UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

IGOR ALEXANDRE DUTRA E SILVA

Sensor tensão-corrente inteligente com monitoramento e controle *on-line* por *smartphone*

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Edison Pignaton de Freitas
Co-orientador: Prof. Dr. Fausto Bastos Líbano

Porto Alegre
2016

"If I have seen further it is by standing on the shoulders of Giants."
(Sir Isaac Newton)

RESUMO

Embora não seja tecnologicamente complexa, a aferição do consumo de energia elétrica dos diversos aparelhos que equipam as residências não é uma tarefa trivial para o usuário comum. Usualmente, essa medição demanda a desmontagem da tomada ou a exposição dos fios do equipamento a ser aferido. Por isso, essa aferição é dispensada e consequentemente não se tem ideia do quanto cada aparelho consome de fato e, portanto, qual o impacto dele nos gastos domésticos com energia elétrica.

Por outro lado, existe uma preocupação crescente com a eficiência energética desses aparelhos, tanto por causa de seu impacto ambiental quanto por causa do seu impacto econômico. Com o barateamento de componentes eletrônicos como microcontroladores, dispositivos de automação residencial e de comunicação sem fio, bem como com a popularização de plataformas de automação, como Arduino, tornou-se viável o desenvolvimento de dispositivos de baixo custo, baixo consumo e de fácil utilização para realizar o monitoramento em tempo real do consumo de energia elétrica individual de todos os dispositivos elétricos e eletrônicos que são utilizados nos ambientes residenciais.

Ante o exposto, esse trabalho propõe-se a apresentar um dispositivo com as características citadas que permita o monitoramento instantâneo do consumo de tensão elétrica e de corrente elétrica dos aparelhos, permitindo a transmissão dos dados por meio de comunicação sem fio, sua visualização e seu processamento em um *smartphone*. Os recursos empregados na prototipagem são a placa Arduino Uno R3, o galvanômetro Allegro ACS712, o sensor de tensão GBK Robotics P8 e o módulo *Bluetooth* HC-05.

Palavras-chave: Sistemas Embarcados. Eficiência energética. Arduino Uno R3. ACS712. HC-05. *Bluetooth*. P8.

Smart current-tension sensor with smartphone control and monitoring

ABSTRACT

Although it is not technologically complex, the measurement of the energy consumption in the great variety of devices that equip consumers' homes is not a trivial task to the common user. Usually, this measurement demands either the disassembling of the electrical outlet or the exposure of the wiring of the device. For that reason, this measurement is dismissed and, consequently, people have no idea about how much each appliance consumes indeed and, therefore, what is its impact on the domestic electric energy budget.

On the other hand, there is a growing concern about the energetic efficiency of such devices, either because of its environmental impact or because its economic impact. Along with the lowering of the prices of electronic components such as microcontrollers, house automation devices and wireless communications, as well as prototyping platforms became popular, like Arduino, the development of low-cost, low-consumption, easy-to-use devices to monitor the electrical consumption of the electrical and electronic devices used in home environments became feasible.

Before these facts, this work proposes to present a device with the mentioned characteristics that allows the instant monitoring of the current and tension consumption of such appliances, allowing the wireless transmission of these data, as well as its presentation and processing on a smartphone. The employed resources in prototyping were the Arduino Uno R3 board, the Allegro ACS712 galvanometer, the GBK Robotics P8 tension sensor and the Bluetooth module HC-05.

Keywords: Embedded Systems. Energetic efficiency. Arduino Uno R3. ACS712. HC-05. Bluetooth. P8.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A	Ampère
API	*Application Programming Interface* (Interface de Programação de Aplicação)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
DSP	*Digital Signal Processor* (Processador de Sinais Digitais)
FTP	*File Transfer Protocol* (Protocolo de Transferência de Arquivo)
HTTP	*Hyper-Text Transfer Protocol* (Protocolo de Transferência de Hipertexto)
Hz	Hertz
I/O	*Input/Output* (Entrada/Saída)
ISA	*Instruction Set Architecture* (Arquitetura de Conjunto de Instruções)
JVM	*Java Virtual Machine* (Máquina Virtual Java)
kWh	Quilowatt-hora
MB	*Megabyte*
OS	*Operating System* (Sistema Operacional)
PC	*Personal Computer* (Computador Pessoal)
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
SI	Sistema Internacional de Unidades
SO	Sistema Operacional
SoC	*System on a Chip* (Sistema em um *Chip*)
TCP/IP	*Transmission Control Protocol / Internet Protocol* (Protocolo de Controle de Transmissão / Protocolo de Internet)
USB	*Universal Serial Bus*
W	Watt

SUMÁRIO

1 INTRODUÇÃO

Devido à falta de informação, é muito raro que um consumidor saiba, com um grau de precisão razoável, quanta energia consomem seus eletrodomésticos e eletro-eletrônicos. Entre as qualidades amplamente anunciadas para esses aparelhos por fabricantes e vendedores, dificilmente são encontradas características relacionadas ao consumo energético, exceto pelo Selo Procel (PROCEL INFO, 2006), o qual apenas categoriza o aparelho de acordo com seu consumo médio. Além disso, características reais de consumo podem variar muito se comparadas aos valores nominais, tanto em função do perfil de uso quanto em função do ambiente onde esses aparelhos são utilizados. Mesmo para o consumidor que é mais preocupado com o gasto com energia elétrica, é difícil realizar a aferição dessas informações, seja por causa da complexidade da utilização de instrumentos para tais fins, seja pelo custo dos mesmos. Além disso, passou a vigorar no Brasil, no ano de 2015, o sistema de bandeiras tarifárias (ANEEL, 2016), o qual insere mais um grau de complexidade no cálculo do consumo de energia elétrica. Por fim, também há uma crescente preocupação com o impacto ambiental que a geração de energia elétrica pode provocar (INATOMI e UDAETA, 2005), o qual também acaba por se traduzir em custos mais elevados para o consumidor final (DIEESE, 2015).

O acesso à eletrônica para sistemas embarcados tem aumentado muito nos últimos anos, tanto em função do barateamento dos componentes eletrônicos quanto em função da criação de plataformas de prototipação que facilitam a sua montagem e trazem ambientes de desenvolvimento utilizando linguagens de mais alto nível. Na carona dessas plataformas de prototipação, diversos componentes têm se popularizado, especialmente sensores e atuadores. Por outro lado, há a recente popularização de *smartphones* no Brasil, bem como a evolução de seu *hardware* de forma a permitir o uso de novas tecnologias e de *softwares* mais complexos. Com a popularização das plataformas de sistemas embarcados, de *smartphones* e de tecnologias que permitem a integração de ambos, a combinação desses elementos permite a criação de diversas soluções para as mais variadas aplicações.

Entre essas aplicações, podem ser citadas automação residencial com controle pelo *smartphone*, sensoriamento de ambientes agrícolas em tempo real[1], tecnologias assistivas para pessoas portadoras de necessidades especiais e medição de grandezas físicas com apresentação dos dados no *smartphone*.

[1] Nesse contexto, "tempo real" refere-se à disponibilidade das informações a qualquer momento, e não ao tempo de resposta do sistema em questão.

1.1 Objetivos do Trabalho

Na esteira da integração entre sistemas embarcados e *smartphones* citada na seção anterior, o presente trabalho propõe-se a estudar as possibilidades e os recursos disponíveis comercialmente para a criação de um sensor tensão-corrente inteligente e de baixo custo com monitoramento e controle *on-line* por *smartphone*, bem como a projetar e criar um protótipo funcional do sensor e do aplicativo propostos.

Propõe-se utilizar a plataforma de prototipação eletrônica Arduino/Genuino Uno R3 (ARDUINO, 2016) para o projeto do sensor e o sistema operacional Android para o projeto do aplicativo de gerenciamento e controle.

1.2 Organização do Texto

O presente trabalho está dividido em seis capítulos. O primeiro capítulo consiste na nesta Introdução. O segundo capítulo apresenta a definição de Sistemas Embarcados, um modelo para a sua representação e os componentes desse modelo, além de trazer conceitos básicos de grandezas elétricas. O terceiro capítulo apresenta o Sistema de Monitoramento de Consumo de Energia Elétrica, define seus requisitos e seu projeto, dividido entre o sensor e o aplicativo para *smartphone*. O quarto capítulo apresenta o desenvolvimento e a prototipação do projeto. O quinto capítulo tem por objetivo avaliar o projeto quanto ao atendimento aos requisitos definidos. O sexto e último capítulo apresenta as conclusões, as dificuldades encontradas durante sua execução, lista as limitações da solução e sugere melhorias.

2 TECNOLOGIAS E CONCEITOS APRESENTADOS

2.1 Sistemas Embarcados

Estima-se que mais de 90% dos processadores fabricados em todo o mundo sejam destinados a dispositivos que não são computadores de propósito geral (CHASE, 2007). Entre as aplicações a que se destinam esses componentes, encontram-se telefones celulares, roteadores e sistemas de controle em veículos (REIS, 2015). Esses sistemas, juntamente com muitos outros, compõem o universo dos Sistemas Embarcados. Essa denominação remonta ao final da década de 1960, quando um pequeno programa escrito em *assembly* para um telefone passou a ser utilizado em outras aplicações (CHASE, 2007). Para equipamentos de grande porte, considera-se que o primeiro sistema embarcado foi o Apollo Guidance Computer (AGC), sistema de navegação do Projeto Apollo[2], que utilizava circuitos integrados de primeira geração, constituído por 4100 portas NOR com três entradas cada (LIMA, 2014). Desde então, a evolução tecnológica tem permitido a expansão da definição desse conceito. A definição de Sistema Embarcado utilizada neste trabalho será apresentada a seguir.

2.1.1 Definição de Sistema Embarcado

Um Sistema Embarcado é um sistema composto por *hardware* e *software* que possui um propósito específico, em contraste com os sistemas computacionais de propósito geral, como PCs, servidores e *notebooks* (NOERGAARD, 2013). Sistemas Embarcados usualmente possuem requisitos não funcionais mais restritivos, especialmente no tocante ao tamanho físico, ao consumo de energia, à durabilidade e ao custo de produção (REIS, 2015). Existem outras características para sistemas embarcados que, no entanto, não são consensuais, especialmente em função da evolução tecnológica e consequente redução nos custos de produção dos componentes envolvidos nesses sistemas (NOERGAARD, 2013).

Smartphones, embora possam apresentar características de computadores de propósito geral – o que fica evidente quando se contabiliza a variedade de aplicativos disponíveis para esses dispositivos – podem ser classificados como sistemas embarcados em função de suas restrições de projeto, como as já citadas referentes às dimensões físicas, ao consumo de

[2] Projeto de exploração espacial do governo norte-americano iniciado na década de 1960 que culminou com a chegada do homem à Lua.

energia, à durabilidade e ao custo. Essa classificação, como já citado anteriormente, no entanto, não é consensual.

2.1.2 Modelo de Sistemas Embarcados

O Modelo de Sistemas Embarcados, conforme descrito por Noergaard (2013), define três componentes: a Camada de *Hardware*, a Camada de *Software* de Sistema e a Camada de *Software* de Aplicação, ilustrados na Figura 2.1. A Camada de *Hardware* contém os componentes físicos que vão integrar o Sistema Embarcado. As camadas de *Software* de Sistema e de *Software* de Aplicação contêm todo o *software* que compõe o Sistema Embarcado e que será processado pelo mesmo.

