ROM	<i>Read Only Memory</i>
SO	Sistema Operacional
SSH	<i>Secure Shell</i>
SQL	<i>Structured Query Language</i>
TI	Tecnologia da Informação
TCP	<i>Transmission Control Protocol</i>
TEDx	<i>Techonology, Entertainment and Design independent presentation</i>
TO	Tecnologia Operacional
UDP	<i>User Datagram Protocol</i>
URL	<i>Universal Resource Locator</i>
USB	<i>Universal Serial Bus</i>
XSRF	<i>Cross Site Request Forgery</i>
XSS	<i>Cross Site Scripting</i>
WPA2	<i>Wifi Protected Access 2</i>



## SUMÁRIO

1 INTRODUÇÃO.....	32
1.1 Motivações.....	32
1.2 Objetivos.....	32
2 REVISÃO BIBLIOGRÁFICA.....	35
2.1 O que é a IoT.....	35
2.2 Perspectiva econômica da IoT.....	39
2.3 Desafios a serem superados pela IoT.....	39
2.4 Vulnerabilidades recorrentes em sistemas IoT.....	41
3 MATERIAIS E MÉTODOS.....	44
3.1 Plataformas.....	44
3.2 Ferramentas de Desenvolvimento.....	45
4 IMPLEMENTAÇÃO.....	48
4.1 Instalação dos <i>drivers</i> para comunicação com a placa Intel Edison.....	48
4.2 Atualização do <i>firmware</i> da placa Intel Edison.....	48
4.3 Estabelecendo conexão serial entre o computador e a placa Intel Edison.....	49
4.4 Estabelecendo conexão entre a Internet e a placa Intel Edison.....	51
4.5 Arquitetura do sistema.....	51
4.6 Desenvolvimento dos programas necessários ao sistema.....	53
5 TESTES E RESULTADOS.....	55
5.1 Testes de funcionalidade.....	55
5.2 Testes de segurança.....	56
6 CONCLUSÕES.....	62
REFERÊNCIAS.....	64
ANEXO A – Códigos.....	69
ANEXO B – Testes básicos em sistemas IoT.....	70



## 1 INTRODUÇÃO

Nas últimas décadas o modo como nos comunicamos, nos relacionamos, produzimos, enfim, como vivemos, mudou, e grande parte de tais mudanças deve-se à evolução exponencial da tecnologia e da ciência. Em meio a esta evolução surgiu a área “Internet das Coisas” [2], resultado da convergência do avanço tecnológico em diversas frentes. No contexto atual, a “Internet das Coisas”, ou no Inglês *Internet of Things* – IoT, como é mais conhecida, vêm proporcionando mais conforto ao usuário final, mais economia às empresas e modelos de negócios inviáveis anteriormente. Entretanto, a IoT possui desafios a serem superados para sua implementação, em particular na segurança de suas aplicações. Questões como uso autorizado do sistema, confidencialidade de informações, integridade e disponibilidade dos sistemas são dificuldades atuais da IoT e serão abordadas neste trabalho.

### 1.1 Motivação

A IoT mostrou demanda crescente nos últimos anos e perspectiva de crescimento continuado nos próximos – ver seção 2.2 Perspectiva Econômica da IoT. Também, a IoT traz grandes desafios na segurança de suas aplicações, e consequentemente grandes oportunidades para proposição de soluções acerca deste problema. Ainda, a Engenharia de Computação fornece boa base para um profissional ou especialista em IoT, que abrange desde o *hardware*, como sensores e atuadores, até o *software*, como protocolos de comunicação e aplicações clientes. Tais tópicos são contemplados no curso de Engenharia de Computação. Por fim, IoT é a área de atuação e interesse do Autor.

### 1.2 Objetivos

Este projeto visa a exploração do conceito da IoT e suas questões de segurança, bem como a criação de um sistema de IoT básico, ou simplesmente sistema IoT, multiuso e o mais seguro possível. Especificamente, o objetivo é estabelecer controle sobre o funcionamento de um atuador via página da Internet, de forma segura e precisa. Neste contexto, o atuador é uma “coisa” – ver seção 2.1 O que é a IoT – controlada mediante a Internet. O atuador controlado pode ser substituído por outro, com características elétricas similares, e o sistema criado deve

controlá-lo com sucesso com pouca ou nenhuma alteração. Desta forma, o sistema criado é multiuso pois independe do dispositivo controlado.

Controlar um atuador pela Internet implica em muitos riscos. Caso um atacante domine o sistema, ele pode ativar o atuador da forma que desejar. Se o atuador tiver capacidade para se danificar ou danificar o sistema, ou ainda pior, ferir um ser humano, então o sistema apresenta riscos altos. Um exemplo de atuador deste tipo é um braço mecânico, que se usado de forma indevida pode causar tragédias dentro das organizações. O controle de atuadores deve ser realizado com a maior segurança possível, inclusive considerando as capacidades que um atacante possa exercer pela Internet sobre o atuador, de forma a limitá-los.

Desta forma, a segurança do sistema é um princípio de desenvolvimento e será considerada do início ao fim do projeto, com objetivo de se garantir a confidencialidade das informações comunicadas via Internet, a integridade e a disponibilidade do sistema.

O capítulo 2 traz um panorama sobre o estado atual da IoT e seus maiores desafios. O capítulo 3 mostra os recursos e métodos usados no projeto. O capítulo 4 elabora a implementação do projeto, desde a preparação para uso dos componentes, passando pela proposição de um sistema, até a escrita dos programas pertinentes ao funcionamento do sistema proposto. O capítulo 5 trata dos testes e resultados obtidos acerca das funcionalidades e segurança do sistema criado. O capítulo 6 expõe as conclusões sobre os trabalhos.



## 2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta conhecimentos e informações sobre IoT, com o foco em sua definição, história, aplicações, perspectiva econômica e desafios.

### 2.1 O que é a IoT?

O termo IoT foi cunhado em 1999 por Kevin Ashton [1], que então falara do potencial da IoT trazer mudanças tão significativas ao mundo quanto a Internet trouxe. Para Ashton, uma “coisa”, ou *thing*, corresponde à união de um dispositivo eletrônico com um sistema que embarca este dispositivo. Como exemplo, uma coisa pode ser uma cafeteira, um animal, um foguete e etc, desde que estes estejam juntos fisicamente de um dispositivo eletrônico capaz de transferir dados. Ashton também observa que o ser humano possui limitações de tempo, atenção e precisão, o que implica em uma capacidade menor de captura de dados pelos seres humanos do que computadores. Em contraponto, estes computadores dedicados à captura de dados são mais eficazes nesta tarefa, e seu uso possibilita um aumento na quantidade de dados capturados e em sua precisão. Por sua vez, uma maior quantidade de dados implica na diminuição de perdas e custos em processos industriais ou organizacionais, pois os dados são usados na otimização destes.

Em contraste com a definição de Ashton para uma “coisa”, o Dr. John Barrett oferece outra definição para este termo em sua palestra TEDx [3]. Para Barrett, uma “coisa” pertencente a um sistema IoT é uma entidade física que possua as seguintes propriedades:

1. Identificação única no sistema. O Dr. Steve Leibson afirma que o IPv6 [4] permite associar um identificador único a cada átomo na superfície do planeta Terra e ainda haveriam identificadores para fazer o mesmo com mais de cem Terras [5]. Desta forma, o IPv6 supre completamente as necessidades de identificadores únicos para as “coisas” em sistemas IoT.
2. Conectividade com a internet. Esta conectividade pode ser com ou sem fio, no entanto sem fio é mais comum em sistemas IoT.
3. Sentidos. Sentidos são dados às “coisas” por meio de sensores.

4. Interface de controle. Computadores, monitores com toque, terminais de comando e controles remotos são exemplos de interfaces de controle. Usualmente estas interfaces acompanham alguma unidade de processamento.

A definição de Ashton acerca de uma “coisa”, no contexto da IoT, contém a definição de Barrett, que é mais restritiva e específica. Ainda, caso uma “coisa” possa exercer ações no meio em que está inserida, esta se trata de um atuador. Caso a “coisa” não possa exercer ações no meio, esta se trata de um sensor.

Dada uma definição de “coisa”, define-se a IoT como um sistema de “coisas” que possuem a capacidade de transferir dados umas às outras, via Internet ou rede dedicada, com ou sem ação humana. Normalmente a transferência de dados entre as “coisas” é sem fio.

Margaret Rouse afirma em seu artigo [6] no website TechTarget que a IoT é o resultado da convergência da evolução de tecnologias sem fio, sistemas eletromecânicos como sensores e atuadores, microserviços e a própria Internet. Esta convergência aproxima a tecnologia operacional – TO, da tecnologia da informação – TI, permitindo que dados operacionais sejam analisados e gerem *insights* para as organizações. Diversas atividades se beneficiam da aplicação de IoT, especialmente: manutenção preventiva de dispositivos, automatização e otimização de processos.

O website Analytics Vidhya [7] cita 10 exemplos de aplicações da IoT, sendo elas:

1. Casas inteligentes.

Conectar os dispositivos eletrodomésticos na Internet confere aos seus usuários maior conforto, economia e opções de uso. Por exemplo, o dono de uma casa inteligente pode desligar as luzes de sua casa remotamente, caso estejam ligadas, levando a uma economia de energia elétrica. Outro exemplo, ilustrando o aumento no conforto dos usuários, é agendar o café para estar pronto quando se chega em casa. Todos os dispositivos eletrodomésticos podem se tornar “coisas”.

2. Dispositivos vestíveis.

Dispositivos vestíveis, como relógios inteligentes, adesivos inteligentes, entre outros, trazem funcionalidades interessantes aos usuários, como o monitoramento do seu estado físico e vital, e novas formas de entretenimento. Existem até adesivos para

medicação automática em determinados horários, com dosagem precisa. Os dispositivos vestíveis também tem visto um grande crescimento na moda.

### 3. Carros conectados.

A comunicação entre os carros, máquina-máquina, facilita o tráfego e ajuda a evitar acidentes. Por exemplo, um carro pode sinalizar aos demais nas redondezas que irá iniciar uma baliza, evitando acidentes causados pela distração humana. Os carros também podem trocar informações entre si para diminuir o tráfego por meio de adaptações nas rotas, evitando congestionamentos.

### 4. IoT industrial

Dados coletados por sensores dentro das fábricas, aliados a *big data analytics*, *machine learning* e *deep learning* têm trazido otimizações para os processos industriais. Além disso, a manutenção preventiva vem ganhando espaço com sensores destinados exclusivamente ao monitoramento do estado das máquinas e outros aparelhos fabris.

