

Escola Politécnica da Universidade de São Paulo
Henrique Sussumu Matsui Kano
Mi Che Li Lee

Sistema de Monitoramento Residencial de Uso de Energia

São Paulo
2015

Escola Politécnica da Universidade de São Paulo
Henrique Sussumu Matsui Kano
Mi Che Li Lee

Sistema de Monitoramento Residencial de Uso de Energia

Dissertação apresentada à Escola Politécnica
da Universidade de São Paulo para a conclu-
ção do curso de graduação
Área de concentração: Engenharia de Com-
putação
Orientação: Professor Doutor Carlos Edu-
ardo Cugnasca

São Paulo
2015

Sumário

Sumário	i
Lista de Figuras	ii
Lista de Tabelas	iii
Lista de Códigos	iv
1 Introdução	1
1.1 Objetivo	1
1.2 Motivação	1
1.3 Justificativa	2
1.4 Organização	2
2 Aspectos Conceituais	4
2.1 Arquitetura MVC	4
2.2 Wireless Sensor Network	5
2.3 Padrão ZigBee e o XBee	5
2.4 Topologias de Rede	6
3 Especificação do Projeto	8
3.1 Escopo	8
3.2 Funções do Sistema	9
3.3 Requisitos não Funcionais	10
3.4 Premissas	10
3.5 Hardware	10
3.5.1 Módulo sensor	10
3.5.2 Coordenador	11
3.5.3 Circuitos	11
3.5.4 Peças	12
3.6 Software	17
3.6.1 Classes e atributos	17

<i>SUMÁRIO</i>	ii
3.6.2 Atores	21
3.6.3 Casos de uso	21
3.6.4 Descrição dos casos de uso	21
3.6.5 Regras de negócio	34
3.6.6 Tecnologia	34
4 Metodologia	36
5 Implementação	38
5.1 Aplicativo Web	38
5.1.1 Telas principais	39
5.1.2 Heroku	41
5.2 Módulo Sensor e Módulo Coordenador	41
5.2.1 XBee: configuração	41
5.2.2 Raspberry: Sistema Operacional	42
5.2.3 Raspberry: Adaptador Wifi	42
5.2.4 Raspberry x Arduino: comunicação	43
5.2.5 Verificador de tensão	46
5.2.6 Sensor de corrente	47
5.2.7 Integrando as partes	47
6 Testes e Avaliação	50
6.1 Introdução	50
6.2 Plano de Testes do Software	50
6.2.1 Funcionalidades Críticas	51
6.2.2 Funcionalidades Não-Críticas	51
7 Considerações Finais	52

Lista de Figuras

2.1	Arquitetura MVC	5
2.2	Topologia de uma rede zigbee	6
3.1	Esquema do Projeto	8
3.2	Diagrama de implantação	9
3.3	Diagrama de blocos do circuito que verifica a tensão	11
3.4	Non-invasive AC current sensor	12
3.5	Raspberry pi 2 modelo B	13
3.6	Arduino UNO R3	14
3.7	XBee Serie 2	15
3.8	XBee Explorer Dongle	16
3.9	XBee shield do arduino UNO	16
3.10	Headers usados no XBee shield	17
3.11	Diagrama de Classes	18
3.12	Diagrama de Navegação	21
4.1	Cronograma geral do projeto	37
5.1	Listagem de equipamentos	39
5.2	Criar meta	40
5.3	AES	40
5.4	Configurações	41
5.5	Adaptador wifi usado no raspberry	43
5.6	Teste de comunicação: Módulo Coordenador	43
5.7	Teste de comunicação: Módulo Sensor	44
5.8	Teste de comunicação: Montagem	44
5.9	Teste de comunicação Raspberry e Arduino	45
5.10	Teste de criação de sensor e de consumo	45
5.11	Teste do transformador	46
5.12	Teste de medição de voltagem	47

Lista de Tabelas

3.1 casos de uso.	22
7.1 Orçamento do projeto.	53

Lista de Códigos

5.1	teste-inicial-arduino.c	44
5.2	teste-inicial-raspberry.py	44
5.3	arduino.c	48
5.4	raspberry.py	49

Capítulo 1

Introdução

1.1 Objetivo

O trabalho propõe a construção de um sistema para monitorar o uso de energia dentro da residência. O sistema consistirá de dispositivos interligados por uma rede sem fio, composta por: uma rede local de sensores que fará o sensoriamento dos parâmetros e se comunicará com uma aplicação em nuvem; e a aplicação em nuvem, que apresentará as métricas enviadas pelos sensores numa interface simples e intuitiva, permitindo também o acesso remoto.

1.2 Motivação

A primeira das principais motivações do grupo foi a preocupação do desenvolvimento de um sistema que englobasse diversas áreas vistas ao longo do curso de engenharia para que fosse possível, ao final do trabalho, uma reflexão pessoal dos integrantes quanto a evolução técnica desses tópicos. A segunda motivação é a resolução de um problema muito comum na atualidade, que é o encarecimento da energia, utilizando como instrumentos as técnicas de automação residencial.

Tem-se como expectativa um produto facilmente aplicável no dia-a-dia e com alguns diferenciais em relação aos produtos existentes no mercado [?] [?] [?] [?], no sentido que usará equipamentos relativamente fáceis de obter com um resultado satisfatório. O projeto propõe um sistema no qual podemos identificar o consumo elétrico por equipamento, ao invés do consumo da rede residencial como um todo e, além disso, ser adaptável às metas de consumo, considerando a experiência do usuário.