Figura 2.1 – Modelo de Sistemas Embarcados

Fonte: Adaptado de Noergaard (2013, p. 10)

Enquanto a primeira camada é obrigatória, as demais nem sempre estão presentes em um sistema embarcado. Um exemplo de aplicação em que as camadas de *software* não estão presentes é um controlador de *videogame*. Nesse caso, embora possa haver alguma programação dentro do equipamento, esta não chega a ser um *software* completo conforme definido na literatura:

> *Software* consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas. (PRESSMAN, 2011, p. 32)

Ou ainda, conforme Sommerville (2011, p. 3): “não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessários para fazer esse programa operar corretamente. ”

2.1.3 Componentes de Sistemas Embarcados

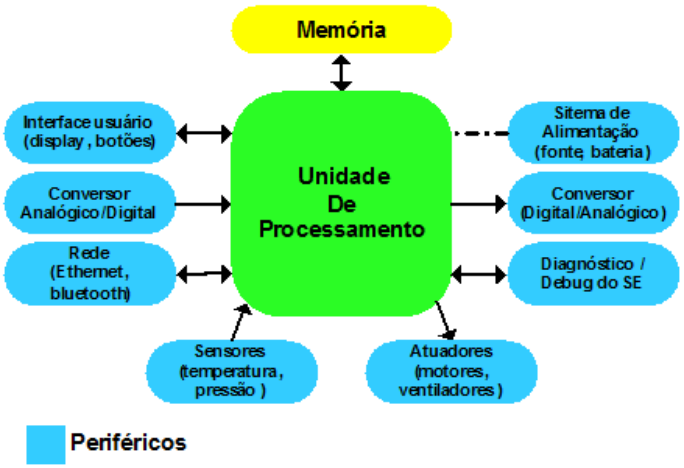
As camadas do Modelo de Sistemas Embarcados podem conter diversos componentes. Para cada camada, serão apresentados os componentes mais comumente encontrados.

2.1.3.1 Componentes da Camada de Hardware

São três os tipos de componentes mais comuns na camada de *hardware*: unidade de processamento, memória RAM e periféricos (DELAI, 2013).

A unidade de processamento é a responsável por processar as instruções do programa de um sistema embarcado. Essa unidade opera sobre a memória e sobre os periféricos, conforme ilustra a Figura 2.2, e pode possuir alguns blocos típicos de computadores de propósito geral, como memória *cache*, unidade lógica e aritmética, unidade de controle, registradores, *etc*.

Figura 2.2 – Componentes de um sistema embarcado



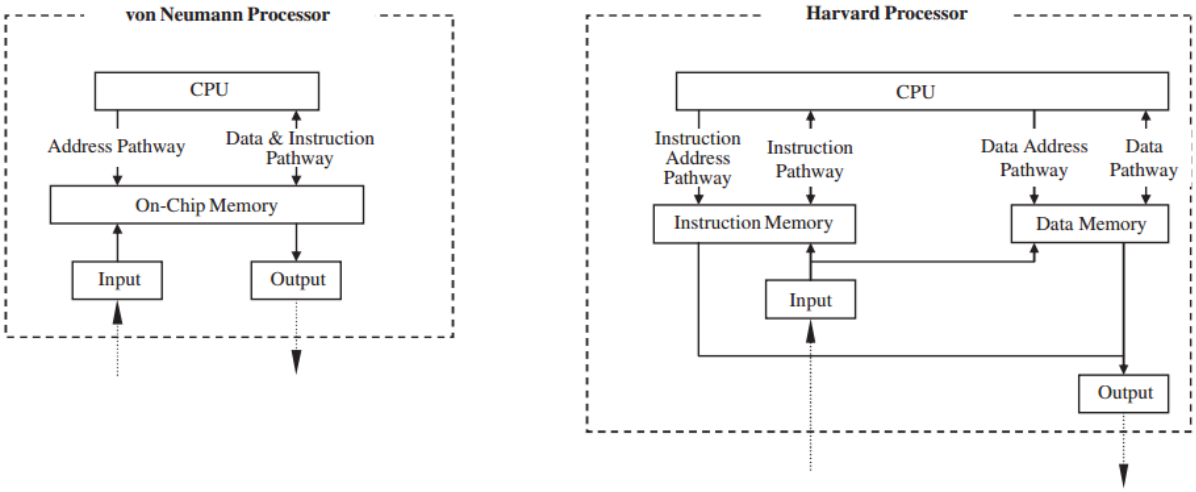
Fonte: Delai (2013)

A memória RAM é onde são armazenadas as instruções do programa e os dados sobre os quais o programa opera. Pode seguir as arquiteturas *Von Neumann* (dados e instruções dividindo a mesma memória) ou *Harvard* (dados e instruções em memórias separadas), esta

última sendo a mais comum em sistemas embarcados. A Figura 2.3 apresenta os diagramas de fluxos de dados para ambas as arquiteturas.

A preferência pela arquitetura *Harvard* é devida ao ganho de desempenho promovido pelo uso de dois barramentos, permitindo que operações e operandos sejam buscados na memória simultaneamente. Um exemplo muito comum que tira proveito dessa arquitetura são os processadores de sinais digitais (*digital signal processing*, ou DSP na sigla em inglês), que têm como característica a execução repetitiva de um trecho de código sobre um fluxo contínuo de dados.

Figura 2.3 – Arquiteturas *Harvard* e *von Neumann*



Fonte: Noergaard (2013, p. 146)

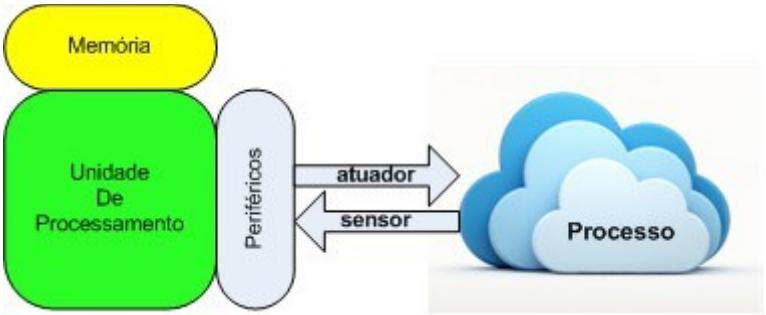
Os periféricos são constituídos por uma gama muito grande de dispositivos que permitem a interação da unidade de processamento com o mundo externo, permitindo inclusive a comunicação com outros sistemas embarcados. Nessa categoria destacam-se os sensores e atuadores, quase onipresentes em sistemas embarcados e responsáveis, respectivamente, por coletar informações do processo que está sendo controlado e por provocar alterações no mesmo.

Dois principais modelos de construção de sistemas embarcados destacam-se quanto à utilização de sensores e atuadores: modelo de malha fechada e modelo de malha aberta (DELAI, 2013).

Nos sistemas embarcados de malha fechada, são coletados dados do ambiente através dos sensores e, de acordo com esses dados e com parâmetros programados, são acionados os atuadores. Esses, por sua vez, provocam alterações no ambiente que são novamente percebidas pelos sensores, realimentando o processo em um *loop*. Pode-se tomar como

exemplo desse modelo um sistema de ar-condicionado, em que a temperatura ambiente é medida e, dependendo do valor medido e da temperatura programada, o compressor é acionado ou interrompido. A Figura 2.4 apresenta o diagrama de um sistema de malha fechada.

Figura 2.4 – Diagrama de um sistema de malha fechada



Fonte: Delai (2013)

Já nos sistemas de malha aberta não há sensores determinando quando ou como os atuadores devem operar. Nesse caso, os atuadores são acionados e interrompidos conforme uma programação prévia. Nesse caso, uma lavadora de louças é um exemplo adequado, pois nela é selecionada uma programação e, após isso, seu funcionamento independe de fatores externos.

2.1.3.2 Componentes da Camada de Software de Sistema

Integram essa camada os componentes que promovem uma abstração para a camada de *software* de aplicação. Os principais componentes dessa camada são os *drivers* de dispositivo, o sistema operacional e o *middleware*. Esses componentes definem subcamadas que podem estar arranjadas de diferentes maneiras.

Os *drivers* de dispositivo são os responsáveis por interagir diretamente com os diferentes componentes de *hardware* de um sistema embarcado. A Figura 2.5 apresenta as diversas formas em que a subcamada de *drivers* pode se apresentar.

O sistema operacional é o responsável por prover abstração entre a camada de aplicação e o hardware subjacente através de bibliotecas padronizadas e APIs e também por gerenciar os recursos de *hardware* e *software*. Assim como acontece com os *drivers*, a subcamada de sistema operacional (SO) pode apresentar diferentes combinações de interação com as demais, conforme ilustra a Figura 2.6.

Figura 2.5 – Camada de *Software* de Sistema - *Drivers*

Fonte: Adaptado de Noergaard (2013, p. 311)

Figura 2.6 – Camada de *Software* de Sistema – SO

Fonte: Adaptdo de Noergaard (2013, p. 383)

Assim como é encontrada uma gama enorme de processadores, também existem diversos SOs para sistemas embarcados, conforme a finalidade da aplicação. Por exemplo, há o Texas Instruments *Real-Time Operating System* (TI-RTOS) para sistemas embarcados de tempo real, o Android para *smartphones*, o Cisco *Internetwork Operating System* (IOS) para dispositivos de rede, entre muitos outros.

O último componente a integrar a camada de *software* de sistema é o *middleware*. Ele é definido como "*software* que foi abstraído da camada de aplicação" (NOERGAARD, 2013, p. 445). Tanenbaum (2003, p. 2) define *middleware* como uma camada de *software* sobre o SO que apresenta um conjunto de computadores como um único sistema coerente. No âmbito os sistemas embarcados, esta definição é muito restritiva, pois aplica-se exclusivamente a sistemas distribuídos, motivo pelo qual optou-se por utilizar a definição de Noergaard. Entre as principais razões para abstrair componentes da camada de *software* de aplicação figuram promover o reuso de código e simplificar o desenvolvimento das aplicações. Assim como as

demais subcamadas, pode combinar-se de diferentes formas, dependendo do projeto. A Figura 2.7 apresenta os possíveis arranjos dessa subcamada.

Um ótimo exemplo de *middleware* é a *Java Virtual Machine* (JVM), que não é parte da aplicação nem do SO, operando como intermediária entre esses dois.

Figura 2.7 – Camada de *Software* de Sistema – *Middleware*

Fonte: Adaptado de Noergaard (2013, p. 445)

2.1.3.3 Componentes da Camada de Software de Aplicação

É na camada de aplicação que estão os programas que implementarão as funções específicas do sistema embarcado. Nessa camada que o projetista concentrará a maior parte de seu esforço de desenvolvimento, pois é onde, seja interagindo com componentes da camada de *software* de sistema, seja interagindo diretamente com o *hardware*, será definido o comportamento do sistema através de interações com os usuários e com o meio. Existem inúmeros exemplos que podem ser apresentados para ilustrar a função dessa camada. Em complemento ao exemplo da camada de *middleware*, pode-se citar a implementação de protocolos da camada de aplicação do modelo TCP/IP, como um navegador *web* HTTP ou um cliente FTP.

2.1.4 Arquitetura de Sistemas Embarcados

A arquitetura de um sistema embarcado é definida como uma representação abstrata desse sistema, sem detalhar a sua implementação. É utilizada para representar o comportamento dos elementos e as suas interações. Esses elementos podem representar partes tanto do *hardware* como do *software* (NOERGAARD, 2013).

A Figura 2.2 é um exemplo de representação da arquitetura de um sistema embarcado, pois exibe componentes (processador, memória, *etc.*) e suas interações (barramentos, *etc.*) sem, no entanto, apresentar detalhes de como operam ou de como interagem entre si e com o ambiente.

Conforme Sommerville (2011), existem diversos padrões de arquiteturas definidos. Esses padrões, no entanto, não devem servir como modelos a serem reproduzidos, mas como pontos de partida para a definição de arquiteturas específicas.

