

Henrique Sussumu Matsui Kano & Mi Che Li Lee

Sistema de Monitoramento Residencial de Uso de Energia

São Paulo

2015

Henrique Sussumu Matsui Kano & Mi Che Li Lee

Sistema de Monitoramento Residencial de Uso de Energia

Trabalho de formatura apresentada à Escola Politécnica da Universidade de São Paulo para a conclusão do curso de graduação em Engenharia de Computação

Escola Politécnica da Universidade de São Paulo
Departamento de Engenharia da Computação e Sistemas Digitais

Orientador: Professor Doutor Carlos Eduardo Cugnasca

São Paulo
2015

Catalogação-na-publicação

Kano, Henrique

Sistema de monitoramento residencial de uso de energia / H. Kano, M. Lee –
São Paulo, 2015.

85 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. De-
partamento de Engenharia de Computação e Sistemas Digitais.

1. SENSOR. 2. COMPUTAÇÃO MÓVEL. 3. CONSUMO DE ENERGIA
ELÉTRICA. 4. INTERNET. I. Universidade de São Paulo. Escola Politécnica.
Departamento de Engenharia de Computação e Sistemas Digitais II. t. III. Lee, Mi

Agradecimentos

Agradeço minha família pelo suporte emocional.

Ao professor orientador pela orientação e dedicação constante ao trabalho.

Aos técnicos Fátima, Daniel do labdigi e Marcelo do labmicro pelo suporte técnico dado.

Ao pessoal da DevTecnologia pela orientação inicial.

Aos meus amigos que sempre deram o suporte nas horas difíceis.

Henrique Sussumu Matsui Kano

Enfim, a jornada foi longa e sem ajuda eu não teria chegado até aqui. Por isso eu quero dizer que estou muito grata por tudo que aconteceu, seja fácil ou difícil. Eu sou grata pelas minhas amizades formadas desde o primeiro ano da Poli. Quero agradecer pelo constante apoio da minha família. Admiro a paciência dos técnicos dos laboratórios do PCS. Sou grata pelo nosso orientador e por todos os nossos professores, por compartilhar suas experiências, nos ajudando da melhor forma possível. E sou muitíssimo grata pela minha igreja e a Deus, por ter me provido tudo que eu precisava.

Mi Che Li Lee

Resumo

O trabalho propõe a construção de um sistema de monitoramento de consumo de energia em uma residência. O sistema é composto por dispositivos interligados por uma rede sem fio e uma aplicação Web. O trabalho também propõe o uso de serviços, recursos e ferramentas gratuitos para o desenvolvimento de uma interface gráfica para visualização dos dados de consumo.

Palavras-chaves: IoT. Rede de Sensores sem fio. Computação em nuvem. Consumo de Energia Elétrica.

Abstract

This work proposes a system for residence energy consumption monitoring with a low investment cost. The system consists of devices connected by a wireless network and a Web application. The proposed system uses services, resources and tools that converges to a user friendly interface so that data collected may be easily analyzed by the user.

Key-words: IoT. Wireless Sensor Network. Cloud Computing. Electric Power Consumption.

Listas de ilustrações

Figura 1 – Desperdício de Energia no Brasil	19
Figura 2 – Arquitetura MVC	21
Figura 3 – Topologia de uma rede zigbee	23
Figura 4 – Esquema do Projeto	27
Figura 5 – Diagrama de implantação	28
Figura 6 – Circuito verificador de tensão	30
Figura 7 – Circuito medidor de corrente	30
Figura 8 – Non-invasive AC current sensor	31
Figura 9 – Raspberry pi 2 modelo B	32
Figura 10 – Arduino UNO R3	33
Figura 11 – XBee Serie 2	34
Figura 12 – XBee Explorer Dongle	35
Figura 13 – XBee shield do arduino UNO	35
Figura 14 – Headers usados no XBee shield	36
Figura 15 – Diagrama de Classes	53
Figura 16 – Diagrama de Classes - Models	58
Figura 17 – Diagrama de Classes - Controllers Principais	59
Figura 18 – Diagrama de Classes - Controllers de Equipamentos	59
Figura 19 – Diagrama de Classes - Controllers de Sensores	60
Figura 20 – Diagrama de Classes - Controllers de Taxas da AES	60
Figura 21 – Diagrama de Classes - Controllers de Configuração	61
Figura 22 – Diagrama de Classes - Controllers de Metas	61
Figura 23 – Diagrama de Classes - Controllers de Consumos	62
Figura 24 – Adaptador wifi usado no raspberry	63
Figura 25 – Teste de comunicação: Módulo Coordenador	64
Figura 26 – Teste de comunicação: Módulo Sensor	64
Figura 27 – Teste de comunicação: Montagem	65
Figura 28 – Teste de comunicação Raspberry e Arduino	66
Figura 29 – Teste do transformador	66
Figura 30 – Teste de medição de tensão	67
Figura 31 – Teste de medição de tensão com arduino - saída do osciloscópio	67
Figura 32 – Medição com amperímetro	68
Figura 33 – Teste de medição da corrente e tensão	70
Figura 34 – Teste de criação de sensor e de consumo	74
Figura 35 – Listagem de equipamentos	75

Figura 36 – Criar meta	76
Figura 37 – AES	76
Figura 38 – Configurações	77
Figura 39 – Gráfico de consumo	77

Lista de tabelas

Tabela 1 – Casos de Uso.	40
Tabela 2 – Orçamento do projeto.	85

Sumário

1	INTRODUÇÃO	17
1.1	Objetivo	18
1.2	Motivação	18
1.3	Justificativa	19
1.4	Organização	20
2	ASPECTOS CONCEITUAIS	21
2.1	Arquitetura MVC	21
2.2	Wireless Sensor Network	22
2.3	Padrão ZigBee e o XBee	22
2.4	Topologias de Rede	23
3	METODOLOGIA	25
4	ESPECIFICAÇÃO DO PROJETO	27
4.1	Escopo	27
4.2	Requisitos não Funcionais	28
4.3	Premissas	28
4.4	Hardware	29
4.4.1	Módulo sensor	29
4.4.2	Coordenador	29
4.4.3	Circuitos	29
4.4.3.1	Verificador de Tensão	29
4.4.3.2	Medidor de Corrente	30
4.4.4	Peças	31
4.4.4.1	Sensor de Corrente Não-invasivo AC	31
4.4.4.2	Raspberry Pi 2 modelo B	32
4.4.4.3	Arduino UNO	33
4.4.4.4	XBee	34
4.4.4.5	XBee Explorer Dongle	35
4.4.4.6	XBee Shield	35
4.4.4.7	Arduino Stackable Header Kit - R3	36
4.5	Software	36
4.5.1	Tecnologia	36
4.5.1.1	Django e Python	37
4.5.1.2	Heroku	38

4.5.2	Funções do Sistema	38
4.5.3	Atores	39
4.5.4	Casos de uso	39
4.5.5	Descrição dos casos de uso	40
4.5.5.1	Função 1: Gerenciar conta	40
4.5.5.2	Caso de Uso 1.1: Fazer cadastro	40
4.5.5.3	Caso de Uso 1.2: Fazer login	41
4.5.5.4	Caso de Uso 1.3: Fazer logout	42
4.5.5.5	Caso de Uso 1.4: Recuperar senha	42
4.5.5.6	Função 2: Gerenciar equipamentos	43
4.5.5.7	Caso de Uso 2.1: Criar equipamento	43
4.5.5.8	Caso de Uso 2.2: Editar equipamento	44
4.5.5.9	Caso de Uso 2.3: Remover equipamento	44
4.5.5.10	Função 3: Gerenciar sensores	45
4.5.5.11	Caso de Uso 3.1: Detectar sensor	45
4.5.5.12	Caso de Uso 3.2: Editar sensor	46
4.5.5.13	Caso de Uso 3.3: Remover sensor	46
4.5.5.14	Função 4: Gerenciar metas	47
4.5.5.15	Caso de Uso 4.1: Criar meta	47
4.5.5.16	Caso de Uso 4.2: Editar meta	48
4.5.5.17	Caso de Uso 4.3: Remover meta	49
4.5.5.18	Função 5: Gerenciar consumos	49
4.5.5.19	Caso de Uso 5.1: Criar consumo	49
4.5.5.20	Caso de Uso 5.2: Visualizar consumo	50
4.5.5.21	Caso de Uso 5.3: Importar consumos	50
4.5.5.22	Caso de Uso 5.4: Exportar consumos	51
4.5.5.23	Caso de Uso 6: Atualizar taxas da AES	51
4.5.5.24	Caso de Uso 7: Configurar sistema	52
4.5.6	Classes e atributos	52
4.5.6.1	Equipment	53
4.5.6.2	Sensor	54
4.5.6.3	Consumption	54
4.5.6.4	User	55
4.5.6.5	Goal	55
4.5.6.6	AESRate	56
5	IMPLEMENTAÇÃO	57
5.1	Aplicação Web	57
5.1.1	Models e Controllers	58

5.2	Módulo Sensor e Módulo Coordenador	62
5.2.1	XBee: configuração	62
5.2.1.1	Módulo Sensor:	62
5.2.1.2	Módulo Coordenador:	62
5.2.2	Raspberry: Sistema Operacional	63
5.2.3	Raspberry: Adaptador Wifi	63
5.2.4	Raspberry x Arduino: comunicação	64
5.2.5	Verificador de tensão	66
5.2.6	Sensor de corrente	67
5.2.7	Integrando as partes	70
6	TESTES E AVALIAÇÃO	73
6.1	Introdução	73
6.2	Plano de Testes do Software	73
6.2.1	Funcionalidades Críticas	73
6.2.2	Funcionalidades Não-Críticas	74
6.3	Integração	74
6.4	Resultados	75
6.4.1	Telas principais	75
6.4.1.1	Listagem de equipamentos	75
6.4.1.2	Criar meta	76
6.4.1.3	AES	76
6.4.1.4	Configurar sistema	77
6.4.1.5	Visualização dos dados medidos	77
7	CONSIDERAÇÕES FINAIS	79
8	TRABALHOS FUTUROS	81
	REFERÊNCIAS	83
	APÊNDICE A – ORÇAMENTO	85

1 Introdução

Escassez de recursos, baixa remuneração, custos crescentes, disponibilidade de recursos hídricos e o desperdício fazem com que a economia de energia elétrica no Brasil seja vista como uma necessidade para os moradores de cidades (1). Grande parte do consumo mensal de energia elétrica é manifestada através do uso de eletrônicos. Portanto são comuns as ideias de compra ou troca de equipamentos por outros mais eficientes quanto ao consumo de energia, ou mudanças nos hábitos de uso destes nas residências. Entretanto, surgem dúvidas: “Qual equipamento devo trocar? Por que o valor da conta de luz veio alta mesmo tendo desligado todos os equipamentos antes de viajar? Será que eu deixei o ferro de passar ligado em casa?” Muitas vezes os moradores de residência não têm as informações necessárias para tomar decisões seguras e proativas ou as informações não estão disponíveis em um formato compreensível.

Há ferramentas existentes que podem ajudar a população na área de controle de consumo de energia elétrica como OpenEnergyMonitor (2) e Neurio (3). Ambas possuem dispositivos para serem instalados diretamente no quadro elétrico, e no caso da Neurio, é possível estimar o consumo por equipamento doméstico. Porém, como garantir que o consumo é de fato daquele equipamento em específico, e não apenas um ruído vindo de outro equipamento? O objetivo do projeto é o desenvolvimento de um sistema de monitoramento de consumo de energia em uma residência. No sistema há dispositivos de medição de consumo por equipamento, ou seja, o usuário não precisa manusear o quadro elétrico, basta conectar os dispositivos medidores aos equipamentos desejados. E além disso, esses dispositivos são interligados por uma rede sem fio, para garantir flexibilidade, facilidade de instalação e robustez, provendo conforto para o ambiente residencial.

Tendo os dados de consumo ao alcance do usuário também não é suficiente, é necessário mostrar transparência no perfil de consumo de seus equipamentos. Para isso, o projeto inclui o desenvolvimento de uma aplicação Web, onde é possível mostrar o consumo através de gráficos, além de outras funcionalidades como conversão de consumo de energia para unidades monetárias e traçar metas de consumo. Essas funcionalidades são alcançáveis remotamente através de serviços em nuvem utilizados, pertencentes à empresa Heroku. Através de dispositivos portáteis que o usuário tenha em mãos como smartphones, tablets e notebooks, este pode utilizar a aplicação Web em qualquer lugar com acesso à internet.

O desenvolvimento do projeto é descrito neste trabalho, sempre dando prefe-

rência para o uso de hardware e software open-source, framework e linguagens de programação gratuitos. Para verificar o funcionamento do sistema, um protótipo do sistema é construído e os resultados estão descritos no trabalho.

1.1 Objetivo

O trabalho descreve o desenvolvimento de um sistema para monitorar o uso de energia elétrica dentro de residência. O sistema consiste de dispositivos interligados por uma rede sem fio, para a coleta de medidas de consumo por equipamento e uma aplicação em nuvem, que apresenta as medidas coletadas através de uma interface simples e intuitiva. Além disso é construído um protótipo para cada componente do sistema.

1.2 Motivação

Em 2015, devido à escassez de chuvas, houve uma queda significativa no nível dos reservatórios das principais hidrelétricas do Brasil e o uso mais intenso de termelétricas. Isso provocou reajustes altos, encarecendo a energia elétrica no país e o custo foi repassado para os consumidores finais, como se pode verificar nos jornais: G1 (4) e O Estado de S. Paulo (5).

Segundo relatório da COPAM/EPE (6), o consumo de energia elétrica pela região Sudeste corresponde a 51,73% da matriz energética do Brasil, ou seja, é a região geográfica que gasta mais energia elétrica. Analisando o estado de São Paulo, o município com maior participação no consumo estadual de energia elétrica é São Paulo, consumindo 39,90% do estado (7). Além disso, notícia da Folha (8) revela que de toda energia consumida em 2014, mais de 10% foi desperdício, dos quais metade corresponde a perdas geradas dos consumidores residenciais (figura 1) através dos eletrodomésticos. Por isso foi escolhido o setor residencial na região metropolitana de São Paulo como foco do projeto.

Dado o cenário desfavorável à geração de energia elétrica e o grande desperdício do consumidor residencial, é imprescindível a tomada de decisões por parte da população. O sistema proposto é composto por dispositivos medidores e de uma aplicação Web que dará ao usuário o conhecimento do perfil de consumo dos seus equipamentos, auxiliando-o a tomar atitudes para diminuição do consumo de energia.

