TRABALHO DE GRADUAÇÃO

ARVIM

Automação Residencial Via Mobile

Álvaro Henrique Figueiredo Nunes

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ARVIM

Automação Residencial Via Mobile

Álvaro Henrique Figueiredo Nunes

Relatório submetido ao Departamento de Engenharia Elétrica, como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação.

Banca Examinadora

Prof. Ricardo Zelenovsky, Doutor
Orientador

Prof. Daniel Chaves Café, Doutor
Examinador

Prof. Georges Daniel Amvame Nze, Doutor
Examinador

“A vida só pode ser compreendida olhando-se para trás; mas deve ser vivida olhando-se para frente”

(Søren Kierkegaard)

DEDICATÓRIA

Dedico este trabalho à minha mãe, Ana Maria, pela enorme paciência ao suportar as intervenções que fiz em casa para o desenvolvimento deste trabalho.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da vida. Aos meus pais, Edmilson e Ana Maria, especialmente à minha mãe pela enorme paciência, carinho, apoio e por ter suportado todas as intervenções que fiz em casa ao desenvolver e implantar este trabalho.

Ao meu grande amigo Santa Cruz pela enorme ajuda que sempre tem me prestado, tirando minhas dúvidas, me fornecendo ideias, me incentivando a descobrir.

Agradeço, também, à minha psicóloga Renata pelo apoio e incentivo, ajudando-me a ver um mundo de perspectivas diversas.

Ao meu orientador, professor Ricardo Zelenovsky, por sua enorme disposição em me orientar, sempre com alegria e bom humor. Por fim, agradeço à Universidade de Brasília por me oferecer, não só conhecimento acadêmico, mas um enorme aprendizado de vida.

Álvaro Henrique F. Nunes

RESUMO

Este trabalho tem por objetivo desenvolver um sistema de automação residencial de baixo custo para monitoramento, climatização e controle da iluminação, de acesso e de dispositivos eletrônicos através da Internet. O acesso a esse sistema se dá por meio de uma página *web* e um aplicativo para dispositivos *mobile* com sistema operacional Android. Como no Brasil a automação de residências ainda se apresenta onerosa e a maioria dos domicílios não são projetados para tal, se faz necessária uma busca por soluções mais acessíveis, com o mínimo de impacto na instalação e sem deixar de visar a qualidade. Para isso, foram testadas quatro composições diversas de *hardware* e *software* até que se atendesse ao propósito. Esse projeto não se limitou somente à fase de prototipagem, sendo implementado no mundo real. Apesar de haver aumento no custo entre a primeira e as versões posteriores, obteve-se um sistema de automação residencial de baixo custo, diversificado e com qualidade. Como sugestão para trabalho futuros, estão a expansão do sistema, o emprego de um modem roteador mais robusto e a compatibilidade do aplicativo com sistemas IOS da Apple.

ABSTRACT

This project aims to develop a low-cost home automation system for monitoring, climatization and control of lighting, access and electronic devices over the Internet. The user is capable of accessing this system through a web page and an app for Android compliant devices. Searching for low-cost automation alternatives it’s nescessary, since home automation still is very expensive and most homes aren’t designed to be automated in Brazil. Furthermore, it is also important keeping right quality without a great impact in house structure. For this purpose, four versions with distinct hardware and software specifications were developed. This project overcame the prototyping phase, being implemented in the real world. Despite of increasing costs among the versions, a low-cost good quality home automation system was achieved. As suggestion for future works, it’s recommended expanding the system and turning the app Apple’s IOS compliant.

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÔNIMOS

AC	*Alternating Current*
ADC	*Analog-to-Digital Converter*
AP	*Access Point*
APK	*Android Package*
ARM	*Advanced RISC Machine*
ARVIM	*Automação Residencial Via Mobile*
CFTV	*Circuito Fechado de Televisão*
CPU	*Central Process Unit*
CSV	*Comma-Separated Values*
DDNS	*Dynamic Domain Name System*
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
GPIO	*General Purpose Input/Output*
HDMI	*High-Definition Multimedia Interface*
HTML	*HyperText Markup Language*
HTTP	*Hyper Text Transport Protocol*
HTTPS	*Hyper Text Transport Protocol Secure*
IDE	*Integrated Development Enviroment*
IP	*Internet Protocol*
IR	*Infrared*
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diode*
NAT	*Network Address Translation*
NVR	*Network Video Recorder*
PEP	*Python Enhancement Proposals*
RTC	*Real Time Clock*
RTSP	*Real Time Streaming Protocol*
SSL	*Secure Socket Layer*
STA	*Station*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
URL	*Uniform Resource Locator*
WSGI	*Web Server Gateway Interface*

SUMÁRIO

1 INTRODUÇÃO

A automação residencial visa trazer maior conforto, segurança e praticidade, utilizando de dispositivos eletrônicos para facilitar a gestão dos recursos residenciais, conforme a necessidade do morador. Essa gestão, de forma autônoma ou manual, se dá por meio do monitoramento do ambiente por câmeras, por sensores de temperatura, umidade, do controle de acesso, do sistema de iluminação, dispositivos eletrônicos e outros. Ela também pode contribuir para a sustentabilidade, como a economia de água, de energia e de outros recursos, assim como auxiliar portadores de necessidades especiais e idosos. “No Brasil, existe um interesse por parte de 78% dos consumidores. No mundo, a média é de 66%” (AURESIDE, 2015).

A grande popularização dos *smartphones* e o maior acesso à Internet propiciaram formas ainda mais cômodas e práticas de se controlar, gerir e monitorar os recursos residenciais, mesmo o usuário estando a vários quilômetros de distância. Diferentemente das residências já projetadas e construídas para serem automatizadas, um dos desafios é adaptar aquelas que não possuem estrutura planejada para automação sem que eleve demais o custo, com o mínimo de impacto na instalação e sem deixar de visar a qualidade. “O custo de um projeto de automação de ambientes varia entre R$ 10 mil e R$ 30 mil [no Brasil]” (G1, 2013).

Segundo a MarketsAndMarkets (2017), o mercado mundial de sistemas de automação residencial foi avaliado em $39,93 bilhões em 2016 e estima-se que atinja $79,57 bilhões em 2022 graças a diversos fatores, dentre eles o significativo crescimento do mercado de *Internet of Things* (IoT) e o grande número de fabricantes expandindo seu portfólio de produtos. Um exemplo disso é o surgimento de opções mais baratas no mercado, como os dispositivos da linha SONOFF da empresa chinesa ITEAD (2017) que permitem uma vasta gama de possibilidades, tais como controlar lâmpadas, dimerizar LEDs RGB, medir consumo de energia e monitorar temperatura e umidade.

Outro desafio é desenvolver produtos mais amigáveis que permitam que o próprio consumidor seja capaz de instalar e configurar. Um produto com essa abordagem é a *Yeelight RGB* da empresa chinesa Yeelink (2017). Essa lâmpada de LED é capaz de se conectar a um ponto de acesso WiFi, permitindo, assim, o acionamento, a alteração da cor da luz e o controle de intensidade da luz através de um aplicativo para *smartphones*.

1.1 MOTIVAÇÃO

Tendo em vista o interesse dos consumidores brasileiros e o alto custo de um projeto de automação residencial, uma busca por alternativas mais baratas torna-se evidente. Embora haja no mercado soluções como as das empresas ITEAD (2017) e Yeelink (2017), a integração desses produtos pode ser trabalhosa, dificultando a vida do usuário que precisará de diferentes meios para utilizar de certos recursos em um ou diversos ambientes.

Com plataformas de código aberto como o Arduino, NodeMCU e Raspberry Pi é possível reduzir o custo da automação, tornar o projeto mais personalizável e integrado, suprindo de forma mais eficiente, e até mesmo mais sustentável, as necessidades do usuário.

1.2 OBJETIVO

O objetivo deste trabalho é desenvolver um sistema de automação residencial de baixo custo para monitoramento, climatização, controle da iluminação, de acesso e de dispositivos eletrônicos via navegador *web* e *mobiles* com o sistema operacional Android.

1.3 METODOLOGIA

Para atingir o objetivo, primeiramente, será desenvolvido um dispositivo central que tratará as requisições feitas pelo usuário e as encaminhará para os dispositivos periféricos. Logo em seguida serão desenvolvidos os dispositivos periféricos responsáveis pelo acionamento de cargas. Feito isso, será construído o dispositivo responsável pelo monitoramento das câmeras.

Após a construção desses dispositivos, serão desenvolvidos os códigos de funcionamento de cada um deles, exceto para o sistema de monitoramento das câmeras, o qual será utilizado um sistema operacional já desenvolvido para essa finalidade. Então, será necessário configurar o *modem* roteador para permitir o acesso externo à rede interna por meio de algumas portas.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho encontra-se organizado segundo os seguintes capítulos: Fundamentação Teórica, Montagem e Configuração, Ensaios e Conclusão. No capítulo 2, Fundamentação Teórica, serão abordados os conceitos utilizados nesse trabalho. Esse capítulo se subdivide em 3 seções: Conceitos de Redes de Computadores, Hardware e Software. A seção Conceito de Redes de Computadores tratará sobre conceitos necessários para o entendimento do funcionamento do projeto. Na seção Hardware se encontram as especificações do *hardware* utilizado no trabalho. Já na última seção, Software, está presente a descrição dos *softwares* utilizados.

No capítulo 3, Montagem e Configuração, serão detalhados os procedimentos realizados para integração entre *hardware* e *software*. Além disso, também será descrito o aplicativo ARVIM, para sistemas Android.

Posteriormente, no capítulo 4, Ensaios, serão descritos os testes realizados com a integração citada no capítulo 3 e, também, uma descrição sobre testes realizados com configurações diferentes de *hardware* e *software* utilizadas em versões anteriores do sistema. Por fim, no capitulo 5, Conclusão, discorrerá sobre a análise dos resultados dos testes realizados, observações sobre o funcionamento do sistema e propostas de continuação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os conceitos necessários para a construção do sistema de automação residencial. Para melhor compreensão, o capítulo encontra-se dividido em três seções. A primeira seção tratará sobre os conceitos de redes computadores de maior relevância para o trabalho. Na segunda seção será especificado o *hardware* utilizado nos dispositivos desenvolvidos. Por fim, a última seção explicará o *software* empregado.

2.1 CONCEITOS DE REDES DE COMPUTADORES

O sistema de automação residencial construído permite a gestão dos recursos residenciais de qualquer lugar com acesso à Internet. Isso pode ser feito tanto por um aplicativo para dispositivos *mobile* com sistema operacional Android quanto pelo *web browser*. Para que fosse possível uma comunicação segura entre o usuário e o sistema de automação, tanto por uma rede interna quanto por uma rede externa ao desse sistema, empregaram-se os conceitos a seguir.

2.1.1 DYNAMIC DOMAIN NAME SYSTEM (DDNS)

O DDNS (*Dynamic Domain Name System*), assim como o DNS (IETF, 1987), permite a tradução de um nome em endereço IP. Entretanto, o DDNS foi projetado para também suportar endereços IP dinâmicos, como aqueles designados pelo servidor DHCP (*Dynamic Host Configuration Protocol*). Tal característica permite que redes domésticas sejam alcançadas, visto que elas recebem um IP dinâmico público do provedor de internet contratado (CARDOSO, 2015).

O serviço DDNS funciona por meio de um software, instalado em um *host* ou roteador, que monitora as mudanças ocorridas no endereço IP dinâmico. Caso ocorra uma mudança, o software informa ao provedor de DDNS para atualizar o endereço IP relativo àquele domínio. Por exemplo, o domínio arvim.ddns.net é traduzido para o endereço IP dinâmico público 177.43.25.11 alocado ao roteador da rede do sistema de automação, entretanto esse IP não permanece sempre o mesmo. Sendo assim, é necessário que o *software* responsável informe ao provedor de DDNS a mudança de endereço para que se possa alcançar a rede desejada.

2.1.2 NETWORK ADDRESS TRANSLATION E PORT FORWARDING

O NAT (*Network Address Translation*) é um protocolo que permite redes IP privadas, que utilizam endereços IP não registrados, tenham acesso à Internet. Ele opera em um roteador, usualmente interligando duas redes, traduzindo os endereços privados na rede interna para endereços legais, antes de os pacotes serem enviados para a outra rede (CISCO, 2014).

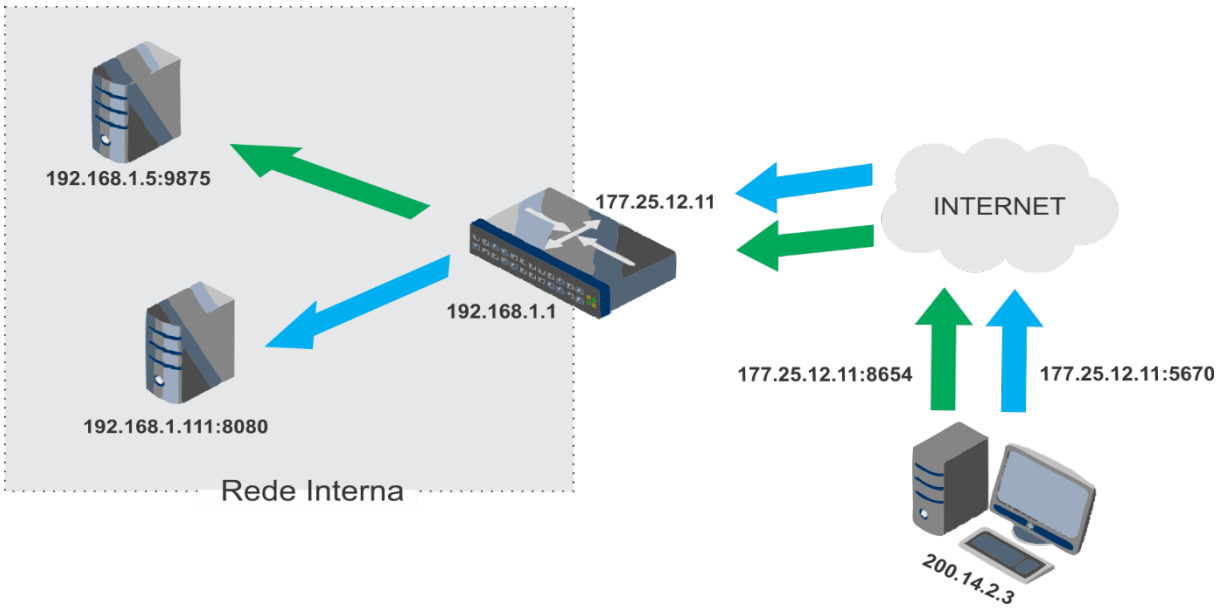

Figura 1- Ilustração do funcionamento do NAT juntamente com o *Port Forwarding* (Própria,2018)

O *Port Forwarding* é uma técnica que redireciona uma requisição de uma combinação de endereço IP e número de porta para outra combinação, durante a passagem dos pacotes pelo *gateway*, tal como um roteador ou *firewall* (TECHOPEDIA, 2017). A figura 1 exemplifica o funcionamento do NAT e do *Port Forwarding*. O cliente de endereço IP 200.14.2.3 faz duas requisições para o IP público 177.25.12.11, uma para a porta 5670 (**177.25.12.11:5670**) e outra para a porta 8654 (**177.25.12.11:8654**). Essas requisições são representadas, respectivamente, pelas setas verde e azul. O roteador que recebe as requisições é identificado na rede externa pelo IP público 177.25.12.11 e ele traduz o endereçamento externo para o interno, encaminhando cada requisição para a combinação de endereço IP local e porta. Logo, a combinação **177.25.12.11:5670** teve como destino a combinação **192.168.1.111:8080** e a combinação **177.25.12.11:8654** teve como destino **192.168.1.5:9875**.

2.1.3 PROXY

O *Proxy* ou servidor *proxy* é basicamente um servidor intermediário que processa as requisições e provê recursos de outros servidores ao cliente. O *proxy* que atua em nome de um serviço ou produtor de conteúdo é denominado *Reverse Proxy* e o que atua em nome de um solicitante ou consumidor de serviço chama-se *Forward Proxy* (JSCAPE, 2012).

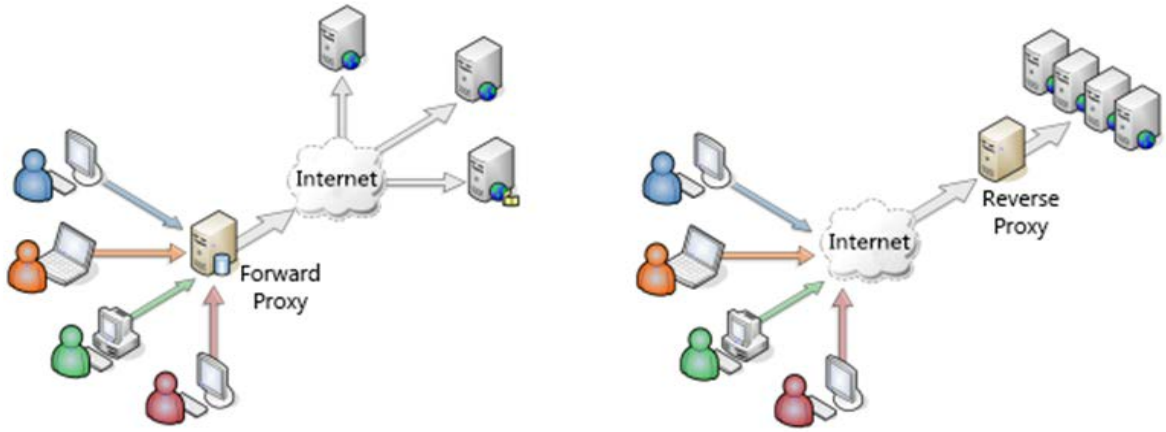

Figura 2 - Ilustração do funcionamento do *Forward Proxy* e *Reverse Proxy* (HAFER, 2014)

2.1.4 REAL TIME STREAMING PROTOCOL (RTSP)

O RTSP (*Real Time Streaming Protocol*) é um protocolo que possibilita a interação cliente-servidor entre a fonte do fluxo de mídia a taxa constante, denominada servidor, e o usuário, denominado transdutor (KUROSE e ROSS, 2009).

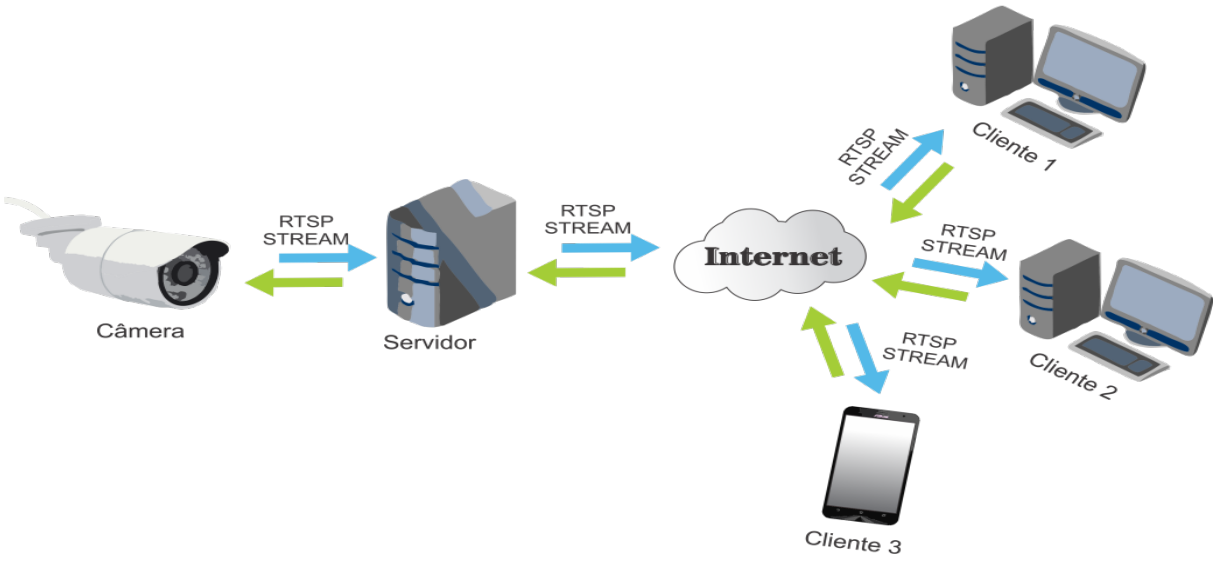

Figura 3- Ilustração do funcionamento do protocolo RTSP. (Própria, 2018)

Esse protocolo tem por funcionalidades as mesmas que um aparelho reprodutor de DVD, ou seja, permite que um transdutor controle o fluxo de mídia mediante comandos de pausa, reinício, retrocesso, avanço rápido e reposicionamento da reprodução. Portanto, o RTSP não

encapsula os comandos no mesmo pacote de transmissão de mídia, mas ele os envia paralelamente em portas diferentes, da mesma maneira que o protocolo FTP (*File Transfer Protocol*) e, por isso, é denominado protocolo fora de banda. O canal de transmissão de mídia é considerado como sendo a banda (KUROSE e ROSS, 2009).