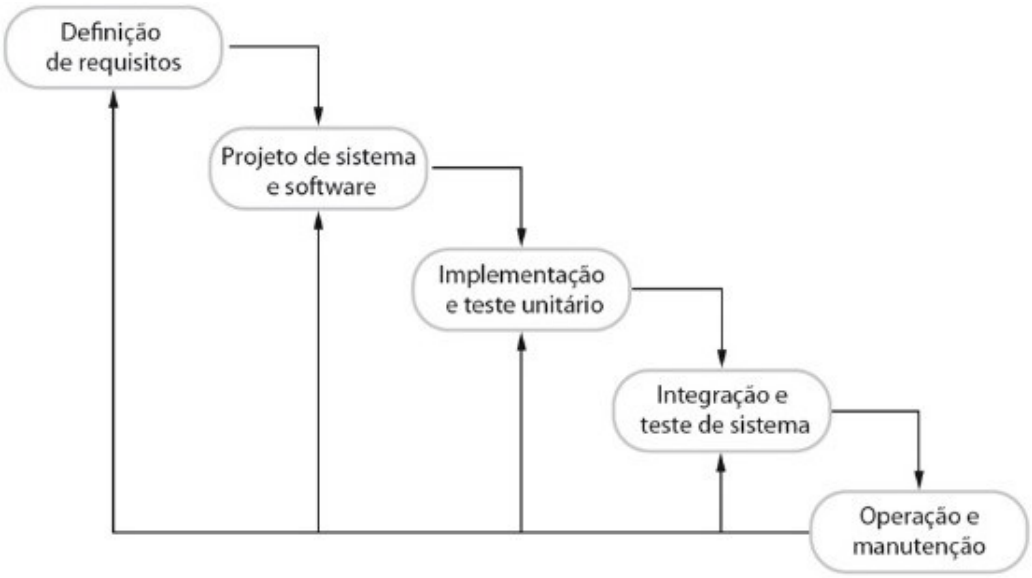
Cabe ressaltar aqui que o conceito de arquitetura de sistemas embarcados é utilizado em diferentes áreas da computação com diferentes significados. Por exemplo, Tutorials Point (2016) utiliza o termo para discorrer tanto sobre a arquitetura de memória (von Neumann x Harvard) quanto sobre a *Instruction Set Arrchitecture* (ISA). Neste trabalho, optou-se por utilizar a definição de Noergaard (2013), pois é mais abrangente e permite definir um processo genérico para a sua construção em diferentes níveis.

Para projetar a arquitetura de um sistema embarcado, Noergaard (2013) define um Modelo de Ciclo de Vida de *Design* e Desenvolvimento de Sistemas Embarcados. Esse modelo detalha as etapas envolvidas no projeto da arquitetura e as agrupa em quatro fases distintas: criação da arquitetura, implementação da arquitetura, teste do sistema e manutenção do sistema.

Percebe-se aqui uma forte inspiração no Modelo de Processo de *Software* em Cascata, divido em cinco etapas: análise e definição de requisitos, projeto de sistema e *software*, implementação e teste unitário, integração e teste de sistema e operação e manutenção (SOMMERVILLE, 2011). A Figura 2.8 torna evidente as semelhanças entre ambos os modelos. Pode-se notar, também, elementos do fluxo iterativo de processo de *software*, caracterizado pela realimentação entre etapas adjacentes.

Conforme Tammy Noergaard (2013), a fase de criação da arquitetura é a mais importante, pois guiará todo o resto do processo. Nessa fase devem ser definidos o *hardware* e o *software* a serem empregados e como os componentes serão integrados.

Figura 2.8 – Modelo de Processo de *Software* em Cascata



Fonte: Sommerville (2011, p. 20)

2.2 Grandezas Elétricas

Para a correta obtenção das informações pretendidas, faz-se necessário compreender alguns conceitos relacionados às grandezas elétricas. Serão apresentadas, a seguir, as grandezas que se pretende medir e algumas relações entre as mesmas.

2.2.1 Tensão Elétrica

A tensão elétrica, também chamada de "diferença de potencial elétrico" é definida como "uma indicação de quanta energia é envolvida na movimentação de uma carga elétrica entre dois pontos no espaço" (TEIXEIRA, 2015). A unidade de medida de tensão elétrica, de acordo com o Sistema Internacional de Unidades (SI), é o Volt (V), e corresponde a um Joule por Coulomb.

Nos ambientes residenciais, comerciais e industriais, a tensão obtida nas tomadas varia com o tempo, a uma taxa de sessenta vezes por segundo, ou 60 Hertz (Hz), invertendo seu sentido a cada ciclo. Por este motivo, é dita "alternada. "

2.2.2 Corrente Elétrica

A corrente elétrica é definida como “a taxa temporal com a qual cargas elétricas atravessam uma determinada superfície ” (TEIXEIRA, 2016). Pode ser descrita por sua média em um intervalo de tempo ou por seu valor instantâneo. Tem como unidade de medida, conforme o SI, o Ampère (A), que corresponde a um Coulomb por segundo.

Assim como ocorre com a tensão, nas instalações residenciais, comerciais e industriais a corrente também alterna de sentido a uma taxa de 60 Hz, motivo pelo qual também é denominada de “alternada. ”

2.2.3 Potência Elétrica

Potência elétrica é “a taxa temporal de gasto ou absorção de energia, medida em Watts (W) ” (TEIXEIRA, 2015). Um Watt corresponde a um Joule por segundo. Pode ser obtida calculando-se o produto entre a tensão e a corrente de um dado circuito.

2.2.3.1 Potências Ativa, Reativa e Aparente

As variações de sentido na tensão e na corrente apresentadas anteriormente fazem com que as representações gráficas das mesmas apresentem um aspecto senóide. Quando equipamentos elétricos dotados de capacitores e indutores são alimentados por corrente alternada, fazem com que haja uma defasagem entre as curvas de tensão e de corrente. Essa defasagem diminui a potência que realmente é convertida em trabalho, ocasionando desperdício de energia elétrica. A partir desta observação, foram definidos quatro conceitos (ELECTRICAL4U, 2016):

- Potência ativa: corresponde à fração da potência que é de fato convertida em trabalho;
- Potência reativa: corresponde à fração da potência que não é convertida em trabalho;
- Potência aparente: é a soma da potência ativa e da potência reativa;
- Fator de potência: é a razão entre a potência ativa e a potência aparente, e pode ser obtido calculando-se o cosseno da diferença de fase entre a corrente e a tensão.

2.2.4 Valor Eficaz

O conceito de valor eficaz diz respeito a grandezas que apresentam variação no tempo, como a tensão e a corrente alternadas. É a média quadrática de uma série de valores, e também pode ser referido como *root mean square* (RMS) (WEISSTEIN, 2016). Em sistemas elétricos com corrente alternada, "está relacionado com o calor dissipado em uma resistência" e é obtido, para ondas senoidais, dividindo-se o valor de pico pela raiz quadrada de dois (NAKASHIMA, 2013).

2.3 Resumo do Capítulo

Foram abordados, neste capítulo, os principais conceitos aplicados na concepção de Sistemas Embarcados. Primeiramente, apresentou-se a definição de Sistemas Embarcados, seguida do modelo utilizado para a representação desses sistemas e das camadas que o compõem. Também foram apresentados alguns dos componentes mais comuns de cada camada. Por fim, definiu-se o que é a arquitetura de um sistema embarcado.

Também foram apresentados conceitos básicos necessários para compreender a medição das grandezas elétricas necessárias.

Os elementos apresentados até agora servirão de base para o desenvolvimento do sensor tensão-corrente que se está propondo implementar.

3 PROJETO DE SENSOR TENSÃO-CORRENTE INTELIGENTE

Este capítulo apresenta a definição do problema que o presente trabalho se propõe a solucionar, os requisitos do sistema, a análise dos requisitos para a determinação da arquitetura do dispositivo e para a definição do comportamento e da apresentação do aplicativo para *smartphone* e os diagramas gerados a partir dessa análise. Em seguida, será feita uma breve apresentação de componentes e tecnologias que podem ser utilizadas para a construção do dispositivo.

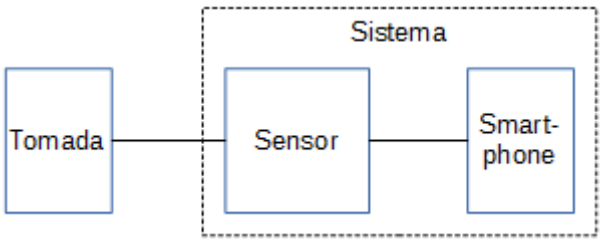
3.1 Problema Proposto

É fundamental para a apresentação de uma solução adequada a clara definição do problema em questão. A partir dessa definição é que serão extraídos os requisitos, os quais servirão de base para todo o projeto.

Posto isso, o problema é definido da seguinte maneira: deve-se criar um dispositivo para medir a corrente e a tensão de um aparelho ligado a uma tomada elétrica e apresentar essas informações de forma inteligível e simplificada em um *smartphone*, correlacionando-as de forma que o usuário possa compreender e comparar o perfil de consumo elétrico de diferentes dispositivos.

No nível mais alto de abstração, a solução para o problema envolve duas entidades, que posteriormente poderão ser subdivididas. São elas o sensor e o aplicativo para *smartphone*. Além disso, também será representado processo que se deseja controlar, que nesse caso consiste na tomada elétrica onde será ligado o sensor. A Figura 3.1 ilustra a arquitetura da solução nesse alto nível de abstração.

Figura 3.1 – Arquitetura da solução



Embora possa parecer trivial, existem diversas questões a serem respondidas antes de se definir o comportamento do sensor e do aplicativo, como a determinação de quais porções dos dados devem ser processadas pelo *smartphone* e quais devem ser processadas pelo

dispositivo, ou a frequência com que os dados devem ser amostrados. Para responder a essas questões, é necessário determinar quais são os requisitos do sistema.

3.2 Requisitos do Sistema

Apesar de não ser evidente, a compreensão dos requisitos de um sistema computacional é uma das tarefas mais complexas do processo de desenvolvimento (PRESSMAN, 2011). Para Sommerville (2011), existem dois níveis principais de abstração de requisitos: os requisitos de usuário, que são declarações em linguagem natural sobre como deve se comportar o sistema e requisitos de sistema, que são mais estruturados e detalhados. De acordo com essa divisão, pode-se considerar a proposição apresentada anteriormente nesse capítulo como os requisitos de usuário e as especificações apresentadas a seguir como requisitos de sistema.

Ainda segundo o autor, os requisitos costumam ser divididos em duas classes: requisitos funcionais e requisitos não funcionais. Os primeiros definem o comportamento do sistema, usualmente definindo as saídas em função das entradas. Já os últimos definem restrições para o sistema, entre as quais pode-se citar tempo de resposta, consumo energético e consumo de memória. Como dito anteriormente, sistemas embarcados são caracterizados por requisitos bastante restritivos quanto às características citadas, motivo pelo qual se deve dar atenção especial a essa classe de requisitos no presente projeto.

A seguir, serão listados os requisitos funcionais e não funcionais do sensor tensão-corrente, depreendidos a partir da proposição apresentada anteriormente neste capítulo.

3.2.1 Requisitos Funcionais

Os requisitos funcionais do sistema proposto são os seguintes:

- Deve ser possível medir a tensão instantânea e a corrente instantânea em uma tomada elétrica residencial;
- Devem ser exibidas na tela do *smartphone* as informações sobre a corrente atualmente medida em Ampères (A), sobre a tensão atualmente medida em Volts (V) e sobre o consumo acumulado desde o início da medição, em quilowatts-hora (kWh);
- Deve ser apresentado o gasto em Reais que a atual medição representa;
- Deve ser exibido um gráfico com o histórico recente da medição de potência real.

3.2.2 Requisitos Não Funcionais

Conforme citado anteriormente, são os requisitos não funcionais que caracterizam um sistema embarcado. Por esse motivo, os requisitos listados nesta seção pautarão a maioria das tomadas de decisão do projeto. Os requisitos não funcionais estão listados em ordem de prioridade, do mais prioritário ao menos prioritário:

- O dispositivo de medição deve ter um baixo custo de produção;
- O dispositivo de medição deve ter dimensões reduzidas, de forma a acoplar-se adequadamente a uma tomada residencial;
- A comunicação entre o dispositivo de medição e o *smartphone* deve ser sem fio, de baixo consumo energético e deve se dar de forma simplificada;
- Os sensores utilizados devem suportar os limites de tensão (250V) e corrente (20A) utilizados em tomadas domésticas (ABNT, 2012);
- A medição deve ocorrer em uma frequência adequada para que a informação seja útil;
- O aplicativo para *smartfone* deve ser agnóstico em relação à tecnologia dos sensores e do microcontrolador utilizados (princípio do encapsulamento).