No mercado há várias tecnologias disponíveis para o desenvolvimento do sistema proposto. Para a medição do consumo no equipamento existem sensores, micro-

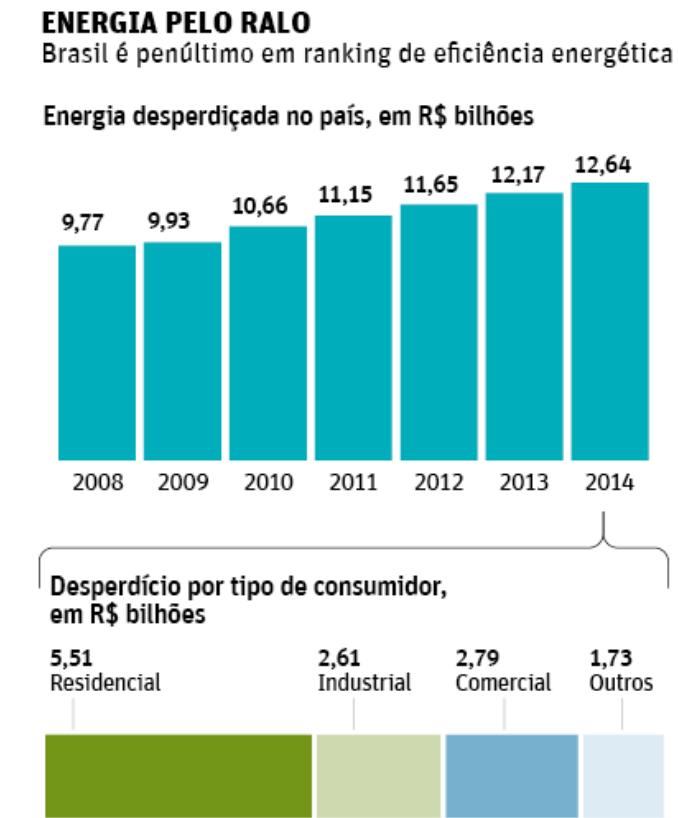


Figura 1 – Desperdício de Energia no Brasil

controladores e dispositivos que permitem comunicação sem fio e adaptadores para acesso à internet. Na parte de software, há linguagens de programação gratuitas para o desenvolvimento da aplicação Web, frameworks para agilizar o desenvolvimento e serviços em nuvem para armazenamento e gerenciamento de banco de dados. A existência de tecnologias acessíveis para resolver o problema atual de consumo de energia das residências da região metropolitana de São Paulo motivou o desenvolvimento desse projeto.

1.3 Justificativa

O sistema proposto atende as necessidades do usuário de várias formas. Através dos gráficos de consumo por equipamento, o usuário poderá perceber que os seus equipamentos em *standby* consomem energia significativa ao longo do tempo. Além disso pode-se detectar possíveis defeitos técnicos nos equipamentos e que, por isso, tendem a consumir mais energia do que deveriam.

Os gráficos do sistema associados a disponibilidade proporcionada pela nuvem, a aplicação Web permite ao usuário monitorar os seus equipamentos a partir de qual-

quer dispositivo móvel que tenha em mãos com acesso à internet. Com isso, é possível detectar fora da residência aparelhos ligados sem necessidade, por exemplo, ar-condicionado numa sala vazia.

O sistema pode ser utilizado como parte da residência, monitorando 24 horas por dia o consumo dos equipamento, ou pode ser utilizado como um serviço. O usuário contrata o serviço e os dispositivos de medição são instalados em sua residência por um período determinado. Após esse período, são feitas avaliações dos consumos por uma equipe técnica, que poderá recomendar a troca ou a diminuição de consumo de determinados equipamentos apontados pelo sistema.

Tendo em vista as aplicações possíveis para o sistema e os benefícios que o sistema traz para o usuário, o desenvolvimento desse sistema é justificável.

1.4 Organização

O documento segue o seguinte formato: no capítulo dois são apresentados os conceitos estudados para o desenvolvimento do projeto.

No capítulo três é apresentado o método adotado pelos integrantes do projeto para a realização do projeto do início ao fim.

No capítulo quatro é apresentada a especificação do sistema para o hardware e software.

No capítulo cinco o processo de implementação do projeto é detalhado.

No capítulo seis é detalhado o procedimento de aceitação do sistema através da aplicação de um plano de testes.

No capítulo sete são apontadas algumas considerações finais como aprendizado, dificuldades e resultados atingidos.

No capítulo oito são citadas algumas ideias que poderiam ser aplicadas a trabalhos futuros.

2 Aspectos Conceituais

As técnicas de monitoramento e sensoriamento podem ser usados para as mais diversas funções e implementadas de diversas maneiras. Nesse trabalho, são usadas técnicas, arquiteturas e tecnologias para monitorar o consumo de energia elétrica em um ambiente residencial por uma rede de sensores na qual cada sensor da rede transmite seus dados a um componente central que se comunica com uma aplicação em nuvem (9).

São abordados vários conceitos vistos em aula. Um deles engloba o universo dos protocolos e componentes de uma rede sem fio. Isso envolve o estudo do protocolo que será utilizado nesse projeto, que é o ZigBee, devido ao seu grande uso na área o que implica em uma grande fonte de informações sobre este(10)(11)(9). Junto a isso são estudados a montagem de circuitos de sensores associados a microcontroladores e a captação dos dados dos sensores por uma central.

Além disso é estudado o desenvolvimento de aplicações para Web e arquitetura de sistemas.

2.1 Arquitetura MVC

A arquitetura de software utilizada é a MVC, composta pelas camadas Model, View e Controller (figura 2)

Model: A camada Model representa a primeira camada de interação com qualquer banco de dados que possa estar sendo usado na aplicação. Ela é responsável por obter, processar, validar dados do banco de dados.

View: A view é responsável por usar as informações disponibilizadas para pro-

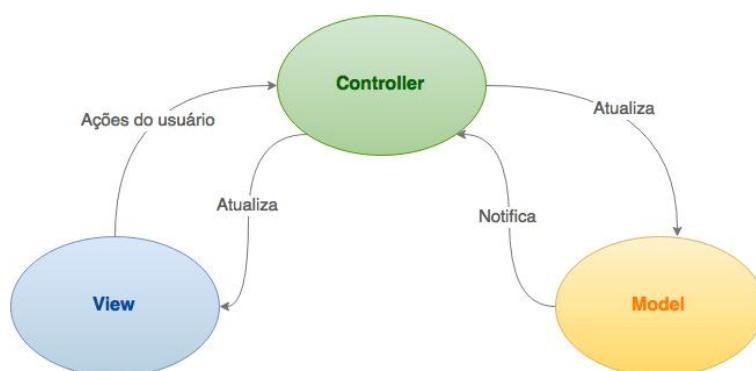


Figura 2 – Arquitetura MVC

duzir qualquer interface de apresentação que a aplicação pode necessitar.

Controller: essa camada lida com as requisições dos usuários. É responsável por se comunicar com a camada Model para realizar operações de busca ou armazenamento de dados e repassar os dados obtidos para a camada View, que irá gerar uma saída resultante para o usuário.

MVC é um padrão de projeto de software recomendado para aplicações de desenvolvimento rápido, de fácil manutenção e modular. E foi escolhido como o ideal para um projeto com um tempo limitado de desenvolvimento e possibilita a divisão fácil de tarefas entre os membros da equipe

2.2 Wireless Sensor Network

Wireless Sensor Network, também conhecido como WSN, é um termo genérico que descreve sistemas que tem como objetivo o sensoriamento e monitoração de algum objeto em uma certa área, em pelo menos uma variável (temperatura, umidade, pressão, cor, etc). Os desafios de tais estruturas se resumem a (12): sensores e nós que compõem a rede e podem ter a função de sensoriamento ou de retransmitir dados a um outro nó ou à estação base para serem salvos e devem formar sozinhos uma rede que consiga garantir que os dados sensoriados chegem à base, protocolo de comunicação entre nós, que podem afetar significantemente no consumo de energia, atrasos de comunicação e eficiência do sistema como um todo, fontes de energia dos nós limitada e soluções de coleta de energia.

2.3 Padrão ZigBee e o XBee

O padrão ZigBee e o dispositivo XBee possuem muitas características configuráveis e que podem servir a várias aplicações(10), porém duas delas são de maior interesse para o trabalho: o baixo consumo de energia e os modos de operação. O XBee pode ser configurado para operar em um dos dois modos: AT ou API. No modo AT há apenas o envio de dados ponto-a-ponto na rede, porém, no modo API, é possível agir na rede durante sua operação com mudanças de configuração de nós, broadcast, confirmação de entrega de pacotes e identificação do endereço dos dados recebidos, o que dá ao sistema um maior controle do todo (13)

2.4 Topologias de Rede

Como os nós dos sensores formam uma rede, é necessário analisar as possibilidades de redes. Como são usados XBee para formar essa rede de sensores, deve-se atentar aos tipos de rede possíveis (10) (13).

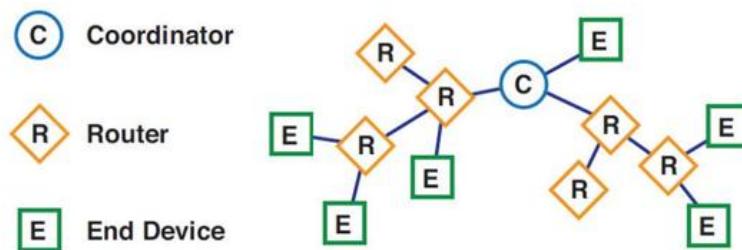


Figura 3 – Topologia de uma rede zigbee

fonte: <http://ftp1.digi.com/support/documentation/html/90001399/90001399_A/Files/XBee-concepts.html>

A rede ZigBee é composta por nós que podem ser de três tipos:

- Coordenador: Nô destino (final) de todos os outros nós, concentrando os dados de todos os nós. Todas as redes possuem apenas um nó desse tipo e como esse tipo de nó não possui a capacidade de dormir ele não deveria ficar em um dispositivo com uma bateria limitada.
- Roteador: Funciona apenas como uma ponte intermediária entre os endpoints e o coordenador, podendo se comunicar com todos os outros tipos de nós, mas também não possuem a capacidade de dormir, logo não podem ser energizados com uma bateria limitada. Normalmente são usados para estender a área de uma rede, aumentando o alcance sensoreada da rede como um todo.
- Dispositivos finais: Nós que são as pontas da rede e que normalmente estão ligados aos sensores. Esses tem a capacidade de dormir e conservar energia enquanto não transmitem e só não conseguem se comunicar com outros nós do mesmo tipo diretamente;

Dadas essas especificações e limitações, as redes formadas por esses componentes podem ser de três tipos (10): estrela, árvore ou mesh. Na estrela os dispositivos finais conversam diretamente com o coordenador, na rede mesh os dispositivos finais estão intermediados por uma malha de roteadores que se organizam para encontrar o melhor caminho de roteadores de um dispositivo final até o coordenador e a árvore é um subcaso da rede mesh onde, devido a bloqueios físicos ou distâncias entre os roteadores, a rede acaba por se tornar uma

árvore ou um grafo onde o caminho de um dispositivo final até o coordenador praticamente não possui alternativas.

3 Metodologia

O trabalho foi desenvolvido com base em uma pesquisa de trabalhos acadêmicos, convergindo em um protótipo funcional. Para tanto, o trabalho foi dividido em algumas etapas principais:

- Etapa 1: Pesquisa de trabalhos relacionados, agregação da base teórica envolvida;
- Etapa 2: Reflexão sobre possíveis caminhos alternativos e desenvolvimento de planos paralelos redundantes;
- Etapa 3: Projeto do sistema, levantamento de peças necessárias, orçamento, preparação do ambiente de trabalho com ferramentas de depuração e testes;
- Etapa 4: Implementação prática do planejamento feito na etapa anterior;
- Etapa 5: Testes comparativos com os resultados esperados;
- Etapa 6: Finalização do projeto e do protótipo;

Com essa metodologia, foi possível levantar os requisitos do sistema sem se desviar do objetivo do projeto, tendo sempre um caminho paralelo a seguir caso ocorresse algum inesperado.

4 Especificação do Projeto

O sistema é composto por duas partes: hardware, ou seja, os componentes como microcontroladores, sensores, entre outros; e software, identificado como o aplicativo de interface entre o usuário e os dados coletados.

O sistema está brevemente descrito através da figura 4 e a disposição dos componentes através do diagrama de implantação da figura 5:

A parte de hardware está separada em dois módulos principais: o módulo de sensoriamento e um módulo coordenador e a parte de software se resume à parte da aplicação web na nuvem.

4.1 Escopo

Coletados os dados, resta mostrar informações úteis ao usuário. Com o consumo de corrente e a tensão da tomada, é possível calcular a potência e alguns outros dados interessantes para o usuário. O aplicativo desenvolvido nesse trabalho é responsável por esse tratamento e a visualização dos dados coletados pelos sensores. As principais metas são as seguintes:

- informar ao usuário sobre o consumo de energia por equipamento em sua residência
- auxiliar o usuário a tomar decisões para diminuir o consumo de energia