1.3 Justificativa

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 do país e o custo foi repassado para os consumidores finais [? ?]. Por isso é imprescindível a tomada de atitudes por parte da população tanto para controlar os gastos na conta de luz quanto para a redução do consumo de eletricidade nas suas residências, aliviando a carga do sistema de produção e distribuição de eletricidade.

Trazendo o problema para a área da engenharia, e sabendo da grande gama de tecnologia disponível, a criação de ferramentas que podem nos auxiliar na monitoração e controle de gastos de energia é favorecida. Existem sistemas no mercado, ou prestes a entrar no mercado, que realiza a função de monitorar o consumo de energia residencialmente como a OpenEnergyMonitor[?], Neurio[?], Green Ant[?]. Entretanto há a preocupação de se desenvolver um sistema, em alguns meses, aproveitando a onda de desenvolvimento e hardware/software open-source, com padronizações e a comercialização de tecnologias de redes de sensores sem fio. Um sistema pode ser assim construído modularmente, permitindo o desenvolvimento rápido de protótipos altamente personalizáveis, de pouco custo e que consomem pouca bateria.

O sistema permitirá medir o consumo por aparelho, ao invés do consumo da rede elétrica residencial como um todo, sendo possível, então detectar possíveis aparelhos “vilões”, por excesso de consumo de energia.

O sistema dará uma visão quantitativa ao usuário, e isso é fundamental tomada de decisões conscientes, resultando em uma administração eficiente dos gastos residenciais.

1.4 Organização

O documento segue o seguinte formato: no capítulo um são apresentados os conceitos do projeto e diagramas do sistema de uma forma genérica, sem mencionar nomes ou marcas de componentes, porém são mencionando os principais módulos do sistema, assim como os nomes utilizados para esses módulos no trabalho todo.

No capítulo dois são listadas e apresentadas as peças e softwares que compõem o sistema propriamente ditos.

No capítulo quatro é apresentado o método de projeto adotada pelo grupo durante o desenvolvimento do projeto, da parte de projeto até sua implementação e conclusão.

No capítulo cinco é detalhado mais sobre como foi feita a implementação, citando detalhes mais técnicos de problemas, soluções e mudanças de projeto.

No capítulo seis é detalhado o procedimento de aceitação do sistema.

No capítulo sete são apontadas algumas considerações finais como comentários e resultados atingidos.

Além do capítulo sete serão colocados anexos extras que são considerados úteis para o entendimento do trabalho.

Capítulo 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 web monitorada por uma rede de sensores na qual cada sensor da rede transmite seus dados a um componente central que se comunica com essa aplicação em nuvem [?].

Serão 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. 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 2.1)

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.

View: A view é responsável por usar as informações disponibilizadas para produzir qualquer interface de apresentação que a aplicação pode necessitar.

Controller: essa camada lida com as requisições dos usuários. É responsável

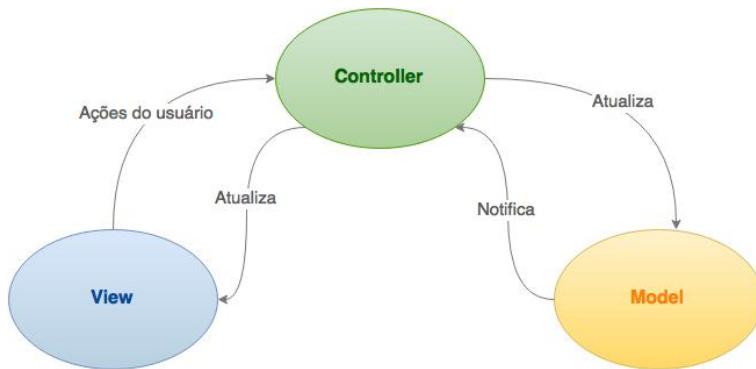


Figura 2.1: Arquitetura MVC

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 [?]: 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 chegam 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[?], 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 modos AT há apenas o envio de dados ponto-a-ponto na rede, porém no modo API é possível agir na rede durante sua operação como 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 [?]

2.4 Topologias de Rede

Como os nós dos sensores formam uma rede, é necessário analisar as possibilidades de redes. Como são usados XBees para formar essa rede de sensores, deve-se atentar aos tipos de rede possíveis [?] [?].

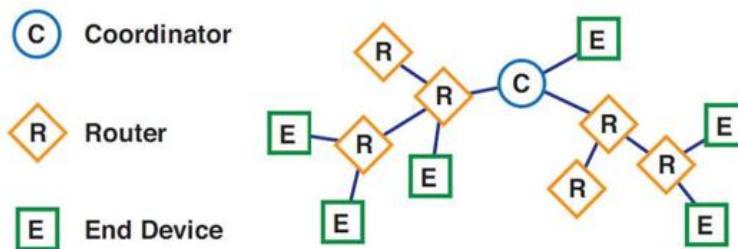


Figura 2.2: Topologia de uma rede zigbee

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. Concentra os dados de todos nós. Todas as redes possuem exatamente um nó desse tipo. Não possui a capacidade de dormir e, portanto, não deveria ficar em um dispositivo a bateria limitada. Pode se comunicar com os outros dois tipos de nós. Também não possuem a capacidade de dormir, logo não podem ser energizados com uma bateria limitada. Normalmente são usados para extender a área de uma rede;
- Roteador: Funciona apenas como uma ponte intermediária entre os endpoints e o coordenador. Pode se comunicar com todos os outros tipos de nós;
- 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;

Dado essas especificações e limitações, as redes formadas por esses componentes podem ser de três tipos [?]: 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.

Devido ao tamanho do projeto, não se justificava usar uma topologia muito complexa, uma vez que a rede teria no máximo quatro nós. Logo, nesse trabalho, é usado uma topologia em estrela.