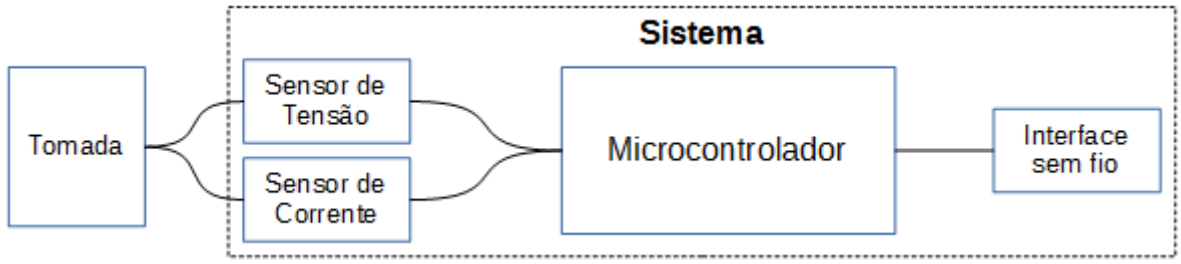
Como se pode perceber, a lista de requisitos não funcionais é mais extensa e mais específica que a lista de requisitos funcionais. Também fica evidente que esses requisitos focam muito mais na construção e programação do dispositivo do que no desenvolvimento do aplicativo para *smartphone*. Em parte, isso se deve ao fato de que há mais decisões a se tomar com relação ao dispositivo do que com relação ao aplicativo, principalmente no tocante à escolha dos componentes de *hardware* (microcontrolador, sensores e interface sem fio).

3.3 Arquitetura do Dispositivo

Antes de escolher os componentes que serão empregados na solução, é necessário determinar a arquitetura do dispositivo. Deve-se determinar os blocos básicos que serão necessários para realizar as tarefas definidas nos requisitos funcionais.

Primeiramente, serão necessários dois sensores, um para medir a tensão e outro para medir a corrente na tomada. Em seguida, essas medições deverão ser processadas por um microcontrolador, de forma que possam ser representadas em forma numérica. Por fim, essas informações deverão ser enviadas ao *smartphone* através de uma interface de comunicação sem fio. Essa arquitetura está ilustrada na Figura 3.2.

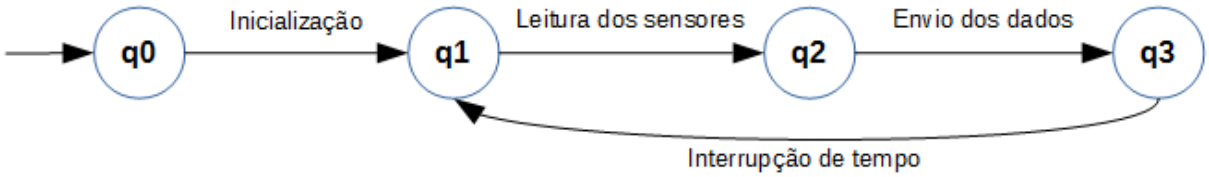
Figura 3.2 – Arquitetura do dispositivo



É importante ressaltar que, apesar da divisão apresentada nessa arquitetura, alguns itens podem ser integrados em um mesmo circuito, como ambos os sensores ou o microcontrolador e a interface sem fio, o que é bastante comum no caso de um *System-on-a-Chip*[3] (SoC).

Já o comportamento do dispositivo pode ser modelado utilizando-se uma máquina de estados finita. Essa representação ilustra o fluxo de execução do dispositivo em termos de eventos que podem ocorrer e alterar o estado da execução. O modelo utilizado para a representação é uma Máquina de Estados de Moore (BRITO, MARTENDAL e OLIVEIRA), representada na Figura 3.3.

Figura 3.3 – Máquina de estados de Moore do dispositivo



Nessa etapa do projeto, é preciso definir o balanceamento do processamento entre o dispositivo e o *smartphone*. Como existem requisitos de custo com relação ao dispositivo, é natural que esse realize o mínimo de processamento possível, de forma que possa ser utilizado um microcontrolador de mais baixo custo. Assim, o processamento mais pesado é realizado pelo aplicativo, pois esse disporá de muito mais memória e poder de processamento.

3.4 Projeto do Aplicativo

Para atender adequadamente aos requisitos, especialmente aos funcionais, faz-se necessário descrever de forma apropriada a apresentação e o comportamento do aplicativo

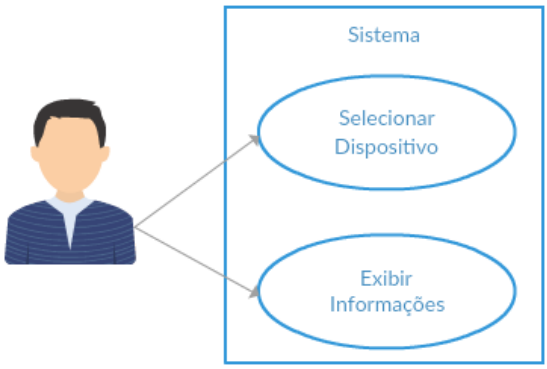
[3] Sistemas compostos por microcontrolador e diversos periféricos em uma única pastilha. Contêm todos os componentes eletrônicos para uma dada aplicação em um único circuito integrado (TECHTARGET, 2016).

para *smartphone*. Para isso, serão empregadas as ferramentas de engenharia de *software* denominadas diagramas casos de uso e *layout* de tela.

Diagramas de casos de uso constituem uma técnica que integra a Linguagem Unificada de Modelagem (UML, na sigla em inglês). Consistem em representações dos atores e suas interações com o sistema (SOMMERVILLE, 2011).

Esse aplicativo é bastante simples, com apenas uma tela e poucas interações possíveis com o usuário. Por isso, somente um diagrama de caso de uso foi definido e é apresentado na Figura 3.4.

Figura 3.4 – Diagrama de casos de uso



Um *layout* de tela é definido por Pressman (2011, p. 301) como:

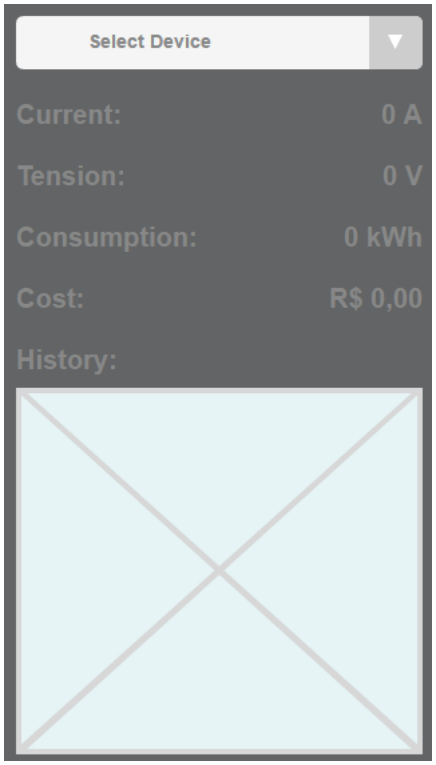
> um processo interativo no qual são realizados o projeto gráfico e o posicionamento dos ícones, a definição de texto de tela descritivo, a especificação e a colocação de títulos para as janelas, bem como a definição de itens de menu principais e secundários.

O *layout* da tela do aplicativo deve possuir os seguintes elementos:

- Caixa de seleção para escolha do sensor;
- Informação de corrente em Ampères;
- Informação de tensão em Volts;
- Informação de consumo em quilowatts-hora;
- Informação de custo em Reais;
- Gráfico com histórico de potência em Watts.

A representação gráfica do *layout* descrito acima é apresentada Figura 3.5.

Figura 3.5 – *Layout* de tela



3.5 Tecnologias e Componentes Utilizados

Para desempenhar a função de cada item definido na arquitetura do dispositivo (Figura 3.2), existem diversos componentes disponíveis no mercado. Serão apresentadas, a seguir, as principais opções de tecnologia e implementação para esses itens.

3.5.1 Sensor de Corrente

Foram encontrados no mercado duas principais linhas de sensores de corrente elétrica. A primeira linha consiste em sensores não invasivos [4] fabricados pela YHDC e possui modelos com faixas de corrente de entrada limitadas entre 0A e 5A, 0A e 10A, 0A e 15A, 0A e 20A, e 0A e 100A, custando R$ 54,90. A outra linha são os sensores ACS712, fabricados pela Allegro. Esses são sensores invasivos e operam com as seguintes faixas de corrente: -5A a +5A, -20A a +20A e -30A a +30A. Apresentam custo de R$ 26,90.

[4] Sensores não invasivos são sensores que não necessitam que o circuito seja aberto para efetuar a medição.

3.5.2 Sensor de Tensão

Apenas um sensor de tensão adequado para o projeto foi encontrado no mercado. Ele é denominado P8 e fabricado pela empresa GBK Robotics. Não foi encontrada documentação oficial sobre o funcionamento desse sensor, de forma que os parâmetros de operação tiveram de ser obtidos de forma experimental. Seu custo no mercado é de R$ 16,89.

3.5.3 Microcontrolador

Sem dúvida o componente com maior abundância de opções. São muitas as variáveis que devem ser levadas em consideração para determinar o microcontrolador que será utilizado em um projeto, desde sua capacidade de processamento e de memória até a disponibilidade de documentação e suporte.

Tendo em vista que se focou no custo e na potência dissipada, foram encontradas diversas opções que atendem às necessidades do presente projeto. Entre os microcontroladores mais utilizados no mercado para esse tipo de aplicação, encontram-se os da família PIC, da fabricante Microchip, e da família ATmega, da fabricante Atmel.

Foram selecionados dois microcontroladores que atendem às necessidades do projeto, um de cada fabricante, para que seja feito um comparativo entre suas características e se possa embasar a opção por um ou por outro. Essa comparação está ilustrada na tabela Tabela 3.1.

Tabela 3.1 – Comparativo microcontroladores

Microcontrolador (frequência)	*Potência dissipada*	*Preço*
PIC24F32KA301-E/P (8MHz)	17,5 mW	U$ 4,04
ATMEGA328P/PU (8MHz)	26 mW	U$ 3,70

3.5.4 Interface *Wireless*

Entre as tecnologias sem fio pesquisadas, duas são praticamente onipresentes em *smartphones*: Wi-Fi e *Bluetooth*. As principais características levadas em consideração para avaliar os componentes foram o seu custo e a complexidade da conexão com o *smartphone*. Para a comunicação Wi-Fi, encontra-se no mercado a família de módulos ESP-8266. Para a comunicação *Bluetooth*, foi encontrado módulo HC-05. A Tabela 3.2 apresenta um comparativo entre os critérios avaliados de cada um.

Tabela 3.2 – Comparativo de interface de comunicação sem fio

Módulo	*Potência dissipada*	*Preço*
ESP-8266 ESP-01	709,5 mW	R$ 29,90
HC-05	115,5 mW	R$ 39,90

3.5.5 Melhores Componentes Encontrados

O sensor de corrente que apresentou as melhores características foi o ACS712 de -20A a +20A porque, além de ser menor e compreender os limites de corrente encontrados em tomadas domésticas, seu preço é muito inferior ao do YHDC. O motivo de não utilizar o sensor com capacidade de 30A é que, ao realizar a quantização, perde-se precisão comparativamente aos sensores com faixas de operação mais restritas, pois, para todos os modelos dessa linha, a faixa de leitura é convertida em uma tensão entre 0V e 5V.

Quanto ao sensor de tensão, não foram encontradas alternativas no mercado, restando como única opção o P8.

Com relação aos microcontroladores, há uma semelhança muito grande entre os dois apresentados. O microcontrolador da família PIC apresenta uma leve vantagem na potência dissipada, enquanto o da Atmega apresenta uma pequena vantagem no custo. Como tanto o preço do controlador quanto a potência dissipada têm pouco impacto no restante do projeto, qualquer um dos microcontroladores pode ser utilizado.

