

DAS Departamento de Automação e Sistemas
CTC **Centro Tecnológico**
UFSC Universidade Federal de Santa Catarina

Desenvolvimento de um Sistema de Aquisição de Imagens Remotas como Suporte de Comunicação Visual do Projeto SIORE

*Monografia submetida à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:*

EEL 5901: Projeto de Fim de Curso

Fabrizio Carlo Mezzari

Florianópolis, maio de 1999.

Desenvolvimento de um Sistema de Aquisição de Imagens Remotas como Suporte de Comunicação Visual do Projeto SIORE

Fabrizio Carlo Mezzari

Esta monografia foi julgada no contexto da disciplina
EEL 5901: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação Industri

Banca Examinadora:

Eng. Guido Garcia D'Angelo
Orientador da Empresa

Prof. Marcelo Ricardo Stemmer
Orientador do Curso

Prof. Augusto Humberto Bruciapaglia
Responsável pela disciplina e Coordenador do Curso

Prof. Werner Kraus Júnior
Avaliador

Aluno Carlos Aurélio Pezzota
Debatedor 1

Aluno César Henrique F. Amendola
Debatedor 2

Agradecimentos

Agradeço aos meus pais e irmãs, pelo apoio, dedicação e amor que sempre me ofereceram, e que assim como eu, estão muito felizes por eu estar concluindo o curso de Engenharia de Controle e Automação e o curso de Administração.

Da mesma forma, agradeço à Gisele Simas e sua família, por se fazerem presentes em minha vida e terem me acompanhado durante a fase de conclusão do curso.

Agradeço a colaboração do Engenheiro Guido Garcia D'Angelo, por me orientado no Projeto de Fim de Curso, e a todos os funcionários da Telcel, que testemunharam, participaram e me auxiliaram no desenvolvimento deste projeto.

Agradeço à Universidade Federal de Santa Catarina, que me ofereceu a oportunidade da realização do Curso, pelo qual me proponho realizar profissionalmente. Em especial, agradeço ao Professor Marcelo Ricardo Stemmer, que me auxiliou no desenvolvimento do projeto e na redação desta monografia, ao Coordenador do Curso, Augusto Humberto Bruciapaglia, por ter me instruído durante os meses de desenvolvimento do projeto, e a todos os professores, que me deram a base de conhecimento para que este trabalho fosse desenvolvido dentro dos padrões de qualidade e funcionalidade.

Resumo

Essa monografia descreve o desenvolvimento de um sistema de aquisição de imagens remotas, que tem como objetivo a comunicação visual e manutenção da segurança das centrais telefônicas, auxiliando o sistema de Tele-Supervisão, conhecido como SIORE – Sistema Integrado de Operações Remotas.

Esse sistema de comunicação visual e segurança incorpora funcionalidades como a aquisição de imagens, o seu tratamento e visualização em um ambiente gráfico propício, além de características voltadas à segurança, como um sistema detector de movimentos e um sistema rastreador de movimentos.

Na sequência deste trabalho, os subsistemas componentes serão descritos detalhadamente e funcionalmente, apresentando-se o equipamento utilizado, o sistema operacional e o ambiente gráfico onde foi gerado.

Como fruto do trabalho, a TELESC irá dispor de uma ferramenta de comunicação e segurança que irá otimizar a comunicação entre setores, melhorar a segurança e propiciar novas utilizações para o projeto desenvolvido.

Palavras-chave: Câmera Digital, Comunicação Visual, Gerenciamento da Segurança.

Abstract

This monograph describes the development of remote acquisition image system, for accomplishment of visual communication and security maintenance of telephonic central, assisting the supervision system known as SIORE – Remote Operation Integrated System.

This visual communication and security system has functional characteristics as image acquisition, its processing and visualization in a protected graphic environment, further security characteristics, as a detection movement system and a tracking movement system.

In the course of this work, these subsystems components will be described detailed and functionally, showing and explaining the devices utilized, the operational system and the graphic environment where it has been generated.

As consequence of this work, TELESC will have a device communication and security, which will improve the communication between sectors, improve the security and propitious new utilities for this developed project.

Keywords: Digital Camera, Visual Communication, Security Management.

Índice

Capítulo 1 – Introdução	12
1.1 Abordagem Inicial.....	12
1.2 Metodologia.....	14
1.3 Sobre o Conteúdo.....	14
Capítulo 2 – A Estrutura da Telesc e o Projeto SIORE	16
2.1 Introdução.....	16
2.2 Redes e Centrais Telefônicas.....	16
2.2.1 Rede Telefônica.....	16
2.2.2 Central Telefônica.....	17
2.3 A Tele-Supervisão.....	19
2.4 A Organização da Telesc.....	19
2.5 Sistemas de Supervisão da Telesc.....	20
2.6 O SIORE.....	20
2.7 Conclusões.....	22
Capítulo 3 – Plataformas de Desenvolvimento do Projeto	23
3.1 Introdução.....	23
3.2 Sistema Operacional: Linux.....	23
3.3. Linguagem de Programação: C.....	24
3.4 O Sistema X Windows.....	25
3.4.1 O Modelo Cliente-Servidor.....	26
3.4.2. Gerenciador de Janelas.....	26
3.4.3. Eventos.....	27
3.4.4. Arquitetura do Software do Sistema X Window.....	27
3.5 Xlib.....	28
3.6 Introdução ao X Graphics.....	29
3.6.1 Pixels e Cores.....	29
3.6.2 Pixels e Planos.....	30
3.6.3 Desenhos e Contexto Gráfico.....	31
3.7 Características das Janelas.....	32
3.7.1 Intensidade, Aparência e Identificação.....	32
3.8 Estruturas de Imagem: XImage.....	33
3.8.1 Manipulação de Imagens.....	33

3.9 XForms	35
3.9.1 Rotinas de Chamada dos Comandos da Interface	37
3.10 MakeFile	38
3.10.1 Compilação com Vários Arquivos	39
3.10.2 A Compilação Separada.....	39
3.10.3 Dependências	40
3.10.4 Como Funciona	41
3.10.5 O Makefile	41
3.11 Conclusões.....	42
Capítulo 4 – A Câmera Digital – Connectix QuickCam	43
4.1 Introdução.....	43
4.2 Descrição Física e Princípio de Funcionamento	43
4.2.1 Hardware QuickCam	43
4.2.2 Operação do CCD.....	44
4.3 Conexões Físicas e Elétricas.....	46
4.3.1 Porta Paralela	46
4.4. Interface do Software.....	50
4.4.1 Software de Interface da Porta Paralela	50
4.4.2 Protocolo de Comunicação	53
4.4.3 Operação da QuickCam.....	67
4.5 Conclusões.....	79
Capítulo 5 – Tratamento e Processamento da Imagem – Driver de Vídeo....	80
5.1 Introdução.....	80
5.2 Procedimento do Tratamento de Imagem.....	80
5.2.1 Display	80
5.2.2 Intensidade (Depth).....	81
5.2.3 Tela (Screen)	81
5.2.4 Identificação da Janela (Window ID)	82
5.2.5 Contexto Gráfico ..	82
5.2.6 Mapa de Cores (Colormap).....	83
5.2.7 Criação da Paleta de Cores – Células de Cores	85
5.2.8 Captura da Imagem.....	85
5.2.9 Manipulação da Imagem	87
5.2.10 Visualização da Imagem Processada.....	87
5.3 Conclusões.....	88

Capítulo 6 – Aceleração do Processamento da Imagem	90
6.1 Introdução.....	90
6.2 Segmento de Memória Compartilhada.....	90
6.3 Procedimento de Utilização da Memória Compartilhada.....	90
6.3.1 Criação da Estrutura de Memória Compartilhada.....	92
6.3.2 Criação do Segmento de Memória Compartilhada	92
6.3.3 Informar o Servidor Sobre o Segmento de Memória.....	93
6.3.4 Utilização da Memória Compartilhada de Imagens X.....	95
6.4 Conclusões.....	96
Capítulo 7 – Interface Gráfica e Funções Desenvolvidas	97
7.1 Introdução.....	97
7.2 Procedimento de Inicialização do Software.....	97
7.3 Funções Controladas pela Interface	99
7.3.1 Dimensão da Tela	99
7.3.2 Qualidade da Imagem	100
7.3.3 Zoom.....	101
7.3.4 Modo de Transmissão.....	101
7.3.5 Motor Remoto	102
7.3.6 Ajuste de Imagem	108
7.3.7 AutoBrilho	109
7.3.8 Detecção.....	111
7.3.9 Fator Detecção.....	113
7.3.10 Rastreador	114
7.3.11 Congelamento da Imagem	117
7.3.12 Armazenagem da Imagem	117
7.3.13 Caixa de Ajuda: “Sobre o Sistema”	118
7.3.14 Bip do Alarme de Detecção.....	119
7.3.15 Relógio.....	119
7.3.16 Velocidade da Imagem.....	119
7.3.17 Saída do Sistema	120
7.4 Conclusões.....	121
Capítulo 8 – Conclusões e Perspectivas	122
Bibliografia	125

Índice de Figuras

Figura 3.1 – Arquitetura do Software do Sistema X Window	28
Figura 3.2 – Clientes e Servidores	29
Figura 3.3 – Mapeamento do Pixel em um Mapa de Cores	31
Figura 3.4 – Compilação com Vários Arquivos	39
Figura 3.5 – A Compilação Separada	39
Figura 3.6 – Dependência Gráfica	40
Figura 3.7 – Compilação e Dependência Gráfica	40
Figura 3.8 – Tradução da Dependência Gráfica	42
Figura 4.1 – Dimensões da Imagem do CCD	44
Figura 4.2 – Modelo do Ciclo de Comando	55
Figura 4.3 – Modelo do Ciclo de Envio de Comando e Parâmetro	57
Figura 4.4 – Modelo do Ciclo de Envio dos Dados de Vídeo	62
Figura 4.5 – Fim da Transmissão dos Dados de Vídeo	62
Figura 4.6 – Processo de Mudança da Porta Paralela	64
Figura 4.7 – Transferência de 12 Bits	65
Figura 4.8 – Fim da Transmissão dos Dados de Vídeo	66
Figura 4.9 – Escolha da Região de Captura da Imagem	71
Figura 4.10 – Exemplo do Cálculo de TransferPerLine	74
Figura 4.11 – Erro na Configuração de TransferPerLine	76
Figura 4.12 – Modo de Compactação para Modo Nybble	78
Figura 4.13 – Modo de Compactação para Modo Byte	79
Figura 7.1 – Tela de Abertura do Software	98
Figura 7.2 – Interface Gráfica do Sistema	99
Figura 7.3 – Escolha da Dimensão da Tela	100
Figura 7.4 – Escolha da Resolução da Imagem	100
Figura 7.5 – Escolha do Zoom	101
Figura 7.6 – Escolha do Modo de Transmissão	101
Figura 7.7 – Desativação do Modo Bidirecional	102
Figura 7.8 – Controle do Motor Remoto	102
Figura 7.9 – Porta Paralela	104
Figura 7.10 – Diagrama de Blocos do Controlador do Motor de Passo	105
Figura 7.11 – Diagrama do Circuito Lógico Controlador do Motor de Passo	105
Figura 7.12 – Trem de Pulsos Proveniente da Paralela para Controle do Motor	106
Figura 7.13 – Trem de Pulsos Invertidos para Inserção no Circuito Lógico	106

Figura 7.14 – Circuito Amplificador Excitador do Motor de Passo	108
Figura 7.15 – Caixa de Ajuste de Imagem e Botão de Ativação	109
Figura 7.16 – Ativação do AutoBrilho	111
Figura 7.17 - Desativar e Ativar Função Detecção	112
Figura 7.18 – Alarme Ativado	112
Figura 7.19 – Caixa de Escolha da Sensibilidade e Botão de Ativação	114
Figura 7.20 – Estados do Rastreador	115
Figura 7.21 – Estados de Funcionamento da Função Congelamento	117
Figura 7.22 – Função Fotografia e Caixa de Escolha de nome para arquivo.....	118
Figura 7.23 – Caixa de Ajuda do Sistema	118
Figura 7.24 – Ativar e Desativar Bip do Alarme	119
Figura 7.25 – Caixa de Saída do Sistema	120
Figura 7.26 – Caixa de Indicação de Retorno da Câmera à Posição Original .	121

Índice de Tabelas

Tabela 4.1 – Modo de Transmissão Unidirecional e Bidirecional.....	49
Tabela 4.2 – Registrador de Dados.....	52
Tabela 4.3 – Registrador de Estado.....	52
Tabela 4.4 – Registrador de Controle.....	53
Tabela 4.5 – Configuração do Comando SendVideoFrame	77
Tabela 7.1 – Estados do Motor com Relação aos Pinos 1 e 14.....	105
Tabela 7.2 – Estados de Saída do Contador e do Circuito: sentido anti-horário	107
Tabela 7.3 – Estados de Saída do Contador e do Circuito: sentido horário.....	107

CAPÍTULO 1

Introdução

1.1. Abordagem Inicial

A Telesc - Telecomunicações Santa Catarina - após o processo de privatização, tornou-se pertencente ao grupo Itália Telecom, Fundos de Pensão do Banco do Brasil, Banco Oportunidade e Fundos de Pensão Sistel, sendo detida a concessão dos direitos de exploração dos serviços de telecomunicações no Estado de Santa Catarina.

A empresa continua com os seus objetivos de melhoria da qualidade do serviço apresentado, tanto ao cliente externo, quanto ao cliente interno. Possui uma área de desenvolvimento tecnológico, situada no Parque Celta, onde uma equipe de técnicos, engenheiros e programadores trabalham em conjunto, em prol do desenvolvimento dos equipamentos e sistemas da empresa.

A Telesc possui um sistema de serviços de Telecomunicações que atende todo o Estado de Santa Catarina, e que caracteriza-se pelo suporte de três elementos constituintes: os equipamentos de transmissão, os equipamentos de comutação e de acesso ao assinante, e por fim, as instalações físicas de infra-estrutura. A multiplicidade de equipamentos que são necessários para a otimização do desempenho dessa Empresa, e a exigência por qualidade nos serviços prestados pelas concessionárias de telecomunicações, impulsionaram a realização de integração da rede e dos serviços de telecomunicações.

As concessionárias de serviços de Telecomunicações têm como um dos objetivos, a manutenção dos serviços dos seus usuários e de seus equipamentos, visando os princípios de qualidade de funcionamento e satisfação do cliente.

As linhas de assinantes constituem os pontos de acesso aos serviços de Telecomunicações. Estes pontos são interligados por meio de equipamentos de comutação, como as centrais telefônicas, e transmissão, e estes devem ser monitorados para detectar falhas eventuais, ou possibilidades de falhas futuras, com o

objetivo de efetuar as correções no tempo mais rápido possível, realizando dessa forma, uma manutenção corretiva e preventiva nos equipamentos.

Para realizar uma integração da rede e dos serviços de telecomunicações, existe a necessidade de padronizar os equipamentos para que a supervisão dos mesmos não seja limitada a um sistema ou plataforma de gerência específicos.

Além dos equipamentos de comutação e transmissão, as concessionárias de Telecomunicações possuem equipamentos como geradores, refrigeradores de ar, além de outros, que também são gerenciados. Detectando-se as possíveis falhas às quais estes equipamentos estão sujeitos, e corrigindo-as em tempo hábil, não haverá transtorno nos serviços prestados, nem um comprometimento significativo da receita de receitas e com o desempenho do sistema de Telecomunicações.

Com base nas constatações e com a nova percepção do mercado, percebe-se a crescente necessidade de estruturação, uniformização e automação dos procedimentos de gerência de falhas em sua rede de telecomunicações, a Telesc iniciou, em 1996, o projeto SIORE - Sistema Integrado de Operações Remotas. O objetivo traçado para esse projeto é especificar e desenvolver um sistema de gerenciamento de equipamentos de telecomunicações seguindo o modelo de gerência SNMP (*Simple Network Management Protocol*), isto é, desenvolver um agente SNMP que execute as tarefas relacionadas a cada equipamento e apresente as informações que estes produzem de modo padronizado, não de modo proprietário, garantindo, desta forma, a interoperabilidade entre os equipamentos da rede e a aplicação de gerenciamento.

A aplicação agente deve possuir capacidade de comunicar-se com a aplicação gerente, que monitorará e controlará os equipamentos da rede de Telecomunicações de Santa Catarina - Telesc, além de coletar as informações que são produzidas pelos elementos de rede.

Como existe a necessidade não apenas de manutenção e supervisão do gerenciamento de falhas, mas também a manutenção da segurança nas centrais telefônicas de grande porte e a comunicação visual entre técnicos (à distância), surge a necessidade de ampliação do projeto SIORE, estendendo o sistema para mais duas novas funcionalidades: o controle da segurança das centrais telefônicas de grande porte e a promoção de uma comunicação visual entre técnicos à distância.

Para tal, a ampliação do SIORE se faz pela adição de um novo sistema, o qual este documento trata do seu desenvolvimento. Consiste de um sistema de aquisição de imagens remotas, que servirá como um suporte de comunicação visual do SIORE.

Esse sistema será capaz de monitorar os ambientes das centrais telefônicas, transmitindo as imagens para uma centro específico, que será operado por um técnico responsável, que desta forma, garantirá a segurança dos ambientes vigiados. É importante deixar especificado que o sistema possui a capacidade de gerenciar automaticamente a segurança dos locais, sem a necessidade de um funcionário permanentemente no controle.

Outra utilização para o sistema, será a comunicação visual entre artes operantes, ou seja, entre o técnico responsável pela manutenção direta dessas centrais, e o operador responsável pela gerência da segurança.

1.2. Metodologia

Por este ser um projeto pioneiro da Telesc, na área de aquisição, controle e gerência de imagens, bem como no que diz respeito à segurança, e não haver disponibilidade de uma plataforma comercial para desenvolvimento deste tipo de sistema, foi preciso buscar ferramentas de domínio público para o desenvolvimento do software. Como fonte de trabalho, utilizou-se livros e principalmente a Internet.

1.3. Sobre o Conteúdo

Esta monografia divide-se em 8 capítulos. O primeiro capítulo ilustra uma abordagem inicial sobre este trabalho, apresentando uma introdução, a metodologia e o conteúdo desenvolvido.

O segundo capítulo apresenta uma abordagem inicial da estrutura da Telesc, referenciando-se os conceitos básicos necessários à compreensão do sistema como um todo. Neste capítulo apresenta-se também uma explanação sobre o projeto

SIORE, o qual o desenvolvimento do Projeto de Fim de Curso está diretamente relacionado.

O terceiro capítulo apresenta as ferramentas utilizadas para o desenvolvimento deste trabalho. Serão apresentadas as plataformas operacionais utilizadas no projeto, como o Sistema Operacional utilizado, a linguagem de programação escolhida, a plataforma gráfica, as bibliotecas para a implementação do driver de vídeo e a interface gráfica e os seus conteúdos conceituais que serão relacionados com o projeto no decorrer desta monografia.

O quarto capítulo apresenta o dispositivo óptico para a aquisição de imagem, a câmera digital da Connectix: QuickCam. Apresentar-se-á uma explicação detalhada do dispositivo, elucidando o seu princípio de funcionamento, seus recursos e os protocolos de comunicação necessários para o seu perfeito funcionamento em conjunto com um computador.

O quinto capítulo refere-se ao desenvolvimento do driver de vídeo. Trata-se do desenvolvimento do tratamento e apresentação dos dados de imagem, gerados pelo dispositivo e tratados pelo computador, via software. É necessário que o leitor deste trabalho, se não possui intimidade com os processos envolvidos, leia os capítulos iniciais, principalmente o terceiro capítulo, que detalha todos os conceitos utilizados no desenvolvimento do projeto.

O sexto capítulo apresenta um sistema de aceleração das imagens. Trata-se do desenvolvimento de um segmento de memória compartilhada. Para o leitor entender como funciona esse processo, é preciso estar a par da plataforma gráfica utilizada no processo, apresentada no terceiro capítulo.

O sétimo capítulo, refere-se sobre a interface gráfica desenvolvida para tornar o sistema mais amigável, prático e dinâmico. Neste capítulo, finalmente, será mostrada a interface gráfica desenvolvida no projeto, explicando e demonstrando todas as funções projetadas para o funcionamento do sistema de aquisição, controle e gerência de imagens.

Por último, o capítulo 8, apresenta as considerações finais sobre o projeto, em conjunto com as conclusões obtidas.

CAPÍTULO 2

A Estrutura da Telesc e o Projeto SIORE

2.1. Introdução

O objetivo deste capítulo é realizar uma explanação referente ao projeto SIORE e o seu contexto na Telesc. Para isso, faz-se uma abordagem inicial na estrutura da Telesc, referenciando-se os conceitos básicos necessários à compreensão do sistema como um todo. Serão introduzidos conceitos referentes a redes de telefonia, centrais telefônicas, a Tele-Supervisão e o Projeto SIORE propriamente dito.

2.2. Redes e Centrais Telefônicas

2.2.1. Rede Telefônica

A rede telefônica da TELECC provê a conversação telefônica entre dois usuários desta rede, a qualquer tempo e distância. Essa conexão deve ser rápida e realizada com um número mínimo possível de tentativas.

Esta rede é composta por um determinado conjunto de centrais de comutação ligadas entre si através de grupos de circuitos telefônicos. Existe um conjunto de regras associado a qualquer rede telefônica, que estabelecem a forma pela qual as chamadas telefônicas originadas pelos assinantes devem ser escoadas através da mesma, e que constituem o Plano de Encaminhamento da rede.

O Plano de Encaminhamento de uma rede telefônica determina a forma como devem ser encaminhadas as chamadas telefônicas, permitindo a determinação do volume de tráfego por rota [SIEMENS, 1975]. Desta forma, é possível dimensionar os grupos de circuitos que interligam as centrais de comutação, definindo o que se denomina de estrutura da rede.

2.2.2. Central Telefônica

Em uma rede telefônica não há interligação permanente entre os assinantes e sim entre eles e um centro de comutação ou central telefônica. Segundo Ribeiro (1980), à medida que cada assinante deseja uma conexão, transmite a solicitação à central e esta realiza uma conexão temporária entre os mesmos, que ficará mantida durante todo o tempo necessário para a comunicação desejada.

Entende-se por comutação a operação que permite realizar esta interligação temporária entre os dois usuários da rede.

A estrutura de hardware das centrais telefônicas pode se dividir em:

Módulo de Comutação: É a parte da central que executa a comutação de voz e dados dos terminais da central, e o controle das vias de sinalização entre processadores, bem como a geração e distribuição dos sinais de sincronismo.

Módulo de Operação e Manutenção: É a parte da central responsável pela realização das funções de operação, manutenção e supervisão, relacionando-se com o meio externo através de periféricos de entrada e saída, que conectados ao módulo de operação e manutenção permite a comunicação entre o operador e o sistema. Estas funções de operação e manutenção são realizadas pelo sistema de Tele-Supervisão.

Módulo de Infra-estrutura: São os equipamentos que realizam a distribuição de energia e, caso o fornecimento externo falhe, a geração dela através sistemas de *no-broke* que contém geradores e baterias, dos sistemas de transmissão de dados podendo ser por microondas, rádio ou outras. Além deste equipamentos existem outros sistemas, como sistemas de refrigeração de ar, antenas e cabos, segurança, outros mais.

As centrais telefônicas constroem-se em dois tipos:

Central local: Central de comutação à qual se ligam linhas de assinantes localizados dentro de sua área de ação. Tipicamente, em áreas urbanizadas, uma

central local serve a assinantes dentro da área de um círculo de raio da ordem de 5 a 6 Km a partir da central.

Central trânsito: Em uma rede telefônica, o ideal seria que as centrais locais estivessem todas ligadas entre si, mas devido ao grande número de centrais este esquema é inviável. Este problema é resolvido interligando-se as centrais locais a uma central específica, que provê as interligações desejadas entre as centrais locais. A central que interliga outras centrais é dita central trânsito, porque as chamadas apenas transitam por ela, não sendo terminadas em assinantes.

2.2.2.1 Centrais de Pequeno Porte

Este tipo de central existe em grande escala, mas com apenas poucas centenas de terminais. Sendo muitas vezes comparáveis com grandes PABX, as Centrais de Pequeno Porte são caixas fechadas com pouco material de apoio desenvolvido para elas e poucas vias de acesso.

2.2.2.2 Centrais de Grande Porte

As centrais que possuem mais de 4000 terminais são denominadas de Centrais de Grande Porte. Estas centrais são poucas em relação ao número total de centrais existentes, e recebem um tratamento diferenciado.

Entre duas centrais locais quaisquer, pertencentes a áreas fechadas diferentes, o tráfego geralmente não justifica a abertura de rota direta. Mas, poderá vir a ocorrer que entre duas centrais locais de grande porte se justifique abrir uma rota direta.

As centrais de grande porte possuem diversas portas permitindo a sua comunicação por diversos meios. Possuem diversos sistemas desenvolvidos pelos fabricantes e pelas empresas usuárias para operacionalizá-las e automatizá-las.

O projeto desenvolvido visa estabelecer a segurança para essas centrais telefônicas de grande porte, através de um sistema de aquisição de imagens por uma câmera digital, transmitindo as imagens para uma centro de gerenciamento

específico. Além da segurança, o sistema criado possibilitará a comunicação visual entre os técnicos operadores da central de gerência com os técnicos de manutenção das centrais de grande porte. Este processo torna a comunicação mais interativa, possibilitando um maior entendimento entre as partes operantes, elucidando quaisquer dúvidas de maneira mais ágil, eficiente e eficaz.

2.3. A Tele-Supervisão

A função da Tele-Supervisão é extrair informações dos diversos elementos de uma rede de telecomunicações, por meio de equipamentos de supervisão remota.

Estas informações são utilizadas durante o processo de tomada de decisões sobre que ações devem ser realizadas diante de determinado problema. No caso de detecção de alarmes, o sistema decodifica-o e determina sua prioridade, e dependendo do grau de prioridade, acionando o técnico responsável de plantão para a central que possui o defeito.

2.4. A Organização da Telesc

Os chamados CGIRs (Centros de Gerência Integrada de Rede), que atuam em cada concessionária de serviços de Telecomunicações, vêm sendo desenvolvidos de maneira a se tornarem capazes de controlar a rede de telecomunicações a partir de um único ponto, centralizando os serviços de gerência da rede.

Atualmente a Telesc possui seis centros regionais, denominados superintendências, que situam-se no Estado de Santa Catarina.

A Superintendência Leste encontra-se em Florianópolis, onde o CGIR, localizado no bairro de Itacorubi, é o centro responsável pela gerência da rede de telecomunicações de Santa Catarina.

Existem vários projetos de gerência sendo desenvolvidos atualmente. Entre os projetos, destaca-se o SIORE - Sistema Integrado de Operações Remotas, que é o responsável pelo controle de equipamentos de telecomunicações, e no qual está

envolvido diretamente este projeto, no que diz respeito ao suporte de comunicação visual como base do gerenciamento de segurança de centrais de grande porte e a comunicação entre técnicos.

2.5. Sistemas de Supervisão da Telesc

Devido ao aumento na estrutura de serviço da TELESC, de 1985 até os dias atuais, visando estabelecer um melhoramento do padrão dos seus serviços prestados ao povo catarinense, desenvolveu-se uma diferenciação dos equipamentos em que se encontram em funcionamento, devido ao crescimento durante o longo período, e a aquisição de equipamentos de diferentes gerações tecnológicas, com diferentes padrões, devido à sua aquisição ser realizada com diferentes fornecedores.

A supervisão ou gerenciamento de falhas é vista na sua maioria isoladamente, dificultando uma visão global do que acontece na rede. Surge, então, a necessidade de um sistema para integrar todos os eventos que ocorrem na rede de telecomunicações, e que são fornecidos pelos diversos sistemas de operação e supervisão.

Estes sistemas de supervisão, que encontram-se em funcionamento na TELESC, estão sendo substituídos, gradativamente, por sistemas padronizados de gerência de redes.

2.6. O SIORE

Denomina-se SIORE, o Sistema Integrado de Operações Remotas, que é uma interface que vai possibilitar aos Sistemas de Operações interagir com os equipamentos de Telecomunicações através de chamadas padrão.

Esta interface provê a coleta de informações e execução de ações sobre os equipamentos através de solicitações remotas ou não.

Este projeto está baseado no desenvolvimento de um agente que situa-se entre os Sistemas de Operações e os equipamentos de telecomunicações.

Entre as cinco funções de gerenciamento (falhas, desempenho, configuração, segurança e contabilização), são identificadas diretamente neste projeto três funções essenciais que estão em vias de implementação: gerência de falhas, configuração e segurança.

A primeira etapa foi alcançada e se destinou à gerência de falhas. A próxima etapa na continuação do desenvolvimento do projeto SIORE, que não foi atingida por completo, deve atentar para a gerência de configuração, e, por último, gerência de segurança.

A principal função de gerenciamento de falhas é a supervisão de alarmes, pois está diretamente relacionada ao gerenciamento da informação sobre as condições de operação que afetam os serviços prestados por uma concessionária de Telecomunicações.

A supervisão de alarmes e telemedidas é uma função de monitoração de equipamentos da rede, enquanto a emissão de telecomandos é uma função de monitoração e controle desses equipamentos.

Para ampliação do SIORE, no que tange a gerência da segurança nas centrais telefônicas de grande porte, bem como o estabelecimento de uma comunicação visual entre as partes operantes, desenvolveu-se este trabalho.