Capítulo 3

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 3.1 e a disposição dos componentes através do diagrama de implantação da figura 3.2:

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

3.1 Escopo

Coletados os dados, resta mostrar informações úteis ao usuário. Com o consumo de corrente e a voltagem da tomada, é possível calcular a potência e alguns outros

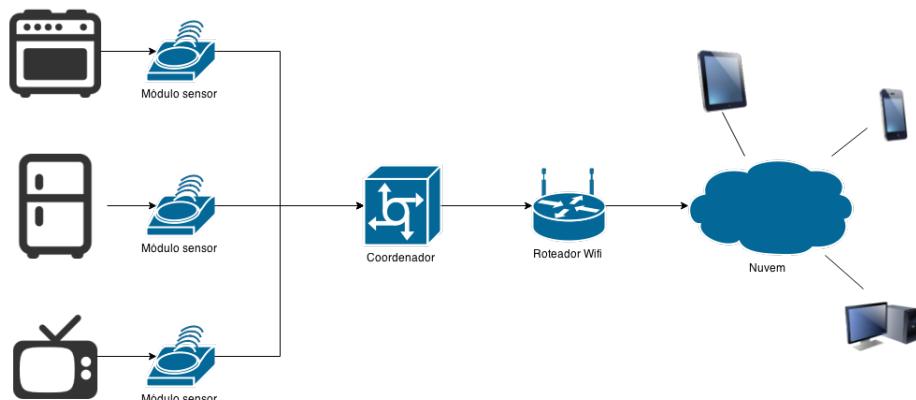


Figura 3.1: Esquema do Projeto

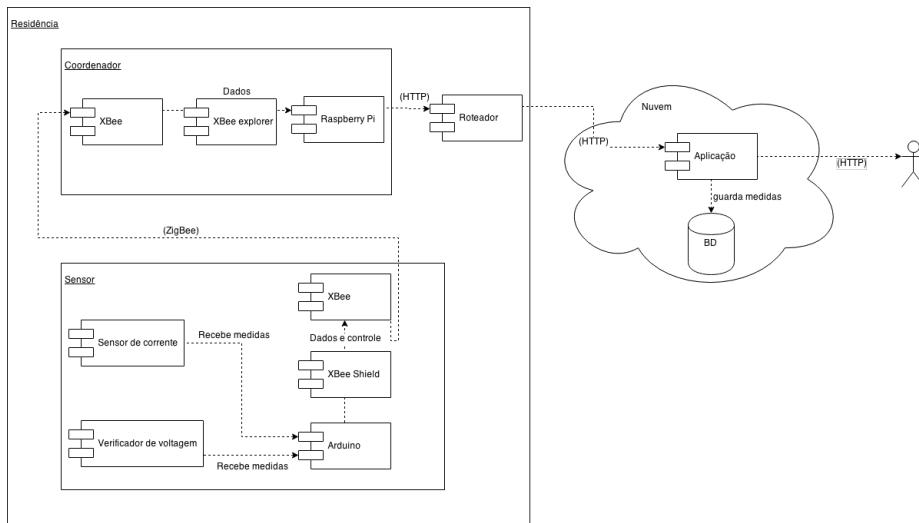


Figura 3.2: Diagrama de implantação

dados interessantes para o usuário. O aplicativo desenvolvido nesse trabalho será responsável por esse tratamento e a visualização dos dados coletados pelos sensores. As principais metas são as seguintes:

- informar ao usuário sobre o gasto de energia por equipamento em sua residência
- auxiliar o usuário a tomar decisões para diminuir o consumo de energia
- permitir o acesso às informações de consumo tanto localmente quanto remotamente

3.2 Funções do Sistema

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

Gerenciar equipamentos: O usuário poderá fazer criação, edição e remoção de equipamentos, os quais serão monitorados pelo sistema

Gerenciar sensores: Os módulos sensores são auto-detectados, e o usuário poderá editar ou removê-los

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

Gerenciar consumo: O módulo coordenador enviará consumos para o sistema estes serão cadastrados. O usuário poderá visualizar os consumos através de gráficos. Além disso o usuário poderá importar ou exportar dados de consumo

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

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

3.3 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
- os componentes físicos do sistema portáteis

3.4 Premissas

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

1. O usuário deve morar em São Paulo
2. Será considerado um fator de potência ideal

3.5 Hardware

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

3.5.2 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

3.5.3 Circuitos

VERIFICADOR DE TENSÃO

No circuito de cada módulo de sensor, são feitas detecções da voltagem (110V ou 220V) para cálculos de potência. O circuito da Figura 3.3 foi simulado no PSpice para esse propósito, com componentes com parâmetros não finais, para uma análise lógica do bloco. O objetivo do circuito da figura dois é indicar se a tensão na tomada é 220V ou 110V, retornando 5V ou 0V, respectivamente.

