LUIZ DIONISIO PEDRINI

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA SENSÍVEL AO TOQUE PARA MONITORAMENTO DE VARIÁVEIS DINÂMICAS DE UM SISTEMA DE ENERGIA ININTERRUPTA

FLORIANÓPOLIS, 2012

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE PÓS-GRADUAÇÃO (LATO SENSU) EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

LUIZ DIONISIO PEDRINI

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA SENSÍVEL AO TOQUE PARA MONITORAMENTO DE VARIÁVEIS DINÂMICAS DE UM SISTEMA DE ENERGIA ININTERRUPTA

Monografia submetida ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Especialista em Desenvolvimento de Produtos Eletrônicos.
Professor Orientador: Fernando S. Pacheco, Dr. Eng.

FLORIANÓPOLIS, 2012

CDD 621.3815
P371d

Pedrini, Luiz Dionísio
Desenvolvimento de uma interface gráfica sensível ao toque para monitoramento de variáveis dinâmicas de um sistema de energia ininterrupta [monografia] / Luiz Dionísio Pedrini; orientação de Fernando S. Pacheco. – Florianópolis, 2013.

1 v. : il.

Monografia de especialização (Desenvolvimento de Produtos Eletrônicos) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. Curso de Pós-graduação *Lato Sensu* em Desenvolvimento de Produtos Eletrônicos.

Inclui referências.

1. Sistema de energia ininterrupta. 2. Display gráfico. 3. Tela sensível ao toque. I. Pacheco, Fernando S. II. Título.

Sistema de Bibliotecas Integradas do IFSC
Biblioteca Dr. Hercílio Luz – Campus Florianópolis
Catalogado por: Edinei Antonio Moreno CRB 14/1065
Rose Mari Lobo Goulart CRB 14/277

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA MONITORAMENTO DE VARIÁVEIS DINÂMICAS DE UM SISTEMA DE ENERGIA ININTERRUPTA

LUIZ DIONISIO PEDRINI

Este trabalho foi julgado adequado para obtenção do Título de Especialista e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação (Lato Sensu) em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 12 de dezembro de 2012.

Banca Examinadora:

_____
Fernando S. Pacheco, Dr. Eng.
Orientador

_____
Clóvis Antônio Petry, Dr. Eng.

_____
Walter Leonardo Prates Perreira, Esp.

Dedico este trabalho a Deus.
Aos meus familiares.
A minha noiva.

“Penso noventa e nove vezes e nada descubro; deixo de pensar, mergulho em profundo silêncio e eis que a verdade se me revela**.”**
(Albert Einstein)

“Muitas das grandes realizações do mundo foram feitas por homens cansados e desanimados que continuaram trabalhando”
(Kléber Novartes)

RESUMO

As interfaces gráficas sensíveis ao toque vêm ganhando destaque e se popularizando com o passar do tempo. A evolução de suas tecnologias faz com que o usuário tenha uma interação mais amigável com os sistemas que as utilizam. Além de serem atrativas a novos produtos, as interfaces gráficas sensíveis ao toque vem ganhando espaço também na atualização de produtos já existentes. Este trabalho apresenta o desenvolvimento de uma interface gráfica que poderá substituir a antiga interface analógica de um sistema de energia ininterrupta. Esta interface gráfica utiliza um display gráfico com sensor *touch screen* em conjunto com a biblioteca gráfica Microchip e se comunica com o sistema de energia ininterrupta através do protocolo UPS Standard Protocol, Desenvolvida na linguagem C e contando com um total de sete telas, a interface permite o monitoramento das variáveis dinâmicas do sistema de energia de ininterrupta de forma satisfatória. Este trabalho também apresenta um software simulador de comunicação, desenvolvido na linguagem Java, com o intuito de facilitar os testes da interface gráfica, sem necessitar que esta esteja conectada ao equipamento real.

Palavras-chave: Sistema de energia ininterrupta, display gráfico, tela sensível ao toque.

ABSTRACT

Touch Screen enabled graphic interfaces had become more popular and have been on the highlights nowadays. The evolution of the technologies of these devices is making the user experience more straightforward. This technology is an attractive solution to new systems in development, and also touch screen enabled interfaces has become a common update on legacy projects. This work presents the development of a touch screen enabled graphic interface that may replace an old analog interface of an uninterruptible power supply. The graphic interface uses a graphic display and a touch screen sensor in combination with the Microchip graphic library and communicates with the UPS through the UPS Standard Protocol. Developed in the C language and counting with seven screens, the graphic interface allows monitoring of the dynamics variables of the UPS. This work also presents the development of a communication simulation software developed in Java, intended to help on the graphic display testing routines, eliminating the need of a real UPS system.

Key-words: Uninterruptible power supply, graphic display, touch screen.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

ABREVIATURAS

UPS	Uninterruptible Power Suply
RGB	Red – Green – Blue
IHM	Interface Humano-Máquina
LED	Light Emitting Diode
TFT	Thin Film Transistor
SRAM	Static random-access memory
RAM	Random-access memory
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter
DMA	Direct Memory Access
PMP	Parallel Master Port
GFX	Graphics Controller Module
GPU	Graphics processing unit
SNMP	Simple Network Management Protocol
ASCI	American Standard Code for Information Interchange

SUMÁRIO

1. INTRODUÇÃO

Um desafio de todo projetista de sistemas eletrônicos é encontrar a melhor forma de realizar a interação do seu sistema com o usuário final. O nível de conforto do usuário ao usar um dado equipamento pode significar o sucesso ou o fracasso do projeto.

Tanto na indústria como na academia, dedica-se bastante tempo à criação e atualização de tecnologias que podem ser utilizadas para tornar a experiência do usuário mais agradável. O que se busca é permitir que o usuário utilize o sistema sem perceber a presença da interface.

As interfaces gráficas e as telas sensíveis ao toque não são tecnologias recentes, porém seu uso tem se acentuado na última década. As telas sensíveis ao toque ganharam destaque com sua ampla utilização em telefones celulares, especialmente após o lançamento do iPhone da Apple em 2007 (GUEDES, 2009).

Os sistemas de energia ininterrupta estão presentes nos mais diversos setores da indústria, comércio e também em residências. Estes sistemas visam suprir a demanda de energia de equipamentos críticos na ocorrência de falhas na rede de alimentação de energia elétrica das concessionárias.

Os sistemas de energia ininterrupta podem ser *nobreaks*, retificadores, inversores, etc. Durante sua operação, estes sistemas monitoram uma grande variedade de parâmetros da rede de energia elétrica para tomada de decisão. Por exemplo, um *nobreak* monitora os parâmetros da rede de energia da concessionária de modo a determinar se sua carga deve ser alimentada pela rede ou pelas baterias (MICROCHIP, 2011c).

Os parâmetros monitorados pelo sistema de energia são também de interesse do usuário que pode, através destes, levantar informações relativas à qualidade da energia entregue pela concessionária, estado dos bancos de bateria e qualidade da energia entregue à carga. Esses parâmetros também podem nortear a programação de manutenções preventivas e substituição de acumuladores de energia.

A empresa EquisulGPL desenvolve e comercializa sistemas de energia ininterrupta, como *nobreak*, retificadores e inversores. Seu portfólio de produtos possui desde *nobreaks* residenciais de 600 VA até equipamentos industriais com mais de 2 MW (EQUISUL, 2012).

Na busca por atualizar os itens de seu portfólio de produtos, bem como agregar novas tecnologias aos produtos em desenvolvimento, a

empresa busca aprimorar suas tecnologias de interface humano-máquina.

Este trabalho, desenvolvido em parceria com a EquisulGPL, propõe o desenvolvimento de um protótipo de interface gráfica sensível ao toque para seus sistemas de energia ininterrupta.

Do protótipo obtido neste trabalho, futuramente, serão derivados os produtos finais adaptados a cada um dos projetos onde forem empregados. Esta especialização se dará nas informações exibidas, porém, utilizando como base o resultado deste trabalho.

Este trabalho também apresenta relevância acadêmica, uma vez que as técnicas nele empregadas podem ser utilizadas por outros acadêmicos em trabalhos derivados ou similares.

Para nortear o desenvolvimento deste trabalho, optou-se por desenvolver a interface de forma que esta possa se comunicar com produtos já desenvolvidos e por isso optou-se por utilizar o protocolo de comunicação UPS *Standard Protocol*, presente em grande parte dos sistemas de energia ininterrupta.

1.2 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma interface gráfica, com suporte a *touch screen*, que se comunicará com um sistema de energia ininterrupta e exibirá suas variáveis dinâmicas. A comunicação é realizada através do protocolo *UPS Standard Protocol*.

1.3 Objetivos Específicos

São objetivos específicos deste trabalho:

- Projetar e desenvolver a interface gráfica que exibirá os valores das variáveis do sistema de energia ininterrupta;
- Projetar e desenvolver o *driver* de comunicação compatível com o protocolo *UPS Standard Protocol*;
- Desenvolver um mecanismo de registro de eventos baseado nas informações recuperadas do sistema de energia ininterrupta; e
- Desenvolver um software simulador de comunicação, compatível com o protocolo *UPS Standard Protocol*, a fim de simplificar a validação da interface.

2. REVISÃO BIBLIOGRÁFICA

O objetivo deste Capítulo é apresentar uma revisão bibliográfica dos temas que permeiam o desenvolvimento deste trabalho. Inicialmente serão apresentados conceitos relativos às interfaces gráficas. A comunicação serial será tratada na sequência. Serão estudadas ainda as características do microcontrolador selecionado para o desenvolvimento e o protocolo de comunicação empregado.\

2.1 Interfaces gráficas

Nesta Seção serão revisados conceitos que estão relacionados ao desenvolvimento de interfaces gráficas. Serão abordados conceitos como sistemas de cores, terminologias e características de *displays* gráficos e telas sensíveis ao toque.

2.1.1 Sistemas de cores

Em sua forma mais básica, as cores estão associadas com o comprimento de onda da luz. As cores variam em um largo espectro, desde comprimentos de onda menores que 200 nm no chamado ultravioleta distante até comprimentos de onda maiores que 1000 nm do espectro infravermelho (STOLFI, 2008).

Apesar deste grande espectro de cores, o olho humano consegue captar apenas uma faixa restrita entre 380 nm, que representa a cor violeta e 740 nm que representa a cor vermelha (RAMOS, 2006). A FIGURA 1 representa o espectro de luz visível ao olho humano.

FIGURA 1 - Espectro de luz visível.

Outra característica do olho humano é que este realiza uma mistura das cores captadas sobrepostas. Por exemplo, se o olho receber duas ondas incidentes misturadas, uma com comprimento de onda equivalente ao vermelho (700 nm) e outra equivalente ao verde (560 nm) ele as interpretará como uma única cor amarela, mesmo que o comprimento de onda para amarelo não esteja presente. Esta característica permite que se use uma combinação de cores básicas para gerar outras cores e é a base para a exibição de imagens coloridas em *displays* (STOLFI, 2006; RAMOS, 2008).

Em um sistema digital as cores devem ser definidas através de valores numéricos, e para isso são necessários padrões de representação destas cores. Um sistema padrão de fato é o sistema RGB, utilizado em televisores, monitores, escâner, etc (ORAN, 2012).

O sistema de cores RGB (*red – green – blue* ou vermelho – verde – azul), também chamado de sistema aditivo, utiliza uma proporção de cada uma destas três cores básicas para representar todo o espectro visível (MIGUEL; RUFINO, 2010).

A quantidade de cores que podem ser exibidas depende da quantidade de bits utilizados na representação RGB e é denominada profundidade de cores. Com três bytes (24 bits) é possível representar 16 milhões de cores. Esta configuração é denominada "*True Color*". Também é comum utilizar 16 bits para representar uma cor, o que origina uma profundidade de 16 mil cores, o que é suficiente para a maioria das aplicações gráficas (ORAN, 2012; MICROCHIP, 2011a).

Em sistemas com 16 bits de profundidade de cor, o sistema RGB divide os bits para representar a proporção de cada uma das três cores básicas. Essa divisão é feita da seguinte forma: os cinco bits mais significativos para a cor vermelha, os cinco bits menos significativos para a cor azul e seis bits restantes para a verde (MICROCHIP, 2011a).

A escolha da cor verde para receber um bit a mais se deve ao fato de o olho humano reconhecer uma gama maior de tonalidades de verde, devido a essa cor localizar-se na região central do espectro visível (PACALE, 2003).

Algumas características das cores RGB podem ser ressaltadas:

- Para criar uma cor básica, basta ajustar o seu valor para o brilho desejado, e manter as outras com valor zero;
- Misturar proporções equivalentes em todas as cores básicas resultará na cor cinza;
- Misturar o valor máximo em todas as cores resultará na cor branca; e
- Misturar o valor mínimo em todas as cores resultará na cor preta.

2.1.2 Terminologia e características de displays gráficos

Um *display*, independente da tecnologia empregada em sua construção, pode ser genericamente definido como um conjunto de pixels. Em um display colorido cada pixel forma um ponto de luz capaz de emitir as três cores básicas: vermelho, verde e azul. Em um display monocromático, cada pixel forma um ponto de luz que pode ser ligado ou desligado (MICROCHIP, 2011).

