

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SANTA CATARINA.**

**CAMPUS JOINVILLE
CURSO SUPERIOR DE TECNOLOGIA EM
MECATRÔNICA INDUSTRIAL**

LUIS ALFREDO DA SILVA

**SISTEMA DE VIGILÂNCIA ONLINE CONFIGURADO E
MONITORADO REMOTAMENTE**

TRABALHO DE CONCLUSÃO DE CURSO

LUIS ALFREDO DA SILVA

**SISTEMA DE VIGILÂNCIA ONLINE CONFIGURADO E
MONITORADO REMOTAMENTE**

JOINVILLE, 2016

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DE SANTA CATARINA.**

CAMPUS JOINVILLE

CURSO DE MECATRÔNICA INDUSTRIAL

LUIS ALFREDO DA SILVA

**SISTEMA DE VIGILÂNCIA ONLINE CONFIGURADO E
MONITORADO REMOTAMENTE**

**Submetido ao Instituto
Federal de Educação,
Ciência e Tecnologia de
Santa Catarina como
parte dos requisitos de
obtenção do título de
Tecnólogo em
Mecatrônica Industrial.**

**Orientador: Rodrigo
Coral, Dr.**

JOINVILLE, 2016

Silva, Luis Alfredo da.

Sistema de vigilância online configurado e monitorado remotamente / Silva, Luis Alfredo da

– Joinville: Instituto Federal de Santa Catarina, 2016. 120 f.

Trabalho de Conclusão de Curso - Instituto Federal de Santa Catarina, 2012. Graduação. Curso Superior de Tecnologia em Mecatrônica Industrial. Modalidade: Presencial.

Orientador: Rodrigo Coral, Dr.

1.Câmera de vigilância 2.Código-fonte aberto 3. Raspberry Pi

I. Sistema de vigilância online configurado e monitorado remotamente

SISTEMA DE VIGILÂNCIA ONLINE CONFIGURADO E MONITORADO REMOTAMENTE

LUIS ALFREDO DA SILVA

Este trabalho foi julgado adequado para obtenção do título de Tecnólogo em Mecatrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso de Mecatrônica Industrial do Instituto Federal de Educação, ciência e Tecnologia de Santa Catarina.

Joinville, XX de dezembro de 2016.

Banca Examinadora:

Prof. Rodrigo Coral, Dr.
Orientador

Prof. Ary Victorino da Silva Filho
Avaliador

Prof. Evandro Rodrigo Dário
Avaliador

AGRADECIMENTOS

Agradeço ao professor Paulo Roberto de Oliveira Bonifácio, pelos vários anos de orientação e coordenação de projeto em iniciação científica onde estive inserido, fator de importância crucial na minha formação em Mecatrônica, pela orientação dada em grande parte da elaboração deste Trabalho de Conclusão e também pelo apoio ao trilhar caminho rumo ao Mestrado.

Ao professor Rodrigo Coral, por também me orientar ao final desse Trabalho de Conclusão, pela bolsa de monitoria em Programação no semestre final de curso e por também me apoiar no caminho rumo ao Mestrado.

Ao professor Fernando Claudio Guesser, pelos anos de orientação em bolsa de monitoria em Física, o que também contribuiu consideravelmente na minha formação, especialmente para um possível futuro na área de docência.

Aos demais professores do Instituto Federal de Santa Catarina, que tornaram possível todo o aprendizado necessário à obtenção do título de Tecnólogo em Mecatrônica Industrial.

Ao próprio Instituto Federal de Santa Catarina, por fornecer toda a infraestrutura necessária ao curso, à pesquisa e à monitoria, além dos recursos e investimento necessários para o desenvolvimento deste Trabalho de Conclusão.

À Lorena Patricia Karnopp Geremias e Ademir Geremias, pela educação familiar e o muito necessário apoio dos pais mesmo durante a graduação.

“O homem científico não pretende alcançar um resultado imediato. Ele não espera que suas ideias avançadas sejam imediatamente aceitas. Seus trabalhos são como sementes para o futuro. Seu dever é lançar as bases para aqueles que estão por vir e apontar o caminho.”

Nikola Tesla

RESUMO

Este projeto tem como objetivo a construção de um sistema de vigilância configurado e monitorado através da Web. Dois protótipos foram construídos, centrados em diferentes versões da placa Raspberry Pi. O projeto também envolve os testes experimentais dos protótipos desenvolvidos. Uma meta do projeto é criar um sistema capaz de operar com igual capacidade quando acessado via computador de mesa, tablet ou celular, e em conexões à Internet domésticas, institucionais ou corporativas. O hardware e o software envolvidos são, em sua maior parte, de código-fonte aberto (*open source*). No andamento do projeto, foi necessário o desenvolvimento de software e de *scripts* em diferentes linguagens, incluindo *shell scripts* e, especialmente, diversas tecnologias de programação e formatação para a Web, como PHP, HTML e CSS. Para permitir o acesso ao sistema via Internet, foi utilizado o Tor, sendo esse um meio inovador e ainda pouco utilizado nesse tipo de aplicação. Os resultados obtidos com os dois protótipos foram satisfatórios, com o sistema sendo capaz de detectar e gravar adequadamente o movimento, funcionar por longos períodos de tempo e retornar à operação automaticamente ao retornar de uma falha na rede elétrica.

Palavras-chave: Câmera de vigilância, Código-fonte aberto, Raspberry Pi

ABSTRACT

This project's objective is building a web-connected, remotely configured and monitored surveillance system. Two prototypes were built, each centered on a different version of the Raspberry Pi board. Additionally, the project involves the experimental tests of the finished prototypes. The intention of the project is building a system equally capable of being operated via a desktop computer, tablet or smartphone, at home, corporate or institutional Internet connections. Most of the software and hardware used is open source. During the project's progress, software development using a couple different languages was required, including shell scripts and, especially, various Web programming and formatting technologies, as PHP, HTML and CSS. To allow the remote access via Web, Tor was used, being this an innovative method that has little use in this kind of application yet. Results obtained with both prototypes were satisfactory, the system was able to detect, capture and record motion, work continuously for a long time and recover automatically from AC line failures, continuing its surveillance tasks after AC supply was back.

Keywords: Surveillance Camera, Open Source, Raspberry Pi

LISTA DE FIGURAS

Figura 1: Placas Raspberry Pi com módulo de câmera	20
Figura 2: Câmera para o Raspberry Pi	22
Figura 3: Cartões de Memória SD e MicroSD	23
Figura 4: Adaptador WiFi USB.....	24
Figura 5: Fontes de energia USB.....	25
Figura 6: Gabinete impresso em 3D para o primeiro protótipo ..	26
Figura 7: Win32DiskImager.....	28
Figura 8: Configuração DHCP WR740N.....	32
Figura 9: <i>Stream</i> ao vivo em computador desktop	56
Figura 10: Configurações em computador desktop.....	56
Figura 11: Lista de arquivos em monitor grande	57
Figura 12: Interface de Arquivos em um Tablet PC.....	58
Figura 13: Interface Web no Android 5	59
Figura 14: Falha no <i>Stream</i> ao Vivo no Nokia 808 PureView	60
Figura 15: Lista de Arquivos no Nokia 808 PureView	61
Figura 16: Sombras no navegador nativo no 808 PureView	62
Figura 17: Interface Web no smartphone Nokia 6210	63
Figura 18: Interface Web no Opera Mini em celular simples.....	64
Figura 19: Erro de renderização no login.....	65
Figura 20: Falta de <i>stream</i> no IE11.	66
Figura 21: Arquivos no IE11.....	66

Figura 22: Sucesso na configuração	67
Figura 23: Aviso de erro.....	68
Figura 24: Sem botão Sair ao desativar login.....	68
Figura 25: Conversão de estilo de fim de linha.	69
Figura 26: Balanço de Branco Pi Camera.	70
Figura 28: Interface Web Primeiro Protótipo - Fonte: O autor....	72
Figura 29: E-mails enviados pelo RPSS em rede doméstica.....	74
Figura 30: Gerador PWM i2c Adafruit.....	77

LISTA DE CÓDIGOS-FONTE

Código-fonte 1: Configuração da rede sem fios	31
Código-fonte 2: Dependências do Motion.....	33
Código-fonte 3: Download do Motion.....	34
Código-fonte 4: Permissões Motion	34
Código-fonte 5: Ajustes motion.conf	36
Código-fonte 6: Instalação do nginx.....	37
Código-fonte 7: Configuração do nginx	38
Código-fonte 8: Configuração do Tor.....	39
Código-fonte 9: Instalação do SSMTP	40
Código-fonte 10: Configuração do SSMTP.....	40
Código-fonte 11: Script de email ao iniciar movimento	42
Código-fonte 12: Configurações email em motion.conf.....	43
Código-fonte 13: Estrutura do CSS - Fonte: O autor.....	51
Código-fonte 14: Regra @media - Fonte: O autor.....	53

LISTA DE ABREVIATURAS E SIGLAS

Sigla	Significado	Tradução / Nota
ABS	Acrylonitrile butadiene styrene	Acrilonitrila butadieno estireno (material plástico)
AC	Alternating Current	Corrente Alternada
ARM	Advanced RISC Machine	Máquina Avançada RISC (arquitetura de CPU)
CNC	Comando Numérico Computadorizado	
CPU	Central Processing Unit	Unidade de Processamento Central
CSI	Camera Serial Interface	Interface serial de câmera
CSS	Cascading Style Sheets	Folhas de Estilo em Cascata
DC	Direct Current	Corrente Contínua
DHCP	Dynamic Host Configuration Protocol	Protocolo de configuração dinâmica do Host
DIY	Do-It-Yourself	Faça-você-mesmo
DNS	Domain Name System	Sistema de Nome de Domínio
EFF	Electronic Frontier Foundation	Fundação da Fronteira Eletrônica
FPS	Frames per second	Quadros por segundo

Sigla	Significado	Tradução / Nota
GPIO	General Purpose Inputs/Outputs	Entradas e Saídas de Uso Geral
GPU	Graphics Processing Unit	Unidade de Processamento Gráfico
HDMI	High-Definition Multimedia Interface	Interface multimídia de alta definição
HSDPA	High Speed Downlink Packet Access	Acesso de Pacotes de Downlink em Alta Velocidade
HTML	Hypertext Markup Language	Linguagem de Marcação de Hipertexto
IoT	Internet of Things	Internet das Coisas
LED	Light Emitting Diode	Diodo Emissor de Luz
LTE	Long Term Evolution	Evolução de Longo Prazo
MAC	Media Access Control	Controle de Acesso ao Meio
PHP	PHP: Hypertext Preprocessor	Preprocessador de Hipertexto (acrônimo recursivo)
PWM	Pulse-Width Modulation	Modulação por Largura de Pulso
RC	Radio Control	Controle por Rádio
RPSS	Raspberry Pi Surveillance System	Sistema de Vigilância com Raspberry Pi (este projeto)
SD	Secure Digital	Cartão de memória SD
SMS	Short Message Service	Serviço de Mensagens Curtas

ce

Sigla	Significado	Tradução / Nota
SMTP	Simple Mail Transfer Protocol	Protocolo Simples de Transferência de E-mail
SoC	System-on-a-Chip	Sistema em um chip
SOCKS	Socket Secure	Socket Seguro (Protocolo de comunicação via Internet)
SSH	Secure Shell	Shell Seguro (acesso remoto)
TCO	Total Cost of Ownership	Custo Total de Posse
USB	Universal Serial Bus	Barramento Serial Universal
UPS	Uninterruptible Power Supply	
V	Volt	Unidade de medida de tensão
WPA2	Wi-Fi Protected Access II	Acesso Wi-Fi Protegido versão II
X	X.Org Server	Servidor gráfico X.org

SUMÁRIO

1 INTRODUÇÃO	3
1.1. OBJETIVOS.....	5
1.2. JUSTIFICATIVAS.....	5
DO CUSTO-BENEFÍCIO:	6
DA RELAÇÃO COM AS DISCIPLINAS DO CURSO:	6
DAS CARACTERÍSTICAS E APLICAÇÕES:	7
APLICAÇÃO LABORATORIAL E INDUSTRIAL:	7
APLICAÇÃO DOMÉSTICA E AUTOMOTIVA:	8
2. FUNDAMENTAÇÃO TEÓRICA.....	12
2.1. OPEN SOURCE, CENA MAKER E INTERNET DAS COISAS	12
O MODELO OPEN-SOURCE.....	12
O MOVIMENTO MAKER	13
A INTERNET DAS COISAS	14
2.2. TECNOLOGIAS DE DETECÇÃO DE MOVIMENTO.....	16
2.3. DYNDNS E TOR	17
2.4. CONHECENDO OS COMPONENTES DE HARDWARE DO PROJETO	20
RASPBERRY PI:	20
ARM E O CHIPSET / SoC:	21
PI CAMERA MODULE:	21
CARTÃO SD:	22
ADAPTADOR WIFI:	23
A FONTE DE ENERGIA:	24
O GABINETE:	25
3. METODOLOGIA	27

3.1. OBTENDO A IMAGEM DO SISTEMA:	27
3.2. CONFIGURANDO A REDE SEM FIOS:	31
3.3. INSTALANDO O MOTION:	33
PARAMETRIZANDO O MOTION	35
3.4. CONFIGURANDO O NGINX:	37
3.5. CONFIGURANDO O TOR	39
3.6. CONFIGURANDO O SSMTP	40
CRIANDO SCRIPTS DE EMAIL	41
3.7. DESENVOLVENDO A INTERFACE WEB: HTML E PHP	44
ARQUIVOS PHP PRINCIPAIS	45
INTERPRETADORES DE CONFIGURAÇÕES (PARSERS) PHP:	46
ARQUIVOS DE CONFIGURAÇÃO:	47
ARQUIVOS CSS:	48
3.8. DESIGN RESPONSIVO E COMPATIBILIDADE	49
3.9. ESTRUTURA BÁSICA DO CSS:	50
 4. RESULTADOS	 55
 4.1 INTERFACE DE USUÁRIO:	 55
COMPUTADOR DESKTOP OU LAPTOP	55
LAPTOP / TABLET PC (1280x800)	57
SMARTPHONE ANDROID (GALAXY GRAN PRIME DUOS)	59
SMARTPHONE NOKIA 808 PUREVIEW (640x360)	59
SMARTPHONE NOKIA 6210 NAVIGATOR (320x240)	62
CELULAR NOKIA 3120 CLASSIC (320x240)	63
LIMITAÇÕES GERAIS DE COMPATIBILIDADE	64
RECURSOS ADICIONAIS DA INTERFACE WEB:	67
 4.2. PROBLEMAS ENCONTRADOS NO DESENVOLVIMENTO DE SOFTWARE:	 69
PROBLEMAS DE SOFTWARE SOLUCIONADOS:	69
O BALANÇO DE BRANCO DA CÂMERA	70

4.3. PROBLEMAS DE HARDWARE ENCONTRADOS:	71
4.4. RESULTADOS DE LONGO PRAZO DO PRIMEIRO PROTÓTIPO:	72
4.5. RESULTADOS DO SEGUNDO PROTÓTIPO:.....	74
ENVIO DE E-MAILS EM REDE COM RESTRIÇÕES DE ACESSO .	74
 5. TRABALHOS FUTUROS	76
 5.1. INSTALAÇÃO DE ATUADORES	76
5.2. DESENVOLVIMENTO DE UMA INTERFACE DE CONFIGURAÇÃO EMERGENCIAL	77
5.3. ALTERAÇÕES NO SISTEMA DE ENVIO DE E-MAILS	78
5.4. CONEXÃO À REDE CELULAR E ENVIO DE SMS	79
5.5 DESENVOLVIMENTO DE UM SISTEMA AUTOMÁTICO DE OBTENÇÃO DE BRIDGES TOR	80
 CONCLUSÃO	81
 REFERÊNCIAS.....	82
 BIBLIOGRAFIA COMPLEMENTAR	85
 GLOSSÁRIO	86

1 INTRODUÇÃO

Com o crescimento da criminalidade e o sentimento de insegurança presente na população, o mercado de segurança eletrônica vem tendo um avanço considerável (Vieira, R., 2009)

Visando estudar uma das formas de atender a esse mercado, este trabalho consiste essencialmente em desenvolver um sistema de vigilância prático e inteligente, a um custo possivelmente reduzido, para uso doméstico, acadêmico ou mesmo em alguns setores da indústria.