Por fim, para decidir qual tecnologia sem fio utilizar, deve-se observar que a interface *Bluetooth* apresente um consumo energético mais de seis vezes menor que o Wi-Fi. Dado que a diferença de custo entre os dois componentes é relativamente pequena e que a complexidade para estabelecer uma conexão entre o *smartphone* e o dispositivo é semelhante em ambos os casos, o *Bluetooth* surge como a melhor opção.

3.5.6 Custo Total Considerando os Componentes Selecionados

O custo total do dispositivo, considerando-se apenas os custos dos componentes de *hardware*, é de R$ 96,34, como demonstra a Tabela 3.3.

Tabela 3.3 – Custo total do *hardware*

Componente	*Preço*
Sensor de corrente	R$ 26,90
Sensor de tensão	R$ 16,89
Microcontrolador	R$ 12,65*
Interface Wireless	R$ 39,90
Total	R$ 96,34

*Conforme cotação do dólar comercial em 16/11/2016

Deve-se ponderar, entretanto, que esses valores são para compras avulsas, e a tendência é de que os preços sejam menores se os produtos forem adquiridos em grandes quantidades. Além disso, alguns componentes podem ser construídos a partir de partes mais básicas, ao invés de serem adquiridos prontos, o que deve reduzir ainda mais o custo de produção do dispositivo.

3.6 Resumo do Capítulo

Neste capítulo, foi apresentada uma melhor definição do problema. Dessa definição foram extraídos os requisitos funcionais e não funcionais. Em seguida, foram definidas as arquiteturas do sistema e do dispositivo. Também foram listados alguns componentes e tecnologias disponíveis no mercado para a construção do dispositivo. Por fim, foram apresentados os componentes que melhor atendiam aos requisitos.

4 PROTOTIPAÇÃO E DESENVOLVIMENTO

Definidos as arquiteturas do sistema e do dispositivo e o comportamento do aplicativo, é necessário construir um protótipo funcional do dispositivo e desenvolver o aplicativo para *smartphone*. Por se tratarem de duas atividades bastante distintas e independentes, essas tarefas serão realizadas de forma independente. Isso é possível se for definida uma interface de comunicação padronizada, de acordo com o que pregam as boas práticas de desenvolvimento de *software*, promovendo o encapsulamento, como define Pressman (2011, p. 215-216): "O princípio de encapsulamento de informações sugere que os módulos sejam 'caracterizados por decisões de projeto que ocultem (cada uma delas) as demais. '"

4.1 Prototipação e Programação do Dispositivo

O primeiro passo para a construção do dispositivo é a prototipação do mesmo. Após o protótipo construído, deve-se realizar a programação do dispositivo, levando em consideração os requisitos definidos no capítulo anterior.

4.1.1 Prototipação do Dispositivo

Para a prototipação do dispositivo foi utilizada a plataforma Arduino Uno. Essa plataforma consiste em uma placa de circuito impresso com um microcontrolador ATmega328P-PU, um oscilador de 16MHz, uma controladora USB e circuitos para retificação de tensão, além de pinos para conexão de alimentação e I/O. O protótipo foi montado com o auxílio de uma *protoboard* com 830 pinos. A Figura 4.1 apresenta o esquema de conexões do protótipo. A montagem final é apresentada nas

O microcontrolador empregado no Arduino Uno é um dos dois apresentados no capítulo anterior. Isso facilita a migração do protótipo funcional para uma placa de circuito impresso, pois basta gravar a programação no Arduino, remover o microcontrolador do Arduino e fixá-lo à uma placa. A Figura 4.2 mostra o protótipo com o microcontrolador diretamente na *protoboard*. Nessa montagem, é necessário configurar o microcontrolador para que ele utilize seu oscilador interno de 8 MHz.

Como o Arduino opera com nível lógico de 5V e o módulo *Bluetooth* opera com 3,3V, foi necessário utilizar um divisor de tensão resistivo, aplicando-se uma resistência *R* entre o

Arduino e o módulo e uma resistência *2R* entre o Arduino e o *ground*. O valor de *R* utilizado foi de 10kΩ.

Figura 4.1 – Esquema de conexões do protótipo funcional

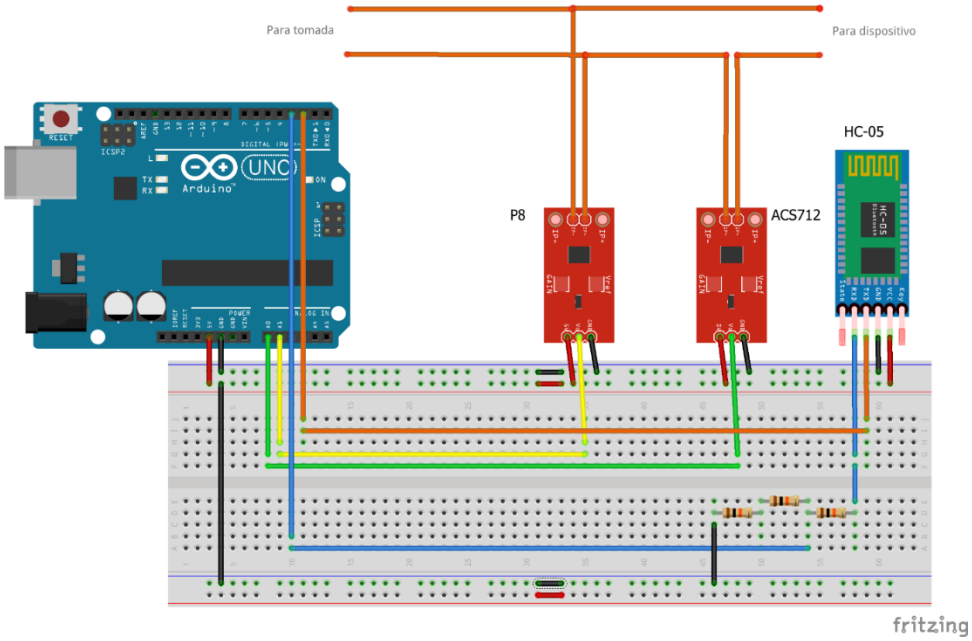
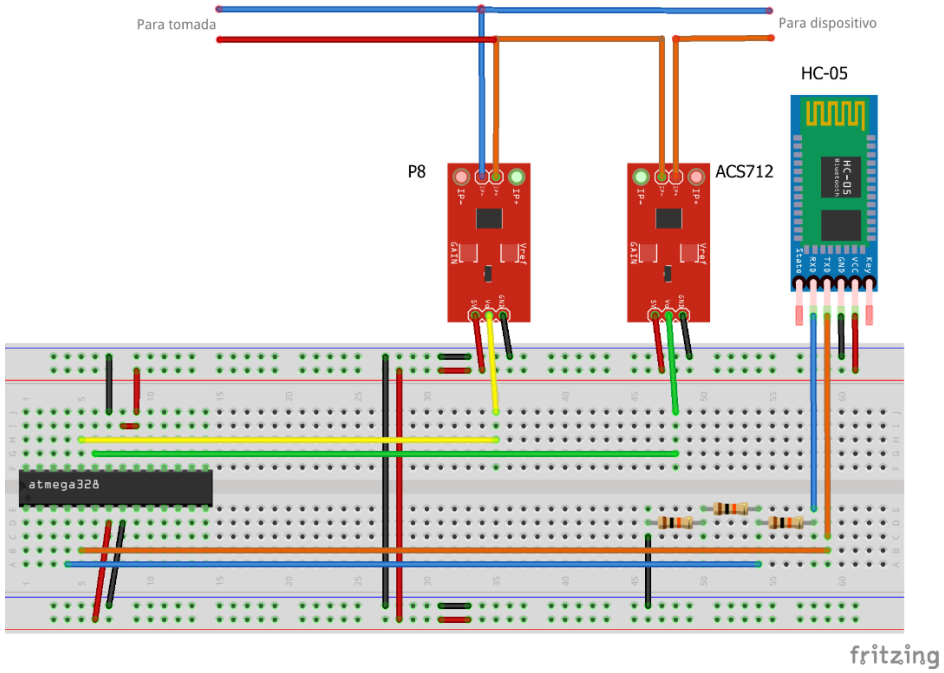
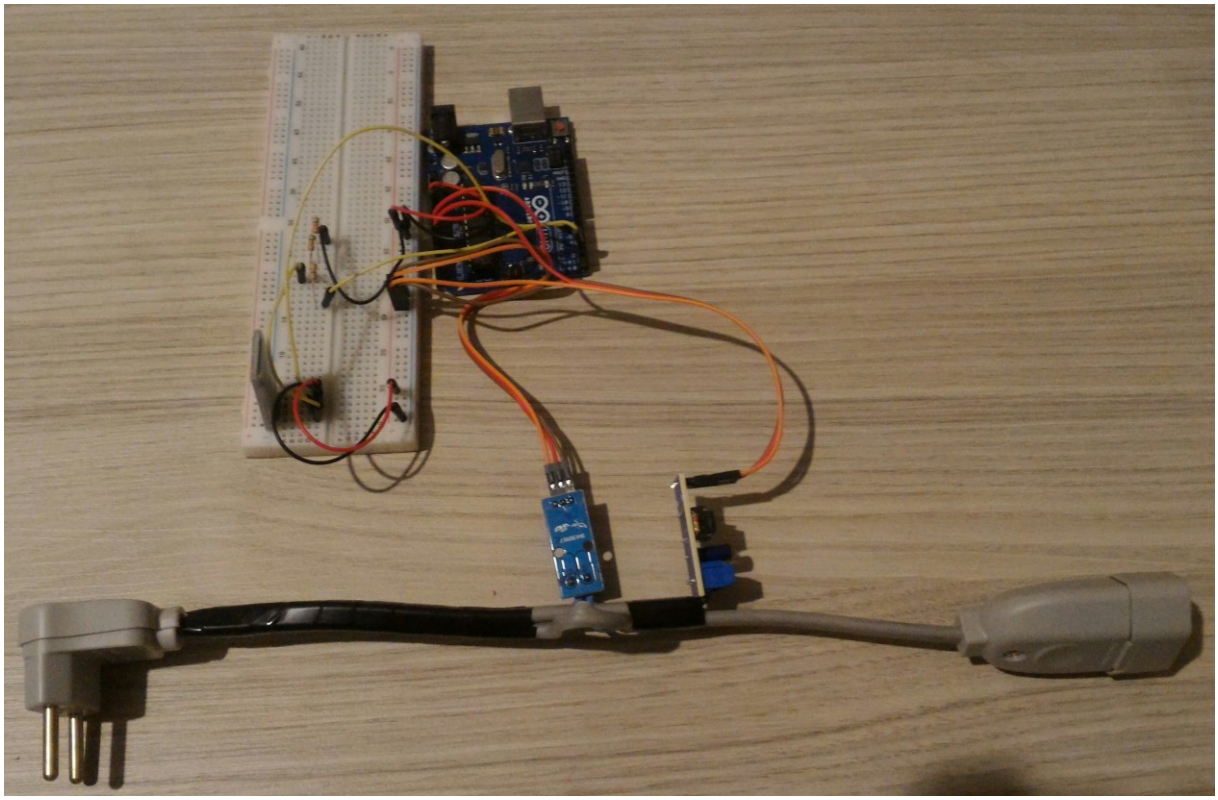


Figura 4.2 – Esquema de conexões do protótipo com microcontrolador



A partir do esquema apresentado, o protótipo foi montado e é apresentado na Figura 4.3.

Figura 4.3 – Montagem do protótipo com os componentes selecionados



4.1.2 Programação do Dispositivo

Com relação à programação do dispositivo, foi necessário levar em consideração a definição de que o mesmo realizará o menor processamento necessário, para que seja possível utilizar componentes com menor poder de processamento e, consequentemente, menor custo e menor consumo de energia. Por isso, foram definidas as seguintes diretrizes:

- Deverá ser enviada, a cada leitura dos sensores, a menor quantidade de informações necessárias para o processamento por parte do aplicativo;
- A eficiência do programa tem prioridade sobre a clareza de código e as boas práticas de programação.