O tamanho de um display está associado a sua área visível e é comumente apresentado em polegadas. A medida é realizada em uma das diagonais da tela, ou seja, um display de 7 polegadas terá uma diagonal visual de 7 polegadas (FUJITSU, 2006; MICROCHIP, 2011) .

A resolução de um display é determinada pela quantidade de pixels usados em sua construção. O formato padrão para se representar a resolução de um display indica a densidade de pixel por coluna e por linha. Por exemplo, um display com resolução de 320x240 terá 320 pixels por linha e 240 pixel por coluna, quando usado em modo paisagem (GUEDES, 2009; MICROCHIP, 2011). Uma televisão *full-hd*, por exemplo, possui 1920 pixels por linha e 1080 pixels por coluna.

Diz-se que um display está sendo usado em modo paisagem quando a orientação da imagem faz com que a largura seja maior que a altura. Quando a largura é menor que altura o display está em modo retrato (GUEDES, 2009). Outra característica dos displays é sua proporção de tela. A proporção de tela é dada pela razão entre a largura

e a altura do display. As proporções de tela mais conhecidas são 4:3 e 16:9 (GUEDES, 2009). A resolução 16:9 é conhecida como *wide screen*, ou tela larga.

Além das características físicas apresentadas, existem algumas características operacionais relativas aos displays gráficos. A seguir são apresentadas as mais importantes.

A taxa de atualização representa quantas atualizações de tela o display suporta por segundo. Esta será a taxa máxima em que o processador poderá atualizar o display. Atualização a uma taxa menor pode comprometer a fluidez da troca de imagens e a intensidade do brilho da imagem (NXP, 2011).

Um *display* comum não possui nenhuma forma de memória e seus pixels devem ter seus valores reescritos constantemente. A não atualização dos valores fará com que o display apague, não apresentando nenhuma imagem (GUEDES, 2009).

Outra característica que deve ser observada é o *ghosting*. Esta característica representa o tempo que o quadro anterior continua visível na tela, em menor intensidade, no evento de uma atualização. O nome *ghosting* vem do fato de essa imagem residual de baixa intensidade aparecer como um fantasma sobre a nova imagem (FUJITSU,2006).

O contraste da tela representa a taxa de diferença entre os pontos claros e escuros do display. Quanto maior o contraste, melhor a definição da imagem no display (MICROCHIP, 2011).

A maioria dos displays necessita de uma fonte luminosa que servirá como base para a iluminação dos pixels. Esta fonte luminosa é conhecida como *backlight*. Este *backlight* pode ser obtido através de lâmpadas fluorescentes ou diodos emissores de luz (LED). Por exemplo, displays do tipo TFT (*Thin film transistor*) utilizam uma camada interna para polarizar a luz do *backlight* para gerar os pixels (FUJITSU,2006).

2.1.3 Telas sensíveis ao toque

A tecnologia de telas sensíveis ao toque permite capturar os pontos pressionados na tela. Esta captura permite que os toques do

usuário na tela sejam convertidos em entradas para o software de controle da interface homem-máquina (IHM).

Apesar do nome “tela sensível ao toque”, esta tecnologia não está relacionada com a tela em si. A tecnologia de telas sensíveis ao toque agrega camadas sobre a tela comum de modo a capturar os toques do usuário. Estas camadas variam de acordo com a tecnologia empregada, dentre as quais podemos citar o *touch screen* resistivo.

No *touch screen* resistivo são empregadas duas películas transparentes recobertas de material condutor elétrico e resistivo translúcido, como, por exemplo, o óxido de índio. Essas duas películas são sobrepostas mantendo-se uma pequena separação entre elas. Quando pressionada, a película superior entra em contato com a inferior, gerando o efeito semelhante a um divisor resistivo (HOYE; KOZAC, 2010).

Os sensores de *touch screen* mais comumente encontrados no mercado possuem quatro, cinco ou oito vias para medição dos pontos de toque. Neste trabalho são utilizados sensores de quatro vias.

Nos sensores de quatro vias, cada película possui terminais que permitem a aplicação de uma diferença de potencial elétrico na camada resistiva. Estes terminais são dispostos de forma perpendicular entre a camada inferior e a superior, de modo a permitir medições em dois eixos distintos (eixos x e y) (DOWNS, 2005).

Para medição do ponto de toque, aplica-se uma diferença de potencial elétrico a uma das películas e mede-se a tensão resultante na outra película. Esta tensão resultante representará o afastamento do ponto pressionado naquele eixo. Para medir a posição tocada no segundo eixo a operação é repetida com a inversão na ordem de utilização das películas (DOWNS, 2005).

Os sinais elétricos obtidos das películas são convertidos para valor numérico através de um conversor analógico/digital para que possam ser processados pelo software que tratará os pontos tocados. O projetista do sistema deve tomar medidas para minimizar a presença de ruídos elétricos no sinal, a fim de evitar leituras com valores errados, ou mesmo que sejam detectados toques inexistentes.

É importante destacar que a presença das duas películas sobre a tela causa uma diminuição do fluxo luminoso da tela que pode chegar a 25% (HOYE; KOZAC, 2010).

2.1.4 O sistema necessário ao funcionamento de um display gráfico

A quantidade de componentes necessários à criação de um sistema gráfico pode variar de acordo com a aplicação, porém, alguns componentes são fundamentais. São eles: o *frame buffer*, o processador de imagens e o controlador do display.

O *frame buffer* é uma região de memória que contêm os valores RGB de cada pixel que será exibido no display. Estes valores serão utilizados pelo controlador do display na rotina de atualização. O *frame buffer* também é acessado pelo processador de imagens quando este necessita alterar o conteúdo da tela (GUEDES, 2009).

O *frame buffer* pode ocupar uma quantidade considerável de memória dependendo da resolução do display e da profundidade de cor selecionado. O tamanho do *frame buffer* em bytes ($T_{frame\ buffer}$) pode ser obtido pela Equação 1, onde n_{pixels} representa o número de pixels e p_{cor}, a profundidade de cor. Por exemplo, em um display de 640x480 pixels, com profundidade de cor de 16 bits, serão necessários 600 kBytes de memória para o *frame buffer*.

$$\mathrm{T_{frame\ buffer}} = \frac{\mathrm{n_{pixels}} \cdot \mathrm{p_{cor}}}{8}$$

EQUAÇÃO 1- Cálculo do tamanho do frame buffer

O *frame buffer* pode estar alocado internamente ao controlador do display, internamente ao processador de imagens, ou ser uma memória externa, como uma SRAM.

O controlador do display tem como função básica atualizar constantemente os pixels do display com os valores armazenados no *frame buffer*.

O processador de imagens realiza a maior parte do processamento do sistema, sendo responsável por tratar previamente as imagens de acordo com a aplicação e transferí-las ao *frame buffer* para que sejam exibidas. O processador deve ter capacidade de processamento suficiente para realizar as transições de imagem requeridas pela aplicação para que sejam exibidas de forma agradável ao olho humano (GUEDES, 2009).

2.2 O microcontrolador PIC24FJ256DA210

O microcontrolador PIC24FJ256DA210 é um microcontrolador da fabricante Microchip, com arquitetura de 16 bits, voltado para aplicações gráficas. Com um encapsulamento de 100 pinos, esse dispositivo dispõe de 84 pinos de I/O. Estão internamente disponíveis 256 kB de memória de programa e 96 kB de memória RAM (MICROCHIP, 2010). Dentre os diversos periféricos presentes, pode-se destacar:

- Interface USB OTG;
- Quatro UARTs;
- Dois canais de DMA; e
- Porta paralela mestre (PMP) que possibilita a interface com dispositivos de armazenamento como memórias flash e SRAM paralelas.

O diferencial deste microcontrolador, no que tange às aplicações gráficas, é a presença de um periférico interno denominado *Graphics Controller Module* (GFX). O GFX é um periférico que atua como controlador de display, fazendo interface direta com displays CSTN (*Color super-twisted nematic display*) e TFT. O GFX possui pinos dedicados tanto para os dados RGB quanto para os sinais de controle dos displays (MICROCHIP, 2009).

Além do GFX, o PIC24FJ256DA210 conta ainda com três unidades de aceleração gráfica por hardware (GPU) que minimizam o tempo necessário às operações de tratamento de imagem e gerenciamento do *frame buffer*. Essas GPUs realizam operações como

movimentação de áreas de memória, descompressão de imagens e renderização de texto sem nenhuma intervenção da CPU (MICROCHIP, 2010).

Usando o GFX em conjunto com a porta paralela mestre, é possível utilizar uma memória SRAM paralela para aumentar o tamanho do *frame buffer*. Todo o acesso à SRAM externa é feito automaticamente pelo módulo GFX (MICROCHIP, 2009).

O módulo GFX é um periférico com um grande número de registradores de configuração. Estes registradores devem ser carregados com os valores corretos de modo a compatibilizar a temporização do GFX com a do display utilizado.

A Microchip disponibiliza gratuitamente uma biblioteca para aplicações gráficas que além de facilitar a configuração do GFX através de um conjunto de macros, fornece ainda uma série de serviços que facilitam a criação de interfaces gráficas.

2.3 A biblioteca gráfica Microchip

A biblioteca gráfica Microchip é uma biblioteca disponibilizada em código fonte, sem custos, para execução em processadores da Microchip. Esa biblioteca está organizada em camadas bem definidas. Cada camada utiliza serviços da camada inferior e fornece serviços às camadas superiores.

A primeira camada representa o *device driver* do controlador de display. Essa camada implementa os serviços mais básicos da aplicação gráfica e faz a interface entre o *frame buffer* e o controlador do display. É esta camada que fornece os serviços de leitura e escrita dos valores de cada um dos pixels do display (MICROCHIP, 2012b). Quando utilizada em conjunto com o PIC24FJ256DA210 é essa camada que realiza a configuração do GFX. Reside aqui também a implementação do código de acesso às GPU do processador e suas funcionalidades (MICROCHIP, 2012a).

Além de dar suporte ao GFX, esta camada possui *device drivers* para diversos controladores de display externos.

Para facilitar a configuração desta camada, o arquivo de cabeçalhos denominado *HardwareProfile.h* contem uma série de macros predefinidos que permitem indicar a temporização necessária para o display que está sendo utilizado, as regiões de memória reservadas ao *frame buffer*, o acesso a memórias externas ao microcontrolador, etc (MICROCHIP, 2012b).

A Microchip disponibiliza versões do arquivo *Hardwareprofile.h* pré-configuradas para os display de seus kits de desenvolvimento, sendo de grande valia para o entendimento de como realizar tal configuração.

A segunda camada da biblioteca é denominada camada primitiva. Esta camada fornece serviços para a criação de formas geométricas básicas como linhas, retângulos, círculos, barras, etc. É esta camada também a responsável pelo processamento de fontes e renderização de caracteres (MICROCHIP, 2012b).

A terceira camada é denominada camada de objetos gráficos. Esta camada utiliza os serviços da camada primitiva para criar objetos gráficos mais complexos (MICROCHIP, 2012b). Estes objetos gráficos englobam os componentes usualmente encontrados em interfaces gráficas, sendo eles:

- campos de texto;
- botões;
- caixas de combinação;
- caixas de seleção;
- tabelas;
- caixas de listagem;
- campos de imagem.

Esta camada é altamente configurável, permitindo que apenas os objetos usados na aplicação sejam compilados, reduzindo a área de memória ocupada pela biblioteca. Esta configuração é feita através de um conjunto de macros disponíveis no arquivo *GraphicsConfig.h* (MICROCHIP, 2012b).

Os objetos desta camada são alocados dinamicamente em memória, fato que permite que objetos sejam inseridos ou removidos da tela a qualquer momento. Esta alocação dinâmica necessita que uma

área da memória RAM seja reservada para o espaço de *heap*. O tamanho da memória *heap* deve ser definido pelo usuário, levando-se em conta o maior número possível de objetos criados simultaneamente (MICROCHIP, 2012b). Por exemplo, um botão ocupa oito bytes da memória *heap* quando criado.

A criação de um objeto consiste na alocação da memória *heap* que irá contê-lo. Cada objeto possui seu próprio serviço de criação, com alguns parâmetros sendo comuns a todos os objetos. O primeiro parâmetro comum é que cada objeto recebe um valor inteiro como identificador. Este identificador será utilizado para permitir a posterior localização deste objeto dentro da coleção de objetos criados (MICROCHIP, 2012b).

Outro conjunto de parâmetros comum à criação de todos os objetos é sua localização na tela. Esta localização é feita através de quatro valores: *left*, *right*, *top* e *bottom*. O valor *left* indica a coluna na qual estará a borda esquerda do objeto, enquanto o valor *right* indicará a coluna da borda direita. De forma similar, o valor *top* indicará a linha que conterá a borda superior do objeto, e o valor *bottom* indicará a linha da borda inferior.

Desta forma a largura do objeto será dada pela expressão (*right* – *left*) e a altura do objeto será dada pela expressão (*bottom* – *top*).

A camada de objetos gráficos permite ainda a criação de esquemas de cor distintos que podem ser atribuídos a um objeto em sua fase de criação. Tais esquemas facilitam a criação da interface, pois torna mais fácil a padronização visual dos objetos. Os esquemas de cor, assim como os objetos podem ser alocados dinamicamente e necessitam de espaço na memória *heap* (MICROCHIP, 2012b).