2.1.5 PROTOCOLOS DE SEGURANÇA: TLS E HTTPS

O TLS (*Transport Layer Security*) é um protocolo de criptografia, sucessor do SSL (*Secure Socket Layer*), que fornece segurança à comunicação. Seus objetivos são prover a integridade dos dados e a privacidade entre duas aplicações computacionais. Esse protocolo autentica e garante transferências de dados a partir da autenticação baseada em certificados e chaves de criptografia simétrica, permitindo às aplicações do cliente ou servidor detectar mensagens adulteradas, interceptação de mensagens e falsificações de mensagens (KUROSE e ROSS, 2009).

O HTTPS (*Hypertext Transport Protocol Secure*) é a implementação do HTTP com SSL ou TLS garantindo a confidencialidade, autenticidade e integridade da comunicação entre o cliente e os servidores web (MICROSOFT, 2015). Uma conexão HTTP sem essa medida pode ser facilmente monitorada e adulterada.

2.2 HARDWARE

Nesta seção serão especificados os componentes de *hardware* utilizados para a construção dos dispositivos do sistema de automação residencial. Após vários testes com outros componentes, foi escolhida a composição abaixo por apresentar uma melhor relação entre qualidade e custo.

2.2.1 PLATAFORMA DE PROTOTIPAGEM ARDUINO

Segundo Michael McRoberts (2015, p. 22) “O Arduino é o que chamamos de plataforma de computação física ou embarcada, ou seja, um sistema que pode interagir com seu ambiente por meio de hardware e software”. Essa plataforma de prototipagem eletrônica de código aberto e *hardware* livre dispõe de um micro controlador, geralmente um Atmel AVR, e entradas e saídas, sendo elas analógicas ou digitais.

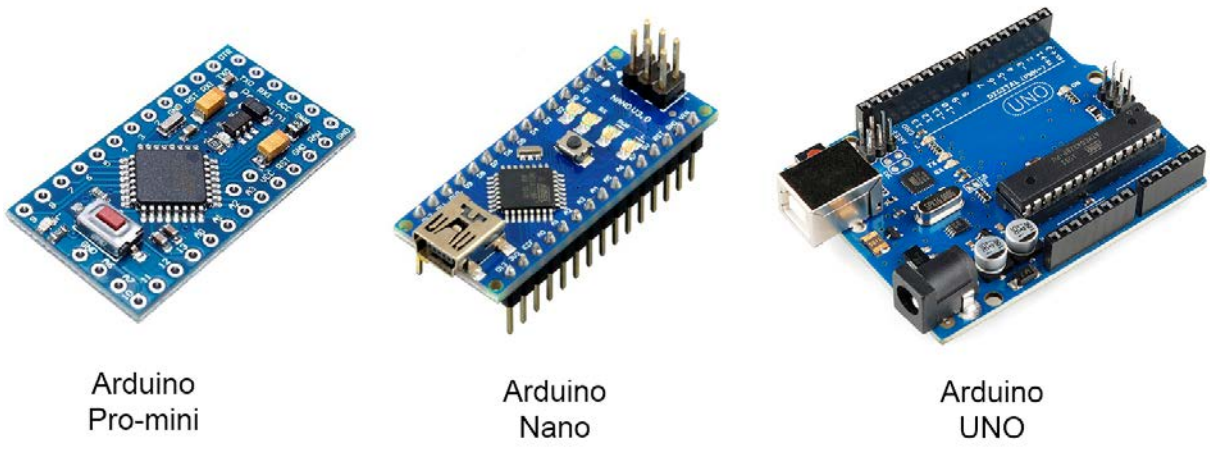

Figura 4 – Versões do Arduino utilizadas no projeto (Própria, 2018)

Nascido no Ivrea Interaction Design Institute, o intuito de sua criação era ser uma ferramenta de prototipagem fácil e rápida, destinada a estudantes sem experiência em eletrônica e programação. Atualmente existem diversas versões do Arduino como: Arduino Pro-Mini, Arduino Nano, Arduino UNO, Arduino Mega e entre outras. Cada versão permite uma aplicação diferente, por exemplo as versões LilyPad Arduino podem ser costuradas em tecidos possibilitando a confecção de trajes, roupas e fantasias com diversas funções, tais como efeitos luminosos e outras aplicações.

Entre as vantagens de se utilizar um Arduino estão o custo relativamente baixo, multiplataforma, ambiente de programação simples e claro, software de código aberto e extensível e hardware aberto e extensível.

2.2.2 PLATAFORMA DE INTERNET DAS COISAS NODEMCU

O NodeMCU é uma plataforma IoT (*Internet of Things*) de código aberto. O *hardware* utilizado é baseado no ESP-12 e o *firmware* executado no ESP8266 Wi-Fi SoC da Espressif Systems (ESPRESSIF, 2018). A designação NodeMCU refere-se ao *firmware* ao invés dos kits de desenvolvimento. Embora o firmware use linguagem de script Lua, atualmente é possível programá-lo em C# utilizando a própria IDE (*Integrated Development Environment*) do Arduino, necessitando apenas da instalação de algumas bibliotecas disponibilizadas nos repositórios GitHub (SZDOIT, 2015).

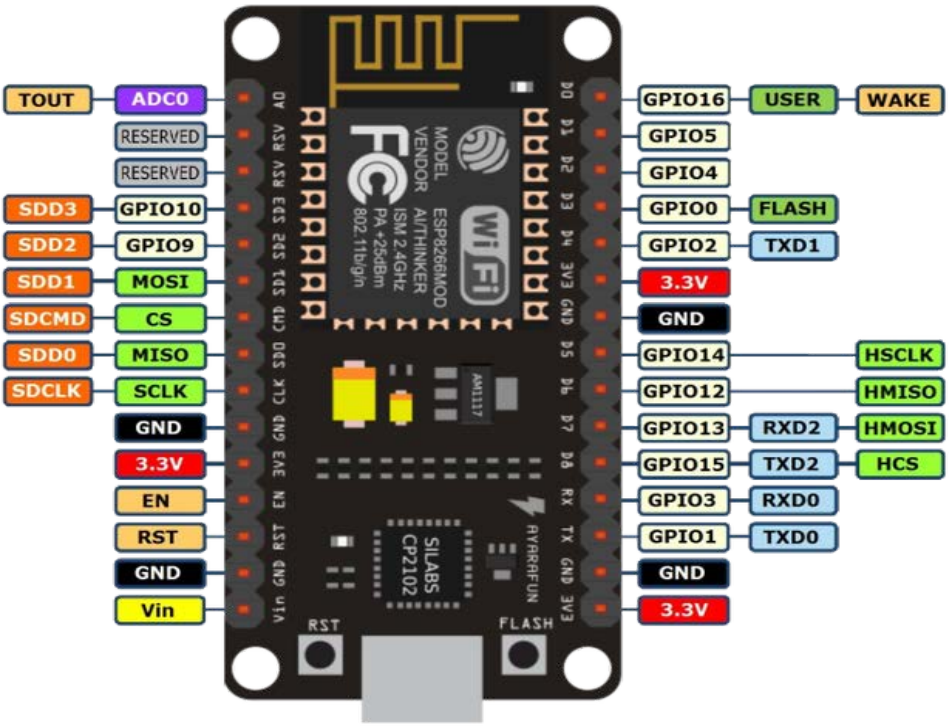

Figura 5: Imagem ilustrativa do *pinout* do NodeMCU 1.0 (HELGESCHNEIDER, 2017)

Desta forma, é possível implementar, utilizando bibliotecas específicas para o NodeMCU na Arduino IDE, de maneira simples, fácil e rápida, pequenos e simples servidores *web* capazes de controlar relês, realizar leitura de sensores e muito mais, de forma *wireless*. Suas especificações de *hardware* são as seguintes:

- 3 modos de funcionamento: STA / AP / STA + AP;
- Pilha de protocolo TCP / IP embutida, suporte à conexão de cliente TCP de múltiplos canais (máximo 5);
- 0 ~ D8, SD1 ~ SD3: usado para GPIO, PWM, IIC; A corrente máxima alcançada por pino é de 15mA;
- AD0: ADC unidirecional;
- Entrada de energia: 4.5V ~ 9V (10VMAX), suporta alimentação USB e USB debug;

- Corrente de trabalho: ≈ 70mA (200mA MAX, contínua), em espera <200uA;
- Taxa de dados de transmissão: 110-460800bps;
- Suporta interface de comunicação de dados UART / GPIO;
- Suporta o *firmware* de atualização remotamente (OTA);
- Suporte a Smart Link;
- Temperatura de trabalho: -40 ℃ ~ + 125 ℃;
- Peso: 7g.

2.2.3 RASPBERRY PI, O MINI COMPUTADOR

O Raspberry Pi 3 é um computador de placa única extremamente pequeno, de alta performance e de baixo custo. Foi criado na Universidade Cambrigde, Reino Unido, com intuito de incentivar as mais diversas faixas etárias a aprender mais sobre ciência da computação (MCMANUS e COOK, 2013).

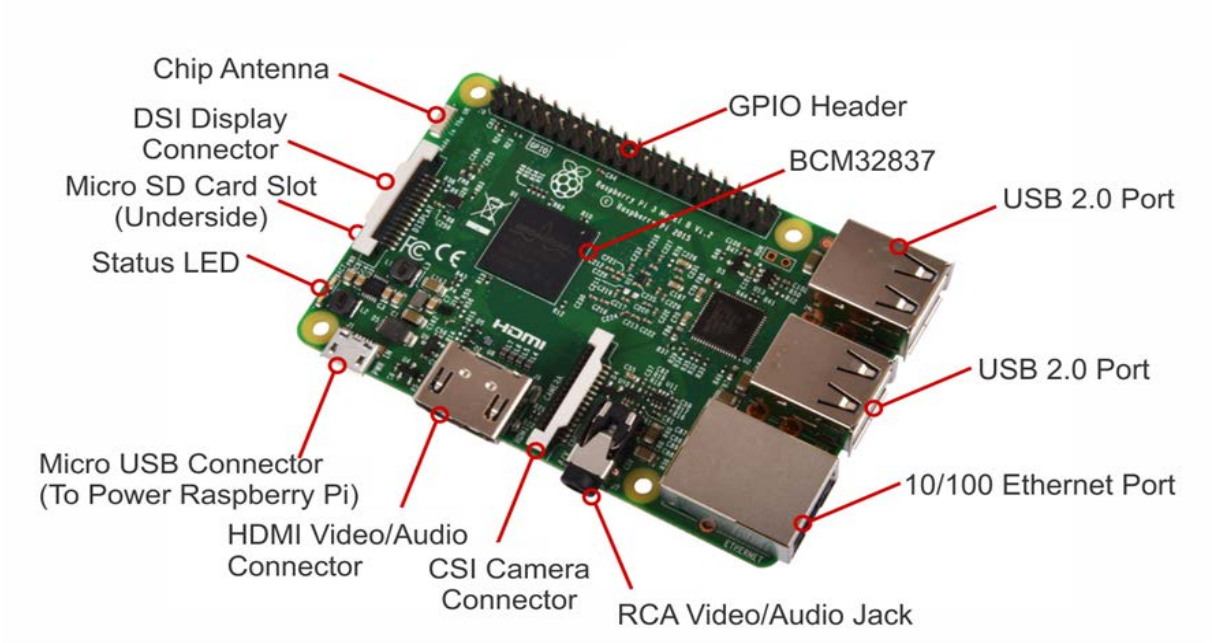

Figura 6- Imagem real do Raspberry Pi com seus componentes indicados (CNXSOFT, 2016)

A versão utilizada foi o Raspberry Pi 3 Model B e sua grande vantagem, em relação às predecessoras, é a integração do WiFi e o Bluetooth na própria placa, não necessitando utilizar adaptador, como nas versões anteriores. As especificações de *hardware* estão listadas abaixo:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN e Bluetooth Low Energy (BLE) integrado
- 40 GPIO pins extensíveis

- Ethernet port e 4 portas USB 2.0
- 4 Pole stereo output e composite video port
- Full size HDMI
- Camera interface (CSI) e Display interface (DSI)
- Micro SD card slot

2.2.4 CÂMERAS IP

Uma câmera IP é um tipo de câmera de vídeo digital utilizada em sistemas de monitoramento, que ao contrário das câmeras de CFTV (*Circuito Fechado de Televisão*), enviam e recebem dados através de uma rede local e da Internet.

Figura 7- Os 3 tipos mais comuns de câmeras para sistemas de vídeo vigilância (Própria, 2018)

Existem tanto as câmeras IP centralizadas, que necessitam de um NVR (*Network Video Recorder*) para gravar e armazenar o vídeo, quanto as descentralizadas que não necessitam de um NVR, podendo armazenar o vídeo em um dispositivo de armazenamento como um cartão microSD. Além disso, existem câmeras IP wireless, facilitando ainda mais a instalação (ALMEIDA, 2015).

2.3 SOFTWARE

O *software* necessário para a construção do sistema será detalhado nessa seção. Primeiramente, será explicitado o *software* utilizado no dispositivo central, aquele que atuará como um servidor *reverse proxy* (subseções 2.3.1 a 2.3.5). Em seguida, na subseção 2.3.6, será caracterizado o sistema operacional que atuará no servidor responsável pelo monitoramento das câmeras. Por fim, serão especificados o sistema operacional Android e o ambiente de desenvolvimento para dispositivos *mobile* Android.

2.3.1 SISTEMA OPERACIONAL RASPBIAN

O sistema operacional Raspbian é baseado no Debian Linux e foi projetado exatamente para o Raspberry Pi. Esse sistema operacional foi criado em 2012 por Mike Thompson e Peter Green como um projeto independente e se encontra em constante desenvolvimento. Ele é altamente otimizado para as CPUs ARM de baixo desempenho da linha Raspberry Pi (RASPBIAN, 2017).

2.3.2 APLICATIVO DE CONTROLE REMOTO VNC

O VNC Viewer é um software de acesso remoto que já vem pré-instalado nas versões mais recentes do Raspbian. Desenvolvido pela empresa RealVNC, esse aplicativo possibilita a configuração e controle do Raspberry Pi sem a necessidade de uma TV ou monitor de computador e mouse para isso. O controle pode ser feito utilizando-se de dispositivos como *smartphones*, *tablets* e computadores de uso pessoal. “Ele [VNC Viewer] consiste em um aplicativo VNC Server para o computador que você deseja controlar, além de programas de suporte” (REALVNC, 2018).

2.3.3 SERVIDOR NGINX

O Nginx é um software de código aberto para web *serving*, *reverse proxying*, *streaming* de mídia, balanceamento de carga e entre outros. Desenvolvido em 2002 pelo desenvolvedor Igor Sysoev, é um servidor web rápido, leve e com múltiplas possibilidades de configuração para uma alta performance (NGINX, 2017).

Por lidar com requisições web através do conceito *event-based web server*, o Nginx, tecnicamente, consome menos memória que o Apache, que é baseado no conceito *process-based server*. “O Nginx pode lidar com cerca de 40% a mais de tráfego em um tempo surpreendente de 300% a menos. Isso torna o Nginx cerca de 4,2 vezes mais rápido que o Apache” (POLLOCK, 2017).

2.3.4 APLICAÇÃO UWSGI

O uWSGI é um software para implementação de um servidor de aplicação que é responsável pela comunicação entre o *framework* e o servidor *web*. Os pontos fortes desse software são a versatilidade, bom desempenho, baixo uso de recursos e a confiabilidade (UWSGI, 2016).

A parte WSGI no nome homenageia uma especificação, disposta no PEP 333 (EBY, 2003), para uma interface padronizada entre servidores *web* e *frameworks* e aplicações Python Web. O objetivo é oferecer uma interface relativamente simples e abrangente, capaz de suportar a maioria das interações entre um servidor *web* e um *web framework*.

2.3.5 MICROFRAMEWORK FLASK

O Flask é um *micro web framework* baseado na biblioteca WSGI Werkzeug e na biblioteca Jinja2 e escrito em Python. Além de possuir a flexibilidade que a linguagem Python dispõe, ele provê um modelo simples para desenvolvimento *web* (RONACHER, 2010).

Entre seus principais recursos estão suporte integrado para testes unitários, contém servidor de desenvolvimento e depurador, *RESTful request dispatching*, utiliza modelos do Jinja2, um mecanismo de template completo para Python, suporta *cookies* seguros do lado do cliente, é compatível com WSGI 1.0 (MAKAI, 2012). Além disso, possui documentação extensa, compatibilidade com o Google App Engine e extensões disponíveis para melhorar os recursos desejados.

2.3.6 SISTEMA OPERACIONAL MOTIONEYE OS

Segundo Calin Crisan (2017), “O motionEyeOS é uma distribuição Linux que transforma um computador de placa única em um sistema de vigilância por vídeo. O sistema operacional é baseado no *BuildRoot* e usa o *motion* como *backend* e *motionEye* para o *frontend*”. Esse versátil e intuitivo sistema operacional possibilita o desenvolvimento de uma central de monitoramento de vídeo semelhante a um NVR (*Network Video Recorder*).

Utilizando esse sistema operacional no Raspberry Pi, torna-se viável o *streaming* de vídeo de várias câmeras IP, assim como *webcams* e a própria câmera desenvolvida para o Raspberry Pi. A partir do *motion*, gravações de vídeo e *snapshots* são feitos conforme a detecção de movimento. Esses registros são armazenados no cartão microSD onde está armazenado o motionEyeOS ou em um dispositivo de armazenamento conectado via USB. Por possuir um servidor FTP, esse sistema operacional permite que os registros de movimento sejam baixados pelo usuário (CRISAN, 2017).

O motionEyeOS foi desenvolvido por Calin Crisan e é compatível com os seguintes computadores de placa única: Banana Pi, Nano Pi Neo2, Oragen Pi One, Odroid C1/C1+, Odroid C2, Odroid XU4, Pine A64/A64+, Raspberry Pi (modelos A, B, A+, B+, Compute Module, Zero e Zero W), Raspberry Pi 2 e Raspberry Pi 3 (modelos B e Compute Module 3).

2.3.7 SISTEMA OPERACIONAL ANDROID

Atualmente desenvolvido pela empresa Google e baseado no núcleo Linux, o Android é um sistema operacional com interface de usuário de interação direta. O Android foi projetado para dispositivos móveis (*mobile*) com telas sensíveis ao toque (*touch screen)*. O *touch screen* permite com que o usuário comande seu dispositivo através da manipulação dos objetos virtuais, por exemplo, o teclado virtual. O fato de o Android ser de natureza de software de código aberto tem incentivado uma grande comunidade de programadores e entusiastas a desenvolverem os mais diversos projetos com esse sistema operacional.

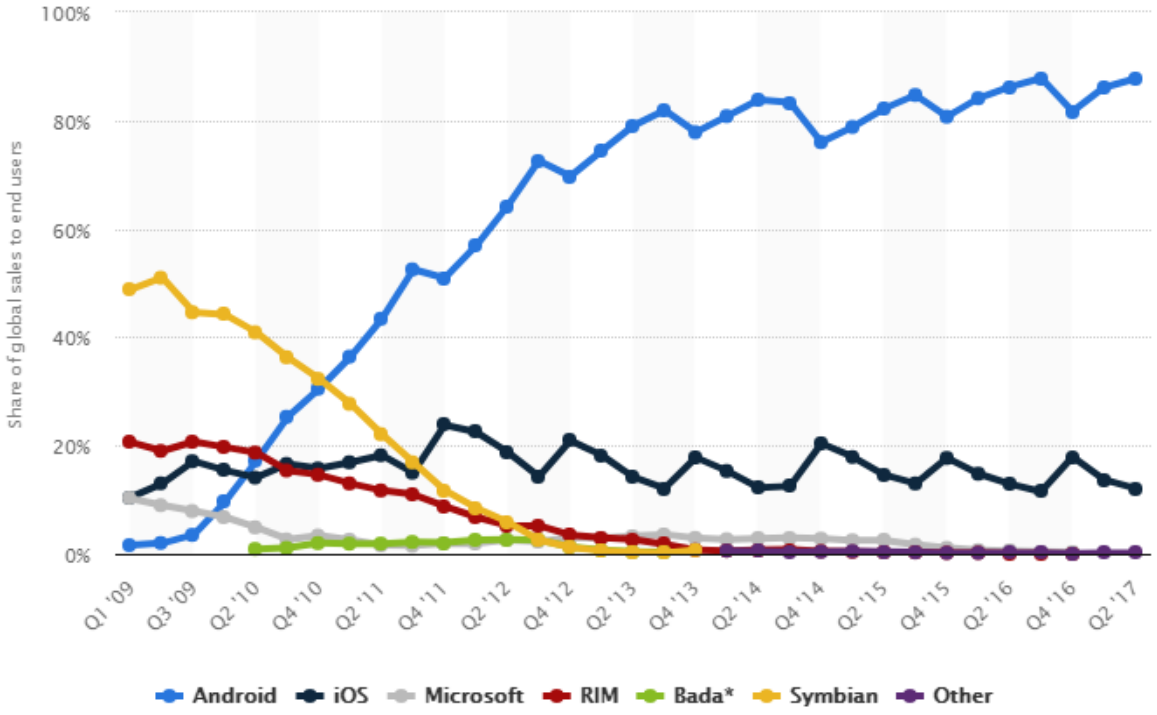

Figura 8- Participação no mercado global dos sistemas operacionais mobile em vendas para usuários finais (STATISTA, 2018)

Por mais que esse sistema operacional domine o mercado global desde 2011 e tenha o maior número de usuários ativos, conforme mostrado nas Figura 8 e 9 respectivamente, os lucros obtidos pelos os desenvolvedores de aplicativos não têm relação com a popularidade. O motivo por trás da baixa rentabilidade deve-se ao fato de que há menos usuários no sistema da Google que estão dispostos a pagar por aplicativos, resultando uma média menor de lucro entre

os sistemas citados. “Em média, cada download realizado na Play Store gera US$ 0,018 para os desenvolvedores, ao mesmo tempo em que na App Store da Apple esse valor chega aos US$ 0,10. No Windows Phone há ainda mais rentabilidade, terminando em US$ 0,15” (HAMAN, 2014).

