UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTOTIPO DE UM SISTEMA GERENCIADOR DE EQUIPAMENTOS ELETRÔNICOS VIA WI-FI PARA ECONOMIA DE ENERGIA

GUSTAVO SABEL

BLUMENAU
2016

GUSTAVO SABEL

PROTOTIPO DE UM SISTEMA GERENCIADOR DE EQUIPAMENTOS ELETRÔNICOS VIA WI-FI PARA ECONOMIA DE ENERGIA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer

BLUMENAU
2016

PROTOTIPO DE UM SISTEMA GERENCIADOR DE EQUIPAMENTOS ELETRÔNICOS VIA WI-FI PARA ECONOMIA DE ENERGIA

Por

GUSTAVO SABEL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
Membro:	Prof. Gabriele Jennrich Bambineti, Especialista – FURB

Blumenau, 01 de julho de 2016

Dedico este trabalho aos mesmo pais que me incentivaram e me apoiaram sempre que precisei.

AGRADECIMENTOS

Ao meus pais pelo carinho, apoio e atenção que sempre me deram.

Aos amigos que fiz na faculdade pela parceria, amizade e apoios que me foram dados.

Ao meu orientador Miguel Alexandre Wisintainer por seu apoio e paciência.

Por fim, dedico a todos que me ajudaram e me apoiaram nessa etapa da minha vida.

“A mente que se abre a uma nova ideia jamais volta ao seu tamanho original.”

Albert Einstein

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo que tem como objeto evitar o desperdício de energia monitorando as salas de aula da FURB e desligando os equipamentos eletrônicos, como aparelhos de ar condicionado e projetores multimídia das salas vazias. Para este propósito, foi desenvolvido um dispositivo para ser adicionado às salas de aula que tem a capacidade de desligar esses equipamentos. Ele foi construído utilizando como componentes principais um sensor de movimento, um emissor de infravermelho (IR), um receptor de IR, um Arduino, um módulo baseado no chip ESP8266EX, que possui Wi-Fi integrado e um sensor de corrente elétrica. O dispositivo construído é capaz de se conectar à rede, de verificar se a sala está vazia pela ausência de movimento e de aprender e emitir comandos IR para desligar os equipamentos. Para atingir os objetivos, também foi construída uma aplicação *web* que é responsável por realizar o monitoramento das salas através dos dispositivos, coletando dados do sensor e analisando a fim de identificar quais salas estão vazias. Caso a sala esteja vazia, notifica o dispositivo para que desligue os equipamentos da sala. Para a construção da aplicação *web*, foram utilizadas as linguagens PHP e JavaScript. Por fim, o protótipo funcionou corretamente sendo possível monitorar as salas e desligar seus equipamentos pela rede Wi-Fi.

Palavras-chave: Internet das coisas. ESP8266. WebSocket. Economia de energia.

ABSTRACT

This work presents the development of a prototype which aims to avoid energy waste by monitoring FURB's classrooms and shutting down electronic equipment, such as air conditioners and multimedia projectors of empty rooms. For this purpose, it was developed a device to be added to the classrooms which has the capability of shutting down these equipment. It was built using as main components a motion sensor, an infrared (IR) emitter, an IR receiver, an Arduino board, a module based on the ESP8266EX chip - which has integrated Wi-Fi - and an electric current sensor. The resulting device is capable of connecting to a network, of checking if the room is empty by the lack of motion and of learning and emitting IR commands for shutting down the equipment. To achieve the goals, it was also built a web application which is responsible of carrying out the monitoring of the rooms through the devices, collecting data from the sensor and analyzing it in order to identify which rooms are empty. If the room is empty, it notifies the device to shut down the room's equipment. The languages PHP and JavaScript were employed in the development of the web application. Lastly, the prototype worked appropriately, being able of monitoring the rooms and shutting down their equipment via the Wi-Fi network.

Key-words: Internet of things. ESP8266. WebSocket. Power saving.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MAC - *Media Access Control*

HTTP - *Hypertext Transfer Protocol*

IoT - *Internet of Things*

I²C - *Inter-Integrated Circuit*

I/O - *Input/Output*

IDE - *Integrated Development Environment*

IR - *Infrared*

JSON - *JavaScript Object Notation*

LED - *Light Emitting Diode*

PIC - *Programmable Interface Controller*

PIR - *Passive infrared*

TCP - *Transmission Control Protocol*

SUMÁRIO

1 INTRODUÇÃO

Existe um grande problema de desperdício de energia no Brasil. Segundo Rodrigo Aguiar, o Brasil desperdiça o equivalente a 60% de toda a produção da Usina de Itaipu (EBC RÁDIOS, 2015). De acordo com a Campos (2014) "O Brasil aparece em 15º lugar entre os 16 maiores países do mundo em um ranking sobre eficiência energética". Esses desperdícios se tornam ainda mais preocupantes no futuro, pois segundo Ecoradar Brasil (2008) "A explosão populacional de seis para dez milhões até 2050 provocará uma redução acentuada de muitos recursos naturais que, atualmente, já se encontram em situação de escassez".

Para evitar uma possível falta de recursos energéticos futuramente, deve-se tomar medidas para diminuir esse desperdício. Atualmente existem algumas soluções e uma delas é a automatização predial e residencial utilizando objetos inteligentes conectados à rede, que está sendo chamado de Internet das Coisas (IoT). Segundo Casini (2014, tradução nossa), com a IoT "podemos reduzir o consumo energético dos edifícios enquanto melhoramos o bem-estar ambiental." A tendência é que a IoT cresça muito nos próximos anos. De acordo com Stamford (2013), acredita-se que em 2020 haverá mais de 20 bilhões de dispositivos conectados à internet, desconsiderando *PCs*, *Tablets* e *smartphones*, sendo que haviam apenas 0,9 bilhões em 2009.

A automação predial e residencial também está cada vez mais próxima da realidade das pessoas. Segundo Bolzani (2004), a automação gera comodidade e economia ao longo do tempo, o que compensa o investimento inicial. "Atualmente, muitos construtores já têm essa visão e os edifícios estão sendo preparados para receber novas tecnologias, o que lhes confere também maior vida útil" (BOLZANI, 2004).

Com a IoT se popularizando, com a automação predial e residencial passando a fazer parte de nosso cotidiano e com a crescente preocupação com desperdício de energia, o presente trabalho tem o objetivo de utilizar esses conceitos a fim de diminuir o desperdício de energia principalmente em ambientes muito grande e de difícil controle.

Este trabalho foi desenvolvido pensando na FURB, por ter uma grande infraestrutura e também por ser uma instituição que se preocupa com o consumo dos recursos naturais do planeta. O site Ecoradar Brasil (2008) diz que "Cada quilograma de material perigoso que é evitado, cada metro cúbico de água que é reduzido, cada quilowatt-hora de energia que é poupado representam benefícios para a empresa e para o meio ambiente."

1.1 OBJETIVOS

O objetivo deste trabalho é criar um sistema protótipo capaz de desligar equipamentos eletrônicos via Wi-Fi.

Os objetivos específicos são:

a) desenvolver um protótipo capaz de diminuir o desperdício de energia através da rede Wi-Fi;
b) validar o funcionamento do protótipo desenvolvido.

1.2 ESTRUTURA

O presente trabalho está organizado em quatro capítulos, sendo o primeiro esta introdução, bem como apresentação de seus objetivos e estrutura. No segundo capítulo há uma fundamentação teórica do trabalho que aborda assuntos como IoT, o *chip* ESP8266EX, sensores infravermelhos passivos, controles remotos via infravermelho, protocolo WebSocket e Domótica. Ao final, são descritos alguns trabalhos correlatos.

O terceiro capítulo refere-se ao desenvolvimento do protótipo, através do levantamento de requisitos, especificação, *hardware* e técnicas utilizadas, códigos implementados e operacionalidade. São apresentados também os resultados e discussões sobre os problemas e soluções encontrados no decorrer do desenvolvimento. Por último, o quarto capítulo traz conclusões a respeito do trabalho e sugestões de extensões observadas ao longo do desenvolvimento do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem o objetivo de explorar os principais assuntos abordados nesse trabalho. Os assuntos foram divididos em sete seções. A seção 2.1 trata sobre a Internet das Coisas (IoT). A seção 2.2 dá detalhes do *chip* ESP8266EX e de alguns de seus módulos. A seção 2.3 define um sensor infravermelho e explica seu funcionamento. A seção 2.4 explica o funcionamento de um controle remoto via infravermelho (IR). A seção 2.5 define o que é WebSocket. A seção 2.6 define domótica e por fim, na seção 2.7 são apresentados os trabalhos correlatos.

2.1 INTERNET DAS COISAS (IOT)

Segundo Stamford (2013, tradução nossa), "A Internet das Coisas é a rede de objetos físicos que contêm tecnologia embarcada para se comunicar, sentir ou interagir com os seus estados internos ou o ambiente externo." Por tanto, existem coisas conectadas à internet desde que a mesma surgiu, pois, computadores também são coisas. Ou seja, não existe um nascimento oficial da IoT, mas o termo surgiu em 1999, por Keven Ashton, um funcionário da Procter & Gamble, durante uma apresentação (EMBARCADOS, 2015).

A IoT, quando apresentada por Keven Ashton, era diferente do que é hoje. Na época o termo foi utilizado para a utilização de *tags* de identificação por rádio frequência (RFID) para conectar dispositivos. A internet sem fio estava começando a surgir e, além disso, com o IPV4, nem era possível que todos os dispositivos tivessem um IP único pois não haveria IPs suficientes. Mas hoje, o termo é aplicado basicamente em redes baseadas em IP (TOZZI, 2016).

A Internet das Coisas está sendo bastante aplicada a residências, com a automatização residencial. A automatização residencial tem crescido bastante ultimamente devido a popularização e baixa dos preços da tecnologia. Esse crescimento também ocorre por causa da nova geração de pessoas que estão adquirindo seu primeiro imóvel. Essa geração convive com a tecnologia e por isso são mais exigentes em relação ao seu uso nas residências (MURATORI; DAL BÓ, 2011).

Até hoje, praticamente toda a informação que está na internet foi capturada e criada por humanos. Mas humanos são limitados em tempo, atenção e precisão, o que significa que são bons em capturar informação sobre as coisas do mundo real. Ao criar objetos inteligentes com sensores próprios, se passa a utilizar informações precisas e em tempo real e se pode tomar decisões mais eficientes. Se fosse possível ter todas as informações sobre as coisas, usando informações capturadas por elas mesmas sem nossa ajuda, seria possível controlar e contabilizar tudo e assim reduzir desperdícios, perdas e custos. (ASHTON, 2009).

2.2 *CHIP* ESP8266EX

O ESP8266EX é um *chip* de baixo custo que possui uma solução de Wi-Fi integrada. Ele mede 5mm x 5mm e possui 3 modos de operação para economizar energia: *active mode*, *sleep mode* e *deep sleep mode* (EXPRESSIF, 2016a). O *chip* trabalha com uma tensão elétrica entre 3V e 3,6V, opera com uma corrente de 80mA em média, suporta temperaturas entre -40ºC e 125ºC e o clock da sua *Central Processing Unit* (CPU) é de 80MHz (EXPRESSIF, 2016b). O Quadro 1 apresenta as especificações técnicas do *chip*.

Quadro 1 - Especificações técnicas do *chip* ESP8266EX

Categories	**Items**	**Parameters**
Wifi	Standards	FCC/CE/TELEC/SRRC
	Protocols	802.11 b/g/n/e/i
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit micro controller
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/ IR Remote Control/GPIO/ADC/PWM
	Operating Voltage	3.0 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Storage Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Fonte: ExpressIf (2016b).

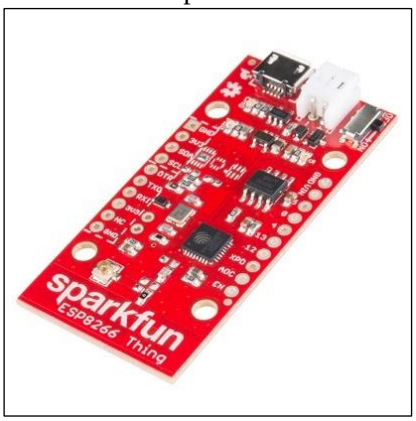
Segundo a ExpressIf (2016b), as suas principais aplicações são: eletrodomésticos, automação residencial, plugs inteligentes e luzes, rede *mesh*, controle industrial sem fio, monitores do bebê, câmeras IP, redes de sensores, eletrônica vestível, dispositivos de localização Wi-Fi, etiquetas de ID de segurança e sistemas de localização Beacons Wi-Fi.

Estão surgindo diversos módulos baseados no *chip* ESP8266EX no mercado. Dentre eles, pode-se citar o SP-WROOM-02 produzido pela ExpressIf e os módulos ESP-NN (Figura 1) produzidos pelo fornecedor AI-Thinkers (ESP8266, 2016). Também estão disponíveis o ESP8266 Thing da SparkFun (2016) da Figura 2 e o MOD-WIFI-ESP8266-DEV da Olimex (2016a) que pode ser visto na Figura 3.

Figura 1 - Módulos da família ESP-NN

Fonte: Curvello (2016).

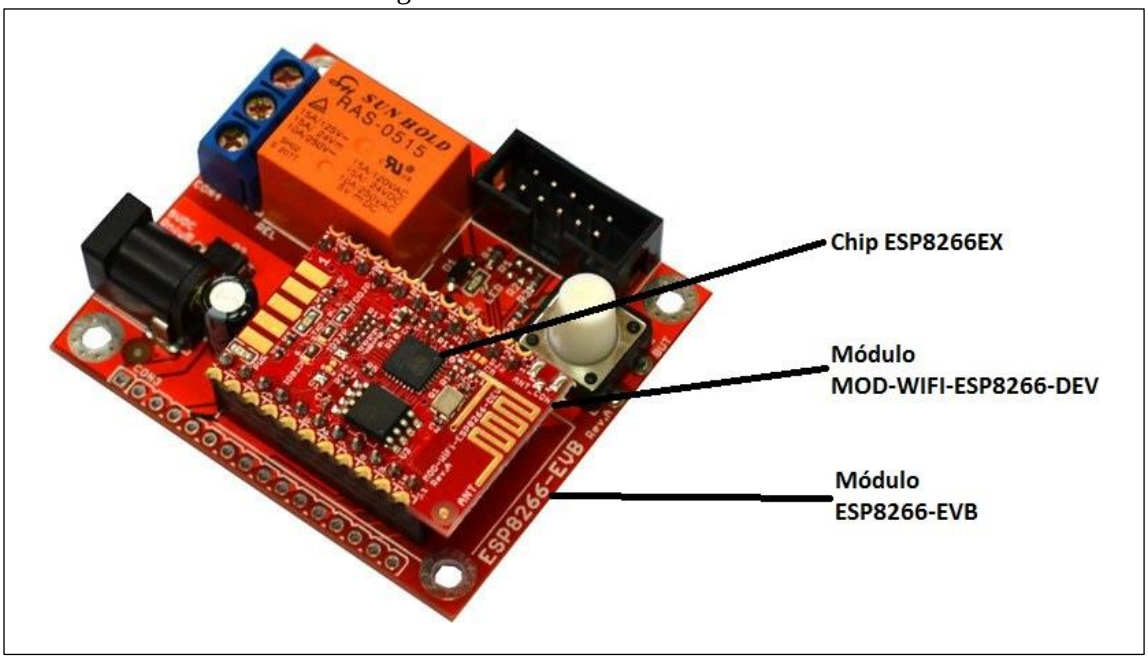
Figura 2 - Módulo SparkFun ESP8266 Thing



Fonte: SparkFun (2016).