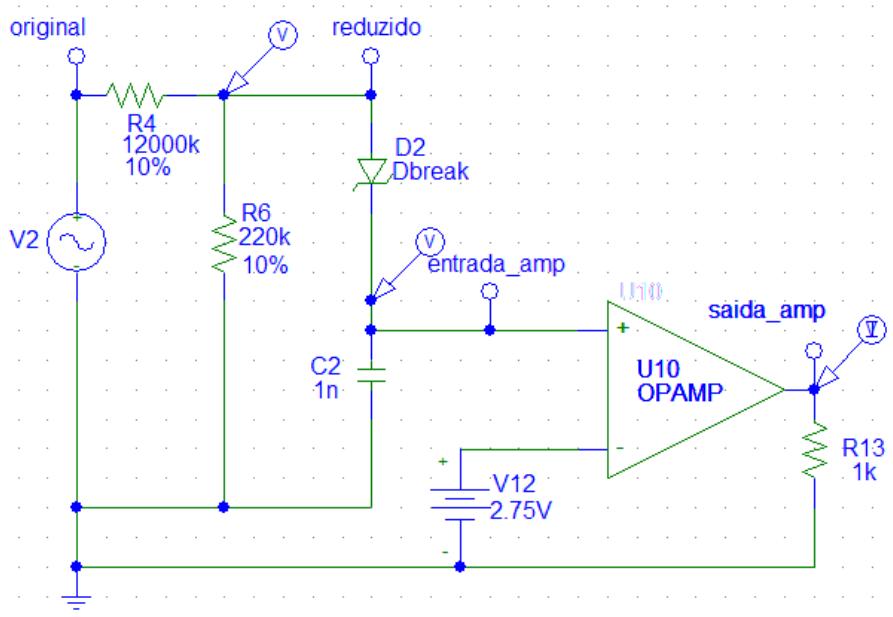


Figura 3.3: Diagrama de blocos do circuito que verifica a tensão

3.5.4 Peças

SENSOR DE CORRENTE NÃO-INVASIVO AC



Figura 3.4: 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 indutor respondendo a um campo magnético formado em volta do fio condutor. Este, em particular, suporta até 30A, e necessita de um resistor de saída para obter a medida desejada em voltagem.

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

RASPBERRY PI 2 MODELO B

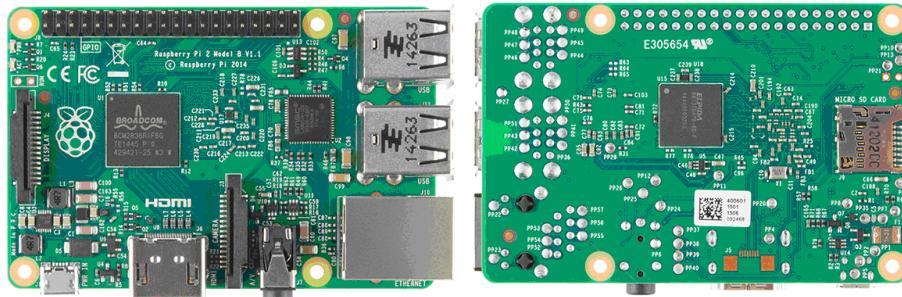


Figura 3.5: Raspberry pi 2 modelo B

A Raspberry Pi 2 Modelo B (figura 3.4) 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

ARDUINO UNO

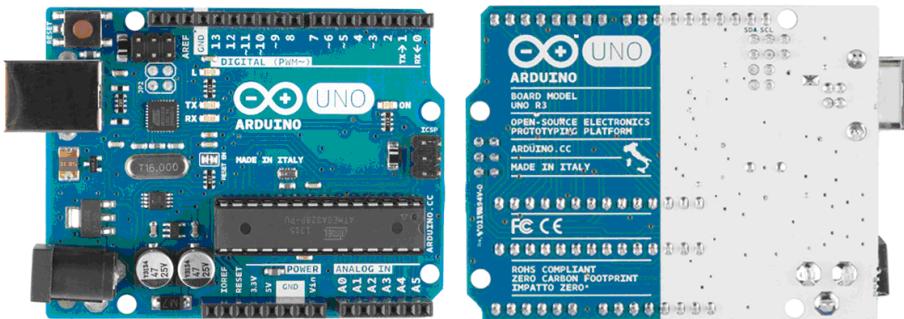


Figura 3.6: Arduino UNO R3

Arduino é uma placa programável open-source . No projeto em questão é utilizado como o microcontrolador, que 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
- Voltagem de entrada - 7-12V
- 14 Pinos Digital I/O (6 PWM de saída)
- 6 Inputs Analógicos
- 32k de memória Flash
- 16Mhz de Relógio

XBEE



Figura 3.7: XBee Serie 2

É um módulo que permite uma comunicação simples e confiável entre microcontroladores, 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

XBEE EXPLORER DONGLE

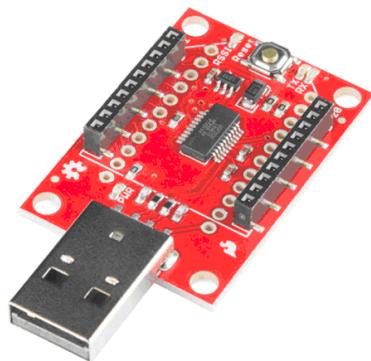


Figura 3.8: XBee Explorer Dongle

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 USB-serial, que traduz os dados entre o computador e o XBee. Possui um botão de reset e um regulador de voltagem para suprir a voltagem 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.

XBEE SHIELD

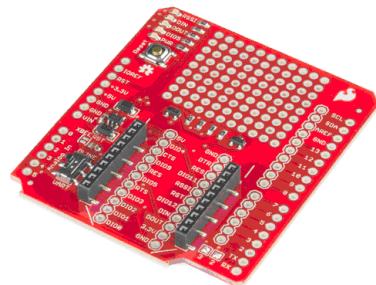


Figura 3.9: XBee shield do arduino UNO

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.

ARDUINO STACKABLE HEADER KIT - R3

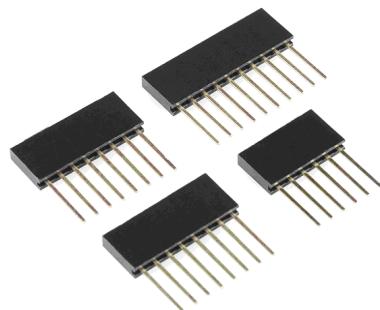


Figura 3.10: Headers usados no XBee shield

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.