### 5. Cidades inteligentes

Distribuição de água, segurança urbana, atendimento emergencial, monitoramento ambiental, transporte público, coleta de lixo e distribuição de energia elétrica são exemplos de serviços urbanos que se beneficiam da quantidade de dados disponibilizada pelos sensores de IoT. De forma semelhante à IoT industrial, as ferramentas estatísticas, como *big data* e *machine learning*, têm papel importante na geração de *insights* a partir dos dados coletados para a otimização e automatização dos serviços urbanos.

### 6. IoT na agricultura

A demanda global por alimentos tem crescido, seguindo o aumento contínuo na população global. É importante que a agricultura evolua para suprir a demanda por alimentos. O uso de sensores nas plantações permite que ações sejam tomadas para otimizar o plantio e a colheita com base nos dados coletados a respeito do solo, do clima e do estado das plantas.

### 7. Varejo inteligente

A IoT fornece novas possibilidades de experiência aos consumidores dentro das lojas, através do uso de dados coletados por sensores dentro das unidades de varejo. Por exemplo, um cliente pode ser atendido ou não por um atendente, de acordo com suas preferências, determinadas anteriormente por uma câmera. Os dados coletados nas lojas físicas são outra fonte de informações para os lojistas, que em sua quase totalidade possuem lojas na Internet. A gestão de estoque também tem se beneficiado da aplicação da IoT.

#### 8. Engajamento energético

O futuro da distribuição de energia elétrica é distribuído. Sensores poderão detectar pontos de falha, gerar rotas alternativas e redirecionar a energia para que virtualmente nunca falte energia para a população. Além disso, cada usuário pode vender energia para a rede caso possua uma forma de geração em sua residência, como um painel solar, e sensores que meçam o quanto foi vendido. A IoT fará com que todos participem mais ativamente da geração e distribuição de energia.

#### 9. IoT na saúde

A aplicação da IoT na saúde permanece um gigante adormecido. A coleta de dados sobre cada indivíduo permitirá tratamentos individuais e com rápida resposta em caso de emergências. Espera-se que nos próximos anos a saúde veja muitas inovações advindas da IoT.

#### 10. IoT na agropecuária

Assim como a agricultura, a agropecuária também se beneficia dos dados disponibilizados pelos sensores da IoT. Por exemplo, sensores detectam se algum animal está doente e informam aos cuidadores, para que este seja isolado e tratado antes que outros animais sejam infectados.

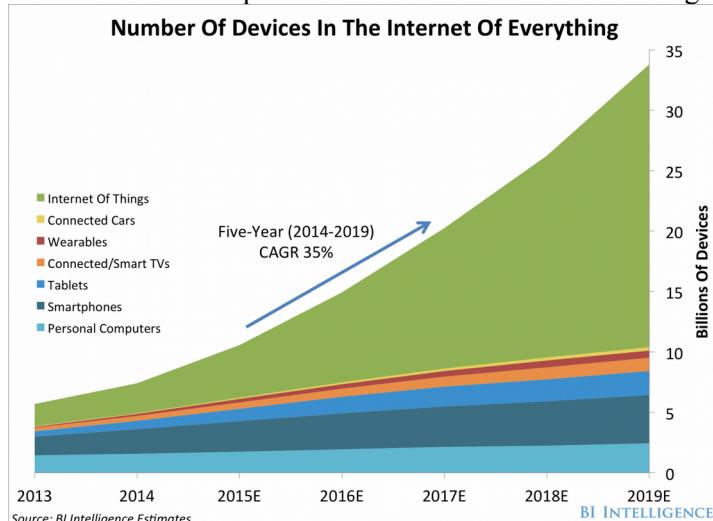
Todos estes exemplos possuem o seguinte aspecto em comum: a aplicação da IoT agregou valor ao produto ou serviço em questão. Além disso, as aplicações de IoT são beneficiadas pela utilização de inteligência artificial e tomada de decisões autônoma, com *deep learning* e *machine learning*, porém manteremos o escopo em sistemas IoT sem inteligência artificial.

## 2.2 Perspectiva Econômica da IoT

De acordo com o relatório da consultoria da informação Gartner [8], por volta de 2020 haverão mais de 30 bilhões de dispositivos conectados à Internet em uso, contra 8,4 bilhões atualmente. A Gartner também afirma neste relatório que em 2020 o setor de IoT corresponderá a \$1,9 trilhão em valor agregado, correspondendo a aproximadamente 2% do valor do produto interno bruto global neste ano.

O *Business Insider*, website americano de notícias, novidades e tendências em negócios, prevê o crescimento da quantidade de dispositivos conectados à Internet ao longo dos anos como mostrado na figura 1.

Figura 1 – Número de dispositivos conectados à Internet ao longo dos anos.



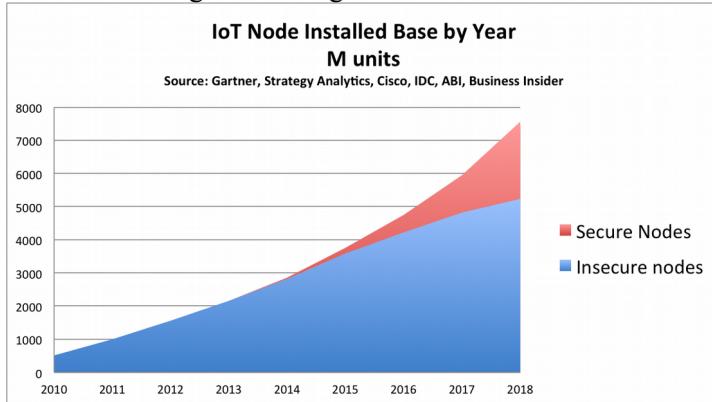
Fonte – BUSINESS INSIDER INTELLIGENCE [9].

Observa-se um crescimento global médio anual de 35% de sensores, atuadores, eletrodomésticos e outras “coisas” conectadas à Internet. Nota-se que a *Business Insider* separou computadores pessoais, smartphones, tablets, smarTVs, Wearables e carros interconectados de outras “coisas” para enfatizar o crescimento exponencial da quantidade de “coisas” distintas conectadas à Internet. De fato, estas “coisas” incomuns e conectadas à Internet possuem o maior crescimento em quantidade global quando comparadas às “coisas” mais comuns.

## 2.3 Desafios a serem superados pela IoT

A IoT ainda possui desafios a serem superados para sua implantação completa. A maior dificuldade enfrentada por sistemas IoT é sua fragilidade em termos de segurança. A figura 2 mostra a crescente preocupação com segurança, manifestada no aumento de nós, ou “coisas”, seguros conectados à Internet.

Figura 2 – Número de nós seguros e inseguros conectados à Internet ao longo dos anos.



Fonte – GARTNER et al. (2014). [10]

Uma das questões mais relevantes de segurança para IoT consiste no domínio das “coisas” constituintes de um sistema IoT para formar uma rede de robôs, ou *botnet* em Inglês, com a finalidade de realizar ataques distribuídos de interrupção de serviço, *Distributed Denial of Service* – DDoS. Em Outubro de 2016 ocorreu um ataque DDoS de larga escala [11] contra a companhia americana Dyn, que presta serviços de nuvem e de resolução de domínios para empresas proeminentes como Amazon, Twitter, Reddit, Spotify, PayPal, Etsy, GitHub, Netflix, Shopify e SoundCloud. O ataque, nomeado “*Mirai botnet*”, foi referido como “Provável maior da história de seu tipo” e deixou fora do ar por um dia os sites das empresas citadas.

Outra dificuldade é o crescimento exponencial da quantidade de dispositivos interconectados, que traz dificuldades em escalabilidade. Estas dificuldades estão sendo gradualmente superadas por meio do avanço em pesquisas sobre padronização e interoperabilidade. Como exemplo de padronização recente pode-se citar o estabelecimento do *Message Queueing Telemetry Transport* – MQTT [12], como algoritmo padrão de comunicação máquina-máquina – M2M, em sistemas IoT pela *Organization for the Advancement of Structured Information Standards* - OASIS [13] em Novembro de 2014. Com a resolução das dificuldades de padronização e interoperabilidade, a escalabilidade em sistemas IoT ocorrerá exponencialmente.

Uma terceira barreira para o desenvolvimento de sistemas IoT é a preocupação com a privacidade dos usuários das “coisas”. Estas “coisas” possuem capacidade de coletar dados acerca de seu uso e seus usuários, e há riscos para estes no caso de seus dados serem usados de formas maliciosas, como furtos coordenados e espionagem. Um ponto vital para garantir a privacidade adequada aos usuários de dispositivos IoT é a evolução dos contratos de privacidade para termos mais compreensíveis e seguros aos usuários finais.

## 2.4 Vulnerabilidades recorrentes em sistemas IoT

A comunidade *Open Web Application Security Project* – OWASP estabelece em seu relatório “*Internet of Things Top Ten*” [14] as dez vulnerabilidades mais recorrentes em sistemas IoT, e são elas:

### 1. Interface de rede insegura

Qualquer indivíduo com acesso as interfaces de rede, ou interfaces *web*, de um sistema IoT, pode atacar esta interface. Como normalmente as interfaces estão disponíveis na Internet, esta vulnerabilidade é facilmente explorada por atacantes. Exemplos de ataques em interfaces de rede inseguras são: injeção de comandos ao banco de dados também conhecido como injeção de *Structured Query Language* – SQL [15]; e execução de *scripts* externos, também conhecido como *Cross Site Scripting* – XSS [16].

### 2. Autenticação insuficiente

Fragilidades nos mecanismos de autenticação de sistemas IoT podem ser explorados por atacantes de forma que estes obtêm acesso não autorizado à partes ou ao todo do sistema. Os danos causados por este tipo de vulnerabilidade são severos, visto que uma vez autenticado no sistema, o atacante pode executar comandos neste. Exemplos de ataques a autenticação insuficiente são: ataques de força bruta e engenharia social.

### 3. Serviços de rede inseguros

Serviços externos de rede, como módulos e pacotes destinados a comunicação na rede, muitas vezes são tidos como seguros apesar de apresentar vulnerabilidades. O uso cego das ferramentas externas pode ser perigoso, especialmente de ferramentas criadas por fontes de pouca ou nenhuma credibilidade. Exemplo de ataque a serviços de rede

inseguros: Exploração de um vazamento de memória presente no código de um módulo de comunicação sem fio, criado por terceiros.