Este projeto visa estabelecer um suporte de comunicação visual entre as centrais de grande porte, monitoradas pelo SIORE, e o Centro de Gerência Integrado de Rede.

Através da aquisição das imagens dessas centrais de grande porte, o CGIR é capaz de se comunicar visualmente com os técnicos responsáveis pela manutenção dessas centrais, esclarecendo dúvidas e tornando a comunicação mais fácil e interativa.

Outra função do projeto será o apoio que este dará ao SIORE com relação à segurança das instalações. Dentro deste projeto foram desenvolvidas algumas funções que possibilitam o sistema atuar como um vigilante, garantindo a segurança das centrais de grande porte.

2.7. Conclusões

Este trabalho consiste de uma nova aplicação, inovadora e sem base específica para desenvolvimento, e que tornará todo o sistema mais interativo, oferecendo uma comunicação mais efetiva entre as partes, e um sistema de segurança apto às necessidades da Telesc.

Os próximos capítulos deste trabalho convergem para o desenvolvimento de um sistema para este fim, relacionando os equipamentos, plataformas e técnicas desenvolvidas, que se tornarão parte do SIORE.



CAPÍTULO 3

Plataformas de Desenvolvimento do Projeto

3.1. Introdução

Neste capítulo, apresentar-se-ão todas as plataformas de desenvolvimento do projeto, incluindo o sistema operacional, a linguagem de programação, os recursos gráficos utilizados, incluindo as bibliotecas para o posterior tratamento de imagem (driver de vídeo). Este capítulo é de extrema importância para o leitor que não possui conhecimentos sobre o sistema operacional e o ambiente gráfico sob o qual o programa foi desenvolvido, pois serão apresentados todos os recursos utilizados no software, nomenclaturas e estruturas de programação.

3.2. Sistema Operacional : Linux

O sistema operacional utilizado pelo SIORE, e que também é utilizado pelo sistema de aquisição de imagens, é o Sistema Operacional Linux.

O LINUX, uma versão do sistema operacional UNIX, alcançou notoriedade ao ser identificado como “o UNIX para PC” [WELSH,1993]. Foi anunciado pela primeira vez em novembro de 1991, desenvolvido por Linus Torvald, estudante de Ciência da Computação da Universidade de Helsinki, Finlândia. Torvald preparou o sistema para rodar em computadores com processador Intel 386 em diante.

O sistema inclui proteção entre processos (crash protection), carregamento por demanda, redes TCP/IP, além de nomes de arquivos com até 255 caracteres, multi-tarefa real, suporte a UNICODE, bibliotecas compartilhadas, memória virtual, etc.

O Kernel é o núcleo do sistema operacional, e está sob os termos do GNU (General Public License).

Linus Torvalds iniciou cortando (hacking) o kernel como um projeto particular, inspirado em seu interesse no Minix, um pequeno sistema UNIX desenvolvido por

Andy Tannenbaum. Ele se limitou a criar, em suas próprias palavras, "um Minix melhor que o Minix".

O LINUX possui diversas distribuições, desenvolvidas por grupos de pessoas diferentes, onde variam, em geral, a maneira de instalar, o número de *softwares* que acompanham o sistema básico e a facilidade de manutenção do sistema.

Algumas versões do LINUX são comerciais, como o Solaris (Sun) e o AIX (IBM). Outras são de livre distribuição, como o próprio Linux, o FreeBSD e o NetBSD. Na verdade, o LINUX é um *freeware*. Pode ser distribuído obedecendo-se o "GNU Public Licence" (GPL).

Criado no final dos anos 60, o UNIX vem sendo desenvolvido em C desde a década de 70, com o objetivo de se tornar um ambiente portátil. Isso ajuda a entender porque ele existe até hoje, ao contrário dos outros sistemas que morrem quando as máquinas ficam obsoletas.

O UNIX é uma marca registrada do Unix Lab. Todos os sistemas baseados naqueles códigos são chamados de uma forma geral de UNIX.

O Linux foi escrito desde o início pelo Linus Torvalds e não contém nenhuma linha de código do UNIX. Mas o Linux foi escrito para ser conforme o padrão POSIX, que deve ser o padrão da API (Application Programming Interface).

Por causa da API POSIX, do conjunto de utilitários do uso do X-Windows (XFree), o Linux é tão parecido com o UNIX que existem empresas que usam o Linux para desenvolver para UNIX que não seja o dela mesma (como exemplo, a IBM e a Microsoft).

Para o desenvolvimento de ferramentas e utilização do sistema, foi desenvolvido para o LINUX um sistema de janelas gráficas num ambiente semelhante ao Windows95: é o X Windows. Este ambiente facilita o acesso ao seu sistema e permite maior acessibilidade. O sistema X Windows será explicado posteriormente.

3.3. Linguagem de Programação: C

C é uma linguagem muito portátil, isto é, programas escritos em C podem executar em várias máquinas sem alterações significativas. A portabilidade é

esperada devido à linguagem ser concisa e ausente de variações significativas nos compiladores. A portabilidade de C e sua performance, que se aproxima da linguagem Assembly, a tornam uma opção apropriada para muitas aplicações. Embora originalmente desenhada para o sistema operacional UNIX, C tem-se expandido para muitos sistemas de microcomputadores.

A arquitetura de C permite a construção de várias sentenças que são muito estranhas para programadores em outras linguagens [SCHILDT, 1989].

O uso da linguagem C para implementação do programa justifica-se no fato das características desta linguagem estarem diretamente ligadas à portabilidade, eficiência e conveniência. Como o sistema operacional que estará nas estações agentes de gerenciamento é o LINUX, é natural pensar que a aplicação de gerenciamento também pode ser melhor configurada e desenvolvida com esta linguagem, permitindo assim utilizar os recursos deste sistema operacional.

3.4. O Sistema X Window

X Window é um sistema baseado em sistemas de janelas com base em rede, onde aplicações podem ser executadas em redes de sistemas de diferentes tipos. Programas podem ser executados em um computador remoto, e os resultados apresentados em estações de trabalho (*workstations*) locais [ADRIAN, 1992].

Um servidor X controla uma tela mapeada em bits (*bitmapped*). De maneira para facilitar a visualização e controle de diversas e diferentes tarefas, ao mesmo tempo, essa tela pode ser dividida em áreas menores denominadas janelas. As janelas, na tela, podem ser manipuladas de tal maneira em que todas podem ser visualizadas ou encobertas umas pelas outras completamente, ou parcialmente. Cada janela (em uma tela rodando o X) pode ser envolvida em uma diferente atividade, e a janela em uso atual é colocada de maneira total, encobrindo parcialmente ou totalmente as demais.

3.4.1. O Modelo Cliente-Servidor

O sistema X Window é um sistema de janelas orientadas à rede. Uma aplicação não precisa estar sendo executada no mesmo sistema que suporta o display. Enquanto muitas aplicações podem executar localmente em estações de trabalho (*WorkStations*), outras aplicações podem executar em outras máquinas, mandando requisições através da rede para um display particular e recebendo eventos de teclado e ponteiros do sistema controlador do display.

O programa que controla cada display é conhecido como servidor. O servidor atua como um intermediário entre programas de usuários, denominados clientes ou aplicações, executando no sistema local ou remoto e recursos do sistema local. O servidor desempenha as seguintes tarefas:

- Permite acesso ao display por múltiplos clientes;
- Interpreta mensagens de rede dos clientes;
- Transfere entradas do usuário para os clientes pelo envio de mensagens de rede;
- Realiza desenhos bidimensionais – gráficos são realizados de preferência pelo display servidor do que pelo cliente;
- Mantém estruturas de dados complexas, incluindo janelas, cursores, fontes e contextos gráficos, como recursos que podem ser compartilhados entre clientes e referenciados simplesmente pelo recurso de Identificação (*ID*). Recursos de manutenção do servidor reduzem o acúmulo de dados que necessitam ser mantidos por cada cliente e o acúmulo de dados que tem que ser transferidos pela rede.

3.4.2. Gerenciador de Janelas

O gerenciador de janelas é outro programa escrito com Xlib, exceto que é dado especial autoridade para o controle de janelas na tela. O gerenciador de janelas permite o usuário mover ou redimensionar janelas, iniciar novas aplicações e controlar o empilhamento de janelas na tela, mas somente de acordo ao policiamento do lay

out do gerenciador. O policiamento do layout de uma janela é um conjunto de regras que especificam dimensões permitidas e posições de janelas e ícones.

3.4.3. Eventos

Como em qualquer sistema de janelas, um cliente X deve estar preparado para responder a qualquer dos diferentes eventos. Esses eventos incluem entrada de dados (*inputs*) do usuário (teclado, mouse), como também interações entre programas. Eventos de diferentes tipos podem ocorrer a qualquer momer. e em qualquer ordem. São colocados em uma fila na ordem de ocorrência, e usualmente são processados por clientes nesta ordem. A programação de controle de eventos permite o usuário dizer ao programa o que fazer, e não o contrário.

A necessidade de manipulação de eventos é a maior diferença entre a programação sob o sistema de janelas e a tradicional programação PC ou Unix. Programas X não utiliza as funções C padrões para capturar caracteres, e não faz votação de prioridade para dados de entrada. Ao invés, existem funções que recebem eventos, e então, o programa deve classificar de acordo com o tipo do evento e desempenhar a resposta apropriada. Mas, ao contrário de programas tradicionais, um programa em X deve estar preparado para certos tipos de entradas de dados a certos períodos de tempo. Em programas X, o usuário está no controle a maior parte do tempo.

3.4.4. Arquitetura do Software do Sistema X Window

Um display servidor é um programa que executa em um sistema que suporta um display gráfico, teclado ou mouse.

Aplicações comunicam-se com o servidor por meio de chamadas para um biblioteca de baixo nível, escrita em rotinas de linguagem C, denominada Xlib. Essa biblioteca fornece funções para conexão de um particular display servidor, cria janelas, desenha gráficos, e outros mais. As chamadas de Xlib são traduzidas em

requisições de protocolos enviados via tcp/ip para o servidor local ou para outro servidor através da rede.

Aplicativos e gerenciadores de janelas podem ser escritos somente com Xlib ou com a configuração de alto nível de bibliotecas de subrotinas conhecidas como kit de ferramentas (*toolkits*). Esse kit implementa a configuração das características da interface do usuário, como menus e botões de comando e permite que aplicações manipulem essas configurações, utilizando técnicas de programação orientada a objetos. A figura 3.1 apresenta a arquitetura do software do Sistema X Window, com todos os seus componentes.

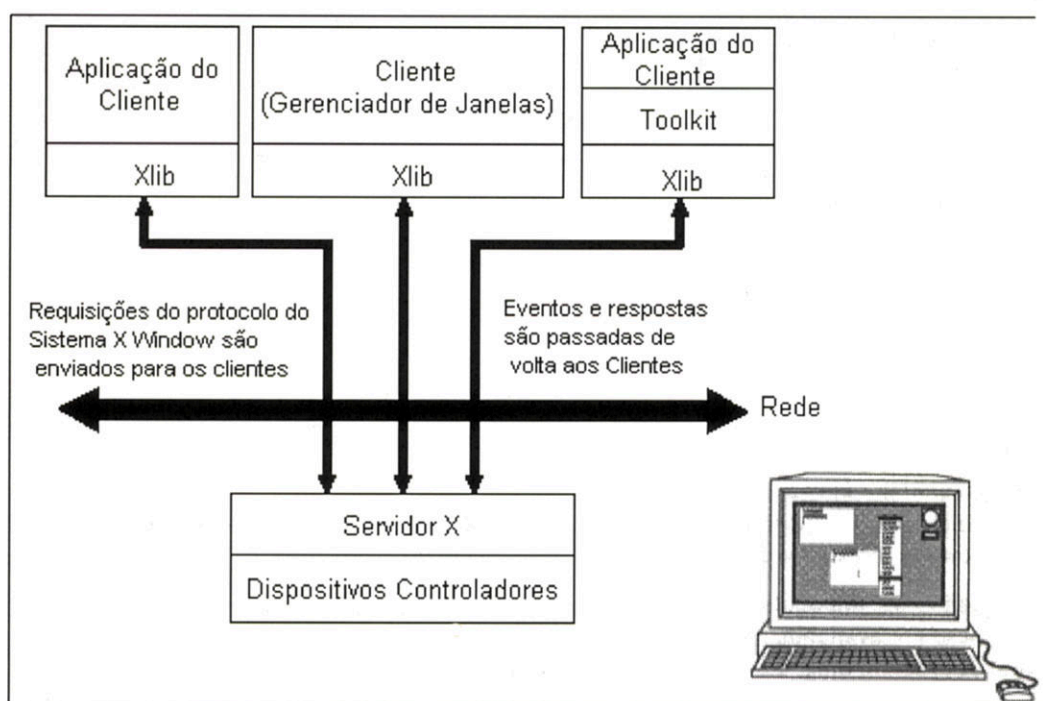


Figura 3.1 – Arquitetura do Software do Sistema X Window

3.5. Xlib

Xlib é uma programação de interface do Sistema X Window, o qual faz parte do Sistema Operacional Linux. A biblioteca X, conhecida como Xlib, é o mais baixo nível de programação de interface para o X Window [ADRIAN, 1992]. Essa biblioteca possibilita o programador escrever aplicações com uma avançada interface baseada em janelas na tela do computador.

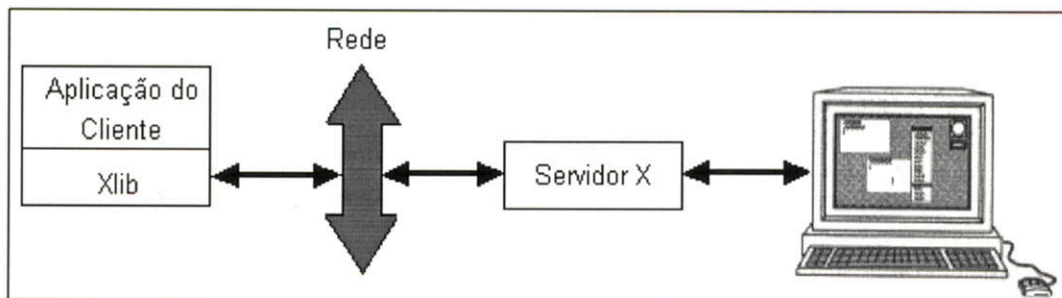


Figura 3.2 – Clientes e Servidores

A figura 3.2, apresenta a estrutura de funcionamento do X Window com o Xlib, em uma aplicação cliente – servidor.

A partir de agora, veremos alguns conceitos necessários de Xlib para que possamos seguir em frente com o desenvolvimento do projeto.

3.6. Introdução ao X Graphics

Para melhor entender o procedimento de tratamento da imagem, existe a necessidade de um entendimento de todas as funções envolvidas e denominações utilizadas, de forma que o leitor possa compreender o processo em si. Esta seção provê uma introdução aos termos e conceitos utilizados em gráficos no Sistema X Window.

3.6.1. Pixels e Cores

O sistema X Window é desenhado para controlar displays gráficos em bitmaps. Em um simples display preto e branco, existe um bit por pixel: o estado desse bit determina onde o pixel será preto ou branco. Em sistemas coloridos ou em sistemas monocromáticos de displays de escalas de cinza, existem múltiplos bits por pixel.

O estado desses múltiplos bits, designados para cada pixel, não controlam diretamente a cor ou a intensidade de cinza da cada pixel. Ao invés disso, eles são utilizados como indexadores para localização, em uma tabela, denominada Mapa de cores (*colormap*).

Em um display colorido, um pixel consiste de três colorações diferentes: vermelho, verde e azul. A intensidade relativa dessas três colorações, apresentam-se para a vista humana, como uma cor única. Dessa maneira, o mapa de cores (*colormap*) é constituído de um array triplo, para as três cores (RGB – *Red, Green, Blue*). Em outras palavras, se o valor dos bits para um dado pixel é 14, os valores de RGB para o décimo quarto membro do mapa de cores serão mostrados na localização da tela. A figura 3.3 representa a transformação dos valores de um byte em uma cor correspondente, formada pelas três cores básicas: vermelho, verde e azul.

Cada membro do mapa de cores é denominado célula de cor (*colorcell*) cada qual traduz um valor de pixel em uma setagem específica de vermelho, verde e azul. Todos os displays mapeados em bits (*bitmapped*) têm pelo menos um mapa de cores básico, mesmo no caso de uma tela monocromática de plano simples, onde o mapa de cores consiste em apenas duas células de cores. Na maioria dos casos, todos os clientes compartilham de um único mapa de cores, alocando somente o número de células de cores que necessitam.

Quando clientes possuem requerimentos especiais, entretanto, X permite que eles possuam células de cores particulares, ou permite a criação de mapas de cores virtuais que são trocados no mapa de cores real, quando necessário.

3.6.2. Pixels e Planos

O número de bits por pixel é também referenciado como o número de planos no display gráfico. Sistemas em preto e branco possuem um plano único, displays coloridos possuem de 4 a 28 planos, e displays de escalas de cinza possuem de 2 a 4 planos, usualmente. X11 suporta até 32 planos.

Como pode ser concluído pela explicação anterior de bits por pixel como indexadores de mapas de cores, o número possível de cores ou tons de cinza que podem ser simultaneamente mostrados na tela é 2^n , onde n é o número de planos no display.

Todos os cálculos gráficos são provenientes do valor do pixel antes que sejam traduzidos em RGB.

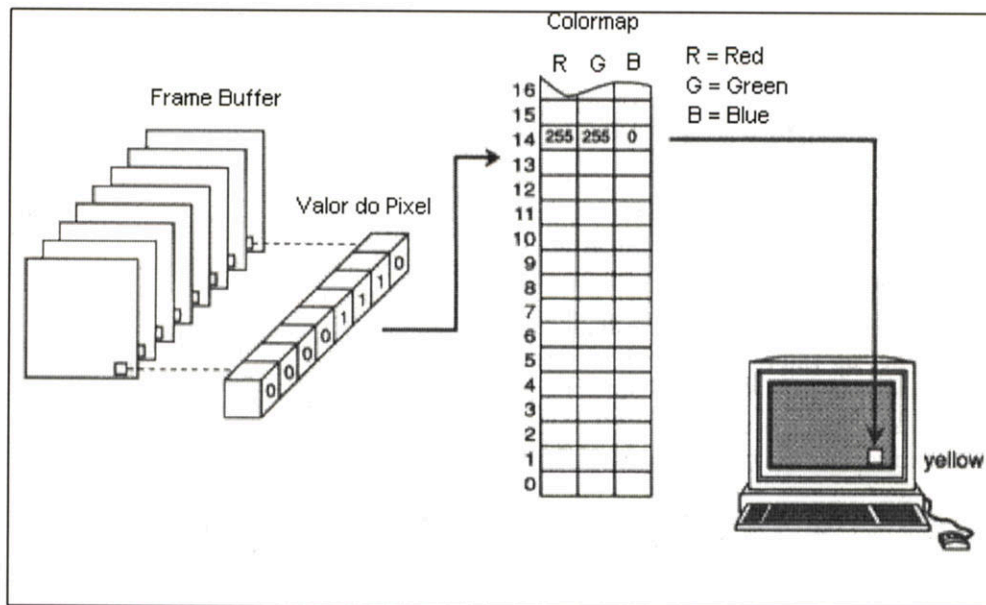


Figura 3.3 – Mapeamento do Pixel em um Mapa de Cores

3.6.3. Desenhos e Contexto Gráfico

Como em qualquer linguagem gráfica, X fornece rotinas de desenho de pontos, linhas, retângulos, polígonos, arcos, textos, etc. Rotinas que desenharam gráficos são genericamente chamados de primitivas gráficas. Mas em X, uma dada primitiva gráfica não contém todas as informações necessárias para desenhar um gráfico em particular. Um recurso do servidor chamado de Contexto Gráfico (GC – *Graphic Context*) especifica as variáveis restantes, como largura de linhas, cores, etc. A identidade (*ID*) de um GC é especificada como um argumento para a rotina de desenho e modifica a aparência de tudo que é desenhado na área de desenho.

O GC deve ser criado pelo cliente antes que o desenho seja feito. O GC criado é armazenado no servidor, dessa maneira a informação que ele contém não precisa ser enviada com todas as primitivas gráficas; somente o ID é passado. Isso aumenta a performance do desenho significativamente, pois reduz o tráfego sobre as conexões entre o Xlib e o servidor. Todas as setagens do GC são aplicáveis a todos os desenhos gráficos que utilizam o GC.

Mais de um GC pode ser criado, e cada um pode ser setado com diferentes valores. Isso permite que um programa troque os GCs e obtenha diferentes efeitos com as mesmas primitivas gráficas.

3.7. Características das Janelas

Nesta seção serão apresentados alguns conceitos sobre as características das janelas utilizadas no desenvolvimento deste projeto.

3.7.1. Intensidade, Aparência e Identificação

A intensidade (*Depth*) e a aparência (*Visual*) de uma janela são designados na criação desta, e não podem ser alterados. A intensidade é o número de planos que são usados para representar escalas em cinza ou cores, em uma janela. A intensidade também é o número de bits por pixel. O máximo de intensidade conseguida para uma janela, é o número de planos suportados pela tela com a qual está associada. Se uma tela tem doze planos, uma janela pode ter até 12 bits por pixel, sendo assim, haverá no máximo 2^{12} possíveis níveis de cinza ou cor.

A aparência se deve às diferenças entre vários tipos de display para a determinação da maneira com que os valores dos pixels são traduzidos em cores visíveis em uma janela particular. Uma tela pode suportar apenas uma aparência, de diversos tipos de aparência.

Para cada janela gerada, é fornecido um número identificador particular. Todas as rotinas que afetam uma janela, em particular, utilizam o seu identificador (*Window ID*) como um argumento, e atua no ambiente desta janela. Dessa maneira, posições na janela são especificadas em relação ao seu canto superior esquerdo. Não é necessário conhecer a posição de uma janela para localizar corretamente subjanelas ou desenhar gráficos dentro da mesma.

3.8. Estruturas de Imagem: XImage

Xlib provê uma estrutura de imagem que é capaz de armazenar toda a informação correspondente para uma área da tela. Xlib fornece as rotinas *XGetImage* e *XPutImage* que utilizam o protocolo para transferir o conteúdo de uma janela ou *pixmap* para uma estrutura de imagem e escrever o conteúdo de uma estrutura de imagem de volta para a janela.

Xlib fornece um mínimo de rotinas para manipulação de estruturas de imagens, incluindo rotinas de criação e inicialização de estruturas de imagens vazias, destruição de estruturas de imagens, captação de pixels, setagem de pixels, extração de uma subimagem de uma imagem, e adicionar um valor constante para todos os pixels de uma imagem. Estas rotinas podem ser relativamente lentas, porque elas trocam o byte e a ordem de bits da imagem, antes de desempenhar a operação, e trocam novamente antes de colocar de volta na imagem.

As rotinas de processamento de imagens fornecidas pelo Xlib são mínimas, elas não fornecem um pacote completo de manipulação de imagem. Entretanto, a estrutura da imagem contém todas as informações necessárias para implementar um pacote completo. Uma aplicação pode implementar suas próprias rotinas para manipular os dados da imagem, diretamente. Contudo, esse código é difícil de escrever de uma maneira portátil e eficiente, por causa do alto número de formatos dos dados de imagem que são possíveis.

3.8.1. Manipulação de Imagens

Para a operação com imagens no X Windows, existem algumas estruturas de comandos utilizadas para tal finalidade, dentre elas, veremos as seguintes estruturas:

XCreateImage(): Seleciona memória para uma estrutura de imagem X, para um Display e Aparência (*Visual*) especificado. Esta função não seleciona espaço para a imagem propriamente dita. Ela inicia a estrutura com disposição de bytes, disposição de bits, valores de unidades de bitmap, e retorna um ponteiro para a estrutura de *Ximage*.

XPutImage(): desenha a imagem na área de desenho especificada.

XDestroyImage(): libera o campo de dados de uma estrutura de imagem se esta foi alocada em uma aplicação. Se a imagem foi criada através do comando *XCreateImage()*, ela libera os dados e a estrutura de imagem. Observa-se que se os dados da imagem são armazenados em uma memória estática no aplicativo, esta não poderá ser liberada. Para liberar uma imagem nesta condição, deve-se setar *NULL* no campo de dados, antes de chamar a função *XDestroyImage()*.

3.8.1.1. Contexto Gráfico

As rotinas de Xlib que desenhavam gráficos são chamadas de primitivas gráficas. Elas desenhavam pontos, linhas, textos e imagens. Mas uma dada primitiva gráfica não contém todas as informações necessárias para desenhar um gráfico em particular. O recurso do servidor chamado Contexto Gráfico (GC) contém valores para variáveis que se aplicam para cada primitiva gráfica. A aparência de tudo que é desenhado por um programa é controlado pelo GC que é especificado com cada primitiva gráfica. Uma exceção dessa regra é a borda e cor de fundo de uma janela. Estas não são afetadas ou controladas pelo GC – são controladas pelas atribuições da janela, e desenhadas pelo servidor.

Existem duas razões para as quais o Xlib foi designado para utilizar o GC. Primeiro, se reduz o tráfego entre o Xlib e o servidor, porque a informação do GC é mantida no servidor e necessita ser enviada apenas uma vez antes da primeira requisição de gráfico. Cada primitiva subsequente que especifica o mesmo GC utilizará os mesmos valores. Quando pequenas aplicações do GC necessitam ser alteradas, apenas a seleção para mudança é enviada, e não todo o GC. Em segundo, é possível criar vários GCs e então simplesmente especifica-se qual GC o usuário quer aplicar para cada requisição gráfica. Isto tem importantes benefícios de desempenho nos servidores que são capazes de capturar múltiplos GCs no display hardware.

O GC também permite uma maior conveniência de programação. Para prover a mesma flexibilidade sem o GC, deve-se especificar um número absurdo de argumentos cada vez que uma primitiva gráfica for requisitada.

Uma explicação mais detalhada necessita ser realizada para esclarecer o papel desempenhado pelas primitivas gráficas e o GC. Deve-se imaginar as primitivas gráficas como sendo uma especificação da forma geral a ser desenhada, enquanto o GC especifica a maneira de desenhá-las. Por exemplo, uma primitiva que desenha um retângulo preenchido especifica o canto superior esquerdo do retângulo na área de desenho e suas dimensões, enquanto o GC especifica a sua cor ou o padrão aplicado (além de outros aspectos). Observa-se que ambos, primitiva gráfica e GC, representam um papel na seleção exata dos pixels a serem desenhados. Neste mesmo exemplo, as primitivas gráficas especificam o início e o fim das linhas, enquanto o GC especifica a largura da linha, o formato das junções e o fim das linhas.

3.9. XForms

XForms é um pacote desenvolvido para auxiliar na criação de interfaces gráficas e programas. É uma biblioteca de rotinas em linguagem C, que permite o usuário construir uma interação de formas com botões, deslizadores (*sliders*), campos de entrada de dados, sintonizadores, etc [ZHAO & OVERMARS, 1995].

A biblioteca Forms utiliza apenas os serviços providos pela biblioteca Xlib, e pode ser compilado em todas as máquinas que possuam o X instalado e possuam compilador ANSI compatível. Sendo baseado na biblioteca Xlib, a biblioteca XForms é pequena e eficiente. Pode ser usada em linguagem de programação C ou C++.

A noção principal da biblioteca XForms, é a criação de formas. Uma forma é uma janela onde diferentes componentes (*widgets*), como botões, janelas de mensagens, menus, etc podem ser colocados. A forma pode ser mostrada e o usuário pode interagir com os diferentes componentes que são colocados nesta. A biblioteca Forms pode ser usada para criar eventos controlados pela conexão de rotinas de chamada (*call backs*) em objetos. Essas rotinas de chamadas são chamadas quando existe uma interação com o objeto (ativação de um botão, por exemplo). Dentro da

rotina de chamada, é determinado o tipo de interação que é realizada e qual a reação que o programa tomará.

XForms constitui-se de dois componentes. O primeiro é o denominado “*fdesign*”, um programa que permite a criação e modificação de janelas que consistem de elementos de interface gráfica, como botões, menus, sliders, etc. O *fdesign* gera um esqueleto de código C que permite, onde o usuário o complementar de acordo com as especificações do próprio programa.

O outro componente é a Biblioteca XForms, que é interligada (*linked*) com o programa o qual o usuário está desenvolvendo.

Através da construção da interface gráfica, pelo *fdesign*, ocorre a geração de dois arquivos que devem ser compilados e interligados com o programa principal, gerando um programa executável que tornará o sistema operacional.

Os arquivos gerados constituem-se por um arquivo de extensão *.c* e um arquivo de extensão *.h*.

Para interligar os programas desenvolvidos no *fdesign* e o programa principal, é necessário da realização da seguinte estrutura de controle de eventos:

Inclusão da biblioteca XForms:

```
#include <forms.h>
```

Inclusão do header desenvolvido para a interface gráfica:

```
#include "interface_grafica.h"
```

Desenvolvimento das Rotinas de Chamada:

```
Call back 1
```

```
...
```

```
Call back n
```

Criação das Formas:

```
Create the forms {
```

```
Create form1()
```

```
...
```

```
Create formn() }
```

```
Create form1 {
```

```
Add widget()
```

```
Add Callback() }
```

```
Create formn {...}
```

```
Programa Principal{
```

```
...
```

```
Comando de reconhecimento e Inicialização da interface gráfica:
```

```
fl_initialize(&argc,argv,0,0,0);
```

```
Create the forms()
```

```
fl_do_forms() }
```

Através de vários comandos, providos pela biblioteca XForms, junto com a biblioteca configura-se as funções da interface, como os botões, reguladores, textos de aviso, etc.