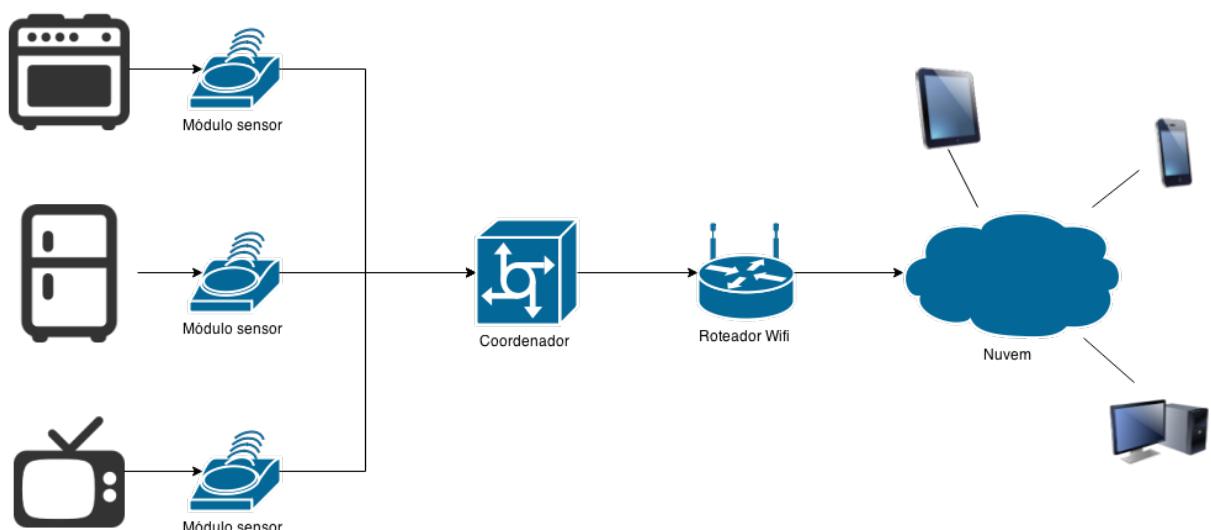


Figura 4 – Esquema do Projeto

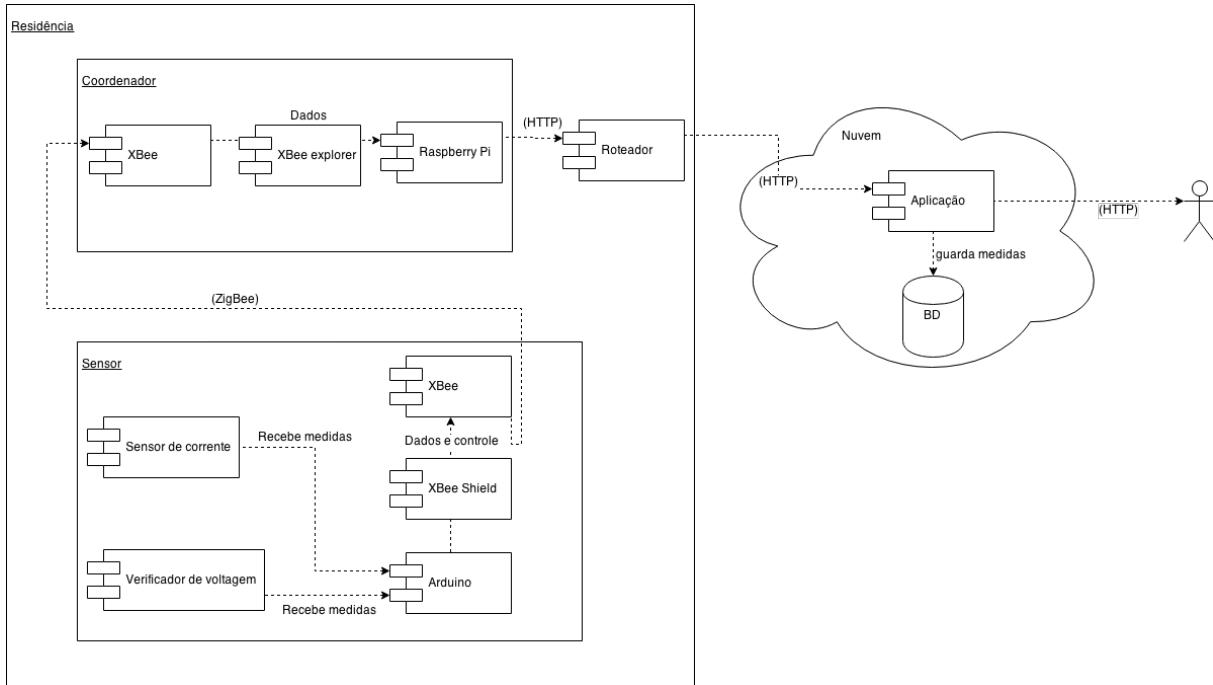


Figura 5 – Diagrama de implantação

- permitir o acesso às informações de consumo tanto localmente quanto remotamente

4.2 Requisitos não Funcionais

- independência do usuário em relação ao técnico do sistema para instalar o sistema em sua residência
- sistema de fácil manuseio pelo usuário morador da residência
- característica portátil para os componentes físicos do sistema

4.3 Premissas

Sendo esse um projeto que visa o sensoriamento e de um monitor para visualizar os dados com o objetivo de dar uma visão geral ao usuário sobre gastos supérfluos e uma relação absoluta do consumo de cada equipamento, o sistema é influenciado por alguns fatores físicos e geopolíticos, o que leva à necessidade de usar algumas premissas que tiveram de ser feitas para ajustar o projeto ao tempo previsto e garantir o funcionamento correto do sistema:

1. O usuário deve morar em São Paulo

2. Será considerado um fator de potência ideal unitário

4.4 Hardware

4.4.1 Módulo sensor

O módulo sensor vai ser responsável por medir e transmitir as informações necessárias para calcular o consumo de energia do equipamento acoplado.

Os componentes físicos do módulo sensor são:

- Circuito Verificador de tensão
- Sensor de Corrente (Non-invasive AC Current Sensor)
- Arduino Uno - R3
- XBee Shield
- XBee 2mW PCB Antenna - Series 2

4.4.2 Coordenador

O módulo coordenador vai ser responsável por fazer requisições para os módulos sensores, tratar os dados de consumo e enviar ao aplicativo na nuvem.

Os componentes do coordenador são:

- Kit Raspberry Pi2 + Fonte + Microsd 8gb + Wifi Usb
- XBee Explorer Dongle
- XBee 2mW PCB Antenna - Series 2

4.4.3 Circuitos

4.4.3.1 Verificador de Tensão

No circuito de cada módulo de sensor, são feitas detecções da tensão (127V ou 220V) para cálculos de potência. O objetivo do circuito da figura 6 é indicar se a tensão na tomada é 220V ou 127V. A saída do circuito é usada como um valor analógico, que

dependendo da tensão de entrada resultará em faixas diferentes para as diferentes tensões.

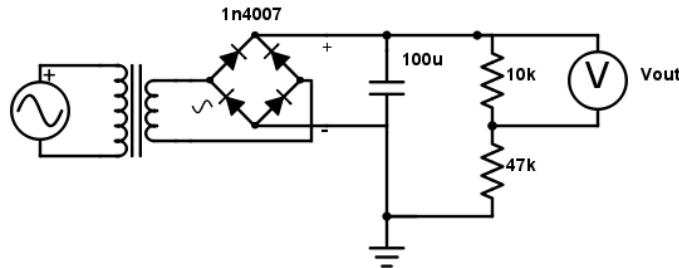


Figura 6 – Circuito verificador de tensão

4.4.3.2 Medidor de Corrente

Ainda no módulo sensor, é necessário obter as medidas do valor eficaz da corrente. O sensor não-invasivo produz uma tensão alternada na saída, e antes da coleta de dados pelo arduino é preciso obter um valor significativo, que não ultrapasse 2.5V. Para isso foi utilizado o circuito da figura 7.

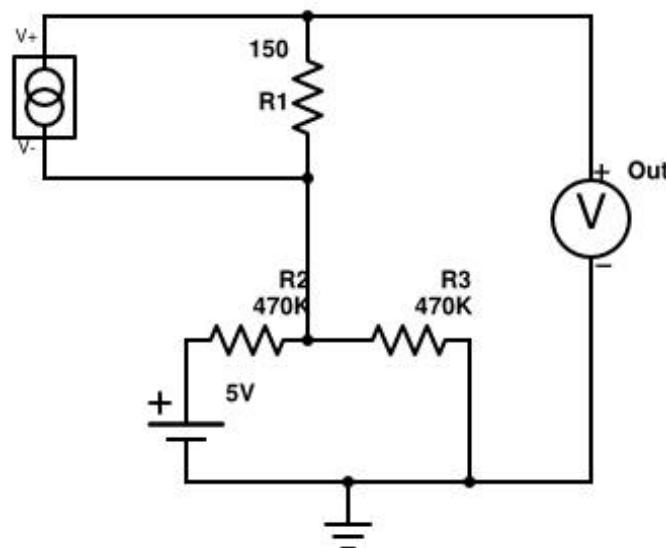


Figura 7 – Circuito medidor de corrente

4.4.4 Peças

4.4.4.1 Sensor de Corrente Não-invasivo AC



Figura 8 – Non-invasive AC current sensor

Esse sensor de corrente consegue medir a corrente que passa por um fio de modo não-invasivo. O sensor funciona como um transformador respondendo a um campo magnético formado em volta do fio condutor. Este, em particular, suporta até 30A de entrada, e necessita de um resistor de saída para obter a medida desejada em tensão.

- Corrente suportada: 30A
- Temperatura de operação: -40°C até 65 °C
- Precisão de 2%

4.4.4.2 Raspberry Pi 2 modelo B

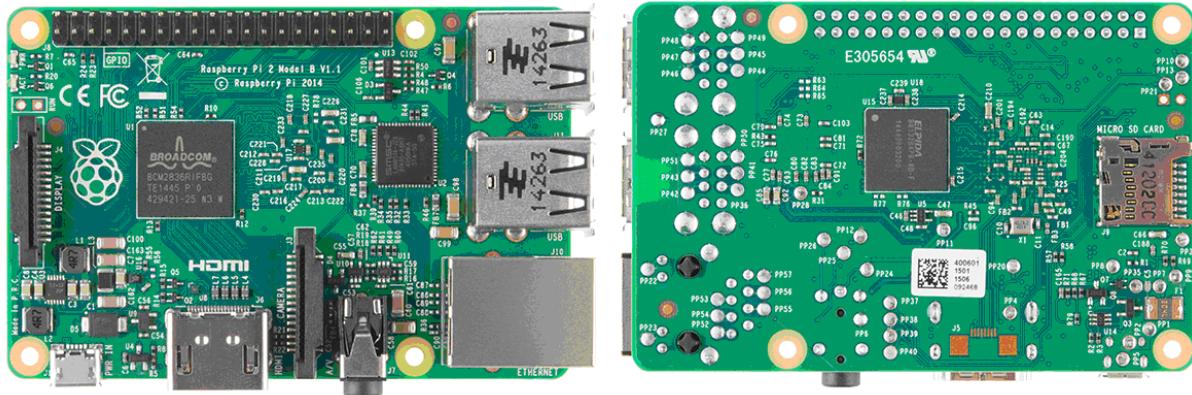


Figura 9 – Raspberry pi 2 modelo B

A Raspberry Pi 2 Modelo B (figura 9) é o computador utilizado no sistema para receber os dados enviados pelos módulos sensores, tratá-los e enviar para o aplicativo. Foi escolhido o Raspberry Pi 2 - Model B por ser mais veloz, por possuir mais entradas USB e ser de alta disponibilidade no mercado, por um preço razoável. O kit inclui a fonte, um cartão microSD de 8GB e um adaptador Wifi USB.

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 40 pinos GPIO
- saída Full HDMI
- porta Ethernet
- entrada para cartão Micro SD
- 4 entradas USB

4.4.4.3 Arduino UNO

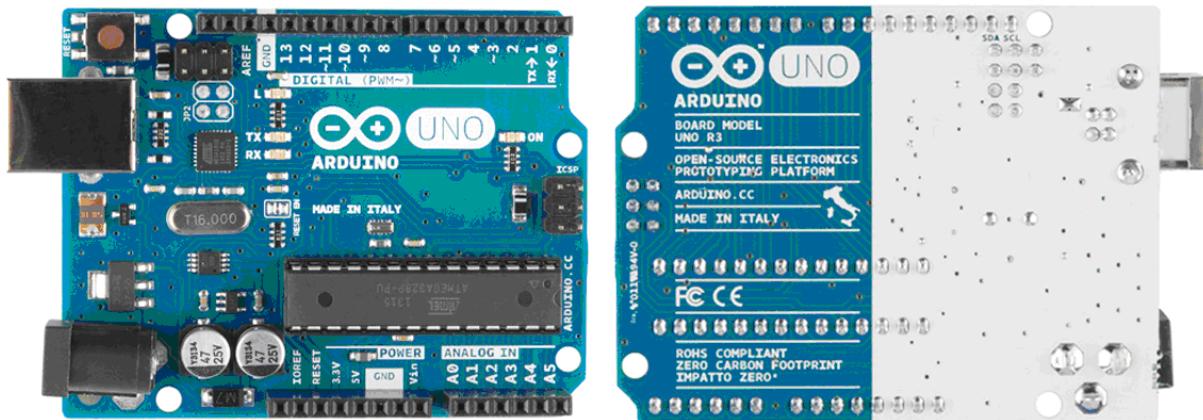


Figura 10 – Arduino UNO R3

Arduino é uma placa programável open-source . No projeto em questão esse componente receberá os dados do sensor, fará um tratamento e terá o envio programado desses para o coordenador. Pelo Arduino ser programável e possuir uma interface muito amigável, simplifica essa ponte entre a coleta de dados e a transmissão. E sua alta disponibilidade no mercado , assim como o raspberry, facilita sua obtenção.

- microcontrolador ATmega328
- tensão de entrada - 7-12V
- 14 Pinos Digital I/O (6 PWM de saída)
- 6 Inputs Analógicos
- 32k de memória Flash
- 16Mhz de Relógio

4.4.4.4 XBee

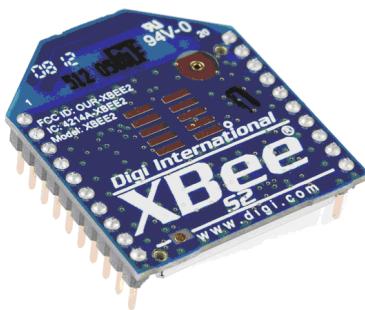


Figura 11 – XBee Serie 2

É um módulo que permite uma comunicação simples e confiável entre micro-controladores, computadores, sistemas através de uma porta serial com um consumo menor de energia. Pode ser utilizado em redes ponto-a-ponto e multi-ponto. Foram escolhidos módulos da série 2 por serem configuráveis. Algumas outras especificações são:

- entradas de 3.3V @ 40mA
- transmissão de dados máxima de 250kbps
- potência de saída: 2mW (+3dBm)
- alcance máximo de 120m
- 08 pinos digitais entrada/saída
- encriptação 128-bit
- configuração local ou remota
- conector de antena RPSMA

4.4.4.5 XBee Explorer Dongle

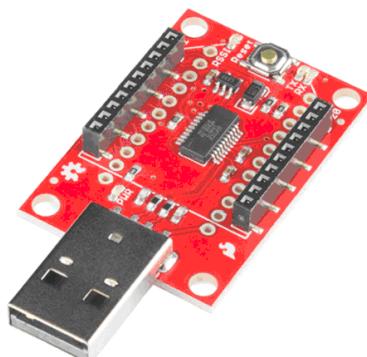


Figura 12 – XBee Explorer Dongle

É um módulo com porta USB que faz a conexão do módulo XBee a um computador. Isso é necessário para ter acesso aos pinos de comunicação serial e de programação. Ele possui um conversor serial, que traduz os dados entre o computador e o XBee. Possui um botão de reset e um regulador de tensão para suprir a tensão necessária para XBee. Além disso possui 4 leds para debug: RX, TX, RSSI e indicador de energia. No projeto, este módulo é utilizado para fazer as configurações iniciais de todos os XBees e para conectar o XBee coordenador ao Raspberry Pi. Apesar de não ser um dispositivo essencial, este facilita muito nas tarefas citadas, principalmente por lidar com a alimentação de 3,3V do XBee.