No centro do sistema é utilizada uma placa com capacidades de computador embarcado, o Raspberry Pi, usado em projetos embarcados diversos -- como exemplo, o monitoramento de atividades de natação (Raulino, M., 2013), também desenvolvido no Instituto Federal de Santa Catarina. A placa também é muito utilizada pela cena *maker open source* mundial.

O software utilizado também é primariamente de código-fonte aberto, e este projeto inclui o desenvolvimento de adaptações e implementação de novos recursos de software, onde se fez necessário, especialmente no desenvolvimento de uma inovadora e intuitiva interface de usuário via Web, capaz de ser operada adequadamente tanto em computadores quanto *tablets*, *smartphones* e até mesmo telefones celulares mais simples.

Dado que o projeto utiliza plataformas inovadoras (a placa utilizada no protótipo final foi lançada em 2015) e é primariamente inspirado em tecnologias comumente utilizadas pelo movimento *maker*, existe muito pouco material puramente acadêmico (na forma de livros e artigos científicos) relacionados

com os meios utilizados na elaboração final do projeto. Nas poucas situações em que há referências acadêmicas, procuramos sempre citá-las (como nas áreas relacionadas a fundamentos teóricos envolvidos no projeto), mas, especialmente nas etapas de construção e configuração efetivas do protótipo, foi necessário o uso de referências (como fóruns e documentação online) que não seguem estritamente padrões acadêmicos. Averiguamos experimentalmente todas as citações para garantir a veracidade e aplicabilidade das mesmas.

Outros projetos bastante similares ao apresentado nesse trabalho, em maior ou menor grau, foram demonstrados anteriormente na comunidade científica internacional:

- Wolf, W. et al. (2002) mencionou a chegada de câmeras inteligentes capazes de detectar e analisar movimento já em 2002, quando as tecnologias necessárias a um sistema do tipo estavam começando a se tornar disponíveis.
- Ansari, A.N. et.al. (2015) apresenta um "sistema de alarme de segurança" capaz de detectar movimento e enviar fotos e vídeos para um serviço online de armazenamento de informação ("servidor na nuvem"), além de permitir "conferência remota de atividade e obtenção de notificações em caso de movimento, através de uma aplicação baseada em Internet das Coisas (*Internet of Things* ou *IoT*). O sistema de Ansari A.N et al se diferencia do aqui apresentado especialmente por ser projetado para operar primariamente em conjunto com um serviço de armazenamento de informação externo ao hardware embarcado.

- Menezes, V.a et al. (2015), por outro lado, também apresenta um projeto de “vigilância e monitoramento usando *Raspberry Pi*”, com detecção de movimento e *streaming* ao vivo, porém as opções de software utilizadas por Menezes, V.a et al. (2015) foram bastante distintas daquelas utilizadas no presente projeto, tornando essa uma solução diferente para um problema similar.

1.1. Objetivos

- Construir um sistema de vigilância utilizando uma placa Raspberry Pi, uma câmera, adaptador AC e módulo para acesso sem fios à Internet (WiFi);
- Atuar alarmes e/ou gravar eventos sob demanda, ou quando a cena estiver em movimento;
- Permitir a visualização online, ao vivo, da imagem da câmera, e o download de gravações via navegador através de uma interface Web;
- Construir um gabinete personalizado para o conjunto, impresso em 3D;
- Aplicar os conhecimentos adquiridos nas várias disciplinas do curso, especialmente Programação, Eletrônica Digital, Eletrônica Analógica, Desenho Técnico, Processos de Fabricação, Projetos de Máquinas, entre outras.

1.2. Justificativas

O projeto, aqui também denominado *RPSS* (de *Raspberry Pi Surveillance System*), visa aproveitar os

conhecimentos obtidos no curso de Mecatrônica Industrial no desenvolvimento de um sistema de vigilância inteligente, com diversos recursos, incluindo os acima citados, e possuindo um custo final reduzido se comparado às opções disponíveis comercialmente.

Do custo-benefício:

Os sistemas de vigilância atualmente disponíveis no mercado, em geral, ou possuem custo exageradamente elevado, ou são excessivamente simplistas, muitas vezes utilizando tecnologias já bastante defasadas e carecendo de recursos essenciais, tais como detecção de movimento com gravação sob demanda, monitoramento online via Web e avisos por e-mail. O sistema aqui apresentado tem custo total aproximado de 500 reais, na data de publicação deste trabalho.

Da relação com as disciplinas do curso:

A principal disciplina aproveitada para o projeto é a de Programação, especialmente em razão dos conceitos de lógica de programação apresentados. A linguagem C, um dos tópicos da disciplina, é bastante similar em alguns aspectos ao PHP, que é a principal linguagem utilizada no projeto (mas não a única, também são utilizados CSS, HTML e *Shell Script*).

Para o entendimento adequado do funcionamento de diversos componentes do sistema, vários conceitos de Eletrônica Digital, Eletrônica Analógica e mesmo Eletrônica de Potência foram necessários.

O gabinete, impresso em 3D, se utilizou especialmente de conhecimentos em Desenho Técnico, para selecionar o

modelo de gabinete adequado, efetuar as adaptações devidas e prepará-lo para fabricação através de impressão 3D.

Das características e aplicações:

O projeto é pensado de forma a suprir simultaneamente as necessidades de diversas aplicações, que não se limitam a uma única classe ou ambiente, ou seja, há aplicações laboratoriais, acadêmicas, residenciais, automotivas e industriais prontamente adequadas ao sistema desenvolvido.

Aplicação Laboratorial e Industrial:

A primeira aplicação onde o sistema (especificamente, o primeiro protótipo) foi extensivamente testado é a de vigilância laboratorial: Várias características desejáveis para essa aplicação são cobertas pelo RPSS:

- A possibilidade de envio instantâneo de avisos por e-mail, tanto para início de atividade de movimento quanto para fim de atividade de movimento, incluindo imagem causadora do "gatilho" de envio do e-mail;
- A possibilidade de se utilizar múltiplos RPSS independentes numa mesma rede, sem trazer qualquer conflito ou complexidade adicional;
- O método de acesso via Web robusto e prontamente compatível com várias topologias mais restritas de rede, como as utilizadas no meio acadêmico e industrial.

Além de laboratórios, diversas outras instalações industriais poderiam ser cobertas por uma ou mais unidades do RPSS. No ramo industrial, são características atraentes apresentadas pelo RPSS:

- O baixo Custo Total de Posse (*TCO* ou *Total Cost of Ownership*);
- O sistema de *login* (de uso opcional) robusto, com suporte a senhas alfanuméricas e endereço de acesso secreto, personalizável ou gerado aleatoriamente através da página de configurações na interface Web.

Todas essas características citadas para os meios laboratorial e industrial tornam o sistema adequado para, por exemplo, monitorar o funcionamento de um equipamento, como um robô, uma impressora 3D ou uma máquina CNC, dentre outros, permitindo que alguém seja imediatamente informado por e-mail, quando, por exemplo, cessar o movimento da máquina em questão, indicando o término de sua tarefa ou alguma possível falha.

O RPSS foi efetivamente implementado e utilizado em sua primeira iteração nas instalações do Laboratório de Projetos Mecânicos do Campus Joinville do IF-SC, um laboratório acadêmico. Mais detalhes sobre esse experimento são apresentados ao final desse trabalho, no capítulo de Resultados.

Aplicação doméstica e automotiva:

No ramo residencial, são as características mais altamente desejáveis trazidas pelo RPSS (todos implementados no protótipo mais atual apresentado):

- O baixo custo de aquisição (especialmente comparado a outros sistemas *on-line*);
- A capacidade de se conectar a uma rede sem fios prontamente disponível e permitir acesso ao *stream* ao

vivo e aos arquivos diretamente pela Internet, em qualquer lugar, sem necessidade de complexas configurações adicionais nem no dispositivo nem em roteadores;

- A possibilidade de operar com um mínimo de fios em locais de difícil acesso: apenas uma tomada elétrica é necessária, sem necessidade de cabeamento adicional para conexão à rede ou similar;
- A capacidade de utilizar um *smartphone*, *tablet* ou mesmo celular comum, para toda a operação e configuração do sistema, online, através de uma interface de usuário intuitiva e adequadamente adaptada a cada dispositivo.

A alimentação via USB permite o uso de determinados tipos de *power banks* (tipicamente usados como carregadores portáteis de celular) na função de UPS (Unidade de fornecimento ininterrupto de energia), permitindo a continuidade do funcionamento do sistema mesmo que ocorra corte total ou parcial no fornecimento da rede elétrica;

Essas características facilitam o uso do sistema desenvolvido, na vigilância de varandas, áreas de entrada e saída residenciais e até mesmo áreas internas residenciais, como um quarto de bebê.

O RPSS também apresenta aplicações no ramo automotivo, especialmente com a possibilidade de operação na forma de *dashcam*, uma câmera interna em um veículo, usada com o intuito de registrar eventos que ocorram especialmente em frente ao veículo, na estrada. Para esse tipo de aplicação, são características desejáveis e entregues pelo RPSS:

- A alimentação via USB e o baixo consumo elétrico, que permitem que o sistema permaneça em funcionamento constantemente, o que permite gravar eventos que ocorram quando o veículo estiver estacionado (como uma tentativa de furto ou adulteração);
- A possibilidade de operar sem conexão alguma com a Internet, gravando e salvando seus registros em cartão de memória;

Embora não implementado no protótipo mais atual, é tecnicamente possível fazer com que o sistema opere conectado à internet diretamente através de um modem *HSDPA/3,5G* ou mesmo *LTE/4G*, o que permitiria que, em caso de sinistro, o sistema enviasse imediatamente e-mail apresentando a cena, mesmo que a câmera seja percebida e desligada após o sinistro, em um veículo sem conexão *WiFi*. Também é possível adicionar ao sistema a possibilidade de incluir hardware receptor GPS para geolocalização de arquivos gravados.

A possibilidade de se utilizar uma conexão móvel também torna o sistema adequado para aplicações como monitoramento de caminhões (permitindo conferir remotamente e ao vivo o estado do caminhão e/ou de sua carga) e ônibus.

Para o protótipo mais atual, é possível estabelecer esse cenário corretamente contanto que o veículo apresente uma conexão à internet compartilhada via *Wi-Fi* com o RPSS.

O uso como *dashcam* (e em algumas outras aplicações), no entanto, pode ser severamente limitado em razão da resolução e taxa de quadros suportados pelo sistema quando operando com detecção de movimento, em razão da capacidade

de processamento limitada do Raspberry Pi, como veremos durante este trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Open source, cena Maker e Internet das Coisas

O modelo *Open source*

O modelo de código-fonte aberto, ou *open source*, originalmente aplicado a softwares, foi posteriormente ampliado de forma a cobrir diversas outras áreas de conhecimento, como hardware, desenho (como projetos 3D editáveis de uso livre), alimentos, bebidas e até mesmo medicamentos (Munos, B. 2006).

Além do uso livre, o modelo permite a livre colaboração no desenvolvimento, permitindo a entrada de um maior número de colaboradores em projetos com grande demanda e trazendo vantagens para todos os usuários do projeto, em um modelo privado-coletivo (Von Hippel, E.a et al. 2003), que adicionalmente traz aprendizado a todos os envolvidos (Lakhani, K.R. et al, 2003).

Em 2012, software open source era encontrado em 75% dos 10 mil websites mais utilizados na Internet (Pingdom.com, 2012) incluindo, por exemplo, o *kernel* (núcleo do sistema operacional) Linux, os servidores Web Apache e nginx e a linguagem PHP.

Esses são softwares também utilizados pelo RPSS, projeto aqui apresentado. O uso de código aberto foi uma característica essencial no desenvolvimento deste projeto, tanto em razão da existência de software adequado ao projeto quanto da filosofia do código-fonte aberto e sua natural abundância de documentação.

O Movimento Maker

Nos tempos recentes, cresceu a popularidade de projetos do tipo faça-você-mesmo (*do-it yourself* ou *DIY*), causando o surgimento e expansão do movimento (ou cena) *Maker*, com um conjunto de valores enfatizando compartilhamento, aprendizado e criatividade acima de lucro ou capital social (Kuznetsov, S. et.al. 2010). Tópicos de interesse típicos no movimento *Maker* incluem especialmente desafios de engenharia em áreas como eletrônica, robótica e impressão 3D (Papavlasopoulou, S. et al. 2017).

Papavlasopoulou, S. et al. (2017) escreve que o movimento *Maker* surgiu e cresceu em razão de diversos avanços técnicos e de infraestrutura moderno. Podemos incluir nesses avanços:

- A maior disponibilidade de acesso à Informação com a expansão da Web;
- A facilidade de acesso a software de código aberto bem-documentado;
- O acesso mais fácil e barato a ferramentas (Nascimento, S. et al. 2016), como hardware de baixo custo;
- Linguagens de programação de alto nível e fácil aprendizado, entre outros.

A maior disponibilidade de tecnologias de prototipagem material acessível, especialmente o advento da impressão 3D, também auxiliou consideravelmente o avanço da cena *Maker* (Valpreda, F. 2015).

A cena *Maker*, composta por programadores de software de código-fonte aberto, hackers de hardware, dentre outros

(como apoiadores de culturas de autossuficiência ou hobbistas diversos), é mantida coesa através de micro gerenciamento, ou "gerenciamento molecular" (von Busch, O. 2013), e é capaz de trazer evolução tecnológica e criativa que possui implicações ilimitadas no mundo da educação (Papavlasopoulou, S. et al. 2017).

Muitas dos conceitos e tecnologias utilizados pela comunidade *open source* e pelos adeptos do movimento *Maker* foram adequadamente aproveitados no desenvolvimento do projeto em apresentação neste trabalho, tornando a cena *Maker* um pilar crucial no desenvolvimento deste projeto.

A Internet das Coisas

Uma área emergente e largamente visada pelo movimento *Maker* é a Internet das Coisas (*Internet of Things* ou *IoT*).

A Internet das Coisas corresponde à interconexão de dispositivos embarcados inteligentes, através da Internet. Estima-se que em 2020 exista um total de 50 bilhões de dispositivos conectados (Evans, D. 2011), e parte considerável desse avanço da Internet das Coisas deve ser proveniente do Movimento *Maker*, visto que, por exemplo, na China (onde o movimento *Maker* está tomando relevância substancial) a cena atual está concentrada em design de hardware relacionado à Internet das Coisas (Lindtner, S. 2014).

Em IoT, sensores e atuadores conectam-se à Internet, tipicamente através de protocolo IP, oferecendo novas formas de interação com consumidores, aprimorando processos para

melhoria de produtividade, automatizando tarefas perigosas e melhor gerenciando riscos (CHUI, M. et.al. 2010)

IoT está comumente relacionada também à computação e armazenamento em nuvem (*cloud computing* e *cloud storage*, onde o processamento ou armazenamento é fornecido via internet por um terceiro como o Google ou a Amazon), embora esse não seja um requisito essencial aos dispositivos integrantes da Internet das Coisas.

Para a Internet das Coisas, os "serviços na nuvem" permitem que o processamento ou armazenamento de informação seja transferido para um terceiro através da Internet, diminuindo a carga sobre o hardware embarcado muitas vezes bastante limitado do dispositivo, ou mesmo permitindo a adição de recursos outrora indisponíveis no mesmo hardware.

Um exemplo de projeto voltado à Internet das Coisas e o Movimento *Maker* também é o apresentado por Ansari, A.N. et.al. (2015), com seu "sistema de alarme de segurança" capaz de detectar movimento e enviar fotos e vídeos para um serviço online de armazenamento de informação ("servidor na nuvem"), além de permitir "conferência remota de atividade e obtenção de notificações em caso de movimento, através de uma aplicação baseada em *IoT*".