Para interligar os programas para criação de um programa executável, é necessário a utilização de um dispositivo para este fim, que será visto na seção 3.10.

3.9.1. Rotinas de Chamada dos Comandos da Interface

Para conseguir uma interação entre o programa principal e a interface gráfica, é necessário realizar uma rotina de chamadas dos objetos da interface criada (*callbacks*). Essas rotinas estarão inseridas no programa principal, e quando alguma função da interface gráfica for solicitada, essas rotinas serão ativadas no próprio programa principal, respondendo aos comandos solicitados.

Para exemplificar, estabelece-se a rotina de chamada para a alteração dos valores das configurações da imagem, ou seja, as mudanças no brilho, contraste e balanço. Para tal, a seguinte função é implementada:

```
void Slider (FL_OBJECT *obj, long val)
{
    switch(val)
    {
        case 0:
            exposure = fl_get_slider_value(obj);
```



```

        SetExposicao();
        break;
    case 1:
        contraste = fl_get_slider_value(obj);
        SetContraste();
        break;
    case 2:
        offset = fl_get_slider_value(obj);
        SetOffset();
        break;
    }
}

```

Slider é o nome da rotina de chamada das funções de configuração da imagem. Essa rotina provê a alteração dos valores de brilho, contraste e balanço. Caso o botão deslizante do brilho for alterado, a função *Slider* é chamada e a variável *exposure*, que indica o valor do brilho, será alterada, capturando o valor do próprio botão deslizante da interface gráfica. Após a alteração do valor, é enviado o comando de mudança do brilho da imagem, referenciado por *SetExposure()*. Esses comandos e funções serão detalhadamente explicadas no próximo capítulo.

3.10. Makefile

O comando **make** permite o gerenciamento de programas extensos ou grupos de programas. Quando o código do programa se torna extenso, nota-se que a recompilação requer um grande período de tempo, em relação à compilação de pequenos programas. Além do mais, apenas as partes do programa que foram alteradas necessitam ser recompiladas, o resto do programa, que permanece inalterado desde a última compilação, não apresenta essa necessidade [STUART, 79].

O programa **make** auxilia no desenvolvimento dos extensos programas pela procura automática das partes do programa que foram alteradas, compilando apenas essas partes do programa que foram alteradas desde a última compilação.

3.10.1. Compilação com Vários Arquivos

Quando o programa torna-se muito extenso, faz sentido dividir o código em partes separadas, em arquivos de extensão `.c`, mais facilmente gerenciados. A figura 3.4 demonstra a compilação de um programa feito por dois arquivos de extensão `.c` e um arquivo de extensão `.h`.

O comando para a compilação é `cc green.c blue.c`. Onde os dois arquivos de extensão `c` são entregues ao compilador. Após a compilação, os dois arquivos de extensão `.o` são linkados juntos no estágio Linker, para criar um programa executável: `a.out`.

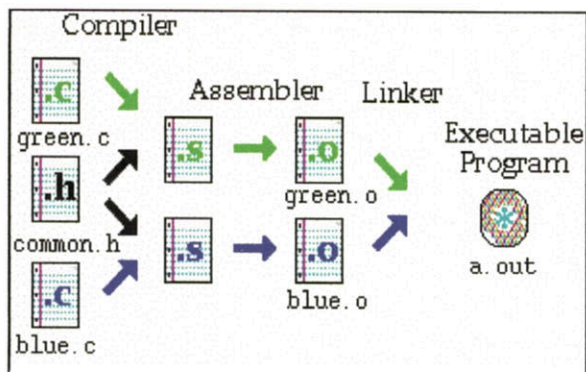


Figura 3.4 – Compilação com Vários Arquivos

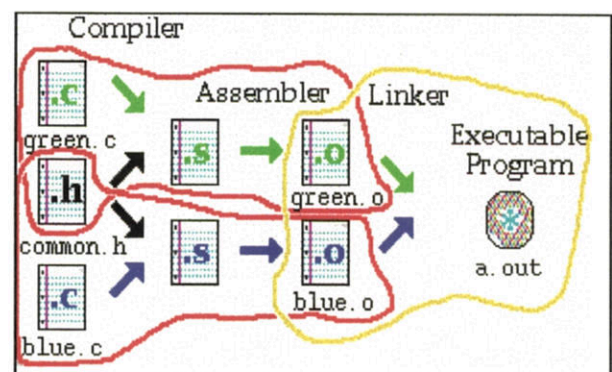


Figura 3.5 – A Compilação Separada

3.10.2. A Compilação Separada

Os passos para a criação de um executável podem ser divididos em dois passos de compilação/reunião (*compiler/assembler*), circutados em vermelho, na figura 3.5, e um passo de interligação (*linker*) final, circulado em amarelo. Os dois arquivos de extensão `.o` podem ser criados separadamente, mas ambos são requeridos no passo final para criar o executável.

É importante salientar que para se criar o arquivo objeto `green.o`, os dois arquivos, `green.c` e o header `common.h` são requisitados. Similarmente, de maneira a criar o programa executável, `a.out`, os arquivos objetos `green.o` e `blue.o` são necessitados.

3.10.3. Dependências

O programa **make** cria programas de acordo com as dependências do arquivo. Por exemplo, para criar um arquivo objeto *program.o*, necessita-se do arquivo *program.c*. Esta seção envolve a denominada “dependência gráfica”, que pode ser vista a seguir.

A figura 3.6 consiste de uma dependência gráfica. Representa um programa feito por cinco arquivos fontes, denominados *data.c*, *data.h*, *io.c*, *io.h* e *main.c*. O topo é o resultado final, o programa chamado *project1*. As linhas que saem dos arquivos em direção aos arquivos inferiores, indicam que estes são as dependências do arquivo de onde a linha se originou.

A figura 3.7 mostra como a dependência funciona no processo de compilação. Suponha que o programa *io.c* tenha sido alterado por alguma razão (correção de um bug, melhoria do código, etc). Ao realizar o procedimento de compilação **make**, o mesmo analisa quais os programas que foram alterados e analisa as suas possíveis dependências. No caso, os programas que serão recompilados serão apenas o *io.c*, *io.o* e o *project1*. Os outros, como seus códigos não foram alterados, suas dependências continuam as mesmas, portanto, não necessitam ser recompilados.

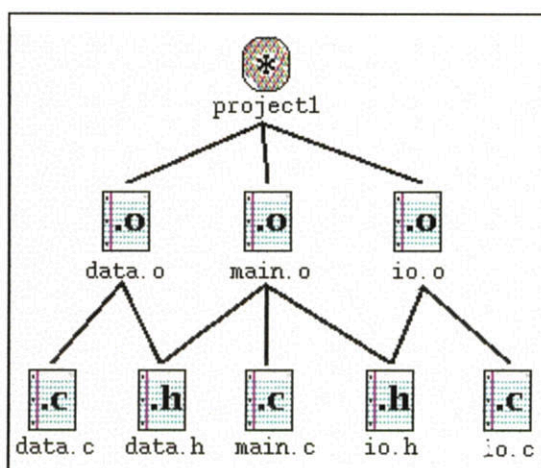


Figura 3.6 – Dependência Gráfica

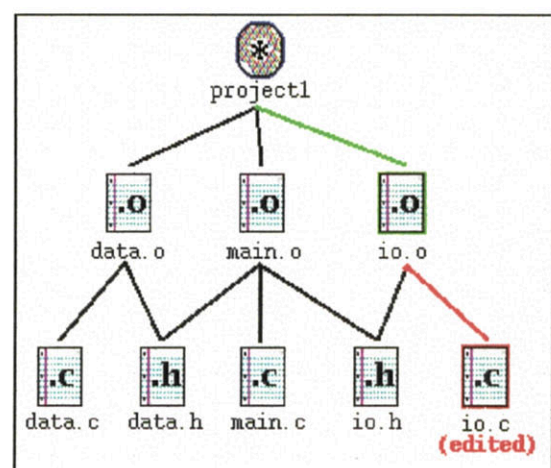


Figura 3.7 – Compilação e Dependência Gráfica

3.10.4. Como Funciona

O programa **make** adquire as dependências gráficas de um texto denominado *makefile* ou *Makefile*, onde é residente no mesmo diretório dos códigos fontes. **Make** checa os tempos de modificações dos arquivos, e quando um arquivo possui data mais atualizada, ou seja, quando este é modificado, executa-se o compilador.

3.10.5. O Makefile

Esta seção descreve o programa **make** mais detalhadamente, pela descrição do arquivo que ele usa, denominado *makefile* ou *Makefile*. Este arquivo determina a relação entre os códigos, objetos e arquivos executáveis.

A tradução da dependência gráfica ocorre de acordo com a figura 3.8. Cada dependência é mostrada no gráfico está circulado por uma cor correspondente no *Makefile*, e que usa o seguinte formato:

Target: arquivos fontes

Command

Um dado alvo no *Makefile* é um arquivo que poderá ser criado ou atualizado quando qualquer dos arquivos fontes forem alterados. Os comandos dados nas linhas subsequentes são executados de maneira a criar o arquivo alvo.

Na tabela da figura 3.8 (*sample makefile*), os arquivos de extensão *.h* são listados, mas não existem referências nos seus comandos correspondentes. Isto ocorre porque os arquivos *.h* são referenciados com o correspondente arquivo *.c* pelo comando, no arquivo *.c*, `#include "file.h"`. Se não for explicitamente incluso no *Makefile*, o programa não será atualizado se for feita alguma alteração no arquivo header (*.h*).

Depois do desenvolvimento do *Makefile*, para executá-lo, basta digitar o comando **make**, e o sistema iniciará o processo de compilação e criação do executável.

A necessidade da utilização desta ferramenta é providencial para a compilação do software de aquisição de imagens, pois o mesmo constitui-se de um programa principal e dois outros programas, gerados na criação da interface gráfica de controle do sistema.

Desta maneira, foi desenvolvido um programa *Makefile*, de forma a executar a interligação e compilação dos três arquivos fontes, para um programa alvo, no caso, que será o programa executável do sistema.

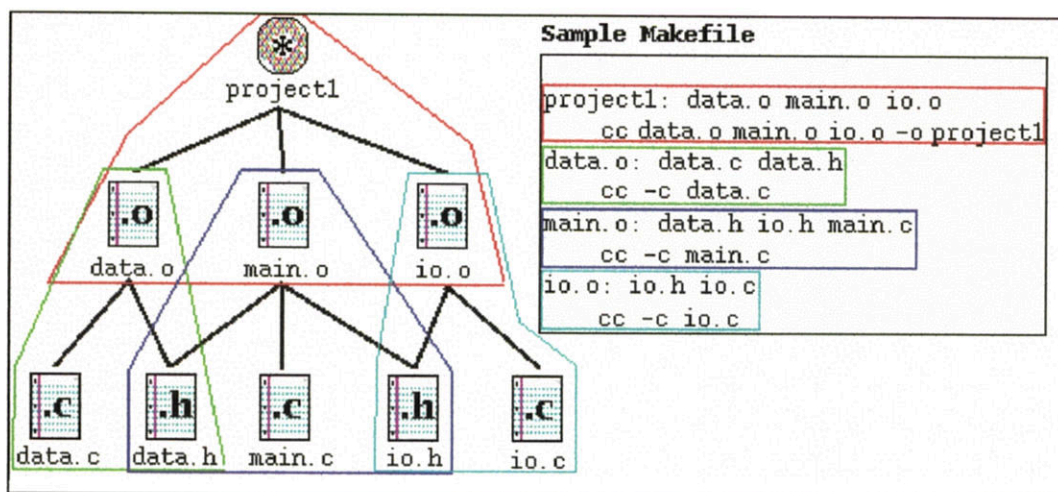


Figura 3.8 – Tradução da Dependência Gráfica

3.11. Conclusões

Através de todas essas abordagens teóricas, o leitor estará apto a entender o processo de desenvolvimento do projeto, como foi realizado o tratamento da imagem, a interface gráfica e a interligação de todos os programas criados para a formação do programa executável.

A partir de agora, analisaremos as características da câmera digital utilizada, explanando seu princípio de funcionamento e todo o protocolo de comunicação câmera – computador.

CAPÍTULO 4

A Câmera Digital – Connectix QuickCam

4.1. Introdução

Neste capítulo, será apresentado o dispositivo de captura de imagem, a câmera digital, seu princípio de funcionamento, seus recursos, protocolos de comunicação e procedimentos para a correta configuração para o perfeito funcionamento do dispositivo.

4.2. Descrição Física e Princípio de Funcionamento

4.2.1. Hardware QuickCam

A câmera QuickCam consiste de uma unidade central denominada detector CCD (charged coupled device). Esse detector de 324x423 pontos (pixels) recebe luz do ambiente, captada pelas suas lentes e transpassada por um filtro de infra-vermelho (IR). O campo de visão das lentes é aproximadamente equivalente ao de uma câmera de 35mm com lentes de 38mm.

Este filtro de infra-vermelho é necessário para compensar a extrema sensibilidade do CCD ao espectro infra-vermelho. Para se ter uma idéia da sensibilidade do sistema, se o filtro for retirado, a câmera é capaz de captar a imagem de um ambiente quando este estiver totalmente escuro, apenas iluminado por um LED infra-vermelho. Conseqüentemente, sem esse filtro, a câmera não conseguiria captar adequadamente a imagem em ambientes iluminados, apresentando uma tela totalmente branca, devido à grande sensibilidade luminosa.

A saída da porta paralela contém um microcontrolador e outros dispositivos eletrônicos que controlam a operação das complexas características da QuickCam. Como a porta paralela não apresenta qualquer saída de alimentação, a QuickCam

obtém sua energia através de um cabo conector da saída do teclado, cuja tensão de alimentação é de cinco volts.

4.2.2. Operação do CCD

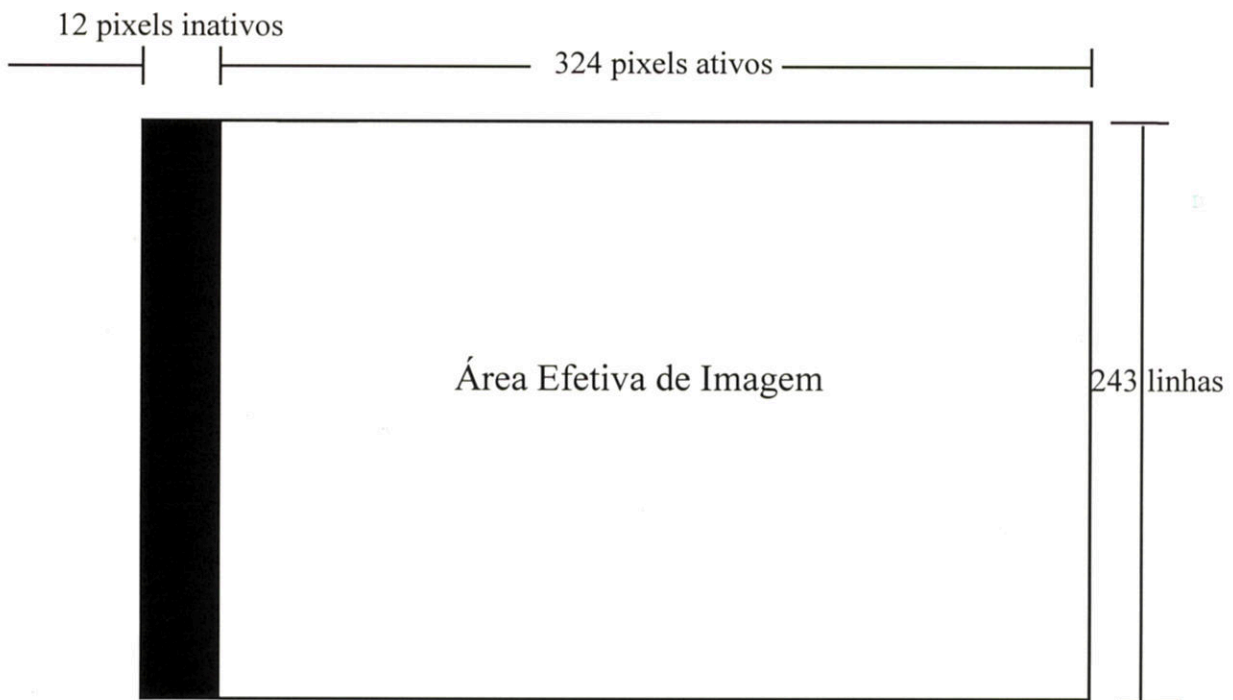


Figura 4.1 – Dimensões da Imagem do CCD

Fisicamente, o CCD (*Charged Coupled Device*) consiste de um array bidimensional de pontos (pixels) sensíveis à luz. Como na figura 4.1, o CCD utilizado pela QuickCam tem uma área de varredura de 336x243 pontos. As primeiras doze colunas de pontos são pontos negros, ou seja, são pontos que nunca recebem qualquer luminosidade. Como existem variações entre as partes do CCD, esses pontos negros oferecem uma referência de calibração padrão. A área de imagem efetiva da câmera fica então com 324 x 243 pixels (pontos sensíveis à luminosidade).

Quando em exposição, os pixels são carregados proporcionalmente com a intensidade luminosa que incide sobre cada um deles. Quanto mais longo o tempo de exposição, mais carregados serão os pixels.

A exposição desses pixels à luminosidade também é reconhecido como sendo um processo de integração, pois a carga desses pixels é incrementada com o tempo

de exposição. Conseqüentemente, se a exposição for longa o suficiente, todos os pixels alcançarão sua carga máxima, resultando em uma imagem completamente branca. Por outro lado, se a exposição for curta demais em um espaço de tempo, os pixels apresentarão uma baixa carga de energia, resultando em uma imagem escura, sem condições de reconhecimento. O ideal do tempo de exposição irá depender da intensidade luminosa do ambiente onde se encontra a câmera digital e outras condições, que devem ser ajustadas dinamicamente pelo aplicativo desenvolvido ou pelo usuário em questão [DURBIN & SHAFER, 1996].

Uma vez completado o intervalo de exposição, a carga armazenada em cada pixel do CCD é transferida para um capacitor de armazenagem (buffer de imagem), onde pode ser mantido por um período de tempo. A capacidade de armazenagem é temporária, fazendo com que toda a carga armazenada se “escoe”, e a imagem inicia um processo de enegrecimento até o seu desaparecimento total. A câmera QuickCam resolve esse problema jogando fora essa imagem armazenada, antes que ela enegreça, e expõe uma nova imagem, no lugar da anterior.

Os dados de imagem são inicialmente retirados do CCD como uma sinal analógico serial. Este sinal consiste de uma componente DC e uma ondulação (*ripple*), que corresponde aos pixels atuantes. Cada crista da ondulação corresponde a um pixel e a amplitude dessa crista é proporcional à intensidade da carga desse pixel. Um conversor A/D converterá esse sinal analógico em um sinal digital, que poderá ser mandado para o PC pela porta paralela.

Como o nível de offset irá variar de parte para parte, o CCD utiliza os pixels negros (que não recebem luminosidade) para calibrar esse nível de offset. Pela observação desses pixels negros é possível determinar o quanto do sinal analógico DC do CCD é offset e o quanto deste mesmo sinal representa a intensidade luminosa em todos os pixels.

Depois desse offset ser subtraído do sinal, o sinal restante, que consistirá na própria ondulação, é passada pelo conversor analógico digital (A/D). Um fator de escala pode ser aplicado a esse processo de conversão para determinar como a área de alcance (*range*) da tensão da ondulação realiza o mapeamento de todo o alcance digital. O efeito da mudança do fator de escala é a mudança no contraste visual da imagem.

Em um extremo do range da tensão observado, apresenta-se muito menor que o range do conversor A/D e a imagem apresentar-se-á cinza em toda a sua

constituição. Do outro extremo do range da tensão observado, apresenta-se muito maior que o range do conversor A/D, e a imagem não conterà qualquer tonalidade cinza, apenas níveis de preto e branco.

O contraste ótimo está em um valor entre esses dois extremos.

4.3. Conexões Físicas e Elétricas

4.3.1. Porta Paralela

A QuickCam comunica os comandos e informações de vídeo através da porta paralela do computador. Essa porta padrão pode ser unidirecional (PCs mais antigos), bidirecional, EPP (Enhanced Parallel Port) ou ECP (Enhanced Capability Port).

4.3.1.1. Modos Nybble (Unidirecional) e Byte (Bidirecional)

Uma porta paralela unidirecional contém 12 saídas, cinco entradas e oito pontos de terra (DB-25). Em portas bidirecionais, algumas das 12 saídas podem ser transformadas em entradas (observação: estas não poderão ser utilizadas como saídas e entradas ao mesmo tempo). A QuickCam não requisita as entradas extras providas pelas portas paralelas bidirecionais, mas essas linhas podem ser utilizadas para aumentar a taxa de transferência de dados. Primariamente, esta capacidade extra é usada pelos dados de vídeo, que vêm em blocos grandes e que necessitam ser transferidos rapidamente. Os comandos e parâmetros, entretanto, que são curtos e transmitidos sem muita frequência, utilizam apenas a capacidade unidirecional da porta paralela.

Para o desenvolvimento do programa da QuickCam, esses dois modos de operação serão denominados de Modo Nybble e Modo Byte.

O Modo Nybble é o único modo de operação para portas unidirecionais e é o modo usado pelos dois tipos de portas para transferência de comandos e seus parâmetros.

O Modo Byte concerne apenas às portas bidirecionais e é utilizado para aumentar a taxa de transferência dos dados da imagem.

O Modo Nybble utiliza quatro bits de entrada para o PC e o Modo Byte usufrui de doze bits de entrada.

4.3.1.2. Pinagem para o Modo Nybble

Fisicamente, a porta paralela é um conector fêmea do tipo DB-25. Em uma porta unidirecional, 12 dos 25 pinos são caracterizados como saídas, cinco como entradas e os oito restantes como linhas de terra. Originalmente, as portas paralelas em PC foram desenvolvidas como interface para a impressora, e somente depois se tornou uma interface para uso geral.

Como resultado, os sinais de descrição para os pinos da porta paralela, são as descrições dos sinais relacionados para a aplicação original desta, ou seja, para a impressora.

A QuickCam não utiliza os pinos da maneira que a impressora os utilizaria, então, essas denominações para esses pinos são impróprias. Para a utilização da QuickCam, esses pinos receberão denominações diferenciadas, com o objetivo de tornar a pinagem de interface QuickCam-PC mais amigável e compreensível para o desenvolvimento da programação. A tabela 4.1 apresenta a denominação original da pinagem da porta paralela, e a nova designação para a utilização da QuickCam.

Cmd[0-7] - Esses pinos são utilizados para transmitir os comandos de 8 bits e os parâmetros de 8 bits para a QuickCam.

Nybble[0-3] - Esses pinos são utilizados para ecoar os comandos de 8 bits (em duas partes de 4 bits), ecoar os parâmetros de 8 bits (também em duas partes de 4 bits), e para enviar os dados de vídeo (em pares de 4 bits) pela porta paralela unidirecional. Observa-se que, apesar do pino Nybble3 (BUSY) ser invertido (nível lógico baixo no software corresponde a nível lógico alto fisicamente), o software não precisa se preocupar com este fato, pois a QuickCam previamente inverte este bit antes de enviá-lo para o computador.

PCAck - Este pino é transitado pelo computador para permitir que a câmera reconheça que foi colocado um comando ou um parâmetro no barramento Cmd[0-7],

ou que o computador está pronto receber o primeiro par de Nybbles de dados de imagem. Este pino é novamente transitado para reconhecimento do eco do primeiro Nybble de um comando ou parâmetro, ou para reconhecimento da recepção do primeiro par de Nybbles de dados de imagem. Este pino é sempre transitado duas vezes em qualquer operação, então ele retorna ao seu estado inicial.

CamRdy1 - Este sinal é transitado pela QuickCam para indicar que a mesma colocou o primeiro Nybble do eco do comando ou parâmetro, ou o primeiro Nybble de um par de dados de imagem no barramento Nybble[0-3]. É transitado novamente para indicar que a QuickCam inseriu o segundo conjunto de bits na porta.

Reset_N - O pino de Reset é alterado de nível lógico alto para nível lógico baixo e de volta para alto, resetando a câmera. Este sinal deve permanecer em nível lógico alto, e é obrigação do software assegurar esta característica.

4.3.1.3. Pinagem para o Modo Byte

Se o PC possui uma porta paralela com característica bidirecional, ele é capaz de tornar o barramento de saída DB[0-7] em entradas, para aumentar a taxa de transferência de vídeo. Neste modo de operação, chamado Modo Byte, a interpretação dos sinais das linhas é diferente da interpretação em Modo Nybble, e então, tem diferentes nomes para os sinais. A tabela 4.1 apresenta a denominação original da pinagem da porta paralela, e a nova designação para a utilização da QuickCam. As designações dos sinais utilizados no Modo Byte são dados abaixo:

Data[0-11] - Este barramento transfere para o computador a palavra de 12 bits de dados de imagem da QuickCam. Observa-se que as informações de vídeo são sempre transmitidas em pares de palavras de 12 bits com complementador no final, se as informações requisitadas não estão divididas exatamente por 24 bits. Existe também um comando para testar estes 12 bits ecoando o comprimento de testes padrão arbitrários de volta para o PC.

Apesar de que seja descrito como 12 bits de barramento, dois conjuntos de leitura de 8-bits são requisitados pelos registradores da porta paralela para adquirir a informação. A primeira leitura, estabelecida pelo Data Register, lê os pinos Data[0-6]

do barramento, e a segunda leitura, do Status Register, lê os pinos Data[7-11] do mesmo.

PCAck - Transitado para deixar que a câmera saiba que está pronta para receber os dados de vídeo, ou que esta tenha recebido o primeiro par de 12-bits de dados de vídeo. Este sinal também é transitado no começo e no final da recepção dos dados de vídeo, para iniciar a transição de DB[0-7] de saídas para entradas e de volta para saídas.

CamRdy2 - Este sinal serve de mesma função como CamRdy1 em Modo Nybble. Transitado (baixo para alto) pela QuickCam para indicar ao computador que esta colocou os primeiros 12-bits do total dos 24 bits de dados de vídeo no barramento de saída Data[0-11], e transitado de novo (alto para baixo) para indicar que a segunda palavra de 12 bits do total de 24 bits foi inserida nas portas Data[0-11].

Reset_N: O mesmo que no modo unidirecional. Transição de alto nível para baixo nível e de volta, reseta a QuickCam.

Modos de Transmissão:		Modo Nybble		Modo Byte	
Denominação Original da Paralela	Pinos do DB-25	Denominações para a QuickCam	Direção	Denominações para a QuickCam	Direção
Strobe	1(-)	Sem uso	-	Sem uso	-
DB0	2	Cmd0	Saída	CamRdy2	Entrada
DB1	3	Cmd1	Saída	Data0	Entrada
DB2	4	Cmd2	Saída	Data1	Entrada
DB3	5	Cmd3	Saída	Data2	Entrada
DB4	6	Cmd4	Saída	Data3	Entrada
DB5	7	Cmd5	Saída	Data4	Entrada
DB6	8	Cmd6	Saída	Data5	Entrada
DB7	9	Cmd7	Saída	Data6	Entrada
ACKIN	10	Nybble2	Entrada	Data10/Nybble2	Entrada
BUSY	11(-)	Nybble3	Entrada	Data11/Nybble3	Entrada
PARAMEND	12	Nybble1	Entrada	Data 9/Nybble1	Entrada
SELECT	13	Nybble0	Entrada	Data 8/Nybble0	Entrada
Autofeed	14(-)	Sem uso	-	Sem uso	-
Error	15	CamRdy1	Entrada	Data7	Entrada
Initialize	16	Reset_N	Saída	Reset_N	Saída
SELECTIN	17(-)	PCAck	Saída	PCAck	Saída
Ground	18-25	Ground	-	Ground	-

Tabela 4.1 - Modo de Transmissão Unidirecional e Bidirecional

4.4. Interface do Software

4.4.1 Software de Interface da Porta Paralela

Para o software, a porta paralela é composta de 3 registradores de 8 bits, ocupando três endereços consecutivos no espaço de E/S. A seguir, apresenta-se os três registradores da porta paralela, e como esta é mapeada e acessada.

4.4.1.1. Tabela da Porta pela BIOS

A porta paralela é identificada pelos endereços de base de E/S, e também pelo número da porta LPT. A BIOS testa e checa endereços específicos de E/S para detectar a presença de uma porta paralela, e constrói uma tabela de endereços de E/S na memória baixa da área de informações da BIOS, começando nos endereços 0x0040:0008 (ou 0x0000:0408). Esta tabela inclui palavras de 16 bits. Cada entrada é o endereço de E/S base da porta paralela. A primeira palavra é o endereço base de E/S de LPT1, o segundo é o LPT2, etc. Se menos de quatro portas forem encontradas, as entradas restantes da tabela serão zero. DOS, e as funções de impressão da BIOS (acessada via int 17h), utilizam esta tabela para traduzir um número de porta LPT para uma porta física de um certo endereço.