3.6 Software

3.6.1 Classes e atributos

A seguir estão descritas as classes do sistema e seus respectivos atributos. As classes implementadas estão representadas pelo diagrama de classes (figura 3.11).

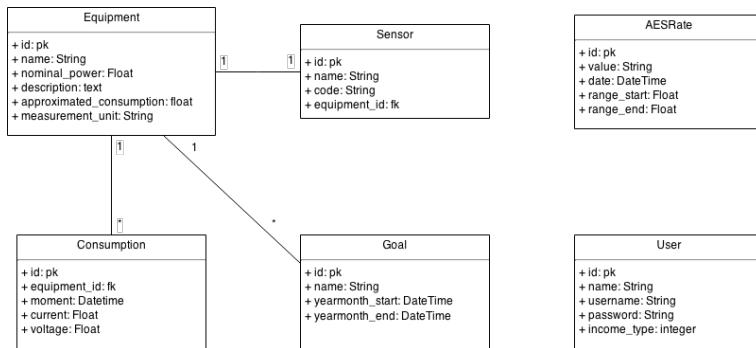


Figura 3.11: Diagrama de Classes

EQUIPMENT

Classe: Equipment**Descrição:** um equipamento monitorado**Atributos:**

1. id (integer): Identificador do equipamento
2. name (String): Nome do equipamento
3. description (Text): descrição do equipamento
4. nominal_power (float): potência do equipamento descrita no manual do equipamento
5. measurement_unit (String): unidade do valor dado em nominal_power
6. approximated_consumption (float): consumo aproximado do equipamento dado pelo fabricante

SENSOR

Classe: Sensor**Descrição:** um sensor do sistema**Atributos:**

1. id (integer): identificador do sensor dentro do software.
2. name (String): nome dado pelo usuário para o sensor

3. code (String): identificador do sensor entre outros sensores. Informação configurada no próprio módulo do sensor.
4. equipment_id (integer): equipamento a qual está associado

CONSUMPTION

Classe: Consumption

Descrição: Representa uma medida feita de um equipamento em um dado instante

Atributos:

1. id (integer): identificador do consumo
2. moment (DateTime): a data e a hora de quando foi feita a medida
3. current (float): corrente no momento da medida em amperes
4. voltage (float): voltagem da tomada do equipamento. 220V ou 110V

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 com Criptografia): Senha do usuário usada para efetuar o login
5. income_type (String): O tipo de renda do usuário, Residencial ou Non-Residencial de baixa renda, definida pela AES eletropaulo.

GOAL

Classe: Goal

Descrição: Representa uma meta de consumo por mês

Atributos:

1. id (integer): identificador da meta

2. name (String): Nome da meta
3. value_in_percent (float): Consumo pretendido em percentagem (em relação ao mês anterior)
4. yearmonth_start (DateTime): Mês/Ano de início do período da meta
5. yearmonth_end (DateTime): Mês/Ano de fim do período da meta

AESRATE

Classe: AESRa

Descrição: Representa a taxa de conversão da AES eletropaulo de kilowatts hora para reais. Esses valores obtidos através do site da AES Eletropaulo: <https://www.aeseletropaulo.com.br/poder-publico/prazos-e-tarifas/conteudo/tarifa-de-energia-eletrica>

Atributos:

1. id (integer): identificador da taxa de conversão
2. value (float): O valor da taxa de conversão no instante
3. date (DateTime): O instante que a taxa de conversão foi buscada
4. range_start (float): O início da faixa que define a taxa de conversão
5. range_end (float): O fim da faixa que define a taxa de conversão

No sistema é possível cadastrar apenas uma entidade principal os equipamentos eletrônicos monitorados, sendo prevista a possibilidade de que um sensor possa mudar de um equipamento para outro, sendo que tal mudança deve ser cadastrada no sistema pelo usuário na tela de configurações, como mostrado na tela de configurações no diagrama de navegação (figura 3.12). Essa possibilidade de mudança da configuração dos sensores explica a relação do equipamento e as medidas de consumo: caso o usuário troque o sensor de equipamento ainda será possível visualizar dados anteriores de outros equipamentos já monitorados.

Os sensores devem ser criados automaticamente pelo sistema uma vez que eles sejam colocados no sistema. Eles enviam um sinal inicial que informa seu identificador para que o sistema o cadastre. O usuário poderá, então, colocar um nome que preferir nesse sensor. Porém, como não há qualquer informação que o sistema pode indicar ao sistema em qual aparelho ou em que tipo de aparelho o sensor foi instalado, tal associação deve ser feita manualmente antes que os dados comecem a ser coletados.

Adquiridas as medidas, é possível visualizar esses dados em uma tabela de consumo na tela de consumo. Nessa tela, é possível construir o gráfico do consumo em função de vários períodos de tempo, no caso, o consumo por dia e por mês, assim como metas de consumo de energia dos equipamentos selecionados e previsões de consumo que serão calculadas a partir do consumo nominal dos equipamentos.