Em nosso projeto, optamos por não utilizar qualquer serviço comercial de armazenamento na nuvem, focando em garantir a independência funcional do sistema (que pode ser acessado via Web, sem necessidade de um servidor externo muitas vezes administrado por uma entidade terceira, tratando-se, portanto, de um sistema descentralizado).

2.2. Tecnologias de detecção de movimento

A detecção de movimentos é uma área da ciência de processamento de imagens e visão computacional, e envolve especialmente:

- Identificação de ocorrência de movimento;
- Identificação e acompanhamento de objetos móveis;
- Supressão de ruídos, sombras e outros fatores capazes de dispararem falsos-positivos.

Yilmaz, A.ad et al (2006) apresenta uma descrição detalhada dos diferentes métodos computacionais conhecidos para detecção de movimento associada a rastreamento de objetos.

Pesquisa e Desenvolvimento relacionados à vigilância de humanos e veículos tem utilizado e aprimorado as tecnologias de detecção de movimento, como demonstrado por trabalhos como o de Hu, W.a et al (2004) e Cucchiara, R.a et al (2003), este último mais voltado a técnicas de identificação e separação entre objetos e suas sombras.

Estudos mais recentes têm incluído conceitos como o de redes neurais artificiais (Maddalena, L.a. et al, 2008) nas técnicas de detecção e identificação de movimento.

No meio nacional, um projeto relacionado à detecção de movimento foi apresentado por Sonaglio, S. (2009), em trabalho relacionado ao curso de Telecomunicações do IF-SC.

2.3. DynDNS e Tor

Para permitir o acesso do RPSS através da Internet, utilizamos o *Tor*.

Conforme a documentação oficial (endereço Web torproject.org/about/overview.html.en), *Tor* corresponde a uma rede de servidores voluntários cujo principal objetivo é auxiliar na privacidade e segurança na Internet, sendo a ferramenta recomendada pela *Electronic Frontier Foundation (EFF)* para a proteção das liberdades civis on-line.

Por essa razão, o *Tor* é largamente utilizado por jornalistas, ativistas políticos, pessoas vivendo em locais onde um *firewall* censura o acesso à informação na Web, ou mesmo pessoas comuns buscando proteger a própria privacidade em uma rede que cada vez mais vende dados de seus usuários de forma comumente inescrupulosa (endereço Web torproject.org/about/torusers.html.en)

Não são essas as razões para o uso do *Tor* no RPSS, mas essas características tornam o *Tor* bastante robusto para operação em condições adversas de permissão de acesso à internet, como em redes protegidas por *firewalls* corporativos ou institucionais.

O uso do *Tor* nesse tipo de aplicação é inovador, visto que a maioria dos projetos similares se utilizam de conexão direta ao IP do cliente, normalmente auxiliada por um serviço de abstração do endereço IP (que é tipicamente dinâmico e varia constantemente em conexões domésticas).

Sistemas do tipo se utilizam de serviços como o *No-ip* ou o *DynDNS*, que funcionam da forma a seguir:

1. Um usuário registrado numa conta do No-ip, DynDNS ou similar, configura um endereço amigável para acesso ao seu servidor na sua conta do *No-ip*, *DynDNS* ou similar;
2. O usuário instala um programa ou script no seu servidor que envia para o *No-ip*, *DynDNS* ou similar o novo endereço IP do servidor toda vez que este sofrer alteração;
3. Para acesso ao seu servidor, o usuário utiliza o endereço amigável configurado na sua conta do *No-ip*, *DynDNS* ou similar, que o redireciona para o endereço IP atual do serviço.

Embora simples de implementar e popular para aplicações onde servidores Web domésticos são necessários, esse tipo de sistema possui severas limitações quando utilizado com conexões à Web restritas: computadores externos precisam ter acesso direto ao servidor, o que normalmente não é possível, por exemplo, em redes corporativas e institucionais.

Como o RPSS tem o objetivo de permanecer funcional tanto em aplicações domésticas quanto corporativas e institucionais (especialmente considerando que o primeiro protótipo já operava em laboratório do campus Joinville do Instituto Federal de Santa Catarina), uma solução para esse problema era necessária e foi encontrada ao, em vez de criar um servidor Web da forma acima, criar um *serviço oculto do Tor*.

Como visto anteriormente, o *Tor* é comumente utilizado em aplicações onde são necessários o anonimato e a ocultação da localização física de seus clientes e servidores. Para o RPSS, não precisamos dessa capacidade, em vez disso, o *Tor* é

utilizado pela sua capacidade de funcionar de forma adequada e confiável criando um servidor Web mesmo sob uma rede institucional ou corporativa que contenha restrições diversas de acesso, sem qualquer necessidade de reconfiguração constante de endereços IP, DNS ou registro de domínio.

Para acessar diretamente um serviço oculto do *Tor*, é necessário um cliente *Tor* instalado no dispositivo, o que pode não ser possível ou prático em *tablets* e celulares.

Uma alternativa, que permite o acesso ao serviço oculto através de qualquer computador ou dispositivo, é o uso de um *proxy* como o Tor2Web ou o Onion.TO, que roteiam uma conexão direta à Web através do *Tor*, permitindo acesso a serviços ocultos sem necessidade de um cliente *Tor* instalado.

Essa alternativa, no entanto, pode limitar a funcionalidade e desempenho da aplicação (visto que há uma camada adicional de abstração que precisa interpretar e adaptar todo o tráfego), embora seu funcionamento tenha se comprovado adequado para conexão ao RPSS através de um dispositivo móvel.

2.4. Conhecendo os componentes de Hardware do Projeto

Raspberry Pi:

Raspberry Pi é a marca dada a uma série de computadores de baixo custo, desenvolvido pela Fundação Raspberry Pi, open source e originalmente desenvolvido para uso especialmente em escolas (Edwards, C. 2013)

Raspberry Pi está disponível em diversas configurações, com diferentes limitações, todas com placa de circuito impresso de dimensões iguais ou menores que um cartão de crédito.

No projeto, utilizamos placas Raspberry Pi B para o primeiro protótipo e Raspberry Pi 2 B para o protótipo final (Figura 1, esquerda e direita, respectivamente).

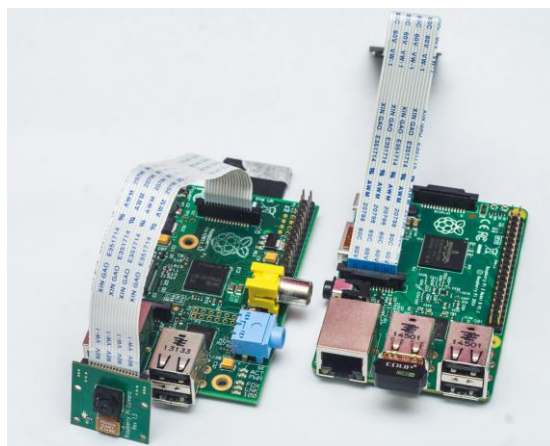


Figura 1: Placas Raspberry Pi com módulo de câmera – Fonte: O autor

ARM e o chipset / SoC:

O mercado de servidores, computadores pessoais e portáteis é atualmente dominado pela arquitetura de CPU x86, enquanto o mercado de dispositivos móveis menores, como celulares ou tablets pertence primariamente à arquitetura de CPU ARM (Blem, E. et al. 2013)

Diferente dos computadores pessoais e servidores tradicionais, que utilizam as CPUs com conjunto de instruções x86, as versões do Raspberry Pi são baseadas em CPUs com o mesmo conjunto de instruções ARM comumente utilizado em dispositivos móveis menores.

Tal característica limita a compatibilidade de hardware e software entre o Raspberry Pi e os computadores mais comuns no mercado, especialmente softwares de código-fonte fechado, visto que a maior parte desses existe apenas para a arquitetura x86. Vários drivers de dispositivo também não podem não oferecer suporte ao Raspberry Pi em razão da CPU ARM.

Porém, dentro do ecossistema de código-fonte aberto, já existe uma grande gama de aplicações suportadas, visto que a comunidade *open source* é que faz as adaptações necessárias para garantir suporte adequado ao hardware do Raspberry Pi. Para o projeto em apresentação, não encontramos restrições causadas por essa limitação.

Pi Camera Module:

Módulo de câmera nativo para o Raspberry Pi, que utiliza conexão e interface desenvolvidas especialmente para o Raspberry Pi (CSI).

O módulo possui sensor de 1/4" e, na época em que o projeto começou a ser desenvolvido, existia em duas versões: com filtro infravermelho e sem esse filtro (*NoIR*), ambas com resolução do sensor de 5 megapixels. Versões com um sensor atualizado, de 8 megapixels, foram lançadas posteriormente.

Utilizamos a opção com o filtro em razão da maior disponibilidade no mercado, embora a versão *NoIR* pudesse apresentar vantagens na aplicação, ao permitir utilização em conjunto com *LEDs* infravermelhos capazes de iluminar a cena sem que humanos percebam a luminosidade.



Figura 2: Câmera para o Raspberry Pi – Fonte: O autor

Cartão SD:

Para ser inicializado e utilizado, o Raspberry Pi necessita de um cartão de memória contendo seu sistema operacional. O cartão de memória segue o padrão SD, no formato SD comum para o Raspberry Pi B do primeiro

protótipo e no formato microSD para o Raspberry Pi 2 B do protótipo final (Figura 3).

O desempenho e a durabilidade do cartão de memória são fatores importantes na escolha desse componente. Para os dois protótipos, utilizamos cartões com taxa de escrita sequencial de pelo menos 20 MB/s e, principalmente, bom desempenho de leitura e escrita aleatórias.



Figura 3: Cartões de Memória SD e MicroSD - Fonte: O autor

Adaptador WiFi:

O sistema desenvolvido visa utilização mínima de fios, com a exigência única de um cabo apenas para alimentação.

De forma a suprir esse requisito, uma conexão sem fios à rede e à Internet é necessária, recurso que não está incluso nas placas Raspberry Pi.

Um adaptador WiFi USB (Figura 4) é utilizado para permitir a conexão sem fios à rede.



Figura 4: Adaptador WiFi USB - Fonte: O autor

A fonte de energia:

O Raspberry Pi é alimentado com 5 V através de uma fonte de energia *microUSB* (Figura 5), semelhante a carregadores de celular.

Há restrições na escolha da fonte, entretanto, especialmente na capacidade mínima de corrente: para o Raspberry Pi B são necessários 700 mA e para o Raspberry Pi 2 B, 1500 mA. Por essa razão, a maioria dos carregadores de celular (que não são capazes de fornecer a corrente necessária) não são adequados.



Figura 5: Fontes de energia USB - Fonte: O autor

O Gabinete:

Para proteger o conjunto Raspberry Pi / Câmera / Cartão SD, optamos por um gabinete projetado especialmente para o conjunto, produzido através de impressão 3D em plástico ABS (Figura 6)

Duas versões diferentes do gabinete foram utilizadas, uma para cada protótipo (visto que as versões utilizam placas com *layout* diferente). Utilizamos modelos 3D editáveis disponíveis online, como referência para a construção do gabinete, que sofreu adaptações através de edições no software *SolidWorks 2013*.

A impressora utilizada foi uma *Up! Mini*, com características de tamanho de impressão, resolução e material adequados para as exigências do gabinete.

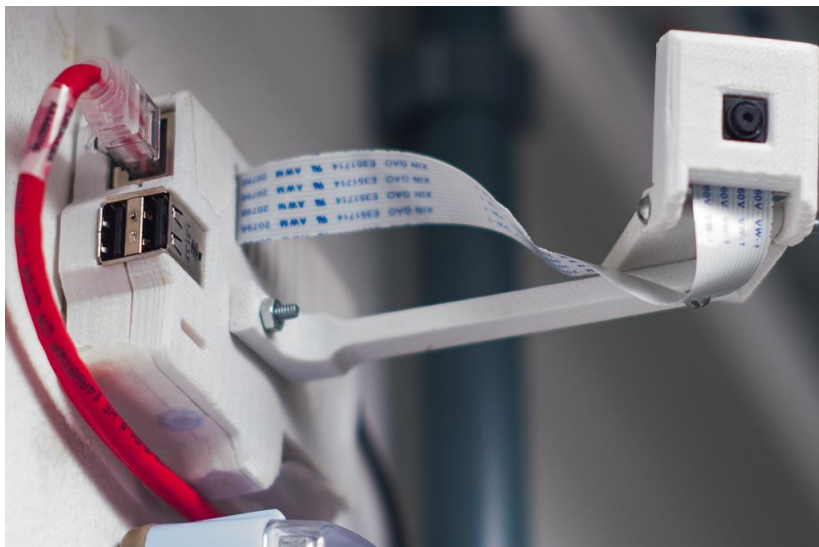


Figura 6: Gabinete impresso em 3D para o primeiro protótipo
Fonte: O autor

3. METODOLOGIA

Neste capítulo tratamos sobre todas as etapas de desenvolvimento do projeto, com detalhamento suficiente de forma a torná-lo reproduzível.

O projeto se desenvolveu ao juntar conhecimentos e tecnologias de diversas fontes, muitas delas provenientes de documentação de hardware e software de código aberto, on-line, cujo ano de publicação e autor não são conhecidos. Por esta razão, fontes que seguem esse estilo foram referenciadas diretamente pelo endereço da Web, durante o texto.

3.1. Obtendo a imagem do sistema:

Tal qual um computador, Raspberry Pi necessita de um sistema operacional. Optamos por utilizar o *Raspbian*, o sistema oficialmente suportado pela Fundação Raspberry Pi, obtido no link <https://www.raspberrypi.org/downloads/> a partir de um computador qualquer contendo um leitor de cartões de memória SD

O primeiro protótipo foi desenvolvido com a versão do Raspbian disponível em março de 2015, *Wheezy*, enquanto o protótipo final usa o Raspbian *Jessie*, obtido em agosto de 2016. As instruções a seguir se baseiam no preparo do protótipo mais recente.

Após o *download* do arquivo de imagem de disco no endereço citado, a imagem do sistema operacional deve ser transferida para um cartão de memória a ser instalado no Raspberry Pi. Para a instalação da imagem do sistema operacional no cartão de memória, utilizamos o aplicativo de

código aberto *Win32DiskImager*, obtido em <https://sourceforge.net/projects/win32diskimager/>

Não é possível copiar diretamente os arquivos do Raspbian utilizando o Windows Explorer porque o Raspbian requer configurações específicas de formato do sistema de arquivos do cartão de memória, onde parte do cartão usa um sistema de arquivos (*ext4*) diferente dos suportados pelo Windows por padrão.

Instalado o *Win32DiskImager*, inserimos um cartão de memória vazio no computador (que será utilizado no Raspberry Pi), abrimos o programa, selecionamos o arquivo de imagem do Raspbian em "*Image File*" e a letra de unidade do leitor de cartão em "*Device*". Clicamos em Write e aguardamos a conclusão do processo (Figura 7):

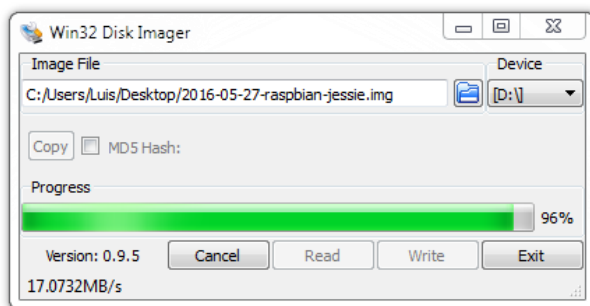


Figura 7: Win32DiskImager
Fonte: O autor.