4.4.4.6 XBee Shield

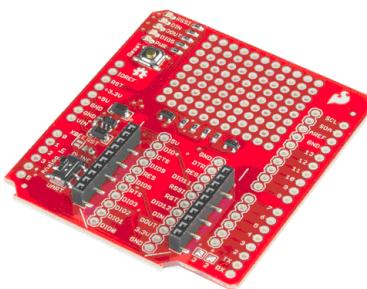


Figura 13 – XBee shield do arduino UNO

É um módulo que faz a conexão entre um módulo XBee e um Arduino. Ele possui opções para escolher se a conexão vai ser nos pinos UART ou qualquer outros pinos digitais do Arduino. A alimentação de 5V vinda do Arduino é regulada para 3.3V

VDC antes de chegar no módulo XBee. O XBee Shield inclui LEDs para indicar a utilização dos pinos DIN, DOUT, RSSI e DIO5 do XBee. É usado um módulo XBee Shield para cada par XBee + Arduino.

4.4.4.7 Arduino Stackable Header Kit - R3

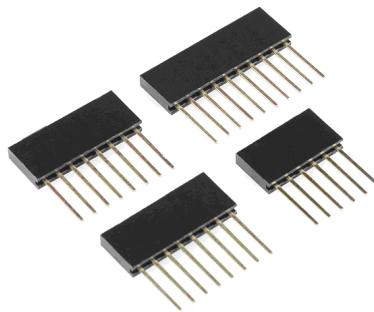


Figura 14 – Headers usados no XBee shield

São conectores usados para encaixar o módulo XBee Shield no Arduino Uno R3. Estão inclusos 4 headers, 2 x 8 pinos, 1 x 10 pinos e 1 x 6 pinos, suficientes para 1 módulo XBee Shield. Como há 2 sensores no projeto, serão usados 2 kits, com um adicional de reserva.

4.5 Software

Para que a aplicação Web possa receber os dados de consumo enviados pelo Módulo Coordenador, é necessário haver uma estrutura de rotas, métodos para lidar com objetos de classes, controladores e páginas de exibição para interagir com o usuário. Para isso, foi escolhida a arquitetura MVC, e um framework que consegue lidar com tal arquitetura. E para tornar a aplicação disponível para o acesso remoto através de smartphones, tablets e notebooks, optou-se por utilizar serviços de nuvem.

4.5.1 Tecnologia

A seguir são apresentadas as tecnologias utilizadas para a implementação da aplicação Web.

4.5.1.1 Django e Python

O framework Django foi utilizado devido ao uso da linguagem python, que é uma linguagem limpa, de fácil utilização e com ampla disponibilidade de bibliotecas gratuitas e de fóruns para auxílio na implementação. Uma das vantagens do python nesse caso é a compatibilidade, pois o hardware foi construído com intenção de aprendizado através da linguagem de programação python (14) (“pi” em “raspberry pi” vem de “python”). Outra vantagem é a experiência dos integrantes do grupo quanto ao manuseio do framework, o que agilizou o desenvolvimento. Outra vantagem é a possibilidade de utilizar, se necessário, o próprio raspberry como servidor da aplicação, bastando instalar as bibliotecas das dependências do Django.

O Django utiliza a arquitetura MVC. Os Models no Django contém os campos e os métodos dos objetos sendo utilizados. Todos as classes de Models herdam métodos e atributos da classe pré-existente Model do Django. Geralmente, cada Model é mapeado para uma tabela no banco de dados.

As Views (na arquitetura MVC) são chamadas Templates no Django. Elas são páginas no formato HTML que podem conter trechos de código em python para mostrar conteúdo do Context. Context é uma estrutura de chave-valor que provê o conteúdo a ser utilizado na página.

Os Controllers são chamados Views. O Django provê os chamados Base Views, que são classes que auxiliam a criação de Controllers. Podem ser utilizados diretamente, sobrescrevendo seus atributos e métodos, ou herdando estes para classes customizadas. Há três Base Views:

View: É a classe a partir da qual se herdam todas as outras classes de Controllers.

TemplateView: Renderiza um Template com um Context.

RedirectView: Redireciona para uma dada URL.

Outros Controllers pré-existentes são os Generic Display Views, que são Controllers genéricos utilizados normalmente para exibir dados de Models:

DetailView: Exibe detalhes um objeto.

ListView: Exibe uma lista de objetos.

Há Controllers utilizados para edição de conteúdo, que são os Generic Editing Views:

FormView: Exibe um formulário e realiza validação e redirecionamento caso não haja erros.

CreateView: Exibe um formulário para criação de objetos. Caso o formulário seja submetido e não haja erros na validação, cria o objeto.

UpdateView: Exibe um formulário para edição de objetos. Caso o formulário seja submetido e não haja erros na validação, salva o objeto.

DeleteView: Exibe uma página com uma caixa de confirmação para deletar um objeto.

4.5.1.2 Heroku

Heroku é uma plataforma em nuvem que fornece múltiplos serviços para dar suporte a uma aplicação web. É possível hospedar aplicações em linguagens como Node, Ruby, Java, PHP, Python, Go, Scala, ou Clojure, e o Heroku a manterá no ar sem a necessidade da intervenção do desenvolvedor. Esse serviço, diferente de opções de outros serviços de nuvem como o da Amazon, se encarrega em configurar o ambiente de execução da aplicação, o que agiliza o processo de colocar a aplicação em produção.

Heroku utiliza os chamados dynos, que representam máquinas/computadores que executam comandos. Cada tipo de dyno possui a sua limitação de memória RAM, fração de CPU, se é dedicada ou não e a velocidade do processamento, que refletem nos custos de aquisição dos serviços, porém, existe a opção gratuita que permite colocar uma aplicação em produção com um processamento suficiente para atender tráfegos pequenos. Nos planos pagos, o sistema é escalável (pode-se alterar o limite do número de processos em execução na máquina do sistema, memória RAM, entre outros) para atender a momentos de tráfego mais intenso.(15)

Durante a fase de teste do sistema em questão é utilizado o plano gratuito do Heroku.

4.5.2 Funções do Sistema

A aplicação Web possui tem como objetivo mostrar os dados de consumo para o usuário, mas a aplicação deve possuir outras funções para alcançar o objetivo. Uma delas é criar uma conta. Para que o usuário possa manter seus dados confidenciais, a aplicação permite autenticação através de senha e nome de usuário. Equipamentos são representados dentro do sistema, para que cada consumo possa se associar a um equipamento, e para isso, é necessário que o usuário gerencie os seus equipamentos. Os sensores também são representados dentro do sistema, pois um usuário tem a

opção de alocar o sensor de um equipamento para outro, caso deseje. O Módulo Coordenador cria os consumos dentro do sistema, enquanto o usuário visualiza, importa e exporta os consumos. Para que a conversão do consumo em unidades monetárias seja possível, o usuário deve atualizar as taxas da AES dentro do sistema. E para fazer a associação entre um sensor e um equipamento e atualizar suas informações de renda, o usuário deve configurar o sistema. A partir dessas informações foi criada a seguinte lista de funções para a aplicação:

Gerenciar contas: O usuário pode fazer cadastro/alteração de conta e autenticação.

Gerenciar equipamentos: O usuário pode fazer a criação, edição e remoção de equipamentos.

Gerenciar sensores: Os módulos sensores são auto-detectados, e o usuário pode editá-los ou removê-los.

Gerenciar metas: O usuário pode criar, editar e remover metas mensais.

Gerenciar consumo: O Módulo Coordenador envia consumos para o sistema. O usuário pode visualizar os consumos através de gráficos. Além disso o usuário pode importar ou exportar dados de consumo.

Atualizar taxas da AES: O usuário pode atualizar as taxas de energia utilizadas para cálculo do custo do consumo.

Configurar sistema: O usuário pode associar os sensores aos equipamentos e escolher um tipo de renda.

4.5.3 Atores

Dois atores foram identificados: o usuário e o módulo coordenador, que são as entidades externas que interagem com o sistema.

usuário: Como o sistema vai ser utilizado apenas pelo(s) responsável(is) pela residência, há apenas um tipo de usuário, que é o usuário comum.

módulo coordenador: É o componente de hardware que enviará informações de consumo para o sistema

4.5.4 Casos de uso

As funções obtidas foram divididas em casos de uso, como mostra a tabela 1.

Funções	Casos de uso
Gerenciar conta	Fazer cadastro Fazer login Fazer logout Recuperar senha
Gerenciar equipamentos	Criar equipamento Editar equipamento Remover equipamento
Gerenciar sensores	Detectar sensores Editar sensor Remover sensor
Gerenciar metas	Criar meta Editar meta Remover meta
Gerenciar consumo	Criar consumo Visualizar consumo Importar consumo Exportar consumo
Atualizar taxas da AES	Atualizar taxas da AES
Configurar sistema	Configurar sistema

Tabela 1 – Casos de Uso.

4.5.5 Descrição dos casos de uso

A seguir são descritos os casos de uso do sistema.

4.5.5.1 Função 1: Gerenciar conta

O usuário pode fazer cadastro/alteração de conta e autenticação.

4.5.5.2 Caso de Uso 1.1: Fazer cadastro

Descrição: inserção de um novo usuário comum no sistema

Evento iniciador: solicitação de cadastro

Atores: usuário

Pré-condição: sistema exibindo tela de solicitação de cadastro

Sequência de eventos:

1. Usuário solicita cadastro
2. Sistema exibe o formulário de cadastro
3. Usuário insere os seus dados
4. Sistema insere o novo usuário e exibe o resultado

Pós-condição: novo usuário cadastrado, usuário é logado automaticamente e é exibida a tela inicial

Extensões:

1. **Usuário a ser cadastrado já existe:** sistema apresenta uma mensagem ao usuário (passo 4)
2. **Dados do usuário não consistentes:** sistema apresenta mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar usuário (passo 4)

4.5.5.3 Caso de Uso 1.2: Fazer login

Descrição: criar uma sessão do usuário no sistema

Evento iniciador: solicitação de login

Atores: usuário

Pré-condição: usuário cadastrado e não há usuário logado

Sequência de eventos:

1. usuário solicita login
2. sistema exibe formulário para login
3. usuário insere os dados de login
4. sistema cria uma sessão para o usuário e redireciona para a página inicial

Pós-condição: sessão criada e sistema exibe a tela inicial

Extensões:

1. **Usuário não encontrado:** sistema apresenta uma mensagem de erro ao usuário (passo 4)
2. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar usuário (passo 4)

4.5.5.4 Caso de Uso 1.3: Fazer logout

Descrição: encerrar a sessão do usuário atual no sistema

Evento iniciador: solicitação de logout

Atores: usuário

Pré-condição: usuário logado

Sequência de eventos:

1. usuário solicita logout
2. sistema encerra a sessão atual, e redireciona para a página de login

Pós-condição: sessão encerrada e sistema exibe tela de login

4.5.5.5 Caso de Uso 1.4: Recuperar senha

Descrição: recuperar a senha do usuário

Evento iniciador: solicitação de recuperação de senha

Atores: usuário

Pré-condição: usuário cadastrado, não há usuário logado e sistema exibindo tela de login

Sequência de eventos:

1. usuário solicita recuperação de senha
2. sistema exibe formulário para recuperação de senha
3. usuário insere o e-mail
4. sistema envia e-mail para recuperar a senha e exibe mensagem
5. usuário clica no link para recuperar senha no e-mail

6. sistema exibe o formulário para recuperar a senha
7. usuário insere os dados pedidos
8. sistema atualiza a senha do usuário, autentica o usuário e redireciona para a tela inicial

Pós-condição: senha do usuário atualizada, usuário autenticado e sistema mostra a tela inicial

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4, 8)
2. **Senha antiga incorreta:** sistema apresenta uma mensagem de erro ao usuário (passo 8)

Inclusões:

1. Buscar usuário (passo 8)

4.5.5.6 Função 2: Gerenciar equipamentos

O usuário pode fazer a criação, edição e remoção de equipamentos.

4.5.5.7 Caso de Uso 2.1: Criar equipamento

Descrição: criar um novo equipamento

Evento iniciador: solicitação de criação de equipamento

Atores: usuário

Pré-condição: usuário logado e sistema exibindo listagem de equipamentos

Sequência de eventos:

1. usuário solicita criação de equipamento
2. sistema exibe formulário para criação
3. usuário insere os dados para criação
4. sistema cria um equipamento e redireciona para a listagem de equipamentos

Pós-condição: equipamento criado e sistema exibe listagem de equipamentos

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)
2. **Equipamento já existe:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar equipamento (passo 4)

4.5.5.8 Caso de Uso 2.2: Editar equipamento

Descrição: editar um equipamento

Evento iniciador: solicitação de edição de equipamento

Atores: usuário

Pré-condição: usuário logado, existem equipamentos e sistema exibindo listagem de equipamentos

Sequência de eventos:

1. usuário seleciona o equipamento desejado para edição
2. sistema exibe formulário para edição
3. usuário altera os dados desejados
4. sistema atualiza o equipamento e redireciona para a listagem de equipamentos

Pós-condição: equipamento atualizado e sistema exibe listagem de equipamentos

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar equipamento (passo 2, 4)

4.5.5.9 Caso de Uso 2.3: Remover equipamento

Descrição: remover um equipamento

Evento iniciador: solicitação de remoção de equipamento

Atores: usuário

Pré-condição: usuário logado, existem equipamentos e sistema exibindo listagem de equipamentos

Sequência de eventos:

1. usuário seleciona o equipamento desejado para remoção
2. sistema pede confirmação para remoção
3. usuário confirma
4. sistema remove o equipamento e redireciona para a listagem de equipamentos

Pós-condição: equipamento removido e sistema exibe listagem de equipamentos

Extensões:

1. **Usuário não confirma:** sistema não remove e volta para a tela de listagem (passo 4)

Inclusões:

1. Buscar equipamento (passo 2, 4)

4.5.5.10 Função 3: Gerenciar sensores

Os módulos sensores são auto-detectados, e o usuário pode editá-los ou removê-los.

4.5.5.11 Caso de Uso 3.1: Detectar sensor

Descrição: detectar um sensor

Evento iniciador: solicitação de detecção de sensores

Atores: usuário

Pré-condição: usuário logado e sistema exibindo listagem de sensores

Sequência de eventos:

1. usuário solicita detecção de sensor
2. sistema detecta e cria um sensor no sistema com status ativo e atualiza a lista de sensores

Pós-condição: sensor criado e sistema exibe listagem de sensores

Extensões:

1. **Sensor já existe no sistema:** sistema atualiza o status do sensor para ativo (passo 2)

Inclusões:

1. Buscar sensor (passo 2)

4.5.5.12 Caso de Uso 3.2: Editar sensor

Descrição: editar um sensor

Evento iniciador: solicitação de edição de sensor

Atores: usuário

Pré-condição: usuário logado, existem sensores e sistema exibindo listagem de sensores

Sequência de eventos:

1. usuário seleciona o sensor desejado para edição
2. sistema exibe formulário para edição
3. usuário altera os dados desejados
4. sistema atualiza o sensor e redireciona para a listagem de sensores

Pós-condição: sensor atualizado e sistema exibe listagem de sensores

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar sensor (passo 2, 4)

4.5.5.13 Caso de Uso 3.3: Remover sensor

Descrição: remover um sensor

Evento iniciador: solicitação de remoção de sensor

Atores: usuário

Pré-condição: usuário logado, existem sensores e sistema exibindo listagem de sensores

Sequência de eventos:

1. usuário seleciona o sensor desejado para remoção
2. sistema pede confirmação para remoção
3. usuário confirma
4. sistema remove o sensor e redireciona para a listagem de sensores

Pós-condição: sensor removido e sistema exibe listagem de sensores

Extensões:

1. **Usuário não confirma:** sistema não remove e volta para a tela de listagem (passo 4)

Inclusões:

1. Buscar sensor (passo 2, 4)

4.5.5.14 Função 4: Gerenciar metas

O usuário pode criar, editar e remover metas mensais.