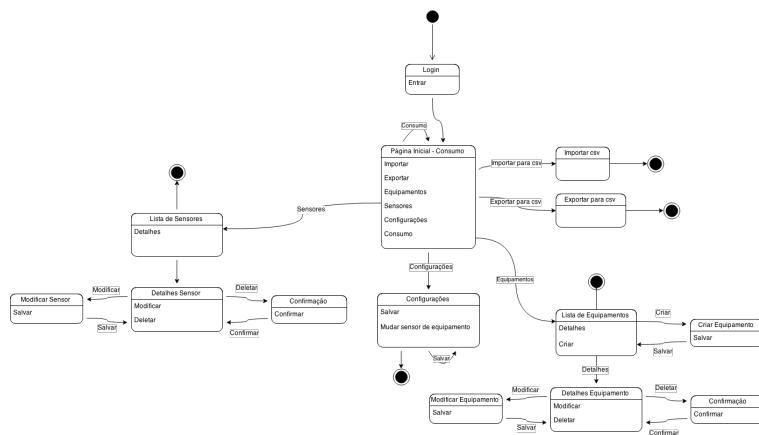


Figura 3.12: Diagrama de Navegação

3.6.2 Atores

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

3.6.3 Casos de uso

Os casos de uso do sistema estão listados na tabela 3.1.

3.6.4 Descrição dos casos de uso

A seguir são descritos os casos de uso do sistema.

CASO DE USO 1: GERENCIAR CONTA

CASO DE USO 1.1: FAZER CADASTRO

Descrição: inserção de um novo usuário comum no sistema

Evento iniciador: solicitação de cadastro

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 3.1: casos de uso.

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)

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)

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

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)

CASO DE USO 2: GERENCIAR EQUIPAMENTOS

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)

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)

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)

CASO DE USO 3: GERENCIAR SENSORES

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)

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)

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)

CASO DE USO 4: GERENCIAR METAS

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)

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)

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)

CASO DE USO 5: GERENCIAR CONSUMOS

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)

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)

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)

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)

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)

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)

3.6.5 Regras de negócio

As medidas serão feitos em uma taxa de uma vez a cada dez minutos, o que leva a 6 amostras por hora, 144 por dia e 4320 por mês. Dado que cada amostra é uma linha de uma tabela, torna-se necessário algum tipo de redução desses dados para que o banco na nuvem não chegue a seu ponto máximo, dado que no projeto será usado um serviço cloud com taxa gratuita e, portanto, muito limitada. Logo, foi decidido que a janela temporal que o usuário conseguirá visualizar será restrita ao mês anterior e o mês atual quando o gráfico for uma função das horas do dia, e os outros dados mais antigos serão agrupados em meses para que ainda possa ser possível visualiza-los quando o gráfico for uma função dos meses. Caso seja necessário, será usado um plano pago, o que resolveria esse problema.

3.6.6 Tecnologia

DJANGO E PYTHON

Para o aplicativo será utilizado o framework Django, 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. Há também algumas outras razões para escolher tais ferramentas:

Uma das vantagens do python nesse caso é uma pequena vantagem de compatibilidade, uma vez que o “pi” em “raspberry pi” vem de “python”, já que o hardware foi construído com a intenção do aprendizado de programação com python [?] (apesar da possibilidade de usar outras linguagens). Outra vantagem provinha do conhecimento prévio dos integrantes do grupo e do framework que reduzia muito o tempo de desenvolvimento. E por último, a possibilidade de, se necessário, rodar o servidor da aplicação no próprio raspberry sem a necessidade de acrescentar muitos outros módulos externos, uma vez que no raspberry python já é uma linguagem nativa, sendo necessário instalar apenas as dependências do django.

HEROKU

Heroku é uma plataforma em nuvem que fornece múltiplos serviços para dar suporte a uma aplicação web. Desenvolvedores web podem fazer deploy de aplicações em linguagens como Node, Ruby, Java, PHP, Python, Go, Scala, ou Clojure, e Heroku o deixará pronto para a utilização do público e a manterá no ar sem a necessidade da intervenção do desenvolvedor. É uma opção para os desenvolvedores que precisam se focar na criação e atualização da aplicação, sem se deixar distrair pela parte de hardware e infraestrutura.

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, porém, existe a opção gratuita que permite colocar uma aplicação no ar, suficiente para atender tráfegos pequenos. Nos planos pagos, o sistema é escalável (pode-se alterar o limite do número de processos executando na máquina do sistema, memória RAM, entre outros) para atender a momentos de tráfego mais intenso.[?]

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

Capítulo 4

Metodologia

Finalizada uma primeira perspectiva na primeira parte do projeto, primeiramente foi decidido um planejamento de como deveria ser o percorrer do projeto ao longo do período de desenvolvimento do trabalho.

O primeiro trabalho a ser feito, com uma maior prioridade, claramente era a compra, uma vez que tal tarefa era um trabalho cujo tempo de resposta (entrega, no caso) era totalmente indendente dos esforços dos componentes do time. Por essa razão, nessa primeira tarefa, os integrantes focaram-se em aprender a usar os componentes comprados, projetar onde e como os componentes seriam utilizados.

Feito o pedido de compras, o segundo passo foi a implementação da parte do sistema que envolvia o software estudado. Para a implementação do software, o grupo utilizou de conhecimentos adquiridos nos períodos de estágio e nas aulas para se organizar e implementar o software o mais rápido possível para que uma maior atenção pudesse ser dada ao hardware, uma vez que esse era a parte do projeto cujo conhecimento era mais reduzido. Para tanto, o sistema utilizado foi: a repartição do software em histórias, e o registro dessas repartições em um organizador (no caso, o aplicativo de gerenciamento de tarefas Trello).

Durante o período de estágio, como as possibilidades de encontros dos membros era menor, as tarefas eram então distribuídas aos membros e uma vez por semana, normalmente no sábado, era realizado um pequeno encontro para discutir o que foi feito, juntar as partes e decidir as próximas tarefas a serem feitas, sendo que o prazo que o grupo escolheu para o software, foi de até o final do período de estágio (final de agosto de 2015) o componente de software estivesse terminado e testado, ou, na pior das hipóteses, terminado.