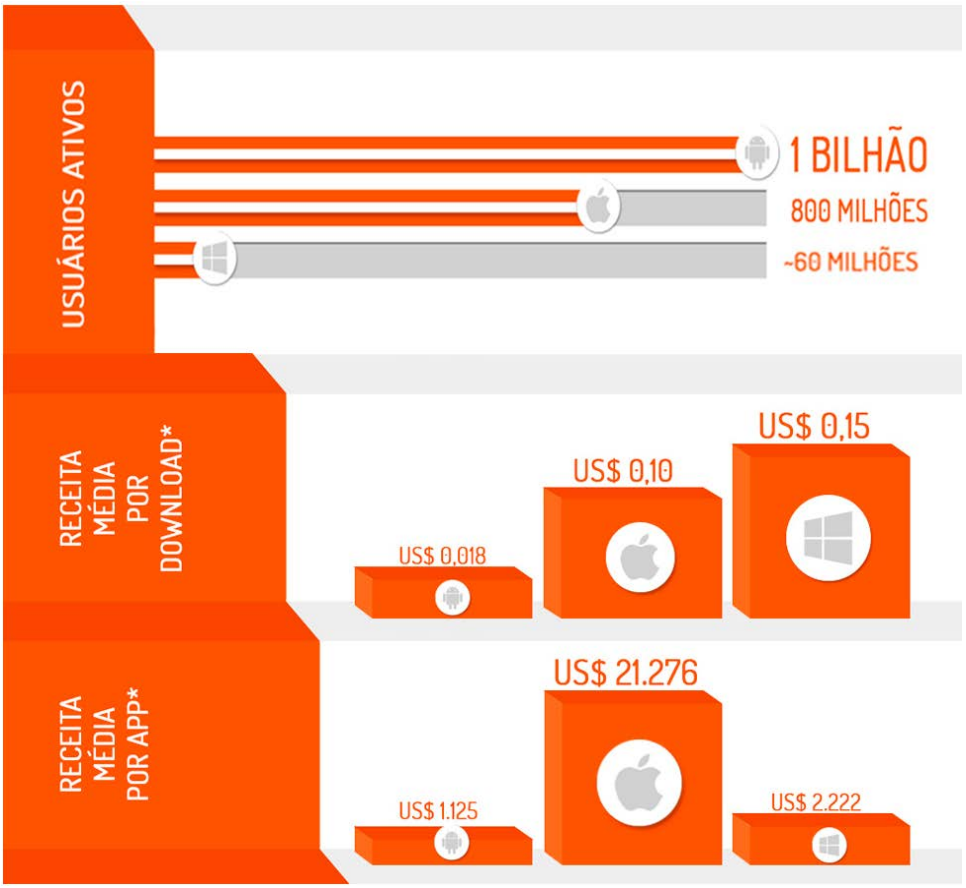

Figura 9 – Comparativo da rentabilidade entre sistemas operacionais *mobile* (HAMAN, 2014)

2.3.8 AMBIENTE DE DESENVOLVIMENTO ANDROID STUDIO

O Android Studio é uma IDE (*Integrated Development Environment)* para desenvolvimento de aplicativos Android baseado no ambiente de desenvolvimento integrado IntelliJ IDEA. O Android Studio dispõe dos seguintes recursos:

- Editor de código e ferramentas avançadas de desenvolvedor do IntelliJ.
- Sistema flexível de compilação baseado no Gradle.
- Emulador rápido com inúmeros recursos.
- Ambiente unificado de desenvolvimento para todos os dispositivos Android.
- Instant Run que possibilita aplicar alterações em aplicativo em execução sem necessidade de compilar novamente o APK.
- Modelos de código e integração com o GitHub.

- Ferramentas e estruturas de teste com variadas possibilidades.
- Ferramentas de verificação de código suspeito, detectando problemas de desempenho, usabilidade e compatibilidade.
- Compatibilidade com C++ e NDK e compatibilidade embutida com Google Cloud Plataform.

Essa IDE proporciona uma programação fácil e rápida. Entre seus principais destaques está a ferramenta de preenchimento inteligente que exibe opções relevantes para o contexto em questão, conhecendo os tipos e fluxos de dados esperados (GOOGLE DEVELOPERS, 2017).

3 MONTAGEM E CONFIGURAÇÃO

O sistema de automação proposto consiste em dois servidores principais, um para atuar como *reverse proxy* e outro para *streaming* e gravação de vídeo das câmeras. Além disso, há os servidores secundários que são dispositivos a serem controlados.

O servidor responsável pelo *reverse proxy* também será responsável pela leitura da temperatura e umidade do ambiente, pelo acionamento de quatro relês e por hospedar a página web do sistema de automação residencial. Esse servidor será chamado de servidor central.

Os servidores secundários serão nomeados servidores de controle. Eles se comunicarão com o usuário por meio do *reverse proxy* realizado pelo servidor central. Em outras palavras, o servidor central servirá de intermediário na comunicação entre o usuário e o servidor de controle. Já o servidor de *streaming* e gravação de vídeo das câmeras será o servidor de monitoramento.

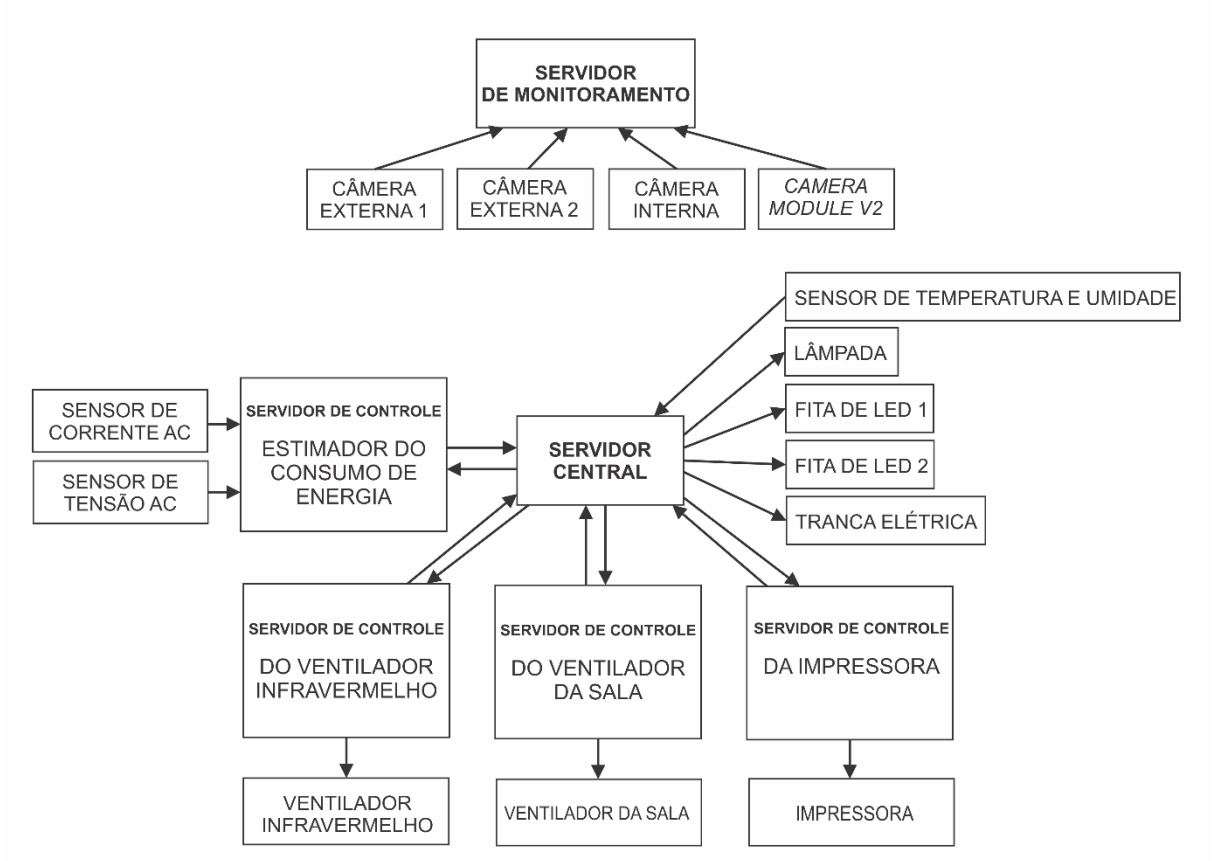

Figura 10 - Diagrama de blocos do sistema ARVIM (Própria, 2018)

Primeiramente será abordada a montagem do *hardware* dos servidores e, posteriormente, a construção e configuração do *software*. Os esquemáticos de montagem e os códigos desenvolvidos encontram-se nos Apêndices A e B, respectivamente

3.1 MONTAGEM DOS DISPOSITIVOS

A montagem dos dispositivos foi dividida em três partes: a montagem do servidor central, dos servidores de controle e do servidor de monitoramento. Conforme definido anteriormente, os servidores de controle são os dispositivos a serem controlados, entretanto há entre eles um servidor de controle diferenciado. Esse servidor é o servidor de controle estimador de consumo de energia que estimará a potência instantânea através das medições realizadas por um sensor de corrente e por um sensor de tensão.

3.1.1 SERVIDOR CENTRAL

A montagem do servidor central foi realizada conforme ilustra o diagrama de blocos da Figura 11. Para possibilitar o acionamento da lâmpada e das duas fitas de LED tanto pelo relê quanto pelo interruptor, foram utilizados interruptores *three way* em paralelo com o módulo relê de quatro canais. Já para a tranca elétrica instalada no portal, ligou-se em série com um dos canais do relê.

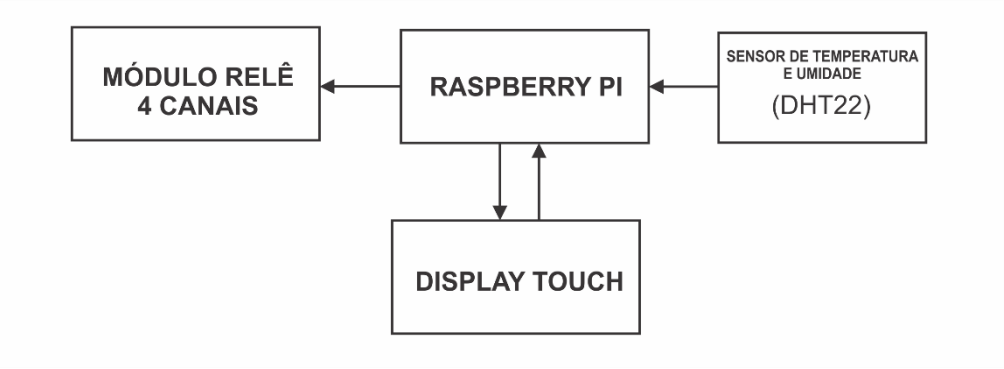

Figura 11 - Diagrama de blocos da montagem do servidor central (Própria, 2018)

Para realizar as medições de temperatura e umidade, utilizou-se o sensor de temperatura DHT22 que permite fazer leituras de temperaturas entre -40°C a +125ºC e umidade entre 0 UR a 100% UR (HU INFINITO, 2018). Além disso, empregou-se um Raspberry Pi Touch Display de tela de 7 polegadas com resolução 800 x 480 *pixels* que se conecta ao Raspberry Pi 3 por meio de uma placa adaptadora (RASPBERRY PI FOUNDATION, 2018).

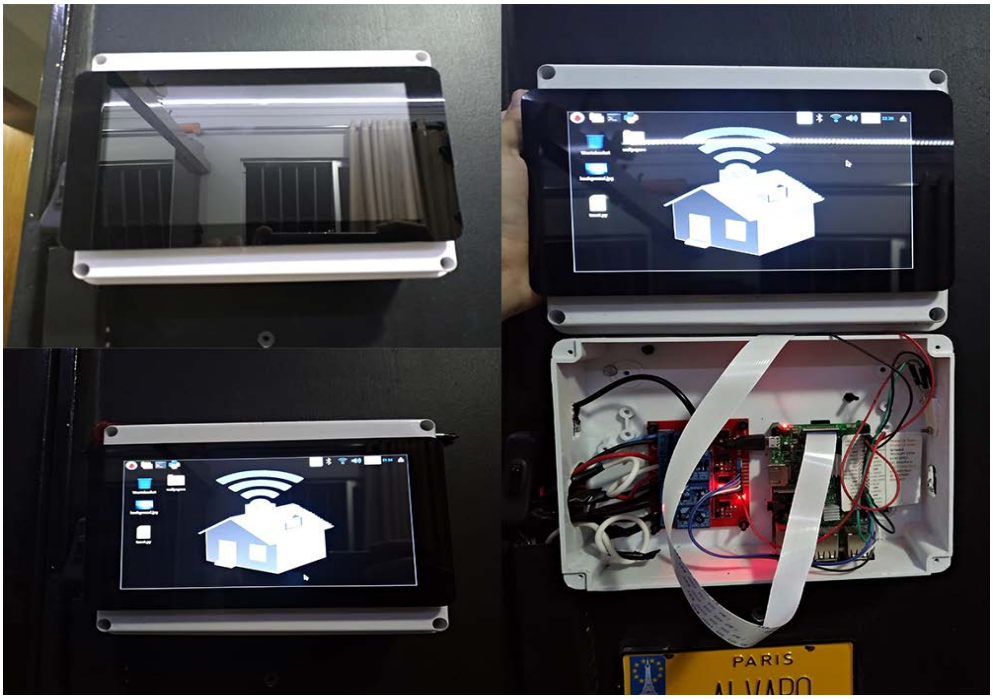

Figura 12 - Foto do servidor central (Própria, 2018)

3.1.2 SERVIDORES DE CONTROLE

Os servidores de controle, como já dito anteriormente, são os dispositivos a serem controlados. Construiu-se quatros servidores: um servidor para controlar a impressora, um para estimar o consumo de energia, outro para controlar um ventilador via infravermelho e, por fim, um para controlar o ventilador da sala. As Figuras de 13 a 16 mostram, respectivamente, os diagramas de blocos desses quatros servidores.

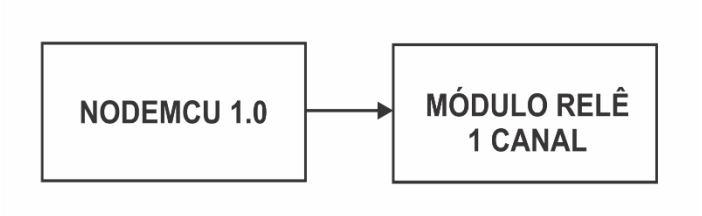

Figura 13 - Diagrama de blocos do servidor de controle da impressora (Própria, 2018)

O servidor de controle da impressora consiste, basicamente, em um NodeMCU que comanda um relê ligado em série com a impressora. Dessa forma, o NodeMCU liga ou desliga a impressora conforme a requisição recebida (Figura 13).

Figura 14 - Foto do servidor de controle da impressora (Própria, 2018)

Para estimar o consumo de energia utilizou-se o sensor P8 para medir a tensão AC e o sensor ACS712 para medir corrente AC. Devido aos sensores possuírem saídas analógicas e ao NodeMCU possuir apenas uma entrada analógica (Figura 5), aplicou-se um Arduino Pro-Mini comunicando-se serialmente com NodeMCU, conforme mostra a Figura 15.

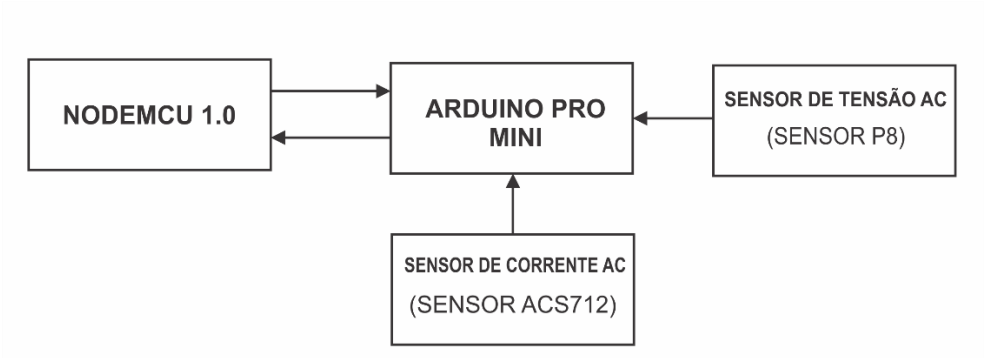

Figura 15 - Diagrama de blocos do servidor de controle estimador de consumo (Própria, 2018)

Figura 16 - Foto do servidor de controle estimador do consumo de energia (Própria, 2018)

A montagem do servidor de controle do ventilador infravermelho foi desenvolvida conforme ilustra a Figura 17. Utilizou-se um Arduino Nano para enviar sinais IR para controlar o ventilador infravermelho. O envio desses sinais é comandado pelo NodeMCU via comunicação serial.

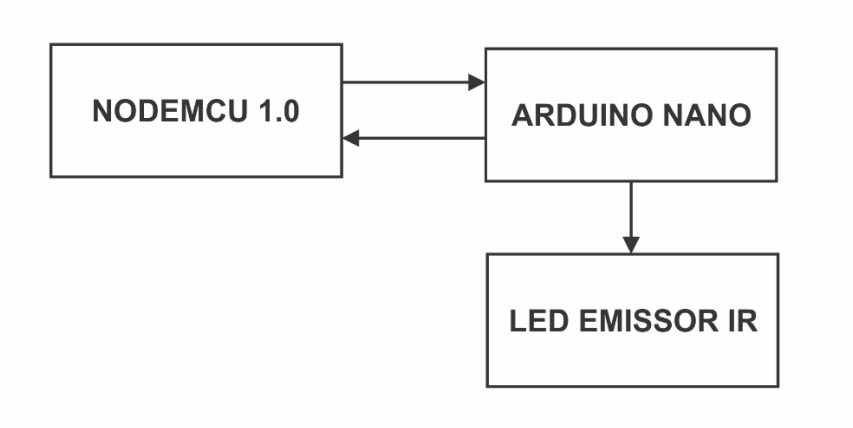

Figura 17 - Diagrama de blocos do servidor de controle do ventilador infravermelho (Própria, 2018)

Figura 18 - Foto do servidor de controle do ventilador infravermelho (Própria, 2018)

No servidor que controla o ventilador da sala, adicionou-se a função de acionamento via sinais infravermelho por meio de um receptor IR (Figura 19). Assim como no servidor central, o módulo relê de 1 canal foi ligado paralelamente a um interruptor *three way* (Figura 20, item 1). Portanto, é possível acionar o ventilador da sala de três formas: via infravermelho, pela Internet e pelo interruptor.