4.4.1.2. Endereçamento da Porta

A porta paralela da placa de vídeo está normalmente em 0x03BC no espaço de endereço de E/S. Este endereço é checado primeiro pela BIOS, então se uma porta existe nele, será a LPT1. A BIOS checa então 0x0378, depois 0x0278. A porta pode ser endereçada diretamente, sem a BIOS, pelo tratamento da porta como 3 registradores de 8 bits em endereços adjacentes no espaço de E/S. Os registradores são definidos com relação ao endereço de base E/S, e estão em IOBASE+0,

IOBASE+1 e IOBASE+2. Sempre utilizando 8-bits de leitura ou escrita quando acessando estes registradores.

4.4.1.3. Registrador de Dados (Data Register)

Este registrador encontra-se em IOBASE+0. Pode ser lido ou escrito (usando por exemplo, as instruções IN e OUT, ou `inportb()` e `outportb()`, ou `inp()` e `outp()`). Escrevendo um byte neste registrador, ocasiona o aparecimento dos valores desse byte nos pinos 2 à 9 no conector D-Sub (a menos que a porta seja bidirecional e esteja setada em modo bidirecional). O valor escrito no registrador permanecerá travado até que outro valor seja escrito. Lendo este registro, o mesmo reproduzirá o estado desses pinos.

4.4.1.4. Registrador de Estado (Status Register)

O registrador de estado está no endereço IOBASE+1. Somente os bits 3-7 do registrador são definidos (bits 0-2 são indefinidos). Este registrador é somente para leitura e qualquer escrita será ignorada. A leitura da porta representa o estado dos cinco pinos de entrada do conector da porta paralela no tempo de acesso à leitura.

Observa-se que o bit 7 (pino 11, BUSY) é um pino invertido, então a tensão neste pino tem a polaridade oposta ao valor lido no mesmo. Uma das funções da QuickCam é inverter apropriadamente este valor, e mandá-lo para `Nybble3/Data11`. Dessa maneira, o software desenvolvido não necessita inverter este bit para adquirir o valor apropriado.

4.4.1.5. Registrador de Controle (Control Register)

O registrador de controle está no endereço IOBASE+2. Este registrador suporta leitura e escrita. Note que os pinos 1, 14 e 17 são invertidos. Nas portas que não possuem capacidade bidirecional, o bit 5 é indefinido e escrevendo nele não surte efeito algum. Em portas bidirecionais, setando o bit 5 em 1, causa a ida dos pinos 2 à

9 do DB-25 ao modo de alta impedância, assim as informações podem ser colocadas nestes pinos (*inputs*). Neste estado bidirecional, os valores escritos no data register serão estocados no registrador, mas não aparecerão nos pinos do conector. Lendo este registrador neste estado, dará o estado dos pinos como conduzidos por uma fonte externa. Em algumas máquinas existe passos adicionais requeridos para fazer dos pinos 2 à 9 entradas. Setando bit 5 retorna a porta ao modo unidirecional.

As tabela 4.2, 4.3 e 4.4 mostram a disposição dos sinais e a pinagem da porta paralela para os três registradores.

Registrador de Dados (Data Register)			
Denominação Padrão do Computador	Pinos do DB-25	Denominações para a QuickCam	Bits do Registrador
DB0	2	Cmd0/CamRdy2	0
DB1	3	Cmd1/Data0	1
DB2	4	Cmd2/Data1	2
DB3	5	Cmd3/Data2	3
DB4	6	Cmd4/Data3	4
DB5	7	Cmd5/Data4	5
DB6	8	Cmd6/Data5	6
DB7	9	Cmd7/Data6	7

Tabela 4.2 – Registrador de Dados

Registrador de Estado (Status Register)			
Denominação Padrão do Computador	Pinos do DB-25	Denominações para a QuickCam	Bits do Registrador
-	-	-	0
-	-	-	1
-	-	-	2
Error	15	CamRdy1/Data7	3
SELECT	13	Nybble0/Data8	4
PARAMEND	12	Nybble1/Data9	5
ACKIN	10	Nybble2/Data10	6
BUSY	11(-)	Nybble3/Data11	7

Tabela 4.3 – Registrador de Estado

Registrador de Controle (Control Register)			
Denominação Padrão do Computador	Pinos do DB-25	Denominações para a QuickCam	Bits do Registrador
Strobe	1(-)	-	0
Autofeed	14(-)	Contrast8/Offset8	1
Initialize	16	Reset_N	2
SELECTIN	17(-)	PCAck	3
(Interrupt control)	-	-	4
(Bidirectional control)	-	-	5
-	-	-	6
-	-	-	7

Tabela 4.4 – Registrador de Controle

4.4.2. Protocolo de Comunicação

4.4.2.1. Comandos da QuickCam

Como os comandos de controle da QuickCam são curtos e mandados relativamente sem frequência, a transferência de comandos é sempre realizada no Modo Nybble. Neste modo, existem 8 bits de saída e 4 bits de entrada para o computador.

Cada comando enviado para a câmera é ecoado devolta, e como existem apenas 4 bits no caminho de volta, o comando de eco é mandado como 2 palavras de 4 bits pelo barramento Nybble[0-3]. Todos os comandos da QuickCam mandam ou recebem parâmetros individuais (O comando SendVideoFrame começa como um comando mandando um parâmetro individual, mas possui um protocolo de leitura que é utilizado depois do parâmetro ser mandado para pegar a informação requerida).

Como o comando, este parâmetro é ecoado de volta pela QuickCam. Para comandos que mandam um parâmetro isolado, o protocolo da QuickCam é:

- 1) Mandar o comando para a QuickCam pelos 8-bits do barramento Cmd[0-7];
- 2) Pegar o Nybble mais significativo do comando ecoado de volta pelos Nybble [0-3];
- 3) Pegar o Nybble menos significativo do comando ecoado de volta pelos

Nybble[0-3];

- 4) Mandar o parâmetro pelos 8-bits do caminho Cmd[0-7];
- 5) Pegar o Nybble mais significativo do comando ecoado de volta pelos Nybble[0-3];
- 6) Pegar o Nybble menos significativo do comando ecoado de volta pelos Nybble[0-3];

Para Comandos que pegam um parâmetro da QuickCam, o protocolo é similar:

- 1) Mandar o comando para a QuickCam pelos 8-bits do caminho Cmd[0-7];
- 2) Pegar o Nybble mais significativo do comando ecoado de volta pelos Nybble[0-3];
- 3) Pegar o Nybble menos significativo do comando ecoado de volta pelos Nybble[0-3];
- 5) Pegar o Nybble mais significativo do parâmetro pelo Nybble[0-3];
- 6) Pegar o Nybble menos significativo do parâmetro pelo Nybble[0-3];

O protocolo para mandar dados de vídeo começa similarmente, apesar de que o protocolo para enviar atualmente os dados do quadro de imagem varia com o modo:

- 1) Mandar o comando SendVideoFrame para a QuickCam pelos 8bits de Cmd[0-7];
- 2) Pegar o Nybble mais significativo do comando ecoado de volta pelos Nybble[0-3];
- 3) Pegar o Nybble menos significativo do comando ecoado de volta pelos Nybble[0-3];
- 4) Mandar o parâmetro de transferência de vídeo para a QuickCam pelos 8bits de Cmd[0-7];
- 5) Pegar o Nybble mais significativo do comando ecoado de volta pelos Nybble[0-3];
- 6) Pegar o Nybble menos significativo do comando ecoado de volta pelos Nybble[0-3];
- 7) Mandar a tela de vídeo (video frame) (sequência de passos depende do modo).

Modelo do Ciclo de Comando

O modelo de ciclo de comando tem o seguinte padrão: mandar um comando, QuickCam ecoa este comando de volta ao PC, mandar um parâmetro para o comando, QuickCam ecoa o parâmetro de volta ao PC.

Comandos e parâmetros são enviados para a QuickCam pelos 8bits de Cmd[0-7] e a QuickCam ecoa de volta pelos 4bits de Nybble[0-3], então cada eco tem dois conjuntos de 4bits de um inteiro de 8 bits.

A figura 4.2 apresenta o modo de execução do ciclo de comando. Em detalhes, a sequência de passos que ocorrem em enviar um comando e seu parâmetro são:

- 1) Um comando é escrito na linha de comando Cmd[0-7] pelo computador;
- 2) O computador indica a validade do comando setando PCAck em nível lógico alto (baixo em software);
- 3) QuickCam ecoa a mais alta ordem de 4 bits do comando para a porta Nybble[0-3];
- 4) QuickCam seta em 1 CamRdy1 para indicar que o primeiro Nybble é válido;
- 5) O computador lê o primeiro Nybble da porta e seta em 0 PCAck para indicar que leu;
- 6) QuickCam ecoa a ordem mais baixa de 4 bits do comando para a porta Nybble[0-3];
- 7) QuickCam seta em 0 CamFdy1 para indicar que o segundo Nybble é válido;
- 8) PC lê o segundo Nybble pela porta.

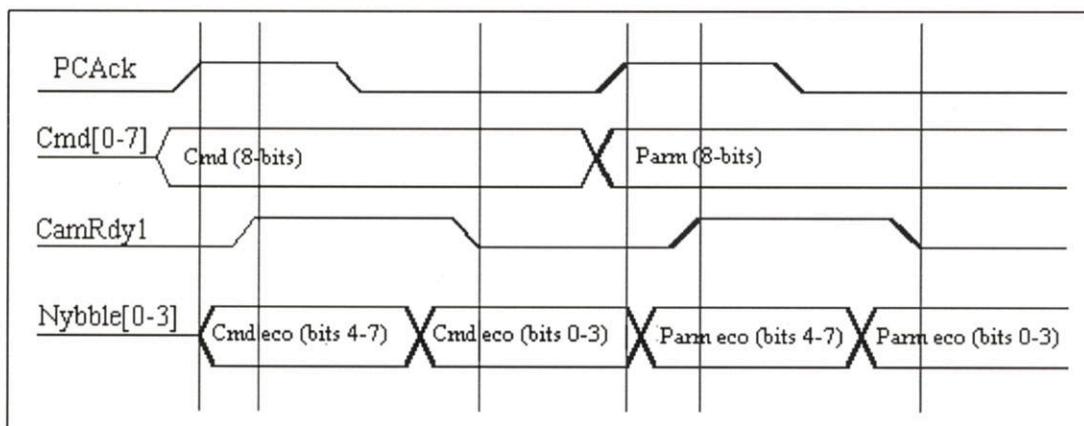


Figura 4.2 – Modelo do Ciclo de Comando

Os passos 1 à 8 são repetidos para enviar o parâmetro depois do comando, com o parâmetro sendo escrito em Cmd[0-7] e ecoado no Nybble[0-3];

Como os passos 1 à 8 são repetidos, a operação com a QuickCam enviando algum dado pelo Cmd[0-7] e recebendo um eco pode ser descrita abaixo. O pseudo-código para implementação dessa operação de envio é mostrado abaixo:

```
MandaComandoOuParâmetro(Data)
{
    /* Pega o mais e o menos significativo bit da palavra no Nybble */
    DataHigh = highNybble (Data);
    DataLow = lowNybble (Data);
    /* Escreve o comando de 8 bits na linha de comando da porta paralela */
    Write (Data, Cmd[0-7]);
    /* Delay para assegurar que o comando tenha setado na porta, então setar em
0 PCAck */
    delay ();
    set (PCAck, 0);
    /* Espera a QuickCam enviar o sinal colocando os primeiros quatro Nybble bits
no barramento Nybble, então lê e checa o Nybble */
    do
        { Ready = read(CamRdy1); }
    while(Ready !=1);
    Nybble = read (Nybble[0-3]);
    if (Nybble != DataHigh)
        HandleEchoError ();
    /* Reconhecimento do recebimento do primeiro Nybble, então espera e lê o
segundo */
    set (PCAck, 1);
    do
        { Ready = read(CamRdy1), }
    While (Ready !=0);
    Nybble = read(Nybble[0-3]);
    if (Nybble != DataLow)
        HandleEchoError();
}
```

Na prática, algumas portas alteram as linhas de Cmd[0-7] vagarosamente. Por isso, é recomendável que, depois de escrever no barramento Cmd[0-7], realize-se um pequeno atraso, antes de setar em zero o PCAck, assegurando assim, que o sinal PCAck vá a nível lógico baixo antes que as saídas da porta sejam estabilizadas. O recebimento de um alto percentual de ecos ruins da QuickCam, é um diagnóstico para este problema, devendo-se então, aumentar o delay.

Tendo definido a função, o modelo de escrita do ciclo é somente duas das operações de “back to back”, uma para o comando e outra para o parâmetro, como segue no pseudo-código:

```

MandaComando (Comando, Parâmetro)
{
    MandaComandoOrParâmetro (Comando);
    MandaComandoOuParâmetro (Parâmetro);
}

```

Comandos que Recebem Parâmetros

Dois dos comandos da QuickCam, GetOffset e SendVersion, adquirem o parâmetro da câmera. Para pegar um parâmetro da QuickCam tem-se o seguinte padrão: mandar o comando, QuickCam ecoa o comando de volta ao computador e este lê o parâmetro da mesma. O comando é enviado para a QuickCam pelos 8bits de Cmd[0-7] e ela ecoa esse comando de volta pelos 4bits Nybble[0-3]. Então cada eco pega dois conjuntos partidos de 4bits (correspondem à informação completa de 8 bits). O parâmetro é também um valor de 8 bits, então ele é quebrado em dois conjuntos de 4 bits para transmissão pelo barramento Nybble[0-3]. A figura 4.3 apresenta o ciclo de envio de comandos e parâmetros, para a câmera.

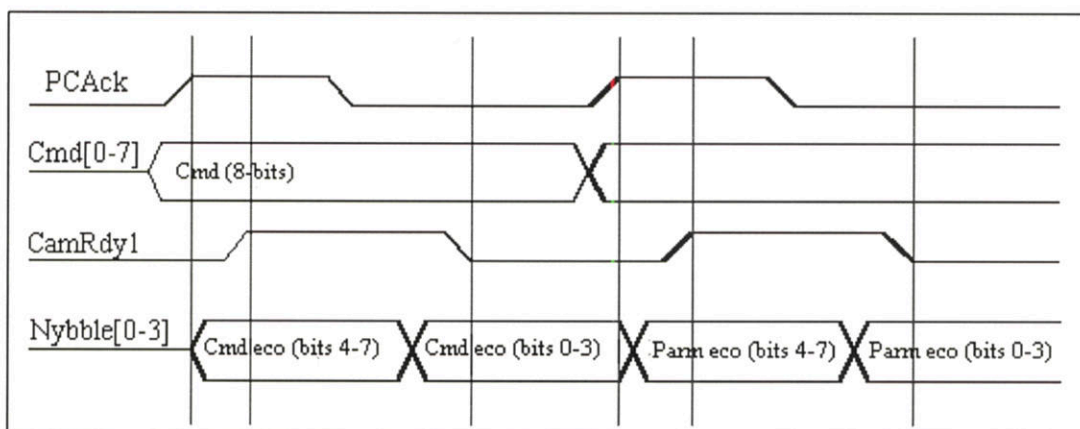


Figura 4.3 – Modelo do Ciclo de Envio de Comando e Parâmetro

A sequência de passos que ocorrem no envio de um comando e na aquisição do parâmetro pela QuickCam são:

- 1) Computador coloca o comando nas saídas Cmd[0-7];
- 2) Computador eleva PCAck (para alto, baixo no software) para indicar que o comando é válido;
- 3) QuickCam coloca os quatro bits mais significativos no Nybble[0-3];
- 4) QuickCam eleva CamRdy1 a 1 para indicar que os primeiros 4 bits são válidos;
- 5) Computador lê o Nybble mais significativos e baixa PCAck para indicar esta operação;
- 6) QuickCam coloca o quatro bits menos significativos no Nybble[0-3];
- 7) QuickCam abaixa CamRdy a zero para indicar que o segundo 4bits são válidos;
- 8) Computador lê o Nybble menos significativos e abaixa PCAck para indicar esta operação;
- 9) Computador transita PCAck de baixo para alto indicando que está pronto para receber o parâmetro;
- 10) QuickCam coloca a alta ordem Nybble (bits 0-3), do parâmetro, em Nybble[0-3];
- 11) QuickCam transita CamRdy1 de baixo para alto para indicar que o primeiro Nybble está pronto;
- 12) Computador lê o primeiro Nybble e transita PCAck de alto para baixo para indicar que foi lido;
- 13) QuickCam coloca a ordem baixa de Nybble (bits4-7) do parâmetro no Nybble[0-3];
- 14) QuickCam transita CamRdy1 de alto para baixo indicando que o segundo Nybble está pronto;
- 15) Computador lê o segundo Nybble.

Segue a seguir, o Pseudo-código:

```
LeituraDeParâmetro()  
{  
  /*Abaixa PCAck para indicar que está pronto para receber o parâmetro*/  
  set (PCAck, 0);
```



```

    /* Espera pela QuickCam sinalizar que colocou os primeiros quatro Nybble
bits no barramento Nybble, então lê */
do { Ready = read (CamRdy1); }
While (ready != 1);
highNybble = read(Nybble [0-3]);
/* Reconhece o primeiro Nybble e espera então pela leitura do segundo */
set (PCAck,1);
do { Ready = read(CamRdy1); }
While (Ready!=0);
lowNybble = read (Nybble[0-3]);
*/Concatena as duas partes no parâmetro de 8 bits */
parâmetro = concatena(HighNybble, lowNybble);
return (parameter);
}

```

Desta forma, enviando um comando e lendo o parâmetro resultado, pode ser completo com o pseudo código abaixo:

```

PegaParâmetroX()
{ Cmd = {o comando para mandar o ParâmetroX};
  MandaComandoOuParâmetro (Cmd);
  parâmetro=LeituraDeParâmetro();
  return(parâmetro); }

```

4.4.2.2. Dados de Vídeo

Um quadro de imagem é mandado pela QuickCam para o computador através da requisição deste (a câmera nunca enviará vídeo ou outra informação espontaneamente). Desde que um grande volume de dados deve ser movido para um driver de vídeo (tratamento de imagem), a transferência de dados de imagem é a preocupação primária, e um protocolo especial deve ser adotado.

Se uma porta bidirecional está disponível, as oito linhas DB[0-7], que são normalmente saídas, podem ser transformadas em entradas no decorrer da transferência. Isto aumentará a taxa de transferência por um fator de três (de quatro bits lidos para doze).

Dados de vídeo podem ser enviados com uma resolução de 4bits/pixel ou 6bits/pixel (16 ou 64 níveis de cinza). Em ambos os modos, Byte ou Nybble, a QuickCam compacta esses pixels no tamanho de quatro ou doze bits, para maximizar a transferência, e os transmite para o computador. Mais ainda, desde que todas as leituras ocorram em pares, despreocupando-se com o modo, o número de palavras (4bits Nybble ou 12 bits words) transmitidas serão sempre pares e o número total de bits transmitidos será sempre divisível por oito ou vinte e quatro. O número de bits de vídeo requisitado tem que ser um múltiplo de oito (Modo Nybble) ou vinte e quatro (Modo Byte). É responsabilidade do software desenvolvido saber o comprimento do bit atual.

Modo Nybble para Transferência

Os computadores mais antigos, que só possuem porta paralela unidirecional, somente utilizam o modo Nybble na transferência de dados. Nesse modo, existem quatro bits de entrada disponíveis na porta paralela por onde os dados de vídeo são movidos. Quando questionado, o comando SendVideoFrame instrui a QuickCam a transferir um quadro inteiro de vídeo para o computador. Este, então, questiona o comando SendVideoFrame com o parâmetro, descrevendo o tipo de transferência de vídeo desejada (incluindo como o vídeo subsequente deverá ser transferido em Nybble ou Modo Byte), usando o ciclo de comando padrão.

Depois do comando e do parâmetro terem sido mandados para a QuickCam, o computador indica que está pronto para receber os dados de vídeo, e a QuickCam começa a enviar pares de quatro bits pelo barramento Nybble[0-3] para o computador. A QuickCam continua a enviar pares de quatro bits Nybbles até que o número total de Nybbles requisitados tenham sido enviados (seção 4.4.3.3. para detalhes). Existe um sinal que indica o final da transferência da imagem (Fim do Quadro).

O software do computador deve computar o número de Nybbles esperados e então, cuidadosamente, contar o número de Nybbles recebidos e simplesmente parar quando todos os dados esperados tenham sido lidos. É importante notificar que, se o computador perder a conta do número de transferência, a QuickCam pode esperar para sempre por outro sinal para enviar ou ler mais dados. A figura 4.4 representa a sequência de envio dos respectivos sinais para realizar a transferência de imagem.

A sequência de passos que ocorrem no envio de dados de vídeo são:

- 1- Computador requisita o comando SendVideoFrame para a QuickCam e envia o parâmetro associado, usando o ciclo de comando padrão.

Para cada $N/2$ pares de Nybble de video data, os passos são repetidos:

- 2- Computador transita PCAck de baixo para alto para indicar que está pronto para receber os dados de vídeo.
- 3- QuickCam coloca o primeiro Nybble do par de Nybbles dos dados de vídeo no barramento Nybble[0-3].
- 4- QuickCam transita CamRdy1 de baixo para alto para indicar que o primeiro Nybble está pronto.
- 5- Computador lê o primeiro Nybble do barramento Nybble[0-3];
- 6- Computador transita PCAck de alto para baixo para indicar que está pronto para ler o próximo Nybble.
- 7- QuickCam coloca o segundo par de Nybble de dados de vídeo no barramento Nybble[0-3];
- 8- QuickCam transita CamRdy1 de alto para baixo para indicar que o segundo Nybble está pronto.

Observa-se que neste caso, como em todos os outros, a QuickCam sempre manda pares de Nybbles pelo barramento Nybble[0-3]. Isto significa que as linhas de sinal PCAck e CamRdy1 sempre terminam no mesmo estado que começaram. Como comentado inicialmente, os dados de vídeo requisitados podem não requerer um número peculiar de Nybbles para transmitir à QuickCam. O número N , de total de Nybbles transmitidos pela QuickCam para uma imagem de vídeo, devem ser computados pelo PC.

A transferência é terminada quando esses N Nybbles tenham sido transmitidos, então o computador deve computar corretamente N e a contagem de Nybbles enquanto elas forem recebidas. A computação de N é descrita na seção 4.4.3.3.

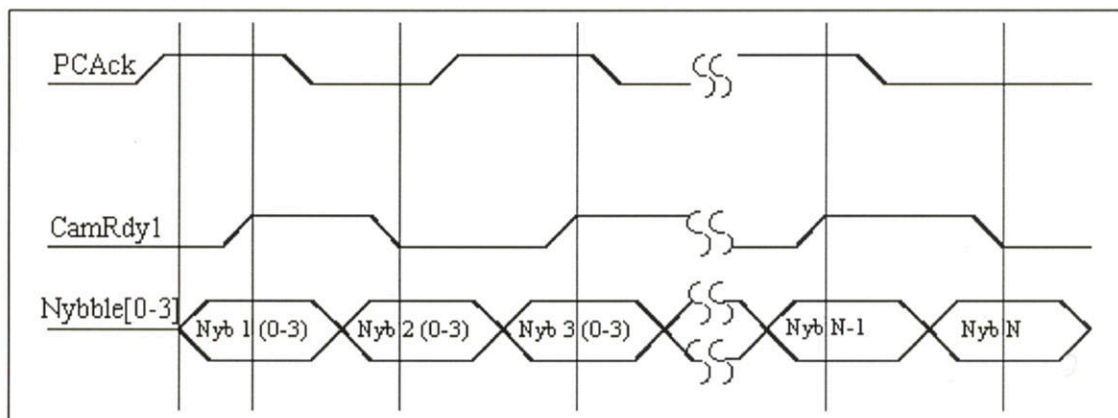


Figura 4.4 – Modelo do Ciclo de Envio dos Dados de Vídeo

Fim de Quadro

Uma vez que todos os dados de vídeo forem enviados para o computador, e antes que este mande um outro comando, deve enviar um último pulso para a câmera (figura 4.5), aumentando o nível de PCAck para um (no software, para zero). Com esta ação, a câmera retornará ao modo comando.

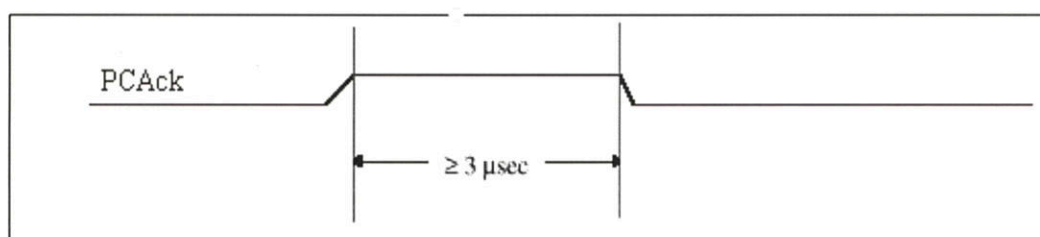


Figura 4.5 - Fim da Transmissão dos Dados de Vídeo

Em detalhes:

- 1- PC sinaliza um último pulso no final da transferência Nybble, elevando PCAck;
- 2- PC mantém PCAck alto por não menos que 3 micro-segundos;
- 3- PC abaixa PCAck para indicar que a câmera deve voltar ao modo comando.

Uma vez terminado o pulso de término de imagem, as comunicações são restabelecidas no modo Nybble com a QuickCam, usando CamRdy1 como seu sinal de handshaking. A câmera deve agora estar pronta para aceitar novos comandos.

Modo Byte de Transferência de Vídeo

Em computadores equipados com porta paralela bidirecional, a transferência de vídeo pode ser aumentada por um fator de três, através da transferência de dados pelo modo Byte.

Transferências em modo Byte transformam o barramento de saída DB[0-7] em barramento de entrada, aumentando o total de números de entrada de quatro para doze. Uma descrição do processo encontra-se abaixo:

- 1) Computador manda o comando SendVideoFrame para a QuickCam junto com o parâmetro que indica que a imagem será transferida no modo Byte (usando o ciclo de comando padrão);
- 2) Computador e a QuickCam transformam o barramento DB[0-7] de saída para entrada. Esse processo também transfere o handshaking de CamRdy1 para CamRdy2.
- 3) O quadro de imagem é transmitido para o computador em pares de 12 bits;
- 4) Computador e QuickCam transforma DB[0-7] em saída novamente. Handshaking transfere de CamRdy2 para CamRdy1, novamente.

Mudança na Porta

Depois do computador ter enviado o comando SendVideoFrame e o parâmetro indicando que a transferência será no modo Byte, este transforma o barramento DB[0-7] de saída para entrada e indica à QuickCam que o procedimento foi realizado (figura 4.6). Por conseguinte, a QuickCam move suas portas e sinais para o mesmo padrão.

Em detalhes, os passos desse processo são:

- 1- Depois da porta ser virada pelo computador, o mesmo aumenta o sinal de PCAck para nível lógico alto;
- 2- QuickCam vira sua porta correspondente e seus sinais através do aumento do nível lógico em CamRdy1.
- 3- O computador reconhece essa mudança baixando a zero o sinal de PCAck.
- 4- QuickCam abaixa CamRdy1 para restabelecer a condição padrão. (PCAck alto e CamRdy1 baixo).

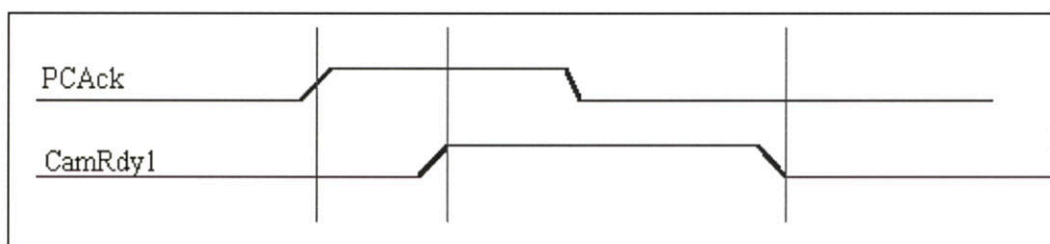


Figura 4.6 – Processo de Mudança da Porta Paralela

No final desse processo, CamRdy2 é o sinal de handshake para a QuickCam (CamRdy1 foi convertida em bit de dado Data7) e a QuickCam está pronta para começar o envio de imagem.

Transferência de 12 Bits

A transferência de 12 bits de vídeo no modo Byte é igual à transferência de dados de vídeo em modo Nybble, exceto pelo fato que CamRdy2 é usado pela QuickCam para handshaking, e os dados são transmitidos em pares de doze bits ao invés de quatro bits.

Assumindo que o computador já requisitou a operação pelo comando SendVideoFrame e seu parâmetro, e já foi estabelecido a mudança no barramento, a sequência de passos que ocorrem para mandar o quadro de imagem são:

Para cada $M/2$ pares de 12 bits de palavra de vídeo, os seguintes passos são repetidos :

- 1- O computador transita PCAck de baixo para alto para indicar que está pronto para receber os dados de vídeo;

- 2- QuickCam coloca a primeira palavra do par de palavras dos dados de vídeo no barramento Data[0-11];
- 3- QuickCam transita CamRdy2 de baixo para alto para indicar que o primeiro Nybble está pronto.
- 4- O computador lê a primeira palavra no barramento Data[0-11];
- 5- Computador transita PCAck de alto para baixo para indicar que está pronto para receber a próxima palavra.
- 6- QuickCam coloca o segundo par de palavras de dados de vídeo no barramento Data[0-11];
- 7- QuickCam transita CamRdy2 de alto para baixo para indicar que o segundo Nybble está pronto.
- 8- O computador lê o segundo Nybble.