O módulo ESP8266-EVB da Olimex (Figura 3) é um facilitador para prototipação pois possui também uma entrada de alimentação de 5V, possui um *relay*[1] e um botão para realizar certas funções como *resetar* ou alterar o modo de operação para permitir atualizar o *firmware*. Além disso, possui um conector *Universal-Extension-Connector* (UEXT)[2] que é uma interface de comunicação serial (OLIMEX, 2016b).

Figura 3 – Módulo ESP8266-EVB



Fonte: Adaptado de Olimex (2016b).

2.3 CONTROLE REMOTO VIA INFRAVERMELHO

Infravermelho é um tipo de radiação eletromagnética, como ondas de rádio, radiação ultravioleta, raios X e micro-ondas. Ela é emitida por qualquer objeto com temperatura maior que 5 graus Kelvin. Essa radiação passa despercebida pelos olhos humanos, mas pode ser sentida na forma de calor. Uma das suas aplicações é para controles remoto, como de TV por exemplo, onde este emite códigos binários em forma de pulsos IR através de um *Light Emitting Diode* (LED). Ou seja, o LED fica acendendo e apagando com em determinada frequência seguindo um padrão. Do outro lado, a TV, possui um receptor que converte os pulsos em um comando que pode ser para desligar, ligar ou aumentar o volume por exemplo. (LUCAS, 2015).

[1] Um *relay* é um interruptor eletromecânico. Ao passar uma corrente elétrica por ele, são abertos ou fechados circuitos. Assim, por exemplo, um dispositivo de 5V, como um Arduino, consegue acender uma lâmpada de 220V. Para isso, a corrente que alimenta a lâmpada deve passar pelo *relay*. Assim que o Arduino manda 5V para o relé, um contato é fechado e a corrente da lâmpada passa pelo relé e acende a mesma.

[2] O conector UEXT, especificado pela Olimex, "suporta três interfaces de comunicação serial: I2C, [Serial Peripheral Interface] SPI e RS232. É uma ótima maneira de adicionar recursos para as placas de desenvolvimento assim o cliente pode escolher qual funcionalidade ele deseja usar." (OLIMEX, 2012, tradução nossa)

Os controles remotos IR funcionam bem, mas têm algumas limitações causada pela natureza do infravermelho. Segundo Layton (2015), o IR funciona em distâncias de apenas 10 metros e o sinal deve ser enviado em linha reta, ou seja, o sinal não atravessa paredes. Outro problema citado por Layton (2015) é que o sinal IR sofre muita interferência dos objetos ao seu redor, principalmente do sol, pois ele emite ondas muito semelhantes ao emitidos pelos controles.

2.4 WEBSOCKET

WebSocket é um protocolo desenvolvido pela *Internet Engineering Task Force* (IETF)[3] para permitir a comunicação de duas vias entre cliente e servidor (IETF, 2011). "O objetivo desta tecnologia é fornecer um mecanismo para aplicativos baseados em navegador que precisam de uma comunicação bidirecional com os servidores" (IETF, 2011, tradução nossa).

Historicamente, a criação de aplicações *web*, que precisam de uma comunicação de duas vias entre um cliente e um servidor, exigem uma utilização muito grande do HTTP para consultar o servidor de atualizações. Isso faz com que servidores precisem de um número diferente de conexões *Transmission Control Protocol* (TCP) para cada cliente e também que as aplicações do lado de cliente precisem manter um mapeamento entre as conexões de entrada e saída para rastrear respostas (IETF, 2011).

Uma solução para esses problemas é a utilização de uma única conexão TCP para o tráfico em ambas as direções e é isso que o protocolo WebSocket fornece. Com o WebSocket é possível abrir uma única conexão TCP entre o cliente e o servidor e assim o aplicativo cliente não precisa ficar fazendo inúmeras requisições ao servidor perguntando por alterações (IETF, 2011).

> WebSocket reduz a latência por que uma vez que a conexão WebSocket é estabelecida, o servidor pode enviar mensagens assim que elas estiverem disponíveis. Por exemplo, diferente de *Polling*, [o protocolo] WebSocket faz uma única requisição. O Servidor não precisa esperar pela requisição do cliente. Similarmente, o cliente pode enviar mensagens para o servidor a qualquer momento. (MOSKOVITS; SALIM; WANG, 2013, p.7, tradução nossa, itálico nosso).

Além disso, apenas a abertura da conexão entre o cliente e servidor é feita utilizando o protocolo HTTP. Depois da conexão aberta, toda a troca de informações é feita diretamente sobre a camada TCP, o que reduz o consumo da rede por não precisar trafegar o cabeçalho do HTTP (IETF, 2011). Outro ponto importante é que não é necessário importar nenhuma API ou

[3] A IETF é uma comunidade internacional aberta composta por designers, operadores, fornecedores e pesquisadores preocupados com a evolução da arquitetura da Internet e do seu bom funcionamento. (IEFT, 2014)

biblioteca para a utilização de WebSocket nos navegadores, pois, segundo WHATWG (2016), a maioria deles já possui suporte nativo e devem seguir a interface definida no Quadro 2.

Quadro 2 - Interface WebSocket definida pela WHATWG

```
enum BinaryType { "blob", "arraybuffer" };
[Constructor(USVString url, optional (DOMString or sequence<DOMString>) protocols = []),
Exposed=(Window,Worker)]
interface WebSocket : EventTarget {
  readonly attribute USVString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long long bufferedAmount;

  // networking
  attribute EventHandler onopen;
  attribute EventHandler onerror;
  attribute EventHandler onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional USVString reason);

  // messaging
  attribute EventHandler onmessage;
  attribute BinaryType binaryType;
  void send(USVString data);
  void send(Blob data);
  void send(ArrayBuffer data);
  void send(ArrayBufferView data);
};
```

Fonte: Adaptado de WHATWG (2016).

2.5 TRABALHOS CORRELATOS

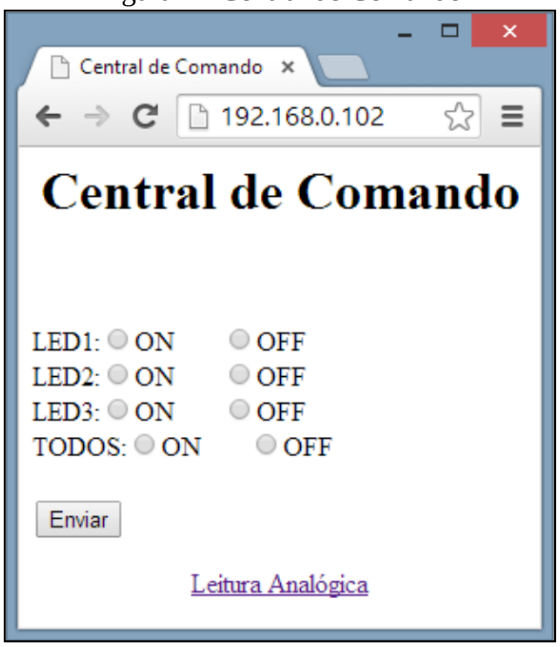
Foram selecionados três trabalhos correlatos os quais tratam com o conceito de automação residencial. São eles o trabalho de Zimmer (2014), o trabalho de Nunes (2014) e do sistema de automação residencial da Fibaro (2015).

2.5.1 SISTEMA DE AUTOMAÇÃO RESIDENCIAL CONTROLADO VIA REDE DOMÉSTICA

No trabalho de Zimmer (2014) foi desenvolvido um sistema de automação residencial, que atua sobre equipamentos eletrônicos, utilizando um microcontrolador *Programmable Interface Controller* (PIC) que se conecta à rede por cabo. No projeto foi utilizado o microcontrolador PIC 18F4620 da Microchip, o módulo HanRun ENC28J60, que possui um conector RJ45, para se conectar à internet e um *display*. O autor também construiu sensores e para detectar se o equipamento está ligado e atuadores para ligar ou desligar o equipamento. Além disso, foi necessário utilizar um expansor de I/Os para barramentos I²C para que um PIC pudesse controlar vários circuitos sensores e atuadores.

O módulo HanRun ENC28J60 cria um servidor que possui uma página com os comandos para ligar e desligar os equipamentos (Figura 4) e uma outra página com o estado dos mesmos. O estado dos equipamentos é capturado pelos sensores e são atualizados em tempo real no PIC 18F4620, porém os dados da página são atualizados somente quando ela é atualizada pelo usuário.

Figura 4 - Central de Comando



Fonte: Zimmer (2014).

Os testes foram feitos com lâmpadas e assim que o usuário clica para desligar ou ligar a luz, o comando é enviado para o HanRun ENC28J60, e o mesmo encaminha o comando para o PIC 18F4620. Este, por sua vez, verifica a requisição e analisa se o comando deve ou não ser aplicado, pois se o comando é para ligar a luz e a mesma já está ligada, o comando deve ser ignorado. Se o comando deve ser enviado, então ele aciona o atuador responsável pelo dispositivo, alterando o estado do *relay*.

Ao final o projeto de automação com o microcontrolador PIC, obteve um desempenho satisfatório aliado a um baixo custo e baixa potência dissipada. Porém o autor identificou que o trabalho tem uma limitação de conexão com mais de um dispositivo, o que poderia ser corrigido utilizando um microcontrolador mais robusto ou outro método de comunicação mais eficiente.

2.5.2 MONITORAMENTO DE CONSUMO DE ENERGIA E ACIONAMENTO REMOTO DE EQUIPAMENTOS POR MEIO DE REDES DE SENSORES SEM FIO

O trabalho de Nunes (2014) tem como objetivo montar uma *Wireless Sensor Network* (WSN) capaz de monitorar remotamente o consumo de energia, além de ligar e desligar equipamentos eletrônicos em ambientes domésticos, comerciais ou industriais.

Para realizar este trabalho, o autor utilizou um Raspberry Pi como dispositivo central da WSN por ter acesso à internet e um alto desempenho. Para cada nó da WSN foi utilizada uma plataforma Arduino Uno, um módulo XBee, um sensor de corrente não invasivo TA12-100 e um circuito de acionamento elétrico baseado em *relays*. O Arduino é o dispositivo principal dos nós WSN, pois ele é quem recebe os comandos do XBee, verifica o sensor de corrente e aciona os circuitos baseados em *relays*.

O XBee é um emissor/receptor de radiofrequência que utiliza o protocolo ZigBee para comunicação. Como o objetivo do trabalho é economizar energia, então esse é o protocolo ideal pois tem um baixo consumo de energia se comparado a outros protocolos como Wi-Fi e Bluetooth. Além disso, é um protocolo facilmente configurável. Os módulos do XBee também que podem funcionar como roteadores, aumentando assim a extensão da rede. O único problema é que o custo desse dispositivo acaba sendo mais elevado que a maioria dos módulos do mercado (NUNES, 2014).

A interface mobile de monitoramento feita pelo autor, que pode ser vista na Figura 5, é simples e consiste em um gráfico que mostra o consumo de energia dos equipamentos medidos pelos nós da WSN. O eixo das abcissas mostra o tempo e o das ordenadas o consumo de cada equipamento em Watts. As curvas de consumo de cada equipamento têm cores diferentes o que facilita a identificação, sendo que uma delas é a medição do ventilador e a outra é a medição de uma lâmpada.

Figura 5 – Ligando a lâmpada no segundo 35 e desligamento do ventilador no segundo 45

Fonte: Nunes (2014).

2.5.3 O SISTEMA FIBARO

Fibaro (2015) é um sistema comercial de automação residencial da Fibar Group e oferece muitas funcionalidades como controlar climatização, controlar iluminação, automatizar de tarefas como irrigação de gramado, segurança da residência, identificação de incêndio, monitorar consumo de energia e vários outros. O sistema possui uma grande quantidade de dispositivos como sensores de movimento, sensores de inundação, sensores de incêndio, tomadas inteligentes que medem o consumo elétrico, além de ser compatível com vários produtos de outros fabricantes.

Todos esses dispositivos podem ser utilizados para configurar ações programadas pelo próprio usuário, utilizando uma interface gráfica, como por exemplo fechar as janelas e desligar os irrigadores caso esteja chovendo, conforme a Figura 6. O sistema também possui procedimentos padrões, como abrir as janelas e portas e soar alarme em caso de incêndio. A

instalação dos dispositivos não é invasiva, pois todos se comunicam via *wireless* utilizando o protocolo Z-Wave e utilizam baterias como fonte de energia.

Figura 6 - Configuração de uma cena

Fonte: Fibaro (2015).

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo, estão descritas as especificações técnicas do protótipo proposto, com seu respectivo levantamento de requisitos, especificação utilizando diagramas de *Unified Modeling Language* (UML), o *hardware* necessário para o desenvolvimento, uma descrição de técnicas e ferramentas utilizadas, detalhamento dos códigos escritos, bem como seu funcionamento após a conclusão do desenvolvimento.

3.1 REQUISITOS

O protótipo é um sistema de monitoramento desenvolvido para ser aplicado na FURB com o intuito de desligar os equipamentos das salas quando as mesmas estiverem desocupadas. Para isso foi necessário construir um dispositivo para ser colocado nas salas que tem capacidade de se conectar à rede e que atue sobre os equipamentos desligando-os utilizando comandos infravermelho, assim como os controles remotos de TV funcionam. Também foi preciso construir uma aplicação para gerenciar esses dispositivos. A fim de facilitar o entendimento do trabalho, o dispositivo que ficará na sala será chamado de Dispositivo ESP, por ter como principal componente o módulo ESP8266-EVB, e a aplicação será chamada de Sistema Gerenciador.

Para atender as necessidades do protótipo foram identificados alguns requisitos funcionais e não funcionais. São eles:

a) aprender comandos IR (Requisito Funcional - RF);
b) desligar equipamentos através de comandos IR (RF);
c) identificar as salas que podem estar vazias (RF);
d) ser capaz de encontrar os Dispositivos ESP na rede (RF);
e) ter uma interface *web* para gerenciar os Dispositivos ESP conectados à rede (RF);
f) utilizar a linguagem PHP para desenvolver o sistema que irá gerenciar os Dispositivos ESP (Requisito não funcional - RNF);
g) utilizar o banco de dados MySQL para o armazenamento (RNF);
h) utilizar conexão WebSocket para se comunicar com o Dispositivo ESP (RNF);
i) utilizar a próprio navegador para comunicar com o Dispositivo ESP (RNF).