É esta característica de alocação dinâmica de objetos que permite a criação de aplicações com múltiplas telas. A troca entre telas consiste em destruir todos os objetos presentes na tela e criar os objetos que compõem a nova tela. A camada de objetos gráficos disponibiliza o serviço *GOLfree()* que destrói todos os objetos já criados (MICROCHIP, 2012a).

Os valores das cores podem ser obtidos em outro arquivo da biblioteca denominado *gfxcolors.h*. Este arquivo contem macros com os valores RGB das cores comumente utilizadas (MICROCHIP, 2012b).

A última camada que compõe a biblioteca é a camada de tratamento de mensagens. Esta camada recebe dados de dispositivos de entrada como telas sensíveis ao toque e teclados. Em conjunto com o driver do sistema *touch screen*, esta camada identifica os toques do usuário na tela e obtem suas coordenadas. O mesmo princípio pode ser usado para teclados capacitivos ou teclados mecânicos (MICROCHIP, 2012a).

Finalmente, acima das camadas da biblioteca gráfica fica a camada de aplicação, que utiliza os serviços da biblioteca gráfica para criar e controlar a interface gráfica.

2.3.1 Funcionamento básico da biblioteca

Para que a biblioteca gráfica funcione corretamente, alguns de seus serviços devem ser chamados ciclicamente. A FIGURA 2 apresenta um trecho de código que exemplifica o ciclo básico de funcionamento da biblioteca, com suporte a *touch screen* e teclado (MICROCHIP, 2012a).

```
while(1)
{
      if(GOLDraw())
      {
            TouchGetMsg(&msg);
            GOLMsg(&msg);
            SideButtonsMsg(&msg);
            GOLMsg(&msg);
      }
}
```

FIGURA 2 - Trecho de código principal do funcionamento da biblioteca gráfica.

A chamada de *GOLDraw()* faz com que a biblioteca atualize os objetos no *frame buffer* e, consequentemente, no display. Essa função retorna um valor lógico verdadeiro caso todos os objetos tenham sido atualizados (MICROCHIP, 2012b).

A chamada da função *TouchGetMsg(&msg);* recupera a última mensagem válida e não processada da camada de mensagens. Neste caso será o último toque na tela não processado. A resposta também pode indicar que não existe nenhum toque a ser processado. A chamada de *SideButtonsMsg(&msg);* realizará o mesmo processo, porém com as entradas via teclado (MICROCHIP, 2012b).

Depois de recuperada uma mensagem, a mesma deve ser tratada pela camada de objetos gráficos. Isto se dá através da chamada da função *GOLMsg(&msg);*. Isto fará com que a camada de objetos gráficos cruze as informações da camada de mensagens com os objetos ativos, disparando um evento case necessário. Por exemplo, se um toque na tela for capturado e passado à camada de objetos, e a posição do toque estiver dentro da área de um botão, será gerado o evento de botão pressionado (MICROCHIP, 2012a; MICROCHIP, 2012b).

A captura dos eventos da camada de objetos gráficos se dá através de uma função específica, definida bela biblioteca gráfica e que deve ser obrigatoriamente implementada na camada de aplicação. A assinatura desta função é: *WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER *pObj, GOL_MSG *pMsg)* (MICROCHIP, 2012b).

O parâmetro *objMsg* define qual o tipo de evento foi reconhecido. O parâmetro **pObj* é um apontador para o objeto associado ao evento. O Parâmetro **pMsg* depende do tipo do evento e pode ou não conter mais informações sobre o mesmo (MICROCHIP, 2012b).

Outra função chamada pela camada de objetos gráficos e que deve obrigatoriamente ser implementada na camada de aplicação é a função *GOLDrawCallback(void)*. Esta função é invocada quando a função *GOLDraw()* acabar de atualizar todos os objetos da tela (MICROCHIP, 2012b). Nesta função é possível desenhar objetos na tela que não sejam fornecidos pela camada de objetos gráficos. É nesta tela

tambem que podem ser realizadas ações necessárias para a troca da tela ativa.

2.4 Comunicação Serial

Uma comunicação de dados é dita serial quando o envio dos dados é feito em uma única linha de dados, bit a bit, diferentemente da comunicação paralela, onde um conjunto de bits é enviado simultaneamente. A comunicação serial pode ser síncrona ou assíncrona (CANZIAN,2006; EPUSP,2012).

Na comunicação serial síncrona, além da linha de dados, também é utilizada uma linha de sincronismo que trafega pulsos de sincronismo (*clock*). O protocolo de comunicação deve estabelecer a polaridade dos pulsos de sincronismo bem como o responsável por sua geração. Uma vez definidos, os pulsos de sincronismos são utilizados pelo receptor para determinar os instantes nos quais serão amostrados os dados na linha de dados (CANZIAN, 2006; EPUSP, 2012).

Na comunicação serial assíncrona, os sinais de sincronismos são transferidos juntamente com os dados. Essa forma de sincronismo deve estar definida pelo protocolo e deve permitir a sincronização entre transmissor e receptor. Na comunicação assíncrona existe um tempo de espera variável entre o final de um bloco de dados e o início do outro. No início de um novo bloco a sincronização deve ser refeita, permitindo que os dados sejam amostrados corretamente pelo receptor (CANZIAN, 2006; EPUSP, 2012).

2.4.1 UART

Um UART (*Universal Asynchronous Receiver/Transmitter* ou transmissor/receptor assíncrono universal) é um dispositivo responsável pela comunicação de dados paralelos em um meio de transmissão serial. Sua principal função é converter dados entre as formas paralela e serial (EPUSP, 2012).

Presente na maioria dos microcontroladores, o UART facilita a implementação de comunicação serial em dispositivos embarcados. O UART normalmente é utilizado com o hardware padrão de comunicação serial, como RS-232, RS-422 e RS-485. Como o sinal de saída do UART normalmente fica limitado às faixas de alimentação do microcontrolador, se faz necessário a utilização de circuitos externos para compatibilizar este sinal com os demais padrões elétricos (EPUSP, 2012).

2.4.2 O Padrão EIA-RS-232

Na década de 60, a utilização de modems começou a se popularizar, à medida que a troca de informações entre computadores utilizando a rede de telefônica começou a se expandir. A comunicação com os modems era feita de forma serial, porém a falta de um padrão fazia com que os cabos, conectores e níveis de tensão fossem incompatíveis entre equipamentos de fornecedores diferentes (CANZIAN, 2006).

Em 1969, a EIA (*Electronic Industries Alliance*), juntamente com a Bell Labs e outros fabricantes começaram a desenvolver um padrão para conexão de equipamentos de comunicação de dados. Este esforço culminou com o protocolo EIA-RS-232, que define sinais e níveis elétricos (CANZIAN, 2006).

Nos anos 80, a crescente indústria de computadores pessoais viu no padrão EIA-RS-232 uma forma apropriada e barata para interconexão de equipamentos. Rapidamente ele se tornou o padrão para conexão de periféricos como impressora, modems e outros PCs.

O padrão sofreu várias evoluções com o passar do tempo, e sua versão mais recente é a ANSI/EIA/TIA-232-F de 1997. Mesmo não sendo a mais recente, a versão mais difundida e mais empregada é a versão EIA-RS-232-C de 1969 (CANZIAN, 2006).

Esse padrão define um conjunto de sinais para a transmissão de dados. A seguir são apresentados os principais sinais definidos pelo protocolo:

- *Signal Ground* (GND): fornece o sinal de referência elétrica (0V) para todas as linhas de dados;
- *Transmited data* (TX ou TD): é o canal de dados associado ao transmissor do equipamento;
- *Received data* (RX ou RD): é o canal de dados associado ao receptor do equipamento;
- *Request to send* (RTS): Indica que o equipamento deseja enviar dados;
- *Clear to send* (CTS): Indica que o outro dispositivo está apto a receber dados.

Os sinais CTS e RTS servem como mecanismo de controle de fluxo entre dois dispositivos. Quando um dispositivo deseja transmitir dados, ele habilita seu sinal RTS. Se o outro dispositivo estiver apto a receber dados, este habilitará o seu sinal CTS e a comunicação será iniciada. Se, a qualquer momento, o dispositivo receptor não estiver mais apto a receber dados, ele desabilita seu sinal CTS e automaticamente a comunicação é suspensa até nova habilitação de CTS (CANZIAN, 2006; EPUSP, 2012).

O padrão EIA-RS-232 estabelece sinais elétricos com variação máxima de $-25V$ a $+25V$. O valor 1 é representado por um sinal que esteja entre -3 e $-25V$, denominado *on* para as linhas de controle e *mark* para as linhas de dados. O valor 0 é representado por um sinal que esteja entre $+3$ e $+25V$, sendo denominado *off* para as linhas de controle e *space* para as linhas de dados. Valores entre $-3V$ e $+3V$ são inválidos. O estado de espera da linha é o estado de ON, ou seja, entre $-3V$ e $-25V$ (CANZIAN, 2006; EPUSP, 2012)..

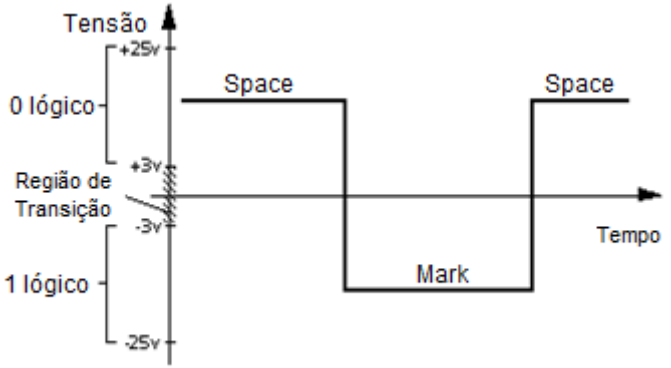

FIGURA 3 - Níveis de sinal elétrico no padrão EIA-RS-232C.

O padrão estabelece ainda que cada bloco de informação seja iniciado por um *start bit*, que é um símbolo *space*. A seguir são transmitidos sete ou oito bits de dados, iniciando pelo bit menos significativo. Após os dados, pode haver ou não um bit de paridade. O bloco de informação é terminado por um ou dois s*top bits*, que são sinais de *mark* (CANZIAN,2006; EPUSP,2012).

O tamanho mais comum de bits de dados é oito, já que desta forma um byte é transmitido a cada bloco. O bit de paridade pode ser utilizado para detecção de erros de comunicação podendo ser selecionada paridade par ou ímpar.

Na paridade par, o bit de paridade é ajustado de forma que a soma da quantidade de bits com valor lógico 1 no campo de dados mais o bit de paridade seja um número par. Na paridade ímpar busca-se obter um número ímpar na contagem de bits com valor lógico 1 (CANZIAN, 2006; EPUSP, 2012).

A taxa de transferência refere-se à velocidade com que os dados são enviados através de um canal e é medido em transições elétricas por segundo. Na norma EIA-RS-232C, ocorre uma transição de sinal por bit, e a taxa de transferência e a taxa de bit (*bit rate*) são idênticas. Nesse caso, uma taxa de 9600 *bauds* corresponde a uma transferência de 9600 bits por segundo (CANZIAN, 2006).

2.5 O protocolo de comunicação UPS Standard Protocol

Esta seção apresenta em detalhes o protocolo de comunicação UPS Standard Protocol.

2.5.1 Histórico

O protocolo de comunicação utilizado neste trabalho foi criado pela extinta companhia System Enhancement Corporation, uma fabricante de sistemas de energia situada nos Estados Unidos e que foi incorporada pela empresa American Power Conversion Corporation em 1997.

Inicialmente proposto em outubro de 1994 sob a denominação de SEC, este protocolo foi sendo constantemente atualizado até 1995, quando passou a ser denominado *UPS Standard Protocol* (SEC,1995).

O *UPS Standard Protocol* passou a ser adotado por diversos fabricantes de sistemas de energia ininterrupta, passando a ser o protocolo de fato para comunicação deste tipo de sistemas. Sua ampla utilização fez que com este protocolo fosse usado como base para o desenvolvimento da RFC1628, que descreve um conjunto padrão de variáveis para dispositivos UPS com comunicação compatível com o protocolo SNMP (*Simple Network Management Protocol*).

Nas próximas Seções deste trabalho, o protocolo será detalhado conforme SEC (1995).

2.5.2 Modelo de comunicação

O *UPS Standard Protocol* foi concebido para uma comunicação no estilo mestre / escravo. Neste protocolo, toda troca de informações é iniciada pelo dispositivo mestre, que na maioria dos casos é um computador. O dispositivo escravo, por sua vez, responde a solicitação do mestre com os dados apropriados. Apesar deste modelo, nenhuma forma de endereçamento foi utilizada, fazendo com o mestre possa comunicar-se somente com um único escravo.

Os dados trafegam entre o mestre e o escravo através de uma interface serial padrão EIA-RS-232-C (ver Seção 2.4). A comunicação serial utiliza oito bits de dados, um bit de parada e nenhum bit de paridade.

A taxa de símbolos padrão é especificado para o valor de 2400 bps (bits por segundo). Esta taxa pode ter seu valor alterado a qualquer momento através de um comando específico que será apresentado posteriormente.