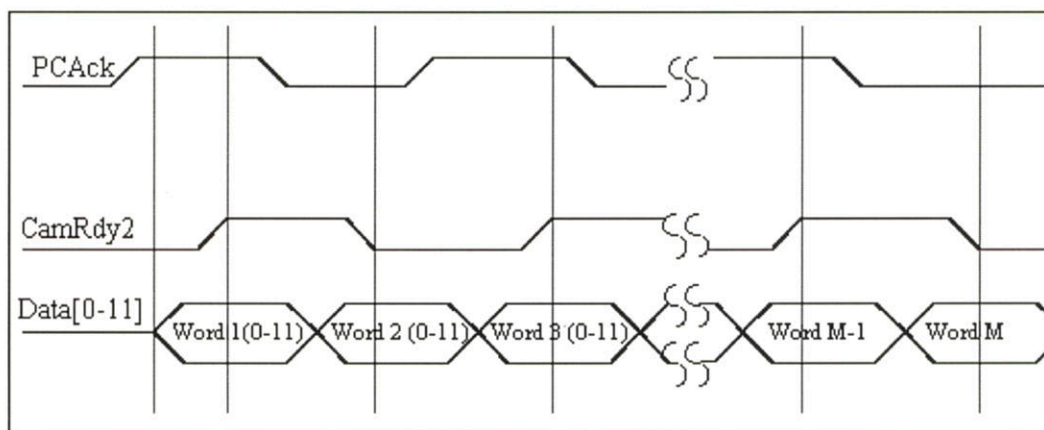


Figura 4.7 – Transferência de 12 Bits

Observa-se que, como em todos os outros casos, a QuickCam sempre manda pares de palavras no barramento Data[0-11]. Isto significa que os sinais das linhas PCAck e CamRdy2 sempre terminam no mesmo estado que começaram (figura 4.7).

É também importante salientar que o total de dados de imagem podem não requerer um número peculiar de palavras. A QuickCam tem que ser setada de modo que a tela sempre termine no final do ciclo de transferência. Em outras palavras, as transferências devem ocorrer sempre em pares. O número M , de total de palavras transmitidas pela QuickCam para o quadro de imagem, devem ser computadas pelo computador.

A transferência é terminada quando essas M palavras tiverem sido transmitidas, então o computador deve computar corretamente M , após então, contar as palavras enquanto as recebe. A computação de M é descrita na seção 4.4.3.3.

Fim de Quadro

Uma vez que todos os dados de vídeo foram transmitidos ao computador, e antes que este envie um outro comando, deve-se virar seus pinos DB[0-7] de volta para condição de saída (precisa retornar ao modo Nybble). O computador indica que irá mudar a porta pelo aumento do nível de PCAck, e indica que completou a operação, baixando o PCAck. A figura 4.8 apresenta o modo de transmissão do sinal PCAck para indicar o término da transmissão dos dados de vídeo.

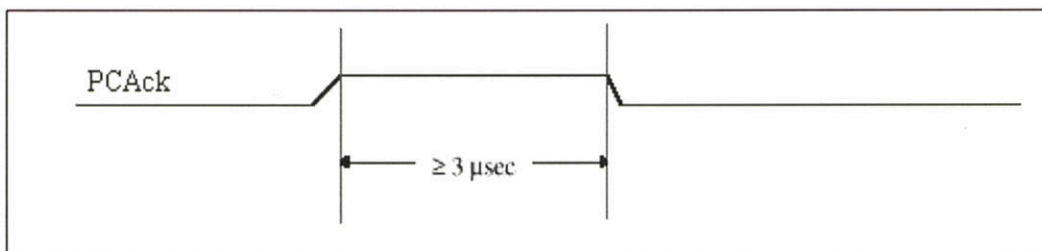


Figura 4.8 – Fim da Transmissão dos Dados de Vídeo

Em detalhes:

- 1- Computador sinaliza suas intenções de virar a porta pelo aumento de PCAck;
- 2- Vira a porta;
- 3- Computador mantém PCAck alto por não menos de 3 micro-segundos;
- 4- Abaixa PCAck para indicar que a porta foi virada.

Uma vez enviado esse pulso de término de imagem, as comunicações são retornadas ao modo Nybble com a QuickCam usando CamRdy1 como sinal de handshaking. QuickCam deve agora estar pronta para aceitar novos comandos. É importante que este protocolo seja seguido para virar a porta, para evitar dano potencial para a câmera ou para o computador.

4.4.3. Operação da QuickCam

As especificações de operação da QuickCam irão variar de aplicação para aplicação. Nesta seção, a operação básica da QuickCam em uma aplicação típica será descrita. Cada aplicação que usa a QuickCam deve seguir os seguintes passos:

- Setup PCAck
- Reset QuickCam
- Checa versão
- Setup Contraste
- Autocalibração CCD offset
- Setup Exposição
- Setup Tamanho do Quadro
- Receber o Quadro de Imagem

É importante que esses passos sejam seguidos na ordem correta. Por exemplo, é importante que o contraste seja setado corretamente antes da autocalibração do offset. As seções que seguem descrevem esses passos em detalhes.

4.4.3.1. Inicialização da QuickCam

Existem três passos principais que devem ser realizados na ordem certa. O primeiro deles é assegurar que PCAck é setado em nível baixo, e o Reset_N setado em alto, para portas bidirecionais, assegurando dessa forma que o barramento DB[0-7] está setado como saída.

A QuickCam deve então ser resetada colocando o Reset_N em nível baixo por um curto período de tempo (a duração exata desse pulso não é importante). Finalmente, a versão deve ser obtida da QuickCam, utilizando o comando SendVersion. Isto irá confirmar que a QuickCam está operante e permite o software

verificar se a QuickCam é uma versão suportada. A versão retornada pela QuickCam, para o computador, é a de valor 0 (zero).

4.4.3.2.Setup Geral

Contraste

O contraste no CCD é a escala fator para a porção AC do sinal de saída de vídeo do CCD (ver seção 3.1.2). Desde que a função de autocalibração de offset assume que o contraste é setado apropriadamente, o contraste deve ser setado antes da autocalibração de set up ser configurada. O comando SetContrast é usado para setar o nível de contraste.

Na maioria das aplicações, este contraste será apropriado e não existirá necessidade de ajustá-lo. Como a câmera é muito sensível ao contraste, se uma aplicação entrega ao usuário qualquer controle sobre o nível de contraste, é importante que exista uma opção para o mesmo resetar o contraste para o seu valor padrão.

Offset (Black Level)

O nível de escuridão do CCD permite que se ajuste a porção DC (offset) do sinal de vídeo do CCD. Desde que existam variações entre CCDs, o CCD tem a construção setada em pixels que não recebem luz. Esses pixels podem ser usados para compensar essa variação do CCD. QuickCam tem um comando implementado que automaticamente calibra o offset, baseado nesses pixels que não recebem luz. Todas as aplicações devem chamar AutoCalibrateOffset.

Desde que o AutoCalibrateOffset é um procedimento com muitos passos, sua complexidade toma algum tempo. O computador deve nomear a QuickCam para determinar quando está terminada o processo de auto-calibração, antes que se possa emitir comandos adicionais. Isto é feito pela emissão repetida do comando GetOffset

e examinando o parâmetro retornado. O valor de offset retornado será de 255 até que a autocalibração esteja completa, no determinado tempo em que será mudado para o valor de autocalibração (um valor entre 1-254).

Como no contraste, a câmera é completamente sensível ao nível de offset. Para a maioria das aplicações, o procedimento de AutoCalibrateOffset é apropriado e deve ser usado.

É possível setar o offset diretamente com o comando SetOffset. Se em aplicações permitem o usuário de alguma maneira mudar o valor de Offset, deve sempre prover uma maneira para o usuário de resetar o offset para o nível de calibração.

Exposição (Tempo de integração)

A duração do tempo que o CCD carrega-se pela recepção de fótons, corresponde ao tempo de exposição (duração em que o obturador permanece aberto) em uma câmera fotográfica comum. Na maioria das aplicações, o tempo de exposição será controlado pelo usuário, com o tempo apropriado de exposição, luz e outras condições. O extensão dos valores válidos de tempo de exposição são entre 1 e 254, que correspondem ao tempo de exposição de 1/1000 segundos à um segundo, respectivamente.

Quando uma imagem for exposta pela duração desejada, é salva em um buffer de imagem no CCD. Este armazenamento capacitivo descarrega-se com o tempo, degradando a qualidade da imagem armazenada. A QuickCam calcula o tempo desde a armazenagem dos dados no buffer de imagem e descarta-a antes que se torne degradada, expondo, em seguida, uma nova imagem. Observa-se que este processo não envia nenhum dado. A QuickCam somente manda dados se requisitada. Esse processo apenas assegura uma imagem de qualidade quando o buffer é requisitado.

Pulsção (Heartbeat)

Enquanto expondo as imagens, a QuickCam é incapaz de obedecer a outros comandos. Comandos emitidos enquanto a QuickCam está expondo a tela não receberão o CamRdy1 handshake de volta da QuickCam até que ela tenha exposto a mesma. Desde que o tempo de exposição pode ser de até um segundo, pode existir um período considerável de não resposta. Dessa forma, quanto mais longo o tempo de exposição, menor a capacidade da QuickCam de responder aos comandos.

Durante o tempo em que a QuickCam não reconhece os comandos, ela periodicamente se comunica com todos os bits do barramento Nybble[0-3] para indicar que ainda está operante (esse sinal periódico é reconhecido como heartbeat). Software deve monitorar o heartbeat (os bits do barramento de Nybble[0-3] ou um bit específico da porta) enquanto espera pelo reconhecimento do comando pelo CamRdy1.

Se a câmera não envia a CamRdy1 handshake e o bit monitorado de Nybble[0-3] não muda de estado em quinhentas leituras do barramento Nybble[0-3], a QuickCam está provavelmente inoperante. Nesse caso, o reset deve ser realizado através do Reset_N.

4.4.3.3. Aquisição dos Dados de Imagem

Os seguintes passos são necessários para adquirir a imagem da QuickCam:

- 1) Decidir que região de imagem do CCD pegar (Top, Left, NumLines, PixelPerLine);
- 2) Escolher profundidade de imagem desejada: 4 ou 6 bits (16 ou 64 níveis de cinza);
- 3) Escolher um modo de barramento de dados: 4 ou 12 bits (Nybble ou Byte mode);
- 4) Escolher um modo de transferência: 1:1, 2:1, 4:1;
- 5) Computar o número de transferências por linha baseados nos itens 1 à 4;
- 6) Setup frame, profundidade, modo de barramento e modo de transferência;
- 7) Adquirir os dados de vídeo.

Escolhendo uma Região para Captura de Imagem

Para o propósito de escolher uma tela, utiliza-se as variáveis Top, Left, NumLines e PixelPerLines. Essas variáveis especificam uma região retangular do CCD (figura 4.9). Apenas informação dentro do definido quadro será transmitida ao computador quando o comando SendVideoFrame for mandado.

Existem três modos de transferência. O primeiro adquire todos os pixels de todas as linhas. Os outros dois modos de transferência reduzem o total de dados transmitidos através do descarte de informações. O segundo modo descarta alternadamente pixel e linha, ou seja, a cada pixel ou linha lida, o modo descarta o pixel ou linha subsequente. O terceiro modo de transferência, descarta três pixels e três linhas de cada quatro pixels e quatro linhas lidas.

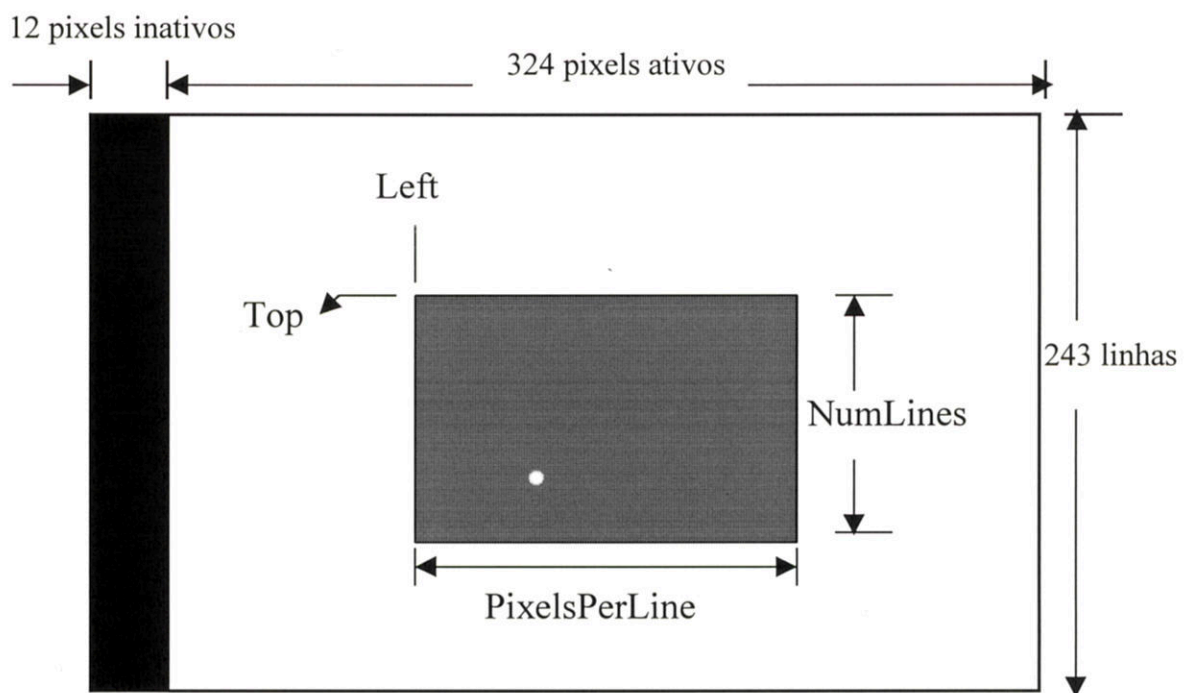


Figura 4.9 – Escolha da Região de Captura da Imagem

Esses modos de transferências mandam dados correspondentes à região retangular do CCD, especificado pelos Top, Left, NumLines e PixelPerLines, mesmo alguns dados desta região são descartados. Esses parâmetros são definidos mais explicitamente a seguir:

Top: esse parâmetro especifica o número da linha que se deseja começar a aquisição de dados. Valores válidos entre 1 e 243;

NumLines: é a densidade da região do CCD que deseja-se mandar. Valores válidos entre 1 e 243. NumLines nunca será enviado para a QuickCam para especificar o quadro de imagem. Assim, NumLines será dividido pelo fator decimal (1, 2 ou 4), para obter o valor de LinesToTransfer. LinesToTransfer é o parâmetro que irá especificar a densidade da tela para a QuickCam;

Left: é o local do primeiro pixel para ser transferido em uma dada linha. Left é referenciado pelo lado esquerdo absoluto do CCD, e não primeiro o esquerdo da imagem efetiva. Left=2, então, é o segundo pixel escuro em uma dada fileira, e Left=14 é o segundo pixel ativo de uma dada fileira. Desde que os parâmetros para a QuickCam podem ser um byte em tamanho, somente valores de Left são permitidos e o número atual que irá ser mandado para a QuickCam como o parâmetro LeftDiv2=Left/2. Desta forma, LeftDiv2=1 é o segundo pixel escuro de uma dada coluna e LeftDiv2=7 é o segundo pixel ativo de uma dada coluna.

PixelPerLine: é a largura da tela em pixels (valores entre 1 e 336). De novo, QuickCam somente recebe byte em tamanho determinado, então o parâmetro que irá ser mandado para a QuickCam não é o PixelPerLine, mas o TransferPerLine, que é um parâmetro na gama de valores entre 1-255. Desde que o cálculo de TransferPerLine envolve uma série de outros parâmetros, este será descrito mais tarde, na seção 4.3.3.5.

Densidade de Tela

É a resolução da escala cinza de uma imagem. QuickCam usualmente suporta 4-bits (16 escalas de cinza) ou 6-bits (64 escalas de cinza) por pixel. A densidade da tela é especificada no bit 1 do parâmetro do comando de SendVideoFrame, como descrito na seção 4.4.3.7.

Modo de Barramento

Modo de barramento é um tipo de transferência, Byte ou Nybble, descrito na seção 4.3. Lembrando que a porta paralela unidirecional somente suporta o modo Nybble e que este modo é de 4bits por leitura e o modo Byte é de 12 bits por leitura. O modo de barramento é especificado no bit 0 do parâmetro do comando de SendVideoFrame.

Modo de Transferência

Existem três modos de transferência suportados pela câmera: 1:1, 2:1 e 4:1. Em modo 1:1, cada pixel da área selecionada será mandado. No modo 2:1, adquire-se uma imagem cobrindo a mesma área de visão, mas apenas com a metade dos pixels, isto é, a câmera joga fora pixel e linhas alternadas. No modo 4:1, adquire-se uma imagem da mesma área de visão, mas apenas com $\frac{1}{4}$ de pixel, ou seja, a câmera joga fora três de cada quatro pixel e três de cada quatro linhas. Essa modo de transmissão permite uma maior taxa de exposição de imagem com uma menor perda de resolução aparente. O modo transfer é especificado no bit2 ou bit3 do parâmetro do comando SendVideoFrame.

Observa-se que os valores NumLines e PixelPerLines são valores calculados com referência na área do CCD, antes de aplicar o fator de redução de dados de 1, 2 ou 4. LinesToTransfer e TransferPerLine, entretanto, são computadas depois de se aplicar o fator redutor de dados.

Cálculo de TransferPerLine

Tendo decidido pelo quadro desejado (dimensões), profundidade de pixel, modo de barramento, modo de transferência, o parâmetro TransferPerLine pode ser computado como segue:

$$\text{TransfersPerLine} = \frac{\text{PixelPerLine} * \text{ImageDepth}}{\text{BitsPerTransfer} * \text{TransferMode} * \text{6bit-nybble special case factor}}$$

TransfersPerLine: deve ser um inteiro (isto é, bits por linha devem ser divididos por 8 ou 24). Se o número de bits em uma linha não é divisível por 8 ou 24, pega-se o próximo valor maior inteiro como TransfersPerLine.

ImageDepth: é também 6 ou 4 bits por pixel. No Modo Nybble, cada leitura pega 4 bits de dados de imagem, mas como a leitura sempre é feita em pares, um simples transferência é de 8 bits. Similarmente, no Modo Byte, cada leitura pega 12 bits, então uma transferência é de 24 bits. Desta forma, BitsPerTransfer é 8 ou 24.

TransferMode: é o fator pelo qual o número total de pixels na tela não computada, é dividida antes da transmissão, uma, duas ou quatro vezes (não descartar os pixels, descartar pixels e linhas alternadas, ou três de cada quatro pixels e linhas). Por último, existe um fator que é somente usado no Modo Nybble em 6 bits (64 cores). Neste modo, um fator adicional de 1/3 é usado para computar o valor final de TransfersPerLine. TransfersPerLine deve ainda ser um inteiro, então uma cuidadosa seleção do tamanho da tela e modos de transferência é requerido.

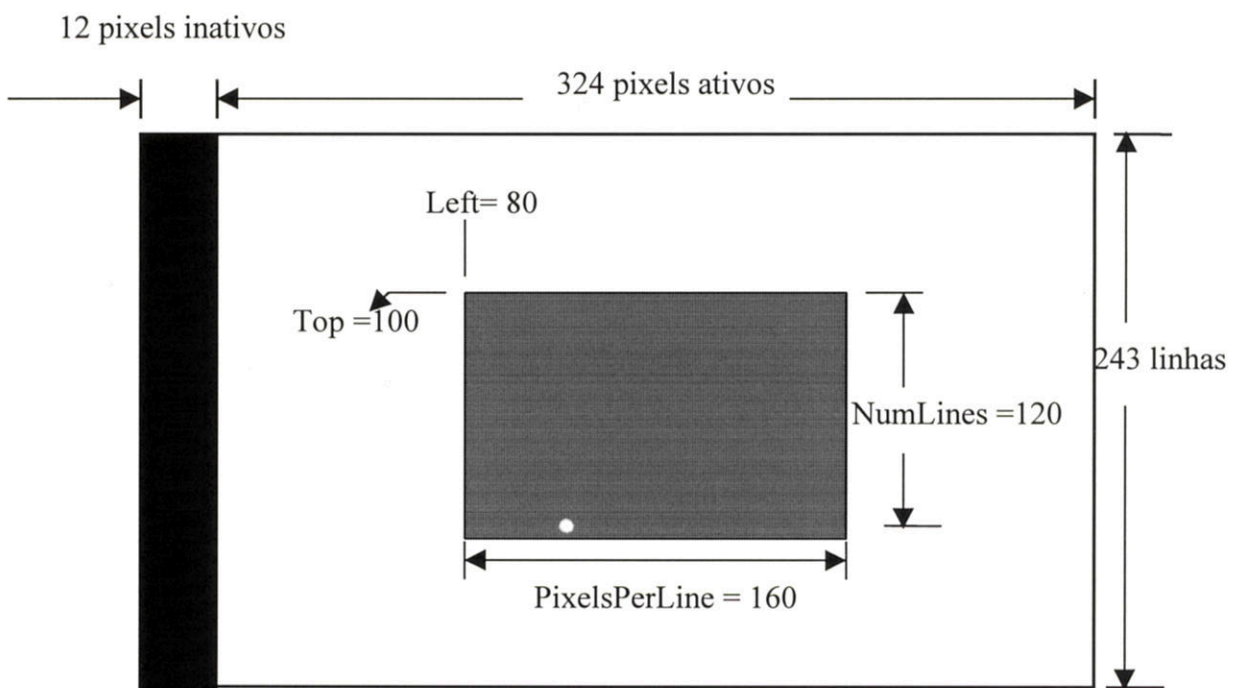


Figura 4.10 – Exemplo do Cálculo de TransferPerLine

Por exemplo, considerando a tela de pixel da figura 4.10, com 160x120. Se esta tela é retirada como imagem de 6 bits (64cores) usando modo de transferência 2:1 (jogar fora pixel e linhas alternadamente) e o modo de barramento Nybble (8bits/transferência), $\text{TransferPerLine} = (160*6)/(2*8)/3=20$. A mesma tela mandada pelo modo byte 1:1 irá requerer 40 transmissões: $\text{TransferPerLine} = (160*6)/(1*8)/3=40$.

Para ver como se manipula com TransfersPerLine não-inteira, considere se a largura da tela foi de 158 em vez de 160 nesses dois casos acima. No primeiro caso $\text{TransfersPerLine} = (158*6)/(2*8)/3 = 19,75$. Então pega-se o inteiro maior mais próximo e faz-se $\text{TransfersPerLine}=20$. Similarmente, a mesma tela mandada no modo de barramento byte em 1:1 com apenas 158 pixels por linha, terá $\text{TransferPerLine} = 39,5$. Pegando o número inteiro maior mais próximo faz-se $\text{TransferPerLine} = 40$. Tendo juntado todos os dados de pixel, teremos informações extras de pixel que poderão ser ignorados.

O método relatado logo acima, para a manipulação de TransferPerLine não-inteira, informado pela empresa, não funciona perfeitamente. Para determinados valores de dimensão de tela, cuja variável torna-se não inteira, surge um problema de concordância entre os modos de transmissão unidirecional e bidirecional com a variável ImageDepth em 4 (qualidade normal – quatro bits por pixel).

O sistema, com este método, funciona perfeitamente até o ponto onde o usuário alterará a aquisição dos dados de imagem com qualidade de 4 bits por pixel ($\text{ImageDepth}=4$), do modo unidirecional para o modo bidirecional. Realizando essa alteração, a imagem projetada na tela contorce-se completamente.

Este fato decorre da má distribuição dos pixels nas colunas da imagem. No caso, o número de colunas não corresponderá ao número de pixels por linha. Por exemplo, para o caso de estabelecer a largura da tela de 320 pixels, o resultado de TransferPerLine , já arredondado, resultaria em 54. Entretanto, a imagem aparece distorcida, pois a tela gerada pela QuickCam apresenta um pixel a menos por linha, fazendo com que o primeiro pixel da segunda linha venha a ser depositado no último pixel da primeira linha. Por consequência, sobrarão dois pixels nas duas últimas colunas da segunda linha, fazendo com que os dois pixels da terceira linha sejam inseridos nesta posição. Desta forma, em um processo cumulativo, a imagem é puxada lateralmente, distorcendo-a completamente, como mostra a figura 4.11.

Para corrigir este erro, é necessário realizar uma redução no valor de PixelsPerLine, ou seja, quando este for de 320 pixels, deve ser reduzido para 318 pixels. Desta maneira, o cálculo de TransferPerLine resulta em um número inteiro (53) e a tela é gerada no tamanho adequado para a inserção da imagem.

Observação: Para cada valor de PixelPerLine estipulado que não resulte um TransferPerLine inteiro (para o caso de modo de transferência bidirecional e qualidade de imagem de 4 bits por pixel), deve ser realizada essa correção. Para PixelsPerLine = 160, o novo valor que resulta em uma imagem perfeita é de 156.

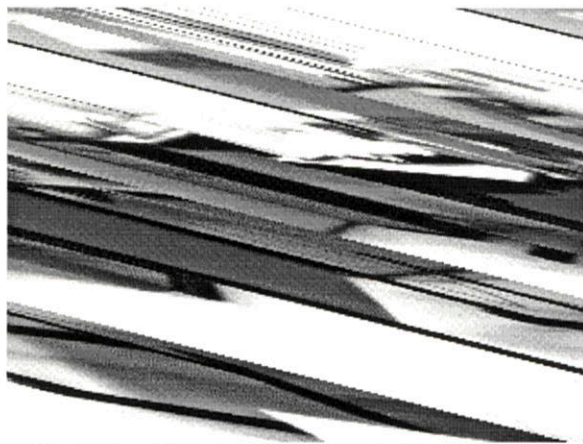


Figura 4.11 – Erro na Configuração de TransferPerLine

Set up da Tela

Depois de escolher a tela, modo de transferência, e calculado TransfersPerLine, a tela é comunicada à QuickCam usando os comandos SetTop, SetLeft, SetNumV e SetNumH, com os seguintes parâmetros:

SetTop = Top

SetLeft = LeftDiv2 = Left/2

SetNumV = LinesToTransfer = NumLines/ TransferMode

SetNumH = TransferPerLine =
$$\frac{\text{PixelsPerLine} * \text{ImageDepth}}{\text{BitsPerTransfer} * \text{TransferMode} * \text{6bit-nybble factor(special case)}}$$

Tratamento dos Dados de Vídeo

Dados de vídeo são requisitados da QuickCam utilizando o comando SendVideoFrame. Existe um número de passos nesse processo:

- 1) Enviar o comando SendVideoFrame para a QuickCam;
- 2) Enviar o parâmetro de 8 bits associado para a QuickCam (descrito abaixo);
- 3) Utilizando o modo de transferência de barramento, repetir o número total de transferências (TotalTransfers):
 - Pegar os primeiros 4 ou 12 bits de um par da QuickCam;
 - Pegar os segundos 4 ou 12 bits de um par da QuickCam;
 - Descompactar os pixels dessas palavras.

Depois de mandar o comando SendVideoFrame, a QuickCam é informada do modo de barramento (Byte ou Nybble), a intensidade dos pixels (4 ou 6 bits), e o modo de transferência (1:1, 2:1, 4:1) pelo setamento apropriado dos bits do parâmetro. Esses bits são definidos como mostra a tabela 4.5.

Bit 0:	0 para transferência Nybble e 1 para transferência Byte.
Bit 1:	1 para 6-bits de intensidade e 0 para 4 bits de intensidade;
Bit 2 e 3:	bit2=0 e bit3=0 utiliza 1:1 (manda todos os pixels da tela)
	bit2=0 e bit3=1 utiliza 2:1(joga fora pixels e linhas alternadas)
	bit2=1 e bit3=0 utiliza 4:1 (joga fora 3 de 4 pixels e linhas
	bit2=1 e bit3=1 configuração inválida
Bit 4:	1 para mandar teste padrão e 0 para não enviar
Bit 5, 6 e 7:	sem uso.

Tabela 4.5 – Configuração do Comando SendVideoFrame

Compactação dos Pixels

Depois do comando e do parâmetro serem mandados, cada leitura subsequente retém os pixels da QuickCam. Cada execução do loop é um ciclo de transferência e o total de números de transferência é computado:

$$\text{TotalTransfers} = \text{TransfersPerLine} * \text{LinesToTransfer} * 3[\text{se 6 bpp (bits por pixel)}]$$

Cada leitura vêm em 4 bits ou 12 bits partidas (então uma transferência é de 8 ou 24 bits). A resolução da imagem (dados de vídeo), contudo, é de 4 ou 6 bits. De maneira para maximizar a transferência, as informações dos pixels são compactadas em 4 ou 12 bits partidos para obter a máxima densidade. A figura 4.12 ilustra a compactação do pixel de 4 bits e o pixel de 6 bits, utilizando o modo de barramento Nybble.