Concluída a gravação da imagem, instalamos o cartão no Raspberry Pi, conectamos teclado e mouse USB, monitor HDMI, Câmera e fonte microUSB, o que executa a primeira inicialização

do sistema operacional, onde devem ser feitas algumas configurações:

Abrimos o terminal (figura) e executamos `sudo raspi-config` para efetuar as primeiras configurações essenciais do sistema operacional, conforme a documentação oficial em <https://www.raspberrypi.org/documentation/configuration/raspi-config.md> :

Alteramos a senha padrão do usuário selecionando "*2 Change Upser Password*". Para os fins desse projeto, a nova senha do protótipo foi definida como:

```
rpssifsc
```

Adicionalmente, em "*3 Boot Options*", selecionamos "*B1 Console*", visto que o sistema de vigilância deverá funcionar de forma autônoma e não terá monitor, teclado ou mouse conectados localmente. Essa operação desabilita o início automático da interface gráfica do sistema operacional e desabilita o *login* automático na conta do sistema (visto que o acesso local não é uma interface para o usuário final do sistema, que acessará a câmera apenas por uma interface via Web).

Configuramos também a opção "*4 Change Wi-Fi Country*" em "*5 Internationalization Options*" como "*BR Brazil*", para certificar-se que a faixa de frequência utilizada pelo Raspberry Pi será a correta e permitida no país.

Também alteramos o layout do teclado de acordo com o teclado utilizado durante a configuração do protótipo ("*Generic 105-key (intl)*") depois "*Portuguese (Brazil)*", mantendo os ajustes restantes no valor padrão). É apresentada uma solicitação para definir a função do atalho de teclado *Ctrl+Alt+Backspace* após a

configuração do layout do teclado, se a mesma deve ou não terminar o servidor gráfico X. A escolha é indiferente para o projeto, visto que não utilizaremos o servidor gráfico X.

Ainda dentro do aplicativo *raspi-config*, selecionamos "6 *Enable Camera*" para habilitar o módulo de câmera instalado no Raspberry Pi. Depois, selecionamos "9 *Advanced Options*" e "A4 *SSH*", para habilitar o servidor SSH, que será útil posteriormente em tarefas de configuração e manutenção durante o desenvolvimento do projeto.

Selecionamos "*Finish*", para terminar a aplicação e aceitamos a solicitação para reiniciar o sistema operacional.

Após o reinício, fazemos *login* com o usuário "*pi*" e a senha que definimos anteriormente. O próximo passo é configurar a rede sem fios.

3.2. Configurando a rede sem fios:

Para a primeira configuração do protótipo, antes da interface de usuário via Web estar desenvolvida, configuramos a rede sem fios em acordo com a documentação oficial em <https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>, da forma a seguir.

Após o *login* no sistema operacional, executamos o comando:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Neste documento, que rege a conexão de rede sem fios, incluímos o seguinte:

```
network={
    ssid="Projetos521"
    psk="ifscprojjoi123"
}
```

Código-fonte 1: Configuração da rede sem fios - Fonte: O autor

Onde *Projetos521* é o nome da rede sem fios que utilizamos em nosso roteador e *ifscprojjoi123* é a senha (*chave WPA2-Personal*) da rede.

Os próximos ajustes podem ser feitos por acesso remoto via SSH, que habilitamos anteriormente, através da rede sem fios recém-configurada.

Fixando o endereço IP:

Para facilitar o acesso remoto e as configurações iniciais (optamos por efetuar testes em rede local antes do sistema ser

totalmente acessível via Web), fixamos o endereço IP local do Raspberry Pi através do painel de controle do roteador.

Com o Raspberry Pi ligado e agora conectado à rede, abrimos, em um navegador em um computador conectado à mesma rede, a página `http://192.168.0.1`. O endereço e as opções de configuração do painel de controle variam com a marca e modelo do roteador, utilizamos um *TPLink TL-WR740N*.

Fazemos *login* no roteador e, em DHCP > Lista de Clientes (Figura 8), encontramos o endereço MAC do dispositivo “*raspberrypi*” conectado à rede. Copiamos este endereço MAC e, em DHCP > Reserva de Endereços, definimos um endereço reservado.



ID	Endereço MAC	Endereço IP Reservado	Estado	Modificar
1	00-23-14-8D-5C-E8	192.168.0.5	Habilitado	Modificar Apagar
2	00-80-2F-15-7D-E7	192.168.0.16	Habilitado	Modificar Apagar
3	E8-4E-06-29-0F-92	192.168.0.2	Habilitado	Modificar Apagar
4	B8-27-EB-F7-8D-25	192.168.0.3	Habilitado	Modificar Apagar

Figura 8: Configuração DHCP WR740N – Fonte: O autor

Note que o endereço MAC é associado ao adaptador de rede sem fios (no caso, um dispositivo USB) e essa configuração deve ser refeita caso o adaptador seja eventualmente substituído. No entanto, essa preocupação deve se tornar desnecessária assim que o acesso via Tor esteja configurado.

Salvamos as configurações e reiniciamos o roteador.

3.3. Instalando o Motion:

Agora que a conexão à rede sem fios está pronta, instalamos e configuramos alguns parâmetros essenciais relacionados com a câmera e a detecção de movimentos.

Utilizamos o Motion, aplicação open source para detecção de movimento, para identificar e gravar a saída da câmera sob demanda. Especificamente, para o protótipo final (que executa o sistema operacional Raspbian Jessie), utilizamos o *fork* de LowflyerUK, que é adequado para operar com a câmera do Raspberry Pi.

As instruções de instalação utilizadas, do próprio LowflyerUK, são encontradas no link <https://www.raspberrypi.org/forums/viewtopic.php?p=864996#p864996>:

Para instalar as dependências necessárias, fazemos *login* no sistema operacional e executamos:

```
sudo apt-get install -y libjpeg-dev libavformat56
libavformat-dev libavcodec56 libavcodec-dev libavutil54
libavutil-dev libbc6-dev zlib1g-dev libmysqlclient18
libmysqlclient-dev libpq5 libpq-dev
```

Código-fonte 2: Dependências do Motion - Fonte: O autor

Após a conclusão desse comando, baixamos o Motion e o descompactamos com os comandos:

```
wget https://www.dropbox.com/s/6ruqgv1h65zufr6/motion-mmal-
lowflyerUK-20151114.tar.gz
tar -zxvf motion-mmal-lowflyerUK-20151114.tar.gz
```

Código-fonte 3: Download do Motion

Para fins de organização e simplicidade, movemos o executável do Motion e seu arquivo de configuração para `/usr/bin/motion` e `/etc/motion.conf`, respectivamente, de forma similar a apresentada por Christoph Buenger e DaSports (2014), que apresentam um tutorial voltado para versões mais antigas do Raspbian e do Motion-MMAL. Também ajustamos as permissões de arquivos de forma similar à apresentada no tutorial especificado acima.

```
sudo mv motion /usr/bin/motion
sudo mv motion-mmcam-both.conf /etc/motion.conf
sudo chmod 664 /etc/motion.conf
sudo chmod 755 /usr/bin/motion
sudo touch /tmp/motion.log
sudo chmod 775 /tmp/motion.log
```

Código-fonte 4: Permissões Motion - Fonte: O autor

Não seguimos as outras recomendações de configuração de `motion.conf` do tutorial de Christoph Buenger e DaSports (2014) porque elas se aplicam a versões do hardware, do sistema operacional e do Motion diferentes das em uso no protótipo final e/ou porque não se aplicam adequadamente às nossas necessidades. Lembrando que começamos com um arquivo `motion.conf` da compilação de LowflyerUK, que é diferente da apresentada no tutorial de Christoph Buenger e DaSports (2014).

As divergências no método usado para instalação do Motion também tornam outros ajustes não aplicáveis ou diferentes dos apresentados no tutorial. Para configurar o início

automático do Motion, usamos o rc.local, conforme a documentação oficial do Raspberry Pi em <https://www.raspberrypi.org/documentation/linux/usage/rc-local.md>, incluindo a linha abaixo no arquivo /etc/rc.local:

```
motion -c /etc/motion.conf &
```

Parametrizando o Motion

Com isso, o Motion deve estar instalado, configurado e funcionando automaticamente nas próximas inicializações do Raspberry Pi. Os ajustes finos relacionados a qualidade de imagem e parâmetros de detecção de movimento são feitos a seguir.

O arquivo de configuração do Motion, /etc/motion.conf, fornece uma infinidade de ajustes. Para o RPSS, as principais configurações alteradas foram:

```
width 640                #padrão: 352
height 480               #padrão: 288
framerate 12
daemon on #executa o Motion em plano de fundo, liberando o
terminal
mmalcam_secondary_buffer_upscale 1 #Define as dimensões da
imagem secundária como igual ao vídeo
lightswitch 0 #desabilita "ignorar lâmpada"
minimum_motion_frames 2 #Inicia gravação somente se houver
movimento em dois quadros consecutivos
pre_capture 0 #frames anteriores ao movimento a incluir no
vídeo
post_capture 4 #frames posteriores ao movimento a incluir no
vídeo
max_movie_time 600       #máximo 10min por arquivo
output_secondary_pictures on #salva foto do evento
output_both_pictures off #salva apenas uma foto em
um tamanho
locate_motion_mode on
```

```
locate_motion_style redcross
target_dir /var/www/html/Arquivos/ #endereço acessível pelo
servidor Web
on_movie_start "bash /etc/motion/scripts/start-email.sh"
on_movie_end "bash /etc/motion/scripts/stop-email.sh"
```

Código-fonte 5: Ajustes motion.conf - Fonte: O autor

O primeiro grupo de configurações listadas relaciona-se com a definição e taxa de quadros dos arquivos gravados. Como veremos posteriormente, a interface Web de usuário desenvolvida para o RPSS oferece a possibilidade de escolher outros valores para esses parâmetros.

As outras configurações correspondem a características de funcionamento do Motion que não serão ajustáveis pelo usuário.

3.4. Configurando o nginx

Utilizamos o nginx como servidor Web, para fornecer as páginas do sistema de vigilância exibidas via navegador. Utilizamos a documentação oficial do Raspberry Pi em <https://www.raspberrypi.org/documentation/remote-access/web-server/nginx.md> como referência para as ações abaixo:

Instalamos o nginx e o executamos pela primeira vez com os comandos:

```
sudo apt-get install nginx
sudo /etc/init.d/nginx start
```

Código-fonte 6: Instalação do nginx - Fonte: O autor

Instalamos o PHP para atender às requisições dinâmicas do usuário via interface Web, no lado do servidor

```
sudo apt-get install php5-fpm
```

Habilitamos o PHP no nginx, alterando o conteúdo do arquivo `/etc/nginx/sites-enabled/default`.

Primeiro, para permitir que a página inicial seja uma página PHP, adicionamos `index.php` à linha:

```
index index.html index.htm index.nginx-debian.html;
```

Depois, para habilitar o suporte a PHP propriamente dito, descomentamos as linhas (removendo o `#` no início de cada linha) abaixo, no mesmo arquivo:

```
location ~ \.php$ {  
    include snippets/fastcgi-php.conf;  
    fastcgi_pass unix:/var/run/php5-fpm.sock;  
}
```

Código-fonte 7: Configuração do nginx - Fonte: O autor

Salvamos o arquivo. Reiniciamos o nginx com o comando:

```
sudo /etc/init.d/nginx restart
```

Visto que as configurações padrão do php5-fpm para Raspberry Pi diferem das configurações padrão "oficiais" do PHP em outras plataformas (como o PHP para Windows) e o código para o RPSS foi escrito em um computador rodando nginx com PHP para Windows (sendo transferido para o Raspberry Pi após o desenvolvimento), é necessário equiparar as configurações do php5-fpm às do PHP padrão.

Para isso, excluimos o arquivo php.ini do php5-fpm, com o comando

```
sudo rm /etc/php5-fpm/php.ini
```

Essa operação corrigiu vários problemas apresentados em razão das configurações do php5-fpm do Raspberry Pi não coincidirem com os padrões originais do PHP.

3.5. Configurando o Tor

Para o acesso à interface Web do RPSS através de qualquer conexão com a Internet, utilizamos o Tor, instalado no Raspberry Pi com o comando abaixo:

```
sudo apt-get install tor
```

Após a instalação do Tor, configuramos um serviço oculto, que fornecerá o acesso à câmera via Web. As configurações do serviço oculto são baseada nas instruções oficiais em <https://www.torproject.org/docs/tor-hidden-service.html.en> e se resumem essencialmente a adicionar as linhas abaixo no arquivo `/etc/torrc`:

```
#Para a interface de usuário, terá endereço conhecido:
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:80
#Para o stream ao vivo, terá endereço aleatório e substituído
a cada reinicialização:
HiddenServiceDir /var/lib/tor/stream/
HiddenServicePort 80 127.0.0.1:8081
```

Código-fonte 8: Configuração do Tor - Fonte: O autor

3.6. Configurando o SSMTP

Para que o RPSS seja capaz de enviar e-mails sob determinadas condições, instalamos o SSMTP, serviço que fornece suporte a servidores SMTP para envio de e-mails, e algumas dependências:

```
sudo apt-get install mailutils  
sudo apt-get install ssmtp  
sudo apt-get install mpack
```

Código-fonte 9: Instalação do SSMTP - Fonte: O autor

Para facilitar a reconfiguração posterior por parte do usuário do endereço de e-mail do RPSS, configuramos o SSMTP de forma a operar com contas de e-mail do serviço Gmail, do Google. Posteriormente o usuário será capaz de alterar o endereço de e-mail, contanto que se utilize um endereço @gmail.com. Dessa forma, o usuário não precisa ter conhecimento das configurações SMTP para alterar o endereço, apenas possuir um endereço válido do Gmail.

Inicialmente, configuramos o SSMTP inserindo no arquivo /etc/ssmtp/ssmtp.conf as informações a seguir:

```
UseSTARTTLS=YES  
root=postmaster  
mailhub=smtp.gmail.com:587  
hostname=raspberrypi  
AuthUser=rpssifsc@gmail.com  
AuthPass=iue5whfawi32
```

Código-fonte 10: Configuração do SSMTP. Fonte - O autor

O endereço de e-mail (*AuthUser*) e senha (*AuthPass*) apresentados acima são de uma conta de usuário do Google criada especialmente para a tarefa de enviar e-mails provenientes desta unidade do RPSS.

É importante notar que, atualmente, o suporte a SMTP por parte do Gmail é opcional e vem desativado por padrão. Caso a conta de usuário seja alterada, pode ser necessário habilitar o suporte para "aplicações menos seguras" nas configurações de conta do Google escolhida, em <https://www.google.com/settings/security/lesssecureapps>

Criando Scripts de E-mail

Utilizamos o SSMTP para permitir que o Motion envie automaticamente e-mails quando iniciar ou cessar algum movimento visto pela câmera.

Os e-mails devem incluir o tipo de evento (início ou fim de movimento) e uma imagem ilustrando o evento iniciado ou finalizado.

Dado que o Motion permite a execução de apenas um comando por tipo de evento, e que o recurso de envio de e-mail deve ser opcional e configurável facilmente pela interface Web, o sistema de envio de e-mails foi feito através de shell scripts, que são scripts executados diretamente por um interpretador nativo do sistema operacional (neste caso, o *bash*).

Para o script chamado (dentro do arquivo `/etc/motion.conf`) ao iniciar um movimento, o código desenvolvido aparece abaixo:

```
#!/bin/bash
if [ $(</var/www/html/ini/start-email.txt) == 1 ]; then
```

```

cd /home/pi/RPSS/Arquivos
anexo=$(ls -t *.jpg | head -n1)
name=$(cat /var/www/html/ini/rpssname.txt)
email=$(cat /var/www/html/ini/email.txt)
printf "Ocorreu um movimento em RPSS $name" | mpack -a -s
"Ocorreu um movimento" -d /dev/stdin -m 0 -c $(file -b --
mime-type $anexo) "/home/pi/RPSS/Arquivos/$anexo" $email
fi

```

Código-fonte 11: Script de e-mail ao iniciar movimento - Fonte: O autor

A linha inicial indica o interpretador utilizado.

A linha `startstatus=$(cat /var/www/html/ini/start-email.txt)` lê o arquivo de configuração que indica se o envio de e-mail está ativo. A condicional abaixo dessa verifica se o envio está ativo, em caso positivo, executa o código necessário, em caso negativo, conclui o script sem efetuar tarefas adicionais.