As mensagens trocadas entre mestre e escravo possuem um formato padrão e todos os bytes da mensagem são codificados como caracteres ASCII. Cada mensagem é composta por quatro campos: Cabeçalho, Tipo, Tamanho e Dados.

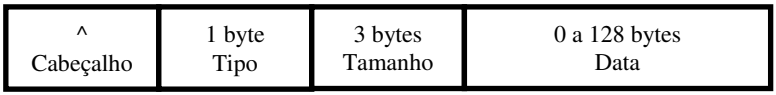

FIGURA 4 - Formato das mensagens.

O campo Cabeçalho é composto por um byte, que contem o caractere '^' (valor ASCII 0x5e), e identifica o início de uma nova mensagem. O campo Tipo é composto também por um byte e seu valor identifica o tipo da mensagem, com valores possíveis apresentados na TABELA 1.

TABELA 1- Identificadores de tipos de mensagens.

Caractere	Valor ASCII	Tipo de mensagem	Sentido de transmissão
0	0x30	Comando Rejeitado	Do escravo para o mestre
1	0x31	Comando Aceito	Do escravo para o mestre
P	0x50	Comando de Requisição de Dados	Do mestre para o escravo
S	0x53	Comando de Escrita de Dados	Do mestre para o escravo
D	0x44	Comando de Resposta de Dados	Do escravo para o mestre

O campo Tamanho é composto por três bytes e representa o número de bytes presente no campo de dados. As mensagens de comando aceito e comando rejeitado não possuem este campo.

O campo Data contém os dados da mensagem. Os dados são separados por vírgula e sua organização depende do comando que a mensagem representa.

2.5.3 Comandos

Os comandos disponíveis no protocolo podem ser divididos em dois grandes grupos: os comandos de requisição de dados e os comandos de escrita de dados. Os comandos de requisição são utilizados para a leitura das variáveis do dispositivo escravo, enquanto os comandos de escrita são utilizados para ajustar valores desse mesmo dispositivo. Cada comando é identificado como um conjunto de três bytes.

2.5.3.1 Comandos de requisição de dados

Os comandos AP1 e AP2 solicitam uma listagem dos parâmetros implementados no escravo. Através deste comando o mestre pode determinar quais comandos o escravo estará apto a responder durante a troca de mensagens. Cada parâmetro é representado por um valor inteiro no intervalo de 1 a 89. O comando AP1 solicita os parâmetros de 1 a 46, enquanto o comando AP2 solicita os parâmetros de 47 a 89. Os parâmetros são agrupados de acordo com a parte do sistema UPS que representam. A TABELA 2 apresenta os parâmetros e seus respectivos valores.

O comando NOM trata das características nominais do sistema de energia, como as tensões nominais de entrada e saída, potências nominais etc.

O comando ST1 solicita ao escravo um conjunto de parâmetros que identifica o status do carregador de baterias e valores dinâmicos como corrente e tensão.

Os parâmetros relativos à rede de entrada podem ser acessados com o comando ST2. Além das informações dinâmicas este comando recupera ainda o número de fases da rede de entrada e um contador de rede alterada que indica quantas vezes a rede esteve fora de seu valores operacionais.

O comando ST3 reúne as informações da rede de saída do *nobreak*, ou seja, da energia que efetivamente está sendo entregue à carga.

O comando ST4 solicita ao escravo um conjunto de parâmetros que identifica o status da rede de by-pass do sistema UPS.

O comando ST5 faz a leitura do estado presente em todos os alarmes disponíveis no equipamento. Um alarme está ativo quando seu valor é 1.

O comando MAN solicita ao escravo uma string de no máximo 32 bytes representando o fabricante do sistema UPS.

O comando MOD solicita ao escravo uma string de no máximo 64 bytes que representa o modelo do sistema UPS.

O comando VER solicita ao escravo uma string de no máximo 12 bytes que representa a versão de software e hardware do sistema UPS.

O comando UID solicita ao escravo uma string de no máximo 64 bytes que representa a identificação do sistema UPS. Esta identificação é definida pelo fabricante.

O comando SDA solicita ao escravo o valor do tipo de desligamento que deverá ser efetuado. O valor 1 indica que o desligamento deve ser apenas da saída, enquanto o valor 2 indica que o sistema inteiro deve ser desligado.

O comando ATR solicita ao escravo o valor do parâmetro de auto religamento. O valor 1 indica que o sistema se religará automaticamente após um comando de shutdown, enquanto que o valor 0 indica que o sistema deverá ser religado manualmente.

O comando UBR solicita ao escravo a taxa de símbolos configurada. O valor é composto por 5 bytes e os valores podem ser 1200, 2400, 4800, 9600 e 19200.

O comando STR solicita ao escravo o resultado do último autoteste executado.

2.5.3.2 Comandos de escrita de dados

Os comandos ATR, NOM, SDA, UBR e UID, apresentados na seção de comandos de requisição de dados, podem ser utilizados também como comandos de escrita de dados. Estes comandos mantêm o mesmo conjunto de parâmetros tanto para requisição como para a escrita de dados.

O comando PSD inicia um desligamento após um atraso especificado no campo de dados. Este atraso tem um tamanho máximo

de 7 bytes e é definido em segundos. Um valor de -1 indica o cancelamento de um desligamento previamente indicado.

O comando RWD inicia um desligamento imediatamente com religamento automático após um atraso especificado no campo de dados. Este atraso tem um tamanho máximo de 7 bytes e é definido em segundos.

O comando TST inicia um teste específico no sistema UPS. O tipo de teste é especificado por 1 byte no campo de dados. O valor 1 inicia um teste geral, o valor 2 inicia um teste de bateria, o valor 3 inicia um teste completo e um valor de -1 cancela o teste atual.

Após iniciado o teste, o mestre deve realizar requisições com o comando STR a fim de determinar o término do teste e seu resultado. O protocolo não define a abrangência de cada teste, ficando esta definição a cargo do fabricante do sistema UPS.

2.5.4 Exemplos de comunicação

A seguir serão apresentados alguns exemplos de troca de mensagem entre mestre e escravo.

Exemplo 1: O mestre solicita a primeira parte dos parâmetros implementados no escravo:

```
Mestre envia: ^P003AP1
      ^ - Caractere do cabeçalho
      P - Tipo = Comando de requisição
      003 - Tamanho = 3 bytes
      AP1 - Data = Comando AP1

Escravo Responde: ^D0092,5,20
      ^ - Caractere do cabeçalho
      D - Tipo = Comando de resposta
      009 - Tamanho = 9 bytes
      2,5,20  = Data:
      2 = Alarme By-pass Anormal
      5 = Alarme falha de fusível
      20 = Corrente de bateria
```

Exemplo 2: O mestre solicita o fabricante do sistema UPS:

```
Mestre envia: ^P003MAN
      ^ – Caractere do cabeçalho
      P – Tipo = Comando de requisição
      003 – Tamanho = 3 bytes
      MAN – Data = Comando MAN
Escravo Responde: ^D018EquisulGPL
      ^ – Caractere do cabeçalho
      D – Tipo = Comando de resposta
      018 – Tamanho = 18 bytes
      EquisulGPL = Data – Fabricante
```

Exemplo 3: O mestre solicita o valor do parâmetro de auto religamento, que não é implementado pelo escravo:

```
Mestre envia: ^P003STR
      ^ – Caractere do cabeçalho
      P – Tipo = Comando de requisição
      003 – Tamanho = 3 bytes
      STR – Data = Comando STR

Escravo Responde: ^0
      ^ – Caractere do cabeçalho
      0 – Tipo = Comando rejeitado
```

3. DESENVOLVIMENTO

Neste capítulo serão apresentados os detalhes de projeto do sistema desenvolvido neste trabalho. O sistema é dividido em cinco blocos funcionais: *driver* de comunicação, controle, biblioteca gráfica, display com *touch screen* e software simulador de comunicação. A Figura 5 apresenta uma visão geral da comunicação entre os blocos.

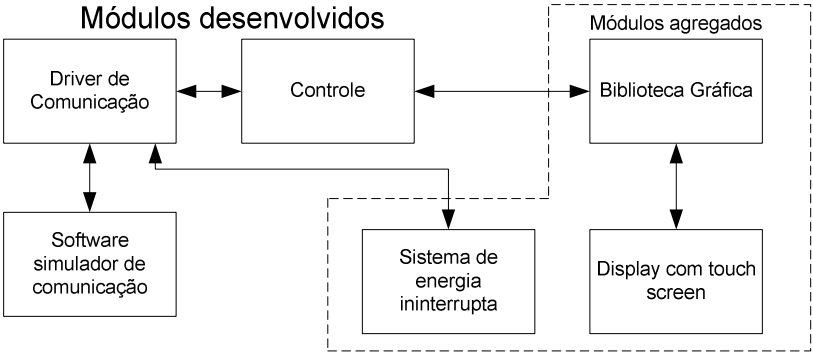

FIGURA 5 - Visão geral do sistema desenvolvido.

O bloco de controle representa o núcleo do sistema, sendo responsável por iniciar o *driver* de comunicação e receber os dados deste. Após receber os dados do *driver*, o controle atualiza as informações necessárias na tela invocando os métodos da biblioteca gráfica. O controle também recebe os eventos gerados pela camada de objetos gráficos da biblioteca gráfica, os trata e gera as alterações necessárias na interface, como, por exemplo, a troca da tela ativa.

O *driver* de comunicação é implementado por máquinas de estados finitos, sendo capaz de enviar e receber mensagens compatíveis com o protocolo *UPS Standard Protocol*.

O bloco da biblioteca gráfica representa todas as camadas da biblioteca gráfica Microchip, apresentadas na Seção 2.3.

O bloco do *display* com *touch screen* representa o hardware que exibe as imagens e reconhece os toques do usuário na tela.

O bloco software simulador de comunicação representa o software para PC desenvolvido com o intuito de possibilitar a simulação

de operação da interface gráfica sem a necessidade desta estar acoplada a um sistema de energia real. Este software permite a manipulação de todos os campos do protocolo de comunicação.

Nas próximas seções serão apresentados detalhes relevantes de cada um dos blocos que compõem o sistema.

3.1 Definição da plataforma de hardware e firmware

Para o desenvolvimento deste trabalho, o hardware necessário deve proporcionar, além do microcontrolador escolhido (PIC24FJ256DA210), comunicação serial EIA-RS-232-C e conexões com o *display* gráfico e com o *touch screen*.

Uma possibilidade seria projetar os circuitos necessários e desenvolver uma placa de circuito impresso. Pesam contra esta possibilidade o tempo disponível para realização deste trabalho, e a complexidade do desenvolvimento da placa de circuito impresso, dado a complexidade da interligação entre o microcontrolador e o display.

Optou-se, portanto, por utilizar um kit de desenvolvimento comercial. O kit selecionado foi o PIC24FJ256DA210 *Development Board*, da Microchip. Este kit atende a todos os requisitos de hardware necessários a este trabalho. A FIGURA 6 apresenta uma imagem deste kit. Este kit foi cedido pela empresa EquisulGPL.

Entre as características deste kit, as que mais contribuem para este trabalho são:

- Fácil interconexão entre microcontrolador e display;
- Conector para depuração *in-circuit*;
- Interface EIA-RS-232;
- Memória FLASH serial de 16 Mbit;
- Memória FLASH paralela de 512 Kbytes;
- Memória SRAM de 512 Kbytes.

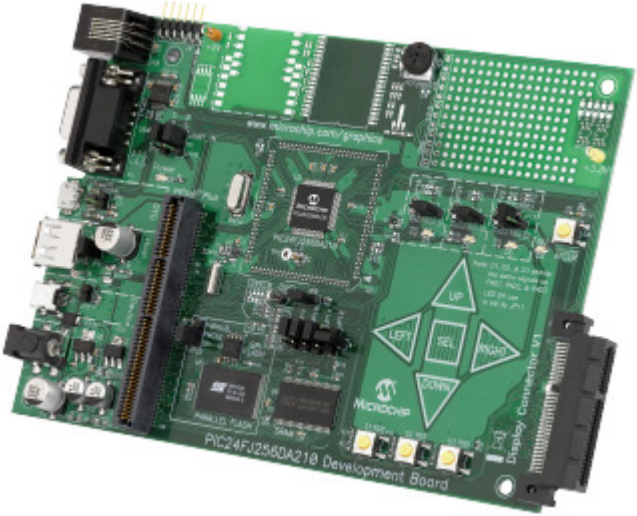

FIGURA 6 - PIC24FJ256DA210 *Development Board.*

Para a parte gráfica foi escolhido o módulo AC164127-8 da Microchip. Este módulo conta com um *display* Truly de 5,7 polegadas com resolução de 640x480. O módulo possui ainda um sensor *touch screen* resistivo de quatro vias. Está presente ainda neste módulo um controlador de *touch screen* com comunicação SPI, que não será utilizado neste trabalho, visto que o *touch screen* será controlado pelo próprio microcontrolador. A FIGURA 7 apresenta uma imagem do módulo de display.