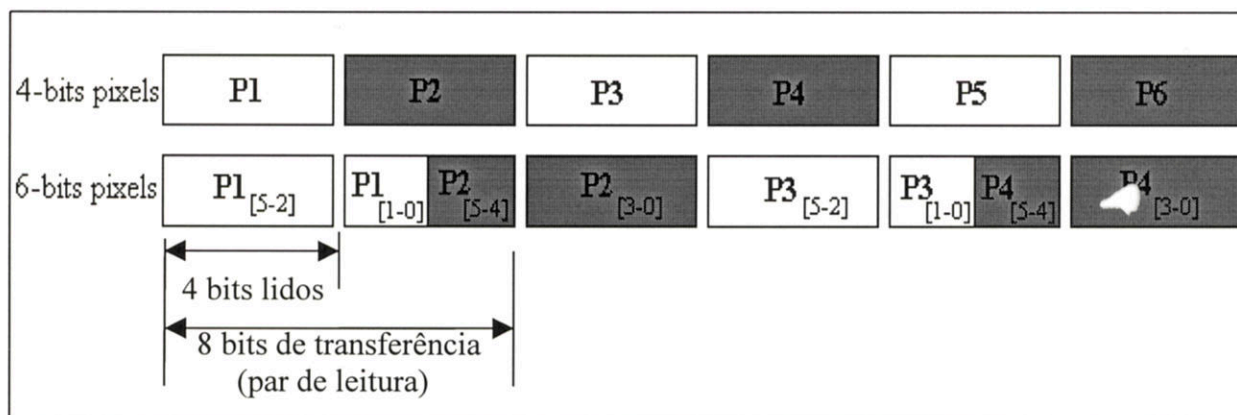


Figura 4.12 - Modo de Compactação para Modo Nybble

Na figura 4.12, P1, P2,...PN são os pixels na ordem da esquerda para a direita e do topo para baixo. A ordem de bits nos pixels com 4 bits de cada leitura é ilustrada com subscrição. Em geral, baixa ordem de bits são para a direita e alta ordem para a esquerda.

A compactação pela transferência de modo Byte é consideravelmente mais complicada. Compactando para pixels de 4 bits e 6 bits utilizando o modo de barramento Byte é ilustrado na figura 4.13.

O diagrama da figura 4.13 é ilustrado em termos de leitura de 12 bits no barramento de dados Data[0-11]. Deve se ter em mente, entretanto, que os pinos de Data[0-11] são conectados a dois registradores de 8bits na porta paralela. Sete bits, Data [0-6], são lidos do Registrador de Dados (Data Register) e os restantes 5 bits, Data[7-11] são lidos do Status Register.

Para pixels de quatro bits, eles são simplesmente colocados em cada 4 bits subsequentes partidos, em um padrão alternado. A ordem de bits de cada pixel é ilustrado pelo número nos cantos. Pixels de 6 bits têm uma compactação similar, exceto que são os 6 bits partidos que alternam. Observa-se que mesmo que o pixel caia em mais fronteiras quando Data[0-11] é visualizada como barramento de 12 bits,

o atual bit será separado em uma maneira menos clara, quando esses limites são mapeados para os registradores da porta paralela (somente um bit de P2, o bit 0, cairá dentro do Data Register, no espaço de Data[0-6], com os outros cinco bits aparecendo no Status Register).

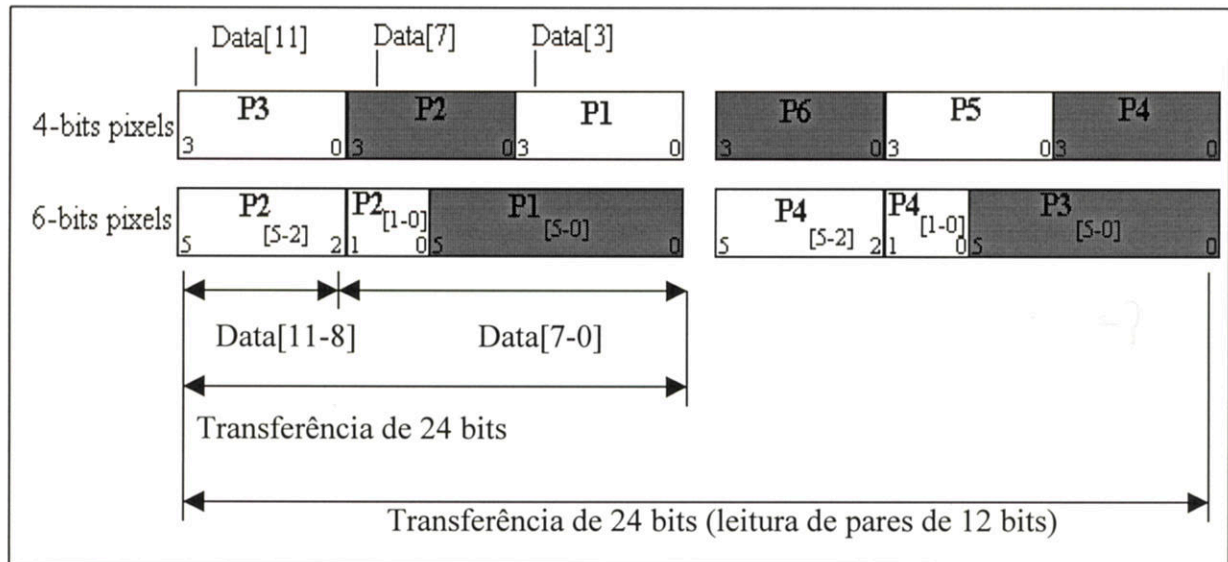


Figura 4.13 - Modo de Compactação para Modo Byte

4.5. Conclusões

O capítulo visto descreveu todo o funcionamento da câmera digital, seus comandos e sequências de configuração para o correto recebimento dos dados de imagem, de acordo com os parâmetros inseridos pelo usuário.

O próximo capítulo apresenta o tratamento dos dados de imagem, já que a câmera só é capaz de transmitir os dados de imagem, não tendo a capacidade de convertê-los em cores, ou tonalidades de cinza. Apresentar-se-á todo o processo do tratamento, desde a aquisição até a apresentação da imagem tratada.

CAPÍTULO 5

Tratamento e Processamento da Imagem - Driver de Vídeo

5.1. Introdução

Neste capítulo serão abordados os métodos que foram desenvolvidos para transformar os dados de imagem adquiridos pela câmera, em imagens com diferentes tonalidades de cinza.

Para entender este capítulo, se o leitor não possui conhecimentos suficientes, é necessário ler sobre o sistema gráfico X Windows e sua biblioteca, mais especificamente o Xlib e o XForms, no capítulo 3.

5.2. Procedimento do Tratamento de Imagem

Para transformar os dados capturados pela QuickCam, que são enviados a um buffer no PC, em uma imagem correspondente, ocorre uma sequência complexa de rotinas e comandos.

A seguir, apresenta-se de forma clara e concisa, as etapas elaboradas para o processamento das imagens do programa.

Antes da captura de qualquer dado de imagem, é necessário estabelecer alguns parâmetros e rotinas que estarão envolvidas no processo de tratamento de imagem do sistema, que serão visualizados a seguir:

5.2.1. Display

Esta é a primeira etapa a ser realizada. O display é o mostrador onde se apresentará as imagens geradas pela QuickCam. No caso, o mostrador será a

interface gráfica criada para manipulação das rotinas e funções da câmera digital (a interface será apresentada e elucidada no capítulo 7).

Como visto no capítulo 3, sobre a biblioteca de auxílio à construção da interface gráfica, o comando que habilita a interface a se apresentar no vídeo é: `fl_initialize` (Seção 3.9 - XForms). Denominando este comando como o mostrador (`display`), habilita-se a interface e a nomeia como sendo o mostrador do programa (`display`), que será utilizado nos comandos de captura e tratamento da imagem, vistos no decorrer deste capítulo.

Estrutura:

```
disp = fl_initialize(&argc,argv,0,0,0);
```

5.2.2. Intensidade (Depth)

O segundo passo a ser definido, é a identificação da Intensidade que a janela criada no sistema X Windows irá trabalhar. Como visto anteriormente, a Intensidade (Depth) indica o número de planos que são usados para representar escalas em cinza em uma janela. Indica também o número de bits por pixel da imagem a ser gerada.

Para obter a intensidade da janela especificada, foi utilizado um comando que reconhece o valor e retorna o mesmo à uma variável designada. Esta variável irá, através de condições especificadas no programa, estabelecer a rotina que convém ser utilizada pelo sistema.

Estrutura:

```
xDepth = fl_state[fl_get_vclass()].depth;
```

5.2.3. Tela (Screen)

Após a aquisição do valor referente à intensidade a ser utilizada, estabelece-se a criação da tela onde será alocado o mostrador (`display`), que apresentará o programa propriamente dito. No caso, cria-se uma variável "tela", que com o auxílio do comando `XDefaultScreen`, estabelece-se uma tela padrão, onde será utilizada mais adiante.

Estrutura:

```
tela = XDefaultScreen(display);
```

onde: disp = display.

5.2.4. Identificação da Janela (Window ID)

Realizados os procedimentos de aquisição de intensidade (depth) e criação da tela (XdefaultScreen), estabelece-se a captura da área de desenho. Através de um comando de captura da identificação da janela (*Window ID*), captura-se a área de desenho onde as imagens serão inseridas.

Estrutura:

```
win = fl_winget();
```

5.2.5. Contexto Gráfico

Tendo em mãos a identificação da janela e o display especificado, realiza-se o procedimento para a criação do contexto gráfico.

GC: é o contexto gráfico do sistema. Deve ser criado antes que a imagem seja capturada e tratada. XCreateGC: cria um novo contexto gráfico para uma dada tela com a intensidade especificada da área de desenho.

```
XCreateGC (display, drawable, valuemask, values);
```

```
Display *display;
```

```
Drawable drawable;
```

```
Unsigned long valuemask;
```

```
XGCValues *values;
```

Argumentos:

- Display: especifica a conexão para um servidor X.
- Drawable: especifica uma área de desenho. O GC criado pode apenas ser usado para desenhar em áreas de desenho da mesma intensidade que essa área de desenho.

- Valuemask: Especifica quais os membros do GC que são setados, utilizando informação na estrutura de valores.
- Values: especifica o ponteiro para uma estrutura de XGCValues que irá prover componentes para o novo GC.

Para o programa desenvolvido, utilizou-se os seguintes parâmetros:

```
gc = XCreateGC (disp, win, 0, &values);
```

- gc: especificará o novo contexto gráfico criado, como argumento em outros comandos subsequentes.
- Disp: display especificado refere-se à interface gráfica criada para a apresentação do programa.
- Win: especifica a área de desenho que se encontra dentro dessa interface gráfica.
- Valuemask: esta variável é dita nula;
- &values: este ponteiro identificará os valores de XGCValues que proverão os componentes para o novo GC.

5.2.6. Mapa de Cores (Colormap)

Após o procedimento de criação do contexto gráfico, estabelece-se a criação do mapa de cores para o sistema de tratamento de imagens. Através da rotina XCreateColormap, cria-se um mapa de cores que suportará os níveis de cinza estabelecido pelas características operacionais da QuickCam.

O comando que cria esse mapa de cores é:

```
XCreateColormap(display, w, visual, alloc)
```

Argumentos:

- Display *display
- Window w;
- Visual *visual;
- Int alloc.

- Display: especifica a conexão com um servidor X, no caso a interface gráfica.
- W: especifica a área de desenho dentro dessa interface gráfica.
- Visual: Especifica um ponteiro para a estrutura de aparência do mapa de cores. A classe de aparência e intensidade devem ser suportadas pela tela.
- Alloc: Especifica quantas entradas de mapa de cores a serem alocadas. Passar ou AllocNone ou AllocAll.

Para o software desenvolvido, estabeleceu-se os seguintes parâmetros:

```
cmap = XCreateColormap(display, win, DefaultVisual(display,tela), AllocNone);
```

- cmap: essa variável será utilizada para identificar, posteriormente, o mapa de cores criado.
- disp: O display especificado refere-se à interface gráfica criada para a apresentação do programa.
- win: especifica a área de desenho que se encontra dentro dessa interface gráfica.
- Visual: DefaultVisual(display,tela): retorna a aparência padrão para o display especificado e a tela especificada.
- AllocNone: como a classe de aparência especificada é *StaticGray*, as células de cores terão valores de somente leitura alocados, então, neste argumento, deve ser especificado *AllocNone*, ou um erro será ocasionado.

Após a criação do mapa de cores, o próximo passo é atribuir à área de desenho esse mapa de cores. Através da estrutura abaixo, realiza-se esse procedimento.

```
XsetWindowColormap(display,w,colormap)
```

Para o software desenvolvido, essa atribuição possui os seguintes argumentos:

```
XsetWindowColormap(display, win, cmap)
```

- Disp: identifica o display, que refere-se à interface gráfica criada.
- Win: especifica a área de desenho dentro dessa interface gráfica.
- Cmap: identifica o mapa de cores criado na função anterior.

5.2.7. Criação da Paleta de Cores – Células de Cores

Como visto anteriormente, um mapa de cores criado pode possuir quantas células de cores o cliente desejar.

Através do comando `XAllocColor()`, é retornado o indexador da célula de cor (o valor do pixel) que contém os valores de RGB que são requisitados ou que contém o valor de RGB próximo ao valor fisicamente possível para apresentação na tela.

Através da rotina criada, estabelece-se a indexação dos valores de pixels referentes à leitura da câmera digital. No caso, através dessa rotina, fica estabelecido uma escala máxima de 64 níveis de cinza que o sistema de imagem suporta, indo do branco ao preto. Assim, cada palavra recebida pelo PC, proveniente da câmera digital, receberá um valor correspondente ao nível de cinza considerado.

5.2.8. Captura da Imagem

Para a aquisição da imagem é necessário primeiramente a criação da estrutura de imagem X, através da função `XCreateImage()`.

`XCreateImage(display, visual, depth, format, offset, data, width, height, bitmap_pad, bytes_per_line)`

Argumentos:

- **Display:** especifica a conexão com o servidor X.
- **Visual:** Especifica um ponteiro para a Aparência que deve ser igual à Aparência da área de desenho onde a imagem será mostrada.
- **Depth:** especifica a intensidade da imagem.
- **Format:** Especifica o formato da imagem. Pode ser `XY_Bitmap`, `XYPixmap`, `Zpixmap`.
- **Offset:** especifica o número de pixels além da inicialização dos dados onde a imagem começa.
- **Data:** especifica um ponteiro para os dados de imagem.
- **Width, height:** especificam a largura e altura da imagem, em pixels.

- `Bitmap_pad`: especifica a quantidade de linhas lidas. Em outras palavras, o início da primeira linha lida é separada da memória do cliente do começo da próxima linha lida por um inteiro múltiplo desses bits. Os valores repassados podem ser: 8, 16 ou 32.
- `Bytes_per_line`: especifica o número de bytes na imagem do cliente entre o começo da primeira linha lida e o início da próxima. Se for passada o valor zero, Xlib assume que a leitura de linha é contínua na memória e, dessa forma, o próprio sistema Xlib calcula o valor de `byte_per_line`.

No programa propriamente dito, a função `XCreateImage` toma a seguinte forma:

```
Ximage = XCreateImage(displ, DefaultVisual(displ,tela), XDepth, ZPixmap, 0,
ximagem, largura/TransferMode, altura/TransferMode, 8, 0);
```

Argumentos:

- `Ximage`: A imagem criada será representada pela variável `Ximage`;
- `Displ`: é o display especificado pelo usuário;
- `DefaultVisual(displ,tela)`: retorna uma Aparência em comum para o display e a tela;
- `XDepth`: especifica a intensidade do sistema. No caso, essa variável foi adquirida por um comando anterior, visto na seção de aquisição da intensidade;
- `ZPixmap`: as imagens possuem este formato;
- `0`: indica que o offset é nulo;
- `ximagem`: é o ponteiro para os dados da imagem;
- `largura/TransferMode` e `altura/TransferMode`: identificam as dimensões da tela a serem adquiridas. Nota-se que são divididas pela variável `TransferMode`, que no caso de se habilitar o zoom parcial, esta variável altera o seu valor de 1 para 2, o que promove a divisão da área projetada por dois.
- `8`: retorna o valor do `bitmap_pad` escolhido;
- `0`: retorna o valor escolhido de `bytes_per_line`. Desta forma, o sistema admite a leitura contínua da linha, e o sistema calcula essa variável automaticamente.

5.2.9. Manipulação da Imagem

Após ter criado a estrutura de Imagem X, através do `XCreateImage()`, inicia-se a captura da imagem.

Enviando para a câmera o comando `SendVideoFrame`, visto no capítulo anterior, a câmera inicia o envio da primeira imagem. Esta imagem é recebida pelo PC e armazenada em um buffer de memória específico (`videobuffer`).

Após a captura da imagem e armazenagem no buffer, inicia-se o processo de indexação dos valores das palavras lidas para a transformação desses em valores correspondentes a níveis de cinza.

Através de uma rotina elaborada, os dados do buffer de imagem, no caso o `videobuffer`, e o ponteiro para os dados de imagem são manipulados de forma a atribuir aos dados do `videobuffer` os valores correspondentes aos 64 níveis de cinza especificados.

Para isso, utiliza-se a tabela de cores criada pelo software, multiplicando seus valores com os respectivos valores do `videobuffer`, e retornando um ponteiro para o ponteiro que especifica a imagem, no caso, `ximagem`.

5.2.10. Visualização da Imagem Processada

Após essa manipulação, a imagem já está pronta para ser visualizada. A função que coloca a imagem na área de desenho especificada é a `XPutImage`.

`XPutImage(display, d, gc, image, src_x, src_y, dest_x, dest_y, width, height)`

- `Display`: especifica a conexão para um servidor X.
- `D`: especifica a área de desenho.
- `Gc`: especifica o contexto gráfico.
- `Image`: especifica a imagem que se deseja combinar na área de desenho.
- `Src_x` e `Src_y`: especifica as coordenadas do canto superior esquerdo do retângulo a ser copiado.
- `Dest_x` e `dest_y`: Especifica as coordenadas X e Y, relativas à origem da área do desenho, onde o canto superior esquerdo do retângulo copiado será inserido.

- Width, Height: especifica a largura e altura, em pixels, da área retangular a ser copiada.

Argumentos utilizados para essa estrutura:

```
XPutImage (disp, win, gc, ximage, 0, 0,
           Interface->ImageBox->x+4+((320-(PixelPerLine/TransferMode))/2),
           Interface->ImageBox->y+4+((240-(NumLines/TransferMode))/2),
           ximage->width, ximage->height);
```

- disp: especifica o display, que no caso é a interface gráfica desenvolvida para o sistema;
- win: especifica a janela criada na interface gráfica para alocação das imagens geradas;
- ximage: é a variável que identifica a estrutura de criação de imagem -> XCreateImage();
- 0,0: indica as coordenadas do canto esquerdo superior da janela, onde a imagem será inserida;
- Interface->ImageBox->x+4+((320-(PixelPerLine/TransferMode))/2), Interface->ImageBox->y+4+((240-(NumLines/TransferMode))/2): indica as coordenadas da janela, na área do display, onde a imagem será inserida. Observa-se a existência de um cálculo com as variáveis PixelPerLine e TransferMode. Esse cálculo resulta na inserção da imagem sempre no centro da janela, não importando o seu tamanho, ou zoom (total ou parcial);
- ximage->width, ximage->height: especifica a largura e altura, em pixels, da área retangular a ser copiada, no caso especificada pela variável ximage.

5.3. Conclusões

O capítulo apresentou o processo de tratamento da imagem, indicando os procedimentos necessários para a correta configuração dos valores dos dados de

imagem, captados pela câmera, e enviados ao computador. É um processo um pouco lento, o que implica numa redução da amostragem das imagens tratadas.

Para resolver este problema, o próximo capítulo descreve um processo de aceleração do tratamento de imagens. Este processo consiste na criação de um segmento de memória compartilhada, que irá otimizar o sistema desenvolvido.

CAPÍTULO 6

Aceleração do Processamento da Imagem

6.1. Introdução

Neste capítulo, será abordado uma estrutura de programa, que otimizará o sistema de aquisição e tratamento. Para o entendimento do princípio do sistema, é necessário que o leitor esteja a par da estrutura do sistema X Windows, referente ao capítulo 3.

6.2. Segmento de Memória Compartilhada

A capacidade básica oferecida é para o compartilhamento de memória Ximage. É essencialmente uma versão da interface Ximage onde a imagem atual é armazenada em um segmento de memória compartilhada, e então não necessita ser movida através do canal de comunicação entre processos, o Xlib. Para imagens de tamanho maior, o uso dessa facilidade pode resultar em aumentos consideráveis de desempenho.

Adicionalmente, algumas implementações oferecem memória compartilhada em pixmaps. Pixmaps são arrays bidimensionais de pixels em um formato especificado pelo servidor X, onde os dados da imagem são armazenados em um segmento de memória compartilhado. Através do uso de pixmaps de memória compartilhada, é possível alterar os dados desses pixmaps sem utilizar as rotinas de Xlib.

6.3. Processo de Utilização da Memória Compartilhada

O código que utiliza a extensão de memória compartilhada deve incluir um número de arquivos *headers*.

```
# include <X11/Xlib.h>
# include <sys/ipc.h>
# include <sys/shm.h>
# include <X11/extensions/XShm.h>
```

Se o sistema utilizado não suporta memória compartilhada, o arquivo XShm.h pode não estar presente. Dessa forma, será realizado uma série de usos de #ifdefs.

Qualquer código que utiliza a extensão de memória compartilhada tem que, primeiramente, checar se o servidor suporta a extensão. Poderá ser executado sobre a rede, ou em algum ambiente onde a extensão não funcionará. Para realizar essa checagem, a rotina abaixo é suficiente:

```
Status XShmQueryExtension (display)
```

```
Display *display
```

ou

```
Status XShmQueryVersion (display, major, minor, pixmaps)
```

```
Display *display;
```

```
int *major, *minor;
```

```
Bool *pixmaps
```

Onde “display” é a tela onde está sendo executado a rotina. Se a extensão de memória compartilhada for usada, o valor retornado para quaisquer das duas funções será verdadeiro (*True*). De outra maneira, o programa operará utilizando as rotinas padrão de Xlib. Quando a Memória Compartilhada está disponível, XshmQueryVersion também retorna “*major*” e “*minor*”, que são os números das versões da implementação da extensão, e “*pixmaps*” que retorna verdadeiro se a memória compartilhada de pixmaps é suportada.

A sequência básica de operações para memória compartilhada para XImage é a seguinte:

- 1- Criar uma estrutura de memória compartilhada para Imagens X.
- 2- Criar um segmento de memória compartilhada para armazenar os dados da imagem.
- 3- Informar o servidor sobre o segmento de memória compartilhado
- 4- Usar a memória compartilhada de Imagens X.

6.3.1. Criação da Estrutura de Memória Compartilhada

Para criar uma Memória Compartilhada de Imagens X, utiliza-se:

```
XImage *XShmCreateImage (display, visual, depth, format, data, shminfo, width, height)
```

```
Display *display;  
Visual *visual;  
unsigned int depth, width, height;  
int format;  
char *data;  
XShmSegmentInfo *shminfo;
```

A maioria dos argumentos são os mesmos da rotina `XCreateImage`. É importante observar que não existem os argumentos "offset", "bitmap_pad" ou "bytes_per_line". Esses argumentos serão definidos pelo próprio servidor, e os códigos a serem implementados serão suportados por eles. A não ser que já tenha sido alocado o segmento de memória compartilhado, deve-se, então, passar em nulo (`NULL`) para o ponteiro "data".

Existe um argumento adicional: "shminfo", que é o ponteiro para a estrutura do tipo `XShmSegmentInfo`. Deve-se alocar uma dessas estruturas, de tal forma que irá possuir um tempo de vida tão longo quanto durar o tempo da memória compartilhada para Imagens X. Não existe a necessidade de iniciar essa estrutura antes de chamar o `XShmCreateImage`.

O valor retornado, será uma estrutura `XImage`, que poderá ser utilizada nos passos subsequentes.

6.3.2. Criação do Segmento de Memória Compartilhada

O próximo passo é a criação do segmento de memória compartilhada. Recomenda-se ser criado depois da criação de `XImage`, pois essa informação é

utilizada em Ximage para saber a quantidade de memória a ser alocada. Para criar o segmento, chama-se a rotina:

```
shminfo.shmid =  
shmget (IPC_PRIVATE, image -> bytes_per_line * image->height, IPC_CREAT|0777);
```

(assumindo que já tenha sido requisitado a memória compartilhada de Ximage “image”). Deve-se seguir as regras e realizar a checagem de erros em todas essas chamadas de sistema. Também deve-se assegurar o uso do campo de bytes_per_line, e não a largura da imagem que será utilizada para criar a Ximage, pois essas poderão ser diferentes.

Observa-se que a identificação da memória compartilhada (*Shared Memory ID*) retornada pelo sistema é armazenada na estrutura shminfo. O servidor necessitará essa identificação para se fixar ao segmento.

O passo a seguir é a de fixação do segmento de memória compartilhada criado ao processo propriamente dito:

```
shminfo.shmaddr = image->data = shmat (shminfo.shmid, 0, 0);
```

O endereço retornado pelo shmat é armazenado em ambas as estruturas de Ximage e shminfo.

Para terminar o complemento da estrutura shminfo, deve-se decidir como o servidor será fixado ao segmento de memória compartilhada, e setar o campo “*ReadOnly*” como segue. Normalmente, utiliza-se:

```
shminfo.readOnly = False;
```

Se for setada em verdadeiro (*True*), o servidor não será capaz de escrever nesse segmento, e então a rotina de chamada XshmGetImage irá falhar.

6.3.3 Informar o Servidor Sobre o Segmento de Memória

O passo seguinte é dizer ao servidor para se fixar ao segmento de memória compartilhada, através da chamada:

```
Status XShmAttach (display, shminfo);
```

Se a operação for de sucesso, o sistema retornará um estado de não-zero, e a `XImage` estará pronta para ser utilizada.

Para escrever a `XImage` de memória compartilhada em uma área de desenho (`X drawable`), utiliza-se `XshmPutImage`:

```
Status XShmPutImage (display, d, gc, image, src_x, src_y, dest_x, dest_y, width, height, send_event)
```

```
    Display *display;
    Drawable d;
    GC gc;
    XImage *image;
    int src_x, src_y, dest_x, dest_y;
    unsigned int width, height;
    bool send_event;
```

A interface é semelhante ao comando `XPutImage`, com a existência de um parâmetro adicional, denominado "*send event*". Se este parâmetro é passado como verdadeiro (`True`), o servidor irá gerar um evento complemento onde a imagem escrita é completada. Então o programa pode determinar onde é seguro começar de novo a manipulação do segmento de memória compartilhada.

O evento complemento é do tipo `XshmCompletionEvent`, que é definido como:

```
typedef struct {
    int type;                /* do evento */
    unsigned long serial;    /* # da última requisição processada */
    Bool send_event;        /* Verdadeiro se vem da requisição de SendEvent */
    Display *display;       /* Display - exibe de onde o evento foi lido */
    Drawable drawable;      /* requisição da área de desenho */
    int major_code;         /* Código ShmReq */
    int minor_code;         /* X_ShmPutImage */
    ShmSeg shmseg;         /* O ShmSeg usado na requisição */
    unsigned long offset;   /* o offset usado no ShmSeg */
} XShmCompletionEvent;
```

O tipo de valor do evento que será utilizado pode ser determinado em tempo real com uma linha na forma:


```
int CompletionType = XShmGetEventBase (display) + ShmCompletion;
```

Se o segmento de memória compartilhada for alterado antes da chegada do evento complemento, os resultados serão inconsistentes.

6.3.4 Utilização da Memória Compartilhada de Imagens X

Para ler os dados de imagem na XImage de memória compartilhada, utiliza-se a seguinte rotina:

```
Status XShmGetImage (display, d, image, x, y, plane_mask)
Display *display;
Drawable d;
XImage *image;
Int x, y;
unsigned long plane_mask;
```

Onde display é a tela de interesse, d é a área de desenho, image é o destino de Ximage, x e y são os offsets em d, e plane_mask define quais os planos a serem lidos.

Para destruir a memória compartilhada de XImage, deve-se, primeiramente, instruir o servidor a se liberar do segmento de memória compartilhada, somente então, destruir o segmento. A rotina é dada abaixo:

```
XShmDetach (display, shminfo);
XDestroyImage (image);
shmdt (shminfo.shmaddr);
shmctl (shminfo.shmid, IPC_RMID, 0);
```

6.4. Conclusões

O sistema de aceleração de imagens se mostrou bastante eficaz no processo de otimização do tratamento das imagens. A velocidade de apresentação das telas teve um incremento na média de trinta por cento, em relação ao sistema original, sem a utilização dessa ferramenta.

Tendo descrito as plataformas de desenvolvimento do software, o princípio de funcionamento da câmera digital, os seus protocolos de comunicação e configuração, bem como as rotinas de tratamento de imagens e aceleração dessas, o próximo capítulo trata, exclusivamente, da apresentação do software, em conjunto com a descrição de suas funções, através da sua interface gráfica desenvolvida.

CAPÍTULO 7

Interface Gráfica e Funções Desenvolvidas

7.1. Introdução

Este capítulo, por final, apresenta o software desenvolvido, através do auxílio da fundamentação teórica descrita nos capítulos iniciais. Será apresentada a interface gráfica criada para a manipulação dos controles e funções da câmera digital, bem como as funções extras desenvolvidas, voltadas para a segurança. Cada função extra desenvolvida será seguida de uma descrição do princípio de funcionamento utilizado para atingir os objetivos almejados.

7.2. Procedimento de Inicialização do Software

Para inicializar o software, a primeira etapa a ser realizada, consiste na rotina de procura automática da câmera, nos três possíveis endereços onde a paralela pode estar conectada (geralmente é no endereço 0x378, mas a porta pode estar conectada pelos endereços 0x278 ou 0x3bc).