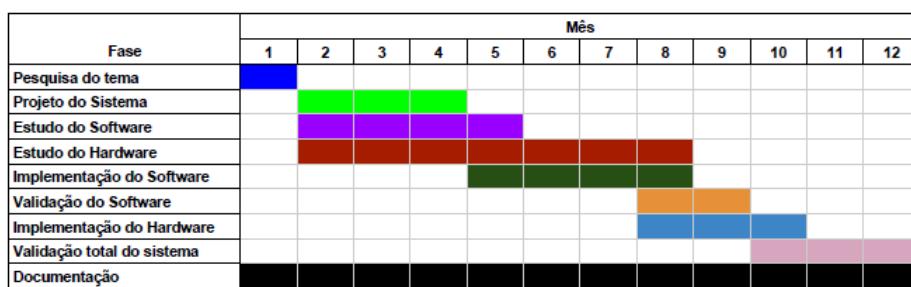


Figura 4.1: Cronograma geral do projeto

Capítulo 5

Implementação

A fase de implementação envolve o desenvolvimento do software e a configuração e montagem dos componentes físicos especificados na seção 3.

5.1 Aplicativo Web

O componente de software teve como base de implementação o framework para desenvolvimento web Django, para a linguagem python. Várias funções especificadas anteriormente possuem uma certa interdependência entre elas, por exemplo, a função Criar Meta necessita que a função Criar Equipamento esteja previamente desenvolvida. Para isso, adotou-se a seguinte ordem de implementação de funções:

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

5.1.1 Telas principais

LISTAGEM DE EQUIPAMENTOS

Nome	Valor Nominal	Ações
Ar condicionado	1300,0	
Forno elétrico	800,0	
Geladeira	1231,0	
Máquina de secar	2000,0	

Mostrando de 1 até 4 de 4 registros

[Criar](#)

Figura 5.1: Listagem de equipamentos

CRIAR META

Nova Meta

Nome	INVERNO - 2015
Gasto Relativo	0.95
Intervalo	10/2015
Equipamento	Geladeira
<input type="button" value="Salvar"/>	

Figura 5.2: Criar meta

AES

AES Eletropaulo

Última atualização: 14/09/2015

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

Mostrando de 1 até 9 de 9 registros

Figura 5.3: AES

CONFIGURAR SISTEMA

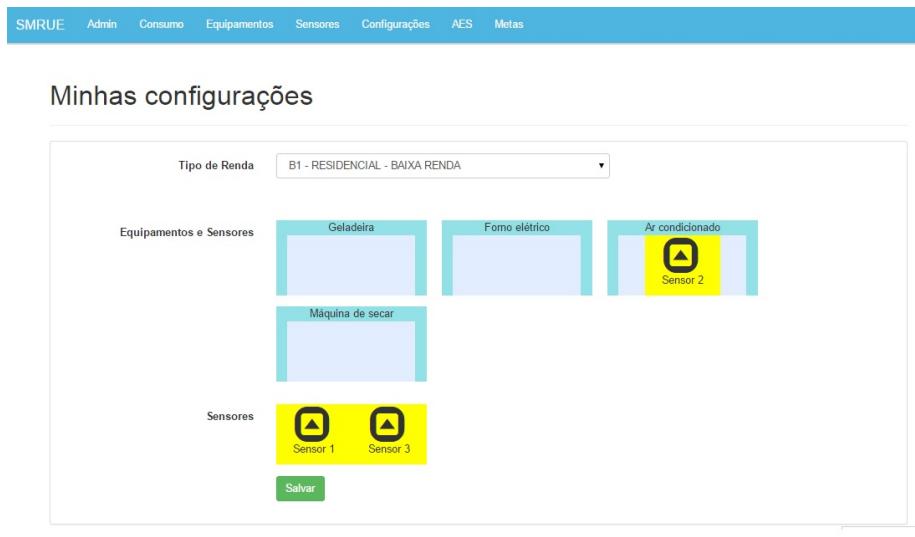


Figura 5.4: Configurações

5.1.2 Heroku

Heroku é uma plataforma que oferece serviços em cloud. Esse serviço é usado no projeto para hospedar a aplicação desenvolvida. A grande vantagem do serviço é o fato de ser gratuita com algumas limitações, que não se tornaram empecilhos para o trabalho, e retirar a responsabilidade do módulo coordenador de segurar a aplicação.

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 [?].

As configurações utilizadas foram:

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

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 sobre um sistema operacional. O sistema operacional usado é o Raspbian, 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 Raspbian ser instalado no cartão de memória, foi necessário configurar o adaptador Wifi. O modelo usado é o TP-Link TL-WN723N.

PROCEDIMENTOS



Figura 5.5: Adaptador wifi usado no raspberry

5.2.4 Raspberry x Arduino: comunicação

Para testar a comunicação entre o Módulo Coordenador (figura 5.6) e o módulo sensor (figura 5.7), foi montado o esquema representado na figura 5.8 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 5.6: Teste de comunicação: Módulo Coordenador



Figura 5.7: Teste de comunicação: Módulo Sensor

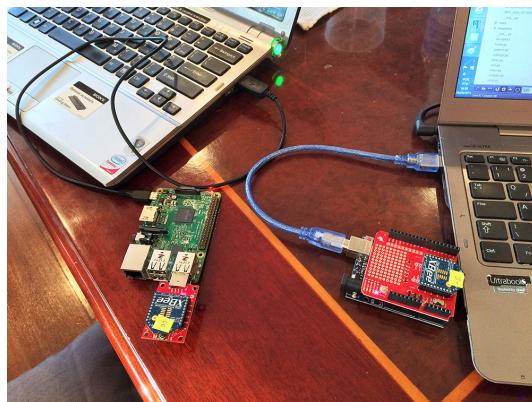


Figura 5.8: Teste de comunicação: Montagem

```
// 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:~/Test $ python python_xbee_test.py
[{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'},
{'source_addr_long': '\x00\x13\xa2\x00\xe4\x85', 'rf_data': '123123', 'source_addr': '\xe6c', 'id': 'rx', 'options': '\x01'}
]

```

Figura 5.9: Teste de comunicação Raspberry e Arduino

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.