A escolha deste display gerou a necessidade de uma alteração no hardware do kit de desenvolvimento. Como o *display* possui uma resolução de 640X480, e deseja-se usar uma profundidade de cor de 16 bits, é necessário um *frame buffer* de 600 Kbytes. Este tamanho é muito maior que a memória interna do microcontrolador, bem como é também maior que a SRAM do kit de desenvolvimento. Para superar esta limitação, a memória SRAM do kit foi alterada para uma memória de 1 Mbyte.

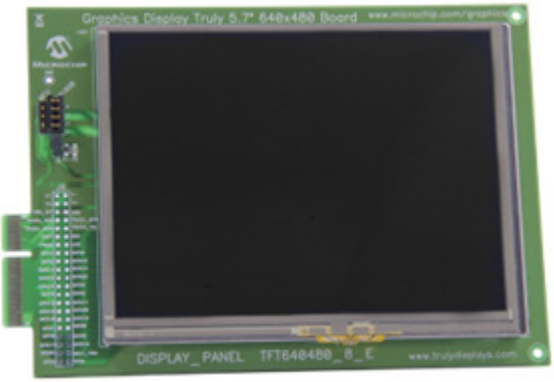

FIGURA 7 - Módulo Display 5,7 polegadas.

O firmware da interface gráfica é codificado em linguagem C, utilizando o compilador C30 da Microchip.

Para o desenvolvimento do software simulador de comunicação foi selecionada a linguagem JAVA.

3.2 O driver de comunicação do protocolo *UPS Standard protocol*

O *driver* de comunicação serial fornece dois serviços básicos. O primeiro é o envio de comandos de requisição de dados ao dispositivo escravo. O segundo é a recepção dos comandos de resposta de dados do escravo.

Estes dois serviços foram projetados como duas máquinas de estados finitos independentes.

A máquina de estado responsável pelos comandos de requisição de dados envia ciclicamente os comandos MOD, VER, ST1, ST2, ST3, ST4 e ST5, com um período de espera (*timeout*) de 250 ms entre cada comando.

Antes do envio de cada comando, esta máquina checa se o comando anterior foi respondido. Caso o comando anterior não tenha sido respondido pelo escravo, o contador de falhas é incrementado. Caso

tenha havido resposta, o contador de falhas é zerado. Se o contador de falhas ultrapassar o limite de 20 falhas, é detectada a falha de comunicação. A Figura 8 apresenta a máquina de estados de envio.

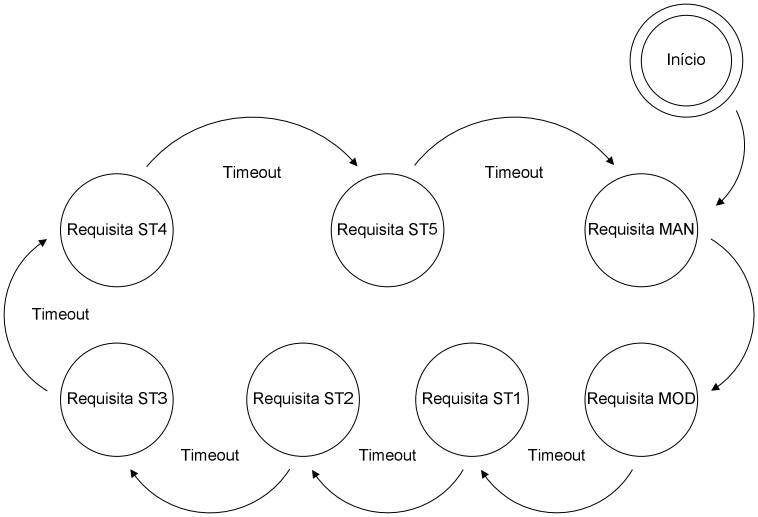

FIGURA 8 - Máquina de estados de envio.

A máquina de estados responsável pela recepção dos comandos de resposta de dados trata os bytes recebidos um a um e os decodifica de forma adequada. A máquina de estados pode voltar a seu estado inicial a qualquer momento com a recepção do caractere de cabeçalho, para evitar que mensagens recebidas incompletas tornem a máquina inoperante ou gerem falsos resultados. A figura 9 apresenta a máquina de estados de recepção.

Quando uma nova mensagem é recebida, esta é identificada e seu campo de dados é processado através de uma função que atua como um *tokenizador*, resgatando os valores delimitados por vírgula.

Pelo fato de os dados serem recebidos como caracteres ASCII no protocolo e serem utilizados também como caracteres ASCII no display, nenhum tipo de conversão é realizada nos dados.

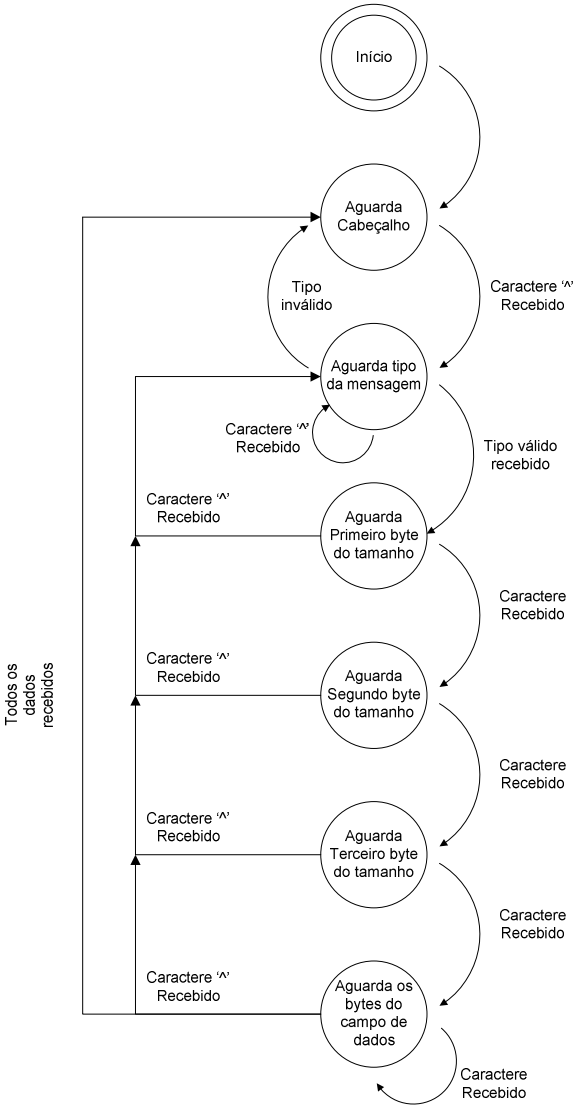

FIGURA 9 - Máquina de estados de recepção.

3.3 O projeto da interface gráfica

A interface gráfica projetada é formada por um total de sete telas: abertura, tela principal, entrada, saída, by-pass, baterias, alarmes e configurações.

3.3.1 A tela de abertura

A tela de abertura é a primeira tela a ser exibida quando o sistema é energizado. Esta tela apresenta as logomarcas da empresa, o modelo do sistema de energia ininterrupta e sua versão. A Figura 10 apresenta o protótipo de tela de abertura.

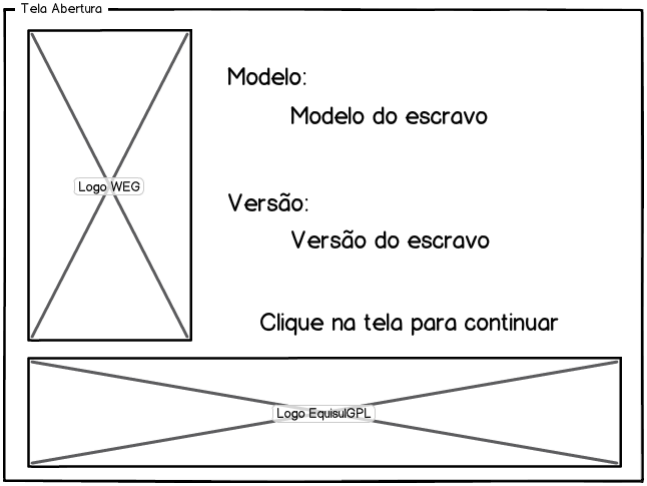

FIGURA 10 - Protótipo da tela de abertura.

3.3.2 A tela principal

Esta tela apresenta uma visão geral do funcionamento do sistema. Uma região central da tela é destinada a contar o diagrama de blocos do sistema de energia. Este diagrama de blocos deve apresentar o caminho

ativo da corrente no sistema. Para este trabalho foi escolhido um *nobreak* online de dupla conversão para criação do diagrama de blocos.

Os blocos do diagrama permitirão ainda que o usuário acesse as telas que exibirão os parâmetros dinâmicos. Por exemplo, quando o usuário tocar no bloco de entrada, será exibida a tela com os parâmetros de entrada do sistema de energia.

Na parte inferior esquerda da tela é reservada uma área para listagem dos eventos detectados. É possível exibir nesta área os oito últimos eventos ocorridos.

A tela principal contém ainda os ícones para acesso às telas de alarmes e configurações e um texto que indica o estado da comunicação.

A Figura 11 apresenta o protótipo da tela principal.

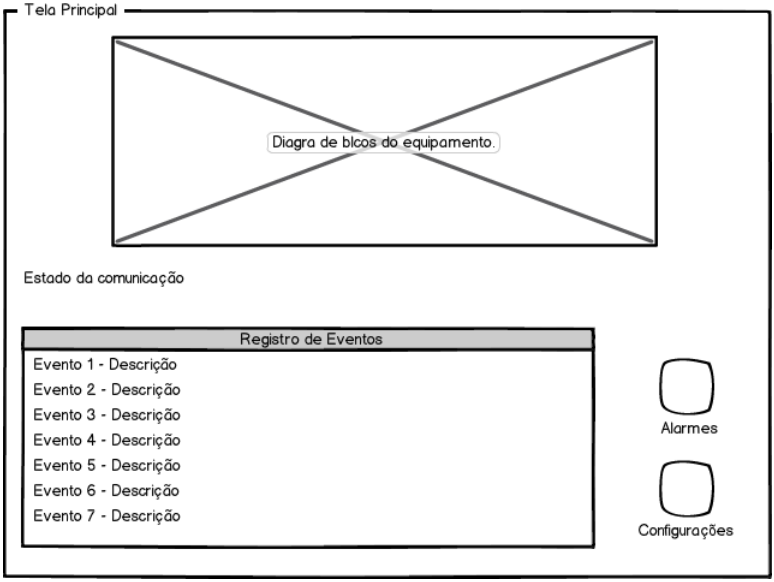

FIGURA 11- Protótipo da tela principal.

3.3.3 A tela de entrada

Esta tela é acessada através do ícone de entrada do diagrama de blocos da tela principal e apresenta os parâmetros da rede de entrada do

sistema de energia ininterrupta. Estes parâmetros são obtidos pelo comando ST2 do protocolo de comunicação. A Figura 12 apresenta o protótipo da tela de entrada.

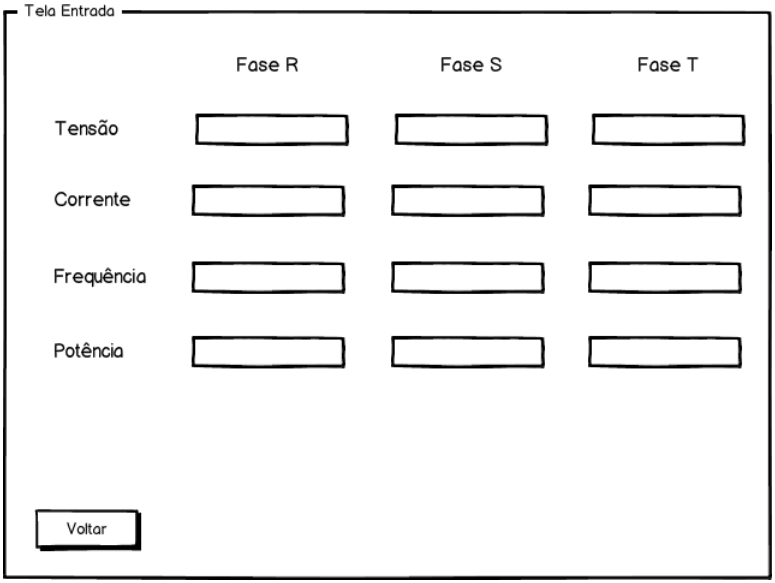

FIGURA 12- Protótipo da tela de entrada.

3.3.4 A tela de saída

Esta tela é acessada através do ícone de saída do diagrama de blocos da tela principal e apresenta os parâmetros da rede de saída do sistema de energia ininterrupta. Estes parâmetros são obtidos pelo comando ST3 do protocolo de comunicação. A Figura 13 apresenta o protótipo da tela de entrada.