4.5.5.15 Caso de Uso 4.1: Criar meta

Descrição: criar uma nova meta

Evento iniciador: solicitação de criação de meta

Atores: usuário

Pré-condição: usuário logado e sistema exibindo listagem de metas

Sequência de eventos:

1. usuário solicita criação de meta
2. sistema exibe formulário para criação
3. usuário insere os dados para criação
4. sistema cria uma meta e redireciona para a listagem de metas

Pós-condição: meta criada e sistema exibe listagem de metas

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)
2. **Meta já existe:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar meta (passo 4)

4.5.5.16 Caso de Uso 4.2: Editar meta

Descrição: editar uma meta

Evento iniciador: solicitação de edição de meta

Atores: usuário

Pré-condição: usuário logado, existem metas e sistema exibindo listagem de metas

Sequência de eventos:

1. usuário seleciona a meta desejado para edição
2. sistema exibe formulário para edição
3. usuário altera os dados desejados
4. sistema atualiza a meta e redireciona para a listagem de metas

Pós-condição: meta atualizada e sistema exibe listagem de metas

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 4)

Inclusões:

1. Buscar meta (passo 2, 4)

4.5.5.17 Caso de Uso 4.3: Remover meta

Descrição: remover uma meta

Evento iniciador: solicitação de remoção de meta

Atores: usuário

Pré-condição: usuário logado, existem metas e sistema exibindo listagem de metas

Sequência de eventos:

1. usuário seleciona a meta desejado para remoção
2. sistema pede confirmação para remoção
3. usuário confirma
4. sistema remove a meta e redireciona para a listagem de metas

Pós-condição: meta removida e sistema exibe listagem de metas

Extensões:

1. **Usuário não confirma:** sistema não remove e volta para a tela de listagem (passo 4)

Inclusões:

1. Buscar meta (passo 2, 4)

4.5.5.18 Função 5: Gerenciar consumos

O Módulo Coordenador envia consumos para o sistema. O usuário pode visualizar os consumos através de gráficos. Além disso o usuário pode importar ou exportar dados de consumo.

4.5.5.19 Caso de Uso 5.1: Criar consumo

Descrição: inserir consumos no sistema

Evento iniciador: solicitação para criação de consumo

Atores: módulo coordenador

Pré-condição: módulo coordenador ligado e sistema online

Sequência de eventos:

1. módulo coordenador solicita criação de consumo
2. sistema cria o consumo

Pós-condição: consumo criado

Extensões:

1. **Perda de conexão:** consumo não é criado (passo 2)

4.5.5.20 Caso de Uso 5.2: Visualizar consumo

Descrição: visualizar os consumos na forma de gráficos

Evento iniciador: solicitação de geração de gráfico

Atores: usuário

Pré-condição: usuário logado, existem consumos e sistema exibindo tela de consumo

Sequência de eventos:

1. usuário configura os parâmetros e solicita geração do gráfico
2. sistema exibe o gráfico de consumo

Pós-condição: sistema exibe gráfico de consumo

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 2)

Inclusões:

1. Buscar consumos (passo 2)

4.5.5.21 Caso de Uso 5.3: Importar consumos

Descrição: importar consumos por csv

Evento iniciador: solicitação de importação de consumos

Atores: usuário

Pré-condição: usuário logado, sistema exibindo tela de consumo

Sequência de eventos:

1. usuário insere o arquivo csv e solicita importação de consumos
2. sistema lê o csv, cria os consumos e exibe tela de consumo

Pós-condição: novos consumos criados e sistema exibe tela de consumo

Extensões:

1. **Dados não consistentes:** sistema apresenta uma mensagem de erro ao usuário (passo 2)

4.5.5.22 Caso de Uso 5.4: Exportar consumos

Descrição: exportar consumos por csv

Evento iniciador: solicitação de exportação de consumos

Atores: usuário

Pré-condição: usuário logado, existem consumos e sistema exibindo tela de consumo

Sequência de eventos:

1. usuário seleciona o período desejado do consumo para exportação
2. sistema disponibiliza o download do csv

Pós-condição: sistema exibe arquivo de csv

Inclusões:

1. Buscar consumo (passo 2)

4.5.5.23 Caso de Uso 6: Atualizar taxas da AES

Descrição: atualizar dados de custo da AES Eletropaulo no sistema

Evento iniciador: solicitação de atualização das taxas

Atores: usuário

Pré-condição: usuário logado e sistema exibindo tela de listagem de taxas

Sequência de eventos:

1. usuário solicita atualização de taxas
2. sistema busca dados do site da AES Eletropaulo e cria taxas no sistema

Pós-condição: taxas criadas e sistema exibe listagem de taxas

Extensões:

1. **Taxa já existe no sistema:** sistema atualiza a taxa correspondente (passo 2)

Inclusões:

1. Buscar taxa (passo 2)

4.5.5.24 Caso de Uso 7: Configurar sistema

Descrição: mudar configuração do sistema

Evento iniciador: solicitação de mudança de configuração do sistema

Atores: usuário

Pré-condição: usuário logado e sistema exibindo tela de configuração

Sequência de eventos:

1. usuário muda a configuração e solicita salvar a configuração
2. sistema salva as configurações e retorna para tela de configuração

Pós-condição: configurações salvas e sistema mostra tela de configuração

Extensões:

1. **Dados inconsistentes:** sistema mostra mensagem de erro ao usuário (passo 2)

Inclusões:

1. Buscar configuração do usuário (passo 2)

4.5.6 Classes e atributos

A partir dos casos de uso foi possível a identificação das classes dentro do sistema: Equipment, Consumption, Sensor, AESRate, Goal, User. Essas classes e suas relações estão representadas pelo diagrama de classes na figura 15.

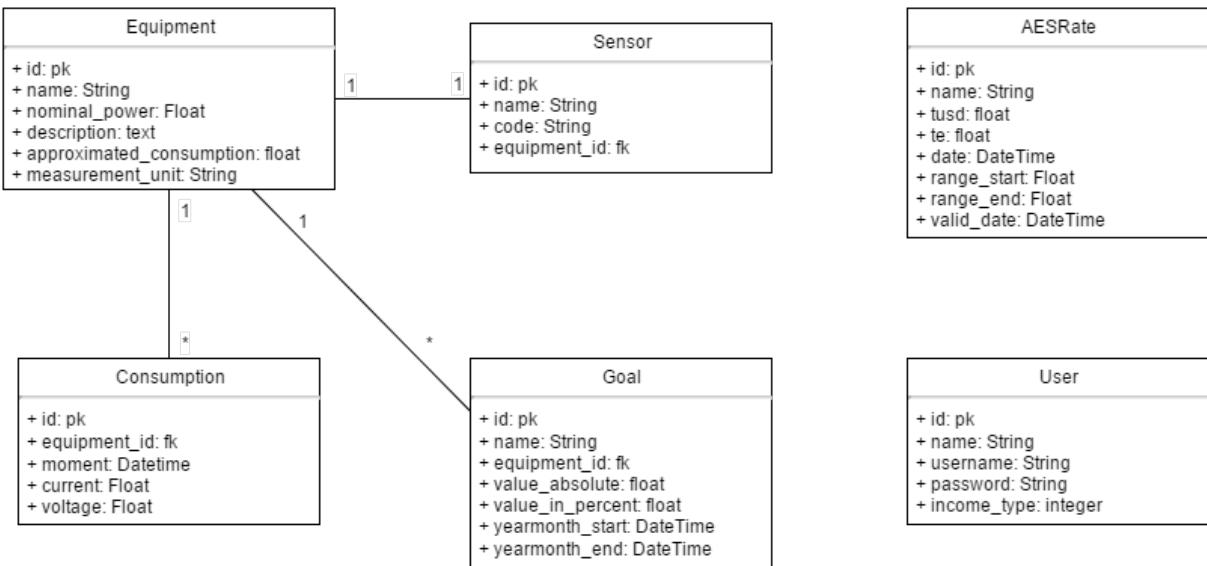


Figura 15 – Diagrama de Classes

A seguir estão descritas as classes do sistema e seus atributos.

4.5.6.1 Equipment

Classe: Equipment

Descrição: Representa um equipamento na aplicação. Os equipamentos são criados pelos usuários dentro do sistema. É necessário possuir um sensor associado para que o equipamento possa ser associado a um consumo.

Atributos:

1. id (integer): identificador do equipamento
2. name (String): nome do equipamento
3. description (Text): descrição do equipamento
4. nominal_power (float): potência nominal do equipamento
5. measurement_unit (String): unidade de medida utilizada em nominal_power
6. approximated_consumption (float): consumo aproximado do equipamento dado pelo fabricante

Relacionamentos:

1. um equipamento possui nenhum ou um sensor
2. um equipamento possui nenhum ou vários consumos
3. um equipamento possui nenhum ou várias metas

4.5.6.2 Sensor

Classe: Sensor

Descrição: Representa um sensor na aplicação. Os sensores são criados automaticamente pelo sistema ao receber um consumo de um sensor não registrado. O usuário poderá, então, editar o nome do sensor. Porém, como não há dado que indique em qual aparelho o sensor foi instalado, tal associação deve ser feita através de configuração (Caso de uso Configurar sistema). Caso um sensor seja alocado de um equipamento para outro, os novos consumos passarão a pertencer ao segundo equipamento.

Atributos:

1. id (integer): identificador do sensor
2. name (String): nome dado pelo usuário para o sensor
3. code (String): identificador do sensor, enviada pelo módulo sensor (endereço MAC do XBee no módulo sensor)
4. equipment_id (integer): equipamento ao qual está associado

Relacionamentos:

1. um sensor pertence a um ou nenhum equipamento

4.5.6.3 Consumption

Classe: Consumption

Descrição: Representa uma medida de consumo feita de um equipamento em um dado instante. Quando um consumo é enviado ao sistema, o valor da corrente, tensão e identificador do sensor são enviados. Caso o identificador do sensor não exista dentro do sistema, uma nova instância de sensor será criada. A partir do momento em que o sensor tiver um equipamento associado, consumos para aquele equipamento poderão ser criados. Caso um sensor seja alocado de um equipamento para outro, os consumos para o equipamento anterior vão continuar pertencendo ao mesmo, enquanto os novos consumos pertencerão ao segundo equipamento.

Atributos:

1. id (integer): identificador do consumo
2. equipment_id (integer): identificador do equipamento

3. moment (DateTime): a data e a hora de quando foi feita a medida
4. current (float): corrente no momento da medida em amperes
5. voltage (float): tensão da tomada do equipamento. 220V ou 127V

Relacionamentos:

1. um consumo pertence a um equipamento

4.5.6.4 User

Classe: User

Descrição: Representa um usuário do sistema.

Atributos:

1. id (integer): identificador do usuário
2. name (String): nome do usuário
3. username (String): nome de usuário usado para efetuar o login
4. password (String encriptado): senha do usuário usada para efetuar o login
5. income_type (String): o tipo de renda do usuário, Residencial ou Residencial de baixa renda, de acordo com a especificação da AES eletropaulo.

4.5.6.5 Goal

Classe: Goal

Descrição: Representa uma meta de consumo para um mês. Ao cadastrar a meta, ela calcula um valor igual a percentagem (value_in_percent) do total de consumo para um equipamento no mês anterior. Ao ser traçado o gráfico do mês pertencente ao da meta para aquele equipamento, um gráfico com o valor da meta será exibido.

Atributos:

1. id (integer): identificador da meta
2. equipment_id (integer): identificador do equipamento
3. name (String): nome da meta
4. value_in_percent (float): consumo pretendido em percentagem (em relação ao mês anterior)
5. value_absolute (float): consumo pretendido (em relação ao mês anterior)

6. yearmonth_start (DateTime): início do período da meta
7. yearmonth_end (DateTime): fim do período da meta

Relacionamentos:

1. uma meta pertence a um equipamento

4.5.6.6 AESRate

Classe: AESRate

Descrição: Representa a taxa de conversão da AES eletropaulo de kilowatts hora para reais. Esses valores são obtidos através da página de tarifas do site da AES Eletropaulo ([16](#)). Caso o usuário queira visualizar o consumo em reais, o usuário deve escolher a opção de integrar o gráfico também (pois as tarifas são calculadas em função da energia consumida, e não em função potência consumida em dado instante). Em seguida, o sistema identifica qual taxa de conversão, dependendo da data do consumo (deve ser maior ou igual a valid_date), tipo de renda (se é igual ao atributo name), faixa de consumo (o valor de consumo deve ser maior ou igual a range_start e menor ou igual a range_end). Identificada a taxa, o sistema multiplica o valor de cada ponto do gráfico com a soma de TE e TUSD da taxa para converter em reais.

Atributos:

1. id (integer): identificador da taxa de conversão
2. name (String): nome da taxa de conversão, o mesmo utilizado pela AES.
3. te (float): tarifa de energia
4. tusd (float): tarifa de uso do sistema de distribuição
5. date (DateTime): o instante que a taxa de conversão foi buscada
6. valid_date (DateTime): data de início da validade das tarifas
7. range_start (float): o início da faixa de consumo que define a taxa de conversão
8. range_end (float): o fim da faixa de consumo que define a taxa de conversão

5 Implementação

A fase de implementação envolve o desenvolvimento do software e a configuração e montagem dos componentes físicos especificados na seção [4](#).