#### 4. Falta de encriptação no transporte de dados

Esta é uma das vulnerabilidades mais graves pois um atacante consegue, de forma passiva, farejar todas as informações sendo transportadas na rede, fazendo com que a confidencialidade das informações seja perdida. Felizmente sua solução é relativamente simples, por meio da aplicação de criptografia na comunicação entre os elementos da rede. Exemplo de ataque à comunicações sem encriptação: ataque de *replay*, no qual um pacote é farejado pelo atacante, alterado de acordo com sua vontade, e reenviado ao destinatário original do pacote.

#### 5. Preocupações com privacidade

Caso os dados do sistema não sejam protegidos adequadamente, como num sistema com autenticação insuficiente, atacantes podem obter acesso a dados pessoais de usuários. Exemplo de ataque à privacidade: Atacante consegue obter imagens pessoais de um indivíduo, mediante uma câmera inteligente insegura, e pede resgate para não divulgar as imagens na Internet.

#### 6. Interface de nuvem insegura

As interfaces com nuvens computacionais apresentam outra superfície de contato entre o sistema e a Internet. Devido ao aumento da popularidade e demanda de nuvens computacionais, se faz importante a robustez destas interfaces. Um exemplo de ataque que explora esta vulnerabilidade é a forja de requisições externas, ou *Cross Site Request Forgery* – XSRF [17]. Este ataque força usuários autênticos a executar ações maliciosas no sistema por meio de requisições falsas criadas por um atacante. Dominar uma nuvem computacional é o sonho dos hackers, pois estas possuem poder computacional elevado, e uma *botnet* criada com “coisas” na nuvem é bastante poderosa em termos computacionais.

#### 7. Interface de dispositivos móveis insegura

Similar as interfaces de rede inseguras, porém com as particularidades de dispositivos móveis. Injeção de comandos ao banco de dados e execução de *scripts* externos são, também neste caso, exemplos de ataques que exploram esta vulnerabilidade.

## 8. Configurabilidade de segurança insuficiente

Caso um sistema IoT não possua opções de segurança, como a possibilidade de se usar encriptação ou a possibilidade de requerir senhas fortes, então o sistema possui vulnerabilidades decorrentes da sua falta de configurabilidade.

Exemplo de configurabilidade de segurança insuficiente: Sistemas IoT com pouca ou nenhuma diferenciação entre usuários comuns e administradores, fazendo com que usuários comuns possuam os mesmos privilégios de administradores no sistema.

## 9. *Software* ou *middleware* inseguros

Assim como o uso de módulos ou pacotes de rede de terceiros contendo falhas pode levar à vulnerabilidades na segurança de um sistema IoT, o mesmo ocorre com o uso de *software* ou *middleware* de terceiros. A questão é a mesma: confiar que recursos criados por terceiros sejam seguros. Caso não sejam, vulnerabilidades estarão presentes no sistema desenvolvido. Exemplo de ataque a *middleware* inseguro: O BlueBorne [18], *worm* [19] criado por hackers que ataca o Bluetooth [20] de dispositivos devido a uma vulnerabilidade presente nos programas controladores, ou *drivers*, do Bluetooh 4.0.

## 10. Segurança física frágil

Esta vulnerabilidade ocorre quando um atacante, em contato físico direto com uma “coisa”, consegue obter controle parcial ou total sobre esta. Por exemplo, um atacante pode abrir um interpretador de comandos localmente na “coisa”, e executar ações de forma que a “coisa” se torne parte de uma *botnet*.

Nota-se que as vulnerabilidades mais recorrentes estão na interface entre as “coisas” e a Internet, e não nas “coisas” em si. Desta forma, atenção especial deve ser dada no desenvolvimento das interfaces de rede no projeto.

Este capítulo abordou que é a IoT, sua definição e breve histórico, suas perspectivas econômicas e aplicações atuais. O próximo capítulo tratará da metodologia do projeto.

### 3 MATERIAIS E MÉTODOS

Esta seção apresenta os componentes, dispositivos e ferramentas de desenvolvimento utilizados no projeto, bem como suas respectivas funções.

#### 3.1 Plataformas

Para o desenvolvimento do projeto foram usados os seguintes componentes e dispositivos eletrônicos:

Figura 3 – Placa de desenvolvimento Intel Edison



Fonte: O Autor (2017).

A placa de desenvolvimento Intel Edison, ilustrada na figura 3, foi utilizada no projeto devido à disponibilidade desta. Outras possibilidades eram a Raspberry Pi 3 [21], Arduino [22], BeagleBone Black [23] e ESP8266 [24].

A tabela 1 abaixo ilustra uma comparação entre as opções de plataformas, com foco nos aspectos mais importantes ao projeto:

Tabela 1 – Comparaçāo entre as plataformas usuais de projetos IoT.

Plataformas >	Intel Edison	Arduino	Raspberry Pi 3	BeagleBone Black	ESP8266
Parâmetros √					
Custo	~ \$50	~ \$20	~ \$35	~ \$45	~ \$2
Poder computacional	400 MHz Dual Core	16 MHz	700 MHz	1 GHz	80 MHz
Suporte da comunidade	Intermediário	Alto	Alto	Baixo	Alto

Recursos de <i>hardware</i>	Alto	Baixo	Intermediário	Alto	Extremamente baixo
Recursos de <i>software</i>	Alto	Intermediário	Alto	Intermediário	Extremamente baixo

Fontes: [25]

Recursos de *hardware* representam os diferentes componentes e chips disponíveis na placa, como wifi, bluetooth, infravermelho, codecs e etc. Recursos de *software* são as ferramentas de software que compõe o *middleware* padrão da placa, como protocolos de comunicação, ferramentas de criptografia, linguagens de programação e etc. As placas ou plataformas mencionadas são focadas em projetos IoT.

Intel Edison foi descontinuada por sua mantenedora, a Intel, porém o projeto estava em sua metade quando este fato foi divulgado e decidiu-se continuar com a utilização desta plataforma.

Figura 4 – Kit de Sensores e Atuadores Grove Starter Kit Plus



Fonte – O Autor (2017).

O Kit de sensores da Grove, ilustrado na figura 4, compatível com a Intel Edison e Arduino, também foi utilizado projeto. Em particular o botão, um sensor, e a buzina, um atuador, foram usados.

Além destes, o computador pessoal do Autor foi utilizado no projeto.

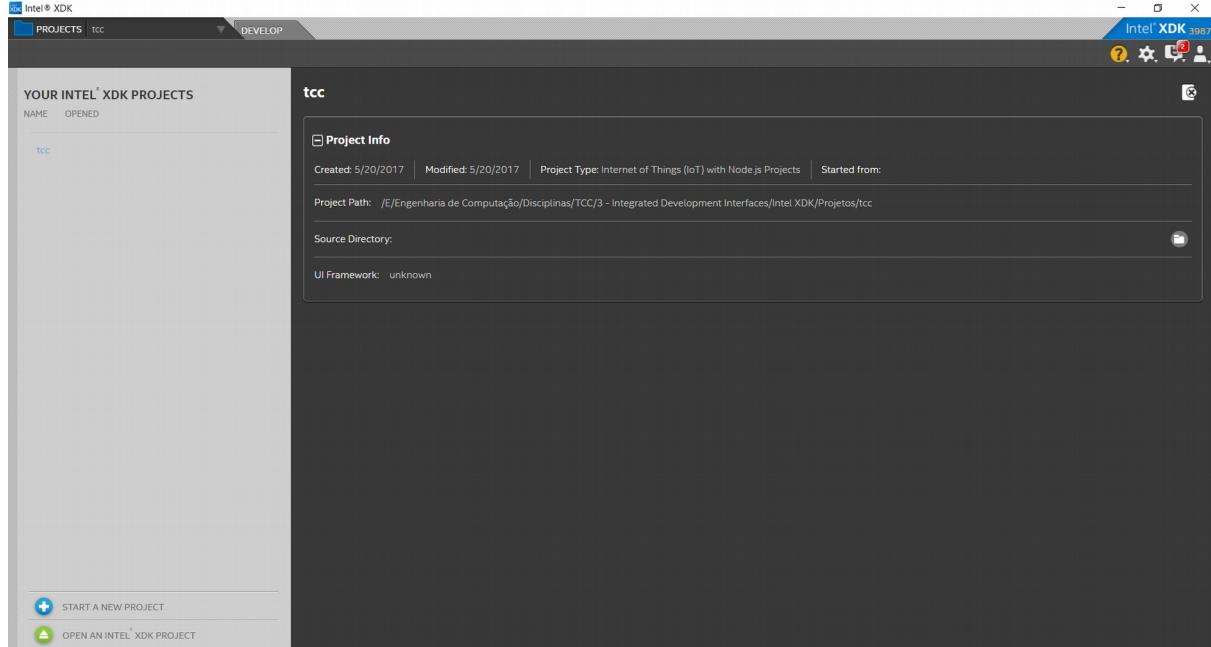
### 3.2 Ferramentas de Desenvolvimento

O projeto foi desenvolvido no Sistema Operacional – SO, *Windows 10 Pro Edition* [26].

A plataforma de desenvolvimento escolhida foi a *Intel Cross Platform Development Kit* – Intel XDK [27], ilustrada nas figuras 5 e 6. Esta plataforma foi escolhida por ter sido

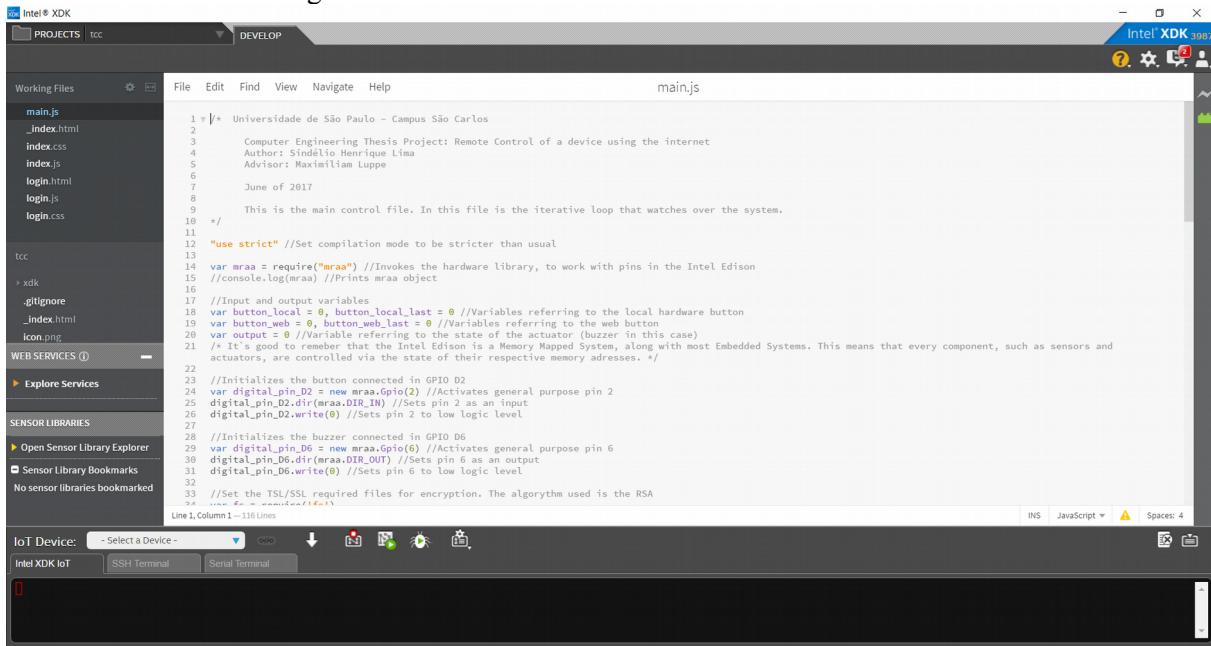
criada especificamente para o desenvolvimento de programas para dispositivos IoT para as placas da Intel, como a Intel Edison.