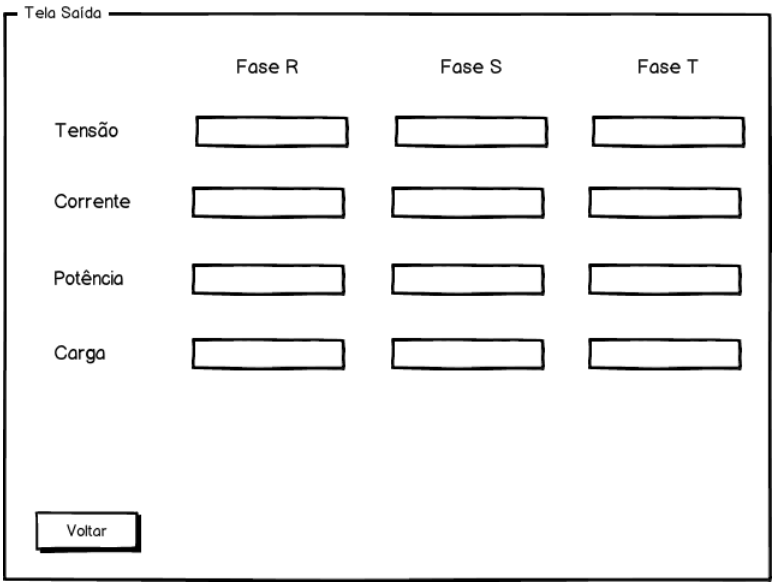

FIGURA 13- Protótipo da tela de saída.

3.3.4 A tela de by-pass

Esta tela é acessada através do ícone de by-pass do diagrama de blocos da tela principal e apresenta os parâmetros de tensão e corrente da rede de saída do sistema de energia ininterrupta. Estes parâmetros são obtidos pelo comando ST4 do protocolo de comunicação. A Figura 14 apresenta o protótipo da tela de by-pass.

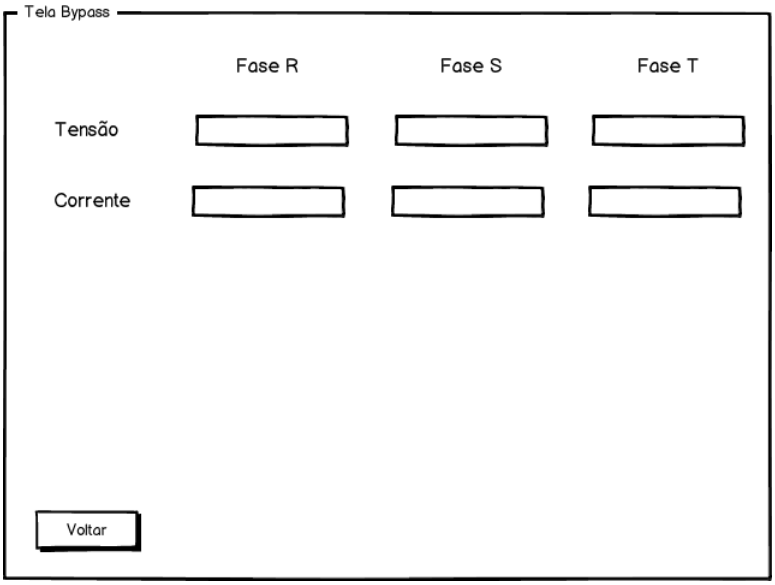

FIGURA 14- Protótipo da tela de by-pass.

3.3.5 A tela de baterias

Esta tela é acessada através do ícone de baterias do diagrama de blocos da tela principal e apresenta os parâmetros de bateria do sistema de energia ininterrupta. Estes parâmetros são obtidos pelo comando ST1 do protocolo de comunicação. A Figura 15 apresenta o protótipo da tela de baterias.

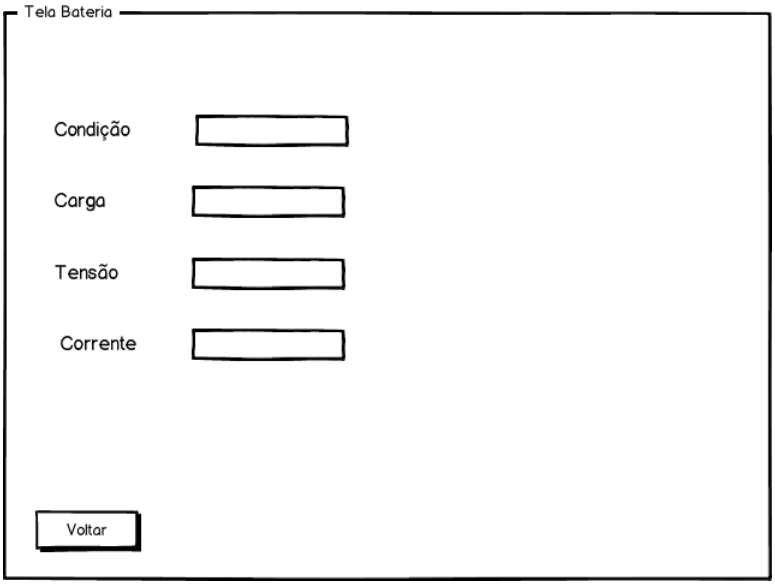

FIGURA 15- Protótipo da tela de baterias.

3.3.6 A tela de alarmes

Esta tela é acessada através do ícone de alarmes da tela principal. Nesta tela é exibida uma lista com os alarmes ativos no sistema de energia ininterrupta, sendo o protótipo apresentado na Figura 16.

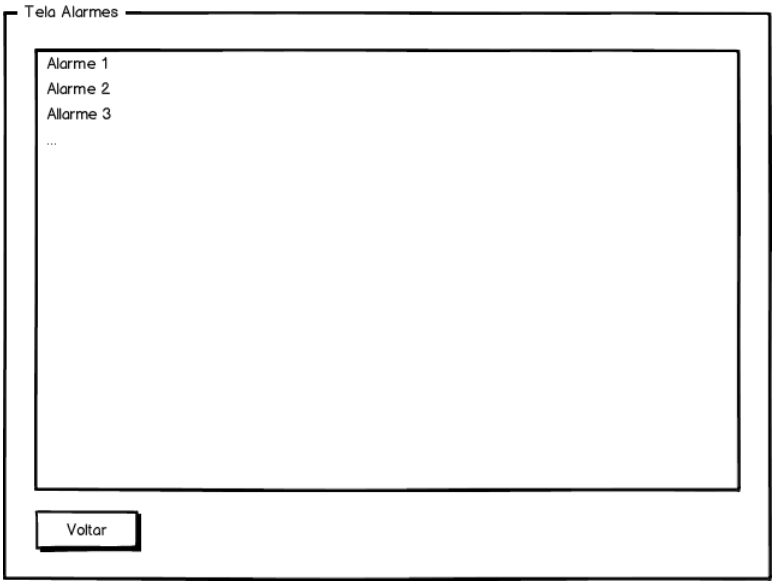

FIGURA 16- Protótipo da tela de alarmes.

3.3.7 A tela de configurações

Esta tela é acessada através do ícone de configurações da tela inicial. Nesta tela é possível realizar o ajuste do relógio interno da interface gráfica.

3.4 O módulo de controle

O módulo de controle é responsável por coordenar a troca de informações entre o *driver* de comunicação e a interface gráfica. Os dados recebidos do *driver* são tratados e então os objetos gráficos são atualizados com os novos valores. Além disso, o módulo de controle trata os eventos recebidos da biblioteca gráfica.

A figura 17 apresenta um fluxograma com o funcionamento básico do módulo de controle.

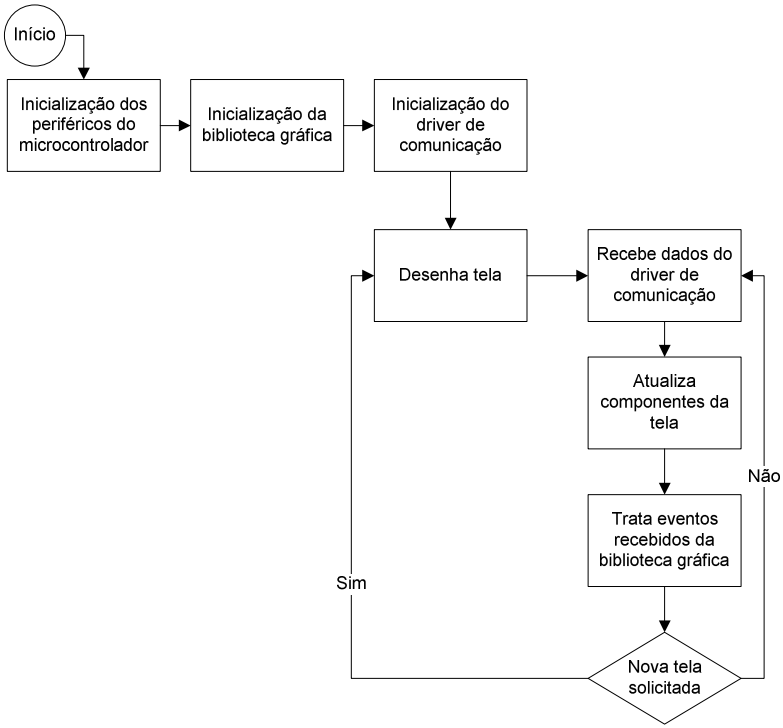

FIGURA 17 - Fluxograma do módulo de controle.

3.4.1 Detecção de eventos

O módulo de controle realiza ainda a tarefa de determinar a ocorrência de um novo evento. Para este projeto serão considerados eventos as mudanças da fonte de energia presente na saída do sistema de energia ininterrupta. Por exemplo, se o sistema passar a usar as baterias como fonte de energia para a saída, será detectado o evento de operação por baterias.

3.4 O software simulador de comunicação

O desenvolvimento deste software teve por objetivo simplificar a simulação de operação da interface gráfica desenvolvida. A utilização do software simulador permite a utilização da interface em locais em que o sistema real não esteja disponível. O Software permite ainda simular condições críticas como sobrecargas e sobretensões, que iriam requerer uma maior preparação caso fossem testadas no equipamento real.

O software simulador de comunicação foi projetado para ser executado em um computador pessoal. Implementado em linguagem Java, o software atua como escravo do protocolo *UPS Standard Protocol* tratando os comandos de requisição de dados e enviando os comandos de resposta de dados apropriados.

O software conta com uma interface gráfica na qual o usuário pode alterar os valores que serão respondidos ao mestre. Visando simular um comportamento dinâmico, o software permite responder os comandos ST1, ST2, ST3, ST4, ST5, MAN e MOD.

O software foi modelado de acordo com as técnicas de orientação a objetos, conforme apresentado na figura 18. A classe SistemaEnergia representa um sistema de energia ininterrupta e é formada pela agregação de um conjunto de classes, cada uma delas contendo os parâmetros de um dos comandos suportados. A classe SimuladorGui fornece a interface gráfica para entrada dos valores. A classe PortaSerial faz a comunicação com o hardware de RS-232 do PC, recebendo e transmitindo os dados da classe UpsProtocolParser, que acessa os valores da classe SistemaEnergia para montar as mensagens de resposta. A classe Control faz a interface entre as classes SistemaEnergia e SimuladorGUI.

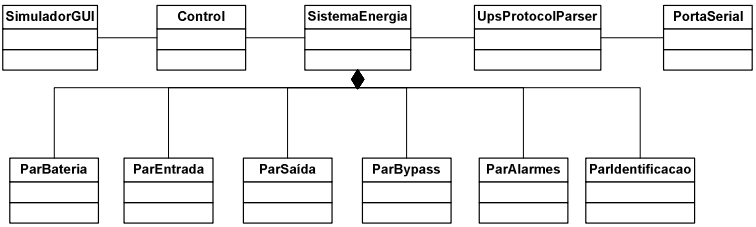

FIGURA 18 - Diagrama de classes do software de simulação de comunicação.

4. RESULTADOS

Neste Capítulo são apresentados os resultados obtidos no desenvolvimento deste trabalho. Inicialmente são apresentados as telas desenvolvidas e os resultados obtidos com o software de simulação de comunicação. Ao final são apresentados os testes executados a fim de validar o sistema desenvolvido.

4.1 Telas desenvolvidas

Nesta seção serão apresentadas as telas desenvolvidas e suas características.

4.1.1 Tela de abertura

A tela de abertura foi desenvolvida de acordo com o protótipo apresentado no Capítulo anterior. Foram utilizados as logomarcas da EquisulGPL, empresa parceira deste trabalho e também do grupo WEG, da qual faz parte a EquisulGPL. A Figura 19 apresenta a tela de abertura desenvolvida.

FIGURA 19 - Tela de abertura.

4.1.2 Tela Principal

A Figura 20 apresenta uma imagem da tela principal desenvolvida. O diagrama de blocos desenvolvido apresenta os ícones para a rede de entrada, by-pass, saída e baterias, além de representar o retificador e o inversor do *nobreak*.

As linhas que interligam os blocos tem sua cor alterada de acordo com a fonte de energia utilizada pela saída. Na figura 20 o diagrama apresenta a saída sendo alimentada pela rede de entrada.

A Figura 20 evidencia ainda a área destinada ao registro de eventos, contendo um evento registrado.

FIGURA 20- Tela principal.

Para facilitar a visualização da presença de um alarme, foi criada uma animação no ícone de alarmes. Quando nenhum alarme está presente o ícone apresenta cor verde. Quando há, ocorre uma alternância entre um ícone de cor vermelha e um ícone cercado por linhas, dando a impressão visual de que a lâmpada vermelha está piscando. A Figura 21 apresenta as três imagens que são utilizadas na composição do ícone de alarmes.