Para identificar o arquivo a anexar no e-mail, a linha

```
anexo=$(ls -t *.jpg | head -n1)
```

salva na variável `$anexo` o nome de arquivo obtido ao listar todos os arquivos `*.jpg` em ordem de data a começar do mais novo e ler o primeiro item.

O comando `mpack` cria o e-mail com as informações coletadas e o envia através do SMTP configurado anteriormente.

Uma variação do mesmo script é usada para a condição de envio de e-mails ao cessar movimento, alterando-se apenas o arquivo de configuração avaliado e o conteúdo da mensagem.

Os scripts foram salvos em `/etc/motion/scripts`, e chamados através do Motion conforme os parâmetros em `motion.conf`:

```
on_movie_start "bash /etc/motion/scripts/start-email.sh"
on_movie_end "bash /etc/motion/scripts/stop-email.sh"
```

Código-fonte 12: Configurações e-mail em `motion.conf` - Fonte: O autor

3.7. Desenvolvendo a interface Web: HTML e PHP

A interface de usuário do sistema é desenvolvida com o uso de três linguagens típicas de aplicações Web: HTML para definir o conteúdo das páginas, CSS para formatar os aspectos visuais da página, como *layout*, cores e animações e PHP para executar todo o processamento referente à operação e configuração do sistema via interface Web.

Neste capítulo tratamos especificamente da estrutura funcional, composta pela formatação HTML e código PHP.

Quando o cliente faz a requisição de uma página PHP, a página requisitada é enviada para o interpretador PHP, em vez de direcionada diretamente ao cliente. O interpretador PHP processa a página e entrega o arquivo resultante. O código PHP nunca é enviado para o cliente, apenas o resultado gerado pelo mesmo.

O código PHP é inserido em meio às tags `<?php` e `?>`, e segue notação parecida com a da linguagem C. É possível inserir código HTML diretamente dentro de um arquivo PHP, desde que fora das tags que delimitam o código PHP propriamente dito. O interpretador executa tudo que encontrar dentro das tags PHP e imprime diretamente no arquivo enviado ao cliente o código HTML que encontrar fora das tags PHP, contanto que o código PHP executado atinja a linha onde esse código HTML é apresentado (por exemplo, se o código HTML ocorre durante uma cláusula `if`, esse código HTML só será impresso na página resultante se a condição `if` for cumprida).

Uma descrição geral de cada arquivo desenvolvido para o RPSS é apresentada no tópico a seguir.

Arquivos PHP principais

A interface Web é composta por vários arquivos executáveis PHP, com funções específicas.

O arquivo `/index.php` é a base da interface Web, e foi desenvolvido de forma a chamar dinamicamente os outros executáveis PHP no momento e local onde forem necessários. Abaixo são apresentadas as funcionalidades trazidas por cada um dos arquivos PHP que compõem a interface:

`/ajuda.php`: Fornece a página de ajuda. Esse executável é chamado por `/index.php`, processado e seu resultado adequadamente injetado na página Web (`/index.php`) quando o usuário chama a página de Ajuda. Embora esse seja um executável PHP, atualmente não há necessidade de nenhum conteúdo dinâmico para essa página, o que faz com que esse na verdade seja um arquivo HTML comum (embora haja a possibilidade de tornar o sistema de ajuda mais dinâmico se desejável).

`/arquivos.php`: Fornece a página de arquivos da interface do RPSS e executa todo o processamento relacionado. Esse executável é chamado por `index.php`, processado e seu resultado adequadamente injetado na página Web quando o usuário chama a página de Arquivos.

`/config.php`: Fornece a página de configuração do RPSS e executa parte do processamento relacionado. Esse executável é chamado por `/index.php`, processado e seu resultado adequadamente injetado na página Web quando o usuário chama a página de Configuração. Esse executável tem a importante tarefa de identificar qual o campo de configuração que sofreu edições e entregar as alterações ao arquivo PHP

interpretador (*parser*) correto para o processamento adequado da alteração.

`/login.php`: O RPSS é capaz de requerer a inserção de uma senha (configurável) para autenticação e acesso à interface de usuário. Este arquivo fornece a interface de *login* e executa a correta autenticação do usuário, caso o requerimento de senha (que é opcional e configurável na página de configuração) esteja ativo.

`/stream.php`: Fornece a página de *stream* ao vivo da interface do RPSS e executa todo o processamento relacionado. Esse executável é chamado por `index.php`, processado e seu resultado adequadamente injetado na página Web quando o usuário chama a página de *Stream* ou qualquer outra página de `index.php` com identificador inválido.

Interpretadores de configurações (parsers) PHP:

Uma pasta `/parsers/` contém os interpretadores que processam as alterações de configuração. Cada *parser* é chamado pelo botão "salvar" do formulário correspondente, conforme a seguir:

- `/parsers/email-parser.php`: Executa as alterações solicitadas nas configurações de envio de e-mails do RPSS;
- `/parsers/motion-parser.php`: Executa as alterações solicitadas nas configurações relacionadas ao uso do Motion;

- `/parsers/name-parser.php`: Executa as alterações solicitadas nas configurações de nome do RPSS, senha e exigência de senha para acesso ao RPSS;
- `/parsers/onion-parser.php`: Executa as alterações solicitadas nas configurações de do endereço Onion e sua chave privada.
- `/parsers/wifi-parser.php`: Executa as alterações solicitadas nas configurações de SSID e chave WPA2 para a rede WiFi.

Ainda na pasta `/parsers/`, os arquivos `/parser/newonion.php` e `/parser/newstreamonion.php`, chamados ao solicitar um novo endereço Onion na interface de configuração, têm a função de gerar uma nova chave Onion aleatória e seu devido endereço, automaticamente; o arquivo `/parser/erase.php` executa a operação de "apagar todos os arquivos" se solicitado e o arquivo `/parsers/reboot.php` prepara o RPSS para ser desligado ou reiniciado, quando solicitado através da interface de configuração.

Arquivos de configuração:

Os *parsers* alteram diversas configurações em vários locais do sistema operacional (por exemplo, `wifi-parser.php` altera as configurações da conexão WiFi diretamente em `/etc/wpa_supplicant/wpa_supplicant.conf`. Porém, diversas configurações referem-se diretamente à parâmetros lidos pela interface Web do RPSS em situações diversas. Esses parâmetros são editados e salvos pelos *parsers* em diferentes arquivos de texto salvos na pasta `/ini/` do servidor Web.

Como exemplo, um *hash* da senha que o usuário definiu para acesso à interface Web é salvo no arquivo `/ini/webhash.txt`

(a senha propriamente dita não é salva em lugar algum no sistema e não pode ser diretamente recuperada).

Arquivos de predefinição do Motion:

Na pasta `/motion/` do servidor Web situam-se os arquivos de configuração do Motion predefinidos para cada possibilidade de configuração do Motion disponível na interface de configuração. O arquivo de configuração atualmente em uso é copiado para `/etc/motion.conf` quando um novo modo de operação do Motion é selecionado nas configurações da interface Web.

Arquivos CSS:

Os arquivos CSS na pasta do servidor Web caracterizam o *layout* visual da interface em diferentes dispositivos e navegadores. Mais detalhes sobre o CSS e sua utilização no RPSS são apresentados no próximo capítulo:

3.8. Design Responsivo e compatibilidade

Todo o código anteriormente apresentado representa a estrutura funcional da interface Web. Os aspectos visuais da interface e como ela se apresenta em cada tipo de dispositivo suportado são fatores controlados primariamente por arquivos de folha de estilo CSS, anexas, mas externas ao código HTML/PHP.

O código CSS desenvolvido se utiliza especialmente de recursos que garantem a responsividade da interface, característica que permite à interface se adaptar dinamicamente ao tipo de dispositivo que está sendo utilizado para acesso. Todos os recursos do RPSS estão disponíveis na interface apresentada para qualquer dispositivo suportado, alterando-se apenas o *layout* de apresentação: o código fornecido é o mesmo para todos os dispositivos.

Três arquivos CSS definem o estilo visual de toda a interface em todas as condições suportadas:

/reset.css: Esse arquivo promove a reconfiguração de vários atributos visuais de um *website*, visando garantir que os padrões definidos no dispositivo do usuário (que depende, entre outras coisas, do navegador utilizado) sejam sobrepostos por valores conhecido, garantindo a consistência visual da interface (cujo visual é definido pelos arquivos CSS citados abaixo)

/w3.css: W3.CSS é um *framework* CSS, de uso livre, desenvolvido pela W3Schools. A função desse arquivo é fornecer um conjunto de ferramentas na forma de classes CSS pré-definidas e que atendem a tendências atuais em design de interfaces para a Web.

Para a interface do RPSS, W3.CSS é utilizado especialmente para fornecer as cores, formatos de botões e listas, efeitos de sombra e animações. As características restantes, incluindo os parâmetros de responsividade do layout (que o tornam automaticamente adaptável a diferentes resoluções de tela em diferentes dispositivos), são definidas pelo terceiro arquivo CSS:

/style.css: Diferente dos dois arquivos CSS anteriores (que são modelos prontos para uso), style.css corresponde à folha de estilos desenvolvido especificamente para a interface do RPSS.

3.9. Estrutura básica do CSS

Arquivos CSS são estruturados da seguinte forma:

```
html,body {
    margin: 0px;
    padding: 0px;
    width: 100%;
    height: 100%;
    background-color: #cacdcd;
    z-index: 0;
}

#userbar {
    position: fixed;
    top: 0px;
    right: 0px;
    width: 100%;
    height: 64px;
    z-index: 1;
}

div#sair-btn {
    display: inline;
```

```

position: absolute;
top: 7px;
right: 104px;
    z-index: 3;
}

input.w3-input, textarea{
    display: inline;
    width: 350px;
    background-color: #cacdc;
    margin-bottom: 6px;
    padding: 6px 8px 4px 8px;
}

```

Código-fonte 13: Estrutura do CSS - Fonte: O autor

Onde:

- Os itens a serem formatados com dadas características são identificados antes das chaves em cada bloco;
- As características a serem formatadas para os itens indicados são identificadas através de tags dentro das chaves, cada tag deve ter um valor e terminar com ponto-e-vírgula;
- Nomes iniciados por cerquilha representam identificadores únicos e nomes iniciados por ponto representam classes, enquanto os itens que não possuem nenhum dos dois indicadores representam tipos de objeto.

Como exemplo, o bloco CSS iniciado por:

```
input.w3-input, textarea{ }
```

formata tanto os objetos de tipo input contendo a classe w3-input quanto todos os objetos do tipo textarea com as características definidas dentro deste bloco. Do mesmo modo,

```
div#sair-btn { }
```

formata especificamente o objeto de tipo div com o identificador único sair-btn (que é o contêiner do botão "sair" da interface de usuário).

Para tornar o layout adaptável a diferentes formatos de tela (como mencionado anteriormente), dois recursos são utilizados:

- A possibilidade de definir valores como largura de um objeto em porcentagem da área total da tela (o que permite, por exemplo, definir uma largura como 100% para preencher toda a tela seja qual for a largura da mesma);
- A regra *@media*, que engloba (na forma de blocos CSS com novos valores) as alterações a serem executadas na interface de usuário quando a mídia de exibição corresponder à especificada na regra.

```
@media screen and (max-width: 801px){  
  
    /*0 código CSS específico para telas com  
    largura menor que 801 pixels vai aqui */  
  
    #userbar {  
        position: absolute;  
        height: 50px;  
        top: 48px;  
        z-index: 1;
```

```

    }

    div#sair-btn {
        right: 76px;
    }

```

Código-fonte 14: Regra @media - Fonte: O autor

No exemplo acima, algumas características da ID #userbar e do div com identificador único #sair-btn são sobrescritas por novos valores, mais adequados a telas de resolução horizontal menor que 801 pixels, e são utilizadas pelo navegador quando a interface é apresentada nesse tipo de tela (por exemplo, um telefone celular ou *tablet*)

É possível que o CSS contenha mais de uma regra @media, permitindo-se estabelecer "limites" para que as características de apresentação da interface se alterem de forma a cobrir uma faixa ampla de dispositivos diferentes. Em style.css do RPSS, quatro regras @media são utilizadas, todas baseadas na largura da área útil da tela no navegador do dispositivo que exibe a interface:

- Uma regra @media para telas de largura superior ou igual 1278 pixels: para a maioria dos monitores de computadores de mesa;
- Uma regra @media para telas de largura menor ou igual a 801 pixels: para telas de tablets, especialmente quando utilizados na vertical, e celulares quando utilizados na horizontal
- Uma regra @media para telas de largura menor ou igual a 400 pixels: para telas de celulares utilizados na vertical

- Uma regra *@media* para telas de largura menor ou igual a 320 pixels: para celulares com tela de resolução mais baixa. Esta última regra visa maximizar a compatibilidade, especialmente com aparelhos mais antigos e mais simples que os *smartphones* comuns atualmente.

Dentro da estrutura *@media*, só é necessário expressar blocos e tags onde ocorrem alterações em relação ao valor "padrão" (que é o utilizado quando nenhuma regra *@media* se aplica às características de exibição do navegador cliente, correspondendo aos blocos que não pertencem à nenhuma declaração *@media*).

As regras *@media* são cumulativas. Quando um dispositivo com tela de 320 pixels de largura é utilizado, tanto as regras para telas "até 320 pixels" quanto aquelas para telas "menor ou igual a 801 pixels" e "menor ou igual a 400 pixels" são interpretadas e aplicadas, em ordem de apresentação no arquivo CSS.

4. RESULTADOS

Os resultados do trabalho desenvolvido foram satisfatórios, e são apresentados a seguir:

4.1 Interface de usuário:

A interface Web desenvolvida apresentou excelente compatibilidade e desempenho compatível com as expectativas de um sistema operado via navegador.

Abaixo são apresentadas imagens comprovando a renderização da interface Web em diferentes dispositivos. São apresentadas fotos reais, em vez de capturas de tela, para demonstrar cada dispositivo realmente testado.

Computador Desktop ou laptop

Em computador desktop, conexão feita diretamente através do Tor utilizado o navegador nativo para clientes Tor no Windows (Tor Browser / Firefox ESR 45), e tamanho da janela mantido nos valores padrão (próximo de 1024x768), o resultado da renderização da interface Web é ilustrado a seguir (Figura 9 e Figura 10).