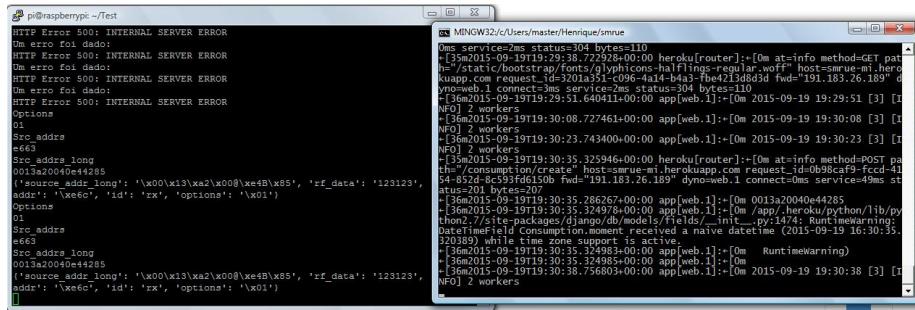


Figura 5.10: Teste de criação de sensor e de consumo

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.

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 110V sendo rebaixada para 7,2 V.

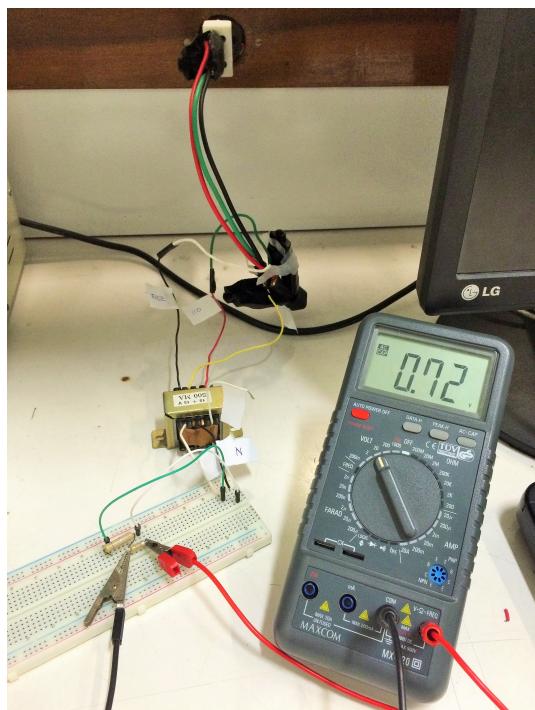


Figura 5.11: Teste do transformador

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 110V no arduino para enviar ao raspberry.

A figura 5.12 mostra a montagem para tal e a figura ?? mostra a saída no osciloscópio.

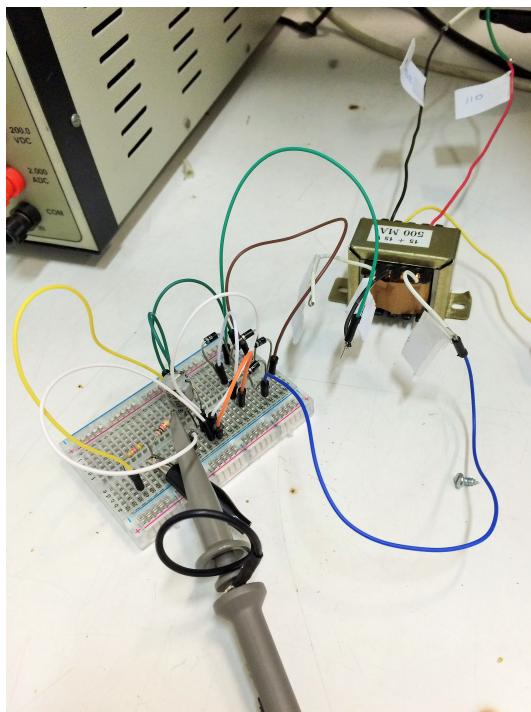


Figura 5.12: Teste de medição de voltagem

Em seguida a saída retificada e dividida foi posta como entrada para um pino de leitura analógica do Arduino para a medição, para 110V e em seguida, 220V. A saída resultante no terminal serial do arduino ?? foi a seguinte

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 voltagem 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,

5.2.7 Integrando as partes

Após os testes anteriores de comunicação entre o módulo sensor, módulo coordenador e a medição de corrente e voltagem, o conjunto foi testado. O código 5.3 foi carregado no arduino para a coleta dos dados de corrente e voltagem 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 5.3: arduino.c

O módulo coordenador foi configurado para executar o seguinte programa em python (código 5.4), 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 httpplib, 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 = urlopen(req)

def readConsumption():
    output_file = open(output, "w")
    while True:
        try:
            response = xbee.wait_read_frame()
            #sendDataToWeb(response)
            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 5.4: raspberry.py

Capítulo 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 decosnumo 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

Capítulo 7

Considerações Finais

O projeto permitiu o uso dos conhecimentos vistos em aula relacionados à micro-eletrônica, 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.

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

Tabela 7.1: Orçamento do projeto.