Figura 5 – Ambiente de projetos da Intel XDK.



Fonte – O Autor (2017).

Figura 6 – Ambiente de desenvolvimento da Intel XDK.



Fonte – O Autor (2017).

A linguagem de programação padrão na plataforma Intel XDK é a Node.js [28], e portanto foi escolhida para uso no projeto. A linguagem Node.js foi criada para otimizar

operações de rede e sistemas de arquivos, que são lentas em geral. A node.js é uma variante do javascript focada em *backend*, ou seja, em servidores. Ressalta-se que a Node.js é uma linguagem assíncrona, *i.e.* uma linguagem com uma única linha de execução porém que não espera chamadas terminarem para prosseguir a execução das instruções seguintes.

Para realizar comunicação serial com a Intel Edison, usou-se o programa Putty [29], mostrado na figura 4.

Para criptografar a comunicação entre o servidor e os clientes usou-se o *Open Secure Sockets Layers – OpenSSL* [30], um kit aberto de ferramentas para segurança.

Todo o *software* usado no projeto é livre, aberto ou licenciado para uso pelo proprietário.

Este capítulo tratou dos recursos de *hardware* e *software* utilizados no projeto. O próximo capítulo discorrerá sobre a implementação do projeto.

## 4 IMPLEMENTAÇÃO

### 4.1 Instalação dos *drivers* para comunicação com a placa Intel Edison

O primeiro passo no projeto foi instalar os programas controladores, ou *drivers*, da comunicação entre o SO Windows 10, no cliente, e a placa Intel Edison, o servidor. É necessária a instalação de dois *drivers* de comunicação, um para *Universal Serial Bus – USB* [31] e outro para *Future Technology Devices International – FTDI* [32]. Os *drivers* foram obtidos em: <https://software.intel.com/en-us/installing-drivers-for-intel-edison-board-with-windows>

### 4.2 Atualização do *firmware* da placa Intel Edison

O segundo passo dado no projeto fora a atualização do *firmware* da placa Intel Edison. O *firmware* consiste num programa permanente escrito em uma memória somente de leitura, *Read Only Memory – ROM*. A Intel Edison possui uma distribuição de Linux denominada “Yocto Project” [33], e duas bibliotecas, “dfu-util.exe” e “libusb-1.0.dll” que compõe seu *firmware*. Portanto, para atualizar o *firmware* da placa é preciso baixar as versões mais atuais destes componentes. Em seguida deve-se extrair estes componentes e colocá-los em uma mesma pasta.

Na sequência deve-se conectar a placa a um computador via conexão USB. Esta conexão fornecerá energia à placa e habilitará a transferência de dados entre o computador e esta. Na placa existe um pequeno seletor que possui duas posições, uma para o modo dispositivo e outra para o modo anfitrião. Para atualizar o *firmware* da placa o modo dispositivo deve estar selecionado, que corresponde à posição mais próxima às duas conexões microUSB da placa.

Após conectar a placa ao computador via conexão USB, deve-se esperar cerca de um minuto até que a placa esteja estável, com SO completamente carregado. Então deve-se abrir um terminal de comandos no SO, e selecionar a pasta onde estão os arquivos componentes do *firmware*, que são a imagem do Yocto Linux e as bibliotecas, e executar o comando “**flashall.bat**” num terminal do SO. Para abrir um terminal de comandos no Windows digita-se “**cmd**” na barra de buscas.

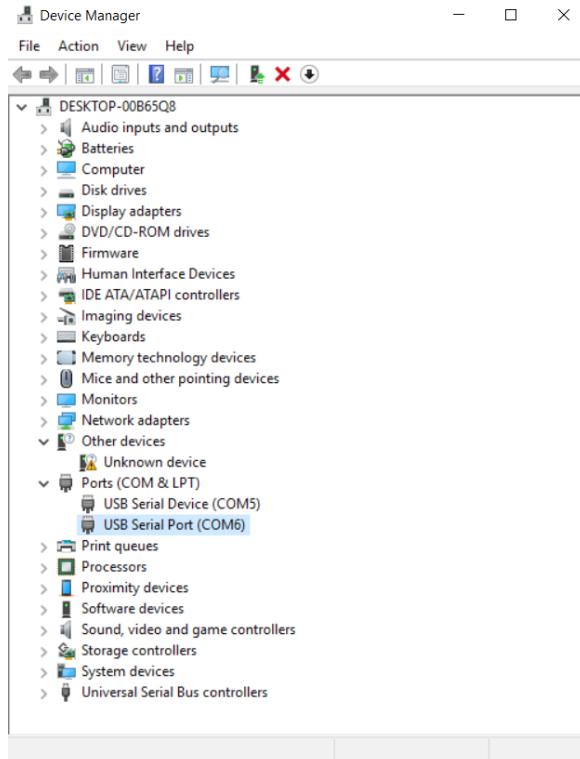
O comando “**flashall.bat**” executa o arquivo de mesmo nome, que por sua vez consiste num conjunto de comandos que atualiza o *firmware* da placa Intel Edison [34].

#### 4.3 Estabelecendo conexão serial entre o computador e a placa Intel Edison

Para dar comandos à placa Intel Edison, usou-se o programa Putty. Este é um programa aberto e criado para realizar conexões *Secure Shell* – SSH [35], e Telnet [36] com o SO Windows.

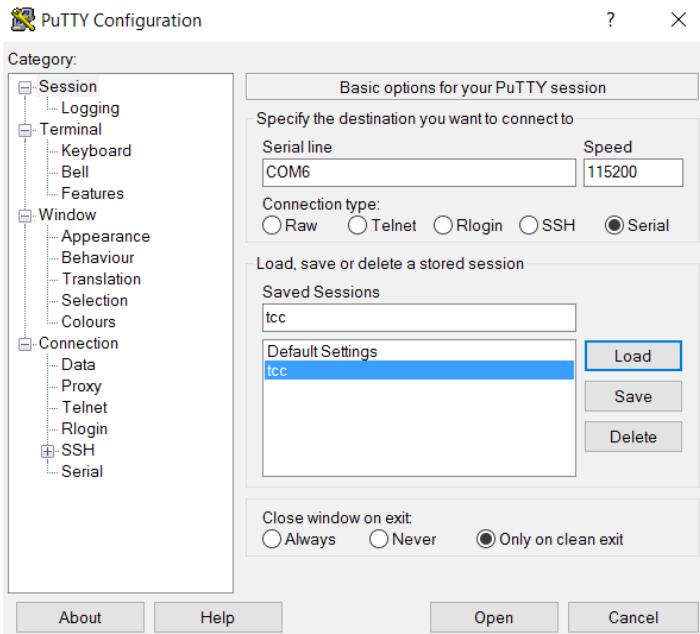
Na tela do Putty, é necessário realizar algumas configurações. Primeiro deve-se selecionar a opção SSH. Em seguida deve-se colocar a taxa de transmissão em 115200. Por fim deve-se escolher a porta de comunicação USB que conecta o computador à Intel Edison. A porta correta é mostrada no gerenciador de dispositivos, como mostra a figura 7. A configuração completa do Putty é mostrada na figura 8.

Figura 7 – Verificação da porta USB correta para comunicação serial com a Edison.



Fonte – O Autor (2017).

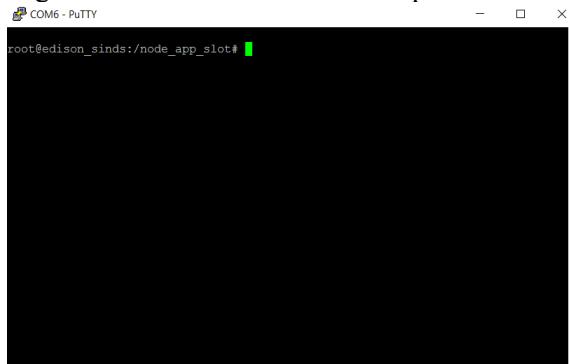
Figura 8 – Configuração do Putty para comunicação serial com a Edison.



Fonte – O Autor (2017).

Após realizada a configuração, resta clicar em “Open” para iniciar a conexão. Uma tela de terminal abre e pode-se dar comandos ao SO Yocto executado na Intel Edison, como mostrado na figura 9. Na primeira conexão serial com a Edison é possível alterar as credenciais de acesso serial à placa.

Figura 9 – Terminal de comandos para a Edison.



Fonte – O Autor (2017).

A tabela 1 abaixo mostra os comandos mais recorrentes para a Edison.

Tabela 2 – Comandos mais comuns para a Edison.

Comando	Função
<code>cat /etc/version</code>	Mostra a versão do SO contido na Edison
<code>ifconfig</code>	Mostra informações sobre o <i>Internet Protocol address</i> – IP, da Edison
<code>configure_edison --name</code>	Altera o nome da Edison
<code>configure_edison --password</code>	Altera a senha da Edison
<code>configure_edison --wifi</code>	Conecta a Edison a uma rede sem fio
<code>configure_edison --help</code>	Mostra uma mensagem de ajuda e mais comandos possíveis para a Edison

Fonte – O Autor (2017).