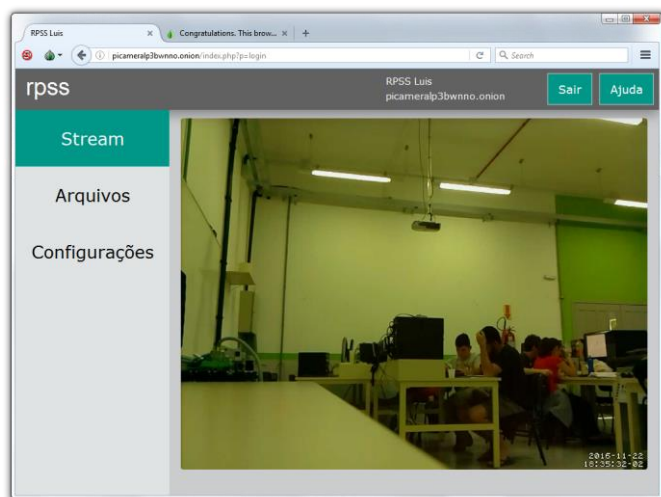


Figura 9: *Stream* ao vivo em computador desktop – Fonte: O autor

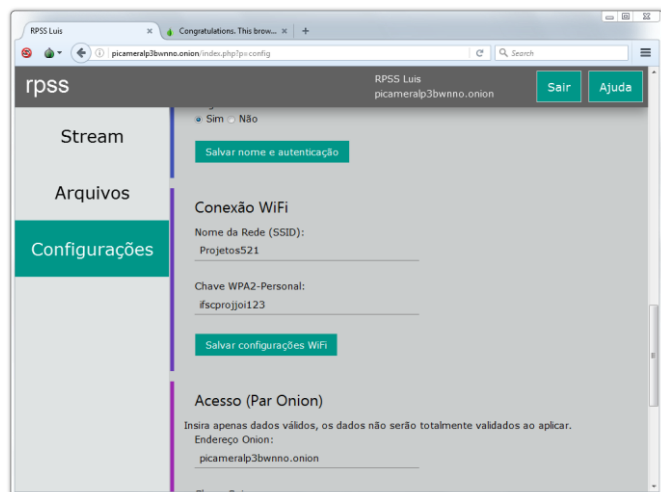
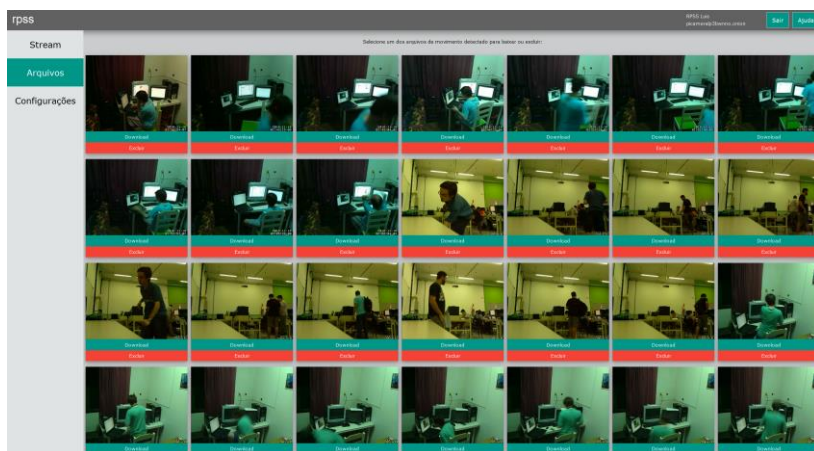


Figura 10: Configurações em computador desktop – Fonte: O autor

A lista de arquivos é projetada para apresentar tipicamente três colunas em telas de baixa resolução, mas ela faz uso total de resoluções maiores. Por exemplo, caso seja utilizada um monitor grande, com resolução de 2560x1440, a lista é capaz de aproveitar todo o espaço disponível apresentando mais colunas de miniaturas (Figura 11):



**Figura 11: Lista de arquivos em monitor grande de alta resolução –
Fonte: O autor**

Os resultados são similares em um computador portátil (*laptop*), contanto que a resolução da tela seja suficientemente alta para que a página seja renderizada com o formato apresentado acima e não um dos formatos para dispositivos menores, como veremos a seguir:

Laptop / Tablet PC (1280x800)

Em computador portátil (*laptop ou tablet PC*) de 12 polegadas com tela de resolução de 1280x800, a interface é

apresentada de forma muito similar, mas o tamanho do *stream* ao vivo e da lista de arquivos é adaptado para caber e/ou melhor aproveitar o tamanho da tela.

Utilizando um Tablet PC orientado na posição Retrato (Figura 12), a interface de usuário automaticamente se adapta à menor largura da tela (que passa a ter menos de 801 pixels), tornando a barra lateral uma nova barra fixa no topo da tela, e a barra fixa do topo uma barra móvel. A interface pode ser operada com uma caneta ativa Wacom da mesma forma que com um mouse. A conexão foi feita através do proxy Onion.TO com o navegador Firefox 48, e a página foi configurada para exibição em tela cheia:

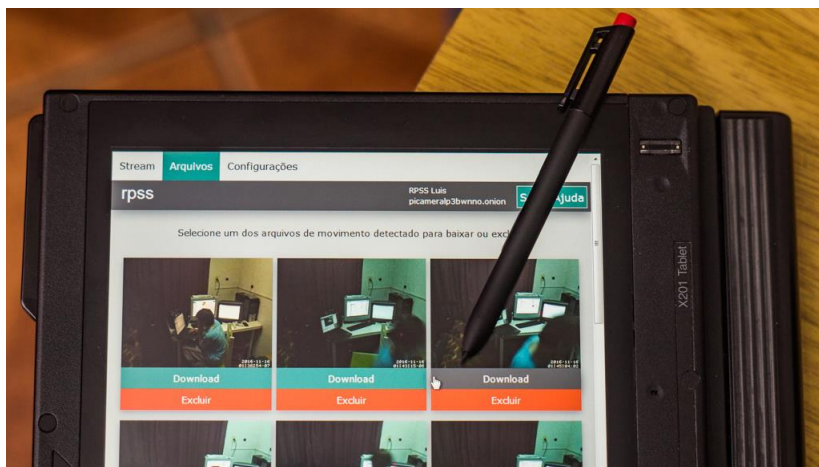


Figura 12: Interface de Arquivos em um Tablet PC – Fonte: O autor

Smartphone Android (Galaxy Gran Prime Duos, 960x540)

Em um smartphone Android com tela sensível ao toque (Samsung Galaxy Gran Prime Duos rodando Android 5.1), utilizando o navegador Chrome, a interface se adapta corretamente à resolução da tela de 960x540, tanto em formato retrato como paisagem. O *stream* ao vivo é exibido corretamente, assim como as páginas de arquivos e configurações, e a interface pode ser adequadamente operada com o toque dos dedos sem quaisquer limitações especiais (Figura 13):

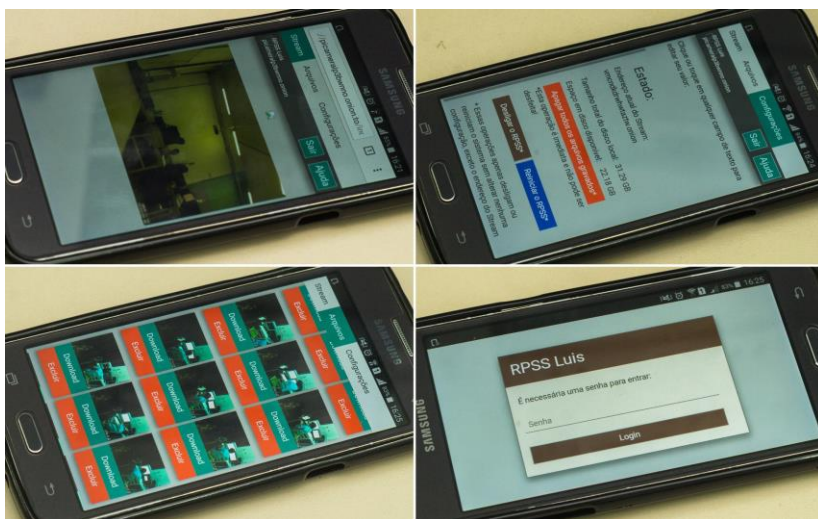


Figura 13: Interface Web no Android 5 – Fonte: O autor

Smartphone Nokia 808 PureView (640x360)

Em um smartphone Symbian de tela sensível ao toque lançado em 2012 (Nokia 808 PureView), utilizando o navegador Opera Mini 12, a interface se adapta corretamente à resolução da tela de 640x360, tanto em formato retrato como paisagem.

Provavelmente devido a limitações do navegador e/ou de desempenho do celular, o *stream* ao vivo carrega com lentidão nesse aparelho em alguns casos (Figura 14). Esse problema não ocorre no smartphone Android testado.

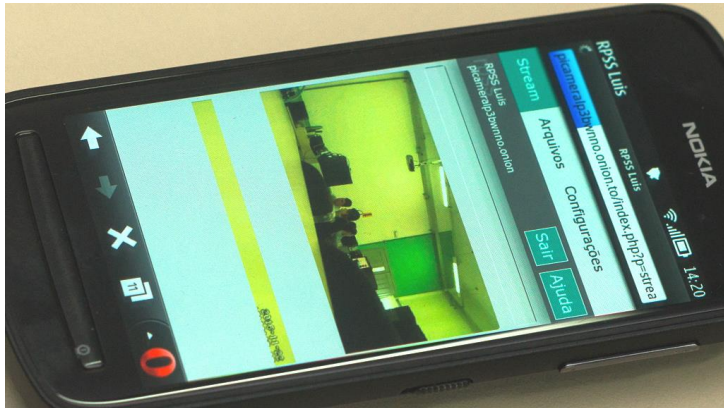


Figura 14: Falha eventual no *Stream* ao Vivo no Nokia 808 PureView
Fonte: O autor

As páginas de arquivos e configurações funcionam corretamente e é possível baixar e assistir os arquivos previamente gravados, em todas as resoluções suportadas pelo RPSS.

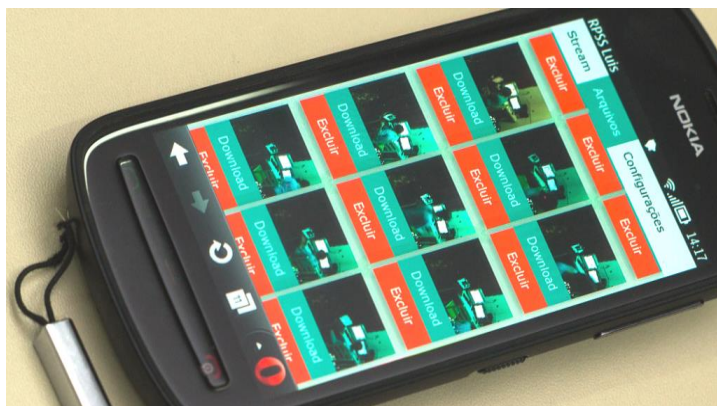


Figura 15: Lista de Arquivos no Nokia 808 PureView - Fonte: O autor

No navegador nativo do Nokia 808 PureView (Web 8.3), as interfaces de arquivos e configurações funcionam corretamente e com ótimo desempenho, mas não há nenhum suporte ao *stream* ao vivo, que não é exibido. Adicionalmente, nesse navegador, os efeitos de sombra da página, embora exibidos, apresentam intensidade incorreta.

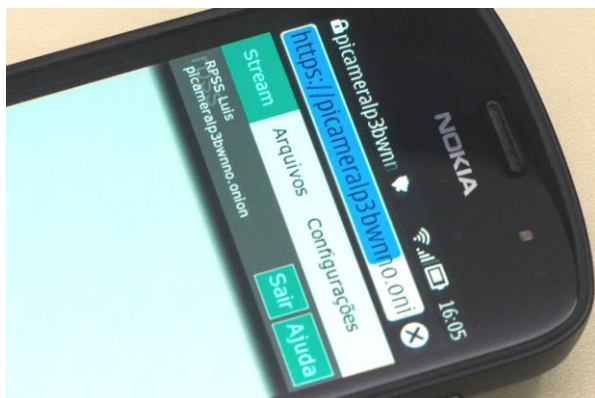


Figura 16: Sombras no navegador nativo no 808 PureView - Fonte: O autor

Smartphone Nokia 6210 Navigator (320x240)

Em um smartphone Symbian S60 clássico (sem tela sensível ao toque), lançado em 2008 (Nokia 6210 Navigator), utilizando o navegador Opera Mini 7.1 (versão nativa para Symbian, existe também uma versão *Java* para celulares comuns, não *smartphones*), as interfaces de arquivos e configurações funcionam corretamente, mas, por limitação do navegador, não há nenhum suporte ao *stream* ao vivo, que não é exibido (Figura 17).

No Opera Mini (que utiliza um *proxy* para pré-processar as páginas antes de enviar ao celular, simplificando suas características e diminuindo seu tamanho), a barra superior também não se mantém fixa no topo da tela, desaparecendo ao rolar a página para baixo. É possível alterar as configurações e baixar os arquivos, mas este *smartphone* não inclui os *codecs*

necessários para abrir os vídeos gravados pelo RPSS em nenhuma das resoluções.



Figura 17: Interface Web no smartphone Nokia 6210 Navigator
Fonte: O autor

Celular Nokia 3120 Classic (320x240)

Em um celular Nokia 3120 classic lançado em 2008 (sistema Nokia S40, não é um *smartphone*), executando o Opera Mini 4.5 e Opera Mini 7.1 (versões *Java* para celulares comuns de menor capacidade), a página do *stream* ao vivo nunca termina o carregamento nem é apresentada mensagem de erro. Porém, ao endereçar diretamente outra página do RPSS, como a de Arquivos ou de Configurações, a página abre e é possível navegar adequadamente por ela, sendo possível alterar configurações, acessar a Ajuda, excluir e baixar arquivos, embora este celular também não incluía os *codecs* necessários para abrir os vídeos gravados pelo RPSS em nenhuma das

resoluções. A Figura 18 apresenta a interface como exibida no Opera Mini 7.1 (retrato) e no Opera Mini 4.5 (paisagem)



Figura 18: Interface Web no Opera Mini em celular simples – Fonte: O autor

Limitações gerais de Compatibilidade

Foram características percebidas na interface Web desenvolvida:

A tela de *login* é exibida com uma pequena falha de renderização em todos os navegadores testados nos smartphones Symbian e no celular S40, onde uma barra inesperada de cor clara aparece no topo da caixa de *login* (Figura 19). Esse erro não ocorre no Chrome para *smartphones* Android nem no Firefox, Internet Explorer ou Tor Browser para Windows. Novamente, pode-se também notar a sombra excessiva no navegador nativo no Nokia 808 PureView, no primeiro quadro da figura.

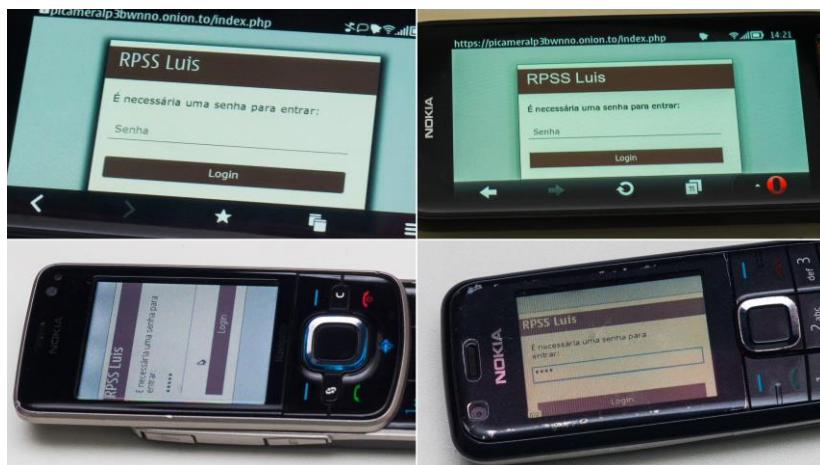


Figura 19: Erro de renderização no login - Fonte: O autor

O protocolo de exibição utilizado pelo *Stream* ao vivo não é suportado no navegador Internet Explorer mais atual (11) nem, por consequência, em *smartphones* Windows Phone ou Windows Mobile 10. Esses *smartphones* ainda podem ser utilizados para alterar configurações ou baixar, excluir ou assistir os arquivos previamente gravados acessíveis pela interface. As miniaturas são exibidas corretamente (Figura 20 e Figura 21).

Por fim, em sistemas iOS, o sistema funciona sem restrições e com desempenho adequado, de forma semelhante à apresentada por *smartphones* Android.

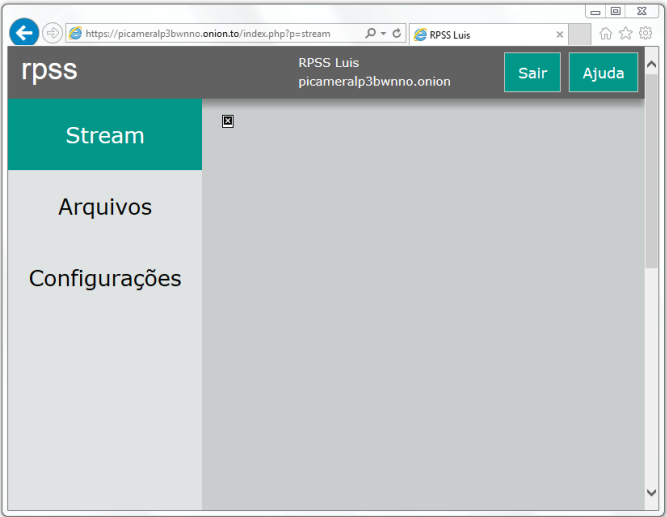


Figura 20: Falta de *stream* no IE11 – Fonte: O autor.