Detectando a presença da câmera, o endereço é setado como sendo da QuickCam. Caso a varredura dos três possíveis endereços não a encontre, o programa entra com uma nova opção, perguntando ao usuário se deseja desistir da procura e fecha o programa, ou inserir manualmente o endereço onde a QuickCam se encontra.

O programa se inicia com a identificação do tipo de porta que o computador possui. Esta verificação se faz necessária, pois caso o computador não possua uma porta paralela com capacidade bidirecional, a função de transferência bidirecional de dados de vídeo deverá ser desativada na interface gráfica.

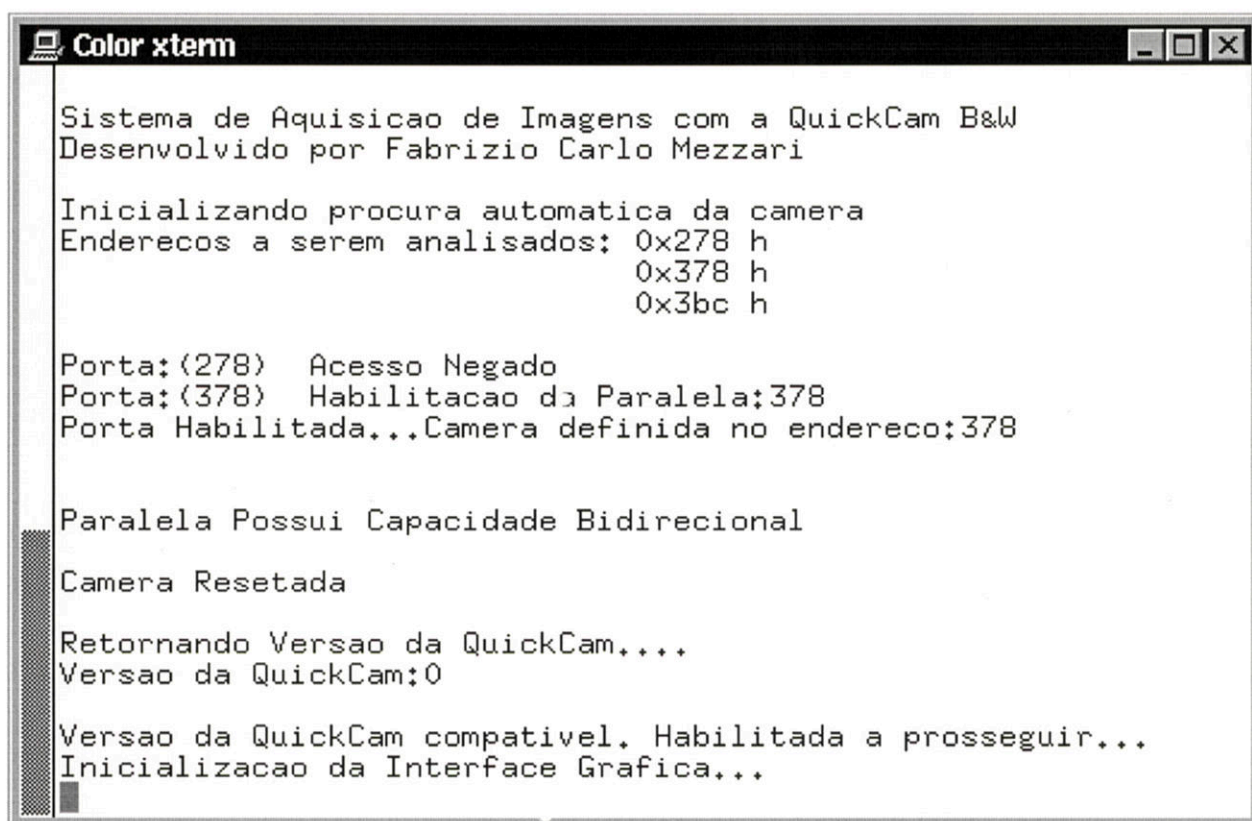
Em seguida, envia-se o comando que reseta a câmera, e logo após, o de reconhecimento da mesma. Este comando é referente à captura da versão da

QuickCam. Se a versão é compatível, o sistema prossegue, caso contrário, o sistema é desativado e o programa fechado.

A partir desse ponto, as configurações de inicialização da câmera são realizadas, conforme escrito no capítulo 4.

Todos esses procedimentos são visualizados pelo usuário, na tela do computador, como mostra a figura 7.1

Após a câmera ser configurada e estar pronta para operação, ativa-se a interface gráfica do sistema. Esta interface possibilitará ao usuário o controle total do sistema, desde as configurações da câmera digital até as funcionalidades extras inseridas. A interface gráfica é visualizada na figura 7.2.



```
Color xterm
Sistema de Aquisicao de Imagens com a QuickCam B&W
Desenvolvido por Fabrizio Carlo Mezzari

Inicializando procura automatica da camera
Enderecos a serem analisados: 0x278 h
                               0x378 h
                               0x3bc h

Porta:(278) Acesso Negado
Porta:(378) Habilidade da Paralela:378
Porta Habilitada...Camera definida no endereco:378

Paralela Possui Capacidade Bidirecional

Camera Resetada

Retornando Versao da QuickCam....
Versao da QuickCam:0

Versao da QuickCam compativel. Habilidade a prosseguir...
Inicializacao da Interface Grafica...
```

Figura 7.1 – Tela de Abertura do Software



Figura 7.2 – Interface Gráfica do Sistema

7.3. Funções controladas pela Interface

7.3.1. Dimensão da Tela

Através desta função, pode-se controlar as possíveis dimensões da imagem que aparecerá na interface. As dimensões pré-estabelecidas são: 320x240, 240x180, 200x140, 160x120, 120x100.

Todos os tamanhos da imagem são dimensionados em pixels, onde estes valores serão enviados para o sistema de configuração da câmera. Após a nova escolha, o sistema de configurações é refeito, e uma nova característica de imagem é gerada, a partir de então.

A interface para a escolha das dimensões é visualizada na figura 7.3.

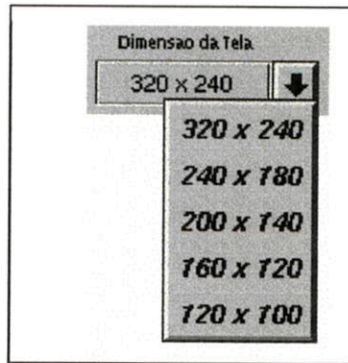


Figura 7.3 - Escolha da Dimensão da Tela

7.3.2. Qualidade da Imagem

Como explicado no capítulo 4, o sistema de aquisição de imagens estabelece duas resoluções para as imagens geradas. A primeira, de menor resolução, é composta no processo de tratamento de imagem com 16 diferentes níveis de cinza. É um processo mais rápido, mas que, em contra partida, fornece uma menor resolução da imagem mostrada na interface.

Essa resolução de menor categoria é denominada de "Normal (4bpp)". O termo 4bpp significa 4 bits por pixel, ou seja, cada pixel corresponderá a uma palavra de 4bits, oferecendo assim, 16 possibilidades de níveis de cinza ($2^4 = 16$).

Para a resolução de maior qualidade, denominada "Superior (6bpp)", existe uma palavra de 6 bits que representará cada pixel da imagem (6 bits por pixel). Com esse tamanho de palavra, é possível representar a imagem com até 64 níveis de cinza ($2^6 = 64$). Entretanto, com uma maior qualidade adquirida, a velocidade de apresentação das imagens é comprometida sensivelmente, em relação à velocidade de apresentação das imagens para a qualidade normal.

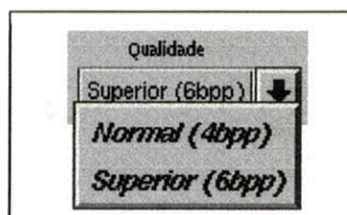


Figura 7.4 – Escolha da Resolução da Imagem

7.3.3. Zoom

Esta função possibilita o usuário controlar o zoom da imagem gerada. Existem duas possibilidades oferecidas pelo sistema.

A primeira é o Zoom Total. É a imagem gerada com 100% de sua capacidade. A segunda possibilidade corresponde ao Zoom Parcial. A imagem gerada possui 50% de sua capacidade total. Neste caso, através de um comando para alterar a configuração do envio de imagens da câmera para o PC, a própria câmera realiza o processo de alteração. A cada duas palavras que representarão dois pixels na imagem a ser processada, a câmera descarta uma delas, enviando para o computador apenas uma das duas palavras, que representarão uma imagem com uma dimensão correspondendo à metade da dimensão total.

Através desse Zoom, a velocidade da amostragem das imagens é acrescida significativamente. A figura 7.5 demonstra como é realizado o processo de escolha do zoom na interface gráfica.

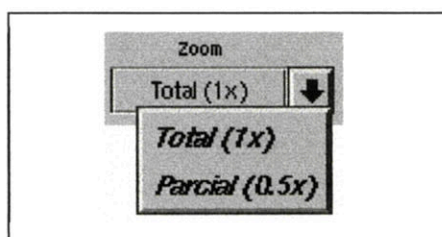


Figura 7.5 – Escolha do Zoom

7.3.4. Modo de Transmissão

Através dos dois botões de comando da interface, Unidirecional e Bidirecional (figura 7.6), o sistema irá alterar o modo de recepção dos dados de imagem recebidas pelo computador, conforme já explicado nos capítulos 4.



Figura 7.6 – Escolha do Modo de Transmissão

Caso a paralela não possua capacidade bidirecional, o qual é detectado na inicialização do software, o botão que ativa esta função é desabilitado permanentemente da interface gráfica, ficando apenas disponível a opção de modo de transmissão unidirecional (figura 7.7).

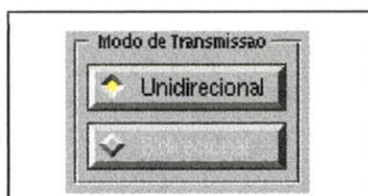


Figura 7.7 – Desativação do Modo Bidirecional

7.3.5. Motor Remoto

Esta função possibilita a movimentação da câmera, na direção horizontal, em ambos os sentidos (direita e esquerda). Esse sistema de movimentação é composto, na interface, por dois botões, correspondendo aos sentidos de direção esquerda e direita, e um cursor (em amarelo), como mostra a figura 7.8, que se desloca no sentido em que a câmera é movimentada, dando ao usuário uma referência da posição de onde a câmera se encontra apontada.

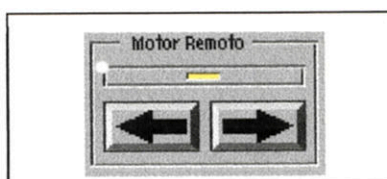


Figura 7.8 – Controle do Motor Remoto

7.3.5.1. O Motor de Passo

Para a realização da movimentação da câmera, foi utilizado um motor de passo.

O motor de passo consiste em um estator com ranhuras, equipado com duas bobinas individuais, e uma estrutura de rotor que não possui enrolamento. O motor de

passo utilizado neste projeto é denominado de motor de passos de ímã permanente, pois é construído com um ímã permanente preso ao seu eixo.

7.3.5.2. Método de Operação

Para o funcionamento sequencial de um motor de passos, as duas bobinas constituintes devem ser ativadas, desativadas e invertidas em uma sequência específica.

As duas bobinas são identificadas como sendo A-A' e B-B'.

Para realizar uma rotação no sentido anti-horário, é necessário obedecer a seguinte sequência de ativação das duas bobinas:

- 1) Energização da bobina A-A'. Neste procedimento, o motor move-se em um passo de sua posição original.
- 2) Bloqueio da bobina A-A' e energização da segunda bobina no sentido B-B'.
- 3) Bloqueio da bobina B-B' e energização da primeira bobina no sentido A'-A.
- 4) Bloqueio da bobina A'-A e energização da segunda bobina no sentido B'-B.
- 5) A partir deste ponto, repete-se todo o procedimento.

Para o sentido horário, inverte-se a sequência de ativação e desativação das duas bobinas.

7.3.5.3. O Controlador do Motor

Como já mencionado, a câmera trabalha através da porta paralela do microcomputador, utilizando os três registradores de leitura e escrita.

O ideal, para o funcionamento do motor, é que este seja controlado também pela porta paralela, mais precisamente por um dos três registradores. Esta necessidade se dá pelo fato de utilizar a mesma saída de controle do PC, evitando uma nova programação de uma outra porta, no caso a serial, que implicaria em mais códigos e aumentaria os custos para a implementação do projeto.

Para utilizar um dos três registradores para o controle do motor, existe a necessidade de visualização das entradas e saídas que a câmera utiliza (figura 7.9).

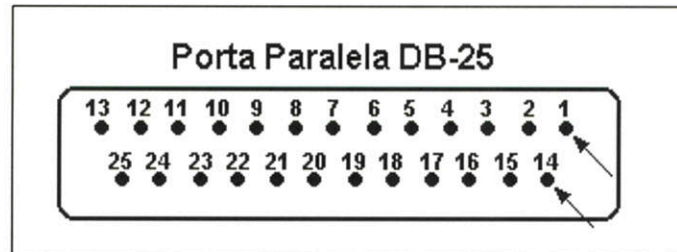


Figura 7.9 – Porta Paralela

Com o auxílio das seções 4.4.1.3, 4.4.1.4 e 4.4.1.5, do capítulo 4, podemos identificar quais os pinos da paralela que a câmera utiliza para receber e enviar dados. Na tabela 4.4, do referente capítulo, apresenta-se uma possibilidade para utilização de dois bits. A câmera digital não utiliza todos os bits fornecidos pelo registrador de controle, sendo os bits 0 e 1 desse registrador não utilizados (todos os bits dos outros dois registradores são utilizados pela câmera). Esses dois bits referem-se aos pinos 1 e 14 da porta paralela, como mostrado na figura 7.9.

Neste caso, esses dois pinos serão os controladores do sentido de rotação do motor de passo, possibilitando a sua movimentação para a direita ou esquerda.

Entretanto, deve se desenvolver uma lógica que permita o controle sequencial das bobinas do motor através desses dois bits livres da porta paralela.

Na tabela 7.1, logo abaixo, os possíveis estados de saída desses dois pinos são mostrados para controlar a movimentação do motor.

Uma observação a ser feita, consiste no fato de que os dois pinos são invertidos, ou seja, quando se coloca nível lógico alto na sua saída, o sinal no pino apresentar-se-á em nível lógico baixo, e vice-versa.

A figura 7.10 apresenta o diagrama de blocos a seguir, apresenta o aspecto de construção do driver do motor de passo.

Pino 1	Pino 14	Estado do Motor
0	0	Motor parado
0	1	Rotação à direita
1	0	Rotação à esquerda
1	1	Sem efeito

Tabela 7.1 – Estados do Motor com Relação aos Pinos 1 e 14

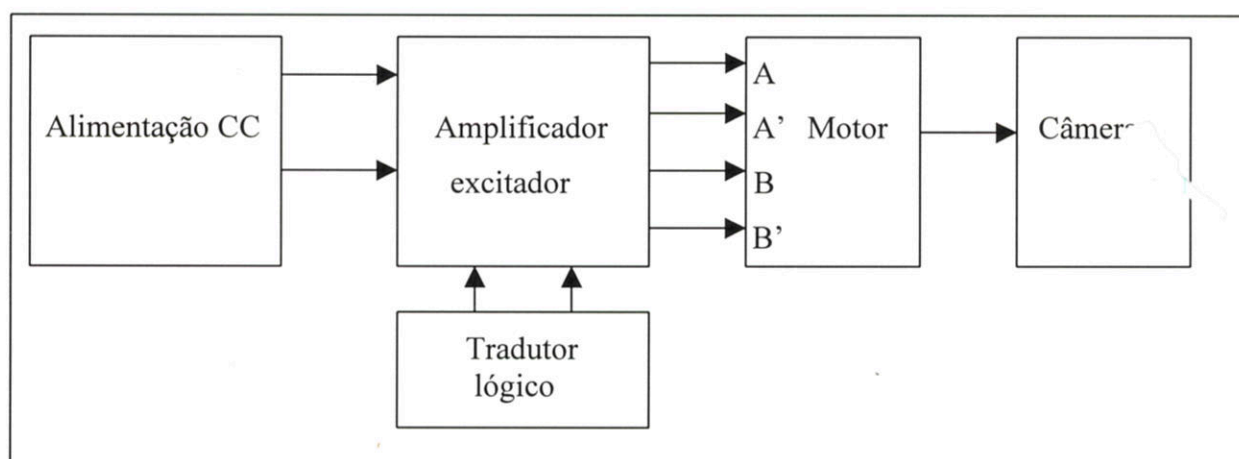


Figura 7.10 - Diagrama de Blocos do Controlador do Motor de Passos

Para a realização da movimentação do motor e a troca de sentido de rotação, foi desenvolvido um controle lógico para essa finalidade, mostrado na figura 7.11.

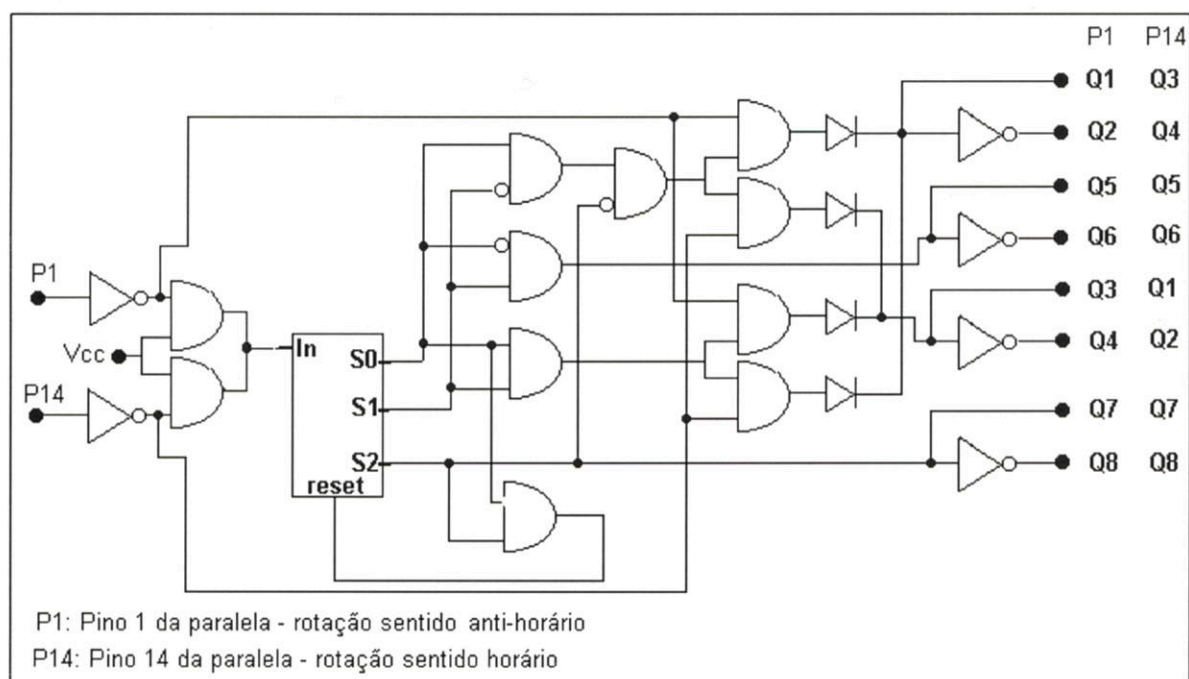


Figura 7.11 - Diagrama do Circuito Lógico Controlador do Motor de Passo

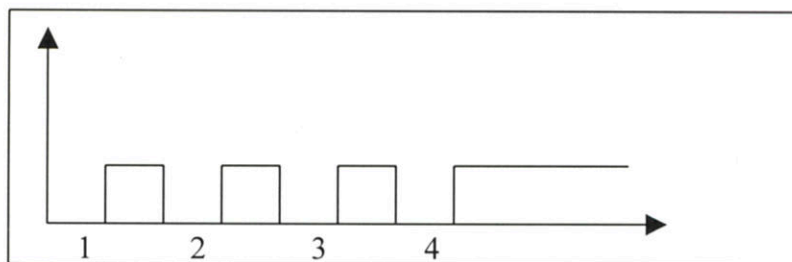


Figura 7.12 – Trem de Pulsos Proveniente da Paralela para Controle do Motor

Para realização dessa operação lógica, a seguinte estrutura de programação teve que ser implementada:

Estrutura de chamada da função do motor:

```
Void Motor (FL_OBJECT * obj, long val);
{
    .....
    /*Estrutura de Envio dos pulsos para rotação no sentido horário: */
    status1= inb(Qc_porta+2);
    status2 = 0x01 | status1; /*sentido anti-horário:status2=0x02|status1;*/
    for (contador = 0; contador < 5;contador++)
    {
        outb(Status2, Qc_porta+2);
        usleep(3000);
        outb(Status1, Qc_porta+2);
        usleep(3000);
    }
    .....
}
```

Como visualizado no circuito lógico, para efeito de proteção e tratamento dos sinais da paralela (referentes ao pino 1 e pino 14), os dois bits dos respectivos pinos são invertidos, pelo fato de serem pinos invertidos da paralela, ou seja, a setagem em nível lógico alto no software, para esses pinos, corresponderá a uma saída física em nível lógico baixo. Dessa forma, o trem de pulsos a ser inserido no circuito será (figura 7.13):

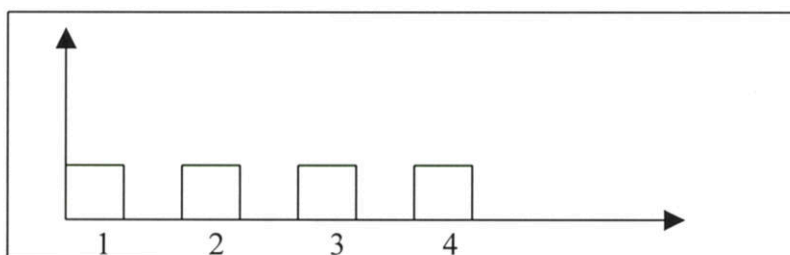


Figura 7.13 – Trem de Pulsos Invertidos para Inserção no Circuito Lógico

Após passar pela porta inversora, o sinal será introduzido em um contador binário, que receberá os pulsos da porta e iniciará a contagem, em código binário. Através da lógica implementada, obtém-se a transformação do código binário em uma sequência que ativará o sistema do motor. As tabelas 7.2 e 7.3, apresentam os estados de saída do contador binário e os respectivos estados de saída do circuito lógico.

Sentido Anti-horário:

Estados de Saída do Contador			Estados de Saída do Circuito			
S2	S1	S0	Q1Q2	Q3Q4	Q5Q6	Q7
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	1	0	1	0	0
1	0	0	0	0	0	1

Tabela 7.2 – Estados de Saída do Contador e do Circuito: sentido anti-horário

Sentido horário:

Estados de Saída do Contador			Estados de Saída do Circuito			
S2	S1	S0	Q1Q2	Q3Q4	Q5Q6	Q7Q8
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	1	0	0	0
1	0	0	0	0	0	1

Tabela 7.3 – Estados de Saída do Contador e do Circuito: sentido horário

As saídas do circuito lógico irão promover a correta excitação do circuito amplificador, que irá controlar a rotação do motor. Conforme figura abaixo (figura 7.14), o circuito é implementado da seguinte forma:

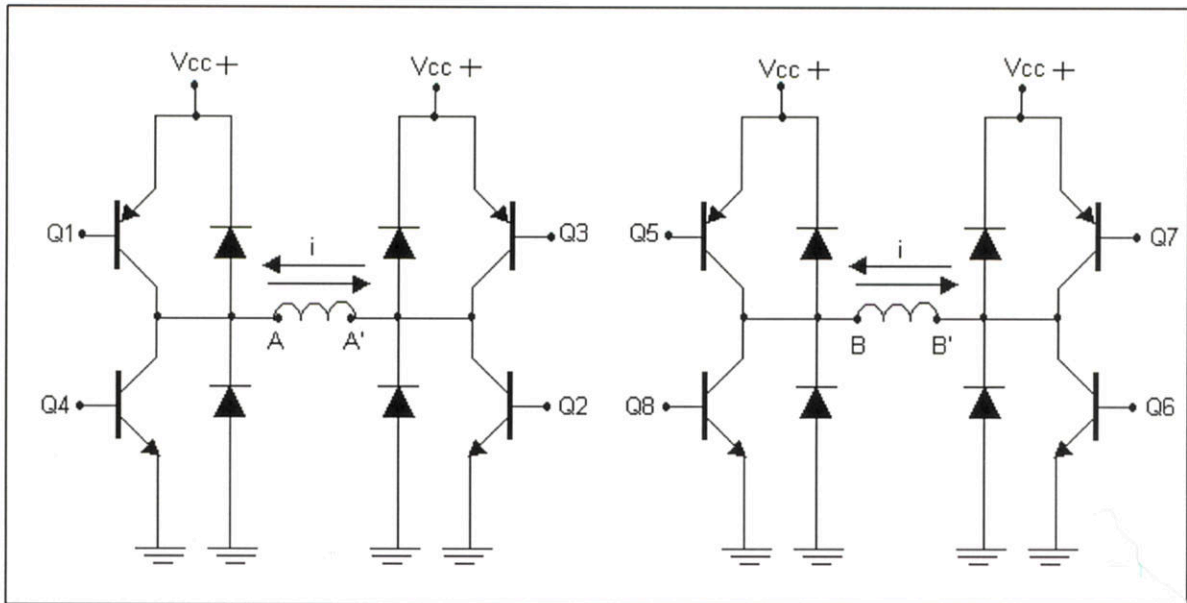


Figura 7.14 – Circuito Amplificador Excitador do Motor de Passo

7.3.6. Ajuste de Imagem

O botão ajuste de imagem, quando ativado, abre uma interface anexa, contendo os possíveis ajustes que podem ser realizados em referência à imagem. Estes ajustes correspondem às seguintes funções:

Brilho: Determina o tempo de exposição em que a câmera estará adquirindo uma leitura dos dados do ambiente. Os valores estabelecidos estão entre 0 e 255. O usuário possui a capacidade de controlar o brilho da imagem de modo a adaptá-la às novas condições do ambiente (mais escuro ou mais claro).

Contraste: Determina o contraste entre os tons de cinza estabelecidos pelo sistema. Pode-se escolher valores entre 0 e 255.

Balanço: Determina o offset desejado pelo usuário, melhorando a apresentação das imagens.

Esses três reguladores da característica da imagem são preestabelecidos e setados quando o sistema entra em operação.

Reseta Valores: após o usuário ter alterado os valores de configuração da imagem, com o objetivo de melhorar a visualização da mesma, dependendo da luminosidade, o mesmo pode retornar aos valores padrões de brilho, contraste e balanço, acionando este comando.

Ok: Para desabilitar essa nova interface, basta acionar o botão de Ok que se encontra na interface. Acionando-o, a interface desaparece e fica-se apenas com a interface principal em funcionamento.

As funções descritas acima, são visualizadas na interface gráfica, como mostra a figura 7.15.

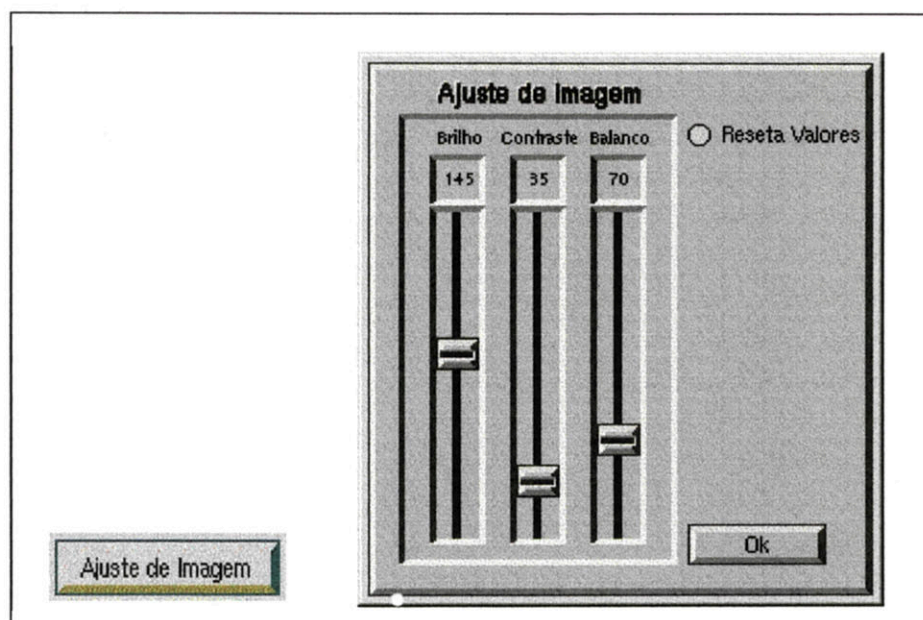


Figura 7.15 – Caixa de Ajuste de Imagem e Botão de Ativação

7.3.7. Auto Brilho

Ativando esta função, o sistema controla automaticamente o nível de exposição da câmera à luminosidade, mais precisamente, o brilho.

Esta função possibilitará o ajuste automático do brilho, conforme a luminosidade no ambiente onde se encontra a câmera for se alterando. Dessa forma, se a intensidade luminosa do ambiente decair, tornando o ambiente mais escuro, o

sistema repassa essa diferença aumentando o brilho da câmera, fazendo com que a imagem, que é mostrada na tela, continue nítida para os olhos do observador.