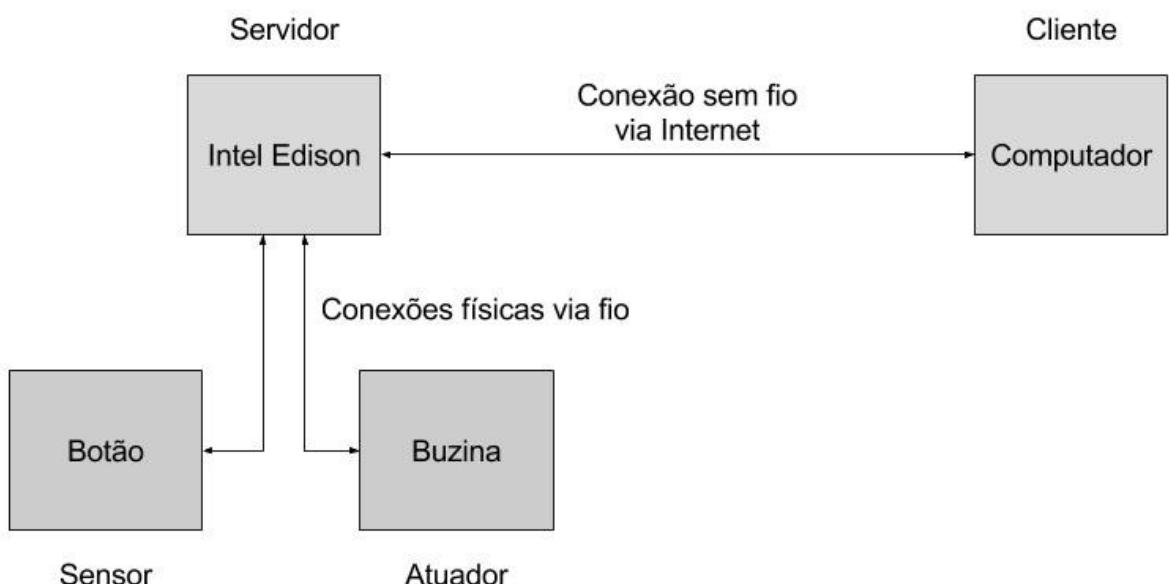
#### 4.4 Estabelecendo conexão entre a Internet e a placa Intel Edison

Para conectar a Edison à Internet, usa-se o comando “`configure_edison --wifi`”. Após iniciado, o comando irá procurar por redes sem fio num raio de até 20 metros e após dez segundos mostrará o resultado da busca. Então pode-se selecionar uma rede para conexão. Dependendo do tipo de protocolo de segurança usado na rede, o procedimento será diferente. O caso mais comum é o de redes com protocolo *Wifi Protected Access 2* – WPA2 [37], nas quais deve-se informar usuário e senha para realizar o *login* na rede.

#### 4.5 Arquitetura do sistema objetivo

O objetivo do projeto é controlar um atuador via Internet. Para realizar o objetivo, o sistema ilustrado na figura 10 foi idealizado.

Figura 10 – Arquitetura do sistema.



Fonte – O Autor (2017).

Por meio desta arquitetura o computador exerce controle sobre o atuador via Internet, cumprindo o objetivo do projeto. Este controle é realizado em dois passos, sendo eles:

1. O computador se comunica via Internet com o servidor, que está sendo executado na Edison. Dessa forma o computador é um cliente do servidor.

2. A Edison por sua vez se comunica localmente com o atuador, controlando-o de acordo com os comandos dados pelo computador.

O atuador selecionado para testes foi uma buzina. Também por objetivos de teste, colocou-se um botão, um tipo de sensor, para ativar localmente a buzina. O computador exerce controle sobre a buzina através de uma página na Internet.

Em termos de segurança, a comunicação entre o computador e o servidor deve ser confidencial a estes. Para tanto será aplicada criptografia na comunicação. Também, o controle do atuador é restrito a usuários autorizados. Para garantir que o controle do atuador seja realizado somente por usuários autorizados, uma página de *login* deve ser criada. A página de *login* deve ser robusta à ataques de força bruta [38].

O funcionamento do sistema é expresso pelo pseudocódigo a seguir:

Código 1

```

Se inicializar a placa
    Inicializar o sistema

Laço principal
    Se usuário tentar conectar-se ao sistema
        Se houver conexão com o servidor
            Exibir página de login
            Se o usuário for autorizado
                Exibir página de controle da buzina
                Se usuário der comando na página de controle
                    Atualizar o estado da buzina de acordo com o
                    comando dado
                    Se usuário não for autorizado
                        Exibir mensagem de falha no login
                Senão houver conexão com o servidor
                    Exibir mensagem de falha na conexão com o servidor

```

Ressalta-se que o usuário do sistema pode sair das páginas de *login* e controle a qualquer momento, pois são páginas normais da rede.

#### 4.6 Desenvolvimento dos programas necessários ao sistema

Os seguintes programas foram escritos para o sistema:

##### 1. main.js

Programa do servidor executado na Edison. O servidor está escrito na linguagem Node.js. O servidor é do tipo *Hyper Text Transfer Protocol Secure* – HTTPS [39], que possui criptografia assimétrica com o algoritmo *Rivest, Shamir and Adelman* – RSA [40]. Para criar as chaves pública e privada, bem como o certificado digital do servidor HTTPS, foi usado o kit aberto de ferramentas OpenSSL, que consiste em uma coleção aberta de ferramentas para segurança digital. Servidores HTTPS são robustos contra ataques de *snifing*, *eavesdropping*, *man in the middle*, *masquerade* e *spoofing* [41]. Para realizar a comunicação entre o servidor e entidades externas, foi usada a *Application Programmer Interface* – API, “*websockets*”, do node.js. Para controlar o sensor e o atuador foi usada a biblioteca MRAA [42], que é uma interface de comunicação de baixo nível com dispositivos eletrônicos em ambientes Linux.

##### 2. login.html

Programa que contém o conteúdo da página de *login*. Este programa está escrito em *Hyper Text Markup Language* – HTML [43], e é um dos três componentes da página de *login*. A página de *login* é vital para que somente usuários autorizados possam acessar a página principal do sistema.

##### 3. login.css

Programa que contém o estilo da página de *login*. Este programa está escrito em *Cascade Style Sheets* – CSS [44], e é o segundo dos três componentes da página de *login*. Este programa remodela a página de *login* com objetivo de tornar mais claras as informações exibidas.

##### 4. login.js

Programa que contém o funcionamento da página de *login*. Esta programa está escrito em javascript [45], e é o terceiro dos três componentes desta página. Este programa dá dinamismo para a página de *login*, fazendo com que esta responda às ações dos usuários.

##### 5. \_index.html

Programa que contém o conteúdo da página de controle do sistema. Este programa está escrito em HTML e é o primeiro dos três componentes da página de controle. Para acessar esta página é necessário realizar login previamente no sistema.

## 6. index.css

Programa que contém o estilo da página de controle. Escrito em CSS, é o segundo dos três componentes da página de controle. Este programa remodela a aparência da página para que as informações sejam mais claras aos usuários.

## 7. index.js

Programa que contém o funcionamento da página de controle. Escrito em javascript, é o terceiro dos três componentes da página de controle. Este programa dá dinamismo à página de controle, fazendo com que esta reaja às ações dos usuários.

## 8. package.json

Programa gerado automaticamente pela Intel XDK que lista as dependências do projeto aos módulos ou pacotes do Node.js. Apesar de não ter sido escrito pelo Autor, é um programa vital para a replicação do projeto no futuro.

## 9. my\_startup.sh

Para conferir ao sistema robustez à interrupções de energia, criou-se um *script* em *bash* [46] para conectar automaticamente à rede sem fio do Instituto de Ciências Matemáticas e Computação – ICMC, e ativar automaticamente o sistema sempre que a Edison fosse iniciada. Este *script*, denominado “my\_startup.sh”, é executado durante a inicialização da Edison, após o SO Yocto ser carregado. O *script* pode ser usado com outras redes sem fio do tipo WPA2, bastando-se trocar o identificador da rede e as credenciais de acesso no *script*.

Os programas de 1 à 7 foram escritos na IDE Intel XDK. O programa 9 foi escrito no Nano [47], editor de texto para terminais de comandos em Linux. Todos os programas criados estão devidamente comentados e anexos neste relatório.

Este capítulo tratou do preparo das plataformas e ambientes de desenvolvimento para uso no projeto. Na sequência, discorreu-se sobre o sistema proposto e suas especificações técnicas, de baixo e alto nível. O capítulo também abordou os programas escritos para o sistema proposto. O próximo capítulo elaborará sobre os testes realizados no projeto e seus resultados.

## 5 TESTES E RESULTADOS

Esta seção apresenta os resultados obtidos com os testes realizados no sistema.

### 5.1 Testes de funcionalidade

A finalidade dos testes de funcionalidade é comparar o comportamento do sistema com o comportamento esperado deste.

#### 1. Teste da resposta do servidor.

Com a Edison devidamente conectada ao computador e à Internet sem fio, tentou-se explorar os recursos disponibilizados pelo servidor sendo executado nesta. Para tanto usou-se um navegador para acessar o endereço do servidor na rede, sendo este:

`https://<IP>:3000`

Sendo o <IP> variável de rede para rede. O navegador mostrou adequadamente a página de login, disponibilizada pelo servidor. Na sequência, tentou-se entrar com credenciais distintas das autorizadas no servidor e nenhuma destas obteve sucesso. Inserindo as credenciais autorizadas, testou-se então a página de controle do sistema. O teste mais importante fora o teste do interruptor presente na página de controle, que inverte o estado da buzina. O interruptor funcionou adequadamente. O agendador de funcionamento também funcionou como esperado, invertendo o estado da buzina no horário agendado.

#### 2. Teste de consistência de dados.

Tentou-se entrar com caracteres estranhos em todos os campos do sistema com entrada para dados. Nenhum dos campos aceitou caracteres inválidos, como esperado. Todos os campos foram testados, sendo eles: login, senha e agendador.

#### 3. Teste de perda de conectividade.

Verificou-se o comportamento do sistema caso a conexão via Internet fosse perdida. O navegador alertou o usuário da perda de conexão, como programado. No servidor também houve um alerta quanto à queda de conexão.

#### 4. Teste de robustez às quedas de energia.

O sistema foi programado para se conectar automaticamente à Internet sem fio predefinida e ativar o servidor também automaticamente. Verificou-se este comportamento em simulações de perda de energia da Edison.

Os testes de funcionalidade foram os primeiros a serem realizados no sistema.