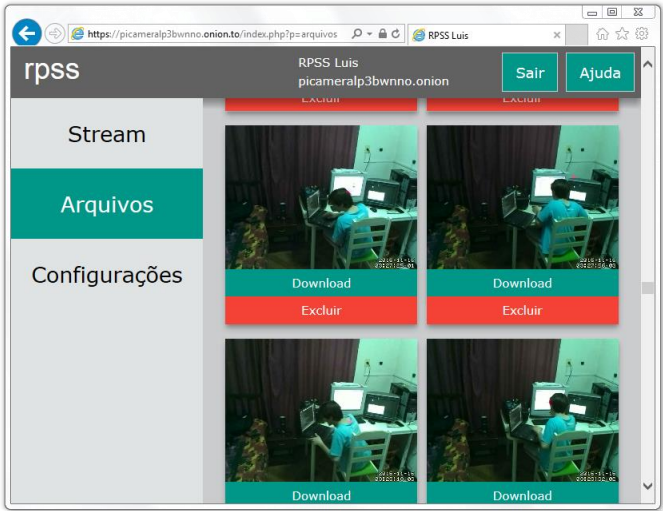


Figura 21: Arquivos no IE11 – Fonte: O autor.

Recursos adicionais da Interface Web

Outras características da Interface Web, não relacionadas à compatibilidade:

Uma caixa de informação verde é exibida no topo da página de configuração toda vez que uma configuração é alterada com sucesso, informando a configuração alterada e seu novo valor, (Figura 22) exceto quando se trata de uma senha, (nesse caso a caixa apenas informa que a configuração foi alterada com sucesso). Uma caixa de informação verde também é exibida na página de Arquivos quando um arquivo é excluído com sucesso.

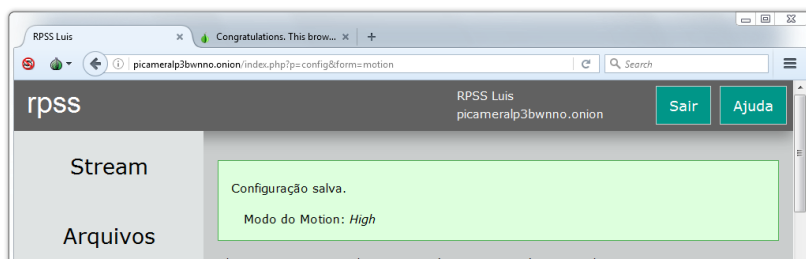


Figura 22: Sucesso na configuração - Fonte: O autor

Caso ocorra alguma falha, é exibida uma caixa de informação vermelha, por exemplo, caso se peça a exclusão de um arquivo já inexistente (Figura 23)

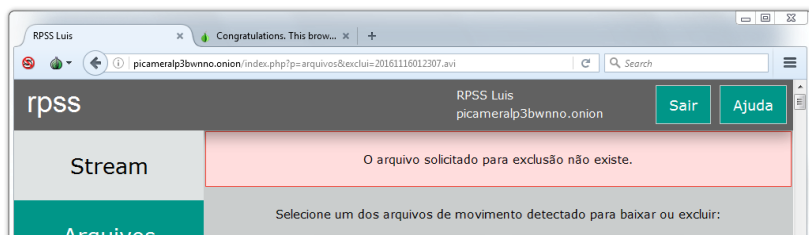


Figura 23: Aviso de erro - Fonte: O autor

O botão "Sair" é oculto da interface caso a exigência de senha ao entrar seja desativada nas configurações (Figura 24), visto que não há para onde "sair" quando o acesso de entrada está definido como livre (isto é, protegido apenas pela confidencialidade do endereço Onion).

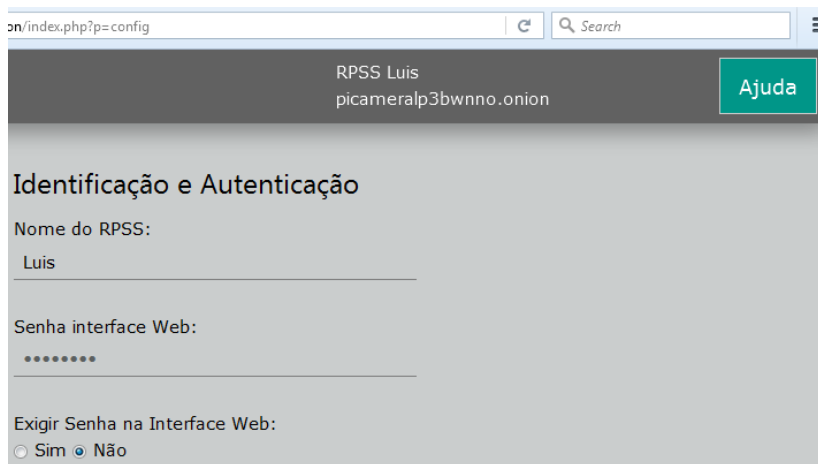


Figura 24: Sem botão Sair ao desativar login - Fonte: O autor

Isso soluciona o problema em primeira instância, mas estudos mais aprofundados são necessários para garantir a segurança das permissões de acesso aos arquivos que sofreram esse tipo de modificação.

O balanço de branco da câmera

Um problema, não crítico e não solucionado, foi apresentado nos arquivos de vídeo gravados pela câmera e o *stream* ao vivo: sob várias condições de iluminação no laboratório 521 do IF-SC, a imagem apresentada apresentou desvio de cor intenso para o amarelo ou, às vezes, verde (Figura 26).

O problema foi diagnosticado como um erro de *firmware* da GPU do Raspberry Pi, que faz o processamento das imagens antes de enviá-las ao Motion na CPU ARM para detecção de movimento, codificação e gravação: a GPU aparentemente erra ao definir o correto balanço de branco da imagem em determinadas condições, e a mesma não fornece o suporte a alteração desse parâmetro de forma suficientemente abrangente para solucionar o problema.



Figura 26: Balanço de Branco Pi Camera – Fonte: O autor.

4.3. Problemas de hardware encontrados:

Um dos primeiros problemas encontrados no desenvolvimento do primeiro protótipo ocorreu imediatamente ao ligar o Raspberry Pi (com o cartão de memória com o sistema operacional já instalado) pela primeira vez.

O sistema não inicializava corretamente, a rede com fio e os dispositivos USB conectados (teclado e mouse) não funcionavam, impedindo a operação e configuração da placa.

Descobrimos que o problema era causado por um cabo USB de baixa qualidade sendo utilizado para transmitir a alimentação da fonte de energia para o Raspberry Pi: a fonte fornece 5,1 V, mas apenas 3,8 V estavam chegando ao Raspberry Pi quando ligado. O Raspberry Pi requer entre 4,75 V e 5,25 V para operar corretamente.

Para detectar o problema, foi necessário o uso de um multímetro configurado para tensão DC, medindo a tensão entre os pontos TP1 (+5V) e TP2 (GND) na placa, conforme sugerido por mahjongg, um dos moderadores do fórum oficial do Raspberry Pi em <https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=58151>

Este problema seria diagnosticado imediatamente se ocorresse no segundo protótipo (Raspberry Pi 2), porque esta já inclui o hardware e software necessários para identificar este tipo de problema, apagando o LED *Power* e indicando através de um ícone na tela a situação de subtensão quando esta ocorre, conforme explicado por um dos engenheiros da Raspberry Foundation no fórum oficial do Raspberry Pi em <https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=82373>

4.4. Resultados de longo prazo do primeiro protótipo:

O primeiro protótipo foi utilizado por vários meses, apenas com os recursos básicos de detecção de movimento e acesso via rede local implementados, no laboratório 521 do IF-SC. Uma interface de usuário via Web bastante simplificada foi desenvolvida para esse protótipo inicial, sendo acessível apenas via rede local e capaz de exibir o *stream* ao vivo e uma lista de arquivos, em uma página HTML simples (Figura 27).



Figura 27: Interface Web Primeiro Protótipo - Fonte: O autor

As configurações de resolução, taxa de quadros e sensibilidade de detecção de movimento escolhidas para esse primeiro protótipo foram:

- Resolução de 640x480
- Taxa de quadros de 2 FPS

As configurações foram escolhidas por limitação de capacidade de processamento da CPU do Raspberry Pi, que nessas configurações já se mantinha constantemente por volta de 80% a 98% de utilização.

Nessas condições, o protótipo foi capaz de funcionar adequadamente por um longo período, gravando corretamente as cenas em que ocorrem movimento e ignorando a grande maioria das cenas sem movimento. Com o nível de atividade do laboratório em questão, o sistema manteve-se gravando por volta de 0,5 GB de dados por dia, o que permite ao cartão de memória utilizado (com espaço disponível de pelo menos 25GB) o uso contínuo por pelo menos 50 dias antes que fosse necessário excluir as gravações antigas.

Embora o cartão de memória utilizado no primeiro protótipo seja um cartão recomendado para uso em telefones celulares (SanDisk Ultra), não houve nenhuma falha causada por uso contínuo por vários meses executando pequenas gravações constantemente.

Este protótipo foi posteriormente atualizado com a nova interface Web (a mesma desenvolvida para o segundo protótipo, embora com alguns recursos removidos ou adaptados), e voltou a ser posto em operação no mesmo laboratório, dessa vez com capacidade de acesso externo via Internet, que funcionou adequadamente.

4.5. Resultados do segundo protótipo:

O segundo protótipo foi testado conectando-se à internet tanto por rede doméstica quanto institucional, e o acesso à interface Web e o *stream* ao vivo funcionou corretamente em todos os testes. Porém, outro recurso apresentou limitações:

Envio de e-mails em rede com restrições de acesso

O envio de e-mails via SSMTP (com acesso ao Gmail) conforme configurado para o RPSS funciona corretamente em conexões domésticas à Internet (Figura 28), mas não funciona dentro de determinadas redes corporativas ou institucionais (como a do próprio Campus Joinville do IF-SC).

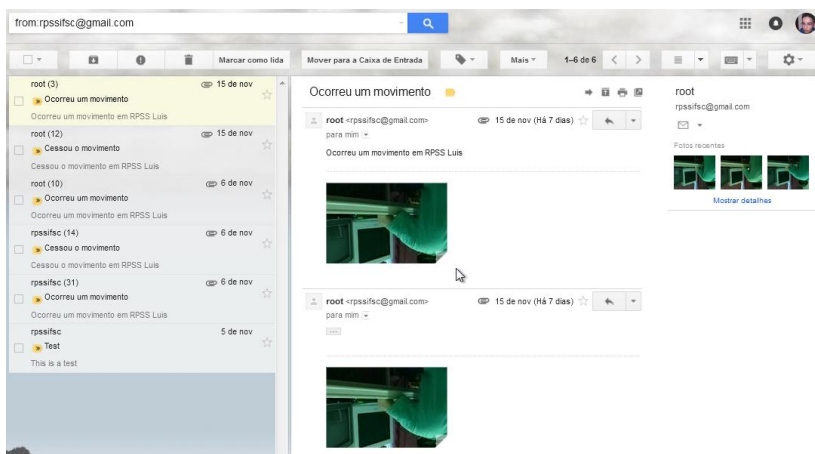


Figura 28: E-mails enviados pelo RPSS em rede doméstica
Fonte: O autor

A solução de rotear os e-mails via Tor, tal qual a interface Web, foi proposta, mas problemas na implementação impediram que esta fosse concluída com sucesso até a data de entrega

deste trabalho, ocorreram problemas ao tentar seguir o passo-a-passo da documentação oficial do Tor para configuração de um *proxy* Tor transparente (através do qual o SSMTP se conectaria ao Gmail, visto que o SSMTP não oferece suporte ao *proxy* SOCKS oferecido pelo Tor para aplicações suportadas).

5. TRABALHOS FUTUROS

5.1. Instalação de atuadores

Optamos por não executar a instalação de um sistema de movimentação da câmera com atuadores, especialmente em razão da complexidade envolvida no desenvolvimento dessa tarefa e o tempo limitado para sua execução. Para essa tarefa, seriam necessários:

- Um gerador PWM independente, como o Adafruit 12-Bit PWM Driver (Figura 29), porque o Raspberry Pi possui um único canal PWM capaz de comandar um único *RC Servo*;
- Um meio compacto, confiável e embarcado de alimentar os atuadores, visto que o próprio Raspberry Pi não é capaz de fornecer a corrente necessária para a correta e segura operação dos mesmos;
- Conhecimentos em operação de dispositivo com barramento i2c através dos pinos de GPIO do Raspberry Pi (se o gerador PWM Adafruit especificado anteriormente fosse o utilizado);
- Possível necessidade de adaptação direta no código-fonte do Motion para possibilitar que o software comandasse os atuadores automaticamente;
- Necessidade de adaptação no desenho 3D do gabinete do sistema para permitir a instalação e operação dos atuadores e as alterações necessárias no sistema de alimentação;

- Necessidade de inclusão no código PHP da Interface Web toda uma plataforma capaz de operar esse recurso adicional.

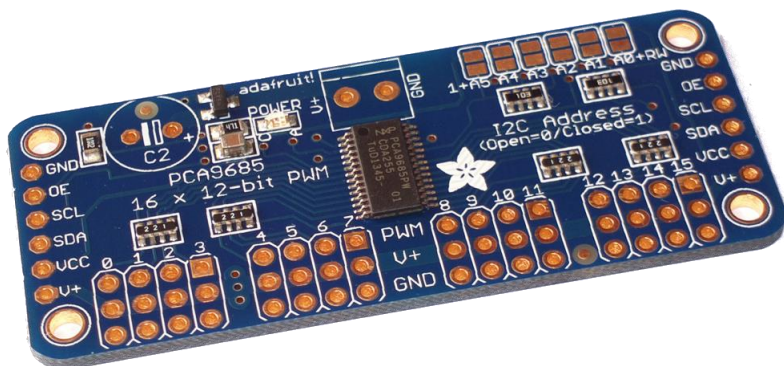


Figura 29: Gerador PWM i2c Adafruit – Fonte: O autor.

5.2. Desenvolvimento de uma interface de configuração emergencial

Outro recurso adicional sugerido para implementação futura é uma interface de configuração emergencial, permitindo a reconfiguração de parâmetros como as credenciais da rede WiFi ou par endereço/chave Onion de acesso ao sistema via Web, caso esses tenham sido atualizados pelo usuário com dados inválidos ou incorretos. A interface também poderia disponibilizar acesso facilitado a *logs* de erros para solução de possíveis problemas.

Essa interface seria acessível através da conexão de rede cabeada do Raspberry Pi, sendo prontamente acessível ao conectar diretamente (ponto-a-ponto) o Raspberry Pi a um computador, tornando-se disponível através de um endereço fixo conhecido (através de um servidor DHCP instalado no próprio

Raspberry Pi, operando de forma similar ao painel de controle de um roteador doméstico.

No estado atual, é possível reconfigurar o sistema em situações emergenciais, via rede local ao se conectar o Raspberry Pi em um roteador doméstico e acessar sua interface Web diretamente através do seu endereço IP. Porém, esse método requer o conhecimento ou descoberta manual do endereço IP fornecido pelo roteador à placa, não sendo um meio tão amigável quanto a solução proposta como trabalho futuro.

A proposta envolve a necessidade de diversos novos ajustes e configurações no sistema de rede, DHCP e de *firewall* do sistema operacional do Raspberry Pi, além de alterações nas configurações do servidor Web nginx (criando uma nova página Web adicional, acessível através da rede ponto-a-ponto cabeada), criação da interface emergencial através de código PHP, HTML e CSS, dentre outros, mas é capaz de trazer melhorias consideráveis de usabilidade do sistema em determinados cenários.