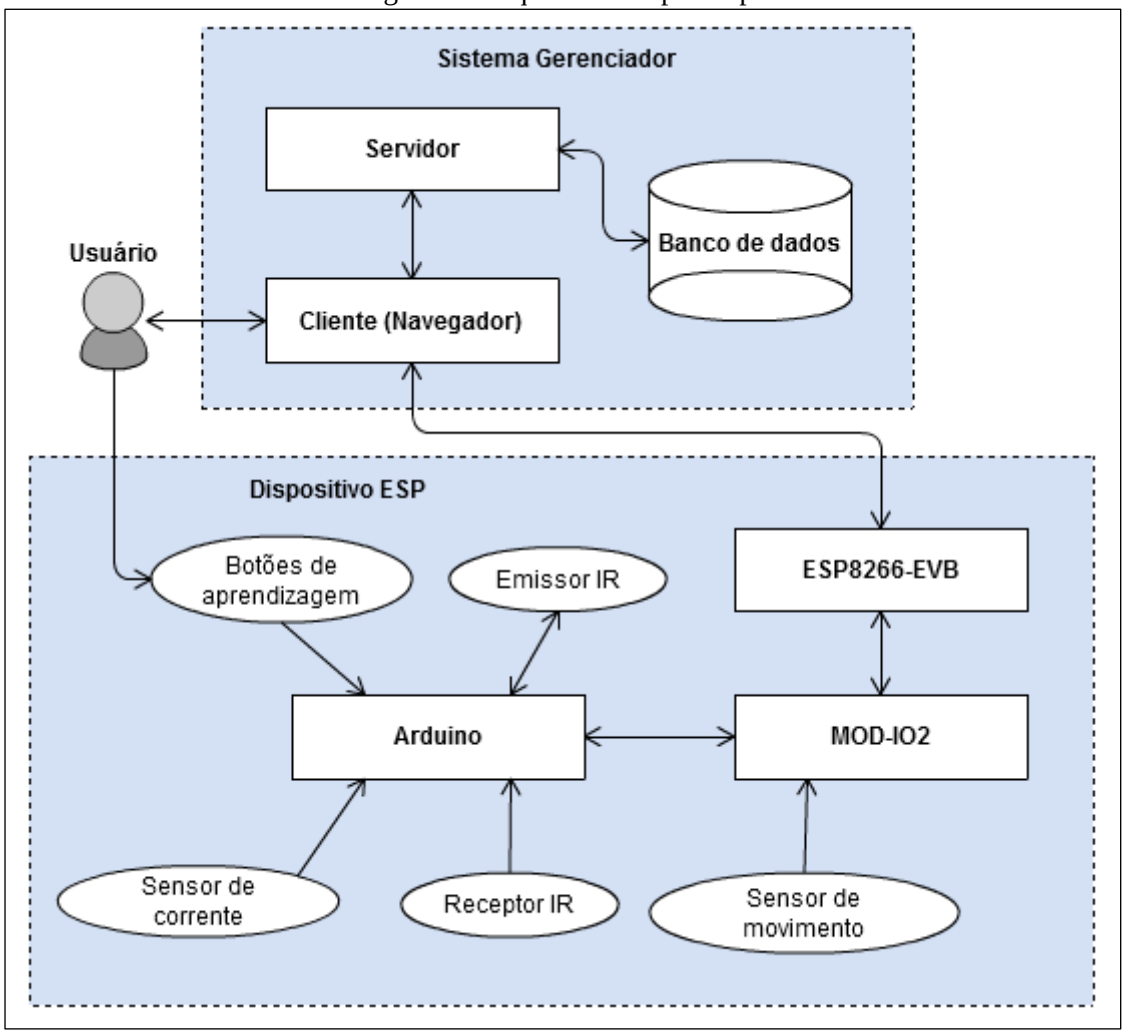
3.2 ESPECIFICAÇÃO

A especificação foi desenvolvida seguindo a metodologia UML na elaboração do diagrama de arquitetura, diagrama de casos de uso e diagrama de atividades. Para tal foi utilizada a ferramenta *web* Cacoo[4].

3.2.1 Arquitetura do protótipo

Na Figura 7 tem-se um diagrama de arquitetura do protótipo.

Figura 7 - Arquitetura do protótipo



Como pode-se ver, o Sistema Gerenciador é composto por 3 partes: Banco de dados, Servidor e Cliente. O Servidor é responsável por manipular o Banco de dados e disponibilizar os dados para a camada cliente. A camada cliente irá funcionar em um navegador, por tanto é uma aplicação web. Essa é a camada de interface para com o usuário e também é ela quem irá

[4] Cacoo é uma ferramenta web para criação de diagramas. Com ele pode-se criar diagramas de UML, protótipos de telas, Modelos Entidade Relacionamento (MER), e vários outros.

monitorar os Dispositivos ESP. Para isso o navegador irá se conectar com o ESP8266-EVB através de uma conexão WebSocket, onde as mensagens trafegadas entre ambos será em formato *JavaScript Object Notation* (JSON).

O Dispositivo ESP é composto basicamente pelo ESP8266-EVB, pelo MOD-IO2 e pelo Arduino. O ESP8266-EVB é o responsável por conectar o dispositivo à rede da FURB e por aguardar se conectar ao Sistema Gerenciador. Assim que à conexão é estabelecida, o Sistema Gerenciador passa a consultar regularmente o estado do Sensor de Movimento que está conectado ao módulo MOD-IO2. Então o ESP8266-EVB retorna o estado de todas as I/Os do MOD-IO2 para o Sistema Gerenciador em um JSON e este verifica o estado da I/O em que o Sensor de movimento está conectado.

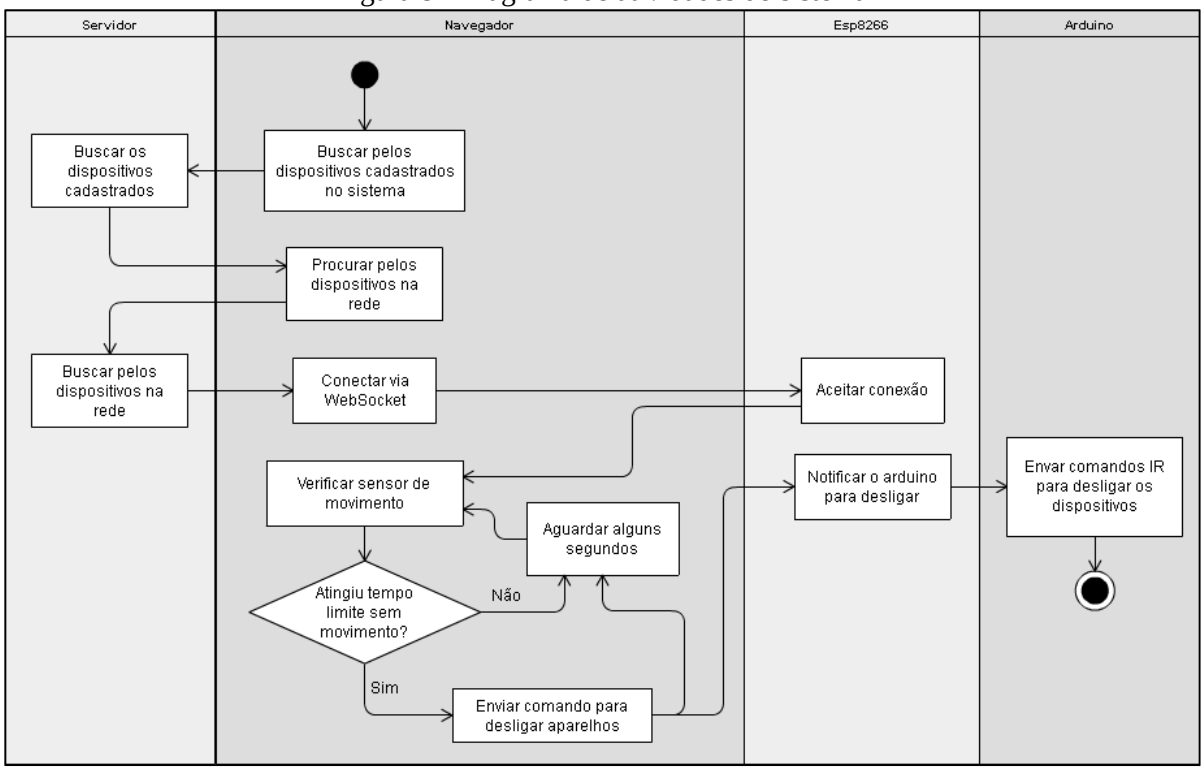
O Arduino é o responsável por aprender os comandos dos controles remotos dos equipamentos que se deseja desligar. Esses comandos são pulsos IR em um determinado padrão que são trafegados entre um controle remoto e o equipamento em que controla. O Arduino tem a capacidade de aprender apenas dois comandos IR diferentes por limitação de memória, sendo que essa aprendizagem deve ser feita pelo usuário diretamente do Dispositivo ESP. Antes de emitir esse comando, o Arduino irá verificar se o equipamento está ligado, mas isso é apenas para um dos dois comandos. Este comando é necessário para os equipamentos em que o comando IR desligar é o mesmo que o comando ligar, pois nesses casos, o comando só pode ser enviado se o equipamento estiver ligado.

Assim que o Sistema Gerenciador detecta que a sala está vazia, por causa de uma ausência de movimento por uma grande quantidade de tempo, então ele pede ao ESP8266-EVB para que desligue os equipamentos da sala. O ESP8266-EVB, por sua vez, pede para o MOD-OI2 para que desligue e este avisa o Arduino o mesmo. Por fim, o Arduino envia os comandos que foram gravados. Mas antes, ele verifica se o equipamento está ligado para um dos comandos. Caso já esteja desligado, esse comando não é enviado.

3.2.2 Diagrama de atividade

A seguir, na Figura 8, tem-se o diagrama de atividades do sistema.

Figura 8 - Diagrama de atividades do sistema



Como pode-se ver, o processo de monitoramento é dividido em quatro partes, sendo que as duas primeiras fazem parte do Sistema Gerenciador e as duas últimas fazer parte do Dispositivo ESP. As quatro partes são as seguintes:

a) a primeira corresponde ao funcionalidade do servidor, o qual tem a função de buscar os Dispositivos ESP pela rede. Além disso, é ele quem armazena as informações do Sistema Gerenciador, como configurações e dados dos Dispositivos ESP cadastrados;

b) a segunda corresponde ao navegador, o qual é responsável pelo monitoramento. Ele é quem requisita para o servidor a identificação dos Dispositivos ESP. Após receber as identificações, se conecta a eles utilizando uma conexão WebSocket. Assim que a conexão for realizada, o navegador passa a monitorar. Quando o ambiente estiver vazio, será enviada uma notificação para o Dispositivo ESP daquele local para que o mesmo desligue os equipamentos;

c) a terceira corresponde ao módulo ESP8266-EVB, o qual é utilizado para conectar o Dispositivo ESP à rede. Esse módulo é o responsável por se comunicar com o Sistema Gerenciador e comandar o Arduino para desligar os equipamentos. Além disso, é quem capta os dados do sensor de movimento.

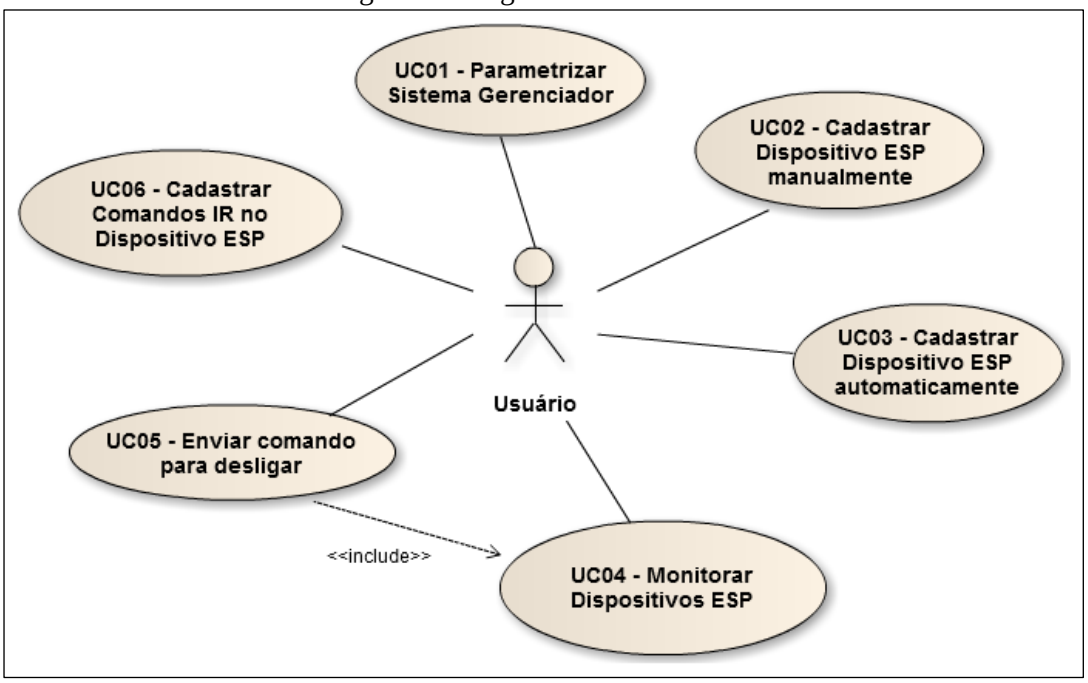
d) a quarta e última etapa corresponde ao Arduino, o qual é responsável por aprender e enviar os comandos IR para desligar para os equipamentos da sala como o ar-

condicionado e o projetor multimídia. Além disso, tem a função de identificar se o equipamento está ligado ou desligado antes de desligar o mesmo.

3.2.3 Diagrama de casos de uso

A Figura 9 mostra o diagrama de caso de uso do Sistema Gerenciador. Os casos de uso do diagrama estão no Apêndice A.