FIGURA 21 - Imagens do ícone de alarmes

Outra indicação visual utilizada na tela principal é o nível de carga da bateria. Para esta indicação foram utilizadas figuras similares a uma pilha, no sentido vertical, contendo quatro seções internas. Cada seção corresponde a 25% de carga da bateria.

A Figura 22 apresenta as quatro imagens possíveis. Quando a carga está entre 0 e 25% é exibida a imagem a, entre 26 e 50% a imagem b, entre 51 e 75% a imagem c e acima de 76% a imagem d.

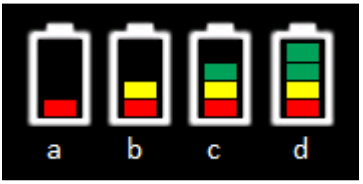

FIGURA 22 - Imagens do indicador do nível de carga da bateria.

4.1.3 Tela de alarmes

A Figura 23 apresenta uma imagem da tela de alarmes desenvolvida. A tela apresenta a listagem dos alarmes ativos no sistema de energia ininterrupta. São mostrados no máximo 14 alarmes simultaneamente.

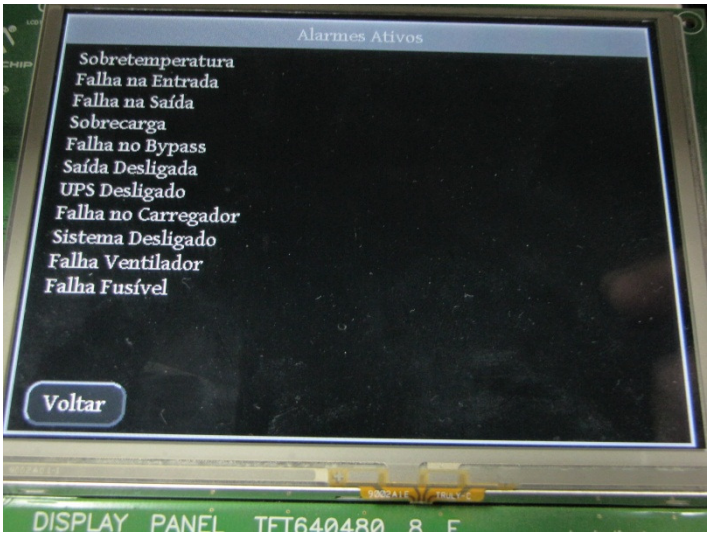

FIGURA 23 - Tela de alarmes

4.1.4 Telas de variáveis dinâmicas

As telas de entrada, saída, by-pass e baterias foram construídas segundo os protótipos apresentados no Capítulo anterior. A Figura 24 apresenta uma imagem da tela de entrada. As demais telas seguem o mesmo padrão de estilo.

FIGURA 24 - Tela de entrada

4.2 O software de simulação de comunicação

O software desenvolvido permite a manipulação de todos os campos do protocolo através de sua interface gráfica. Todos os valores inseridos pelo usuário são testados de forma a garantir que sejam válidos e estejam dentro do intervalo de dados esperado. Para os campos que possuem um conjunto determinado de valores, foram utilizados componentes do tipo caixa de combinação com todos os valores possíveis.

Pelo fato de ter sido desenvolvido em linguagem JAVA, o software é multiplaforma e foi testado em Windows 7, Mac OS X e Fedora Linux.

A figura 25 apresenta uma imagem da interface gráfica do software de simulação de comunicação.

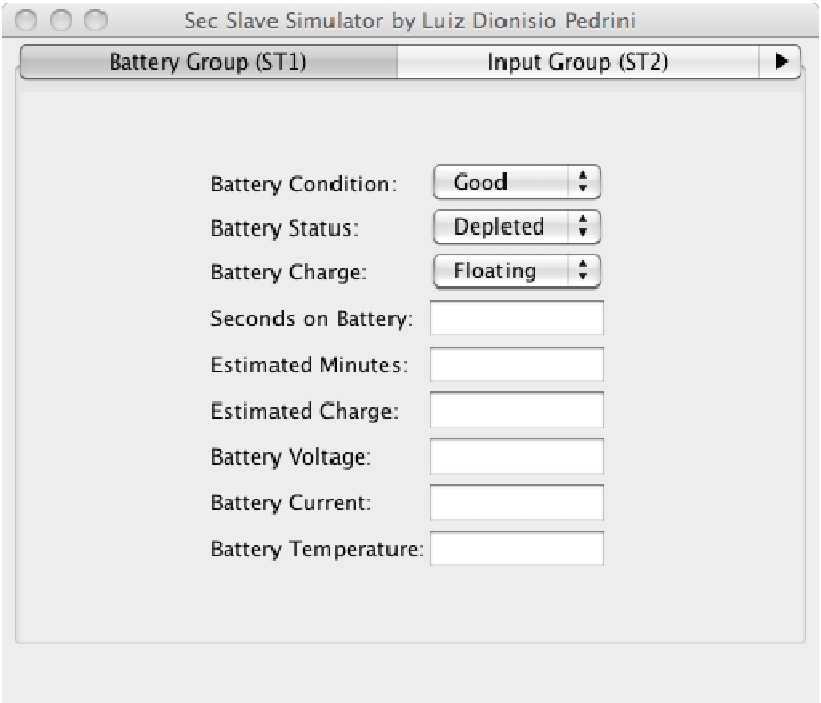

FIGURA 25 - Tela do software de simulação de comunicação

4.3 Testes

Nesta Seção serão apresentados os testes realizados de modo a verificar as funcionalidades dos módulos desenvolvidos.

4.3.1 Testes do software simulador de comunicação

Este teste tem por objetivo verificar o funcionamento do software de simulação de comunicação desenvolvido. Neste teste, conjuntos conhecidos de valores foram ajustados na interface gráfica do software de simulação de comunicação. Após isso, com o auxílio de um software de comunicação serial, foram enviados os comandos de requisição de dados e checados os comandos de resposta de dados recebidos.

4.3.1.1 Resultados obtidos

Para todos os conjuntos de dados de entrada utilizados, os comandos de reposta de dados estavam corretamente formatados, apresentando os valores corretos em todos os parâmetros.

Este resultado garante que este software pode ser utilizado para simular um sistema de energia ininterrupta, permitindo testar o funcionamento da interface sem a necessidade desta estar acoplada ao sistema real. Isto também permite simular facilmente situações críticas.

4.3.2 Testes do driver de comunicação

Para testar o *driver* de comunicação utilizou-se um software de monitoramento de tráfego da porta serial. Neste software foram obtidos os comandos de requisição de dados gerados pelo *driver*.

Através deste software também foi possível enviar comandos de resposta de dados a fim de verificar o funcionamento da máquina de estados de recepção do *driver* de comunicação.

4.3.2.1 Resultados obtidos

Foi verificado que todos os comandos de requisição estão de acordo com a especificação do protocolo. Também verificou-se que o tempo de *timeout* de 250 ms entre cada comando é respeitado.

Foi possível também validar a robustez da máquina de estados de recepção. Todas as mensagens válidas foram tratadas corretamente, enquanto as mensagens mal formadas foram descartadas.

Ficou evidenciado também que a recepção parcial de uma mensagem não faz com que a máquina deixe de identificar corretamente a próxima mensagem.

4.3.3 Teste de comunicação com o software simulador

Neste teste foi realizada a comunicação da interface gráfica com o software simulador. Este teste visou estabelecer uma comunicação com um maior volume de mensagens trocadas.

Neste teste também foi validada a detecção de perda de comunicação. Para tanto, em determinados momentos foi finalizado o

software simulador e verificada a indicação de falha de comunicação na interface.

4.3.3.1 Resultados obtidos

A comunicação entre a interface gráfica e o software de simulação foi estabelecida conforme esperado. Todas as mensagens foram corretamente identificadas e os dados presentes no software foram corretamente transmitidos à interface.

No teste de perda de comunicação, toda vez que o software de simulação foi finalizado, a falha de comunicação foi corretamente detectada. Com o software reaberto, a interface gráfica voltou a indicar comunicação normal.

4.3.4 Teste de comunicação com equipamento real

Neste teste foi realizada a comunicação da interface gráfica com um equipamento real, verificando a comunicação em condições reais de operação.

4.3.4.1 Resultados obtidos

A comunicação entre a interface gráfica e o equipamento real foi estabelecida conforme esperado. Todas as mensagens foram corretamente identificadas tanto pelo equipamento como pela interface. Em nenhum momento houve perda de comunicação.

O teste foi realizado por um período de 48 horas, e durante este período o equipamento operou pela rede, por baterias e pelo by-pass.

4.3.5 Teste de navegação entre telas

Este teste tem por objetivo verificar o funcionamento do mecanismo da troca de telas da interface gráfica. Para tanto, as telas foram acessadas em sequências aleatórias e os resultados foram verificados.

4.3.5.1 Resultados obtidos

Em todas as trocas de solicitadas a tela foi exibida corretamente. A transição entre a tela principal e as demais telas ocorre de forma agradável.

Quando a transição ocorre de qualquer tela para a tela principal, nota-se uma demora maior na construção da tela. Este fato ocorre devido à maior quantidade de itens que devem ser criados.

4.3.6 Teste do comportamento dinâmica das telas

Este teste tem por objetivo verificar o comportamento dinâmico de todas as telas. Foi verificada a atualização dos valores na tela, bem como o comportamento das animações. Durante este teste a interface gráfica permaneceu em comunicação com o software simulador, para garantir que os valores a serem exibidos fossem conhecidos.

4.3.6.1 Resultados obtidos

Em todas as telas foi possível observar que os valores foram exibidos de forma correta. A animação do ícone de alarme foi exibida corretamente na presença de um ou mais alarmes ativos. O diagrama de blocos exibiu corretamente o caminho da corrente entre os blocos.

Neste teste ficou evidenciado que os componentes do tipo lista da biblioteca gráfica, utilizados na tela principal para o registro de eventos e na tela de alarmes para listagem de alarmes ativos, têm um tempo de atualização proporcional ao número de itens presentes na lista. Isto faz com que a atualização de uma lista com vários itens seja mais lenta, fazendo com essa pareça piscar na tela.

5. CONCLUSÃO

Este trabalho detalhou a implementação de uma interface gráfica sensível ao toque que se comunica com um sistema de energia ininterrupta através do protocolo UPS Standard Protocol.

No Capítulo 2 foi apresentada uma revisão dos conceitos que fundamentam este trabalho. Foram analisadas as características de displays gráficos, da biblioteca gráfica e do microcontrolador escolhido, bem como da comunicação serial e do protocolo de comunicação.

No Capítulo 3 foram apresentados os tópicos relativos ao desenvolvimento deste trabalho. Após apresentada uma visão geral do sistema, foram tratados de detalhes específicos de cada um dos blocos desenvolvidos. Foram também apresentados os detalhes de projeto da interface gráfica e do software de simulação de comunicação.

No Capítulo 4 foram apresentados os resultados obtidos neste trabalho bem como detalhados os testes realizados a fim de verificar o funcionamento do sistema.

Este trabalho teve um resultado satisfatório, alcançando todos os objetivos propostos. A interface gráfica desenvolvida se comunica de forma satisfatória tanto com o software de simulação quanto com o equipamento real.

Vale destacar que a utilização da biblioteca gráfica Microchip diminuiu consideravelmente o tempo necessário ao desenvolvimento deste projeto.

Pode-se dizer que a contribuição deste trabalho é o detalhamento da construção da interface gráfica. O conhecimento deste trabalho pode servir de base para produção de novos trabalhos correlatos.

5.1 Trabalhos futuros

Como sugestões de trabalhos futuros, podemos considerar:

- acrescentar os comandos de escrita de dados, permitindo que sejam realizados testes no sistema de energia ininterrupta;
- criar novos drives de comunicação para outros protocolos;
- desenvolvimento da placa de circuito impresso;
- testar display de outros tipos, tamanhos e resoluções;

- realizar testes de usabilidade;
- realizar testes de compatibilidade eletromagnética,

REFERÊNCIAS

CANZIAN, Edmur. **Comunicação serial – RS232.** 2006. Disponível em <http://www.capriconsultorios.com/Aula4-Comun_serial.pdf> Acessado em: 24 de novembro de 2012.

DOWNS, Rick. **Using resistive touch screens for human/machine interface**. 2005. Disponível em: <http://www.kltouch.com/soft/tuzhi/zl/ti.pdf> Acessado em: 24 de novembro de 2012

EQUISULGPL. **Sistemas de energia ininterrupta**. 2012. Disponível em: <http://ecatalog.weg.net/files/wegnet/WEG-critical-power-sistemas-de-energia-ininterrupta-50030513-catalogo-portugues-br.pdf>. Acessado em: 24 de novembro de 2012.

EPUSP. **Comunicação serial assíncrona.** 2012. Disponível em <http://www.pcs.usp.br/~labdig/pdffiles_2012/tx_e_rx_as.pdf> Acessado em: 24 de novembro de 2012.

FUJITSU. **Fundamentals of Liquid Crystal Displays** – How they work and what they do. 2006. Disponível em: <http://www.fujitsu.com/downloads/MICRO/fma/pdf/LCD_Backgrounder.pdf> Acessado em: 24 de novembro de 2012.

GUEDES, Gildásio. **Interface humano computador.** 1. ed. Teresina: Editora da UFPI, 2009.