### 5.2 Testes de segurança

Os testes de segurança visam encontrar falhas e vulnerabilidades no sistema para que estas possam ser corrigidas. Os testes escolhidos tiveram como base o relatório da OWASP [14], com as dez maiores vulnerabilidades em sistemas IoT, no intuito de tornar o sistema o mais robusto possível aos ataques mais comuns. Os seguintes testes foram realizados:

#### 1. Injeção de comandos no banco de dados

Tentou-se injetar comandos de banco de dados em todos os campos de inserção no sistema. Nada ocorreu, como esperado, pois o sistema não possui banco de dados.

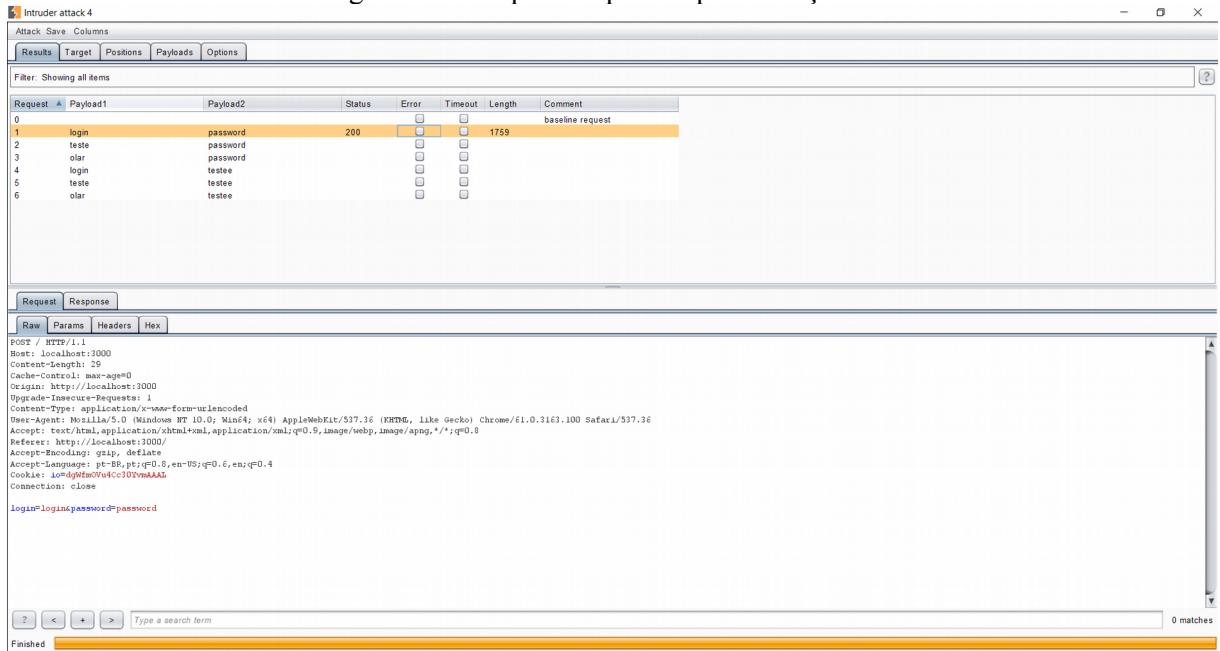
#### 2. Execução de *scripts* externos

Tentou-se também executar comandos de linguagens de *scripts*, como javascript, nos campos de inserção. Nada ocorreu, como esperado.

#### 3. Ataque de força bruta para obter as credenciais da página de login

Usando a ferramenta BurpSuite [48], realizou-se um ataque de força bruta na página de *login*. Observou-se que o BurpSuite envia pacotes do tipo *post* diretos ao formulário de autenticação, não acessando o sistema via *get* no servidor. Isso significa que toda segurança anterior ao processamento do formulário é inútil diante de um ataque de força bruta. Assim é necessário corrigir o sistema para que um número limitado de requisições do tipo *post* sejam processadas pelo servidor, inviabilizando ataques de força bruta para obter as credenciais de autenticação. A figura 11 a seguir mostra a interface do BurpSuite após a simulação do ataque de força bruta:

Figura 10 – BurpSuite após ataque de força bruta



Fonte – O Autor (2017).

Observa-se uma resposta diferente para as credenciais corretas.

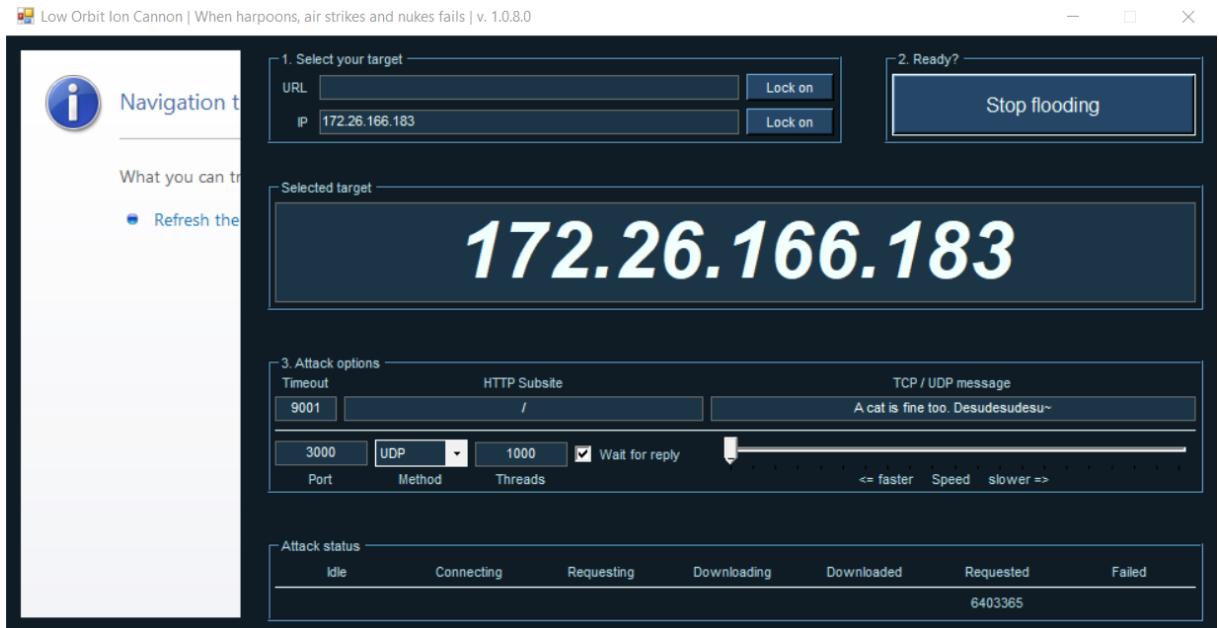
#### 4. Ataques de interrupção de serviço via *pings*

Pacotes do tipo *ping* foram enviados ao sistema, no entanto o servidor não responde a pacotes deste tipo, e logo ataques de *ping*, como o *ping of death*, são inofensivos ao sistema.

#### 5. Ataque de interrupção de serviço via enchente de requisições

Usou-se a ferramenta *Low Orbit Ion Cannon* – LOIC [49], para atacar o sistema com enchetes de requisições. Três tipos de protocolos foram usados nas requisições: HTTPS, *Transmission Control Protocol* – TCP e *User Datagram Protocol* – UDP. Ataques com os protocolos HTTPS e TCP não tiveram efeito no sistema, porém o protocolo UDP interrompeu a disponibilidade de serviço deste. O protocolo UDP é mais promíscuo que os demais, o que faz com que o servidor aceite e processe as requisições UDP enviadas no ataque. Porém, mais pesquisa é necessária para compreender exatamente porque uma enchente de requisições UDP é capaz de interromper o serviço do sistema. A figura 12 abaixo mostra a interface da LOIC configurada para realizar uma enchente UDP:

Figura 12 – LOIC configurado para realizar enchente de pacotes UDP.



Fonte – O Autor (2017).

Uma medida estudada para evitar ataques deste tipo foi duplicar o servidor em outra porta aleatória, que somente é revelada quando ocorre uma enchente no sistema. Usuários autênticos serão capazes de identificar a réplica e migrar para o uso do novo servidor, enquanto *bots* ou requisições feitas por ferramentas de DoS não terão facilidade para se conectar ao servidor replicado.

## 6. Acesso a recursos do sistema sem autenticação

Tentou-se acessar a página de controle do sistema, através de um *get* em sua *Universal Resource Locator* – URL. O resultado inicial era a possibilidade de se entrar na página de controle do sistema sem estar devidamente autenticado no sistema. Esta vulnerabilidade foi corrigida através da criação de pastas com recursos públicos e privados no servidor, sendo os recursos privados disponibilizados do servidor aos clientes somente após a autenticação no sistema. Ressalta-se que o servidor disponibiliza os recursos de forma estática, ou seja, os clientes não têm poder de modificação sobre o que lhes é disponibilizado.

A URL que permitia o ataque era a seguinte:

[https://<IP>:3000/\\_index.html](https://<IP>:3000/_index.html)

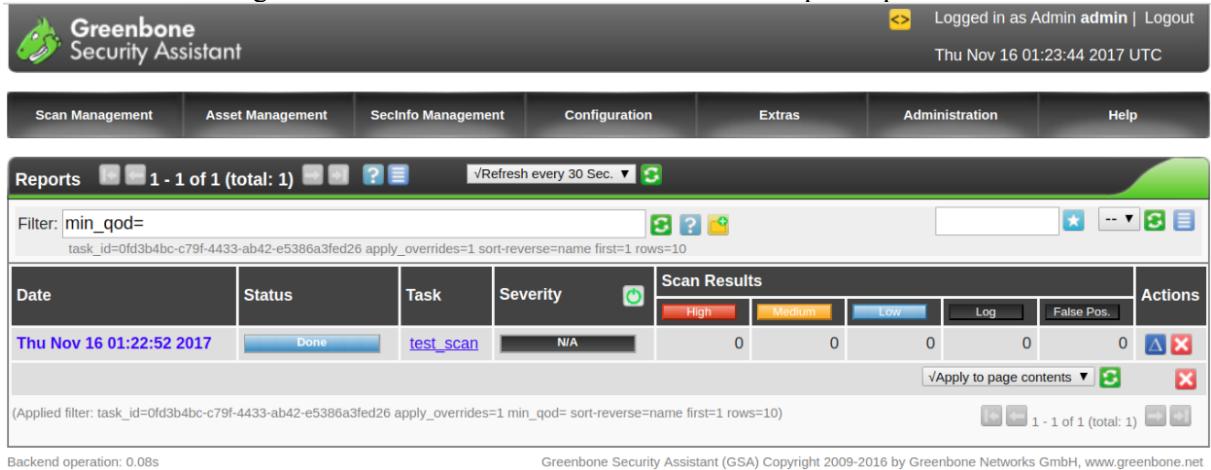
No qual <IP> corresponde ao IP do servidor na rede.