Figura 9 - Diagrama de caso de uso



3.3 IMPLEMENTAÇÃO

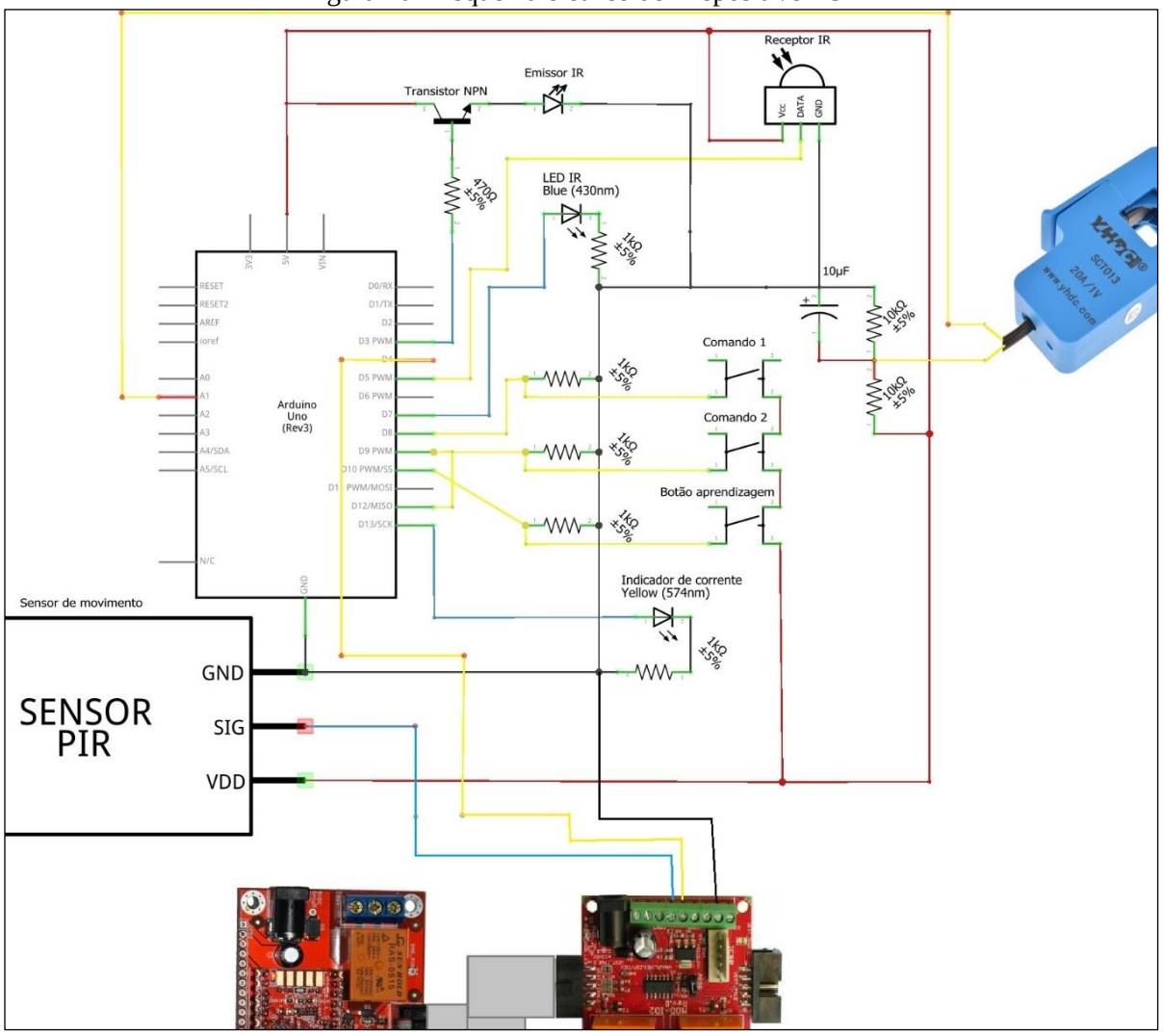
A seguir será mostrado em detalhes a construção do Dispositivo ESP, as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Construção do Dispositivo ESP

Para o protótipo do Dispositivo ESP que ficará na sala, foi utilizado um Arduino Uno, um ESP8266-EVB e um MOD-IO2, sendo que esses dois últimos são módulos produzidos pela Olimex. Também foi necessário um emissor de sinal IR, um receptor IR, resistores, fios conectores, botões, capacitores, transistores, um sensor de movimento e um sensor de corrente não invasivo. Pode-se conferir em detalhes o *hardware* utilizado no Apêndice C. Na Figura 10 pode-se ver o esquema elétrico completo do Dispositivo ESP que foi montado utilizando a ferramenta Fritzing[5].

[5] Fritzing é uma ferramenta *open-source* para criação de protótipos e esquemas elétricos.

Figura 10 - Esquema elétrico do Dispositivo ESP



Conforme o esquema elétrico, o ESP8266-EVB é conectado ao MOD-IO2 através do conector UEXT e ambos se comunicam via barramento I²C. O MOD-IO2 tem sua I/O 3 conectado ao sensor de movimento e a sua I/O 2 à I/O 4 do Arduino. Além disso, o *ground* do MOD-IO2 e o *ground* do Arduino são conectadas.

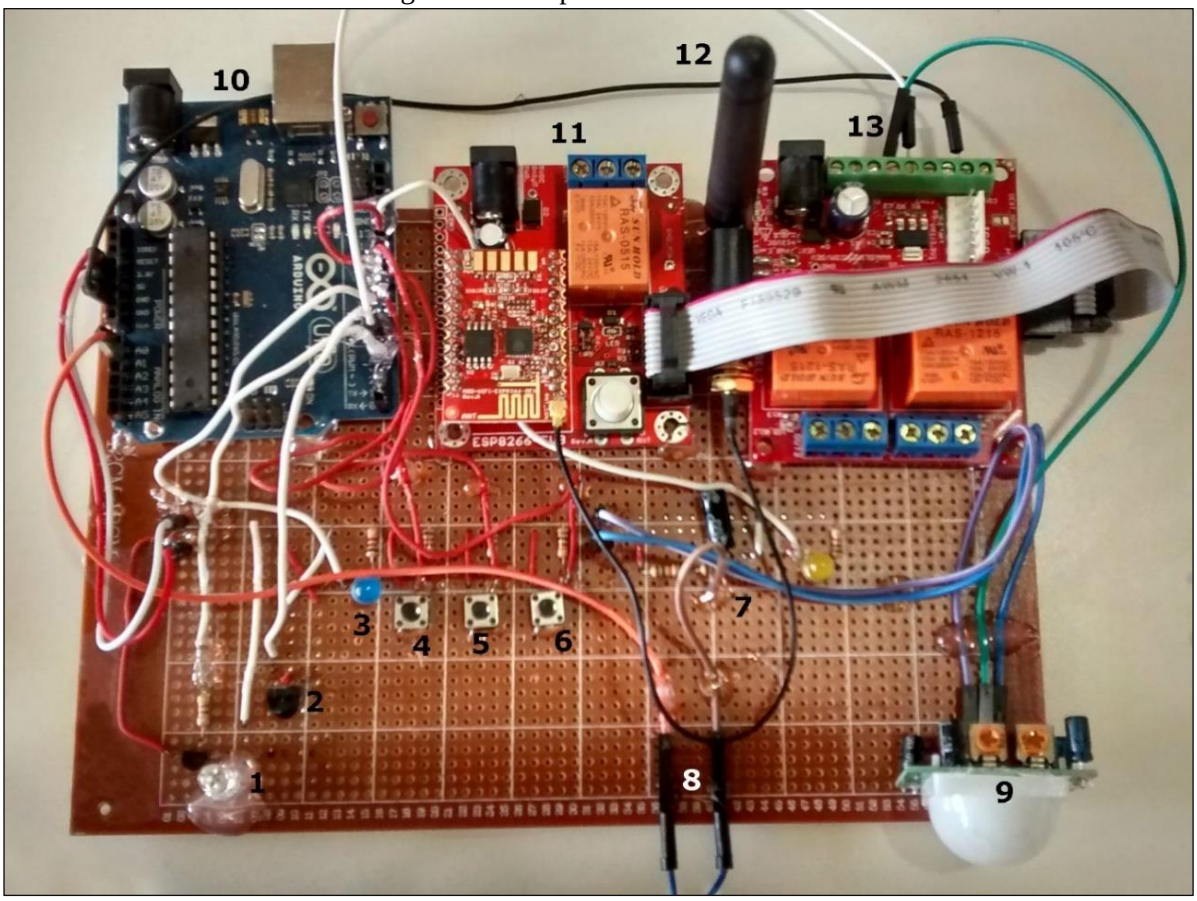
O Arduino tem a I/O 3 ligada à base de um transistor que por sua vez está ligado à um LED Emissor de IR, que é o responsável por emitir os comandos IR aprendidos. O transistor é necessário para aumentar o alcance do sinal emitido para até 10 metros. Se o LED fosse ligado diretamente à I/O 3, o alcance do sinal seria de apenas 3 metros. A I/O 5 é ligada ao receptor de IR, que é o responsável por receber os comandos dos controles remotos dos equipamentos que se deseja desligar. E a I/O 7 é legada à um LED responsável por notificar quando algum comando IR é aprendido ou emitido.

As I/Os 8, 9 são ligadas à botões que armazenam os comandos 01 e 02 e a I/O 10 é conectada ao botão de aprendizagem. Esses botões são utilizados pelo usuário para cadastrar os

comandos IR e também para emitir os comandos para testes. O botão do comando 02 também é ligado à I/O 12 para identificar que o seu comando aprendido só será enviado se for identificado pelo sensor de corrente que o equipamento está ligado. Essas quatro I/Os são ligadas em resistores *pull-down*[6].

O sensor de corrente tem duas saídas, sendo que uma é ligada à I/O A1 do Arduino e a outra é ligada à um pequeno circuito composto por dois resistores e um capacitor. Esse circuito em por objetivo normalizar a tensão que vem do sensor de corrente para que o Arduino consiga ler. Isso é necessário pois a tensão que vem do sensor é alternada, que varia entre 1V e -1V, e o Arduino não consegue ler entradas negativas. Por tanto, esse circuito adiciona 2,5V, fazendo com que varie entre 1,5V e 3,5V, tornando às entradas positivas. O capacitor é necessário para diminuir os ruídos das leituras. O Arduino também tem a I/O 13 ligada a um LED que se acende quando é detectado que o equipamento que o sensor de corrente está medindo está ligado. Na Figura 11 pode-se ver o dispositivo depois de montado.

Figura 11 – Dispositivo ESP montado



[6] Resistor *Pull-down* é utilizado para manter a tensão de algum circuito em 0V. Ou seja, o respectivo circuito é ligado ao *ground* através de um resistor para garantir uma tensão que represente nível lógico baixo enquanto não estiver recebendo tensão de outro lugar.

O dispositivo será melhor detalhado a seguir:

a) o item 1 da Figura 11 é o LED emissor de IR. Esse é o responsável por emitir os comandos infravermelho para desligar os equipamentos eletrônicos da sala;
b) o item 2 é o receptor de IR. Ele é o responsável por aprender os comandos dos controles dos equipamentos;
c) o item 3 é um LED que acende quando um comando IR é emitido ou aprendido;
d) os itens 4 e 5 são botões que representam os comandos 01 e 02 respectivamente. Quando pressionados sozinhos, fazem com que o Dispositivo ESP emita os comandos IR aprendidos para eles. Isso serve apenas para testar se o protótipo está funcionando e se os comandos foram aprendidos corretamente;
e) o item 6 é o botão responsável pela aprendizado. Enquanto ele estiver pressionado, o Arduino estará em modo de aprendizagem. Nesse modo, o sistema estará esperando por comandos IR e assim que algum controle remoto emitir um comando, o código é lido pelo Arduino e vinculado aos botões 4 ou 5. Mas para isso, os botões 4 ou 5 também devem estar pressionados para saber à qual botão o vínculo será feito. Ou seja, ao pressionar o botão de aprendizado e o botão 5 ao mesmo tempo, qualquer comando IR lido pelo Arduino, será vinculado ao botão 5. Por tanto, o Dispositivo ESP só terá a capacidade de aprender dois comandos, um para cada botão;
f) o item 8 é a entrada do sensor de corrente elétrica não invasivo. Nesse item existem duas entradas, uma delas vai direto para o Arduino, em uma de suas entradas analógicas, enquanto a outra vai para o item 7. O item 7 é um circuito regularizador de corrente para que o Arduino consiga interpretar o sinal corretamente. Ali também tem um LED amarelo que acende quando é detectado corrente elétrica. Ou seja, os itens 7 e 8 são responsáveis por identificar se o equipamento da sala está ligado ou não;
g) o item 9 é o sensor de movimento que está ligado ao MOD-IO2;
h) o item 10 é o Arduino, que é responsável por emitir os comandos IR, por aprender os novos comandos IR e verificar a existente de corrente elétrica;
i) o item 11 é o ESP8266-EVB. Este é quem se conectará ao Wi-Fi e receberá os comandos do Sistema Gerenciador;
j) o item 12 é uma antena para amplificar o sinal de Wi-Fi do ESP8266-EVB. Ela não é necessária pois o próprio ESP8266-EVB possui uma antena, mas foi adicionado por haver muitos fios em volta da antena e isso estava gerando muita interferência

e dificultando a conexão com o mesmo;

k) o item 13 é o MOD-IO2, este é um módulo que tem por objetivo disponibilizar novas I/Os para o ESP8266-EVB. Ele é o responsável por passar o comando de desligar para o Arduino e também como entrada para o sensor de movimento.

Na Figura 12 pode-se ver o sensor de corrente utilizado para verificar se um determinado equipamento eletrônico está ligado. Para que funcione, apenas um dos fios do equipamento deve ser passado por ele, caso contrário, os campos magnéticos gerados pelos fios irão se anular e não será detectada corrente elétrica.

Figura 12 - Sensor de corrente

3.3.2 Técnicas e ferramentas utilizadas

Para o desenvolvimento, o sistema é dividido em 3 módulos: Servidor, cliente e protótipo de *hardware*. Para o desenvolvimento do servidor foi utilizada a linguagem PHP. Para codificar, utilizou-se a *Integrated Development Environment* (IDE) Eclipse. Para o servidor *web* foi escolhido o Apache e para o armazenamento dos dados escolheu-se o MySql. Foi utilizada também o phpMyAdmin que é uma interface gráfica para gerenciar o MySql.

Ainda no servidor, foi utilizada uma ferramenta chamada ARP Scan para buscar os Dispositivos ESP conectados à rede e seus respectivos endereços *Media Access Control* (MAC). Como essa ferramenta só existe para ambientes Linux, o servidor foi instalado no sistema operacional Lubuntu[7]. Na camada de cliente foram utilizadas as linguagens JavaScript,

[7] Lubuntu é um sistema operacional mais leve e rápido. O seu núcleo é baseado no Linux e Ubuntu e tem seu foco em velocidade e eficiência energética. (Lubuntu, 2016)

CSS e HTML e as bibliotecas jQuery, Bootstrap e animate.css. Também foi utilizada a API do WebSocket do próprio navegador e para a codificação foi utilizada a IDE Eclipse.

Na camada de *hardware* do protótipo, que é o Dispositivos ESP, para desenvolver o código para o Arduino foi utilizada a IDE Arduino. Para realizar a leitura de corrente elétrica, foi utilizada a biblioteca EmonLib da OpenEnergyMonitor e para aprender e emitir comandos IR, foi utilizada a biblioteca IRremote. O código fonte, tanto do Sistema Gerenciador quanto do Dispositivo ESP, foram gerenciados pelo site de controle de versão GitHub.

3.3.3 Código fonte de aplicação

Nesta seção são apresentados trechos de código para destacar aspectos importantes do projeto que estão divididos entre os módulos navegador, servidor e Arduino.