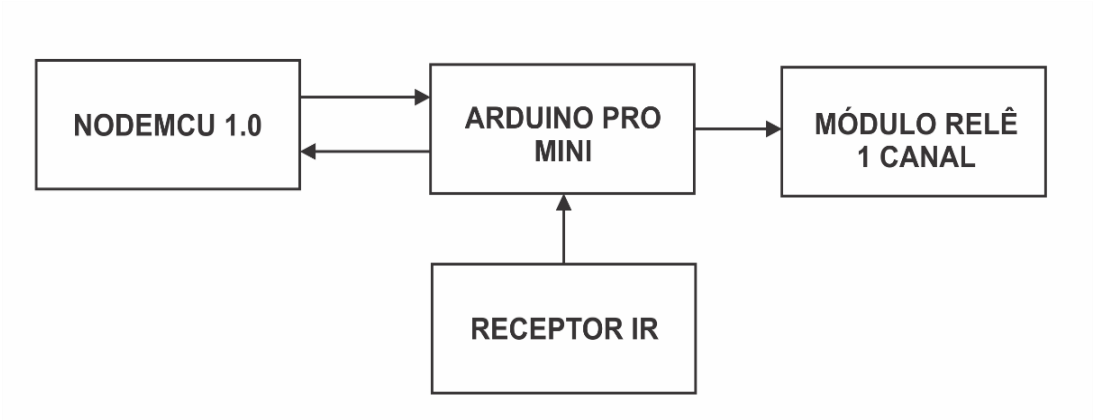

Figura 19 - Diagrama de blocos do servidor de controle do ventilador da sala (Própria, 2018)

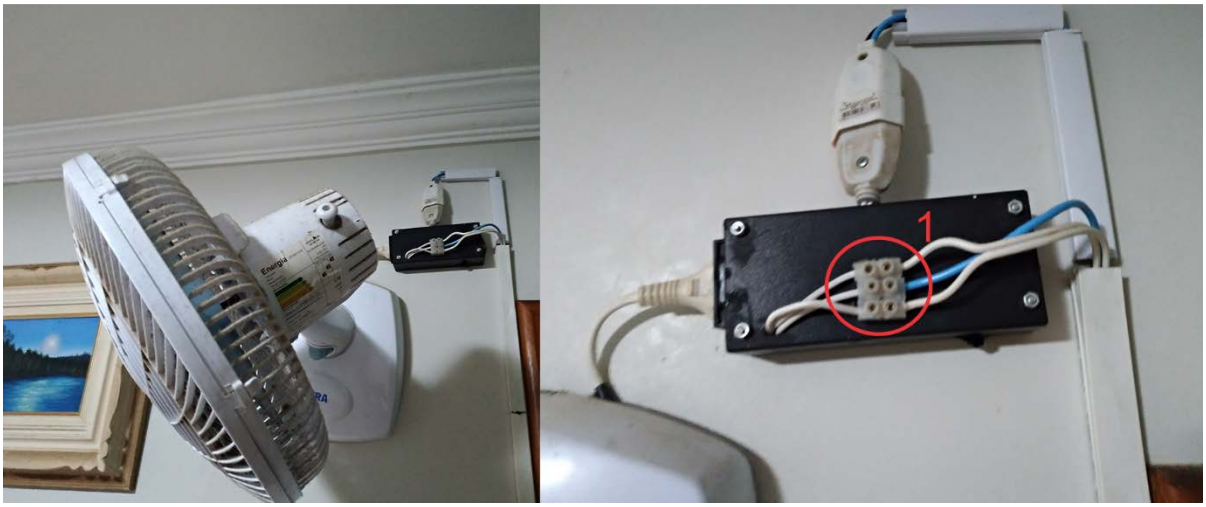

Figura 20 - Foto do servidor de controle ventilador da sala (Própria, 2018)

3.1.3 SERVIDOR DE MONITORAMENTO

O servidor de monitoramento se resume a um Raspberry Pi 3 com uma *Camera Module V2* de 8 *megapixels* (RASPBERRY PI FOUNDATION, 2018). Ele foi montado na porta da residência deste aluno para que a câmera registre a movimentação do corredor. Além disso, também é responsável por integrar outras três câmeras espalhadas pela residência. O seu papel é, estritamente, realizar o *streaming* e gravação de vídeo das câmeras.

Figura 21 - Raspberry Pi 3 e a *Camera Module V2* (RASPBERRY PI FOUNDATION, 2018)

3.2 CONFIGURAÇÃO DOS DISPOSITIVOS

A escolha da disposição de um servidor central com o Raspberry Pi 3 e outros servidores de controle empregando o NodeMCU deve-se ao fato de o NodeMCU não ser robusto quanto à hospedagem de uma página web e da impossibilidade da implementação do protocolo de segurança TLS. Neste capítulo serão abordados o desenvolvimento e a configuração do *software* dos servidores e do aplicativo para *mobiles* Android.

Primeiramente é necessário configurar o DDNS no modem roteador para permitir que a rede local seja acessada externamente. Para tal, foi escolhido o provedor No-IP que possibilita a criação de uma conta gratuita com 3 nomes de *host* (NO-IP, 2018). Então, ao acessar localmente o modem roteador, basta entrar na guia DDNS e efetuar o *login* da conta informando o nome de domínio para funcionamento do DDNS.

3.2.1 SERVIDOR PRINCIPAL

Com o Raspberry Pi já com um microSD do sistema operacional Raspbian e conectado ao WiFi do modem roteador, utiliza-se o VNC Viewer para acessar remotamente o Raspberry Pi. Em seguida cria-se no diretório "/home/pi", uma pasta para manter todos arquivos do servidor principal, tais como as imagens da página *html*, a própria página *html*, os arquivos *css* e o certificado digital TLS. Logo após, cria-se um *virtual environment* nesse diretório, para que sirva de *container* para rodar as aplicações do Nginx, uWSGI, Flask. Depois da instalação dessas aplicações e da instalação da biblioteca do sensor DHT22 (PYTHON SOFTWARE FOUNDATION, 2015), são criados e desenvolvidos dois *python scripts*, um para o *framework* Flask e outro que realizará a leitura do sensor DHT22 e do medidor de consumo e armazenará os dados lidos em um arquivo CSV (*Comma-Separated Values*), a página *html*, o arquivo *css*, e os arquivos de configuração do uWSGI e do Nginx (KHONONOV, 2013).

Após isso, já se pode instalar a ferramenta Certbot (ELECTRONIC FRONTIER FOUNDATION, 2018) para gerar certificados da autoridade de certificação Let's Encrypt (INTERNET SECURITY RESEARCH GROUP, 2018) e ativar automaticamente o HTTPS no servidor Nginx. Posteriormente, cria-se o *shell script* "renovação.sh" para realizar a renovação do certificado e envio para um e-mail próprio, pois o certificado é válido por apenas três meses. Criado o *shell script* que realiza a renovação, usa-se o Crontab para agendar a execução do *script* "renovação.sh" de três em três meses.

Após tudo o que foi citado e com a página *html* e seu demais arquivos prontos, agora é o momento em que se inicia o serviço do Nginx e do uWSGI Emperor. Para que não seja preciso iniciar o servidor via comando de terminal, foram criados dois *shell scripts*, um para inicialização do servidor e outro para executar o *python script* do sensor DHT22. Esses *shell scripts* são executados juntamente com a inicialização do Raspbian. Para que eles sejam executados na inicialização do sistema operacional, basta adicionar comandos para a execução desses s*cripts* no arquivo "/etc/rc.local".

O *python script* do *framework*, permite, além de executar as tarefas do servidor r*everse proxy* repassando as requisições recebidas para os servidores de controle, realizar a leitura do arquivo CSV com os dados de temperatura, umidade e de consumo de energia e disponibiliza-los em uma página *html*, juntamente com os demais elementos que controlam os dispositivos. Esse *python script* possui dois tipos de URL (*Uniform Resource Locator*), um que retorna uma página *web* e outro que retorna uma *string* com a ação que foi realizada pelo servidor. Isso viabiliza a comunicação do aplicativo Android a ser desenvolvido com o servidor.

Por fim, realiza-se, no modem roteador, o encaminhamento de portas (*port fowarding)* das portas externas 80 (HTTP) e 443 (HTTPS) para as portas 80 e 443 do servidor de controle. Para maior segurança, o arquivo de configuração do Nginx foi escrito de forma a redirecionar requisições HTTP para HTTPS. Também foi implementado um sistema de *login*.

3.2.2 SERVIDORES DE CONTROLE

A construção dos códigos para os NodeMCUs e Arduinos dos servidores de controle foi realizada na IDE (*Integrated Development Environment*) do Arduino por meio das bibliotecas: Arduino Thread (SEIDEL, 2018), Arduino core for ESP8266 WiFi chip (GROKHOTKOV, 2018) e IRremote Arduino (SHIRRIFF, 2012).

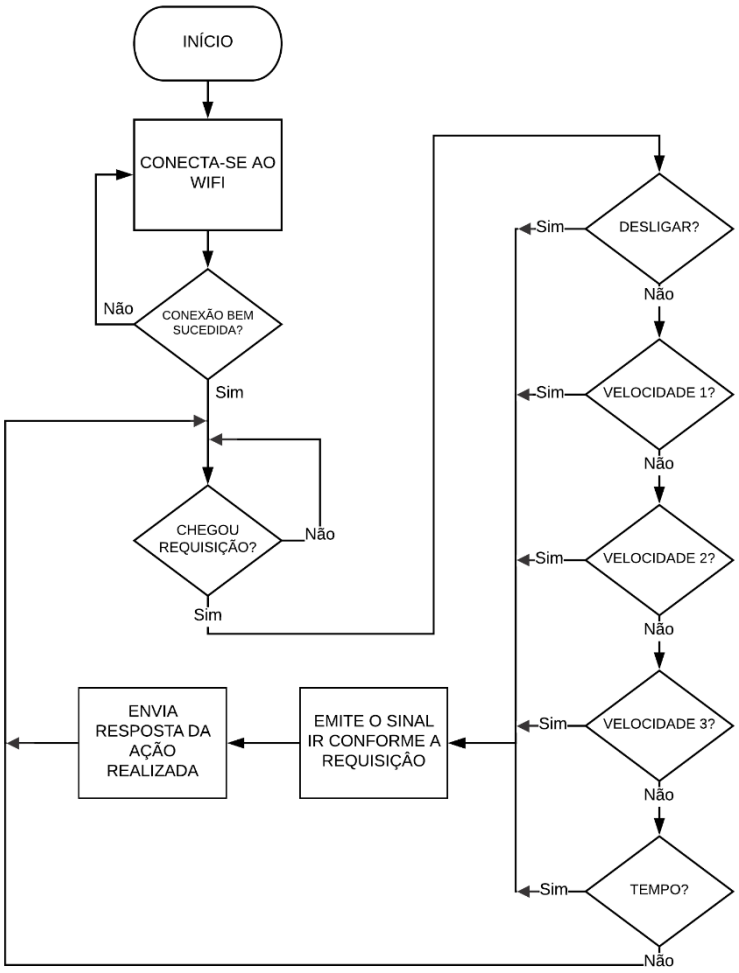

Figura 22 - Fluxograma do servidor de controle do ventilador infravermelho (Própria, 2018)

Para controlar o ventilador infravermelho foi necessário realizar a leitura do sinal IR enviado pelo controle do ventilador para cada botão para que se pudesse reproduzir, exatamente, esses sinais utilizando o Arduino. Isso foi realizado com o código exemplo “IRrecvDumpV2.ino” da biblioteca IRremote Arduino empregando um Arduino e um receptor IR com três terminais. Após isso, foram construídos os códigos para o NodeMCU e para o Arduino conforme o fluxograma da Figura 18. A ideia é rodar somente o servidor no NodeMCU para que ele atenda as requisições feitas através da Internet e, caso essa requisição seja correspondente a uma das cinco funções do ventilador, informar, via comunicação serial, ao

Arduino para emitir o sinal IR, acionando o ventilador. Portanto, o Arduino seria responsável apenas por reproduzir esse sinal, conforme informado pelo NodeMCU.

Os servidores de controle da impressora e do ventilador da sala são muito semelhantes, exceto por ser possível acionar o relê do servidor do ventilador da sala via infravermelho também. Logo, escolheu-se uma das teclas do controle remoto da televisão da sala para controlar o servidor do ventilador e realizou-se a leitura do sinal infravermelho emitido. Em seguida foram construídos os códigos para o NodeMCU e Arduino do servidor do ventilador da sala, conforme a Figura 19, onde o NodeMCU recebe as requisições e informa ao Arduino como proceder. Nesse código do Arduino utilizou-se a biblioteca Arduino Thread para criar um *thread* que verifica a recepção de sinais infravermelhos e aciona o relê caso seja recebido o sinal informado no código. Para o servidor da impressora, a ideia é mais simples. O próprio NodeMCU irá acionar o relê por um de seus GPIOs (*General Purpose Input/Output)*.

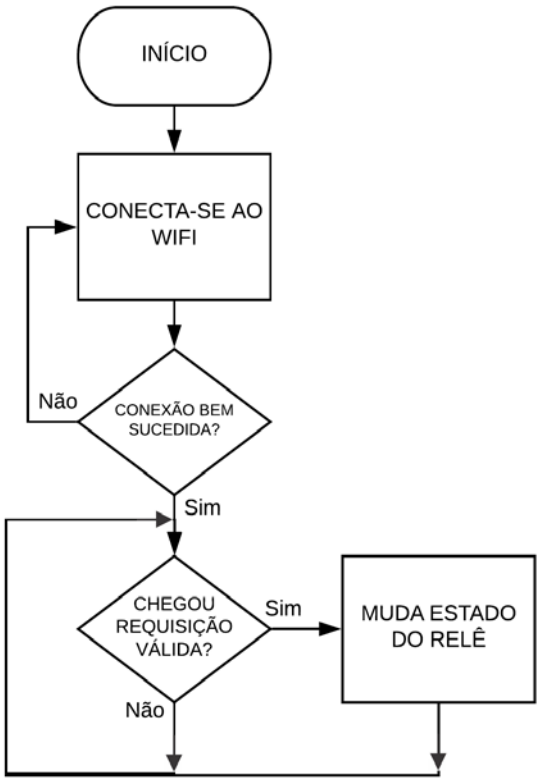

Figura 23 – Fluxograma do servidor de controle do ventilador da sala e da impressora (Própria, 2018)

Por fim, foram desenvolvidos os códigos do servidor de controle do consumo de energia. O código para o NodeMCU é muito similar aos anteriores, entretanto, ao invés de repassar comandos para o Arduino, ele receberá os dados da leitura do sensor de tensão, do sensor de corrente e da estimativa da potência instantânea por meio do Arduino via comunicação serial. Sendo assim, criou-se um *thread* para a comunicação serial no NodeMCU. Entretanto, para o

Arduino não foi feito o mesmo, pois ele realizará a leitura dos sensores ACS712 e P8 (Figura 14), estimará o consumo de energia e, em seguida, os enviará através da comunicação serial ao NodeMCU.

A estimativa da corrente AC foi realizada utilizando a biblioteca Filters, por meio de uma classe chamada RunningStatistics que permite a obtenção da variância, média e desvio padrão de uma variável estatística durante uma janela de tempo (ARDUINO, 2018). Sendo assim, a corrente AC é obtida através do valor do coeficiente angular e do ponto interceptação da reta da regressão linear dos valores de corrente lidos por um wattímetro e dos valores de desvio-padrão dos dados brutos do sensor calculados pela classe RunningStatistics (AUTODESK, 2016). Esses valores são obtidos no processo de calibração do sensor ACS712 que não será abordada neste trabalho. Já para o sensor de tensão P8, a tensão AC foi calculada da seguinte forma:

$$V_{AC} = RawData \times \frac{5}{1024} \times \frac{370}{4.299}$$

Onde RawData é o valor bruto lido do sensor de tensão AC. A primeira fração da equação é o cálculo da resolução do conversor ADC (*Analog-to-Digital Converter*) de 10 bits do Arduino. Já a segunda fração é a sensibilidade do sensor P8. A potência pode ser estimada como a multiplicação da corrente pela tensão.

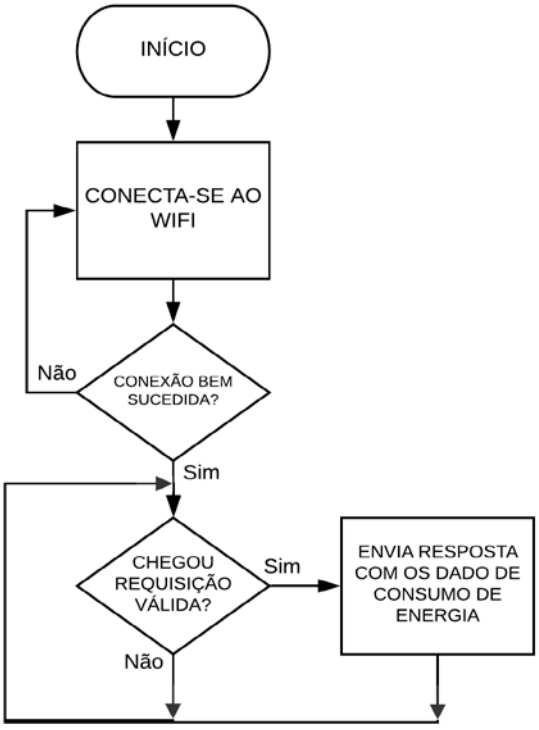

Figura 24 - Fluxograma do servidor de controle estimador do consumo de energia (Própria, 2018)

3.2.3 SERVIDOR DE MONITORAMENTO

Com o sistema operacional MotionEye OS instalado no microSD e o Raspberry Pi conectado via Ethernet ao roteador, configura-se a conexão WiFi do Raspberry Pi ao roteador. Depois disso, com as câmeras ligadas e conectadas à rede WiFi, adicionam-se e configuram-se as câmeras IP ao MotionEye OS por meio do URL. Feito isso, realiza-se o *port forwarding* para que requisições HTTP com porta de destino 9980 sejam redirecionadas do roteador para esse sistema de vídeo vigilância.

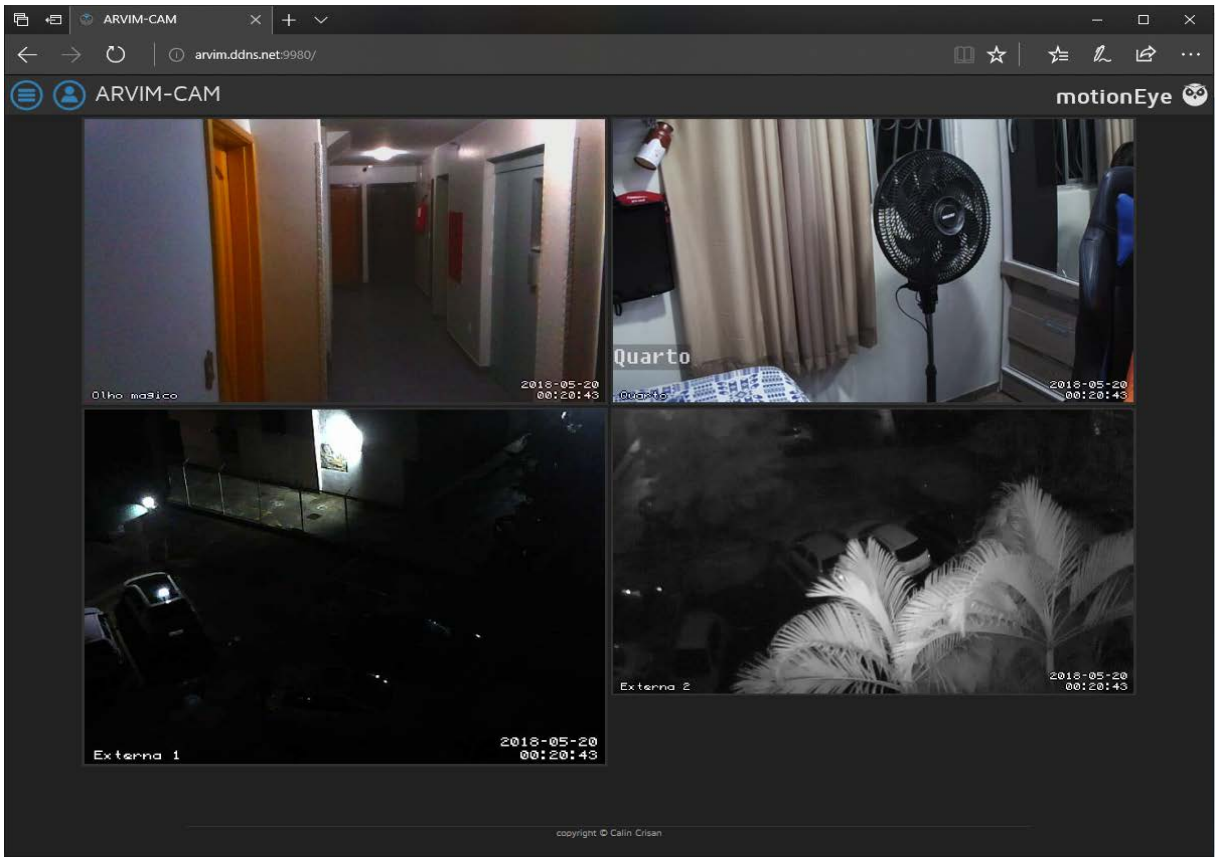

Figura 25 - Imagem do acesso realizado ao servidor de monitoramento via Microsoft Edge (Própria, 2018)

Figura 26 - Captura de tela do aplicativo Onvifer (Própria, 2018)

Não foi implementado o HTTPS por meio do servidor central atuando como *reverse proxy*, pois houve uma queda significativa na taxa de quadros do *streaming* de vídeo das câmeras ao testa-lo com HTTPS. Para visualização das imagens das câmeras adicionadas ao MotionEye OS no smartphone com sistema operacional Android foi utilizado um aplicativo chamado Onvifer, disponível na PlayStore da Google.

3.2.4 APLICATIVO ANDROID ARVIM

O aplicativo ARVIM para *mobiles* com sistema operacional Android foi desenvolvido utilizando a IDE Android Studio. Mediante este aplicativo é possível controlar todo o sistema de automação residencial e obter os dados de temperatura, umidade e consumo de energia. O controle se dá por meio de requisições HTTPS, que só é possível realizar com a integração do certificado digital obtido na configuração do servidor central.