5.1 Aplicação Web

Vários casos de uso especificados anteriormente possuem uma certa interdependência entre eles, por exemplo, o caso de uso Criar Meta necessita que o caso de uso Criar Equipamento esteja previamente desenvolvido. Para isso, adotou-se a seguinte ordem de implementação:

1. Fazer cadastro
2. Fazer login
3. Fazer logout
4. Recuperar senha
5. Criar equipamento
6. Editar equipamento
7. Remover equipamento
8. Criar meta
9. Editar meta
10. Remover meta
11. Atualizar taxas da AES
12. Detectar sensores
13. Editar sensor
14. Remover sensor
15. Configurar sistema
16. Criar consumo

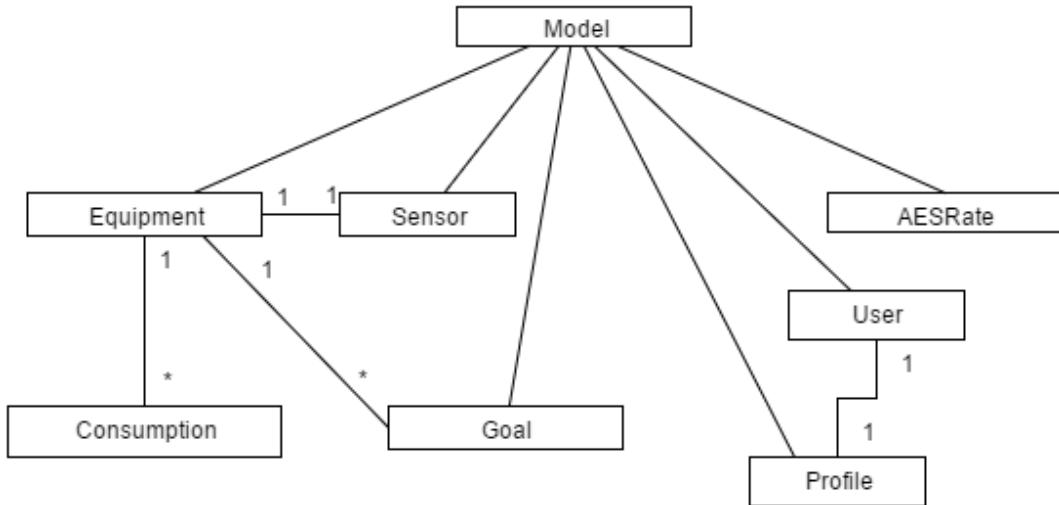


Figura 16 – Diagrama de Classes - Models

17. Importar consumo
18. Exportar consumo
19. Visualizar consumo

5.1.1 Models e Controllers

Antes de implementar os casos de uso, as classes iniciais foram adaptadas para os Models do Django e a organização das classes no framework está representada na figura 16. Essas classes são responsáveis por acessar, criar, alterar ou remover dados dos equipamentos, sensores, taxas da AES, metas e usuários do banco de dados. Nos diagramas de classes simplificados a seguir foram representadas apenas as classes criadas para o projeto e as classes às quais estão diretamente relacionadas, ou seja, as demais classes do Django estão omitidas. A diferença observada em relação ao primeiro diagrama de classes (figura 15) foi o fato de haver uma classe pré-existente User. A opção mais adequada para esse caso é criar um Model diferente (chamada Profile nesse caso) que tenha uma Foreign Key para um User. A partir de então, um usuário poderá possuir um atributo adicional - income_type - bastando criar um objeto da classe Profile. Essa é a opção mais adequada, uma vez que não é necessário criar uma nova classe de usuário, reescrevendo todos os atributos e métodos que o Django já oferece.

No projeto em questão são utilizados Controllers Principais (figura 17):

IndexView: Exibe página principal sem usuário logado.

HomeView: Exibe página principal do usuário logado.



Figura 17 – Diagrama de Classes - Controllers Principais

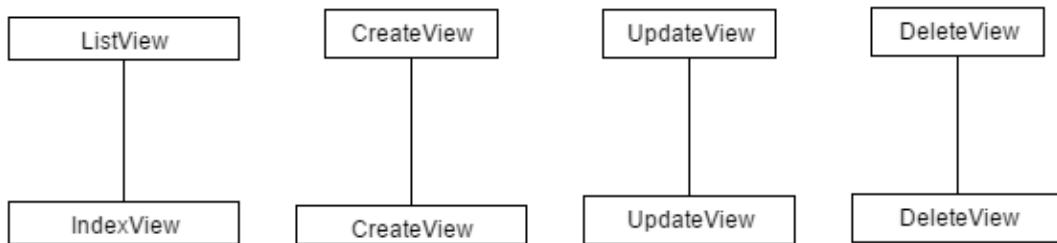


Figura 18 – Diagrama de Classes - Controllers de Equipamentos

LoginView: Exibe formulário para login e realiza autenticação.

LogoutView: Encerra a sessão do usuário.

Os Controllers que lidam com os equipamentos estão representados na figura 18. E os Controllers são:

IndexView: Exibe página de listagem dos equipamentos.

CreateView: Exibe página com formulário para criação de um equipamento, cria um equipamento, realiza redirecionamento após a criação.

UpdateView: Exibe página com formulário para edição de um equipamento, edita um equipamento, realiza redirecionamento após a edição.

DeleteView: Remove um equipamento.

Os Controllers que lidam com os sensores estão representados na figura 19. E os Controllers são:

IndexView: Exibe página de listagem dos sensores.

UpdateView: Exibe página com formulário para edição de um sensor, edita um sensor, realiza redirecionamento após a edição.

DeleteView: Remove um sensor.

Os Controllers que lidam com as taxas da AES estão representados na figura 20. E os Controllers são:

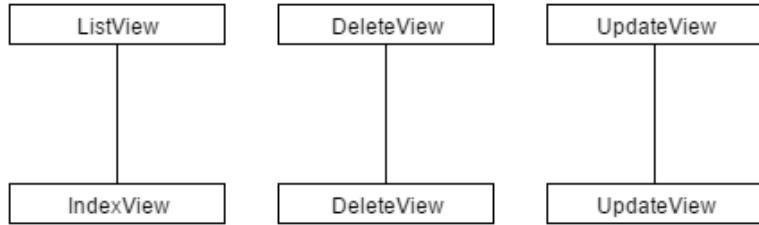


Figura 19 – Diagrama de Classes - Controllers de Sensores

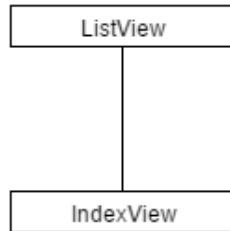


Figura 20 – Diagrama de Classes - Controllers de Taxas da AES

IndexView: Exibe página de listagem das taxas AES.

Os Controllers que lidam com as configurações do sistema (e do usuário) estão representados na figura 21. E os Controllers são:

ConfigView: Exibe página com formulário para configuração do sistema, configura o sistema.

UsercreateView: Exibe página com formulário para registro de um usuário, registra um usuário e realiza redirecionamento após registro.

UserUpdateView: Exibe página com formulário para alteração dos dados de um usuário, altera os dados do usuário e realiza redirecionamento após alteração dos dados.

PasswordUpdateView: Exibe página com formulário para edição de senha do usuário, edita a senha do usuário e redireciona após a alteração da senha.

PasswordRecoverView: Exibe página com formulário para recuperação de senha do usuário, manda e-mail de recuperação de senha.

PasswordRecoverDoneView: Redireciona depois do envio de e-mail para recuperação de senha.

PasswordResetView: Exibe página com formulário para reconfigurar a senha do usuário, reconfigura a senha do usuário.

PasswordResetDoneView: Redireciona depois da reconfiguração de senha do usuário.

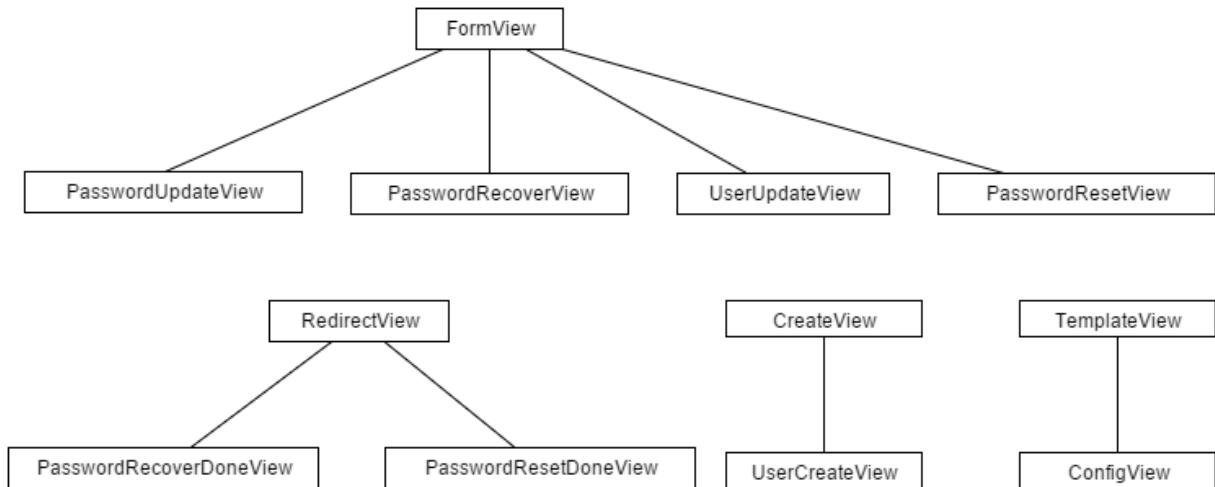


Figura 21 – Diagrama de Classes - Controllers de Configuração

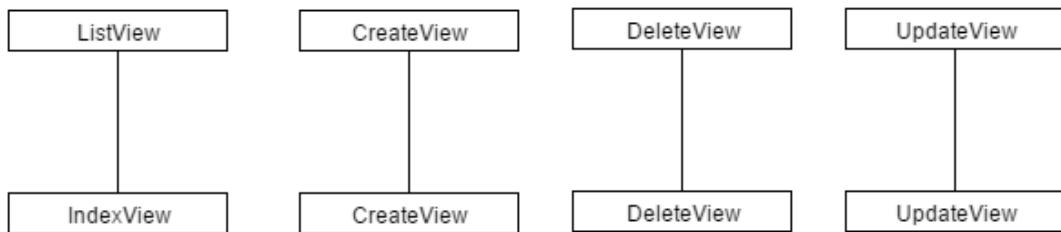


Figura 22 – Diagrama de Classes - Controllers de Metas

Os Controllers que lidam com as metas estão representados na figura 22. E os Controllers são:

IndexView: Exibir página de listagem das metas.

CreateView: Exibir página com formulário para criação de uma meta, cria uma meta, realiza redirecionamento após criar a meta.

UpdateView: Exibir página com formulário para edição de uma meta, edita a meta, realiza redirecionamento após editar a meta.

DeleteView: Remove uma meta.

Os Controllers que lidam com os consumos estão representados na figura 23. E os Controllers são:

GraphicView: Exibe o gráfico de consumo

No mesmo arquivo com os controllers dos consumos estão localizados os métodos de criação do consumo (a ser chamado quando o Módulo Coordenador enviar uma requisição para a aplicação), importar consumos por arquivo CSV, exportar consumos por arquivo CSV e processar os dados para renderizar através do GraphicView.

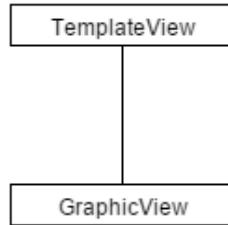


Figura 23 – Diagrama de Classes - Controllers de Consumos

5.2 Módulo Sensor e Módulo Coordenador

5.2.1 XBee: configuração

O tutorial usado para configurar a rede dos dispositivos XBee encontra-se em ([17](#)).

As configurações utilizadas foram:

5.2.1.1 Módulo Sensor:

- Versão de Firmware:
 - Product Family: XB24-ZB
 - Function Set: ZigBee Router AT
 - Firmware Version: 22A7
- Pad ID: BABABA
- Destination Address High: 13A200
- Destination Address Low: 40E4429B
- Data bits: 8
- Baud Rate: 9600
- Parity: No parity
- Stop Bits: One Stop Bit

5.2.1.2 Módulo Coordenador:

- Versão de Firmware:
 - Product Family: XB24-ZB

- Function Set: ZigBee Coordinator API
- Firmware Version: 21A7
- Pad ID: BABABA
- Destination Address High: 13A200
- Destination Address Low: 40E44285
- Data bits: 8
- Baud Rate: 9600
- Parity: No parity
- Stop Bits: One Stop Bit

5.2.2 Raspberry: Sistema Operacional

Raspberry Pi é um computador, como tal é possível usá-lo com um sistema operacional. O sistema operacional usado é o Raspian, baseado no Debian customizado para rodar no raspberry pi. Apesar de teoricamente não ser necessário o raspberry usar um sistema operacional, facilita muito o desenvolvimento tal equipamento conter uma interface amigável para seu uso.

5.2.3 Raspberry: Adaptador Wifi

Após o sistema Raspian ser instalado no cartão de memória, foi necessário configurar o adaptador Wifi (figura 24). O modelo usado é o TP-Link TL-WN723N.



Figura 24 – Adaptador wifi usado no raspberry

5.2.4 Raspberry x Arduino: comunicação

Para testar a comunicação entre o Módulo Coordenador (figura 25) e o módulo sensor (figura 26), foi montado o esquema representado na figura 27 e foram executados dois programas. O programa 5.1 foi utilizado no módulo sensor para transmitir o valor "123123" por um datagrama a ser enviado pela rede do XBee. Já o programa 5.2 foi utilizado no Módulo Coordenador para receber o datagrama ZigBee e imprimir no console do Raspberry.



Figura 25 – Teste de comunicação: Módulo Coordenador



Figura 26 – Teste de comunicação: Módulo Sensor

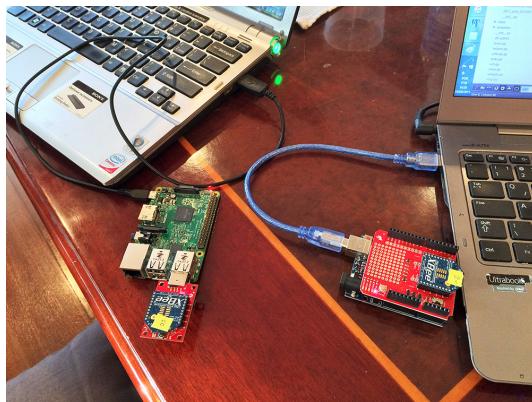


Figura 27 – Teste de comunicação: Montagem

```
// SoftwareSerial para comunicacao com XBee:  
#include <SoftwareSerial.h>  
SoftwareSerial XBee(2, 3); // RX, TX  
  
void setup(){  
    XBee.begin(9600);  
    Serial.begin(9600);  
}  
  
void loop(){  
    XBee.print("123123");  
    delay(1000);  
}
```

Listing 5.1 – teste-inicial-arduino.c

```
from xbee import ZigBee  
import serial  
  
ser = serial.Serial('/dev/ttyUSB0', 9600)  
xbee = ZigBee(ser)  
  
while True:  
    try:  
        response = xbee.wait_read_frame()  
        print(response)  
    except:  
        break;  
  
ser.close()
```

Listing 5.2 – teste-inicial-raspberry.py

```
pi@raspberrypi ~]$ python python_xbee_test.py
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
{'source_addr_long': '\x00\x13\xA2\x00@\x4B\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
```

Figura 28 – Teste de comunicação Raspberry e Arduino

Após esse teste, foi feito um pequeno teste para observar o que ocorre com múltiplos módulos sensores no sistema. Logo, o teste acima foi realizado com dois módulos com frequências diferentes de envio para garantir que não ocorre colisão.