3.3.3.1 Navegador: monitoramento

O trecho de código do Quadro 3 (linha 94) demonstra como criar uma conexão do navegador com o módulo ESP8266-EVB do Dispositivo ESP. Na linha 96, é atribuída uma função ao objeto do `WebSocket` que será executando quando a conexão for realizada. Ao conectar, é passado como parâmetro um objeto JSON com o usuário e senha do ESP8266-EVB para fazer o *login*. Na linha 117, é atribuída uma função para o atributo `onmessage`. É por essa função que são recebidas as mensagens do módulo. Na linha 134 é atribuída uma função que será executada quando ocorrer algum erro na conexão e na linha 144 é atribuída uma função que será executada quando a conexão for fechada.

Quadro 3 - Criando a conexão WebSocket com o ESP8266-EVB

```
93  this.conectar = function(ip, macAddress) {
94      var socket = new WebSocket('ws://' + ip + '/events');
95
96      socket.onopen = function() {
97          abrindoConexao = true;
98          this.send(JSON.stringify({
99              User : login,
100             Password : password
101         }));
102     };
117     socket.onmessage = function(event) {
118         try {
119             var mensagem = JSON.parse(event.data);
120             if (abrindoConexao && mensagem.Status == "Authorization success") {
121                 sockets[ip] = socket;
122                 onConnect(true, socket);
123                 abrindoConexao = false;
124             } else {
125                 onMessage(event, socket);
126             }
127         } catch (e) {
128             console.log(e.message);
129         }
130     };
134     socket.onerror = function(event) {
135         if (!abrindoConexao) {
136             onError(event, socket);
137         }
138         console.log(event);
139     };
144     socket.onclose = function(event) {
145         if (abrindoConexao) {
146             onConnect(false, socket);
147             abrindoConexao = false;
148         } else {
149             onDisconect(event, socket);
150         }
151         console.log(event);
152     };
```

O Quadro 4 representa a função que notifica o Dispositivo ESP para que ele desligue os equipamentos eletrônicos da sala. Para isso, nas linhas 65 a 71 é enviado um comando informando que o `GPIO2` do MOD-IO2 deve receber o valor `1`. Já nas linhas 73 a 81, é enviado um outro comando depois de 5 segundos para que o `GPIO2` receba o valor `0`. Isso faz com que a saída 2 do MOD-IO2 fique em `ALTO` por 5 segundos e depois volte para `BAIXO`. Essa entrada é monitorada pelo Arduino e quando isso acontece, ele interpreta que deve enviar os comandos IR salvos para os equipamentos da sala.

Quadro 4 - Enviando o comando desligar para o ESP8266-EVB

```
63      this.desligar = function(ip, status) {
64          if (socket = getSocket(ip)) {
65              socket.send(JSON.stringify({
66                  URL : '/mod-io2',
67                  Method : 'POST',
68                  Data : {
69                      GPIO2 : 1
70                  }
71              }));
72
73              setTimeout(function() {
74                  socket.send(JSON.stringify({
75                      URL : '/mod-io2',
76                      Method : 'POST',
77                      Data : {
78                          GPIO2 : 0
79                      }
80                  }));
81              }, 5000);
82          }
83      }
```

O Quadro 5 ilustra a função que consulta o estado do sensor. O que ela faz é solicitar ao ESP8266-EVB as informações sobre o MOD-IO2, que é onde o sensor de movimento está conectado. Como pode ser visto, essa função não tem retorno, e isso ocorre por que o retorno virá somente posteriormente na função `onmessage` do WebSocket (Quadro 3).

Quadro 5 - Consultando o estado do sensor do Dispositivo ESP

```
30      this.consultarSensor = function(ip) {
31          if (socket = getSocket(ip)) {
32              socket.send(JSON.stringify({
33                  URL : '/mod-io2',
34                  Method : 'GET',
35                  Data : {}
36              }));
37          }
38      }
```

Após a requisição, a função `onmessagem` irá receber um parâmetro JSON conforme o Quadro 6. Desse parâmetro será extraído o valor do campo `GPIO3` da linha 10. Quando este valor estiver com `1`, significará que o sensor está detectando movimento e quanto estiver com `0` não está detectando movimento.

Os campos GPIOs são as portas I/Os disponibilizadas pelo MOD-IO2. Para esse projeto foi utilizado apenas a `GPIO3` para o sensor de movimento e o `GPIO2` para notificar o Arduino que este deve enviar desligar os equipamentos elétricos do seu ambiente.

Quadro 6 - Retorno da consulta do estado do sensor

```
{
    "Device" : "MOD-IO2",
    "Status" : "OK",
    "Data" : {
        "Relay1" : 0,
        "Relay2" : 0,
        "GPIO0" : 70,
        "GPIO1" : 218,
        "GPIO2" : 0,
        "GPIO3" : 0,
        "GPIO4" : 1,
        "GPIO5" : 0,
        "GPIO6" : 0
    },
    "I2C_Address" : "0x21"
}
```

3.3.3.2 Servidor

No Quadro 7 está o método em PHP que faz a busca pelos Dispositivos ESP presentes na rede. Na linha 21, o sistema executa o programa Arp scan, para identificar o endereço MAC dos dispositivos na rede. O retorno do programa (Figura 13) será atribuído à variável `$output` na forma de uma lista, onde cada item dessa lista conterá uma linha do retorno. Na linha 26, o sistema percorre cada linha da lista e na linha 27, verifica-se se a linha corresponde a um IP com um endereço MAC. Isso foi feito para ignorar as linhas que não correspondem a endereços, como as primeiras e últimas linhas da Figura 13. Na linha 29, a linha é dividida em *tokens* para separar o IP do endereço MAC e na linha 30, eles são colocados em um `array`, cujo IP é a chave e o endereço MAC o valor.

Quadro 7 - Busca dos Dispositivos ESP conectados à rede

```
19 function BuscarDispositivos() {
20     $cmd = "sudo arp-scan --localnet --timeout=1000";
21     exec ( $cmd, $output );
22
23     $dispositivos = array ();
24
25     $expressao = "/^(\d(\d?){2}.?){4}\s([a-f0-9]{2}:?){6}/i";
26     foreach ( $output as $linha ) {
27         if (preg_match ( $expressao, $linha ) == 1) {
28             $linhaQuebrada = array ();
29             $linhaQuebrada = preg_split ( "/\s/", $linha );
30             $dispositivos [$linhaQuebrada [0]] = $linhaQuebrada [1];
31         }
32     }
33
34     return $dispositivos;
35 }
```

Figura 13 - Retorno do Arp scan

```
gustavo@gustavo-vm:~$ sudo arp-scan --localnet --timeout=1000
[sudo] senha para gustavo:
Interface: ens33, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.8.1 with 256 hosts (http://www.nta-monitor.com/tools/arp-sca
n/)
192.168.1.1     e8:de:27:66:2f:4a       (Unknown)
192.168.1.107   0c:84:dc:ff:35:a9       (Unknown)
192.168.1.100   18:fe:34:a2:09:a9       (Unknown)
192.168.1.104   fc:0f:e6:3e:fa:9a       Sony Computer Entertainment Inc.
192.168.1.109   34:97:f6:2f:3f:a5       (Unknown)
192.168.1.103   c0:65:99:b8:9e:50       (Unknown)
192.168.1.101   14:1a:a3:1c:62:c2       (Unknown)

7 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.8.1: 256 hosts scanned in 3.170 seconds (80.76 hosts/sec). 7 r
esponded
```

3.3.3.3 Arduino

O Quadro 8 apresenta o principal método do Arduino: o `loop` o qual permanece em execução após a inicialização do sistema. Nas linhas 64 a 67, é buscado o estado dos pinos da placa. A linha 74 verifica se o ESP8266-EVB está requisitando que os equipamentos sejam desligados. Nas linhas 76 e 77, os comandos são enviados.

Na linha 84, está sendo utilizado o objeto `irReceptor` (da biblioteca `IRremote`) para verificar se algum comando IR foi recebido. Se sim, então retorna `true` e na variável `resultadoIR` vem o comando recebido, que é repassado para o método `criarComando` das linhas 86 e 89.

Quadro 8 - Loop principal do Arduino

```
63  void loop() {
64      int botao1Estado = digitalRead(PIN_BOTAO_1);
65      int botao2Estado = digitalRead(PIN_BOTAO_2);
66      int botaoAprenderEstado = digitalRead(PIN_BOTAO_APRENDER);
67      int entradaEsp8266Estado = digitalRead(PIN_ESP8266);
68
69      if(botaoAprenderEstado == LOW) {
70          if (ultimoBotaoAprenderEstado == HIGH) {
71              Serial.println("Busca por sinal IR finalizada");
72          }
73          temCorrente = verificarCorrente();
74          if(ultimaEntradaEsp8266Estado == LOW && entradaEsp8266Estado == HIGH) {
75              Serial.println("Comando recebido do ESP8266");
76              validarEnviarComando(COMANDO_1);
77              validarEnviarComando(COMANDO_2);
78          }
79      } else {
80          if (ultimoBotaoAprenderEstado == LOW) {
81              Serial.println("Aguardando sinal IR");
82              irReceptor.enableIRIn(); // Ativa o receptor IR
83          }
84          if (irReceptor.decode(&resultadoIR)) {
85              if(botao1Estado == HIGH){
86                  criarComando(&resultadoIR, COMANDO_1, &comando);
87              }
88              if(botao2Estado == HIGH){
89                  criarComando(&resultadoIR, COMANDO_2, &comando);
90              }
91              irReceptor.resume(); // reinicia receptor
92          }
93      }
94      ultimoBotao1Estado = botao1Estado;
95      ultimoBotao2Estado = botao2Estado;
96      ultimoBotaoAprenderEstado = botaoAprenderEstado;
97      ultimaEntradaEsp8266Estado = entradaEsp8266Estado;
98  }
```

No Quadro 9 está o método que monta o comando e armazena na memória do Arduino. Esse método recebe por parâmetro uma *struck* `resultadoIR`, um número `comandoID` e uma *struck* `comando`. O `resultadoIR` contém o comando IR do controle remoto capturado pelo receptor de IR do Arduino, mas que nesse método será convertido para uma *struct* do tipo `Comando`. O `comandoID` é o índice do comando que será armazenado, que nesse trabalho pode ser 0 ou 1. O último parâmetro, a *struck* `comando`, é utilizado para retornar o comando IR pronto.

Nas linhas 102 o comando recepe o seu índice. Na linha 102, é atribuido ao `comando` se ele deve verificar que o equipamento que deseja desligar esta ligado antes de ser enviado. A constante `PIN_VERIFICADOR` é uma das I/Os do arduino, que nesse trabalho é o 10. Na linha 104, o `comando` recebe o tamanho do comando IR interceptado pelo Arduino.

Nas linhas 106 a 113, o vetor rawbuf da *struct* `resultadoIR` é convertido para milissegundos e repassado para o `comando`. Os índices pares do vetor são *marks* e é nesse momento o LED emissor de IR deve emitir IR, e os ímpares são os *spaces* e nesse momento o emissor de IR fica apagado. Essa conversão é necessária pois a biblioteca captura o comando IR do controle remoto por interrupção, que ocorre a cada 50 milissegundos. Por tanto, antes de passar para a *struct* `comando`, deve-se multiplicar pela constante `USECPERTICK`, cujo valor é 50. Além disso, para reduzir o ruído do receptor, antes de serem passados para o `comando`, as *marks* ainda são subtraídos pela constante `MARK_EXCESS` (linha 108), cujo valor é 100, e os *spaces* são somados pela mesma constante (linha 111).

Quadro 9 - Método que cria o comando IR

```
100 void criarComando(decode_results *resultadoIR, int comandoId, Comando * comando) {
101     digitalWrite(PIN_IR_STATUS, HIGH);
102     comando->id = comandoId;
103     comando->verificarCorrente = digitalRead(PIN_VERIFICADOR) == HIGH;
104     comando->codeLen = resultadoIR->rawlen - 1;
105
106     for (int i = 1; i <= comando->codeLen; i++) {
107         if (i % 2) { // Mark
108             comando->rawCodes[i - 1] = resultadoIR->rawbuf[i] * USECPERTICK - MARK_EXCESS;
109         }
110         else { // Space
111             comando->rawCodes[i - 1] = resultadoIR->rawbuf[i] * USECPERTICK + MARK_EXCESS;
112         }
113     }
114     gravarNoEEPROM(comando);
115     digitalWrite(PIN_IR_STATUS, LOW);
116 }
```

O Quadro 10 apresenta o método que valida se o comando pode ser enviado e o envia. A linha 119 verifica se o comando está salvo na memória e, se sim, o comando é carregado e passado para a variável `comando` (linha 120). Depois, a linha 121 verifica se deve verificar a existência de corrente elétrica. Se não, então o comando é enviado (linha 122), caso contrário, então vai para a linha 125. Se for detectado corrente elétrica, o comando é enviado (linha 126), espera-se 2 segundos (linha 127), verifica-se o sensor novamente (linha 128) e depois volta para

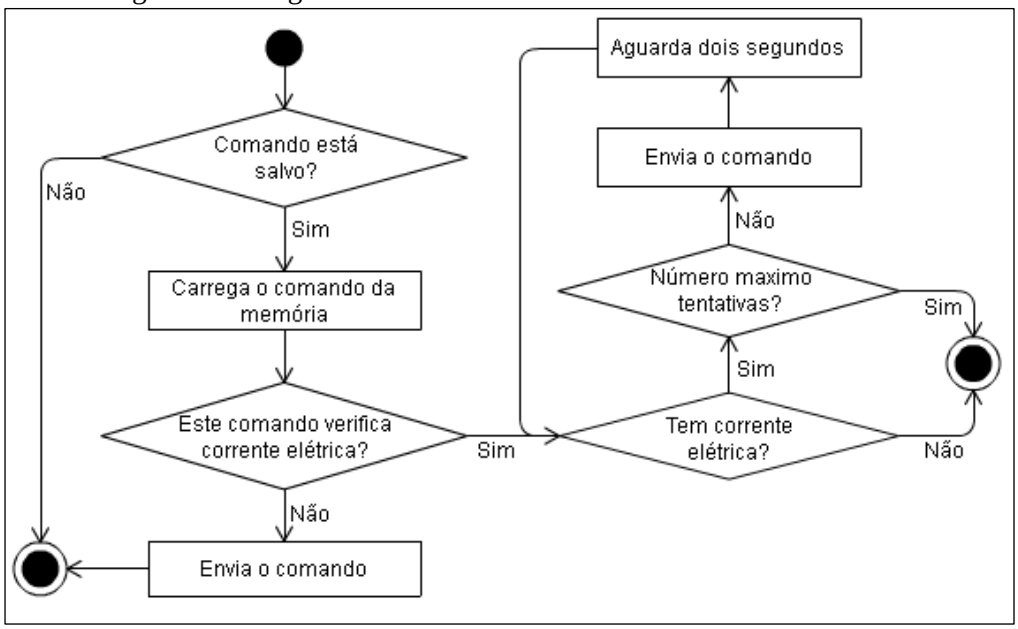
linha 125. Caso ainda tenha corrente elétrica, o processo é feito novamente até que não tenha corrente ou exceda o número máximo de tentativas.