## 7. Scaneamento de vulnerabilidades do sistema

Por fim, diversas ferramentas de scaneamento foram acionadas a fim de se encontrar vulnerabilidades no sistema. As ferramentas utilizadas e os respectivos resultados foram:

Ferramenta: OpenVAS – *Open Vulnerability Assessment Scanner* [50]. Ferramenta de código aberto para scaneamento e gerenciamento de vulnerabilidades. Dentre as ferramentas de código aberto, é considerada pela comunidade a mais completa. O scaneamento com esta ferramenta não revelou vulnerabilidades, como mostra o relatório na figura 13, a seguir:

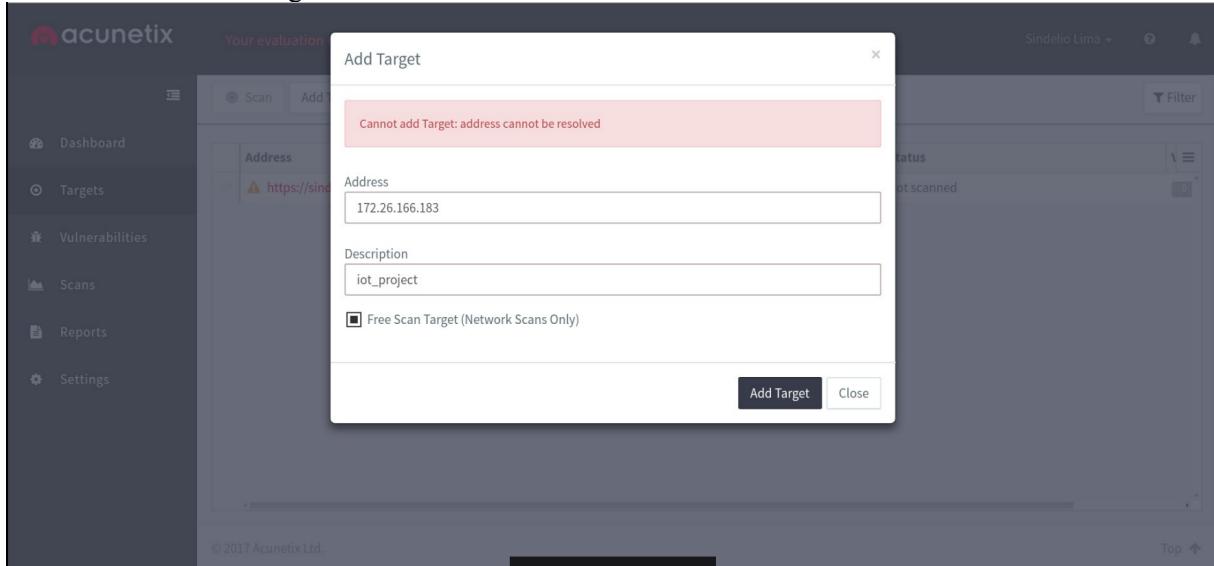
Figura 13 – Relatório de vulnerabilidades criado pelo OpenVAS.



Fonte – O Autor (2017).

Ferramenta: Acunetix online [51]. Scaneador de vulnerabilidades de rede. É uma ferramenta privada, porém com alguns recursos disponíveis para teste. No entanto não foi possível realizar o scaneamento, como mostrado na figura 14, pois o sistema não possui IP fixo. Esta ferramenta não dá suporte para scanear o *localhost*.

Figura 14 – Tentativa de uso do scaneador Acunetix online.



Fonte – O Autor (2017).

Ferramenta: nmap [52] online. O nmap é uma ferramenta para varredura de portas. Executaram-se scans para portas TCP e UDP. O teste com esta ferramenta não revelou vulnerabilidades, como mostrado na figura 15.

Figura 15 – Relatório de escaneamento de portas UDP gerado pelo nmap online.

Fonte – O Autor (2017).

Ressalta-se que o sistema possui uma vulnerabilidade à enchente de pacotes UDP, e que esta vulnerabilidade não foi mostrada pelo nmap. Assim, é importante o uso de diversas ferramentas e ataques contra o sistema para conclusão precisa do nível de segurança atingido.

Ferramenta: nikto [53] online. Scaneador de vulnerabilidades em servidores de rede. Assim como ocorreu com o Acunetix online, também não foi possível scanear o sistema com a ferramenta nikto devido à necessidade de IP fixo por estas ferramentas. A figura 16 mostra que a URL do sistema não é acessível pelo nikto.



Fonte – O Autor (2017).

Além destas ferramentas, o BurpSuite apresenta um scaneador de vulnerabilidades bem aceito pela comunidade, porém privado. Tentou-se entrar em contato com a PortSwigger, empresa desenvolvedora do BurpSuite, para teste completo da ferramenta, porém não houve resposta.

O sistema não mostrou vulnerabilidades de acordo com as ferramentas usadas, o que se justifica por duas razões. A primeira é que o sistema criado é simples e pequeno, não possuindo vasta gama de recursos como portas e recursos distribuídos, o que por fim diminui a probabilidade de existência de vulnerabilidades. A segunda é que todos os recursos usados no desenvolvimento do sistema são de código aberto e amplamente usados pela comunidade, como o node.js. Tais recursos são testados incansavelmente pela comunidade, e por isso apresentam poucas vulnerabilidades detectáveis por ferramentas de penetração.

Este capítulo mostrou os testes realizados no sistema projetado, de funcionalidade e segurança, e seus respectivos resultados. O próximo capítulo encerrará o texto primário com as conclusões acerca do projeto.

## 6 CONCLUSÕES

A respeito da IoT, esta se desenvolverá em um primeiro momento com foco em telemetria. Caso um sistema de sensoriamento à distância seja dominado por um atacante, este terá acesso às informações do sistema. Como exemplo, o atacante pode: vender as informações, quebrando a privacidade e confidencialidade destas; pedir resgate pelas informações do sistema; derrubar o sistema; divulgar as informações para o público e etc. No entanto, o ponto principal é que apesar de todos os ataques possíveis, não é possível causar danos físicos ao sistema ou aos usuários deste, como acontece no controle de atuadores pela Internet. Por esta razão a IoT se desenvolverá em telemetria inicialmente. Conforme a segurança das aplicações de telemetria for desenvolvida e testada, o controle de atuadores será viabilizado.

A maior parte das fragilidades de sistemas IoT está nas interfaces de interação com usuários, particularmente nas interfaces *web*. Atenção deve ser dada as páginas *web* nos projetos IoT, como a página de autenticação neste projeto.

Qualquer objeto físico pode se tornar uma “coisa”, desde que possua “sentidos”, conectividade com a Internet e uma interface com os usuários. No futuro a tendência é que muitas “coisas” distintas, e até mesmo inusitadas, surjam.

A respeito do sistema criado, as funcionalidades deste, como o interruptor na página da Internet para troca do estado da buzina e o agendador de funcionamento desta, foram desenvolvidas num primeiro momento. Porém, atribuir segurança ao sistema foi chave para sua utilização. Elementos como a página de autenticação, a criptografia na comunicação entre o servidor e os clientes, o mecanismo para evitar ataques de força bruta, a replicação do servidor em outra porta, e etc, dão robustez ao sistema. Excluindo estes elementos, o sistema não seria utilizável.

Também foram primordiais os testes realizados no sistema, com ataques reais para compreender as vulnerabilidades deste. Os testes revelaram algumas vulnerabilidades à ataques de força bruta e ataques de interrupção de serviço que não teriam sido detectados de outra forma.

Quanto as dificuldades enfrentadas no projeto, as maiores foram as seguintes:

- Atualizar o firmware da Intel Edison não foi trivial, pois as ferramentas disponibilizadas pela Intel não funcionaram e houve a necessidade de realizar a atualização manual, como mostrado no capítulo 3: Implementação.

- A Intel Edison foi descontinuada quando o projeto estava funcional. No entanto optamos por continuar o projeto pois este aproximara-se dos objetivos almejados. Ressalta-se que o projeto pode ser portado para outras plataformas , como para a Raspberry Pi 3 ou BeagleBone Black, pois as bibliotecas e módulos utilizados, dentre as mais importante a MRAA e o Node.js, são multiplataforma, ou seja, dão suporte para diversas plataformas de desenvolvimento.
- A quantidade de conteúdo absorvida foi considerável. A necessidade de aprender as linguagens Node.js, HTML, Javascript, CSS, bem como a utilizar a OpenSSL para criptografia, compreender sistemas *web* em geral, conceitos de *frontend* e *backend* e o restante de recursos utilizados no sistema faz com que o projeto possua uma curva de aprendizado longa, antes da aplicação dos conhecimentos e ferramentas estudados.
- Não obteve-se acesso à ferramenta de testes de penetração profissional BurpSuite. O projeto se beneficiaria do uso pleno desta ferramenta, além da versão gratuita.

Conclui-se que o projeto atingiu o objetivo inicial de controlar um atuador via Internet, de forma razoavelmente segura, contudo existe margem para desenvolvimento futuro.

Os próximos passos são:

- Subir o servidor para uma nuvem computacional, com IP fixo. Desta forma, o servidor poderá ser acessado de qualquer local.
- Portar o sistema para outra plataforma, como a Raspberry Pi 3. A Intel Edison foi descontinuada, e é interessante continuar o projeto em uma plataforma com suporte de longo prazo e comunidade ativa.
- Implementar o controle de atuadores mais complexos, com mais estados possíveis.
- Implementar o controle de múltiplos atuadores simultaneamente.
- Correção da vulnerabilidade à encheres UDP nas APIs utilizadas.

Por fim, a IoT é uma área promissora da tecnologia, porém ainda em seus estágios iniciais. O desenvolvimento e uso da IoT depende primordialmente de segurança, que aos poucos vem ganhando mais atenção nas aplicações.