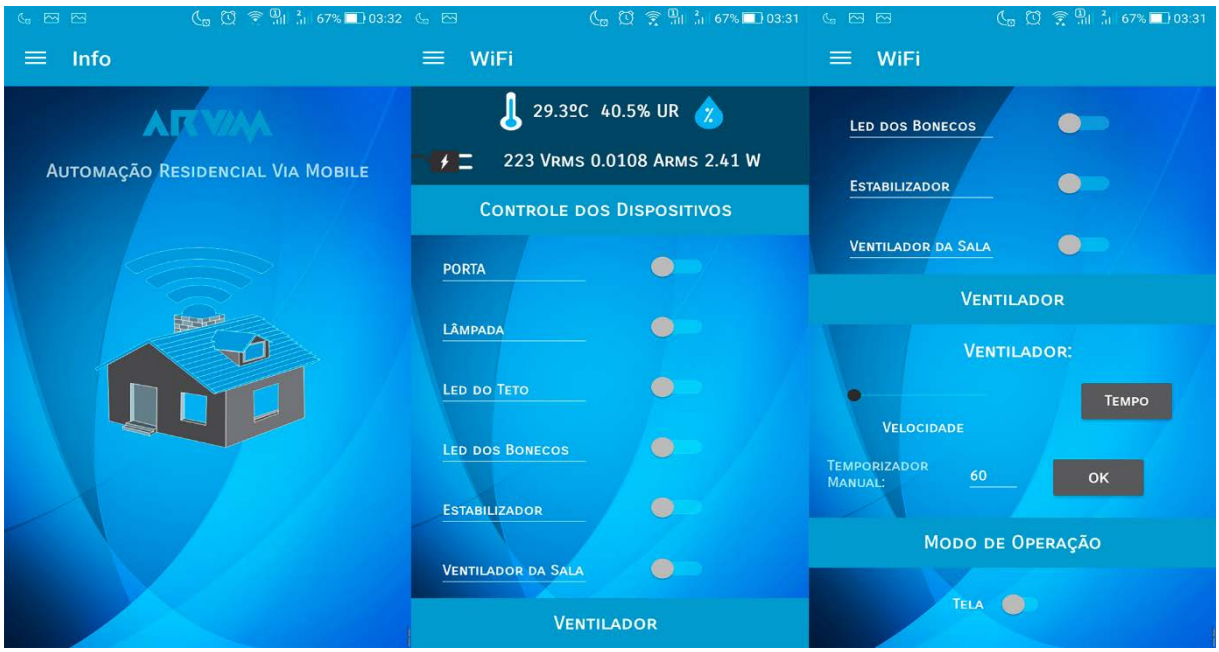

Figura 27 – Capturas de tela do aplicativo ARVIM em funcionamento (Própria, 2018)

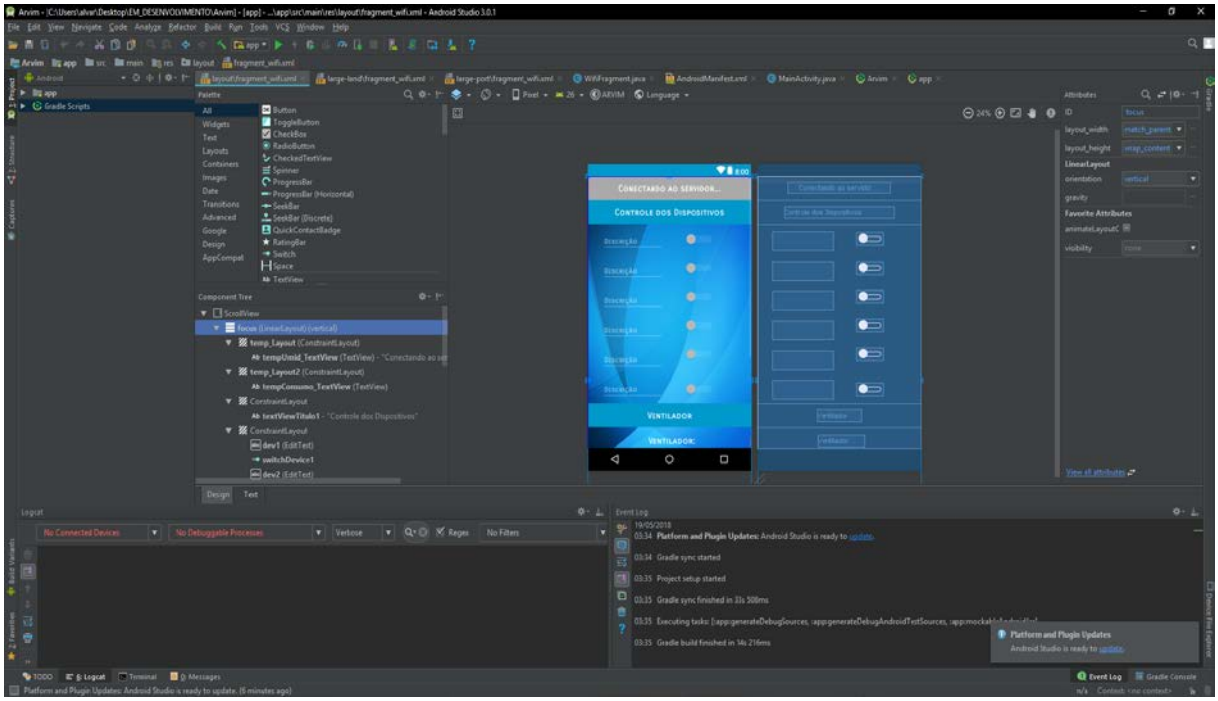

Figura 28 - Captura de tela do desenvolvimento do aplicativo ARVIM na IDE Android Studio (Própria, 2018)

Para maior segurança, foi implementada a autenticação de usuário no aplicativo. Portanto, o usuário precisa configurar, na sessão configurações, o nome de domínio, porta, *login* e senha. Como há persistência de dados no aplicativo, o usuário necessita fazê-lo apenas uma vez.

A instalação do ARVIM é feita através do Android Studio e para que isso ocorra é necessário habilitar o modo desenvolvedor no dispositivo a ser instalado. Esse aplicativo funciona em

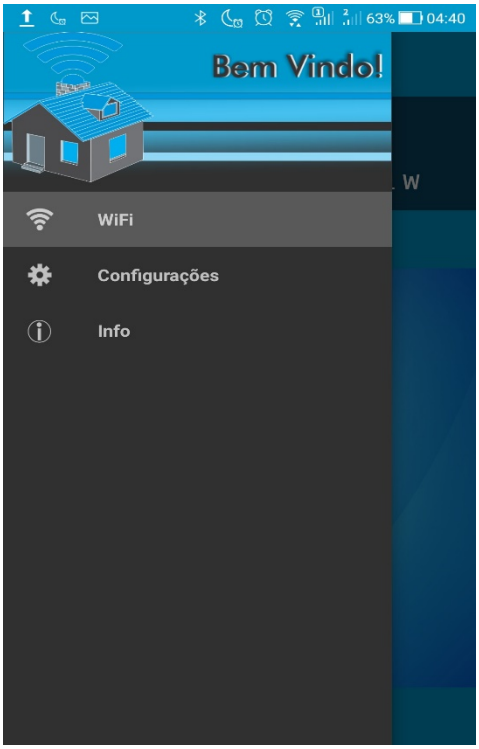

Figura 29 - Captura de tela do *Navigation Drawer* (Própria, 2018)

mobiles Android que possuam versão igual ou superior ao Android 4.4, denominado Kit Kat. Como o certificado digital é renovado de três em três meses, é necessário reinstalar o aplicativo a cada três meses com o novo certificado para que continue funcionando.

A interface de usuário foi desenvolvida de forma a ser a mais amigável possível com implementação do *Navigation Drawer* que cria uma gaveta de navegação (GOOGLE DEVELOPERS, 2018) e *Shared Preferences* que promove a persistência de dados (GOOGLE DEVELOPERS, 2018). Além disso, o usuário pode personalizar o nome dos dispositivos controlados modificando o texto presente nas caixas de texto ao lado dos botões.

4 ENSAIOS

Até que se chegasse à composição de *hardware* e *software* descritos nesse trabalho, houve versões anteriores com disposições diferentes. Na primeira versão, empregou-se um Arduino Uno, um módulo Bluetooth HC-06, um módulo relê de oito canais, um sensor de temperatura LM35DZ e um display LCD 16x02. Assim, era possível controlar, via Bluetooth, um fecho eletromagnético para abrir uma porta e uma lâmpada. O display mostrava a temperatura ambiente fornecida pelo sensor LM35DZ.

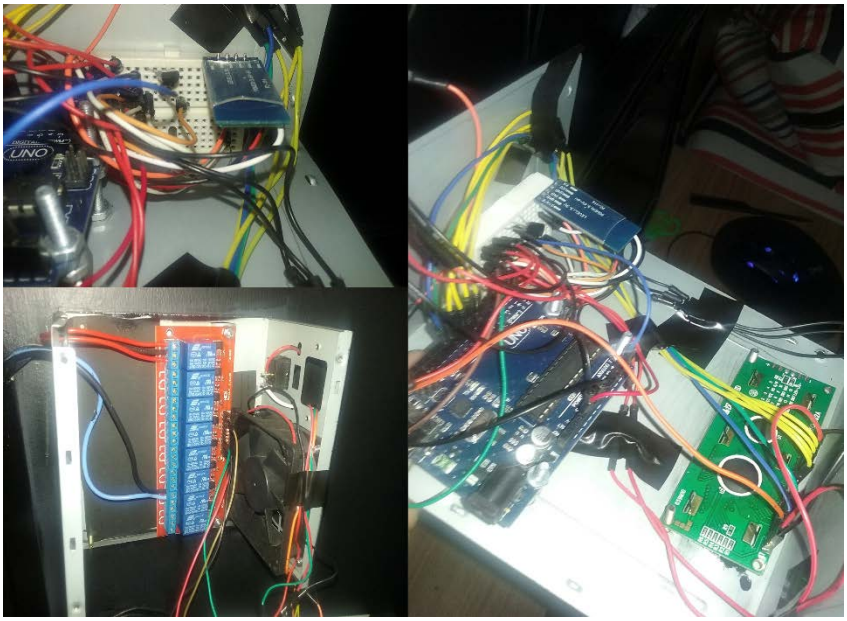

Figura 30 - Primeira versão do ARVIM (Própria, 2018)

Apesar de cumprir sua função, essa primeira versão mostrou-se muito limitada e pouco prática para o usuário. A conexão com o módulo Bluetooth mostrou-se instável, muitas vezes não sendo possível conectar-se ao módulo HC-06. Além disso, apresentava travamento dos relês e erro na comunicação com o display LCD, conforme ilustrado na Figura 27.

Figura 31- Erro apresentado no display LCD da primeira versão do ARVIM (Própria, 2018)

O aplicativo para dispositivos *mobile* dessa primeira versão foi desenvolvido no MIT App Inventor (MIT, 2018) e sua interface de usuário era a mais simplista possível, possuindo apenas alguns botões para conectar-se ao ARVIM, via Bluetooth, e outros dois botões para controla-los. Um dos destaques desse aplicativo era poder controlar o sistema de automação via comandos de voz.

Figura 33- Foto do Arduino Mega e do módulo ARVIM (Própria, 2018)

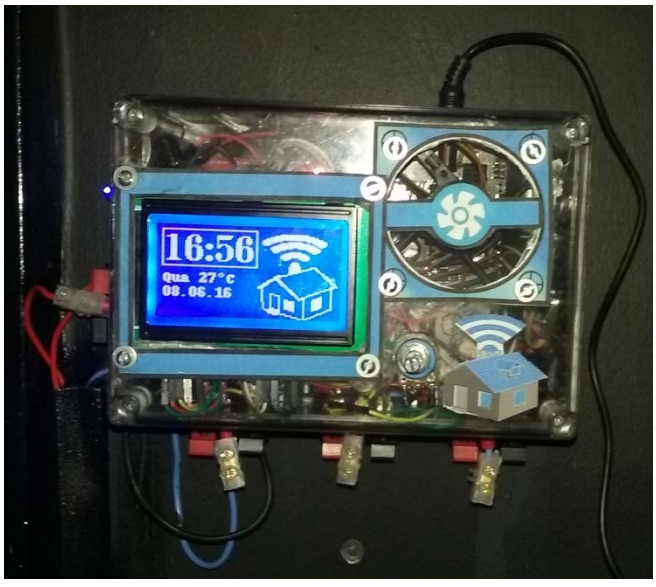

Figura 32 - Foto da segunda versão do ARVIM (Própria, 2018)

Na segunda versão, o ESP8266 ESP-01 foi incorporado ao projeto sendo possível controlar os relês através da Internet. Ao invés de um display LCD 16x02, foi utilizado um display gráfico GLCD 128x64 possibilitando ilustrar desenhos, efeitos gráficos e caracteres personalizados. Trocou-se o Arduino Uno por um Arduino Mega e o módulo relê de 8 canais por um de 4 canais.

Foram adicionados dois reguladores de tensão, LM2576T 3.3V e LM2576T 5V, para alimentação dos componentes. Adicionou-se, também, um módulo *tiny* RTC (*Real Time Clock*) DS1307 e um LED emissor de infravermelho. Para melhor integração dos componentes, foi desenvolvido um módulo para o Arduino Mega conforme ilustrado na Figura 28. Essas mudanças permitiram o controle do ventilador infravermelho e a implementação de um relógio mostrado no display gráfico. O aplicativo ARVIM dessa versão foi desenvolvido no Android Studio e possuía uma interface de usuário mais amigável que a anterior, permitindo que o relógio fosse ajustado via Bluetooth.

Após realizar os testes, foram descobertos diversos problemas na segunda versão. O ESP8266 ESP-01 apresentava uma série de travamentos e lentidão para responder às requisições. O módulo RTC não persistia os dados do relógio após desligar o equipamento e religa-lo, mesmo utilizando uma bateria CR2032. Por último, o problema de travamento dos relês continuava presente.

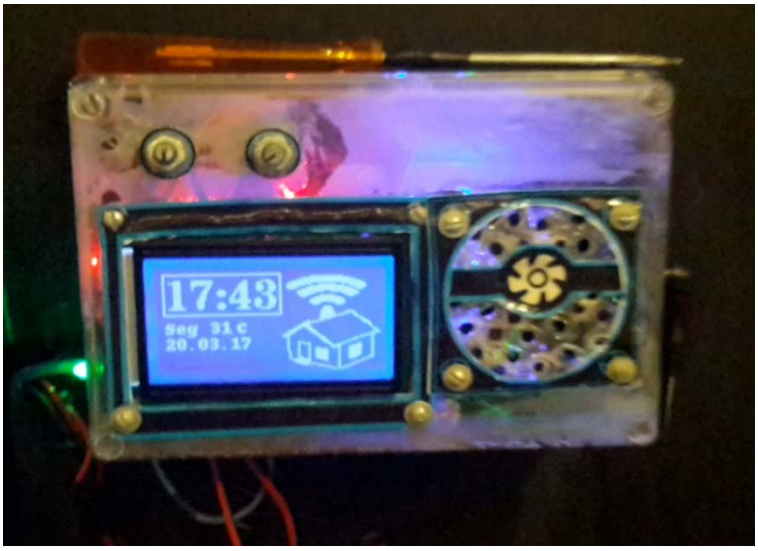

Figura 34- Foto da terceira versão do ARVIM (Própria, 2018)

Tendo esses problemas em vista, foi desenvolvida a terceira versão do ARVIM. Nessa versão foram substituídos o Arduino Mega por um Arduino Nano, o ESP8266 ESP-01 pelo NodeMCU 1.0 e o módulo *tiny* RTC DS1307 pelo módulo RTC DS3231. A temperatura do ambiente agora era obtida pelo módulo RTC e não mais pelo sensor LM35DZ. Embora corrigidos os problemas de travamento ao atender requisições com o ESP8266 ESP-01, da não persistência de dados do RTC e do travamento dos relês, havia a questão de o NodeMCU não ser robusto o suficiente para hospedar uma página *web* bem elaborada e da impossibilidade da implementação do HTTPS. Outro problema era a centralização do sistema em um só aparelho. Isso dificultava e limitava o controle de dispositivos que estivessem longe do ARVIM, pois o limite de

dispositivos a serem controlados correspondia ao número de canais do módulo relê e dispositivos infravermelho próximos. Ademais, seria preciso conectar o dispositivo a ser controlado e o relê com cabos.

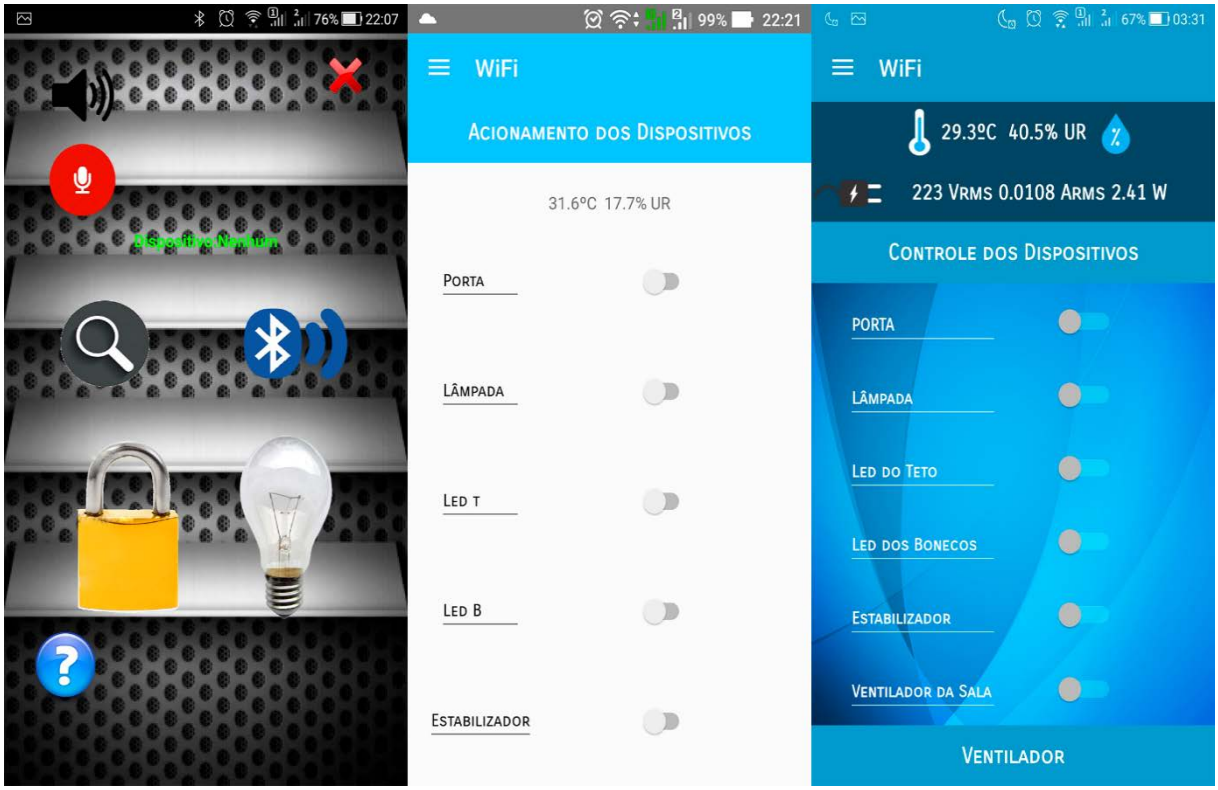

Figura 35 - Evolução do aplicativo *mobile* ARVIM (Própria, 2018)

Por fim, foi construída a última versão do sistema de automação residencial ARVIM. Para resolver os problemas de hospedagem de página e de segurança via HTTPS, foi utilizado o Raspberry Pi 3 em um servidor central atuando como servidor *reverse proxy*, conforme visto nas seções 3.1.1 e 3.2.1. Também foram criados servidores de controle, utilizando o NodeMCU, para serem acoplados nos dispositivos a serem controlados, visto nas seções 3.1.2 e 3.2.2. Dessa forma, possibilitou-se controlar dispositivos mais distantes.

Essas mudanças proporcionaram uma grande melhora no desempenho, maior estabilidade e segurança no atendimento das requisições via Internet. Outras mudanças foram o uso do sensor de temperatura e umidade DHT22, a retirada do suporte Bluetooth, a adição de um servidor de controle que monitora o consumo de energia e a criação de um sistema de vigilância com câmeras IP. Esse sistema de vigilância detecta movimento, grava o evento e o armazena por um tempo personalizável. Também permite o envio de uma notificação ao usuário do evento ocorrido, caso seja assim configurado.

Ainda assim, foram notados pequenos problemas no sistema. Há o problema intermitente de travamento no modem roteador, fornecido pelo provedor de internet, que impede que *port forwarding* seja realizado, sendo necessário reinicia-lo. Também há um problema intermitente na emissão do sinal infravermelho para o ventilador, sendo necessário emiti-lo mais de uma vez para efetuar a ação.