Quadro 10 - Método que verifica e emite o comando

```
118  void validarEnviarComando(int comandoId) {
119      if(comandoEstaSalvoEEPROM(comandoId)) {
120          carregarComandoEEPROM(comandoId, &comando);
121          if(!comando.verificarCorrente) {
122              enviarComando(&comando);
123          } else {
124              int numMaxTentativas = 6;
125              while(temCorrente && numMaxTentativas-- > 0) {
126                  enviarComando(&comando);
127                  delay(2000);
128                  temCorrente = verificarCorrente();
129              }
130          }
131      }
132  }
```

O código do Quadro 10 pode ser melhor entendido com o diagrama de atividades da Figura 14.

Figura 14 - Diagrama de atividade do envio de comando do Arduino



O Quadro 11 mostra a emissão do comando. O seu funcionamento é bem simples pois ele apenas chama o método `sendRaw` do objeto `irEmissor` da biblioteca `IRremote`. Os parâmetros são o código do comando, o tamanho do código e a frequência em que ele deve ser emitido, que nesse caso é 38Mhz. Ao final, na linha 138, é gerada uma espera de 50 milissegundos para garantir que, caso o método `enviarComando` seja chamado duas vezes consecutivas, eles tenham um intervalo, garantindo que um comando não interfirá no outro.

Quadro 11 - Método que emite o comando IR

```
134  void enviarComando(Comando *comando) {
135      digitalWrite(PIN_IR_STATUS, HIGH);
136      irEmissor.sendRaw(comando->rawCodes, comando->codeLen, 38);
137      digitalWrite(PIN_IR_STATUS, LOW);
138      delay(50);
139  }
```

Para realizar a verificação de corrente, é utilizado um objeto da classe `EnergyMonitor`, da biblioteca `EmonLib`, que é inicializado no método `setup`. O objeto possui o método `calcIrms(int)` que é utilizado na linha 145 do Quadro 12. Esse método espera por parâmetro um número, que é a quantidade de amostras que serão utilizadas para determinar a corrente, ou seja, o Arduino consulta o sensor de corrente 1480 vezes e calcula a média para saber o valor da corrente.

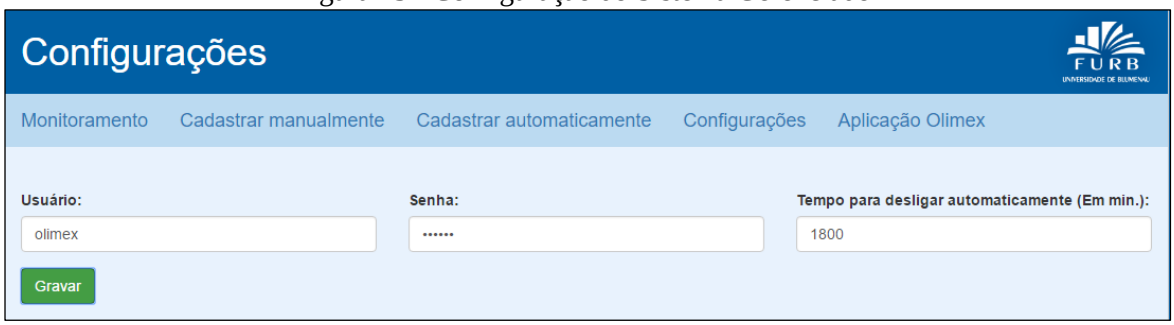
Quadro 12 - Verificando corrente elétrica

```
142  int verificarCorrente() {
143    int temCorrente = 0;
144
145    double corrente = energyMonitor.calcIrms(1480);
146
147    if(corrente > LIMITE_CORRENTE) {
148      temCorrente = 1;
149    }
150
151    if(temCorrente) {
152      digitalWrite(PIN_STATUS_SENSOR_CORRENTE, HIGH);
153    } else {
154      digitalWrite(PIN_STATUS_SENSOR_CORRENTE, LOW);
155    }
156
157    return temCorrente;
158  }
```

3.3.4 Operacionalidade da implementação

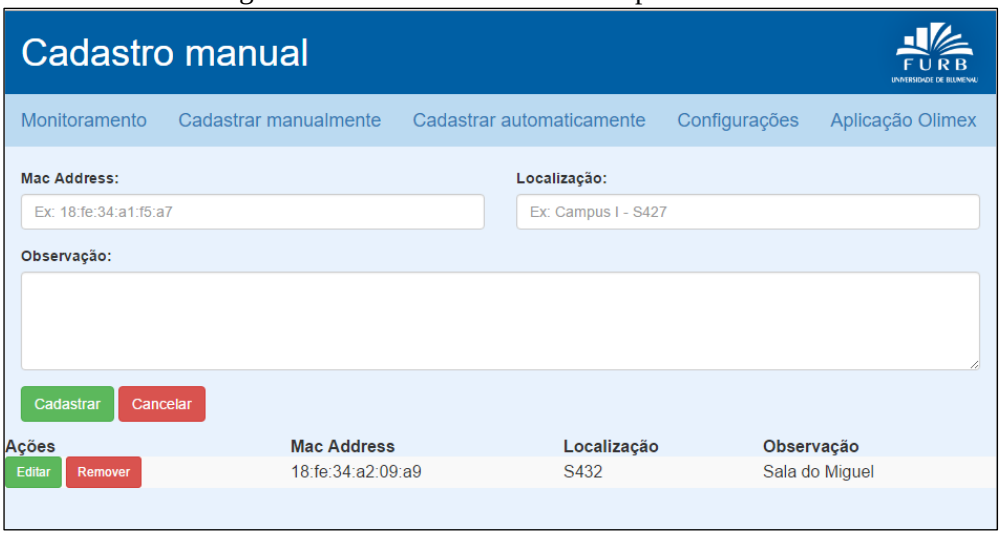
A primeira etapa envolve a configuração do ESP8266-EVB do Dispositivo ESP. Essa configuração está descrita no Apêndice B. Depois disso, deve-se configurar o Sistema Gerenciador para que possa se conectar ao Dispositivo ESP. Para isso, deve-se acessar o menu de configurações (Figura 15) e informar os campos `Usuário` e `Senha`, que devem ser os mesmos parametrizados no ESP8266-EVB no Apêndice B. O campo `Tempo para desligar automaticamente (Em min.)` é o tempo em minutos em que a sala deve ficar sem movimento para o sistema classificar a sala como vazia. E quando a sala for considerada vazia, um comando para desligar os equipamentos eletrônicos da sala é enviado.

Figura 15 - Configuração do Sistema Gerenciador



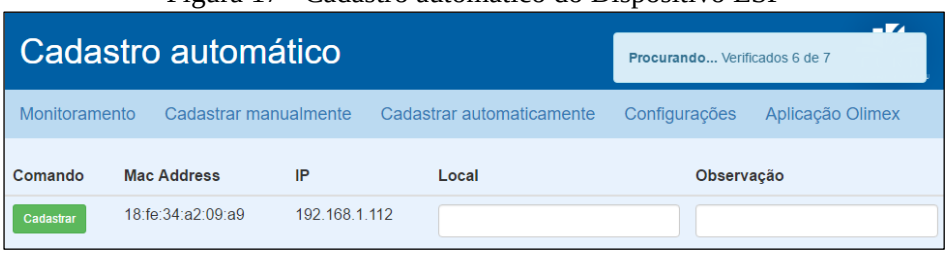
Em seguida, deve-se cadastrar o Dispositivo ESP no Sistema Gerenciador pelo seu endereço MAC. Isso pode ser feito manualmente ou automaticamente. Da forma manual, o usuário insere o endereço MAC do módulo, conforme a Figura 16.

Figura 16 - Cadastro manual do Dispositivo ESP



Da forma automática (Figura 17) o sistema busca pelos Dispositivos ESP por toda a rede e tenta se conectar em cada um utilizando o usuário e senha parametrizados. Ao conectar, o endereço MAC e o IP do módulo são exibidos para o usuário e ele pode cadastrar no sistema clicando no botão `Cadastrar`.

Figura 17 - Cadastro automático do Dispositivo ESP



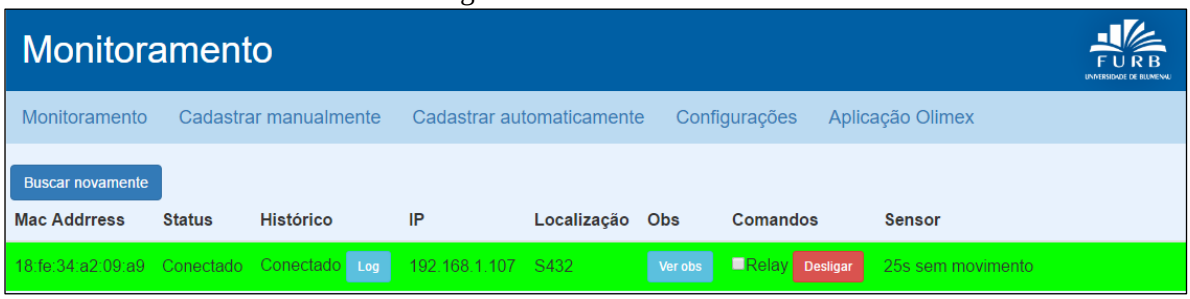
Depois que todos os Dispositivos ESP estiverem cadastrados, então deve-se iniciar o monitoramento. Para isso, basta clicar no menu `Monitoramento` e abrirá a tela da Figura 18. Nessa tela primeiramente serão listados todos os Dispositivos ESP cadastrados e então o

Sistema Gerenciador tentará se conectar em cada um deles. Após se conectar, inicia-se o processo de consultas ao sensor de movimento, que é feita a cada 5 segundos.

Nessa tela de monitoramento temos um grid que possui a coluna `Mac Address` que mostra o endereço MAC do Dispositivo ESP. Possui a coluna `Status` que exibe a situação da conexão. Tem a coluna `Histórico` que contém a última ação do Dispositivo ESP e um comando `Log`. Ao clicar nesse comando, é exibido um histórico de todos as ações e os horários em que aconteceram. Então se o sistema detectou que a sala está vazia e mandou o comando de desligar, isso constará no histórico, assim como também eventuais falhas de conexão. Possui a coluna `IP`, que exibe o IP adquirido pelo Dispositivo ESP. Tem a coluna `Localização` que exibe a sala ou local em que o dispositivo se encontra, sendo que esse campo é informado no cadastro do mesmo. Possui a coluna `Obs` que possui uma observação que também foi cadastrado pelo usuário. Possui a coluna `Comandos` possui o campo `Relay` que se marcado ativará o *relay* do ESP8266-EVB. Esse comando é apenas um meio rápido e fácil de testar a conexão. Essa coluna também possui o comando `Desligar`, que ao ser pressionado, envia o comando desligar para o Dispositivo ESP. Por último, o grid possui a coluna `Sensor`, que indica se a sala está ou não com movimento. Se está sem movimento, então mostra por quanto tempo.

Quando o tempo sem movimento ultrapassar o valor parametrizado, isso indicará que a sala do Dispositivo ESP ficou vazia. Então o Sistema Gerenciador envia automaticamente um comando para que os equipamentos da sala sejam desligados. Mas o usuário também tem a opção de enviar este comando manualmente, antes que atinja o tempo limite, clicando no botão `Desligar`.

Figura 18 - Monitoramento



3.4 RESULTADOS E DISCUSSÕES

A princípio, seria utilizado apenas o ESP8266-EVB da Olimex no Dispositivo ESP, mas durante o desenvolvimento houve a necessidade de incluir um Arduino para que fosse possível gravar e emitir os comandos IR e também para fazer a leitura da corrente elétrica. Como o ESP8266-EVB não tem I/Os disponíveis, foi necessário utilizar também um outro módulo que adicionasse mais portas I/O, o MOD-IO2, também da Olimex.

Um dos problemas encontrados durante o desenvolvimento foi localizar o Dispositivo ESP na rede. O ideal seria que ele se conectasse ao servidor, mas não se descobriu como fazer isso, e alterar o *firmware* do ESP8266-EVB seria muito complexo e demorado. Por esse motivo, optou-se procurar o Dispositivo ESP na rede pelo seu endereço MAC. O problema é que isso só funciona se ele estiver na mesma sub-rede do servidor do Sistema Gerenciador, o que pode ser um problema em infraestruturas que possuem sub-rede diferentes.

Outro problema foi que o ESP8266-EVB do Dispositivo ESP não consegue se conectar diretamente à rede da FURB pois a mesma utiliza um protocolo de segurança que o *firmware* da Olimex não dispõe. Para resolver o problema foi utilizado um roteador em modo *bridge*. Dessa forma, o ESP8266-EVB se conecta à rede à FURB através de um roteador auxiliar.

Também se teve problemas em desligar projetores da marca EPSON. Os testes iniciais estavam sendo feitos apenas em aparelhos de ar condicionado, que geralmente possuem um comando para ligar e outro diferente para desligar. Já os projetores multimídia da marca EPSON utilizam o mesmo comando, tanto para ligar quanto para desligar. Para resolver esse problema, foi necessário fazer com que o Dispositivo ESP conseguisse identificar se o equipamento está ligado ou não. Para isso foi utilizado um sensor de corrente não invasivo e, dessa forma, o Dispositivo ESP só envia o comando se o equipamento estiver ligado. Se depois de dois segundos, o equipamento continuar ligado, o sinal é enviado novamente até que o equipamento esteja desligado. Isso também resolve outro problema desses projetores, que precisam receber o comando para desligar duas vezes para que ele desligue.

A biblioteca IRremote do Arduino, que é responsável pela aprendizagem e envio dos comandos IR, apresentou problemas ao emitir comandos do protocolo NEC. Por alguma razão, o comando era aprendido corretamente, mas ao emitir o equipamento não reconhecia o sinal. Para resolver o problema, o código do Arduino foi alterado para armazenar e emitir apenas o código bruto do comando IR. Isso resolveu o problema.

A identificação de corrente elétrica também apresentou algumas dificuldades. Mesmo ao medir algum equipamento desligado, o sistema estava identificando uma corrente elétrica de 0,06A em média. Foram feitos vários ajustes no protótipo, com a soldagem dos dispositivos em uma placa para minimizar interferências externas que pudessem estar gerando esse valor, mas ainda assim o problema persistiu. Então a solução adotada foi estipular um valor limite de corrente para o protótipo classificar o equipamento como desligado. Esse valor foi de 0,15A.

Outro problema identificado é o fato de o processo de monitoramento do protótipo ser feito pelo navegador com JavaScript. Isso faz com que cada usuário conectado ao sistema de monitoramento crie sua própria conexão com o Dispositivo ESP, e assim cada um irá iniciar

seu próprio monitoramento. No entanto, esse era um problema já esperado, pois o sistema foi criado como protótipo e foi utilizado o navegador justamente por já ter a API WebSocket implementada, e dessa forma facilitar o desenvolvimento.