A partir dessas diretrizes, determinou-se o seguinte:

- As leituras dos sensores serão apenas convertidas nas unidades de medida preestabelecidas e enviadas para o *smartphone*. Embora essa conversão pudesse ser feita no aplicativo, isso é necessário para atender ao requisito de que o aplicativo deve ser agnóstico em relação à tecnologia do *hardware* utilizado no dispositivo;

- Cada mensagem enviada pelo dispositivo conterá a tensão em Volts, a corrente em Ampères e um *timestamp*;
- Será utilizado o temporizador interno do microcontrolador para acionar a rotina de leitura dos sensores a uma frequência determinada;
- Será criada uma única rotina que fará a leitura dos sensores, a conversão para as unidades de medida determinadas e o envio para a interface *wireless*.

No presente projeto, não há camada de *software* de sistema. Todo o *software* que roda no dispositivo corresponde à camada de aplicação. O código do sensor está listado no Apêndice – LISTAGEM DE CÓDIGO DO DISPOSITIVO.

4.2 Desenvolvimento do Aplicativo para *Smartphone*

Enquanto na construção do sensor a maior parte do tempo foi despendida na determinação de seus componentes, o maior esforço no desenvolvimento do aplicativo para *smartphone* envolveu o estudo do *framework* do Android, de seus elementos e da comunicação através de *Bluetooth*.

Ainda seguindo as especificações dos requisitos listados anteriormente, o processamento necessário para apresentar as informações úteis foi todo realizado no aplicativo. Para realizar os cálculos necessários, as seguintes características foram levadas em consideração:

- A corrente informada pelo sensor é a instantânea e, por isso, cada leitura podia apresentar módulo e sinal diferentes;
- A tensão informada pelo sensor é RMS;
- A taxa de amostragem utilizada foi de 125 Hz (uma amostra a cada oito milissegundos);
- A cada período da amostra eram enviados a corrente, a tensão e um *timestamp*.

Com base nisso, é possível obter as seguintes informações através do processamento dos dados obtidos pelo aplicativo:

- A forma de onda original da corrente (60 Hz) – o Teorema de Nyquist estabelece que, para recuperar um sinal de frequência H, é necessário amostrá-lo com uma frequência maior ou igual a $2H$ (TANENBAUM, 2003).
- O valor de corrente RMS;
- O valor de tensão médio;

- O consumo acumulado em kWh;
- O custo total do consumo do aparelho desde o início da medição atual, em Reais;
- A potência aparente – sem a forma de onda da tensão, não é possível calcular o fator de potência e, consequentemente, a potência ativa (ELECTRICAL4U, 2016).

Com relação à interface do aplicativo, são apresentadas a tela inicial do mesmo e a listagem dos dispositivos *Bluetooth* pareados (Figura 4.4), além de gráficos com as medições da potência aparente, do momento em que um ferro elétrico é ligado e do momento em que o mesmo é desligado (Figura 4.5).

Diversas iterações foram realizadas no processo de desenvolvimento. As principais visavam a atingir os seguintes objetivos:

- Inicializar o adaptador *Bluetooth* do *smartphone* e listar os dispositivos pareados;
- Estabelecer uma conexão com o sensor;
- Receber os dados do sensor em uma *thread* secundária (para evitar que a interface do usuário trave entre um recebimento e outro);
- Otimizar o recebimento dos dados para suportar uma frequência de amostragem mais alta;
- Atualizar os valores nos elementos da interface do usuário;
- Implementar um gráfico para exibir o histórico recente de algumas informações.

Figura 4.4 – Tela inicial e lista de dispositivos pareados

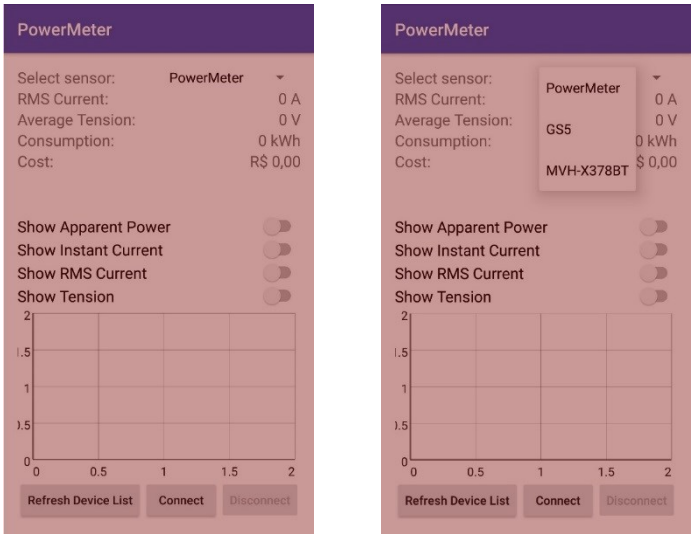
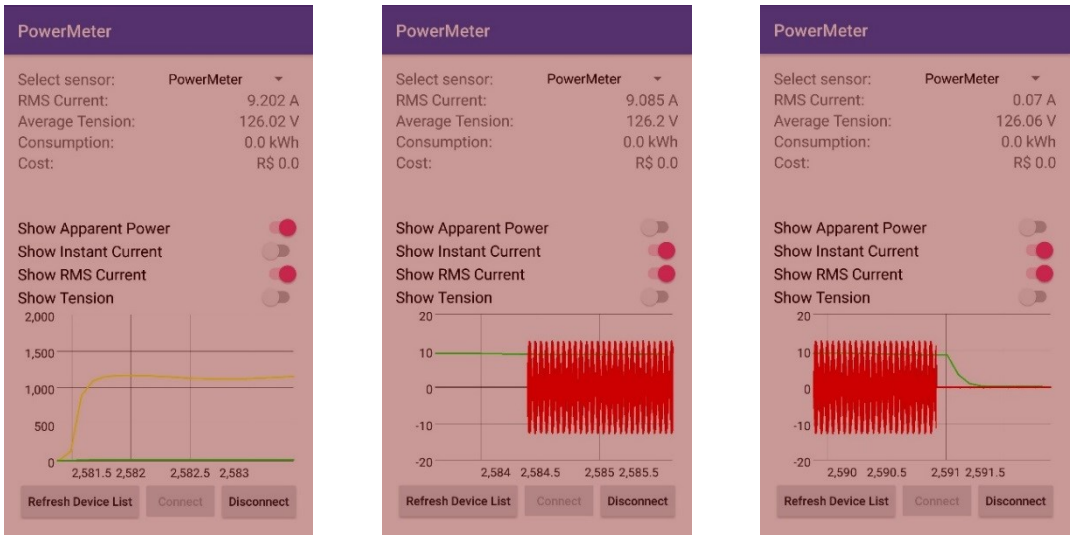


Figura 4.5 – Gráficos de potência aparente (amarelo), de corrente instantânea (vermelho) e de corrente RMS (verde)



Esse fluxo iterativo, típico do processo de *software* incremental, permitiu que o desenvolvimento sempre progredisse a partir de algo pronto, de forma que ao final de uma iteração todos os recursos desenvolvidos nas iterações anteriores estivessem em pleno funcionamento (SOMMERVILLE, 2011).

Além disso, a cada iteração, algumas peculiaridades surgiam, de forma que os incrementos seguintes iam sendo adaptados para acomodar essas peculiaridades. Como exemplo, pode-se citar o problema de eficiência encontrado ao receber os dados do sensor, que requereu uma iteração somente para otimizar o processamento dos dados recebidos. Outro

exemplo foi o acréscimo de novos elementos à interface do usuário, cuja necessidade somente foi percebida ao término de uma das iterações.

Por fim, a rapidez com que se obtém um *software* já com alguma funcionalidade aumenta a satisfação no processo de desenvolvimento, motivando a conclusão de um incremento e o início do próximo.

4.3 Recursos de Desenvolvimento Empregados

4.3.1 Programação do Dispositivo

Para a programação do dispositivo foi utilizada a IDE do Arduino, na versão 1.6.11. A programação neste ambiente utiliza a linguagem C. Foram empregadas, também, duas bibliotecas, a SoftwareSerial, que permite a comunicação com dispositivos utilizando o formato de dados do padrão RS232 (SILVEIRA, 2016); e a TimerOne, que permite a utilização do temporizador interno do microcontrolador ATmega328 para chamadas periódicas de rotinas.

4.3.2 Desenvolvimento do Aplicativo

O ambiente de desenvolvimento utilizado para a construção do aplicativo para *smartphone* foi o Android Studio 2.2. Esse aplicativo foi totalmente desenvolvido utilizando a linguagem Java. Também foi utilizada a biblioteca para geração de gráficos GraphView.

4.4 Resumo do Capítulo

Este capítulo apresentou os passos do desenvolvimento de um protótipo do sensor de tensão e corrente e de sua programação, trazendo alguns dos recursos utilizados e os diagramas das conexões realizadas durante o processo de montagem. Também foi abordado o processo de desenvolvimento do aplicativo para *smartphone*, desde aspectos de interface de usuário até o fluxo de desenvolvimento utilizado.

5 AVALIAÇÃO DO PROJETO

Diversos aspectos em um projeto de sistemas computacionais devem ser levados em consideração durante a sua avaliação. Com relação a sistemas embarcados, é importante que os requisitos possam ser medidos, caso contrário a avaliação quanto ao atendimento às especificações fica inviabilizada (KOOPMAN, 2010).

A seguir, são apresentados o ambiente de testes, os critérios de qualidade avaliados, as metodologias utilizadas e os resultados obtidos. Também serão discutidos os atendimentos aos requisitos funcionais e não funcionais.

5.1 Ambiente de Testes

Para a realização dos testes, foram utilizados os seguintes recursos:

- *Smartphone* Asus Zenfone 2 com processador Intel Atom Z3580 *quad-core* de 2,33 GHz, 4 GB de memória RAM e sistema operacional Android versão 6.0.1;
- *Tablet* Samsung Galaxy Tab 3 com processador ARM Cortex-A9 *dual-core* de 1,5 GHz, 1,5 GB de memória RAM e sistema operacional Android versão 4.4.2;

5.2 Critérios Avaliados

Como a solução apresentada é constituída de duas partes bem distintas, em alguns casos os critérios de avaliação poderão ser considerados distintamente entre essas partes. Em outras situações, há critérios que se aplicarão apenas a uma parte do projeto.

5.2.1 Tempo

Apesar de os sistemas computacionais operarem a altas taxas de frequência e possuírem grande capacidade de processamento, os requisitos temporais sempre são motivo de preocupação. Isso porque, apesar de velozes, esses sistemas precisam lidar com um grande volume de dados e, em muitos casos, compartilhar recursos entre diversas tarefas.

Na aplicação proposta, são dois os pontos onde o desempenho temporal será avaliado: na leitura e transmissão da tensão e da corrente por parte do sensor e no processamento e exibição desses dados no aplicativo.

5.2.2 Eficiência de Recursos

A eficiência está associada à boa utilização dos recursos físicos e do tempo que uma aplicação leva para apresentar os resultados desejados (PRESSMAN, 2011). No presente projeto, essa avaliação será focada no aplicativo para *smartphone*, pois os recursos do dispositivo estão todos disponíveis para o sensoriamento.

5.2.3 Eficiência Energética

Por estar calcado na questão da eficiência energética, é imperativo que esse trabalho foque na avaliação do consumo de energia pelo sensor e pelo aplicativo. O primeiro porque não é desejável aumentar significativamente o consumo de energia apenas para realizar a medição do consumo em si, e o segundo porque *smartphones* são dispositivos com capacidade de alimentação limitada, de forma que um aplicativo ineficiente diminuiria a duração da bateria e consequentemente afastaria o interesse dos usuários.

5.3 Resultados dos Testes

Nesta seção serão apresentados os resultados dos testes propostos na seção anterior e será feita uma breve análise desses resultados.

5.3.1 Tempo

Para o dispositivo de sensoriamento, foi avaliada a capacidade de ler e enviar os dados dos sensores a uma frequência adequada. A frequência determinada foi de 125 Hz, para que fosse possível recuperar a forma de onda da corrente. Assim, o microcontrolador foi programado para acionar a rotina de leitura dos sensores e envio dos dados a cada oito milissegundos.