## REFERÊNCIAS

1. ASHTON, K. That “Internet of things” thing. **RFID Journal**, v.22, p.97-114, Jun. 2009.
2. GREENGARD, S. **The internet of things** Cambridge: The MIT Press, c2015.
3. BARRET, J. The Internet of things. **TEDxCIT** 5 OUTUBRO 2012. Disponível em: <<https://www.youtube.com/watch?v=QaTIt1C5R-M>>. Acesso em: 25 ago. 2017.
4. **IPv6**. Disponível em: <<http://ipv6.br/>>. Acesso em: 30 ago. 2017.
5. LEIBSON, S. **IPv6: how many IP addresses can dance on the head of a pin?** MARÇO 2008. Disponível em: <<http://www.edn.com/electronics-blogs/other/4306822/IPv6-How-Many-IP-Addresses-Can-Dance-on-the-Head-of-a-Pin->>>. Acesso em: 30 ago. 2017.
6. ROUSE, M.; WIGMORE, I. **Internet of things (IoT)** JULHO 2016. Disponível em: <<http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>>. Acesso em: 30 ago. 2017.
7. ANALYTICS VIDHYA **10 Real World Applications of Internet of Things** 26 AGOSTO 2016. Disponível em: <<https://www.analyticsvidhya.com/blog/2016/08/10-youtube-videos-explaining-the-real-world-applications-of-internet-of-things-iot/>>. Acesso em: 28 out. 2017.
8. GARTNER. **Top 10 IoT technologies for 2017 and 2018** 22 JANEIRO 2016. Disponível em: <<https://www.gartner.com/login/loginInitAction.do?method=initialize&TARGET=http%253A%252F%252Fwww.gartner.com%252Fdocument%252F3188520%253Fref%253DsolrAll%2526refval%253D162721561%2526qid%253D679d9fd23e31525ae29692fa25d946ac>>. Acesso em: 12 jul. 2017.
9. BUSINESS INSIDER INTELLIGENCE. **Internet of everything** 8 ABRIL 2015. Disponível em: <<http://www.businessinsider.com/internet-of-everything-2015-bi-2014-12>>. Acesso em: 12 jul. 2017.
10. PUBNUB. **Without security, is the Internet of Things just a toy?** 30 JANEIRO 2015. Disponível em: <<https://www.pubnub.com/blog/2015-01-30-without-security-internet-things-just-toy/>>. Acesso em: 28 out. 2017.
11. DYN. **Statement on 10/21/2016 DDoS attack** 22 OUTUBRO 2016. Disponível em: <<https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/>>. Acesso em: 12 jul. 2017.
12. MQTT.org. **Protocol specifications** Disponível em: <<http://mqtt.org/documentation>>. Acesso em: 17 jul. 2017.
13. ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS). **MQTT version 3.1.1 becomes an OASIS standard** 30 OUTUBRO 2014. Disponível em: <<https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard>>. Acesso em: 15 set. 2017.
14. OPEN WEB APPLICATION SECURITY PROJECT (OWASP). **Internet of things top ten** 2014. Disponível em:

<[https://www.owasp.org/images/7/71/Internet\\_of\\_Things\\_Top\\_Ten\\_2014-OWASP.pdf](https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf)>. Acesso em: 17 jul. 2017.

15. OWASP. **SQL injection** 10 ABRIL 2016. Disponível em:

<[https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)>. Acesso em: 28 out. 2017.

16. OWASP. **Cross Site Scripting (XSS)** 6 ABRIL 2016. Disponível em:

<[https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))>. Acesso em: 28 out. 2017.

17. OWASP. **Cross Site Request Forgery (CSRF)** 20 JUNHO 2016. Disponível em:

<[https://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery\\_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))>. Acesso em: 28 out. 2017.

18. ARMIS. **The attack vector “BlueBorne” exposes almost every connected device** AGOSTO 2017. Disponível em: <<https://www.armis.com/blueborne>>. Acesso em: 30 out. 2017.

19. EMSISOFT. **What is a computer worm and how does it spread?** 21 JULHO 2017. Disponível em: <<https://blog.emsisoft.com/2017/07/21/computer-worms>>. Acesso em: 30 out. 2017.

20. **Bluetooth.** Disponível em: <<https://www.bluetooth.com/what-is-bluetooth-technology>>. Acesso em: 28 out. 2017.

21. **Raspberry Pi 3.** Disponível em: <<https://www.raspberrypi.org>>. Acesso em: 16 set. 2017.

22. **Arduino.** Disponível em: <<https://www.arduino.cc>>. Acesso em: 16 set. 2017.

23. **BeagleBone Black.** Disponível em: <<https://beagleboard.org/black>>. Acesso em: 16 set. 2017.

24. **ESP8266.** Disponível em:

<<http://espressif.com/en/products/hardware/esp8266ex/overview>>. Acesso em: 16 set. 2017.

25. IBM. **Choosing the best hardware for your next IoT project** 5 MAIO 2017.

Disponível em: <<https://www.ibm.com/developerworks/library/iot-lp101-best-hardware-devices-iot-project/index.html>>. Acesso em: 30 out. 2017.

26. **Windows 10 Pro.** Disponível em: <<https://www.microsoft.com/en-us/windows>>. Acesso em: 16 set. 2017.

27. **Intel XDK.** Disponível em: <<https://software.intel.com/en-us/forums/intel-xdk>>. Acesso em: 16 set. 2017.

28. **Node.js.** Disponível em: <<https://nodejs.org/en>>. Acesso em: 16 set. 2017.

29. **Putty.** Disponível em: <<http://www.putty.org>>. Acesso em: 16 set. 2017.

30. **OpenSSL.** Disponível em: <<https://www.openssl.org>>. Acesso em: 16 set. 2017.

31. **USB**. Disponível em: <<http://www.usb.org/home>>. Acesso em: 16 set. 2017.
32. **FTDI**. Disponível em: <<http://www.ftdichip.com/>>. Acesso em: 16 set. 2017.
33. **Yocto Project**. Disponível em: <<https://www.yoctoproject.org/>>. Acesso em: 16 set. 2017.
34. INTEL CORPORATION. **Flashing your firmware manually** Disponível em: <<https://software.intel.com/en-us/flashing-the-firmware-on-intel-edison-board>>. Acesso em: 16 set. 2017.
35. **SSH**. Disponível em: <<https://www.ssh.com/>>. Acesso em: 16 set. 2017.
36. **Telnet**. Disponível em: <<http://www.telnet.org/>>. Acesso em: 16 set. 2017.
37. **WPA2**. Disponível em: <[https://w1.fi/wpa\\_supplicant/](https://w1.fi/wpa_supplicant/)>. Acesso em: 16 set. 2017.
38. OWASP. **Brute force attack** 31 JANEIRO 2017. Disponível em: <[https://www.owasp.org/index.php/Brute\\_force\\_attack](https://www.owasp.org/index.php/Brute_force_attack)>. Acesso em: 30 out. 2017.
39. **HTTPS**. Disponível em: <<https://tools.ietf.org/html/rfc2818>>. Acesso em: 16 set. 2017.
40. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, v.21, n.2, p.120-126, 1978. Disponível em: <<https://people.csail.mit.edu/rivest/Rsapaper.pdf>>. Acesso em: 16 set. 2017.
41. COMPTECHDOC. **Security Attacks** Disponível em: <<http://www.comptechdoc.org/independent/security/recommendations/secattacks.html>>. Acesso em: 16 set. 2017.
42. INTEL CORPORATION. **libmraa - low level skeleton library for communication on GNU/Linux platforms** Disponível em: <<https://github.com/intel-iot-devkit/mraa>>. Acesso em: 16 set. 2017.
43. **HTML**. Disponível em: <<https://www.w3.org/TR/html5/>>. Acesso em: 16 set. 2017.
44. **CSS**. Disponível em: <<https://www.w3.org/Style/CSS/Overview.en.html>>. Acesso em: 16 set. 2017.
45. **Javascript**. Disponível em: <<https://www.javascript.com/>>. Acesso em: 16 set. 2017.
46. **Bash**. Disponível em: <<https://www.gnu.org/software/bash/>>. Acesso em: 16 set. 2017.
47. **Nano**. Disponível em: <<https://www.nano-editor.org/>>. Acesso em: 16 set. 2017.
48. **BurpSuite**. Disponível em: <<https://portswigger.net/burp>>. Acesso em: 30 out. 2017.
49. **Low Orbit Ion Cannon**. Disponível em: <<https://github.com/NewEraCracker/LOIC>>. Acesso em: 30 out. 2017.

50. **OpenVAS.** Disponível em: <<http://www.openvas.org/>>. Acesso em: 14 nov. 2017.
51. **Acunetix.** Disponível em: <<https://www.acunetix.com/>>. Acesso em: 14 nov. 2017.
52. **nmap.** Disponível em: <<https://pentest-tools.com/network-vulnerability-scanning/tcp-port-scanner-online-nmap>>. Acesso em: 14 nov. 2017.
53. **nikto.** Disponível em: <<https://pentest-tools.com/website-vulnerability-scanning/web-server-scanner>>. Acesso em: 14 nov. 2017.



## ANEXO A – Códigos

Todos os códigos estão disponíveis em:

[https://github.com/sindelio/iot\\_project.git](https://github.com/sindelio/iot_project.git)

## ANEXO B – Testes básicos em sistemas IoT

Segue uma lista de testes básicos de segurança para sistemas IoT.

1. Verificar o estado de segurança de APIs, módulos, pacotes, *firmware* e qualquer outro recurso de terceiros antes de utilizá-los no sistema. Uma pesquisa por vulnerabilidades conhecidas pode evitar transtornos mais tarde.
2. Verificar a presença de encriptação no transporte de dados via Internet.
3. Verificar a resposta à injeção de código nas interfaces de rede do sistema. Em todos os campos de inserção do sistema, testar comandos como:

```
<script> alert('hacked!') </script>
```

Não deve haver resposta do sistema. Este teste também é conhecido como teste de *Cross Site Scripting – XSS*.

4. Verificar a resposta à injeção de comandos em bancos de dados, como SQL, PostgreSQL, CouchDB, MongoDB e etc. Em todos os campos de inserção do sistema, testar comandos como:

```
$username = 1' or '1' = '1
```

Não deve haver resposta do sistema.

5. Testar atravessamento ou *bypassing*. Verificar se é possível acessar recursos do sistema exclusivos à usuários autenticados através da inserção de URLs específicas no navegador. Por exemplo:

`www.site.com/login=1/private`

6. Verificar a qualidade das senhas usadas no sistema. Senhas de no mínimo 12 caracteres devem ser usadas para evitar ataques de força bruta bem sucedidos. Também é importante travar a conta caso muitas tentativas falhas de *login* ocorram. Algumas aplicações IoT vem com senhas padrão de fábrica, ressalta-se a importância de trocar estas senhas que são conhecidas por todos.
7. Verificar acesso local à “coisas” pertencentes ao sistema IoT. Todas “coisas” devem possuir sistema de autenticação para acesso local a seus recursos computacionais, preferivelmente via *handshake* criptográfico.