Figura 36 - Acesso à página *web* do sistema ARVIM via Microsoft Edge (Própria, 2018)

Conforme visto na Figura 31, o estado dos dispositivos é mostrado na página *web* do sistema. Entretanto, esse não é o estado real do dispositivo e sim o estado relê, exceto para o servidor da impressora. Isso ocorre devido aos dos dispositivos serem ligados em paralelo com um interruptor *three way*. Também é importante ressaltar que nenhum estado é mostrado no aplicativo *mobile*. Uma boa proposição de trabalho futuro seria implementar a detecção do estado real do dispositivo e mostra-lo na página *web* e no aplicativo.

No total, esse sistema de automação controla sete aparelhos eletrônicos: duas fitas de LED, uma lâmpada, uma tranca elétrica, um ventilador infravermelho, uma impressora e um ventilador comum. O sistema de vigilância possui três câmeras IP, uma interna e duas externas, e uma *Camera Module V2*, totalizando quatro câmeras.

4.1 CUSTOS ESTIMADOS

Apesar de um aumento de 1215,11% no custo entre a primeira e a última versão conforme a Tabela 1, a última apresenta um aumento enorme no quesito estabilidade, segurança, praticidade e diversidade de recursos.

COMPARATIVO ENTRE VERSÕES ARVIM		
VERSÃO	PREÇO	AUMENTO DO CUSTO EM RELAÇÃO À VERSÃO ANTERIOR (%)
1	R$ 209,38	-
4	R$ 2.753,57	1215,11%

Tabela 1 - Comparativo do custo entre a primeira e última versão do ARVIM (Própria, 2018)

Nota-se da Tabela 2 que a terceira versão foi uma evolução muito positiva em relação à segunda versão, afinal, houve uma redução no custo e foram sanados boa parte dos problemas da versão anterior.

COMPARATIVO ENTRE VERSÕES ARVIM		
VERSÃO	PREÇO	AUMENTO DO CUSTO EM RELAÇÃO À VERSÃO ANTERIOR (%)
1	R$ 209,38	-
2	R$ 448,04	113,98%
3	R$ 441,60	-1,44%
4	R$ 2.753,57	523,54%

Tabela 2 - Comparativo do custo dos componentes necessários entre as versões do ARVIM (Própria, 2018)

Na Tabela 3, estão apresentados todos os gastos realizados com componentes da quarta versão do ARVIM. Todos esses custos foram arcados por conta própria. Vale ressaltar que não foram incluídos o valor da mão de obra para instalação desse sistema e valores dos materiais necessários para adaptação à estrutura do domicílio, como cabos e canaletas. O detalhamento dos custos das outras versões encontra-se no Apêndice C.

ARVIM VERSÃO 4			
Componentes	**Undidades**	**Preço Unitário**	**Preço**
ARDUINO NANO	1	R$ 23,90	R$ 23,90
ARDUINO PRO MINI 328 - 5V/16MHZ	2	R$ 19,90	R$ 39,80
CABO DE FORCA PARALELO - PLUG BIPOLAR MACHO	3	R$ 0,89	R$ 2,67
CAIXA PATOLA PB 055	2	R$ 35,85	R$ 71,70
CAIXA PLÁSTICA MULTIUSO SIBRATEC 10X10X8 CM	1	R$ 40,00	R$ 40,00
CAIXA PLÁSTICA MULTIUSO SIBRATEC 20X15X8 CM	1	R$ 70,00	R$ 70,00
CAIXA PLÁSTICA MULTIUSO SIBRATEC 20X20X8 CM	1	R$ 50,00	R$ 50,00
CAIXA PLÁSTICA PATOLA PB-107	1	R$ 40,90	R$ 40,90
CAIXA PLASTICA PATOLA PB114/2 PRETA	1	R$ 42,94	R$ 42,94
CAMERA IP EXTERNA SEM FIO WIRELESS FULLSEC	1	R$ 239,99	R$ 239,99
CAMERA IP WIRELESS EXTERNA	1	R$ 227,90	R$ 227,90
CAMERA IP WIRELESS INTERNA COM MOVIMENTO	1	R$ 114,90	R$ 114,90
CAMERA V2 MODULE RASPBERRY PI	1	R$ 128,76	R$ 128,76
CASE PARA CÂMERA RASPBERRY	1	R$ 35,80	R$ 35,80
CHAVE GANGORRA MINI 2T	3	R$ 0,79	R$ 2,37
FABRICAÇÃO DAS PCBS	1	R$ 70,00	R$ 70,00
FECHO ELÉTRICO HDL FECHADURA FEC-91LA LONGO TRINCO AJUSTÁVEI	1	R$ 103,89	R$ 103,89
FONTE TRA 400	1	R$ 76,70	R$ 76,70
KIT BÁSICO RASPBERRY PI 3 PI3 - SD SANDISK 16GB CLASS 10	2	R$ 265,00	R$ 530,00
LCD 7 OFFICIAL TOUCH SCREEN COM CASE PARA RASPBERRY PI 3 PI3	1	R$ 500,00	R$ 500,00
MINI FONTE 100/240VAC PARA 5VDC (HLK-PM01)	3	R$ 25,39	R$ 76,17
MÓDULO DE RELÉS 5V - 4 CANAIS	1	R$ 22,90	R$ 22,90
MÓDULO RELÉ 5V/1CANAL	2	R$ 5,90	R$ 11,80
MÓDULO SENSOR DE CORRENTE ELÉTRICA ACS712 - 5A	1	R$ 16,99	R$ 16,99
NODEMCU 1.0 ESP 12-E	3	R$ 42,90	R$ 128,70
PLUGS TOMADA FÊMEA	2	R$ 5,00	R$ 10,00
RECEPTOR INFRA-VERMELHO (IR) CODIFICADO 3T	1	R$ 0,99	R$ 0,99
SENSOR DE TENSÃO AC P8	1	R$ 37,90	R$ 37,90
SENSOR DE UMIDADE E TEMPERATURA - DHT22	1	R$ 35,90	R$ 35,90
TOTAL			**R$ 2.753,57**

Tabela 3 - Tabela de custo dos componentes do ARVIM (Própria, 2018)

5 CONCLUSÃO

O sistema de automação residencial ARVIM não se limitou apenas à fase de prototipagem, mas a sua implementação, de fato, no mundo real. Seu funcionamento é estável e cumpre exatamente o seu propósito. As versões anteriores foram fundamentais para que se chegasse à construção da disposição da versão atual. A interdisciplinaridade foi um dos pilares para que esse trabalho fosse possível, pois foram necessários a execução e o aprendizado de tarefas de diferentes áreas.

O maior empecilho na implementação de um sistema como esse é a adaptação necessária para residências que não foram projetadas para serem automatizadas. Embora houvesse essa dificuldade, o sistema foi implantado com êxito através de modificações simples na estrutura da residência, tal como o uso de uma instalação elétrica aparente.

Sabendo que o custo de um projeto de automação residencial varia entre R$10.000,00 e R$30.000,00 (G1, 2013), atingiu-se uma redução de 72,46% nos custos, considerando o custo mínimo. Vale ressaltar que nesse trabalho não foram envolvidos custos de mão de obra e que projetos são altamente personalizáveis, influenciando diretamente nos custos.

O acesso externo ao sistema ARVIM foi possível graças ao serviço gratuito de DDNS fornecido pela empresa No-IP (2018), proporcionando um acesso rápido. Para atingir um sistema ainda mais estável, seria necessário utilizar um modem roteador mais robusto para realizar as operações do *port forwarding.*

Proposições interessantes para trabalhos futuros seriam a detecção do real estado do dispositivo e não o estado relê e a extensão do controle de dispositivos para mais cômodos da casa. Outra boa proposição seria a utilização de um sensor mais preciso para a medição de tensão AC e obter uma calibração mais acurada para o sensor ACS712, por meio do uso de um wattímetro.

Quanto ao aplicativo *mobile*, uma interface ainda mais amigável seria necessária conforme o sistema fosse alcançando outros cômodos, dispondo de recursos como *widgets* personalizáveis e promovendo maior customização. Por fim, o desenvolvimento da compatibilidade do aplicativo tanto para sistemas operacionais Android quanto para sistemas IOS da empresa Apple.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, C. D. Câmeras IP - 1ª parte. **institutocftv.com.br**, 2015. Disponivel em: <http://www.institutocftv.com.br/cameras-ip.html>. Acesso em: 28 abr. 2018.

ANICAS, M. OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs. **digitalocean.com**, 2014. Disponivel em: <https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs>. Acesso em: 10 ago. 2017.

ARDUINO. A realtime digital signal processing (DSP) library for Arduino. **playground.arduino.cc**, 2018. Disponivel em: <https://playground.arduino.cc/Code/Filters>. Acesso em: 19 maio 2018.

AURESIDE. Automação Residencial: demanda na Construção Civil. **aureside.org.br**, 2015. Disponivel em: <http://www.aureside.org.br/noticias/automacao-residencial--demanda-na-construcao-civil>. Acesso em: 22 abr. 2018. Citado na página 5.

AUTODESK. SIMPLIFIED ARDUINO AC CURRENT MEASUREMENT USING ACS712 HALL EFFECT SENSOR. **instructables.com**, 2016. Disponivel em: <https://www.instructables.com/id/Simplified-Arduino-AC-Current-Measurement-Using-AC/>. Acesso em: 19 95 2918.

BIYEE SCITECH. Monitor de Câmera IP ONVIF (Onvifer). **play.google.com**, 2018. Disponivel em: <https://play.google.com/store/apps/details?id=net.biyee.onvifer&hl=pt_BR>. Acesso em: 28 abr. 2018.

CARDOSO, M. O que é DDNS. **interside.org**, 2015. Disponivel em: <http://www.interside.org/2015/11/o-que-e-ddns.html>. Acesso em: 7 ago. 2017.

CISCO. Network Address Translation (NAT) FAQ. **cisco.com**, 2014. Disponivel em: <https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/26704-nat-faq-00.html>. Acesso em: 7 ago. 2017.

CNXSOFT. Raspberry Pi 3 Board is Powered by Broadcom BCM2837 Cortex A53 Processor, Sells for $35. **cnx-software.com**, 2016. Disponivel em: <https://www.cnx-software.com/2016/02/29/raspberry-pi-3-board-is-powered-by-broadcom-bcm2827-cortex-a53-processor-sells-for-35/>. Acesso em: 28 abr. 2018. il.color.

CRISAN, C. What is motionEyeOS? **github.com**, 2017. Disponivel em: <https://github.com/ccrisan/motioneyeos/wiki>. Acesso em: 30 jul. 2017.

EBY, P. J. PEP 333 -- Python Web Server Gateway Interface v1.0. **python.org**, 2003. Disponivel em: <https://www.python.org/dev/peps/pep-0333/>. Acesso em: 28 abr. 2018.

ELECTRONIC FRONTIER FOUNDATION. certbot. **github.com**, 2018. Disponivel em: <https://github.com/certbot/certbot>. Acesso em: 18 maio 2018.

ELEKTRONICS. Types of Arduino Board and Selection Guide. **elektronicsweb.wordpress.com**, 2017. Disponivel em: <https://elektronicsweb.wordpress.com/2017/01/27/types-of-arduino-board-and-selection-guide/>. Acesso em: 28 abr. 2018. il. color.

ESPRESSIF. ESP8266. **espressif.com**, 2018. Disponivel em: <https://www.espressif.com/en/products/hardware/esp8266ex/overview>. Acesso em: 28 abr. 2018.

G1. Empresa fatura R$ 1 milhão com serviços de automação residencial. **g1.globo.com**, 2013. Disponivel em: <http://g1.globo.com/economia/pme/noticia/2013/05/empresa-fatura-r-1-milhao-com-servicos-de-automacao-residencial.html>. Acesso em: 21 abr. 2018. Citado na página 5.

GOOGLE DEVELOPERS. Conheça o Android Studio. **developer.android.com**, 2017. Disponivel em: <https://developer.android.com/studio/intro/index.html>. Acesso em: 10 ago. 2017.

GOOGLE DEVELOPERS. Segurança com HTTPS e SSL. **developer.android.com**, 2017. Disponivel em: <https://developer.android.com/training/articles/security-ssl.html?hl=pt-br>. Acesso em: 11 ago. 2017.

GOOGLE DEVELOPERS. Criação de uma gaveta de navegação. **developer.android.com**, 2018. Disponivel em: <https://developer.android.com/training/implementing-navigation/nav-drawer>. Acesso em: 19 maio 2018.

GOOGLE DEVELOPERS. SharedPreferences. **developer.android.com**, 2018. Disponivel em: <https://developer.android.com/reference/android/content/SharedPreferences>. Acesso em: 19 maio 2018.

GROKHOTKOV, I. Arduino core for ESP8266 WiFi chip. **github.com**, 2018. Disponivel em: <https://github.com/esp8266/Arduino>. Acesso em: 18 maio 2018.

HAFER, C. What's the difference between a reverse proxy and forward proxy? **quora.com**, 2014. Disponivel em: <https://www.quora.com/Whats-the-difference-between-a-reverse-proxy-and-forward-proxy/answer/Cameron-Hafer?share=a84639b7&srid=icOu>. Acesso em: 27 abr. 2018. il. color.

HAMAN, R. iOS, Android e Windows Phone: números dos gigantes comparados [infográfico]. **tecmundo.com.br**, 2014. Disponivel em: <https://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm>. Acesso em: 28 abr. 2018. il. color.

HAN, S. Let's encrypt with Raspberry Pi. **blog.meinside.pe.kr**, 2015. Disponivel em: <https://blog.meinside.pe.kr/Lets-Encrypt-with-Raspberry-Pi/>. Acesso em: 18 maio 2018.

HELGESCHNEIDER. Tut02: ESP8266 mit Arduino – Externe LED mit Vorwiderstand schalten. **wvssiot.wordpress.com**, 2017. Disponivel em: <https://wvssiot.wordpress.com/2017/02/03/tut01-esp8266-mit-arduino-externe-led-mit-vorwiderstand-schalten/>. Acesso em: 28 abr. 2018. il.color.

HU INFINITO. Sensor de Umidade e Temperatura - DHT22. **huinfinito.com.br**, 2018. Disponivel em: <http://www.huinfinito.com.br/sensores/470-sensor-de-umidade-e-temperatura-dht22.html?search_query=dht22&results=1>. Acesso em: 16 maio 2018.

IETF. RFC 1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. **ietf.org**, 1987. Disponivel em: <https://www.ietf.org/rfc/rfc1035.txt>. Acesso em: 27 abr. 2018.

INTERNET SECURITY RESEARCH GROUP. Let's Encrypt. **letsencrypt.org**, 2018. Disponivel em: <https://letsencrypt.org/>. Acesso em: 18 maio 2018.

ITEAD. Home>SmartHome. **itead.cc**, 2017. Disponivel em: <https://www.itead.cc/smart-home.html>. Acesso em: 24 abr. 2018. Citado nas páginas 5 e 6.

JSCAPE. Forward Proxy vs Reverse Proxy. **jscape.com**, 2012. Disponivel em: <http://www.jscape.com/blog/bid/87783/Forward-Proxy-vs-Reverse-Proxy>. Acesso em: 27 abr. 2018.

KHONONOV, V. Serving Flask with Nginx. **vladikk.com**, 2013. Disponivel em: <http://vladikk.com/2013/09/12/serving-flask-with-nginx-on-ubuntu/>. Acesso em: 9 ago. 2017.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet:** Uma Abordagem Top-Down. 5ª. ed. São Paulo: Pearson Addison-Wesley, 2009. 72,439,521-524. p.

MAKAI, M. Flask. **fullstackpython.com**, 2012. Disponivel em: <https://www.fullstackpython.com/flask.html>. Acesso em: 9 ago. 2017.

MARKETSANDMARKETS. Home Automation System Market worth 79.57 Billion USD by 2022. **marketsandmarkets.com**, 2017. Disponivel em: <https://www.marketsandmarkets.com/PressReleases/home-automation-control-systems.asp>. Acesso em: 22 abr. 2018. Citado na página 5.

MCMANUS, S.; COOK, M. **Raspberry Pi for Dummies**. 1ª. ed. Hoboken: John Wiley & Sons, Inc., 2013. 7-19 p.

MCROBERTS, M. **Arduino Básico**. 2ª. ed. São Paulo: Novatec, 2015. 22-25 p.

MICROSOFT. What is TLS/SSL. **docs.microsoft.com**, 2009. Disponivel em: <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc784450(v=ws.10)>. Acesso em: 8 ago. 2017.

MICROSOFT. How TLS/SSL. **docs.microsoft.com**, 2015. Disponivel em: <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc783349(v=ws.10)>. Acesso em: 9 ago. 2017.

MIT. Anyone Can Build Apps That Impact the World. **appinventor.mit.edu**, 2018. Disponivel em: <http://appinventor.mit.edu/explore/about-us.html>. Acesso em: 19 maio 2017.

MITCHELL, B. What Does Dynamic DNS Mean? **lifewire.com**, 2018. Disponivel em: <https://www.lifewire.com/definition-of-dynamic-dns-816294>. Acesso em: 3 mar. 2018.

NGINX. nginx. **nginx.org**, 2017. Disponivel em: <http://nginx.org/en/?_ga=2.20385980.507464465.1503116102-668628875.1503116102>. Acesso em: 10 ago. 2017.

NGINX. WHAT IS NGINX? **nginx.com**, 2017. Disponivel em: <https://www.nginx.com/resources/glossary/nginx/>. Acesso em: 9 ago. 2017.

NO-IP. Acesso Remoto fácil. **noip.com**, 2018. Disponivel em: <https://www.noip.com/pt-BR>. Acesso em: 18 maio 2018.

POLLOCK, G. Apache vs Nginx. **upguard.com**, 2017. Disponivel em: <https://www.upguard.com/articles/apache-vs-nginx>. Acesso em: 28 abr. 2018.

PYTHON SOFTWARE FOUNDATION. Adafruit_Python_DHT 1.1.2. **pypi.org**, 2015. Disponivel em: <https://pypi.org/project/Adafruit_Python_DHT/1.1.2/#description>. Acesso em: 18 maio 2018.

RASPBERRY PI FOUNDATION. Camera Module V2. **raspberrypi.org**, 2018. Disponivel em: <https://www.raspberrypi.org/products/camera-module-v2/>. Acesso em: 18 maio 2018. il.color.

RASPBERRY PI FOUNDATION. Raspberry Pi Touch Display. **raspberrypi.org**, 2018. Disponivel em: <https://www.raspberrypi.org/products/raspberry-pi-touch-display/>. Acesso em: 16 maio 2018.

RASPBIAN. Welcome to Raspbian. **raspbian.org**, 2017. Disponivel em: <https://www.raspbian.org/>. Acesso em: 27 abr. 2018.

REALVNC. Perguntas frequentes. **realvnc.com**, 2018. Disponivel em: <https://www.realvnc.com/pt/connect/download/vnc/>. Acesso em: 28 abr. 2018.

RONACHER, A. Flask web development, one drop at a time. **flask.pocoo.org**, 2010. Disponivel em: <http://flask.pocoo.org/>. Acesso em: 10 ago. 2017.

SEIDEL, I. ArduinoThread. **github.com**, 2018. Disponivel em: <https://github.com/ivanseidel/ArduinoThread>. Acesso em: 18 maio 2018.

SHIRRIFF, K. IRremote Arduino Library. **github.com**, 2012. Disponivel em: <https://github.com/z3t0/Arduino-IRremote>. Acesso em: 17 maio 2018.

STATISTA. Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017. **statista.com**, 2018. Disponivel em: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>. Acesso em: 28 abr. 2018. il. color.

SZDOIT. User Manual for ESP12E Dev Kit. **smartarduino.gitbooks.io**, 2015. Disponivel em: <https://smartarduino.gitbooks.io/user-manual-for-esp-12e-devkit/content/chapter1.html>. Acesso em: 9 ago. 2017.

TECHOPEDIA. Port Forwarding. **techopedia.com**, 2017. Disponivel em: <https://www.techopedia.com/definition/4057/port-forwarding>. Acesso em: 27 abr. 2018.

UWSGI. The uWSGI project. **uwsgi-docs.readthedocs.io**, 2016. Disponivel em: <https://uwsgi-docs.readthedocs.io/en/latest/>. Acesso em: 28 abr. 2018.

YEELINK. Yeelight LED Bulb (Color). **yeelight.com**, 2017. Disponivel em: <https://www.yeelight.com/en_US/product/wifi-led-c>. Acesso em: 25 abr. 2018. Citado nas páginas 5 e 6.

ZELENOVSKY, R.; MENDONÇA, A. **Eletrônica Digital**. 2ª. ed. Rio de Janeiro: MZ, 2007. 9,10 p.