Utilizando-se um console de terminal em um PC conectado ao dispositivo via *Bluetooth*, verificou-se que o processo de leitura e envio dos dados ocorreu na frequência determinada. Portanto, o resultado do teste de temporalidade no sensor foi satisfatório.

Com relação ao aplicativo, tanto no Asus Zenfone 2 quanto no Samsung Galaxy Tab 3 o desempenho apresentado foi satisfatório. Em ambos os casos a interface foi responsiva, sem apresentar travamentos, mesmo com as quatro séries de dados sendo apresentadas

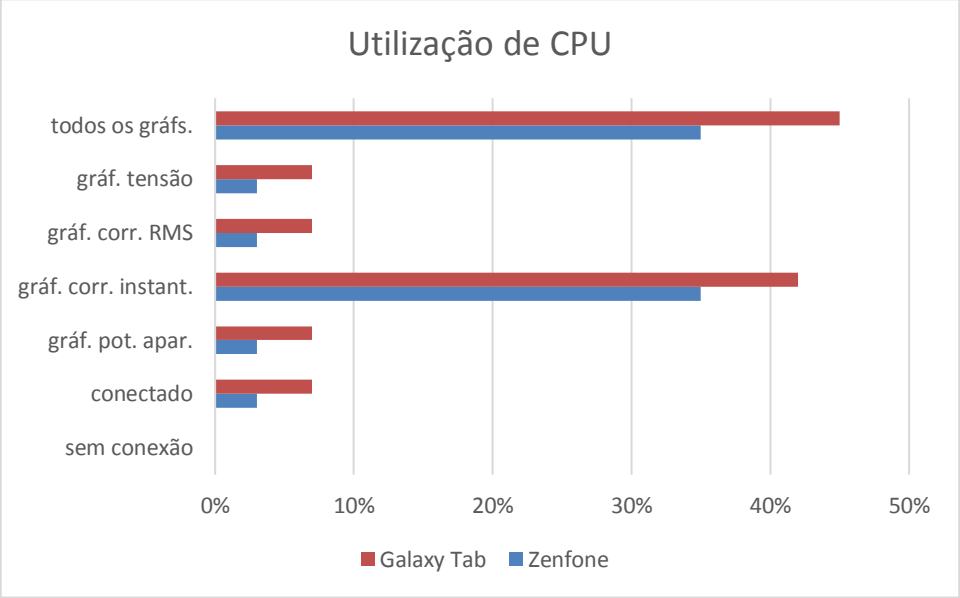
simultaneamente no gráfico. O processamento dos dados recebidos também foi satisfatório, sem que houvesse perdas devidas à alta frequência de transmissão dos dados.

5.3.2 Eficiência de Recursos

Quanto à utilização de memória, o limite padrão do *heap*[5] no *runtime* dos dispositivos avaliados, de aproximadamente 25 MB, não impediu o funcionamento do aplicativo, mostrando-se suficiente para a execução do aplicativo.

A medição da utilização de CPU foi realizada em sete situação distintas: sem conexão com nenhum dispositivo; conectado, mas sem exibir nenhuma informação no gráfico; exibindo cada uma das séries de dados no gráfico e exibindo todas as sérias de dados no gráfico. O perfil de utilização de CPU é apresentado na Figura 5.1.

Figura 5.1 – Gráfico de utlização de CPU



Os dados apresentados foram obtidos através do Android Monitor no Android Studio, recurso que mede a utilização de CPU do processo que está sendo monitorado.

Pode-se perceber nos casos apresentados que, em ambos os dispositivos, é a exibição do gráfico de corrente instantânea que provoca o maior incremento na utilização de CPU. Isso ocorre porque esse item é atualizado a cada leitura realizada (uma vez a cada oito milissegundos), enquanto os demais itens são atualizados a cada cinquenta leituras realizadas

[5] Área de memória de um processo destinada a manter as estruturas de dados de alocação dinâmica utilizadas durante o tempo de vida desse processo.

(400 milissegundos). Esse valor foi obtido experimentalmente e é necessário para que variações bruscas na leitura sejam suavizadas.

5.3.3 Eficiência Energética

Para medir o consumo de energia provocado pelo aplicativo, foi utilizado o aplicativo PowerTutor (UNIVERSITY OF MICHIGAN, 2011). A Figura 5.2 apresenta o gráfico de consumo energético no Zenfone 2 e a Figura 5.3 apresenta o mesmo gráfico para o Galaxy Tab 3.

Figura 5.2 – Consumo energético do aplicativo no Zenfone 2

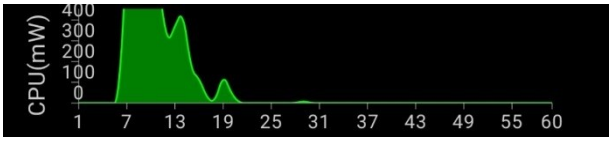
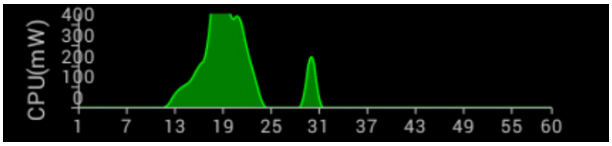


Figura 5.3 – Consumo energético do aplicativo no Galaxy Tab 3



Em ambos os casos, o perfil de consumo foi semelhante. No Galaxy Tab 3, no entanto, há um pico inicial maior. Embora não se possa afirmar com certeza, esse pico parece estar associado à inicialização dos aplicativos de um modo geral. Nota-se que o gráfico do Zenfone permanece por mais tempo no nível máximo. Isso é devido ao fato de que o aplicativo permaneceu mais tempo em execução nesse caso.

Com relação ao consumo dos periféricos, especialmente da interface *Bluetooth*, o aplicativo utilizado não mede essa informação.

Para o dispositivo, a avaliação quanto ao consumo energético foi feita baseando-se nas características nominais dos seus componentes ativos (microcontrolador e adaptador *Bluetooth*). Dessa forma, o consumo considerado é o somatório das potências dissipadas pelo microcontrolador (17,5 mW) e pela interface *Bluetooth* (115,5 mW), que totalizam 133 mW. Percebe-se que o consumo do dispositivo é bem inferior ao do aplicativo.

5.4 Atendimento aos Requisitos Funcionais

Será apresentada uma breve avaliação quanto ao atendimento aos requisitos funcionais, acompanhadas de comentários e de uma análise ao final da seção.

- Deve ser possível medir a tensão instantânea e a corrente instantânea em uma tomada elétrica residencial: atendido parcialmente, pois não foi possível medir a tensão instantânea devido à falta de sensores adequados no mercado.
- Devem ser exibidas na tela do *smartphone* as informações sobre a corrente atualmente medida em Ampères, sobre a tensão atualmente medida em Volts e sobre o consumo acumulado desde o início da medição, em quilowatts-hora (kWh): totalmente atendido, considerando-se os valores RMS dessas grandezas;
- Deve ser apresentado o gasto em Reais que a atual medição representa: totalmente atendido. No entanto, o preço do kWh foi definido dentro do código do aplicativo, sem a possibilidade de alteração desse dado por parte do usuário.
- Deve ser exibido um gráfico com o histórico recente da medição de potência real: parcialmente atendido, pois não foi possível obter a tensão instantânea, necessária para o cálculo do fator de potência, apenas a tensão RMS.

Embora o projeto não tenha atendido totalmente a alguns dos requisitos funcionais, pode-se considerá-lo bem-sucedido como prova de conceito, uma vez que a substituição de um sensor por uma versão mais apropriada seja suficiente para adequar o projeto à especificação.

5.5 Atendimento aos Requisitos Não Funcionais

A seguir, são apresentadas as avaliações quanto ao atendimento aos requisitos não funcionais, comentários sobre essas avaliações e uma breve análise.

- O dispositivo de medição deve ter um baixo custo de produção: atendido. Com valor abaixo de R$ 100,00, foi possível construir um sensor capaz de avaliar duas grandezas e de realizar um processamento relativamente complexo sobre esses dados;
- O dispositivo de medição deve ter dimensões reduzidas, de forma a acoplar-se adequadamente a uma tomada residencial: atendido. O tamanho reduzido dos componentes utilizados já permite a construção de um dispositivo compacto, e no

caso de produção em série, pode-se construir todos os componentes em uma única placa de circuito impresso, reduzindo ainda mais as suas dimensões;

- A comunicação entre o dispositivo de medição e o *smartphone* deve ser sem fio, de baixo consumo energético e deve se dar de forma simplificada: atendido. A utilização do padrão *Bluetooth* permitiu a conectividade simplificada, além de um consumo energético muito baixo se comparado com a alternativa do Wi-Fi;
- Os sensores utilizados devem suportar os limites de tensão e corrente utilizados em tomadas domésticas (250V e 20A): atendido. Por haver normas que determinam os limites de operação das tomadas, os fabricantes têm parâmetros para construir seus componentes de forma que sejam compatíveis com o que se encontra nas instalações domésticas;
- A medição deve ocorrer em uma frequência adequada para que a informação seja útil: atendido. Por deixar o processamento pesado para o *smartphone*, a programação do microcontrolador ficou enxuta, viabilizando a execução da rotina de leitura e transmissão dos dados em pouco tempo;
- O aplicativo para *smartfone* deve ser agnóstico em relação à tecnologia dos sensores e do microcontrolador utilizados (princípio do encapsulamento): atendido. O desempenho do microcontrolador escolhido é suficiente para realizar a conversão da leitura de um nível de tensão (saída do sensor) em um valor padronizado dentro do período determinado para a execução da rotina (oito milissegundos).

A construção do dispositivo, quanto aos requisitos não funcionais, foi bastante satisfatória. A etapa de seleção dos componentes foi fundamental para esse sucesso, especialmente a seleção do microcontrolador utilizado, que permitiu o atendimento aos critérios de custo e consumo sem comprometer a capacidade de processamento necessária.

Um processo de pesquisa de mercado mais apurado e a construção de alguns blocos, especialmente sensores, a partir de componentes mais básicos deve ser capaz de reduzir ainda mais os custos de produção, as dimensões e o consumo de energia do dispositivo.

5.6 Resumo do Capítulo

Este capítulo introduziu alguns aspectos da avaliação de sistemas computacionais como um todo e de sistemas embarcados em específico. Em seguida, foram apresentados os recursos utilizados para realizar os testes, os critérios avaliados no presente projeto e os resultados das avaliações. Por fim, foi realizada uma análise do atendimento aos requisitos funcionais e não funcionais estabelecidos no Capítulo 3.

6 CONCLUSÃO

Os principais objetivos deste trabalho foram estudar o paradigma de sistemas embarcados e aplicá-lo na construção de uma solução que fosse economicamente viável e que apresentasse baixo consumo energético. Além disso, o problema proposto aborda um tema que tem tido foco em cada vez mais instituições, sejam elas públicas ou privadas, que é a questão da eficiência energética.

Com relação aos sistemas embarcados, mostrou-se que, embora seja um conceito criado há cinquenta anos, é hoje onipresente e se encontra em plena expansão, dominando o mercado que antes era exclusivo de computadores pessoais (CHAFFEY, 2016). Por conta dessa longevidade, sua definição também está em constante evolução.

Sobre a solução proposta, o resultado obtido pode ser considerado bastante satisfatório, tendo atingido quase plenamente o objetivo proposto. Embora tenha havido dificuldade em atender a alguns dos requisitos, foi demonstrado que pequenas adaptações podem solucionar esses problemas. Assim, o produto final construído conseguiu provar que a criação de um dispositivo com as características propostas não apenas é possível como pode ser aprimorada para o desenvolvimento de um produto com aplicação profissional.

Os desafios mencionados acima e as possibilidades de melhoria serão apresentados a seguir.