O Dispositivo ESP conseguiu aprender e replicar os comandos de todos os equipamentos eletrônicos testados. Os testes foram feitos em aparelhos de ar condicionado da marca Komeco, em televisores da marca CCE e Semp Toshiba e projetores da marca EPSON. Ele também conseguiu identificar corrente elétrica dos equipamentos com sucesso.

Por fim, o protótipo construído conseguiu atender os requisitos, superando todas adversidades no desenvolvimento. Com isso os objetivos que o protótipo se propôs a atender foram cumpridos. Ou seja, foi construído um dispositivo capaz de evitar o desperdício de energia da FURB atuando sobre os equipamentos das salas através um sistema de gerenciamento.

O Quadro 13 apresenta de forma comparativa algumas características dos trabalhos correlatos e do trabalho proposto.

Quadro 13 - Características dos trabalhos correlatos e o proposto

Trabalhos relacionados/ Característica	**Zimmer (2014)**	**Nunes (2014)**	**Fibaro (2015)**	**Trabalho proposto**
Faz medição do consumo de energia		X	X	
Não necessita alterar infraestrutura			X	X
Consulta o estado do equipamento eletrônico	X	X	X	X
Atua sobre o equipamento eletrônico	X	X	X	X
Possível acessar de pela internet – IoT	X	X	X	X
Permite programar ações de acordo com variáveis do ambiente			X	
Utiliza IR para atuar sobre os equipamentos eletrônicos				X
Utiliza relês para atuar sobre os equipamentos eletrônicos	X	X	X	
Utiliza rede sem fio	X	X	X	X
Utiliza protocolo TCP/IP	X			X
Utiliza conexão WebSocket				X

Se compararmos com o trabalho de Zimmer (2014), pode-se ver que o trabalho proposto faz quase tudo que ele faz. Porém o trabalho proposto não aciona *relays* o que impede que sejam apagadas luzes e outros equipamentos sem interface IR. O trabalho de Zimmer (2014) também tem um custo menor por não utilizar Wi-Fi e sim conexão a cabo, mas isso torna a sua instalação mais custosa por necessitar a passagem de fios para conectá-lo à rede.

Outro problema é que se o trabalho de Zimmer (2014) fosse aplicado à equipamentos que não utilizam interruptores, como aparelhos de ar condicionado ou projetores, os mesmos quando fossem desligados não poderiam mais ser ligados utilizando o controle pois o *relay* teria cortado a corrente elétrica do equipamento. Por tanto, para poder religar, o usuário teria que acessar o sistema para liga-lo e depois também ligar utilizando o controle. Já no trabalho

proposto isso não é necessário pois os equipamentos são desligados via comandos IR, o que é basicamente o mesmo que o controle faria se fosse desligado pelo usuário.

Ao comparar com o trabalho de Nunes (2014), pode-se verificar que ele consegue medir o consumo de energia, o que torna interessante para análises tomadas de decisão para medidas de economia de energia. Também é interessante por utilizar o XBee, pois o mesmo utiliza o protocolo de comunicação ZigBee, que é de baixo consumo de energia, no entanto também uma desvantagem por ser um dispositivo caro, sendo que o trabalho proposto utiliza um módulo de menor custo.

Uma desvantagem do trabalho de Nunes (2014) para o propósito desse trabalho na utilização do protocolo de comunicação ZigBee é que o mesmo não estaria aproveitando a infraestrutura já existente de Wi-Fi disponibilizada pela FURB. Além disso, o trabalho de Nunes (2014) também utiliza *relays* para atuar sobre os equipamentos eletrônicos, o que gera o mesmo problema do trabalho de Zimmer (2014) com a questão do desligamento dos equipamentos do tipo que não são ligados por interruptores.

A Fibaro (2015) permite a criação de sistemas de automação bastante completos, com a possibilidade de ações programas pelo próprio usuário e também permite fazer medições de consumo de energia. Seria um sistema perfeito se não fosse o alto custo dos equipamentos. Além disso, o sistema é feito para residências e ele não seria o ideal para grandes estruturas como centenas de sensores. O trabalho também não reaproveitaria a infraestrutura de Wi-Fi da FURB por utilizar o protocolo de comunicação Z-Wave, assim como o trabalho de Zimmer (2014).

Por fim, o trabalho proposto se mostra interessante por não ter a necessidade de alterar a infraestrutura do ambiente, pois utiliza comandos IR para atuar sobre os equipamentos eletrônicos. Também é relevante por utilizar módulos de baixo custo e, além disso, por utilizar conexões WebSocket com os módulos ESP8266-EVB, diminuindo assim tráfego de dados e tendo um melhor aproveitamento da rede.

4 CONCLUSÕES

Este trabalho descreve o projeto de um protótipo de monitoramento de espaços públicos, como a FURB, objetivando diminuir o gasto com energia elétrica e vigias. Conforme apresentado, foi concebido um protótipo de dispositivo e uma aplicação *web* que demonstram a viabilidade do projeto.

As ferramentas utilizadas serviram satisfatoriamente para criar o protótipo, que foi capaz desligar os equipamentos via IR através da rede Wi-Fi. A utilização do módulo ESP8266-EVB foi de grande utilidade por ter um firmware desenvolvido para IoT, por permitir conexões WebSocket e por ser de fácil prototipação. O WebSocket foi importante por reduzir o número de requisições e o tamanho dos pacotes trafegados. Os testes demonstraram que o protótipo funciona e que pode ser utilizado para controlar grandes espaços.

O trabalho foi importante por explorar uma forma de automatização que pode reduzir os gastos além de criar um ponto de partida para que surjam soluções melhoradas. Contribuiu também por explorar um dos módulos baseados no *chip* ESP8266EX, que são mais acessíveis financeiramente e que estão sendo bastante utilizados para IoT. Com isso foi possível identificar as principais dificuldades em utilizar o módulo e os pontos que ainda devem ser explorados para se tornar uma tecnologia mais eficiente.

4.1 EXTENSÕES

Dentre as extensões possíveis, destaco aqui as mais importantes para um melhor funcionamento do protótipo:

a) implementar o monitoramento no servidor em vez de fazer isso no navegador. Hoje quem se conecta com o ESP8266-EVB é o próprio navegador, pois qualquer navegar atualmente já tem suporte nativo para WebSocket. No entanto isso poderia ser feito no próprio servidor, utilizando, por exemplo, a linguagem NodeJs que possui bibliotecas disponíveis para realizar conexões WebSocket;

b) conseguir fazer com que o ESP8266-EVB se conecte diretamente na rede da FURB. Atualmente, para que o módulo se conecte à rede da Universidade, é necessário que ele se conecte através de um roteador em modo *bridge*. Talvez seria interessante implementar o protocolo necessário para que o mesmo consiga se conectar diretamente. Ou talvez, isso pudesse ser resolvido dando algum tipo de tratamento diferenciado para esses módulos pela rede da FURB;

c) substituir o ESP8266-EVB e o MOD-IO pelo módulo MOD-WIFI-ESP8266-DEV que tem um custo financeiro menor. Hoje se está utilizando esses módulos pois são

mais fáceis para prototipar, mas são muito mais caros e por isso seria interessante substituir por um módulo mais simples;

d) deixar de utilizar o Arduino. Nesse trabalho, o Arduino foi necessário para aprender e emitir comandos IR e também para detectar corrente elétrica, mas tudo isso pode ser implementado diretamente em algum módulo baseado no *chip* ESP8266EX;
e) conseguir procurar pelos Dispositivos ESP em toda a rede da FURB, sem a necessidade de se ter o endereço MAC cadastrado. No trabalho atual, para que se consiga se conectar à ele, precisa-se do seu endereço MAC e isso impede que o sistema funcione em outras sub-redes. Então seria interessante achar uma maneira diferente de encontrar os encontrar. Talvez a melhor maneira seja o próprio Dispositivo ESP se conectar ao servidor;
f) tornar o sistema mais seguro, pois esse trabalho não preocupou-se com a segurança do sistema já que o foco era a criação de um protótipo funcional;
g) aproveitar melhor a conexão WebSocket, fazendo com que o próprio Dispositivo ESP da sala notifique o servidor quando o status do sensor se alterar. Dessa forma o servidor não precisa ficar consultando repetidamente o status do sensor e isso faria com que se gastasse menos processamento do servidor e menos consumo da rede, além de consumir menos energia;
h) aproveitar a detecção de corrente elétrica para também medir o consumo de energia dos mesmos. No trabalho atual já é utilizado um sensor de corrente para verificar se o equipamento está ligado. Esse mesmo sensor poderia ser utilizado para monitorar o consumo do equipamento.

REFERÊNCIAS

ASHTON, Kevin. That ‘internet of things’ thing. **RFiD Journal**, [S.I], v. 22, n. 7, p. 97-114, jun. 2009. Disponível em: <http://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf>. Acesso em: 05 out. 2015.

BOLZANI, Caio A. M. **Residências Inteligentes**. São Paulo: Editora Livraria da Física, 2004.

CAMPOS, Álvaro. **Brasil fica em 15º em ranking de eficiência energética**, jul. 2014. Disponível em: <http://exame.abril.com.br/brasil/noticias/brasil-fica-em-15o-em-ranking-de-eficiencia-energ%C3%A9tica>. Acesso em: 07 jul. 2016.

CASINI, Marco. **Internet of things for Energy efficiency of buildings**, 2014. Disponível em: <http://www.academia.edu/9369984/M._Casini_Internet_of_things_for_Energy_efficiency_of_buildings_ICAEE_2014_NICE>. Acesso em: 27 set. 2015.

CURVELLO, André. **Apresentando o módulo ESP8266**, 2016. Disponível em: <http://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 02 jun. 2016.

EBC RÁDIOS. **Encontro discute eficiência energética e novas soluções tecnológicas de consumo**, 2015. Disponível em: <http://radios.ebc.com.br/revista-brasil/edicao/2015-08/encontro-discute-eficiencia-energetica-e-novas-solucoes-tecnologicas>. Acesso em: 19 set. 2015.

ECORADAR BRASIL, 2008. Disponível em: <http://www.furb.br/ecoradar/brasil/index.htm>. Acesso em: 04 set. 2015.

EMBARCADOS.**IoT - Internet das Coisas - Mocinha ou Vilã?**, jul. 2015. Disponível em: <http://www.embarcados.com.br/editorial-iot-internet-das-coisas>. Acesso em: 07 jul. 2016.

ESP8266. **Communit wiki**: Modules, 2016. Disponível em: <http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family>. Acesso em: 22 jun. 2016.

EXPRESSIF, 2016a. Disponível em: <https://espressif.com/en/products/hardware/esp8266ex/overview>. Acesso em: 20 abr. 2016.

_____. **ESP8266EX:** Datasheet, 2016b. Disponível em: <http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 21 jun. 2016.

FIBARO. **Fibaro System**, 2015. Disponível em: <http://www.fibaro.com/en/the-fibaro-system>. Acesso em: 12 set. 2015.

IETF. **The WebSocket Protocol**, dez. 2011. Disponível em: <http://tools.ietf.org/html/rfc6455>. Acesso em: 19 set. 2015.

IETF. **About the IETF**, [2014]. Disponível em: <https://www.ietf.org/about/>. Acesso em: 09 jul. 2016.

LAYTON, Julia. **How Remote Controls Work**, [2015]. Disponível em: <http://electronics.howstuffworks.com/remote-control.htm>. Acesso em: 02 out. 2015.

LUBUNTU. [2016]. Disponível em: <http://lubuntu.net>. Acesso em: 13 jul. 2016.

LUCAS, Jim. **Livescience**: What Is Infrared?, 2015. Disponível em: <http://www.livescience.com/50260-infrared-radiation.html>. Acesso em: 12 out. 2015.

MCCOMB, Gordon; SHAMIEH, Cathleen. **Eletrônica para Leigos**. 2. ed. Tradução Fernando Effiore; Roberto Assis Rezende. Rio de Janeiro: Alta Books, 2012.

MOSKOVITS, Peter; SALIM, Frank; WANG, Vanessa. **The Definitive Guide to HTML5 WebSocket**. [S.I.]: Apress, 2013.

MURATORI, José R.; DAL BÓ, Paulo H. **Automação residencial**: Histórico, definições e conceitos, [2012?]. Disponível em: <http://www.osetoreletrico.com.br/web/documentos/fasciculos/Ed62_fasc_automacao_capI.pdf>. Acesso em: 20 jun. 2016.

NUNES, Ivan de O. **Monitoramento do Consumo de Energia e Acionamento Remoto de Equipamentos por Meio de Redes de Sensores Sem Fio**. 2014. 57f. Trabalho de conclusão de curso (Graduação - Curso de engenharia de computação) - Universidade Federal do Espirito Santo, Vitória.

OLIMEX. **MOD-WIFI-ESP8266-DEV**, [2016a]. Disponível em: <https://www.olimex.com/Products/IoT/MOD-WIFI-ESP8266-DEV/open-source-hardware>. Acesso em: 21 jun. 2016.

_____. **ESP8266-EVB**, [2016b]. Disponível em: <https://www.olimex.com/Products/IoT/ESP8266-EVB/open-source-hardware>. Acesso em: 21 jun. 2016.

_____. **Universal EXTension connector (UEXT)**, out. 2012. Disponível em: <https://www.olimex.com/Products/Modules/UEXT/resources/UEXT_rev_B.pdf>. Acesso em: 21 jun. 2016.

SPARKFUN. **SparkFun ESP8266 Thing**, [2016]. Disponível em: <https://www.sparkfun.com/products/13231>. Acesso em: 22 jun. 2016.

STAMFORD, Conn. **Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020**, dez. 2013. Disponível em: <http://www.gartner.com/newsroom/id/2636073>. Acesso em: 18 set. 2015.

TOZZI, Chistopher. **IoT Past and Present: The History of IoT, and Where It's Headed Today**, abr. 2016. Disponível em: <http://talkincloud.com/cloud-computing/iot-past-and-present-history-iot-and-where-its-headed-today>. Acesso em: 08 jul. 2016.

WHATWG. **HTML**: Living Standard, jun. 2016. Disponível em: <https://html.spec.whatwg.org/multipage/comms.html#network>. Acesso em: 23 jun. 2016.

ZIMMER, Alan Valente. **Sistema de automação residencial controlado via rede doméstica**. 2014. 40 f. Trabalho de conclusão de curso (Bacharelado - Engenharia Elétrica) - Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá. Disponível em: <http://hdl.handle.net/11449/123024>. Acesso em: 20 set. 2015.

APÊNDICE A – Casos de uso

O caso de uso `UC01 – Parametrizar sistema` (Quadro 14) demonstra como o ator Usuário parametriza o sistema.

Quadro 14 - Detalhamento do `UC01 – Parametrizar sistema`

Número	01
Caso de uso	Parametrizar sistema
Ator	Usuário
Cenário principal	1. O usuário seleciona o menu configurações; 2. O usuário informa o campo Login; 3. O usuário informa o campo Senha; 4. O usuário informa o campo Tempo; 5. O usuário clica em Gravar.
Pós-condição	O sistema apresenta uma mensagem informando que a configuração foi alterada com sucesso

O caso de uso `UC02 – Cadastrar Dispositivo ESP manualmente` (Quadro 15) demonstra como o ator Usuário cadastra um Dispositivo ESP manualmente.