5.2.5 Verificador de tensão

Para verificar a tensão foi usado um transformador para rebaixar a tensão da tomada. Na imagem é possível observar uma tomada de 127V sendo rebaixada para 7,2 V.

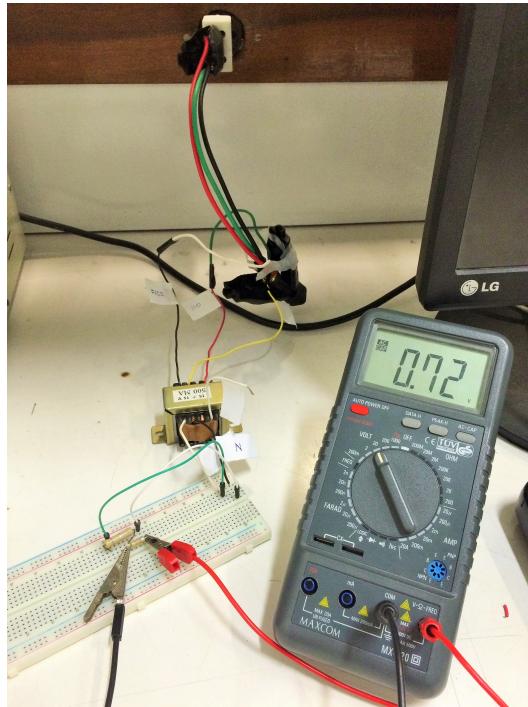


Figura 29 – Teste do transformador

Em seguida a saída foi retificada com um retificador de onda completa e filtrada com um capacitor e dividida por divisão de tensão com resistores para que fosse possível diferenciar uma tensão de 220V e 127V no arduino para enviar ao raspberry.

A figura 30 mostra a montagem para tal e a figura 31 mostra a saída no osciloscópio.

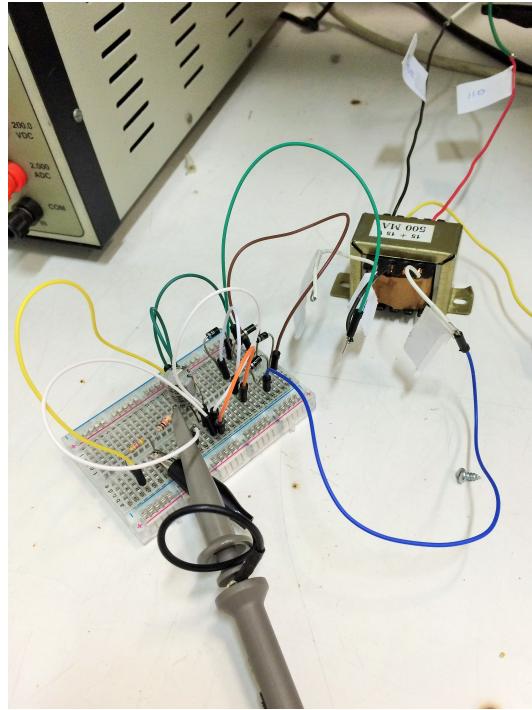


Figura 30 – Teste de medição de tensão

Em seguida a saída retificada e dividida foi posta como entrada para um pino de leitura analógica do Arduino para a medição, para 127V e em seguida, 220V. A saída resultante no terminal serial do arduino **31** foi a seguinte

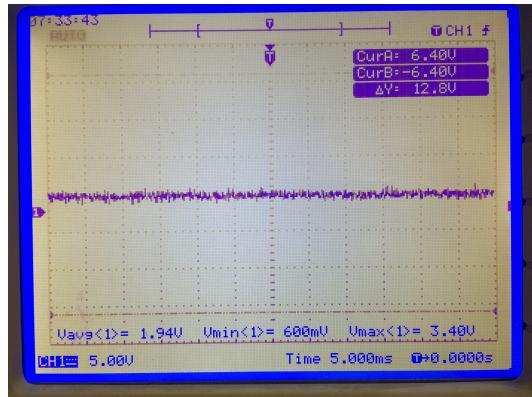


Figura 31 – Teste de medição de tensão com arduino - saída do osciloscópio

5.2.6 Sensor de corrente

Para o sensor de corrente, dado que a saída do sensor é alternada, foi necessário fazer um pequeno circuito para que as informações coletadas estivessem em uma faixa legível ao arduíno. Para isso, foi colocado uma resistência entre as saídas do sensor, resistência essa que deveria, em 30A (corrente máxima do sensor), resultar em uma tensão máxima de 2,5V para então adicionar um offset de 2,5V, possibilitando

amostragens em todos os pontos da onda. Como apenas o valor de pico é relevante para calcular a potência aparente. No caso, a resistência utilizada foi de 150 ohms.

Já no arduino, para fazer a leitura, não era possível apenas ler diretamente o valor da entrada analógica, pois estávamos interessados no valor eficaz da corrente. Por isso foi utilizado da raiz da média dos quadrados (tradução livre de RMS - root mean square), fazendo várias leituras sucessivas, assim como o nome do método diz, faz-se a raiz da média dos quadrados das leituras para obter o valor eficaz.

Com o circuito de corrente pronto, foi possível fazer algumas leituras teste. No caso foi usado o ferro de passar para que fosse usada uma corrente mais alta.

Como se pode ver nos testes, apesar da corrente lida no amperímetro ser 9,3A eficazes (figura 32), as leituras no arduíno resultaram em valores entre 8,9 e 8,8A eficazes (figura 33), resultando em um erro de aproximadamente 7%, que é um erro relativamente alto se pensado em uma leitura de longo prazo.

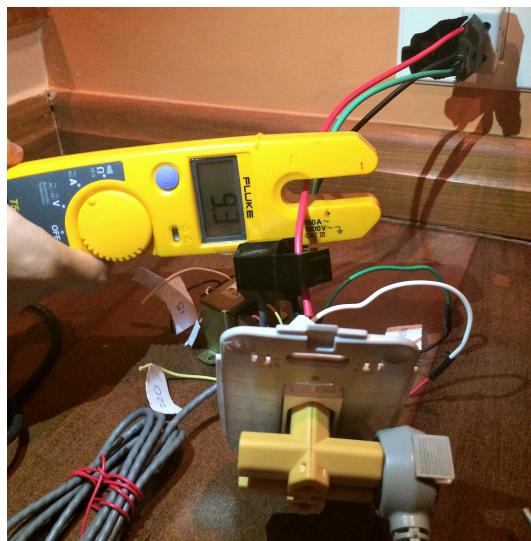


Figura 32 – Medição com amperímetro

```
// SoftwareSerial para comunicacao com XBee:
#include <SoftwareSerial.h>
SoftwareSerial XBee(2, 3); // RX, TX
#include "EmonLib.h"

EnergyMonitor emon1;
double current = 0;
double voltage = 0;
int currentPin = 5;
int voltagePin = 3;
String bufferString = "";
double burdenResistor = 150.0;
double CYCLE_TIME = 1 / 60;
```

```
int buffer[170];

double analogToDouble(int reading){
    return (double)reading / 1024 * 5;
}

double readVoltage(){
    double reading = analogToDouble(analogRead(voltagePin));
    if(reading < 2.5){
        return 110;
    }else{
        return 220;
    }
}

double readCurrent(int samples){
    double sum = 0;
    int sample;
    double convertedSample;
    double max = 0;
    for(int i = 0; i < samples; i++){
        offsetI = (offsetI + (sampleI-offsetI)/1024);
        filteredI = sampleI - offsetI;

        sample = analogRead(currentPin);
        convertedSample = (analogToDouble(sample) - 2.45) / burdenResistor;
        sum += convertedSample * convertedSample;
    }
    return sqrt(sum / samples) * 2000.0;
}

void setup(){
    Serial.begin(9600);
    XBee.begin(9600);
    emon1.current(currentPin, 100.0);
}

void loop(){
    bufferString += "{ 'v' : ";
    bufferString += readVoltage();
    bufferString += ", 'i' : ";
    bufferString += readCurrent(1667);
    bufferString += " }";
    Serial.println(bufferString);
    XBee.print(bufferString);
    bufferString = "";
}
```

```

    delay(1000);
}

```

Listing 5.3 – Leitura da corrente

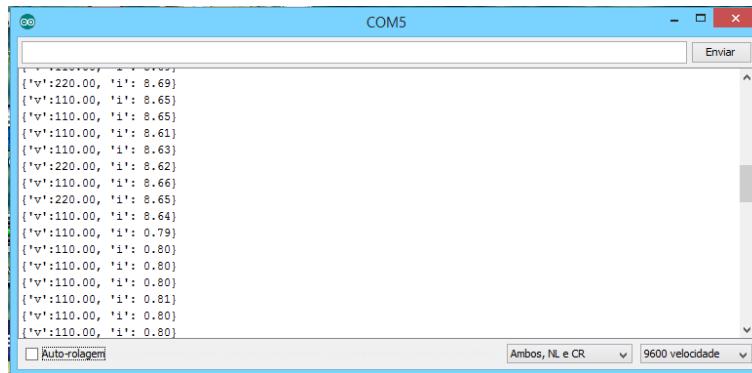


Figura 33 – Teste de medição da corrente e tensão

5.2.7 Integrando as partes

Após os testes anteriores de comunicação entre o módulo sensor, módulo coordenador e a medição de corrente e tensão, o conjunto foi testado. O código 5.4 foi carregado no arduino para a coleta dos dados de corrente e tensão e o envio destes para o módulo coordenador.

```

// SoftwareSerial para comunicacao com XBee:
#include <SoftwareSerial.h>
SoftwareSerial XBee(2, 3); // RX, TX

float current = 0;
float voltage = 0;
int currentPin = 5;
int voltagePin = 3;
String bufferString = "";

float analogToFloat(int reading){
    return (float)reading / 1024 * 5;
}

float readVoltage(){
    float reading = analogToFloat(analogRead(voltagePin));
    if(reading < 2.5){
        return 110 * sqrt(2);
    }else{
        return 220 * sqrt(2);
    }
}

```

```

float readCurrent(){
    return analogToFloat(analogRead(currentPin));
}

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

void loop(){
    bufferString += "'v':";
    bufferString += analogToFloat(analogRead(voltagePin));
    bufferString += ",'i':";
    bufferString += readCurrent();
    bufferString += "}";
    Serial.println(bufferString);
    XBee.print(bufferString);
    bufferString = "";
    delay(100);
}

```

Listing 5.4 – arduino.c

O módulo coordenador foi configurado para executar o seguinte programa em python (código 5.5), logo após ligar o raspberry, para a inicialização da interface entre os sensores e a aplicação web. Uma vez com o programa em execução, o módulo coordenador estará pronto para escutar as requisições de envio de dados coletados dos módulos sensores.

```

from xbee import XBee, ZigBee
import serial
import httplib, urllib, urllib2

WEB_ENDPOINT = "http://smrue-mi.herokuapp.com/consumption/create"
ser = serial.Serial('/dev/ttyUSB0', 9600)
xbee = ZigBee(ser)
output = "output.txt"

def sendDataToWeb(hashFormattedData):
    params = urllib.urlencode({
        "code": hashFormattedData['source_addr_long'].encode("hex"),
        "current": 1,
        "voltage": 1
    })
    req = urllib2.Request(WEB_ENDPOINT, params)
    response = urllib2.urlopen(req)

```

```
def readConsumption():
    output_file = open(output, "w")
    while True:
        try:
            response = xbee.wait_read_frame()
            output_file.write("Options\n")
            output_file.write(str(response['options'].encode("hex"))$)
            output_file.write("Src_addrs"\n")
            output_file.write(str(response['source_addr'].encode("h$"))
            output_file.write("Src_addrs_long"\n")
            output_file.write(str(response['source_addr_long'].enco$)
            output_file.write(str(response)+"\n")
            print(response)

        except Exception as e:
            print("Um erro foi dado:")
            print(e)

        except KeyboardInterrupt:
            break
        output_file.close()
        ser.close()

readConsumption()
```

Listing 5.5 – raspberry.py

A idéia dos códigos é que o arduino envie seu endereço MAC para diferenciá-lo na rede e uma string formatada como um JSON para que esse seja convertido no raspberry e as informações sejam extraídas.

6 Testes e Avaliação

6.1 Introdução

Assim como foi especificado pelo grupo no planejamento, foram consideradas duas fases de testes, uma para o software e outra para integração, sendo que na fase de integração também será testado o hardware, uma vez que o hardware não possui nenhuma habilidade de regras de negócio, o foco foi no software que as possui e a integração que era um ponto que envolvia a confirmação da captura dos dados e sua visualização.

6.2 Plano de Testes do Software

As funcionalidades foram divididas em funcionalidades críticas e funcionalidades não-críticas. As funcionalidades críticas são aquelas que, caso falhassem, interromperiam o desenvolvimento do projeto e as não-críticas eram funcionalidades que em caso de falha atrapalhariam o desenvolvimento do trabalho, se tornando um incômodo, porém não o interromperiam. Isso ajudou os membros na hora dos testes ao indicar a pessoa fazendo o teste se o teste deveria ser mais intensamente testado ou não. Cada teste é subdividido em testes de falha e teste de sucesso. No teste de falha, o sistema se encarrega de tratar informações ou comportamentos não esperados num fluxo de trabalho normal (ex: submissão de formulário em branco, erro de autenticação, objetos não encontrados no banco de dados, entre outros). No teste de sucesso, o sistema realiza as tarefas esperadas para um fluxo de trabalho normal (ex: submissão de dados válidos num formulário)

6.2.1 Funcionalidades Críticas

- realizar login
- visualizar o consumo através de gráficos por equipamento ou total
- Obtenção dos dados da AES eletropaulo
- importar os dados de consumo de um equipamento

6.2.2 Funcionalidades Não-Críticas

- exportar os dados de consumo de um equipamento

Com essas funcionalidades categorizadas, foram criados casos de teste para verificar o funcionamento do sistema.

1. Criação de entidade
2. Leitura de entidade
3. Atualização de entidade
4. Remoção de entidade

Feitos esses testes, foi possível descobrir erros, sendo os mais graves na visualização de dados no gráfico e na obtenção de dados do AES devido à mudanças da estrutura do HTML do site, o que impossibilitava a obtenção dos dados, o que mostrou que tal funcionalidade deve ser mantida periodicamente para que tal funcionalidade funcione sempre, pois não é possível prever tais mudanças.