APÊNDICE A – ESQUEMÁTICO DAS MONTAGENS

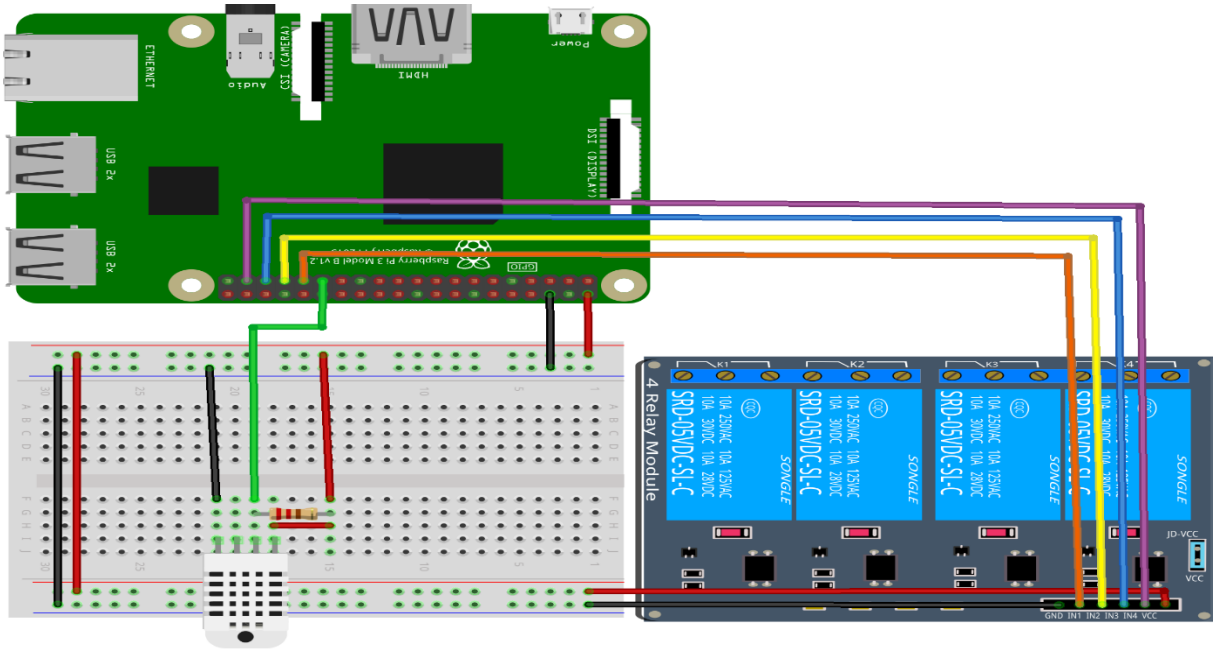

Figura 37 - Esquemático da montagem do servidor central (Própria, 2018)

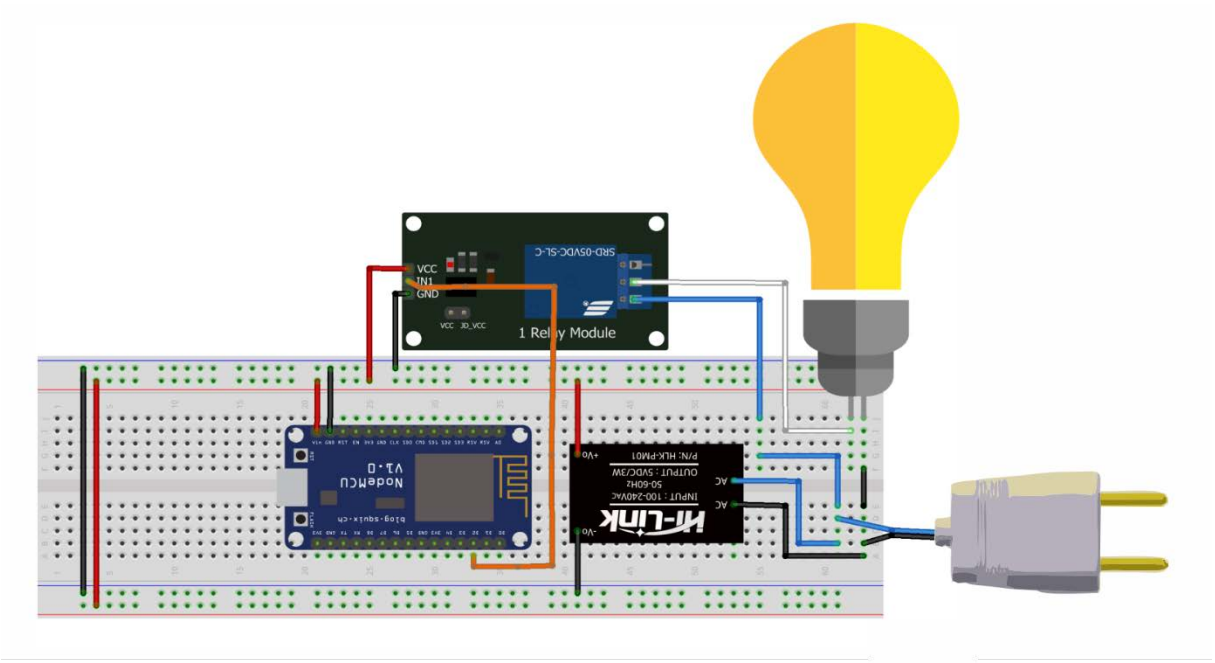

Figura 38 - Esquemático da montagem do servidor de controle da impressora (Própria, 2018)

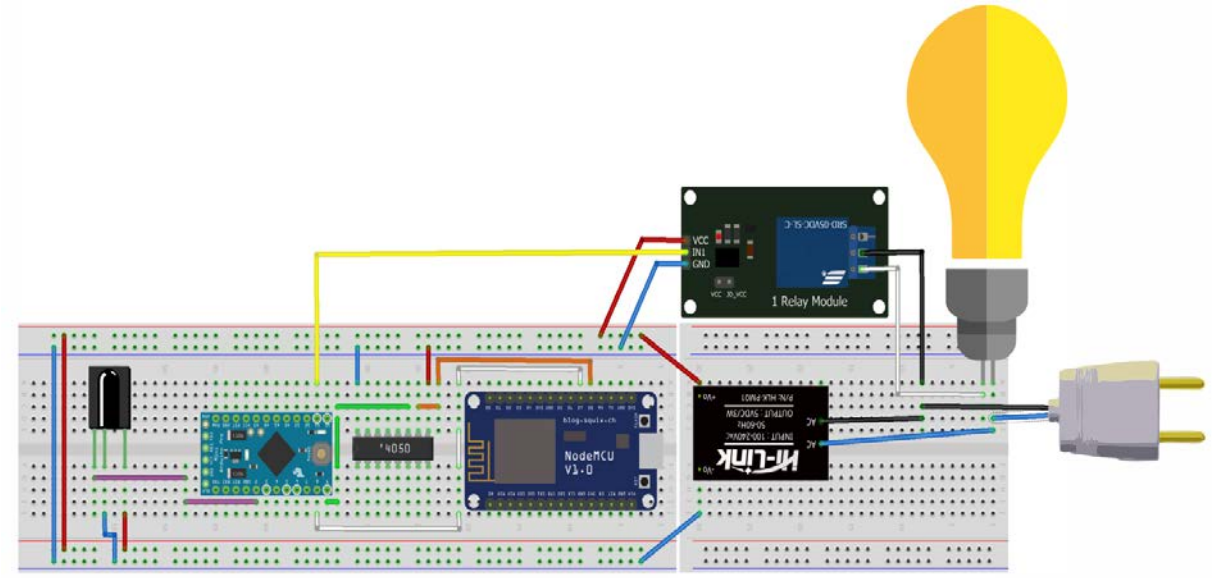

Figura 40 - Esquemático da montagem do servidor de controle do ventilador da sala (Própria, 2018)

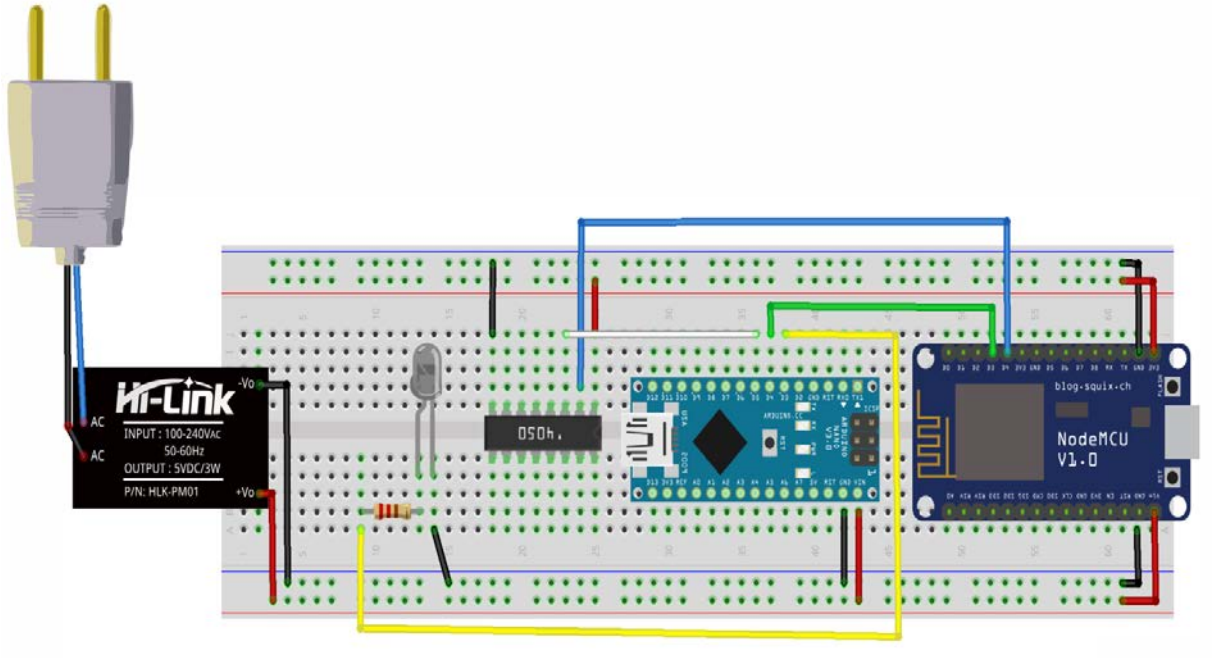

Figura 39 - Esquemático da montagem do servidor de controle do ventilador infravermelho (Própria, 2018)

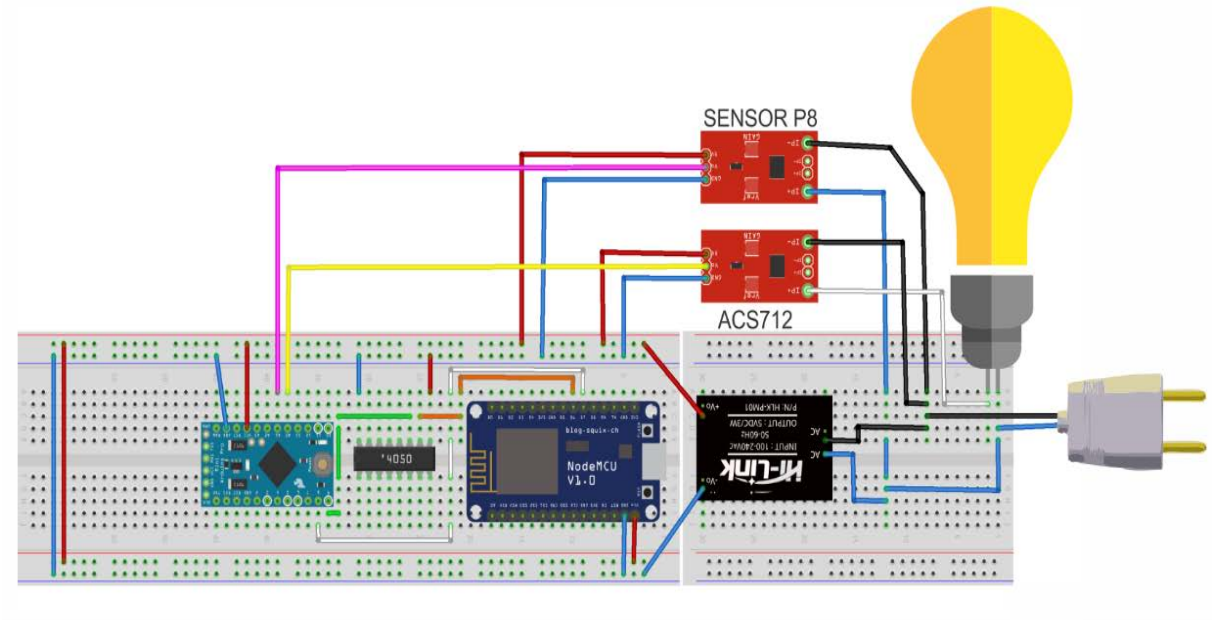

Figura 41 - Esquemático da montagem do servidor de controle estimador do consumo de energia (Própria, 2018)

APÊNDICE B – CÓDIGOS

Código para o Arduino Pro-Mini do servidor de controle estimador do consumo de energia.

```
#include <Filters.h>
#include <SoftwareSerial.h>
#define RX 8
#define TX 9
#define RST_NODEMCU 3

SoftwareSerial mySerial(RX, TX); // RX, TX

double voltage = 0, current = 0, power = 0;


String response = "";
int val = 3;
void setup()
{
  analogReference(DEFAULT);
  mySerial.begin(9600);
  Serial.begin(9600);
  pinMode(RST_NODEMCU, OUTPUT);
  digitalWrite(RST_NODEMCU, LOW);
  delay(100);
  digitalWrite(RST_NODEMCU, HIGH);
}

void loop() {
  //readEsp(1000);
  measure_current(5000);
}

double getVoltageRms() {
  double RawData = 0, AcVoltage = 0;

  RawData = analogRead(A1);
  RawData = (RawData * 5) / 1023;
  AcVoltage = (RawData * 370) / 4.29999;
  return AcVoltage;
  //return RawData;
}

float measure_current(unsigned long printPeriod) {// em milisegundos

  float testFrequency = 60;                     // frequencia do sinal
em Hz
  float windowLength = 20.0 / testFrequency;   // Janela de duraçao para
avaliaçao
  int sensorValue = 0;
  float intercept = -0.1129; // deve ser ajustado com base no teste de
calibraçao
  float slope = 0.0405; // deve ser ajustado com base no teste de
calibraçao
  float current_amps; // corrente instantanea em A
  Serial.print( "\n" );
  voltage=0;
```

```
  for (int i = 0 ;i<21;i++){
    voltage += getVoltageRms();
    delay(100);
  }
  voltage = voltage/20;
  Serial.print("\t Voltage:");
  Serial.print(voltage);
  // Marca tempo em ms desde a ultima leitura
  unsigned long previousMillis = 0;
  RunningStatistics inputStats;                 // cria a classe
estatistica para avaliar o sinal bruto
  inputStats.setWindowSecs( windowLength );
  previousMillis = millis();   // atualiza tempo

  while ((unsigned long)(millis() - previousMillis) < printPeriod) {
    sensorValue = analogRead(A0);  // faz a leitura do valor analogico
    inputStats.input(sensorValue);  // registra por meio da funçao
estatistica
  }

  // mostra o valor de entrada e seu sigma correpondente
  Serial.print( "\tSigma: " ); Serial.print( inputStats.sigma() );
  // converte o sigma em valor de corrente
  current_amps = (intercept + slope * inputStats.sigma());
  if (current_amps < 0.09999) current_amps = 0;
  else current_amps = current_amps-0.09;
  current = current_amps;
  power = current_amps * voltage;

  Serial.print( "\tCurrent: " ); Serial.print( current_amps , 4);
Serial.print( "A rms");
  Serial.print( "\tPower Comsuption: " ); Serial.print( current_amps *
voltage ); Serial.print( "W");
  Serial.println("\n");

  String resposta = String(voltage,0) + " Vrms," + String(current, 4) +
" Arms," + String(power, 2) + " W,";
  Serial.println(resposta);
  mySerial.println(resposta);
}
```

Listing B1 - Código para o Arduino Pro-Mini do servidor de controle estimador do consumo de energia.

Código para o NodeMCU do servidor de controle estimador do consumo de energia.

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <Thread.h>
#include <ThreadController.h>

#define TX D7
#define RX D6
const char* ssid = "ARES";
const char* password = "a552538lich";
String response = "";
SoftwareSerial mySerial(RX, TX ); // RX, TX
ThreadController cpu;
Thread threadLeituraArduino;
//Cria uma instancia do servidor
//especificando a porta do servidor como argumento
WiFiServer server(8008);

//funçao para leitura da comunicaçao serial
void readArduino(){
  long int time = millis();
  String temp = "";
  while ( (time + 500) > millis()) {
    while (mySerial.available()) {
      char c = mySerial.read();
      temp += c;
    }
  }
  if (temp != "") {
    response = temp;
    Serial.println(response);
  }
}

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  //configura a instancia de thread com as funçoes
  threadLeituraArduino.setInterval(500);
  threadLeituraArduino.onRun(readArduino);
  //Adciona a instancia de thread ao gerenciador de threads
  cpu.add(&threadLeituraArduino);
  delay(10);

  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  //Conecta-se a rede Wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite(2, !digitalRead(2));
    Serial.print(".");
  }
```

```
  digitalWrite(2, 0);
  Serial.println("");
  Serial.println("WiFi connected");

  //Inicia o servidor
  server.begin();
  Serial.println("Server started");

  // Printa o endereço IP do servidor
  Serial.println(WiFi.localIP());
}

void loop() {
  //inicia o gerenciador de threads
  cpu.run();
  // Verifica se ha cliente conectado
  WiFiClient client = server.available();
  if (!client) {

    return;
  }

  //Aguarda o cliente enviar dados
  Serial.println("new client");
  while (!client.available()) {
    delay(1);
  }

  //Realiza a leitura da primeira linha da requisiçao
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  //Verifica se a requisiçao bate
  if (req.indexOf("/medidor_consumo/leitura") != -1) {
    Serial.println("Enviando dados...");
  }

  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  client.flush();

  // Preparaçao da resposta
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n ";
  s += " " + response;
  s += "</html>\n";

  // Envia resposta ao cliente
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // O cliente será automaticamente desconectado
  // quando a funçao retorna e o objeto cliente sera destruido}
```

Listing B2 - Código para o NodeMCU do servidor de controle estimador do consumo de energia.

Código para o NodeMCU do servidor de controle da impressora.

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#define RELE D2
const char* ssid = "ARES";
const char* password = "a552538lich";
String response = "";
//Cria uma instancia do servidor
//especificando a porta do servidor como argumento
WiFiServer server(8008);

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  delay(10);

  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  //Conecta-se a rede Wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite(2, !digitalRead(2));
    Serial.print(".");
  }

  digitalWrite(2, 0);
  Serial.println("");
  Serial.println("WiFi connected");

  //Inicia o Servidor
  server.begin();
  Serial.println("Server started");

  // Printa o endereço IP do servidor
  Serial.println(WiFi.localIP());
}

void loop() {
  // Verifica se ha cliente conectado
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  //Aguarda o cliente enviar dados
  Serial.println("new client");
  while (!client.available()) {
    delay(1);
  }

  //Realiza a leitura da primeira linha da requisiçao
  String req = client.readStringUntil('\r');
```

```
  Serial.println(req);
  client.flush();

  //Verifica se a requisiçao bate
  if (req.indexOf("/rele/toggle") != -1) {
    digitalWrite(RELE,!digitalRead(RELE));
  }

  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }


  client.flush();

  // Preparaçao da resposta
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n ";
  s += " " + response;
  s += "</html>\n";

  // Envia resposta ao cliente
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // O cliente será automaticamente desconectado
  // quando a funçao retorna e o objeto cliente sera destruido
}
```

Listing B3 - Código para o NodeMCU do servidor de controle da impressora.