Quadro 15 - Detalhamento do `UC02 – Cadastrar Dispositivo ESP manualmente`

Número	02
Caso de uso	Cadastrar Dispositivo ESP manualmente
Ator	Usuário
Cenário principal	1. O usuário seleciona o menu Cadastrar Manualmente; 2. O usuário informa o endereço MAC do Dispositivo ESP; 3. O usuário informa a Localização do Dispositivo ESP; 4. O usuário informa uma Observação para o Dispositivo ESP; 5. O usuário clica em Cadastrar.
Cenário de exceção 1	No passo 2, se for informado um endereço MAC já cadastrado, ao clicar em gravar no passo 5, será exibida uma mensagem informando que o endereço MAC já está cadastrado e o registro não será gravado.
Pós-condição	O sistema apresenta uma mensagem informando que a configuração foi alterada com sucesso

O caso de uso `UC03 – Cadastrar Dispositivo ESP automaticamente` (Quadro 16) demonstra como o ator Usuário cadastra um Dispositivo ESP sem precisar informar o endereço MAC.

Quadro 16 - Detalhamento do `UC03 – Cadastrar Dispositivo ESP automaticamente`

Número	03
Caso de uso	Cadastrar Dispositivo ESP automaticamente
Ator	Usuário
Cenário principal	1. O usuário seleciona o menu Cadastrar automaticamente; 2. O sistema procura pelos Dispositivos ESP na rede; 3. O sistema encontra os Dispositivos ESP e os coloca em uma tabela. Cada linha da tabela terá um campo `Local`, um campo `Observação` e um botão `Cadastrar`; 4. O usuário informa o local para o Dispositivo ESP; 5. O usuário clica no botão Cadastrar; 6. O sistema troca o botão Cadastrar pelo botão Remover, indicando que o registro está gravado.
Cenário alternativo 1	No passo 3, caso o Dispositivo ESP já esteja cadastrado, o botão que irá aparecer será o Remover ao invés do Cadastrar e o fluxo continua no passo 6

O caso de uso `UC04 – Monitorar Dispositivo ESP` (Quadro 17) demonstra como o ator Usuário monitora os Dispositivos ESP.

Quadro 17 - Detalhamento do UC04 – Monitorar Dispositivo ESP

Número	04
Caso de uso	Monitorar Dispositivo ESP
Ator	Usuário
Cenário principal	1. O usuário seleciona o menu Monitoramento; 2. O sistema busca os Dispositivos ESP e exibe na tela; 3. O sistema procura pelos Dispositivos ESP na rede; 4. O sistema encontra e altera o status do Dispositivo ESP para Encontrado; 5. O sistema tenta se conectar ao Dispositivo ESP; 6. O sistema se conecta e altera o status para Conectado; 7. O sistema atualiza o tempo que a sala do Dispositivo ESP se encontra sem movimento; 8. Retornar para o passo 7.
Cenário alternativo 1	No passo 4, se o Dispositivo ESP não for encontrado: 4.1. O sistema muda o status para Não encontrado; 4.2. O usuário clica no comando Buscar novamente; 4.3. O sistema tenta buscar novamente todos os Dispositivos ESP que estão com o status Não encontrado; 4.4. Retornar para o passo 4.
Cenário alternativo 2	No passo 6, se o sistema não conseguir se conectar: 6.1. O sistema muda o status para Não conectado; 6.2. O sistema aguarda alguns segundos; 6.3. O sistema tenta se conectar novamente; 6.4. Retornar para o passo 6.
Cenário alternativo 3	No passo 7, se a sala ultrapassar o tempo limite sem movimento cadastrado: 7.1. O sistema avisa o Dispositivo ESP que o mesmo deve desligar os equipamentos da sala; 7.2. O Dispositivo ESP envia o comando 01 armazenado; 7.3. O Dispositivo ESP envia o comando 02 armazenado se o equipamento que se deseja desligar estiver ligado; 7.4. Retornar para o passo 8.
Cenário alternativo 4	No passo 7, se a conexão cair: 7.1. O sistema muda o status para Erro na conexão; 7.2. Retornar para o passo 5.

O caso de uso UC05 – Enviar comando para desligar (Quadro 18) demonstra como o ator Usuário pode enviar o comando de desligar para o Dispositivo ESP manualmente.

Quadro 18 - Detalhamento do UC05 – Enviar comando para desligar

Número	05
Caso de uso	Enviar comando para desligar
Ator	Usuário
Pré-condições	Estar com o menu Monitoramento aberto; Estar com o status do Dispositivo ESP como Conectado.
Cenário principal	1. O usuário clica no botão Desligar; 2. O sistema notifica o Dispositivo ESP que o mesmo deve desligar os equipamentos do local.
Pós-condições	Ao clicar no comando Histórico, deve aparecer um registro informando que um comando Desligar foi enviado.

O caso de uso UC06 – Cadastrar Comandos IR no Dispositivo ESP (Quadro 19) demonstra como o ator Usuário cadastra os comandos infravermelho no Dispositivo ESP.

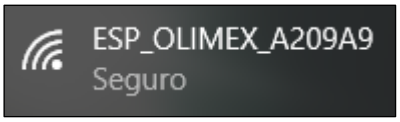
Quadro 19 - Detalhamento do UC06 - Cadastrar Comandos IR no Dispositivo ESP

Número	06
Caso de uso	Cadastrar Comandos IR no Dispositivo ESP
Ator	Usuário
Cenário principal	1. O usuário pressiona o comando 03 2. O usuário pressiona o comando 01; 3. O usuário clica sobre o comando desligar o controle remoto do equipamento desejado apontando para o Dispositivo ESP; 4. O Arduino do Dispositivo ESP armazena o comando 01 recebido na memória EEPROM; 5. O usuário solta o comando 01; 6. O usuário pressiona o comando 02; 7. O usuário clica sobre o comando desligar o controle remoto do equipamento desejado apontando para o Dispositivo ESP; 8. O Arduino do Dispositivo ESP armazena o comando 02 recebido na memória EEPROM; 9. O usuário solta os comandos 02 e 03.

APÊNDICE B – Configurando o ESP8266-EVB

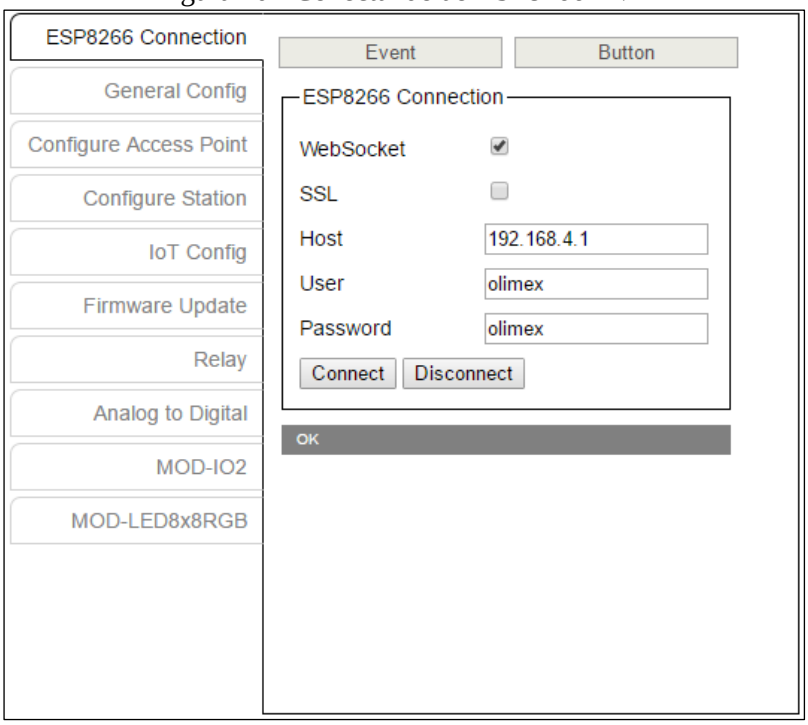
Primeiramente, deve-se instalar o *firmware* da Olimex no ESP8266-EVB. Conforme a Olimex (2015a) o *firmware* pode ser encontrado no seu repositório ESP8266 no GitHub. Depois do firmware instalado o ESP8266-EVB vem configurado por padrão como *Station* e *Access Point*. Então para configurá-lo, deve-se conectar a ele utilizando um SSID que será algo semelhando à Figura 19 e a senha é por padrão `olimex-ap`.

Figura 19 - *Access Point* do ESP8266-EVB



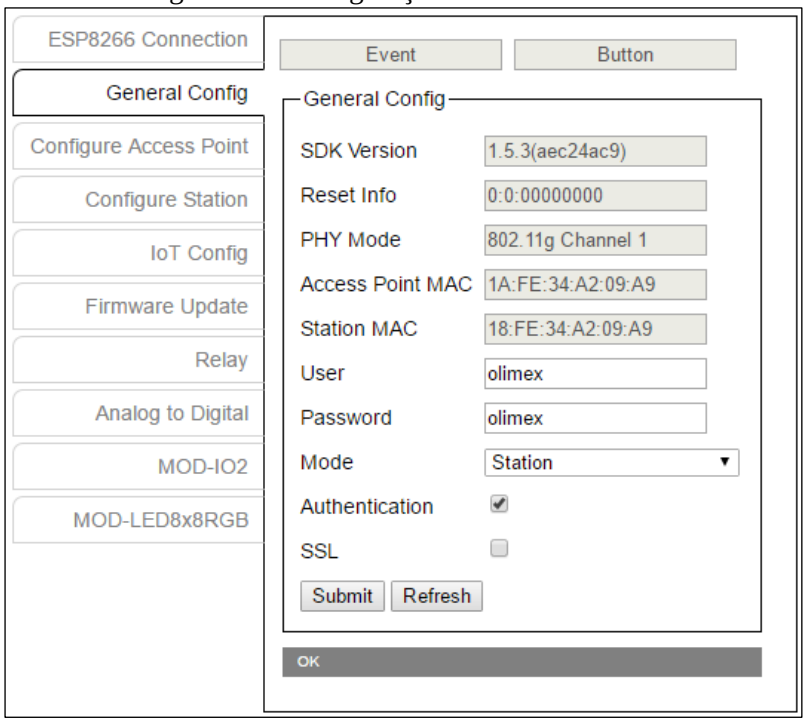
Depois de conectado à rede, deve-se configurar o ESP8266-EVB. Para isso foi utilizada uma aplicação disponibilizada pela própria Olimex feita em HTML e JavaScript que pode ser vista na Figura 20. Essa aplicação se encontra no mesmo repositório que está o *firmware* do ESP8266-EVB. Para realizar as configurações, deve-se conectar a ele utilizando o IP `192.168.4.1`, usuário `olimex` e senha `olimex`, conforme a Figura 20.

Figura 20 - Conectando ao ESP8266-EVB



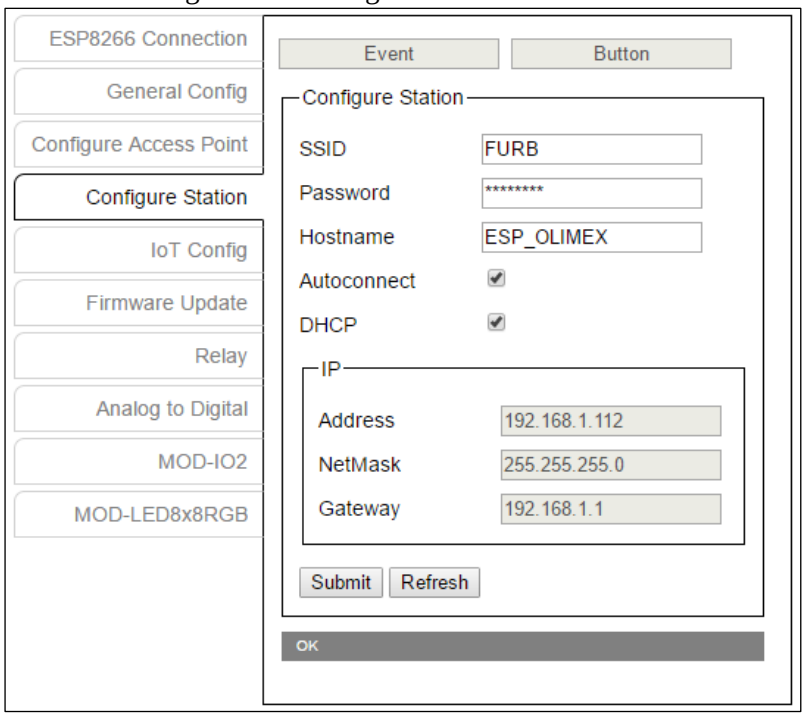
Depois, deve-se alterar o usuário e senha do ESP8266-EVB se necessário. Também deve-se configurar o campo `Mode` para `Station`. Dessa forma, ele não ficará mais disponível como `Acess Point`. Também deve-se marcar o campo `Authentication` e desmarcar o campo `SSL` conforme a Figura 21.

Figura 21 - Configuração do ESP8266-EVB



Depois deve-se configurar o ESP8266-EVB para se conectar à rede desejada, conforme a Figura 22, inserindo o `SSID` e senha da rede para se conectar.

Figura 22 - Configurando o modo Station



APÊNDICE C – Peças utilizadas e seus preços

A Tabela 1 contém todo o *hardware* que foi utilizado no desenvolvimento do protótipo e seus preços aproximados. Foi desconsiderado os materiais para a construção como soldador, solda, energia elétrica, cola quente entre outros. Os valores foram extraídos dos sites de componentes eletrônicos Proesi e FILIPEFLOP.

Tabela 1 - Custo do protótipo

Peça	**Quantidade**	**Valor unitário**	**Total**
Arduino Uno	1	60,00	60,00
ESP8266-EVB da Olimex	1	40,00	40,00
MOD-IO2 da Olimex	1	50,00	50,00
HC-SR501 Sensor de Movimento PIR Brick	1	17,00	17,00
Placa Fenolite Perfurada 12x18cm	1	20,00	20,00
Resistor 470Ohms	1	0,02	0,02
Resistir 1kOhm	5	0,02	0,10
Resistor 10kOhms	2	0,02	0,04
Foto Diodo Emissor Infravermelho 5mm	1	0,89	0,89
IRM3638 3V Foto Receptor Infravermelho	1	6,34	6,34
Led Difuso 3mm	2	0,17	0,34
Chave Táctil KFC-A06 2T	3	0,20	0,60
Transistor - PN2222 / 2N2222	1	0,22	0,22
Fio Jumper 2X0,5mm	3 metros	0,80	2,40
Capacitor Eletrolítico 10µF 25V	1	0,25	0,25
Sensor de corrente não invasivo 20A SCT-013	1	70,00	70,00
		Total	**268,20**