HOYE, Timothy; KOZAC, Jospeh. TOUCH SCREEN. **A pressing technology.** 2010. Disponível em : <http://136.142.82.187/eng12/history/2010/pdf/1118.pdf > Acessado em: 24 de novembro de 2012.

MICROCHIP. **Graphics Controller Module (GFX)**. 2009. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39731a.pdf> Acessado em: 24 de novembro de 2012.

MICROCHIP. **Developing Embedded Graphics Applications using PIC Microcontrollers with Integrated Graphics Controller**. 2011. Disponível em: <http://ww1.microchip.com/downloads/en/appnotes/01368a.pdf> Acessado em: 24 de novembro de 2012.

MICROCHIP. **Graphics Library Help**. 2011b. Disponível em: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en547784> Acessado em: 24 de novembro de 2012.

MICROCHIP. **Offline UPS Reference Design Using the dsPIC DSC**. 2011c. Disponível em: <http://ww1.microchip.com/downloads/en/AppNotes/01279B.pdf> A Acessado em: 24 de novembro de 2012.

MICROCHIP. **PIC24FJ256DA210 Family Data Sheet**. 2010. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39969b.pdf> Acessado em: 24 de novembro de 2012.

MIGUEL, Jair D.; RUFINO, Sandra. **O sistema de cores em computação gráfica**. 2010 Disponível em <http://bt.fatecsp.br/system/articles/119/original/trabalho9.pdf> Acessado em: 24 de novembro de 2012.

NXP. **Introduction to graphics and LCD technologies**. 2011. Disponível em: <http://ics.nxp.com/literature/presentations/microcontrollers/pdf/graphics.lcd.technologies.pdf> Acessado em: 24 de novembro de 2012.

ORAN, Andrew; ROTH, Vince; **Color space basics.** 2012. Disponível em: <http://www.amiaconference.com/techrev/V12-05/papers/colorspace.pdf> Acessado em: 24 de novembro de 2012.

PACALE, Danny. **A review of RGB color spaces.** 2003. Disponível em:

<http://www.babelcolor.com/download/A%20review%20of%20RGB%20color%20spaces.pdf> Acessado em: 24 de novembro de 2012.

RAMOS, André. **Fisiologia da visão**. 2006. Disponível em: <http://wwwusers.rdc.puc-rio.br/imago/site/semiotica/producao/ramos-final.pdf> Acessado em: 24 de novembro de 2012.

STOLFI, Guido. **Percepção visual humana.** 2008. Disponível em: <http://www.lcs.poli.usp.br/~gstolfi/mack/Ap2_PercepVisual_M8.pdf> Acessado em: 24 de novembro de 2012.

SEC, System Enhancement Corporation. **UPS Standard Protocol.** 1995.

APÊNDICES

APÊNDICE A – Parâmetros dos comandos do protocolo UPS Standard Protocol

Neste apêndice é apresentado um detalhamento dos parâmetros de cada um dos comandos do protocolo *UPS Standard Protocol*.

A Tabela 2 apresenta os identificadores que podem ser recebidos como resposta aos comandos AP1 e AP2.

TABELA 2 - Identificadores dos parâmetros

Valor	Parâmetro	Grupo
1	Aguardando energia	Alarmes
2	By-pass anormal	Alarmes
3	Falha no carregador	Alarmes
4	Falha de ventilador	Alarmes
5	Falha de fusível	Alarmes
6	Falha geral	Alarmes
7	Entrada Anormal	Alarmes
8	Saída Anormal	Alarmes
9	Saída desligada	Alarmes
10	Sobrecarga	Alarmes
11	Desligamento iminente	Alarmes
12	Desligamento pendente	Alarmes
13	Sistema desligado	Alarmes
14	Sobretemperatura	Alarmes
15	UPS desligado	Alarmes
16	Alarme Audível	Nominal
17	Auto religamento	Configuração
18	Modo de carga de bateria	Bateria
19	Condição das baterias	Bateria
20	Corrente de bateria	Bateria
21	Data de instalação de baterias	Nominal
22	Estado das baterias	Bateria
23	Temperatura das baterias	Bateria
24	Tensão de bateria	Bateria
25	Corrente da fase 1 de by-pass	By-pass
26	Corrente da fase 2 de by-pass	By-pass
27	Corrente da fase 3 de by-pass	By-pass

28	Frequência do by-pass	By-pass
29	Numero de fases de by-pass	By-pass
30	Potência fase 1 de by-pass	By-pass
31	Potência fase 2 de by-pass	By-pass
32	Potência fase 3 de by-pass	By-pass
33	Tensão da fase 1 de by-pass	By-pass
34	Tensão da fase 2 de by-pass	By-pass
35	Tensão da fase 2 de by-pass	By-pass
36	Carga estimada de bateria	Bateria
37	Minutos restantes estimados	Bateria
38	Limite superior de tensão para transferência	Nominal
39	Identificação (UID)	Identificação
40	Corrente da fase 1 de entrada	Entrada
41	Corrente da fase 2 de entrada	Entrada
42	Corrente da fase 3 de entrada	Entrada
43	Frequência da fase 1 de entrada	Entrada
44	Frequência da fase 2 de entrada	Entrada
45	Frequência da fase 3 de entrada	Entrada
46	Contador de rede alterada	Entrada
47	Número de fases de entrada	Entrada
48	Potência fase 1 de entrada	Entrada
49	Potência fase 2 de entrada	Entrada
50	Potência fase 3 de entrada	Entrada
51	Tensão da fase 1 de entrada	Entrada
52	Tensão da fase 2 de entrada	Entrada
53	Tensão da fase 3 de entrada	Entrada
54	Limite inferior de tensão para transferência	Nominal
55	Fabricante	Identificação
56	Modelo	Identificação
57	Validade das baterias	Nominal
58	Frequência nominal de entrada	Nominal
59	Tensão nominal de entrada	Nominal
60	Tempo em bateria baixa	Nominal
61	Frequência Nominal de Saída	Nominal
62	Potência Ativa Nominal	Nominal
63	Tensão Nominal de Saída	Nominal

64	Potência Aparente Nominal	Nominal
65	Corrente da fase 1 de saída	Saída
66	Corrente da fase 2 de saída	Saída
67	Corrente da fase 3 de saída	Saída
68	Frequência de saída	Saída
69	Carga da fase 1 de saída	Saída
70	Carga da fase 2 de saída	Saída
71	Carga da fase 3 de saída	Saída
72	Numero de fases de saída	Saída
73	Potência fase 1 de saída	Saída
74	Potência fase 2 de saída	Saída
75	Potência fase 3 de saída	Saída
76	Fonte de energia da saída	Saída
77	Tensão da fase 1 de saída	Saída
78	Tensão da fase 2 de saída	Saída
79	Tensão da fase 3 de saída	Saída
80	Desligamento com religamento	Controle
81	Segundos em bateria (RWD)	Bateria
82	Tipo de desligamento	Controle
83	Desligamento com atraso (PSD)	Controle
84	Versão	Identificação
85	Detalhes do resultado do auto teste	Teste
86	Resultado do último autoteste	Teste
87	Tipo de teste (TST)	Teste
89	Taxa de símbolos	Configuração

Os valores recebidos como resposta ao comando NOM são apresentados na Tabela 3.

TABELA 3 - Parâmetros do comando NOM

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Tensão nominal de entrada	3	Volts
Frequência nominal de entrada	3	0,1 Hertz
Tensão Nominal de Saída	3	Volts
Frequência Nominal de Saída	3	0,1 Hertz

Potência Aparente Nominal	5	Volt-Ampére (VA)
Potência Ativa Nominal	5	Watts
Tempo em bateria baixa	2	Minutos
Alarme Audível	1	1 = Desabilitado 2 = Habilitado 3 = Mudo 4 = Desabilitado até ocorrer bateria baixa
Limite inferior de tensão para transferência	3	Volts
Limite superior de tensão para transferência	3	Volts
Data de instalação de baterias	8	mmddaaaa mm = mês dd = dia aaaa = ano
Validade das baterias	5	Dias

A Tabela 4 apresenta os parâmetros que compõem a resposta ao comando ST1.

TABELA 4 - Parâmetros do comando ST1

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Condição das baterias	1	0 = Boa 1 = Fraca 3 = Substituir
Estado das baterias	1	0 = Bateria normal 1 = Bateria baixa 2 = Batteria esgotada.
Modo de carga de bateria	1	0 = Flutuação 1 = Carga 2 = Repouso 3 = Descarga
Segundos em bateria	5	Segundos
Minutos restantes estimados	3	Minutos
Carga estimada de bateria	3	Percentual
Tensão de bateria	4	0,1 Volt

Corrente de bateria	4	0,1 Ampére
Temperatura das baterias	2	Graus Celsius

Os parâmetros do comando ST2 são apresentados na Tabela 5.

TABELA 5 - Parâmetros do comando ST2

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Contador de rede alterada	3	Inteiro
Número de fases de entrada	1	Inteiro (1 a 3)
Frequência da fase 1 de entrada	3	0,1 Hertz
Tensão da fase 1 de entrada	4	0,1 Volts
Corrente da fase 1 de entrada	4	0,1 Ampére
Potência fase 1 de entrada	5	Watts
Frequência da fase 2 de entrada	3	0,1 Hertz
Tensão da fase 2 de entrada	4	0,1 Volts
Corrente da fase 2 de entrada	4	0,1 Ampére
Potência fase 2 de entrada	5	Watts
Frequência da fase 3 de entrada	3	0,1 Hertz
Tensão da fase 3 de entrada	4	0,1 Volts
Corrente da fase 3 de entrada	4	0,1 Ampére
Potência fase 3 de entrada	5	Watts

A Tabela 6 apresenta os parâmetros do comando ST3.

TABELA 6 - Parâmetros do comando ST3

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Fonte de energia da saída	1	0 = Rede 1 = Bateria 2 = By-pass 3 = Redutor 4 = Elevador 5 = Outro

Frequência de saída	3	0.1 Hertz
Numero de fases de saída	1	Inteiro (1 a 3)
Tensão da fase 1 de saída	4	0,1 Volts
Corrente da fase 1 de saída	4	0,1 Ampére
Potência fase 1 de saída	5	Watts
Carga da fase 1 de saída	3	Percentual
Tensão da fase 2 de saída	4	0,1 Volts
Corrente da fase 2 de saída	4	0,1 Ampére
Potência fase 2 de saída	5	Watts
Carga da fase 2 de saída	3	Percentual
Tensão da fase 3 de saída	4	0,1 Volts
Corrente da fase 3 de saída	4	0,1 Ampére
Potência fase 3 de saída	5	Watts
Carga da fase 3	3	Percentual

A Tabela 4 apresenta os parâmetros que compõem a resposta ao comando ST4.

TABELA 7 - Parâmetros do comando ST4

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Frequência do by-pass	3	0,1 Hertz
Numero de fases de by-pass	1	Inteiro (1 a 3)
Tensão da fase 1 de by-pass	4	0,1 Volts
Corrente da fase 1 de by-pass	4	0,1 Ampére
Potência fase 1 de by-pass	5	Watts
Tensão da fase 2 de by-pass	4	0,1 Volts
Corrente da fase 2 de by-pass	4	0,1 Ampére
Potência fase 2 de by-pass	5	Watts
Tensão da fase 3 de by-pass	4	0,1 Volts
Corrente da fase 3 de by-pass	4	0,1 Ampére
Potência da fase 3 de by-pass	5	Watts

Na tabela 8 são apresentados os alarmes especificados pelo protocolo. Estes alarmes compõem a resposta ao comando ST5.

TABELA 8 - Parâmetros do comando ST5

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Sobretemperatura	1	1 = alarme presente 0 = alarme ausente
Entrada Anormal	1	1 = alarme presente 0 = alarme ausente
Saída Anormal	1	1 = alarme presente 0 = alarme ausente
Sobrecarga	1	1 = alarme presente 0 = alarme ausente
By-pass anormal	1	1 = alarme presente 0 = alarme ausente
Saída desligada	1	1 = alarme presente 0 = alarme ausente
UPS desligado	1	1 = alarme presente 0 = alarme ausente
Falha no carregador	1	1 = alarme presente 0 = alarme ausente
Sistema desligado	1	1 = alarme presente 0 = alarme ausente
Falha de ventilador	1	1 = alarme presente 0 = alarme ausente
Falha de fusível	1	1 = alarme presente 0 = alarme ausente
Falha geral	1	1 = alarme presente 0 = alarme ausente
Aguardando energia	1	1 = alarme presente 0 = alarme ausente
Desligamento pendente	1	1 = alarme presente 0 = alarme ausente
Desligamento iminente	1	1 = alarme presente 0 = alarme ausente

Por fim, a Tabela 9 apresenta os possíveis valores retornados ao comando STR.

TABELA 9 - Parâmetros do comando STR

Nome do Parâmetro	Tamanho máximo	Unidade ou valor
Resultado do último autoteste	1	0 = Nenhum teste realizado 1 = Passou no teste 2 = Teste em execução 3 = Teste geral falhou 4 = Teste de bateria falhou 5 = Teste completo falhou
Detalhes do resultado do autoteste	64	Texto com mais detalhes sobre o resultado do teste.