6.3 Integração

Uma rota foi criada no sistema na parte web para receber requisições de cadastro de consumo pelo módulo coordenador. O objetivo do segundo teste foi detectar a presença do sensor na rede e cadastrá-lo caso não estivessem registrado no banco de dados. Além disso foi necessário verificar que existe um equipamento associado àquele sensor, caso contrário, o sistema não aceitará o registro de consumos. Para isso, o programa do primeiro teste foi reescrito para receber o pacote e então enviá-lo via http para cadastrar o consumo teste.

```

pi@raspberrypi: ~/Test
HTTP Error 500: INTERNAL SERVER ERROR
Um erro foi dado:
HTTP Error 500: INTERNAL SERVER ERROR
Um erro foi dado:
HTTP Error 500: INTERNAL SERVER ERROR
Um erro Foi dado:
HTTP Error 500: INTERNAL SERVER ERROR
Options
O1
Src_addr
e663
Src_addr_long
0013a20040e44285
{'source_addr_long': '\x00\x13\xa2\x00@\xe4B\x85', 'rf_data': '123123',
addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
Options
O1
Src_addr
e663
Src_addr_long
0013a20040e44285
{'source_addr_long': '\x00\x13\xa2\x00@\xe4B\x85', 'rf_data': '123123',
addr': '\xe6c', 'id': 'rx', 'options': '\x01'}

```

```

pi MINGW32:/c/Users/master/Henrique/smrule
0ms service=2ms status=304 bytes=110
-[35m2015-09-19T19:29:38.722928+00:00 heroku[router]:-[0m at=info method=GET path="/static/bootstrap/fonts/glyphicons-regular.woff" host=smrule-mi.herokuapp.com] request_id=320la351-c096-4a14-b443-fbe4213d6d3 fwd="191.183.26.189" dyno=web.1 connect=0ms service=2ms status=200 bytes=110
-[36m2015-09-19T19:29:31.640411+00:00 app[web.1]:-[0m 2015-09-19 19:29:51 [3] [INFO] 2 workers]
-[36m2015-09-19T19:30:08.727461+00:00 app[web.1]:-[0m 2015-09-19 19:30:08 [3] [INFO] 2 workers]
-[36m2015-09-19T19:30:23.743400+00:00 app[web.1]:-[0m 2015-09-19 19:30:23 [3] [INFO] 2 workers]
-[35m2015-09-19T19:30:35.325946+00:00 heroku[router]:-[0m at=info method=POST path="/consumption/create" host=smrule-mi.herokuapp.com request_id=0b98caf9-fcd-4154-852d-8c593fd6150b fwd="191.183.26.189" dyno=web.1 connect=0ms service=49ms status=201 bytes=207
-[36m2015-09-19T19:30:35.286267+00:00 app[web.1]:-[0m 0013a20040e44285
-[36m2015-09-19T19:30:35.324978+00:00 app[web.1]:-[0m /app/heroku/python/lib/python2.7/site-packages/django/db/models/fields/_init_.py:1474: RuntimeWarning: DateTimeField Consumption.moment received a naive datetime (2015-09-19 16:30:35.320389) while time zone support is active.
-[36m2015-09-19T19:30:35.324983+00:00 app[web.1]:-[0m RuntimeWarning)
-[36m2015-09-19T19:30:35.324985+00:00 app[web.1]:-[0m
-[36m2015-09-19T19:30:38.756803+00:00 app[web.1]:-[0m 2015-09-19 19:30:38 [3] [INFO] 2 workers

```

Figura 34 – Teste de criação de sensor e de consumo

Em um primeiro momento, quando o sensor nem está no sistema, o servidor retorna um erro HTTP 500 (erro de servidor), isso porque apesar do sensor ter sido criado no sistema, este não está vinculado a nenhum equipamento. Após vincular o sensor a um equipamento na tela de configurações, pode-se ver que o servidor então retorna um status 201, que mostra que o consumo foi criado.

6.4 Resultados

Os testes revelaram algumas necessidades adicionais do sistema, como a necessidade de manutenção periódica da aplicação na nuvem, assim como outras imperfeições que limitaram a exatidão do sistema. Porém os resultados foram positivos ao resultar em uma arquitetura escalável e modularizada, o que aumenta as possibilidades do sistema.

6.4.1 Telas principais

6.4.1.1 Listagem de equipamentos

Nome	Valor Nominal	Ações
Ar condicionado	1300,0	
Forno elétrico	800,0	
Geladeira	1231,0	

Pesquisar

Mostrando de 1 até 3 de 3 registros

Criar

Figura 35 – Listagem de equipamentos

6.4.1.2 Criar meta

Nova Meta

Nome	<input type="text" value="Meta 1"/>
Gasto Relativo (%)	<input type="text" value="0.80"/>
Equipmento	<input type="text" value="Geladeira"/>
<input type="button" value="Salvar"/>	

Voltar

Figura 36 – Criar meta

6.4.1.3 AES

AES Eletropaulo

Nome	Intervalo	TUSD	TE	Válido a partir de
B1 - RESIDENCIAL	qualquer consumo	0,198960	0,237150	04/07/2015
B1 - RESIDENCIAL - BAIXA RENDA	até 30 kW	0,067470	0,083000	04/07/2015
B1 - RESIDENCIAL - BAIXA RENDA	31 kW até 100 kW	0,115660	0,142290	04/07/2015
B1 - RESIDENCIAL - BAIXA RENDA	101 kW até 220 kW	0,173480	0,213430	04/07/2015
B1 - RESIDENCIAL - BAIXA RENDA	a partir de 220 kW	0,192760	0,237150	04/07/2015
B2 - COOPERATIVA DE ELETRIFICAÇÃO RURAL	qualquer consumo	0,139270	0,166000	04/07/2015
B2 - RURAL	qualquer consumo	0,139270	0,166000	04/07/2015
B2 - SERVIÇO PÚBLICO DE IRRIGAÇÃO	qualquer consumo	0,119380	0,142290	04/07/2015
B3 - DEMAIS CLASSES	qualquer consumo	0,198960	0,237150	04/07/2015

Figura 37 – AES

6.4.1.4 Configurar sistema

The screenshot shows the 'CONFIGURAÇÕES' (Configurations) section of the SiME system. At the top, there is a dropdown menu labeled 'Tipo de Renda' (Income Type) set to 'B1 - RESIDENCIAL - BAIXA RENDA'. Below this, under 'Equipamentos e Sensores' (Equipment and Sensors), there are three blue boxes representing 'Geladeira' (Refrigerator), 'Forno elétrico' (Electric Oven), and 'Ar condicionado' (Air Conditioner). Each box contains a yellow circular icon with a signal symbol, indicating a sensor is connected. The identifier 'sensor_0013a20040c27fd1' is visible below the first icon. A 'Sensores' (Sensors) section is also present.

Figura 38 – Configurações

6.4.1.5 Visualização dos dados medidos

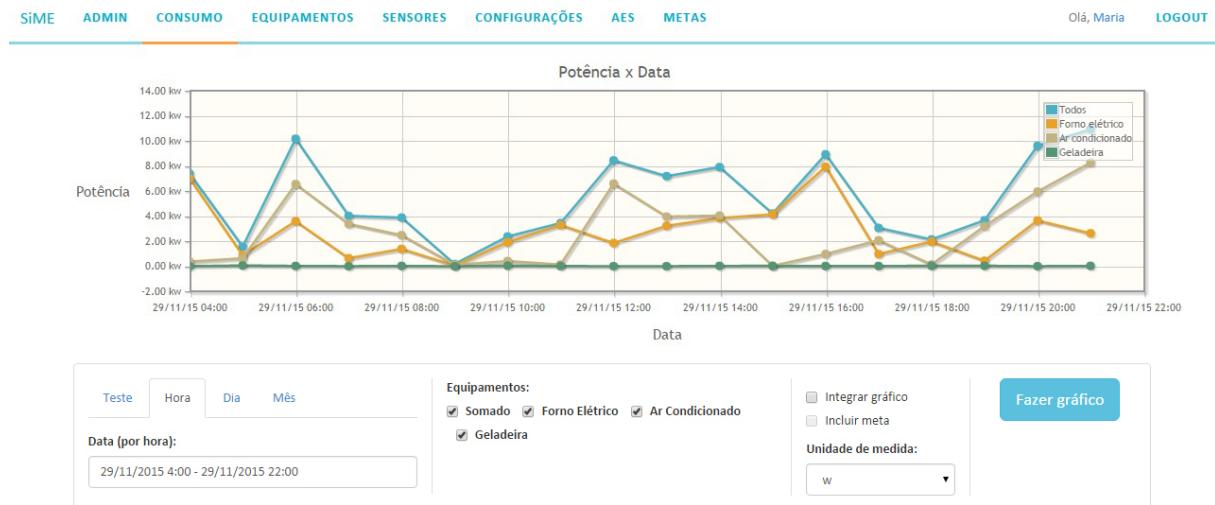


Figura 39 – Gráfico de consumo

7 Considerações Finais

O projeto permitiu o uso dos conhecimentos vistos em aula relacionados à microeletrônica, redes, sistemas digitais, engenharia de software, banco de dados, entre outros. Pelo fato da implementação ter envolvido uma grande variedade de conhecimentos cujas noções básicas foram ensinadas no percorrer da faculdade, o projeto foi muito interessante do ponto de vista de aprendizado, porém dificultou muito a implementação que divergiu um pouco da área de especialização do curso, mas ainda concentrando-se na engenharia.

Ainda relacionado, um grande desafio esteve na parte da implementação dos circuitos, uma vez que o projeto exigia o manuseio direto da rede elétrica, isso deixou os integrantes bem apreensivos, sendo que ao final do projeto o medo foi superado.

O sistema tinha como objetivo prover ao usuário do sistema uma noção do consumo de cada equipamento para que esse tivesse mais controle sobre seus gastos. Além disso, o sistema deveria ser simples de montar e de baixo valor. Pode se dizer que tais objetivos foram alcançados, apesar do alto custo de investimento inicial (vide apêndice A), sendo que cada equipamento adicional a ser monitorado custaria um adicional de aproximadamente 200 reais (contando o custo do dólar em aproximadamente 4 reais). A teoria envolvida também pode ser considerada simples, dado que hoje em dia há uma diversidade de fontes abertas para o aprendizado. Portanto, o gargalo para a construção do sistema se torna mesmo a vontade que o usuário tem em aprender as técnicas, tecnologias e ferramentas envolvidas e a disponibilidade monetária que esse possui.

8 Trabalhos Futuros

O trabalho ainda não passa de um protótipo, tendo vários pontos de melhoria. Das melhorias possíveis do sistema, podem ser citadas:

- Controle da rede de sensores pela aplicação ou outros meios: uma das funcionalidades envisionadas no início do projeto era a possibilidade de controlar a rede pela aplicação a rede para, por exemplo, controlar a periodicidade das amostragens dos módulos sensores ou forçar os módulos a dormir, o que possibilitaria ao sistema monitorar de uma maneira mais eficiente os equipamentos existentes. Tal funcionalidade é possível devido a utilização de XBee no sistema.
- Medição de potência real e reativa: no trabalho, uma das premissas foi o uso de um fator de potência unitário, o que pode, na maioria dos casos, ser aproximada em residências. Porém, se tais grandezas fossem medidas seria possível um controle da qualidade da rede, aproximando o usuário à qualidade do serviço do distribuidor. Ou então o uso do sistema em empresas, que usam mais intensamente a rede.
- Possibilitar vários usuários no sistema: ao possibilitar a entrada de mais de um usuário no sistema possibilitaria escalar o protótipo para um produto onde vários usuários poderiam, por exemplo, comparar seus gastos com outras residências similares aumentando o grau de educação dele.

Referências

- 1 EFLUL. 2015. Disponível em: <<http://www.eflul.com.br/consumidores/como-economizar-energia>>. 17
- 2 OPENENERGYMONITOR. *Open-source tools for energy monitoring and analysis.* 2015. <<http://openenergymonitor.org/emon/>>. 17
- 3 INC., N. 2015. Disponível em: <<http://neur.io/>>. 17
- 4 G1. *Aneel aprova aumento de até 17,04% nas tarifas da Eletropaulo.* 2015. <<http://g1.globo.com/economia/seu-dinheiro/noticia/2015/06/aneel-aprova-aumento-de-ate-1704-nas-tarifas-da-eletropaulo.html>>. 18
- 5 S.PAULO, O. E. de. *Reajustes deixam tarifa de energia em novo nível.* 2015. <<http://www.energia.sp.gov.br/lenoticia.php?id=733>>. 18
- 6 COPAM/EPE. Resenha mensal do mercado de energia elétrica. mar 2015. 18
- 7 DIGITAIS, P. M. Itu é um dos municípios que mais consomem energia na região. sep 2015. 18
- 8 PAULO, F. de S. *Desperdício consome 10% da energia elétrica no país, diz associação.* 2015. <<http://www1.folha.uol.com.br/mercado/2015/02/1586778-desperdicio-consome-10-da-energia-eletrica-no-pais-diz-associacao.shtml>>. 18
- 9 FERDOUSH, S. M. *A LOW-COST WIRELESS SENSOR NETWORK SYSTEM USING RASPBERRY PI AND ARDUINO FOR ENVIRONMENTAL MONITORING APPLICATIONS.* Dissertação (Mestrado) — University of North Texas, may 2014. 21
- 10 RAMOS, J. de S. B. *Instrumentação Eletrônica sem fio - Transmitindo Dados com Módulos XBee ZigBee e PIC16f877A.* 1. ed. [S.I.]: Editora Érica Ltda., 2012. 21, 22, 23
- 11 BELL, C. *Beginning Sensor Networks with Arduino and Raspberry Pi - Learn how to make Arduino and Raspberry Pi-based sensor networks.* [S.I.]: Technology in Action, 2013. 21
- 12 YICK BISWANATH MUKHERJEE, D. G. J. Wireless sensor network survey. *Computer Networks*, 2008. 22
- 13 CO., D. *Digi - Products Documentation.* 2015. <http://ftp1.digi.com/support/documentation/html/90001399/90001399_A/Files/XBee-concepts.html#_Toc384719504>. 22, 23
- 14 FOUNDATION, R. P. *Raspberry Pi Site Oficial.* 2015. <<https://www.raspberrypi.org/>>. 37
- 15 INC., H. *Heroku Platform.* 2015. <<https://www.heroku.com/home>>. 38

- 16 ELETROPAULO, A. 2015. Disponível em: <<https://www.aeseletropaulo.com.br/poder-publico/prazos-e-tarifas/conteudo/tarifa-de-energia-eletrica/>>. 56
- 17 INC., D. I. *Exploring XBee and XCTU*. 2015. <<https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu>>. 62

APÊNDICE A – Orçamento

Componentes	qtd	Total(R\$)	Total(US\$)
XBee 2mW PCB Antenna	4		US\$103,80
XBee Explorer Dongle	1		US\$24,95
Kit Raspberry Pi2	1	R\$309,89	
Sensores de Corrente	3		US\$29,85
XBee Shield	3		US\$44,85
Arduino Stackable Header Kit	4		US\$6,00
Arduino Uno - R3	3	R\$ 137,70	
Totais		R\$447,59	US\$209,45

Tabela 2 – Orçamento do projeto.