Código para o Arduino Nano do servidor de controle do ventilador infravermelho.

```
#include <IRremote.h>
#include <SoftwareSerial.h>
#define RX 11
#define TX 12
IRsend irsend;
SoftwareSerial mySerial(RX, TX); // RX, TX

unsigned int irSignal_tempo[] = {1300, 400, 1300, 350, 450, 1200, 1300,
350, 1300, 400, 450, 1150, 500, 1150, 500, 1200, 1300, 350, 500, 1200,
450, 1200, 450};
unsigned int irSignal_desliga[] = {1300, 400, 1250, 400, 450, 1200,
1250, 400, 1300, 400, 400, 1250, 400, 1250, 1250, 400, 450, 1200, 450,
1200, 450, 1250, 450};
unsigned int irSignal_1[] = {1300, 400, 1250, 400, 450, 1200, 1300, 350,
1300, 400, 400, 1250, 450, 1200, 450, 1200, 450, 1250, 400, 1200, 1300,
400, 450};
unsigned int irSignal_2[] = {1300, 400, 1300, 350, 450, 1250, 1250, 400,
1250, 400, 450, 1200, 450, 1250, 400, 1200, 500, 1200, 1250, 400, 450,
1200, 450};
unsigned int irSignal_3[] = {1250, 400, 1300, 350, 450, 1250, 1250, 400,
1250, 400, 450, 1200, 450, 1250, 400, 1250, 450, 1200, 450, 1200, 450,
1200, 1300};
int khz = 38; // 38kHz frequencia da onda portadora para o protocolo NEC
String response = "";

void setup(){
  mySerial.begin(9600);
  Serial.begin(9600);
}

void loop() {
  readEsp(1000);
}

//funçao que realiza a leitura da comunicaçao serial
void readEsp(const int timeout)
{
  long int time = millis();
  response = "";

  while ( (time + timeout) > millis()) {
    while (mySerial.available()) {
      char c = mySerial.read();
      response += c;
      if (c == '\n') {
        verificaResposta(response);
      }
    }
  }
  if (response != "")
    Serial.print(response);
}

//função para emitir sinal IR conforme requisiçao
void verificaResposta(String resposta) {
```

```
  if (resposta != "") {
    if (resposta == "d\n")
    {
      Serial.println("Deligando ventilador");
      irsend.sendRaw(irSignal_desliga, sizeof(irSignal_desliga) /
sizeof(irSignal_desliga[0]), khz);
      delay(40);
      irsend.sendRaw(irSignal_desliga, sizeof(irSignal_desliga) /
sizeof(irSignal_desliga[0]), khz);
    }
    else if ( resposta == "t\n") {
      //Aciona Temporizador

      Serial.println("Acionando Temporizador");
      irsend.sendRaw(irSignal_tempo, sizeof(irSignal_tempo) /
sizeof(irSignal_tempo[0]), khz);
      delay(40);
      irsend.sendRaw(irSignal_tempo, sizeof(irSignal_tempo) /
sizeof(irSignal_tempo[0]), khz);
    }
    else if ( resposta == "v1\n") {
      //Velocidade 1

      Serial.println("Velocidade 1");
      irsend.sendRaw(irSignal_1, sizeof(irSignal_1) /
sizeof(irSignal_1[0]), khz);
      delay(40);
      irsend.sendRaw(irSignal_1, sizeof(irSignal_1) /
sizeof(irSignal_1[0]), khz);
    }
    else if ( resposta == "v2\n") {
      //Velocidade 2

      Serial.println("Velocidade 2");
      irsend.sendRaw(irSignal_2, sizeof(irSignal_2) /
sizeof(irSignal_2[0]), khz);
      delay(40);
      irsend.sendRaw(irSignal_2, sizeof(irSignal_2) /
sizeof(irSignal_2[0]), khz);
    }
    else if ( resposta == "v3\n") {

      Serial.println("Velocidade 3");
      irsend.sendRaw(irSignal_3, sizeof(irSignal_3) /
sizeof(irSignal_3[0]), khz);
      delay(40);
      irsend.sendRaw(irSignal_3, sizeof(irSignal_3) /
sizeof(irSignal_3[0]), khz);
    }
  }
}
```

Listing B4 - Código para o Arduino Nano do servidor de controle do ventilador infravermelho

Código para o NodeMCU do servidor de controle do ventilador infravermelho.

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#define TX D3
#define RX D4
const char* ssid = "ARES";
const char* password = "a552538lich";
String response="";
SoftwareSerial mySerial(RX, TX ); // RX, TX
//Cria uma instancia do servidor
//especificando a porta do servidor como argumento
WiFiServer server(8008);

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  delay(10);


  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  //Conecta-se a rede Wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite(2,!digitalRead(2));
    Serial.print(".");
  }

  digitalWrite(2,0);
  Serial.println("");
  Serial.println("WiFi connected");

  //Inicia o Servidor
  server.begin();
  Serial.println("Server started");

  // Printa o endereço IP do servidor
  Serial.println(WiFi.localIP());
}

void loop() {
  // Verifica se ha cliente conectado
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  //Aguarda o cliente enviar dados
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }
```

```
  //Realiza a leitura da primeira linha da requisiçao
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  //Verifica se a requisiçao bate
  if (req.indexOf("/ventilador/desligado") != -1){
     mySerial.println("d\n");
     Serial.println("Ventilador desligado!");
     response="Ventilador desligado!";
  }

  else if (req.indexOf("/ventilador/temporizador") != -1)
   {
     mySerial.println("t\n");
     response="Temporizador acionado!";
     Serial.println("Temporizador acionado!");
  }
  else if (req.indexOf("/ventilador/velocidade_1") != -1)
   {
     mySerial.println("v1\n");
     Serial.println("Ventilador ligado na velocidade 1");
     response="Ventilador ligado na velocidade 1";
  }
  else if (req.indexOf("/ventilador/velocidade_2") != -1)
   {
     mySerial.println("v2\n");
     Serial.println("Ventilador ligado na velocidade 2");
     response="Ventilador ligado na velocidade 2";
  }
  else if (req.indexOf("/ventilador/velocidade_3") != -1)
   {
     mySerial.println("v3\n");
     Serial.println("Ventilador ligado na velocidade 3");
     response="Ventilador ligado na velocidade 3";
  }
  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  client.flush();

  // Preparaçao da resposta
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n ";
 s += " "+response;
  s += "</html>\n";

  // Envia resposta ao cliente
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");
  // O cliente será automaticamente desconectado
  // quando a funçao retorna e o objeto cliente sera destruido
}
```

Listing B5 - Código para o NodeMCU do servidor de controle do ventilador infravermelho.

Código para o NodeMCU do servidor de controle do ventilador da sala.

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#define TX D7
#define RX D8
//#define RST_PROMINI D1
const char* ssid = "ARES";
const char* password = "a552538lich";
String response = "", estado = "desligado";
SoftwareSerial mySerial(RX, TX ); // RX, TX
//Cria uma instancia do servidor
//especificando a porta do servidor como argumento
WiFiServer server(8008);

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  delay(10);

  pinMode(2, OUTPUT);
  digitalWrite(2, 0);

  //Conecta-se a rede Wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    digitalWrite(2, !digitalRead(2));
    Serial.print(".");
  }

  digitalWrite(2, 0);
  Serial.println("");
  Serial.println("WiFi connected");

  //Inicia o Servidor
  server.begin();
  Serial.println("Server started");

  // Printa o endereço IP do servidor
  Serial.println(WiFi.localIP());
}

void loop() {
  // Verifica se ha cliente conectado
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  //Aguarda o cliente enviar dados
  Serial.println("new client");
  while (!client.available()) {
    delay(1);
  }
```

```
  //Realiza a leitura da primeira linha da requisiçao
  String req = client.readStringUntil('\r');
  Serial.println(req);
  client.flush();

  //Verifica se a requisiçao bate
  if (req.indexOf("/ventilador_sala/toggle") != -1) {
    mySerial.println("T\n");
    Serial.println("Ventilador Acionado!");
    if(estado=="ligado") estado="desligado";
    else if(estado=="desligado") estado="ligado";
    response = "Ventilador " + estado;
  }
  else if (req.indexOf("/ventilador_sala/ligado") != -1) {
    mySerial.println("l\n");
    Serial.println("Ventilador ligado!");
    estado="ligado";
    response = "Ventilador ligado!";
  }
  else if (req.indexOf("/ventilador_sala/desligado") != -1) {
    mySerial.println("d\n");
    Serial.println("Ventilador desligado!");
    response = "Ventilador desligado!";
    estado="desligado";
  }

  else if (req.indexOf("/ventilador_sala/leitura") != -1) {
    Serial.println("Lendo estado ...");
    Serial.println(estado);
  }
  else if (req.indexOf("/ventilador_sala/reseta") != -1) {
    Serial.println("Resetando ...");
    mySerial.println("R\n");
    response = "Reset Iniciado...";

  }
  else {
    Serial.println("invalid request");
    client.stop();
    return;
  }

  client.flush();

  // Preparaçao da resposta
  String s = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n ";
  s += " " + response;
  s += "</html>\n";

  // Envia resposta ao cliente
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");
  // O cliente será automaticamente desconectado
  // quando a funçao retorna e o objeto cliente sera destruido
}
```

Listing B6 - Código para o NodeMCU do servidor de controle do ventilador da sala.

Código para o Arduino Pro-Mini do servidor de controle do ventilador da sala.

```
#include <IRremote.h>
#include <SoftwareSerial.h>
#define RX 8
#define TX 9
#define RELE 11
#define RECV_PIN 2
#define RST_NODEMCU 3

SoftwareSerial mySerial(RX, TX); // RX, TX
IRrecv irrecv(RECV_PIN);
decode_results results;


String response = "";
int val=3;
void setup()
{
  mySerial.begin(9600);
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  pinMode(RELE, OUTPUT);
  pinMode(RST_NODEMCU, OUTPUT);
  digitalWrite(RELE, HIGH);
  digitalWrite(RST_NODEMCU, LOW);
  delay(100);
  digitalWrite(RST_NODEMCU, HIGH);
}

void loop() {
  readEsp(1000);
  verifica_IR();
}

void readEsp(const int timeout)
{
  long int time = millis();
  response = "";

  while ( (time + timeout) > millis()) {
    while (mySerial.available()) {
      char c = mySerial.read();
      response += c;
      if (c == '\n') {
        verificaResposta(response);

      }
    }
  }
  if (response != "")Serial.print(response);

}
```

```
//função para acionamento dos rele
void verificaResposta(String resposta) {
  if (resposta != "") {
    if (resposta == "l\n")
    {
      Serial.println("Ventilador da Sala Ligado");
      digitalWrite(RELE, LOW);
    }
    else if (resposta == "d\n")
    {
      Serial.println("Ventilador da Sala Desligado");
      digitalWrite(RELE, HIGH);
    }
     else if (resposta == "T\n")
    {
      Serial.println("Ventilador da Sala Acionado");
      digitalWrite(RELE, !digitalRead(RELE));
    }
     else if (resposta == "R\n")
    {
      delay(1000);
      Serial.println("Reset iniciado!");
      digitalWrite(RST_NODEMCU, LOW);
      delay(250);
      digitalWrite(RST_NODEMCU, HIGH);
      delay(1000);

    }
  }
}

void verifica_IR(){
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
     if(results.value==0xE0E0A857){
      digitalWrite(RELE, !digitalRead(RELE));
      Serial.println("Ventilador da Sala Acionado");
    }
    irrecv.resume(); // Recebe próximo valor
  }
}
```

Listing B7 - Código para o Arduino Pro-Mini do servidor de controle do ventilador da sala.

Código do servidor central de leitura do sensor DHT22 e do servidor de controle estimador do consumo de energia.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Carrega as bibliotecas

import sys
import codecs
import Adafruit_DHT
import RPi.GPIO as GPIO
import time
import csv
from datetime import datetime
from flask import request, Response
from requests.exceptions import ConnectionError
import requests

# Define o tipo de sensor
# sensor = Adafruit_DHT.DHT11

sensor = Adafruit_DHT.DHT22

GPIO.setmode(GPIO.BOARD)
time_counter = 0

# Define a GPIO conectada ao pino de dados do sensor

pino_sensor = 5

# Informacoes iniciais

print '*** Lendo os valores de temperatura e umidade'

while 1:
    if time_counter == 2160:
        print 'Limpando arquivo...'
        f = open('dados_dth22.csv', 'w+')
        f.close()
        f = open('dados_medidor_consumo.csv', 'w+')
        f.close()
        time_counter = 0
    else:

      # Efetua a leitura do sensor

        (umid, temp) = Adafruit_DHT.read_retry(sensor, pino_sensor)

      # Caso leitura esteja ok, mostra os valores na tela

        try:
            esp8266response = \

requests.get('http://192.168.15.15:8008/medidor_consumo/leitura'
                                ).content
            esp8266response = esp8266response.replace('<!DOCTYPE HTML>'
                    , '')
            esp8266response = esp8266response.replace('<html>', '')
            esp8266response = esp8266response.replace('</html>', '')
            esp8266response = esp8266response.replace('\n', '')
```

```
            esp8266response = esp8266response.replace('\t', '')
            esp8266response = esp8266response.replace('"', '')
            esp8266response = esp8266response.replace('\r', '')
          #print(esp8266response)
        except ConnectionError, erro:
          print (erro)
            esp8266response = "Erro de conexão ao medidor,"

        if umid is not None and temp is not None:
            temp_str = '{0:0.1f}'.format(temp)
            umid_str = '{0:0.1f}'.format(umid)

        # print (("Temperatura = {0:0.1f}  Umidade =
{1:0.1f}\n").format(temp, umid))

         # print("Gravando dados no arquivo CSV...")

            with open('/home/pi/servidor_arvim/dados_dth22.csv', 'a'
                      ) as dados:
                dadosFileWriter = csv.writer(dados, delimiter=';')
                dadosFileWriter.writerow([temp_str, umid_str,
                        datetime.now()])
            dados.close()

            with
open('/home/pi/servidor_arvim/dados_medidor_consumo.csv'
                      , 'a') as dados:
                dadosFileWriter = csv.writer(dados, delimiter=';')
                dadosFileWriter.writerow([esp8266response,
                        datetime.now()])
            dados.close()

         # print ("Aguarda 5 segundos para efetuar nova leitura...\n")

            time.sleep(5)
            time_counter = time_counter + 1
        else:

         # Mensagem de erro de comunicacao com o sensor

            print 'Falha ao ler dados do DHT22 !!!'
```

Listing B8 - Código do servidor central de leitura do sensor DHT22 e do servidor de controle estimador do consumo de energia.

APÊNDICE C – TABELAS DE CUSTOS DAS VERSÕES ARVIM

ARVIM VERSÃO 1			
Componentes	Undidades	Preço Unitário	Preço
ARDUINO UNO	1	R$ 35,90	R$ 35,90
DISPLAY CRISTAL LÍQUIDO (LCD 16X02 - AZ/BR)	1	R$ 20,99	R$ 20,99
FONTE 12VDC 110/220V 1,5A C/PLUG P4	1	R$ 16,90	R$ 16,90
JUMPERS MISTOS (FXF - MXF - MXM)	2	R$ 18,90	R$ 37,80
LM35DZ	1	R$ 7,10	R$ 7,10
MÓDULO BLUETOOTH HC-06	1	R$ 39,90	R$ 39,90
MÓDULO RELÉ 5V 8 CANAIS	1	R$ 44,89	R$ 44,89
PROTOBOARD-MINI 170 PONTOS	1	R$ 5,90	R$ 5,90
TOTAL			**R$ 209,38**

Tabela 4 - Tabela de custos do ARVIM versão 1 (Própria, 2018)

ARVIM VERSÃO 2			
Componentes	Undidades	Preço Unitário	Preço
ARDUINO MEGA	1	R$ 73,39	R$ 73,39
CAIXA PATOLA PB-170	1	R$ 69,72	R$ 69,72
CHAVE MINI ALAVANCA ON-OFF-ON (SPDT-180º)	1	R$ 2,43	R$ 2,43
CONECTOR BORNE PRESSÃO RETANGULAR - 2T	4	R$ 1,55	R$ 6,20
COOLER COM BUCHA 40X40X10MM	1	R$ 8,35	R$ 8,35
DISPLAY CRISTAL LÍQUIDO GRÁFICO (GLCD 128X64 - AZ/BR)	1	R$ 59,90	R$ 59,90
FONTE 12VDC 110/220V 2,5A C/PLUG P4	1	R$ 19,90	R$ 19,90
JUMPERS MISTOS (FXF - MXF - MXM)	3	R$ 18,90	R$ 56,70
LM35DZ	1	R$ 7,10	R$ 7,10
MÓDULO ARVIM	1	R$ 50,00	R$ 50,00
MÓDULO BLUETOOTH HC-06	1	R$ 39,90	R$ 39,90
MÓDULO DE RELÉS 5V - 4 CANAIS	1	R$ 22,90	R$ 22,90
MÓDULO RTC COM DS1307 E EEPROM 24C32	1	R$ 7,90	R$ 7,90
MÓDULO WIFI ESP8266 ESP-01	1	R$ 22,50	R$ 22,50
POTENCIOMETRO LINEAR 10K - CURSOR METAL	1	R$ 1,15	R$ 1,15
TOTAL			**R$ 448,04**

Tabela 5 - Tabela de custos do ARVIM versão 2 (Própria, 2018)

ARVIM VERSÃO 3			
Componentes	Undidades	Preço Unitário	Preço
ARDUINO NANO	1	R$ 23,90	R$ 23,90
CAIXA PATOLA PB-170	1	R$ 69,72	R$ 69,72
CHAVE MINI ALAVANCA ON-OFF-ON (SPDT-180º)	1	R$ 2,43	R$ 2,43
CONECTOR BORNE KRE - 3T (PASSO 5MM - AZUL)	4	R$ 0,85	R$ 3,40
CONECTOR PLUG P4 FÊMEA	1	R$ 1,50	R$ 1,50
COOLER COM BUCHA 40X40X10MM	1	R$ 8,35	R$ 8,35
DISPLAY CRISTAL LÍQUIDO GRÁFICO (GLCD 128X64 - AZ/BR)	1	R$ 59,90	R$ 59,90
FONTE 12VDC 110/220V 2,5A C/PLUG P4	1	R$ 19,90	R$ 19,90
JUMPERS MISTOS (FXF - MXF - MXM)	3	R$ 18,90	R$ 56,70
MÓDULO ARVIM	1	R$ 50,00	R$ 50,00
MÓDULO BLUETOOTH HC-06	1	R$ 39,90	R$ 39,90
MÓDULO DE RELÉS 5V - 4 CANAIS	1	R$ 22,90	R$ 22,90
MÓDULO RELÉ 5V/1CANAL	1	R$ 5,90	R$ 5,90
NODEMCU 1.0 ESP 12-E	1	R$ 42,90	R$ 42,90
POTENCIOMETRO LINEAR 10K - CURSOR METAL	2	R$ 1,15	R$ 2,30
REAL TIME CLOCK RTC DS3231	1	R$ 31,90	R$ 31,90
TOTAL			**R$ 441,60**

Tabela 6 - Tabela de custos do ARVIM versão 3 (Própria, 2018)

ARVIM VERSÃO 4			
Componentes	Undidades	Preço Unitário	Preço
ARDUINO NANO	1	R$ 23,90	R$ 23,90
ARDUINO PRO MINI 328 - 5V/16MHZ	2	R$ 19,90	R$ 39,80
CABO DE FORCA PARALELO - PLUG BIPOLAR MACHO	3	R$ 0,89	R$ 2,67
CAIXA PATOLA PB 055	2	R$ 35,85	R$ 71,70
CAIXA PLÁSTICA MULTIUSO SIBRATEC 10X10X8 CM	1	R$ 40,00	R$ 40,00
CAIXA PLÁSTICA MULTIUSO SIBRATEC 20X15X8 CM	1	R$ 70,00	R$ 70,00
CAIXA PLÁSTICA MULTIUSO SIBRATEC 20X20X8 CM	1	R$ 50,00	R$ 50,00
CAIXA PLÁSTICA PATOLA PB-107	1	R$ 40,90	R$ 40,90
CAIXA PLASTICA PATOLA PB114/2 PRETA	1	R$ 42,94	R$ 42,94
CAMERA IP EXTERNA SEM FIO WIRELESS FULLSEC	1	R$ 239,99	R$ 239,99
CAMERA IP WIRELESS EXTERNA	1	R$ 227,90	R$ 227,90
CAMERA IP WIRELESS INTERNA COM MOVIMENTO	1	R$ 114,90	R$ 114,90
CAMERA V2 MODULE RASPBERRY PI	1	R$ 128,76	R$ 128,76
CASE PARA CÂMERA RASPBERRY	1	R$ 35,80	R$ 35,80
CHAVE GANGORRA MINI 2T	3	R$ 0,79	R$ 2,37
FABRICAÇÃO DAS PCBS	1	R$ 70,00	R$ 70,00
FECHO ELÉTRICO HDL FECHADURA FEC-91LA LONGO TRINCO AJUSTÁVEI	1	R$ 103,89	R$ 103,89
FONTE TRA 400	1	R$ 76,70	R$ 76,70
KIT BÁSICO RASPBERRY PI 3 PI3 - SD SANDISK 16GB CLASS 10	2	R$ 265,00	R$ 530,00
LCD 7 OFFICIAL TOUCH SCREEN COM CASE PARA RASPBERRY PI 3 PI3	1	R$ 500,00	R$ 500,00
MINI FONTE 100/240VAC PARA 5VDC (HLK-PM01)	3	R$ 25,39	R$ 76,17
MÓDULO DE RELÉS 5V - 4 CANAIS	1	R$ 22,90	R$ 22,90
MÓDULO RELÉ 5V/1CANAL	2	R$ 5,90	R$ 11,80
MÓDULO SENSOR DE CORRENTE ELÉTRICA ACS712 - 5A	1	R$ 16,99	R$ 16,99
NODEMCU 1.0 ESP 12-E	3	R$ 42,90	R$ 128,70
PLUGS TOMADA FÊMEA	2	R$ 5,00	R$ 10,00
RECEPTOR INFRA-VERMELHO (IR) CODIFICADO 3T	1	R$ 0,99	R$ 0,99
SENSOR DE TENSÃO AC P8	1	R$ 37,90	R$ 37,90
SENSOR DE UMIDADE E TEMPERATURA - DHT22	1	R$ 35,90	R$ 35,90
TOTAL			**R$ 2.753,57**

Tabela 7 - Tabela de custos do ARVIM versão 4 (Própria, 2018)