5.3. Alterações no sistema de envio de e-mails

O sistema de envio de e-mails implementado não funciona em redes onde conexões SMTP são bloqueadas, como ocorreu em experimento nos laboratórios do IF-SC Campus Joinville.

Uma possível solução seria rotear os e-mails via Tor, tal qual a interface Web, através da configuração de um proxy Tor transparente (através do qual o SSMTP se conectaria ao Gmail, visto que o SSMTP não oferece suporte ao *proxy SOCKS*

utilizado por padrão por aplicações que oferecem suporte ao Tor nativamente)

Outra possibilidade é a de substituir completamente o sistema de e-mails por um sistema independente de provedores externos (como o Gmail), enviando os e-mails diretamente ao endereço de destino utilizando um servidor de e-mail próprio, conectado nativamente via Tor, possivelmente utilizando o mesmo endereço Onion da Interface Web para identificação de destinatário. Essa proposta apresenta seus próprios desafios, entretanto, e sua viabilidade precisa ser mais cautelosamente aferida.

Ainda relacionado ao sistema de e-mails, pode-se propor a adição de mais opções de configuração, permitindo, por exemplo, o estabelecimento de um calendário para habilitar ou desabilitar automaticamente o envio de e-mails em determinadas datas e horários, ou o suporte, através da interface Web, a scripts mais complexos para definição das condições de envio de e-mail, tornando o sistema mais versátil, autônomo e melhor configurável pelo usuário.

5.4. Conexão à rede celular e envio de SMS

Conforme mencionado anteriormente, é tecnicamente possível fazer com que o sistema opere conectado à internet diretamente através de um modem *HSDPA/3,5G* ou mesmo *LTE/4G*, o que permitiria que, em caso de sinistro, o sistema enviasse imediatamente e-mail apresentando a cena mesmo se a conexão WiFi fosse interrompida.

Outra possibilidade adicional trazida pela adição de um meio de acesso à rede celular é a possibilidade de implementar, além de e-mail, o envio de mensagens de texto (SMS), criando a

possibilidade de envio avisos diversos através de um meio potencialmente mais acessível que e-mail.

5.5 Desenvolvimento de um sistema automático de obtenção de bridges Tor

Em redes onde até mesmo o acesso direto ao Tor é bloqueado (possível em algumas redes corporativas ou institucionais), pode ser necessário o uso de *bridges* (pontes de acesso à rede Tor, que podem ser fornecidas por voluntários e obtidas através de requisição - respondida automaticamente - ao endereço bridges@torproject.org), incluindo também a possibilidade de uso de transportes plugáveis (que visam tornar inviável a detecção da conexão como uma conexão ao Tor, aumentando a possibilidade de sucesso na conexão)

Esses *bridges* não podem ser permanentemente definidos no arquivo `/etc/tor/torrc`, porque não é possível garantir que um *bridge* obtido publicamente estará permanentemente disponível, logo, para a configuração de *bridges*, é necessária a configuração de um sistema que obtém regularmente uma lista de *bridges* válidos e atualiza o arquivo de configuração do Tor.

Isso pode ser dificultado em razão do possível bloqueio de conexões SMTP nas mesmas redes onde houver a necessidade de obtenção da lista de *bridges* por e-mail, tornando necessária a utilização de algum meio alternativo para obtenção automatizada da lista de bridges.

CONCLUSÃO

A proposta do trabalho, a de desenvolver um Sistema de Vigilância online monitorado e configurado remotamente, pôde ser cumprida com sucesso, inclusive se utilizando de táticas inovadoras para esse tipo de aplicação:

O sistema desenvolvido se utiliza especialmente de tecnologias de código-fonte aberto, se assemelhando a projetos comumente desenvolvidos por integrantes do movimento *Maker*,

O hardware escolhido, especialmente a placa Raspberry Pi, se mostrou adequado aos requisitos do projeto, assim como o gabinete construído através de impressão 3D.

A interface de usuário via Web foi completamente desenvolvida especialmente para esse projeto, visando operação fácil, bom desempenho e especialmente alta compatibilidade com dispositivos móveis, e apresentou bons resultados em testes reais de uso.

Foi utilizado um meio inovador para permitir o acesso ao dispositivo via Internet, utilizando serviços ocultos do Tor, com excelentes resultados permitindo o acesso mesmo quando o sistema é instalado dentro de uma rede corporativa ou institucional.

Por fim, algumas melhorias futuras foram propostas, e o detalhamento a respeito da viabilidade e método sugerido de implementação para cada uma foi apresentado, de forma a facilitar estudos futuros com o intuito de aprimorar esse projeto ou adaptá-lo em todo ou em parte à novas aplicações.

REFERÊNCIAS

Blem, E., Menon, J., & Sankaralingam, K. (2013). **Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures**. Paper presented at the Proceedings - International Symposium on High-Performance Computer Architecture, 1-12.

Buenger, Christoph., DaSports. **Raspberry Pi as low-cost HD surveillance camera** <<http://www.codeproject.com/Articles/665518/Raspberry-Pi-as-low-cost-HD-surveillance-camera>> 12 de maio de 2014. Acesso em 23/11/2016

Cucchiara, R., Grana, C., Piccardi, M., & Prati, A. (2003). **Detecting moving objects, ghosts, and shadows in video streams**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(10), 1337-1342.

Edwards, C. (2013). **Not-so-humble raspberry pi gets big ideas**. Engineering and Technology, 8(3), 30-33.

Hu, W., Tan, T., Wang, L., & Maybank, S. (2004). **A survey on visual surveillance of object motion and behaviors**. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 34(3), 334-352.

Kuznetsov, S., & Paulos, E. (2010). **Rise of the expert amateur: DIY projects, communities, and cultures**. Paper presented at the NordiCHI 2010: Extending Boundaries - Proceedings of the 6th Nordic Conference on Human-Computer Interaction, 295-304.

Lakhani, K. R., & Von Hippel, E. (2003). **How open source software works: "free" user-to-user assistance**. Research Policy, 32(6), 923-943.

Maddalena, L., & Petrosino, A. (2008). **A self-organizing approach to background subtraction for visual surveillance applications**. IEEE Transactions on Image Processing, 17(7), 1168-1177.

Menezes, V., Patchava, V., & Gupta, M. S. D. (2016). **Surveillance and monitoring system using raspberry pi and SimpleCV**. Paper presented at the Proceedings of the 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015, 1276-1278.

Munos, B. (2006). **Can open-source R&D reinvigorate drug research?** Nature Reviews Drug Discovery, 5(9), 723-729.

Nascimento, S., & Pólvara, A. (2016). **Maker cultures and the prospects for technological action**. Science and Engineering Ethics, , 1-20.

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2017). **Empirical studies on the maker movement, a promising approach to learning: A literature review**. Entertainment Computing, 18, 57-78.

Pingdom.com (2012). **75% of top 10k websites served by open source software**. <<http://royal.pingdom.com/2012/05/22/75-percent-top-10k-websites-served-by-open-source-software/>> 23 de Maio de 2012. Acesso em 23/11/2016

RAULINO, Mario F. **RaspberryPi e RFID no monitoramento de atividades de natação TCC** (graduação) - Instituto Federal de Santa Catarina. CST Sistemas de Telecomunicações, São José, 2013

Sonaglio, S. **Deteção automática de movimento através de sinais de vídeo : estudo e implementação de um sistema**. TCC (graduação) - Instituto Federal de Santa Catarina. CST Sistemas de Telecomunicações, São José, 2009

Valpreda, F. (2015). **3D printing awareness the future of making things**. Paper presented at the Proceedings of SPIE - the International Society for Optical Engineering, , 9398

VIEIRA, Ramon A. **Análise de aspectos relativos à QoS de um dispositivo DVR.** TCC (graduação) - Instituto Federal de Santa Catarina. CST Sistemas de Telecomunicações, São José, 2009

von Busch, O. (2013). **Molecular management: Protocols in the maker culture.** Creative Industries Journal, 5(1-2), 55-68.

Von Hippel, E., & Von Krogh, G. (2003). **Open source software and the "private-collective" innovation model: Issues for organization science.** Organization Science, 14(2), 209-223+225.

Wolf, W., Ozer, B., & Lv, T. (2002). **Smart cameras as embedded systems.** Computer, 35(9), 48-53.

Yilmaz, A., Javed, O., & Shah, M. (2006). **Object tracking: A survey.** ACM Computing Surveys, 38(4)

BIBLIOGRAFIA COMPLEMENTAR

Shell Scripting Tutorial <<http://www.shellscript.sh/>>
Acesso em 23/11/2016

Configuring Hidden Services for Tor <<https://www.torproject.org/docs/tor-hidden-service.html.en>> Acesso em 23/11/2016

PHP Documentation <<http://php.net/docs.php>> Acesso em 23/11/2016

Raspberry Pi - Downloads <<https://www.raspberrypi.org/downloads/>> Acesso em 23/11/2016

Raspberry Pi Documentation <<https://www.raspberrypi.org/documentation/>> Acesso em 23/11/2016

Tor: Overview <<https://www.torproject.org/about/overview.html.en>> Acesso em 23/11/2016

W3Schools.com CSS Tutorial <<http://www.w3schools.com/css/default.asp>> Acesso em 23/11/2016

W3Schools.com HTML5 Tutorial <<http://www.w3schools.com/html/default.asp>> Acesso em 23/11/2016

W3Schools.com PHP 5 Tutorial <<http://www.w3schools.com/php/default.asp>> Acesso em 23/11/2016

W3Schools.com W3.CSS Tutorial <<http://www.w3schools.com/w3css/default.asp>> Acesso em 23/11/2016

Who uses Tor? <<https://www.torproject.org/about/torusers.html.en>> Acesso em 23/11/2016

GLOSSÁRIO

Verbetes mencionados em *itálico* na descrição de um verboete possuem sua própria entrada no glossário:

Android: Sistema Operacional utilizado na maioria dos smartphones e tablets da atualidade (a partir de 2008), baseado no *Kernel Linux*.

Caneta Wacom: Caneta utilizada em digitalizadoras Wacom, com detecção de proximidade e sensibilidade variável à pressão, utilizadas em mesas digitalizadoras e alguns *tablets*.

Chipset: Circuito integrado contendo meios de comunicação entre o processador, memória, entre outros, de um dispositivo. Pode incluir o próprio processador, vide SoC.

Codec: Dispositivo ou *software* com o objetivo de decodificar um sinal ou protocolo, especialmente audiovisual.

CSS: Folhas de estilo em cascata, definem os aspectos visuais de uma página Web.

Dashcam: Câmera tipicamente utilizada no pára-brisa de veículos, com o objetivo de registrar eventos que ocorram em frente ao veículo.

DHCP: Protocolo de configuração dinâmica do Host, utilizado para conceder dinamicamente *endereços IP* a diferentes clientes.

DNS: Sistema que traduz endereços amigáveis da Internet em *endereços IP*, permitindo a localização e conexão dos clientes ao servidor requisitado.

Driver de dispositivo: *Software* com o intuito de operar ou estabelecer comunicação com um dispositivo de *hardware*.

EFF: Fundação da Fronteira Eletrônica, organização sem fins lucrativos com o objetivo de proteger os direitos de liberdade de expressão e privacidade na Internet.

Endereço IP: Endereço fornecido para identificação de dispositivos que se comunicam via Internet, pode ser fornecido dinamicamente através de um servidor *DHCP*.

Firewall: Sistema cuja função é proteger um dispositivo ou rede conectada à internet contra acesso não autorizado.

Firmware: software de baixo nível embarcado em um hardware, com a função de comandar diretamente esse hardware e permitir comunicação do mesmo com o mundo externo.

Fork: Versão alternativa de um *software* open source, tipicamente mantido por diferentes líderes de projeto, criada para atender às necessidades de um subgrupo específico de usuários do software original.

Framework: Conjunto classes ou bibliotecas para uma determinada linguagem de programação, visando simplificar o desenvolvimento de software ao fornecer funcionalidades pré-desenvolvidas.

Hardware: Conjunto de componentes físicos, tipicamente eletrônicos, capaz de executar *software* em um sistema computacional.

Hash: Algoritmo capaz de codificar uma informação de forma irreversível. Comumente utilizado para armazenamento indireto de senhas (obtendo-se o *hash* da senha digitada e comparando com o *hash* salvo, caso não).

HTML: Linguagem de marcação de hipertexto, utilizada para escrita de páginas da Web.

HSDPA/3.5G: Tecnologia que aprimora a Terceira Geração (3G) de redes celulares, permite uma expressivamente maior taxa de transferência de download em comparação à geração anterior.

Java: Linguagem de programação cujo código pode ser executado por diversos dispositivos embarcados, especialmente telefones celulares mais simples (não smartphones).

Kernel: Núcleo de um sistema operacional, inicializando o sistema e possuindo controle total sobre o mesmo durante todo o tempo de execução, por exemplo, o Kernel Linux, utilizado em diversos *sistemas operacionais*.

LTE/4G: Tecnologia de transmissão de dados via rede celular de quarta geração (4G), permite uma expressivamente maior taxa de transferência de download e upload em comparação à geração anterior.

Log: Registro automatizado de atividades executadas por um software, pode ser utilizado para identificação de erros.

MAC: Endereço físico de um dispositivo em uma infraestrutura de rede, seja cabeada ou sem fios.

Onion: Protocolo de comunicação utilizado pela rede Tor, capaz de proteger a identidade tanto de seus clientes quanto seus servidores.

Parser: Interpretador ou avaliador, software ou script com o objetivo de interpretar ou avaliar uma informação decidindo como atuar sobre ela.

Power bank: Dispositivo portátil de recarga elétrica para outros dispositivos, como *smartphones* e *tablets*. Utiliza uma fonte de energia externa para alimentar uma bateria que, quando carregada, pode ser utilizada para recarregar outro dispositivo.

Proxy: *Software* com o objetivo de atuar como intermediário em uma conexão de rede entre um cliente e um servidor.

PWM: Modulação por Largura de Pulso, técnica de modulação digital de sinal elétrico, capaz de codificar uma informação a ser transmitida, ou limitar a potência média de um sinal.

RC Servo: Atuador comum em aplicações de rádio controle, capaz de posicionar-se precisamente em uma faixa de 180 graus e/ou produzir oscilações precisas na mesma faixa.

S40: Interface de usuário do *sistema operacional* da Nokia para telefones celulares simples, que não executam o sistema operacional *Symbian*.

S60: Interface de usuário utilizada em smartphones *Symbian* mais antigos, como o Nokia 6210 Navigator.

Sistema Operacional: *Software* que gerencia toda a plataforma de *hardware* e software em um determinado dispositivo.

Smartphone: Telefone celular com quantidade considerável de recursos de computador pessoal, como multitarefa e capacidade de instalação e execução nativa de *softwares*.

SoC: Sistema em um chip, circuito integrado que contém, sozinho, todos os principais componentes necessários para o funcionamento de um sistema computacional, tipicamente com exceção da memória RAM e de armazenamento.

SOCKS: Protocolo de Internet utilizado em comunicações via *proxy*.

Software: Conjunto de informações capaz de controlar um *hardware* a fim de executar uma tarefa em um sistema computacional.

SSH: Shell seguro, meio de acesso remoto à linha de comando de um computador através da rede local ou Internet.

Stream, streaming: Método de transferência de informação (especialmente mídia audiovisual) através de rede, onde não é necessário salvar ou baixar o arquivo completo antes de visualizá-lo.

Symbian: *Sistema Operacional* utilizado em diversos smartphones de diversos fabricantes, especialmente Nokia, produzidos entre 2002 e 2012.

Tablet: Computador pessoal móvel com tela sensível ao toque. Sua operação pode se assemelhar a *smartphones* ou a computadores de mesa, a depender do sistema operacional utilizado.

WPA2: Tecnologia de proteção de redes WiFi.

WiFi: Tecnologia sem fios de acesso à rede.

Web: Parte da Internet correspondente ao seu conjunto de páginas e a ligação entre elas.