6.1 Dificuldades Encontradas

O maior desafio encontrado para o êxito da aplicação proposta foi o desenvolvimento do aplicativo para *smartphone*. O desenvolvimento com o *framework* do Android é muito prolixo, demandando a instanciação de muitos elementos, além de impor algumas restrições não usuais nos ambientes tradicionais, como uma maior compartimentalização dos recursos, dos processos e das *threads*. Embora compreensível e até desejável, essas características implicam em uma curva de aprendizado maior e desestimulam a utilização do encapsulamento, um dos pilares da Programação Orientada a Objetos (MACHADO, 2016).

Em especial, a comunicação através da interface *Bluetooth* demandou muita pesquisa e diversas tentativas. Nas primeiras iterações do desenvolvimento do aplicativo, o desempenho do processamento dos dados obtidos através dessa interface foi bastante baixo, impedindo a obtenção de dados na frequência desejada. Somente após uma revisão mais cuidadosa na

rotina que processava essas informações é que foi possível processar os dados a uma taxa satisfatória.

Outra dificuldade que surgiu durante o projeto foi a configuração do módulo *Bluetooth* HC-05 para comunicação com o Arduino, em função da diferença de nível lógico citada anteriormente.

6.2 Limitações Apresentadas

Apesar de ter atendido à maioria dos requisitos, a aplicação desenvolvida apresentou algumas limitações. A principal delas está relacionada à impossibilidade de medir a potência real. O fator que impede essa medição é a ausência de um sensor que meça a tensão instantânea. Isso é necessário para recuperar a forma de onda de tensão.

Algumas outras limitações também surgiram, principalmente com relação aos recursos do aplicativo para *smartphone*. A mais evidente é a falta de um campo para informar o preço do kWh para que se possa realizar o cálculo do gasto total de energia. Também não é possível armazenar os dados obtidos, eles somente são exibidos na interface do aplicativo enquanto o mesmo estiver em execução, sendo descartados quando o aplicativo é fechado.

6.3 Contribuições

A principal contribuição do presente trabalho está na demonstração da possibilidade de integração entre *smartphones* e dispositivos embarcados de baixo custo para realizar tarefas que são tradicionalmente realizadas por equipamentos dedicados. Os *smartphones* hoje têm um poder de processamento muito grande, com a capacidade de transformar dados obtidos através de sensores de baixo custo em informações úteis.

Outra contribuição deste projeto é a apresentação da possibilidade de transição de um protótipo desenvolvido com plataformas populares para um produto final baseado nos mesmos componentes utilizados na prototipagem.

6.4 Propostas para Desenvolvimentos Futuros

Por fim, este projeto deixa margem para a evolução em diversas direções, desde o aprimoramento do aplicativo até a melhoria do dispositivo. A seguir, são listadas algumas das sugestões de recursos que podem ser agregados:

- Monitoramento de diversos sensores simultaneamente. A modificação no aplicativo para *smartphone* pode viabilizar o sensoriamento de diversos dispositivos, permitindo uma melhor comparação e a totalização do consumo em um só lugar;
- Persistência de dados. O aplicativo pode armazenar os dados obtidos na memória interna ou mesmo em um servidor remoto;
- Configuração do dispositivo a partir do *smartphone*. Podem ser incluídas opções para o usuário alterar algumas configurações do dispositivo a partir do aplicativo, como o nome do sensor e o PIN para a conexão *Bluetooth*.
- Medição de mais de um aparelho em um único dispositivo. Com a adição de mais sensores, é possível realizar a medição de diversos aparelhos através de um único dispositivo;
- Utilização de sensores mais robustos para ambientes comerciais e industriais. Com a substituição dos sensores, é possível utilizar o dispositivo em ambientes com limites de tensão e de corrente maiores;
- Utilização de um sensor de tensão que obtenha a forma de onda. Isso permitirá, como já citado, a medição do fator de potência, o que também se insere no âmbito de aplicações comerciais e industriais;
- Centralização dos dados. É possível substituir o *smartphone* por um PC ou servidor, de forma que possam ser coletados e armazenados dados de um número grande de sensores. Nesse caso, talvez seja necessário a substituição da interface *Bluetooth* por uma interface Wi-Fi ou a utilização de unidades intermediárias;

Essas são algumas das propostas de evolução do projeto proposto. Embora esta ainda não seja uma versão final de um produto viável, este capítulo mostra os caminhos que podem ser seguidos para que este projeto possa ser produzido e comercializado, além de contribuir para a eficiência energética tanto em ambientes residenciais quanto em ambientes comerciais e industriais.

REFERÊNCIAS

ABNT. ABNT NBR 14136:2012. **ABNT Catálogo**, 2012. Disponivel em: <http://www.abntcatalogo.com.br/norma.aspx?ID=306416>. Acesso em: 21 nov. 2016.

ANEEL. Bandeiras Tarifárias. **Agência Nacional de Energia Elétrica - ANEEL**, 24 fev. 2016. Disponivel em: <http://www.aneel.gov.br/tarifas-consumidores/-/asset_publisher/e2INtBH4EC4e/content/bandeira-tarifaria/654800>. Acesso em: 03 out. 2016.

ARDUINO. Arduino - Introduction. **Arduino**, 2016. Disponivel em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 03 out. 2016.

BRITO, R. C.; MARTENDAL, D. M.; OLIVEIRA, H. E. M. Máquinas de estados finitos de Mealy e Moore, Florianópolis.

CHAFFEY, D. Mobile Marketing Statistcs compilation. **Smart Insights**, 2016. Disponivel em: <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>. Acesso em: 21 nov. 2016.

CHASE, O. Ly on-line. **Ly on-line**, dez. 2007. Disponivel em: <http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>. Acesso em: 09 out. 2016.

DELAI, A. L. Sistemas Embarcados: A Computação Invisível. **Guia do Hardware**, 2013. Disponivel em: <http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html>. Acesso em: 13 out. 2016.

DIEESE. **Nota Técnica - Comportamento das tarifas de energia elétrica no Brasil**. 147. ed. [S.l.]: [s.n.], 2015. Disponivel em: <http://www.dieese.org.br/notatecnica/2015/notaTec147eletricidade.pdf>. Acesso em: 29 set. 2016.

ELECTRICAL4U. Power Factor | Calculation and Power Factor Improvement. **Electrical4u**, 2016. Disponivel em: <http://www.electrical4u.com/electrical-power-factor/>. Acesso em: 20 nov. 2016.

INATOMI, T. A. H.; UDAETA, M. E. M. Análise dos impactos ambientais na produção de energia dentro do planejamento integrado de recursos. **III Workshop Internacional Brasil-Japão: Implicações Regionais e Globais em Energia, Meio Ambiente e Desenvolvimento Sustentável**, 2005.

KOOPMAN, P. **Better Embedded System Software**. [S.l.]: Drumnadrochit Press, 2010.

LIMA, T. AGC: o primeiro grande sistema embarcado. **Embarcados**, 07 ago. 2014. Disponivel em: <http://www.embarcados.com.br/agc-primeiro-grande-sistema-embarcado/>. Acesso em: 29 out. 2016.

MACHADO, H. Os 4 pilares da Programação Orientada a Objetos. **DevMedia**, 2016. Disponivel em: <http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 21 nov. 2016.

NAKASHIMA, K. Universidade Federal de Itajubá. **Valor Médio Eficaz**, 2013. Disponivel em: <http://www.elt09.unifei.edu.br/roteiroslab/rms.pdf>. Acesso em: 11 dez. 2016.

NOERGAARD, T. **Embedded Systems Architechture:** A Comprehensive Guide for Engineers and Programmers. 1. ed. Waltham: Newnes, 2013.

PRESSMAN, R. S. **Engenharia de Software:** Uma Abordagem Profissional. 7. ed. Porto Alegre: AMGH, 2011.

PROCEL INFO. Selo Procel. **Procel Info**, 2006. Disponivel em: <http://www.procelinfo.com.br/main.asp?TeamID=%7B88A19AD9-04C6-43FC-BA2E-99B27EF54632%7D>. Acesso em: 03 out. 2016.

REIS, F. Introdução aos Sistemas Embarcados. **Bóson Treinamentos em Tecnologia**, 2015. Disponivel em: <http://www.bosontreinamentos.com.br/eletronica/eletronica-geral/introducao-aos-sistemas-embarcados/>. Acesso em: 09 out. 2016.

SILVEIRA, C. B. Desvendando a Comunicação RS232. **Citisystems**, 2016. Disponivel em: <https://www.citisystems.com.br/rs232/>. Acesso em: 12 dez. 2016.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

TANENBAUM, A. S. **Redes de Computadores**. 4. ed. Rio de Janeiro: Campus, 2003.

TECHTARGET. What is a System on a Chip (SoC)? **TechTarget**, 2016. Disponivel em: <http://internetofthingsagenda.techtarget.com/definition/system-on-a-chip-SoC>. Acesso em: 15 nov. 2016.

TEIXEIRA, F. B. Definição de Potência e Energia (Alexander & Sadiku). **Energia Elétrica**, 2015. Disponivel em: <http://www.energiaeletrica.net/definicao-de-potencia-e-energia-alexander-sadiku/>. Acesso em: 11 dez. 2016.

TEIXEIRA, F. B. O que é a Tensão Elétrica? **Energia Elétrica**, 2015. Disponivel em: <http://www.energiaeletrica.net/tensao-eletrica/>. Acesso em: 11 dez. 2016.

TEIXEIRA, F. B. O que é a Corrente Elétrica? **Energia Elétrica**, 2016. Disponivel em: <http://www.energiaeletrica.net/corrente-eletrica/>. Acesso em: 11 dez. 2016.

TUTORIALS POINT. Embedded Systems: Architecture Types. **Tutorials Point**, 2016. Disponivel em: <https://www.tutorialspoint.com/embedded_systems/es_architectures.htm>. Acesso em: 6 nov. 2016.

UNIVERSITY OF MICHIGAN. PowerTutor. **University of Michigan**, 2011. Disponivel em: <http://ziyang.eecs.umich.edu/projects/powertutor/>. Acesso em: 15 dez. 2016.

WEISSTEIN, E. W. Root-Mean-Square. **Wolfram Math World**, 2016. Disponivel em: <http://mathworld.wolfram.com/Root-Mean-Square.html>. Acesso em: 20 nov. 2016.

APÊNDICE – Listagem de código do dispositivo

```
#include <SoftwareSerial.h>
#include "TimerOne.h"

#define INTERVALO 8000               // Duração do timer Timer1 (em microssegundos)

SoftwareSerial BTSerial(2, 3);       // Pinos da interface serial com o módulo Bluetooth

int currentSensorPin = A0;           // Pino analógico do sensor de corrente
int tensionSensorPin = A1;           // Pino analógico do sensor de tensão

unsigned int tempo;                  // Variável para armazenar timestamp

float currentValue = 0;              // Variável para armazenar corrente em Ampères
float tensionValue = 0;              // Variável para armazenar tensão em Volts
float voltsporUnidade = 0;           // Variável para auxiliar na conversão da corrente para
Ampères

void setup() {

        // Setup do timer de hardware Timer1
        Timer1.initialize(INTERVALO);
        Timer1.attachInterrupt(readSensors);
        Timer1;

        // Setup dos pinos de leitura dos sensores
        pinMode(currentSensorPin, INPUT);
        pinMode(tensionSensorPin, INPUT);

        // Cáluclo para auxiliar na conversão da leitura de corrente
        voltsporUnidade = 5.0 / 1024.0;

        // Inicialização da interface serial para o módulo Bluetooth
        BTSerial.begin(38400);
}


// Rotina que lê os sensores,
// calcula os valores e envia
// por Bluetooth
void readSensors() {

        // Lê os valores dos sensores e converte nas unidades de medida padronizadas
        currentValue = ((analogRead(currentSensorPin)-511.0) * voltsporUnidade) / 0.1;
        tensionValue = analogRead(tensionSensorPin) * 0.146;

        // Obtêm o tempo de milissegundos desde a inicialização do dispositivo
        tempo = millis();

        // Envia os valores por Bluetooth
        BTSerial.print(currentValue,3);
        BTSerial.print(",");
        BTSerial.print(tensionValue,3);
        BTSerial.print(",");
        BTSerial.print(tempo);
        BTSerial.print("\n");
}

// Laço principal, não faz nada
void loop() {

}
```