Da mesma forma, se por algum motivo a intensidade luminosa do ambiente aumenta, o controle automático do brilho diminui o valor do mesmo, fazendo com que a imagem reduza seu brilho, adequando-se às novas condições.

Princípio de Funcionamento

Sabe-se que os dados de imagem tratados e processados pelo software, formam os pixels visualizados na imagem que se apresenta na interface gráfica. Estes pixels possuem, como já explicado no capítulo 3 e 5, diferentes níveis de cinza, estabelecendo assim uma mapa de cores, mais precisamente um mapa que identificará um nível de cinza de acordo com o valor correspondente do pixel. Desta forma, cada pixel possuirá um nível de cinza que varia de 0 até 63.

Fazendo uma aquisição dos dados da imagem, adquire-se o valor que cada pixel possuirá (entre 0 e 63). Ao final da aquisição dos valores, estes são somados e se extrai uma média, que identifica o brilho médio do sistema de imagens.

Caso o ambiente torne-se mais escuro, os pixels que serão monitorados na próxima leitura, possuirão um brilho médio abaixo do estabelecido como padrão, e desta forma, ativa-se o comando de controle de brilho da câmera, aumentando seu valor, gradualmente, até que o novo brilho médio se equipare ao brilho médio padrão.

Da mesma forma estabelece-se para o caso de aumento da luminosidade do ambiente. O brilho médio será maior que o padrão, e o sistema corrigirá diminuindo o brilho da câmera.

Uma observação a ser feita é que o brilho médio padrão, é estabelecido como sendo uma faixa de valores, para que o sistema de brilho não fique em oscilação, corrigindo permanentemente o brilho da imagem ao redor de um único valor.

Para ativar o brilho automático, basta acionar , na interface gráfica, o botão indicando “Ativar AutoBrilho”. Acionando-o, a função é ativada e o botão apresenta-se com um novo rótulo, indicando “Desativar AutoBrilho”. Ao acionar novamente o botão, o sistema de controle automático do brilho é desativado, e o rótulo do botão retorna ao inicial (“Ativar AutoBrilho”).

Além deste indicativo de acionamento da função, a caixa de ajuste de imagem, que é ativada com o botão “Ajuste de Imagem”, apresentar-se-á com uma nova coloração no regulador de brilho (de preto para amarelo), indicando que o sistema de AutoBrilho está controlando o regulador. Da mesma forma, quando a função for desativada, a coloração do regulador retornará ao padrão.

A realização de testes e validação é simplesmente feita através da alteração da iluminação no recinto (abertura e fechamento da janela, ligar e desligar luzes). As alterações da intensidade luminosa do ambiente foram perfeitamente compensadas pelo sistema de AutoBrilho.

A figura 7.16 apresenta a função na interface gráfica, em modo ativado.

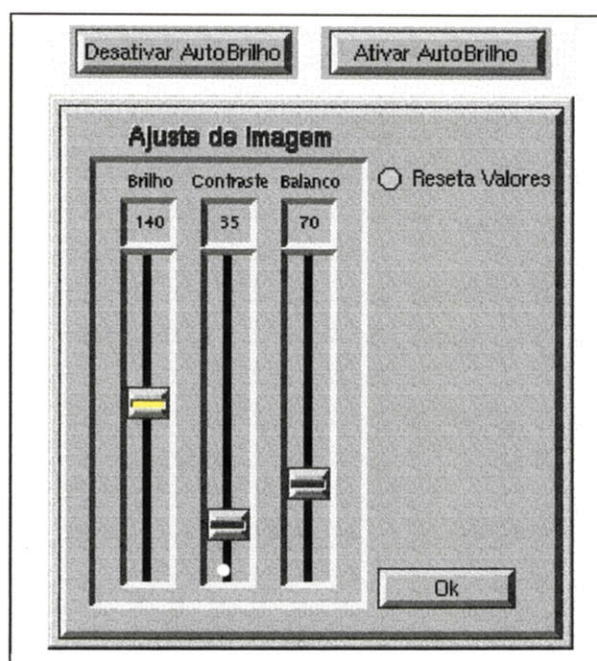


Figura 7.16 – Ativação do AutoBrilho

7.3.8. Detecção

Esta função, quando ativada (figura 7.17), possibilita a realização de uma vigilância do local onde se encontra instalada a câmera. Qualquer alteração do ambiente, tendo como consequência a movimentação de algum objeto ou pessoa no local, a câmera detecta a movimentação e aciona um alarme, disparado em forma de

Bip na interface, com o acendimento de uma luz indicativa, junto com um texto especificando a detecção de movimentos no local monitorado (figura 7.18).

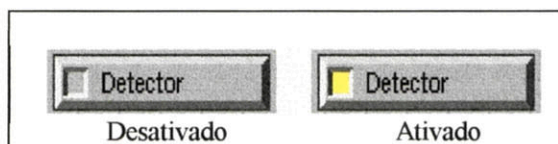


Figura 7.17 – Desativar e Ativar Função Detecção

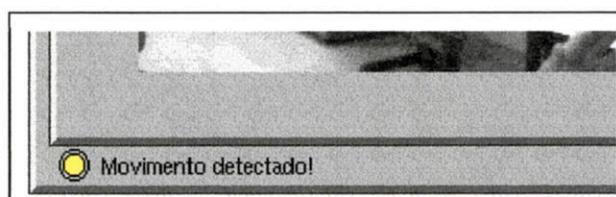


Figura 7.18 – Alarme ativado

Princípio de Funcionamento

Esse sistema é baseado na comparação de imagens do ambiente. A câmera capta uma imagem, inicialmente, armazenando-a. A segunda imagem é captada e comparada com a primeira. Caso ocorram variações das imagens, é ativado um alarme de segurança.

As imagens, como já explicado, são compostas por diferentes níveis de cinza. Para comparar duas imagens, é realizada a comparação pixel a pixel dessas duas imagens. Caso o valor absoluto entre os dois pixels das duas imagens for igual, significa que os pixels apresentam o mesmo nível de cinza. Caso a diferença dos valores absolutos seja superior a um valor predeterminado, esse pixel alterado é contabilizado em uma variável para este fim, denominada mudapixel.

Esse valor preestabelecido foi desenvolvido de modo a diminuir a sensibilidade entre os pixels comparados, otimizando o sistema. Deste modo, para a aquisição da imagem realizada com qualidade normal (4bpp), o valor do parâmetro de comparação entre os pixels fica estipulado em seis. Isto quer dizer, que se os dois pixels comparados possuírem uma diferença de seis níveis de cinza, os mesmos serão considerados diferentes, e então, é contabilizado na variável mudapixel.

Para o caso da qualidade da imagem ser Superior (6bpp), o valor do parâmetro de comparação entre os pixels fica estipulado em dez. Desta forma, se os dois pixels comparados possuírem uma diferença de níveis de cinza maior que dez, contabiliza-se na variável mudapixel.

A seguir, apresenta-se uma demonstração da comparação dos pixels:

Pixel 1 da imagem 1: valor de 20

Pixel 1 da imagem 2: valor de 40

Diferença dos valores: $20 - 40 = -20 \rightarrow$ valor absoluto = 20.

Como 20 é maior que 6 ou 10 (dependendo da qualidade da imagem), a variável mudapixel é acrescida em seu valor, por uma unidade.

Quando toda a tela é processada e comparada com a anterior, se o valor da variável mudapixel for maior que o valor pré-estipulado, é acionado o alarme do sistema, indicando a alteração da imagem e a possível presença de algum intruso no ambiente monitorado.

Caso este valor não ultrapasse ao determinado, o alarme não é acionado e a variável mudapixel é zerada, para o próximo processo de comparação de imagens.

Os testes e validação são realizados posicionando alguém, no caso o próprio desenvolvedor do projeto, em qualquer lugar a qualquer distância da câmera. Qualquer movimento realizado, o mínimo que seja feito, a câmera detecta a presença e ativa o alarme sonoro e luminoso.

7.3.9. Fator Detecção

Essa função possibilita o controle da sensibilidade apresentada para a função detecção. Dessa maneira, escolhe-se o valor que será comparado com a variável mudapixel, em cada término do processo de comparação de imagens.

Este botão, caso acionado, exibe uma interface auxiliar que apresenta cinco níveis de sensibilidade. O usuário poderá escolher o melhor nível de sensibilidade que convém para a detecção de movimentos no ambiente monitorado.

Essa função foi criada porque existirão várias câmeras em ambientes diferentes, cada uma possuindo sua característica de iluminação. Para ambientes onde a iluminação altera-se de modo mais abrupto que outros, estabelece-se uma

sensibilidade menor, mas que não compromete a detecção de qualquer movimentação não desejada no local.

Os cinco níveis de sensibilidade são representados pelos fatores 1 a 5 (figura 7.19). Após o usuário ter escolhido o fator que mais lhe é conveniente para a estação monitorada, aciona-se o botão **Ok**, onde a interface auxiliar é desabilitada, e no botão da interface principal, que corresponde ao fator de detecção, estará indicado o novo fator escolhido para o processo de detecção de movimentos.

Para realizar os testes, o mesmo princípio foi empregado na validação do sistema detector. Dependendo do fator de sensibilidade, movimentos muito sutis deixavam de ser detectados (com fator de sensibilidade 1 – menor sensibilidade). Isto permite o usuário adaptar da melhor forma possível o Sistema Detector para qualquer ambiente onde a câmera esteja posicionada.



Figura 7.19 – Caixa de Escolha da Sensibilidade e Botão de Ativação

7.3.10. Rastreador

Esse subsistema, anexo ao sistema de detecção, é capaz de detectar uma movimentação no local monitorado e, através do acionamento automático do motor remoto, instalado na base da câmera, localiza o movimento e posiciona a câmera em frente à alteração detectada no recinto.

Caso seja um intruso, o sistema irá movimentar a câmera de modo a seguir os movimentos do indivíduo, seguindo-o para a esquerda ou direita, tendo o mesmo

sempre centrado na imagem. A figura 7.20, logo abaixo, apresenta os possíveis estados em que pode se encontrar esse sistema.



Figura 7.20 – Estados do Rastreador

Princípio de Funcionamento

O funcionamento do Rastreador está baseado na representação da imagem em uma matriz, correspondente ao seu tamanho. Cada pixel receberá um valor que o localiza nessa matriz, por exemplo, o segundo pixel da primeira coluna será representado com um valor que o sistema saberá a sua localização (M21).

Junto com o sistema de detecção, os valores dos pixels, que por comparação forem diferentes, serão contabilizados na variável mudapixel, conforme explicado anteriormente, e sua localização também será contabilizada.

Para contagem da localização, para cada dimensão de tela, o sistema irá representar os dados da imagem de modo a serem representados e localizados em uma matriz.

Criada a matriz, determina-se a sua coluna central, dividindo a imagem em duas metades. Caso o pixel alterado esteja à esquerda da coluna central da matriz, contabiliza-se a mudança em uma variável, que corresponderá ao número de pixels alterados no lado esquerdo da matriz (coluna_esquerda). Para os pixels alterados que se localizam no lado direito da matriz, serão contabilizados na variável coluna_direita.

Ao final do processo de comparação de imagem, compara-se as duas variáveis coluna_esquerda e coluna_direita, e a que possuir um valor maior, aciona o motor para a esquerda ou direita, dependendo do resultado.

Para realizar esse processo, essas duas variáveis são comparadas por um valor de referência, pois caso fosse estabelecido apenas a pura diferença entre elas, o motor ficaria sendo acionado incessantemente, o que não é o desejado, pois o objeto detectado já estaria centrado na imagem.

Dessa forma, se a diferença for maior que um valor estipulado, como por exemplo, se a mudança de pixels à esquerda for maior em 40, que a mudança dos pixels do lado direito, é acionado o motor para a esquerda, e vice-versa.

Para cada ciclo de aquisição e comparação da imagem, o motor é acionado até que o objeto detectado esteja posicionado no centro da imagem adquirida.

Dessa forma, o sistema é capaz de detectar a presença de alguém, enviar um alarme para a central de controle, e além disso, seguir a movimentação do intruso, acionando o motor da base da câmera, posicionando a mesma na direção do objeto ou indivíduo detectado.

Uma observação a ser acrescida, é que a aplicação dessa função só estará habilitada a partir do momento em que a função detecção de movimentos seja acionada. Caso contrário, o botão que habilita a função Rastreador, na interface gráfica, estará desabilitado, não respondendo às tentativas de ativação do usuário. Para indicação da sua condição de desabilitação, o rótulo do botão encontra-se descolorido. Caso o botão de detecção é acionado, o sistema destrava o botão de Rastreador, que é indicado pela coloração do rótulo do botão.

Para realização dos testes, e consequente validação, o sistema emprega o mesmo princípio da realização de testes para o Sistema Detector. Uma pessoa se coloca na frente da câmera, a uma distância considerável, e desloca-se na direção paralela à câmera, em um sentido. A câmera começa a detectar a presença da pessoa no recinto, e com o Sistema Rastreador ativado, inicia-se o processo de rotação da câmera, seguindo o movimento. Se a pessoa parar de se mover, a câmera continua rotacionando até que a imagem do indivíduo esteja centrada na tela do dispositivo.

Invertendo o sentido de movimentação, a câmera também inverterá o sentido de rotação, continuando a rastrear a pessoa.

Movimentos rápidos demais, não são suportados pelo Sistema Rastreador, já que a velocidade de apresentação das imagens é relativamente baixa, impossibilitando a câmera de seguir os movimentos.

Apesar dessa limitação, o sistema é válido, apenas necessitando de um computador de maior capacidade de velocidade e uma câmera digital mais atualizada, com maior capacidade de transmissão.

7.3.11. Congelamento da Imagem

Através dessa função, é possível congelar a imagem instantaneamente. Esta função possibilita uma melhor escolha da imagem para a armazenagem da mesma em um arquivo do sistema (Foto). Além do congelamento da imagem, essa função desativa a maior parte das funções que se encontram na interface gráfica, sendo esta desabilitação indicada pela descoloração dos rótulos das funções.

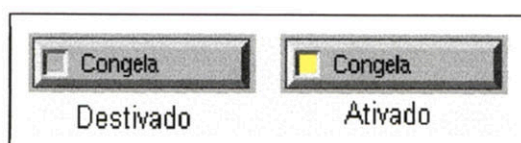


Figura 7.21 – Estados de Funcionamento da Função Congelamento

Este processo de desabilitação é realizado, pois caso as funções da interface, principalmente aquelas em que alteram a configuração da câmera, permaneçam habilitadas, a alteração dessas, com a imagem congelada, provoca um erro do sistema, ocasionando o fechamento do programa. Dessa forma, como medida de segurança, as funções são desabilitadas momentaneamente, até que o botão congela seja desativado.

Apenas algumas funções da interface não serão desabilitadas, como a caixa de ajuda do sistema e o botão de foto.

A figura 7.21 apresenta os modos de operação dessa função: ativada e desativada.

7.3.12. Armazenamento da Imagem

Essa função possibilita a armazenagem da imagem em um arquivo, que pode ser escrito em qualquer diretório do sistema operacional, com qualquer nome.

O seu formato foi determinado como *pgm (portable gray map)*. A figura 7.22 apresenta a ativação da função e o a caixa de escolha do nome que se deseja colocar para arquivamento, bem como o diretório correspondente.

7.3.13. Caixa de Ajuda: “Sobre o Sistema”

Esse botão abre uma caixa de ajuda, que explica todas as funções do sistema. Para sair dessa interface anexa, basta acionar o botão **Ok**. A figura 7.23 apresenta esta caixa de ajuda.

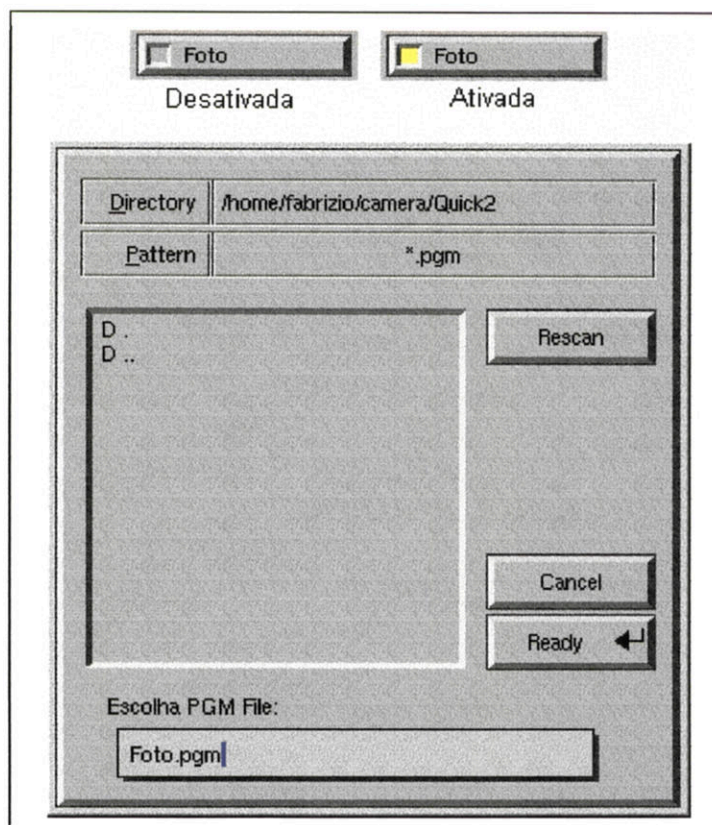


Figura 7.22 – Função Fotografia e Caixa de Escolha de Nome para Arquivo



Figura 7.23 – Caixa de Ajuda do Sistema

7.3.14. Bip do Alarme de Detecção

Existe um botão, na interface gráfica, que possibilita o usuário desligar o alarme sonoro de detecção de movimentos. Para isto, basta acionar o botão designado como “Som Lig.”. Quando acionado, o sistema sonoro do alarme é desativado, e o rótulo do botão é alterado para “Som Deslig.”. Dessa maneira, o alarme permanecerá em funcionamento normal, apenas que no caso de uma possível detecção, somente o sinal luminoso e o texto de aviso indicarão a detecção do movimento.

Acionando novamente essa função, ativa-se novamente o Bip do alarme, e ocorre a mudança no rótulo da função, “Som Lig.”, indicando que o som foi novamente ativado (figura 7.24).

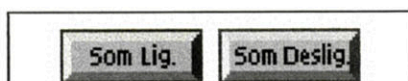


Figura 7.24 – Ativar e Desativar Bip do Alarme

7.3.15. Relógio

O Sistema possui um relógio para informação do usuário.

7.3.16. Velocidade da Imagem

Existe na programação do sistema, um algoritmo capaz de calcular a velocidade de apresentação das imagens na interface gráfica e converter essa informação em unidades de tela por segundo (tps – telas por segundo).

Um quadro localizado na interface gráfica é atualizado a cada ciclo de amostragem da imagem, sendo redimensionado com o novo valor de velocidade.

A figura 7.2 apresenta a interface gráfica por completo, e nesta, pode ser visualizado a função de amostragem da velocidade.

Princípio de Funcionamento

Esse sistema de cálculo da velocidade é realizado pela contagem do tempo entre o primeiro comando de captação dos dados da imagem da câmera, até o último comando, que realiza a inserção da imagem na interface gráfica. Para possibilitar a aquisição desse período, existe um comando na biblioteca do Linux que possibilita a cronometragem do tempo entre esses dois comandos.

A partir do tempo adquirido, realiza-se a transformação desse valor para apresentá-lo em unidades de tela por segundo, transformando o período adquirido em frequência de amostragem.

7.3.17. Saída do Sistema

Esta função, como o próprio nome já esclarece, realiza o fechamento do programa. Ao acionar esse botão, exibe-se uma caixa auxiliar, onde existe a pergunta de confirmação da saída de operação do sistema. Dentro dessa caixa, existe as duas opções de resposta (sim ou não). Caso a resposta “Não” seja a escolhida, a caixa auxiliar se fecha e o sistema continua em operação normal.

Se a resposta for positiva, o sistema inicia o processo de desligamento dos dispositivos envolvidos no sistema.

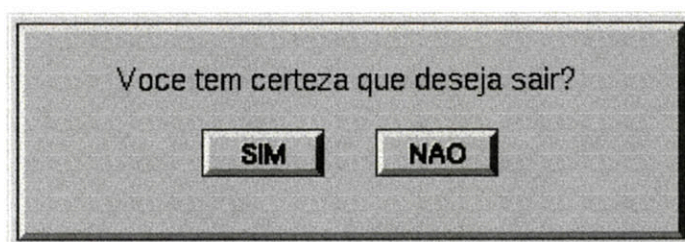


Figura 7.25 – Caixa de Saída do Sistema

Em primeiro lugar, o sistema identifica a posição atual em que se encontra a câmera. Se a mesma estiver fora de sua posição original, o sistema ativa, automaticamente, o motor remoto, de modo que este posicione a câmera na sua

marca original. Enquanto ele realiza essa operação, uma caixa de explicação é ativada, como mostra a figura 7.26.

Essa caixa, será ativada, se e somente se, a câmera estiver fora de sua posição original.

O retorno da câmera à sua posição normal, é conseguida graças à uma variável localizadora do sistema, que indica quantos passos o motor está para a direita ou para a esquerda. A posição original se encontra quando esta variável é zerada.

Após o sistema ter reposicionado a câmera em seu ponto original, desfaz-se a configuração da memória compartilhada, reseta-se a câmera e fecha-se a interface gráfica, ao final.

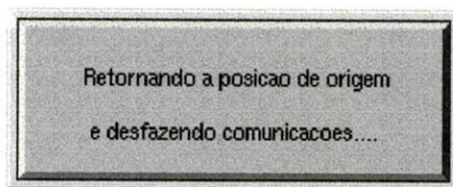


Figura 7.26 – Caixa de Indicação de Retorno da Câmera à Posição Original

7.4. Conclusões

O sistema apresentado foi desenvolvido visando a comunicação e a segurança das centrais telefônicas da planta da Telesc. Os testes realizados validaram todas as funções desenvolvidas para o sistema de segurança e para o sistema de imagens.

A única desvantagem encontrada para o sistema é a tecnologia disponível utilizada para a sua implementação. O computador onde o sistema foi desenvolvido, não possuía tecnologia de última geração, o que limitava a capacidade de velocidade de manipulação e processamento das imagens. A câmera digital, utilizada no projeto (única disponível), já se encontra defasada, em relação ao mercado, o que estabeleceu uma limitação no que diz respeito à capacidade de transmissão das imagens (velocidade).

Apesar dos imprevistos, o objetivo da implementação do sistema de comunicação visual e segurança foram concluídos com êxito e dentro do período estipulado pela programação.

CAPÍTULO 8

Conclusões e Perspectivas

Os sistemas de supervisão da Telesc são de grande importância para a manutenção dos serviços, oferecidos pela Empresa, para os milhares de clientes, distribuídos em todo o Estado de Santa Catarina.

Através do SIORE, a Telesc tem a capacidade de estabelecer uma comunicação entre as centrais telefônicas, a fim de promover um gerenciamento integrado do sistema, recebendo os sinais das centrais em um centro de gerência específico (CGIR). A integração e padronização da Tele-Supervisão é um passo importante para o aprimoramento dos serviços de controle de falhas, gerência e manutenção de todos os sistemas de sua planta.

Além dos serviços de manutenção dos equipamentos, o SIORE será capaz de estabelecer uma comunicação visual entre técnicos, seja qual for a distância entre o centro de gerência integrado e as centrais telefônicas. Esta comunicação, mais interativa e eficiente, é o primeiro passo para o desenvolvimento tecnológico na área de vídeo conferência da Telesc.

O sistema desenvolvido tem suas características voltadas também para a manutenção da segurança dessas centrais, possibilitando um controle ótimo dos ambientes monitorados pelas câmeras. As funções relacionadas no sistema, no que se refere à segurança, foram desenvolvidas com as técnicas adquiridas durante o desenvolvimento do trabalho, pelo próprio autor.

Os conhecimentos de informática, Engenharia de Software, eletrônica analógica e eletrônica digital, obtidos durante o Curso, foram de grande auxílio no desenvolvimento do driver do motor de passo, do circuito lógico controlador do motor e do software propriamente dito.

O sistema apresenta uma interface gráfica de fácil manuseio, onde as funções estão bem especificadas e com um lay-out que possibilita o usuário ativar ou desativar quaisquer funções, sem prejudicar o seu campo de visão, no que se refere à visualização das imagens.

O software desenvolvido, possui capacidade de controle de todas as funções inerentes à câmera, como opções para o tamanho da tela, resolução da imagem,

modos de transferência (unidirecional e bidirecional), configuração da imagem (brilho, contraste e balanço), aproximação da imagem (zoom), armazenamento de imagens na forma de fotografias, modo de congelamento da imagem, e outros mais, descritos no decorrer desta monografia.

As técnicas desenvolvidas para a segurança dos ambientes a serem monitorados, como sendo a *Detecção de Movimentos*, *Ajuste Automático de Brilho*, *Motor Remoto* e *Localização dos Movimentos*, são os primeiros passos para um sistema totalmente automático de controle da segurança.

Para tornar o sistema completamente automático, não necessitando de um operador humano, o próximo passo a ser desenvolvido, se refere à implementação de uma função capaz de adquirir e armazenar as imagens de vídeo, reproduzindo-as quando for requisitada. Trata-se de um sistema de gravação de vídeo, que junto com as funções referenciadas no parágrafo anterior, tornarão o sistema completamente automático, no que tange à segurança das centrais telefônicas de grande porte.

O programa desenvolvido, em linguagem C, necessitou da utilização de várias técnicas, de estruturação e orientada a objetos, de modo que fosse otimizado toda a sua estrutura, não havendo a necessidade de repetições de código, o que fatalmente, prejudicaria o desempenho do software de aquisição e tratamento de imagens.

Pelo fato do sistema operacional não possuir plataformas de desenvolvimento comercial tão amplas quanto de um sistema operacional Windows, as técnicas foram desenvolvidas com base em pesquisas realizadas pela Internet, onde utilizou-se ferramentas de domínio público (como o XForms, desenvolvimento da interface gráfica), e livros, que elucidavam a grande parte das dúvidas, que surgiam no decorrer deste trabalho.

O software, com certeza, sofrerá evoluções em sua versão inicial, inserindo novas funções no sistema, que têm o intuito de otimizar, cada vez mais, o objetivo pelo qual este foi criado.

Como fruto deste trabalho, a Telesc dispõe de uma ferramenta de comunicação visual e segurança que irá otimizar a comunicação estabelecida entre as centrais telefônicas e centros de gerência, melhorar a segurança dos ambientes a serem monitorados, e com certeza, abrirá novas oportunidades de desenvolvimento de aplicações a serem inseridas em sua planta.

Uma dessas futuras aplicações, com base no desenvolvimento deste trabalho, já pode ser visualizada em um futuro próximo, e se refere à parte de vídeo

conferência, que com absoluta certeza, não se limitará somente entre centrais telefônicas e centros de gerências, mas se estabelecerá uma rede de comunicação visual entre todos os setores da empresa, otimizando a comunicação e o desempenho da Telesc, através de uma sinergia entre os setores e o dinamismo que já faz parte de sua característica básica.

Bibliografia

- ADRIAN, Nye. **Xlib Programming Manual**. Volume 1, Third Edition: ISBN 1-56592-002-3, July , 1992.
- ADRIAN, Nye. **Xlib Reference Manual**. Volume 1, Third Edition: ISBN 1-56592-002-3, July , 1992.
- APOSTILA. **Básico de Sistemas de Telecomunicações**. Telecomunicações de Santa Catarina - TELESC S.A., Centro de Treinamento, 1996.
- DURBIN, James & SHAFER, John. **Connectix PC QuickCam Interface Specification**. Version 1.3 – 1996 Connectix Corporation.
- MIZRAHI, Victorine Viviane. **Treinamento em linguagem C**. São Paulo: McGraw-Hill, 1989.
- PUGH, Kenneth. **Programando em linguagem C**. Tradução José Renato Adorni Martins, Roberto Carlos Mayer, revisão técnica Roberto Carlos Mayer. São Paulo: McGraw-Hill: Newstec, 1990.
- RIBEIRO, Marcello. **Telecomunicações: Sistemas Analógicas – Digitais**. EMBRATEL, Livros Técnicos e Científicos, Editora S.A. 1980.
- SCHILDT, Herbert. **C Avançado: guia do usuário**. Tradução Cláudio Gaiger Silveira; revisão técnica Mônica Soares Rufino. São Paulo: McGraw-Hill, 1989.

- SEDRA, Adel S. **Microeletrônica**. Volume 1. Tradução Romeu Abdo; revisão técnica Antônio Pertence Júnior. São Paulo: Makron Books, 1995.
- SIEMENS, A.G. **Teoria do Tráfego Telefônico: Tabelas e Gráficos**. Volume 1. Editora Edgard Blucher Ltda. 1975.
- STUART Feldman. **Make - A Program for Maintaining Computer Programs**. Software - Practice and Experience, 9:255-265, 1979.
- TORO, Vicent Del. **Fundamentos de Máquinas Elétricas**. Tradução: Onofre de Andrade Martins. Practice Hall do Brasil.
- WELSH, Matt. **The Linux Bible: Linux Installation and Getting Started**. Linux Documentation Project, 1º edição, Agosto de 1993.
- ZHAO, T.C. & OVERMARS, M. **Forms Library – A Graphical User Interface Toolkit for X**. Copyright © 1995.